Verilog Tutorial-3

BLOCKING AND NON-BLOCKING

Recap...

• The left-hand side of a procedural assignment statement can be one of:

a) A register type variable (“reg”, “integer”, “real”, or “time”)

b) A bit select of these variables (e.g. sum[15])

c) A part select of these variables (e.g. IR[31:26])

d) A concatenation of any of the above

• The right-hand side can be any expression consisting of “net” and “register” type variables that evaluates to a value.

***• Procedural assignment statements can only appear within procedural blocks (“initial” or “always”).

Blocking-Assignment

- General syntax:

  variable_name = [delay_or_event_control] expression;

- The “=“ operator is used to specify blocking assignment.

- Blocking assignment statements are executed in the order they are specified in a procedural block.
  - – The target of an assignments gets updated before the next sequential statement in the procedural block is executed.
  - – They do not block execution of statements in other procedural blocks.

- This is the recommended style for modeling combinational logic.

Example

```
integer x, y, z;

initial

 begin

   x = 5; y = 10; z = 20; \\ statement 1

   x = y + z; \\statement 2

   y = x -z;  \\statement 3

   z = x +y; \\statement 4

 end
```

Statement 1: primary initialization
x=5 y=10 z=20
Statement 2 : x=30 y=10 z=20
Statement 3 : x=30 y= 10 z=20
Statement 4 : x=30 y= 10 z= 40

(ii) Non-Blocking Assignment

• General syntax:

variable_name <= [delay_or_event_control] expression;

• The “<=“ operator is used to specify non-blocking assignment.

• Non-blocking assignment statements allow scheduling of assignments without blocking execuDon of statements that follow within the procedural block.

- – The assignment to the target gets scheduled for the end of the simulaDon cycle (at the end of the procedural block).
- – Statements subsequent to the instrucDon under consideraDon are not blocked by the assignment. –
- Allows concurrent procedural assignment, suitable for sequential logic.
- PRIMARIY recommended for sequential circuit with common clock.

Example

```
integer a, b, c;
 initial
  begin
   a = 10; b = 20; c = 15;
  end
 initial
  begin
   a <= #5 b + c;
   b <= #5 a + 5;
   c <= #5 a – b;
  end
```

- Initially, a=10, b=20, c=15
- a becomes 35 at time = 5
- b becomes 15 at time = 5
- c becomes -10 at time = 5

Thumb rule to keep in mind , Evaluate all the right-hand expression and then assign the values to the variable.

Importance of non-blocking assignment:- Swapping two variables

```
always @(posedge clk)
  a = b;
  b = a;
```

```
always @(posedge clk)
 a <= b;
 b <= a;
```

Topics Covered

- Writing Module
- Writing basic testbench
- Structural Modelling
- Behavioural Modelling
- Procedural Assignment – initial block , always block , loop statements
- Blocking Assignment
- Non-Blocking Assignment

These are the most important constructs in the verilog, anything further will either be the application of the above concepts or different representation of it.

Topics left

- Modelling Finite state machine
- Writing Controller Module
- Concept of synthesizable verilog codes

If Required

- Switch Modelling
- Modelling memory/registers

Thank You