Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Chao Liu

Entitled 3D EM/MPM MEDICAL IMAGE SEGMENTATION USING AN FPGA EMBEDDED DESIGN IMPLEMENTATION

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

Dr. Lauren Christopher
Chair

Dr. Maher Rizkalla

Dr. Paul Salama

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Lauren Christopher

Approved by: Brian King 07/20/2011
Head of the Graduate Program Date

Graduate School Form 20
(Revised 9/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

3D EM/MPM MEDICAL IMAGE SEGMENTATION USING AN FPGA EMBEDDED DESIGN IMPLEMENTATION

For the degree of Master of Science in Electrical and Computer Engineering

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Chao Liu

Printed Name and Signature of Candidate

07/08/2011

Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

3D EM/MPM MEDICAL IMAGE SEGMENTATION USING AN FPGA EMBEDDED DESIGN IMPLEMENTATION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Chao Liu

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2011

Purdue University

Indianapolis, Indiana

To my family

ACKNOWLEDGMENTS

Foremost, special thanks to my advisor Prof. Lauren Christopher of the Department of Electrical and Computer Engineering, for her support of my study. She helped me on all the time of research and writing of this thesis. I would also like to thank my committee members Prof. Paul Salama and Prof. Maher Rizkalla; and to Prof. Brain King for helping on my thesis writing. Thanks to Yan Sun, Weixu Li and Jim Jirgal who have helped me a lot in my research. For the ultrasound data used to test these system, thanks to Karmanos Cancer Institute.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Liu, Chao. M.S.E.C.E., Purdue University, August 2011. 3D EM/MPM Medical Image Segmentation Using An FPGA Embedded Design Implementation. Major Professor: Lauren Christopher.

This thesis presents a Field Programmable Gate Array (FPGA) based embedded system which is used to achieve high speed segmentation of 3D images. Segmentation is performed using Expectation-Maximization with Maximization of Posterior Marginals (EM/MPM) Bayesian algorithm. In this system, the embedded processor controls a custom circuit which performs the MPM and portions of the EM algorithm. The embedded processor completes the EM algorithm and also controls image data transmission between host computer and on-board memory. The whole system has been implemented on Xilinx Virtex 6 FPGA and achieved over 100 times improvement compared to standard desktop computing hardware.

1. INTRODUCTION

In recent years, three dimensional images are becoming more and more popular due to advanced visualization techniques. In the medical area, computed tomography (CT) and magnetic resonance (MR) are widely used for patient diagnosis. Because it is difficult for doctors to diagnose and make a treatment plan using only 2D images, 3D image diagnostic equipment such as 3D and 4D ultrasound are widely used. For tissues surrounded by layers of different texture in some hidden angle, segmented 3D images in the visualization can improve clinical understanding. Therefore, fast image segmentation is the first step for a good visualization. This thesis describes the acceleration of 3D image segmentation.

Several image segmentation algorithms have been proposed. After comparing different Bayesian segmentation techniques, the Expectation Maximization /Maximization of Posterior Marginals (EM/MPM) algorithm is used because of its improved performance in noisy images [1,2]. The EM/MPM algorithm combines the EM algorithm for parameter estimation with the MPM algorithm for segmentation [3]. This algorithm classifies every pixel in an image by assigning a cost to an incorrect segmentation based on the number of incorrectly classified pixels and iteratively finding the best probabilistic solution which fits the data. However, the processing speed is an issue that must be overcome. An embedded hardware-software system implements the algorithm in order to improve throughput on large medical image volumes.

Multimedia technology is popularized in consumer electronics, therefore we see increased use of image processing systems. New products require greater image capacity, higher image quality, and faster image processing speed. Many image processing systems are implemented on graphic controllers with Digital Signal Processors (DSP) or

PC software. Software instructions in series can not meet the real-time requirement of high speed image processing. Therefore, additional effort is needed to control DSP work flow. For the segmentation algorithm, choosing FPGA hardware improves the speed.

What follows is a review of literature for FPGAs applied to imaging tasks. As reported in [4], the FPGA improved the speed of medical image processing systems. Xinxi Zhang, Yong Li, Jinyang Wang, Yulin Chen [5] designed a high-speed image processing system in their paper. They addressed the need for system integration and high-speed image processing for a vehicle-loaded computer. Their system realizes functions of image acquisition, storing, preprocessing, processing and display by using hardware on a FPGA chip and system software. Using programmable chips, reconfiguration technology, and parallel processing technology, their system has high integration, quick image processing speed; resulting in strong real-time capability. But their system does not include image segmentation.

Standard computing architectures are not well suited for parallel processing and have restricted memory bandwidth for large image volumes. Therefore, a Xilinx Virtex 6 Field Programmable Gate Array (FPGA) is chosen to improve image processing throughput for the segmentation algorithm. Field programmable gate arrays (FPGAs) provide designers with the ability to create hardware circuits quickly. Increases in FPGA configurable logic capacity and decreasing FPGA costs have enabled designers to incorporate FPGAs more readily in their designs.

FPGA vendors have begun providing configurable soft processor cores that can be synthesized onto their products. While FPGAs with soft processor cores provide designers with increased flexibility, such processors typically have degraded performance and energy consumption compared to hard-core processors. Roman Lysecky and Frank Vahid [6] studied the potential of a MicroBlaze soft-core based warp processing system to eliminate the performance and energy overhead of a soft-core processor compared to a hard-core processor. They demonstrate that the soft-core based processor yields performance and energy consumption competitive with hard-core processor, thus expanding the usefulness of soft processor cores on FPGAs to a broader range of applications. Ralf Joost and Ralf Salomon [7] report that by applying specific hardware components attached to a soft-core processor, the FPGA uses both software capabilities and resources for hardware implementation. In our implementation of the EM/MPM, the MicroBlaze soft core processor from Xilinx performs the EM calculation and controls the hardware MPM logic.

A key design criteria for our design is the memory and bus interface. External on-board memory, Double-Data-Rate Three Synchronous Dynamic Random Access Memory(DDR3 SDRAM) is used to store both the image data from the host computer and the segmented image results from MPM.

On the Xilinx prototyping board, the data bus transfers between DDR3 and PC is done using a PCI Express bus. A number of commercial bus technologies for high speed data transfer have been proposed including: Serial Rapid I/O, 10 Gigabit Ethernet and Peripheral Component Interconnect Express (PCI Express). Heidi Frock, Mike Geruso and Mark Wetzel [8] reported a survey of these high speed buses and conclude that PCI Express is the most promising technology for use in systems because it provides the best bandwidth and latency. In addition, a Direct Memory Access (DMA) controller has been applied in our system which provides DMA services to devices on the Processor Local Bus (PLB). This is important because it transfers programmable quantity of data between DDR3 and PCI-Express Endpoint without MicroBlaze processor intervention.

This thesis is composed of 5 chapters. In Chapter 2, the EM/MPM algorithm is introduced. The basic idea of this algorithm is reviewed. How this algorithm is implememted and the reason to choose it in this system is analyzed. The theoretical relationship between EM and MPM algorithm is explained in this chapter, which helps in understanding the whole system design.

Chapter 3 explains the detailed implementation of the whole embedded system. Four major modules named MPM, PCIE, MPMC(NPI), EM are described separately. The EM/MPM implementation is clearly described in this chapter.

Furthermore, Chapter 4 shows the experimental results. Compared with PC software implementation and with previous simulation results using MPM algorithm, we see greater than 100 times speed advantages for this embedded system.

Finally, Chapter 5 concludes, confirming the processing speed improvement of the whole system in the FPGA and outlining future work.

2. 3D EM/MPM ALGORITHM

2.1 Introduction

The EM/MPM algorithm is based on the EM algorithm for parameter estimation and the MPM algorithm for segmentation [3]. In this chapter, system overviews are provided of the EM and MPM algorithm. For a given 3D image, the source image grey level data, denoted as Y, are considered a 3D volume of random variables. For medical images, the model assumes that Y contains Gaussian noise due to the imaging process, plus the true underlying tissue characteristics. The segmentation result approximates the true tissues, denoted as X, without noise or distortion, as is shown in Fig 2.1. This segmentation is also a 3D volume where there is assigned a class label corresponding to every pixel in the source 3D image. The class label is taken from a set of N labels. Described here is the optimization process by which we classify the pixels into the N labels.

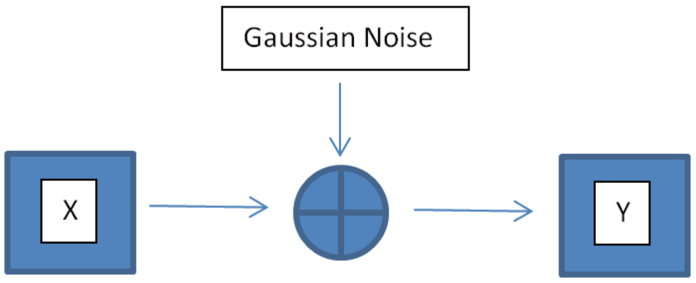


Fig. 2.1. 3D image model

The EM/MPM algorithm consists of two parts: Expectation-Maximization (EM) and Maximization of the Posterior Marginals (MPM). The EM algorithm finds the estimates for Gaussian mean and variance, while MPM classifies the pixels into N class labels, using the estimated parameters from EM. The basic structure of the image processing is a 3D neighborhood of pixels. In the 3D image research field, this forms a mathematical structure called a Markov Random Field (MRF). The MRF is useful because it guarantees local convergence in iterative algorithms which are based on it. The 3D 6-pixel neighborhood which we use is: right, left, above, below, front, and back around a center pixel, as is shown in Fig 2.2.

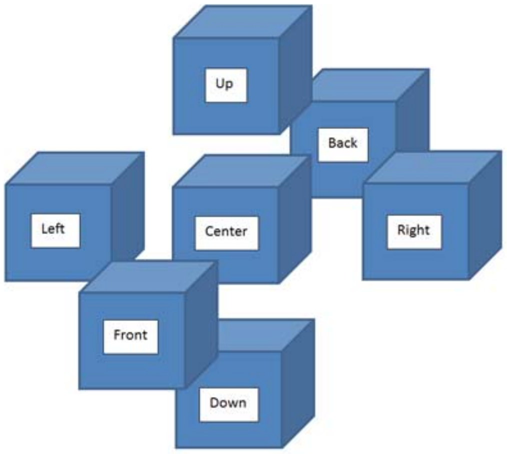


Fig. 2.2. 3D 6-pixel neighborhood

At the beginning of the segmentation process, a random class label is initialized into every pixel in X. An evenly distributed vector of means and variances is used to model noise. Then, the estimate of X (the segmentation output, or class labeling) is formed by iterating several times through the 3D data. For MPM, convergence is achieved by choosing the class label that minimizes the expected value of the number of misclassified pixels, as proved in [9]. For a mixture of Gaussians, in which the random variable Y is dependent on X, the probability density function is modeled in Equation 2.1:

$$f_{Y|X}(y|x,\theta) = \prod_{s\epsilon S} \frac{1}{\sqrt{2\pi\sigma_{x_s}^2}} \exp\left\{-\frac{(y_s-\mu_{x_s})^2}{2\sigma_{x_s}^2}\right\} \tag{2.1}$$

σ = variance for each class

μ = mean for each class

x_s = the center pixel

y_s = the source image

Where θ is the vector of means and variances of each class (or tissue type), and the set S is the 3D volume of pixels with s denoting a single pixel.

Since we are assuming Bayesian dependence, we can use the $p(x)$ to help solve this equation, resulting in Equation 2.2.

$$p(x) = \prod_{s\epsilon S} \frac{1}{\sqrt{2\pi\sigma_{x_s}^2}} exp\{-\frac{(y_s-\mu_{x_s})^2}{2\sigma_{x_s}^2} - \sum_{[r,s]\epsilon C} \beta t(x_s, x_r)\} \tag{2.2}$$

C = clique of X

β = weighting factor for amount of spatial interaction

Here, $p(x)$ represents the tissue probable distribution in the 3D volume depending on the neighborhood class labels. This formulation will favor a class label for a center pixel that is similar to the largest number of neighboring class labels.

In order to get the approximation of this marginal conditional probability mass function at each pixel, a Gibbs sampler is used to generate a Markov chain X(t). After all the pixels have been processed through several iterations, EM uses class persistence from these iterations to estimate the new means and variances, θ, which is the input to MPM for the next iterative segmentation. After tens of EM iterations, the result of EM/MPM algorithm will converge to the highest probability segmentation. These segmentation algorithms were tested with tomographic ultrasound volumes from Karmanos Cancer Institute of breast cancer patients. Figure 2.3 and Figure 2.4 show the ultrasound images before and after segmentation.

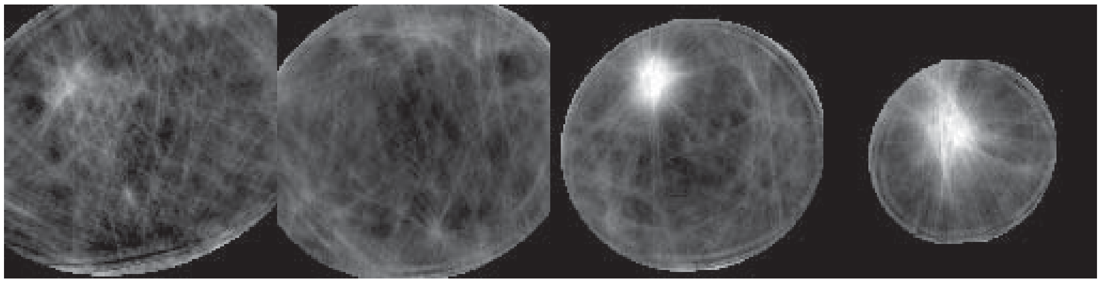


Fig. 2.3. Original ultrasound images, breast cancer

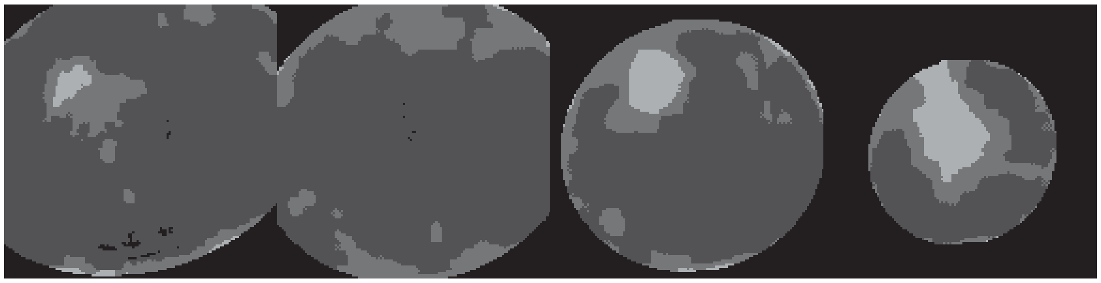


Fig. 2.4. Segmentation results

2.2 Expectation-Maximization

Expectation Maximization (EM) is an iterative procedure for estimation of the mean and the variance of each segmentation classes. At each iteration, two steps are performed: the expectation step and the maximization step. First maximization step is performed, then the segmentation is done in the expectation step, iterating to find the best log-likelihood of the probability that a particular pixel belongs to one of the k classes [4]. The means and variances are represented by the vector: $\theta = (\mu_1, \sigma_1, \ldots, \mu_k, \sigma_k)$. The MPM probability, $p_{x_s|Y}(k|y, \theta(w-1))$, where w is the number of EM iteration, passed out of the MPM loop, is directly applied in the EM update Equations for μ_k, σ_k^2, shown below [1–3].

$$\mu_k(w) = \frac{1}{N_k(p)} \sum_{s \epsilon S} y_s p_{x_s|Y}(k|y, \theta(w-1)) \tag{2.3}$$

$$\sigma_k^2 = \frac{1}{N_k(w)} \sum_{s \epsilon S} (y_s - \mu_k(w))^2 \, p_{x_s|Y}(k|y, \theta(w-1)) \tag{2.4}$$

Where N corresponds to the probability weighted number of pixels in a particular class:

$$N_k(w) = \sum_{s \epsilon S} p_{x_s|Y}(k|y, \theta(w-1)) \tag{2.5}$$

2.3 3D Maximization of Posterior Marginals

At each pixel, the MPM optimization uses the Gaussian distribution of each class and the class probability of the neighborhood pixels. As in the literature [1,2,4], the 3D pixel neighborhood is defined by the function $t(x_r, x_s)$, where x_s is the center pixel to be assigned, and x_r are the nearest 6 pixel neighbors: up, down, left, right, next slice, previous slice. This neighborhood prior probability is defined below as the probability of the segmentation choice of x, given the segmentation choices of the neighbors.

$$p(x) = \frac{1}{z} \exp\left(- \sum_{\{r,s\} \epsilon C} \beta t(x_r, x_s) \right) \tag{2.6}$$

Z = normalizing value

$t(x_r, x_s) = 0, x_r = x_s; t(x_r, x_s) = 1, x_r \neq x_s$

The Gaussian and the prior probability are combined in an equation to be optimized by FPGA hardware computational block. This is calculated in the log domain, eliminating constants and exponentials, where the current segmentation estimate, x_s, is computed by:

$$argmax \left\{ -log\sigma_{x_s} - \frac{\left(y_s - \mu_{x_s}\right)^2}{2\sigma_{x_s}^2} - \sum_{\{r,s\}\epsilon C} \beta t\left(x_r, x_s\right) - \sum_{\{r\}\epsilon C} \gamma_{x_r} \right\} \tag{2.7}$$

3. HARDWARE IMPLEMENTATION ON FPGA

This chapter shows the structure of the whole system and introduces the basic steps of the process. Then, each of main parts of this embedded system are explained in detail.

3.1 The Structure and Main Work Steps of This System

This system consists of a host PC and an external FPGA development board (Xilinx ML605). The host PC sends 3D image data to the FPGA development board through the PCI-Express bus. Then, after FPGA hardware completes the EM/MPM image segmentation processing, the FPGA sends the new 3D image data back to PC. Figure 3.1 shows the system structure; and the following are the working steps of this system:

Step 1: PC sends source image data (y) to PCIe_bridge Endpoint through PCI-Express bus.

Step 2: DMA controller provides Direct Memory Access (DMA) services – transfers programmable quantity of data from PCI-Express Endpoint to DDR3 without MicroBlaze processor intervention.

Step 3: Microblaze generates initial values for MPM logic IP. And sends them to MPM through Processor Local Bus (PLB).

Step 4: MPM logic IP gets data from external Memory (DDR3) and then starts image segmentation processing using MPM algorithm and repeats 7 iterations. At the same time, hardware accumulators begin calculating component parts of $p_{x_s}(k|y, \theta(w-1))$ and $N_k(w)$ for EM.

Step 5: Microblaze calculates the updated EM θ; computed using equation 2.1 and 2.2 for use as in the next iteration of MPM.

Step 6: Then steps 4-5 are repeated, terminating when the change in θ is less than a set threshold.

Step 7: DMA controller transfers the results data from DDR3 to PCI-Express Endpoint without MicroBlaze processor intervention.

Step 8: PC gets the segmented image data from PCIe_bridge Endpoint through PCI-Express bus.

3.2 The Critical Components of This System

3.2.1 MicroBlaze Embedded Processor

The MicroBlaze Embedded processor soft core is a reduced instruction set computer (RISC) optimized for implementation in Xilinx Field Programmable Gate Arrays (FPGAs). It is highly configurable and parameterized to allow selective enabling of additional functionality [10]. In this system, the processor is responsible for the overall coordination of all the components. It calculates new means and variances by using EM algorithm and moves data to and from MPM logic through a Processor Local Bus (PLB).

3.2.2 MPM Logic IP

MPM logic is the main algorithm calculation engine. The implementation of this logic in detail is in [4]; especially useful is the Ping-Pong internal memory architecture which is used to solve the memory bandwidth problem. As shown in Figure 3.1, the MPM logic is created as a custom peripheral IP (Intellectual Property) and connected to the Xilinx core PLB through the IP interface (IPIF). The IPIF is designed to provide a user with a quick to implement interface between the IBM PLB Bus and a User IP core [11], as shown in Figure 3.2. The logic end interface Xilinx IP

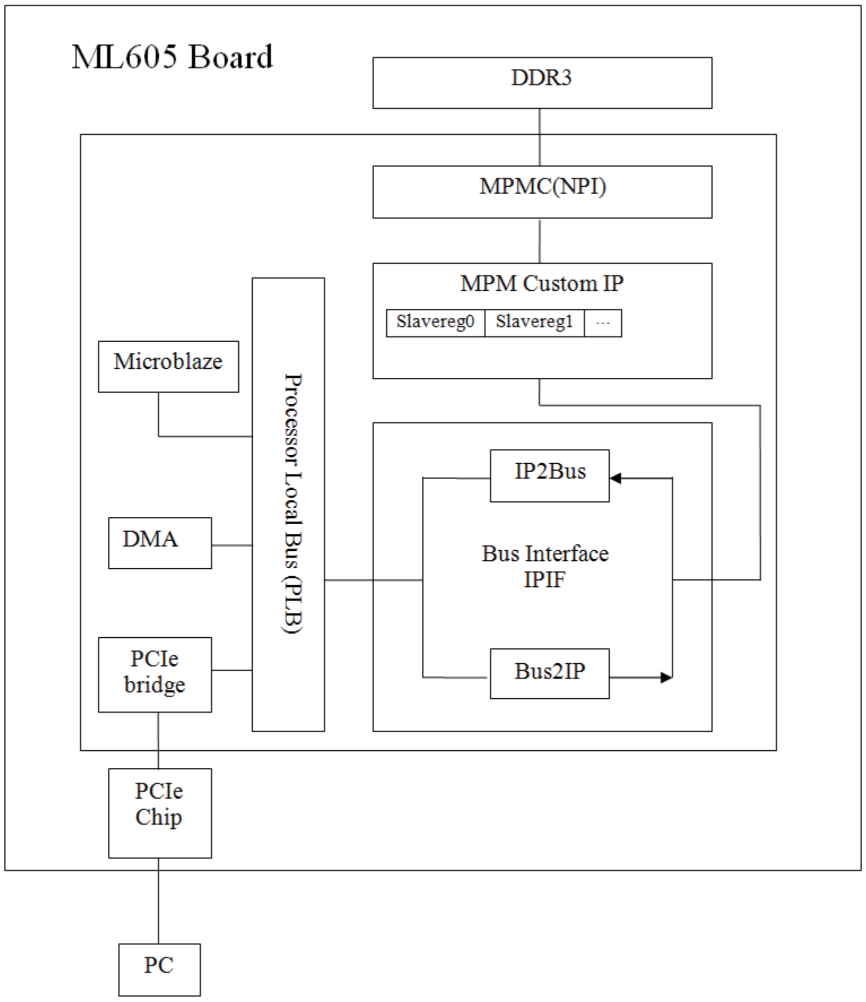


Fig. 3.1. System architecture

Interconnect (IPIC), is common between the PLB IPIF and the user logic IP. This allows user logic IP to use the PLB IPIF for programming control registers. Figure 3.3 shows a block diagram of the PLB IPIF. The diagram also indicates the modules ports as they relate to the IPIF services. The PLB IPIF provides nine services for the User, of which, seven can be optioned in or out. The basic element of the design is the Slave Attachment. This block provides the basic functionality for PLB Slave operation. It implements the protocol and timing translation between the PLB Bus and the IPIC. In this system, only slave registers (slvreg0, slvreg1...) are used for data transfer between MicroBlaze and MPM logic IP through IPIC (IP interface) as shown in Figure 3.3.

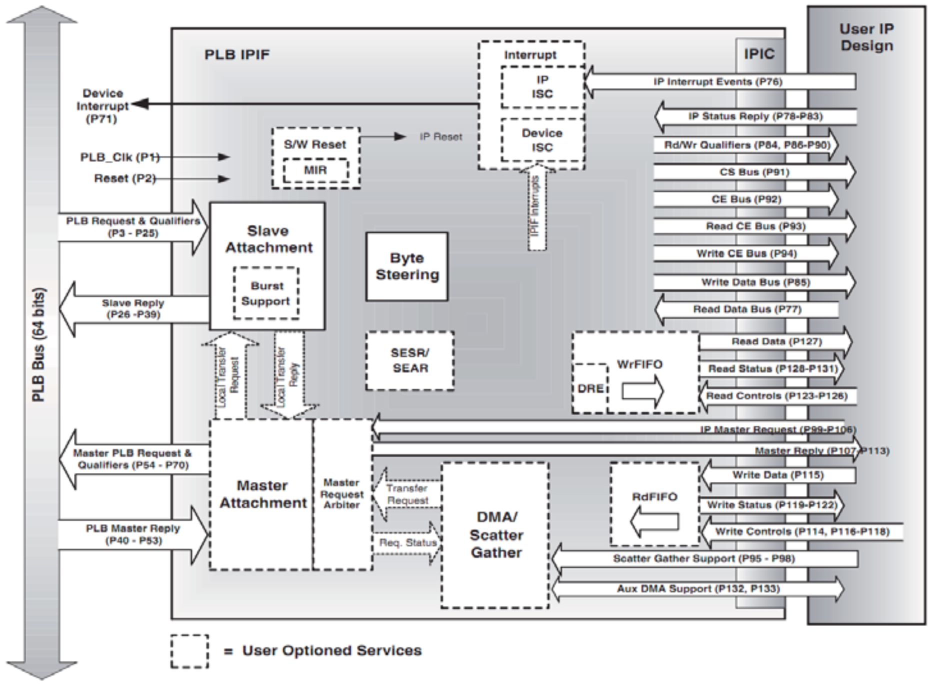


Fig. 3.2. PLB IPIF block diagram

From the algorithm analysis, The EM/MPM algorithm diagram is shown in Figure 3.4. The blocks named Y and Xt are external memory for storing source image Y and segmentation result Xt. The blocks named EM and Basic Parameter will be implemented on the on-Board CPU which is named as Microblaze. It can be seen that each EM iteration calculates the mean and variance of each class based on segmentation results from MPM loops. These new mean and variance are sent to MPM algorithm as input information for a new segmentation processing at the next EM iteration.

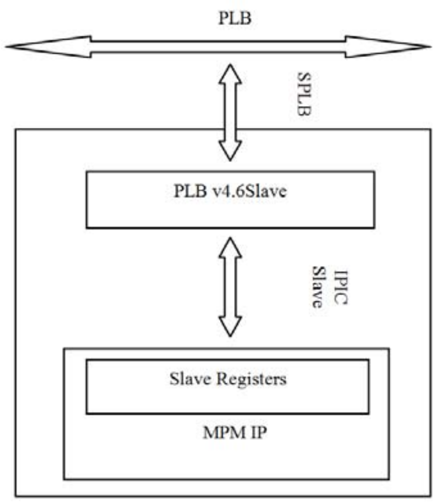


Fig. 3.3. PLB IPIF

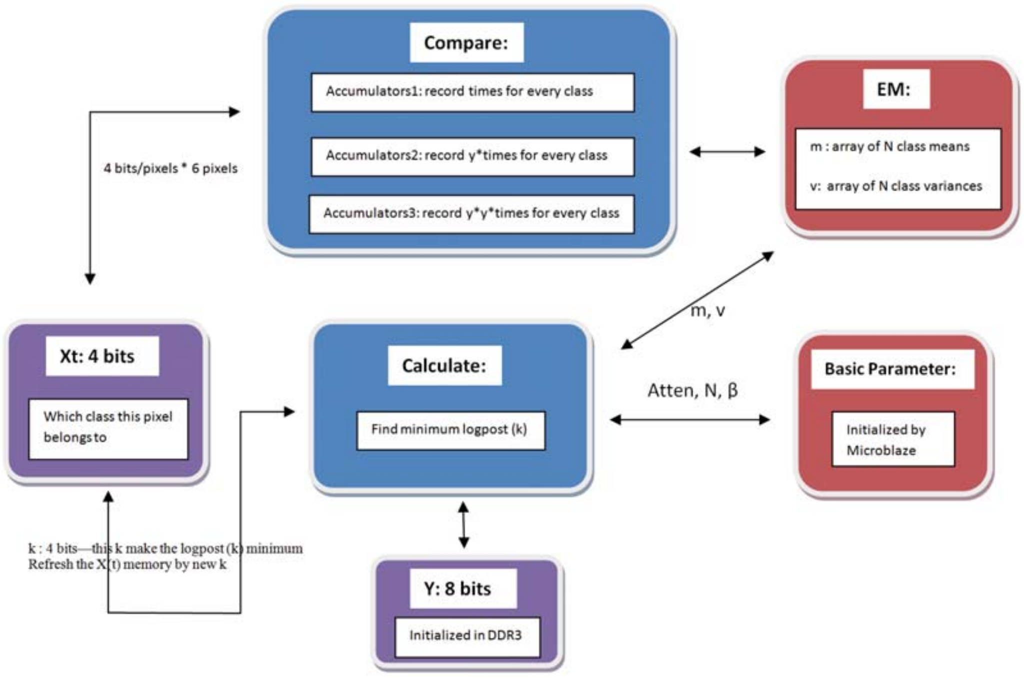

Compare:
Accumulators1: record times for every class
Accumulators2: record y*times for every class
Accumulators3: record y*y*times for every class
4 bits/pixels * 6 pixels
EM:
m : array of N class means
v: array of N class variances
m, v
Xt: 4 bits
Which class this pixel belongs to
Calculate:
Find minimum logpost (k)
Atten, N, β
Basic Parameter:
Initialized by Microblaze
k : 4 bits—this k make the logpost (k) minimum
Refresh the X(t) memory by new k
Y: 8 bits
Initialized in DDR3

Fig. 3.4. Block diagram for algorithm

The MPM segmentation process is to minimize the exponential part for every class with respect to each pixel in Equation 3.1, is named as logpost(k) here:

$$logpost(k) = -log\sigma_{x_s} - \frac{(y_s - \mu_{x_s})^2}{2\sigma_{x_s}^2} - \sum_{[r,s]\epsilon C} \beta t(x_s,x_r) \tag{3.1}$$

The computational block will classify every pixel, assigning it the class label k, based on smallest (magnitude) logpost(k). This computational core is named MPM top. It accomplishes the calculation of two important outputs: Accumulators and Xt out. The Xt out is the current estimated class for each pixel, which represents the current segmentation of the input. The Figure 3.5 is the core diagram. The main ports for MPM top are listed below and the ports named "s2p_ena", "beta", "class", "d_in", "m_in", "b_in", "con_in", "atten", "icm", "varmean" are from the previous MPM design [4]. "Accumulator", "Accumulator(Y)", "Accumulator(YY)" are designed for EM convengence and the other ports are designed as interface between MPM and external memory.

Input:

(1) s2p_ena: calculation enable signal. When ena1 = 1, computational block works.

(2) beta :MPM parameter - weight of spatial interaction (about 2.4 is normal).

(3) class : number of segmentation classes (labels) expected in image

(4) d_in: d = 2*v(v is array of N variances (per class)).

(5) m_in: array of N means (per class) (and in VarMean case this is slope of line).

(6) b_in: array of N variable intercepts in VarMean case.

(7) con_in: $con = log(\sqrt{2 \times \pi \times v})$.

(8) atten: if 0, do not use gamma, else use gamma and scale by atten value.

(9) icm: ICM = 1 if using Besag's ICM, =0 if using Simulated Annealing for optimization in MAP; =2 if using MPM.

(10) varmean: = 1 if using variable mean model (LMS fit to a line), = 0 if not using this method.

(11) app_rd_data: This provides the output data from read commands.

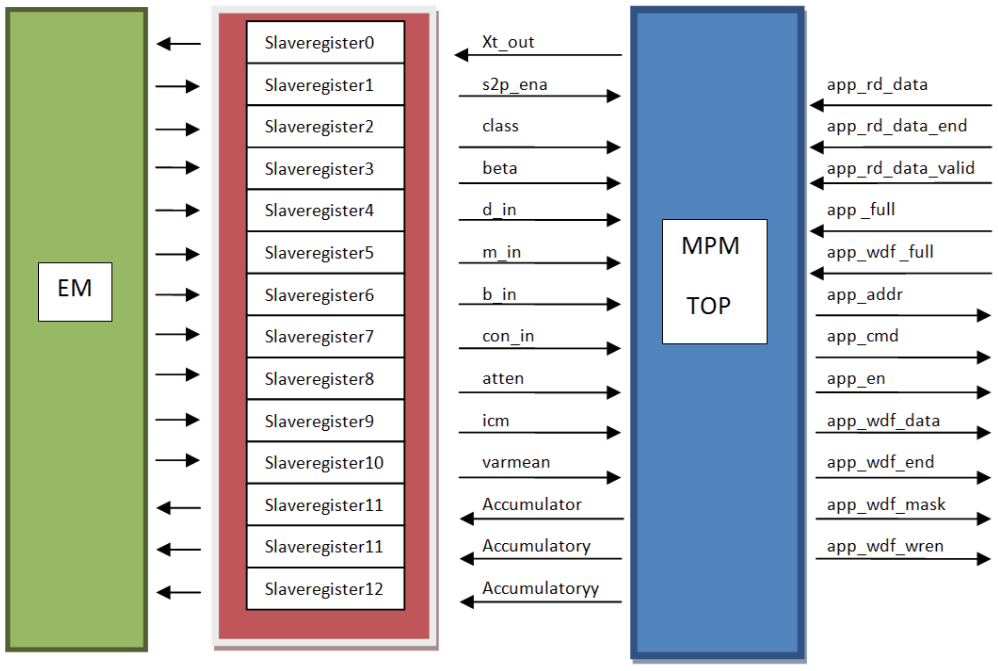


Fig. 3.5. Block diagram for MPM

(12) app_rd_data_end: This active-High output indicates that the current clock cycle is the last cycle of output data on app_wdf_data.

(13) app_rd_data_valid: This active-High output indicates that app_rd_data is valid.

(14) app_full: This output indicates that the MPM is ready to accept commands. If the signal is deasserted when app_en is enabled, the current app_cmd and app_addr must be retried until app_rdy is asserted.

(15) app_wdf_full: This output indicates that the write data FIFO is ready to receive data. Write data is accepted when app_wdf_rdy = 1'b1 and app_wdf_wren = 1'b1.

Output:

(1) Xt out: output of this computational block. It illustrates the class with this central pixel most likely belongs to.

Table 3.1
PCIe_bridge settings

IPIF BAR High Address	0xafffffff
IPIF BAR Base Address	0xa0000000
Remote PCI device BAR to which IPIF BAR is translated	0x0000000
Power of 2 defining the Size in Bytes of PCI BAR Space	32
Remote PLB device BAR to which PCI BAR is translated	0x90000000

(2) Accumulator: output accumulation to assist in EM sum for mean & variance.

(3) Accumulator(Y): output accumulation to assist in EM sum for mean.

(4) Accumulator(YY): output accumulation to assist in EM sum for variance.

(5) app_addr: This input indicates the address for the current request.

(6) app_cmd: This input selects the command for the current request.

(7) app_en: This is the active-High strobe for the app_addr, app_cmd, app_sz, and app_hi_pri inputs.

(8) app_wdf_data: This provides the data for write commands.

(9) app_wdf_end: This active-High input indicates that the current clock cycle is the last cycle of input data on app_wdf_data.

(10) app_wdf_mask: This provides the mask for app_wdf_data.

(11) app_wdf_wren: This is the active-High strobe for app_wdf_data.

3.2.3 PCI-Express Bus Controller Bridge

We compared the ML605 Evaluation board's interfaces: Serial Rapid I/O, 10 Gigabit Ethernet; and the PCI Express (Peripheral Component Interconnect Express). The PCI Express was chosen as data transfer bus between PC and the board because it is the highest speed and best bandwidth and latency available. The ML605 Evaluation board has an 8-lane PCIe edge connector (Figure 3.6) which performs data transfers at the rate of 2.5 GT/s.

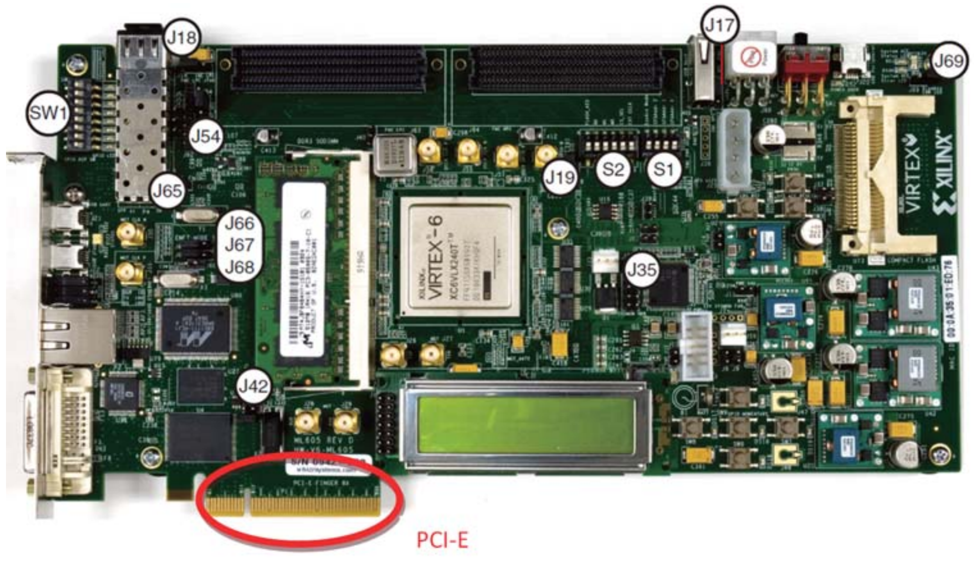


Fig. 3.6. PCIe on board

The PLBv46 Bridge is used to provide transaction level translation of PLB bus commands to PCIe [12]. The key to realize efficient data transfer is address translation. PCIe address space is quite different from PLB address space. So in order to access from one address space to another address space, address translation is needed. PLB Bridge has IPIF base address registers (BARs) and the generics C_IPIFBAR_NUM, C_IPIFBAR_HIGHADDR, C_IPIFBAR2PCIBAR, C_IPIFBAR, and C_IPIFBAR_AS are used to configure the BARs. As is shown in Table 3.1

In this system, "IPIF BAR Base Address" of the PLB address is set as 0xa0000000. "Remote PCI device BAR to which IPIF BAR is transferred" is set as 0x00000000 in IPIF BAR column and "Remote PLB device BAR to which PCI BAR is transferred" is set as 0x90000000 in PCIe BAR column (the base address of DDR3 is 0x90000000). Then data can be read and writen from PCIe BAR at the address (0x00000000 + offset) by PC. Under the Windows 7 operating system, Windriver is set as the PCI-Express driver development tool. Windriver's PLX Company's development driver package [13] can be operated directly and software applications are easily developed.

3.2.4 DMA Transfer Mode

DMA is technique used for efficient transfer of data to and from host CPU system memory. In our system, we choose XPS Central DMA Controller. The XPS Central DMA Controller operates on the PLB using independent master and slave interfaces. When the registers of it are being read and writen, it performs as a slave. And when it is doing DMA operation, it performs as a master [14]. The operation of XPS Central DMA Controller determined by the values been written into the DMA registers such as Source Address Register, Destination Address Register, DMA Control Register, etc. After initiating these registers in our design, we set source address as the PCI-Express Endpoint base address and set destination address as the DDR3 base address when the system transfers data from PC to DDR3. Similarly, the source and destination address are exchanged when the system transfers data from DDR3 to PC.

3.2.5 MPMC(NPI)

Memory arrangement and interface design is very important for this system. So we choose MPMC instead of original memory interface controller due to its convenience in embeded system design. From the algorithm analysis above, current segmentation Xt and original source image Y are volume inputs to this system. Refreshed segmentation Xt is volume output data of this system. According to the system requirements, every Xt is assigned with 4 bits (maximum 16 class labels) and every Y is assigned 8 bits greyscale. Therefore, 12 bits are needed to read in and 4 bits write out per pixel.

From analysis of algorithm, Y of central pixel and Xt of its six neighbors should be sent to calculation blocks at same time. So the source image volume Y and the current segmentation field volume Xt are both read in data flows. In order to read in data more efficiently, Y of Slice(n) is stored with Xt of Slice(n+1) in the same address. The 3D image size is 128 * 128 * 128, with the external memory storage arrangement is as follows: First, Xt of Slice(1) is generated randomly and stored in inner memory on chip; then Y of Slice(1) and Xt of Slice(2) are read in; then Xt of

Table 3.2
External memory storage arrangement

Y of Slice 1	Xt of Slice 2
Y of Slice 2	Xt of Slice 3
Y of Slice 3	Xt of Slice 4
...	...
...	...
...	...
Y of Slice 126	Xt of Slice 127
Y of Slice 127	Xt of Slice 128
Y of Slice 128	Xt of Slice 1

Slice(1), Y of Slice(1) and Xt of Slice(2) are sent into sixteen calculation blocks to get new Xt of Slice(1); Finally when the refreshed Xt of Slice(1) is ready, it will be sent out to external memory in the same address of Y of Slice(128). This process is repeated for Y of Slice(2) and Xt of Slice(3). All the slices are processed according to the order listed in Table 3.2. The advantage to store Y and Xt in this order is that Y and Xt can be sent into calculation blocks together. Y is needed one slice ahead of Xt, and this arrangement makes pipelining more efficient. Both Y of central slice and Xt of back slice will be used in refreshing central pixels. When they are read in at the same time and Xt of front slice is already kept in inner memory so read-in Y doesn't need to wait for Xt of the back slice.

Between external memory and inner memory, there is an interface to rearrange Xt and Y data to fit the computational cores. At the same time, this interface should control slice storage of Xt and Y. Figure 3.7 is shows this control process.

The PLBv46 Bridge design provides full bridge functionality between the Xilinx PLB and user IP. It provides a 32 bit transform. In Figure 3.7, Y1, Y0 refer to Y of Slice 2 and Slice 1, and Xt1, Xt0 refer to Xt of Slice 2 and Slice 1. Y is 8 bits

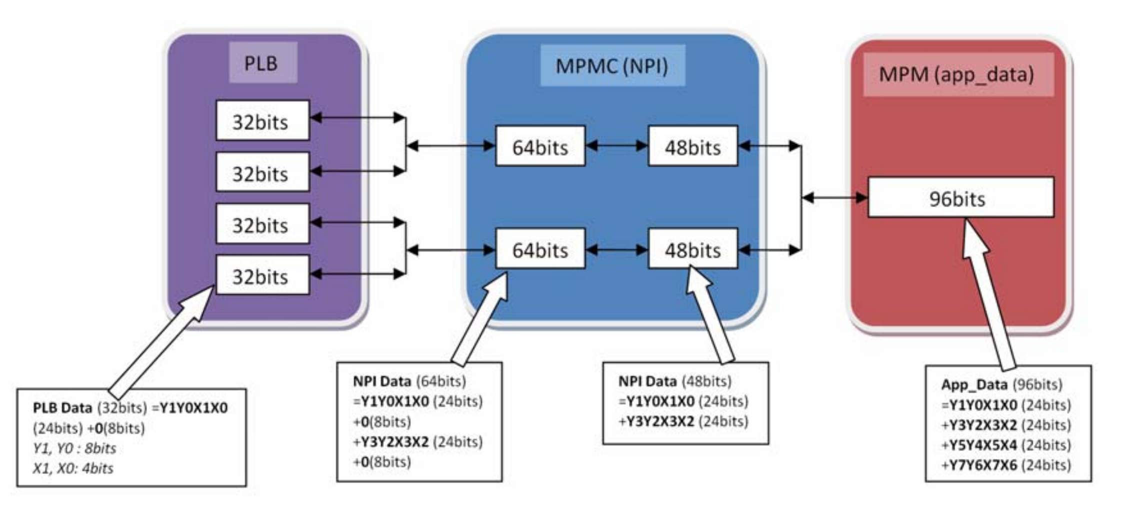

PLB
32bits
32bits
32bits
32bits
MPMC (NPI)
64bits
48bits
64bits
48bits
MPM (app_data)
96bits
PLB Data (32bits) =Y1Y0X1X0 (24bits) +0(8bits)
Y1, Y0 : 8bits
X1, X0: 4bits
NPI Data (64bits) =Y1Y0X1X0 (24bits) +0(8bits) +Y3Y2X3X2 (24bits) +0(8bits)
NPI Data (48bits) =Y1Y0X1X0 (24bits) +Y3Y2X3X2 (24bits)
App_Data (96bits) =Y1Y0X1X0 (24bits) +Y3Y2X3X2 (24bits) +Y5Y4X5X4 (24bits) +Y7Y6X7X6 (24bits)

Fig. 3.7. Data rearrangement

data and Xt is 4bits data. So Y1Y0Xt1Xt0 is 24bits. In order to satisfy the 32 bits transform, 8 bits of 0s is added.

MPMC is a fully parameterizable memory controller that supports various RAM. MPMC provides access to memory for one to eight ports, where each port can be chosen from a set of Personality Interface Modules (PIMs) that permit connectivity into PowerPC 405 processors and MicroBlaze processors using Processor Local Bus (PLB)v4.6 and the Xilinx CacheLink (XCL) structures. For low-level direct access to the memory controller core, a Native Port Interface (NPI) PIM is available for soft memory controllers [15]. (Figure 3.8)

Virtex6 chip is used in this system, according to MPMC Architecture-Specific Features (Figure 3.9), MPMC supports 32 bits data tranform to DDR3 memory and Native Port Interface (NPI) PIM is also available. MPMC supports NPI data widths of 32 and 64 bits. As is shown in Figure 3.9, this system uses 64 bits data width and in the NPI logic core all added 0s are cut off; app_data then gets 96 bits data and sends it to MPM (MPM reading process). So the clock frequency ratio of PLB,

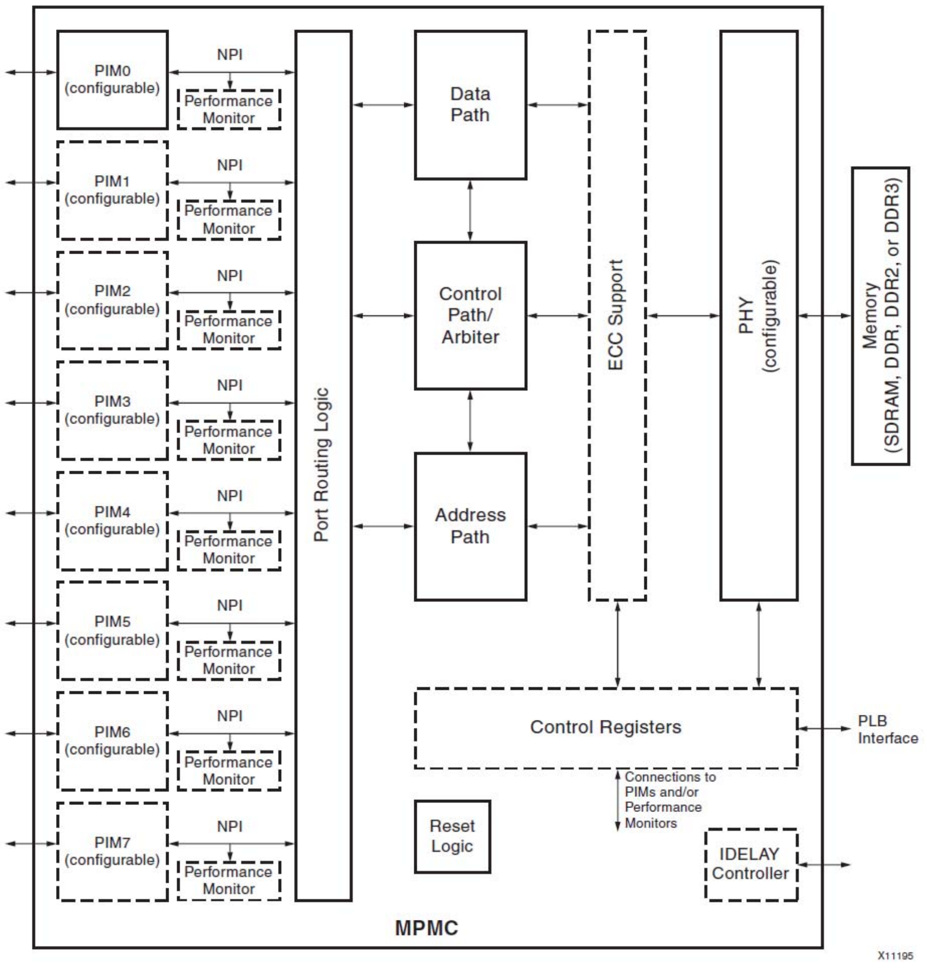
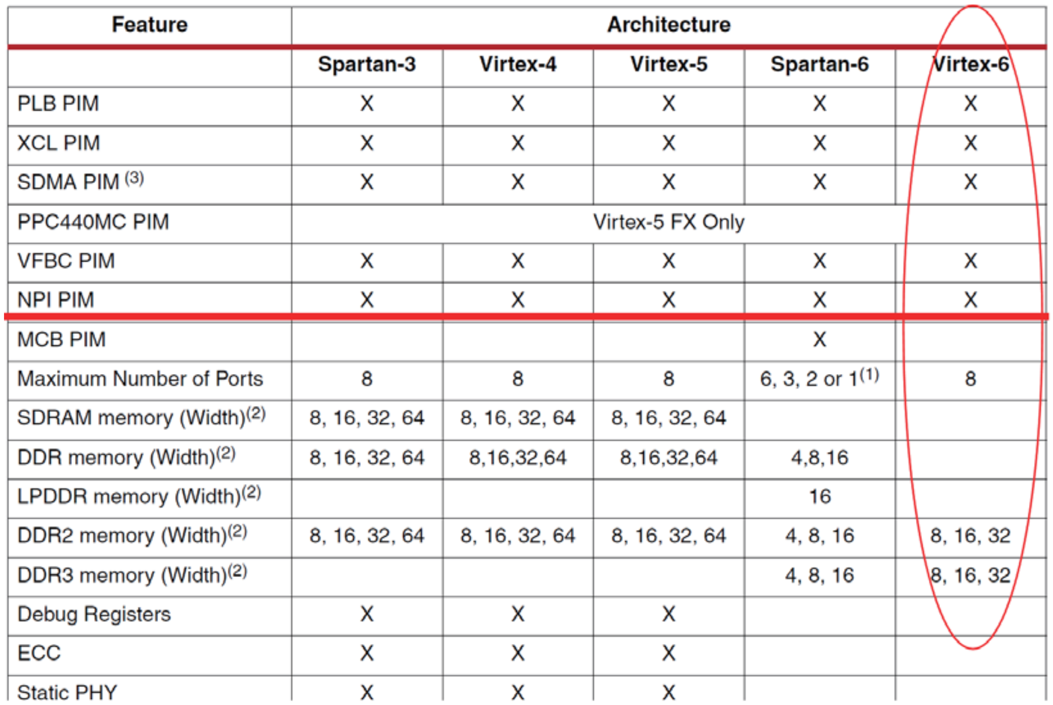


Fig. 3.8. MPMC soft memory controller architecture block diagram

MPMC and MPM is 1:2:4. Figure 3.10 shows the MPMC Interface of this system. Port0 connects with PLBv46, Port1 connects with NPI and leave the others inactive.

The MPM and memory interface is connected in Figure 3.11. MPMC controls DDR3 and the NPI interface is used to connect with MPM. The main ports for MPMC NPI are listed below:

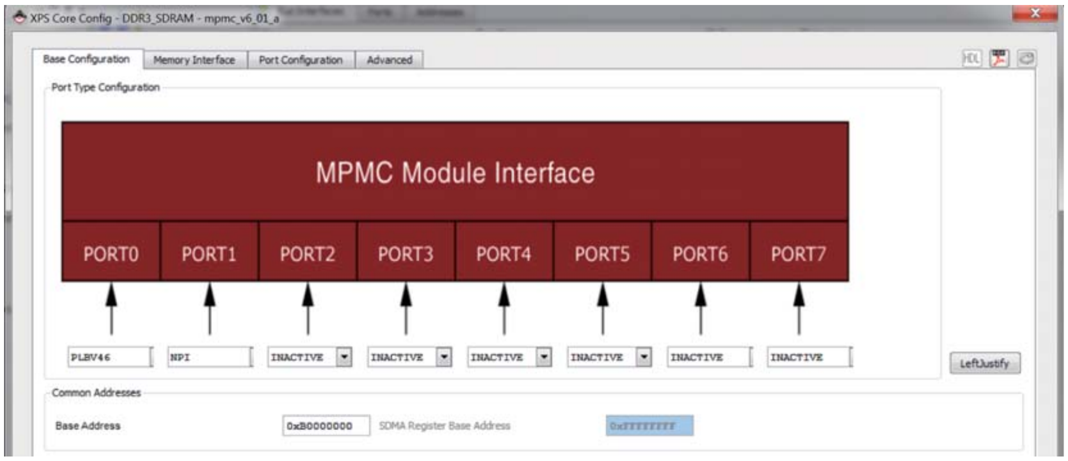
(1)MPMC_RdFIFO_Data: Data to be popped out of MPMC read FIFOs. Only valid a certain number of cycles after MPMC_RdFIFO_Empty is deasserted.



Feature	Architecture				
	Spartan-3	Virtex-4	Virtex-5	Spartan-6	Virtex-6
PLB PIM	X	X	X	X	X
XCL PIM	X	X	X	X	X
SDMA PIM [3]	X	X	X	X	X
PPC440MC PIM	Virtex-5 FX Only				
VFBC PIM	X	X	X	X	X
NPI PIM	X	X	X	X	X
MCB PIM				X	
Maximum Number of Ports	8	8	8	6, 3, 2 or 1[1]	8
SDRAM memory (Width)[2]	8, 16, 32, 64	8, 16, 32, 64	8, 16, 32, 64		
DDR memory (Width)[2]	8, 16, 32, 64	8,16,32,64	8,16,32,64	4,8,16	
LPDDR memory (Width)[2]				16	
DDR2 memory (Width)[2]	8, 16, 32, 64	8, 16, 32, 64	8, 16, 32, 64	4, 8, 16	8, 16, 32
DDR3 memory (Width)[2]				4, 8, 16	8, 16, 32
Debug Registers	X	X	X		
ECC	X	X	X		
Static PHY	X	X	X		

Fig. 3.9. MPMC architecture-specific features

(2)MPMC_RdFIFO_Empt: When this active high signal, is de-asserted,(0), it indicates that enough data is in the read FIFOs to assert.



PORT0	PORT1	PORT2	PORT3	PORT4	PORT5	PORT6	PORT7
PLBV46	NPI	INACTIVE	INACTIVE	INACTIVE	INACTIVE	INACTIVE	INACTIVE

Base Address 0xB0000000 SDMA Register Base Address 0xFFFFFFFF

Fig. 3.10. MPMC module interface

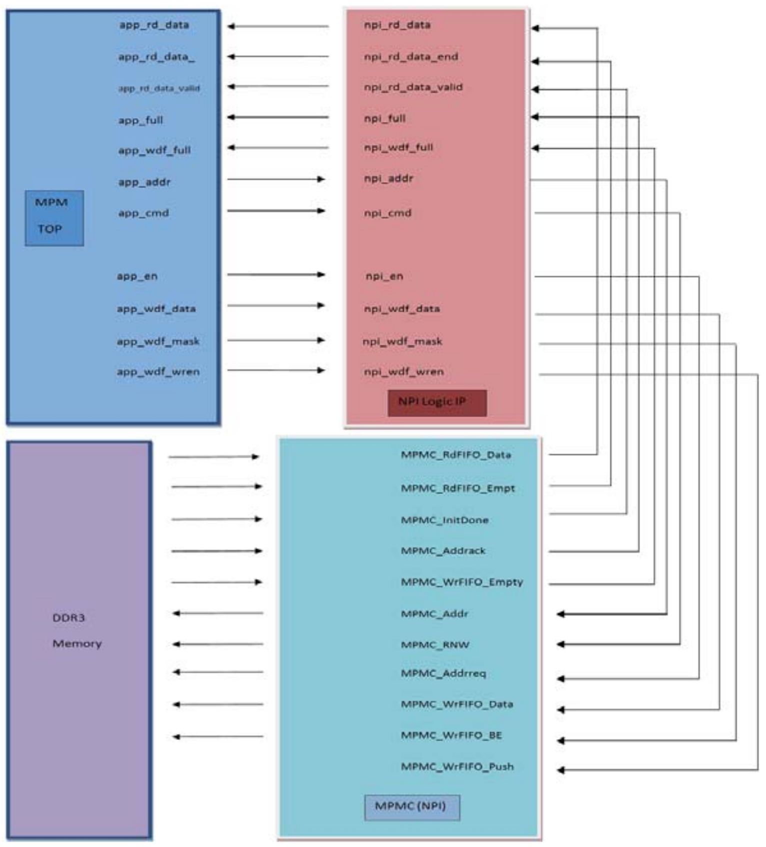


Fig. 3.11. Memory interface connection

(3)MPMC_InitDone: It indicates that initialization is complete and that FIFOs are available for use.

(4)MPMC_Addrack: This active high signal indicates that MPMC has begun arbitration for address request. Valid for one cycle of MPMC_Clk0. MPMC_AddrReq must be deasserted on the next cycle of MPMC_Clk0 unless NPI is requesting a new transfer.

(5)MPMC_WrFIFO_Empty: This active high signal indicates that there are less than C_MEM_DATA_WIDTH bits of data in the write FIFO.

(6)MPMC_Addr: Indicates the starting address of a particular request. Only valid when MPMC_AddrReq is valid.

(7)MPMC_RNW: Read/Not Write: 0 = Request is a Write request. 1 = Request is a Read request. Only valid when MPMC_AddrReq is valid.

(8)MPMC_Addreq: This active high signal indicates that NPI is ready for MPMC to arbitrate an address request. This request cannot be aborted. Must be asserted until MPMC_AddrAck is asserted.

(9)MPMC_WrFIFO_Data: Data to be pushed into MPMC write FIFOs. Only valid with MPMC_WrFIFO_Push.

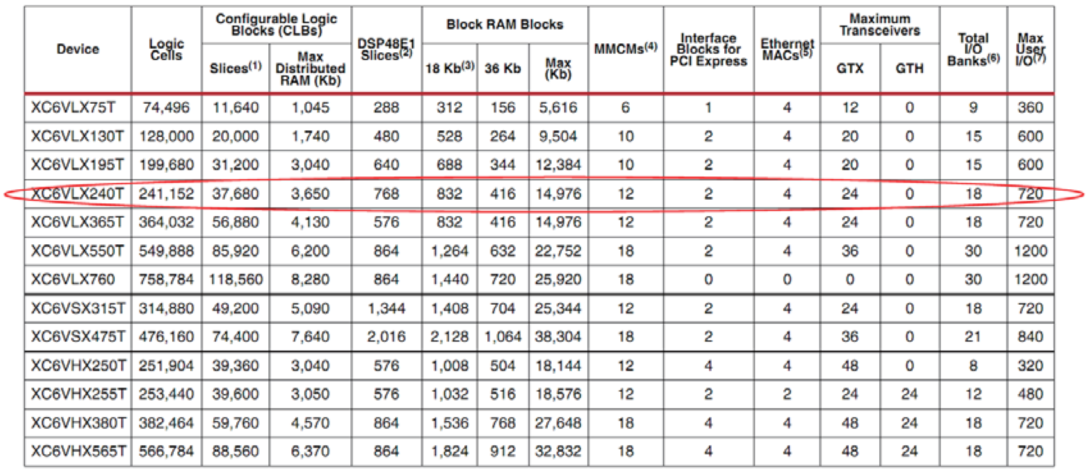
(10)MPMC_WrFIFO_Be: Indicates which bytes of MPMC_WrFIFO_Data to write. Only valid with MPMC_WrFIFO_Push.

(11)MPMC_WrFIFO_Push: This active high signal indicates push WrFIFO_Data into write FIFOs. Must be asserted for one cycle of MPMC_Clk0. Cannot be asserted while MPMC_InitDone is 0.

4. ON BOARD RESULTS AND ANLAYSIS

4.1 Hardware Synthesis Resource Analysis

This system is based on Xilinx Virtex-6 FPGA ML605 Evaluation Kit. It has XC6VLX240T-1FFG1156 device, 200 MHz oscillator (differential), PCI Express Gen1 8-lane (x8) and Gen2 4-lane (x4), and DDR3 SODIMM (512 MB). It provides a development environment for system designs that demand high-performance, serial connectivity and advanced memory interfacing [15]. The basic on-board resource is shown circled in Figure 4.1.



Device	Logic Cells	Configurable Logic Blocks (CLBs)		DSP48E1 Slices(2)	Block RAM Blocks			MMCMs(4)	Interface Blocks for PCI Express	Ethernet MACs(5)	Maximum Transceivers		Total I/O Banks(6)	Max User I/O(7)
		Slices(1)	Max Distributed RAM (Kb)		18 Kb(3)	36 Kb	Max (Kb)				GTX	GTH		
XC6VLX75T	74,496	11,640	1,045	288	312	156	5,616	6	1	4	12	0	9	360
XC6VLX130T	128,000	20,000	1,740	480	528	264	9,504	10	2	4	20	0	15	600
XC6VLX195T	199,680	31,200	3,040	640	688	344	12,384	10	2	4	20	0	15	600
XC6VLX240T	241,152	37,680	3,650	768	832	416	14,976	12	2	4	24	0	18	720
XC6VLX365T	364,032	56,880	4,130	576	832	416	14,976	12	2	4	24	0	18	720
XC6VLX550T	549,888	85,920	6,200	864	1,264	632	22,752	18	2	4	36	0	30	1200
XC6VLX760	758,784	118,560	8,280	864	1,440	720	25,920	18	0	0	0	0	30	1200
XC6VSX315T	314,880	49,200	5,090	1,344	1,408	704	25,344	12	2	4	24	0	18	720
XC6VSX475T	476,160	74,400	7,640	2,016	2,128	1,064	38,304	18	2	4	36	0	21	840
XC6VHX250T	251,904	39,360	3,040	576	1,008	504	18,144	12	4	4	48	0	8	320
XC6VHX255T	253,440	39,600	3,050	576	1,032	516	18,576	12	2	2	24	24	12	480
XC6VHX380T	382,464	59,760	4,570	864	1,536	768	27,648	18	4	4	48	24	18	720
XC6VHX565T	566,784	88,560	6,370	864	1,824	912	32,832	18	4	4	48	24	18	720

Fig. 4.1. Virtex-6 FPGA feature summary by device

With MPM computation cores, external memory interface and EM algorithm implemented on chip, the resource usage in synthesis report is shown in Table 4.1. It is

Table 4.1
Device utilization summary

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	55,175	301,44	18%
Number of Slice LUTs	55,865	150,72	37%
Number used as logic	42,263	150,720	28%
Number used as Memory	7,182	58,400	12%
Number of occupied Slices	17,631	37,680	46%
Number with an unused Flip Flop	16,875	56,904	29%
Number with an unused LUT	1,039	56,904	1%
Number of fully used LUT-FF pairs	38,990	56,904	68%
Number of slice register	3,868	301,440	1%
Number of bonded IOBs	74	600	12%
Number of bonded IPADs	4	62	6%
Number of bonded OPADs	2	40	5%
Number of RAMB36E1/FIFO36E1s	170	416	40%
Number of RAMB18E1/FIFO18E1s	3	832	1%
Number of BUFG/BUFGCTRLs	12	32	37%
Number of ILOGICE1/ISERDESE1s	34	720	4%
Number of OLOGICE1/OSERDESE1s	75	720	10%
Number of DSP48E1s	531	768	69%
Number of IODELAYE1s	46	720	6%
Number of MMCM_ADVs	2	12	16%
Number of PCIE_2_0s	1	2	50%

shown that all the resource usage is under 70%, which makes on-chip implementation achievable and scalable.

4.2 Results Analysis

The design is based on Xilinx EDK12.1 using Xilinx Vertex6lx240t FPGA. The Xilinx ML605 development board is plugged in Dell Precision T3500 Workstation (Figure 4.2). The workstation has an Intel Quad Core Xeon 64bit CPU, 6GB of RAM. The test case for this system is a 128*128*128 3D medical image. The Y data and Xt data are 8 bits and 4 bits respectively. In the system, the 128 slice images are used as input sequence. Every slice does 7 MPM iterations and 50 EM iterations. The read in and write out clock for external DDR3 memory is set at 200MHz. The clock for the computational core is 50MHz. The external memory clock for the I/O interface is limited to 333MHz. The input data are: original image information Y, prior segmentation Xt for each pixel, and class means and variance for each class.

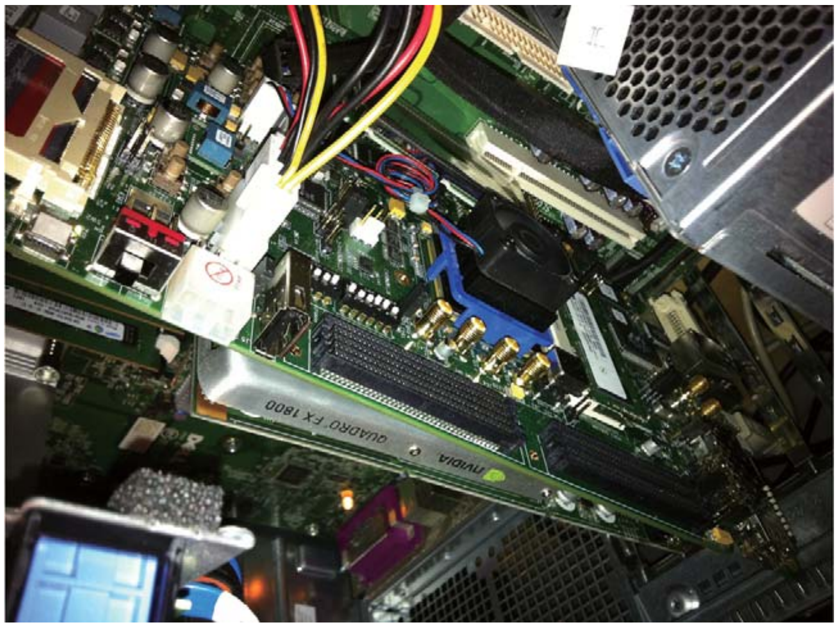
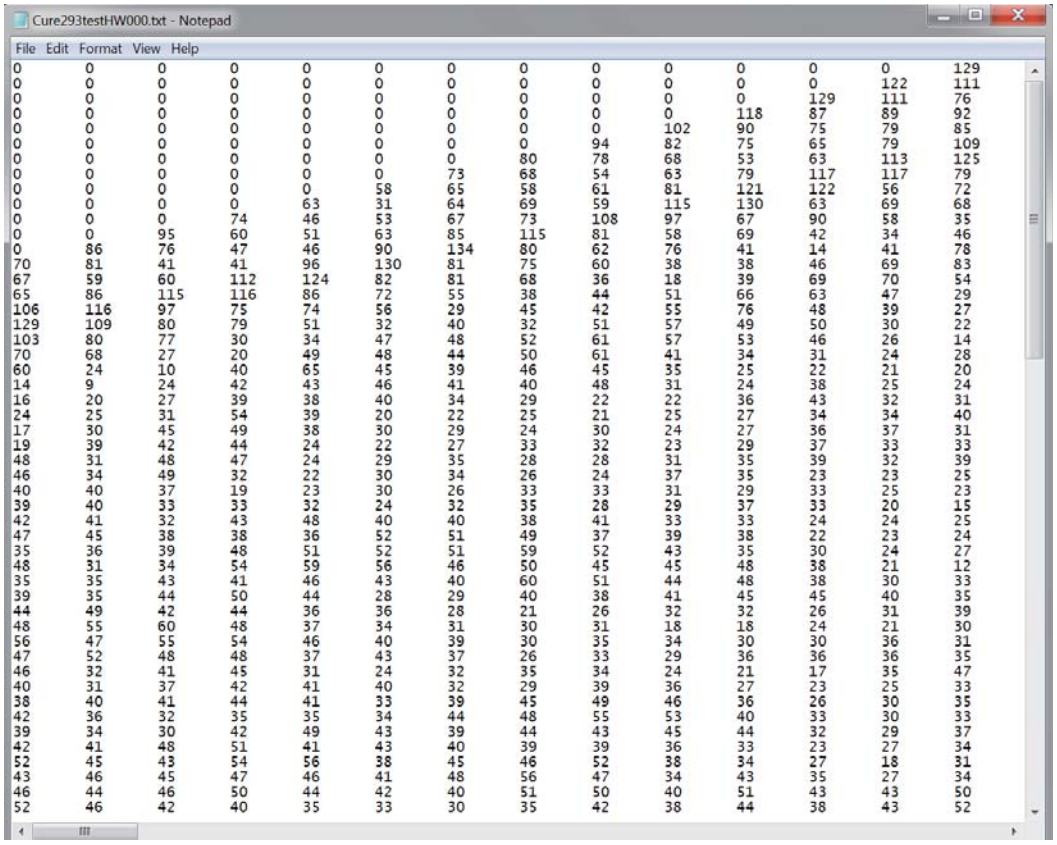


Fig. 4.2. Xilinx development board and Dell workstation

The software application is developed by Visual Studio 2010 and Windriver. In this system, first all the image data are saved as 128*128 matrix in a text files (Figure 4.3) then changed to one dimensional matrix with hex format (Figure 4.4) . When

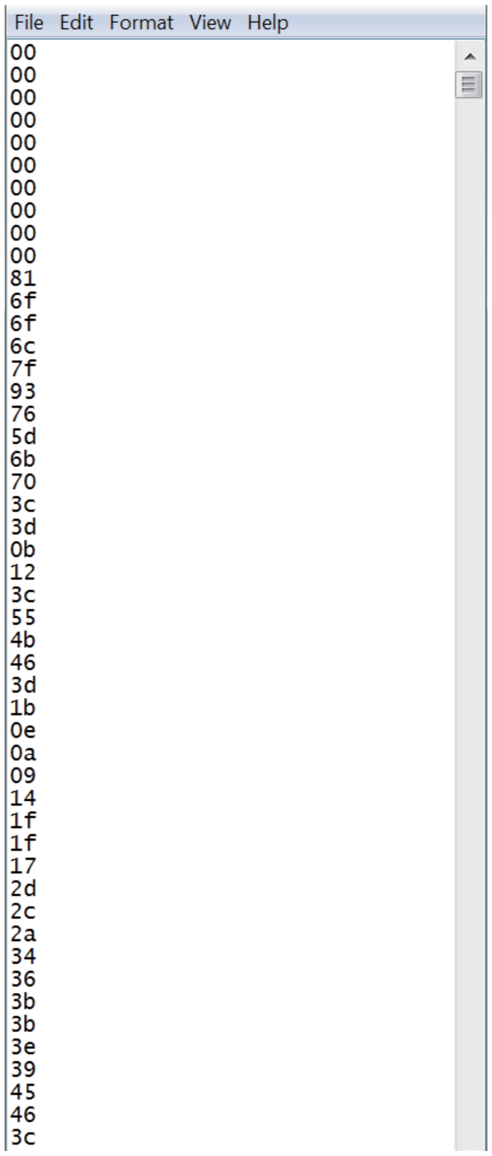


```
Cure293testHW000.txt - Notepad
File  Edit  Format  View  Help
0     0     0     0     0     0     0     0     0     0     0     0     0     129
0     0     0     0     0     0     0     0     0     0     0     0     122   111
0     0     0     0     0     0     0     0     0     0     0     129   111   76
0     0     0     0     0     0     0     0     0     0     118   87    89    92
0     0     0     0     0     0     0     0     0     102   90    75    79    85
0     0     0     0     0     0     0     0     94    82    75    65    79    109
0     0     0     0     0     0     0     80    78    68    53    63    113   125
0     0     0     0     0     0     73    68    54    63    79    117   117   79
0     0     0     0     0     58    65    58    61    81    121   122   56    72
0     0     0     0     63    31    64    69    59    115   130   63    69    68
0     0     0     74    46    53    67    73    108   97    67    90    58    35
0     0     95    60    51    63    85    115   81    58    69    42    34    46
0     86    76    47    46    90    134   80    62    76    41    14    41    78
70    81    41    41    96    130   81    75    60    38    38    46    69    83
67    59    60    112   124   82    81    68    36    18    39    69    70    54
65    86    115   116   86    72    55    38    44    51    66    63    47    29
106   116   97    75    74    56    29    45    42    55    76    48    39    27
129   109   80    79    51    32    40    32    51    57    49    50    30    22
103   80    77    30    34    47    48    52    61    57    53    46    26    14
70    68    27    20    49    48    44    50    61    41    34    31    24    28
60    24    10    40    65    45    39    46    45    35    25    22    21    20
14    9     24    42    43    46    41    40    48    31    24    38    25    24
16    20    27    39    38    40    34    29    22    22    36    43    32    31
24    25    31    54    39    20    22    25    21    25    27    34    34    40
17    30    45    49    38    30    29    24    30    24    27    36    37    31
19    39    42    44    24    22    27    33    32    23    29    37    33    33
48    31    48    47    24    29    35    28    28    31    35    39    32    39
46    34    49    32    22    30    34    26    24    37    35    23    23    25
40    40    37    19    23    30    26    33    33    31    29    33    25    23
39    40    33    33    32    24    32    35    28    29    37    33    20    15
42    41    32    43    48    40    40    38    41    33    33    24    24    25
47    45    38    38    36    52    51    49    37    39    38    22    23    24
35    36    39    48    51    52    51    59    52    43    35    30    24    27
48    31    34    54    59    56    46    50    45    45    48    38    21    12
35    35    43    41    46    43    40    60    51    44    48    38    30    33
39    35    44    50    44    28    29    40    38    41    45    45    40    35
44    49    42    44    36    36    28    21    26    32    32    26    31    39
48    55    60    48    37    34    31    30    31    18    18    24    21    30
56    47    55    54    46    40    39    30    35    34    30    30    36    31
47    52    48    48    37    43    37    26    33    29    36    36    36    35
46    32    41    45    31    24    32    35    34    24    21    17    35    47
40    31    37    42    41    40    32    29    39    36    27    23    25    33
38    40    41    44    41    33    39    45    49    46    36    26    30    35
42    36    32    35    35    34    44    48    55    53    40    33    30    33
39    34    30    42    49    43    39    44    43    45    44    32    29    37
42    41    48    51    41    43    40    39    39    36    33    23    27    34
52    45    43    54    56    38    45    46    52    38    34    27    18    31
43    46    45    47    46    41    48    56    47    34    43    35    27    34
46    44    46    50    44    42    40    51    50    40    51    43    43    50
52    46    42    40    35    33    30    35    42    38    44    38    43    52
```

Fig. 4.3. Original Y matrix

image processing starts, the first task is to initialize all Y and Xt to external memory. Figure 4.5 shows the sofware application on PC which is used to send or receive image data through PCI Express. In the main menu, choose “3. Read/write memory and IO addresses on the device”. Then choose “5. Write to active address space”, if PC transfers data to DDR3 (Figure 4.6). And choose “4. Read from active address space”, if PC receives data from DDR3 (Figure 4.7).

Then after all the Y and Xt are available in external DDR3 memory, the read in process starts, this is achieved in 0.053742s. Upon being read-in, the Y and Xt are sent to calculation cores to process. For this 128*128*128 volume 3D image with 7 MPM iterations complete, the time is 0.089619s. Then for EM convergence, the total segmentation for this size volume is 11.129754s (Figure 4.8). 11 seconds (0.2

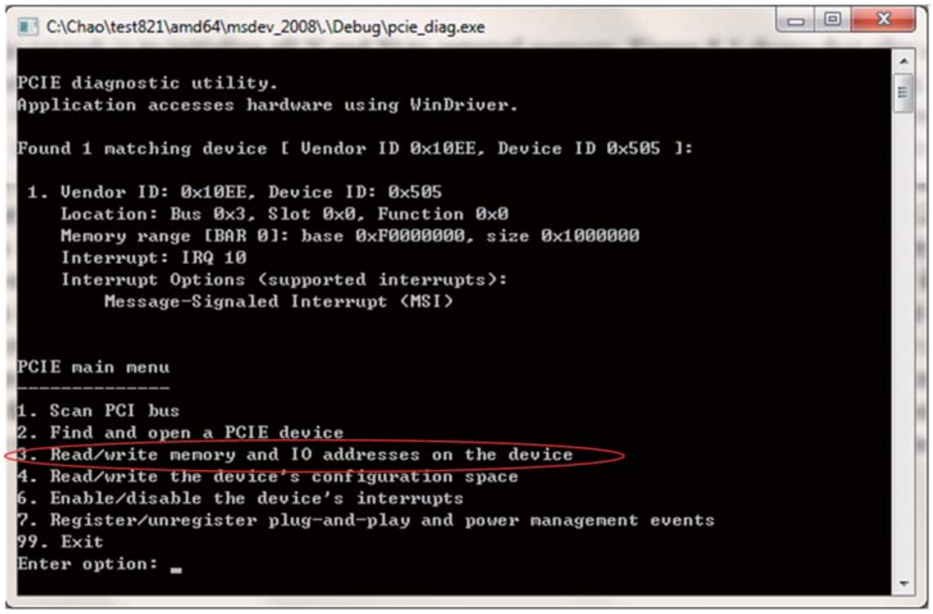


```
00
00
00
00
00
00
00
00
00
00
81
6f
6f
6c
7f
93
76
5d
6b
70
3c
3d
0b
12
3c
55
4b
46
3d
1b
0e
0a
09
14
1f
1f
17
2d
2c
2a
34
36
3b
3b
3e
39
45
46
3c
```

Fig. 4.4. Original Y matrix (hex matrix)

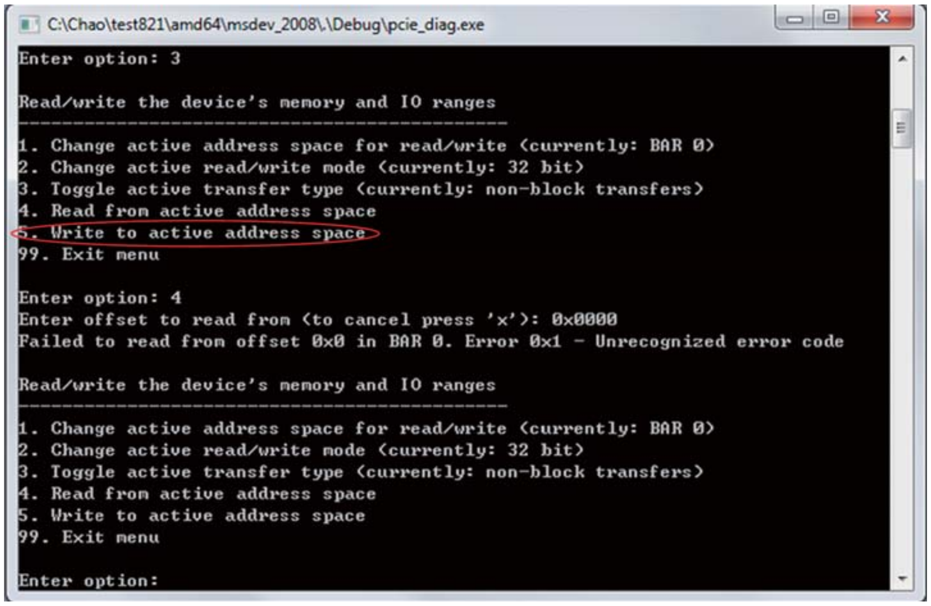
minutes) of processing time with the hardware acceleration, compared to 23 minutes on a quad core PC, thus this system achieved a more than 100 times acceleration, even compared to parallel coding in quad core.

The results can be pulled out to a text file and using IMAGEJ software to export image. After 7 MPM iterations complete and at the first EM iterarion, using current



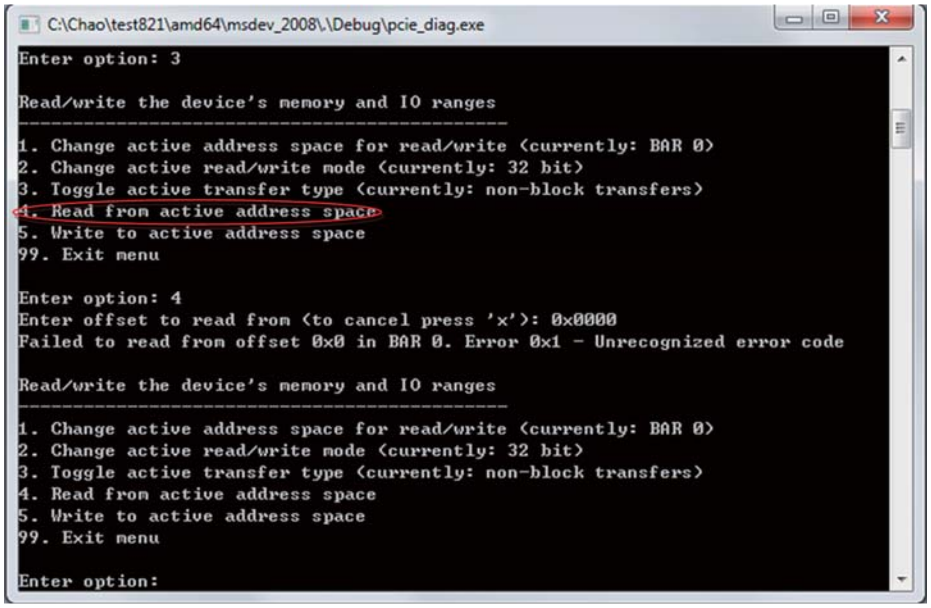
```
PCIE diagnostic utility.
Application accesses hardware using WinDriver.

Found 1 matching device [ Vendor ID 0x10EE, Device ID 0x505 ]:

 1. Vendor ID: 0x10EE, Device ID: 0x505
    Location: Bus 0x3, Slot 0x0, Function 0x0
    Memory range [BAR 0]: base 0xF0000000, size 0x1000000
    Interrupt: IRQ 10
    Interrupt Options (supported interrupts):
        Message-Signaled Interrupt (MSI)


PCIE main menu
--------------
1. Scan PCI bus
2. Find and open a PCIE device
3. Read/write memory and IO addresses on the device
4. Read/write the device's configuration space
6. Enable/disable the device's interrupts
7. Register/unregister plug-and-play and power management events
99. Exit
Enter option: _
```

Fig. 4.5. Software application on PC



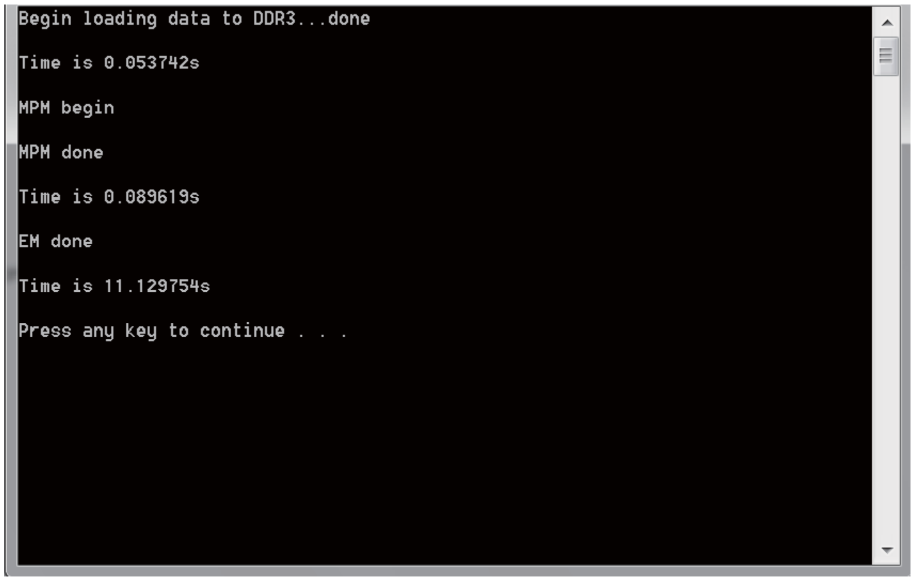
```
Enter option: 3

Read/write the device's memory and IO ranges
---------------------------------------------
1. Change active address space for read/write (currently: BAR 0)
2. Change active read/write mode (currently: 32 bit)
3. Toggle active transfer type (currently: non-block transfers)
4. Read from active address space
5. Write to active address space
99. Exit menu

Enter option: 4
Enter offset to read from (to cancel press 'x'): 0x0000
Failed to read from offset 0x0 in BAR 0. Error 0x1 - Unrecognized error code

Read/write the device's memory and IO ranges
---------------------------------------------
1. Change active address space for read/write (currently: BAR 0)
2. Change active read/write mode (currently: 32 bit)
3. Toggle active transfer type (currently: non-block transfers)
4. Read from active address space
5. Write to active address space
99. Exit menu

Enter option:
```

Fig. 4.6. Write to DDR3



```
C:\Chao\test821\amd64\msdev_2008\.\Debug\pcie_diag.exe
Enter option: 3

Read/write the device's memory and IO ranges
---------------------------------------------
1. Change active address space for read/write (currently: BAR 0)
2. Change active read/write mode (currently: 32 bit)
3. Toggle active transfer type (currently: non-block transfers)
4. Read from active address space
5. Write to active address space
99. Exit menu

Enter option: 4
Enter offset to read from (to cancel press 'x'): 0x0000
Failed to read from offset 0x0 in BAR 0. Error 0x1 - Unrecognized error code

Read/write the device's memory and IO ranges
---------------------------------------------
1. Change active address space for read/write (currently: BAR 0)
2. Change active read/write mode (currently: 32 bit)
3. Toggle active transfer type (currently: non-block transfers)
4. Read from active address space
5. Write to active address space
99. Exit menu

Enter option:
```

Fig. 4.7. Read from DDR3



```
Begin loading data to DDR3...done

Time is 0.053742s

MPM begin

MPM done

Time is 0.089619s

EM done

Time is 11.129754s

Press any key to continue . . .
```

Fig. 4.8. Time

mean and variance, one slice of the result is shown in Figure 4.9. The final result after EM convergence is shown in Figure 4.10.

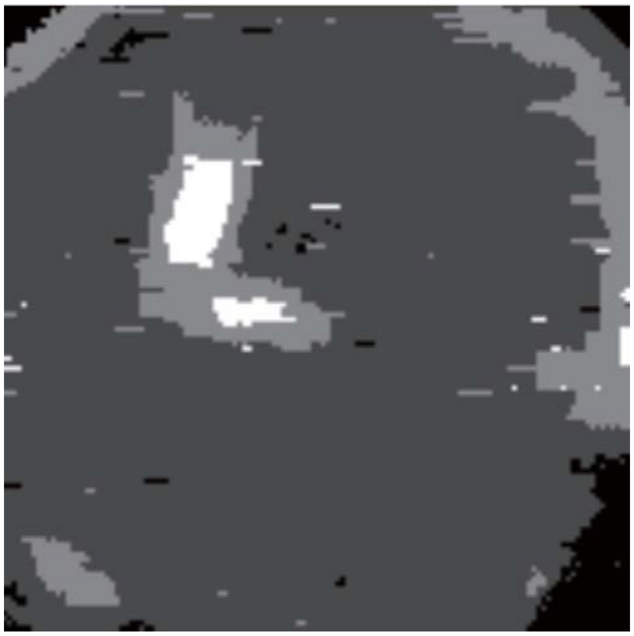


Fig. 4.9. Result of Xilinx hardware segmentation (After 7 MPM iterations)

Figure 4.11 is the final result from the standard desktop computer using software to process the same data. Two more pairs of results can be shown in Figure 4.14. It can be concluded from above images that hardware and software results are almost the same. And after EM convergence, the result is much better than only using MPM. Based on the results above, the hardware advantage is 100 times the processing speed of using software on standard desktop computer (Figure 4.12).

The processing time is also compared with the reference literatures shardware implementations based on alternate 3D segmentation algorithms. The result is shown in Figure 4.13. Taking the published data from reference [16], they chose 512*512*512 size, so we scaled down their time to 31.35ms (divided their time by 64), in order to match the 128*128*128 size for comparision to our data set. Although their system use Virtex II FPGA, after considering the Virtex-6 family consumes 15 percent less

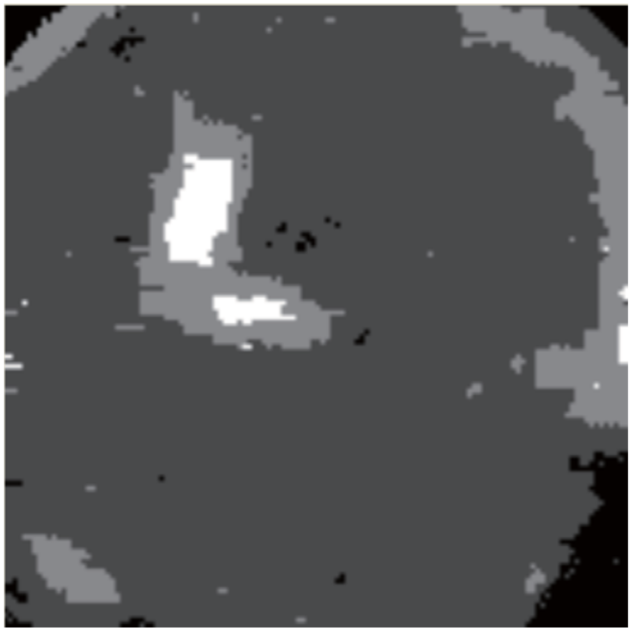


Fig. 4.10. Result of Xilinx hardware segmentation(EM/MPM both completed)

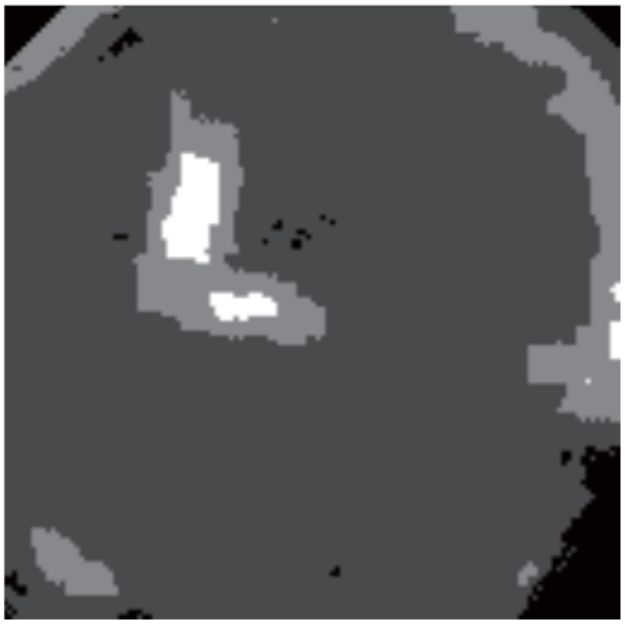


Fig. 4.11. Result of PC software segmentation

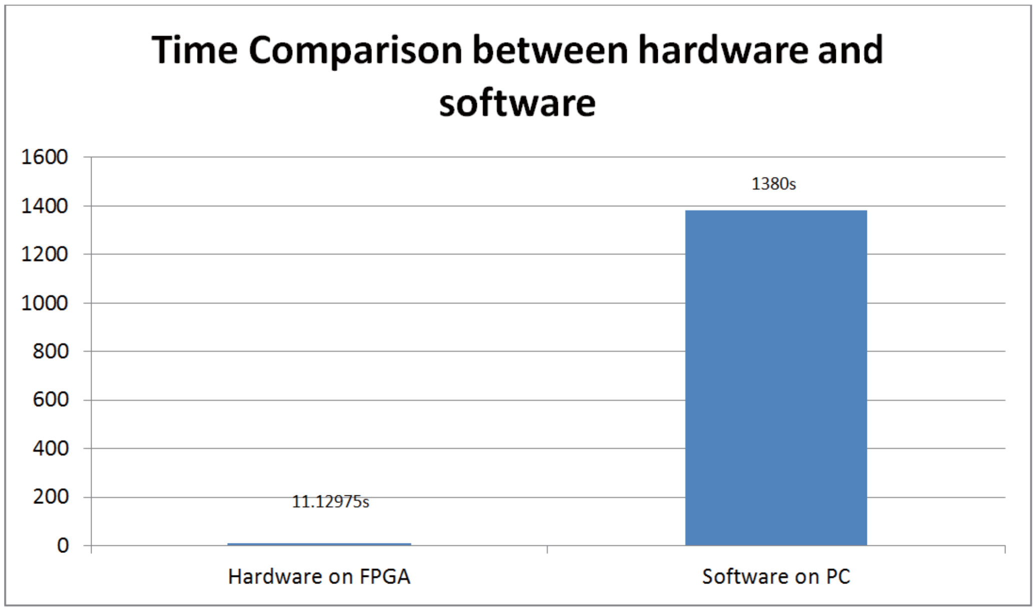


Fig. 4.12. Time comparison between hardware and software for EM/MPM

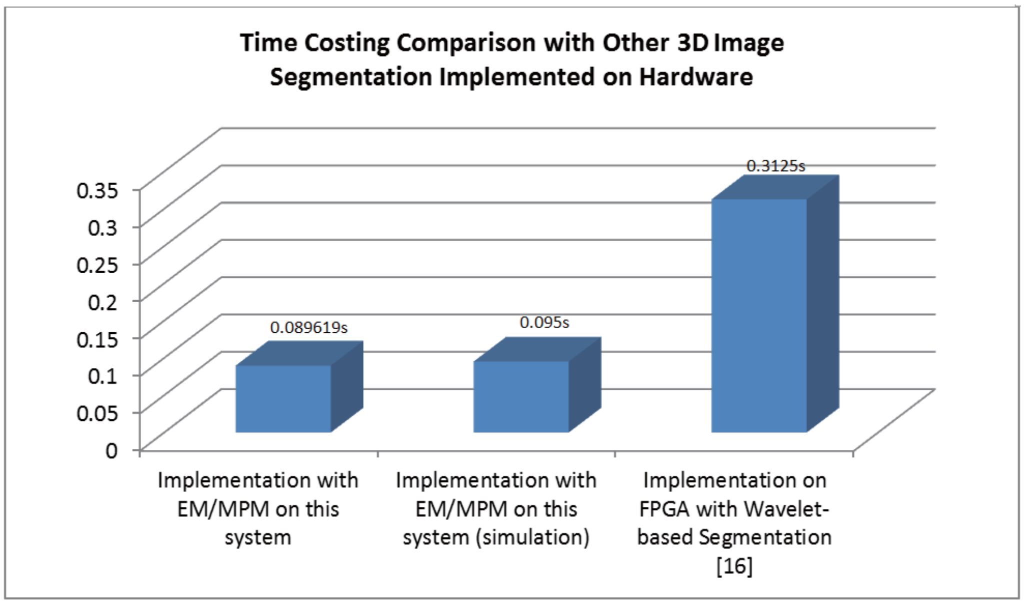


Fig. 4.13. Time comparison with other 3D image segmentation implemented in hardware

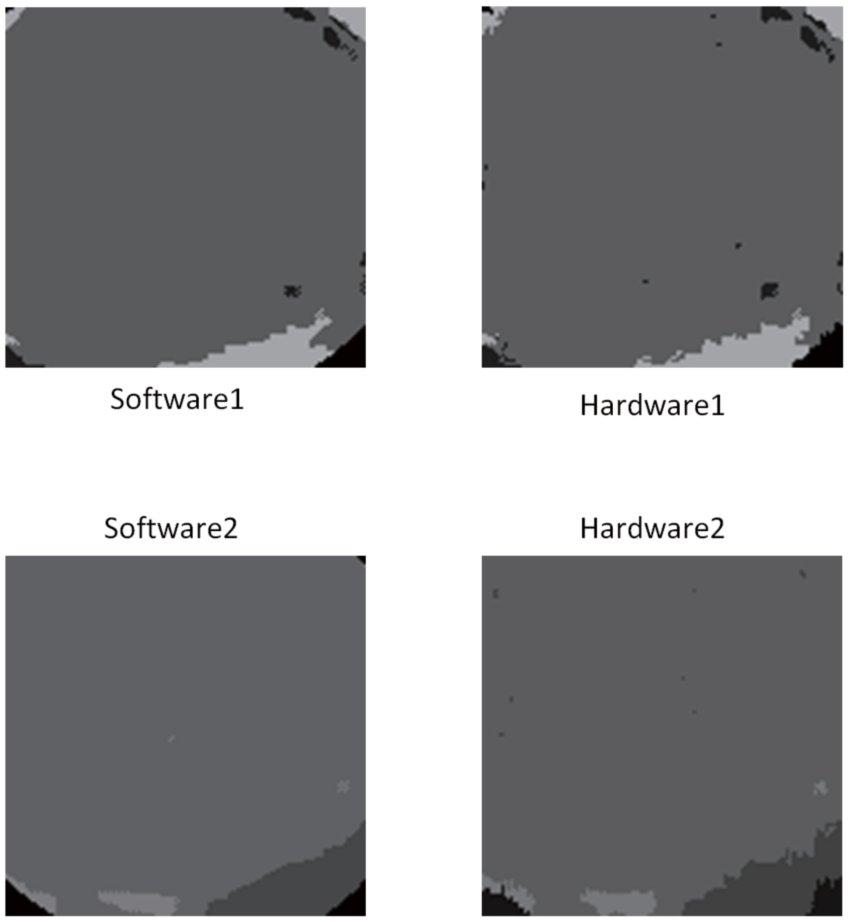


Fig. 4.14. Results comparison between software and hardware

power and has 15 percent improved performance over competing Virtex II FPGAs, it still can be seen that our hardware implementation based on EM/MPM algorithm makes a signicant acceleration againts both the software implementation and alternate hardware segmentation.

5. CONCLUSION AND FUTURE RESEARCH

5.1 Future Work

Currently our data set is 128*128*128 pixels, which is limited by LUTs and DSP48s. For larger volume 3D image, we can choose the FPGA with more resource. In future work, 256*256*128 pixels data set will be used. Xilinx XC6VSX475T is considered to be chosen. Atlas and attenuation compensation hardware can be added in the MPM block to improve segmentation accuracy. New GUI interface also will be designed to replace ImageJ tasks and a software interface which is used to connect with common medical image programs like Slicer or Visualization Toolkit (VTK) will be designed.

5.2 Conclusion

A new hardware implementation embedded design for EM/MPM algorithm has been proposed in this thesis. This system is based on Xilinx ML605 development board. This new system is designed to accelerate whole image segmentation process compared to software. Through implementing multiple computational cores on chip and Microblaze, it has been proved that this embedded system does speed up the whole 3D image segmentation process by more than 100 times and is an improvement from the literature by more than 3 times. In Chapter 1, compared with several image segmentation algorithms applied on 3D image segmentation, implementing EM/MPM algorithm on hardware was proposed because of its good performance, especially in noise. Also, it is shown that the hardware-software implementation on FPGA has several advantages compared to software solution both from processing speed and resource cost aspects. In Chapter 2, the concept of EM/MPM algorithm was briefly introduced. And in Chapter 3, critical components of this system have been discussed. Microblaze processor is responsible for the overall coordination of all the components; MPM logic is the main algorithm calculation engine. MPMC(NPI) is used to control the external memory and PCI-Express is chosen as data transfer bus between PC and the board. In Chapter 4, system synthesis report was analyzed and found out that all the resource cost is controllable and achievable on Xilinx Virtex6 development board. Then the image segmentation result is compared to image segmentation software result. The two results are essentially the same, taking into account the random variable limitations. This shows that the EM/MPM hardware design was successfully implemented on FPGA. Finally, the speed comparison between the hardware implementation and the software solution is shown. The hardware speeds up the whole 3D image segmentation process by more than 100 times compared to software, and by more than 3 times compared to other hardware segmentation results from the literature.

LIST OF REFERENCES

LIST OF REFERENCES

[1] L. Christopher, E. Delp, C. Meyer, and P. Carson, “3-D bayesian ultrasound breast image segmentation using the EM-MPM algorithm,” *Proceedings of the IEEE Symposium on Biomedical Imaging*, 2002.

[2] L. Christopher, E. Delp, C. Meyer, and P. Carson, “New approaches in 3D ultrasound segmentation,” *Proceedings SPIE and IST Electronic Imaging and Technology Conference*, 2003.

[3] M. L. Comer and E. J. Delp, “The EM/MPM algorithm for segmentation of textured image: Analysis and further experimental results,” *IEEE Transactions on Image Processing*, vol. 9, no. 10, 2000.

[4] Y. Sun and L. Christopher, “3D image segmentation implementation on FPGA using the EM/MPM algorithm,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010.

[5] X. Zhang, Y. Li, J. Wang, and Y. Chen, “Design of high-speed image processing system based on FPGA,” *Electronic Measurement and Instruments*, 2009.

[6] R. Lysecky and F. Vahid, “A study of the speedups and competitiveness of FPGA soft processor cores using dynamic hardware/software partitioning,” *Design, Automation and Test in Europe*, 2005.

[7] R. Joost and R. Salomon, “Hardware-software co-design in practice: A case study in image,” *IEEE Industrial Electronics, IECON 32nd Annual Conference*, 2006.

[8] H. Frock, M. Geruso, and M. Wetzel, “A survey of high speed bus technologies for data movement in ATE systems,” *IEEE Autotestcon*, 2006.

[9] J. Marroquin, S. Mitter, and T. Poggio, “Probabalistic solution of ill-posed problems in computational vision,” *Journal of the American Statistical Association*, vol. 82, no. 89, 1987.

[10] I. Xilinx, “Microblaze processor reference guide,” *Xilinx Document UG081(v11.1), Xilinx Inc*, 2010.

[11] I. Xilinx, “PLB IPIF,” *Xilinx Document DS448(v2.02a), Xilinx Inc*, 2005.

[12] I. Xilinx, *IP LogiCORE PLBv46 RC/EP Bridge for PCI Express.* 2010.

[13] Q. Tang, WeimingHou, WeiweiFei, HuizhiCai, and Y. Li, “The design for astronomical digital images’ real-time data acquisition and processing system based on FPGA and PCI bus,” *Testing and Diagnosis, IEEE Circuits and Systems International Conference*, 2009.

[14] I. Xilinx, "Logicore IP XPS central DMA controller," *Xilinx Document DS579(v2.02a), Xilinx Inc*, 2010.

[15] I. Xilinx, "Memory interface solution," *Xilinx Document User's Guide 086(v3.6), Xilinx Inc*, 2009.

[16] P. V.Dillinger, J. Leinen, J. Suslov, S. Patzak, R. Winkler, and H. Schwan, "FPGA based real-time image segmentation for medical systems and data processing," *Real Time Conference, 14th IEEE-NPSS*, 2005.