Labb 2 johannes joujo

Task 1

In task 1 i created a file named fa and put the following inside of it. It is a 1-bit full adder that has two inputs a and b with two outputs s and cout. It calculates the sum and carry-out of a binary addition operation.

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 02/17/2022 11:54:26 AM
-- Design Name:
-- Module Name: fa - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
```

```
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity fa is
port (a, b, cin : in std_logic;
s, cout : out std_logic);
end fa;
architecture Behavioral of fa is
begin
 s <= a xor b xor cin;
  cout <= (a and b) or (b and cin) or (cin and a);
end Behavioral;
```

Explaining the implementation/code

Port a,b and cin are the input while s and cout are output ports. First a,b and cin is being added together with xor. The carryout is being calculated using different combinations of a,b and cin, with or operations linking them together.

A and b are the two inputs being added together and if there is a previous carry out it is also added.

The testbench was given to me by the teacher. When I put the file fa in the development tools project with the corresponding testbench I got this result. I got the architecture behavioral from internet[1].

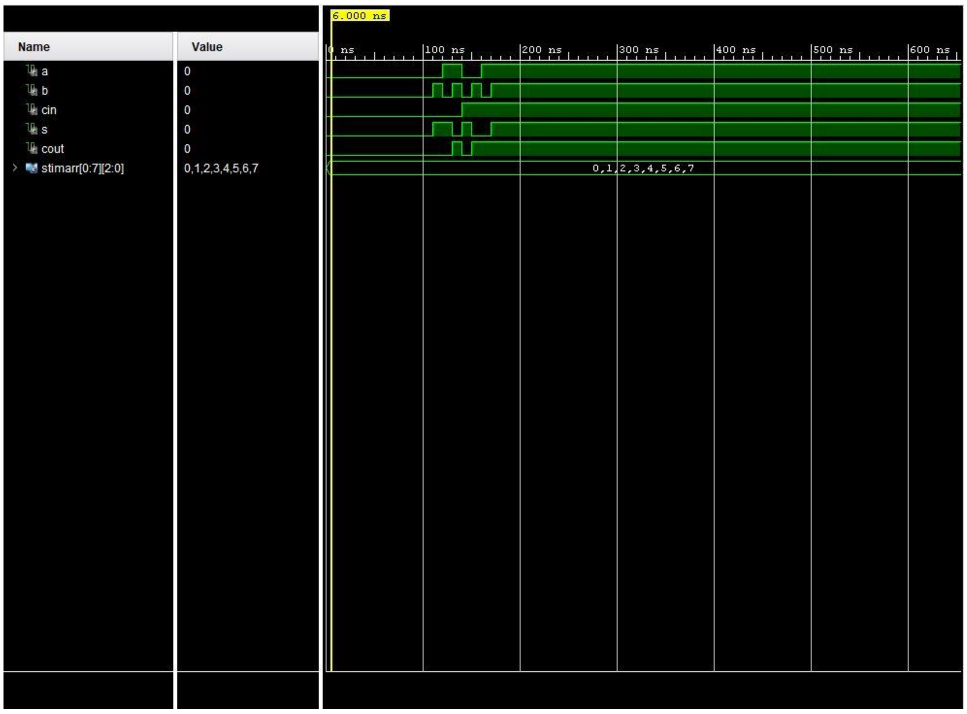


Figure 1 shows the result of the one-bit full adder.

In this case we see that all the inputs and the output is 0. During the test the cursor was moved to check if the adder was working correctly.

The 1-bit full adder has one output and three inputs a, b and cin. a and b are new inputs while cin is a carry in.

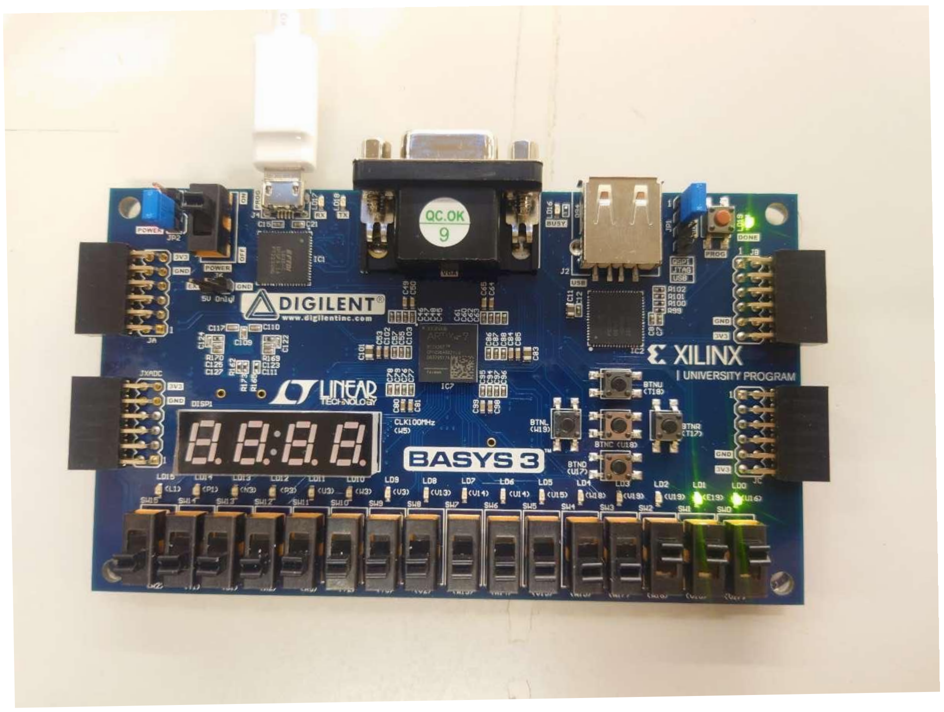


Figure 2 shows FPGA board.

I added a constraints file to the project that would let me connect this FPGA board and tested my adder. When the FPGA board was connected successfully, I could see a light turn on when the output was 1 and off when the output was 0.

Task 2 I made add8 and add8_behave. It is written in a higher abstract level with the + operator. I put both in one file.

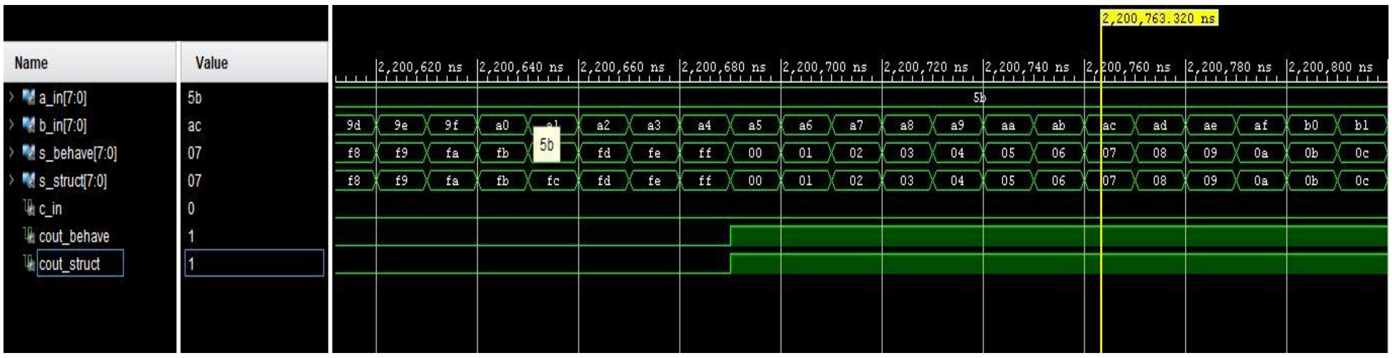
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity add8 is
port ( a, b : in std_logic_vector(7 downto 0); cin : in std_logic;

sum : out std_logic_vector(7 downto 0); cout : out std_logic);
end add8;
architecture Behavioral of add8 is component fa is
port(a, b, cin : in STD_LOGIC;
s: out STD_LOGIC;
cout: out STD_LOGIC);
end component;
signal carry : std_logic_vector(6 downto 0);
begin
a0: fa port map (a=>a(0), b =>b(0), cin=> cin, s => sum(0), cout=>carry(0));
a1: fa port map (a=>a(1), b =>b(1), cin=> carry (0),s => sum(1), cout=>carry(1)); a2 :fa port
map (a=>a(2), b =>b(2), cin=> carry (1),s => sum(2), cout=>carry(2)); a3 : fa port map
(a=>a(3), b =>b(3), cin=> carry (2),s => sum(3), cout=>carry(3)); a4: fa port map (a=>a(4), b
=>b(4), cin=> carry (3),s => sum(4), cout=>carry(4)); a5: fa port map (a=>a(5), b =>b(5),
cin=> carry (4),s => sum(5), cout=>carry(5)); a6: fa port map (a=>a(6), b =>b(6), cin=> carry
(5),s => sum(6), cout=>carry(6)); a7:fa port map (a=>a(7), b =>b(7), cin=> carry (6),s =>
sum(7), cout=>cout);
end Behavioral;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use ieee.std_logic_unsigned.all; use
IEEE.NUMERIC_STD.ALL; use IEEE.STD_LOGIC_arith.ALL;

entity add8_behave is
port ( a, b : in std_logic_vector(7 downto 0); cin : in std_logic;
sum : out std_logic_vector(7 downto 0); cout : out std_logic);
end add8_behave;
architecture Behavioral of add8_behave is signal sum_cout: std_logic_vector (8 downto 0);
begin
sum_cout<=(cin&a) + (cin&b); sum <=sum_cout(7 downto 0); cout<= sum_cout(8);
end Behavioral;
```

Explaining the implementation/code

Add8 has two 8-bit inputs and a carry in, it has two outputs sum and carry out. The sum is added with bit wise addition between a and b. If there is a carry out from a previous addition it will be added to the bit wise addition also.


Name	Value
a_in[7:0]	5b
b_in[7:0]	ac
s_behave[7:0]	07
s_struct[7:0]	07
c_in	0
cout_behave	1
cout_struct	1

2,200,763.320 ns
2,200,620 ns
2,200,640 ns
2,200,660 ns
2,200,680 ns
2,200,700 ns
2,200,720 ns
2,200,740 ns
2,200,760 ns
2,200,780 ns
2,200,800 ns
5b
9d 9e 9f a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af b0 b1
f8 f9 fa fb 5b fd fe ff 00 01 02 03 04 05 06 07 08 09 0a 0b 0c
f8 f9 fa fb fc fd fe ff 00 01 02 03 04 05 06 07 08 09 0a 0b 0c

Figure 3 shows result from 8-bit full adder.

Started by connecting 8 1-bit adders together to form one 8-bit adder. I kept following the instructions.

Discussion

This lab taught me how to implement a 1 and 8-bit full adders and how to connect to the FPGA board.

References

[1]https://allaboutfpga.com/vhdl-code-for-full-adder/