ESE 382 Digital System Design Using VHDL and PLDs

Spring 20, Ken Short
February 19, 2020 1:07 pm

Laboratory 5: Behavioral Style Combinational Design Using Case, If, and Loop Statements - Gray Code to Binary Code Conversion

To be performed the week starting March 1st.

Prerequisite Reading
1. Chapter 5 of the text.

Purpose
The purpose of this laboratory is to provide you experience in using three different forms of sequential VHDL constructs in designing combinational circuits. You will describe the Gray Code to Binary Code Decoder from Laboratories 3 and 4 using behavioral style architectures.

The case, if, and loop statements are sequential and can only appear inside a process or a subprogram. In this design you will write descriptions containing a process statement in the architecture body and sequential statements inside the process statement. The entirety of a process statement, from its process label through all of its included sequential statements to the end process statement, constitutes a single concurrent statement.

Some designs are more easily described using one kind of statement than others. As you become more experience with VHDL you will become better at choosing, from the start, the best style for describing a particular design. You will see from the tasks in this laboratory that some of the statements are more appropriate (efficient) than others for a particular design description.

Design Tasks

Design Task 1: Gray Code to Binary Code Decoder Described Using a Case Statement and Scalar Inputs and Outputs
In this task you are to write a design description for the Gray Code to Binary Code Decoder that uses only a case statement. Do not use variables in your description. Refer to Laboratory 4, Design Task 2, for the basic definition of the function of the Gray Code to Binary Code Decoder. The entity name, inputs, outputs, and function remain the same. Name the architecture `gray_bin_case`.

Use don't cares to allow the synthesizer and place and route tools to generate minimal logic. To be efficient, your description needs to use aggregates and either type qualification or a local signal vector.

Functionally simulate your design using the self-checking testbench provided for Laboratory 4. Even though you have been provided a self-checking testbench, also visually verify the outputs in the waveform editor.

Your design description source file listing is required as part of your prelab that is due when you enter the laboratory.

Design Task 2: Gray Code to Binary Code Decoder Described Using a Case Statement and Vector Inputs and Outputs

In this task you are also to write a design description for the Gray Code to Binary Code Decoder that uses only a case statement. Do not use variables in your description. However, while the entity name and function remain the same, the inputs and outputs are to be declared as vectors. The input vector is named `g` and the output vector is named `b`. Name the architecture `gray_bin_case_vect`.

Use the self-checking testbench you modified for Design Tasks 3 and 4 of Laboratory 4 for this design. Examine this testbench in detail. Note that it includes verification of outputs for input combinations where the output is a don't care. This is what we want for functional simulation. If this testbench is used for post-synthesis or timing simulation, it could be modified so that outputs are not verified for input combinations that should generate don't care outputs.

Functionally simulate your design using the self-checking testbench you modified for Laboratory 4. Even though you have been provided a self-checking testbench, also visually verify the outputs in the waveform editor.

Your design description source file listing is required as part of your prelab that is due when you enter the laboratory.

Design Task 3: Gray Code to Binary Code Decoder Described Using an If Statement

Repeat Task 2, except write the design description using an If statement(s). Do not use variables in your description. Again, be sure to use don't cares to allow the synthesizer and place and route tools to generate minimal logic. Name the architecture `gray_bin_if_vect`.

Functionally simulate your design using the self-checking testbench you modified for Laboratory 4. Even though you have been provided a self-checking testbench, also visually verify the outputs in the waveform editor.

Your design description source file listing is required as part of your prelab that is due when you enter the laboratory.

Design Task 4: Gray Code to Binary Code Decoder Described Using a Loop Statement

As previously stated, different styles and different constructs in the behavioral style may be more or less efficient than other approaches depending on the function being implemented. When there

is a logical or arithmetic relationship between the outputs an inputs, an algorithm may be written that more concisely describes the design.

Write a design description for a 4-bit Gray to Binary Decoder with vector input and outputs using looping. This description will not be able to use don't cares to simplify the logic. You will need to use a variable(s) to implement the algorithm using a loop.

Looping is an advantageous approach when the function being implemented can be computed by examining the inputs to the entity in sequence or computing the outputs in sequence. Or, when the algorithm is of an iterative nature. When this is not the situation, a description using a loop can be difficult to write.

See the article on Gray to Binary conversion in the folder for this laboratory to get an idea of the algorithmic nature of the relationship. Once this algorithm is described in VHDL for a 4-bit Gray to Binary Decoder, it is trivial to modify the code to describe larger *n*-bit decoders Gray to Binary Decoders. Thus, the code for a 16-bit Gray to Binary Decoder is no larger than that for a 4-bit decoder.

Functionally simulate your design using the self-checking testbench you modified for Laboratory 4. Even though you have been provided a self-checking testbench, also visually verify the outputs in the waveform editor.

Your design description source file listing is required as part of your prelab that is due when you enter the laboratory.

Laboratory Tasks

Laboratory Task 1: Functional Simulation of Designs
Using Aldec Active-HDL create a new workspace named `lab05`. In this workspace you must create the four designs. Import your VHDL design description file and testbench for each design.

Functionally simulate each design and get a TA's signature verifying that each design is functionally correct.

Laboratory Task 2: Synthesize Your Design from Design Task 4
Synthesize your design descriptions for Design Task 4. View the synthesized logic. To accomplish this, from the **HDL Analyst** menu select **Technology,** then **Flattened to Gates View**. Use the information from the flattened to gates view and the place and route tool's Chip Report to determine the amount of logic required for this design.

You must submit, as part of your report, a copy of the flattened to gates view of your design.

Laboratory Task 3: Timing Simulation of Design from Design Task 4

You must implement your design for Design Task 4.

Using ispLEVER Classic, place and route the synthesized logic to fit a Lattice ispGAL22V10C-10LJ. In ispLEVER Classic, print the pre-fit equations, post-fit equations, and the chip report.

The place and route process generates a timing model from the EDIF file. Recall that the timing model file has the suffix `.vhq`. Perform a timing simulation of the timing model generated by ispLEVER Classic.

Laboratory Task 4: Device Programming
Program a ispGAL22V10C-10LJ and verify the functionality of your design description for Design Task 4.

Have a TA sign off whether your programmed PLD meets the design specification.

Questions
1. What rules insure that a process statement properly describes a combinational circuit, and not a circuit with latches or flip-flops (memory)?

2. For this decoder, which of the three behavioral approaches to describing the architecture leads to the most efficient description? Explain your answer.

3. Are the signal assignment statements inside a process sequential or concurrent statements?

4. In terms of a simulation, what causes your process to be executed? How long does a process in a design description take, in simulated time, to execute during a functional simulation?

5. What is the propagation delay for the designs for which you performed a timing simulation?

6. Assuming 20 ns delay between applying each input combination, for how many ns must the simulation be run for all possible input combinations to be applied by the testbench?