Magnetic Resonance Image Segmentation and its VHDL Implementation

by

Zheng Wei LI

A project

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Engineering

In the Department of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2005

Borrow List

Ryerson University requires the signatures of all persons using or photocopying this thesis.

Please sign below, and give address and date.

Name	Address	Signature	Date

Magnetic Resonance Image Segmentation and its VHDL Implementation

Zheng Wei LI
Master of Engineering
Electrical and Computer Engineering Department
Ryerson University 2005

Abstract

From experiments, it is shown that the Co-occurrence matrix for one still MRI brain image does not provide enough information for segmentation. The 6D Co-occurrence image segmentation idea for 3D MRI image is modified and implemented in 2D MRI image segmentation. That idea is to take two or three images as input at the same time and then process them with 3D Co-occurrence matrix. With this kind of processing a lot of information was brought into the Co-occurrence matrix, which is enough to segment the images. To compare the result, some other segmentation ideas were tested in this project. From the results, it can be seen that the MRI image segmentation based on the Co-occurrence texture analysis with two images or three images sampling is practical and the result is satisfying.

The segmentation is simulated in MATLAB. After the simulation, the segmentation is implemented in FPGA using VHDL. MODELSIM is used for FPGA functionality simulation. The result is close to the MATLAB simulation. This makes it possible to implement the system with FPGA hardware.

Keywords: MRI, image segmentation, Co-occurrence, texture analysis, FPGA, VHDL.

ACKNOWLEDGMENTS

I would like to bring my sincere thanks to my supervisor Prof. Lev Kirischian for helping me accomplish my studies in RYERSON. His broad knowledge and rich engineering experience gave me great help during my studies. I will memorize his kindness and professional engineering ethics forever.

I need thank all the faculty and staff of the Electrical and Computer Engineering Department. It is their hardworking attitude and innovative intelligence that lead this department to a better and better tomorrow.

The special thanks also need give to my wife and my lovely daughter for their complete support for my work and study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

FPGA------Field Programmable Gate Array

MRI ------Magnetic Resonance Image

2D ------2 Dimension

3D ---- --3 Dimension

RAM------Random Access Memory

SRAM----Static Random Access Memory

PC---------Personal Computer

DSP-------Digital Signal Processing

IIGGAD—Intensity(i), Intensity(k), Gradient magnitude(i), Gradient magnitude(k), Angle and Distance

IID--------- Intensity(i), Intensity(k), and Distance

L1 Distance----City Block Distance

IDM--------Inverse Difference Moment

MP---------Maximum Probability

CHAPTER I

PROJECT INTRODUCTION

1.1 Motivation

In recent years, the achievements in semiconductor have made it possible to implement some very complex algorithms into hardware. One of the greatest achievements is the Field Programmable Gate Array (FPGA) technology. Nowadays, there are more and more logic resources in FPGA, its lower price and higher capacity and especially its unique architecture has encompassed a lot of aspects of digital image and video processing. Its high speed has made it possible to do real-time image and video processing. The rapid prototyping character of FPGA also gives the engineers great help in changing the design flexibly. Its platform character also brings the flexibility to test multiple algorithms without wasting time on multiple hardware platforms.

For more than one century, scientists and engineers have been and continue trying to create the "intelligent machine". The computer is one of such machines. Digital Image Processing, Digital Audio Processing and Digital Video Processing are all the research subjects. At the same time the computer system engineers are trying to implement the newest algorithm with the most advanced technology. Among the targets the scientists and engineers are searching for, the visibility is one of the most important one. With visibility, the machine can recognize the object. The development of this kind of technology has been widely used everywhere, such as the vehicle plate recognition system, finger print recognition system.

In such systems, partial "machine vision" functions are implemented. In these systems, image segmentation is one of the very important steps for a machine to recognize the target. So it has always been an interesting and important subject for engineers to implement the new segmentation methods with the cutting edge technology. Driven by the same purpose, in this project, the Co-occurrence matrix based image segmentation is simulated in MATLAB and implemented in FPGA.

1.2 Objective of The Project and Approach

Image segmentation is an important and perhaps the most difficult low-level task [1]. Its application is widely spread into almost everywhere of image processing.

The target of this project is:

1. To create a new segmentation algorithm with the texture analysis technique and implement it on hardware platform (FPGA card) using VHDL Language. The application is on the Magnetic Resonance Image (MRI). The 2 Dimensions (2D) Co-occurrence matrix based image segmentation algorithm will be altered to fit the MRI image.

2. The algorithm will be verified by simulation in MATLAB. The result from MATLAB will be compared with some traditional ways of image segmentation.

3. The VHDL implementation and simulation for FPGA based platform. Because of the natural difference between computer based MATALB simulation and hardware based ModelSim simulation, the results were compared. The speed, cost is different between the computer based system and FPGA based system. VHDL is hardware description language, so it is hardware oriented, so a lot of functions such as division and logarithm in MATLAB need a lot of logic resources in FPGA. So the implementation in

FPGA is quite different with its counter part in MATLAB. Some implementation techniques will be emphasized in this project.

1.3 Original Contributions

Texture feature analysis has been extensively used in image analysis. The 2D texture analysis has been studied pretty well.

1. There are a lot of discussions about 3 Dimensions (3D) texture analysis also [8]. The 2D Co-occurrence matrix is useful to do texture analysis. The Co-occurrence matrix based image segmentation is quite effective. As it can be seen, the MRI image does not have too much texture information inside. But is it possible to do the segmentation with the texture analysis based Co-occurrence matrix? In this project, such a segmentation idea is presented and verified.

2. As it is recognized in the engineering field, there is always a distance between the theoretical result and practical performance. It means it is not always possible to implement the theoretical simulation with real hardware. So in this project, the Co-occurrence matrix based image segmentation is implemented in FPGA with VHDL and its result is compared with the MATLAB simulation result. Some ideas of implementation such as Co-occurrence matrix calculation, logarithm calculation are implemented, tested and presented.

3. In this project, MATLAB and MODELSIM FPGA simulation are connected by the design of test bench. This is useful for further studies and research.

1.4 Thesis Organization

This thesis has six chapters and two appendices. A brief introduction is held in the first chapter. The motivation, objective and approach will be presented in this chapter. The second chapter is theory overview and related works. Image segmentation theory and texture analysis and Co-occurrence matrix is discussed here. The general image processing system is introduced in this chapter also. The related works and some precedent studies and applications are in this chapter.

Chapter three is "Development of the Co-occurrence Matrix Based MRI Image Segmentation System". In this chapter, the whole process of the IID Co-occurrence Matrix based segmentation method is described. The simulation result from MATLAB is provided and analyzed.

Chapter four provides the FPGA implementation process of the segmentation method discussed in previous chapter. The comparison between FPGA and MATLAB based simulation is presented. Different hardware modules described by VHDL is also discussed in this chapter.

Chapter five is result comparison and analysis. The conclusion and future work will be addressed in this chapter also.

The MATLAB simulation program will be provided appendix A and VHDL simulation program will be provided in appendix B, C, D. Appendix E will provide the LUT for calculations.

CHAPTER II

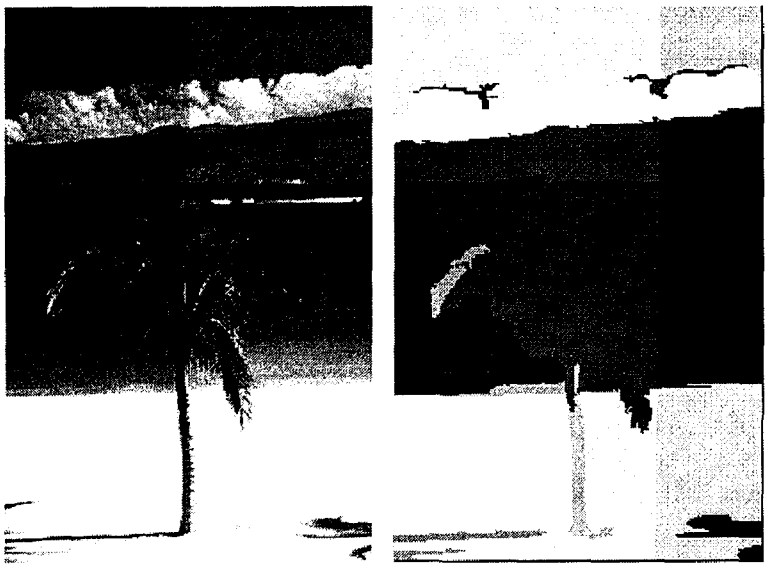
THEORY OVERVIEW AND RELATED WORKS

2.1 Image Segmentation and its Applications

As mentioned in the introduction part, image segmentation is a very difficult low level task. Image segmentation has been extensively used everywhere of image processing, such as image compression, image retrieval, object recognition. Basically the image segmentation process is defined as one that partitions a digital image into disjoint (non-overlapping) regions [2]. It refers to the grouping of image elements that exhibit "similar" characteristics [1]. Since seventies scientists have researched on the 2D image segmentation. Till now some mature and practical ideas have been developed and proved like the image segmentation by thresh-holding (global thresh-holding, adaptive thresh-holding, optimal threshold selection), gradient-based segmentation methods, edge detection and linking. All these ideas are belong to region approach, boundary approach and edge approach respectively. The region approach attempts to assign all the pixels to different particular object or region, like the histogram techniques. If the algorithm tries to locate the boundaries that exist between the different regions, it is called boundary approach, like the gradient image THRESHOLDING and LAPLACIAN Edge Detection. The edge approach is between the region approach and boundary approach. It tries to identify the pixels on the edge, it means the pixels between the different regions. After the pixels are identified, it will be linked to form the boundary. Recent years, a lot of new algorithms have been created, like the 3D Co-occurrence matrix based image segmentation relies on the texture pattern of the object, the segmentation using

genetic and hybrid search method [3] and the brain tissue segmentation based on Gegenbauer reconstruction pre-processing [4].

Figure 2-1 is one sample of image segmentation. 2-1(a) is the original image and 2-1(b) is the segmented image. This sample is segmented by the region approach.



(a) Original Image (14) (b) (Segmented Image (15)

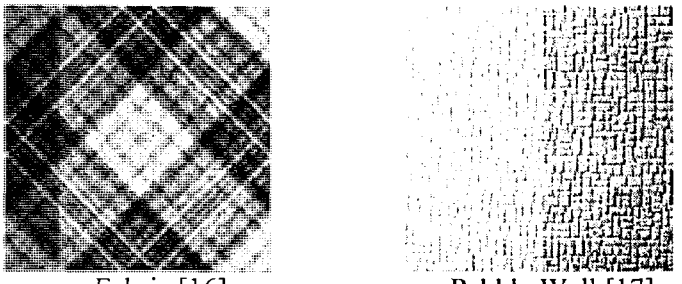
Figure 2-1 Sample of Image Segmentation

Because image segmentation is a low-level task, so its applications have been widely spread everywhere of image and video processing such as image coding and content-based retrieval [20], feature extraction, recognition and extraction.

In the real world, with the fast development of the micro-electronics technology, the application of the segmentation based image processing system is becoming deeper and wider. The vehicle plate recognition system can be seen on the high way and the apartment building. The high end industrial inspection system can be found in all kinds of industries.

2.2 Texture Analysis and Co-occurrence Matrix Based Texture Analysis

Texture analysis has drawn a lot of attention in the passing decades. With the different texture type and the specific image analysis problems, texture analysis can be found in a lot of applications. Figure 2-2 is some pictures with strong texture pattern in them. The picture on the left side is some kind of fabric and the picture on



Fabric [16] Pebble Wall [17]

Figure 2-2 Images with Strong Texture Pattern

the right side is an image of a pebble wall. Figure 2-3 is some pictures with less texture pattern in them. The picture on the left side is the research subject of this

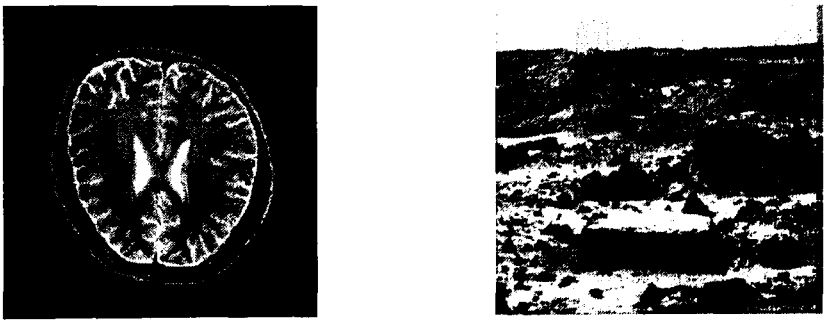


Figure 2-3 Images with Less Texture Pattern [18]

project, the MRI brain image. The picture on the right side is the picture of the Mars taken by the Mars Pathfinder. The tissue characteristic on the above four images are all different. So the ways to segment these images are different. For images with

strong texture information, the texture analysis is needed for segmentation. Among the different texture analysis techniques, Co-occurrence matrix is a special one. Haralick et al. [5] and Galloway [6] suggested the gray level Co-occurrence matrix and run length matrix and they have been extensively used in texture classification and segmentation. The successful synthesis of textures motivates the use of Co-occurrence as features for texture classification [9].

Table 2-1 is one sample of the Co-occurrence Matrix. Table 2-1(a) is a 4x4 image matrix, the gray level is from 0 to 3. Table 2-1(b) is the Co-occurrence Matrix of 2-1(a) with distance 1 and direction 0 degree. Each element in Table 2-1 (b) shows the occurrence rate of the gray-level at distance 1 and direction 0. For example, the "2" (shaded) means that it happens twice from gray-level "1" to gray-level "2", which is exactly what happened in Table 2-1 (a).

Table 2-1 (a) 4x4 Image Matrix

1	1	2	2
3	3	3	0
1	2	3	3
2	1	0	1

Table 2-1 (b) Sample of Co-occurrence Matrix

0	1	0	0
1	1	2	1
0	1	1	1
1	0	0	3

As it can be seen from Table 2-1, the Co-occurrence Matrix is to identify the occurrence rate of each gray level at certain distance and direction. Obviously it is a very good tool for texture analysis.

2.3 General Image Processing System

Most of the image processing system has some common parts like image capture, image segmentation and feature extraction, classification and recognition.

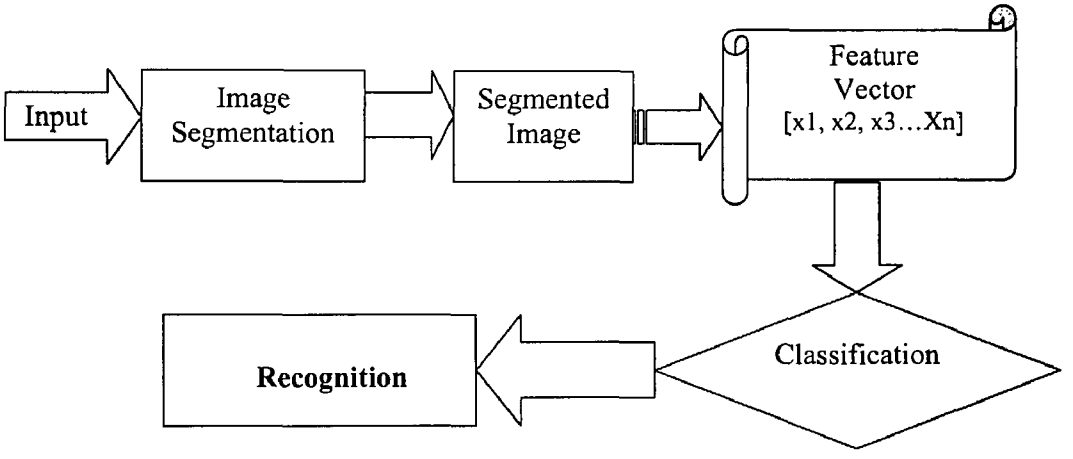


Figure 2- 4 General Pattern Recognition System

A general image processing system, or more specifically the pattern recognition system was presented in Figure 2-4. As we can see from the above model, image segmentation process plays a fundamental role in all the subsequent interpretation process, like image retrieval, pattern recognition, and feature analysis.

The image processing system can be divided into two categories. The first category is the PC based image processing system, which will process the image with software. The second category is the hardware based system. Normally the hardware based

system is better in cost, processing speed and size. In this project, the MATLAB simulation can be considered as the PC based image processing system and the VHDL implementation is developed towards the direction of hardware based system.

2.4 Related Works

As it is mentioned in previous chapters, image segmentation is very important. So this subject has been extensively studied. A lot of traditional ways of image segmentation are based on gray-level thresh-holding, image gradient or edge detection and linking. A lot of new algorithms are appearing and seem to be effective, like the "Adaptive Image Segmentation Based on Color and Texture" [11] proposed by Jinqing Chen from ECE department of Northwestern University and Aleksandra Mojsilovic from IBM research centre. The adaptive algorithms are gradually taking more places nowadays. Bir Bhanu from University of California presented the adaptive genetic way of segmentation in "Adaptive Image Segmentation Using Genetic and Hybrid Search Methods" [3].

Haralick et al. [5] and Galloway [6] suggested the gray level Co-occurrence matrix and run length matrix and they have been extensively used in texture classification and segmentation. Gabriele Lohmann used the Co-occurrence matrix as the tool of texture analysis and synthesis in "Analysis and synthesis of Textures: A Co-occurrence-Based Approach". Vassili etc presenting a new method of MRI texture analysis using Co-occurrence matrix. That method is based on extended, multi-sort Co-occurrence matrices that combine intensity, gradient and anisotropy image features in a systematic and consistent way. This method has pushed the application of Co-occurrence matrix to a new

high point. That also means the hardware realization of the algorithm will be too expensive, that makes it impossible to do it with current technology.

There are different ways to implement the segmentation algorithms. Since the computer was invented, people have been using it as the implementation tools, especially now days when computers are becoming more and more powerful. The algorithm can be programmed by different languages such as C++, Basic, Java or MATLAB. But no matter which program to use, there is always a computer, which is too big in size and too expensive in cost and also comparably slow in speed. Because this type of implementation always needs to be programmed, so it is called software based.

So when real-time processing is in need, the hardware based implementation is always the first choice. The hardware based systems are also different by the hardware it uses. For some small system, the micro-processor based embedded system is normally used. Before the Field Programmable Gate Array (FPGA) technology became powerful enough, the Digital Signal Processing (DSP) based system was very popular. One sample of this system is developed by V.Gemignani etc. in "Real-Time Implementation of A new Contour Tracking Procedure in a Multi-Processor System" [12]. With the development of the FPGA technology towards the direction of larger capacity, faster speed, much more flexible configuration, the FPGA based system is becoming more and more popular. Some system is based on DSP+FPGA combination. A lot of systems are based on FPGA only. Mr. Steffen Klupsch presented a typical system in [13]. A very important feature of FPGA is so called "rapid prototyping". The engineers can have the hardware prototype in hand at the beginning stage of the project, but there still leaves a lot more space for them to change the

design or even the methodology without changing the hardware, because that type of job can be done in VHDL.

Summary: Brief description of image processing and segmentation theory. The general image processing system is discussed. The implementation with PC and hardware were compared in this chapter. The related works and precedent studies were presented in this chapter.

CHAPTER III

DEVELOPMENT OF THE CO-OCCURRENCE MATRIX BASED IMAGE SEGMENTATION SYSTEM

Introduction:

This project is trying to find a way to segment the 2D MRI image with Co-Occurrence matrix. As it is known the Co-occurrence matrix has been widely used in image texture analysis, but the application in image segmentation be can rarely seen. One simple reason is that there is no obvious texture structure in MRI brain image (this is proved through experiment in this project). From the paper of Vassili A. Lovalev *al.* [8], some innovative ideas for the Co-occurrence matrix's application in the 3D MRI image is brought up. Could the same idea be used on the 2D image? How and what to modify and then improve it to fit the 2D MRI brain image segmentation? Is the result acceptable? In this project all these questions were answered.

3.1 Algorithm

As discussed in previous chapter, the six dimensional Co-occurrence matrix Segmentation has to be modified to fit the 2D image. This section will focus on this.

3.11 Six-Dimensional Co-occurrence Matrix

If given a direction and a distance (one pixel, two pixels, etc) in an image. Then the element E_{ij} of the Co-occurrence matrix **P** for an object is the number of times, divided

by **M**, that gray levels *i* and *j* occur in two pixels separated by that distance and direction in the object, where **M** is the number of pixel pairs contributing to **P**. The matrix **P** is **(NxN)**, where the gray scale has **N** shades of gray. Once there is the Co-occurrence matrix, texture features can be computed from it.

The author of [8] proposed a new 3D texture analysis of magnetic resonance imaging brain datasets. It is based on extended, multi-sort Co-occurrence matrices that employ intensity, gradient and anisotropy image features in a uniform way [8]. This idea does not change the basic idea of Co-occurrence, but increase the sensitivity and specificity of Co-occurrence descriptors. And the new extended 3D Co-occurrence idea has the rotation and reflection invariance, which is very important to the texture analysis of the MRI image. According to this article, if we have an arbitrary voxel (voxel is the element of the 3D image) pair *(i, k)* defined on discrete voxel lattice by voxel indexes $i = (x_i, y_i, z_i)$, $k = (x_k, y_k, z_k)$ and with the Euclidean distance $d(i,k)$. The intensities of these voxels are $I(i)$ and $I(k)$, local gradient magnitudes by $G(i)$, $G(k)$, and the angle between their 3-D gradient vectors by $a(i,k)$. Then the general, six-dimensional (6D) Co-occurrence matrix is defined as [8]

$$W = \left\| w(I(i), I(k), G(i), G(k), a(i,k), d(i,k) \right\| \tag{1}$$

$$G(i) = \sqrt{G_x^2(i) + G_y^2(i) + G_z^2(i)} \tag{2}$$

$$a(i,k) = \cos^{-1}(g(i) \bullet g(k)) \tag{3}$$

Equation (1) is consisted of "Intensity (I), Intensity (k), Gradient (I), Gradient (K), Angle (I,K) and Distance (I, K). So this kind of Co-occurrence matrix like (1) is called IIGGAD Matrix [8], it is a 6D matrix.

3.1.2 Three Dimensional Co-occurrence Matrix

From IIGGAD, some reduced versions of the above IIGGAD matrix, such as intensity (IID), gradient magnitude (GGD) and gradient angle (GAD) matrices can be derived [8]. As the author mentioned, the IID version of the IIGGAD matrix is the simple 3-D version of the traditional Co-occurrence matrix. And in this project, this version will be used to segment the 2-D MRI image. There are several mature texture features based on this IID Co –Occurrence [8]. (P is the Co-occurrence matrix element)

ENTROPY:
$$H = \sum_{i=1}^{N}\sum_{j=1}^{N} P_{ij} \log P_{ij} \tag{4}$$

INERTIA:
$$I = \sum_{i=1}^{N}\sum_{j=1}^{N} (i-j)^2 P_{ij} \tag{5}$$

ENERGY:
$$E = \sum_{i=1}^{N}\sum_{j=1}^{N} \left|P_{ij}\right|^2 \tag{6}$$

CONTRAST:
$$C = \sum_{i=1}^{N}\sum_{j=1}^{N} \left|i-j\right| P_{ij}^{\ 2} \tag{7}$$

INVERSE DIFFERENCE MOMENT (IDM):

$$IDM = \sum_{\substack{i=1 \\ i\neq j}}^{N}\sum_{\substack{j=1 \\ i\neq j}}^{N} \frac{P_{ij}^2}{|i-j|} \tag{8}$$

MAXIMUM PROBABILITY

$$MP = MAX(P_{ij}) \tag{9}$$

CORRELATION:

$$r = \frac{N\sum xy - (\sum x)(\sum y)}{\sqrt{[N\sum x^2 - (\sum x)^2][N\sum y^2 - (\sum y)^2]}} \tag{10}$$

where x and y are elements of the Co-occurrence matrix.

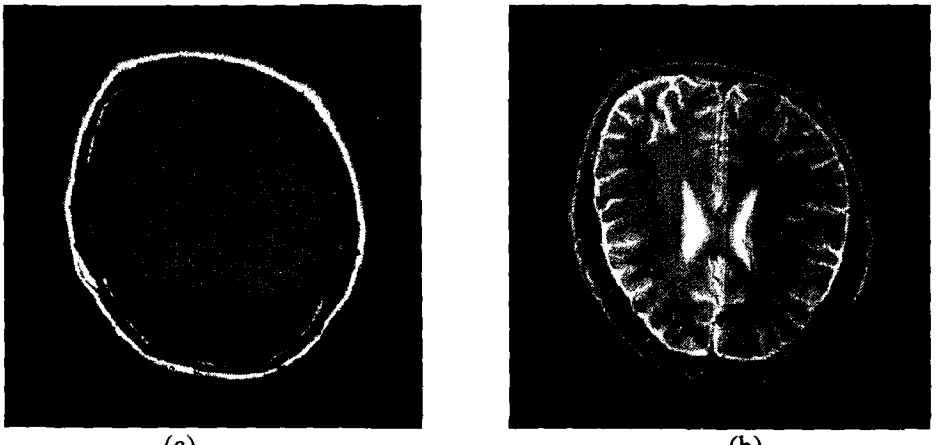
According to [8], L1 distance (City Block Distance) is needed to classify ROI (region of interest) or AOI (area of interest) and the background. "sample" is for ROI and "control" is for background in this project. L1 distance is defined as follow:

$$L1(V^m, V^n) = \frac{\sum_{i=1}^{N} \left\| x_i^m - x_i^n \right\|}{\sum_{i=1}^{N} x_i^m + \sum_{i=1}^{N} x_i^n} \tag{11}$$

V^m, V^n are vectors of Co-occurrence feature descriptor for samples and controls, they are made up by the features like ENTROPY, ENERGY, CONTRAST, INVERSE DIFFERENCE MOMENT (IDM) and MAXIMUM PROBABILITY (MP).

3.1.3 The City Block Distance Histogram of Samples and Controls of Image

The target for this project is to segment the different tissues in the MRI images based on their different texture character. In order to have a more straightforward idea about this project, here is the brief introduction of the original image used in the test. Figure 3-1(a)



(a) (b)

Figure 3-1 MRI Images
(Courtesy of the Biomedical Engineering Research Group of Ryerson University)

and Figure 3-1(b) are both 256x256 MRI image. Both images have 256 gray levels. Figure 3-7 is another image with 256 gray levels. The purpose of getting different sizes and different types of images is to check if there is any difference in the segmentation results.

At the beginning of this project, one of the tissues in the image was used as sample and the other as controls (background). Nine samples were taken from the ROI and

Figure 3-2 Samples of Material a

Figure 3-3 Samples of Material b

Figure 3-4 Samples of Material c

also from backgrounds. Figure 3-2, 3-3 and 3-4 are the samples of materials a, b and c respectively. The samples are taken by 4x4 block, so there are 16 pixels in one sample. The images shown in Figure 3-2, 3-3 and 3-4 are enlarged 7.5 times to be visible. All the sample images have to be uniformed before the calculation of Co-occurrence matrix. As it is mentioned before, the raw image has 256 gray-levels. If the image is not uniformed, there won't be any useful information in the Co-occurrence Matrix. Table 3-1 shows a 4x4

Table 3-1 4x4 Sample Block

180	197	90	27
27	250	125	94
94	29	34	32
67	78	56	67

image block. The Co-occurrence matrix of this block will be a 256x256 matrix, most of its elements will be "0" except at location (180, 197), (197, 90), (90, 27), (250, 125), (125, 94), (94, 29), (29, 34), (34, 32), (67, 78), (78, 56), (56, 67), there is a "1" at these locations. But even these "1"s are so sparse, there won't be enough information for processing. If the 4x4 block is uniformed by reducing the gray-level to 4:

$$0—63\rightarrow 0,\ 64—127\rightarrow 1,\ 128—191\rightarrow 2,\ 192—255\rightarrow 3$$

the uniformed image shows in Table 3-2.

Table 3-2 Uniformed Image Block

3	3	1	0
0	3	1	2
1	0	0	0
0	0	0	0

Table 3-3 is the Co-occurrence Matrix of Table 3-2 at distance "1" and direction "0" degree.

Table 3-3 The Co-occurrence Matrix of Table 3-2

5	0	0	1
2	0	1	0
0	0	0	0
0	2	0	1

The Co-occurrence matrix of the reduced image has more useful information. From the samples shown above in Figure 3-2, 3-3 and 3-4, the Co-occurrence matrix of each sample block can be calculated after reducing them into 4 gray-levels. Then the values of the corresponding elements of the 9 Co-occurrence matrixes are added together, then

divided by 9, the average Co-occurrence matrix for all sample ROI can be calculated as shown in Table 3-4, 3-5, 3-6, for material a, b and c respectively.

Table 3-4 Average Co-occurrence Matrix of Sample a

3.0000	0.6667	0.7778	0.2222
1.2222	0.7778	0.2222	0.2222
1.1111	0.7778	0.1111	0.3333
0.5556	0.5556	1.1111	0.3333

Table 3-5 Average Co-occurrence Matrix of Sample b

1.7778	0.7778	0.3333	0.5556
0.5556	1.0000	0.7778	0.4444
0.6667	0.5556	1.2222	0.8889
0.1111	0.4444	0.5556	1.3333

Table 3-6 Average Co-occurrence Matrix of Sample c

2.2222	0.7778	0.3333	0
1.1111	0.6667	1.0000	0.2222
0	1.0000	1.1111	0.8889
0.2222	0.4444	0.7778	1.2222

Table 3-4 is the average Co-occurrence matrix of material a, Table 3-5 is the average Co-occurrence matrix of material b and Table 3-6 is the average Co-occurrence matrix of material c. Each ROI's feature vector can be calculated from the average Co-occurrence matrix. Each feature vector consists of "Entropy", "Energy", etc, defined by equation (4) to (10) respectively. Table 3-7, 3-8, 3-9 is the feature vector list of material a, b and c.

Table 3-7 Feature Vector of Material a

Sample Quantity	9
Entropy	0.4129
Energy	16.2222
Contrast	22
Inverse Difference Moment	5.0864
Max Probability	3

Table 3-8 Feature Vector of Material b

Sample Quantity	9
Entropy	-2.5270
Energy	11.6296
Contrast	17.6667
Inverse Difference Moment	4.6574
Max Probability	1.7778

Table 3-9 Feature Vector of Material c

Sample Quantity	9
Entropy	0.1339
Energy	13.7531
Contrast	11.5556
Inverse Difference Moment	5.8302
Max Probability	2.2222

So L1 distance can be calculated from the ROI feature vectors verses the ROI samples' average feature vectors. For the samples of the background, the same thing can be done: get each background sample's feature vector and calculate the L1 distance with these feature vectors verses the background samples' average feature vector.

To test the effectiveness of L1 distance, 18 L1 distances were calculated. The histogram of the L1 distance can be drawn. Figure 3-5 is the L1 histogram of Figure 3-1(b) The Y axis is the L1 distance value. On the X axis, 1 to 9 is the L1 distance of the background samples versus the ROI samples' average feature vectors, 10 to 18 is the L1 distance of the background samples versus background samples' average feature vectors. From this histogram, it can be seen there is very strong difference between the L1 distance

of the ROI and background. That means the segmentation can be done based on the Co-occurrence matrix.

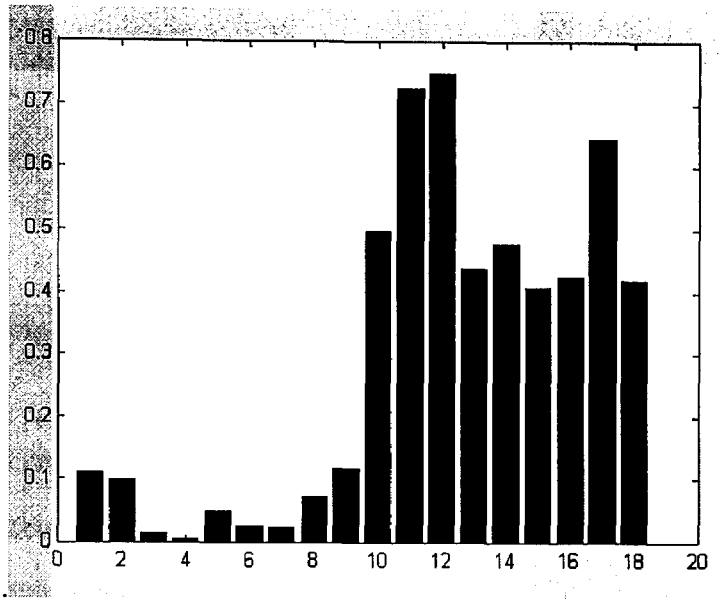


Figure 3-5—the L1 histogram of samples and controls for image

3.1.4 L1 Distance Mapped Image

From previous section, it has approved that L1 distance is effective to do the segmentation. Based on that, a 5x5 window is running through the image. Each window's L1 distance is calculated (each window's feature vector verses the average ROI feature vector). Different gray levels were assigned to the different L1 distance in the new image. The gray level based L1 probability map image is shown as below. Figure 3-6 is the L1 distance mapped image of Figure 3-1(a). Figure 3-8 is the L1 mapped image of Figure 3-7. Comparing with the original images, these images show not only the L1 distance, but also the segmentation information. The segmentation seems to be well done, but obviously this can not be used as the final segmentation result. .

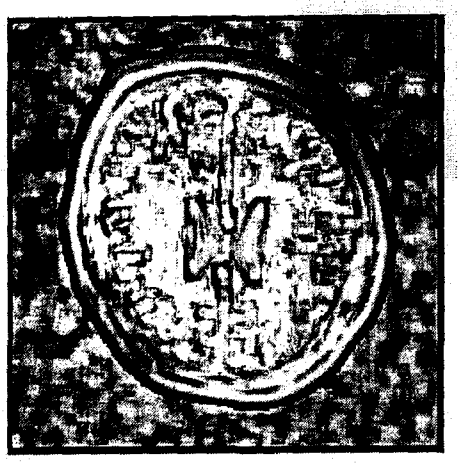


Figure 3-6 L1 distance mapped image of 3-1(a)

In Figure 3-6 and Figure 3-7, the L1 distance mapped image of image 3-6 and 3-8 shows clearly the edges of the different tissues. But this is not the aim of segmentation. For

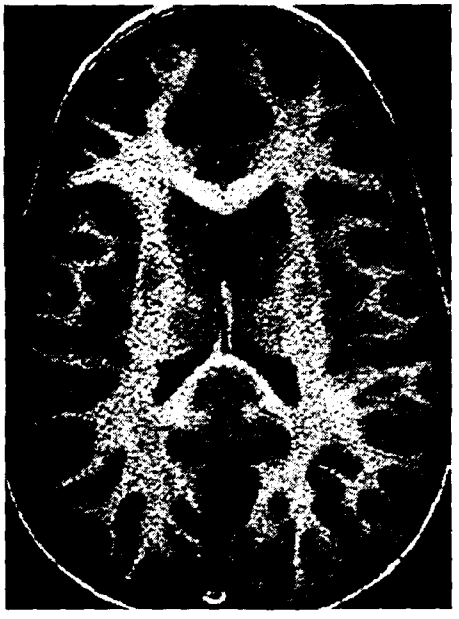


Figure 3-7
RAW Image

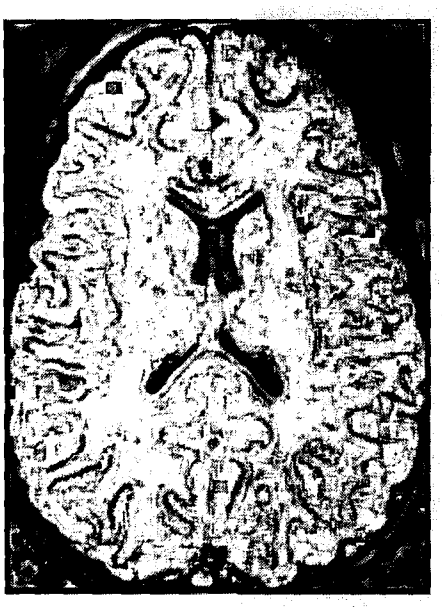


Figure 3-8
L1 Distance Mapped Image of Figure 3-7

segmentation, different materials need to be separated clearly with different colors or different gray scales to identify different parts in the image. But for the L1 distance, it is

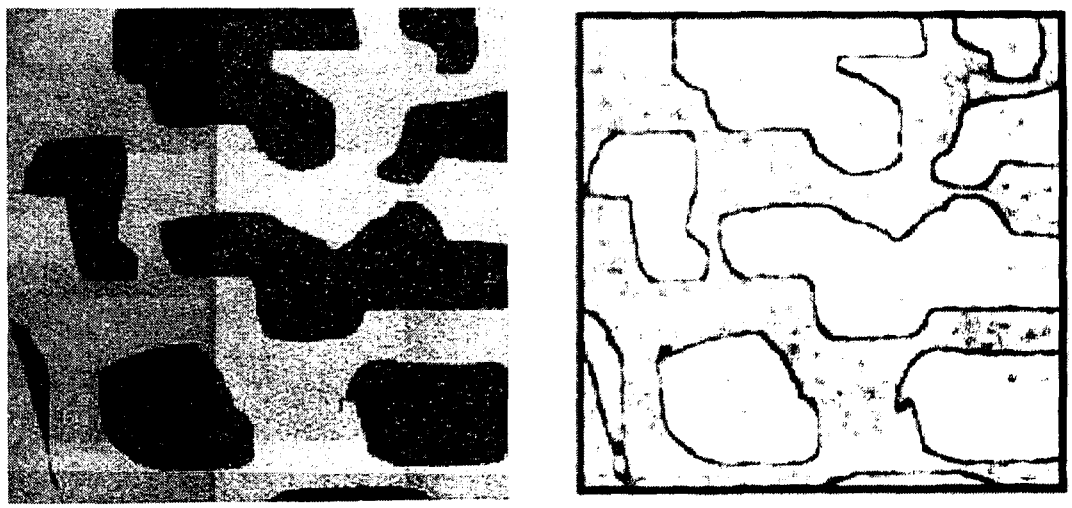
very hard to set the different threshold. L1 distance mapped image might be used for edge detecting, but it needs more research and studies. So far, it proves L1 distance is a very good feature for segmentation, but how to use it becomes a new problem.

3.2 Applications on Different Materials

It seems if the threshold of the L1 distance can be found, the segmentation problem can be easily solved.

3.2.1 The Application on Material with Strong Texture Pattern

Co-occurrence matrix well describes the occurrence frequency of the different gray-levels at certain distance and direction, so it should have good segmentation result on images with strong texture. To prove this, one example is given here. As shown in Figure 3-9(a), it is an image with very strong texture information inside. Two types of samples are



(a) (b)

Figure 3-9 Application on Image with Strong Texture

taken from the original image, from these samples, the feature vectors are calculated. Then a 4x4 window is running through the image, the feature vector of each block will be used to calculate the L1 distance against the two different textures. The smaller L1 distance means the current block belongs to that texture with which it is calculating the L1 distance. Figure 3-9(b) is the segmented image, the result is satisfactory. So the next step is to apply this technique on the images without strong texture information.

3.2.2 The Application on Material without Strong Texture Pattern

Section 3.2.1 proves that the algorithm to be effective on the segmentation for images with strong texture. But as it is so obvious for us, the MRI brain image datasets do not have very strong texture information inside. What is the result will be if same idea is applied to such images? So for images in Figure 3-1, instead of taking only two kinds of samples, three kinds of samples are taken from the original image.

Figure 3-10 Segmented Result Based on One Input Image

Three average feature vectors for these three kinds of samples need to be calculated. And then the running window will go through the image. For each window, first to calculate the L1 distance verses all three samples to get three L1 distance for each window (or the small image block). Second to look for the least L1 for this window, the least L1 means this window is very close to that kind of sample. Then assign different colors or gray levels to this kind of tissues. After finishing running the whole image, a new segmented image is stocked in a matrix. This image will only have three different colors or three different gray levels. So the purpose of segmentation is achieved. According to above ideas, segmentation is done in the new programs. The segmentation result is shown in Figure 3-10. Unfortunately, the result is not acceptable. If take a look back at the sampling on the original image, it can be found out that when the samples are taken from one kind of material, there are almost no useful information in the Co-occurrence matrix of the samples. The histogram of theses samples L1 distance verses the average feature vectors proves this. From the original concept of Co-occurrence, it is known that it is obvious that the Co-occurrence for this kind of material is so random and sparse that can rarely provide any useful information. The different tissues taken from the MRI brain image are almost uniform, that means there is not much texture information in it. It will be discussed it in the next section.

3.3 The Specific Approach for MRI Brain Image

Something has to be changed in order to segment the MRI images successfully with the IID Co-occurrence matrix. After some trial and tests were done, the solution is found.

3.3.1 The Result Based On Two Input Images

From the medical knowledge of the MRI brain images, it is known that Figure 3-1(a) and Figure 3-1(b) are the same images but with different features shown up. With the different feature shown up, these two images have different gray-level at the same spot. That is what the Co-occurrence matrix needs to do the segmentation. This gives a chance to use the Co-occurrence matrix. So this time three kinds of samples will still be taken. A 4x4 sample block is taken from image a, and another 4x4 sample from image b, these two 4x4 block form a 4x8 sample block. Nine such samples for each kind of materials will be taken in total. Figure 3-10 is the samples taken from two images. The average Co-occurrence matrix can be calculated. From the average Co-occurrence matrix, the average

Figure 3-11 Samples Taken From Two Input Images

feature vector of the three materials will be calculated. The average Co-occurrence matrix is shown in Table 3-10, 3-11, 3-12.

Table 3-10 Average Co-occurrence Matrix of Sample a (Figure 3-11-a)

19.8889	0.5556	0.8889	3.6667
0.1111	0.5556	0.5556	0.2222
0	0.3333	1.4444	1.3333
0	0	1.1111	14.3333

Table 3-11 Average Co-occurrence Matrix of Sample b (Figure 3-11-b)

20.0000	0.5556	2.0000	2.4444
0	1.1111	0.5556	0
0	0.4444	5.7778	1.4444
0	0	1.6667	9.0000

Table 3-12 Average Co-occurrence Matrix of Sample c (Figure 3-11-c)

5.3333	2.2000	0.7333	0.1333
1.8000	5.9333	2.8667	0.6667
0.7333	3.2667	7.9333	2.4667
0.4000	1.0667	2.2667	7.2000

After all the feature vectors were got, a 4x8 running window will be used to sample the original images. The L1 distance will be calculated verses the three average feature vectors. Once the least L1 is found, the current sample block belongs to that kind of material that calculated the L1 distance with. The current block is labeled with different colors or different gray-level into a new image matrix. The segmentation would be done once finish running through the whole image. The average feature vector is shown in Table 3-13, 3-14 and 3-15.

Table 3-13 Feature Vector of Material a (Figure 3-11-a)

Sample Quantity	9
Entropy	146.2917
Energy	621.4444
Contrast	41.4444
Inverse Difference Moment	4.6852
Max Probability	19.8889

Table 3-14 Feature Vector of Material b (Figure 3-11-b)

Sample Quantity	9
Entropy	135.4420
Energy	531.2716
Contrast	34.6667
Inverse Difference Moment	5.4383
Max Probability	20

Table 3-15 Feature Vector of Material c (Figure 3-11-c)

Sample Quantity	9
Entropy	90.3208
Energy	219.4533
Contrast	32.4667
Inverse Difference Moment	15.7259
Max Probability	7.9333

Figure 3-12 and Figure 3-13 are the segmented images of Figure 3-1a (or 3-1b). Figure 3-12 is shown with different gray scales and Figure 3-13 is shown as different colors. It can be seen that the image is very well segmented according to their texture structure. Because only 3 samples are taken from the original image and no sample were taken from the black background, so it seems a messy in the area around the brain MRI. But that does not affect our aim of segmentation the MRI brain image.

Figure 3-12 (a) The Segmented Image With Different Gray Levels

Figure 3-12 (b) The Segmented Image With Different Colors

The above results are based on the original image from 3-1(a) and 3-1(b). If the two input images are exchanged. The segmentation result is like Figure 3-13. Please notice that the edge of the image is processed, so it looks better.

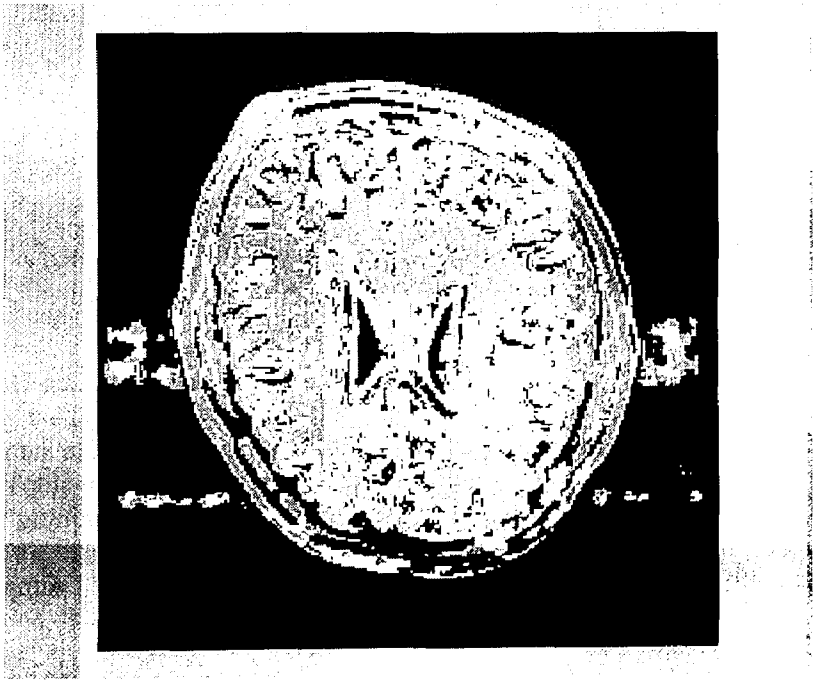


Figure 3-13 The Result Based On Two Input Images

3.3.2 The Feature Vector Analysis

As shown in Figure 3-11. 3-12 and 3-13, the segmented results are acceptable but there is still space for improvements. It seems that the sampling idea used to construct some texture is working well. So if continue to do this with more images, the result should be much better. Another original image is added in the sampling. This time, a 5x5 sampling block is used instead of 4x4 block. By doing this, only the center point of each block was set to the level it belongs to. So the final segmented image will be much smoother. The third image is shown in Figure 3-14.

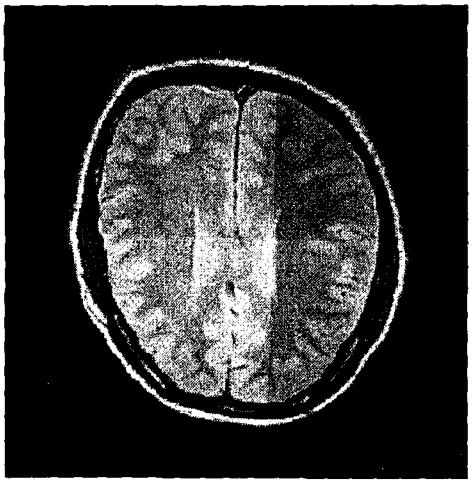


Figure 3-14 The Third Original Image
(Courtesy of the Biomedical Engineering Research Group of Ryerson University)

Same as the two input images, for each kind of material, nine samples are taken from 3 original images at the same location. Because it is time consuming to take all the samples manually, so the sample capture program is useful here. The 3 samples are shown as Figure 3-15, 3-16 and 3-17

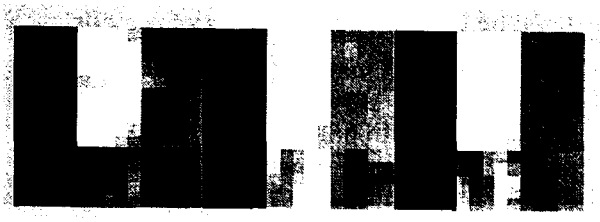


Figure 3-15 Sample of Material a From 3 Images

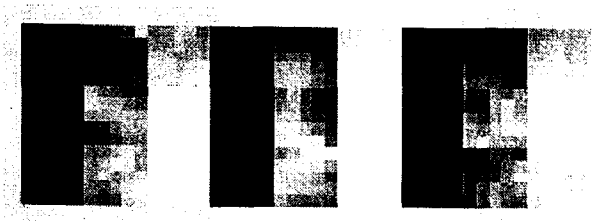


Figure 3-16 Sample of Material b From 3 Images

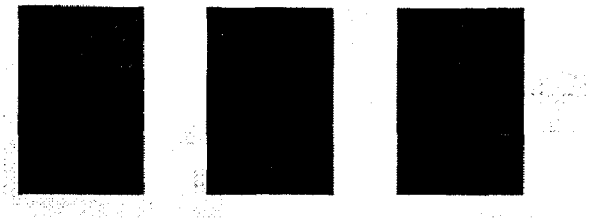


Figure 3-17 Sample of Material c From 3 Images

From this three samples, the average Co-occurrence matrix are shown in tables below.

Table 3-16 Average Co-occurrence Matrix of Material a (Figure 3-15)

19.8889	0.5556	0.8889	3.6667
0.1111	1.6667	1.6667	0.4444
0	1.8889	18.0000	1.6667
0	0	5.2222	14.3333

Table 3-17 Average Co-occurrence Matrix of Material b (Figure 3-16)

20.3333	2.0000	3.0000	0
0	4.2222	1.5556	1.5556
0	1.1111	11.6667	3.6667
0	0	0.4444	20.4444

Table 3-18 Average Co-occurrence Matrix of Material c (Figure 3-17)

34.3333	3.0000	0	4.2222
3.0000	4.6667	0	0.7778
0	0	0	0.1111
0	0	0.1111	19.7778

Using the above average Co-occurrence matrix, the feature vectors are in Table 3-19,

Table 3-19 Feature Vector of Material a (Figure 3-15)

Sample Quantity	9
Entropy	239.1657
Energy	978.9383
Contrast	49.4444
Inverse Difference Moment	11.8519
Max Probability	19.8889

Table 3-20 Feature Vector of Material b (Figure 3-16)

Sample Quantity	9
Entropy	242.7560
Energy	1.0181e+003
Contrast	27
Inverse Difference Moment	9.9167
Max Probability	20.4444

Table 3-21 Feature Vector of Material c (Figure 3-17)

Sample Quantity	9
Entropy	287.9804
Energy	1.6282e+003
Contrast	47.3333
Inverse Difference Moment	6.8858
Max Probability	34.3333

3-20 and 3-21. Figure 3-18-A is the feature vectors of three different materials. Figure 3-18-B is the plotting of 5 features of the 3 different samples. On the X axis, 1-9 is sample1, 9 to 18 is sample 2, 19 to 28 is sample 3. It is obvious that the 3 images based feature vectors are very effective The L1 distance is the combination of all the features. The image is segmented based on L1 distance.

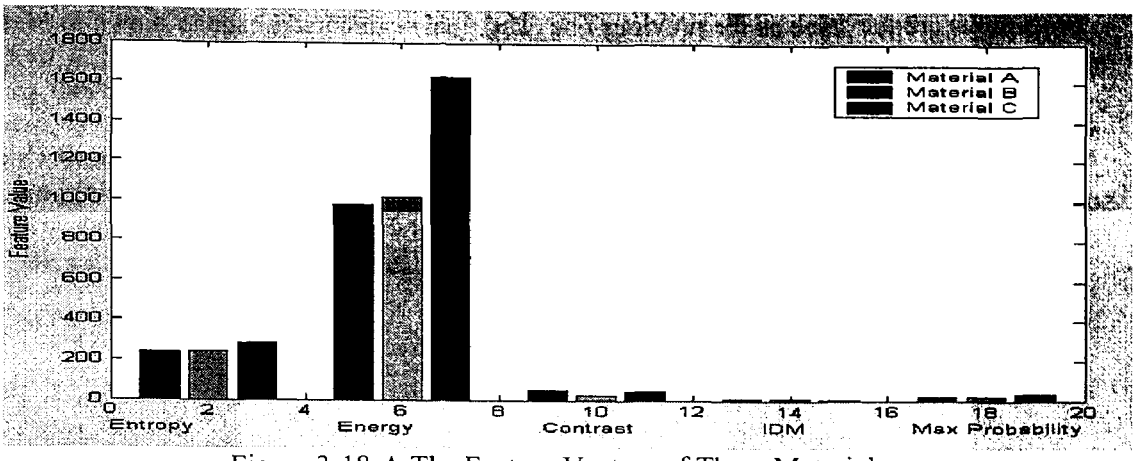


Figure 3-18-A The Feature Vectors of Three Materials

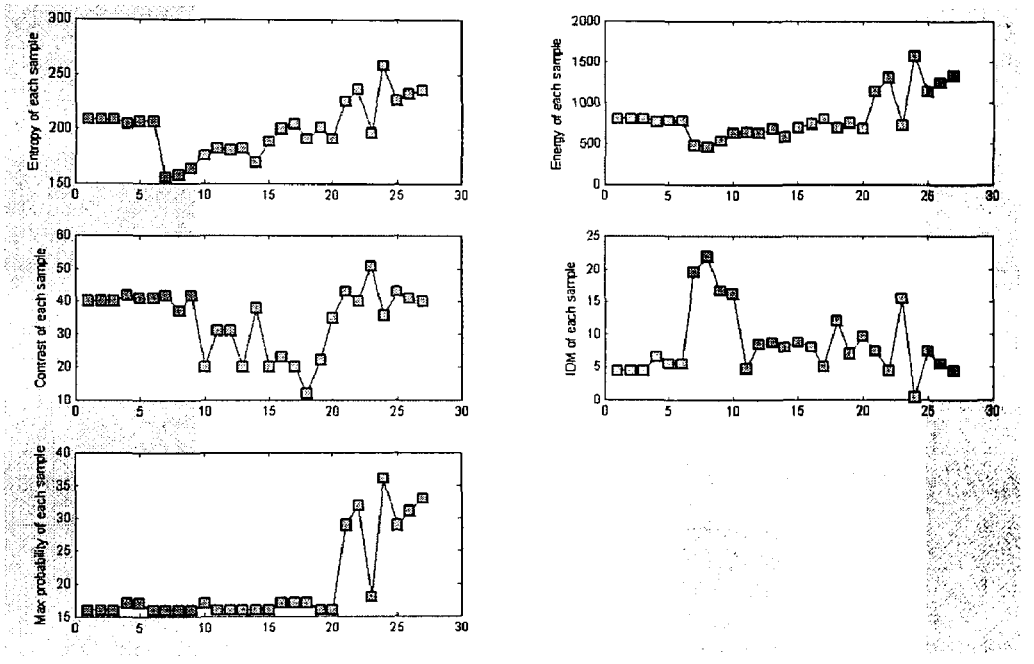


Figure 3-18-B Three Images Based Feature Vector Histogram

3.3.3 The Result Based on Three Input Images

For each kind of material, nine samples are taken from 3 original images at the same location. The sample capture program is used here. This time a threshold was held to filter out the background (pure black), so the final result looks much nicer. All the

procedures are the same as the previous one. The final segmented image is shown in Figure 3-19. In Figure 3-19 it is clear to be seen that around the brain there are skin and some other stuff. That kind of stuff is not needed in the brain segmentation. If taken out that stuff in this project, the result is shown like Figure 3-20. In order to compare, the gray level is assigned differently.

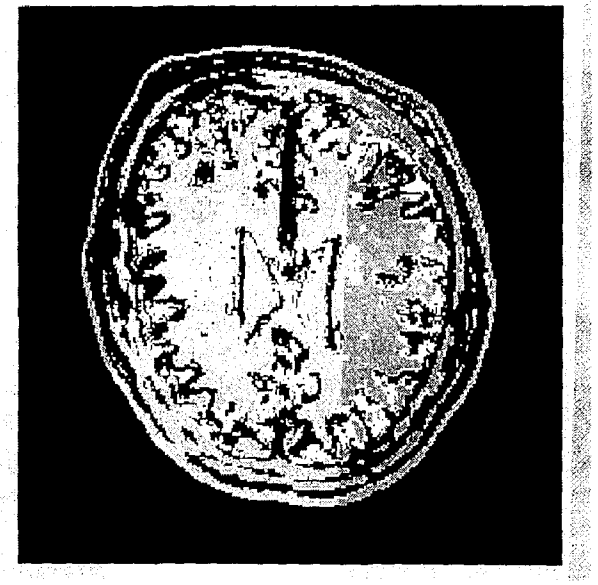


Figure 3-19 the segmented image with 3 samples from 3 original images

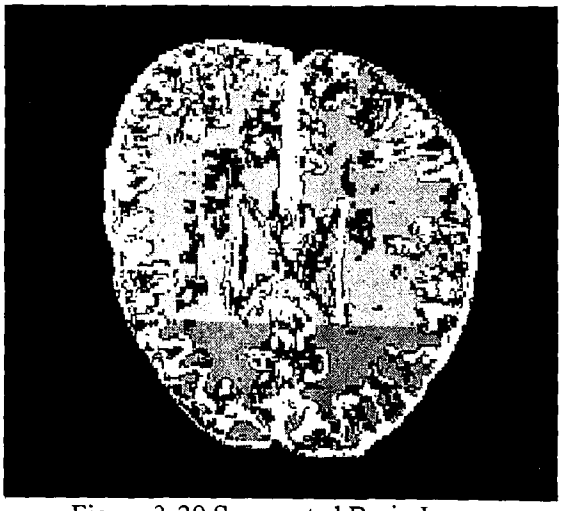


Figure 3-20 Segmented Brain Image

From Figure 3-19 and Figure 3-20, a conclusion can be drawn. The result with sampling from 3 images is much better than the one with sampling from two images. The materials are almost clearly segmented. That proves the idea of constructing the texture and then using Co-occurrence to analyze the brain image is a practical way.

Summary: The whole process of the development of the IID based MRI image segmentation system is presented in this chapter. Algorithm modification was discussed in details. The trial and test approach for the approval of the algorithm is included in this chapter. The effectiveness of L1 distance and feature vectors is approved. The final result is presented and approved to be effective.

CHAPTER IV

IMPLEMENTATION OF MRI IMAGE SEGMENTATION WITH HARDWARE DESCRIPTION LANGUAGE

Introduction:

One important part of this project is to implement the segmentation algorithm in FPGA using VHDL language. All the simulations in MATLAB in section III of this project can be considered the PC based implementation. In MATLAB, the images are considered as matrixes and MATLAB is pretty good at processing matrixes. The PC based implementation also provides the engineers and researchers very good debugging environment. So at the algorithm development stage, the PC based implementation is necessary and useful.

In most of the cases, the implementation has to be on a hardware device. DSP (Digital Signal Processing) device provides a very good solution because some of the DSPs provide floating point calculations, high speed real time processing, etc.

With the development of FPGA technology, it's becoming more and more popular in digital signal processing. Its reconfiguring and parallel processing capability is very useful for image and video processing.

The FPGA implementation with VHDL is hardware based, so some operations like matrix processing in MATLAB needs a lot of logic resources. A lot of such processing as division, multiplication and logarithm or floating point calculation become quite different in VHDL comparing with their counterparts in MATLAB. In MATLAB, all these

calculations can be done in floating point. But in FPGA, these operations are done in fixed point calculations. For sure this will bring some differences to the final result. At the end of this section, the results are compared with each other.

4.1 Block Diagram of The Algorithm Implementation in FPGA Using VHDL

Chapter two and three already described the outline of the theoretical approach of the IID based MRI image segmentation. From there, a brief diagram can be drawn as Figure 4-1.

The input data is a 4x4 or 5x5 image block. The average Co-occurrence matrix of the input data is calculated. From the Co-occurrence matrix, the feature vectors are calculated. These feature vectors are used to calculate the L1 distance with the three different material samples. The feature vectors of the three different materials are predefined. The minimum L1 distance is found that means the current block belongs to this type of material. Certain gray-level is assigned and the result is stored and output for display.

In this project, Block Random Access Memory (BRAM) module and Shift Register module are used for implementation. The input data is loaded from the hard-drive from the PC by test bench and stored in a BRAM module named "RAM144x2". "RAM16x8" is the BRAM module defined to store the Co-occurrence Matrix. "IIRAM-Syn" is the BRAM module defined to store the final output data for display.

"REGISTER1" is the Shift Register module defined for data delay and "REGISTER2" is the Shift Register module defined for address delay.

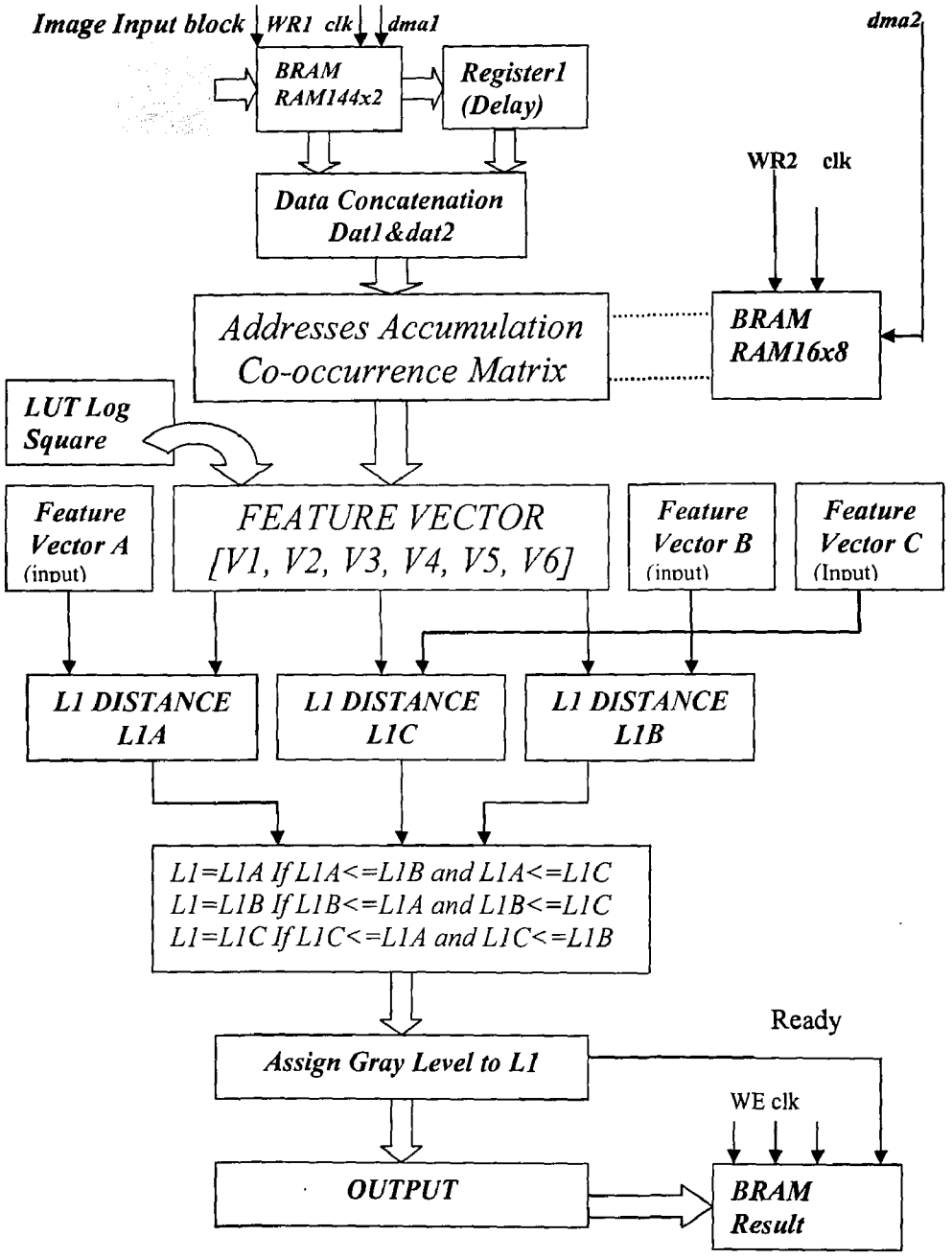


Figure 4-1 Block Diagram of Algorithm Implementation in Hardware Platform Using VHDL

4.2 The Implementation of Co-occurrence Matrix Calculation

The Co-occurrence matrix calculation in MATLAB has been introduced in previous section. MATLAB can handle matrix flexibly and images are considered as matrices. VHDL is hardware description language, and it describes has to be practically embedded into hardware. So it can not transform matrix directly. Comparing with MATLAB, another big difference is the input data. In MATLAB, the input data is the 256x256 raw image matrix. With the matrix index, the program can take any block of the image and process it. In VHDL or actual hardware, the data is stored in memory. Normally, it is serially read into the next processing word by word. For example, the data in MATLAB is in the following version:

"00","01","10","11",

"10","01","00","01",

"10","11","10","01",

"10","01","10","11",

The above matrix format needs 2-dimensional index to address each element. In this particularly implementation with VHDL, the above matrix is input like this: ["00","01","10","11","10","01","00","01","10","11","10","01","10","01","10","11"], which only needs one dimension index. (Please be noted that there are ways to index two or three dimensional matrix in VHDL). Because of the matrix processing, the Co-occurrence matrix calculation is just a few lines of program in MATLAB:

if (a1 == 1)

```
for row = 1:rows
    for col = 1:cols - d
        i = image(row, col);
        j = image(row, col+d);
        CM(i,j) = CM(i,j) + 1;
if (a2 == 1)
CM(j,i) = CM(j,i) + 1;
    end
  end
 end
end
```

the above program is to find the Co-occurrence matrix with distance 1 and direction of 0 and 180 degree. The idea is to start from the top left point, moving to the direction of east and west, each two adjacent pixels, say (r1, c1) and (r1, c2) with gray level 3 and 6 respectively, causes the value of element (3, 6) in the Co-occurrence matrix increment 1. Thus a certain texture pattern of the regarding image could be presented in its corresponding Co-occurrence matrix. Obviously, circuitry described in VHDL can not calculate the Co-occurrence matrix like this. In this project, the calculation of Co-occurrence Matrix is done as following:

First, the input data is loaded into "RAM144x2". The output of "RAM144x2" is named "dat2". (See Figure 5-3).

Second, "dat2" is loaded into "REGISTER1". The output of "REGISTER1" is called "dat1". Comparing with "dat2", "dat1" is one clock cycle delayed.

Third, "dat1" is logically concatenated with "dat2" to get an output. The block diagram of these steps is shown in Figure 4-3.

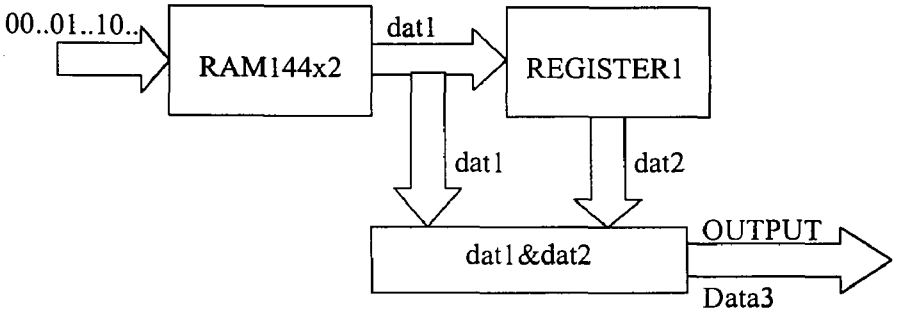


Figure 4-3 Diagram of Data Concatenation

The simulation result of the data concatenation process Figure 4-4. "data1" presents "dat1" in Figure 4-3, "data2" presents "dat2" in Figure 4-3. "Data3" is the output.

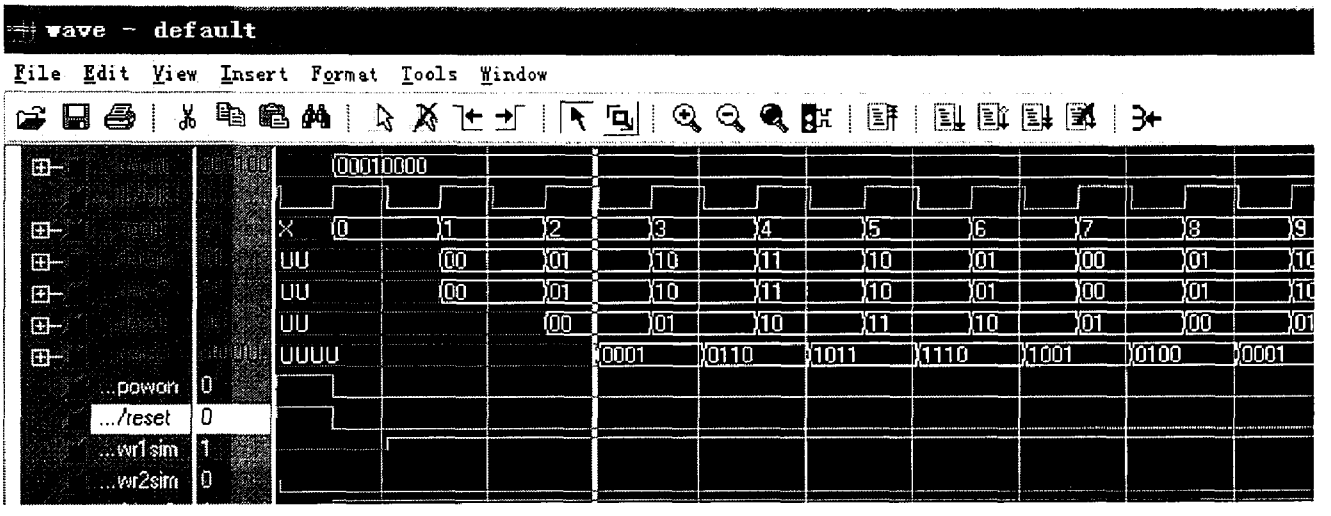


Figure 4-4 Simulation Result of Data Concatenation

As it is shown in Figure 4-4, "00" *&* "01"="0001", "01" *&* "10"= "0110". "*&*" is the concatenation function in VHDL. If the input image is a 4x4 block, there will be 16 output results. These results are used as the address of "RAM16x8", this RAM is used to store the value of the Co-occurrence matrix. That means when the input is the 1x16 one

dimension array such as:

["00","01","10","11","10","01","00","01","10","11","10","01","10","01","10","11"], there will be 15 addressees ["0001", "0110", "1011", "1110", "1001", "0100", "0001", "0110", "1011", "1110", "1001", "0110", "1001", "0110", "1011"].

The fourth step of Co-occurrence matrix calculation is "Addresses Accumulation". "RAM16x8"'s contents are all "0"s when reset. Whenever there is an address from the address concatenation process, the content of the address pointing to will increase 1. For example, when there is the first "1011", the content of address "1011" pointing to will become 1, if there is another one before all the address ends, the content will become 2. This is shown as Figure 4-5.

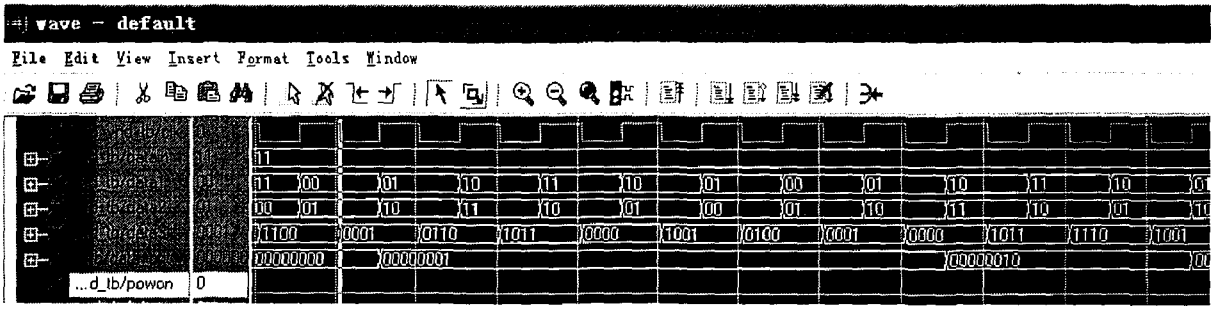


Figure 4-5 Addresses Accumulation

To show clearly the idea, the one dimension address array is transformed into Table 4-1. The elements which do not have any address will be filled with 0.

Table 4-1 Addresses Accumulation

	00	01	10	11
00	0	1+1	0	0
01	1	0	1+1+1+1	0
10	0	1+1+1	0	1+1+1
11	0	0	1+1	0

Address "00 01" shows up twice so the content of "0001" becomes 2, "01 10" shows

up three times, so the content of "01 10" becomes 3, the same thing with the rest of the addresses. The simulation result is shown in Figure 4-6.

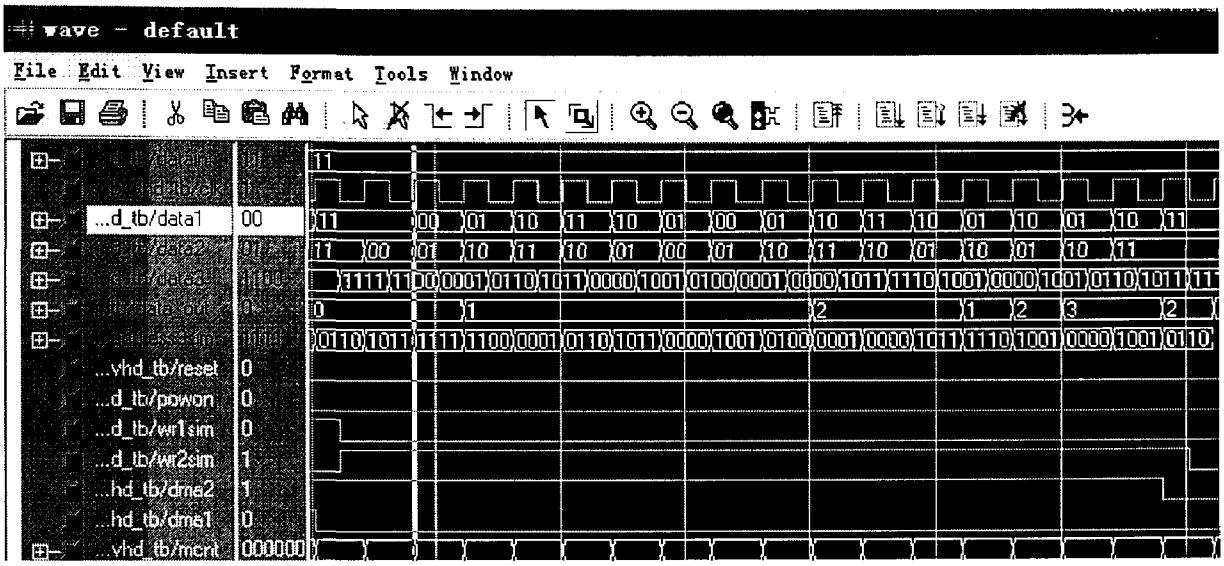


Figure 4-6 Co-occurrence Calculation

As shown in Figure 4-6, the operation starts from "0001" (the line points to). "Data_out" changes to 2 when the second "0001" shows up. After all the addresses have been checking through, the results of the Co-occurrence matrix is in Table 4-2.

Table 4-2 Co-occurrence Matrix Result

	00	01	10	11
00	0	2	0	0
01	1	0	4	0
10	0	3	0	3
11	0	0	2	0

The simulation result is in Figure 4-7.

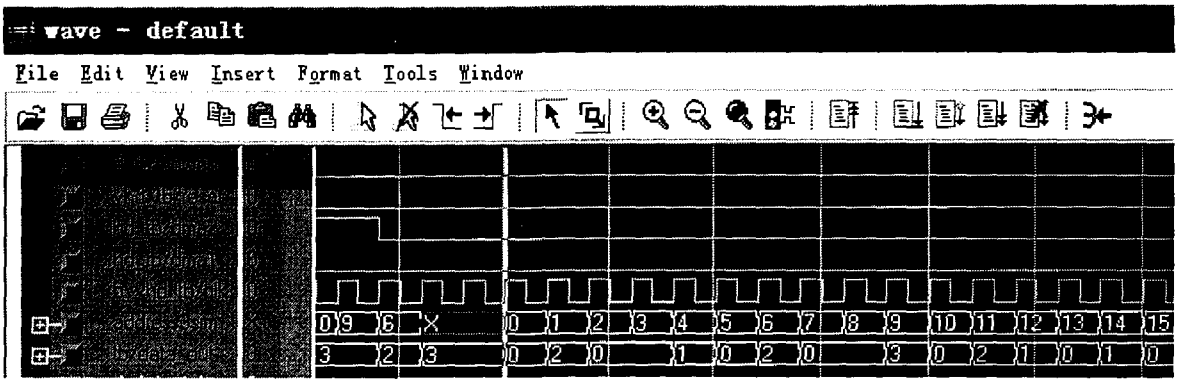


Figure 4-7 Co-occurrence Simulation Result

As shown in Figure 4-7. "Address3sim" is the address for "RAM16x8". The result of Co-occurrence Matrix is stored in this Block RAM. The results match Table 4-2. But neither Figure 4-7 nor Table 4-2 gives the right result. If checking carefully of this data array again:

["00","01","10","11","10","01","00","01","10","11","10","01","10","01","10","11"]

Figure 4-8 One Dimension Data Array

If this array is put back into a matrix:

"00","01","10","11"

"10","01","00","01"

"10","11","10","01"

"10","01","10","11"

Figure 4-9 Matrix Version of Figure 4-3

As predefined in the former section of this project, the distance of the Co-occurrence is "1" and the direction is 0 degree. But when the data is collected as an array, there will be some problems at the gray area shown in Figure 4-8. The gray area in Figure 4-8 is the same gray area in Figure 4-9. In Figure 4-8, from "11" to "10" in the gray

area, the distance is "1" and the direction is "0" degree, but in Figure 4-9, the distance is still 1 but the direction is not "0" degree, it is kind of 180 degree(in fact, there is no such definition here). So as shown in Figure 4-9, from "11" to "10", "01" to "10", "01" to "10", where the arrows are pointing, the three addresses have to be removed from the addresses list. To simplify the logic, in this project, all these addresses at the corners are set to "0000" by the following VHDL process:

```
outaddress:process (maincount)
    begin
      If (maincount="0000000000010111") or (maincount="0000000000011011") or
                                          (maincount="0000000000011111") then
                outaddr1 <="0000";
                else
                outaddr1 <= dat1 & dat2;
            end if;
    end process;
```

In this process, *MAINCOUNT* is the counter, when it reaches the turning point at "0000000000010111", "0000000000011011" and "0000000000011111", it will changes these addresses to "0000". The simulation is shown is Figure 4-10.

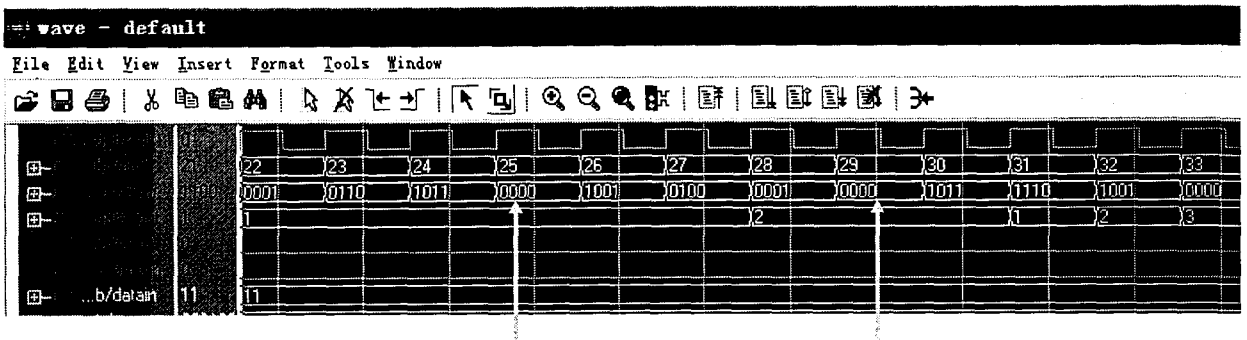


Figure 4-10 Turning Points

In Figure 4-10, the arrows are pointing to the addresses which have been changed to "0000". After these addresses have been changed to "0000", the contents of these addresses will be right, but the content of address "0000" is pointing to will have 3 extra "1"s added to it. So to get the right value, the content of address "0000" has to subtract 3. This is done by the following process:

```
if (addr="0000") then
      ram (0) := conv_std_logic_vector((unsigned(ram(0)) - 3),8);
    dataout <= ram (address);
    else
                dataout <= ram (address);
end if;
```

In the end, the right Co-occurrence matrix is in Table 4-3

Table 4-3 The Right Co-occurrence Matrix

	00	01	10	11
00	0	2	0	0
01	1	0	*2*	0
10	0	3	0	3
11	0	0	*1*	0

Comparing with Table 4-2, the underlined number is different. The simulation is in Figure 4-11.

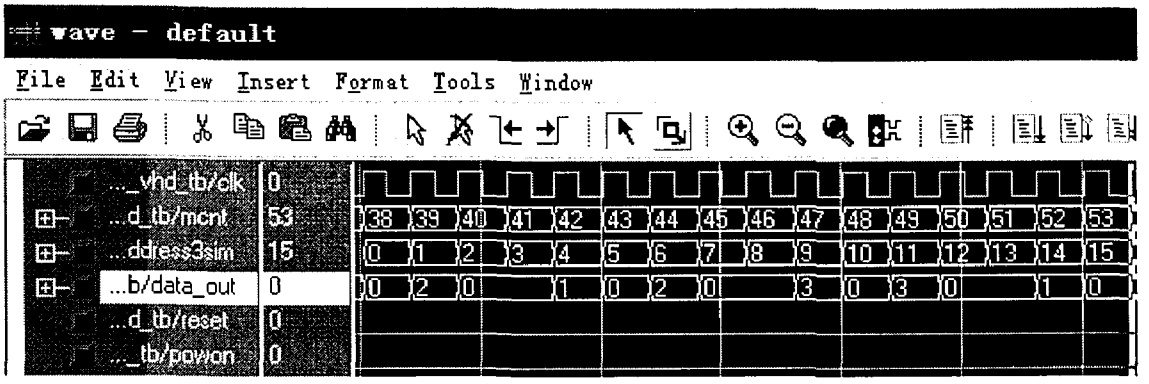


Figure 4-11 The Final Result of Co-occurrence Matrix

As shown in Table 4-3 and Figure 4-11. The result of the Co-occurrence matrix is right. If the input image block is changed from 4x4 to 5x5, there will be 4 addresses changed to "0000", so the content of "0000" has to subtract "4" to get the right value. When the Co-occurrence matrix is ready, it is time to calculate the feature vectors and the L1 distance. This will be discussed in the next section.

4.3 The Feature Vector and L1 Distance Calculation Implemented with VHDL

As described in section 4, when the Co-occurrence matrix is ready, it is time to calculate the feature vectors. Equations (4) to (10) present the mathematical formation of all the feature vectors. In VHDL, it is a little bit harder to do the mathematical operations. This section will describe the details one by one. Before starting the calculations, here is the brief introduction of the Co-occurrence matrix used here. The

Co-occurrence matrix result is from three input images. The input data is a 5x15 image block which is from three different images. It is shown in Figure 4-12.

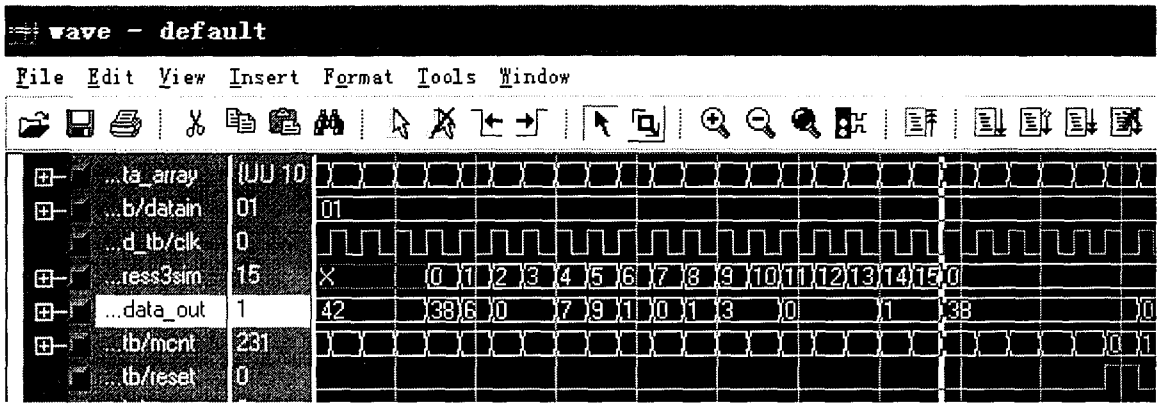


Figure 4-12 The Co-occurrence Waveform of 5x15 Image Input Block

The Co-occurrence matrix is shown in Table 4-4.

Table 4-4 The Co-occurrence Matrix of 5x15 Image Input Block

	00	01	10	11
00	38	6	0	0
01	7	9	1	0
10	1	3	3	0
11	0	0	1	1

A. ENTROPY

As equation (3) shows, function LOGARITHM needed to calculate the entropy. There is not such function in VHDL standard library, unless the third-party core is purchased. In this project, the maximum value for P is 70, the minimum value is 0. So the LUT table will be very useful here. All 70 values of *PlogP* were pre-calculated and stored in the LUT. So when there is any input P, there is a corresponding *PlogP*. The ENTROPY will be all the *PlogP* added together. All

the calculations in the LUT is round up with integer, because the fraction calculations are very resources consuming in VHDL.

B. ENERGY

As indicated in equation (6), ENERGY is the square sum of all Co-occurrence matrix elements. So the function of square is very important here. In VHDL there is not square function available in standard library. To simplify the logic, in this project, LUT is still the best solution. When the Co-occurrence value is input one by one, the corresponding square will be summed up and the result is the ENERGY.

C. INTERIA

In equation (5), there is the definition of INTERIA. Three steps to calculate INTERIA. 1. The absolute value of the difference between the row index and column index of the Co-occurrence Matrix, called $|i-j|$.

2. The square of $|i-j|$, this is done using the same idea as LUT.
3. Multiply the result of step 2 with the corresponding Pij. (Pij is the Co-occurrence matrix element).

D. CONTRAST

As described in equation (7), contrast is the sum of product of $(i-j)$ and the square of Pij. The square can be found in the LUT, $(i-j)$ is simple to approach.

E. INVERSE DIFFERENCE MOMENT

Equation (8) describes the Inverse Difference Moment. It is very similar with contrast. Instead of multiplication, the square divided by the absolute difference of $(i-j)$, then sum up, the result is the IDM.

F. MAXIMUM PROBABILITY

Just to compare and always keep the larger value, when finish checking all the elements of the Co-occurrence matrix, there is the maximum probability.

The simulation result of all the feature calculations is in Figure 4-13

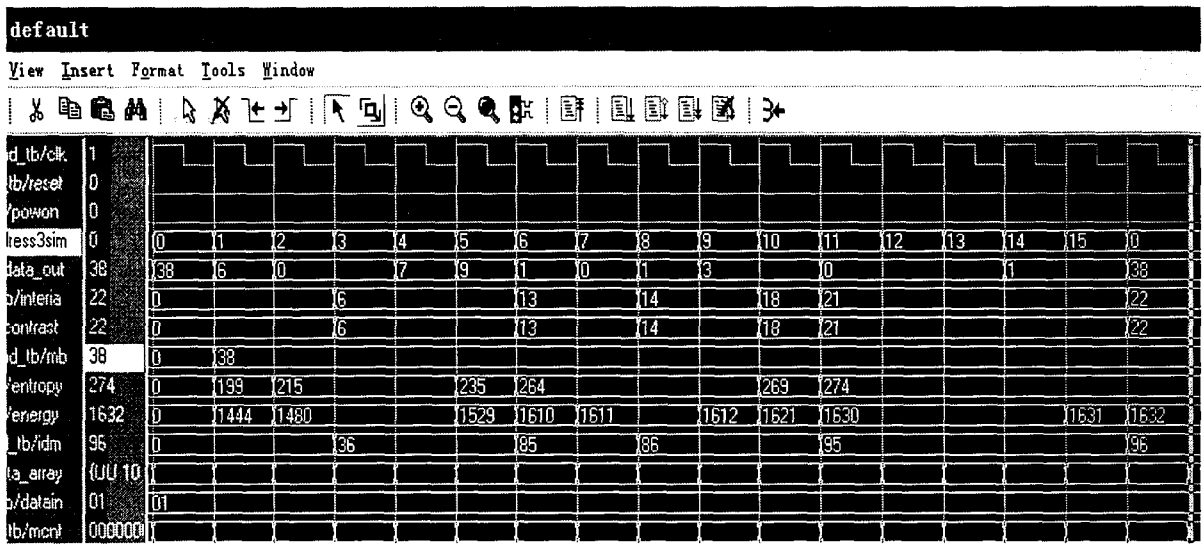


Figure 4-13 Feature Calculation

One input image block's feature vectors calculation is shown Figure 4-13. "Address3Sim" is the address of the block memory (RAM16x8) which stores the Co-occurrence matrix information. There are 16 values, so the addresses are from 0 to 15. The addresses and corresponding Co-occurrence matrix value are:0→38, 1→6, 2→0, 3→0, 4→7, 5→9, 6→1, 7→0, 8→1, 9→3, 10→3, 11→0, 12→0, 13→0, 14→1, 15→1. Taking ENERGY as one example: ENERGY=38x38+6x6+0x0+0x0+7x7+9x9+... =1444+6x6+0x0+0x0+7x7+9x9+...=1480+0x0+0x0+7x7+9x9+...=1529+9x9+...=1610+....This is same as shown in Figure 4-13.

When all the calculations are done, the results are shown in the last clock cycle of Figure 4-13. Putting the results in a vector, it is [22, 22, 38, 274, 1632, 96]. The

elements are corresponding to INTERIA, Contrast, Maximum Probability, Entropy, Energy and Inverse Difference Moment respectively.

G. L1 Distance Calculation

When the feature vectors are ready, it is time to calculate the L1 distance. As shown in equation (11), the L1 distance calculation involves the division. As known, it is straight forward for FPGA to do the divisions of any number which is the power of 2. But in this case, the division has a lot of numbers which are not the power of 2. To simplify the calculation, the division part is removed here. So the L1 distance will be the sum of the difference of the feature vectors. This for sure will introduce some deviations to the results.

The simulation result is shown in Figure 4-14.

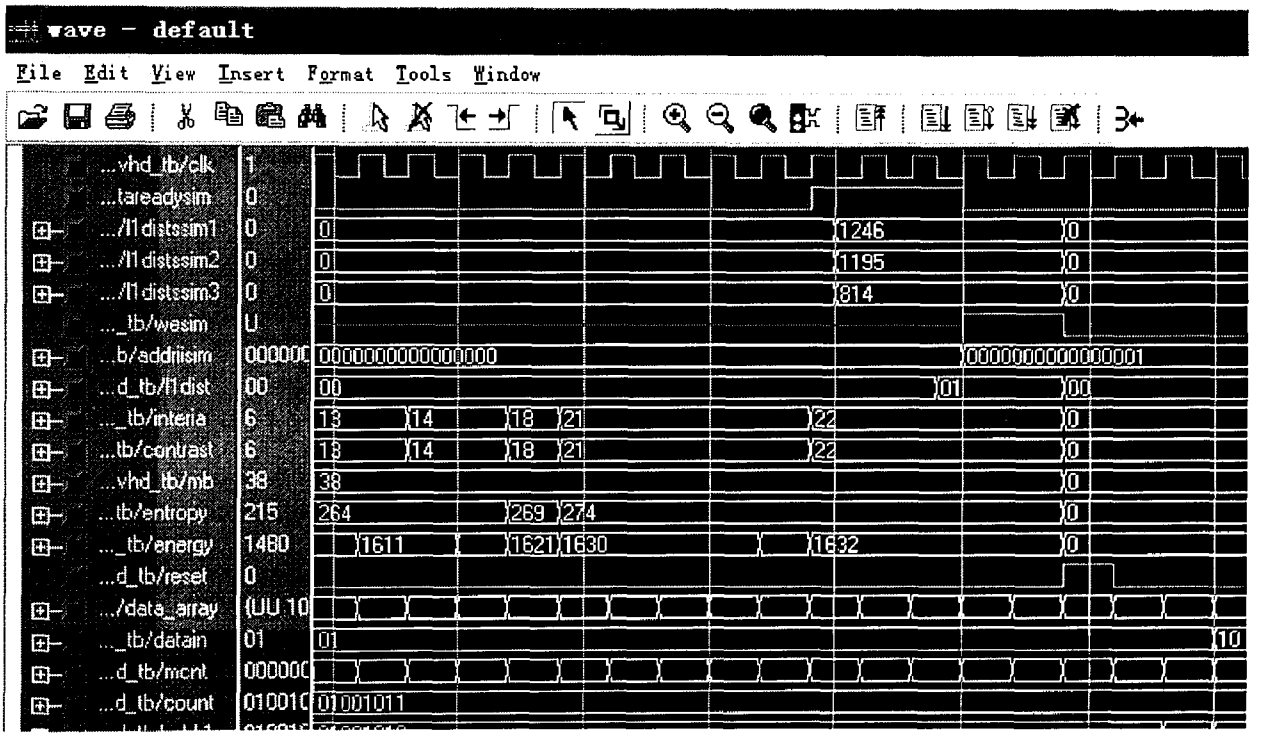


Figure 4-14 L1 Distance Calculation

In Figure 4-14, "L1distssim1" and "L1distssim2" and "L1distssim3" are the three L1 distance simulation results, they are 1246, 1195 and 814 respectively. The minimum value will be chosen from the three values. In this case, "L1distssim3" is the one. So the first block is judged to belong to the third type of materials (which the L1 distance is calculating against with). "01" will be assigned to this block. When the write enable signal "WESIM" is ready, "01" is written to address "0000000000000001" of "addriisim". "Addriisim" is the address output Block RAM ("IIRAM-SYN"). After the whole image is finished, there should be 65536 values in this memory.

4.4 The Block RAM Module for Data Input and Output

Most of the XILINX devices provide embedded dual-port RAM modules. Virtex-II devices feature a large number of 18 Kb block SelectRAM memories. [10] The block SelectRAM memory is a True Dual-Port RAM, offering fast, discrete, and large blocks of memory in the device. The memory is organized in columns, and the total amount of block SelectRAM memory depends on the size of the Virtex-II device. The 18 Kb blocks are cascadable to enable a deeper and wider memory implementation, with a minimal timing penalty incurred through specialized routing resources. [19]

The Block RAM is used a few times in this project. There are three Block RAM Modules used in this project. The first one is for the image block input. The second one is for Co-occurrence matrix. The third one is for the final result output. All these three Block RAMs are used as single ported only.

The image input Block RAM is for 256 pixels, there are 8 bit per pixel, so this RAM is 256x8. In this project, maximum there are 75 input values, so it is more than enough. The

control signals are address (address1Sim) and Read and Write Enable (Wr1Sim). Figure 4-15 is write operation of this Block RAM. When "wr1sim" is set to "1", the Block RAM is in "Write" mode, the data ("datain") is written into this Block RAM, one data per clock cycle. "address1sim" is the writing address.

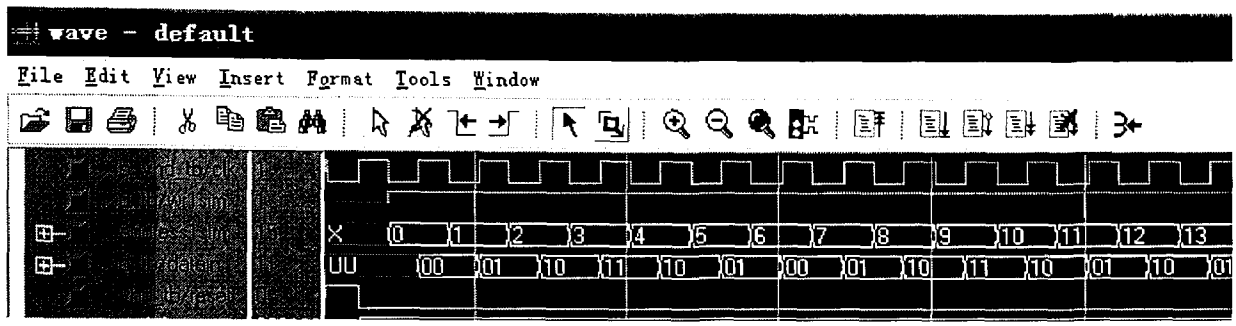


Figure 4-15 Write Operation of Input Block RAM

Figure 4-16 is the read operation of input RAM. When "wr1sim" is set to "0", the Block RAM is in "Read" mode, the data ("data2") is read out from this Block RAM, one data per clock cycle. "address1sim" is the reading address.

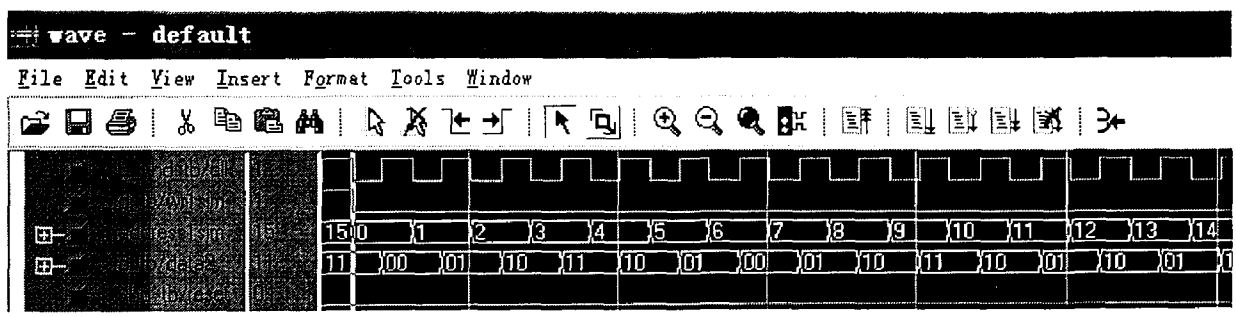


Figure 4-16 The Read Operation of Input Block RAM.

For the Co-occurrence matrix Block RAM (RAM16x8), as discussed in previous section, if the write enable signal is ready, "wr2"="1", when the address of this Block RAM appears, the corresponding content of that address pointing to will increase "1". This is done as shown in Figure 4-17. This RAM is defined as 16x8, because there are only 16 values in this Block RAM.

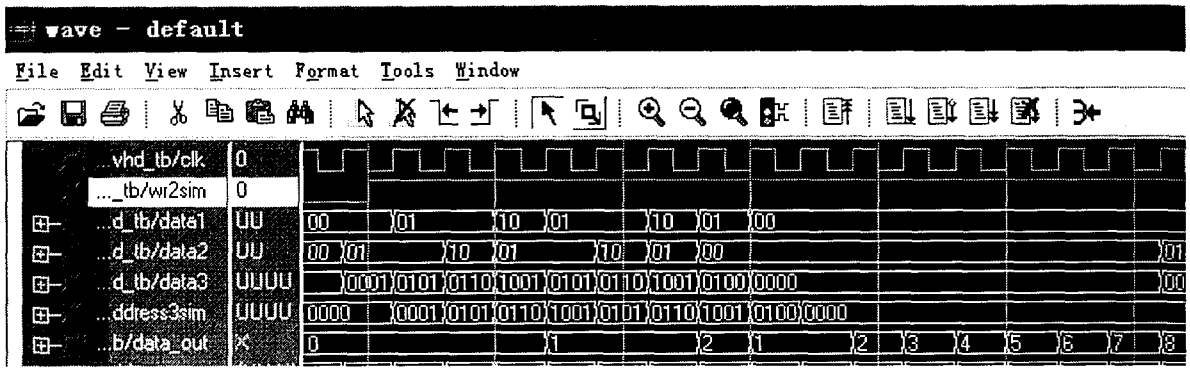


Figure 4-17 Co-occurrence Matrix Block Ram Write Operation

It can be seen from Figure 4-17, the content of address "0000" has been increasing from 0 to 8, and "1001" from 0 to 2. "Address3Sim" is the address for Co-occurrence matrix Block RAM. The read operation is in Figure 4-18.

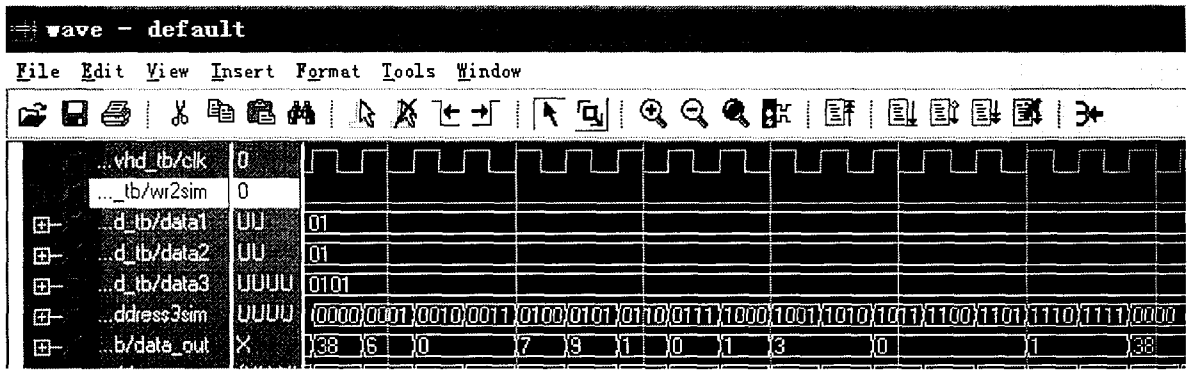


Figure 4-18 Read Operation of Co-occurrence Matrix

When "wr2sim" =0, the read signal is effective. At this time, the content is sending out to "data-out" as shown in Figure 4-18.

For the above Block RAM, only the read and write control signals are not enough. There are some other signals to control the read and write signals, they will be discussed in the later section.

The final result output Block RAM is already discussed in previous section, so it won't be repeated here.

4.5 The Register Module for Data Delay and Address Delay

In this project, register modules are used to delay the signals. The first register module is used to delay the input data. The purpose of this delay is to produce the addresses of the Co-occurrence matrix. As shown in Figure 4-19, "data1" is one clock cycle after "data2". The second register module is used to delay the address. As seen in Figure 4-20, "address3" is one clock cycle after "address2".

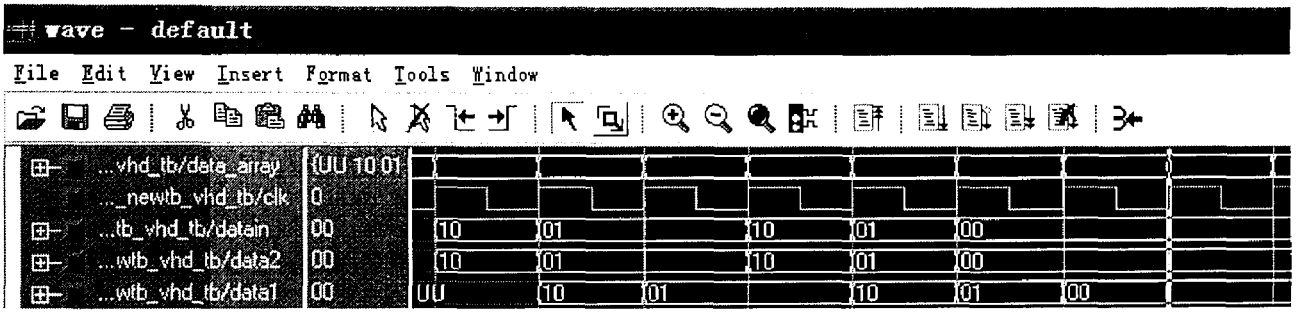


Figure 4-19 Input Data Delay

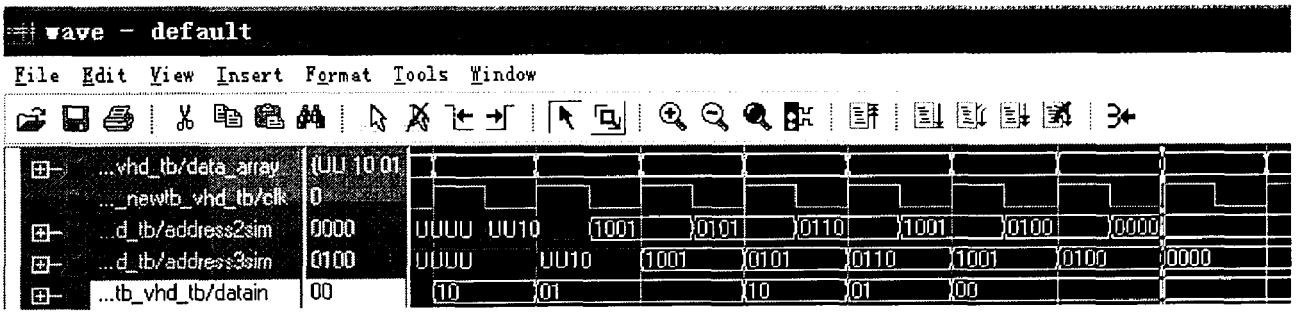


Figure 4-20 Address Delay

4.6 Test Bench Design for simulation

"With gate counts and system complexity growing exponentially, engineers confront the most perplexing challenge in product design: functional verification". [7] So functional verification is a very important step to the design. In this project, the software used for functional verification is MODELSIM from Model Technology. The version used is ModelSim XE II 5.7 C, which is the custom version with XILINX ISE 6. So it is very convenient to use with XILINX design environment.

4.6.1 Input Data Preparation and Organization

In this project, the input image size is 256x256, so there are total 65536 pixels in the original image. As discussed in section IV, if three images were taken as input, and each block took 25 (5x5) pixels. These three "5x5" blocks are put together from left to right, so each image block is 5x15=75 pixels. In order to improve the segmentation quality, when the window is scanning through the image, each time the window only move one pixel. So only the pixel at the centre point is replaced with the new gray level. In this case, there are 63504 image blocks (because of the edge). In total there are 4762800 pixels as input.

As mentioned in previous section, the input data is in a one dimensional array: 1x75. In this project, MATLAB is used to pre-organize the input data. MATLAB puts all the 5x15 image blocks into a data array, and then put all these data into a binary file sequentially from the first image block to the last one. Test Bench will read this file from the hard-drive of the PC.

Test Bench handles the binary file as the following process:

```
Read_From_File: process(clk1)
      Variable indata_line:line;
               Variable indata:integer;
               variable h    :integer :=0;
   file input_data_file:text open read_mode is "C:/PROJECT/matlab/data/orig2.bin";
begin
      if rising_edge(clk1) and (h<=1024145) then      --1024145--4762800
                readline(input_data_file,indata_line);
                read(indata_line, indata);
                ID<=conv_std_logic_vector(indata,2);
                 data_array(h) <=ID;
                 h:=h+1;
                  if endfile(input_data_file) then
                        report "end of file";
                        file_close(input_data_file);
              file_open(input_data_file, "C:/PROJECT/matlab/data/orig2.bin");
         end if;
       end if;
end process;
```

Orig2.bin is the binary file of the input data which is saved in the hard-drive under the directory of "C:/PROJECT/matlab/data". In this process, "CLK1" is used. It is a different clock. It is only for reading the data from the binary file. This clock is independent with the processing clock "CLK". The function of this process is to open the binary file and write the data into "data_array", one data per clock cycle. The reading process is ahead of time of the processing. So when the main process reads data from "data_array", the data is always ready there. "h" is to setup to stop reading at certain time. For example, in the program shown above, the reading process will stop at the 1024145th data. This is good for debugging the program.

Test bench needs read each image block for processing. The reading operation is done by the following process

```
for j in 0 to 65535 loop
               ……
               k:=j*75+1;
               ……
```

```
for i in 0 to 74 loop
addr1 <= conv_std_logic_vector(i,8);
.....
DATAIN<=Data_array(i+k);
end loop;
```

There are total 65535 image blocks, each time 75 data is read into the program for processing. The reading of these 75 data is done by the internal loop. "Addr1" is the address for input Block RAM "RAM144x2". ***K*** controls which image block to read. For some reason the first data of "Data_array" is not an effective one, so the first data is skipped by adding one to K (***k:=j*75+1*** instead of ***k:=j*75***).

4.6.2 Output Data Organization

The output is the reversed operation of the input. When all the output data is ready in the RAM, it will be sent out. The following process is for data output:

```
write_to_file: PROCESS (clk1)
      variable outdata_line: line;
     variable outdata:integer:=0;
    variable holdon:std_logic;
file output_data_file: text open write_mode is
"C:/PROJECT/matlab/vhdl_output521_2.bin";
  begin
     if (WEsim='1') then
          if rising_edge(clk1) then
               outdata:=abs(CONV_integer(L1Dist));
                    write(outdata_line, outdata);
                    writeline(output_data_file, outdata_line);
           end if;
        end if;
  end process;
```

In this process, when "WEsim" is effective, the data in this Block RAM will be written in the file named "vhdl_output521_2.bin". This file is saved on the hard-drive

under the directory of "C:/PROJECT/matlab". In both input and output processes, "std.textio.all" needs to be included in the library.

This file will be loaded by MATLAB. MATLAB takes the data and reform into an image, the image is the segmented image.

The idea used to communicate between MATLAB and ModelSim in this project make it possible to test and verify the algorithm and simulations result. This is good because the final result can be verified and viewed before the design approach to the hardware stage.

4.6.3 Timing Analysis of Control Signals

In previous section, all the components have been introduced. The timing of these components has to be controlled to get the right result. In logic design, timing is very important. In this project, the control signals are introduced by two parts. The first part is the Co-occurrence. The simulation waveform is in Figure 4-21.

As shown in Figure 4-21, "Clk1" is an independent clock, it is only for reading all the data from PC to test-bench. "data_array" is the array where test-bench storages all the data. Because this "data_array" needs to load 4762800 bytes into the memory of the PC, so the simulation process is very slow and resource consuming, the PC was frozen many times during the simulations. "CLK" is the clock signal which drives the rest of the logics. "POWERON" signal is the initial reset signal, so when this signal is "1", the whole system will reset. That means "reset" signal is effective. The reason to have "POWERON" is that the "reset" signal is effective for every image

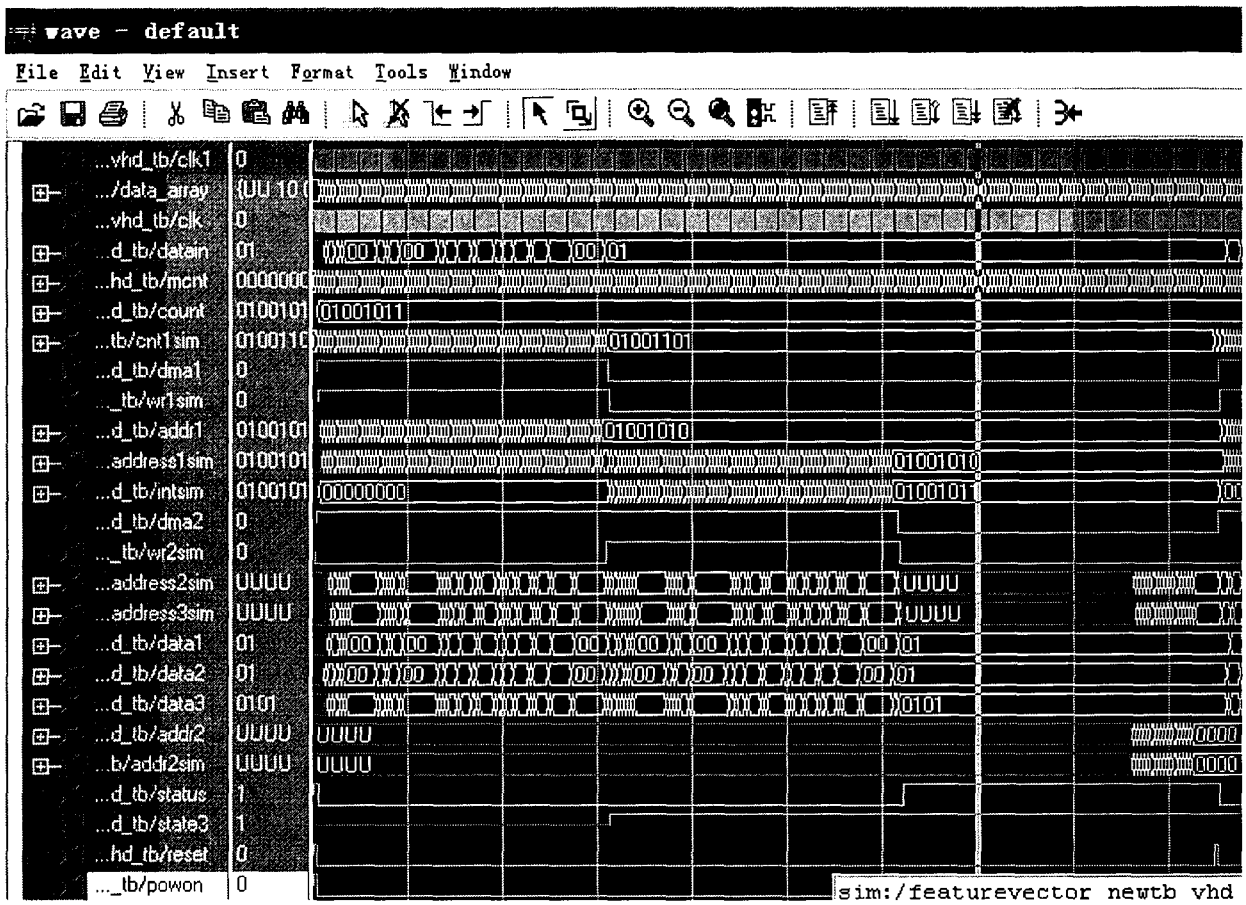


Figure 4-21 The Control Signals of Co-occurrence Matrix Calculation

block. When "reset" is effective "dma2" is set to "1", "count" is set to "75", which is the total quantity of each image block. When "dma1" is "1", "WR1" is set to "1" immediately. So the data is read into the program. "cnt1sim" is the counter for reading the data, when it reaches "75", that means the data reading process finish for the current image block. Then "WR1" is set to "0". When "dma2" and "wr2" are both effective "1", the input RAM is in read mode, so the data is read out from the RAM and the calculations of Co-occurrence matrix starts. "WR2" is effective means

"RAM16x8" for Co-occurrence matrix is in "write" mode. After the Co-occurrence matrix is calculated, the values are saved in this RAM. When "WR2" is set to "0", "RAM16x8" is at "read" mode, the Co-occurrence matrix value is read out for feature vector calculation. The feature vector and L1 distance calculation control signals are shown in Figure 4-22.

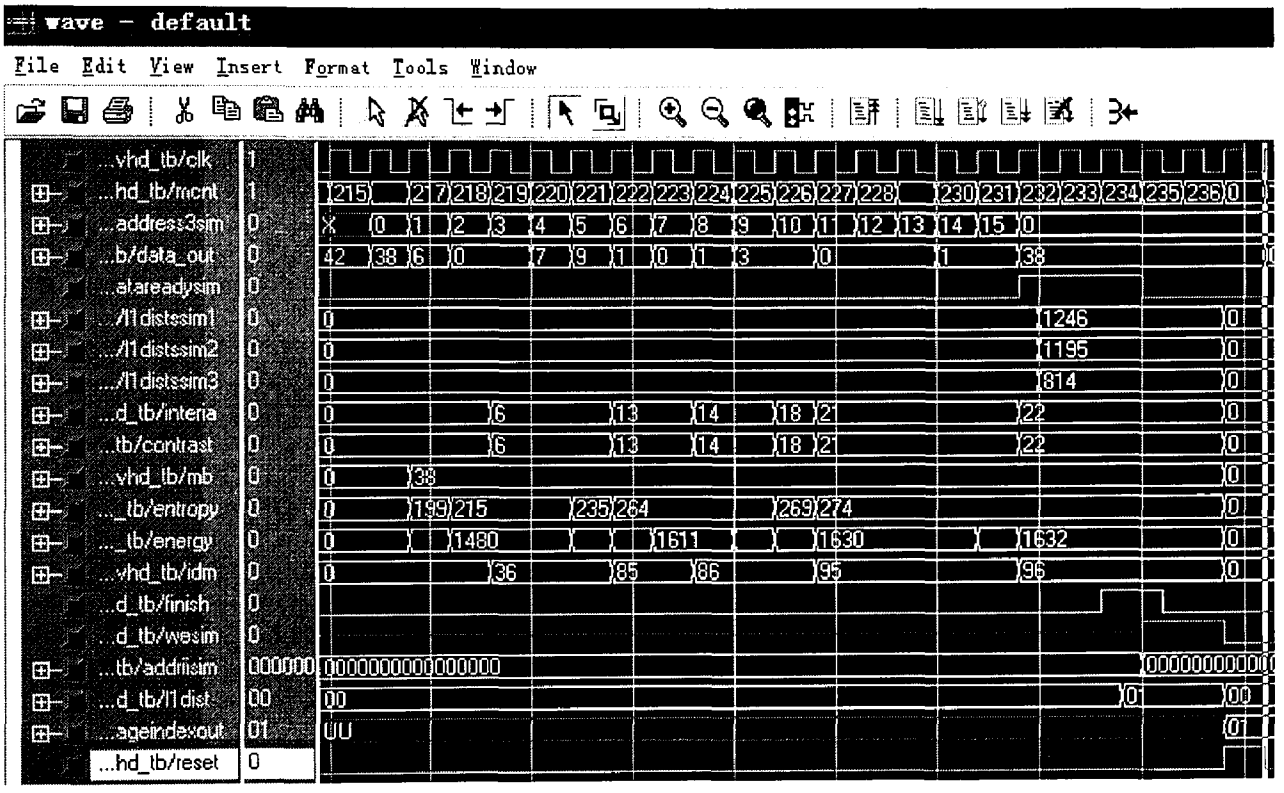


Figure 4-22 Feature Vector and L1 Distance Calculation Control Signals

The Co-occurrence matrix data are loaded out to calculate the feature vectors. When the feature vectors are ready, "datareadysim" is set to be effective for three clock cycles. In this three clock cycles, the L1 distance is calculated. When L1 distance

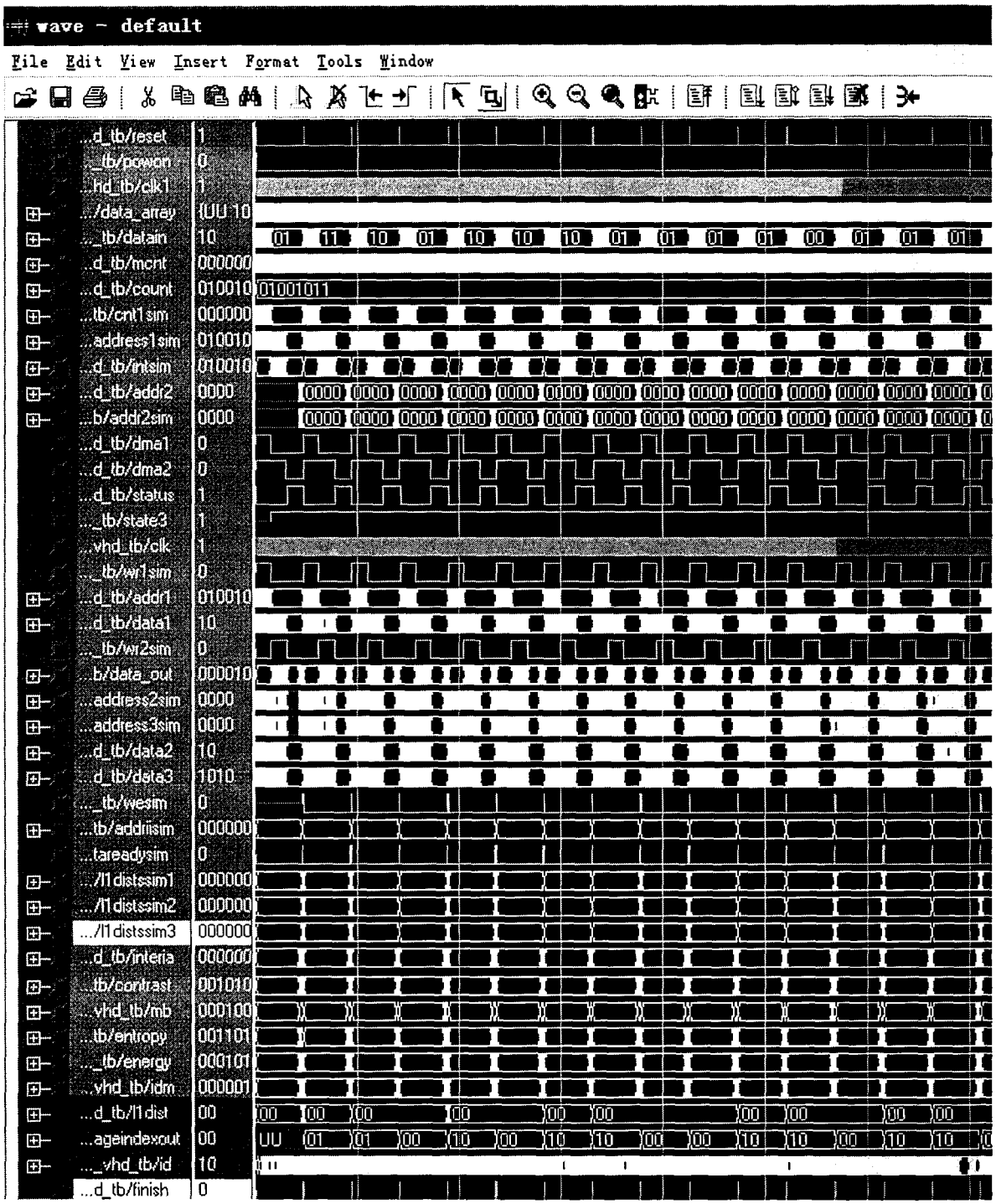


Figure 4-23 The Overview of All The Simulation Waveforms

calculation is done, "finish" signal is set to be effective. "finish" makes "WE" ("wesim" in Figure 4-22) effective. "WE" is the "write-enable" signal for the output Block RAM, when this signal is ready, the gray-level for the current image block is written into this Block RAM. Figure 4-23 is the overview of all the simulation waveforms of this project.

In Figure 4-23, there are around 16 image block processing. For the whole image there are around 65535 such processing.

Summary:

The IID based MRI image segmentation was implemented in FPGA using VHDL. The implementation scheme is presented and the specific approaches for Co-occurrence matrix calculation implementation, logarithm calculation implementation and square calculation implementation are discussed. The logic hardware modules like Block RAM, Register are presented. The timing controls and data input and output were discussed in details. The simulation waveforms are presented.

CHAPTER V

RESULT COMPARISION, ANALYSIS AND CONCLUSION

For the above simulations, the FPGA chip used is XILINX "XC2V2000-6bf957". This device has 24, 192 logic cells and 1008 K bits BRAM and 624 user IO. [10] APPENDIX A1 is the device utilization summary. As shown in APPENDIX A1, the slices are not enough to accomplish the project. There are two ways to solve this problem. One way is to change the device which has more resources. This for sure will increase the cost. Another way is to optimize the design to reduce the unnecessary logic resources.

The timing report of this project utilizing this device is shown in APPENDIX A2. The timing report in APPENDIX A2 does not have practical value because the device does not have enough resources. For comparison, device XC2V4000 was selected. XC2V4000 has 51,400 logic cells, which doubled the logic cells of XC2V2000 [10]. The rest of the resources like BRAM, Multipliers and user I/Os also doubled the ones in XC2V2000. APPENDIX A3 is the device utilization summary of XC2V4000.

As shown in APPENDIX A3, XC2V4000 has enough slices to accomplish this project. Its timing report is in APPENDIX A4. From APPENDIX A2 and A4, it can be seen that there is almost no difference between these two timing reports. The maximum required output time is a little bit delayed in APPENDIX A4.

Same as MATLAB based simulation, in this project FPGA based simulation has three different results classified by one, two or three input images. One input image based

segmentation result is very close to Figure 3-10. Two input images based segmentation is very close to Figure 3-12. Both of these results are not very satisfying results, so there is no need to discuss any more here. Figure 5-1 is the three input image based segmentation result. Comparing with Figure 3-12, it is very close. The result is satisfying. But because of the calculation errors between FPGA based and MATLAB based simulations, there are some differences between these two results. With more detailed guidance of pathological knowledge, these differences can be defined as good or bad, so the segmentation result can be improved easily. If considering the result in Figure 5-1 as a good segmentation result, a FPGA based segmentation system can be implemented with hardware system. This system for sure will include FPGA, SRAM. Such system has its own advantage and disadvantage against the PC based system.

Figure 5-1 FPGA Based Segmentation Simulation Result

Table 5-1 Comparison between PC based System and Hardware Based System

	Matlab (PC) Based System	Hardware (FPGA) Based System	ADVANTAGE
SPEED	180 Second	0.7S	**257 times faster real-time processing**
COST	~1000$	~100$	**~10 times cheaper**
SIZE	0.032M3	0.0015M3	**20 times smaller**

From Table 5-1, it can be seen the hardware based system has a lot of advantages. Real-time processing is one of the most important characters of the FPGA based system. And the processing speed won't change too much when the size of input images are increasing. That is because of the parallel processing ability of the system. The input images can always be divided into several images and be processed at the same time. But for the PC based system the increasing size of input images means to dramatically increase the processing time.

SUMMARY

This project went through the whole process of algorithm modification, theoretical verification and simulation by MATLAB and most importantly the implementation with FPGA using VHDL.

The simulation with MATLAB approves the IID based segmentation to be an effective way for MRI image segmentation. Because of its special characteristics, the gradient or edge detection based segmentation idea can not segment the exact border of the different materials inside the MRI image. The IID based segmentation is based on the Co-

occurrence of the different materials, so it has brought some effective elements into the segmentation.

The implementation with FPGA in VHDL is an important part of this project. VHDL is hardware description language, so for MATLAB and C++, it is software programming, but for VHDL, it is hardware designing. So at the beginning of this project, a lot of troubles showed up like changing the method from "programming" to "designing". The MATLAB programming is more like a sequential thing, but the VHDL design is a parallel thing, so the "timing" becomes extremely important. After the barriers were conquered one by one, the implementation with FPGA using VHDL was successfully accomplished. The result is very close to the MATLAB based simulation result. The result is satisfying. That approves the IID based MRI segmentation can be practically implemented with FPGA hardware. And the FPGA hardware based system is faster and more economical.

There are still a lot of things can be done regarding the extension of this project. To physically implement the system will top everything else. It will take a lot of guts and time to do it. On the algorithm side, the different directions and different distances based Co-occurrence matrix can be tried for segmentation. And a lot of new algorithms can be adapted to this system also, such as the self-organized segmentation system [21]. And with more pathological research on the subject can help the system with more practical value.

References

1. R. M. Haralick and L.G. Shapiro, "Survey: image Segmentation," *Computer Vision, Graphics, Image Proc.*, vol.29. pp.100-132, 1985.
2. R.M.Haralick and L.S. Shapiro, *Computer vision* (Vol.1). Addision Wesley Reading, MA pp.46-50 (1992).
3. BIR BHANU, SUNGKEE LEE and SUBHODEV DAS. "Adaptive Image Segmentation Using Genetic and Hybrid Search Methods". *IEEE Transactions on Aerpspace and Electronic Systems* VOL.31, NO.4 pp.1268-1291, OCTOBER 1995.
4. Archibald R; Chen KW; Gelb A; Renaut R. "Improving tissue segmentation of human brain MRI through preprocessing by the Gegenbauer reconstruction method". *NEUROIMAGE* 20(1): pp 489-502. 2003
5. R,M. Haralick, K. Shanmugan, and I. Dinstein, "Textural features for image Classification," *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-3, pp.610-621, June 1973.
6. M. M. Galloway, "Texture analysis using gray level run length," *Comput. Graph. Image Processing*, vol. 4, pp. 172-179, 1975.
7. Janick Bergeron "Writing TestBenches—Functional Verification of HDL Models". Qualis Design Corporation, Kluwer Academic Publishers, 2000.pp 1-3
8. Vassili A. Kovalev, et. "Three-Dimensional Texture Analysis of MRI Brain Datasets", *IEEE Trans on Medical Image Processing*, vol.20, NO.5, pp424-433. May 2001

9. Gabriele Lohmann, "Analysis and synthesis of Textures: A Co-occurrence-based Approach". *Comput. & Graphics*, Vol.19, No.1, pp. 29-36, 1995

10. XILINX.Virtex II Platform FPGA User Guide. Page 112. UG002 (v2.0) 23 March 2005. XILINX

11. Unqing Chen, Aleksandra Mojsilovic. "Adaptive Image Segmentation Based on Color and Texture". *Proceedings of IEEE International Conference on Image Processing (ICIP'02)*, Rochester, New York, Sept. 2002 pp 777-780.

12. V. Gemignani, M. Demi, M. Paterni, and A. Benassi, "Real-time implementation of a new contour tracking procedure in a multi-processor dsp system," Circ. Sys. Commun. Comp., pp.3521-3526, 2000.

13. Steffern Klupsch, etc. "Real Time Image Processing Based on Reconfigurable Hardware Acceleration". *Proceedings of IEEE Workshop Heterogeneous Reconfigurable Systems On Chip (SoC)*, (no pp). April 2002.

14. http://people.cs.uchicago.edu/~pff/segment/beach.gif

15. http://people.cs.uchicago.edu/~pff/segment/beach-seg.gif

16. http://q3.beyondirc.net/textures/fabric.zip

17. http://www.theswancorp.com/.../ images/pebble.jpg

18. http://robotics.stanford.edu/~ruzon/NASA/MPF_sol3_lander.jpg

19. Virtex-II Platform FPGAs: Complete Data Sheet. Page 3. DS031 (V3.4) March.1, 2005. XILINX

20. G.Qiu, "Constraint Adaptive Segmentation For Color Image Coding and Content-Based Retrieval". IEEE Workshop on Multimedia Signal Processing, October, Cannes, France. pp.269-274 IEEE 2001.

21. Axel Wismuller, Frank Vietze, Johannes Behrends, Anke Meyer-Baese, Maximilian Reiser, Helge Ritter "Fully automated biomedical image segmentation by self-organized model adaptation" *Neural Networks* 17 pp. 1327–1344. 2004

APPENDIX A1 Device Utilization Summary for XC2V2000

Device utilization summary:
Selected Device : 2v2000bf957-6

Number of Slices:	*16916*	*out of 10752*	*157% (*)*
Number of Slice Flip Flops:	*1221*	*out of 21504*	*5%*
Number of 4 input LUTs:	*15934*	*out of 21504*	*74%*
Number of bonded IOBs:	*220*	*out of 624*	*35%*
Number of MULT18X18s:	*2*	*out of 14*	*14%*
Number of GCLKs:	*1*	*out of 16*	*6%*

WARNING: Xst:1336 - (*) More than 100% of Device resources are used

APPENDIX A2 Timing Report for Device XC2V2000

TIMING REPORT

Clock Information:

Clock Signal	*Clock buffer (FF name)*	*Load*
clk	*BUFGP*	*3269*

Timing Summary:
Speed Grade: -6

Minimum period: 22.062ns (Maximum Frequency: 45.327MHz)
Minimum input arrival time before clock: 7.879ns
Maximum output required time after clock: 11.056ns
Maximum combinational path delay: 4.915ns

APPENDIX A3 Device Utilization Summary of XC2V4000

Device utilization summary:

Selected Device : 2v4000bf957-6

Number of Slices:	*16916*	*out of 23040*	*73%*
Number of Slice Flip Flops:	*1221*	*out of 46080*	*2%*
Number of 4 input LUTs:	*15934*	*out of 46080*	*34%*
Number of bonded IOBs:	*220*	*out of 684*	*32%*
Number of MULT18X18s:	*2*	*out of 20*	*10%*
Number of GCLKs:	*1*	*out of 16*	*6%*

APPENDIX A4 Timing Report of Device XC2V4000

TIMING REPORT

Clock Information:

Clock Signal	*Clock buffer (FF name)*	*Load*
clk	*BUFGP*	*3269*

Timing Summary:

Speed Grade: -6

Minimum period: 22.062ns (Maximum Frequency: 45.327MHz)
Minimum input arrival time before clock: 7.879ns
Maximum output required time after clock: 11.056ns
Maximum combinational path delay: 4.915ns

APPENDIX B : MATLAB PROGRAMS

APPENDIX B1: MAIN PROCESSING PROGRAM

```
%L1A_STD.m - Zhengwei LI
%find the threshholds of distance in feature space (L1 distance)
%by applying samples and controls to the experiment

function [] = L1_STD(image_src, image_src1, image_src2)


%size of ROI
ROI = 5;
%number of gray levels the image ROI reduced to
gray_bins = 4;
%descriptor: distance
distance = 1;



%read samples inx1=imread('ca1.jpg');
a1=imread('a1.jpg');
a1=rgb2gray(a1);
a2=imread('a2.jpg');
a2=rgb2gray(a2);
a3=imread('a3.jpg');
a3=rgb2gray(a3);
a4=imread('a4.jpg');
a4=rgb2gray(a4);
a5=imread('a5.jpg');
a5=rgb2gray(a5);
a6=imread('a6.jpg');
a6=rgb2gray(a6);
a7=imread('a7.jpg');
a7=rgb2gray(a7);
a8=imread('a8.jpg');
a8=rgb2gray(a8);
a9=imread('a9.jpg');
a9=rgb2gray(a9);
a1_1=imread('a1_1.jpg');
a1_1=rgb2gray(a1_1);
a2_1=imread('a2_1.jpg');
a2_1=rgb2gray(a2_1);
a3_1=imread('a3_1.jpg');
a3_1=rgb2gray(a3_1);
```

```
a4_1=imread('a4_1.jpg');
a4_1=rgb2gray(a4_1);
a5_1=imread('a5_1.jpg');
a5_1=rgb2gray(a5_1);
a6_1=imread('a6_1.jpg');
a6_1=rgb2gray(a6_1);
a7_1=imread('a7_1.jpg');
a7_1=rgb2gray(a7_1);
a8_1=imread('a8_1.jpg');
a8_1=rgb2gray(a8_1);
a9_1=imread('a9_1.jpg');
a9_1=rgb2gray(a9_1);

a1_2=imread('a1_2.jpg');
a1_2=rgb2gray(a1_2);
a2_2=imread('a2_2.jpg');
a2_2=rgb2gray(a2_2);
a3_2=imread('a3_2.jpg');
a3_2=rgb2gray(a3_2);
a4_2=imread('a4_2.jpg');
a4_2=rgb2gray(a4_2);
a5_2=imread('a5_2.jpg');
a5_2=rgb2gray(a5_2);
a6_2=imread('a6_2.jpg');
a6_2=rgb2gray(a6_2);
a7_2=imread('a7_2.jpg');
a7_2=rgb2gray(a7_2);
a8_2=imread('a8_2.jpg');
a8_2=rgb2gray(a8_2);
a9_2=imread('a9_2.jpg');
a9_2=rgb2gray(a9_2);

S=[a1, a1_1, a1_2, a2, a2_1, a2_2, a3, a3_1, a3_2; a4, a4_1, a4_2, a5, a5_1, a5_2, a6, a6_1,
a6_2; a7, a7_1, a7_2, a8, a8_1, a8_2, a9, a9_1, a9_2];
%S = imread(sam_src1);
imshow(S,[]);
SAM_big = S;
[maxrow, maxcol] = size(SAM_big);
sample_num = 0;
for i = 1 : ROI : maxrow - ROI +1
  for j = 1 : ROI + ROI + ROI : maxcol - ROI - ROI - ROI + 1
    sample_num = sample_num + 1;
    SAM(:, :, sample_num) = SAM_big(i:i+ROI-1, j:j+ROI+ROI+ROI-1);
    SAMa(:, :, sample_num) = SAM_big(i:i+ROI-1, j:j+ROI-1);
    SAMb(:, :, sample_num) = SAM_big(i:i+ROI-1, j+ROI:j+ROI+ROI-1);
    SAMc(:, :, sample_num) = SAM_big(i:i+ROI-1, j+ROI+ROI:j+ROI+ROI+ROI-1);
```

```
  end
end

%reduced sample ROIs
SAM_RE = zeros(ROI, ROI+ROI+ROI, sample_num);
%cooccurrence matrices computed from sample ROIs
SAM_CO_M = zeros(gray_bins, gray_bins, sample_num);
%average cooccurrence matrix for samples
REP_M = zeros(gray_bins, gray_bins);

%computation of each sample's cooccurrence matrix
T_CORRELATION = 0;
for i = 1:sample_num
  %A_Correlation(i) = Correlation_I(SAM(:, :, i));
  %T_CORRELATION = T_CORRELATION + A_Correlation(i);

  T_R1 = Correlation_I([CooccurrenceM(ReduceImage4(SAMa(:, :, i)), gray_bins,
 distance), CooccurrenceM(ReduceImage4(SAMb(:, :, i)),gray_bins, distance)]);
  T_R2 = Correlation_I([CooccurrenceM(ReduceImage4(SAMa(:, :, i)), gray_bins,
 distance), CooccurrenceM(ReduceImage4(SAMc(:, :, i)),gray_bins, distance)]);
  T_R(i) = (T_R1 + T_R2)/2;
  T_CORRELATION = T_CORRELATION + T_R(i);
  SAM_RE(:, :, i) = ReduceImage4(SAM(:, :, i));
  SAM_CO_M(:, :, i) = CooccurrenceM(SAM_RE(:, :, i), gray_bins, distance);
end

%computation of average cooccurrence matrix for Class A (ca) samples
for i = 1:gray_bins
  for j = 1:gray_bins
    for k = 1:sample_num
      REP_M(i, j) = REP_M(i, j) + SAM_CO_M(i, j, k);
    end
    REP_M(i, j) = REP_M(i, j)/sample_num;
  end
end
display (REP_M);
%computation of the typical feature vector from
%the average cooccurrence matrix
T_ENTROPY = Entropy(REP_M);
waterfall(Entropy(REP_M));
T_ENERGY = Energy(REP_M);
T_CONTRAST = Contrast(REP_M);
T_I_D_M = I_D_M(REP_M);
T_CORRELATION = T_CORRELATION/sample_num;
T_MAX_P = Max_P(REP_M);
```

```
b1=imread('b1.jpg');
b1=rgb2gray(b1);
b2=imread('b2.jpg');
b2=rgb2gray(b2);
b3=imread('b3.jpg');
b3=rgb2gray(b3);
b4=imread('b4.jpg');
b4=rgb2gray(b4);
b5=imread('b5.jpg');
b5=rgb2gray(b5);
b6=imread('b6.jpg');
b6=rgb2gray(b6);
b7=imread('b7.jpg');
b7=rgb2gray(b7);
b8=imread('b8.jpg');
b8=rgb2gray(b8);
b9=imread('b9.jpg');
b9=rgb2gray(b9);
b1_1=imread('b1_1.jpg');
b1_1=rgb2gray(b1_1);
b2_1=imread('b2_1.jpg');
b2_1=rgb2gray(b2_1);
b3_1=imread('b3_1.jpg');
b3_1=rgb2gray(b3_1);
b4_1=imread('b4_1.jpg');
b4_1=rgb2gray(b4_1);
b5_1=imread('b5_1.jpg');
b5_1=rgb2gray(b5_1);
b6_1=imread('b6_1.jpg');
b6_1=rgb2gray(b6_1);
b7_1=imread('b7_1.jpg');
b7_1=rgb2gray(b7_1);
b8_1=imread('b8_1.jpg');
b8_1=rgb2gray(b8_1);
b9_1=imread('b9_1.jpg');
b9_1=rgb2gray(b9_1);

b1_2=imread('b1_2.jpg');
b1_2=rgb2gray(b1_2);
b2_2=imread('b2_2.jpg');
b2_2=rgb2gray(b2_2);
b3_2=imread('b3_2.jpg');
b3_2=rgb2gray(b3_2);
b4_2=imread('b4_2.jpg');
b4_2=rgb2gray(b4_2);
b5_2=imread('b5_2.jpg');
```

```
b5_2=rgb2gray(b5_2);
b6_2=imread('b6_2.jpg');
b6_2=rgb2gray(b6_2);
b7_2=imread('b7_2.jpg');
b7_2=rgb2gray(b7_2);
b8_2=imread('b8_2.jpg');
b8_2=rgb2gray(b8_2);
b9_2=imread('b9_2.jpg');
b9_2=rgb2gray(b9_2);
cb=[b1, b1_1, b1_2, b2, b2_1, b2_2, b3, b3_1, b3_2; b4, b4_1, b4_2, b5, b5_1, b5_2, b6,
b6_1, b6_2; b7, b7_1, b7_2, b8, b8_1, b8_2, b9, b9_1, b9_2];
figure;
imshow(cb,[]);


%read controls in
%C = imread(sam_src2);
%CON_big = rgb2gray(C);
CON_big = cb;

[maxrow, maxcol] = size(CON_big);
control_num = 0;
for i = 1:ROI:maxrow - ROI+1
  for j = 1:ROI+ROI+ROI:maxcol - ROI - ROI - ROI + 1
    control_num = control_num + 1;
    CON(:, :, control_num) = CON_big(i:i+ROI-1, j:j+ROI+ROI+ROI-1);
    CONa(:, :, control_num) = CON_big(i:i+ROI-1, j:j+ROI-1);
    CONb(:, :, control_num) = CON_big(i:i+ROI-1, j+ROI:j+ROI+ROI-1);
    CONb(:, :, control_num) = CON_big(i:i+ROI-1, j+ROI+ROI:j+ROI+ROI+ROI-1);
  end
end

%reduced control ROIs
CON_RE = zeros(ROI, ROI+ROI+ROI, control_num);
%cooccurrence matrices computed from control ROIs
CON_CO_M = zeros(gray_bins, gray_bins, control_num);
CON_M = zeros(gray_bins, gray_bins);

%computation of each sample's cooccurrence matrix
CON_CORRELATION = 0;
for i = 1:control_num
  %B_Correlation(i) = Correlation_I(CON(:, :, i));
  %CON_CORRELATION = CON_CORRELATION + B_Correlation(i);
  CON_R1 = Correlation_I([CooccurrenceM(ReduceImage4(CONa(:, :, i)), gray_bins,
distance), CooccurrenceM(ReduceImage4(CONb(:, :, i)),gray_bins, distance)]);
```

```
  CON_R2 = Correlation_I([CooccurrenceM(ReduceImage4(CONa(:, :, i)), gray_bins,
distance), CooccurrenceM(ReduceImage4(CONb(:, :, i)),gray_bins, distance)]);
  CON_R(i) = (CON_R1 + CON_R2)/2;
  CON_CORRELATION = CON_CORRELATION + CON_R(i);
  CON_RE(:, :, i) = ReduceImage4(CON(:, :, i));
  CON_CO_M(:, :, i) = CooccurrenceM(CON_RE(:, :, i), gray_bins, distance);
end
%computation of average cooccurrence matrix for Class B (cb) samples
for i = 1:gray_bins
  for j = 1:gray_bins
    for k = 1:control_num
      CON_M(i, j) = CON_M(i, j) + CON_CO_M(i, j, k);
    end
    CON_M(i, j) = CON_M(i, j)/control_num;
  end
end
display (CON_M);
%computation of the typical feature vector of cb from
%the average cooccurrence matrix
CON_ENTROPY = Entropy(CON_M);
CON_ENERGY = Energy(CON_M);
CON_CONTRAST = Contrast(CON_M);
CON_I_D_M = I_D_M(CON_M);
CON_CORRELATION = CON_CORRELATION/control_num;
CON_MAX_P = Max_P(CON_M);

c1=imread('c1.jpg');
c1=rgb2gray(c1);
c2=imread('c2.jpg');
c2=rgb2gray(c2);
c3=imread('c3.jpg');
c3=rgb2gray(c3);
c4=imread('c4.jpg');
c4=rgb2gray(c4);
c5=imread('c5.jpg');
c5=rgb2gray(c5);
c6=imread('c6.jpg');
c6=rgb2gray(c6);
c7=imread('c7.jpg');
c7=rgb2gray(c7);
c8=imread('c8.jpg');
c8=rgb2gray(c8);
c9=imread('c9.jpg');
c9=rgb2gray(c9);
c1_1=imread('c1_1.jpg');
c1_1=rgb2gray(c1_1);
```

```
c2_1=imread('c2_1.jpg');
c2_1=rgb2gray(c2_1);
c3_1=imread('c3_1.jpg');
c3_1=rgb2gray(c3_1);
c4_1=imread('c4_1.jpg');
c4_1=rgb2gray(c4_1);
c5_1=imread('c5_1.jpg');
c5_1=rgb2gray(c5_1);
c6_1=imread('c6_1.jpg');
c6_1=rgb2gray(c6_1);
c7_1=imread('c7_1.jpg');
c7_1=rgb2gray(c7_1);
c8_1=imread('c8_1.jpg');
c8_1=rgb2gray(c8_1);
c9_1=imread('c9_1.jpg');
c9_1=rgb2gray(c9_1);

c1_2=imread('c1_2.jpg');
c1_2=rgb2gray(c1_2);
c2_2=imread('c2_2.jpg');
c2_2=rgb2gray(c2_2);
c3_2=imread('c3_2.jpg');
c3_2=rgb2gray(c3_2);
c4_2=imread('c4_2.jpg');
c4_2=rgb2gray(c4_2);
c5_2=imread('c5_2.jpg');
c5_2=rgb2gray(c5_2);
c6_2=imread('c6_2.jpg');
c6_2=rgb2gray(c6_2);
c7_2=imread('c7_2.jpg');
c7_2=rgb2gray(c7_2);
c8_2=imread('c8_2.jpg');
c8_2=rgb2gray(c8_2);
c9_2=imread('c9_2.jpg');
c9_2=rgb2gray(c9_2);

cc=[c1, c1_1, c1_2, c2, c2_1, c2_2, c3, c3_1, c3_2; c4, c4_1, c4_2, c5, c5_1, c5_2, c6,
c6_1, c6_2; c7, c7_1, c7_2, c8, c8_1, c8_2, c9, c9_1, c9_2];
figure;
imshow(cc,[]);
%subplot(3,1,1);imshow(S)
%subplot(3,1,2);imshow(cb)
%subplot(3,1,3);imshow(cc)
%read classC in
%CC = imread(sam_src2);
%CON_big = rgb2gray(C);
```

```
CC_big = cc;

[maxrow, maxcol] = size(CC_big);
cc_num = 0;
for i = 1:ROI:maxrow - ROI+1
  for j = 1:ROI+ROI+ROI:maxcol - ROI - ROI - ROI+1
    cc_num = cc_num + 1;
    CC(:, :, cc_num) = CC_big(i:i+ROI-1, j:j+ROI+ROI+ROI-1);
    CCa(:, :, cc_num) = CC_big(i:i+ROI-1, j:j+ROI-1);
    CCb(:, :, cc_num) = CC_big(i:i+ROI-1, j+ROI:j+ROI+ROI-1);
    CCc(:, :, cc_num) = CC_big(i:i+ROI-1, j+ROI+ROI:j+ROI+ROI+ROI-1);
  end
end

%reduced cc ROIs
CC_RE = zeros(ROI, ROI+ROI+ROI, cc_num);
%cooccurrence matrices computed from control ROIs
CC_CO_M = zeros(gray_bins, gray_bins, cc_num);
CC_M = zeros(gray_bins, gray_bins);

%computation of each sample's cooccurrence matrix
CC_CORRELATION = 0;
for i = 1:cc_num
  %C_Correlation(i) = Correlation_I(CC(:, :, i));
  %CC_CORRELATION = CC_CORRELATION + C_Correlation(i);
  CC_R1 = Correlation_I([CooccurrenceM(ReduceImage4(CCa(:, :, i)), gray_bins,
distance), CooccurrenceM(ReduceImage4(CCb(:, :, i)),gray_bins, distance)]);
  CC_R2 = Correlation_I([CooccurrenceM(ReduceImage4(CCa(:, :, i)), gray_bins,
distance), CooccurrenceM(ReduceImage4(CCc(:, :, i)),gray_bins, distance)]);
  CC_R(i) = (CC_R1 + CC_R2)/2;
  CC_CORRELATION = CC_CORRELATION + CC_R(i);
  CC_RE(:, :, i) = ReduceImage4(CC(:, :, i));
  CC_CO_M(:, :, i) = CooccurrenceM(CC_RE(:, :, i), gray_bins, distance);
end
%computation of average cooccurrence matrix for Class C (cc) samples
for i = 1:gray_bins
  for j = 1:gray_bins
    for k = 1:cc_num
      CC_M(i, j) = CC_M(i, j) + CC_CO_M(i, j, k);
    end
    CC_M(i, j) = CC_M(i, j)/cc_num;
  end
end
display (CC_M);
%computation of the typical feature vector from
%the average cooccurrence matrix
```

```
CC_ENTROPY = Entropy(CC_M);
CC_ENERGY = Energy(CC_M);
CC_CONTRAST = Contrast(CC_M);
CC_I_D_M = I_D_M(CC_M);
CC_CORRELATION = CC_CORRELATION/cc_num;
CC_MAX_P = Max_P(CC_M);

%output the typical feature vector
CA_REP = {'sample number', sample_num;
          'ENTROPY = ', T_ENTROPY;
          'ENERGY = ', T_ENERGY;
          'CONTRAST = ', T_CONTRAST;
          'IDM = ', T_I_D_M;
          'CORRELATION = ', T_CORRELATION;
          'MAX PROBABILITY = ', T_MAX_P
         }
display(CA_REP);

CB_REP = {'CB sample number', control_num;
       'ENTROPY = ', CON_ENTROPY;
       'ENERGY  = ', CON_ENERGY;
       'CONTRAST= ', CON_CONTRAST;
       'IDM     = ', CON_I_D_M;
       'CORRELATION=', CON_CORRELATION;
       'MAX PROBABILITY= ', CON_MAX_P
     }
display(CB_REP);
CC_REP = {'CC sample number', cc_num;
          'ENTROPY = ', CC_ENTROPY;
          'ENERGY = ', CC_ENERGY;
          'CONTRAST = ', CC_CONTRAST;
          'IDM = ', CC_I_D_M;
          'CORRELATION = ', CC_CORRELATION;
          'MAX PROBABILITY = ', CC_MAX_P
         }
display(CC_REP);

%SEG
%read in the original image
IMAGE = imread(image_src);
IMAGE1 = imread(image_src1);
IMAGE2 = imread(image_src2);
IMAGE = rgb2gray(IMAGE);
IMAGE1 = rgb2gray(IMAGE1);
IMAGE2 = rgb2gray(IMAGE2);
[maxrow, maxcol] = size(IMAGE);
```

```
SEGEDIMAGE=zeros(maxrow, maxcol);

%probability map
PM = zeros(maxrow, maxcol);
PM_all = zeros(maxrow, maxcol);

%maximum L1 and minimum L1
maxL1 = 0;
minL1 = 0;

c=ROI/2;



%calculate the probability map based on L1 distance
%the original image is scanned by size of ROI
for i = 1 : 1:maxrow-ROI+1
  for j = 1 : 1: maxcol - ROI+1

      %calculate ROI's cooccurrence matrix
      IMAGE_ROIa = IMAGE(i:i + ROI-1, j:j + ROI-1);
      IMAGE_ROIb = IMAGE1(i:i + ROI-1, j:j + ROI-1);
      IMAGE_ROIc = IMAGE2(i:i + ROI-1, j:j + ROI-1);
      IMAGE_ROI = [IMAGE_ROIa, IMAGE_ROIb, IMAGE_ROIc];

IMAGE_ROI_AVERAGE_GRAYLEVEL=AVERAGE_GRAYLEVEL(IMAGE_ROI);

IMAGE_ROI_AVERAGE_GRAYLEVEL1=AVERAGE_GRAYLEVEL(IMAGE_ROIb);
      if(IMAGE_ROI_AVERAGE_GRAYLEVEL<15)
        SEGEDIMAGE(i:i+ROI-1, j:j+ROI-1)=1;
      elseif(IMAGE_ROI_AVERAGE_GRAYLEVEL1<61)
        SEGEDIMAGE(i:i+ROI-1, j:j+ROI-1)=1;
      else

      Ma = CooccurrenceM(ReduceImage4(IMAGE_ROIa), gray_bins, distance);
      Mb = CooccurrenceM(ReduceImage4(IMAGE_ROIb), gray_bins, distance);
      Mc = CooccurrenceM(ReduceImage4(IMAGE_ROIc), gray_bins, distance);
      Mab =[Ma, Mb];
      Mac =[Ma, Mc];
      IMAGE_ROI_CORRELATION1 = Correlation_I(Mab);
      IMAGE_ROI_CORRELATION2 = Correlation_I(Mac);
      IMAGE_ROI_CORRELATION = (IMAGE_ROI_CORRELATION1 +
       IMAGE_ROI_CORRELATION2)/2;

      IMAGE_ROI_RE = ReduceImage4(IMAGE_ROI);
      IMAGE_ROI_CO_M = CooccurrenceM(IMAGE_ROI_RE, gray_bins, distance);
```

```
%calculate ROI's feature vector
IMAGE_ROI_ENTROPY = Entropy(IMAGE_ROI_CO_M);
IMAGE_ROI_ENERGY = Energy(IMAGE_ROI_CO_M);
IMAGE_ROI_CONTRAST = Contrast(IMAGE_ROI_CO_M);
IMAGE_ROI_I_D_M = I_D_M(IMAGE_ROI_CO_M);
IMAGE_ROI_MAX_P = MAX_P(IMAGE_ROI_CO_M);

%calculate ROI's L1 distance
I_tA = abs(IMAGE_ROI_ENTROPY-T_ENTROPY) +
abs(IMAGE_ROI_ENERGY-T_ENERGY) + abs(IMAGE_ROI_CONTRAST-
T_CONTRAST) + abs(IMAGE_ROI_I_D_M-T_I_D_M) + abs(IMAGE_ROI_MAX_P-
T_MAX_P) + abs(IMAGE_ROI_CORRELATION-T_CORRELATION);
IA = IMAGE_ROI_ENTROPY + IMAGE_ROI_ENERGY +
IMAGE_ROI_CONTRAST + IMAGE_ROI_I_D_M + IMAGE_ROI_MAX_P +
IMAGE_ROI_CORRELATION;
tA = T_ENTROPY + T_ENERGY + T_CONTRAST + T_I_D_M + T_MAX_P +
T_CORRELATION;
IMAGE_ROI_L1A = I_tA/(IA + tA);

I_tB = abs(IMAGE_ROI_ENTROPY-CON_ENTROPY) +
abs(IMAGE_ROI_ENERGY-CON_ENERGY) + abs(IMAGE_ROI_CONTRAST-
CON_CONTRAST) + abs(IMAGE_ROI_I_D_M-CON_I_D_M) +
abs(IMAGE_ROI_MAX_P-CON_MAX_P) + abs(IMAGE_ROI_CORRELATION-
CON_CORRELATION);
IB = IMAGE_ROI_ENTROPY + IMAGE_ROI_ENERGY +
IMAGE_ROI_CONTRAST + IMAGE_ROI_I_D_M + IMAGE_ROI_MAX_P +
IMAGE_ROI_CORRELATION;
tB = CON_ENTROPY + CON_ENERGY + CON_CONTRAST + CON_I_D_M +
CON_MAX_P + CON_CORRELATION;
IMAGE_ROI_L1B = I_tB/(IB + tB);


I_tC = abs(IMAGE_ROI_ENTROPY-CC_ENTROPY) +
abs(IMAGE_ROI_ENERGY-CC_ENERGY) + abs(IMAGE_ROI_CONTRAST-
CC_CONTRAST) + abs(IMAGE_ROI_I_D_M-CC_I_D_M) +
abs(IMAGE_ROI_MAX_P-CC_MAX_P) + abs(IMAGE_ROI_CORRELATION-
CC_CORRELATION);
IC = IMAGE_ROI_ENTROPY + IMAGE_ROI_ENERGY +
IMAGE_ROI_CONTRAST + IMAGE_ROI_I_D_M + IMAGE_ROI_MAX_P +
IMAGE_ROI_CORRELATION;
tC = CC_ENTROPY + CC_ENERGY + CC_CONTRAST + CC_I_D_M +
CC_MAX_P + CC_CORRELATION;
IMAGE_ROI_L1C = I_tC/(IC + tC);

IMAGE_ROI_L1=IMAGE_ROI_L1A;
```

```
      if IMAGE_ROI_L1>IMAGE_ROI_L1B
        IMAGE_ROI_L1=IMAGE_ROI_L1B;
      end
      if IMAGE_ROI_L1>IMAGE_ROI_L1C
        IMAGE_ROI_L1=IMAGE_ROI_L1C;
      end
      if IMAGE_ROI_L1==IMAGE_ROI_L1A
        SEGEDIMAGE(i:i+ROI-1, j:j+ROI-1)=4;
      end
      if IMAGE_ROI_L1==IMAGE_ROI_L1B
        SEGEDIMAGE(i:i+ROI-1, j:j+ROI-1)=2;
      end
      if IMAGE_ROI_L1==IMAGE_ROI_L1C
        SEGEDIMAGE(i:i+ROI-1, j:j+ROI-1)=3;
      end

    end
      %mark probability label in full range, not regarding threshhold
    %PM_all(i+c, j+c) = IMAGE_ROI_L1;
    %if IMAGE_ROI_L1 > maxL1
    %  maxL1 = IMAGE_ROI_L1;
    %end
    %if minL1 > IMAGE_ROI_L1
    %   minL1 = IMAGE_ROI_L1;
    %end



    %mark probability label to the center of the ROI according to threshhold
    % if L1_low >= IMAGE_ROI_L1
    %    PM(i + c, j + c) = L1_low;
    % elseif L1_high > IMAGE_ROI_L1 > L1_low
    %    PM(i + c, j + c) = IMAGE_ROI_L1;
    % else
    %    PM(i + c, j + c) = L1_high;
    % end


  end
end

error_size = 4;
for i = 1 : 1:maxrow
  for j = error_size+1 : 1: maxcol-error_size
    if(((SEGEDIMAGE(i, j)~=1)&SEGEDIMAGE(i, j-
error_size)==1)&(SEGEDIMAGE(i, j+error_size)==1))
```

```
        SEGEDIMAGE(i, j)=1;
      end
    end
end

figure;
imshow(SEGEDIMAGE, []);
%display(SEGEDIMAGE);
%convert the probability map to 256 level gray image
%M = find(PM == 0);
%PM(M) = L1_high;
%display(PM);
%IMAGE_PM = mat2gray(PM, [L1_high, L1_low]);
%imshow(IMAGE_PM, []);


%M = find(PM_all == 0);
%PM_all(M) = maxL1;
%display(PM_all);
%IMAGE_PM_all = mat2gray(PM_all, [maxL1, minL1]);
%imshow(IMAGE_PM_all, []);
```

APPENDIX B2: PROGRAM FOR CO-OCCURRENCE CALCUALTION

```
% CooccurrenceM.m - Zhengwei LI
% calculates the cooccurence matrix, CM, of an image, i
% d is the distance of the i pixel from the j pixel, and should be a two element vector, such as [1,1]

function [CM] = CooccurrenceM(image, gray_bins, d)

image = double(image);
[rows, cols] = size(image);

%choose which type(s) to proceed
a1 = 1;  % --0
a2 = 0;  % --180
a3 = 0;  % --45
a4 = 0;  % --225
a5 = 0;  % --90
a6 = 0;  % --270
a7 = 0;  % --135
a8 = 0;  % --315


% Initialize P to be all zeros
CM = zeros(gray_bins, gray_bins);


%0 and 180
if (a1 == 1)
 for row = 1:rows - d
  for col = 1:cols - d
    i = image(row, col);
    j = image(row + d, col + d);
    CM(i,j) = CM(i,j) + 1;
    if (a2 == 1)
       CM(j,i) = CM(j,i) + 1;
    end
  end
 end
end


%45 and 225
if (a3 == 1)
```

```
for row = (1 + d):rows
  for col = 1:(cols - d)
     i = image(row, col);
     j = image(row - d, col + d);
     CM(i,j) = CM(i,j) + 1;
     if (a4 == 1)
        CM(j,i) = CM(j,i) + 1;
     end
  end
 end
end

%90 and 270
if (a5 == 1)
 for row = (1 + d):rows
  for col = 1:cols
     i = image(row, col);
     j = image(row - d, col);
     CM(i,j) = CM(i,j) + 1;
     if (a6 == 1)
        CM(j,i) = CM(j,i) + 1;
     end
  end
 end
end

%135 and 315
if (a7 == 1)
 for row = (1 + d):rows
  for col = (1 + d):cols
     i = image(row, col);
     j = image(row - d, col - d);
     CM(i,j) = CM(i,j) + 1;
     if (a8 == 1)
        CM(j,i) = CM(j,i) + 1;
     end
  end
 end
end
```

APPENDIX B3: PROGRAM FOR IMAGE REDUCTION

```
%ReduceImage4.m  - Zhengwei LI
%convert the input image to 4 gray level grayscale image
%and shift all of the levels up by one to enable the gray
%level to represent an index into the co-occurrence matrix
%(index of a matrix can't be 0).

function y = ReduceImage(x)

x = double(x);
x = mat2gray(x);      % scale
x = grayslice(x,4);   % reduce the number of gray levels to eight


a = find(x==0);       % find the levels
b = find(x==1);
c = find(x==2);
d = find(x==3);

x(a) = 1;             % shift them by one
x(b) = 2;
x(c) = 3;
x(d) = 4;


y =(x);
```

APPENDIX B4: PROGRAM FOR ENTROPY CALCULATION

```
% Entropy.m - Zhengwei LI
% calculates the entropy given a co-occurrence matrix, m

function [Entropy] = Entropy(M)

[rows,cols] = size(M);

Entropy = 0;

for i = 1:rows
  for j = 1:cols
    if M(i,j) ~= 0
      Entropy = Entropy + M(i,j) * log2(M(i,j));
```

```
    end
  end
end
```

APPENDIX B5: PROGRAM FOR ENERGY CALCULATION

```
% Energy.m - Zhengwei LI
% calculates the energy given a co-occurrence matrix, M

function [Energy] = Energy(M)

[rows,cols] = size(M);

Energy = 0;

for i = 1:rows
  for j = 1:cols
    Energy = Energy + M(i,j) ^ 2;
  end
end
```

APPENDIX B6: PROGRAM FOR CONTRAST CALCULATION

```
% Contrast.m - Zhengwei LI
% calculates the contrast given a co-occurrence matrix, P

function [Contrast] = Contrast(M)

[rows,cols] = size(M);

Contrast = 0;

for i = 1:rows
  for j = 1:cols
    if M(i,j) ~= 0
      Contrast = Contrast + (abs(i - j))^2 * M(i,j);
    end
  end
end
```

APPENDIX B7: PROGRAM FOR INVERSE DIFFERENCE MOMENT CALCULATION

```
% I_D_M.m - Zhengwei LI
% calculates the inverse difference moment given a co-occurrence matrix, M

function [I_D_M] = I_D_M(M)

[rows, cols] = size(M);

I_D_M = 0;

for i = 1:rows
  for j = 1:cols
    if i ~= j
      if M(i,j) ~= 0
        I_D_M = I_D_M + (M(i,j)) / ((abs(i - j)) ^ 2);
      end
    end
  end
end
```

APPENDIX B8: SUBPROGRAM FOR MAXIMUM PROBABILITY CALCULATION

```
% Max_P.m - Zhengwei LI
% calculates the maximum probability given a co-occurrence matrix, M

function [Max_P] = Max_P(M)

[rows,cols] = size(M);

Max_P = 0;

for i = 1:rows
  for j = 1:cols
    Max_P = max(M(i, j), Max_P);
  end
end
```

APPENDIX B9: PROGRAM FOR CORRELATION CALCULATION

```
% Correlation_I.m - Zhengwei LI
% calculates the correlation of pixels in two images. This feature vector
% can be calculated from the image itself. no need for cooccurrence matrix
% here. illustrated as follow:


function [R] = Correlation_I(I)

I = double(I);
I = mat2gray(I);      % scale
[rows,cols] = size(I);

%initilize variables in the correlation function
R = 0;
x = 0;
y = 0;
Sxy = 0;
Sx = 0;
Sy = 0;
Sx2 = 0;
Sy2 = 0;
N = rows * (cols/2);

%calculate the correlation
for i = 1:rows
  for j = 1:(cols/2)

     x = I(i, j);
     y = I(i, j + (cols/2));
     Sxy = Sxy + x * y;
     Sx = Sx + x;
     Sy = Sy + y;
     Sx2 = Sx2 + x * x;
     Sy2 = Sy2 + y * y;

  end
end

R = (N * Sxy - Sx * Sy)/sqrt((N * Sx2 - Sx * Sx) * (N * Sy2 - Sy * Sy));
```

APPENDIX B10: PROGRAM FOR CONTRAST CALCULATION

```
% Contrast.m - Zhengwei LI
% calculates the contrast given a co-occurrence matrix, P

function [Contrast] = Contrast(M)

[rows,cols] = size(M);

Contrast = 0;

for i = 1:rows
  for j = 1:cols
    if M(i,j) ~= 0
      Contrast = Contrast + (abs(i - j))^2 * M(i,j);
    end
  end
end
```

APPENDIX C: PROGRAM TO TRANSFORM IMAGE DATA INTO DATA FILE FOR VHDL TESTBENCH

```
function [] = getorigout(image_src, image_src1, image_src2)


%size of ROI
ROI = 5;
%number of gray levels the image ROI reduced to
gray_bins = 4;
%descriptor: distance
distance = 1;

%read in the original image
IMAGE = imread(image_src);
IMAGE1 = imread(image_src1);
IMAGE2 = imread(image_src2);
IMAGE = rgb2gray(IMAGE);
IMAGE1 = rgb2gray(IMAGE1);
IMAGE2 = rgb2gray(IMAGE2);
[maxrow, maxcol] = size(IMAGE);

%probability map
PM = zeros(maxrow, maxcol);
```

```
PM_all = zeros(maxrow, maxcol);

%maximum L1 and minimum L1
maxL1 = 0;
minL1 = 0;

c=ROI/2;


fid = fopen('d:\mri\mri-three-imagebased\sample1\orig5x5.bin','w');
%calculate the probability map based on L1 distance
%the original image is scanned by size of ROI
for i = 1 : 1:maxrow-ROI+1
  for j = 1 : 1: maxcol - ROI+1

      %calculate ROI's cooccurrence matrix
      IMAGE_ROIa = IMAGE(i:i + ROI-1, j:j + ROI-1);
      IMAGE_ROIb = IMAGE1(i:i + ROI-1, j:j + ROI-1);
      IMAGE_ROIc = IMAGE2(i:i + ROI-1, j:j + ROI-1);
      IMAGE_ROI = [IMAGE_ROIa, IMAGE_ROIb, IMAGE_ROIc];
      tempBlock = reshape(double(IMAGE_ROI'), 75, 1);
      fprintf(fid,'%d\n', tempBlock);


  end
end

fclose(fid);
```

APPENDIX D: PROGRAM TO DISPLAY VHDL TESTBENCH RESULT

```
fid=fopen('d:\MRI\Mri-Three-imageBased\sample1\vhdl_output521_2.bin', 'r');
InputLabel = fscanf(fid, '%4d');
InputLabel
fclose(fid);
L = size(InputLabel);
L(1)
TempImg = zeros(63504, 1);
TempImg = TempImg + 4;

count = 1;
%for i = 1 : 1 : 65536
for i = 1 : 2 : L(1)
```

```
   TempImg(count, 1) = InputLabel(i);
   count = count + 1;
end

%fid = fopen('d:\mri\mri-three-imagebased\sample1\vhdl5x5.bin','w');
%fprintf(fid,'%d\n',TempImg(1:L(1), 1));
%fclose(fid);


L = size(TempImg);

L(1)

OutImg = reshape(TempImg, 252, 252);

OutImg = OutImg';
figure;
imshow(OutImg, []);
```

APPENDIX E: VHDL TESTBENCH PROGRAM

```
-- VHDL Test Bench Created from source file featurevector.vhd -- 10:29:53 04/10/2005
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test. Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_signed.all;
use std.textio.all;

ENTITY featurevector_newTB_vhd_tb IS
END featurevector_newTB_vhd_tb;

ARCHITECTURE behavior OF featurevector_newTB_vhd_tb IS

    COMPONENT featurevector
    PORT(
        DATAIN : IN std_logic_vector(0 to 1);
        clk : IN std_logic;
        clk1 : IN std_logic;
        count : IN std_logic_vector(0 to 7);
        addr1 : IN std_logic_vector(0 to 7);
        addr2 : IN std_logic_vector(0 to 3);
        dma1 : IN std_logic;
        dma2 : IN std_logic;
        reset : IN std_logic;
        POWON : IN std_logic;
        data_out : INOUT std_logic_vector(0 to 7);
        ENTROPY : INOUT std_logic_vector(0 to 9);
        INTERIA : INOUT std_logic_vector(0 to 7);
        ENERGY : INOUT std_logic_vector(0 to 12);
        CONTRAST : INOUT std_logic_vector(0 to 7);
        IDM : INOUT std_logic_vector(0 to 12);
        MB : INOUT std_logic_vector(0 to 7);
        L1Dist : INOUT std_logic_vector(0 to 1);
        Finish : INOUT std_logic;
```

```
        MCNT : OUT std_logic_vector(0 to 15);
        cnt1sim : OUT std_logic_vector(0 to 7);
        address1sim : OUT std_logic_vector(0 to 7);
        intsim : OUT std_logic_vector(0 to 7);
        address2sim : OUT std_logic_vector(0 to 3);
        address3sim : OUT std_logic_vector(0 to 3);
        addr2sim : OUT std_logic_vector(0 to 3);
        status : OUT std_logic;
        WEsim : OUT std_logic;
        addriisim : OUT std_logic_vector(0 to 15);
        state3 : OUT std_logic;
        wr1sim : OUT std_logic;
        wr2sim : OUT std_logic;
        DataReadySim : OUT std_logic;
        L1DistsSim1 : OUT std_logic_vector(0 to 12);
        L1DistsSim2 : OUT std_logic_vector(0 to 12);
        L1DistsSim3 : OUT std_logic_vector(0 to 12);
        data1 : OUT std_logic_vector(0 to 1);
        data2 : OUT std_logic_vector(0 to 1);
        data3 : OUT std_logic_vector(0 to 3);
        IminusJ2sim : OUT std_logic_vector(0 to 7);
        ImageIndexout : OUT std_logic_vector(0 to 1)
        );
    END COMPONENT;

type testdata_array is array ( 0 to 5160 ) of std_logic_vector (0 to 1);  --1024145
signal data_array : testdata_array;


--      type testdata_array1 is array ( natural range <> ) of std_logic_vector (0 to 1);
--  constant all_test_data1 : testdata_array1 :=
--      (   "01","10","11","00",
--          "11","10","01","10",
--          "11","00","11","10",
--          "11","10","11","00"
--       );
    SIGNAL DATAIN     : std_logic_vector(0 to 1);
    SIGNAL clk        : std_logic;
    SIGNAL MCNT       : std_logic_vector(0 to 15);
    SIGNAL count      : std_logic_vector(0 to 7);
    SIGNAL addr1      : std_logic_vector(0 to 7);
    SIGNAL cnt1sim    : std_logic_vector(0 to 7);
    SIGNAL address1sim : std_logic_vector(0 to 7);
    SIGNAL intsim     : std_logic_vector(0 to 7);
    SIGNAL address2sim : std_logic_vector(0 to 3);
    SIGNAL address3sim : std_logic_vector(0 to 3);
```

```
    SIGNAL addr2       : std_logic_vector(0 to 3);
    SIGNAL addr2sim    : std_logic_vector(0 to 3);
    SIGNAL dma1        : std_logic;
    SIGNAL dma2        : std_logic;
    SIGNAL status      : std_logic;
    SIGNAL state3      : std_logic;
    SIGNAL wr1sim      : std_logic;
    SIGNAL wr2sim      : std_logic;
    SIGNAL clk1        : std_logic;
    Signal WEsim       : std_logic;
    Signal addriisim   : std_logic_vector (15 downto 0);

    SIGNAL DataReadySim : std_logic;
    SIGNAL L1DistsSim1  : std_logic_vector (0 to 12);
    SIGNAL L1DistsSim2  : std_logic_vector (0 to 12);
    SIGNAL L1DistsSim3  : std_logic_vector (0 to 12);
    SIGNAL data1       : std_logic_vector(0 to 1);
    SIGNAL data2       : std_logic_vector(0 to 1);
    SIGNAL data3       : std_logic_vector(0 to 3);
    SIGNAL data_out    : std_logic_vector(0 to 7);
    SIGNAL reset       : std_logic;

    SIGNAL POWON       : std_logic;
    SIGNAL INTERIA,CONTRAST,MB : std_logic_vector(0 to 7);
    SIGNAL ENTROPY     : std_logic_vector(0 to 9);
    SIGNAL ENERGY, IDM :  std_logic_vector(0 to 12);
    SIGNAL IminusJ2sim :  std_logic_vector(0 to 7);
    Signal L1Dist      :  std_logic_vector(0 to 1);
   Signal ImageIndexout:  std_logic_vector(0 to 1);
    SIGNAL ID          : std_logic_vector(0 to 1);
    Signal Finish      :  std_logic;
    Signal holdon      :std_logic;

BEGIN

    uut: featurevector PORT MAP(
        DATAIN     => DATAIN,
        clk        => clk,
        MCNT       => MCNT,
        count      => count,
        addr1      => addr1,
        cnt1sim    => cnt1sim,
        address1sim => address1sim,
        intsim     => intsim,
        address2sim => address2sim,
        address3sim => address3sim,
```

```
        addr2       => addr2,
        addr2sim    => addr2sim,
        dma1        => dma1,
        dma2        => dma2,
        status      => status,
        state3      => state3,
        wr1sim      => wr1sim,
        wr2sim      => wr2sim,
        clk1        => clk1,
        WEsim       => Wesim,
        addriisim   => addriisim,

        DataReadySim => DataReadysim,
        L1DistsSim1  => L1DistsSim1,
        L1DistsSim2  => L1DistsSim2,
        L1DistsSim3  => L1DistsSim3,
        data1       => data1,
        data2       => data2,
        data3       => data3,
        data_out    => data_out,
        reset       => reset,

        POWON       => POWON,
        ENTROPY     => ENTROPY,
        INTERIA     => INTERIA,
        ENERGY      => ENERGY,
        CONTRAST    => CONTRAST,
        IDM         => IDM,
        IminusJ2sim  => IminusJ2sim,
        MB          => MB,
        L1Dist      => L1Dist,
        ImageIndexout=> ImageIndexout,
        Finish      => Finish
    );



-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  variable k:integer;
  BEGIN
      POWON <= '1';
  for j in 0 to 68 loop
```

```
reset <= '1';
 clk<='1';
 wait for 10 ns;
 clk <= not clk;
 wait for 10 ns;
 POWON <= '0';
 reset <= '0';
 dma2 <=  '1';                --write ram by means of dma;
 count<="01001011";
 dma1 <=  '1';
 k:=j*75+1;

 clk<='1';
 wait for 10 ns;
 clk <= not clk;
 wait for 10 ns;

 for i in 0 to 74 loop
 addr1 <= conv_std_logic_vector(i,8);
 clk <= not clk;
 wait for 10 ns;
 clk <= not clk;
 wait for 10 ns;
 DATAIN<=Data_array(i+k);
 end loop;

clk <= not clk;
 wait for 10 ns;
 clk <= not clk;
 wait for 10 ns;

 dma1 <= '0';

 for i in 0 to 75 loop    --15
     clk <= not clk;
     wait for 10 ns;
     clk <= not clk;
     wait for 10 ns;
 end loop;

 dma2 <=  '0';                --write ram by means of dma;

  for i in 0 to 60 loop
     clk <= not clk;
     wait for 10 ns;
     clk <= not clk;
```

```
            wait for 10 ns;
          end loop;

        for i in 0 to 16 loop
            addr2 <= conv_std_logic_vector(i,4);
            clk <= not clk;
            wait for 10 ns;
            clk <= not clk;
            wait for 10 ns;
        end loop;


           clk <= not clk;
            wait for 10 ns;
            clk <= not clk;
            wait for 10 ns;

            clk <= not clk;
            wait for 10 ns;
            clk <= not clk;
            wait for 10 ns;

            clk <= not clk;
            wait for 10 ns;
            clk <= not clk;
            wait for 10 ns;

            clk <= not clk;
            wait for 10 ns;
            clk <= not clk;
            wait for 10 ns;


            clk <= not clk;
            wait for 10 ns;
            clk <= not clk;
            wait for 10 ns;
      end loop;
  END PROCESS;

-- *** End Test Bench - User Defined Section ***
Read_From_File: process(clk1)
       Variable indata_line:line;
        Variable indata:integer;
        variable h    :integer :=0;
        file input_data_file:text open read_mode is "C:/PROJECT/matlab/data/orig2.bin";
```

```
begin
       if rising_edge(clk1) and (h<=1024145) then       --1024145--4762800
          readline(input_data_file,indata_line);
          read(indata_line, indata);
          ID<=conv_std_logic_vector(indata,2);
           data_array(h) <=ID;
           h:=h+1;
            if endfile(input_data_file) then
               report "end of file";
               file_close(input_data_file);
               file_open(input_data_file, "C:/PROJECT/matlab/data/orig2.bin");
          end if;
        end if;
end process;


clock_gen:process
begin
     Clk1<='1';
      wait for 10 ns;
      clk1<='0';
      wait for 10 ns;
end process;




 write_to_file: PROCESS (clk1)
        variable outdata_line: line;
       variable outdata:integer:=0;
        variable holdon:std_logic;
       file output_data_file: text open write_mode is
"C:/PROJECT/matlab/vhdl_output521_2.bin";
  begin
       if (WEsim='1') then
          if rising_edge(clk1) then
                  outdata:=abs(CONV_integer(L1Dist));
                   write(outdata_line, outdata);
                   writeline(output_data_file, outdata_line);
            end if;
        end if;
  end process;

 END;
```

(5) BC-74-158