Introduction to Digital Systems

Part I (4 lectures)

2020/2021

Introduction

Number Systems and Codes

Combinational Logic Design Principles

Arnaldo Oliveira, Augusto Silva, Iouliia Skliarova

Lecture 4 contents

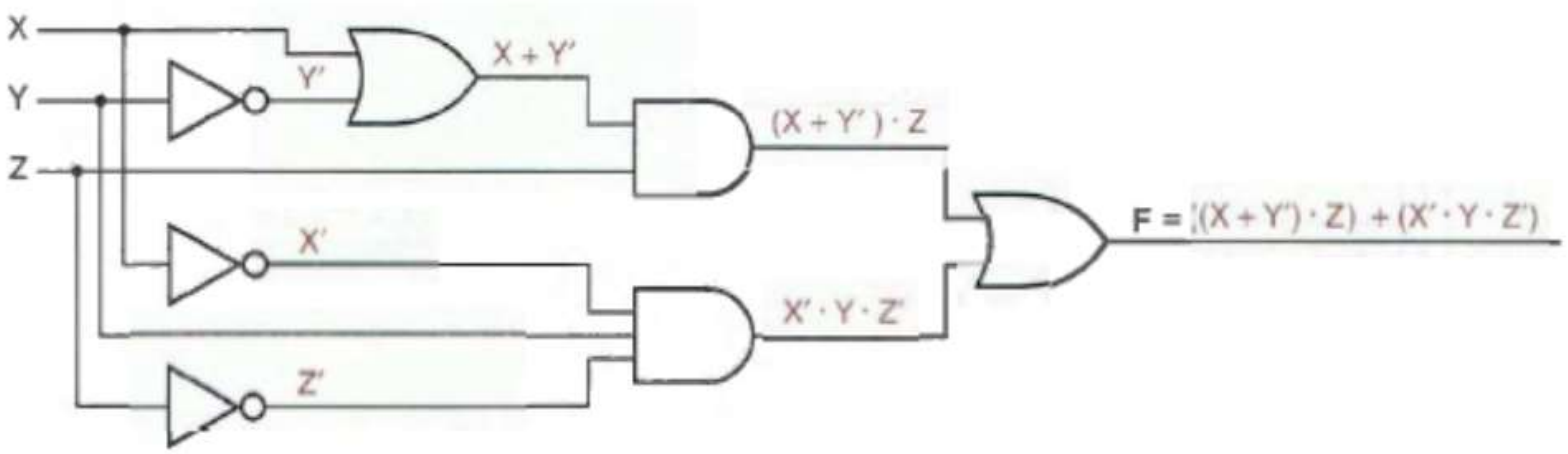
- Combinational-circuit analysis
- Minimization of Boolean functions
  - Karnaugh maps
- Combinational-circuit synthesis

Combinatorial Circuit Analysis

- In combinational circuit analysis we start with a logic diagram and proceed to a formal description of the function performed by that circuit, such as a truth table or a logic expression.
- To analyze a simple combinational circuit, use an algebraic approach, i.e. build up a parenthesized logic expression corresponding to the logic operators and structure of the circuit.
  - Start at the circuit inputs and propagate expressions through gates toward the output.
  - Using the theorems of switching algebra, you may simplify the expressions as you go, or you may defer all algebraic manipulations until an output expression is obtained.

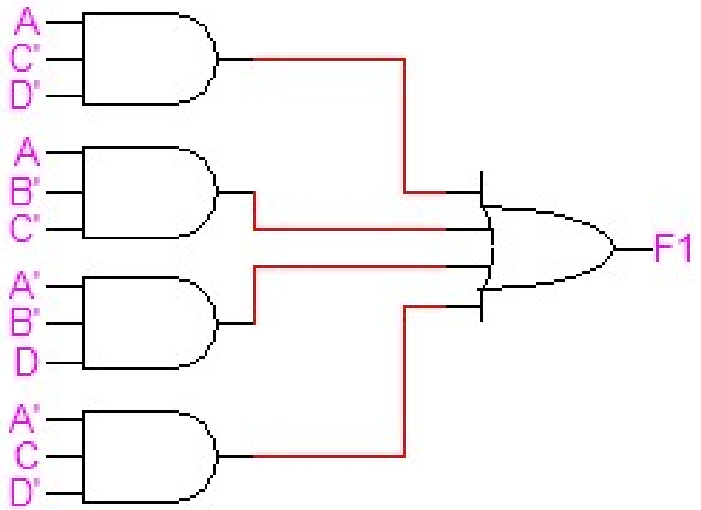
Combinatorial Circuit Analysis (cont.)

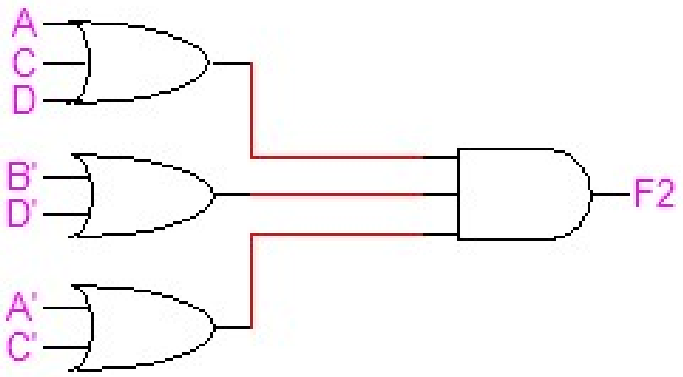
- What function is performed by the following circuit?
- Is it possible to reduce the number of components?
- How many delay levels does the circuit have?
- Is it possible to reduce the number of delay levels?
- Is it possible to implement the circuit with NAND gates only?



Combinatorial Circuit Analysis (cont.)

- Compare the following circuits in terms of implementation costs.
- Do the circuits implement the same function?





Minimization of Boolean Functions

- Try minimizing the following function:

$$f(x,y,z) = \bar{x} \cdot y \cdot \bar{z} + \bar{x} \cdot y \cdot z + x \cdot z + \bar{y} \cdot z + y \cdot \bar{z}$$

- Depending on the sequence of theorems you apply, you might get:

$$f(x,y,z) = \bar{x} \cdot y + x \cdot z + \bar{y} \cdot z + y \cdot \bar{z}$$ 4 terms, 8 literals

  - Or:

$$\begin{aligned} f(x,y,z) &= \bar{x} \cdot y \cdot \bar{z} + y \cdot \bar{z} + z \cdot (\bar{x} \cdot y + x + \bar{y}) = \\ &= y \cdot \bar{z} + z \cdot (y + x + \bar{y}) = \\ &= y \cdot \bar{z} + z = \\ &= y + z \end{aligned}$$ 2 literals

Minimization of Boolean Functions

- Algebraic minimization is prone to errors.
- Irreducible expressions may not be minimal.
- There may be more than one minimum expression.
- A more formal minimization method is needed.
- Minimization criteria:
  - Minimize the number of terms
  - Minimize the number of literals
  - Do not consider the cost of input inverters

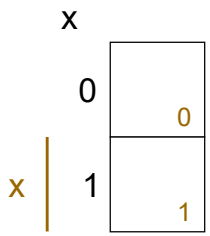
Minimizing Sums of Products

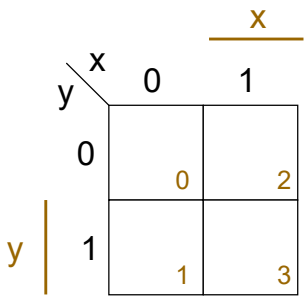
- A **minimal sum** of a logic function *f* is a sum-of-products expression for *f* such that no sum-of-products expression for *f* has fewer product terms, and any sum-of-products expression with the same number of product terms has at least as many literals.
- The minimal sum has the fewest possible product terms (first-level gates and second-level gate inputs) and, within that constraint, the fewest possible literals (first-level gate inputs).
- All minimization methods are based on the application of the combining theorem:

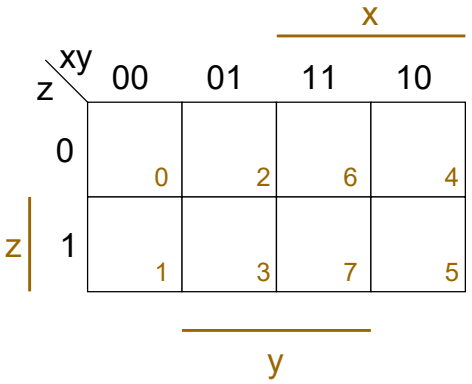
$$x \cdot y + x \cdot \bar{y} = x$$

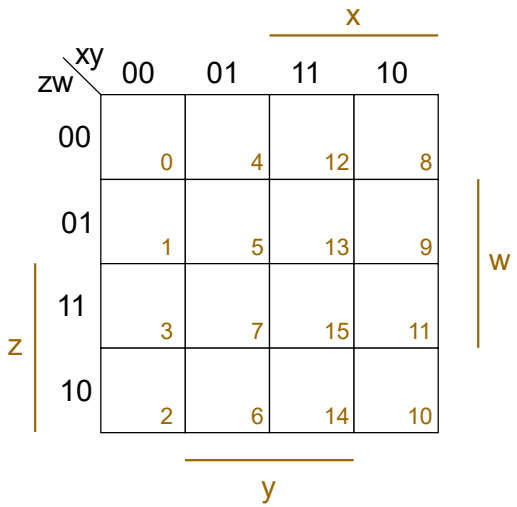
Karnaugh Maps

- A **Karnaugh map** is a graphical representation of the truth table of a Boolean function.









Karnaugh Map Structure

- The rows and columns of a Karnaugh map are labeled (coded) so that the input combination for any cell is easily determined from the row and column headings for that cell.
- Rows and columns are labeled with Gray codes for the respective input variables.
- All pairs of cells that are adjacent in a Karnaugh map, have codes with the Hamming distance = 1.
- The small number inside each cell is the corresponding minterm number in the truth table.
  - For example, cell 5 in the 3-variable (x,y,z) map corresponds to the truth-table row in which x y z = 101_2=5_{10}.
- When we draw the Karnaugh map for a given function, each cell of the map contains the information from the like-numbered row of the function's truth table - a 0 if the function is 0 for that input combination, a 1 otherwise.

Combining Cells

- Each input combination with a "1" in the truth table corresponds to a minterm in the logic function's canonical sum. Since pairs of adjacent "1" cells in the Kamaugh map have minterms that differ in only one variable, the minterm pairs can be combined into a single product term using the generalization of the combining theorem: $term \cdot x + term \cdot \bar{x} = term$.
- Thus, we can use a Kamaugh map to simplify the canonical sum of a logic function.
- In many logic functions the cell-combining procedure can be extended to combine more than two 1-cells into a single product term.
- The number of cells combined will always be a power of 2.
- In general, 2^i 1-cells may be combined (circled) to form a product term containing $n - i$ literals, where n is the number of variables in the function.

“Reading” Combined 1-Cells

- We can determine the literals of the corresponding product terms directly from the map; for each variable we make the following determination:
  - If a circle covers only areas of the map where the variable is 0, then the variable is complemented in the product term.
  - If a circle covers only areas of the map where the variable is 1, then the variable is uncomplemented in the product term.
  - If a circle covers areas of the map where the variable is 0 as well as areas where it is 1, then the variable does not appear in the product term.
- Finally, a sum-of-products expression for a function must contain product terms (circled sets of 1-cells) that cover all of the 1s and none of the 0s on the map.

Definitions

- A logic function $P(x_0, ..., x_{n-1})$ **implies** a logic function $F(x_0, ..., x_{n-1})$ if for ever input combination such that P = 1, then F = 1 also.
- A **prime implicant** of a logic function $F(x_0, ..., x_{n-1})$ is a normal product term $P(x_0, ..., x_{n-1})$ that implies F**,** such that if any variable is removed from P, then the resulting product term does not imply F.
- A **distinguished cell** of a logic function is an input combination that is covered by only one prime implicant.
- An **essential prime implicant** of a logic function is a prime implicant that covers one or more distinguished cells.

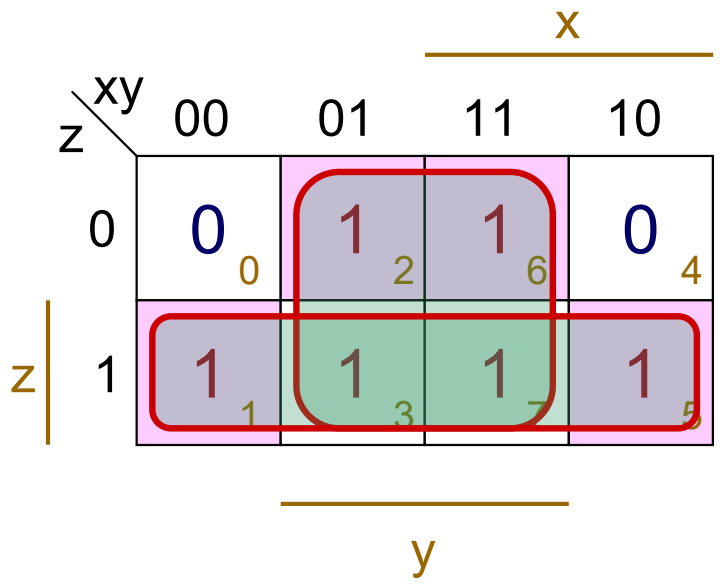
Minimizing Sums of Products with Karnaung Maps

- For an *n*-variable logic function, fill in the Karnaugh map with 0s and 1s.
- Locate all the **prime implicants**.
  - In terms of a Karnaugh map, a **prime implicant** is a **circled set of 2^i (i=[0..2^n-1], i.e. 1, 2, 4, 8,…) adjacent 1-cells**, such that if we try to make it larger (covering twice as many cells), it covers one or more 0s.
- Identify all the **distinguished 1-cells**.
  - A **distinguished 1-cell** is an input combination that is covered by only one prime implicant.
- Identify all **essential prime implicants**.
  - An **essential prime implicant** is a prime implicant that covers one or more distinguished 1-cells.
- Since an essential prime implicant is the **only** prime implicant that covers some 1-cell, it **must** be included in every **minimal sum** for the logic function.
- Determine how to cover the 1-cells, if any, that are not covered by the essential prime implicants.
  - Choose the smallest number of “bigger” (covering more 1-cells) prime implicants.

Karnaugh Maps for 3 variables

$$f(x,y,z) = \bar{x} \cdot y \cdot \bar{z} + \bar{x} \cdot y \cdot z + x \cdot z + \bar{y} \cdot z + y \cdot \bar{z}$$

	x	y	z	f(x,y,z)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

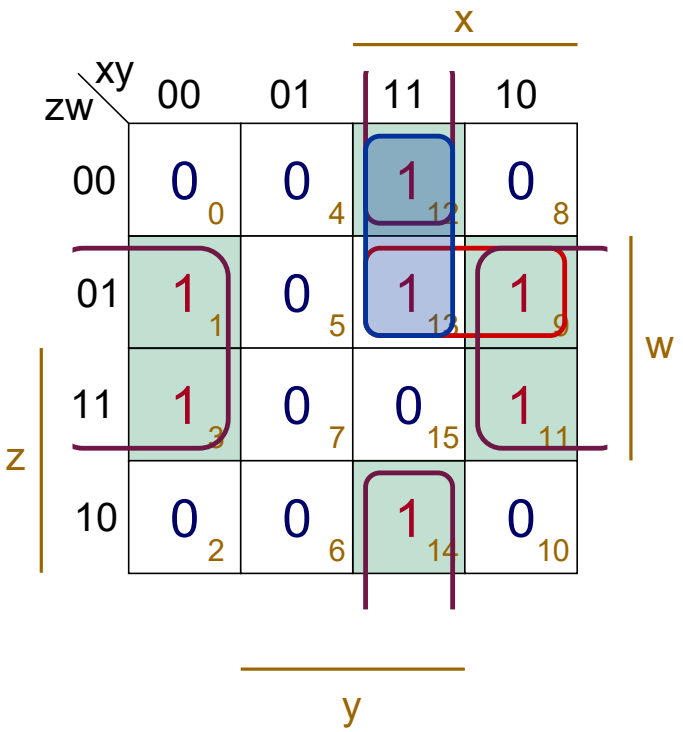


$$f(x,y,z) = y + z$$

2 prime implicants
4 distinguished 1-cells
2 essential prime implicants

Karnaugh Maps for 4 variables

	x	y	z	w	f(x,y,z,w)
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0



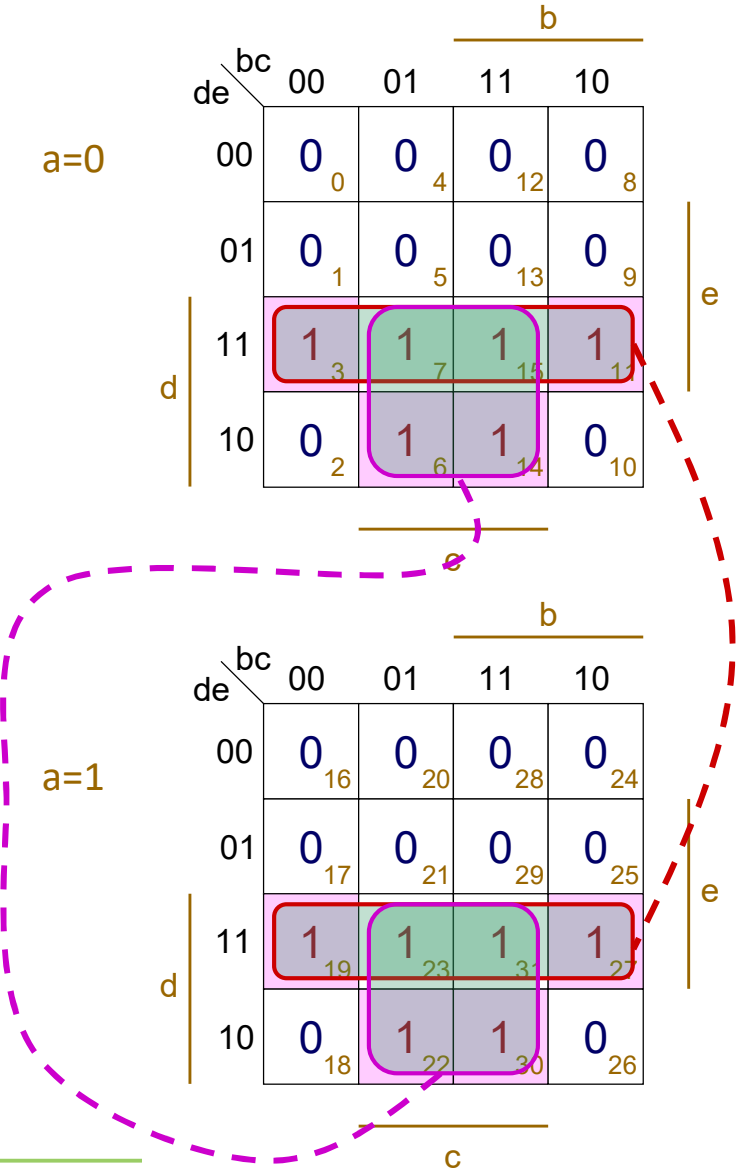
4 prime implicants

4 distinguished 1-cells

2 essential prime implicants

$$f(x,y,z,w) = \overline{y} \cdot w + x \cdot y \cdot \overline{w} + x \cdot y \cdot \overline{z}$$

Karnaugh Maps for 5 variables



$$f(a,b,c,d,e) = \overline{a}\cdot\overline{b}\cdot d\cdot e + \overline{a}\cdot c\cdot d + \\ + \overline{a}\cdot b\cdot d\cdot e + a\cdot d\cdot e + a\cdot d\cdot c\cdot \overline{e}$$

- 2 prime implicants
- 8 distinguished 1-cells
- 2 essential prime implicants

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

Minimizing Products of Sums

- A **minimal product** of a logic function *f* is a product-of-sums expression for *f* such that no product-of-sums expression for *f* has fewer sum terms, and any product-of-sums expression with the same number of sum terms has at least as many literals.
- The minimal product has the fewest possible sum terms (first-level gates and second-level gate inputs) and, within that constraint, the fewest possible literals (first-level gate inputs).
- All minimization methods are based on the application of the combining theorem:

$$(x + y) \cdot (x + \bar{y}) = x$$

Minimizing Products of Sums with Karnaung Maps

- For an *n*-variable logic function, fill in the Karnaugh map with 0s and 1s.
- Locate all the **prime implicates**.
  - In terms of a Karnaugh map, a **prime implicate** is a **circled set of 2^i (i=[0..2^n-1], i.e. 1, 2, 4, 8,…) adjacent 0-cells**, such that if we try to make it larger (covering twice as many cells), it covers one or more 1s.
- Identify all the **distinguished 0-cells**.
  - A **distinguished 0-cell** is an input combination that is covered by only one prime implicate.
- Identify all **essential prime implicates**.
  - An **essential prime implicate** is a prime implicate that covers one or more distinguished 0-cells.
- Since an essential prime implicate is the **only** prime implicate that covers some 0-cell, it **must** be included in every **minimal product** for the logic function.
- Determine how to cover the 0-cells, if any, that are not covered by the essential prime implicates.
  - Choose the smallest number of "bigger" (covering more 0-cells) prime implicates.

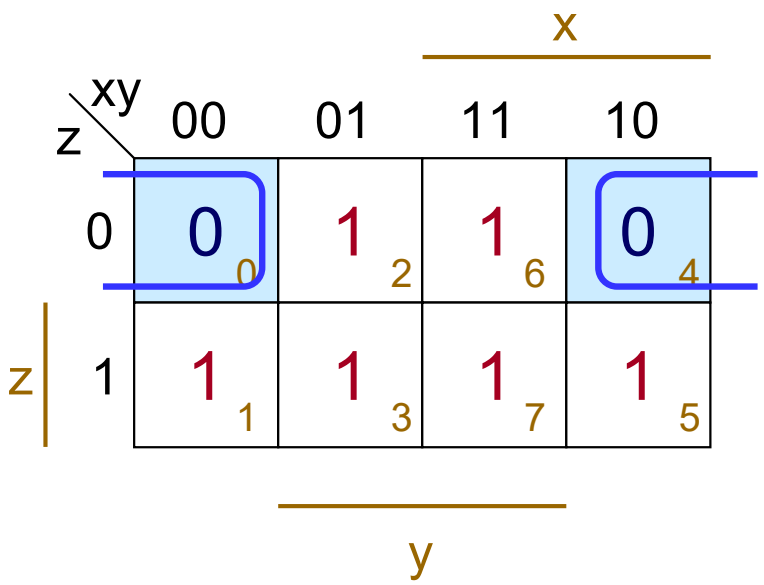
“Reading” Combined 0-Cells

- We can determine the literals of the corresponding sum terms directly from the map; for each variable we make the following determination:
  - If a circle covers only areas of the map where the variable is 1, then the variable is complemented in the sum term.
  - If a circle covers only areas of the map where the variable is 0, then the variable is uncomplemented in the sum term.
  - If a circle covers areas of the map where the variable is 0 as well as areas where it is 1, then the variable does not appear in the sum term.
- Finally, a product-of-sums expression for a function must contain sum terms (circled sets of 0-cells) that cover all of the 0s and none of the 1s on the map.

Karnaugh Maps for 3 variables

$$f(x,y,z) = \bar{x} \cdot y \cdot \bar{z} + \bar{x} \cdot y \cdot z + x \cdot z + \bar{y} \cdot z + y \cdot \bar{z}$$

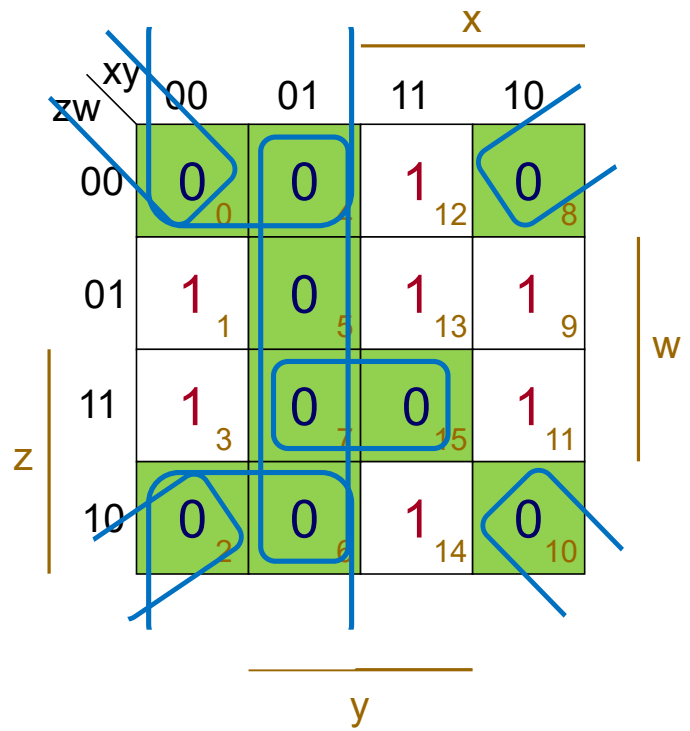
	x	y	z	f(x,y,z)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



$$f(x,y,z) = y + z$$

Karnaugh Maps for 4 variables

	x	y	z	w	f(x,y,z,w)
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0



4 prime implicates

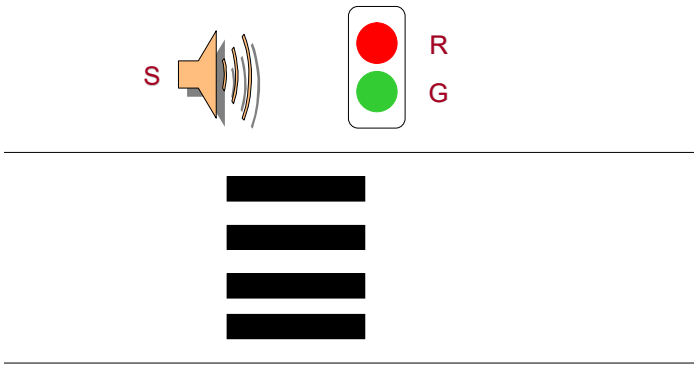
4 distinguished 0-cells

3 essential prime implicates

$$f(x,y,z,w) = (y+w)\cdot(x+\bar{y})\cdot(\bar{y}+\bar{z}+\bar{w})$$

Irrelevant Input Conditions (Don’t Cares)

- Sometimes the specification of a combinational circuit is such that its output doesn't matter for certain input combinations, called **don't-cares**.
- This may be true because the outputs really don't matter when these input combinations occur, or because these input combinations never occur in normal operation.
- The actual circuit will produce at the outputs a valid value (either 0 or 1) for all irrelevant conditions. However, the designer is “free” to assign 0 or 1 to the respective outputs.



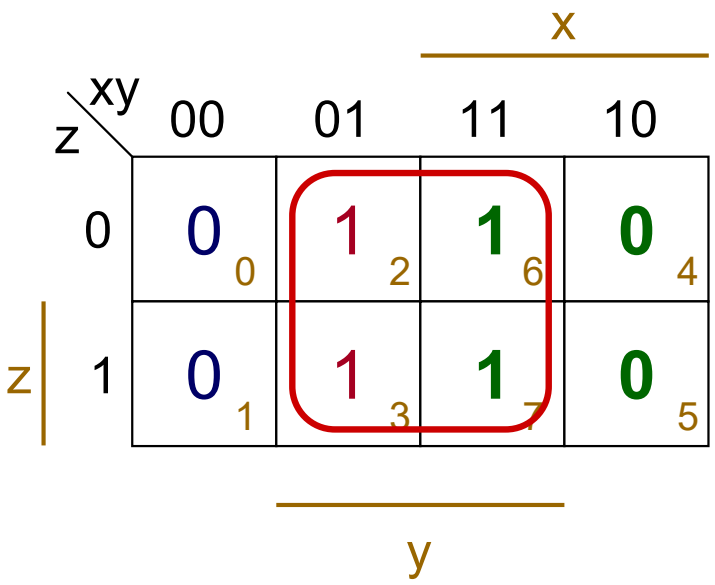
G (green)	R (red)	S (sound)
0	0	0
0	1	0
1	0	1
1	1	x

x – *don’t care*

Karnaugh Maps and Don't Cares

- In a Karnaugh map the irrelevant combinations should assume values that allow to reduce the number of literals in each of the prime implicants (implicates).

	x	y	z	f(x,y,z)
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	x
5	1	0	1	x
6	1	1	0	x
7	1	1	1	x



$$f(x, y, z) = y$$

Combinatorial Circuit Synthesis

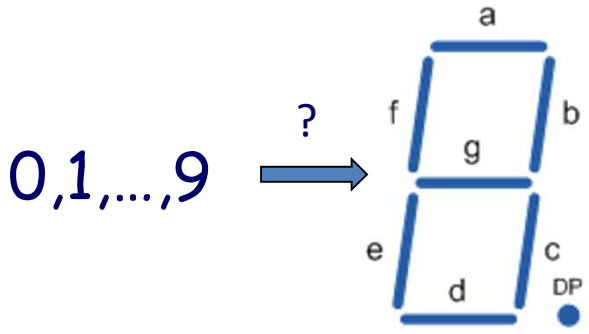
- In combinational circuit analysis we start with a logic diagram and proceed to a formal description of the function performed by that circuit, such as a truth table or a logic expression.
- In *synthesis* we do the reverse, starting with a formal description and proceeding to a logic diagram.

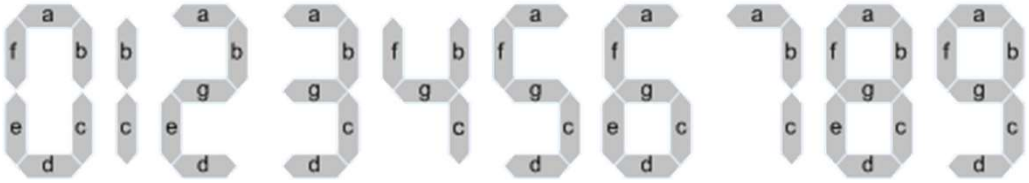
Combinatorial Circuit Synthesis Steps

- Identify inputs and outputs.
- Construct the truth table.
- Get the minimal expressions for outputs.
- Draw the circuit.

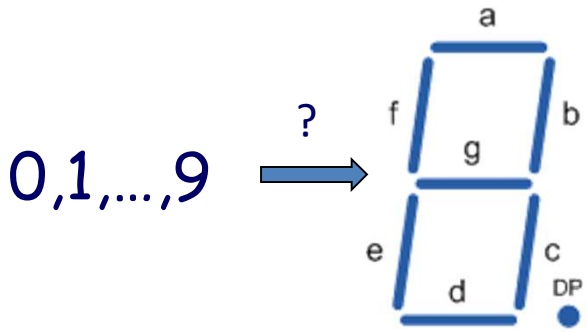
Example:

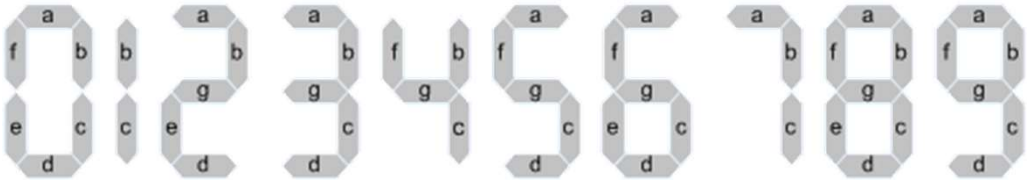
Design a 7-segment display controller (to show decimal digits).

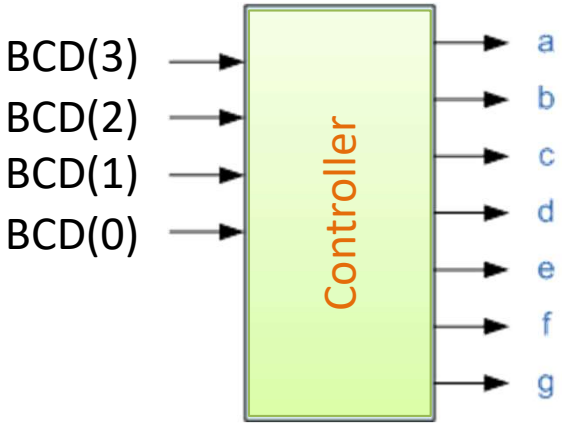




Design of a 7-Segment Display Controller



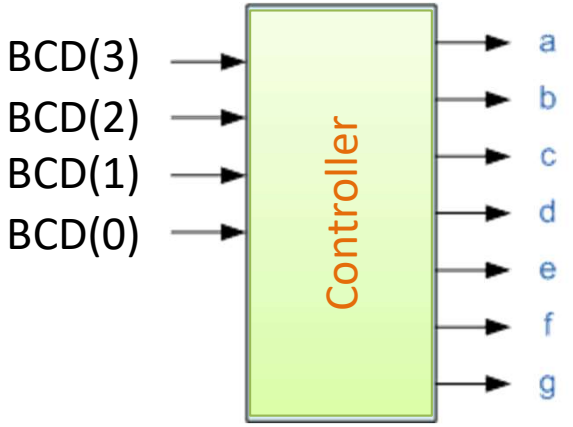




BCD	digit	Individual segments						
		a	b	c	d	e	f	g
0000	0	1	1	1	1	1	1	
0001	1		1	1				
0010	2	1	1		1	1		1
0011	3	1	1	1	1			1
0100	4		1	1			1	1
0101	5	1		1	1		1	1
0110	6	1		1	1	1	1	1
0111	7	1	1	1				
1000	8	1	1	1	1	1	1	1
1001	9	1	1	1	1		1	1
101x	x	x	x	x	x	x	x	x
11xx	x	x	x	x	x	x	x	x

Design of a 7-Segment Display Controller (cont.)

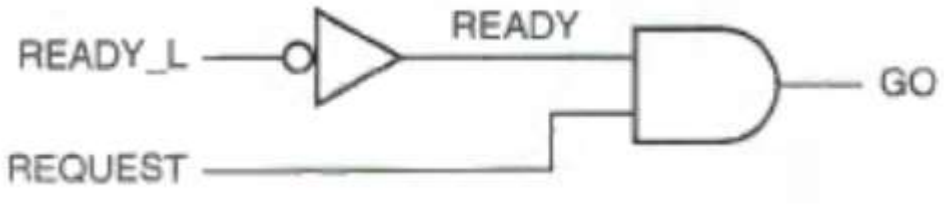
- Determine the minimal expressions for all the outputs *a*..*g*.
- Implement the obtained functions with logic gates (or more complex blocks – to be considered in the Part II).
  - a(BCD(3), BCD(2), BCD(1), BCD(0))
  - ..
  - g(BCD(3), BCD(2), BCD(1), BCD(0))



Signal Names and Active Levels

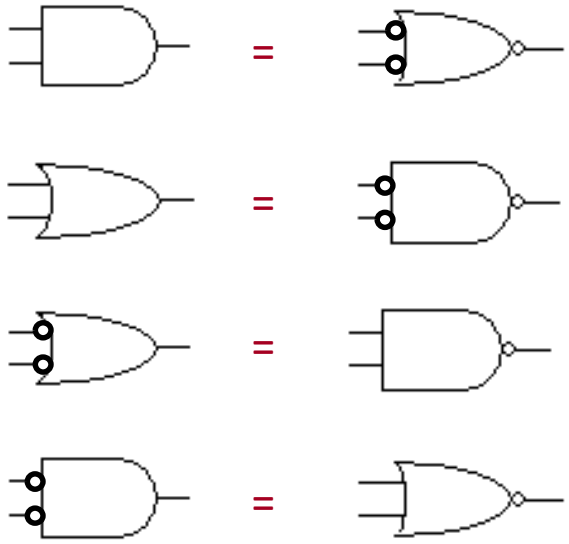
- Each signal name should have an **active level** associated with it.
- A signal is **active high** if it performs the named action or denotes the named condition when it is HIGH or 1.
- A signal is **active low** if it performs the named action or denotes the named condition when it is LOW or 0.
- The active level of each signal in a circuit is normally specified as part of its name, according to some convention.
- We'll use the following convention:
  - An active low signal name has a suffix of **_L,** and an active-high signal has no suffix.

Example:



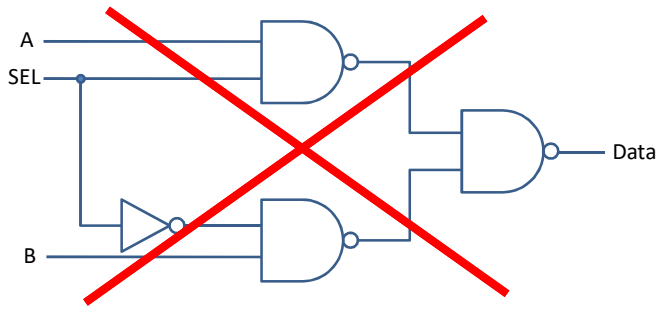
Active Levels for Pins

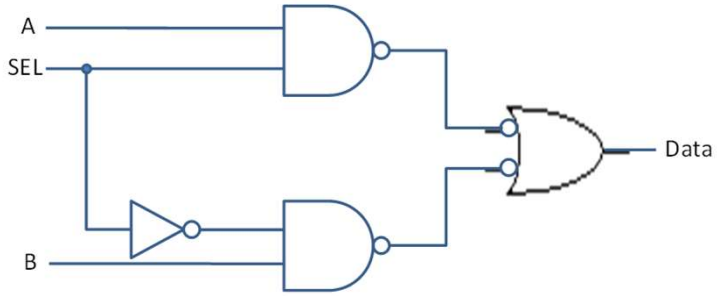
- An **inversion bubble** indicates an **active-low pin** and the absence of a bubble indicates an **active-high pin**.
- The inversion bubbles indicate that 0s must be applied to the input pins to activate the logic functions, and that the outputs are 0 when they are "doing their thing.“
- According to the DeMorgan’s theorems:



Bubble-to-Bubble Design

- **Bubble-to-bubble logic design** is the practice of choosing logic symbols and signal names, including active-level designators, that make the function of a logic circuit easier to understand.



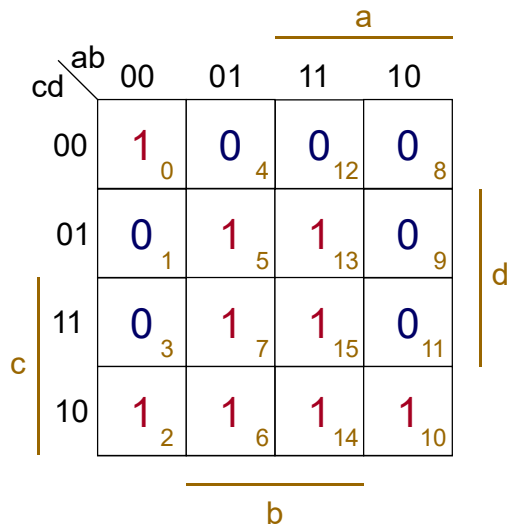


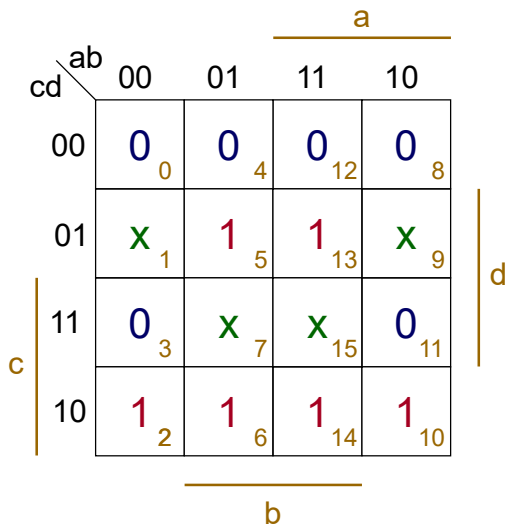
Exercises

- Using Karnaugh maps, find a minimal sum-of-products and a minimal product-of-sum expressions for each of the following four logic functions.

$$f(x,y,z) = x \cdot y + \bar{x} \cdot \bar{z} + y \cdot z$$

$$f(x,y,z) = y + \bar{x} \cdot \bar{z} = (\bar{x} + y) \cdot (y + \bar{z})$$





Exercises (cont.)

- If the canonical sum for an *n*-input logic function is also a minimal sum, how many literals are in each product term of the sum? Might there be any other minimal sums in this case?

- Find examples of 4-variable functions that meet the following criteria:
  - A function having two minimal sums-of-products;
  - A function whose minimal sum-of-products has **the same** number of terms and literals as the minimal product-of-sums;
  - A function whose minimal sum-of-products has **more** terms **and** literals than the minimal product-of-sums;
  - A function whose minimal sum-of-products has **less** terms **and** literals than the minimal product-of-sums;

- Repeat for 3 variables.

Exercises (cont.)

- Fill in the Karnaugh map for the following function and determine the minimal sum-of-products.

$$f(x,y,z) = (x \oplus y) \cdot z + \overline{x} \cdot (y \oplus z)$$

- How many prime implicants and essential prime implicants does the following function have?

$$f(a,b,c,d) = \sum m(2,6,7,8,9,10,13,15)$$

- Is it possible for a 3-variable function (different from constant 0) to have 0 essential prime implicants?