Module #09 : Verilog HDL Procedures

9.1: Always Block:

- All always blocks in a module execute simultaneously unlike conventional programming languages, in which all statements execute sequentially.
- The always block can be used to imply latches, flip-flops or combinational logic.
- All the statements enclosed within begin ... end, executes sequentially.
- The always block is triggered to execute by the level, positive edge or negative edge of one or more signals (separate signals by the keyword or).
- Procedures can be named. In simulation one can disable named blocks. For synthesis it is mainly used as a comment.

Syntax 1:

always *@(event_1 or event_2 or ...)*
begin
... statements ...
end

Syntax 2:

always *@(event_1 or event_2 or ...)*
begin*: name_for_block*
... statements ...
end

Module #09 : Verilog HDL Procedures

Example 9.1:

always @(*a or b) // level-triggered; if a or b changes levels*

always @*(posedge clk); // edge-triggered: on +ve edge of clk*

Module #09 : Verilog HDL Procedures

9.2: Initial Block (Not Synthesizable):

- The initial block is like the always block except that it is executed only once at the beginning of the simulation.
- It is typically used to initialize variables and specify signal waveforms during simulation.
- Initial blocks are not supported for synthesis.

Syntax:

```
initial
begin
... statements ...
end
```

Example 9.2:

```
initial
begin
    clr = 0; // variables initialized at
    clk = 1; // beginning of the simulation
end
inital // specify simulation waveforms
begin
a = 2'b00; // at time = 0, a = 00
#50 a = 2'b01; // at time = 50, a = 01
#50 a = 2'b10; // at time = 100, a = 10
end
```