Lab 4 Advanced: Keyboard & Audio

Submission Due Dates:

Demo:	2024/10/29 17:20
Source Code:	2024/10/29 18:30
Report:	2024/11/03 23:59

Objective

1. Getting familiar with the modeling of finite state machines in Verilog.
2. Getting familiar with the FPGA design flow and the keyboard control of the demo board.
3. Getting familiar with the control of the audio peripheral Pmod I2S2.

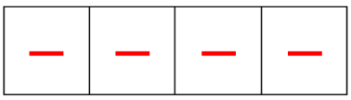
Action Items

1. lab4_1 (20%)

A. Description

Design a 7-segment controller with a keyboard in Verilog. The controller will shift the displayed number to the right each time a numeric key (0–9) is pressed, as shown in the following:

a. Initial



b. Press 5

c. Press 6

d. Press 7, and then 8

8	7	6	5

e. **Press 9**

9	8	7	6

f. **Press the rst button**

—	—	—	—

B. I/O signal specification

a. **clk**: the clock signal with the frequency of 100MHz (connected to pin W5)

b. **rst**: the asynchronous positive reset (connected to BTNC).

c. **digit[3:0]**: signals to enable one of the 7-segment digits.

d. **display[6:0]**: signals to control the digits on the 7-segment display.

e. **PS2_DATA, PS2_CLK**: connect to the keyboard IP.

C. Rules & Hints

a. The 7-segment should be **"----"** after pressing the rst.

b. **When a key is pressed and held, no other key should become active.** For example, if you press and hold the '2' key, pressing '3' will have no effect until '2' is released. If '3' is still held down when '2' is released, it is acceptable for '3' to either become active or remain inactive..

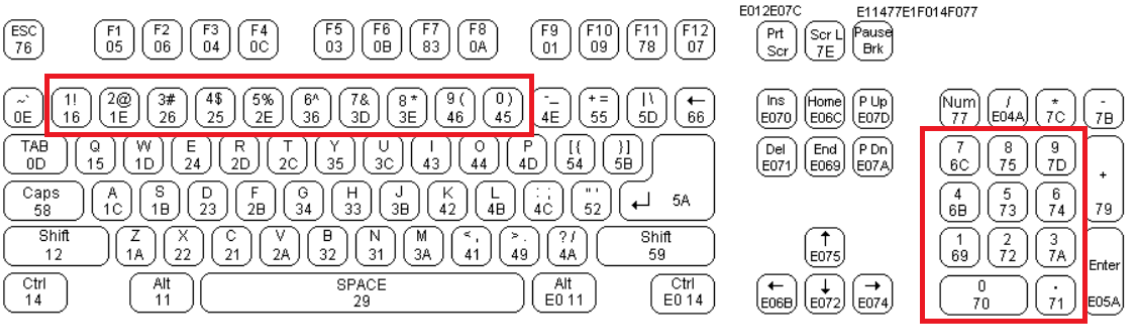
c. **When a key is pressed and held, the corresponding number should be processed only once.** For example, if you press and hold the '2' key while the 7-segment display is "----", it should update to "2---" and not repeatedly to "2222."

d. If you are unfamiliar with the usage of the `KeyboardDecoder` module, you can start by running and modifying the provided `SampleDisplay.v`. Be sure to include the keyboard control IP if required.

e. PS2_CLK and PS2_DATA are the inout signals, that is, bidirectional signals. Don't change them to either input or output.

f. You don't need to demonstrate lab4_1. This part will be verified together with lab4_2.

g. We will test both sets of numeric keys in the demo.



h. Use the following template for your design (you should determine the proper data type of each IO):

```
module lab4_1 (
  input wire clk,
  input wire rst,
  inout wire PS2_DATA,
  inout wire PS2_CLK,
  output reg [3:0] digit,
  output reg [6:0] display
);

  /* Note that output ports can be either reg or wire.
   * It depends on how you design your module. */
  // add you design here
endmodule
```

i. Demo video: https://drive.google.com/file/d/1MGD6TEMlBCd_EmuTTJ7yQJM_iaxxLqzl/view?usp=drive_link

2. lab4_2 (40%)

A. Description

Design a “Whack-A-Mole” game using a keyboard in Verilog. In this game, players use a mallet to hit toy moles that appear at random, controlled by a Linear Feedback Shift Register (LFSR), and disappear back into their holes after one second. The game begins when the **start** button is pressed, with players hitting the moles via the keyboard and the score displayed on the 7-segment display.

I/O signal specification

a. **clk**: the clock signal with the frequency of 100MHz (connected to pin W5)

b. **rst**: the asynchronous positive reset (connected to BTNC).

c. **start**: the button to start the game (connected to BTNR).

d. **LED[15:0]**: represent the holes and flash in final state.

e. **digit[3:0]**: signals to enable one of the 7-segment digits.

f. **display[6:0]**: signals to control the digits on the 7-segment display.

g. **PS2_DATA, PS2_CLK**: connect to a keyboard IP.

States of the game and their detail descriptions:

a. Initial state

- The FSM will be reset to the Initial state (i.e., after pressing the **rst** button).
- The FSM will transit to the Set state when pressing the **start** button.
- The 7-segment display shows “----” in this state.
- All the LEDs are off.

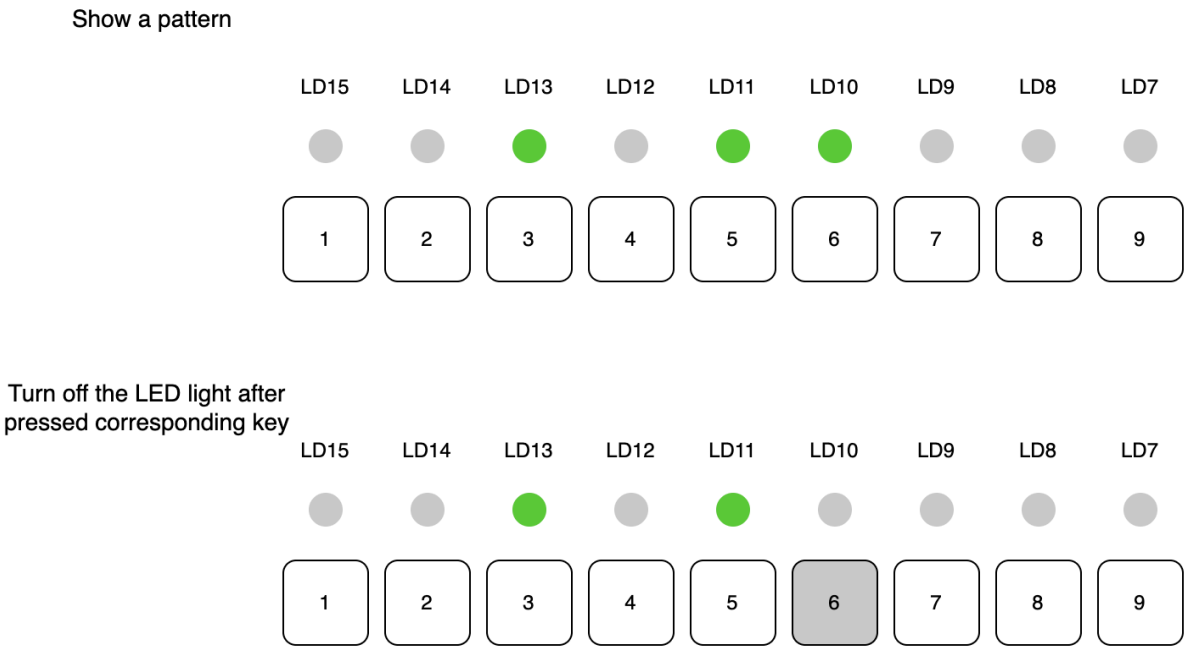
b. Set state

- In this state, set the time and target scores. Use the space key to toggle between the two settings. The time range is 1-30, and the target score range is 1-10. Testing will be limited to these ranges, so handling out-of-range values is not required.
- You can set the time or target score with the number keys (0-9).
- The 7-segment display is divided into two parts: the leftmost two digits represent the time, and the rightmost two digits represent the target score. With the default settings, the 7-segment display initially shows “3010,” with “30” as the default time and “10” as the default target score. You will begin by setting the time. When a number key is pressed, the leftmost two digits of the 7-segment display should shift left. The same behavior applies when setting the target score.
- LEDs will indicate the current setting mode:
    - Time setting: LED[15:8] **on** — LED[7:0] **off**
    - Target score setting: LED[15:8] **off** — LED[7:0] **on**
- Switch to the Game state when pressing the **start** button.

c. Game state

- The 7-segment display is divided into two parts: the leftmost two digits represent the time, counting down from 30 seconds with the default setting, and the rightmost two digits record the score. Players earn one point for each successful hit on a toy mole. For example, the display will be **“3000”** initially and change to **“2900”** after one second, then to **“2901”** after hitting a mole immediately. The game ends and transitions to the Final state either when the 30 seconds expire or the player reaches 10 points with the default target score setting.
- LEDs (LD15-LD7) represent the moles with their positions updated randomly every second using a Linear Feedback Shift Register (LFSR).
- Game control by the keyboard:

  The number keys (1-9) represent whacks and correspond to the LD15 to LD7 holes. Players can hit a mole by pressing the matching key, and only one key is active at a time. An example is provided below.



d. Final state

If the player reaches the target score of 10, they win. Otherwise, they lose.

- The 7-segment display shows the game result.

  I.e., WIN: “- W I N”

  or LOSE: “L O S E”.

- All the LEDs will flash three times. One flash consists of the LEDs staying on for 0.5 seconds and off for 0.5 seconds. After three flashes (a total of 3 seconds), the state machine will go to the Initial state.

B. Rules & Hints

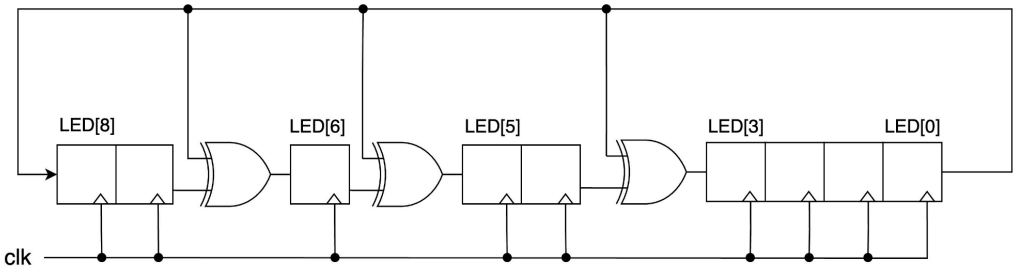
a. You must design at least one **finite state machine (FSM)** with a minimum of four states: **Initial**, **Set**, **Game**, and **Final**. Additional states or multiple FSMs are allowed, but be sure to explain your design thoroughly in the report.

b. When a key is pressed and held, no other key should become active.

c. When a key is pressed and held, the corresponding number should be processed

only once.

d. Use the LFSR discussed in class to generate pseudo-random numbers. You can either use the LFSR outputs directly as mole patterns or use the random numbers to look up predefined mole patterns. Ensure that the number of patterns exceeds 10. Below is an LFSR example:



e. PS2_CLK and PS2_DATA are the inout signals, that is, bidirectional signals. Don't change them to either input or output.

f. Use the following template for your design (you should determine the proper data type of each IO):

```
module lab4_2 (
  input wire clk,
  input wire rst,
  input wire start,
  inout wire PS2_DATA,
  inout wire PS2_CLK,
  output reg [15:0] LED,
  output reg [3:0] digit,
  output reg [6:0] display
);

  /* Note that output ports can be either reg or wire.
   * It depends on how you design your module. */
  // add you design here
endmodule
```

g. Demo video: https://drive.google.com/file/d/15vKjyErTO2J6F5N7sl2vnMamE4KWs6jK/view?usp=drive_link

3. lab4_3 Piano (40%)

A. Description

Design an electronic piano using a computer keyboard as the instrument's input.

a. Upon reset, the volume level defaults to 3, and the note starts in the fourth octave (C4-B4).

b. Volume levels range from 1 to 5.

c. The octave ranges spans from 3 to 5.

d. For the note control and volume control, please refer to points C and D.

B. I/O Signal Specification

BtnC	Reset (Asynchronous positive reset)
BtnU	Volume Up
BtnD	Volume Down
BtnL	Octave Down
BtnR	Octave Up
LED 0 ~ 4	Volume Indicator
Pmod JB 1~6	Pmod I2S
7-Segment PS2_DATA, PS2_CLK	Displaying Note Connect to a keyboard IP

C. Note Control

a. The piano plays a note when the corresponding key is pressed and stops when the key is released.

b. A note consists of two parts: pitch and octave.

- E.g., the notation C4 indicates that its pitch is C, and its octave is 4.

c. The octave is controlled by the **BtnL** and **BtnR** buttons.

d. Pressing the **BtnR** raises the note by one octave.

- Raising a note by one octave doubles its frequency.
- E.g., A4 (440 Hz) -> A5 (880 Hz)

e. Pressing **BtnL** lowers the note by one octave.

f. Pressing **BtnR** at octave level 5 or **BtnL** at octave level 3 has no effect.

g. The pitch is controlled using the computer keyboard.

h. Key-to-pitch mapping:

Key	a	s	d	f	g	h	j
Pitch	C (Do)	D (Re)	E (Mi)	F (Fa)	G (Sol)	A (La)	B (Ti)

i. A sound is produced when a valid key is pressed.

j. When a key is pressed and held, no other key should become active.

D. Volume Control

a. The piano supports five distinct volume levels, controlled by **Volume Up** (**BtnU**) and **Volume Down** (**BtnD**).

- Define your own volume levels, ensuring that each level is clearly distinguishable from the others.
- At the lowest volume level, the sound should remain audible.

b. Pressing **Volume Up** at level 5 or **Volume Down** at level 1 has no effect.

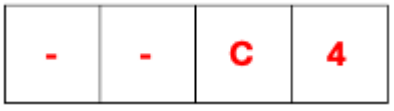
c. LEDs 0-4 indicate the current volume level:

Volume Level	LED 4	LED 3	LED 2	LED 1	LED 0
Level 1	○	○	○	○	●
Level 2	○	○	○	●	●
Level 3	○	○	●	●	●
Level 4	○	●	●	●	●
Level 5 (the loudest)	●	●	●	●	●

E. The 7-segment Display

a. Display the note currently being played.

b. For example, after a reset, pressing the key "a" will show the corresponding note on the display.



c. For more examples, please refer to the demo video.

F. Rules & Hints

a. Complete the required modules in the provided template directory.

b. The lab4_3_template directory also includes some example modules discussed in class, which may be helpful for your final project too.

c. Use the **Note-to-Frequency Map** as a reference for mapping notes to their

corresponding frequencies:

Note	C4	D4	E4	F4	G4	A4	B4
Frequency	262	294	330	350	392	440	494

d. Demo video: https://drive.google.com/file/d/1ujGkxKDQ3o-WQfi7Dvqtl-cNcmeaW1I3/view?usp=sharing

Questions and Discussion

Please answer the following questions in your report.

A. In lab4_1, how did you prevent continuous detection of a positive signal when a key is pressed and held down?

B. In lab4_3, how did you control the volume? Please explain the underlying principle.

Guidelines for the report

Refer to the guidelines in the report template (or in the previous lab assignments).

Grading policy (subject to change): Part (A): **35%**; Part (B): **50%**; Part (C): **10%**; (D): **5%**

Attention

- ✔ DO NOT include KeyboardDecoder, onepulse, debounce , keyboard_decoder and speaker_control into the .v files.
- ✔ You must hand in the file named **lab4_1.v**, **lab4_2.v, and lab4_3.v**. DO NOT hand in any compressed ZIP file, which will be considered an incorrect format.
- ✔ If you create several modules for your design, merge them all into one Verilog file.
- ✔ You should also hand in your report as **lab4_report_StudentID.pdf** (i.e., lab4_report_111456789.pdf).
- ✔ You should be able to answer questions about this lab from the TA during the demo.

- ✔ You need to prepare the bitstream files before the lab demo to make the demo process smooth.
- ✔ Feel free to ask questions about the specification on the EECLASS forum.