DAY 32 - 111 DAYS VERIFICATION CHALLENGE

Topic: Verilog data types

Skill: Verilog

DAY 32 CHALLENGE:

1. Explain the difference between 'reg' & 'wire' with an example?

wire

- Represents a continuous connection.
- Used for combinational logic.
- Cannot store values; they reflect the current state of their drivers.
- Often used to connect different modules or for the output of combinational logic.

reg

- Represents storage elements.
- Used for sequential logic or to hold values across clock cycles.
- Can store values assigned in procedural blocks (always, initial).

```
Module reg_map_to_wire(A, B, C, f2, f1);

      Input A, B, C ;

      Output f1, f2 ;

      wire  A, B, C ;

      reg f1, f2 ;

always @(A or B or C)

begin

      f1 = ~(A & B);

      f2 = f1 ^ C ;

end

endmodule
```

2. Explain ternary operator (?) with an example.

The ternary operator in Verilog (like in many other programming languages) is a shorthand for an if-else statement. It takes three operands: a condition, a result if the condition is true, and a result if the condition is false. The general syntax is:

```
condition ? true_expression : false_expression
```

Example

```
module mux2to1 (

    input a,

    input b,

    input sel,

    output y

);

    assign y = (sel) ? b : a;

endmodule
```

3. Identify the base formats of nos. in following declarations:

I. **12345 (Hint: Since nothing is specified it will take default no. format. Tell me what's that default format)** ➔ decimal

II. **'o25** ➔ octal

III. **'h9** ➔ hexadecimal

IV. **'b1** ➔ binary

4. What is the value of following declarations: (Hint: If something is 'x' you can calculate the range or possible values, if some bit is high impedance, calculate the value as per other bits & say this particular bit will be high impedance)

I. **1'bz** ➔ high impedance

II. **32'h1x792** ➔ range is 32'b10792 to 32'h11792

III. **16'o4z** ➔ this is an 16 bit octal number and LSB digit are high impedance

IV. **16'h2xz** ➔ this is also a 16 bit hexadecimal number with LSB bit high impedance and range is 16'h20z to 16'h20z.

5. Explain how negative nos. can be declared & stored in Verilog.

In Verilog, negative numbers can be declared and stored using two's complement representation. Two's complement is a method for encoding signed numbers where the most significant bit (MSB) indicates the sign of the number: 0 for positive and 1 for negative.

6. Explain the following keywords in Verilog with an example:

I. Time

The time keyword in Verilog is used to declare variables that store simulation time values. The time type is a 64-bit unsigned integer.

```
module time_example;
  time current_time;

  initial begin
    #5;
    current_time = $time;
    $display("Current simulation time: %0t", current_time);
  end
endmodule
```

II. Integer

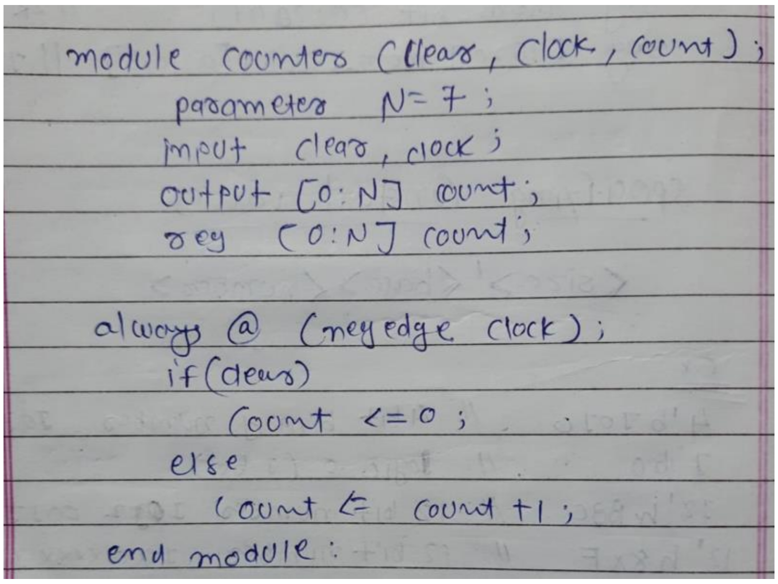
The integer keyword is used to declare variables that store integer values. Integers in Verilog are 32-bit signed values by default.

```
module integer_example;
  integer count;

  initial begin
    count = -10;
    $display("Count: %d", count);
    count = count + 20;
    $display("Updated Count: %d", count);
  end
endmodule
```

III. Parameter

The parameter keyword is used to define constants that can be used to parameterize modules. Parameters make modules more flexible and reusable.



```
module counter (clear, clock, count);
      parameter  N= 7 ;
      input  clear, clock;
      output [0:N] count;
      reg   [0:N] count;

   always @ (negedge clock);
      if(clear)
          count <= 0 ;
      else
          count <= count +1 ;
   end module :
```

IV. Defparam

The defparam keyword is used to override the default parameter values of a module after it has been defined. It is considered an older style compared to direct parameter passing.

```
module defparam_example #(parameter WIDTH = 8) (
  input [WIDTH-1:0] in,
  output [WIDTH-1:0] out
);
  assign out = in;
endmodule

module test;
  reg [7:0] data;
  wire [7:0] result;

  defparam_example my_instance (
    .in(data),
    .out(result)
  );
  defparam my_instance.WIDTH = 16;

  initial begin
    data = 8'hA5;
    #1;
    $display("Result: %h", result);
  end
endmodule
```