COL215: Hardware Assignment 3

Ansh 2022CS51135 Raj 2022CS51643

November 15, 2023

Contents

1 Approach

The objective of this assignment is to implement an image filtering operation on a given image using a specified kernel and display both the original and modified images on a VGA monitor. The key components include generating ROM/RAM using Vivado's memory generator, carrying out the filtering operation, and utilizing an FSM for program flow control.

The first module, primarily responsible for Multiplier-Accumulator (MAC) unit, has been tailored to function like an adder for the 9 values of kernel cells into one cell after a filter operation. It collaborates seamlessly with a Read-Only Memory (ROM) unit, which is powered by COE files, and efficiently transfers the generated gradient data into Random-Access Memory (RAM). To substantiate the correctness and accuracy of our edge detection process, rigorous simulations have been conducted.

The second module, known as the 'Base' module, plays a pivotal role in controlling VGA timing, specifically generating the essential 'Hsync' and 'Vsync' signals. These signals are instrumental in orchestrating the correct positioning of the displayed content within the designated display region. To achieve this, a clock divider module is employed, diligently updating horizontal and vertical counters, effectively determining the coordinates of each pixel within the display.

The image filtering logic involves the use of Finite State Machine (FSM) to control the MAC unit and perform data transfers between ROM/RAM and the local registers. This FSM ensures the proper flow of operations during the image filtering process.

It is important to emphasize that the 'Base' module's code is expertly crafted to handle VGA parameter synchronization. In summary, our approach demonstrates a proficient and systematic handling of VGA timing and pixel display control. The seamless synergy of these modules ensures the optimal display of video content on a VGA monitor, marking a significant milestone in VGA system design. This report details our design, implementation, and testing processes.

2 Memory Generator

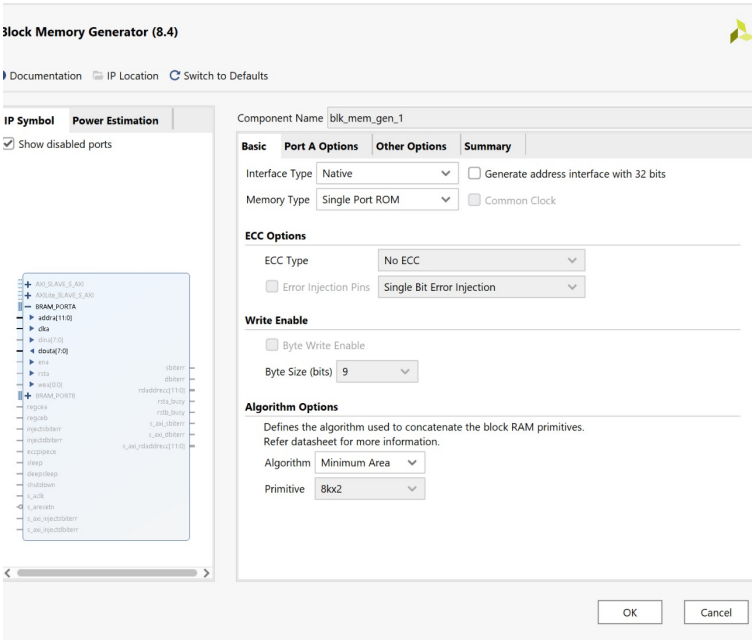

Figure 1: Block Diagram for ROM generator

$$New\_I(i,j) = (I(i,j) - old\_min) * \frac{new\_max - new\_min}{old\_max - old\_min} + new\_min \quad (1)$$

After computing the output image, we need to search for the minimum and maximum pixel values, and normalize the pixel values based on these values. For the given normalization, the new minimum is 0 and new maximum is 255, leading to the computation shown in Equation 2.

$$New\_I(i,j) = (I(i,j) - min) * \frac{255 - 0}{max - min} + 0 \quad (2)$$

Figure 2: formula for Normalization

2.1 Generation

We configured the memory with specific depth 4096 and width 8, and observed an increasing change in the number of Look-Up Tables (LUTs) used, indicating the impact of our memory settings. This provides insights into the resource utilization and efficiency of our memory design.

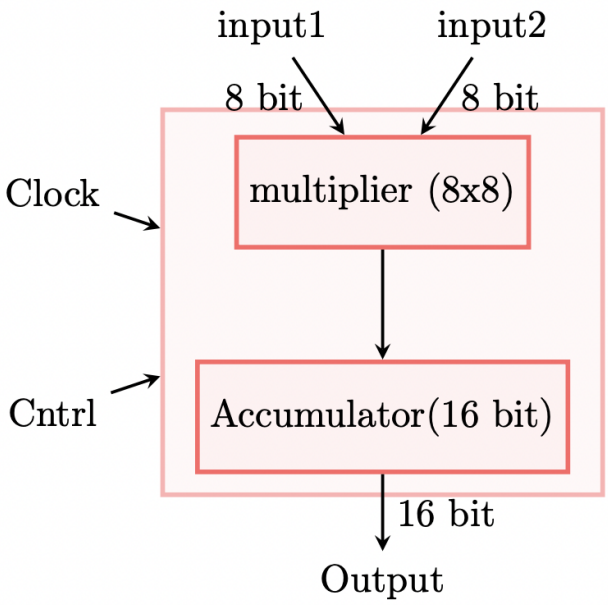

Figure 3: MAC unit flow

3 FSM Image Filter Operation and VGA Module controller

3.1 Filter Operation FSM Logic

We design Filter module for kernel filtering and real-time display. This module employs memory components (RAM and ROM) to process data using MAC multiplication and normalization operations. To do this we use FSM essentially to control the pixel choice and filtering from mode(0-8) as filtering is for 3x3 kernel. Then we apply normalization to finally range in the output of the operation in desirable 0-255 range. FSM helps in choosing the control flow of the pixel based on previous values inside the ROM as we don't want to update the pixel value yet. For MAC unit, we take signed veectors instead of taking integer to bascially facilate multiplication of signd*signed which is easy in vhdl. Then we proceed by accumulating value on every rising clock edge. for the MAC unit, we initialize a 16 bit acvumlating value sort of like a register to control addition using signal bit on plusclkedge. We know range is -127 to 127, so we take 8 bit vector.

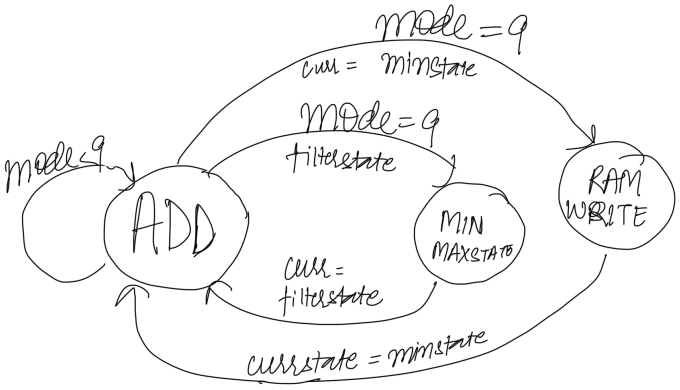

Figure 4: FSM Diagram

3.2 Base Module

We've designed a VHDL 'Base' module for our VGA system, orchestrating precise VGA timing, pixel generation, and video display control. This module features meticulously defined input and output ports for managing VGA parameters and a 'pixel-tick' counter aka Clock divider. A 'RegisterState' process ensures accurate 'hsync' and 'vsync' signal generation. The 'Vid' process governs 'video-on' to align with the VGA display area. By skillfully connecting internal signals to vital output ports, our code flawlessly manages VGA timing and video display, making it a pivotal element in rendering video content on a VGA monitor. Simulation hsync and vsync comes out to be 0 in the display regions, because 1 is low and 0 is high.

3.3 Display Module

Through edge detection, we derive convolution values for each memory address. These convolution values represent the intensity changes in the image, a critical aspect of edge detection. Subsequently, we utilize the Base module to position the pixels accurately on the display by controlling the horizontal and vertical coordinates.

To clarify, the RGB (Red, Green, Blue) values start as identical, but we reduce their bit depth from 8 bits to 4 bits by discarding the least significant 4 bits. This process is integral to the image processing pipeline. In our code snippets, particularly within the ROM module, you can observe how we verify the correctness of the Base module's functionality by evaluating and processing the data using the convolution values. This helps ensure the accurate positioning and presentation of image content on the display.

4 References

- `https://embeddedthoughts.com/2016/07/29/driving-a-vga-monitor-using-an-fpga/`
- `https://www.ece.ualberta.ca/~elliott/ee552/studentAppNotes/1998-w/Altera-UP1-Board-Map/vga.html`

5 Images, Synthesis Report, Displays, and Diagrams

Figure 5: Waveforms for part 3

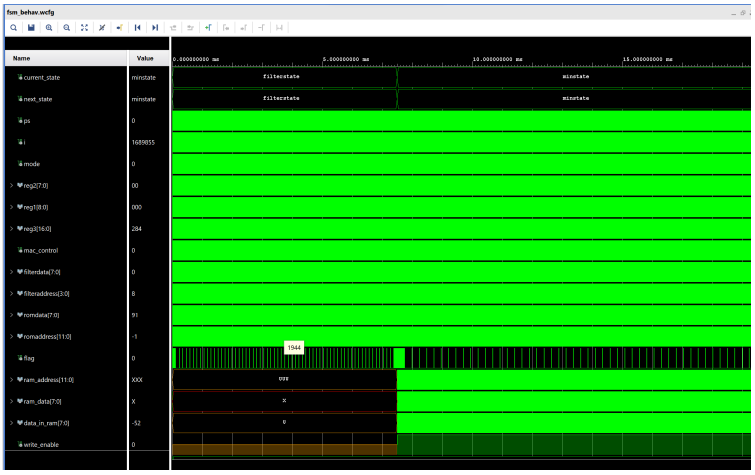

Figure 6: Waveforms

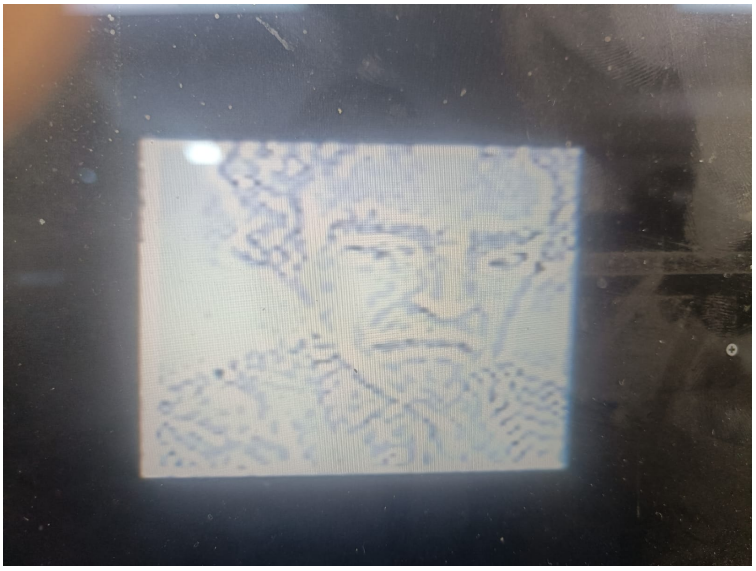

Figure 7: Results