Progress report On

VLSI DESIGN OF FREQUENCY DOMAIN DENOISING FILTERS AND REALIZATION USING FPGA

Submitted for Partial Fulfillment of the Requirements

For the Degree of

Master of Engineering

In

Electronics & Telecommunication Engineering

(Specialization: Digital System & Instrumentation)

By

SHIRSHENDU ROY

(Registration No: 141500178, Exam Roll No: 320714001)

Under the Guidance of

Dr. AYAN BANERJEE

Associate Professor

Department of Electronics & Telecommunication Engineering

Department of Electronics & Telecommunication Engineering

INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, Shibpur, Howrah – 711103

DEC 2015

INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, Shibpur

HOWRAH-711103

Forward

*I hereby forward the Progress report entitled "**VLSI design of Frequency Denoising Filters and Realization Using FPGA"** prepared by Shirshendu Roy under my guidance and supervision in partial fulfillment of the requirements for the degree of **Master of Engineering** in **Department of Electronics & Telecommunication Engineering** with specialization in **Digital System and Instrumentation** from **INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, Shibpur, Howrah- 711103***

.... ***(Dr Ayan Banerjee)***

Associate Professor

Countersigned by *Electronics & Tele-Communication*

…………………………….. ....

(Dr. Santanu Das) ***(Dr. Amit Kr Das)***

Professor and Head *Dean Academic*

Department of electronics and Tele-communication Engineering IIEST, Shibpur

IIEST, Shibpur

INDIAN INSTITUE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR

HOWRAH- 711103

CERTIFICATE OF APPROVAL

The forgoing progress report is hereby approved as a creditable study of an engineering subject titled "**VLSI DESIGN OF FREQUENCY DOMAIN DENOISING FILTERS AND REALIZATION USING FPGA**" carried out and presented satisfactorily to warrant its acceptance as a pre- requisite to the Degree of ***Master of Engineering*** of this University. It is understood that by this approval the undersigned do not necessarily approve any statement made, opinion expressed and conclusion drawn therein but approve the Progress report only for the purpose for which it is submitted.

Countersigned by:

……….……….…..………………

Dr. AYAN BANERJEE

(Thesis Supervisor)
Associate Professor
Department of Electronics & Telecommunication Engineering,
IIEST, Shibpur
Howrah-711103

Board of Examiners:

(1) ……………………….…….

(2) ……………………….…….

INDIAN INSTITUTE OF ENGINEERING & SCIENCE TECHNOLOGY, SHIBPUR
HOWRAH – 711103

ACKNOWLEDGEMENT

I, hereby, want to take the opportunity to express my profound gratitude and respect to **Dr. Ayan Banerjee**, Associate Professor, Department of **Electronics & Telecommunication Engineering**, Indian Institute of Engineering Science and Technology, Shibpur, for his valuable advices, resourceful guidance, active supervision and constant encouragement without which it would not have been possible to submit the Progress report in such a shape in time.

I also want to express my gratitude towards Dr. Santanu Das, Professor and Head, Department of Electronics and Tele-communication Engineering, Indian Institute of Engineering Science and Technology, Shibpur and all those who offered helping hands to complete this Progress report directly or indirectly.

I also thankfully acknowledge the assistance received from my friends and others for their cooperation during the preparation of the Progress report.

Date: …………………… ………………………………………………

Shirshendu Roy

*Registration No:**141500178***
Exam Roll No:320714001
IIEST, Shibpur
Howrah-711103(W.B)

INDEX

Introduction

DIGITAL image processing is an ever expanding and dynamic area with applications reaching out into our everyday life such as medicine, space exploration, surveillance, authentication, automated industry inspection and many more areas. Applications such as these involve different processes like image enhancement, image restoration and object detection.

During image acquisition and processing images are often corrupted by various kind of noises. Lost of information through Image degradation is a major problem in image processing. Restoration of an image from its degraded version is itself a challenging task. So many algorithms for image restoration for various kind of noises have been developed. Implementing such algorithms on a general purpose computer can be easier, but not very time efficient due to additional constraints on memory and other peripheral devices. Application specific hardware implementation offers much greater speed than a software implementation. With advances in the VLSI (Very Large Scale Integrated) technology hardware implementation has become an attractive alternative. Implementing complex computation tasks on hardware and by exploiting parallelism and pipelining in algorithms yield significant reduction in execution times.

There are two types of technologies available for hardware design. Full custom hardware design also called as Application Specific Integrated Circuits (ASIC) and semi custom hardware device, which are programmable devices like Digital signal processors (DSPs) and Field Programmable Gate Arrays (FPGA’s). Full custom ASIC design offers highest performance, but the complexity and the cost associated with the design is very high. The ASIC design cannot be changed and the design time is also very high. ASIC designs are used in high volume commercial applications. In addition, during design fabrication the presence of a single error renders the chip useless. DSPs are a class of hardware devices that fall somewhere between an ASIC and a PC in terms of the performance and the design complexity.

Field Programmable Gate Arrays are reconfigurable devices. Hardware design techniques such as parallelism and pipelining techniques can be developed on a FPGA, which is not possible in dedicated DSP designs. Implementing image processing algorithms on reconfigurable hardware minimizes the time-to-market cost, enables rapid prototyping of complex algorithms and simplifies debugging and verification. Therefore, FPGAs are an ideal choice for implementation of real time image processing algorithms.

First chapter is dedicated to discuss about the theory behind the frequency domain filtering and about frequency domain mean filter which is to be implemented. The FPGA design for the frequency domain mean filter is explained block by block in 2nd chapter. 3rd chapter concludes the design and tells about the future scope work.

Chapter 1

Theoretical Background

Some of the noise types such as periodic noise which is periodic stripe like structure on the image can’t be fully removable in spatial domain. The nature of this type of noise is easily recognizable in frequency domain. That is when the domain is transformed from spatial to frequency domain by taking FFT over the image.

1.1 Frequency Domain Filtering

The Fourier Transform is of fundamental importance to image processing. It allows us to perform tasks which would be impossible to perform any other way; a powerful alternative to linear spatial filtering; it is more efficient to use the Fourier transform than a spatial filter for a large filter. The Fourier Transform also allows us to isolate and process particular image frequencies and so perform low-pass and high-pass filtering with a great degree of precision. Some noise such as periodic noise can’t be removed in spatial domain, can be removed easily in frequency domain.

1.1.1 The one-dimensional discrete Fourier Transform

Fourier transform of a discrete function of on variable, f(x), x=0,1,2,…..,M-1,is given by the equation

$$F(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)e^{-\frac{j2\pi ux}{M}} \qquad \textit{for u=0,1,2,.....M-1}$$

Similarly original function can be obtain from

$$f(x) = \sum_{x=0}^{M-1} F(u)e^{\frac{j2\pi ux}{M}} \qquad \textit{for x=0,1,2,.....M-1}$$

The Fast Fourier Transform. Without an efficient method of calculating DFT, it would be only of academic interest, and of no practical use. However, there are a number of extremely fast and efficient algorithms for computing a DFT; such an algorithm is called a fast Fourier transform, or FFT. The use of an FFT vastly reduces the time needed to compute a DFT in order to apply Fourier transform to an image we have to search for expression for two dimensional DFT.

1.1.2 Two Dimensional DFT and its inverse

In two dimensions, the DFT takes a matrix as input, and returns another matrix, of the same size, as output. If the original matrix values are f(x,y) where x and y are the indices, then the output matrix values are F(u ,v) .

The definition of the two-dimensional discrete Fourier transform is very similar to that for one dimension .The forward and inverse transforms for an MxN image. Where for notational convenience we assume that the y indices are from 0 to M-1 and x indices are from 0 to N-1 are

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{[-2\pi j\left(\frac{xu}{M}+\frac{yv}{N}\right)]}$$

$$f(x,y)\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(u,v)\, e^{[2\pi j\left(\frac{xu}{M}+\frac{yv}{N}\right)]}$$

These equations look complex for calculation but if we use property of separability then the equations can be written in the following manner.

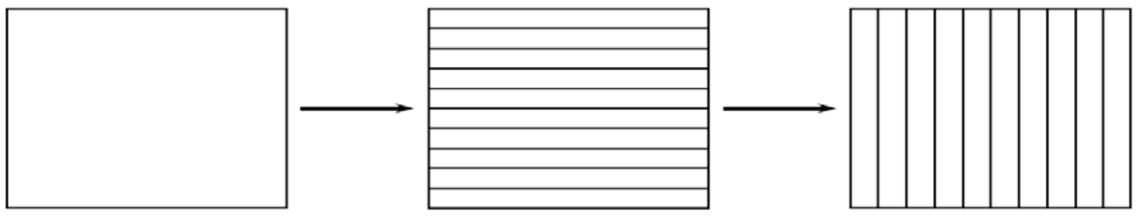
$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{[-2\pi j\left(\frac{xu}{M}+\frac{yv}{N}\right)]}$$

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-2\pi j\frac{xu}{M}} e^{-2\pi j\frac{yv}{N}}$$

$$F(u,v) = \sum_{y=0}^{N-1}\{\sum_{x=0}^{M-1} f(x,y)\, e^{-2\pi j\frac{xu}{M}}\}e^{-2\pi j\frac{yv}{N}}$$

Here we can see that the term in parenthesis is of single variable only. So for a matrix, two dimensional DFT can be obtained by first performing one dimensional DFT with respect to one variable then performing one dimensional DFT with respect to other variable on that previously obtained value.

So, it can be seen that Two dimensional DFT can be performed by first computing one dimensional DFT row wise for each row then performing column wise DFT for each column.



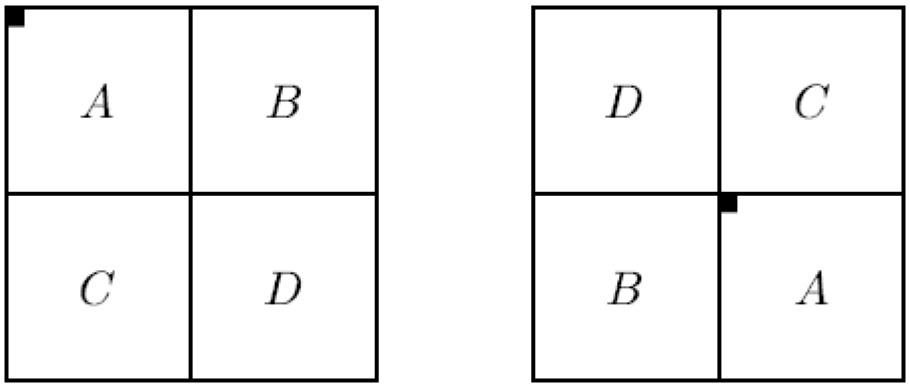
The DC coefficient

The F(0,0) of DFT matrix is called the DC coefficient This term is equal to the sum of all terms of the original matrix. If we set u and v equal to 0 equation reduces to

$$F(0,0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)$$

Shifting Operation

For purpose of display, it is convenient to have the DC coefficient at the centre. This will be happened if all elements f(x,y) in the matrix are multiplied by $(-1)^{x+y}$ before the transform. In the diagram the DC coefficient is the top left hand element of sub matrix and is shown as a black square.



An FFT After shifting

1.1.3 Basic of filtering in Frequency domain

Basic filtering steps are as followed.

1. Perform 2-D FFT on the image matrix. (by performing row wise FFT for each row and then perform column wise FFT for each column on the result matrix obtained by row wise FFT)

2. Perform shifting operation (FFT shift) to center the DC coefficient.

3. Multiply F(u ,v) by filter function means to apply filtering algorithms in this step .

4. Perform inverse shifting operation.

5. Perform the IFFT (by procedure same as for FFT) to get original matrix in spatial domain.

1.2 Periodic Noise removal

Periodic noise is a repetitive signal which is added to the main signal. This periodic noise in a digital image is repetitive spatial pattern which effectively degrades the image quality. There are some different sources for creation of periodic noises in a digital image. Electrical or electromechanical inferences in imaging systems, electrical inference in image receiver systems, and unequal sensitivity of detectors are the main sources. Periodic noise is mainly of three types -1.fully periodic in nature 2.quasi periodic 3.stripe type .Periodic noise cannot be simply removed by spatial filters as its periodicity occurring or its variation is not predictable. Periodic noises are usually modeled by summing several sinusoidal functions with different amplitudes and frequencies; therefore, in the frequency domain the noisy image appears like stars with high amplitude. So in frequency domain it can easier to remove periodic noise by selecting those stars like regions by some band selective filters.

The removal of periodic noise can be done by the basic frequency domain filters such as Band reject filter and Notch reject filters. Designers further searched for efficient filters which removes periodic noise but retains the image. Some of them are

1. Frequency Domain Mean Filter
2. Frequency Domain Median Filter
3. Optimum Notch Filter
4. Adaptive Notch filter

Out of this filters Frequency domain mean filter is the basic filter which is similar to mean filter in spatial domain. FPGA design for frequency domain mean filter which is described in chapter 2 can be adopted to design other filters too. Only thing is to change the filtration block. The theoretical background for frequency domain mean filter is described below.

1.2.1 Frequency-Domain Masked Mean Filter

The basic idea of this type of periodic noise reduction filters is similar to the frequency-domain median filter. The frequency-domain masked mean filter uses the masked mean values instead of the median ones [20]. The masked mean value is defined over an *N*×*N* masked window. All values in the *N*×*N* masked window are '1', except the center which is considered '0'. It means that the center of the *N*×*N* local window is omitted in mean value computations. Suppose that *S*(*u, v*) is the masked mean value of the pixels of the window, *X*(*u* ,*v*) is the pixel on which the window is centered .So the reconstructed value is given by the following equation .

$$Y(u,v)=\begin{cases} X(u,v)/\delta & \text{if} \quad \dfrac{S(u,v)}{X(u,v)}>\theta \\ & \text{and} \quad (u,v)\neq(0,0) \\ & \\ X(u,v) & \qquad \text{otherwise} \end{cases}$$

Where δ is selected based on the periodic noise reduction power of this filter. Θ is the predefined threshold to detect noise magnitudes .Like the previous filter, the frequency of (0,0) should be unchanged [20].

Chapter 2

FPGA implementation of Frequency domain mean filter

Frequency domain filtering for noise removal involves some steps that are not present in case of spatial filtering as frequency domain filtering works in frequency domain. Steps are already discussed in the previous chapter when discussing about frequency domain filtering .Different steps of frequency domain mean filtering are illustrated below.

1. FFT operation on the noisy image.
2. Shifting operation to shift the dc value at the centre.
3. Absolute operation to perform the filtering operation.
4. Mean filtering operation.
5. Inverse shifting operation.
6. IFFT operation to regain the image after noise removal.

Approach to design a filter in hardware may be different from its counterpart in software due to design complexity and hardware requirements .Design with less hardware & with high throughput is always a challenge in hardware implementation of any algorithms .Objective is also to take care of the time taken for filtering operation.

The main architecture design frequency domain averaging filter is given below in fig 1.

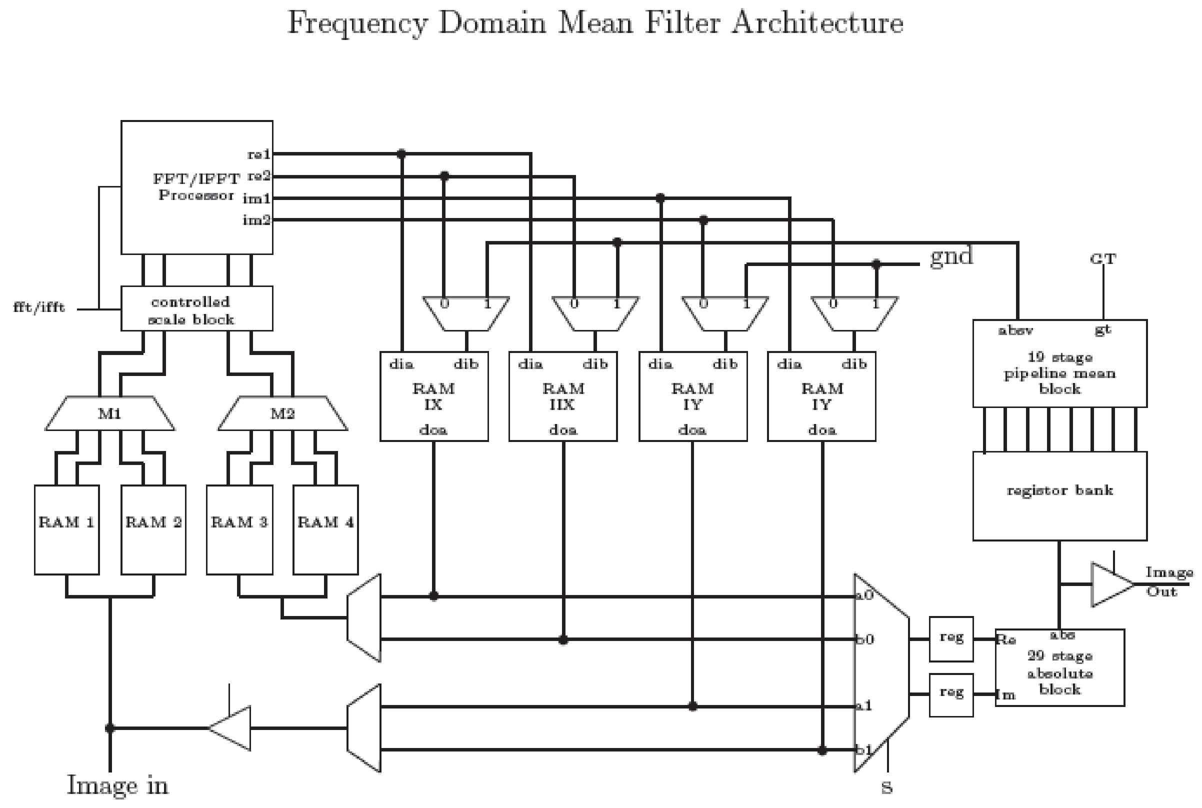


Fig-1

At the first step to perform FFT operation on an NXN image ,an N point FFT/IFFT processor is considered .Image pixels are first written into RAM1then in the next phase pixels are written into RAM2 then again pixels are written into RAM1.This switching of RAMs continues until all the pixels are accessed .All the RAMs are implemented by dual port memory element which have simultaneous read and write facility .While in the phase 2 pixels are being written into RAM2 ,pixels are being read from RAM2, again in phase 1 reverse process is going on .This process continues .A 4:2 MUX is suitably controlled to select between two RAMs .Same type of arrangement is there for imaginary values when IFFT operation takes place. A controlled 1/N multiplication scale block is there before FFT/IFFT processor with FFT/IFFT control signal .When FFT operation takes place multiplication by 1/N is not taken place.

The N point FFT/IFFT processor should be of throughput 4 (two real & two imaginary values) and there must be a provision of giving a pulse after the FFT/IFFT operation latency period i.e. whenever the first output comes out of the processor it gives a pulse output to start the rest of the process .It indicates that FFT/IFFT operation is complete and data are available at output bus.

The data bus here is considered to be of 16 bit unsigned integer for representing fixed point representation format .11 bits from MSB side are representing the integer value and the remaining 5 bits are representing the fractional part .The choice of the data frame length depends on the image size and how much precision is required .The dc value which is the average value of an image differs image to image .It is required to retain that dc value after filtering operation .So data frame length depends upon the dc value. Higher the dc value, higher will be the no bits required ,so the hardware complexity will be more .Double type representation significantly reduces the pixel values ,so reduces the dc value that reduces no of bits required .16 bits representation proves well up to 64x64 image size signifying the disadvantage of fixed point representation .

Real values (re1 & re2) are written into RAM IX during phase1 in alternate addresses in a column wise fashion through port a & b .In phase 2 real values starts written into RAM IIX in a same manner .So half of the real values are stored in RAM IX and other half is stored in RAM IIX .Complex values (im1 & im2) are written into RAM IY & RAM IIY in a similar fashion during phase 1 & 2 .All memory elements RAM IX ,RAM IIX ,RAM IY are RAM IIY half frame buffers to store half the values after FFT.

2.1 FFT/IFFT processor

Converting an image into frequency domain requires a FFT operation .Direct implementation of 2-D FFT processor [14] is a slow process and also hardware requirement is more. 2-D FFT processor can be designed by means of available 1-D FFT processors[15,16,17] .For that image is processed row wise through the 1-D FFT processor and then results are stored and transposed by RAM array transposer .Output of the transposer then again processed through the 1-D FFT processor to get the 2-D FFT result .So in place of two 1-D FFT processor one 1-D processor can do the job but only requirement is that to rearrange the results in column wise .Several parallel 1-D FFT processors[15,16,17] of low throughput to high throughput are available. Throughput is an important parameter to design FFT processors .Higher will be the radix of operation lower will be the time for processing and throughput will be higher but hardware requirement and complexity will also be higher. A radix-2 parallel 1-D FFT processor is considered for avoiding complexities.

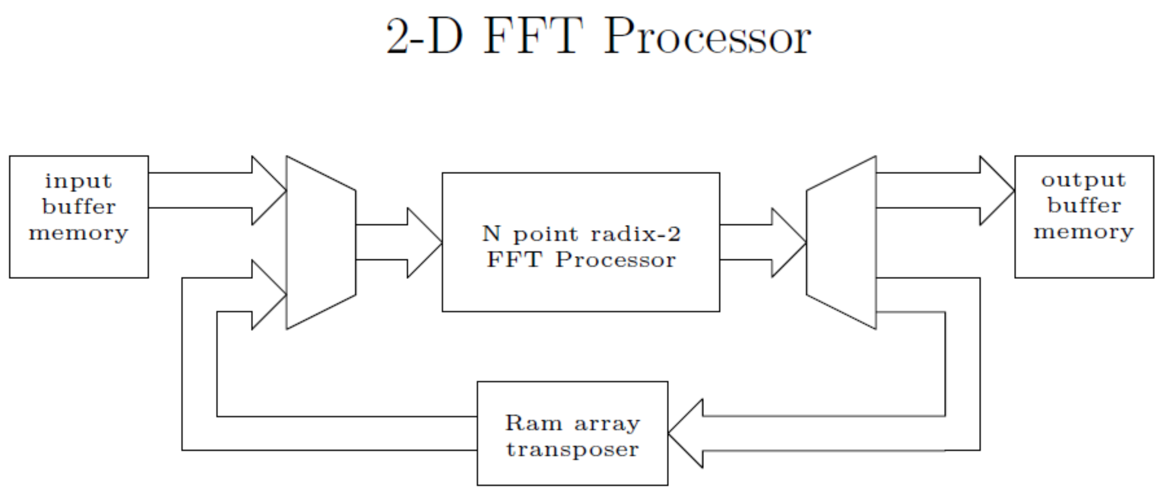


Fig -2

At the last step an IFFT operation is required to regain the filtered image .An IFFT and FFT operation both can achieved by a single processor having provision to change the twiddle factors as per FFT/IFFT operation is required and having a provision for extra multiplication factor 1/N in case of IFFT operation .The extra division by N can be achieved by controlled 1/N scaling block described below in figure.

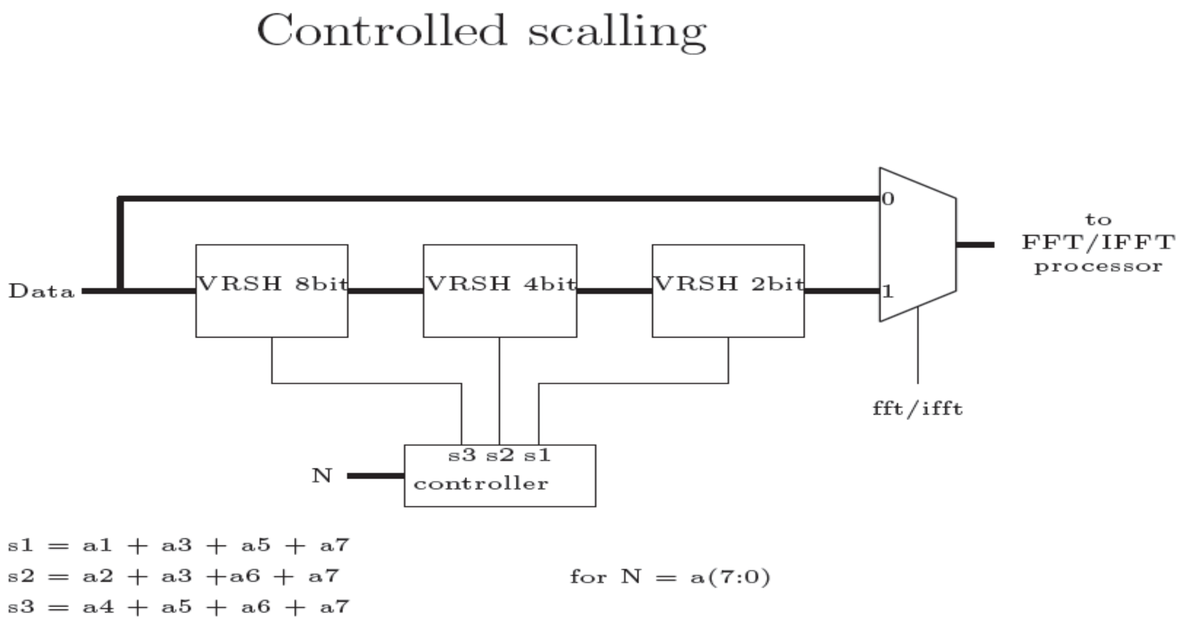


Fig -3

Here in this block all the VRSH blocks are controlled hard left shifters with a control input s. If s is equal to 1 then right shift take place otherwise passes as it is .For these case hardware minimization can be achieved as both the operation are not taking place at the same time .Several proposed designs [18] are available for implementing FFT /IFFT processors.

2.2 Shifting Operation

Next step is to perform FFT shift to bring the dc value at the centre .For that consider the whole image after FFT operation is divided virtually in 4 parts A, B, C & D .A virtual vertical line divides the image in two parts LHS & RHS .After FFT shift the location of the blocks are diagonally interchanged .To do the shifting operation means to flip the image once vertically and then flip it horizontally.

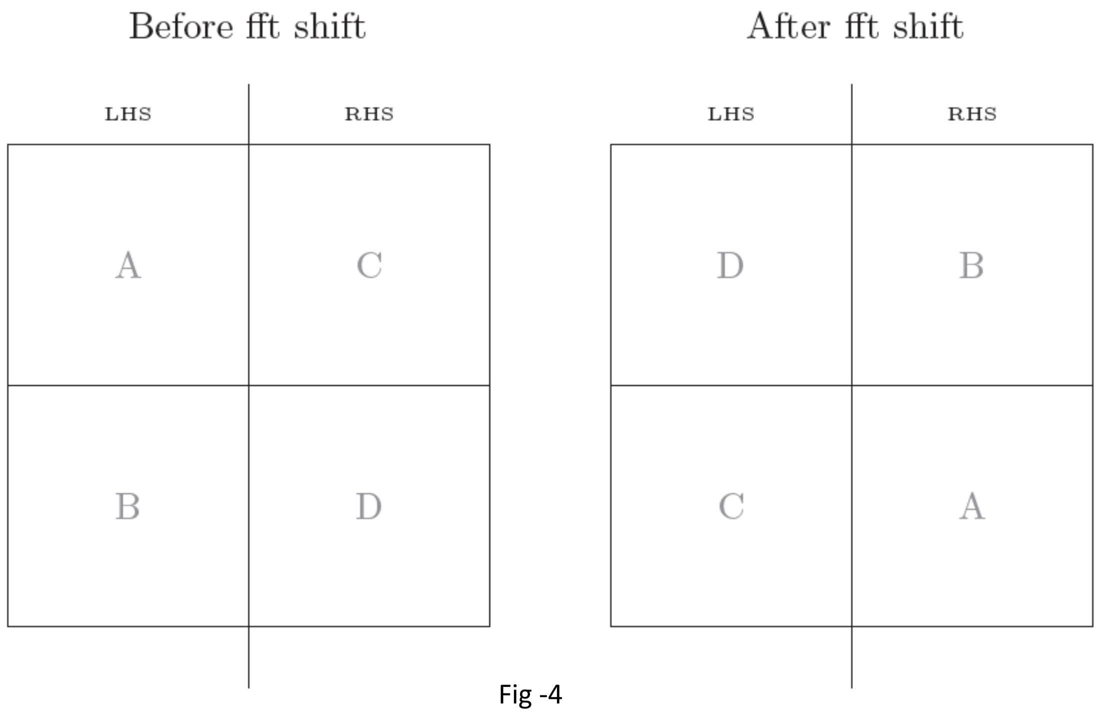


Fig -4

All the data are written into the RAMs in column wise fashion .Consider during phase 1 LHS of the image data is stored in RAM IX & IY and during phase 2 RHS side is written into RAM IIX & IIY .All the RAMs has address locations 0 to (N/2-1) .Now to perform a vertical flip, data are read from RAM IX & IY during phase 5 through port A while RAM IIX & IIY are busy in writing .Data are read in a fashion to perform horizontal flip simultaneously i.e. first read from block B then from block A again read from block B and then from block A .This process continues until all the pixels are read .Reading address is generated by a FFT shift counter . Once all the values are read from RAM IX & IY ,during phase 6 reading from RAM IIX & IIY is take place in a similar fashion as in case of RAM IX & IY .A 4 : 2 MUX selects real & imaginary values from RAM IX & IY first then selects from RAM IIX & IIY and fed them to a absolute block to perform absolute operation .

Data which are written in RAM X & Y are not changed to perform FFT shift but read in a manner to perform the same .Here if all the pixels are written into a single RAM then shifting operation can’t be done until all the values are written in the RAM .So by writing the pixels in two similar RAM there is a time saving of 6N clk periods.

2.3 Address & Phase generator

Address & phase generator provides address location to which data is to be written into or from which data is to be read from the memory location and also generates phase signals during which read or writing operation is to be carried out .For dual port memory there are two address bus for port A & port B .The overall address & phase generator consists of 4 main counter blocks namely Even & Odd counter ,FFT shift address counter ,Restoration counter and an up counter .The block diagram for address & phase generator is given below :

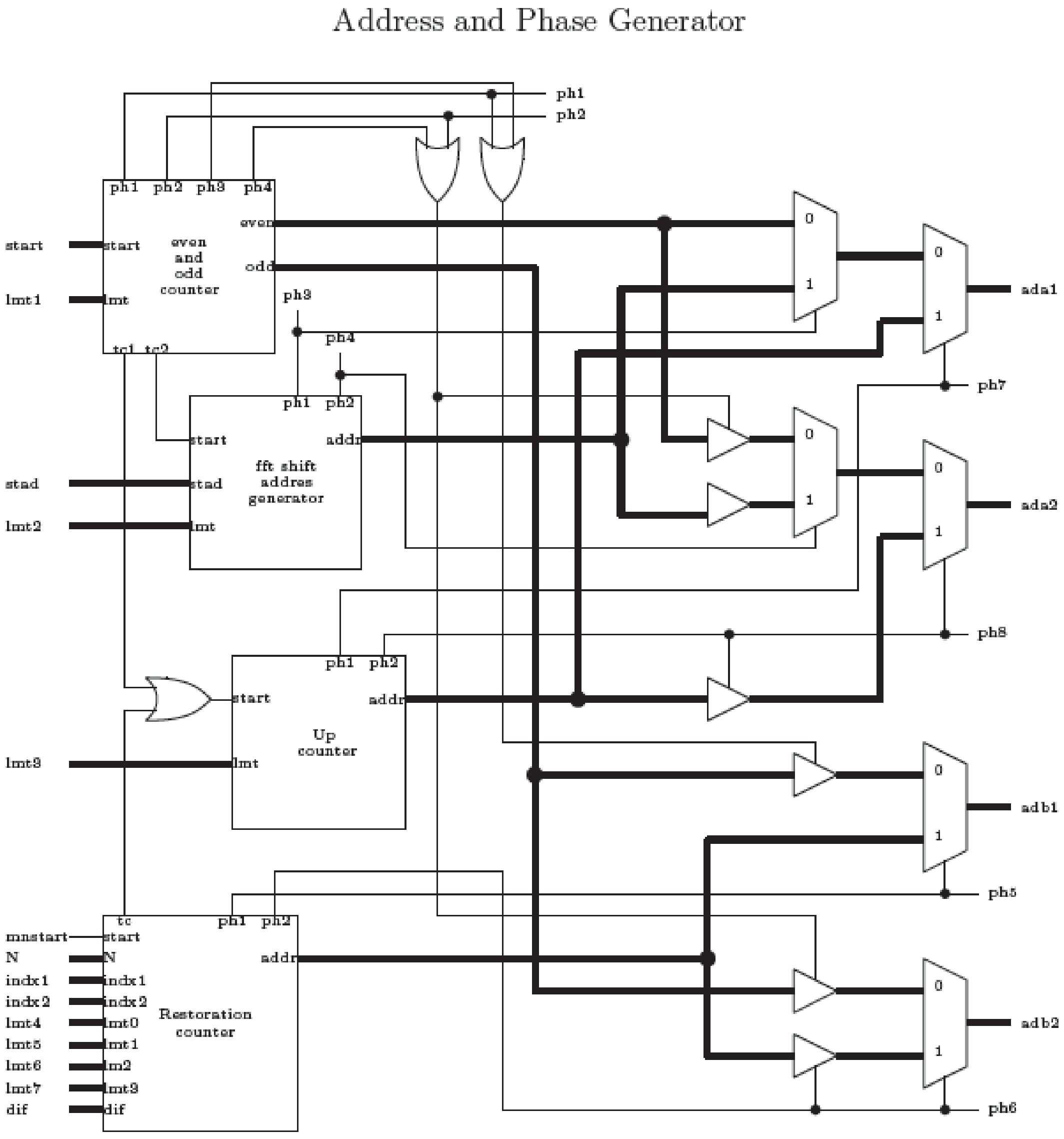


Fig -5

There are four address buses namely ada1 and adb1 is for RAM IX & IY whereas ada2 and adb2 is dedicated to RAM IIX and IIY.

All the parameters which controls the address & phase generator block can be obtained from size (N) of the squared image.

For 3x3 window operation	***For 5x5 window operation***
$starting\ address = \frac{N}{2}(N-1) = stad$	$starting\ address = \frac{N}{2}(N-1) = stad$
$indx1 = \frac{N}{2}(N-1) - (N-1) = stad - (N-1)$	$indx1 = \frac{N}{2}(N-1) - (N-1) = stad - (N-1)$
$indx2 = indx1 + N = stad + 1$	$indx2 = indx1 + N = stad + 1$
$indx2 - indx1 = dif = \frac{N}{2} + 1$	$indx2 - indx1 = dif = \frac{N}{2} + 1$
$lmt3 = \frac{N}{2} - 3, lmt4 = lmt3 + N$	$lmt3 = \frac{N}{2} - 3, lmt4 = lmt3 + N$
$w = \frac{N}{2} - 1, lmt5 = lmt6 - w, lmt5 = N - 1$	$w = \frac{N}{2} - 1, lmt5 = lmt6 - w, lmt5 = N - 1$

Table 1

2.3.1 Phase generator block

It's a logic block which generates phase (ph) when giving a pulse at start input of any duration .It provides enable signal to counters which generates addresses .It can be stopped by giving an another pulse at stop input .

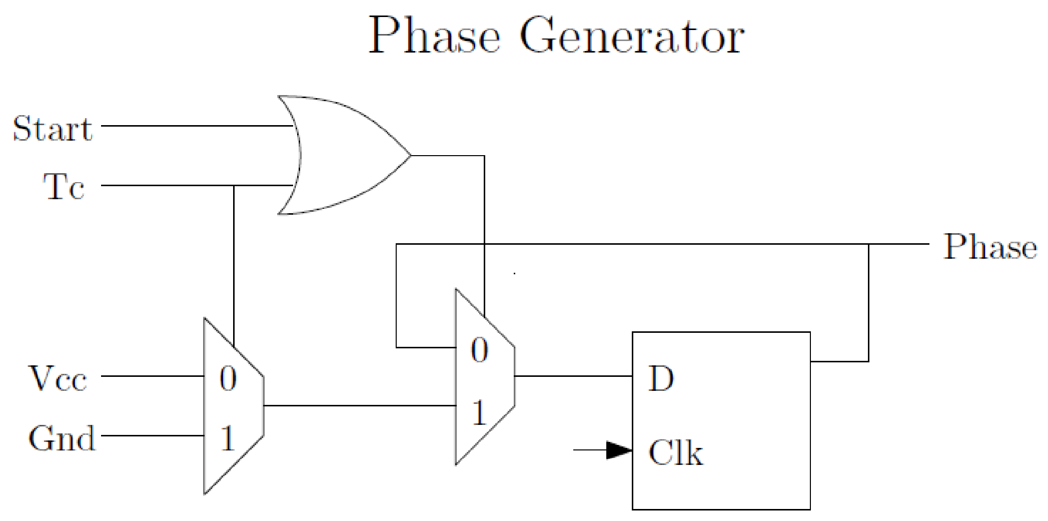


Fig -6

2.3.2 Even & Odd address generation

As FFT processor with throughput 4 is considered with two real and two imaginary values ,it is required to store the values at the same time .So real values are simultaneously written into consecutive even and odd address location .The block diagram for even & odd counter is shown below :

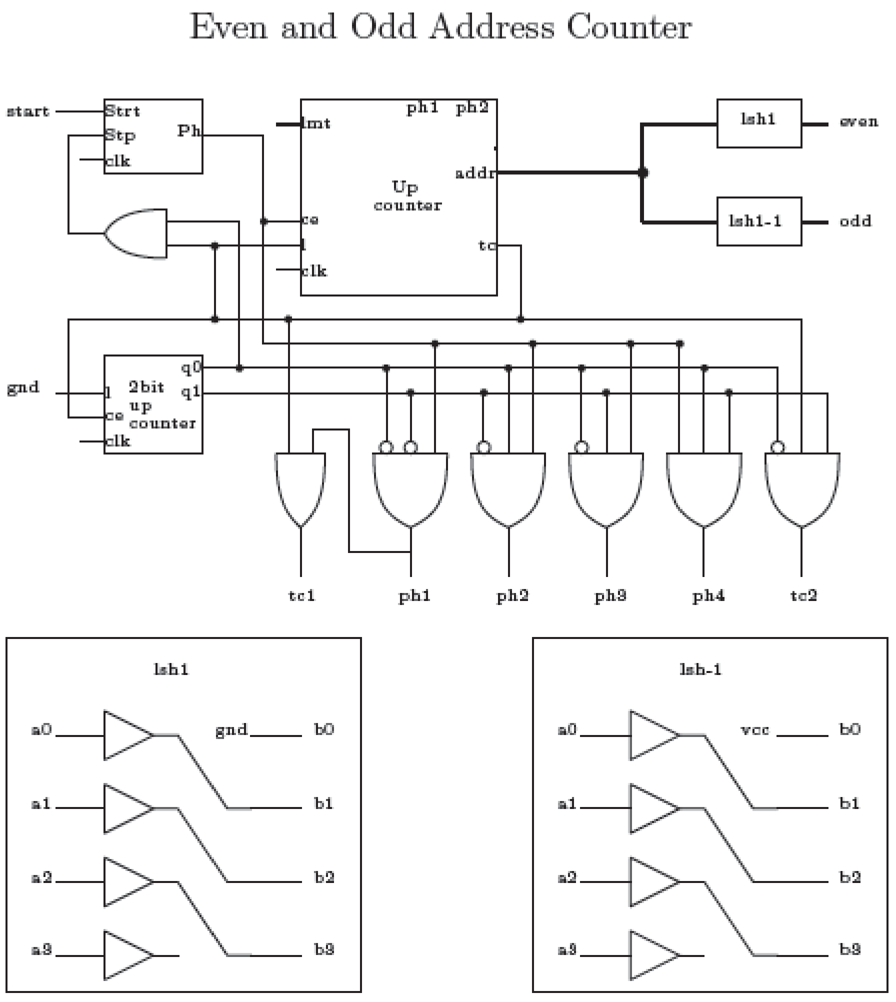


Fig -7

Start signal starts phase generator and provides enable to the up counter .Address bus passes through lsh1 which is 1 bit hard left shifter generates even address and same address bus when passes through lsh-1which is also a 1bit hard left shifter with LSB at VCC, generates odd address .The up counter equality comparator generates terminal count (tc) signal when count is equal to lmt .This tc signal again loaded the up counter .A 2 bit up counter is there to generate phase signals (ph1 ,ph2, ph3, ph4) , generates start pulse for FFT shift counter (tc1) & start pulse for up counter (tc2) .Signal tc increments the 2 bit up counter .After FFT process when start pulse come it generates tc2 ,ph1 ,ph2 and when again after IFFT process start pulse come then it generates tc1,ph3,ph4 .

2.3.3 FFT shift counter

After getting the start pulse (tc1) from even & odd counter it starts .Phase generator provides enable signal for all the counters .Start signal loads the up counter1 & down counter with initial address stad and (stad – 1) respectively .Both starts counting after getting enabled .Up counter 2 generates terminal count signal whenever it counts (N/2 – 1) and loads itself and up counter 2 .Up counter 2 gets initial address stad which is loaded by start signal and from down counter which is loaded by terminal count signal of up counter 2 .Up counter 1 and down counter must be of same size (0 to (NxN/2 – 1)) .A toggle flip flop is provided to generate phase signals which is toggled by terminal count signal of up counter 1.Up counter 1 generates terminal count signal when it counts equal to lmt.

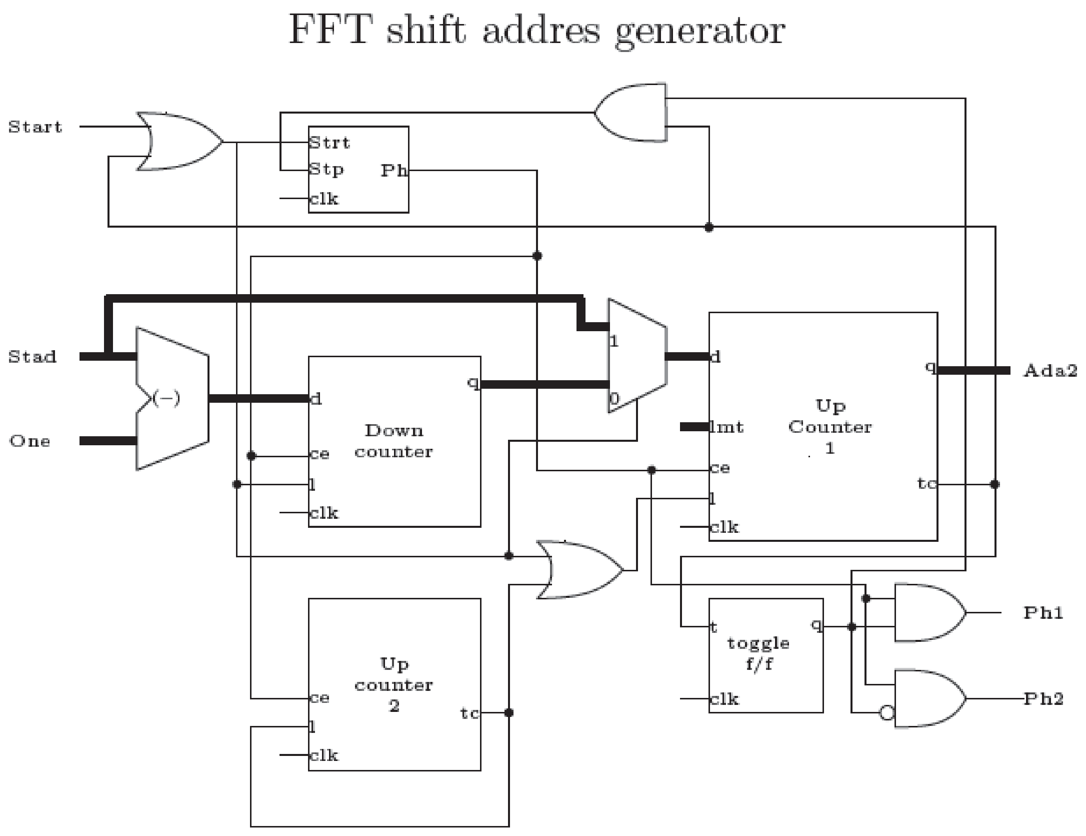


Fig -8

2.3.4 Restoration counter

Restoration counter generates the address locations where corrected pixel values is to be replaced .After processing by mean filter block the corrected pixel values are feedback to the RAM X & Y to replace the noisy pixels in order to avoid the requirement of another inverse shifting process .It generates those addresses only where replacement is to be done in each block of pixels.

The mnstart pulse is available two clock cycle before when the correct absv value to be replaced is available. At first step mnstart pulse starts the 1 bit up counter and also provides indx1 at input data bus of up counter1 but do not loads it .Whenever 1 bit up counter generates terminal count signal ,it loads the up counter1 with indx1 & up counter2 with din and both the up counter starts counting from it .Whenever up counter2 counts up to lmt2 ,it generates tc1 and generates tc2 when it count up to lmt3 .Up counter2 counts up to (0 to N/2 - 1).Signal tc2 loads it again .The value of din is 2 for 3x3 window & 4 for 5x5 window .

Restoration process is started from block B then in block A and then again in block B .This process continues for all the current pixels to be checked .So input data bus of up counter1 selects through a 2:1 mask .For block B it starts from indx1 and in block A it starts from (indx1 – dif) .Whenever tc2 signal comes it stops the up counter1 and starts the 1 bit counter .And then again terminal count of 1 bit counter starts up counter1 & up counter2.And process continues for all the addresses in block B and block A. Every time the up counter1 starts it covers one column from block B to A then there is a gap of two clk cycles due two 1 bit counter .For 5x5 window there will be a 2 bit counter in place of 1 bit counter only because to avoid faulty values (absv) operated on faulty current pixel in mean filter block .When up counter1 counts up to lmt0 it generates a terminal count signal which indicates that restoration in block B & A is over. The variable w represents restoration width of a block.

Restoration process is done only when the GT signal from mean filter block and write enable signal of port b of RAM X & Y are both high .When restoration process in RAM IX & IY is done up counter2 is loaded with indx2 and follows similar process .When up counter1 counts up to lmt1 it indicates end of restoration and stops the overall restoration counter .Again toggle f/f is provided to generate phase signal.

2.3.5 Up counter

As the restoration is done in each block by restoration counter block, so there is no need of reverse shifting separately .An simple up counter is used for reading the values .The terminal count of restoration counter starts the up counter. The reading process from RAM IX & IY is taken place through port A during phase 9.The real & imaginary values are then available for FFT/ IFFT processor for IFFT operation. If reading from RAM IX & IY is done then reading from RAM IIX & IIY is started during phase 10.

Restoration Counter

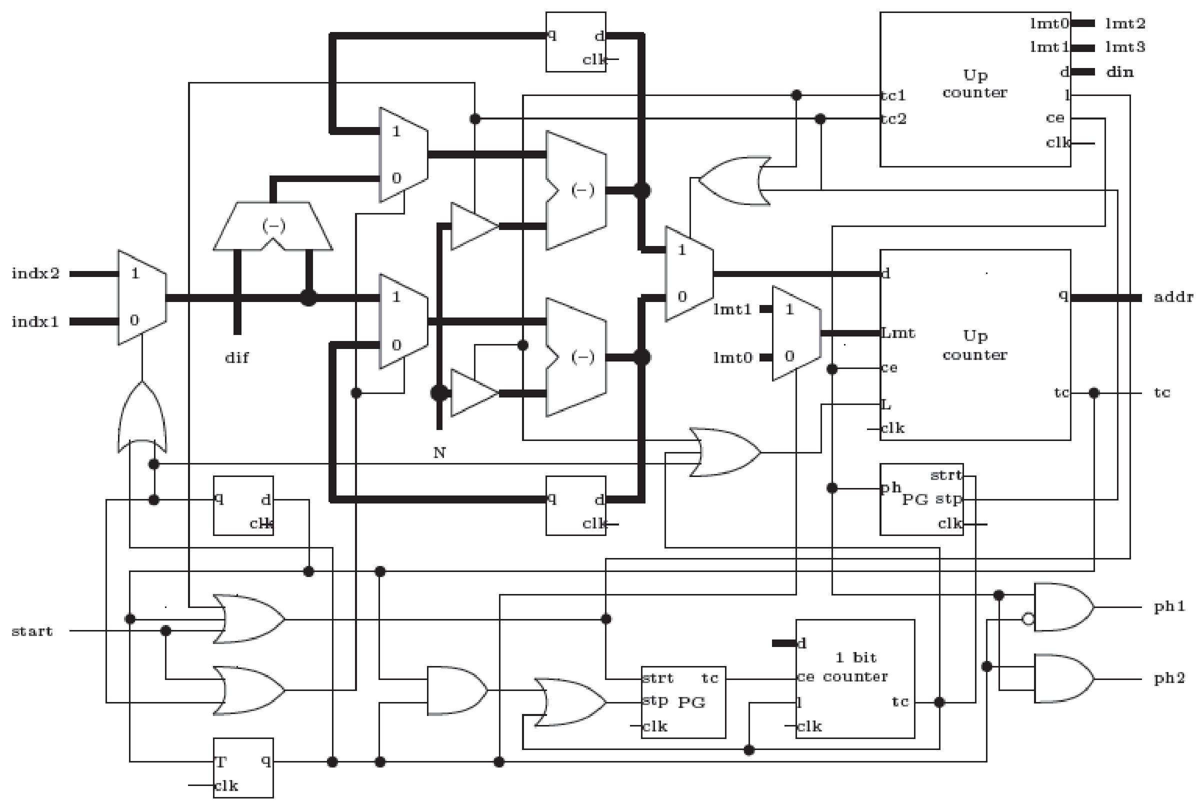


Fig -9

2.3.6 Phase Generation

The address and phase generator block generates address locations where from data is to be read and where to be the data is to be write. It also generates phase during which data operation is to be takes place. The phase signals are important control parameter which controls RAM switching operation and also masking operation. The phase diagram is given below in figure 10.

When after FFT operation start pulse arises, the even & odd counter starts and writing into RAM IX & IY is done during phase 1.Even & odd counter is active during phase 1 & phase 2.During phase writing into RAM IIX & IIY is dine. Then after phase 1 FFT shift counter is started and reading operation from RAM IX & IY is started. The FFT shift counter is active for phase 5 & phase 6. During phase 6 reading from RAM IIX & IIY is done. Then when after absolute operation and register bank waiting time a pulse mnstart is come then the restoration counter started which generates phase 7 & 8. During phase 7 restorations is done in RAM IX & IY and as soon as restoration is done in RAM IX & IY, up counter is started to start reading from RAM IX & IY during phase 9. During phase 8 & phase 10 similar processes is going on for RAM IIX & IIY .Even & odd counter generates other two phase signals phase 3 & 4 after IFFT operation takes place when again start signal comes.

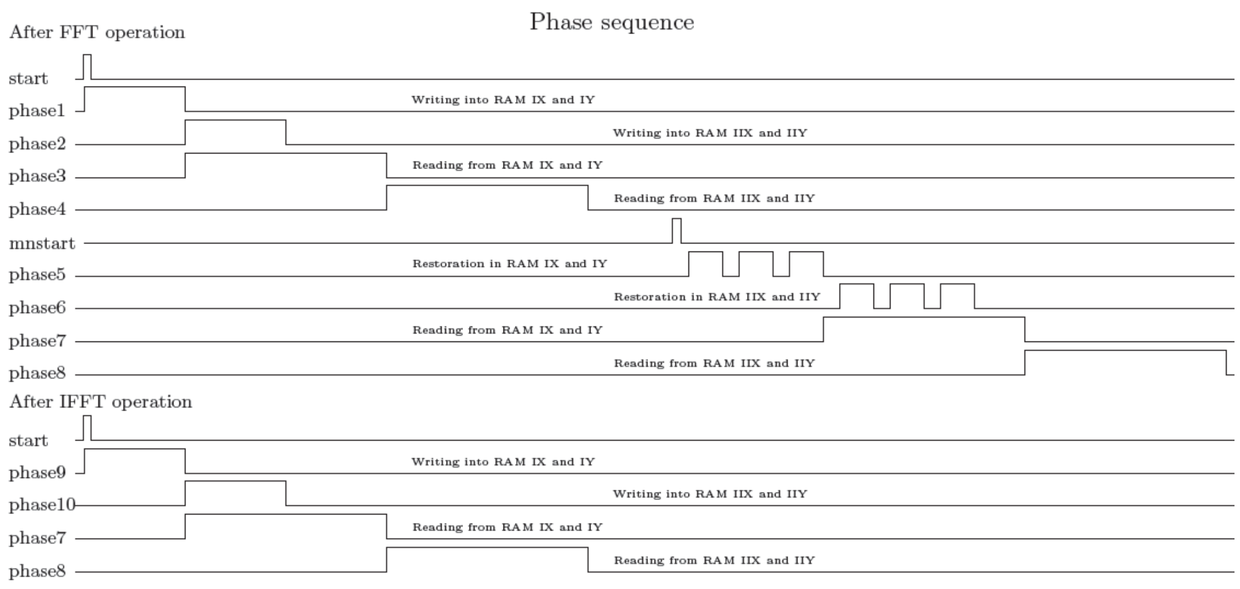


Fig -10

2.4 Absolute Block

Absolute operation is necessary to for mean filtering operations .Taking real & imaginary values from memory absolute operation is taken place .Absolute operation can be done in many way .CORDIC algorithm [] provides absolute value in vectoring mode with no of iterations .But hardware requirement is much more when pipelined CORDIC is used .The basic building block for finding absolute is used here which is shown below.

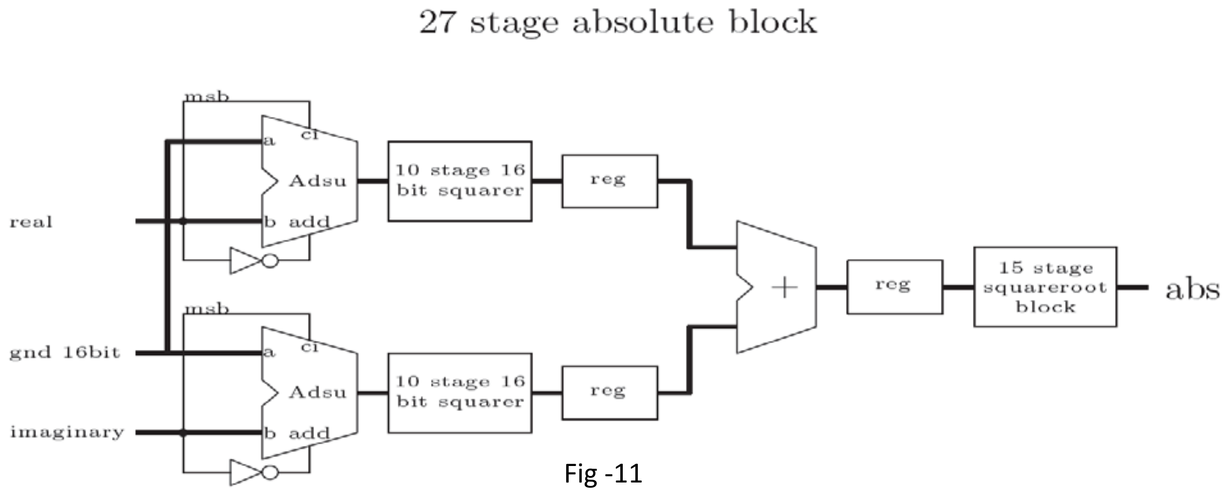


Fig -11

After FFT operation data can be negative or positive but for absolute operation to be carried out always positive numbers must be considered otherwise it will give wrong result .So at the initial stage there are two ADSUB blocks are placed which are controlled add/sub block .If data is negative, it converts the data into an equivalent positive number by simply taking two's compliment of the data .MSB of the input data detects if it is a negative data or not . The dc value should passed as it is though it's MSB is 1.

Two 10 stages 16 bit squarer for both real and imaginary are there to calculate square value. The pipelined implementation VEDIC squarer [1] minimizes hardware requirement, as its involves only dedicated square operation.

After square operation both the outputs are added by a 32 bit adder and then it is processed by the 32 bit square root block having 15 stage to find .There are so many algorithms [9] to find square root of a number but when the algorithms are to implemented in hardware few satisfy the minimum requirement of hardware and of less complexity. Some of the algorithms are namely direct methods[2,3,6], algorithms based on Newton Rampson formula [4,5],normalization techniques and approximation from real functions .Direct methods includes restoring and non-restoring algorithm .These are also can subdivided into non-restoring and restoring type. Direct methods are ideal to implement in hardware where as Newton Rampson formula based algorithms are more used in software due to more complexity to implement. CORDIC algorithm developed by Volder [7] extended by DeLugish [8] to calculate square root of a number. The FPGA implementation of non-restoring square root algorithm is followed here which Yamin Li et al. describes in [10],[11]. A 32 bit square root block with total 15 pipelined stages is designed.

2.5 Register Bank

A register bank is used to realize the 3x3 or 5x5 window .For 3x3 window operations the block diagram of the register bank is shown below.

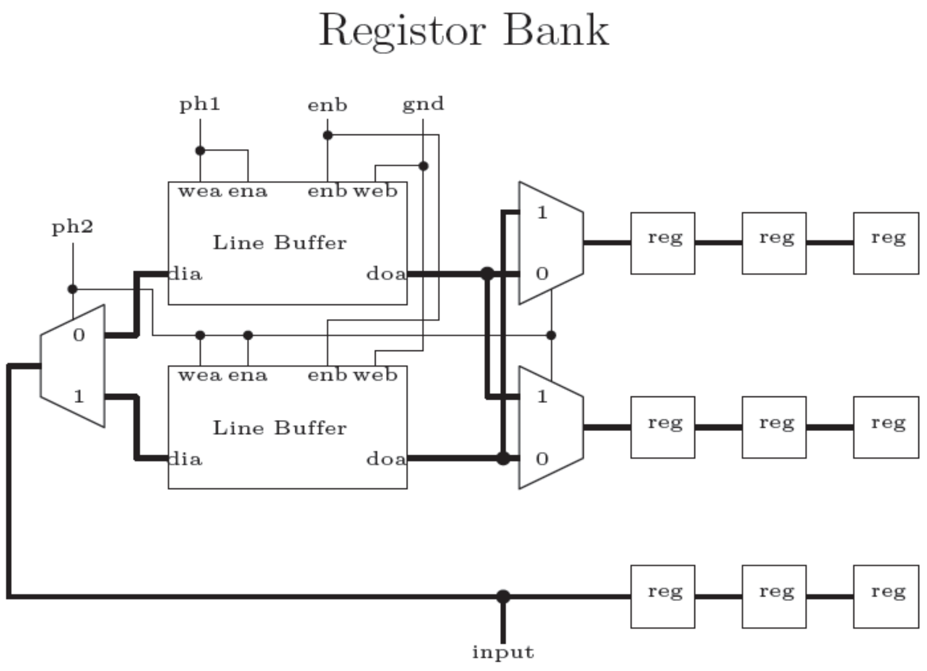


Fig -12

Two line buffer which is realized by dual port RAM having address location 0 to (N-1) and 9 registers are used as to form 3x3 window. In [22] Pei-Yin Chen first uses this kind of structure to realize window operation. Higher order window can also be realized just increasing the no of resistors and line buffers. Input data is given to a 16 bit DEMUX as well as to a register .During phase1 data is written into line buffer 1throug port A while during phase 2 data is written into line buffer 2. Phase 1 and Phase 2 are two non-overlapping phases which odd columns in linebuffer1 and even columns in line buffer2. Data is read simultaneously from both the buffers through port.

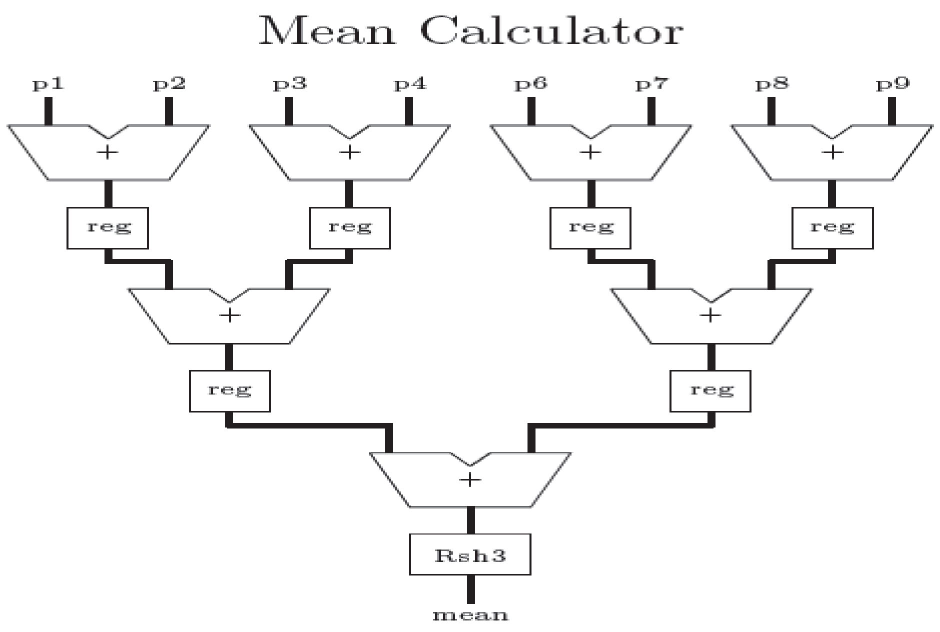
2.6 Pipeline mean block

The pipeline mean block takes input from register bank and process them .The 8 data values except the current pixel are input two the two stage mean block .Mean block calculates sum of all the values and calculates mean. A 3 bit hard right shifter divides sum by 8 to calculate mean which avoids the requirement of a divider. The mean block is shown in fig

Current pixel (cp) is divided by mean value by a 15 stage divider and the result is compared with a threshold value by comp1.Comp1 generates GT signal if the cp is less than the result after division .If GT signal is high then replace the corrupt pixels by current pixel divided by 50 otherwise retain the value. So to implement the logic here all the current pixels except dc value are divided by 50 but write those values of absv whenever GT is signal is high .So in one path cp is input to the MUS through only resistors and in another path cp is input the MUX through a scale block which divides the current pixel by 50 .Comp 2 detects which current pixel value is dc value and gives LT signal which passes cp without division. Dc value detection is done by comparing sum for 3x3 windows with the current pixel. For higher frequency periodic noise cp is multiplied by 2 and then compared with sum to detect dc value

Scale block

Division by 50 is realized by a scale block. Division by 50 is same as multiplication by .02 which can be represented very nearly equal to as $((2^{-6} + 2^{-8}) + 2^{-11})$. So input data is right shifted by 6 bit, 8 bit and by 11 bit and added .Result of after addition is input data divided by 50. Right shifting operation is achieved by hard right shifters.


Mean Calculator
p1
p2
p3
p4
p6
p7
p8
p9
+
reg
Rsh3
mean

.

Fig -13

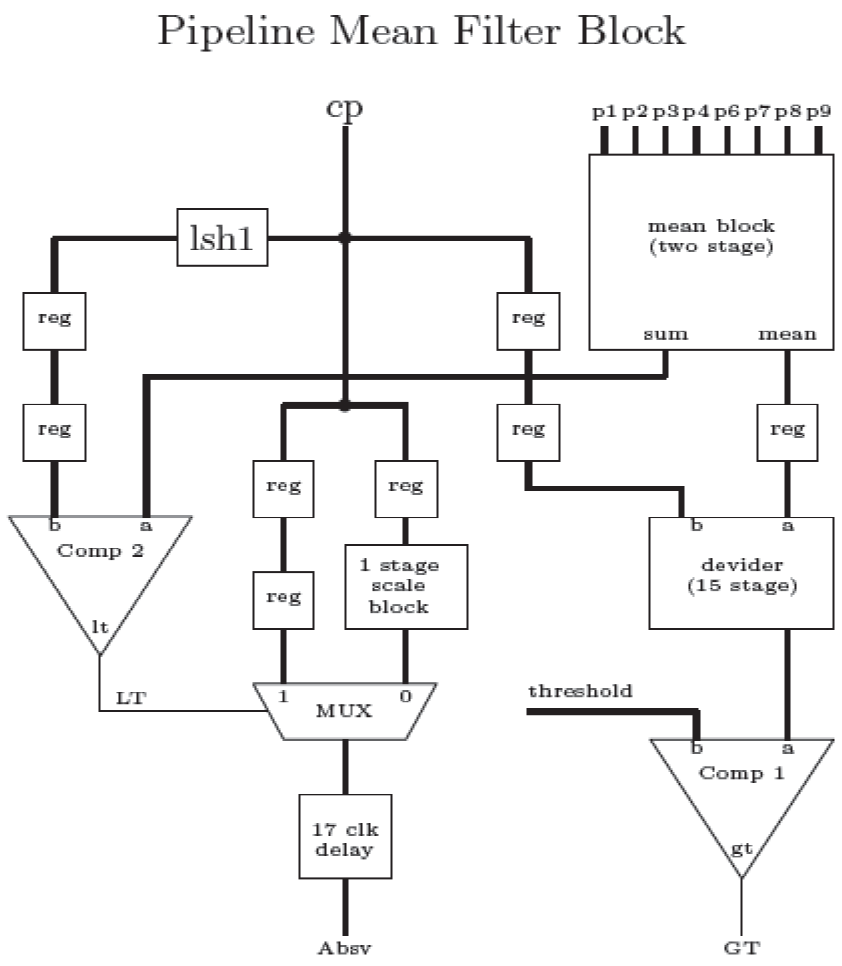

Pipeline Mean Filter Block
cp
p1 p2 p3 p4 p6 p7 p8 p9
lsh1
mean block (two stage)
sum
mean
reg
b
a
Comp 2
lt
LT
1 stage scale block
1
0
MUX
17 clk delay
Absv
devider (15 stage)
threshold
Comp 1
gt
GT

Fig -14

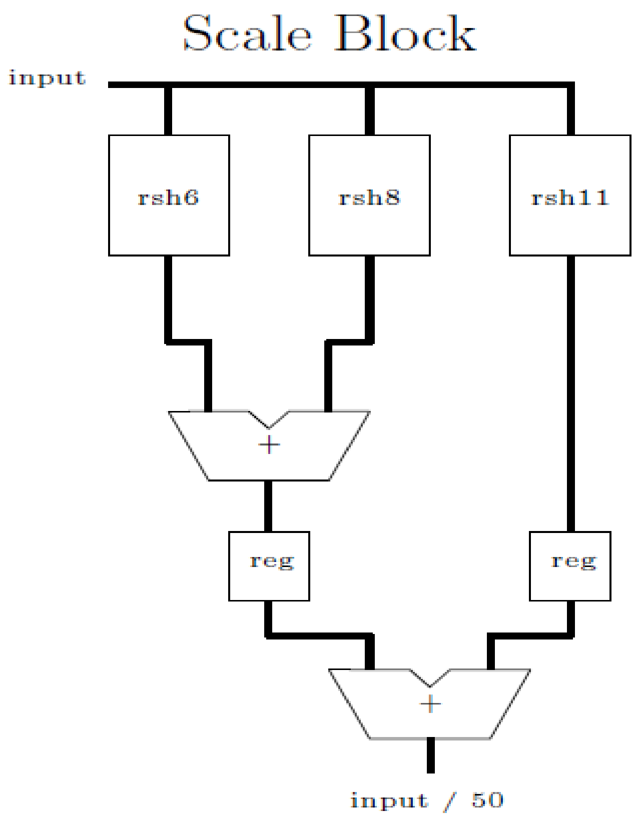


Fig -15

Chapter 3

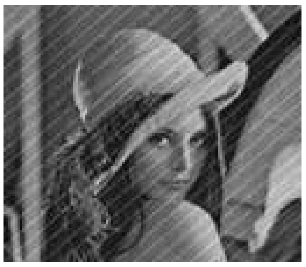
Experiemental Results & Future Scope

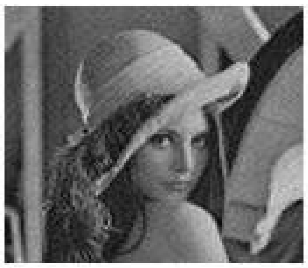
3.1 Software Performance

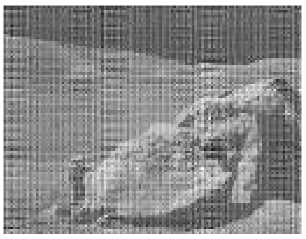
Frequency domain mean filter is one of the basic filter to remove periodic noise which is similar to the mean filter in spatial domain. The permormance is depends on the mask size used for the window operation and also on the threshold value which is not not adaptive in this case.For high frequency periodic noise 5x5 window operation gives the better result.The results are shown below.

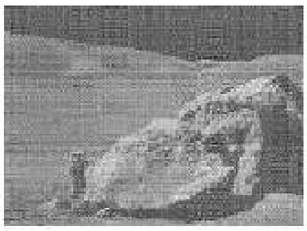
Noisy Image

Noise free image









Total time for processing for 128x128 noisy image is 2.913170 seconds

3.2 Experimental Setup & design Performance

To test the design in an FPGA board, every individual blocks need to be assembled and then combined designed is to loaded to FPGA board. There are five main building blcoks are there in the design namely FFT/IFFT processor, memory bank (RAM X & Y, address generation & phase generator), absolute blcok, register bank and pipe line mean block .As discussed size dc value depends on the size of image, so higher the image size higher will be the size of dc value. So data bus length will also increase. So design is tested as a prototype design for 8x8 image size. Design algorithm can be applied to any image size. 16 bit data bus is used in fixed point unsigned integer format. Adress bus can be of minimum 5 bits. All memory elements are used is dual port RAM type. Design is fully pipelined with single clock frequency upto 50 Mhz .Due to complexity of the design and hardware requirement FFT/IFFT processor is not included in the main design. 2-D Fourier Transform is taken in matlab and results are are then processed throgh FPGA for further processing. Also output of FPGA is further imported to matlab to take inverse fourier transform to see the actual noise free image .

Design is tested on XILINX Spartan 3E XC3S500E of package FT256 with speed grade -4 which has 500k gates available and 50Mhz on board clock oscillator .Simulation software XILINX 12.1 is used to generated .bit file.The steps which followed are

1. Main design has four inputs re1, re2, im1 and im2 which are supposed to reicive data after FFT operation. In matlab FFT of the image is taken and the data after FFT are logged in four coefficeint file (.coe). The coefficient files are loaded in xilinx single port rom by xilinx IP core.
2. Xilinx IP roms are included in the main design & data are read from the roms by simple addressing by up counter. This data are to be processed by the main design. One control input start from the fpga push button switch starts the process and image size N should be given from board .
3. Result should be sent to computer for futher processing in matlab. To see the result in the computer interfacing between the board and the computer is necessary. Interface is done through UART protocol over serial communication. The main design output is input to the uart blcok that converts the the for serial communication. Baud rate of 57600 is used .Software putty is used to read the data from the board. As soon as the start signal is given data will be displayed in putty window in hexadecimal format and also logged in a file for further processing .

3.3 Simulation result

Simulation result is given below for 8x8 image and 5x5 window operation .

Real matrix after FFT operation								
26.951	-0.9554	1.3207	0.0291		1.0926	0.0291	1.3207	0.9554
0.2034	-1.5262	-0.0857	1.1328		0.4247	0.9125	-0.0426	-0.0566
-0.2682	0.1238	0.4011	-0.186		0.1008	0.3804	-0.6313	0.6073
0.099	0.3156	-0.3339	-0.2735		0.1965	0.0618	-0.3396	0.6898
0.4448	0.0578	-0.3174	0.256		-0.0197	0.256	-0.3174	0.0578
0.099	0.6898	-0.3396	0.0618		0.1965	-0.2735	-0.3339	0.3156
-0.2682	0.6073	-0.6313	0.3804		0.1008	-0.186	0.4011	0.1238
0.2034	-0.0566	-0.0426	0.9125		0.4247	1.1328	37.914	36.474

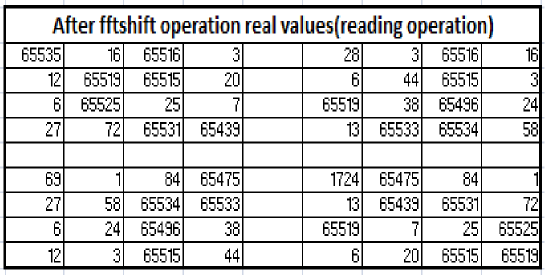
Imaginary data after FFT operation								
0	1.5528	0.1625	0.0626		0	-0.0626	-0.1625	-1.5528
-0.7497	0.1705	-0.9058	-0.3305		0.2751	0.3052	0.3857	0.9103
-0.3294	-0.2732	0.5151	-0.2195		-0.064	0.7206	-0.1222	0.1854
0.2645	0.0494	-0.126	0.2703		-0.3759	-0.0069	-0.1135	-0.0288
0	0.0998	-0.1274	-0.0747		0	0.0747	0.1274	-0.0998
-0.2645	0.0288	0.1135	0.0069		0.3759	-0.2703	0.126	-0.0494
0.3294	-0.1854	0.1222	-0.7206		0.064	0.2195	-0.5151	0.2732
0.7497	-0.9103	-0.3857	-0.3052		-0.2751	0.3305	0.9058	-0.1705

Converted to equivalent integer (6 bit fraction)								
1724	65475	84	1		69	1	84	65475
13	65439	65531	72		27	58	65534	65533
65519	7	25	65525		6	24	65496	38
6	20	65515	65519		12	3	65515	44
written to RAM IX columnwise					written to RAM IIX columnwise			
28	3	65516	16		65535	16	65516	3
6	44	65515	3		12	65519	65515	20
65519	38	65496	24		6	65525	25	7
13	65533	65534	58		27	72	65531	65439
with 2 FFT output value /clock								

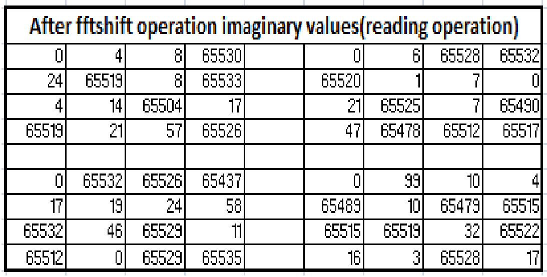
RAM IX addres(writing)					RAM IIX addres(writing)			
0	8	16	24		0	8	16	24
1	9	17	25		1	9	17	25
2	10	18	26		2	10	18	26
3	11	19	27		3	11	19	27
4	12	20	28		4	12	20	28
5	13	21	29		5	13	21	29
6	14	22	30		6	14	22	30
7	15	23	31		7	15	23	31
even & odd counter active port-A & B								

Converted to equivalent integer (6 bit fraction)								
0	99	10	4		0	65532	65526	65437
65489	10	65479	65515		17	19	24	58
65515	65519	32	65522		65532	46	65529	11
16	3	65528	17		65512	0	65529	65535
written to RAM IY columnwise					written to RAM IIY columnwise			
0	6	65528	65532		0	4	8	65530
65520	1	7	0		24	65519	8	65533
21	65525	7	65490		4	14	65504	17
47	65478	65512	65517		65519	21	57	65526
with 2 FFT output value /clock								

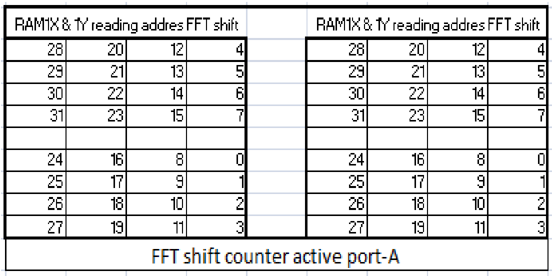
RAM IY addres(writing)					RAM IIY addres(writing			
0	8	16	24		0	8	16	24
1	9	17	25		1	9	17	25
2	10	18	26		2	10	18	26
3	11	19	27		3	11	19	27
4	12	20	28		4	12	20	28
5	13	21	29		5	13	21	29
6	14	22	30		6	14	22	30
7	15	23	31		7	15	23	31
even & odd counter active port-A & B								



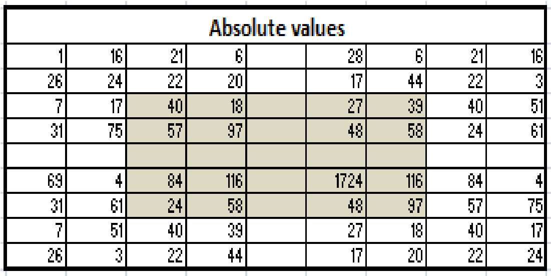
After fftshift operation real values(reading operation)								
65535	16	65516	3		28	3	65516	16
12	65519	65515	20		6	44	65515	3
6	65525	25	7		65519	38	65496	24
27	72	65531	65439		13	65533	65534	58
69	1	84	65475		1724	65475	84	1
27	58	65534	65533		13	65439	65531	72
6	24	65496	38		65519	7	25	65525
12	3	65515	44		6	20	65515	65519



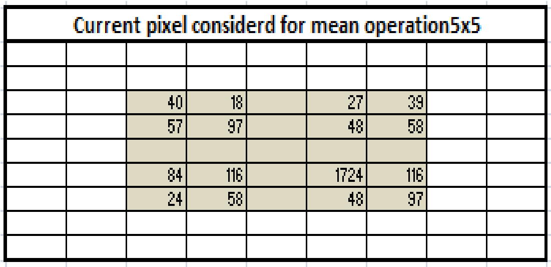
After fftshift operation imaginary values(reading operation)								
0	4	8	65530		0	6	65528	65532
24	65519	8	65533		65520	1	7	0
4	14	65504	17		21	65525	7	65490
65519	21	57	65526		47	65478	65512	65517
0	65532	65526	65437		0	99	10	4
17	19	24	58		65489	10	65479	65515
65532	46	65529	11		65515	65519	32	65522
65512	0	65529	65535		16	3	65528	17



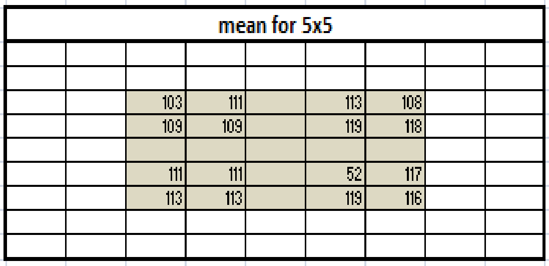
RAM1X & 1Y reading addres FFT shift					RAM1X & 1Y reading addres FFT shift			
28	20	12	4		28	20	12	4
29	21	13	5		29	21	13	5
30	22	14	6		30	22	14	6
31	23	15	7		31	23	15	7
24	16	8	0		24	16	8	0
25	17	9	1		25	17	9	1
26	18	10	2		26	18	10	2
27	19	11	3		27	19	11	3
FFT shift counter active port-A								



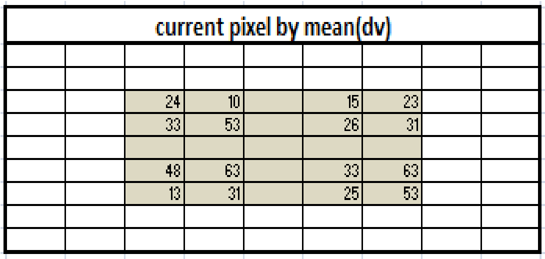
Absolute values								
1	16	21	6		28	6	21	16
26	24	22	20		17	44	22	3
7	17	40	18		27	39	40	51
31	75	57	97		48	58	24	61
69	4	84	116		1724	116	84	4
31	61	24	58		48	97	57	75
7	51	40	39		27	18	40	17
26	3	22	44		17	20	22	24



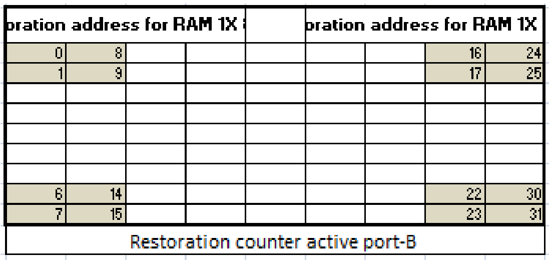
Current pixel considerd for mean operation5x5								
		40	18		27	39		
		57	97		48	58		
		84	116		1724	116		
		24	58		48	97		



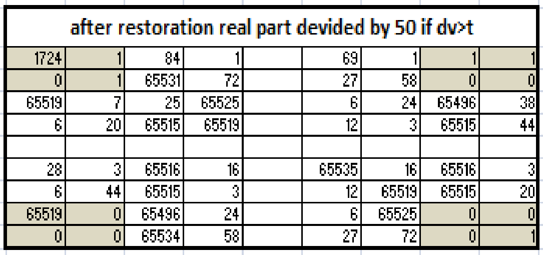
mean for 5x5								
		103	111		113	108		
		109	109		119	118		
		111	111		52	117		
		113	113		119	116		



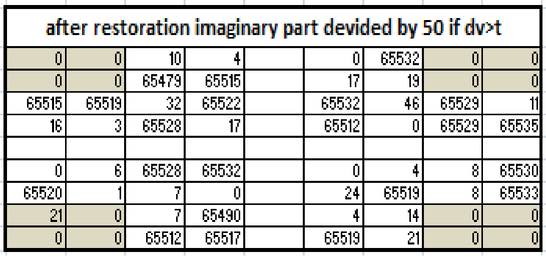
current pixel by mean(dv)								
		24	10		15	23		
		33	53		26	31		
		48	63		33	63		
		13	31		25	53		



pration address for RAM 1X					pration address for RAM 1X			
0	8						16	24
1	9						17	25
6	14						22	30
7	15						23	31
Restoration counter active port-B								



after restoration real part devided by 50 if dv>t								
1724	1	84	1		69	1	1	1
0	1	65531	72		27	58	0	0
65519	7	25	65525		6	24	65496	38
6	20	65515	65519		12	3	65515	44
28	3	65516	16		65535	16	65516	3
6	44	65515	3		12	65519	65515	20
65519	0	65496	24		6	65525	0	0
0	0	65534	58		27	72	0	1



after restoration imaginary part devided by 50 if dv>t								
0	0	10	4		0	65532	0	0
0	0	65479	65515		17	19	0	0
65515	65519	32	65522		65532	46	65529	11
16	3	65528	17		65512	0	65529	65535
0	6	65528	65532		0	4	8	65530
65520	1	7	0		24	65519	8	65533
21	0	7	65490		4	14	0	0
0	0	65512	65517		65519	21	0	0

3.4 Timing analysis

Timing analysis is an important issue in VLSI design. If an algorithm is to be implemented by hardware, besides other paRAMeters the time requirement for processing should be reduced. Hardware implementation of frequency domain algorithms involves task of conversion to frequency domain. So FFT and IFFT operation takes more time than time taken by actual algorithm .

Toal processing time is the summation of two times the processing of 2-d FFT/IFFT processor, time to write data in RAM IX or RAM IIX, two times the latency period of absolute blcok, time taken in register bank to form 3x3 masking operation, latency period of mean filter and time taken for restoration process .

The processing time bewteen after FFT operation and before IFFT operatin is

Total_time = Time to write into RAM IX or IIX (NxN/4) + absolute blcok latency period

(28 clk) + regitstor bank waiting time (2N) + latency period for

mean filtration (21 clk) + restoration time ((N/2 -1)xN) + NxN clk cycle

In table 2 processing time is calculated for different image size and compared with software processing time.

Processing time calculated (after FFT operation and before IFFT operation)		
Image size	Time taken for FPGA design for 50 MHZ clk	Time taken in software
8x8	0.03380us	0.203677s
16x16	0.10260us	0.291281s
32x32	0.37460us	0.285138s
64x64	1.45620us	0.750869s
128x128	5.76980us	1.914085s

. Table 2

3.5 Conclusion & Future Scope of Work

The literature describes the prototype design for frequency domain mask mean filter and realization using FPGA. Only by enlarging the size of memory elements the design can be realised for higher order image size. Other design constraints are to be enlarged acordingly.

An algorithm when implemented in hardware should give same performance as it gives in case of sofware applications .But when an algorithm is implemented in hardware, it has some amount of error due to some approximations. Some of the approximations are necessary in case of hardware application to minimize the hardware requirement. Like fixed multiplication of filxed division can be performed by scale blcok with approximation. So scale blcok has some amount of error depending upon the no of bits used to represent fractional part in case of fixed point integer representation .Higher the bits are used lower will be the error but will not be error free .As a scale blcok can’t replace a devider or multiplier. FFT/IFFT processor will have some error and this error will propagete through the process then in the scale blcok it will have extra added error. Cotrolled 1/N devider will produce some amount of error and again IFFT process will add some error. So error is added due to two times by FFT/IFFT processor, by two scale blcoks and by mean calculator. So in order to get right result at the output sufficient no of bits must be there so that error can be neglegted.

Another aspect of the design is that it depends on how FFT/IFFT processsor perfoms .If throughput is 2 then also time taken between FFT operation and IFFT operation is more or less same. But FFT/IFFT processor with throughput 2 is very slow. So throughput 4 or higher is prefferd. For higher throughput no of RAMs will differ and masking operation will be arranged acordingly. The design is fully pipelined and maximum frequency it can achieve is 1/(time to add two 32 bit number).

Hardware implementation of the frequency domain mean filter is a first effort to realize frequency domain filters while designs are only available to realize the spatial domain filters. The design demonstrates the basic strategy to design frequency domain filters. Other type of filters are also can be designed just by replacing the mean filter blcok .The frequency domain median filter [19] is similar in operation but in case of mean median is to be calculated. Adaptive optimum notch filter [21] shows better result in removal of periodic noise than the other two algorithms. A specific design which is less complex and shows better result can be implemented in hardware. Processing time taken can be much more in case of software but that can be minimised when implemented in hardware. As the strategy has been defined, the future scope of work will confined in implementing an algorithm which shows better result. Threshold value which is manualy selected in case of frequency domain mean filter is adaptive in adaptive optimum notch filter. So the method of adapting the threshold is also to be implemented in hardware .

References

1. Vaijyanath kunchigi , Linganagouda Kulkarni and Subhash Kulkarni ‘’ Low power Square and Cube Architectures Using Vedic Sutras’’Fifth International Conference on Signals and Image Processing,2014.

2. MAJERSKI, S. : ‘Square-rooting algorithms for high-speed digital circuits’, IEEE Trans. Comput., 1985, C-34, (8), pp. 724-733

3. SCOTT, N.R.: ‘Computer number systems & arithmetic’ (Prentice-Hall, Englewood Cliffs, 1985)

4. KUNZ, K.S.: ‘Numerical analysis’ (McGraw-Hill, New York, 1957)

5. SOUTHWORTH, R.W., and DELEEUW, S.L.: ‘Digital computation on numerical methods’ (McGraw-Hill, New York, 1965)

6. ERCEGOVAC, M.D.: ‘An on-line square rooting algorithm’, Proc. 4th IEEE Symposium on Computer Arithmetic, Santa Monica, CA, pp. 183-189, October 1978

7. VOLDER, J.E.: ‘The CORDIC trigonometric computing technique’, *IRE Trans. Electron. Comput.,* 1959, EC-8, (9), pp. 33C334 .

8 DELUGISH, B.G.: ‘A class of aglorithms for automatic evaluation of certain elementary functions in a binary computer’. Ph.D. Dissertation, Dept. of Computer Science, University of Illinois, Urbana, June 1970 .

9 P. Montuschi, PhD and Prof. M. Mezzalama,” Survey of square rooting algorithms “*IEE PROCEEDINGS, Vol. 137, Pt. E, No. I , JANUARY 1990* .

10 Y. Li and W. Chu, “A New Non-Restoring Square Root Algorithm and Its VLSI Implementations”, Proc. Of 1996 IEEE International Conference on Computer Designs:VLSI in Computers and Processors, Austin, Texas, USA, October 1996,pp538-544 .

11 Li Yamin, Chu Wanming, "Parallel-Array Implementations of a Non-restoring Square Root Algorithm," Computer Design: VLSI in Computers and Processors, 1997,pp:690-695.

12 K N Vijeyakumar, Dr V Sumathy, P Vasakipriya and A Dinesh Babu “FPGA Implementation of Low Power High Speed Square Root Circuits “IEEE International Conference on Computational Intelligence and Computing Research ,2012 .

13 N.U. Chowdary and Willen Steenmart, “A HIGH SPEED TWO-DIMENSIONAL FFT PROCESSOR”,in IEEE,1984.

14 Teresa *M. Pytosh, Albert0 M. Magnani,* “A NEW PARALLEL 2-D FFT ARCHITECTURE” V2.4 in IEEE,1990.

15 Shiqun Zhang, Dunshan Yu, “Design and Implementation of A Parallel Real-time FFT Processor”,in IEEE,2004

16 Sheng Chen, Ying Yu,Binglai Chen,Songlin Lai,Yibin Zeng,Yazhen Zhang and Cunguang Wang, “Design of a 128-point Fourier Transform Chip for UWB Applications” in IEEE ,2006

17 Ren Chen, Neungsoo Park and Viktor K. Prasanna , “High Throughput Energy Efficient Parallel FFT Architecture on FPGAs” . by DARPA under grant number HR0011-12-2-0023.

18 Dong-Sun Kim, *Member*, IEEE, and Seung-Yerl Lee. “Dual Input Radix 23 SDF IFFT/FFT Processor for Wireless Multi-Channel Real Sound Speakers using Time Division Duplex Scheme” ,in IEEE,2009

19 Aizenberg, C. Butakoff, "Frequency Domain Median-like Filter for Periodic and Quasi-Periodic Noise Removal," SPIE Proceeding, Vol. 4667, 181-191, 2002.

20 Aizenberg, C. Butakoff, J. Astola, K. Egiazarian, "Nonlinear Frequency Domain Filter for the Quasi-Periodic Noise Removal," in Proc. 2002 International TICSP Workshop on Spectral Methods and Multirate Signal Processing, pp. 147-153.

21 4. Adaptive Optimum Notch Filter for Periodic NoiseReduction in Digital Images by Payman Moallemi * and Majid Behnampourii *Amirkabir / Electrical & Electronics Engineering / Vol . 42 / No.1 / Spring 2010*

22 A Low-Cost VLSI Implementation for Efficient Removal of Impulse Noise by Pei-Yin Chen, *Member, IEEE*, Chih-Yuan Lien, *Student Member*, *IEEE*, and Hsu-Ming Chuang , IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 18, NO. 3, MARCH 2010

23 Ayan Banerjee Anindya Sundar Dhar ,Swapna Banerjee ,“ FPGA realization of a CORDIC based FFT processor for biomedical signal processing” an ieee journal .

24 R.C. Gonzalez, R.E. Woods, Digital Image Processing, 3rd Edition, Prentice Hall, 2007.

.