Ain Shams University

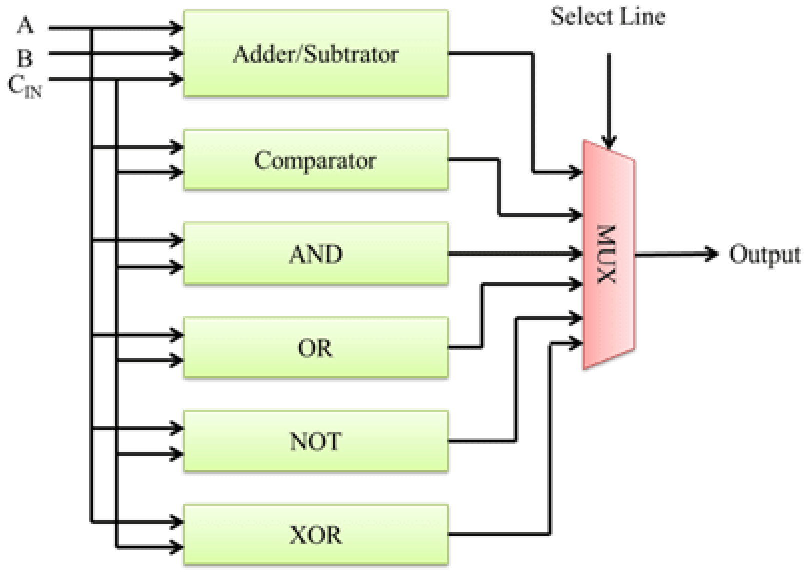
Faculty of Engineering (ICHEP)

Mechatronics Engineering & Automation Program

Logic Design (CSE 111)

Presented to: Dr Mohamed Omar & Eng Osama Samy

Major Task – Phase 1



Name	*ID*
1- Mohamed Mohamed Taha	21P0137
2- Omar ElShawaf	21P0410

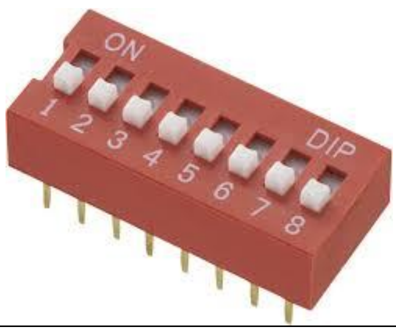
Table of Contents

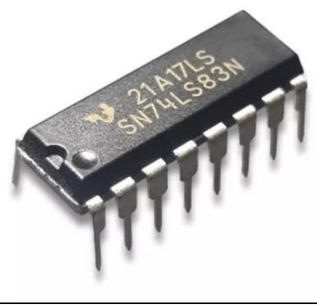
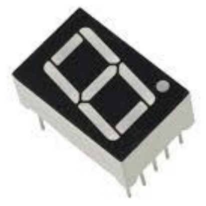
Introduction:

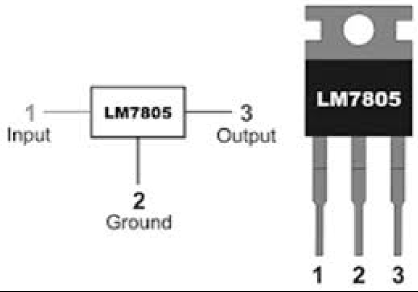
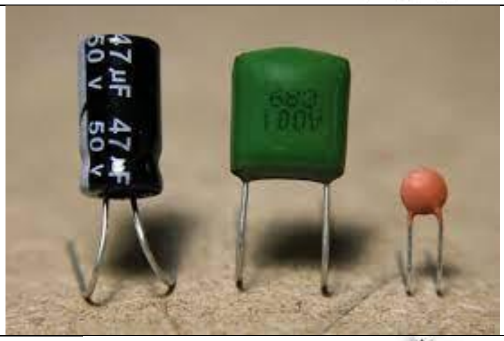
We're designing and implementing our own ALU that will take 2 numbers as inputs (4-bits each) and will output the result on 7-segment display. The user will be able to select which function to perform. We've done the following operations:

1- **Addition**
2- **Subtraction**
3- **Multiplication**
4- **AND**
5- **OR**
6- **Complement A**
7- **Increment A**

Components:

DIP Switch	
Dual 4-to-1 Mux (74153)	
Quad 2-input XOR (7486)	

Quad 2-input AND (7408)	
Quad 2-input OR (7432)	
Hex inverter (7404)	
4-bit Full Adder (7483)	
7-Segment Display (Common Anode)	

7-Segment Decoder (7447)	
LEDs	
Voltage Regulator (7805)	 1 Input LM7805 3 Output 2 Ground LM7805 1 2 3
Capacitors	
Jumpers	

9V Adapter	

Truth Tables:

1- Complement Operation:

- **The complement operation is applied on input A only**
- **Here, we just toggle the 0’s & 1’s to get the output**

A (Input)				Y (Output)			
A3	A2	A1	A0	Y3	Y2	Y1	Y0
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	0
0	0	1	0	1	1	0	1
0	0	1	1	1	1	0	0
0	1	0	0	1	0	1	1
0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	0	0
1	0	0	0	0	1	1	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	1	1
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	0

2- OR Operation

- **Here, OR operation is applied on both inputs**
- **We take each 2 corresponding bits and apply an OR operation on them according to the following truth table:**

A (Input 1)				B (Input 2)				Y (Output)			
A3	A2	A1	A0	B3	B2	B1	B0	Y3	Y2	Y1	Y0
0	0	0	1	1	1	1	0	1	1	1	1
0	0	1	0	0	1	0	1	0	1	1	1
0	0	1	1	1	0	0	0	1	0	1	1
0	1	0	0	1	0	0	1	1	1	0	1
0	1	0	0	1	1	1	0	1	1	1	0

- **Now, we'll demonstrate some outputs of the OR operation on 4 bits of both inputs:**

A0 (Input 1)	B0 (Input 2)	Y0 (Output)
0	0	0
0	1	1
1	0	1
1	1	1

3- AND Operation

- **Here, AND operation is applied on both inputs**
- **We take each 2 corresponding bits and apply an AND operation on them according to the following truth table:**

A0 (Input 1)	B0 (Input 2)	Y0 (Output)
0	0	0
0	1	0
1	0	0
1	1	1

- **Now, we'll demonstrate some outputs of the AND operation on 4 bits of both inputs:**

A (Input 1)				B (Input 2)				Y (Output)			
A3	A2	A1	A0	B3	B2	B1	B0	Y3	Y2	Y1	Y0
0	0	0	1	1	1	1	0	0	0	0	0
0	0	1	0	0	1	1	1	0	0	1	0
0	0	1	1	1	0	0	0	0	0	1	1
1	1	0	0	1	0	0	1	1	0	0	0
1	1	0	0	1	1	1	0	1	1	0	0

4- Addition Operation:

- **Here, we add both inputs**
- **We take each 2 corresponding bits and apply an addition operation on them according to the following truth table:**

A0 (Input 1)	B0 (Input 2)	Cin (Carry in)	S0 (Output)	C0 (Carry)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- **Now, we'll demonstrate some outputs of the Addition operation on 4 bits of both inputs:**

A (Input 1)				B (Input 2)				Y (Output)				
A3	A2	A1	A0	B3	B2	B1	B0	Y4	Y3	Y2	Y1	Y0
0	0	0	1	1	1	1	0	0	1	1	1	1
0	0	1	0	0	1	1	1	0	1	0	0	1
0	0	1	1	1	0	0	0	0	1	0	1	1
1	1	0	0	1	0	0	1	1	0	1	0	1
1	1	0	0	1	1	1	0	1	0	1	1	0

- Output here is in 5 bits since the max of addition of 2 4-bit inputs is 30, so we need an extra bit to represent these numbers

5- *Subtraction Operation:*

- Here, we subtract input B from input A
- To perform subtraction, we'll bring 2's complement of input B by bringing 1's complement first (toggling 1's & 0's) then adding 1, then add this to input A
- If the carry over bit is 1, number is positive and will be removed and the output will be in 4 bits
- If the carry over bit is 0, number is negative, so we'll bring 2's complement of result and output will be 2's complement (result of subtraction) and display the negative sign
- Here are 2 examples to differentiate both:

① 15 − 3 = 12

$$\begin{array}{r} 1111 \\ -\,0011 \\ \hline \end{array}$$

↓

$$\begin{array}{r} {}^{1}{}^{1}{}^{1}\;\; \\ 1111 \\ +\,1101 \\ \hline \cancel{1}1100 \end{array}$$

2's complement (0011)

↓

1's complement + 1

↳
$$\begin{array}{r} 1100 \\ +\;\;\;\;1 \\ \hline 1101 \end{array}$$

Carry over bit = 1 ∴ number is +ve

$$\boxed{\text{Output} = 1100}$$

$$\begin{array}{r} \hline \textcircled{0}1001 \end{array}$$

0110

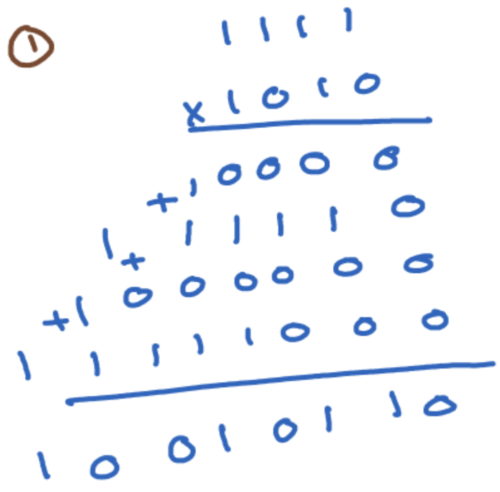
Carry over bit = 0 ∴ number is −ve

Output = − 2's complement (1001)

$$\boxed{\text{Output} = -(0111)}$$

6- *Multiplication Operation:*

- **Here, we multiply A and B**
- **It's like the mathematical multiplication, we take each bit from input B and multiply it with all bits of input A, and we add when we move on to the next bit of input B**
- **Here's an example to understand:**



Output : 10010110

- **Here, the maximum is 225 which will need 8 bits for representation and 3 7-segments**

7- *Increment Operation:*

- **Here, we add 1 to input A**
- **Here's the truth table for the operation:**

A (Input)				Y (Output)				
A3	A2	A1	A0	Y4	Y3	Y2	Y1	Y0
0	0	0	0	-	0	0	0	1
0	0	0	1	-	0	0	1	0
0	0	1	0	-	0	0	1	1
0	0	1	1	-	0	1	0	0
0	1	0	0	-	0	1	0	1
0	1	0	1	-	0	1	1	0
0	1	1	0	-	0	1	1	1
0	1	1	1	-	1	0	0	0
1	0	0	0	-	1	0	0	1
1	0	0	1	-	1	0	1	0
1	0	1	0	-	1	1	0	1

1	0	1	1	-	1	1	0	0
1	1	0	0	-	1	0	1	1
1	1	0	1	-	1	1	1	0
1	1	1	0	-	1	1	1	1
1	1	1	1	1	0	0	0	0

- **Here, we have the last one as 5 bits since the answer is 16 and this needs 5 bits to represent**

Designing Circuit

1- Complement Operation:

- **Here are the equations of the output**

Equations

$$Y3 = A3'$$

$$Y2 = A2'$$

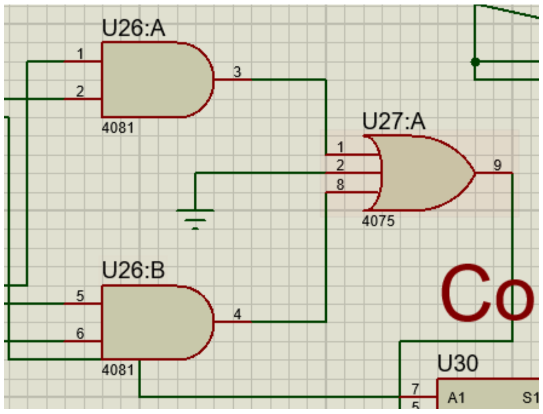
$$Y1 = A1'$$

$$Y0 = A0'$$

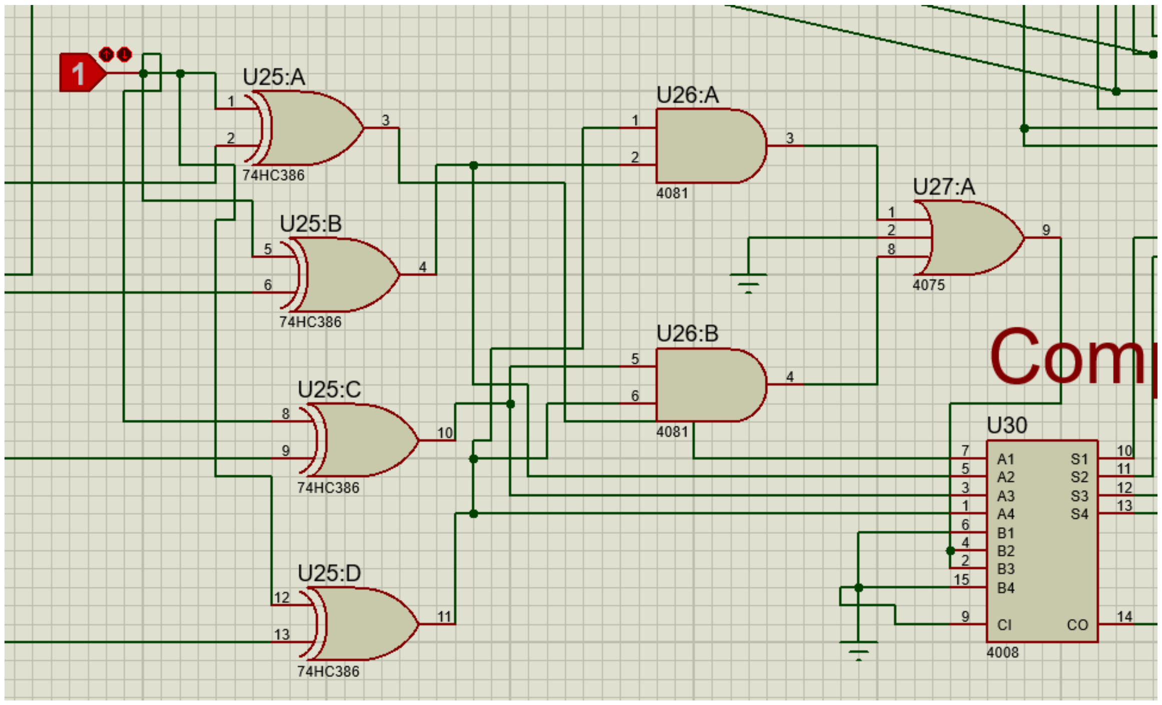
- **To bring complement, instead of using not, we're going to use an XOR IC, where each input bit will be XOR with 1 to give the complement according to this XOR truth table:**

A0 (Input 1)	B0 (Input 2)	Y0 (Output)
0	0	0
0	1	1
1	0	1
1	1	0

- **Here, the maximum number is 15, but 1 7-segment can display up to 9 maximum, so we need to convert the 15 into binary coded decimal (BCD) to be able to display on 7 segment**
- **Here's the circuit that detects if the result is greater than 9 or not:**



- **If number is greater than 9, we need to add 6, so we're going to use a 4-bit full adder in this part**
- **The 4 outputs of the adder will go to 7-segment display, and the carry over bit will go to another 7 segment display to show the 1**
- **Here's an image of the circuit after implementation on proteus:**



2- OR Operation:

- **Here are the equations of the output:**

Equations

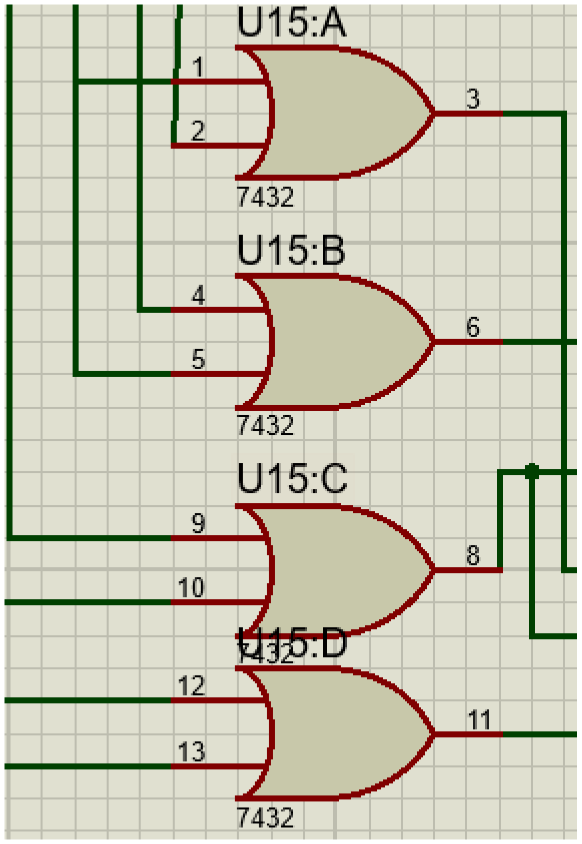
$$Y3 = A3 + B3$$

$$Y2 = A2 + B2$$

$$Y1 = A1 + B1$$

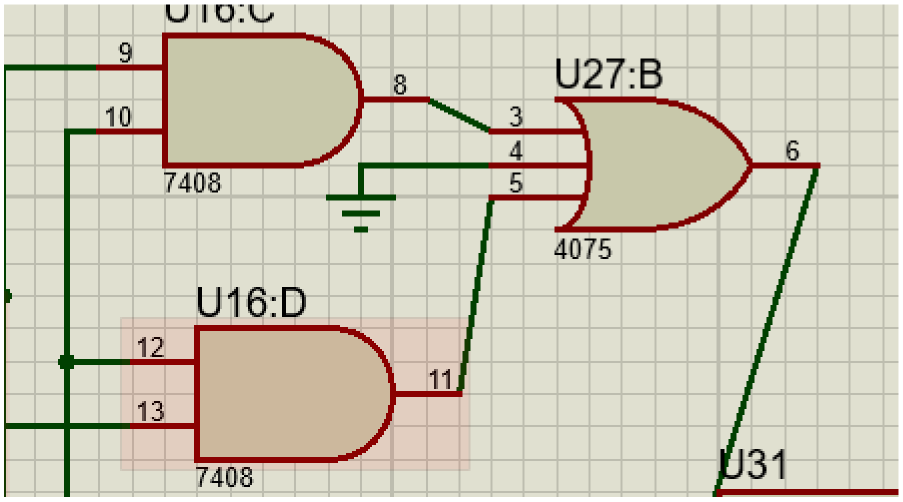
$$Y0 = A0 + B0$$

- To bring output, we're just going to use an OR IC to do an OR operation for each 2 corresponding bits of the inputs
- The circuit will look like this:

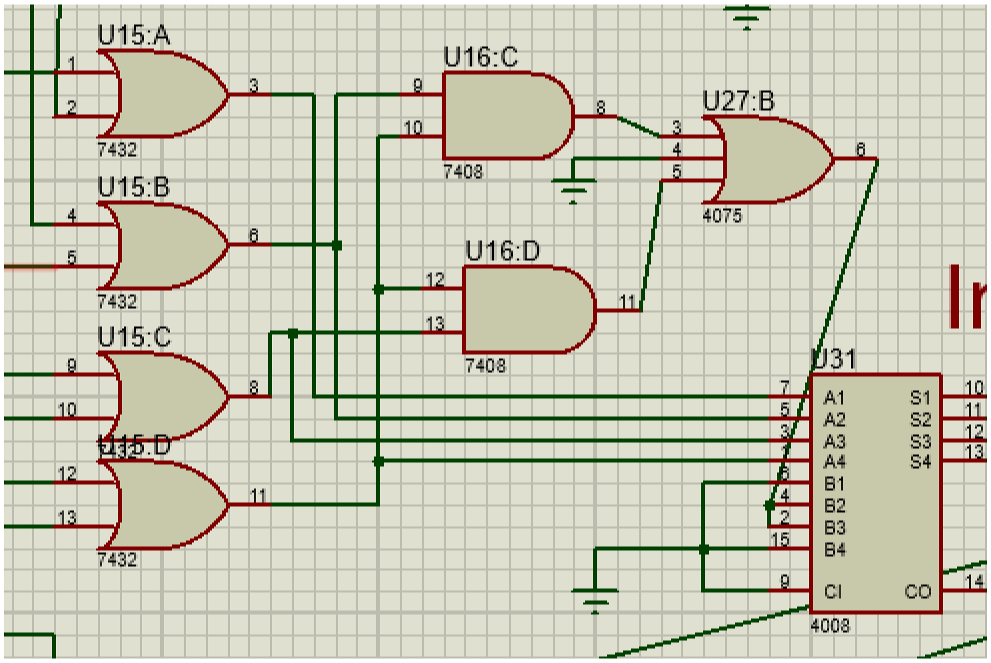


The inputs to the OR gates are coming from the inputs A & B

- Here, the maximum number is 15, but 1 7-segment can display up to 9 maximum, so we need to convert the 15 into binary coded decimal (BCD) to be able to display on 7 segment
- Here's the circuit that detects if the result is greater than 9 or not:



- If number is greater than 9, we need to add 6, so we're going to use a 4-bit full adder in this part
- The 4 outputs of the adder will go to 7-segment display, and the carry over bit will go to another 7 segment display to show the 1
- Here's an image of the circuit after implementation on proteus:



3- AND Operation:

- Here are the equations of the output:

Equations

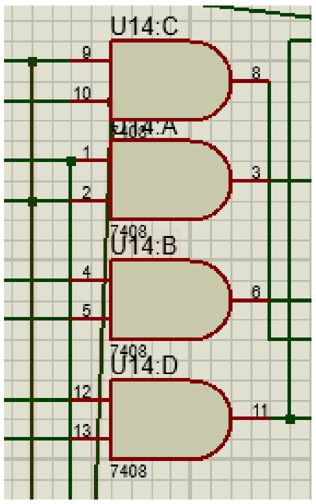
$$Y3 = A3 . B3$$

$$Y2 = A2 . B2$$

$$Y1 = A1 . B1$$

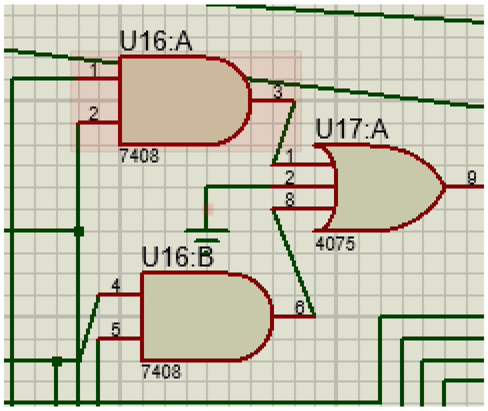
$$Y0 = A0 . B0$$

- To bring output, we're just going to use an AND IC to do an AND operation for each 2 corresponding bits of the inputs
- The circuit will look like this:

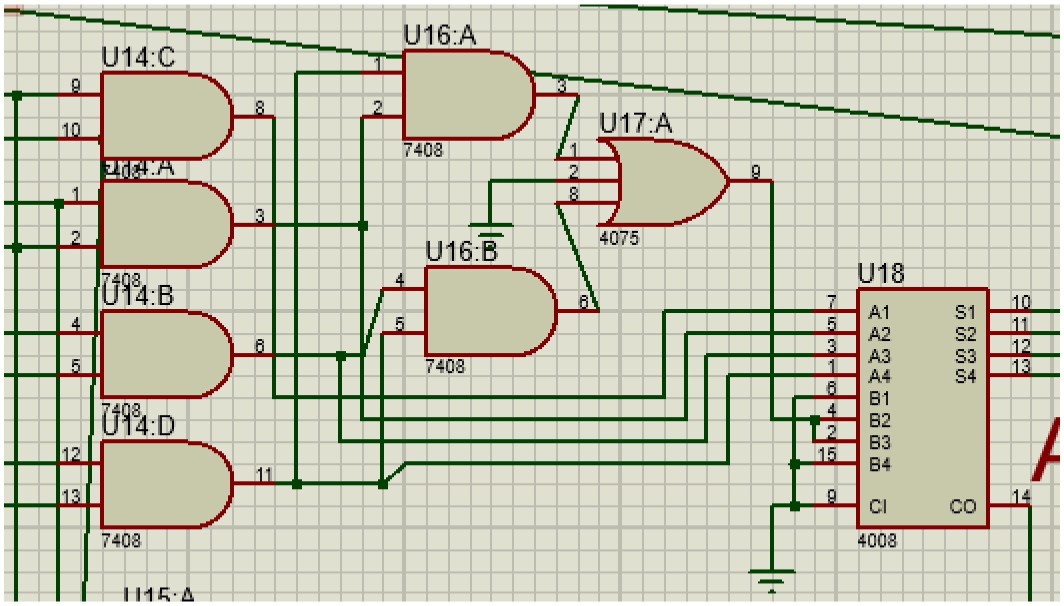


The inputs to the OR gates are coming from the inputs A & B

- Here, the maximum number is 15, but 1 7-segment can display up to 9 maximum, so we need to convert the 15 into binary coded decimal (BCD) to be able to display on 7 segment
- Here's the circuit that detects if the result is greater than 9 or not:



- If number is greater than 9, we need to add 6, so we're going to use a 4-bit full adder in this part
- The 4 outputs of the adder will go to 7-segment display, and the carry over bit will go to another 7 segment display to show the 1
- Here's an image of the circuit after implementation on proteus:

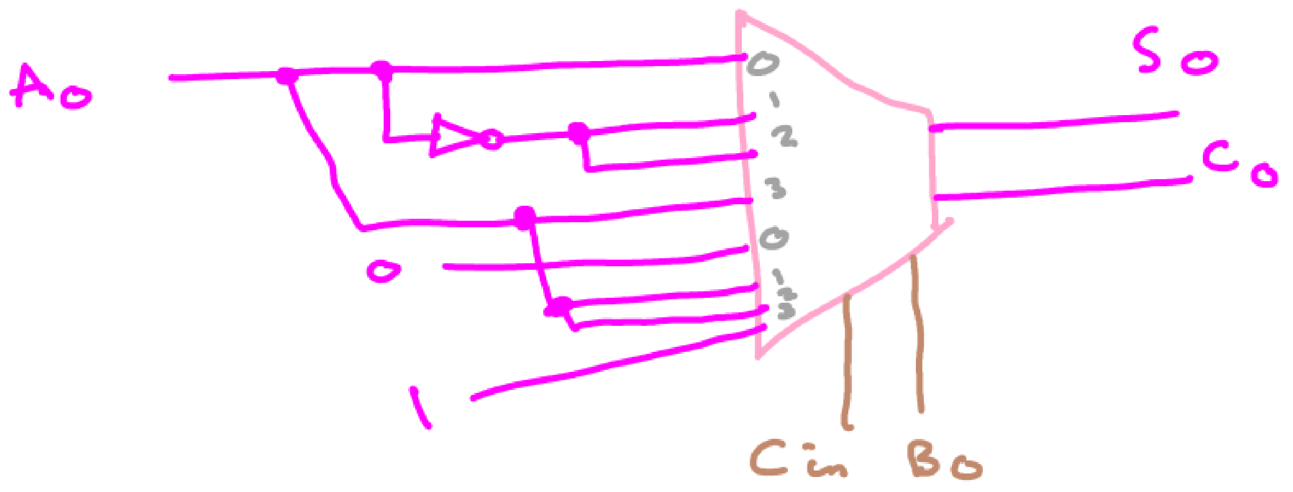


4- *Addition Operation:*

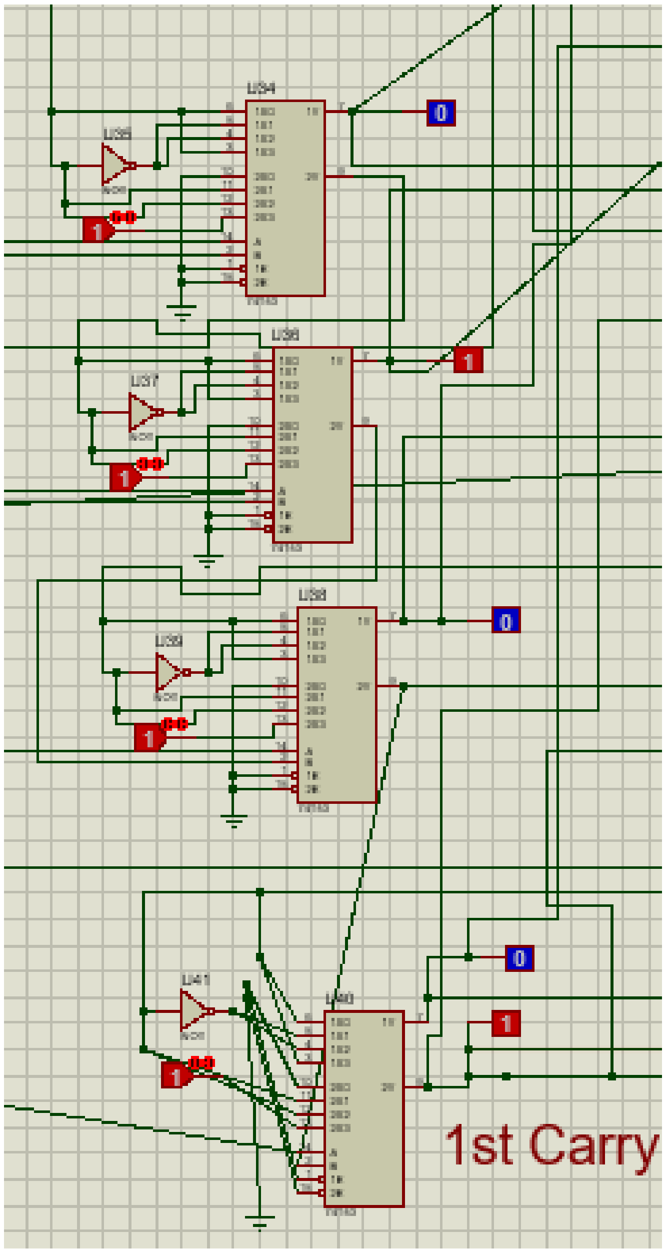
- We're going to implement addition using 4 Dual 4-to-1 Multiplexers (74153)
- According to the truth table of the adder and to implement using multiplexer, we're going to make 2 bits as selector and one of them as input but will have some logic gates before it enters multiplexer
- The dual 4-to-1 multiplexer will have 2 outputs, one for the sum (output bit), and one for the carry (which will be carry in in full adder but here will go as selector to 2nd mux)
- Since we have 8 options and want to implement a 4-to-1, we must divide the 8 options into 4 groups and bring an equation of the sum and carry in terms of the input
- Here, we'll show you how the input will enter the first multiplexer as an example:

B_0	C_{in}	A_0	S	C	
0	0	0	0	0	Group 1 $S = A_0$
0	0	1	1	0	$C = 0$
0	1	0	1	0	Group 2 $S = A_0'$
0	1	1	0	1	$C = A_0$
1	0	0	1	0	Group 3 $S = A_0'$
1	0	1	0	1	$C = A_0$
1	1	0	0	1	Group 4 $S = A_0$
1	1	1	1	1	$C = 1$

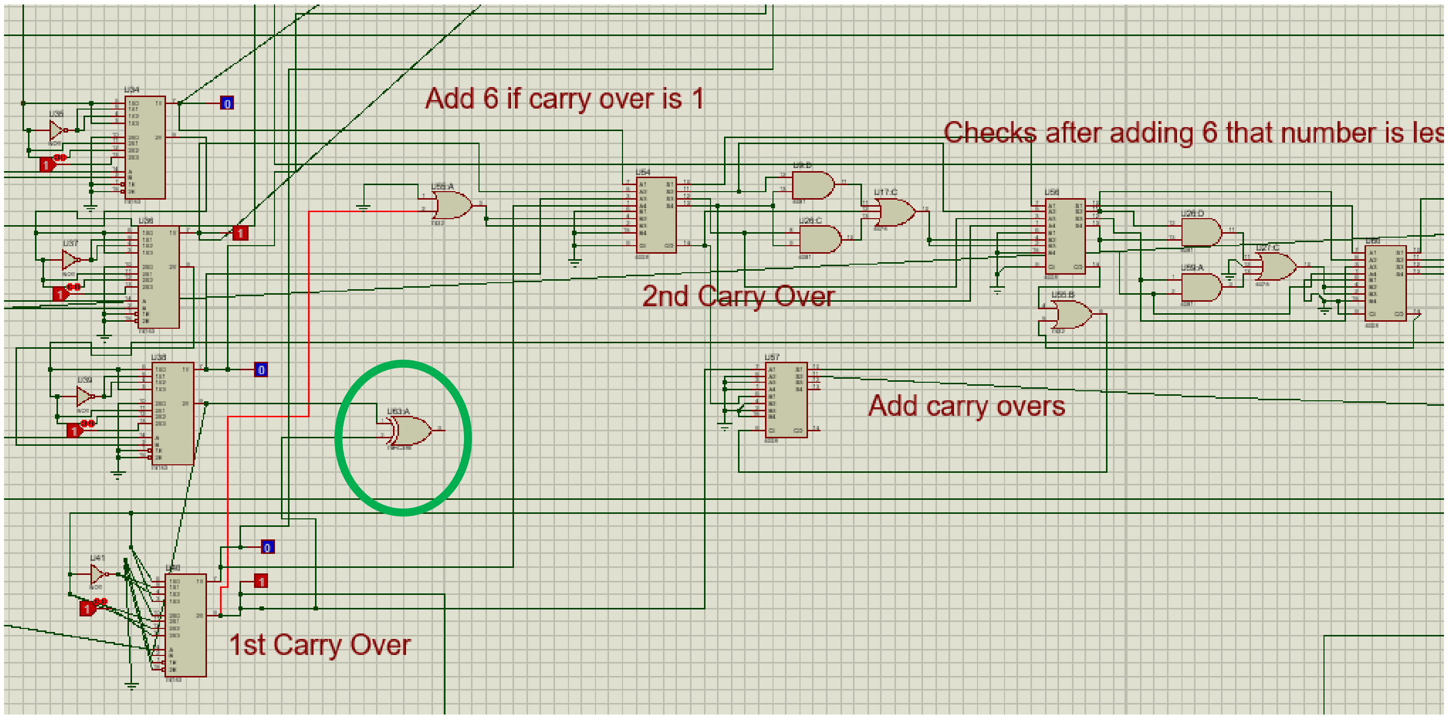
$S = \Sigma(1,2,4,7)$ $\qquad$ $C = \Sigma(3,5,6,7)$



- **Same process will be repeated for the 4 multiplexers**
- **Here's the circuit after implementation on proteus:**



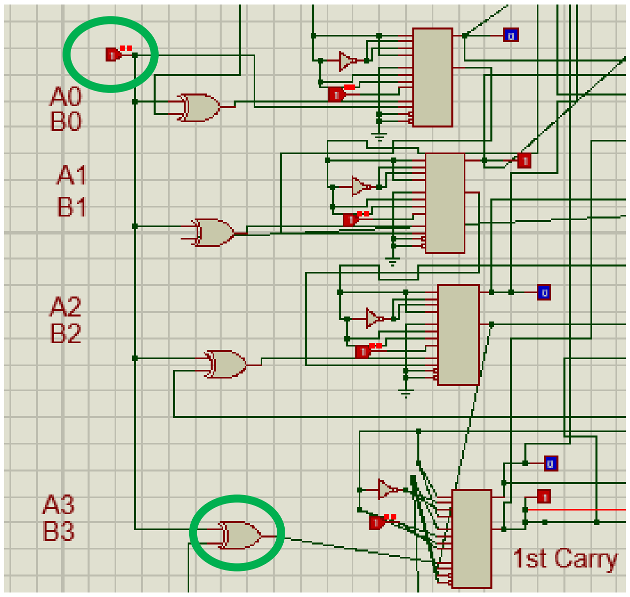
- Here, the maximum number is 30, so we’ll present this on 2 7-segment displays, and this will need 4 4-bit full adder to do this
- First adder, is to add 6 if carry over from the multiplexers is 1
- Before entering second adder, we’ll check that number is less than 9, if not we’ll add 6 in second adder
- Before entering 3rd adder, we’ll again that number is less than 9, if not we’ll add 6 in third adder
- 4th adder is to add the carry overs and then to display on 2nd 7-segment
- Here’s an image of the circuit:



This was not used (this was to detect overflow), but eventually, we didn't need to use it

5- Subtraction Operation:

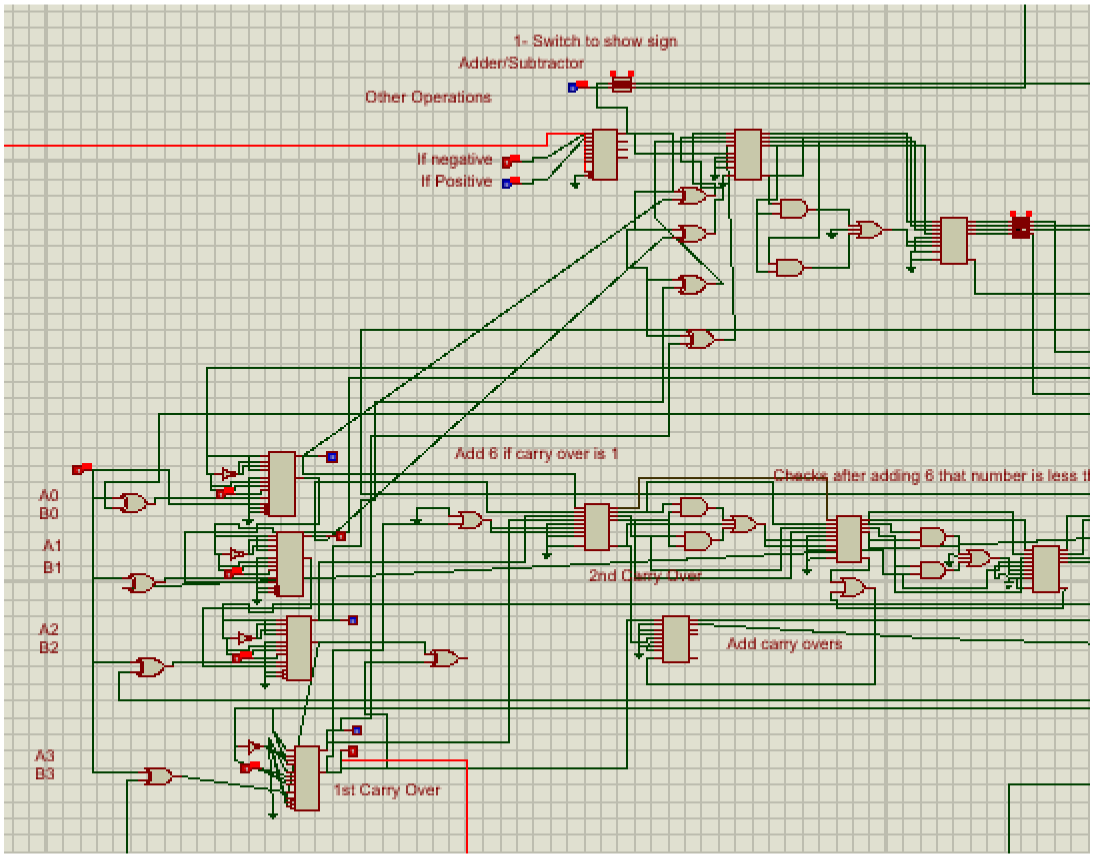
- **We're going to implement subtraction along with the addition using 4 Dual 4-to-1 Multiplexers (74153) in same circuit**
- **To subtract as we've mentioned above, we need the 2's complement, so we're going to put each input bit of B into an XOR Gate with 1 to bring the complement, and there'll be a control switch that will turn 1 when subtraction, so this will add 1 to the multiplexers**
- **This control switch will be 0 when performing addition, anything XOR with 0 is the same, and 0 will be added to the multiplexers so addition isn't affected**
- **Here's the part of the circuit we're talking about:**



Control Switch

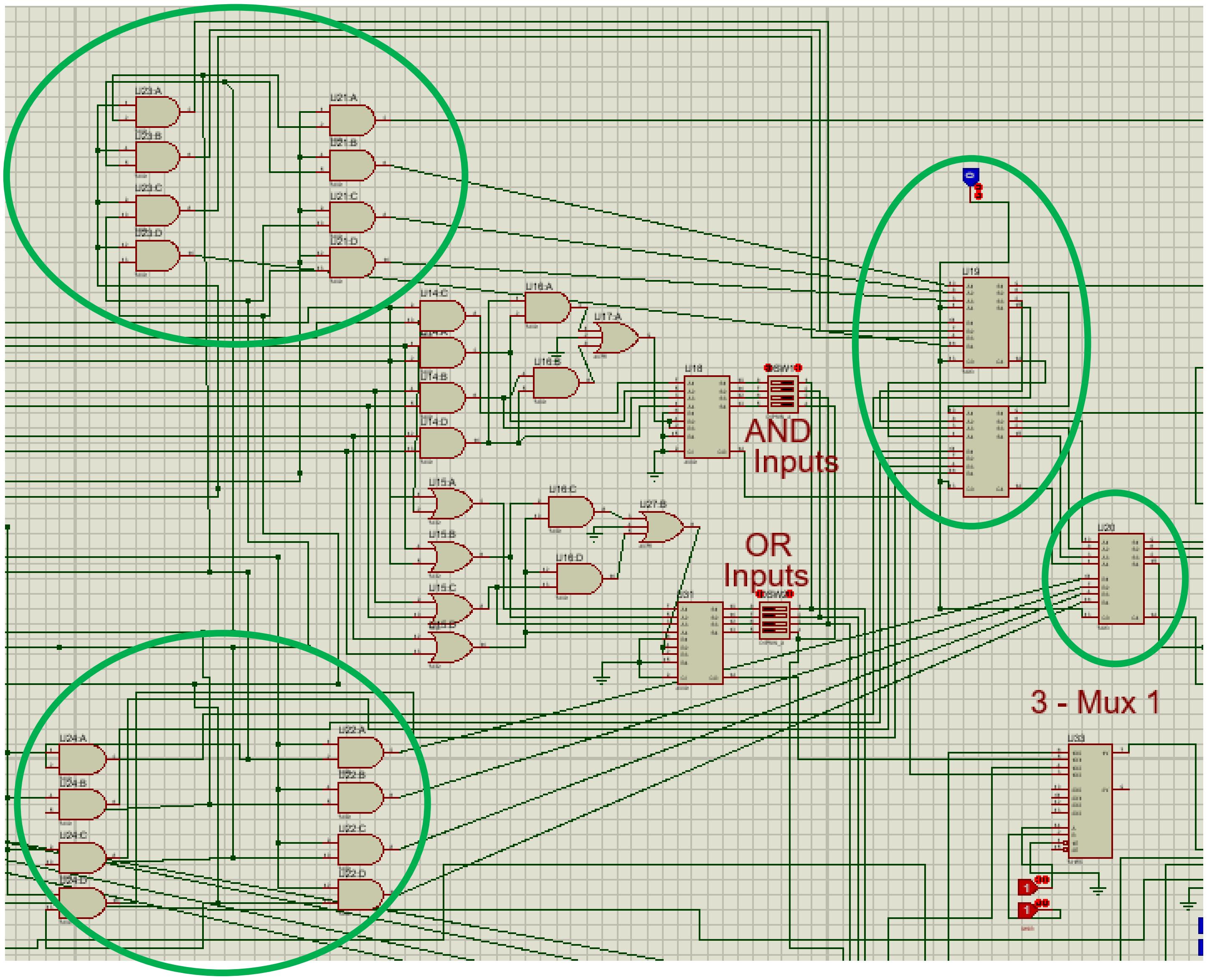
XOR Gates

- **Now as we've mentioned above as well, number maybe negative or positive, so I need a circuit to do a negative checker**
- **Instead of using magnitude comparator, we've used a Quad 2-to-1 mux (although we only need 1 2-to-1 mux) and a 4-bit adder to implement this**
- **The mux has as a selector the carry over bit, if 0, number is negative, so 1 will be passed to the adder which will add 1 to the adder after the result being XOR with 1 first, and this will be your output**
- **If 1, 0 is passed, so nothing is changed and the result will be output as it is**
- **Obviously, we'll need to check if the output is greater than 9 or not to be able to be displayed on 7-segment**
- **Moreover, the sign of the output will appear on a 3rd 7-segment, and this will be enabled by a switch (Will talk about it in details further on)**
- **Here's the circuit (Addition & Subtraction together):**

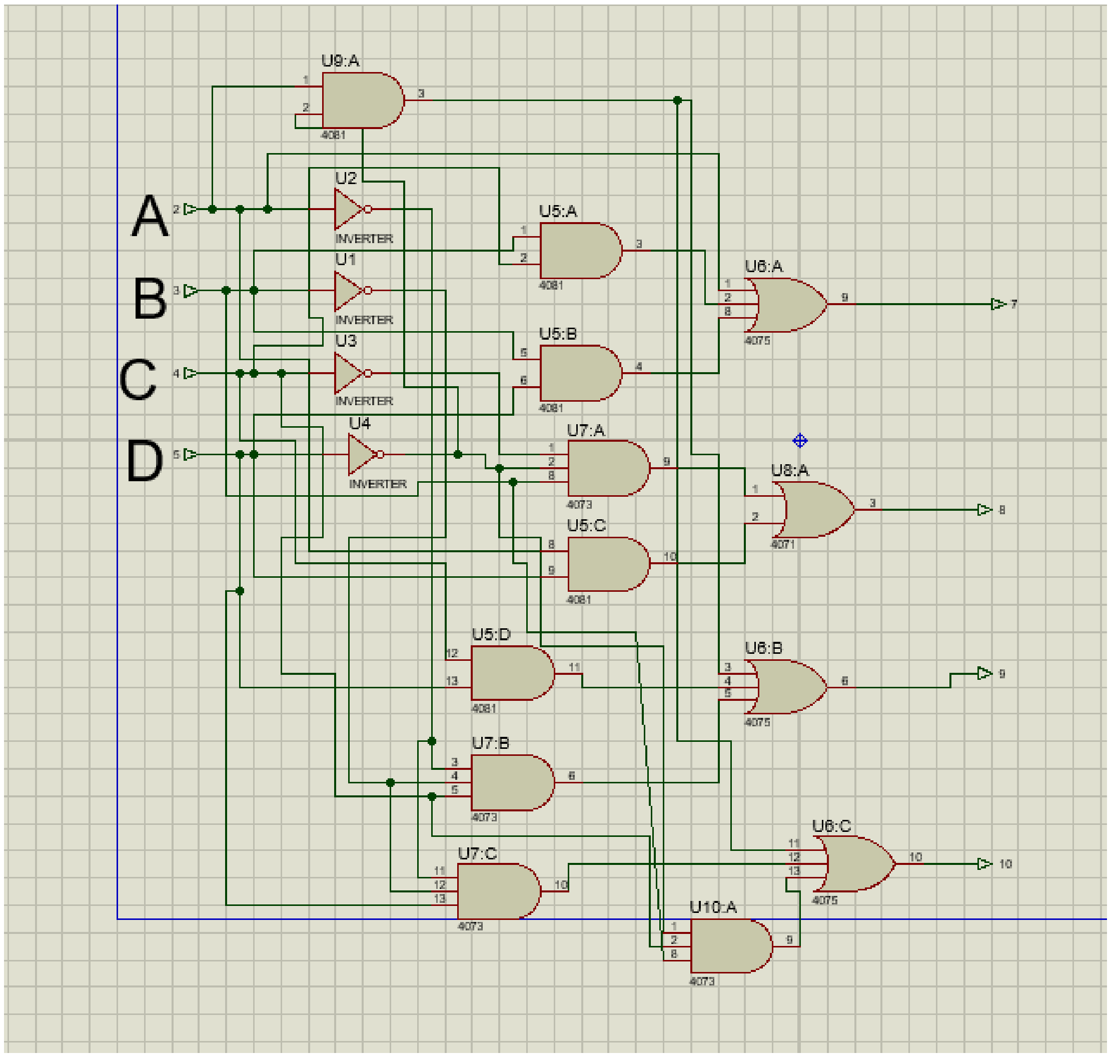


6- Multiplication Operation:

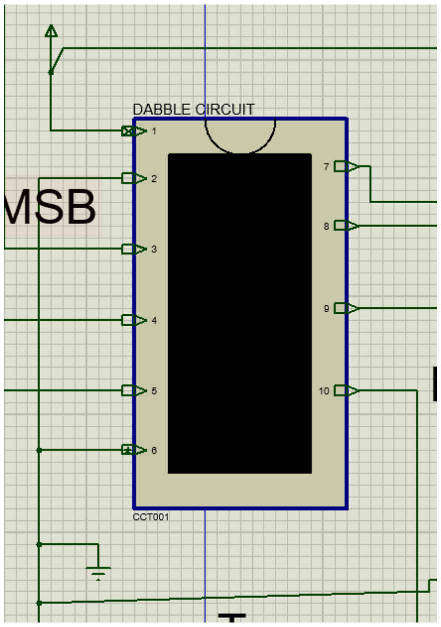
- **We're going to implement multiplication using AND ICs and 4-bit full adders (4 AND IC and 3 4-bit full adders)**
- **Here's the circuit we're talking about:**



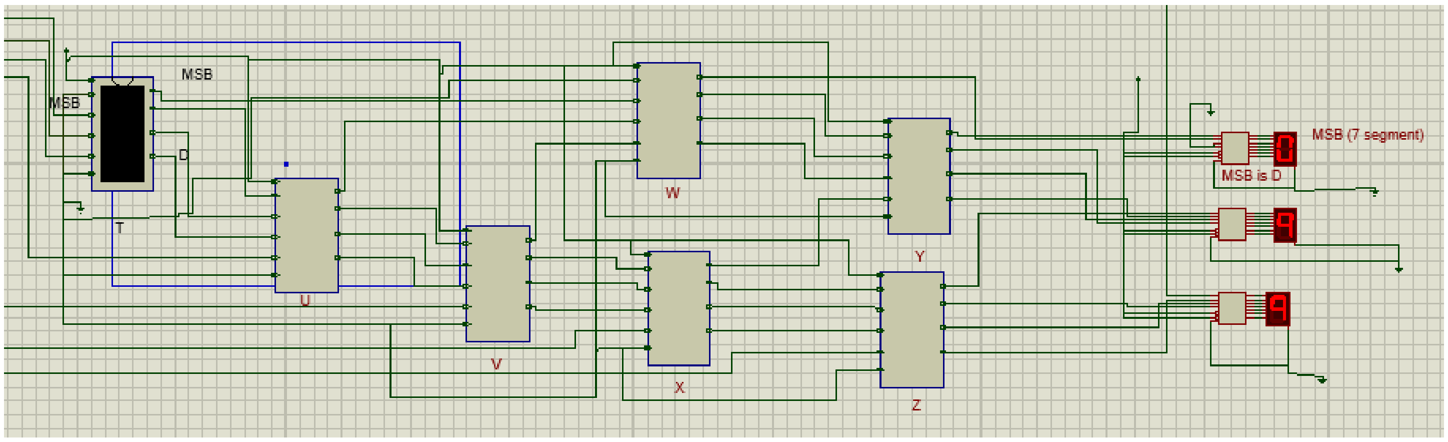
- **Now, the problem we've encountered is how we're going to display the output on 7-segment since here we need 3 of them**
- **After searching, we've found a circuit called double dabble circuit, this is a way which can be used to display a number on 3 7-segments**
- **This is the double dabble circuit:**



- **To present the number on 3 7-segments, we've found out that we need to repeat the circuit above 7 times, but to make it easier, we've created a sub-circuit (our own IC) in proteus to ease things up**
- **Here's our IC (sub-circuit):**

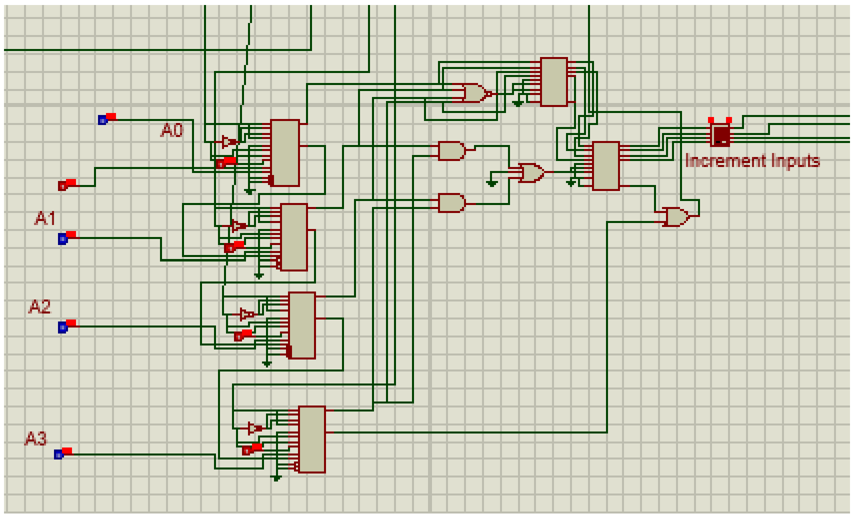


- **Now, here's the full circuit to display on 3 7-segments:**



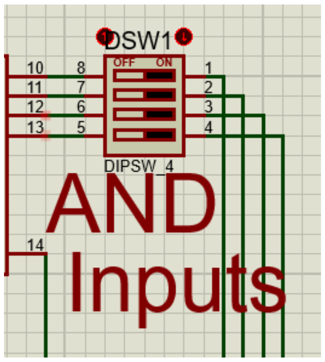
7- Increment Operation:

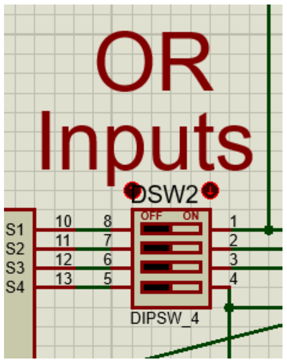
- **We're going to implement increment operation the same way as we did in addition using 4 multiplexers, but the difference is that the 2nd number won't be input B, it'll always be 1**
- **So, here's the circuit:**



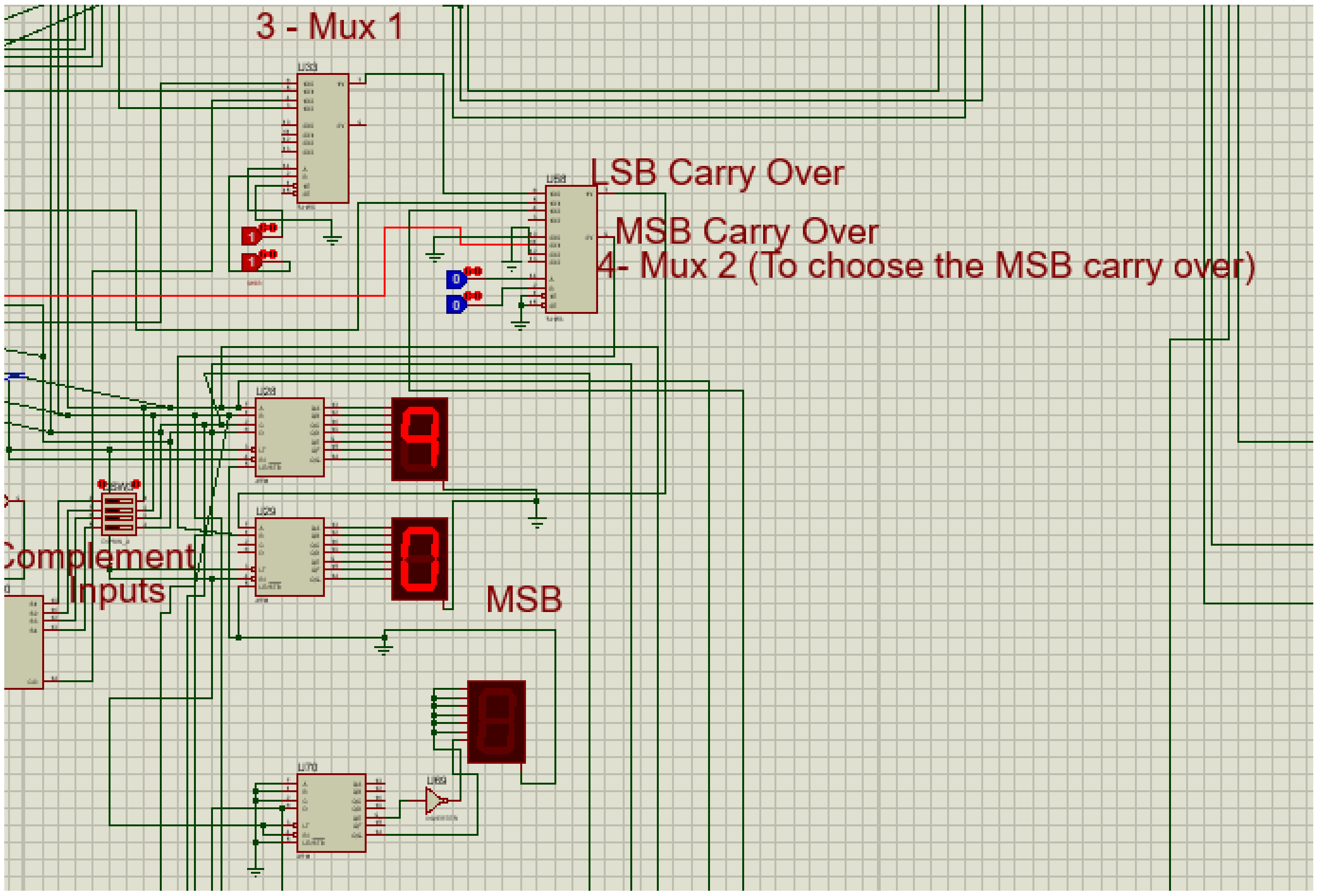
8- Putting the operations together:

- **Now, we've done every operation by itself, time to put them together**
- **We've decided that instead of controlling or selecting the output using multiplexers, we can do that using switches**

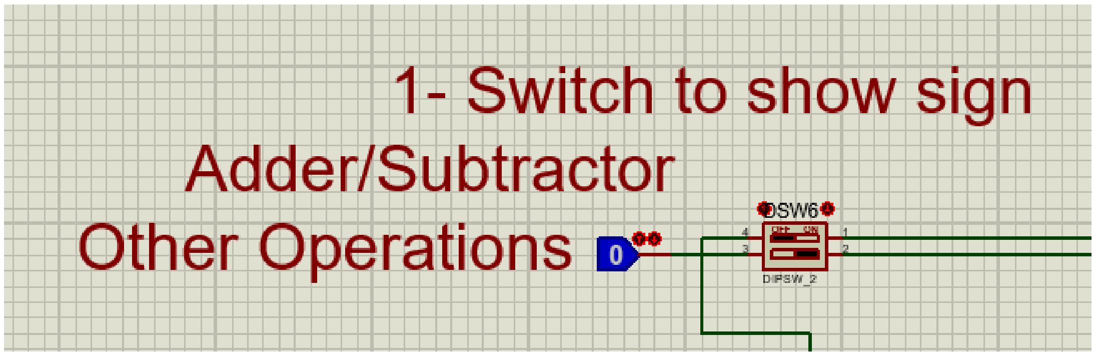




- **From each operation except multiplication, the inputs to the least significant digit will for each operation have a switch (If you want addition, turn on switch of inputs of addition & so on)**
- **Now, the problem is with the carry bit(s), to solve this we've used 3 4-to-1 multiplexers (2 74153)**
- **1st mux is to choose the carry of the following operations: increment, OR, AND, Complement (as their carry will always be 1)**
- **2nd mux (2nd 74153) will choose between the carry of above operations and adder/subtractor carry**
- **It'll have the least significant carry bits in one mux and the most significant carry bit in another mux (the 4 operations mentioned above will have the options here as 0)**
- **Here's the circuit:**



- **Now, there's one more problem which is the sign, here we've created a switch also to be able to choose the sign you want for your required operation**
- **Here's the switch:**



9- Some Notes:

① Input DIP switch

A_0	A_1	A_2	A_3	B_0	B_1	B_2	B_3
1	2	3	4	5	6	7	8

- First 4 is number A (4 is most significant)

② There's a DIP switch for each 4 bits coming out from each operation

Complement				AND				OR				Increment			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

- Here, we choose which 4 inputs that'll appear

③ Multiplexer (4-to-1)

- Multiplexer chooses between the carry outs of operation to appear

P in	0	$\rightarrow$ Carry Increment
P in	1	$\rightarrow$ Carry OR
P in	2	$\rightarrow$ Carry complement
P in	3	$\rightarrow$ carry AND

- Selectors here are going to be done with DIP switch (ON & OFF)

④ 2nd Multiplexer

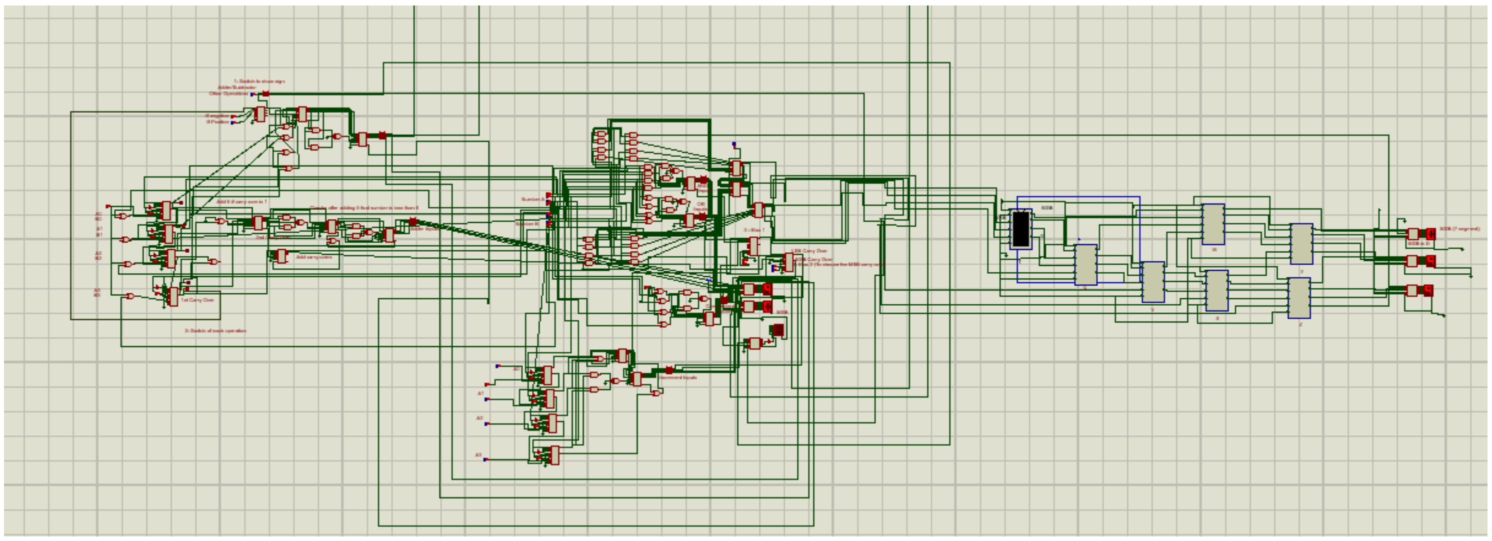
- To choose between the carries to be shown

00	$\rightarrow$ carries of OR/AND/complement/Increment
01	$\rightarrow$ Carries of Adder
10	$\rightarrow$ Carries of Subtractor

Optimization:

- For the adder/subtractor circuit, we've used leds to show the output instead of 4 4-bit full adders, OR ICs and AND ICs
- Also, the multiplier, we've used LEDs since instead of implementing the double dabble circuits which there are 7 of them and would need around 30 ICs to implement
- Furthermore, for the complement/AND/OR operations, instead of making each one of the operations go into 2 ANDs and 1 OR to see if greater than 9 or not, and then adding 6 using a full adder for each operation, we've made that only once as we've put a switch which will then move on to 1 detect greater than 9 circuit and 1 full adder instead of 1 for each operation
- For the carry bits, we didn't implement the multiplexers in our simulation, instead we used LEDs in multiplier, adder and subtractor, and we used 1 full adder to calculate the 4 inputs and the carry to go straightforward to the 7-segment (2nd 7-segment since the other 3 operations don't have a carry digit greater than 1

Software Implementation:



https://drive.google.com/file/d/1L1Dk9dYDGlhXK6tvpFIjWa2t982Nt8iu/view?usp=sharing

Hardware Implementation