Module #16 : Verilog HDL System Tasks and Functions

16.1: System Tasks and Functions:

- These are tasks and functions that are used to generate input and output during simulation.
- Their names begin with a dollar sign ($).
- Verilog contains the pre-defined system tasks and functions, including tasks for creating output from a simulation.
- Operations such as displaying the screen, monitoring values of nets, stopping and finishing are done by system tasks.

Examples : $display, $time, $finish etc

Module #16 : Verilog HDL System Tasks and Functions

16.2: $display, $strobe, $monitor:

- Display Selected Variables
- These commands have the same syntax, and display text on the screen during simulation.
- $display and $strobe display once every time they are executed, whereas $monitor displays every time one of its parameters changes.
- The difference between $display and $strobe is that $strobe displays the parameters at the very end of the current simulation time.
- Format characters include %d (decimal), %h (hexadecimal), %b (binary), %c (character), %s (string) and %t (time), %m (hierarchy level).

Module #16 : Verilog HDL System Tasks and Functions

Syntax:

$display (“format_string”, par_1, par_2, ...);
$strobe (“format_string”, par_1, par_2, ...);
$monitor (“format_string”, par_1, par_2, ...);
$displayb (as above but defaults to binary..
$strobeh (as above but defaults to hex..
$monitoro (as above but defaults to octal..

Example 16 .1:

```
initial begin // c below is in submodule submod1.
$displayh (b, d, submod1.c); //No format, display in hex.
$monitor (“time=%t, d=%h, c=%b”,$time, a, submod1.c);
end


initial
      #1 a=1; b=0;
       $fstrobe(hand1, a, b);
         b=1;
end
                ;will write 1 1 for a and b.
```

Module #16 : Verilog HDL System Tasks and Functions

16.3: $time, $stime, $realtime:

- These return the current simulation time as a 64-bit integer, a 32-bit unsigned integer, and a real number, respectively.
- If the current simulation time is too large and the value does not fit in 32 bits, the $stime function only returns the 32 low order bits of the value.

Example 15.2:

```
integer cur_time ;
cur_time = $time ;
integer cur_time ;
cur_time = $stime ;
real cur_time ;
cur_time = $realtime ;
```

Module #16 : Verilog HDL System Tasks and Functions

16.4: $reset, $stop(n), $finish(n):

- $reset resets the simulation back to time 0.
- $stop halts the simulator and puts it in the interactive mode where the user can enter commands.
- $finish exits the simulator back to the operating system.

Note: Remember that **$finish** control system task makes the simulator exit, however **$stop** simply suspends simulation.

- By default the argument is 1

Argument (n)	Description
0	No Messages
1	Simulation Time and Location
2	Simulation Time, Location, Memory Consumption and CPU time used in Simulation

Module #16 : Verilog HDL System Tasks and Functions

Example 16.3:

$stop *; Suspend simulation and print message (default argument = 1)*

#150 $finish(2) *; Exits simulator after 150 time units from the last executed statement and prints message (argument == 2).*

Module #16 : Verilog HDL System Tasks and Functions

16.5: $deposit:

- $deposit sets a net to a particular value, overwriting what was put there by the “circuit”. Good for test benches.

Syntax:

$deposit (net_name, value);

Example 15.4:

$deposit (b, 1’b0);

$deposit (outp, 4’b001x);// outp is a 4-bit bus

Module #16 : Verilog HDL System Tasks and Functions

16.6: $random :

- $random generates a random integer every time it is called. If the sequence is to be repeatable, the first time one invokes random give it a numerical argument (a seed). Otherwise the seed is derived from the computer clock.

Syntax:

*runny = **$random**;*
*xyzzy = **$random(integer);***
integer seeds the generator
*randy=**$random%integer;***
integer sets the upper limit of the range of the random integers.

Example 15.4:

```
reg [3:0] xyz;
initial begin
 xyz= $random (7); // Seed the generator so number
// sequence will repeat if simulation is restarted.
forever xyz = #20 $random;
// The 4 lsb bits of the random integers will transfer into the
// xyz. Thus xyz will be a random integer 0 ≤ xyz ≤ 15.
```

Module #16 : Verilog HDL System Tasks and Functions

16.7: $dumpfile, $dumpvar, $dumpon, $dumpoff, $dumpall:

- These can dump variable changes to a simulation viewer.
- The dump files are capable of dumping all the variables in a simulation.
- This is convenient for debugging, but can be very slow.

Module #16 : Verilog HDL System Tasks and Functions

Syntax:

$dumpfile*(“filename.dmp”)*

$dumpvar *dumps all variables in the design.*

$dumpvar(1, top) *dumps all the variables in module top.*

$dumpvar(2, top) *dumps all the variables in module top and 1 level below.*

$dumpvar(n, top) *dumps all the variables in module top and n-1 levels below.*

$dumpvar(0, top) *dumps all the variables in module top and all level below.*

$dumpon *initiates the dump.*

$dumpoff *stop dumping.*

Example 16.5:

```
module testbench:
 reg a, b; wire c;
initial begin;
$dumpfile(“cwave_data.dmp”);
$dumpvar //Dump all the variables
// Alternately instead of $dumpvar, one could use
$dumpvar(1, top) //Dump variables in the top module.
// Ready to turn on the dump.
$dumpon
a=1; b=0;
topmodule top(a, b, c);
end
```

Module #16 : Verilog HDL System Tasks and Functions

16.8: $fopen, $fdisplay, $fstrobe, $fmonitor and $fwrite:

These commands write more selectively to files.

- $fopen opens an output file and gives the open file a handle for use by the other commands.
- $fclose closes the file and lets other programs access it.
- $fdisplay and $fwrite write formatted data to a file whenever they are executed. They are the same except $fdisplay inserts a new line after every execution and $write does not.
- $strobe also writes to a file when executed, but it waits until all other operations in the timestep are complete before writing.
- Thus initial

```
#1 a=1; b=0;
 $fstrobe(hand1, a, b);
  b=1;
end
```

;will write write 1 1 for a and b.

- $monitor writes to a file whenever any one of its arguments changes

Module #16 : Verilog HDL System Tasks and Functions

Syntax:

handle1=*$fopen("filenam1.suffix")*
handle2=*$fopen("filenam2.suffix")*
$fstrobe(handle1, format, variable list) *//strobe data into filenam1.suffix*
$fdisplay((handle2, format, variable list) *//write data into filenam2.suffix*
$fwrite((handle2, format, variable list) *//write data into filenam2.suffix all on*
// one line. Put \n in the format string
// where a new line is desired.

Module #16 : Verilog HDL System Tasks and Functions

Example 16.5:

```
module testbench:
 reg [15:0]a; reg clk; integer hand1;
initial begin;
hand1=$fopen("datastuff.txt");
 forever @(posedge clk) begin
 $fstrobe (hand1, "time=%5t, a=%h, c=%b",
$time, a, submod1.c);.
 end // Never put statements after a forever block.
 end
 initial begin
 clk=0; a=8'h2b;
 forever #5 clk=~clk;
 end // Never put statements after a forever block
initial begin
 a=a+8;
 #3000 $fclose (hand1); // Close the file
 $finish;
 end
submod submod1(a, clk); // with internal variable c.
endmodule
------------------------------ Output ----------------------
time= 5, a=2b, c=0
time= 10, a=2c, c=1
```