# Sobel Operator on Basys 3 with VGA I/O

Jaison George, Sumit Mondal

Hariank Mistry, Alex Saad-Falcon

# Table of Contents:

# 1. Introduction

Many image processing techniques involve convolutions with different kinds of filters. For example, convolving with [1, -1] in the x and y dimensions gives a discrete first derivative in those dimensions.

One especially useful filter is the Sobel operator: it is a mix of a first derivative and an averaging filter, and it takes the form below:

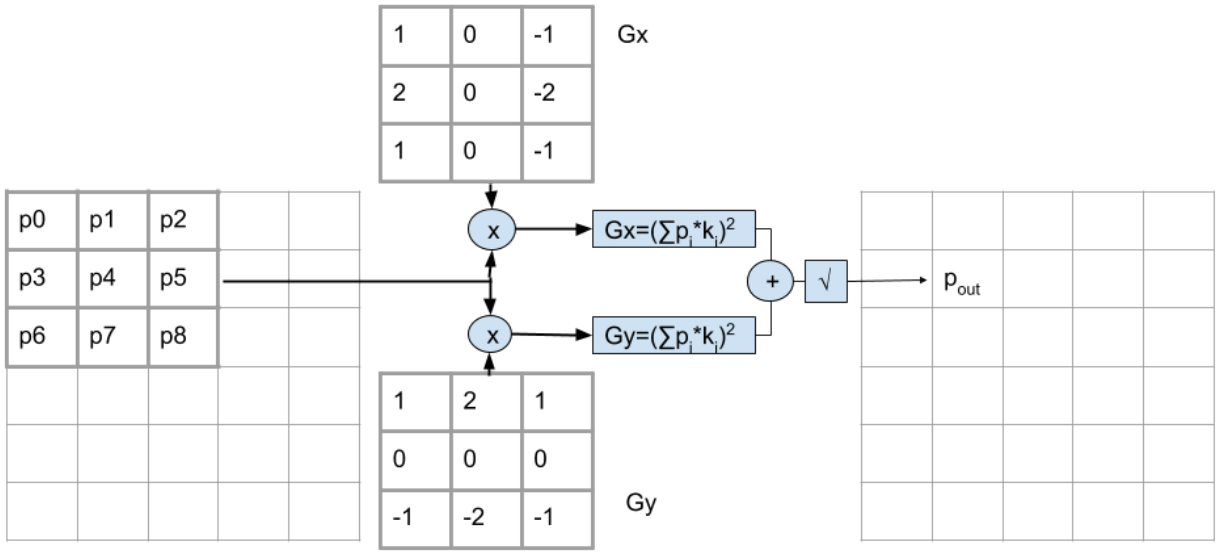


**Figure 1.** Sobel kernels to detect horizontal and vertical edges respectively.

For the Gx case, the derivative (gradient) is taken in the x-direction, and the weighted averaging is done in the y-direction. Each of these 2-dimensional filters is slid across the image's x and y dimensions, and then the Euclidean norm is taken to compute the magnitude. Areas of high variation (edges and textures) will have large derivative magnitudes and will be highlighted with bright pixel values. Areas of low variability (solid structures) will be dark.

Performing the operation on a text file image is easy in ModelSim, but moving the design to an FPGA, specifically the Basys 3, adds a great deal of complexity. To accomplish this, the team designed BRAM storing of input and output values, a Sobel buffer to feed in the 9 required values for the operator, and a VGA controller to output to a display.

# 2. Final Features

- Static bitmap image instantiated in BRAM IP (320x240 BMP image in Black and White)
- BRAM buffering of the entire image: 76800 8-bit pixel values
- Sobel buffering of the 9 pixel values needed by the Sobel kernel to compute new Pixel_out.
- Sobel kernel that convolves and computes the horizontal and vertical image gradients. It takes the absolute value of the sum of the gradient outputs.
- Output buffering and generation of Hsync and Vsync values using BRAM
- VGA output to screen (640X480)

# 3. Target Specifications

- Master CLK at 25 MHz
- VGA output at 60 Hz, using appropriate Hsync and Vsync signals
- Output image will be a 320x240 resolution image, displayed in the top left quadrant of the display
- Output Image is to be Black and White

# 4. Design and Micro Architectures

*Overall Architecture*

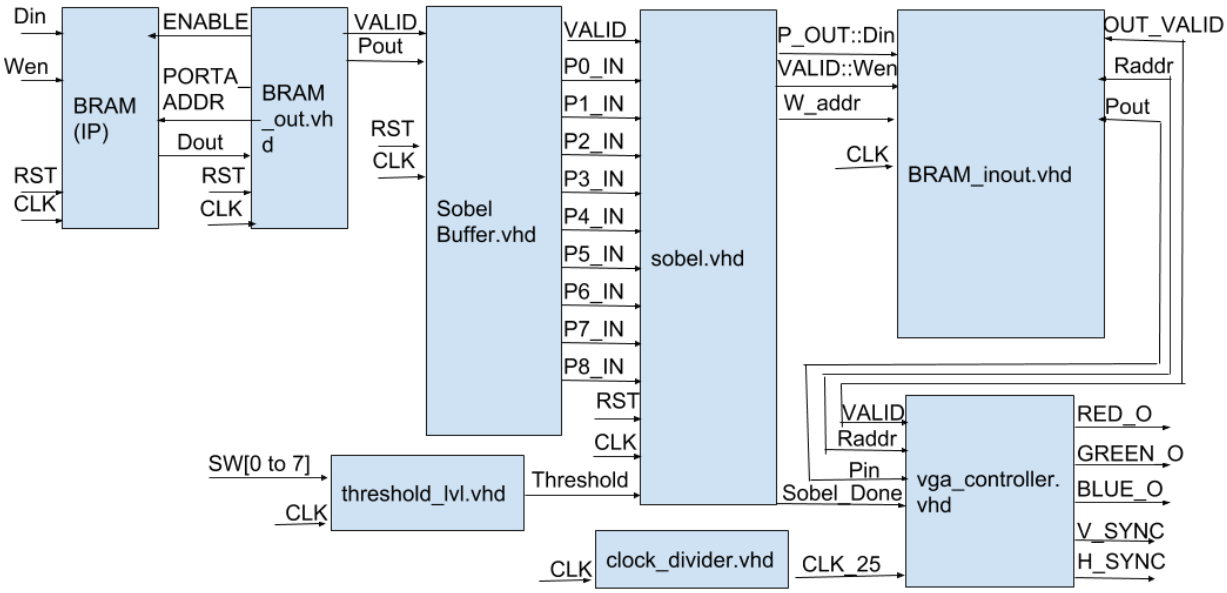


**Figure 2.** Overall architecture for the top level Sobel edge detector design

## *4.1 Cluster 1: BRAM OUT*

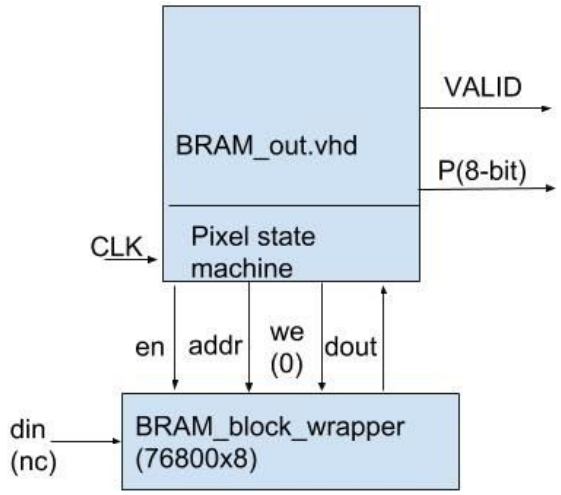


**Figure 3 (a).** BRAMout.vhd


**Figure 3(b)**. Zero padded image, Original image depicted by the blue boxed and the zero pad pixels colored yellow.

The memory space is read-only with a 12-bit width and 76800 depth, which is about a third of the 1800 Kilobit BRAM on the Basys 3. For testing, the memory is initialized with data from a 320x240 gray-scale image (8 bit pixel data) in a .coe file generated from MATLAB. The .coe file is a 76800x1 column vector of 8-bit data. The memory is enabled “always on” so that the VGA_input block can always read. BRAM_out.vhd reads pixels from a flattened (1D) 320X240 image array stored BRAM IP with address range from 0 to 76799. BRAMout uses a state machine to output each pixel individually from the 3x3 sobel block. It also uses zero padding on edges to make sure that the output image stays the same size (Please see figure 3b). Specifically, There are nine cases involved in determining a correctly zero-padded image. The blue boxes depict the original 320x240 image, and the yellow edges depic the zero padded values. This new zero padded image will be streamed to the sobel_buffer block (shown in Figure 3B).

## *4.2 Cluster 2: Sobel buffer*

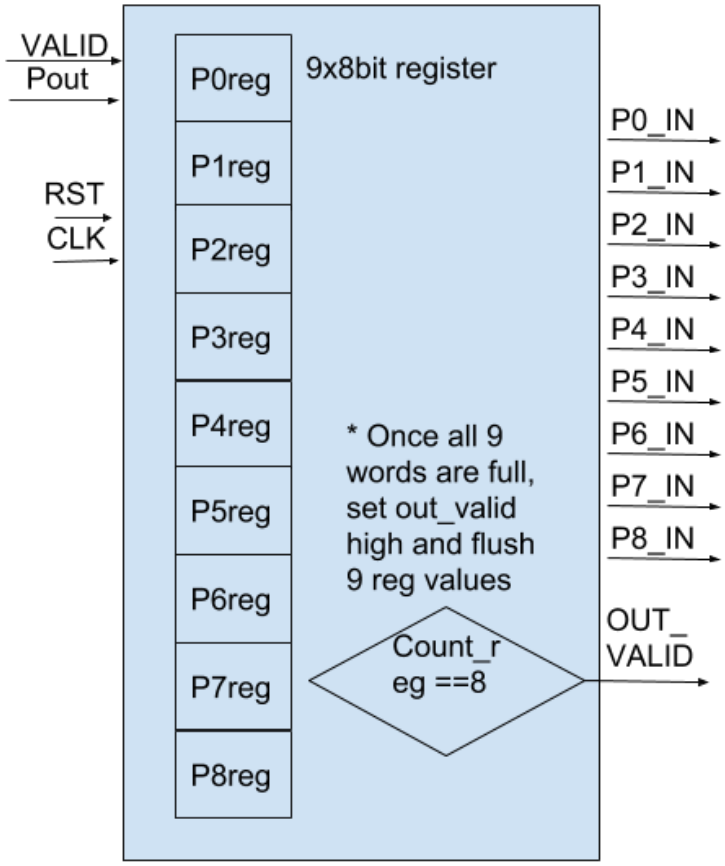


**Figure 4.** Sobel buffer to read and hold 9 pixels from BRAMout.vhd (which provides a zero-padded stream of pixels).

The Sobel buffer block takes the serial zero-padded pixels from the BRAMout.vhd and stores them until all 9 pixels needed for the Sobel operator are ready. Once this happens, the 9 pixels are output along with a out_valid flag set to high for the Sobel block. The 9 registers holding the pixels are flushed,out_valid flag is set low until the next 9 registers are read in.

## *4.3 Cluster 3: Sobel operator*

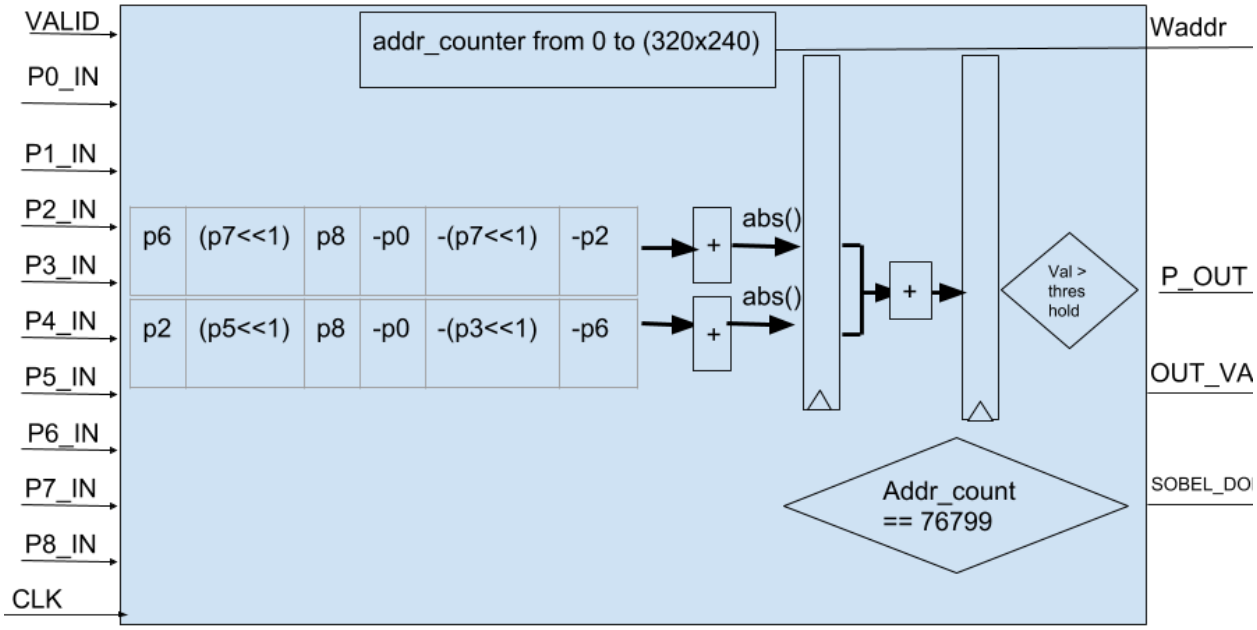


**Figure 5.** Sobel Microarchitecture

This micro architecture is the core of the project. Given the 9 buffered pixel values from the previous block, the Sobel operator in the x- and y-directions are computed. The output is computed by taking the sum of the absolute values of the operator in order to avoid computing the Euclidean norm. Since the Sobel kernel is convolution involves multiplying with 0, 1,-1,2 or-2, we can simplify the total number of operations to compute one [ixle out to the example shown in Figure 5. To multiply by 2 we shift-left by 1. Once the Gx and Gy gradients are computed, the sum of absolute values is a simple addition with removal of the sign bit, whereas the Euclidean norm would involve squaring the operators, adding them together, and taking the square root of the result. The latter is much more costly in time and area. The sobel operator also sends a logic signal to the VGA_controller once it has completed all 76800 valid pixels out, aka the BRAM_inout RAM is completely full. It also sends the Waddr to the BRAMio block so that it knows the address to where it needs to store the output pixel.

## *4.4 Cluster 4: BRAM_IO*

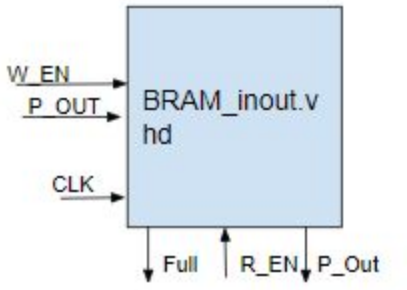


**Figure 6**. BRAM IO Block Diagram

The BRAM IO is non-initialized memory space with 8-bit width and 76800 depth. The memory is single port read/write; the Sobel operator writes to this BRAM block, and the output VGA controller reads from this BRAM block. The sobel operator writes to this block the image data from left to right on each row, going down the rows of the image. Because of this, BRAM IO requires Write Enable (the valid signal from the Sobel operator) and Read Enable from the VGA controller. Once all of the 76800 pixel outputs have been received by this block, The upstream sobel_top.vhd sends sobel_done signal to the VGA controller so that the VGA controller can now start reading the pixels out of the BRAM_IO block.

## *4.5 Cluster 5: VGA Controller*

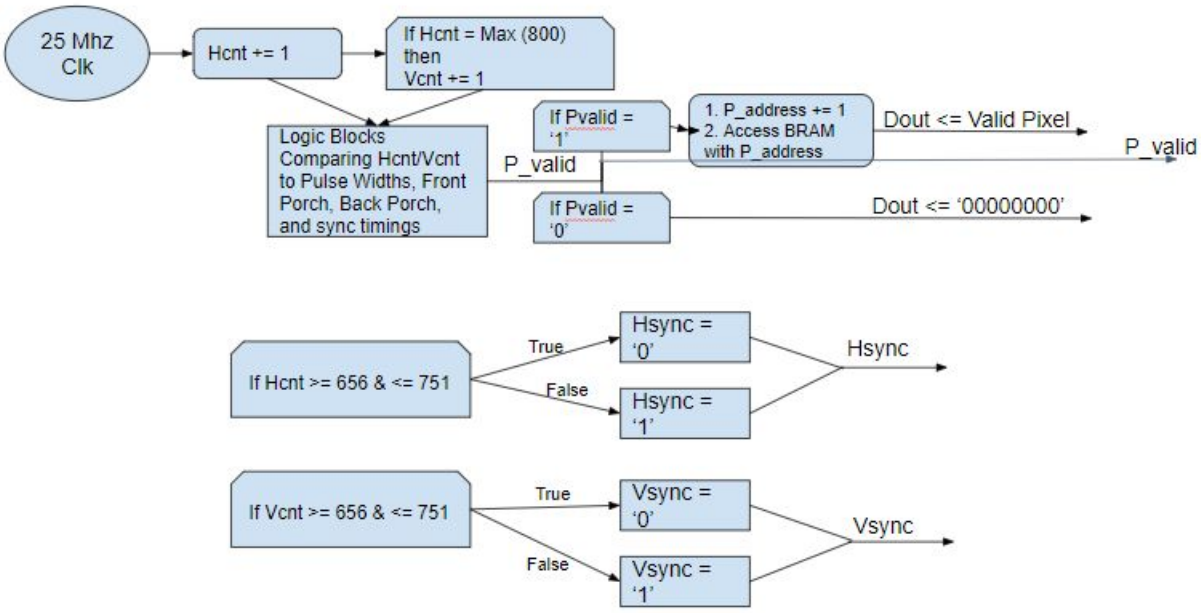


**Figure 7.** VGA Controller Diagram

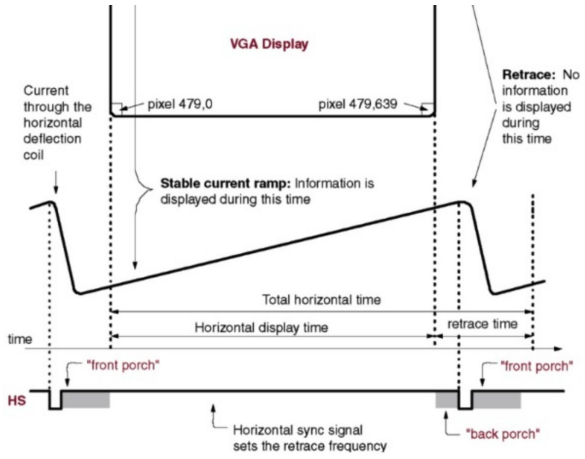


**Figure 8**. Sync Timing

This block reads in data from BRAM IO after the Sobel operator has written to it. The input image is specified as only 320x240 pixels because of memory constraints. The VGA output here is 640x480, so anywhere but the top-left quadrant will send black pixels (“00000000”). The VGA controller has simple gate logic to infer when to set Hsync and Vsync high according to the second image in this sub chapter.

# 5. Implementation Battles and Solutions

## *5.1 Input Solution*

The first implementation battle the team encountered was how to input images into the Basys 3. Various methods were proposed, including UART to read from a USB, storing the image in BRAM, and inputting from a VGA camera. All of these methods require different buffering techniques. If values are streamed from a VGA camera, the values need to be buffered every two rows (using a FIFO queue) to compute the 3x3 Sobel operator on the image. If the image is stored in BRAM, each group of 9 pixels can be accessed at any time, so no queue is needed.

Given the slow speed of UART and the low early testability of a VGA camera, the team decided to work with BRAM. This made the FIFO implementation obsolete, but improved the overall architecture by reusing the BRAM buffering for both the input and output.

## *5.2 Clocking*

Another implementation battle involved the overall clock rate for the system. VGA requires a 25 MHz clock, but the Basys 3 can clock up to 100MHz. The Sobel computation before VGA takes input around 76,800 clock cycles to complete, but at 100MHz this is only 3ms. Reducing the clock frequency by a factor of 4 will increase this to 12ms, which is still real-time for this application.

The team decided on 25MHz for the system to have a uniform clock rate. A clock divider is needed to increase the clock period from 10ns to 40ns, but this would have been necessary for the VGA controller regardless.

# 6. Testbenches

A separate testbench was created for each of the micro-architectures in this project.

## *6.1 Cluster 1: BRAM Out*

Since we are using the IP generator tool on Vivado for the BRAM_block_wrapper, we had to simulate our BRAM_out block on Vivado using the supplied .coe file of our image. Checking the Vivado simulation waveform, we can verify that it is correct by checking the address of each clock cycle, and the valid1-3 flags, which determine whether the current pixel is a padded value or not, are also used to verify correct functional operation.

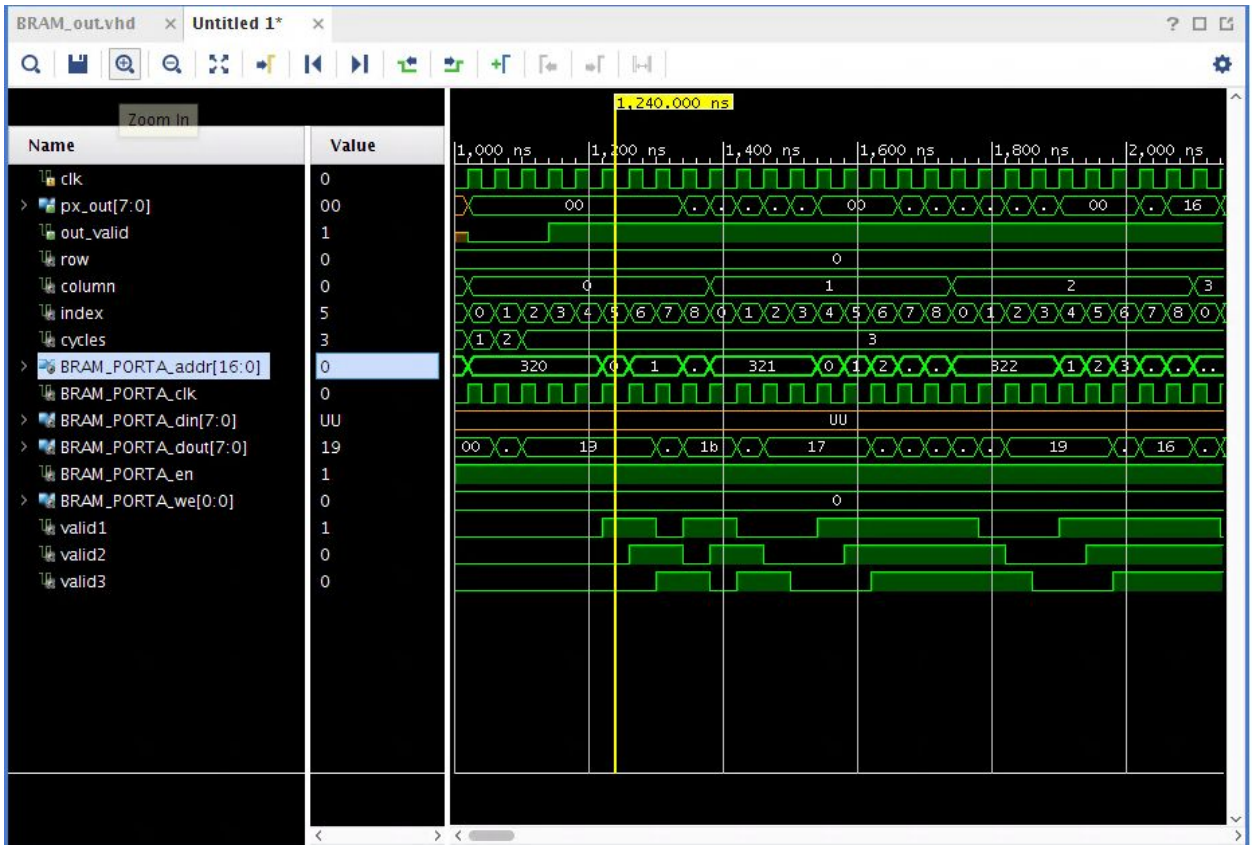


**Figure 9.** BRAM_out.vhd testbench results

## *6.2 Cluster 2: Sobel Buffer*

The test bench for the sobel buffer was written with an input file that has pixel data that is an input from BRAM_out. The input file is formatted such that the each line has an 8-bit pixel value, fed in serially from BRAM_out. The Sobel buffer stores each pixel value until it gathers all 9, then outputs the 9 8-bit pixel values to an output file. The 9 pixel values correspond to those needed for the 3x3 matrix for convolution. The output file is formatted such that the 9 pixel values are separated by commas on one line, and only prints to the file if the output is valid. A reference file was created from matlab and the comparison can be seen below of the output file and the output reference file.

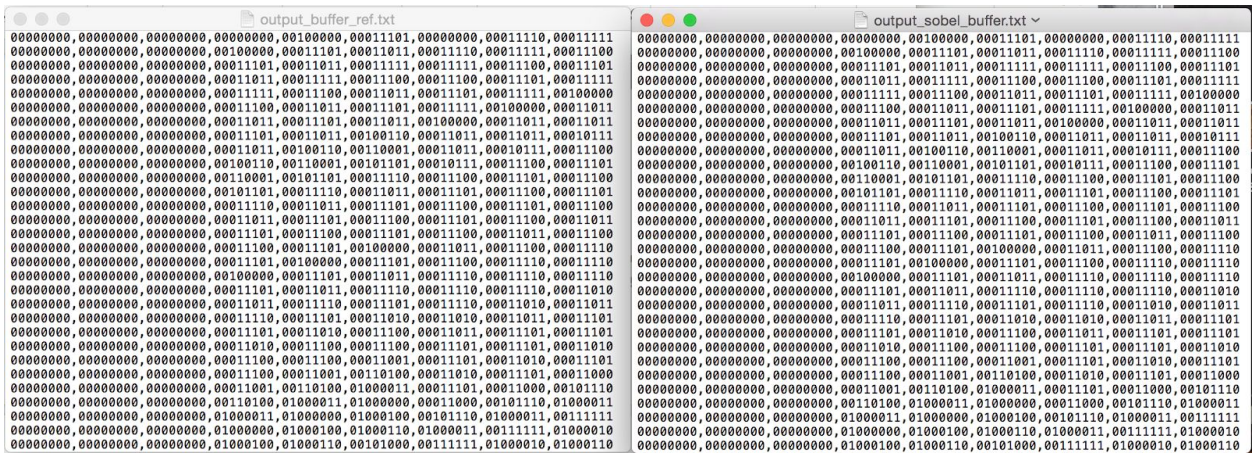


**Figure 10.** Sobel Buffer Output Reference File and Output file.

## *6.3 Cluster 3: Sobel*

The test bench for the sobel image used an input text file where each line was a 9 pixel Sobel block. The text file was generated using a test image in MATLAB. The testbench also outputs the sobel pixel results into a text file. The text file was then reconstructed into an image using MATLAB. The original and sobel images are shown above. The output image verifies the proper functionality of the sobel block.

**Figure 11 (a).** Original test image for Sobel

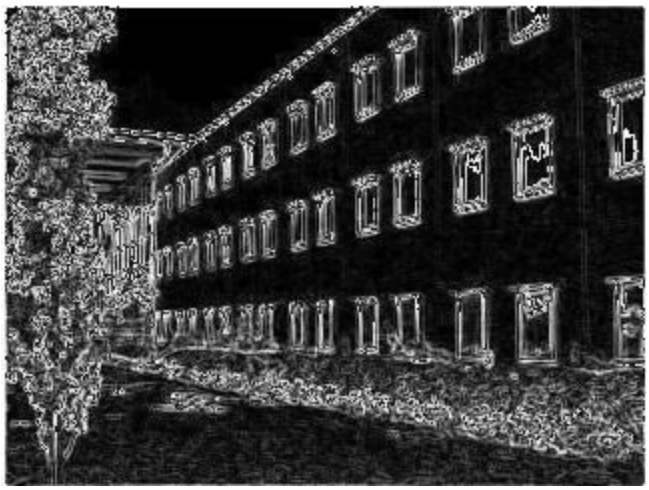

**Figure 11 (b).** Verified Image of Sobel

The following is the test image input and ouput from tb_sobel.vhd, as verified with Yanshen:

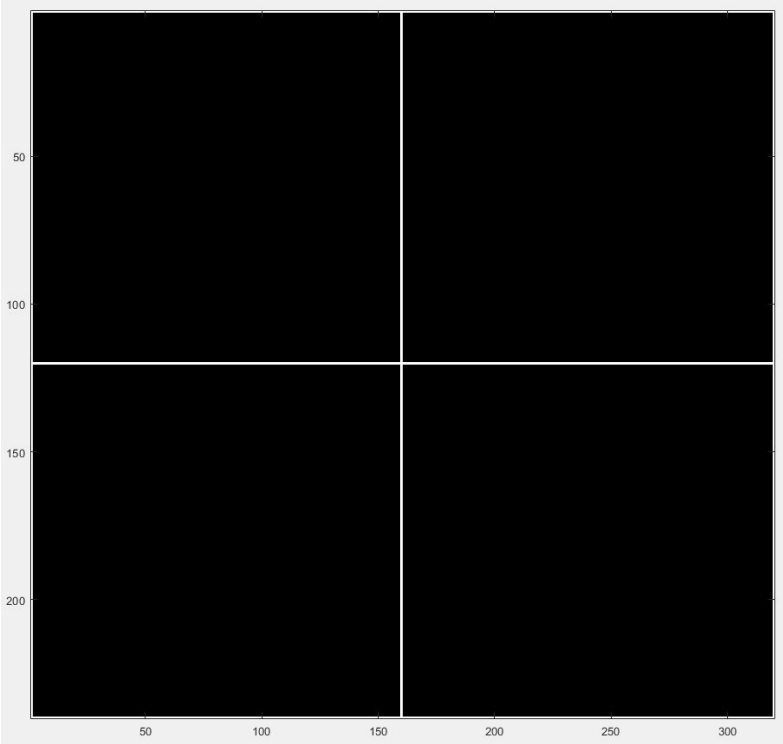


**Figure 12 (a).**Original test image suggested by Yanshen

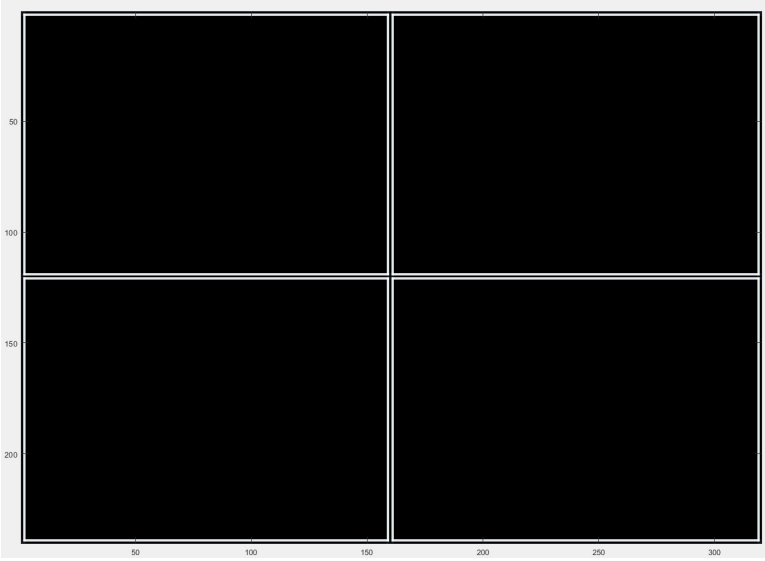


**Figure 12 (b).**Verified Image of output by the testbench for the sobel_top.vhd file: tb_sobel.vhd

## *6.4 Cluster 4: VGA Controller*

The VGA controller is tested by creating a BRAM block with pre-initialized memory from file. The BRAM is then instantiated in the VGA block for testing purposes only, in the actual design this connection will be done in the top level. The VGA controller is tested for its ability to access the

BRAM and set the Vsync and Hsync signals correctly. This testbench input file only provides the reset signal levels, such that it sets the reset high for 2 cycles then low for the necessary amount of time. The wave-form can be seen below. The Vsync display time is approximately 15.3 ms, which corresponds to a refresh rate of approximately 60 Hz. One can also see the pixel out value is always 0 for the second half of the Vsync Cycle due to us only displaying a 320x240 image on a 640x480 display. Specifically the top-right, bottom-left and bottom-right quadrants are sent a blank pixel (“00000000”)

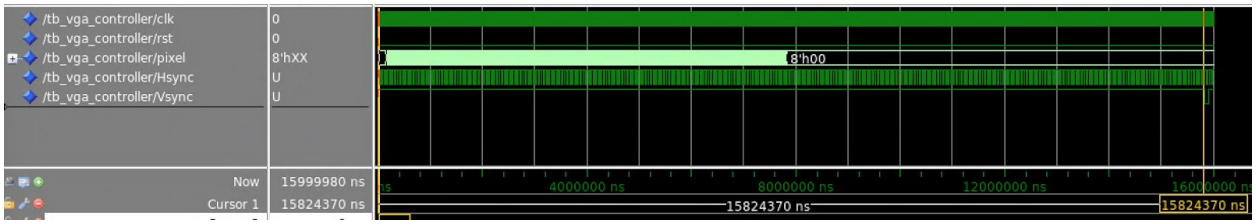


**Figure 13. VGA Controller Test Bench**

## *6.5 Overall Architecture File*

The top level controller is tested very simply. It reads from an input text file that gives it reset values, the first 5 cycles are reset high, and then the rest is set to reset low. The Testbench gives an input clock of 100 MHz, which is of course divided by the clk_divider block. Because the memory is pre-initialized, there is no other inputs.

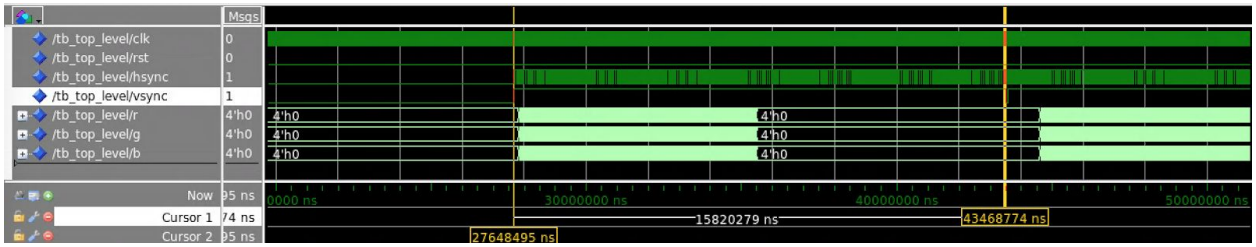


**Figure 14.** Top Level Simulation Results

The screenshot shows the two cursors placed where the Vertical Sync Signal First goes high, then goes low. This is the total display time, which should be approximately 15.36 ms, which equates to a Refresh Rate of ~ 60 Hz. The difference in ns between going high and ow of our Vsync is 43468744 - 27648495 = 0.01582s or 15.82 ms which equations to a refresh rate of approximately 60 Hz. It can also be seen that RGB only has values for the first half of the Vsync and this makes sense because we are displaying a 320x240 image, which is only ½ of the vertical length of the screen.
Before the Vsync goes high the R,G,B outputs zero because the preprocessing of the image is not complete. The VGA controller does not become active and start sending R,G,B values until the BRAM_IO block is fully initialized with data from the Sobel operator. The time before the

start of Vsync is the necessary time of preprocessing before the image is displayed through VGA.

# 7. Functional Simulation and Utilization/Timing

*Overall Architecture:*

Timing Summary

```
    WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints      WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints     WPWS(ns)
TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
    -------      -------  ---------------------  -------------------      -------      -------  ---------------------  -------------------     --------
--------  ----------------------  --------------------
      3.631        0.000                      0                   65        0.103        0.000                      0                   65
4.500        0.000                      0                   34

All user specified timing constraints are met.
```

Logic Elements

```
+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs              |  551 |     0 |     20800 |  2.65 |
|   LUT as Logic          |  551 |     0 |     20800 |  2.65 |
|   LUT as Memory         |    0 |     0 |      9600 |  0.00 |
| Slice Registers         |  426 |     0 |     41600 |  1.02 |
|   Register as Flip Flop |  426 |     0 |     41600 |  1.02 |
|   Register as Latch     |    0 |     0 |     41600 |  0.00 |
| F7 Muxes                |   16 |     0 |     16300 |  0.10 |
| F8 Muxes                |    0 |     0 |      8150 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

```
+-------------------+------+-------+-----------+-------+
|     Site Type     | Used | Fixed | Available | Util% |
+-------------------+------+-------+-----------+-------+
| Block RAM Tile    |   31 |     0 |        50 | 62.00 |
|   RAMB36/FIFO*    |   31 |     0 |        50 | 62.00 |
|     RAMB36E1 only |   31 |       |           |       |
|   RAMB18          |    0 |     0 |       100 |  0.00 |
+-------------------+------+-------+-----------+-------+
```

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Bonded IOB | 16 | 16 | 106 | 15.09 |
| IOB Master Pads | 7 | | | |
| IOB Slave Pads | 8 | | | |
| Bonded IPADs | 0 | 0 | 10 | 0.00 |
| Bonded OPADs | 0 | 0 | 4 | 0.00 |
| PHY_CONTROL | 0 | 0 | 5 | 0.00 |
| PHASER_REF | 0 | 0 | 5 | 0.00 |
| OUT_FIFO | 0 | 0 | 20 | 0.00 |
| IN_FIFO | 0 | 0 | 20 | 0.00 |
| IDELAYCTRL | 0 | 0 | 5 | 0.00 |
| IBUFDS | 0 | 0 | 104 | 0.00 |
| GTPE2_CHANNEL | 0 | 0 | 2 | 0.00 |
| PHASER_OUT/PHASER_OUT_PHY | 0 | 0 | 20 | 0.00 |
| PHASER_IN/PHASER_IN_PHY | 0 | 0 | 20 | 0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY | 0 | 0 | 250 | 0.00 |
| IBUFDS_GTE2 | 0 | 0 | 2 | 0.00 |
| ILOGIC | 0 | 0 | 106 | 0.00 |
| OLOGIC | 0 | 0 | 106 | 0.00 |

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| DSPs | 3 | 0 | 90 | 3.33 |
| DSP48E1 only | 3 | | | |

Power:

| | |
|---|---|
| Total On-Chip Power (W) | 0.073 |
| Dynamic (W) | 0.003 |
| Device Static (W) | 0.070 |
| Effective TJA (C/W) | 5.0 |
| Max Ambient (C) | 84.6 |
| Junction Temperature (C) | 25.4 |
| Confidence Level | Low |
| Setting File | --- |
| Simulation Activity File | --- |
| Design Nets Matched | NA |

## *7.1 Cluster 1: BRAM out:*

Timing Summary

Design Timing Summary

| WNS(ns) | TNS(ns) | TNS Failing Endpoints | TNS Total Endpoints | WHS(ns) | THS(ns) | THS Failing Endpoints | THS Total Endpoints | WPWS(ns) | TPWS(ns) | TPWS Failing Endpoints | TPWS Total Endpoints |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 27.985 | 0.000 | 0 | 576 | 0.068 | 0.000 | 0 | 576 | 4.500 | 0.000 | 0 | 169 |

Logic

1. Slice Logic

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Slice LUTs | 289 | 0 | 20800 | 1.39 |
| LUT as Logic | 289 | 0 | 20800 | 1.39 |
| LUT as Memory | 0 | 0 | 9600 | 0.00 |
| Slice Registers | 130 | 0 | 41600 | 0.31 |
| Register as Flip Flop | 130 | 0 | 41600 | 0.31 |
| Register as Latch | 0 | 0 | 41600 | 0.00 |
| F7 Muxes | 16 | 0 | 16300 | 0.10 |
| F8 Muxes | 0 | 0 | 8150 | 0.00 |

3. Memory

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Block RAM Tile | 19 | 0 | 50 | 38.00 |
| RAMB36/FIFO* | 19 | 0 | 50 | 38.00 |
| RAMB36E1 only | 19 | | | |
| RAMB18 | 0 | 0 | 100 | 0.00 |

Power

| | |
|---|---|
| Total On-Chip Power (W) | 0.074 |
| Dynamic (W) | 0.003 |
| Device Static (W) | 0.071 |
| Effective TJA (C/W) | 5.0 |
| Max Ambient (C) | 84.6 |
| Junction Temperature (C) | 25.4 |
| Confidence Level | Medium |
| Setting File | --- |
| Simulation Activity File | --- |
| Design Nets Matched | NA |

## *7.2 Cluster 2: Sobel Buffer*

LUTs

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Slice LUTs | 35 | 0 | 20800 | 0.17 |
| LUT as Logic | 35 | 0 | 20800 | 0.17 |
| LUT as Memory | 0 | 0 | 9600 | 0.00 |
| Slice Registers | 178 | 0 | 41600 | 0.43 |
| Register as Flip Flop | 178 | 0 | 41600 | 0.43 |
| Register as Latch | 0 | 0 | 41600 | 0.00 |
| F7 Muxes | 0 | 0 | 16300 | 0.00 |
| F8 Muxes | 0 | 0 | 8150 | 0.00 |

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Bonded IOB | 83 | 0 | 106 | 78.30 |
| IOB Master Pads | 40 | | | |
| IOB Slave Pads | 41 | | | |
| Bonded IPADs | 0 | 0 | 10 | 0.00 |
| Bonded OPADs | 0 | 0 | 4 | 0.00 |
| PHY_CONTROL | 0 | 0 | 5 | 0.00 |
| PHASER_REF | 0 | 0 | 5 | 0.00 |
| OUT_FIFO | 0 | 0 | 20 | 0.00 |
| IN_FIFO | 0 | 0 | 20 | 0.00 |
| IDELAYCTRL | 0 | 0 | 5 | 0.00 |
| IBUFDS | 0 | 0 | 104 | 0.00 |
| GTPE2_CHANNEL | 0 | 0 | 2 | 0.00 |
| PHASER_OUT/PHASER_OUT_PHY | 0 | 0 | 20 | 0.00 |
| PHASER_IN/PHASER_IN_PHY | 0 | 0 | 20 | 0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY | 0 | 0 | 250 | 0.00 |
| IBUFDS_GTE2 | 0 | 0 | 2 | 0.00 |
| ILOGIC | 0 | 0 | 106 | 0.00 |
| OLOGIC | 0 | 0 | 106 | 0.00 |

Power

| | |
|---|---|
| Total On-Chip Power (W) | 0.539 |
| Dynamic (W) | 0.468 |
| Device Static (W) | 0.071 |
| Effective TJA (C/W) | 5.0 |
| Max Ambient (C) | 82.3 |
| Junction Temperature (C) | 27.7 |
| Confidence Level | Low |
| Setting File | --- |
| Simulation Activity File | --- |
| Design Nets Matched | NA |

## *7.3 Cluster 3: Sobel Top*

Logic

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Slice LUTs | 178 | 0 | 20800 | 0.86 |
| LUT as Logic | 178 | 0 | 20800 | 0.86 |
| LUT as Memory | 0 | 0 | 9600 | 0.00 |
| Slice Registers | 68 | 0 | 41600 | 0.16 |
| Register as Flip Flop | 68 | 0 | 41600 | 0.16 |
| Register as Latch | 0 | 0 | 41600 | 0.00 |
| F7 Muxes | 0 | 0 | 16300 | 0.00 |
| F8 Muxes | 0 | 0 | 8150 | 0.00 |

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Slice | 69 | 0 | 8150 | 0.85 |
| SLICEL | 50 | 0 | | |
| SLICEM | 19 | 0 | | |
| LUT as Logic | 178 | 0 | 20800 | 0.86 |
| using O5 output only | 2 | | | |
| using O6 output only | 119 | | | |
| using O5 and O6 | 57 | | | |
| LUT as Memory | 0 | 0 | 9600 | 0.00 |
| LUT as Distributed RAM | 0 | 0 | | |
| LUT as Shift Register | 0 | 0 | | |
| LUT Flip Flop Pairs | 22 | 0 | 20800 | 0.11 |
| fully used LUT-FF pairs | 12 | | | |
| LUT-FF pairs with one unused LUT output | 10 | | | |
| LUT-FF pairs with one unused Flip Flop | 10 | | | |
| Unique Control Sets | 7 | | | |

Power

| | |
|---|---|
| Total On-Chip Power (W) | 13.523 (Junction temp exceeded!) |
| Dynamic (W) | 13.313 |
| Device Static (W) | 0.210 |
| Effective TJA (C/W) | 5.0 |
| Max Ambient (C) | 17.4 |
| Junction Temperature (C) | 92.6 |
| Confidence Level | Low |
| Setting File | --- |
| Simulation Activity File | --- |
| Design Nets Matched | NA |

## *7.4 Cluster 4: VGA Controller:*

Timing Summary

```
------------------------------------------------------------------------------------------------
| Design Timing Summary
| ---------------------
------------------------------------------------------------------------------------------------

    WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints      WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints     WPWS(ns)
TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
    -------      -------  ---------------------  -------------------      -------      -------  ---------------------  -------------------     --------
--------  ----------------------  -------------------
     35.482        0.000                      0                   61        0.174        0.000                      0                   61
4.500        0.000                       0                   32

All user specified timing constraints are met.
```

## Logic

```
+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs              |   34 |     0 |     20800 |  0.16 |
|   LUT as Logic          |   34 |     0 |     20800 |  0.16 |
|   LUT as Memory         |    0 |     0 |      9600 |  0.00 |
| Slice Registers         |   31 |     0 |     41600 |  0.07 |
|   Register as Flip Flop |   31 |     0 |     41600 |  0.07 |
|   Register as Latch     |    0 |     0 |     41600 |  0.00 |
| F7 Muxes                |    0 |     0 |     16300 |  0.00 |
| F8 Muxes                |    0 |     0 |      8150 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

```
+----------------------------+------+-------+-----------+-------+
|         Site Type          | Used | Fixed | Available | Util% |
+----------------------------+------+-------+-----------+-------+
| Bonded IOB                 |   20 |     1 |       106 | 18.87 |
|   IOB Master Pads          |    9 |       |           |       |
|   IOB Slave Pads           |   11 |       |           |       |
| Bonded IPADs               |    0 |     0 |        10 |  0.00 |
| Bonded OPADs               |    0 |     0 |         4 |  0.00 |
| PHY_CONTROL                |    0 |     0 |         5 |  0.00 |
| PHASER_REF                 |    0 |     0 |         5 |  0.00 |
| OUT_FIFO                   |    0 |     0 |        20 |  0.00 |
| IN_FIFO                    |    0 |     0 |        20 |  0.00 |
| IDELAYCTRL                 |    0 |     0 |         5 |  0.00 |
| IBUFDS                     |    0 |     0 |       104 |  0.00 |
| GTPE2_CHANNEL              |    0 |     0 |         2 |  0.00 |
| PHASER_OUT/PHASER_OUT_PHY  |    0 |     0 |        20 |  0.00 |
| PHASER_IN/PHASER_IN_PHY    |    0 |     0 |        20 |  0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY |    0 |     0 |       250 |  0.00 |
| IBUFDS_GTE2                |    0 |     0 |         2 |  0.00 |
| ILOGIC                     |    0 |     0 |       106 |  0.00 |
| OLOGIC                     |    0 |     0 |       106 |  0.00 |
+----------------------------+------+-------+-----------+-------+
```

## Power

```
+--------------------------+-------+
| Total On-Chip Power (W)  | 0.071 |
| Dynamic (W)              | 0.000 |
| Device Static (W)        | 0.070 |
| Effective TJA (C/W)      | 5.0   |
| Max Ambient (C)          | 84.6  |
| Junction Temperature (C) | 25.4  |
| Confidence Level         | Low   |
| Setting File             | ---   |
| Simulation Activity File | ---   |
| Design Nets Matched      | NA    |
+--------------------------+-------+
```

# 8. Work Distribution

*8.1 BRAM Out*
Top Level - Hariank
Testbench - Hariank
Sample input - Hariank

*8.2 Sobel buffer*
Top Level - Sumit
Testbench - Sumit
Sample input - Sumit

*8.3 Sobel operator*
Top Level - Jaison and Alex
Testbench -Jaison and Alex
Sample input - Alex

*8.4 BRAM IO*
Top Level - Sumit
Testbench - Sumit
Sample input - Sumit

*8.5 VGA Controller*
Top Level - Jaison/Sumit
Testbench - Sumit
Sample input - Sumit

*8.6 Clock Divider*
Top Level - Jaison
Testbench - Jaison
Sample input - Jaison

*8.7 Top Level Controller*
Top Level - Sumit
Testbench - Sumit
Sample input - Sumit

*8.8 Final Verification, Paper, and Demo*
Alex, Hariank, Jaison, Sumit

*8.9 FIFO **(Deprecated)***
Top Level - Alex

# 9. References


1. A Descriptive Algorithm for Sobel Image Edge Detection, O. R. Vincent, O. Folorunso Proceedings of Informing Science & IT Education Conference (InSITE) 2009, http://proceedings.informingscience.org/InSITE2009/InSITE09p097-107Vincent613.pdf
2. Kanopoulos, Nick. “Design of an Image Edge Detection Filter Using the Sobel Operator.” IEEE Journal of Solid-State Circuits, vol. 23, no. 2, Apr. 1988, pp. 358–367., ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=996.
3. OV7670 Datasheet: https://www.voti.nl/docs/OV7670.pdf
4. Basys3 Datasheet: https://reference.digilentinc.com/_media/basys3:basys3_rm.pdf