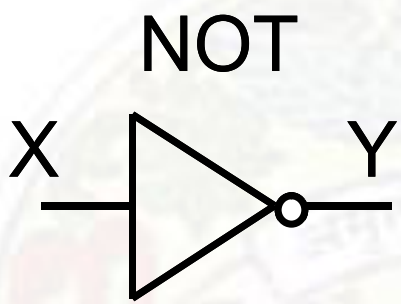
Basic Logic Gates

By: Dr. Yogendar Kr. Verma

Basic Logic Gates and Basic Digital Design

- **NOT, AND, and OR Gates**
- NAND and NOR Gates
- Exclusive-OR (XOR) Gate
- Multiple-input Gates

NOT Gate -- Inverter



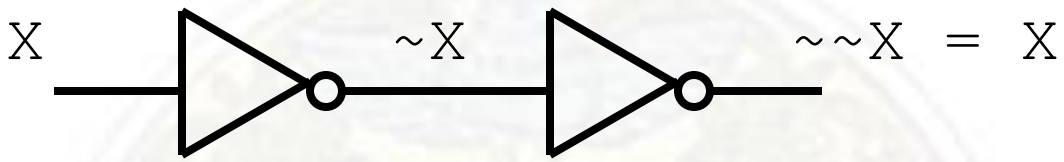
Y = ~X

X	Y
0	1
1	0

NOT

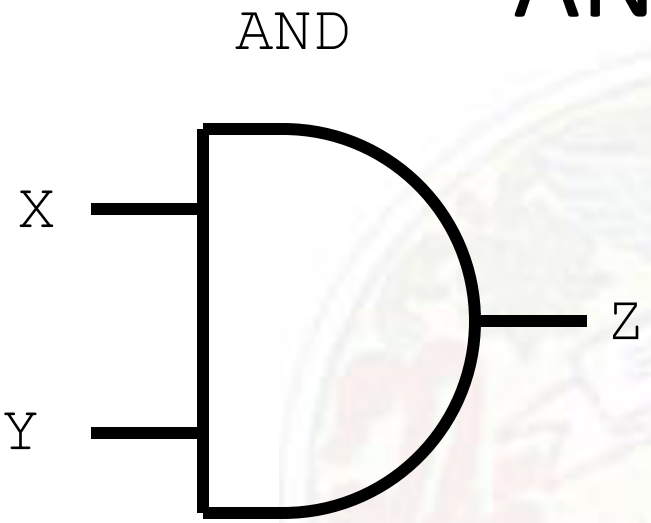
- `Y = ~X` (Verilog)
- `Y = !X` (ABEL)
- `Y = not X` (VHDL)
- `Y = X'`
- `Y = ¬X`
- $Y = \overline{X}$ (textook)
- `not(Y,X)` (Verilog)

NOT



X	~X	~~X
0	1	0
1	0	1

AND Gate



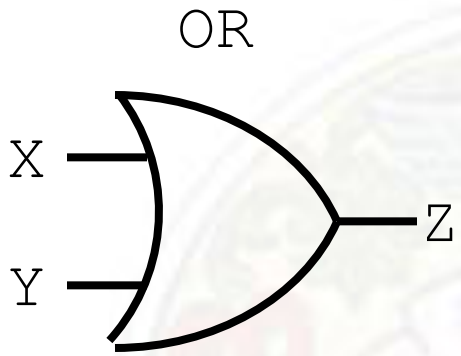
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Z = X & Y

AND

- X & Y (Verilog and ABEL)
- X **and** Y (VHDL)
- X ∧ Y
- X ∩ Y
- X * Y
- XY (textbook)
- **and**(Z,X,Y) (Verilog)

OR Gate



Z = X | Y

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

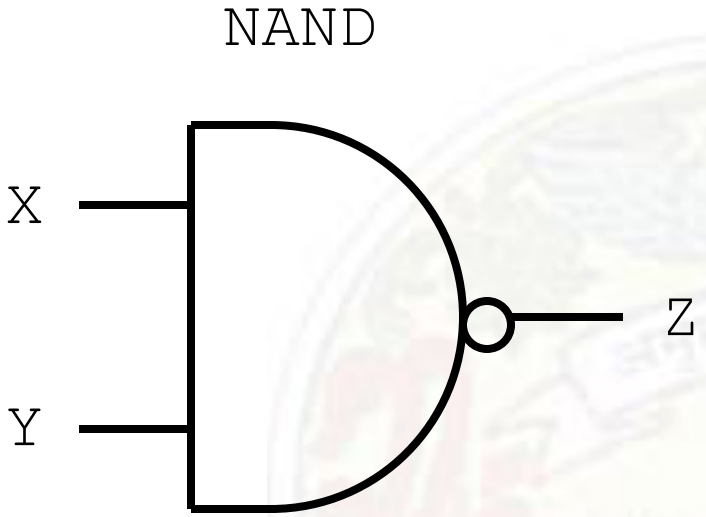
OR

- `X | Y` (Verilog)
- `X # Y` (ABEL)
- `X or Y` (VHDL)
- `X + Y` (textbook)
- `X V Y`
- `X U Y`
- `or(Z,X,Y)` (Verilog)

Basic Logic Gates and Basic Digital Design

- NOT, AND, and OR Gates
- **NAND and NOR Gates**
- Exclusive-OR (XOR) Gate
- Multiple-input Gates

NAND Gate

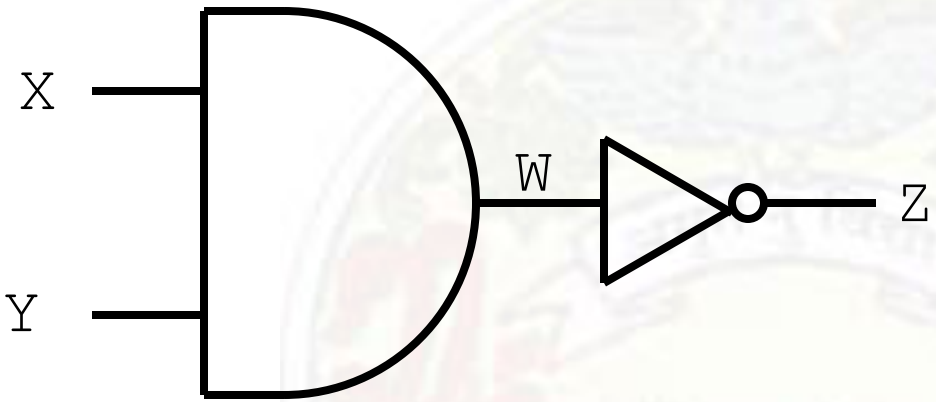


X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

```
Z = ~(X & Y)
nand(Z,X,Y)
```

NAND Gate

NOT-AND

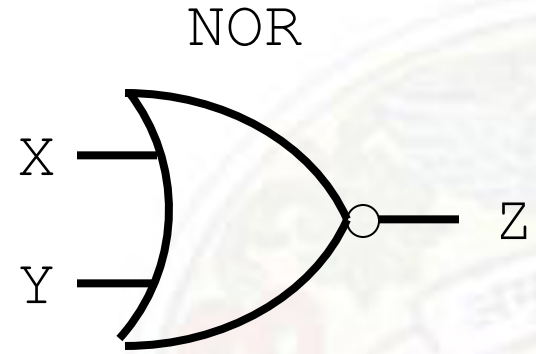


X	Y	W	Z
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

$W = X \ \& \ Y$

$Z = \sim W = \sim (X \ \& \ Y)$

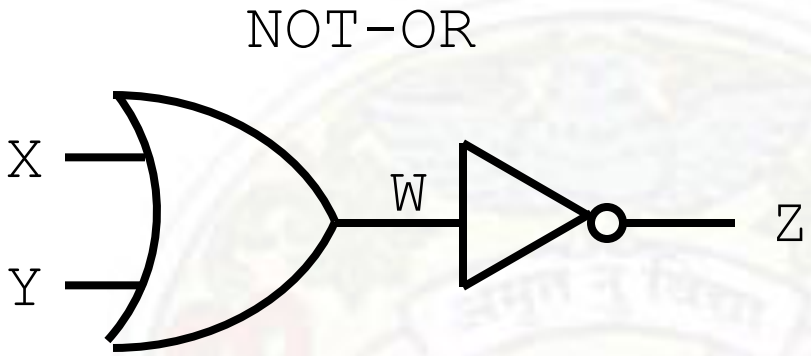
NOR Gate



X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

```
Z = ~(X | Y)
nor(Z,X,Y)
```

NOR Gate



X	Y	W	Z
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

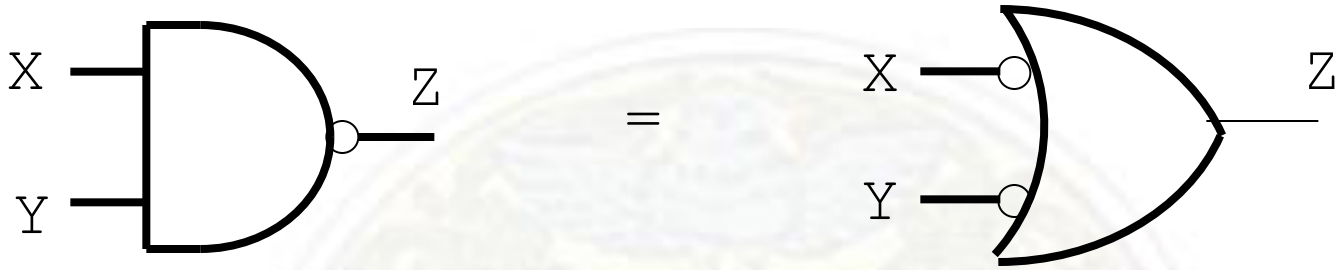
$$W = X \mid Y$$

$$Z = \sim W = \sim (X \mid Y)$$

Basic Logic Gates and Basic Digital Design

- NOT, AND, and OR Gates
- NAND and NOR Gates
- Exclusive-OR (XOR) Gate
- Multiple-input Gates

NAND Gate



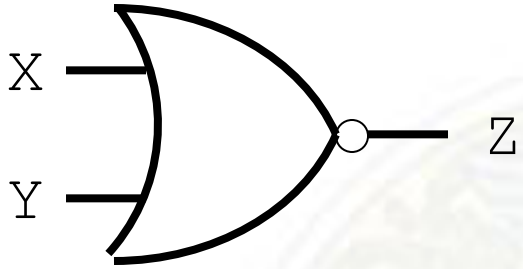
Z = ~(X & Y)

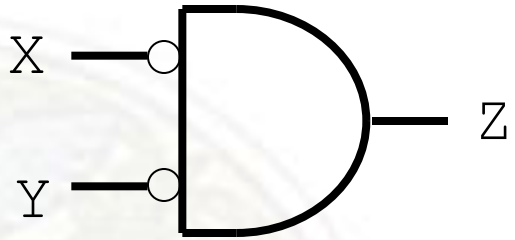
X	Y	W	Z
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Z = ~X | ~Y

X	Y	~X	~Y	Z
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

NOR Gate





`Z = ~(X | Y)`

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

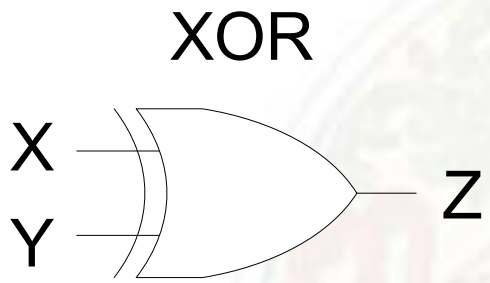
`Z = ~X & ~Y`

X	Y	~X	~Y	Z
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

Basic Logic Gates and Basic Digital Design

- NOT, AND, and OR Gates
- NAND and NOR Gates
- **Exclusive-OR (XOR) Gate**
- Multiple-input Gates

Exclusive-OR Gate



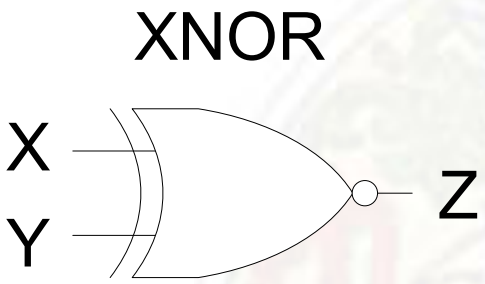

```
Z = X ^ Y
xor(Z,X,Y)
```

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

XOR

- `X ^ Y` (Verilog)
- `X $ Y` (ABEL)
- `X @ Y`
- $X \oplus Y$ (textbook)
- **`xor`**`(Z,X,Y)` (Verilog)

Exclusive-NOR Gate




```
Z = ~(X ^ Y)
Z = X ~^ Y
xnor(Z,X,Y)
```

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

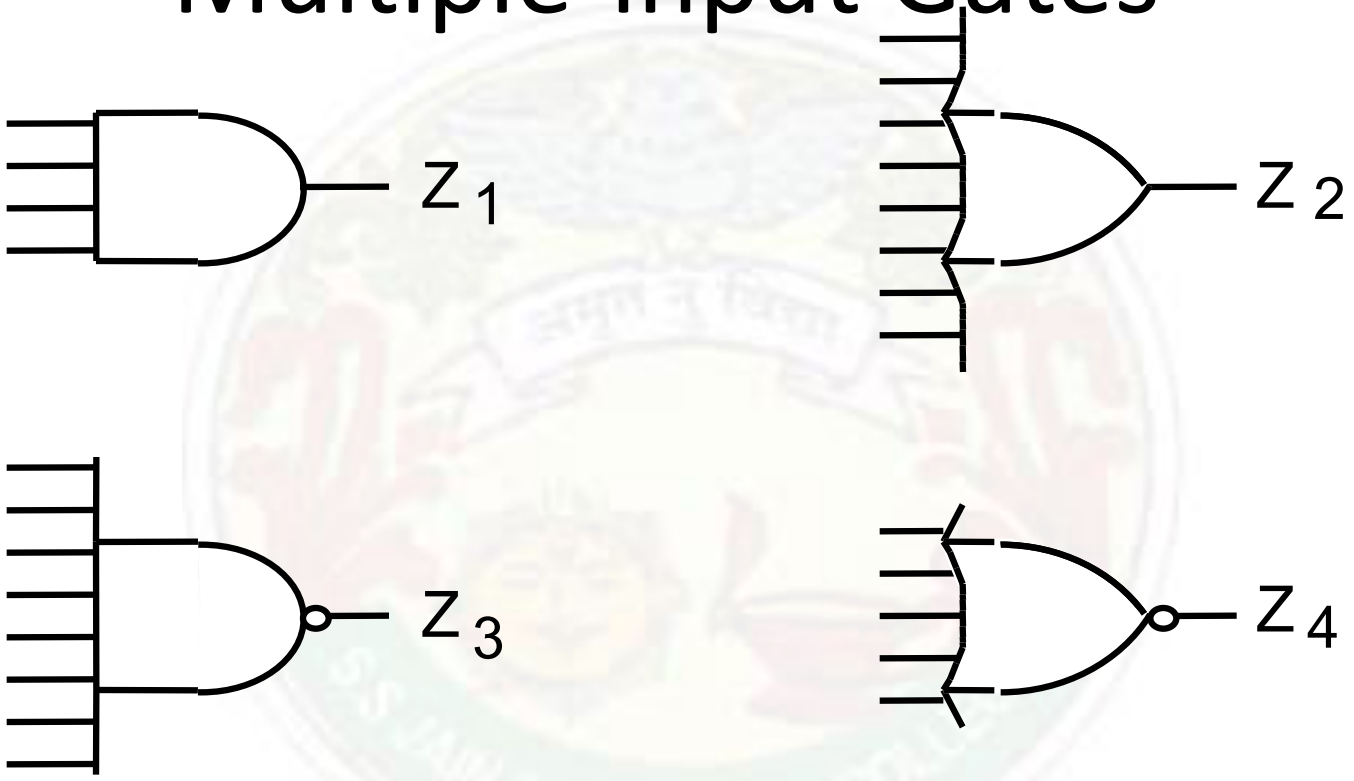
XNOR

- `X ~^ Y` (Verilog)
- `!(X $ Y)` (ABEL)
- `X @ Y`
- X Y
- **xnor**`(Z,X,Y)` (Verilog)

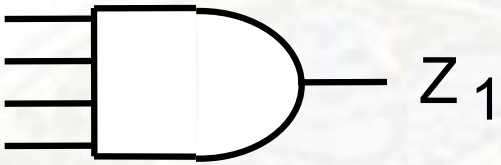
Basic Logic Gates and Basic Digital Design

- NOT, AND, and OR Gates
- NAND and NOR Gates
- Exclusive-OR (XOR) Gate
- **Multiple-input Gates**

Multiple-input Gates



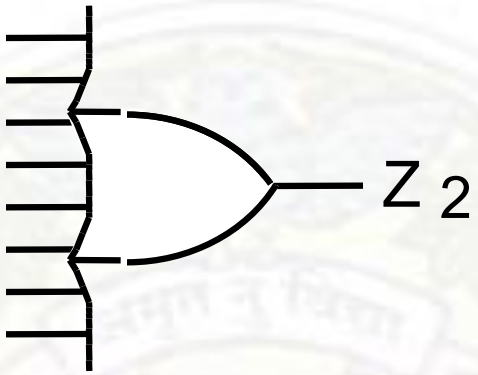
Multiple-input AND Gate



Output Z_1 is HIGH only if all inputs are HIGH

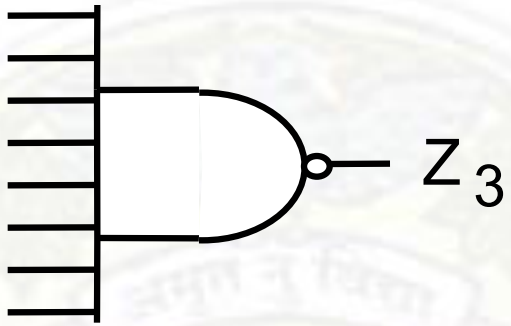
An open input will float HIGH

Multiple-input OR Gate



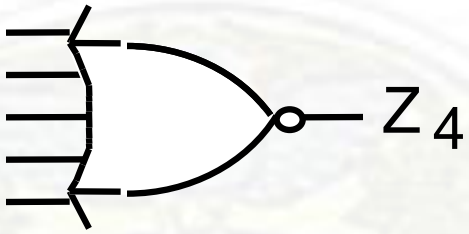
Output Z_2 is LOW only if all inputs are LOW

Multiple-input NAND Gate



Output Z_3 is LOW only if all inputs are HIGH

Multiple-input NOR Gate



Output Z_4 is HIGH only if all inputs are LOW

Thanks you……