Logic Project

Phase 1

Name: Basim Sherief Zeenelabdeen

ID:1210207

Name: Amira Ibrahim Ahmed Ibrahim

ID: 1210097

Name: Mohamed Nabil Elsayed

ID: 1210291

Workload distribution

Name: Basim Sherief Zeenelabdeen -1210207- Reminder

Name: Amira Ibrahim Ahmed Ibrahim- 1210097- Addition & Subtraction

Name: Mohamed Nabil Elsayed – 1210291 – Binary to BCD conversion, BCD to seven segment decoder and seven segment display

1-Remainder

First operand				Second operand			SW actual Output						Expected Output						status
sign of A	A0	A1	A2	B0	B1	B2	R2	R1	R0	sign	Div by zero flag	zero flag	R2	R1	R0	sign	Div by zero flag	zero flag	
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	right
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	right
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	right
0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	1	right
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	right
0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	right
0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	right
0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	right
0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	right
0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	right
0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	right
1	1	1	1	1	1	1	0	0	0	0 as no -zero	0	1	0	0	0	0 as no -zero	0	1	right
1	1	1	1	1	0	1	0	1	0	1	0	0	0	1	0	1	0	0	right
1	1	1	1	1	1	0	0	0	1	1	0	0	0	0	1	1	0	0	right
1	1	1	1	0	1	1	0	0	1	1	0	0	0	0	1	1	0	0	right
0	1	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	right
0	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	right
0	1	1	1	0	1	0	0	0	1	1	0	0	0	0	1	1	0	0	right
1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	1	1	0	0	right
1	0	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	1	right

Description and logic of remainder:

I have divided the cases into 2 sections

First section if A>=<B except if B ==2,1,0 will refer to that in second section

If A>=<B I check if A>2B or A>B if A>2B then I subtract A-2B and that would be the result ;since the largest number is 7 and lowest is 3 so 7-6=1 and that the greatest gap if A>B not 2B ,notice (I used 4 bit comparator and just shifted one bit to the right to get 2B), if A>B not 2B then A-B directly will give result for example 7>4 but 7 not greater than 8 so 7-4 will give reminder directly

Second section if B==2,1,0

If B=2 A%2 then its equal to **A and 1** it's a special property that always gets result right

If B=1 then that would always give zero anything is divisible by 1 so no reminder

If B=0 needless to say you cannot divide by zero so no reminder and div by zero flag should be 1 with output equal zero.

That’s the summary. I used MUXs to handle and merge the 4 cases together using 4x1 mux. Also, to avoid negative zero as a result (R0+R1+R2) AND sign of A

2-Adder &subtractor

Addition:

First operand				**Second operand**				**Actual Output**					**Expected Output**					**Status**
sign	A2	A1	A0	sign	B2	B1	B0	sign	R3	R2	R1	R0	sign	R3	R2	R1	R0	
0	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	Right
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	Right
0	0	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1	1	Right
1	1	0	1	0	1	1	0	0	0	0	0	1	0	0	0	0	1	Right
1	1	1	0	0	0	1	1	1	0	0	1	1	1	0	0	1	1	Right

Subtraction:

First operand				**Second operand**				**Actual Output**					**Expected Output**					**Status**
sign	A2	A1	A0	sign	B2	B1	B0	sign	R3	R2	R1	R0	sign	R3	R2	R1	R0	
0	1	0	1	1	1	0	1	0	1	0	1	0	0	1	0	1	0	Right
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	Right
0	0	1	1	0	1	0	0	1	0	0	0	1	1	0	0	0	1	Right
1	1	0	1	0	1	1	0	1	1	0	0	0	1	1	0	0	0	Right
1	1	1	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	Right

Description and logic of the Addition and Subtraction Operations:

I have separated the magnitude bits and the sign bit, then I managed to build a circuit that calculate two things A+B as a magnitude with all it’s possible cases and A-B with all it’s possible cases (if A>B & if B>A & if B=A). Then I inserted the outputs to 2x1 Multiplexers to select the result I want depending on the sign bit of A & B.

For the sign bit I had to build a 3-bit comparator to compare between the magnitude of A & B together to be able to decide the value of the sign bit I have made a truth table and implemented it using Mux.

3-Binary to BCD & 7 segment

Since the team consists of 3 members only, our calculator does not have the multiplication section, hence the biggest number we have is +7 + 7 = +14 (0111 + 0111 = 01110), and therefore we had only 4 binary bits in our calculator. As a result, the tens section will either have 1 or 0. After these conclusions, I have designed a circuit that converts 4-bit binary number into BCD number, but for numbers from 10 to 14 (1010 to 1110 in binary) will have a BCD number different from their binary numbers (they will have a fifth BCD bit). After that, we now have BCD numbers, and we want to decode them then display them on seven segment display. Then I used three 7448 ICs for sign bit, tens, and units in BCD numbers to display them on seven segment LEDs. Finally, the “div by zero” flag, “zero” flag and “sign” flag which are outputs of the ALU made, I linked each output to a LED to illuminate if the output is 1 .

Truth tables for this part:

Binary to BCD

A0	A1	A2	A3	T0	U3	U2	U1	U0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	0	0	1	0	0
0	0	1	1	1	0	0	1	0
0	1	0	0	0	0	0	1	0
0	1	0	1	1	0	0	0	0
0	1	1	0	0	0	1	1	0
0	1	1	1	1	0	1	0	0
1	0	0	0	0	0	0	0	1
1	0	0	1	0	1	0	0	1
1	0	1	0	0	0	1	0	1
1	0	1	1	1	0	0	1	1
1	1	0	0	0	0	0	1	1
1	1	0	1	1	0	0	0	1
1	1	1	0	0	0	1	1	1
1	1	1	1	1	0	1	0	1