Term Project

Assigned: 8/12/2024
Due: 29/12/2024

Electronic Battleship Game

You will design an electronic game of battleship that takes two players to play and implement it on FPGA device. You will be asked to demonstrate your implementation on FPGA device.

The playground is 2-dimensional 4x4 area for each player. You will use LEDs and seven segment display (SSD) to display inputs, turns, score, and ships sinking on FPGA.

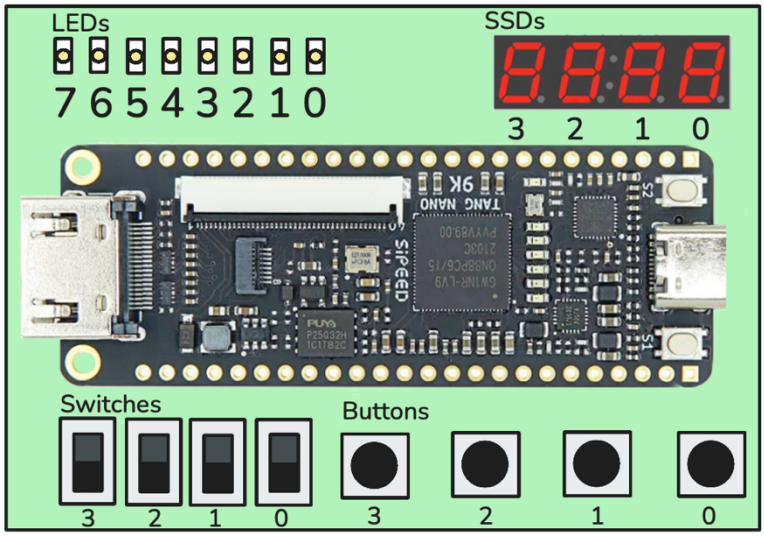

Figure 1 - FPGA Board

Assume that we have two players: A standing on the left-hand side and B on the right-hand side. The players will act using the leftmost and rightmost buttons (BTN3 for A, BTN0 for B), reset the game using "`rst`" (BTN2) and start the game using "`start`" (BTN1). After resetting, the board will display "IDLE" in the SSDs and light up the LEDs 7,4,3,0. After starting, the game will take A's ship coordinates first. The players will enter their coordinates using switches and pressing their buttons. Switches from 3 to 2 (MSB to LSB) will be the X coordinate, and switches from 1 to 0 will be the Y coordinate. It will take four inputs per player. It will show the player ("A" or "B") on SSD3 before asking for players' inputs for one second. When taking the inputs, the leftmost LED will be lit for A, and the rightmost LED will be lit for B. Again, while taking the inputs, SSD1 will display the live X coordinate (according to switch positions of switches 3,2) and SSD0 will display the live Y coordinate (according to switch positions of switches 1,0) in real time. The input count will be shown in the LEDs 5 to 4 (MSB to LSB) for A and 3 to 2 for B. The game will show "Erro" on SSDs when the player tries to place a ship on an existing ship, and light up LEDs 7,4,3,0 for one second.

After getting the inputs from both players, the initial score will be shown in the corresponding SSDs (SSD2, SSD1, SSD0) as “0-0” and the LEDs 7,4,3,0 will light up for one second. Then, starting with A, the players will take turns to shoot the ships’ coordinates on the opponent's map using their switches and buttons similar to when they were taking inputs. The X, Y coordinates will be shown on SSDs (SSD1 for X, SSD0 for Y) again in real time. While shooting, the score will be displayed on the LEDs (LED 5-4 for score of A, LED 3-2 for score of B). Also, the leftmost LED will light up indicating the turn for A, and the rightmost LED will light up indicating the turn for B.

When a player shoots at the given coordinate, the SSDs will show the updated score (“A-B” in SSD2, SSD1, SSD0) and light up all LEDs for one second if it sinks a ship. If the player does not sink a ship, all the LEDs will turn off for one second.

When a player’s score reaches four, they win the game. In this case, the score (in score SSDs) and the winner (SSD3) will be displayed. And all LEDs will blink with 1-second period (as a designer, you can change this LED dance pattern to light up certain LEDs and do a different pattern).

For the rest of the design details, you may visit office hours to interact with the finished game.

Instructions for Phase II:

In this phase, we are giving you the ASM Chart for the project. You can find it on SUCourse under “Term Project Materials”. You will take this ASM Chart as reference and design your Verilog module according to it. Furthermore, you are not allowed to change the interface of the module. This includes the signal names and module names. For example, your module name should stay as “battleship”.

You will design the contents of the battleship module (**battleship.v**), and write a testbench (**battleship_tb.v**) for testing possible gameplay scenarios. An example scenario might be the following.

The board has been reset and the start button has been pressed. Player A places their ships at (0,0), (0,1), (1,0), (1,1) and B places the ships at the same coordinates. Player A sinks three ships of B and B sinks four ships of A. Thus, the game must be won by B. You can create similar test cases to check if your design satisfies the description in this document and the ASM Chart given to you. Submitting only one testbench with one gameplay scenario is enough, however more testing is highly suggested for catching bugs in your designs.

Instructions for Phase III:

In this phase, we are giving you the modules needed for implementing your design on the FPGA. SSD module will be utilized to interface with the SSDs and will use the original clock signal. The debouncers and battleship module will use the divided clock. Clock divider module provides a 50 Hz clock signal. The assignments for **LED and SSD assignments should be done in a combinational block.**

Project Plan and Deadlines

Students must follow a project plan and demonstrate that they met a specific deadline by submitting their work. Each work item in the project plan will be graded separately. The project plan and grade of each work item is as follows:

1. REQUIREMENTS: Project requirements are given to the students.
   Dec 8, 2024. (not graded)
2. STATE DIAGRAMS: Algorithmic State Machine diagrams are submitted to SUCourse.
   15th of December, 23:59, 2024. (10 %)
3. SIMULATION: Verilog simulation (All designed Verilog modules, testbench) is submitted to SUCourse.
   24th of December, 23:59, 2024. (30%)
4. IMPLEMENTATION: The circuit is realized on FPGA and final files are submitted to SUCourse.
   3rd of January, 11:29 AM, 2024.
5. DEMO: The project is demonstrated to the assistants.
   Details will be announced on SUCourse. (60%)

Note that these deadlines are hard, and there will be no additional time.

Notes:

- You can work with your group partner in this project.
- When submitting, name the zip file as "SUID1_SUID2_tpX.zip" (e.g. 26797_22580_tp3.zip) for Phase III and two students, and "SUID1_tp3.zip" (e.g. 26797_tp3.zip) for a single person.
  - Include all modules.
- There can be a maximum of **two** group members.
- You must use Verilog.
- Do not use generative AI tools (such as ChatGPT) in your work. The AI tools might give different people similar designs. We do not differentiate or tolerate this from regular plagiarism.

The instructor has the right to have an oral interview for the term project (until the end of the final exam period.)

- Students who will have the oral interview may be selected randomly or according to a suspicious situation observed by TAs or the instructor.
- Performance of the student in an oral examination may affect their grades of the term project they have been called upon.
- If a student fails to show up at an oral exam in person, they will automatically get 0 (zero).

Appendix A – Extra Modules Needed for the Implementation on FPGA

In the final implementation of your circuit, you will need some extra modules, with which we provide you under the assignment. These are “clock divider”, “debouncer” and “seven segment driver” modules. There are comments inside the modules about the units the modules implement.

➢ The clock divider module (clk_divider.v) generates a clock signal with a frequency of 50 Hz, from a 100 MHz input clock

➢ The debouncer (debouncer.v) circuit gets the input from a push button and detects the rising edge of the push button signal. This module will be driven by the divided clock.

➢ The seven segment driver (ssd.v) module, drives the segments. This module will be driven by the clock signal, which is generated by the oscillator of the board. Also, your circuit must use the divided clock.

An SSD and its control signals ‘pgfedcba’ are shown below. Using 8-bit ‘pgfedcba’ control signals, you can display different digits and signs on an SSD. For example, ‘pgfedcba’ should be ‘00111111’ to display 0 and ‘01001111’ to display 3. There are four SSDs on the FPGA board. Your Verilog module should have 8-bit output for each SSD.

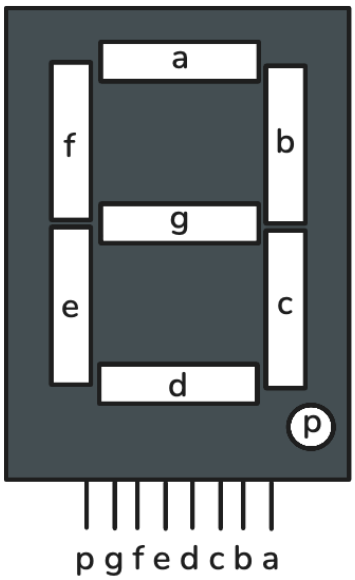

Figure 2 - Seven Segment Display Control Signals