

# Digital System Design with HDL (I)

## Lecture 3

Dr. Ming Xu

Dept of Electrical & Electronic Engineering

XJTLU

1

## In This Session

- Verilog Operators

### Arithmetic Operators

+	$m + n$	Add n to m
-	$m - n$	Subtract n from m
-	$-m$	Negate m (2's complement)
*	$m * n$	Multiply m by n
/	$m / n$	Divide m by n
%	$m \% n$	Modulus of m / n. The result takes the sign of the first operand, e.g.

$5 \% 2 = 1$     $5 \% -2 = 1$     $-5 \% 2 = -1$     $-5 \% -2 = -1$

2

## Operators

### Bitwise Operators

~	$\sim m$	Invert each bit of m (1's complement)
&	$m \& n$	AND each bit of m with each bit of n
	$m   n$	OR each bit of m with each bit of n
^	$m \wedge n$	Exclusive OR each bit of m with n
~^	$m \sim \wedge n$	Exclusive NOR each bit of m with n
^~	$m \wedge \sim n$	

3

## Operators

~	
0	1
1	0
x	X
z	X

&	0	1	x	z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

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

^	0	1	x	z
0	0	1	x	x
1	1	0	x	x
x	x	x	x	x
z	x	x	x	x

4

## Operators

### Logical Operators (1-bit T/F result)

**!** **!m** Is m false?  
**&&** **m && n** Are both m and n true?  
**||** **m || n** Is either m or n true?

5

## Operators

### Reduction Operators (1-bit T/F result): calculated by recursively applying bit-wise operation on all bits of the operand

**&** **&m** AND all bits in m together  
**~&** **~&m** NAND all bits in m together  
**|** **|m** OR all bits in m together  
**~|** **~|m** NOR all bits in m together  
**^** **^m** Exclusive OR all bits in m  
**~^** **~^m** Exclusive NOR all bits in m  
**^^** **^^m** Exclusive NOR all bits in m

6

## Operators

### Reduction Operators

value	&	~&		~	^	~^
4'b0000	0	1	0	1	0	1
4'b0001	0	1	1	0	1	0
4'b0011	0	1	1	0	0	1
4'b0111	0	1	1	0	1	0
4'b1111	1	0	1	0	0	1
4'b01xx	0	1	1	0	x	x
4'b01z0	0	1	1	0	x	x

7

## Operators

### Relational Operators (1-bit T/F result)

**<** **m < n** Is m less than n?  
**>** **m > n** Is m greater than n?  
**<=** **m <= n** Is m less than or equal to n?  
**>=** **m >= n** Is m greater than or equal to n?  
**==** **m == n** Is m equal to n?  
**!=** **m != n** Is m not equal to n?  
**===** **m === n** Is m identical to n?  
**!==** **m !== n** Is m not identical to n?

8

## Operators

### Equality Operators vs. Identity Operators

- Equality operators (== and !=) compare logic values of 0 and 1.
- They produce an x, If either operand contains an x or z.
- Identity operators (=== and !==) compare logic values of 0, 1, X and Z. Bits with x and z must match for the result to be true.
- They produce either 0 or 1.

9

## Operators

### Equality Operators vs. Identity Operators

Examples:

```
reg [3:0] a, b;  
a = 4'b1100; b = 4'b101x;  
  
a == 4'b1z10 // false - 0  
a != 4'b100x // true - 1  
b == 4'b101x // unknown - x  
b != 4'b101x // unknown - x  
b === 4'b101x // true - 1  
b !== 4'b101x // false - 0
```

10

## Operators

### Logical Shift Operators

```
<< m << n    Shift m left n-times  
>> m >> n    Shift m right n-times
```

The vacant bits are filled with 0.

### Arithmetic Shift Operators

```
<<< m <<< n    Shift m left n-times, the vacant bits  
                  are filled with 0.  
  
>>> m >>> n    Shift m right n-times, the vacant bits  
                  are filled with the leftmost bit (the  
                  sign bit for a signed integer).
```

11

## Operators

### Shift Operators

Example:

```
// X=4'b1100;  
Y = X >> 1;           // Y is 4'b0110  
Y = X << 2;           // Y is 4'b0000  
X << 1'bz = 4'bxxxx  
X >> 1'bx = 4'bxxxx  
integer a, b, c;      // signed data types  
a = -8;               // a = 11...11000  
b = a >>> 3;          // b = 11...11111 , b = -1 decimal  
c = a <<< 2;          // b = 11...1100000, b = -32 decimal
```

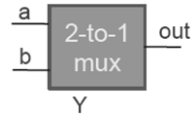
12

## Operators

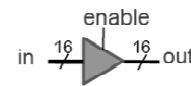
### Conditional Operator

**?: sel?m:n** If sel is true, select m; else select n

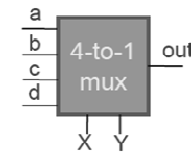
// model functionality of a 2-to-1 mux  
assign out = Y ? b : a;



// model functionality of a tri-state buffer  
assign out = enable ? in : 16'bz;



// nested conditional operators. 4-to-1 mux  
assign out = X ? ( Y ? d : c ) : ( Y ? b : a );



13

## Operators

### Concatenation Operator

**{ } {m,n}** Concatenate m to n, creating larger vector

The operands must be sized.

// A = 1'b1, B = 2'b00, C = 2'b10, D = 3'b110

Y = { B, C }; // Y is 4b'0010

Y = { A, B, C, D }; // Y is 8b'10010110

Y = { B[0], D[2], 2'b11 }; // Y is 4b'0111

14

## Operators

### Replication Operator

**{ } {n{m}}** Replicate m n-times

A = 1'b1; B = 2'b00; C = 2'b10; D = 3'b110;

Y = { 4{A} }; // Results in Y = 4'b1111

Y = { 4{A}, 2{B} }; // Results in Y = 8'b11110000

Y = { 4{A}, 2{B}, C }; // Results in Y = 10'b1111000010

15

## Operators

### Precedence of Verilog operators

Operator type	Operator symbols	Precedence
Complement	! ~ -	Highest precedence
Arithmetic	* / + -	
Shift	<< >>	
Relational	< <= > >=	
Equality	== !=	
Reduction	& ~& & ~&   ~	
Logical	&&	
Conditional	?:	Lowest precedence

16