Demonstration Agenda:

1. tb\_addr\_calc.sv
   1. Show our algorithm for calculating the address (zigzag).
   2. Show the waveform of the test bench with unsigned decimal mode. The correct output is based on the algorithm introduced. The last output for row\_cnt should be 596 and the col\_cnt should be 792.
2. tb\_win\_buf.sv
   1. Show our algorithm for filling the window buffer
   2. Show the waveform of the test bench with unsigned decimal mode. Show that the window buffer can be filled with the correct order with given condition.
3. tb\_addr\_calc\_win\_buf.sv
   1. Show the waveform of the test bench with unsigned decimal mode. The address calculation block will output the address calculated to SRAM. And then SRAM would send the corresponding data to the window buffer block. And the test bench simulate this process.
4. tb\_gaussian\_blur.sv
   1. Show the waveform of the test bench with unsigned decimal mode. The input is a 3x3 matrix (from window buffer) and show that the output is just a 8 bit value which is the average value of all the 9 pixels.
5. tb\_gaussian\_addr\_calc.sv
   1. Show the waveform of the test bench with unsigned decimal mode. The input is an enable signal. Once the signal is enabled, the block would count once (x or y). And the pattern is that when x goes to 5, y increases by 1 and x rollover back to 0.
6. tb\_gaussian\_buffer.sv
   1. Show the waveform of the test bench with unsigned decimal mode. The input is a pixel value (from gaussian blur) and the xy coordinate information (from gaussian address calculation). The output is a 6x6 matrix with all the values filled in at the correct position.
7. tb\_sobel\_fir.sv
   1. Show the waveform of the test bench with unsigned decimal mode. The correct version should be that the input is a 6x6 matrix (from the gaussian buffer) and the output should be 2 4x4 matrix given the sobel algorithm below:

|  |  |  |
| --- | --- | --- |
| **-1** | **0** | **1** |

|  |
| --- |
| **-1** |
| **0** |
| **1** |

The first one is the algorithm for x matrix, and the second one is the algorithm for y matrix.

1. tb\_magnitude.sv
   1. Show the waveform of the test bench with unsigned decimal mode. The input should be 2 4x4 matrix (from the sobel filter) and the output should be the absolute values of the input.
2. tb\_gradient.sv
   1. Show the waveform of the test bench with unsigned decimal mode. The input should be 2 4x4 matrix (from the magnitude block) and the output should be a 4x4 matrix with correct x^2+y^2 value calculated.
3. tb\_mag\_wrapper.sv
   1. Show the waveform of the test bench with unsigned decimal mode. This is the combination of the magnitude and gradient blocks. So the input are 2 4x4 matrix (from the sobel filter) and the output is a 4x4 matrix with correct x^2+y^2 value calculated.
4. tb\_output\_addr.sv
   1. Show the waveform of the test bench with unsigned decimal mode. The input should be a 4x4 matrix (from the gradient block) and col\_cnt and row\_cnt (from addr\_calc block) and the output should be a 4x4 matrix and a address calculated to store the matrix in the SRAM.
   2. The 4x4 matrix is created by comparing the values of the input 4x4 matrix with a threshold value (we made it to be 2500 which is 50^2 that has been experimented by a python program). If the value in the input matrix is greater than the threshold, the corresponding pixel should be 255, and 0 otherwise.
5. tb\_edge\_detector.sv
   1. This is the test bench for the whole project. Show the waveform of the test bench with unsigned decimal mode. As shown, it can be seen that the data can be send in, and all the blocks perform as expected. However, for some reason, we cannot get correct output.

Fixed Criteria:

1. Test benches exist for all top-level components and the entire design. The test benches for the entire design can be demonstrated or documented to cover all of the functional requirements given in the design specific success criteria.
2. Test Benches exists for all top level components: tb\_addr\_calc.sv, tb\_win\_buf.sv, tb\_gaussian\_blur.sv, tb\_gaussian\_addr\_calc.sv, tb\_gaussian\_buffer.sv, tb\_sobel\_fir.sv, tb\_magnitude.sv, tb\_mag\_wrapper.sv tb\_gradient.sv, tb\_output\_addr.sv.
3. Entire design synthesizes completely, without any inferred latches, timing arcs, and, sensitivity list warnings.
4. Finished with the main control unit and the wrapper.
5. Run our tb\_MCU and show the waveform with all the data and state changed.
6. File read into the test bench as the form of SRAM
7. SRAM cannot correctly send the corresponding data out, which is our problem for now.
8. Source and mapped version of the complete design behave the same for all test cases. The mapped version simulates without timing errors except at time zero.

a. Source behaved well for all the top level blocks.

b. All blocks except for window buffer, gaussian blur and output address blocks behave the same for source and mapped version.

4. A complete IC layout is produced that passes all geometry and connectivity checks.

The layout is generated with no connectivity warnings or errors.

5. The entire design complies with targets for area, pin count, throughput (if applicable), and clock rate. The final targets for these parameters will be determined by course staff based on your design review. Failure to reach any of the targets will result a score of 1 out of 2 provided that you are within 50% on area, 10% on pin count, and 25% on throughput. Doing worse in any category will result in a score of 0 out of 2.

a. Our targets of area is about 14.2 mm^2 and our actual area is 14.5 mm^2.

b. Clock rate is 500 MHz.

Specific success criteria:

1. Demonstrate by simulation of a Verilog test bench that gaussian block works produce the correct blurred pixel given a 3x3 matrix directly from the window buffer.

***Showed by demonstration agenda 6.***

1. Demonstrate by simulation of a Verilog test bench that sobel filter works produce the correct 4x4 matrix of xy sets given a 6x6 blurred pixel matrix from the window buffer.

***Showed by demonstration agenda 7.***

1. Demonstrate by simulation of a Verilog test bench that gradient calculator works produce the correct squared valued based on the input xy set.

***Showed by demonstration agenda 8, 9 and 10.***

1. Demonstrate by simulation of a Verilog test bench that the address calculation block calculate the correct address so that data w/r can be implemented correctly. And it should make sure that the window buffer stores the correct 8\*8 pixels value.

***Showed by demonstration agenda 1 and 2.***