Labs 2 - VHDL Design

Submitted by **Meldrick Reimmer, Khoi Pham and Selma Boudissa**

Contents

1 Introduction

VHDL Language (Very high speed integrated circuit Hardware Description Language) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits. We use this language to sketch the logical patterns and simulate the electrical system for real life application.

2 Problem 1

Import the The std_logic Libraries IEEE. This creates std_logic type in standard 1164 since we only need STD_LOGIC in order to create our logic I/O ports to simulate this system. We use architecture keyword to describe the functionality of this entity. In this case, it's a logical operator process. The code lines inside Behavioral will be triggered every time this entity is call.

Figure 1: Test bench

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity And3 is
    Port (
        a : in  STD_LOGIC;
        b : in  STD_LOGIC;
        c : in  STD_LOGIC;
        s : out  STD_LOGIC);
end And3;

architecture Behavioral of And3 is

begin
    s <= (a and b and c);
end Behavioral;
```

Figure 2: VHDL code

2.1 Test Bench:

Test bench in vhdl is where you put your models in action, create a scenario for our previously built entities to work in a simulation environment. So we can observe its performance.

```
stimulus: process

begin
    a<='0';
    b<='0';
    c<='0';
    wait for 100 ns;
    a<='1';
    b<='0';
    c<='0';
    wait for 100 ns;
    a<='1';
    b<='1';
    c<='0';
    wait for 100 ns;
    a<='1';
    b<='1';
    c<='1';
    wait for 100 ns;
    a<='1';
    b<='0';
    c<='1';
    wait for 100 ns;
    a<='1';
    b<='1';
    c<='0';
    wait for 100 ns;
    a<='0';
    b<='0';
    c<='1';
end process ;
```

Figure 3: VHDL code

Analyze: In this test bench we use this stimulus scenario, by the beginning the binary value of input ports are 0 and will be altered after a step of 100ns so we can watch output changes in vhdl performance graph.

Figure 4: Test bench

Observation s is the result of logical operation of a b & c, In this AND3 algorithm, s only be 1 where a b & c is 1 at the same time.

Figure 5: Test bench

3 Problem 2

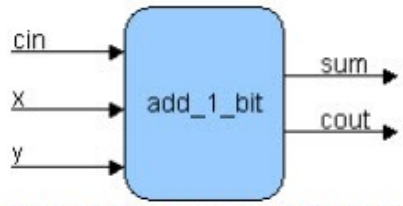

Figure 1 - 1-Bit Full Adder Component

1-bit full Adder is characterized by the following equations:

1. sum = x xor y xor cin
2. cout = (x and y) or (x and cin) or (x and cout)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Add1 is
    Port (
        a : in  STD_LOGIC;
        b : in  STD_LOGIC;
        cin : in STD_LOGIC;
        s : out  STD_LOGIC;
        cout : out  STD_LOGIC);
end Add1;

architecture Behavioral of Add1 is

begin
    s <= a xor b xor cin;
    cout <= (a and b) or (a and cin) or (b and cin);
end Behavioral;
```

Figure 6: VHDL code

ADD1 3 inputs 2 outputs with above logical operation for 3 arguments described.

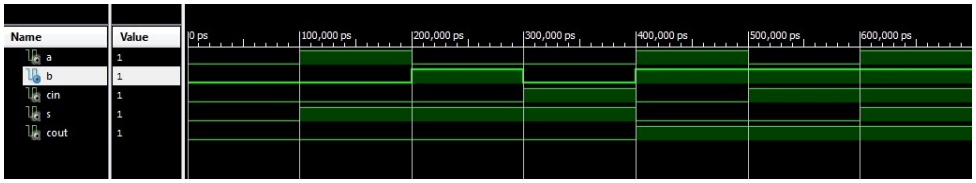

Figure 7: Test bench

Observation: Same as AND3, the s and cout is the result of logical operation, the difference here is only the operation algorithm .

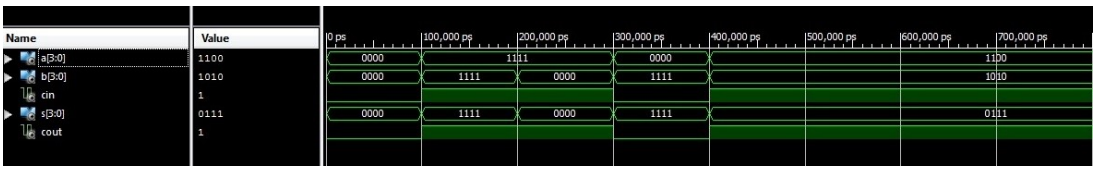

Figure 8: Test bench

4 Problem 3: 4 bit full adder

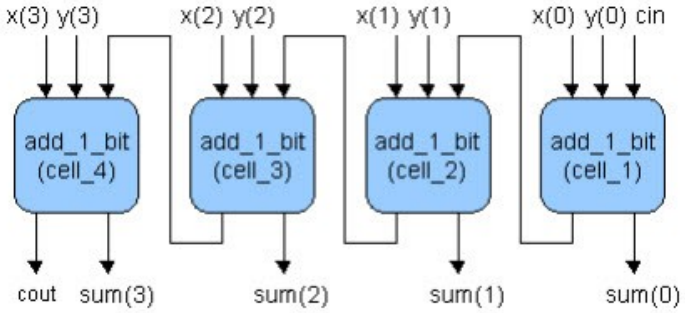

Figure 2 - 4-Bit Unsigned Adder

gincenter

Observation: Since Add4 consists of 3 inputs and 2 outputs which has same characteristic described Add1, moreover, Add4 js a a cascade of 4 Add1 chained together, share the same clock. Which means the cout of the first Add1 will be cin of the next Add1 until all 4 Add1 is processed, the cout of the last Add1 will be the final output of this system.

```
-- i_carry is the cout of the first adder and will become the cin of the
-- next adder, keep doing this till the end to achieve 4 bit adder system
adder1: Add1
port map (a(0),b(0),cin,s(0),i_carry(0));
adder2: Add1
port map (a(1),b(1),i_carry(0),s(1),i_carry(1));
adder3: Add1
port map (a(2),b(2),i_carry(1),s(2),i_carry(2));
adder4: Add1
port map (a(3),b(3),i_carry(2),s(3),cout);
```

Figure 9: VHDL code

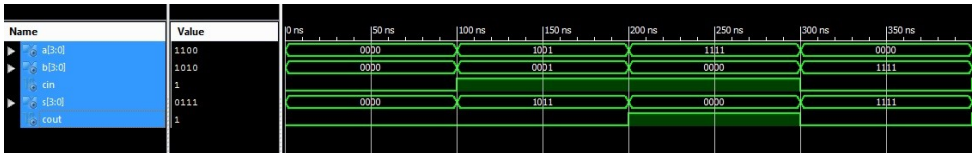

Figure 10: Test bench

Observation: Every pair of a & b at a specific bit (ie: at bit 0, a is 1 and b is 1) together will c will join in the logical operation to come up with the result of s (at the correspond bit) and cout.

Figure 11: Test bench

5 Problem 4

D Flipflop entity implemented:

```
library ieee;
use ieee.std_logic_1164.all;

entity flipflop is
    port(
        D, clk : in STD_LOGIC;
        q : out STD_LOGIC);
end flipflop;

architecture Behavioral of flipflop is
begin
    process(clk)
    begin
        -- if the clock is at rising edge (hit maximum)
        -- assign input value to the output
        if (clk='1' and clk'event) then
            q <= D ;
        end if;
    end process
end Behavioral;
```

Figure 12: VHDL code

Analyze: Flip flop is the system use a variable name clock to handle the output, in my example it's "rising edge", the time where the clock hit maximum value, output will get updated by input, otherwise, it keeps the same value.

Figure 13: Test bench

Observation: The q will have the same value with d (1 or 0) whenever the clock is at 1 (maximum)

6 Problem 5

8 bit counter flip flop algorithm:

```
process (clk, reset) begin
    -- reset the counter
    if (reset = '1') then
        counter <= (others=>'0');
    elsif (rising_edge(clk)) then
    -- the counter will increase by 1
    -- everytime the clock hit maximum value
        counter <= counter + 1;
    end if;
end process;

-- output the counter
s <= counter;
```

Figure 14: VHDL code

Analyze: The counter will start counting as long as the reset is set to 0, otherwise reset counter to 0. The clock will tick throughout the time but according to the state of the clock in stimulus, if the clock hit maximum, it count, otherwise keeps same value. Note that the "rising_edge(clk)" sentence here is the same way to say "clk=’1’ and clk’event".

Figure 15: Test bench

Observation: s will increase it bit value by 1 every time the clock hit its maximum value (1). Otherwise, s keeps same value.

7 Problem 6

In digital circuits, a shift register is a cascade of D flip flops, sharing the same clock, in which the output of each flip-flop is connected to the 'data' input of the next flip-flop in the chain, resulting in a circuit that shifts by one position the 'bit array' stored in it, 'shifting in' the data present at its input and 'shifting out' the last bit in the array, at each transition of the clock input.

```
-- the first input "d" will be init by 1 in s
-- be shift to the left side until the last b
FF0: FlipFlop
port map (D, clk, shifter(0));

FF1: FlipFlop
port map (shifter(0), clk, shifter(1));

FF2: FlipFlop
port map (shifter(1), clk, shifter(2));

FF3: FlipFlop
port map (shifter(2), clk, shifter(3));

FF4: FlipFlop
port map (shifter(3), clk, shifter(4));

FF5: FlipFlop
port map (shifter(4), clk, shifter(5));

FF6: FlipFlop
port map (shifter(5), clk, shifter(6));

FF7: FlipFlop
port map (shifter(6), clk, shifter(7));
```

Figure 16: VHDL code

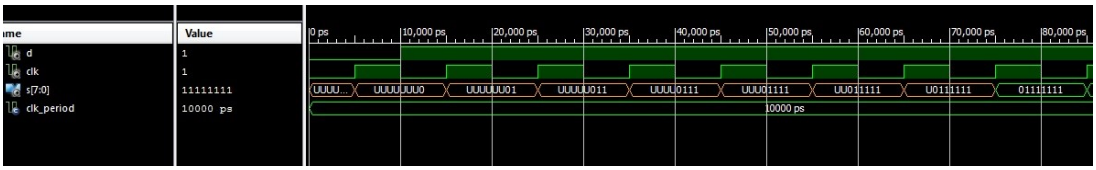

Figure 17: Test bench

Observation: The value 1 from the starting bit is shifted to the left every time the clock hit it maximum value. However, I think this solution is not quite true since the old bit still keeps it value 1.