Main Design Project

Introduction

In order to gain some experience with using macros we will exploit some of the features of our boards to construct a counter that will count from 0 to 59 with the counts displayed in two of our four 7-segment displays. In addition we will instrument one of the sliding switches to enable counting with a light emitting diode (LED) indicating its state and a push button to reset the counters momentarily to zero. The DI01 board is constructed in such a fashion that individual segments of the four displays are connected in parallel and are turned on when the particular connection is grounded. The choice of which of the four displays to activate is made by bringing the chosen display's anode to a positive logic value. Read over the description of the DI01 module, in particular the description of the seven segment displays and how the individual segments combine to form numbers.

Macros

In your design you will need to use several macros one of which will be the 0 to 5 counter you constructed in last week's lab and another will be a routine to translate the four bit output from the decimal counters into the pattern of 0's and 1's corresponding to the activated segments on a 7 segment display. This will be the file "hex2led.vhd" which will be provided for you. A third macro, also provided for you, will divide the 50 MHz clock down to something that we can easily display and verify. On the basis of this macro you will write another (in vhdl) to generate a toggling frequency clock which will be an input to the macro multiplexing the two 4-bit output streams from your counter onto the 7-segment display. The last macro, designed by you, will be a multiplexer to carry the two bit streams representing the two decimal digits to our seven-segment displays.

Procedure

Open Project Navigator and start a new project choosing "schematic" as the highest level module. Verify that the Device Family is "Spartan2", the Device "xc2s30" and the Package "tq144".

The Counter

Find and add the source files for the counter you made in last weeks lab and turn it into a symbol. From "Symbols" select the counter macro CD4CE, connect signals from the clock and clear lines to both counters in parallel. Also run a wire connecting the CEO output from the CD4CE counter to the CE input of the CB4RE. Make all the appropriate connections to the input and output lines and vectors. After you are through your design should look something like that in Figure 1. Test this design by creating a Testbench Waveform and running the simulation program to see the response of the counter. Be sure to make the clock stream long enough so that more than 60 output counts are observed. This will ensure that the full functionality of the counter is tested. If your design is successful create a schematic symbol for your counter and verify that it is available as a symbol on the schematic entry page.

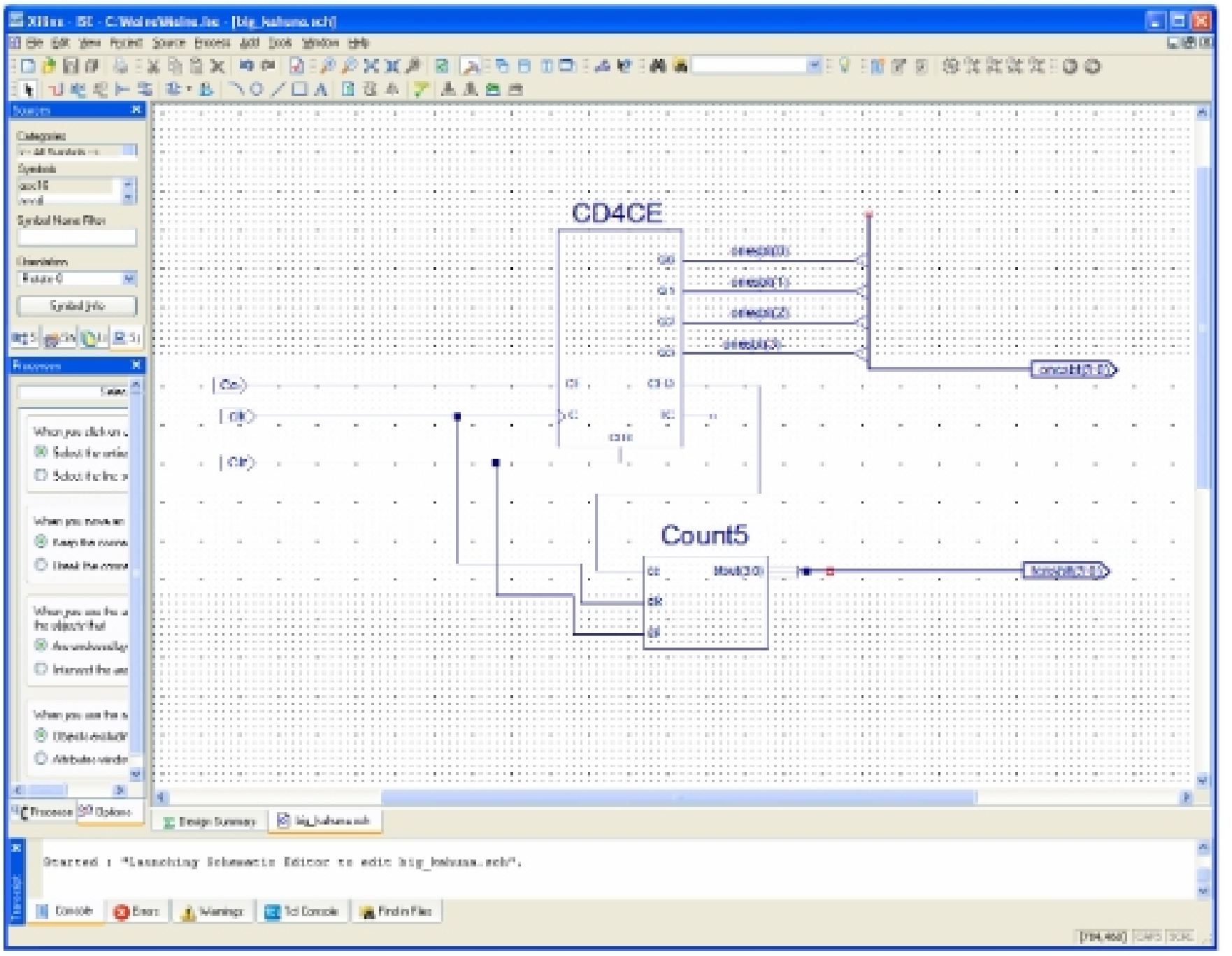

Figure 1: Counter Schematic Layout

The Hex to LED Converter

We will use VHDL to generate this macro.

1. In Project Navigator, select **Project→New Source**.
   The New Source dialog box opens.
2. Select the source type **VHDL Module**.
3. In the File Name field, type **hex2led**
4. Click **Next**.
   The hex2led component has a 4-bit input port named HEX and a 7-bit output port named LED. First enter the port named HEX as follows:
5. Click in the Port Name field and type **HEX**.
6. In the Direction field, set the direction to **in**.
7. In the MSB field, enter **3**, and in the LSB field, enter **0**.
8. Repeat the previous steps for the LED(6:0) output bus. Be sure that the direction is set to out.
9. Select **Next** to complete the Wizard session.

10. Select **Finish**. The "skeleton" HDL file opens in the ISE Text Editor.

In the HDL file, the ports are already declared and some of the basic file structure is already in place. Key words are displayed in blue, data types in red, comments in green and values in black.
Next we will use some synthesis templates to finish this design.

1. In Project Navigator, select Edit→Language Templates.
2. Locate the template called "7-segment display Hex conversion" for VHDL located under the Synthesis Constructs heading in Coding Examples/Misc.
3. To preview the Converter template, click the template in the hierarchy. The contents display in the right-hand pane.
4. Copy the contents of this template and add it to your hex2led.vhd file under the architecture begin statement.
5. Save the file by **File→Save**.
6. In Project Navigator, select hex2led.vhd in the Project window.

Double-click Check Syntax in the Processes for Current Source window. This launches the ISE Text Editor.
If no errors are found, create a schematic symbol of this file and make sure it is available in the Symbols are on the schematic entry page.

The Frequency Dividers

Your TA will tell you where you can pick up the HDL file for dividing the 50 MHz clock frequency by 2^{23}. Does the resulting frequency seem reasonable for your display (i.e. is there sufficient time between states for your eyes to distinguish between two adjacent states)? If not, examine the syntax and make an appropriate adjustment to the HDL file (you should need to change two instances of the number 23).

Only one of your two 4-bit outputs can be applied to the seven segment display at any instant. You will need a second frequency divider to toggle between these two outputs. The frequency should be high enough so that your eyes perceive the displays as continuous. How high a frequency is reasonable for this function? Examine the syntax of the file and then ***create*** another one to provide the frequency to toggle the displays. Do this by adding a new source (VHDL module) to your project and replace the all of the hdl source code with a copy of the code from your 2^{23} divider. Make the appropriate edits to this code (in addition to the two instances of the power of 2, there should be three instances of the new file's name).

The Multiplexer

Design a macro to take as inputs two four-bit streams of numbers and a toggle level and output one of the four-bit streams depending on the level of the toggle signal. In doing this you may find the function "M2_1" useful. Create waveforms and run a simulation to exercise this function and assure yourself of its proper operation.