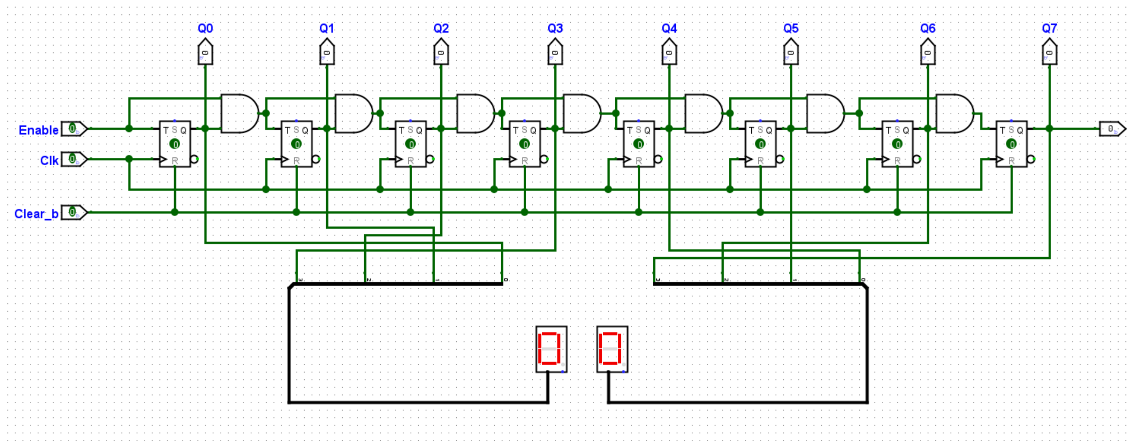
CSC258 Lab5 Report

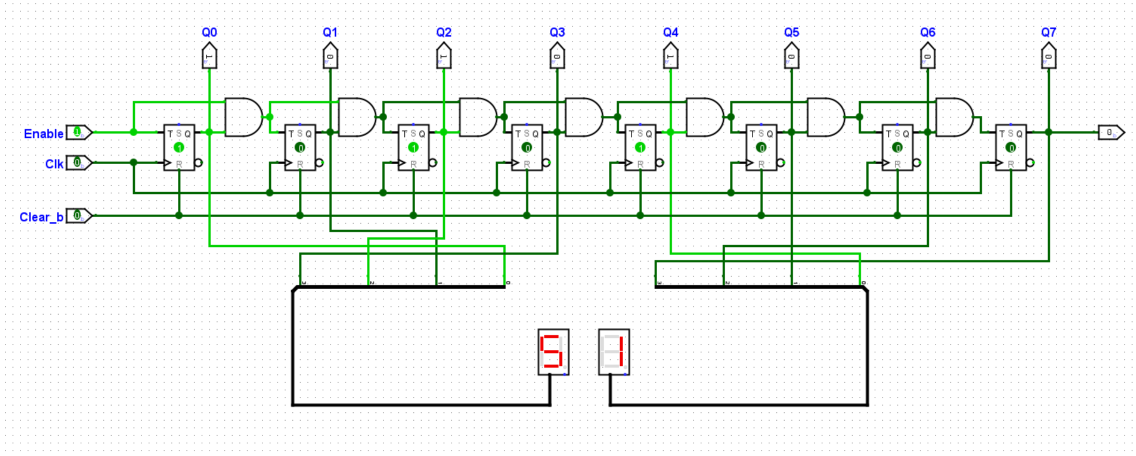
Tianle Wang 1006337028

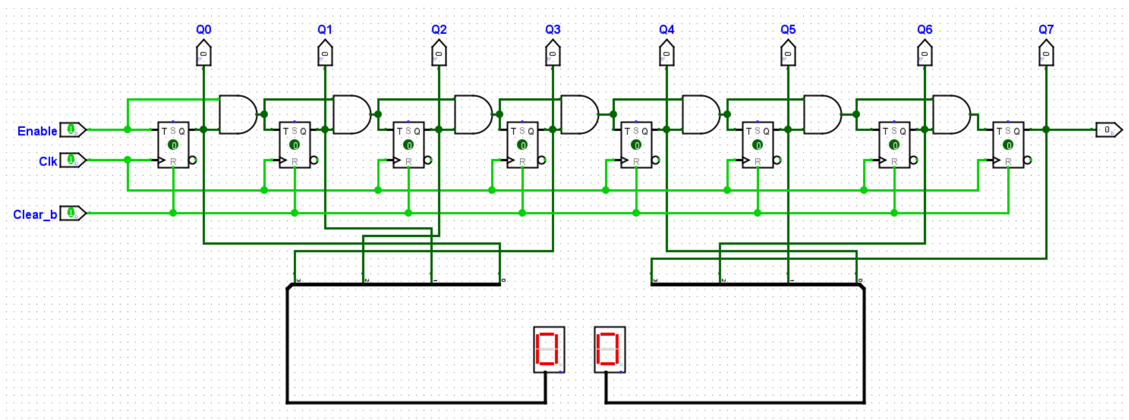
Part I

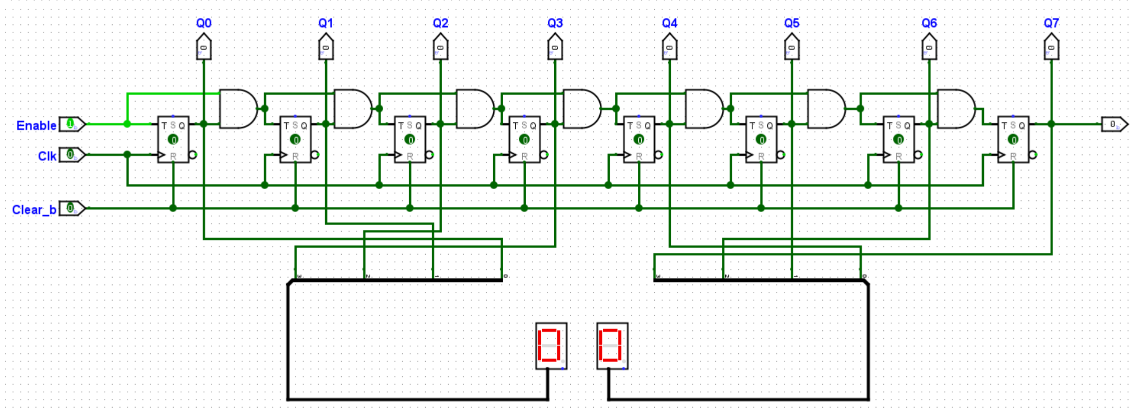
1. Draw the schematic for an 8-bit counter using the same structure as shown in Figure
2. Annotate all Q outputs of your schematic with the bit of the counter ($Q_7Q_6Q_5Q_4Q_3Q_2Q_1Q_0$) that they correspond to (where Q_7 is the most significant bit and Q_0 is the least significant bit. Label the inputs according to the diagram in Figure 1.
3. Build the circuit corresponding to your schematic. Your circuit should use the flip-flop module provided by Logisim, instantiated eight times to create the counter.
4. Connect the Q output bits to two seven-segment displays so you can monitor the output values.
   1-4 showing below:



5. Test your modules with *Poke* to verify its correctness. You will need to reset (clear) all your flip-flops early in your simulation, to ensure that your circuit starts in a known state. Include screenshots of simulation output in your prelab.







Part II A

1. The check for the maximum value is not necessary in the example above. Explain why in your prelab report.
   - Since we use `AND` gate to collection the 4 digits, if it reach the maximum `1111`, then the `AND` gate will return 1, then the `1` will turn on the the signals of *M1* and *M2* that load a new multi-bit value provided. In the figure given, the provided value is `0` so that when the counter device reach the maximum, it will be set into `0` in next clk cycle.
2. If you wanted this 4-bit counter to count from 0-9, how would you adjust the circuit above.

- I will change the `AND` gate to connect only the last digit and the first digit so that when `1001` then it will be set into `0` in next clk cycle

3. In *Properties* there is a setting called *Action On Overflow*. Explain how each value for this setting responds to overflow by experimenting with this setting and describing the results.

If the `AND` gate does not connect so that the action worked

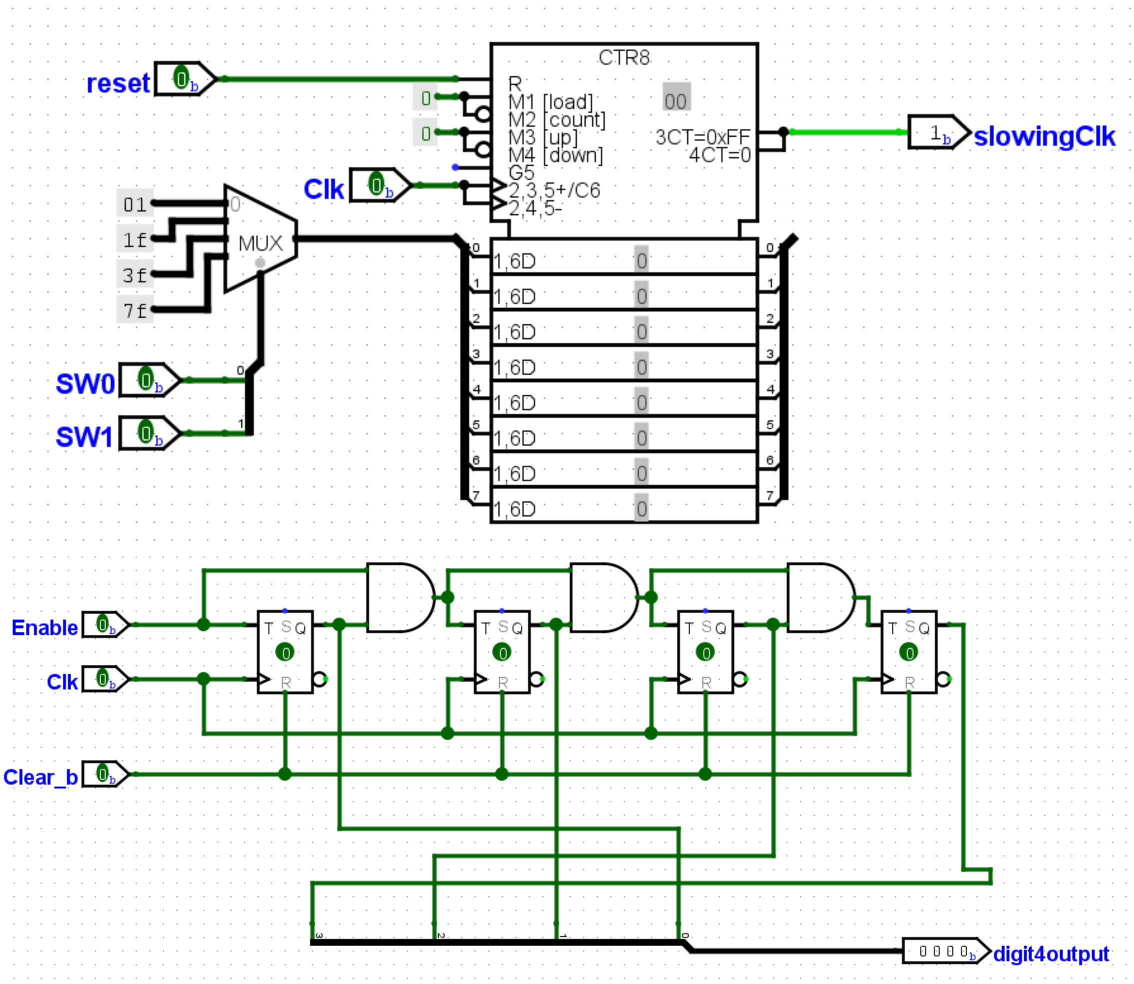
1. Wrap Around: the next value will be 0
2. Stay at Value: If the count device reach the maximum value then the value will stay at this maximum for any clk cycles
3. Continuing Counting: continue increase/decrease but keep the data bits
4. Load next Value: loading the given value provide at bottom left

Part II B

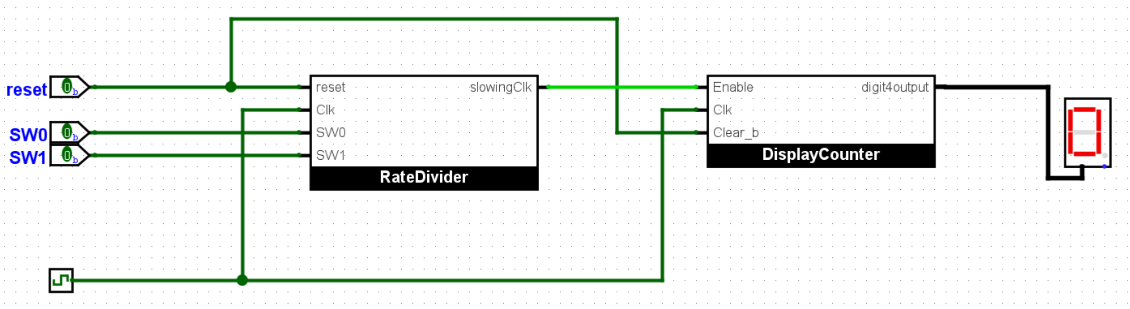
1. Calculate how large a counter would be required to count 50 million clock cycles, as illustrated by Figure 3. How many binary bits would that counter need to represent such a value?
   - each clk cycles we increase 1, so that after 50 million clock cycles we can at most have a value of 50 million, convert into binary, we got $2^{25} < 5000,0000 < 2^{26}$ so that e

should have at least 26 binary bits to represent the value

2. Draw a schematic of the circuit you wish to build. In particular, show how the outputs of the RateDivider feed into the inputs of the DisplayCounter, and how the high-level inputs and outputs connect to your entire circuit. Mentally step through the usage of this circuit to ensure that your circuit handles the necessary use cases to the best of your understanding.

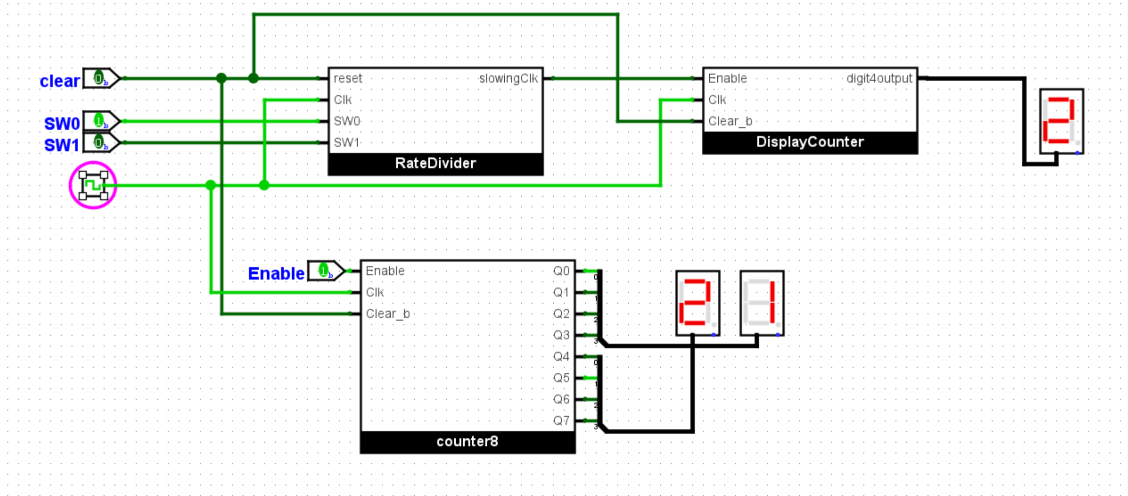


3. Build the circuit that realizes the behaviour described in your schematic. Your main circuit should have one clock input, one reset input and two external switch inputs (SW[1] and SW[0]) for the RateDivider. The circuit should have a seven-segment display as output.

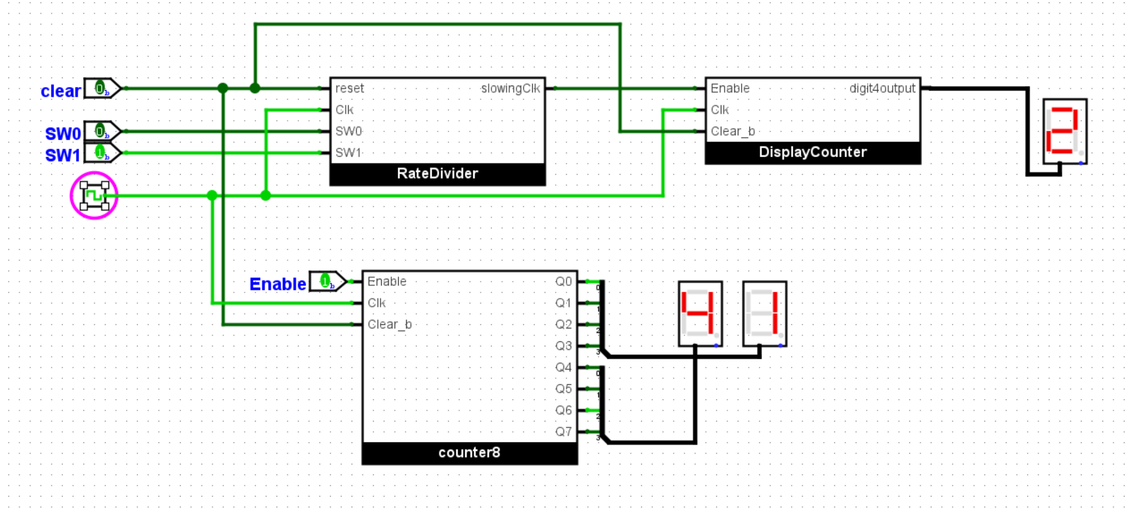


4. Simulate your modules with Poke. Choose test cases that make you feel confident about your counter's correctness in preparation for you in-lab demo. Make sure to include a few selected screenshots of these cases when you hand in your prelab. You will also need to think about the best way to simulate this kind of circuit. For example, how many 32 Hz clock pulses will you need to simulate to show that the RateDivider is properly outputting a 1 Hz pulse?

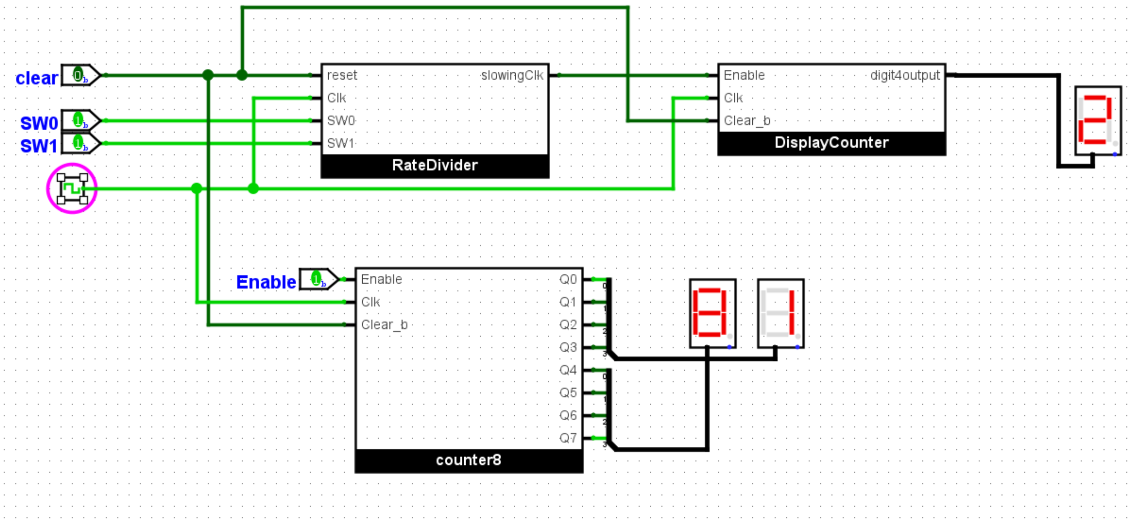
For `SW0=1,SW1=0` the speed we slowed into is `1HZ`. That's is both display start from 1, but after 1 second pass, the slowed one only increase 1, the other increase 20 in hex or 32 in dec



`SW0=0, SW1=1` the speed we should slowed into is `0.5HZ`. Both start from 1, the normal one increase 64 in dec, and the slowed one increase 1



`SW0=1, SW1=1` the speed we should slowed into is `0.25HZ`. Both start from 1, the normal one increase 128 in dec, and the slowed one increase 1



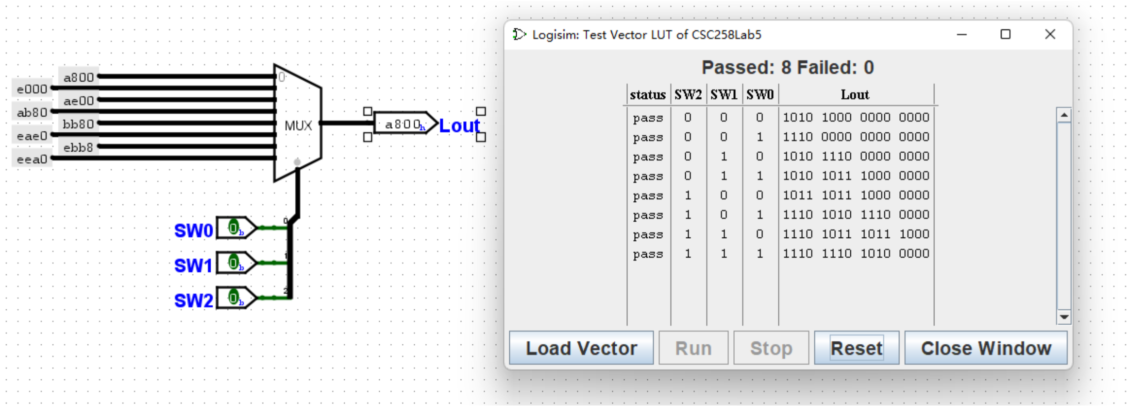
5. Label your high-level Logisim design with the appropriate input and output names that would be used on the DE1-SoC board (e.g. use HEX0 as the seven-segment display from the DisplayCounter while using switches SW1−0 to control the rate at which these hex digits are updated). You will also need to use additional switches (e.g., the clear signal).

Part III

1. Use Table 1 to determine your codes and bit-width.

Letter	Morse Code	Pattern Representation	HEX
S	•••	1010 1000 0000 0000	A800
T	—	1110 0000 0000 0000	E000
U	••—	1010 1110 0000 0000	AE00
V	•••—	1010 1011 1000 0000	AB80
W	•——	1011 1011 1000 0000	BB80
X	—••—	1110 1010 1110 0000	EAE0
Y	—•——	1110 1011 1011 1000	EBB8
Z	——••	1110 1110 1010 0000	EEA0

2. Design your circuit by first drawing a schematic of the circuit. Think about and work through your schematic to make sure that it will work according to your understanding. You can use Figure 4 as your starting point. You should include the schematic in your prelab.
3. Build the circuit in Logisim that realizes the behaviour described in your schematic.
4. Simulate your LUT with test vectors and your shifter module with Poke( ). Choose test cases that make you feel confident about your LUT and shifter's correctness in preparation for your in-lab demo. You are strongly encouraged to complete this step before performing your lab demo. Document the test cases that you considered (especially the corner cases) in your prelab report and include screenshots of the most interesting cases.



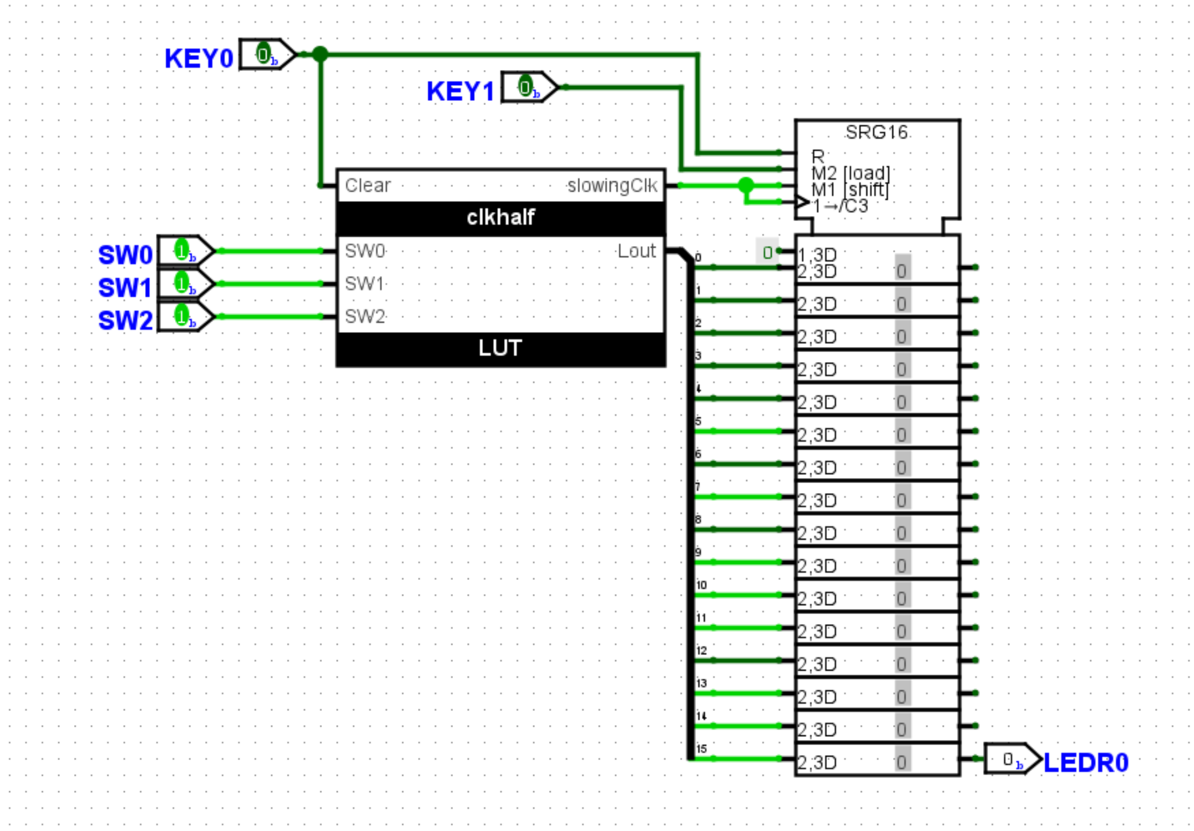
Passed: 8 Failed: 0

status	SW2	SW1	SW0	Lout
pass	0	0	0	1010 1000 0000 0000
pass	0	0	1	1110 0000 0000 0000
pass	0	1	0	1010 1110 0000 0000
pass	0	1	1	1010 1011 1000 0000
pass	1	0	0	1011 1011 1000 0000
pass	1	0	1	1110 1010 1110 0000
pass	1	1	0	1110 1011 1011 1000
pass	1	1	1	1110 1110 1010 0000

5. Label the inputs and outputs of your Logisim design to correspond to the inputs and outputs on the DE1-SoC board. The following table summarizes the inputs and outputs that you will use:

Input/Output	Purpose
SW[2:0]	Choose one of the 8 letters S to Z
KEY[1]	Start displaying Morse code for chosen letter
KEY[0]	Asynchronous reset
LEDR[0]	Output used to display Morse Code

2-5:


KEY0
KEY1
SRG16
R
M2 [load]
M1 [shift]
1→/C3
Clear
slowingClk
clkhalf
SW0
SW1
SW2
Lout
LUT
1,3D
2,3D
LEDR0