Basic Logic Gates

Md Saeed Siddik

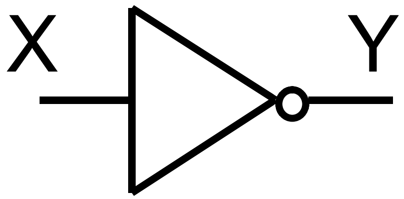
Assistant Professor,
IIT Dhaka University

Basic Logic Gates and Basic Digital Design

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

NOT Gate -- Inverter

NOT



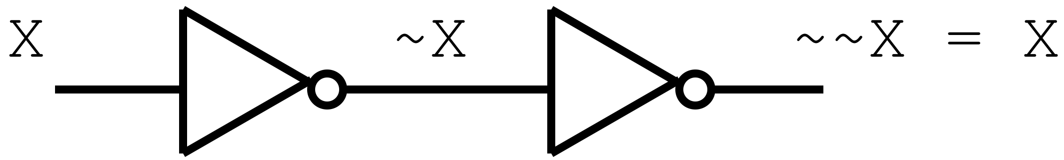
Y = ~X

X	Y
0	1
1	0

NOT

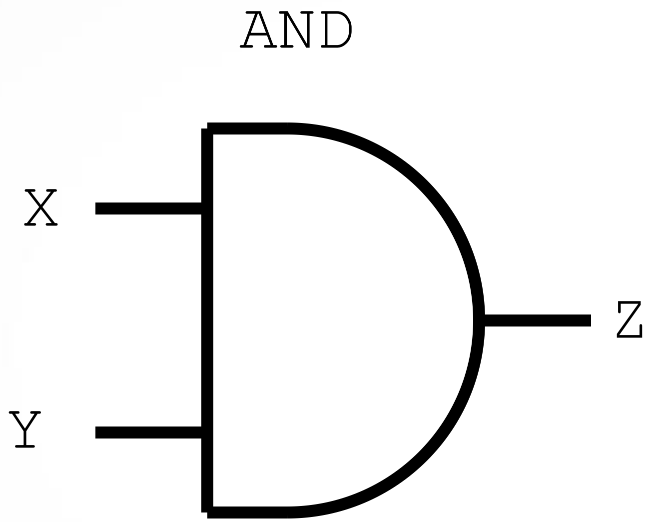
- Y = ~X (Verilog)
- Y = !X (ABEL)
- Y = **not** X (VHDL)
- Y = X′
- Y = ¬X
- Y = $\overline{X}$ (textook)
- **not**(Y,X) (Verilog)

NOT



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

AND Gate



Z = X & Y

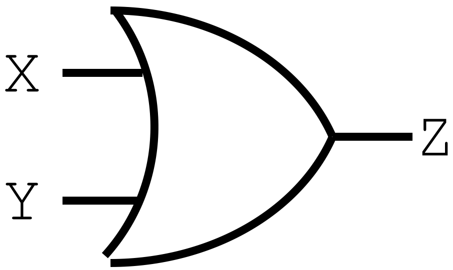
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

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

OR



Z = X | Y

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

OR

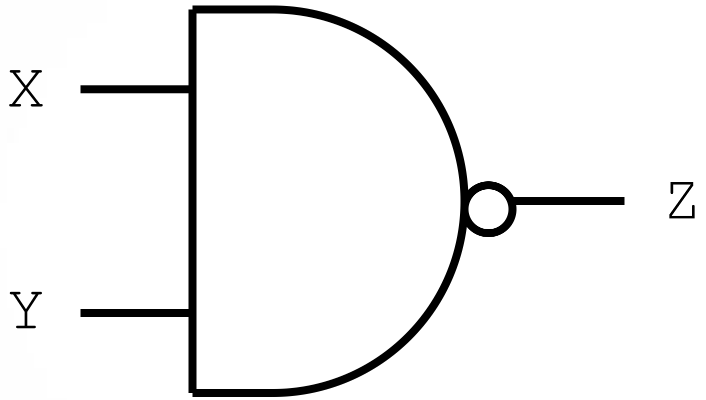
- X | Y (Verilog)
- X # Y (ABEL)
- X **or** Y (VHDL)
- X + Y (textbook)
- X V Y
- X U Y
- **or**(Z,X,Y) (Verilog)

Basic Logic Gates and Basic Digital Design

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

NAND Gate

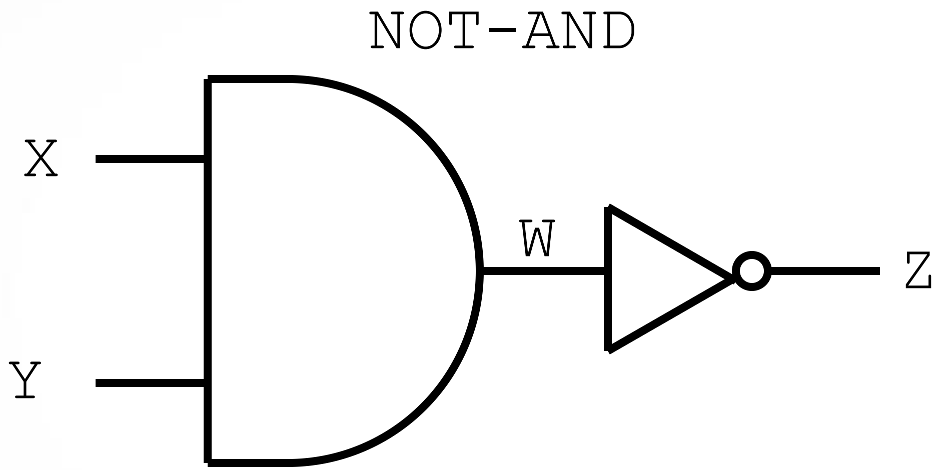
NAND



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

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

NAND Gate



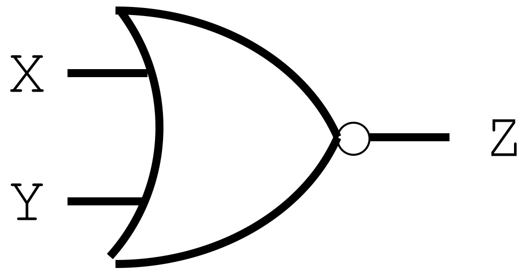
X	Y	W	Z
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

```
W = X & Y

Z = ~W = ~(X & Y)
```

NOR Gate

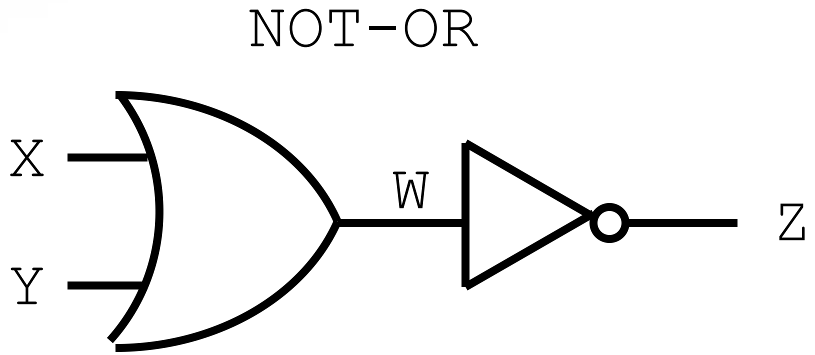
NOR




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

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

NOR Gate



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

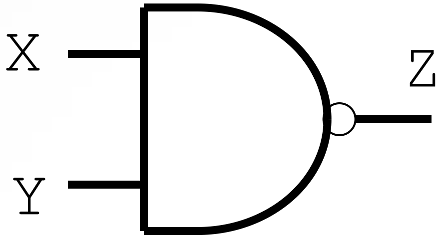
W = X | Y

Z = ~W = ~(X | Y)

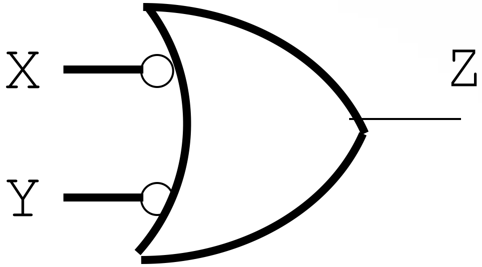
Basic Logic Gates and Basic Digital Design

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

NAND Gate



=



Z = ~(X & Y)

Z = ~X | ~Y

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

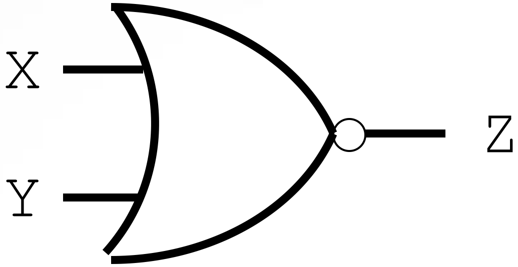
X	Y	~X	~Y	Z
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

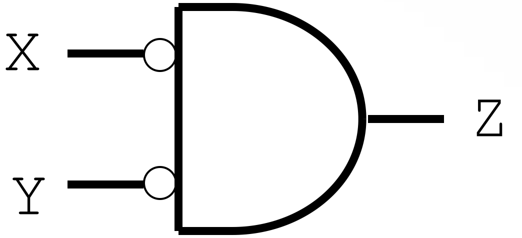
De Morgan’s Theorem-1

```
~(X & Y) = ~X | ~Y
```

- NOT all variables
- Change & to | and | to &
- NOT the result

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

De Morgan’s Theorem-2

```
~(X | Y) = ~X & ~Y
```

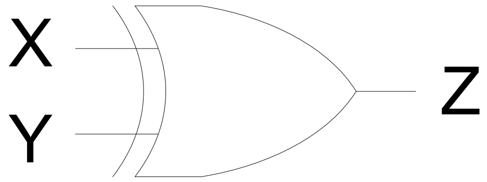
- NOT all variables
- Change & to | and | to &
- NOT the result

Basic Logic Gates and Basic Digital Design

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

Exclusive-OR Gate

XOR



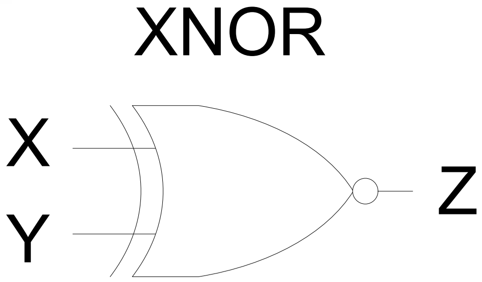

```
Z = X ^ Y
xor(Z,X,Y)
```

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

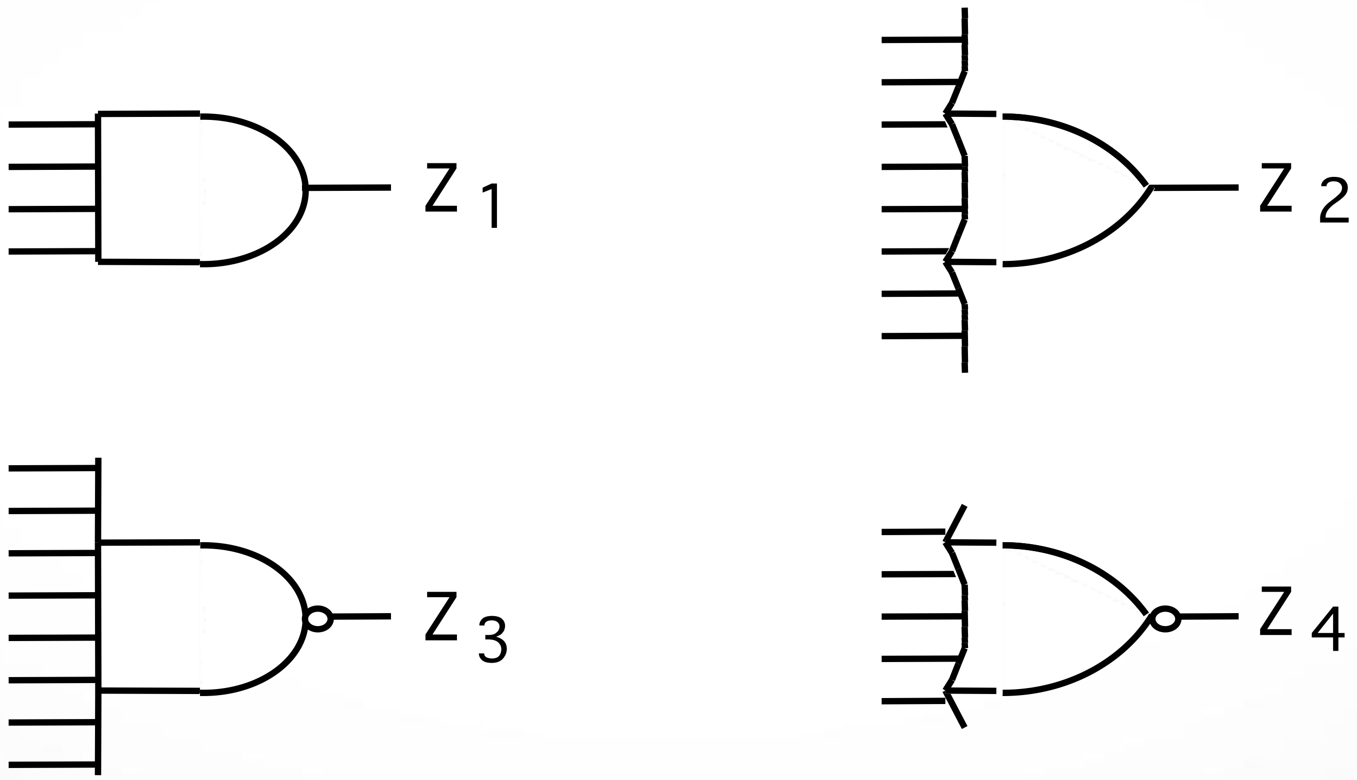
XNOR

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

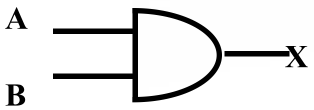
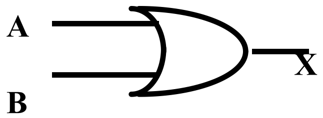
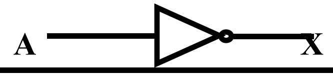
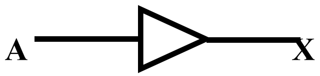
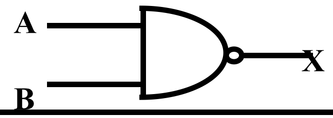
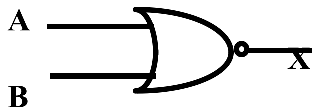
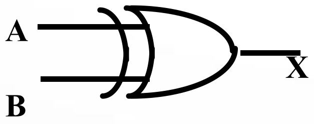
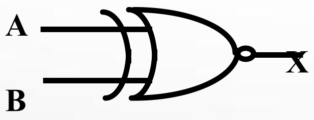
Basic Logic Gates and Basic Digital Design

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

Multiple-input Gates



COMBINATIONAL GATES

Name	Symbol	Function	Truth Table
AND		**X = A • B** **or** **X = AB**	**A B X** **0 0 0** **0 1 0** **1 0 0** **1 1 1**
OR		**X = A + B**	**A B X** **0 0 0** **0 1 1** **1 0 1** **1 1 1**
I		**X = A’**	**A X** **0 1** **1 0**
Buffer		**X = A**	**A X** **0 0** **1 1**
NAND		**X = (AB)’**	**A B X** **0 0 1** **0 1 1** **1 0 1** **1 1 0**
NOR		**X = (A + B)’**	**A B X** **0 0 1** **0 1 0** **1 0 0** **1 1 0**
XOR **Exclusive OR**		**X = A ⊕ B** **or** **X = A’B + AB’**	**A B X** **0 0 0** **0 1 1** **1 0 1** **1 1 0**
XNOR **Exclusive NOR or Equivalence**		**X = (A ⊕ B)’** **or** **X = A’B’+ AB**	**A B X** **0 0 1** **0 1 0** **1 0 0** **1 1 1**

https://logic.ly/demo/

Live demo

BOOLEAN ALGEBRA

Boolean Algebra

Boolean Algebra

- Algebra with Binary(Boolean) Variable and Logic Operations
- Boolean Algebra is useful in Analysis and Synthesis of Digital Logic Circuits
- Input and Output signals can be represented by Boolean Variables
- Terminology:
  - *Literal:* A variable or its complement
  - *Product term:* literals connected by •
  - *Sum term:* literals connected by +

Boolean Algebra Properties

Let X: boolean variable, 0,1: constants

1. $X + 0 = X$ -- Zero Axiom
2. $X \bullet 1 = X$ -- Unit Axiom
3. $X + 1 = 1$ -- Unit Property
4. $X \bullet 0 = 0$ -- Zero Property

Boolean Algebra Properties (cont.)

Let X: boolean variable, 0,1: constants

5. $X + X = X$ -- Idepotence
6. $X \bullet X = X$ -- Idepotence
7. $X + X' = 1$ -- Complement
8. $X \bullet X' = 0$ -- Complement
9. $(X')' = X$ -- Involution

Algebraic Manipulation

- Boolean algebra is a useful tool for simplifying digital circuits.
- Why do it? Simpler can mean cheaper, smaller, faster.
- Example: Simplify F = x'yz + x'yz' + xz.

$$
\begin{aligned}
F &= x'yz + x'yz' + xz \\
&= x'y(z+z') + xz \\
&= x'y\bullet 1 + xz \\
&= x'y + xz
\end{aligned}
$$

Algebraic Manipulation (cont.)

- Example: Prove

  $x'y'z' + x'yz' + xyz' = x'z' + yz'$

- **Proof:**

  $x'y'z' + x'yz' + xyz'$

  $= x'y'z' + x'yz' + x'yz' + xyz'$

  $= x'z'(y'+y) + yz'(x'+x)$

  $= x'z' \bullet 1 + yz' \bullet 1$

  $= x'z' + yz'$

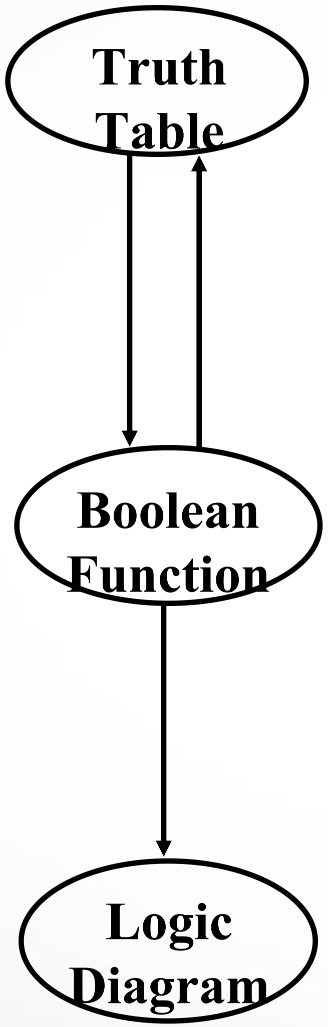
Truth Table

The most elementary specification of the function of a Digital Logic Circuit is the Truth Table

Table that describes the Output Values for all the combinations of the Input Values, called *MINTERMS*

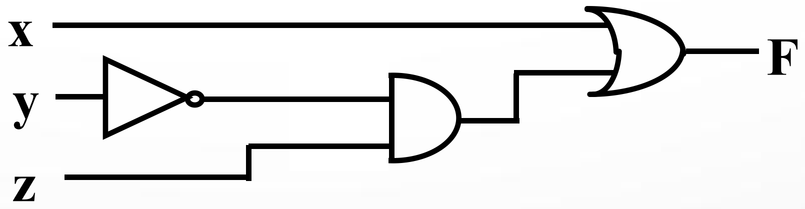
- n input variables $\rightarrow 2^n$ minterms

LOGIC CIRCUIT DESIGN



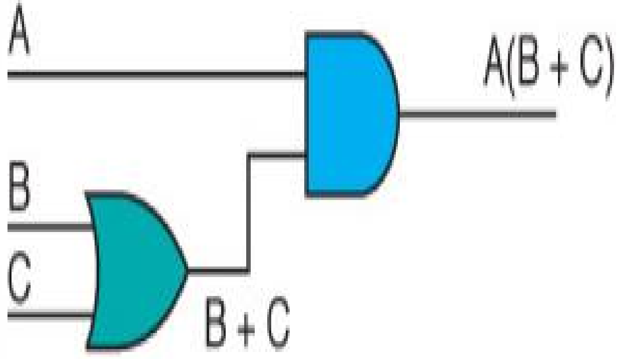
x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$$F = x + y'z$$



Combinational Circuits

Consider the following Boolean expression A(B + C)



A	B	C	B + C	A(B + C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Theorem to Prove

(A+B) . (A+C) = A+(B . C) (Distributive Law)

A	B	C	A+B	A+C	(A+B)· (A+C)	B· C	A+(B· C)
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Daily Life Example: Choosing a Movie

- You're trying to decide what movie to watch tonight. You have three criteria:
  - A: The movie is a Comedy
  - B: The movie has good reviews
  - C: Your friend is available to watch
- Let's say you decide on the following rule for watching a movie:

"We watch a movie if it's a Comedy AND has good reviews, OR if my friend is available."

Boolean algebra

- In Boolean algebra, this would look like: (A x B) + C

A (Comedy)	B (Good Reviews)	C (Friend Available)	A· B (Comedy AND Good Reviews)	(A· B)+C (Watch Movie?)
0 (No)	0 (No)	0 (No)	0	0 (No Movie)
0 (No)	0 (No)	1 (Yes)	0	1 (Watch Movie)
0 (No)	1 (Yes)	0 (No)	0	0 (No Movie)
0 (No)	1 (Yes)	1 (Yes)	0	1 (Watch Movie)
1 (Yes)	0 (No)	0 (No)	0	0 (No Movie)
1 (Yes)	0 (No)	1 (Yes)	0	1 (Watch Movie)
1 (Yes)	1 (Yes)	0 (No)	1	1 (Watch Movie)
1 (Yes)	1 (Yes)	1 (Yes)	1	1 (Watch Movie)

End of this Lecture