ANSWER KEY & SCHEME OF EVALUATION (IN SEM-1)

Name of the Course: Digital VLSI Design   Course Code: 22EC2222

Batch: Y22   A.Y. & SEM: AY 23-24   Department: ECE

1 A) Discuss different modelling styles in Verilog HDL.

Verilog HDL (Hardware Description Language) offers several modeling styles to represent and simulate digital systems. These styles allow designers to describe the hardware at various levels of abstraction, catering to different stages of the design process.

Behavioral Modeling: This style focuses on describing the functionality of the system without specifying its structure. It uses high-level constructs like if, case, while, for loops, and procedural blocks (initial and always). Behavioral modeling is best suited for writing test benches, specifying complex algorithms, and early design exploration.

Dataflow Modeling: In this style, the focus is on the flow of data through the system using continuous assignment statements with operators and expressions. It represents the system as a set of interconnected functional blocks, allowing the description of combinational logic in a concise manner. The assign statement is commonly used in dataflow modeling.

Structural Modeling: This style closely resembles the physical structure of the hardware. It involves the instantiation and interconnection of basic components like gates, flip-flops, and user-defined modules. Structural modeling is useful for representing the design at a low level, providing a clear view of the hardware's interconnected components.

1 B) Identify the keyword used in dataflow modelling in Verilog HDL

The keyword used in dataflow modelling is "always"

1 C) Identify the type of modelling which uses basic primitives in Verilog

Primitives are used in Gate level modelling/Structural modelling

1 D) Discuss different types of PLDs.

Programmable Logic Devices (PLDs) are integrated circuits that can be programmed to perform specific logic functions. They provide a flexible and efficient way to design custom logic circuits without the need for manufacturing new chips. PLDs can be categorized into several types, each offering different levels of complexity, programmability, and functionality.

Simple Programmable Logic Devices (SPLDs): SPLDs are the most basic form of PLDs, consisting of a small number of logic gates and flip-flops. They are typically used for simple logic functions. SPLDs can be further divided into two main types:

Programmable Array Logic (PAL): A PAL has a programmable AND array and a fixed OR array. It is used for implementing combinational logic circuits.

Programmable Logic Array (PLA): Unlike PALs, PLAs have both programmable AND and OR arrays, offering more flexibility in designing complex logic functions.

Complex Programmable Logic Devices (CPLDs): CPLDs contain multiple SPLDs within a single chip, allowing for the implementation of more complex logic functions. They have a predictable timing structure and are suitable for moderate to complex logic circuits. CPLDs are structured around multiple logic blocks or macrocells, which are interconnected through a programmable switch matrix.

Field-Programmable Gate Arrays (FPGAs): FPGAs are the most advanced and flexible type of PLDs. They consist of an array of programmable logic blocks and a hierarchy of reconfigurable interconnects that allow the blocks to be wired together, much like a one-chip programmable digital circuit. FPGAs can implement highly complex digital computations and are used in a wide range of applications, from digital signal processing to software-defined radio and beyond.

1 E) Infer key aspects of design methodology using Verilog HDL.

Design methodology using Verilog HDL emphasizes a structured approach to digital system design, allowing for efficient representation, simulation, and synthesis of hardware. Key aspects include:

Abstraction Levels: Verilog supports design at behavioral, dataflow, and structural levels, facilitating top-down or bottom-up approaches depending on complexity and requirements.

Modularity: Encouraging the use of modules to encapsulate functionality, promoting reusability and simplifying testing.

Simulation and Testbenches: Critical for verifying design correctness before synthesis, using testbenches to simulate various scenarios and monitor outputs.

Synthesis: Transforming Verilog code into hardware, requiring careful consideration of synthesizable constructs to ensure the resulting hardware matches design intentions.

Timing Analysis: Verifying that design meets required timing specifications, crucial for ensuring performance in real-world applications.

1 F) Describe some advantages of design using HDLs

Designing with Hardware Description Languages (HDLs) such as Verilog or VHDL offers several significant advantages in the field of digital system design:

Abstraction: HDLs enable designers to describe complex hardware at various levels of abstraction, from high-level behavioral descriptions to low-level structural details. This flexibility allows for a more manageable and understandable design process, especially for complex systems.

Reusability: HDLs facilitate the creation of modular designs, where individual modules can be tested and verified independently and then reused in other projects. This modularity significantly speeds up the design process and enhances productivity by allowing designers to leverage pre-existing components.

Simulation and Verification: Before any physical prototype is built, HDLs allow the design to be thoroughly simulated and verified against expected outcomes. This capability helps identify and rectify errors early in the design process, reducing development time and costs associated with physical prototyping and debugging.

Portability: Designs described in HDLs are independent of the underlying hardware, making them portable across different technology platforms. A design can be synthesized and implemented on various types of programmable logic devices, such as CPLDs and FPGAs, without changing the high-level description.

Synthesis Optimization: HDLs allow for automated synthesis tools to optimize the design for specific criteria, such as speed, area, or power consumption. This automatic optimization can result in highly efficient hardware that might be difficult to achieve through manual design methods.

Name of the Course: Digital VLSI Design Course Code: 22EC2222

Batch: Y22 A.Y. & SEM: AY 23-24 Department: ECE

2 A) Examine the functionality of the bufif0 gate in Verilog and explain its use in digital circuit modelling.

The bufif0 gate in Verilog HDL is a tristate buffer with an enable control, which functions similarly to a standard buffer but includes an additional control input that determines the output state. The bufif0 gate outputs the input signal when the control signal is low (0) and goes to high impedance (Z) when the control signal is high (1). This behavior is opposite to the bufif1 gate, which drives the output when the control signal is high.

Functionality:

Input: The bufif0 gate has two input ports and one output port. The two inputs are the data input (D) and the control input (E).

Output: The output (Q) mirrors the data input (D) when the control input (E) is low (0). When the control input (E) is high (1), the output (Q) goes to a high impedance state (Z), effectively disconnecting the output from the circuit.

2 B) Explore the application and significance of the ternary operator in Verilog.

The ternary operator in Verilog, often referred to as the conditional operator, is a powerful tool for concise code writing and decision making within hardware description. Its syntax is similar to that found in C and C++ programming languages, and it is expressed as:

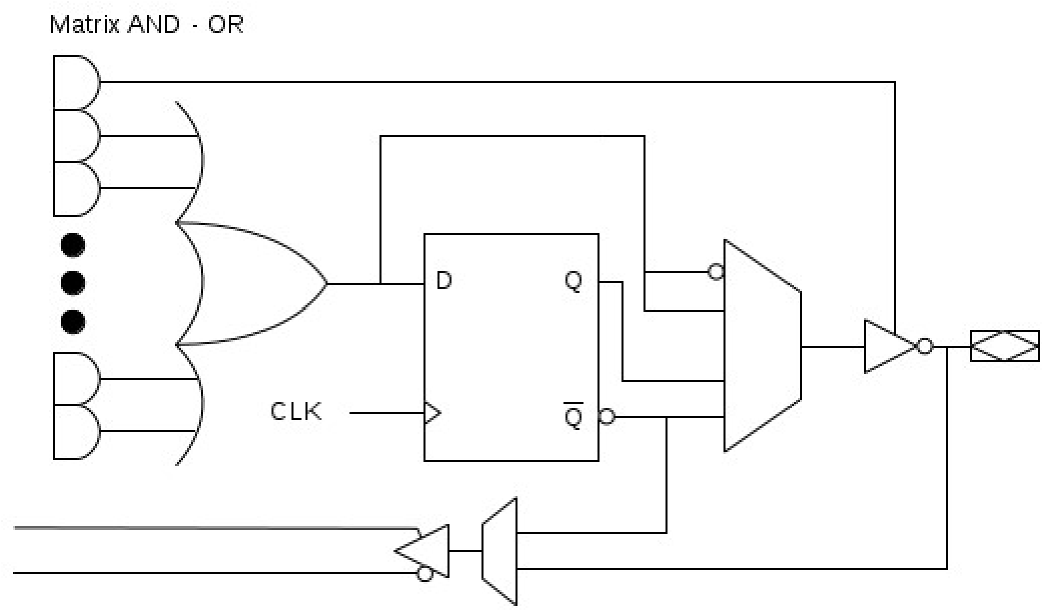
condition ? true_expression : false_expression;

2 D) Demonstrate the working principle of a Microcell in a CPLD.

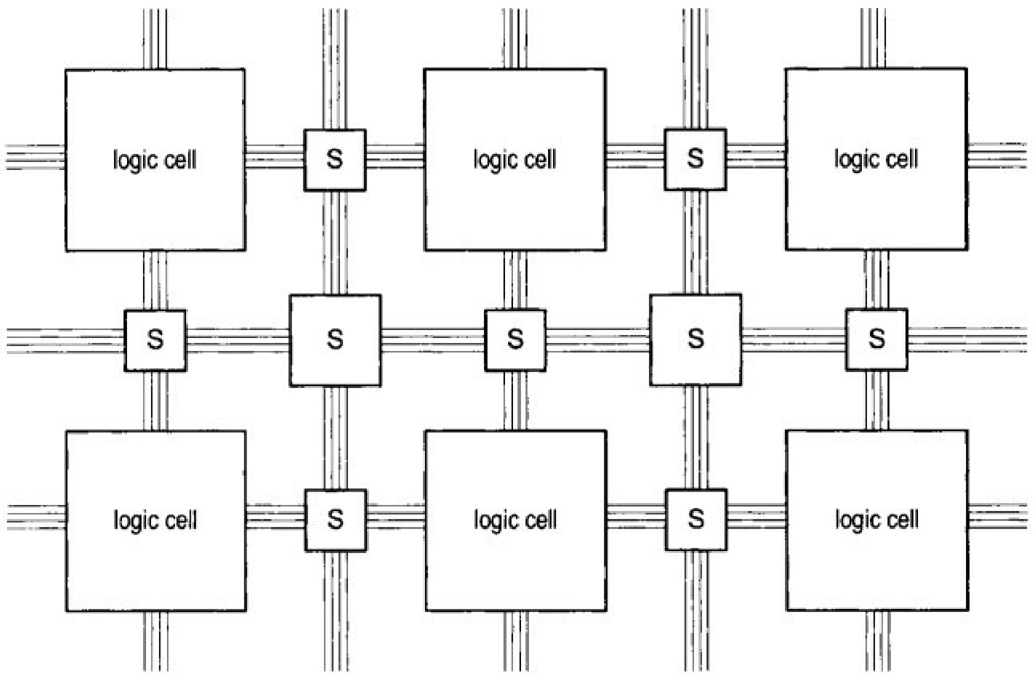
A macrocell array is an approach to the design and manufacture of ASICs. Essentially, it is a small step up from the otherwise similar gate array, but rather than being a prefabricated array of simple logic gates, the macrocell array is a prefabricated array of higher-level logic functions such as flip-flops, ALU functions, registers,

Name of the Course: Digital VLSI Design **Course Code: 22EC2222**

Batch: Y22 **A.Y. & SEM: AY 23-24** **Department: ECE**



2 D) Illustrate the components and arrangement of an FPGA using a block diagram.



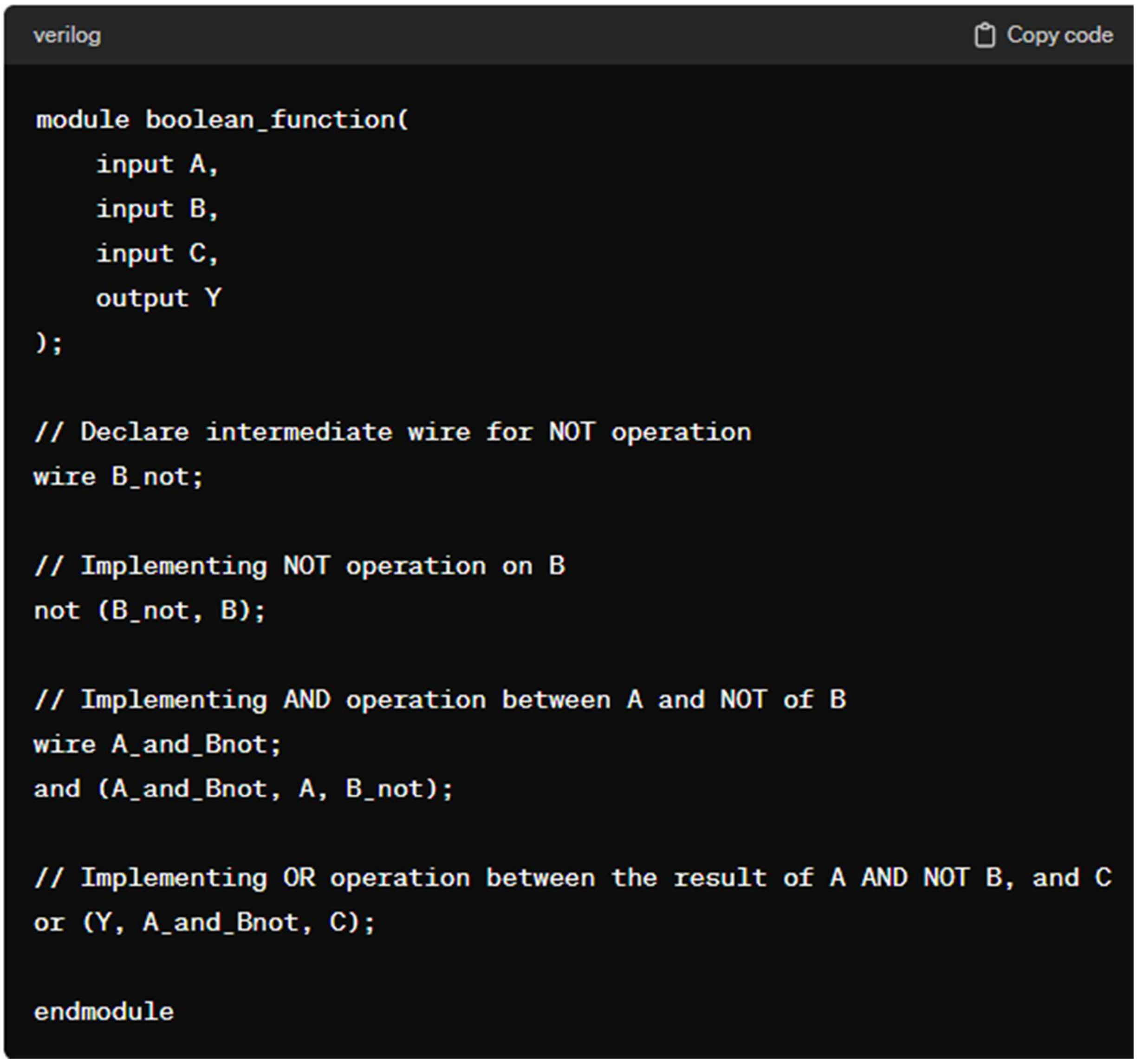
ANSWER KEY & SCHEME OF EVALUATION (IN SEM-1)

Name of the Course: Digital VLSI Design **Course Code: 22EC2222**

Batch: Y22 **A.Y. & SEM: AY 23-24** **Department: ECE**

3 A) Apply gate level modelling to implement the Boolean function,

Y = AB’ + C in Verilog HDL. Write the code in a neat and readable manner



```verilog
module boolean_function(
    input A,
    input B,
    input C,
    output Y
);

// Declare intermediate wire for NOT operation
wire B_not;

// Implementing NOT operation on B
not (B_not, B);

// Implementing AND operation between A and NOT of B
wire A_and_Bnot;
and (A_and_Bnot, A, B_not);

// Implementing OR operation between the result of A AND NOT B, and C
or (Y, A_and_Bnot, C);

endmodule
```

Name of the Course: Digital VLSI Design   Course Code: 22EC2222

Batch: Y22   A.Y. & SEM: AY 23-24   Department: ECE

3 B) Write a testbench using Verilog HDL for the design in the question above.

```
`timescale 1ns / 1ps

module boolean_function_tb;

// Testbench uses reg for inputs and wire for outputs

reg A, B, C;

wire Y;

// Instantiate the Device Under Test (DUT)

boolean_function uut (   .A(A),    .B(B),    .C(C),    .Y(Y));

initial begin    // Initialize Inputs

   A = 0; B = 0; C = 0;

   // Wait 100 ns for global reset to finish

   #100;

        // Apply a sequence of test vectors

   A = 0; B = 0; C = 0;

   #10; // Wait for 10 ns

   A = 0; B = 0; C = 1;

   #10;

   A = 0; B = 1; C = 0;

   #10;

   #10;

   A = 1; B = 1; C = 1;

   #10;

   $finish;

end

endmodule
```

ANSWER KEY & SCHEME OF EVALUATION (<u>IN SEM-1</u>)

Name of the Course: Digital VLSI Design   Course Code: 22EC2222

Batch: Y22   A.Y. & SEM: AY 23-24   Department: ECE

4 A) Imagine you are a digital system designer working on a security system for a smart building. The system involves two key sensors, P and Q, to detect motion and door status, respectively. The alarm should be triggered when either motion is detected, or door is open. Identify the logic of the circuit and write the Verilog HDL for the same.

```
module security_system(

    input P,  // Motion sensor

    input Q,  // Door status sensor

    output Alarm  // Alarm trigger

);



// Implementing the Alarm logic based on sensors P and Q

assign Alarm = P | Q;

endmodule

`timescale 1ns / 1ps

module security_system_tb;

// Declare inputs as reg and output as wire
reg P, Q;
wire Alarm;

// Instantiate the Device Under Test (DUT)
security_system uut (
    .P(P),
    .Q(Q),
    .Alarm(Alarm)
);

initial begin
    // Initialize Inputs
    P = 0; Q = 0;

    // Display header for easier reading of test results
    $display("Time\tP\tQ\tAlarm");
    $display("----------------------------------");
    // Monitor changes on inputs and outputs
    $monitor("%g\t%b\t%b\t%b", $time, P, Q, Alarm);

    // Apply test vectors
    #10 P = 0; Q = 0; // Both sensors are inactive
```

```
  #10 P = 1; Q = 0; // Motion detected, door is closed
  #10 P = 0; Q = 1; // No motion, door is open
  #10 P = 1; Q = 1; // Motion detected and door is open

  // Wait for a moment to observe the last change
  #10;

  // Finish the simulation
  $finish;
end
endmodule
```

5 A) Illustrate the difference between a synchronous and asynchronous sequential circuit

Synchronous and asynchronous sequential circuits are two fundamental types of digital circuits used to implement memory elements and state-based behaviors in digital systems. The key difference between them lies in their timing and control mechanisms for state transitions.

Synchronous Sequential Circuits:

Clock-Driven: Synchronous sequential circuits rely on a global clock signal to control state changes. All state transitions occur in lockstep with the edges of the clock signal (typically on the rising or falling edge).

Predictable Timing: Because changes occur only at specific times dictated by the clock signal, the timing of state transitions is predictable, which simplifies timing analysis and design.

Stability and Noise Immunity: The reliance on a clock edge for state transitions provides a level of immunity to glitches or short pulses on inputs, making these circuits more stable and less prone to erroneous behavior due to noise.

Example Applications: Microprocessors, digital counters, and register-based operations commonly use synchronous sequential circuits.

Asynchronous Sequential Circuits:

Event-Driven: Asynchronous sequential circuits do not use a global clock signal. Instead, state transitions are triggered by changes in input signals. They react immediately to input changes without waiting for a clock signal.

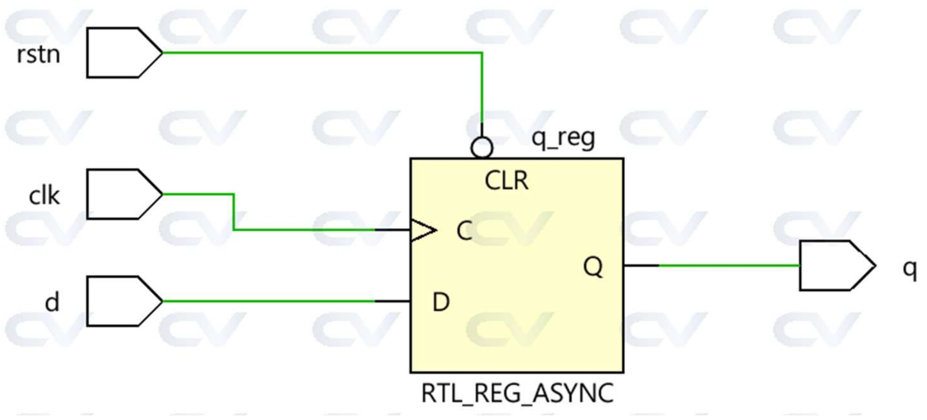
Complex Timing Analysis: The absence of a clock signal can lead to more complex timing analysis due to the potential for race conditions and hazards. Designers must carefully consider propagation delays to ensure reliable operation.

ANSWER KEY & SCHEME OF EVALUATION (IN SEM-1)

Name of the Course: Digital VLSI Design **Course Code: 22EC2222**

Batch: Y22 **A.Y. & SEM: AY 23-24** **Department: ECE**

5 B) You are a digital circuit designer tasked with enhancing the robustness and reliability of a data storage unit in a critical aerospace control system. The data storage unit utilizes flip-flops to retain crucial information related to the system's state. The scenario involves the necessity for asynchronous inputs, particularly a reset functionality, in the flip-flop design. Illustrate the operation of such flip flop with asynchronous reset



6 A) Compute the number of states and number of flip flops employed in a 3-bit counter

A 3-bit counter has 3 bits to represent its state, meaning it can count from 0 to 23−123−1, which is 0 to 7. Therefore, the number of states in a 3-bit counter is 8 (2323).

Since each flip-flop can store one bit of information, the number of flip-flops required to implement a 3-bit counter is exactly 3. Each flip-flop represents one bit of the counter's binary value, allowing the counter to represent any binary number from 000 to 111 (0 to 7 in decimal).

6 B) You are tasked with designing a 3-bit circuit that is going through the following states: 000, 111, 010, 011, 100, 110, 101, 011 and back again. Identify the type of circuit

It is a non-binary counter