Middle East Technical University - Department of Computer Engineering

CENG 232

Logic Design

Spring '2023-2024

Lab 4

Part 1 & Part2 Due Date: 2 June 2024, Monday, 23:55
No late submissions

1 Part 1: Bitwise Manipulation Memory (BMM) (30 Points)

1.1 Introduction

This assignment involves implementing a combined memory system called the Bitwise Manipulation Memory (BMM) System, integrating Read-Only Memory (ROM) and Random Access Memory (RAM) to facilitate both static data retrieval and dynamic data manipulation.

1.2 System Overview

The BMM system is designed to leverage static data stored in ROM as reference or control signals for dynamic operations performed in RAM. This system aims to illustrate the practical applications of combining different types of memory to achieve more complex functionality in digital systems.

1.3 Module Descriptions

1.3.1 ROM Module

The ROM module stores **8-bit read-only data that are addressed in 3 bits.** It operates as follows:

- **Works asynchronously**, i.e., not triggered by a clock pulse.
- Assigns the value that resides in the given index of the ROM to **dataOut**.

Port Definition:

```
module ROM (
    input [2:0] addr,        // Address input
    output reg [7:0] dataOut    // Data output
);
```

Note: Table 1, below, shows the data values stored in the ROM:

Table 1: Initial Values Stored in the ROM

Address	Data Stored (8-bit)
0	00000000
1	00000001
2	00000010
3	00100011
4	11000100
5	01100101
6	00011110
7	01000101

1.3.2 Bitwise Manipulation RAM Module

The RAM module in this system performs dynamic operations on data inputs, influenced by the outputs of the ROM module and external inputs.

Functionality:

- **Write Mode:** Performs specified bitwise operations between the input data (`dataIn`) and the data fetched from ROM (`romData`). Write mode **works synchronously**, **triggered by the positive edge of the clock signal** (`CLK`).
- **Read Mode:** Simply retrieves and outputs data stored at a specified RAM address. Read mode **works asynchronously.**

Operation Codes:

- 00 ⇒ AND
- 01 ⇒ OR
- 10 ⇒ XOR
- 11 ⇒ NAND

Port Definition:

```
module Bitwise_Manipulation_RAM (
    input mode,                     // 0 for write, 1 for read
    input [2:0] addr,               // RAM address
    input [1:0] operation,          // Operation code
    input [7:0] dataIn,             // Input data
    input [7:0] romData,            // Data from ROM used in operations
    input CLK,                      // Clock signal
    output reg [7:0] dataOut        // Output data
);
```

1.3.3 Combined Memory System Module

This upper module orchestrates the operations between the ROM and RAM modules, effectively combining their functionalities.

Functionality:

- Controls the flow of data between the ROM and RAM.
- Manages addresses and operation codes based on system requirements.

Port Definition:

```
module Combined_Memory_System (
    input mode,                         // W/R mode of RAM
    input [2:0] systemAddr,             // System address
    input [7:0] dataIn,                 // System data input
    input [1:0] operation,              // Operation code
    input CLK,                          // Clock signal
    output [7:0] systemOutput           // System output
);
```

2 Part 2: Seating System (70 Points)

In this part of the assignment, your task is to enhance the Seating system introduced in the 2nd part of lab 3. Here, the seating system is expanded to have 4 categories of areas: Loud, Quiet, Individual, and Zoom Rooms, each with all plugged seats, eliminating the need for a plug option. Additionally, each area has 16 seats, all free of charge. Here, the seating management system operates in four modes, each tailored for specific operations: Student Entrance Mode, Student Exit Mode, Search Mode, and List Mode.

2.1 Module Definition

```
module SeatingSystem(
    input [1:0] mode, // Modes: 01 for Entrance, 00 for Exit, 10 for Search, 11 for List
    input [5:0] userID, // Format: [AreaID(2 bits)][UserID(4 bits)]
    input CLK,
    output reg [1:0] selectedAreaId, // Shows user-selected area
    output reg [5:0] numberOfInsideUser, // Number of users inside
    output reg [5:0] listOutput, // Lists the IDs of students in the area
    output reg AlreadyInside, // Indicates if a student is inside
    output reg NotInside  // Indicates if a student is not inside
);
```

2.2 System Specifications

1. **I/O Specifications:**

   (a) `Inputs:`

   - `mode`: Indicates the mode of the system (01 for Student Entrance Mode, 00 for Student Exit Mode, 10 for Search Mode, 11 for List Mode).
   - `userID`: A 6-bit number with the first two bits specifying the area ID (00 for Loud, 01 for Quiet, 10 for Individual, 11 for Zoom rooms) and the remaining four bits representing the user's unique ID within that area.
   - `CLK`: System clock signal.

   (b) `Outputs:`

   - `selectedAreaId`: 2-bit output indicating the area selected by the user.
   - `numberOfInsideUser`: 6-bit output showing the current number of users inside a selected area.
   - `listOutput`: 6-bit output entry of the list of the IDs of students currently in a selected area.
   - `AlreadyInside`: Flag that indicates if a student is inside the selected area.
   - `NotInside`: Flag that indicates if a student is not inside the selected area.

2.2.1 Modes of Operation

(a) **Student Entrance Mode (01):**

- Entrance is permitted only in this mode.
- A student can enter an area if **their ID's area code (leading 2 bits of `userID`) matches the area ID** of the area they are attempting to enter and **the seat designated by their ID is not already occupied**.
- If the seat is occupied or the same person is already in the area, the system issues an `AlreadyInside` warning.
- Successful entry updates the system with the student's ID and adjusts `selectedAreaId` and `numberOfInsideUser`.

(b) **Student Exit Mode (00):**

- Exit is allowed only in this mode.
- A student can leave only if they are already inside the area.
- Exiting updates the system by removing the student's ID and adjusts `selectedAreaId` and `numberOfInsideUser`.

- If the student is not found inside, the system issues a `NotInside` warning.

(c) **Search Mode (10):**

- This mode checks if a student with the given ID is inside the area specified by the first two bits of `userID`.
- Updates `AlreadyInside` or `NotInside` accordingly, along with `selectedAreaId`.

(d) **List Mode (11):**

- Lists all student IDs present in the specified area in the order they were recorded by updating `listOutput` at each CLK pulse, cycling through reserved seats. **We will provide showcases.**

2. **Area Capacities:** Each area has a capacity of 16 seats.
3. **Initial Conditions:** All areas start empty, with all seat records set to 0.

2.3 Figure Reference

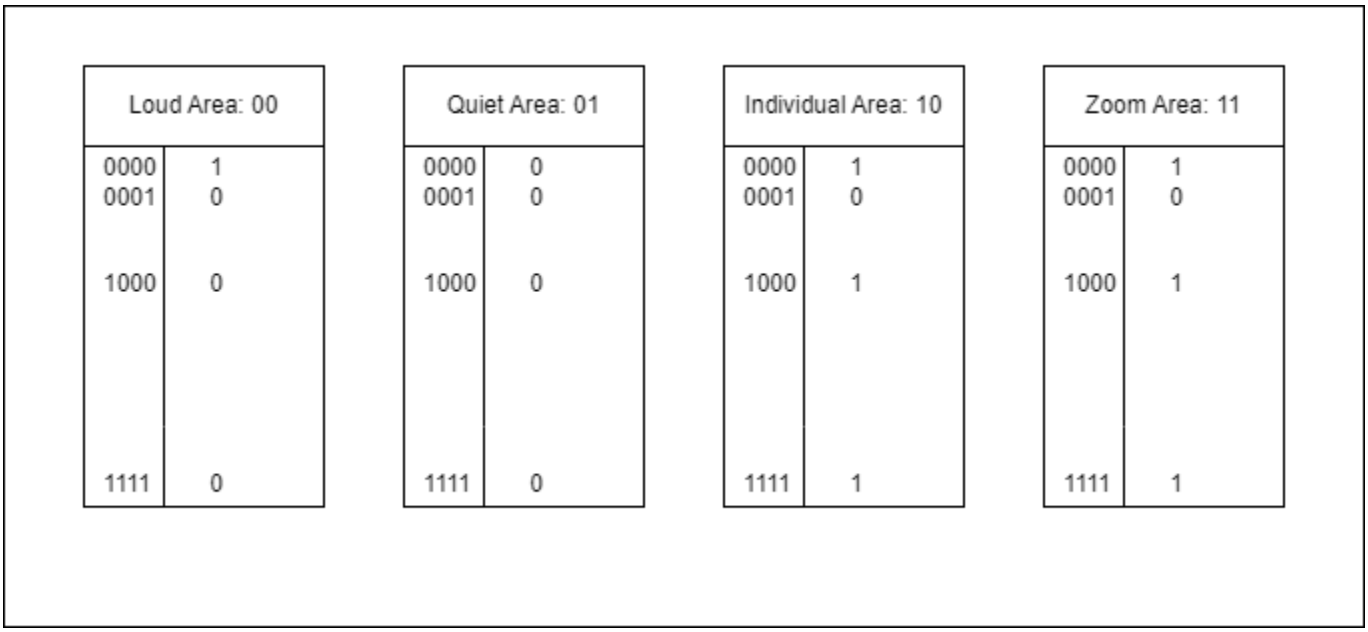

Figure 1: Example of Area Layout

2.4 Input/Output Specifications

Type	**Name**	**Description**
Input	`mode`	2-bit mode selector
Input	`userID`	6-bit ID
Input	`CLK`	Clock signal
Output	`selectedAreaId`	2-bit output showing area
Output	`numberOfInsideUser`	2-bit output showing users
Output	`listOutput`	4-bit output listing IDs
Output	`AlreadyInside`	Flag for user presence
Output	`NotInside`	Flag for user absence

Table 2: Input/Output Specifications of the Seating System Module

2.5 FPGA Implementation

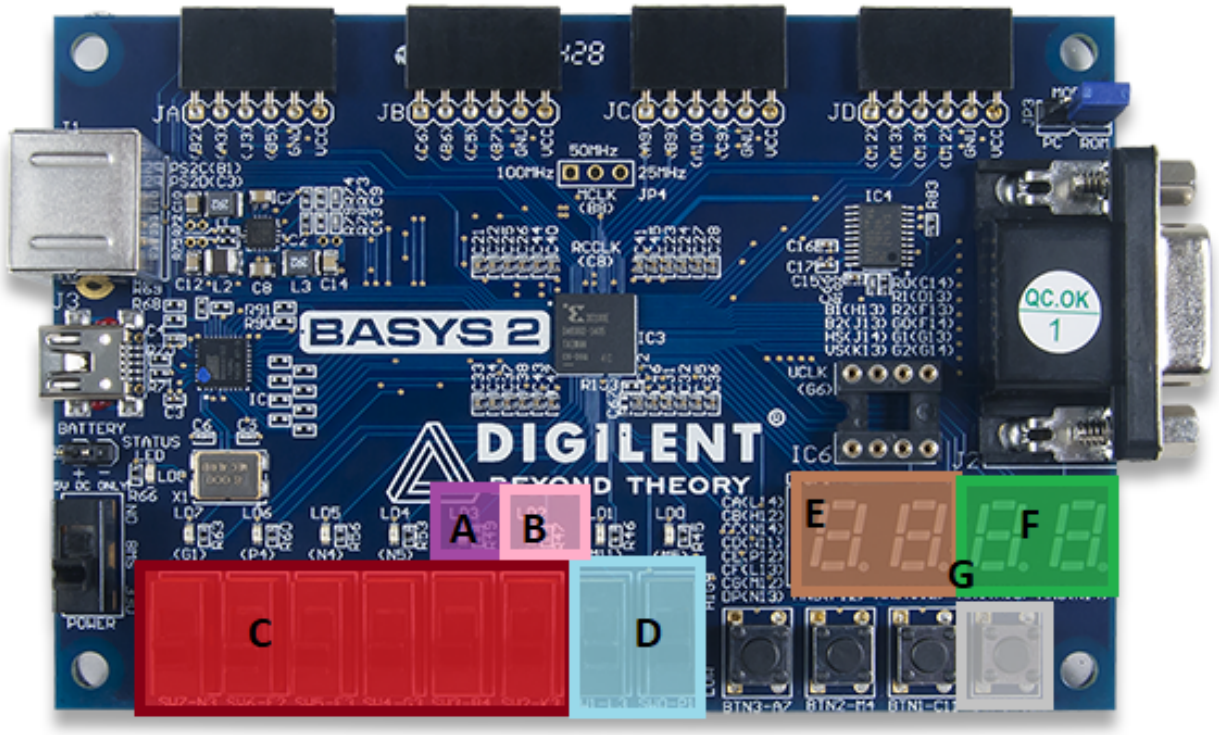

Figure 2: Board figure with labels

Name	FPGA Board	Description	Alphabets
AlreadyInside	led[3]	Fifth LED from the top	A
NotInside	led[2]	Sixth LED from the top	B
userID	sw[7:2]	5 left-most switches	C
mode	sw[1:0]	Middle switches	D
selectedAreaId	7-segment display left 2 digits		E
numberOfInsideUser	7-segment display right 2 digits		F
listOutput	7-segment display 4 digits		G

Table 3: Module I-O to FPGA Board I-O Mappings

Note: The right bottom button is CLK.

2.6 Deliverables

- Implement your module in a single Verilog file. Upload a .zip archive containing your implementations for **lab4_1.v** and **lab4_2.v** files to ODTUClass system.
  Do **NOT** submit your testbenches, bit files or other project files. You can share your testbenches on ODTUClass discussion page.
- Submit the file through the ODTUClass system before the deadline given at the top.
- Use the ODTUClass discussion for any questions regarding the homework.
- This is individual work, any kind of cheating is not allowed.
- Implementations will be tested using custom test benches in a black-box fashion. We will use **Xilinx** and **iverilog** to conduct these tests. Please make sure that your code compiles over these tools.

3 Testbenches & Simulation

You are provided with testbenches to test your implementations in a simulation environment, however, please note that they do not cover all possible cases. That is, passing all test cases provided does not guarantee a full credit. Hence, you are recommended to extend the testbenches provided.

4 Grading Criteria

The grading for this assignment is structured as follows:

1. **Part 1 (30 Points):** Part 1 of the assignment will be evaluated through black box test benches in a simulation environment.
2. **Part 2 (70 Points):** Part 2, which involves the implementation of the Seating System module, will also be assessed using black box test benches in a simulation environment.
3. **Bonus: FPGA Implementation (20 Points):** Originally a core component of the assignment, the FPGA implementation task, is now **bonus** due to incompatibilities with the Xilinx software versions supporting the FPGA boards we have and the dependency issues many face. Completing this task will provide an additional 20 points, making it possible for you to earn up to 120% on this homework.

   Additionally, it is important to note that not being able to test on FPGA boards does not necessarily imply that your code will fail. In fact, if your implementation passes a comprehensive set of simulation test cases, it is likely to function correctly. Therefore, even if you are unable to program the FPGA boards, you can still earn bonus points as long as your code successfully passes the simulation tests.

4.1 Recommended Tools for Simulation

Given the issues with Xilinx installation, the alternative platforms recommended for the simulation tasks:

- **Icarus Verilog (iVerilog)** - Suitable for local simulation.
- EDA Playground - Useful for online simulation. When using EDA Playground, ensure to select Icarus Verilog under the Tools and Simulators tab and update the simulation flag from -g2012 to -g2005.

5 Plagiarism

For any questions or discussions, if they do not contain specific code snippets etc., please use the discussion thread on ODTUClass so that everyone can benefit and be kept up-to-date. In such cases where you need to share specific code/pseudo code (anything not abstract regarding your solution), please do not hesitate to reach out via e-mail. You are encouraged to answer each other's questions and discuss among yourselves provided you do not share any code etc. Such actions of sharing are not to be tolerated. Similarly, submitting someone else's work in part or whole is a direct violation of academic integrity and will be subject to disciplinary action. This also holds for online/AI sources, including Google, ChatGPT, and Copilot. **Make use of them responsibly** - refrain from generating the code you are asked to implement. Remember that we also have access to these tools, making it easier to detect such cases. Your implementations will be subject to similarity checks through advanced tools and those that show high levels of similarity may be considered instances of plagiarism. **In short, please do not resort to practices violating your integrity - we are here to help.**