Project Report

FPGA Implementation of Image Filter for Denoising and Smoothing

Kurupati Sai Pruthvi Teja

Contents

List of Figures

Introduction

Problem Statement: To implement an image filter on a FPGA that can perform filtering operations on 32x32 8-bit gray scale image.

Image Filtering: Image filtering is technique in which an image is modified or enhanced to highlight or remove certain features. Some types of filtering techniques are smoothing, denoising, edge detection, sharpening etc. There are two types of filtering methods namely spatial filtering and frequency domain filtering.

In this project we have implemented two spatial filters, Gaussian filter, and Median Filter. These both filters come under the category of neighborhood pixel filtering as the filtered pixel vale depends on the current pixel and its neighborhood pixel values. Generally spatial filtering is 2D convolution of given image with a filter specific kernel. In this project the filters are implemented such that the filtered pixel value depends only on the nearest neighborhood values.

Gaussian Filtering:

- Gaussian filter is non uniform low pass filter in which kernel coefficients decreases when we farther away from the kernel's center.
- The way pixel values decreases are similar to gaussian function hence it is called as Gaussian filter.
- The kernel can have any dimension starting from 3x3. In this project we have used 3x3 kernel.
- Gaussian kernel of 3x3 order is $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
- This filter is used for smoothing of an image.

Median Filtering:

- Median filtering is also a neighborhood type filtering in which the filtered pixel is median of the current pixel and eight neighborhood pixels. It is nonlinear method filtering for smoothing.
- Since we are finding median for given pixel values median filter does not use convolution process like the gaussian filter.
- While implementing this filter also we have used only nearest neighborhood pixels.
- While implementing median filter initially the pixels values need to be sorted and the central value is the median.
- This filter is used for denoising the image which has some random noise.

Architecture

Specifications:

- The image filter can perform either gaussian filtering or median filtering on a given image.
- The gaussian filtering involves multiplication and addition operations and finally average according to the kernel values.
- The median filtering involves sorting of given inputs and middle value is taken as output.
- Gaussian filtering is used for the smoothing the image and Median filter is used for denoising the image. The type of filtering can be selected by a user input.
- The input image is of gray scale and each pixel value is encoded as 8-bit.
- The design is powered by 100MHz clock.

Top Level Architecture:

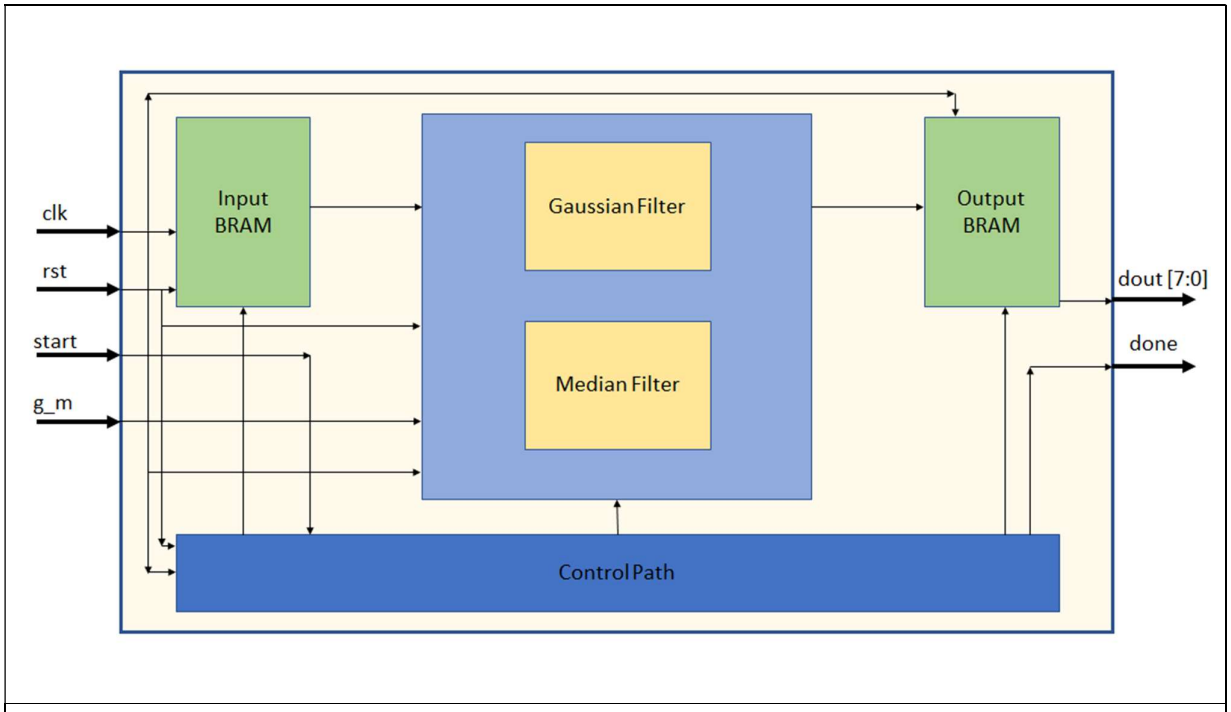

Figure 1. Top Level Architecture of the Design

- The Input BRAM is a block RAM in which input image is stored. The width is 8-bit and depth is 1156. The input image is padded with zeros around it so that filtering operation can also be performed at the edge pixel positions. So, the dimension of image becomes 34x34 i.e., 1156-pixel values.

- The Output BRAM is a block RAM in which filtered image is stored. The width is 8-bit and depth is 1156. The depth is kept same as input block ram so that addressing is in a synchronous manner.
- The Gaussian Filter and Median Filter combinedly forms image filtering block. The type of filtering can be selected by a user input g_m pin.
- The control path gives control signals for the operations and controls addressing of block RAM during operations.

Input and Output Ports:

- clk – Global clock of frequency 100MHz.
- rst – Reset
- start – starting of filtering operation on image.
- g_m – selection of filtering type. 1 for median and 0 for gaussian.
- done – signal that indicates filtering operation is completed.
- dout – filtered pixel values that can be used for external display.

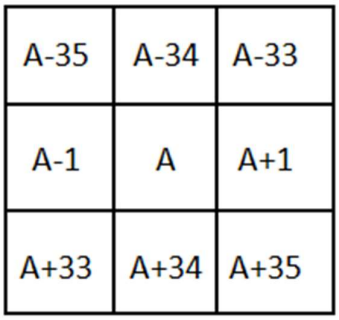

Figure 2. Neighborhood addressing of pixel

Since we are performing operations on 3x3 pixel values, and the pixel values are stored in sequential manner in the BRAM. The addressing is done according to calculated address values that are shown in figure 2. The address decoding logic block implements this function.

Gaussian Filter:

Figure 3 shows the architecture of the gaussian filtering block. It takes the address values from the top module. This address is used to decode the neighborhood pixel address, using this address the pixel value is fetched from BRAM. The address decoding block is shown in figure 4. The data that is fetched from the BRAM has 2 clock cycles

delay. So, a delay block is used for synchronized operations. This block is shown in figure 5.

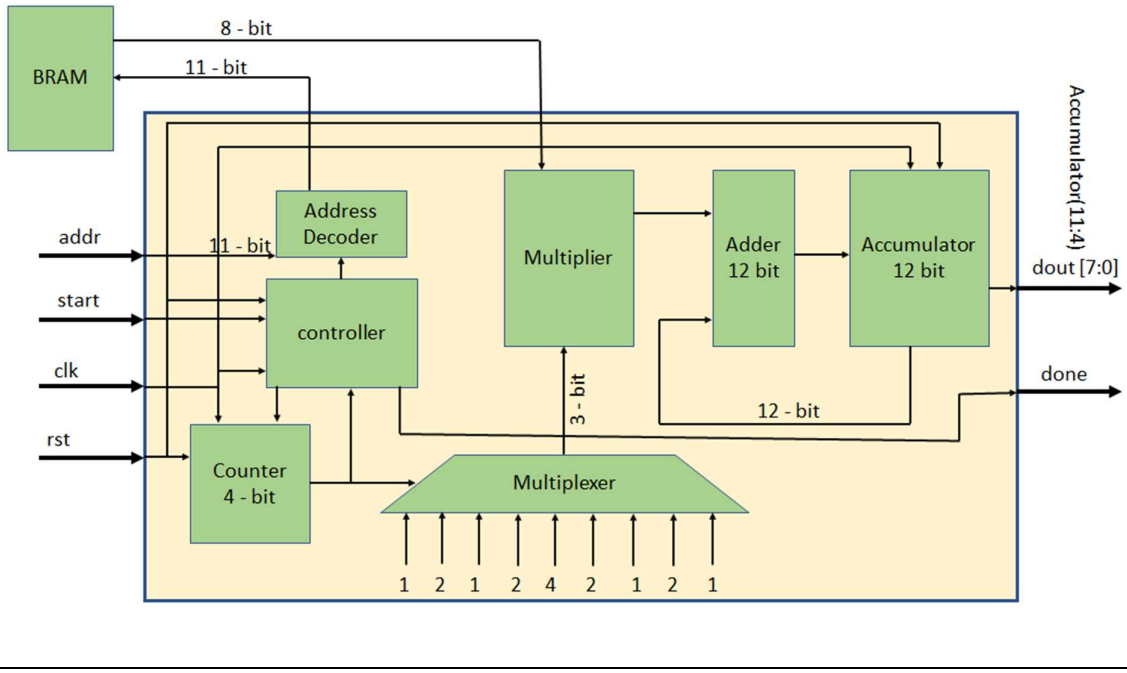

Figure 3. Architecture of Gaussian Filter

The filtered pixel from the gaussian is stored in another BRAM for which the address input is same as the address input of gaussian filter block.

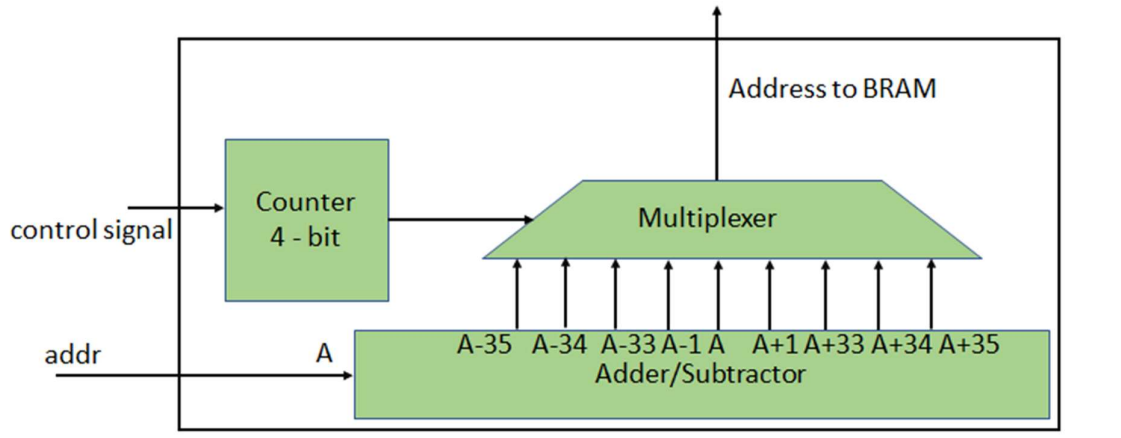

Figure 4. Address Decoder

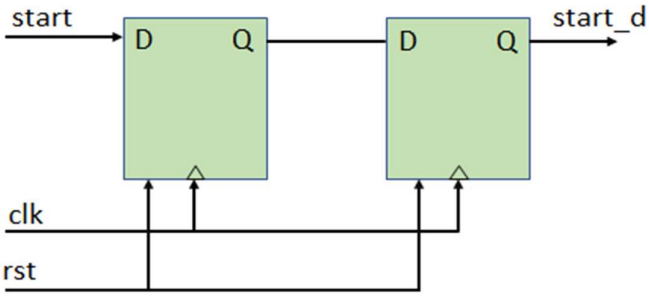

Figure 5. Delay Circuit

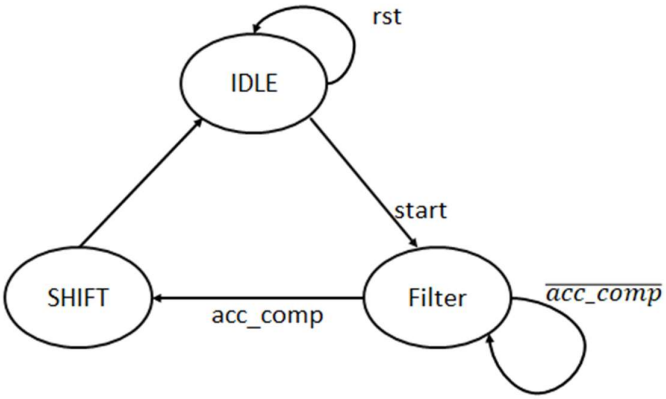

Figure 6. FSM for Addressing in Gaussian Filter

The above figure shows the state diagram for the addressing operation in gaussian filter. When reset is applied the machine will be in IDLE state, when start is applied it goes to FILTER state. In FILTER state when acc_comp goes high it goes in SHIFT state. When start signal arrives a 4-bit counter starts running from zero. The counter output is taken as the select lines for the address decoding multiplexer. As we need nine-pixel values after 9 counts of the counter acc_comp signal is made high, and we go into next state where the addressing stops. The input address is given to a block in which required pixel values is calculated by adding or subtracting a specific value that are shown in the figure 2.

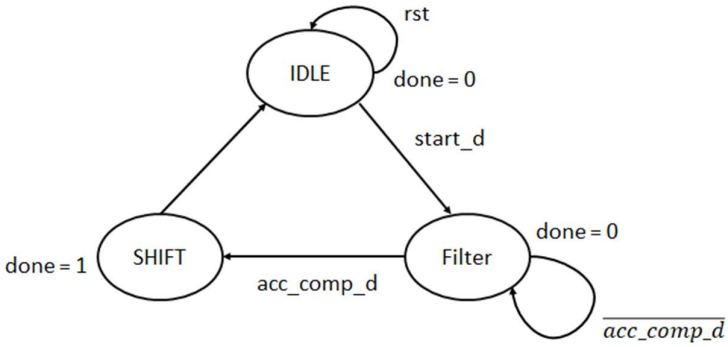

Figure 7. FSM for Gaussian Filtering Operation

The above figure shows the state diagram for the convolution operation in gaussian filter. It is same as addressing state machine but here start_d signal is 2 cycles delayed version of start signal as the data fetch from BRAM takes two cycles. During FILTER state MAC operation is done with selected weigths. This machine has done output, it goes high when the complete operation is performed. In SHIFT state the accumulator value is left shifted by 4 to realize division with 16. To minimize error initially the accumulator is loaded with 8.

Median Filter:

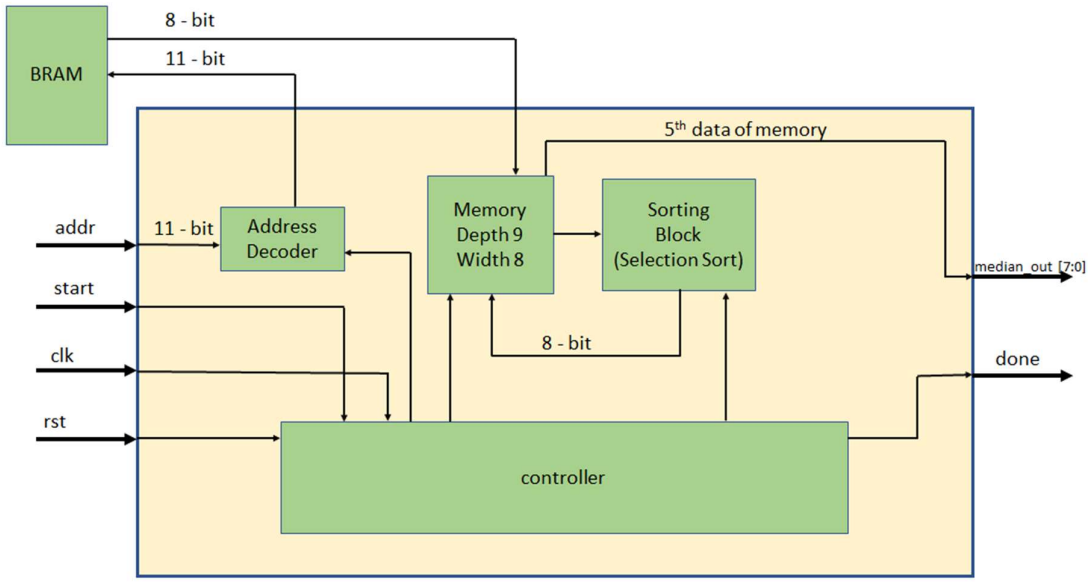

Figure 8. Architecture of Median Filter

The median filter block consists of address decoder, a temporary memory to store the 9-pixel values to evaluate the median, and the sorting block to sort the data. After storing the data, the data is sorted using section sort algorithm. The selection sort algorithm is stable and has complexity of $O(n^2)$. After the sorting the center vale is taken as the median of the data. This median filter block takes sequentially the address of each data then using the address decoder block other elements of the kernel can be accessed. The selection sort algorithm repeatedly scans for the minimum element and replaces it with the pivot data. At end the values are arranged in ascending order.

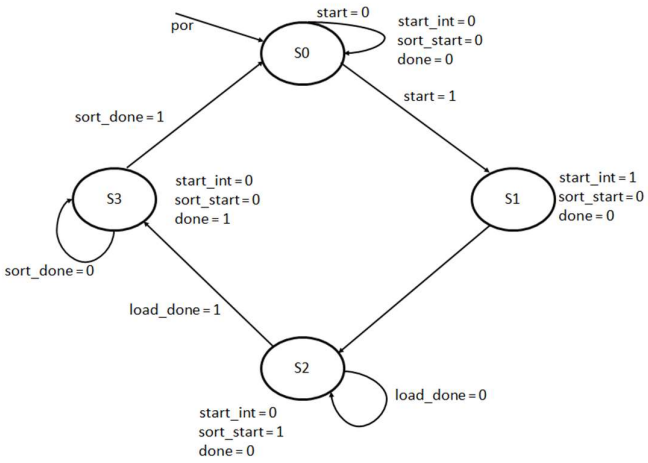

Figure 9. FSM for Median Filter

The median block state machine implementation for the control path is shown above. The global start signal indicates the starting of the block. Then start_int signal starts the block operation with accessing all the values required for operation and storing it in memory array. The load_done indicates that the all the 9-pixel values have been retrieved. Then the sort_start starts the sorting algorithm. The sorting process completion is signaled by sort_done. The power on reset state is the state S0.

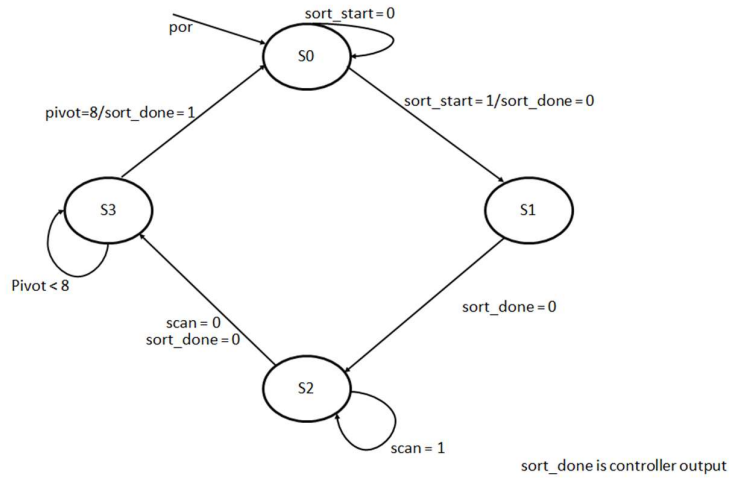

Figure 10. FSM for Selection Sort

The above state diagram represents the algorithm for selection sort. The sort_start stars the sorting process in sorting process initially the first element is taken as pivot the entire array is scanned for any element less than pivot. If found, then pivot is replaced by that element. This process is repeated sequentially for all the elements. And when the pivot reaches to last element the elements would be sorted. The sort_done signals the completion of sorting. Then the center value is retrieved o give as the median out.

Results

Generating Test Inputs:

We have obtained a 32x32 image from the internet. This is image is converted into gray scale and then exported into binary data using MATLAB. By using this data, a coe file is formed and loaded into BRAM. For median filtering we have added gaussian noise into the input image.

Obtaining Filtered Results:

To get hardware results, while simulation we have stored data out from the filter into an array. The data from this array is exported into a text file. This text file is used to construct filtered image in MATLAB.

The filtered images in MATLAB and Hardware are plotted in MATLAB and are shown in below figures.

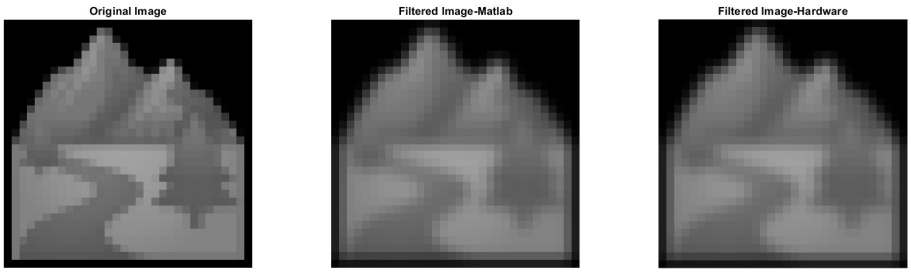

Figure 11. Gaussian Filter Result

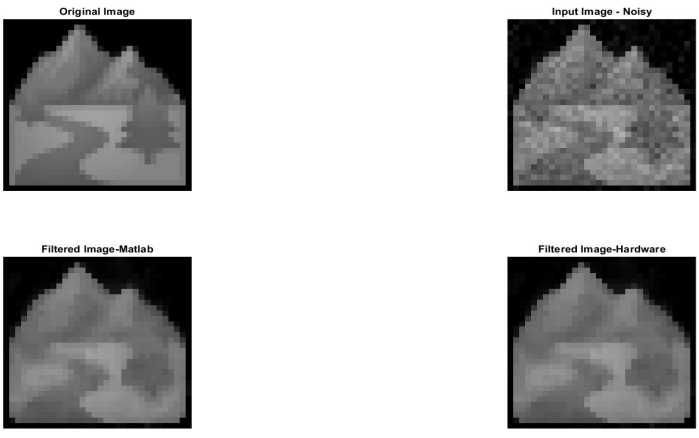

Figure 12. Median Filter Result

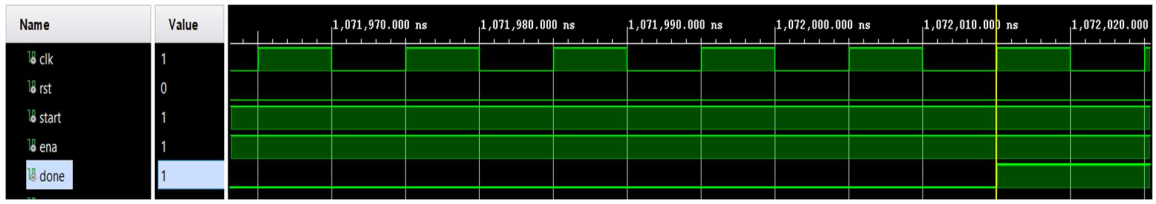

Figure 13. Behavioral Simulation Waveform

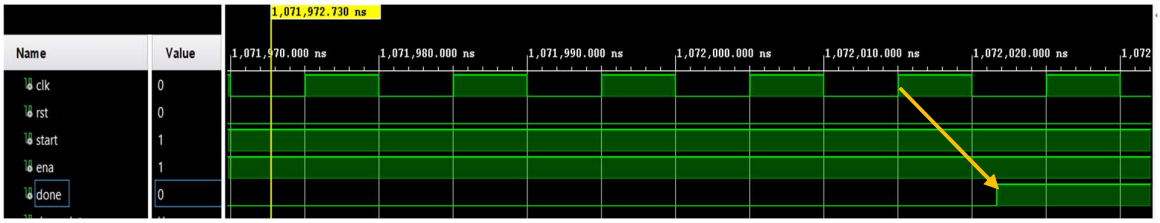

Figure 14. Post Implementation Timing Simulation Waveform

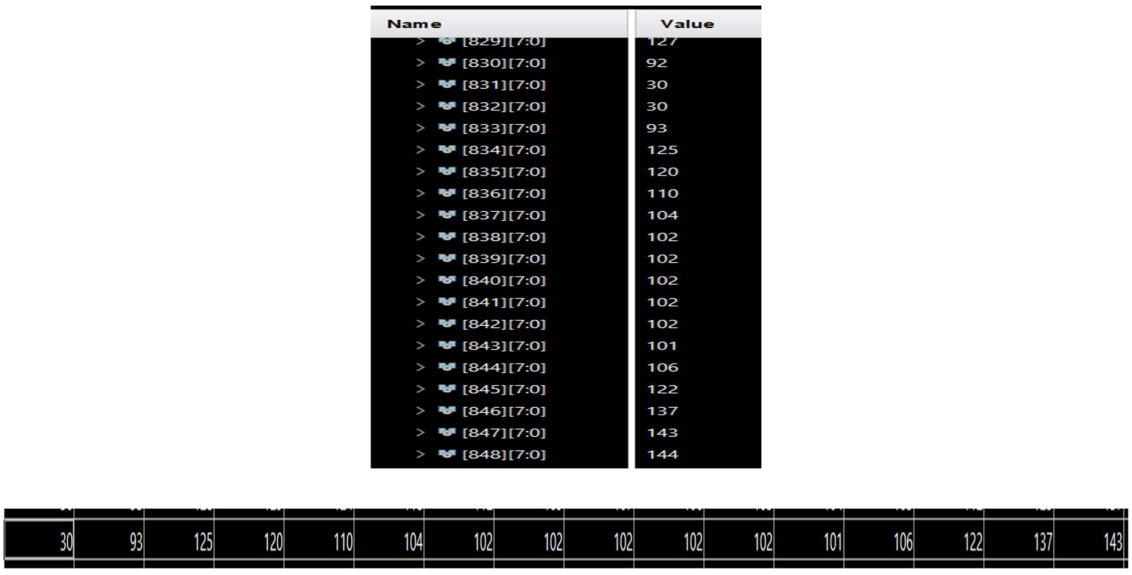

Figure 15. Comparison of Hardware and MATLAB Simulation for Gaussian Filter

The filtered image pixel values are compared with MATLAB simulation results and verified. Some of the values at intermediate addresses are taken just to visualize. These are shown in figure 15 and 16.

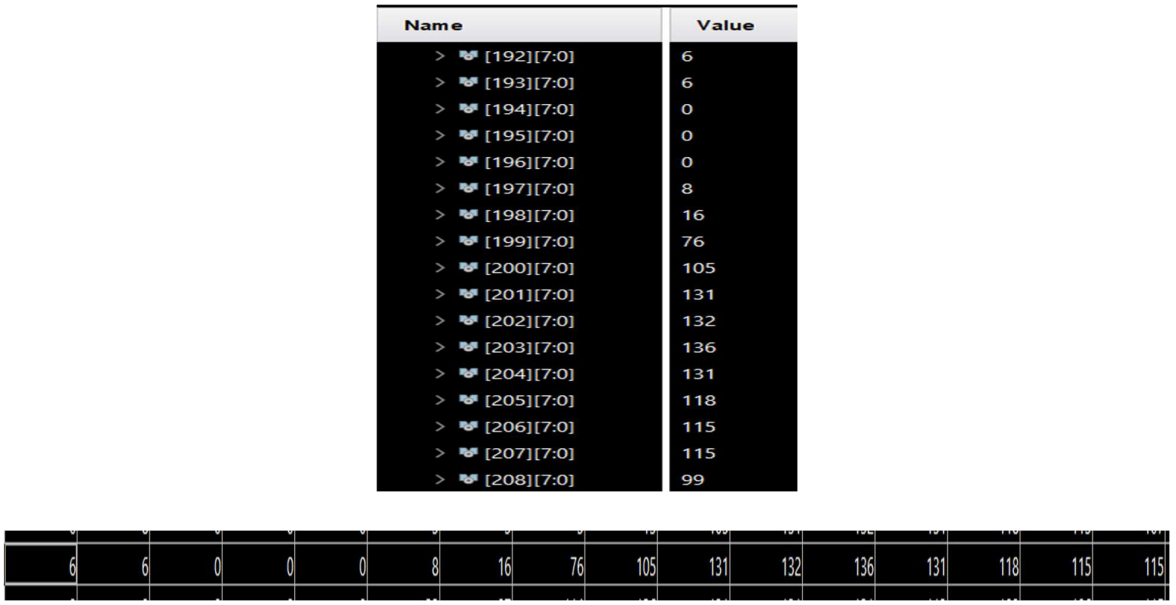

Figure 16. Comparison of Hardware and MATLAB Simulation for Median Filter

Summary

Target Device: xc7a35tcpg236-1 (Artix-7 FPGA Board)

Performance Specifications:

- Clock frequency – 100MHz
- Worst case slack – 0.235 ns
- Maximum clock frequency – 102.41 MHz
- Delay for gaussian filter – 133.2 us
- Delay for median filter – 932 us

Resource	Utilization	Available	Utilization...
LUT	153	20800	0.74
FF	166	41600	0.40
IO	5	106	4.72
BUFG	1	32	3.13

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	Bonded IOB (106)	BUFGCTRL (32)
N top_module	153	166	78	153	5	1
DUT2 (Sorting)	96	110	46	96	0	0
DUT3 (mul_add)	12	8	8	12	0	0

Figure 17. Resource Utilization

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.235 ns	Worst Hold Slack (WHS):	0.067 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	465	Total Number of Endpoints:	465	Total Number of Endpoints:	167

All user specified timing constraints are met.

Figure 18. Static Timing Analysis Report

Total On-Chip Power:	**0.07 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	59.7°C (11.9 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

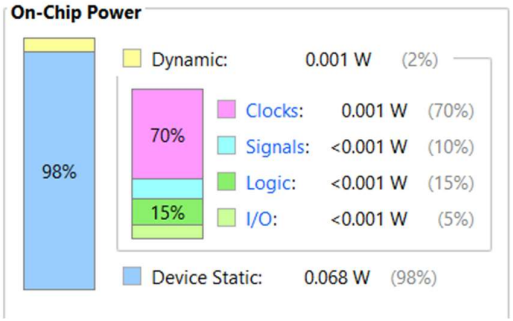

Figure 19. Power Estimation

Conclusions

In this project we have implemented an image filtering block which can perform either gaussian or median filter. We have optimized design in such a way that area and delay are minimum. For realizing addition and multiplication we have used existing operators. In median filtering we have used selection sort algorithm which gives stable results. The median filtering is performed on the image that is added with gaussian noise to the original image. The filtered image results obtained from the hardware design are same as the software simulation results.

The design can be further improved by adding additional functionalities such as pixel to pixel filtering techniques like brightness control, inversion, binarization of image. The gaussian filter can be further improved by adding programmable weights so that averaging, sharpening and any other convolution for given image can be performed.