EITF35 - Introduction to the Structured VLSI Design

Fall 2018

Interfacing Keyboard with FPGA Board

(FPGA Interfacing)

Teacher: Dr. Liang Liu

v.1.0.0

Abstract

This document describes the basic behavior of a PC keyboard and how to interface it with the FPGA as lab work, which is part of course EITF35 "Introduction to Structured VLSI" at EIT, LTH. The lab will result in a FPGA implementation of a keyboard controller. The designs are to be realized using VHDL. The simulation can be done using ModelSim, whereas the synthesis for FPGA's implementation can be done using Xilinx Vivado.

Contents

1 KEYBOARD

1.1 Introduction

In this lab you will create a keyboard decoder, and interface it with an FPGA platform. This will help in becoming more familiar with the VHDL code implementation of a small design and the use of FPGA to implement and test it. This lab therefore requires a FPGA board (provided in lab), FPGA programming tool (Vivado) and any hardware simulators like ModelSim or Vivado inbuilt simulator. You will use a typical HDL flow, wherein, first you write the HDL code, the run a functional HDL simulation, and latter synthesis the design and test it in the FPGA board.

1.2 Reading Keyboard Scan Codes Through the PS/2 Interface on the Board

The keyboards provided in the lab has a USB interface, however, don't worry, we won't ask you to implement a USB host, which is a very complex design. In our case fortunately the FPGA board has a USB host (PIC microcontroller) implemented already. This USB host converts the keyboard interface to old PS/2 format interface, and we will design the controller for this interface.

The old PS/2 PC keyboard transmits data in a clocked serial format consisting of a start bit, 8 data bits with least significant bit (LSB) first, an odd parity bit and a stop bit. The clock signal is only active during data transmission. The generated clock frequency is usually in the range of 10 - 30 kHz. Each bit should be read on the falling edge of the clock.

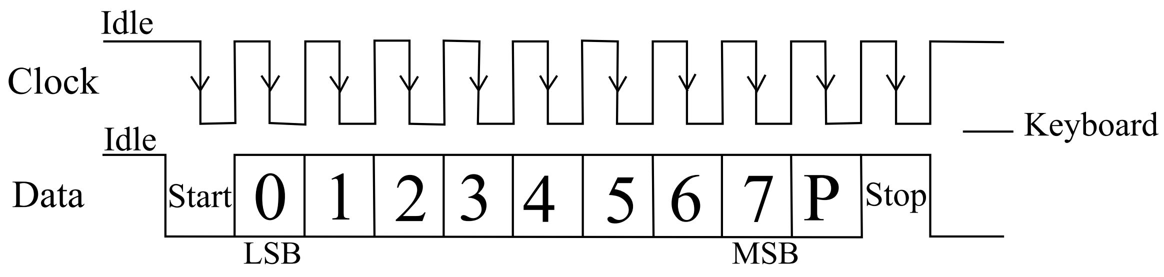

Figure 1: Key Board interface waveform

In figure 1, the waveform represents a one byte transmission from Keyboard. The keyboard may not generally change its data line on the rising edge of the clock as shown in the diagram i.e., there can be some skew in the data generation and clock edge. The data line only has to be valid on the falling edge of the clock. The LSB is always sent first.

1.3 Scan codes and commands

1.3.1 Keyboard layouts with codes

When a key is pressed, the keyboard transmits a 'make' code consisting of an 8-bit 'scan' code denoting the key pressed, when the key is released, a 'break' code is sent. The

Table 1: Port Interface Description

Name	Type	Range	Description
kb_data	in	std_logic	The scan code bits enter through this input.
kb_clk	in	std_logic	The keyboard clock signal enters through this input.
sys_clk	in	std_logic	The system clock used to clock the system.
sc	out	std_logic_vector(7 downto 0)	Drive the bargraph (pins LD0-LD7) on the Board.
num	out	std_logic_vector(6 downto 0)	Drive the seven segment display on the Board.
seg_en	out	std_logic_vector(3 downto 0)	Enable signal for Seven Segments (An0-An3) on the Board.

'break' code consists of the same 8-bit scan code preceded by a special code - 'F0'H. More information can be found in Appendix A, "PC AT keyboard ref".

1.4 Port Definitions

The inputs and outputs of the circuit is defined in Table.1, the keyboard produces keyboard clock and keyboard data. The FPGA uses a global clock generated by the oscillator on the board. The outputs from the FPGA are scan code, seven segment data and seven segments enable signals.

1.5 Procedure Definitions

Within the main body of the architecture section, following processes need to be implemented. For consistency purposes use the same names for the processes:

sync_keyboard Synchronize the external inputs to the system clock. Create an internal copy of the keyboard signals that only changes values on the positive edge of the system clock.

detect_falling_kb_clk Create a single bit output which is high during one system clock cycle when a falling edge of the synchronized kb_clk signal has been detected. This can be done by comparing the current value of the synchronized kb_clk signal with the old value (stored in the previous clock period) of the synchronized kb_clk signal. Remember that the output signal from this process should not be used as a clock, as this would create a design with multiple clock domains.

convert_scancode When edge_found is 'high' this process shifts the data bit on the kb_data input into a shift register and user right shift to reverse the order of arriving data. After 11 shifts (bits from kb_data), the lower 8 bits of the register will contain the scan code, the upper 2 bits will store the stop and parity bits, and the start bit will have been shifted through the entire register and discarded.

valid_scancode In order to extract the data from the shift register a counter (-mod 11) needs to be implemented. A valid_scancode signal needs to be generated (counter overflow) to indicate that the data has been extracted and can be used by other modules in the design.

The value in the shift register can directly be applied to the seven segment LED. Since the bargraph segments are active-high, a segment will light for every '1' bit in the shift register. If the scan code in the shift register matches the codes for the digits 0-9, then the correct LED seven segments will be activated to display the corresponding digit. If the scan code does not match one of these codes, the letter 'E' is displayed. There are four seven segments on the board and they are active low. To enable the seven segments you have to assign a low signal to one of the four control signals in order to operate a particular seven segment.

1.6 Tips and Tricks

To avoid a lot of trouble in the design phase a few hints are needed,

- Do not use multiple clock definitions, i.e., use only one statement with 'EVENT defining a clock.
- The use of 'EVENT and 'LAST_VALUE are not useful in the context of synthesis except for defining the rising or falling edge of clock.
- Try to avoid software style of coding (variable, for loops etc), and make sure you understand what hardware will be created for every block of code written.

1.7 Interface

The Xilinx synthesis tool needs to know how the FPGA is connected to the external world. A .xdc file is generated for this purpose. The file consists of the mapping of IO pins described in VHDL with physical pins in board. This will be provided along with the manual.

2 Assignment

Create a VHDL based design that accepts scan codes from a keyboard attached to the PS/2 interface of the Board. The binary pattern of the scan code is displayed on the bar-graph LEDs. You have to fulfill the following functionalities :

- If scan code for one of the '0'-'9' keys arrives, the numerals should be displayed on the correct Seven Segment display of the Board.
- The next scan codes pressed should be displayed to right of the first one and so-on.
- A 'E' (-Error) needs to be displayed if other keys are pressed.

2.1 Tasks/Requirements

2.1.1 Lab Preparations

Here are the tasks for lab preparation (also look at the lecture for more details and deadline dates)

- Go through the manual, keyboard interface, and some basics of FPGA flow.

• Understand and map the block diagram with the top level VHDL code provided. [1]

• Draw the circuits for Edge detector, and understand the syncrhonizer circuit provided in manual.

• Write the VHDL code for both edge detector and syncrhonizer on paper. And the student should be able to explain the code/hardware.

• Understand the requirements and figure out the keyboard controller.

A testbench will be provided that can be used by students to check if the keyboard decoder works during simulation.

2.1.2 Lab Requirements

Here are the requirements to pass the laboratory

•Implement the design using the provided VHDL files, and verify using the testbench in modelsim simulator or vivado simulator.

•Synthesize the design and demonstrate the required functionality on the FPGA board.

• Make sure to verify the design on FPGA for different scenarios before approval.

Hints :

- Check the reset condition, all display should be switched off at reset.
- Pressing "123456" one by one, the display should look the following:

Seven Segement Display

DISPLAY OFF				press (sw0) reset
			1	press 1
		1	2	press 2
	1	2	3	press 3
1	2	3	4	press 4
2	3	4	5	press 5
3	4	5	6	press 6

3 APPENDIX

3.1 Description of a PC Keyboard

The original keyboard design had a single chip microprocessor, but now a customised controller chip is used. This keyboard controller chip takes care of all keyboard matrix scanning, key de-bouncing and communications with the computer, and has an internal buffer if the keystroke data cannot be sent immediately. The PC motherboard decodes the data received from the keyboard via the PS/2 port using interrupt IRQ1. The one thing

[1]The VHDL file along with testbench has been provided to ease the work load for many first time VHDL/FPGA users. The students should basically fill-up the architecture for each module.

that these keyboards do not generate is ASCII values. With a typical AT keyboard having more than 101 keys, a single byte could not store codes for all the individual keys, plus these keys along with shift, control, or alt, etc. Also for some functions there is no ASCII equivalent, for example ‘page up’, ‘page down’, ‘insert’, ‘home’, etc.

Once the keyboard controller finds that a key has been pressed or released, it will send this keystroke information, known as scan codes, to the PIC micro-controller. There are two different types of scan codes - make codes and break codes. The communications protocol is bi-directional, but here we only discuss the keyboard to host part.

3.1.1 make code

A make code is sent whenever a key is pressed or held down. Each key, including ‘shift’, ‘control’ and ‘alt’, sends a specific code when pressed. Cursor control keys, ‘delete’, ‘page up’, ‘page down’, ‘ins’, ‘home’ and ‘end’, send extended make codes. The make code is preceded by ‘E0’h to indicate an extended code. The only exception is the ‘pause’ key that starts with a unique ‘El’h byte.

3.1.2 break code

A break code is sent when a key is released. The break code is the make code preceded by ‘F0’h byte. For extended keys the break code has an ‘E0’h preceding the ‘F0’h and make code value. The only exception is the ‘pause’ key as it does not have a break code and does not auto-repeat when held down.

3.1.3 key code

Every key is assigned its own unique code so that the host computer processing the information from the keyboard can determine exactly what happened to which key simply by looking at the scan codes received. There is no direct relationship between the scan code generated by a particular key and the character printed on the key top.

The set of make and break codes for each key comprises a scan code set. There are three standard scan code sets -numbered 1, 2, and 3 - stored within the keyboard controller. Scan code set 1 is retained for compatibility for older IBM XT computers. Scan set 3 is very similar to the set 2 but the extended codes are different. Scan code set 2 is the default for all AT keyboards and all scan codes discussed here are from this set.

3.1.4 scan code

If, for example, you press ‘shift’ and ‘A’ then both keys will generate their own scan codes, the ‘A’ scan code value is not changed if a shift or control key is also pressed. Pressing the letter ‘A’ generates ‘lC’h make code and when released the break code is ‘F0’h, ‘lC’h. Pressing ‘shift’ and ‘A’ keys will generate the following scan codes:

•The make code for the ‘shift’ key is sent ‘12’h.

•The make code for the ‘A’ key is sent ‘lC’h.

•The break code for the ‘A’ key is sent ‘F0’h, ‘lC’h.

•The break code for the ‘shift’ key is sent ‘F0’h,‘12’h.

If the right shift was pressed then the make code is ‘59’h and break code is ‘F0’h, ‘59’h. By analyzing these scan codes the PC software can determine which key was pressed. By looking at the shift keystroke the software can distinguish between upper and lower case.

3.1.5 Keyboard serial data

The AT keyboard transmission protocol is a serial format, with one line providing the data and the other line providing the clock. The data length is 11 bits with one start bit (logic 0), 8 data bits (lsb first), odd parity bit and a stop bit (logic 1). The clock rate is approximately 10 to 30 kHz and varies from keyboard to keyboard.

3.2 Scan code tables

The scan codes generated, when a keyboard key is pressed and when its released is given by table 2.

Table 2: Keyboard alpha numeric scan codes:

key	**make**	**break**
A	‘lC’h	‘F0’h ‘1C’h
B	‘32’h	‘F0’h ‘32’h
C	‘21’h	‘F0’h ‘21’h
D	‘23’h	‘F0’h ‘23’h
E	‘24’h	‘F0’h ‘24’h
F	‘2B’h	‘F0’h ‘2B’h
G	‘34’h	‘F0’h ‘34’h
H	‘33’h	‘F0’h ‘33’h
I	‘43’h	‘F0’h ‘43’h
J	‘3B’h	‘F0’h ‘3B’h
K	‘42’h	‘F0’h ‘42’h
L	‘4B’h	‘F0’h ‘4b’h
M	‘3A’h	‘F0’h ‘3A’h
N	‘31’h	‘F0’h ‘31’h
0	‘44’h	‘F0’h ‘44’h
P	‘4D’h	‘F0’h ‘4D’h
Q	‘15’h	‘F0’h ‘15’h
R	‘2D’h	‘F0’h ‘20’h
S	‘IB’h	‘F0’h ‘1B’h
T	‘2C’h	‘F0’h ‘2C’h
U	‘3C’h	‘F0’h ‘3C’h
V	‘2A’h	‘F0’h ‘2A’h
W	‘ID’h	‘F0’h ‘1D’h
X	‘22’h	‘F0’h ‘22’h
Y	‘35’h	‘F0’h ‘35’h
Z	‘lA’h	‘F0’h ‘1A’h
1	‘16’h	‘F0’h ‘16’h
2	‘lE’h	‘F0’h ‘1E’h
3	‘26’h	‘F0’h ‘26’h
4	‘25’h	‘F0’h ‘25’h
5	‘2E’h	‘F0’h ‘2E’h
6	‘36’h	‘F0’h ‘36’h
7	‘3D’h	‘F0’h ‘3D’h
8	‘3E’h	‘F0’h ‘3E’h
9	‘46’h	‘F0’h ‘46’h
0	‘45’h	‘F0’h ‘45’h

3.3 Synchronizer

A synchronizer circuit is required when two devices/modules are running with different clocks. In this assignment the keyboard needs to be syncrhonized with the FPGA system clock. Luckily the keyboard clock is quite slow (in KHz) compared to FPGA system clock (in MHz). Hence, a simple back-to-back flip-flop based synchronizer is sufficient, as shown in Figure 2.

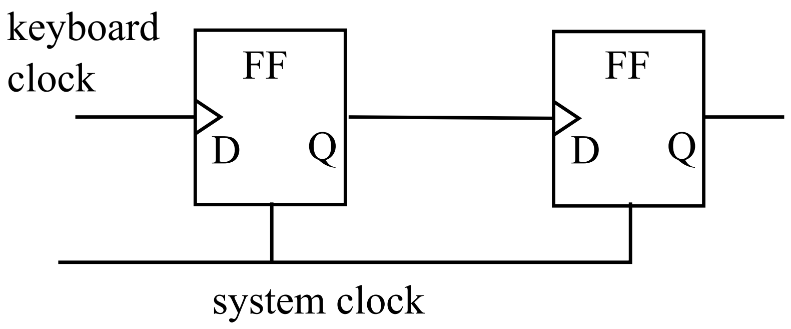

Figure 2: Synchronizer Circuit

3.4 7-Segment Driver

The FPGA board in the lab provides four common-anode type 7-segment displays (AN3-An0). The segments of each display are called a, b, up to g as shown in Figure 3. In order to reduce the number of connections on the board cathode pins of different displays are connected together to form seven common terminals between displays for every segment a, b up to g. To illuminate a segment of a particular display, e.g. segment e of the fourth display, one needs to apply a “0” to Anode3 (in order to select the third display) and a “0” to cathode node e (to light segment e).

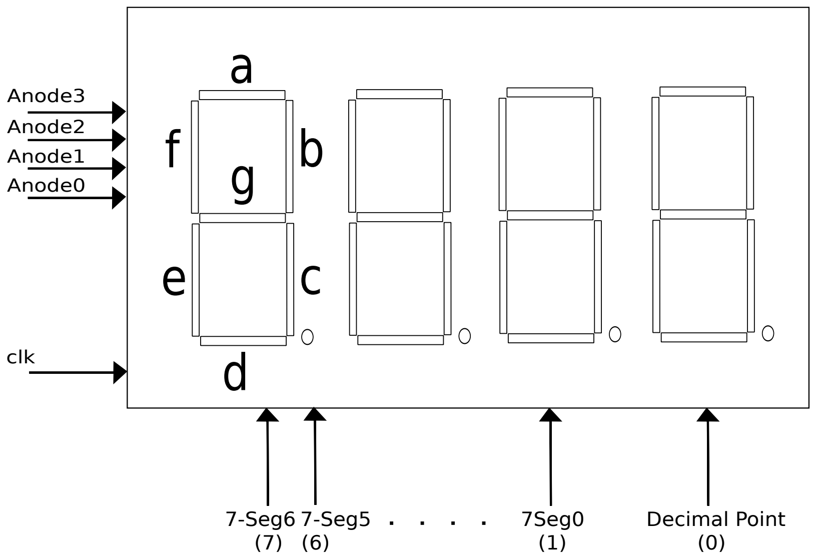

Figure 3: 7-segment display

Due to a shared bus between the segments, 7-segment modules are not lit continuously. They are rather sequentially enabled and disabled with a frequency such that the human eye will have the illusion that the display is continuous. Thus, if two or more digits are to be displayed, time multiplexing with a reasonable cycling interval between the digits of the 7-segment display must be employed.

3.5 Files and Description

For the Lab2 assignment the block diagram and architecture details are provided to the students. Most of the VHDL files provided are self explanatory and has in-line comments to guide students. The block diagram for the keyboard interface is shown in 4. Students are free to modify and instantiate new modules if required. The reset in the provided sw0 button and is active high.

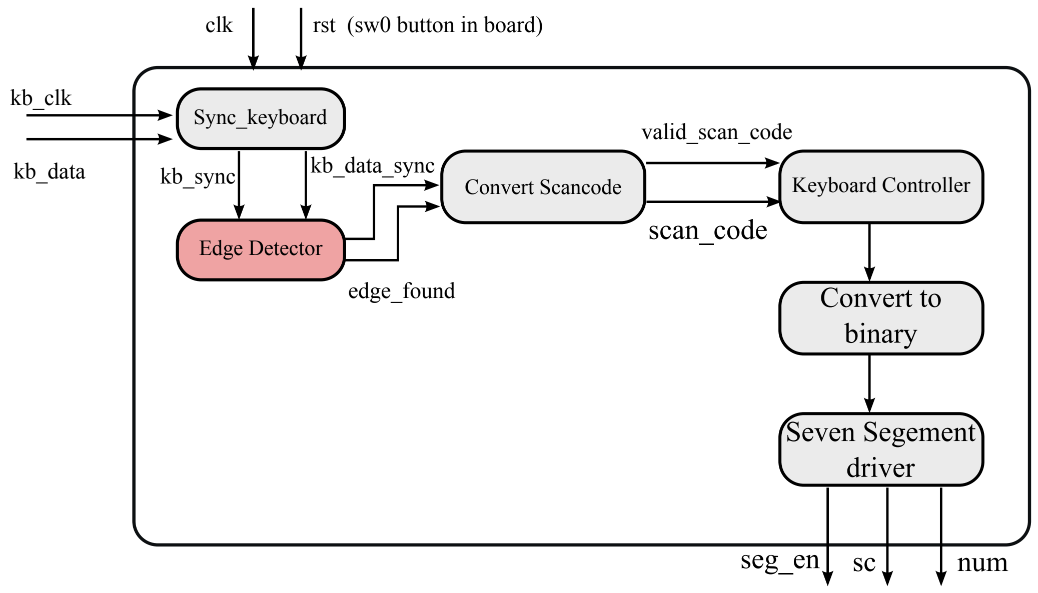

Figure 4: Key Board Block Diagram

Each of the sub-module has to be implemented in the corresponding VHDL file. The top level integration is done in keyboard top.vhd file.

The module Sync_keyboard basically should synchronize the keyboard clock and data (slower rate in few KHz) with the FPGA system clock (in MHz). A simple synchronizer would be to use 2-flip flops and sample the slower clock using faster clock.

After synchronization a falling edge_detection needs to be performed. This can be implemented by delaying the input *kb_clk_sync* by 1 system clock cycle (using a Flip-Flop) and then compare with current value.

NOTE: MAKE SURE 'EVENT IS USED ONLY FOR SYSTEM CLK !!

The module convert scancode is simple shift register which converts the serial data into parallel. Probably requires a counter to indicate when the shifted value is valid.

Keyboard controller for Lab2 requires a state-machine which will handle the different sequences of scancode.

Converting the *scan_code* to binary and the seven segment driver can be implemented as a combinational logic with a LUT (Look-Up-Table).

File Name	Function
keyboard_top.vhd	Top level integration file
sync_keyboard.vhd	Synchronize keyboard data
edge_detector.vhd	Edge detection circuit
convert_scancode.vhd	Convert serial data to parallel
keybaord_ctrl.vhd	Keyboard controller
convert_to_binary.vhd	Convert scan code to binary
binary_to_sg.vhd	Seven segment driver
keyboard_top.xdc	Pin mapping to FPGA board.
tb_pkg.vhd	Required for some functions in testbench
tb_keyboard.vhd	test bench to drive stimulus
input.txt	Keys to be sent to design via testbench.

3.6 Further reading

3.6.1 Web references

Some external links for MOdelsim and Vivado tutorials apart from class lecture. Older version links but not much change in the flow.
`http://www.xilinx.com/training/vivado/`
`https://www.youtube.com/watch?v=NCyENtIxqQ8`

Interfacing the AT keyboard
`http://www.beyondlogic.org/keyboard/keybrd.htm`