Digital Design - Assignment Report: Group 101

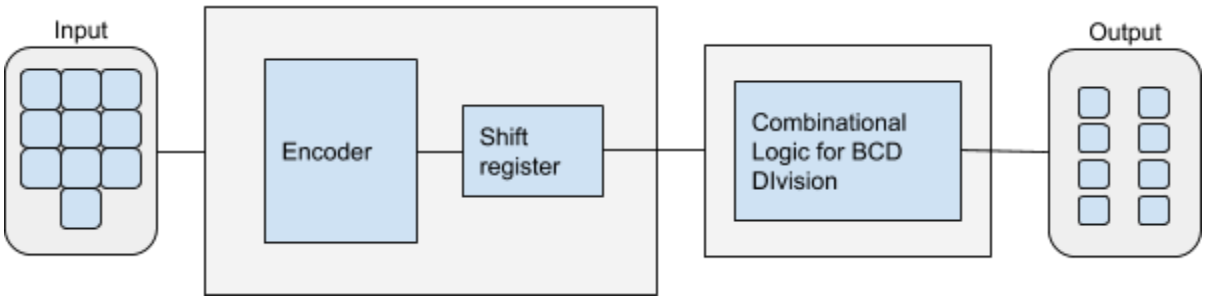
Group Members: (ID no. order)

1. Kushal Joseph Vallamkatt 2019A7PS0135G
2. Aryan Ashish Tyagi 2019A7PS0136G
3. Shrey Nandlal Pandit 2019A7PS0138G - Group Representative
4. Rachit Jain 2019A7PS0140G
5. Priyansh Mehta 2019A7PS0142G
6. Apurv Amar Botle 2019A7PS0143G

Problem Statement:

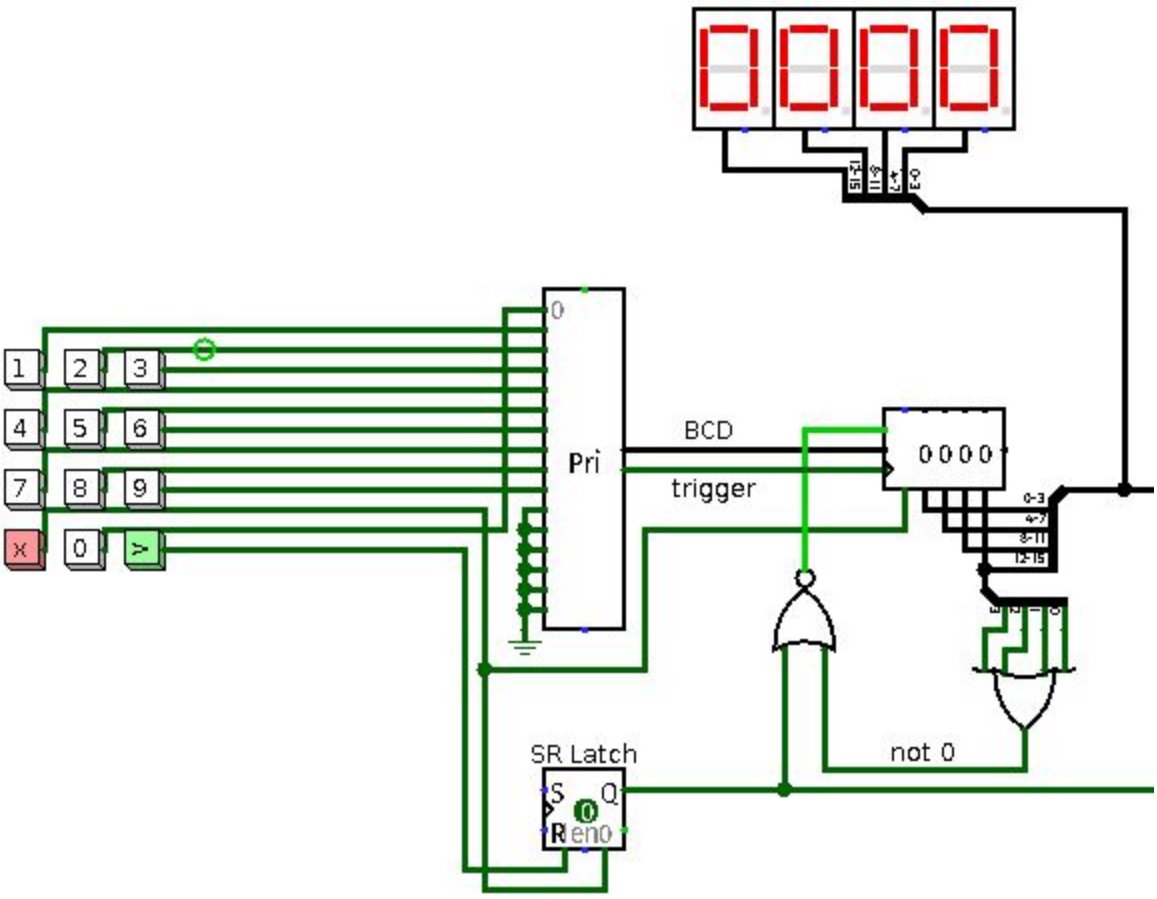
Design a cash change machine that takes an input in Rupees for any value between 1 and 1000. Only whole numbers are allowed. The machine will then dispense change using Rs. 100, Rs 50, Rs. 10 and 1 rupee coins only. You can design the system so that the maximum number of 100 and 50 rupees notes are used.

Top Level Block Diagram



Brief Circuit Explanation:

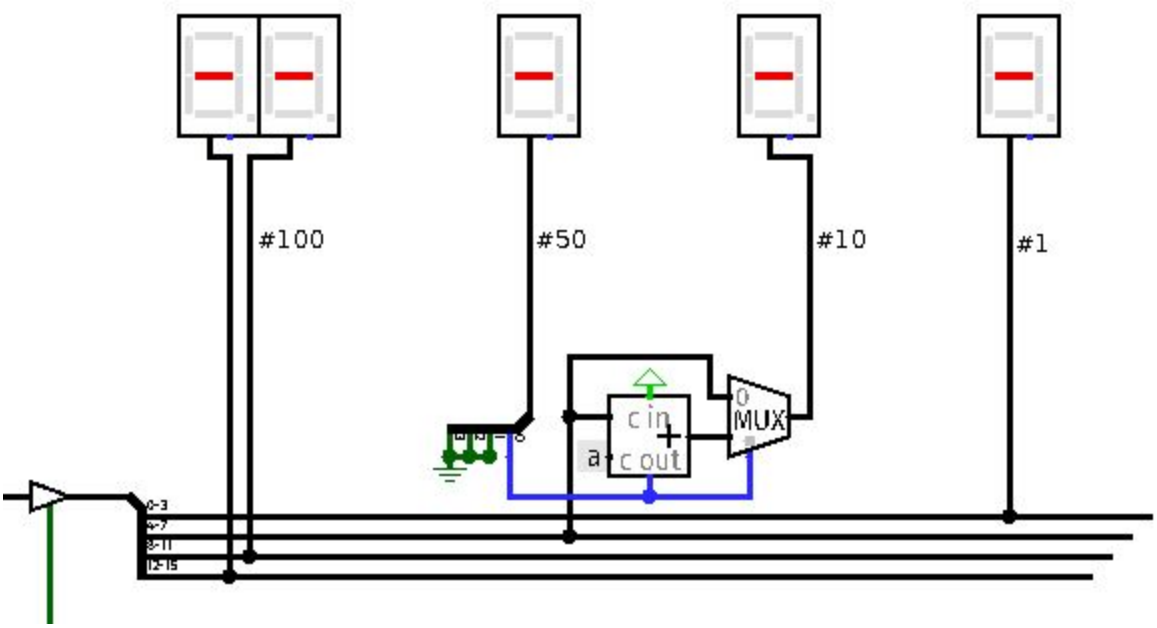
1. Obtaining and showing the user's input:



- The user's input is obtained via selection buttons, which are connected to a 16x4 Priority encoder. The buttons 0-9 are connected to 10 of the first ports and the rest are grounded. This would lead to a 4-bit output which is the BCD representation of the selected number.

- This BCD output is fed into a shift register, whose clock is triggered by at least one of the encoder's inputs being High. In this way the user may enter 4 digits which are stored into the shift register
- After 4 digits are already entered, the last BCD digit entered must obviously have at least one High (since it would not be 0), hence the next OR gate changes its output to 1. This in turn, makes the next NOR output to 0, deactivating the shift register for the time being. (refer diagram)
- Confirming input: The user clicks on the ">" button, which sets the latch, which in turn makes the NOR gate output to the shift register 0, hence the user may not enter any more digits. If and only if the input is confirmed, and hence the latch is set, the input moves on through a controlled buffer into the next part of the circuit.
- As the numbers are entered, they are stored into the hex-displays for the user to view their input.
- Resetting input: On clicking the "x" button, the Latch and the shift registers are reset, hence the input shown in the hex displays also goes to 0000.

2. Logic for the problem statement:

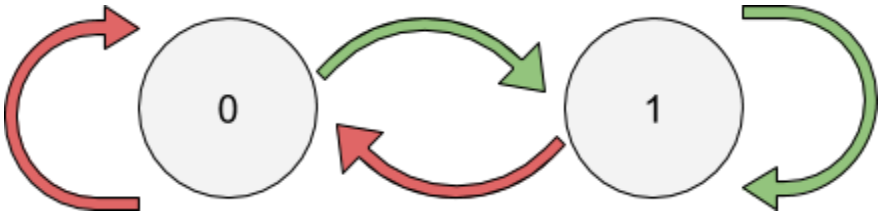


- The logic for the circuit is built using simple logic components like Multiplexer, adder.
- The number of 100s is simply the number obtained in the first 2 digits of the user's input.
- The number of 50s is = 1, iff, the 3rd digit is greater than or equal to 5. In this case the number of 10s is =(3rd digit - 5). Else, the number of 50s is 0, and the number of 10s is equal to the third digit.
- The number of 1s is simply equal to the 4th digit.
- The BCD (4-bit) lines are fed into the hex displays which show the required output, marked as #100s, #50s, #10s, and #1s.

Assumptions

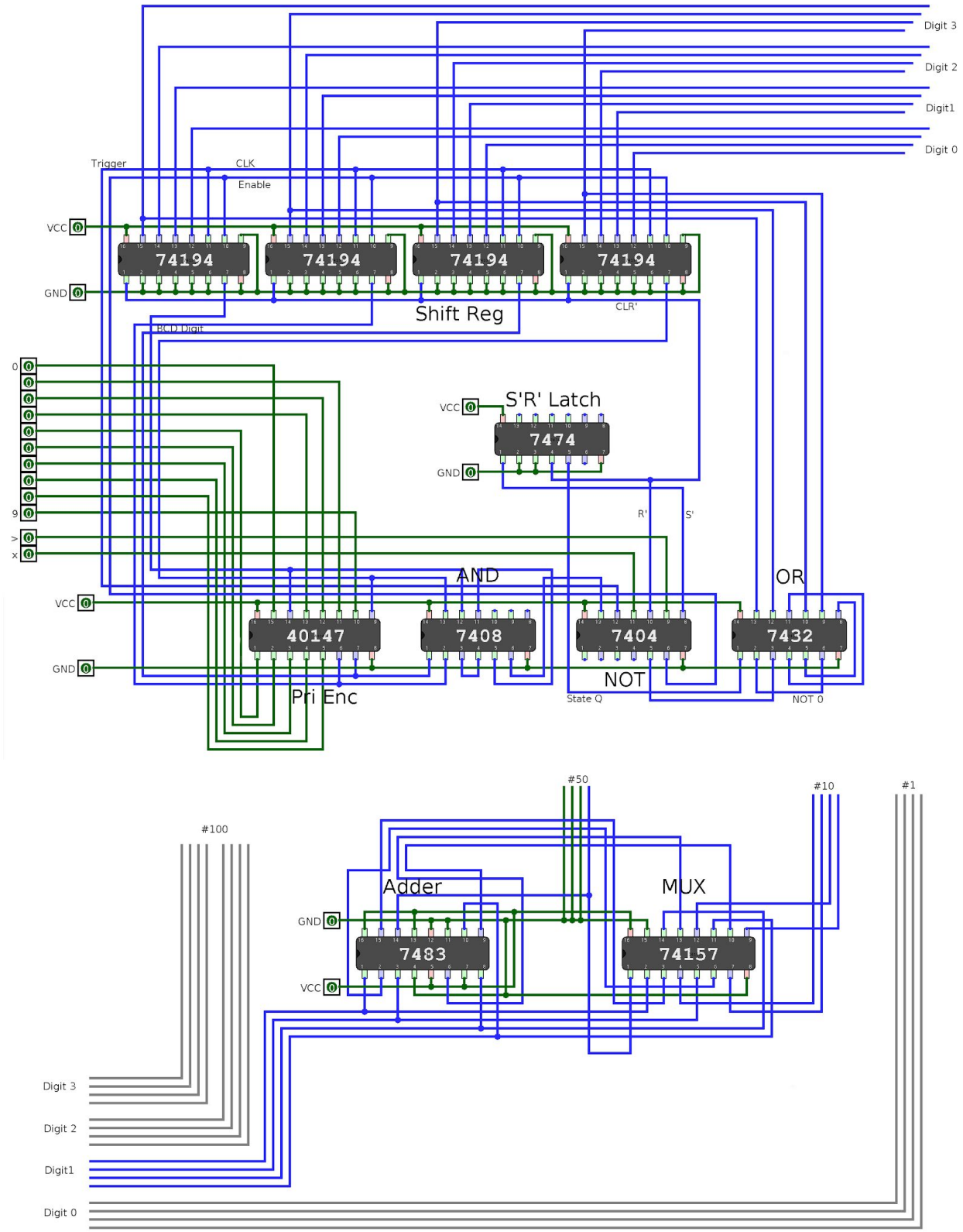
- The question asks for an input in the range of 0-1000. Our circuit can handle numbers from 0-9999.
- The user enters the number sequentially.
- The user presses the next (>) button to get the output.

State diagram

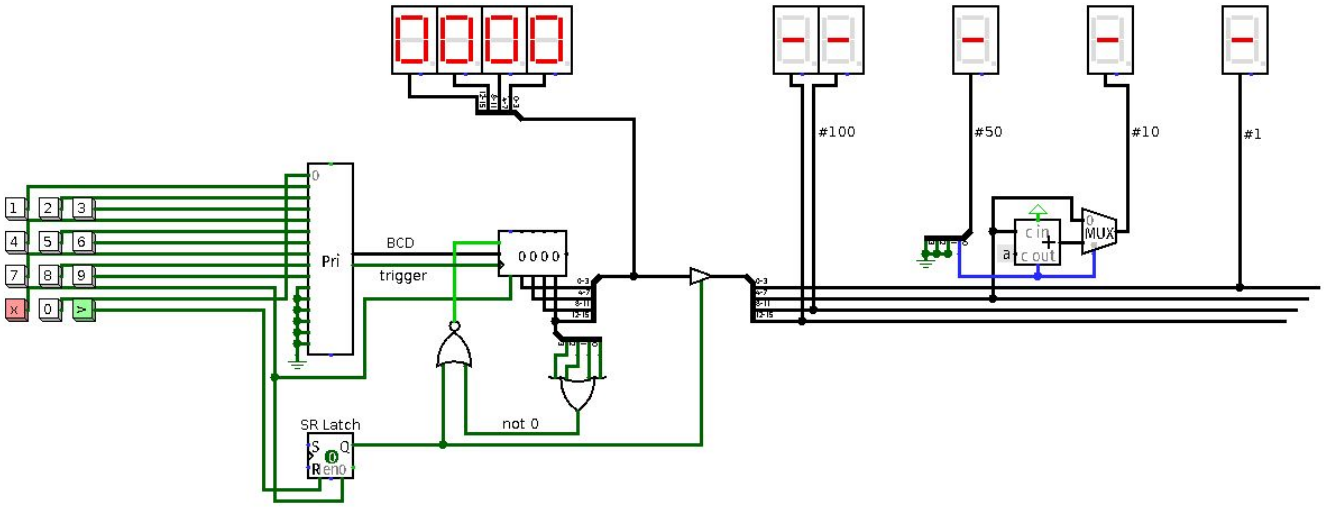


There are only two states -- one for receiving inputs (0) and other for displaying outputs (1).
The latch used for the states is in RESET (0) when the user has not finalized the full input, or they have not pressed the “>” button.
Upon pressing the “>” button, the latch is SET (1), and this opens the connection to the other part of the circuit to display the required output. Using other combinational logic, this also prevents the user from entering any more digits.
The states are controlled asynchronously using a SR-NOR latch. In the diagram above green colored arrow is for S (or >) and red is for R (or x)

Pin out diagram



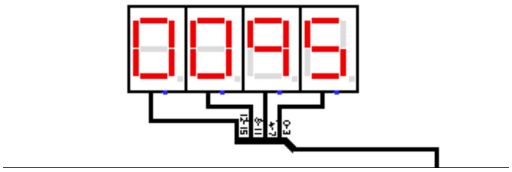
Logisim Implementation



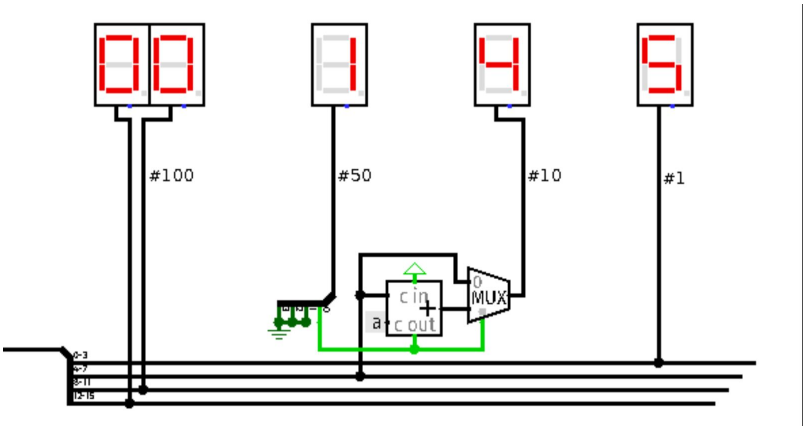
Sample input/output combination

1. Sample input sequence 1: “1-1-1-x-9-5->”

   (To demonstrate the use of “x” button to clear full input and enter once again)

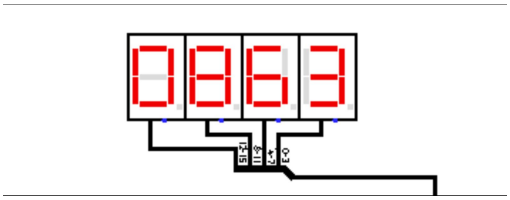


   Output:

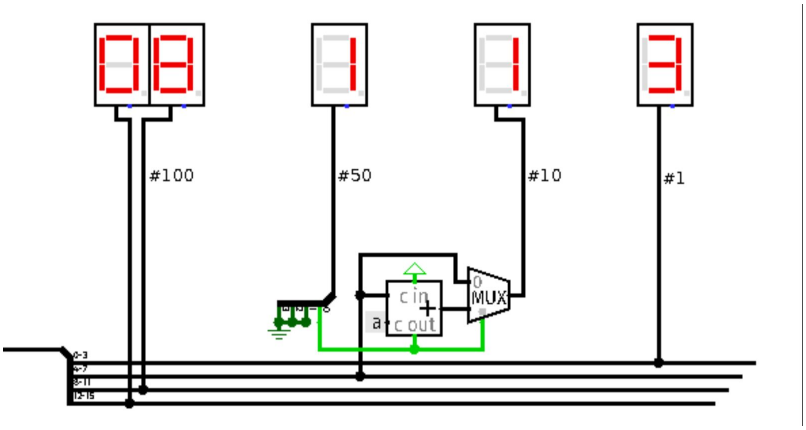


2. Sample input sequence 2: “0-0-0-0-8-6-3->”

   (The above demonstration shows that pressing 0 multiple times before any other number doesn’t change anything because the defaults stored in the shift register is 0)

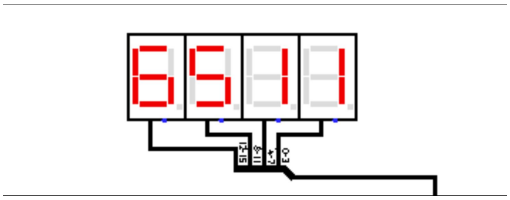


Output:

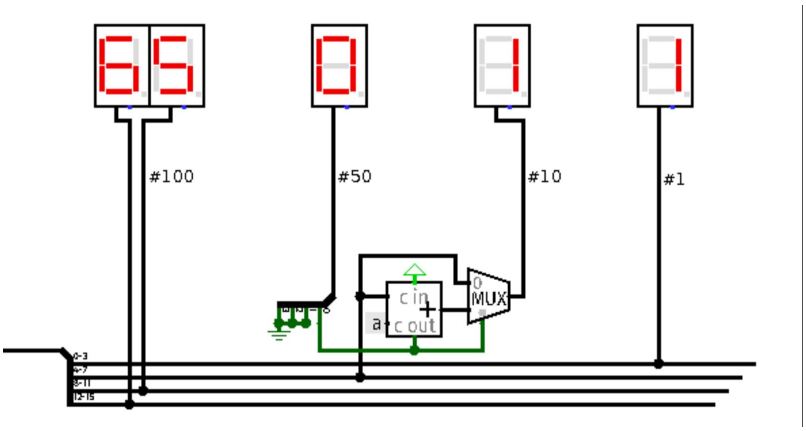


3. Sample input sequence 3: “6-5-1-1-9-8-7-6->”

(To demonstrate that once 4 numbers are entered, the combinational logic deactivates the shift register and it may not accept any more numbers.)



Output:



Extended Circuit Implementation

We propose to make a **Vending machine** that replicates the transaction between the customer and the cashier.
The input to the circuit (machine) is the amount of Cash given (Value) and the price of the Item (MRP). The circuit then calculates the difference between the amounts using a BCD subtractor and returns the money in the form of change of Rs.100 Rs.50 Rs.10 Rs.1 50p 25p.
The circuit is capable of handling numbers from 0 to Rs.9999.99 and able to return values in decimal form.

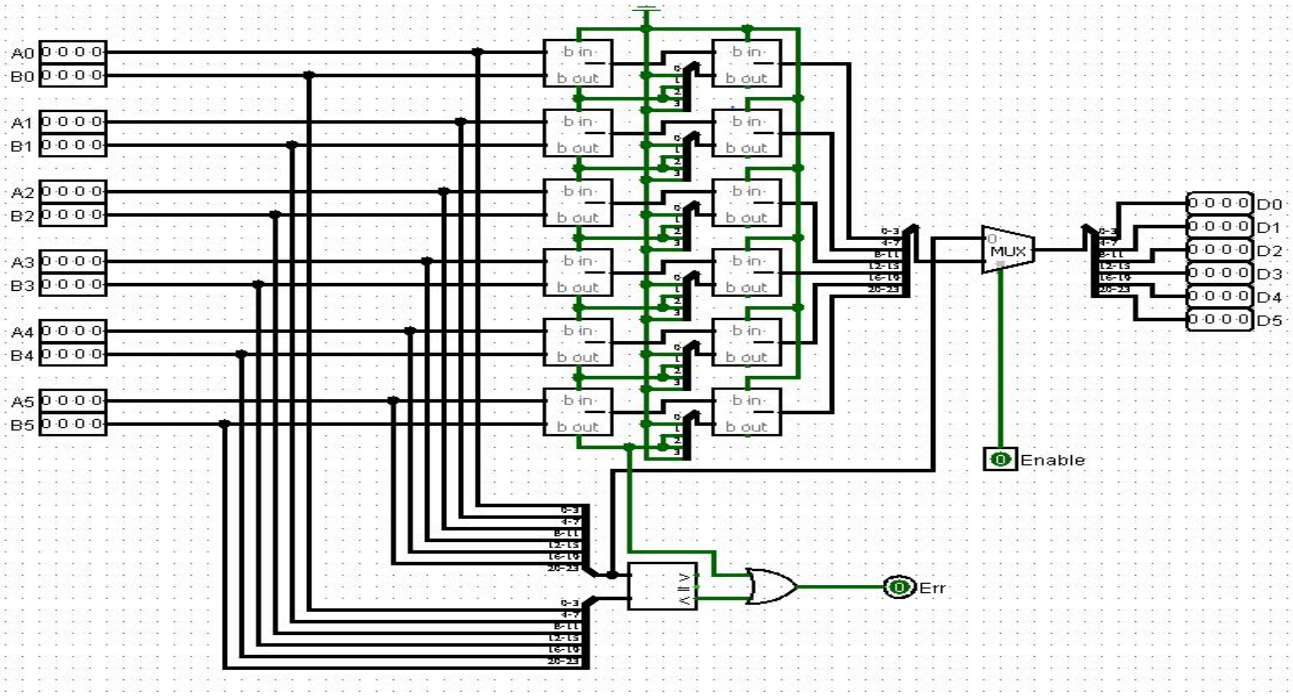
Additional functionalities implemented

1. Two inputs

   The user can enter two numbers, the output will be calculated from the difference of the two numbers. The order of the numbers does not matter.

2. BCD Subtractor

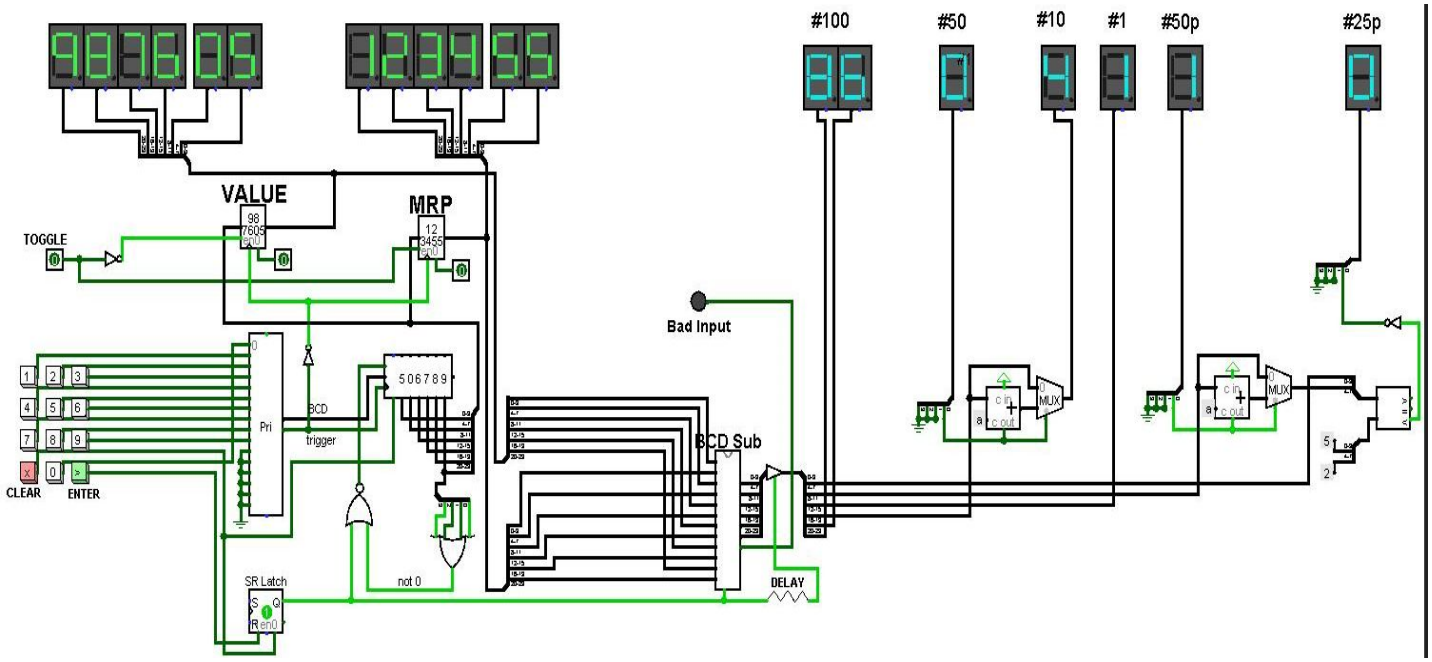
   A BCD subtractor can be constructed using a binary subtractor, MUX, and a comparator for error detection.



Explanation of BCD Subtractor-

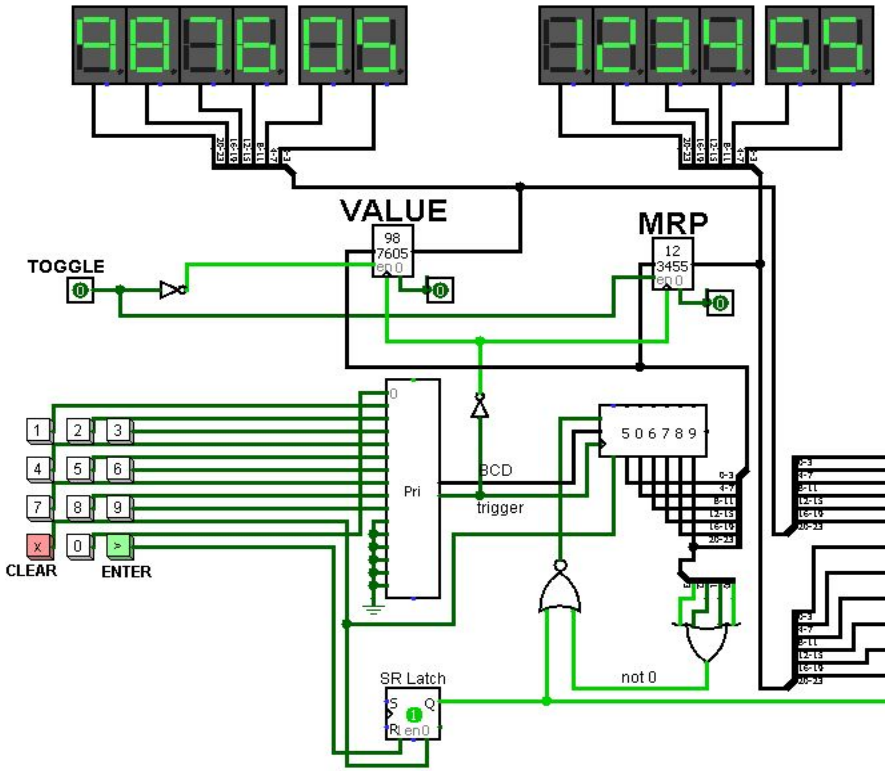
- The BCD subtractor works using 2 sets of binary subtractors both of them connected together.
- The numbers are subtracted pair wise (A0 and B0 , A1 and B1 and so on).
- The output of this is passed into the other subtractor as Minuend and the Borrow out from the previous subtractor is passed as the subtrahend in the case where there is a borrow out, the number equals to 0110 else if the borrow out is 0 then the subtrahend is passed as 0000.
- The output of the second subtractor is then merged together using an inverted splitter and passed into a 2x1 MUX.
- The role of MUX is to pass the output only when the enable is on (1) and the input original number when the enable is off (0).
- A comparator is also used to detect any error, which in this case is when the second number is greater than the first . It is assumed that this does not happen and is considered as a user assumption.

Logisim implementation of the extended version



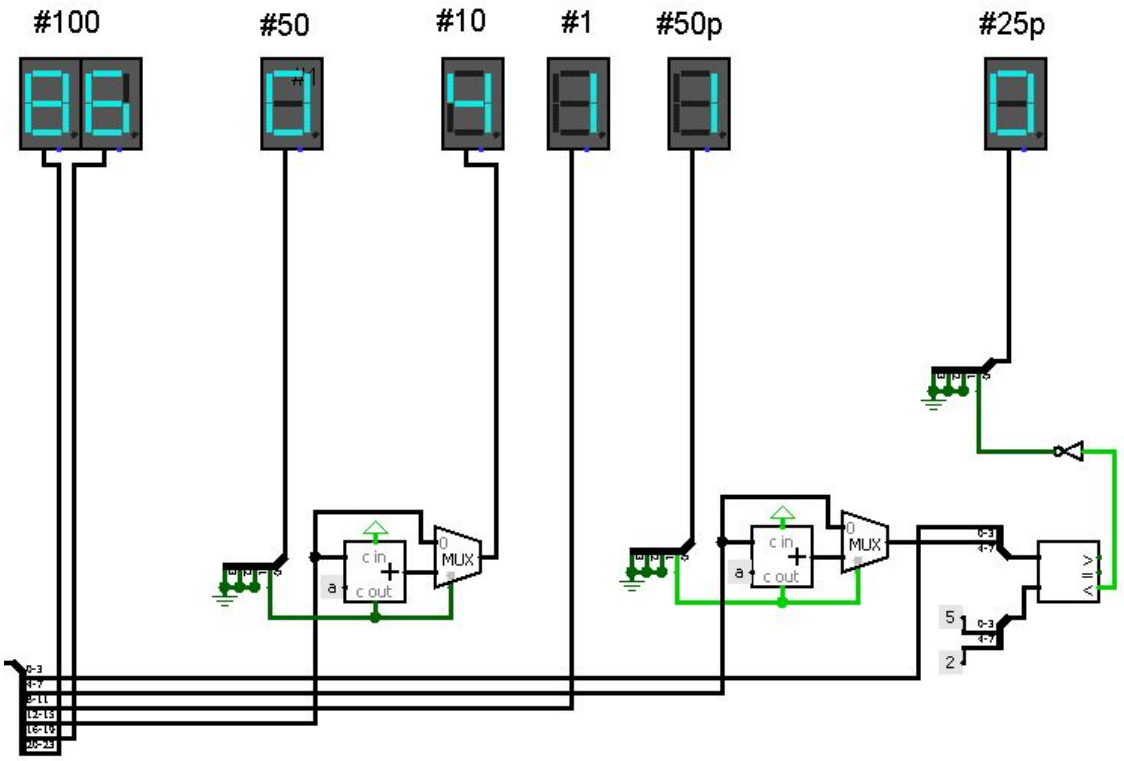
Explanation of the circuit

Input section



- The input section now consists of 2 registers that store the value of Value and MRP as input from the user
- Separate clear functions for Value and MRP have been implemented.
- User first enters the value of the amount he has paid and then Flips the toggle provided to further enter the value of MRP.
- Finally the Enter button must be pressed for the circuit to work. This would process the difference between the two and display the change that needs to be given.

Explanation of output



- The logic of #50p output is the same as that of #50 rupees as explained in the main circuit.
- The output of the MUX is either the difference between digit at the tenth’s place and 5, or the digit itself, depending on the carry output of the adder.
- This output is fed into the 4-7 bit input of the splitter and the hundredth place digit of the user input is fed into the 0-3 bit input of the splitter. Let us assume the output of the splitter is the number, N.
- The comparator compares the number, N, with 25. If N is greater than 25, then the output, i.e. #25p, is 1 else, it is 0.

Assumptions for the Extended Version:

1. There are 2 modes for this circuit, operated by an input pin-button. If the pin is 0, user inputs the amount of cash given and if the pin is 1, the user inputs the Cost price of the item
2. After completing the amount for either, the user changes the pin output (by clicking) and also Must CLEAR the shift register (by pressing “x”), to enter the second input. Once both inputs are entered the user views the desired output by, as usual, pressing the “>” button.
3. It is obvious that the Cash entered must be >= The Cost price. If this does not follow, an LED labelled “bad input” lights up and the output displayed is invalid.
4. We have used 50p and 25p as they are more common, and coins like 10p and 1p are outdated and not in use. The user must enter an amount which is a multiple of 25p or 50p else Least amount of these coins needed will be displayed, i.e :
   The extended circuit does all that the main circuit does along with the following

- If the fractional part of the input is 50, the number of 25p coins dispensed is 0 and that of 50p coins is 1.
- If the fractional part is between 25 and 50, the number of 25p coins dispensed is 1 and that of 50p coins is 0.
- If the fractional part is greater than 74, the number of both, 25p and 50p, coins dispensed is 1.

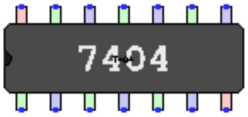
5. Assumption for decimal input:

   The user will input the number upto two decimal places.

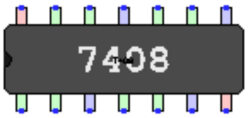
- For example, if the user wishes to enter 95.50, the input should be 9-5-5-0->.
- If the user enters 9-5-5->, the circuit will assume the input to be 9.95.
- Also, if the user wishes to enter a whole number, say 195.00, the input should be 1-9-5-0-0->.

Bill of materials/ Appendix

- Hex Inverting Gates (DM7404)
  https://drive.google.com/drive/folders/11h-J6kwWPyvJCp8FzIjhZ-0lM7yNGDo5

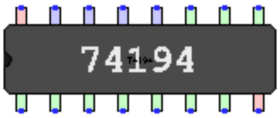


- Quad 2-Input AND Gates (DM7408)
  https://drive.google.com/drive/folders/11h-J6kwWPyvJCp8FzIjhZ-0lM7yNGDo5



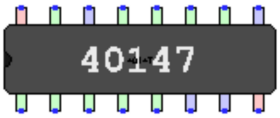
- Quad 2-Input OR Gates (DM7432)
  https://pdf1.alldatasheet.com/datasheet-pdf/view/131609/NSC/DM7432.html

- 4bit Universal Shift Register (74LS194)
  https://drive.google.com/drive/folders/11h-J6kwWPyvJCp8FzIjhZ-0lM7yNGDo5

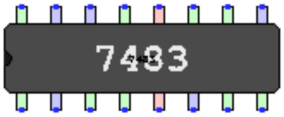


- Dual D-type flip-flop (7474)
  https://drive.google.com/drive/folders/11h-J6kwWPyvJCp8FzIjhZ-0lM7yNGDo5

- 10-to-4 line Priority Encoder (CD40147BMS)
  https://pdf1.alldatasheet.com/datasheet-pdf/view/66402/INTERSIL/CD40147BMS.html



- 4bit Adder (SN54/74LS83A)
  https://drive.google.com/drive/folders/11h-J6kwWPyvJCp8FzIjhZ-0lM7yNGDo5



- Quad 2:1 MUX (74HC157)
  https://assets.nexperia.com/documents/data-sheet/74HC_HCT157.pdf