Carleton University

Department of Systems and Computer Engineering

SYSC 3006 Fall 2016

Computer Organization

Lab #1

The operation of a processor is controlled by a finite state machine (FSM). This lab will introduce how to implement an FSM using a ROM (Read Only Memory). The ROM-based implementation is different than the combinational logic approach used in the design process presented in ELEC 2607.

In this lab you will:

- Design a ROM-based FSM.
- Implement the circuit using Logisim.
- Log the simulation of the circuit behaviour

In addition to lab-specific details, this description identifies the following types of information spread across the entire document:

[Background]: technical material relevant to the lab

[In the Lab]: activities to perform in the scheduled lab session

[Checkpoint]: questions/tasks to check your understanding; TA might evaluate you on them during the lab as well.

Suggestion: Read this document completely and carefully before deciding what to do.

Please bring questions about the lab to class for discussion at the beginning of class.

[Background] Read the Logisim Guide for SYSC 3006 and complete the contained tutorial exercises as PreLab preparation. Note: Logisim has a learning curve that requires the tool to be used a bit before trying to do anything useful (like the lab) … there may not be enough time to complete the lab in the scheduled session without some initial exposure to Logisim prior to the scheduled session.

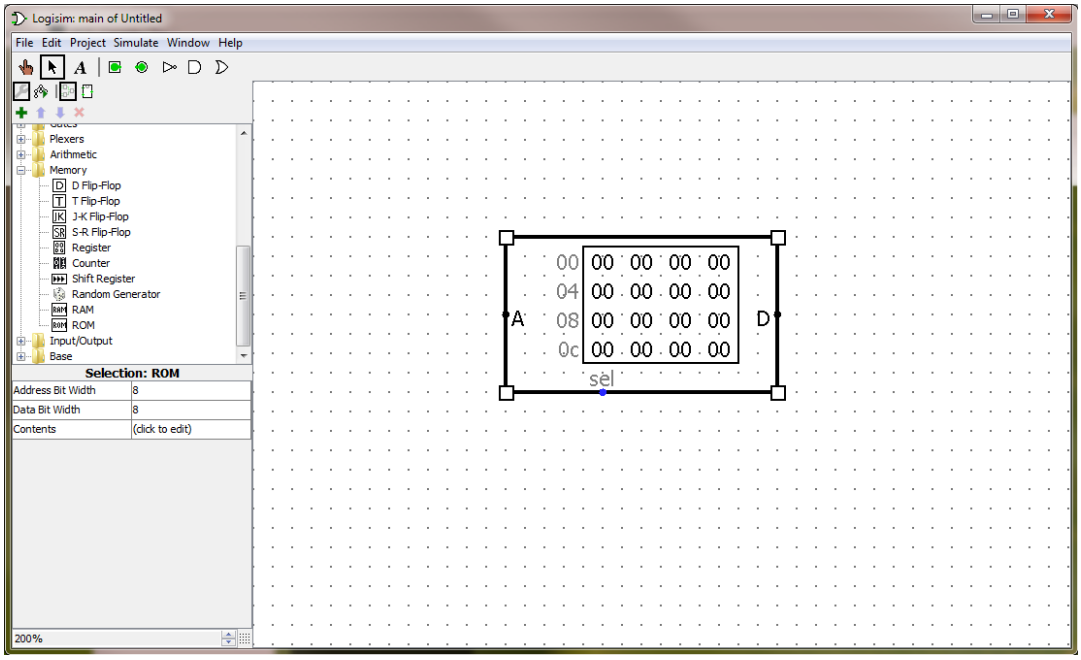
Read Only Memory (ROM)

[Background] See the lecture slides for an introduction to memory components.

An n-bit memory component is a bank of n-bit *words*. Each word holds an n-bit value (its *contents* or *data*). Each word has a unique *address* that identifies the word. The contents of a ROM are static (fixed) and determined when the ROM is manufactured. A ROM supports a Read operation but does not support a Write operation.

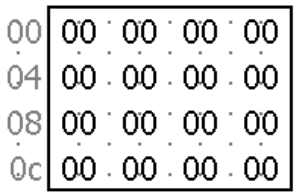
Logisim has a built-in ROM component. The ROM's attributes allow the specification of the ROM's Address Bit Width (determines the number of words in the bank of memory) and the ROM's Data Bit Width (determines the number of bits in each data word). To Read the value of a specific word in a ROM, the address of that word must be presented at the Address inputs, and the contents of the word will be output at the Data outputs.

When A ROM component is placed on the Logisim canvas, it looks like this:



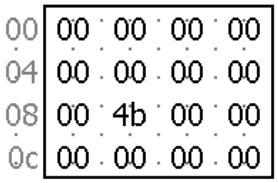
The default Address Bit Width attribute is 8, and therefore the ROM contains 2^8 words (i.e. 256 words). The default Data Bit Width attribute is 8, and therefore each word contains 8 bits. The "A" pin on the component represents the 8 Address inputs (the A "pin" is actually a bundle of 8 pins). The "D" pin on the component represents the 8 Data outputs (the D "pin" is also a bundle of 8 pins). The attributes of the ROM can be changed to change the number of words in the ROM (change the Address Bit Width) or the size of each word (change the Data Bit Width).

The values stored in the ROM are also displayed in the rows:

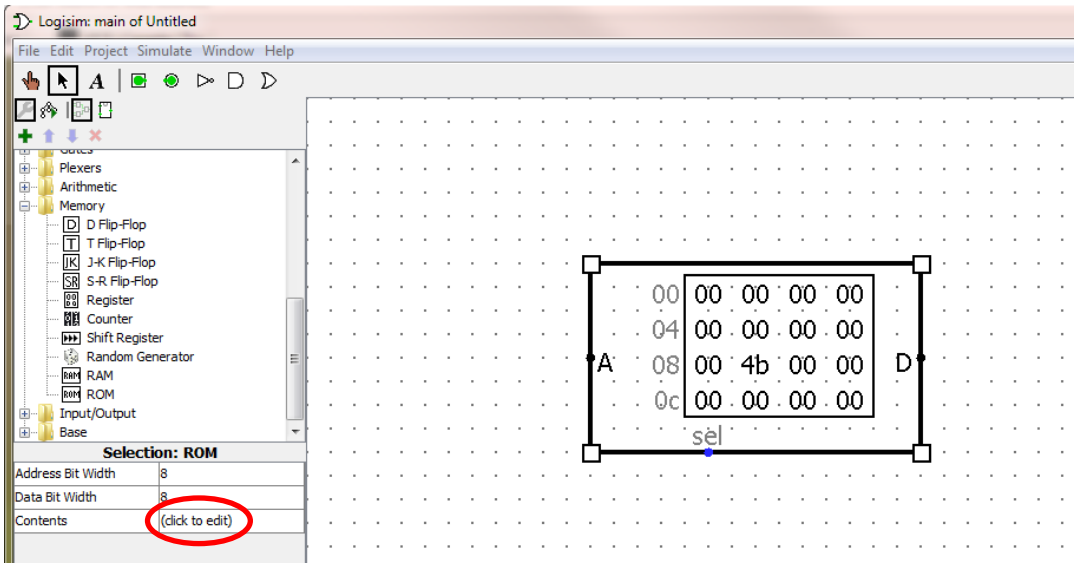


The lighter grey numbers down the left side of the box represent addresses. The addresses are displayed as hexadecimal values (00, 04, 08, 0c). The address to the left of a row represents the address of the first word in the row. Since each row contains 4 words in this case, the start addresses of successive rows go up by 4. The box shows only 16 of the 256 words. The values of the words corresponding to the addresses are shown in hexadecimal in each row. Note that each word is initialized to 0 by default.

The values of the words can be changed using the Poke tool. Poke the word to be changed and enter the hexadecimal value for the word. Here, the word at address 09 has been modified to contain the value 4b:



The values stored in the ROM can also be edited by selecting the ROM (so its attributes are shown) and then clicking on the "(click to edit)" field of the Contents attribute (try it and see!):



ROM-Based FSM

[Background] See lecture slides for a general discussion of FSMs.

The behaviour of an FSM is typically specified in a State Table. The Lab1 State table specifies the Lab1 FSM behaviour:

Current State	Outputs (binary) O2 O1 O0	Next State
0	0 0 0	4
1	0 1 0	5
2	0 1 0	3
3	0 0 1	0
4	0 0 1	1
5	1 0 0	2

Table 1 Lab1 State Table

In a ROM-based implementation of an FSM, each word of the ROM implements one line of the State Table. The address of a word corresponds to the line of the State Table that it implements (e.g. for Table 1, the word at address 2 would correspond to State 2). Each word of ROM data contains the outputs and the Next State information for the line it implements (e.g. for Table 1, the binary value stored in the word at address 2 would encode the outputs 010 and the next state 3).

An FSM must have a memory element (in 2607 it was a set of flip flops) that remembers the current state during operation. In a ROM-based implementation of an FSM, the current state value stored in the memory element is used as a ROM address to select the word containing the encoding of the outputs and next state associated with the current state.

[Checkpoint] What is the minimum ROM Address Bit Width needed to implement Table 1, and why? What is the minimum ROM Data Bit Width needed to implement Table 1, and why?

[In the Lab] Implement Table 1 as a ROM-based FSM:

Use a Logisim ROM to implement the State Table. Configure the ROM for the minimum Address and Data Bit Widths that can be used for this lab (as per your answers to the questions above). Program the contents of the ROM appropriately.

Use a Logisim Register component (label it “Current State”) as the memory element to hold the Current State value during operation. Configure the Current State Register appropriately, and connect its outputs as the ROM address inputs.

Use the output from the Toggle Switch design presented in the Logisim Guide for SYSC 3006 as the clock input to the Current State Register, and configure the Register to be falling edge sensitive.

Direct the ROM Data outputs through a splitter. Configure the splitter to have the minimum possible number of wires to successfully implement the lab. Split the wires containing the Next State encoding into a bundle and feed this bundle back to the Current State Register data inputs. Split the remaining wires out individually.

Connect each individual wire output from the splitter to an individual LED. Label the LEDs O2 O1 and O0 so they correspond with the Lab1 State Table.

Log the Output LED values (see Simulate → Logging to set up Logisim to log data during simulation).

Toggle the Toggle Switch enough times to demonstrate the correct implementation of the FSM.

Record the simulation log as proof that the system works.

Throughout the implementation, configure all components to have the minimum possible capacities and signals needed to implement the FSM.

Show the TA your working system and the simulation log before the end of your scheduled lab session.

Lab 1 Grading Scheme: 10 marks total.

Note about making pdf files (for PreLabs in future labs): The SYSC lab has PDFCreator installed. PDFCreator is a free program that can also be downloaded for use at home from http://sourceforge.net/projects/pdfcreator/ . When installed, it appears as a printer. Printing a file to this printer will generate a pdf file as output. (Prof. Pearce uses PDFCreator to create all of the pdf files for his courses.)

In this course, you might find it useful to build documents (like PreLabs and Reports) as a Word document and then print the complete document using PDFCreator.