Architectural Design to Characterize Malignant Breast Lesion

Subarna Chatterjee
Dept. of IT, B.E.S.U., Shibpur, Howrah – 711103, India

Ajoy Kumar Ray
B.E.S.U., Shibpur, Howrah, India
Dept. of EEC, IIT Kgp, India

Rezaul Karim
Dept. of Radiodiagnosis, IPGMER, Kolkata-9, India

Arindam Biswas
Dept. of IT, B.E.S.U., Shibpur, Howrah-711103,India

ABSTRACT
Breast cancer, the second-leading cause of cancer deaths in American women, is the disease women fear most all over the world. Efficient technique for medical diagnosis is defined to provide better chance of a proper treatment. In this paper, we present a new parallel processing architecture to perform medical detection that uses multiple ultra-sono-graphic features and a diagnostic algorithm for identifying breast nodule malignancy. The algorithm has been implemented in the first phase without breaking them into macro-blocks, and in the next phase after breaking the frames into the respective macro-blocks. MATLAB has been used for the simulation of the algorithm and the results obtained are presented in this paper. We have also done the simulation and FPGA based synthesis of the proposed architecture for the most commonly used target hardware to analyze the hardware cost.

1. INTRODUCTION
Different macro imaging like X-Ray, USG and Breast MRI play a major role in detection in which Sono-mammogram is promising one. Sono-mammogram is non-invasive imaging and cost effective for mass screening.

Again using X-ray, calcification may be observed, but micro-calcification in the early stages cannot be detected properly. Moreover it has limitations when the surrounding breast tissue is dense or if the lesion is diffuse; also it is not possible to differentiate between solid and cystic lesions. Ultrasound penetrates dense breast tissue very well and is an excellent supplement whereas X-ray mammography is prohibited in young or pregnant patients. Sono-mammogram performs well in distinguishing cystic from solid masses in which accuracy is almost 96% [11]. It can be used for imaging several types of breast conditions, including both benign and malignant lesions with significant freedom in obtaining images of the breast from almost any orientation.

Ultrasound technique is used for differentiation of malignant and benign tumors in breast by evaluating several quantitative features like tumor size, shape, sharpness of margins, spiculation, calcifications, hypo-echogenicity, thick echo-genic halo, angular margins, lobulations, acoustic shadowing, duct extension, and branch pattern etc.

This study includes seventy seven sono-mammograms of benign and malignant lesions, confirmed through biopsy by onco-pathologists and radiologists.

In this paper, we have simulated an efficient image processing technique which utilizes a parallel data memory. The parallel data memory provides a faster data access for processing the image data. The proposed parallel architecture ensures high-speed operation and full utilization of the processing resources.

Main Results: We have captured ultrasound images, collected and considered those frames which are suspected by radiologists. For processing of images we have used median filter to reduce the speckle noise, unsharp masking for contrast enhancement , binary thresholding and edge detection for mass segmentation and measured mass perimeter; calculated mass centroid, mass perimeter radial samples, FFT of the radial samples at higher frequencies for detection. We simulate the algorithm without breaking the image into macro-blocks (MB), and after breaking the image into the respective MB. We also analyze the results for different USG images in both the two cases. A parallel architecture based implementation of this algorithm is proposed. The parallel architecture features scalable, minimum processing elements, minimum clock cycles, low pin count so that it can achieve high performance, flexibility and low power consumption. Lastly the hardware requirement is estimated for various well known target architectures.

The organization of the paper is as follows: Section 2 describes the Methodology. Section 3 and 4 cover Methods, and Simulation Results of the algorithm for Diagnosis. Section 5 contains the Proposed Architecture of the algorithm. Performance analysis and comparison is described in Section 6 and concluding remarks in Section 7.

2. METHODOLOGY
2.1 Ultrasonic image database
The USG images, in this paper, were provided by the IPGMER & SSKM Hospital, Kolkata, India. The database contained 77 digital ultrasonic images, composed of 46 benign and 31 malignant images confirmed through biopsy by pathologists and radiologists.

2.2 Feature extraction
USG features that most reliably characterize masses as benign, are a round or oval shape, circumscribed margin [3][4]. Features that characterize masses as malignant include irregular shape, micro-lobulations. A few gently curving, circumscribed lobulations (macro-lobulations) are considered as benign features, whereas many small lobulations of 1-2 mm (micro-lobulation) are considered malignant [3].

2.3 Image processing

2.3.1 Median Filtering

The ultrasonic images suffer from speckle noise due to interference of back-scattered signal, and this noise significantly degrades the image quality and hinders to obscure the fine details [2]. The image is preprocessed with 3 × 3 median filter.

To perform median filtering at a point in an image, we store the pixel and its neighboring pixel values in an array, sort the array in numerical order, determine their median and assign this value to the pixel. The algorithm of Median Filter is described below

$$Y(i,j) = \text{median}[\text{sort}\{ x(i,j) ,(i,j) \in w \}] \quad (1)$$

Where w is a neighborhood centered around the location (i,j) in the image.

2.3.2 Unsharp Masking

Unsharp masking is a well-known technique to enhance the edges of the structures in an image [8]. To improve the perceptibility of edges of breast nodule, 5 × 5 unsharp masking has been used after median filtering. The elements with meaningful signal level have been emphasized and the contrast between nodule and background has been enhanced.

Unsharp masking is used to sharpen image by subtracting a blurred version of an image from the image itself. The algorithm of Unsharp Masking filter is as below

$$I_{sharp}(i,j) = I(i,j) + k.g(i,j) \quad (2)$$

where $g(i,j) = I(i,j) - Ismooth(i,j)$;
$I(i , j)$ = original image;
$I_{smooth} (i,j)$ = smoothened version of $I(i,j)$;
k = scaling constant; 0.2 to 0.7;

Image is smoothened by convolving with a Gaussian filter with specified standard deviation σ, to reduce noise. Mathematically, we can write the 2-D convolution [6] as:

$$I_{smooth} (i,j) = O(i,j) = I(i,j) \otimes K(k,l) \quad (2.1)$$

Where $K(k, l)$ = convolution kernel
$I(x , y)$ = original image
$O(x , y)$ = filtered image
$m \times n$ = size of convolution kernel

2.3.3 Segmentation

After contrast enhancement, the image is converted to binary image using multi-value thresholding[12]. The threshold is determined by the histogram of the image. Thresholding produces a relatively large region together with many separated or inter-connected regions. Among the remaining objects, one closest to the center of region of interest (ROI) of the lesion area is then automatically selected as the nodule.

$$Y(i, j) = \begin{cases} 0 & \text{if } T_1 < x(i , j) \le T_2 \\ 1 & \end{cases} \quad (3)$$

2.3.4 Edge Detection

After segmentation, the object is smoothened by filling the holes inside the nodule by applying image opening and closing operation with a structuring element[8]. Finally, we get the boundary pixels of the nodule by removing interior pixels. Boundary pixels can be obtained by comparing the 4 neighboring pixel coordinates. The whole image processing procedure is illustrated in Figure 1(a- h). We simulate the algorithm without breaking the image of size 240×240 into MB, and after breaking the image into 225 MB each of size 16 × 16 matrix.

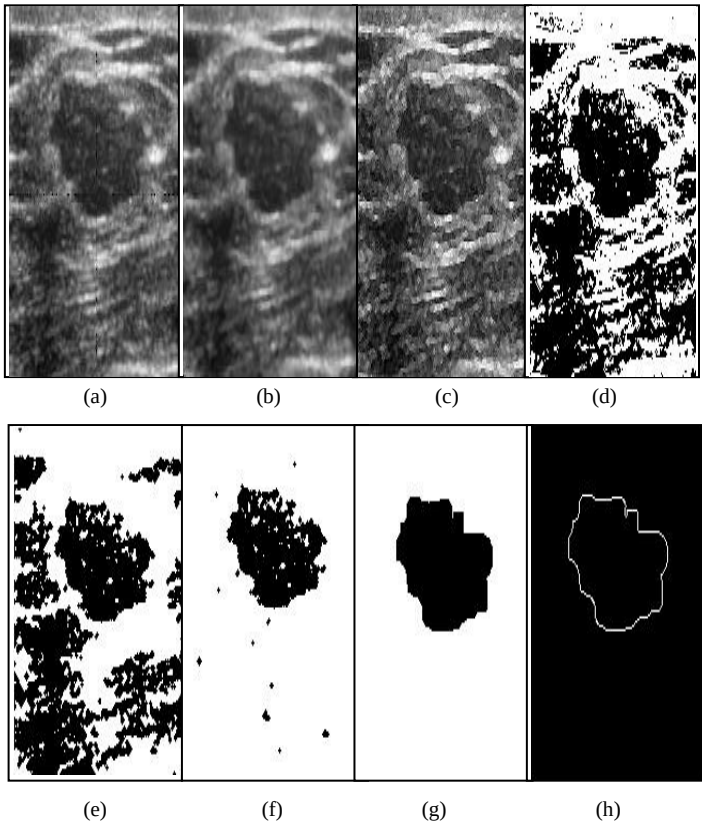


Fig 1: Image Processing Steps: (a) Original Image captured by USG device, (b) Median filtered Image, (c) Image after Unsharp Masking, (d)-(e)Images after Thresholding and (h) Detected Edge

The basic morphological operators are erosion and dilation [2].Dilation and Erosion of an image A by structuring element B is given by:-

$$D(A, B) = A \oplus B = \bigcup_{\beta \in B} (A + \beta) \quad (4)$$

$$E(A, B) = A \Theta B = \bigcap_{\beta \in B} (A - \beta) \quad (5)$$

Opening is the erosion of an image followed by dilation. On the other hand closing is the dilation of an image followed by erosion. They are given by:-

$$O(A,B) = A \circ B = (A \Theta B) \oplus B \quad (6)$$

$$C(A,B) = A \bullet B = (A \oplus B) \Theta B \quad (7)$$

3. METHODS

3.1 Radial distance signal

The radial distance function is computed in two steps - i) the centroid of the mass is computed, and ii) the acquired centroid of the mass is identified as the center of a polar coordinate system and the mass borderline is sampled to extract the radial distance sequence:

$$r(i) = \sqrt{(x(i) - X_0)^2 + (y(i) - Y_0)^2} \qquad (8)$$

$$\theta(i) = \tan^{-1}\left(\frac{y(i) - Y_0}{x(i) - X_0}\right) \qquad (9)$$

Where i = 1, 2, . . . ,N ; The point (X_0, Y_0) is the acquired centroid of the mass and N is the number of borderline samples.

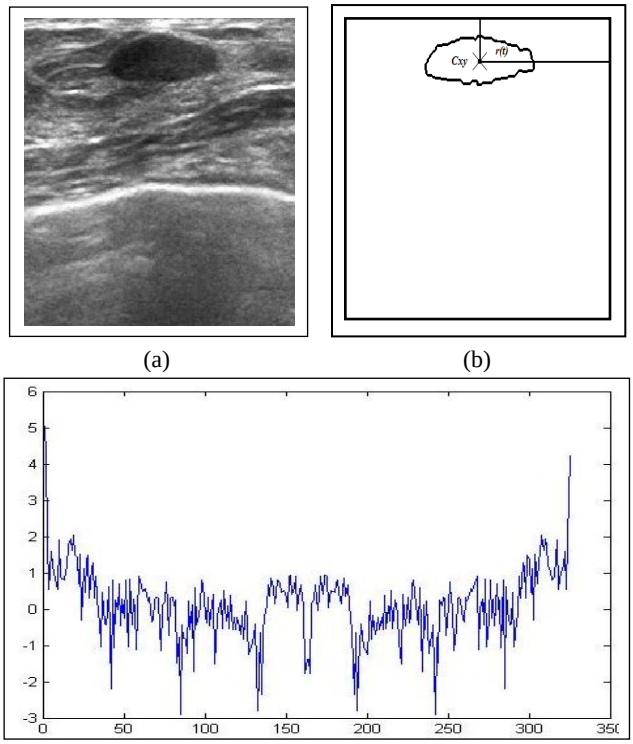
3.2 FFT Calculation

The Fast Fourier Transform (FFT) F(i) is calculated by the following formula:

$$F(i) = \sum_{n=1}^{N} r(n) \times e^{\frac{-j2\pi(n-1)(i-1)}{N}} \qquad 1 \le i \le N. \qquad (10)$$

Where $r(n)$ is the radial distance function. We use FFT to convert an image into its frequency domain to remove unwanted frequency information before analyzing and processing the image. We consider only those samples that are in high frequency range, between π/4 to π.

4. SIMULATION RESULTS

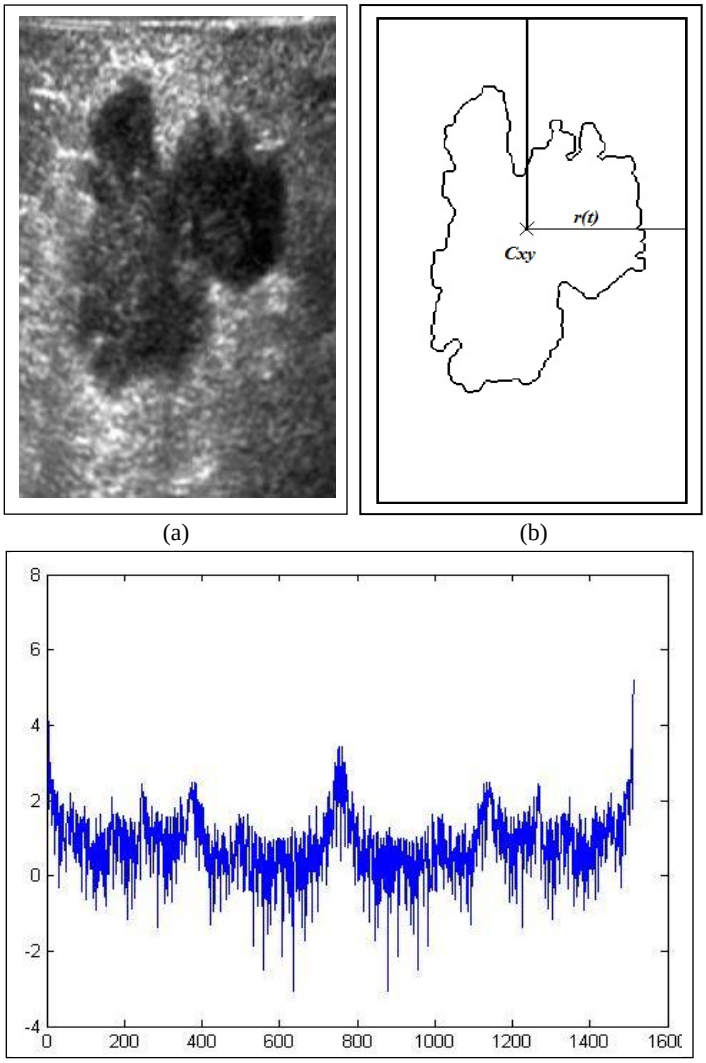


(a) (b)

(c)

Fig 2: Typical example of Benign nodule: (a) Original Image, (b) Detected Edge, (c) Amplitude of FFT, R(ω)

We simulate the algorithm, and after breaking the image into 225 MB each of size 16 × 16 matrix. We calculate the centroid of the mass, radial distance and FFT at high frequency range, between π/4 to π in the 1st case. In the 2nd case the total number of borderline pixel (BP) is 1536. We divide it into 6 MBs of size 16×16 and calculate the centroid of mass, radial distance and FFT at high frequency range, between π/4 to π. In both the two cases we analyze the results for seventy seven USG images and get the same results.



(a) (b)

(c)

Fig 3: Typical example of malignant nodule with: (a) Original Image, (b) Detected Edge, (c) Amplitude of FFT, R(ω)

Figure 2 and Figure 3 display the original image, segmented mass with its centroid, and log amplitude of fast Fourier transform R(ω)[at high frequency range(π/4 to π)] of benign and malignant nodule, respectively. The amplitude spectrum of FFT of the malignant nodule[Figure 3(c)] contains larger high-frequency components (π/4 to π) and sharp peak than that of the benign nodule[Figure 2(c)].

5. PROPOSED ARCHITECTURE FOR DIAGNOSIS

5.1 Overview of the Architecture

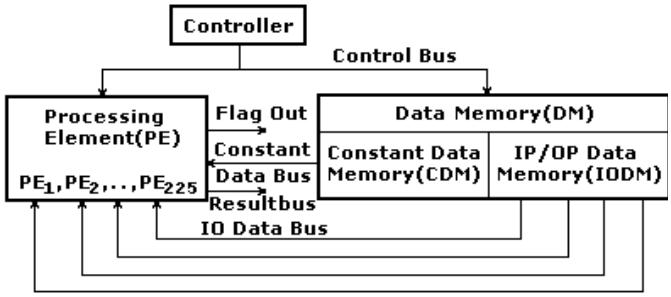


Fig 4: Architectural description of the controller

To propose hardware architecture for this medical detection algorithm, we need a very high amount of hardware resources. Complexity can be reduced by parallelizing the operation needed for diagnosis.

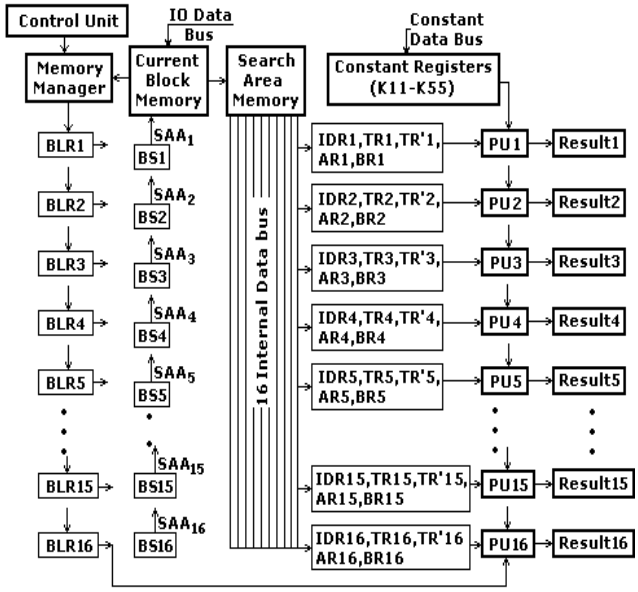


Fig 5: Architecture of each PE for macro-block of size 16×16

The basic building blocks of the architecture of the Controller and Processing Elements (PE) are as shown in Fig.4 & Fig. 5 This architecture is divided into 225 PEs.

When architecture starts, controller activates PEs & memory manager (MM). According to the operation activated by the control unit, block selectors (BS_0 to BS_{16}) of each PEs generates the search area address (SAA_1 to SAA_{16}) and send it to the current block memory (CBM) and MM reads one line of the CBM. Data stored in the SAA of CBM are sent to the search area memory (SAM) and data read by MM from CBM are stored in the block line register (BLR_0 to BLR_{16}) of each Processing Elements (PE_1, PE_2, PE_3,….PE_{225}). Now Processing Units (PU_1 to PU_{16}) for each PEs get the data from SAM for processing.

Each PE consists of a CU, memory manager (MM), Processing Units (PU_1 to PU_{16}), block selectors (BS_1 to BS_{16}), current block memory (CBM), search area memory (SAM), block line register (BLR_1 to BLR_{16}), ALU (ALU_1 to ALU_{16}), comparator and six sets of registers. These registers are function dependent constant registers (K_{11}–K_{55}), input data registers (IDR_1–IDR_{16}), temporary registers (TR_1–TR_{16}), temporary registers (TR^1_1–TR^1_{16}), accumulator registers (AR_1–AR_{16}), buffer registers (BR_1–BR_{16}), inter buffer registers (IBR_{11}–IBR_{1648}) and output result registers ($Result_1$…$Result_{16}$) and ($Result_{11}$…..$Result_{116}$). The architecture allows parallel loading of data into all the six sets of registers, execution of ALU operations and storing the output results concurrently, providing an efficient configuration for implementing DSP functions like median filter, unsharp filter, edge detection, FFT, etc. To exploit inherent parallelism exhibited by DSP functions, a number of such PEs are connected through an inter PE router for data communication. Registers may be used to preserve the values of intermediate results for any possible subsequent operation(s) when needed.

The register sets along with the connections are built up to maintain maximum flexibility and performance at minimum control requirements for all DSP functions. Data memory (Fig. 4) is divided into two parts, Constant Data Memory (CDM) and Input / Output Data Memory (IODM). The constant data are loaded into the registers (K_{11}–K_{55}) via constant data bus and input data are loaded into Current Block Memory (CBM) of each PEs. The constant memory holds only the function dependent constants required by that particular PE. The IODM hold the incoming data from the external world and the output data from the 'result buses'. By separating constant data from the input/output data, the overall data bus bandwidth increases, thereby increasing the processor performance.

5.2 Hardware Estimation

5.2.1 Hardware Estimation of Median Filtering

For 3x3 median filter, suppose that the search range $[-p, p] = [-1, 1]$. The search is restricted to a specific area in the frame which is called search area. Pixel coordinates of both search area and the reference blocks (Figure 6) are shown below.

lx (x,y) = median(sort [in(x-1,y-1), in(x-1,y), in(x-1,y+1), in(x,y-1), in(x,y), in(x,y+1), in(x+1,y-1), in(x+1,y), in(x+1,y+1)]);

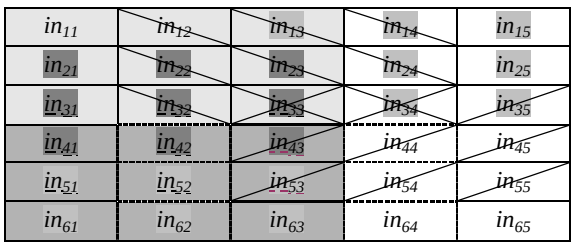


Fig 6: The reference blocks and its search area

From calculation it seems that there is an overlapped search range between the search areas belonged to the reference block *lx(2,2) and lx(2,3). So* if the reference block *lx(2,2)* and *lx(2,3)* begin to conduct the process at the same time we need temporary buffers to store the sorted pixel values.

Here we consider 225 Processing Elements (PE). At 1st clock 16×3 bit value is loaded to the input data registers (IDR_1–IDR_{16}), temporary registers (TR_1–TR_{16}), and temporary registers (TR^1_1–TR^1_{16}) at 3rd clock sorted value of the registers (IDR_1–IDR_3), (TR_1–TR_3), and (TR^1_1–TR^1_3) are stored in the inter buffer registers (IBR_{11}–IBR_{19}) (here we consider mesh network) and at 4th clock median value of the registers (IBR_{11}–IBR_{19}) are stored in the output result registers ($Result_1$). We get the 1st sampled result of the 1st column of the 16 × 16 bit value at 4th clock cycle.

Time complexity in this case is = *4* + (*16* −*1*) = *19* clock cycles;
Time complexity for image size *240×240* is = T(*240×240*)

= *4* + (*240* −*1*) = *243* clock cycles;

For general case, time complexity = T(*N×N*) = *4* + (*N* − *1*) = *N* + *3* ≈ *N* clock cycles;

5.2.2 Hardware Estimation of Unsharp Masking

From eq(2.1) putting i=j=1 we get,

$$O(1,1) = \sum_{k=1}^{m}\sum_{l=1}^{n} I(k,l)K(k,l)$$

putting m = n = 5 the size of the convolution kernel we get,

$O(1,1) = I_{11}K_{11} + I_{12}K_{12} + I_{13}K_{13} + I_{14}K_{14} + I_{15}K_{15} + I_{21}K_{21} + I_{22}K_{22} + I_{23}K_{23} + I_{24}K_{24} + I_{25}K_{25} + I_{31}K_{31} + I_{32}K_{32} + I_{33}K_{33} + I_{34}K_{34} + I_{35}K_{35} + I_{41}K_{41} + I_{42}K_{42} + I_{43}K_{43} + I_{44}K_{44} + I_{45}K_{45} + I_{51}K_{51} + I_{52}K_{52} + I_{53}K_{53} + I_{54}K_{54} + I_{55}K_{55}$

$= Y_{11}+Y_{21}+Y_{31}+Y_{41}+Y_{51}$

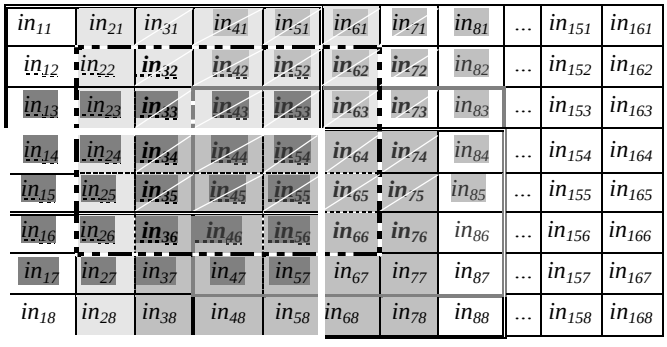


Fig 7: The reference blocks and its search area

Figure 7 shows that there is an overlapped search range between the search areas belonging to the reference blocks *O(1,1), O(2,1) and O(1,2).* Reference blocks *O(1,1), O(2,1), O(3,1)… O(15,1)* and *O(16,1)* begin to conduct the process at the same time; so we need temporary buffers to store the pixel values in-order to reuse the data of the overlapped range. *O(1,2)* starts execution after delay of one clock cycle to the block *O(1,1). O(1,2)* reuses the data of the overlapped range, so no additional input bus is needed. Here also we need 225 Processing Elements (PE).

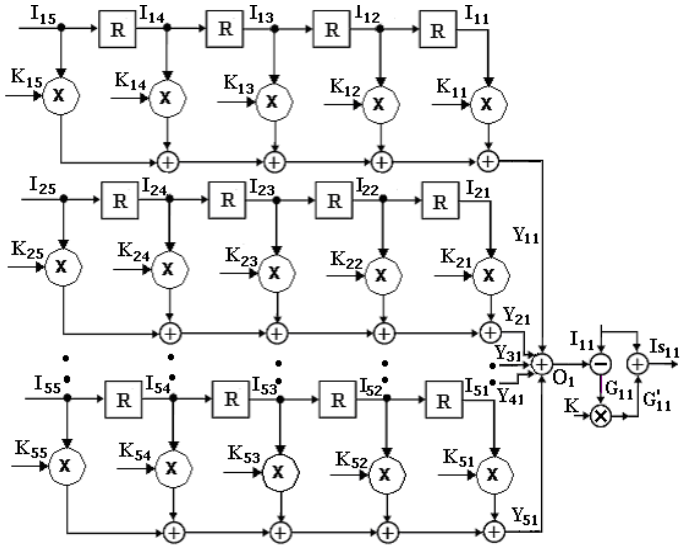


Fig 8: Operation performed in each PU for 5x5 Unmask Filter

From eq (2) putting i=j=1 we get, $I_{sharp}(1,1) = I_s(1,1) = I(1,1) + G'(1,1) = I(1,1) + k.G(1,1) = I(1,1) + k.[I(1,1) - O(1,1)];$

Figure 8 shows the operation performed in each PUs (PU_1 to PU_{16}) for the Unsharp Filter for kernel size 5×5; it simply maps each arithmetic operator in the algorithm into a separate hardware element. The filter comprised of 5 x 4 = twenty delay operators (20 D FF), 5×5 +1 = twenty six multipliers, and 5 x 4 + 2 = 22 adder and one sub-tractor.

The execution take ($T_m+4T_a+T_a+T_s+T_m+T_a$) = $2T_m+6T_a+T_s$ clock cycle, where T_a = addition time, T_s = subtraction time, and T_m = multiplication time, let $T_s = T_m = T_a = 1$ clock cycle; then $2T_m+6T_a+T_s = 2+6+1 = 9$ clock cycle when the size of kernel is *5* by *5* matrix.

At 1st clock *16×5* bit value is loaded to the input data registers (IDR_1–IDR_{16}), temporary registers (TR_1 – TR_{16}), (TR^1_1–TR^1_{16}), accumulator registers (AR_1–AR_{16}), buffer registers (BR_1 – BR_{16}). Calculated values are stored in the inter buffer registers (IBR_{11}–IBR_{1648}). At *9*th clock output result of the registers (IBR_{11}–IBR_{1648}) are stored in the output result registers ($Result_1$).We get the 1st sampled result of the 1st column of the *16 × 16* bit value at *9*th clock cycle.

Time complexity for UF (*16 ×16* MB) is = *9* + (*16* − *1*) = *24* clock cycles;

Time complexity for image size *240×240* is = T(*240×240*)

$= 9 + (240 - 1) = 248$ clock cycles;

For general case, time complexity = $T(N \times N) = 9 + (N - 1)$ $= N + 8 \approx N$ clock cycles;

5.2.3 Hardware Estimation of Edge Detection

Figure 9 shows the operation performed in each PUs (PU_1 to PU_{16}) for the Dilation & Erosion operation for kernel size 5x5. The Dilation filter comprised of four delay operators (4 D FF), five adder, and four 'or' operators and the Erosion filter comprised of four delay, five sub-tractor, four 'and' operators.

From figure we see that execution take , $(T_a+4T_{or}+T_{or}+T_s+ 4T_{and} +T_{and}) = T_a+5T_{or}+T_s+5T_{and}$ clock cycle,

where T_a = addition time, T_s = subtraction time, and T_{and} , T_{or} = time needed by and, or gate; let $T_s = T_a = 1$ clock cycles, T_{and}, T_{or} = 1 clock cycles ; then $T_a+5T_{or}+T_s+5T_{and} = 1+5+1+5 = 12$ clock cycles when the size of kernel is 5 by 5 matrix.

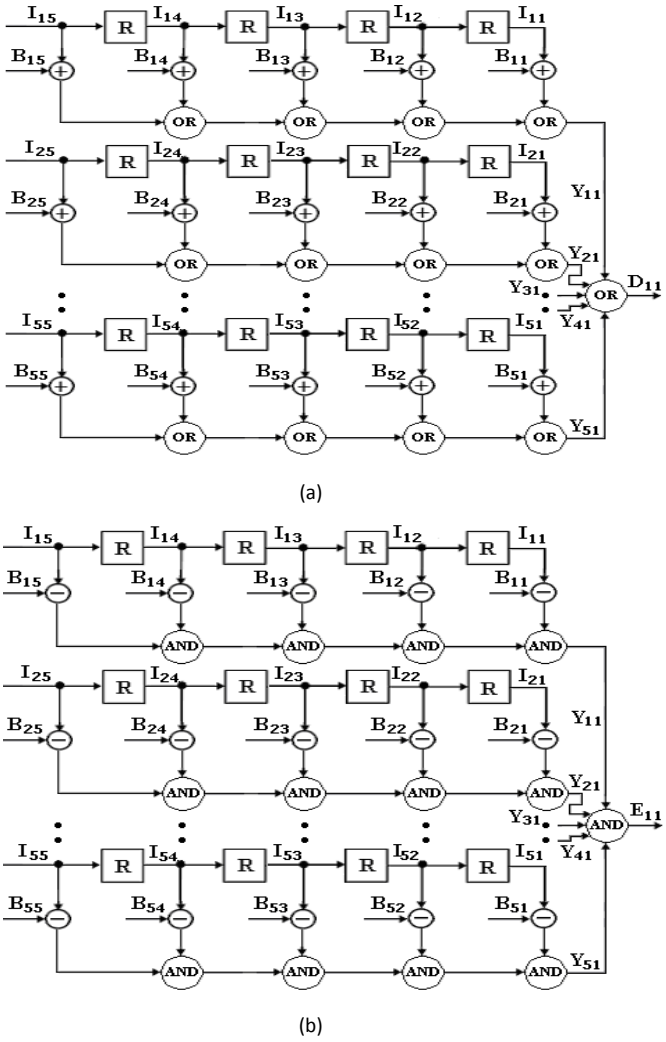


Fig 9: Operation performed in each PU for kernel size 5x5 (a) Dilation & (b) Erosion

Time complexity for opening and closing operation, i.e. for 16×16 MB is $= 2 \times (12 + (16 - 1)) = 54$ clock cycles;

Time complexity for image size 240×240 is = $T(240 \times 240) = 2 \times (12 + (240 - 1)) = 502$ clock cycles;

For general case, time complexity = $T(N \times N)$

$= 2 \times (12 + (N - 1)) = 2N + 22 \approx N$ clock cycles;

Boundary pixels can be obtained by comparing the 4 neighboring pixel coordinates as:-

	x , y + 1	
x − 1, y	x , y	x + 1, y
	x , y − 1	

$O(2,2) = [in(1,2), in(2,2), in(3, 2), in(2, 1), in(2,3)]$;

It needs 5 comparisons to find the borderline pixel coordinates (BPC). At the same time we use five register to store the X & Y coordinates of the borderline pixels, the sum of the X & Y coordinates of the borderline pixel, and the rest is a counter to count the number of BPC. We calculate the *Xp* & *Yp* by dividing the 3rd & 4th register value by the 5th register value. The centroid (X*o*, Y*o*) is calculated by taking the sum of X*p* & Y*p* of all the PEs.

At 1st clock 16×5 bit value is loaded to the input data registers (IDR_1−IDR_{16}), temporary registers (TR_1−TR_{16}), (TR^1_1−TR^1_{16}) accumulator registers (AR_1−AR_{16}), buffer registers (BR_1−BR_{16}). After comparison the 1st sample *O(2,2)* is generated and stored in the output result registers ($Result_1$) at 6th clock cycle. We use inter buffer registers (IBR_{11}-IBR_{162}) to store the X & Y coordinates of the borderline pixels, (IBR_{13} & IBR_{14}) the sum of the X & Y coordinates of the borderline pixel, IBR_{15} the number of BPC. We calculate the *Xp* & *Yp* by dividing the (IBR_{13} & IBR_{14}) value by the IBR_{15} value and store it in IBR_{16} and IBR_{17}. Now we add the IBR_{16} and IBR_{17} value for 225 PEs and store the value (centroid) in the output result registers ($Result_{11}$, $Result_{12}$).

Time complexity for boundary and centroid calculation (image size $N \times N$) = $T(N \times N) = (6 + (N - 1) + 2) = 7 + N - N$ clock cycles;

5.2.4 Hardware Estimation of Radial distance signal

As the number of borderline pixel (BP) is much less than the number of image pixel we consider BP only. During the estimation of borderline pixels we also estimate the centroid of the mass and save them. Total number of BP is 1536 for *225* PEs. We divide the BP into 6 MBs of size *16×16* and send the X & Y coordinates of the BP, centroid of mass to each PE. Here we need only 6 PEs instead of 225 PE. As a result the rest of the

PE can be used to do some other job. The acquired centroid of mass is identified as the center of polar coordinate system and the radial distance sequence and angle (θ) at every point is calculated.

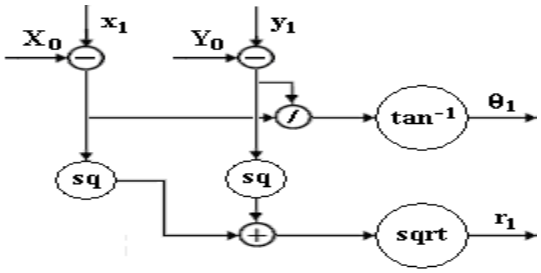


Fig 10: Operation performed for Radial distance signal

Figure 10 shows the detailed operation performed for radial distance signal. From figure we see that execution take ($T_s + T_{sq} + T_a + T_{sqrt}$) clock cycle, where T_a = addition time, T_s = subtraction time, and T_{sq} ,T_{sqrt} = square and square-root time; let $T_s = T_a = T_{sq}$,$T_{sqrt} = 1$ clock cycles; then $T_s + T_{sq} + T_a + T_{sqrt}$ $= 1+1+1+1 = 4$ clock cycles.

Here we use input data registers (IDR_1–IDR_{16}), temporary registers (TR_1–TR_{16}), (TR^1_1–TR^1_{16}) to store the value of centroid of mass and X & Y coordinates of each BP, at 4th clock the 1st sample of radial distance and angle(θ) at every point are calculated and stored in the output result registers ($Result_1$, $Result_{11}$).

Time complexity in this case, i.e. for *16 ×16* MB is

$= 4 + (16 - 1) = 19$ clock cycles;

Time complexity for image size *240×240* is

$= T(240\times240) = 4 + (240 - 1) = 243$ clock cycles;

For general case, time complexity = $T(N\times N)$

$= 4 + (N - 1) = N + 3 \approx N$ clock cycles;

5.2.5 Hardware Estimation of FFT

The computation is performed in 4 stages using 8 PU. Addition and multiplication after subtraction are performed using same data input in each stage. So ($T_s + T_m$) clock cycle is required in each stage, where T_a = addition time, T_s = subtraction time, and T_m = multiplication time; we get the first sample output after 4 × ($T_s + T_m$) + $3T_r$ clock cycle, where T_r = routing time, let $T_s = T_m = T_r = 1$ clock cycles, then 4 × ($T_s + T_m$) + $3T_r$ = 8+3 = 11 clock cycles.

Here we use input data registers (IDR_1–IDR_{16}) to store radial distance at every point, at 11th clock the 1st sample of FFT amplitude is calculated and stored in the output result registers ($Result_1$, $Result_{11}$).

Time complexity for FFT, i.e. for *16 ×16* MB is $T(16\times16)$

$= 11 + (16 - 1) = 26$ clock cycles;

Time complexity for image size *240×240* is = $T(240\times240) = 11 + (240 - 1) = 250$ clock cycles;

For general case, time complexity = $T(N\times N) = 11 + (N - 1)$ $= 10 + N \approx$ N clock cycles;

Total Time Complexity in parallel execution (PE) in this case is $= (N + 3)+(N + 8)+(2N + 22) + (N + 3)+ (N + 10) = 6N + 46 \approx$ N clock cycles;

6. SYNTHESIS RESULTS

We simulate the architectural block of Median filter, Unsharp Masking, Edge Detection, Estimation using Xilinx-ISE6 and synthesize on different device family. The synthesis reports are shown in Table 1 and Table 2. The percentage value in the parenthesis shows utilization ratio of the amount of available resources.

Table 1
Device utilization summary for 5x5 Unsharp Filter

	Spartan-3 S50	Virtex - v50	Virtex-II V2000	Virtex-II Pro VP70	Virtex-IV VFX60
# Slices	372 (48%)	787(102%)	372 (3%)	372 (1%)	350 (1%)
# Slice FFs	240 (15%)	240(15%)	240 (1%)	240 (0%)	200 (0%)
# 4 input LUTs	498 (32%)	1283(83%)	498 (2%)	498 (0%)	498 (0%)
# bonded IOBs	50 (40%)	50 (27%)	50 (8%)	50 (5%)	50 (12%)
# GCLKs	25 (625%)	1 (25%)	25 (44%)	25 (7%)	1 (3%)
Memory Usage	76804 KB	78196 KB	113584KB	217284KB	238472KB

Table 2
Device utilization summary

	FFT	USF	MF	EDF
#Slices(33088)	409(1%)	395(1%)	390(1%)	1088(3.29%)
#Slice FFs(66176)	390(0%)	272(0%)	288(0%)	840(1. 27%)
#4 I/P LUTs(66176)	600(0%)	538(0%)	768(1%)	1276(1.93%)
#bonded IOBs(964)	49(5%)	50(5%)	130(13%)	200(20.7%)
#MULT18x18s(328)	8(2%)	25(7%)	0	0
#GCLKs(16)	1(6%)	1(6%)	1(6%)	1(6%)
Memory Usage	214680KB	215256KB	168088KB	65760KB
CPU Time	72. 57MHz	83.29MHz	200.64MHz	182.43MHz

The synthesis results of the proposed architecture are summarized in Table 2. The synthesis was targeted to Xilinx Virtex–II Pro VP70 FPGA and the ISE synthesis tool was used [10]. The synthesis results indicated that the designed architecture used 2282 Slices (6.896% of total device resources), 1790 Slice FFs(2.7% of total device resources), 3182 LUT (4.81% of total device resources), 429 IOB (44.50% of total device resources), 33 MULT18x18 (10.06% of total device resources).

The synthesis results show that the designed architecture can reach the processing rate of PAL frames (640×480 pixels). Higher search range will allow better quality results when processing high resolution videos. The performance can be

better if the level of parallelism is increased or if a faster target device is used. The architecture is scalable, and as the mass is not fixed, we can implement filter of different order.

7. CONCLUSION

This paper presents FPGA based hardware architecture for medical diagnosis. The detection algorithm determines breast nodule malignancy using digital image processing and statistical calculation based on sonographic features.

This architecture considers blocks with 16x16 samples and it uses a search area with 16x16 samples, a search range of [−1, +1] or [−4, +5] or [−15, +16]. This solution was specifically designed to meet the requirements of PAL with 640x480 pixels per frame.

Our future work is to develop a classifier system using Artificial Neural Network and estimate to the accuracy for classifying benign and malignant tumors on USG.

8. REFERENCES

[1] "Mayo Clinic Guide to Women's Cancers", Mayo Foundation for Medical Education and Research (MFMER), 1998-2008.

[2] S. Joo1, W. K. Moon , H. C. Kim, "Computer-aided Diagnosis of Solid Breast Nodules on Ultrasound with Digital Image Processing and Artificial Neural Network" IEEE 26th Annual International Conference of the IEEE EMBS, San Francisco, CA, USA, pp. No 1397-1400, Sept, 2004.

[3] A.R. Pande, B. Lohani, P. Sayami, S. Pradhan, "Predictive value of ultrasonography in the diagnosis of palpable breast lump" Kathmandu University Medical Journal, Vol. 1, No. 2, pp. No 78-84, 2003.

[4] B.D. Fornage, N. Sneige, M.J. Faroux, E. Andry, "Sonographic appearance and ultrasound-guided fine-needle aspiration biopsy of breast carcinomas smaller than 1 cm3" Medical Ul-trasound Journal, Vol-9, pp. No 559-568, 1990.

[5] D. Zhang G. Lu "An Edge and Color Oriented Optical Flow Estimation Using Block Match-ing" Gippsland School of Comp & Info Tech Monash University Churchill, Victoria 3842.

[6] H.S. Neoh, A.Hazanchuk "Adaptive Edge Detection for Real-Time Video Processing using FPGAs".

[7] S. Kittitornkun ,Y.H.Hu "Frame-Level Pipelined Motion Estimation Array Processor" IEEE Trans. on Circuits and Systems for Video Technology, Vol. 11, No. 2, Feb. 2001.

[8] J. C. Tuan, T. S. Chang, and C. W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," IEEE Transactions on Circuits and Sys-tems for Video Technology, Vol. 12, No. 1, 61-72,January 2002.

[9] H. W. Feng, B. Y. Long, M. Z. Gang "Efficient Frame-Level Pipelined Array Architecture for Full-Search Block-Matching Motion Estimation" IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005. Vol. 3,2887- 2890, May 2005.

[10] Xilinx INC. Xilinx: The Programmable Logic Company. Disponível em (2006), www.xilinx.com.

[11] T. Mansoor, A. Ahmad, S. H. Harris, Ahmad "Role of Ultrasonography in the Differential Diagnosis of Palpable Breast Lump" Indian Journal of Surgery, Vol. 64, No. 6, Nov - Dec. 2002, pp. 499-501.

[12] T. Acharya, A. K. Ray, Image Processing Principles and Applications, A Wiley-Interscience Publication, USA , 2005.Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems.