LAB5
Sobel Edge Detector

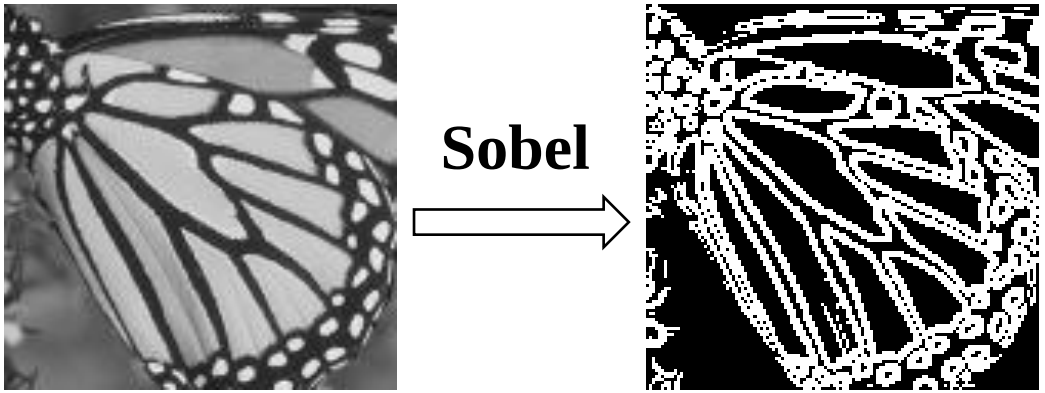
Outline

- Introduction
- Hardware Description
- Implementation
- Criteria
  - Grading Policy
  - Requirement & File Format
  - Command in Makefile

Introduction

- **Sobel operator** is an edge detection algorithm used in image processing and computer vision to find the edges of objects within an image.

- Three main steps:
  1. padding
  2. find gradient
  3. binarization



Hardware Description

Signal	I/O	length	Description
clk	I	1	Clock signal
rst	I	1	Active high synchronous positive edge triggered reset
pixel_in	I	8	Unsigned pixel to be processed
busy	O	1	Pixel_in stop inserting next pixel when busy is high
pixel_out	O	8	Unsigned pixel after process
valid	O	1	Test bench read pixel_out when valid pull high

Implementation

- Padding

We adopt “replicate padding” in this lab where the border of an image is extended by replicating the pixel values from the nearest edge or corner.

Pixel 0	…	Pixel 127
⋮	⋮	⋮
Pixel 16256	…	Pixel 16383

origin image
128*128

Pixel 0	Pixel 0	Pixel 1	…	Pixel 126	Pixel 127	Pixel 127
Pixel 0	Pixel 0	Pixel 1	…	Pixel 126	Pixel 127	Pixel 127
⋮	⋮	⋮		⋮	⋮	⋮
Pixel 16256	Pixel 16256	…			Pixel 16383	Pixel 16383
Pixel 16256	Pixel 16256	…			Pixel 16383	Pixel 16383

after padding
130*130

Implementation

- Find gradient

  Convolution with sobel operator Sx and Sy to get Gx and Gy

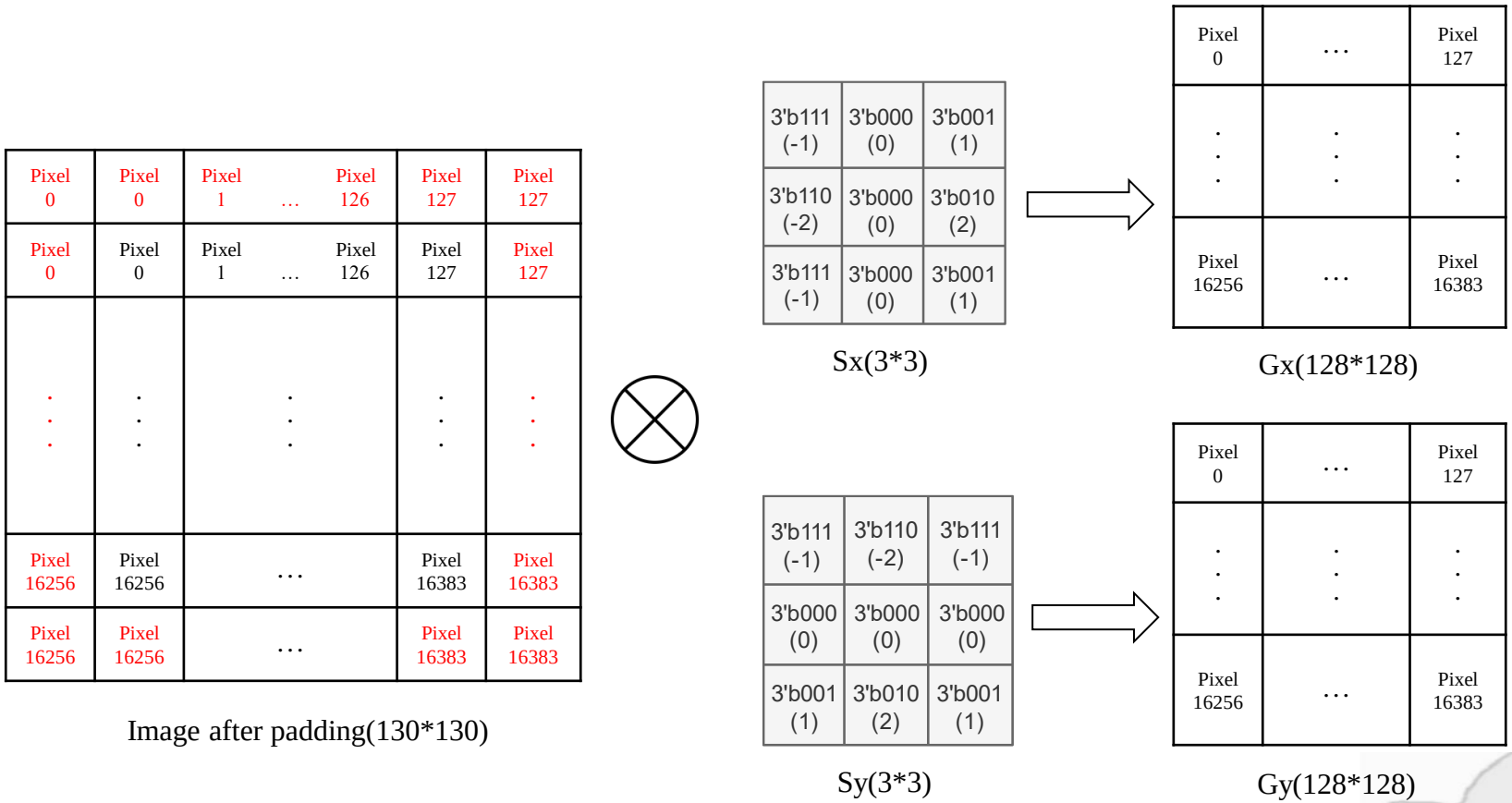


Image after padding(130*130)

Sx(3*3)

Gx(128*128)

Sy(3*3)

Gy(128*128)

Implementation

- Find gradient

  After getting Gx, Gy. We can finally obtain gradient(G) by

$$G = \sqrt{Gx^2 + Gy^2}$$

Pixel 0	…	Pixel 127
⋮	⋮	⋮
Pixel 16256	…	Pixel 16383

Gradient matrix
128*128

Implementation

- Find gradient

  Convolution implementation example:

$$1*(-1)+2*0+3*1+0*(-2)+1*0+2*2+3*(-1)+0*0+1*1=4$$

1	2	3	0	1
0	1	2	3	0
3	0	1	2	3
2	3	0	1	0
2	3	0	4	1

$\otimes$

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

=

4	4	-4
-4	4	2
-8	-1	3

Implementation

- Find gradient

  Convolution implementation example:

$$2*(-1)+3*0+0*1+1*(-2)+2*0+3*2+0*(-1)+1*0+2*1=4$$

1	2	3	0	1
0	1	2	3	0
3	0	1	2	3
2	3	0	1	0
2	3	0	4	1

$\otimes$

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

=

4	4	-4
-4	4	2
-8	-1	3

Implementation

- Find gradient

  Convolution implementation example:

$$1*(-1)+2*0+3*1+0*(-2)+1*0+0*2+0*(-1)+4*0+1*1=3$$

1	2	3	0	1
0	1	2	3	0
3	0	1	2	3
2	3	0	1	0
2	3	0	4	1

$\otimes$

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

=

4	4	-4
-4	4	2
-8	-1	3

Implementation

- Binarization

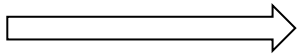
We iterate through each element in the gradient matrix. If G_i is greater than the threshold value, we set it to 255; otherwise, we set it to 0. (threshold value is set to $127_{decimal}$)

After binarization, it is already our output!

139	123	231
0	222	213
52	127	128

Gradient matrix

Binarization



255	0	255
0	255	255
0	0	255

Output feature map

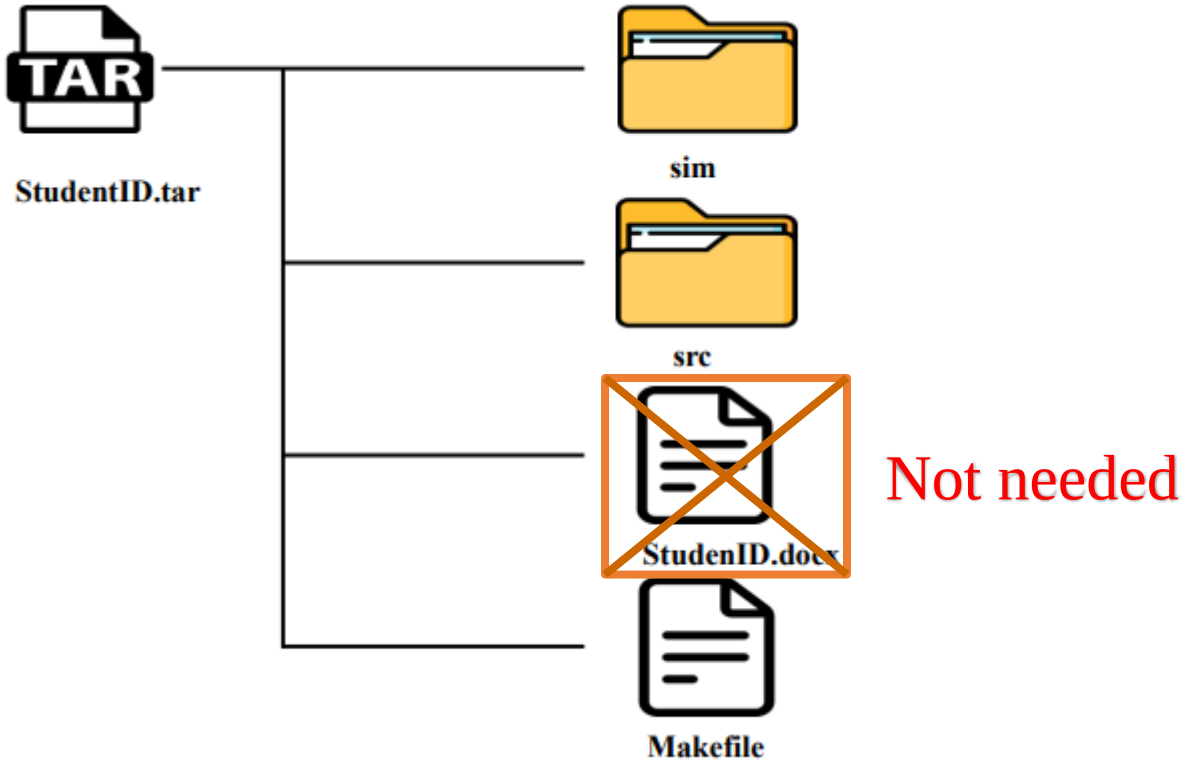
Criteria

- Grading Policy

One of the testbench fail	(0%)
All three testbench pass	(70 %)
All testbench pass and cycle count less than 20000	(100 %)

Criteria

- Requirement & File Format



Criteria

- Command in Makefile

Situation	Command
Run all pictures	make vcs_all
Picture1 simulation	make vcs1
Picture2 simulation	make vcs2
Picture3 simulation	make vcs3
Launch nWave	make wave
Delete waveform files	make clean
Compress homework to tar format	make tar