University of Tehran
College of Engineering
School of Electrical & Computer Engineering

Experiment 3
Sessions 5, 6, 7

Sequential Synthesis and FPGA Programming

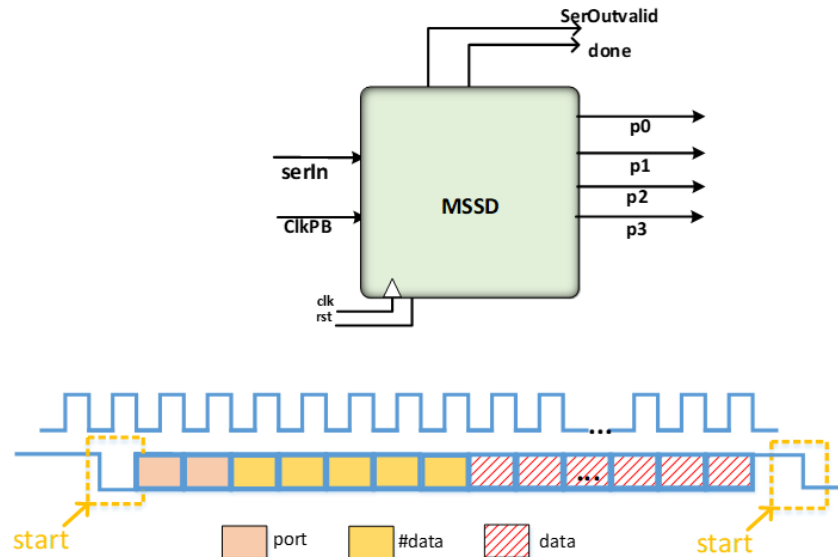### Digital Logic Laboratory
ECE 045, ECE 895
Laboratory Manual

Spring 1404

Contents

Figure 1: Multi-channel Serial Transmitter

Introduction

The first goal of this experiment is to introduce the concepts of state machines that are mostly used for controllers. The second goal is to get familiar with FPGA devices and implementation.

By the end of this experiment, you will learn:

- Concepts of state machines and Sequence detectors

- Huffman coding style

- Design simulation

- Synthesis

- FPGA programming and implementation

1 Multi-channel Serial Transmitter

In this experiment, you will design a Multi-channel Synchronous Serial communication Demultiplexer referred to as MSSD. The source provides a stream of 1-bit data that includes the size of the data to be transferred, the destination port, and the actual data.

The top-level view and ports of this design are shown in figure 1. Serial bits of data appear on the *serIn* input of MSSD. Normally, in *"no transmission"* mode, *serIn* is 1. Transmission begins when *serIn* makes a 1 to 0 transition. The 0 value bit marks the beginning of the transmission and has

Figure 2: Overall view of the design
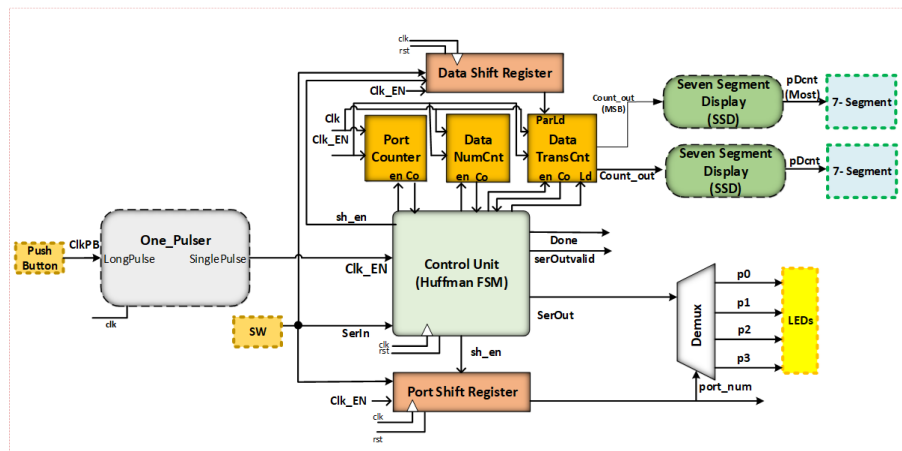
no other information. The two bits that follow are the port number, p, MSB first in time. The next 5 bits are the number of bits, n, MSB comes first in time. With $1 + 2 + 5 + n$ clock cycles after *serIn* becomes 0, it will return to 1 and another transmission begins with another start-bit. Data on *serIn* are synchronized with the MSSD clock, clk. This demultiplexer extracts the destination port and the number of bits that will go to the destination port. Following this, the next n bits will be transmitted to the destination port. Four output ports, px, will be used to transfer the data of each destination and one output port will be used to show the destination port's data count on the seven-segments (pDcnt).

In addition to these output ports, a *done* signal, and a *serOutvalid* signal are the outputs of MSSD. The *done* signal becomes 1 when the data transmission to the specified port is completed.

2 RTL Design

figure 2 shows the RTL design of this experiment. As shown six main components are needed for describing this design.

2.1 Onepulser

The one-pulser module provides a clock-enable input for the controller of this design. This input (clkEn) is used for controlling the clock when the circuit is implemented on an FPGA board. The one-pulser connects to a push-button on your board (clkPB) and when pressed it creates a single pulse that is synchronized with the system clock. The output of this circuit connects to the clock enable input (clkEn) of the sequence detector, shift registers, and counters.

1. Write the Verilog description of the one-pulser module.

2. Test your design in Modelsim.

2.2 Finite State Machine and the counters

The second part of this experiment is a Huffman style FSM. When the clock enable push-button is pushed, the FSM checks its input and decides which state to go based on the value of the *serIn*. The sequence of the state machines will be based on the start and end flow of figure 1. The push-button is pressed for every input that the state machine receives.

The FSM waits for the *serIn* to become one and after that, it again waits for it to be zero to start sending data. In the next state, the port number will be extracted. This requires the port counter to be enabled and wait in this state until all port bits are extracted. In the next state, the number of bits, *n*, will be extracted. This state takes five clock cycles, and then the load value of the data transfer counter is ready. After that, the data transfer counter will be enabled and will count for the next *n* consecutive clock cycles. During this state, the *serOutvalid* remains one. The signal *done* will be set in the last state of the controller.

1. Show the state diagram of the sequence detector.

2. Write the Verilog description of the FSM module.

3. As mentioned, three counters are required: one for counting the port detection clock cycles, another for counting five clock cycles to determine the number of data bits (*n*), and the last one for counting cycles for transferring data bits. These counters are referred to as the *port counter*, *data number counter*, and *data transfer counter*, respectively. Please write the Verilog description of the counter modules.

2.3 Shift Registers and Demultiplexer

To extract the port number, you need a shift register unit that shifts in and stores the value of the serial input for two clock cycles. This port number will be used as the select input of the demultiplexer of figure 2.

Four channels of the MSSD module will be selected for sending the data based on the port number. This will be done using a demultiplexer with p0 to p3 output ports. In addition to the data, the design will show the number of transmitted data on a seven-segment display. For this, you need another shift register to store the number of data bits, *n*. The parallel output of this shift register is used as the load value of the data transfer counter. The data transfer counter counts down from *n* to zero and shows the number of remaining data bits in each transmission.

1. Write the Verilog description of the shift register modules.

2. Write the Verilog description of the demultiplexer module.

2.4 Seven Segment Display

The last part of this experiment is a display module that is used for displaying the counter output on the seven segments of your FPGA board. Seven-segment receives a 4-bit input and displays the HEX value on its 7-bit output.

1. Write the Verilog description of the seven-segment display module.

Design Synthesis and FPGA Programming

3 MSSD Implementation

In this part, you will use the DE1 development board for your design implementation. Before implementing your design, make a top-level Verilog description that includes all six parts (one-pulser, FSM, counters, shift registers, demuxe, and the seven-segment display modules) connected together.

1. Make a Quartus project.

2. Add the top-level Verilog codes to your project.

3. Connect the main FPGA clock to the clock input of your circuit.

4. Connect a switch key to the *serIn* of the serial transmitter circuit, another push-button to the input of the one-pulser circuit.

5. Perform the synthesis of your design.

6. Program the Cyclone II device.

7. For the output display, use four output LEDs for displaying p0 to p3 serial data. Use two other LEDs for the *done* and *serOutvalid* signals. Use two seven-segments for displaying the HEX format of pDcnt. The seven-segment is dedicated to displaying the destination port's data count.

8. When applying a serial input, change the switch key of the *serIn* according to the value you want on the input and then press the ClkPB push-button once.

Appendices

Appendix A   Using Quartus II

A.1   Create the Project

1. Click on *File ▷ New Project Wizard*

2. Create an appropriate directory for your project and complete the form

3. Select the FPGA device as *Cyclone EP2C20F484C7* and then click *Finish*

4. From *File ▷ New* select the *Veriog HDL File*

5. Write your Verilog code in this window

A.2   Compilation

- Select *Processing ▷ Start Compilation*

- Click on quick shortcut ▶

There may be lots of warnings and some errors after compilation. The warnings are not so important while the errors should be completely removed.

A.3   Pin Assignment

After successfully compiling your design, you should set your design with physical pins of the Cyclone II FPGA on DE1 board. You can find the position and the index of each pin from the DE1 user manual.
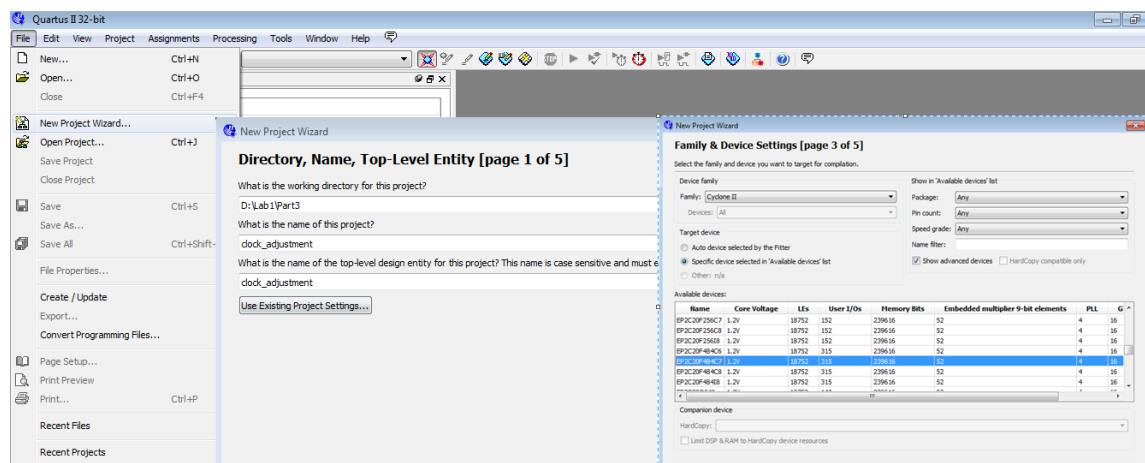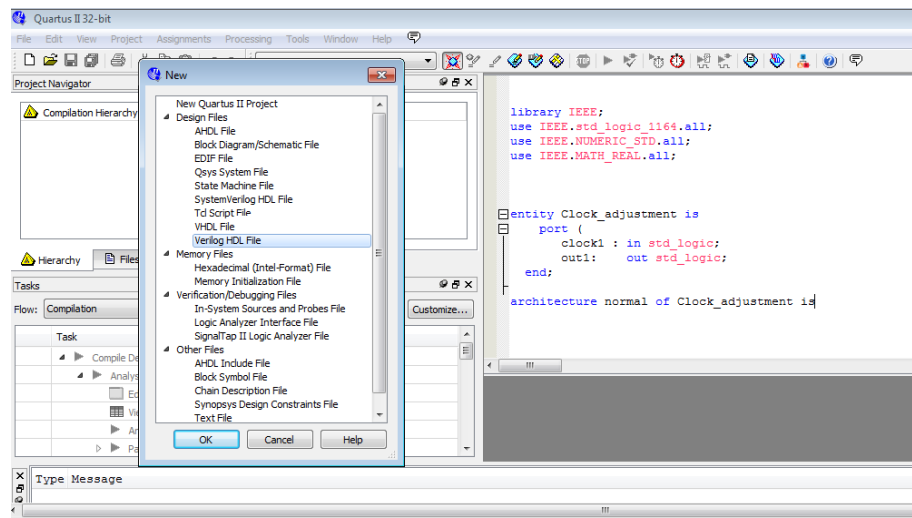
Figure 3: Creating new project

Figure 4: Writing Verilog HDL code

From *Assignments ▷ Pin Planner*, Pin Planner window will appear and you can select the corresponding locations from the list. When you are done with this, recompile your project.

A.4 Program the Design

1. Click on the ⬆ icon on the toolbar to open *Programmer*

2. Choose *USB Blaster* from the *Hardware setup* in the new window

3. Click on the start button

Your design will be programmed on the board when it is finished.

A.5 Examine the Timing and Resources

Now you can examine the resource usage of your design from *Processing ▷ Compilation Report* or by clicking on quick shortcut ⬇, and the timing from the *Tools ▷ TimeQuest Timing Analyzer*.

Acknowledgment

This lab manual was prepared and developed by Katayoon Basharkhah, Ph.D. student of Digital Systems at the University of Tehran, under the supervision of Professor Zain Navabi.

This manual has been revised and edited by Zahra Mahdavi Ph.D. student of Digital Systems at the University of Tehran, Zahra Hojati, and Iman Rasouli undergraduate students of Electrical Engineering at the University of Tehran.

Figure 5: Pin Planner window