Project #1: Real-Time Video Processing

Objective:

The purpose of this project is to develop a video processing design using DE10-Lite FPGA development kit and D8M camera module.

Requirements:

- Quartus Prime Lite 17.1/or above
- ModelSim Starter Edition
- DE10-Lite development board
- Terasic D8M-GPIO is an 8-megapixel camera kit
- VGA display

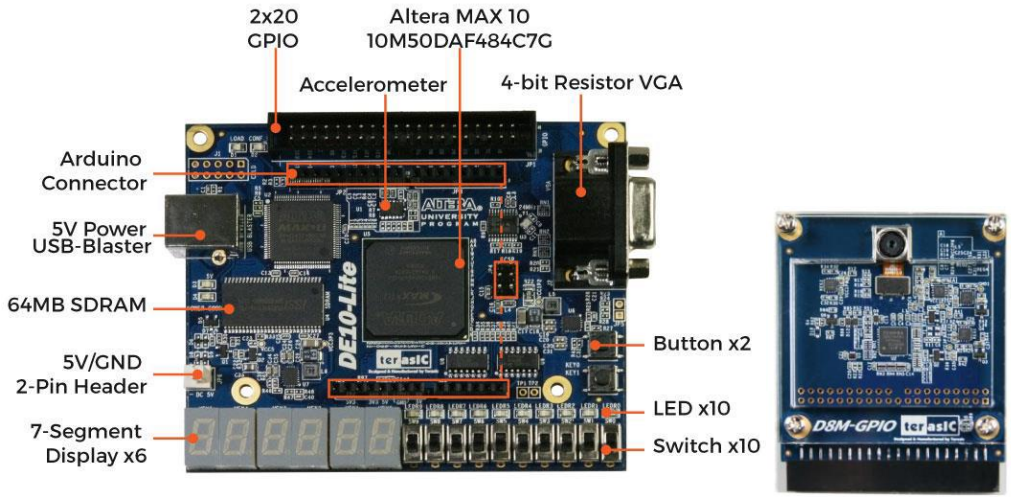

Figure 1. DE10-Lite development board and D8M camera module

Description:

You will design a real-time image/video processing that implements edge detection algorithm on FPGA hardware. Edge detection can be realized by combination of two line buffers as well as a filter kernel. The image captured by the camera module is interfaced with FPGA through MIPI. The incoming video stream is fed into the edge detection module for processing. The edge detected pixels then stream to a video display through VGA.

There is also an accelerometer (G-sensor) on the FPGA board. As an additional step the data read from the G-sensor can also get buffered and displayed on the monitor through VGA. For the purpose of the project, following tasks need to be completed:

1. Edge detection wrapper
    a. Sliding window
    b. Sobel filter
    c. Line buffer
2. Accelerometer (optional)
3. VGA Plotter (optional)
4. Top module

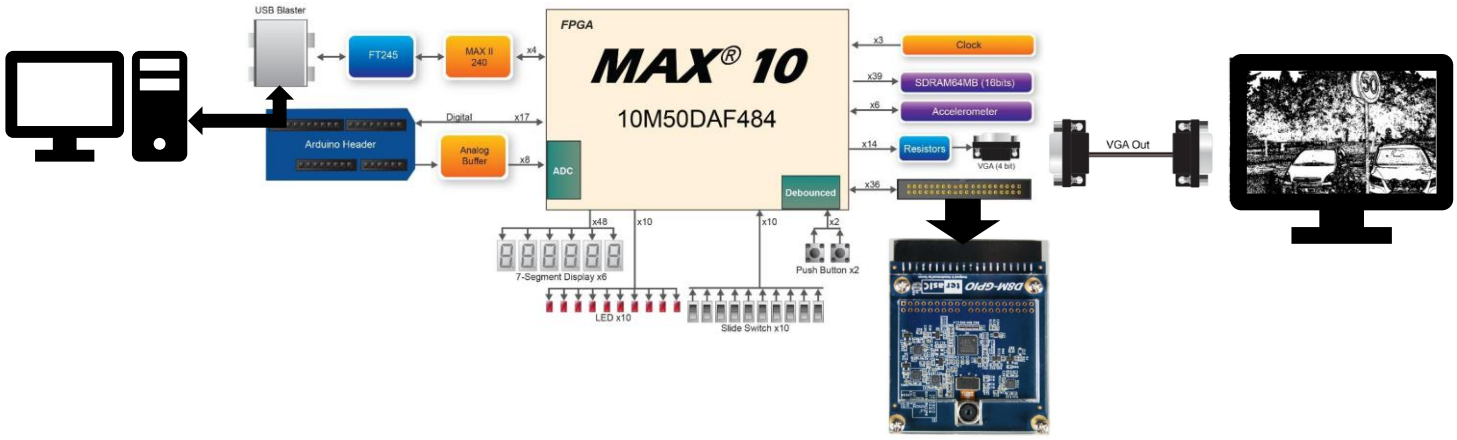

Figure 2. System Overview

Tasks:

1. Edge Detection:

This module can be comprises of two different modules namely; sliding window and edge detection filter. The sliding window forms a 3x3 neighborhood where the filtering process is done within this window. The sliding window itself can contain two line buffers each one is responsible for buffering one line of the pixels.

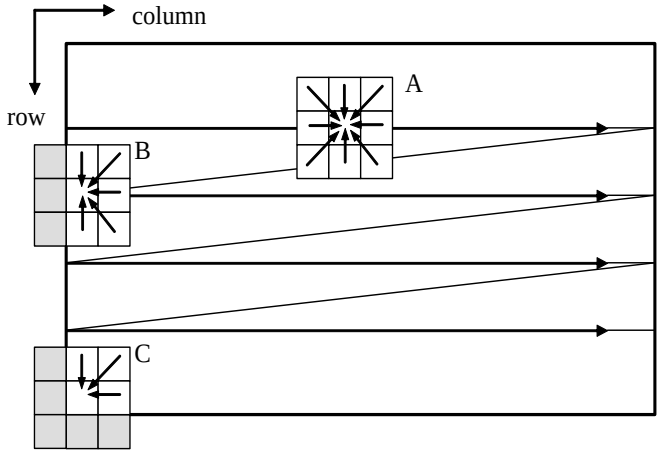

Figure 3. A 3-by-3 pixel sliding window at different image positions

Above figure also depicts the window at different positions. When the sliding window is positioned at the border (position B and C) of the video frame (image), you can see that some of the pixel values within the neighbourhood must be estimated since it is not possible to access pixels outside the frame border that does not exist. For this project, the border conditions are not being considered.

Figure 4 depicts a memory architecture that provides the necessary storage of previous pixel values in order to facilitate the sliding window shown in Figure 3. The group of elements labelled L2 are two First-In-First-Out buffers (FIFO). These buffers must hold exactly the number of pixels in one row and are usually referred to as line buffers. The other group of elements labelled L1 are registers that must reside close to the data path where the filter computation is done. Therefore L1 constitutes a cache level 1 and L2 constitutes level 2 for this memory hierarchy. L1 is obviously a smaller cache copy of pixel values which are also stored in L2. The registers in L1 are typically implemented in a selection of the flip-flops that are numerously distributed over the FPGA chip. The line buffers are implemented in the FPGA block rams available on chip. An additional counter is needed for every line buffer for the modulus incrementing of memory addresses.

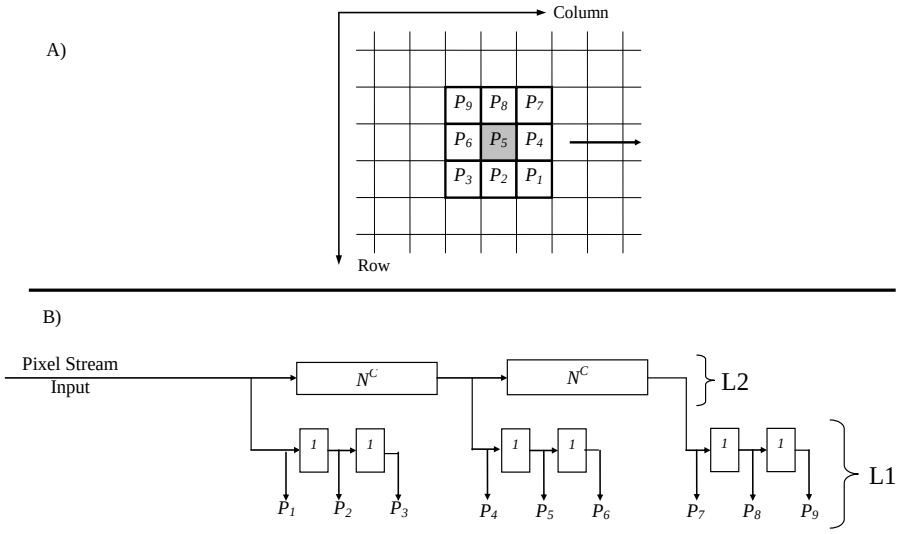

Figure 4. Memory architecture

To implement this design, 9 registers are defined to save the 9 pixel values for sliding window. Two line buffers are employed to buffer line pixel, and then the value of register is being shifted every clock cycle. Line valid and frame valid signals are transferred respectively to get module synchronized. Figure 5 indicates one typical implementation for the sliding window.

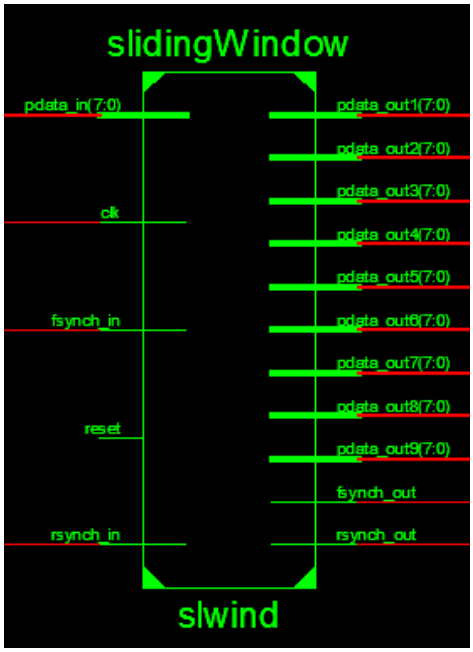

Figure 5. Sliding window module

In order to calculate the edges of the video data, the Sobel operator should be used as the kernel. the operator calculates the opposite of the gradient of the image intensity at each point, giving the direction of the largest possible change from light to dark and the rate of change in that direction. The result therefore shows how "abruptly" or "smoothly" the image changes at that point and therefore how likely it is that that part of the image represents an edge, as well as how that edge is likely to be oriented.

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Gx and Gy are the Sobel operators in both vertical and horizontal directions and the magnitude is calculated using following formula:

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

So, first by buffering two lines 3x3 window will be sliding through the whole image and the 9 pixels will be the inputs for the Sobel operator. Then, the kernel is convolved with the window and finally a pixel will be sent out as output pixel.

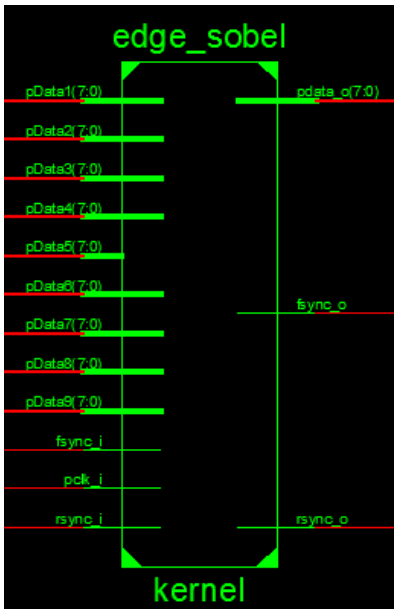

Figure 5. Sobel filter module

Having calculated the output pixel, it sends out frame and line valid signals to the other module. To calculate the edges, the Sobel operator is convolved with the 9 neighbourhood pixels in both vertical and horizontal directions. Then its magnitude is calculated and assigned to the output signal.

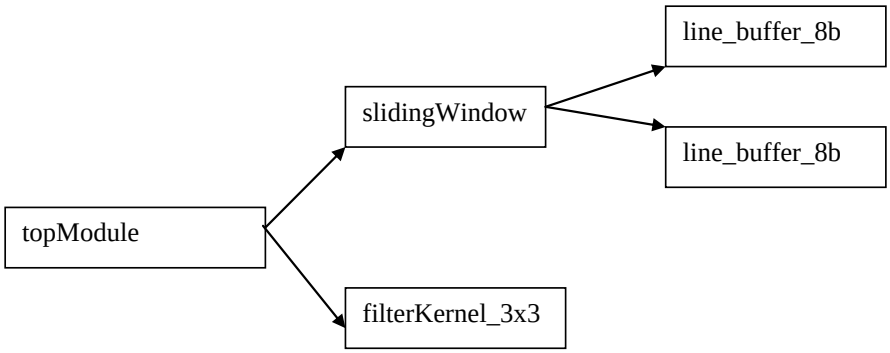

Figure 6. Edge detection module

The module shown in figure 7 is only the top module for the mentioned entities and contains the sliding window, line buffers, and Sobel operator modules. What happens in this module is that for each clock cycle a pixel data is transferred in and out the module, where the input pixel is coming from camera and the output data goes to VGA module to decide what kind of data must be written to the VGA display.

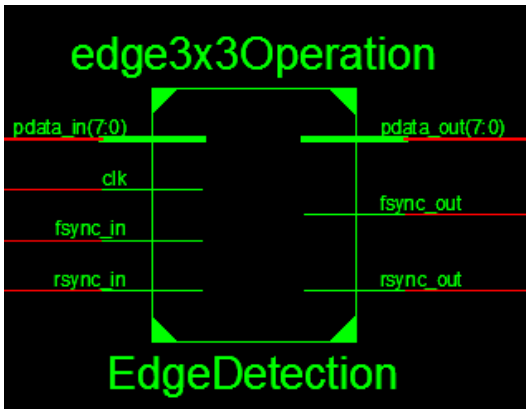

Figure 7. Edge detection wrapper

Following figures show the simulation result for whole edge detection module containing sliding window, line buffers and Sobel operator. This step can be used to verify correct implementation of edge detection.

Figure 8. Test image to verify the edge detection functionality

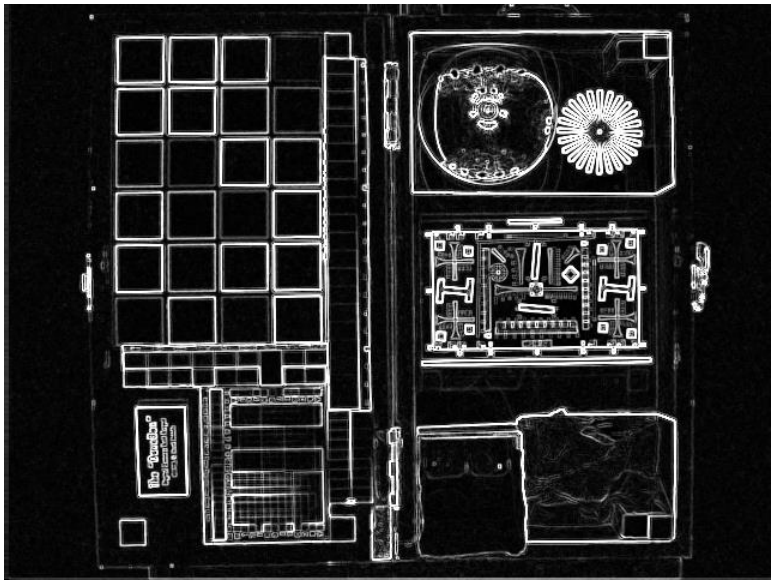

Figure 9. Output image which confirms the edge detection functionality

2. Accelerometer

The FPGA development board comes with a digital accelerometer sensor module (ADXL345), commonly known as G-sensor. This G-sensor is a small, thin, ultralow power assumption 3-axis

accelerometer with high-resolution measurement. Digitalized output is formatted as 16-bit in two's complement and can be accessed through SPI (3- and 4-wire) and I2C digital interfaces.

With GSENSOR_CS_N signal to high, the ADXL345 is in I2C mode. With the GSENSOR_SDO signal to high, the 7-bit I2C address for the device is 0x1D, followed by the R/W bit. This translates to 0x3A for a write and 0x3B for a read. An alternate I2C address of 0x53 (followed by the R/W bit) can be chosen by low the GSENSOR_SDO signal. This translates to 0xA6 for a write and 0xA7 for a read. The data read from G-sensor should properly buffered and sent to VGA display. You can consult to G-sensor demo in Terasic website for reference.

More information about this chip can be found in its datasheet, which is available on manufacturer's website or in the directory \Datasheet\G-Sensor folder of DE10-Lite system CD.

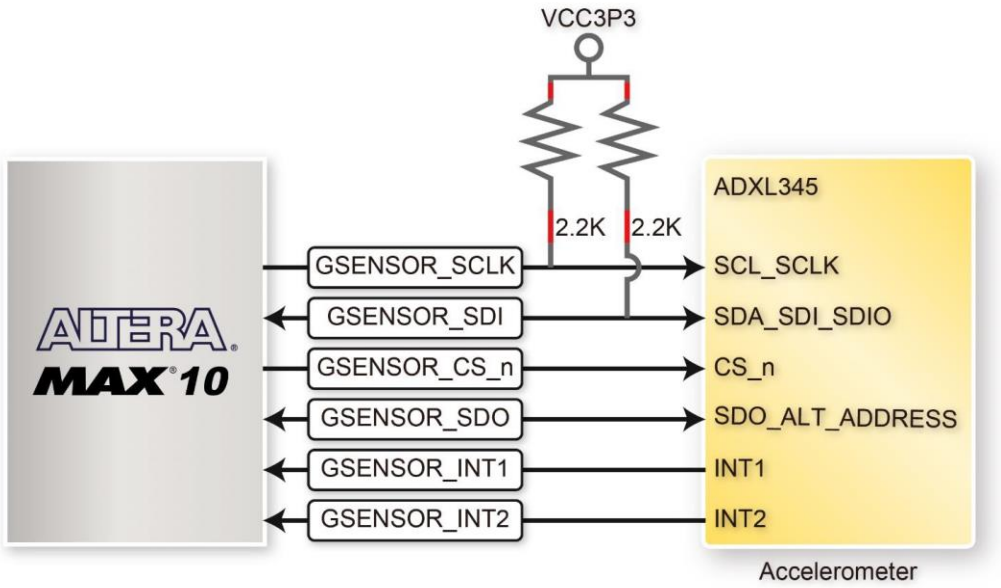

Figure 10. Connections between the accelerometer sensor and MAX 10 FPGA.

Buffer sensor data:

In order to be able to write the sensor data in VGA display, storing data is inevitable. A buffer is used to save the sensor data and read them out at the proper time. The point is that to plot the sensor data on screen we need two pointers for the buffer. One is used for writing into buffer and the other for reading out to monitor. To achieve this, the write pointer is added whenever a new data is coming from range sensor data and the read pointer is incremented by pixel clock. By implementing this strategy the plot will be shown on the screen.

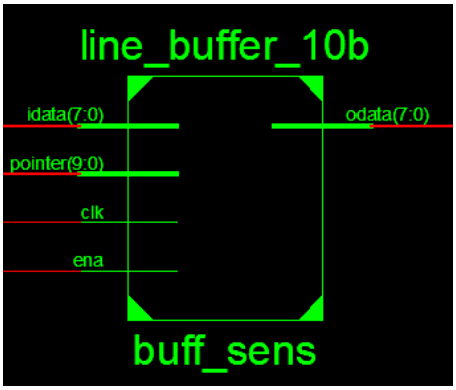

Figure 11. RAM entity to store sensor data

3. VGA_plotter:

The final module to design is VGA_plotter in which you can decide how to translate and show G-sensor data to display. All data coming from the edge detection module as well as buffered sensor data go through this module. Two counters can defined to keep track of rows and columns of the screen and decide where to plot the sensor data and where to read the camera data. For instance, 64 rows can be used to plot the sensor data and the rest of screen shows the camera data.

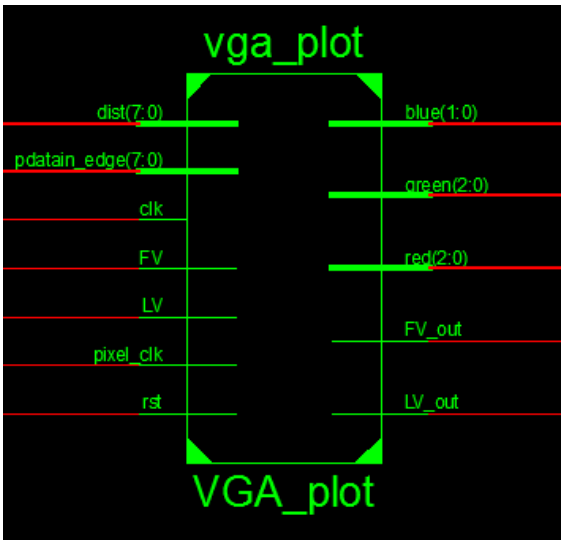

Figure 12. VGA_plot entity

4. Top module:

Finally the top module is a structural entity where all the components are declared and connected together. The ports of the top module then connected to appropriate pins of the FPGA. After synthesizing the codes, translate, map and place & route, the final steps is to program the device.

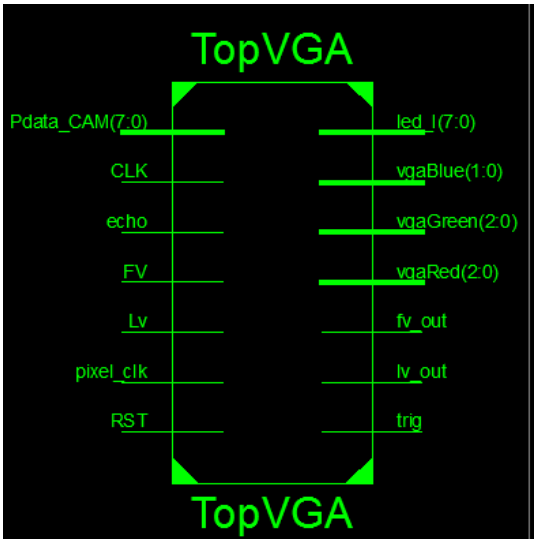

Figure 13. Top module

Outcome:

The final outcome of the design should be similar to shown in figure 14 where edge detected video is displayed on the monitor, and G-sensor data is also shown with different color codes (representing different sensoe values) in the middle of the screen.

Figure 14. Edge-detected video on FPGA