2) a) i) B = &a;

i) B = 0 — It means B has atleast 1 '0'.

ii) B = 1 — It means B has no '0'.

ii) B = ^a;

i) B = 1 — It has odd no. of ones

B = 0 - It has ~~odd~~ even no. of ones.

b) i) Right shift - used for divison by 2^n factor

Left shift - ~~division~~ by Used for multiplication by factor of 2^n and general multiplication

ii) 65 * 37 implement using shift

$= (64 + 1) * 37$

$= (2^6 + 1) * 37 = 2^6 * 37 + 37$

= (left shift 37 by 6) + 37

37 =

100101 = 37

37 shifted = 37 << 6 = 100101 000 000

$+$ 100101

✓ 100101 100101

= 2405

d) Same symbols different meaning:

Bitwise = &, |, ^, ~^

Unary = &, |, ^, ~^

e) A = 6; B = 0; C = x.

A && B = 6 && 0

= True && False

= False = 0

ii) A || !B = True or !False

= True = 1.

iii) C || B

x || 0 = x or 0 = x.

iv) A = 8 B = -9 , C = x

i) A && B = True && True = True

ii) A || !B ⇒ True or !True = True

iii) C || B ⇒ X or True = True.

f) !1011 = !True = False = 0

~1011 = 0100

g) Code -

3) a = b << c;

Operand b, c

operator <<.

4) Left shifting makes multiplication by a factor of 2^n. This is useful in multiplication of any number. ie, 35 × 31, (32-1)35

Ex

$= (2^5 - 1)35$

$= 2^5 \times 35 - 35$

= (left shift 35 by 5) - 35

Ans: = (35 << 5) - 35

5) a = -1, b = 1, c = a && b;

c = -1 && 1

True & True = True = c

∴ c = 1

6) a = real = 3

integer, b = 10

real $c = \frac{b}{a} = \frac{10}{3} = 3.333$

7) reg [5:0] a; reg [-2:-1] b;

a = 95   b = 69

95 = 64 + 31

c = a & b

69 = 64 + 5

```
=  1011111
   1000101
   -------
   1000101 = 69
```

c = a | b;

```
   1011111
   1000101
   -------
   1011111  = 95
```

8) Unary reduction gives us info about vectors.

^ = gives idea about parity.

& = gives idea about presence of 0 in the vector.

| = gives idea about presence of 1

9) Operators are used directly with the vectors. but with arrays they need to work one vector at a time.

10) Toto we use bit masking.

∴ Input & 16'b 0000_0000_1111_0000

∴ Eg 1011_1010_0011_1111

& 0000_0000_1111_0000

0000_0000_0011_0000.

└ Ans.

11) Types of operator in verilog :-

a) Arithmetic :- {+ - * / %}

b) Logical {&&, ||}

c) Relational {<=, >=, ==, <, >}

d) Concatenation - {}

e) Bitwise - &, |, ^, ~^

f) Unary reduction &, |, ^, ~^

g) Shift <<, >>

h) Conditional ? () : ()

i) Case equality !==, ==, ===

j) Repetition