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 with a push button to reset the counters momentarily to zero. The DIO1 board is constructed in such a fashion that individual segments of the four displays are connected in parallel and they 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 DIO1 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 the 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

Have the write-up for the last week's lab handy. Open Project Navigator and start a new project, choosing "schematic" as the highest level module. Verify that the Device Family is "Spartan2", the Device is "xc2s30" and the Package is "tq144".

The Counter

Find and add the source files for the counter you made in the last week's lab by double-clicking on "Add Existing Sources". To arrive at a symbol, go to the schematic and check for errors, by clicking "Check Schematic" under the tools menu; now create the symbol as described in the last week's write-up.

Create a new schematic, for a 0-to-59 counter, into which you need to bring in your symbol. From "Symbols" select next 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 Fig. 1. Note that that I/O for your symbol may be

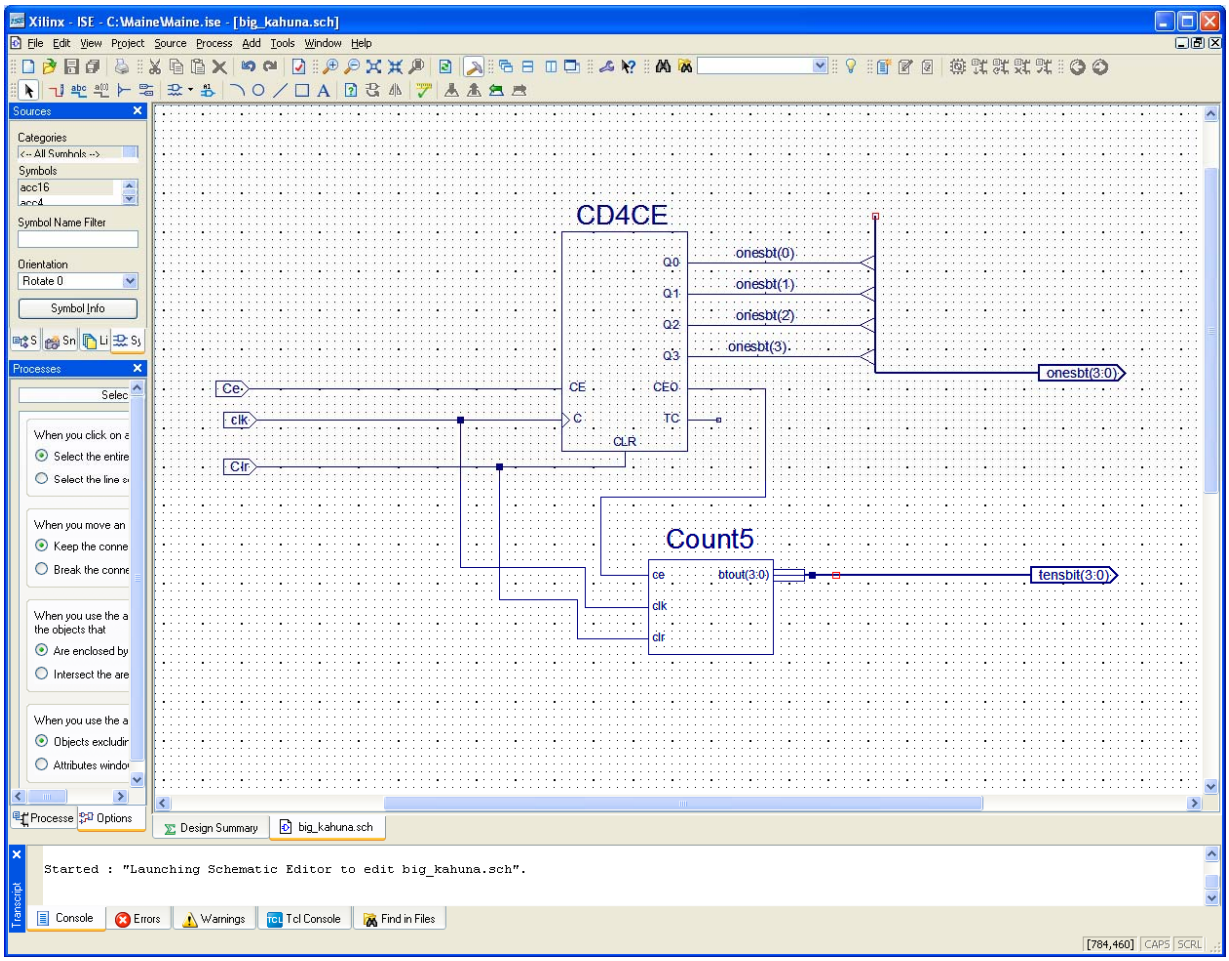

Figure 1: Counter Schematic Layout.

differently ordered than in Fig. 1. Test your 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 new counter and verify that it is available as a symbol on the schematic entry page.

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. Check "Bus".
8. In the MSB field enter **3**, and in the LSB field, enter **0**.
9. Repeat analogous steps for the LED(6:0) output bus. Be sure that the direction is set to **out**.
10. Select **Next** to complete the Wizard session.
11. 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 are in red, comments in green and values in black.

Next we will use some synthesis templates to finish this design.

1. In Project Navigator under the "Sources" tab, 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

Next, you will need to create two frequency dividers that will work together with the system clock running at the frequency of 50 MHz. Both dividers will act to produce periodic signals running at a much slower frequency than the system clock. The changes for one signal will be counted by your counter. Those counts should accumulate slowly enough so that your eyes are able to perceive changes in the counter displays due to individual clicks. The changes for the second signal will be exploited in manipulating the displays and should occur at a high enough frequency so that your eyes cannot catch up and perceive the manipulation.

Open up an HDL file for dividing the 50 MHz clock frequency by 2^{23}. That file is located in the subfolder Digilent of our class directory on drive C, under the name dividivi.vhd. You should not use the original file directly within your project, but rather save that file, possibly after necessary editing, in your project folder. Else you may copy contents of the original file into a file created by you and stored on the drive D. Does the frequency resulting from the division by 2^{23} seem reasonable when recording counts of the slowed down clock with your anticipated display, i.e. is there sufficient time between subsequent states for your eyes to distinguish between those states? (You do not want to

end up watching a blurred counter display.) 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 a seven segment display at any instant. You will need a second frequency divider file to toggle between these two outputs and connect them to different 7-segment LED displays. The frequency should be high enough so that your eyes do not notice the manipulation and perceive the displays as continuous. How high a frequency is reasonable for this function? After you examine the syntax of your first file, ***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 all of the hdl source code with a copy of the code from your original (or modified) divider by 2^{23}. 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 to change). Create symbols from both of your frequency dividers.

The Multiplexer

Design a macro to take as inputs two 4-bit streams of numbers and a toggle level and to output one of the 4-bit streams depending on the level of the toggle signal. A convenient way to accomplish this is to use the "M2_1" function, a basic 2-to-1 multiplexer, which is in your list of available symbols. Fig. 2 shows a fully completed multiplexer macro. If you have time, create waveforms and run a simulation to test your circuit and assure yourself of its proper operation. Again create a symbol.

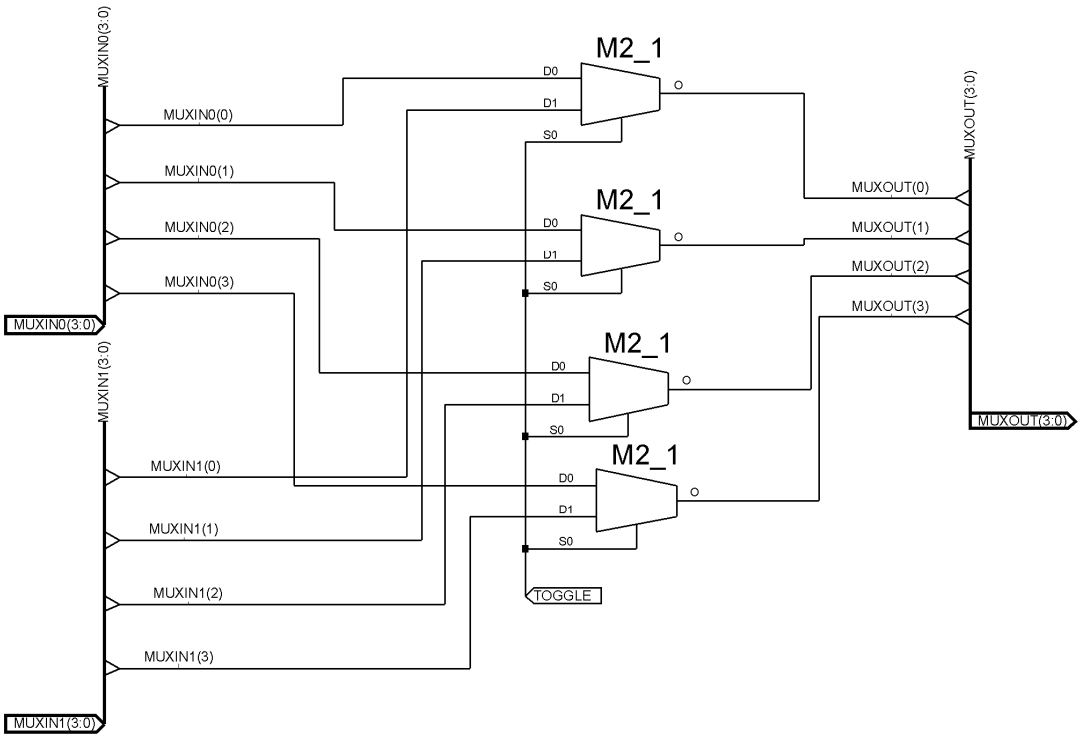

Figure 2: Multiplexer constructed from M2_1 units.

The Design

You now have all the pieces for the design, the counter, the multiplexer, the Hex-to-LED converter and the frequency reducers. Your final design should look something like that in Figure 3. Be sure you understand the operation of each of the functions named in the diagram. To comprehend how a 7-segment display works, see the DIO1 Manual.

If all is working well, you can start tying up the output pins via the user constraint file. Referring to the pin numbering given in the DIO1/D2XL pinout page (XLIO file), tie the clock to the 50MHz system clock (pin 91), the CE line to a selected switch and the CLR line to some button. To indicate that the CE line is on, route its output to the LED opposite the switch. The seven data lines from hex2led should tie to the cathodes of the 7-segment counter LEDs, lumped on the DIO1 board according to the 7 segments (A-G). The lines selecting the digit of the counter need to be tied to the anodes of the seven-segment counter LEDs, lumped on the board according to the displayed digit (1-4). To figure out which FPGA pin to assign to which I/O of your design, following the above, peruse the table in XLIO file and deduce which signal abbreviations, associated with pins, are of interest to you. See also markings on the DIO1 board.

Download your design onto FPGA. Exercise all the functions of your counter and show your instructor that it works as planned.

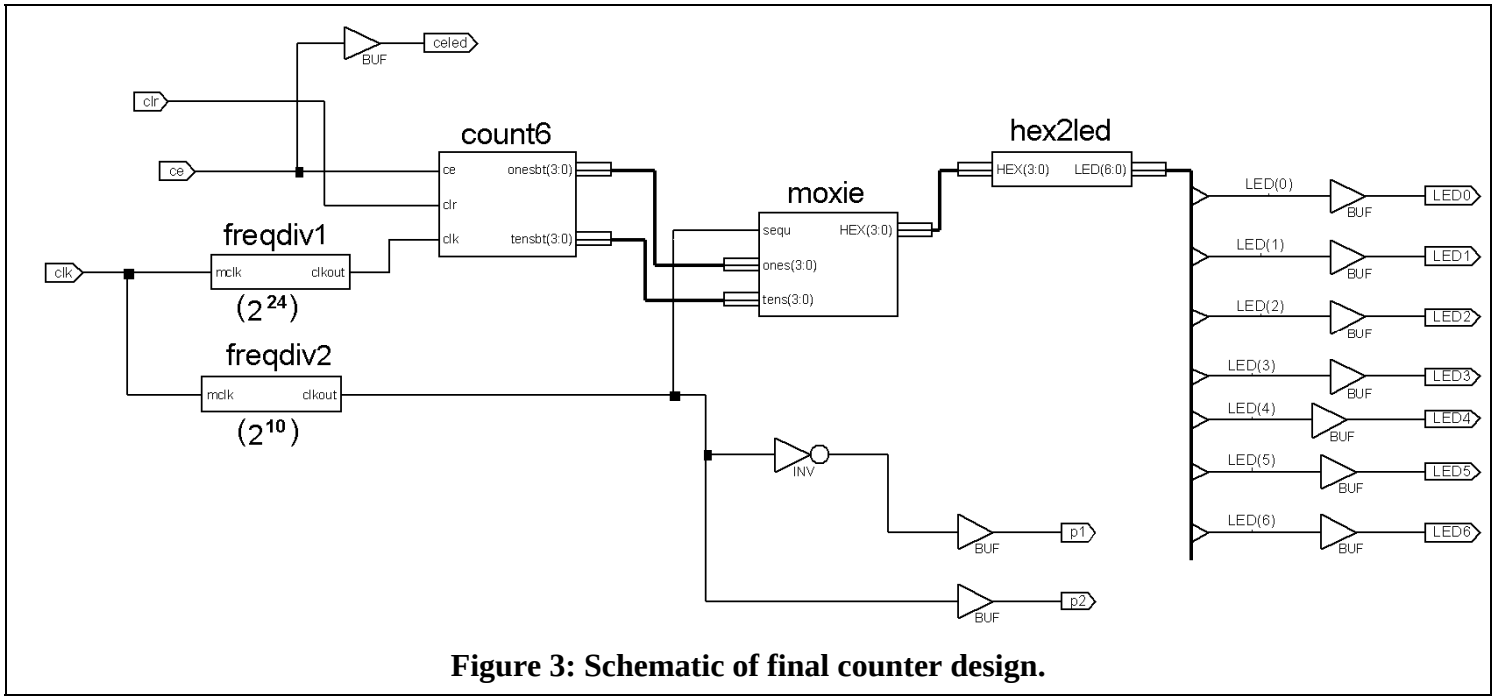

Figure 3: Schematic of final counter design.