

UNIT 8A

Computer Organization: Boolean Logic, Computational Circuits

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

1

Boolean Logic & Truth Tables

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

A	B	$A \wedge B$ (A AND B) (conjunction)	$A \vee B$ (A OR B) (disjunction)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

A	$\neg A$ (NOT A) (negation)
0	1
1	0

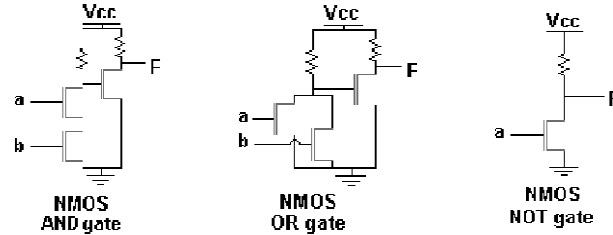
15110 Principles of Computing,
Carnegie Mellon University - CORTINA

2

Gates

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

- Circuit diagram:



- Abstract Diagram:



15110 Principles of Computing,
Carnegie Mellon University - CORTINA

3

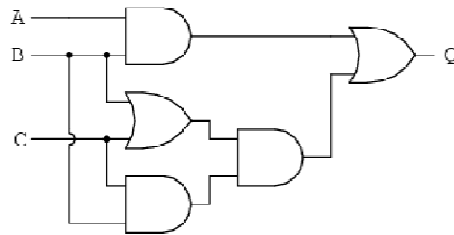
Properties

- 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: $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
- Identity: $a \wedge 1 = a$ $a \vee 0 = a$
- Idempotence: $a \wedge a = a$ $a \vee a = a$
- Complementation: $a \wedge \neg a = 0$ $a \vee \neg a = 1$
- Double Negation: $\neg \neg a = a$

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

4

Equivalence



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

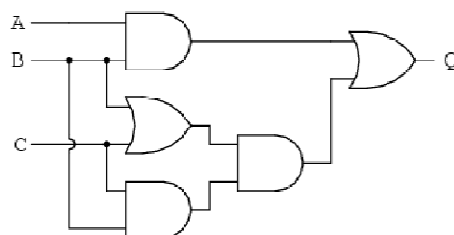
A	B	C	Q
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

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

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

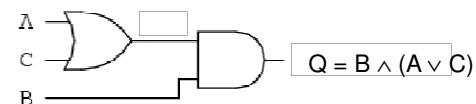
5

Equivalence



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

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



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

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

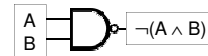
15110 Principles of Computing,
Carnegie Mellon University - CORTINA

6

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”): $A \text{ nand } B = \neg(A \wedge B)$



- nor (“not or”): $A \text{ nor } B = \neg(A \vee B)$



- xor (“exclusive or”):
 $A \text{ xor } B = (A \text{ and not } B) \text{ or } (B \text{ and not } A)$



15110 Principles of Computing,
Carnegie Mellon University - CORTINA

7

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 (x <= 15) or (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 (x >= 15) and (x <= 110) then ...`

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

8

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	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

9

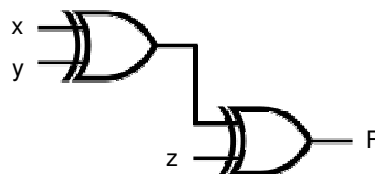
A Logical Function Using Gates

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

$$F = x \oplus y \oplus z$$



15110 Principles of Computing,
Carnegie Mellon University - CORTINA

10