

Real Time Hardware Accelerator for Image Filtering

Susana Ortega-Cisneros¹, Miguel A. Carrasco-Díaz¹, Adrian Pedroza de-la-Cruz¹
Juan J. Raygoza-Panduro², Federico Sandoval-Ibarra¹, and Jorge Rivera-Domínguez²

¹ CINVESTAV, Unidad Guadalajara, Av. Del Bosque 1145, Col. El Bajío,
Zapopan, Jalisco, C.P. 45091, México

{sortega, mcarrazco, apedroza, sandoval}@gdl.cinvestav.mx

² Department of Electronics, CUCEI, Universidad de Guadalajara, Blvd. Marcelino García
Barragán 1421, Guadalajara, Jalisco, C.P. 44430, México

{juan.raygoza, jorge.rivera}@cucei.udg.mx

Abstract. The image processing nowadays is a field in development, many image filtering algorithms are tested every day; however, the main hurdles to overcome are the difficulty of implementation or the time response in a general purpose processors. When the amount of data is too big, a specific hardware accelerator is required because a software implementation or a generic processor is not fast enough to respond in real time. In this paper optimal hardware implementation is proposed for extracting edges and noise reduction of an image in real time. Furthermore, the hardware configuration is flexible with the ability to select between power and area optimization or speed and performance. The results of algorithms implementation are reported.

Keywords: Image filtering, image convolution, edge detecting, noise reduction, field-programmable gate array, FPGA, hardware accelerator, hardware design, high performance computing, image kernel, Verilog.

1 Introduction

One of the main study fields of signal processing is image processing. With the current technology available to capture digital images, it is easy to get a good image quality, nevertheless the problem now is obtaining useful information about the image contents as a detection of objects, persons, is to get relevant information in the images.

There are many techniques to filter images and to extract information. However, the algorithms are commonly implemented in software, but the hardware is not optimized to get the best performance while the process is being executed, resulting in high power consumption per image processed. Much time is required in order to obtain the transformation per image, and when the frames are back to back, many systems are not capable of giving the required throughput for real time processing.

This paper proposes hardware architecture with specific acceleration in order to obtain filtered images in real time, with minimal power consumption, allowing data streaming with the ability to process one image after another.

2 The Convolution Filter

The convolution is one of the most common operations used in image processing, but also one the most demanding computations, when an image is processed. The convolution is an image transformation, in which it is necessary to calculate each of the pixels in the array in order to form a new image based on the source pixel and its neighborhood. When the kernel size increases, the complexity increases as well.

When the convolution filter is applied, a new smooth, fuzzy or blurry image is obtained, it is possible to obtain edge detection. The convolution filter is also called convolution kernel because there exists a singular matrix to obtain different image transformation [1, 2].

$$K = \begin{bmatrix} k_{m-1,n-1} & k_{m-1,n} & k_{m-1,n+1} \\ k_{m,n-1} & k_{m,n} & k_{m,n+1} \\ k_{m+1,n-1} & k_{m+1,n} & k_{m+1,n+1} \end{bmatrix}$$

Where K represents a kernel matrix.

$$P = \begin{bmatrix} P_{m-1,n-1} & P_{m-1,n} & P_{m-1,n+1} \\ P_{m,n-1} & P_{m,n} & P_{m,n+1} \\ P_{m+1,n-1} & P_{m+1,n} & P_{m+1,n+1} \end{bmatrix}$$

Where pixel P{m,n} and its 8-neighbors are represented in matrix P.

$$C = \left| \frac{\sum_{m=-p}^q \left(\sum_{n=-p}^q k_{m,n} P_{m,n} \right)}{\sum_{m=-p}^q \left(\sum_{n=-p}^q k_{m,n} \right)} \right| \quad (1)$$

In order to obtain the pixel array of the target image, equation (1) must be calculated for each source pixel [3].

Since image filtering is realized by means of a linear function that performs the convolution function C[m,n]. A given transformation is applied with a fixed kernel K[d,d] of d size to each pixel P[m,n] of a given image I[x,y], with a pixel array of x rows and y columns. The C transformation of equation (3) must be done for each pixel that belongs to each image to be filtered. This case allows the design of a fixed kernel architecture and a mechanism to handle consecutive images as an infinite data stream.

3 Architecture of the Hardware Accelerator

The architecture is composed of a data feeder, a data collector, the buffer storage, the control unit, and the image convolution machine [4]. Before the operation starts, the configuration parameters are written into the control unit registers, enabling the control unit to wait for the start signal. After a start signal is received, each clock period, while exist valid data in the input port, many image rows are stored in each buffer as the size of the kernel matrix. When all buffers are full, the control unit proceeds to fill the calculation matrix and the first partial results are obtained. After the Kernel size clock cycles for each consecutive clock cycle, one resulting pixel is delivered by the accumulator pipeline as an output pixel. The full picture of the hardware architecture is shown in Fig. 1. Each part is explained in detail below.

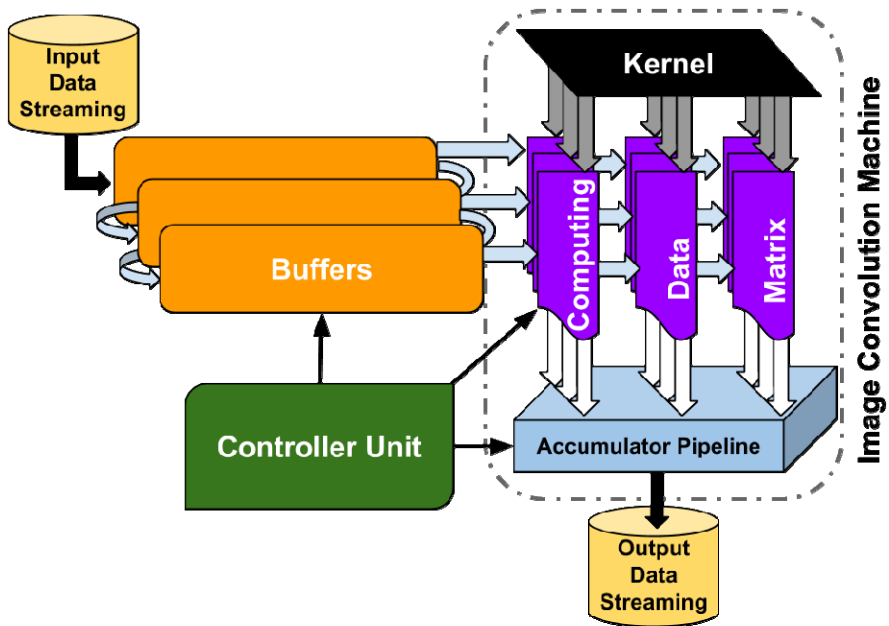


Fig. 1. Architecture of the hardware accelerator

3.1 Data Buffer

The data streaming comes from the source image, each row from top to bottom and each pixel from left to right in each line. In order to calculate the target pixel the transformation matrix and the source pixel with its neighbors is required. The amount of concentric neighbors is directly proportional to the size of the kernel. Therefore, as many buffers as kernel size are required, with at least as much capacity as elements in row for each buffer. It is important to have a semicircular feedback line at the output of each buffer to hold data to feed the next buffer.

3.2 Image Convolution Machine

The image convolution machine is composed of the kernel matrix [5], the computing matrix, and the accumulation pipeline [6]. The kernel matrix has the coefficients of the filter for each image [7]. The computing matrix is formed by a network of multipliers and accumulators as the arithmetic part of the transformation [8]. The accumulation pipeline contains the registers to store the initial values, partial accumulations and the final result. It is important to note that the system is able to deliver an output pixel for each clock period and its latency is the magnitude of the kernel size. Fig 2 a) shows the organization of the functional units, as well as the array of multipliers and accumulators, and the enough registers to store partial results for next iterations. Fig 2 b) shows the pixel matrix of an example image and one characteristic kernel. Fig 2 c) shows the contents of the buffer, when the first pixel is completely calculated and the second and third is partially accumulated for a matrix kernel of size 3.

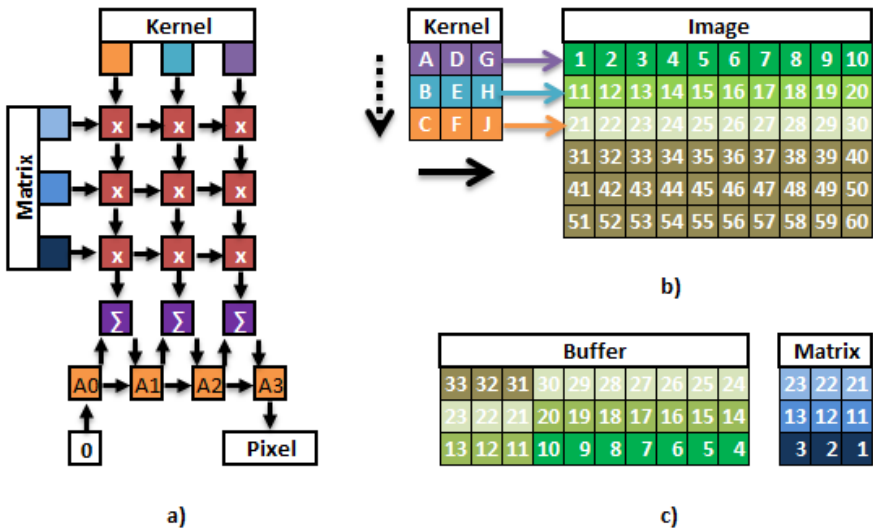


Fig. 2. Process steps for convolution filter. a) Functional units for multiplication, accumulation and partial results. b) Kernel and image computing representation. c) Pixel arrangement in buffers and matrix.

3.3 The Control Unit

The control unit is responsible for maintaining the correct functionality in time and order of each module in the accelerator. This module is composed of a finite state machine, counters, and comparators that help to drive the control lines of the entire system. The finite state machine is composed of 5 states: the state "IDLE" is the idle state and is the default state when the system comes out of reset, and when a new image is loaded into the input port. The state "BUFFER" is responsible for the proper

functionality of the buffers during the process of data streaming storage. The "WAIT" state is reached in case the data streaming is interrupted, paused or the data feeder system indicates that data are invalid. The functionality of the image convolution machine is performed while the system is in "STREAM" state. When an image is fully processed, the "STOP" state is reached and the system is cleaned to be ready for process the following image. Fig 3 shows the state diagram of the Finite State Machine (FSM).

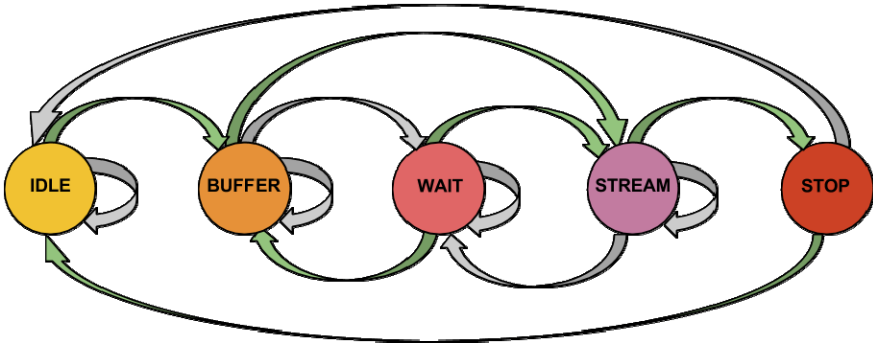


Fig. 3. Finite State Machine (FSM)

4 Implementation

The logic utilization of the FPGA Altera Stratix V with part number 5SGSMD5K2F40C2 is shown in Table 1.

Table 1. FPGA logic utilization

FPGA Components	% Utilization	Component Utilization
Logic utilization ALMs	1 %	1981 / 172,600
Total registers	1 %	2712 / 172,600
Pins	24 %	206 / 864
Block memory bits	9 %	3,670,016 / 41,246,720
DSP block 18-bit elements	1 %	6 / 1590

For each image size, the processing time is linear.

$$T = (K \cdot R + L + K + 8) \left(\frac{1}{F} \right) \tag{2}$$

The processing time for any image is calculated by the equation (2) where K is Kernel size, R is the image rows, L is the length and F is the operation frequency. The operation frequency range is 303.4 MHz for 85° C and 318.37 MHz for 0° C.

5 Results

The image convolution with different kernels matrix were tested and the original images are shown in Table 2. Each original image has 1024x720 pixels and the time to process each image is the same: 2.44 ms [9].

Table 2. Original Images

$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ <p>Original</p>		
---	---	--

In order to implement an edge detection kernel, it is necessary that the sum of all matrix elements results is zero, with at least two elements different from zero. Table 3 shows different levels of detection.

Table 3. Edge detection effect













$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ <p>Edge detect low</p>		
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ <p>Edge detect middle</p>		
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} E$ <p>dge detect High</p>		

Table 4 shows an image with sharpened enhancement, and low and high blur effect. Almost all random kernel matrixes produce some level of sharp or blur image transformation [10].

Table 4. Sharpen and blur effects

$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 8 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ <p>Sharpen</p>		
$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ <p>Blur Low</p>		
$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 10 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ <p>Blur High</p>		

6 Conclusions

In the output images it is easier to see the relevant information and the hardware cost is the same as for each of the different transformations.

The power consumption, utilized area, and the time spent on the process of filtering in the hardware accelerator is less expensive than software processing, allowing image filtering in real time.

This work includes the design, implementation, and verification of the hardware accelerator. All code is parameterized, enabling the data bus reconfiguration and the kernel size in a short period of time.

In the implementation, the slowest and most expensive hardware module was the division, but the solution is to include a 2-step pipeline in order to achieve improved performance and higher operation clock frequency.

Another characteristic is that it is possible to route the accelerator several times in the same FPGA with the same or a different kernel array, taking advantage of the parallel structure, performing the same process on different images, different processes to the same image, or both. The only restriction is the maximum memory available in the device.

Acknowledgments. Financial support provided in part by CONACyT (National-Council of Science and Technology) as a doctoral fellowship.

References

1. Daza, S.M., Vega, F., Matos, L., Moreno, C.T., Diaz, M.L., Daza, Y.M.: Image encryption based on convolution operation in the gyrator transform domain. In: IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, pp. 1527–1529 (2012)
2. Zhu, X., Li, X.: Image kernel for recognition. In: 9th International Conference on Signal Processing, ICSP 2008, pp. 1620–1624 (2008)
3. Li, B., Hu, J.: Design and Implementation of Image Effects Based on Convolution Algorithm. In: 2013 Fifth International Conference on Computational and Information Sciences (ICIS), pp. 144–147 (2013)
4. Chan, S.H.: Constructing a sparse convolution matrix for shift varying image restoration problems. In: 2010 17th IEEE International Conference on Image Processing (ICIP), pp. 3601–3604 (2010)
5. Odone, F., Barla, A., Verri, A.: Building kernels from binary strings for image matching. *IEEE Transactions on Image Processing* 14, 169–180 (2005)
6. Carlo, S.D., Gambardella, G., Indaco, M., Rolfo, D., Tiotto, G., Prinetto, P.: An area-efficient 2-D convolution implementation on FPGA for space applications. In: 2011 IEEE 6th International Design and Test Workshop (IDT), pp. 88–92 (2011)
7. Mori, J.Y., Llanos, C.H., Berger, P.A.: Kernel analysis for architecture design trade off in convolution-based image filtering. In: 2012 25th Symposium on Integrated Circuits and Systems Design (SBCCI), pp. 1–6 (2012)
8. Hartung, S., Shukla, H., Miller, J.P., Pennypacker, C.: GPU acceleration of image convolution using spatially-varying kernel. In: 2012 19th IEEE International Conference on Image Processing (ICIP), pp. 1685–1688 (2012)
9. Russo, L.M., Pedrino, E.C., Kato, E., Roda, V.O.: Image convolution processing: A GPU versus FPGA comparison. In: 2012 VIII Southern Conference on Programmable Logic (SPL), pp. 1–6 (2012)
10. Ketan, T., Au, O.C., Yuanfang, G., Jiahao, P., Jiali, L., Lu, F.: Arbitrary factor image interpolation by convolution kernel constrained 2-D autoregressive modeling. In: 2013 20th IEEE International Conference on Image Processing (ICIP), pp. 996–1000 (2013)