Operators

Operators

- The operator symbols are similar to those in C language.
- With these operators we can carry out specified operations on the operands and assign the results to a net or a vector set of nets as the case may be.

Operator types

The operators can be :

- arithmetic
- logical
- relational
- equality
- bit wise
- reduction
- shift
- concatenation
- replication
- conditional

Arithmetic operators

Expressions constitute operators and operands.

operation	symbol	operand
Multiply	*	binary
Divide	/	binary
Add	+	binary
Subtract	-	binary
Modulus	%	binary

Arithmetic operator examples

Syntax:

```
a * b // multiply a and b
 a / b  // divide  a by b
a+b    // add a and b
a - b   // subtract b from a
a%b  //  modulus of a by b
```

Example: 2

```
a=3’b011    b=3’b010   d=4   e=3
c=a * b   //  c= 3’b110
c= a / b  //  c= 1
c= a+b   //   c= 3’b101
c= a-b    //    c=3’b001
c=d/e     //  c=1
```

Arithmetic operators

13 % 4   // evaluates to 1.

-9 % 2 // evaluates to -1, takes sign of the first operand

- In arithmetic operations, if any operand bit has a value x, then the result of the entire expression is x.

- The size of the result is determined by the size of the largest operand.

Logical operators

Logical operator evaluates always to a one bit value either true(1) or false (0) or x (don’t care) . If any operand bit is either x or z it is equivalent to x

operation	symbol	operand
logical and	&&	binary
logical or	||	binary
logical not	!	unary

Logical operator examples

Example:1

```
a1 = 1’b0;   // 0 is false;
a2 = 1’b1;   // 1 is true

  a1 && a2   is 0 (false)
  a1 || a2   is 1 (true)
  !a2     is 0 (false)
```

Example:2

```
 a=2’b10 ;   b=2’b00
                     a && b    //  evaluates to  0  ( 1 && 0)
a=2’b1x    b=2’b11
                   a || b      // is unknown, evaluates to x.
```

- .

Relational operators

- Relational operations return logical 0 or 1. If there is any x or z bit in operand then it will return x.

Operation	Symbol	Operand
greater	>	Binary
less than	<	Binary
Greater than or equal to	>=	Binary
Less than or equal to	<=	Binary

Relational operator examples

a = 5   b = 6   c = 2’b1x   d=2’b10

a >b // evaluates to 0

a <= b // evaluates to 1

d >= c // evaluates to x

Equality operators

Equality operators are the following

Operation	Symbol	Operand
logical equality	==	binary
logical inequality	!=	binary
case equality	===	binary
case inequality	!==	binary

Equality operators

- Equality operator can return 1 or 0.
- Logical equality operator (== , !=) will return x if any of the operand bit has x.
- Case equality operator compares both operand bit by bit including x and z bit. If it matches then returns 1 or else it returns 0. It doesn’t return x.

Example – Equality Operator

```
a=3; b=5; c=3’b100; d=3’b101; e=4’b1xxx;

f=4’b1xxx;      g=3’b1xxz

a !=b   // evaluates to 1.

e===f  // evaluates to 1.

f===g  // evaluates to 0.

d == e  // evaluates to x
```

Bitwise operators

Bitwise operations are performed on each bit of the operand

Operation	Symbol	Operand
Bitwise and	&	Binary
Bitwise or	|	Binary
Bitwise negation	~	Unary
Bitwise xor	^	Binary
Bitwise xnor	~^ or ^~	Binary

Bitwise operator - Examples

```
 a = 3’b111;    b = 3’b101;   d = 3’b1x1;

c = ~a;        // c = 3’b000
c = a & b;   // c = 3’b101
c = a & b;   // c = 3’b111
c = a ^ b;   //  c = 3’b010
c = a & d;    //  c = 3’b1x1
```

Reduction operators

Reduction operators are unary operators

Operation	Symbol	Operand
reduction and	&	unary
reduction nand	~&	unary
reduction or	|	unary
reduction nor	~|	unary
reduction xor	^	unary
reduction xnor	~^ or ~^	unary

Reduction operator - Examples

```
x = 5’b01100

c = &x     // c= 0 & 1&1&0&0     c=0

c = |x     // c= 0|1|1|0|0          c=1

c = ^x    // c=0^1^1^0^0       c=0
```

Shift operators

Shift operator can be shift left or shift right

Operation	Symbol	Operand
shift right	>>	unary
shift left	<<	unary

Example:

a = 4’b1011;

y = a >> 2;   // y =4’b0010,   0’s filled in MSB

y = a << 2;   // y = 4’b1100,   0’s filled in LSB

Concatenation operators

- Concatenation operator is used to append multiple operands.
- The operand must be sized.

```
a=3’b101;  b=3’b111;

y = {a,b};              // y = 6’b101111
y = {a,b,3’b010};    // y =9’b101111010
```

Replication operators

Replication operator is used to concatenate same number.

```
a=3’b101   b =2’b10

y = {2{a}};           // result  of y is  6’b101101
y = {2{a},2{b} };  // result  of y is 10’b1011011010
y = { 2{a},2’b10};  // result   of y is 8’b10110110
```

Conditional operators

Conditional operator ? :

format:

conditional_expr ? true expr : false expr;

eg:

assign out = control ? I1 : l2;

control	out
1	I1
0	I2

Conditional Operator -Example

```
module mux_con(out,s0,s1,i);
input s0,s1;
input [3:0]i;
output out;
wire out;
assign out = s1 ? ( s0 ? i[3]:i[2]) : (s0 ? i[1]:i[0]) ;
endmodule
```