Term Project - Electronic Mastermind Machine

You will design an electronic game of mastermind, which is a code-breaking game for two players (i.e. code-maker and code-breaker) to play, and implement it on FPGA device. You will be asked to demonstrate your implementation on FPGA device.

The match will consist of a maximum of three rounds. Before the first round, one of the players will select to be code-maker and after each round, the roles will be swapped. When a player wins two rounds, the game must return to the start state for a new game, when a player presses his/her button. Initially, the 7-segment displays (SSDs) will show “A-b” message. As one of the players presses his *enter* button – called the “code-maker” – SSD will display the initial score “0-0”, then the active player (i.e. “P-A or “P-b”) for two seconds. From then on, any input from the other player must be discarded, until the turn changes. Then, the code-maker enters four (4) letters, whose binary codes are as given in *Table I*, one-by-one and from left-to-right, using his/her *enter* button in between the letters. Each of the selected letters will be displayed in the corresponding SSD, one at a time. SSDs of the previous letters will display “-”. After the code-maker presses his *enter* button for the 4th time, the turn passes over to the other player – called the “code-breaker” – and SSD will show, again, which player’s turn it is, then the number of lives left (e.g. “L-3” before the first try) for approximately two seconds. From then on, any input from the other player must be discarded, until the turn changes.

Code	Letter	Code	Letter
000	-	100	F
001	A	101	H
010	C	110	L
011	E	111	U

Table I: Code-Letter Pairs

Then, the code-breaker enters four (4) letters, one-by-one and from left-to-right, using his *enter* button in between the letters. Each of the selected letters will be displayed in the corresponding displayer of the SSD, all given at a time. After the code-breaker presses his *enter* button for the 4th time, the LEDs of the FPGA board will indicate the correctness of his guess (details about the LEDs is discussed below). If the sequence given by the code-breaker is the same as the sequence given by the code-maker, then the score of the code-breaker will be incremented by one and after the code-breaker presses his *enter* button, the current score of the game will be displayed in the SSD for approximately two seconds. Otherwise, the circuit checks the correctness of each letter (both the letter itself and its place). The code-breaker has three (3) chances to guess the code correctly and by pressing his/her *enter* button the corresponding player can start his next guess. If he cannot find the correct code in his lifetime, then the score of the code-maker will be incremented by one. The circuit will display the correct code and then the current score of the game in the SSD for approximately 2 seconds.
At any time, active-low reset input will put the circuit in its initial state, i.e. the game starts from the scratch.

Design Details

<u>Player Inputs</u>

- “enter” button for Player A – BTN3
- “enter” button for Player B – BTN0
- reset – BTN2
- Input letters – SW2, SW1 and SW0
    - SW0: least significant bit

<u>7-segment displays (SSDs)</u>

- Start state – AN2, AN1 and AN0
    - AN2: *A*
    - AN1: ‘-‘
    - AN0: b
- Display score state – AN2, AN1 and AN0
    - AN2: *scoreA*
    - AN1: ‘-‘
    - AN0: scoreB
        - i.e. 0-0, 1-1
- Active player state – AN2, AN1 and AN0
    - AN2: *P*
    - AN1: ‘-‘
    - AN0: activePlayer
        - i.e. A, b
- Left lives display state – AN2, AN1 and AN0
    - AN2: *L*
    - AN1: ‘-‘
    - AN0: activePlayer
        - i.e. 3, 2, 1
- Code-maker enters the letters – AN3, AN2, AN1 or AN0
    - AN3: most significant bit of the input sequence
    - AN0: least significant bit of the input sequence
        - i.e.: A***, -L**, --E*, ---F, where * indicates an empty space
- Code-breaker enters the letters – AN3, AN2, AN1 or AN0
    - AN3: most significant bit of the input sequence
    - AN0: least significant bit of the input sequence
        - i.e.: A***, AL**, ALE*, ALEF, where * indicates an empty space

<u>LEDs</u>

- LD7-LD6, LD5-LD4, LD3-LD2 and LD1-LD0 pairs (name, *x*-*y*) correspond to the entered four (4) letters from the most significant bit to the least significant one, respectively
    - *xy* = 11, if the letter is correctly placed
    - *xy* = 01, if there is such letter in the sequence but his place is incorrect
    - *xy* = 00, if there is no such letter in the sequence
- All LEDs will blink to indicate the end of the game. (See the demo board.)

Bonus & Additional Work as a Makeup for a Missed Lab (or poorly performed one, individual work required.)

If a student missed a lab, (s)he can do this part as a makeup for the missed lab. If you do not prefer to substitute a lab grade, this part will be reflected to your term project grade as 10% bonus.

This time, Player A will compete against the circuit as code-breaker (the circuit will act as code maker.). If SW3 is turned on then Player A presses his/her enter button, the circuit will turn to single player mode. In this mode, the circuit will generate a 4-letter code, randomly using a random number generator (RNG). RNG module (rng_module.v) generates 1-bit random number in each clock cycle. When the circuit enters single player mode, your design will prepare a 4-letter code. Then, Player A will act as code-breaker. S/he will have three lives and will play for one round only.

An Example Template

See the attachment…

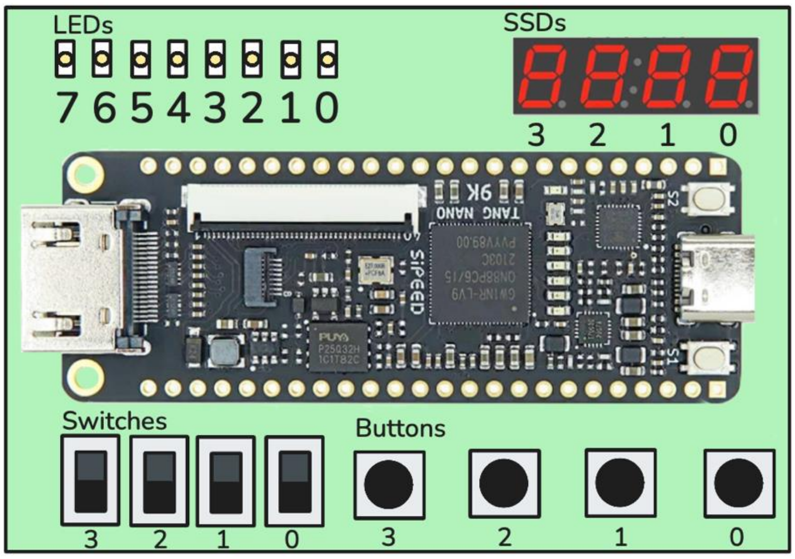
Details on Simulation Phase

During the simulation demos, you are only expected to show the functionality (i.e. the correct transitions between the states) of your Finite State Machine (FSM) circuitry. You do not need to add any other parts to your design during the simulation demo. Besides, the simulation demo should be performed on the <u>simplest version of the final project description</u>, i.e. it should NOT include the bonus part.

For this phase, you are expected to develop a test scenario which covers the following cases:

- You may assume that the clock frequency is 2 Hz (just wait 4 clock cycles to display any message for 2 seconds.)
- Player A starts the game.
- Player A enters the code. During this state,
    - enter B button should not work.
    - “-” is not accepted as a letter.
- Player B guess the correct code.
- Player B enters the code. During this state,
    - enter A button should not work.
    - “-” is not accepted as a letter.
- Player A do not guess the correct code.

Details on Implementation Phase



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 mastermind modules 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.** You can only modify the contents of **mastermind** and **top** modules.

See the FPGA Setup Guide to see how to use the board.

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 18, 2025 (not graded)

2. ASM CHART: Algorithmic State Machine diagrams are submitted to SUCourse.
   Dec 25, 2025 23:55 (%10)

3. SIMULATION: Icarus Verilog simulation is submitted to SUCourse.
   Jan 01, 2026 23:55 (%30)

4. IMPLEMENTATION: The circuit is implemented on FPGA and final files are submitted to SUCourse.
   Jan 08, 2026 23:55 (%60)

5. DEMO: The project is demonstrated to the assistants. (Attendance mandatory)
   TBA (%0)

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

Notes:

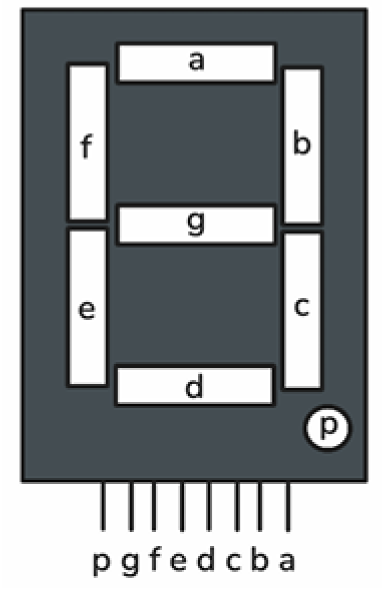
- You can work with your lab partner in this project.
- You are expected to use Verilog language.
- You are required to use only the Verilog primitives (syntax, operations etc.), which were covered in CS 303.

Appendix A – Extra Modules Needed for the Implementation on FPGA

In the final implementation of your circuit, you will need some extra modules, with are provided 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 the clock source of the board
- 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. There are four SSDs on the FPGA board.


a
f
b
g
e
c
d
p
p g f e d c b a