CpE 487 Digital Design Lab

Lab 2: Four Digit Hex Counter

1. Implementing the Counter

In Lab1, we built a single-digit (4-bit) hex counter that displayed its value on one of the 7-segment displays. In this lab, we will extend our design to create a four digit (16-bit) counter that will display its value using all four 7-segment displays on the Nexys2 board.

1.1 Time Multiplexing the Displays

The first issue we have to deal with is how to display 4 different digits on the 7-segment displays if they all share the same cathode lines (CA~CG). We do this my time multiplexing the displays; that is, we only drive one display at a time. Suppose we turn on *display0* for a few milliseconds by enabling its common anode *AN0* and decoding *count (0~3)* to drive the cathode lines. Then we switch to *display1* for a few milliseconds by turning off *AN0*, turning on *AN1* and decoding *count(4~7)* to drive the cathode lines. We then shift to *display2* for a few milliseconds and then finally *display3* for a few milliseconds, after which we go back and start again at *display0*. Each digit is this illuminated only one quarter of the time, but if we do this multiplexing fast enough (at least 60 complete cycles per second), it will appear to our eye as if all four displays are continuously illuminated – each with their own 4-bits of information. Timing waveforms for this technique are shown in Figure 1.

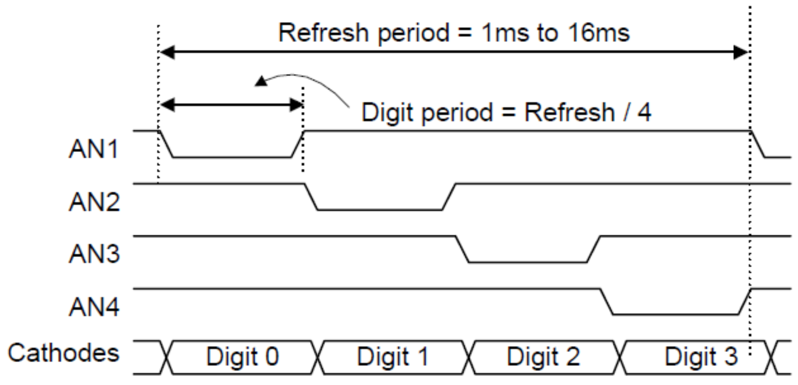


Figure 1 Four Digit 7-segment Display Timing Diagram

1.2 VHDL Design

a) Open your Lab1 project *hexcounter* in the ISE WebPack software. Use **File→Copy Project** to make a copy of your project named *hex4digit*. Note that even ‘though we have created a new project, the ISE software still has the old project *hexcounter* open. So, close the old project *hexcounter* and open the new project *hex4counter*.
b) Expand the *hexcount* hierarchy in the *Hierarchy* window and open up the source code to the *counter* module. Modify the *counter* module to generate a 16-bit *count* value using bits 23~38 of the binary counter as shown in Figure 2. We also create a second output

mpx. This will be used to multiplex the four 7-segment displays. We drive *mpx* using binary counter bits 17-18. These will generate a 0-3 count sequence counting at a frequency of $50\text{ MHz} \div 2^{17} \approx 381\text{ Hz}$. The sequence will repeat at a frequency of approx. $381 \div 4 = 95\ Hz$ which is fast enough to eliminate any visual flicker in the 7-segment displays.

c) Double-click **Check Syntax** in the *Process* window to make sure the *counter* module is free of syntax errors.

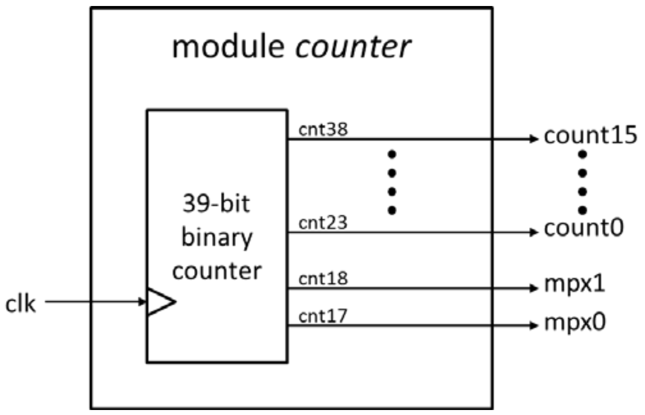


Figure 2 Modified *counter* module

d) Now open the source code for the top level module *hexcount*. Modify the *hexcount* module to implement the structure shown in Figure 3. The *mpx* output from the new *counter* module now drives the *dig* input of the *leddec* module. The 2-bit *mpx* signal is also used to select which 4-bits of the 16-bit *count* output should be sent to the *data* input of the *leddec* module. You will need to add some code to the *hexcount* module to perform this data multiplexing operation.

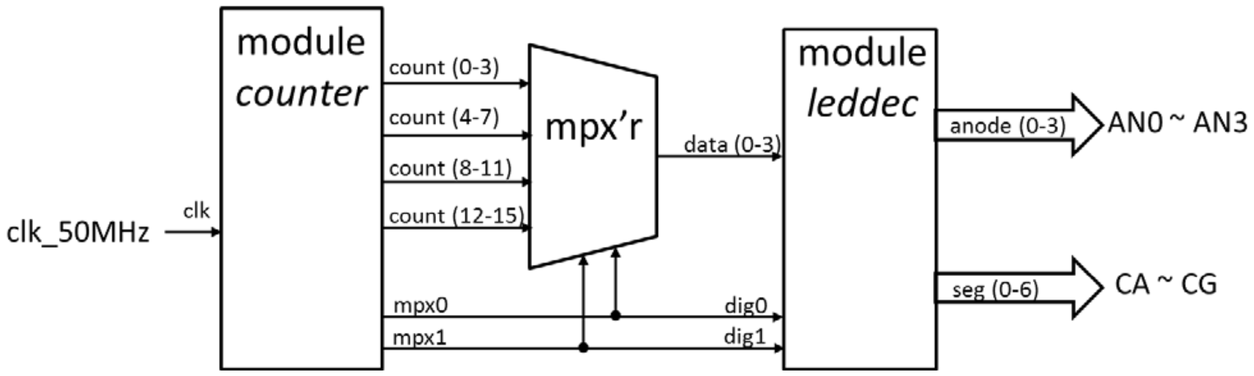


Figure 3 Modified *hexcount* module

1.3 Synthesize, Download & Run

Run the **Synthesis** process on the module *hexcount* to check for any VHDL design errors. Then run **Impement Design** and **Generate Programming File**. Once you have created the *hexcount.bit* configuration file, you can load it into the FPGA with the *Adept* software. Remember to browse to the correct project so that you download the new *hexcount.bit* – not the old one from Lab1.

2. Storing Configuration Code in the Platform Flash

When we download the FPGA configuration (.bit) file directly into the FPGA, the configuration data is stored in SRAM on the FPGA device. This SRAM is erased whenever power is removed or the FPGA board is reset. Once you have completed a design, you may wish to store your configuration data in the Platform FLASH ROM so that your design will be loaded and executed directly on power-up. To do this, we need to create a .mcs file to program the Platform FLASH.

Highlight the *hexcount* module in the Hierarchy window. Expand the **Configure Target Device** command in the *Process* window and double click on **Manage Configuration Project.** An *ISE iMPACT* window will pop up as in Figure 4.

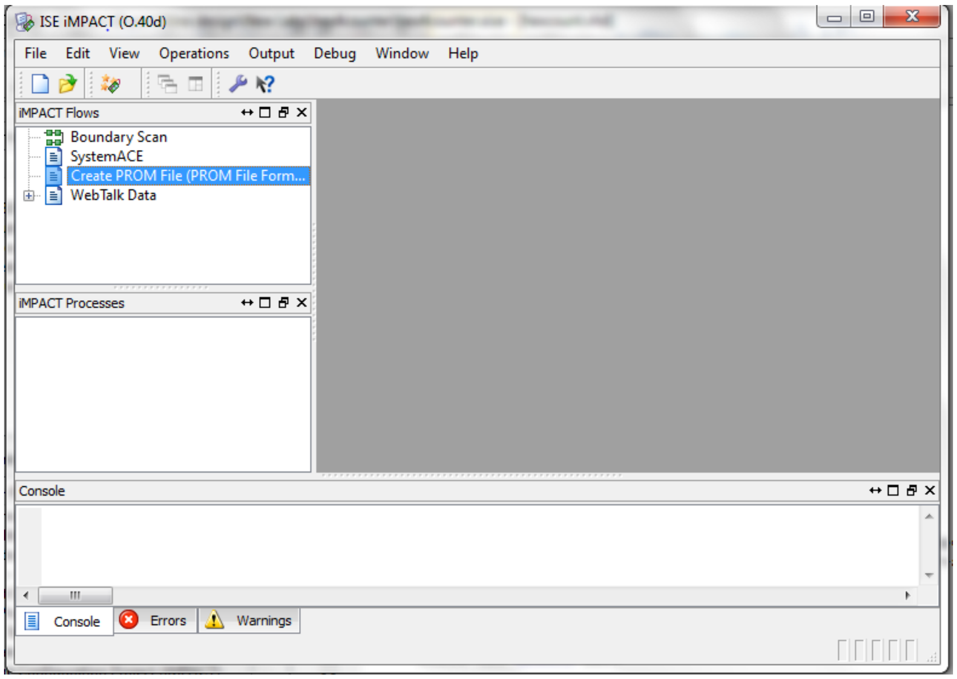


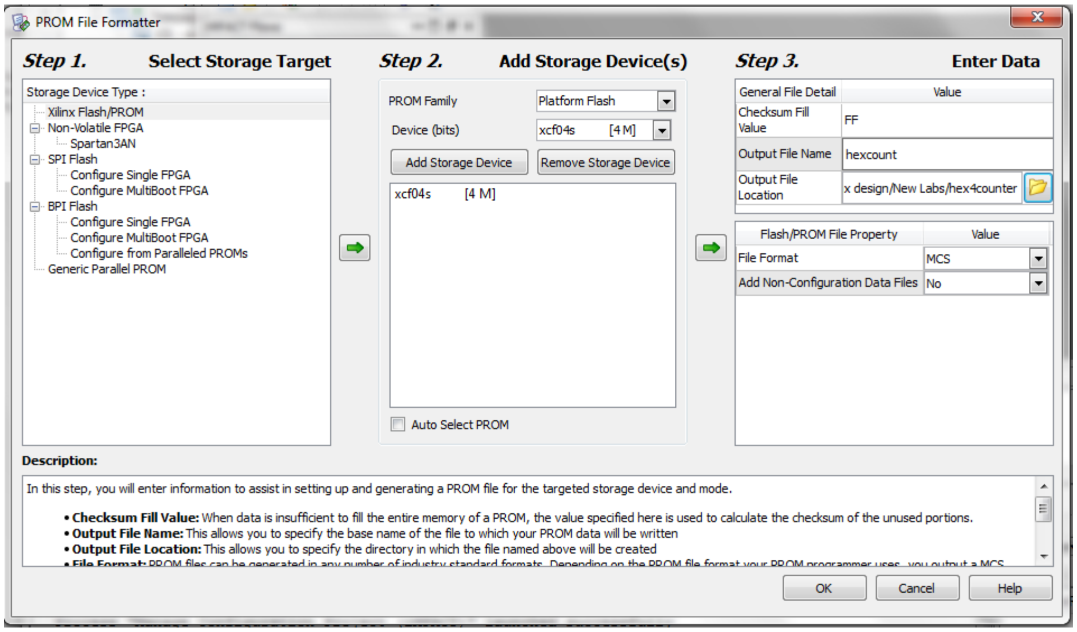
Figure 4 ISE Impact Window

Double click on **Create PROM File** in the *iMPACT Flows* window. This brings up the *PROM File Formatter* window.

Step 1: Select **Xilinx Flash/PROM** in the *Storage Device Type* and click the green arrow.

Step2: Select **Platform Flash** as the *PROM Family* and **xcf04s** as the *Device*. Click **Add Storage Device** and then click the green arrow.

Step 3: Enter *hexcount* as the *Output File Name*. Use the browser button to set the *Output File Location* to the current project (*hex4counter*) folder. The formatter window should now appear as in Figure 5.



Storage Device Type :
- Xilinx Flash/PROM
- Non-Volatile FPGA
  - Spartan3AN
- SPI Flash
  - Configure Single FPGA
  - Configure MultiBoot FPGA
- BPI Flash
  - Configure Single FPGA
  - Configure MultiBoot FPGA
  - Configure from Paralleled PROMs
- Generic Parallel PROM

PROM Family: Platform Flash
Device (bits): xcf04s [4 M]
Add Storage Device | Remove Storage Device
xcf04s [4 M]
Auto Select PROM

General File Detail	Value
Checksum Fill Value	FF
Output File Name	hexcount
Output File Location	x design/New Labs/hex4counter

Flash/PROM File Property	Value
File Format	MCS
Add Non-Configuration Data Files	No

Description:

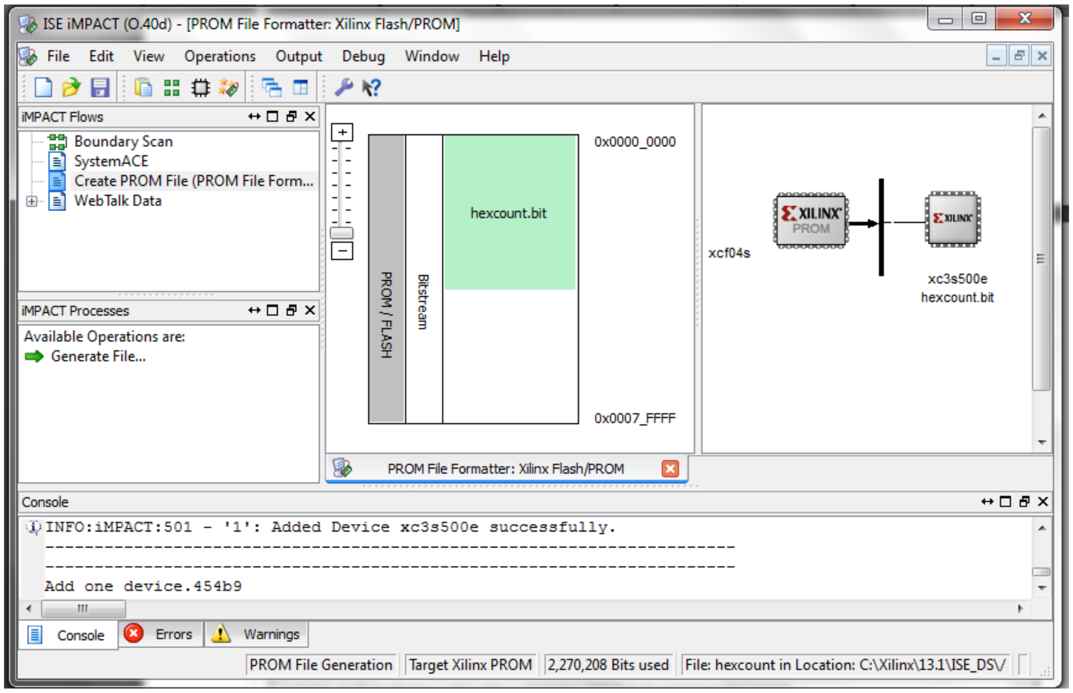
In this step, you will enter information to assist in setting up and generating a PROM file for the targeted storage device and mode.

- **Checksum Fill Value:** When data is insufficient to fill the entire memory of a PROM, the value specified here is used to calculate the checksum of the unused portions.
- **Output File Name:** This allows you to specify the base name of the file to which your PROM data will be written
- **Output File Location:** This allows you to specify the directory in which the file named above will be created
- **File Format:** PROM files can be generated in any number of industry standard formats. Depending on the PROM file format your PROM programmer uses, you output a MCS

OK | Cancel | Help

Figure 5 ISE Completed PROM File Formatter Window

Click **OK** at the bottom of the window. Click **OK** on the *Add Device* pop-up window and an *Add Device* browsing window appears. It should be pointing to your current project folder. Double click *hexcount.bit*. Another pop-up asks you if you want to add another device file. Select **No** and then click **OK**. The *ISE impact* Window should now appear as in Figure 6.



```
INFO:iMPACT:501 - '1': Added Device xc3s500e successfully.
----------------------------------------------------------------------
----------------------------------------------------------------------
Add one device.454b9
```

PROM File Generation | Target Xilinx PROM | 2,270,208 Bits used | File: hexcount in Location: C:\Xilinx\13.1\ISE_DS\

Figure 6 ISE Completed ISE impact Window

Double-click **Generate File** in the *iMPACT Processes Window*. This generates the *hexcount.mcs* file and gives you a blue “Generate Succeeded” pop-up when complete.

You can now use the *Adept* software to program the Platform Flash. Select the *Config* tab and use the **Browse** button on the PROM line to select the file he*xcount.mcs*. Click the **Program** button. It may take about one minute to complete the programming process. Once the programming is complete, push the reset button on the Nexys2 board. This will download your FPGA configuration from the Platform FLASH and begin executing four digit counter. Try turning the power off to your Nexys2 board. Turn it on again and your program should once again load and start running.