Gray-scale Image Blurring On FPGA part of AMD Xilinx Zynq-7000 All Programmable SoC

Darshil Shah
Department of Electronics and Communication Engineering
Nirma University
Ahmedabad, India
22BEC111

***Abstract*—This paper presents an efficient implementation of image blurring algorithms on Field Programmable Gate Arrays (FPGAs), leveraging their parallel processing capabilities to achieve high performance. We are designing parallel hardware for convolution so that image blurring can be fast from traditional sequential processor.**

***Keywords*—FPGA, Multiplication and Accumulation, SoC, Blurring, Image Processing, Gray-scale image, Direct memory access(DMA), AXI interconnects**

I. Introduction

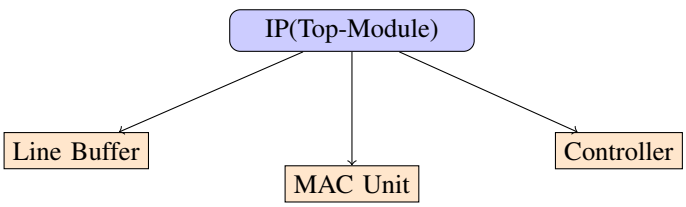
Image processing has become an integral part of modern technological applications, ranging from medical imaging and autonomous vehicles to surveillance systems and augmented reality. Among Image processing technique image blurring is very important technique for both practical and aesthetic purpose. Blurring helps reduce noise, enhances image quality which important for after processing.

Traditional software-based approaches to image blurring often struggle to meet the demands of real-time processing especially with the increasing resolution of images. because of sequential nature of general purpose processor, image progressing through software is slower. In contrast, FPGA is faster because of their inherent parallelism. If we implement efficient hardware on FPGA, it can significantly accelerate processing tasks, making them ideal for applications requiring high throughput and low latency.

II. Hardware Design of image processing IP

In our IP design there are mainly three components for image blurring

- 1.Line Buffer
- 2.MAC(Multiplication and Accumulation) unit
- 3.Controller



A. Line Buffer

Line buffer is simple FIFO(First In First Out) which acts like small memory unit which stores the 1 row of pixel data. So if our image is 512x512 then its store 512 pixel data, and in output it output kernel size(only row) of data in one shot. Here we are using 3x3 kernel in future so our Line Buffer output is 3 pixel data.

B. Multiplication and Accumulation unit

MAC(Multiplication and Accumulation) unit is the main operating part of IP. MAC unit is doing the actual convolution between our image pixel data and our kernel data. So in our MAC unit's input will be total 3x3 pixel data and its output will be 1 pixel which is convoluted pixel with our kernel. Here we are using pipelining for better performance because first multiplication should be done then accumulation can be made,

An average filter replaces each pixel's value with the average of the values in its neighborhood. For a 3x3 average filter, the kernel is defined as:

$$K = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

We are using this filter in our MAC unit.

C. Controller

As per the name this unit Controls the operations of our IP. its input is one pixel and output will be 3x3 pixel data from our line buffer. In Controller there is basically three(In our IP 4 for better performance) line buffer instantiated. Basically this block controls when should pixel data can go to MAC unit and when should not. We are also using interrupt for when should our ip is ready to output convoluted pixel data and when it not.

D. Top Module

Our IP is using AXI (Advanced eXtensible Interface) Stream interconnect for communication between another part of block design (here DMA). So if we talk about IP's Ports:

- clock
- reset
- interrupt

Slave interfaces:

- inputData
- inptutDataValid
- outputDataReady

Master interfaces:

- outputData
- outputDataValid
- inputDataReady

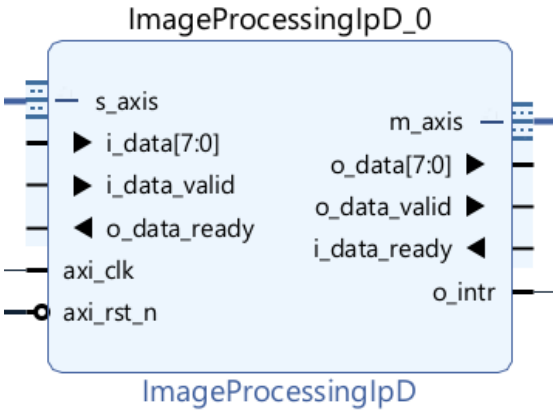


Fig. 1. Image Blurring IP

III. System Design

We can divide our whole process into 3 parts:

1) Sending our image from host to Soc
2) Image Inversion By Hardware
3) Receiving Image from Soc to host

For sending image from our host to soc we are UART. Now UART is controlled by PS part of SOC. Also PS part is storing our whole image in RAM.

Now for performing Image blurring we need our image's data in our IP which is in PL part. so that we are using ACP port in this case for communication between PS-PL.(We can use other port also like HP or GP port). also we are using DMA(Direct Memory Access) as intermediate between PS and our IP because of better performance and using AXI stream interface. (PS can communicate with Memory Map device which in this case is DMA and DMA can communicate with Stream device also which in this case is our IP.)

For receiving image from SOC to our host we are using UART. UART is controlled by PS part of SOC.

So basically image sending and receiving is happening because of PS part and main image blurring is happening in PL part.

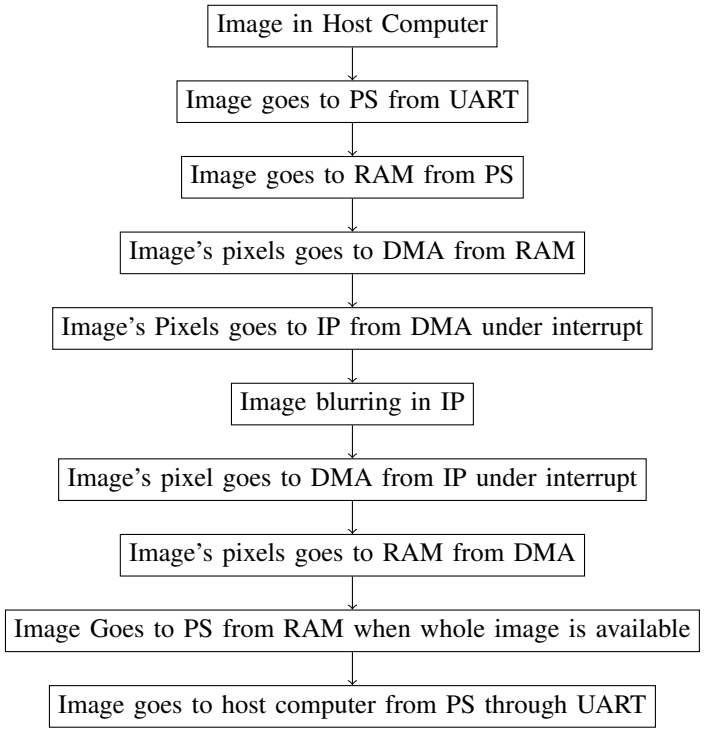


Fig. 2. Flowchart of Image Blurring

IV. Block Design

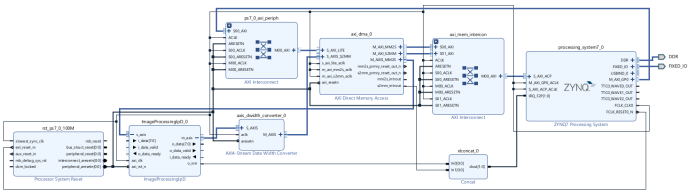


Fig. 3. Block Design for image blurring

Here there are total five AXI(Advanced eXtensible Interface)interconnections are used.

1) Master AXI-stream memory mapped to stream:For writing data from DMA to IP
2) Slave AXI-stream stream to memory mapped:for writing data from IP to DMA
3) Master AXI stream to memory mapped:for writing data to memory from DMA
4) Master AXI memory mapped to stream:writing data to DMA from memory
5) Slave AXI lite:configuration of DMA from processor

V. Results

Fig. 4. Original Image

Fig. 5. Blurred Image

VI. Summary

This paper discusses the implementation of image processing techniques using Field Programmable Gate Arrays (FPGAs) to achieve efficient and high-performance image blurring through convolution. The methodology typically involves designing a system that uses a 3x3 kernel to apply an average filter or similar blurring techniques to images.

The FPGA implementation allows for parallel processing capabilities, which significantly enhance the speed of the image blurring operation compared to traditional CPU-based approaches. The paper outlines the design flow, including the architecture of the system, the use of Direct Memory Access (DMA) for efficient data transfer between memory and the image processing unit, and how data is handled within the FPGA environment.

VII. Further Topics to be covered

- Real time Image Processing
- Image blurring with another filters
- Image edge detections
- Comparison between software and hardware approach (speed and performance)

References

[1] D. Bailey, Design for Embedded Image Processing on FPGAs, Wiley-IEEE Press, 2011.
[2] Xilinx, Inc., "Zynq 101" webpage.
[3] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 3rd Edition, Pearson, 1998.
[4] ZedBoard website.