Laboratory Session 3

Introduction to VHDL: Control Structures and Arrays

Objective

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

Software Packages

- Altera Quartus II
- ModelSim Software

1. Introduction

Control structures, loops and arrays are part of the behavioral VHDL. In this session common control structures are explored, namely, the If statement, CASE statement, and FOR loops.

- **If Statement**

```
if condition_1 then
        sequential statements
elsif condition2 then
        sequential statements
else
        sequential statements
end if;
```

- **Case Statement**

```
case expression is
   when choice =>
     sequential statements
   when choice =>
      sequential statements
end case;
```

- **For Loop**

```
optional_label: for parameter in range loop
        sequential statements
end loop label;
```

Along with the control structure a new data type will be introduced. Arrays are a collection of the same type used to consolidate and simplify the access of multiple elements.

type type_name **is array** (range) **of** element_type;

2. Samples

2.1. Is-Equal Comparator

For this exercise, we will implement a comparator that takes two integer values and checks if they are equal. The comparison is done using if statements. Note that for an if-statement to work, it needs to be within a process block; which takes a sensitive list of variables to check for. Create a new folder (e.g. Equal). Then, create a project (e.g. Equal) following the same steps as in previous labs. Create a new VHDL file and paste the following code.

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

entity Equal is
port (
        a, b: in integer;
        c: out std_logic
);
end entity;

architecture beh of Equal is
begin
        process(a,b)
        begin
                if (a = b) then
                        c <= '1';
                else
                        c <= '0';
                end if;
        end process;
end architecture;
```

Compile the code and produce all the necessary analysis files. In addition, apply the following to determine the power consumption of the design:

1. Processing > PowerPlay Power Analyzer Tool
2. A new tab will open, click on Start at the bottom left of tab.
3. In the compilation report tab, expand PowerPlay Power Analyzer folder found under table of contents. Then, select Summary.

Test the functionality of your implementation using ModelSim

2.2. 3-8 Decoder

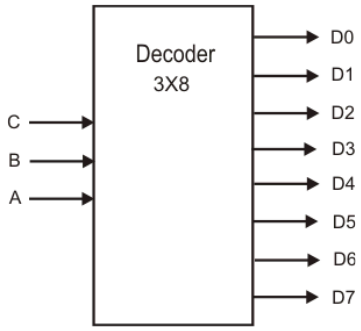

Figure 1. Decoder diagram

Create a new project and use the following VHDL code that corresponds to a 3-8 Decoder (See Figure 1):

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity Decoder is
port(
        A,B,C: in std_logic;
        D: out std_logic_vector(7 downto 0)
);
end entity;
architecture beh of Decoder is
signal v: std_logic_vector(2 downto 0);
begin
        v <= A & B & C;
        process(v)
        begin
                D <= (others => '0');
                case (v) is
                        when "000" => D(0) <= '1';
                        when "001" => D(1) <= '1';
                        when "010" => D(2) <= '1';
                        when "011" => D(3) <= '1';
                        when "100" => D(4) <= '1';
                        when "101" => D(5) <= '1';
                        when "110" => D(6) <= '1';
                        when "111" => D(7) <= '1';
                        when others => D <= (others => '0');
                end case;
        end process;
end architecture;
```

1. Verify the RTL and produce all necessary analysis files.
2. Check the power consumption.
3. Verify the functionality of the Decoder using ModelSim.

2.3. Array Search

In this exercise, we will implement an array search using array and for-loop. The array has predefined values from 0 to 9. The system has a single integer input as the value to be searched for and single integer output indicating the index in the array - if the value is found

Create a new VHDL file and paste the following code:

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

entity Array_Search is
port(
        x: in integer;
        index: out integer
);
end entity;
architecture beh of Array_Search is
type IntArr is array(1 to 10) of integer;
signal arr1: IntArr := (0,1,2,3,4,5,6,7,8,9);
begin
        process(x)
        begin
                for i in 1 to arr1'Length loop
                        if (x = arr1(i)) then
                                index <= i;
                                exit;
                        end if;
                        index <= -1;
                end loop;
        end process;
end architecture;
```

1. Verify the RTL and produce all necessary analysis files.
2. Check the power consumption.
3. Verify the functionality of the Array Search using ModelSim.

3. Practice Exercises

3.1. Multiplexer

Design and implement an 8-1 Multiplexer using case statements

3.2. Array Adder

Design and implement an Array Adder. An Array Adder zips two integer arrays with addition and stores the result in a third array.

3.3. Array-Mapped Increment

Design and implement an Array-Mapped Increment circuit that adds two to all elements in an array. If an array value exceeds 9, that array element is reset to zero.

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 1 and 2 will be requested, so maintain your project file for the snapshots of the work done.