Ryan Zurrin
John Osthoff
ENT 233 Digital Circuits

December 15th, 2019

FINAL PROJECT / Coffee & Cocoa Vending Machine
The Design Process from First Ideas to Simulation Implementation.
THE STRUGGLE IS REAL

By Ryan Zurrin

1. Project introduction

For the final project in digital circuits we were giving an outline and project requirements which had to be fulfilled for the project to be considered a success. The Final project is to be a ***Coffee and Hot Cocoa Vending machine***. The requirements for this design are as followed:

- **Sensors** which must detect if there is a current supply in the machine of the following items:
  1. Cups (a requirement for any of the items to be dispensed)
  2. Sugar
  3. Cream
  4. Coffee
  5. Cocoa
  6. Change Available
- **Inputs** which will be simulated using push buttons:
  1. $.25 cents (the quarter slot)
  2. $1 dollar
  3. $5 dollar
  4. Coffee selected
  5. Cocoa selected
  6. Cream selected
  7. Sugar selected
- **Indicators** which will be logic indicators, LED's, and 7-segment displays:
  1. Decimal display of money added to machine
  2. Product availability indicator
  3. Successful Vend
  4. Unsuccessful Vend
  5. Change due as decimal display
  6. Any indicators within the circuit that would help a technician with any troubleshooting or repair procedures

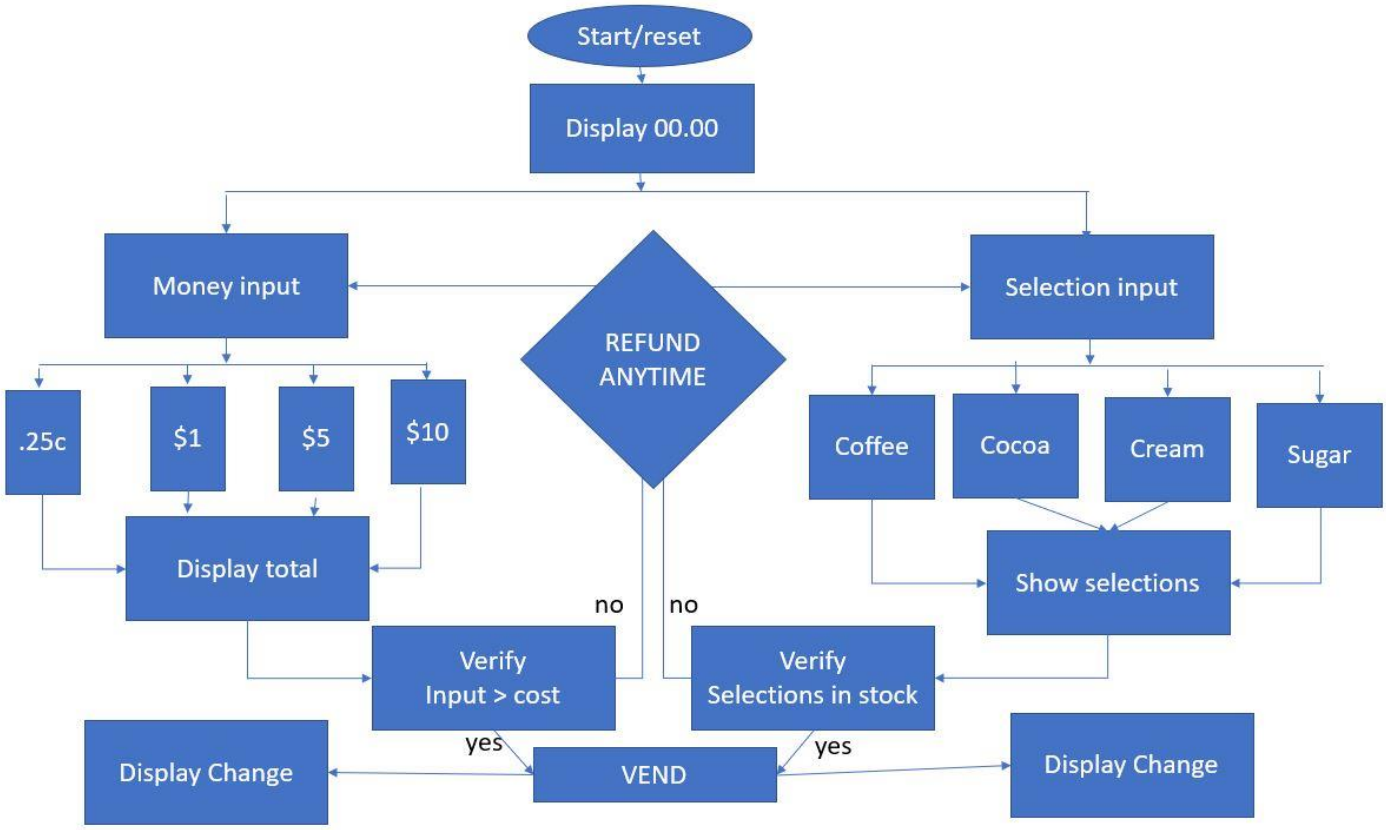
There is a lot to think about and consider in this design and many ways to approach a working solution. When I was first trying to reason out how I was going to approach this I really didn't know exactly what to do. With so many things to think about, where do I even start? I just read and reread the requirements and remembered our instructor John talking a lot about modular design and how we should approach the design in smaller chunks breaking it up, essentially making it all manageable. If this is the best approach, then what modules will there be and how will they all work? I needed to start with the basics, to get out the pencil and paper and start scribbling ideas down as well as put together a flow chart of the process and how it should play out. From this I should be able to determine the order of my design and which modules should take on a greater precedence. I do this based on circuit design difficulty levels and how much other parts of design rely on any modules. The documentation that follows is the complete process from beginning to end of this project and how I designed a working solution which should be able to work in a real world prototype if it was to be built up based on this circuit design.

Things to note, this design was built using multiSIM software which is decent design software but there were some minor setbacks while moving forward. First off, clock speeds in a simulated circuit do not reach high speeds so using ripple counters to keep track of things in the simulation ended up being much slower than I would have liked them to be. And simulation memory size had to be upped to 30GB to accommodate the simulation data being produced.

In the beginning the Boss gave us The Design requirements,

And these requirements were void and without any form, there was an emptiness upon the schematic, but the spirit of ideas ran deep, and from the dark depths of emptiness a schematic was filled with function and logic, So when the Boss Man said let there be cocoa there was, and likewise when he said let there be coffee there also was, of course supply line dependent that is. And when the Boss seen his ideas become real, he was pleased by this and he drank, coffee of course, all while smiling knowing yet again, another great circuit has been born. Now Let the design games begin!

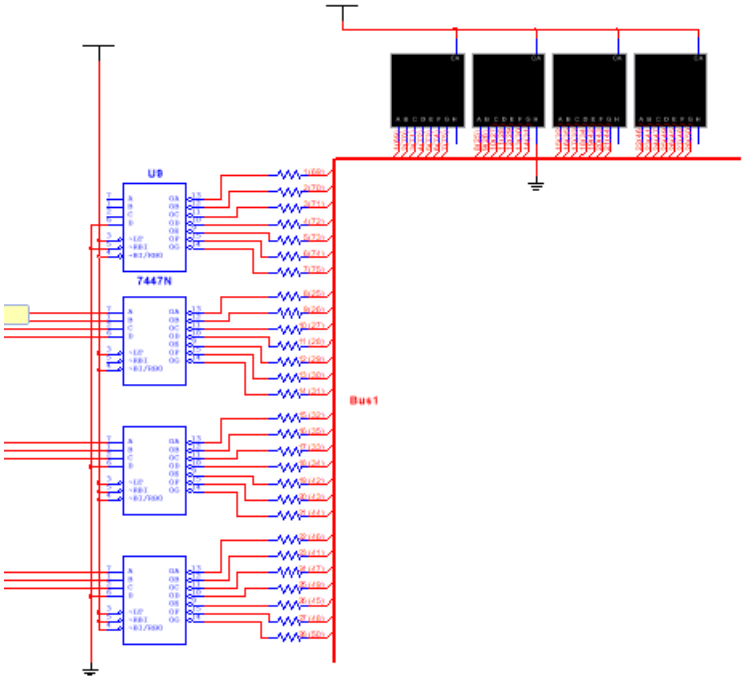
FLOWCHART (kind of) for hot cocoa and coffee vending machine.



This isn't an actual flow chart but just a flow of what should kind of be happening and when. It gives me a general idea of the types of modules Ill need to work on in order to break down the design properly that is. What I'm thinking when looking at this is a display for money and either the same or separate display for the change, though I am pretty sure it will need all its own ICs and circuitry. I am looking at a module for the .25c, the $1's, $5's, and $10's. It will need to do some kind of math to keep a running total so that will be part of the display module I believe and then we will be using a comparator chip of some kind to make sure we are validating the money input against the price of items which will be adjustable by dipswitches on the internal components of the board. There will also be a module for the stock of items and the verifying of the costumer's choice against the stock line as I like to call it. If stock line HIGH there are items, if low there are not. So there will need to be some kind of way to verify that and of course no good vending machine was ever built that didn't have a manual coin or money return lever or button that would release the money that was put in the machine if the costumer decided that they didn't want to buy an item. So, in all I am looking at about 5 modules I believe. Display the money in, display the money-out, count the money, verify the money and items, price setting and item stock, and of course the trusty old coin return. Let us now look at some of the modules I have used and see what they are about one at a time and then we will look at the joining of these modules.

MODULES USED

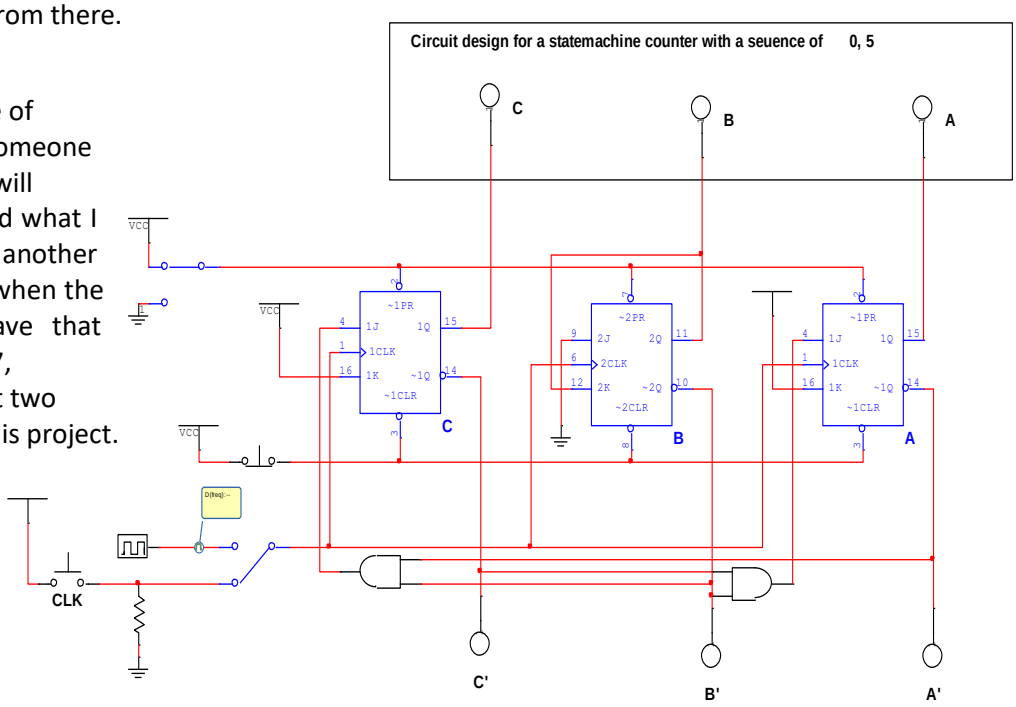
The Display Module



This is the layout of where the money in the machine will be displayed. I had to decide how I was going to make this work properly, that was the fun and tricky part. What do I know, let's start there shall we? I know the machine will take quarters and dollars and ones and fives. That means that the last two will be for the cents that get inserted and if its only quarters that means the last two need to go from 00, to 25, to 50 to 75 and back to zero. That tells me the very last display will only ever have to show two states. A 0 and a 5. So, all I need to do is come up with a way for it to turn on a 5 whenever there is a 1 in the LSB spot. That shouldn't be to bad. Maybe a counter that sends a high to the 0101 on the 7447 that is controlling the penny spot display. I also know that the next one will have to go in a 0, 2, 5, 7, back to 0 pattern and it will have to increment the one next to it by one every time it counts 4. Because four quarters is a dollar of course, we all know that. So if the we have the quarters incrementing the dollars we will need the dollars to increment the tens place which holds its position as the MSB in this system so using these 7447s we know they are counting up to 9 in BCD and then they will need to go back down to one so using just a binary count wont really work out because as soon as we get to 1010 there will be an error on the display. This can be circumvented be building up a circuit that can add 6 to the count which is just saying to go back to zero all while adding one to the more significant bit living next door. There was a great example in out books that worked wonderful in the later design when I decided to finally add it. Of course, I went and tried to do this with some other techniques at first and though they worked in certain circumstances they didn't work in all. This I will explain in more detail as this report gets going on the intricacies of this circuit. So, my first order of business is to make the counters for the quarter spot display and take it from there.

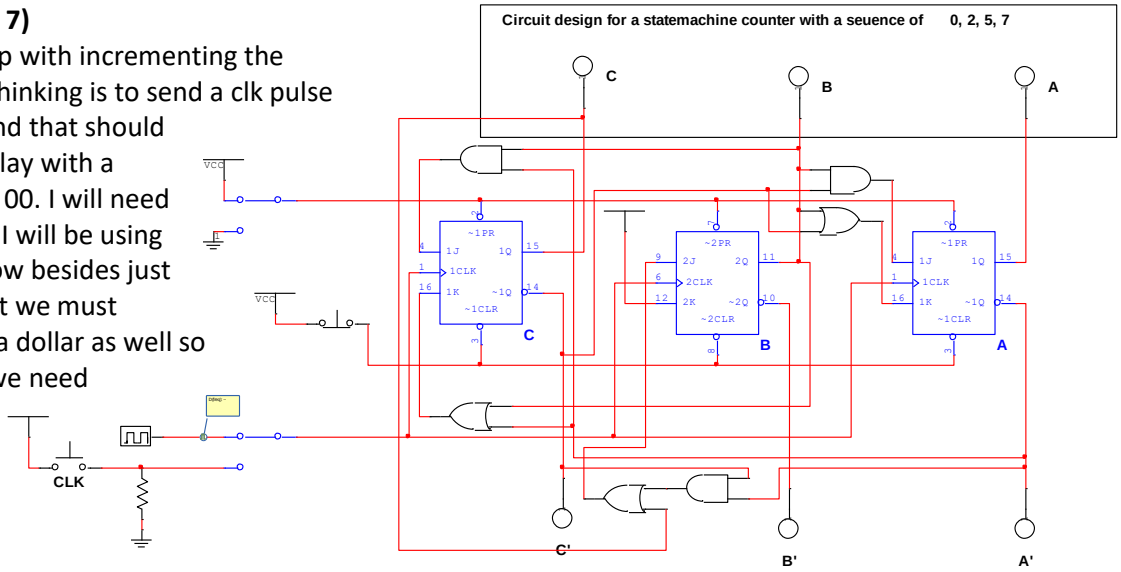
COUNTER MODULE #1 (0, 5)

This counter will count in a sequence of 0, 5. My thinking is that every time someone Hits the quarter button that a pulse will be sent into this and it will then send what I need to the last display so if I hook up another counter that also gets incremented when the quarter button is pushed and have that counter using a sequence of 0, 2, 5, 7, that should essentially move the last two displays in the fashion needed for this project.



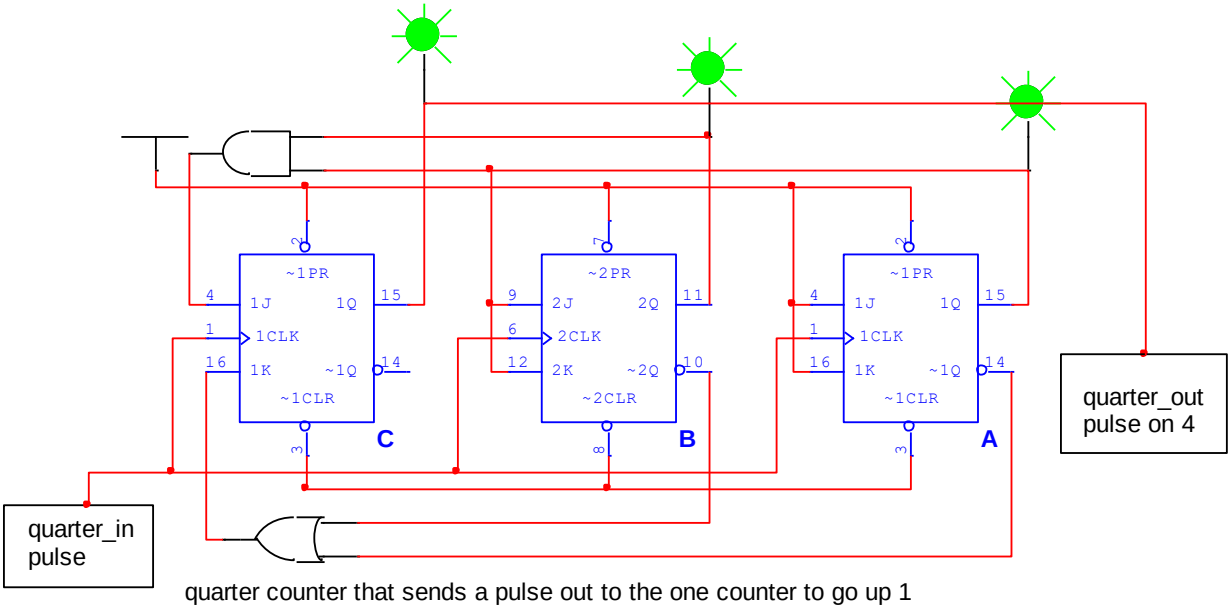
COUNTER MODULE #2 (0, 2, 5, 7)

This is the counter that will help with incrementing the Tens place penny display. My thinking is to send a clk pulse Into this and the 0,5 counter and that should Cycle through the last two display with a Display showing 00, 25, 50, 75, 00. I will need to start considering the means I will be using to increment the ones place now besides just hitting the 1 which you can, but we must remember that 4 quarters are a dollar as well so any time we get four of them we need the dollar to go up one more.



Incrementing the Dollar:

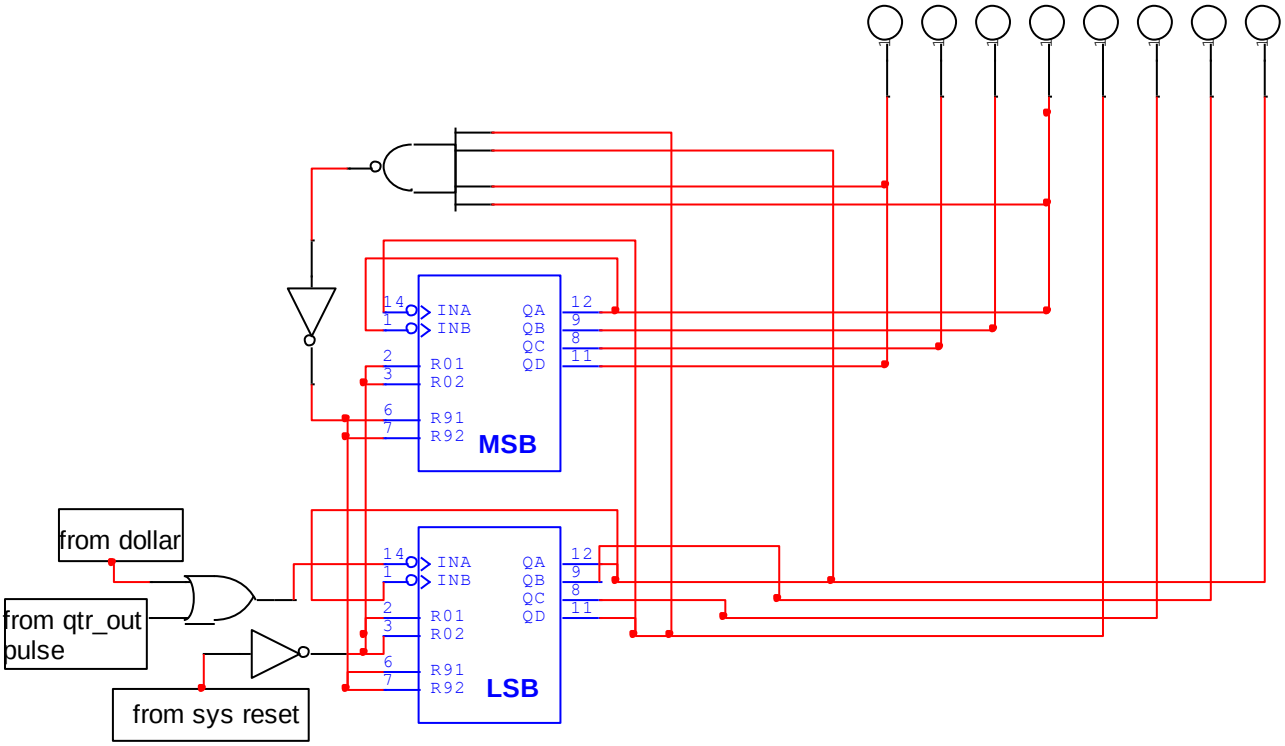
So, my initial thinking was that this should be straight forward with two counters already built both being tied to the same pulse from the quarter button, all I would need to determine is where to grab a signal from the BUS now. So, what I did is make a counter that would count to four and I would take the HIGH when it got to four and send it to the one counter for a 4:1 count. This circuit timer looks something like this:



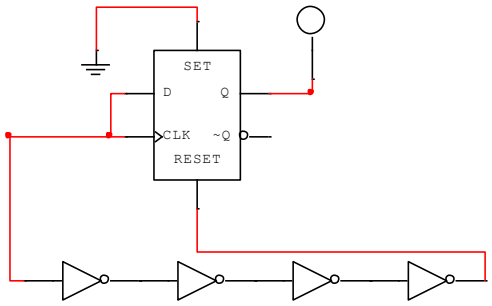
quarter counter that sends a pulse out to the one counter to go up 1

This is the counter I had hooked up to the one counter so that every time that four quarters were added the system would increment the one by one and this in turn was connected to the display that was showing the ones place. To note this 7-segment display had the decimal grounded out so it would show all the time essentially providing a way to see the numbers in money format to two decimal places as would be expected from a money display. I did however run into one issue with this setup. In testing when I would add four quarters to the system the one counter would go up by one like expected but if I didn't add a fifth quarter to the system, I wouldn't be able to add any dollars at all. This was the result of the fact that the quarter counter wasn't being reset or anything it would sit at a four and this was actually holding the like HIGH so when I was trying to add a dollar it wouldn't register at all and that wasn't a good thing now was it. I had to find a way to just send a sing pulse that would shut off and go back low after this way the system wasn't being tied down by unwanted signals. I did come up with a solution to this and it involved adding a little something to this counter so instead of sending a signal out to the BUS itself it would go through a different circuit which would reset itself and give me the desired results. Let us first have a look at what I was using for the dollar counter and then a look at the addition to the above circuit that was used to obtain my accurate count.

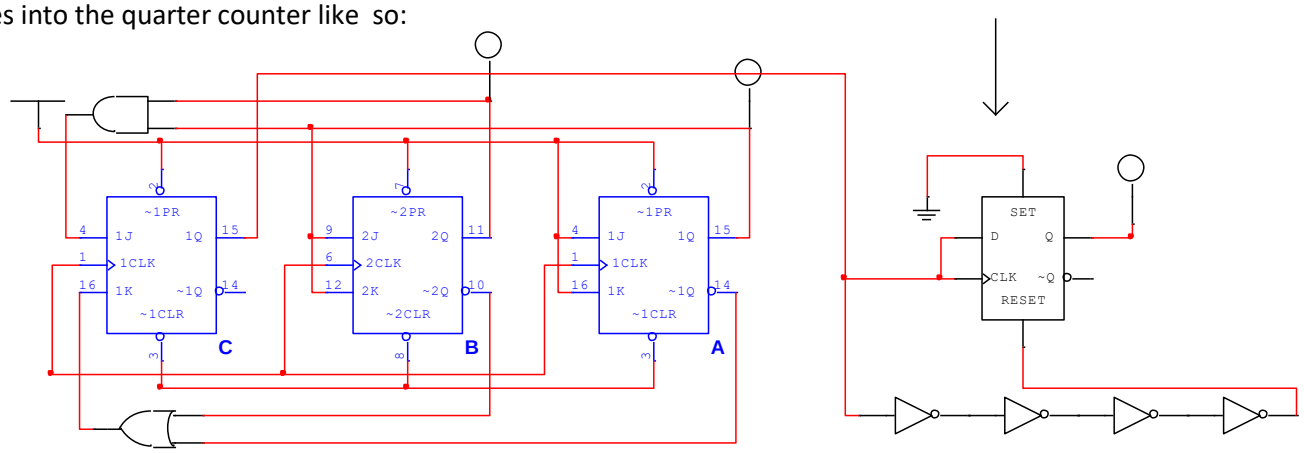
The Dollar Counter



What you are looking at there is a series of decade counters so that when one gets to 9 and turns back over to 0 it will increment the MSB counter to one so for every 10 of the LSB we get a 1 count on the MSB. This is used with the 7-segment displays as they do not use binary count above 1001 anyway so I was hoping by tying these into the displays that everything would work out for me. So far everything was going well as to be expected and this was counting ones in and was adding one with 4 quarters and with the help of this:



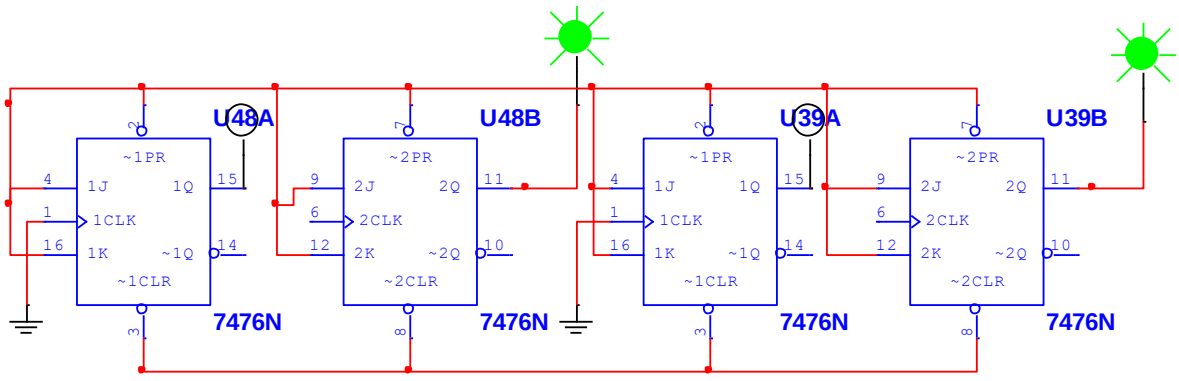
Here is how I was able to keep the line free for the ones after a qtr. four count. This just ties into the quarter counter like so:



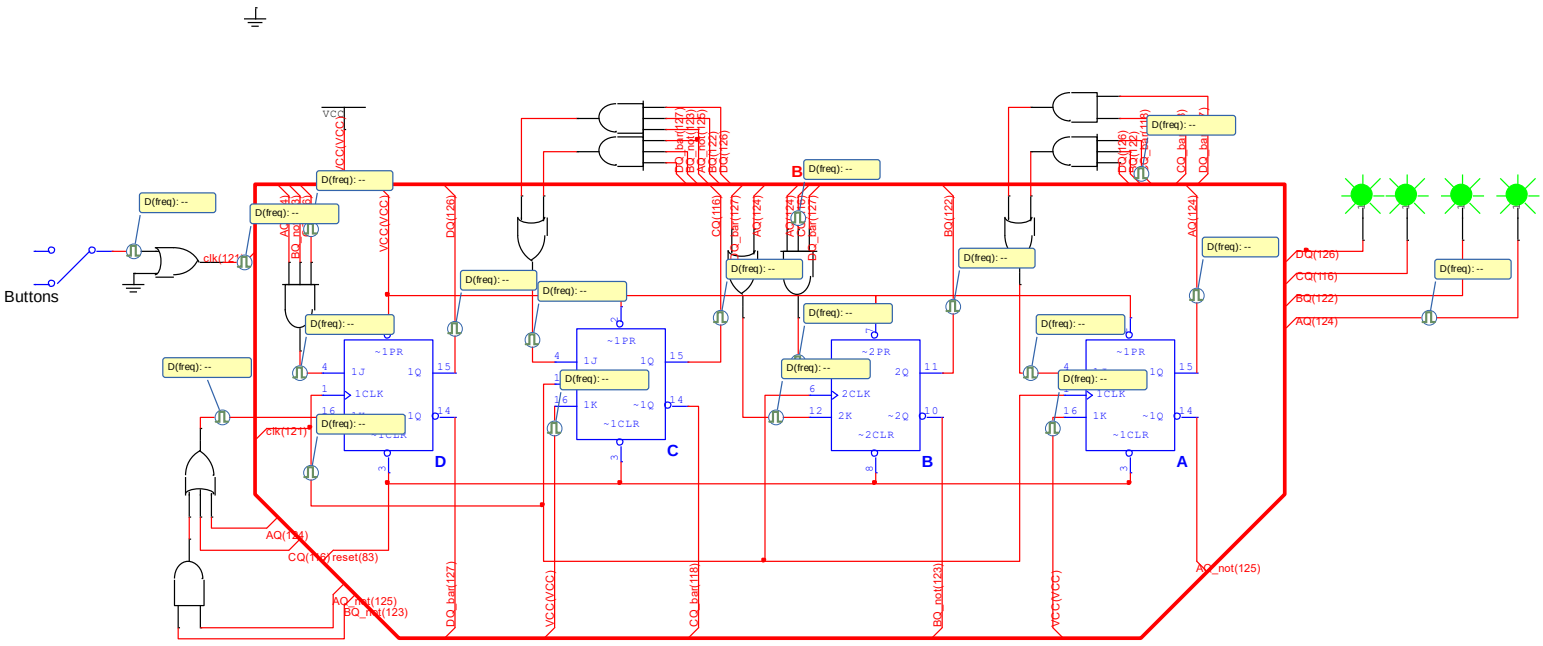
And all that little addition does is take a HIGH signal that is holding HIGH and puts a single pulse out on the BUS, so I don't end up tying up the line with a HIGH that I just really needed for a nanosecond. I ended up using this in a couple different area of my circuit and I was happy with the results of this little doodad. I think I will call it the Buzz Killer, if a HIGH can be represented as a Buzz this little doohickey definitely kills the Buzz. Bad joke I know, sorry. Back to the circuit now. After hooking up these things we should have a working display. We have counters for the quarters, for the dollars and with the two-decade counters if I tie the MSB to the tens place display there should be no problems.

I would like to add that this was NOT my first approach. I had originally built up a ring counter and was trying to use that but with a ring counter on the initial reset it starts as 1000 and then 0100 to 0010, 0001, 0000, back to 1000. And my initial thinking was to grab the signal when it got back around but being reset in that spot was causing me design problems that seemed more challenging to work out then what it was worth, especially knowing there were far simpler solutions then making that work. The problem kept occurring that the ring counter would be off by one and looking back I realize now that it was because the decade counter I was trying to pulse was negative edge triggered and because this was still early in the design process I had not yet developed the Buzz Killer or I could of used that to solve the problem as well but at the time I scrapped the idea of the ring counter and moved to the counter I showed previously in the write up and when I was having problems with this one as well that is when I decided I need to design something that would take a active HIGH and just make a pulse out of it, so yeah that is the history of the Buzz Killer.

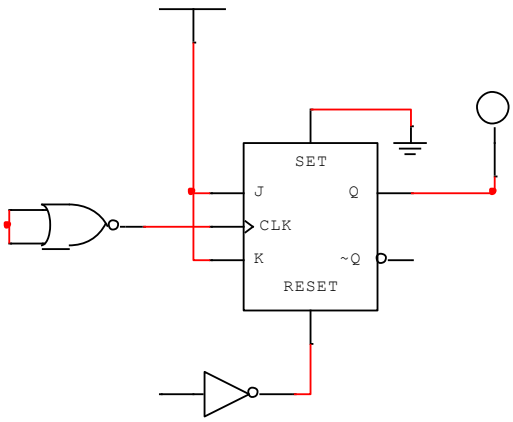
With the quarters and the ones taken care of and working properly it was time to move up to the fives. So what this button would need to do is make the dollar display a five if you hit it once and a zero if you hit it twice and a five again if you hit it a third time so essentially I was needing a 0, 5 from it. Hmmm, didn't I already have a shift counter built up to do just that? I did, now, how could I use this, if possible. This was my original thinking. But what I ended up doing was creating series of four JK flipflops set to parallel load and I also had an output line from each of these locations that went up to a 7483, 4 bit adder with fast carry where it would take in the 0101 and then output the 0101 where I would grab the 0101 with the 7447 for decoding into the proper 5 on the 7 segment display. So that was fine and dandy but how should I get it to go to 10 then if someone puts in two fives? Well time for another shift counter. This time with a sequence of 0000, 0101, 1010, 1111, 0000 etc. if I took the MSB when it went to 10 and sent this up the BUS to the ten spot display and told it to show a one and the five would be gone because the bits feeding the adder would drop to a 0, I should be able to get my 10 up on the displays. Here are a couple of these new counters and PL JK FF's I was building and working with in my design:



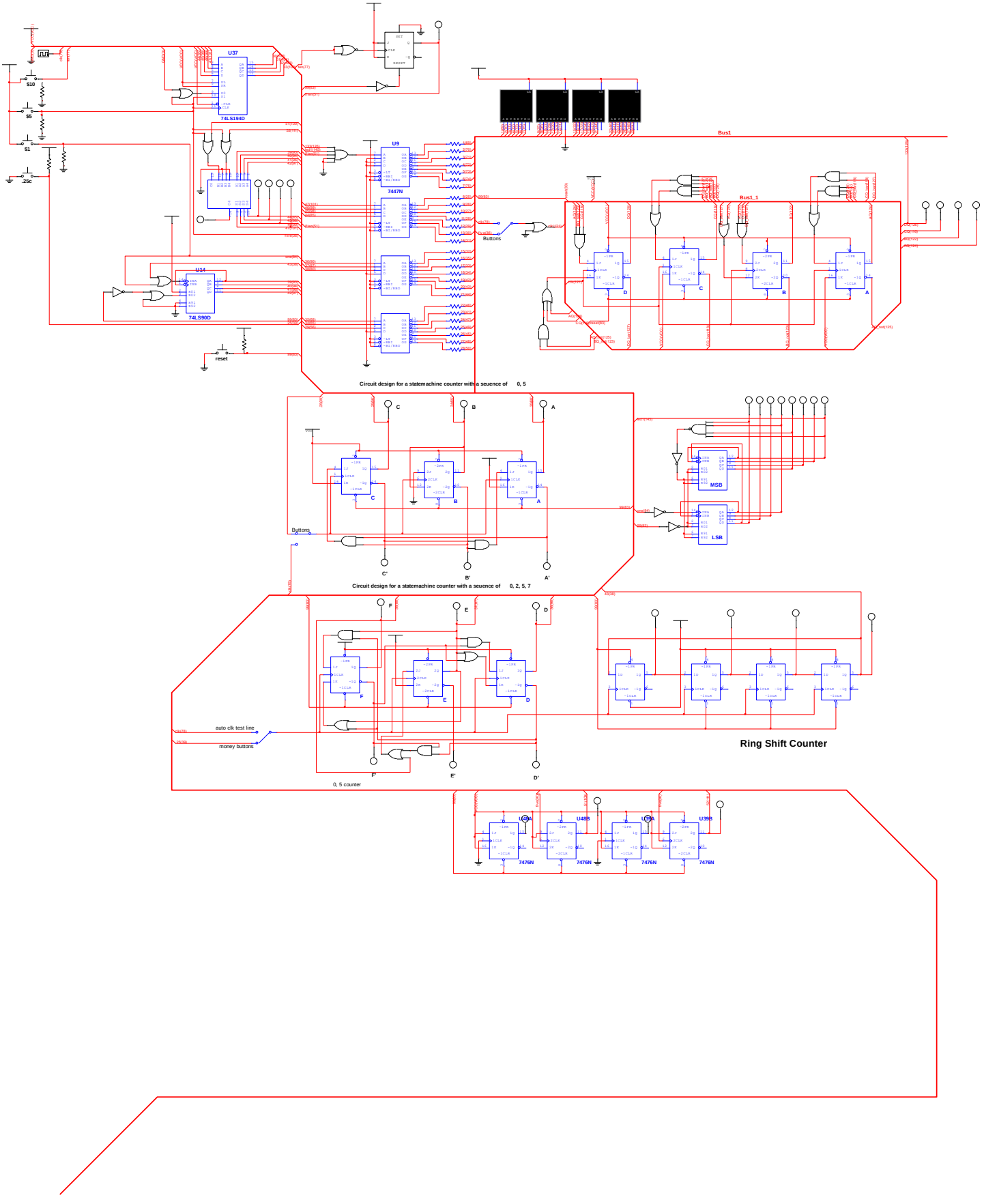
These here are the JK flipflops I was using to add to the 7483 in order to get it to put out a 0101 for decoding by the 7447 into bcd for the 7 segment displays. This is probably not a great design solution and I believe may have led to design complications further down the line. But what I believe is that though not the best this does become a good option for adding some form of system redundancy for making sure a "thing" happened no matter what. The nest circuit is the shift counter counting from in fives from zero up to fifteen.



As well in my early designs I had hooked into the bus a 74194 because I was thinking I may want to parallel load it with data as its being sent in by the money buttons where could rig it to parallel load out to some JKs for transport to the 8 bit twos complement adder / subtractor that I had yet to fully realize in these early preliminary stages but I was always trying to think a bit ahead and scheme up ways to accomplish tasks at hand, and the task at hand was make the display work with those buttons, worry about the math later, right? Wrong as I came to find out real fast, displaying numbers using logic and doing the proper math as I went were something I was thinking more about. Next, I want to show a little page sized example of the schematic thus far in the design process. This is a great example of early on designs and how they evolve along a path depending on the project requirements. One last thing before the full-page view is for the ten button all I did was use a JK FF in toggle so when someone hit the button it would toggle on sending a signal to the display to show a 1 on screen. The JK was set up like this:

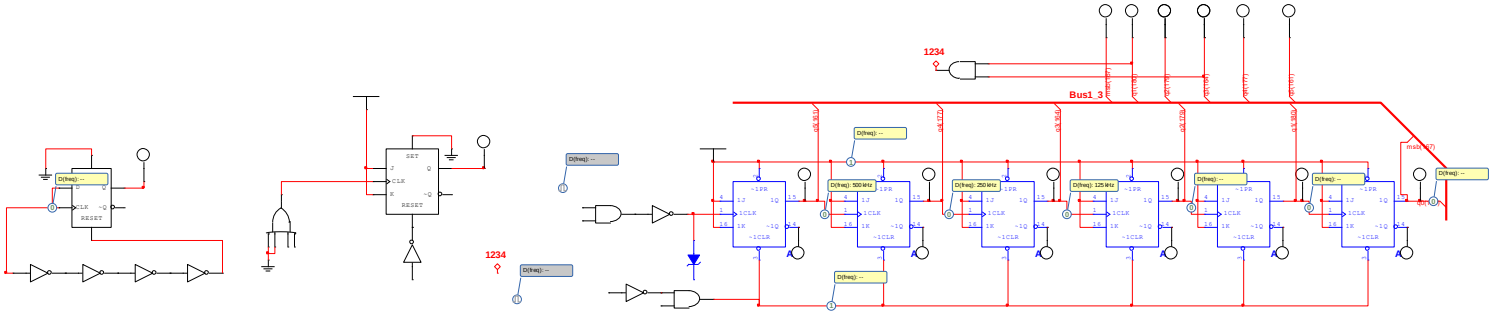


I had the signal coming into an inverter, but it was an NOR gate with the the pulse going to both gate inputs so when it went low it would send a HIGH toggling it to Q=1 Q'=0. This was an original design relic that got Removed in later revisions of the design. I believe I had originally had an active
LOW system which I didn't go with and some of the early modules and ideas I used ended up making their way to the schematic to ultimately be removed when trouble started occurring. It isn't wise to
mix the two either, well maybe with a reset but I don't think
you would want an active low ten button and an active high one and five and then an active float quarter button, kidding. Unless, Hmm!??
Can you have an active float so when you dropped into a float state you Trigged an event or sequence of some sort?
Things to ponder later, I guess. Well without any further ado here is one of the Earliest designs I had up and running for the display and money button module

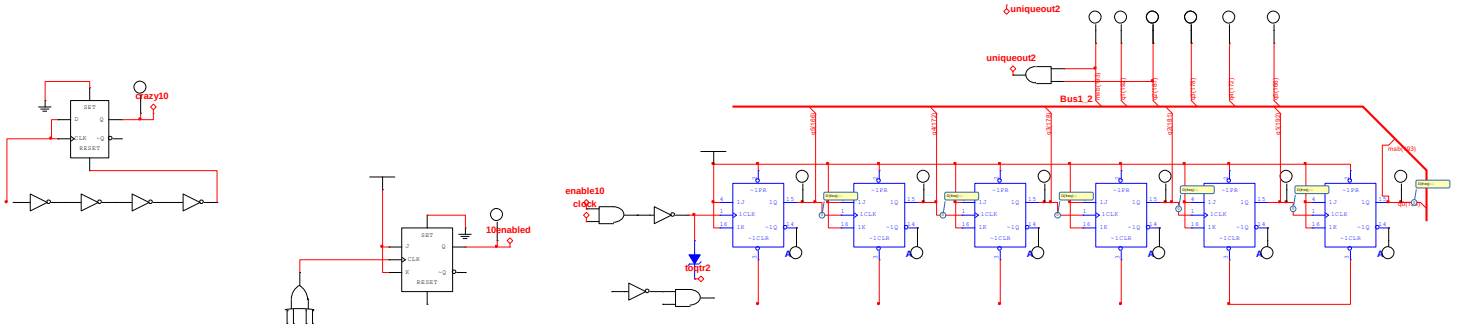

Bus1
Bus1_1
Buttons
reset
Circuit design for a statemachine counter with a seuence of 0, 5
Circuit design for a statemachine counter with a seuence of 0, 2, 5, 7
MSB
LSB
auto clk test line
money buttons
0, 5 counter
Ring Shift Counter

A Project Evolves, in comes the MATH of it all.

With the display working like it should when someone hits one of the buttons, for the most part anyway there were a couple issues with it when someone put in all quarters up to ten it didn't change over like it should or if you added a five and then some quarters and ones up over 1010 again and things got screwy I guess you could say. I knew from the tips and from our fabulous leader John Osthoff that the solution lay on page 290 of our text books in the way of adding 0110 to the 10 to make it go back to zero, or one or any of the acceptable BCD digits. But I wasn't exactly there yet in my design even though I should have already had that worked in it took me a few designs before fully implementing this into the design. My first order of business, only because I knew knowing an exact count of quarters was essential in the doing of the math for the return change, or at least I thought that at the time when really all I needed was the binary representation of these and not an actual counting down or up like my design does, it counts. So, for the dollar that is actually 0010 quarters and a quarter is of course 0001 and a five would be 20 quarters being represented in binary as 10100. So that would make the ten-dollar button represent 40 quarters or in my mind 40 clock pulses. How could I make one clock pulse turn into 4 or 20 or 40? With a clock enable using a toggle JK flipflop rigged to a shift counter counting up to whatever I need by running an and gate coming from the bits I need it to stop at I can send that signal back into the JK to turn it back off. If I just grab the clock pulses as they are sent into the counter and use that to drive another counter that doesn't reset and keeps a running total of all the timers outputs added together I would have my total quarter count in binary. I needed to build more counters. So, let me show you each of these counters and just to note after all has been said and done, I could have shaved one FF off of the five counter. Not sure why I got the sixth bit in there that is never used because it always resets after getting to 20. Speaking of the five counter here it is for you in all its glory:

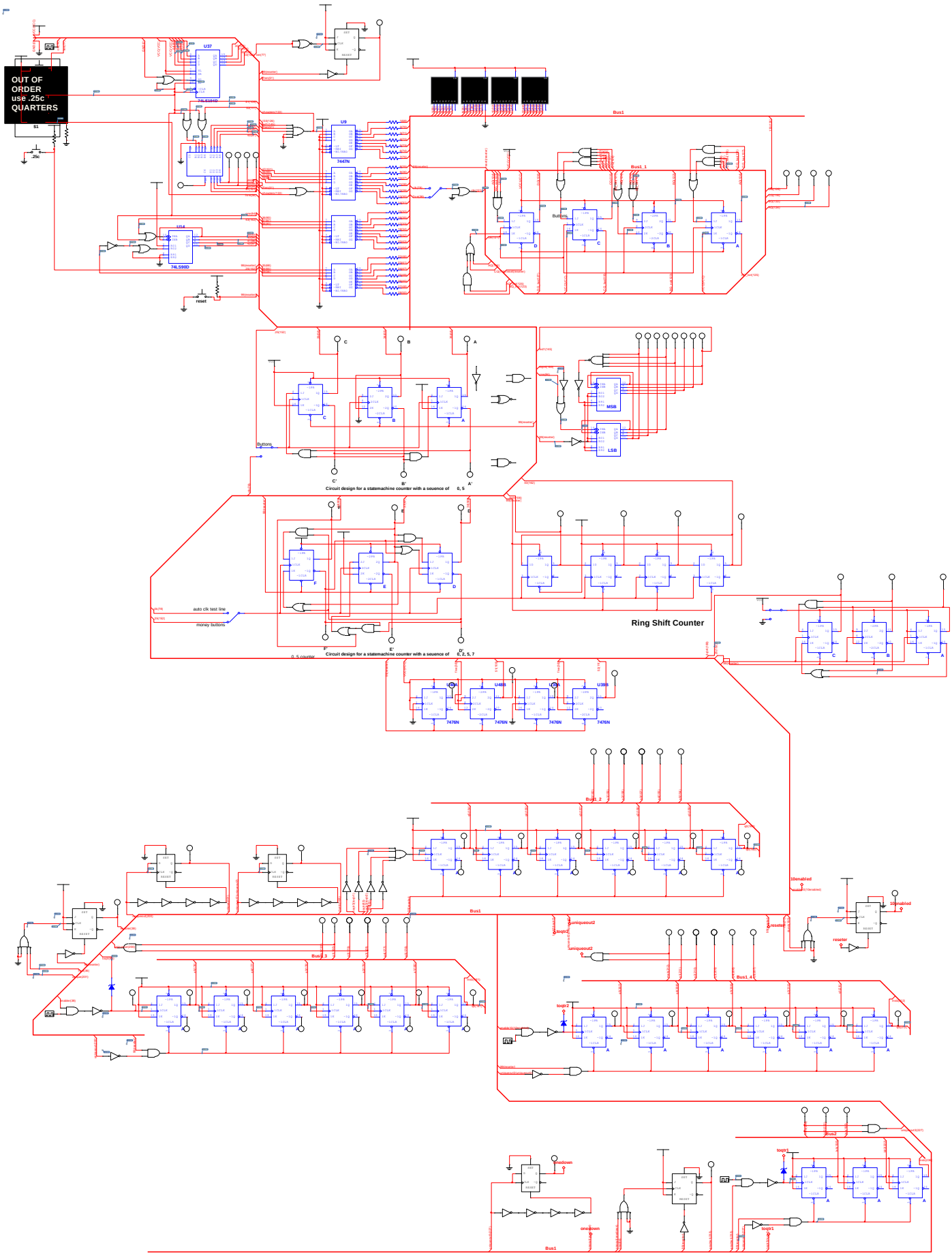


And next let's see the ten counter:

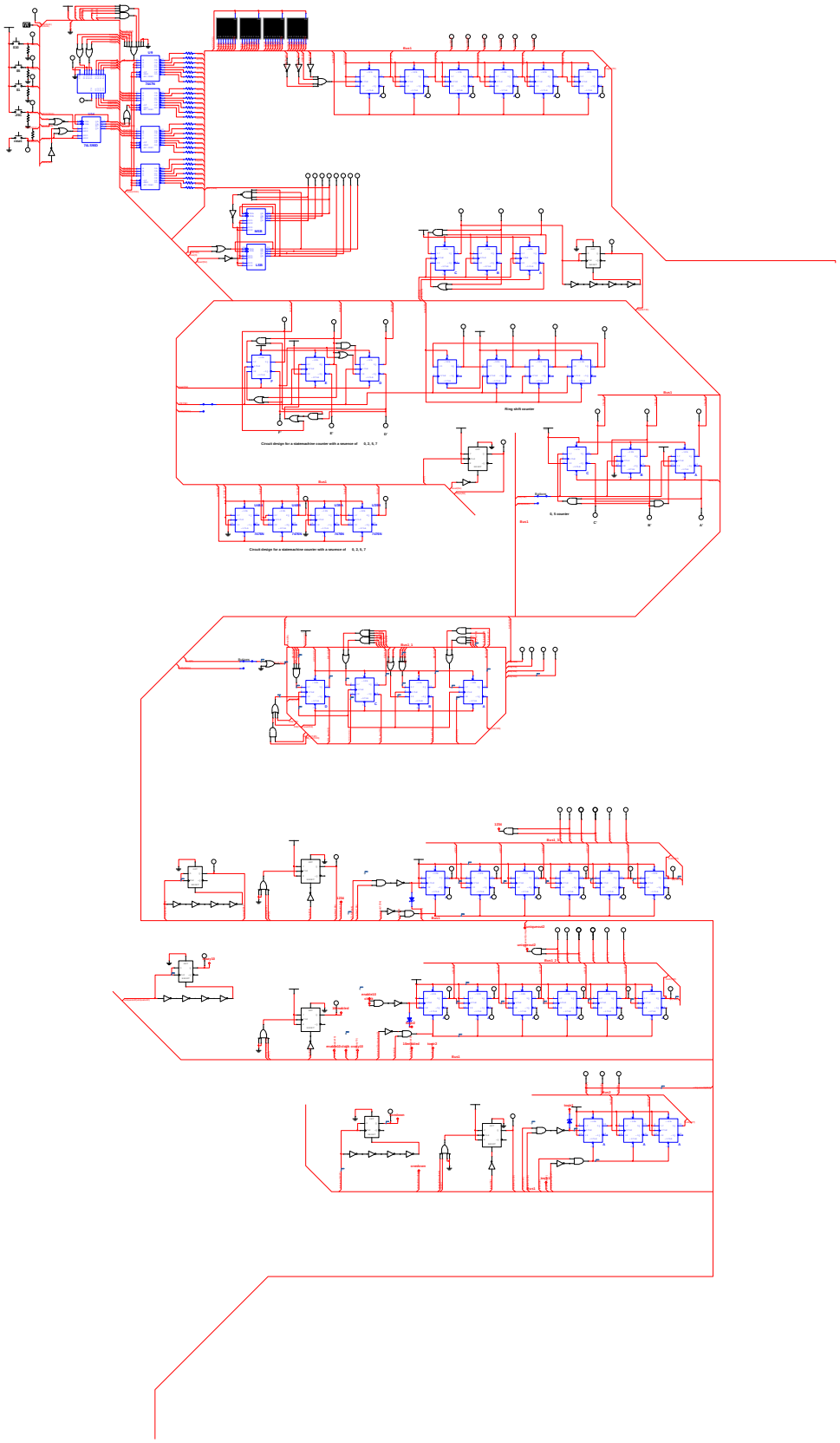


Yup you guessed it, they are the same counter just set to stop at different points in the count. One counts to 20 the other counts to 40. Which in simulation time is like 5 minutes. I imagine that in a real circuit that these times wouldn't barely be noticeable and would be fully acceptable in a vending machine type system. Maybe not so in something like a aircraft or military grade circuit but for you average 9 to5 40 hour a week circuit it should fully do the trick.

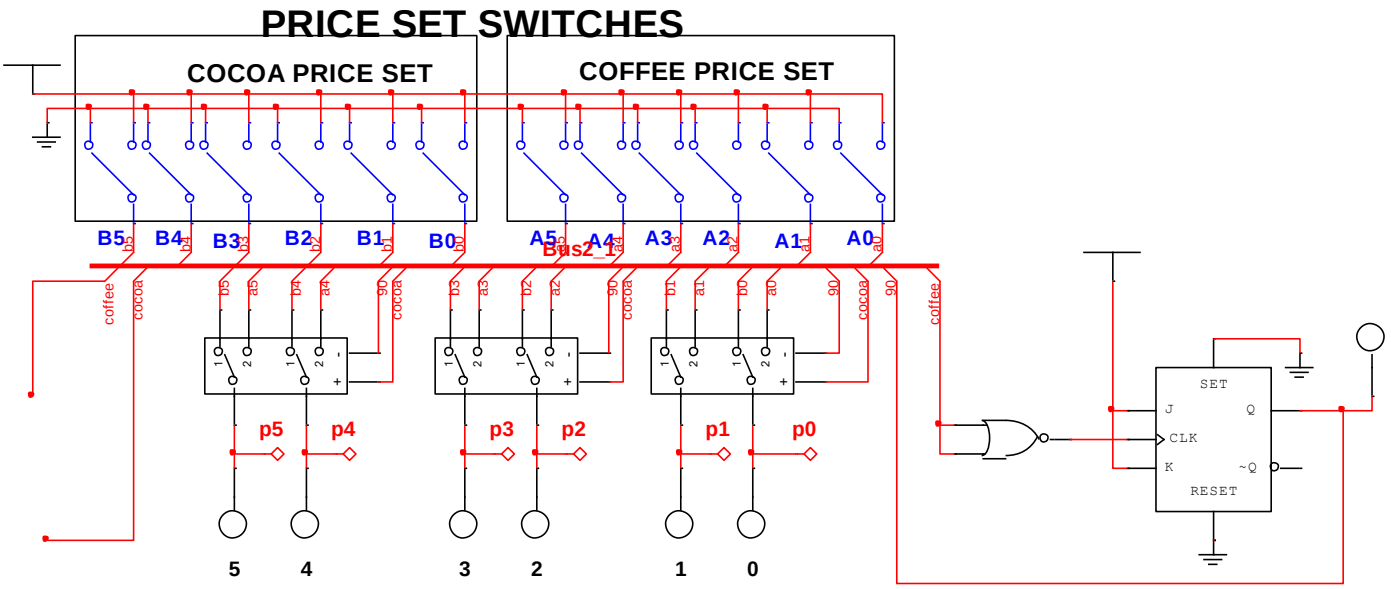
With all these hooked in and counting I just needed one more counter right? The one to keep the total of all these counters. So I just used one more of these except it didn't have a stopper on it so to speak it would just keep counting until its bits got full and then would restart all over again. The next page is a view of the schematic with all these counters hooked in.


OUT OF ORDER use .25c QUARTERS
$1
.25c
U37
74LS154D
U9
7447N
U14
74LS90D
reset
Bus1
Bus1_1
Buttons
A
B
C
D
MSB
LSB
Circuit design for a statemachine counter with a seuence of 0, 5
C'
B'
A'
Circuit design for a statemachine counter with a seuence of 0, 2, 5, 7
E
F
D
E'
F'
D'
auto clk test line
money buttons
0, 5 counter
Ring Shift Counter
U4B
U4B3
U3B
U3B3
7476N
Bus1_2
Bus1_3
Bus1_4
10enabled
resetter
touhr2
touhr1
uniqueout2
Bus2
oneup
onedown

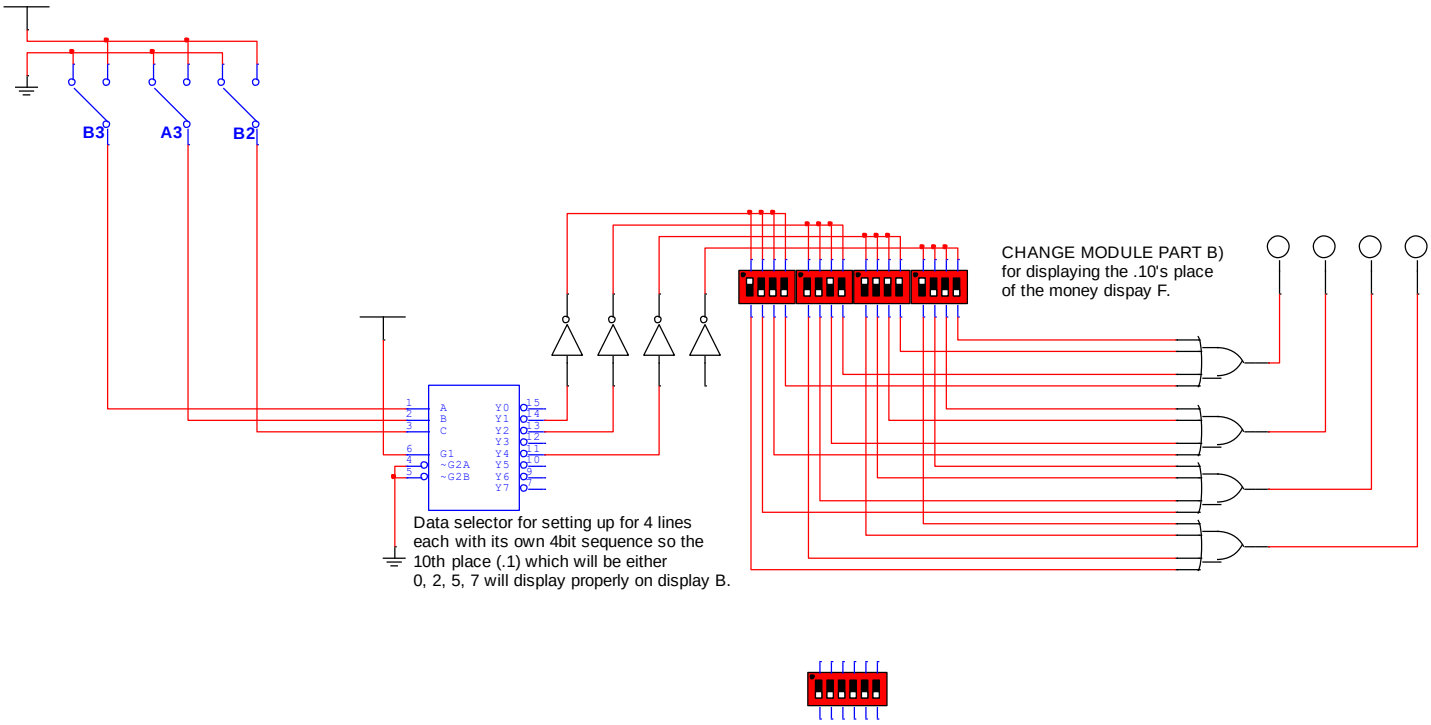
And then I didn't really like the way it was all set up and kind of cluttered so I decided to move some things around a little bit. I thought it would be nice to see the total count up top with the buttons and controls, or where the controls were going to be anyway. So this is a view of the circuit with them changes made. As you can see this has everything discussed thus far implemented in the design and it does seem to work mostly with a few things needing to be worked out but it is a good base for me to start building around. What I needed to come up with now was and inventory system which is just switches that are on or off sending a signal that there is a product or there isn't a product. As well we need an way to set the prices for the items in the inventory and we also need a way to determine the change from a successful vend of a product and let us not forget the good old coin or deposit return in case someone decides they don't want something after all. So there is still a fair amount of work ahead of me here and I got some ideas rolling in my brain so let me share those with you now. For the match all I need is a 6bit 2's compliment subtractor/adder that takes as one input the bits from the product selection line and then compares that to the money in the machine which will be sitting on the quarter counter I built up. So, drop them bits to be compared and I can use this for verifying a good vend, because we don't want someone putting a dollar in and getting a two-dollar coffee. That means we need a way to make sure that the money in is equal to or greater than the item cost. For this a couple magnitude comparators should do the trick and set them up to check more then four bits by chaining two of them together. So, the next few things we will look at are the price setting, in order to do any comparing we need a price to compare with and we need a way to do this. I had went through a couple different ways of doing this and the first way was using a couple voltage toggle switch doodads but this would always have a default price on the line which was ok but I knew I could do better than that and ultimately did. The final design can set the price on up to five different items if really need be. With one being a default to zero for the reset but let us work there slowly.



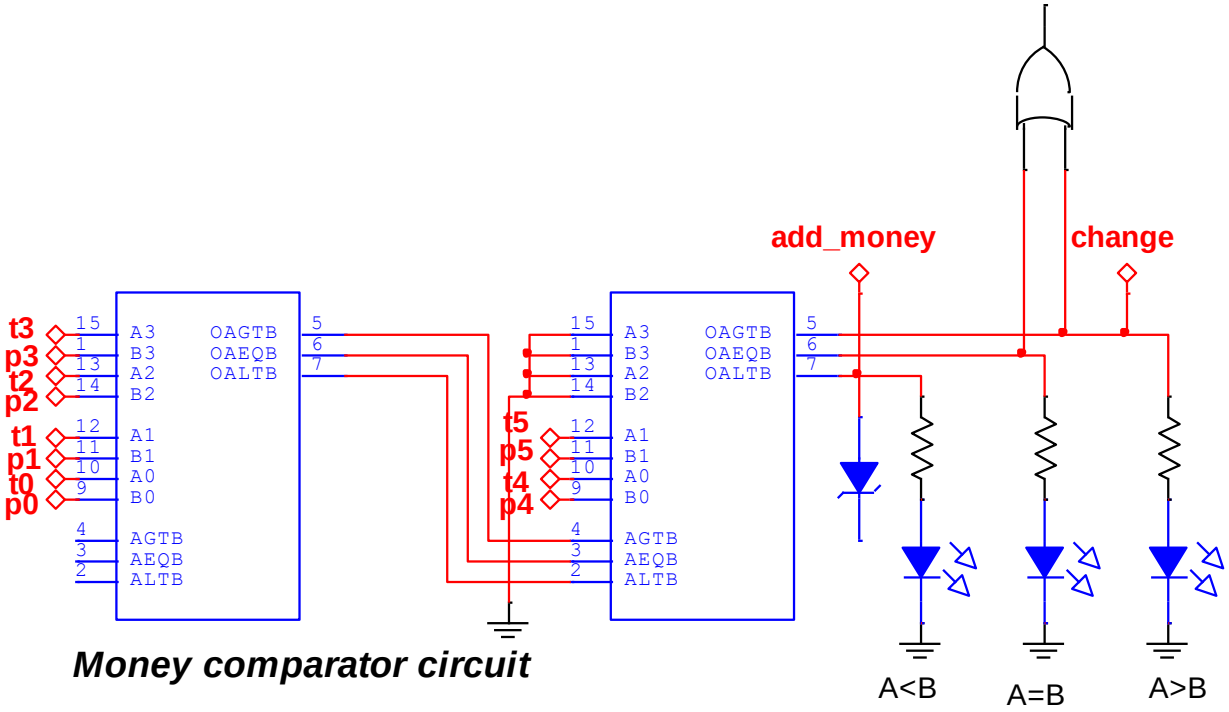
PRICE SETTING design #1



This design was good and actually worked ok but as I was researching and trying to find a better way, I ultimately landed on 4bit DIIP Switches which ultimately led to the 6 bit DIP Switches that are currently in use in the design. Here are the 4bit ones:



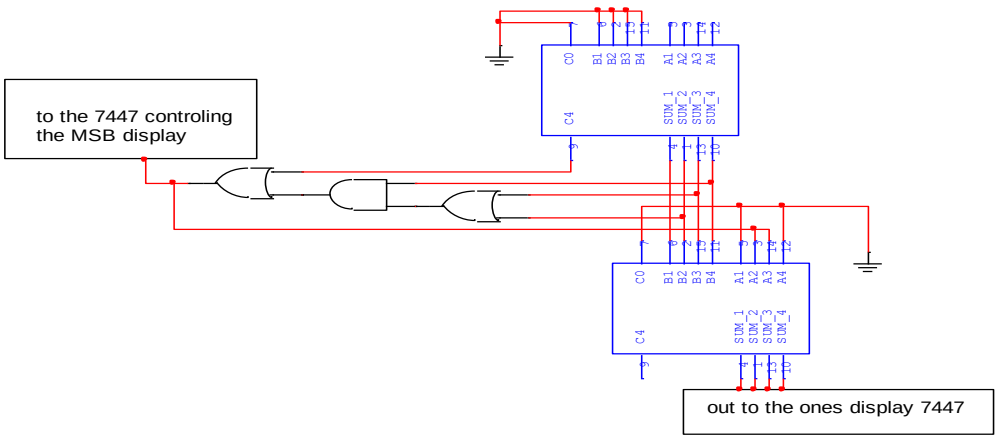
So these worked out wonderful for my design and I ultimately used this means as well for setting the sequence in the quarter's counter display on the return money output. All I have it doing for that is taking as input the A and B spots and nothing in the C so that gives a possible 4 different outcomes which I use to drive the 0, 2, 5, and 7. As well as this same design turned to six bit for the price setting on the final schematic design. So, after having a means of selecting the prices I needed a way to compare the two things, money in to item cost. With this was the use of the two 7485 four bit magnitude comparators which linked together can now compare two 8bit string and for this project the most we needed was six bits so we didn't need to use the last two bits so I just tied them to ground for safe keeping. Here is a look at the comparator circuit now



Money comparator circuit

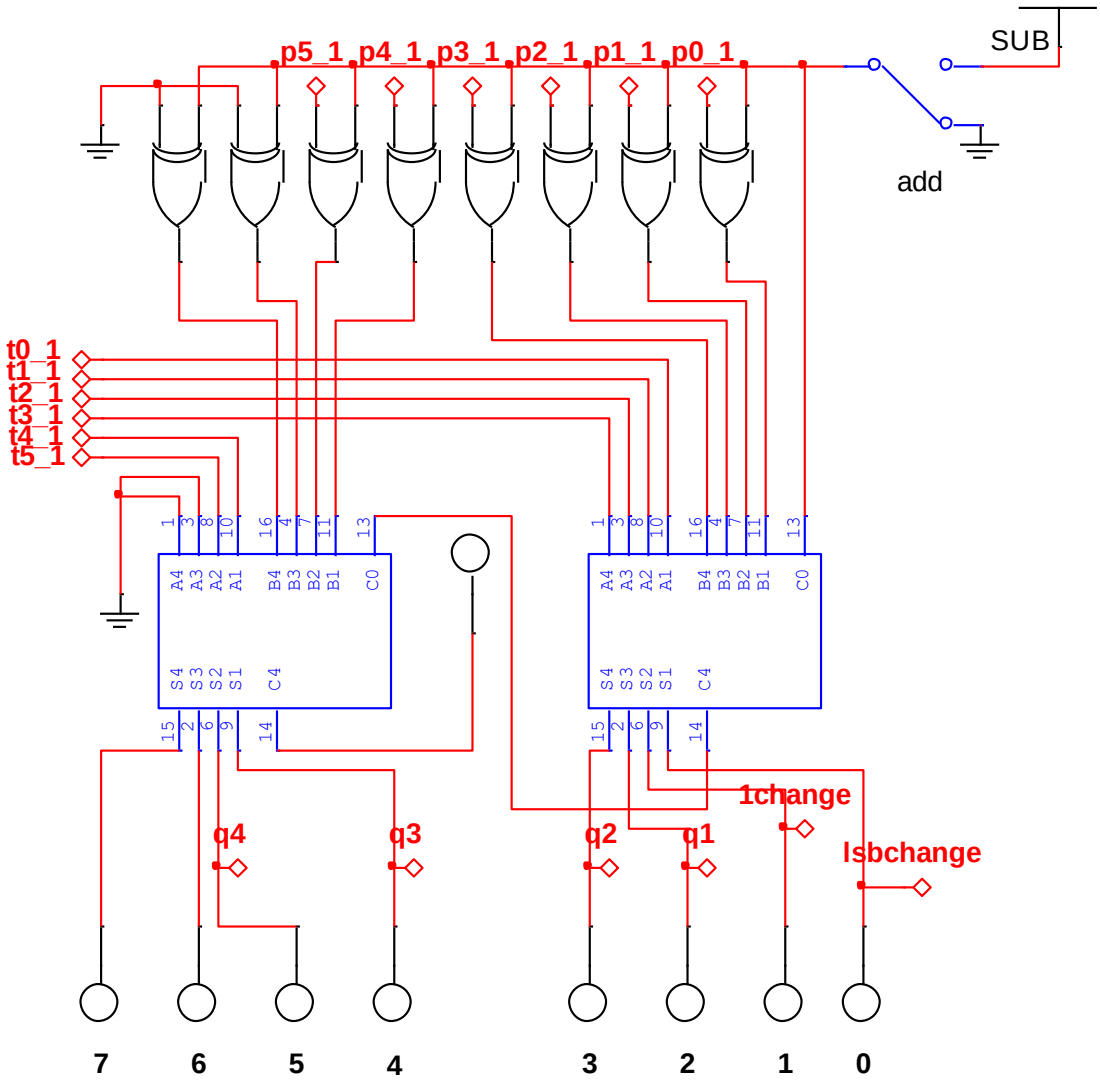
What that OR gate goes up to that you are not seeing here is a line to the BUS named vend_ok. This is one of the required signals used for verifying a successful vend. The other two lines you are seeing are the one used for activating the change dispenser and the other used to tell the customer to add money if they hadn't added enough yet after making their selection. This one thing is why I decided against using a money setting scheme that had a default price on the line is the fact that is would always start off by saying add more money to the machine. I didn't particularly like that, so I had to find a better way. So now it will only tell you to add more money after a selection is made if enough money hasn't been added yet.

After we do out comparing, we need to do some math and for that we are going to need another kind of IC. I ended up using two 7483's which are four-bit full adders with fast carry. I had already used one of these earlier in my design so figured I would stay with like chips in my circuit if possible. My final design uses 7 of these chips in it. The reason for that is because when I built up the circuit to add six to the BCD counters they use two of the 7483's each as well and I needed them for both displays so that is why it got up to seven. I originally thought I would not need to use them because the way I was pulling data from the counters at certain times I really thought I had covered my bases and would get proper display after a 1001 but I was sadly mistaken and in some of my later testing there was bugs popping up that the only real solution was to add 6. Or maybe not the only but was by far the industry standard way to achieve what I wanted to achieve so I had decided I better add these to the design as well and I was starting to think I needed to absolutely incorporated it into the change display at point zero instead of waiting till the very end to try and add it. That circuit looks something like this actually:



Then we have the circuit that does the math. This was the two 7483 like I was discussing which are the same chip I just showed that added 6 to the binary number to correct the BCD number. It is just set up a little different that's all and its rigged so it can either do adding or do subtraction. In this circuit we need it to subtract though I left an internal switch the owner could make it add if they wanted to collect more money, someone buys a coffee and all of a sudden they owe you another 1.75, what is going on. Sorry another bad joke. What documentation wouldn't be complete without a few bad jokes, right.

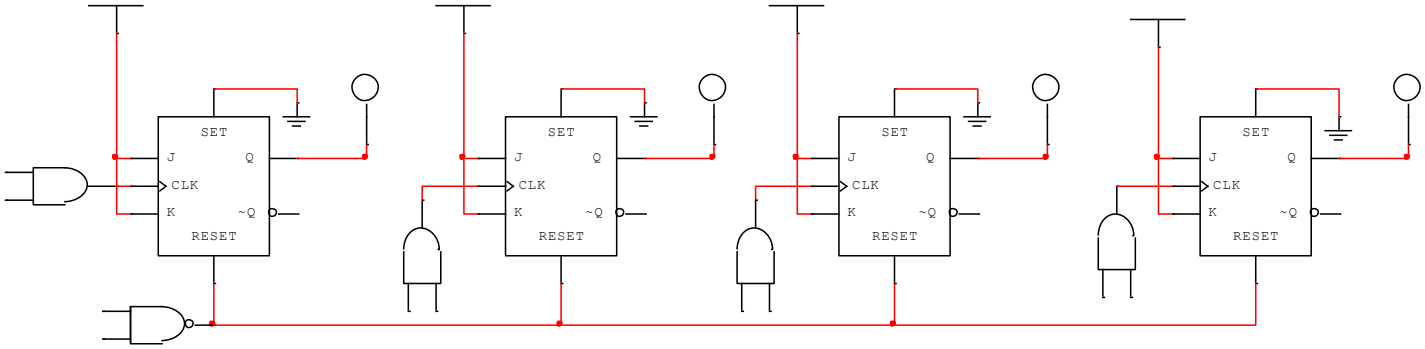
The 8-bit 2's complement subtractor



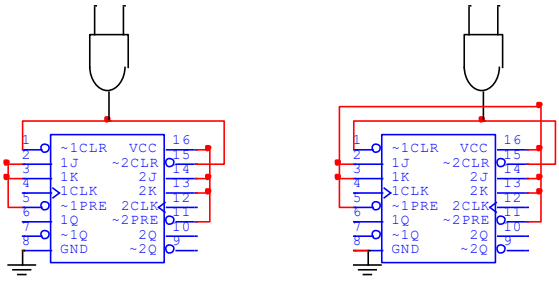
adder/subtractor that is used to display the change due

This is connected to the price selection bits and the quarter counting bits so when there is more money then the item it will activate and display the change when there is a successful vend.

On that note how do we verify that there is a successful vend you ask? Well I built up a circuit that has an 2-input AND gate running to the clk and in the and gate is one line from the supply sensor and another from the item selected signal so in order for it to toggle HIGH there needs to be both conditions met and if they are met then we can say ok to the vend line when it is asking if this is an ok vend. The circuit for verifying supply looks something like this:

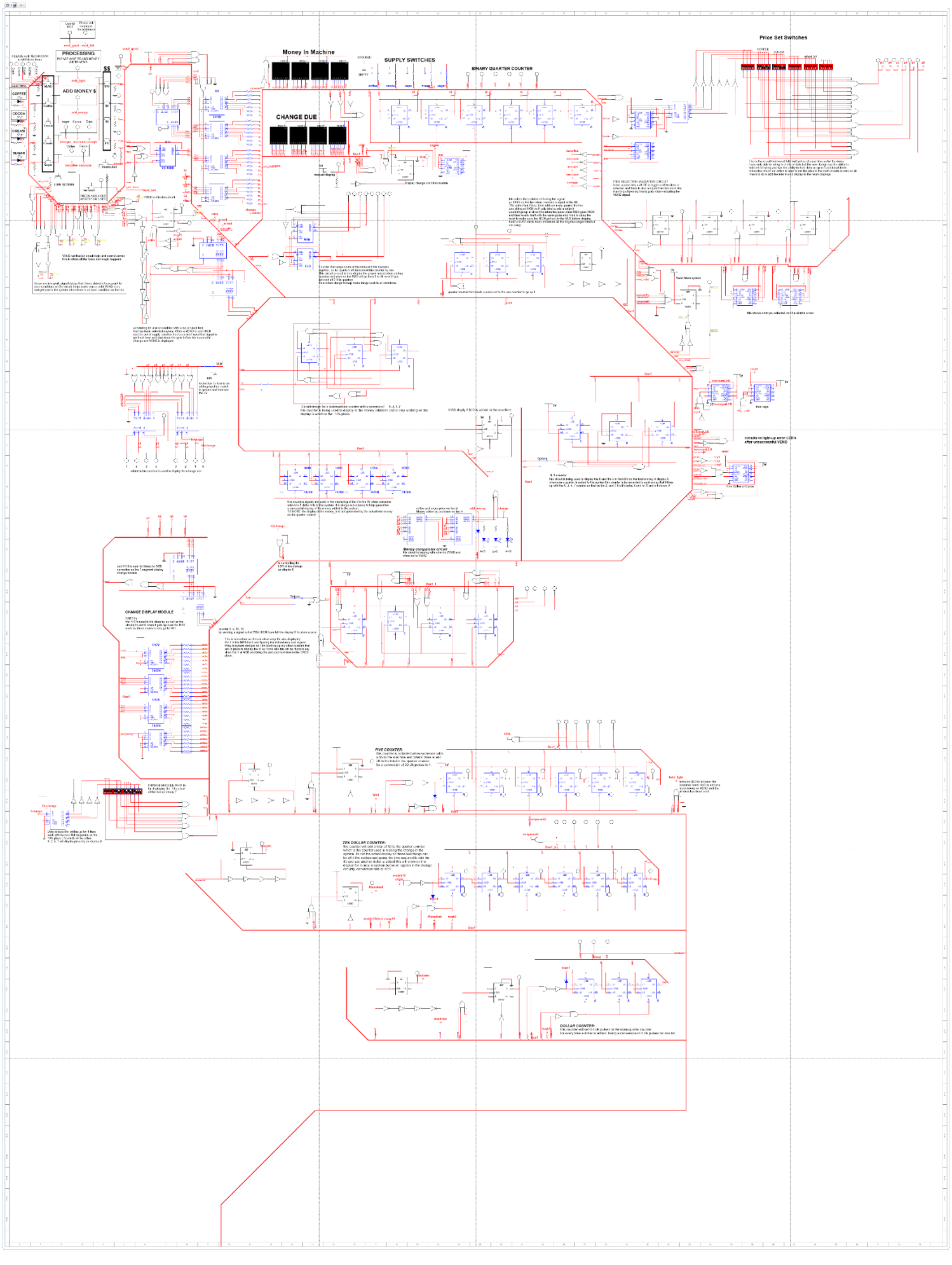


In testing for setting up a stock resupply reset so that you didn't need to re-add money every time items were added back to the stock line. All you would have to do is re-enter the items you wanted to order. I also set up a bunch of JK FF IC's to control the display LEDS and Sensor information they are smaller and use much less space setting up and I probably could of used them all over this design to make it a bit simpler looking though you wouldn't know at first glance what you were looking at. These little chips look like this



Cute little suckers if you ask me. And I used a bunch of these in the design to turn on lights and switch displays and to do other various task that I would consider just busy work to make things more understandable, like why didn't I get my coffee with cream and sugar? Oh because that light there says that it is out of creamer oh and look there, there was already a light displayed as soon as the sensors no longer detect a stock supply but we cannot count on a customer to look to these lights before ordering every time so there needed to be some way to tell them after an unsuccessful vend what was up. There is also a JK FF set up in toggle with an 8-input or gate that has all the possible vend conditions running to it so if it is a successful vend it will toggle and then there is another with the unsuccessful conditions that set off the NO VEND condition.

That is the circuit, I am sure that I have missed a few minor details but these are the circuit modules that make this circuit work and I think this project was by far some of the most fun and stress that I've willingly put myself through. I am a perfectionist which isn't always a good thing, especially when you are running against a clock, obvious race condition there, one that if I don't flex a bit on my standards will surely lose. This design is fully functioning and meets every design requirement as stated in the project outline. I believe if I had just a bit more time I could have done even better, possibly eliminating some unneeded circuity and increased simulation speeds. To note on researching simulation times in multisim there is no way to simulate in real world timing like I would have liked due to all the mathematical engines calculating underneath it all there is a heavy toll on the processing power. I learned there is a time scale in the lower right hand corner that is measuring the time the circuit would of taken to run in the real world had it been a real circuit and not a simulation. This is measured in milliseconds and it takes about a min of actual time in the simulation to simulate what would have taken an actual circuit about a millisecond. That I found to be pretty useful and interesting. The last page of this documentation is a view of all these pieces put together in the working circuit. I have also included a 3x4 printout of the circuit blueprint for you to keep for future reference should you ever feel the need. Thank you John and see you in c++ II.


Money In Machine
SUPPLY SWITCHES
BINARY QUARTER COUNTER
Price Set Switches
COFFEE
COCOA
CREAM
SUGAR
PROCESSING
PLEASE WAIT WHILST MONEY ORDER VERIFIED
ADD MONEY $
CHANGE DUE
CHANGE DISPLAY MODULE
FIVE COUNTER:
TEN DOLLAR COUNTER:
DOLLAR COUNTER:
Money comparator circuit
circuits to light-up error LED's after unsuccessful VEND