FPGA Implementation of Power & Area Efficient Bilateral Filter for Image De-Noising

Sreeja Sivan
ME Applied Electronics
The Kavery Engineering College

Mr. Sivakumar ,ME
Assistant Professor/ECE
The Kavery Engineering College

***Abstract*-The Bilateral filters are very popular for denoising in image processing, because it reduces noise while preserving details. This work studies concept of the processing of entire filter window at one clock cycle. By the exploitation of the separability and the symmetry of one filter component, the complexity of the design is widely reduced. Thus the Bilateral filter is implemented as a highly parallelised pipeline structure with very economic and effective utilization of dedicated resources. Due the modularity of the filter design, kernels of different sizes can be implemented with our design and given instructions for scaling. The major contribution of this paper is the detailed description of a novel FPGA design architecture of the bilateral filter on register-transfer level (RTL). Most of the early works rely on GPUs for hardware acceleration. However, in fields of applications in which high power efficiency is crucial, an FPGA solution is preferable. For area and power efficiency, the proposed system is implemented with a modified radix-8 booth multiplier and implemented in low cost Sparten3 FPGA Platform.**

INTRODUCTION

Image processing is widely used in many fields, such as medical imaging, scanning techniques, printing skills, face recognition, and so on. In general, images are often corrupted by impulse noise in the procedures of image acquisition and transmission. The noise may seriously affect the performance of image processing techniques. Hence, an efficient denoising technique becomes a very important issue in image processing. According to the distribution of noisy pixel values, impulse noise can be classified into two categories: fixed valued impulse noise and random-valued impulse noise. The former is also known as salt-and-pepper noise because the pixel value of a noisy pixel is either minimum or maximum value in grayscale images. Image processing is a physical process used to convert an image signal into a physical image. The image signal can be either digital or analog. The actual output itself can be an actual physical image or the characteristics of an image. The most common type of image processing is photography. In this process, an image is captured or scans using a camera to create a digital or analog image. In order to produce a physical picture, the image is processed using the appropriate technology based on the input source type. In digital photography, the image is stored as a computer file. This file is translated using photographic software to generate an actual image.

The colors, shading, and nuances are all captured at the time the photograph is taken the software translates this information into an image. When creating images using analog photography, the image is burned into a film using a chemical reaction triggered by controlled exposure to light. The field of digital imaging has created a whole range of new applications and tools that were previously impossible. Face recognition software, medical image processing and remote sensing are all possible due to the development of digital image processing. Specialized computer programs are used to enhance and correct images.

EXISTING SYSTEM

The bilateral filter embodies the idea of a combination of domain and range filtering. The domain filter averages the nearby pixel values and acts thereby as a low-pass filter. The range filter stands for the nonlinear component and plays an important part in edge preserving. This component allows averaging of similar pixel values only, regardless of their position in the filter window. If the value of a pixel in the filter window diverges from the value of the pixel being filtered by a certain amount, the pixel is skipped.

Design concept of Bilateral Filtering

The bilateral filter can be realized as a highly parallelized pipeline structure giving great importance to the effective resource utilization. The design concept for the implementation of the bilateral filter is subdivided into three functional blocks. .

Figure shown below presents these units and their order in the concept. The input data marked by “Data_in” are read line by line and arranged for further processing in the register matrix. The second unit is the photometric filter which weights the input data according to the intensity of the processed pixels. The filtering is completed by the geometric filter, and the filtered data are marked by “Data_out.”

Figure. 3.1 Order of the functional units of the bilateral filter

PROPOSED SYSTEM

Register matrix

The photometric filter component, also often referred to as a range filter , is a nonlinear filter. It means that the filter coefficients change for every filter position. The filter window is shifted first along the input lines representing

the image rows, moving one row down every time the precedent row has been filtered. At least five lines have to be stored for the period of time during which a line is filtered. The five input lines are called image rows These five rows include the row to be filtered, two foregoing rows, and two succeeding rows.

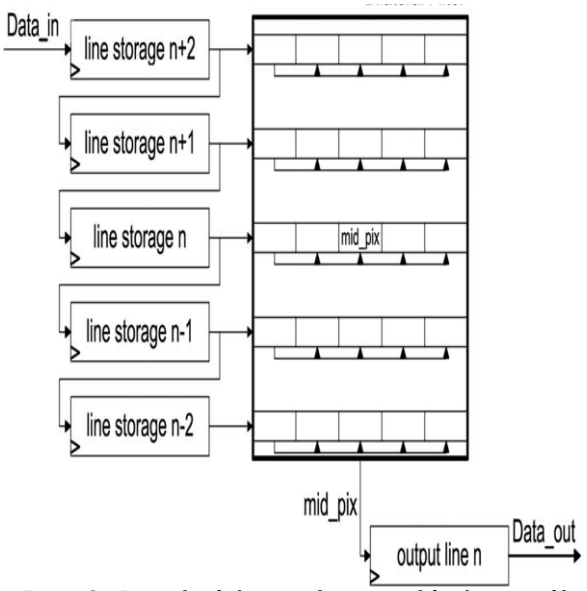

Figure. 3.2 Principle of the input data retrieval for the image filtering.

The pixel being filtered is marked by "mid_pix." This pixel and its neighbourhood in the solid box represent the kernel of the bilateral filter. After the middle row has been filtered, the outer foregoing row "line storage n-2" moves out of the register matrix. As the input data are read into the register matrix pixel by pixel, the content of the line storages and of the filter kernel is shifted by one pixel at each clock event. The single registers are interconnected in a manner that, aside from the shift of the filter window by one pixel, the entire kernel is provided to the next filter stage simultaneously.

This is an important advantage of the presented kernel-based design concept as no extra data buffer is required. The output of the register matrix is sorted into groups, in this case into six groups, and fed into the photometric filter component with the quadruple pixel clock frequency synchronously. . The quadruplication of the filter processing clock is implemented by setting the select signal of the multiplexers four times in one pixel clock.

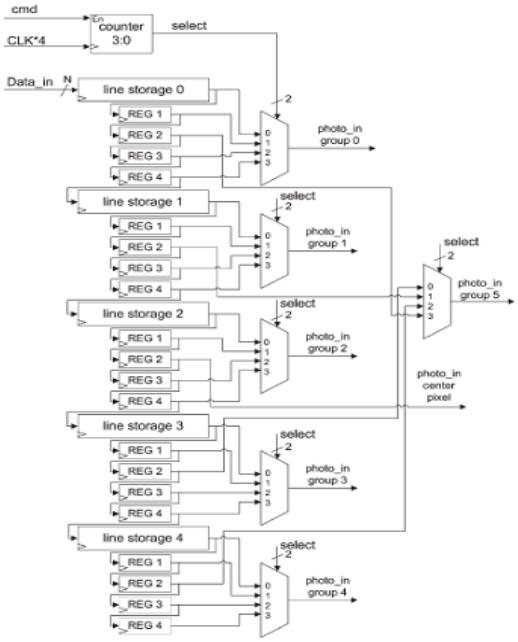

Figure. 3.3 Register matrix of kernel-based design concept

Photometric Component

After the register matrix has been filled, the grouped image data are provided to the photometric filter component. The output of the photometric filter consists of the following:

1) weighted pixels sorted into groups 0 . . . 5;
2) the weighted pixel being filtered, marked by "mid_pix";
3) photometric coefficients corresponding to groups 0 . . . 5.

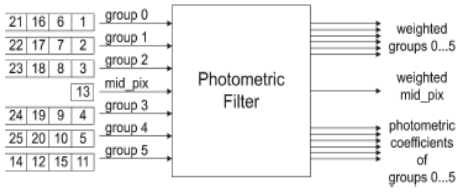

Figure. 3.4 Abstract illustration of the photometric filter component.

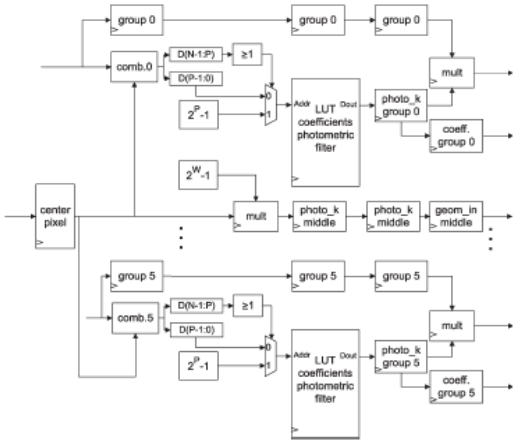

Figure 3.5 Photometric filter component.

A detailed functional flow block diagram of the photometric filter is shown above. The pixel in the center of the filter window has to be available during the calculation of the required 24 pixel weights. Latching the centered pixel allows the computation of the gray value differences between the centered pixel and the remaining pixels inside of the filter window. Each group contains four pixels. A separate pipeline belonging to each group makes it possible to process the entire neighborhood of "mid_pix" at one pixel clock signal. All six pipelines are designed identically.

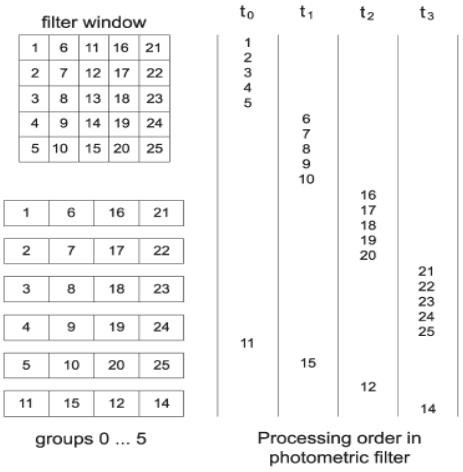

Figure.3.6 Processing order of input data in the photometric filter component.

The way of arranging and the processing order of the input data of the photometric component are shown in figure above. At the first internal clock event t0, the first pixels of each group are provided to the respective pipeline. At the second internal clock t1, the second pixels of each group enter the component. This organization of groups allows the processing of the whole filter window in four internal clock cycles corresponding to one pixel cycle.

The combinatory blocks "comb.0 . . . 5" compute the absolute gray value difference. To avoid the calculation of the expensive exponential, all possible values of the function (2) are precalculated and stored in the lookup table (LUT). The pixel in the center of the filter window does not belong to any group and is processed separately. This pixel is multiplied by the highest coefficient $2W - 1$ and delayed by registers "photo_k middle" and "geom_in middle" for synchronicity.

Geometric Component

Because of the separability, the geometric filter is split into the vertical and horizontal parts. Therefore, 2-D filtering is replaced by successive 1-D filtering in vertical and horizontal directions. This solution is preferred in the design of the geometric filter because 1-D filtering can be implemented more efficiently. Both parts are implemented twice to filter the weighted image data and the photometric weights simultaneously.

The input of the vertical component parts is the 2-D array of the filter window and the 2-D array of the corresponding coefficients. Each output is a 1-D vector in which each entry represents one filtered and cumulated column. The output of the geometric filter consists of the filtered unnormalized gray value (kernel result) and the normalization factor. Smaller groups allow for better handling of the design. For this reason, the pixels are divided into groups of four . After the accumulation of the pixels according to their symmetry, the sum is multiplied by the corresponding coefficient. The horizontal processing is done in the same way.

1) Vertical Component Part: The first stage of the geometric component is the vertical part which is picture below. The pixels of the first column numbered 1, 2, 3, 4, 5 and the first pixel of the middle column numbered 11 enter the vertical component part simultaneously. The geometrically symmetrical pixels are cumulated at first and then multiplied by the geometric weight coefficient. All coefficients for the geometric filter are constant for the chosen filter window size. After the multiplication, the weighted values are summed up by the adder tree to one value at each internal clock event.

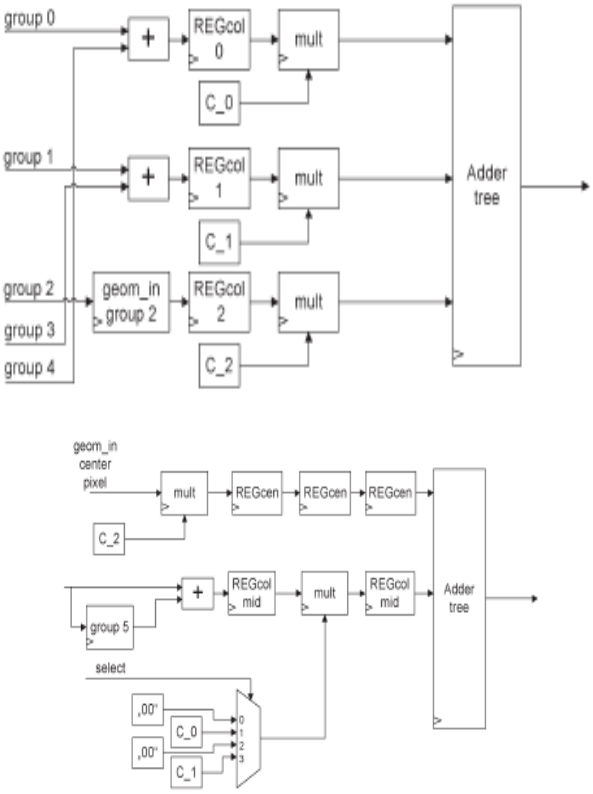

Figure. 3.7 Vertical part of geometric filter component.

The centered pixel is weighted and delayed by "REGcen" so that this pixel and the remaining pixels in the centered column can be fed to the input of the adder tree simultaneously. The remaining pixels enter the dedicated processing path one by one. They were multiplexed in the register matrix in the way that they can be combined pairwise and multiplied by the same coefficient in the geometric component. In order to weight the pixels in a proper way, every incoming pixel is stored in the register

"REGcol mid" so that the subsequently calculated sum is valid every second internal clock event.

2) Horizontal Component Part: In Fig below, the horizontal part of the geometric component is displayed. After processing in the vertical dimension, the filter window is reduced to one row, and its elements are computed at one internal clock event each. In order to be able to reuse the symmetrical design, the values of the filtered columns 0, 1, 3, 4 are stored in the shift registers according to the order of their reception.

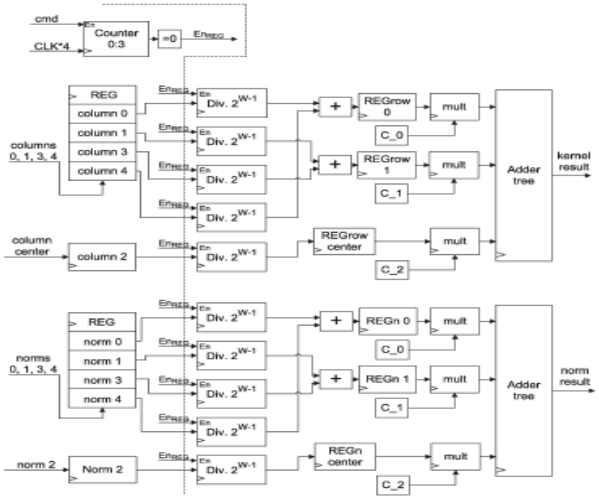

Figure.3.8 Horizontal part of geometric filter component
At every pixel clock signa

l, the valid column values are written to the registers which perform the division of the weighted gray values by the normalized ones. The division is implemented through a shift operation. The remaining processing is similar to the processing described in the previous paragraph. The geometrically symmetrical pixels are cumulated at first and multiplied afterward by the geometric weight coefficient. The system can be implemented in a low cost hardware platform namely Spartan 3 FPGA kit. The multiplier circuit in the geometric filter section can be modified with a radix-8 booth multiplier to facilitate the area and energy efficiency of the system.

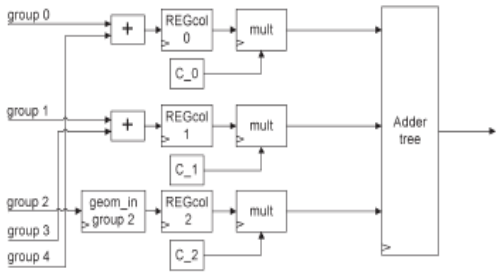

Figure. 3.9 Geometric filter component

A binary multiplier is an electronic circuit used in digital electronics, such as a computer, to multiply two binary numbers. The multiplier is the basic key component of any digital signal processing system and these are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors and multimedia applications. The type of the multiplier used for an application is based upon the requirements of the application.

Modified Booth Multiplier

In Modified Booth, the number of partial products reduced by N/2, that is half of total partial products as compare to simple multiplication process. So, clearly if the number of partial products become reduced, the area of the multiplier also will reduced and automatically as the result of it, the speed will increased. So, this multiplier is more efficient.

Modified Booth Algorithm

The Modified Booth algorithm is the most frequently used method to generate partial products. The partial products are reduced by N/2 by using this algorithm. So as the result of this, the multiplier can be implemented using less hardware components as compare to conventional multiplier. This algorithm can save multiplier layout area and reduces delay at the same time which are the important design advantages.

One of the method for high speed multiplier is to enhance the parallelism by reducing the number of calculating stages .Booths encoding reduces partial products to N/2. It converts the multiplier from radix-2 to radix-4 using redundant digit set {-2, -1, 0, 1, 2}. So in new multiplier with radix-4 there are only N/2 digits. New multiplier's 'P' digits are defined by following formula: Pj = Y2j + Y2j-1 -2Y2j+1 with Y-1 = 0

The Modified Booth Algorithm takes following steps for multiplication:

- Modified Booth Encoder.
- Partial Product Generator.
- Sign Extension

Modified Booth Encoder (MBE)

Modified Booth encoding is most often used to avoid variable size partial product arrays. Before designing a MBE, the multiplier B has to be converted into a Radix-4 number by dividing them into three digits respectively according to Booth Encoder Table given afterwards. Prior to convert the multiplier, a zero is appended into the Least Significant Bit (LSB) of the multiplier. approach Instead of eight partial products being generated using conventional multiplier. Table 1. shows the truth table for a Booth encoder. The encoder takes inputs YN+1, YN and YN-1 from the multiplier bus and produces a 1 or a 0 for each operation: single, double, negative.

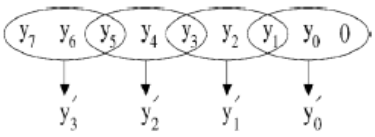

Figure 3.10. Grouping of bits for MBE scheme

This Booth multiplier technique is to increase speed by reducing the number of partial products by half. The

operand that is booth encoded is called multiplier, and the other operand is called multiplicand.

Table 3.1 -Truth Table For Booth Encoder

Y_{N+1}	Y_N	Y_{N-1}	Operation	X	2X	Neg
0	0	0	+0 x X	0	0	0
0	0	1	+1 x X	1	0	0
0	1	0	+1 x X	1	0	0
0	1	1	+2 x X	0	1	0
1	0	0	-2 x X	0	1	1
1	0	1	-1 x X	1	0	1
1	1	0	-1 x X	1	0	1
1	1	1	-0 x X	0	0	1

In most of the cases MBE scheme is used for generating PP, because of its ability to reduce the number of PP by half[7]. The truth table shows the function of booth encoder. If a 3-bit binary input sequence is given at the input, and perform the operation as mentioned infront of it, the partial products will be generated.

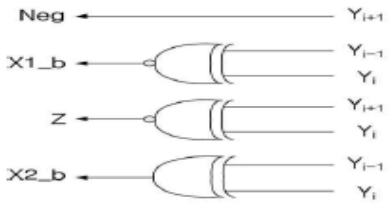

Figure 3.11. Booth Encoder

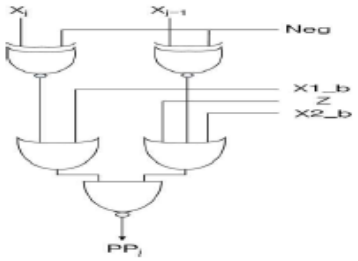

Figure 3.12. Booth Decoder

Partial Product Generator (PPG)

Partial product generator is the combination circuit of the product generator. Product generator is designed to produce the product by multiplying the multiplicand X by 0, 1, -1, 2 or -2. For product generator, multiply by zero means the multiplicand is multiplied by "0". Multiply by "1" means the product still remains the same as the multiplicand value. Multiply by "-1" means that the product is the two"s complement form of the number. Multiply by "-2" is to shift left one bit the two"s complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by one place.

Sign Extension Corrector

X7	Y_{N+1}	Y_N	Y_{N-1}	E
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0

Table 3.2 Truth Table for Sign Extension when X7 is zero.

Sign Extension Corrector is designed to enhance the ability of the booth multiplier to multiply not only the unsigned number but as well as the signed number. As shown in Table 3.2 when bit 7 of the multiplicand X(X7) is zero(unsigned number) and Yn+1 is equal to one, then sign E will have one value (become signed number for resulted partial product).

X7	Y_{N+1}	Y_N	Y_{N-1}	E
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Table 3.3 Truth Table for Sign Extension when X7 is one.

However when both the value of A7 and Yn+1 are equal either to zero or one, the sign E will have a value zero(unsigned number). For the case when all three bits of the multiplier value Yn+1, Yn and Yn-1 are equal to zero or one, the sign E will direct have a zero value independent to the X7 value.

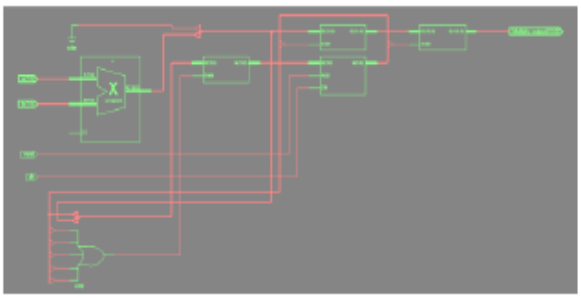

Figure 3.13 RTL schematic of 8 x 8 bit multiplier

Performance Analysis

The comparison of the filtering capability between the Matlab implementation and the ModelSim simulation. Between the Matlab implementation and the ModelSim simulation, no visually distinguishable difference can be registered.

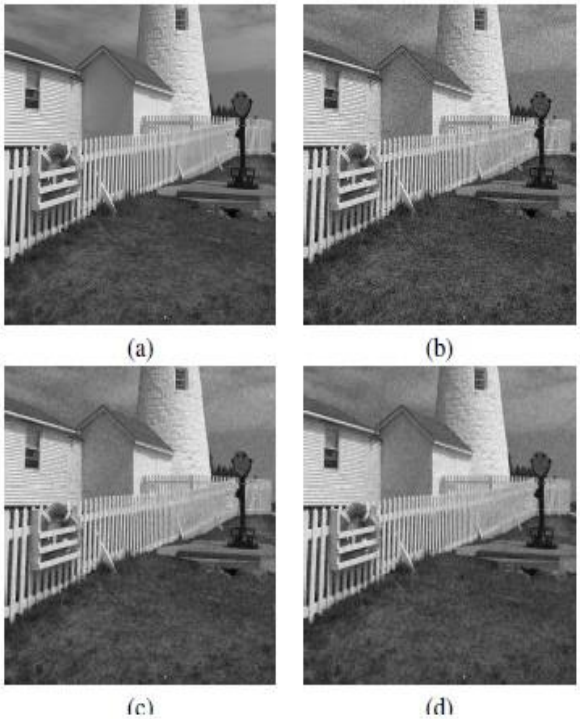

Figure .3.14 (a) Original image. (b) Noisy image. (c) Filtering in Matlab. (d) Filtering in ModelSim.

INTRODUCTION ABOUT SYSTEM SOFTWARE

MATLAB (The Language of Technical Computing)

MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java. We can use MATLAB for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers and scientists in industry and academia use MATLAB, the language of technical computing.

CONCLUSION

In this paper, we have given a detailed description of an FPGA design of the bilateral filter for real-time image processing. The advantages of our design can be summarized in following points.

1) The filter design for a kernel size of 5 × 5 shown here utilizes the FPGA resources economically, which makes it feasible to implement the filter on a common mediumsized FPGA.
2) The introduced register matrix at the first stage of the filter makes external image storage redundant, contributing to the decrease of the resource demand of the filter implementation.
3) The shown architecture is synchronous and capable of real-time processing supporting high clock frequencies. Maximal operating frequency depends on the chosen FPGA family.
4) Conceiving our filter architecture, we kept in mind the scalability of the design in order to enable the implementation of arbitrary filter window size with low effort.
5) The shown filter architecture assures a constant processing delay independent of the filter window size. The total delay is the sum of the processing delay and the fill-up time of the line storages which depends on the kernel size and image width.

3.4 Future Scope for the Project

The proposed work will be simulated using Modelsim and Matlab, synthesis by using Xilinx Software and implemented using Spartan3 FPGA platform. I expect the power and area will reduce, considerably.

REFERENCES

1. Anna Gabiger-Rose , Matthias Kube, Robert Weigel and Richard Rose, “An FPGA-Based Fully Synchronized Design of a Bilateral Filter for Real-Time Image Denoising”
2. C. Tomasi and P. Manduchi, “Bilateral filtering for gray and color images,” in *Proc. IEEE ICCV*, 1998, pp. 839–846.
3. B. Zhang and J. P. Allebach, “Adaptive bilateral filter for sharpness enhancement and noise removal,” IEEE Trans. Image Process., vol. 17, no. 5, pp. 664–678, May 2008.
4. M. M. Bronstein, “Lazy sliding window implementation of the bilateral filter on parallel architectures,” *IEEE Trans. Image Process.*, vol. 20, no. 6, pp. 1751–1756, Jun. 2011.
5. F. Hannig, M. Schmid, J. Teich, and H. Hornegger, “A deeply pipelined and parallel architecture for denoising medical images,” in *Proc. IEEE FPT*, 2010, pp. 485–490.
6. A. Gabiger, M. Kube, and R. Weigel, “A synchronous FPGA design of a bilateral filter for image processing,” in Proc. IEEE IECON, 2009, pp. 1990–1995.