Efficient Hardware Implementation of 2D Convolution on FPGA for Image Processing Application

Manupoti sreenivasulu
Electronics& Telecommunication Engineering
NIT RAIPUR
cnu8501@gmail.com

T. Meenpal
Electronics& Telecommunication Engineering
NIT RAIPUR
tmeenpal.etc@nitrr.ac.in

***Abstract*—Most image processing algorithms are regional and two dimensional (2D) by nature. This implies that 2D convolver function has great consequences for image processing application. 2D Convolution filtering is a technique that can be used for an immense array of image processing objective some of which include that as images sharpening, image smoothing, edge detection, and texture analysis. Our main goal in this paper is to develop an efficient architecture for the 2D convolution using control blocks. Hardware implementation of this 2D algorithm can be realized by a reduced number of shift registers, multipliers, adders, and control blocks, thus leading to considerable hardware saving and fewer number LUTs. Simulations had done in Verilog and prototyped on devices technology of Xilinx Spartan 3E Field Programmable Gate Arrays (FPGA) platform. The proposed 2D convolution architecture approach significantly faster and maximum time required after clock time is less compared to the existing 2D convolution implementation. The hardware created with the proposed architecture will save a lot of computational time.**

Keywords—2D convolution, Field-programmable gate arrays (FPGA), Image processing, Xilinx Spartan 3E.

I. INTRODUCTION

Digital image processing techniques are motivated by three major important applications. The first application was an improvement of the pictorial information for human perception. So it implies that whatever image have, that want to enhance the quality of the image, means that the image wants to increases quality of the original image. So that the image will have the better look compared to the original image. The second important application of digital image processing techniques is for autonomous machine applications. There is a third application which is efficient storage and transmission. If store the image on a computer, then image need a certain amount of disk space [1]. So that the disk space required for storing the images will be less, otherwise more memory need for storing the images.

Two Dimensional (2D) convolution is over a large area used in image processing and video processing application. 2D convolution block is very important for Digital image processing for image smoothing, image sharpening, edge detection [2] [3] [4]. 2D Convolution is one of the most important operations for the signal and image processing applications. It could operate in 1D, 2D, and 3D thus enabling speech processing, image processing, and video processing respectively. In the convolution 2D spatial which is mostly used in image processing for the feature extraction sum information and also used for the spaces application [4]. Therefore want to achieve the best trade-off between the area (less number of LUTs), speed and mouldability. The proposed to implement 2D convolution architecture on Xilinx Spartan 3E FPGA platform.

The rest of this paper is organized as follows. Section 2 introduces the 2D convolution. Section 3 discusses the proposed architecture and the hardware implementation for 2D convolution. Section 4 discusses the MATLAB results of this paper, and Section 5 discusses the simulation results, Section 6 introduces performances comparison and section 7 discuss the conclusion of this paper.

II. 2D CONVOLUTION

Image processing filtering concept useful for many applications, for example, when the image dispatch or image acquisition stage or image compression the image that time noises signal will be easily introduced into the original images. So need additional step required for reduction of the noises in the original image. Image filter concept plays a major role in digital image processing tasks such as image segmentation, image smoothing, sharpening, and edge detection. A filter can be defined by a kernel, which is a small array applied to each pixel and its neighbors with in the image. In most applications, the center of the kernel is aligned with the current pixel. Image convolute with kernel gets the resultant image. In the convolution concept kernel. matrices are played the major role, different kernel matrices give the different resultant of the images. Based on the convolution operation can be achieved by the blurring, sharpening, edge detection, and all other operation depending upon on the selection of the kernel matrices. Convolution of the mathematical equation can represent the below.

$$F(x,y)=\frac{\sum_{x=1}^{3}\sum_{y=1}^{3} f(x,y).h(x,y)}{\sum_{x=1}^{3}\sum_{y=1}^{3} h(x,y)} \quad (1)$$

Image pixel values represent the f(x, y), and kernel convolution matrix represents the h(x, y), and resultant of the convolution for the image matrix F(x, y). Fig. 1 diagram shows the convolution operation, here image kernel matrices consider 3×3, convolution with the image 4×4 matrices gets the resultant convolution of the matrices. The elements of the resulting array (which is the same size as the kernel) are averaged, and the original pixel values are replaced with the resultant of the convolution output values. The process will be repeated, now the total image pixel values in the matrices replaced with the convolution of the resultant values. When kernel pixel value outside the matrices, implies that resultant of the convolution pixel value should be zero. The input to the block is the central pixel value to be processed and along with its neighboring pixels.

978-1-5386-8158-9/19/$31.00 ©2019 IEEE

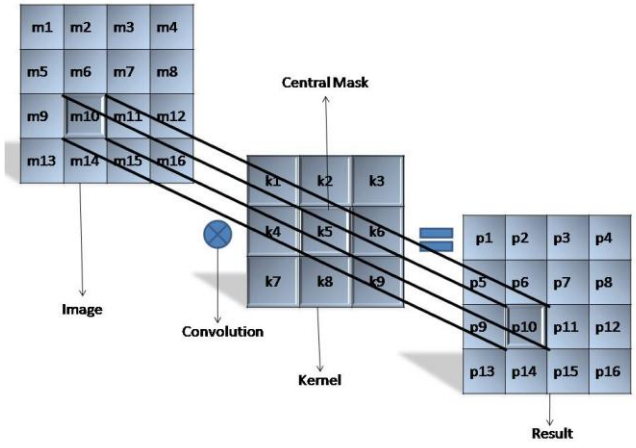

Fig. 1. Convolution of image and Kernel matrices.

Fig. 1 shows the convolution of the image and the kernel matrices, here image represents the 4×4 matrices and kernel matrices represents the 3×3 and resultant image represents the 4×4. The below equation (2) represents how the convolution operation should be performed.

$$P10 = (m5 \times k1) + (m6 \times k2) + (m7 \times k3) + (m9 \times k4) + (m10 \times k5) + (m11 \times k6) + (m13 \times k7) + (m14 \times k8) + (m15 \times k9). \quad (2)$$

Based on this get the p10 convolution resultant values. The same procedure repeats, finds the p1, p2, p3, p4, p5, p6, p7, p8, p9, p11, p12, p13, p14, p15, p16. When doing the convolution operation image resolution increases, mean multiplication and additions increases, implies that propagation delay also increased. So design the effective architecture for the 2D convolution. Let assume kernel matrices are 3×3, 9×9, 12×12, here kernel matrices increases implies that multiplication and additions will be increases means hardware cost will be increased.

III. PROPOSED ARCHITECTURE AND HARDWARE IMPLEMENTATION FOR 2D CONVOLUTION

Fig. 2 shows the convolution of the proposed architecture. The proposed architecture is a combination of the control block read and control block write and pipelining shift registers and ALU block. Based on the shift registers intend to move the pixel values one by one and each register consists of eight bits. In the proposed architecture the Pipelined shift register 1 and the pipelined shift register 2 sizes based on the image resolution. For example, if the image resolution is 4×4, it implies that the pipelined shift register 1 should be allotted six registers and the pipelined shift register 2 also have the same six registers and each register have eight bits. The pipelined shift register 3 always encompass only three registers.

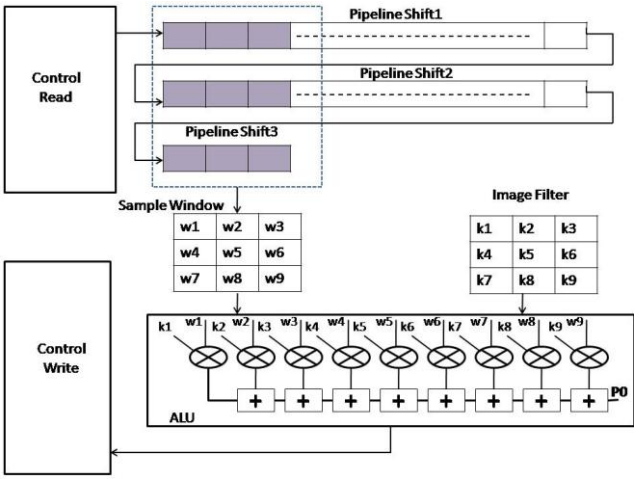

Fig. 2. The proposed architecture for the 2D convolution.

The allocation of the registers in the pipelined shift register 3, is independent of the image resolution because the kernel matrix dimension is 3×3. Let us assume that the kernel matrix the order 9×9, the inference being that time pipelined shift register 3 consists of nine registers and each register has the eight bits. So concluding that the pipelined shift register 3 register allocation is based on the kernel size rather than the image resolution. Let consider another example suppose where the image resolution is the order 16×16, which shows that both the pipelined shift register 1and pipelined shift register 2 are allocated with eighteen registers and each register have the eight bits. Thus conclude that for every completion of the image line the addition of two zeros is imperative. Line buffers instead of frame buffers, because the former significantly improve the performances and reduces the memory required compared to the latter. In the architecture, there are three main blocks one of them is ALU. ALU consists of adders, multipliers. Fig. 3 shows the read control block. Control block read consists of the ROM memory, mux, and counter. The ROM memory serves the purpose of storing the image pixel values prior to performing the convolution.

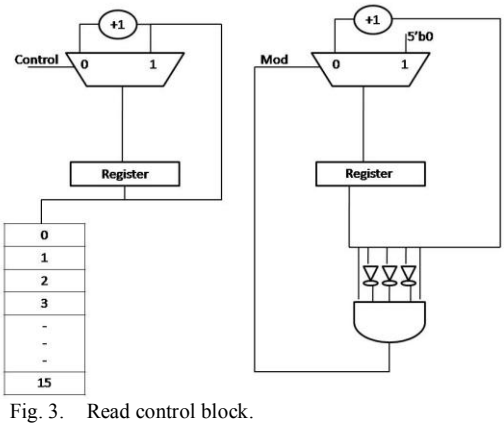

Fig. 3. Read control block.

Read control block consists of a count block and mod count block. Further, the count block consists of a ROM, mux, register, and memory. For example, let consider an image having an order of 4×4 matrices it implies that there is a total of 16 pixels stored in the memory. When the read

these pixels from the memory it will be done in line by line manner.

1	**2**	**3**	4
5	**6**	**7**	**8**
9	**10**	**11**	**12**
13	**14**	**15**	**16**

Fig. 4. Image resolution matrix.

Let consider the example here, see the Fig. 4 image resolution having the 4×4 resolution. First line completion of the reading the pixels from the image pixel 1 2 3 4, have to add two zeros in the pipelined shift registers and again read from the second line 5 6 7 8 after completion of which have added two zeros in the shift registers. Repeat the same process till the end of the reading all image pixels in the memory. In order to add two zeros, so need the read control block as shown in Fig. 3. In the read control block, the left-hand side is count block and the right-hand side is mod count block. In mod count block AND operation is performed because at the instant of completion of the line to terminate the reading of the pixel values from the memory. Here count and mod count blocks consist of the MUX. Using MUX add the two zeros in after each completion of the line. Now after reading the sixth address (image 4×4 resolution) from ROM memory, the convolution operation will commence. After this operation, to store the resultant value in the memory. After the operation of the first line, to wait for two cycles because added two zeroes prior.

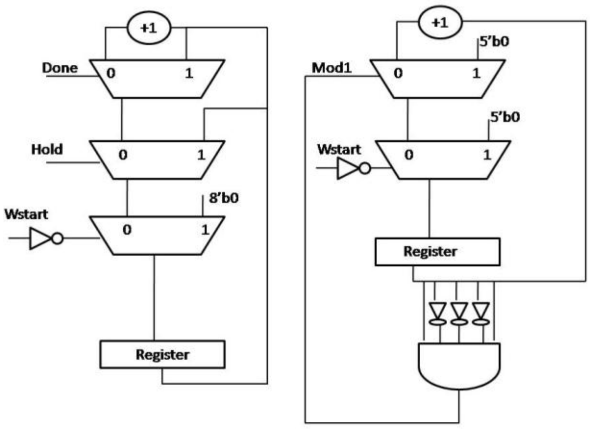

Fig. 5. Write control block

Again repeat the same procedure and store the second line results and the same process will be repeated until the completion of all the lines. After completion of all the line to stop writing pixel values in the memory, this is achieved by using the write control block. Fig. 5 shows the write control block.

IV. MATLAB RESULTS

Below filtering operation can be implemented using the convolution kernel matrices.

0	**0**	**0**
0	**1**	**0**
0	**0**	**0**

Fig. 6. All pass filter for image kernel

Fig. 6 represents all pass filter image kernel, it is also called the identity filter. Here property of the identity filter is whatever input image have, that image apply to this filter get same resultant input image after completion convolution operation. Here test image Fig. 7 size is 16×16, and also see the resultant (Fig. 8) of the convolution of the pixel values.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 7. Input image (16×16) pixel values

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 8. Outputs of convolution (16×16) stored in RAM.

V. SIMULATION RESULTS

The proposed architecture is implemented on Spartan 3E XC3S100E-5CP132 devices with the 16×16 image resolution, and kernel 3×3 matrix.

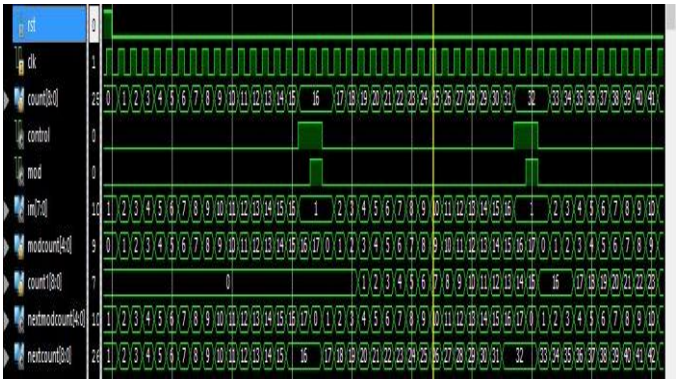

Fig. 9. Read control block outputs.

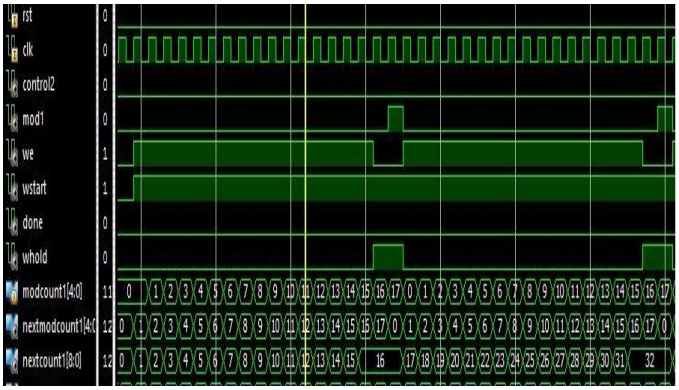

Fig. 10. Write control block outputs.

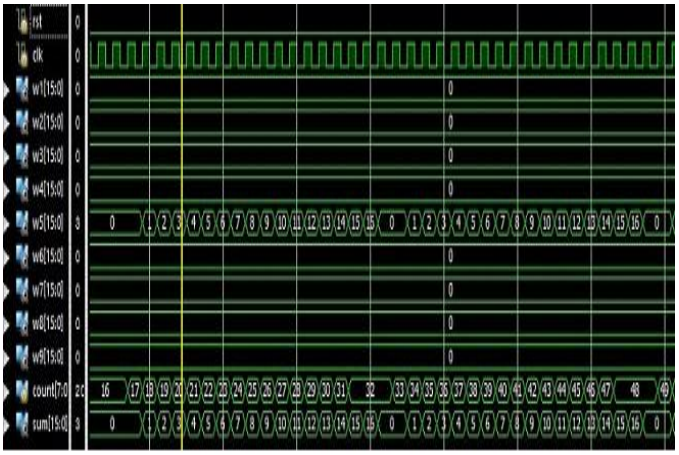

Fig. 11. ALU block outputs.

TABLE. I. Devices Utilization Summary

Device Utilization Summary			
Logic Utilization	Used	Avalable	Utilizaton
Number of Slice Flip-flop	46	1920	2%
Number of 4 input LUTs	49	1920	2%
Number of occupied Slices	35	960	3%
Number of bonded IOBs	18	83	21%

VI. PERFORMANCES COMPARISON

In this paper discuss the hardware implementation of 2D convolution in Verilog. The proposed hardware architecture is shown Fig. 2 was modeled in Verilog and synthesized using FPGA, this device named as Spartan 3E XC3S100E-5CP132. In this devices, critical path delay reduced more compared with the existing architecture and also requires the less number of slices. Here, considered the image of order 16×16 resolution the propagation delay was found to be 4.040 ns. This delay will be increased when the image resolution like 720 × 480 or 600 × 560. The differences between the time delays of convolution architecture resolution 16 × 16 images negligible, but when the image resolution has more will save a lot of computational time. Here after completion of the convolution operation, the resultant of the convolution values stores in the memory.
The below table shows the improvement in results obtained using the proposed architecture compared to existing architecture [5].

TABLE. II. Maximum output time required after clock performance of the existing [5] and proposed architecture.

Parameter		**Value**
Maximum output time required after the clock	Existing	5.962 ns
	proposed	4.040 ns
Maximum output frequency required after the clock	Existing	167 MHZ
	proposed	247.524 MHZ

VII. CONCLUSION

The design proposed in this paper a new architecture for the implementation of 2D convolution. It is simple and efficient hardware implementation, consisting of only a few multiplexers, ALU blocks and control blocks. Implementation on Xilinx and Spartan 3E, it is verified that area reduction achieved and critical path delay also reduced in this devices.

.

REFERENCES

[1] Gonzales and Woods, "Digital Image Processing", Pearson Education, India, Third Edition.

[2] Jamro, E.; Wiatr, K., "Convolution operation implemented in FPGA structures for real-time image processing," Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on 2001, pp.417,422.

[3] Wiatr, K.; Jamro, E., "Implementation image data convolutions operations in FPGA reconfigurable structures for real-time vision systems," Information Technology: Coding and Computing, 2000. Proceedings. International Conference on 2000, pp.152,157.

[4] S. Di Carlo, G. Gambardella, M. Indaco, D. Rolfo, G. Tiotto, and P. Prinetto, "An area-efficient 2-D convolution implementation on FPGA for space applications " *2011 IEEE 6th International Design and Test Workshop (IDT)*, Beirut, 2011, pp. 88-92.

[5] Sriram V.B., Prasad Sawant, Kavit Kamath, Kanika Wadhwa, Ganesh Gore, Sneha Revankar, "Implementation Of 2D Convolution Algorithm On FPGA For Image Processing Application", International Journal of Electrical, Electronics, and Data Communication (IJEEDC), Volume-3, Issue-12, 2015, pp 22-25.

[6] F. Cardells-Tormo, P. -. Molinet, J. Sempere-Agullo, L. Baldez, and M. Bautista-Palacios, "Area-efficient 2D shift-variant convolvers for FPGA-based digital image processing *" International Conference on Field Programmable Logic and Applications, 2005.*Tampere, 2005, pp. 578-581.

.