Intro to Digital Logic, Lab 8

Final Project

Lab Objectives

It's time to prove yourself with the DE1 board and your digital design skills!

Note that this lab is MUCH more complex than previous labs, and will take a LOT of time. Read the entire lab carefully, and START EARLY. You will need the time!

We cannot stress enough the importance of good planning and proper testing as you go!

Project Ground Rules

You will design a significant project on the DE1 board and are given a fair amount of freedom to do so, particularly if you decide to propose your own project. You may decide to make use of the following as you see fit:

- **Verilog `generate` statements:** See the "*Verilog Tutorial*" on the course website.
- **Prototyping board:** The labeled pins (*Figure 1*) are connected to the FPGA and usable. You can access these pins using a bus connection of "`inout [35:0] GPIO_0`" (inout for input/output) to your top-level module (similar to `KEY`, `SW`, `LEDR` in previous labs). `GPIO_0[0]` is the leftmost connection (`AC18`), and `GPIO_0[35]` is the rightmost connection (`AJ21`).
- **Breadboard** (white)**:** All of your digital logic should be done *inside* of the FPGA (Verilog), but the breadboard can be used as a place to make electrical connections to additional input/output (I/O) devices. Recall that there are wires and wire cutters available in lab for you to use.
- **16×16 Bicolor LED Expansion Board:** You can check one out from the ECE store (ECE 147). See the "*LED Board*" tutorial and files links on the course website for usage and getting started.
- **Multiple "clock speeds":** Your *entire* circuit should use a single clock – either `CLOCK_50` or one of the outputs from `clock_divider` circuit. If you need multiple clock trigger speeds, first set the clock to the fastest speed that you need and then use a counter to generate a slower `Enable` signal. All slower elements/modules will still use the faster clock, but will only change state when the slower counter signal occurs.
- **Additional input/output devices:** You can use other I/O ports on your FPGA or other devices, but you are responsible for procuring these yourself. You should run your idea by the staff first to make sure your proposed device doesn't handle too much of the digital logic design for you.

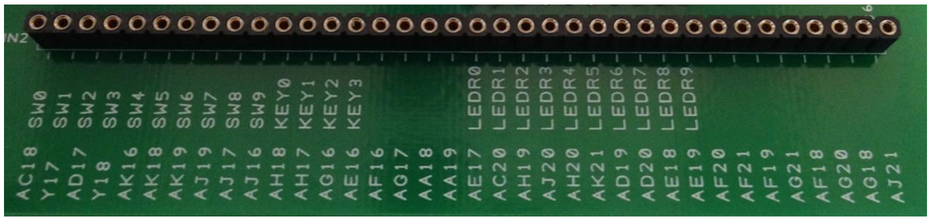
IN2
SW0 SW1 SW2 SW3 SW4 SW5 SW6 SW7 SW8 SW9 KEY0 KEY1 KEY2 KEY3
LEDR0 LEDR1 LEDR2 LEDR3 LEDR4 LEDR5 LEDR6 LEDR7 LEDR8 LEDR9
AC18 Y17 AD17 Y18 AK16 AK18 AK19 AJ19 AJ17 AJ16 AH18 AH17 AG16 AE16 AF16 AG17 AA18 AA19 AE17 AC20 AH19 AJ20 AH20 AK21 AD19 AD20 AE18 AE19 AF20 AF21 AF19 AG21 AF18 AG20 AG18 AJ21

Figure 1: *Close-up image of the general-purpose input/output (GPIO) pin connections on the prototyping board.*

Project Grading

Check-In: 10 points for demonstrating sufficient planning and implementation work one week into the project. This will be done during your usual demo slot. *Requirements:* (1) a labeled top-level block diagram of your proposed implementation and (2) Verilog implementation of one new module and testbench.

Standard: 100-160 points for correctness, style, and testing. The base project value can be found in the project descriptions. **In terms of your course grade, 150 points will be considered 100% for the project.**

Bonus: Up to 20 points for adding cool or interesting features. For each of the projects, there are lots of ways to make them more useful, efficient, or fun! However, you will get *much* more credit for a simple, working design than an awesome design that doesn't work. **The best plan is to get the basic system working and then add frills if/when you have extra time.**

Early Finish: Up to 10 points for finishing your project ahead of time. Encouragement to start early! Your project is due by 11:59 pm of the last day of class (June 3, 2022). Finishing and submitting ahead of the deadline will earn you 2 extra points for each *school* day you are early, with a maximum of 10 points (*e.g.*, 10 points for May 27, 8 points for May 30).

You may schedule project demo times outside of the normal lab hours with the TAs as needed.

Project Demonstration/Turn-In Requirements

Project Report (*before* the deadline, submit as PDF on Gradescope)

- A User's Manual for your project, which must include the following information *in your own words*:
    1) How does someone use/operate/play your design?
    2) A block diagram of your entire system with a brief description of the interconnections.
    3) A brief description of each module you created and its purpose. For modules that include finite state machines, include a state diagram for each one. Include ModelSim simulation results that reasonably prove each module's correct behavior.
    4) Briefly describe how you went about testing your system.
- How many hours (estimated) it took to complete your project in total, including reading, planning, designing, coding, debugging, and testing.
- As *separate* files, upload the Verilog code for all design files, including testbenches.
- As a *separate* file, upload your DE1_SoC.sof bitfile for your final design.

In-Person Demo (*during* your demo slot or scheduled separately with TAs)

- Demonstrate your complete system working on the DE1-SoC board, along with any additional features you added.
- Be prepared to answer questions on both the theoretical and practical parts of the lab.

Lab 8 Rubric

Grading Criteria	Points
Q1: Instructions for the user	5 pts
Q2: System block diagram	5 pts
▪ Explanation of modules and interconnections	4 pts
Q3: Rundown of modules (split points amongst all used)	
▪ Descriptions of modules	9 pts
▪ State diagrams and notes (where applicable)	6 pts
▪ ModelSim screenshots and explanations	9 pts
Q4: Testing overview	5 pts
Time spent	2 pts
Verilog code and .sof bitfile uploaded	5 pts
BONUS for additional features	(20 pts)
BONUS for Early Finish	(10 pts)
LAB DEMO	50-110 pts
	100-160 pts

Possible Projects

Listed below are a number of different possible designs. You can pick any of that these you prefer, including the "Venture Capitalist" project that lets you make up your own project. These projects vary a bit in difficulty and the project values generally reflect this, but your experience may vary.

Project 1: Frogger [150 points]

The urban horticulture program on campus thinks we can eliminate roadkill incidents by training the local wildlife to avoid cars. Your job is to develop a high-tech traffic simulator so they can learn how to safely cross the road (see https://en.wikipedia.org/wiki/Frogger).

Basic Requirements:

- A board of at least 4×8 of LEDs with easily recognizable ends of the road (where cars start/end).
- One LED color representing cars and another LED color representing your frog.
- Cars moving in one direction in some basic pattern.
- User has left, right, back, forward buttons to move the frog.
- If the frog ends up in the same square as a car, they become roadkill soufflé (squish!).
- If the user gets the frog from one side of the board to the other, they win and play again.
- The frog should be able to move faster than the cars.

Project 2: Conway’s Game of Life [135 points]

I’m tired of people saying we should get a life – we’ve got Conway’s Game of Life (http://en.wikipedia.org/wiki/Conway's_Game_of_Life)! This is a simple computation that can give rise to very interesting behaviors over time.

Basic Requirements:

- A board of at least 8×8 LEDs with easily recognizable edges.
- User must be able to input a new starting pattern then “start life.”
- Once you reach the end of your board, your game should wrap around and continue.

Project 3: Dancing with your Thumbs [150 points]

Implement a simplified version of DDR (https://en.wikipedia.org/wiki/Dance_Dance_Revolution).

Basic Requirements:

- One LED color to represent moving “arrows” and another LED color to represent the “hit zone.”
- Your machine will have four vertical tracks of lights of length eight or more, each with a corresponding button for user input.
- The system will randomly turn on a light at the “top” of a track and move it quickly towards the “bottom.” The 2nd light from the bottom of the track is part of the “hit zone.”
- The user’s goal is to press the corresponding button for the track when the light is in the hit zone. Scoring: in hit zone (+2), off by one (+1), any other position (–2), off end of track (–2)
- Your system should track and display the user score over time.

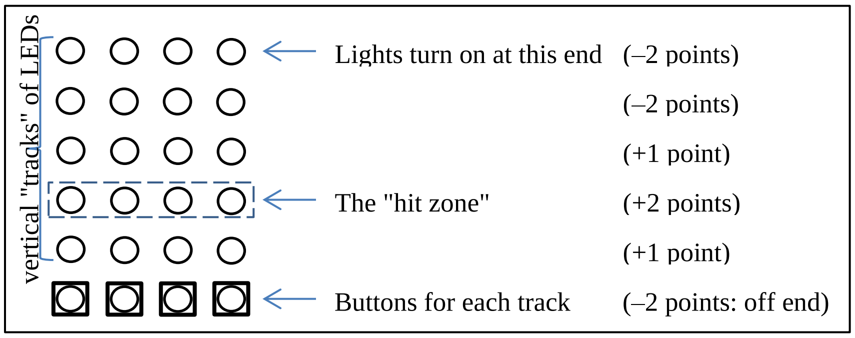

Figure 2: *Basic DDR implementation. Points are awarded if the user presses the track button while the light is in that row. Note that your track length must be at least 8 long, not the 6 shown here.*

Project 4: Flappy Bird [155 points]

Flappy Bird (https://en.wikipedia.org/wiki/Flappy_Bird) has our hero the Caped Cardinal (a red dot) flapping through a maze of “pipes” (green vertical lines).

Basic Requirements:

- The hero will “move forward” through a maze of randomly-generated pipes at a constant speed.
- Single button input: The red dot goes up when the button is pressed and down when the button is released.
- The user’s goal is to avoid the pipes for as long as possible by flying through the gaps.
- Your system should track and display the user score. Be prepared to display a score up to at least 999.

Project 5: Elevator [100 points]

The CSE building freight elevator is down again! The faculty and staff have decided that a student should design a system that will work more reliably than the current one. Your task is to implement the floor selection system from within the elevator (*i.e.*, passenger enters elevator and selects destination floor) – *you do not have to worry about "calling" the elevator*.

Basic Requirements:

- Take inputs that cover floors 1 through 4 of the building (see *Figure 3*).
- The elevator cannot teleport! When traveling between two floors, the elevator must go through every intermediate floor.
- The freight elevator has both East and West doors. The doors should only open when we are stopped. Doors should not open if there isn't an exit on that side for a particular floor. It is okay to open both doors simultaneously, if both exits are available.
- When stopped at a floor, the elevator will not move again until the "Close" button is pressed. Your system should be able to accept multiple destinations while stopped.
- The next floor to stop at is first determined by the current traveling direction of the elevator (*e.g.*, two passengers enter on floor 2: P1 going to floor 1 and P2 going to floor 3. If the elevator was traveling downward, then the elevator should go to floor 1 before floor 3).
- Your system should have indicators of elevator direction (Up/Down), open doors (West/East), and current floor. You may choose to use floor indicator lights or 7-segment displays.

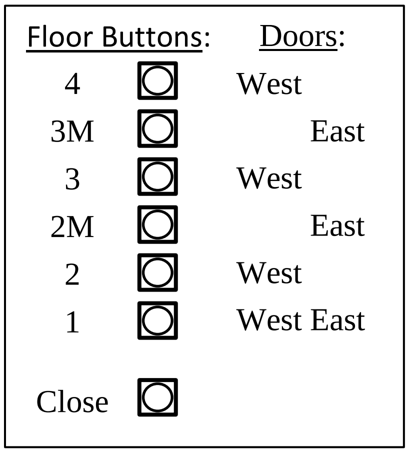

Figure 3: *Elevator selection buttons and doors that should open for each floor.*

Project 6: Connect Four [130 points]

Implement a game of Connect 4 (https://en.wikipedia.org/wiki/Connect_Four).

Basic Requirements:

- An 8×6 game board with easily recognizable borders.
- Player pieces will be shown using different LED colors.
- User input is most easily done using 8 switches to indicate which column to play the next piece in. Think about how to avoid playing multiple pieces on a single switch flip.
- There must be a turn indicator (which player's turn it currently is)
- At end game (player victory or draw), there must be an outcome indicator (*e.g.*, 7-seg message).
- A reset button or switch will restart the game.

Project 7: Tic-Tac-Toe [135 points]

Implement a game of Tic-Tac-Toe (https://en.wikipedia.org/wiki/Tic-tac-toe).

Basic Requirements:

- A board of at least 11×11 LEDs, making sure to show gridlines.
- Two clearly distinguishable player symbols (*e.g.*, X's and O's).
- User input needs to happen via "next" and "select" buttons, NOT switches. The current space under consideration should be indicated on the board. Pressing "next" will move the indicator to the next open space (including wrap-around). Pressing "select" will place the current player's symbol in the selected space and start the next player's turn.
- At end game (player victory or draw), there must be an outcome indicator (*e.g.*, 7-seg message).
- A reset button or switch will restart the game.

Project 8: Snakes & Apples [160 points]

The folks at WSU want to test out Washington State's new crop of Cosmic Crisp apples by feeding 'em to snakes! Implement a version of the Snake game (https://en.wikipedia.org/wiki/Snake_(video_game)).

Basic Requirements:

- A board of at least 8×8 LEDs.
- The snake is represented by a green dot head and trailing body segments (start with two trailing body segments); apples are red dots.
- When the snake head hits an apple, a point is scored, the snake grows longer, and a new apple appears.
- If the snake ever runs into any part of its own body, it dies.
- Your system should allow the snake to grow long enough for the game to become challenging.
- Your system should track and display the user score.

Hint: The easiest way to build this game is likely to build it like Tug of War – design a cell for each board position, and have it know how long to keep the light on once the snake head passes over that location.

Project 9: Venture Capitalist [To Be Negotiated]

This is a project of your choice. It must be approved by the venture capitalists – that would be your Instructor – before you can start. Set up a time to meet with your Instructor to discuss your plans well in advance. Note: *You will not receive credit for a Venture Capitalist project if you do not have a written project spec with the Instructor's signature on it in advance*.