Midterm Exam ECE 448
Spring 2021
Friday, March 12
15 points

Instructions:
Zip all your deliverables into an archive <last_name>.zip and submit it through Blackboard no later than Friday, March 12, 11:30 AM EST.

The EXAM circuit implements an edge detection filter on a grayscale image:

1. Edge Detection filter

Edge detection is used to extract the edges of a gray scale image. Generally, edges are sharp changes in the intensity of pixels in an image. Since the main characteristic of the edges is sharp transition, they can be easily extracted using High Pass Filters (HPF). These low-level information are used in higher levels of processing for feature extraction and AI applications.

A grayscale image is a matrix of 8-bit unsigned integers, in which each number represents a pixel. Pixel values are integers that range from 0 (black) to 255 (white). These values represent the amount of gray intensity to be displayed for that particular portion of the image.

Input:

M: a grayscale image of dimensions 6 rows by 6 columns. Before the beginning of processing, this image is stored in the internal ROM at locations 0...35. An address at which a pixel with coordinates (x, y) is stored is given by $6 \cdot x+y$, where x=0...5 and y=0...5.

Output:

Z: a grayscale image of dimensions 4 rows by 4 columns. After the end of processing, this image is stored in the internal RAM.

Edge Detection Window:

As you can see in the following figure, this algorithm has a 3*3 sliding window which simply multiplies the constants with the corresponding pixels in the image.

1	2	1
0	0	0
-1	-2	-1

Edge Detection 3*3 Window

Since there are multiplication, addition, and subtraction on 8-bit values, and considering that the final pixel value should be in the range of 0 to 255, there is a high chance of overflow/underflow. In these cases, the architecture should be able to map values greater than 255 to 255, and values less than 0 to 0.

Algorithm:

Pseudocode:

//Reset State//
Done = 0

//Wait on Start signal
Wait until Start = 1

//Start Edge detection (Central Pixel is indexed as (i, j), and M is an input image matrix)

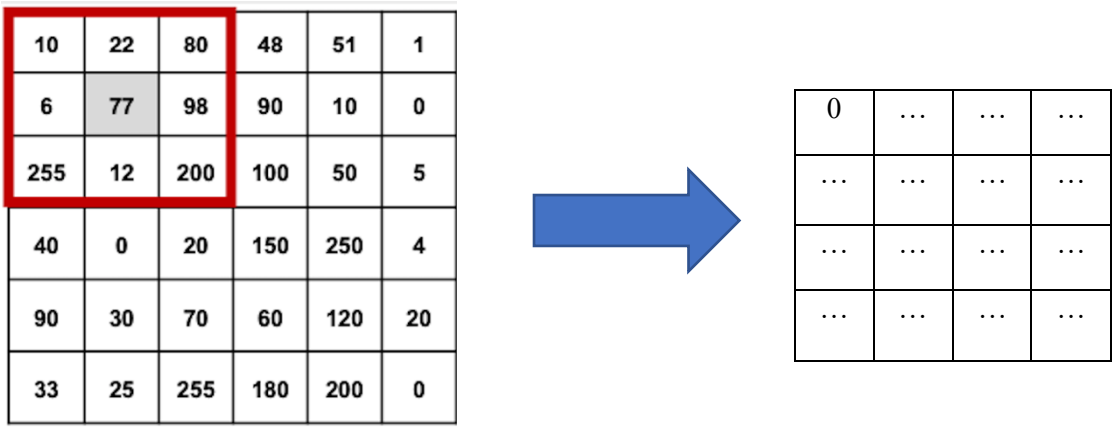
```
for i in 1 to DIM-2
 for j in 1 to DIM-2
   Tmp = M (i-1, j-1) + 2*M (i-1, j) + M (i-1, j+1) - M (i+1, j-1) - 2*M (i+1, j) - M (i+1, j+1)
   if (Tmp < 0) then
     Tmp = 0
   elsif (Tmp > 255)
     Tmp = 255
   end if
   Z (i-1, j-1) = Tmp
 end for
end for
```

//The end
Done = 1

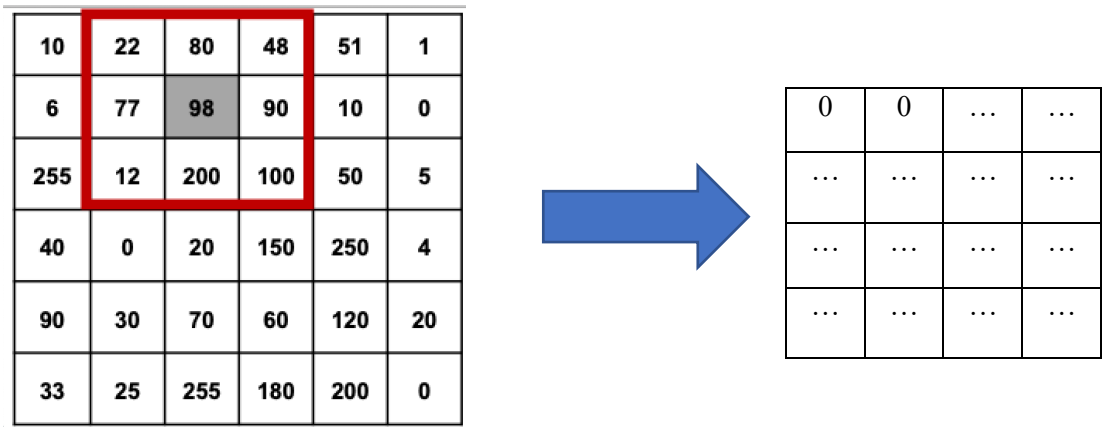
This procedure is illustrated below:

Calculating 1st output pixel:

Calculate (10 + 2*22 + 80 - 255 – 2*12 - 200).
If negative, replace with 0. If greater than 255, replace with 255.



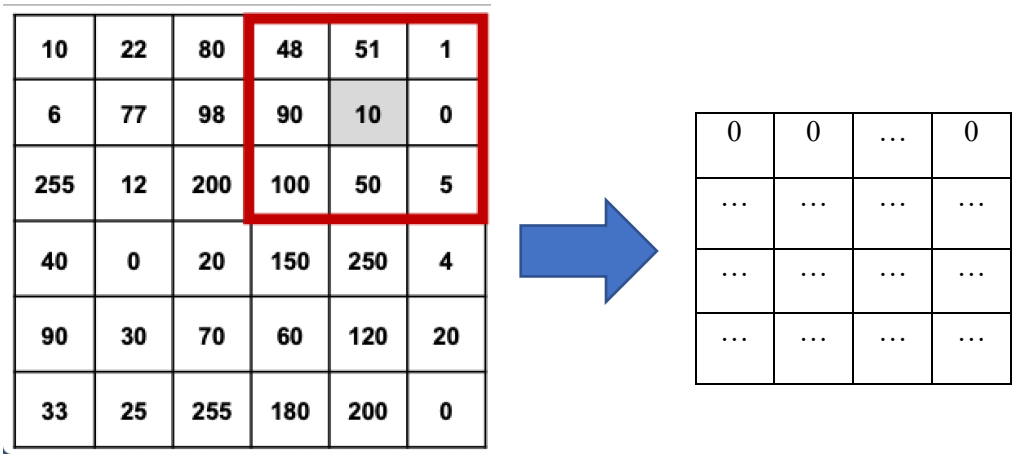
Calculating 2nd output pixel:
Calculate (22 + 2*80 + 48 - 12 – 2*200 -100).
If negative, replace with 0. If greater than 255, replace with 255.



……

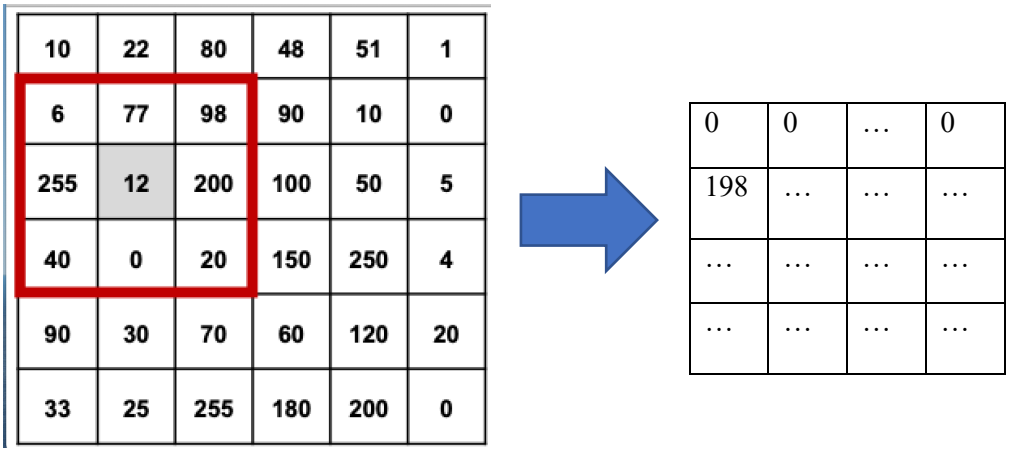
Calculating 4th output pixel:

Calculate (48 + 2*51 + 1 – 100 – 2*50 - 5).
If negative, replace with 0. If greater than 255, replace with 255.



Calculating 5th output pixel:

Calculate (6 + 2*77 + 98 – 40 – 2*0 - 20).
If negative, replace with 0. If greater than 255, replace with 255.



10	22	80	48	51	1
6	77	98	90	10	0
255	12	200	100	50	5
40	0	20	150	250	4
90	30	70	60	120	20
33	25	255	180	200	0

0	0	…	0
198	…	…	…
…	…	…	…
…	…	…	…

Final values of the image pixels:

0	0	0	0
198	173	0	0
255	255	140	0
0	0	0	74

NOTE 1: All memory read operations (ROM, RAM) are synchronous.

For verification, you can easily take a look at the output signal of the Edge RAM after writing your testbench.
After verification, please upload a screenshot of your testbench output waveform to the submission folder.
If you have implemented your design correctly, you should see the following values:

"0, 0, 0, 0, 198, 173, 0, 0, 255, 255, 140, 0, 0, 0, 0, 74"

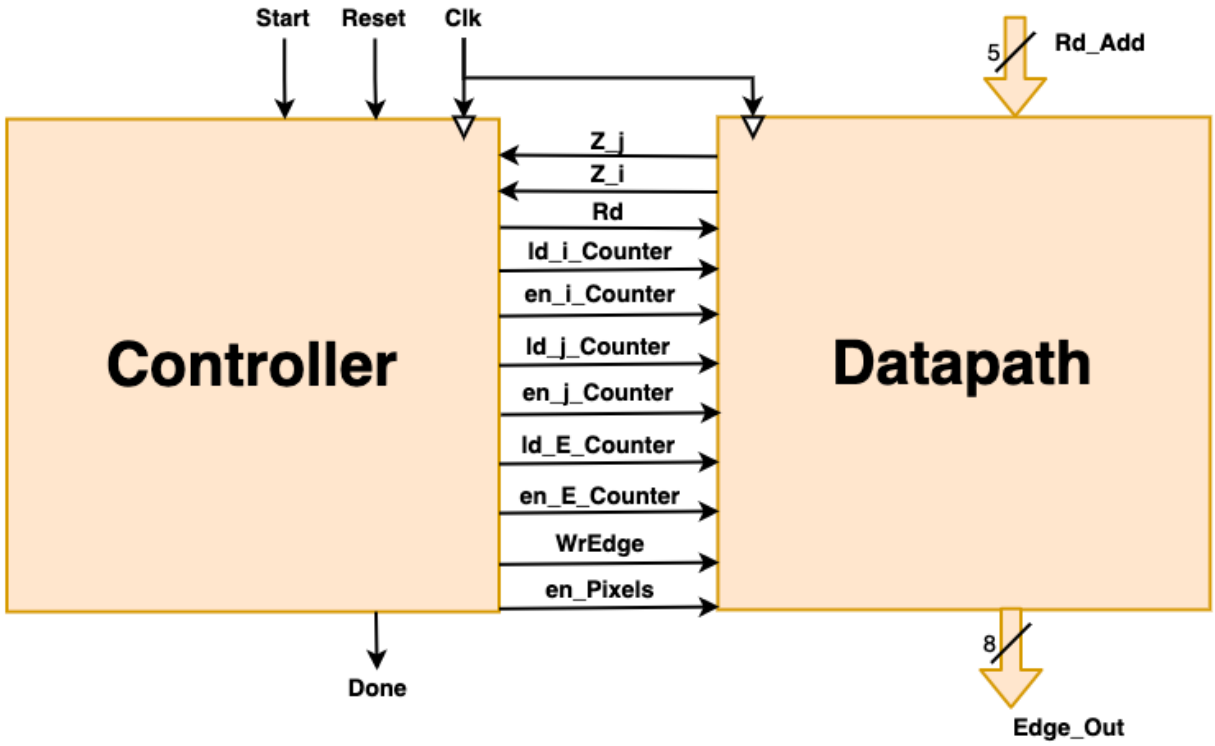
The circuit is specified below using its

- Table of input/output ports
- Interface with the division into the Datapath and Controller
- Block diagram of the Datapath
- ASM chart of the Controller.

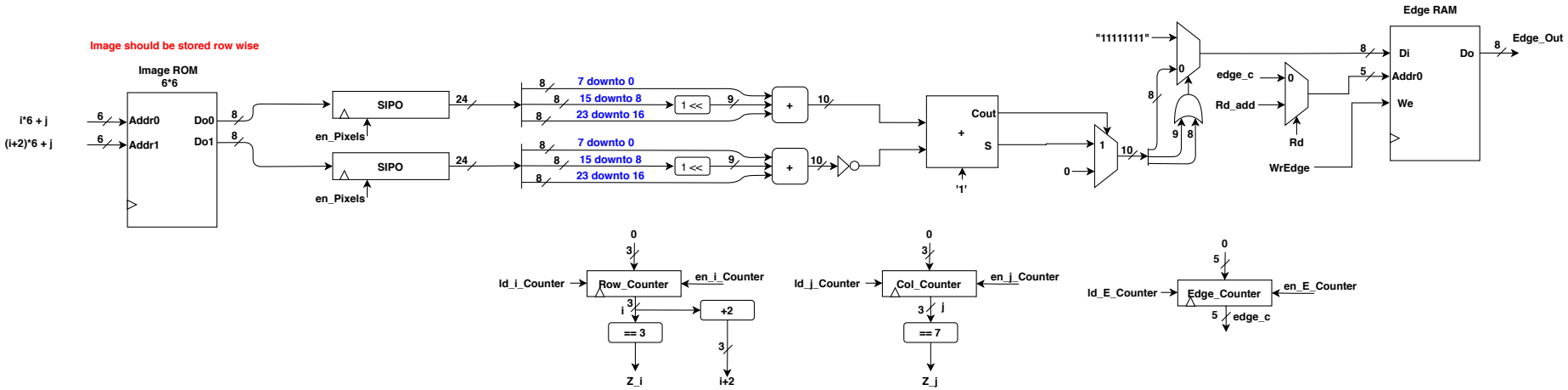
Table of input/output ports:

Port	Width	Meaning
Clk	1	System clock
Reset	1	System reset
Start	1	Start of the edge detection algorithm. Edge detection starts when this signal is asserted.
Edge_Out	8	Output of the RAM storing extracted edges
Rd_Add	5	Address of the Edge RAM. When Done = 1, this signal can be used to read the content of the Edge RAM.
Done	1	Output control signal indicating that the edge detection operation is complete

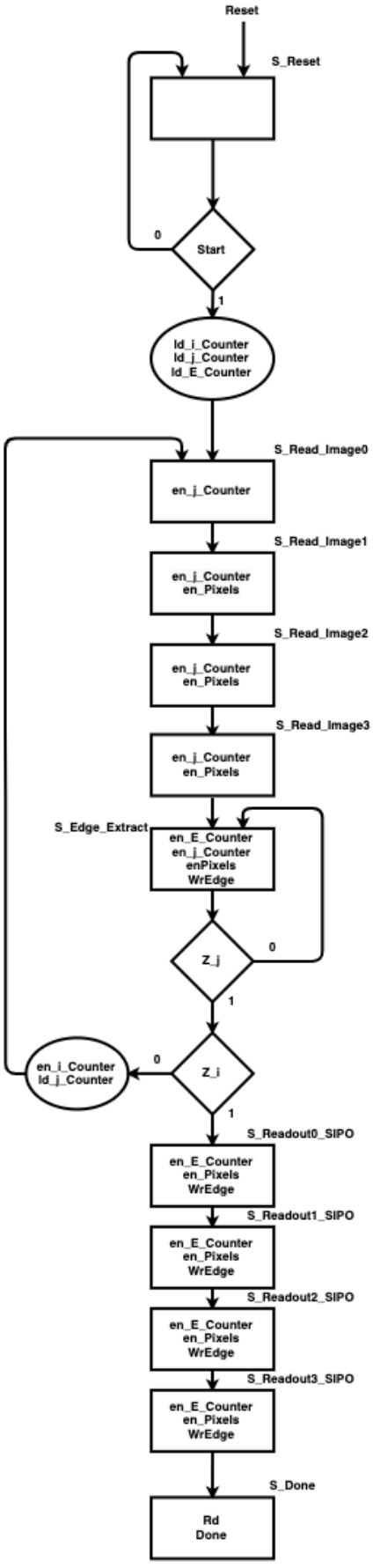
Top-level circuit interface:



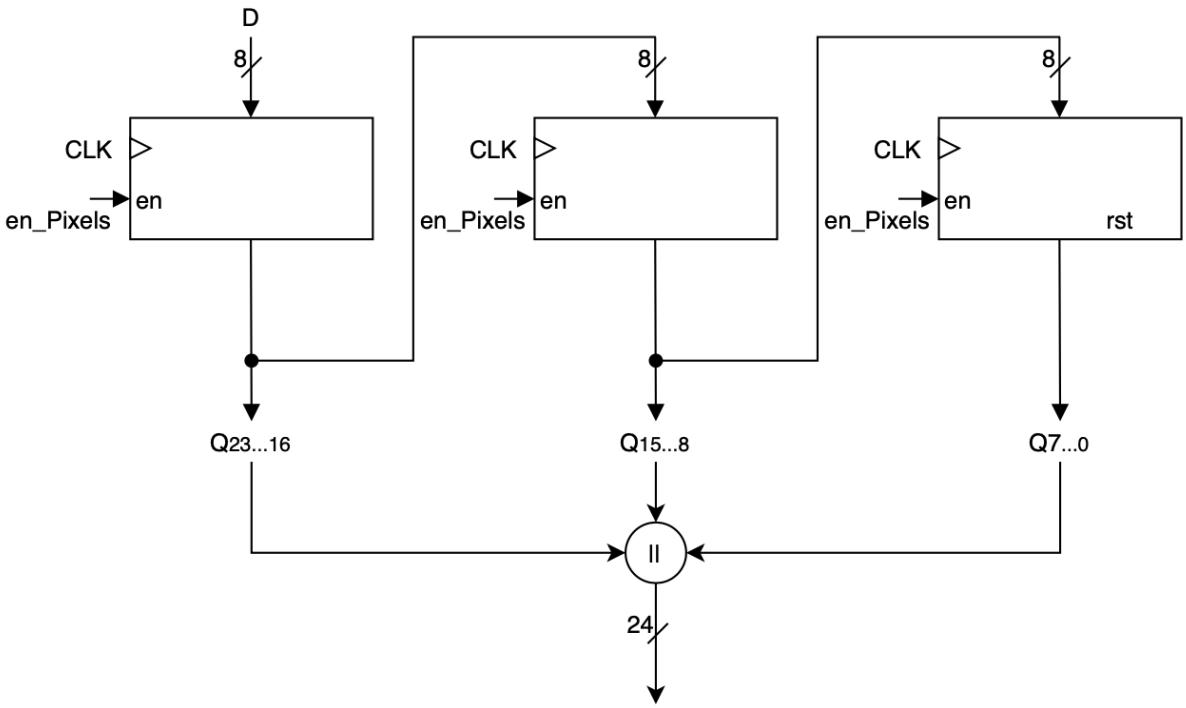
Block diagram of the Datapath:



ASM Chart:


Reset
S_Reset
0
Start
1
ld_i_Counter
ld_j_Counter
ld_E_Counter
S_Read_Image0
en_j_Counter
S_Read_Image1
en_j_Counter
en_Pixels
S_Read_Image2
en_j_Counter
en_Pixels
S_Read_Image3
en_j_Counter
en_Pixels
S_Edge_Extract
en_E_Counter
en_j_Counter
enPixels
WrEdge
Z_j
0
1
Z_i
0
en_i_Counter
ld_j_Counter
1
S_Readout0_SIPO
en_E_Counter
en_Pixels
WrEdge
S_Readout1_SIPO
en_E_Counter
en_Pixels
WrEdge
S_Readout2_SIPO
en_E_Counter
en_Pixels
WrEdge
S_Readout3_SIPO
en_E_Counter
en_Pixels
WrEdge
S_Done
Rd
Done

Block diagram of internal structure of the SIPO:



The input of the SIPO is an 8-bit vector. In each clock cycle, the output of the of previous register will be used as an input of the next register. The final output is the concatenation of three 8-bit vector which generates 24-bit output.

NOTE 2: You should use two instances of this SIPO in the edge detector architecture.

NOTE 3: Please pay attention to the order of the output bits. Each 8-bit chunk of data should be used in a correct way to be able to generate the correct output. (The orders are highlighted in BLUE in the datapath schematic)

NOTE 4: The width of S (the output of the subtractor) signal inside the datapath is 10 bits.

NOTE 5: You should initialize the ROM with the image showed on pages 3 – 5 of this specification. Please note that the image should be stored row wise.

Tasks:

Perform the following tasks:

1. Write a synthesizable VHDL code representing the circuit.
2. Write a testbench verifying the operation of your circuit, and the input image given in the algorithm description on pages 3 and 4.
3. Perform functional simulation of your circuit and use it to debug your VHDL code.
4. Synthesize your circuit.
5. Implement your circuit using
    a. FPGA family: Artix-7
    b. Part name: XC7A35TCPG236-1
    c. Speed Grade: -1
6. Based on the implementation reports, determine the number of CLB slices, LUTs, flip-flops, and pins used by the circuit.

Deliverables:

1. VHDL code of your entire circuit.
2. VHDL code of your testbench.
3. Timing waveforms from the functional simulation demonstrating outputs generated by your circuit.
4. FPGA resource utilization (as defined in Task 6 above).

If you are interested in learning more about edge detection algorithms, please take a look at Image Edge Detection.