Chapter 3
Implementation Technology

- To the students: Seeing that you have not yet covered transistors in your courses, the first part of this chapter is only briefly scanned. You can read though it, but the content will not be examined.

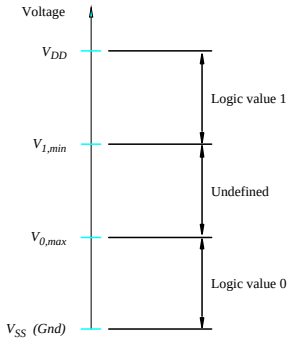
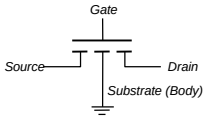
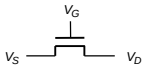


 Figure 3.1 Logic values as voltage levels

(a) A simple switch controlled by the input x



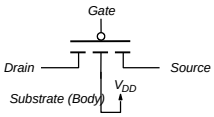
(b) NMOS transistor



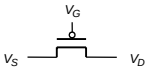
(c) Simplified symbol for an NMOS transistor

 Figure 3.2 NMOS transistor as a switch

(a) A switch with the opposite behavior of Figure 3.2*a*

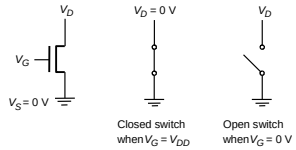


(b) PMOS transistor

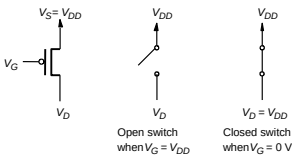


(c) Simplified symbol for an PMOS transistor

 Figure 3.3 PMOS transistor as a switch

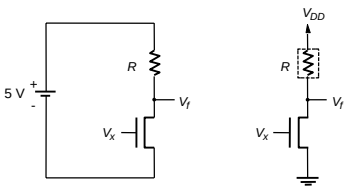


(a) NMOS transistor

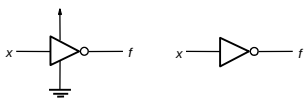


(b) PMOS transistor

 Figure 3.4 NMOS and PMOS transistors in logic circuits

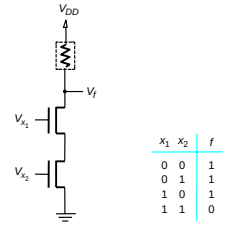


(a) Circuit diagram (b) Simplified circuit diagram



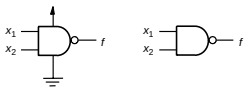
(c) Graphical symbols

 Figure 3.5 A NOT gate built using NMOS technology



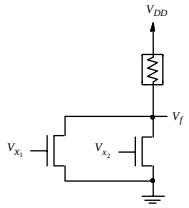
x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

(a) Circuit (b) Truth table



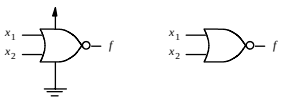
(c) Graphical symbols

Figure 3.6 NMOS realization of a NAND gate



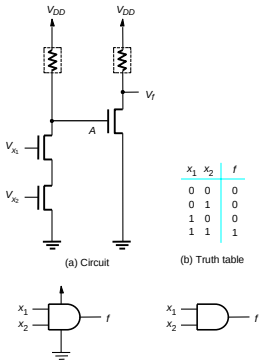
x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	0

(a) Circuit (b) Truth table



(c) Graphical symbols

Figure 3.7 NMOS realization of a NOR gate

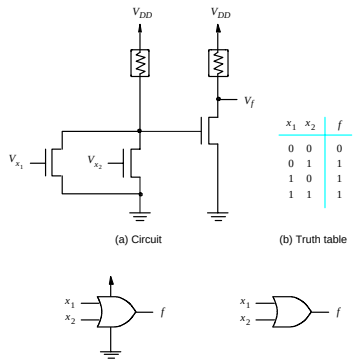


x_1	x_2	f
0	0	0
0	1	0
1	0	0
1	1	1

(a) Circuit (b) Truth table

(c) Graphical symbols

Figure 3.8 NMOS realization of an AND gate



x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

(a) Circuit (b) Truth table

(c) Graphical symbols

Figure 3.9 NMOS realization of an OR gate

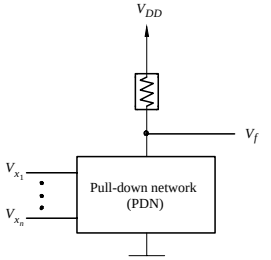


Figure 3.10 Structure of an NMOS circuit

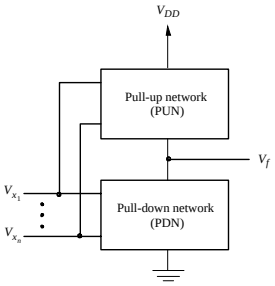
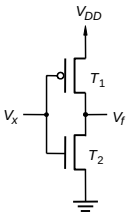


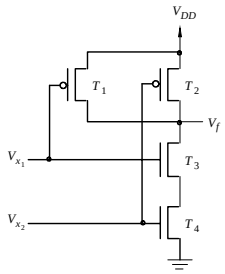
Figure 3.11 Structure of a CMOS circuit



x	T_1	T_2	f
0	on	off	1
1	off	on	0

(a) Circuit (b) Truth table and transistor states

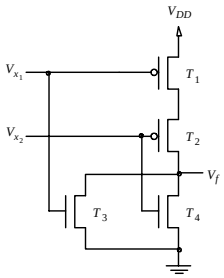
Figure 3.12 CMOS realization of a NOT gate



x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	1
1	0	off	on	on	off	1
1	1	off	off	on	on	0

(a) Circuit (b) Truth table and transistor states

Figure 3.13 CMOS realization of a NAND gate



x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	0
1	0	off	on	on	off	0
1	1	off	off	on	on	0

(a) Circuit (b) Truth table and transistor states

Figure 3.14 CMOS realization of a NOR gate

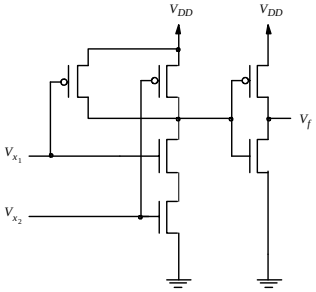


Figure 3.15 CMOS realization of an AND gate

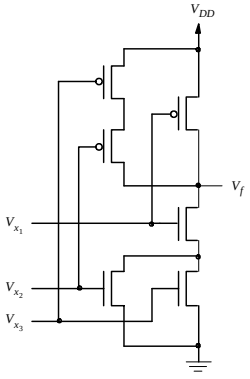


Figure 3.16 A CMOS complex gate

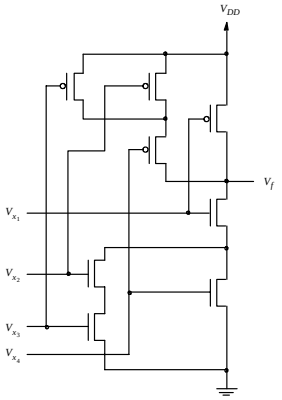
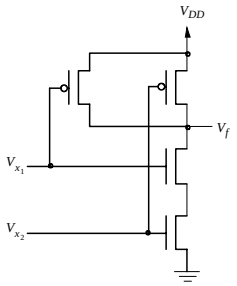


Figure 3.17 A CMOS complex gate



(a) Circuit

V_{x_1}	V_{x_2}	V_f
L	L	H
L	H	H
H	L	H
H	H	L

(b) Voltage levels

Figure 3.18 Voltage levels in a CMOS circuit

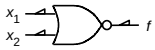
V_{x_1}	V_{x_2}	V_f
L	L	H
L	H	H
H	L	H
H	H	L

(a) Voltage levels

x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

(b) Positive logic truth table and gate symbol

x_1	x_2	f
1	1	0
1	0	0
0	1	0
0	0	1



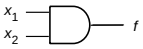
(c) Negative logic truth table and gate symbol

Figure 3.19 Interpretation of voltage levels

V_{x_1}	V_{x_2}	V_f
L	L	L
L	H	L
H	L	L
H	H	H

(a) Voltage levels

x_1	x_2	f
0	0	0
0	1	0
1	0	0
1	1	1

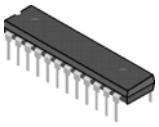


(b) Positive logic

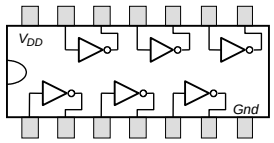
x_1	x_2	f
1	1	1
1	0	1
0	1	1
0	0	0

(c) Negative logic

Figure 3.20 Interpretation of voltage levels



(a) Dual-inline package



(b) Structure of 7404 chip

Figure 3.21 A 7400-series chip

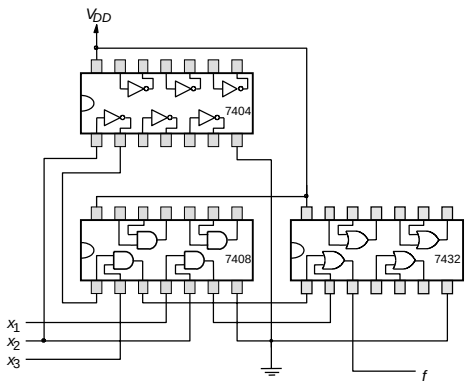


Figure 3.22 Implementation of $f = x_1\bar{x}_2 + x_2x_3$

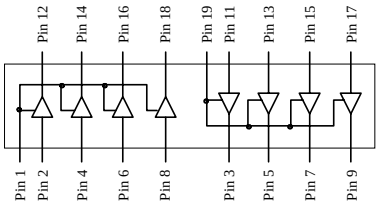


Figure 3.23 The 74244 buffer chip

Optimized Implementation of Logic Functions

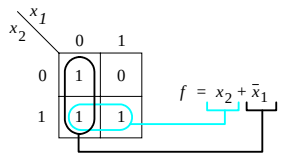
- 4.1 Karnaugh Map
  - Minimization of logic functions

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

(a) Truth table

x_2 \ x_1	0	1
0	m_0	m_2
1	m_1	m_3

(b) Karnaugh map

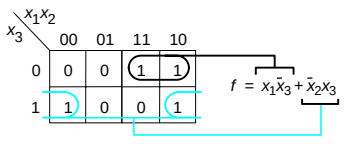


x_1	x_2	x_3	
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

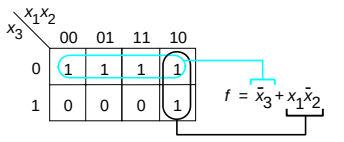
(a) Truth table

x_3 \ x_1x_2	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

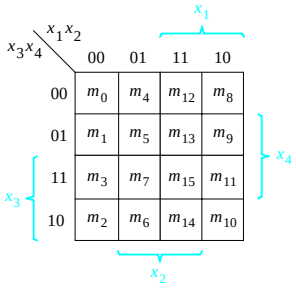
(b) Karnaugh map

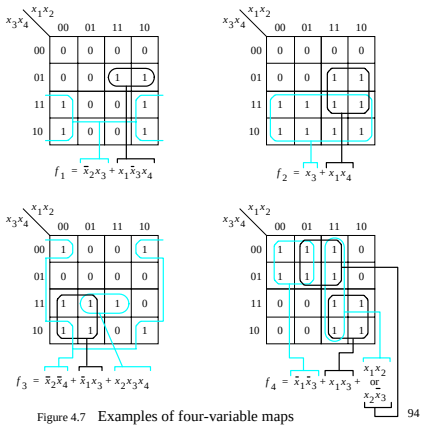


(a) The function of Figure 2.18

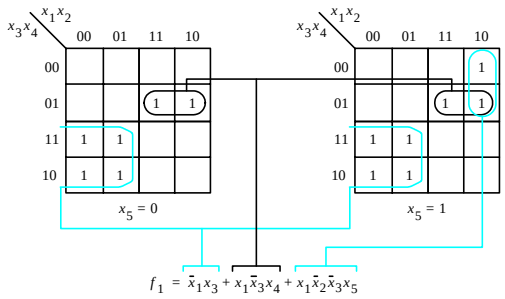


(b) The function of Figure 4.1





July 2000 Figure 4.7 Examples of four-variable maps 94

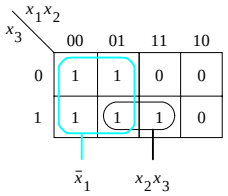


July 2000 Figure 4.8 A five-variable Karnaugh map 95

4.2 Strategy for Minimization

- 4.2.1 Terminology
  - Literal: Each appearance of a variable, either uncomplemented or complemented, is called a literal e.g. x1•x2' • x3 has three literals.
  - Implicant: A product term that indicates the input valuation(s) for which a given function is equal to 1 is called an implicant of the function. (see figure 4.9 with 11 implicants).

July 2000 Univ. of Stellenbosch - Digital Systems 144 96



July 2000 Figure 4.9 Three-variable function $f = \Sigma\, m(0, 1, 2, 3, 7)$ 97

Prime implicants (Afr: Priem Implikant)

An implicant is called a *prime implicant* if it cannot be combined into another implicant that has fewer literals or it is an implicant that cannot be made larger.

July 2000 Univ. of Stellenbosch - Digital Systems 144 98

Cover (Afr: Dekking)

A collection of implicants that account for all valuations for which a given function is equal to 1 is called a *cover* of that function.

In figure 4.9:

A cover consisting of minterms:

f=x1'•x2' • x3'+x1' • x2' • x3+x1' • x2 • x3'+x1' • x2 • x3+x1 • x2 • x3

Another valid cover is:

f = x1' • x2'+x1' • x2+x2 • x3

The cover comprising the prime implicants is:

f = x1' + x2 • x3

July 2000 Univ. of Stellenbosch - Digital Systems 144 99

Cost (Afr:koste)

- A good measure of the cost is the number of gates (+ the number of inputs).
- The cost of f = x1 •x2' + x3 •x4' is 9.

4.2.2 Minimization Procedure

- Essential prime implicant:
  - If a prime implicant includes a minterm for which f = 1 that is not included in any other prime implicant, then it must be included in the cover and is called an essential prime implicant.
- Minimization steps:
  - Generate all prime implicants
  - Find the set of essential prime implicants
  - Cover non-essential PI's with minimum cost

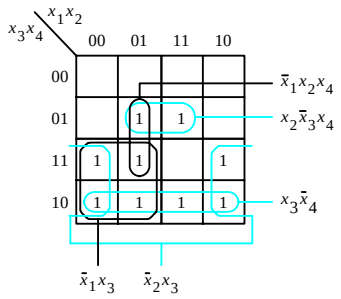


 Figure 4.10 Four-variable function $f = \Sigma\, m(2, 3, 5, 6, 7, 10, 11, 13, 14)$

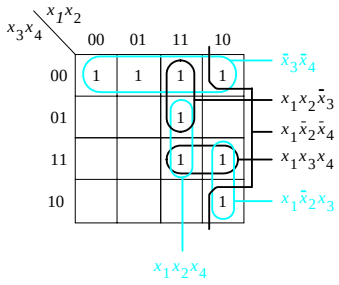


 Figure 4.11 The function $f = \Sigma\, m(0, 4, 8, 10, 11, 12, 13, 15)$

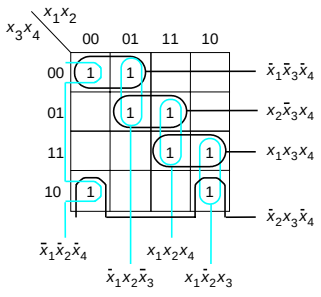
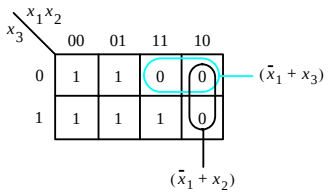


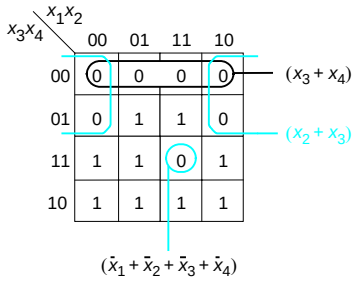
 Figure 4.12 The function $f = \Sigma\, m(0, 2, 4, 5, 10, 11, 13, 15)$

4.3 Minimization of Product-of-Sums Forms

- Group 0's in Karnaugh map in stead of 1's
- Duality principle
- Result gives NOT f.



July 2000 Figure 4.13 POS minimization of $f = \Pi M(4, 5, 6)$ 106



July 2000 Figure 4.14 POS minimization of $f = \Pi M(0, 1, 4, 8, 9, 12, 15)$ 107

4.4 Incompletely specified functions

- Don't-care conditions (d's in Karnaugh map)
- See example in figure 4.15

July 2000 Univ. of Stellenbosch - Digital Systems 144 108

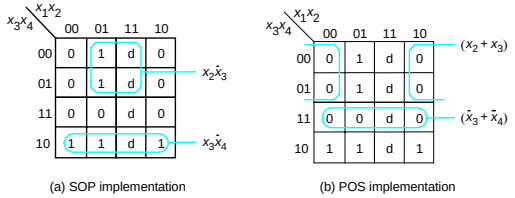
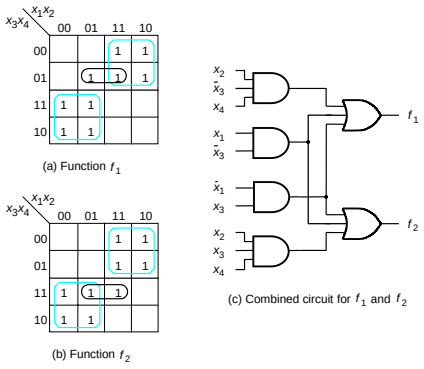


Figure 4.15 Two implementations of $f = \Sigma m(2, 4, 5, 6, 10) + D(12, 13, 14, 15)$ 109

4.5 Multiple-output Circuits

- Multiple bit outputs
- Minimize with Karnaugh map for each bit
- Combine input logic (product terms) for lower cost circuit
- See figure 4.16

July 2000 Univ. of Stellenbosch - Digital Systems 144 110



July 2000 Figure 4.16 An example of multiple-output synthesis 111

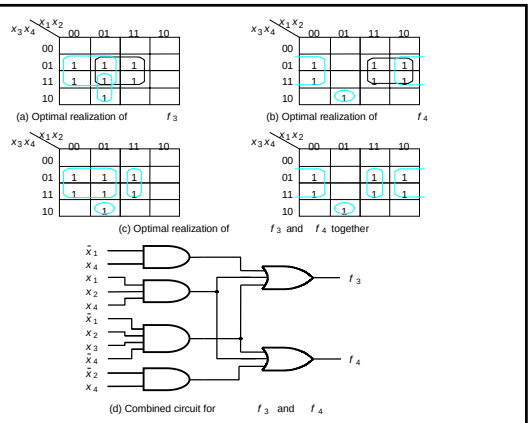


Figure 4.17 An example of multiple-output synthesis

4.6 NAND and NOR logic networks

- Use De-Morgan to modify any network to one containing only NAND or only NOR gates
- See figures 4.18, 19, 20

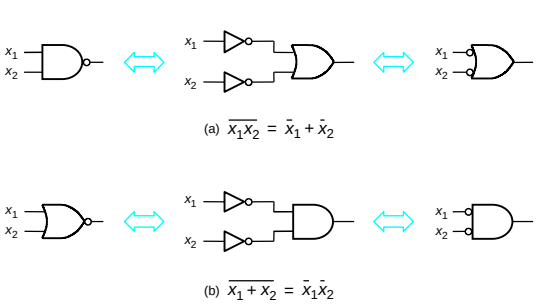


Figure 4.18 DeMorgan's theorem in terms of logic gates

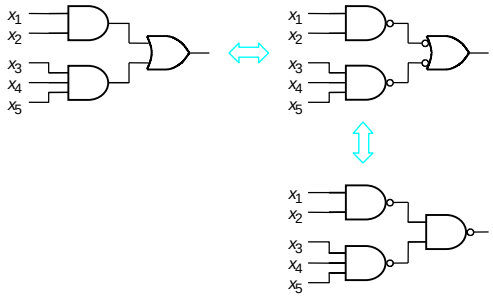


Figure 4.19 Using NAND gates to implement a sum-of-products

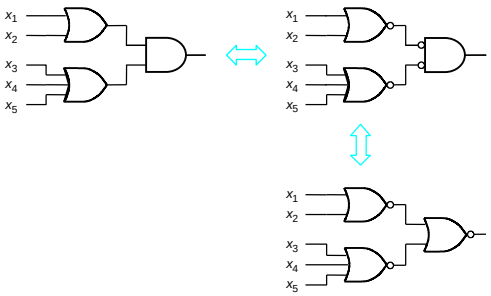


Figure 4.20 Using NOR gates to implement a product-of-sums

- Students: Skip rest of chapter 4 for now.