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

16EC303–VLSI DESIGN

III YEAR/ V SEMESTER

UNIT 5-SPECIFICATION USING VERILOG HDL

TOPIC 5 & 6 –OPERATORS AND TIMING CONTROLS

OUTLINE

- INTRODUCTION-HISTORY
- OPERATORS USED IN VERILOG
  - PRECEDENCE RULES FOR OPERATORS
  - OPERATORS USED IN VERILOG (CONT.)
  - EQUALITY AND IDENTITY OPERATORS
  - OPERATORS
  - LITERAL NUMBERS
  - ACTIVITY
  - PROCEDURAL TIMING CONTROLS
  - INTRA-ASSIGNMENT TIMING CONTROL
  - VERILOG-HDL STRUCTURAL LANGUAGE
- SUMMARY

OPERATORS USED IN VERILOG

- VERILOG LANGUAGE OPERATORS

Arithmetic Operators	+, -, *, /, %
Relational Operators	<, <=, >, >=
Equality Operators	==, !=, ===, !==
Logical Operators	!, &&, ||
Bit-Wise Operators	~, &, |, ^, ~^
Unary Reduction	&, ~&, |, ~|, ^, ~^
Shift Operators	>>, <<
Conditional Operators	?:
Concatenations	{}

VERILOG HDL

- Precedence Rules for Operators

Operator Precedence Rules	
! ~ * / % + - << >> < <= > >= == != === !== & ^ ^~ | && || ?: (ternary operator)	highest precedence  lowest precedence

OPERATORS USED IN VERILOG (CONT.)

- The Relational Operators Defined

Relational Operators	
a < b	a less than b
a > b	a greater than b
a <= b	a less than or equal to b
a >= b	a greater than or equal to b

- The Equality Operators Defined

Equality Operators	
a ===b	a equal to b, including x and z
a !==b	a not equal to b, including x and z
a ==b	a equal to b, result may be unknown
a != b	a not equal to b, result may be unknown

EQUALITY AND IDENTITY OPERATORS

- equality operator

==	0	1	x	z
0	1	0	x	x
1	0	1	x	x
x	x	x	x	x
z	x	x	x	x

```
a = 2'b0x
b = 2'b0x

if (a == b)
  $display("a is equal to b");
else
  $display("a is not equal to b");
```

result : a is not equal to b

- identity operator

===	0	1	x	z
0	1	0	0	0
1	0	1	0	0
x	0	0	1	0
z	0	0	0	1

```
a = 2'b0x
b = 2'b0x

if (a === b)
  $display("a is identity to b");
else
  $display("a is not identity to b");
```

result : a is identity to b

OPERATORS

- ***Operators***
  - **Unary Operator** assign a = ~b;
  - **Binary Operator** assign a = b&c**;**
  - **Ternary Operator** assign out = sel ? a: b; //2-to-1 multiplexer
- ***Comments***
  - **One Line Comment**

// this is an example of one line comment

  - **Multiple Line Comment**

/* this is an example of multiple line comment */

  - **Error Comment Remarks**

/* Error comment remark */ */

OPERATORS USED IN VERILOG

- **Index:**
  - example: a[11:6], b[2], ...
- **Concatenation:** {n{<exp> <, <exp>>*}} adder4

  a1(sum,carry,{a[2],a[2:0]},b[3:0]); assign {carry, sum} = a+b+ci;

sign = {4{in[3]}, in}; temp = 2'b01;

out = {2{2'b10}, 2'b11, temp}; //out=8'b1010_1101

A B

- **Arithmetic operation:** +,-,*
  - example: a=b+c;

x=y*z;

- **Condition:** = =, !=, >, <, >=, <=, ...
  - example: assign b = (a = = 0) ;

LITERAL NUMBERS

- Literal integers are interpreted as decimal numbers in the machine word size (32 bits) by default.
- Size and base my be explicitly specified
- **<size>'<base><value>**
  - <size>: size in bits as a decimal number.
  - <base>: b(binary), o(octal), h(hex), d(decimal).
  - <value>: 0-9, a-f, x, z, ? (must be legal number in <base>)
  - Four types of logic value

    0 (logical 0), **1** (logical 1), **x** (unknown), **z** (high impedence)

LITERAL NUMBERS (CONT.)

- ***Examples***
- 12
- 8'd45
- 10'hF1
- 1'B1
- 32'bz
- 6'b001_010

- 32-bit decimal
- 8-bit decimal
- 10-bit hex (left-extended with zero)
- 1-bit binary
- 32-bit Z
- 6-bit binary with underscore for readability.

Underscores are ignored.

X and Z values are automatically extended.

A question mark ? in <value> is interpreted as a Z.

ACTIVITY

DEBATE

PROCEDURAL TIMING CONTROLS

- **Three Types of Timing Controls**

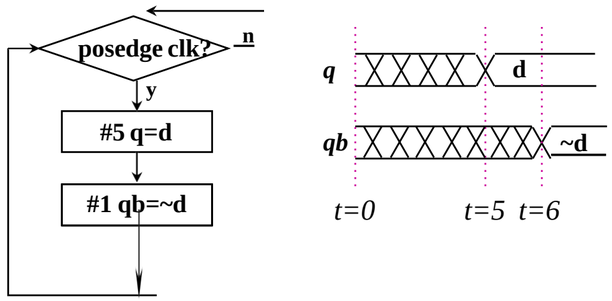
#<delay> : Simple delay.

@(<signal>) : Event control with edge-trigger and level-sensitive controls. wait(<expr>) : Level-sensitive control.

- **Edge-Trigger Control**

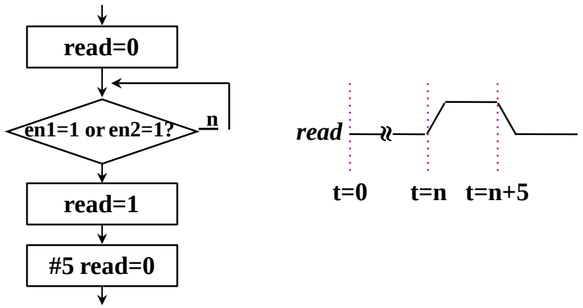
posedge: positive edge. EX: always @(posedge clk) negedge: negative edge. EX: always @(negedge clk)

- **Examples**

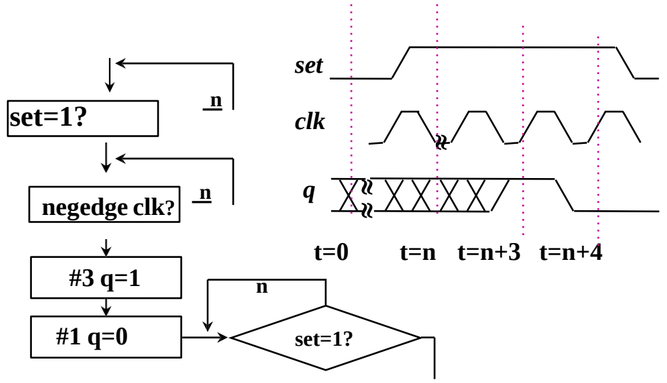


PROCEDURAL TIMING CONTROLS

```
initial  begin  read=0;
wait(en1|en2) read=1;
#5 read=0;
end
```




```
always wait(set)
begin
@(negedge clk);  #3
q=1;
#1 q=0;
wait(!set);  end
```



INTRA-ASSIGNMENT TIMING CONTROL

- **Previously described timing control.**

#100 clk = ~ clk; @(posedge clock) q = d;

- **Intra-assignment timing control.**

clk = #100 ~ clk;

q = @(posedge clock) d;

- **Simulators perform two steps when encounter an intra assignment timing control statement.**
  - Evaluate the RHS immediately.
  - Execute the assignment after a proper delay.

VERILOG-HDL STRUCTURAL LANGUAGE

Intra-Assignment Timing Control

- Intra-assignment timing control can be accomplished by using the following constructs

With intra-assignment construct	With intra-assignment construct

a = #10 b;	begin
	temp = b; #10 a = temp;
	end

a = @(posedge clk) b;	begin
	temp = b;
	@ (posedge clk) a = temp; end

a = repeat(3)@(posedge clk) b;	begin
	temp = b;
	@ (posedge clk) @ (posedge clk)
	@ (posedge clk) a = temp; end

15

ASSESSMENT

1.List out the Operators with precedence

2. **< >'< > < >** - fill up Literal numbers syntax

3. Tell me the literal numbers

- 12
- 8'd45
- 10'hF1
- 1'B1
- 32'bz
- 6'b001_010

4. #<?????> : Simple delay.

@(<??????>)

SUMMARY & THANK YOU