Module #03 : Data Types

3.1. Value Set:

- Verilog HDL consists only four basic values
- Almost all Verilog data types store all these values
- Values are –

A) 0 (Logic Zero or False Condition)
B) 1 (Logic One or True Condition)
C) X (Unknown Logic Value)
D) Z (High Impedance State)

Module #03 : Data Types

3.2. Wire:

- Represents a physical wire in a circuit and is used to connect gates or modules.
- A value of a wire can be read but not assigned to, in a procedural block or in a function
- A wire does not store its value
- Must be driven by a continuous assignment statement

Syntax:

```
wire wire_variable_scaler;
wire [MSB : LSB] wire_variable_vector;
```

Example 3.1:

```
wire a;     //simple 1 wire
wire b;
wire c;
wire [7:0] data; // A cable of 8 wires
assign c = a & b;
```

Module #03 : Data Types

3.3. Reg:

- Declare type reg for all data objects on the left hand side of expressions in procedural blocks (initial and always) and functions.
- Reg data type must be used for latches, flip-flops and memories

Syntax:

```
reg reg_variable_1_bit;
reg [MSB : LSB] reg_variable_vector;
```

Example 3.2:

```
reg a;  // simple 1 bit reg variable
reg b;
reg [7:0] data;
```

Module #03 : Data Types

3.4. Input, Output, Inout:

- These keywords declare input, output and bidirectional ports of a module or task.
- Input and inout ports are of type wire
- An output port can be configured to be of type wire or reg. The default type is wire

Syntax:

```
input simple_port;
input [MSB : LSB] input_port_list;
output [MSB : LSB] output_port_list;
inout [MSB : LSB] inout_port_list;
```

Example 3.3:

```
module sample(a, b, c, d);
 input a;  // input defaults to wire
 input b;
 output c;
 output [2:0] d; /*two bit output, declare its output as reg */
  reg [2:0] d;
```

Module #03 : Data Types

3.5. Integer:

- Integers are general purpose variables.
- They are mainly used for loop indices, parameters, and constants
- Implicitly they are type of reg but they store data as signed numbers however explicitly declared reg types store data as unsigned
- Default size is 32 bits
- If they hold constants, during synthesis, synthesizer adjusts them to the minimum width needed at compilation

Syntax:

integer integer_variable;

Example 3.4:

integer a; //single 32 bit integer
assign b= 63; // 63 defaults to a 7 bit variable

Module #03 : Data Types

3.6. Supply0, Supply1:

- Supply0 and Supply1 define wires tied to Logic 0 (Ground) and Logic 1 (Power), respectively.

Syntax:

```
supply0 logic_0_wire;
Supply1 logic_1_wire;
```

Example 3.5:

```
supply0 gnd_wire;
supply1 pwr_wire;
```

Module #03 : Data Types

3.7. Time:

- A 64-bits quantity that can be used in conjunction with the $time system task to hold simulation time
- Time is not used for synthesis (Not Synthesizable) and hence only used for simulation

Syntax:

time time_variable;

Example 3.6:

time current_simulation_time;
current_simulation_time = $time;

Module #03 : Data Types

3.8. Parameter:

- Parameters allows constants like word length to be defined symbolically in one place. This makes it easy to change the word length later, by changing only the parameter

Syntax:

```
parameter par1 = value1;
parameter par2 = value2,
               par3 = value3, . . . ;
parameter [range] param4 = value4;
```

Example 3.7:

```
parameter NUM_OF_BITS = 8;
parameter ADD = 2'b00, SUB = 2'b01;
Parameter [2:0] Last_State = 3'b111;
```