An-Najah National University

Faculty of Engineering and IT

جامعة النجاح الوطنية

كلية الهندسة وتكنولوجيا المعلومات

Digital Circuit Design Lab2

Dr. Ashraf Armoush

Wednesday 2:00pm – 5:00 pm

First Semester

Experiment Information	
Experiment Name: Two Bit Adder	Experiment Number: #2
Performed: 16 of Sep, 2020	Submitted: 25 of Sep, 2020
Partner Students	
1- Taher Anaya	2- Hadi Jml

Introduction:

In this experiment, we are going to construct various type of adders, starting off with half adder, then full adder, and finally a 2-bit adder, then we are also going to synthesize these adders after simulation them on FPGA.

Objectives:

- Constructing half adder, full adder, and 2-bit adder.
- Writing user defined modules and using them as component in other modules.
- Simulation a VHDL code and synthesizing it on an FPGA.

Procedure:

One of the objectives of this experiment is to construct modules and using them as component in bigger ones, so the first step in this experiment is to build the base module which is a **Half Adder**.

Half Adder

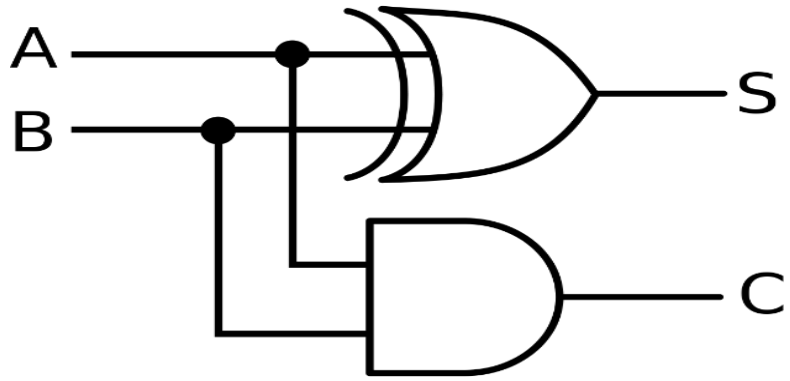
A half adder is a combinational arithmetic circuit that adds two bits and produces a sum bit(S) and carry bit (C) as the output. If A and B are the input bits, then sum bit (S) is the X-OR of A and B and the carry bit (C) will be the AND of A and B. From this it is clear that a half adder circuit can be easily constructed using one X-OR gate and one AND gate.

The truth table of a half adder is shown in the figure below.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Procedure: *(cont.)*

And the circuit design of a half adder is shown in the figure below.

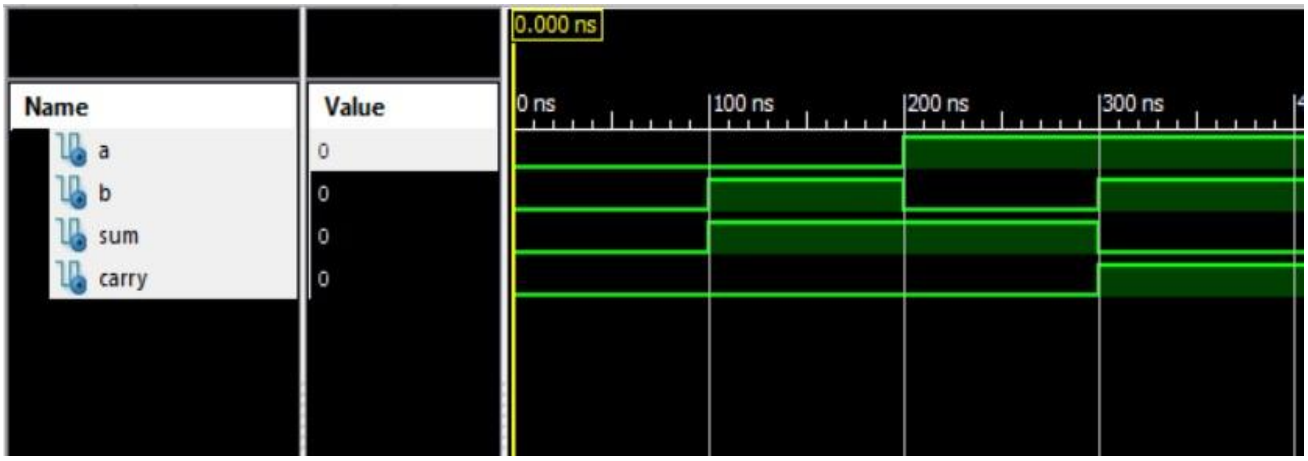


And the VHDL code of a half adder is shown in the figure below.

```
 1 library IEEE;
 2 use IEEE.STD_LOGIC_1164.ALL;
 3
 4 -- Entity Of Half Adder
 5 entity HalfAdder is
 6     Port ( A     : in  STD_LOGIC;
 7            B     : in  STD_LOGIC;
 8            Sum   : out  STD_LOGIC;
 9            Carry : out  STD_LOGIC);
10 end HalfAdder;
11
12 -- Architecture Of Half Adder
13 architecture Behavioral of HalfAdder is
14
15 begin
16
17    Sum   <= A xor B;
18    Carry <= A and B;
19
20 end Behavioral;
```

Procedure: *(cont.)*

And to ensure that out implementation is correct, we had to create a test bench file and test out every single possible input value and comparing the output with the expected result.



As we can see, no issues were found and the result was exactly as expected. The following step is to test the half adder on the FPGA, now we need to create a UCF file to map the inputs and outputs in VHDL to the switches and LEDs on the FPGA.

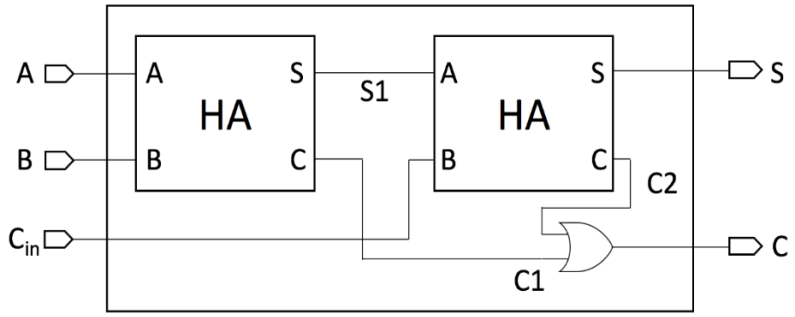
Since we have the half adder ready, we can move on to the next step which is constructing a Full Adder using Two Half Adders.

Full Adder

A full Adder is a combinational arithmetic circuit that adds three inputs and produces two outputs. The first two inputs are **A** and **B** and the third input is an input carry as **C-IN**. The output carry is designated as **C-OUT** and the normal output is designated as **S** which is SUM.

Procedure: *(cont.)*

The circuit design of a full adder using two half adders as shown in the figure below.

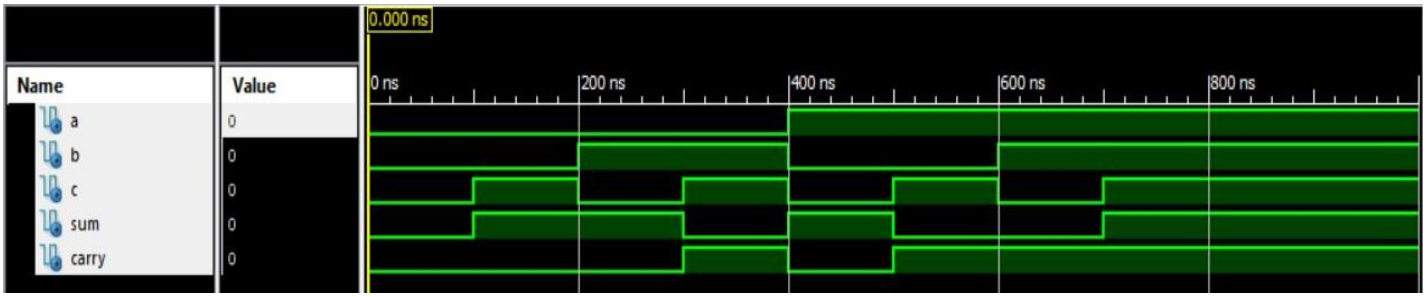


And the VHDL code of a full adder is shown in the figure below.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  -- Entity of Full Adder
5  entity fullAdder is
6      Port ( A : in  STD_LOGIC;
7             B : in  STD_LOGIC;
8             C : in  STD_LOGIC;
9             Sum : out  STD_LOGIC;
10            Carry : out  STD_LOGIC);
11 end fullAdder;
12
13 -- Architecture of Full Adder
14 architecture Behavioral of fullAdder is
15
16    component HalfAdder is
17     Port ( A     : in  STD_LOGIC;
18            B     : in  STD_LOGIC;
19            Sum   : out  STD_LOGIC;
20            Carry : out  STD_LOGIC);
21    end component;
22    Signal Sum1, Carry1, Carry2: STD_LOGIC;
23 begin
24
25    ha1: HalfAdder port map(A, B, Sum1, Carry1);
26    ha2: HalfAdder port map(Sum1, C, Sum, Carry2);
27    Carry <= Carry1 or Carry2;
28
29 end Behavioral;
30
```

Procedure: *(cont.)*

To make sure that our implementation is correct, we created a test bench file to test out every single possible input value and compare the output with the expected result.



As we can see, no issues were found and the result was exactly as expected.

The following step is synthesizing the full adder on the FPGA, so we need to create a UCF file to map the inputs and outputs in VHDL to the switches and LEDs on the FPGA, just like we did with the half adder.

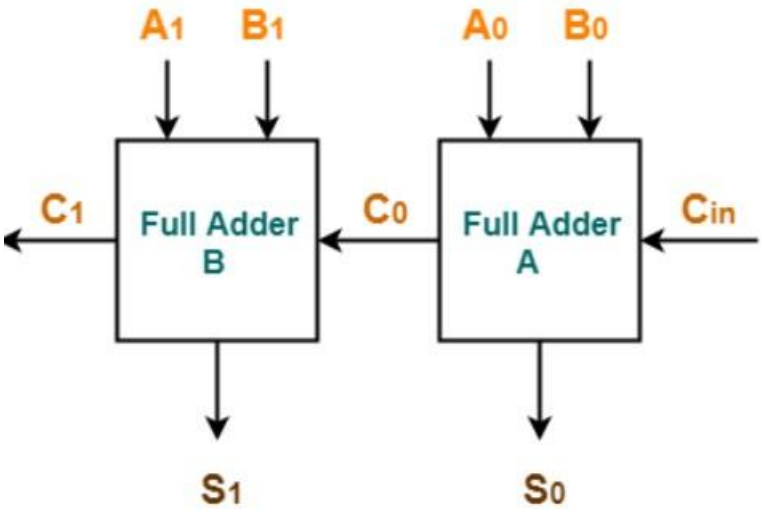
Since we have the full adder ready, we can move on to the next step which is constructing a **2bit adder** using Two Full Adders.

2-Bit Adder

A 2-bit adder is simply two full adders cascaded together, with each full adder representing a single weighted column in a long binary addition.

Procedure: (cont.)

The circuit design of a 2-bit adder using two half adders as shown in the figure below.



Now and before moving on to the next step, our FPGA doesn't have enough switches to map five inputs, so we had to purposely get rid of Cin and replace it with an internal '0' signal.

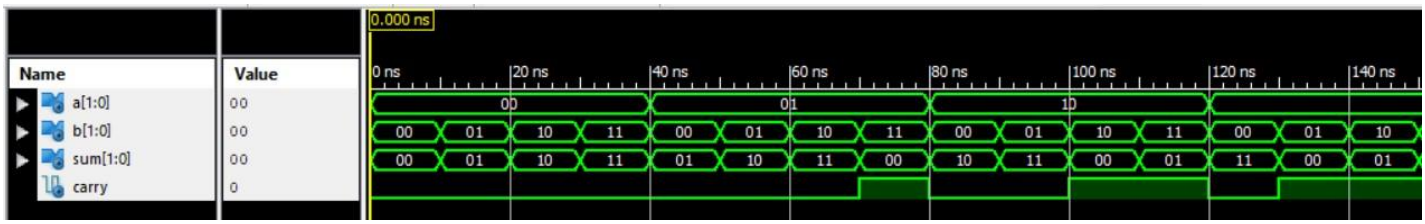
Procedure: *(cont.)*

Now we are ready to implement the 2-bit adder using VHDL:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  -- Entity of Two Bit Adder
5  entity TwoBitAdder is
6      Port ( A : in  STD_LOGIC_VECTOR (1 downto 0);
7             B : in  STD_LOGIC_VECTOR (1 downto 0);
8             Sum : out  STD_LOGIC_VECTOR (1 downto 0);
9             Carry : in  STD_LOGIC);
10 end TwoBitAdder;
11
12 -- Architecture of Two Bit Adder
13 architecture Behavioral of TwoBitAdder is
14
15    component fullAdder
16     Port ( A : in  STD_LOGIC;
17            B : in  STD_LOGIC;
18            C : in  STD_LOGIC;
19            Sum : out  STD_LOGIC;
20            Carry : out  STD_LOGIC);
21    end component;
22
23    Signal zero: STD_LOGIC := '0';
24    Signal tempCarry: STD_LOGIC;
25
26 begin
27
28    fa1: fullAdder port map(A(0), B(0), zero, Sum(0), tempCarry);
29    fa2: fullAdder port map(A(1), B(1), tempCarry, Sum(1), Carry);
30
31 end Behavioral;
32
```

Procedure: *(cont.)*

To make sure that our implementation is correct, we created a test bench file to test out every single possible input value and compare the output with the expected result.



As we can see, no issues were found and the result was exactly as expected.

The Final step is synthesizing the 2-bit adder on the FPGA, so we need to create a UCF file to map the inputs and outputs in VHDL to the switches and LEDs on the FPGA, just like we did with the half adder and full adder.

Conclusion:

In the end, we were able to construct full adders using two half Adders, and 2-bit adder using full adders. We also know how to simulation VHDL code before synthesize it, and how to map different variables in the VHDL code to the switches/LEDs on the FPGA, and finally we learned how to install a VHDL code on the FPGA and synthesizing it.