IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 4, Issue 1, Ver. II (Jan. 2014), PP 75-80
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Design and Implementation of Neighborhood Processing Operations on FPGA using Verilog HDL

K. Babu Ravi Teja, Abhilash S. Warrier, Akshay S. Belvadi, Dhiraj R. Gawhane
VLSI Design, Visvesvaraya National Institute of Technology, Nagpur, Maharashtra, India

Abstract: *This paper discusses the image enhancement techniques by a hardware custom processor implemented on FPGA using VERILOG Hardware Description Language. Using the moving window filter mechanismthe processor will perform Low Pass Filtering (Smoothening), High Pass Filtering (Sharpening), High Boost Filtering (Edge-Detection) and Zooming. We use grayscale image to demonstrate the effectiveness of the processor. The processor achieves image enhancement in spatial domain through Neighborhood processing operations. The hardware processing of the image is advantageous in terms of speed of operation and parallel processing as against classical software simulations.*

Keywords:*Digital Image Processing, Image Enhancement Technique, NeighborhoodProcessing, Moving Window Filter, VERILOG, FPGA.*

I. INTRODUCTION

Real-time digital image processing usually requires high throughput rate and large amount of operations to be performed. To cater to this need, parallel processing in the form of custom hardware or multiprocessing in the form of software task are therefore indispensable. For a particular application, hardware solutions are always preferable for their lower system cost. In this work, we propose hardware architecture to perform a few selected neighborhood processing operations. The purpose of designing sucharchitectureis to implement neighborhood image processing as fast as possible with the resources currently available[4]-[5].

Over the past few years, programmable devices like field programmable gate arrays (FPGA), complex programmable logic devices (CPLD) and application specific integrated circuits (ASICs) have became more and more important for digital image processing. In this work, we are using FPGA to implement and test our design. To program these devices we use Hardware Description Language (HDL). Since, HDL enables description of circuits precisely; it provides an opportunity for the designers to optimize speed of operation from the available timing constraints [1],[10].

Image enhancement aims at improving the interpretability or perception of information in image for subjective measure. Image enhancement techniques can be broadly divided into two categories: spatial domain techniques and frequency domain techniques. The spatial domain technique refers to enhancement of image based on operations performed directly on the pixels of the image. Frequency domain techniques achieves enhancement through the use of mathematical transforms such as Fourier transforms [8]. Also, the operations performed on the image can be classified into two categories; Point Operation and Neighborhood Operation. In this work, we are focusing on Neighborhood Operations.

II. NEIGHBORHOODOPERATIONS

These operations are very common in image processing and the computations are done in spatial domain. Unlike point operations, the neighborhood operations perform modification of pixel value depending on the selected pixel andits neighboring pixels. This operation requires a mask to define which pixels are to be considered for processing. The shape (square or cube) of the mask depends on the image dimensions.Since we are operating ona gray scale (2D) image, we use square masks for processing. The size of the mask should be '*N x N*' where '*N*' is an odd number. In this work, for the purpose of computation, we use '*3 x 3*' mask window.The element values of the mask vary with each image enhancement operation.

The algorithm for performing neighborhood operation is as follows;

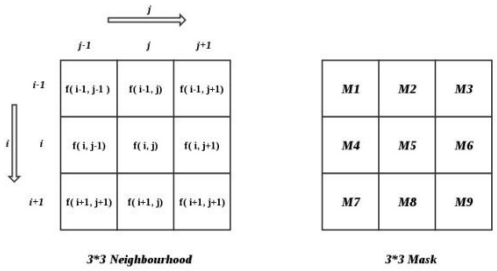


Fig 1: Neighborhood Processing

- To achieve neighborhood processing we place the mask '*M*' in Fig.1 on an image neighborhood '*f*'. Initially, the mask is placed such that the center of the mask *M5* coincides with pixel *f(i,j)*. Each element of the mask is multiplied with the corresponding value of the image.
- After the summation of the computed products, we replace the original center pixel valuef(i,j)with the newly computed value*g(i,j)*. Where *g(i,j)*is computed as:

$$g(i,j) = f(i-1,j-1)*M1 + f(i-1,j)*M2 + \ldots + f(i,j)*M5 + \ldots + f(i+1,j+1)*M9 \quad (1)$$

- Once *g(i,j)* is computed, the mask is shifted by one pixel such that the center of the mask *M5* coincides with *f(i,j+1)*.The procedure is repeated till the last pixel of the image.[8]
-

2.1. LOW PASS FILTERING

This removes the high frequency content from the image. This reduces the noise present in the image data as noise is normally high frequency signal.Generally, the image background is low frequency region whereas edges are high frequency region due to steep transitions in grayscale. By performing low pass filtering, we can smoothen out the edges in the image. Usually, an image is smoothed by decreasing the disparity between pixel values by averaging nearby pixels. Thus, this operation is also termed as smoothening.

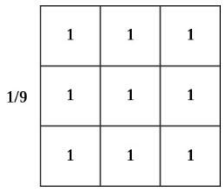
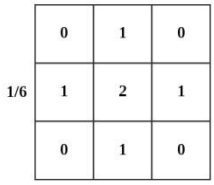


Fig 2a: Low Pass Mask



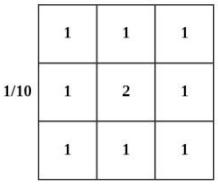


Fig 2b: Alternative Low Pass Mask

The standard '*3 x 3*' low pass averaging maskis as shown in Fig. 2a. This mask is normally used when the image contains Gaussian noise. For illustration in this particular work, we have used the same considering that the image has Gaussian noise.Some of the other averaging filter masks are as shown in Fig. 2b

2.2. HIGH PASS FILTERING

This operation eliminates the low frequency content while retaining or enhancing the high frequency components. A high-pass filtered image would have no background as background is of low frequency. It would have enhanced edges. We know that, sharpening an image increases the contrast between bright and dark regions to bring out features.The enhanced edge is obtained by sharpening of image.Thus, this operation is also termed as sharpening.

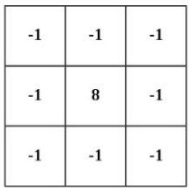
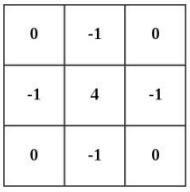


Fig 3a: High Pass Mask



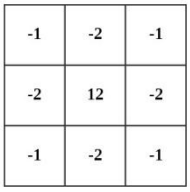


Fig 3b: Alternate High Pass Mask

The standard '*3 x 3*' high pass filtering mask is as shown in Fig. 3a. The mask window for sharpening usually contains a single positive value at its center, which is completely surrounded by negative values.It is required that the summation of the coefficients of the high pass filter mask must be 'zero'.

2.3. HIGH BOOST FILTERING

It is often desirable to emphasize high frequency components of the imagewithout eliminating low frequency components. In such a case, the high-boost filter can be used. In High Boost Filtering, a part of the background is retained unlike High Pass Filtering wherein only edges are enhanced.

The '*3 x 3*' high boost filtering mask is as shown in Fig. 4. It is required that, in accordance with the mask in Fig.4,the summation of the coefficients of the high pass filter mask must be 'unity'.

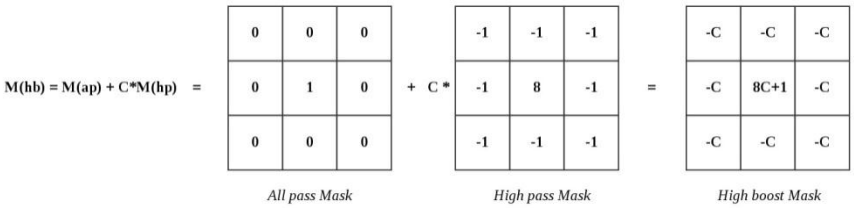


Fig 4: High Boost Mask generation

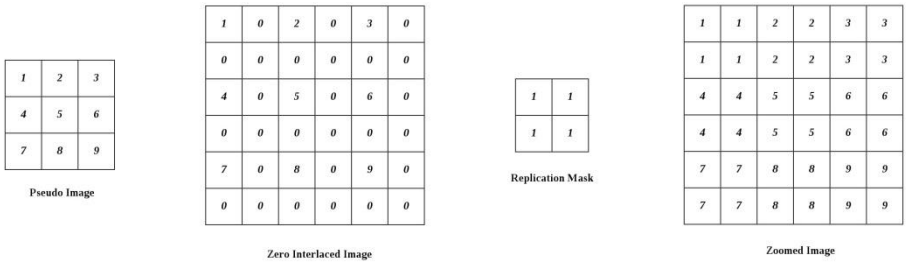
Where, '*M(hb)'* is the high boost filter mask, '*M(ap)'* is the all pass filter mask, '*M(hp)'* is the high pass filter mask and '*C'* is a constant such that $C \epsilon N$ and approx.$C \leq 5$, for better subjective measure.

2.4. ZOOMING

This operation can be performed by using two different techniques:

- Zooming by Replication.
- Zooming by Interpolation.

In this work we use Zooming by Replication technique. This technique is faster and requires less memory, but the trade-off being that Zooming by Interpolation is more effective in subjective measure.



Illustrated as in Fig. 5, in Zooming by Replication, depending on the measure of zoom, zero interlacing is done to increase the image size proportionately. Further on, convolution is performed with a replication mask to obtain a zoomed image. Replication mask is a standard '*2x2*' mask as illustrated in Fig 5.

III. DESIGN OF THE IM-PRO IIARCHITECTURE

The architecture of the proposed processor Im-Pro II is as shown in the Fig 6. The three main sub blocks of the processor are Memory Unit, Control Unit and the Processing Unit. A detailed description of each of these blocks is provided in the following subsections. The external world has an option to access the 8-bit data line, clock which can be used to synchronize the processor with any external device to which it is to be interfaced, op-code which chooses the operation to be performed on the image, Read Write controller which says about writing or reading the memory. After processing is done, the data is stored with in the processor and can be read at any point of time.

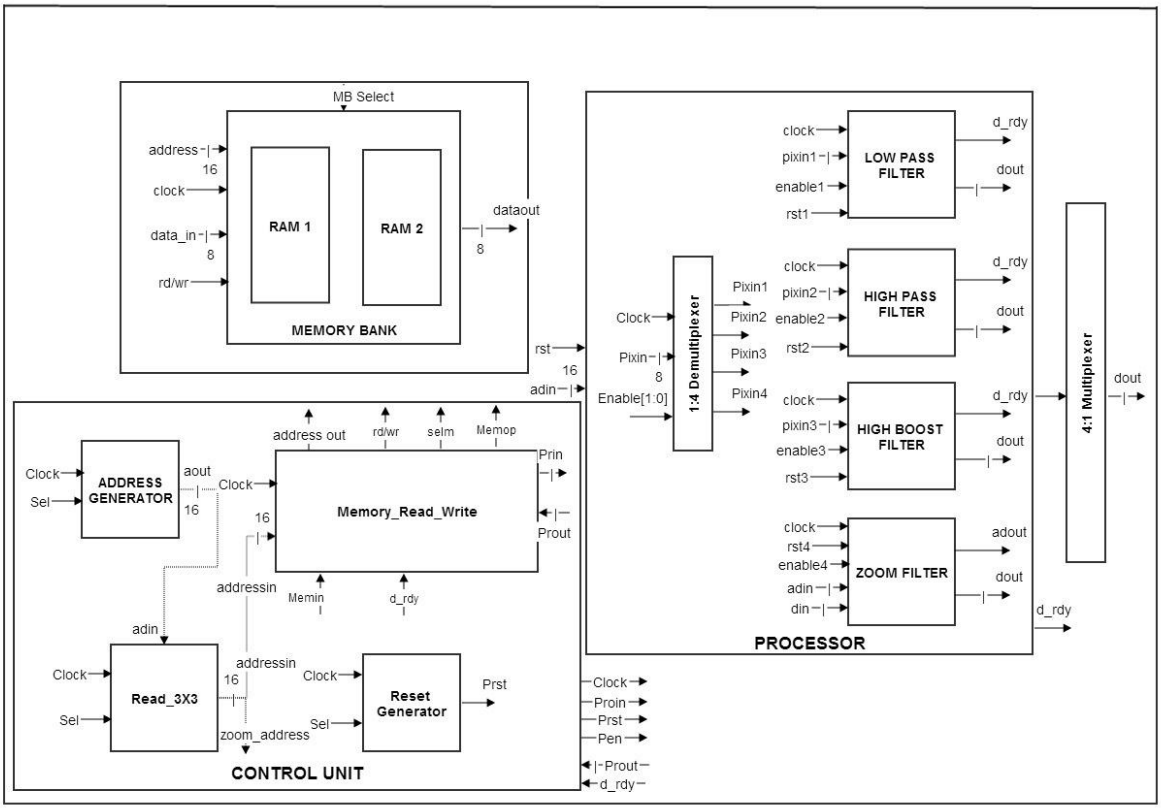


Fig 6: Im-Pro II Architecture

2.5. MEMORY UNIT

The memory unit is capable of storing data upto 128KB. This memory space is logically divided into two equal partitions such an image of size 256x256 can be handled with ease. Each pixel is addressable and can be varied without disturbing any of its neighbors. The convention used in this design is that the lower half of the address space referred to as 'Memory Bank 1' is used to store the input image while the upper half is used to store the image formed after processing of the given image.

The process of storing the image into the memory referred to as loading consumes N^2 clock cycles for an '*NxN*' image. The same is the case while retrieving the processed output. Though this appears to be a sluggish design, it is inevitable, keeping in mind the restriction on the number of IOBs.

2.6. CONTROL UNIT

This is the most crucial sub-block responsible for controlling the flow of data within the processor. The sub-blocks in this unit are reset generator, address generator, pixel reader and memory interface. Address Generator generates the neighbor addresses which are fed to the memory interface block responsible for obtaining the data from the input image stored in '*Memory Bank 1'* of the memory unit. Processed output is written back into the '*Memory Bank 2'.* A memory select signal is used to choose between the two banks.

The reset generator is a pulse generator which is issued periodically to clear the results of the previous unit operation. Address Generator will seed the initial address or the address of the center pixel to the pixel reading module, which in turn generates the neighborhood addresses and issues them serially to the memory interfacing module. Thisthen supplies the '*pixel reader*' with the pixels storedin the increasing order of their addresses. These values are then fed to the Neighborhood Processing unit.

2.7. NH PROCESSOR

This is the major sub-block of the design responsible for the hectic number crunching on the input image. This processing unit has four sub-modules viz. LPF,HPF,HBF and Zooming. As the name suggests these modules performs the low pass filtering, high pass filtering, high boost filtering and the zoom operations respectively.

LPF is implemented using a mask discussed earlier.For efficient implementation type two mask of size '*3x3*' is employed. Each clock cycle one pixel is supplied from the control unit as discussed above. These input pixels are multiplied with the respective weights.Here the logic shift operation is used for hardware efficiency

and speed improvement. The summation of these outputs is done with the help of a 12-bit adder. The output is divided by '8' using the shift right operation which is equivalent rounding of the output to the lower bound.

HPF and HBF are also similarly implementedusing the masks discussed in the earlier sections. The implementation is modified such that the summation of all the pixels excluding the center pixel is obtained first. Then, this value is subtracted from the weighted-center pixel value using the 2's complement arithmetic. The carry of this operation is used as a flag for checking the negative output which will be subsequently rounded to zero. The three operations discussed till now will need 9 clock cycles, one clock cycle per pixel. A flag d_rdy (data ready) is used to indicate the readiness of the output.

Zooming is achieved by using a simple strategy of repeating the pixel in its three neighbors which is achieved by generating these addresses and copying the pixel into these locations. So this operation needs four clock cycles.

IV. IMPLEMENTATION AND RESULTS

Similar to our previous work, as presented in paper [1], we will be using RAM's to store and access the data. In this work, we are performing neighborhood processing, original image and processed image must be separately stored unlike in point processing operations. Since we are working on grayscale images, we require two RAM's to store data.

In all the neighborhood processing operations discussed earlier, the pixels at the image edges are inaccessible to the mask as the mask center cannot overlap with the pixel. These pixels are not considered during processing. The RAM is initialized to 'zero' before processing, due to which a black boundary is formed around the processed image (in 8-bit grayscale format, '0' is black and '255' is white). The data lost at the edges can be neglected since they do not contain any necessary information as compared to the pixels at the center of the image.

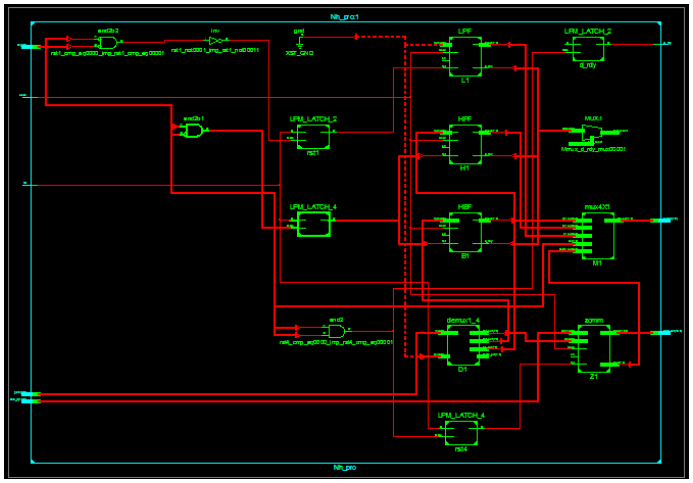


Fig 7: RTL Schematic of Processing Unit – Im-Pro II

The designed Im-Pro II processor is implemented using VERILOG HDL. Design, synthesis and simulation are done using Xilinx ISE Design Suite 14.2 and hardware emulation is done on Spartan-3E FPGA from Xilinx.

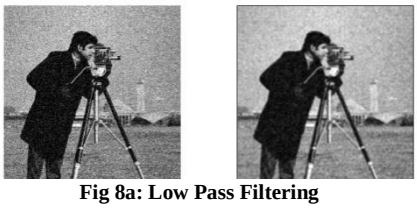


Fig 8a: Low Pass Filtering

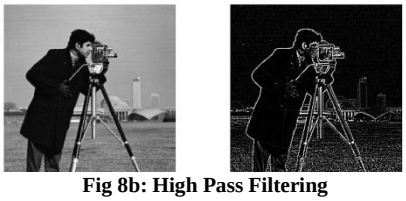


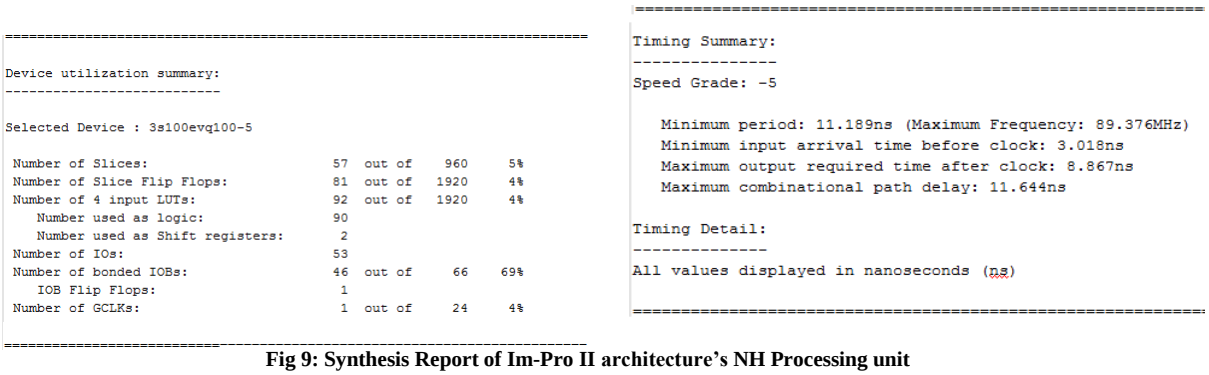
Fig 8b: High Pass Filtering

Fig 8c: High Boost Filter

Fig 8d: Zooming by Replication

In the set of figures Fig. 8a, 8b, 8c and 8d, the images on the left are of the original image and the ones on the right are the processed images. All the images are of size 256x256 pixels. In case of zooming, the original image of size 128x128 is considered which after processing generates a 256x256 processed image for a zooming of 200%.

2.8. SYNTHESIS REPORT



```
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s100evq100-5

 Number of Slices:                      57  out of    960     5%
 Number of Slice Flip Flops:            81  out of   1920     4%
 Number of 4 input LUTs:                92  out of   1920     4%
    Number used as logic:               90
    Number used as Shift registers:      2
 Number of IOs:                         53
 Number of bonded IOBs:                 46  out of     66    69%
    IOB Flip Flops:                      1
 Number of GCLKs:                       1  out of     24     4%

-------------------------------------------------------------------------
```

```
=========================================================

Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 11.189ns (Maximum Frequency: 89.376MHz)
   Minimum input arrival time before clock: 3.018ns
   Maximum output required time after clock: 8.867ns
   Maximum combinational path delay: 11.644ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================
```

Fig 9: Synthesis Report of Im-Pro II architecture's NH Processing unit

Fig. 8 shows the synthesis report of NH Processing unit of the proposed Im-Pro II architecture. The VERILOG coding was done using structural modeling. This reduces the inferences of extra circuitry during synthesis.The number of Flip-Flop and Latch inferred were greatly reduced owing to the use of structural modeling. This solves the latching problem which is very prominent in behavioral modeling.

V. CONCLUSION AND FUTURE SCOPE

A novel architecture design, dealing with the image processing paradigm effectively and efficiently is proposed. This architecture is tailored for the neighborhood operations for filtering application. Other neighborhood operations can be easily modeled by changing the mask sets. This architecture does not support floating-point operations, which are required for complex operations such as Laplacian filtering and advanced transforms.However, using of floating-point numbers will add much more flexibility and versatility for the processor at the cost of increased area, power and design complexity. A floating-point coprocessor can be implemented in conjunction with the existing architecture for performance enhancement.Parallel processing and pipelining technique will increase the speed of computation at the cost of increased area and power.

ACKNOWLEDGEMENTS

The Authors would like to thank the Department of Electronics Engineering, Visvesvaraya National Institute of Technology, Nagpur for Technology Support.

REFERENCES

[1] Dhiraj R Gawhane et al ,*Int.J.Computer Technology & Applications 2014,Vol 5 (1),87-91.*
[2] Guangda Su; Jiongxin Liu; Yan Shang; Boya Chen; Shi Chen, "*Theory and application of image neighborhood parallel processing," Image Processing (ICIP), 2009 16th IEEE International Conference on , vol., no., pp.2313,2316, 7-10 Nov. 2009*
[3] Rao, DagguVenkateshwar, et al. "*Implementation and evaluation of image processing algorithms on reconfigurable architecture using C-based hardware descriptive languages." International Journal of Theoretical and Applied Computer Sciences 1.1 (2006): 9-34.*
[4] Nelson, A. E. (2000). *Implementation of image processing algorithms on FPGA hardware* (Doctoral dissertation, Vanderbilt University).
[5] Chan, S.C.; Ngai, H.O.; Ho, K.L., "*A programmable image processing system using FPGA," Circuits and Systems, 1994. ISCAS '94., 1994 IEEE International Symposium on , vol.2, no., pp.125,128 vol.2, 30 May-2 Jun 1994*
[6] Koenderink, J.J.; Van Doorn, A.J., *"Generic neighborhood operators," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.14, no.6, pp.597,605, Jun 1992*
[7] Lougheed, R. M., &McCubbrey, D. L. (1980, May). *The cytocomputer: A practical pipelined image processor. In Proceedings of the 7th annual symposium on Computer Architecture (pp. 271-277). ACM.*
[8] Gonzalez, R. C., Woods, R. E., &Eddins, S. L. (2009).”*Digital image processing using MATLAB*” (Vol. 2). Knoxville: Gatesmark Publishing.
[9] Burger, Wilhelm, and Mark James Burge. “*Principles of digital image processing: core algorithms*” Vol. 2. Springer, 2009.
[10] Padmanabhan, T. R., &Sundari, B. B. T. (2008). “*Design through Verilog HDL*”. John Wiley & Sons.