Module #08 : Verilog HDL Timing Controls

8.1: Delay Control (Not synthesizable):

- Also known as Inter Assignment Delay
- This specifies the delay time units before a statement is executed during simulation.
- A delay time of zero can also be specified to force the statement to the end of the list of statements to be evaluated at the current simulation time.

Syntax:

***#delay** statement;*

Example 8 .1:

***#5** a = b + c; // evaluated and assigned after 5 time units*

***#0** a = b + c; // very last statement to be evaluated*

Module #08 : Verilog HDL Timing Controls

8.2: Intra-Assignment Delay (Not synthesizable):

- This delay #Δ is placed after the equal sign.
- The left-hand assignment is delayed by the specified time units, but the right-hand side of the assignment is evaluated before the delay instead of after the delay.
- This is important when a variable may be changed in a concurrent procedure.

Syntax:

*variable = #Δ**t** expression;*

Example 8.2:

```
assign a=1; assign b=0;
always @(posedge clk)
b = #5 a; // a = b after 5 time units.
always @(posedge clk)
c = #5 b; /* b was grabbed in this parallel procedure before
the first procedure changed it. */
```

Module #08 : Verilog HDL Timing Controls

8.3: Wait Statement (Not synthesizable):

- Delay executing the statement(s) following the wait until the specified condition evaluates to true

Syntax:

wait (condition_expression) *statement;*

Example 8.3:

wait (!c) a = b; // wait until c=0, then assign b to a

Module #08 : Verilog HDL Timing Controls

8.4: Event Control, @:

- This causes a ***statement*** or ***begin-end*** block to be executed only after specified events occur.
- An event is a change in a variable and the change may be: a positive edge, a negative edge, or either (a level change), and is specified by the keyword *posedge*, *negedge*, or no keyword respectively.

Note:

- For synthesis one cannot combine level and edge changes in the same list.
- For flip-flop and register synthesis the standard list contains only a clock and an optional reset.
- For synthesis to give combinational logic, the list must specify only level changes and must contain all the variables appearing in the right-hand-side of statements in the block.

Syntax:

*@ (**posedge** variable or*
***negedge** variable)*
statement;
*@ (**variable or variable . . .**)*
statement;

Example 8.4:

always @(posedge clk or negedge rst)
if (rst) Q=0; else Q=D; // Definition for a D flip-flop.
@(a or b or e); // re-evaluate if a or b or e changes.
sum = a + b + e; // Will synthesize to a combinational adder