Chapter 1
Boolean Algebra
and
Combinational Logic

This chapter introduces to the idea of digitally representing analog quantities and goes step by step through the main concepts of the Boolean algebra: variables, functions, truth tables, operations, and properties. The chapter is quite detailed and accompanied by many examples and exercises in order to provide a precise framework of the fundamentals of digital design. It includes the theorems which constitute the foundation for the application of the Boolean algebra to logic networks, with a precise focus on their application for combinational network design.

INTRODUCTORY CONCEPTS

Analog and Discrete Variables
模拟与离散变量

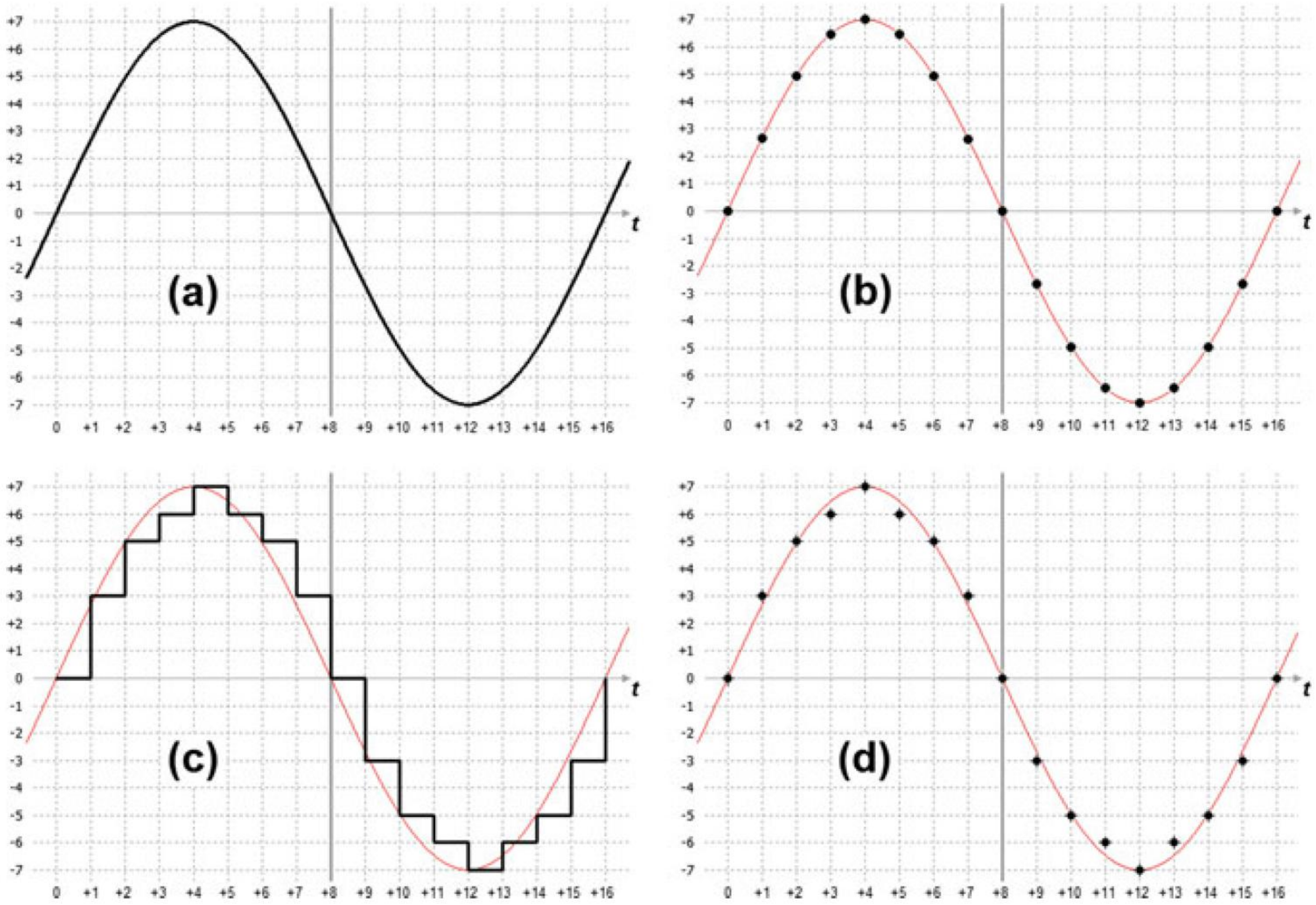
Information is a reduction of uncertainty

- information is associated with “before” and “after,” in relation to an event having a probability to happen
- information can be conveyed through the employment of physical quantities variables, changing in time or space.
- To the aim of studying information processing, we will not refer directly to a physical quantity variable, but rather to its numeric representation, indicated with “G,” and its variation over time, indicated with “T.”

Analog and Digital

- If G can vary continuously between a value and another one, assuming all the infinite intermediate values, we are employing an "***analog***" representation.
- If G can assume only a limited number of values, we are employing a "discrete" (or "***digital***") representation, becoming "binary" (or "Boolean") if the numbering system uses only two symbols.
- Analog to Digital Converters (ADC, 模数转换器)
- Digital to Analog Converters (DAC, 数模转换器)
- Analog → math, Digital → discrete math (离散数学), When limit G to two values, i.e., the binary case → Boolean algebra (布尔代数)
- Binary variables: {0, 1}, {−1,+1}, {L, H} (Low and High) or {T, F} (True or False)

Analog and Digital



a) continuous quantity variable changing over time;
b) continuous quantity variable sampled over time;
c) quantity variable quantized in amplitude and continuous over time;
d) quantity variable quantized in amplitude and sampled over time.

Asynchronous & Synchronous (异步和同步)

- In the digital field, we will make use of both the representation over continuous time, called "***asynchronous***," and the one over discrete time, called "***synchronous***."
- A logic network is called ***synchronous*** if its parts operate simultaneously, according to a common synchronization signal (***clock***);
- It is instead defined as ***asynchronous*** if its parts operate in an autonomous mode among each other.

Advantages and disadvantages of Digital

- an analog value is a pure mathematical abstraction with **infinite *accuracy***
- a binary is easily storable with two of the physical values
- binary variables are ***less sensitive*** to a possible damaging
- the accuracy of the system can be easily controlled by choosing the number of bits that code the information.
- devices processing digital information, namely digital systems, are simpler to design, though the practical realization requires a higher number of circuital components

Boolean Variables and Functions

- Let X be a certain discrete variable. We will call Boolean variable any discrete variable that can assume only ***two values***. These values are denoted as follows:
  - X = 0 false
  - X = 1 true
- If we have the Boolean variables X1, X2, . . ., Xn, f (X1, X2, . . . , Xn) is called a Boolean function. It can assume only the values 0 and 1. This function associates a Boolean value to every element in its domain. The domain of a function of n-variables is composed of all the 2^n combinations of their values. Therefore, domain's elements are ***countable***. Two functions are equivalent if they assume the same value for any combination of their variables' values.

Truth Tables （真值表）

- A function can be represented in the truth table.
- For a three-variable function X1, X2, X3, we can construct a table with all the values assumed by f.

X_1	X_2	X_3	$f(X_1, X_2, X_3)$
0	0	0	$f(0,0,0)$
0	0	1	$f(0,0,1)$
0	1	0	$f(0,1,0)$
0	1	1	$f(0,1,1)$
1	0	0	$f(1,0,0)$
1	0	1	$f(1,0,1)$
1	1	0	$f(1,1,0)$
1	1	1	$f(1,1,1)$

Introductory Concepts

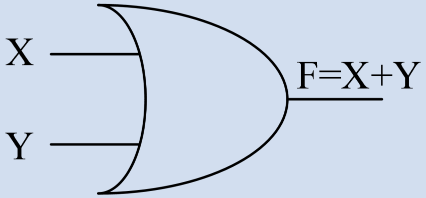
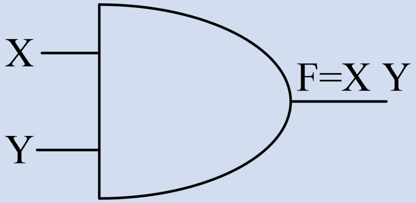
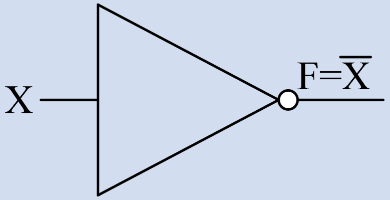
Boolean Algebra
布尔代数

Definition

Boolean algebra provides the necessary tools to calculate and interpret information presented in binary form. Boolean algebra is an algebraic system (a set of elements to which a set of operations is associated), defined by:

- The set of values {0,1};
- The operations OR, AND, and NOT;
- The equivalence operator “=”, along with the properties reflexive, symmetric, and transitive.

Operation

Operation	OR (logical sum)	AND (logical product)	NOT (negation)
Algebraic symbols	$X + Y$ $X \cup Y$ $X \vee Y$ $X \; or \; Y$	$X \cdot Y = X \;\; Y$ $X \cap Y$ $X \wedge Y$ $X \; and \; Y$	$\overline{X}$ $-X$ $!\,X$ $not(X)$
Truth table	X Y X+Y 0 0 0 0 1 1 1 0 1 1 1 1	X Y X Y 0 0 0 0 1 0 1 0 0 1 1 1	X $\overline{X}$ 0 1 1 0
Circuit diagram symbols			

in VHDL (AND gate, OR gate)

```
library ieee;
use ieee.std_logic_1164.all;

ENTITY AND2_gate IS
    PORT( I0, I1: IN std_logic;
              O: OUT std_logic );
END AND2_gate;

ARCHITECTURE behavioral OF AND2_gate IS
BEGIN
    O <= (I0 and I1);
END behavioral;
```

```
library ieee;
use ieee.std_logic_1164.all;

ENTITY OR2_gate IS
    PORT( I0, I1: IN std_logic;
              O: OUT std_logic );
END OR2_gate;

ARCHITECTURE behavioral OF OR2_gate IS
BEGIN
    O <= (I0 or I1);
END behavioral;
```

The Fundamental Properties

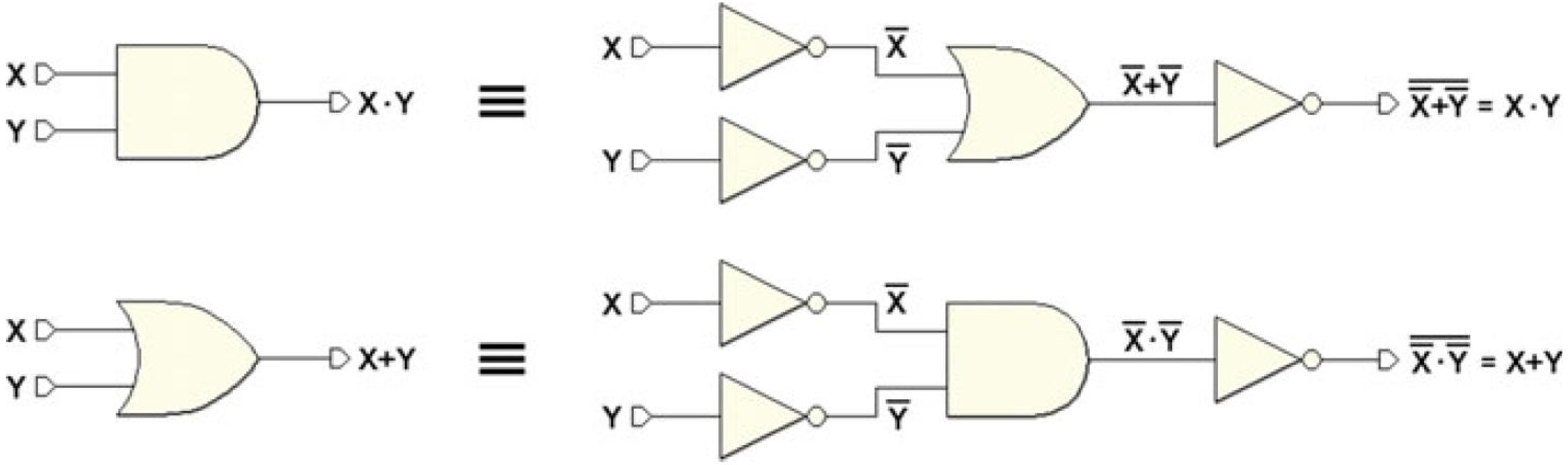
- Conventions (惯例)
  - AND is prioritized over OR (e.g., $A + B\,C = A + (B\,C)$)
- Duality Principle (对偶原理)
  If a given expression is valid, its dual expression is also valid. The dual expression is obtained by switching the OR with the AND and the 0 constants with the 1 constants from the original expression.
  - $X + 1 = 1 \rightarrow X \bullet 0 = 0$; $X + 0 = X \rightarrow X \bullet 1 = X$
- Idempotent Law (幂等定律):
  - $X + X = X$
  - $X \bullet X = X$

The Fundamental Properties

- Commutative Law (交換定律)
  - $X + Y = Y + X$
  - $X\,Y = Y\,X$
- Associative Law (結合定律)
  - $(X + Y) + Z = X + (Y + Z) = X + Y + Z$
  - $(X\,Y)\,Z = X\,(Y\,Z) = X\,Y\,Z$
- Distributivity (分配性)
  - $(X + Y)\,(X + Z) = X + \;Y\,Z$
  - $X\,Y + X\,Z = X\,(Y + Z)$
- Complementation (互补)
  - $X + \overline{X} \; = 1$
  - $X\,\overline{X} \; = 0$
- Absorption Law (吸收定律)
  - $X + X\,Y = X$
  - $X + \overline{X}\,Y = X + Y$
  - $X\,(X + Y) = X$
  - $X\,(\overline{X} + Y) = X\,Y$
- Logic Adjacency (逻辑相邻)
  - $Y\,X + Y\,\overline{X} = Y$
  - $(Y + X)\,(Y + \overline{X}) = Y$
- Consensus (一致性)
  - $X\,Y + Y\,Z + Z\,\overline{X} = X\,Y + Z\,\overline{X}$
  - $(X + Y)\,(Y + Z)\,(Z + \overline{X}) = (X + Y)\,(Z + \overline{X})$
- Involution (内卷)
  - $\overline{\overline{X}} = X$

The Fundamental Properties

- De Morgan's Theorem (摩根定理)
  - $X \cdot Y = \overline{\overline{X} + \overline{Y}}$
  - $X + Y = \overline{\overline{X} \cdot \overline{Y}}$
- Generalized De Morgan's Theorem
  - $X_1 X_2 \cdots X_n = \overline{\overline{X_1} + \overline{X_2} + \cdots + \overline{X_n}}$
  - $X_1 + X_2 + \cdots + X_n = \overline{\overline{X_1} \cdot \overline{X_2} \cdot \cdots \cdot \overline{X_n}}$



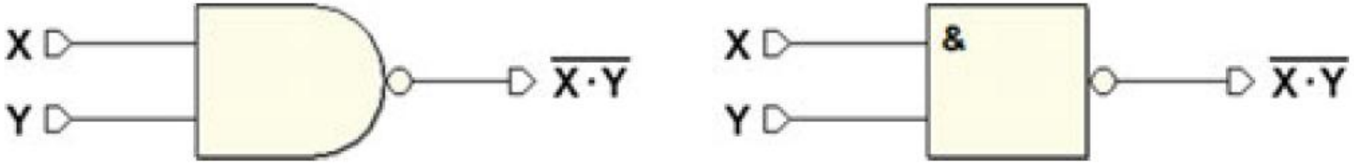
Other Operations

- NAND

$X \text{ nand } Y = \overline{XY}$

X	Y	$(X \text{ nand } Y)$
0	0	1
0	1	1
1	0	1
1	1	0

Circuital symbols:

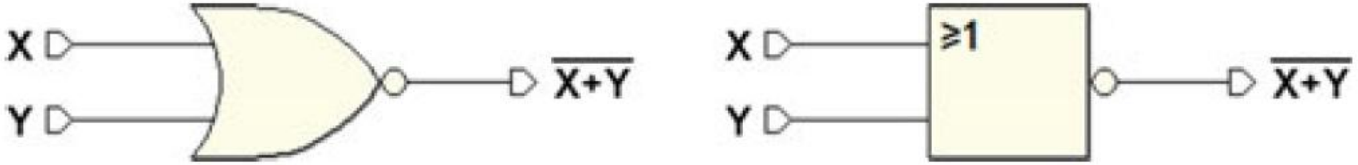


- NOR

$X \text{ nor } Y = \overline{X + Y}$

X	Y	$(X \text{ nor } Y)$
0	0	1
0	1	0
1	0	0
1	1	0

Circuital symbols:



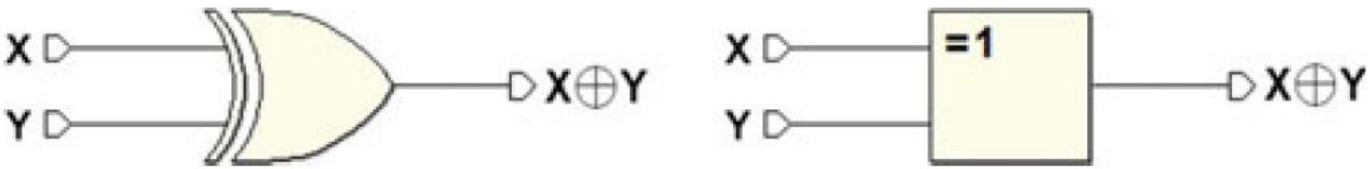
XOR (Exclusive OR)

$$X\ xor\ Y = X \oplus Y = X\overline{Y} + \overline{X}Y$$
$$X\ xnor\ Y = \overline{X \oplus Y} = XY + \overline{X}\,\overline{Y}$$

$$X_1 \oplus X_2 \oplus \cdots \oplus X_n = \begin{cases} 1\ if\ there\ is\ an\ odd\ number\ of\ inputs = 1 \\ 0\ if\ there\ is\ an\ even\ number\ of\ inputs = 1 \end{cases}$$

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

Circuital symbols:

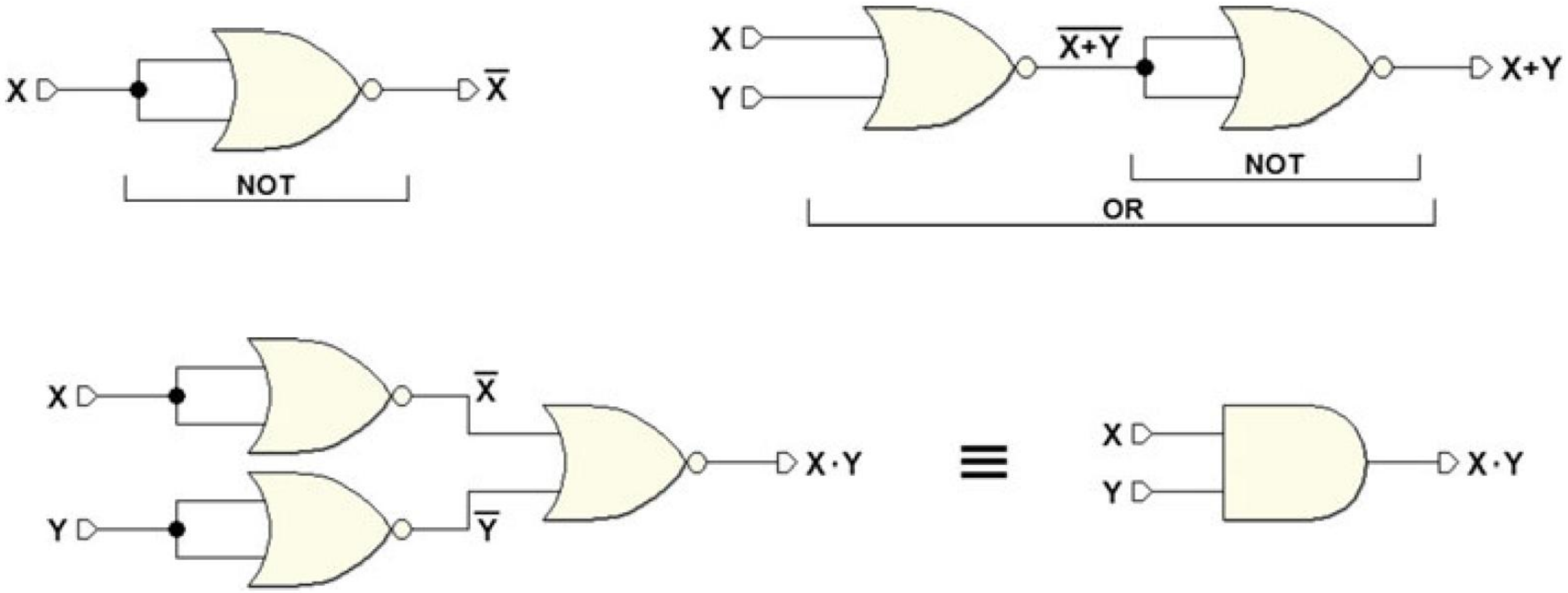


Functionally Complete Operation Sets

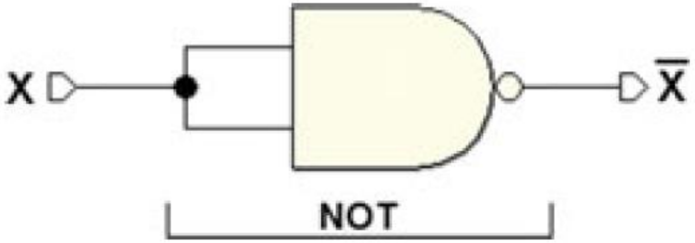
- {AND, OR, NOT}
- {NOR}
- {NAND}
- {OR, NOT}
- {AND, NOT}
- {EXOR, AND}
- {EXOR, OR}

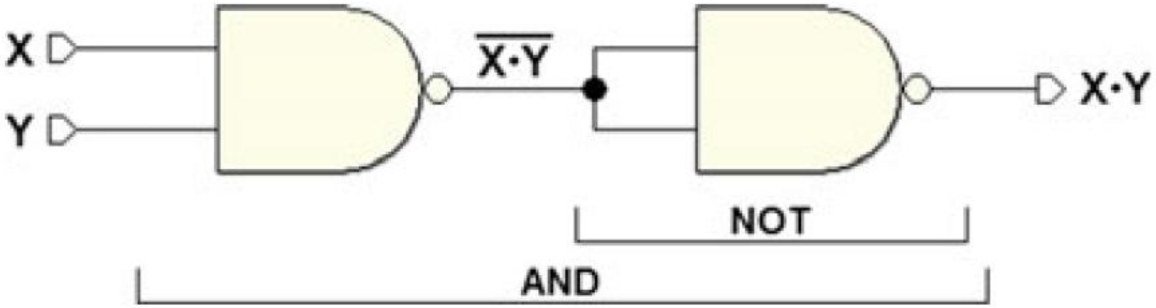
- these are a sufficient basis to construct all of Boolean algebra.
- in practice, only {NOR} and {NAND} sets are used.

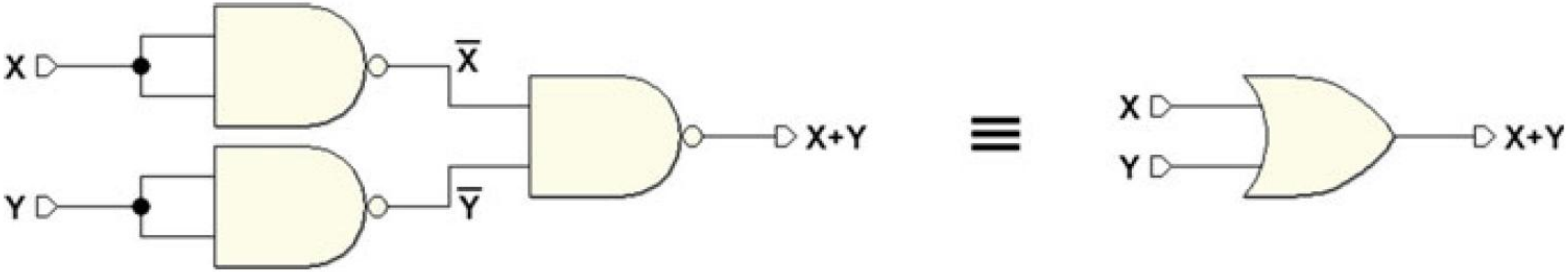
{NOR} Set


X
$\overline{X}$
NOT
X
Y
$\overline{X+Y}$
X+Y
NOT
OR
X
Y
$\overline{X}$
$\overline{Y}$
X·Y
≡
X
Y
X·Y

{NAND} Set


X
$\overline{X}$
NOT


X
Y
$\overline{X \cdot Y}$
$X \cdot Y$
NOT
AND


X
Y
$\overline{X}$
$\overline{Y}$
X+Y
≡
X
Y
X+Y

Assignment 1-1 (p.27: 1, 2, 3)

1. Negate the following term and transform it into a four-term logical sum.
   $\overline{A}\ \mathrm{B}\ \overline{C}\ \mathrm{D}$
2. Negate the following term and then transform it into a single product term.
   $\overline{A} + \overline{B} + \mathrm{C}$
3. Using the theorems of Boolean algebra, minimize the following logical expression:
   $A + AB + CB + C\overline{B}$

Shannon's Expansion Theorem - First Form

- ***sum of products (SOP)***, or first canonical form, or AND–OR form

  $$f(X_1, X_2, \cdots, X_n) = X_1 \cdot f(1, X_2, \cdots, X_n) + \overline{X_1} f(0, X_2, \cdots, X_n)$$
  $$= X_1 X_2 \cdot f(1,1, X_3, \cdots, X_n) + \overline{X_1} X_2 f(0,1, X_3, \cdots, X_n)$$
  $$+ X_1 \overline{X_2} \cdot f(1,0, X_3, \cdots, X_n) + \overline{X_1}\,\overline{X_2} \cdot f(0,0, X_3, \cdots, X_n)$$
  $$= \cdots$$

- Example

  $$f(A, B, C) = A\,B\,C \cdot f(1,1,1) + A\,B\,\overline{C} \cdot f(1,1,0)$$
  $$+ A\,\overline{B}\,C \cdot f(1,0,1) + A\,\overline{B}\,\overline{C} \cdot f(1,0,0)$$
  $$+ \overline{A}\,B\,C \cdot f(0,1,1) + \overline{A}\,B\,\overline{C} \cdot f(0,1,0)$$
  $$+ \overline{A}\,\overline{B}\,C \cdot f(0,0,1) + \overline{A}\,\overline{B}\,\overline{C} \cdot f(0,0,0)$$

Example

$$f(A,B,C) = A\,B\,C\cdot f(1,1,1) + A\,B\,\overline{C}\cdot f(1,1,0)$$
$$+ A\,\overline{B}\,C\cdot f(1,0,1) + A\,\overline{B}\,\overline{C}\cdot f(1,0,0)$$
$$+ \overline{A}\,B\,C\cdot f(0,1,1) + \overline{A}\,B\,\overline{C}\cdot f(0,1,0)$$
$$+ \overline{A}\,\overline{B}\,C\cdot f(0,0,1) + \overline{A}\,\overline{B}\,\overline{C}\cdot f(0,0,0)$$

$$= A\,B\,C\cdot 0 + A\,B\,\overline{C}\cdot 1 + A\,\overline{B}\,C\cdot 0$$
$$+ A\,\overline{B}\,\overline{C}\cdot 1 + \overline{A}\,B\,C\cdot 0 + \overline{A}\,B\,\overline{C}\cdot 0$$
$$+ \overline{A}\,\overline{B}\,C\cdot 1 + \overline{A}\,\overline{B}\,\overline{C}\cdot 0$$

$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$	$\boldsymbol{f(A,B,C)}$
0	0	0	$\boldsymbol{f}(0,0,0) = 0$
0	0	1	$\boldsymbol{f}(0,0,1) = 1$
0	1	0	$\boldsymbol{f}(0,1,0) = 0$
0	1	1	$\boldsymbol{f}(0,1,1) = 0$
1	0	0	$\boldsymbol{f}(1,0,0) = 1$
1	0	1	$\boldsymbol{f}(1,0,1) = 0$
1	1	0	$\boldsymbol{f}(1,1,0) = 1$
1	1	1	$\boldsymbol{f}(1,1,1) = 0$

Shannon's Expansion Theorem - Second Form

- ***product of sums (POS)***, or second canonical form, or OR-AND form

$$f(X_1, X_2, \cdots, X_n) = (X_1 + f(0, X_2, \cdots, X_n)) \cdot (\overline{X_1} + f(1, X_2, \cdots, X_n))$$
$$= (X_1 + X_2 + f(0,0, X_3, \cdots, X_n)) \cdot (\overline{X_1} + X_2 + f(1,0, X_3, \cdots, X_n))$$
$$\cdot (X_1 + \overline{X_2} + f(0,1, X_3, \cdots, X_n)) \cdot (\overline{X_1} + \overline{X_2} + f(0,0, X_3, \cdots, X_n))$$
$$= \cdots$$

- Example

$$f(A,B,C) = (A + B + C + f(0,0,0) \cdot (A + B + \overline{C} + f(0,0,1))$$
$$\cdot (A + \overline{B} + C + f(0,1,0)) \cdot (A + \overline{B} + \overline{C} + f(0,1,1))$$
$$\cdot (\overline{A} + B + C + f(1,0,0)) \cdot (\overline{A} + B + \overline{C} + f(1,0,1))$$
$$\cdot (\overline{A} + \overline{B} + C + f(1,1,0)) \cdot (\overline{A} + \overline{B} + \overline{C} + f(1,1,1))$$

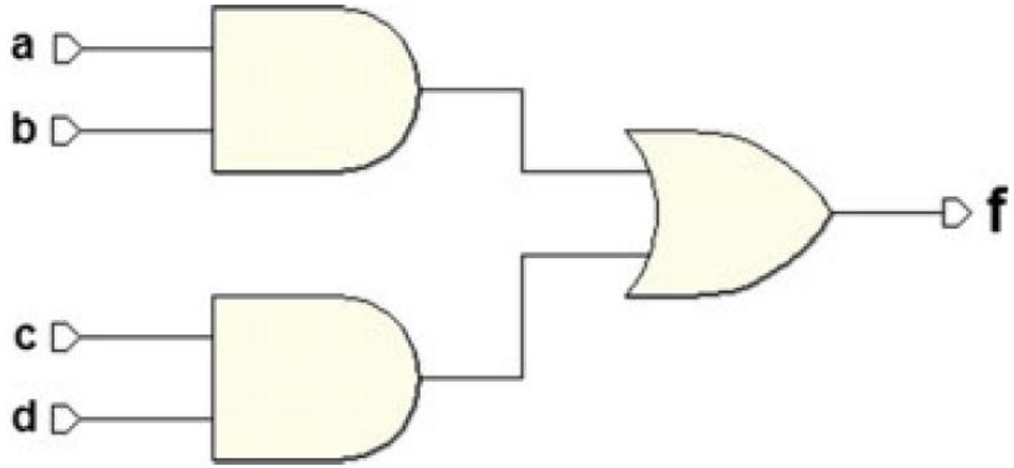
Example

$$
\begin{aligned}
f(A, B, C) = (A + B + C + f(0,0,0) \\
\cdot (A + B + \overline{C} + f(0,0,1)) \\
\cdot (A + \overline{B} + C + f(0,1,0)) \\
\cdot (A + \overline{B} + \overline{C} + f(0,1,1)) \\
\cdot (\overline{A} + B + C + f(1,0,0)) \\
\cdot (\overline{A} + B + \overline{C} + f(1,0,1)) \\
\cdot (\overline{A} + \overline{B} + C + f(1,1,0)) \\
\cdot (\overline{A} + \overline{B} + \overline{C} + f(1,1,1))
\end{aligned}
$$

A	B	C	$f(A, B, C)$
0	0	0	$f(0,0,0) = 0$
0	0	1	$f(0,0,1) = 1$
0	1	0	$f(0,1,0) = 0$
0	1	1	$f(0,1,1) = 0$
1	0	0	$f(1,0,0) = 1$
1	0	1	$f(1,0,1) = 0$
1	1	0	$f(1,1,0) = 1$
1	1	1	$f(1,1,1) = 0$

Level of Boolean Expressions

- f = a + b is a one-level expression
- f = ab + c is a two-level expression
- f = ab + cd is a two-level expression
- The more levels there are, the longer the delays



- we suppose all the input variables and their complemented forms to be available
  - $f = \bar{a}b + \bar{c}d$ is a two-level Boolean expression

Literals

- Literals are the number of input variables that make up a Boolean expression (not to be confused with the number of variables).
- For example: if f (a, b) is a logical function with two binary variables a and b:
  - f = a + b has 2 literals
  - f = ab + ab has 4 literals.

Minterms

- If an AND term in a Boolean expression contains all the direct or negated variables in the entire expression, it is called a fundamental product, or minterm.
- $X1\ X2\ \overline{X3}, \overline{X1}\ X2\ \overline{X3}, \overline{X1}\ \overline{X2}\ X3$, etc. in $f(X1, X2, X3)$
- An n-variable function has 2^n minterms
- Among all the possible combinations of variables, there is ***only one*** for which a certain minterm equals 1

Maxterms

- If an OR term in a Boolean expression contains all the direct or negated variables in the entire expression, it is called a fundamental sum, or maxterm.
- $X1 + X2 + \overline{X3}, \overline{X1} + X2 + \overline{X2}, \overline{X1} + \overline{X2} + X3$, etc. in $f(X1, X2, X3)$
- An n-variable function has 2^n maxterms
- There is ***only one*** combination of variables for which a certain maxterm equals 0

Implicants (蕴涵)

- Given the Boolean expressions f and g, g is an implicant of f: g implies f ($g \Rightarrow f$) or f covers g ($f \supset g$). ***f always = 1 when g = 1***.
  - $f(X, Y, Z) = XY + Z$
    - $XY \Rightarrow f$
    - $Z \Rightarrow f$
  - Every time Z and/or XY equal 1, f also equals 1. XY and Z are therefore implicants of f. X does not imply f : in fact if X equals 1 f does not necessarily equal 1.

Prime Implicants (素蘊涵)

- g is a prime implicant of f if:
  - $g \Rightarrow f$ ($f \supset g$);
  - g is not covered by another ***implicant with fewer literals***.
- f = XY + X + Z
  - $XY \Rightarrow f$ (not prime, $X \supset XY$, X term has one literal < XY's two literal)
  - $X \Rightarrow f$ (prime)
  - $Z \Rightarrow f$ (prime)
  - Y is not implicant of f

Assignment 1-2 (p.27-28: 4, 5, 6)

4. Using the theorems of Boolean algebra, it is possible to prove that the following expression equals 1 only when A and B are contemporaneously at 1 or when D is at 1 and C is contemporaneously at 0:
   $F = \overline{\overline{A}\,C\,B + \overline{A}\,C\,\overline{B} + \overline{A}\,\overline{D} + \overline{B}\,C\,D\ + \overline{B}\,C + \overline{B}\,\overline{D}}$
5. Minimize the following logical function with the criteria of Boolean algebra.
   $\mathrm{Y} = \overline{A}(A + B) + \overline{C} + B\,C$
6. Using De Morgan’s theorem, minimize the following logical function:
   $Y = \overline{A + A\,\overline{B} + C\,D}$

INTRODUCTORY CONCEPTS

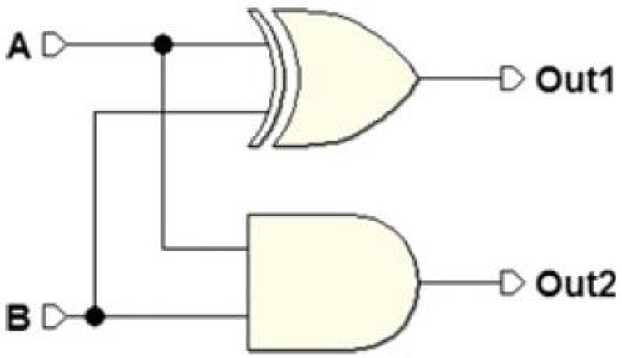
Combinational Networks

组合网络

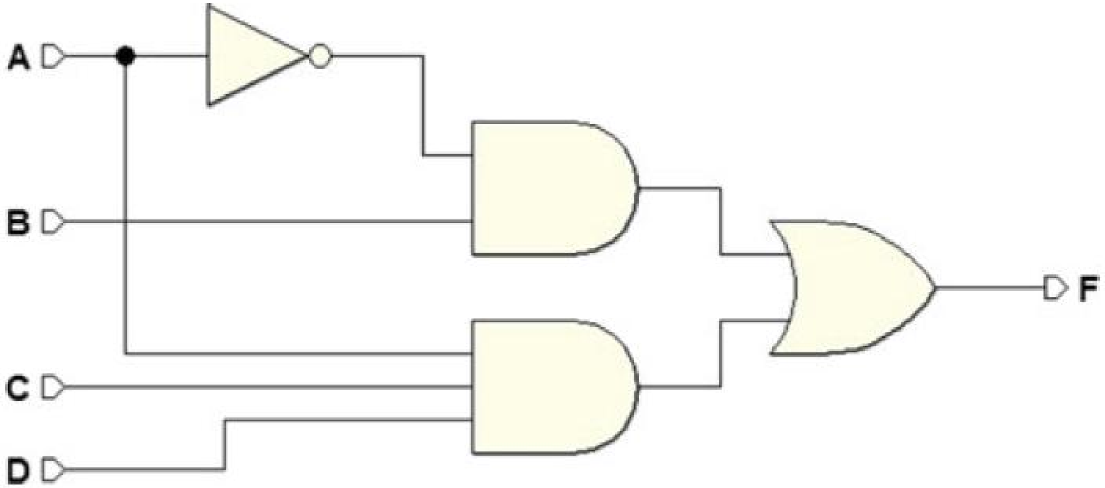
combinational (組合) & sequential (時序) networks

- A combinational network is defined as a logical circuit whose output depends only on ***the combination of its inputs***.
- The sequential network's output does not only depend on the values of the inputs in that time but also on the "***inputs history***." In other words, we will see that these networks have memory.
- ***A combinational network can be described in terms of a Boolean function but a sequential network can't.***

Logical Network Analysis

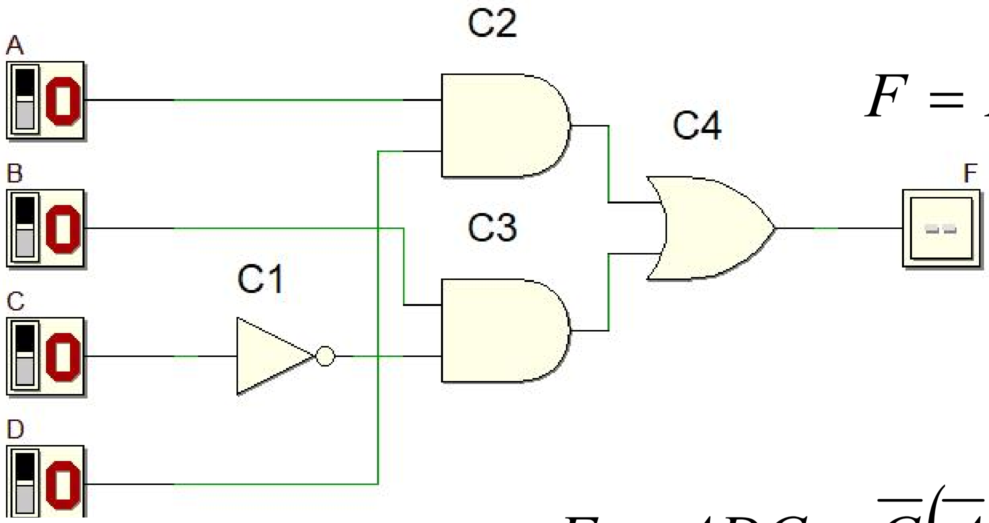


A	B	$Out1$	$Out2$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



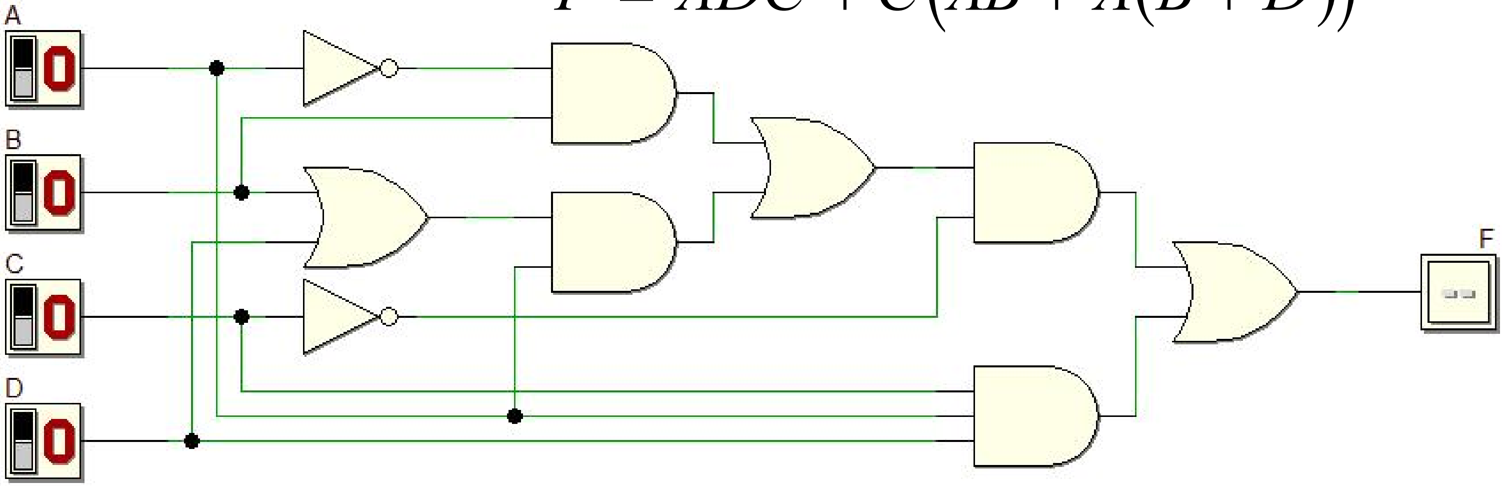
A	B	C	D	$\overline{A}$	$\overline{A}\,B$	$A\,C\,D$	F
0	0	0	0	1	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	1	0	0	0
0	0	1	1	1	0	0	0
0	1	0	0	1	1	0	1
0	1	0	1	1	1	0	1
0	1	1	0	1	1	0	1
0	1	1	1	1	1	0	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	0	1	1
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	1	1

Circuit Schematic of Logical Network



$$F = AD + \overline{C}B$$

$$F = ADC + \overline{C}\left(\overline{A}B + A(B + D)\right)$$



$$
\begin{aligned}
F &= ADC + \overline{C}\left(\overline{A}B + A(B + D)\right) \\
&= ADC + \overline{C}\left(\overline{A}B + AB + AD\right) \\
&= ADC + \overline{C}\left(B\left(\overline{A} + A\right) + AD\right) \\
&= ADC + \overline{C}\left(B + AD\right) \\
&= ADC + B\overline{C} + A\overline{C}D \\
&= AD\left(C + \overline{C}\right) + B\overline{C} \\
&= AD + B\overline{C} \\
&= AD + \overline{C}B
\end{aligned}
$$

Logical Network in VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.all;

ENTITY Circuit1 IS
    PORT(  iA, iB, iC, iD:     IN  std_logic;
           ooF:                OUT std_logic);
END Circuit1;

ARCHITECTURE structural OF Circuit1 IS
    COMPONENT NOT_gate IS
        PORT(  I: IN std_logic;
               O: OUT std_logic );
        END COMPONENT;
    COMPONENT AND2_gate IS
        PORT(  I0, I1: IN std_logic;
               O: OUT std_logic );
        END COMPONENT;
    COMPONENT OR2_gate IS
        PORT(  I0, I1: IN std_logic;
               O: OUT std_logic );
        END COMPONENT;

    SIGNAL S1, S2, S3: std_logic;

BEGIN -- structural
    C1: NOT_gate PORT MAP(iC, S1);
    C2: AND2_gate PORT MAP(iA, iD, S2);
    C3: AND2_gate PORT MAP(iB, S1, S3);
    C4: OR2_gate PORT MAP(S2, S3, ooF);
END structural;
```

Defining the Behavior of a Logical Network

$F = A \oplus B \oplus C$

A	B	C	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	.

$\rightarrow$

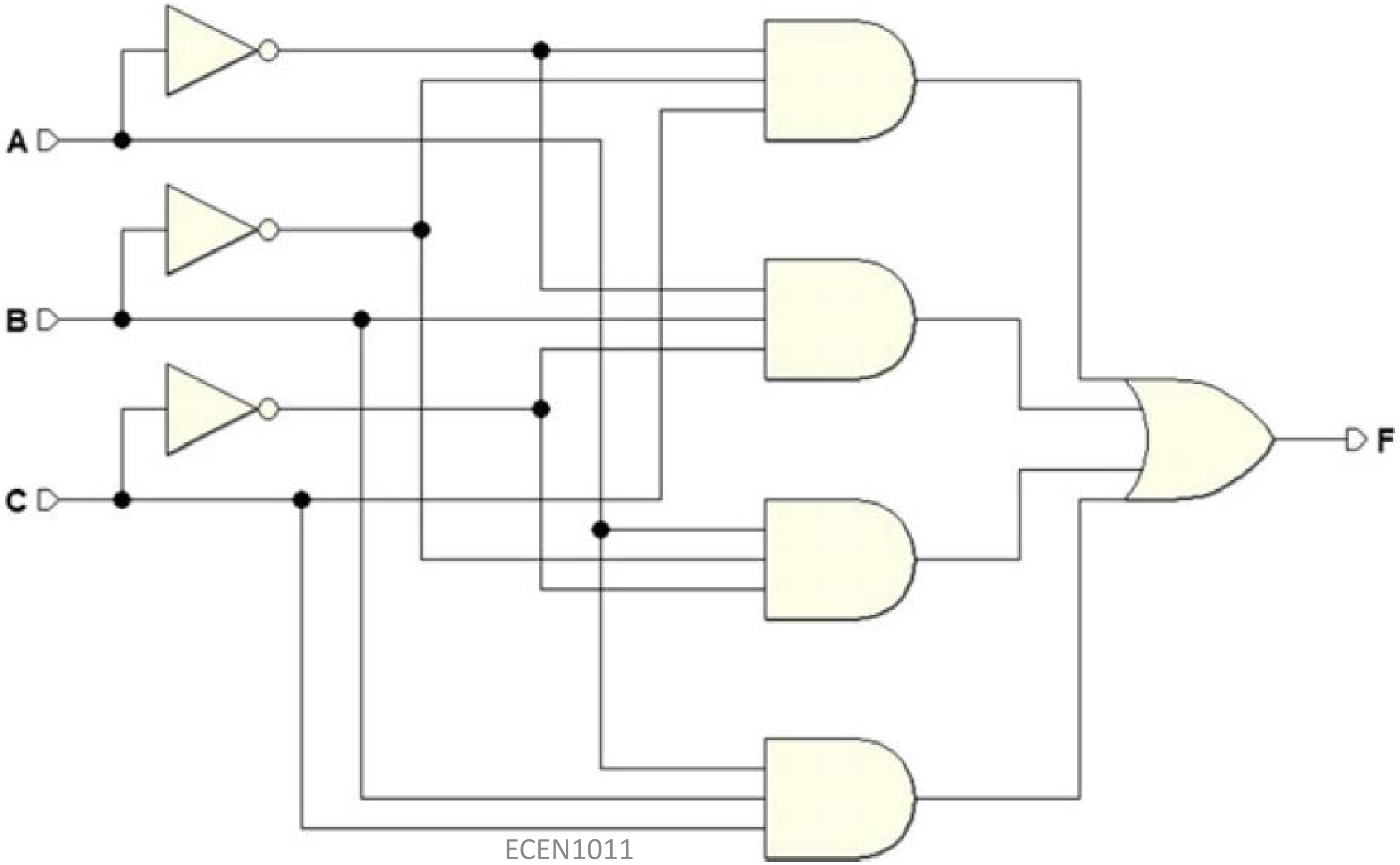
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Circuit Schematic from the Truth Table

A	B	C	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

- first form (AND–OR) of Shannon's Theorem

$$F = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$



Example: Controlling a Heating System

The system is composed of a thermostat, a heater, and a switch. T for temperature, I for input switch , and C for control are Boolean variables: the first two are inputs, the last, an output.

Example: Controlling a Heating System

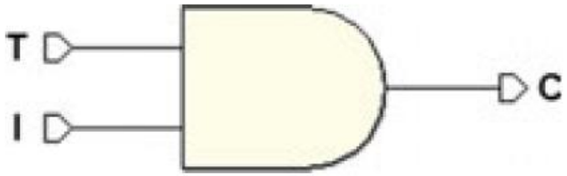
- Defining the Variables
  - T =0 (if t ≥ t0)   T =1 (if t < t0)
  - C = 0 (heater OFF) C = 1 (heater ON)
  - I = 0 (switch OFF) I = 1 (switch ON)
- Defining the Network Operation

• if I is OFF $\Rightarrow$ heater is OFF $(C = 0)$
• if I is ON but $(t \geq t_0)$ $\Rightarrow$ heater is OFF $(C = 0)$
• if I is ON and $(t < t_0)$ $\Rightarrow$ heater is ON $(C = 1)$

$\Rightarrow$

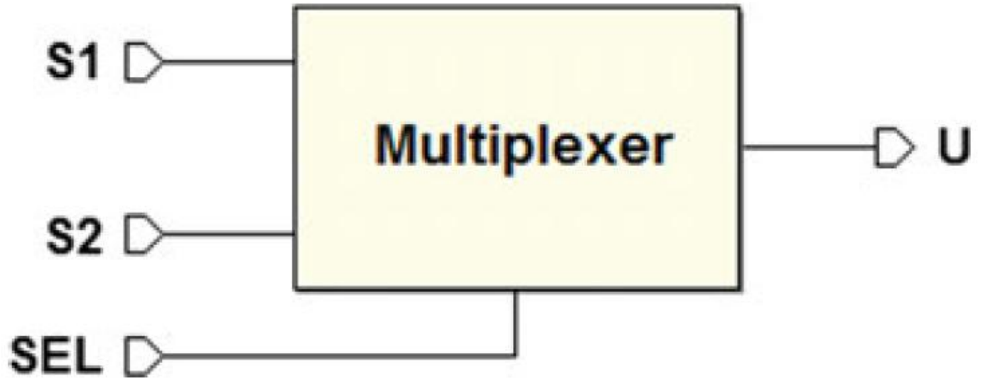
I	T	C
0	0	0
0	1	0
1	0	0
1	1	1

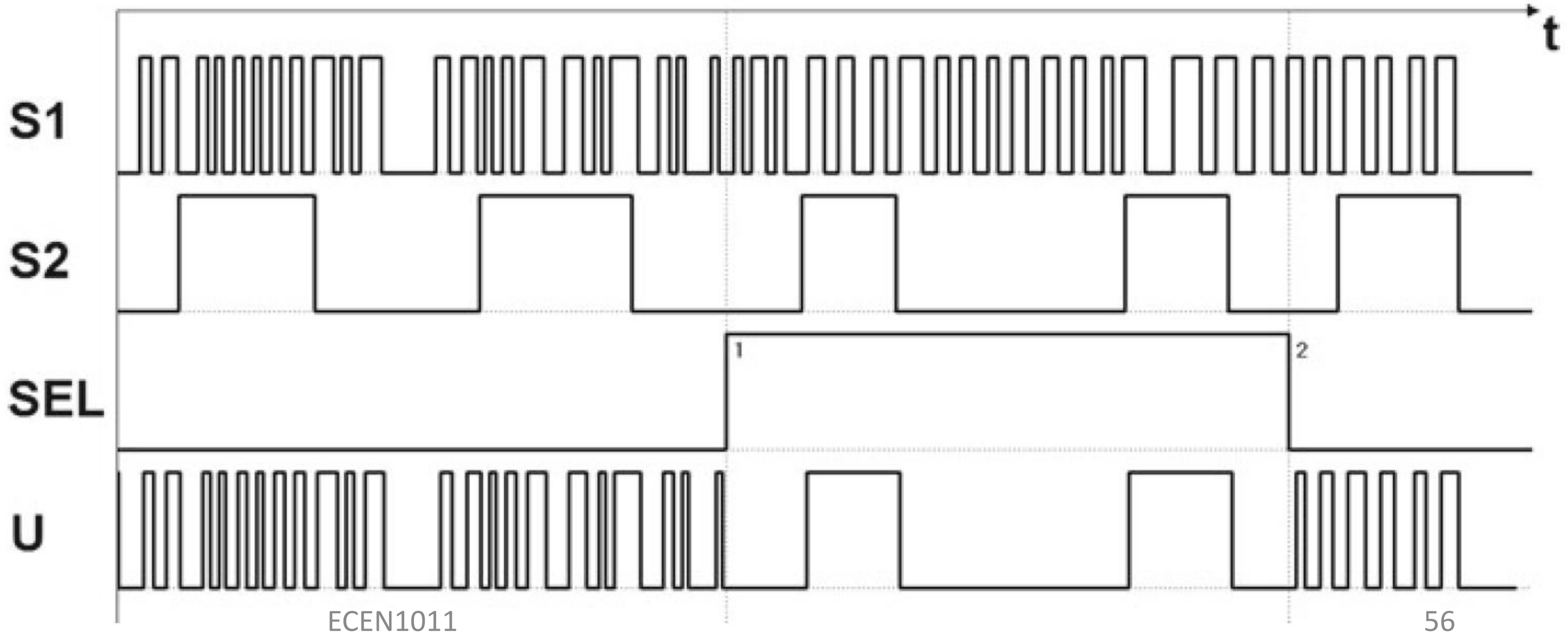
- Synthesis
  - C = I · T



Example: Two Channels Multiplexer (Selector)

This is a system that provides a Boolean output variable (U) that copies one of the two possible inputs (S1, S2), depending on the value of a control variable (SEL).

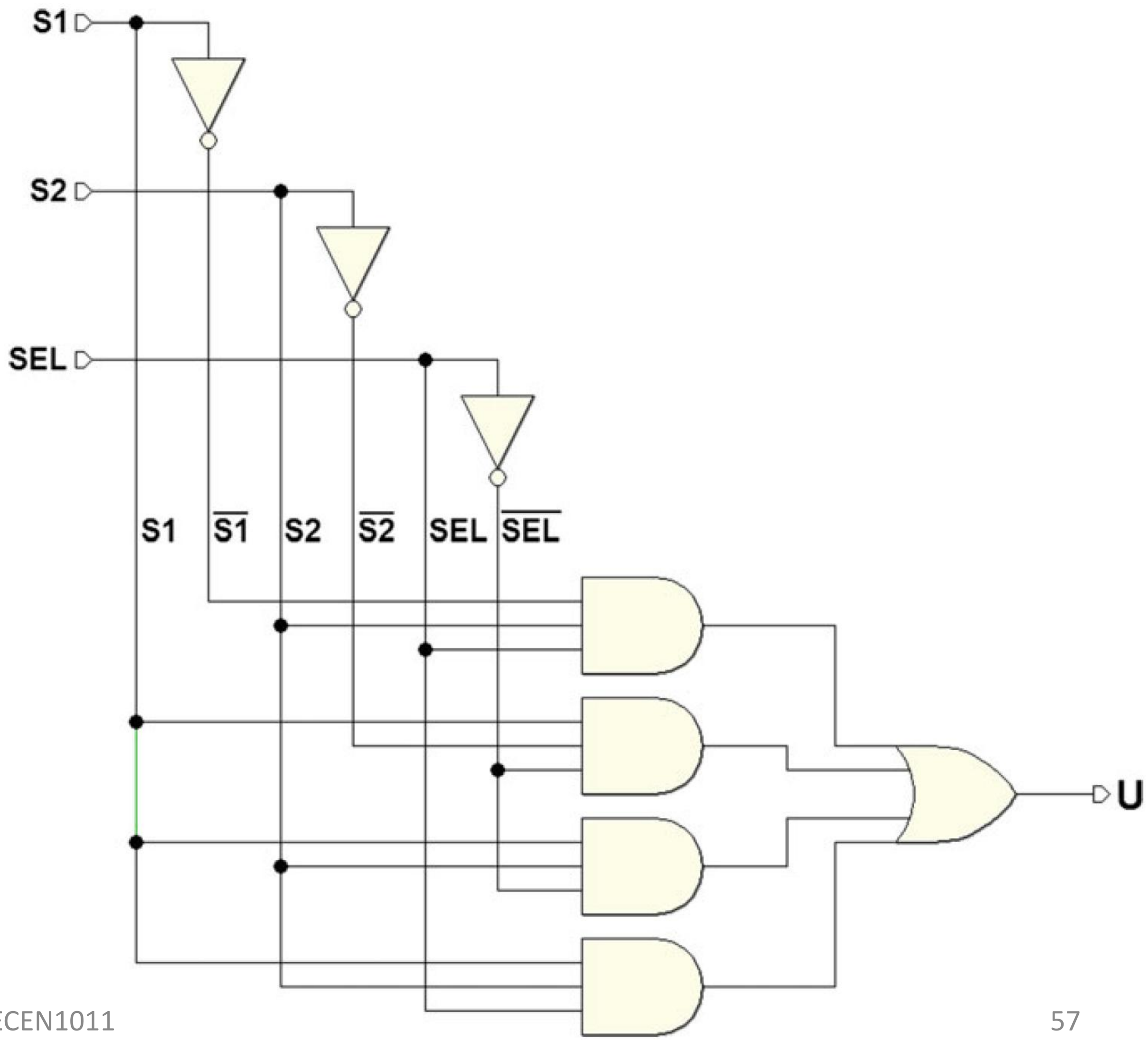




Example: Two Channels Multiplexer (Selector)

SEL	$S1$	$S2$	U	
0	0	0	0	
0	0	1	0	
0	1	0	1	(a)
0	1	1	1	(b)
1	0	0	0	
1	0	1	1	(c)
1	1	0	0	
1	1	1	1	(d)

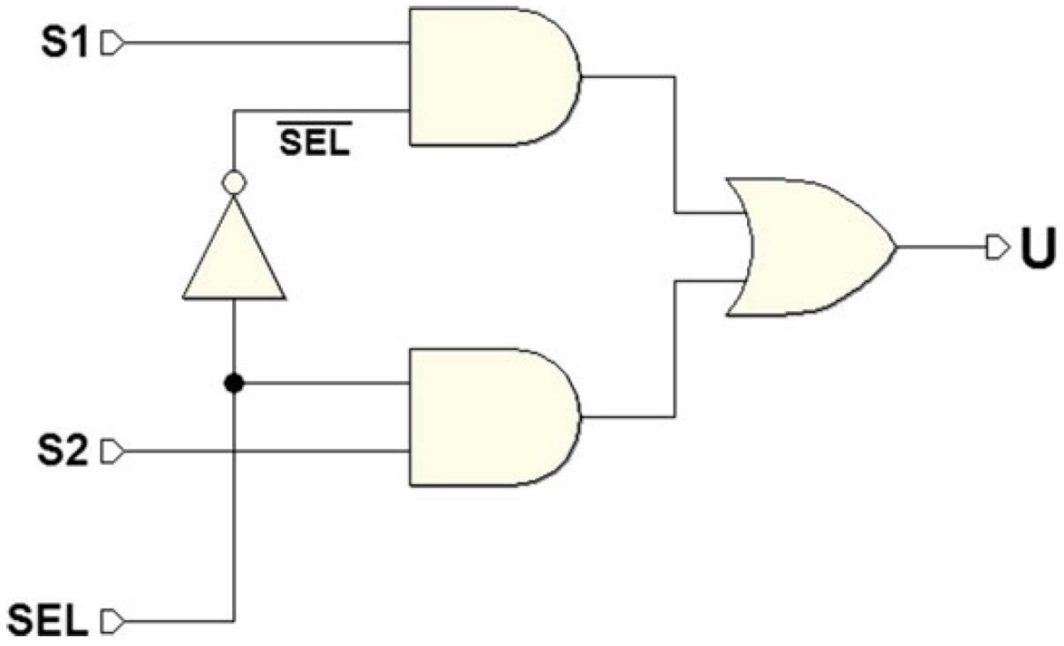
$$
\begin{aligned}
U = {} & \overline{SEL} \cdot S1 \cdot \overline{S2} + && \text{(a)} \\
& \overline{SEL} \cdot S1 \cdot S2 + && \text{(b)} \\
& SEL \cdot \overline{S1} \cdot S2 + && \text{(c)} \\
& SEL \cdot S1 \cdot S2 && \text{(d)}
\end{aligned}
$$



Example: Two Channels Multiplexer (Selector)

- Minimizing

$$
\begin{aligned}
U &= \overline{SEL}\ S1\ \overline{S2} + \overline{SEL}\ S1\ S2 + SEL\ \overline{S1}\ S2 + SEL\ S1\ S2 \\
&= \overline{SEL}\left(S1\ \overline{S2} + S1\ S2\right) + SEL\left(\overline{S1}\ S2 + S1\ S2\right) \\
&= \overline{SEL}\left(S1\left(\overline{S2} + S2\right)\right) + SEL\left(\left(\overline{S1} + S1\right)S2\right) \\
&= \overline{SEL}\ S1 + SEL\ S2
\end{aligned}
$$

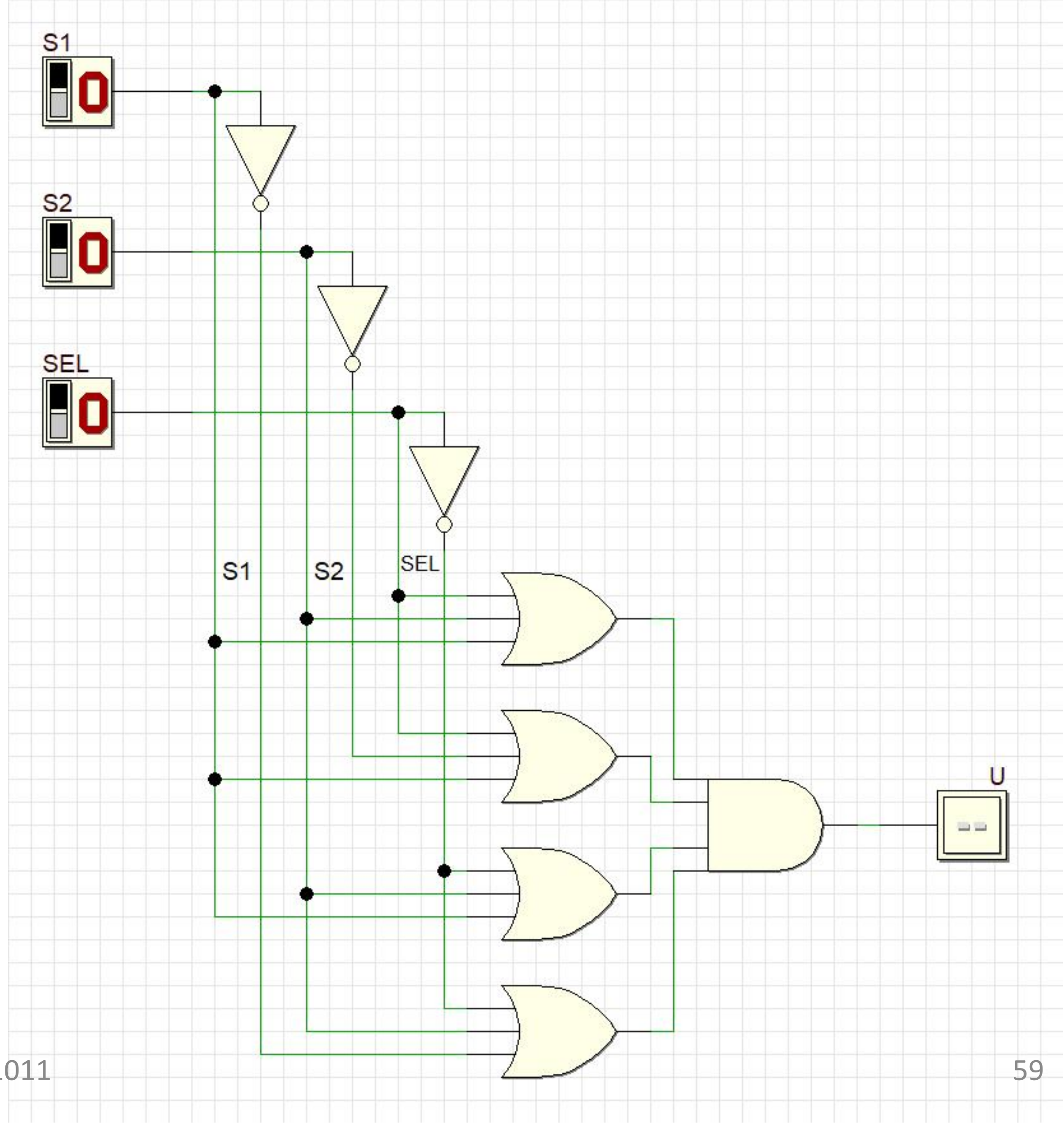


- Originally the expression had 12 literals; now it only has four.

Example: Two Channels Multiplexer (Selector)

SEL	$S1$	$S2$	U	
0	0	0	0	(a’)
0	0	1	0	(b’)
0	1	0	1	(a)
0	1	1	1	(b)
1	0	0	0	(c’)
1	0	1	1	(c)
1	1	0	0	(d’)
1	1	1	1	(d)

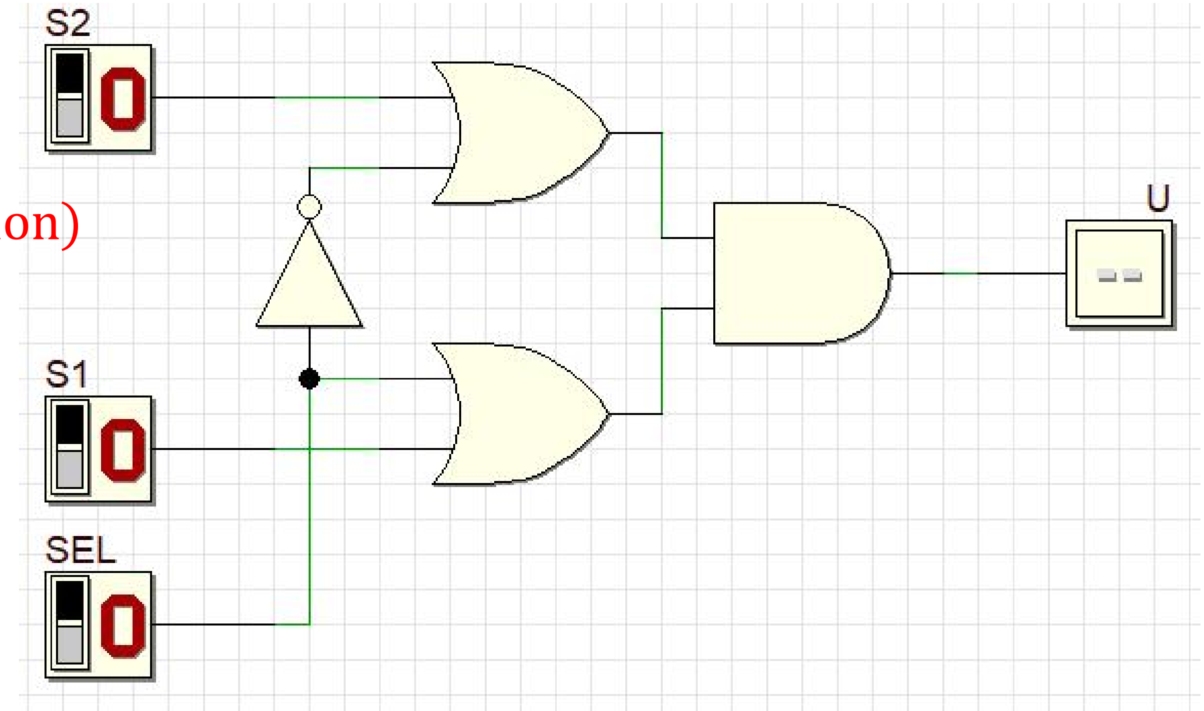
$$U = (SEL + S1 + S2) \cdot \quad \text{(a')}$$
$$(SEL + S1 + \overline{S2}) \cdot \quad \text{(b')}$$
$$(\overline{SEL} + S1 + S2) \cdot \quad \text{(c')}$$
$$(\overline{SEL} + \overline{S1} + S2) \cdot \quad \text{(d')}$$



Example: Two Channels Multiplexer (Selector)

- Minimizing

$$U = (SEL + S1 + S2) \cdot (SEL + S1 + \overline{S2}) \cdot (\overline{SEL} + S1 + S2) \cdot (\overline{SEL} + \overline{S1} + S2)$$
$$= ((SEL + S1) + (S2\overline{S2})) \cdot \left((\overline{SEL} + S2) + (S1\overline{S1})\right)$$
$$= \boldsymbol{(SEL + S1) \cdot (\overline{SEL} + S2)}$$
$$= SEL \cdot \overline{SEL} + S1 \cdot \overline{SEL} + SEL \cdot S2 + S1 \cdot S2$$
$$= S1 \cdot \overline{SEL} + S2(SEL + S1)$$
$$= S1 \cdot \overline{SEL} + S2(SEL + \overline{SEL} \cdot S1) \quad \text{(Absorption)}$$
$$= S1 \cdot \overline{SEL} + SEL \cdot S2 + \overline{SEL} \cdot S1 \cdot S2$$
$$= S1 \cdot \overline{SEL}\,(1 + S2) + S2 \cdot SEL$$
$$= S1 \cdot \overline{SEL} + S2 \cdot SEL$$



- Originally the expression had 12 literals; now it only has four.

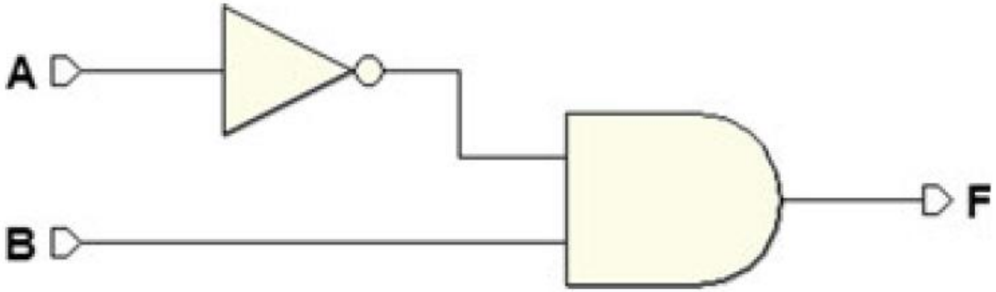
Two Channels Multiplexer in VHDL

```
library ieee;
use ieee.std_logic_1164.all;

ENTITY Multiplexer_2_1 IS
    PORT(    S1, S2, SEL: IN  std_logic;
             U: OUT std_logic );
END Multiplexer_2_1;

ARCHITECTURE behavioral OF Multiplexer_2_1 IS
BEGIN
    U <=  S1 when (SEL = '0') else
          S2 when (SEL = '1') else 'X';
END behavioral;
```

```
U <=  (not(SEL) and S1) or (SEL and S2);
```

Assignment 1-3 (p.28: 7, 8)

7. Design the circuit that implements the expression $\overline{A}\,B + A\,\overline{B}\,\overline{C}\,(B + C)$, and verify that it is equivalent to the following network:



8. Design the circuits corresponding to the following logical expressions:

   (a) $C = (A + B) + \overline{A\,B}$
   (b) $D = A\,(B + C)\,\overline{C}$
   (c) $E = \overline{A}\,D\,\overline{(C + \overline{D})}$
   (d) $G = \overline{A}\,B\,\overline{C} + D\,(C + A\,\overline{B}\,C)$

Assignment 1-4 (p.28: 9, 10)

9. Do the following conversions between the canonical forms:

   (a) $F = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C}$ to the OR–AND form.

   (b) $G = (\overline{A} + B + C)(\overline{A} + \overline{B} + \overline{C})(A + B + C)$ to the AND–OR form.

10. Analytically derive the logical function of the following circuit: