SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECB302–VLSI DESIGN

III YEAR/ V SEMESTER

UNIT 5-SPECIFICATION USING VERILOG HDL

TOPIC 7 & 8 –PROCEDURAL ASSIGNMENTS AND CONDITIONAL STATEMENTS

OUTLINE

- TWO PROCEDURAL CONSTRUCTS
- PROCEDURAL ASSIGNMENTS
- BLOCKING & NON BLOCKING - PROCEDURAL ASSIGNMENT
- HIGH-LEVEL PROGRAMMING LANGUAGE CONSTRUCTS
- LOOP STATEMENTS
- ACTIVITY
- CONDITIONAL STATEMENTS
- DATA TYPES
- DECISION-MAKING CONTROLS
- CONTINUOUS ASSIGNMENT-EXAMPLE
- SUMMARY

TWO PROCEDURAL CONSTRUCTS

- – initial Statement
- – always Statement
- initial Statement : Executes only once
- always Statement : Executes in a loop
- Example:

```
...
initial begin
  Sum = 0;
  Carry = 0;
end
...
```

```
...
always @(A or B) begin
  Sum = A ^ B;
  Carry = A & B;
end
...
```

PROCEDURAL ASSIGNMENTS

- Assignments made within procedural blocks are known as procedural assignments.
- The left-hand side of procedural assignment must be a data type in the **register** class.

Example

```
initial begin
out=0;
#10 en1=~net23;
#5 set=(r1|en1)&net4;
end
```

BLOCKING& NON-BLOCKING PROCEDURAL ASSIGNMENT

- **Blocking procedural assignment.**

```
rega = #100 regb;
rega = @(posedge clk) regb;
```

- **Non-Blocking procedural assignment.**

```
rega <= #100 regb;
rega <= @(posedge clk) regb;
```

- **Schedule the assignment without blocking the procedural flow.**
- **Simulators perform two steps when encounter an non-blocking procedural assignment statement.**
  - Evaluate the RHS immediately.
  - Schedule the assignment at a proper time.

BLOCKING PROCEDURAL ASSIGNMENT

```
initail begin
   a = #10 1;
   $display("current time = %t a = %b", $time, a);   → evaluate at time = 10, a = 1
end
```

Evaluate RHS (RHS = 1)

```
initail begin
   a <= #10 1;
   $display("current time = %t a = %b", $time, a);   →   evaluate at time = 0, a = x
end
```

6

NON BLOCKING PROCEDURAL ASSIGNMENT

EVENT CONTROL

- Event Control
  - Edge Triggered Event Control
  - Level Triggered Event Control

- Edge Triggered Event Control

  @ (posedge CLK) //Positive Edge of CLK

  Curr_State = Next_state;

- Level Triggered Event Control

  @ (A or B) //change in values of A or B

  Out = A & B;

@ negedge	@ posedge
$1 \rightarrow x$	$0 \rightarrow x$
$1 \rightarrow z$	$0 \rightarrow z$
$1 \rightarrow 0$	$0 \rightarrow 1$
$x \rightarrow 0$	$x \rightarrow 1$
$z \rightarrow 0$	$z \rightarrow 1$

HIGH-LEVEL PROGRAMMING LANGUAGE CONSTRUCTS

◆**forever** loop

example

```
forever #100 clk=~clk;
always #100 clk=~clk;
```

◆**repeat** loop

example

```
repeat(mem_depth)  begin
   mem[address]=0;
   address=address+1;  end
```

◆while loop

example

```
while(val[index]==1'b0)  index=index-1;
```

◆for loop

example

```
for(index=0;index<size;  index=index+1)
if(val[index]==1'bx)
$display("found an x");
```

ACTIVITY

GROUP DISCUSSION

LOOP STATEMENTS

Loop Statements

1. Repeat
2. While
3. For

Repeat Loop

Example:

```
repeat (Count)
 sum = sum + 5;
```

If condition is a x or z it is treated as 0camb

- **While Loop**
  - Example:

```
while (Count < 10) begin
 sum = sum + 5;
 Count = Count +1;
end
```

  - If condition is a x or z it is treated as 0

- **For Loop**
  - Example:

```
for (Count = 0; Count < 10; Count = Count + 1)
 begin
 sum = sum + 5;
end
```

CONDITIONAL STATEMENTS

- if Statement
- Format:

```
if (condition)
   procedural_statement
else if (condition)
   procedural_statement
else
   procedural_statement
```

- Example:

```
if (Clk)
   Q = 0;
else
   Q = D;
```

CONDITIONAL STATEMENTS (CONT.)

- Case Statement
- Example 1:

```
case (X)
  2'b00: Y = A + B;
  2'b01: Y = A – B;
  2'b10: Y = A / B;
endcase
```

- Example 2:

```
case (3'b101 << 2)
  3'b100: A = B + C;
  4'b0100: A = B – C;
  5'b10100: A = B / C; //This statement is
executed
endcase
```

CONDITIONAL STATEMENTS (CONT.)

- Variants of case Statements:
  - casex and casez

- casez – z is considered as a don’t care

- casex – both x and z are considered as don’t cares

- Example:

```
casez (X)
  2’b1z: A = B + C;
  2’b11: A = B / C;
endcase
```

DATA TYPES

Net Types: Physical Connection between structural elements

Register Type: Represents an abstract storage element.

Default Values

Net Types : z

Register Type : x

Net Types: wire, tri, wor, trior, wand, triand, supply0, supply1

Register Types : reg, integer, time, real, realtime

DATA TYPES(CONT.)

- Net Type: Wire

  wire [msb : lsb] wire1, wire2, …

  – Example

    wire Reset; // A 1-bit wire

    wire [6:0] Clear; // A 7-bit wire

- Register Type: Reg

  reg [msb : lsb] reg1, reg2, …

  – Example

  reg [3: 0] cla; // A 4-bit register

  reg cla; // A 1-bit register

- **Restrictions on Data Types**

- **Data Flow and Structural Modeling**
  – Can use only ***wire*** data type
  – Cannot use ***reg*** data type

- **Behavioral Modeling**
  – Can use only ***reg*** data type (within initial and always constructs)
  – Cannot use ***wire*** data type

DECISION-MAKING CONTROLS

if statement

example

if (set = = 1) out = 1;

if (clear = = 0) q = 0; else q = d;

case statement

example

case(instruction)

2'b00: out = a + b; 2'b01: out = a - b; default:

out=0;

endcase

CONTINUOUS ASSIGNMENT

- Continuous assignment provide a means to abstractly model combinational hardware driving values onto nets. An alternate version of the 1-bit full adder is shown blow:

```
module FA(Cout, Sum, a, b, Cin);  output         Cout, Sum;
input a, b, Cin;
assign    Sum = a ^ b ^ Cin,
Cout = (a & b) | (b & Cin) | (a & Cin);
endmodule
```

- Logic loop of Continuous Assignment

```
assign a = b+a;
```

ASSESSMENT

1.initial Statement : Executes ------------

2.always Statement : Executes -----------

3. List out Restrictions on Data Types

4.Write VERILOG HDL code for 1-bit full adder using Continuous assignment

SUMMARY & THANK YOU