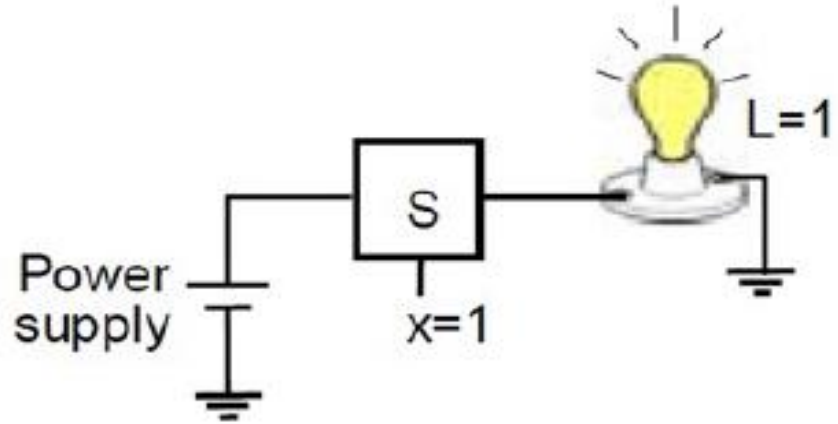
Lecture 3
LOGIC GATES

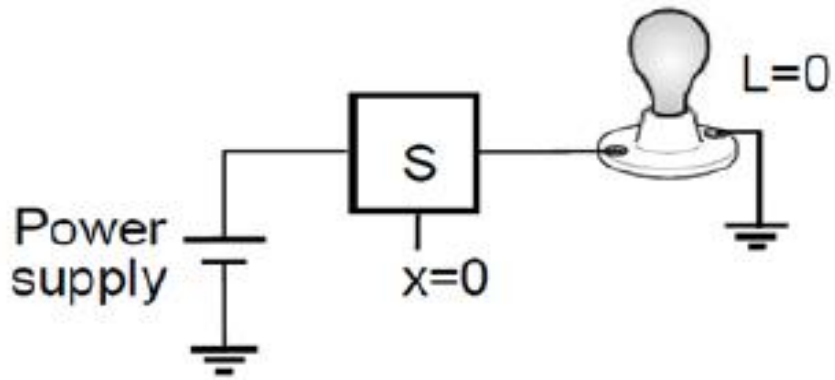
prepared by Eng.shada Abduladeem

Variable and Functions

Logic Expression

- We can describe the system as a logic expression
  - Input: switch control **x**
  - Output: light bulb **L**
- Light L is a function of the input variable x
  - L = 1 if x = 1
  - L =0 if x = 0
  - **Thus, L(x) = x**

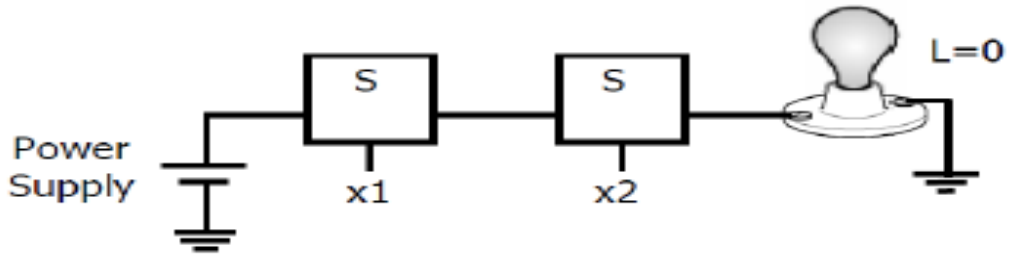


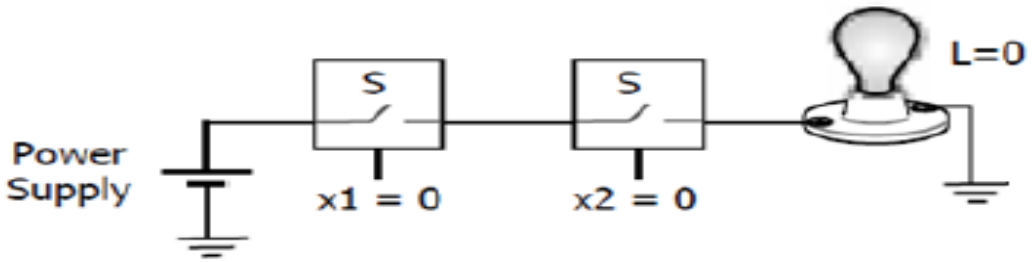


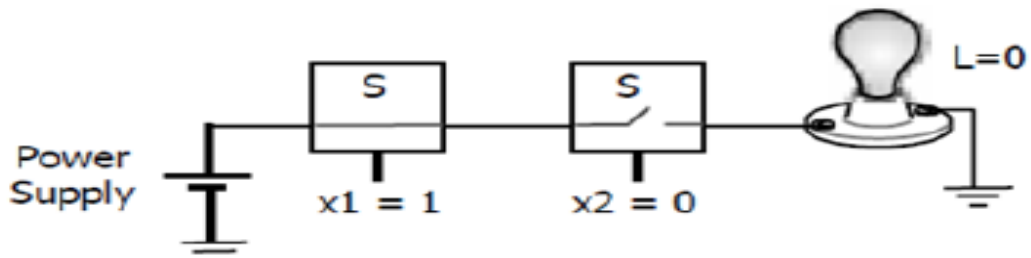
Variable and Functions

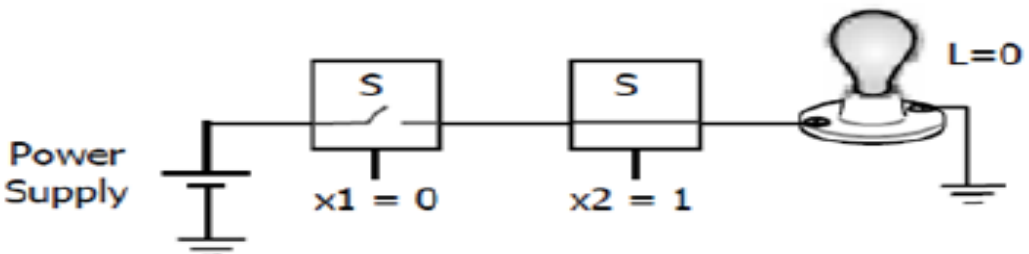
Two Switch Application-Series Connection

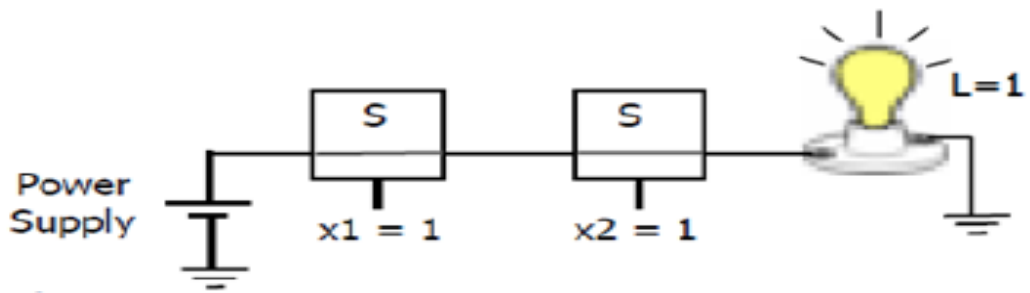
- What if we use two switches to control the light?
  - x1, x2 are control inputs to the switches
  - We first connect switches in series
- When will the light be on?











- Logical expression to describe behaviour
  - $L(x1, x2) = x1 \cdot x2$

The AND Gate

- **This is an AND gate.**
- **So, if the two inputs signals are asserted (high) the output will also be asserted. Otherwise, the output will be deasserted (low).**

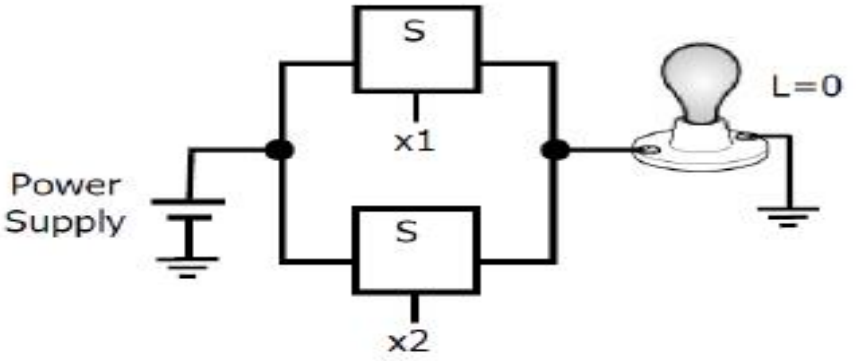
Truth Table

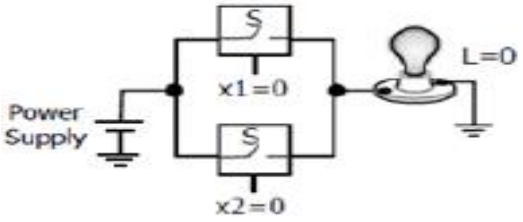
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

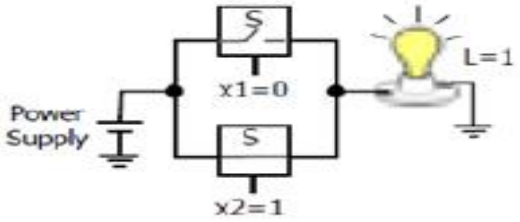
Variable and Functions

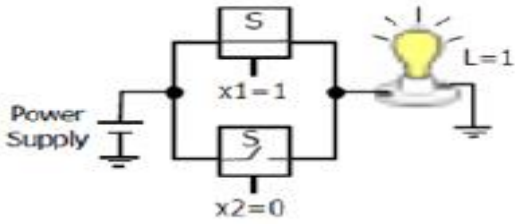
Two Switch Application-Parallel Connection

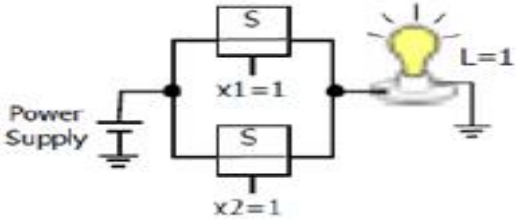
- We again use two switches to control the light
  - x1, x2 are control inputs to the switches
  - Connect switches in parallel
- When will the light be on?











- Logical expression to describe behaviour
  - L(x1, x2) = x1 + x2

The OR Gate

- **This is an OR gate.**
- **So, if either of the two input signals are asserted, or both of them are, the output will be asserted.**

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Inversion

NOT Operation Notation

- Different names for NOT operation
  - Complement, Invert, Inverse
- Different notations for NOT operation
  - $\overline{x} = x' = !x = \sim x$

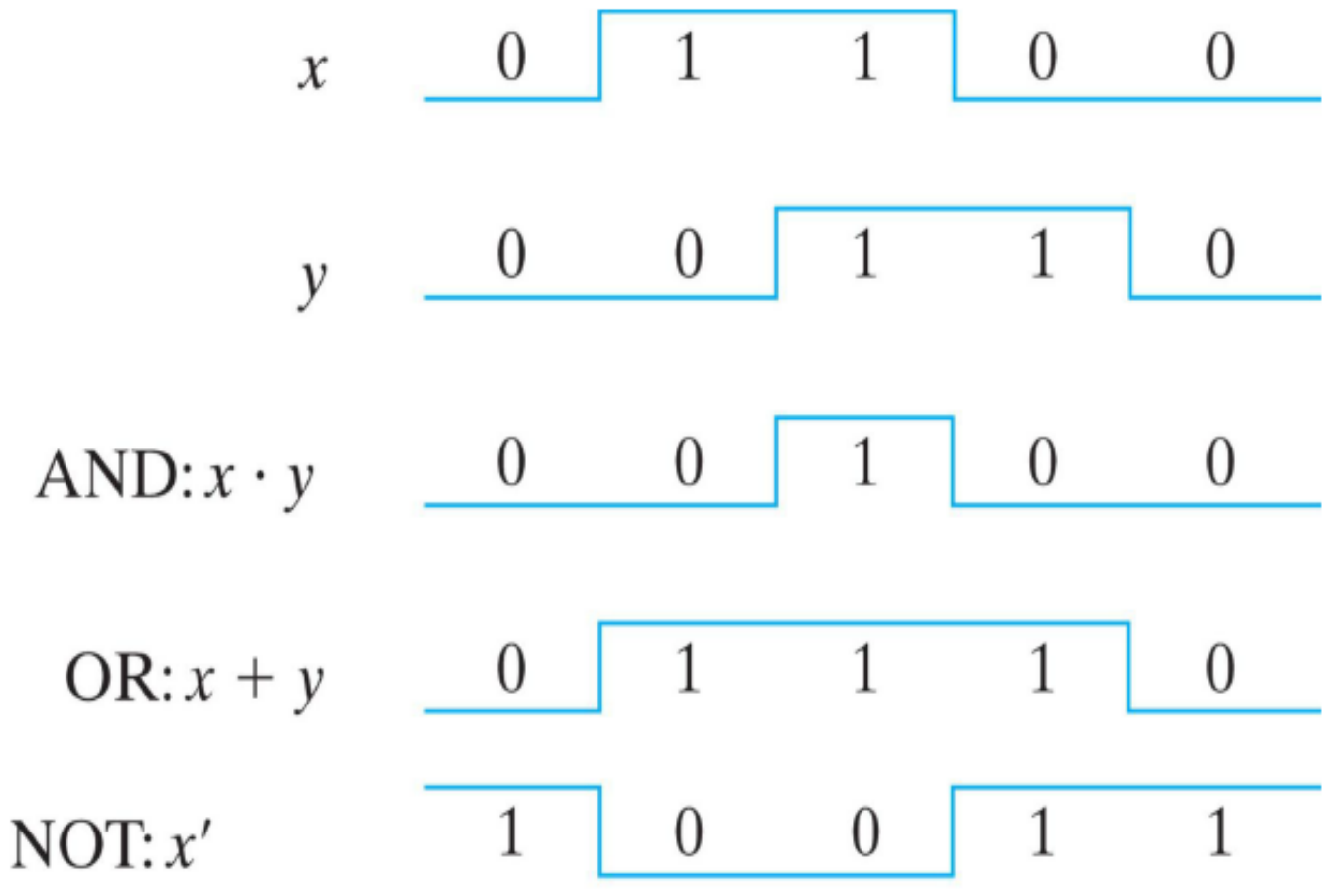
Symbol

Truth Table

A	Y
0	1
1	0

Input Output

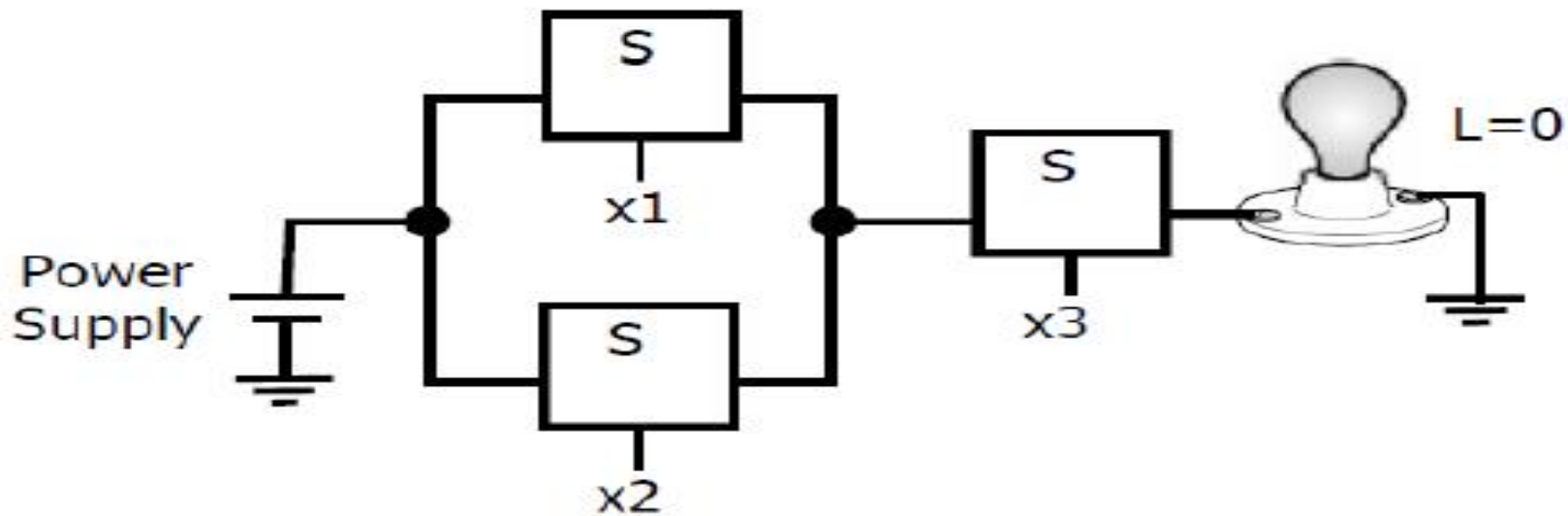
Input output signals for gates

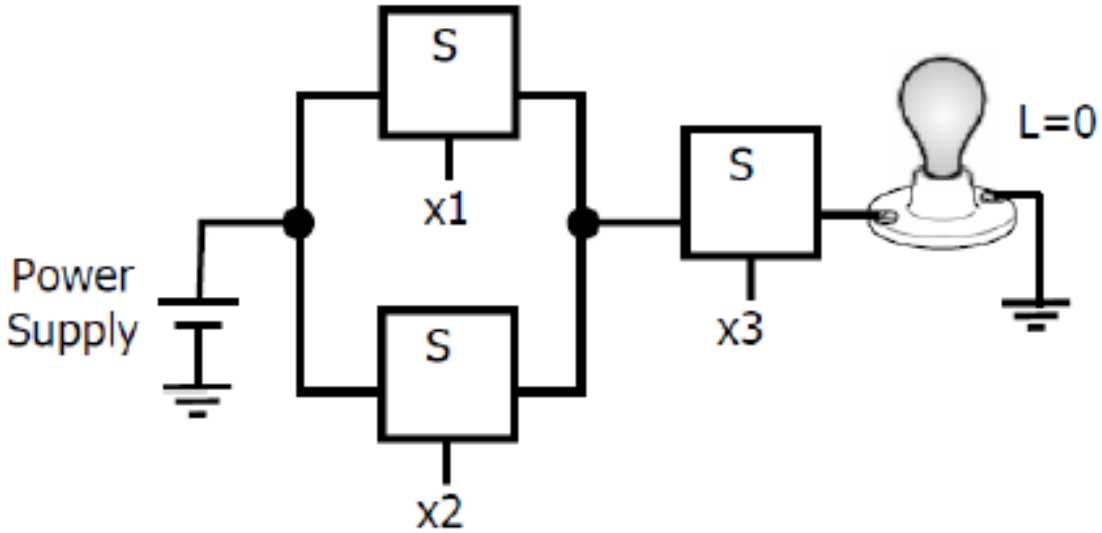


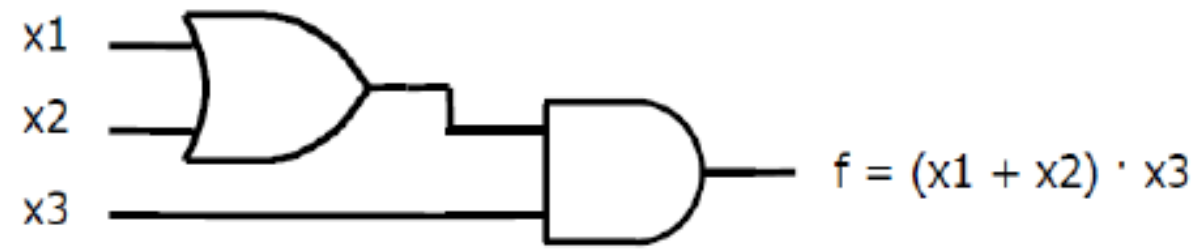
Variable and Functions

Three Switch Application

- AND and OR functions
  - Building blocks for larger circuits
- Example with three switches
  - x1, x2, and x3 are control inputs to the switches
  - Series-parallel connection
- When will the light be on?



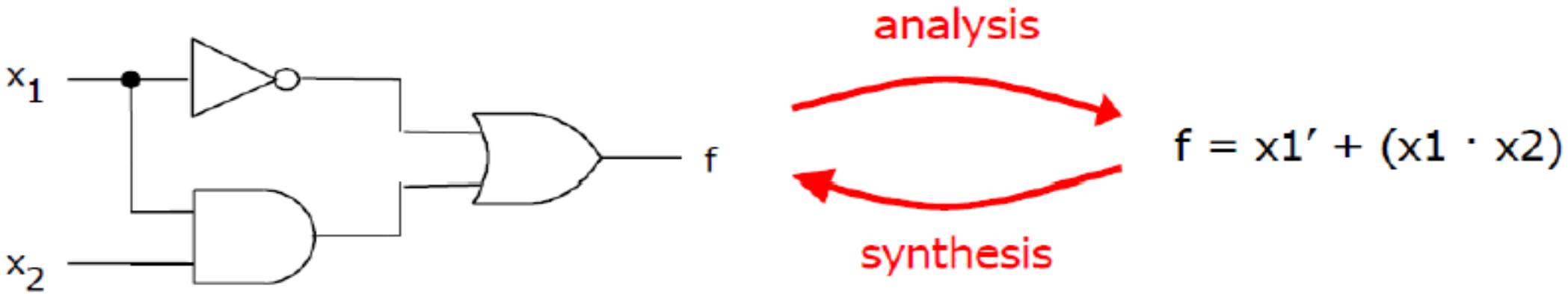

S
x1
S
x2
S
x3
L=0
Power
Supply


x1
x2
x3
f = (x1 + x2) · x3

Logic Gates and Networks

Digital System Designer Concern

- Digital system designer faced with two basic issues
  - Determine function of an existing logic network → **Analysis**
  - Designing a logic network to implement the desired function → **Synthesis**

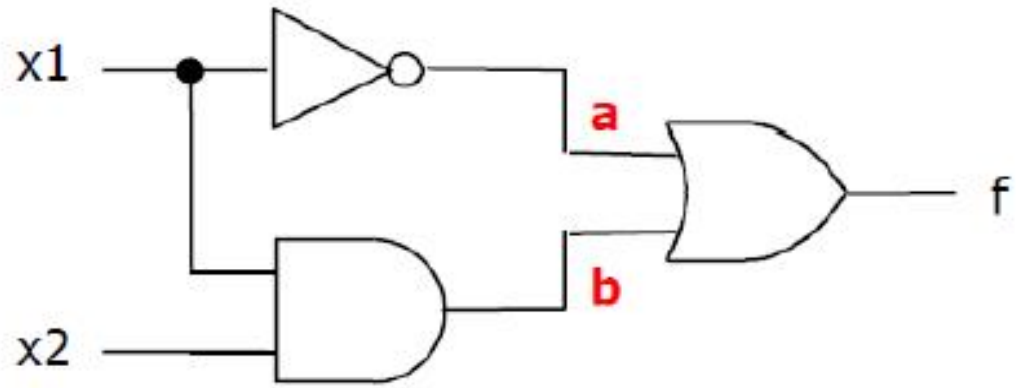


STEP 1 :
ANALYSING
LOGIC CIRCUITS

Logic Gates and Networks

Analysis of Logic Network

- Convert logic network into logic expression



f =

f = a + b

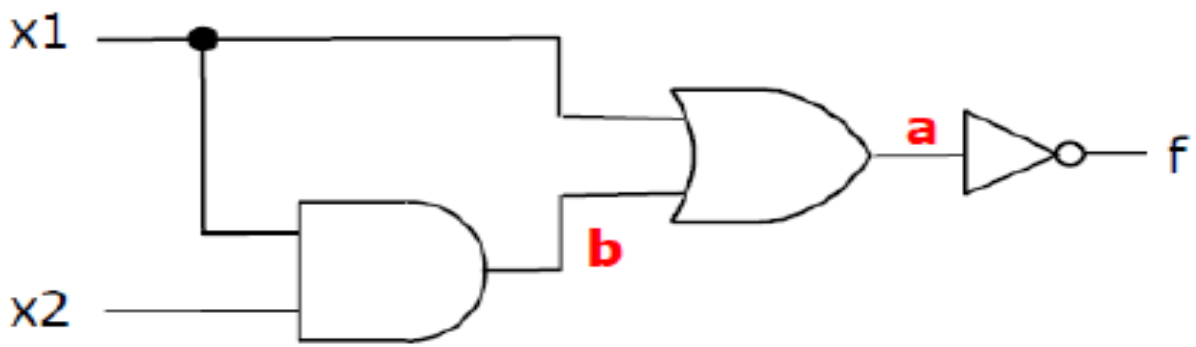
f = x1′ + b

f = x1′ + (x1 · x2)

Logic Gates and Networks

Analysis of Logic Network

- Convert logic network into logic expression



f =

f = a´

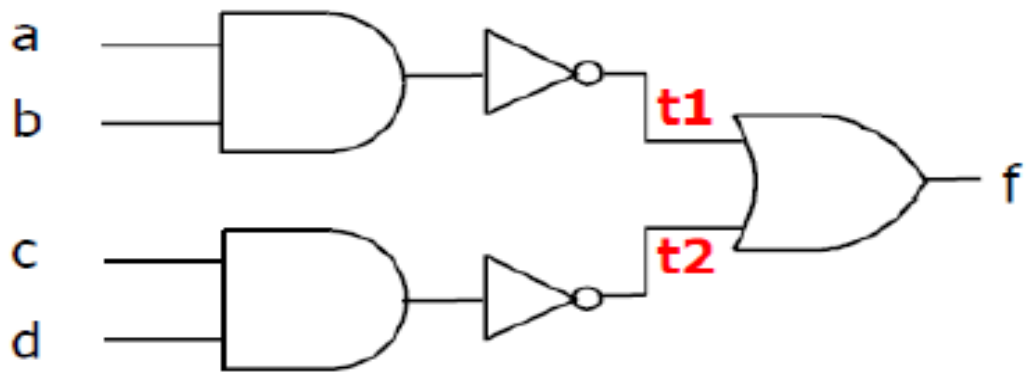
f = (x1 + b)´

f = (x1 + (x1 · x2))´

Logic Gates and Networks

Analysis of Logic Network

- Convert logic network into logic expression



$f =$

$f = t1 + t2$

$f = (a \cdot b)' + t2$

$f = (a \cdot b)' + (c \cdot d)'$

TO TEST CIRCUIT FUNTIONALITY

1. TRUTH TABLE

Truth Table

Introduction

Left Side:
All possible combinations for input values

No of combinations = 2^n
where n = No of inputs

x1	x2	x1·x2	x1+x2
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Right Side:
Values for outputs

Truth table for AND Operation

Truth table for OR Operation

Truth Table

Introduction

- Advantages
  - Only one truth table
  - Intuitive to read
- Disadvantages
  - Size explosion

a	b	F
0	0	0
0	1	0
1	0	1
1	1	1

2-input truth table

a	b	c	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

3-input truth table

a	b	c	d	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

4-input truth table

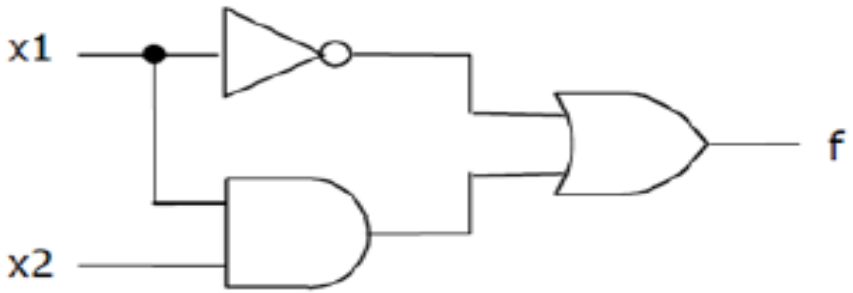
How many lines are needed if you have 8 inputs?

5-input truth table?

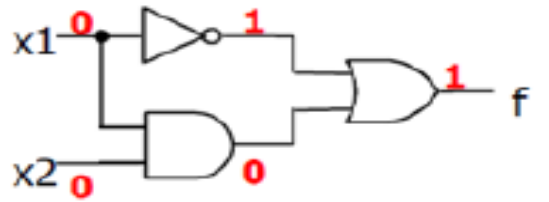
Logic Gates and Networks

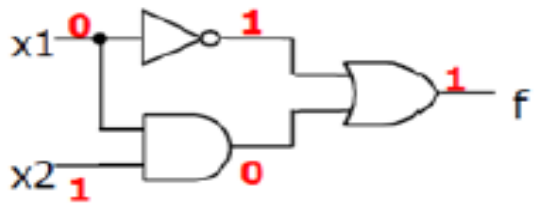
Analysis of Logic Network

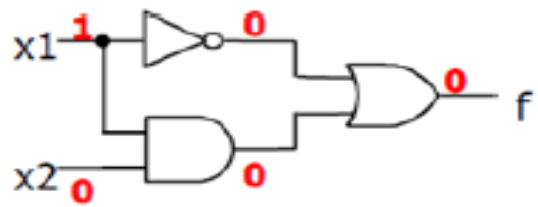
- Another way to specify circuit functionality – truth table
  - Consider what happens to output f for all possible inputs

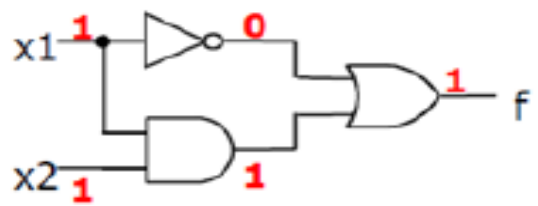


$f = x1' + (x1 \cdot x2)$



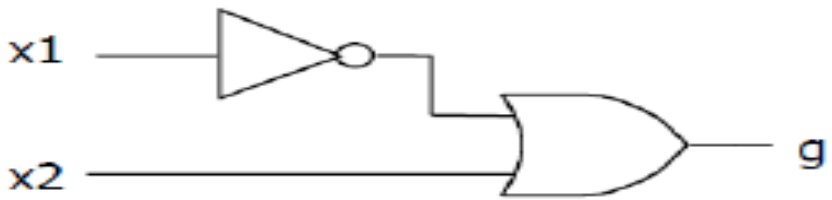






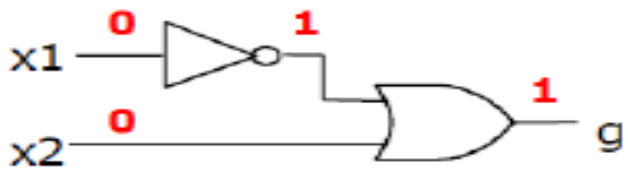
x1	x2	F
0	0	1
0	1	1
1	0	0
1	1	1

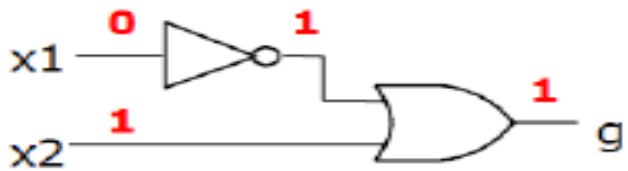
Example 2 :

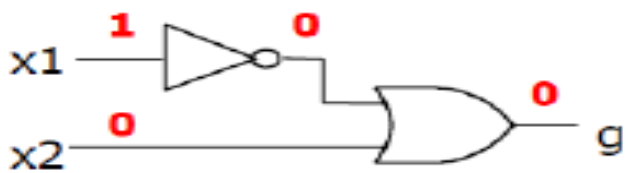


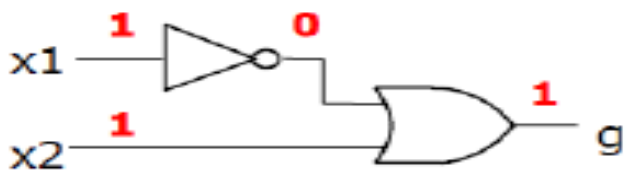
$g = x1' + x2$

x1	x2	G
0	0	1
0	1	1
1	0	0
1	1	1





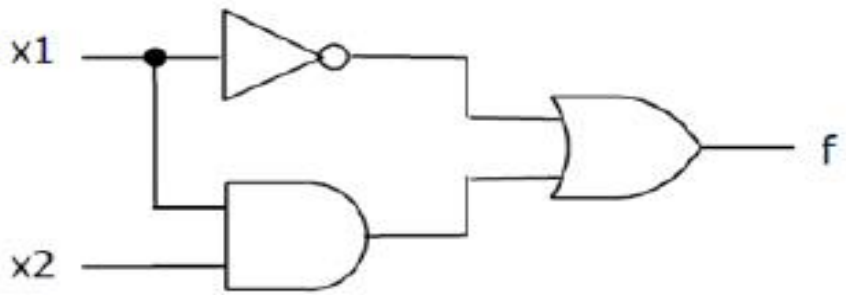




Logic Gates and Networks

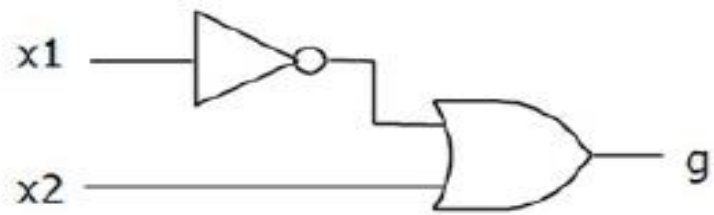
Functional Equivalence

- You may have noticed we got the same truth table for both examples
  - Logic circuits are functionally equivalent
- Logic function can be implemented in a variety of ways
  - Designer wants to use the simpler one – lower cost
  - How do we determine the best implementation?
    - Manipulate function using a set of rules (Boolean Algebra)
    - Other techniques discussed in Chapter 4



$f = x1' + (x1 \cdot x2)$

x1	x2	F
0	0	1
0	1	1
1	0	0
1	1	1



$g = x1' + x2$

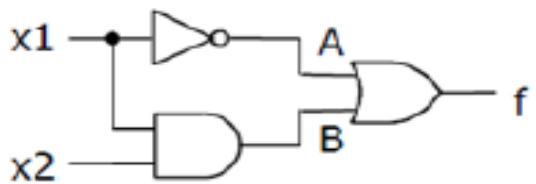
x1	x2	G
0	0	1
0	1	1
1	0	0
1	1	1

2.Timing diagram.

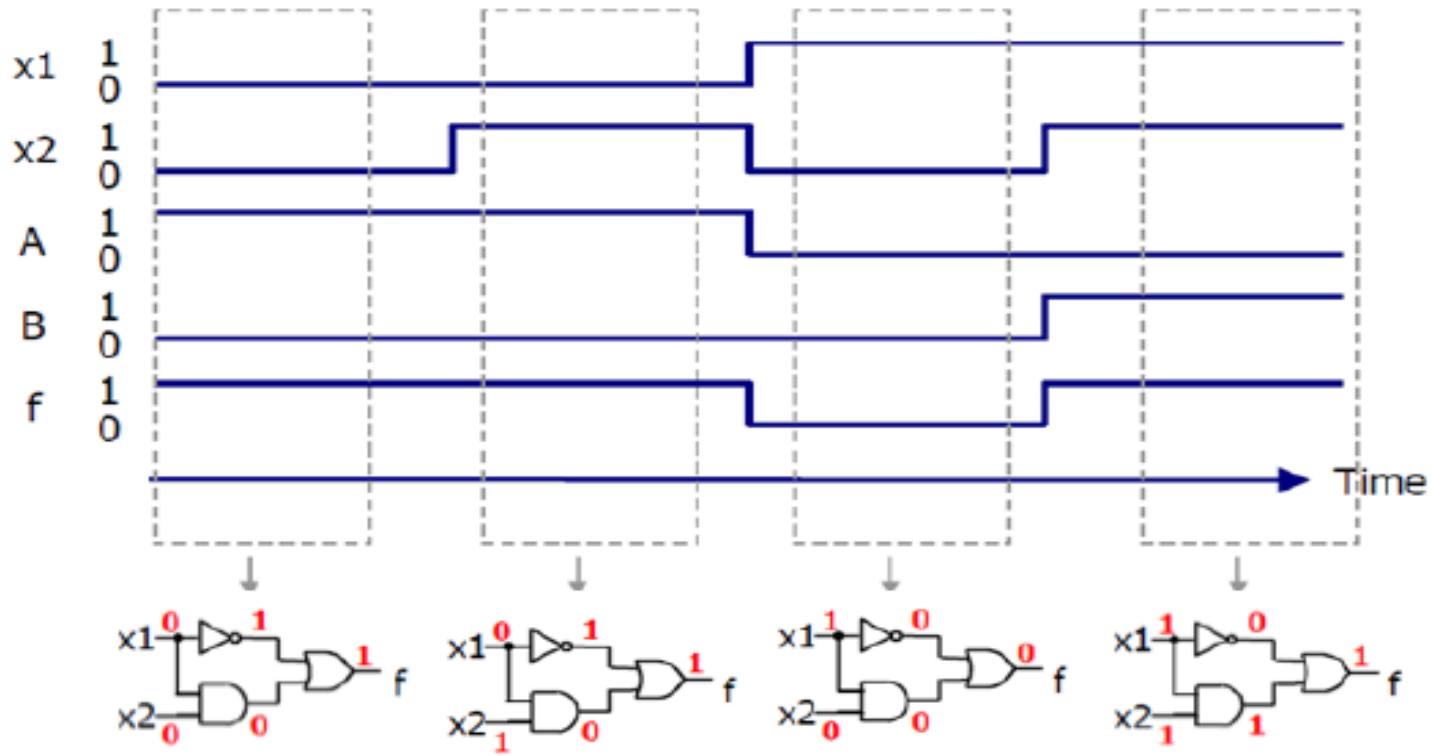
Logic Gates and Networks

Timing Diagram

- Yet another way to describe logic circuit behaviour – timing diagram
  - Time runs from left to right
  - Each input value is held for a fixed period
  - Waveform shown indicating inputs, output, and internal signal values



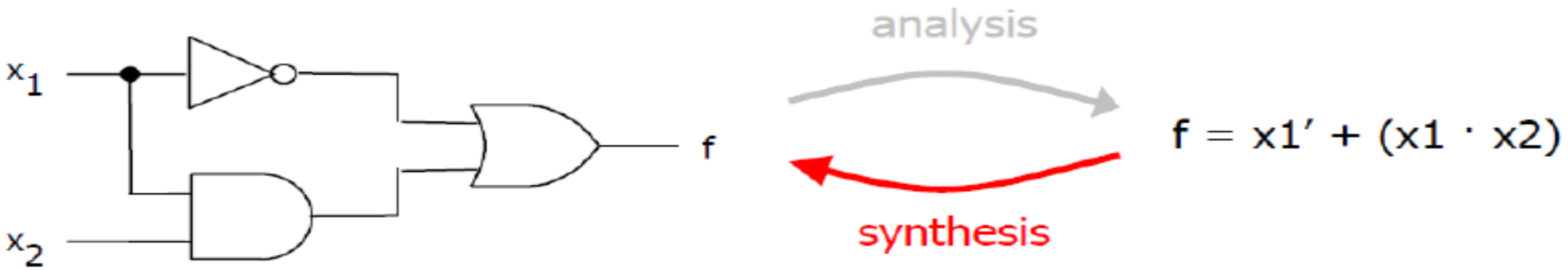
x1	x2	F
0	0	1
0	1	1
1	0	0
1	1	1



Logic Gates and Networks

Synthesis of Logic Network

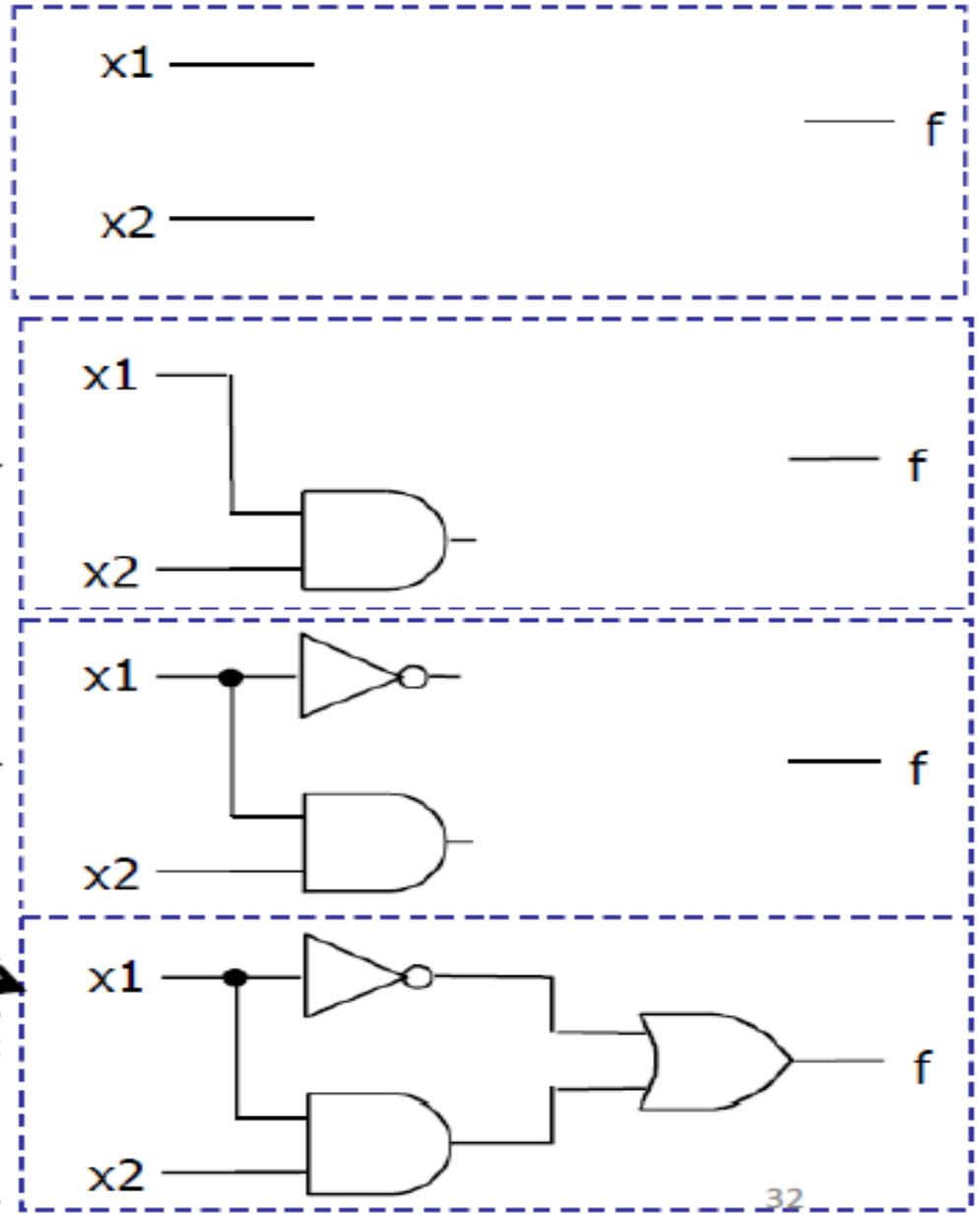
- Second design issue – Synthesis
  - How do I go from a logic expression to a logic circuit?



Logic Gates and Networks

Synthesis of Logic Network

- Convert f= x1´ + (x1 · x2) to a logic circuit
  - What are your inputs?
    - x1, x2
  - What are/is your output(s)?
    - F
  - How is the function evaluated?
    - Parentheses
    - NOT operation
    - OR operation

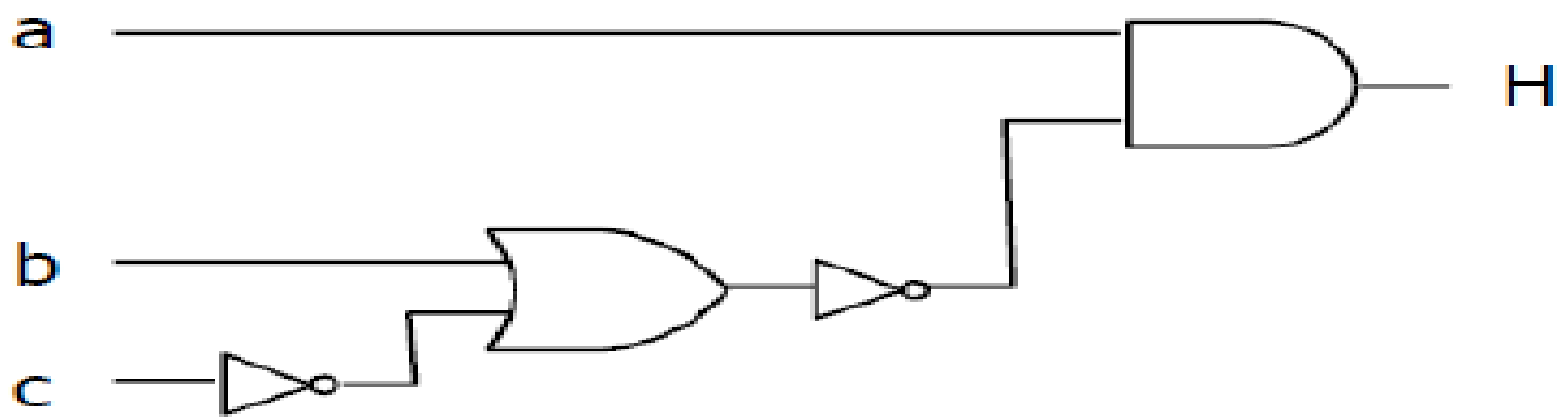


Symbol	Name	Description
( )	Parentheses	Evaluate expression nested in parentheses first
´	NOT	Evaluate from left to right
·	AND	Evaluate from left to right
+	OR	Evaluate from left to right

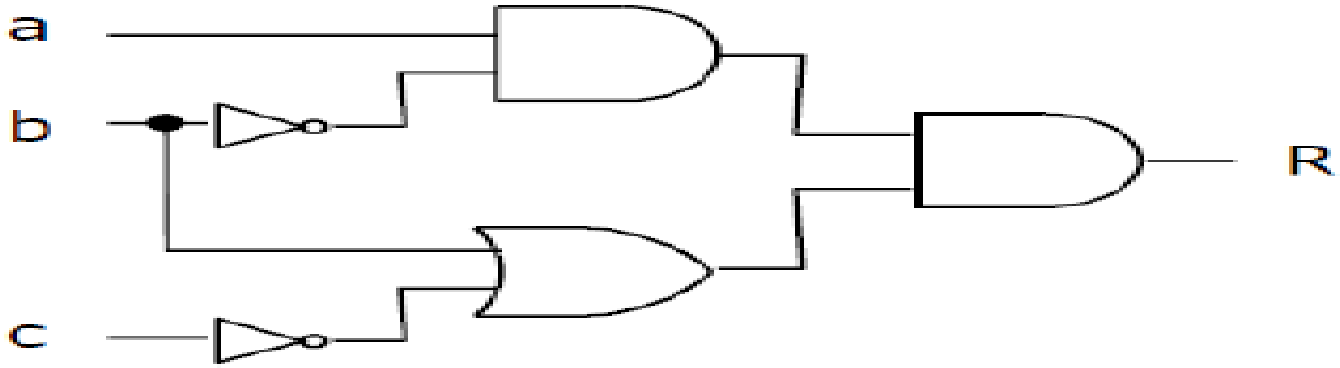
Convert $H = a \cdot (b + c')'$ to a logic circuit

Convert $R = (a \cdot b') \cdot (b + c')$ to a logic circuit

- Convert H = a · (b + c´)´ to a logic circuit



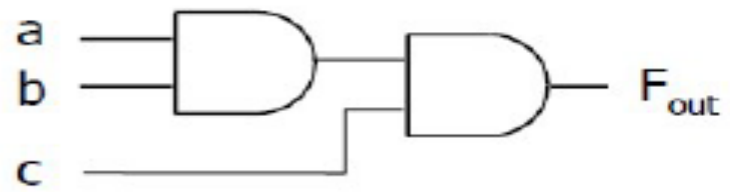
- Convert R = (a · b´) · (b + c´) to a logic circuit

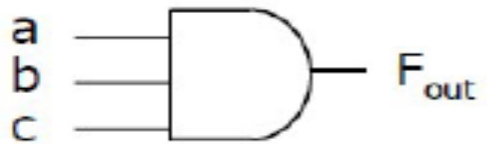


Logic Gates and Networks

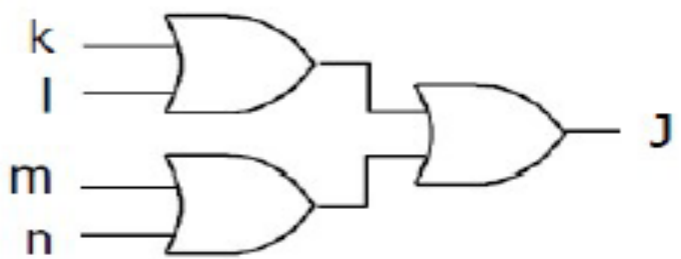
Synthesis of Logic Network

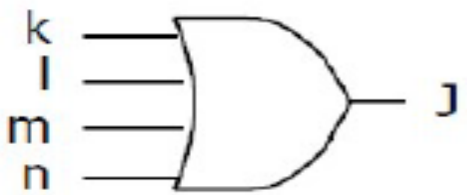
- Convert Fout = (a · b · c) to a logic circuit
  - How do we implement a 3-input gate?
    - Using 2-input AND gates
    - Using 3-input AND gates





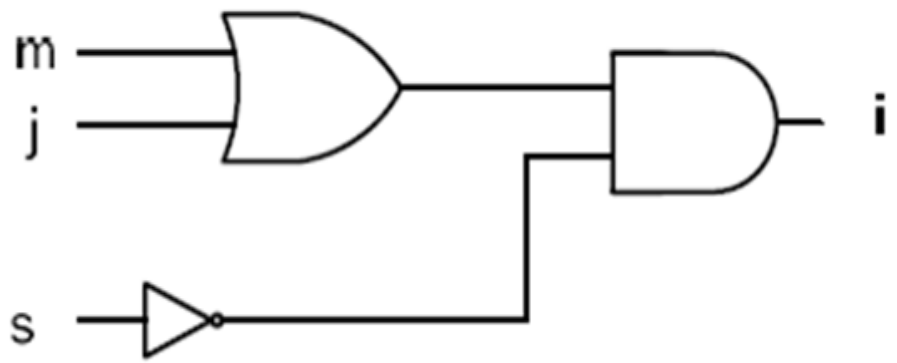
- Same for OR gates
- J = k + l + m + n





Boolean Algebra

- Everyday Boolean Logic:
  - I will **(i)** go to lunch if Mohammed **(m)** goes OR Jawad **(j)** goes, AND Sami **(s)** does NOT go.
    - i -> will I go?
    - m -> Does Mohammed go?
    - j -> Does Jawad go?
    - s -> Does Sami go?



$$i = (m + j) \cdot (!s)$$

m	j	s	i
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Boolean Algebra : Single Variable Theorems

- We can derive single variable theorems (rules)
  - Assume x is a variable

AND

a	x	F
0	0	0
0	1	0
1	0	0
1	1	1

OR

a	x	F
0	0	0
0	1	1
1	0	1
1	1	1

AND

a	x	F
0	0	0
0	1	0
1	0	0
1	1	1

AND

a	x	F
0	0	0
0	1	0
1	0	0
1	1	1

5a. $x \cdot 0 = 0$ (Null Elements)
5b. $x + 1 = 1$

6a. $x \cdot 1 = x$ (Identity)
6b. $x + 0 = x$

7a. $x \cdot x = x$ (Idempotent)
7b. $x + x = x$

8a. $x \cdot x' = 0$ (Complement)
8b. $x + x' = 1$

9a. $x'' = x$ (involution)

Boolean Algebra : Duality

- **Principle of duality**
  - Given a logic expression, its dual is obtained by replacing all "·" operators with "+" operators and all 0's with 1's
  - The dual of any true statement is also true
  - Axioms and theorem listed in pairs to illustrate duality

1a. $0 \cdot 0 = 0$
1b. $1 + 1 = 1$
2a. $1 \cdot 1 = 1$
2b. $0 + 0 = 0$
3a. $0 \cdot 1 = 1 \cdot 0 = 0$
3b. $0 + 1 = 1 + 0 = 1$
4a. If $x = 0$, then $x' = 1$
4b. If $x = 1$, then $x' = 0$

5a. $x \cdot 0 = 0$
5b. $x + 1 = 1$
6a. $x \cdot 1 = x$
6b. $x + 0 = x$
7a. $x \cdot x = x$
7b. $x + x = x$
8a. $x \cdot x' = 0$
8b. $x + x' = 1$
9. $x'' = x$

Boolean Algebra : Properties

- 2- and 3-variable identities – Properties
  - x, y, and z are variables

10a. $x \cdot y = y \cdot x$ *(Commutative)*
10b. $x + y = y + x$
11a. $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ *(Associative)*
11b. $x + (y + z) = (x + y) + z$
12a. $x \cdot (y + z) = x \cdot y + x \cdot z$ *(Distributive)*
12b. $x + (y \cdot z) = (x + y) \cdot (x + z)$ *this one is tricky!*
13a. $x + x \cdot y = x$ *(Absorption)*
13b. $x \cdot (x + y) = x$
14a. $x \cdot y + x \cdot y' = x$ *(Combining)*
14b. $(x + y) \cdot (x + y') = x$
15a. $(x \cdot y)' = x' + y'$ *(DeMorgan's Theorem)*
15b. $(x + y)' = x' \cdot y'$
16a. $x + x' \cdot y = x + y$
16b. $x \cdot (x' + y) = x \cdot y$

Boolean Algebra: Properties

- How can we prove these properties?
  - Truth table
  - Algebraic Manipulation
- Show DeMorgan's Theorem works using a truth table
  - $(x \cdot y)' = x' + y'$

LHS **RHS**

x	y	$(x \cdot y)$	$(x \cdot y)'$	x'	y'	$x' + y'$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Truth tables produce same result – functions are equivalent!

Boolean Algebra: Algebraic Manipulations Example

- Prove $(x1 + x3) \cdot (x1' + x3') = x1 \cdot x3' + x1' \cdot x3$ using algebraic manipulation

$LHS = (x1 + x3) \cdot (x1' + x3')$

Use Distributive property 12a - $x \cdot (y + z) = x \cdot y + x \cdot z$

$LHS = (x1 + x3) \cdot x1' + (x1 + x3) \cdot x3'$

Use Distributive property again

$LHS = (x1 \cdot x1') + (x3 \cdot x1') + (x1 \cdot x3') + (x3 \cdot x3')$

Use complement property 8a - $x \cdot x' = 0$

$LHS = 0 + (x3 \cdot x1') + (x1 \cdot x3') + 0$

Use Identity 6b - $x + 0 = x$

$LHS = (x3 \cdot x1') + (x1 \cdot x3')$

Use commutative property 10a - $x \cdot y = y \cdot x$
and 10b - $x + y = y + x$

$LHS = (x1 \cdot x3') + (x1' \cdot x3)$

LHS matches RHS of the initial equation

Practice

Example 2.4 Consider the logic equation

$$x_1 \cdot \bar{x}_3 + \bar{x}_2 \cdot \bar{x}_3 + x_1 \cdot x_3 + \bar{x}_2 \cdot x_3 = \bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot x_2 + x_1 \cdot \bar{x}_2$$

The left-hand side can be manipulated as follows

$$\begin{aligned} \text{LHS} &= x_1 \cdot \bar{x}_3 + x_1 \cdot x_3 + \bar{x}_2 \cdot \bar{x}_3 + \bar{x}_2 \cdot x_3 && \text{using } 10b \\ &= x_1 \cdot (\bar{x}_3 + x_3) + \bar{x}_2 \cdot (\bar{x}_3 + x_3) && \text{using } 12a \\ &= x_1 \cdot 1 + \bar{x}_2 \cdot 1 && \text{using } 8b \\ &= x_1 + \bar{x}_2 && \text{using } 6a \end{aligned}$$

The right-hand side can be manipulated as

$$\begin{aligned} \text{RHS} &= \bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot (x_2 + \bar{x}_2) && \text{using } 12a \\ &= \bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot 1 && \text{using } 8b \\ &= \bar{x}_1 \cdot \bar{x}_2 + x_1 && \text{using } 6a \\ &= x_1 + \bar{x}_1 \cdot \bar{x}_2 && \text{using } 10b \\ &= x_1 + \bar{x}_2 && \text{using } 16a \end{aligned}$$

16a. $x + x' \cdot y = x + y$