Final Project, Part I: Keypad Scanner

Due on April 29th, 11:59 pm

Introduction:

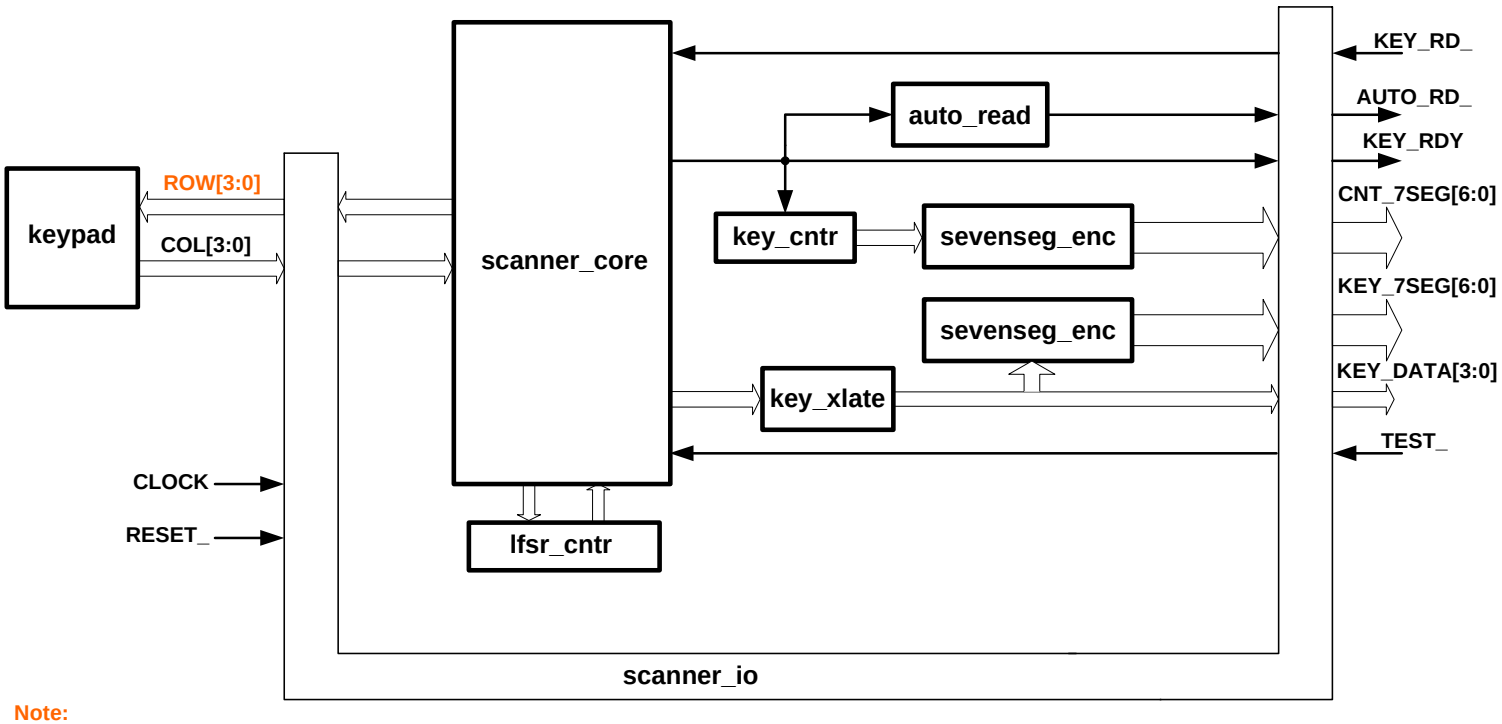
The objectives of Part I of the Final Project are:

- RTL coding of a sequential state machine to implement a keypad scanner
- Utilize Linear Feedback Shift Register (LFSR) counters
- Utilize a BFM in the test bench
- logic synthesis
- place and route of the netlist for implementation in an FPGA
- Thorough testing of the design in its RTL, gate-level netlist and hardware abstractions.

Final Project, Part I Description:

Design a state machine to scan and debounce a 16-key keypad. Use Verilog HDL to implement and verify the design. Use ***Modelsim*** for simulation and ***Quartus II*** for synthesis, place and route. Target the implementation to the Altera DE2 board to demonstrate that the scanner is fully functional. Use the 7-segment display resources of the DE2 board to display the number of keystrokes and the key value of the last key pressed. The block diagram below shows a typical partitioning of the logic.

16-Key Keypad Scanner Architecture



Note:
ROW[3:0] is a tri-stateable output

Figure 1: **Block Diagram of Keypad Scanner**

Scanner Core Logic Behavior:

- The 16-key keypad (provided) is constructed in matrix fashion having 4 row and 4 column pins. To scan the keypad, the row pins are driven low (asserted) ***one at a time***. In order to prevent potential signal contention, the row pins are driven using open drain outputs. The four column pins from the keypad are externally pulled up and are sampled periodically to determine if any keys on the selected (asserted) row are being pressed. A **single scan** is defined as the process of sequentially asserting each of the four row pins.
- A key should not be recognized as pressed unless the following conditions are true:
    1. After scanning the keyboard (all 16 keys) a minimum of 1 time, ***no keys are pressed***.
    2. One and ***only one key*** is consistently pressed for a minimum of ***4 sequential scans***.
- The scan rate of the keypad should be adjusted such that a key will not be recognized in less than 8 milliseconds.
- After a key is recognized as pressed, the state machine enters a unique state to signal the target system that a valid key is ready to be read. While in this state, the KEY_RDY signal is asserted.
- The design should include a holding register for the key location (rowcol[3:0]) that indicates the row (bits 3 and 2) and column (bits 1 and 0) of the key that was pressed. **At the conclusion of (single) scans 2, 3 and 4, it is not necessary or desirable to compare the location of a pressed key with that obtained during the first scan.** It is not possible for the key location detected for the pressed key to change within the key recognition time-frame (8 msec). Thus, the key location stored during the last (4^{th}) scan can be safely assumed to be the same value as the key location detected at during all other scans.
- When the low-true (active low) KEY_RD_ signal is driven by the target system (in this case, your test module) the KEY_DATA[3:0] is captured. KEY_DATA[3:0] is the hexadecimal value corresponding to the label on the last key pressed. It is expected that the KEY_RD_ signal will stay asserted (low) until the KEY_RDY signal deasserts.
- When the KEY_RD_ signal de-asserts (goes high), the state machine begins to look for ***no keys pressed*** (see bullet 2, statement 1).

LFSR Counter:

The purpose of the LFSR counter is to control the scan rate of the scanning. The total time from "key pressed" to "key recognized" should be no less than 8 milliseconds so as to mask any bouncing that may be taking place. **For example**, assume the following:

1) the scanner has 4 rows
2) the scanner scans the keypad 4 times before recognizing a key
3) the LFSR clock frequency is 1 MHz

Then the counter length (number of clocks before timeout) would be **(8.0 E-3 * 1.0 E6) / (4 * 4) = 500**.
Therefore, an LFSR counter having a length of 9 bits would be appropriate. **THE FORMULA VALUES STATED ABOVE ARE CONSISTANT WITH THE EXAMPLE. You will have to recalculate the counter length for your lab design based on the DE2 board 50MHz clock source.**

The LFSR counter module actually has two instances of LFSR counters. The LFSR module has one decoded output (counter time out) from each of its LFSR counters. The two decoded outputs are 1) lfsr_lto and 2) lfsr_sto. The lto signal is a "long" counter timeout that will yield an 8 msec key recognition time. The sto signal should decode a shorter, 15-state LFSR counter. The scanner_core module selects the lto or sto signal depending on whether the design is in test mode or not. Use test mode (TEST_ low) for all of your simulations. Use non-test mode (TEST_ high) for demonstrations on the DE2 board.

The LFSR counters should be asynchronously reset by the RESET_ signal and synchronously cleared by lfsr_en.

Key_cntr:

The key_cntr module contains a synchronous binary counter that is asynchronously reset by RESET_. This counter increments each time KEY_RDY asserts. The challenge is to clock this counter using the CLOCK input. *It is considered bad practice for synchronous digital designs to clock flip flops directly from non-clock signals.* Therefore, using "posedge KEY_RDY" is not permitted.

Key_xlate:
The key_xlate module translates the row/column code generated by the core logic to a 4-bit key value code that corresponds to the hexadecimal key label for each of the 16 keys. This module is comprised of purely combinatorial logic.

Sevenseg_enc:
This module encodes 4 bits of binary information into a 7-bit pattern suitable for driving one of the 7-segment LED displays contained on the DE2 board. Each of the 7-segment display segments will light when its associated FPGA pin is driven low. Refer to the DE2 board user manual for the pin assignments for the 7-segment devices.

Auto Read:
The schematic diagram for this module illustrated in Figure 2. This module is a small state machine that generates a read pulse whenever the o_key_rdy signal asserts. When the AUTO_RD_ pin is connected to the KEY_RD_ pin, KEY_DATA is assumed to be captured by the target system (e.g. your test module). The scanner acknowledges the key data "read" by de-asserting the KEY_RDY pin at the next clock.

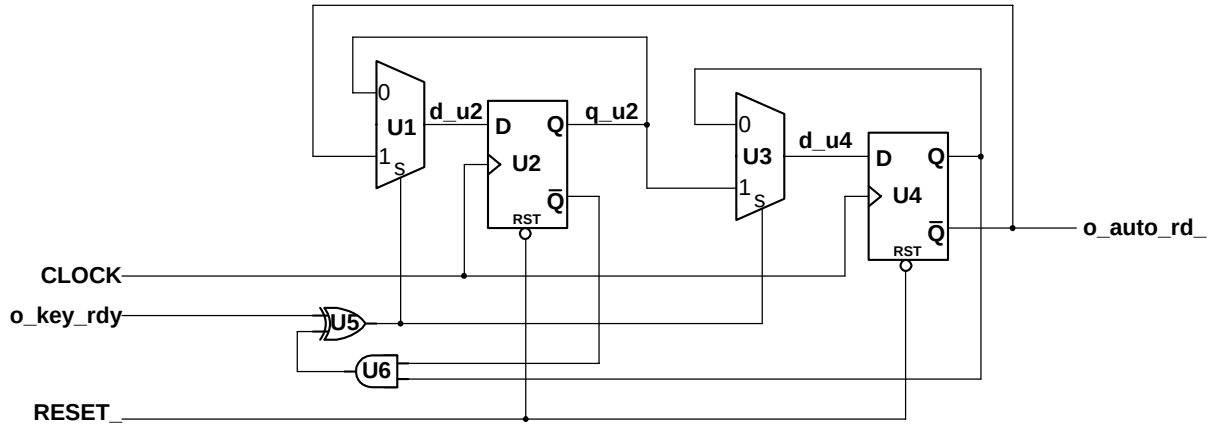


Figure 2: **Auto Read Circuit**

Scanner Outputs:

KEY_DATA[3:0]	a 4-bit bus to deliver key data to the target system. This data will be used to drive the red LEDs on the DE2 board.
ROW[3:0]	A 4-bit, one-hot zero bus used to drive the keypad rows low, one at a time. Un-driven rows should be in the "high-Z" state. These signals are driven out on the JP1 connector of the DE2 board.
KEY_RDY	a one-bit high true signa1indicating that a key has been pressed and is ready to be read by the target system. This signal will be used to drive a green LED on the DE2 board.
AUTO_RD_	The AUTO_RD_ signal is generated each time the KEY_RDY signal asserts. This pin remains asserted until the KEY_RDY signal has been sampled as de-asserted. This signal will be driven to a pin on the JP1 connector of the DE2 board.
KEY_7SEG[6:0]	These pins are used to drive the HEX0 seven-segment LED display on the DE2 board. This output indicates the value of the last key pressed.
CNT_7SEG[6:0]	These pins are used to drive the HEX1 seven-segment LED display on the DE2 board. This output indicates the number of keystrokes received (modulo 16) since the scanner has been reset.

Scanner Inputs:

CLOCK	Used to clock ALL flip-flops – no exceptions. Use a 50MHz clock source in your test bench. Use the 50MHz clock source provided on the DE2 board.
RESET_	This signal is used to initialize all flip-flops. Connect this pin to KEY0 on the DE2 board.
TEST_	Used to select a shorter LFSR count sequence so that ***ModelSim*** simulations execute faster. This input pin should be terminated high using the DE2 board switch 0 (SW0) for hardware demonstrations so that an 8 msec debounce period is realized.
COL[3:0]	This 4-bit bus is used to sample the states of the 4 columns of keys. These inputs should be pulled up in your test bench. These inputs should be configured to have weak pull-ups in your Quartus II Settings configuration. These signals will be connected to pins on the JP1 connector of the DE2 board.
KEY_RD_	a one-bit low true signal used to signal the scanner that KEY_DATA has been captured. This signal will be connected to a pin on the JP1 connector of the DE2 board.

Resources:

Project Hardware Kit – Provided:

1) Altera DE2 board with power supply
2) USB download cable
3) 16-key keypad with 4 row pins and 4 column pins
4) 40-pin ribbon cable

Project Development Toolkit provided:

1) Keypad BFM
2) LFSR Utility

Lab Execution Steps

Design Implementation	Use Verilog HDL digital design techniques to implement the design.
Verification	Create a test bench and use ***ModelSim*** to test the design.
Synthesis, place and route	Use ***Quartus II*** to synthesize the logic targeting the FPGA on DE2 board.
Pin Assignments	Use ***Quartus II*** to assign the device I/O pins compatible with DE2 infrastructure
Hardware Configuration	Configure the DE2 board for hardware test.
Programming	Download the program to the DE2 board.
Final Hardware Test	Reset the DE2 board and FPGA using the designated RESET_ button. Exercise the keypad and observe proper operation of the 7-segment LEDs.

Tasks and Deliverables:

To receive a full credit, follow these steps.

Step 1 *a)* Design the specified Keypad Scanner using Verilog HDL synthesizable RTL and an appropriate module hierarchy. Include the keypad BFM in your test bench.
b) Simulate the design using ***ModelSim*** and demonstrate its proper functioning.
c) Deliver the test bench environment including *Verilog* RTL Code and Simulation results.

Step 2 *a)* Using ***Quartus II***, synthesize the RTL level Verilog Keypad Scanner targeting the FPGA device on DE2 board.
b) Assign pins to the FPGA as according to the keypad you've been issued and recompile your design.
c) Configure the DE2 board with the programming file, add the appropriate hook-up wires (22 gauge recommended) to connect AUTO_RD_ to KEY_RD_.
d) Board demonstration to TA. Demonstration is to be done *AFTER* you have finished and submitted your other deliverables. You will have a week after the Part I due date to complete this.

Submission:

All submissions should be deposited in the dropfolder. Make sure that you include at least TWO directories One will be your ModelSim project folder while the other will contain an archive of your QUARTUS II files (.qar file).

Due Date:

This lab is due electronically (via dropfolder) by 11:59 p.m. on April 29th. You have to demonstrate to the TA by May 6th. Note that you need to submit your Lab2 before demonstrating to the TA.