Lab3: Intro to Structural VHDL
CDA 4203L
TA: Andrew White
Sept 10, Sept 15

Overview
The purpose of this lab is to introduce structural VHDL modeling.

Structural VHDL Introduction
A structural VHDL model is a description of a system in terms of the interconnection of its components, rather than a description of the behavior of the system. A structural model views the hardware as a schematic of the circuit. In this type of modeling, the components and the interconnections are visible, but the internal functions of the components are hidden.

VHDL uses the construct of components to define the interface of the elements used in a structural description. The components are first declared in the architecture before the reserved word "begin" (this area is referred to as the *declarative region* of the architecture). The signals used for interconnecting components are also declared within this region. A component is then placed into the architecture (referred to as *component instantiation*) after the reserved word "begin" (this area is referred to as the *architecture body*), and is connected to its input and output signals.

Component Declaration
The component declaration defines the interface of a component. Before a component can be used within an architecture, it must first be declared.

Syntax

component *component_name*
port (*inputs/outputs…* **);**
end component;

ex.

```
component C_multiplier
        port(   input1 : in std_logic_vector (7 downto 0);
                input2 : in std_logic _vector (7 downto 0);
                output : out std_logic _vector (14 downto 0));
end component;
```

Component Instantiation
A component instantiation statement represents an instance of a component in the architecture body of the architecture it appears. Labels are used to allow multiple instances of the same component within an architecture. The inputs and outputs of the component must be "associated" with signals for them to be used. Association of the inputs and/or outputs can be done in one of two ways: named or positional. **Only named association should be used for lab assignments.** The reason for this is that named

association explicitly associates signals and is not affected by changes in the input/output order within the port interface.

Syntax

Instantiation_Label : *component_name*
port map (*signal_of_component* => *signal_of_architecture,*
-- All input/output signals of a component that are to be
-- used must be associated to signals within the architecture
-- to be used, this is referred to as named association
another_signal_of_comp => *another_signal_of_arch* **);**

ex.

```
Mult0: c_multiplier
        port( input1 => mult_in1,
              input2 => mult_in2,
              output => output);
```

Each signal used to interconnect components that is not an input or output declared in the entity needs to be declared in the declaration *region* of the architecture. Each interconnection needs to have its own signal declared. Think of this as naming the wires used to wire the components together. Also, each instance of a component must have a unique label.

See attached VHDL files for example of an entire structural circuit.

Lab Procedure

Create a new project in the Xilinx ISE Tools.

Using a separate VHDL file for each, create a behavioral model and add to project each of the following single bit width gates:
AND
OR (can be 2 or 3 input)
Inverter

Using these basic gates, create a new Structural VHDL circuit that models the following equation:

$$F = \overline{AC} + \overline{C}D + \overline{B}E$$

For Synthesis, select the VHDL file that contains the structural model (this is referred to as the *Top Level Design*) in the **Sources in Project** window. Then in the **Processes for Current Source** window, select **Synthesize** and click **Process -> Run**.

For Simulation using Modelsim, create a new project and add all of the files created in steps 2 and 3 to the project. Click **Compile -> Compile All**.

Simulate only the top level design, and do so exhaustively (all possible combinations) using the **repeat** option with the **force** command and specifying each successive input to have twice the period as the previous. **Verify that the circuit has been implemented correctly.**

Lab Report
The lab report will consist of a cover sheet, the VHDL source files (4), and the simulation waveforms.

```
-- add_slice.vhd
-- Full Adder slice
-- Example for CDA 4203 Lab 3

library ieee;
use ieee.std_logic_1164.all;

entity add_slice is
        port ( input1, input2, carry_in : in std_logic;
                carry_out, output : out std_logic);
end add_slice;

architecture behav_example of add_slice is
begin

        process(input1,input2,carry_in)
        begin
                output <= (input1 XOR input2) XOR carry_in ;
                carry <= (input1 AND input2) OR (carry_in AND (input1 XOR input2));
        end process;

end behav_example;
```

```
-- This is an example of a structural architecture
-- For a 3 bit half adder made with component add_slice
-- Found in add_slice.vhd

library ieee;
use ieee.std_logic_1164.all;

entity lab3_example is
	port (A, B : in std_logic_vector(2 downto 0);
		Z : out std_logic_vector(3 downto 0));
end lab3_example;

architecture struct_example of lab3_example is
	-- Declarative Region of Architecture

	-- Component Declarations
	component add_slice
		port ( input1, input2, carry_in : in std_logic;
			carry_out, output : out std_logic);
	end component;

	-- Signal Declarations used inside architecture
	signal carry : std_logic_vector(1 downto 0);

begin
	-- Body of Architecture

	-- Instantiate an adder slice
	ADD0: add_slice
		port map( input1 => A(0),
			input2 => B(0),
			carry_in => '0',
			carry_out => carry(0),
			output => Z(0));

	ADD1: add_slice
		port map( input1 => A(1),
			input2 => B(1),
			carry_in => carry(0),
			carry_out => carry(1),
			output => Z(1));

	ADD2: add_slice
		port map( input1 => A(2),
			input2 => B(2),
			carry_in => carry(1),
			carry_out => Z(3),
```

```
                output => Z(2));

end struct_example;
```