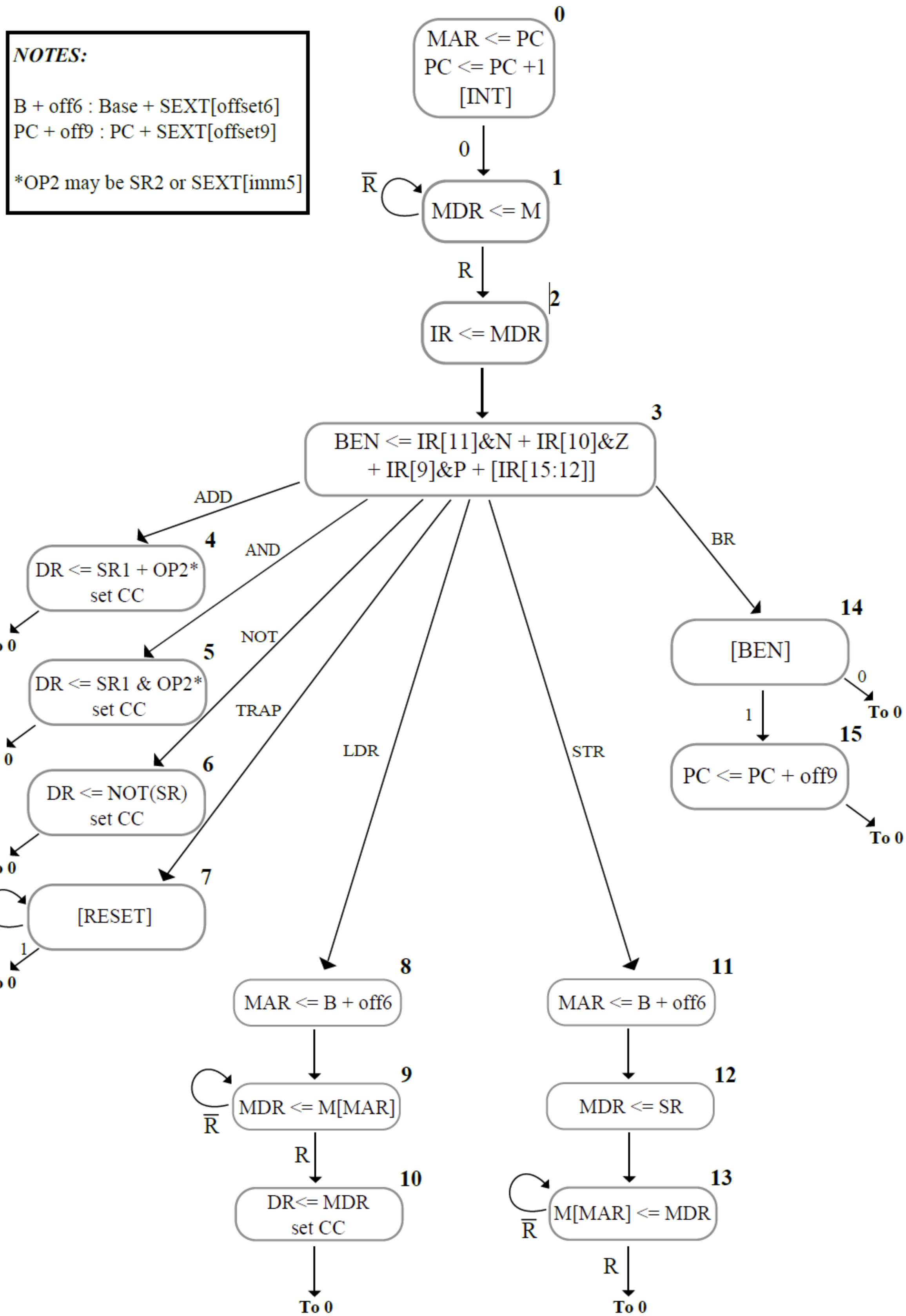
Lab 8 (optional): LC-3 Lite Control Unit

The LC-3 Lite Control Unit is provided in order to test the rest of your design. This page contains documentation, as well as Jade windows, for those who would like to design their own Control Unit. **This is an optional and ungraded lab.**

The Control Unit consists of a FSM as well as a memory called a *Control Store*. The Control Store contains all of the control bits for each cycle of the FSM. The Control Store is addressed using the four FSM state bits. When Reset = 1, the FSM starts in state 0, the first state of instruction fetch. Address 0 in the Control Store memory contains the control bits for that state, which are read and applied to the datapath. For example, the LD.PC bit at address 0 is a 1, since the PC register is updated with the incremented PC in the first instruction fetch cycle.

Below is a state diagram of the Control Unit FSM design:

Help



The FSM consists of a total of 16 states, requiring four state bits. When Reset = 1, the FSM begins in state 0. This is

Help

accomplished using D flip-flops with reset and set capability, connecting Reset to the reset FF inputs, and connecting the set FF inputs to ground. The TRAP-Halt instruction causes the FSM to be put into an endless loop, which freezes the design to permit register and memory values to be checked for correct operation.

Although the FSM diagram includes the R input from memory for completeness, since we assume a single cycle memory, the FSM state equations ignore R. The derivation of these equations is shown below:

	Opcode	BEN	Current State	Next State
	XXXX	X	0 0 0 0	0 0 0 1
	XXXX	X	0 0 0 1	0 0 1 0
	XXXX	X	0 0 1 0	0 0 1 1
(add)	0 0 0 1	X	0 0 1 1	0 1 0 0
(and)	0 1 0 1	X	0 0 1 1	0 1 0 1
(branch)	0 0 0 0	X	0 0 1 1	1 1 1 0
(LDR)	0 1 1 0	X	0 0 1 1	1 0 0 0
(not)	1 0 0 1	X	0 0 1 1	0 1 1 0
(STR)	0 1 1 1	X	0 0 1 1	1 0 1 1
(halt)	1 1 1 1	X	0 0 1 1	0 1 1 1
	XXXX	X	0 1 0 0	0 0 0 0
	XXXX	X	0 1 0 1	0 0 0 0
	XXXX	X	0 1 1 0	0 0 0 0
halting	XXXX	X	0 1 1 1	0 1 1 1
LDR	XXXX	X	1 0 0 0	1 0 0 1
LDR	XXXX	X	1 0 0 1	1 0 1 0
LDR	XXXX	X	1 0 1 0	0 0 0 0
STR	XXXX	X	1 0 1 1	1 1 0 0
STR	XXXX	X	1 1 0 0	1 1 0 1
STR	XXXX	X	1 1 0 1	0 0 0 0
no branch	XXXX	0	1 1 1 0	0 0 0 0
branching	XXXX	1	1 1 1 0	1 1 1 1
branching	XXXX	X	1 1 1 1	0 0 0 0

State Transition Equations:

$$\mathrm{S}_3 = S_3\overline{S_2}\,\overline{S_1}+S_3\overline{S_2}S_1S_0+S_3S_2\overline{S_1}\,\overline{S_0}+S_3S_2S_1\overline{S_0}(BEN)+\overline{S_3}\,\overline{S_2}S_1S_0[\overline{O_3}\,\overline{O_0}+\overline{O_3}O_2O_1O_0]$$

$$\mathrm{S}_2 = \overline{S_3}S_2S_1S_0+S_3\overline{S_2}S_1S_0+S_3S_2\overline{S_1}\,\overline{S_0}+S_3S_2S_1\overline{S_0}(BEN)+\overline{S_3}\,\overline{S_2}S_1S_0[\overline{O_3}\,\overline{O_1}\,O_0+\overline{O_3}\,\overline{O_2}\,\overline{O_1}\,\overline{O_0}+O_3]$$

$$\mathrm{S}_1 = \overline{S_3}S_2S_1S_0+S_3\overline{S_2}\,\overline{S_1}S_0+S_3S_2S_1\overline{S_0}(BEN)+\overline{S_3}\,\overline{S_2}S_1S_0[\overline{O_3}\,\overline{O_2}\,\overline{O_1}\,\overline{O_0}+O_3+\overline{O_3}\,O_2O_1O_0]+\overline{S_3}\,\overline{S_2}\,\overline{S_1}S_0+\overline{S_3}\,\overline{S_2}\,S_1\overline{S_0}$$

$$\mathrm{S}_0 = \overline{S_3}S_2S_1S_0+S_3\overline{S_2}\,\overline{S_1}\,\overline{S_0}+S_3S_2\overline{S_1}\,\overline{S_0}+S_3S_2S_1\overline{S_0}(BEN)+\overline{S_3}\,\overline{S_2}S_1S_0[O_2O_0]+\overline{S_3}\,\overline{S_2}\,\overline{S_0}$$

S_3-S_0 are the next states while S_3-S_0 (italicized) are the current states. O_3-O_0 are the opcode (IR[15:12]) while *BEN* is the branch enable. The equations for S_3-S_0 can be implemented using NOT, AND, and OR gates, and the four state bits using D flip-flops with reset/set capability.

The contents of the Control Store memory are shown below:

		*****************LOAD SIGNALS*********************						~~~~~~~~~~~~~~~MUX CTL SIGNALS~~~~~~~~~~~~~~~							
	STATE	PC	IR	REG	MAR	MDR	CC	PC	ADDR1	ADDR2	SR1	MDR	ALUK[1:0]	R/W	BUS.OUT[1:0]
Fetch 1	0 0 0 0	1	0	0	1	0	0	0	0	0	0	0	0 0	0	1 0
Fetch 2	0 0 0 1	0	0	0	0	1	0	0	0	0	0	0	0 0	0	0 0
Fetch 3	0 0 1 0	0	1	0	0	0	0	0	0	0	0	0	0 0	0	0 0
Decode	0 0 1 1	0	0	0	0	0	0	0	0	0	0	0	0 0	0	0 0
ADD	0 1 0 0	0	0	1	0	0	1	0	0	0	0	0	0 0	0	1 1
AND	0 1 0 1	0	0	1	0	0	1	0	0	0	0	0	0 1	0	1 1
NOT	0 1 1 0	0	0	1	0	0	1	0	0	0	0	0	1 0	0	1 1
TRAP-Halt	0 1 1 1	0	0	0	0	0	0	0	0	0	0	0	0 0	0	0 0
LDR	1 0 0 0	0	0	0	1	0	0	0	0	0	0	0	0 0	0	0 1
LDR	1 0 0 1	0	0	0	0	1	0	0	0	0	0	0	0 0	0	0 0
LDR	1 0 1 0	0	0	1	0	0	1	0	0	0	0	0	0 0	0	0 0
STR	1 0 1 1	0	0	0	1	0	0	0	0	0	0	0	0 0	0	0 1
STR	1 1 0 0	0	0	0	0	1	0	0	0	0	1	1	1 1	0	1 1
STR	1 1 0 1	0	0	0	0	0	0	0	0	0	0	0	0 0	1	0 0
BR	1 1 1 0	0	0	0	0	0	0	0	0	0	0	0	0 0	0	0 0
BR	1 1 1 1	1	0	0	0	0	0	1	1	1	0	0	0 0	0	0 0

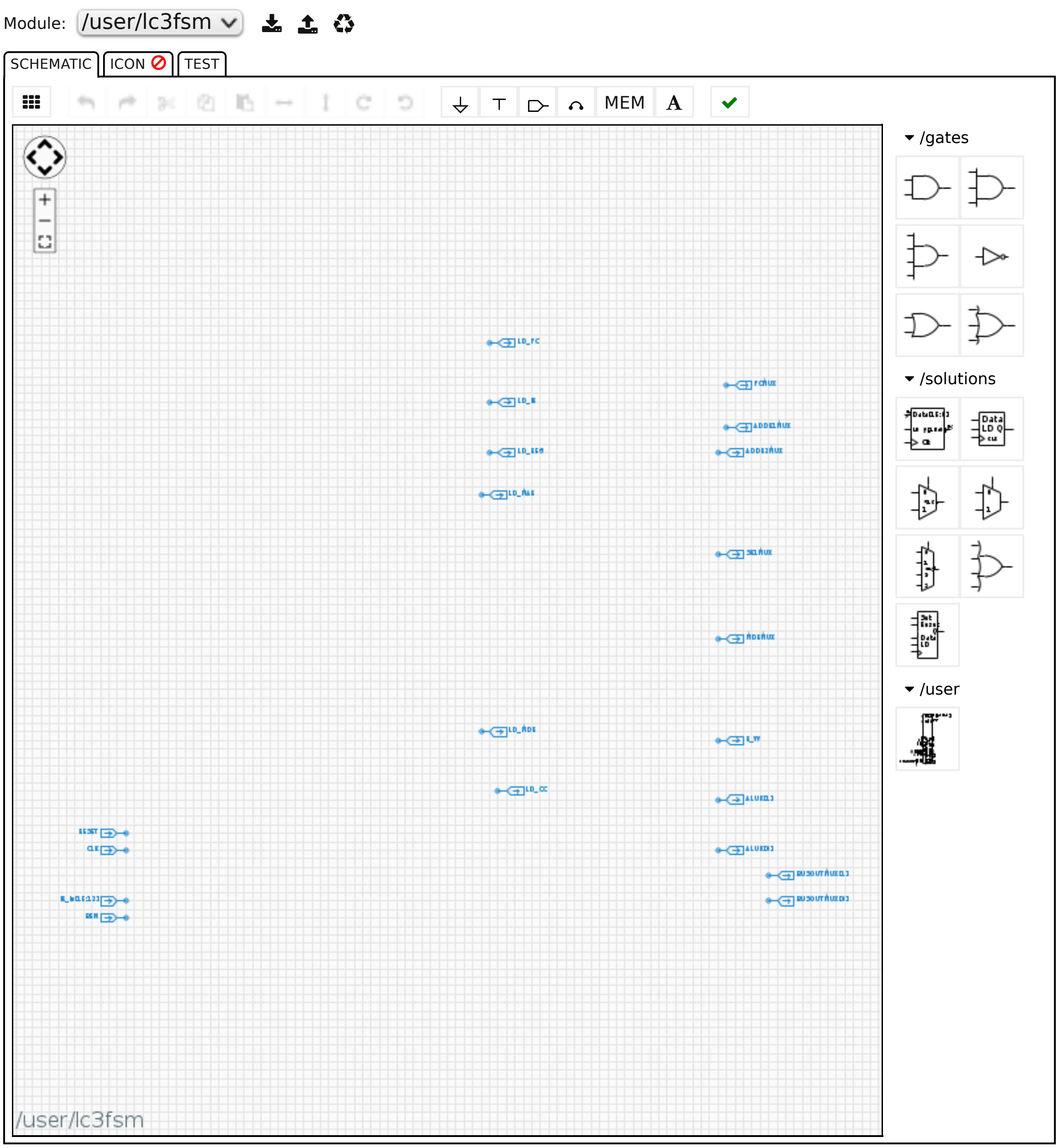
Each row of the table shows the state name, state address, and the values of the 16 control signals that control the LC-3 Lite datapath. These consist of 6 LD controls, 5 2-to-1 mux controls, ALUK[1:0], the memory R/W, and the 4-to-1 BUSOUT mux control. The Control Store can be created from a Jade memory with 16 locations, each of which is 16 bits wide. An Excel

Help

version of the Control Store can be found here.

There are two Jade windows below. One to design your FSM, and the second is a copy of the complete LC-3 lab. You can load your LC-3 Top design and replace the solutions FSM with your version to test it. Note the Control Unit lab has a user-writable Test tab. We've filled it with a comprehensive test, but you can write your own tests to make sure your Control Unit is working as well. You can also test smaller parts of the unit at a time, by writing different tests. Refer to the "How to Use Jade" page for testing information.

LC-3 LITE CONTROL UNIT



Click component to select, click and drag on background for area select, shift-click and drag on background to pan

Jade 2.2.43 (2015 © MIT EECS)

Help

Error detected:
Circuit does not have a node named "ld_pc".

Circuit does not have a node named "ld_ir".

Circuit does not have a node named "ld_reg".

Circuit does not have a node named "ld_mar".

Circuit does not have a node named "ld_mdr".

Circuit does not have a node named "ld_cc".

Circuit does not have a node named "pcmux".

Circuit does not have a node named "addr1mux".

Circuit does not have a node named "addr2mux".

Circuit does not have a node named "sr1mux".

Circuit does not have a node named "mdrmux".

Circuit does not have a node named "aluk[1]".

Circuit does not have a node named "aluk[0]".

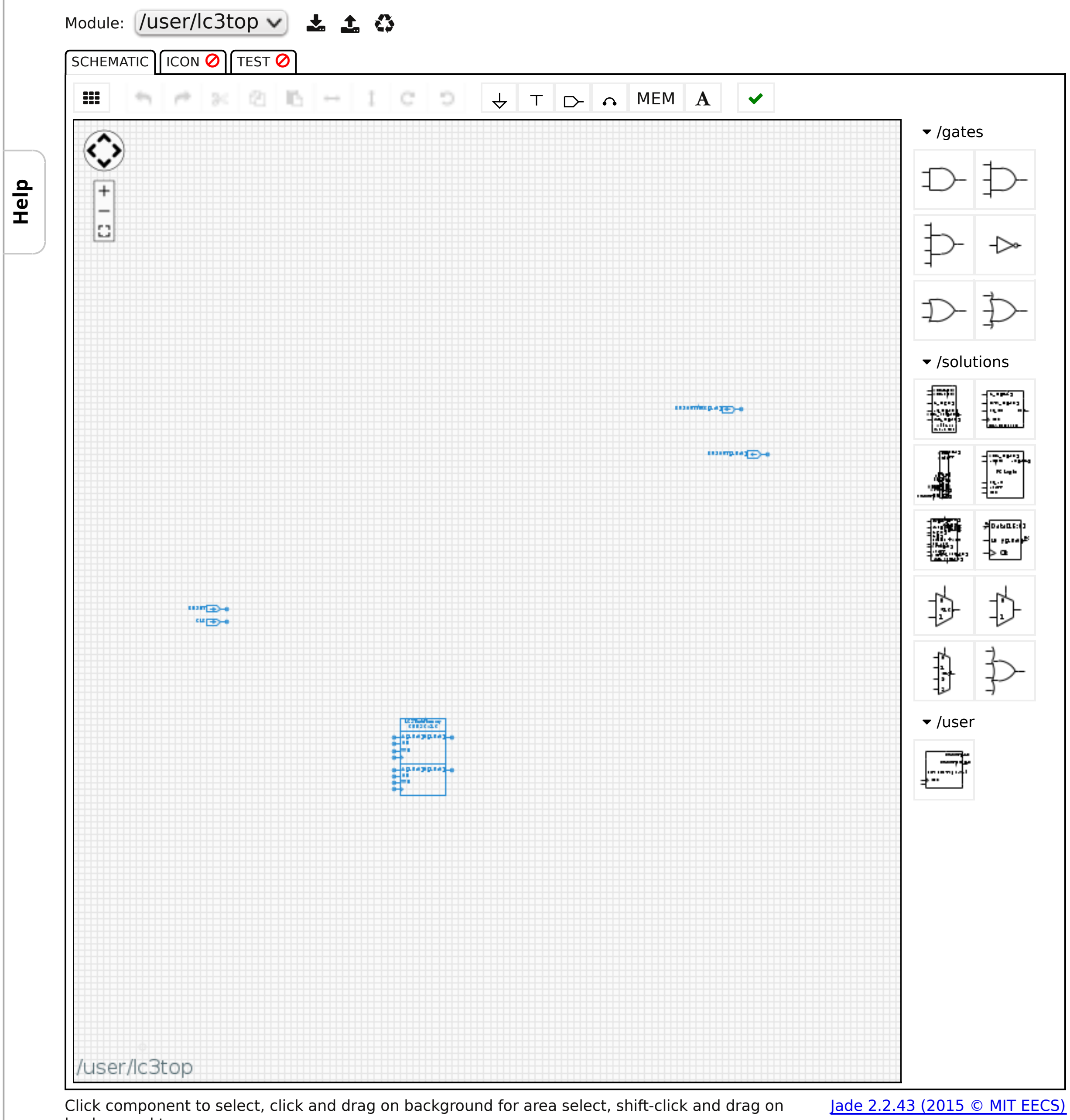
Circuit does not have a node named "r_w".

Circuit does not have a node named "busoutmux[1]".

Circuit does not have a node named "busoutmux[0]".

Check

LC-3 LITE



Error detected:

Circuit does not have a node named "busout[15]".

Circuit does not have a node named "busout[14]".

Circuit does not have a node named "busout[13]".

Circuit does not have a node named "busout[12]".

Circuit does not have a node named "busout[11]".

Circuit does not have a node named "busout[10]".

Circuit does not have a node named "busout[9]".

Help

Circuit does not have a node named "busout[8]".

Circuit does not have a node named "busout[7]".

Circuit does not have a node named "busout[6]".

Circuit does not have a node named "busout[5]".

Circuit does not have a node named "busout[4]".

Circuit does not have a node named "busout[3]".

Circuit does not have a node named "busout[2]".

Circuit does not have a node named "busout[1]".

Circuit does not have a node named "busout[0]".

Circuit does not have a node named "busoutmux[1]".

Circuit does not have a node named "busoutmux[0]".

Check

Because this is an optional, ungraded, assignment, feel free to share code and screen shots in the discussion. However, put "[spoiler]" in your title to warn other students who may want to do the lab on their own without any hints.

Show Discussion

New Post