Laboratory Session 5

Introduction to VHDL: Additional Statements (2)

Objective:

- Understanding the basics of VHDL descriptions
- Use Quartus IDE
- Use Modelist Simulator

Software Packages

- Altera Quartus II Software
- ModelSim Altera Software

1. Introduction

In VHDL, sequential statements are implemented using processes. A process block can have an optional label and it takes a sensitivity input list. The sensitivity list can include inputs and signals that the activates the process block upon changing. In item declaration, variables can be defined. Note that when a value is written to a signal, the updated value is read in the next cycle whereas variables are updated immediately.

```
my_label: process(sensitivity_list) is
   <item_declaration>
begin
   <sequential_statements>
end process my_label;
```

A function in VHDL is used to simplify code development. Statements in a function are executed sequentially.

```
function function_name (parameter_list) return type is
        declarations
begin
        sequential statements
end function_name;
```

Generate is a VHDL statement that can replicate components to run concurrently

```
label: for parameter in range generate
         concurrent statements
end generate label;
```

2. Samples

2.1.Four-Bit Adder

Use the following VHDL code in a new project.

```
library ieee;
use ieee.std_logic_1164.all;
entity Four_Bit_Adder is
port(
        a,b: in std_logic_vector(3 downto 0);
        cin: in std_logic;
        s: out std_logic_vector(3 downto 0);
        cout: out std_logic
);
end entity;

architecture struct of Four_Bit_Adder is

component Full_Adder is
port(
        a,b,cin : in STD_LOGIC;
        s,cout : out STD_LOGIC
);
end component;

signal c: std_logic_vector(5 downto 0);
begin
        c(0) <= cin;
        U: for I in 0 to 3 generate
                FA: Full_Adder port map (a(I),b(I),c(I),s(I),c(I+1));
        end generate;
        cout <= c(4);

end architecture;
```

Save the file using the entity name (Four_Bit_Adder) in the created folder. Check the RTL, Power Consumption, Fmax and State Machine Viewer. Verify the functionality using ModelSim.

The code for the Full_Adder is below for your convenience. You do not need to create an entire new project for the Full_Adder. Adding it as a VHDL file to your Four_Bit_Adder project is enough.

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

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

architecture beh of Full_Adder is
begin

s <= a xor b xor cin;
cout <= (a and b) or (b and cin) or (cin and a);

end architecture;
```

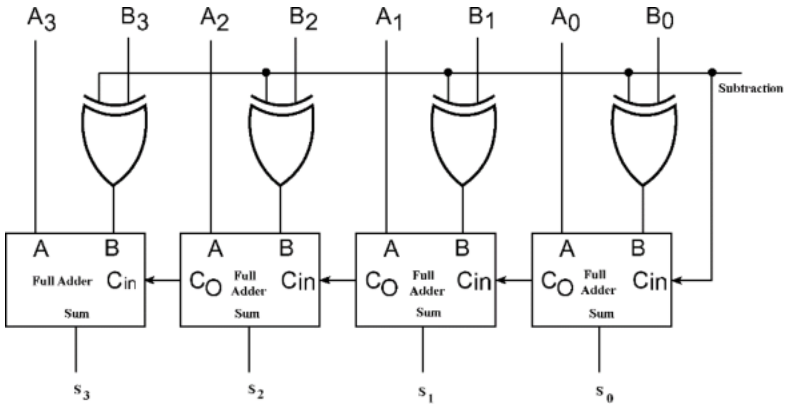
2.2. Exercise: 32-Bit Adder

Design and implement a 32-bit adder using Generate of smaller size bit adders. Check the RTL, Power Consumption, Fmax and State Machine Viewer.

3. Homework

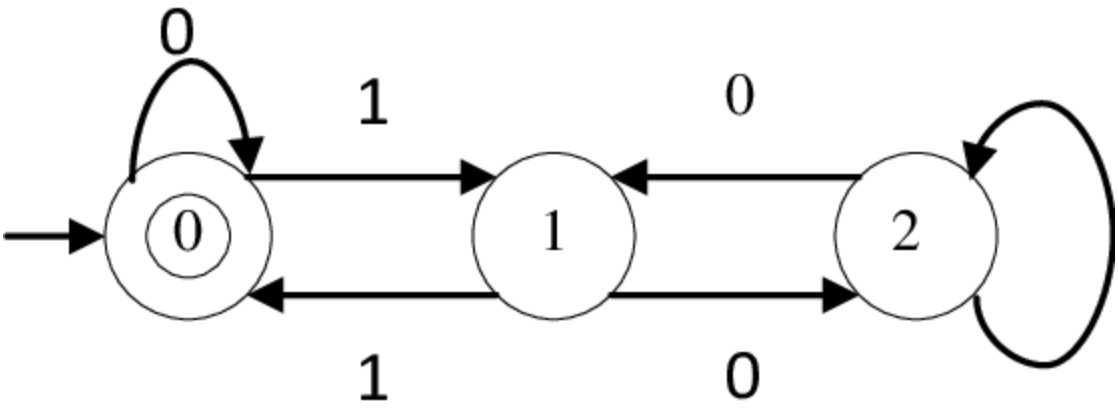
3.1. 32-Bit Adder/Subtractor

Design and implement a 32-bit adder/subtractor using Generate of smaller size bit Full_Adder. Check the RTL, Power Consumption, Fmax and State Machine Viewer. The following is a 4-bit adder/subtractor diagram for reference:



3.2. FSM under VHDL

Design the following FSMD using Cases and if statements



After compiling, check the RTL, power consumption and maximum frequency:

- Check the **maximum frequency** as follows:

1. Go to the compilation report tab
2. On the left pane, under table of contents expand TimeQuest Timing Analyzer.
3. Expand Slow Model. Note: in case there are several Slow Models expand the one with the highest temperature as this analysis are device based and we are not specifying our device so far.

4. Select Fmax Summary

- To **verify the FSM**, go to Tools → Netlist Viewer → State Machine Viewer to view the available states, make sure that there are 4 states S0, S1, S2 and S3. Verify using ModelSim.

4. Deliverables

Practical demonstration of the sample exercises will be requested during the lab. Keep in mind that a combined report that presents the analysis and testing of the sample exercises as well as the practice exercises of lab 4 (both weeks) will be requested, so maintain your project file for the snapshots of the work done.