Term Paper On

STUDY OF FREQUENCY DOMAIN DENOISING FILTERS AND REALIZATION USING FPGA

Submitted for Partial Fulfilment 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

May 2015

INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, Shibpur

HOWRAH-711103

Forward

I hereby forward the Term Paper entitled "**Study Of Frequency Denoising Filters and Realization Using FPGA"** prepared by Shirshendu Roy under my guidance and supervision in partial fulfilment 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
Electronics & Tele-Communication

Countersigned by

……
(Dr. Santanu Das)

Professor and Head
Department Electronics &
Telecommunication Engineering,
IIEST, Shibpur

.....
(Dr. Amit Kr Das)

Dean Academic
IIEST, Shibpur

INDIAN INSTITUE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR

HOWRAH- 711103

CERTIFICATE OF APPROVAL

The forgoing Term Paper is hereby approved as a creditable study of an engineering subject titled "**STUDY 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 term paper 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 AND 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 term paper 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 term paper directly or indirectly.

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

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 kinds 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 theoretical background i.e. about various kind of noise models known to us and about filters which are innovated till now for reduction of those noises. Introductory lessons about FPGA are discussed in the 2^{nd} chapter .3^{rd} chapter is concentrating upon FPGA implementations of image denoising algorithms which are implemented so far and tells about filter design & our proposed work .

Chapter 1

Theoretical Background

The sources of noise in digital images arise during image acquisition (digitization) and transmission .Imaging sensors can be affected by ambient conditions .Interference can be added to an image during transmission .Image restoration attempts to restore images that have been degraded .Identify the degradation process and attempt to reverse it .Similar to image enhancement, but more objective

The degradation process is modelled as a degradation function together with additive noise term operates on input image f(x, y) to produce a degraded image g(x, y). From a given g(x, y) to estimate ḟ(x, y) of the original image need some knowledge about degradation function and noise term. The more we know about h(x, y) and η(x, y), ḟ(x, y) will be more close to f(x, y).

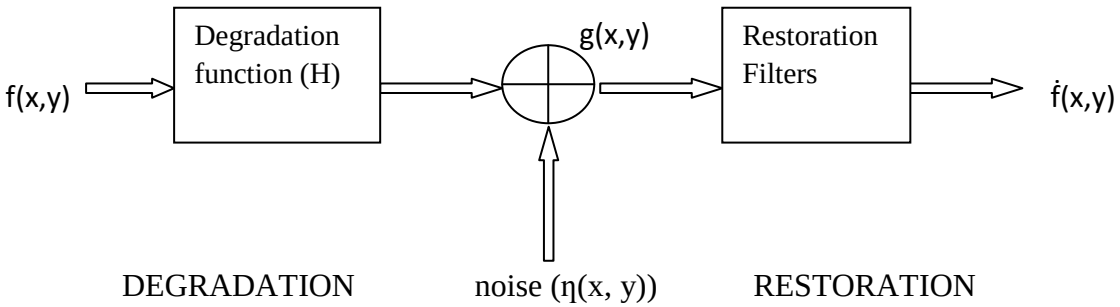
If H is a linear position invariant process then the degraded image in spatial domain is given by:

$$G(x, y) = h(x, y) * f(x, y) + \eta(x, y)$$

where, h(x,y) is the spatial the spatial representation of degradation function and * indicates that its a convolution operation. So, in frequency domain degraded image can be represented as:

$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

Where the terms in capital letters represents Fourier transforms of corresponding terms.



1.1 Noise Models:-

In order to restore an image needs proper knowledge of degradation function and noise behaviour whether it is additive or multiplicative .The spatial descriptor with which we shall be concerned is statistical behaviour of gray level values in noise component. This may be considered random variables characterized by probability Density Function (PDF) of a noise model .Following are the most common PDFs found in image processing applications.

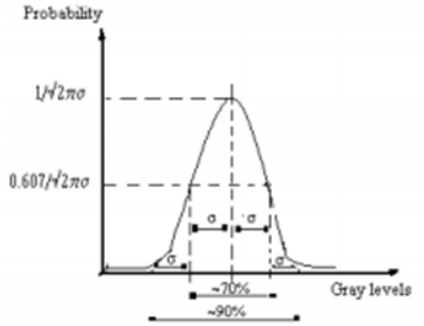
Gaussian Noise:

Because of its mathematical tractability in both spatial and frequency domain, Gaussian noise models are frequently used. The Gaussian noise has a normal (Gaussian) probability density function.

$$PDF_{Gaussian} = \frac{1}{\sqrt{2\pi\,\sigma^2}} e^{-\frac{(g-\mu)^2}{2\sigma^2}}$$

where, g=grey level μ= mean

σ^2 = variance

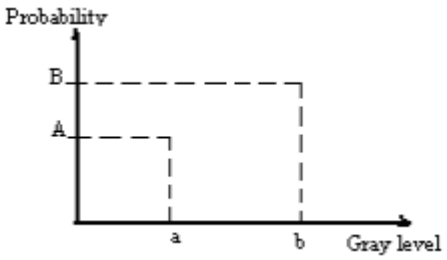


Approximately 70% of the values are contained between μ±σ and 90% of the values are contained in μ ± 2σ.

Salt & Pepper noise :

In salt & pepper noise only two possible values possible ,a and b,and probabilty of obtaining each of them is less than 0.1 .(otherwise noise will vastly dominate the image)For a Gray scale image intensity value for pepper noise is 255 and that of salt noise is 0 .The pdf of salt & pepper noise is as shown below .

$$PDF = \begin{cases} A, & for\ g = a(\text{"pepper"}) \\ B, & for\ g = b(\text{"}salt\text{"}) \end{cases}$$



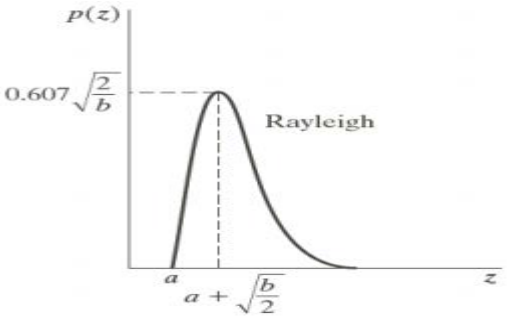
Rayleigh Noise Model :

The pdf of Raylegh noise is given by

$$p(z) = \begin{cases} \frac{2}{b} e^{-\frac{(z-\mu)^2}{2\sigma^2}}, & z \geq a \\ 0, & x < a \end{cases}$$

$\mu = a + \sqrt{\pi b/4}$ and

$\sigma^2 = \frac{b(4-\pi)}{4}$



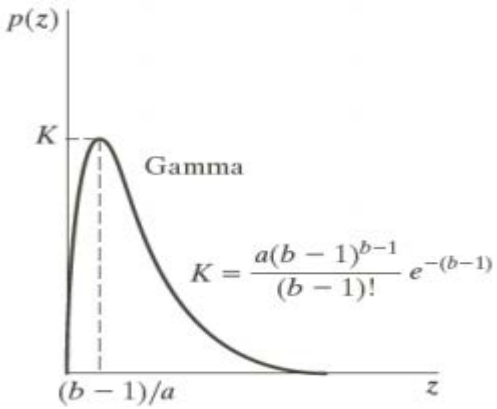
Displacement from origin and the fact that basic shape of the density is skewed to the right .Rayleigh density function can be quite useful to approximate skewd histograms .

Erlang(Gamma) Noise Model :

The pdf of Gamma noise model is given by

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az}, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

$\mu = \frac{b}{a}$ and $\sigma^2 = \frac{b}{a^2}$



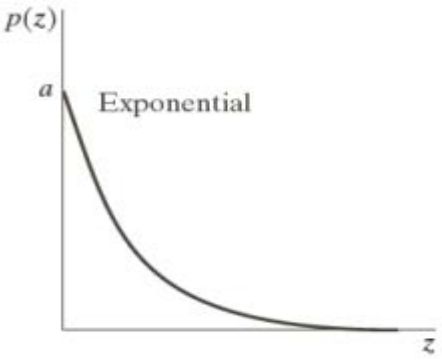
Where the parameters are such that a>0 , and ! represents factorial .when the denominator is gamma function then pdf is called Gamma model.

Exponential Noise Model :

The pdf of exponential noise model is given by

$$p(z) = \begin{cases} ae^{-az}, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

$\mu = \frac{1}{b}$ and $\sigma^2 = \frac{1}{a^2}$

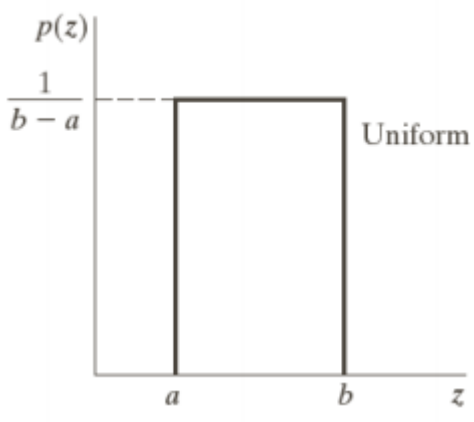


Where a >0 this is a special case of Erlang pdf with b=1.

Uniform Noise Model :

The pdf of uniform noise model is

$$p(z) = \begin{cases} \frac{1}{b-a}, & a \le z \le b \\ 0, & otherwise \end{cases}$$



$$\mu = \frac{a+b}{2} \text{ and } \sigma^2 = \frac{(b-a)^2}{12}$$

Periodic Noise:

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.

1.2 Spatial Filtering:

When only degradation present in image is noise and it is additive in nature then the image can be restored by simply subtracting the noise term from the image.

$$g(x,y) = f(x,y) + \eta(x,y)$$

Spatial filtering is a method of choice in situations where noise is additive in nature.

1.2.1 Linear Mean Filtering

Arithmetic Mean Filter:

The *arithmetic mean* filter is a very simple one. S_{xy} is the sub image window of size mxn. Mean of all the pixels of S_{xy} is calculated .Value of restored image at any point (x, y) is simply the mean of S_{xy}.

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s,t)$$

The operation can be implemented using a simple mask of size mxn (generally 3x3 or may be of 5x5 for higher noise frequency) .Mean filter smoothes the local variations in an image with a result of blurring

Geometric Mean Filter:

Geometric mean filter can be constructed by calculating Geometric mean by following expression in a size of window of mxn.

$$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

Achieves comparable smoothing to the arithmetic mean, but tends to lose less image detail

Harmonic Mean Filter:

Harmonic mean can be calculated as follows

$$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$$

Works well for salt noise, but fails for pepper noise .Also does well for other kinds of noise such as Gaussian noise

Contra-harmonic Mean:

Contra-harmonic mean filter restores image with the expression

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

Q is the *order* of the filter and adjusting its value changes the filter's behaviour Positive values of *Q* eliminate pepper noise .Negative values of *Q* eliminate salt noise .It reduces to arithmetic mean filter with Q=0 and reduces to Harmonic mean filter with Q=-1

1.2.2 Non-linear Order Statistics Filter:

Order statistics filter are those whose response is based upon the ordering of the pixels in the image area which is encompassed by the filter .The response of the filter at any point is determined by the ranking result.

Median Filter:

This is the best known order statistics filter .It replaces the value of a pixel by median value calculated in the specific window S_{xy}.

$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{median}\{g(s,t)\}$$

Excellent at noise removal, without the smoothing effects that can occur with other smoothing filters .Particularly good when salt and pepper noise is present

Max Filter and Min Filter:

The expression for Max filter is as follows

$$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}} \{g(s,t)\}$$

The filter is useful in finding the brightest point in an image. Also as pepper noise has low values, so pepper noise can be removed by Max filter.

The expression for Min filter is

$$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\}$$

Min filter is useful in finding the darkest point in an image .So Min filter can used to remove salt noise.

Midpoint Filter:

The midpoint filter simply computes the midpoint between maximum and minimum valued pixel in the area encompassed by the filter

$$\hat{f}(x,y) = \frac{1}{2}\left[\max_{(s,t)\in S_{xy}} \{g(s,t)\} + \min_{(s,t)\in S_{xy}} \{g(s,t)\} \right]$$

This filter combines the effect of average and order statistics filter .Good for random Gaussian and uniform noise

Alpha-Trimmed Mean Filter:

Suppose we delete the d/2 lowest and d/2 highest gray level values of g(x, y) in the neighbourhood $S_{x,y}$ and g_r (s, t) represents the remaining (mn-d) pixels .A filter by averaging these remaining pixels is thus formed .

$$\hat{f}(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{xy}} g_r(s,t)$$

Where d can range between 0 to mn-1 .When d = 0 alpha trimmed filter reduces to simple arithmetic filter and when d =mn-d/2 this filter reduces to median filter .It is useful when situations involving multiple kind of noise.

1.2.3 Adaptive Spatial Filters:

Weighted Median Filter

Weighted median filter is the extension of median filter. In weighted median filter, it specified pixels within a local neighborhood are repeated a given number of times in the Computation of the median value. The basic idea is to give weight to the each pixel. Each pixel is given a weight according to its spatial position in the window.

Centered Weighted Median Filter

The centered weighted median filter is similar to weighted median filter. Center weighted median filter is giving more weight only to the central value of each window.

Tri state Weighted Median Filter

Tri state weighted median filter, is working according to the threshold value. The range of threshold is 0 to 255. First, it takes a pixel from the window then it checks with the current pixel and the threshold value. If the current pixel is less than the threshold value, we can say that current pixel is an uncorrupted pixel and keep the pixel as it is, else calculate Center Weighted Median filter and Standard Median filter value. Check, whether the threshold is placed between CWM and SM, and then the corrupted pixel is replaced with CWM value. If the SM is greater than the threshold then the SM value is placed instead of noisy pixel.

Adaptive Median Filter

In Adaptive median filter, it uses various window sizes to noise reduction. Size of window increases until correct value of median is calculated and noise pixel is replaced with calculated median value. In this case, two conditions are used: one is to detect corrupted pixels and second is to check correctness of median value. If the calculated median value is less than minimum value present in window and greater than maximum value present in window then median value is treated as corrupted value. If the calculated median value is corrupted, then increase the window size and again calculate the median value until we get correct median value

Switching Median Filter

In switching median filter, there are two steps there. First, a test decides whether or not a given pixel is contaminated by impulsive noise. A pixel is contaminated, if the absolute difference between the median value in its neighborhood and the value of current pixel itself is greater than a given threshold. If contaminated, a classical median filter is applied. If not, the current pixel is noise free and will not be modified**.**

Progressive Switching Median Filter

Progressive switching median filter is used for removing salt and pepper impulsive noise from the image. In this case, first take one pixel check whether the pixel value is less than the minimum value present in the window value and also check whether the pixel value is greater than the maximum value present in the window then it is a corrupted pixel. Corrupted pixel is replaced my median value. If the calculated median value is corrupted pixel, then increase the window size and recalculate the median value until get correct median value.

Rank-Order Based Adaptive Median Filter

In Rank-order based Adaptive Median Filter, a pixel is retained if the median of a window around it is between the minimum and the maximum value of the window and the pixel also is between the minimum and the maximum value of the window, In case, the median of the

window is between the minimum and the maximum value of the window and the pixel does not lie between the minimum and the maximum value of the window, then the pixel is replaced by the median value; otherwise, the size of the window is increased and the pixel will be replaced by the median of the increased size window, provided, this new median is between the minimum and the maximum value of the window, otherwise, the window size is again increased up to some pre-fixed maximum level, beyond which the central pixel is left unchanged.

Cloud Model Filter

In cloud model filtering, there are two operation is there. First is noise detection second is filtering noise. For noise detection it is check whether the current pixel is between maximum and minimum value of the window. If it is between maximum and minimum value then it is called uncorrupted pixel. If it is not between the values then it is called corrupted pixel. By the help of weighted fuzzy mean filter we changed the corrupted pixel value.

1.3 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.

1.3.1 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

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

Similarly original function can be obtain from

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

The Fast Fourier Transform. Without an efficient method of calculating a DFT, it would remain 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.

1.3.2 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 equations can be written in the following manner.

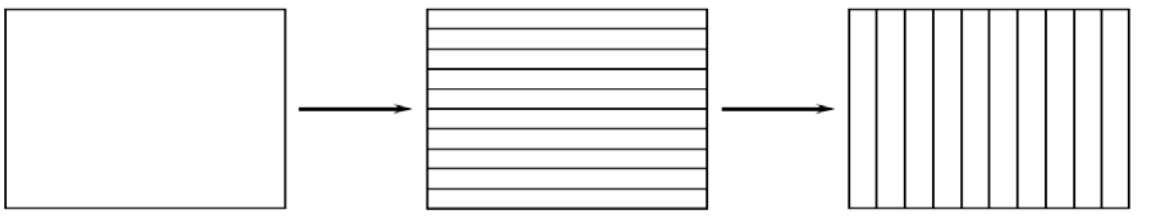
$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{[-2\pi j\left(\frac{xu}{M}+\frac{yv}{N}\right)]}$$

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

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

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 .



(a) Original image (b) DFT of each row of (a) (c) DFT of each column of (b)

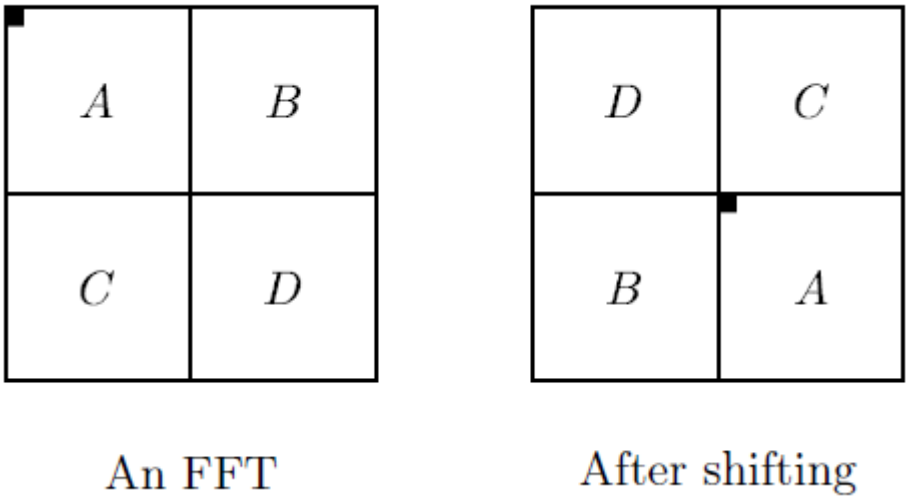
The DC Coefficient.

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.

For purposes of display, it is convenient to have the DC coefficient in the centre of the matrix. This will happen 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 submatrix and is shown as a black square.



An FFT After shifting

1.3.3 Basic of filtering in Frequency domain:

Basic filtering steps are as followed.

1. Perform two dimensional 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.

1.4 Basic Frequency Domain Filters:

Low pass filtering;

Suppose we have a Fourier transform matrix F, shifted so that the DC coefficient is in the centre. Since the low frequency components are towards the centre, we can perform low pass filtering by multiplying the transform by a matrix in such a way that centre values are maintained, and values away from the centre are either removed or minimized. One way to do this is to multiply by an ideal low-pass function H (u, v) defined by:

$$H(u,v) = \begin{cases} 1, & D(u,v) \le D_0 \\ 0, & D(u,v) > D_0 \end{cases}$$

D_0 is a specified non negative quantity and D(u, v) is the distance from (u, v) to the origin of the frequency quantity .

High pass filtering

Just as we can perform low pass filtering by keeping the centre values of the DFT and eliminating the others, so high pass filtering can be performed by the opposite: eliminating centre values and keeping the others.

$$H(u,v) = \begin{cases} 1, & D(u,v) > D_0 \\ 0, & D(u,v) \le D_0 \end{cases}$$

D_0 is a specified non negative quantity and D(u, v) is the distance from (u, v) to the origin of the frequency quantity .

Butterworth Filtering:

Ideal filtering simply cuts off the Fourier transform at some distance from the centre. This is very easy to implement, as we have seen, but has the disadvantage of introducing unwanted artifacts, ringing, into the result. One way of avoiding this is to use as a filter matrix, a circle with a less sharp cutoff .A popular choice is to use Butterworth filters.
Transfer function of a Butterworth low pass filter is

$$H(u,v) = \frac{1}{1+[D(u,v)/D_0]^{2n}}$$

And that of a high pass filter is

$$H(u,v) = \frac{1}{1+[D_0/D(u,v)]^{2n}}$$

Gaussian Filters:

Filters based on Gaussian functions are of particular importance as their shapes are easily specified and because both forward and inverse Fourier transforms of a Gaussian function are real Gaussian functions .Transfer function of a Gaussian low pass filter is

$$H(u, v)=e^{-\frac{D^2(u,v)}{2D_0{}^2}}$$

And that of a Gaussian high pass filter is

$$H(u, v) =1 - H(u, v)=e^{-\frac{D^2(u,v)}{2D_0{}^2}}$$

Band Reject Filter

Band reject filters remove or attenuate a band of frequencies about the origin of Fourier transform .Expression for band reject filter is.

$$H(u,v)=\begin{cases}1 & if\ D(u,v) < D_0 - \frac{W}{2} \\ 0 & if\ D_0 - \frac{W}{2} \le D(u,v) \le D_0 + \frac{W}{2} \\ 1 & if\ D(u,v) > D_0 + \frac{W}{2}\end{cases}$$

Where D (u, v) is the distance from the centre of the centred frequency rectangle, W is the width of the band and D_0 is radial centre .

Similarly Butterworth band reject filter of order n is given by expression.

$$H(u ,v) =\frac{1}{1+[\frac{D(u,v)D_0}{D^2(u,v)-D_0{}^2}]^{2n}}$$

And Gaussian band reject filter is given by

$$H(u, v) = 1 - e^{-\frac{1}{2}[\frac{D^2(u,v)-D_0{}^2}{D(u,v)W}]^2}$$

Band Pass Filter:

Band pass filter performs the opposite operation of a band reject filter. Transfer function of band pass filter can be obtained from the expression of band reject filter.

$$H_{bp}(u, v) = 1 - H_{br}(u, v)$$

Notch Filtering

A notch filter rejects (or passes) frequencies in predefined neighbourhood about the centre frequency .Due to the symmetry of the Fourier transform notch filters must be appear in symmetric pairs about the origin in order to meaningful result .Exception of that rule is when notch filter is placed in centre frequency. No of pairs of notch filter as well as shape of notch areas of can be implemented arbitrarily.

The transfer function of a ideal notch reject filter of radius D_0, with centres at (u0, v0) and symmetry at (-u0, -v0) is

$$H(u,v) = f(x) = \begin{cases} 0, & if\ D_1(u,v) \le D_0\ or\ D_2(u,v) \le D_0 \\ 1, & otherwise \end{cases}$$

Where

$$D1(u,v) = \left[(u - \frac{M}{2} - u_0)^2 + (v - \frac{N}{2} - v_0)^2\right]^{\frac{1}{2}}$$

$$D2(u,v) = \left[(u - \frac{M}{2} + u_0)^2 + (v - \frac{N}{2} + v_0)^2\right]^{\frac{1}{2}}$$

The assumption is that the centre of the frequency rectangle has been shifted to the point (M/2, N/2), according to the filtering procedure .therefore values of (u0 ,v0) are with respect to the shifted centre .

Transfer function of butter worth notch filter is of order n is

$$H(u,v) = \frac{1}{1+\left[\frac{D^2(u,v)}{D_1(u,v)D_2(u,v)}\right]^n}$$

A Gaussian notch reject is of the form

$$H(u,v) = 1 - e^{-\frac{1}{2}\left[\frac{D_1(u,v)D_2(u,v)}{{D_0}^2}\right]}$$

This filter become high pass with $u_0 = v_0 = 0$

Expression of notch pass filter can be obtained from notch reject filter expression is as

$$H_{np}(u,v) = 1 - H_{nr}(u,v)$$

1.5 Periodic Noise Removal

It is previously discussed that periodic noise can't be easily removed in spatial domain .Then to remove periodic noise one must go for frequency domain as periodic noise gives star like high amplitude region located at certain distance of on a radius from the average value of the image itself. So, removing periodic noise form an image involves removing a particular range of frequencies from that image.Several such filter algorithms have been developed, some of them are discussed below.

1.5.1 Band Reject Filters

Band reject filters can be used for periodic noise removal. Since band reject filters attenuate a band of frequency about the origin of the 2-D Fourier transform, these filter can be uses in periodic noise reduction applications where the general location of the noise components is approximately known in the frequency domain [1].Unfortunately, when the distances of the periodic noise components about the origin of the 2-D Fourier transform are different, there is necessary to use either a wide band reject filter, or several narrow band-reject filters. In both cases, the restored image may lose some important image information.

1.5.2 Notch-Reject Filters

A notch-reject filter attenuates frequencies in predefined neighborhoods about a center frequency of the 2-D Fourier transform. Therefore, this filter can be used to reduce the periodic noise effects where the main frequencies of the periodic noises are known .Automatic determining of both the periodic noise main frequencies and the corresponding band-width are challenge problems in this filter.

1.5.3 Optimum Notch Filters

In a real system contaminated by periodic noise, the output image tends to contain 2-D periodic structure superimposed on the input image with patterns more complex than several simple sinusoidal signals. In this condition, two mentioned methods are not always acceptable because they may remove much image information in filtering process.
Optimum notch filter [1] tries to minimize local variance of the restored image . At the first stage, principal contribution of the inference repetitive pattern is extracted from the noisy image, and then the output image is restored by subtracting a variable weighted portion of the repetitive pattern from the contaminated image. The extractions of the repetitive pattern is implemented in frequency-domain by applying a proper notch-pass filter on every periodic noise frequency, and then by applying inverse 2-D Fourier transform to restore the repetitive pattern in spatial-domain.
The 2-D Fourier transform of the inference repetitive pattern, $N(u,v)$, is given by

$$N(u,v) = H_{np}(u,v)G(u,v)$$

Where, $H_{np}(u,v)$ is superimposed response of all necessary notch-pass filters and $G(u,v)$ is the 2-D Fourier transform of the contaminated image. The proper selecting of $H_{np}(u,v)$ is challenge problem in the optimum notch filter.

Then, the corresponding repetitive pattern in the spatial-domain, $\eta(x, y)$, is obtained by taking inverse Fourier transform of $N(u ,v)$. For an additive noise model, if $\eta(x,y)$ is known perfectly, subtracting the repetitive pattern from the noisy image, $g(x,y)$, obtains noise-less input image, $f(x,y)$. On the other hand, the filtering procedure by applying $H_{np}(u,v)$ usually yields only an approximation of the true repetitive pattern. In order to minimize the effects of components not present in the estimate of real true repetitive pattern, subtracting a variable weighted portion of $\eta(x, y)$ from $g(x, y)$ is done to obtain an estimate of $f(x ,y)$.

$$\dot{f}(x,y) = g(x,y) - w(x,y)\eta(x,y)$$

Where $\hat{f}(x, y)$ is the estimate of $f(x, y)$ and $w(x, y)$, the weighting function, is to be determined such that the variance of the output is minimized over a specific neighborhood of every point (x, y). Usually, the noisy image is partitioned to several non-overlapped neighborhood of size $(2a+1)\times(2b+1)$ about points (x, y), and corresponding $w(x, y)$ in each neighborhood is considered constant. For each neighborhood, the constant weighting function $w(x, y)$ is obtained by equation as shown .Which tries to minimize local variance of the restored image, over corresponding neighborhood.

$$w = w(x,y) = \frac{\overline{g(x,y)\eta(x,y)} - \overline{g}(x,y)\overline{\eta}(x,y)}{\overline{\eta^2}(x,y) - \overline{\eta}^2(x,y)}$$

Where $g(x, y)$ and $\eta(x, y)$ are the local mean of the corrupted image and the noise pattern, respectively, $\eta^2(x, y)$ is the local mean of square of the noise pattern, and $g(x, y)\eta(x, y)$ is the local mean of the corrupted image multiply by the noise pattern, all of them over corresponding neighborhood. In order to apply the optimum notch filter, firstly the noise pattern is extracted, then for each neighborhood, the constant weighting function $w(x, y)$ is obtained and finally, the restored image is obtained.

1.5.4 Frequency-Domain Median Filters

The frequency-domain median filter which is used to reduce the periodic noise effect contains two basic steps. In the first step, the median value of amplitudes of each frequency is computed over a predefined window. Then frequencies of the periodic noise are detected by comparing the median value of each frequency with its corresponding amplitude. In the second step, amplitude of each noisy frequency is replaced by it corresponding median value. The realization of the frequency-domain median filter for the frequency of (u, v) is summarized by

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

where $X(u, v)$ is $|G(u, v)|$, $\mathrm{Med}(X(u, v))$ is the median value of $X(u, v)$ over neighborhood frequencies of the (u, v) central frequency, θ is a predefined fixed threshold value and $Y(u, v)$ is the estimate of $|F(u, v)|$. Since $X(0,0)$ is the DC component of the contaminated image and its value is usually very large, the frequency-domain median filter should not apply on the frequency of (0,0). The window size of 5×5, 7×7, 9×9 and 11×11 are proposed for the frequency-domain median filters. In fact, the window size is related to band-width of periodic noise, and should be selected by trial and error. Meanwhile, the predefined fixed threshold value of 3 and 6 are proposed for the window sizes of 5×5 and 7×7, respectively [2].

1.5.5 Frequency-Domain Masked Mean Filters

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. The masked mean value is defined over an $N \times N$ masked window. All values in the $N \times N$ masked window are '1', except the center which is considered '0'. It means that the center of the $N \times N$ 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 .

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

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 [3].

1.5.6 Adaptive Notch Filter

This is filter [4] is based on optimum notch filter but adaptive in nature. Main frequencies of periodic noises in spectral domain can be detected by applying a proper threshold after masking the low frequency of the input image.

In order to accurately detect the corresponding pattern of the periodic noise in spatial domain, it is necessary to consider all contaminated frequencies around the main frequencies of the periodic noise. In this method, these contaminated frequencies are detected by a simple region growing around each main frequency of the periodic noise. After providing the pattern of the periodic noise in spatial domain by applying the inverse 2-D Fourier transform on all frequencies of the periodic noises, the input image is restored by the conventional optimum notch filter. The amplitude of frequencies of the periodic noise is locally greater than its neighborhood. Therefore, they can be detected by applying a proper threshold in the spectral of image. On the other hand, the low frequency region of ordinary images contains most of image information image and corresponding amplitude is greater than other frequencies. It means that it is necessary to mask the low frequency region before applying the threshold. The radius of this mask as R_{mask} is considered.

Non-overlapped concentric circles at frequency center, with radius of R and width of *w*, in the image spectra is considered .After masking the low frequency region ,for detecting the main frequency of each periodic noise, equation for proper threshold is

$$A_{thr} = (A_{max} + A_{mean})/2$$

A_{Thr} is the proposed threshold, A_{max} and A_{mean} are the maximum and mean of amplitudes of spectra, respectively, after masking the low frequency region. This simple threshold has been already used for effectively detecting the high temperature point targets in infrared image. Detecting Other Frequencies of Periodic Noise Applying the computed threshold detects all main frequencies of periodic noises, but for accurate extraction of the periodic noise pattern in spatial domain, the other frequencies of the periodic noise should be detected and considered. In this step, a simple region growing algorithm which considers a 3×3 window

around each main frequency as primary region is proposed in the spectral domain. At the first step of region growing, a surrounding 5×5 window is considered and the proposed region growing algorithm tries to find which frequencies in perimeter of surrounding 5×5 window are contaminated by the periodic noise by comparing the amplitude of neighborhood frequencies. Each frequency in perimeter of 5×5 window is considered as periodic noise, if the corresponding amplitude is less than the amplitude of neighborhood frequencies in 3×3 window. If the number of frequencies contaminated by periodic noise in perimeter of 5×5 window is greater than the predefined threshold, N_{Thr}, the region growing process repeats for a surrounding 7×7 window. In the proposed region growing algorithm, the N_{Thr} is set to half of pixels in perimeter of surrounding window which is ($2n$-2) where n is surrounding window size and n is always odd number. After stopping the proposed region growing algorithm, all frequencies of the grown region are considered as frequencies of the periodic noise. After detecting the periodic noises optimum notch filter removes the noise corrupted pixels.

CHAPTER 2

FIELD PROGRAMMABLE GATE ARRAY

Often, the cost, speed or power dissipation of a microprocessor may not meet system goals and alternative solution is required .A variety of programmable chips are available that can be more efficient than general purpose microprocessors yet faster to develop than dedicated chips:

- Chips with programmable logic arrays
- Chips with programmable interconnect
- Chips with reprogrammable logic and interconnect.

The system designer should be familiar with these options for two reasons

- First it allows the designer to competently asses a particular system requirement for an IC and recommend a solution, given system complexity ,the system operation cost goals time to market goals and any other tops level concerns

- Second it familiarizes the IC designer with methods of making any chip reprogrammable at the hardware level and hence more useful and of wider spread use.

2.1 Field Programming Gate arrays

Field-Programmable Gate Arrays (FPGAs) use the high circuit densities in modern processes to construct ICs that, as their name suggests, are completely programmable even after a product is shipped or “in the field”. Two basic versions exist.

- The first uses a special process option such as a fuse or antifuse to permanently program interconnect and personalize logic. These are one-time programmable.

- The second type uses static RAM or flash memory to configure routing and logic functions.

In general, an FPGA chip consists of an array of logic cells surrounded by programmable routing resources.

As an example of the first type of FPGA, devices manufactured by Actel embed an array of logic modules within an interconnect matrix that is formed on the top metal layers. Successive routing channels run vertically or horizontally. A special one-time programmable contact, called an *antifuse*, is placed at the intersection of routing traces. These normally have high resistance (effectively an open circuit). Upon application of a special programming voltage across the contact, the resistance permanently drops to a few ohms. CMOS switches allow the programming voltage to be directed to any antifuse in the chip. The advantage of this type of routing is that the size of the programmable interconnect is tiny:- the intersection area of two metal traces. Moreover, the on-resistance is low compared to a CMOS switch, so

the circuit speed is not compromised. The disadvantage is that the interconnect is not reprogrammable, so once a chip is programmed, its function is fixed to the extent that the interconnect has been personalized.

Figure 2.0 shows the floor plan of a simplified FPGA. The chip is composed of an array of *configurable logic blocks* (CLBs). Metal routing tracks run vertically and horizontally between the arrays of CLBs. These terminate at the gray blocks, which are routing switches that can be implemented using antifuses, CMOS transmission gates, or tri-state buffers. The routing resources can also be connected to the inputs and outputs of the adjacent CLBs. CLBs use programmable lookup tables to compute any function of several variables. Configurable I/O cells that can be used as input, output, or bidirectional pads surround the core array of CLBs.

A simple SRAM-based FPGA logic cell is shown in Figure 2.1. It is composed of a 16 × 1 static RAM as the logic element. This provides for any logic function of four variables merely by loading the RAM with the appropriate contents. Table 2.0 illustrates how the table should be loaded to perform various logic functions. A full adder can be implemented in two CLBs (one for carry and one for sum). The CLB shown also provides an optional output register. While it may seem inefficient or slow to use a RAM to perform logic, specially designed single-data line RAMs are small and fast in current processes, and resources such as the routing tend to dominate modern designs from a density and speed viewpoint. FPGAs have matured to the stage where they boast millions of logic gate equivalents supported by megabits of RAM. I/Os can operate in excess of 10 GHz. FPGAs frequently have embedded microprocessor cores and DSP accelerator hardware. Their low up-front cost and ease of correcting design errors makes them the best choice now for many low- to medium-volume custom logic applications.

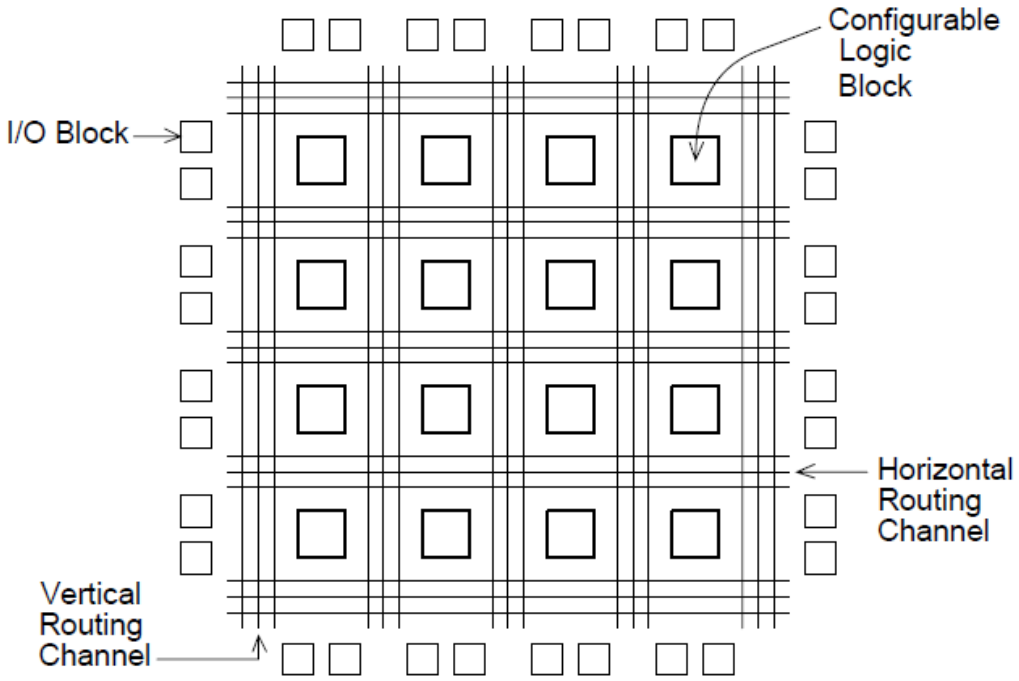


Fig 2.0

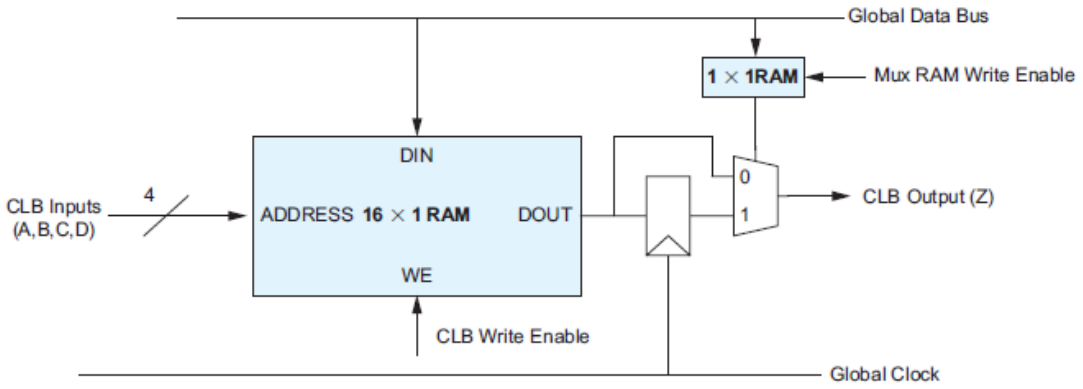


Fig 2.1

Address	*ABCD*	*A · B · C · D*	*~A*	SUM(*A,B,C*)
0	0000	0	1	0
1	1000	0	0	1
2	0100	0	1	1
3	1100	0	0	0
4	0010	0	1	1
5	1010	0	0	0
6	0110	0	1	0
7	1110	0	0	1
8	0001	0	1	0
9	1001	0	0	1
10	0101	0	1	1
11	1101	0	0	0
12	0011	0	1	1
13	1011	0	0	0
14	0111	0	1	0
15	1111	1	0	1

Table 2.0

Commercially Available FPGAs

This section provides a detailed description of three commercially available FPGA families, including those from Xilinx Co., Actel, and Altera. These particular FPGAs have been chosen because they are representative examples of state-of-the-art devices and they are in widespread use. Each device is described in terms of its general architecture, its choice of programmable cell, its routing architecture, and its CAD routing tools.

Xilinx FPGAs

There are three families of Xilinx FPGAs, called the XC2000, XC3000, and XC4000 corresponding to first, second, and third generation devices. Table 2.1 gives an indication of the logic capacities of each generation by showing the number of CLBs and an equivalent gate count. The gate count measure is given in terms of "equivalent to an MPGA of the same size." All FPGA manufacturers quote logic capacity by this measure, but it is questionable whether the figures quoted by each are realistic. The numbers given in Table 2.1, and in similar tables that appear later in this chapter, should be interpreted accordingly. The design of the Xilinx CLB and routing architecture differs for each generation, so they will each be described in turn.

Series	Number of CLBs	Equivalent Gates
XC2000	64 - 100	1200 - 1800
XC3000	64 - 320	2000 - 9000
XC4000	64 - 900	2000 - 20000

Table 2.1

2.1 Simplified block Diagram of Xilinx Spartan 3E FPGA

The CLB, illustrated in Figure 2.8, utilizes a hierarchical arrangement of look-up tables that yields a greater logic capacity per CLB than in its lower versions. The CLB can implement two independent functions of four variables, any single function of five variables, any function of four variables together with some functions of five variables, or some functions of up to nine variables. The CLB has two outputs, which may be either combinational or registered.

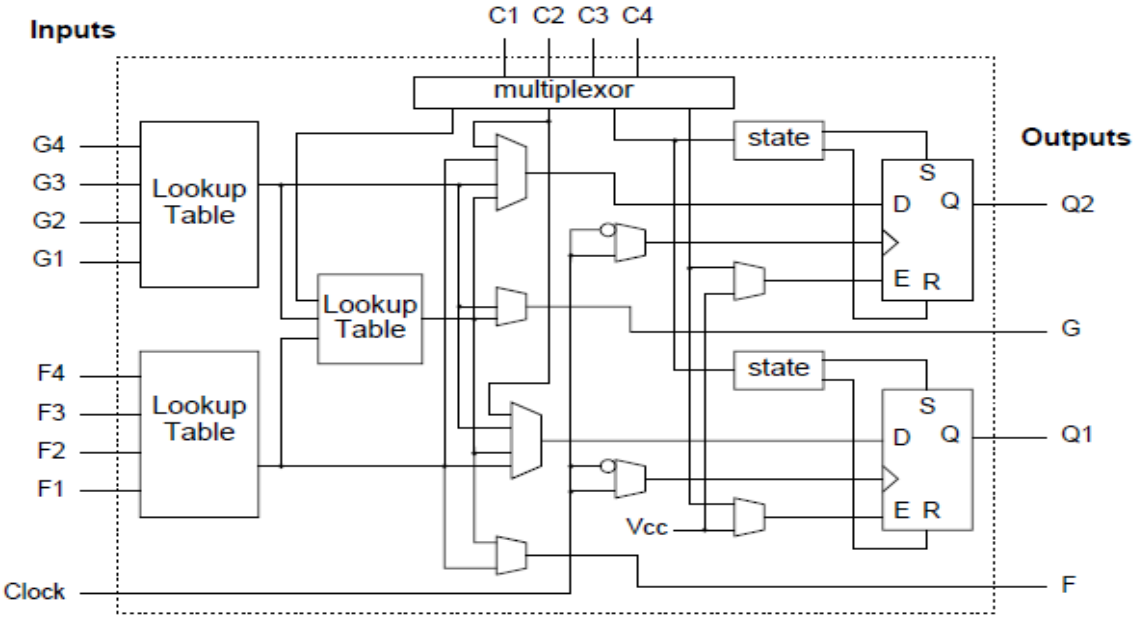


Fig 2.8 Simplified *CLB Architecture*

The routing architecture employs three types of routing resources: Direct interconnect, General Purpose interconnect, and Long Lines. The Direct interconnect provides connections from the output of a CLB to its right, top, and bottom neighbors. For connections that span more than one CLB, the General Purpose interconnect provides horizontal and vertical wiring segments, Each wiring segment spans only the length or width of one CLB, but longer wires can be formed because each switch matrix holds a number of routing switches that can interconnect the wiring segments on its four sides. Note that a connection routed with the General Purpose interconnect will incur significant routing delays because it must pass through a routing switch at each switch matrix. Connections are required to reach several CLBs with low skew can use the Long Lines, which traverse at most one routing switch to span the entire length or width of the FPGA. In the higher versions Direct interconnect and General Purpose interconnect lines are replaced by with two new resources, called Single-length Lines and Double-length Lines. The Single-length Lines are intended for relatively short connections or those that do not have critical timing requirements, are shown in Figure 2.9, where each X indicates a routing switch.

As the figure 2.10 shows, the Double length Lines are similar to the Single-length Lines, except that each one passes through half as many switch matrices. This scheme offers lower routing delays for moderately long connections that are not appropriate for the low-skew Long Lines. For clarity, neither the Single-length Lines nor the routing switches that connect to the CLB pins are shown in Figure 2.10.

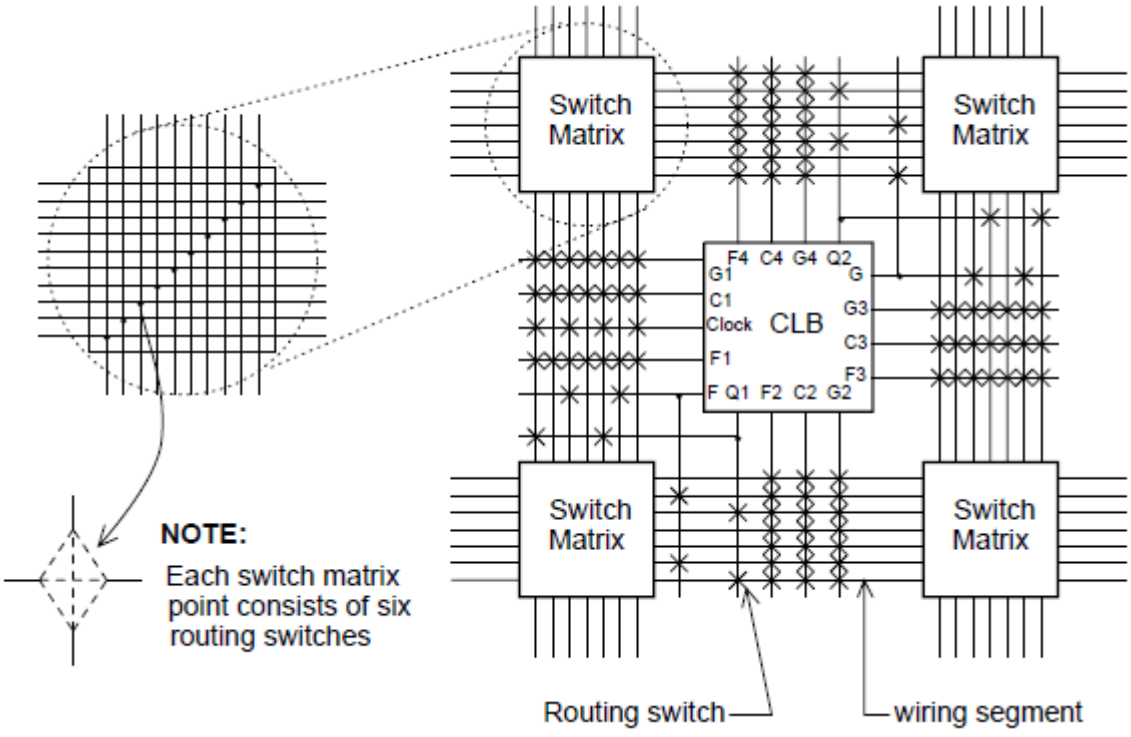


Fig 2.9 *Routing structure*

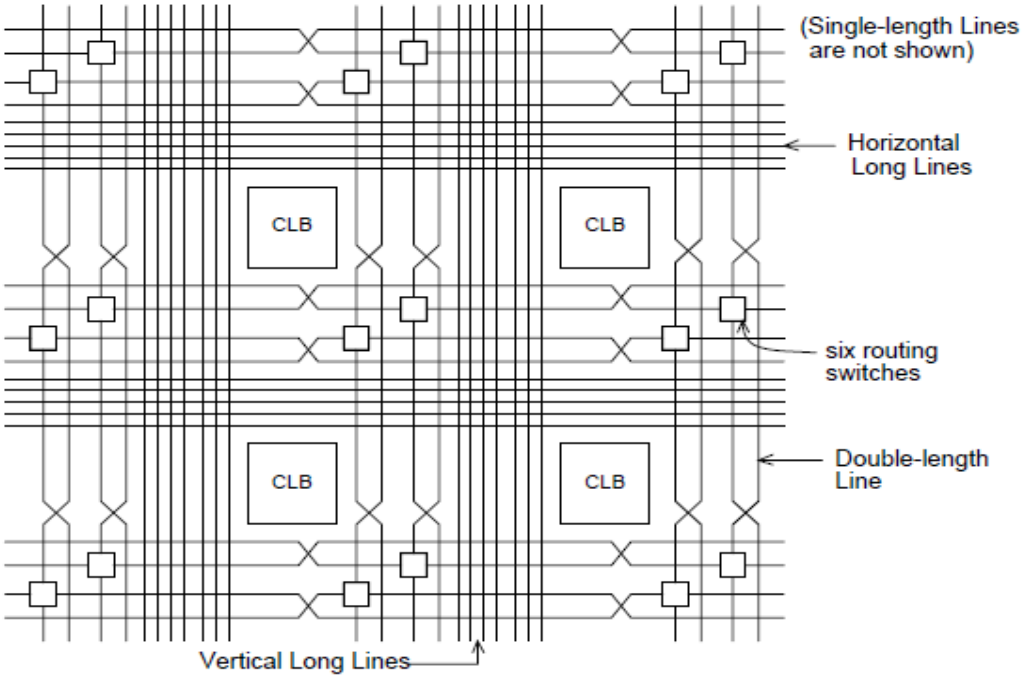


Fig 2.10 *double length line and long lines*

The architecture of input-output buffer is shown below:

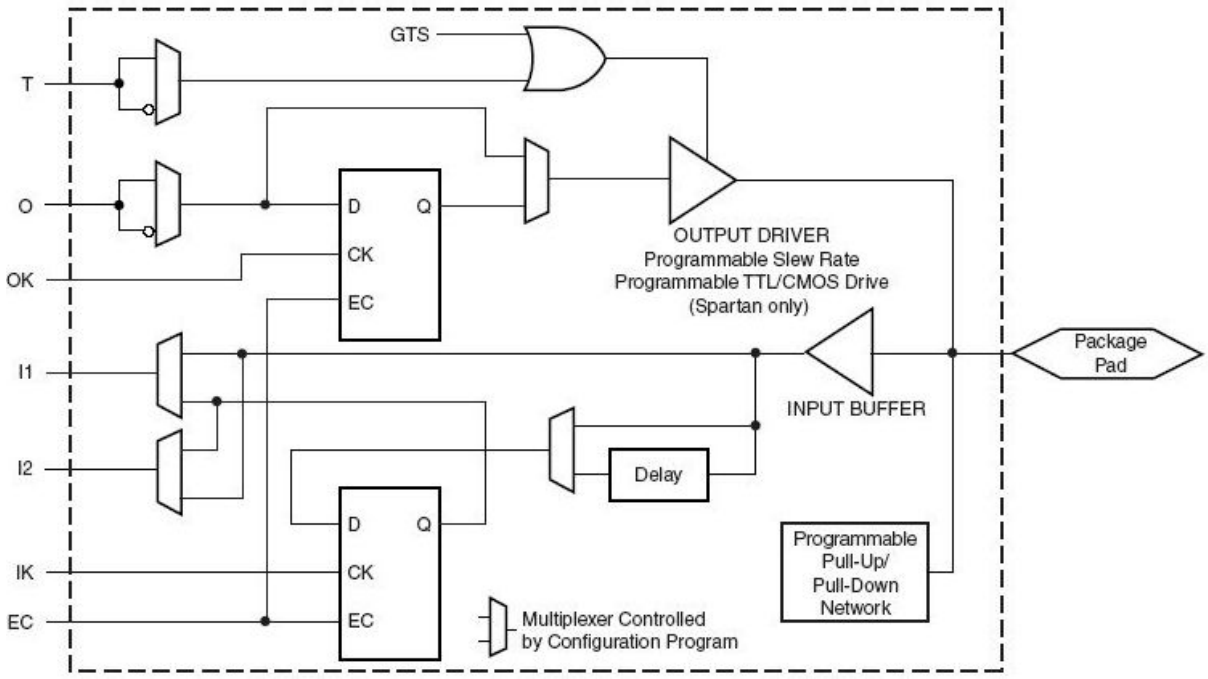
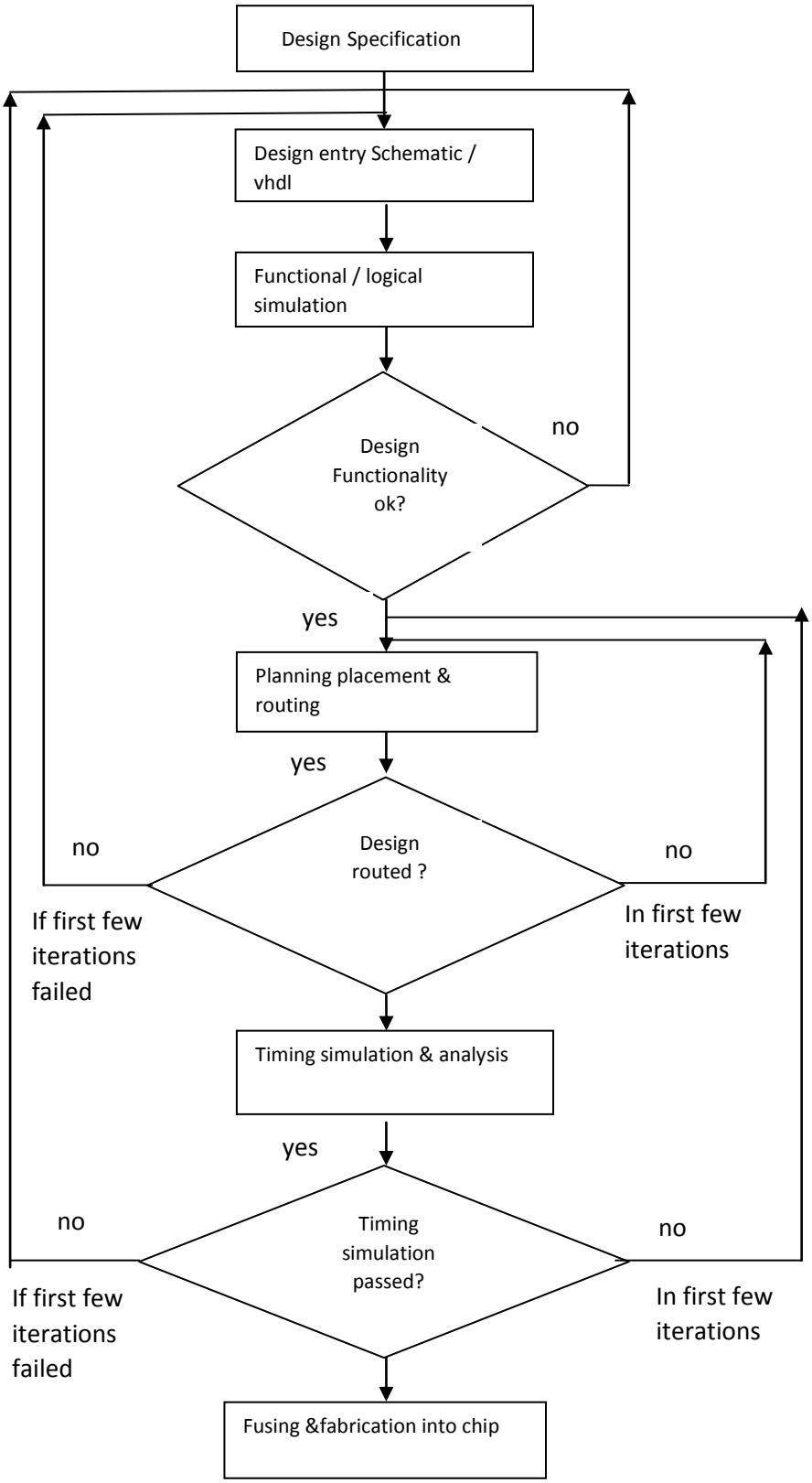


Fig 2.11 *Input Output Block*

2.2 VLSI design flow chart

It order to have a chip at its final packaged form, whether it is ASIC design or a FPGA design , several steps to be followed to have a fully functional chip .Whole VSLI design flow is represented in a flow chart below .At the very beginning in the design specification ,the algorithm must be known in its mathematical form .No of input-output pins in chip must calculated (optimized).Some of other constraints which limits our design must be known .These are maximum no of transistors to be used ,no of clocks to be used ,maximum clock speed to be used which determines the speed of the chip ,and power consumption requirement .Logic design can be done in either schematic or in VHDL language .Once logic design completed then functional simulation is done to verify the design .If design simulation is ok then the very next step is to go for planning ,placement and routing (PPR).Here logic blocks and wires are placed and routed in such a way to minimize delay .If the design is routed then go for actual timing simulation .If design passed the timing analysis then we must go for final step that is fusing and fabrication .But if not routed properly or timing analysis wrong then after some iterations go back to PPR step and again starts routing .Also if few iterations failed then go back to initial stage and check if there is any fault .Properly routed design is very much preferred ,several algorithms are there to rout a design ,generally place most connected blocks nearer .Local or global routing ,all are performed in simulation software .For digital design the software XILINX performs all the steps in a predefined manner .Everything is loaded here and can be performed and at the end design simulated in XILINX can be loaded to FPGA kit .Here we are using XILINX Spartan 3E series ,Device XC3S500E with package FT-256 .

Below the flow chart of VLSI design is shown.


Design Specification
Design entry Schematic / vhdl
Functional / logical simulation
Design Functionality ok?
no
yes
Planning placement & routing
yes
Design routed ?
no
If first few iterations failed
no
In first few iterations
Timing simulation & analysis
yes
Timing simulation passed?
no
If first few iterations failed
no
In first few iterations
Fusing &fabrication into chip

Chapter 3

Study and Design of Filter

3.1 Literature review

Median filter is a robust method to remove the impulsive noise from an image. It is a computationally intensive operation, so it is hard to implement it in real time. Median filter is a spatial filtering operation, so it uses a 2-D mask that is applied to each pixel in the input image. To apply the mask means to centre it in a pixel, evaluating the covered pixel bright nesses and determining which brightness value is the median value. The median value is determined by placing the pixels in ascending order and selecting the centre value. The obtained median value will be the value for that pixel in the output image.

Several algorithms to find median value have been implemented in FPGA for real time application .The algorithm [5] to find median value by sorting a matrix (3x3) of pixels by row wise followed by a column wise sorting, and then by sorting the diagonal elements, median value is obtained. This algorithm is implemented in [6] ,which employs a hybrid ,parallel serial input scheme .Here the hardware requirement is significantly less as compared to parallel design [7] with same speed whereas much faster than serial counterpart [7].

The median filter is also implemented by reconfigurable hardware in [8] for real time applications. It uses the UART interface with pc to receive image pixels from PC and RAM to store the pixels. Then median value is calculated .Simulated on XILINX XC3S500E of Spartan 3E, shows better performance than the offline implementation.

The classic approach for sorting 9 pixels and selecting the median, 41 basic nodes are necessary in this sorting network .Where every basic node finds max and min of two pixels .This approach has suffered several proposals of optimization, reducing the number of necessary nodes. Among them optimized sorting network presented by Morcego et al. [9] which composed of 27 basic nodes. Smith [10] shows a better way to find median value with very lower no of basic nodes (19 instead of 27) .Median filter implemented in [11] uses the median finding network very similar to the network proposed by Smith[10] .This scheme uses the minimum exchange network required to produce the median from nine pixels by performing a partial sorting. Here in this design 32 bit PCI interface bus which is there in the FPGA kit is used acquire image pixels that means four 8bit pixels per clock .and output is also 32bit word .By using the pipelining 4 median values are obtained per clock cycle.

Adaptive median filter is implemented for real time applications in [12] ,shows a fully pipelined operation. 24 pixels sorting network provides median of 5x5 or 3x3 neighbourhood pixels .If the pixels are corrupted then replace with median value of surrounding pixels .Compares between results obtained for 5x5 and 3x3 window mask .

The VLSI implementation of a selective median filter for the real-time applications is presented in [13] . The proposed design is based on a novel bit-level running algorithm with a modular and parallel in structure. This paper presents the VLSI implementation of a selective median filter with the high sample rate to support the real-time applications. According to the

proposed processing sequence, it is performed from the MSB to the LSB in parallelism and pipelined in the bit-level with regular data flow for the deletion and insertion.

A novel FPGA-based implementation of median and weighted median filters for image processing is described in [14] .Here principle of histogram based median filtering is used ,an alternative implementation of median filtering for arbitrarily large windows. The architecture is immune to changes in window size, the area being determined solely by the bit width. This allows for a flexible window-size that can change from one calculation to the next .For lower window size this design may not be faster but it has flexibility for higher window size.

Although the median filter is simple and provides a reasonable noise removal performance, it blurs image details and causes the useful information in the image to be lost. The most common solution to this problem is the switching median filter [15], which combines the median filter with an impulse detector. The impulse detector first determines whether a pixel in the image is corrupted or not. If the pixel is identified by the detector as a corrupted pixel, it is replaced with the output of the median filter. Otherwise, it is left unchanged. The proposed technique [16] is one of the switching median filters and implemented with an effective VLSI architecture. The Filter is composed of two sections –Impulse Detector and noise filter .Here 3x5 window is selected to confine the memory requirement in two line buffer .Impulse detector checks whether a pixel is corrupted or not by checking it with two adaptively calculated threshold values (Th_{max} ,Th_{min}) and sets a binary flag .In the noise filter section, 6 edges in different direction are considered (D_1 to D_6),calculated by taking absolute differences of pixels .If a pixel belong to a edge found corrupted by impulse detector then binary bit (b_1 t0 b_7)is set to indicate that the edge contains noisy pixels .If certain edge is found to have noisy pixel then value of that edge difference is replaced by 255 (maximum difference value).After tuning the different edges this way min edge direction is found (D_{min}).Then average of two pixels belong to D_{min} is calculated . Now if a pixel is corrupted by noise then it is replaced by that average value or it is remain same .Here noise is removed as well as edge preservation technique is used .This design proves better when the image is under light corruption. Furthermore, the method can detect edges in the diagonal, horizontal and vertical directions efficiently. The resulting design can achieve 150 MHz with only 7886 gate counts by using TSMC 0.18μm technology.

Simple Edge Preserved Denoising (SEPD) technique for fixed value impulse noise removal is implemented in [17]. It preserves edge features .A new stage, impulse arbiter is added here for preserving edges .The extreme data detector checks for pixels corrupted by impulse noise and sets a binary bit .If not corrupted then output remain same. By observing the spatial correlation, the edge-oriented noise filter pinpoints a directional edge and uses it to generate the estimated value of current pixel. To locate the edges twelve directional differences are defined (D1 to D12) for 3x3 mask. Only those composed of noise-free pixels are taken into account to avoid possible misdetection .If a pixel belong to a directional edge is corrupted which is known by checking a four bit flag B set by extreme data detector .Edges containing corrupted pixels are discarded .Here at a time four edge can be represented. If B = 1111 ,that means all four edges containing corrupted pixels ,so reconstructed pixels value will be a weighted average of previously denoised pixels .Otherwise Minimum valued edge direction is found out and average value of pixels belong to minimum edge is calculated. Impulse arbiter preserves the edge situated on a mask .If a pixel corrupted it will take extreme value but if the pixel has extreme value then it might not be a corrupted pixel ,it can be edge pixels .So to that difference between pixel value and its reconstructed value is calculated and if this difference value is compared with a threshold value .The reduced SEPD technique which

uses only 3 edge directions is also implemented which shows slightly poorer image quality at lower cost .Two line buffer (dual port rams) used instead of full frame buffer .

Edge Preserving Algorithm for Impulse Noise Removal which is implemented in [18] using the same algorithm and its implementation as described in [16] but shows a comparison between pipelined and without pipelined implementation .Pipelining fastens the operation compare to it's without pipeline counterpart but need more hardware.

Adaptive decision-tree-based denoising method (DTBDM) and its VLSI architecture for removing random-valued impulse noise described in [19]. DTBDM consists of two components: decision-tree-based impulse detector and edge-preserving image filter .As the impulse noise is not fixed rather random so a Decision tree based impulse detector is employed and this detector has 3 main blocks:– Isolation module, Fringe module and Similarity module to detect whether a pixel is noisy or not .As value of impulse noise pdf is distributed over 0 to 255 .The edge preserving filter uses the Adaptive edge preserving technique .

3.2 Filter Design

All the works Previously discussed are based on the hardware implementation of the spatial domain filters that mainly designed to remove impulse noise .Median filter and its variants (switching median filter, adaptive median filter, weighted median filter) is implemented in hardware for real time realization .But as we discussed in the previous chapter that some noises can't be removed in spatial domain .Just we have seen periodic noise which is occurred mainly due to electrical interference can't be removed In spatial domain ,so it must to go for frequency domain .Hardware realization of a real time denoising filter for periodic noise is not so much straightforward as it involves domain transform .Performing real time FFT operation is itself a tedious job. This is the base of our proposed work.

It is discussed in the first chapter that so many methods of filtering of periodic noise have been developed till date – Band reject filtering, Notch reject filtering, frequency domain mean filtering, Frequency domain median filtering, Optimum notch filtering, adaptive notch filtering and developing. Steps of frequency domain filtering is also discussed in 1st chapter .Only the filtering step is different but other steps are same for all the filters .So ,a basic filter is to realized first in hardware then go for the complex methods. We have chosen the basic filter to be frequency domain mean filter to realize in hardware implementation as its simplicity.

3.3 Hardware Implementation of Frequency Domain Mean Filter

Steps and its possible hardware realization are discussed below just to have a preliminary idea.

Fourier transform of the image is to obtained .2-D Fast Fourier transform of the 2-D image is to obtained by 1-D Fast Fourier transform operated row wise and column wise .And as performing FFT we gets real and imaginary data, so to work on we must take magnitude value. So, a absolute operation needed .we may go for serial FFT with CORDIC algorithm as

described in detail in [20] ,but speed is a serious issue in real time application so it's better to go for parallel FFT processor .Absolute operation can be done by CORDIC algorithm. Design of a parallel FFT processor is not of our concern here. For our convenience simple design of 8-point parallel FFT is considered.

DC component, that means the average value of the image should be shifted to the centre. So a shifting operation is done. Shifting operation is not so straight forward as it's not so simple .It can be done by delaying the pixels and masking at proper time but again not feasible for bigger image. If the pixels are stored in a RAM (Dual Port Ram is considered here for simultaneous write and reading) then shifting is achieved by reading pixels from the desired location by address pointed by suitable counter or writing pixels at desired location after FFT. RAM switching technique [20] is used here .During first phase half the image pixels are written and shifted in a RAM and during 2nd phase half the pixels are written and shifted in other RAMs .

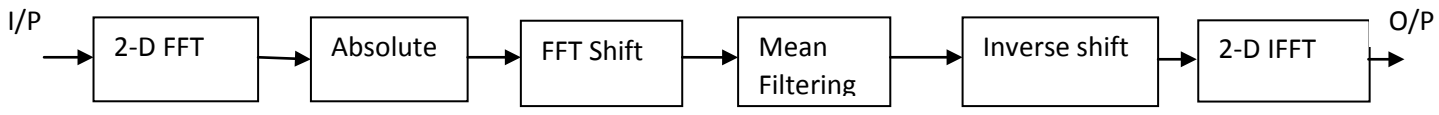
Filtering operation is done after shifting operation .For filtering operation small mask 3x3 is defined .So as the name suggests mean of the pixels excluding the pixels on which the mask is operating say f(x, y), of that window is calculated .So for 3 x 3, mean of 8 no of pixels .Now the pixel on which the mask is operating (f(x, y)) divided by mean. If the result is greater than some threshold value then f(x, y) is retained and if the value is greater than the threshold value then the reconstructed value of f(x, y) is f(x ,y) divided by 50 .This operation is to be carried out for whole image .

During first phase shifting operation is done on half of the pixels and during 2nd half simultaneously shifting operation and filtering is done on other half of pixels. Data exchange is done between the RAMs in order to maintain the pipelining operation. Now in respect of hardware design, mean can be calculated by adding pixels and divide by fixed value 8 ,which is achieved by passing the sum value to a 3 bit fixed right shift .A dedicated divider and comparator is needed .Divide by fixed value 50 can be achieved by a simple scale block ,constitute of various variable right shift blocks .For real time operation parallelism and pipelining should be introduced ,so to increase the speed of operation .Also with the limitation of cost and aim of increasing throughput with having certain latency period .

After filtering operation is done, inverse operations are to be carried out in order to reconstruct the image. So, inverse shifting operation is done by the same procedure.

Inverse FFT operation to be carried out next by same procedure as mentioned above. Here the multiplicative factor 1/MN, for an M x N image is the difference .This fixed multiplication again can achieve by scaling as mentioned previously.

So the block diagram of the process is-



3.4 Proposed work

Logical simulation of the above mentioned filter design part is verified but pipelining is to be introduced without increasing hardware count. As for example a dedicated divider is used in our design. A serial divider will not meet the design requirements as it is time consuming. So a parallel divider must be considered. Then the pipelined stages are introduced in the divider circuit. Processing time for each stage must be of a simple full adder/ subtractor delay time, These minimum time sets the design limitation. Like divider others blocks like scale block, mean block are to be realized by the same way. FFT processor design part is not our objective but to interface with the existing FFT processor is necessary without hampering the pipelining process. It is important to check for the total time, the design takes to restore the whole image. So soon after the design verification is done, timing simulation is to be done for the whole assembled design. Minimization of time will be our primary objective, as for real time application, operation needs to be faster than the software counterpart. We will try to achieve an online processing in our design. Finally, the design will be implemented on XILINX FPGA of SPARTAN3E device XC3S500E kit.

Conclusion

All the noise removal filters can be realized by software which is a offline procedure ,doesn't fulfil the requirements of real time application .All the software based implementations is slower than the online real time Hardware based implementations .By implementing those filters in FPGA ,speeds up the performance .Some of the constraints are to be made for real time designs. At lower cost higher speed must be achieved .So the FPGA implementations are preferred over other VLSI designs .Our goal will be to meet those constraints by adopting pipelining and parallel processing with least hardware counts.

Implementation of frequency domain mean filter is basis of realization of complex algorithms for periodic noise removal .So, frequency domain mean filter will first be realized then other complex filter which takes more time for its operation and shows better result than the others ,will be implemented. Various denoising parameters are to be verified in order to analyze design performance. Comparison of the parameters including time of the algorithms between its Matlab implementation and hardware implementation is needed.

References

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

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

3. 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.

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*

5. P. G. Poonacha ,Kapil Khanna “Real time median filter “,private communication .

6. “ FPGA implementation of Median Filter “-by RajuI Maheswari ,S S S Rao PG Poonacha an IEEE conference 1997

7. M Karaman ,L onural and A atalar “ design and implementation of a general purpopose median filter in vlsi “,vlsi signal processing ,III,IEEE press,pp.111-119,1988 .

8. Reconfigurable Hardware for Median Filtering for Image Processing Applications by Tripti Jain, Prashant Bansod, C. B. Singh Kushwah and Mayenk Mewara , Third International Conference on Emerging Trends in Engineering and Technology

9. B. Morcego, J. Frau, A. Català. “Suavizado de Imágenes en Tiempo Real mediante Filtrado por Mediana Utilizando Arrays Sistólicos”, *Proc. of VII DCIS*, pp. 545-546, (1992).

10. J.L. Smith. “Implementing Median Filters in XC4000E FPGAs”, *Xcell*, 23(4), pp. 16, (1996).

11. “ An FPGA-based implementation for median filter meeting the real-time requirements of autom1ated visual inspection systems ” by Miguel A. Vega-Rodríguez, Juan M. Sánchez-Pérez, Juan A. Gómez-Pulido

12. Real-Time Adaptive Image Impulse Noise Suppression by Ioannis Andreadis and Gerasimos Louverdis IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. 53, NO. 3, JUNE 2004

13. VLSI Implementation of Enhanced Edge Preserving Impulse Noise Removal Technique 2013 26th International Conference on VLSI Design and the 12th International Conference on Embedded Systems , P. Deepa, C. Vasanthanayaki

14. novel fpga-based implementation of median and weighted median filters for image processing by *Suhaib A. Fahmy*§, *Peter Y. K.Cheung*§ *and Wayne Luk*‡ an IEEE 2005 conference

15. T. Sun and Y. Neuvo, "Detail-preserving median based filters in image processing," *Pattern Recognit. Lett.*, vol. 15, pp. 341-347, Apr. 1994.

16. " A Real-time Image Denoising Chip " by Pei-Yin Chen, Chih-Yuan Lien, and Yi-Ming Lin 2008 IEEE

17. 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

18. Edge Preserving Algorithm for Impulse Noise Removal using FPGA by S.Jayanthi Sree ,S.Ashwin, S.Aravind Kumar, a IEEE 2012 conference .

19. An Efficient Denoising Architecture for Removal of Impulse Noise in Images by Chih Yuan Lien, Chien-Chuan Huang, Pei-Yin Chen, Member, IEEE, and Yi-Fan Lin ,an IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 4, APRIL 2013

20. " FPGA realization of a CORDIC based FFT processor for biomedical signal processing " by Ayan Banerjee Anindya Sundar Dhar ,Swapna Banerjee ,an ieee journal .