Digital Circuits I

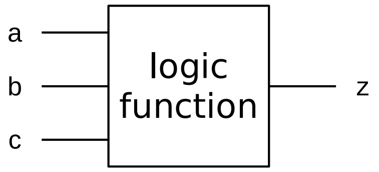
DAPA
E.T.S.I. Informática
Universidad de Sevilla
10/2012

Jorge Juan <jjchico@dte.us.es> 2010, 2011, 2012

Contents

- Logic functions
  - Logic operators
  - Logic expressions
  - Canonical forms
  - Design approach
- Boolean algebra
- Expression minimization
- Don't cares
- Functional analysis
- Timing analysis

Logic functions



Formal specification (truth table)

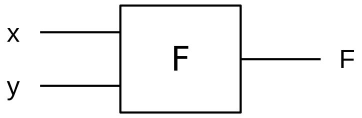
a b c	z
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

Verbal specification

Output a "1" if two or more inputs are equal to "1". Output "0" otherwise.

- Truth table: formal specification. Output value for every input combination (n inputs: 2^n input values).
- Combinational (digital) circuits implement logic functions.

Two-input logic functions



x y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		AND					XOR	OR	NOR	XNOR					NAND	

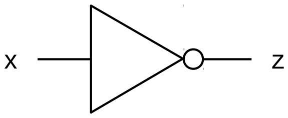
In general, there are $2^{(2^\wedge n)}$ functions of n variables

Logic operators and corresponding gates

NOT

x	z
0	1
1	0

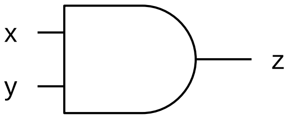
$$z = \overline{x}$$



AND

x y	z
0 0	0
0 1	0
1 0	0
1 1	1

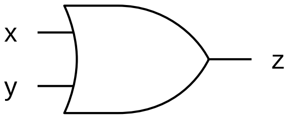
$$z = x \cdot y$$



OR

x y	z
0 0	0
0 1	1
1 0	1
1 1	1

$$z = x + y$$

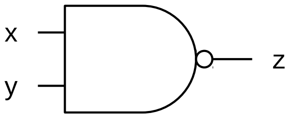


Logic operators and corresponding gates

NAND

x y	z
0 0	1
0 1	1
1 0	1
1 1	0

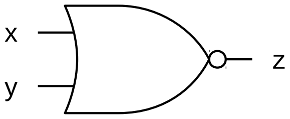
$$z = \overline{x \cdot y}$$



NOR

x y	z
0 0	1
0 1	0
1 0	0
1 1	0

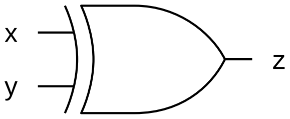
$$z = \overline{x + y}$$



XOR

x y	z
0 0	0
0 1	1
1 0	1
1 1	0

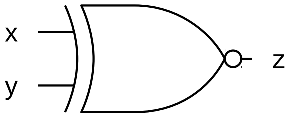
$$z = x \oplus y = \overline{x}y + x\overline{y}$$



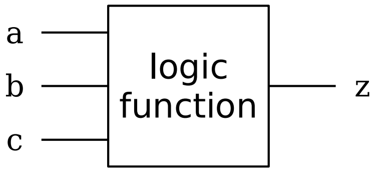
XNOR

x y	z
0 0	1
0 1	0
1 0	0
1 1	1

$$z = \overline{x \oplus y} = \overline{x}\,\overline{y} + xy$$



Logic expressions



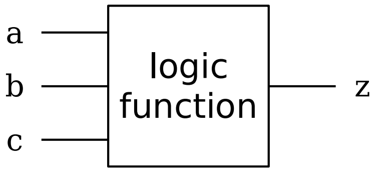
An expression for a function can always be obtained by combining the NOT, AND and OR operators.
Method:
1. For every "1" of the function, build a product term that is "1" for that input combination only.
2. Sum (OR) all the terms.

The resulting expression is a sum-of-products (S-O-P)
Terms containing all the literals are "**minterms**".
The sum-of-minterms is known as "**canonical form of minterms**"

a b c	z
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

$$z = \bar{a}\,b\,c + a\,\bar{b}\,c + a\,b\,\bar{c} + a\,b\,c$$

Logic expressions



An expression for a function can always be obtained by combining the NOT, AND and OR operators.
Method:
1. For every "0" of the function, build a sum term that is "0" for that input combination only.
2. Multiply (AND) all the terms.

The resulting expression is a product-of-sums (P-O-S)
Terms containing all the literals are "**maxterms**".
The sum-of-maxterms is known as "**canonical form of maxterms**"

a b c	z
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

$$z = (a+b+c)\cdot(a+b+\bar{c})\cdot(a+\bar{b}+c)\cdot(\bar{a}+b+c)$$

Canonical form notation

a b c	minterms	m-notat.
0 0 0	$\overline{a}\,\overline{b}\,\overline{c}$	m0
0 0 1	$\overline{a}\,\overline{b}\,c$	m1
0 1 0	$\overline{a}\,b\,\overline{c}$	m2
0 1 1	$\overline{a}\,b\,c$	m3
1 0 0	$a\,\overline{b}\,\overline{c}$	m4
1 0 1	$a\,\overline{b}\,c$	m5
1 1 0	$a\,b\,\overline{c}$	m6
1 1 1	$a\,b\,c$	m7

$$z = \overline{a}bc + a\overline{b}c + ab\overline{c} + abc$$
$$z = m3 + m5 + m6 + m7$$
$$z = \Sigma(3,5,6,7)$$

minterm: input combination for which the function is "1"

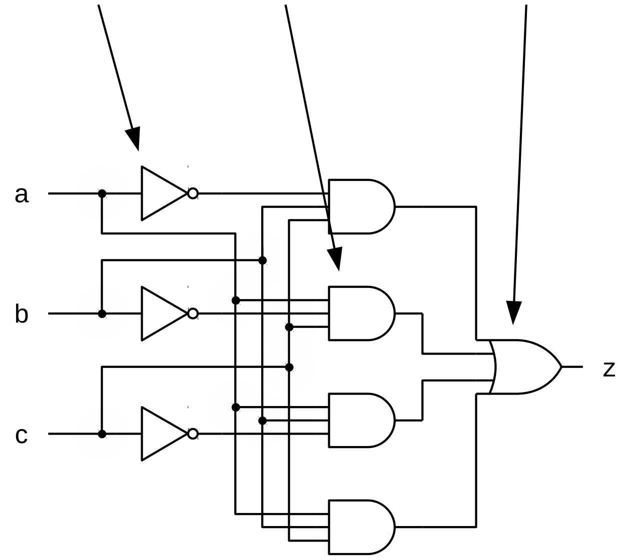
a b c	maxterm	M-notation
0 0 0	$a+b+c$	M0
0 0 1	$a+b+\overline{c}$	M1
0 1 0	$a+\overline{b}+c$	M2
0 1 1	$a+\overline{b}+\overline{c}$	M3
1 0 0	$\overline{a}+b+c$	M4
1 0 1	$\overline{a}+b+\overline{c}$	M5
1 1 0	$\overline{a}+\overline{b}+c$	M6
1 1 1	$\overline{a}+\overline{b}+\overline{c}$	M7

$$z = (a+b+c)(a+b+\overline{c})(a+\overline{b}+c)(\overline{a}+b+c)$$
$$z = M0\ M1\ M2\ M4$$
$$z = \Pi(0,1,2,4)$$

maxterm: input combination for which the function is "0"

Logic design approach

$$z = \overline{a}\,b\,c \;+\; a\,\overline{b}\,c \;+\; a\,b\,\overline{c} \;+\; a\,b\,c$$



S-O-P can be directly mapped to a digital circuits using three basic gates:

- Inverters (complements)
- AND (products)
- OR (sum)

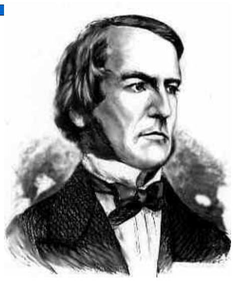
¿Is the result minimum?

Contents

- Logic functions
- **Boolean algebra**
- **Expression minimization**
- Don't cares
- Functional analysis
- Timing analysis

Boolean algebra

- {NOT, AND, OR} operators are a "Boolean algebra"



George Boole (1815-1864)

Elementary	$x+0 = x$ $x+1 = 1$ $x+x = x$ $x+\overline{x} = 1$ $\overline{(\overline{x})} = x$	$x \cdot 1 = x$ $x \cdot 0 = 0$ $x \cdot x = x$ $x \cdot \overline{x} = 0$
Commutativity	$x+y = y+x$	$x \cdot y = y \cdot x$
Associativity	$(x+y)+z = x+(y+z)$	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$
Distributivity	$x \cdot (y+z) = (x \cdot y)+(x \cdot z)$	$x+(y \cdot z) = (x+y) \cdot (x+z)$
Reduction	$\mathbf{(x \cdot y)+(x \cdot \overline{y}) = x}$	$\mathbf{(x+y) \cdot (x+\overline{y}) = x}$
De Morgan's	$\overline{(x+y+\dots)} = \overline{x} \cdot \overline{y} \cdot \dots$	$\overline{(x \cdot y \cdot \dots)} = \overline{x}+\overline{y}+\dots$

Duality: if an expression holds, the expression that results from interchanging + with · and 0 with 1 also holds.

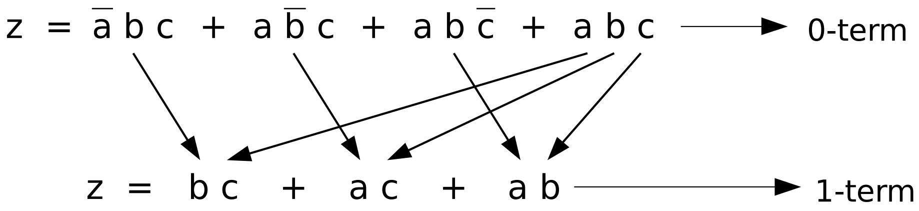
Boolean algebra

- Precedence of · over +
  - x+(y·z) = x+y·z
- "·" can be omitted
  - x+y·z = x+yz

Minimization

$$\boxed{xy + x\bar{y} = x}$$

"x" can be any expression



$$z = \bar{a}\,b\,c + a\,\bar{b}\,c + a\,b\,\bar{c} + a\,b\,c \longrightarrow$$ 0-term

$$z = b\,c + a\,c + a\,b \longrightarrow$$ 1-term

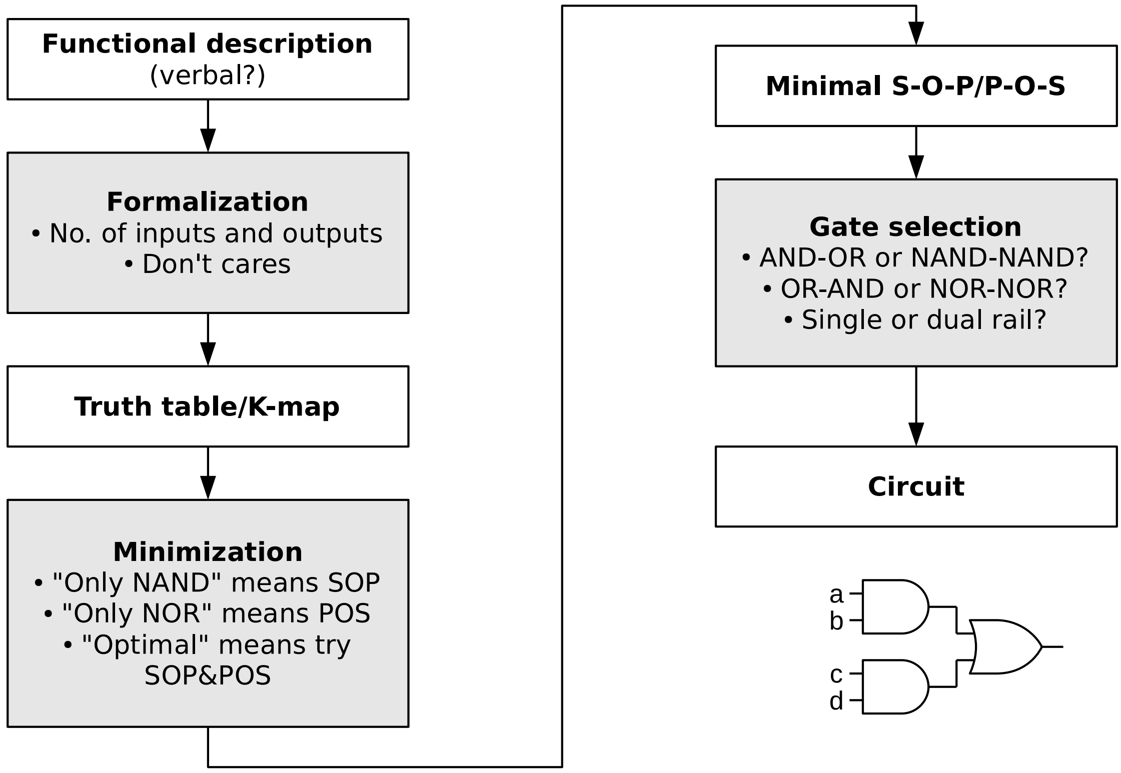
One term can be used more than once (we cannot simplify one term at a time).

Implicant of a function
- Product term that can be part of a S-O-P expression of a function
- Product term that "covers" some minterms of a function

Minimization summary

- Karnaugh maps (manual)
  - Tedious and error-prone for 5 or 6-input functions. Unfeasible for >6.
- Quine-McCluskey
  - Not feasible for a large number of inputs.
  - Computational time increases exponentially with n. of inputs (32 inputs → ~10^{15} prime implicants)
- Alternative: Heuristic logic minimizers
- Implemented in logic synthesis tools

Manual design process (summary)



Functional description (verbal?)

Formalization
- No. of inputs and outputs
- Don't cares

Truth table/K-map

Minimization
- "Only NAND" means SOP
- "Only NOR" means POS
- "Optimal" means try SOP&POS

Minimal S-O-P/P-O-S

Gate selection
- AND-OR or NAND-NAND?
- OR-AND or NOR-NOR?
- Single or dual rail?

Circuit

Design example

Design a combinational circuit with four inputs (x3, x2, x1, x0) that represent the bits of a BCD digit X, and two outputs (c1, c0) that represents the bits of a magnitude C, where C is the quotient of the division X/3.

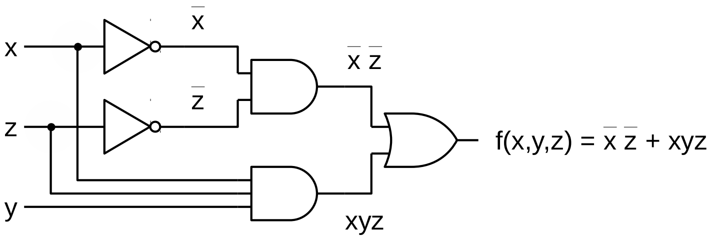
E.g. if X=7 → C=2, that is, (x3,x2,x1,x0)=(0,1,1,1) → (c1,c0)=(1,0)

Design the circuit using a minimum two-level structure of only NAND gates.

Contents

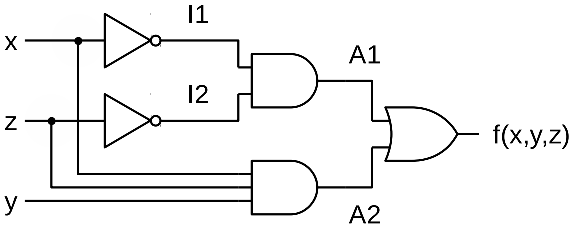
- Logic functions
- Boolean algebra
- Expression minimization
- Don't cares
- **Functional analysis**
- **Timing analysis**

Functional analysis



- Method. Starting at primary inputs:
  - For each gate with known inputs, calculate the equation at the gate's output.
  - Repeat until the equations of all circuit outputs are known.
  - Convert to a more useful representation: truth table, K-map, …
  - Give a verbal description of the operation (if possible).

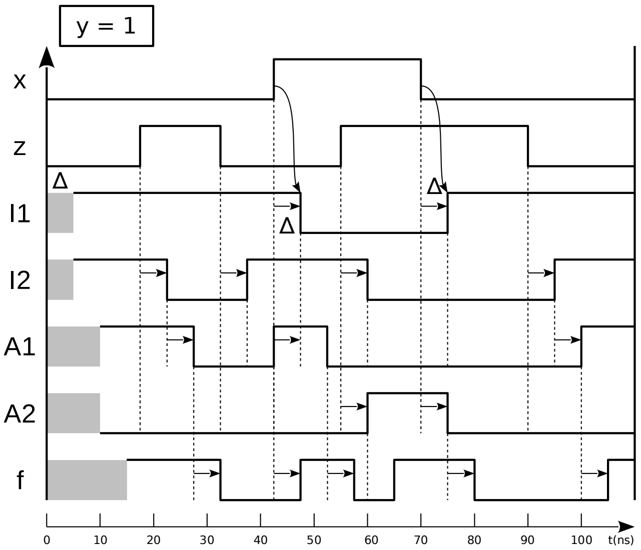
Timing analysis



$I1 = \bar{x}$
$I2 = \bar{z}$
$A1 = I1\ I2$
$A2 = x\ y\ z$
$f = A1 + A2$

y = 1

$I1 = \bar{x}$
$I2 = \bar{z}$
$A1 = I1\ I2$
$A2 = x\ z$
$f = A1 + A2$



Method:

- For every gate: equations of the output as a function of the inputs.
- Substitute constant input values (DO NOT SUBSTITUTE ANYTHING ELSE)
- Draw node curves from primary inputs to the outputs. Delay every output transition by the gate's delay time (simple model: same delay for every gate: Δ)