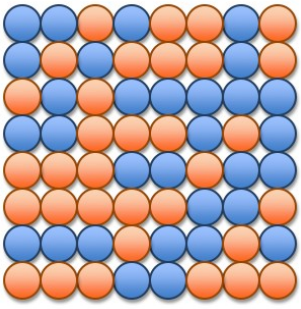


UNIT 8A

Computer Organization: Boolean Logic, Computational Circuits

Boolean Logic & Truth Tables

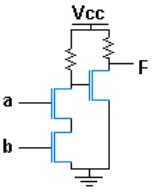
- Computer circuitry works based on Boolean logic: operations on true (1) and false (0) values.

A	B	A ∧ B (A AND B) (conjunction)	A ∨ B (A OR B) (disjunction)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

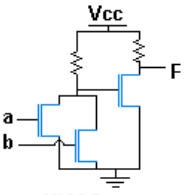
A	¬A (NOT A) (negation)
0	1
1	0

Gates

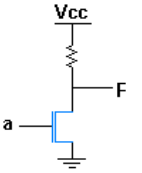
- A gate is an electronic device that implements a logical function. (Images from Wikipedia)
- Circuit diagram:



NMOS AND gate



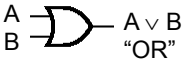
NMOS OR gate



NMOS NOT gate

- Abstract Diagram:

A ∧ B "AND"



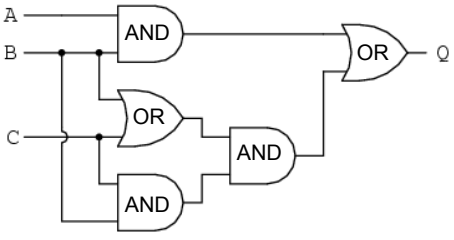
A ∨ B "OR"

¬ A "NOT"

Properties (Similar to × and +)

- Commutative: $a \wedge b = b \wedge a$ $a \vee b = b \vee a$
- Associative: $a \wedge b \wedge c = (a \wedge b) \wedge c = a \wedge (b \wedge c)$
  $a \vee b \vee c = (a \vee b) \vee c = a \vee (b \vee c)$
- Distributive: $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
  $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ ← Not true for + and ×
- Identity: $a \wedge 1 = a$ $a \vee 0 = a$
- Dominance: $a \wedge 0 = 0$ $a \vee 1 = 1$
- Idempotence: $a \wedge a = a$ $a \vee a = a$
- Complementation: $a \wedge \neg a = 0$ $a \vee \neg a = 1$
- Double Negation: $\neg \neg a = a$

Equivalence

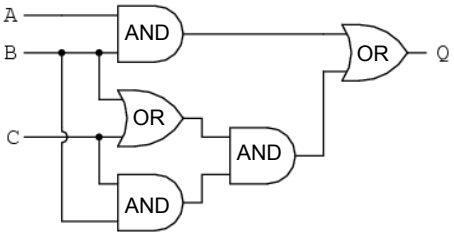


$Q = (A \wedge B) \vee ((B \vee C) \wedge (C \wedge B))$

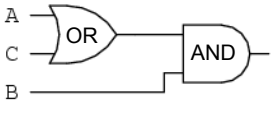
A	B	C	Q
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

http://www.allaboutcircuits.com/vol_4/chpt_7/6.html

Equivalence



$Q = (A \wedge B) \vee ((B \vee C) \wedge (C \wedge B))$



$Q = B \wedge (A \vee C)$

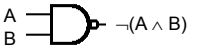
A	B	C	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

http://www.allaboutcircuits.com/vol_4/chpt_7/6.html

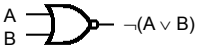
More gates

A	B	A nand B	A nor B	A xor B
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	0

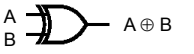
- nand ("not and"): A nand B = not (A and B)



- nor ("not or"): A nor B = not (A or B)



- xor ("exclusive or"):
  A xor B = (A and not B) or (B and not A)



DeMorgan's Law

Nand: $\neg(A \wedge B) = \neg A \vee \neg B$

```
if not (x > 15 and x < 110) then ...
```

is logically equivalent to

```
if (not x > 15) or (not x < 110) then ...
```

Nor: $\neg(A \vee B) = \neg A \wedge \neg B$

```
if not (x < 15 or x > 110) then ...
```

is logically equivalent to

```
if (not x < 15) and (not x > 110) then ...
```

A Logical Function Using Gates

3-bit odd parity checker

$F = (\neg x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge \neg z) \vee (x \wedge \neg y \wedge \neg z) \vee (x \wedge y \wedge z)$

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

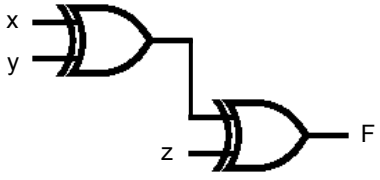
A Logical Function Using Gates

3-bit odd parity checker

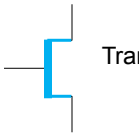
$F = (\neg x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge \neg z) \vee (x \wedge \neg y \wedge \neg z) \vee (x \wedge y \wedge z)$

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

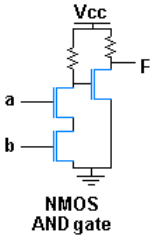
$F = x \oplus y \oplus z$

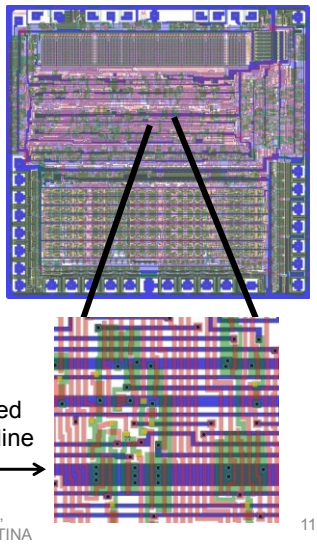


Gates Are Built From Transistors



Transistor (switch)





Every spot where a red line crosses a green line is a transistor. →