Lab 1 Report

Team

Κριθαράκης Εμμανουήλ
Φωτάκης Τζανής

Preparation

To begin with, the lab was divided into 2 parts. In the first part, we have to do as a preparation the block diagram, logic equations and the truth table of a circuit of logic gates. To be more specific, the logic gates were AND, XOR, NOT and NOR. In the second part, we should have done a fulladder of 2 bits, which have had as components 2 halfadders. The block diagrams, truth tables and logic equations of both part are displayed below in picture 1 and 2 respectively.

Circuit 1

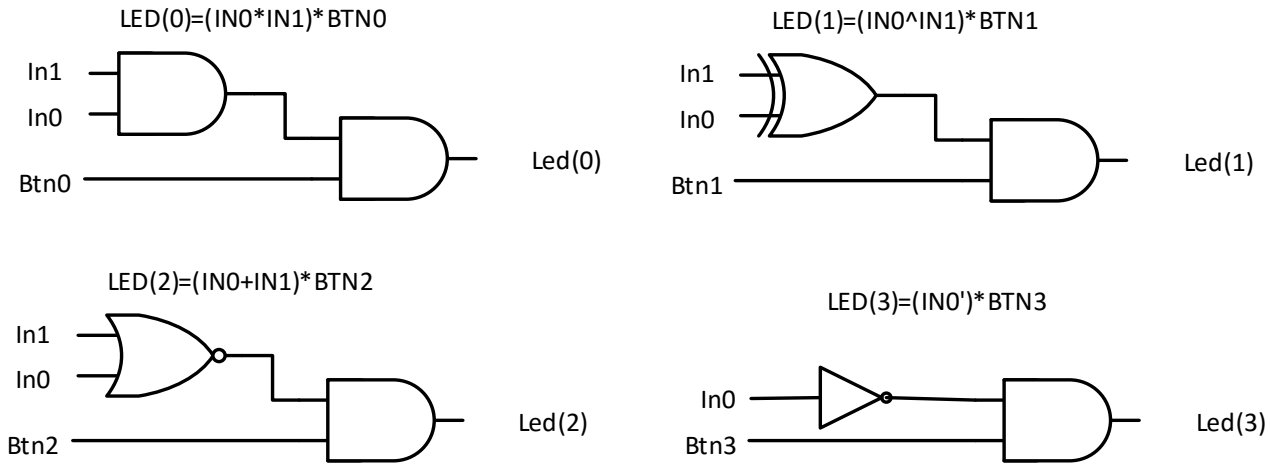
Logic Equations

- LED(0)=(IN0*IN1)*BTN0
- LED(1)=(IN0^IN1)*BTN1
- LED(2)=(IN0+IN1)*BTN2
- LED(3)=IN0*BTN3
- LED(4)=IN0
- LED(5)=IN1
- LED(6)=0
- LED(7)=0

Truth Table

IN1	IN0	BTN0	BTN1	BTN2	BTN3	LED7-LED0
0	0	0	0	0	0	00000000
0	0	1	1	1	1	00001100
0	1	0	0	0	0	00010000
0	1	1	1	1	1	00010010
1	0	0	0	0	0	00100000
1	0	1	1	1	1	00101010
1	1	0	0	0	0	00110000
1	1	1	1	1	1	00110001

Circuit Design



Circuit 2

Logic Equations

Half-Adder

- $\text{Sum} = A \oplus B$
- $\text{Cout} = A * B$

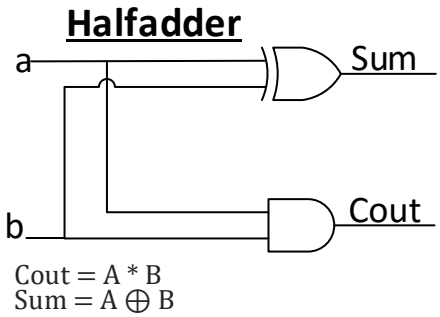
Full-Adder

- Sum = $A \oplus B \oplus \text{Cin}$
- Cout = $\text{Cin}*(A \oplus B)) + (A*B)$

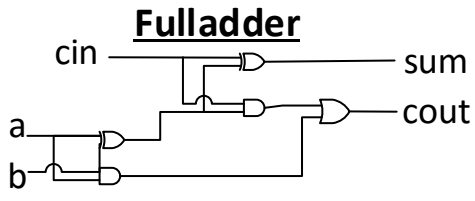
Truth Table

A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Circuit Design



$\text{Cout} = A * B$
$\text{Sum} = A \oplus B$



Sum = $A \oplus B \oplus \text{Cin}$
Cout = $\text{Cin}*(A \oplus B)) + (A*B)$

Overview

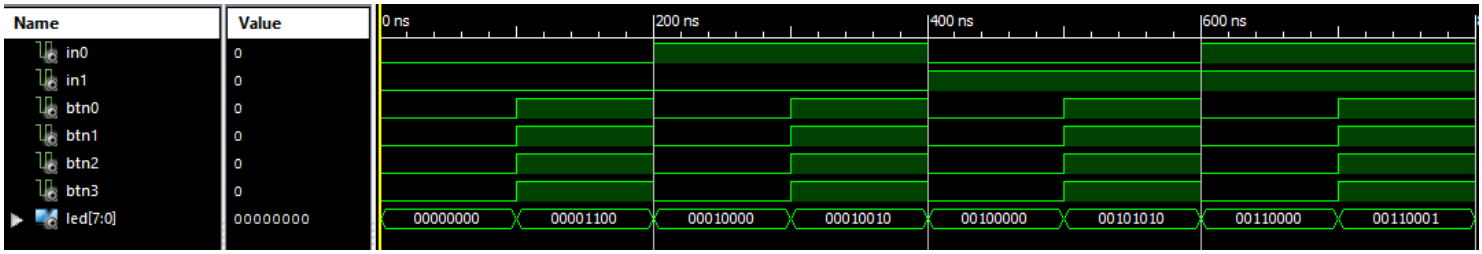
The purpose of the first circuit was to introduce us in the vhdl language and the idea of the design flow (front end – back end). This first circuit has some inputs (in0, in1) and buttons (btn0, btn1, btn2, btn3) and some outputs (a vector of leds from 0 to 7). First of all, at the vhd file we create not only the entity (the interface of circuit) giving the outputs and inputs but also in the architecture part (the implementation of circuit) the appropriate code (logic equations) in order to create a circuit with the inputs and outputs that it should have.

The purpose of the second circuit was to help us understand better the design of structural architecture. The second circuit of fulladder has some inputs (x, y, cin) and some outputs carry and sum. Following the some way with the first circuit and with the slight difference of the use of the component halfadder (which has some inputs (x, y) and some outputs carry and sum) we create the vhd file. To elaborate it, having created the vhd file for halfadder (the architecture part include the appropriate logic equations) all we had to do was to port map the right inputs and outputs of the instantiations of halfadder in order to create the fulladder (and to add a logic equation for the output of carry out of fulladder).

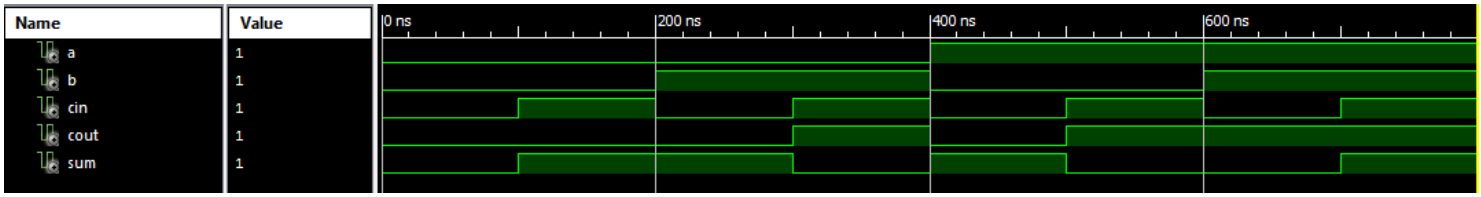
Waveforms -Simulation

Having done the vhd files for both circuits all we have to do was to create test benches to check if the circuits work as we want. In order to do that, we should take into consideration some of the possible inputs and see if the outputs are the correct ones. The test benches help us as they produce simulations of how the circuit is going to work with specific inputs. Here are the simulations:

Circuit 1



Circuit 2



Conclusion

This first lab exercise helped us a lot to understand not only how the Xilinx program works but also how we should follow the design flow in order to produce a full-working circuit with the characteristics we want. Furthermore, we produced a UCF file (user constrain file), from which we realize that we – the designers are able to make sure that specific inputs - outputs of our design are going to be the desirable ones in the fpga chip.

Code

Circuit 1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Curcuit1 is
   Port ( in0 : in  STD_LOGIC;
          in1 : in  STD_LOGIC;
          btn0 : in  STD_LOGIC;
          btn1 : in  STD_LOGIC;
          btn2 : in  STD_LOGIC;
          btn3 : in  STD_LOGIC;
          led : out  STD_LOGIC_VECTOR (7 downto 0));
end Curcuit1;
architecture Behavioral of Curcuit1 is
begin
led<=(0=>(in0 and in1) and btn0,
                    1=>(in0 xor in1) and btn1,
                    2=>(in0 nor in1) and btn2,
                    3=>(not in0) and btn3,
                    4=>in0,
                    5=>in1,
                    others=>'0'
                    );
end Behavioral;
```

Circuit 2

Half Adder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity halfAdder is
   Port ( a : in  STD_LOGIC;
```

```
        b : in  STD_LOGIC;
        cout : out  STD_LOGIC;
        sum : out  STD_LOGIC);
end halfAdder;
architecture Behavioral of halfAdder is
begin
          cout<=a and b;
          sum<=a xor b;
end Behavioral;
```

Full Adder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity FullAdder is
   Port ( a : in  STD_LOGIC;
        b : in  STD_LOGIC;
        cin : in  STD_LOGIC;
        cout : out  STD_LOGIC;
        sum : out  STD_LOGIC
        );
end FullAdder;
architecture structural of FullAdder is
signal cout1: std_logic;
signal sum1: std_logic;
signal cout2: std_logic;
component halfAdder
          port (
                        a :in std_logic;
                        b :in std_logic;
                        cout :out std_logic;
                        sum :out std_logic
          );
end component;
begin
          halfAdder1: halfAdder port map(a=>a,b=>b,cout=>cout1,sum=>sum1);
          halfAdder2: halfAdder port map(a=>sum1,b=>cin,cout=>cout2,sum=>sum);
          cout<=cout1 or cout2;
end structural;
```