ESE 382 Digital System Design Using VHDL and PLDs

Spring 20, Ken Short
February 19, 2020 11:41 am

Laboratory 4: Dataflow Style Combinational Design Using Selected Signal Assignment, Conditional Signal Assignment, and Table Lookup - Gray Code to Binary Code Conversion

This laboratory is to be performed the week starting Feb. 23rd.

Prerequisite Reading

1. Chapter 4 of the text.
2. Help file documentation on *workspaces* in Active HDL Help Menu.

Purpose
As you saw in Laboratory 3, any combinational function can be described using a canonical sum-of-products (CSOP) Boolean expression or a canonical product-of-sums (CPOS) Boolean expression. This is a straight forward, though not concise, way of expressing a combinational function. Since the synthesizer and place-and-route tool minimize the CSOP and CPOS Boolean expressions, the implemented logic corresponds to the simplified sum-of-products or simplified product-of-sums, respectively, for the function.

There are actually only two forms of concurrent signal assignment statements: selected and conditional. A Boolean expression concurrent signal assignment statement is a simplified form of a conditional signal assignment statement. However, for instructional purposes we treat the Boolean expression form separately. As a result, three kinds of concurrent signal assignment statements are considered:

• Concurrent signal assignment statement using a Boolean expression
• Selected signal assignment statement
• Conditional signal assignment statement

Still another simplified form of conditional signal assignment is table lookup. In this laboratory, you will use selected signal assignment, conditional signal assignment and table lookup statements to implement different architectures for the Gray Code to Binary Code Decoder from Laboratory 3.

The Boolean expression concurrent signal assignment statement approach has an important limitation. If you have an application where there are input combinations where one or more outputs are don't cares, you would like for the synthesizer to be able to take advantage of this fact to possibly further minimize the logic. In VHDL, you cannot express input combinations where the outputs are don't cares using Boolean expressions.

Selected signal assignment statements, conditional signal assignment statements, and table lookup can specify input combinations where outputs are don't cares. So, we will also consider a Gray Code to Binary Code Decoder where some of the Gray code input combinations will never be applied to the decoder. So for those input combinations, the outputs are all don't cares.

You will create four VHDL design descriptions for 4-bit Gray Code to Binary Decoders. You will perform functional simulations for all of your design descriptions and a timing simulation for one of your designs. Finally, you will program a Lattice ispGAL22V10C-10LJ and verify its operation using a hardware test station.

Design Tasks

All of the designs that follow are basically implementations of the 4-bit Gray Code to Binary Code Decoder from Laboratory 3. However, there are variations in the specifications, entity declarations, and architectures.

Design Task 1: Selected Signal Assignment Approach

The first design must implement the design entity `gray_bin` defined in Laboratory 3. However, it must be done using a selected signal assignment.

Using Aldec Active-HDL, create a new workspace named `gray_to_binary_2`. In this workspace create a design named `gray_bin_selected`. The entity name remains `gray_bin`. The architecture must be named `selected`. You must write the design description using a selected signal assignment statement.

Functionally simulate your design using the self-checking testbench provided. Even though you have been provided a self-checking testbench, also visually verify the outputs in the waveform editor. When your design is working properly, make a change in your selected signal assignment statement that will cause an incorrect output. See if the self-checking testbench catches it and how it displays the error. Remove the error that you intentionally put in your design description before continuing.

Note that because it uses the `to_string` function, the testbench requires that the VHDL compiler be set for VHDL 2008 compliance. Remember to make this setting Design > Settings > Compilation > VHDL Standard version > VHDL 1076-2008. Also, be sure to Enable Debug in the same window.

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

Design Task 2: Selected Signal Assignment Approach with Don't Cares

Your second design must also implement the design entity `gray_bin` defined in Laboratory 3 by using a selected signal assignment. However, the application that uses this version of the `gray_bin` decoder has the rotary encoder attached to a shaft that can only turn through an angle of from 0 up to, but not including, 270 degrees. So, some rotary encoder output values will never

be generated in this application. Therefore, these rotary encoder outputs will never be input to design entity `gray_bin`. For all these outputs from the rotary encoder (inputs to `gray_bin`) the outputs of `gray_bin` are don't cares.

Create a new design named `gray_bin_selected_dc` in the existing workspace. The entity name remains `gray_bin`. The architecture must be named `selected_dc`. Write the design description using a selected signal assignment statement that takes advantage of the don't cares in the application, so that the synthesizer can produce the simplest logic.

Functionally simulate your design using the self-checking testbench provided. 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: Conditional Signal Assignment Approach with Don't Cares
The third design must implement the design entity `gray_bin` as defined in Design Task 2 of this laboratory. However, for this design the inputs and outputs of `gray_bin` must be defined in the entity declaration as vectors. The input is a 4-element std_logic_vector `g`. The output is a 4-element std_logic_vector `b`.

Create a new design named `gray_bin_conditional_dc` in the existing workspace. The entity name remains `gray_bin`. The architecture must be named `conditional_dc`. Write the design description using a conditional signal assignment statement that takes advantage of the don't cares in the application, so that the synthesizer can produce the simplest logic.

The self-checking testbench used in Design Tasks 1 and 2 must be modified to create a testbench for this design. The modification is necessary because the design entity's inputs and outputs are described as vectors. Name the modified testbench `gray_bin_vec_selfcheck_TB.vhd`. Functionally simulate your design using your modified self-checking testbench. 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 and modified testbench listing are required as part of your prelab that is due when you enter the laboratory.

Design Task 4: Table Lookup Approach with Don't Cares
The fourth design must also implement the design entity `gray_bin` defined in Design Task 3 of this laboratory. However, this design must be implemented as a table lookup.

Create a new design named `gray_bin_table_lookup_dc` in the existing workspace. The entity name remains `gray_bin`. The architecture must be named `table_lookup_dc`. Write the design description using a concurrent signal assignment statement that implements a table

lookup and takes advantage of the don't cares in the application, so that the synthesizer can produce the simplest logic.

Use the testbench you modified for Design Task 3 for this design task.

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: Selected Signal Assignment Approach

Using Aldec Active-HDL create a new workspace named `gray_to_binary_2`. In this workspace create a design named `gray_bin_selected`. Import your VHDL source file for the first design description and the testbench.

Compile your design description and the testbench. Functionally simulate your design using the testbench. The testbench provided is a self-checking testbench, but you still should carefully examine the waveforms produced to determine if they meet the system's requirements. If the outputs from your design are not correct, modify your design description until they are.

The testbench generates all 16 possible combinations of the four inputs.

Have a TA verify and sign whether the waveforms from your functional simulation meet the design specification.

Laboratory Task 2: Selected Signal Assignment Approach with Don't Cares

Repeat the process in Laboratory Task 1, for your second design description. Add this design to the existing workspace. Name the design `gray_bin_selected_dc`.

Have a TA verify and sign whether the waveforms from your functional simulation meet the design specification.

Laboratory Task 3: Conditional Signal Assignment Approach with Don't Cares

Repeat the process in Laboratory Task 2, for your third design description. Add this design to the existing workspace. Name the design `gray_bin_conditional_dc`. You must modify the testbench provided to work for this design, since the design entity's inputs and outputs are described as vectors.

Have a TA verify and sign whether the waveforms from your functional simulation meet the design specification.

Laboratory Task 4: Table Lookup Approach with Don't Cares

Repeat the process in Task 3, for your fourth design description. Add this design to the existing workspace. Name the design `gray_bin_table_lookup_dc`. Use the testbench you modified for Design Task 3.

Have a TA verify and sign whether the waveforms from your functional simulation meet the design specification.

Laboratory Task 5: Synthesis, Place and Route and Hardware Verification of a Design
For either your design from Design Task 3 or Design Task 4, use Synplify to synthesize the logic for a Lattice ispGAL22V10C-10LJ. Add the pin attributes given to you in the laboratory before you perform the synthesis. After synthesis is complete, from the **HDL Analyst** menu select **RTL,** then **Hierarchical View.** Print this diagram of the synthesized logic. Next from the **HDL Analyst** menu select **Technology,** then **Flattened to Gates View.** Print this diagram of the synthesized logic.

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

Perform a timing simulation of the timing model generated by ispLEVER Classic.

Have the TA verify and sign whether the waveforms from your timing simulation are correct.

Program an ispGAL22V10C-10LJ and verify its functional performance against what was predicted by your simulations.

Note all the actual output values from your design for the input combinations that are not valid inputs to the Gray Code to Binary Code Decoder.

Have the TA verify and sign whether your programmed SPLD meets the design specification.

Questions
1. What is the difference in the waveforms from the simulation of the two designs `gray_bin_selected` (Design Task 1) and `gray_bin_selected_dc` (Design Task 2)? What in your source files causes this difference?

2. The testbench used in Design Task 2 tells you for which input combinations the outputs were in error. How could you modify that testbench so that it also tells you what the actual output was and what it should have been for each output that was incorrect?

3. How could you modify the testbench used in Design Task 2 so that you do not get errors for the input combinations where you don't care what the outputs are?

4. Compare the HDL Analyst hierarchical view of the design you synthesized for this Laboratory with the HDL Analyst hierarchical view of your design from Laboratory 3. Are the diagrams identical? If not, how do they differ? Are they functionally the same? If not, how do they differ functionally?

5. Compare the pre-fit equations and post-fit equations from ispLEVER Classic for the design you synthesized for this laboratory with the equations from your design for Laboratory 3. Are the equations the same? If not, how do they differ?

6. Explain the differences in the output values when input combinations that are not valid inputs to the decoder are applied to your functional simulation and to your timing simulation. What determined these output values for each of your designs? If the rotary encoder providing inputs to your design generates only Gray codes for angles from 0 up to 270 degrees, does the design you synthesized meet the design requirements?