DIGITAL TECHNICS I

Dr. Bálint Pődör

Óbuda University, Microelectronics and Technology Institute

5. LECTURE: LOGIC MINIMIZATION

1st year BSc course 1st (Autumn) term 2018/2019

1

5. LECTURE: LOGIC MINIMIZATION

1. Revision and summary: canonic forms, minterms, maxterms, minimization, etc.

Repetitio est mater studiorum

2. Incompletely specified logic functions

3. Logic synthesis examples using Karnaugh map

4. Basics of tabular/numeric minimization (Quine-McCluskey algorithm) with demo

2

REVISION AND SUMMARY

- Combinational networks....
- Disjunctive and conjunctive canonic forms ...
- Minterms and maxterms ...
- Adjacency, minimization, prime implicants…
- Graphic minimization...
- Karnaugh map ...

Repetitio est mater studiorum

3

CANONIC ALGEBRAIC FORMS OF LOGIC FUNCTIONS

Because a logic function can have several equivalent algebraic forms, the basis of the synthesis is one of the canonical forms (extended SOP or extended POS forms).

The disjunctive canonical form (extended sum-of-product, SOP) is given as a sum of conjunctive terms, i.e. minterms.

The conjunctive canonical form (extended product-of-sum, POS) is given as a product of disjunctive terms, i.e. maxterms.

Repetitio est mater studiorum

4

MINTERMS AND MAXTERMS

All minterm is the inverse of a maxterm and vice versa.

$k = 2^n-1$ and

$$\overline{m_i^n} = M_{k-i}^n$$

and

$$M_i^n = \overline{m_{k-i}^n}$$

The indices of minterms and maxterms, i ands 2^n-1-i are the complements of each other.
In their binary forms the digits 0 and 1 are interchanged. The sum of the pairs of indices is 2^n-1, which in binary form contains only the digit 1.

Repetitio est mater studiorum

5

ADJACENT MINTERMS, MINIMIZATION

Adjacent minterms: only one logic variable asserted and negated respectively, all others are the same.

Process of contraction and minimization:
1. The adjacent minterms are contracted, the corresponding variables are eliminated.
2. In the new form the adjacent terms are again contracted, etc.
3. The process is continued till from the terms obtained no more variables can be eliminated by further contraction.

The terms obtained such way are called prime implicants of the function.

Repetitio est mater studiorum

6

TWO-LEVEL COMBINATIONAL NETWORKS (AND-OR, AND OR-AND RESPECTIVELY)

The disjunctive canonic and conjunctive canonic forms represent such two-level networks (logic sum or OR connection of minterms realized by AND gates, or logic product or AND connection of maxterms realized by OR gates).

The reductions or contractions performed during minimization result in simpler but also two-level AND-OR, or OR-AND networks respectively.

Repetitio est mater studiorum

7

KARNAUGH MAP, K-MAP

The Karnaugh map is also known as Veitch diagram (K-map or KV-map) in short). First described by Maurice Karnaugh (Bell Labs, 1950), and Edward W. Veitch (1952).

Edward W. Veitch, *A chart method for simplifying truth functions*, May 1952, Proc. Assoc. for Computing Machinery, Pittsburgh
Maurice Karnaugh, *The map method for synthesis of combinational logic circuits*, Trans. AIEE, pt. I, 72(9), 553-599, November 1953.

The Karnaugh map, besides aiding fast and transparent minimization of logic functions having not too many (say less than 7 or 8) variables, can also be used to identify and eliminate potential hazard phenomena, which would be much more difficult to achieve using Boolean algebraic methods only.

For straightforward minimization however, it is more clever to use an appropriate software …

8

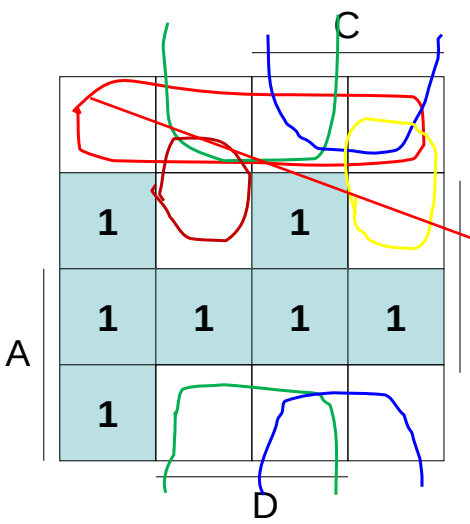
SIMPLEST CONJUNCTIVE FORM

The simplest product-of-form can (conjunctive form) can also be readily obtained from the Karnaugh map.

The minterms of the negated functions should be covered with loops, this gives the simplest sum-of-product form of the negated function. Then applying the De Morgan theorems the simplest sum-of product form of the original function is readily obtained.

9

EXAMPLE: FINDING THE SIMPLEST CONJUNCTIVE FORM (PRODUCT-OF-SUMS)



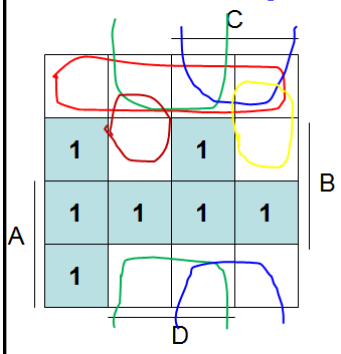
Three 4-loops and two-2-loops can be found when looping the maxterms.
E.g. the alegebraic form of the upper 4-loop, when taking the compemeted variables is

(A + B)

Maxterms: in the K map with minterms, we consider the cells containing 0, and on the edge we complement the labelling of variables.

10

SIMPLEST CONJUNCTIVE FORM (PRODUCT-OF-SUMS)



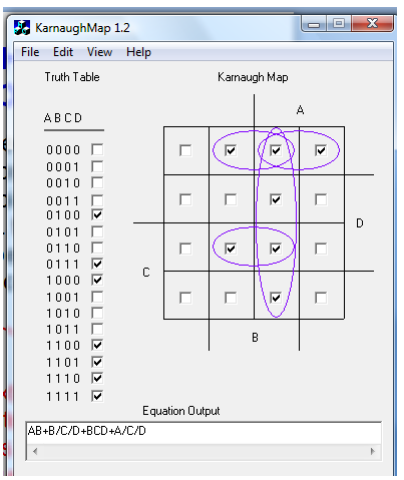
$$F = \bar{A}\bar{B} + \bar{B}\bar{D} + \bar{A}C\bar{D} + A\bar{C}\bar{D} + \bar{B}\bar{C}$$

$$F = (A+B)(B+\bar{D})(A+C+\bar{D})(A+\bar{C}+\bar{D})(\bar{B}+\bar{C})$$

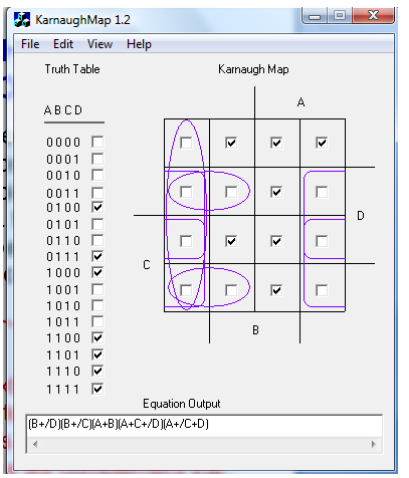
The same can be read from the Karnnaugh map too.

11

SOP AND POS COVERS



Sum-of-products



Product of sums

12

INCOMPLETELY SPECIFIED LOGIC FUNCTIONS

During looping (contracting) the values not specified (don't care terms or cells) can be freely chosen as 1 or 0, depending on which leads to a simpler solution.

Three types of mark in a Karrnaugh map (minterms!)

1	the function contains the minterm,
0	the function does not contain the minterm,
X	the minterm values is not specified (don't care).

(Instead of 0 sometimes the cell simpy remains empty)

Alternative notations: d (don't care)

13

INCOMPLETELY SPECIFIED LOGIC FUNCTIONS

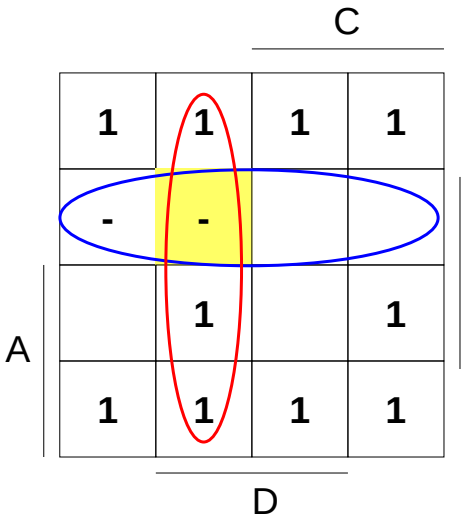
When minimizing incompletely logic functions it can happen that it is advantageous to fix the don't care values differently for SOP and for POS network.

In this case the complexity (e.g. pin number) of the two solutions can e different.

When implementing the circuit, the real minimal network can only obtained by heuristics.

14

EXAMPLE: ASSIGNING VALUE TO THE ”DON’T CARE” CELL

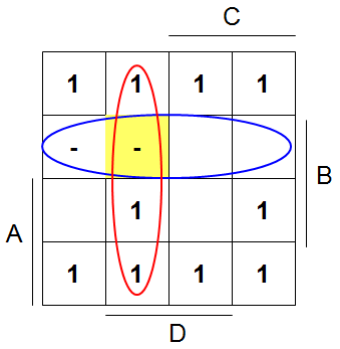


It is practical to take the value of the ”yellow” cell in the case of SOP (disjunctive) optimization as 1,

however in the case of POS (conjunctive) looping it is better to assign a value of 0 to this cell!

15

SIMPLEST ALGEBRAIC FORMS



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

$F_k = (A+B)(\overline{B}+\overline{C}+D)(B+\overline{C}+\overline{D})$

Accounting also for the inverters to generate the necessary negated input variables, SOP version involves 11, the POS form involves 14 pins!

16

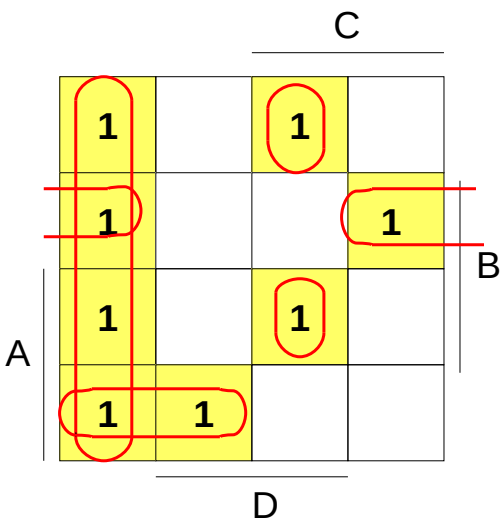
LOGIC SYNTHESIS EXAMPLES (1 AND 2)

1. Synthesize a 4-input (ABCD), 1-output (F) combinational network the F output of which is 1, if the binary numbers (MSB is A) present on the input are divisible with 3 or 4. Draw the Karnaugh map and the conceptual logic diagram

2. Repeat above, if on the input only BCD (8-4-2-1) coded decimal digit can arrive.

17

SYNTHESIS (1): SOLUTION (SOP)



Divisible by 3:
0,3,6,9,12,15
Divisible by 4:
0,4,8,12
The logic function to be implemented
$F = \Sigma^4(0,3,4,6,8,9,12,15)$
Optimized:

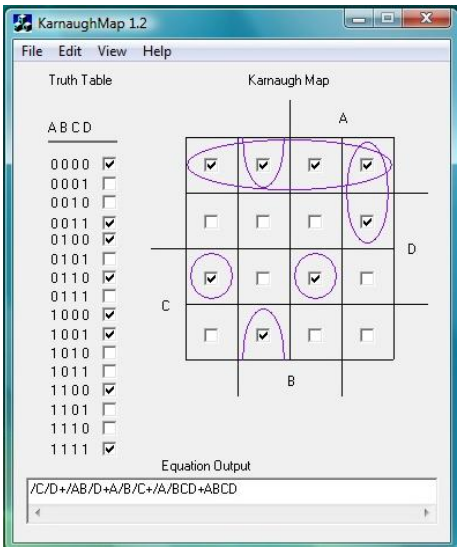
$F = \overline{C}\,\overline{D} + \overline{A}\,B\,\overline{D} + A\,\overline{B}\,\overline{C}$

$+ A\,B\,C\,D + \overline{A}\,\overline{B}\,C\,D$

(Perhaps XOR logic?)

18

SYNTHESIS (1): MACHINE SOLUTION (SOP)



19

SYNTHESIS (1): SSI GATES

Conceptual logic diagram:

1 pc 2-input AND
2 pc 3-input AND
2 pc 4-input AND
1 pc 5-input OR gate.

Optimized network: 21 pins (gate inputs).

Implementation (e.g.):

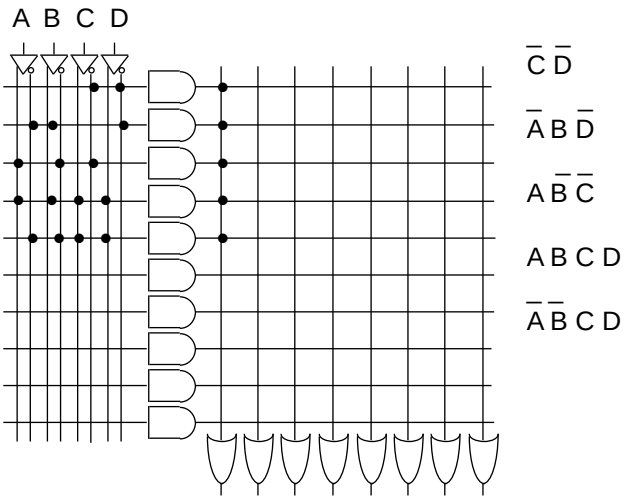
1/4 pc	74LS00 (4x2 input NAND)
2 pc	74LS20 (2x4 input NAND)
1 pc	74LS30 (1x8 input NAND)

Extended SOP (canonic form), a total of 8x4 + 1x8 = 40 pins (gate inputs) would be necessary.

In evaluating designs we will use the total pin number as the cost function.

20

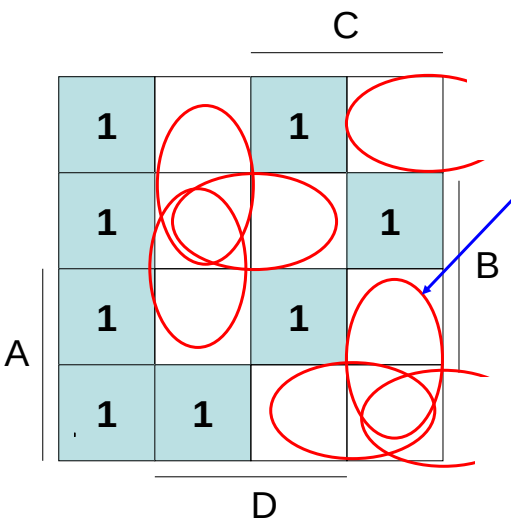
SYNTHESIS (1): PLA IMPLEMENTATION



PLA: a plane of AND gates followed with a plane of OR gates, interconnections can be established by the user.

21

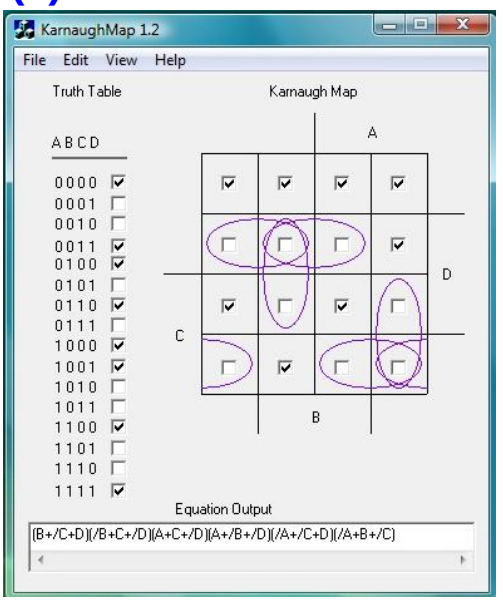
SYNTHESIS (1): SOLUTION (POS)



etc., altogether six sum terms, each containing three variables

22

SYNTHESIS (1): MACHINE SOLUTION (POS)



23

SYNTHESIS (1): SSI GATES

Conceptual logic diagram (OR-AND):

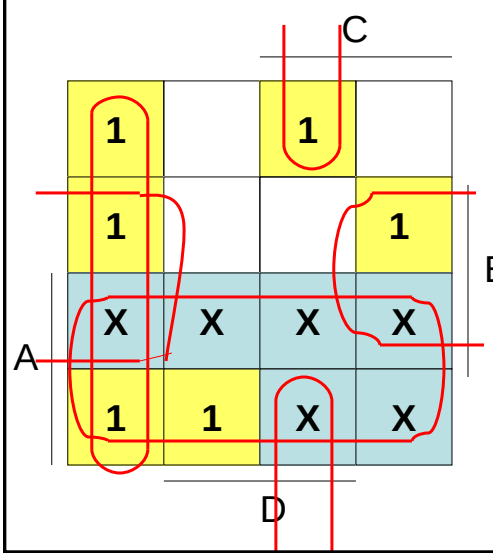
6 pc 3-input OR gate,
1 db 6 input AND gate.

Optimized network: 24 pins (gate inputs).

The extended SOP form (conjunctive canonic form) contains 8 maxterms, therefore 8x4 + 1x8 = 40 pins (gate inputs) would be needed.

24

SYNTHESIS (2): SOLUTION (SOP)



Divisible by 3: 0,3,6,9
(12,15 excluded!)
Divisible by 4: 0,4,8
(12 excluded!)

The logic function:

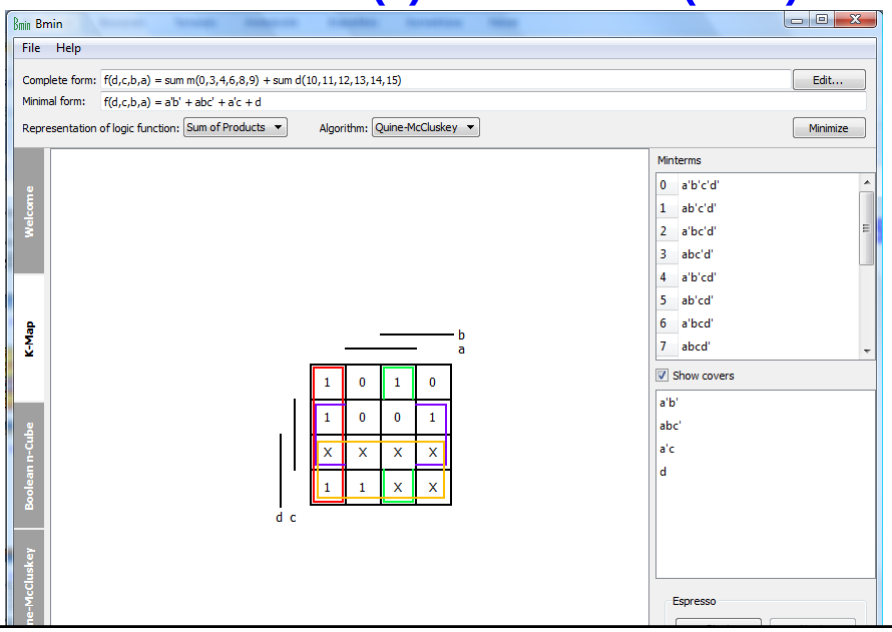
$F = \Sigma^4((0,3,4,6,8,9) + (10\text{-}15))$

Optimized:

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

25

SYNTHESIS (2): SOLUTION (SOP)



SYNTHESIS (2): SOLUTION (SOP)

Principal logic diagram consists of:

- one direct connection,
- two 2-input AND gates,
- one 3 input AND gate,
- one 4-input OR gate.

Minimized network: 12 pins (gate inputs).
Implementation (modular logic):

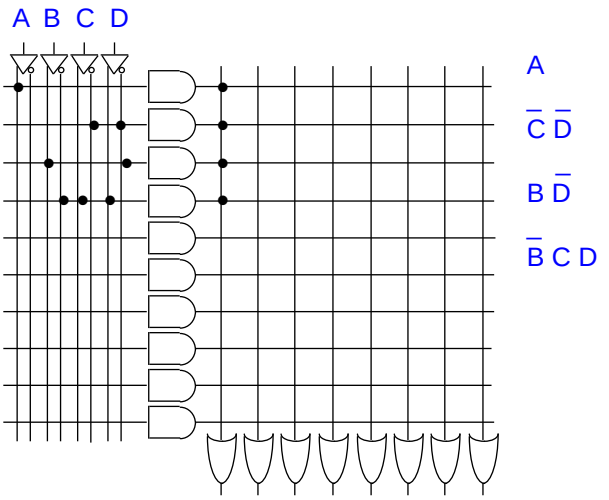
- 3/4 7400 (4x2-input NAND)
- 1 7420 (2x4-input NAND)

Implementation of the extended SOP form would need 6x4 + 1x6 = 30 gate inputs.

Note: the OR-AND network would result in a somewhat simpler network (11 pins).

27

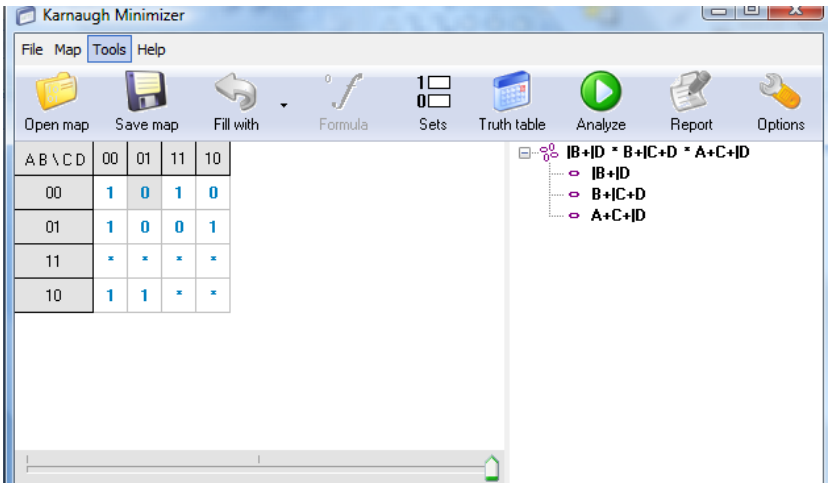
SYNTHESIS (2): PLA IMPEMENTATION



PLA: a plane of AND gates followed with a plane of OR gates, interconnections can be established by the user.

28

SYNTHESIS (2): SOLUTION (POS)



29

SYNTHESIS EXAMPLE (3): 2-BIT (SIMPLE) COMBINATIONAL ADDER

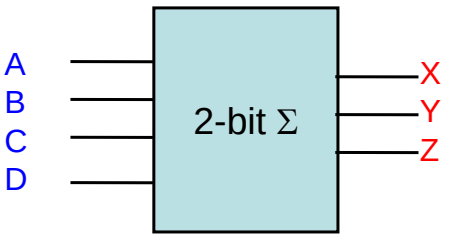
A, B, C, D are the inputs, X, Y, Z are the outputs of a combinational circuit. If the input is interpreted as two 2-bit numbers (AB, A is the MSB, and CD, C is the MSB), the output be the sum of the two binary numbers present at the input, (XYZ, X is the MSB), i. e. XYZ = AB + CD. E. g. 101 = 11 + 10 (binary addition).

Derive the truth table of the network.
Give the simplest Boolean function separately for each output.
Construct the 2-bit adder.

30

SYNTHESIS EXAMPLE (3): SOLUTION

Two-bit combinational adder



E.g. if A B C D = 1 1 0 1 then X Y Z = 1 1 0

because A B 1 1
+ C D 0 1
= X Y Z 1 1 0

31

SYNTHESIS (3): TRUTH TABLE OF 2-BIT COMBINATIONAL ADDER

A	B	C	D	X	Y	Z
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
…	…	…	…	…	…	…
1	0	1	0	1	0	0
…	…	…	…	…	…	…
1	1	1	1	1	1	0

From the truth table the logic functions to be implemented

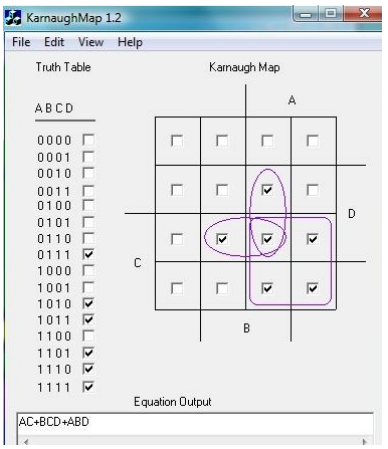
$X = \Sigma^4(7,10,11,13\text{-}15)$

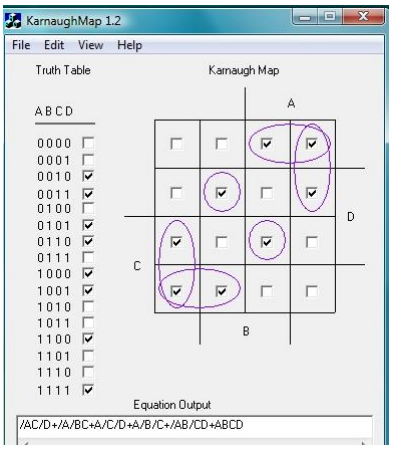
$Y = \Sigma^4(2,3,5,6,8,9,12,15)$

$Z = \Sigma^4(1,3,4,6,9,11,12,14)$

32

SYNTHESIS (3): AND-OR OPTIMIZATION





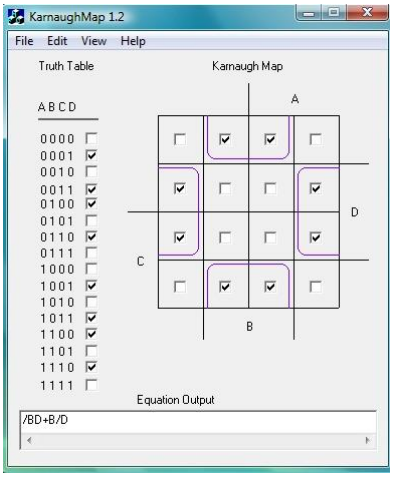
$X = A\,C + B\,C\,D + A\,B\,D$

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

(perhaps XOR logic?)

33

SYNTHESIS (3): AND-OR OPTIMIZATION



$Z = \overline{B}\,D + B\,\overline{D}$ (perhaps XOR logic?)

34

2-BIT BINARY FULL ADDER

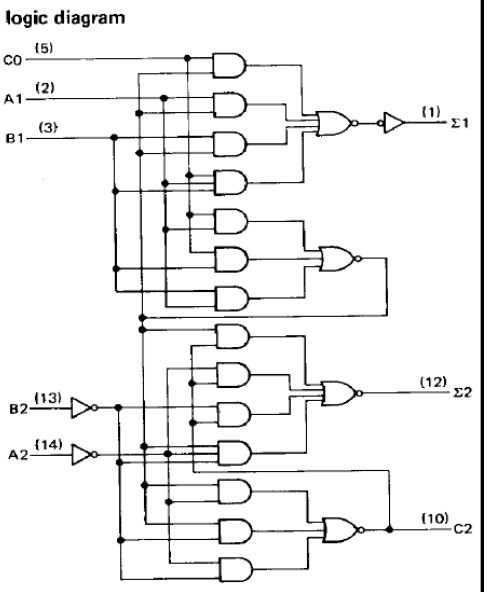
Two-bit binary full adders
5482/7482

FUNCTION TABLE

INPUTS				OUTPUTS					
				WHEN C0 = L			WHEN C0 = H		
A1	B1	A2	B2	Σ1	Σ2	C2	Σ1	Σ2	C2
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

H = high level, L = low level

logic diagram



2-BIT BINARY FULL ADDER: DESCRIPTION

These full adders perform the addition of two 2-bit binary numbers. The sum (Σ) outputs are provided for each bit and the resultant carry (C2) is obtained from the second bit. Designed for medium-to-high-speed, multiple-bit, parallel-add/serial-carry applications, these circuits utilize high-speed, high-fan-out transistor-transistor logic (TTL) and are compatible with both DTL and TTL logic families. The implementation of a single-inversion, high-speed, Darlington-connected serial-carry circuit within each bit minimizes the necessity for extensive "look-ahead" and carry-cascading circuits.

36

ANALYSIS OF 2-BIT ADDERS

2-bit (simple) adder: 11 gates, two-level circuit, delay two gate units, no input carry, no chaining possibility.

2-bit full adder: 18 gates, more than two levels, delay longer (four gate units), input carry handling, chaining possibility.

Outlook: multibit (parallel) adders: define/implement 1-bit full adder (3 inputs: two operands and carry in, 2 outputs: sum and carry out), and chain them ….

37

CALENDAR SUBSYSTEM

Determine number of days in a month (to control watch display)
Used in controlling the display of a wrist-watch LCD screen

Inputs: month, leap year flag
Outputs: number of days

Use software implementation to help understand the problem

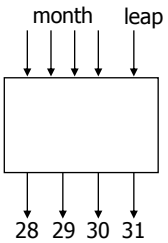
```
integer number_of_days ( month, leap_year_flag) {
     switch (month) {
          case  1: return (31);
          case  2: if (leap_year_flag == 1)
                      then return (29)
                      else return (28);
          case  3: return (31);
          case  4: return (30);
          case  5: return (31);
          case  6: return (30);
          case  7: return (31);
          case  8: return (31);
          case  9: return (30);
          case 10: return (31);
          case 11: return (30);
          case 12: return (31);
          default: return (0);
     }
}
```

38

FORMALIZE THE PROBLEM

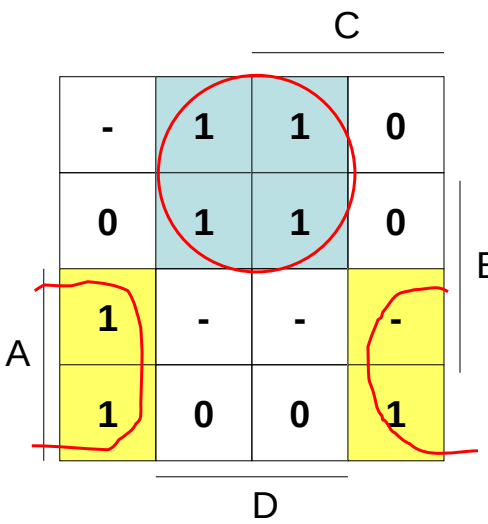
Encoding:
Binary number for month: 4 bits
4 wires for 28, 29, 30, and 31
one-hot – only one true at any time

Block diagram:



month	leap	28	29	30	31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
0101	–	0	0	0	1
0110	–	0	0	1	0
0111	–	0	0	0	1
1000	–	0	0	0	1
1001	–	0	0	1	0
1010	–	0	0	0	1
1011	–	0	0	1	0
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–

CALENDAR: 31-DAY MONTHS



Colour code shows the minimal covering

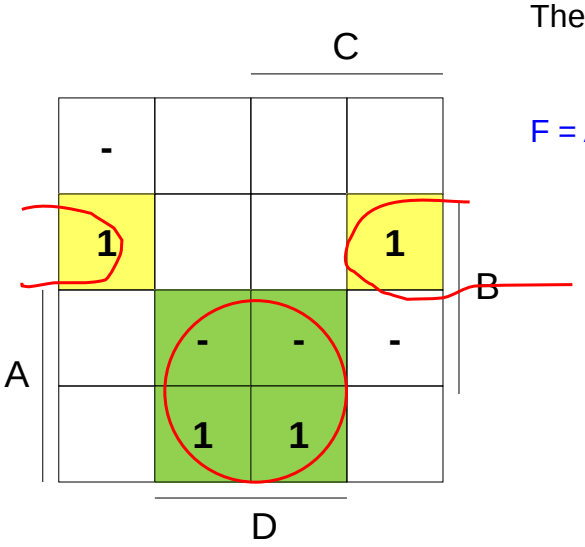
$F = \overline{A}\,D + A\,\overline{D}$

The don't care terms can be used advantageously in the minimization

CALENDAR: 30-DAY MONTHS

The optimal cover

$F = A\,D + \overline{A}\,B\,\overline{D}$



41

IMPLEMENTATION

- Discrete gates
  - 28 = m8′ m4′ m2 m1′ leap′
  - 29 = m8′ m4′ m2 m1′ leap
  - 30 = m8′ m4 m1′ + m8 m1
  - 31 = m8′ m1 + m8 m1′
- Can translate to S-o-P or P-o-S

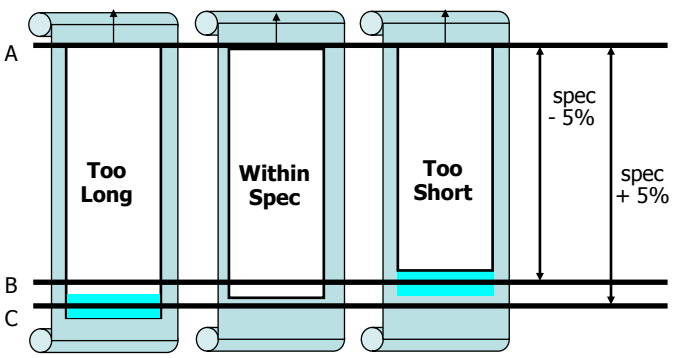
month	leap	28	29	30	31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
0101	–	0	0	0	1
0110	–	0	0	1	0
0111	–	0	0	0	1
1000	–	0	0	0	1
1001	–	0	0	1	0
1010	–	0	0	0	1
1011	–	0	0	1	0
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–

PRODUCTION LINE CONTROL

- Rods of varying length (+/-10%) travel on conveyor belt
  - Mechanical arm pushes rods within spec (+/-5%) to one side
  - Second arm pushes rods too long to other side
  - Rods that are too short stay on belt
  - 3 light barriers (light source + photocell) as sensors
  - Design combinational logic to activate the arms
- Understanding the problem
  - Inputs are three sensors
  - Outputs are two arm control signals
  - Assume sensor reads "1" when tripped, "0" otherwise
  - Call sensors A, B, C

SKETCH OF THE PROBLEM

- Position of Sensors
  - A to B distance = specification – 5%
  - A to C distance = specification + 5%



FORMALIZE THE PROBLEM

Truth Table

Show don't cares

A	B	C	Function
0	0	0	do nothing
0	0	1	do nothing
0	1	0	do nothing
0	1	1	do nothing
1	0	0	too short
1	0	1	don't care
1	1	0	in spec
1	1	1	too long

logic implementation now straightforward
just use three 3-input AND gates

"too short" = AB'C'
(only first sensor tripped)

"in spec" = A B C'
(first two sensors tripped)

"too long" = A B C
(all three sensors tripped)

PROGRAMMABLE LOGIC GATE

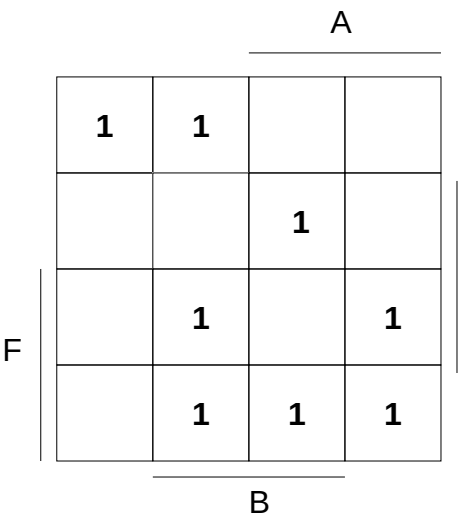
Construct a programmable logic gate!
The network has two data inputs (A, B) and two control inputs (F, G).
The gate, depending on the control code should behave as specified below:

FG	Output
00	*NEGATED* A
01	A *AND* B
10	A *OR* B
11	A *XOR* B

Try to reduce the number of gate inputs (pin number) as far as possible.

46

TRUTH TABLE / KARNAUGH MAP



NEGATED A

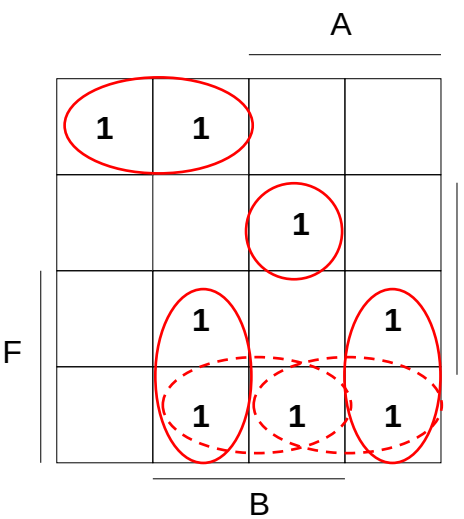
A *AND* B

A *XOR* B

A *OR* B

47

MINIMAL DISJUNCTIVE COVER



SOP: minimal cover with 21 gate inputs

NEGATED A

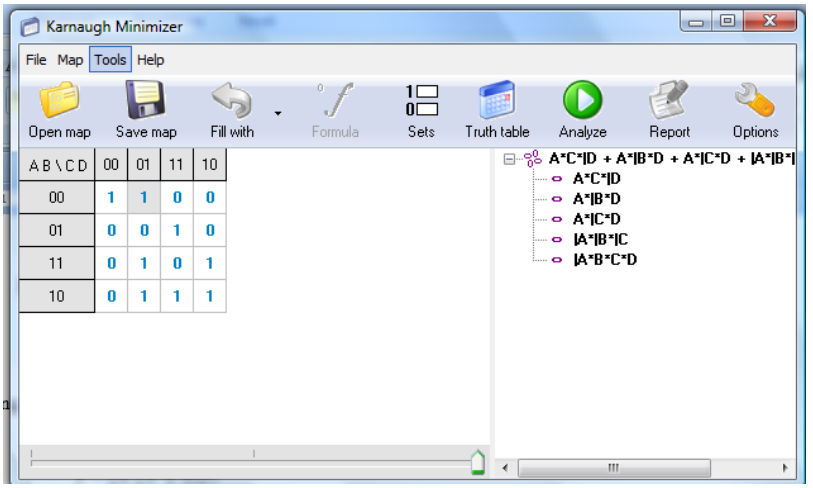
A *AND* B

A *XOR* B

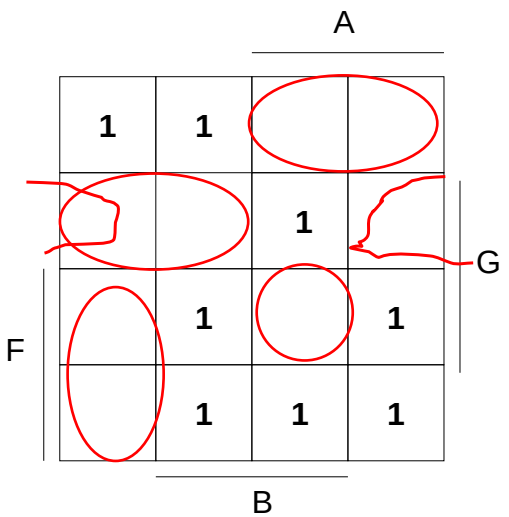
A *OR* B

48

MINIMAL DISJUNCTIVE COVER



MINIMAL CONJUNCTIVE COVER



POS: minimal cover with 21 gate inputs

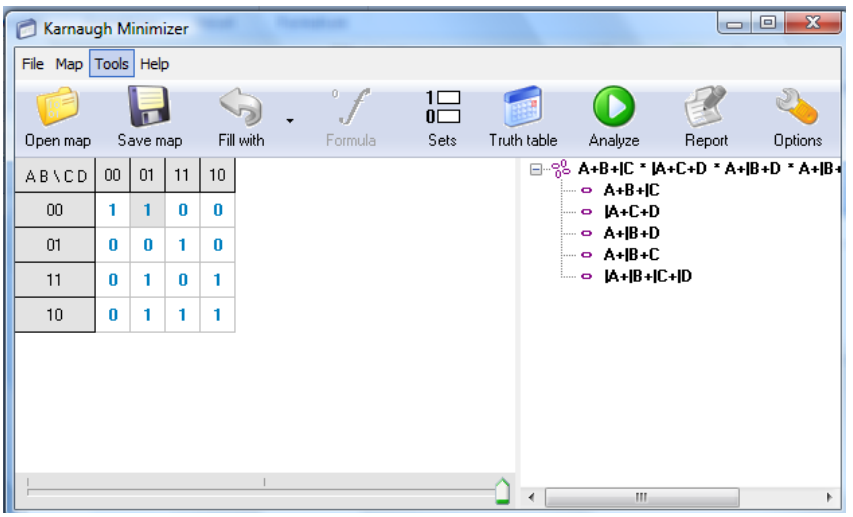
NEGATED A

A *AND* B

A *XOR* B

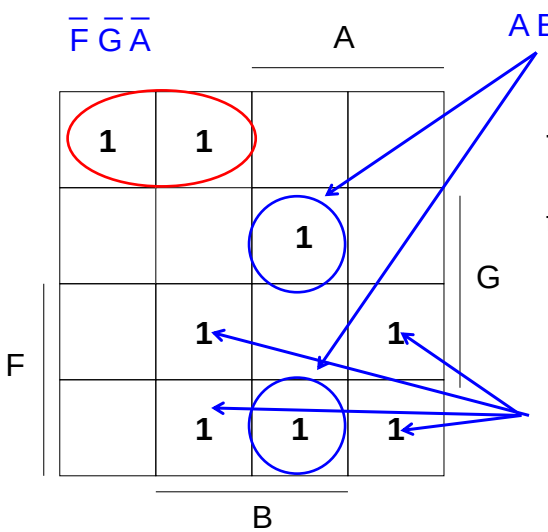
A *OR* B

MINIMAL CONJUNCTIVE COVER



51

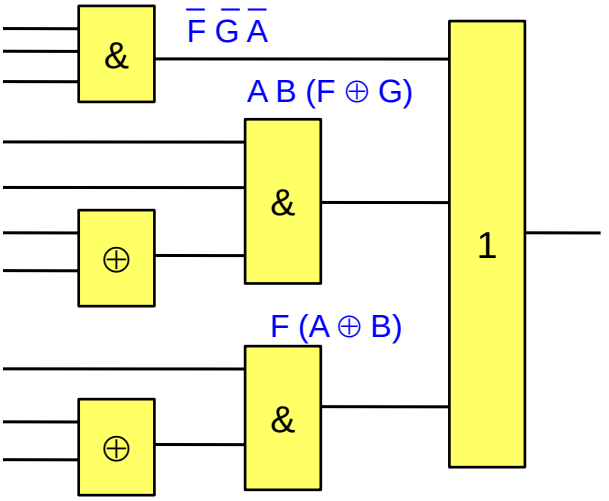
AND/OR/XOR VERSION



Total pin number: only 15!
However the result will be a three level network.

52

AND/OR/XOR IMPLEMENTATION



53

THE QUINE-MCCLUSKEY METHOD

An alternative to using K-maps is the Quine-McCluskey algorithm. The Quine-McCluskey algorithm provides a systematic approach for finding the prime implicants and selecting a minimal cover. It is functionally equivalent to the Karnaugh mapping, but the tabular form makes it more efficient for use in computer algorithms, and also give a deterministic way to check that the minima form of a Boolean function has been reached. It is sometimes referred to as the tabulation method.

54

COMPLEXITY

The tabular method is more practical than Karnaugh mapping when dealing with more than four variables, it has also a limited range of use since the runtime of the algorithm grows exponentially with the input size.

For a function of n variables the upper bound on the number of prime implicants is $3^n/n$ (i.e. 20 for n = 4, 48 for n =5, 121 for n = 6, 312 for n = 7, etc.). If n = 32 there may be over $6{,}5 \times 10^{15}$ prime implicants.

Functions with a large number of variables have to be optimized with potentially non-optimal heuristic methods.

55

ADJACENCY OF MINTERMS

The minimization is based on finding and grouping the adjacent minterms, then terms, till the further not reducible prime implicants are arrived at.

The necessary and sufficient condition of the adjacency of two minterms can be given by three statements, which should be fulfilled simultaneously.

It is important the these three statements (conditions) can be given based only on the lower indexes of the minterms.

Note: see Arató's text for details.

56

ADJACENCY CONDITION NO. 1

Two minterms are adjacent, if the difference between their decimal index is a power of two.

6	0110	$\overline{A}\,B\,C\,\overline{D} + \overline{A}\,\overline{B}\,C\,\overline{D} \rightarrow \overline{A}\,C\,\overline{D}$
2	0010	
4		

This is a necessary but not suffcient condition.

Counterexample: it is fulfilled for minterms with index 2 (i.e. 0010) and 4 (i.e. 0100), however they are not adjacent.

57

ADJACENCY CONDITION NO. 2

Two minterms are adjacent, if their binary weights (number of 1s) differ by 1.

6	0110 (2)	$\overline{A}\,B\,C\,\overline{D} + \overline{A}\,\overline{B}\,C\,\overline{D} \rightarrow \overline{A}\,C\,\overline{D}$
2	0010 (1)	
4	(1)	

This is also necessary but not sufficient condition, because just this is the condition which is not fulfilled for minterms m2 and m4 figuring in the previous counterexample.

58

ADJACENCY CONDITION NO. 3

Two minterms are adjacent, if, the decimal index of the minterm with larger binary weight is also larger than the decimal index of the other minterm.

6	0110 2	$\bar{A}\,B\,C\,\bar{D} + \bar{A}\,\bar{B}\,C\,\bar{D} \rightarrow \bar{A}\,C\,\bar{D}$
2	0010 1	
4	1	

$6 > 2$ and $2 > 1$

This is also a necessary but in itself not sufficient condition, because e.g. the minterms m7 and m9, for which the first two conditions are fullfilled, fail this 3rd condition.

59

QUINE-MCCLUSKEY ALGORITHM

It can be exaclty proven that the simultaneous fullfillement of the three conditions stated above is not only necessary but also sufficient condition for the adjacency of the two minterms in question.

This forms the basis of the Quine-McCluskey algorithm.

60

QUINE-MCCLUSKEY ALGORITHM

The Quine-McCluskey algorithm of numeric or tabular minimization analyzing solely the minterm indices based on these three conditions finds all possible adjacent pairs of minterms, then it repeats the process till finds all the prime implicants.
The method therefore involves the following two steps:

1. Finding all prime implicants of the function.

2. Use those prime implicants in a prime implicant chart to find the essential prime implicants of the function, as well as other prime implicants that are necessary to cover the function.

61

Q-M WORKED EXAMPLE

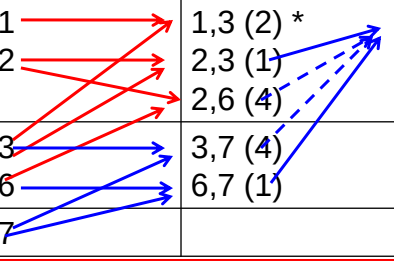
$F(A,B,C) = \Sigma^3(1,2,3,6,7)$ is to be minimized

001	HW = 1	(1)	m1, m2
010	1	(2)	m3, m6
011	2	(3)	m7
110	2		
111	3		

Group and arrange minterms in an implicant table according to their Hamming weight. Neighbours can differ only in one place. Minterms in group 2 can have neighbours only from group 1 or 3.

IMPLICANT TABLE

Size	**Minterms** m(i)	**One-cube** m(i,j)	**Two-cube** m(i,j,k,l)
1	m1 m2	1,3 (2) * 2,3 (1) 2,6 (4)	2,3,6,7 (1,4) *
2	m3 m6	3,7 (4) 6,7 (1)	
3	m7		



Merge terms from adjacent groups with decimal index differing by 1, 2, 4, 8, etc. Mark terms used. Terms can be used several times.

All terms should be accounted for. Terms which cannot be merged further are the prime implicants (*).

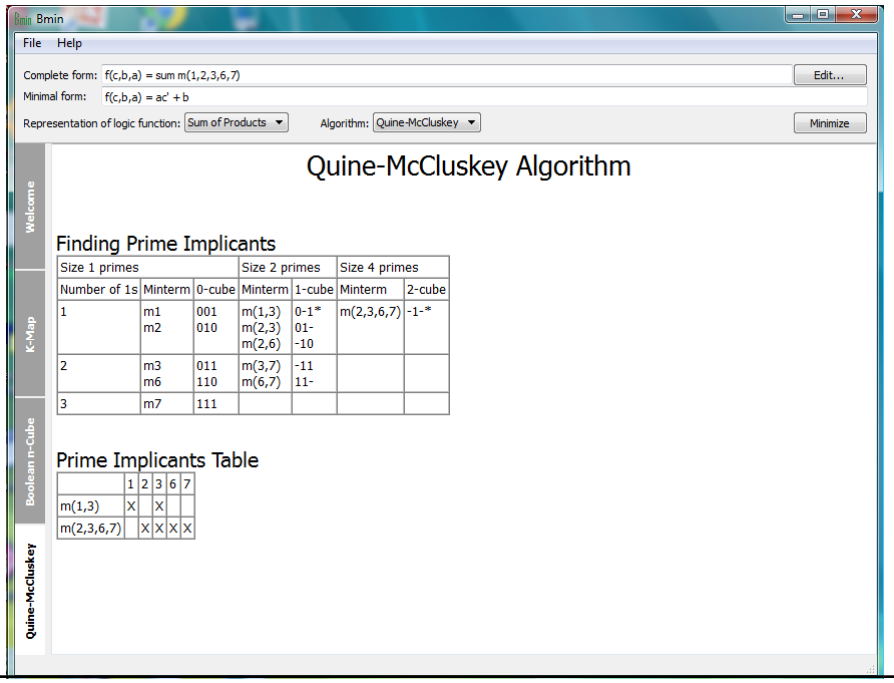
COVERING TABLE

Prime implicants	**Minterms** 1	2	3	6	7
(2,3,6,7) *		X	X	X	X
(1,3)*	X		X		

Construct a prime implicant or covering table as shown. The minterm m2 occurs only in one column, therefore m(2,3,6,7) is an essential prime implicant. This takes care of m3, m6, and m7 too. Continue … m(1,3) is also necessary because of m1.

$$F(A,B,C) = m(2,3,6,7) + m(1,3) = B + \bar{A}\,C$$

X 1 X 0 X 1



Using the Q-M Procedure with Incompletely Specified Functions

1. Use minterms and don't cares when generating prime implicants

2. Use only minterms when finding a minimal cover

REVIEW QUESTIONS

1. What are the don't-care conditions?

2. Explain the terms (a) *prime implicant*, and (b) *essential prime implicant*, (c) *non-essential prime implicant*.

3. List and discuss the necessary conditions for establishing adjacency when applying the numerical/tabular minimization (Quine-McCluskey algorithm).

4. Find and download from the the internet an appropriate software for Karnaugh map and Quine-McCluskey minimization, and learn its use.

67

PROBLEMS AND EXERCISES

1. Give the Karnaugh map of the 4-input (**A,B,C,D**) single output (**F**) combinational circuit, the output of which is **1** if
- inputs **A** and **B** have different values when inputs **C** and **D** have the same value,
or
- input **B** is the same as input **D** when inputs **A** and **C** have different values.
When writing the Karnaugh map take into account that those input combinations where all inputs have the same value never occur.

2. Given the three-variable logic function

$$\mathbf{F(A,B,C) = \Sigma^3(0,2,3,4)}$$

Find the minimized product-of-sums (POS) form.

(ANS: $\mathbf{F= (\overline{A} + B) (\overline{B} + C)}$)

68

PROBLEMS AND EXERCISES

3. Draw the simplest two-level AND-OR as well as the simplest two-level OR-AND logic diagrams for the following logic functions:

$$F = \Sigma^4 (1,2,3,5,9,10,11,14)$$

$$G = \Sigma^4 (6,9,10,11,14,15)$$

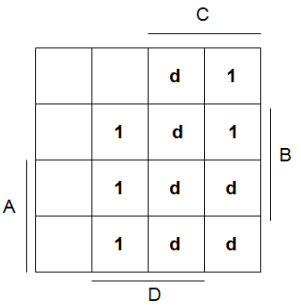
$$H = \Sigma^4 (0,4,6,8,12,14)$$

HINT: Use Karnaugh maps

69

PROBLEMS AND EXERCISES

4. In the Karnaugh map below, **d** means *don't care*, i.e. we can assign either a **0** or **1** to a cell which contains a *d*. Using these *don't care* terms, find the simplest logic expression and realization of the function.



(Source: D. L. Schilling, C. Belove, Electronic circuits: Discrete and integrated, McGraw-Hill, Inc., 1983)

70

PROBLEMS AND EXERCISES

5. Three-way light control switch problem. Assume a large room has three doors and that a switch near each door controls a light in the room. The light is turned on or off by changing the state of any one of the switches. More specifically, the following should happen:
a. The light is OFF when all three switches are open.
b. Closing any one switch will turn the light ON.
c. Then closing the second switch will have to turn OFF the light.
d. If the light is off when the two switches are closed, then by closing the third switch
the light will turn ON.

71

PROBLEMS AND EXERCISES

6. Using the Quine-McCluskey method
a. find all prime implicants of the function below,
b. determine the minimal cover.

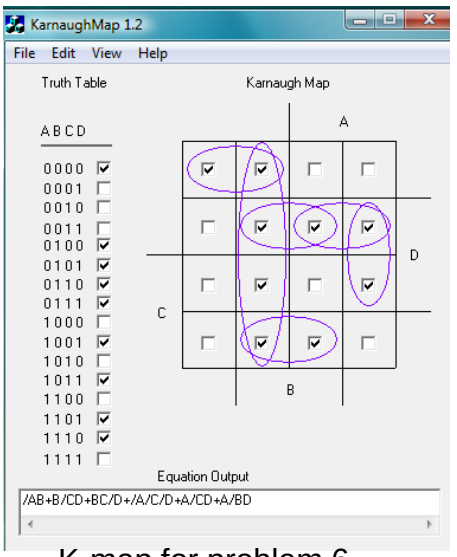
$$\mathbf{F(A, B, C, D) = \Sigma^4(0, 4, 5, 6, 7, 9, 11, 13, 14)}$$

ANS/HINT:

Six prime implicants be found (see K-map below). Four of them are essential prime implicants, and any one of the remaining two prime implicants added will result in minimal cover.
Therefore two equivalent minimal circuits exist.

72

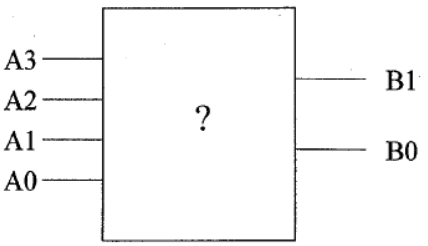
PROBLEMS AND EXERCISES



K-map for problem 6.

PROBLEMS AND EXERCISES

7. A “coincidence unit” has four binary inputs A0, A1, A2, A3 and two binary outputs B0, B1. If one or zero of the inputs is HIGH, then the outputs remain LOW. If two or more inputs are HIGH, then the outputs encode a 2-bit binary number, which is the address of the most significant HIGH input. (A3 is the MSB). For example, A3A2A1A0 = 0100 gives B1B0 = 00, and A3A2A1A0 = 0111 gives B1B0 = 2.



a. Find minimized logic expressions for B0 and B1.

b. Implement your circuit with standard gate logic. It should be possible to do this with gates having no more than two inputs.

KARNAUGH MAP SOFTWARES

kmap12.exe www.puz.com/sw/karnaugh/

kmin.zip karnaugh.shuriksoft.com/

KMapSimulator.zip members.cox.net/cyclone1980/KMapSimulation10Embedded.htm

Bmin Karnaugh map, Quine-McCluskey, Espresso

The softwares can handle both algebraic forms:
SOP: sum-of-products, disjunctive algebraic form
POS: product-of-sums, conjunctive algebraic form

Some softwares can also handle don't care terms.

75

END OF LECTURE

76