DIGITAL TECHNICS

Dr. Bálint Pődör

Óbuda University,
Microelectronics and Technology Institute

2. LECTURE: LOGIC NETWORK MINIMIZATION

2nd (Autumn) term 2018/2019

1

2. LECTURE: CONTENTS

2

CANONICAL FORMS OF LOGIC FUNCTIONS

It is expedient to base the synthesis of combinational circuit on the algebraic form of the logic function to be realized. 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.

3

EMPHASIS: DISJUNCTIVE CANONICAL FORM EXTENDED SOP

Disjunctive canonic form (or extended sum-of-product form): Algebraic form consisting of sum of logic product terms (AND-OR) having the distinctive property that in each product term all variables are contained either in asserted or in negated form.

E. g.

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

$$F(ABC) = m_2^3 + m_3^3 + m_4^3 + m_6^3$$

$$F = \Sigma^3(2,3,4,6)$$

4

EMPHASIS: CONJUNCTIVE CANONICAL FORM EXTENDED POS

Conjunctive canonic form (or extended product-of-sum form): Algebraic form consisting of product of logic sumt terms (OR-AND) having the distinctive property that in each sum term all variables are contained either in asserted or in negated form.

$$\overline{\overline{F(ABC)}} = F(ABC) =$$

$$(A + B + C)\ (A + B + \overline{C})\ (\overline{A} + B + \overline{C})\ (\overline{A} + \overline{B} + \overline{C})$$

$$F(ABC) = M_7{}^3M_6{}^3M_2{}^3M_0{}^3$$

$$F(ABC) = \Pi^3(0,2,6,7)$$

5

CONVERSION BETWEEN CANONICAL FORMS: ”GOLDEN RULE”

Function (extended SOP): list minterms of value 1

$$F(ABC) = \Sigma^3(2,3,4,6)$$

Negated function: list minterms of value 0

$$\overline{F}(ABC) = \Sigma^3(0,1,5,7)$$

To obtain maxterms and extended POS form complement minterm indices

$$F(ABC) = \Pi^3(0,2,6,7)$$

6

SIMPLIFICATION/MINIMIZATION OF LOGIC FUNCTION

Aim: To find the most economic or cheapest implementation of the specified combinational network.

Specification: text, truth table, Boolean expression, canonical form, etc.

What is economical cheap? Depends on the ”environment” (hardware base).

Discrete ICs (LSI/MSI, several gates in one IC): minimize the number of ICs, or the number of gates, or number of inputs (pins).

Programmable logic: minimization of the resources (logic cells) used.

VLSI: minimize the chip area, time delays, etc.

Simplest analysis: minimization of the number of inputs (pin count).

7

SIMPLIFICATION GOALS

Goal - minimize the cost of realizing a logic function.

Cost measures and other considerations:

- Number of gate inputs
- Number of gates
- Number of levels
- Gate fan in and/or fan out
- Interconnection complexity
- Preventing hazards

Two-level realizations:

- Minimize the number of gates (terms in logic function)
- Minimize the fan in (literals in logic function i.e. inputs/pins in gates)

8

SIMPLIFICATION/MINIMIZATION

In the past the main aim was to minimize the number of gate circuits implementing a given combinational circuit in order to decrease the number of electronic components.

Nowadays the main motivation for the minimization of logic network is to decrease the logic resources in a PLD of FPGA, to decrease the area in VLSIs, and to increase the operational speed and reliability of the circuits.

”The smallest number of failures are caused by those components which are NOT included in the network.” (Dr. Tóth Mihály, professor emeritus, Székesfehérvár.)

9

LOGIC DESIGN: MINIMIZATION METHODS

Intuition
Quickly run out of steam

Truth tables
Algebra, Quine’s method

Minterm or maxterm
De Morgan

Graphical
Boolean cubes
Karnaugh maps

Computer algorithms
Quine-McClusky (tabular) method

10

QUINE'S METHOD

Quine's method helps to perform the algebraic minimization in a systematic and (hopefully) error-free way.

The method consists of factoring out the common factors from pairs of minterms in such a way that in the parentheses only the sum of one variable and its complement remained, which sum is trivially 1.
The process is applied to all possible pairs, then it is repeated with the new sets of terms, etc.

The method thus generates all prime implicants, therefore in the second phase the essential prime implicants should be selected to obtain the minimal cover.

11

QUINE'S METHOD: DEMO

$F(A,B,C) = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C}$
(minterms: 2 3 4 6)

Minterms	I.	II.	III.
2	$\overline{A}B\overline{C}$ √	(2,3) $\overline{A}B$	No entries
3	$\overline{A}BC$ √	(2,6) $B\overline{C}$	
4	$A\overline{B}\,\overline{C}$ √	(4,6) $A\overline{C}$	
6	$AB\overline{C}$ √		

All prime implicants are contained in column II.

12

QUINE'S METHOD: DEMO

Prime implicant table

	2	3	4	6
(2,3) $\overline{A}B$*	X	X		
(2,6) $B\overline{C}$	X			X
(4,6) $A\overline{C}$*			X	X

Identification of essential prime implicants to obtain the minimal cover

Minimal cover: $F(A,B,C) = \overline{A}B + A\overline{C}$

13

THE QUINE-MCCLUSKEY METHOD

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 gives a deterministic way to check that the minimal form of a Boolean function has been reached.

It is sometimes referred to as the tabulation method.

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 $6.5x10^{15}$ prime implicants.

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

15

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 formulated in three statements, which should be fulfilled simultaneously.

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

Note: see Arató’s text for details.

16

ADJACENCY CONDITION FOR TWO MINTERMS NO. 1

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

$$\begin{array}{ll} 6 & 0110 \\ 2 & 0010 \\ \hline 4 = 2^2 & \end{array} \qquad \overline{A}\,B\,C\,\overline{D} + \overline{A}\,\overline{B}\,C\,\overline{D} \rightarrow \overline{A}\,C\,\overline{D}$$

This is a necessary but not suffcient condition.

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

17

ADJACENCY CONDITION FOR TWO MINTERMS NO. 2

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

$$\begin{array}{lr} 6 & 0110\ (2) \\ 2 & 0010\ (1) \\ \hline 4 & (1) \end{array} \qquad \overline{A}\,B\,C\,\overline{D} + \overline{A}\,\overline{B}\,C\,\overline{D} \rightarrow \overline{A}\,C\,\overline{D}$$

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.

18

ADJACENCY CONDITION FOR TWO MINTERMS 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
2      0010 1
---------------
4           1
```

$$\bar{A}BC\bar{D} + \bar{A}\bar{B}C\bar{D} \rightarrow \bar{A}C\bar{D}$$

$6 > 2$ 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.

19

QUINE-MCCLUSKEY ALGORITHM

It can be shown that the simultaneous fulfillment of these three conditions is not only necessary but also sufficient for the establishment of the adjacency of two minterms.
The Quine-McCluskey algorithm is based on this.

The Quine-McCluskey method 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.

20

USING 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

THE PEOPLE

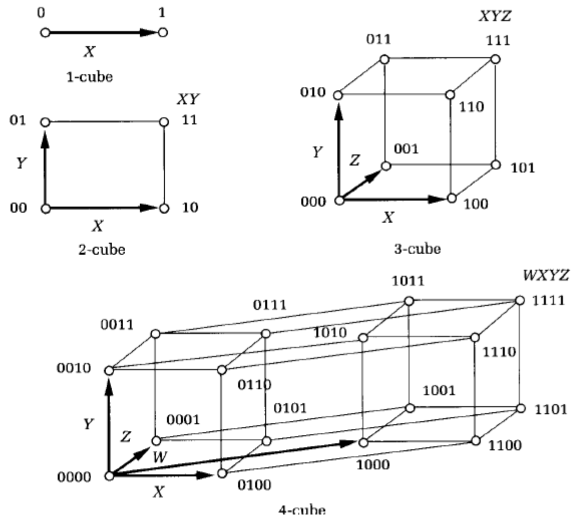
Willard Van Orman Quine (1908-2000)
Spent his entire career teaching philosophy and mathematics at Harvard University, his alma mater, where he held the Edgar Pierce Chair of Philosophy from 1956 to 1978.
A computer program whose output is its source code is called a ”quine” named after him.

Edward McCluskey
McCluskey worked on electronic switching systems at the Bell Telephone Laboratories from 1955 to 1959
1959: Princeton. Professor of Electrical Engineering and Director of the University Computer Center.
1966: joined Stanford University,
Currently Emeritus Professor of Electrical Engineering and Computer Science.

22

BOOLEAN CUBES

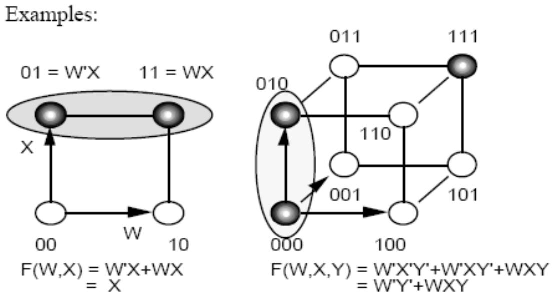
An n-input Boolean function can be represented graphically as a "cube" in an n-dimensional Boolean space.



23

LOGIC MINIMIZATION USING BOOLEAN CUBES

Examples:

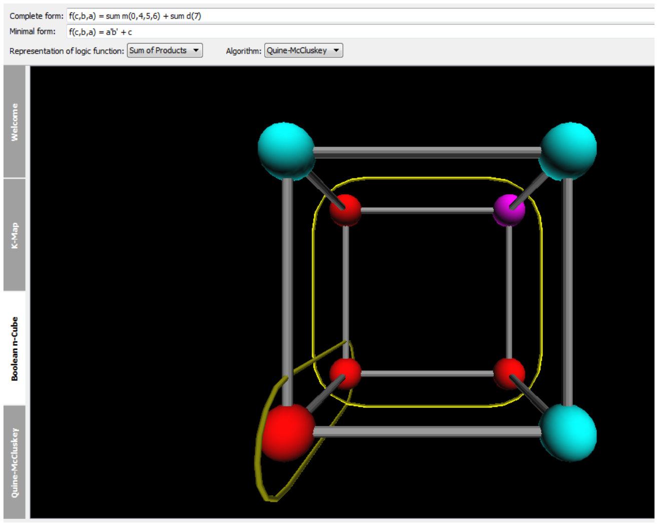


NOTE: Any two nodes (minterms/maxterms) of the function that are one unit distant apart (adjacent) may be grouped and expressed as the literal that remains constant while moving from one point to the other, i.e.
ab'+ab = a(b'+b) = a or
(a+b')(a+b) = a.

Klinkhachorn:Cpe 71:9/25/1999

24

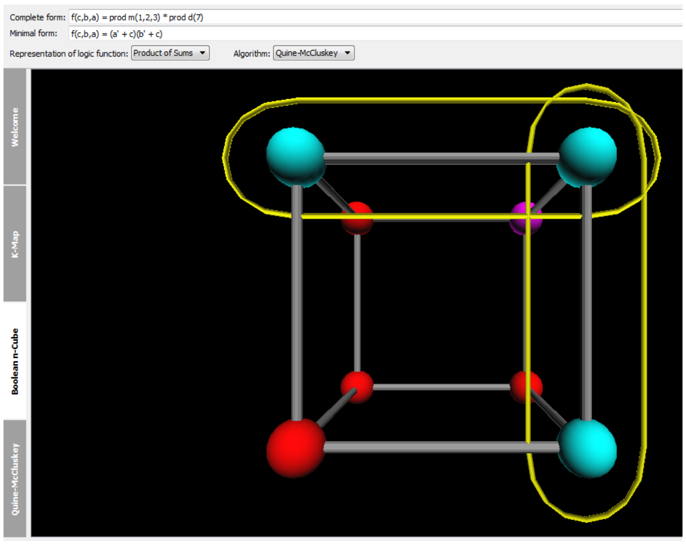
BOOLEAN CUBE EXAMPLE: SOP


Complete form: f(c,b,a) = sum m(0,4,5,6) + sum d(7)
Minimal form: f(c,b,a) = a'b' + c
Representation of logic function: Sum of Products
Algorithm: Quine-McCluskey

F= $\Sigma^3((1,4,5,6) + X(7))$ SOP looping

25

BOOLEAN CUBE EXAMPLE: POS


Complete form: f(c,b,a) = prod m(1,2,3) * prod d(7)
Minimal form: f(c,b,a) = (a' + c)(b' + c)
Representation of logic function: Product of Sums
Algorithm: Quine-McCluskey

F= $\Sigma^3((1,4,5,6) + X(7))$ POS looping

26

The Karnaugh map:

Construction, properties, and applications

What everybody should know about the Karnaugh map….

27

THE KARNAUGH MAP (K-MAP)

Karnaugh maps (K-maps) - convenient tool for representing switching functions of up to six variables. In fact it is a practical rearrangement of the truth table based on the concept of adjacency.

K-maps form the basis of useful heuristics for finding minimal SOP and POS representations.

An *n*-variable K-map has 2^n cells with each cell corresponding to a row of an *n*-variable truth table.

K-map cells are labeled with the corresponding truth-table row.

K-map cells are arranged such that adjacent cells correspond to truth rows that differ in only one bit position (*logical adjacency*).

Switching functions are mapped (or plotted) by placing the function's value (*0*,*1*,*d*) in each cell of the map.

28

THE KARNAUGH MAP (K-MAP)

The Karnaugh map, known also as Veitch diagram is a graphic tool to facilitate management of Boolean expressions.

It was invented in 1953 by *Maurice Karnaugh*, an engineer at Bell Labs, and is an improvement on the Veitch diagram (*Edward W. Veitch* in 1952).

The Map Method for Synthesis of Combinational Logic Circuits, Trans. AIEE, pt. 1, 72 (9) 593-599, November 1953

Normally, extensive calculations are required to obtain the minimal form of a Boolean function.

Instead the Karnaugh maps make use of the human brain's excellent pattern matching capability to arrive at the simplest expression.

In addition, K-maps permit the rapid identification and elimination of potential race hazards, something that Boolean equations alone cannot do.

29

THE PEOPLE

Maurice Karnaugh (1924-   ), American physicist.

Yale University, BSc. in maths and physics (1949), MSc. (1950), PhD in physics (1952) (*The Theory of Magnetic resonance and Lambda-Type Doubling in Nitric-Oxide*).

Bell Labs (1952-1966), IBM (1966-1989).

30

THE KARNAUGH MAP (CONT'D)

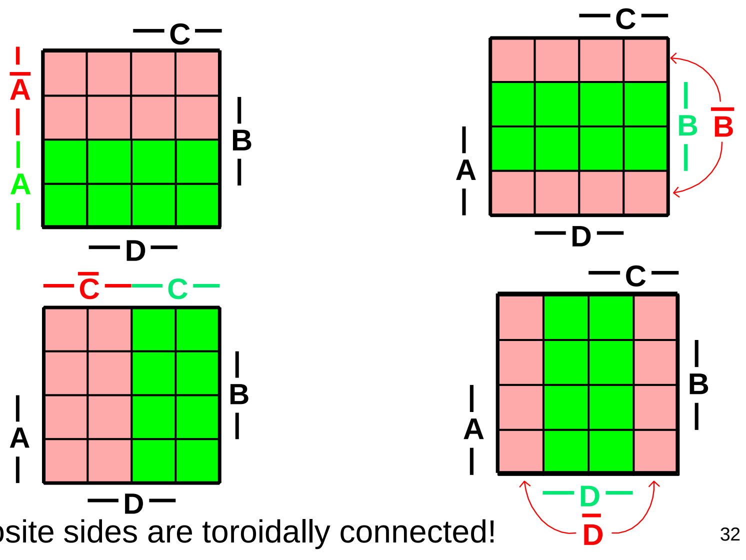
Rectangular logic diagrams by *Allan Marquand* and *Lewis Carrol* (of *Alice in Wonderland* fame) were the forerunners of a modern form known today as Karnaugh maps.

Since 1950 Karnaugh maps quickly became one of the mainstays of the digital logic and computer designer's tool-chest.

Basic references:
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.

31

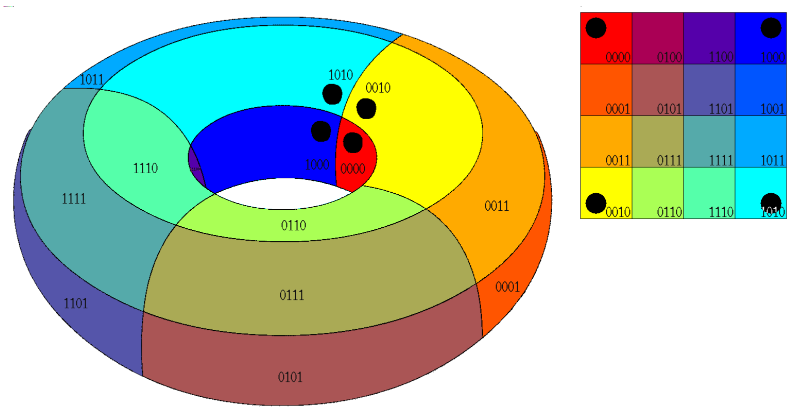
KARNAUGH MAP – GENERALIZATION OF THE VENN DIAGRAM



The opposite sides are toroidally connected!

32

FOUR VARIABLE K-MAP ON TOROID



33

ON VENN DIAGRAMS

Venn diagrams are illustrations in the branch of mathematics known as set theory. They show the mathematical or logical relationships between different groups of things (sets). A Venn diagram shows all the possible logical relations between sets.

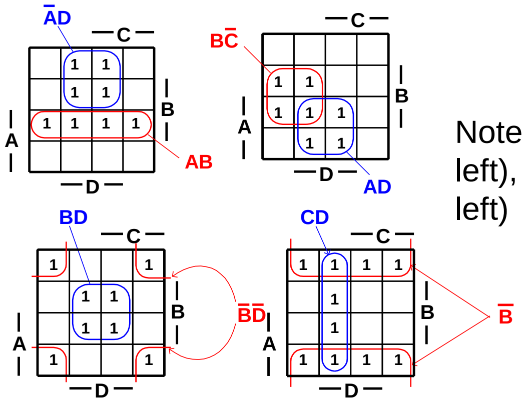
John Venn (1834-1923) was a British philosopher and mathematician who introduced the Venn diagram in 1881.

A stained glass window in Caius College, Cambridge, where he studied and spent most of his life, commemorates *John Venn* and represents a Venn diagram.

Similar and related diagrams:
Euler diagrams, Johnston diagrams, Karnaugh maps, Peirce diagrams, Edward’s Venn diagrams.

34

EXAMPLES OF COVERING (LOOPING) ON 4-VARIABLE KARNAUGH MAP

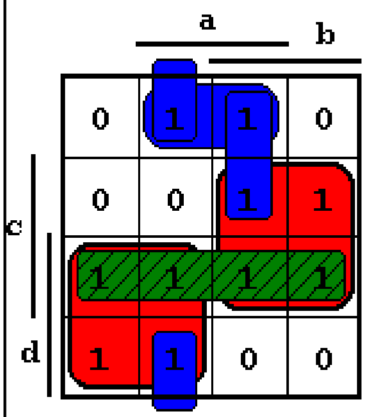


Note: race (static) hazard (top left), symmetric function (down left)

A four-variable map has 16 cells. The four-variable map is continuous from the left edge to the right edge and is also continuous from the top edge to the bottom edge.

35

KARNAUGH MAP: PRIME IMPLICANTS



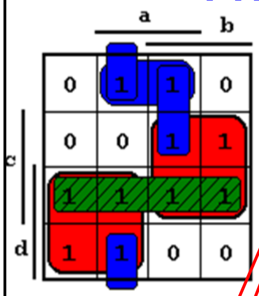
🟥 Essential prime implicants

▨ Non-essential prime implicants

🟦 Classification is conditional

36

PRIME IMPLICANTS TABLE



Essential PIs: a and b cover 6,7,8,9,12,13,14,15

Remaining 1 and 3 can be covered by PI d.

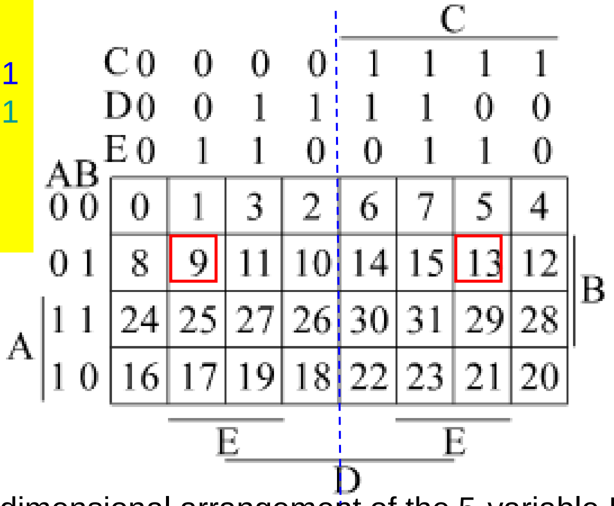
	1	3	6	7	8	9	12	13	14	15
a 6, 7,14,15			x	x					x	x
b 8,9,12,13					x	x	x	x		
c 12,13,14,15							x	x	x	x
d 1,3	x	x								
e 1,9	x					x				
f 3,7		x		x						

IN PLANE (REFLECTION) KARNAUGH MAP FOR 5 VARIABLES

Adjacency:

9 →0 1 0 0 1
13→0 1 1 0 1

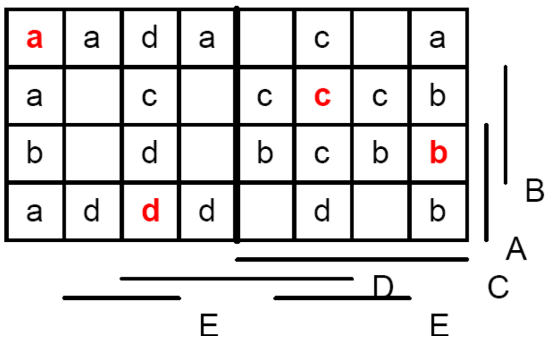
Reflection symmetry



	C	0	0	0	0	1	1	1	1
	D	0	0	1	1	1	1	0	0
	E	0	1	1	0	0	1	1	0
AB	0 0	0	1	3	2	6	7	5	4
	0 1	8	9	11	10	14	15	13	12
	1 1	24	25	27	26	30	31	29	28
	1 0	16	17	19	18	22	23	21	20

For a 3-dimensional arrangement of the 5-variable K-map and looping on it cf. Zsom Vol. 1, p.129

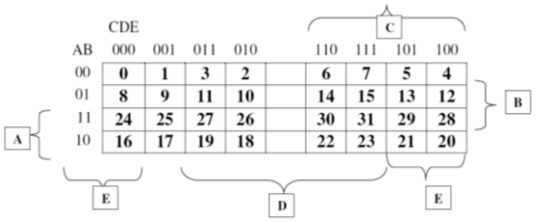
ADJACENCY FOR 5-VARIABLES (REFLECTION MAP)



a	a	d	a		c		a
a		c		c	c	c	b
b		d		b	c	b	b
a	d	d	d		d		b

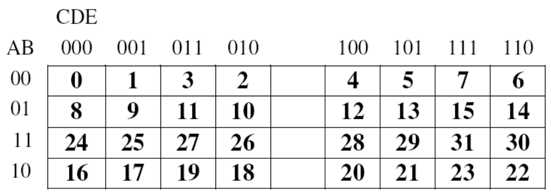
There are several different formats of a 5-variable K map, the two most popular ones are the *reflection map* and *overlay map*. The above version of the five-variable Karnaugh map, is a Gray code map or reflection map. The top of the map is numbered in full Gray code. The Gray code reflects about the middle of the code.

KARNAUGH MAPS: FIVE VARIABLES



AB \ CDE	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

Minterm assignment in five-variable reflection Karnaugh map



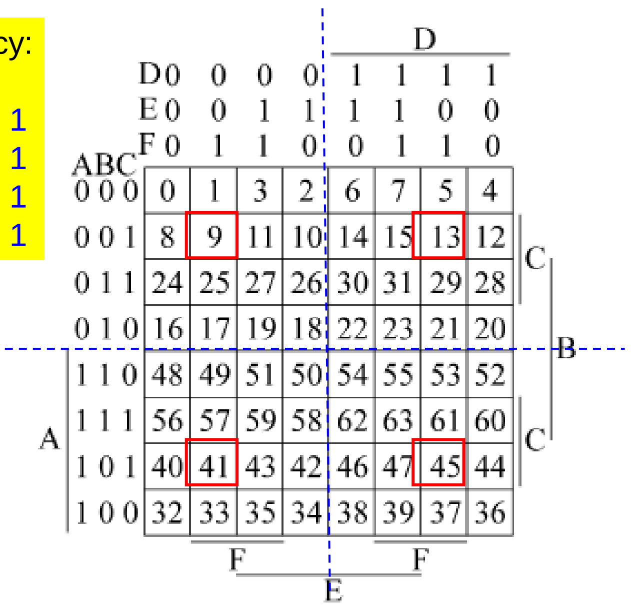
AB \ CDE	000	001	011	010		100	101	111	110
00	0	1	3	2		4	5	7	6
01	8	9	11	10		12	13	15	14
11	24	25	27	26		28	29	31	30
10	16	17	19	18		20	21	23	22

Minterm assignment in five-variable overlay Karnaugh Map

KARNAUGH MAP FOR 6 VARIABLES

Adjacency:

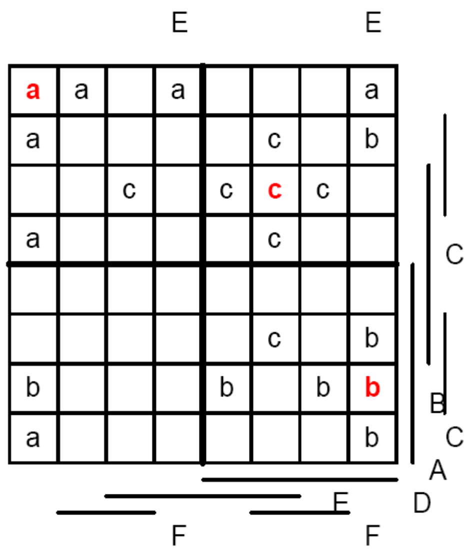
0 0 1 0 0 1
0 0 1 1 0 1
1 0 1 0 0 1
1 0 1 1 0 1



ABC \ DEF	000	001	011	010	110	111	101	100
000	0	1	3	2	6	7	5	4
001	8	9	11	10	14	15	13	12
011	24	25	27	26	30	31	29	28
010	16	17	19	18	22	23	21	20
110	48	49	51	50	54	55	53	52
111	56	57	59	58	62	63	61	60
101	40	41	43	42	46	47	45	44
100	32	33	35	34	38	39	37	36

41

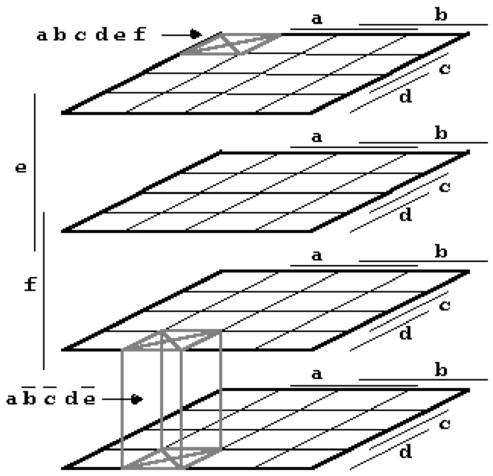
ADJACENCY FOR 6-VARIABLES



a	a		a				a
a					c		b
		c		c	c	c	
a					c		
					c		b
b				b		b	b
a							b

42

KARNAUGH MAP VARIATIONS FOR 6-VARIABLES



Three dimensional arrangement of 6 variable K map.
(For an example of looping cf. Zsom Vol. 1. p. 129.)

43

Further applications of Karnaugh maps:

Don’t care terms, symmetric functions, analysis of combinational circuits, race hazards, K map softwares

44

DON’T CARE TERMS

Don’t care terms in the truth table:
1. The given input combination cannot occur.
2. The given input combination can occur, but the logic network following the output does not consider it.

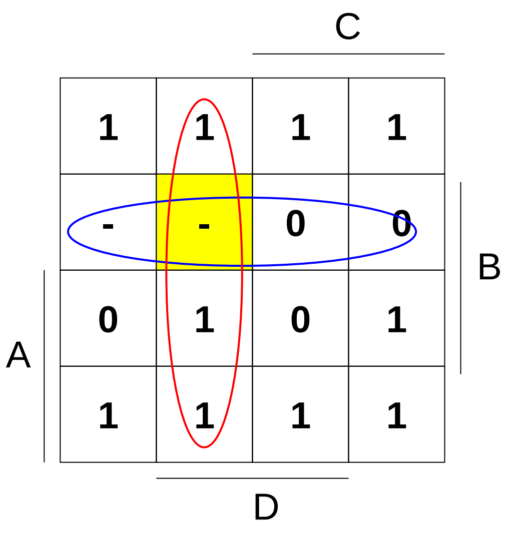
Representative example: BCD-to-seven segment display decoder/driver.

Optimal covers utilize don’t care terms.

Don’t care terms exist when planning a combinational circuit, but NEVER when a realized CC is analyzed!

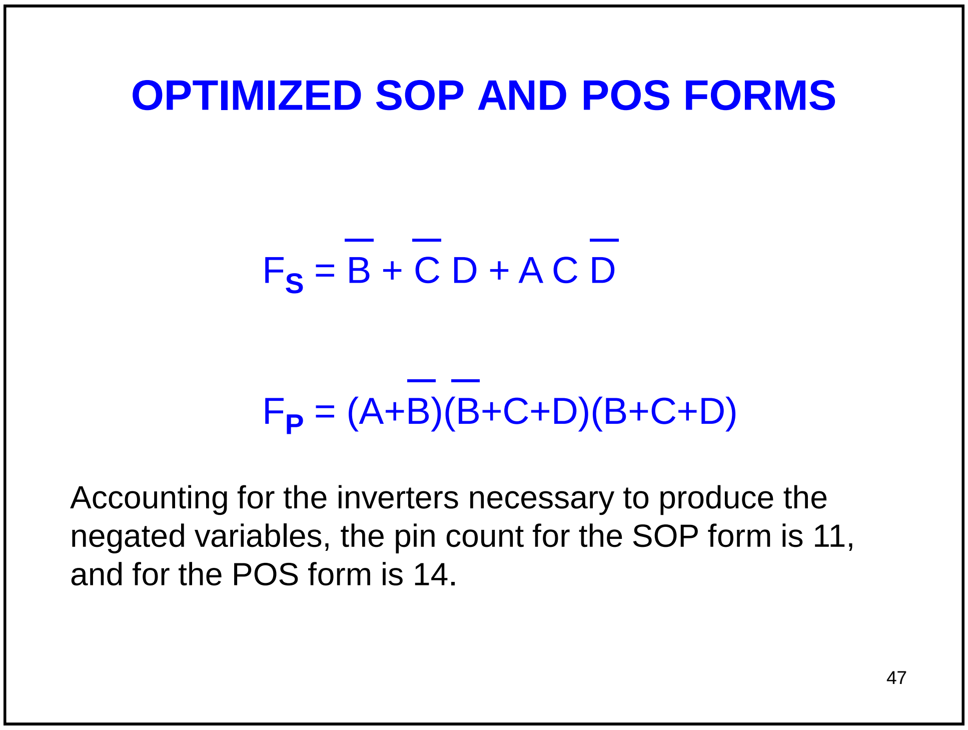
45

FIXING THE VALUE(S) OF THE DON’T CARE TERMS



The yellow coded cell is assigned a value of 1 when the sum of products (SOP) form is evaluated, and a value of 0 when the product of sums (POS) form is evaluated!

46



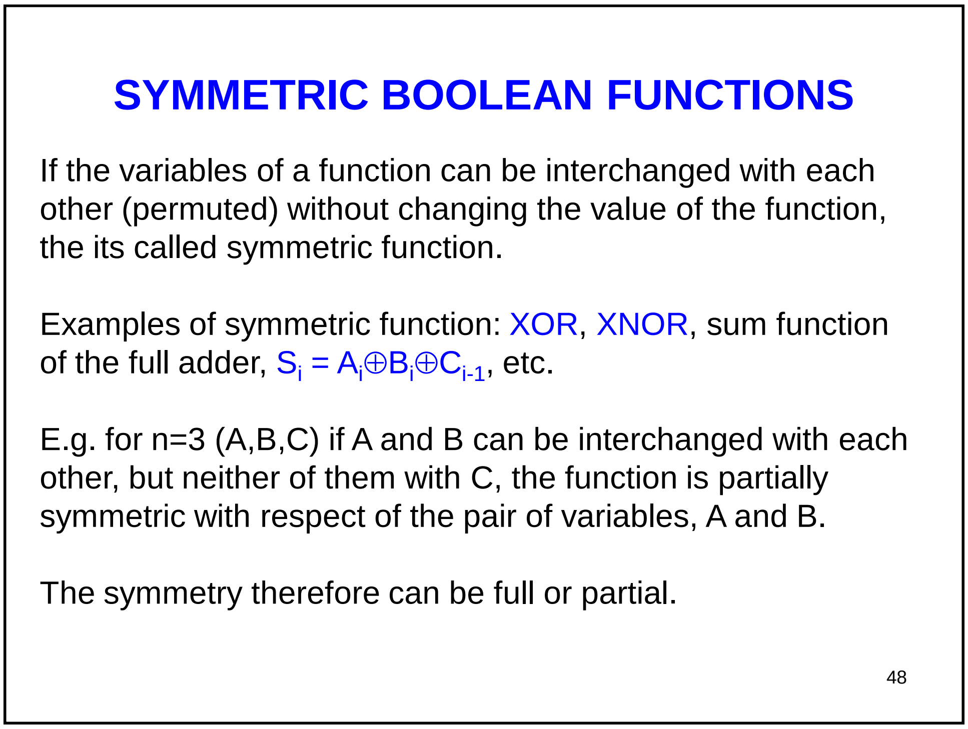
OPTIMIZED SOP AND POS FORMS

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

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

Accounting for the inverters necessary to produce the negated variables, the pin count for the SOP form is 11, and for the POS form is 14.

47



SYMMETRIC BOOLEAN FUNCTIONS

If the variables of a function can be interchanged with each other (permuted) without changing the value of the function, the its called symmetric function.

Examples of symmetric function: XOR, XNOR, sum function of the full adder, $S_i = A_i \oplus B_i \oplus C_{i-1}$, etc.

E.g. for n=3 (A,B,C) if A and B can be interchanged with each other, but neither of them with C, the function is partially symmetric with respect of the pair of variables, A and B.

The symmetry therefore can be full or partial.

48

EXCLUSIVE OR LOGIC

The symmetric functions have special characteristics, like they form a ”chessboard” pattern on the Karnaugh map (at least partially), and they can be simplified by using XOR functions as functional elements.

Reduction of a function to XOR form is characterized by a Karnaugh map where the 1s are diagonally opposite to each other.

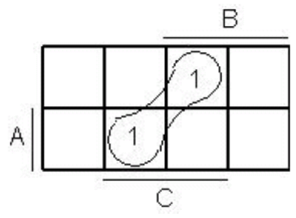
In the general context of minimization of Boolean functions XOR gates can, for certain problems, provide a more economic implementation than by using other logic gates.

Two examples are the 1-bit full adder, and the binary-to-Gray code conversion.

49

UTILIZATION OF SYMMETRY: EXAMPLE

$F^3 (A,B,C)=\sum(3,5)$



XOR, XNOR
Look for “checkerboard” squares
Depends whether XOR/XNOR gates available

$$F^3=(A,B,C) = \overline{A}BC + A\overline{B}C = C \,\&\, (A\overline{B} + \overline{A}B) = C \,\&\, (A \oplus B)$$

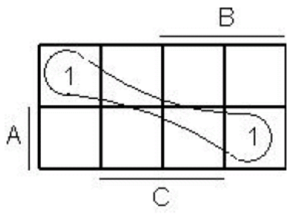
AND-OR (NAND-NAND) implementation: 8 pins 3 gates

Implementation using XOR: 4 pins 2 gates

Symmetry: partial, with respect to A and B

50

UTILIZATION OF SYMMETRY: EXAMPLE



$$F^3(A,B,C) = \bar{A}\bar{B}\bar{C} + AB\bar{C} = \bar{C}\&(AB + \bar{A}\bar{B}) \quad = C(\bar{A} \oplus B)$$

AND-OR (NAND-NAND) implementation: 8 pins 3 gates

Implementation using XOR: 4 pins 2 gates

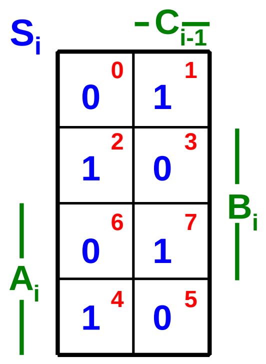
Symmetry: partial, with respect to A and B

51

EXAMPLE: S_i (SUM) FUNCTION OF THE 1 BIT FULL ADDER

i	(4) A_i	(2) B_i	(1) C_{i-1}	S_i
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

„chessboard pattern” symmetric function

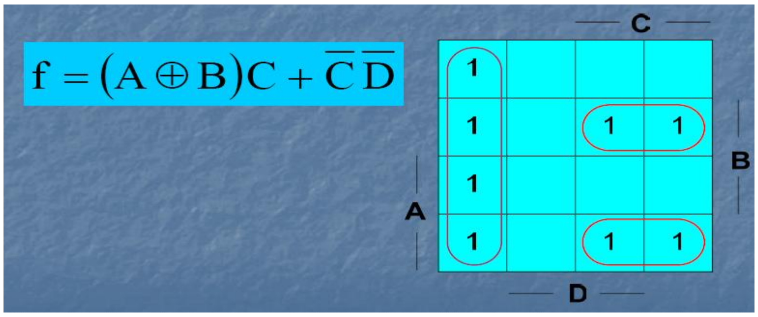


Sum function $S_i = A_i \oplus B_i \oplus C_{i-1}$

52

EXAMPLE: PARTIALLY SYMMETRIC FUNCTIONS

Chessboard pattern (right half of the map)



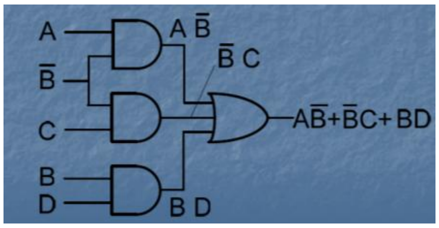
Pin counts:

AND/OR implementation (2 level):	11
AND/OR/XOR implementation (3 level):	8

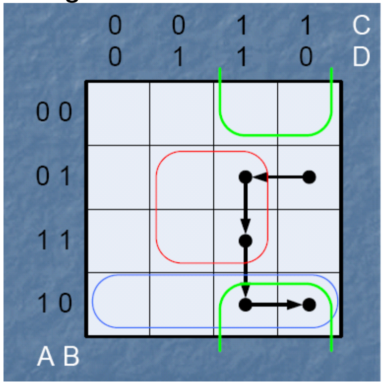
53

ANALYSIS OF COMBINATIONAL CIRCUITS

The change sequence of the input states and variables can be followed using the K map.



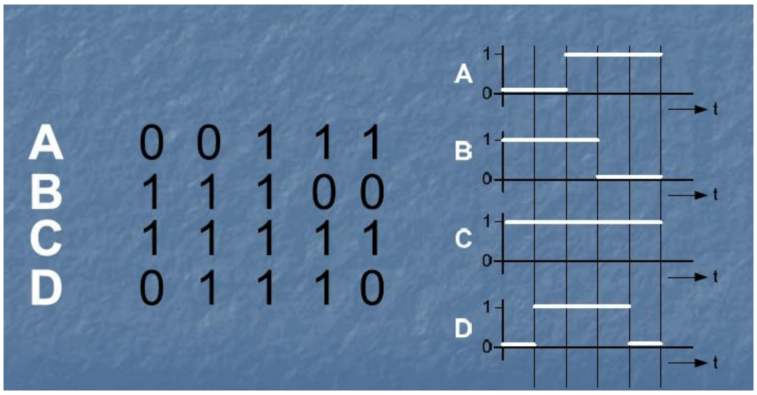
Loops in the K map corresponding to the AND gates.



A	B	C	D
0	1	1	0
0	1	1	1
1	1	1	1
1	0	1	1
1	0	1	0

Control handover between AND gates: static race hazards

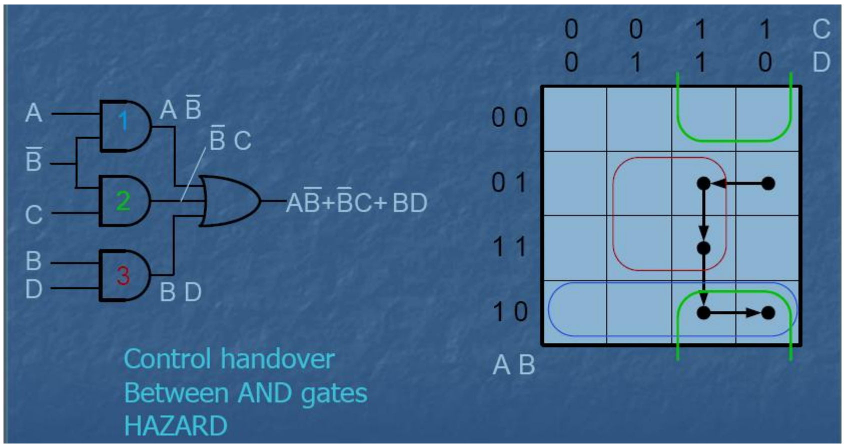
STATE SEQUENCE



The state sequence and the time diagram of input variables

55

CONTROL HANDOVER BETWEEN GATES: HAZARDS



Loops in the K map corresponding to the AND gates

56

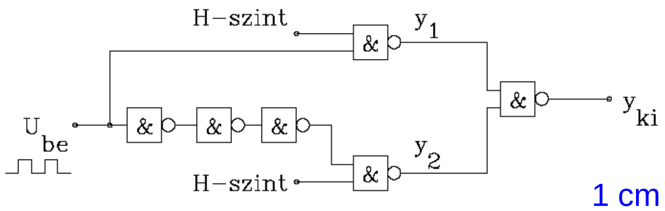
HAZARDS

A hazard or glitch is a fault in the logic system due to a change at the input. A static hazard is when the output of a logic circuit momentarily changes when its final value is the same as its value before the hazard (when the output is "trying" to remain the same, it jumps once, then settles down). A dynamic hazard (or oscillation hazard) is where a logic circuit will momentarily change back to its original value while changing to a new value.

The cause of hazards is the timing delay of different components in the circuit. The resulting glitches in the circuit may or may not induce additional problems - other than increased issues due to switching noise. It is good design practice to design circuits to minimize these hazards.

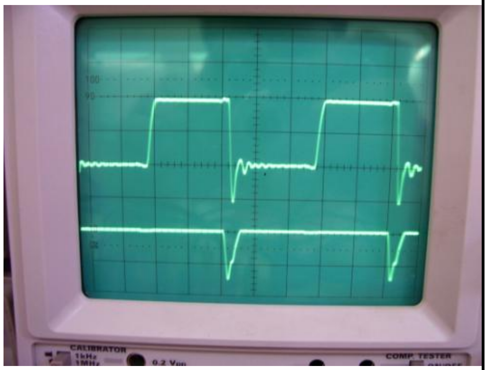
57

STATIC HAZARD: EXAMPLE

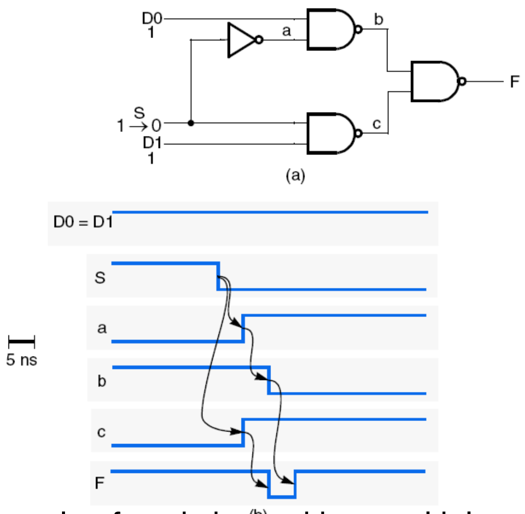


1 cm = 100 nsec

The brief pulse or glitch in the output is caused by the difference in the propagation delay of the signals through the gates.
Measured width at 50% level: app. 40 nsec, series 74 one gate average delay app. 13 nsec.
(student's measurement)



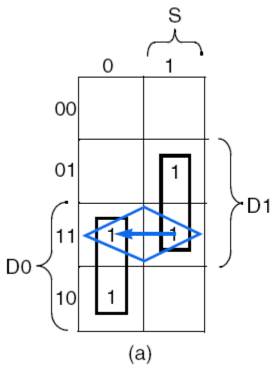
STATIC HAZARD IN A MULTIPLEXER

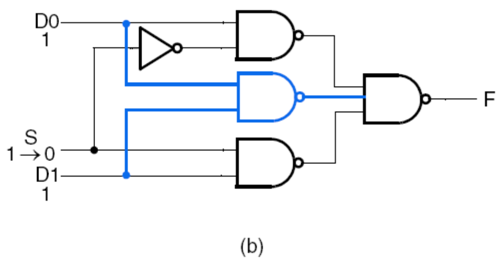


Example of static hazard in a multiplexer

59

STATIC HAZARD PREVENTION

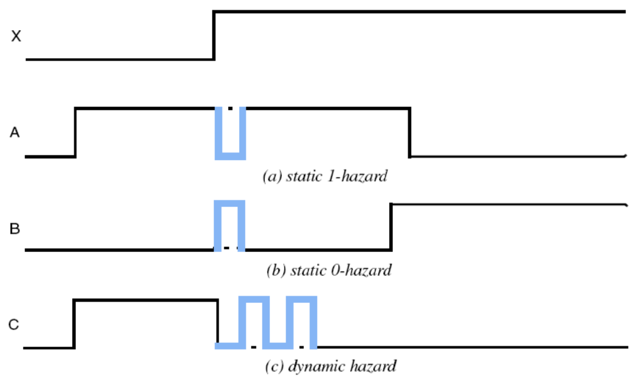




Multiplexer: Karnaugh map and static hazard prevention logic

60

STATIC HAZARDS



There are two types of static hazards: a low-going glitch (or static one hazard) is where the high output transitions to a low and back high (1-0-1) and a high-going glitch (or static zero hazard) is where the low output transitions to a high and back low (0-1-0).

61

STATIC HAZARDS

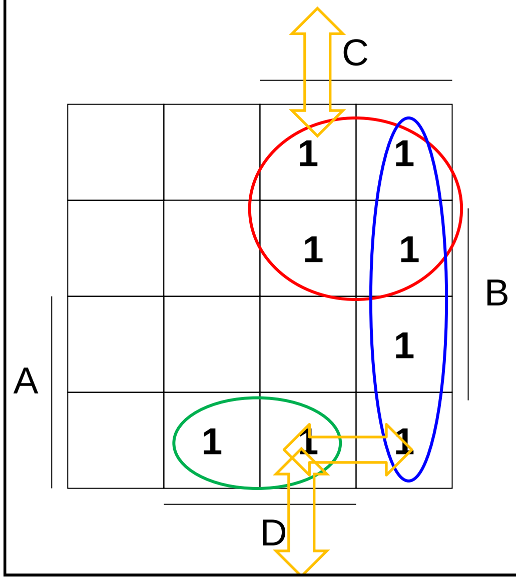
Static 0 hazards occur in product-of sums implementations, but do not occur in sum-of-products implementations. Static 1 hazards occur in sum-of-products implementations, but do not occur in product-of sums implementations.

Adding logic redundancy using a Karnaugh map is the easiest way to eliminate static hazards.

Static hazards can be eliminated using a sum-of-product s implementation containing every prime implicant.

62

STATIC HAZARDS

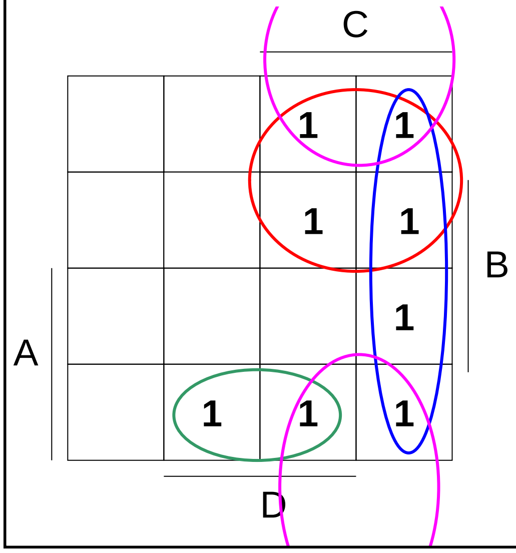


K maps are useful for detecting and eliminating statc hazards.

An additional term $\bar{B}C$ would eliminate the potential static hazards, bridging between the green and blue output state or blue and red output states.

63

ELIMINATION OF STATIC HAZARDS



An additional term $\bar{B}C$ eliminates the potential static hazards, bridging between the green and blue output state or blue and red output states. The term is redundant in terms of the logic state of the system, but such redundant terms are often needed to assure race-free dynamic performance.

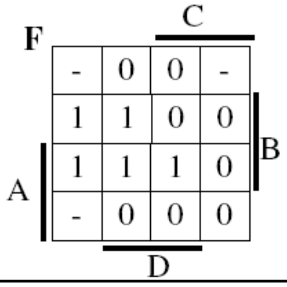
64

STATIC HAZARD IN NON-COMPLETELY DETERMINED LOGIC CIRCUITS

If the logic network to be designed is not completely determmined i.e. it contains don’t care terms than the way to find the simplest (minimal) two level hazard-free network is much less systematic.

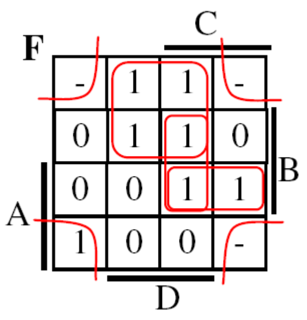
STATIC HAZARD IN NON-COMPLETELY DETERMINED LOGIC CIRCUITS

Given a logic function on the Karnaugh table (dash denotes don’t care terms).
1. Find and its simplest two-level hazard free disjunctive (SOP) realization. The input combinations corresponding to the don’t care terms cannot occur physically.
2. Find its simplest two-level hazard free disjunctive (SOP) realization, if the realized circuit cannot contain static hazard!

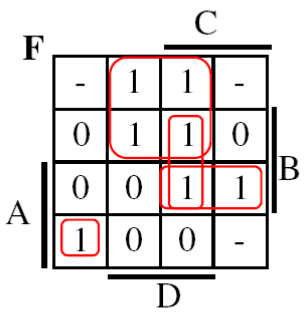


-	0	0	-
1	1	0	0
1	1	1	0
-	0	0	0

STATIC HAZARD IN NON-COMPLETELY DETERMINED LOGIC CIRCUITS



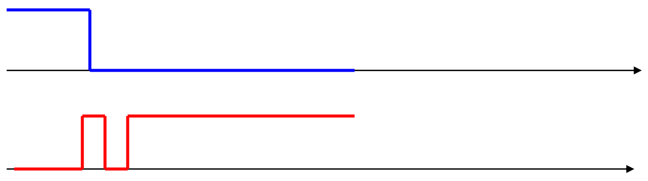
Solution case 1



Solution case 2

DYNAMIC HAZARD

A dynamic hazard (or oscillation hazard) is where a logic circuit will momentarily change back to its original value while changing to a new value.
In: 1⇒0, out 0 ⇒1, by logic
However out: 0 ⇒1 ⇒0 ⇒1).



68

DYNAMIC HAZARD

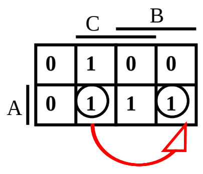
Dynamic hazards can only occur in three-level networks or above, if on any of the levels a static hazard is present.

Therefore eliminating the possible static hazards on each individual level, the dynamic hazard is also eliminated automatically.

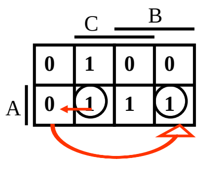
69

FUNCTIONAL HAZARD

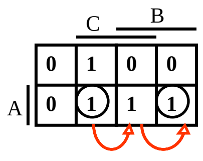
It can occur if two or more input variables change simultaneously.
E.g. for the transition 101⇒110 two different time sequence is possible, therefore on the output an unwanted 0 state can occur for a short time.



1⇒1



1⇒0⇒1



1⇒1⇒1

70

FREVISION QUESTIONS

1. Describe the Quine’s algebraic minimization method.

2. Describe the Quine-McCluskey algorithm.

3. Define and explain the following concepts: *static hazard*, *dynamic hazard* and *functional hazard*.

4. Describe and explain the method of eliminating dtatic hazards in a combinational circuit.

PROBLEMS AND EXERCISES

1. Given the three-variable logic function

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

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

2. 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. 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.

PROBLEMS AND EXERCISES

3. Design a logic circuit with two inputs, A and B, and two outputs, X and Y, so that it operates as follows:
(1) X and Y are both HIGH as long as A is HIGH, regardless of the level of B.
(2) If A pulses LOW, the LOW will appear at X if B=0 or at Y if B=1.

4. Implement the logic function shown below:
(a) using a two level minimal system (minimal cover),
(b) without static hazard.

$F(A,B,C,D,E) = \Sigma^5 (2,6,8,10,12,14,17,19,21,23,26,27,30,31)$

(HINT: the minimized SOP form contains five 4-cube (i.e. 3-variable) loops. Two additional product terms are necessary to eliminate the static race hazards.)
Supplementary exercise: Repeat the minimization using the Quine-McCluskey algorithm too.

73

END OF LECTURE

Appendix: 1. Quine-McCluskey demo
2. 6-variable Karnaugh map demo

74

QUINE-MCCLUSKEY DEMO

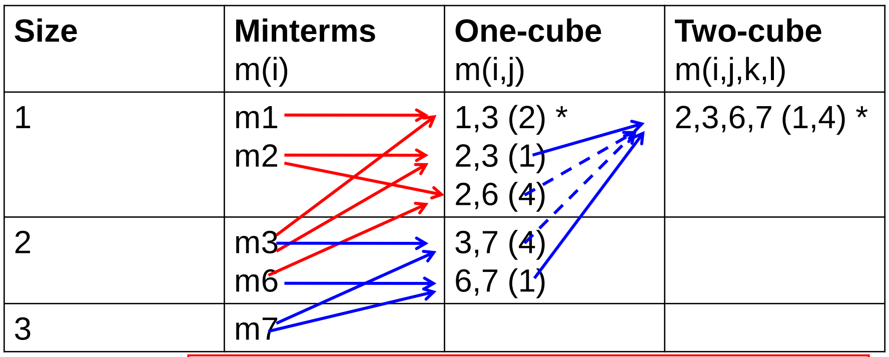
$F(A,B,C) = \Sigma^3(1,2,3,6,7)$

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

(1)	m1, m2
(2)	m3, m6
(3)	m7

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. The result is:

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

X 1 X 0 X 1

EXAMPLE: MINIMIZATION ON THE SIX-VARIABLE KARNAUGH MAP

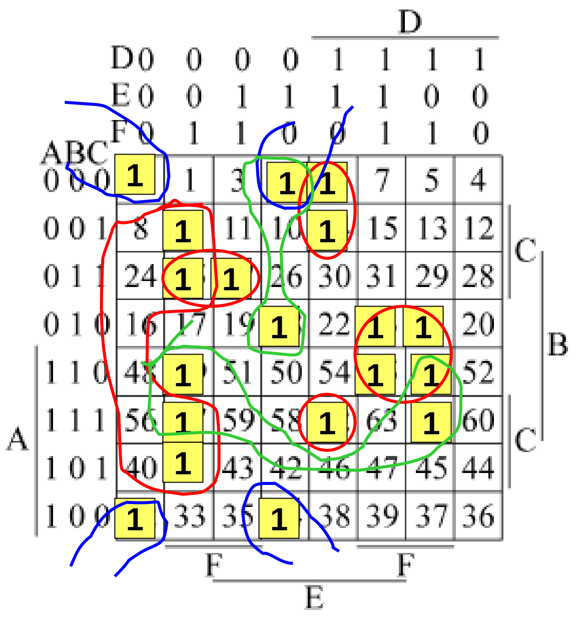
Function to be minimized (19 minterms):

$$F(A,B,C,D,E,F) =$$

$$\Sigma^6 (0,2,6,9,14,18,21,23,25,27,32,34,41,49,53,55,57,61,62)$$

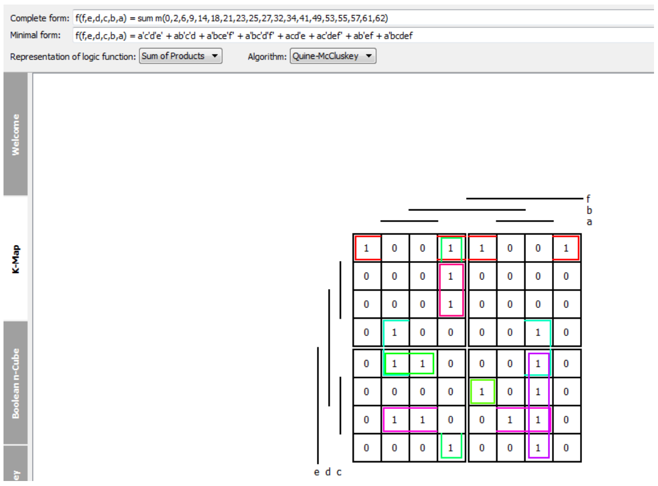
78

EXAMPLE: LOOPING FOR SIX VARIABLES



ABC \ DEF	000	001	011	010	110	111	101	100
000	1	1	3	1	1	7	5	4
001	8	1	11	10	1	15	13	12
011	24	1	1	26	30	31	29	28
010	16	17	19	1	22	1	1	20
110	48	1	51	50	54	1	1	52
111	56	1	59	58	1	63	1	60
101	40	1	43	42	46	47	45	44
100	1	33	35	1	38	39	37	36

EXAMPLE: LOOPING FOR SIX VARIABLES


Complete form: f(f,e,d,c,b,a) = sum m(0,2,6,9,14,18,21,23,25,27,32,34,41,49,53,55,57,61,62)

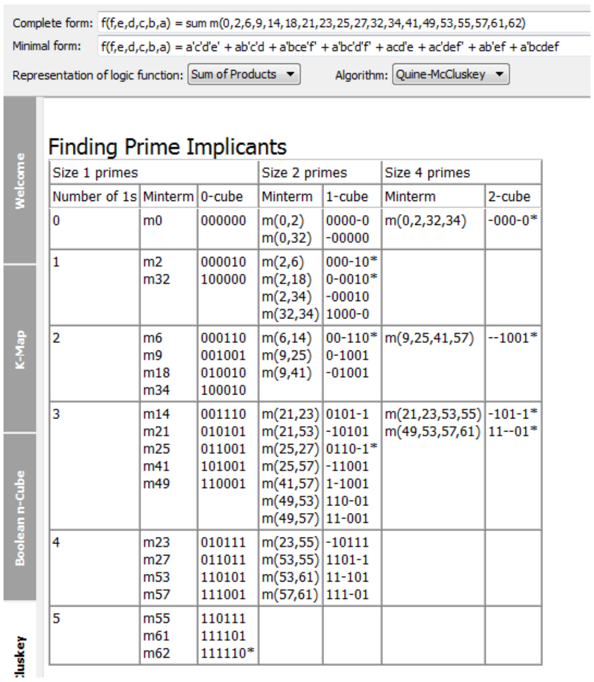
Minimal form: f(f,e,d,c,b,a) = a'c'd'e' + ab'c'd + a'bce'f' + a'bc'd'f' + acd'e + ac'def' + ab'ef + a'bcdef

Representation of logic function: Sum of Products

Algorithm: Quine-McCluskey

1	0	0	1	1	0	0	1
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	1	0	0	0	0	1	0
0	1	1	0	0	0	1	0
0	0	0	0	1	0	1	0
0	1	1	0	0	1	1	0
0	0	0	1	0	0	1	0

Bmin Boolean minimizer



Complete form: f(f,e,d,c,b,a) = sum m(0,2,6,9,14,18,21,23,25,27,32,34,41,49,53,55,57,61,62)

Minimal form: f(f,e,d,c,b,a) = a'c'd'e' + ab'c'd + a'bce'f' + a'bc'd'f' + acd'e + ac'def' + ab'ef + a'bcdef

Representation of logic function: Sum of Products

Algorithm: Quine-McCluskey

Finding Prime Implicants

Size 1 primes			Size 2 primes		Size 4 primes	
Number of 1s	Minterm	0-cube	Minterm	1-cube	Minterm	2-cube

0	m0	000000	m(0,2)
			m(0,32)
				0000-0
				-00000
					m(0,2,32,34)	-000-0*

1	m2
	m32
		000010
		100000
			m(2,6)
			m(2,18)
			m(2,34)
			m(32,34)
				000-10*
				0-0010*
				-00010
				1000-0		

2	m6
	m9
	m18
	m34
		000110
		001001
		010010
		100010
			m(6,14)
			m(9,25)
			m(9,41)
				00-110*
				0-1001
				-01001
					m(9,25,41,57)	--1001*

3	m14
	m21
	m25
	m41
	m49
		001110
		010101
		011001
		101001
		110001
			m(21,23)
			m(21,53)
			m(25,27)
			m(25,57)
			m(41,57)
			m(49,53)
			m(49,57)
				0101-1
				-10101
				0110-1*
				-11001
				1-1001
				110-01
				11-001
					m(21,23,53,55)
					m(49,53,57,61)
						-101-1*
						11--01*

4	m23
	m27
	m53
	m57
		010111
		011011
		110101
		111001
			m(23,55)
			m(53,55)
			m(53,61)
			m(57,61)
				-10111
				1101-1
				11-101
				111-01		

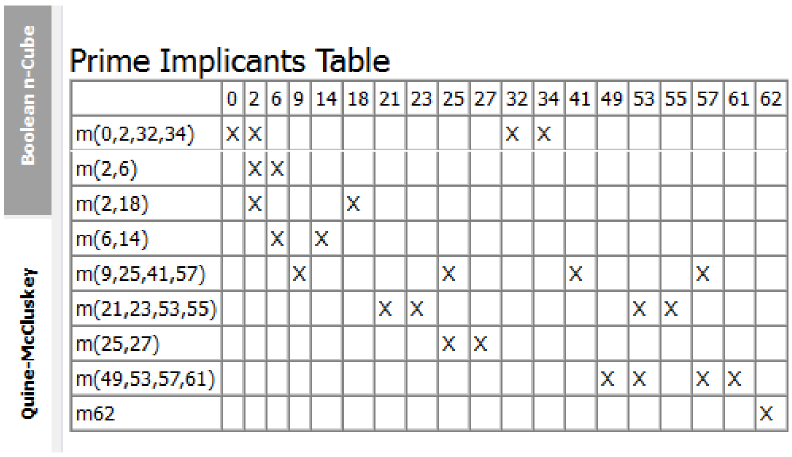
5	m55 m61 m62	110111 111101 111110*				

Bmin

Quine-McCluskey minimization

81

PRIME IMPLICANT TABLE



Prime Implicants Table

	0	2	6	9	14	18	21	23	25	27	32	34	41	49	53	55	57	61	62
m(0,2,32,34)	X	X									X	X							
m(2,6)		X	X																
m(2,18)		X				X													
m(6,14)			X		X														
m(9,25,41,57)				X					X				X				X		
m(21,23,53,55)							X	X							X	X			
m(25,27)									X	X									
m(49,53,57,61)														X	X		X	X	
m62																			X

Bmin

82