Lecture - 5

☆ Verilog Operators

* Arithmatic Operators

- Unary operator
  - + : unary plus : Eg: +A
  - − : unary minus : Eg: −A
- Binary operator
  - + : binary plus
  - − : binary minus
  - * : multiply
  - / : divide
  - % : modulus
  - ** : exponential

* Logical Operators

! logical negation && logical AND || logical OR	* These should be used for checking condition like in "if" statement

* Relational Operators

!=	not equal
==	equal
>=	greater or equal
<=	less or equal
>	greater
<	less

* Bitwise Operators

~	bitwise NOT
&	bitwise AND
|	bitwise OR
^	bitwise XOR
~^	bitwise XNOR

Eg:
```
wire a, b, f1;
assign f1 = ~(a|b);
```

* Reduction Operators (unary operators)
  - input : a single word operand (Eg: 0 1 1 1 0)
  - output : a single bit output
  - Eg:

i/p: a = 1 0 0 1

o/p: b

- Operators

&	bitwise AND
|	bitwise OR
~&	bitwise NAND
~|	bitwise NOR
^	bitwise XOR
~^	bitwise XNOR

Eg:

```
wire [7:0] a, b, c;
wire f1, f2, f3;
assign a = 4'b0111;
assign b = 4'b1100;
assign c = 4'b1010;
assign f1 = ^a
assign f2 = & (a & b)
assign f3 = (^a) & (~b)
```

* Shift Operators

insert zeros { >> shift right ——→ Eg: assign target = data >>3;
<< shift left

insert MSB { >>> arithmatic shift right

insert zeros { <<< arithmatic shift left

- >> & >>> work in similar way
- << & <<< work in similar way

* Conditional Operator

- cond_expr ? true_expr : false_expr ;

- Eg:
```
wire a, b, c ;
wire [7:0] x, y, z ;
assign a = (b > c) ? b : c ;
assign z = (x == y) ? x+2 : x-2 ;
```

* Concatenation Operation

- {..., ..., ...}

- Eg:
```
assign f = {a, b} ;
assign f = {a, 3'b101, b} ;
assign f = {x[2], y[0], a} ;
```

* Replication Operator

- {n {m}} ⟶ repeat 'm', 'n' times

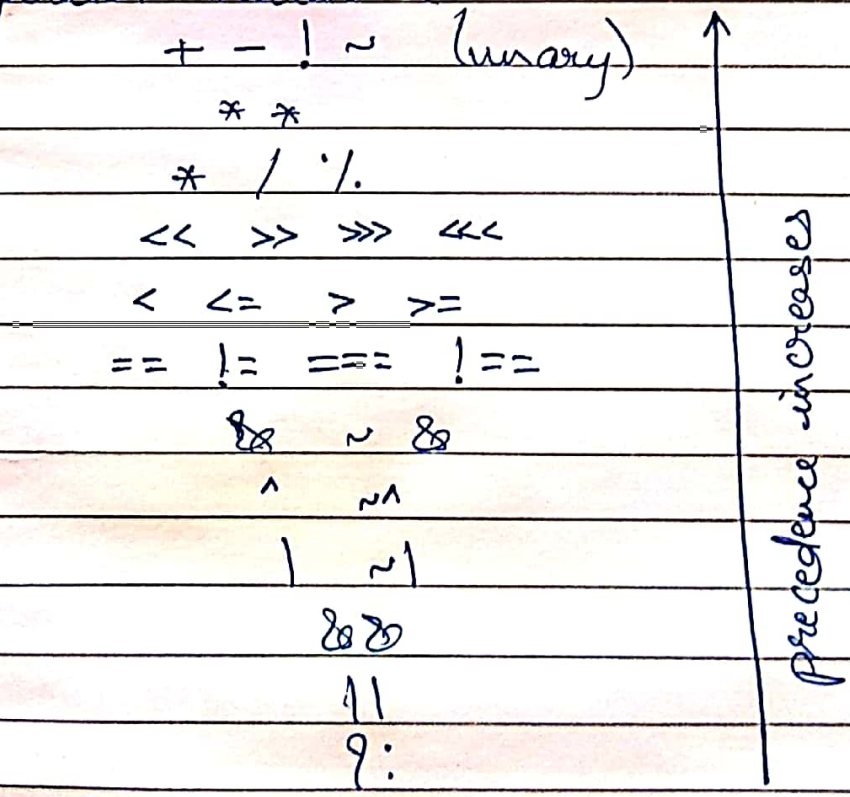
- Eg: assign f = {2'b10, 3{2'b01}, x}

  ↳ f ⇒ 1 0 01 01 01 x

* Eg:

```
module operator_eg(x,y,f1,f2);
    input x,y;
    output f1,f2;
    wire [9:0] x,y;
    wire [4:0] f1;
    wire f2;
    assign f1 = x[4:0] & y[4:0];
    assign f2 = x[2] | ~f1[3];
    assign f2 = ~ & x;
    assign f1 = f2 ? x[9:5] : x[4:0];
endmodule
```

* Operator Precedence



- all operators associate left to right in an expression except ?:

Eg: a + b + c

1st

2nd

left ⟶ right

- parentheses can be used to change the precedence.

☆ Homework

* statement 1: assign f = a + b;
  statement 2: f = a + b;

Q] When to use assign statement ?
Q] Find the meaning of : ==, !=, ===, !==.