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 5

Combining Canny Edge Detector with System Bus, SRAM and UART

Purpose:

In this lab, we will combine Canny Edge Detector, SRAM and UART through System bus which are done in previous labs with SystemC, and will use Vista as a tool for simulation and verification.

Preparation:

1. Combining Canny Edge Detector

We have designed Canny edge detector in Lab4 with SystemC. In the previous lab, the image data moves between test-bench and the modules, so large internal memory has been declared in the test-bench. This has allowed the modules can have only small size buffers to operate some algorithms to

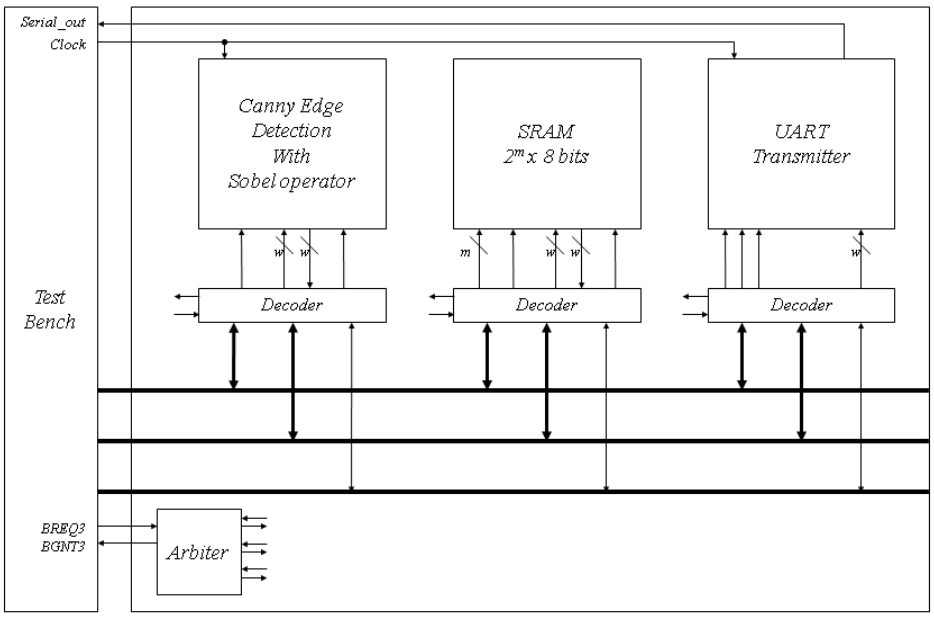

Figure 1 Structure of the entire system

detect edge, so does not need to have large size memory in our modules. 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 bmp images in/out and control all blocks. It can still have internal memory to read and write bmp files.

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

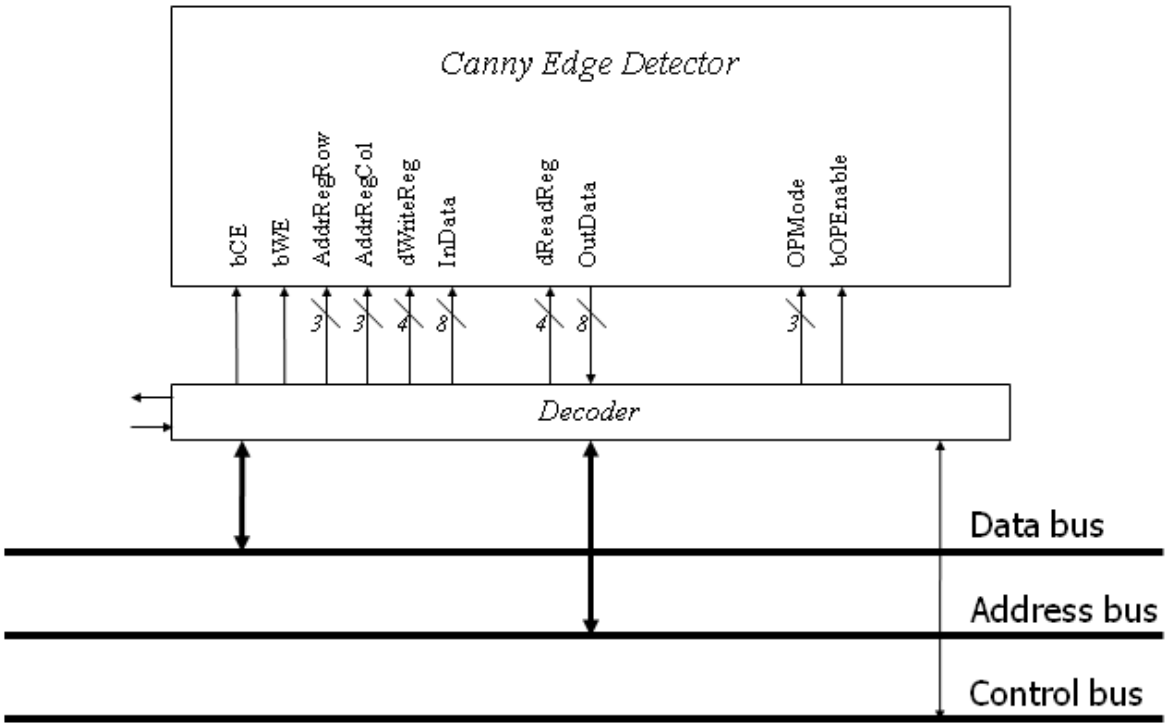

Figure 2 Interconnected signals for Canny Edge Detector

We will use a combined system which was done in Lab3. In the lab, we already combined memory module and UART transmitter through system bus. We will attach Canny Edge Detector which was done in Lab4 into the combined system which was done in Lab3. Also, the arbiter block which has been modified for additional modules connected will be given.

To attach the module to the 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 address map.

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

[15:8] Reserved	[7:5] AddrRegRow	[4:2] AddrRegCol	[1] bWE	[0] 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 needed to be fetched is transmitted from the memory 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 back.

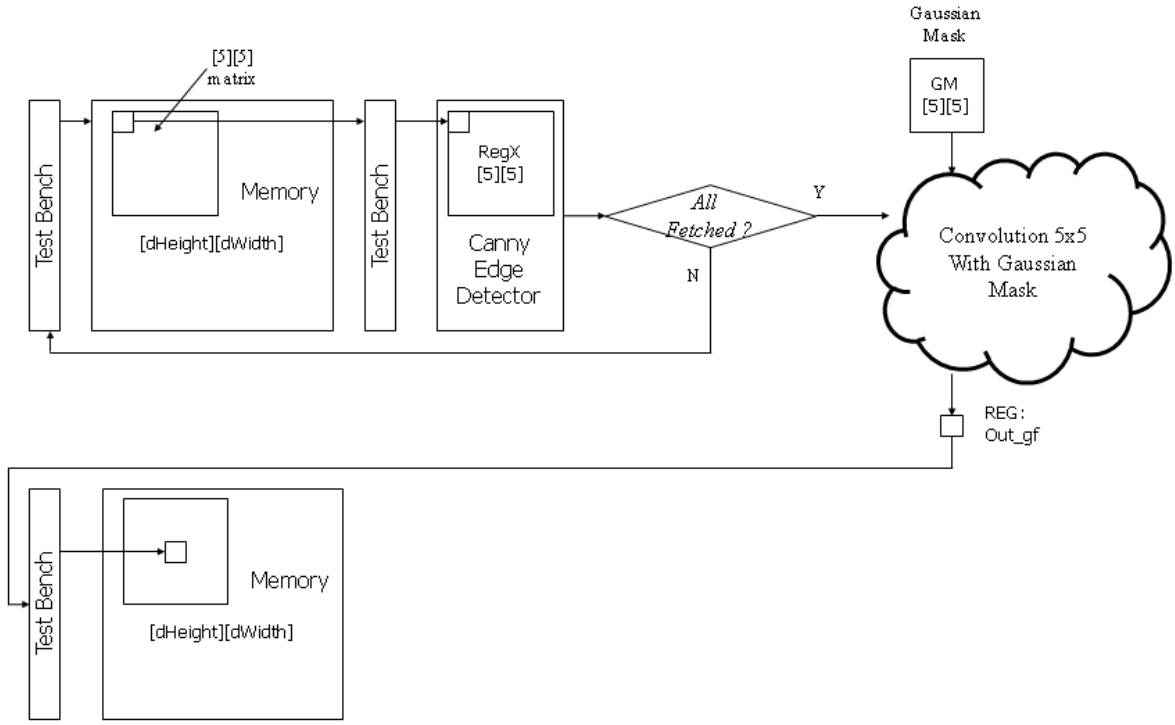

Figure 4 Data flow of Noise Reduction process

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.

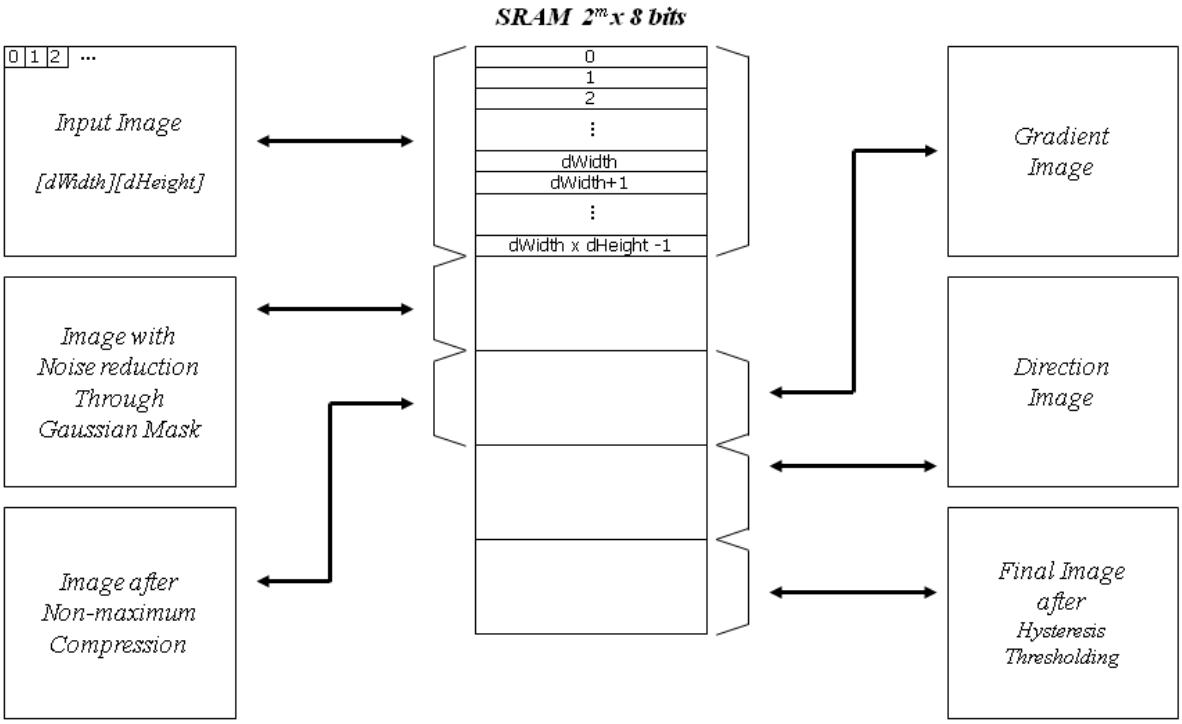

Figure 5 Memory allocation

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 ‘Lab5_code.tar.gz’ file from the lecture website. You will design modules in these files after you decompress the file.

a. Make working folder for this lab.

HOME]$ mkdir ECEN468/Lab5/SRC (make work folder)
HOME]$ cd ECEN468/Lab5/SRC (move to work folder)
Move the file ‘Lab5_code.tar.gz’ into the working folder
ECEN468/Lab5/SRC]$ tar xvfz Lab5_code.tar.gz (decompress the file)

Please check the files decompressed. It should contain the files below.
Canny_Edge_WRAP.cpp, Canny_Edge_WRAP.h, Arbiter.cpp, Arbiter.h, test.h, test.cpp and main.cpp

b. Also, copy the files below that you designed from Lab3 to this lab folder.

SRAM_WRAP.cpp, SRAM_WRAP.h, SRAM.cpp
UART_XMTR.cpp, UART_XMTR.h, UART_XMTR_WRAP.cpp, UART_XMTR_WRAP.h

c. Also, copy Canny_Edge.cpp, Canny_Edge.h that you designed from Lab4 to this lab folder.

d. Insert your code into the module files. Your module should satisfy the requirements. In the test-bench given, there are codes for Gaussian filter process and how to get gradient and direction information. You will be able to verify your Canny_Edge_WRAP.cpp file with this test-bench file. Please show your result to TA within your regular lab hour to get credit for lab evaluation.

e. Insert your code into the test-bench(test.cpp). In the test-bench given, there are reference codes for Gaussian filter process (Do_3x3_Gaussian()) and how to get gradient and direction information(Do_3x3_Sobel()). You should add your codes for non-maximum suppression (Do_3x3_NMS()). Please show your result to TA within your regular lab hour to get credit for lab evaluation.

f. Insert your code into the test-bench (test.cpp) for hysteresis thresholding process (Do_3x3_Hysteresis()). Please show your result to TA within your regular lab hour to get credit for lab evaluation.

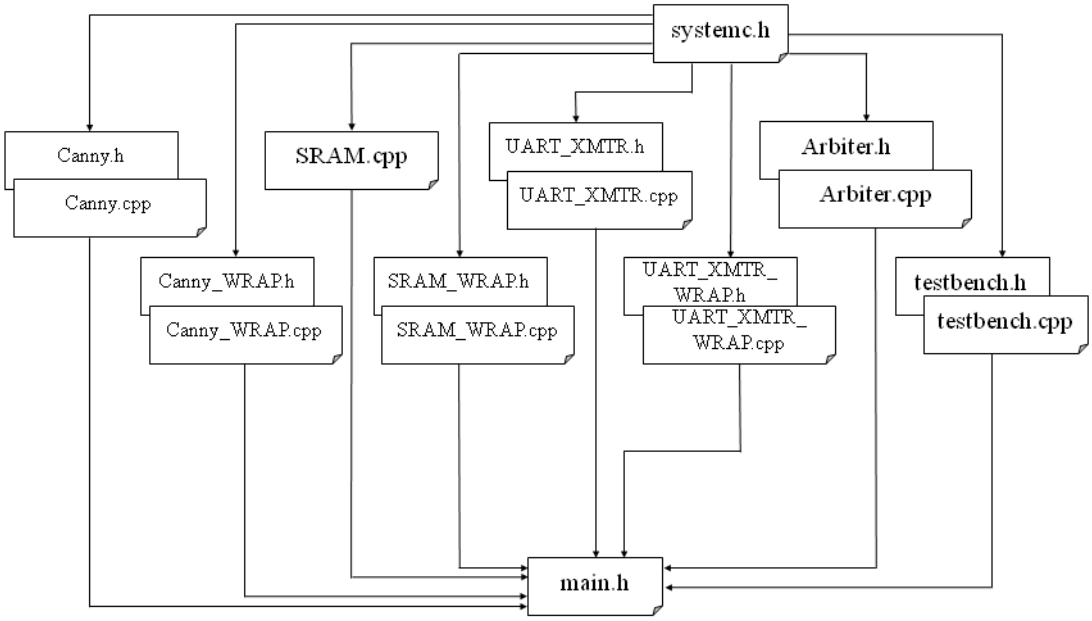

Figure 6 Hierarchy of files used in this lab

4. Tips for simulation

a. How to reduce simulation time
   - It will take a few minutes to simulate 200x200 size of bmp file. Instead of simulating large size bmp file, you can simulate with small size bmp file in your first step.
   - If you want to use small size bmp file (kodim23_50.bmp given), please change the input file name in test.cpp. (Also, you should change the variable 'DummyData' to 2.)

   200x200 size bmp file : DummyData should be 0.
   50x50 size bmp file: DummyData should be 2.

Report Requirements:

1. Your design modules should satisfy constraints below.
   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 score will be given based on your result, not your code.
   e. 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. Canny_Edge_WRAP.cpp with detailed comments. (5 points)
   b. The function Do_3x3_NMS and Do_3x3_Hysteresis with detailed comments. (10 points)
   c. output (*.bmp files) (input image : cman_200.bmp) (15 points)
      (Please refer to the reference output results given and matching ratio at the end of the simulation.)
   d. An additional simulation (5 points)
      - Please use different input image (kodim21_200.bmp given) in test.cpp
      - You should use these thresholds in Canny_Edge.cpp. (high : 20, low :5)
      - Add the output images in your reports.
        (Please refer to the reference output results given and matching ratio at the end of the simulation.)

Note : Please send **ONLY source codes and test-bench(excluding results)** to TA's email address . (5 points)