CPSC 359 – Fall 2019
Assignment 2: A Sequential Circuit for Slow Integer Division

due: 0900h 21-Oct-2019

Background

In this assignment, you will *program* a sequential circuit to implement and algorithm for integer division. While the circuit is very specific to this task, it is a simplification of many of the elements of a CPU. Your *program* will be equivalent to the *microcode* in a CPU that establishes the machine-code/assembly language for the device.

Objective

Design and implement a sequential logic circuit that divids two unsigned 32-bit integers.

Details

Algorithm 1 shows how to divide two unsigned integers using binary arithmetic. I have provided you with a *Logisim* template to implement this algorithm (Figure 1).

Algorithm 1 Algorithm to divide two unsigned integers (the slow way).

Require: unsigned operands a and b
Ensure: $x = a/b$, $a = a \bmod b$
1: $x \leftarrow 0$
2: **while** $a >= b$ **do**
3: $\quad x \leftarrow x + 1$
4: $\quad a \leftarrow a - b$
5: **end while**

There are three 32-bit registers in the template. These correspond to the values of a, b and x in Algorithm 1. Two of the registers have 2-into-1 multiplexers connected to the D input. When setting one of these values, the state of the multiplexer selects the source of the new value loaded into the register on a low-to-high clock transition. Table 1 summarizes the loaded values for the three registers. The template has additional components to compute values for $a < b$, $x + 1$, and $a - b$.

The template has a start button to initiate the algorithm. Alternatively, if you are stepping through the states to debug your circuit, there is an input pin that you can use to statically set the value of the start signal (make stepping a bit easier).

register	sel = 0	sel = 1
a	$a \leftarrow$ first operand	$a \leftarrow a - b$
b	$b \leftarrow$ second operand	
x	$x \leftarrow 0$	$x \leftarrow x + 1$

Table 1: Source values for register loads for given multiplexer select (sel) values.

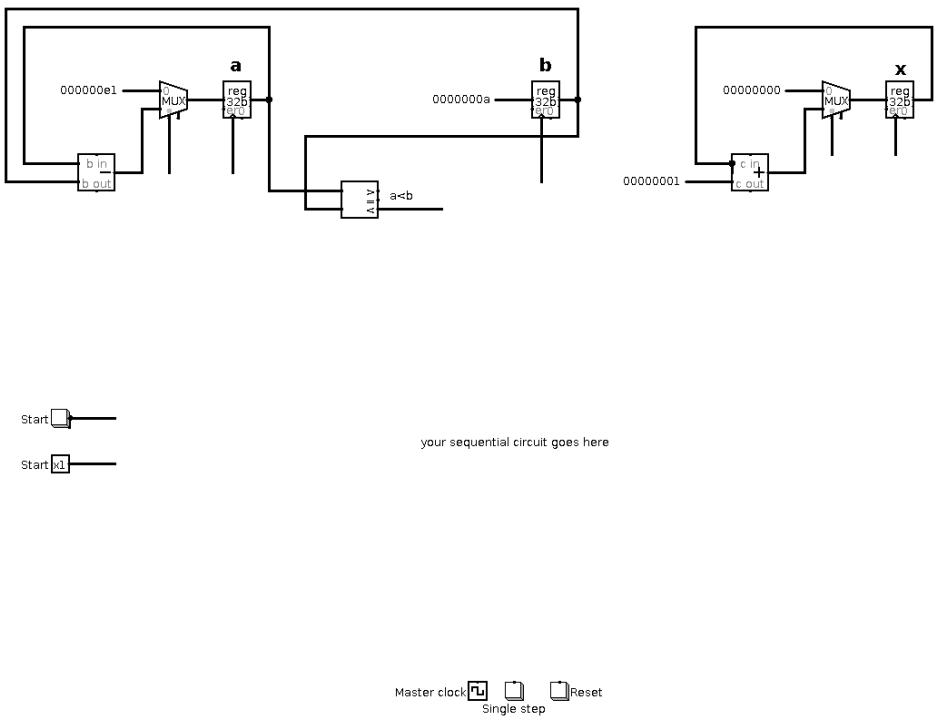

Figure 1: Template circuit for multiplying unsigned integers.

This circuit is sufficient to execute Algorithm 1. You should take some time to play with the template to do an iteration or two of the loop in the division algorithm. Once you are comfortable that you understand how this circuit can divide numbers, create a state diagram for Algorithm 1.

Finally, implement the your state diagram in a sequential circuit using a read-only memory (ROM), as discussed in class. Your sequential controller will need a clock signal to drive it. If you then click *Ticks Enabled* under the *Simulate* menu in Logisim, the circuit should quickly compute the quotient of the two 32-bit constants provided for a and b. You can step through the states one clock pulse at a time with the *ctrl-T* key, or by using a button to the sequential circuit's clock input.

What you need to do

To design the circuit and complete the assignment, you need to do the following.

1. Identify all the inputs and outputs for your sequential circuit.
2. Design a finite state machine that will provide the required signals to the circuit in the template.
3. Design the combinational logic (or ROM programming) to implement your finite state machine.
4. Implement your design in Logisim.

Deliverables

Turn in the following items for evaluation.

1. All written work to show the steps in your analysis and design for items 1 through 3 above.
2. All files for your Logisim implementation of your design.

Note, while you may use discrete logic gates to implement the combinational logic required, you will find it easier to use a read-only memory (ROM) as described in class.

Hints

1. Make sure you have an idle state. That way your machine can sit in constant state while you set inputs or read outputs.
2. A *start* signal can be used to initiate the computation.
3. The registers load when their clock signal goes high. It is important that the multiplex select bit be correct and stable when that happens. One way to do this is to set the select bit in the previous state, and maintain its value in the state that brings the *load* signal high.
4. I wrote a short Python program to help me with the ROM programming. It can save you a lot of time.

Evaluation

1	Identification of input/output/state variables.	2pts
2	Finite state machine - does task correctly.	6pts
3	Design of combinational logic/ROM programming.	5pts
4	Logisim implementation - runs correctly.	5pts
5	Logisim implementation - inputs/outputs labelled.	2pts
	Total	20pts

There is no team work. Each student should submit their own solution.

Late work

After the deadline and up to 24hrs late: -5pts. After 24hrs and up to 48hrs late: -10pts. Over 48hrs late: -20pts, i.e., no assignment will be accepted beyond 48hrs after the deadline.

Plagiarism

Work must me the sole work of the individual submitting the work.

You may find that the design task for this assignment is similar to other well-known circuits. Nevertheless, you should go through the design process yourself, and submit your own work. If you do borrow from other sources, cite the source and clearly indicate what you have borrowed, keeping in mind the design must be substantially your own. If you cite your sources, worst case you may receive a reduced grade for borrowing too much. If you borrow, but do not cite, that is plagiarism and academic misconduct, and carries severe penalties as determined by the Faculty of Science.

As a guideline, consider the 20-minute rule. Talk with your colleagues and consult other sources (cite them please). Wait at least 20 minutes, then do your work to be sure that it is your own. Less then 20 minutes usually means that you are merely copying work from the original source.