Fall 2020 - ECE 394: Digital Systems Lab

LAB REPORT Lab 8: 4-bit RPN Calculator

Group No: 4 **Date: 11/30/2020**

Group Members: Sayem Ahmed, Taylor Bukoski, Pedro Mena

Name: Sayem Ahmed

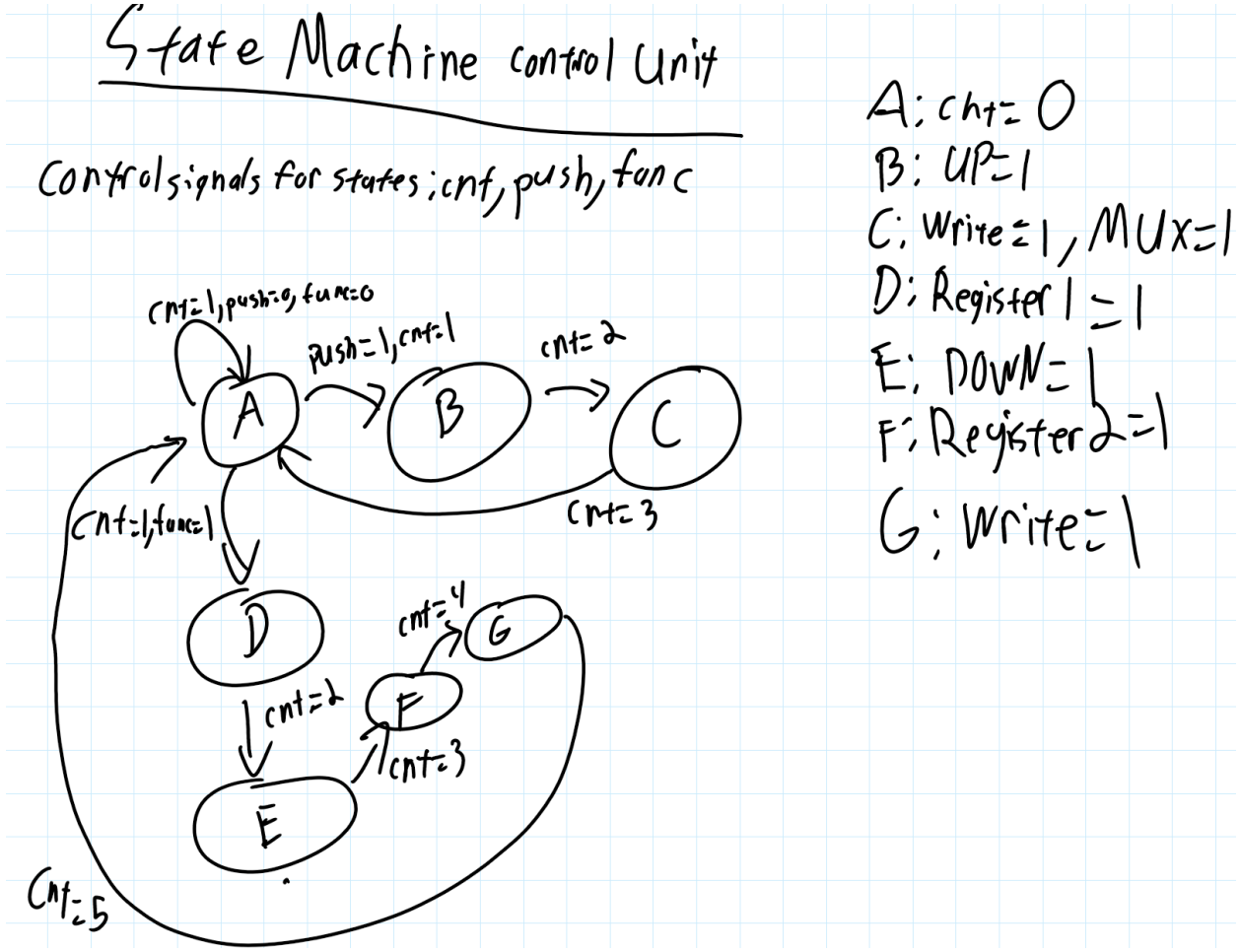
Name: Taylor Bukoski

Name: Pedro Mena

Procedure:

Constructing the Control Unit:

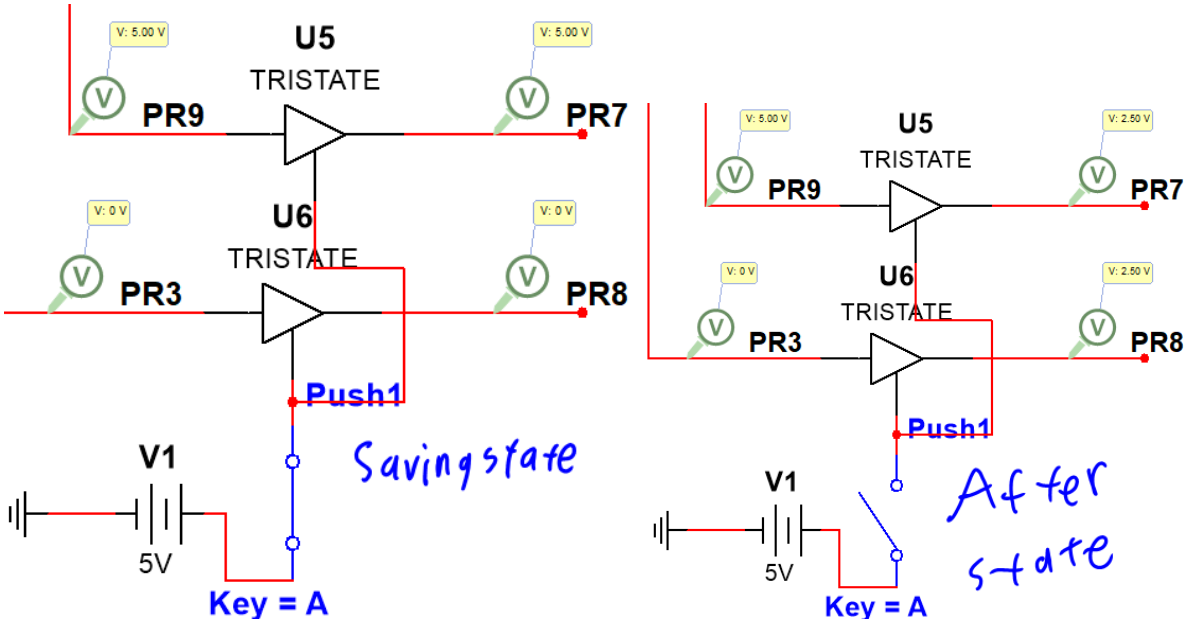
For this lab we need to create a control unit that will send the signals for our calculator, to do this we decided to go with a state machine approach in order to accomplish our goals. For this case we will have two main paths in our states one path for when we wish to push to the stack and another path for doing functions, one path will use switches X as input while the other path will use switches Y as Input. We will also implement a counter in order to perform the necessary steps for each task the control unit has to complete.



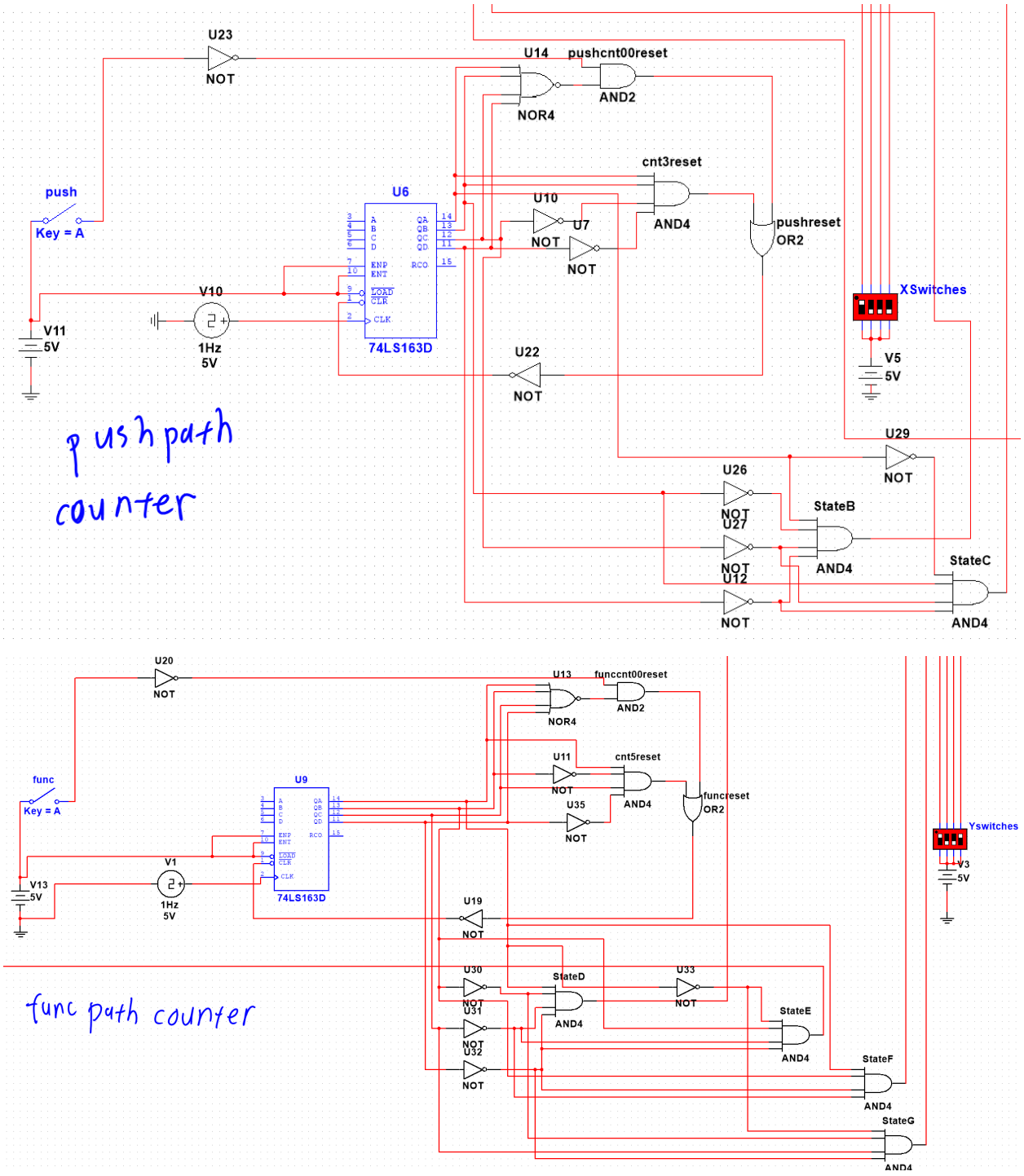
Implementing the state machine in Multisim:

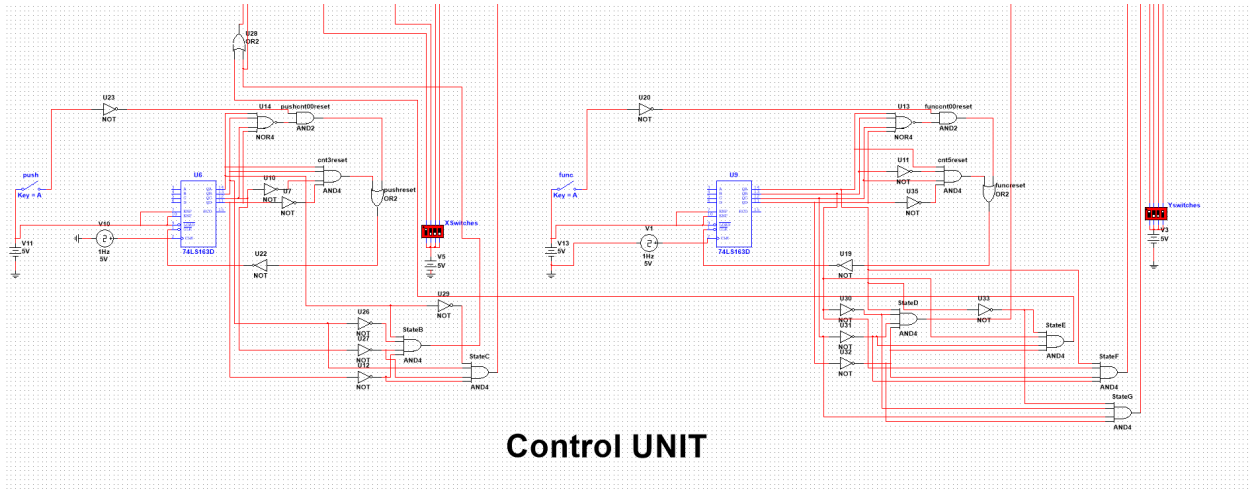
When constructing the state machine in multisim we need a few basic components, for one we need a counter in this case we used a 74LS163D, a variety of gates to check the output of our counter, and our switches x, y, and for the inputs func, and push, lastly we need a square wave generator. Now an important detail of this design is that the inputs push and func do not stay on the entire time they are running only once to begin their paths this means we need a way to keep track of what path we are running through since depending on the path we take we need to return to State A at a counter value of 3 or a counter value of 5. Ideally we would use a tri state to save

which signal was sent on our run and then use that to know when to return to state A and then reset the tristate upon reaching state A. However multisim seems to have an issue with tri states we encountered every time we attempted to use them in this lab, for one tri states don't store a high value of 5V but instead 2.5V which isn't recognized as a digital 1 by the IC's as they all require a 5V signal to recognize a digital 1. Not only that but it seems to always output 2.5V even if the saved value should be 0, an example of the error is listed below.



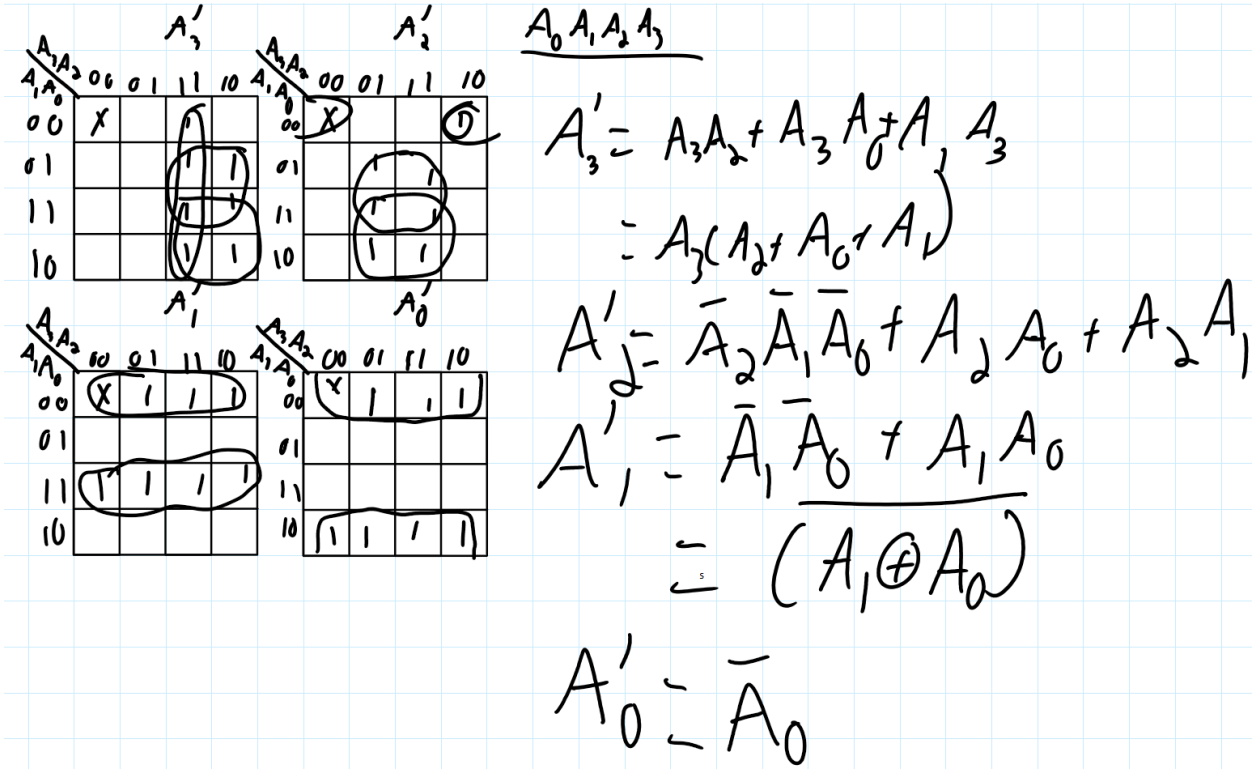
As a result of this issue not only are we unable to use tristates here we will not be able to use them later on as well this will be important for implementing the registers. As a result of not being able to use tri-states another approach was taken instead we would use 2 different counters each one having the input push and func to themselves respectively. This allows us to tell when we should reset but adds complexity and overall space and power usage into our design and would be avoided for a hardware implementation. To make sure that we don't perform accidental commands when no task is being requested we have to make sure that our counter resets to state A unless the signal func or push is sent. To do this we simply detect if we are in state A by checking for counter output 0000 and if we are we check if our push or func signal is high if so we allow the counter to proceed if not we send a clear signal to the counter. When we are conducting tasks we will be checking what the counter's output is and using a series of gates send the necessary signals from our control unit to our core hardware to correctly correspond with the states established in our state machine. Next we will reset when our counter reaches an output of 3 for the push counter and when it reaches 5 for our func counter. Lastly our two other switches x and y will not be modified and will instead be hardwired to the internal hardware of our calculator.


U23
NOT
U14
pushcnt00reset
NOR4
AND2
cnt3reset
push
Key = A
U6
U10
U7
NOT
NOT
AND4
pushreset
OR2
XSwitches
V10
V11
5V
1Hz
5V
74LS163D
V5
5V
U22
NOT
push path
counter
U29
NOT
U26
NOT
U27
NOT
U12
NOT
StateB
AND4
StateC
AND4
U20
NOT
U13
funccnt00reset
NOR4
AND2
U11
cnt5reset
NOT
AND4
funcreset
OR2
U35
NOT
func
Key = A
U9
Yswitches
V1
V13
5V
1Hz
5V
74LS163D
V3
5V
U19
NOT
U30
NOT
U31
NOT
U32
NOT
StateD
AND4
U33
NOT
StateE
AND4
StateF
AND4
StateG
AND4
func path counter

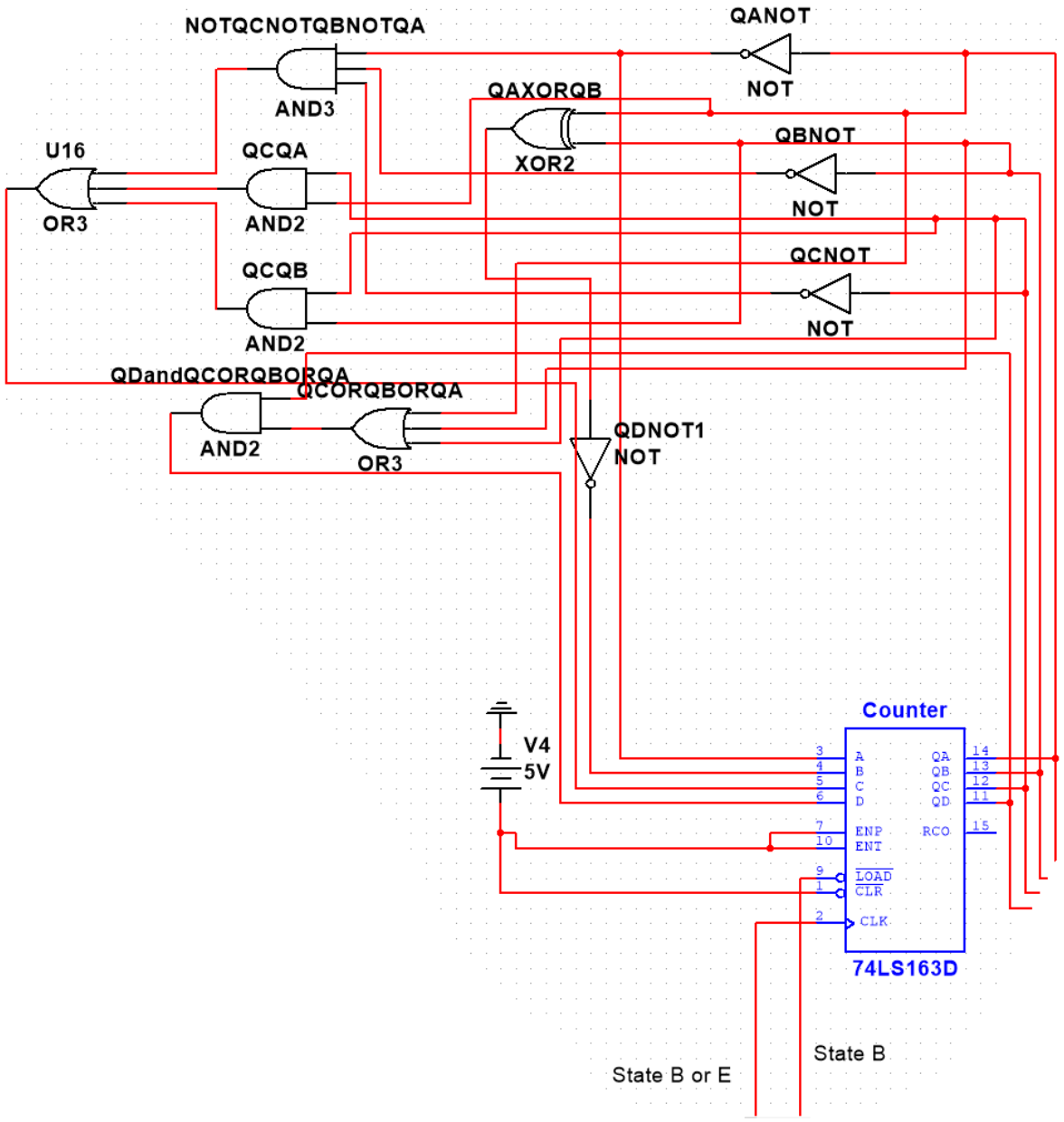


Implementing our counter:

Our counter in this lab will have to accomplish 2 tasks incrementation and decrementation, incrementation is simple as the default function of a counter is to increment whenever a clock cycle goes by. However, decrementation as it is not built into the counter requires some extra wiring on our part, however thankfully the counter we are using in this lab, the 74ls163D that we used for the control unit, has a load feature meaning that we can decrement by simply taking the current output of our counter subtracting by 1 and loading onto the counters input. We decided to implement this using karnaugh maps constructing a karnaugh map and the corresponding functions for each bit going into the counter.



Using those functions we can construct the following counter

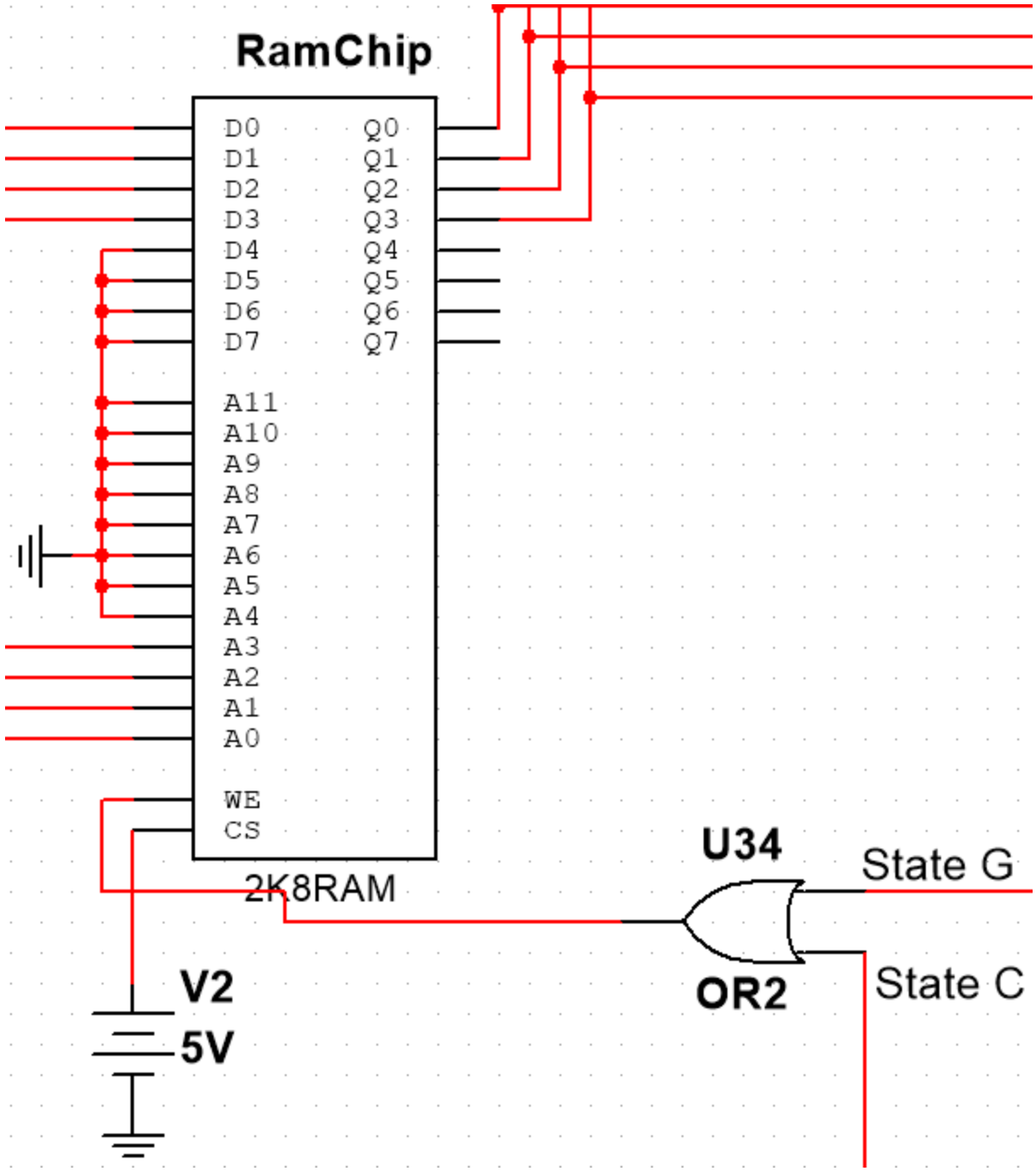


An important note is that since our load signal loads on low when we increment we have to specify we are not loading, and when we decrement we simply have to send a clk signal, that is why state B and E both send a clock signal but only state B sends a load signal. The inputs enp, ent and clr are also set high for this counter, while the output RCO is never used.

Implementing the RAM chip

There is only one ram chip we were able to find in multisim and it was the 2K8RAM chip, an important note about this chip is that has 8 inputs both for the address and actual values stored, however since all our chips and values are 4 bits we will simply ground the extra bits we do not need. The CS signal is used for outputting values, however since our circuit works even if we are

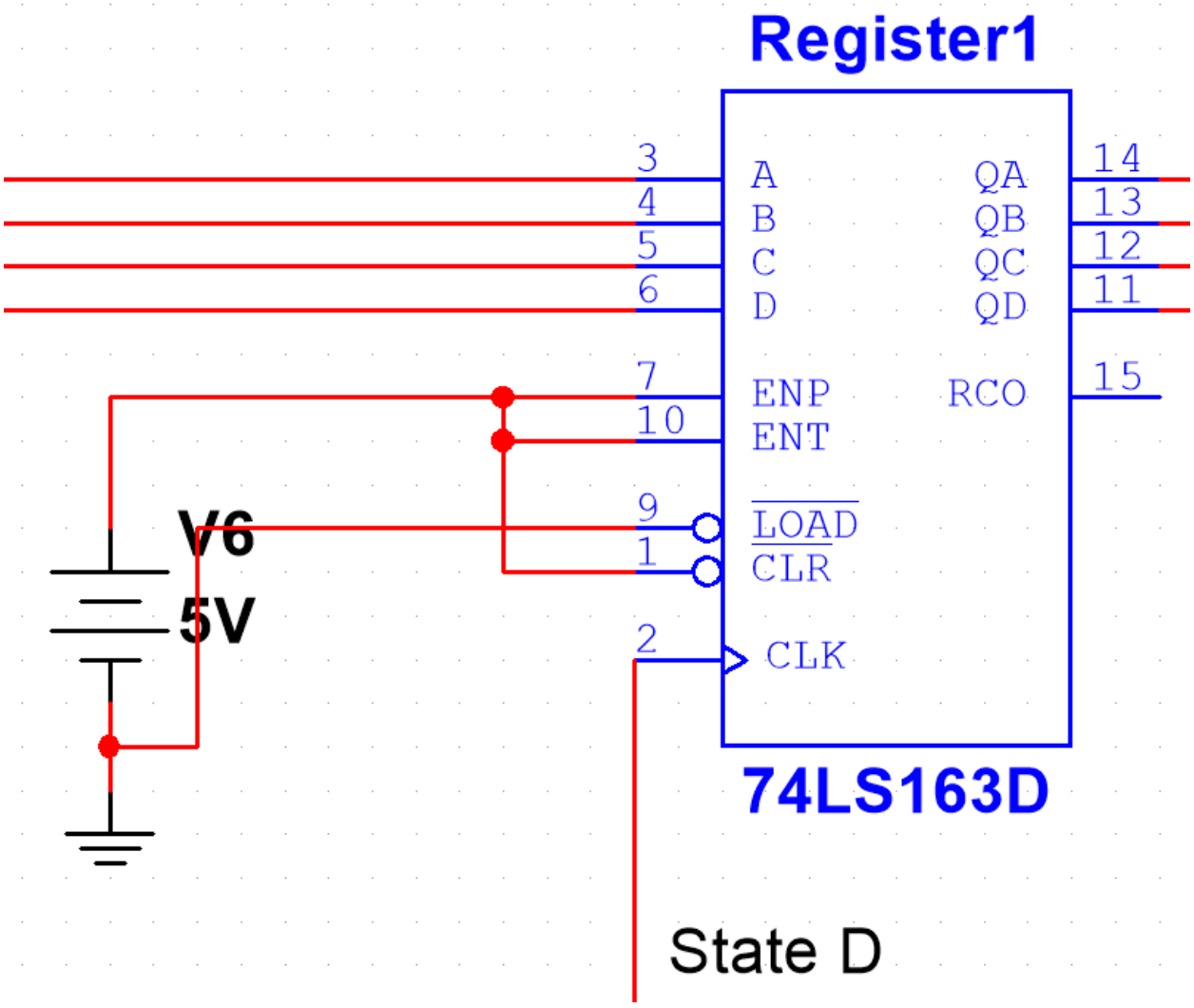
constantly outputting we will simply assert it high. Inputs for the value being stored will come from the MUX and the address being selected will come from the counter. The WE input will write to the Ram chip when asserted high thus we set that signal to equal states G and C.



Implementing Register 1

The Registers for this lab are vital to achieving our calculator, that is because our ram chip can only output one value at a time but our ALU requires 2 inputs to perform an operation, thus we must save one of our operands onto a register and then output the other directly from the ram

chip. Ideally we would like to use tristates to accomplish this goal, however as mentioned previously they did not work properly at least for us and so we had to take another approach. So we decided to implement our register using a counter chip because as shown previously we can implement a counter that can load values and will not take in new values unless a load signal is sent. Thus we decided to implement our Registers using the same counter chip 74ls163D, however we will have it set up so the load signal is always active and thus when we send a clk signal to the counter it will simply load the value on its inputs into the counter and always output whatever value is currently saved.



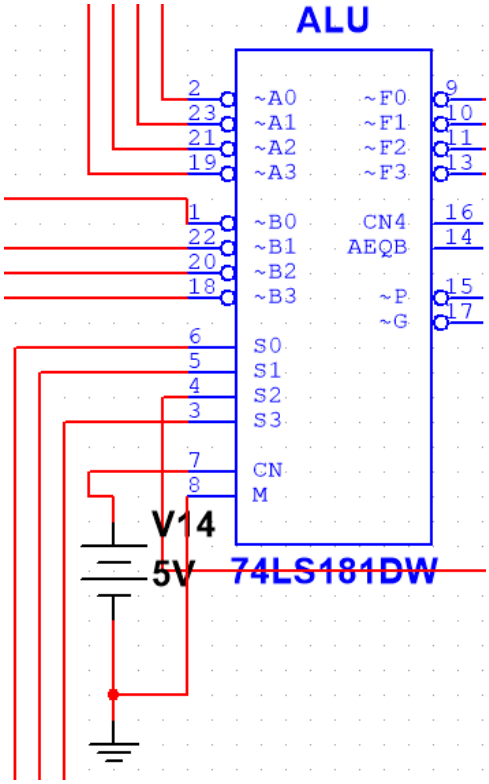
We want to load when at State D and the inputs enp, ient, and clr are all set high whereas our load is set low since it is asserted low and thus a low signal signifies a load. These output values besides RCO will be sent to the first input for the ALU.

Implementing the ALU:

For this lab we need to use an ALU in order to perform the actual calculations of oru calculator. For this lab we have chosen to use the 74ls181DW, this ALU has two inputs A and B, input A will come from Register 1 whereas input B will come directly from the RAM. An important note about this ALU is that its function is chosen by 6 inputs S0-S3 and CN and M for this lab we have chosen to set CN to high and M to low with S0-S3 being chosen by the control unit from switches Y with the different combinations for S0-S3 that will give our functions being listed below.

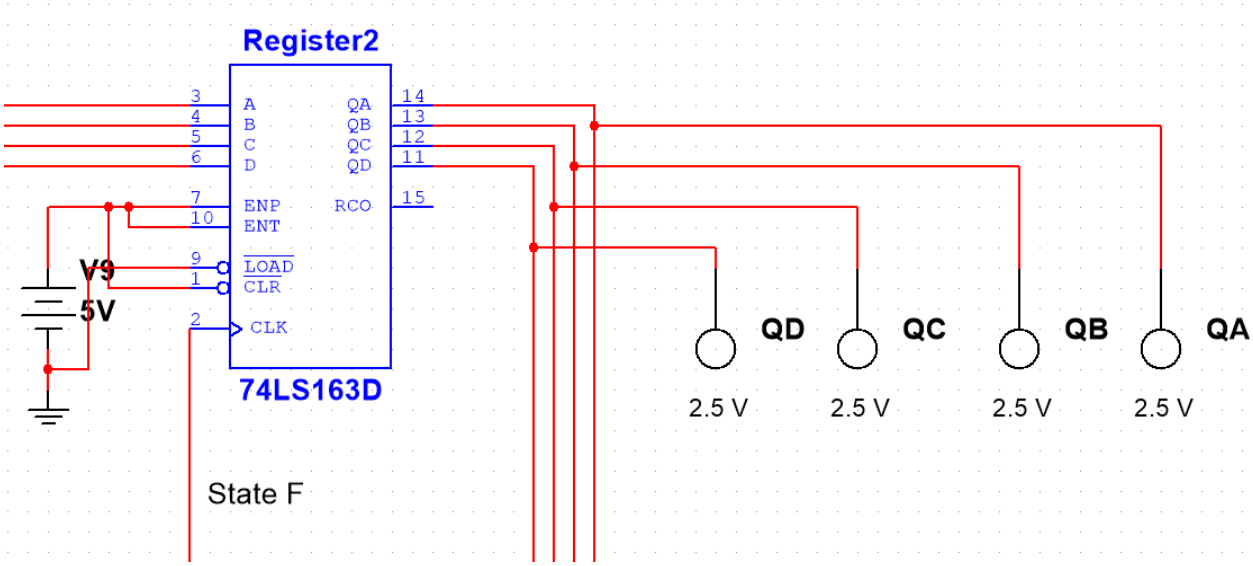
S3	S2	S1	S0	Arithmetic (Note 2) (M = L) (C_n = H)
L	L	L	L	A
L	L	L	H	A + B
L	L	H	L	A + $\overline{B}$
L	L	H	H	minus 1
L	H	L	L	A plus A$\overline{B}$
L	H	L	H	(A + B) plus A$\overline{B}$
L	H	H	L	A minus B minus 1
L	H	H	H	AB minus 1
H	L	L	L	A plus AB
H	L	L	H	A plus B
H	L	H	L	(A + $\overline{B}$) plus AB
H	L	H	H	AB minus 1
H	H	L	L	A plus A (Note 1)
H	H	L	H	(A + B) plus A
H	H	H	L	(A + $\overline{B}$) plus A
H	H	H	H	A minus 1

The outputs of the ALU will go to Register 2 and since the ALU is always outputting we do not need to send a signal to the ALU for any specific state.



Implementing Register 2:

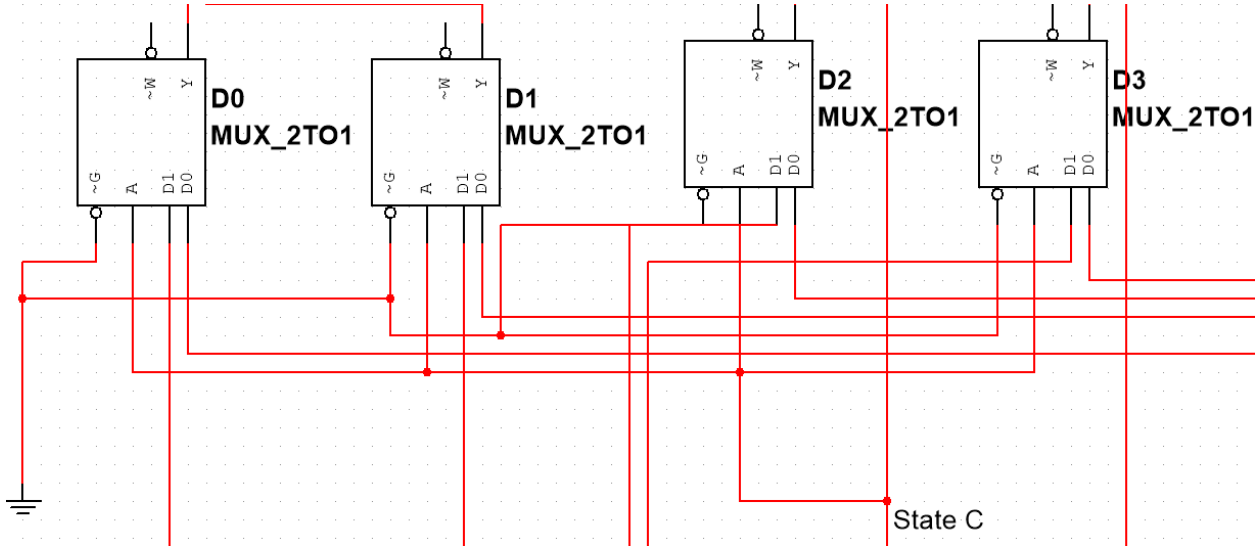
As stated previously since we could not use tri states in this lab we implemented our Registers using a counter. Register 2 will be implemented in nearly the same way as Register 1 however, this time our Register will load its values at state F and our inputs are coming from the ALU instead of the RAM chip.



Because Register 2 will act as our output we have hooked up 4 LEDs to it.

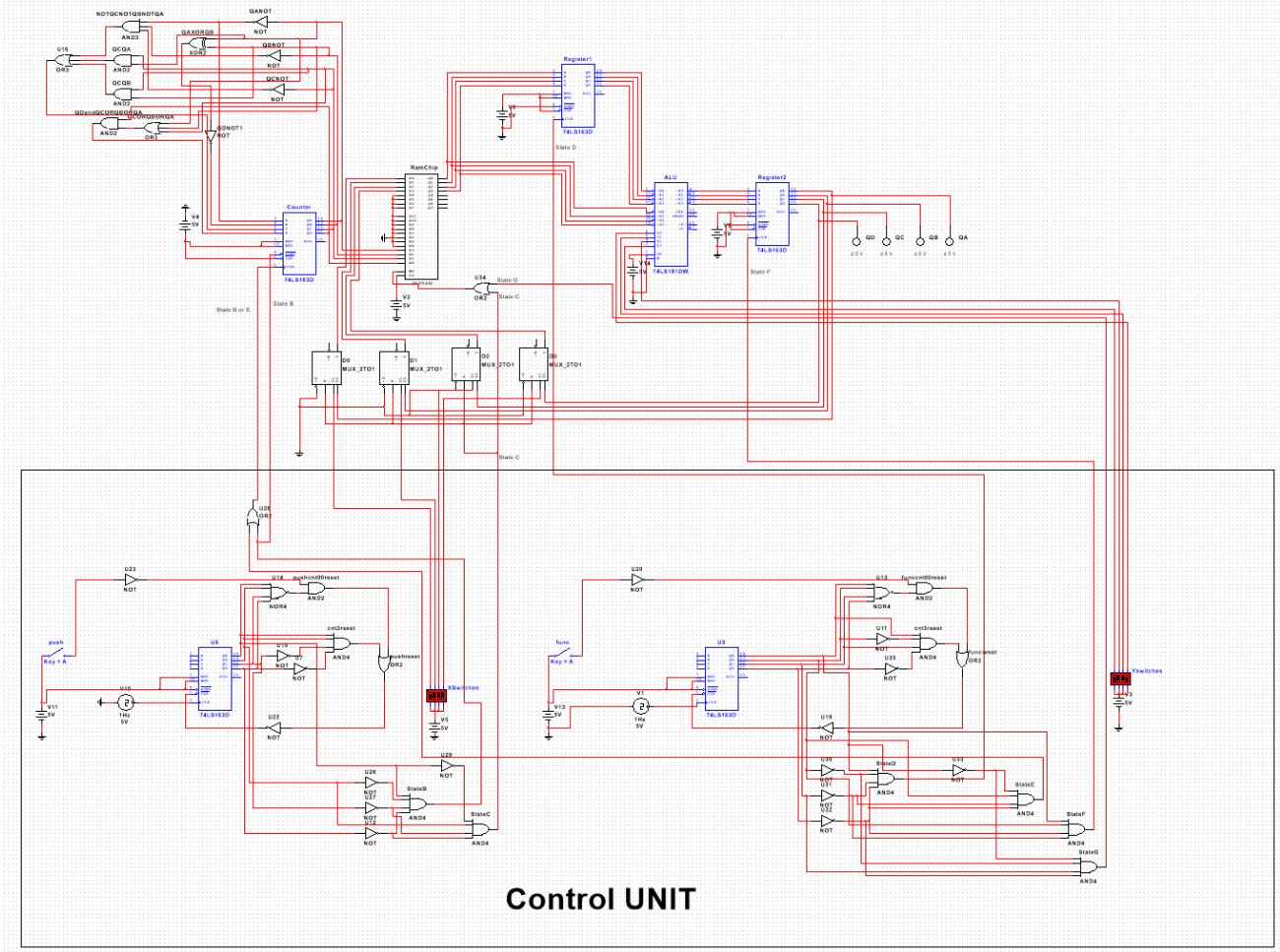
Implementing the MUX:

Our MUX in this lab will decide what inputs are going to be sent to the RAM chip, because of the fact that we save all our outputs back onto RAM and any variables that we want to directly load from switches X also have to be sent onto RAM we need a MUX in order to decide which will get priority depending on the current operation. For our purposes we want to load the inputs for switches X at State C and to load the inputs from Register 2 at state F. However, since the MUX we are using only has two inputs and the Ram chip itself only loads values when a load signal is sent there is no harm in having one input be prioritized over the other. That is to say we can simply construct our MUX to choose Switches X at State C and for all other states the MUX will be loading from Register 2, this way we minimize the hardware used and complexity. For this lab we had issues in finding a mux with a 4 bit input so we decided to simply use 4 1 bit muxes all with the same select signal.



A is set to equal State C thus accomplishing our goal of only choosing from switches X at state C and defaulting to Register 2 in all other states. G is our ground and all of switches-X signals are sent to D1 and all of Register 2's signals are sent to D0. The outputs will be sent to the ram chip with only Y sending output where as W is not wired

Overall Calculator:



Conclusion: This lab taught us about many different concepts some of them include, working with state machines on a more detailed level. Understanding control units and knowing how to design for user input but also minimize the amount of work the user has to input in order to get the desired output. Just as an example there are 8 different control signals for this lab however, we were able to cut it down to just 4 inputs for the user and depending and only 2 inputs need to be used for a single function. Another important concept we learned was that of working with what you have in this case we were not able to use tri states, and often had to use chips that got the job done but carried some extra costs, either in complexity or wasted space, however we were still able to complete the lab with the circuit from the user's perspective being no different from what they would have to deal with had the circuit been implemented properly. Overall our circuit functioned properly and will seem normal to any user using it possibly the only point of failure is that because we have to different counters in the control unit it means that it would be possible for a user to send both a push and func signal at the same time which would likely result in an incorrect output, however that is neither in spec and could be easily prevented had we been able to use tri states as previously planned. Some Final notes about the control unit are that due to the 1hz clock rate we set for ease of use the user will have to leave the switches for push and func for at least a second for the signal to be received and then let go. Also to note that our bits for switches X and the output are set up so the MSB is on the left and our LSB is on the right.