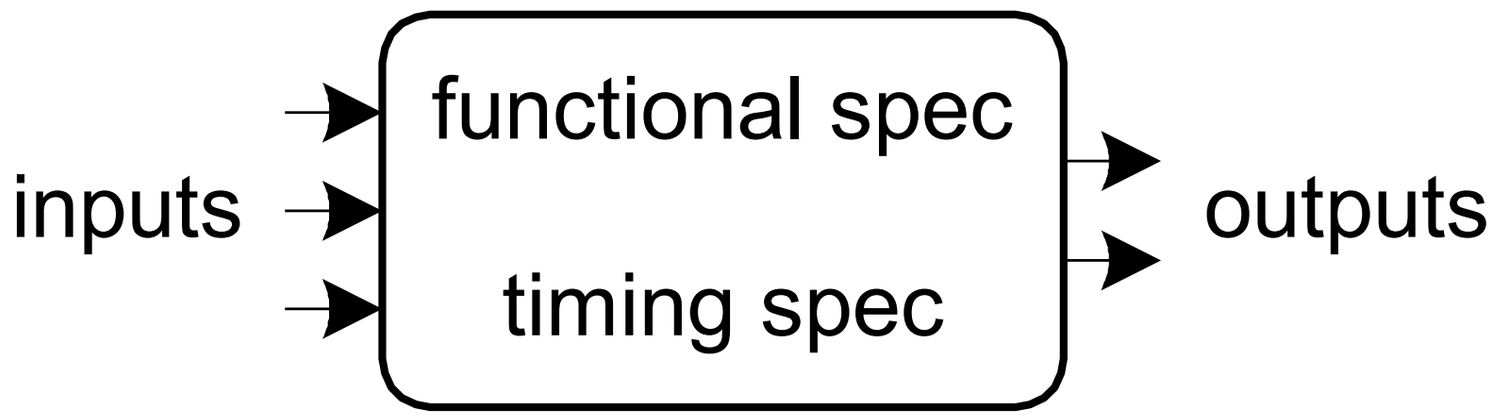
Combinational Circuits

Logic Design

Introduction

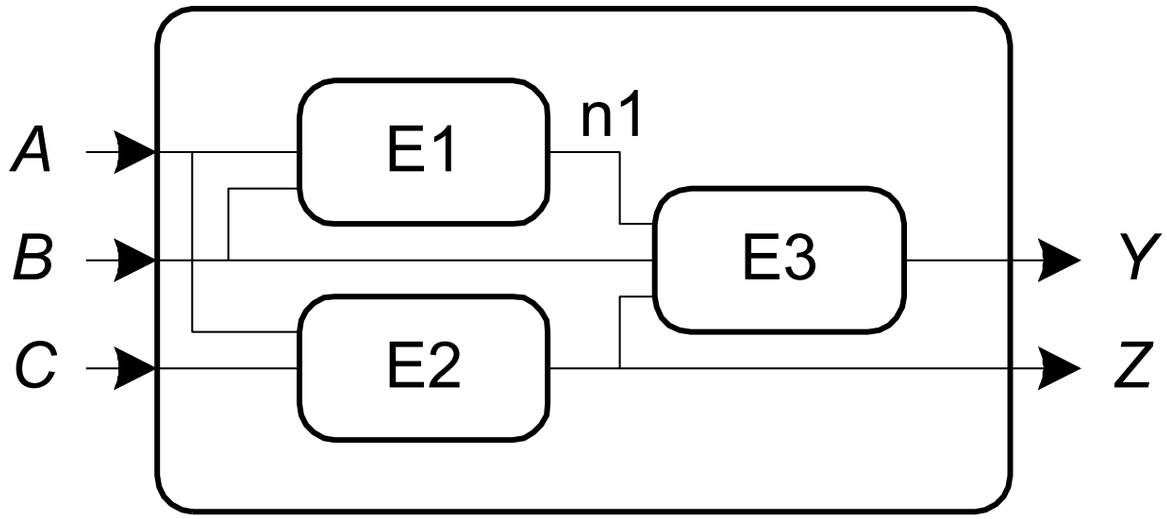
A logic circuit is composed of:

- Inputs
- Outputs
- Functional specification
- Timing specification



Circuits

- Nodes
  - Inputs: *A*, *B*, *C*
  - Outputs: *Y*, *Z*
  - Internal: n1
- Circuit elements
  - E1, E2, E3
  - Each a circuit

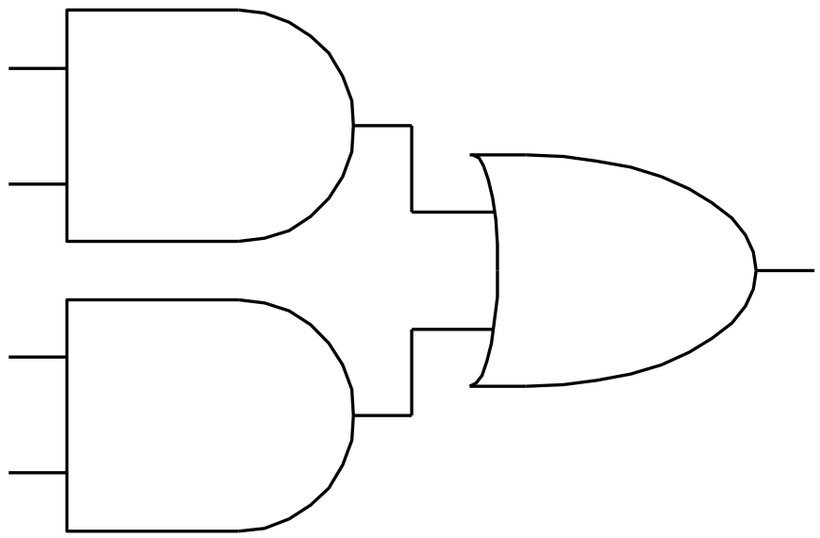


Types of Logic Circuits

- **Combinational Logic**
  - –Memoryless
  - –Outputs determined by current values of inputs
- **Sequential Logic**
  - –Has memory
  - –Outputs determined by previous and current values of inputs

Combinational Composition

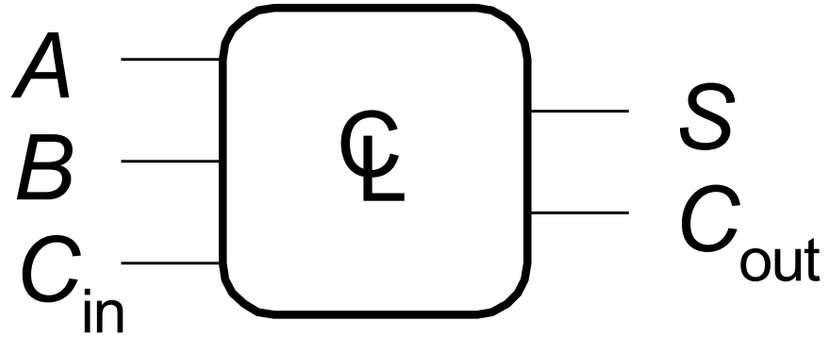
- Every element is combinational
- Every node is either an input or connects to *exactly one* output
- The circuit contains no **cyclic paths**
- **Example:**



Example

$S = \mathrm{F}(A, B, C_{\text{in}})$

$C_{\text{out}} = \mathrm{F}(A, B, C_{\text{in}})$



$$S = A \oplus B \oplus C_{\text{in}}$$

$$C_{\text{out}} = AB + AC_{\text{in}} + BC_{\text{in}}$$

Design Procedure

- Specification
  - Write a specification for the circuit if one is not already available
- Formulation
  - Derive a truth table
- Optimization
  - Apply 2-level and multiple-level optimization
- Technology Mapping
  - Map the logic diagram to the implementation technology selected
- Verification
  - Verify the correctness of the final design manually or using simulation

Example: BCD to Excess-3 Code Converter

- Specification
  - —BCD code words for digits 0 through 9
    - 4-bit patterns 0000 to 1001, respectively
  - —Excess-3 code words for digits 0 through 9
    - 4-bit patterns consisting of 3 (binary 0011) added to each BCD code word

Example: BCD to Excess-3 Code Converter

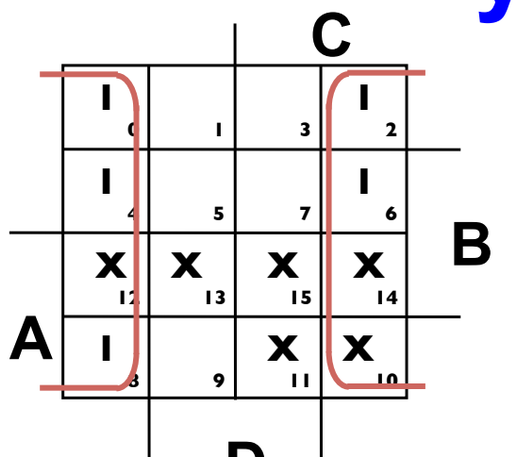
- Formulation
  - Variables
    - BCD: A,B,C,D
  - Variables
    - Excess-3: W,X,Y,Z
  - Don't Cares
    - BCD 1010 to 1111

Input BCD A B C D	Output Excess-3 WXYZ
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0

Example: BCD to Excess-3 Code Converter

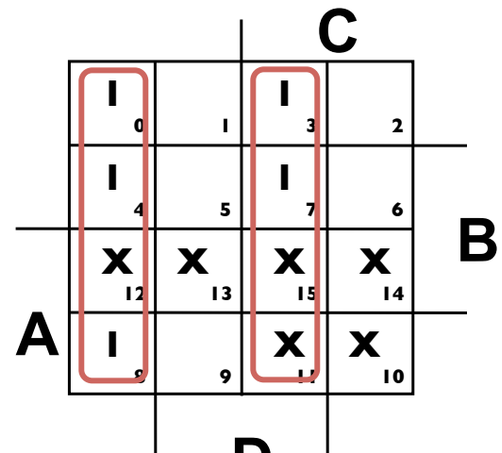
- Optimization

z



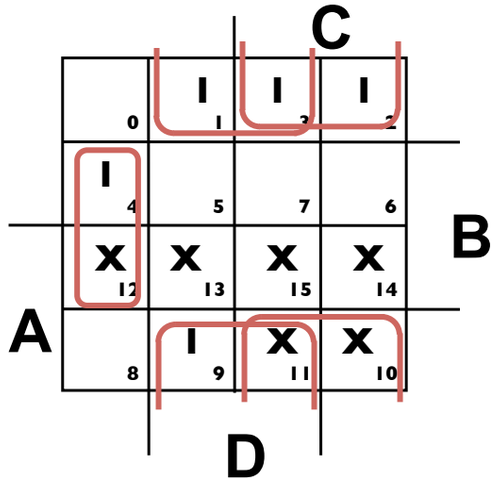
Z=D’

y



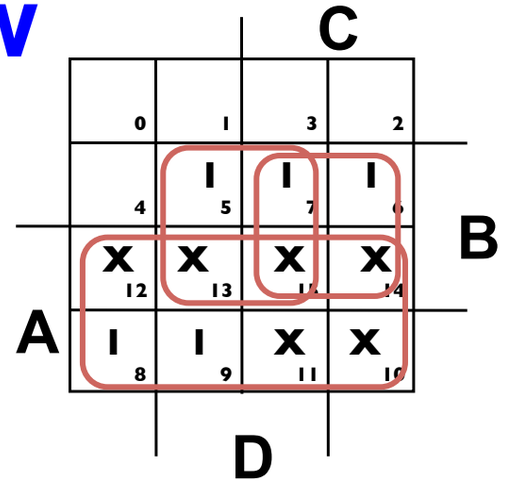
Y=CD+C’D’

X



W = B’C+B’D+BC’D’

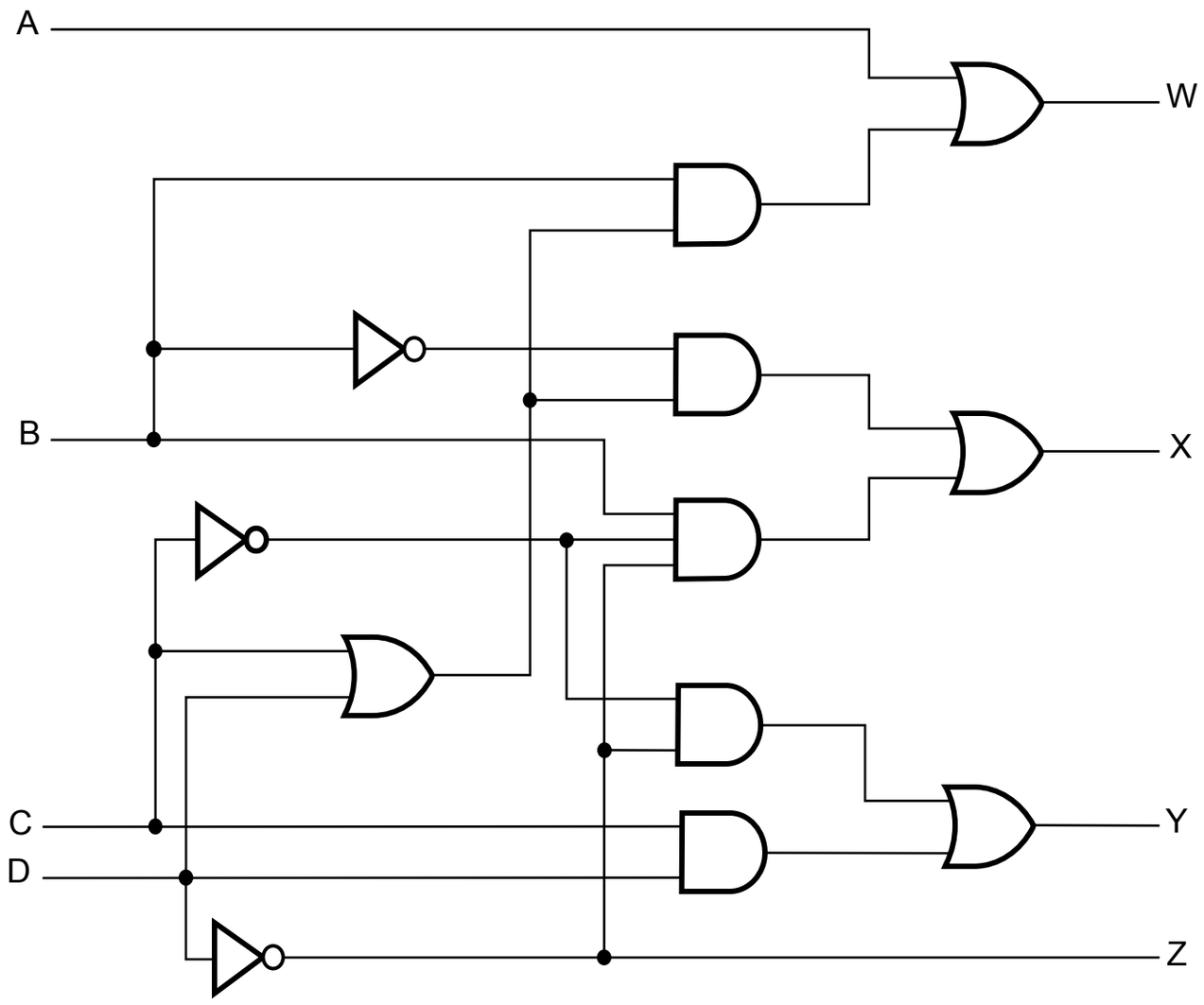
W



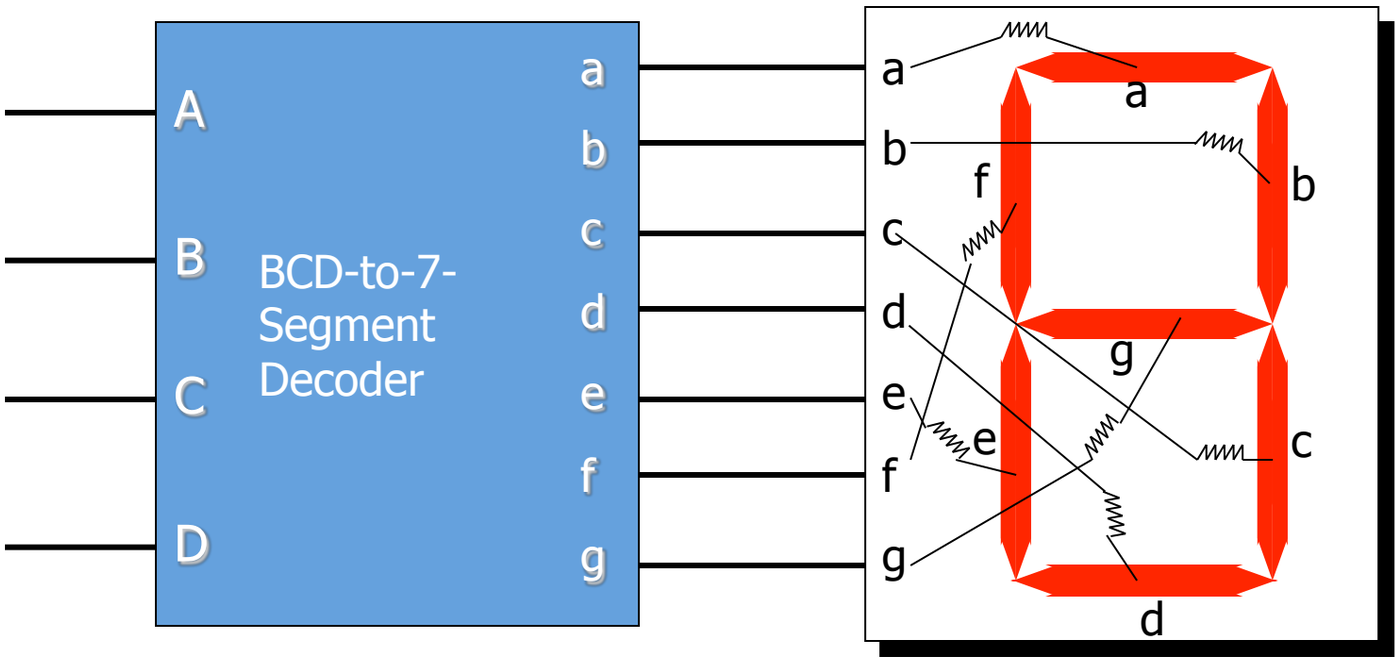
W = A+ BC+BD

Example: BCD to Excess-3 Code Converter

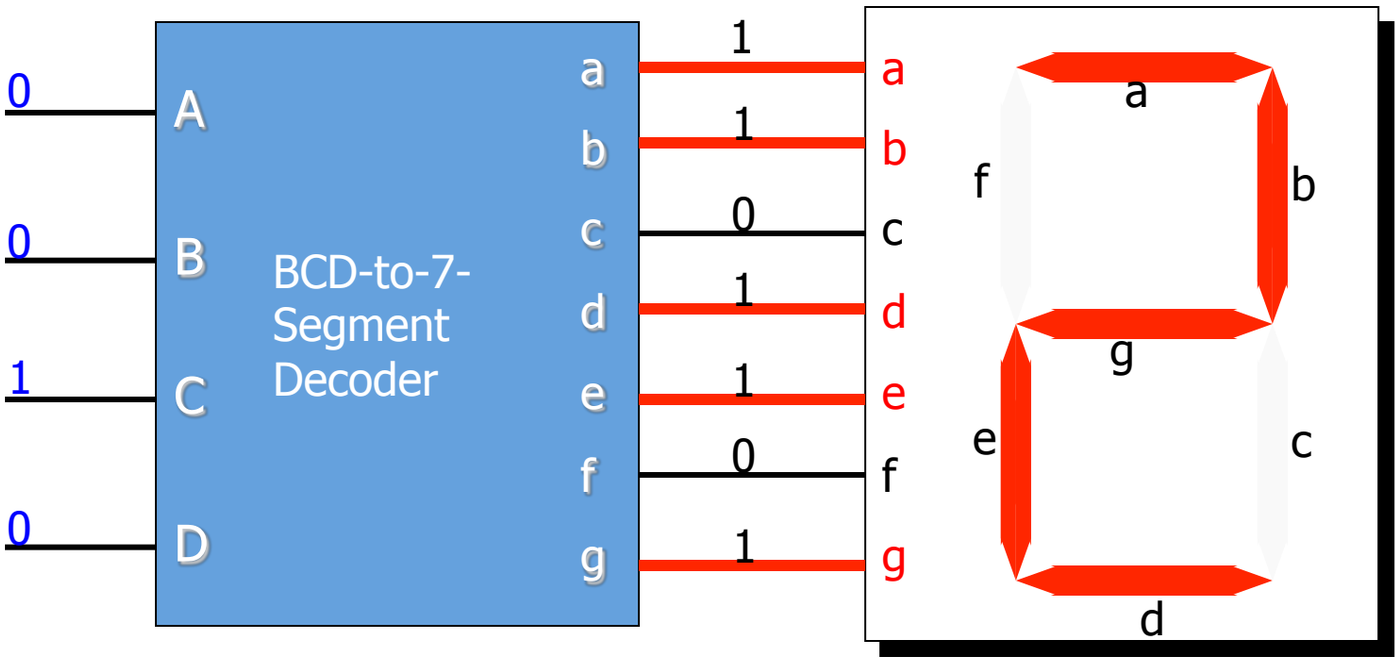
- Technology Mapping



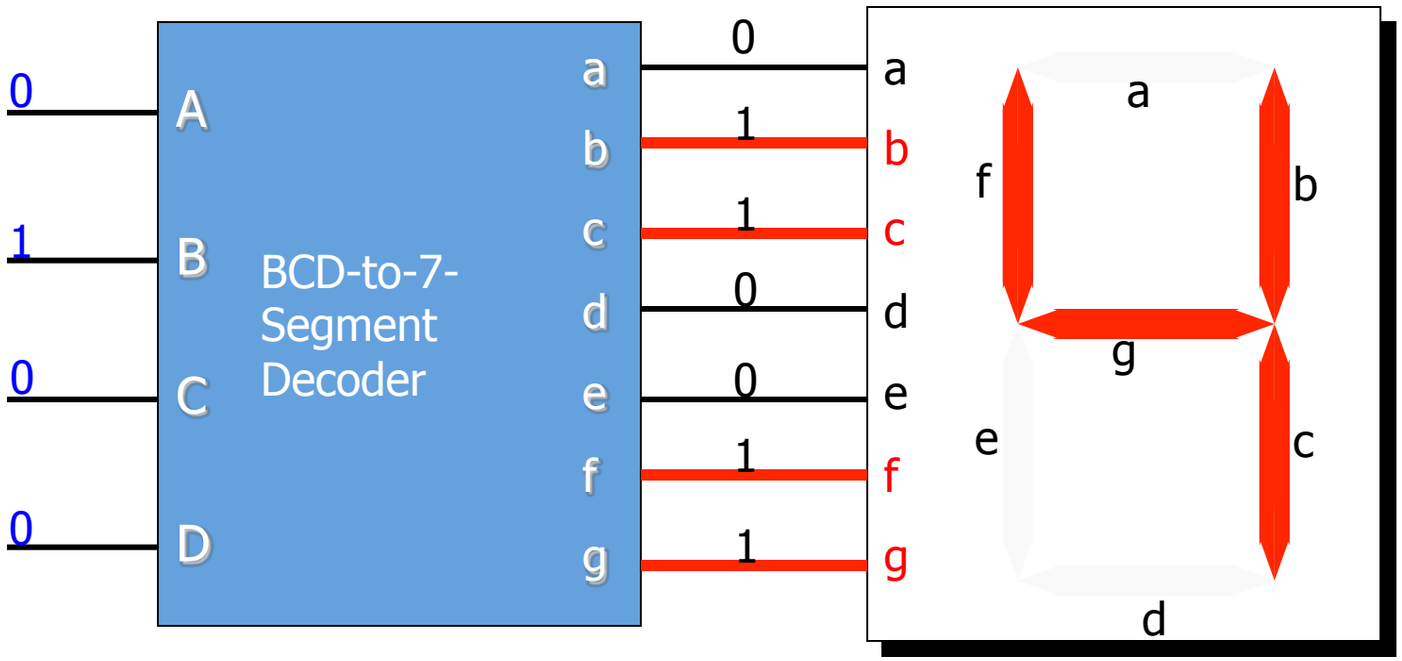
Example: BCD-to-7-Segment Decoder



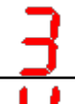
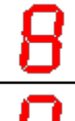
Example: BCD-to-7-Segment Decoder



Example: BCD-to-7-Segment Decoder

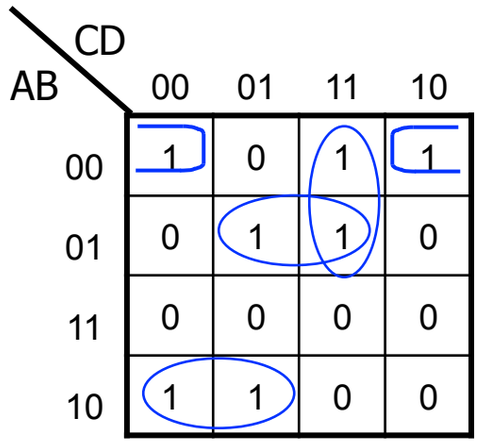


Example: BCD-to-7-Segment Decoder

	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	0	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	0	0	1	1	
>10	All other inputs				0	0	0	0	0	0	0	

Example: BCD-to-7-Segment Decoder

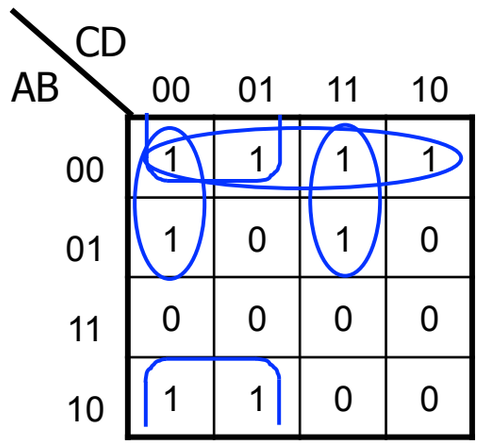
	A	B	C	D	a
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
>10	All other inputs				0



$$a = \overline{A}\,\overline{B}\,\overline{D} + \overline{A}CD + \overline{A}B\overline{D} + A\overline{B}\,\overline{C}$$

Example: BCD-to-7-Segment Decoder

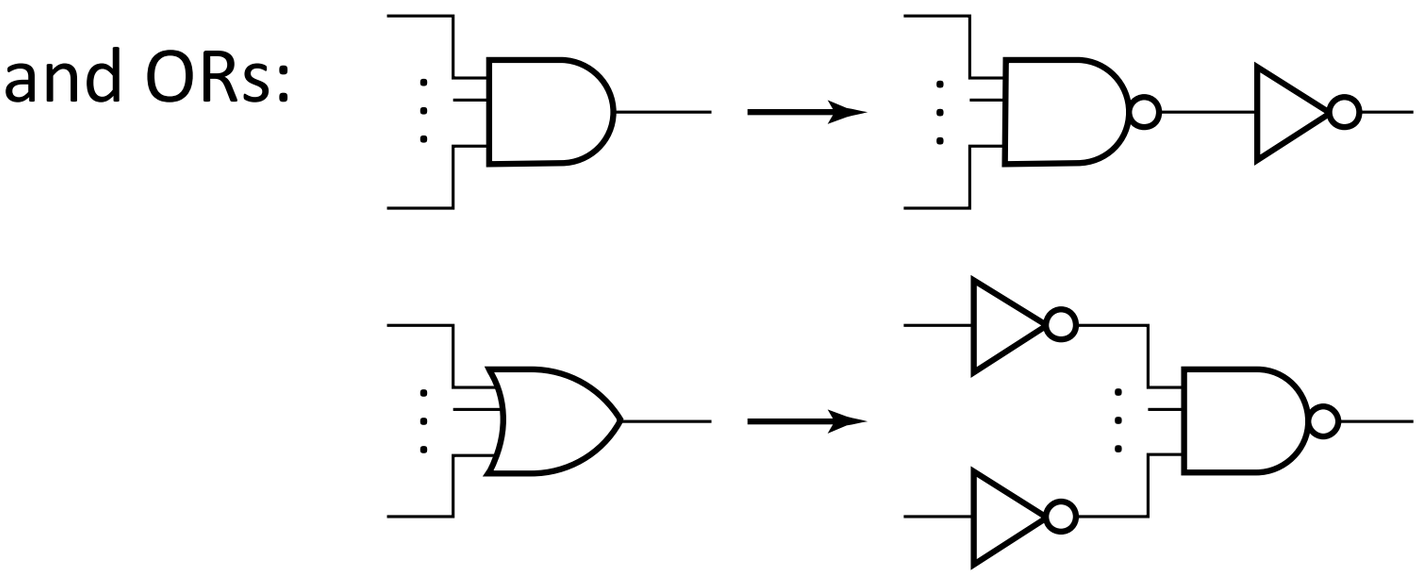
	A	B	C	D	b
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
>10	All other inputs				0



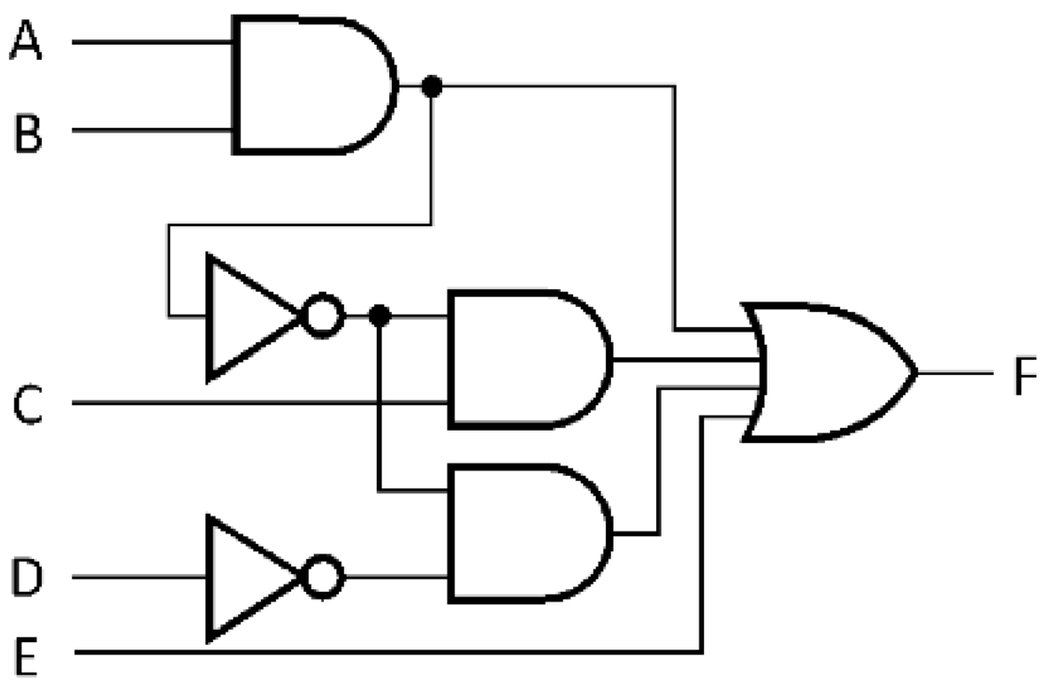
$$b = \overline{B}\,\overline{C} + \overline{A}\,\overline{B} + \overline{A}\,\overline{C}\,\overline{D} + \overline{A}CD$$

NAND Mapping Algorithm

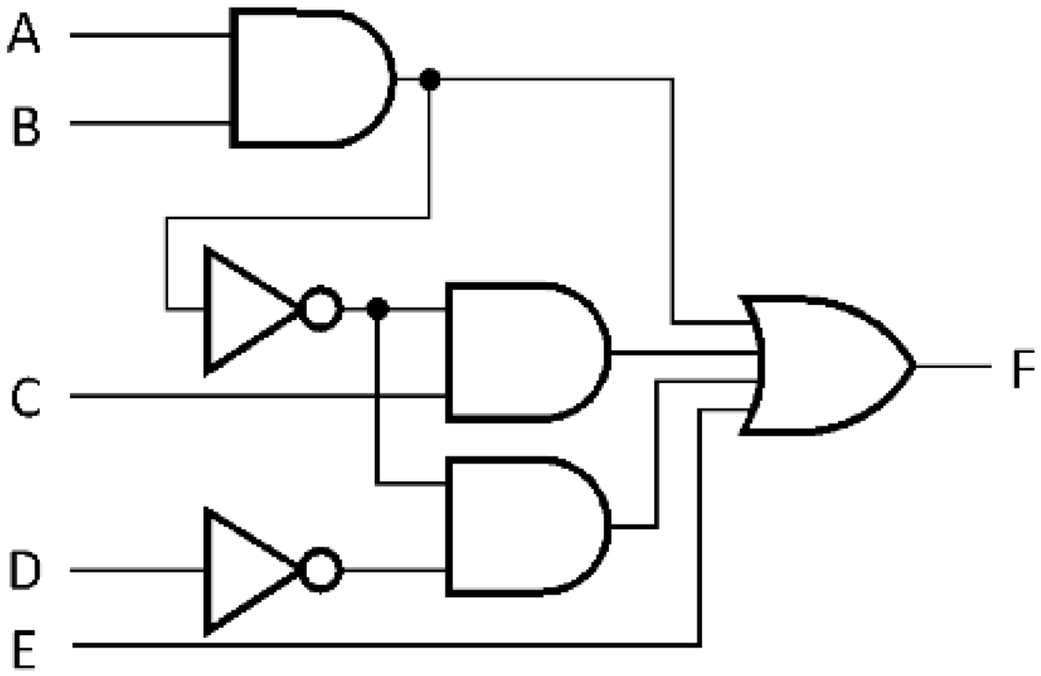
- Replace ANDs and ORs:

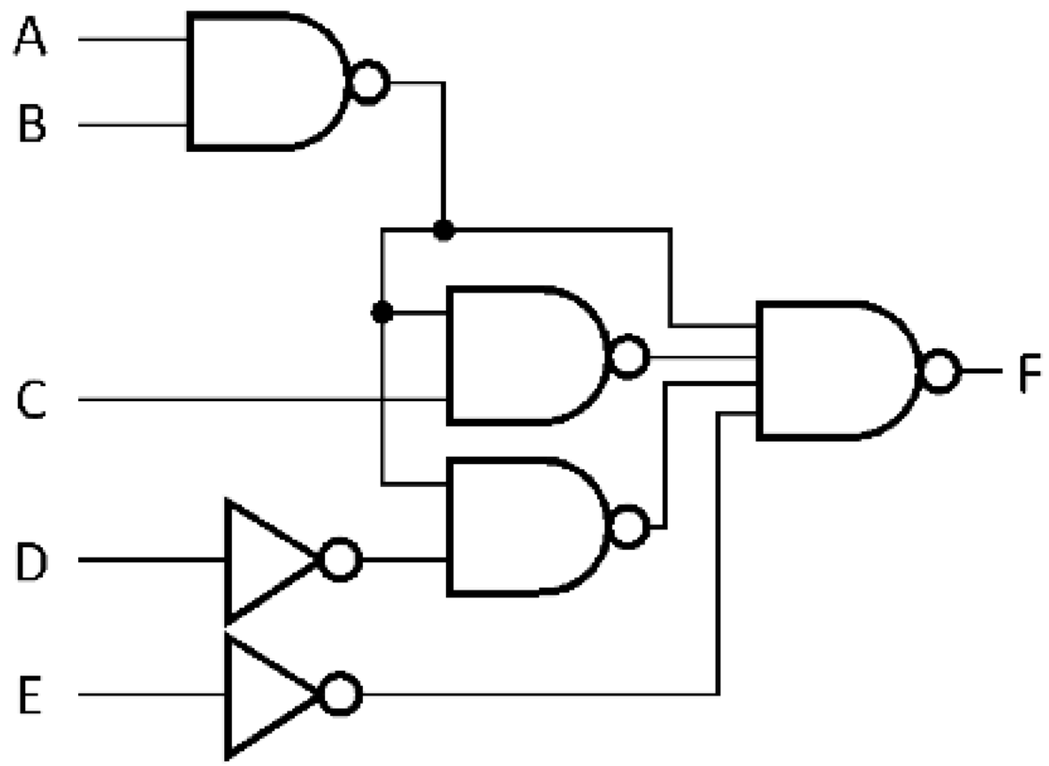


NAND Mapping Example



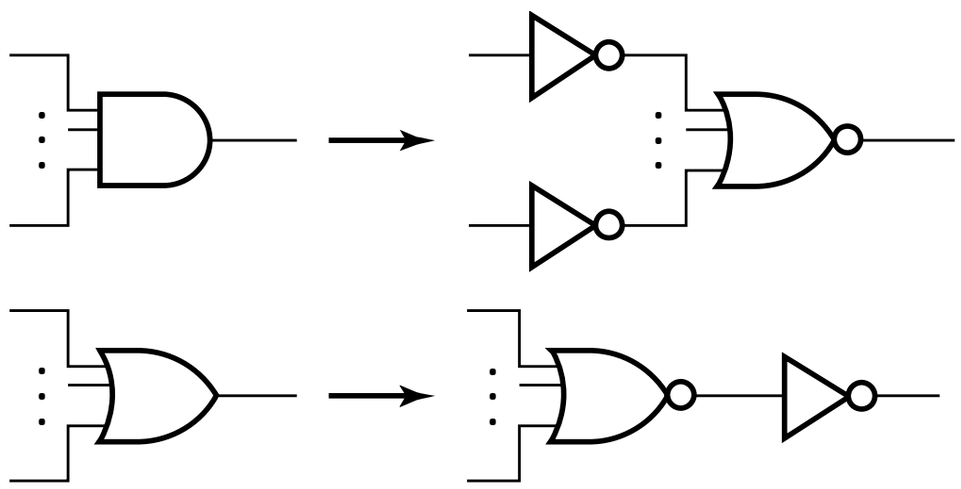
NAND Mapping Example



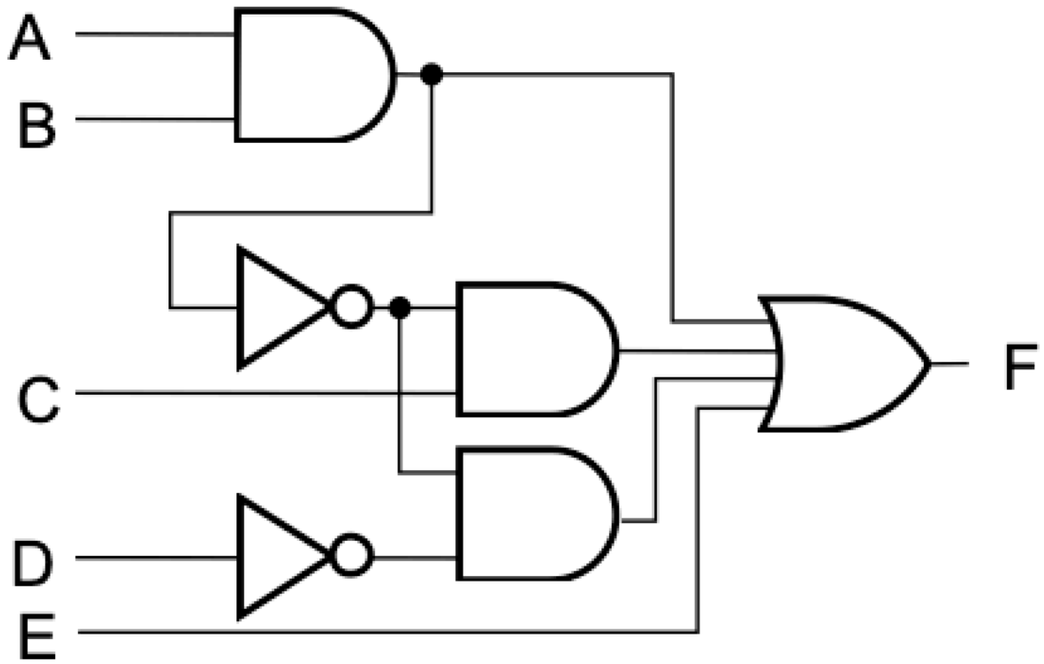


NOR Mapping Example

- Replace ANDs and ORs:



NOR Mapping Example



NOR Mapping Example