Q2.

Nibble Subcirtut: In the nibble sub-circuit I am using and gates between the read_write signal and the data input, so the register values will only change if the corresponding data bit is 1 and the read_write is 1 as well. And I am anding the and value of the read_write and enable with the clock input, so the clock tick will change for the registers only if the read_write and enable are both 1.

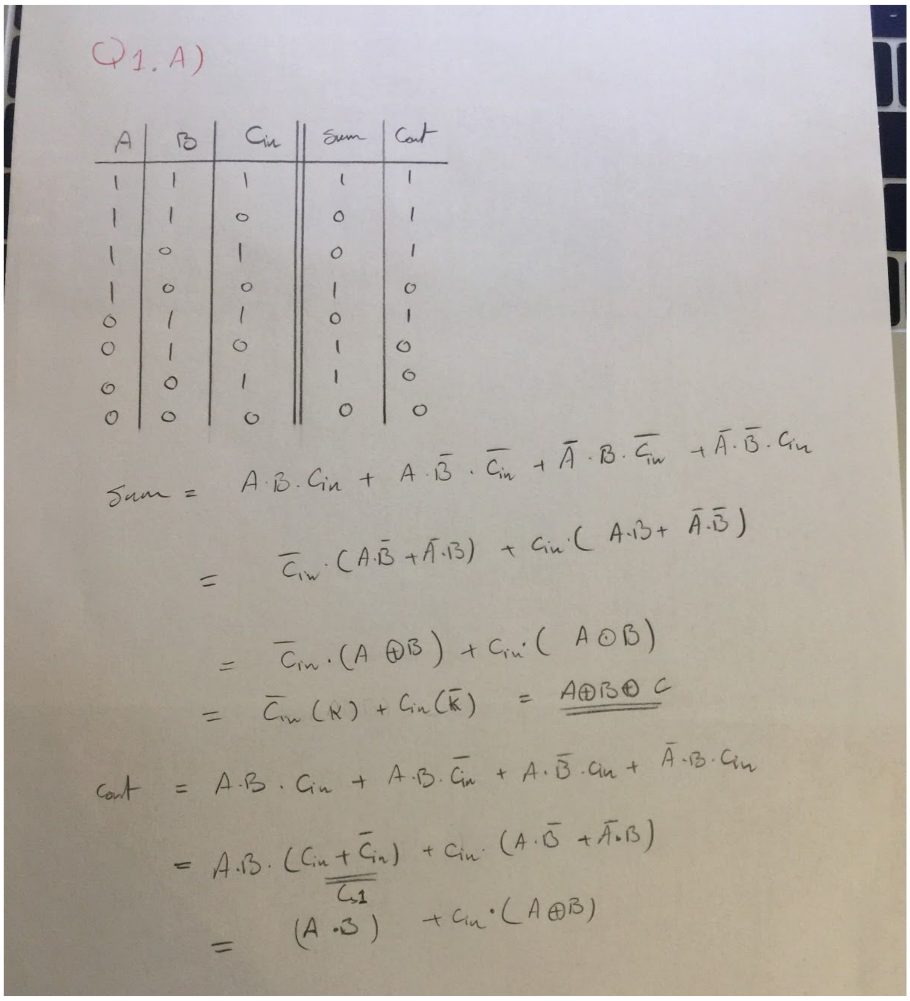
Multiplexor Subcirtut: I am passing the address and data values to the subcircuit and passing the values to the corresponding address output. Whichever address that is getting a value will have an enable of 1, we are going to pass this enable value to the nibble in the ram circuit.

Mask Subcircut: To stop the output values of the nibbles to get mixed with each other I have created a bitmask that will help us pass out only the bits from the selected nibble. I am using this after each nibble in the 16 Nibble Ram.

16 Nibble Ram Circuit: I am passing the address and data bits to the multiplexor and giving the output value to the data input of all the 16 nibbles. I have connected the clock and read_write to all the nibbles. I have designed the multiplexor in a way that only the nibble that the address indicates get the data values and the amenable set to 1. I have attached a mask after each nibble that will make sure we are only passing through the bits of the selected nibble. I am orring all the values of the nibble outs and passing the result to the Ram Out.

Q1.

1-bit Adder: I have designed a simple 1-bit adder, using the lecture slides. You can see my calculations below:



Q1. A)

A	B	C_{in}	Sum	Cout
1	1	1	1	1
1	1	0	0	1
1	0	1	0	1
1	0	0	1	0
0	1	1	0	1
0	1	0	1	0
0	0	1	1	0
0	0	0	0	0

$$Sum = A \cdot B \cdot C_{in} + A \cdot \bar{B} \cdot \overline{C_{in}} + \bar{A} \cdot B \cdot \overline{C_{in}} + \bar{A} \cdot \bar{B} \cdot C_{in}$$

$$= \overline{C_{in}} \cdot (A \cdot \bar{B} + \bar{A} \cdot B) + C_{in} \cdot (A \cdot B + \bar{A} \cdot \bar{B})$$

$$= \overline{C_{in}} \cdot (A \oplus B) + C_{in} \cdot (A \odot B)$$

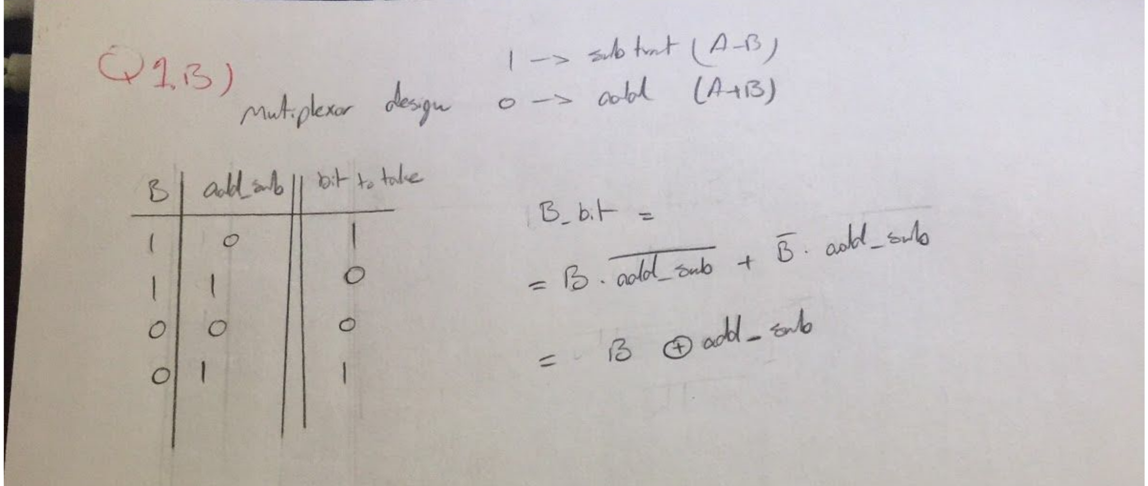
$$= \overline{C_{in}} (K) + C_{in} (\bar{K}) = \underline{\underline{A \oplus B \oplus C}}$$

$$Cout = A \cdot B \cdot C_{in} + A \cdot B \cdot \overline{C_{in}} + A \cdot \bar{B} \cdot C_{in} + \bar{A} \cdot B \cdot C_{in}$$

$$= A \cdot B \cdot \underbrace{(C_{in} + \overline{C_{in}})}_{=1} + C_{in} \cdot (A \cdot \bar{B} + \bar{A} \cdot B)$$

$$= (A \cdot B) + C_{in} \cdot (A \oplus B)$$

4-bit adder: I am passing the B values and the add_sub bit through a XOR gate and to the adder, you can see my calculations below. I am adding the A bits directly to the adder. I am rippling the count bits through the adders and using the last one to determine the overflow. And I am using OR gates between all the counts to determine the zero value.



Q1. B)

Mutiplexor design

1 -> subtract (A-B)
0 -> add (A+B)

B	add_sub	bit to take
1	0	1
1	1	0
0	0	0
0	1	1

$$B\_bit =$$

$$= B \cdot \overline{add\_sub} + \bar{B} \cdot add\_sub$$

$$= B \oplus add\_sub$$