Digital Logic
Lecture Content

- Boolean Algebra
- Gates & Circuits
- Combinational Circuits
- Minimizing Circuits
- Sequential Circuits

Digital Logic
Learning Objectives

- **Understand the basic operations of Boolean algebra**
- **Distinguish among the different types of flip-flops**
- **Use Karnaugh maps and the Quine-McCluskey method to simplify Boolean expressions**

Boolean Algebra

- Mathematical discipline used to design and analyze the behavior of the digital circuitry in digital computers and other digital systems
  - Named after George Boole
  - Proposed basic principles of the algebra in 1854

- Claude Shannon suggested Boolean algebra could be used to solve problems in relay-switching circuit design

- Is a convenient tool:
  - Analysis: It is an economical way of describing the function of digital circuitry
  - Design: Given a desired function, Boolean algebra can be applied to develop a simplified implementation of that function

Boolean Variables and Operations

- As in any algebra, we have variables and operations
  - A variable may take on the value 1 (TRUE) or 0 (FALSE)

- **AND ($A \cdot B$)**
  - Yields true (binary value 1) if and only if both of its operands are true
  - The AND operation takes precedence over the OR operation
  - Can be represented by simple concatenation instead of the dot operator

- **OR ($A + B$)**
  - Yields true if either or both of its operands are true

- **NOT ($\overline{A}$)**
  - Inverts the value of its operand

Boolean Operators

- With two input variables:

P	Q	NOT P $(\overline{P})$	P AND Q $(P \cdot Q)$	P OR Q $(P + Q)$	P NAND Q $(\overline{P \cdot Q})$	P NOR Q $(\overline{P + Q})$	P XOR Q $(P \oplus Q)$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	0

- Extended to more than input variables:

Operation	Expression	Output = 1 if
AND	$A \cdot \mathrm{B} \cdot \ldots$	All of the set $\{A, B, \ldots\}$ are 1.
OR	$A + B + \cdots$	Any of the set $\{A, B, \ldots\}$ are 1.
NAND	$\overline{A \cdot B \cdot \ldots}$	Any of the set $\{A, B, \ldots\}$ are 0.
NOR	$\overline{A + B + \cdots}$	All of the set $\{A, B, \ldots\}$ are 0.
XOR	$A \oplus B \oplus \cdots$	The set $\{A, B, \ldots\}$ contains an odd number of 1.

Basic Algebraic Transformations

Basic Postulates		
$A \cdot B = B \cdot A$	$A + B = B + A$	Commutative Laws
$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	$A + (B \bullet C) = (A + B) \cdot (A + C)$	Distributive Laws
$1 \cdot A = A$	$0 + A = A$	Identity Elements
$A \cdot \bar{A} = 0$	$A + \bar{\mathrm{A}} = 1$	Inverse Elements
Other Identities		
$0 \cdot A = 0$	$1 + A = 1$	
$A \cdot A = A$	$A + A = A$	
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$	Associative Laws
$\overline{A \cdot B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A} \cdot \bar{B}$	DeMorgan's Theorem

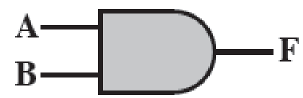
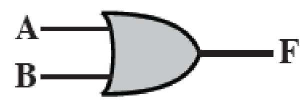
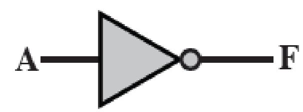
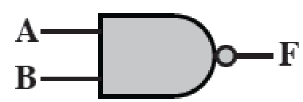
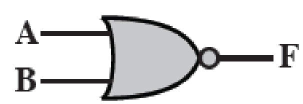
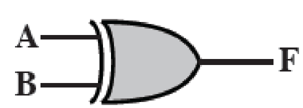
Digital Logic
Lecture Content

- Boolean Algebra
- Gates & Circuits
- Combinational Circuits
- Minimizing Circuits
- Sequential Circuits

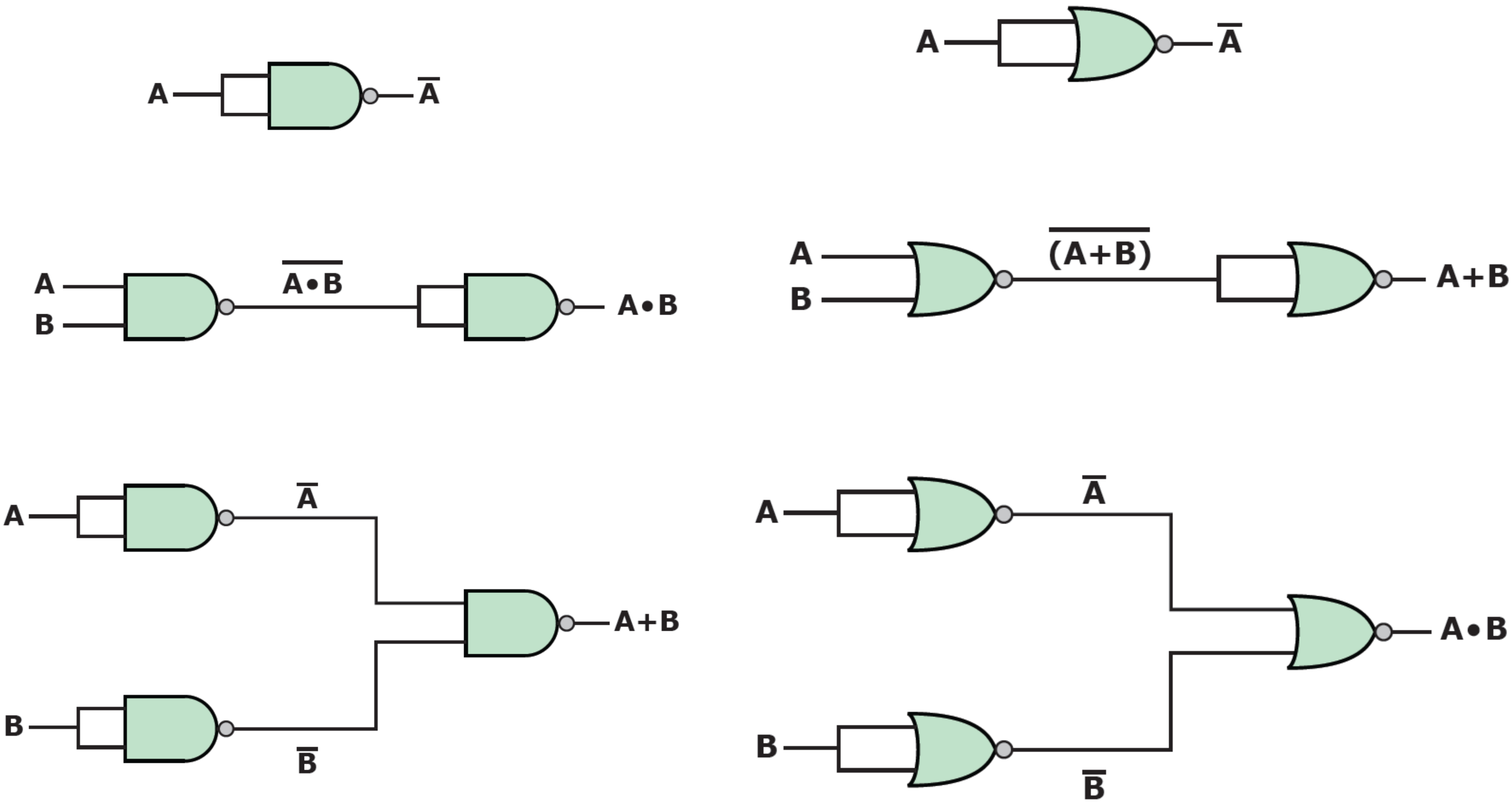
Basic Electronic Circuits

- A gate is an electronic circuit that produces an output signal that is a simple Boolean operation on its input signals.

- We say that to **assert** a signal is to cause a signal line to make a transition from its logically false (0) state to its logically true (1) state.

Name	Graphical Symbol	Algebraic Function	Truth Table
AND		$F = A \cdot B$ or $F = AB$	A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = A + B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$F = \overline{A}$ or $F = A'$	A | F 0 | 1 1 | 0
NAND		$F = \overline{AB}$	A B | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{A + B}$	A B | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR		$F = A \oplus B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Using NAND and NOR Only



Digital Logic
Lecture Content

- Boolean Algebra
- Gates & Circuits
- Combinational Circuits
- Minimizing Circuits
- Sequential Circuits

Combinational Circuits

- An interconnected set of gates whose output at any time is a function only of the input at that time
  - The appearance of the input is followed almost immediately by the appearance of the output, with only gate delays
  - Consists of n binary inputs and m binary outputs

- Can be defined in three ways:
  - Truth table
    - For each of the 2^n possible combinations of input signals, the binary value of each of the m output signals is listed
  - Graphical symbols
    - The interconnected layout of gates is depicted
  - Boolean equations
    - Each output signal is expressed as a Boolean function of its input signals

Small Example: A Truth Table

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

Small Example: Boolean Function

- We can transform the truth table easily into a boolean expression
  - We simply look at the position where F is true:
    - $\bar{A}B\bar{C}$
    - $\bar{A}BC$
    - $AB\bar{C}$

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

Small Example: Boolean Function

- We can transform the truth table easily into a boolean expression
  - We simply look at the position where F is true:
    - $\bar{A}B\bar{C}$
    - $\bar{A}BC$
    - $AB\bar{C}$
  - This results in the following expression:

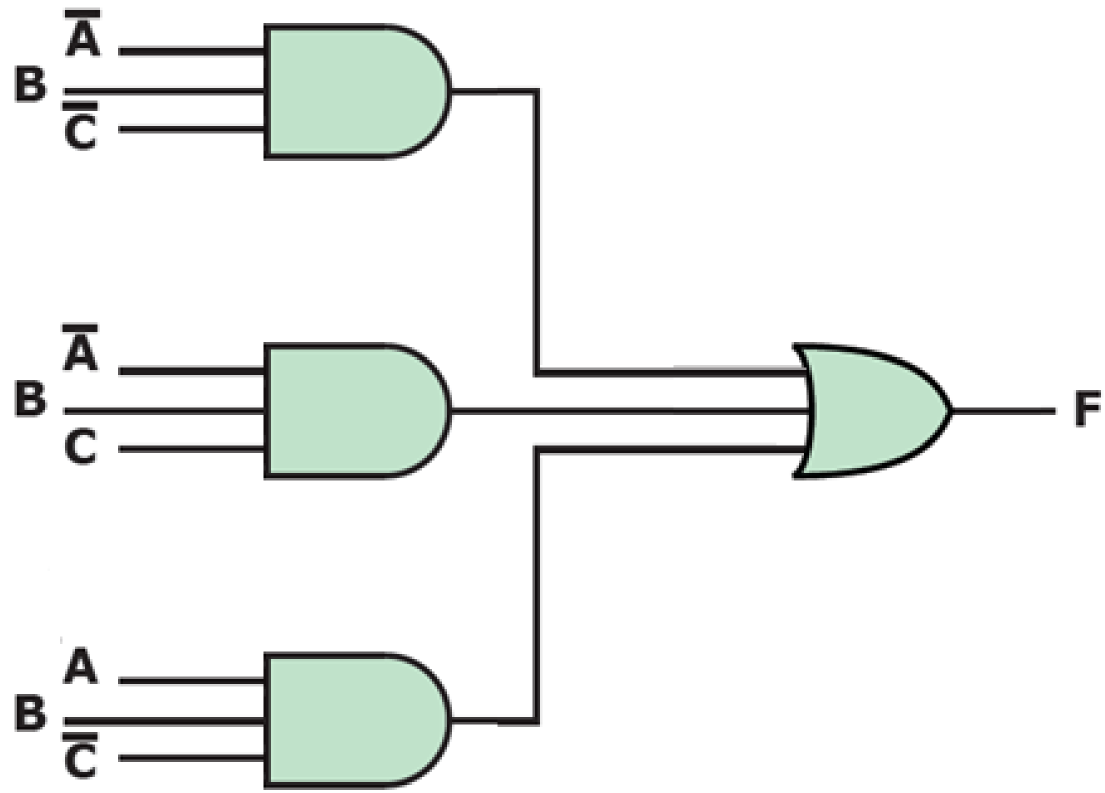
$$\boldsymbol{F = \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}}$$

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

- This is called **Sum of Products (SOP)**

Small Example: Circuit Implementation

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



Small Example: Another Boolean Expression

- We could also look at the positions where F is false:
  - $\bar{A}\bar{B}\bar{C}$
  - $\bar{A}\bar{B}C$
  - $A\bar{B}\bar{C}$
  - $A\bar{B}C$
  - $\bar{A}\bar{B}\bar{C}$

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

Small Example: Another Boolean Expression

- We could also look at the positions where F is false:
  - $\bar{A}\bar{B}\bar{C}$
  - $\bar{A}\bar{B}C$
  - $A\bar{B}\bar{C}$
  - $A\bar{B}C$
  - $\bar{A}\bar{B}\bar{C}$
- The output is 1 when all of these terms are false:

$$F = \overline{(\bar{A}\bar{B}\bar{C})} \cdot \overline{(\bar{A}\bar{B}C)} \cdot \overline{(A\bar{B}\bar{C}\ )} \cdot \overline{(A\bar{B}C\ )} \cdot \overline{(\bar{A}\bar{B}\bar{C})}$$

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

Small Example: Another Boolean Expression

- We can also say that the output is 1 if none of the input combination producing 0 is true:
  - $\bar{A}\bar{B}\bar{C}$
  - $\bar{A}\bar{B}C$
  - $A\bar{B}\bar{C}$
  - $A\bar{B}C$
  - ABC
- The output is 1 when all of these terms are false:

$$F = \overline{(\bar{A}\bar{B}\bar{C})} \cdot \overline{(\bar{A}\bar{B}C)} \cdot \overline{(A\bar{B}\bar{C})} \cdot \overline{(A\bar{B}C)} \cdot \overline{(ABC)}$$

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

- Following DeMorgan:

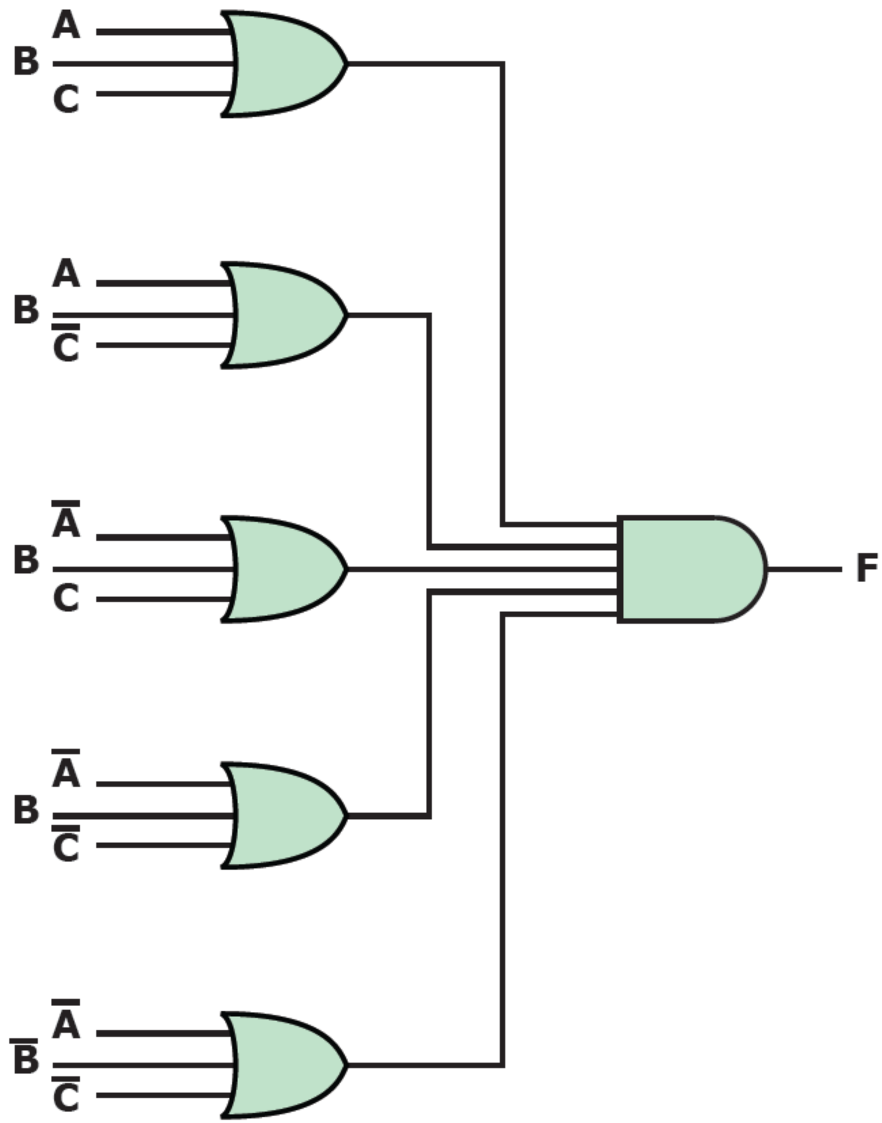
$$\boldsymbol{F = (A + B + C) \cdot (A + B + \bar{C}) \cdot (\bar{A} + B + C) \cdot (\bar{A} + B + \bar{C}) \cdot (\bar{A} + \bar{B} + \bar{C})}$$

Digital Logic
Lecture Content

- Boolean Algebra
- Gates & Circuits
- Combinational Circuits
- Minimizing Circuits
- Sequential Circuits

Small Example: Circuit Implementation

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



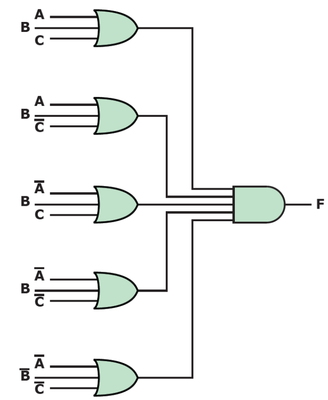
Small Example: Another Boolean Expression

- The last function is called **Product of Sums (POS)**
- Observations:
  - There are several possibilities to express the same function
  - Some are more clever than others
- Considerations:
  - We could use the function with less gates
  - Or it may be preferable to use only NANDs or NORs
- But very often, there is an much smaller expression than SOP or POS

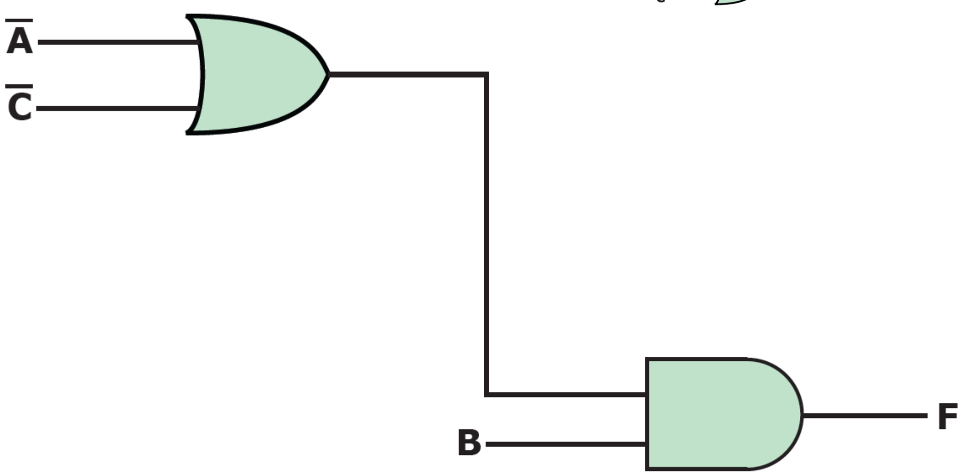
Small Example: Simplified Implementation

- The given Function can also be expressed as

$$\boldsymbol{F = B(\overline{A} + \overline{C})}$$



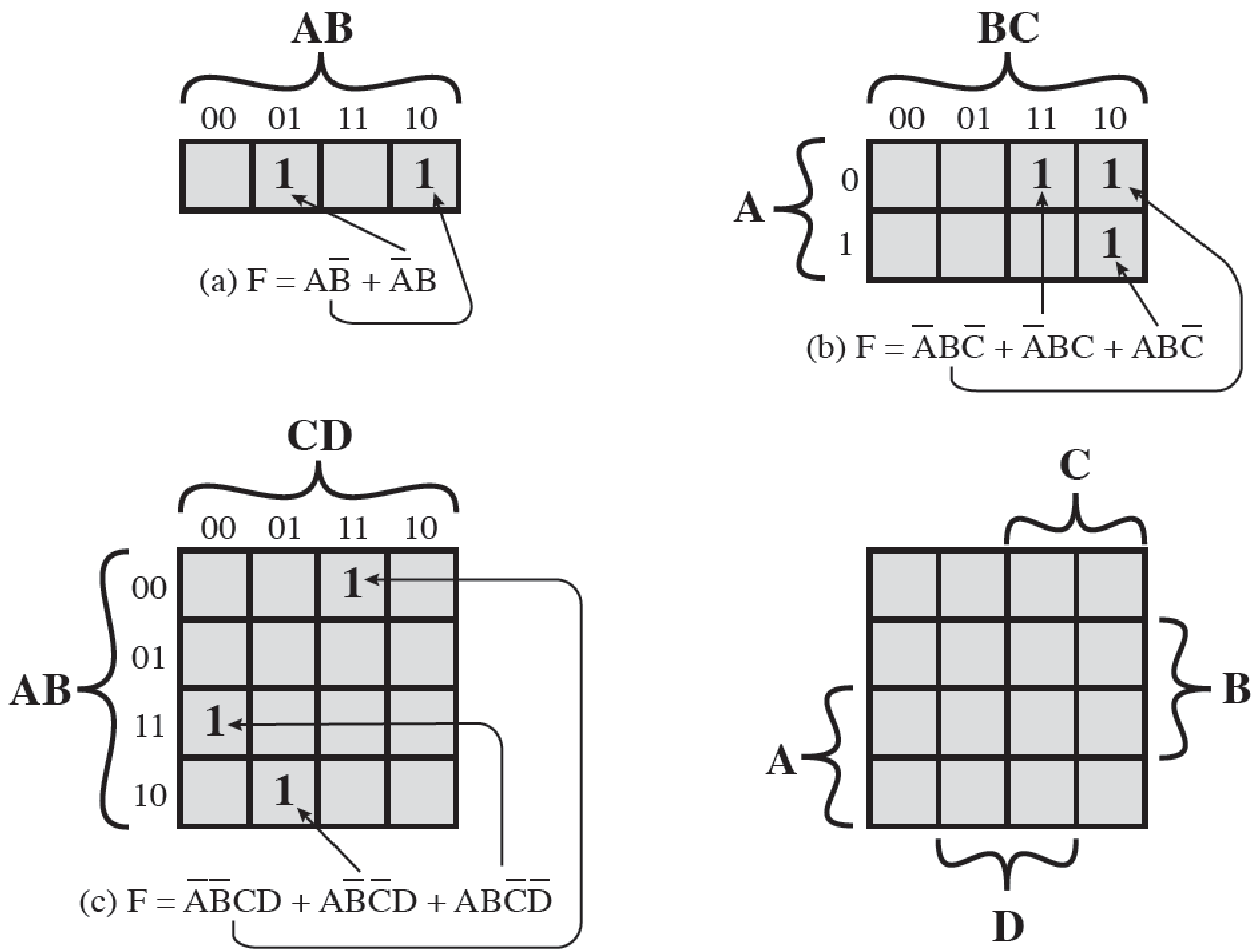
A	B	C	F
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**0**



How to Find Simplified Expressions

- By algebraic simplifications
  - Kind of cumbersome, hard to find
- **Karnaugh Maps**
  - Very easy
  - Only up to 4 variables (more are possible but then it gets complicated again)
- **Quine-McCluskey Tables**
  - Algorithmic Approach
  - Works for more than four variables

Karnaugh Maps



(a) $F = A\overline{B} + \overline{A}B$

(b) $F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$

(c) $F = \overline{A}\,\overline{B}CD + A\overline{B}\,\overline{C}D + AB\overline{C}\,\overline{D}$

(d) Simplified Labeling of Map

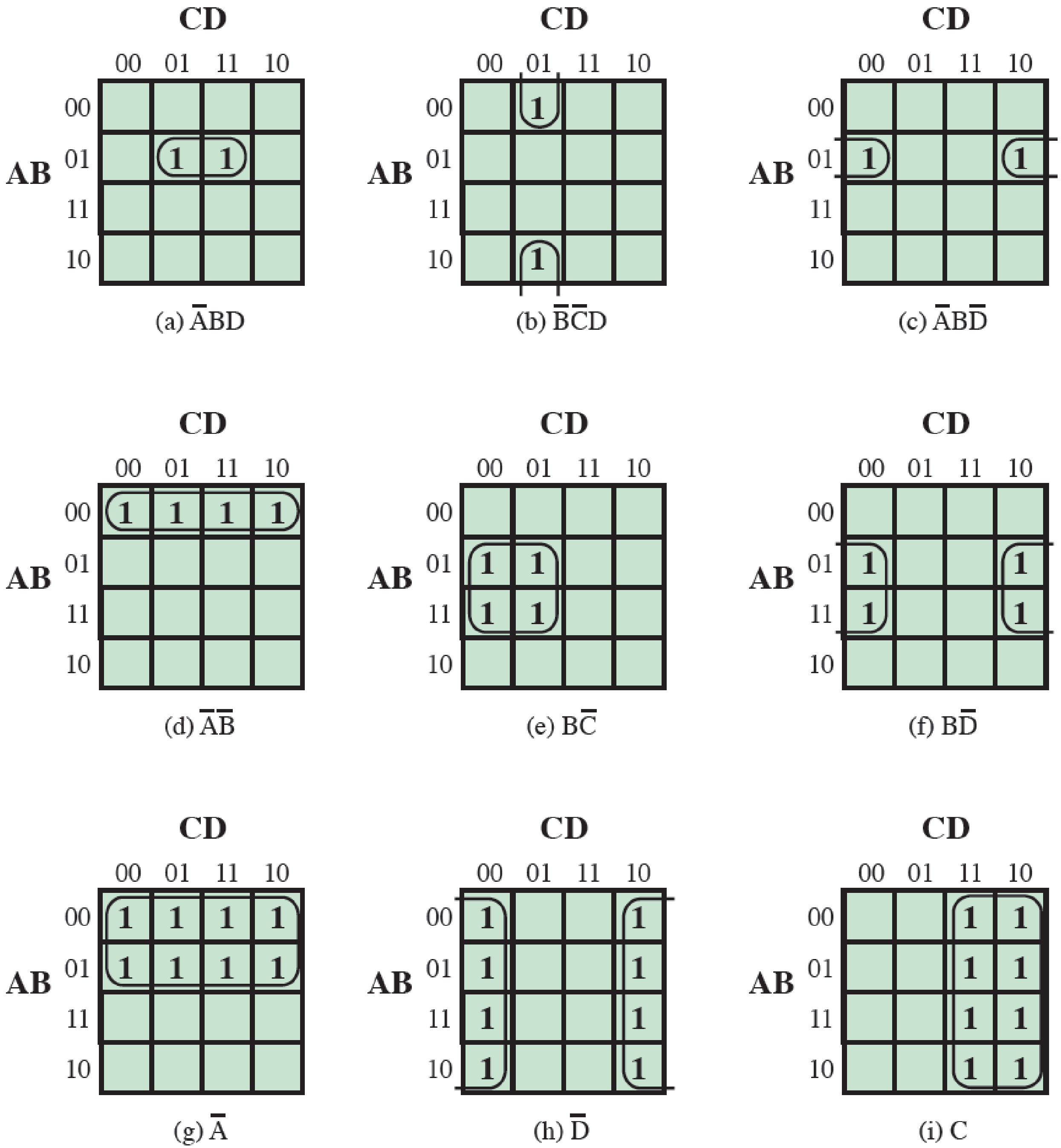
Karnaugh Maps: How Does This Help Us?

- The Maps show the variables and their connection
- We can write a simple algebraic expression by looking at the arrangement of the 1s on the map

- Observation:
  - Whenever two 1s are adjacent, then the corresponding product terms differ in only one variable.
  - In such a case, the two terms can be merged by eliminating that variable.

- Example:

$$\boldsymbol{RX + R\overline{X} = R}$$

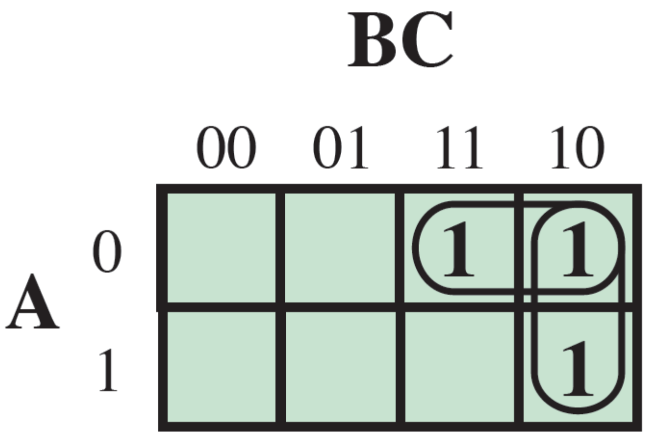

(a) $\overline{A}BD$
(b) $\overline{B}\overline{C}D$
(c) $\overline{A}B\overline{D}$
(d) $\overline{A}\overline{B}$
(e) $B\overline{C}$
(f) $B\overline{D}$
(g) $\overline{A}$
(h) $\overline{D}$
(i) C

Karnaugh Maps: Usage

- Find the largest block of size 1, 2, 4, or 8 in each dimension
- Select additional blocks (as large as possible and as few as possible) until all 1s are marked
- No non-1 must be selected
- Each one can be member of several blocks
- Write down the equation corresponding to the blocks

Karnaugh Maps: Our Example

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



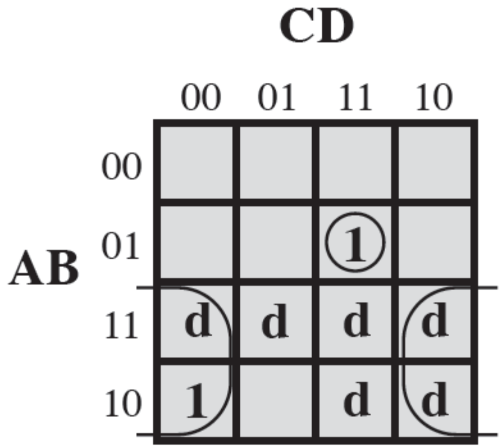
(a) $F = \overline{A}B + B\overline{C}$

Another Example: Decimal Incrementer

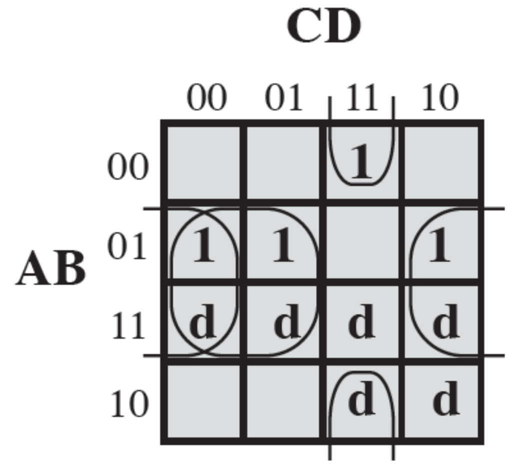
	Input					Output			
Number	**A**	**B**	**C**	**D**	**Number**	**W**	**X**	**Y**	**Z**
0	0	0	0	0	**1**	0	0	0	1
1	0	0	0	1	**2**	0	0	1	0
2	0	0	1	0	**3**	0	0	1	1
3	0	0	1	1	**4**	0	1	0	0
4	0	1	0	0	**5**	0	1	0	1
5	0	1	0	1	**6**	0	1	1	0
6	0	1	1	0	**7**	0	1	1	1
7	0	1	1	1	**8**	1	0	0	0
8	1	0	0	0	**9**	1	0	0	1
9	1	0	0	1	**0**	0	0	0	0
	1	0	1	0		d	d	d	d
	1	0	1	1		d	d	d	d
	1	1	0	0		d	d	d	d
	1	1	0	1		d	d	d	d
	1	1	1	0		d	d	d	d
	1	1	1	1		d	d	d	d

Don't care!

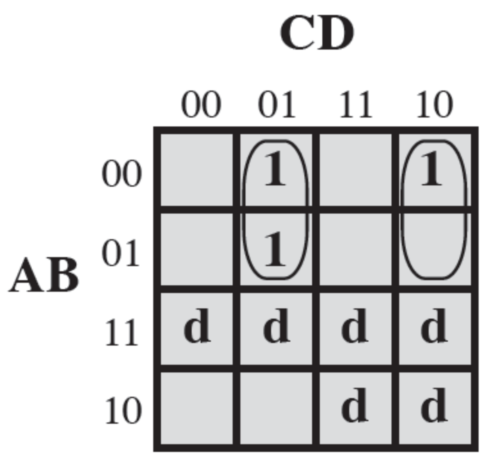
Karnaugh Maps: Decimal Incrementer



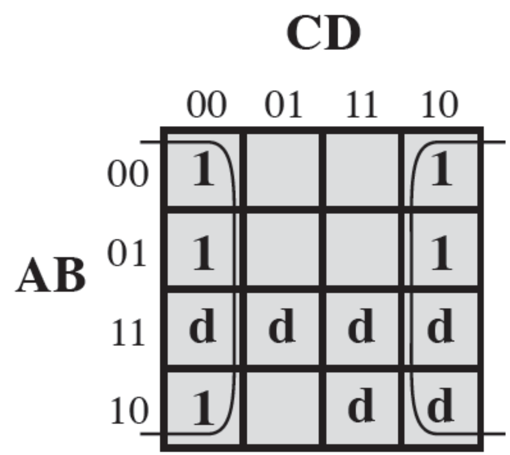
(a) $W = A\overline{D} + \overline{A}BCD$



(b) $X = B\overline{D} + B\overline{C} + BCD$



(c) $Y = \overline{A}\overline{C}D + \overline{A}C\overline{D}$



(d) $Z = \overline{D}$

The Quine-McCluskey Method

- The Karnaugh Maps are very inconvenient for more than 4 variables

- Other methods are required: The Quine-McCluskey Method

- Let's assume we have the following term:

$$\boldsymbol{F = ABCD + AB\overline{C}D + AB\overline{C}\,\overline{D} + A\overline{B}CD + \overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}D}$$

The Quine-McCluskey Method

- In the first step, we order the terms in groups:
  - We count the number of complemented variables
  - Terms with the same number of complemented variables form a group

$\overline{A}\overline{B}\overline{C}D$	$AB\overline{C}\overline{D}$	$AB\overline{C}D$	$ABCD$
	$\overline{A}BC\overline{D}$	$A\overline{B}CD$	
	$\overline{A}B\overline{C}D$	$\overline{A}BCD$	

- We now look for term differing in only one variable
- They can be replaced by one term without the differing variable
- Because of the ordering: Potential terms can only be in the group to the right

The Quine-McCluskey Method

$\overline{A}\overline{B}\overline{C}D$	$AB\overline{C}\overline{D}$	$AB\overline{C}D$	$ABCD$
	$\overline{A}BC\overline{D}$	$A\overline{B}CD$	
	$\overline{A}B\overline{C}D$	$\overline{A}BCD$	

The Quine-McCluskey Method

$\overline{A}\,\overline{B}\,\overline{C}D$	$AB\overline{C}\,\overline{D}$	$AB\overline{C}D$	$ABCD$
	$\overline{A}BC\overline{D}$	$A\overline{B}CD$	
	$\overline{A}B\overline{C}D$	$\overline{A}BCD$	

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$
	$\bar{A}BC\bar{D}$	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	

The Quine-McCluskey Method

$\overline{A}\overline{B}\overline{C}D$	$AB\overline{C}\overline{D}$	$AB\overline{C}D$	$ABCD$
	$\overline{A}BC\overline{D}$	$A\overline{B}CD$	
	$\overline{A}B\overline{C}D$	$\overline{A}BCD$	

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$
	$\bar{A}BC\bar{D}$	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$	

$\bar{A}\bar{C}D$

The Quine-McCluskey Method

$\overline{A}\overline{B}\overline{C}D$ ✓	$AB\overline{C}\overline{D}$	$AB\overline{C}D$	$ABCD$
	$\overline{A}BC\overline{D}$	$A\overline{B}CD$	
	$\overline{A}B\overline{C}D$ ✓	$\overline{A}BCD$	

$\overline{A}\overline{C}D$

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$
	$\bar{A}BC\bar{D}$	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$	

	$\bar{A}\bar{C}D$	$AB\bar{C}$

The Quine-McCluskey Method

$\overline{A}\,\overline{B}\,\overline{C}D$ ✓	$AB\overline{C}\,\overline{D}$ ✓	$AB\overline{C}D$ ✓	$ABCD$
	$\overline{A}BC\overline{D}$	$A\overline{B}CD$	
	$\overline{A}B\overline{C}D$ ✓	$\overline{A}BCD$	

	$\overline{A}\,\overline{C}D$	$AB\overline{C}$

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$
	$\bar{A}BC\bar{D}$	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$	

	$\bar{A}\bar{C}D$	$AB\bar{C}$	

The Quine-McCluskey Method

$\overline{A}\,\overline{B}\,\overline{C}D$ ✓	$AB\overline{C}\,\overline{D}$ ✓	$AB\overline{C}D$ ✓	$ABCD$
	$\overline{A}BC\overline{D}$	$A\overline{B}CD$	
	$\overline{A}B\overline{C}D$ ✓	$\overline{A}BCD$	

$\overline{A}\,\overline{C}D$ $\quad\quad$ $AB\overline{C}$

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$
	$\bar{A}BC\bar{D}$	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$	

	$\bar{A}\bar{C}D$	$AB\bar{C}$	

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$
	$\bar{A}BC\bar{D}$	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$	

	$\bar{A}\bar{C}D$	$AB\bar{C}$	

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$
	$\bar{A}BC\bar{D}$ ✓	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$ ✓	

	$\bar{A}\bar{C}D$	$AB\bar{C}$	
		$\bar{A}BC$	

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$
	$\bar{A}BC\bar{D}$ ✓	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$ ✓	

	$\bar{A}\bar{C}D$	$AB\bar{C}$	
		$\bar{A}BC$	
		$B\bar{C}D$	

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$
	$\bar{A}BC\bar{D}$ ✓	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$ ✓	

$\bar{A}\bar{C}D$	$AB\bar{C}$
	$\bar{A}BC$
	$B\bar{C}D$

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$
	$\bar{A}BC\bar{D}$ ✓	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$ ✓	

	$\bar{A}\bar{C}D$	$AB\bar{C}$	
		$\bar{A}BC$	
		$B\bar{C}D$	
		$\bar{A}BD$	

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$ ✓
	$\bar{A}BC\bar{D}$ ✓	$A\bar{B}CD$	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$ ✓	

	$\bar{A}\bar{C}D$	$AB\bar{C}$	ABD
		$\bar{A}BC$	
		$B\bar{C}D$	
		$\bar{A}BD$	

The Quine-McCluskey Method

$\overline{A}\,\overline{B}\,\overline{C}D$ ✓	$AB\overline{C}\,\overline{D}$ ✓	$AB\overline{C}D$ ✓	$ABCD$ ✓
	$\overline{A}BC\overline{D}$ ✓	$A\overline{B}CD$ ✓	
	$\overline{A}B\overline{C}D$ ✓	$\overline{A}BCD$ ✓	

	$\overline{A}\,\overline{C}D$	$AB\overline{C}$	ABD
		$\overline{A}BC$	ACD
		$B\overline{C}D$	
		$\overline{A}BD$	

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$ ✓
	$\bar{A}BC\bar{D}$ ✓	$A\bar{B}CD$ ✓	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$ ✓	

	$\bar{A}\bar{C}D$	$AB\bar{C}$	ABD
		$\bar{A}BC$	ACD
		$B\bar{C}D$	BCD
		$\bar{A}BD$	

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$ ✓
	$\bar{A}BC\bar{D}$ ✓	$A\bar{B}CD$ ✓	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$ ✓	

$\bar{A}\bar{C}D$	$AB\bar{C}$	ABD
	$\bar{A}BC$	ACD
	$B\bar{C}D$	BCD
	$\bar{A}BD$	

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$ ✓
	$\bar{A}BC\bar{D}$ ✓	$A\bar{B}CD$ ✓	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$ ✓	
	$\bar{A}\bar{C}D$	$AB\bar{C}$	ABD ✓
		$\bar{A}BC$	ACD
		$B\bar{C}D$ ✓	BCD ✓
		$\bar{A}BD$ ✓	
			BD

The Quine-McCluskey Method

$\bar{A}\bar{B}\bar{C}D$ ✓	$AB\bar{C}\bar{D}$ ✓	$AB\bar{C}D$ ✓	$ABCD$ ✓
	$\bar{A}BC\bar{D}$ ✓	$A\bar{B}CD$ ✓	
	$\bar{A}B\bar{C}D$ ✓	$\bar{A}BCD$ ✓	

	$\bar{A}\bar{C}D$	$AB\bar{C}$	ABD ✓
		$\bar{A}BC$	ACD
		$B\bar{C}D$ ✓	BCD ✓
		$\bar{A}BD$ ✓	

			BD

The Quine-McCluskey Method

- We have already reduced the number of terms to 5:

$$F = BD + \bar{A}\bar{C}D + \bar{A}BC + AB\bar{C} + ACD$$

- Some of might still be redundant

	$ABCD$	$AB\bar{C}D$	$AB\bar{C}\bar{D}$	$A\bar{B}CD$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}\bar{B}\bar{C}D$
BD								
$\bar{A}\bar{C}D$								
$\bar{A}BC$								
$AB\bar{C}$								
ACD								

The Quine-McCluskey Method

- First, we mark all those cells which intersecting terms are compatible
- That is, the variables in the row have the same value as in the column

	$ABCD$	$AB\bar{C}D$	$AB\bar{C}\bar{D}$	$A\bar{B}CD$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}\bar{B}\bar{C}D$
BD	o	o			o		o	
$\bar{A}\bar{C}D$							o	o
$\bar{A}BC$					o	o		
$AB\bar{C}$		o	o					
ACD	o			o				

The Quine-McCluskey Method

- Now, we mark all those o's, which are alone in their column:

	$ABCD$	$AB\bar{C}D$	$AB\bar{C}\bar{D}$	$A\bar{B}CD$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}B\bar{C}\bar{D}$
BD	o	o			o		o	
$\bar{A}\bar{C}D$							o	x
$\bar{A}BC$					o	x		
$AB\bar{C}$		o	x					
ACD	o			x				

The Quine-McCluskey Method

- Now, we mark all those x in which row there is already a marked x:

	$ABCD$	$AB\overline{C}D$	$AB\overline{C}\overline{D}$	$A\overline{B}CD$	$\overline{A}BCD$	$\overline{A}BC\overline{D}$	$\overline{A}B\overline{C}D$	$\overline{A}\overline{B}\overline{C}D$
BD	O	O			O		O	
$\overline{A}\overline{C}D$							x	x
$\overline{A}BC$					x	x		
$AB\overline{C}$		x	x					
ACD	x			x				

The Quine-McCluskey Method

- If all columns are covered, we have a final result:

$$F = \bar{A}\bar{C}D + \bar{A}BC + AB\bar{C} + ACD$$

- If there would be an uncovered column, we have to select further rows until all rows are covered

	$ABCD$	$AB\bar{C}D$	$AB\bar{C}\bar{D}$	$A\bar{B}CD$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}\bar{B}\bar{C}D$
BD	O	O			O		O	
$\bar{A}\bar{C}D$							X	X
$\bar{A}BC$					X	X		
$AB\bar{C}$		X	X					
ACD	X			X				

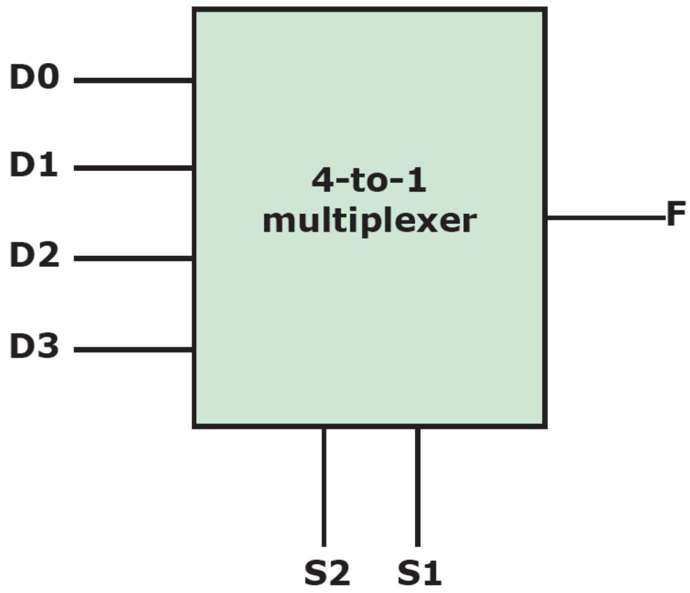
Some More Important Circuits

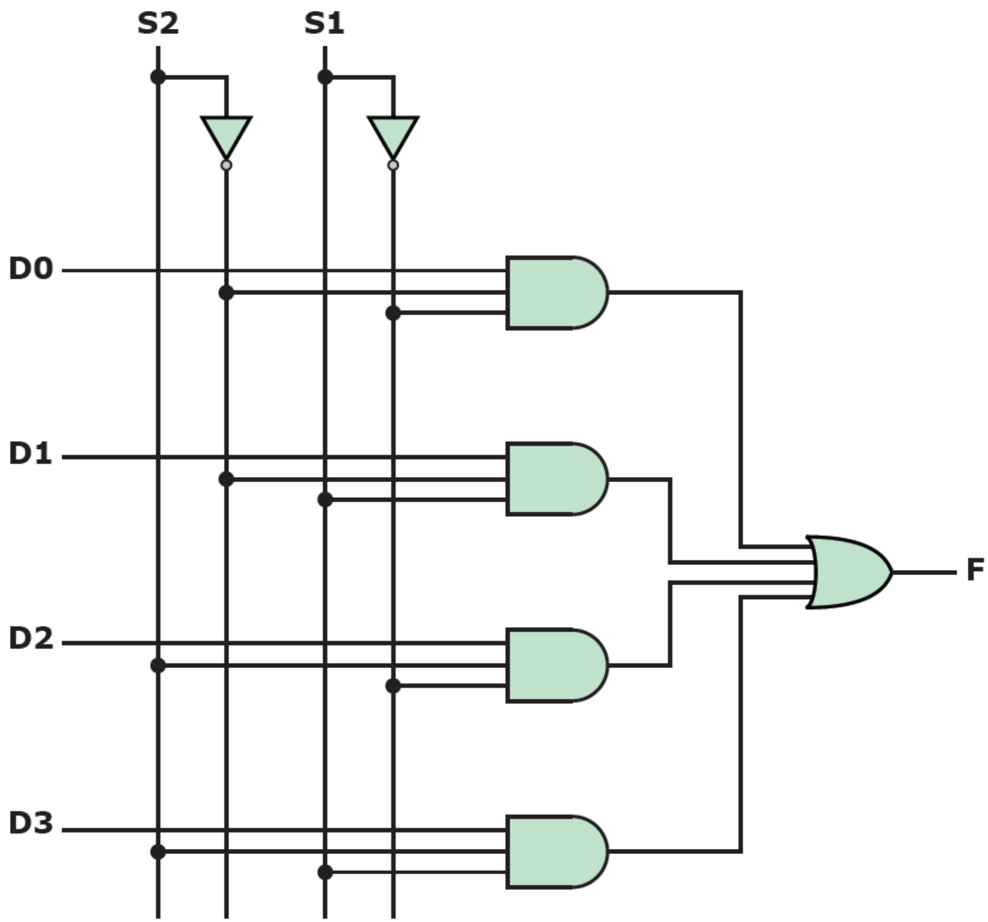
- Multiplexer:
  - Connects multiple inputs to one output.
  - At any time, only one input is connected to the output
  - Used for signal or data routing

- Decoder:
  - Has a number of output lines of which only one is asserted
  - Generally, they have n inputs and 2^n output lines

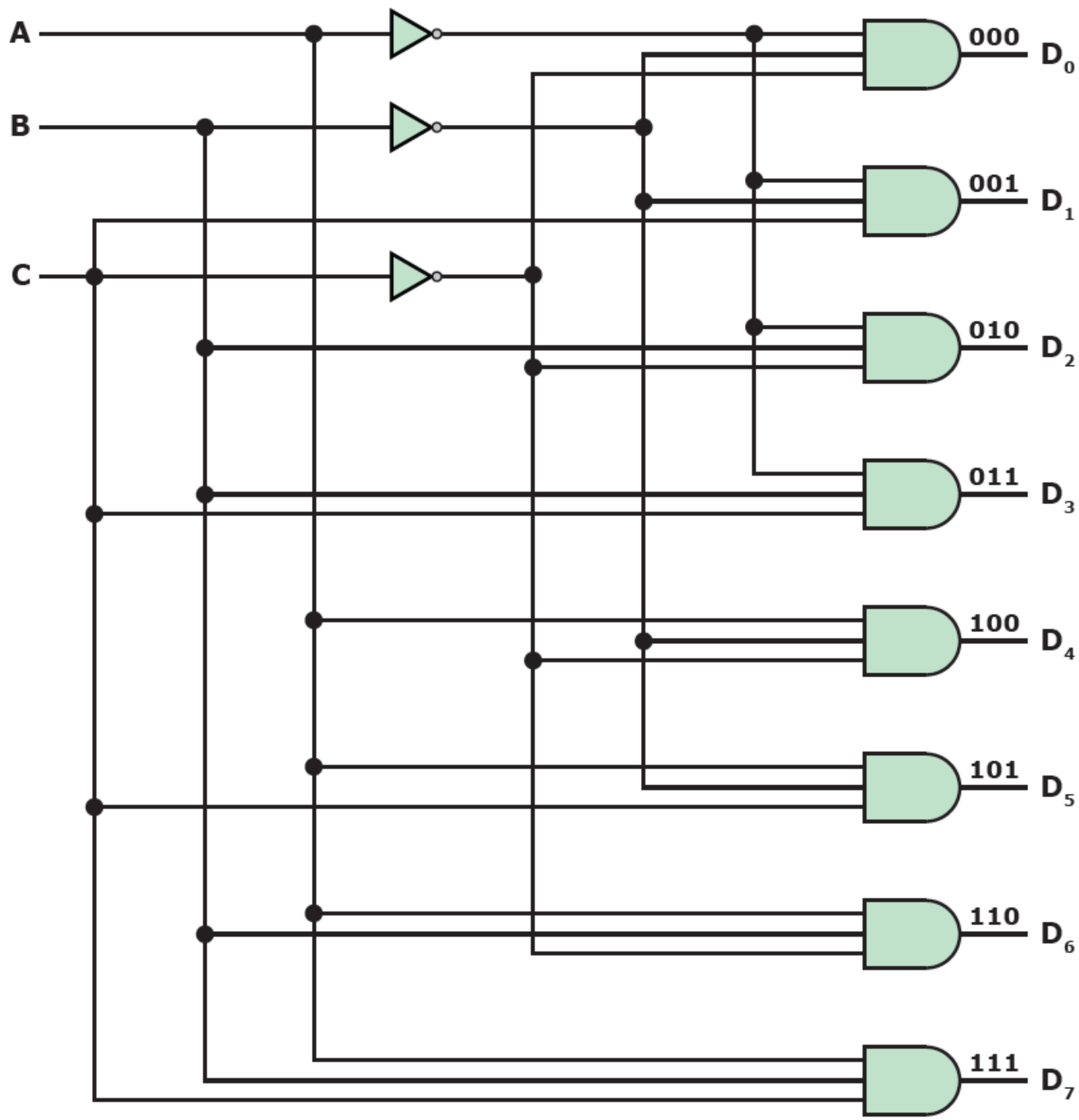
- Adders:
  - Pretty obvious

A Multiplexer

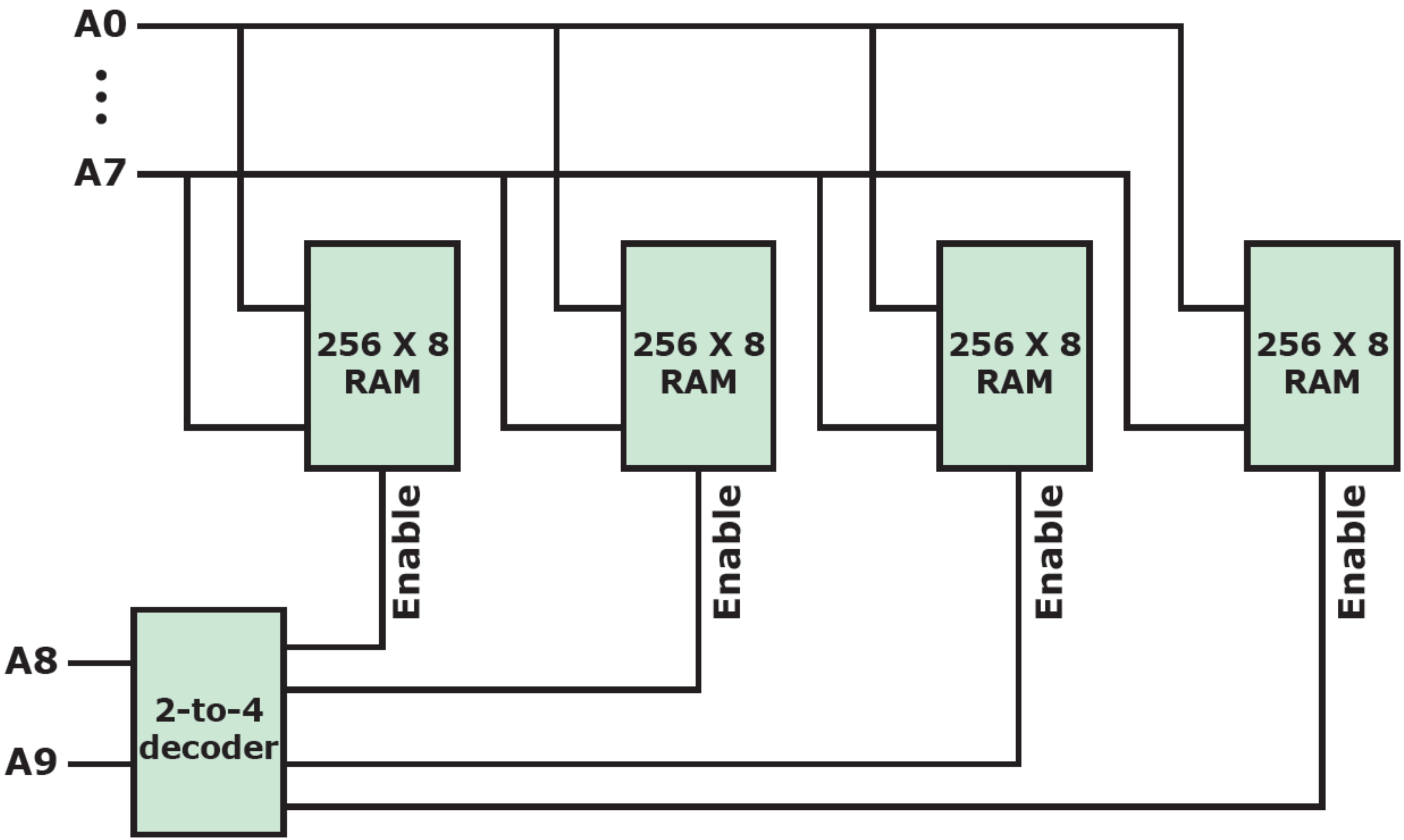




Decoder



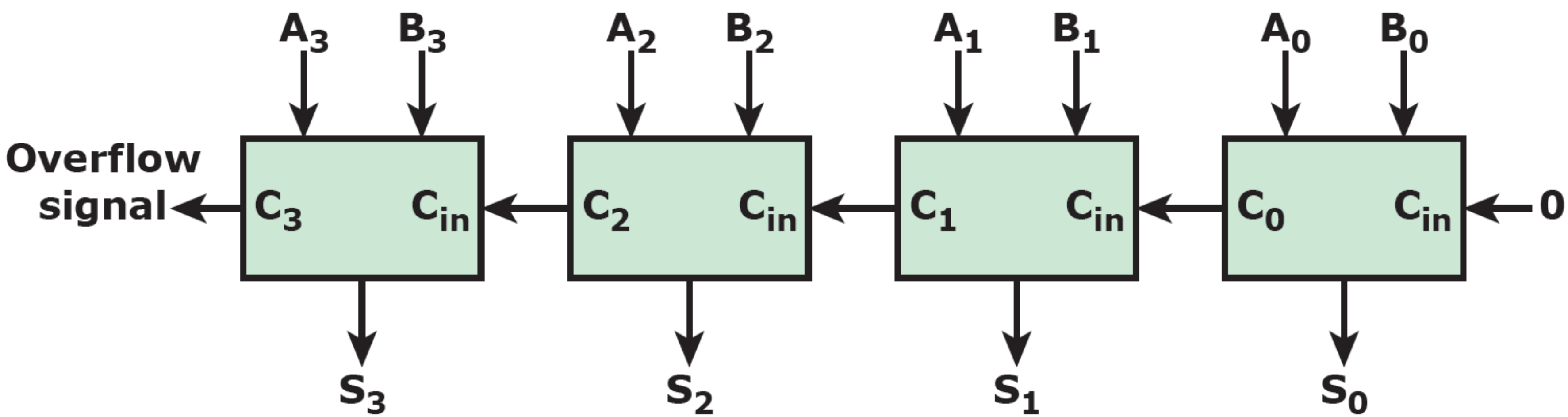
Use of Decoders



Truth Table of A One Bit Adder

C_{in}	A	B	Sum	C_{out}
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

Combined One Bit Adder



Problems with this Design

- Each Adder has to wait for the Carry signal of the other adders
- This delay becomes unacceptable for larger adders

Problems with this Design

- Each Adder has to wait for the Carry signal of the other adders
- This delay becomes unacceptable for larger adders

Carry Look Ahead:

- For each bit, we can already determine the carry bit:

$$C_0 = A_0B_0$$

$$C_1 = A_1B_1 + A_1A_0B_0 + B_1A_0B_0$$

$$C_2 = A_2B_2 + A_2A_1B_1 + A_2A_1A_0B_0 + A_2B_1A_0B_0 + B_2A_1B_1 + B_2A_1A_0B_0 + B_2B_1A_0B_0$$

Problems with this Design

Carry Look Ahead:

- For each bit, we can already determine the carry bit:

$$C_0 = A_0 B_0$$

$$C_1 = A_1 B_1 + A_1 A_0 B_0 + B_1 A_0 B_0$$

$$C_2 = A_2 B_2 + A_2 A_1 B_1 + A_2 A_1 A_0 B_0 + A_2 B_1 A_0 B_0 + B_2 A_1 B_1 + B_2 A_1 A_0 B_0 + B_2 B_1 A_0 B_0$$

- Each carry bit can be expressed this way in a SOP
- They are getting increasingly complex:
  - A n-Bit adder requires $2^n - 1$ AND gates and an OR gate with $2^n - 1$ inputs
  - This is normally only done up to 8 Bit
  - Larger Adder are then built of 8-Bit adders

Digital Logic
Lecture Content

- Boolean Algebra
- Gates & Circuits
- Combinational Circuits
- Minimizing Circuits
- Sequential Circuits

Sequential Circuits

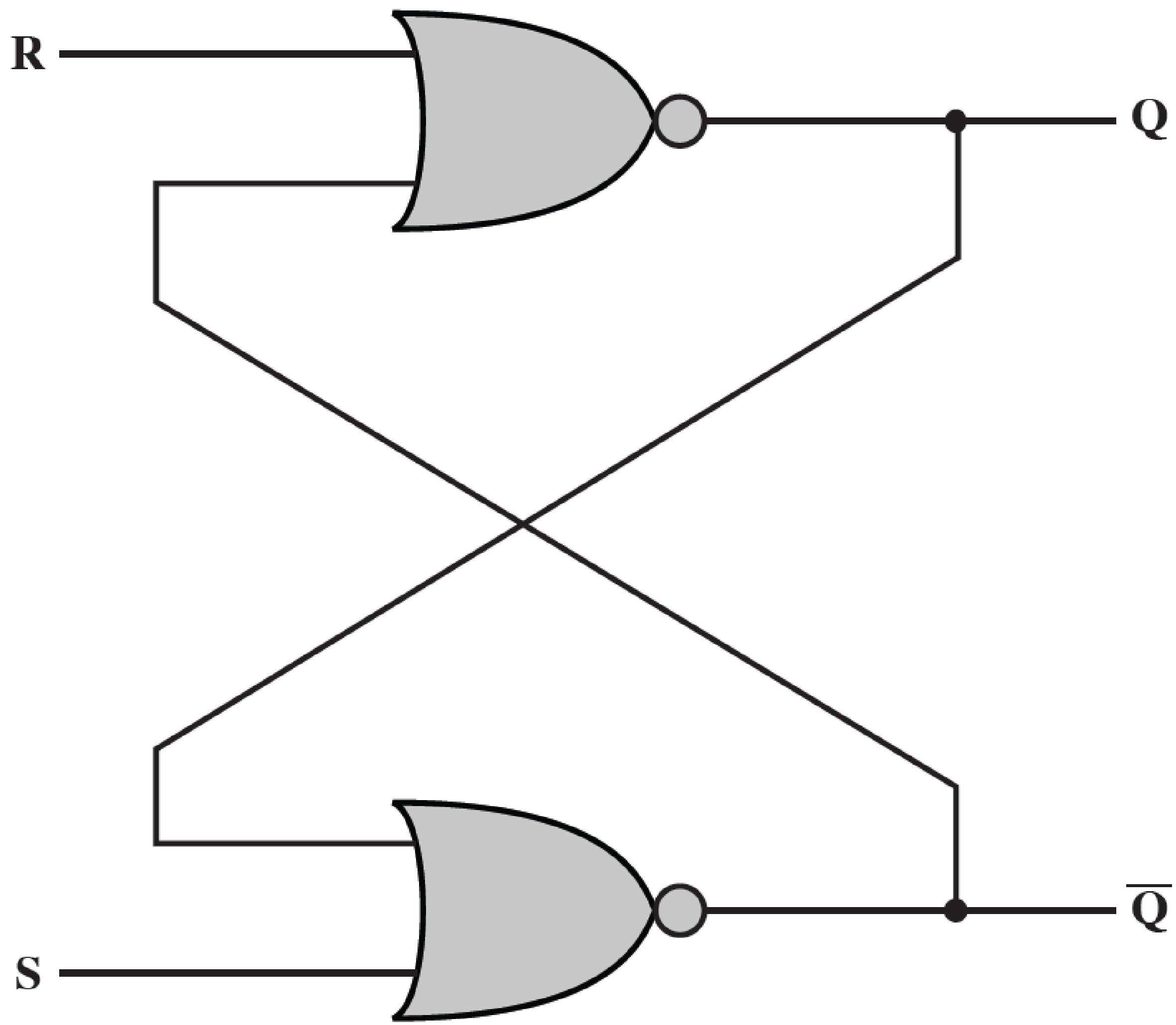
- Combinatorial Circuits implement most important function of computers
- But they are state-less, thus they only depend on the input
- No memory function available
- Sequential Circuits:
  - The current output depends on the current input and the current state of that circuit

Flip-Flop

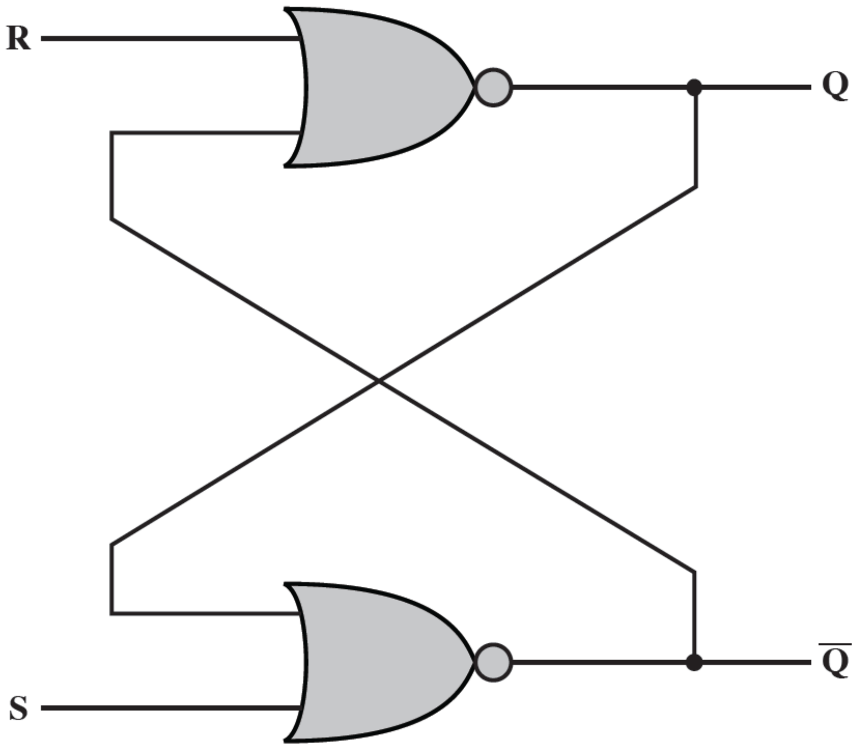
- Simplest form of sequential circuit

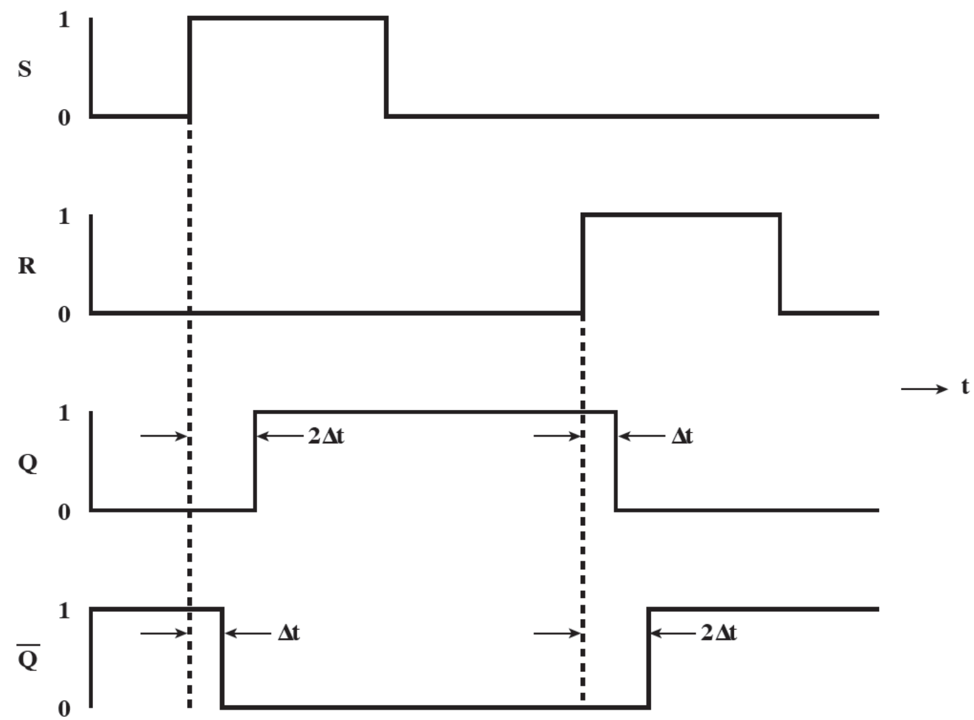
- There are a variety of flip-flops, all of which share two properties:
  - The flip-flop is a bistable device. It exists in one of two states and, in the absence of input, remains in that state. Thus, the flip-flop can function as a 1-bit memory.
  - The flip-flop has two outputs, which are always the complements of each other.

S-R Latch



S-R Latch



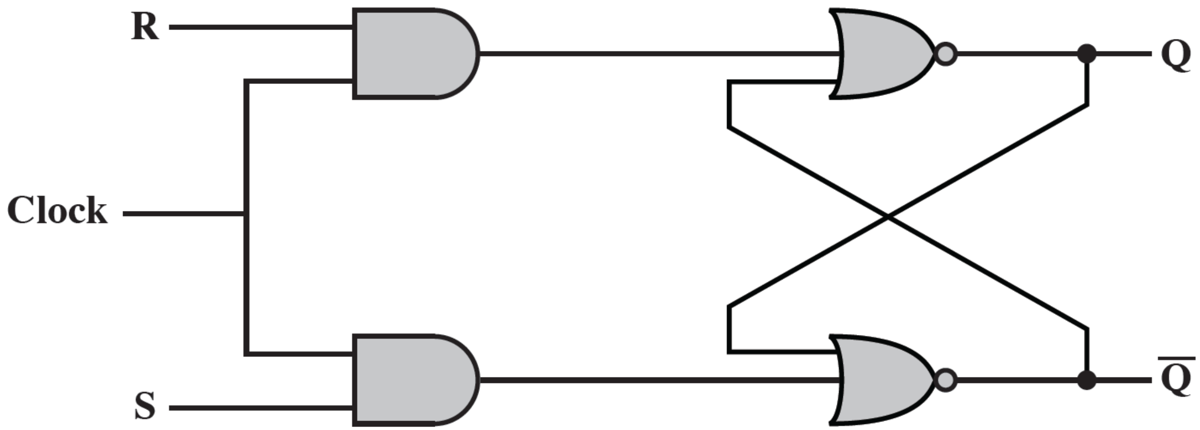


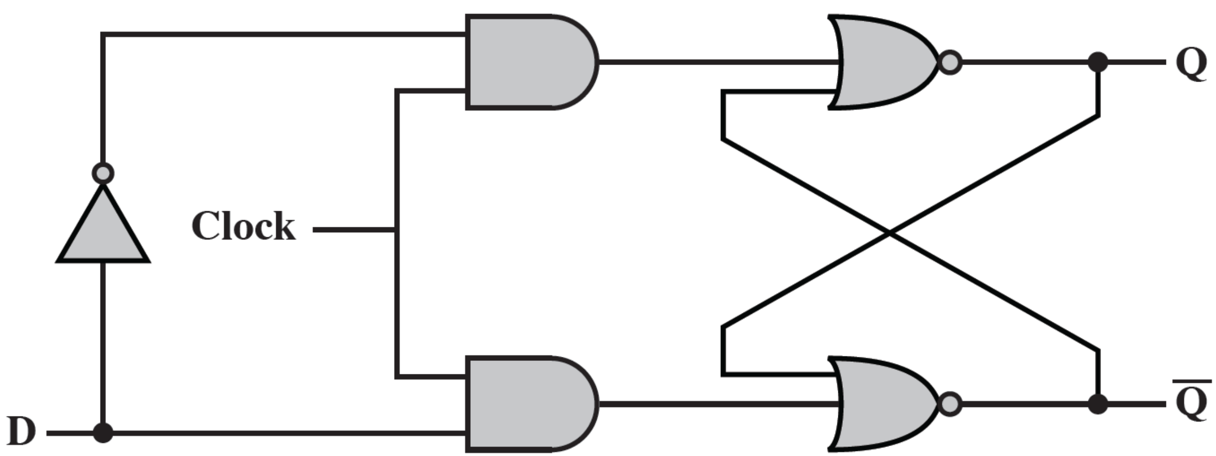
Characteristic Table of the S-R Latch

SR	Q_n	Q_{n+1}
00	0	0
00	1	1
01	0	0
01	1	0
10	0	1
10	1	1
11	0	-
11	1	-

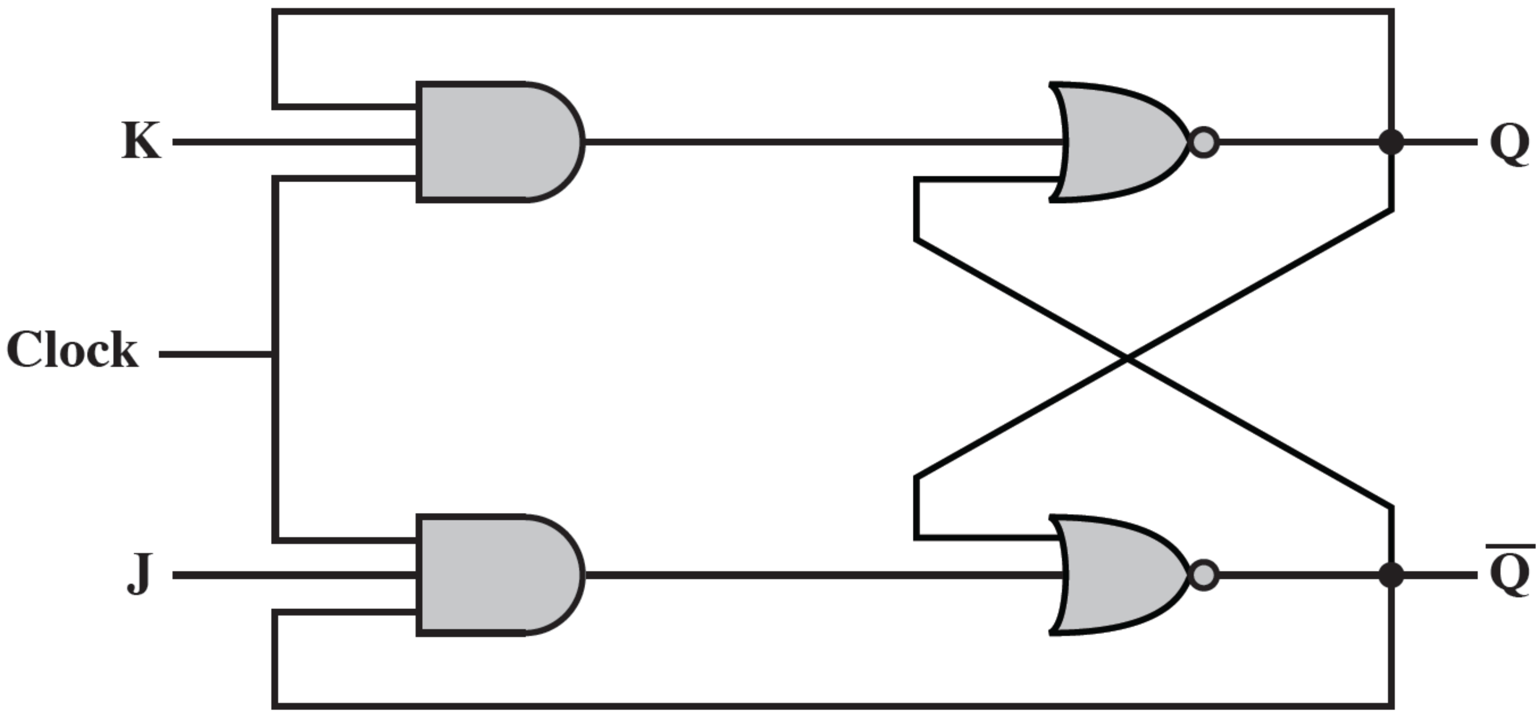
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	-

Clocked S-R Flip-Flop & D Flip-Flop

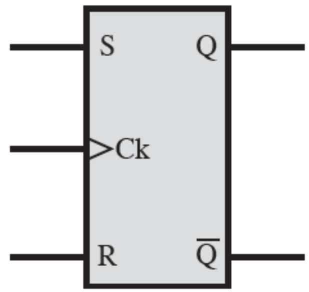




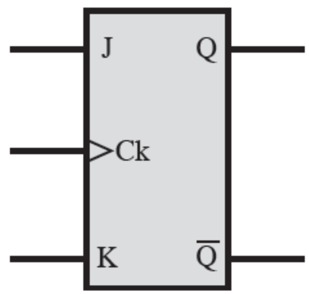
J-K Flip-Flop



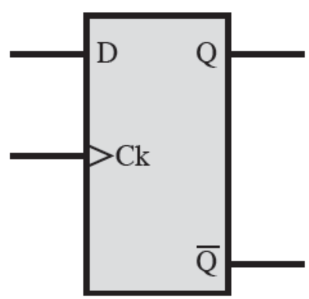
Flip-Flop Overview

<table>
<tr><th>Name</th><th>Graphical Symbol</th><th>Truth Table</th></tr>
<tr><td>S-R</td><td>
</td><td>

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	–

</td></tr>
<tr><td>J-K</td><td>
</td><td>

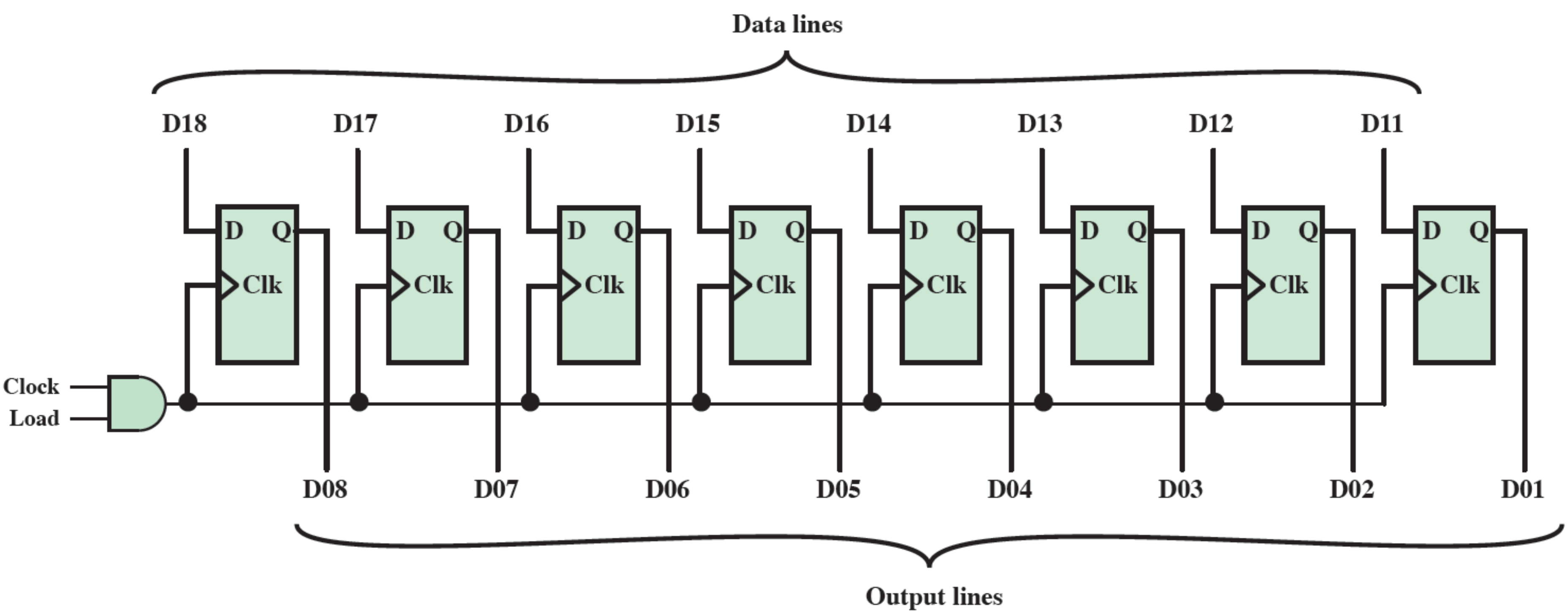
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

</td></tr>
<tr><td>D</td><td>
</td><td>

D	Q_{n+1}
0	0
1	1

</td></tr>
</table>

8-Bit Parallel Register



5-Bit Shift Register