Middle East Technical University - Department of Computer Engineering

CENG 232

Logic Design

Spring '2020-2021

Lab 4

Part 1 Due Date: 14 June 2021, Monday, 23:55
Part 2 Due Date: 21 June 2021, Monday, 23:55
No late submissions

1 Part 1: Encoded Memory (20 pts)

In this part, you are expected to implement basic memories as 2 Verilog modules. The modules together take a binary number and an operation type (read/write) and then according to the operation type, either calculates and records the XOR encoding of the given number to the given memory index (write mode), or returns the previously XOR-encoded data from the given index of the memory (read mode).

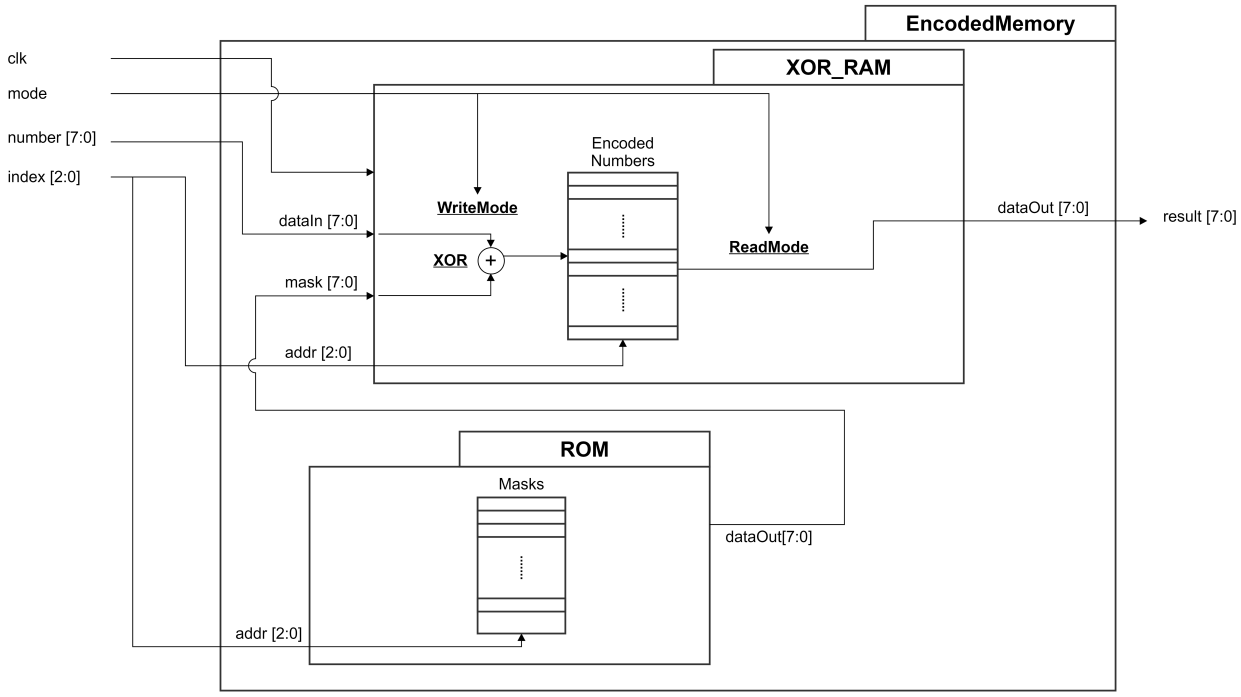


Figure 1: Illustration of the modules.

1.1 ROM Module

The ROM uses 3-bit addresses (**addr**). Each memory location contains 8 bits of data. You should take into consideration the specifications below:

1. This ROM works asynchronously, it is not triggered by a clock pulse.
2. It returns the cell value with the given index (**addr**) as output (**dataOut**) immediately.
3. The values of ROM should be as provided in the following table:

addr	Data Value
0	00000000
1	01010101
2	10101010
3	00110011
4	11001100
5	00001111
6	11110000
7	11111111

Use the following lines as the port definition of the ROM Module:

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

1.2 XOR_RAM Module

The RAM uses 3-bit addresses (**addr**). Each memory location contains 8 bits of data. You should take into consideration the specifications below:

1. Initially, the values of all RAM registers will be 0.
2. There are 2 modes in this module:
   - 0 - write mode
   - 1 - read mode
3. Write mode is synchronous. It is triggered by the positive edge of the clock (CLK).
4. Read mode is asynchronous. That means it is not triggered by a clock pulse.
5. In write mode, an **XOR** operation will be done with **dataIn** and the **mask**. The result ($dataIn \oplus mask$) will be written in **addr** location of the RAM.
6. In read mode, no write operation to the memory is conducted. The value stored in the **addr** location of the RAM will be returned in dataOut.
7. The initial value of **dataOut** should be 0. It can only be changed in read mode. It should retain the last read value when the module is in write mode.

Use the following lines as the port definition of the XOR_RAM Module:

```
module XOR_RAM (
input mode,
input [2:0] addr,
input [7:0] dataIn,
input [7:0] mask,
input CLK,
output reg [7:0] dataOut);
```

1.3 EncodedMemory Module

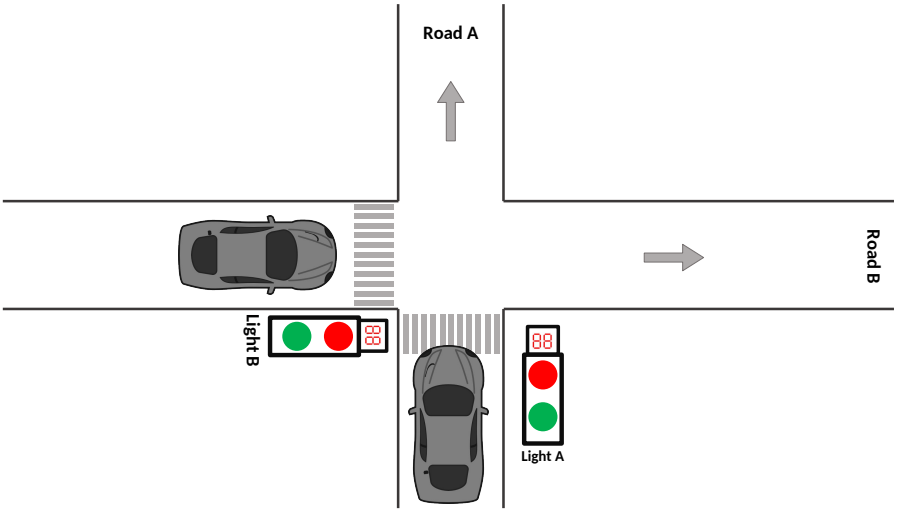
This is the upper module, in which inputs and outputs of other modules are defined. The inputs of this module; mode, CLK, index, number and result are distributed to ROM and XOR_RAM modules. **You should not edit this module.**
Illustration of the overall Encoded Memory system is given in Figure 1.

```
module EncodedMemory (
input mode,
input [2:0] index,
input [7:0] number,
input CLK,
output [7:0] result);
```

1.4 Deliverables

- Implement both modules in a single Verilog file (provided to you as **lab4_1.v**).
  Do **NOT** submit your testbenches. You can share your testbenches on ODTUClass discussion page.
- Submit the file through the ODTUClass system before the deadline given at the top.
- Any kind of cheating is not allowed.

2 Part 2: Intersection Traffic Simulator (80 pts)



In this part, you are expected to implement an intersection traffic simulation system where the duration of the traffic lights adaptively changes in response to the current traffic load on the roads. In this intersection traffic simulation system, there are two roads, *RoadA* and *RoadB*, and two traffic lights, *LightA* and *LightB*. The details of the system is described below:

1. The traffic lights display either red light or green light.
2. The lights turn green after a specified time period.
3. Roads are one lane and one way.
4. Cars move along a straight line, they do not turn left or right.
5. The traffic lights on *RoadA* and *RoadB* operate alternately with one unit of time (1 second) lag. To illustrate, when *LightA* turns red, *LightB* turns green after one unit of time and vice versa.
6. The minimum/maximum duration (in seconds) for a light to stay red depends on whether or not it's rush hour, and is given in Table 1:

Road	Rush Hour		Regular	
	Min	**Max**	**Min**	**Max**
A	30	60	40	70
B	40	70	50	80

Table 1: Min and max waiting times at red lights.

7. **Initially**, you should consider minimum durations for both lights.
8. Rush hours are 7:00:00-8:59:59 AM and 5:00:00-6:59:59 PM (all inclusive).
9. When the light turns red on a road, the system starts counting the number of waiting cars on that road.
10. The maximum number of cars that simultaneously exist in the intersection is 30.
11. The duration of the light to stay red will be dynamic with respect to the following rules. Suppose that *current_turn* represents the red light duration of the road that is about to turn green. **Just before the light turns green**, check the number of cars (C) on that road. If:
    (a) $0 <= C <= 10$, then that road's next red light duration will be $current\_turn + 5$.
    (b) $11 <= C <= 19$, then that road's next red light duration will not be updated.
    (c) $20 <= C$, then that road's next red light duration will be $current\_turn - 5$.

    With this approach, red light will stay longer if the traffic on that road is not that busy. If the traffic on that road is heavy, than the red light duration will be shorter.
12. You should keep the minimum/maximum duration values in mind when a light turns red. Depending on whether it is rush hour or not, the system should not wait longer than the maximum duration value defined in Table 1 (even if the calculated duration is greater), and similarly, the system should not wait shorter than the minimum value (even if the calculated duration is less) at the red light.
13. After the clock proceeds, and the duration becomes zero on a red light, first it triggers the *other* light to turn red; and after one unit of time (as stated in Item 5), the red light turns green.
    Below is an example showing the states of the lights in rush hour.

CLK	Light A	Light B	Remaining Time	Notes
0	Green	Red	1	One unit of time left on red light.
1	Red	Red	0	First, Light A turns red so that no collision occurs.
2	Red	Green	30	Then, Light B turns green.

14. A detector will be present on the intersection which will signal this system with add/remove operations. Using these signals, system will catch and record the cars that passed on the red light.
    - You should record each car plate (*plateIn*) waiting for each road. When a car enters the road (either in red or green light), its plate should be added to the **end of** the list for **that** road (*mode* = 010 or 011).
    - You can assume that there won't be a car with plate 00000. Moreover, each car has a unique plate.
    - You can assume that cars leave the road in "first in first out" manner. When a car leaves a road (*mode* = 000 or 001), you should remove the first car from the front of that road (Hint: ignore *plateIn* while removing cars).
    - Note that the number of **remove** operations signalled by the detector will not be more than the number of **add** operations.
    - The system stores the plates of the cars that violate traffic rules: whenever a car leaves the road (either roadA or roadB) in **red light**, its plate should be added to the blacklist. The blacklisted cars should be added in the order they violated the rule. There is only one blacklist that stores all fined cars.
15. The blacklist should be reset at midnight (12:00:00 AM). Moreover, maximum number of cars that can be held by the blacklist is 30. You can assume that there won't be more than 30 cars in the blacklist.

16. The system starts at 6:00 am. In the **initial** state, the traffic light on *RoadA* is green whereas the traffic light on *RoadB* is red, which implies the remaining time is 50 unit for the light on *RoadB*. And, there are no cars on both *RoadA* and *RoadB*.
17. 1 second in real life corresponds to 1 clock cycle in the system. Therefore, the system will increment its time in each clock cycle.
18. To be more clear, in each clock cycle, the system has three tasks:
    (a) If the system is in display mode (mode=1xx), show the blacklisted cars on *blackListDisplay* continuously at each clock cycle. Do not update the time.
    (b) If the system is not in display mode:
        - Update the time; set or reset the rush hour warning.
        - Decrement the remaining waiting time at the red light. If it hits 0, change the lights accordingly.

2.1 Input/Output Specifications

Name	**Type**	**Size**
mode	Input	3
plateIn	Input	5
action	Input	1
clk	Input	1
greenForA	Output	1
greenForB	Output	1
rushHourWarning	Output	1
hour	Output	4
minute	Output	6
second	Output	6
am_pm	Output	1
numOfCarsA	Output	5
numOfCarsB	Output	5
remainingTime	Output	7
blackListDisplay	Output	5

- **mode** is used for determining whether to apply add/remove operation to the list of a road or display the cars that are in the blacklist.
  - Add Remove Operations: These operations are asynchronous and are **not** triggered by the clock pulse. These are triggered by the positive edge of *action* (Hint: Use *posedge action*).
    * 000 : Remove the first car from the front of *RoadA*.
    * 001 : Remove the first car from the front of *RoadB*.
    * 010 : Add a car with the given *plateIn* to the end of *RoadA*.
    * 011 : Add a car with the given *plateIn* to the end of *RoadB*.
  - Display Operation: This operation is synchronous and is triggered by the clock (clk) pulse (*action* has no affect). **In display mode, clock cycles do not affect the current red light's remaining time, hour, minute, second.**
    * 1xx : Display the plates of the cars that were fined and placed into the blacklist using *blackListDisplay*. This operation is synchronous. You should list one plate per *clk*. Each time this task is activated, the plates should be displayed from the beginning.
      · If the list empty, *blackListDisplay* should show 0.
      · If there is one car in the list, *blackListDisplay* should only show the plate of that car on each cycle.
      · If there is more than one car in the list, blacklisted cars should be listed in the order that they have entered the blacklist. Single car is displayed on each clock cycle. After all cars are displayed, listing starts from the beginning again.
    * 0xx : *blackListDisplay* should continue to show the last displayed plate number until the midnight reset (as stated in Item 15).

- **plateIn** represents the car that enters the road (depending on the **mode**). It is only used in add operations.
- **action** is used to trigger add/remove operations. When it changes from 0 to 1, corresponding action should occur. It has no affect on any other operation.
- **blackListDisplay** is used to display the plate numbers in blacklist when display mode is active (*mode*=1xx).
- **greenForA** shows whether *LightA* is green or not.
- **greenForB** shows whether *LightB* is green or not.
- **rushHourWarning** shows whether it's rush hour or not.
- **hour, minute, second** represent the current time of the day in 12-hour clock.
- **am_pm** shows whether it's AM or PM. Its value is 0 when it is *AM* time and 1 when it is *PM*.

24 Hour Clock	12 Hour Clock			
	Hour	Minute	Second	am_pm
00:00:00	12	0	0	0 (AM)
00:15:23	12	15	23	0 (AM)
01:15:44	1	15	44	0 (AM)
12:00:00	12	0	0	1 (PM)
12:15:11	12	15	11	1 (PM)
13:15:59	1	15	59	1 (PM)

- **numOfCarsA** shows the number of cars waiting on *RoadA*.
- **numOfCarsB** shows the number of cars waiting on *RoadB*.
- **remainingTime** shows the remaining time on the red light.

2.2 Simulation

Due to long timing nature of the system, you probably will use high amount of clock cycles. While running testbenches, Xilinx software initially runs only certain amount of clock cycles on iSim Simulator. In order to run the entire testbench click "**Simulation→Run all**" on iSim simulator.

2.3 Deliverables

- Implement the module in a single Verilog file (provided to you as **lab4_2.v**).
  Do **NOT** submit your testbenches. 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.
- Any kind of cheating is not allowed.