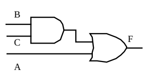
Gate Logic: Two Level Canonical Forms

Contemporary Logic Design
Two-Level Logic

Sum of Products

A	B	C	Minterms
0	0	0	$\overline{A}\,\overline{B}\,\overline{C} = m_0$
0	0	1	$\overline{A}\,\overline{B}\,C = m_1$
0	1	0	$\overline{A}\,B\,\overline{C} = m_2$
0	1	1	$\overline{A}\,B\,C = m_3$
1	0	0	$A\,\overline{B}\,\overline{C} = m_4$
1	0	1	$A\,\overline{B}\,C = m_5$
1	1	0	$A\,B\,\overline{C} = m_6$
1	1	1	$A\,B\,C = m_7$

Shorthand Notation for Minterms of 3 Variables



2-Level AND/OR Realization

product term / minterm:

ANDed product of literals in which each variable appears exactly once, in true or complemented form (but not both!)

F in canonical form:

F(A,B,C) = Σm(3,4,5,6,7)
= m3 + m4 + m5 + m6 + m7
= A' B C + A B' C' + A B' C + A B C' + A B C

canonical form/minimal form

F = A B' (C + C') + A' B C + A B (C' + C)
= A B' + A' B C + A B
= A (B' + B) + A' B C
= A + A' B C
= A + B C

F = (A + B C)' = A' (B' + C') = A' B' + A' C'

Gate Logic: 2 Level Canonical Forms

Contemporary Logic Design
Two-Level Logic

Product of Sums / Conjunctive Normal Form / Maxterm Expansion

A	B	C	Maxterms
0	0	0	$A + B + C = M_0$
0	0	1	$A + B + \overline{C} = M_1$
0	1	0	$A + \overline{B} + C = M_2$
0	1	1	$A + \overline{B} + \overline{C} = M_3$
1	0	0	$\overline{A} + B + C = M_4$
1	0	1	$\overline{A} + B + \overline{C} = M_5$
1	1	0	$\overline{A} + \overline{B} + C = M_6$
1	1	1	$\overline{A} + \overline{B} + \overline{C} = M_7$

Maxterm Shorthand Notation for a Function of Three Variables

Maxterm:

ORed sum of literals in which each variable appears exactly once in either true or complemented form, but not both!

Maxterm form:

Find truth table rows where F is 0
0 in input column implies true literal
1 in input column implies complemented literal

F(A,B,C) = ΠM(0,1,2)
= (A + B + C) (A + B + C') (A + B' + C)

F'(A,B,C) = ΠM(3,4,5,6,7)
= (A + B' + C') (A' + B + C) (A' + B + C') (A' + B' + C) (A' + B' + C')

Gate Logic: Two Level Canonical Forms

Contemporary Logic Design
Two-Level Logic

Sum of Products, Products of Sums, and DeMorgan's Law

F' = A' B' C' + A' B' C + A' B C'

Apply DeMorgan's Law to obtain F:

(F')' = (A' B' C' + A' B' C + A' B C')'

F = (A + B + C) (A + B + C') (A + B' + C)

F' = (A + B' + C') (A' + B + C) (A' + B + C') (A' + B' + C) (A' + B' + C')

Apply DeMorgan's Law to obtain F:

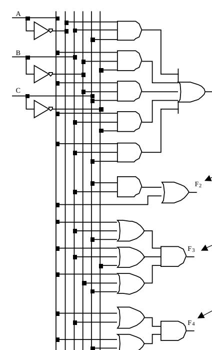
(F')' = {(A + B' + C') (A' + B + C) (A' + B + C') (A' + B' + C) (A' + B' + C')}'

F = A' B C + A B' C' + A B' C + A B C' + A B C

Gate Logic: Two-Level Canonical Forms

Contemporary Logic Design
Two-Level Logic

Four Alternative Implementations of F:



Canonical Sum of Products

Minimized Sum of Products

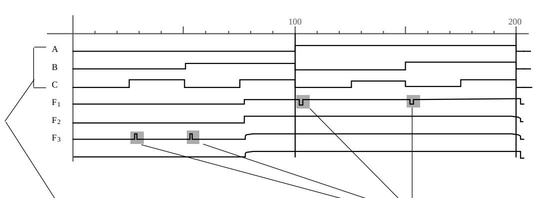
Canonical Products of Sums

Minimized Products of Sums

Gate Logic: Two-Level Canonical Forms

Contemporary Logic Design
Two-Level Logic

Waveform Verification of the Three Alternatives



Eight Unique Combinations of Three Inputs

Except for *timing glitches*, output waveforms of the three implementations are essentially identical

Gate Logic: Two-Level Simplification

Contemporary Logic Design
Two-Level Logic

Algebraic Simplification:

not an algorithm/systematic procedure

how do you know when the minimum realization has been found?

Computer-Aided Tools:

precise solutions require very long computation times, especially for functions with many inputs (>10)

heuristic methods employed — "educated guesses" to reduce the amount of computation good solutions not best solutions

Still Relevant to Learn Hand Methods:

insights into how the CAD programs work, and their strengths and weaknesses

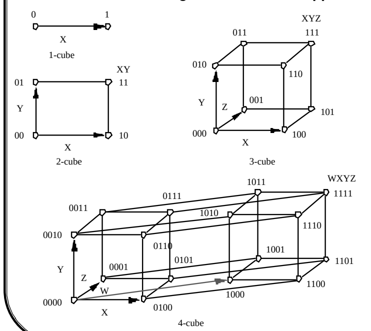
ability to check the results, at least on small examples

don't have computer terminals during exams

Gate Logic: Two-Level Simplification

Boolean Cubes

Visual technique for identifying when the Uniting Theorem can be applied

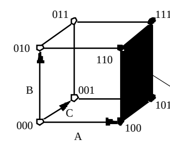


Just another way to represent the truth table

n input variables = n dimensional "cube"

Gate Logic: Two-Level Simplification

Subcubes of Higher Dimensions than 2



F(A,B,C) = Σm(4,5,6,7)

On-set forms a rectangle, i.e., a cube of two dimensions

represents an expression in one variable i.e., 3 dimensions - 2 dimensions

A is asserted and unchanged
B and C vary

This subcube represents the literal A

Gate Logic: Two-Level Simplification

In a 3-cube:

a 0-cube, i.e., a single node, yields a term in three literals

a 1-cube, i.e., a line of two nodes, yields a term in two literals

a 2-cube, i.e., a plane of four nodes, yields a term in one literal

a 3-cube, i.e., a cube of eight nodes, yields a constant term "1"

In general,

an m-subcube within an n-cube (m < n) yields a term with n - m literals

Gate Logic: Two-Level Simplification

Karnaugh Map Method

hard to draw cubes of more than 4 dimensions

K-map is an alternative method of representing the truth table that helps visualize adjacencies in up to 6 dimensions

Beyond that, computer-based methods are needed

2-variable K-map

B \ A	0	1
0	0	2
1	1	3

3-variable K-map

C \ AB	00	01	11	10
0	0	2	6	4
1	1	3	7	5

4-variable K-map

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

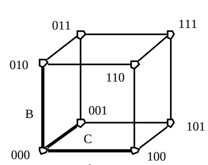
***Numbering Scheme:* 00, 01, 11, 10**
Gray Code — only a single bit changes from code word to next code word

Gate Logic: Two-Level Simplification

Karnaugh Map Method

Adjacencies in the K-Map

C \ AB	00	01	11	10
0	000	010	110	100
1	001	011	111	101



Wrap from first to last column

Top row to bottom row

Gate Logic: Two-Level Simplification

K-map Method Examples: 4 variables

CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	1	1	1	1
10	1	1	1	1

F(A,B,C,D) = Σm(0,2,3,5,6,7,8,10,11,14,15)

F =

Gate Logic: Two-Level Simplification

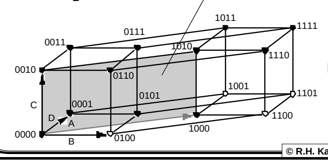
K-map Method Examples: 4 variables

CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	1	1	1	1
10	1	1	1	1

F(A,B,C,D) = Σm(0,2,3,5,6,7,8,10,11,14,15)

F = C + A' B D + B' D'

Find the smallest number of the largest possible subcubes that cover the ON-set



K-map Corner Adjacency Illustrated in the 4-Cube

Gate Logic: Two-Level Simplification

K-map Method: Circling Zeros

CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	1	1	1	1
10	1	1	1	1

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

Replace F by $\overline{F}$, 0's become 1's and vice versa

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

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

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

Gate Logic: Two-Level Simplification

K-map Example: Don't Cares

Don't Cares can be treated as 1's or 0's if it is advantageous to do so

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	1	1	0	0
10	0	X	0	0

F(A,B,C,D) = Σm(1,3,5,7,9) + Σd(6,12,13)

F = w/o don't cares

F = w/ don't cares

Gate Logic: Two-Level Simplification

K-map Example: Don't Cares

Don't Cares can be treated as 1's or 0's if it is advantageous to do so

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	1	1	0	0
10	0	X	0	0

F(A,B,C,D) = Σm(1,3,5,7,9) + Σd(6,12,13)

F = A'D + B' C' D w/o don't cares

F = C' D + A' D w/ don't cares

By treating this DC as a "1", a 2-cube can be formed rather than one 0-cube

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	1	1	0	0
10	0	X	0	0

In PoS form: F = D (A' + C')

Same answer as above, but fewer literals

Gate Logic: Two Level Simplification

Definition of Terms

***implicant*: single element of the ON-set or any group of elements that can be combined together in a K-map**

***prime implicant*: implicant that cannot be combined with another implicant to eliminate a term**

***essential prime implicant*: if an element of the ON-set is covered by a single prime implicant, it is an essential prime**

Objective:

grow implicants into prime implicants

cover the ON-set with as few prime implicants as possible

essential primes participate in ALL possible covers

Gate Logic: Two Level Simplification

Examples to Illustrate Terms

CD \ AB	00	01	11	10
00	0	1	1	0
01	1	1	1	0
11	1	0	1	1
10	0	0	1	1

6 Prime Implicants:

A' B' D, B C', A C, A' C' D, A B, B' C D

essential

Minimum cover = B C' + A C + A' B' D

CD \ AB	00	01	11	10
00	0	0	1	0
01	1	1	1	0
11	0	1	1	1
10	0	1	0	0

5 Prime Implicants:

B D, A B C', A C D, A' B C, A' C' D

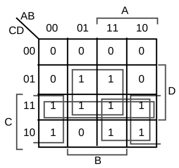
essential

Essential implicants form minimum cover

Gate Logic: Two Level Simplification

Contemporary Logic Design
Two-Level Logic

More Examples



Prime Implicants:

B D, C D, A C, B' C

essential

Essential primes form the minimum cover

Gate Logic: Two-Level Simplification

Contemporary Logic Design
Two-Level Logic

Algorithm: Minimum Sum of Products Expression from a K-Map

Step 1: **Choose an element of ON-set not already covered by an implicant**

Step 2: **Find "maximal" groupings of 1's and X's adjacent to that element. Remember to consider top/bottom row, left/right column, and corner adjacencies. This forms *prime implicants* (always a power of 2 number of elements).**

Repeat Steps 1 and 2 to find all prime implicants

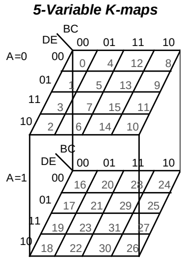
Step 3: **Revisit the 1's elements in the K-map. If covered by single prime implicant, it is *essential*, and participates in final cover. The 1's it covers do not need to be revisited**

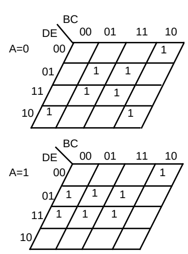
Step 4: **If there remain 1's not covered by essential prime implicants, then select the smallest number of prime implicants that cover the remaining 1's**

Gate Logic: Two-Level Simplification

Contemporary Logic Design
Two-Level Logic

5-Variable K-maps





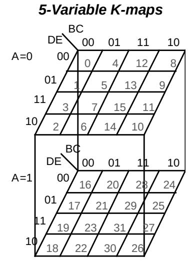
f(A,B,C,D,E) = Σm(2,5,7,8,10, 13,15,17,19,21,23,24,29 31)

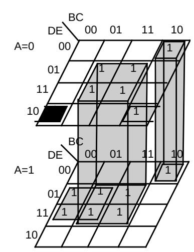
=

Gate Logic: Two-Level Simplification

Contemporary Logic Design
Two-Level Logic

5-Variable K-maps





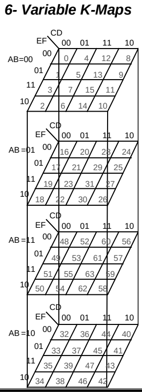
f(A,B,C,D,E) = Σm(2,5,7,8,10, 13,15,17,19,21,23,24,29 31)

= C E + A B' E + B C' D' E' + A' C' D E'

Gate Logic: Two Level Simplification

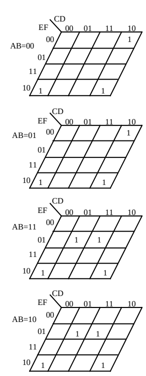
Contemporary Logic Design
Two-Level Logic

6- Variable K-Maps



f(A,B,C,D,E,F) = Σm(2,8,10,18,24, 26,34,37,42,45,50, 53,58,61)

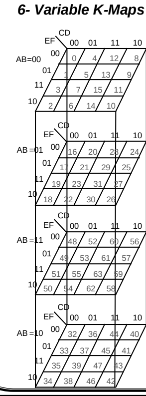
=



Gate Logic: Two Level Simplification

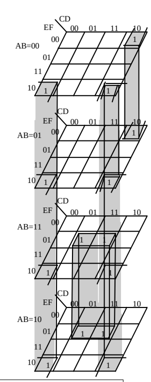
Contemporary Logic Design
Two-Level Logic

6- Variable K-Maps



f(A,B,C,D,E,F) = Σm(2,8,10,18,24, 26,34,37,42,45,50, 53,58,61)

= D' E F' + A D E' F + A' C D' F'



Gate Logic: CAD Tools for Simplification

Espresso Method: Overview

1. Expands implicants to their maximum size
   Implicants covered by an expanded implicant are removed from further consideration
   Quality of result depends on order of implicant expansion
   Heuristic methods used to determine order
   Step is called EXPAND
2. Irredundant cover (i.e., no proper subset is also a cover) is extracted from the expanded primes
   Just like the Quine-McCluskey Prime Implicant Chart
   Step is called IRREDUNDANT COVER
3. Solution usually pretty good, but sometimes can be improved
   Might exist another cover with fewer terms or fewer literals
   Shrink prime implicants to smallest size that still covers ON-set
   Step is called REDUCE
4. Repeat sequence REDUCE/EXPAND/IRREDUNDANT COVER to find alternative prime implicants
   Keep doing this as long as new covers improve on last solution
5. A number of optimizations are tried, e.g., identify and remove essential primes early in the process

Gate Logic: CAD Tools for Simplification

Espresso Inputs and Outputs

$f(A,B,C,D) = m(4,5,6,8,9,10,13) + d(0,7,15)$

Espresso Input

```
.i 4          -- # inputs
.o 1          -- # outputs
.ilb a b c d  -- input names
.ob f         -- output name
.p 10         -- number of product terms
0100  1       -- A'BC'D'
0101  1       -- A'BC'D
0110  1       -- A'BCD'
1000  1       -- AB'C'D'
1001  1       -- AB'C'D
1010  1       -- AB'CD'
1101  1       -- ABC'D
0000  -       -- A'B'C'D' don't care
0111  -       -- A'BCD don't care
1111  -       -- ABCD don't care
.e            -- end of list
```

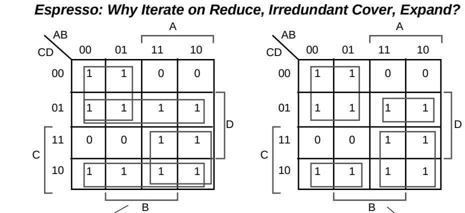
Espresso Output

```
.i 4
.o 1
.ilb a b c d
.ob f
.p 3
1-01  1
10-0  1
01--  1
.e
```

$f = A\,C'\,D + A\,B'\,D' + A'\,B$

Gate Logic: CAD Tools for Simplification

Espresso: Why Iterate on Reduce, Irredundant Cover, Expand?



Initial Set of Primes found by Steps1 and 2 of the Espresso Method

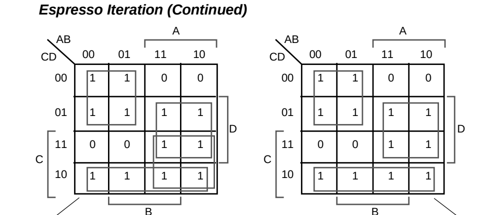
4 primes, irredundant cover, but not a minimal cover!

Result of REDUCE: Shrink primes while still covering the ON-set

Choice of order in which to perform shrink is important

Gate Logic: CAD Tools for Simplification

Espresso Iteration (Continued)



Second EXPAND generates a different set of prime implicants

IRREDUNDANT COVER found by final step of espresso

Only three prime implicants!

Two-Level Logic: Summary

Primitive logic building blocks
INVERTER, AND, OR, NAND, NOR, XOR, XNOR

Canonical Forms
Sum of Products, Products of Sums

Incompletely specified functions/don't cares

Logic Minimization
Goal: two-level logic realizations with fewest gates and fewest number of gate inputs

Obtained via Laws and Theorems of Boolean Algebra

or Boolean Cubes and the Uniting Theorem

or K-map Methods up to 6 variables

or Quine-McCluskey Algorithm

or Espresso CAD Tool