Logic Design

Chapter 4: Optimized Implementation of Logic Functions

McMaster University

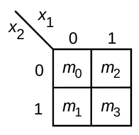
Introduction

- The combining property allows us to replace two minterms that differ in only one variable with a single product term that does not include that variable.
- Combining property can be used to reduce the number of product terms in SOP
- Karnaugh map provides a systematic way of performing this optimization

$$F(A,B,C) = A'B'C' + AB'C'$$
$$F = (A + A')B'C'$$
$$F = B'C'$$

x_1	x_2	
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

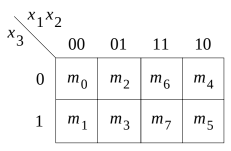
(a) Truth table



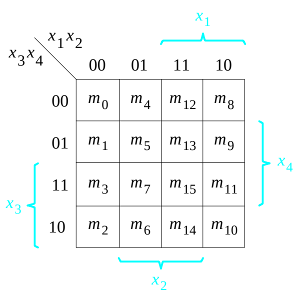
(b) Karnaugh map

x_1	x_2	x_3	
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

(a) Truth table



(b) Karnaugh map



K-map

- How to use the K-map to find a minimum cost implementation?
- Find as few as possible and as large as possible group of adjacent 1s that cover all cases where the function has a value of 1
- Large group of 1s: fewer number of variables in the corresponding product term -> gates with fewer number of inputs
- Fewer number of groups: less AND gates
- Keep in mind that cost is the number of gates and number of inputs to the gate
- Size of groups: 1, 2, 4, 8, ….

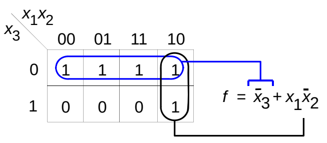
K-map

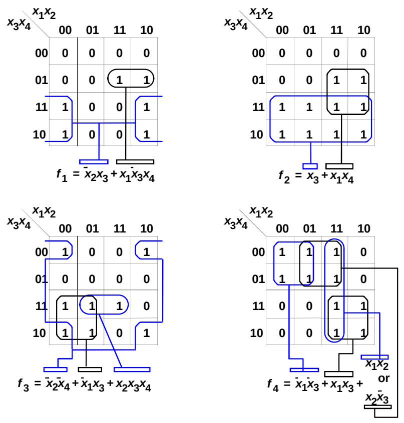
- Adjacent cells: cells that differ only in one variable
- Cells in left and right edge of the K-map are adjacent
- Cells in top and bottom edge of the K-map are adjacent
- You can visualize the map as being folded around a cylinder
- Four corners of the map are adjacent to each other and can form a group

K-map

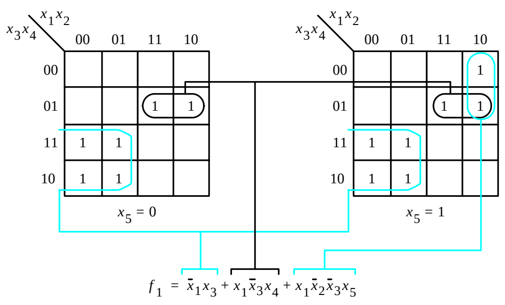
- How to find out the product term corresponding to a group?
- The product term corresponding to a group must include only those variables that have the same value (0 or 1) for all the cells in the group
- Note: if it helps in forming bigger groups a cell can be used for more than once
- This is fine since $m_i = m_i + m_i$

Row number	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0




$f_1 = \bar{x}_2x_3 + x_1\bar{x}_3x_4$
$f_2 = x_3 + x_1x_4$
$f_3 = \bar{x}_2\bar{x}_4 + \bar{x}_1x_3 + x_2x_3x_4$
$f_4 = \bar{x}_1\bar{x}_3 + x_1x_3 +$ x_1x_2 or $x_2\bar{x}_3$

Five Variable Map


$x_5 = 0$
$x_5 = 1$
$f_1 = \bar{x}_1x_3 + x_1\bar{x}_3x_4 + x_1\bar{x}_2\bar{x}_3x_5$

Minimization Procedure

- Selection of groups on the K-map may be made more systematic using implicants.
- Literal: Each appearance of a variable, either uncomplemented or complemented, in a product term, is called a literal
- Example: $x_1x_2'x_3$ has three literals.
- A product term for which a given function is equal to 1 is called an *implicant*.
- Example: do the example on board

Minimization Procedure

- If the removal of any literal from an implicant P results in a product term that is not an implicant of the function, then P is a *prime implicant*.
- A collection (set) of implicants that account for all variations for which a given function is equal to 1 is called a *cover* of that function.
- *Cost* of the function: total number of gates plus total number of all inputs to all gates
- The cover consisting of prime implicants leads to the lowest-cost implementation.

Minimization Procedure

- How to determine the minimum-cost subset of prime implicants that will cover the function?
- If a prime implicant includes a minterm for which f=1 that is not included in any other prime implicant, then it is called an essential prime implicant and must be included in the cover
- Do the example on the board

Minimization Procedure

- Process of finding a minimum-cost circuit:

1. Generate all prime implicants for the give function f
2. Find the set of essential prime implicants
3. If the set of essential prime implicants covers all valuations for which f=1, then this set is the desired cover of f. Otherwise, determine the nonessential prime implicants that should be added to form a complete minimum-cost cover

Minimization Procedure

- The choice of nonessential prime implicants to be included in the cover is governed by the cost considerations
- One approach: arbitrarily select one nonessential prime implicant and include it in the cover, and determine the rest of the cover. Determine another cover, assuming that this prime implicant is not in the cover. Choose the less expensive cover.
- Do an example on board

Minimization of POS Forms

- Find the minimum cost SOP implementation of the complement of f (look for zeros in the K-map)
- Apply DeMorgan’s theorem to obtain the simplest POS realization
- Do the example on board

Incompletely specified functions

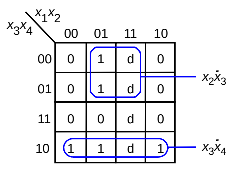
- Functions with undefined outputs for some input combinations are called "incompletely specified functions".
- These "don’t care" conditions may be used to advantage to provide further simplification of the function
- The designer can assume that the function value of don’t care conditions is 0 or 1 whichever is more useful.
- To indicate don’t-care conditions on the K-map, they are normally marked as “d”

Incompletely specified functions

$$f(x_1,\ldots,x_4)=\sum m(2,4,5,6,10)+D(12,13,14,15)$$

$$f=\bar{x}_1x_2\bar{x}_3+\bar{x}_1x_3\bar{x}_4+\bar{x}_2x_3\bar{x}_4$$

$$f=x_2\bar{x}_3+x_3\bar{x}_4$$



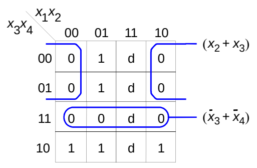
(a) SOP implementation

Incompletely specified functions

$$f(x_1,\ldots,x_4)=\sum m(2,4,5,6,10)+D(12,13,14,15)$$

$$f=(x_2+x_3)(\bar{x}_3+\bar{x}_4)(\bar{x}_1+\bar{x}_2)$$

$$f=(x_2+x_3)(\bar{x}_3+\bar{x}_4)$$

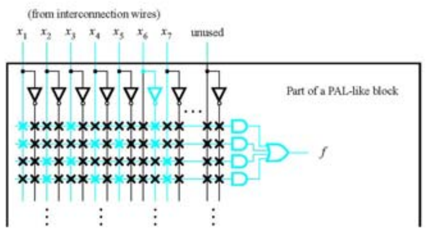


(b) POS implementation

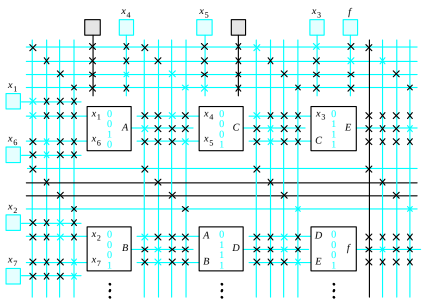
Multilevel Synthesis

- Minimum cost sum-of-products or product-of-sums realizations have two levels (stages) of gates
- Sum-of-product: first level AND gates connected to a second level OR gate
- Product-of-sum: first level OR gates connected to a second level AND gate
- As the number of inputs increases a two-level circuit may result in fan-in problem
- Whether or not this is an issue depends on the type of technology used to implement the circuit

$$f(x_1,x_2,\ldots,x_7)=x_1x_3\bar{x}_6+x_1x_4x_5\bar{x}_6+x_2x_3x_7+x_2x_4x_5x_7$$



(from interconnection wires)
x_1 x_2 x_3 x_4 x_5 x_6 x_7
unused
Part of a PAL-like block
f



Multi-level synthesis

- To solve the fan-in problem f should be expressed in a form that has more than two levels
- Multi-level logic expression
- Two important techniques for synthesis of multi-level circuits:

1. Factoring
2. Functional decomposition

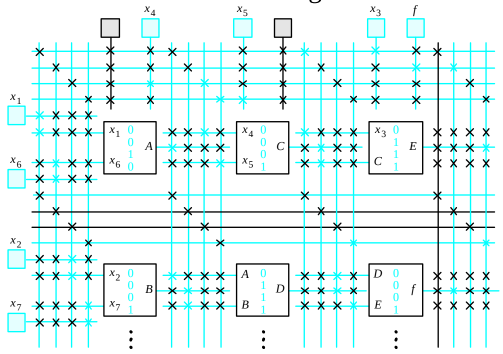
Factoring

$$f(x_1, x_2, \ldots, x_7) = x_1 x_3 \overline{x_6} + x_1 x_4 x_5 \overline{x_6} + x_2 x_3 x_7 + x_2 x_4 x_5 x_7$$

$$f = x_1 \overline{x_6}(x_3 + x_4 x_5) + x_2 x_7 (x_3 + x_4 x_5)$$

$$f = (x_1 \overline{x_6} + x_2 x_7)(x_3 + x_4 x_5)$$

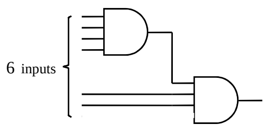
Factoring



Factoring

$$f = x_1 \overline{x_2} x_3 \overline{x_4} x_5 x_6 + x_1 x_2 \overline{x_3}\,\overline{x_4}\,\overline{x_5} x_6$$

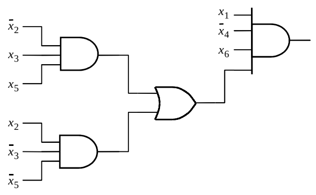
$$f = x_1 \overline{x_4} x_6 (\overline{x_2} x_3 x_5 + x_2 \overline{x_3}\,\overline{x_5})$$



Factoring

$$f = x_1 \overline{x_2} x_3 \overline{x_4} x_5 x_6 + x_1 x_2 \overline{x_3}\,\overline{x_4}\,\overline{x_5} x_6$$

$$f = x_1 \overline{x_4} x_6 (\overline{x_2} x_3 x_5 + x_2 \overline{x_3}\,\overline{x_5})$$



Functional decomposition

- Complexity of a logic circuit can often be reduced by decomposing a two-level circuit into sub-circuits where one or more sub-circuit implements functions that may be used in several places to construct the final circuit

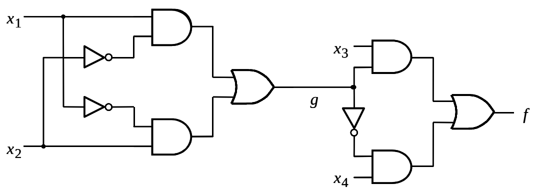
$$f = \overline{x_1} x_2 x_3 + x_1 \overline{x_2} x_3 + x_1 x_2 x_4 + \overline{x_1}\,\overline{x_2} x_4$$

$$f = (\overline{x_1} x_2 + x_1 \overline{x_2}) x_3 + (x_1 x_2 + \overline{x_1}\,\overline{x_2}) x_4$$

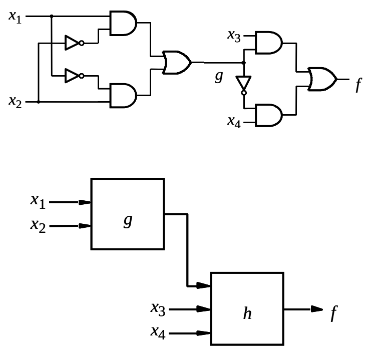
$$g = \overline{x_1} x_2 + x_1 \overline{x_2}$$

$$\overline{g} = x_1 x_2 + \overline{x_1}\,\overline{x_2}$$

$$f = g x_3 + \overline{g} x_4$$

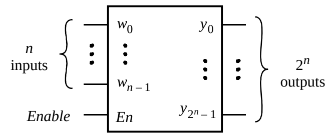


Functional decomposition



Decoders

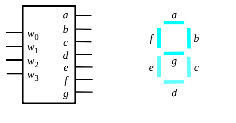
- Decoder: decodes encoded information
- A binary decoder is a logic circuit with n inputs and 2^n outputs
- Only one output is asserted at a time (corresponding to one valuation of inputs)
- Enable: En=0 none of the decoder outputs is asserted



Binary coded decimal (BCD)

- Each digit in a decimal number is represented by its binary form
- Since there are 10 digits we need 4 bits per digit

Decimal digit	BCD code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001



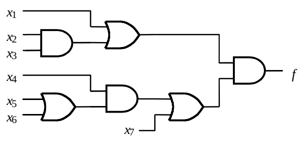
(a) Code converter (b) 7-segment display

w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

(c) Truth table

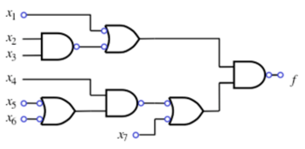
Figure 6.25. A BCD-to-7-segment display code converter.

Multi-level NAND and NOR Circuits

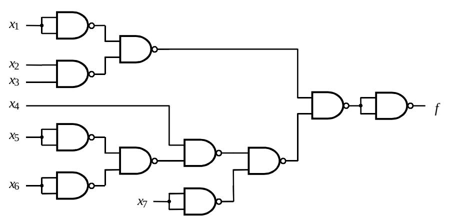


(a) Circuit with AND and OR gates

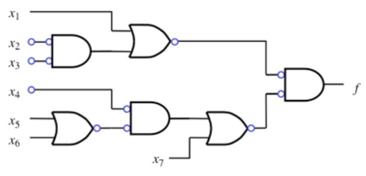
Multi-level NAND and NOR Circuits



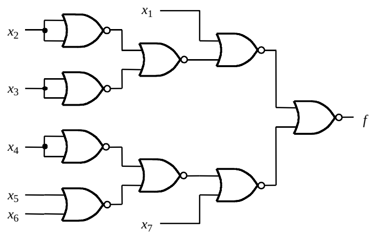
Multi-level NAND and NOR Circuits



(c) NAND-gate circuit



(a) Inversions needed to convert to NORs

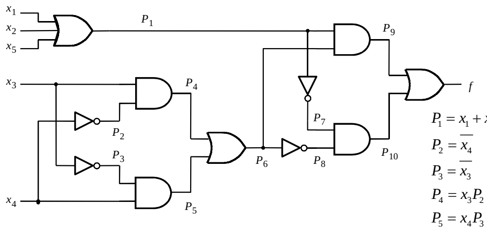


(b) NOR-gate circuit

Analysis of Multi-level Circuits

- In order to derive the function of a multi-level circuit, we have to trace the circuit either by tracking the inputs and working towards the outputs or the other way.
- Finding the function of intermediate points is helpful.

Analysis of Multi-level Circuits



$P_1 = x_1 + x_2 + x_3,$

$P_2 = \overline{x_4}$

$P_3 = \overline{x_3}$

$P_4 = x_3 P_2$

$P_5 = x_4 P_3$

$P_6 = P_4 + P_5$

$P_7 = \overline{P_1}$

$P_8 = \overline{P_6}$

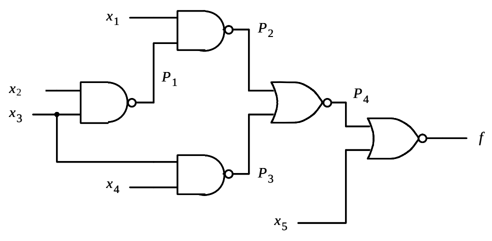
$P_9 = P_1 P_6$

$P_{10} = P_7 P_8$

Analysis of Multi-level Circuits

$$f = P_9 + P_{10} = P_1P_6 + P_7P_8$$
$$= (x_1 + x_2 + x_5)(P_4 + P_5) + \overline{P_1}\,\overline{P_6}$$
$$= (x_1 + x_2 + x_5)(x_3P_2 + x_4P_3) + \overline{x_1}\,\overline{x_2}\,\overline{x_5}\,\overline{P_4}\,\overline{P_5}$$
$$= (x_1 + x_2 + x_5)(x_3\overline{x_4} + \overline{x_3}x_4) + \overline{x_1}\,\overline{x_2}\,\overline{x_5}(\overline{x_3} + x_4)(\overline{x_4} + x_3)$$
$$= x_1x_3\overline{x_4} + x_1\overline{x_3}x_4 + x_2x_3\overline{x_4} + x_2\overline{x_3}x_4 + x_5x_3\overline{x_4} + x_5\overline{x_3}x_4 +$$
$$\overline{x_1}\,\overline{x_2}\,\overline{x_5}\,\overline{x_3}\,\overline{x_4} + \overline{x_1}\,\overline{x_2}\,\overline{x_5}x_4x_3$$

Analysis of Multi-level Circuits



$$P_1 = \overline{x_2 x_3}$$
$$P_2 = \overline{x_1 P_1} = \overline{x_1} + \overline{P_1}$$
$$P_3 = \overline{x_3 x_4} = \overline{x_3} + \overline{x_4}$$
$$P_4 = \overline{P_2 + P_3}$$
$$f = \overline{P_4 + x_5} = \overline{P_4}\,\overline{x_5} = \overline{\overline{P_2 + P_3}}\,\overline{x_5}$$
$$= (\overline{x_1} + \overline{P_1} + \overline{x_3} + \overline{x_4})\overline{x_5}$$
$$= (\overline{x_1} + x_2 x_3 + \overline{x_3} + \overline{x_4})\overline{x_5}$$
$$= \overline{x_1}\,\overline{x_5} + x_2\overline{x_5} + \overline{x_3}\,\overline{x_5} + \overline{x_4}\,\overline{x_5}$$