Name	**Muhammad Asad**
Reg. #	**2019-EE-~~383~~**
Marks	3.5 12-11-20

Experiment # 5

To Understand Combinational Circuit and Applications

Objective:

- *To Design* and function of 1-bit half adder &1-bit full adder
- Parallel 4-bit adding circuit, Serial 4-bit adding circuit

Apparatus:

Workstation Core 2 Duo, Uni*Tr@inLucas Nulle*, Card, Jumpers wires, Power Supplies.

Theory:

- ***Adding circuits***

In arithmetic, we are familiar with the addition of decimal numbers from **0** to **9**. A number must be carried when the addition is greater than **9**. This example shows the process of adding two 4-digit decimal numbers.

```
   7 4 6 0
                ← 1st addend
+  3 5 7 5      ← 2nd addend
   1 1 1        ← Carry
-------------   ← Total
 1 1 0 3 5
```

Binary numbers only contain the digits **0** und **1** and one must be carried when the sum is greater than **1**.

Decimal		*Binary*
6	→	1 1 0
+ 7	→	+ 1 1 1
		1 1
13	→	**1 1 0 1**

The **rule of addition** is therefore:

Y	**X**	**S1**	**C1**
0 +	0 =	0	0
0 +	1 =	**1**	0
1 +	0 =	**1**	0
1 +	1 =	0	**1**

S1: Sum **C1**: Carry

Adding circuits are constructed to realize the rules of addition. They must be able to add and to accommodate carrying. The combination of these two elements produces a half adder.

A half adder can add two individual binary numbers but it does not take into account any carry from a preceding stage. A binary adder with a third input for the carry from a previous stage is termed a **full adder**.

The following truth table including the carry from a previous stage applies:

Carry from preceding stage
Addend
Addend

↓ ↓ ↓

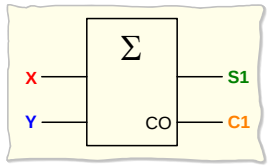
C_{n-1}	Y	X	S	C_n
0	0	0	0	0
0	0	1	**1**	0
0	1	0	**1**	0
0	1	1	0	**1**
1	0	0	**1**	0
1	0	1	0	**1**
1	1	0	0	**1**
1	1	1	**1**	**1**
			↑	↑

Total

Carry

Half adder

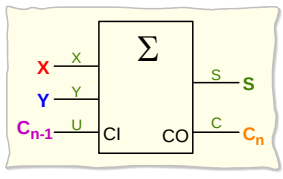
This experiment studies the basic functionality of a 1-bit half adder. The 1-bit half adder has two inputs and two outputs.



Symbol for a 1-bit half adder.

Full adder

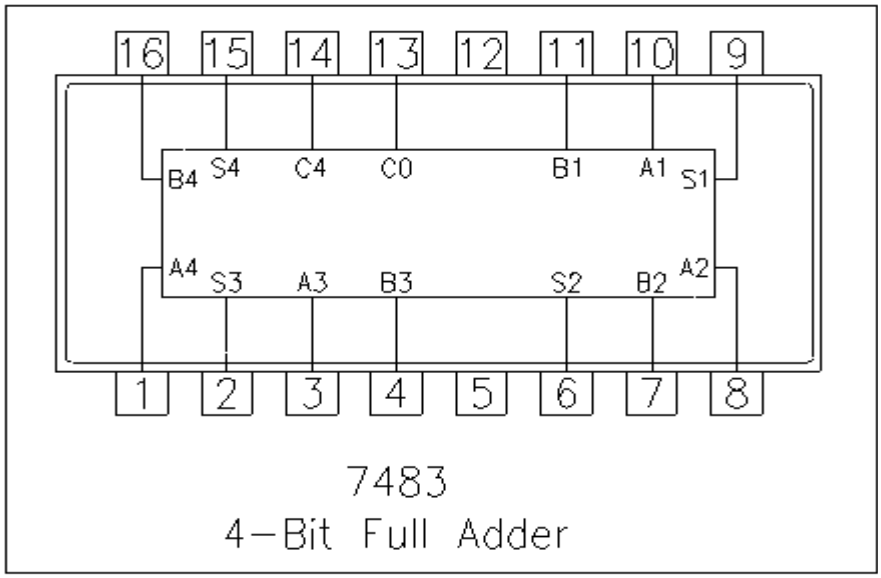
This experiment studies the behavior of a 1-bit full adder. The **full adder** extends the half adder and has 3 inputs instead of the half adder's 2. The third input allows the circuit to take into account a carry from a previous stage of addition.

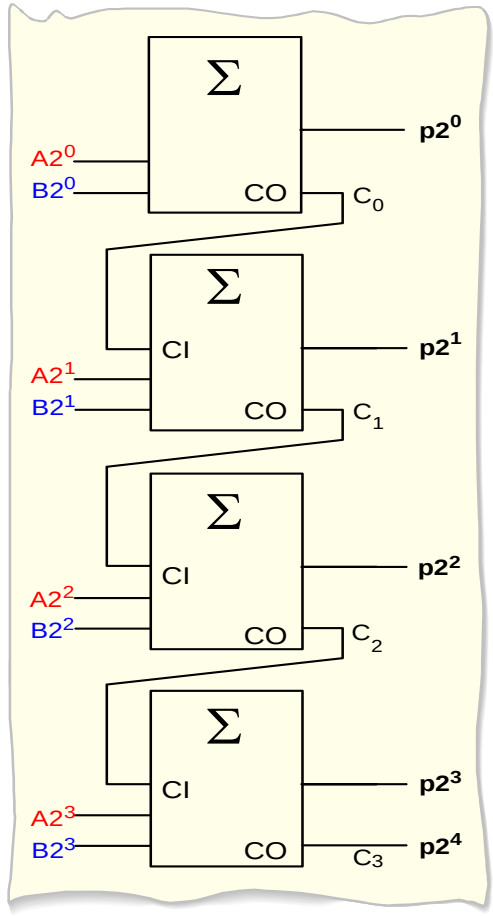


Symbol for a 1-bit full adder

Parallel Adder

If a full adder is used for each of the bits (actually a half adder would suffice for the LSB since there is no previous stage to produce a carry), addition of all the individual bits can proceed in parallel.

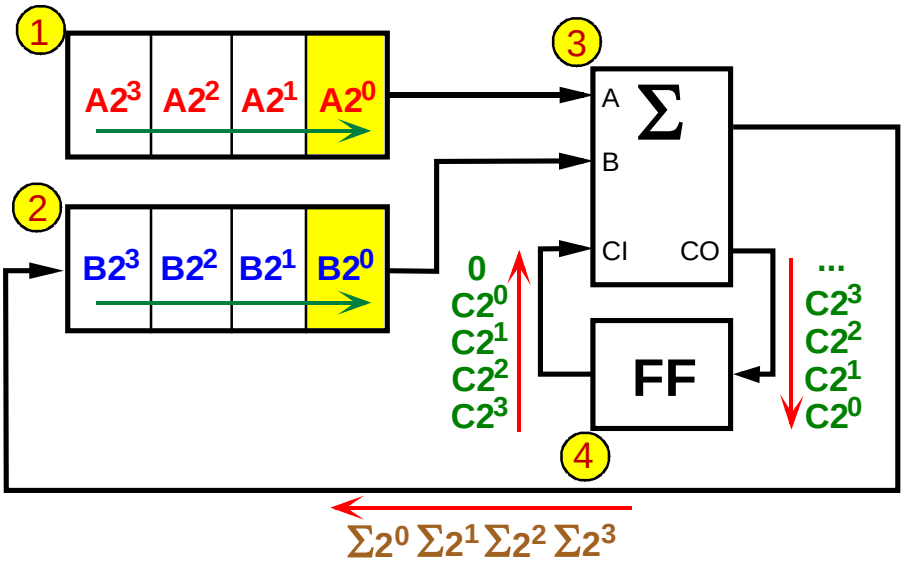




If more than four bits are to be added, several 4-bit adders can be coupled together (cascaded).

Serial Adder

If only one full adder is available, then addition can only be performed serially. This requires a circuit where the input data are temporarily saved and input to the adder in sequence. Also, the carry resulting from each intermediate addition needs to be saved for the subsequent addition stage. Finally, it is also necessary to store the results of each intermediate addition so that the individual bit totals can be combined to produce a parallel output once the last bit (MSB) has been added.

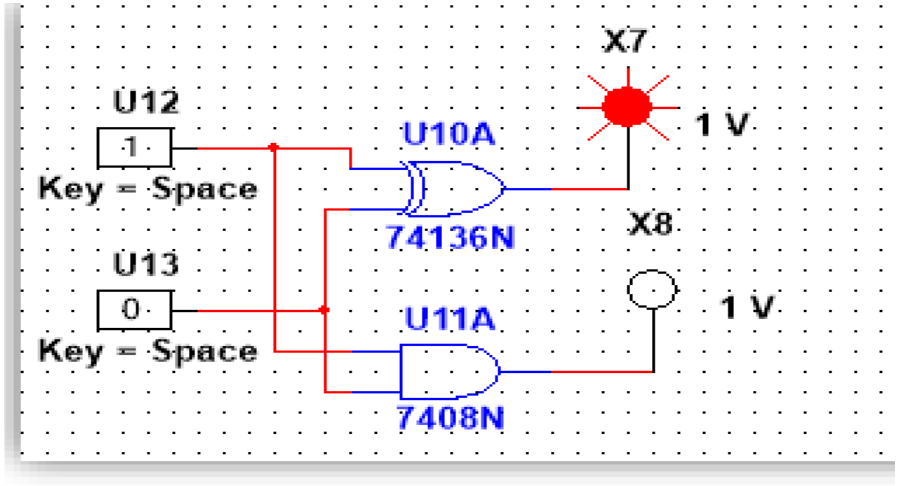


1. Shift register for the bit pattern of one number **A**.
2. Shift register for the bit pattern of the second number **B**.
3. 1-bit full adder
4. D-type flip-flop to store the carry bit

Procedure and observation:

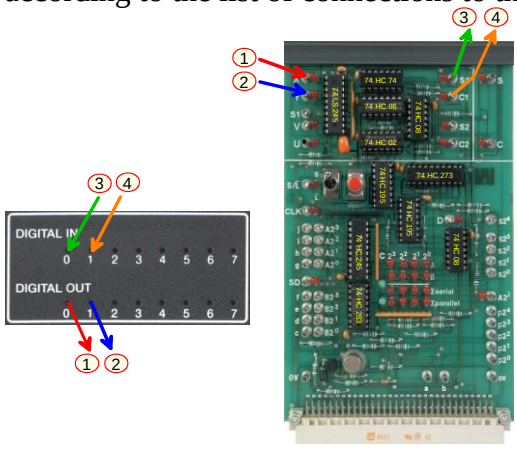
1. Design a half adder circuit and show all the calculations and equation along with circuit diagram in lab report.

The Half wave Adder circuit:



Experiment steps: Half adder

1. For half adder, connect the terminals on the card to those of the Uni*Tr@in*-I according to the list of connections to the right

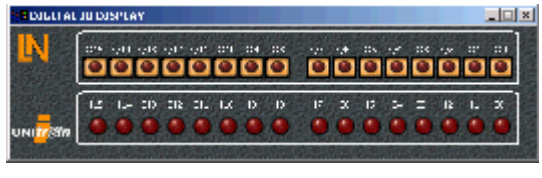


Click on the picture for an illustration of the wiring

List of connections

From	*To*
Digital out 0	Terminal **X**
Digital out 1	Terminal **Y**
Digital in 0	Terminalv**S1**
Digital in 1	Terminal **C1**

2. Open the following virtual instrument from the *Instruments* menu - Digital input and output



3. Set up the required bit patterns for **X** and **Y** and note the resulting outputs **S1** and **C1**.

Y	X	C1	S1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Derive the logical operations for the sum S1 and the carry C1

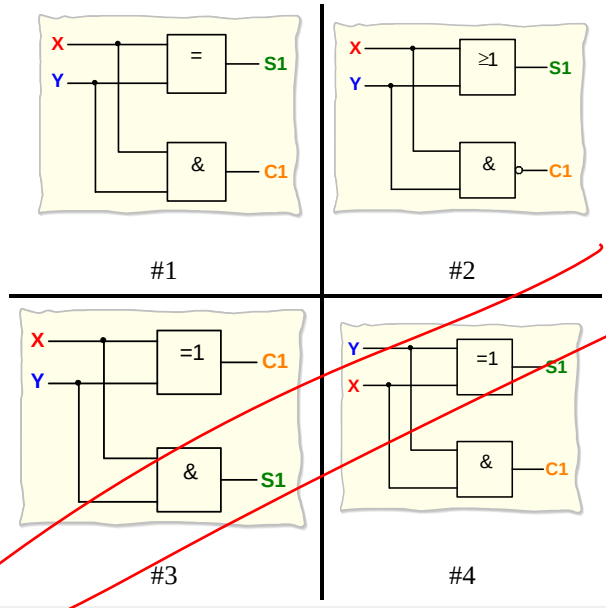
S1 = Y'X + YX'

C1 = YX

What do you conclude from the behavior of the circuit?

Ans: I conclude that Half Adder circuit only add Two (1-bit) numbers and give both Sum(S1) and Carry(C1) on output. We can easily write the logical equation for Sum(S1) and Carry(C1) by using truth table of Two (1-bit) numbers.At last, I observe that the Half Adder circuit make XOR gate for Sum(S1) and AND gate for Carry(C1).

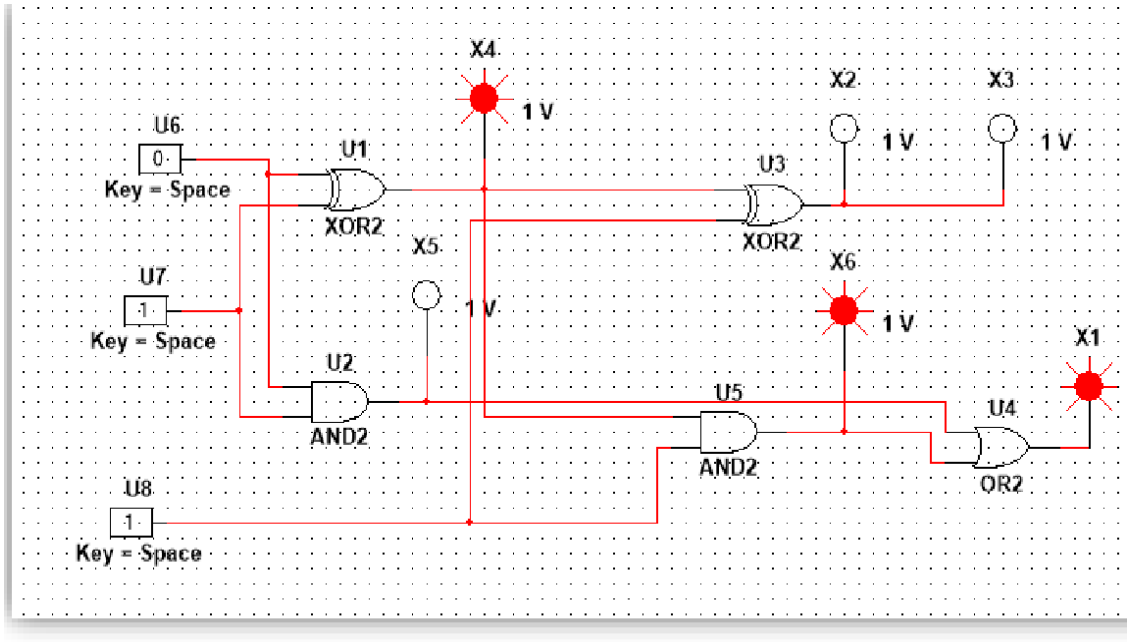
4. **Which of the following illustrations shows the correct set-up for a 1-bit half adder?**



Full adder:

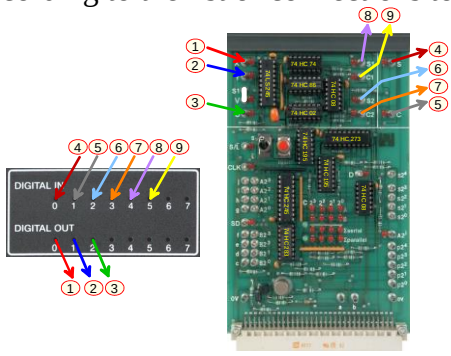
Design a full adder circuit by using two half adders. Show all the calculations and equations along with circuit diagram in the lab report.

The Full wave Adder circuit:



Experimental steps: Full adder

1. For full adder, connect the terminals on the card to those of the Uni*Tr@in*-I according to the list of connections to the right

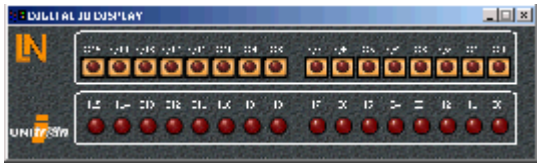


Click on the picture for an illustration of the wiring.

List of connections

From	*To*
Digital out 0	Terminal **X**
Digital out 1	Terminal **Y**
Digital out 2	Terminal **U**
Digital in 0	Terminal **S**
Digital in 1	Terminal **C**
Digital in 2	Terminal **S2**
Digital in 3	Terminal **C2**
Digital in 4	Terminal **S1**
Digital in 5	Terminal **C1**
Jumper	
V - S1	

2. Open the following virtual instrument from the *Instruments* menu:
   - Digital inputs



3. Set the required bit patterns for **X**, **Y** and **U** (representing the carry C_{n-1}) and note the resulting outputs **C1**, **S1**, **C2**, **S2**, **C** and **S**.

	Input			*Intermediate result*				*Final result*	
	U	**Y**	**X**	**C1**	**S1**	**C2**	**S2**	**C**	**S**
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	1
2	0	1	0	0	1	0	1	0	1
3	0	1	1	1	0	0	1	1	0
4	1	0	0	0	0	0	1	0	1
5	1	0	1	0	1	0	1	1	0
6	1	1	0	0	1	1	0	1	0
7	1	1	1	1	0	1	0	1	1

Logic operations for the sum S and carry C

S = U'Y'X + U'YX' + UY'X' + UYX

C = U'YX + UY'X + UYX' + UYX

= X(U'Y + UY') + UY(X' + X)

= X(U'Y + UY') + UY

Equation Design step ?

K-map