# DAY 63 - 111 DAYS VERIFICATION CHALLENGE

Topic: SV Data types

Skill: System Verilog

DAY 63 CHALLENGE:

## 1. Explain *enum* with an example

enum (enumeration) in SystemVerilog is a user-defined data type that allows the definition of a set of named values, which are often used to represent states or modes in a finite state machine (FSM), or other scenarios where a variable can take on a predefined set of values.

```
typedef enum logic [1:0] {
    IDLE = 2'b00,
    START = 2'b01,
    RUNNING = 2'b10,
    STOP = 2'b11
} state_t;
state_t current_state;
initial begin
    current_state = START;
    if (current_state == START) begin
        $display("System is starting");
    end
end
```

In this example, state_t is an enumerated type that defines four states: IDLE, START, RUNNING, and STOP. The current_state variable is of type state_t, and it can only hold one of the four enumerated values.

## 2. What are different functions in System Verilog for an enum?

SystemVerilog provides several built-in methods for working with enum types:

- **first():** Returns the first enumerated value.
- **last():** Returns the last enumerated value.
- **next(value):** Returns the next enumerated value after the current value. Returns the first value if the current value is the last.

- **prev(value):** Returns the previous enumerated value before the current value. Returns the last value if the current value is the first.
- **num():** Returns the total number of enumerated values.

```
initial begin
    state_t next_state;

    next_state = current_state.next; // Move to the next state
    $display("Next state is: %0d", next_state);

    next_state = current_state.prev; // Move to the previous state
    $display("Previous state is: %0d", next_state);

    $display("First state: %0d", state_t::first());
    $display("Last state: %0d", state_t::last());
    $display("Total number of states: %0d", state_t::num());
end
```

## 3. What is a String in System Verilog?

In SystemVerilog, a string is a variable-sized array of characters. Strings are used to store and manipulate sequences of characters and can be declared using the string keyword.

```
string my_string;
initial begin
    my_string = "Hello, SystemVerilog!";
    $display("My string is: %s", my_string);
end
```

## 4. What are different functions in System Verilog for String?

SystemVerilog provides several built-in functions for manipulating strings:

- **len():** Returns the length of the string.
- **toupper():** Converts the string to uppercase.
- **tolower():** Converts the string to lowercase.
- **compare():** Compares two strings lexicographically.
- **substr(start, length):** Returns a substring starting at start position with the specified length.

- **itoa():** Converts an integer to a string.
- **atoi():** Converts a string to an integer.

```
string my_string = "SystemVerilog";
string new_string;
int result;
initial begin
    $display("String length: %0d", my_string.len());

    new_string = my_string.toupper();
    $display("Uppercase string: %s", new_string);

    result = my_string.compare("SystemVerilog");
    $display("Comparison result: %0d", result);

    new_string = my_string.substr(0, 6);
    $display("Substring: %s", new_string);

    int num = 12345;
    new_string = $sformatf("%0d", num);
    $display("Integer to string: %s", new_string);

    int converted_num = new_string.atoi();
    $display("String to integer: %0d", converted_num);
end
```

## 5. What is the difference between time & realtimef.

**time:**

- time is a SystemVerilog data type used to represent simulation time. It is a 64-bit unsigned integer, which means it can store time values as discrete, whole numbers.
- Use Case: Suitable for storing and manipulating simulation time in ticks.

**realtime:**

- realtime is a floating-point data type that represents time with decimal precision.

- Use Case: Useful when you need to store and manipulate time with fractional values (e.g., 1.5 ns) or when precise time calculations are required.

```
time t1;
realtime rt1;
initial begin
  t1 = 50;
  rt1 = 50.25;
  $display("Time: %0t", t1);
  $display("Real time: %0.2f", rt1);
end
```

## 6. How to write floating point & exponential numbers in System Verilog? Explain with an example?

SystemVerilog allows you to write floating-point numbers using the standard decimal notation or exponential notation.

- Floating-point numbers: These are numbers with a decimal point.
  - Example: 3.14, 0.001, -15.0.
- Exponential notation: It is used to represent very large or very small numbers conveniently.
  - Example: 1.23e3 (which equals 1230), 4.56e-2 (which equals 0.0456).

```
real pi = 3.14159;
real small_number = 2.5e-10;
real large_number = 6.02e23;

initial begin
    $display("Pi: %0.5f", pi);
    $display("Small number: %0.10f", small_number);
    $display("Large number: %0.2e", large_number);
End
```

## 7. Explain void data type with an example.

The void data type in SystemVerilog is used to specify that a function does not return any value. It is similar to the void type in languages like C and C++. This is useful when a function is intended to perform an action but not return any value to the caller.

```
void print_message(string msg);
    $display("Message: %s", msg);
endfunction
initial begin
    print_message("Hello, SystemVerilog!");
end
```

## 8. How can I convert a real data type to an int?

To convert a real data type to an int in SystemVerilog, you can use the **$rtoi()** system function, which truncates the fractional part of the real number.

```
real real_num = 3.75;
int int_num;
initial begin
    int_num = $rtoi(real_num);
    $display("Real number: %f, Integer number: %0d", real_num, int_num);
end
```

## 9. What is the use of Event data type in SV?

The event data type in SystemVerilog is used to trigger and synchronize processes. An event is an object that can be triggered by the -> operator, and processes can wait for the event to be triggered using the @ operator.

```
event my_event;

initial begin
    #5 -> my_event; // Trigger the event after 5 time units
End

initial begin
    @my_event; // Wait for the event to be triggered
    $display("Event triggered at time %0t", $time);
end
```

## 10. What is the use of parameter data type in SV?

The parameter data type in SystemVerilog is used to define constants that can be used throughout a module. Parameters can be overridden at the time of instantiation, making them useful for creating reusable modules with configurable properties.

```
module my_module #(parameter WIDTH = 8) (
    input logic [WIDTH-1:0] data_in,
    output logic [WIDTH-1:0] data_out
);
    assign data_out = data_in;
endmodule


module top;
    logic [15:0] data;
    logic [15:0] result;
    my_module #(16) u1 (.data_in(data), .data_out(result));
    initial begin
        data = 16'hABCD;
        #1;
        $display("Result: %h", result);
    end
endmodule
```

## 11. Create a struct packet with following fields:

### 32-bit address

### 64-bit data

### 1 bit valid

You can define a struct in SystemVerilog to group together multiple related fields into a single composite data type. Below is an example of a struct named packet with a 32-bit address, 64-bit data, and a 1-bit valid field.

```
typedef struct {
    logic [31:0] address;
    logic [63:0] data;
    logic valid;
} packet_t;
```

```
packet_t my_packet;


initial begin
    my_packet.address = 32'hDEADBEEF;
    my_packet.data = 64'h123456789ABCDEF0;
    my_packet.valid = 1'b1;


    $display("Packet - Address: %h, Data: %h, Valid: %b",
my_packet.address, my_packet.data, my_packet.valid);
end
```