

DAY 37 - 111 DAYS VERIFICATION CHALLENGE

Topic: Case statements & Delays

Skill: Verilog

DAY 37 CHALLENGE:

1. Describe following delays with example:

I. Regular Assignment Delay

Regular assignment delay is used in procedural assignments within always or initial blocks. It specifies the delay before the assignment takes effect.

```
module regular_assignment_delay;
    reg a, b;

    initial begin
        a = 1'b0;
        #5 a = 1'b1; // Delay of 5 time units
        #10 b = a;    // Delay of 10 time units
    end
endmodule
```

II. Implicit Continuous Assignment Delay

Implicit continuous assignment delay is specified when assigning values to nets using continuous assignments (assign statements). This delay models the propagation delay through combinational logic.

```
module implicit_continuous_assignment_delay;
    wire a, b, c;

    assign #3 a = b & c; // Delay of 3 time units before 'a' is
                        // assigned the value of 'b & c'

    initial begin
        b = 1'b1;
        c = 1'b0;
        #5 c = 1'b1; // At time 5, 'c' changes to 1
    end
endmodule
```

III. Net Declaration Delay

Net declaration delay is specified directly in the declaration of the net. This delay applies to all assignments to the net, modeling the delay through the net itself.

```
module net_declaration_delay;
    wire #4 a, b; // Delay of 4 time units for 'a' and 'b'

    assign b = 1'b1; // 'b' will become 1 after 4 time units

    initial begin
        #6 a = b; // After 6 time units, 'a' is assigned the value
        of 'b' with an additional delay of 4 time units
    end
endmodule
```

2. What is delta simulation time?

Delta simulation time is a concept used in digital simulation, particularly in event-driven simulators like those used for simulating hardware description languages (HDLs) such as Verilog and VHDL. It refers to the infinitesimally small time steps that the simulator uses to resolve events that occur at the same simulation time.

3. Explain transport delay inertial delay?

Transport Delay

Transport delay represents the time it takes for a signal to propagate through a circuit element, transferring the signal to the output after a fixed delay, regardless of the signal duration. This type of delay does not filter out short pulses or glitches.

Inertial Delay

Inertial delay models the actual behavior of physical devices that do not respond to short pulses or glitches. It filters out pulses shorter than the specified delay. The signal only propagates if it remains stable for at least the duration of the delay.

4. What is meant by inferring latches & how to avoid it?

Inferring Latches: In digital design, latches can be unintentionally inferred if a combinational logic block does not specify all possible output conditions. This usually happens when an if-else or case statement is incomplete, leading to the synthesis of a latch to hold the previous value.

Avoiding Latches:

- Ensure all branches of an if-else statement or case statement are covered.
- Use default in case statements.
- Assign default values at the beginning of the procedural block.

5. Explain the repeat loop.

① "repeat loop."

```
repeat (<expression>)
    sequential - statement;
```

→ The "repeat" construct executes the loop a fixed number of times.
→ it cannot be used to loop on a general logical expression like "while".

→ The expression in the "repeat" construct can be a constant, a variable or a single value.
continuously driven (like clk)

```
repeat (a)
begin
end
```

initial []
always []

→ Initial and always block are not a update every repeat and not value which is 0 or 1.

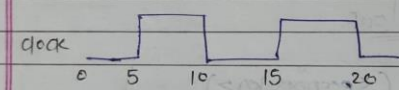
- If it is a variable or a signal value, it is evaluated only when the loop starts and not during execution of the loop.

→ sequential statements can be single or group

Ex: reg clock; // Exactly 100 clk are generate

```
initial
begin
    clock = 1'b0;
    repeat (100)
        #5 clock = ~clock;
    end
```

clock



6. Why is it necessary to list all inputs in the sensitivity list of a combinational circuit?

In a combinational circuit, it is crucial to list all inputs in the sensitivity list to ensure the block is re-evaluated whenever any input changes. This ensures the output always reflects the current state of all inputs, preventing missed updates and ensuring correct functionality.

7. Explain following statements in Verilog:

I. Casex

casex statement allows x (unknown) and z (high impedance) values in both case expression and case items to act as wildcards, meaning they can match any value.

II. Casez

casez statement treats z values in the case expression and case items as wildcards but does not treat x values as wildcards.

8. What is Verilog parallel case and full case statements?

Parallel Case

A parallel case statement indicates that each case item should be mutually exclusive, ensuring only one case item matches. This is typically ensured through synthesis pragmas or directives.

Full Case

A full case statement ensures that all possible values of the case expression are covered. This typically involves including all binary combinations or using a default case to catch unspecified values.

9. Design 4-bit ripple carry adder in Verilog.

```
module full_adder (  
    input a, b, cin,  
    output sum, cout  
);  
    assign sum = a ^ b ^ cin;  
    assign cout = (a & b) | (b & cin) | (cin & a);  
endmodule  
  
module ripple_carry_adder_4bit (  
    input [3:0] a, b,  
    input cin,  
    output [3:0] sum,  
    output cout  
);  
    wire [2:0] carry;  
  
    full_adder fa0 (a[0], b[0], cin, sum[0], carry[0]);  
    full_adder fa1 (a[1], b[1], carry[0], sum[1], carry[1]);  
    full_adder fa2 (a[2], b[2], carry[1], sum[2], carry[2]);  
    full_adder fa3 (a[3], b[3], carry[2], sum[3], carry[3]);  
endmodule
```