International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Efficient Pipelined FPGA Implementation of Steerable Gaussian Smoothing Filter

Shraddha Barbole[1], Dr. Sanjeevani Shah[2]

[1]Electronics & Telecommunication Department, SKN College of Engineering, Vadgaon Bk, Pune, India

Abstract: ***Smoothing filters have wide area of applications such as image and video analysis, which extends to edge detection, motion analysis, line parameter estimation, and texture analysis. To achieve smoothing, it is essential to have directional smoothing filters which can be oriented in any arbitrary direction. For real time applications, hardware devices having capability of parallel processing can be used. The steerability is the property, in which several filtering operations outputs are linearly combined to achieve output of a directional filter which is arbitrarily oriented. Though the literature describes the several efficient FPGA implementations of the convolution operation for non-separable and separable, limited work is available related to steerable filter implementations. In this system, steerable Gaussian smoothing filters are implemented on an FPGA platform using Virtex-V ML506 evaluation board. The output is displayed on VGA display. The algorithm uses delaying of intermediate outputs which reduces memory requirements. This allows simultaneous implementation of both horizontal and vertical convolution. Due to pipelined approach, memory resources and other device utilization is reduced. The key advantages of FPGAs over DSP implementations include integration, performance and customization using design techniques of parallel and pipeline operations. A pipelined approach of convolution gives the less number of resources.***

Keywords: Steerability, Virtex-V, Parallel Processing, Pipelined Approach.

1. Introduction

Smoothing is achieved by convolution of an original image with an appropriate mask, such as a Gaussian mask. Convolution is a common image processing operation. It can be given as sum of products of the input image and a convolution Kernel. Using convolution, several imaging operations can be achieved depending on the selection of values in the convolution kernel. As directional or orientation filters [7] are widely used in computer vision and image processing applications its response at any arbitrary position and orientation is obtained by tuning the filter to all possible positions and orientations. However, such an approach requires a large number of computations, and is thus not easily implementable in real time. For instance, a reduction in the number of multipliers can be achieved through the use of separable filters. A separable filter of size $N \times N$ can be expressed as convolution between two filters of sizes N × 1 and 1 × N respectively.

$$H(m,n) = \sum_{i=0}^{height-1} \sum_{j=0}^{width-1} g(i,j) f(m-i, n-j) \qquad (1)$$

Where f denotes the input image, h is the output, g is the filter image whereas m and n denotes image dimensions. Gaussian mask required for 1D convolution has the following form.

Using this formula Gaussian coefficients are calculated.

$$g(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2} \qquad (2)$$

where g is the weight of Gaussian kernel at the location with coordinates x and y. The σ parameter is the standard deviation of the Gaussian distribution .The term $\frac{1}{2\pi\sigma^2}$ is normalization constant.

2. Literature Survey

In this paper, a Gaussian steerable multidirectional filter bank implementation on FPGA is proposed. Literature gives very little work on steerable filter implementation on FPGA. In [1], Area-Efficient 2-D shift-variant convolvers for FPGA-based digital image processing are proposed. Several novel FPGA-efficient architectures for generating a moving window over a row-wise print path are proposed and provided a criteria to choose the optimum one for any design point. Hui Zhang *et al.* proposed a Multiwindow Partial Buffering Scheme for FPGA Based 2-D Convolvers [2]. Erke Shang *et al.* in [3] put forth about Architectures for Generalized 2D FIR Filtering using Separable Filter Structures. The problem of generalized 2D FIR filtering for large filter kernel sizes can be computationally prohibitive when required in real-time implementation. In [4], C.S Bouganis *et al*. emphasized on steerable pyramid wavelet construction for image decomposition and feature detection, and its implementation on FPGA. Erke Shang *et al*. used steerable filters for lane detection and implemented it on FPGA [5]. For these approaches, large number of basis filters would be required. Instead, regardless of the desired angular resolution, Gaussian smoothers can be steered via the application of three 1D filtering operations [7].

Two approaches of separable convolution that are unpipelined and pipelined convolution can be used. But unpipelined approach is not efficient as it requires more memory and delay. This approach uses separate BRAM for storing the intermediate results and cannot start the next step until first step is completed. Simultaneous horizontal and vertical operations can't be implemented. As a result it increases delay, additional memory and resources. The pipelined approach which is used in [7] uses FIFO to store intermediate results. Hence in this paper this pipelined steerable convolution approach is modified by using delay

elements to delay the previous outputs instead of using any memory element to store intermediate results.

Steerability implies that the output of a filtering operation, $O_\theta(x,y)$ using a filter oriented at an angle θ can be computed as the linear combination of a finite set of M outputs $\{O\theta_0(x,y)$, $O\theta_1(x,y)$, ……., $O\theta_{M-1}(x,y)\}$ obtained by applying the same filter oriented at directions θ_0 , θ_1 , ……., θ_{M-1} respectively. A 2D separable and steerable filter can be written as:

$$g\theta(x,y) = \sum_{r=-R}^{R} g_{iso}(x - r\cos(\theta), y - r\sin(\theta)) g^{1D}(r) \quad (4)$$

The filter described in equation (4) can be applied to an image $I\ (x,\ y)$ in two steps. In the first step, the filter $g_{iso}(x,y)$ is applied to the image.

$$I_{iso}(x,y) = I(x,y) * g_{iso}(x,y) \quad (5)$$

In the second step, the following operation is applied to the image $I_{iso}(x,y)$

$$I_\theta(x,y) = \sum_{r=-R}^{R} I_{iso}(x - r\cos(\theta), y - r\sin(\theta)) g^{1D}(r) \quad (6)$$

The operation described in (5) and (6) is equivalent to the operation where Gaussian directional smoothing filter (DSF) oriented at direction θ filters the input image $I(x, y)$.

3. System Design

System architecture is shown in Figure 1. Input image which is to be smoothed is stored in BRAM. A pipelined steerable convolution algorithm is applied on this image. It gives horizontally and vertically smoothed images. These smoothed images are displayed on VGA monitor.

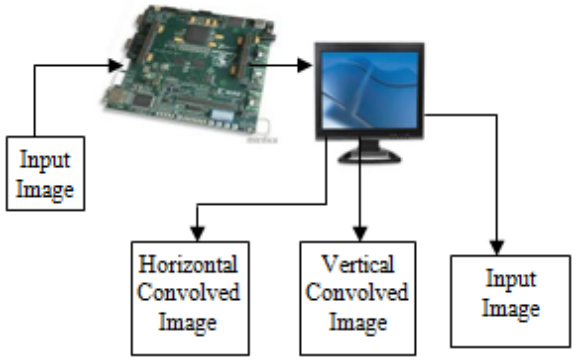

Figure 1: System Architecture

Figure 2 shows complete system implementation. BRAM is used to store a test image using .coe file. After 9 × 1 vertical convolution, the operation is performed so that after 9 clocks a horizontal convolution operation is possible. The vertical convolution results are delayed. Hence simultaneous vertical and horizontal convolution operation is possible. A pipelining technique can be used to obtain higher throughput. The Xilinx Platform Studio is used to generate a .bit file for the haze image and then downloaded on Virtex 5 evaluation board. Stepwise explanation of algorithm is given as below.

Step 1: Storing of image in BRAM
First step is to store an image of size 48 ×48 in BRAM, also a Gaussian mask is stored which are then read by read controller. An input image is stored using a block memory generator. Out of five memory types, True Dual Port RAM (TDPRAM) is selected for storing an image.

Step 2: Vertical 9 ×1 Convolution
After getting input image pixels and Gaussian coefficients from BRAM, these are used for vertical convolution process. In this step a 9×1 Gaussian Kernel is rotated over an image. Pixel value and Gaussian kernel value gets multiplied and added to get convolved output. This requires latency of two clock cycles.

Step 3: Delay of vertical convolution outputs
To avoid the memory required to store intermediate results of convolution, instead of using BRAM or other memory a delay element is used to delay the output.

Step 4: Horizontal 1 × 9 Convolution
In pipelined architecture, vertical and horizontal convolutions are performed simultaneously. For horizontal convolution, a Gaussian mask of size 1×9 is used. To start the horizontal convolution, 9 pixels should be available at a time. Hence as soon as the 9 outputs of vertical convolution are available, horizontal convolution is started.

Step 5: Storing of Convolution Results in BRAM
The horizontal convolution results are stored in BRAM. These are the obtained resuts after application of vertical and then horizontal convolution.

Step 6: Vertical 7 × 1 and Horizontal 1 × 7 Convolution
The last stage of the Gaussian steerable approach uses an operation equivalent to 1D filtering at the direction of interest. This operation is applied to the smoothed image obtained by the previous stages, which include filtering of the original image using an isotropic Gaussian filter.

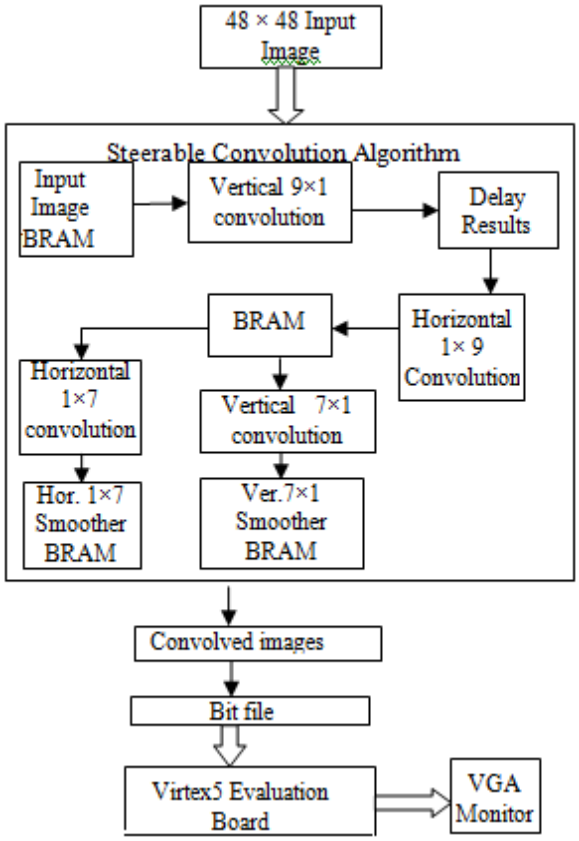

Figure 2: System Implementation Model

4. Selection of Gaussian Mask

Image pixels are represented using 8 bits. Separable Gaussian masks of 1×9 and 9×1 derived using equation (2.3) & (2.4) with mean equal to zero, σ equal to 1 and normalizing factor N = 0.0016 are shown below in Table 3.1. and Table 3.2.
When these values of mean, standard deviation are put in equation (2.2) which is given below, it gives the values of Gaussian mask.

Table 1: Vertical and Horizontal Gaussian mask with Mean = 0, σ = 1 and N = 0.0016

100
61
14
1
0
0
0
0
0

100	61	14	1	0	0	0	0	0

$$g(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2}$$

Consider above equation, where σ = 1 and x^2 = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 respectively. Values of x^2 are put one by one and multiplied by 256 because all values are represented in 8 bits in FPGA. For $x^2 = 0$, $g(0) = 100$. Similarly other values can be obtained. The same coefficient values are used for vertical as well as horizontal Gaussian mask.

Table 2: Vertical and Horizontal Gaussian mask with Mean = 0, σ_x = 3, σ_y = 5, N = 0.001

8
33
76
100
76
33
8

8	33	76	100	76	33	8

Steerability is applied on the obtained results of separable convolution between the 48×48 image and the Gaussian masks of 9 × 1 and 1 × 9. Further experimentation is done on image of size 158 × 158 and 256 × 256. For applying steerability, a steerable Gaussian mask is derived using equation (2.9) and decimation factor using the equation (2.10). The derived steerable Gaussian masks of 7 × 1 and 1 × 7 for mean = 0, σ_x = 3, σ_y = 5, Normalizing factor N = 0.001

5. Results and Discussions

In this work, experimentation is done on image of size 48 × 48; it is again extended for image of size 158 × 158 and 256 × 256. Gaussian mask is selected of size 9 × 9 and for further operations it is 7 × 7. The Gaussian coefficients are calculated using its formula. Standard deviation and normalization factor decided accordingly. The convolution output is displayed using output device VGA display. Target device used is VIRTEX -V evaluation board. Xilinx ISE 14.2 is used for simulation and implementation. First image is stored in TDPRAM. Memory block generator is generated using IP core generator. Minimum area algorithm is selected. Write width and read width is selected as 8. Write depth and read depth for 48 × 48 image is 2304, for 158 × 158 it is 24964 and for 256 × 256 it is 65536. Memory initialization is done by selecting .coe file.

Figure 3: Steerable convolution output

Consider a convolution operation output shown in Figure 3. Here convolution of 7 pixels is performed with Gaussian mask of size 7 × 1 and 1 × 7. Output is obtained after the latency of 2 clock cycles as it is required for multiplication and addition. Normalization is done by dividing this output by sum of gaussian kernel.

$$\{[(52\times8) + (42\times33) + (31\times76) + (48\times100) + (54\times76) + (37\times33) + (50\times8)] / (8+33+76+100+76+33+8)\} = 44$$

RTL schematic of this design is given in Figure 4. It has number of components like counters steerability flag component. The implementation of proposed algorithm is carried out for three different sizes of images. These are 48 × 48, 158 × 158, 256 × 256. Device utilization summary of this method is compared for these three image sizes. It is given in Table 3. Following observations are drawn from Table 3.

- Utilization of slice registers for 48 × 48 size image is 850(2%), for 158 × 158 image it is 537(2%) and for 256 × 256 it is 521(1%). This indicates that, utilization of slice registers decreases with increase in size of an image.

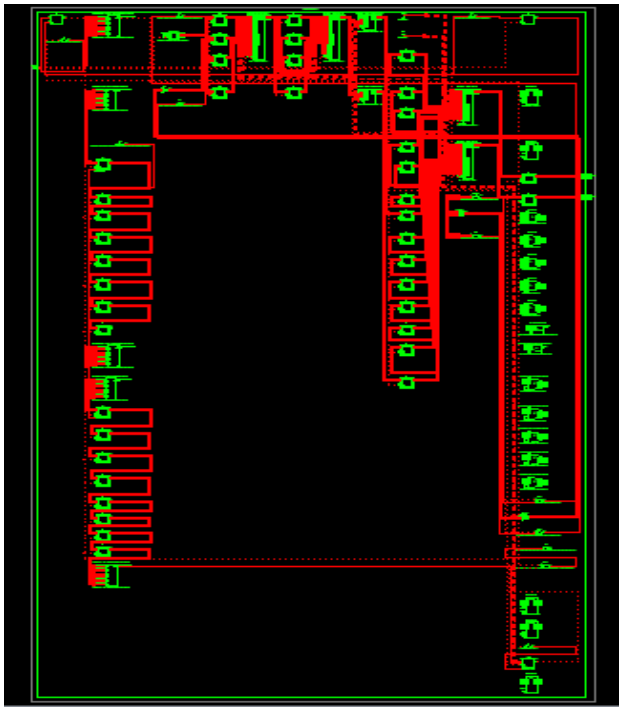

Figure 4: RTL schematic of steerable convolution

Table 3: Device utilization summary of different sizes of images

Device Utilization Summary (Estimated values)			
Logic Utilization	**48 × 48**	**158 × 158**	**256 × 256**
Number of Slice Registers	850(2%)	537(2%)	521(1%)
Number of slice LUTs	762(2%)	505(2%)	459(1%)
Number of used LUT-FF pairs	334(26%)	225(26%)	225(26%)
Number of bonded IOBs	18(3%)	17(3%)	17(3%)
Number of Block RAM	5(3%)	20(3%)	48(36%)
Number of BUFG	2(6%)	2(6%)	3(9%)
Number of DSP48Es	15(5%)	17(5%)	17(5%)

- Slice LUTs used are 762(2%) for 48 × 48 image, 505(2%) for 158 × 158 image and 459 for 256 × 256 image. It gives reduced number of sources for large size of image.
- LUT-FF pairs used are 334, 225 and 225 respectively. It decreases with increase in size of image.
- Bonded IOBs used are 18, 17 and 17 which decreases as size of image increases.
- Utilization of block RAM is 5(3%) for 48 × 48 image, 20(6%) for 158 × 158 image and 48(36%) for 256 × 256 image. It increases as size of image increases.
- BUFG and DSP48Es increases with size of image.

The device utilization summary of this method when compared with the method used in [10], it gives reduced number of resource utilization shows that the proposed methodology is efficient. The comparison is given in Table 4. The observations drawn from this table are as follows:

Table 4: Comparison of this method with previous steerable convolution method

Method	*LUT*	*IOB*	*BRAM*	*Multipliers*	*Clocks per Pixel*
Pipelined Steerable	22464 (82%)	22 (4%)	39 (29%)	32 (24%)	2
ProposedPipelined Steerable	505 (1%)	17 (3%)	20 (15%)	17 (5%)	2

- The steerable implementation of pipelined horizontal and vertical convolution operation which is used in [7] requires huge number of resources.
- Proposed method of pipelined steerable horizontal and vertical convolution gives better performance with less number of resource utilization.
- Number of clock cycles required for both the operations are same. It requires two clock cycles for operation of per pixel.

The implementation of proposed method requires 9 clock cycles for first vertical convolution, after which both horizontal and vertical convolution starts simultaneously. For next step of convolution it adds delay of 7 clock cycles to get first output. The convolution operation adds latency of two clock cycles, one for multiplication and other for addition of pixels. Along with all the devices, time required to implement this method can be calculated. This time is compared with the method used in [7]. This comparison is given in Table 5.

Table 5: Comparison of FPGA execution times

Method	Previous Pipelined Steerable Impleme-tation	C Based Steerable Implemen tation	Proposed Pipelined Steerable Implemen tation
Execution Time for 158×158 Image	0.492 ms	6.5 ms	0.112 ms

The output images obtained after convolution are displayed on VGA display. Due to small size of image VGA display gives poor quality of image. Hence for better quality and for ease of observation size of image is increased to 256 × 256. These increased sizes of images, displayed on VGA are

shown in Figure 5, Figure 6 and Figure 7. The experimental setup is shown in Figure 8.

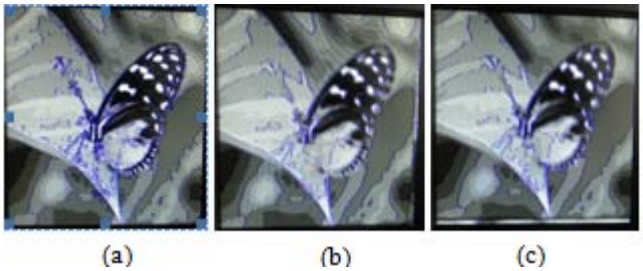

Figure 5: (a) Input Image (b) Vertical Convolution Output (c) Horizontal Convolution Output

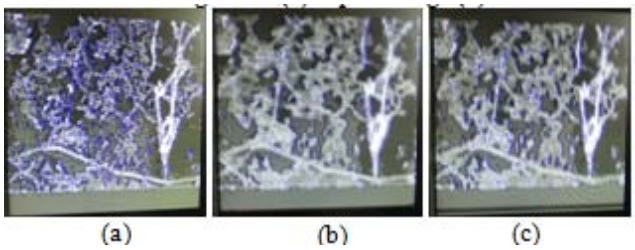

Figure 6: (a) Input Image (b) Vertical Convolution Output (c) Horizontal Convolution Output

Figure 7: (a) Input Image (b) Vertical Convolution Output (c) Horizontal Convolution Output

Figure 8: Experimental Setup

6. Conclusion

Steerable implementation in horizontal and vertical directions using the pipelined separable convolution method requires fewer resources. It gives this complicated operation easily and with less utilization of resources.

Utilization of slice registers, slice LUTs, LUT-FF pairs and bonded IOBs decreases as size of image increases. Utilization of block RAM, BUFG and DSP48Es increases with size of image. The steerable implementation of pipelined horizontal and vertical convolution operation which is used in previous method requires huge number of resources. Proposed method of pipelined steerable horizontal and vertical convolution gives better performance with less number of resource utilization. Number of clock cycles required for both the operations are same. It requires two clock cycles for operation of per pixel.

References

[1] Francisco Cardells-Tormo and Pep-Lluis Molinet, "Area-Efficient 2-D Shift-Variant Convolvers for FPGA-Based Digital Image Processing", IEEE Transactions On Circuits And Systems—Ii: Express Briefs, Vol. 53, No. 2, February 2006, pp 105-109.

[2] Hui Zhang, Mingxin Xia, and Guangshu Hu, "A Multiwindow Partial Buffering Scheme for FPGA Based 2-D Convolvers, IEEE Transaction on Circuits and Systems, Feb 2007.

[3] D. Venkateshwar Rao and M. Venkatesan , "An Efficient Reconfigurable Architecture and Implementation of Edge Detection Algorithm using Handle-C", International Journal of Engineering and Applied Sciences, 2006.

[4] M.S. Andrews, "Architectures for Generalized 2D FIR Filtering using Separable Filter Structures", Proceeding of Acoustics, Speech and Signal Processing, 1999.

[5] C.S Bouganis, P.Y.K Cheung, J.Ng and A. Bharath, " A Steerable Complex Wavelet Construction and its Implementation on FPGA", in Proc. International Conference on Field Programmable Logic and Applications, 2004, pp.394-403.

[6] Erke Shang, Jian Li, Xiangjing An and Hangen He, "Lane Detection using Steerable Filters and FPGA-based Implementation", 2011 Sixth International Conference on Image and Graphics,2011, pp 908-911.

[7] A. Joginipelly, Alvaro Varela, Dimitrios Charalampidis, et.al. "Efficient FPGA Implementation of Steerable Gaussian Smoothers", 44th IEEE Southeastern Symposium on System Theory University of North Florida, Jacksonville, FL,2012, pp.78-82.

[8] Dimitrios Charalampidis, "Efficient Directional Gaussian Smoothers" , IEEE Geoscience And Remote Sensing Letters, Vol. 6, No. 3, July 2009, pp. 14-19, pp 383-387.

[9] V. Lakshmanan, "A Separable Filter for Directional Smoothing", IEEE Geoscience And Remote Sensing Letters, Vol. 1, No. 3, July 2004 pp 192-195.

[10] Arjun Joginipelly, "Implementation of Separable & Steerable Gaussian Smoothers on an FPGA" University of New Orleans Theses and Dissertation, Dec. 2010.

[11] Xilinx, "Block Memory Generator v 2.8 Logic Core", Sept 2008.

[12] Xilinx, "Logic Core IP Multiplier v 11.2", Sept 2009.

[13] Virtex-5 FPGA Configuration User Guide UG191 (v3.11) October 19, 2012

Author Profile

Shraddha A. Barbole was born in 1990. She has received B.E degree in Electronics and Telecommunication Engineering from Solapur University, persuing her M.E degree in Electronics and Telecommunication with specialization in VLSI and Embedded System from Smt. Kashibai Navale College of Engineering, Vadgaon(bk), Pune in Pune university.

Dr. Sanjeevani K. Shah obtained her phD (E&TC) in 2011 from university of pune. Worked in Philips India Ltd. for three Years. Thereafter has twenty seven years of teaching experience. Presently working as Head of Post graduate department E&TC in STES's SKN College of engineering. Published books on Industrial Electronics, Communication, Applied electronics and has published 25 papers.