EE 244 Final Project Report

Image Processing In VHDL

Oğuzhan İzgi
Ahmet Furkan Akıncı

Project Advisor: Şenol Mutlu

07.06.2023

Table of Contents

İçindekiler

1 INTRODUCTION

This report explores the implementation of kernel filtering using VHDL (VHSIC Hardware Description Language) and the display of the results on a VGA (Video Graphics Array) monitor. Kernel filtering, a critical concept in image processing, involves modifying an image based on a specific pattern or a "kernel".

2 PROBLEM STATEMENT

In the rapidly developing field of digital image processing, high-speed, real-time image filtering is increasingly required in various applications, including video surveillance, medical imaging, and graphic design. Most image filtering processes are executed through software-based solutions, which may not meet the demand for real-time image processing due to their potential latency and slower processing speeds.

Hardware-based solutions, such as FPGA (Field Programmable Gate Array) implementations, can provide a faster alternative. However, the efficient and effective design of kernel filters using VHDL Language and the display of results on a VGA (Video Graphics Array) monitor pose significant challenges.

The problem lies mostly in storing an image in the limited sized memory of the FPGA board and synchronizing the process such that each pixel is created in a clock cycle without any significant delay that affect the result.

3 RELATED BACKGROUND

The images indeed consist of pixel and intensity values. Each pixel value has three intensity value which indicates colors of this pixel: red, green, blue respectively. For instance, if a pixel is yellow, the intensity value of red and green is maximum, that of blue is minimum.
For processing images, we use kernel filters. Kernel filters is simply a matrix. Each filtering process has a special kernel filter. Images are processed with kernel filter through convolution operation.

4 DESIGN

4.1 Definitions, Acronyms and Abbreviations

Kernel Filter: A kernel filter, also known as a convolutional kernel or filter, is a small matrix of weights used in convolutional neural networks (CNNs) for image processing tasks A kernel filter is applied to image input for cropping different objects and shapes from the drawing.

Dual Port ROM: A Dual Port ROM (Read Only Memory) is a type of memory device that features two separate access ports for reading data: a pair of address input lines, and a pair of data output lines. This dual-port functionality allows two independent devices or processors to access the ROM concurrently without causing conflicts or needing to wait for the other to complete its operation.

The dual port ROM is typically used in systems that require high-speed data access or in applications where simultaneous, independent data reads are necessary. Dual-port ROMs can contribute to the overall efficiency and performance of such systems.

4.2 VGA Driver

First of all, it is supposed to display an image through VGA display. In order to do this, the raw data of the image, which consists of intensity values of each pixel respectively is needed. It is desired to process images rapidly, so images are processed in grayscale, that is to say, red, green and blue intensity values are same. In this way, much less hardware power is required in the project.

In order to display the image, signals called horizontal and vertical synchronization signals are needed. The synchronization signal consists of two regularly occurring negative pulses. The negative pulses on the horizontal synchronization signal indicate the beginning and end of a line, guaranteeing that the monitor accurately displays the pixels within the visible screen area boundaries. To generate these signals accurately, counters called h_count and v_count are used. The value h_count controls horizontal signal, v_count controls vertical signal. They control these signals according to values in the table below:

25 MHz pixel clock and 60 Hz ± 1 refresh						
Symbol	**Parameter**	**Vertical Sync**			**Horizontal Sync**	
		Time	**Clocks**	**Lines**	**Time**	**Clocks**
T_S	Sync pulse time	16.7ms	416,800	521	32 µs	800
T_{DISP}	Display time	15.36ms	384,000	480	25.6 µs	640
T_{PW}	Pulse width	64 µs	1,600	2	3.84 µs	96
T_{FP}	Front Porch	320 µs	8,000	10	640 ns	16
T_{BP}	Back Porch	928 µs	23,200	29	1.92 µs	48

Figure 1 Signal time intervals

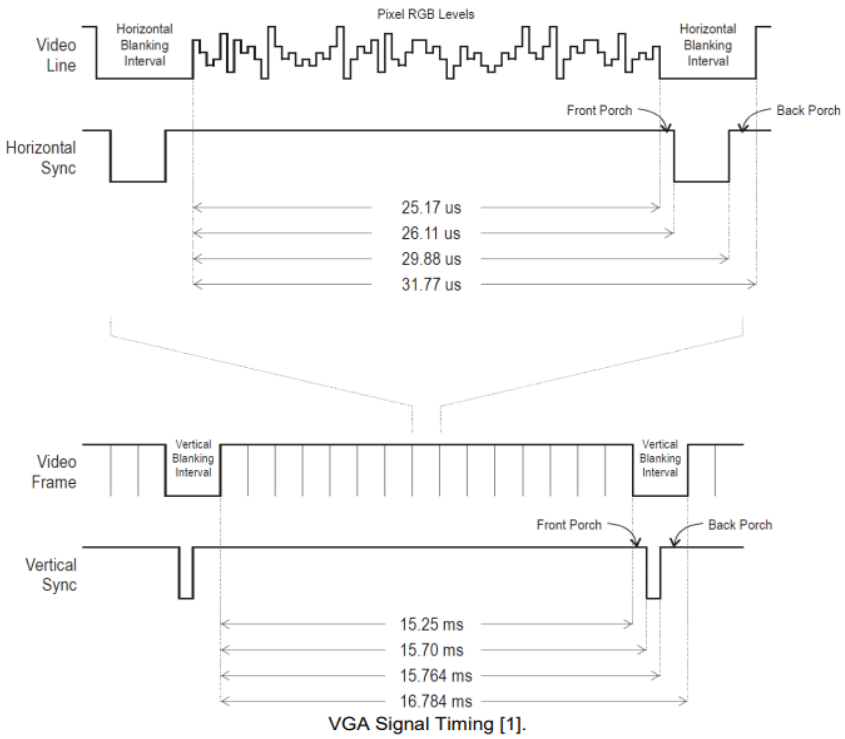

Figure 2 Synchronization signals

In this table, the time intervals of each signal and the values corresponding to h_count and v_count values are demonstrated. These h_count and v_count values are calculated for 25 MHz VGA clock signal. H_count and v_count values also show which pixel is being displayed. This data is useful for displaying image.

During the process pixels in the data is obtained by indexing method. For indexing the data, the h_count and v_count values which are mentioned above are used. These intervals are longer than display time, because these values count not only within the displaying interval, but also the intervals which are called front porch, back porch etc. This situation is demonstrated in the figure below:

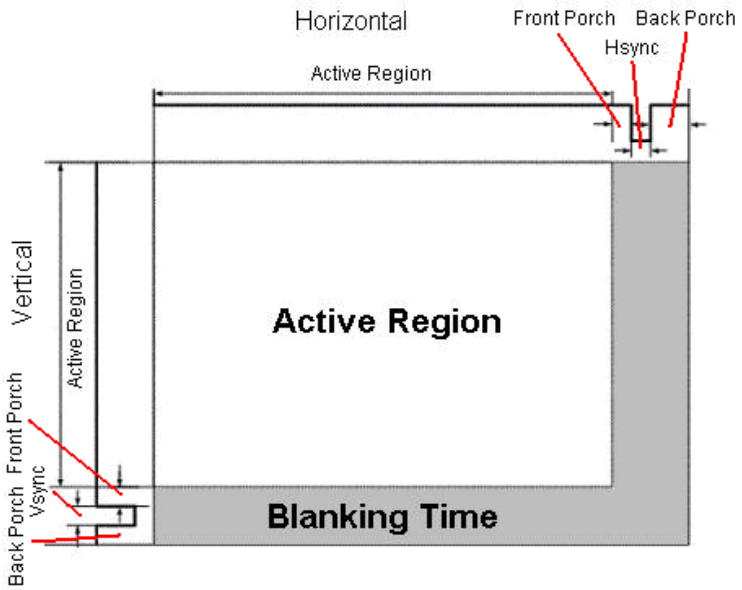

Figure 3 Signal intervals

As indicated in the figure above, h_count and v_count counts for all pulses. For h_count: display region (640 clock cycle), front porch (16 clock cycle), pulse width (96 clock cycle), and back porch (48 clock cycle). For v_count: display region (480 clock cycle), front porch (10 clock cycle), pulse width (2 clock cycle), and back porch (29 clock cycle). The signals are generated as pulse width, back porch, display region and front porch respectively. In this pattern, our h_count and v_count values are 144 and 32 at the beginning of display region respectively. Thus, to use these values to obtain pixel intensities from our data, we should subtract 144 and 32 respectively. In this way, our pixel values can be determined accurately. For example, in the first pixel, our h_count and v_count values are 144 and 32 respectively. If these values are subtracted 144 and 32 respectively, they will be zero, which indicates the index of first pixel.

4.3 Dual Port Rom

To store images in the device, block memory fields of *SPARTAN6* is utilized. To initialize a kernel at least two pixels were required. By using dual port ROM, two pixels can be obtained from the memory at the same clock cycle. In order to create dual port ROM components, core generator application of *Xilinx* is used. Memory fields are initialized with *.coe* files.

Contents of the *.coe* files are created by using a python script that just simply resizing the given image to monitor resolution and listing the pixel intensity of each pixel starting from the most top left pixel to the most down right pixel.

4.4 Image Processor

After obtaining each pixel values in an appropriate order, a kernel filter is used to process image. To filter image with kernel filter, convolution operation is used. This process is shown below:

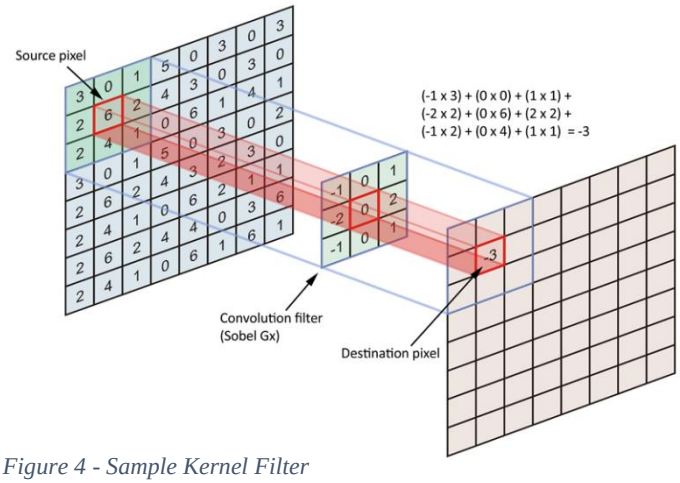

Figure 4 - Sample Kernel Filter

1. Place the kernel filter on top of a pixel in the image.
2. Multiply each value in the kernel with the corresponding pixel value in the image.
3. Sum up the multiplied values.
4. Replace the original pixel value with the summed value obtained in the previous step.
5. Slide the kernel to the next pixel in the image and repeat steps 2-4.
6. Continue this process until the kernel has covered the entire image.

By convolving the kernel filter on the image, we obtain a new image called convolved or filtered image. The resulting image highlights features depending on the type of kernel. For example, a kernel designed for edge detection can emphasize the edges of an image, while a blurred kernel will smooth the image by averaging nearby pixel values.

After filtering the image, it is displayed on the screen.

5 RESULTS

After implementing the desired algorithms, a variety of images were tested, with edge detection kernel. The system was generally able to apply filters and display the results in real time for standard VGA resolution 640x480.

The results obtained from a sample image are shown in the below images:

Figure 5 - Original Image

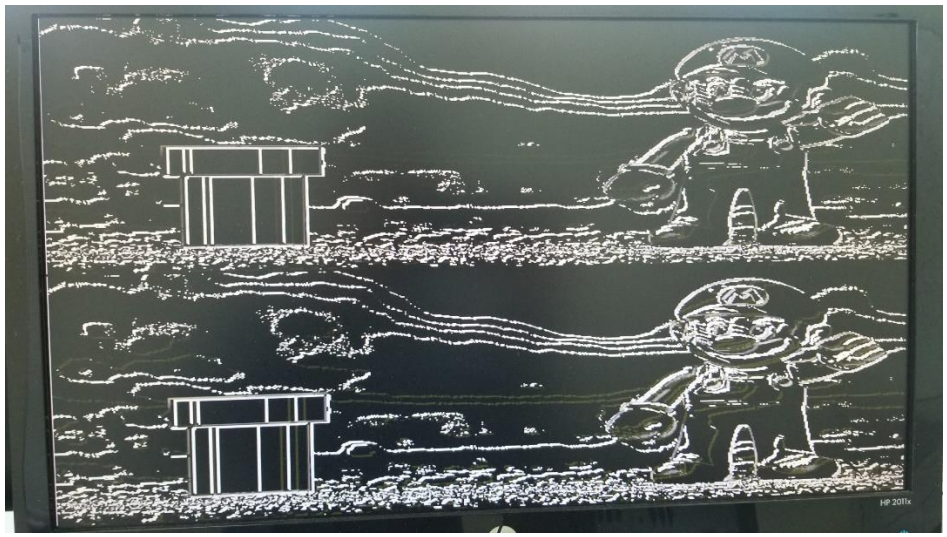

Figure 6 - Filtered Image (edge detection)

6 CONCLUSION

This project demonstrated the potential of VHDL for image processing applications and the feasibility of displaying the processed image data on a VGA monitor.

The future work can involve applying more complex filters and using 9 port memory or equivalent system that can reach 9 pixels of the image at the same time.

7 REFERENCES

1. **https://setosa.io/ev/image-kernels/**

2. **http://www.ue.eti.pg.gda.pl/fpgalab/zadania.spartan3/zad_vga_generacja_i_uzywanie_pamieci_rom_en.html**