ECEN 468 Advanced Logic Design
Department of Electrical and Computer Engineering
Texas A&M University

(Lab exercise created by Jaeyeon Won and Jiang Hu)

Lab 11

Combining Canny Edge Detector with System Bus, Memory and UART

Purpose:

In this lab, we will combine Canny Edge Detector, SRAM and UART through System bus which were done in previous labs with Verilog, and will use Design Analyzer as a tool for synthesis as well as Lab6 to Lab9. Also, your module will be verified using functional simulation and gate simulation with the netlist synthesized.

Preparation:

1. Combining Canny Edge Detector

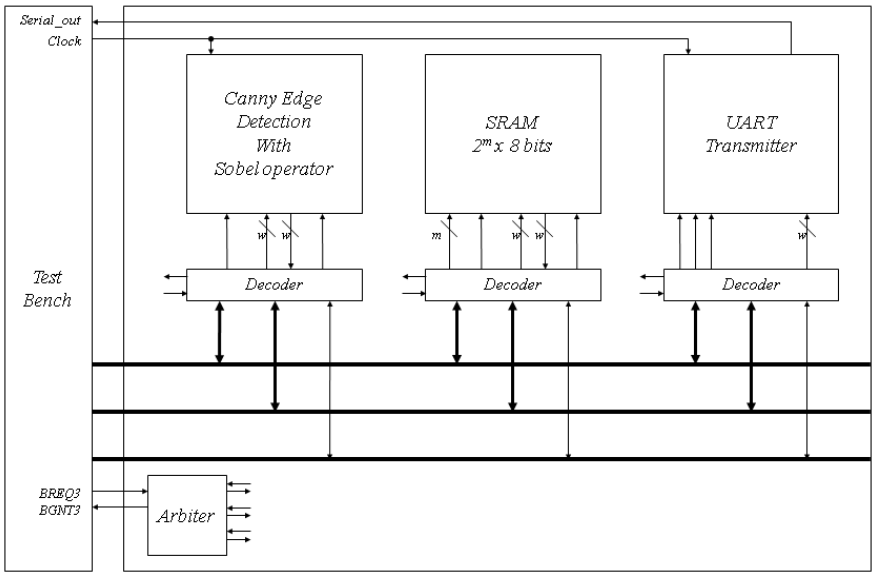

Figure 1 Entire system

We designed Canny edge detector in Lab9 with SystemC. In the previous lab, the image data moves between test-bench and the modules. This allows the top module can have only small size buffers to operate some algorithms to detect edge, so does not need to have large size memory. In this lab, we will use memory module which was done in lab1. Figure 1 shows the entire system we will design in this lab. The test-bench is related only to parcing circuit for bmp images in/out and control all of blocks. It can still have large size of memory to read and write bmp files.

Figure 2 shows signals interconnected between Canny Edge Detector module and its decoder.

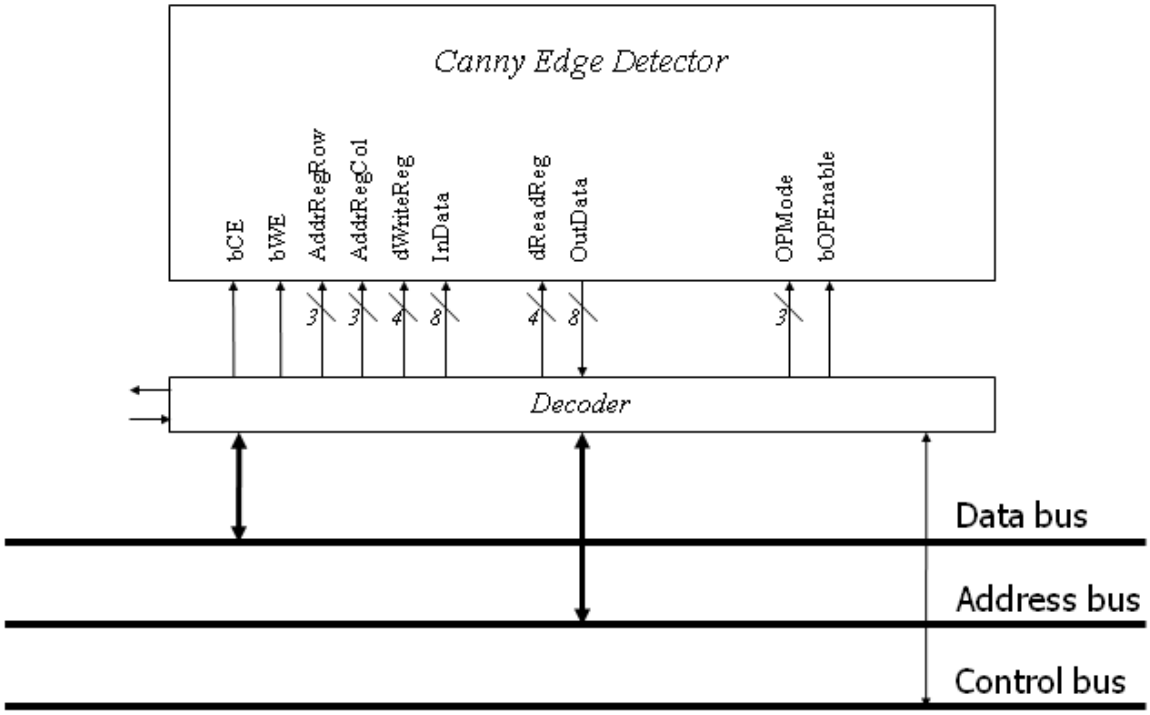

Figure 2 Interconnected signals for Canny Edge Detector

We will use a combined system which was done in Lab8. In the lab, we already combined memory module and UART transmitter through system bus. We will attach Canny Edge Detector which was done in Lab9 into the combined system which was done in Lab8. Also, you will modify the arbiter block when any modules are added into the system.

To attach the module with system bus, it also requires address decoder (wrapper) to decode control signals from the main controller such as test-bench or CPU. The 4 most significant bits are used for identification number of devices. We assume that the ID of Canny Edge Detector is **0100** which is a unique id for specific module. The remaining is used for control signals and address signals for control. Figure 3 shows the circuit for decoder and address map.

[31:28]	[27]	[26:24]	[23:20]	[19:16]
ID	bOPEnable	OPMode	dWriteReg	dReadReg

[15:8]	[7:5]	[4:2]	[1]	[0]
Reserved	AddrRegRow	AddrRegCol	bWE	bCE

Figure 3 Address map to control Canny Edge Detector

Figure 4 shows an example of the operation for noise reduction process. In the first step, the test-bench sends all of data to memory. And then, only data which are needs to be fetched is transmitted to the test-bench, and into canny edge detector module finally. After completion of the block processing in the detector module, the data will be read by test-bench and it will be sent to proper memory.

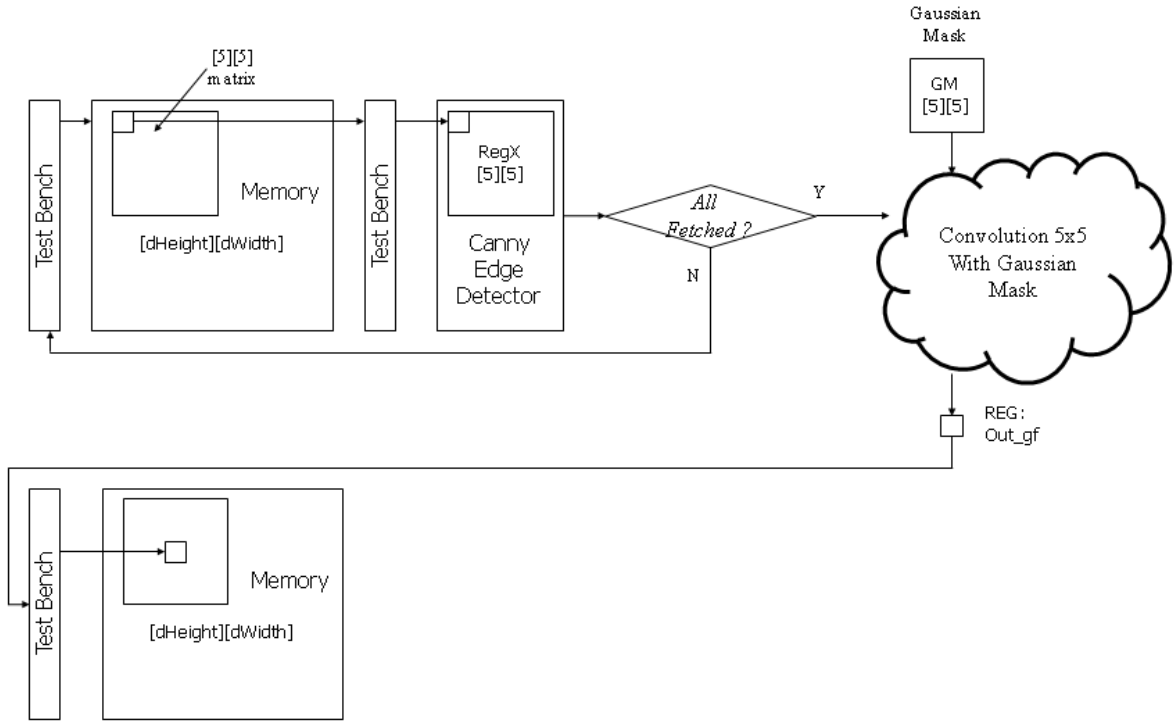

Figure 4 Data flow of Noise Reduction process

For the purpose of debugging, the system may need to send several serial messages through UART. In this lab, the messages should be sent per every phases such as noise reduction phase, gradient extraction phase, non-maximum suppression phase and hysteresis thresholding phase.

The messages below should be sent through UART.

Noise reduction phase : ‘N’ = 0x4E
Gradient extraction phase : ‘G’ = 0x47
Non-maximum suppression phase: ‘S’ = 0x53
Hysteresis thresholding phase: ‘H’ = 0x48

2. Reconfiguration of SRAM

We will expand the size of memory elements to support Canny Edge Detector. Figure 5 shows the memory maps to support it. While it operates with Canny edge detector, it needs some space to store intermediate data such as an image that noise is reduced, gradient information, direction information and an image after NMS process as well as original image and final image after hysteresis thresholding. Thus, you will assign memory addresses for each memory blocks. Figure 5 is one of the examples of memory allocation.

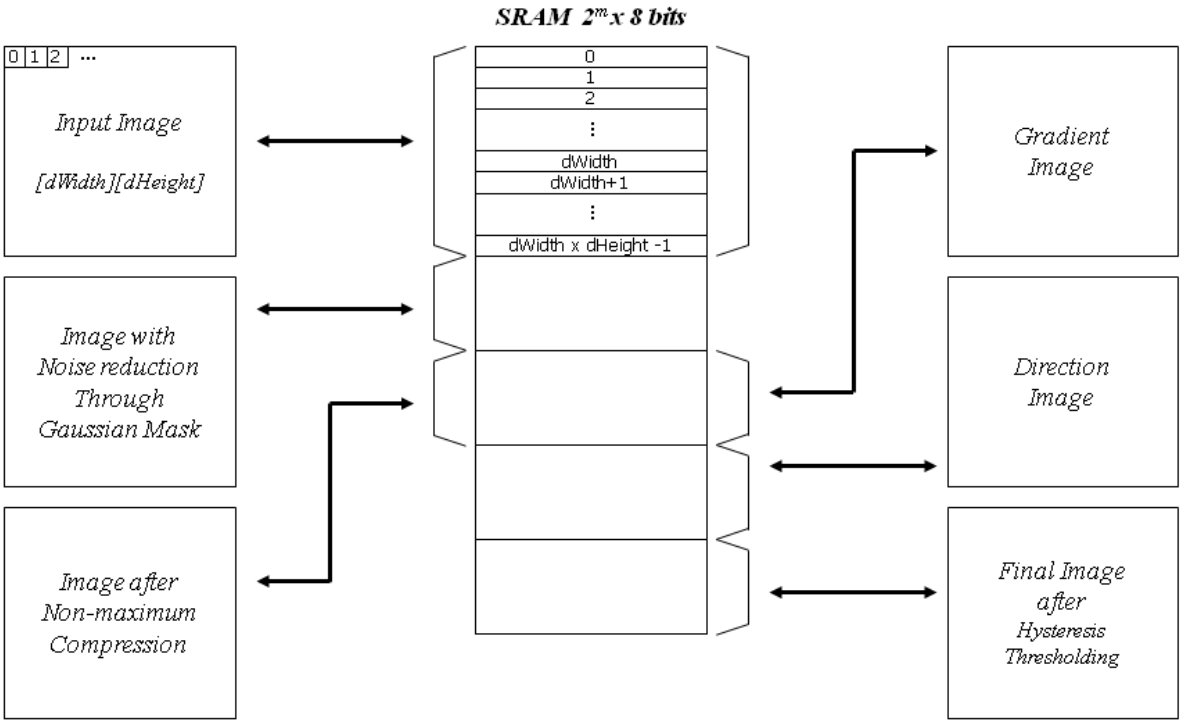

Figure 5 Memory allocation

The address and size of the memory depend on the size of image will be processed. You need to change the size of memory by estimating the maximum size of images. For example, if the maximum size of the image you want to simulate is 200x200 pixels, the size of memory will be used must be larger than 200x200x5 x 8bits and the maximum address of the memory 200x200x5. To support the range, the minimum number of pins required is 18 because the number 2^{18} is larger than 200x200x5.

3. Functional Simulation

Once the design is completed it must be tested. The functionality of the design module can be tested by applying a testbench and checking the results. The testbench module can instantiate the design module and directly drive the signals in the design module. The testbench can be complied along with the design module. At the end of compilation the simulation results will be displayed.

Download 'Lab11_code.tar.gz' file from the lecture website. The file contains WRAP_CANNY.v, Arbiter.v, systemtop.v, tb.v, generic.sdb, osu018_stdcells.v and osu018_stdcells.db. You will design modules in these files after you decompress the file.

a. Make working folder for this lab.

HOME]$ mkdir ECEN468/Lab11/SRC (make work folder)
HOME]$ cd ECEN468/Lab11/SRC (move to work folder)
Move the file 'Lab11_code.tar.gz' into the working folder
ECEN468/Lab11/SRC]$ tar xvfz Lab11_code.tar.gz (decompress the file)

Please check the files decompressed. It should contain the files below.
WRAP_CANNY.v, Arbiter.v, systemtop.v, tb.v,
generic.sdb, osu018_stdcells.v and osu018_stdcells.db

b. Also, copy the files below that you designed from Lab9 to this lab folder.
WRAP_SRAM.v, SRAM.v, virtSRAM.v, WRAP_UART.v, UART_XMTR.v
Control_Unit.v, Datapath_Unit.v

c. Also, copy CannyEdge.v that you designed from Lab10 to this lab folder. You should use 10 as the high threshold and 3 as the low threshold.

d. Insert your code into the test-bench(tb.v). In the test-bench given, there are the codes for parsing an input bitmap file and generating output files. You should insert your test-bench to verify memory operation, UART transmitter, gradient and direction, non-maximum suppression and hysteresis thresholding process referring Gaussian smoothing process.

f. Start Functional Simulation

ECEN468/Lab11/SRC]$ vcs tb.v
ECEN468/Lab11/SRC]$ vcs tb.v +v2k (if you use two-dimensional array)

If it shows compile errors, please check your codes again (go to step d) or make sure that your previous

modules are correct. If compilation is complete, go to next step to get the results.

ECEN468/Lab11/SRC]$ simv

If it is not same to your estimated results, please check your codes again (go to step d).

Please show your result to TA within your regular lab hour to get credit for lab evaluation.

4. Starting design analyzer

In this lab, we will synthesize whole modules except SRAM and generated its netlist to do gate-simulation. The 256K SRAM is used for this system, and too large to synthesize. In general, memory modules are replaced in a layout process with standard memory modules are provided by chip vendor. Thus, we will synthesize all modules except SRAM, and connect the module to the netlist after synthesis. You will see virtSRAM.v. The module should have exactly same in and out ports to your SRAM module. There is one difference in its behavior which does not have large size of internal memory and simple behavior. We will use this module instead of SRAM.v for synthesis.

Open systemtop.v file. And modify `include "SRAM.v" to `include "virtSRAM.v".

To start design analyzer, please follow steps. For your convenience, please start *design_analyzer* from separate directory with verilog files(*.v) because it generates a lot of intermediate files while it goes further.

Make directory for design analyzer. Now, you should be in ECEN468/Lab11/SRC.

ECEN468/Lab11/SRC]$ mkdir DV_WORK	(make work folder)
ECEN468/Lab11/SRC]$ cd DV_WORK	(move to the work folder)
ECEN468/Lab11/SRC/DV_WORK]$ design_vision &	(execute Design Analyzer)

5. Procedures of synthesis process:

Please refer to the previous manuals.

6. Gate Simulation

a) Copy Verilog netlist and sdf file to work folder.

ECEN468/Lab11/SRC/DV_WORK]$ cp systemtop_gate.v **../.**	(copy it to work folder)
ECEN468/Lab11/SRC/DV_WORK]$ cp systemtop.sdf **../.**	(copy it to work folder)

ECEN468/Lab11/SRC/DV_WORK]$ cd **..** (move to work folder)

b) Follow the steps below to make your testbench for gate simulation

Now we will make testbench for gate simulation. For gate simulation, we need systemtop_gate.v (netlist), systemtop.sdf (SDF file), osu018_stdcells.v and testbench.

We used virtSRAM.v instead of SRAM.v. Open Mainsystem_gate.v file, then you will see the SRAM module which has been synthesized with "virtSRAM.v". We will use original SRAM module in SRAM.v file for gate simulation. Delete the SRAM module in systemtop_gate.v because we will not use this SRAM module synthesized.

Please make sure those files should be in the current working folder. If not, please let TA know.

ECEN468/Lab11/SRC]$ cp tb.v gate_tb.v (copy new testbench)

Please check that the below lines are in gate_tb.v

On top of the file,

```
`timescale 1ns/10ps
`include "systemtop_gate.v"
`include "osu018_stdcells.v"
`include "SRAM.v"                // We will use original SRAM module.
```

On the bottom of the file, **(This line should be inside module)**

```
Initial
        $sdf_annotate("systemtop.sdf", systemtop_01);
```

Delete the line `include "systemtop.v" in your testbench

If you want to generate the waveform file, change the name of dump file to "wave_gate.dump" in your testbench. Please note that the size of the file will be huge, and be careful when you use the generation. You may delete the file after completion of this lab for your linux server capacity allowed.

c) Start gate simulation

ECEN468/Lab11/SRC]$ vcs gate_tb.v

If you see the error below, please make changes at module WRAP_SRAM, WRAP_UART and WRAP_CANNY in systemtop_gate.v file.

Error : The 32-bit expression "Addressbus" is connected to 20-bit port "AddressBus".

tri [19:0] AddressBus → tri [31:0] AddressBus;

Error : The 32-bit expression “Addressbus” is connected to 4-bit port “AddressBus”.
tri [31:28] AddressBus → tri [31:0] AddressBus;

Error : The 32-bit expression “Addressbus” is connected to 16-bit port “AddressBus”.
tri [31:16] AddressBus → tri [31:0] AddressBus;

If there is no syntax error, then do next step.
ECEN468/Lab11/SRC]$ simv

Please compare the result with the functional simulation result. If it is not same to your estimation, start simulation again from the functional simulation after you modify design modules.

Please show your result to TA within your regular lab hour to get credit for lab evaluation.

7. How to use the comparator given

To compare your results with the reference outputs given, you may use the another testbench file given. All names of output files(images) should not be changed.

ECEN468/Lab11/SRC]$ vcs tb_comp.v +v2k

Requirements:

1. General requirements.
   a. It should control SRAM, UART and Canny edge detector correctly
   b. Detailed comments.
   c. Please do not change the names of the signals given and the other functions not mentioned in the report. (You will lose some points if you change any names of the input or output signals.)
   d. You may modify any parts in the test-bench given to get the results only if it is resaonable. Please write detailed comments if you have modified.
   e. You score will be given based on your result, not your code.
   f. Late penalty : 20% of total score will be deducted on each subsequent weekday after due date.

2. Submit hardcopy of your report to TA. The report should include contents below
   a. The output images using functional simulation (input : kodim20_200.bmp) (8 points)
      (Please refer to the reference output results given and a comparator.)
   b. The output images using gate simulation (input : kodim20_200.bmp) (7 points)
      (Please refer to the reference output results given and a comparator.)
   c. Analysis on how many bytes to read from and write to Canny for each operation. (7 points)
   d. Analysis on how many bytes to read from and write to SRAM for each operation. (7 points)
   e. Only the parts modified in tb.v and gate_tb.v, if you have modified.(not full source) (2 points)
   f. Calculate the maximum clock speed of the test-bench you used if you have changed clock period of the testbench. Otherwise, calculate the frequency given in the testbench. (2 points)
      (You will get scores based on the maximum clock speed.)
   g. Show your final results to TA. (5 points)

Note : Please send **<u>ONLY source codes and test-bench(excluding results)</u>** to TA's email address. (2 points)

All source codes(*.v), tb.v, gate_tb.v, systemtop_gate.v, systemtop.sdf

---