Design of Double-Digit Calculator in Circuit Maker

Name: *Rekha Lokesh* **Roll No:** *19EC10052*

This is an assignment addendum of the depth course Digital Electronic Circuits Lab (EC39003) taken in Autumn Semester 2021-22.

Problem Statement:

Design a **PCB model** of an **automated non-ALU** Double-Digit Calculator Digital Circuit in **Circuit Maker**

Procedure:

a) Take input in **single ASCII keypad** in seq: a, b, op, c, d, '=' which mean perform (ab)op(cd) and display result after pressing =
i.e., ab+cd or ab-cd where ab and cd are BCD representation of two numbers p and q

b) Each press binary values are stored in registers (6 PIPO registers). After detecting '=' using its ASCII value, convert ab and cd in **BCD form to Binary equivalents** A and B using BCD to Binary Algorithm (self-found).
Since A = a*10+b => A = a*(8+2) + b where a*8 is left shift of a by 3 and a*2 is left shift of a by 1. So, A = a<<3+a<<1+b = (a<<3) + b + (a<<1)
Hence **A = (a<<3) + b + (a<<1)** and **B = (c<<3) + d + (c<<1)** ……. can be done using 3 adders for finding A and 3 for B……. result will be A_H & A_L And B_H & B_L

c) The 7-bit A and 7-bit B are added or subtracted depending on operation with 8th bit represented as resultant sign for op = '-' and as resultant bit value when op = '+'
i.e., Let C = A + B and C be $c_0, c_1, \ldots, c_7$
So **c_7 = 1/0** when op is '+'
Else **c_7 is 0** for op = '-' since for – max value is -99 and c_7 is 1 for |C|>127

d) The **operation** is **detected** using **3rd least significant bit of register 3** and $reg3_2 = 1$ if op is '-' else 0

e) In case of '–', C is computed by subtracting B from A using **2s complement subtraction**
   i. Take 2s complement of subtrahend i.e., take its 1s complement and add 1
   ii. Add minuend and 2s complement of subtrahend.
   iii. If there is a **carry** in the overall sum, **ignore it** since the **result is +ve** i.e., if minuend and subtrahend are 7-bit binary numbers (as in this case) then if **8th least significant bit gets carry** (i.e., it becomes 1 if initially 0) then ignore it as the result is +ve

iv. Else **take 2s complement** of the **7 bits ignoring the 8th bit** and that's the result as the **result is -ve**.

f) Now, following above steps, you get final result **res** in 8-bit binary. Convert this result from **binary to BCD** using **shift and add** algorithm in **Double Dabble Binary to BCD Conversion Algorithm**.

g) This algo says:

   i. Shift all the bits **left sequentially k** number of times where **k is the length of the bit stream** i.e., 8 for 8-bit binary number. The 4-length columns to the left are units, tenths, hundredths etc.

   ii. If at any step **excluding after last shift**, the value in any column becomes **greater than 4, then add 3** to it and then continue shifting.

   Note:

   1) >4 since if its 5,6 and so on then in next shift, value will be greater than 10 which isn't a digit in decimal)

   2) Add 3 since after the next shift it will add 6 to >=10 values which make it BCD equivalent of 2-digit numbers.

      E.g..., curval = 5 => after next shift => 10 whose BCD is 16 since

      $(0001)_2(0000)_2 = (10)_{10}$

      So curval+=3 => curval = 8 => curval<<=1 => curval = 16 = BCD equivalent of 10

   iii. This will give the BCD of any binary value which is easy to represent in digital circuits using 7 seg display

h) So, in our case, res is 8-bit binary which requires 8-3 = 5 set of comparator and adder in 1st iteration, 5-3 = 2 set in 2nd iteration (stop iteration). So, total 5+2 = **7 sets** of **comparators and adders** required.

   Note:

   i. -3 since at first, we take the 1st three most significant bits (bypassing 2 shifts) and last shift in circuit is required as no addition is being made there. That last digit (res_0) directly goes to least significant bit in units place of BCD.

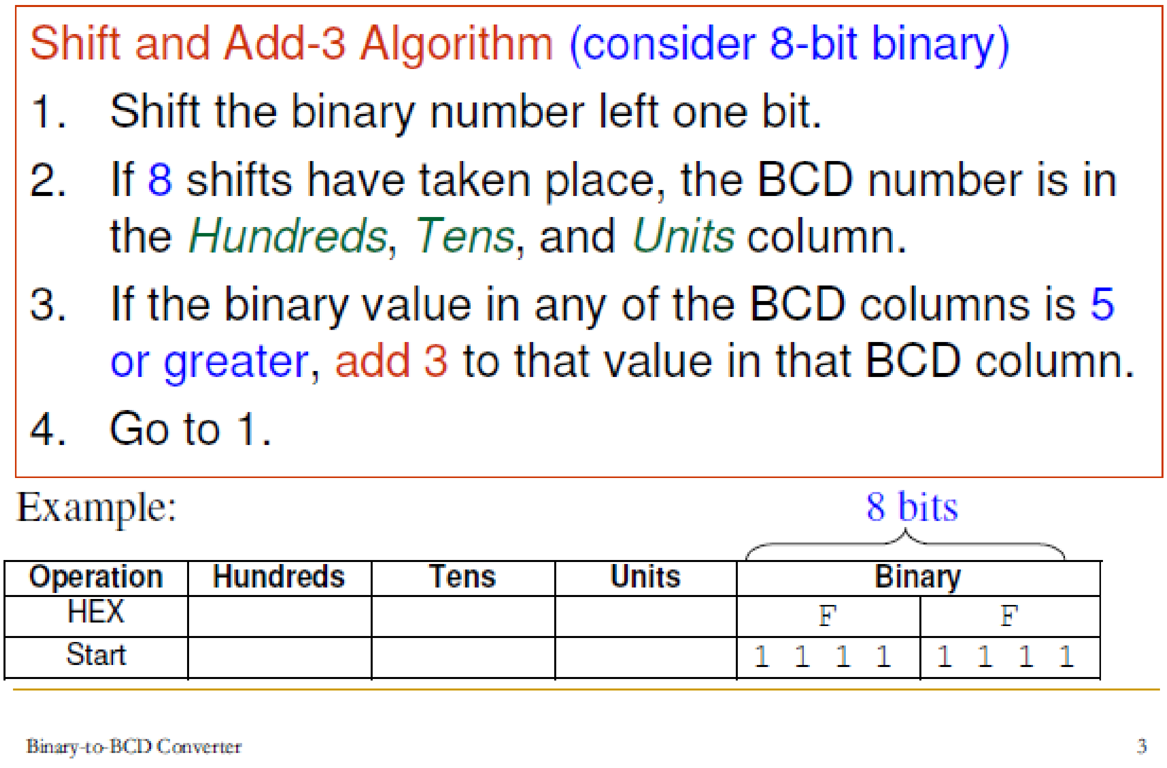
   ii. Above, iteration stops when current length of bits are <= 3.

i) In the final result display of the circuit, each place (units, tenths, hundredths) are displayed using three 7-segment display connected to 7447 (BCD to 7 segment decoder) whose inputs are equivalent 4-length BCD codes of individual places. The final res lines are BCD lines (they are divided in chunks of 4 from left most significant bits place).

j) For the hundredths place, the possible display can be '-' or '1'.

i. '-' is possible when operation is subtraction and 8th bit is 0 in case of subtraction (as said in 2s complement subtraction above).

ii. Otherwise, display is blank/'1' which is detected easily from least significant bit in hundredths place of BCD result.



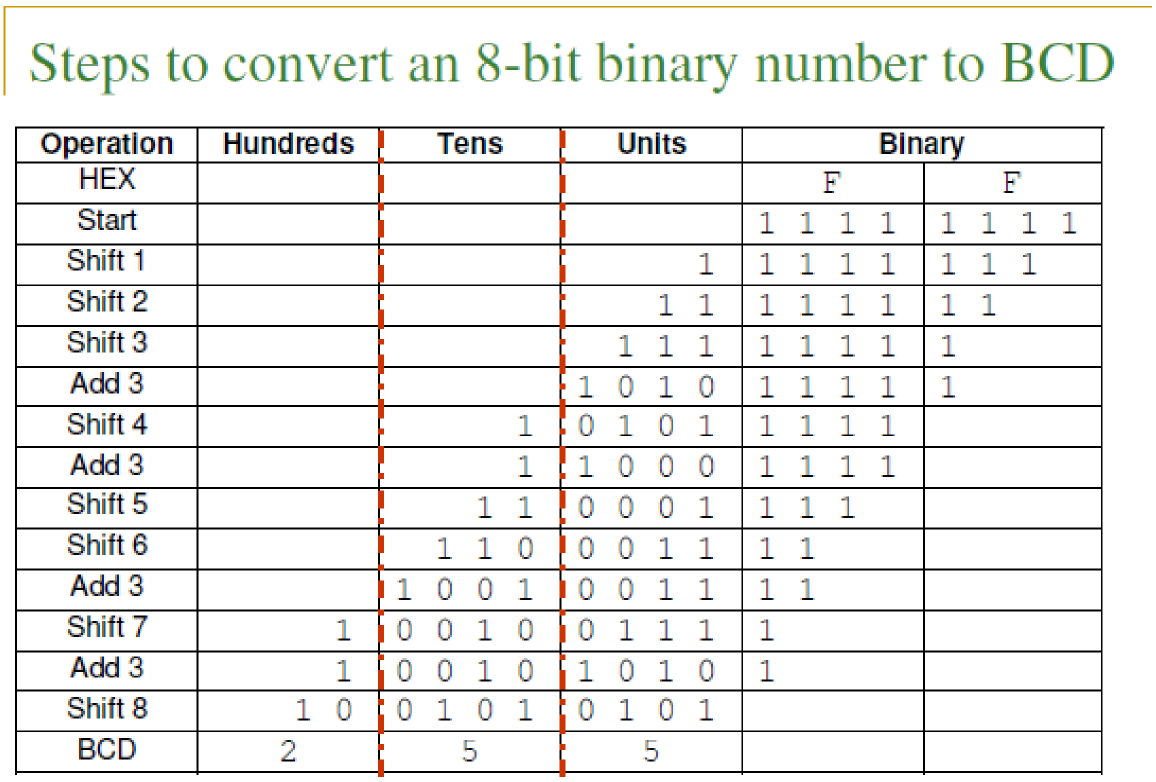
Shift and Add-3 Algorithm (consider 8-bit binary)

1. Shift the binary number left one bit.
2. If 8 shifts have taken place, the BCD number is in the *Hundreds*, *Tens*, and *Units* column.
3. If the binary value in any of the BCD columns is 5 or greater, add 3 to that value in that BCD column.
4. Go to 1.

Example:

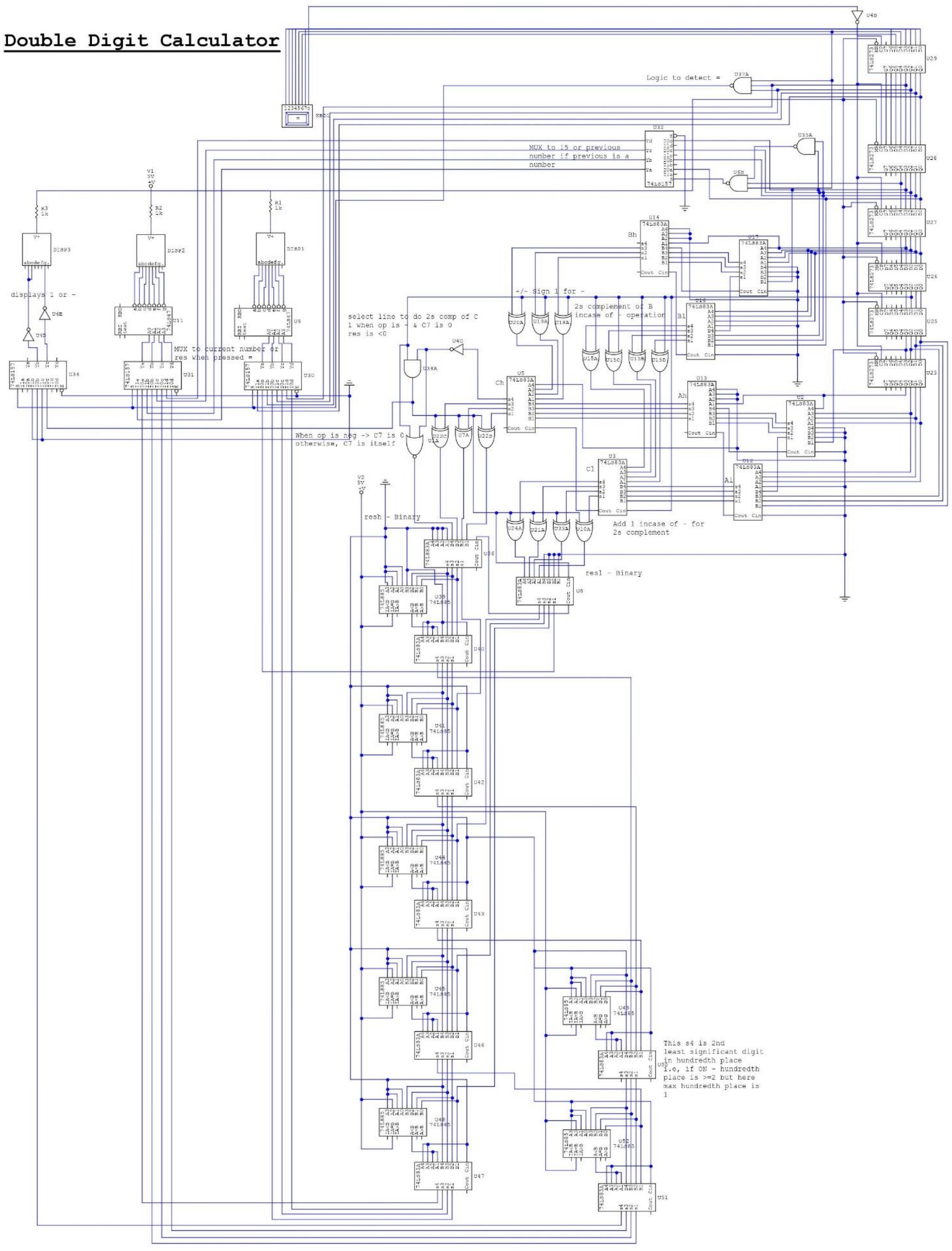
8 bits

Operation	Hundreds	Tens	Units	Binary	
HEX				F	F
Start				1 1 1 1	1 1 1 1



Steps to convert an 8-bit binary number to BCD

Operation	Hundreds	Tens	Units	Binary	
HEX				F	F
Start				1 1 1 1	1 1 1 1
Shift 1			1	1 1 1 1	1 1 1
Shift 2			1 1	1 1 1 1	1 1
Shift 3			1 1 1	1 1 1 1	1
Add 3			1 0 1 0	1 1 1 1	1
Shift 4		1	0 1 0 1	1 1 1 1	
Add 3		1	1 0 0 0	1 1 1 1	
Shift 5		1 1	0 0 0 1	1 1 1	
Shift 6		1 1 0	0 0 1 1	1 1	
Add 3		1 0 0 1	0 0 1 1	1 1	
Shift 7	1	0 0 1 0	0 1 1 1	1	
Add 3	1	0 0 1 0	1 0 1 0	1	
Shift 8	1 0	0 1 0 1	0 1 0 1		
BCD	2	5	5		


Double Digit Calculator
Logic to detect =
MUX to 15 or previous number if previous is a number
displays 1 or -
MUX to current number or res when pressed =
select line to do 2s comp of C 1 when op is - & C7 is 0 res is <0
+/- Sign 1 for -
2s complement of B incase of - operation
When op is neg -> C7 is 0 otherwise, C7 is itself
resh - Binary
Add 1 incase of - for 2s complement
res1 - Binary
This s4 is 2nd least significant digit in hundredth place i.e, if ON - hundredth place is >=2 but here max hundredth place is 1

For Further Reference:

GITHUB Link – *Link to the circuit and other files*