**Department of Electrical and Electronic Engineering**

**Islamic University of Technology (IUT)**
**A subsidiary organ of the OIC**

# Project Manual

## EEE 4308: Digital Electronics Lab

**Name of the Project:**
Digital Snake Ladder Game

# Project Description

## Introduction:

The digital version of the Snake and Ladder game is played on a 4x4 LED grid board that represents the game positions, ranging from 0 to 15. Each block on the grid corresponds to a unique game position, and the players' movements are visually represented by LEDs. The Board Module serves as the visual representation of the digital snake and ladder game. Each block contains four LEDs: Player 1 LED indicates Player 1's position. Player 2 LED indicates Player 2's position. Snake LED marks the presence of a snake on that block. Ladder LED marks the presence of a ladder on that block.

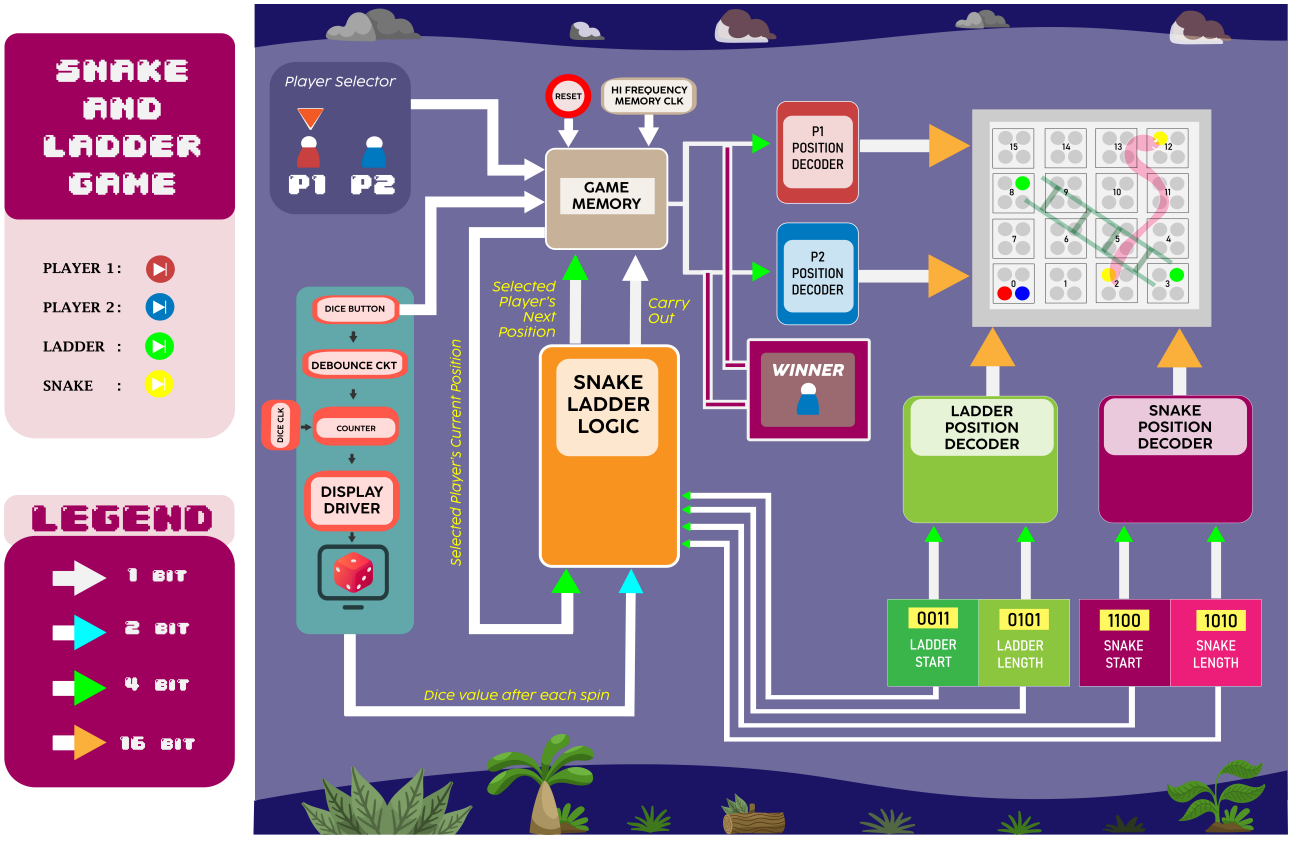



**Fig. 1: Block diagram of the project.**

Different colored LEDs are used for each type to ensure easy identification. For instance, you might use green LEDs for snakes shown in figure 2. The snake and ladder LEDs are statically lit to represent the fixed positions of snakes and ladders on the board. The player LEDs are controlled based on the players' current positions from the Memory Module. The board features different colored LEDs to indicate the positions of the players, as well as the locations of snakes and ladders. Player 1 and Player 2 each have their designated LED colors, and as they move across the board based on dice rolls, their positions are updated in real-time. Snakes and ladders are permanently represented on the board by static LEDs, with snakes pulling players down from higher positions to lower ones, and ladders advancing players upwards. The objective of the game is to be the first

player to reach the last position (position 15), and the game provides visual feedback using LEDs to signal when a player has won.

## Project Specification

**Dice Module:** This module generates a 2-bit pseudo-random number between 1 and 3. The Dice Module uses a high-speed counter to generate a random number between 1 and 3 and shows the result on a 7-segment display, simulating a dice roll. The quick number cycling and the player's timing while releasing the dice button provides the unpredictability factor.

**Board Module:** The game board consists of 16 blocks (0-15), each block featuring four LEDs to represent Player 1, Player 2, snakes, and ladders. Using LEDs organized in a 4x4 grid, the Board Module provides a visual representation of the game board. Player positions, snakes, and ladders are displayed with distinct colored LEDs for simple identification.

**Snake and Ladder Logic Module:** The Snake and Ladder Logic Module, which determines the current player's next 4-bit location depending on the dice value and whether any snake or ladder has been encountered or not. The current position, the value of the dice, and the locations of the snakes and ladders on the board are among the inputs that this module handles.

**Player Selector Module:** The Player Selector Module controls player involvement by letting users switch between person 1 and Player 2.

**Memory Module:** Using flip-flops or registers, the Memory Module keeps track of both players' current locations between turns and is also connected with snake ladder logic module from which it receives the next location of the player.

**Reset Module:** To facilitate game control, the Reset Module provides a means to reset the game to its initial state, resetting player positions without the need for power cycling.

**Decoder module (Position Decoders):** The system also has Position Decoders, which are independent 4 to 16 decoders for player positions, ladders, and snakes.

**Winning Module:** When a player reaches position 15, the Winning Module recognizes it, activates buzzers and LEDs to indicate a win, and notifies the other players right away of the result of the game.

## Submodule/Block Specification:

This section discusses the specifications/requirements for each block/submodule shown in the project diagram (Fig. 1) -

## Dice Module Specification

Every time the dice button is pressed, the Dice Module is made to produce a random integer between 1 and 3. To guarantee randomization, the high-speed counter cycle and the intrinsic unpredictability of the button release timing are used. The module rapidly cycles through the numbers 1 through 3, which are displayed on a 7-segment display. The dice button initiates the counter, and when the button is released, it stops and locks in the current number determined by the dice roll. Furthermore, the module has an RC time constant that can be changed to modulate the clock frequency, allowing the counter to slow down. This slowing down helps us see the counter cycling through the numbers for testing and demonstration periods.

| Input | Description |
|---|---|
| Dice Button | Triggers the dice roll and starts cycling numbers. |

| Output | Description |
|---|---|
| Random Number (2 bits) | Generates a pseudo- random integer between 1 and 3 |

A counter system, clock generator, button control, and display interface are used in the construction of the digital dice simulator for random number production between 1 and 3. The JK or D-type flip-flops can be used in the construction of the counter. The counter is driven by a high-frequency clock generator that may be made with an astable 555-timer or an RC oscillator. The dice button control mechanism uses a debounce circuit to remove false triggers from mechanical bounce. A 7-segment display linked to a BCD to 7-segment decoder is used to show the output. The debounce circuit, clock generator, counter, display driver, and 7-segment display itself are among the crucial parts of the system that are shown in the block diagram. Functionality-wise, the dice button is pressed to start the counter, which quickly cycles through the numbers until the button is released, at which time the held and displayed number is revealed.

## Snake and Ladder Logic Module Specification

The Snake and Ladder Logic module determines the next position of the player after rolling the dice in the digital snake and ladder game. It processes the current player position, the dice value, and the positions and lengths of snakes and ladders to compute the player's new position. This module ensures that if a player lands on a snake or ladder, their position is adjusted accordingly.

| Input | Description |
|---|---|
| Current Player Position | 4-bit value representing the current player position |
| Dice Value | 2-bit value representing the dice roll |
| Snake Head Position | 4-bit value representing the head of the snake |
| Snake Length | 4-bit value representing the length of the snake |
| Ladder Start Position | 4-bit value representing the bottom of the ladder |
| Ladder Length | 4-bit value representing the length of the ladder |

| Output | Description |
|---|---|
| Next Player Position | 4-bit value representing the updated player position after accounting for dice roll, snake, and ladder |

It is intended to identify a player's future location on the game board following a dice roll. This module considers the player's present location, the value of the dice, and the arrangements of the snakes and ladders on the board. The module initially computes a tentative location by adding the dice value to the existing position using adders. It then uses comparators to see if this approximate location aligns with the head of a snake or the bottom of a ladder.

When the player lands on a snake, their position is decreased by the length of the snake; however, if a ladder is met, their position is increased by the length of the ladder. One way can be a multiplexer that is driven by the comparator outputs chooses the right next position based on these criteria. Additionally an optional criteria is that, priority logic has to be implemented to handle scenarios where both a snake and a ladder condition might be met, prioritizing ladder movements over snakes to maintain. The module operates in a straightforward manner, calculating the approximate position, checking for ladders or snakes, modifying the position as necessary, and then producing the final position for the next player.

## Player Selector Module Specification

It allows the game to identify which player's turn it is by selecting between Player 1 and Player 2. This module uses switches to input the player's selection and provides a signal that can be used by other modules to process the game logic accordingly.

| Input | Description |
|---|---|
| Player Selector Switch | Toggle between Player 1 and Player 2 |

| Output | Description |
|---|---|
| Active Player Signal | A logic signal that indicates which player is currently active. This goes to an LED. |

This module lets players easily switch between Player 1 and Player 2 by either physical switches or a toggle mechanism. Other system parts, such memory and game logic modules, employ the matching player indication signal that the module creates by analyzing the status of these switches to perform activities for the active player. Pull-down resistors, debounce circuits, and visual indicators like LEDs are used in the design to provide player selection by giving a signal of the active player. One way can be setting up Player 1 and Player 2 to be represented by either two independent switches or a single toggle switch. You can use pull-down resistor if needed. Another alternative way is the output of Player 1 can be connected to ground (0V) and Player 2 to Vcc (5V) via a single toggle switch. To ensure compatibility with other digital modules, the output signal is given a logic low (0) for Player 1 and a logic high (1) for Player 2. LEDs has to be added as visual indicators to clearly show the current player. Debounce circuits reduce erroneous triggers caused by mechanical switch bounce.

## Memory Module Specification

The Memory Module is responsible for storing and updating the current positions of each player in the digital snake and ladder game. It interfaces with the Snake and Ladder Logic module to receive the next position of a player after a dice roll and updates the stored position accordingly. The module handles two players, each with their own 4-bit memory segment to represent positions on a 4x4 game board (positions 0 to 15).

| Input | Description |
|---|---|
| Current Player Selector | Signal indicating the current active player |
| Next Position | 4-bit value representing the player's next position |
| Dice Pressed Signal | Trigger for updating position based on dice roll |
| Carry Out (optional) | A signal from the Snake and Ladder Logic module, possibly indicating an overflow or special condition (e.g., reaching the end of the board). |

| Output | Description |
|---|---|
| Player 1 Current Position | 4-bit value representing Player 1’s current position |
| Player 2 Current Position | 4-bit value representing Player 2’s current position |

This module can employ two separate 4-bit D-type flip-flop registers, each dedicated to storing the current position of Player 1 and Player 2 respectively. A demultiplexer (DEMUX) can be utilized to select the appropriate register for writing based on the active player, as determined by the Player Selector input. When a dice is rolled, the Snake and Ladder Logic module calculates the Next Position, which is then routed to the selected player's register through the DEMUX, ensuring that only the active player's position is updated while the other remains unchanged. Combine Current Player Selector and Dice Pressed signals so that when the dice is pressed only then the memory is updated. You may do that using a Parallel In Parallel Out (PIPO) shift register. In order to route the stored player position to the player position decoder you may use a multiplexer (MUX). Additionally, a reset function initializes both registers to the starting position at the beginning of the game. Both registers can have access to the Next Position input, but only the targeted register will take the data. Since the outputs from both registers are always available, real-time player locations may be retrieved and used by modules such as Snake and Ladder Logic and the Display.

## Reset Module Specification

It offers a straightforward way to return the game to its starting point. The Memory Module receives a signal from the reset switch, which returns both players' locations to the initial state (e.g., position 0). This guarantees that the game may be resumed whenever desired without requiring a system power-cycle.

| Input | Description |
|---|---|
| Reset Button | A physical push-button switch that, when pressed, generates a reset signal. |

| Output | Description |
|---|---|
| Reset Signal | A logic signal sent to the Memory Module to reset the players' position registers. |

When the reset switch is hit, the Reset Module resets the positions of both players, allowing the game to revert to its original condition. Upon activation, the module sends the Memory Module a reset signal, which causes the Memory Module to reset the current locations of both players to the initial position (e.g., position 0). When the switch is disengaged, the reset signal becomes inactive and resumes its active state, allowing for regular gaming. In order to provide a precise and clean reset, the module also has a debounce circuit that stops erroneous signals caused by mechanical switch bounce. When the reset is engaged, an optional LED indication can be added to give visual cues that the game is prepared for a restart.

## Board Module Specification

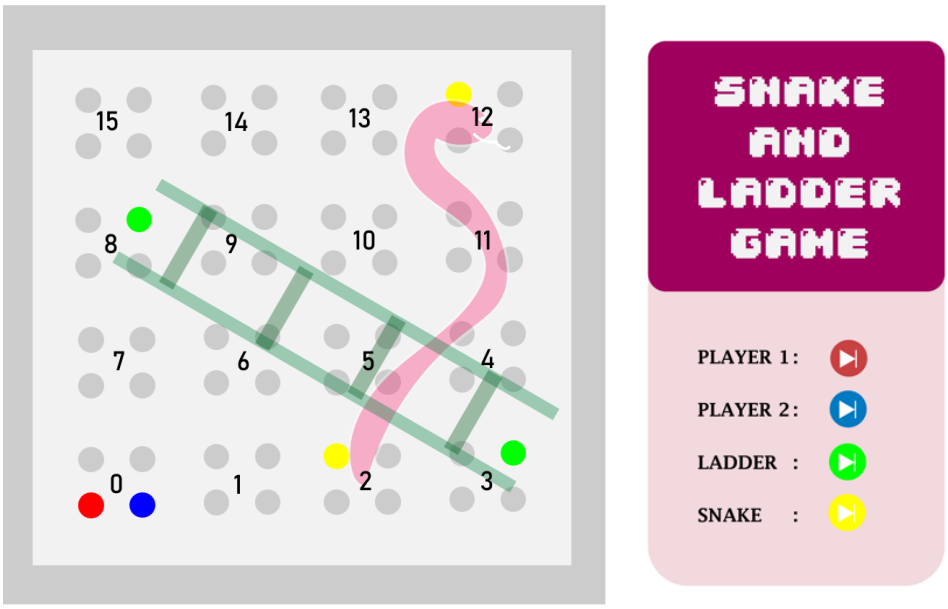



**Figure 2: The Board Module**

The Board Module serves as the visual representation of the digital snake and ladder game. It consists of a 4x4 grid, totaling 16 blocks, corresponding to positions 0 to 15 on the game board. Each block contains four LEDs: **Player 1 LED:** Indicates Player 1's position. **Player 2 LED:** Indicates Player 2's position. **Snake LED:** Marks the presence of a snake on that block. **Ladder LED:** Marks the presence of a ladder on that block. The digital Snake & Ladder game's Board Module depicts the numbers 0 through 15 on a 4x4 grid, with each block denoting a distinct 4-bit binary number. Four LEDs are built into each block: red for the first player, blue for the second, green for snakes, and yellow for ladders. In order to depict the permanent locations of snakes and ladders on the board, the snake and ladder LEDs are static and illuminated.

Refer to **Figure 2**, where the ladder between positions 3 and 8 is highlighted by a green LED. When a player lands on position 3, they will automatically be moved to position 8 through the ladder. Similarly, observe the yellow LED snake between positions 12 and 2. When a player reaches position 12, which contains the snake's head, they will be "eaten" and moved back down to position 2. The player LEDs, on the other hand, are dynamic and update according to the dice

roll. If both players are in the same position, both player LEDs can be lighted at the same time on the same block. When a player reaches the last block (position 15), they have won the game.

The snake and ladder LEDs on block 0 of each player's LEDs light up their respective locations when the game starts. The Memory Module adjusts the player's location based on dice rolls, and this information is then transmitted to the Board Module decoders. These decoders turn off the previous player LEDs and activate the relevant ones on the updated position. If any player reaches the last position he has won. Power supply capacity must be sufficient to handle the cumulative current draw of all active LEDs. Proper layout and wiring are essential for signal integrity, minimizing voltage drops, and ensuring reliable operation.

## Decoder module Specification

### Snake Position Decoder

The locations of the snakes on the gaming board are determined by the Snake Position Decoder Module. It computes the position of the snake's tail using two 4-bit inputs: the length and the snake's beginning position (head). The Snake LEDs on the board are then controlled by the module by decoding these positions, which shows where the snake's head and tail are. To implement this, the module first uses a 4-bit subtractor to compute the tail position. Then, two decoders are employed: one for the snake's head and another for the tail. Each decoder is a 4-to-16-line decoder that activates the respective LED at the calculated head and tail positions on the game board. The head position decoder activates the LED corresponding to the snake's starting point, while the tail position decoder activates the LED corresponding to the calculated end point. Current-limiting resistors are used with the LEDs to prevent overcurrent and ensure stable operation.

| Input | Description |
|---|---|
| Snake Head Position | 4-bit value representing the snake head |
| Snake Length | 4-bit value representing the snake Length |

| Output | Description |
|---|---|
| Snake Head LED | Lights up LED representing the snake head |
| Snake Tail LED | Lights up LED representing the snake tail |

### Ladder Position Decoder Module Specification

The ladder locations on the game board are ascertained and shown by the Ladder Position Decoder Module. The Ladder Start Position (bottom) and Ladder Length are its two 4-bit inputs. By adding the length to the start position, a 4-bit adder calculates the ladder top position. Afterwards, the

Ladder Top and Ladder Bottom locations on the board are lit by means of two 4-to-16 decoders controlling LEDs. The LEDs are protected from damage by current-limiting resistors, and the module makes sure that overflow situations are handled correctly to avoid incorrect locations (this part is optional).

| Input | Description |
|---|---|
| Ladder Bottom Position | 4-bit value representing the ladder bottom |
| Ladder Length | 4-bit value representing the ladder Length |

| Output | Description |
|---|---|
| Ladder Bottom LED | Lights up LED representing the ladder bottom |
| Ladder Top LED | Lights up LED representing the ladder top |

**Player Position Decoders Module Specification**

The **Player Position Decoders Module** is responsible for displaying the current positions of Player 1 and Player 2 on the game board by controlling their respective LEDs. It receives 4-bit inputs for each player's position from the Memory Module and decodes these positions using two separate 4-to-16-line decoders—one for each player. The decoders activate the appropriate LED on the board corresponding to each player's current position. To ensure proper operation, current-limiting resistors are used to protect the LEDs. The module is designed to handle situations where both players occupy the same block, ensuring both Player 1 and Player 2 LEDs can light up simultaneously without interference.

| Input | Description |
|---|---|
| Player 1 Current Position | 4-bit value representing Player 1's position |
| Player 2 Current Position | 4-bit value representing Player 2's position |

| Output | Description |
|---|---|
| Player 1 LED | Lights up Player 1's position LED |
| Player 2 LED | Lights up Player 2's position LED |

## Winning Module

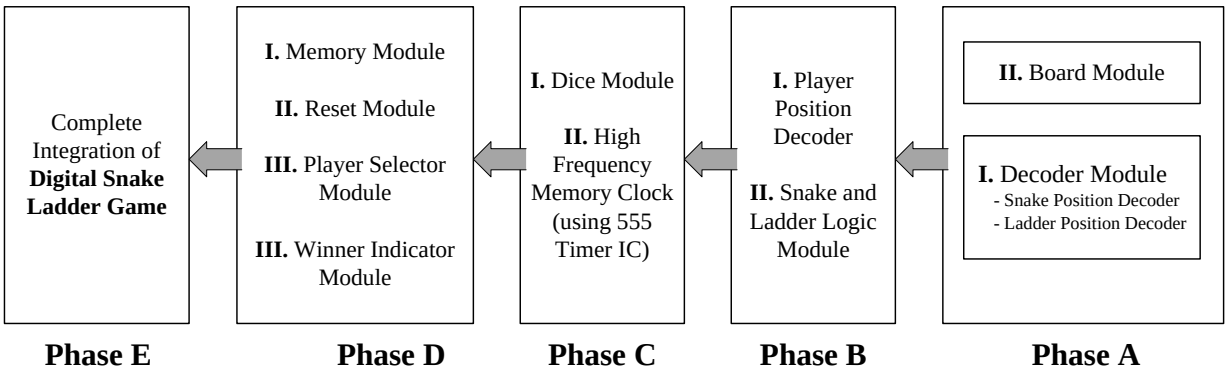
The Winning Module compares each player's current position to binary 1111 in order to determine when a player achieves the ultimate position (15). A dedicated LED and buzzer on the winning player's profile turn on if a match is detected. You can use comparators to design this module

| Input | Description |
|---|---|
| Player 1 Current Position | 4-bit value representing Player 1’s position |
| Player 2 Current Position | 4-bit value representing Player 2’s position |

| Output | Description |
|---|---|
| Player 1 Win Indicator | Activates Player 1’s win LED and buzzer |
| Player 2 Win Indicator | Activates Player 2’s win LED and buzzer |

# Evaluation Process

## Design Phases:




## Rubrics for Project:

| Criteria | Scoring Level | | | | Maximum percentage out of the total marks for the project |
|---|---|---|---|---|---|
| | **Excellent** | **Good** | **Satisfactory** | **Not Satisfactory** | |
| | 100% of total percentage allotted | 80% of total percentage allotted | 60% of total percentage allotted | = or < 40% of total percentage allotted | |
| **Phase A** | Submodules of Phase A are perfectly designed and working with best performance | Submodules of Phase A are designed adequately but needs further modification | Submodules Phase A are designed moderately and working with errors | Submodules of Phase A are poorly designed and not working | **15%** |
| **Phase B** | Submodules of Phase B are perfectly designed and working with best performance | Submodules of Phase B are designed adequately but needs further modification | Submodules Phase B are designed moderately and working with errors | Submodules of Phase B are poorly designed and not working | **25%** |
| **Phase C** | Submodules of Phase C are perfectly designed and working with best performance | Submodules of Phase C are designed adequately but needs further modification | Submodules Phase C are designed moderately and working with errors | Submodules of Phase C are poorly designed and not working | **15%** |
| **Phase D** | Submodules of Phase D are perfectly designed and working with best performance | Submodules of Phase D are designed adequately but needs further modification | Submodules Phase D are designed moderately and working with errors | Submodules of Phase D are poorly designed and not working | **25%** |
| **phase E** | Whole system is perfectly integrated and working with best performance | Whole system is integrated appropriately but needs further modification | Whole system is moderately integrated and working with errors | Whole system is poorly integrated and not working | **20%** |

- The marks allocated on the above rubrics are shown as percentage of the full marks allowed for the **Project Completion** only. Marks distribution of EEE 4308 course is provided below:

| Assessment Method | Marks (as %) |
|---|---|
| Lab Performance | 15% |
| Lab Report | 10% |
| Open Ended Lab | 25% |
| **Project Completion** | **40%** |
| Project Presentation | 5% |
| Project Report | 5% |

- In each of the criteria mentioned in the rubrics, **50%** of that section's marks will be for **software simulation** and **50%** of the mark will be for **hardware implementation**. For example, if 20% of the total project mark is allocated for Phase E, then 10% mark is for software simulation and 10% mark is for hardware implementation in Phase E.

- If a group fails to complete a Design Phase within its deadline and submits after the deadline, they will be **penalized by one scoring level** for each one-week delay.

  For example, if a group submits Design Phase B after its given deadline and their submitted design is working perfectly, still they will not get "Excellent" scoring level (100%) for Design Phase B. They will get "Good" scoring level (80%), if their delay is less than a week. If they miss the deadline by more than one week, they will receive a "Satisfactory" score (60%). If the delay is more than two weeks, they will get "Not Satisfactory" score (= or < 40%).

Mapping of Rubrics and Program Outcomes (POs):

| Criteria | Program Outcomes (POs) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| **Phase A** | | √ | √ | | √ | | | | √ | | | |
| **Phase B** | | √ | √ | | √ | | | | √ | | | |
| **Phase C** | | √ | √ | | √ | | | | √ | | | |
| **Phase D** | | √ | √ | | √ | | | | √ | | | |
| **Phase E** | | √ | √ | | √ | | | | √ | | | |

Mapping of Rubrics and Program Outcomes (POs) with Marks Distribution:

| Criteria | Marks (as %) | Mark distributions (as %) on POs | | | |
|---|---|---|---|---|---|
| | | PO2 | PO3 | PO5 | PO9 |
| **Phase A** | 15% | 3.75% | 3.75% | 3.75% | 3.75% |
| **Phase B** | 25% | 6.25% | 6.25% | 6.25% | 6.25% |

| | | | | | |
|---|---|---|---|---|---|
| **Phase C** | 15% | 3.75% | 3.75% | 3.75% | 3.75% |
| **Phase D** | 25% | 6.25% | 6.25% | 6.25% | 6.25% |
| **Phase E** | 20% | 5% | 5% | 5% | 5% |
| **Total** | **100%** | **25%** | **25%** | **25%** | **25%** |

For continuous evaluation and supervision of each group, one course teacher will be designated. For each design phase, all groups must **demonstrate their project progress** and **submit a progress report** to their respective supervisors. At the same time, they can consult with their supervisor whenever they face any problem regarding the project. The list of project groups and their supervisor is given below:

| Group Name | Supervisor |
|---|---|
| ChatGPT | Mr. Hayder Jahan Parash |
| Project ASCII | |
| Hexabit | |
| Faycal's Team | |
| Bitter/s | Mr. Nadim Ahmed |
| Team BitsQ | |
| ENIAC | |
| DYNAMIC THINKER'S | |
| Linear | Mr. A. K. M. Rakib |
| NSB | |
| Binary Nexus | |
| Team Cybertron | |
| Team Sadaqah | |
| Green Card | Mr. Abdullah Taharat |
| Watt Wizards | |
| Mini Mints | |
| Automata | |
| Trailblazers | |
| Team Muffins | Mr. Mohammad Abrar Kabir |
| Maroon 6 | |
| Team Whiplash | |
| Grill Chicken | |
| Team genjam | Mr. Ahmed Jawad Rashid |
| Possible Team | |
| Electrifying | |
| Team name to be decided by the members | |

## Suggested List of Components:

- 555 timer IC must be used to generate clock pulse.
- The following is a list of suggested ICs that may be used in the project:

| Functional Name | Type |
|---|---|
| Basic Logic Gate | AND, OR, XOR, NAND, NOR, XNOR, NOT (2 input, 3 input, 4 input, etc) |
| Decoder | 4 to 16, BCD to Seven segment |
| Demultiplexer | 1 to 2 |
| Multiplexer | 2 to 1 |
| Adder | Binary |
| Flip-flop | J-K, D, SR Latch |
| Register | 4-bit |
| Comparator | Binary |
| Other ICs | 7-Segment Display, LEDs, Switches |

However, if you need to use any IC other than the ICs mentioned above, you are advised to take prior permission from your course teachers.

## Tips for fixing CPU load issue:

1) Go to System which is on the top bar
2) Select Set Animation Options and then select SPICE Options
3) Change "Default Settings" (Bottom left) into "Settings for Better Convergence" from the drop-down menu
4) Press load then press ok

## Report Outline:

- Title Page: Course code and name, Project name, Group Name, Group members' name and student ID
- Main Body: Objective and brief description of the Project
  For Each Phase: Block Diagram, algorithm, Truth Table, Boolean expression, Circuit Diagram, Equipment list, PCB design diagram, pictures of hardware implemented Phase.
  Complete Project: Block Diagram, Circuit Diagram, PCB design diagram, pictures of hardware implemented Project.
- Budget/Expenditure of the Project
- Problems Faced
- Discussion: Outcomes, Limitations, Future development

## Project Implementation and Submission Timeline:

| Task | Submission Deadline* |
|---|---|
| Project Introduction | November 10, 2024 |
| Phase A and B (SW) | November 19, 2024 |
| Phase C, D, E (SW) | November 29, 2024 |
| Phase A, B, C (HW) | December 9, 2024 |
| Phase D, E (HW) | December 16, 2024 |
| Complete Project Submission | **December 17, 2024** |
| Project Demonstration/Presentation | December 18, 2024 |
| Project Report Submission | December 20, 2024 |

*****On or before

**The project groups are highly encouraged to submit the complete project before the deadline which will provide extra advantage in getting better grades.**

## A note on PLAGIARISM

Plagiarism is the misrepresentation of the work of another as your own. It is academic theft; it is a serious infraction of the University Honor Code. Instances of plagiarism or any other cheating will be reported to the Department and will at the very least result in failure of this course.