DAY 35 -111 DAYS VERIFICATION CHALLENGE

Topic: Procedural blocks

Skill: Verilog

DAY 35 CHALLENGE:

1. What is the purpose of 'initial' & 'always' block in Verilog?

- **'initial' Block:** Used for specifying the initial conditions or setup in a simulation. It executes only once at the beginning of the simulation.
- **'always' Block:** Used for describing the behaviour of the circuit that needs to be evaluated continuously. It can be triggered by changes in signals specified in the sensitivity list or at regular time intervals.

2. Do 'initial' & 'always' block synthesize in Verilog?

- **'initial' Block:** Generally, 'initial' blocks are not synthesizable because they are intended for simulation purposes.
- **'always' Block:** 'always' blocks are synthesizable and are commonly used to describe combinational and sequential logic.

3. Explain 'final' block with an example.

- The 'final' block is used in simulation to specify tasks to be executed at the end of the simulation. It is not synthesizable.

```
final begin
  $display("Simulation ended.");
end
```

4. How can a 'forever' loop be stopped in Verilog?

- A 'forever' loop can be stopped using control signals and conditional statements within the loop.

```
reg stop;
initial stop = 0;

always begin
  forever begin
    // Loop body
    if (stop) begin
      disable loop_label;
    end
```

```
    end
  end
```

5. Can 'initial' & 'always' blocks be used inside a 'task"?

No, 'initial' and 'always' blocks cannot be used inside a 'task'. Tasks are meant to define procedural code that can be called from 'initial' or 'always' blocks.

6. What happens if there is no sensitivity list?

If there is no sensitivity list in an 'always' block, it will not be triggered by any event and will result in a simulation loop if not properly terminated.

```
always begin
  // Code without sensitivity list
end
```

7. What are Blocking and non-blocking in Verilog?

- Blocking Assignment (=): Executes statements sequentially. The next statement starts after the current one completes.
- Non-blocking Assignment (<=): Schedules the assignment to happen at the end of the current time step, allowing parallel execution.

```
reg a, b, c;

// Blocking
a = b;
b = c;

// Non-blocking
a <= b;
b <= c;
```

8. Explain continuous assignment with an example?

- Continuous assignment is used for assigning values to net variables (wire) using the assign keyword.

```
wire a, b, c;
assign c = a & b;
```

9. Design & execute following in Verilog:

I. 8:3 priority encoder using 'case' statement

```
module priority_encoder(
  input [7:0] D,
  output reg [2:0] y);

  always@(D) begin
    casex(D)
```

```
      8'b1xxx_xxxx: y = 3'b111;
      8'b01xx_xxxx: y = 3'b110;
      8'b001x_xxxx: y = 3'b101;
      8'b0001_xxxx: y = 3'b100;
      8'b0000_1xxx: y = 3'b011;
      8'b0000_01xx: y = 3'b010;
      8'b0000_001x: y = 3'b001;
      8'b0000_0001: y = 3'b000;
      default: $display("Invalid data received");
    endcase
  end
endmodule
```

II. D latch

```
module d_latch (
  input D,
  input En,
  output reg Q
);
always @(En or D) begin
  if (En)
    Q = D;
end
endmodule
```

III. Decade counter

```
module decade_counter (
  input clk,
  input reset,
  output reg [3:0] count
);
always @(posedge clk or posedge reset) begin
  if (reset)
    count <= 4'b0000;
  else if (count == 4'b1001)
    count <= 4'b0000;
  else
    count <= count + 1;
end
endmodule
```