UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 5 Examination in Engineering: November 2024

Module Number: EE5260 **Module Name: Hardware Description Language**

[3 Hours]

[Answer **all questions**, each question carries **ten** marks]

Q1 a) Explain the differences between Hardware Description Languages (HDL) and traditional software programming languages, highlighting their unique features and applications.

[2.0 Marks]

b) Counters are sequential digital circuits commonly used for counting purposes, typically implemented using flip-flops. They are classified into up counters, down counters, or up/down counters. Depending on the required counting range, counters can be designed with various bit widths, such as 2-bit, 3-bit, or more. Counters often include a reset option, allowing the count to restart from the initial position.

i) Draw the state diagram for a 3-bit Up/Down counter.

[2.0 Marks]

ii) Design the Verilog module for a 3-bit Up/Down counter, including a reset option.

[4.0 Marks]

iii) Write the Verilog testbench code to verify the counter implemented in part Q1 b) ii).

[2.0 Marks]

Q2 a) What is an incomplete specification in Verilog and how can it be avoided? Provide examples with relevant code snippets to support your explanation.

[1.5 Mark]

b) A demultiplexer (often abbreviated as Demux) is a digital circuit that takes a single input signal and directs it to one of several output lines, based on the value of select inputs. Furthermore, it performs the reverse operation of a multiplexer, which combines multiple inputs into a single output.

i) Create a Verilog module for a 1 to 4 Demux as shown in figure Q2 b).

[1.5 Marks]

ii) Using the module designed in part Q2 b) i), create a 1 to 8 demultiplexer. (Hint: The select lines and enable lines of each 1 to 4 Demux will serve as select inputs for the final design.)

[5.0 Marks]

iii) Write a Verilog testbench code to verify the functionality of the 1 to 8 demultiplexer designed in part Q2 b) ii).

[2.0 Marks]

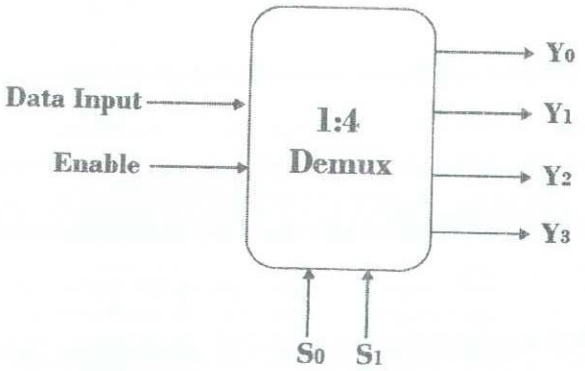

Figure Q2 b): 1 to 4 Demultiplexer.

Q3 a) Explain the difference between synchronous and asynchronous sequential circuits in Verilog, including examples of each.

[2.0 Marks]

b) A Traffic Light Controller is an essential real-time system that manages the timing and order of traffic signals at intersections, ensuring safety and efficiency. This system can be implemented using a Finite State Machine (FSM) to control the signal changes (Green, Yellow, Red) for vehicles traveling in different directions. Consider a basic intersection with two traffic directions: North-South (NS) and East-West (EW).

Requirements:

- The traffic lights should switch from green to yellow to red based on specific timing intervals.
- Only one direction should have a green light at any given time to avoid collisions.
- The lights should follow a continuous pattern that allows each direction to safely pass through the intersection in turn.

Operation:

- The FSM starts with a green light for North-South and a red light for East-West. After 10 clock cycles, the North-South light switches to yellow as a warning, while East-West remains red.
- Following another 10 cycles, the North-South light turns red, and the East-West switches to green.
- After 10 more cycles, the East-West changes to yellow, preparing to stop, and then the North-South light returns to green, repeating the cycle.

i) Assuming the sequence repeats indefinitely and each state lasts for a fixed time interval, fill the state transition table Q3 b i) for this traffic control system.

Table Q3 b i): The state transition table for the traffic control system

Current State	NS Light (North-South)	EW Light (East-West)	Next State
NS_GREEN	Green	Red	NS_YELLOW
NS_YELLOW			
EW_GREEN			
EW_YELLOW			

[1.5 Marks]

ii) Design the Verilog module for the traffic control system based on the transition table in Q3 b i).

[6.5 Marks]

Q4 a) Describe the front-end design process in the Integrated Circuit (IC) design flow, including:

- Specification and Design Entry
- RTL Design and Synthesis

For each stage, explain the objectives, tools used (e.g., Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL), Verilog), and how functional verification is integrated.

[2.0 Marks]

b) i) What are standard cells, and what role do they play in the IC design flow?

[1.0 Marks]

ii) Explain why standard cells are crucial for design automation, especially in the following stages:

- RTL to Gate-level Synthesis
- Physical Design (Placement & Routing)

[2.0 Marks]

c) i) What is a simulation environment, and its functionality in the testing stage, in the context of Field-Programmable Gate Array (FPGA) design process?

[2.0 Marks]

ii) Explain the behavior simulation stage in the FPGA design process. Briefly describe the primary design model used during this stage.

[3.0 Marks]

Q5. a) Explain why serially connected inverting buffers are included in the logic level inputs of commercially available general-purpose logic ICs. Use the reset input of the CD4022 IC, as shown in Figure Q5 a), as an example to support your explanation.

[2.0 Marks]

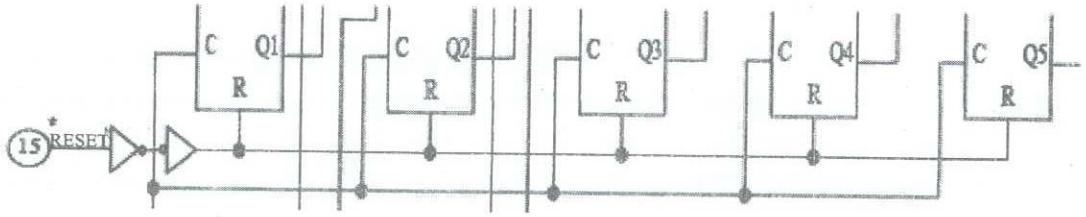

Figure Q5 a): The reset input of the CD4022 IC.

b) Briefly explain the meaning and functionality of the following Verilog code lines.

i) `assign Out = (A == 4'b1111) ? B : 1'b0;`

ii) `B <= 8'b00000001 << A;`

[2.0 Marks]

c) The following code snippets contain errors or bad practices in Verilog. Identify the issues in each snippet, explain them briefly, and provide the corrected version of the code.

i)

```
module test(
    input wire [2:0] select,
    output reg out
);
always @( select) begin
    case (select)
        2'b000: out = 1'b0;
        2'b101: out = 1'b1;
        2'b010: out = 1'b0;
        2'b011: out = 1'b1;
       default: out = 0;
    endcase
end
endmodule
```

ii)

```
module testbench( );
    reg CLOCK;
    dummy_module dut (
        .CLOCK(CLOCK)
    );
    initial begin
        forever #1 CLOCK = ~CLOCK;
    end
endmodule
```

[3.0 Marks]

d) Write a Verilog module named ***Dff*** to model a generic D flip-flop as shown in Figure Q5 d) i). Using this ***Dff*** module, create another Verilog module called ***Div4*** as depicted in Figure Q5 d) ii) that divides the input clock frequency by 4. Implement the frequency division using the ***Dff*** module and ensure the designs adhere to the naming conventions provided in the figures Q5 d) i) and Q5 d) ii).

[3.0 Marks]

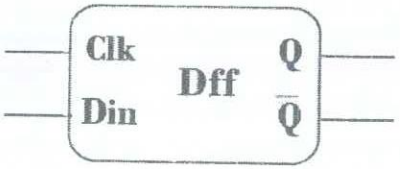

Figure Q5 d) i): The D Flip Flop.

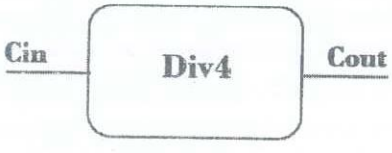

Figure Q5 d) ii): The frequency divider.