

Introduction to Sequence Detectors and CPLDs

Introduction

In this lab we will review the steps to create and compile a project on programmable logic using ISE and we will introduce the required steps to download the project into a chip. The programmable device you are you going to use is the Coolrunner CPLD.



In front of you is a black plastic board with two circuit boards on it. There should be labels. One should say FPGA and the other CPLD. We are going to use the CPLD board today.

We are using a Coolrunner XPLA CPLD mounted on a Digilent demo board like the one in the picture above. The CPLD is the square 44-pin chip in the middle of the board. Locate the power strip for the two boards and switch the power on. If the CPLD was previously programmed, you may see some lights begin to flash. These power strips should be left off when the boards are not in use to avoid unnecessary wear and tear. Next; locate the A/B switch box on top of the computer and ensure it is set to CPLD.

Goals

After completing this lab, you will be able to:

- Perform the basic operations to create, synthesize, simulate and download a project into a CPLD.
- Recognize basic characteristics of a CPLD
- Identify capabilities of XCR development board
- To use behavioral level of abstraction to describe a sequence detector.

Design Description : Sequence Detectors

Sequence detection is the act of recognizing a predefined series of inputs. Note that this is a sequential circuit; therefore we'll need a clock. Since the XCR board we are working with doesn't include a clock, we will use the slowest clock we have: your finger!

Create a sequence detector using VHDL that has the following characteristics:

- Serial input X
- The sequence to be detected is 10010.
- Five bits wide output Z with value 00000 when no sequence is detected, 00001 when the first bit of the sequence is detected, 00010 for the second bit, 00100 for the third bit, 01000 for the fourth and 10000 when the whole sequence is detected.
- If input Y is one, the system is reset to output 00000.

Design Analysis

The sequence detector created here will detect the sequence 10010. As seen in classes the design must start with a state diagram as shown in figure 1.

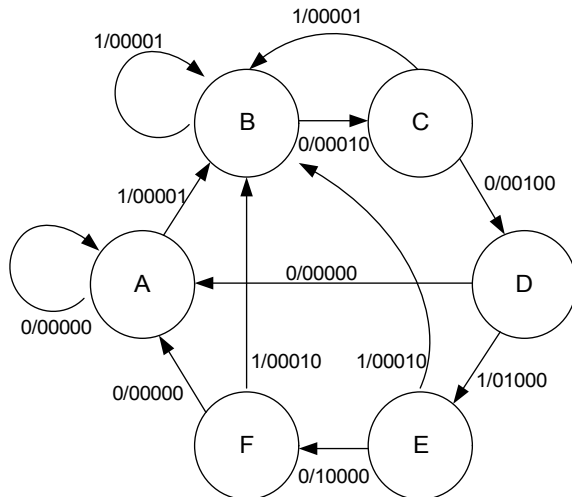


Figure 1; State diagram

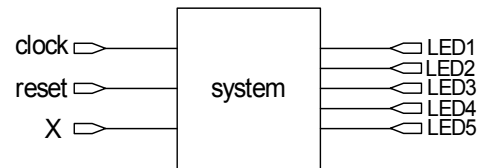


Figure 2; Block diagram

A basic block diagram of the sequence detector is shown in figure 2.

A basic introduction to CPLDs can be found in the lecture notes of this lab. Make sure you go through it since you will need that information to answer some of the questions in this lab.

Start the Project Navigator and Create the Project

Step 1

1. Open ISE software; Go to **Start Menu** → **Programs** → **Xilinx ISE 5** → **Project Navigator**. (or you can also look for the ISE icon on your desktop)
2. Create the sequence_detector project; In the Project Navigator, select **File** → **New Project** and setup options as in figure 3.
3. Click **OK**.

Note: It is essential to choose the correct CPLD part number for the project to work. There are ways to change this part number after creating the project, but it is just easier to choose the correct one at a first time.

4. Acquire files for the project; Download the files sequence.vhd and sequence_tb.vhd provided at the lab website and save them in the directory **C:/summer04/student_name/sequence**. Make sure you are saving them without a ".txt" extension but a ".vhd" extension.
5. Add VHDL code to the project; Go to **Project** → **Add source**, select the sequence.vhd (vhd module) and sequence_tb.vhd (test bench associated to sequence.vhd).

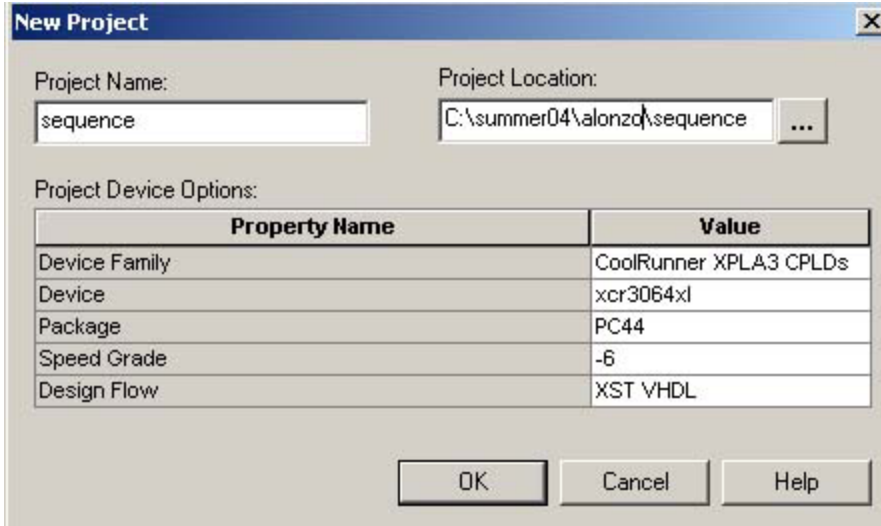


Figure 3; Creation of a project. The format for **Project Location** must be "C:\summer04\student_name\"

Synthesize and Simulate the System

Step 2

1. Synthesize the project; Highlight the file sequence.vhd in the **Sources in Project** window and double click on **Synthesize** in the **Processes for current sources** window. Make sure there are no warnings or errors.
2. Run behavioral simulation; Highlight the file sequence_tb.vhd in the **Sources in Project** window and double click on **Simulate Behavioral VHDL Model** in the **Processes for current sources** window.
3. Verify the operation of the sequence detector by looking at the input and output signals in the **Wave** window (figure 4) from the simulator.

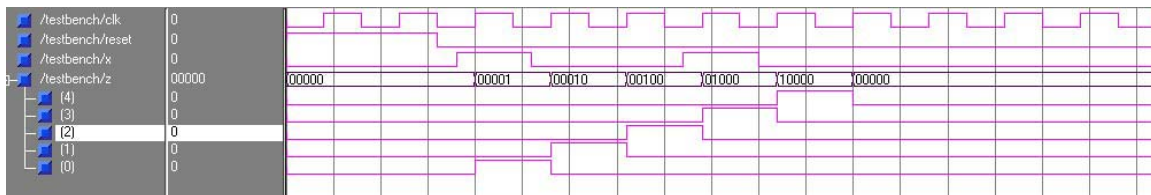


Figure 4; Simulation output.

Note: This current simulation says that if the required sequence is accompanied with a low reset and rising clock edges on each instance of the bit, the result will be a high output on Z. It says nothing about what will happen given any other set of inputs.

Note: Z is a bus; to expand the bus into its individual signals click on the "+" symbol by the side of the signal's name. Also; sometimes it is useful to change the radix of the signal. Right click on the bus Z and change the radix of the signal to decimal or hexadecimal.

Explore a Different Input Case through Simulation Step 3

1. Open the test bench; Double click on the file sequence_tb.vhd in the **Sources in Project** window to open the file.
2. Modify input sequence; Modify lines below the comment – *Following lines specify the input sequence of the simulation* – so that the input sequence is now 001110010.

Implementing the Design on the XCR board Step 4

In this section we will map the three inputs of the project to three switches in the board and the five outputs to five LEDs. The chip on the XCR board is a Coolrunner XCR3064xl (datasheet can be found at the lab website) with 44 pins of which you must choose 8.

1. Finding CPLD pins available; Look at the board's datasheet (it can be download from the lab website) and find table 3.
2. Creating a User Constraints file; Download the file sequence.ucf from the lab website and complete it using the information from table 3. Make sure you download the file with extension ".ucf" and NOT ".txt". You can edit the file using any text editor.
3. Adding User Constraints file to the project; Go to **Project** → **Add source**, select sequence.ucf file.
4. Create programming file; Highlight sequence.vhd file in the **Sources in Project** window and double click in **Generate Programming File** in the **Processes for Current Source** window.
5. Reading implementation results; Expand process **Implement Design** on **Processes for Current Source**. Expand **Fit** process under **Implement Design** process and double click on **Fitter Report**. Answer question 4 based on the information of this report.

Note: Figure 5 shows a brief explanation of what is happening when you synthesize a project.

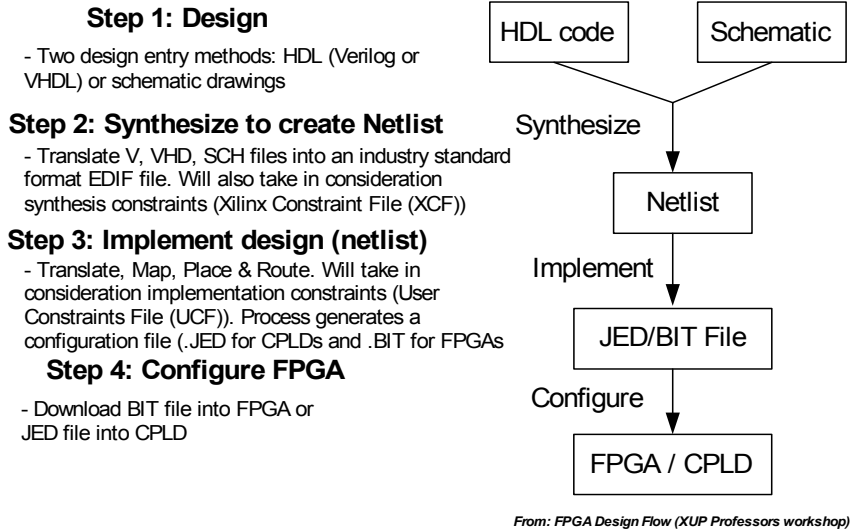
Note: By double click on the sequence.ucf file in the **Sources in Project** window a graphical interface for entering pins information can be seen and modified if necessary.

Note: A graphical view of the chip and its pin assignment can be obtained by: highlight sequence.vhd in **Sources in Project** window; expand **Implement Design** under **Processes in Project** window; expand **Fit** under **Implement Design** process and double click on **View Fitted Design (Chip Viewer)**.

Note: At this point make sure you have the power to the CPLD board switched on (turn the power strip and make sure black slide switch next to the parallel cable connector is set to EXT and the A/B switchbox on the top of the computer is set to CPLD).

6. Launching iMPACT; Highlight sequence.vhd in **Sources in Project** window; Expand **Generate Programming File** tasks on the **Processes for Current Source** window and double click on **Configure Device (iMPACT)**. Choose **Configure Devices** → **Next** → **Boundary-Scan Mode** → **Next** → **Automatically connect to cable and identify Boundary-Scan chain** → **Finish** → **Ok**. An **Assign New Configuration File** window must be the result of this operation.

Xilinx Design Process



From: FPGA Design Flow (XUP Professors workshop)

Figure 5; Xilinx Design Process

Note: iMPACT stands for Intelligent Multi-purpose Programming And Configuration Tool. Note that you can only have one instance of iMPACT opened. If you have more than one you will an error indicating problems in the communication with the chip.

- Programming the device; Browse for the file sequence.jed in your project directory using the resulting window from previous step. Click **Ok**. Click on the Xilinx chip to highlight it and go to **Operations** → **Program**. Figure 6 shows the result of this operation and the options that must be selected. Click **Ok**.

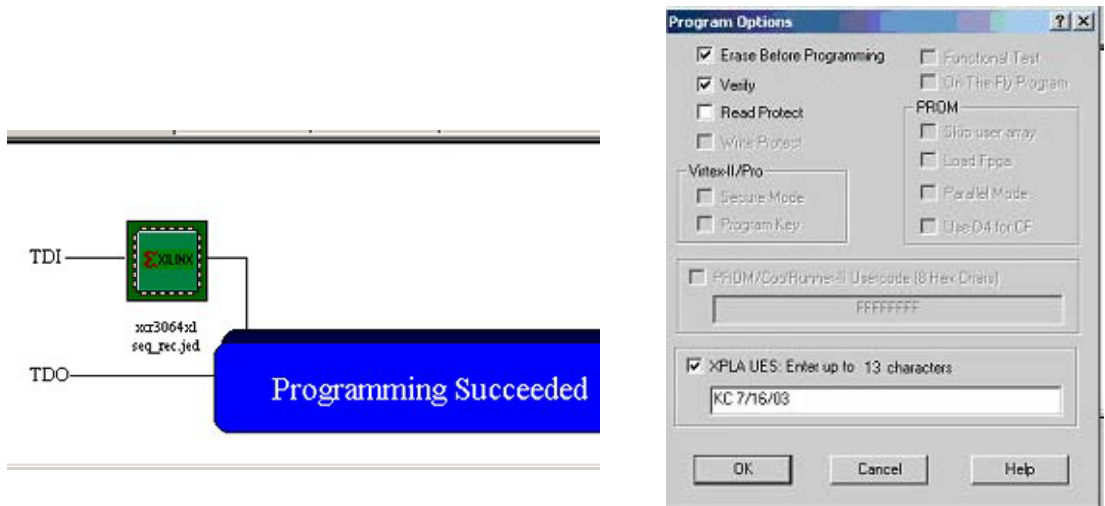


Figure 6; iMPACT window and options for CPLD configuration.

Note: By checking the option **Read Protect** on **Program Options** window (figure 6) we disable the CPLD design from being read back. This is particular useful for intellectual property protection. At the bottom of **Program Option** window there is a check for XPLA UES. UES

stands for user electronic signature. A message (i.e. name and date) can be recorded on the chip for future references.

Project Testing

Step 5

1. Reset the board; Set the switch designed for the reset signal to high (push it away from you).
2. Test reset state; With the reset switch set up to high, change input X (by manipulating switch assigned to signal X) and generate several clock cycles (by pushing the button assigned to the clock signal).
3. Looking for a sequence; Set the reset signal to low. Give a value to the input signal X by manipulating the corresponding switch and generate a clock cycle (by pushing the corresponding button) to make the system read the input X.
4. Demonstrating your system to your T.A.; Introduce a wrong sequence to the system and show the corresponding state changes according to your state diagram. Introduce a right sequence and show the corresponding state changes according to your state diagram. Ask your T.A. to sign your final report.
5. Testing CPLD memory capacity; Switch the power off and on to the power strip.

Final Task

Step 6

1. Design a system: Design a system able to recognize the sequence 00101.
2. Implementing your design: Follow steps 1 to 5 and implement your design on the XCR board. Demonstrate your system to your T.A and ask him/her to sign your final report.

Bonus Activity

Step 7

The VHDL code given for the sequence detector (sequence.vhd) contains several processes. From what we have seen in classes we know that all of those processes are executed in parallel and within the processes, each instruction is executed in a sequential way. The first of the process describes the reset signal behavior. It is an asynchronous reset.

1. Change your design of Step 6 so that reset becomes synchronous (for the system to acknowledge the reset signal, a rising clock edge must occur)