# REAL TIME IMAGING AND CONTROL

## FPGA PROJECT: Image Processing on VHDL

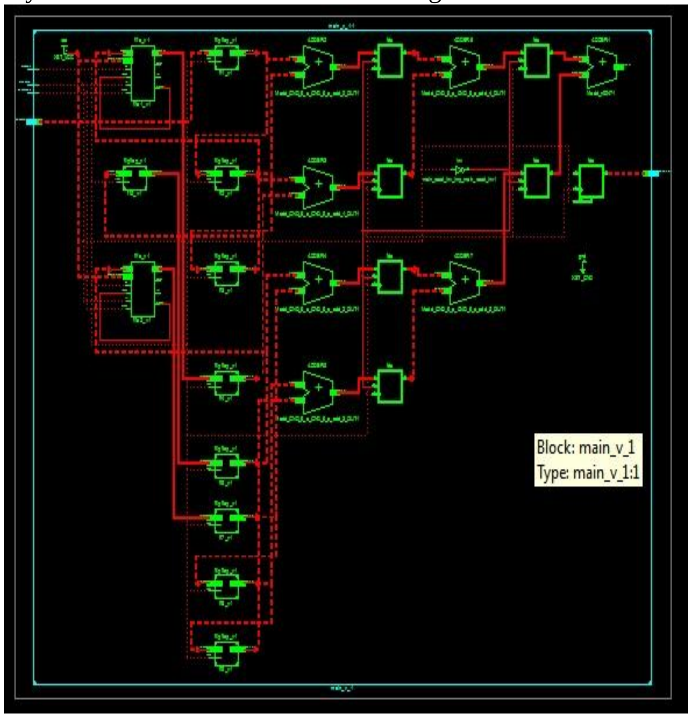
Submitted By Chalikonda Prabhu Kumar (ChaKon)     Student Id: 12012543

*Introduction*—**The main motivation of this project is to get an experience with VHDL programming. The image processing techniques applied for images using digital electronics components on VHDL platform. The main advantages of VHDL are it can handle parallel operations; it can do the operation with low size memory when compared with normal processing, the processing time is less.**

**The main task is smoothing the image and applying sobel operator to get the edges of the image. In this report section 1 is implementation and design of Image smoothing, section 2 for edge detection and section 3 and 4 followed by problems and discussion, and conclusion.**

## I. *Image smoothing*

According to the given inputs architecture is done. I designed my structure as shown in the below image.

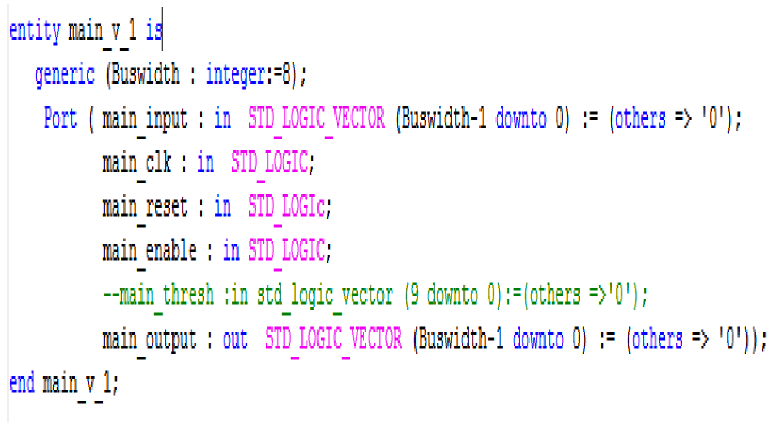


In this I used 9 flipflops and 2 fifo generators. Then I used adders to smooth the image. The above schematic diagram when I used adders. I tried with other thing as well it gives better result when compared to above operation.

After getting the pixel value from each flip flop I converted into integer and then I am summing up all pixels values and dividing with 9 gives the blurry image. The schematic diagram of the converting into integer and multiplying with kernel and taking is complicated to show. In fact it is like global model for applying any kind of kernel

To make it clear I am explain the implementation part step by step. I did main project file with inputs and outputs. The code I did for this. I made it generic somehow if there are more bits we can do operation without any failure. If there are more bits the remaining bits will be with zeros. For example in generic if we declare Buswidth as 32 the resultant output is 12 bits the remaining 20 bits are zeros.

The structure of the main is shown below.



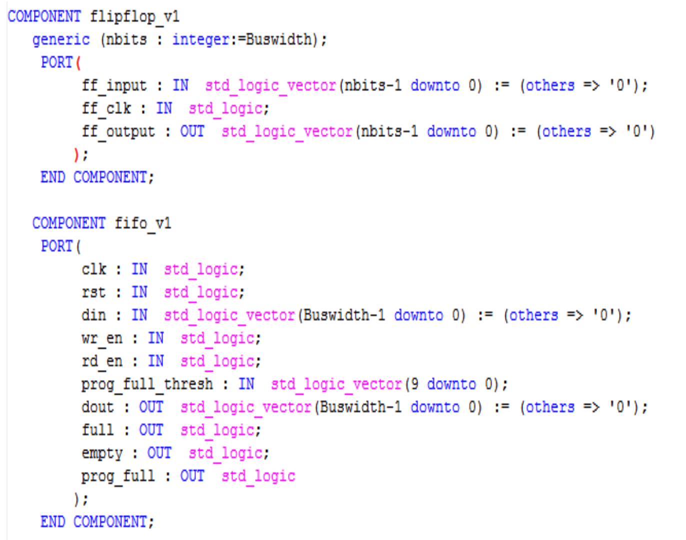
```
entity main_v_1 is
  generic (Buswidth : integer:=8);
   Port ( main_input : in  STD_LOGIC_VECTOR (Buswidth-1 downto 0) := (others => '0');
          main_clk : in  STD_LOGIC;
          main_reset : in  STD_LOGIc;
          main_enable : in STD_LOGIC;
          --main_thresh :in std_logic_vector (9 downto 0):=(others =>'0');
          main_output : out  STD_LOGIC_VECTOR (Buswidth-1 downto 0) := (others => '0'));
end main_v_1;
```

The architecture of main file is complicated to show. To be precise it has components of flipflop and fifo generator. How this will works is, flipflop will gives output when clock is at rising edge else it will give zeros. While coming to the FIFO generator wr_en and rd_en will plays a key role this is because when rd_en is 1 then it will read the data from the file once it is full then it will start writing. In this case threshold is also important. We came to know that fifo is full because of this threshold in my case I set it to 125 in binary form.

The whole idea of the project reading the data file is first we will read first line of the image. First we will get 3 outputs from each flip flop when fifo is full then the output from the fifo is connected to the flipflop 4 and the same repeats. Finally the outputs of each filpflop (9) gives p1,p2…..p9.

The advantage of FIFO is it stores all the remaining pixels lets say 125 (depending on the threshold given). When the operation is done then it will release next pixel. So it is names as first in first out.

The components of flipflop and fifo are shown in the architecture.

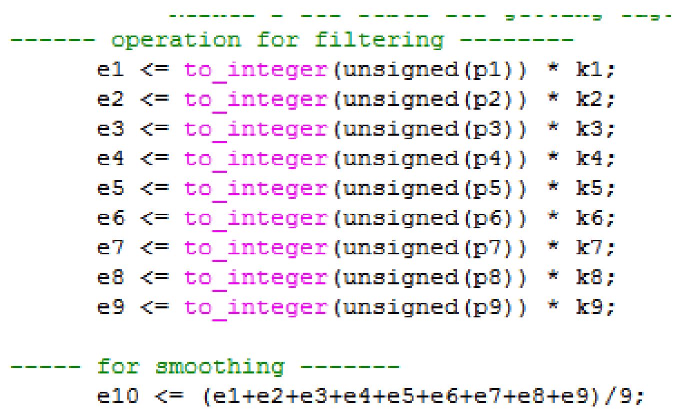


```
COMPONENT flipflop_v1
   generic (nbits : integer:=Buswidth);
    PORT(
         ff_input : IN  std_logic_vector(nbits-1 downto 0) := (others => '0');
         ff_clk : IN  std_logic;
         ff_output : OUT  std_logic_vector(nbits-1 downto 0) := (others => '0')
        );
    END COMPONENT;

   COMPONENT fifo_v1
    PORT(
         clk : IN  std_logic;
         rst : IN  std_logic;
         din : IN  std_logic_vector(Buswidth-1 downto 0) := (others => '0');
         wr_en : IN  std_logic;
         rd_en : IN  std_logic;
         prog_full_thresh : IN  std_logic_vector(9 downto 0);
         dout : OUT  std_logic_vector(Buswidth-1 downto 0) := (others => '0');
         full : OUT  std_logic;
         empty : OUT  std_logic;
         prog_full : OUT  std_logic
        );
    END COMPONENT;
```

As shown in the schematic diagram above connect three flipflops each other i.e. output of one flipflop as input to the other. The output of third flipflop is connected to the fifo with necessary condition for reading, writing and prog_full will give an output after storing 125 pixels in it. This fifo output is give as input to the 4rth flipflop and so on. By doing this we will get 9 bits of output which covers all 3x3 of the window to do smoothing. Then I converted each into the integer. The converted integer is multiplied with the kernel value this applies filter. I did the same process for detecting the edges as well. But some time it giving me the good output sometime it is not giving me an output as expected. But I tried with other for smoothing but it does not give like converting the p1,….,p9 as integers. I tried with adders it gives results but smoothing is not good when compared with the integer.

The code for implementing filtering is shown below.



```
------ operation for filtering --------
      e1 <= to_integer(unsigned(p1)) * k1;
      e2 <= to_integer(unsigned(p2)) * k2;
      e3 <= to_integer(unsigned(p3)) * k3;
      e4 <= to_integer(unsigned(p4)) * k4;
      e5 <= to_integer(unsigned(p5)) * k5;
      e6 <= to_integer(unsigned(p6)) * k6;
      e7 <= to_integer(unsigned(p7)) * k7;
      e8 <= to_integer(unsigned(p8)) * k8;
      e9 <= to_integer(unsigned(p9)) * k9;

----- for smoothing -------
      e10 <= (e1+e2+e3+e4+e5+e6+e7+e8+e9)/9;
```

Changing the k1 to k9 values for the getting the edges.

The results I got when I do with the adders is shown below.

From the above image we can see that image is smoothed but when doing with the adders we can see that each pixel clearly in then image but the result obtained when doing with the integer is shown below.

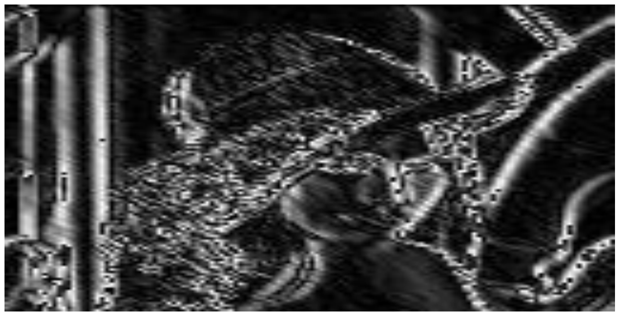
The above image converting to integer and doing average gives better results.

## II. *Image Filtering*

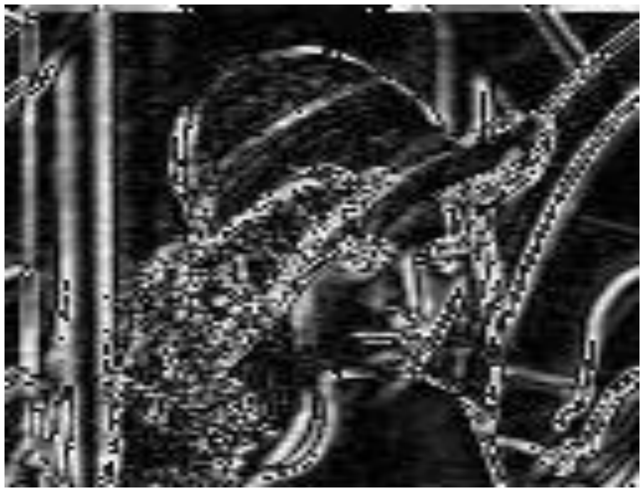
For the image filtering we have give a sobel operator to detect the edges. Instead of changing the whole code I change the values of the kerenel according to the kernel that was given for the project.

Here I am getting edges but the edges were not that much strong and smooth as to be when applied sobel filter in general.

The results for the sobel kernel [-1,0,1;-2,0,2;-1,0,1] is giving me  the output like this initially when I changed to make it wilt switch case it turned into worst.



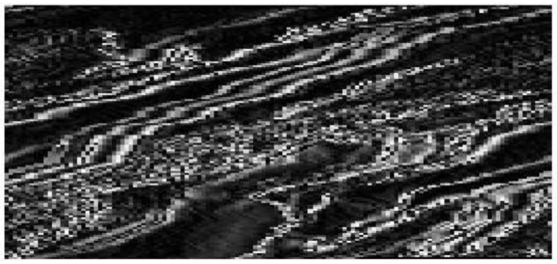
The result obtained when sobel kernel [-1,-2,-1;0,0,0;1,2,1] is giving results as shown below.



In the test bench we have to read the data file and write as well. For this necessary libraries should use like textio, std_logic etc.

## III Problems and Discussion:

1. When doing the coding part for smoothing I tried with adders it gives result not as good as when converted into string. In fact what I observed in the output of the smoothed image first row is missing probably because of applying filter from 2nd row and 2nd column. And last four bits are zeros. In order to overcome this issues the image should pad with zeros then applying smoothing will give better results.
2. For getting the edges of the image I changed the kernel applied for smoothing at the beginning it gives better result when I modified to make code more optimized it is giving the worst results as shown below I try ro debug because of time lagging I am not able to do that.



3. Without any knowledge on VHDL it hard to work. To be honest by doing project and coding it gave good plat form to learn about coding and concepts in deep. It will be more practice if we tried any of the labs during class session to debug the errors.
4. In xlinks division cannot be synthesis, Actually we need the behavior of the architecture it work pretty good.
5. Unwanted libraries lead to errors as well.

## IV Conclusion

In this it is easy to work on this platform applying digital electronics concepts for image process is more good and reliable. The processing time is less for doing necessary operation. But without previous know knowledge on vhdl it is hard to work completely feeding on the forums and google is quiet complicated.

## V REFERENCES:

1. Project guide lines in class, sample codes
2. Xlinks forum
3. VHDL premier J.Bhasker book
4. www.google.com