Basic Concepts in VerilogHDL

Arrays

- Arrays are multiple elements that are 1-bit or n-bits wide.
- Multi-dimensional arrays can also be declared with any number of dimensions.

```
integer count[0:7]; // An array of 8 count variables

reg data[31:0]; // Array of 32 one-bit data register variables

integer matrix[4:0][0:255]; // 2D array of integers

reg [4:0] port_id[0:7]; // Array of 8 port_ids; each of 5 bits wide
```

Memories

- Memories are modeled in Verilog as a one-dimensional array of registers
- Each element of the array is known as an element or word and is addressed by a single array index.

```
reg mem1bit[0:1023];
```
// Memory mem1bit with 1K 1-bit words

```
reg [7:0] membyte[0:1023];
```
// Memory membyte with 1K 8-bit words

```
membyte[511]
```
// Fetches 1 byte word whose address is 511.

Parameters

- Constants are defined in a module by the keyword parameter.
- Parameters cannot be used as variables.

```
parameter port_id = 5; // Defines a constant port_id
parameter cache_line_width = 256;
```

- The `localparam` keyword is used to define parameters when their values should not be changed

```
localparam state1 = 4'b0001,state2 = 4'b0010;
```

System Tasks

- Verilog provides standard system tasks for certain routine operations.
- All system tasks appear in the form $<keyword>
- Operations such as displaying on the screen, monitoring values of nets, stopping, and finishing are done by system tasks.

Displaying information

- $display is the main system task for displaying values of variables or strings or expressions.

```
$display(p1, p2, p3,....., pn);
```

- Examples

```
//Display the string in quotes
$display("Hello Verilog World");
-- Hello Verilog World


reg [4:0] port_id;
$display("ID of the port is %b", port_id);
-- ID of the port is 00101
```

Monitoring information

- `$monitor` - monitor a signal when its value changes

```
$monitor(p1, p2, p3,....., pn);
```

- continuously monitors the values of the variables or signals and displays all parameters in the list whenever the value of any one variable or signal changes
- Examples

```
initial
begin
$monitor("Value of clock=%b reset = %b",clock,reset);
end
```

```
-- Value of signals clock = 0 reset = 1
-- Value of signals clock = 1 reset = 1
```

Stopping and finishing Simulation

- `$stop` is provided to stop during a simulation, puts the simulation in an interactive mode

- `$finish` task terminates the simulation

- Examples

```
initial
begin
clock = 0;
reset = 1;
#100 $stop; // suspend the simulation at time = 100
#900 $finish; // terminate the simulation at time = 1000
end
```

Compiler Directives

- `` `timescale <reference_time_unit> / <time_precision> ``
- <reference_time_unit> specifies the unit of measurement for delays
- <time_precision> specifies the precision to which the delays are rounded off during simulation

```
//Reference time unit is 100 nanoseconds and precision
  is 1 ns
`timescale 100 ns / 1 ns
```

Tasks and Functions