North South University

Department of Electrical & Computer Engineering

Lab Report

Experiment No:	**02**
Experiment Title:	**Design of a 4-bit by 4-bit Binary Multiplication Unit**
Course Code:	**CSE332L**
Course Name:	**Computer Organization & Architecture Lab**
Name & ID:	**Ishrat Jahan,1921909042**
Date of Experiment:	**04/11/2021**
Date of Submission:	**06/11/2021**

Objective:

1. To construct a 4-bit by 4-bit Binary Multiplication Unit.

2. Understanding how the Binary Multiplication Procedure works.

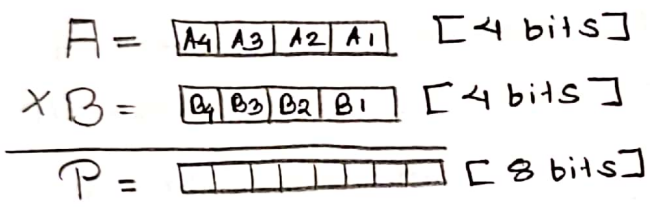
3. Labelling the inputs, outputs and correct selection of the Adder.

4. Adjusting the input's and output's bits, splitter's fan out and bit width as well as the Adder's data bit's, for proper functioning of the 4-bit Binary Multiplication Unit.

5. Check Multiplying bits and show the sum outputs.

Theory:

4-bit by 4-bit Multiplication Unit: It carries out the operation between two 4-bit number. The operation carried out is multiplication. It is regarded as "Unit" because when we design them once, we will be using them as sub-circuits for other different operations.



A and B are known as operands and each of them are of 4-bits. When we multiply two of them, the Product we will get will be m+n bits; where m = bits of A and n = bits of B.

Therefore, for 4 bit by 4-bit, the Product is 4+4 = 8 bits. When we get any such product's bits by adding the bits of operands, the max bit is usually the (m+n) bit and (m+n-1) bit will be the min bit. because the extra bit is the carry.

For example: When two 4-bits numbers are multiplied, we get the max bit as 8 bit whereas the min bit is 7 bit. The 8th bit is the carry.

The overall process is conducted for unsigned binary numbers.

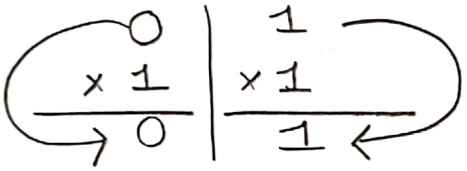
Multiplicand and Multiplier: The number to be multiplied is the Multiplicand. The number with which we multiply is called the Multiplier.

Example:

$$\begin{array}{r} 245 \longrightarrow \text{Multiplicand} \\ \times\ 11 \longrightarrow \text{Multiplier.} \\ \hline 245 \\ 245\,X \\ \hline \end{array}$$

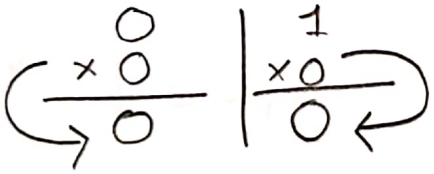
For binary multiplication, the intermediate products are simple as they are either multiplied by 0 or 1.

When multiplier bit = 1



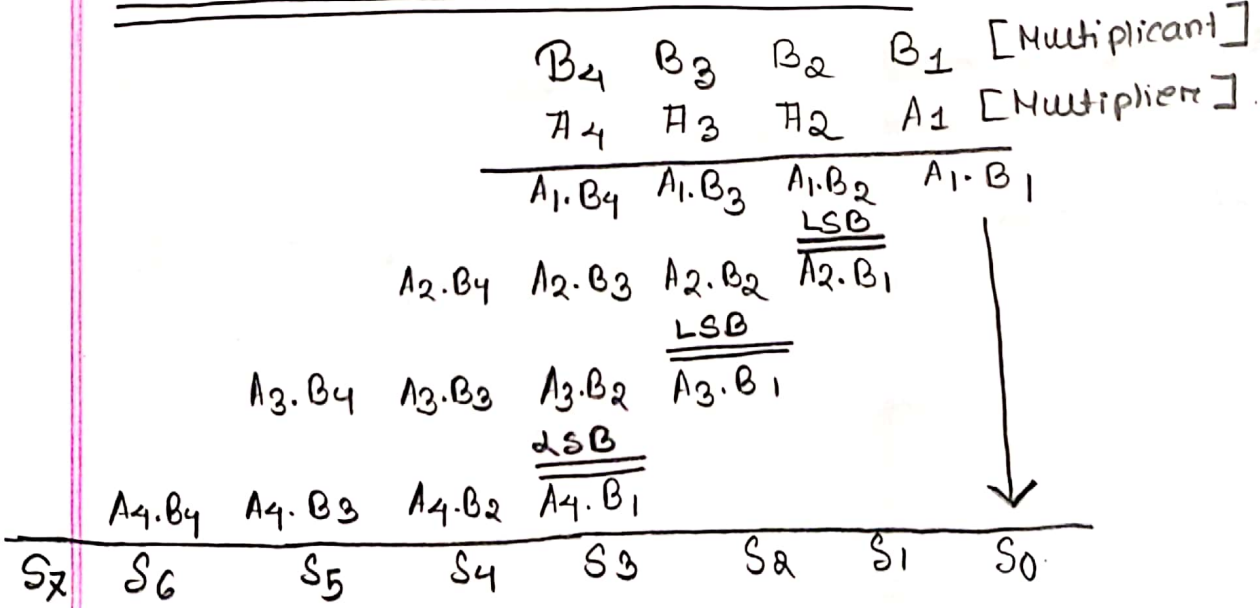
The product is the copy of the multiplicand.

When multiplier bit = 0



The product is always 0.

Multiplying Procedure and Adder:



The operation between $A_1 \cdot B_1$, $A_1 \cdot B_2$ ... is AND procedure, and A_1 and B_1, B_2 .... are all 1 bit there multiplying the multiplicand with the first multiplier will result in 1 bit output. This is the case between all the multiplicand and multiplier multiplication. As a result, we use an AND gate of 1 bit. for all these process.

After getting the product, they are all added using the Adder. We are using adder because we have carry, otherwise we would use OR gate.

$C_3\ C_2\ C_1\ C_0$ ← Carry in

A = | 4 | 3 | 2 | 1 |

B = | 4 | 3 | 2 | 1 |

S = C_4 | 4 | 3 | 2 | 1 |

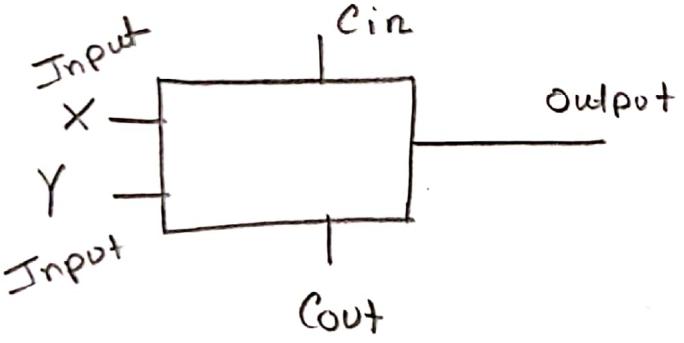


Fig: Adder

Wire Colors: Red: It arises because of conflicting values on the wire.

Blue: When the wire is not connected to any input or output value.

Orange: Incompatible data bits.

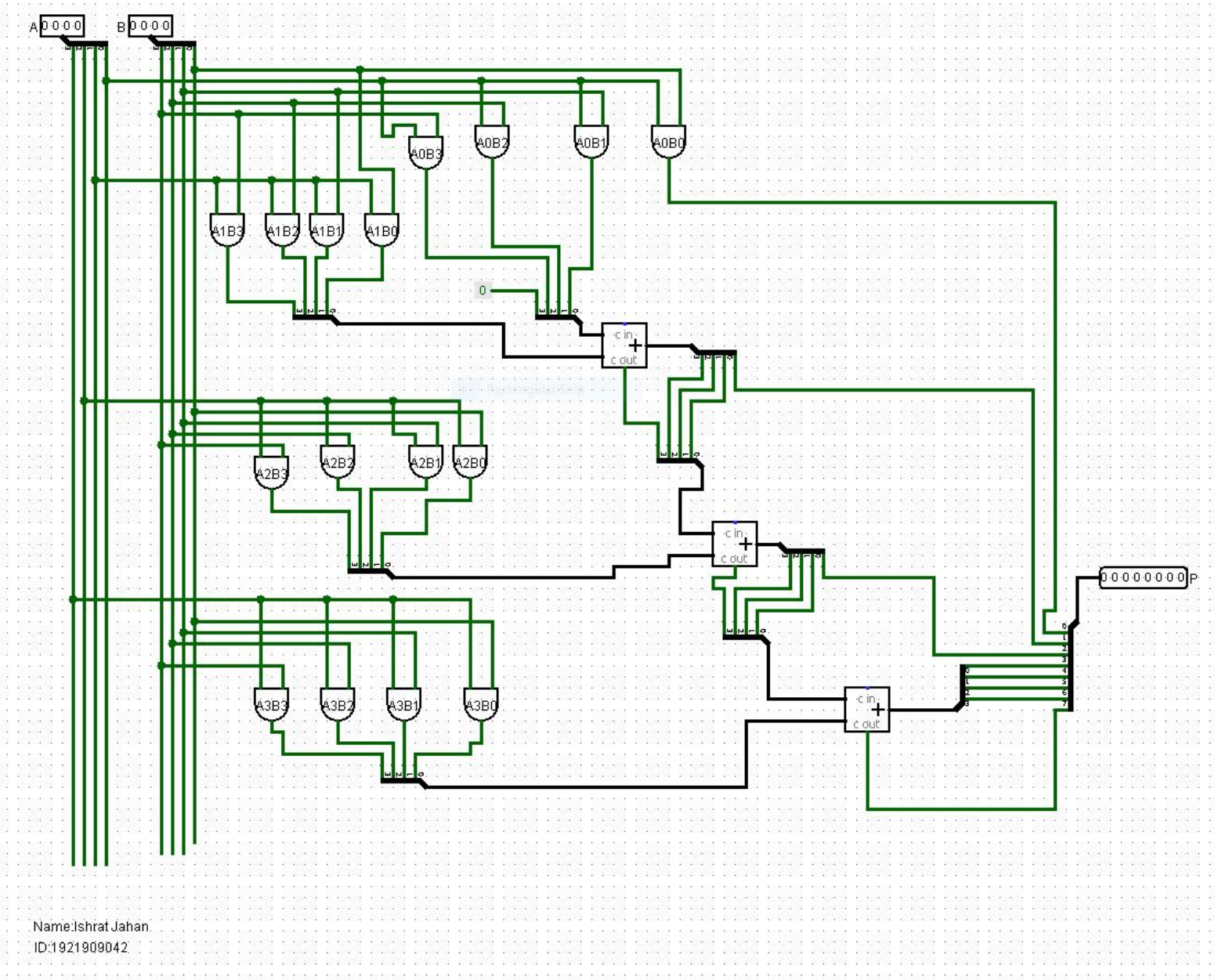
Multiplication Hardware IC: For the multiplication product of the Multiplicand with the Multiplier it is denoted as:-

$$A_1 \cdot B_1 = \Sigma 1$$

$$A_2 \cdot B_2 = \Sigma 2$$

$$A_3 \cdot B_3 = \Sigma 3$$

$$A_4 \cdot B_4 = \Sigma 4.$$



Name:Ishrat Jahan
ID:1921909042

Table: 1 Theoretical.

Multiplicand				Multiplier				Product								Result in Decimal.
B4	B3	B2	B1	A4	A3	A2	A1	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	
1	0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	8x9=72.
0	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	5x2=10.
0	1	1	1	0	0	1	1	0	0	0	1	0	1	0	1	7x3=21
0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	4x8=32
0	1	0	1	0	1	1	0	0	0	0	1	1	1	1	0	5x6=30
1	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	9x4=36
1	1	1	1	1	0	1	1	1	0	1	0	0	1	0	1	15x11=165

Table: 2 Experimental.

Multiplicand				Multiplier				Product								Result in Decimal.
B_4	B_3	B_2	B_1	A_4	A_3	A_2	A_1	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	
1	0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	8x9=72
0	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	5x2=10
0	0	1	1	0	1	1	1	0	0	0	1	0	1	0	1	7x3=21
0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	4x8=32
0	1	0	1	0	1	1	0	0	0	0	1	1	1	1	0	5x6=30
1	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	9x4=36
1	1	1	1	1	0	1	1	1	0	1	0	0	1	0	1	15x11=165

DISCUSSION:

The experiment of lab02 was based on "Design of a 4-bit by 4-bit Binary Multiplication Unit". In this lab we learned how to design a multiplication unit to multiply two 4-bits operand in logisim.

At first, we studied the concept of 4 bit- by 4-bit multiplication unit. We learnt that, it will multiply two unsigned binary numbers and it is regarded as "Unit" because of it's use as subcircuit in other circuits. Then we learnt, how to know what will be the output bit by using (m+n) formula where m and n are the bit's of the operands. The result we get will be known as maximum bit as there will be a carry in the product. Therefore the minimum bit will be (m+n-1) bits.

Secondly, we proceeded to learn what is multiplicand and multiplier. The number we are multiplying is the multiplicand and the number we are multiplying it with is the multiplier. For example: Multiply 3 by 2; Here 3 is the multiplicand and 2 is the multiplier. Moreover, we learnt how binary multiplication results in simple intermediate products. For example, if the multiplier is 1, then the product is the multiplicand and if the multiplier is 0, then the product is 0.

After that we learnt how the multiplication of the two 4-bit operands will work. While multiplying, we first multiply the multiplicand with the least significant bit of the multiplier. For example:- The multiplicand is $(B_3 B_2 B_1 B_0)$ and multiplier is $(A_3 A_2 A_1 A_0)$. For the first, step, we get $(A_0B_0, A_0B_1, A_0B_2, A_0B_3)$. as product. From here, the A_0B_0 gets directly into the final product and the LSB shifts to the left. Therefore, A_0B_1

becomes the LSB. Then we do the same procedure for multiplying the multiplicand and multiplier, everytime the LSB keeps shifting to the left. After that we add the products of each step using Adders and eventually with the last adder we get the final output. For example, $A_0B_3, A_0B_2, A_0B_1, A_0B_0$ is the first product from which A_0B_0 gets into the final product directly. Then we get second product $A_1B_3, A_1B_2, A_1B_1, A_1B_0$ which we add with the first product using Adder. In this way, the whole procedure is carried out till the most significant bit is multiplied with the multiplicand. Following this, we used our theoretical knowledge to complete Table:1.

Then we proceeded to design the circuit in Logisim. At first, I constructed the 4 bits input. For that I took a input pin and changed it's data bits to 4 and labelled it as 'A'. I did the same thing for the second input 'B'. Then I took two splitter from the wiring section, changed it's fan out and bit width to 4, changed it's facing and connected it to the both inputs. The 0 part in each splitter stands for A_0 for A and B_0 for B. Likewise the splitter connected to A input has $(A_0A_1A_2A_3)$ and the splitter connected to B has $(B_0B_1B_2B_3)$.

Secondly, from the gates section I took an AND gate, changed the bits to 1, inputs to 2 and labelled it as A_0B_0. As the multiplication between the multiplicand $(B_3B_2B_1B_0)$ and the LSB of the multiplier (A_0) is an AND process we are using AND gate. As each of A_0, B_0 is 1 bit, we kept the data bit to 1. Then I copied and pasted the AND gates three more time for the remaining operations (A_0B_1, A_0B_2, A_0B_3). I took the inputs according to the gates labelled, and connected each of them with its AND gate. As A_0B_0

will be directly into final product, so I drew the output line from the labelled 'A_0B_0' AND gate and kept it aside. The rest other output was connected to a splitter. As A_0B_1 was the LSB, it was connected with the splitter's 0 part and other were connected to 1 and 2. For the splitter's 3 part, I took the constant pin from the wiring section, changed it's value to 0x0 and connected with with the splitter. The first product of multiplicand x lsb of multiplier was completed. I repeated the same steps for multiplying (A_1B_0, A_1B_1, A_1B_2 and A_1B_3) using AND gates just like before. and all the outputs from these gates were as well connected to ~~th~~ another splitter I took from the wiring section. The final part of this step was to add the both products (A_0B_1, A_0B_2, A_0B_3) and (A_1B_0, A_1B_2, A_1B_2, A_1B_3). For that, I took an 'Adder' from the arithmatic section, changed it's data bits to 4, same as the splitters. Then I connected both the splitters to the Adder's input. The output we will get from this 'First Adder' will be the, one of the input in the 'Second Adder'. Therefore, the splitter was connected to the Adder's ~~part~~ output. The splitter's 0 part was left unattended as it calculates (A_0B_1 + A_1B_0) and the rest other including the Cout was connected to another splitter. This will be used as input for the second adder.

I carried out the same steps using AND gates like before for calculating (A_2B_0, A_2B_1, A_2B_2, A_2B_3). There output's were as well connected to the splitter.

The splitter's output along with the 'First Adder's' splitter output was connected to a second Adder's input. I repeated the same steps with the second Adder's output just what I did with the first Adder. except I left the 0 part untouched.

The last step was multiplying (A_3B_0, A_3B_1, A_3B_2, A_3B_3) which I did in a similar way like the previous ones. The last four AND gates output's were connected to the splitter and along with the 'Second Adder's' output, I input-ted them in the Third Adder's input section.

For displaying the final output, I knew it will be 8 bits, so I took a output pin, changed it's data bit's to 8. As it will display 8 bits, therefore I connected a splitter with fan out and bit-width changed to 8, with it. The A_0B_0 output from the AND gate was connected with the 0 part, the output from the 'First Adder' whose splitter's 0 part was left unattended, was connected with the Output splitter's 1 part. Likewise, the 0 part from 'Second Adder' was connected with the 3. From the 'Third Adder' the 4 bit's output were connected to 4, 5, 6 accordingly and it's Cout with the splitter's 7 section. The whole circuit was completed.

Finally, after completion, I checked the outputs and matched them with the Table: 1 and filled out the Table: 2. As all of the values matched, the circuit worked correctly.

The problem that I faced during this experiment was that, I kept forgetting to label the AND gates. As there was a lot of operations and circuit's involved, correcting it became confusing if I made a small mistake even once. Other than that is it is built step by step slowly, the circuit can be constructed error free.