© 2020 IJSRST | Volume 5 | Issue 7 | Print ISSN : 2395-6011 | Online ISSN: 2395-602X

4th National Conference on Advances in Engineering and Applied Science
Organized by : Anjuman College of Engineering and Technology (ACET) Nagpur,
Maharashtra, India, In association with
International Journal of Scientific Research in Science and Technology

FPGA Implementation of Filtered Image Using 2D Gaussian Filter

Mrs. Ruhina Quazi, Ankit Bodele, Amit Fernandez, Sylvester Clarke, Tushar Madavi
Department of Electronics and Telecommunication , ACET, Nagpur, Maharashtra, India

ABSTRACT

Image filtering is one of the very useful techniques in image processing and computer vision. It is used to eliminate useless details and noise from an image. In this paper, a hardware implementation of image filtered using 2D Gaussian Filter will be present. The Gaussian filter architecture will be described using a different way to implement convolution module. Thus, multiplication is in the heart of convolution module, for this reason, three different ways to implement multiplication operations will be presented. The first way is done using the standard method. The second way uses Field Programmable Gate Array (FPGA) features Digital Signal Processor (DSP) to ensure and make fast the scalability of the effective FPGA resource and then to speed up calculation. The third way uses real multiplier for more precision and a the maximum uses of FPGA resources. In this paper, we compare the image quality of hardware (VHDL) and software (MATLAB) implementation using the Peak Signal-to-Noise Ratio (PSNR). Also, the FPGA resource usage for different sizes of Gaussian kernel will be presented in order to provide a comparison between fixed-point and floating point implementations.

Keywords : Gaussian Filter; convolution; fixed point arithmetic; Floating point arithmetic; FPGA

I. INTRODUCTION

Convolution has been widely used in computer vision and image processing, including object recognition [2] and image matching [3], However, convolution operation typically requires a significant amount of computing resources [4].Image filtering is applied as pre-processing to eliminate useless details and noise from an image. It is produced by convolution between an image and 2D Gaussian mask. In the literature, several efficient FPGA implementations of the 2D convolution operation have been proposed [5]–[9].

Hanumantharaju et al. [10] proposed a hardware architecture suitable for FPGA/ASIC implementation of a 2D Gaussian surround function for image processing application which offers a savings of memory. Barbole et al. [11] implemented steerable Gaussian smoothing filters on an FPGA platform based on a VirtexV ML506 using the pipelined approach and DSP which reduces memory requirements. Talbi et al.[5] developed architecture for separable and two-dimensional Gaussian smoothing filters, which was implemented in the VirtexV FPGA platform. They prove that the first approach is significantly faster than the second one. In the same year, Cabello et al. [2] implemented a 2D Gaussian Filter in FPGA using fixed-point arithmetic and floating point arithmetic, they found that increasing the kernel sizes, they reduced the computational costs using floating point arithmetic

In this paper, a Gaussian filter on an Field Programmable Gate Array (FPGA) platform will be implemented. We will focus in the main bloc which is the convolution module based on the multiplication operation. Thus, the multiplier is in the heart of the proposed design. For this, the standard multiplier will be firstly implemented. Then, in order to accelerate calculus and to minimize resource use, FPGA features will be used which are DSP (Digital Signal Processor) and RAMs. Finally, in order to have more precision in image output, a real multiplier proposed in [13] will be used to implement the entire architecture. It is a new way to do a multiplication between two real numbers. Our application is implemented by two tools such as MATLAB and VHDL, and simulated on the ISE simulator.

The remainder of this paper is as follows. Section 2 introduces the image filtering algorithm. The hardware implementation of image filtering is presented in section 3. In section 4, the hardware optimization of convolution module based on changing the multiplier will be discussed. Experimental results are given in section 5. Finally, a conclusion will be done in section 6.

II. IMAGE FILTERING ALGORITHM

Smoothing filters are widely used in many applications such as object recognition, matching, classification, etc. They are applied as pre-processing for removing useless details and noise [14]. We will focus on image filtering based on Gaussian filter.

A. Gaussian mask

Gaussian filter is one of the most important and widely used filtering algorithms in image processing [5]. Gaussian filter (G) is defined in equation 1.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (1)$$

where G is the Gaussian mask at the location with coordinates x and y,α is the parameter which defines the standard deviation of the Gaussian. If the value of is large, the image smoothing effect will be higher.

B. Convolution operation

In general, smoothing can be effected by convolve the original image I(x,y) of the size h x w with a Gaussian mask G(x,y) as illustrated in equation 2. It is obtained by computing the sum of products among the input image and a smaller Gaussian matrix of the size(3×3). A 2D convolution using a 3×3mask and 3×3 input image is illustrated in Figure reffig1.

$$f(x,y) = \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} G(i,j) I(x-i, y-j) \quad (2)$$

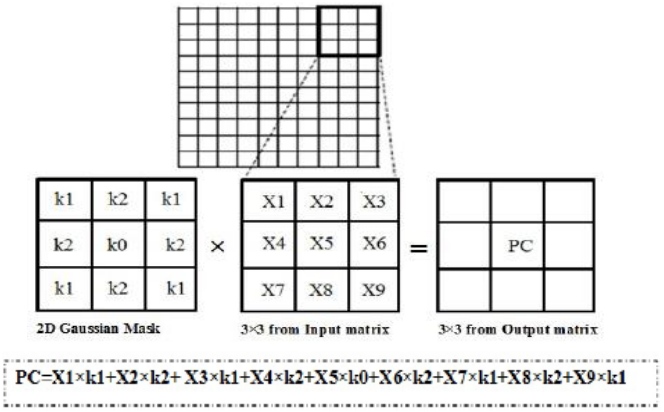

Fig. 1: Convolution operation

III. HARDWARE IMPLEMENTATION OF IMAGE FILTERING

In this section, the proposed architecture design of the Gaussian filter will be presented

A. Block diagram of image filtering

Figure 2 illustrates the block diagram of image filtering. First, the input image and the Gaussian mask are read and saved by MATLAB. Next, These

values are converted into a vector in a text file extension *.coe using the MATLAB tool and loaded the text file in block RAM (BRAM). The text file of Gaussian mask and image is stored respectively in BRAM1 and BRAM2. After that, the convolution operation is effected between these pixel values of two BRAM (1 and 2) using VHDL tool and saving the obtain results in another block (BRAM3). Finally, the text file of BRAM3 is converted by MATLAB tool in order to display the results form an image. The next step, we defined each block of diagram in Figure 2.

B. Synchronous architecture hardware of image filtering

Figure 3 depicts the block diagram of synchronous image filtering which contains a set of modules: Control Module, 3 BRAMs (matrix of input image, matrix of Gaussian mask, matrix of filtered image) and convolution Module.

1) Gaussian Filter

The convolution of an image with a Gaussian mask

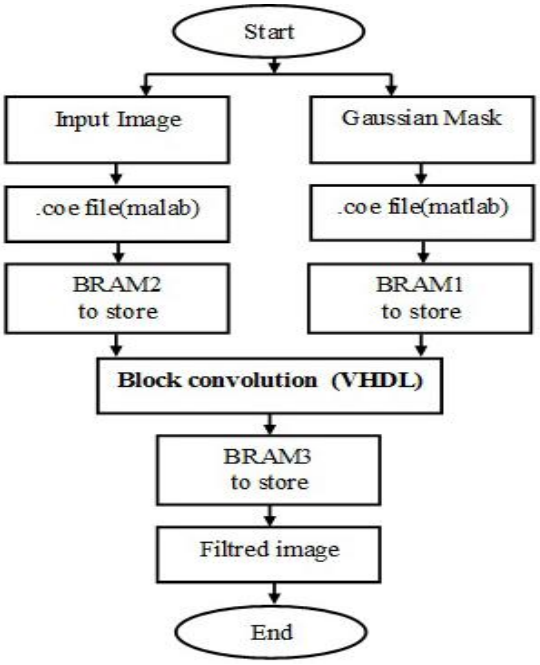

Fig. 2: Block diagram of image filtering

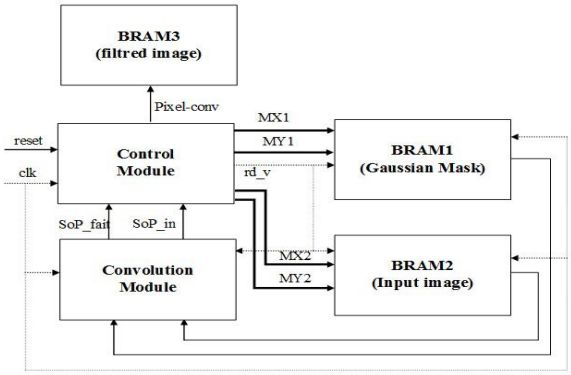

Fig. 3: Synchronous architecture of image filtering

involves floating point multiplications, which consumes considerable hardware resources. The Gaussian mask size (3×3) is presented by the matrix below by choosing the standard deviation equal to 0.5.

$$\begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}$$

Then, it is necessary to convert the floating point coefficients to fixed integer point coefficients for hardware implementation of the Gaussian filter. In the convolution process, each mask values has to be multiplied with each element of the image and then divided by a power of 2 [15], [16]. The approximation of the Gaussian mask is presented by equation below.

$$\begin{cases} G(x,y) = \frac{1}{2^8}\begin{bmatrix} 3 & 21 & 3 \\ 21 & 158 & 21 \\ 3 & 21 & 3 \end{bmatrix} \\ = \begin{bmatrix} 0.0117 & 0.082 & 0.0117 \\ 0.082 & 0.6172 & 0.082 \\ 0.0117 & 0.082 & 0.0117 \end{bmatrix} \end{cases}$$

2) Block RAM

In Xilinx FPGAs, a Block RAM (BRAM) is a dedicated two-port memory that stores up to 36Kb of data. The FPGA contains many of these blocks. Inside of each, small logic block is a configurable lookup table. It is normally used for logic functions, and it can be also reconfigured as a few bits of RAM. Several of them can be combined into a larger RAM which is denoted by a distributed RAM. BRAM is synchronous, this means that the read and write operations from

and to the memory are based on the clock input signal. The read and write operations are also dependent on the read/write enable ports. In our case, BRAM2 is used to store the data test image using .coe file which is generated with Matlab tool, and a BRAM1 is used to store the .coe file of Gaussian mask, which are then read by the control module. BRAM3 will save the data filtered.

3) Control module

The control unit is an important step of the proposed synchronous architecture. It allows to generate the address to BRAMs (1 and 2) and transfers the data from each BRAM to the corresponding convolution module for computing the Sum of Products (SoP) between these values, after that the convoluted value is stored in BRAM3. The control module is designed as a Finite State Machine (FSM) simulated in VHDL. Figure 4 illustrates the Finite State Machine (FSM) of the control module.

In the first state, initialization parameter will be affected. Then in state 1, the signal rd-v will be putted to 1 to access both memories. FSM increments the counter MY1 and MY2 when the MX1 and MX2 counter are finished addressing a line of image pixel block (3 by 3) and the same Gaussian block. This process is repeated the addressing of the blocks, if it is completed then goes to state 2 if not it returns to state 1. States 2 and 3 represent two late cycles to synchronize system signal. After that, it goes to state 4 where the machine puts the rd-v signal to zero in order to stop the addressing of the two memories and goes to state 5. In the state 5, the machine tests the SoP-fait signal, if it is equal to zero then it returns to the same state, if not it stored the value of SoP-in a table. After that, it increments the counter one " i " or " j " in order to read a new block, if " i " is different to the (length of size image -1) and " j "is different (width of size image -1) then returns to state 1. If not goes to state 6 (end process). Where,

X is the length of size image -1) and Y is the width of size image -1.

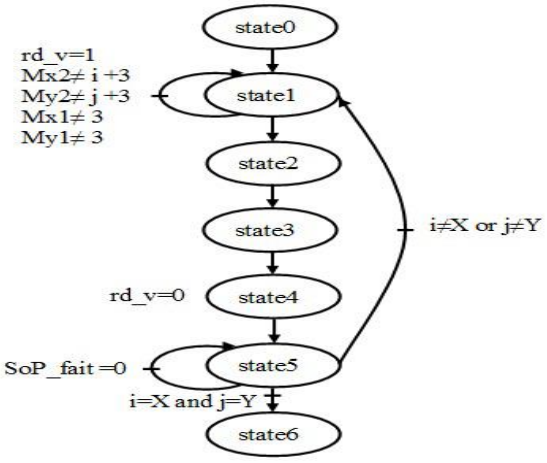

Fig. 4: FSM of the control unit

4) Convolution Module

Convolution module focuses on the calculation of the sum of products (SoP) between pixels in BRAM1 and BRAM2 for a window of 3 by 3. Equation 5 depicts an example of the convolution module between Gaussian mask integer and matrix 3×3 from input image.

A. Performance Measures

The Peak Signal to Noise Ratio (PSNR) is the most used parameter to evaluate image quality in the literature [11], [17]– [20], [22]. PSNR value can be computed by comparing two images which are original image and filtered image. The PSNR was used to measure the image quality. A higher PSNR value indicates that the filtered image contains better image quality. The PSNR has been calculated as follows;

$$PSNR = 10 \log_{10}\left(\frac{2552}{MSE}\right) \quad (3)$$

Where, MSE is the Mean Square Error (equation4)between the original image (I1(m,n)) and the filtered image (I2(m,n)), with, m and n are pixels of image M N.

$$MSE = \frac{1}{M \times N} \sum_{m=1}^{M} \sum_{n=1}^{N} (I_1(m,n) - I_2(m,n))^2 \quad (4)$$

B. Simulation results in MATLAB and VHDL

In this section, simulation and implementation results will be done. Figure 6 presents the filtered image by two tools which are MATLAB and ModelSim-SE (VHDL)

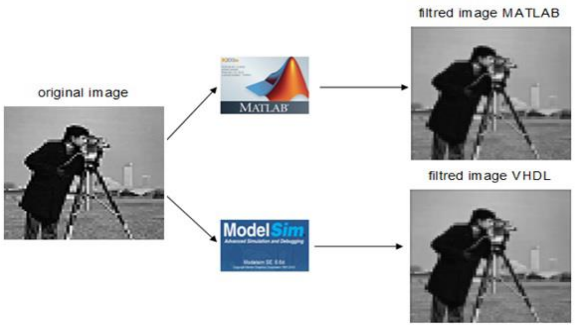

Fig. 6: Resulting filtered image in both MATLAB and VHDL

The kernel size 3×3 will be conserved and sigma values will be changed in order to see their impact in the filtered image. Figure 7, 8 and 9 illustrate the filtered image by the software (MATLAB) and hardware (VHDL) implementations. We can deduce that the blurring effect increases proportional to the sigma value (respectively 0.5, 1 and 1.5).

For different sigma values, Table I resumes the corresponding PSNR of images (in both VHDL and MATLAB).

For sigma equal 0.5, we observe that the PSNR (VHDL)obtains better result compared to PSNR (MATLAB). So, when increase sigma, the PSNR value of MATLAB and VHDL are decreased. Figure 10 shows the comparison between PSNR values both resulting image in MATLAB and VHDL.

Normally, if PSNR value is more than 40 dB, this is an indication that the quality of the image is good. But, if the image is mean quality, the PSNR value is less than 30 db which is the case of our selected image. We note that when we vary the sigma value the effect of smoothing increase and the PSNR decrease.

TABLE I: PSNR values for different output images in VHDL and MATLAB

	PSNR (MATLAB)	PSNR (VHDL)
Sigma = 0.5	25. 2236	27.3294
Sigma = 1	19.8879	20.3760
Sigma = 1.5	18. 0441	19.6098

IV. CONCLUSION

Hardware implementation of the Gaussian filter is faster than software one. Thus, using FPGA we are able to process the filtering at the same time of reading the image. In this paper, we have presented the implementation of two-dimensional convolution on a Xilinx VirtexV FPGA platform based on a state machine. We implemented Gaussian filters with different sigma values. Then we optimized the proposed architecture using different multipliers. At the first, we used the standard multiplication "" used in VHDL language. Then we explored FPGA features and DSP blocks. Finally, we introduced floating point arithmetic. Performances and results show that area and resources utilization decrease specially when using DSP and BRAM of FPGA. Also, speed increase comparing to the other solutions. By using floating point arithmetic the image has more precision and result seems to be is better.

V. REFERENCES

[1]. H. Kopka and P. W. Daly, A Guide to LATEX, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2]. DG. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision; 60(2), pp. 91-110, 2004.

[3]. L. Kabbai, M. Abdellaoui, A. Douik, New robust descriptor for image matching, Journal of Theoretical and Applied Information Technology, 87(3), pp. 451- 460, 2016.

[4]. L.Rao, B.Zhang,J. Zhao, Hardware Implementation of Reconfigurable 1D Convolution, Journal of Signal Processing Systems, 82(1), pp. 1-16, 2016.

[5]. F.Talbi, F.Alim, S. Seddiki, I. Mezzah, B. Hachemi , Separable Convolution Gaussian Smoothing Filters on a Xilinx FPGA platform, International conference on innovative computing technology (INTECH),Galcia, pp.112-117, May 2015.

[6]. M. Neggazi, M. Bengherabi, A. Amira,. Boulkenafet, An Efficient FPGA Implementation of Gaussian Mixture Models Based Classifier, IEEE. International Workshop on Systems, Signal Processing and their Applications (WoSSPA), Algiers , pp. 367-371.May 2013.

[7]. H. Zhang, M. Xia, and G. Hu, A Multiwindow partial buffering scheme for FPGA based 2-D convolvers, IEEE Transactions on Circuits and Systems II: Express Briefs, 54(2), pp. 200 - 204, February 2007.

[8]. L. Chang, J. Hernndez Palancar, L.E. Sucar, M. Arias-Estrada, FPGAbased detection of SIFT interest key points, Machine vision and applications,24(2), pp.371-392, 2013.