Real Time Imaging and Control, Supervised by ***Prof. Julien Dubois***

2D IMAGE FILTERING USING VHDL ON FPGA

Botros Karim, Tsagatakis Ioannis
Universite de Bourgogne - uB
kim0.swimmer@gmail.com, jtsagata@gmail.com

Project Overview

This Project is about doing a 2D real time filtering using VHDL **"Very High Speed Integrated Circuit Hardware Language"** on Nexys4 Field programmable Gate Array, This task may seem too trivial,but on the contrary it is very crucial process nowadays, Only FPGAs, ASIC, or Digital Signal Processors (DSP) can filter image in real time. filtering images in real time can be used by many applications in many fields such as industry, medicine, and military applications. that is why we need Digital signal processors or FPGAs Using such technology is a must in non-delay tolerant applications.

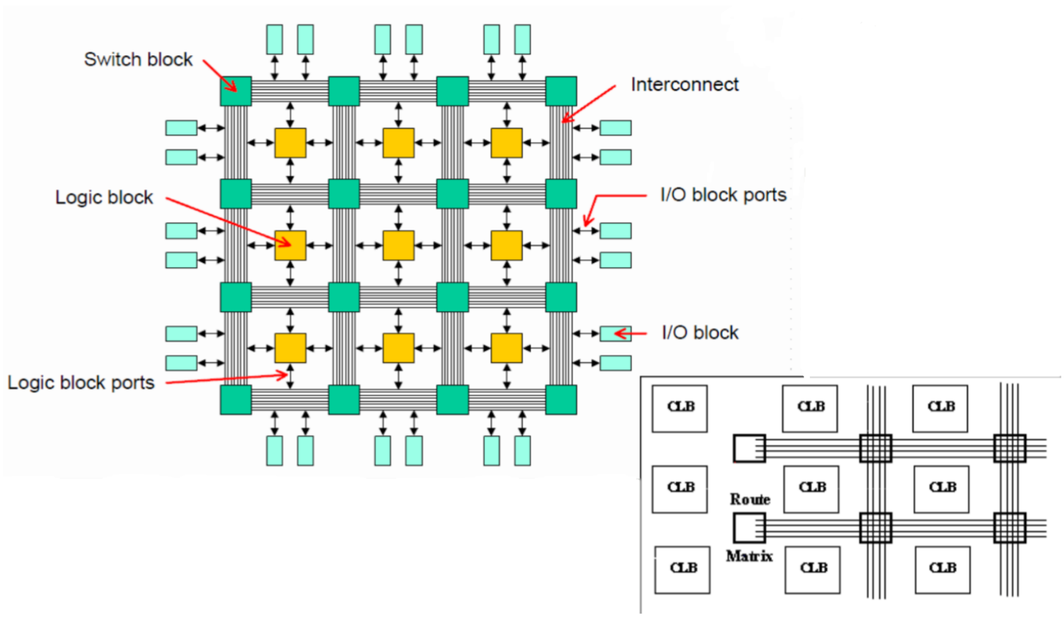


Figure 1: FPGA Structure

FPGA is a group of logical arrays which gained alot of popularity in the real-time applications, since it is durable and effcient in real time and of-course has parallel processing, in which it may compute many logical operations (logical gates) in parallel, resulting in saving alot of time compared to serial processing or other processing techniques. the disadvantage of FPGA is that it has fixed number of gates that can be used, so its architecture is less flexible than other processors.but nowadays Manufacturers downsize the gates so they can allocate more gates on the same board, therefore increasing the functionality of the FPGA board.

- The Image used "Lena" is of size 128 X 128
- The Filter Size is 3 x 3
- Software Used Xilinx ISE (Integrated Synthesis Environment)
- Language is : VHDL

Flow of Project

1. Cache Memory ex: FIFO
2. Processing Block ex: adders, multipliers, multiplexers

Cache Memory

the cache memory acts as a data accumulator for the processing unit to process the data, since the minimum number of data units (PIXELS) in our case is 9 pixels because the filter size is 3 X 3. Therefore this cache memory accumulates the data from the camera and feeds it to the processing unit.

Note: if the filter size is 3 X 1 or 1 X 3 in size then we only need a cache memory to save 3 data units (Pixels)

Processing Block

processing block consists mainly of the filter which consists of the adders and multipliers, it processes the pixels provided by the cache memory in a pipeline structure.And saves the output into a file. synchronization between multipliers and adders is managed here as well.

Nexys4 Board

it is a field programmable gate array that is based on the latest Artix-7. it has USB, ethernet,12-bit VGA and other ports. Also it includes many built in devices and sensors, such as: accelerometer, temperature sensor, MEMs digital microphone, speaker amplifier, and a lot of Input/Output devices. therefore this board can be used in wide field of applications.

Specifications:

1. 240 DSP Slices
2. 16 user Switches
3. Analog to Digital Converter (XADC)
4. 4860 kbits of fast block RAM
5. 15850 logic slices
6. internal clock speed exceeds 450MHZ

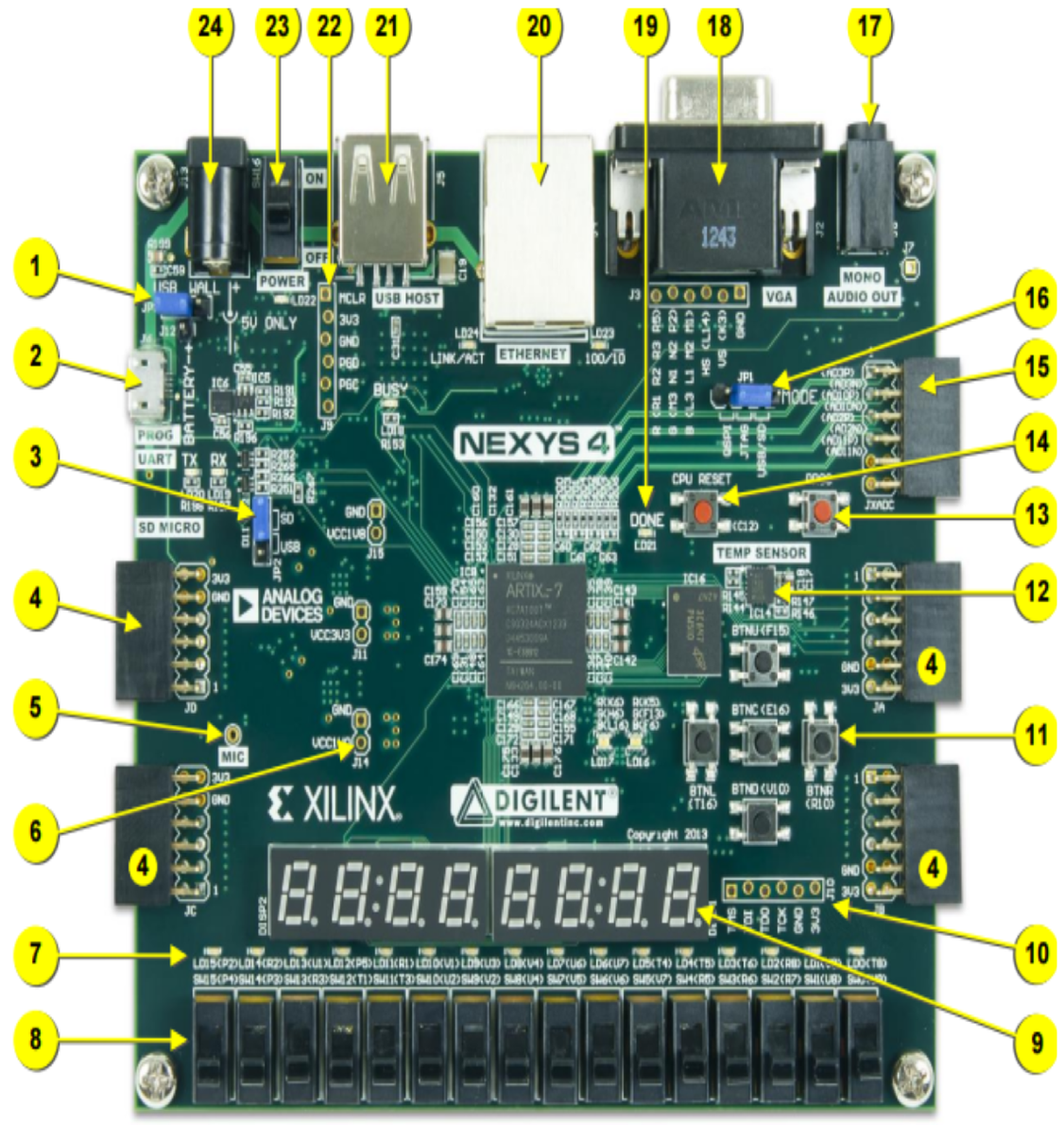


Figure 1. Nexys4 board features

Callout	Component Description	Callout	Component Description
1	Power select jumper and battery header	13	FPGA configuration reset button
2	Shared UART/ JTAG USB port	14	CPU reset button (for soft cores)
3	External configuration jumper (SD / USB)	15	Analog signal Pmod connector (XADC)
4	Pmod connector(s)	16	Programming mode jumper
5	Microphone	17	Audio connector
6	Power supply test point(s)	18	VGA connector
7	LEDs (16)	19	FPGA programming done LED
8	Slide switches	20	Ethernet connector
9	Eight digit 7-seg display	21	USB host connector
10	JTAG port for (optional) external cable	22	PIC24 programming port (factory use)
11	Five pushbuttons	23	Power switch
12	Temperature sensor	24	Power jack

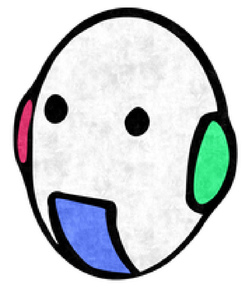
Figure 2: FPGA Ports and devices

Real Time Imaging and Control, Supervised by ***Prof. Julien Dubois***

L'université de Bourgogne est membre de UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Botros Karim, Tsagatakis Ioannis

kim0.swimmer@gmail.com, jtsagata@gmail.com



FIFO

FIFO (First Input First Output) is a memory block that allows us to queuing the data stream, in our case it is the camera feed of pixels, therefore the first entry is processed before other entries, **FIFOs** are widely used in communications and networks, gateways and many other applications that require sequential data processing. After the FIFO recieves the enable signal it allowes to recieve the data from the Camera Feeder, but this FIFO has $prog_full_thres$ which means the FIFO is full and the first entry (input) is ready to be written on the Dout

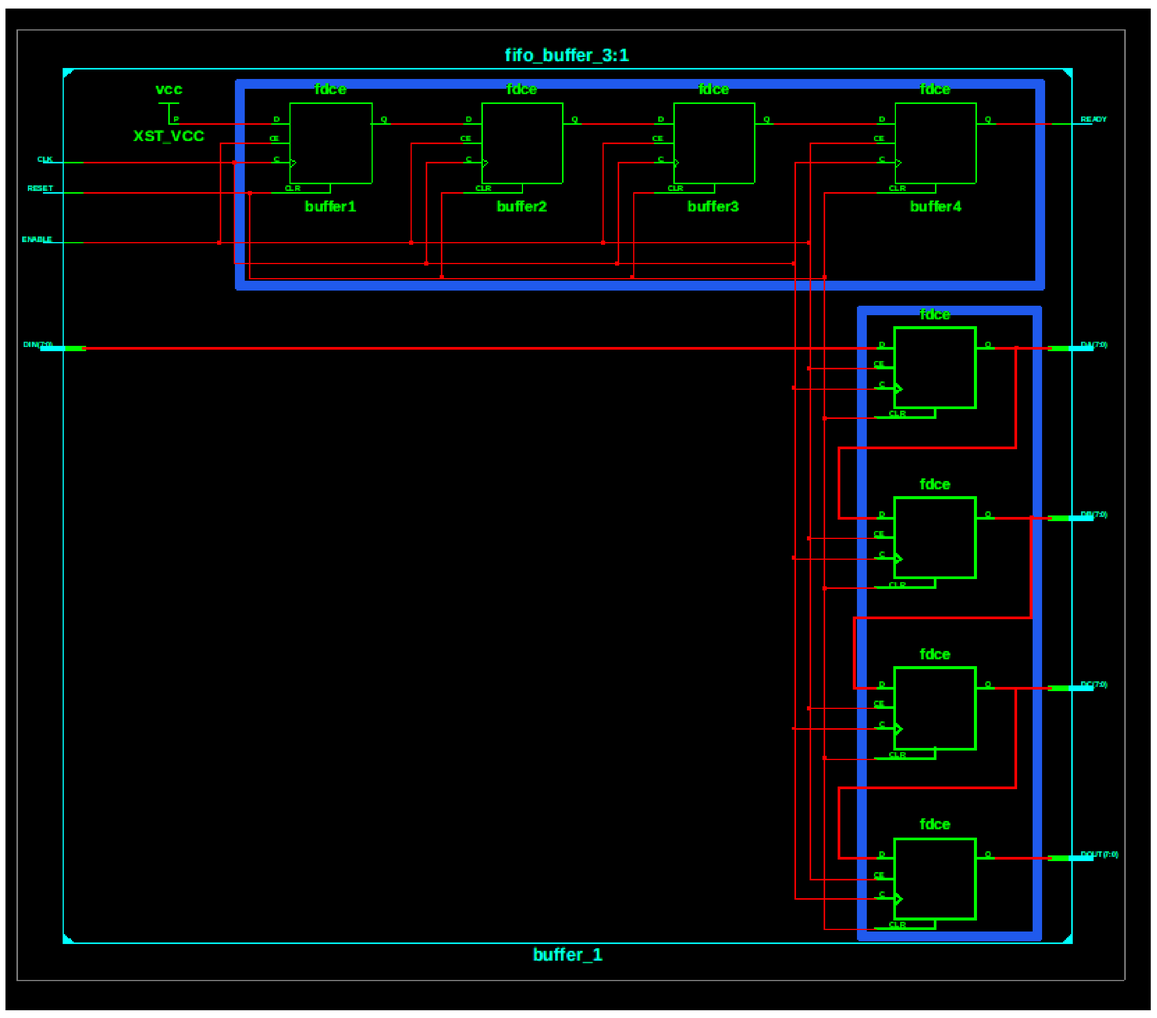


Figure 3: FIFO schematic

Also in the test bench we created a simulation for the FIFO behaviour we fed a Dummy feed at the beginning to simulate the feed of the camera to the **FIFO** memory and then we observe the output of the **FIFO** over the clock cycles

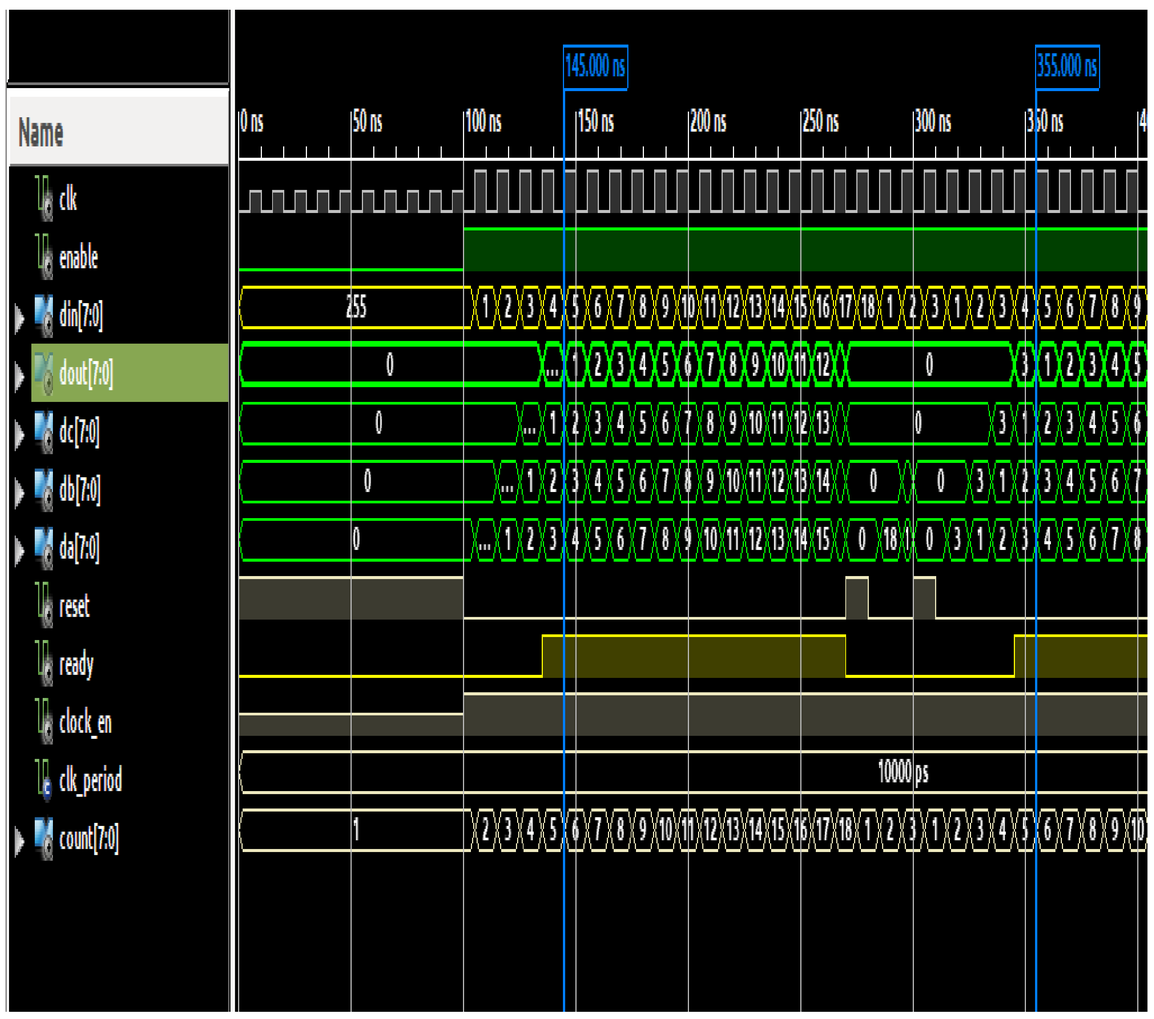


Figure 4: FIFO Simulation

Adders

Big part of filtering an image is doing the summation of the multiplication results of each pixel with its corresponding value in the kernal so if the 3 X 3 kernal is defined by:

$$\begin{vmatrix} k11 & k12 & k13 \\ k21 & k22 & k23 \\ k31 & k32 & k33 \end{vmatrix}$$

so we multiply the first pixel by k11 and then the next pixel by k12 and so on,, on the first 9 pixels. then we **add** all the results of the 9 multiplications. here comes the importance of using adders. below you will find our efficient implementation of an **adders**, it was created to allow the maximum number of additions in the same clock cycle

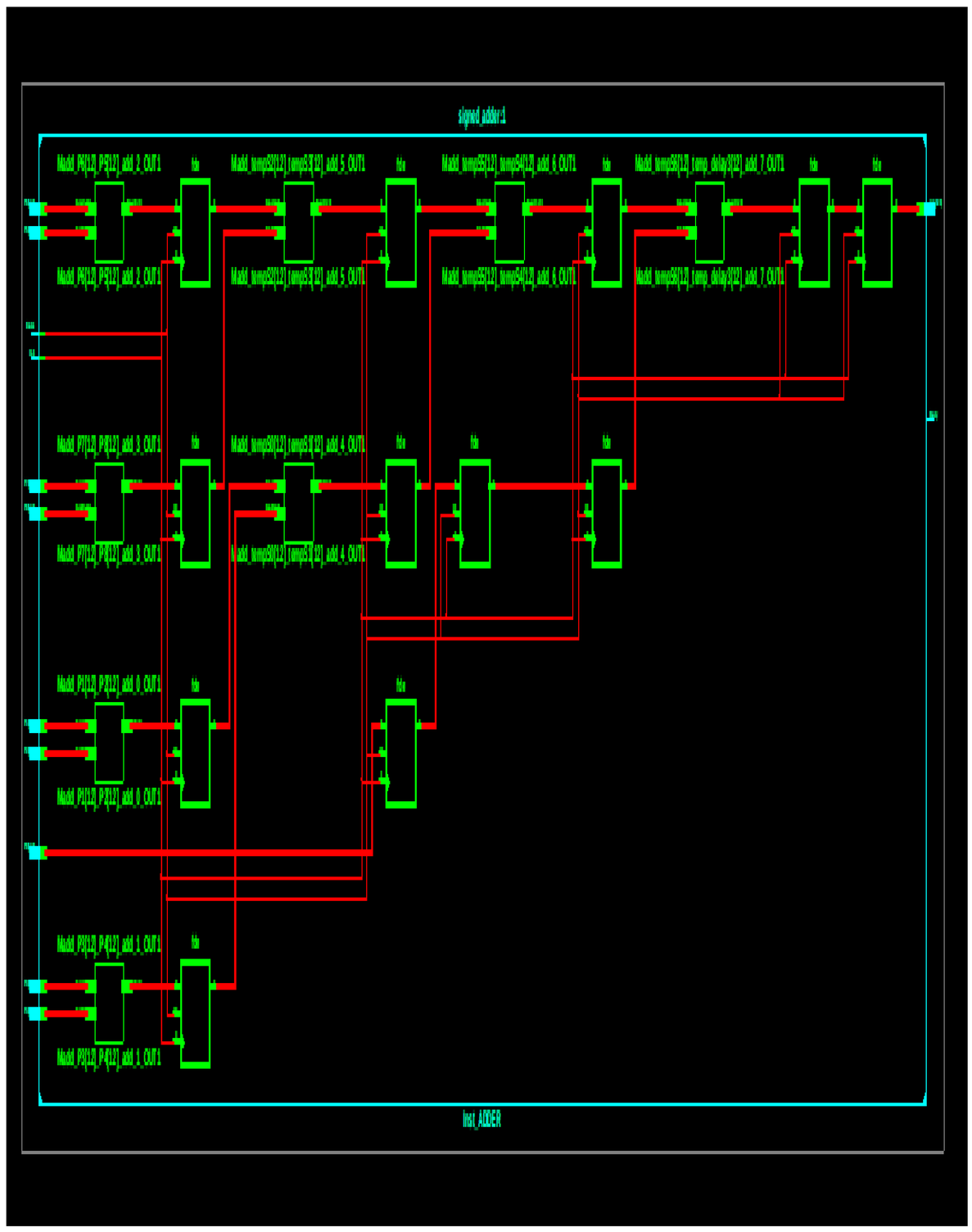


Figure 5: ADDER

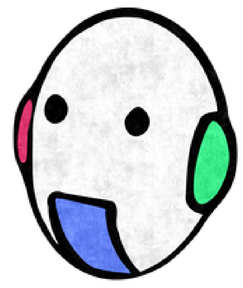
Multipliers

As we spoke previously about the adders in filtering, there is also the multiplication part which is the part in which the values of the kernal is multiplied by its corresponding pixel from the camera feed. Multipliers are composed of adders and shift registers, Also Shifting left by n bits means multiplying by 2^n, and shifting right by n bits has the division effect by 2^n

L'université de Bourgogne est membre de UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Botros Karim, Tsagatakis Ioannis

kim0.swimmer@gmail.com, jtsagata@gmail.com



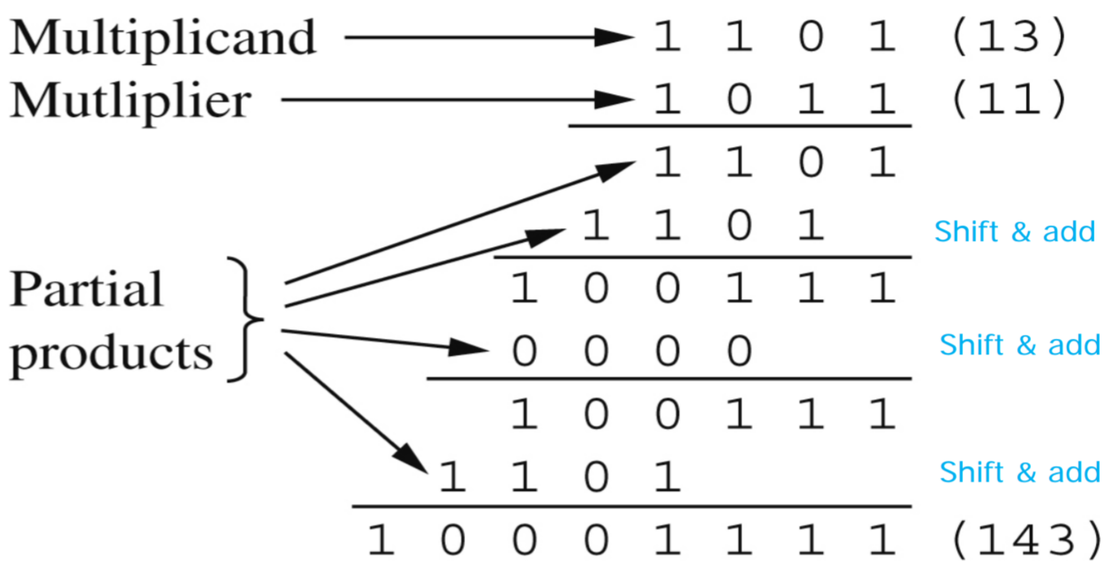


Figure 6: Multiplication in the sense of Logic Gates

therefore we combined the 2 major components in the filter which are the adders and the multipliers, and did testbench for this connected components

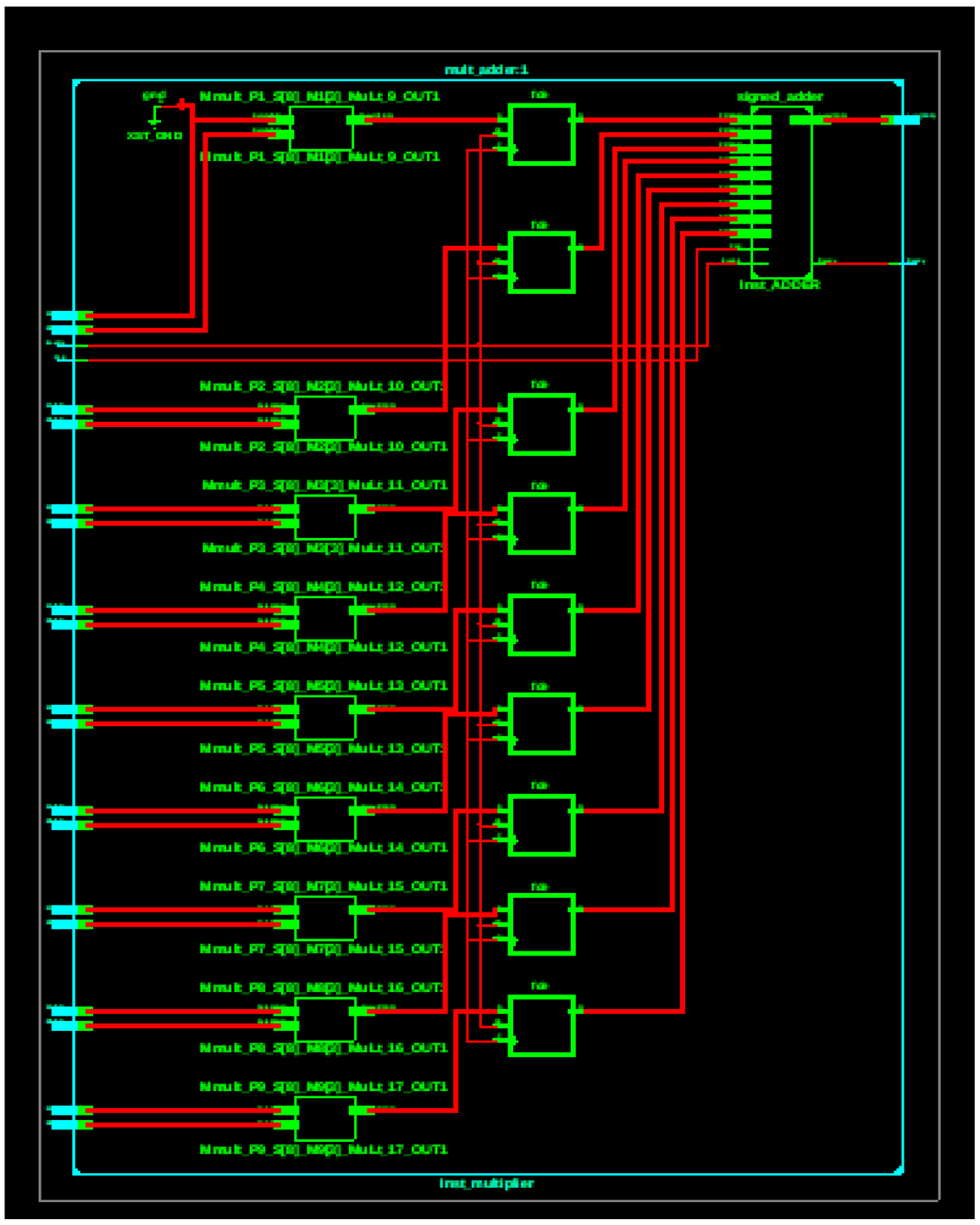


Figure 7: Multiplier with Adder

then the results is summed by a signed adder

Filter Bank

Filter bank is where we save all the filters ex: average filter, Gaussian filter, median filter. so the cache memory feeds this filter bank, but this filter bank have multiplexer which controls the feed direction to which filter is to be processed.

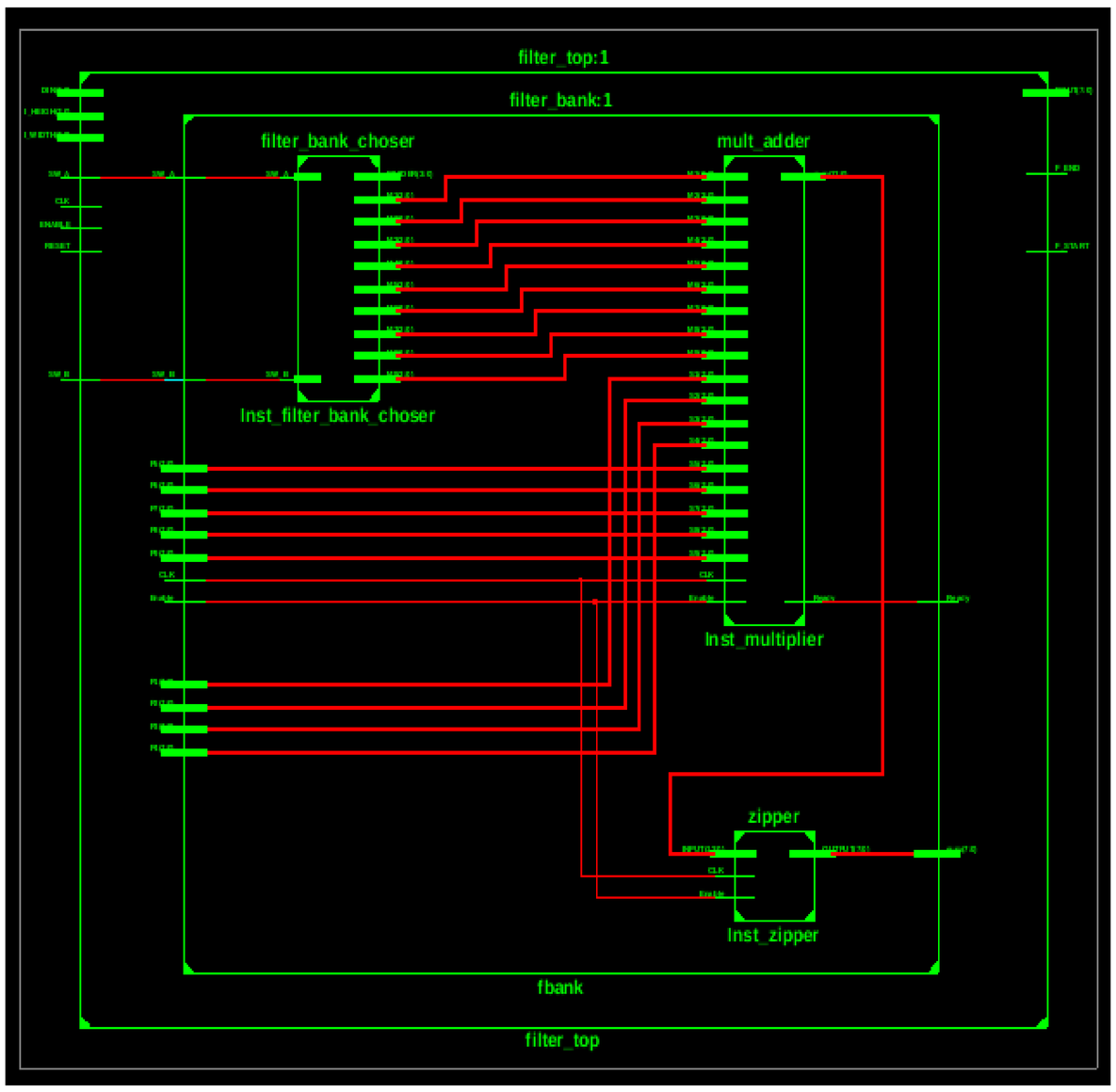


Figure 8: Filter Bank TOP

Filter Bank Multiplexer

filter bank multiplexer is essential for choosing the right filter to process the feed from the camera. we needed to create this component since we need our program to be compatible with many different kinds of filters such as average or Gaussian filters

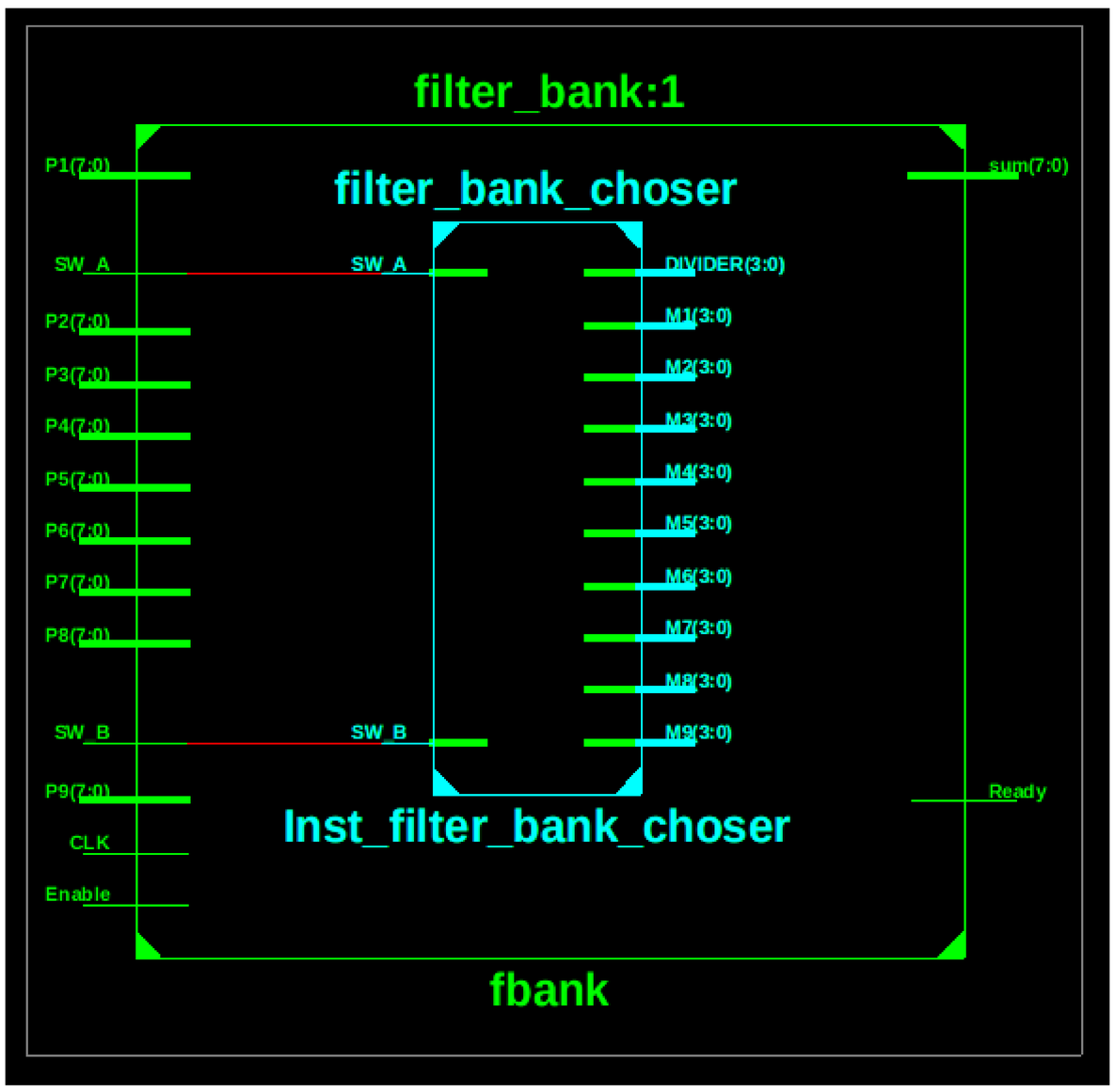


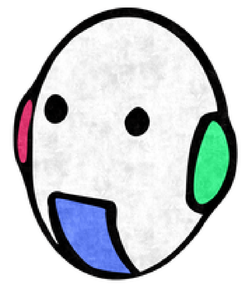
Figure 9: Filter Bank Multiplexer

so in order to connect the cache memory with the filter bank we needed to create somehow a multiplexer (controller) to connect both with the functionality

L'université de Bourgogne est membre de UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Botros Karim, Tsagatakis Ioannis

kim0.swimmer@gmail.com, jtsagata@gmail.com



we need for this project

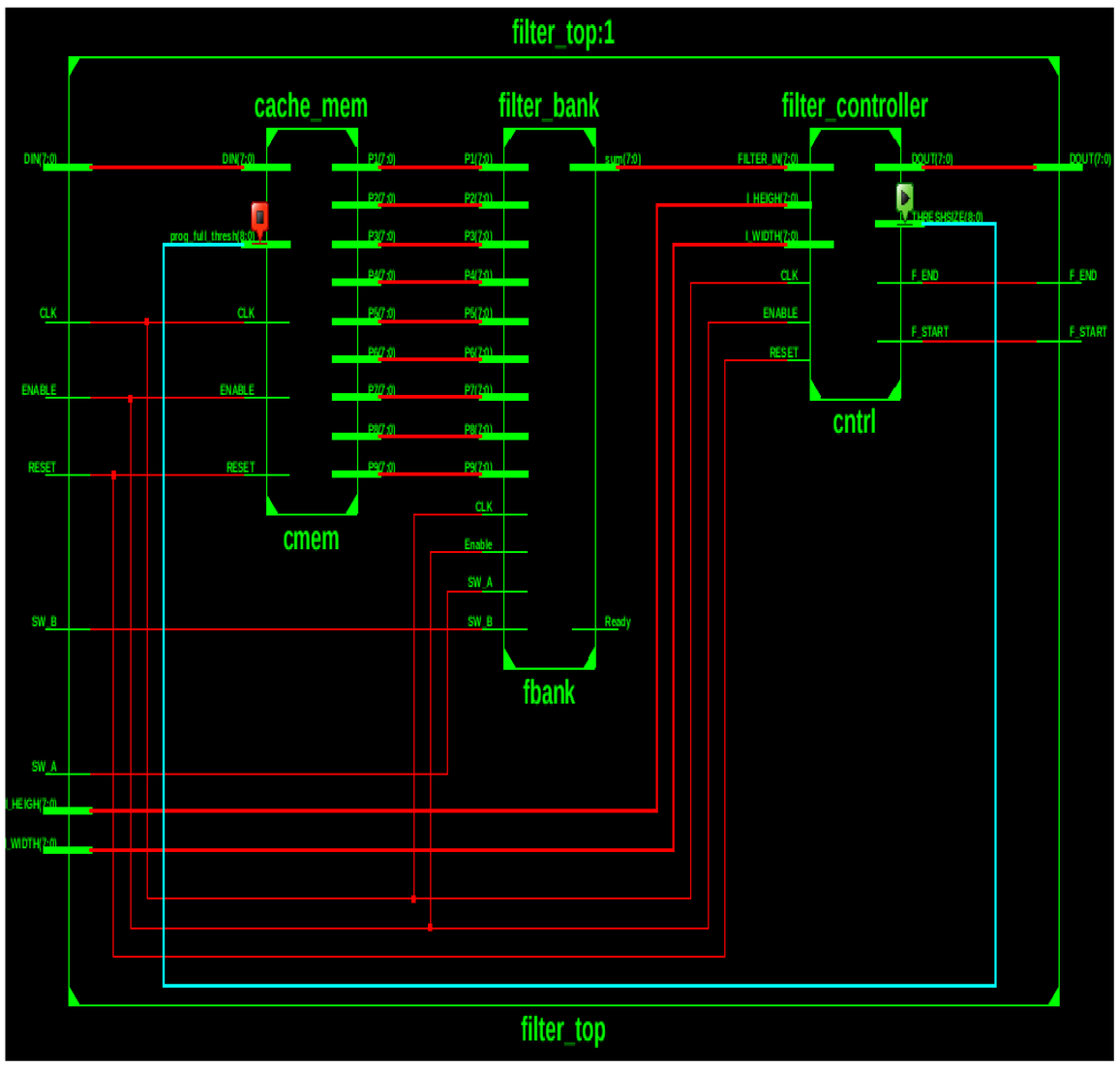


Figure 10: Top Component Filter

Cache Memory

the main purpose of the Cache memory is to store data that is always accessed by the software during the operation of filtering, fast access means high speed filtering, therefore this cache memory enable the program to access 3 x 3 neighborhood pixels simultaneously in one clock cycle, its structure is composed of Delayed flip-flops and FIFO memory

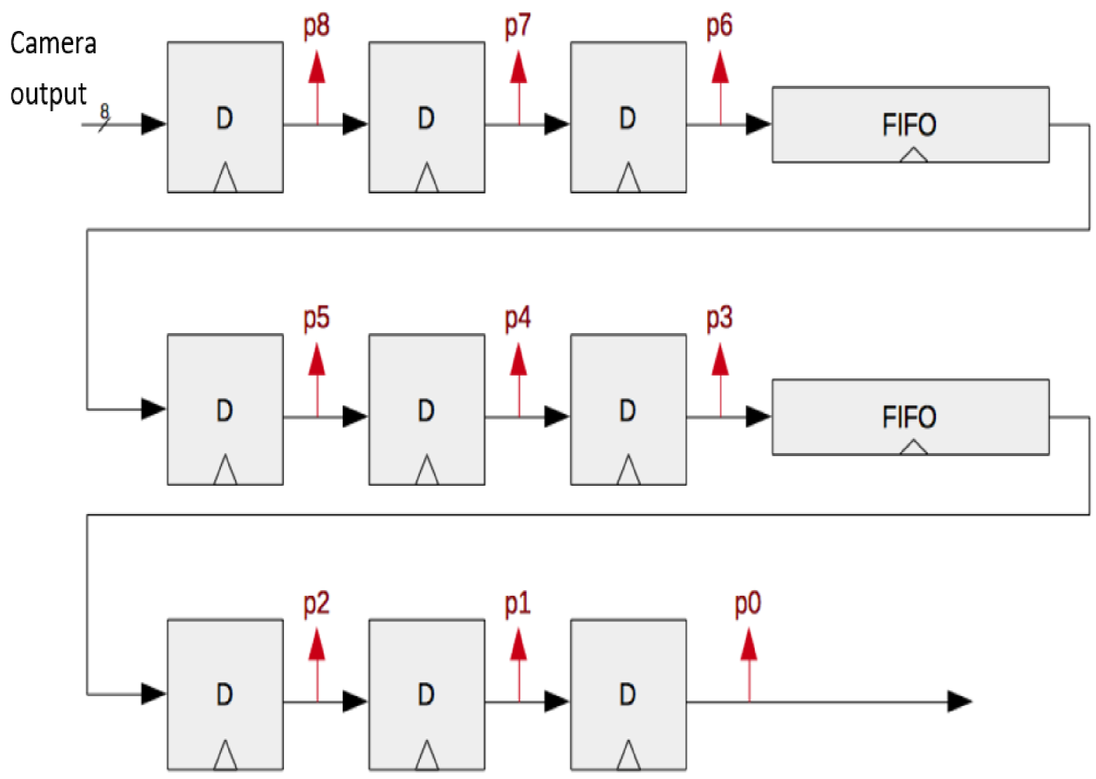


Figure 11: Cache memory Structure

so we combined the two components the FIFO and the D-Flip-Flops to create this cache memory in Xilinx and create testbench for it as well,please view our implementation below:

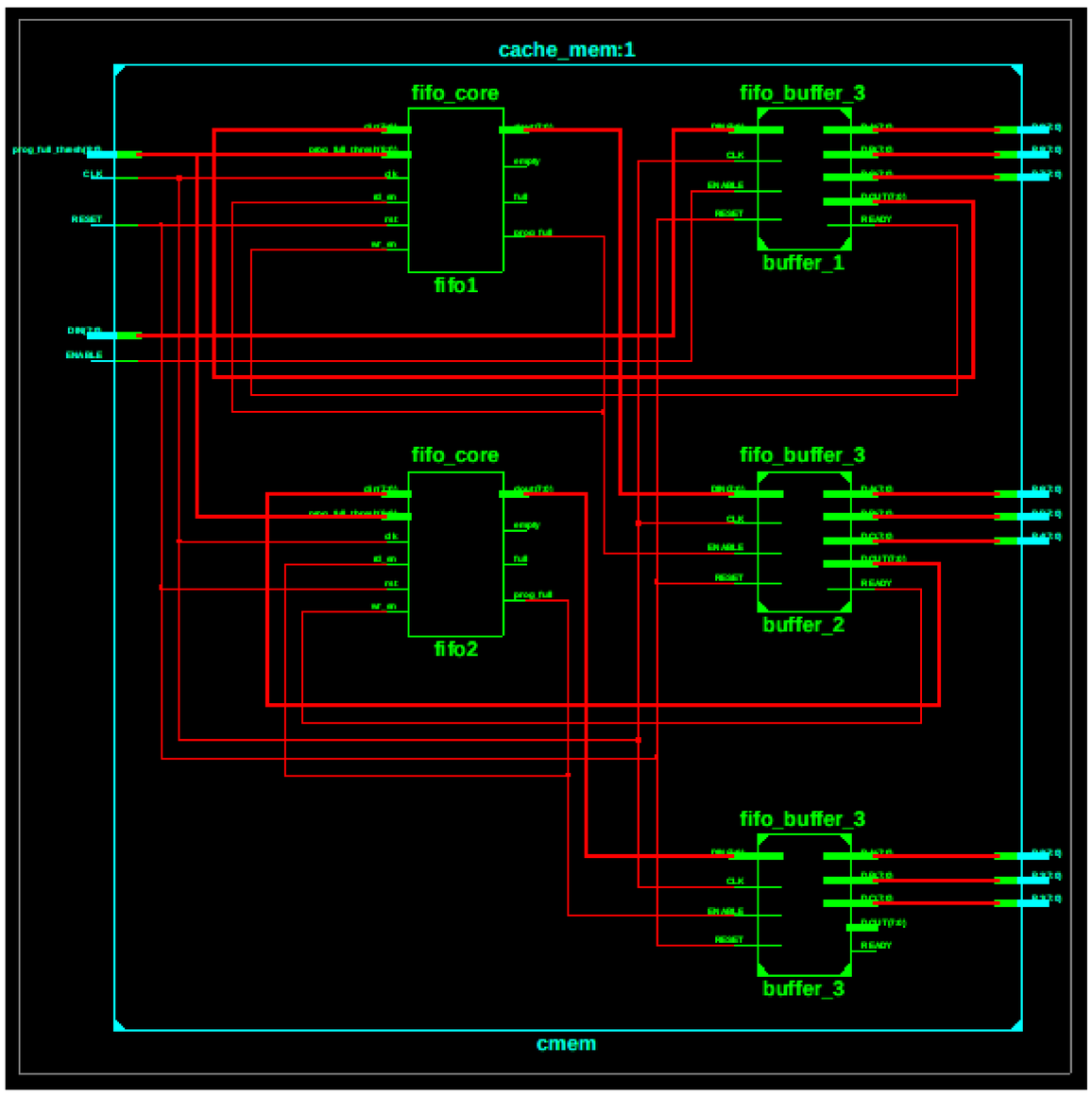


Figure 12: Cache memory implementation

Zipper

the zipper is a module that takes the filter's output and trims it down to the range between 0 and 255.

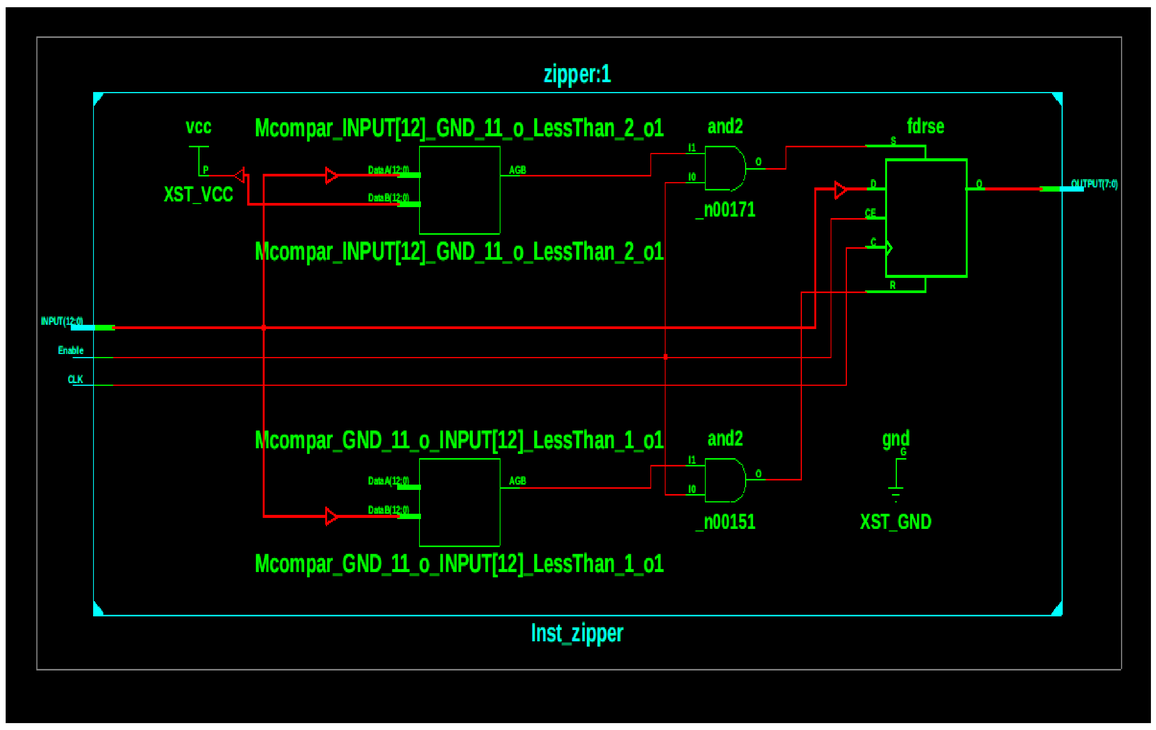


Figure 13: Zipper

Conclusion Results

we have simulated the 2D filtering of Lena image with VHDL in Real time using FPGA nexys4. also we developed a cache memory for simultaneous pixel access of 3 x 3 neighborhood, we also designed a pipeline and connected all these components togther and also simulated a dummy feed to our code and we also created a testbench for each of our components to test it and see how fast it is.

please find the project link HERE

please find the results link HERE