BOOLEAN ALGEBRA: PART 2

- Optimization: principles, KMap
- Incompletely defined Boolean function

BOOLEAN EXPRESSION SIMPLICITY: INTUITION

$$m_0 + m_1 + m_4 + m_5 = F(a,b,c) = \sum m(0,1,4,5)$$

	abc	F(a,b,c)
0	000	1
1	001	1
2	010	0
3	011	0
4	100	1
5	101	1
6	110	0
7	111	0

F(a,b,c) = a'b'c' + a'b'c + ab'c' + ab'c (EXPR 1)
= a'b' (c'+c) + ab' (c'+c)
= a'b' + ab' (EXPR 2)
= (a'+a) b'
= b' (EXPR 3)

EXPR 3 seems to be much simpler than **(EXPR 2)** or even **(EXPR 1)**

K MAPS 2 VARIABLES (1)

Regular Truth table
for 2 variables

00	
01	
10	
11	
ab	

K Map for 2 variables:
swap of the last 2 rows

00	
01	
11	
10	
ab	

When moving from one row to the next, only one variable changes.
Two contiguous cells correspond to two adjacent complete minterms

2 VARIABLES K MAPS: A FEW INTERESTING FUNCTIONS

ab	
00	1
01	1
11	
10	

a’b’ + a’b = a’

ab	
00	
01	1
11	1
10	

a’b + ab = b

ab	
00	
01	
11	1
10	1

ab + ab’ = a

ab	
00	1
01	
11	
10	1

a’b’ + ab’ = b’

All of the blocks of two adjacent cells (tiles) correspond to 1 variable « product ». The first and last rows are adjacent

ab	
00	1
01	1
11	1
10	

F(a,b) = a'b' + a'b + ab
Canonical Form
For F

ab	
00	1
01	1
11	1
10	

F(a,b) = a' + b
Minimal SOP for F

PRINCIPLE: try to cover Support of F with blocks of 1 cell or better with blocks of 2 adjacent cells (called tiles).

- Use the largest possible tiles
- OK if a same cell is covered several times
- All of the tiles used need to be covered by Support of F

3 VARIABLES K MAP: INTERESTING 2 CELLS TILES

ab \ c	0	1
00	1	1
01		
11		
10		

a’b’

ab \ c	0	1
00		
01	1	1
11		
10		

a’b

ab \ c	0	1
00		
01		
11	1	1
10		

ab

ab \ c	0	1
00		
01		
11		
10	1	1

ab’

ab \ c	0	1
00	1	
01	1	
11		
10		

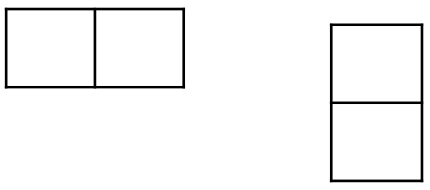
a’c’

ab \ c	0	1
00		
01		1
11		1
10		

bc

….

All of the blocks (tiles) consisting of two adjacent cells having one of the shapes below correspond to a 2 variables product:



3 VARIABLES K MAP: INTERESTING 4 CELLS TILES

ab \ c	0	1
00	1	1
01	1	1
11		
10		

a'

ab \ c	0	1
00		
01	1	1
11	1	1
10		

b

ab \ c	0	1
00		
01		
11	1	1
10	1	1

a

ab \ c	0	1
00	1	1
01		
11		
10	1	1

b'

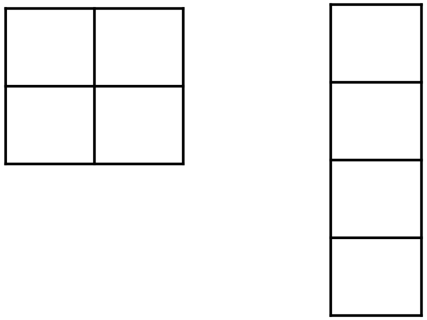
ab \ c	0	1
00	1	
01	1	
11	1	
10	1	

c'

ab \ c	0	1
00		1
01		1
11		1
10		1

c

All of the tiles consisting of 4 adjacent cells and having one of the shapes below, correspond to a 1 variable product:



USING 3 VARIABLES K MAPS (1)

ab \ c	0	1
00	1	1
01	1	1
11	1	
10	1	

F(a,b,c) = a’ + c’
Minimal SOP for F

ab \ c	0	1
00	1	
01	1	1
11	1	
10		

F(a,b,) = a’b + a’c’ + bc’
Minimal SOP for F

PRINCIPLE: try to cover Support of F with blocks of 1 cells or better blocks of 2 or 4 adjacent cells (called tiles).

- Use the largest possible tiles (the larger the tiles are, the simpler the corresponding arithmetic expressions).
- OK if a same cell is covered several times
- All of the tiles used need to be covered by Support of F

USING 3 VARIABLES K MAPS (2)

ab \ c	0	1
00	1	
01	1	
11		
10		1

F(a,b,c) = ab’c + a’c’
Minimal SOP for F

ab \ c	0	1
00	1	
01	1	1
11		
10		1

F(a,b,) = a’b + a’c’ + ab’c
Minimal SOP for F

If necessary, blocks of 1 cell have to be used (see above).

4 VARIABLES K MAP

	00	01	11	10	cd
00					
01					
11					
10					
ab					

Attention: first and last row are adjacent and first and last columns are adjacent too.

A standard truth table would be an array with 16 rows. Here the K Map is an array of 4 rows x 4 columns.

Again, when moving from one row (resp. column) to the next one. Only one variable is altered.

ab \ cd	00	01	11	10
00				
01		1		
11		1		
10				

bc’d

ab \ cd	00	01	11	10
00				
01				
11				1
10				1

acd’

ab \ cd	00	01	11	10
00	1			
01				
11				
10	1			

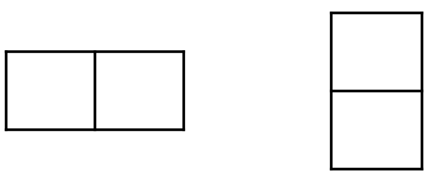
b’c’d’

ab \ cd	00	01	11	10
00				
01	1			1
11				
10				

…

a’bd’

All of the tiles consisting of two adjacent cells and having one of the shapes below, correspond to a 3 variables product:



4 VARIABLES K MAP: INTERESTING 4 CELLS TILES

ab \ cd	00	01	11	10
00	1			
01	1			
11	1			
10	1			

c’d’

ab \ cd	00	01	11	10
00				
01				
11	1			1
10	1			1

ad’

ab \ cd	00	01	11	10
00	1			1
01				
11				
10	1			1

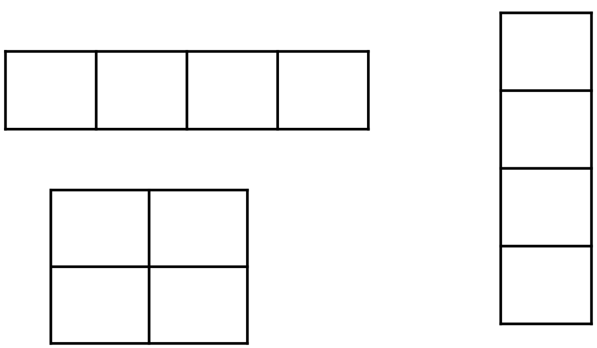
b’d’: 4 corners

ab \ cd	00	01	11	10
00				
01	1	1	1	1
11				
10				

……

a’bd’

All of the tiles consisting of 4 adjacent cells and having one of the shapes below, correspond to a 2 variables product:



ab \ cd	00	01	11	10
00			1	1
01			1	1
11			1	1
10			1	1

c

ab \ cd	00	01	11	10
00	1			1
01	1			1
11	1			1
10	1			1

d’

ab \ cd	00	01	11	10
00	1	1	1	1
01				
11				
10	1	1	1	1

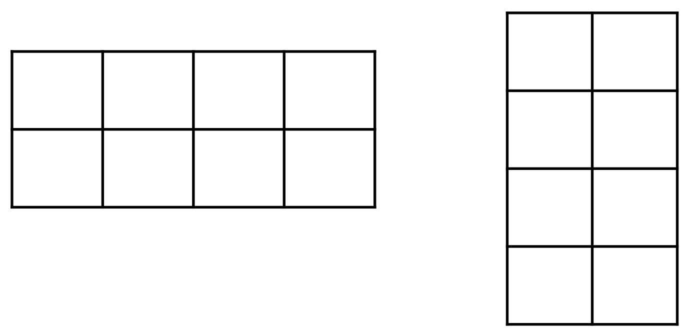
b’

ab \ cd	00	01	11	10
00				
01	1	1	1	1
11	1	1	1	1
10				

b

……

All of the tiles consisting of 8 adjacent cells and having one of the shapes below correspond to a 1 variable product:



USING 4 VARIABLES K MAPS (1)

	00	01	11	10	*cd*
00				1	
01		1	1	1	
11				1	
10				1	
ab					

F(a,b,) = cd’ + a’bd
Minimal SOP for F

PRINCIPLE: try to cover Support of F with blocks of 1 cell or better blocks of 2, 4 and 8 adjacent cells (called tiles).

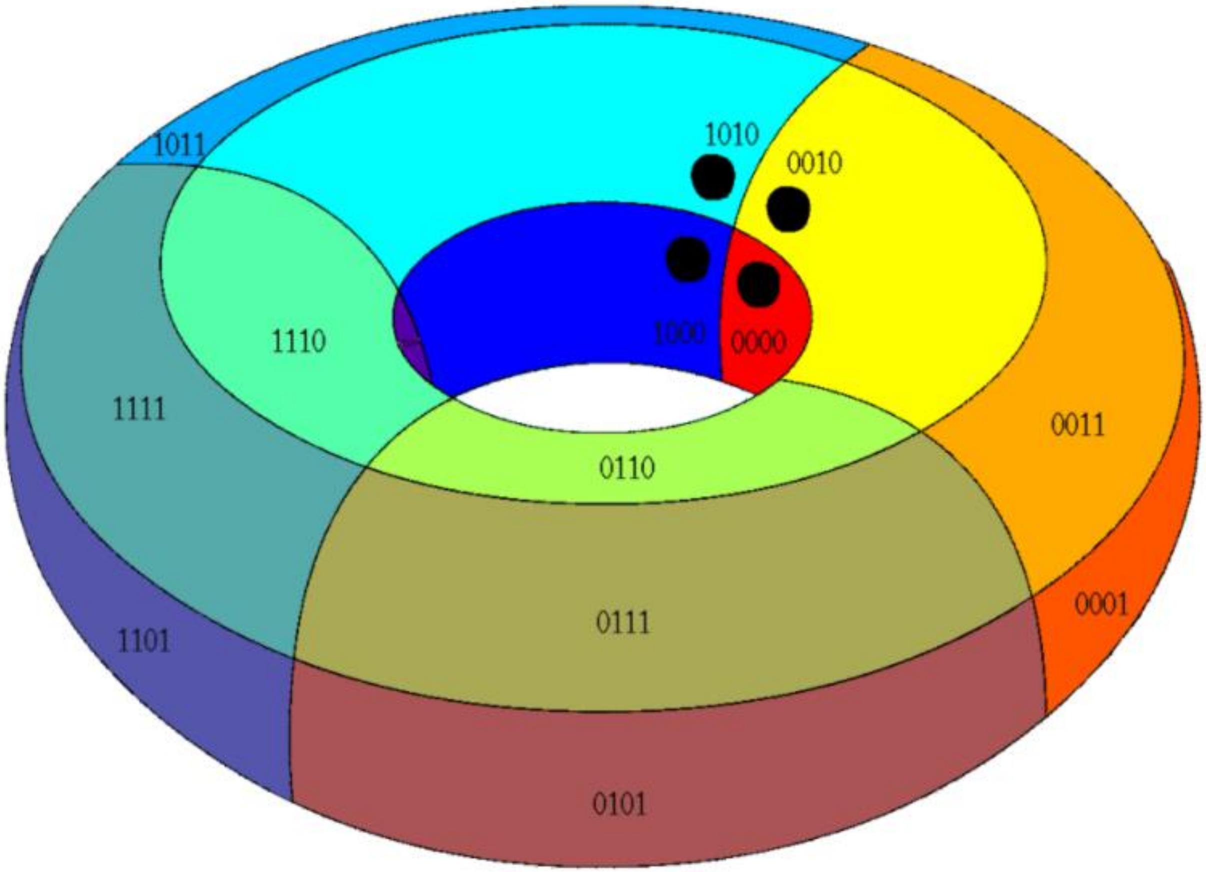
- Use the largest possible tiles (the larger the tiles are, the simpler the corresponding arithmetic expressions).
- OK if a same cell is covered several times
- All of the tiles need to be covered by Support of F

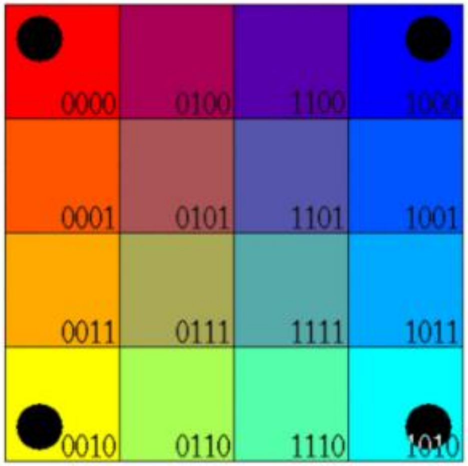
USING 4 VARIABLES K MAPS (2)

ab \ cd	00	01	11	10
00				
01	1	1	1	1
11				
10				1

F(a,b,) = a’b + ab’cd’
Minimal SOP for F

4 VARIABLES KMAP ON A TORUS



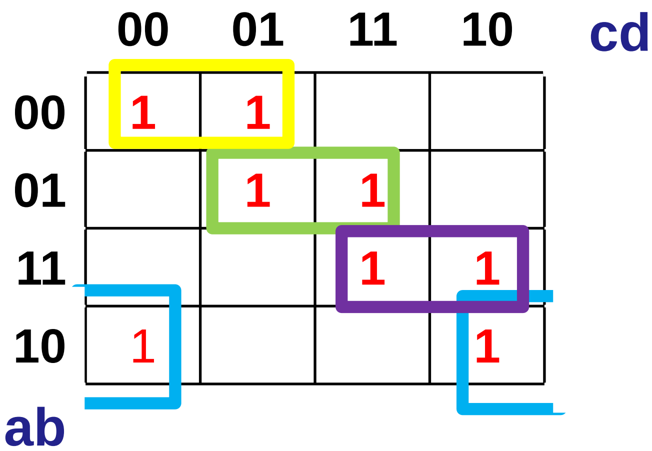


SOURCE OF THE DRAWINGS : WIKIPEDIA

ab \ cd	00	01	11	10
00	1	1		
01		1	1	
11			1	1
10	1			1

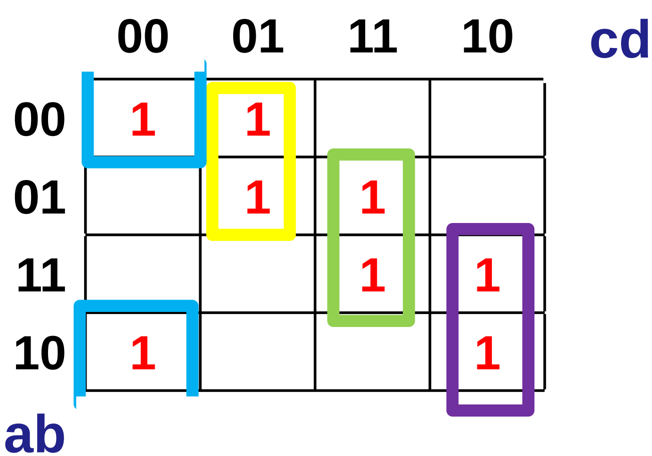
$$F(a,b,c,d) = a'b'c' + a'bd + abc + ab'd'$$

MINIMAL SOP 1



F(a,b,c,d) = a'b'c' + a'bd + abc + ab'd'

MINIMAL SOP 1



F(a,b,c,d) = a'c'd + bcd + acd' + b'c'd'

MINIMAL SOP 2

K MAP FOR 5 VARIABLES

PRINCIPLE: concatenate two 4 variables K Maps: the first one direct and the second one mirrored to respect the principle moving from one column to the next gets only one variable changed

	e = 0				*e = 1*					
	00	**01**	**11**	**10**	**10**	**11**	**01**	**00**	***cd***	
00		1					1			a’b’c’d
01										
11		1	1			1	1			abd
10										
ab										

DIFFICULTY: adjacency can be difficult to uncover. Cells symmetric with respect to the orange vertical line are adjacent. Cells with 1 in blue (resp. green are adjacent).

K MAP FOR 6 VARIABLES

PRINCIPLE: concatenate four 4 variables K Map: the first one direct and the others one mirrored to respect the principle moving from one column (resp. row) to the next gets only one variable changed

		e = 0				***e = 1***				
		00	**01**	**11**	**10**	**10**	**11**	**01**	**00**	***cd***
f = 0	**00**									
	01									
	11		1					1		
	10									
f = 1	**10**									
	11		1					1		
	01									
	00									
	ab									

The cells with 1's in green are adjacent!! They correspond to: **abc'd**

FIRST PART OF THE TRUTH TABLE

a_3	a_2	a_1	a_0	A	B	b_3	b_2	b_1	b_0
0	0	0	0	0	1	0	0	0	1
0	0	0	1	1	2	0	0	1	0
0	0	1	0	2	3	0	0	1	1
0	0	1	1	3	4	0	1	0	0
0	1	0	0	4	5	0	1	0	1
0	1	0	1	5	6	0	1	1	0
0	1	1	0	6	7	0	1	1	1
0	1	1	1	7	8	1	0	0	0
1	0	0	0	8	9	1	0	0	1
1	0	0	1	9	0	0	0	0	0

SECOND PART OF THE TRUTH TABLE

a_3	a_2	a_1	a_0	A	B	b_3	b_2	b_1	b_0
1	0	1	0	X	X	d	d	d	d
1	0	1	1	X	X	d	d	d	d
1	1	0	0	X	X	d	d	d	d
1	1	0	1	X	X	d	d	d	d
1	1	1	0	X	X	d	d	d	d
1	1	1	1	X	X	d	d	d	d

X in A and B columns correspond to (normally) non occurring entries… The input should a valid 8421 code. Correspondingly the out is considered as non occurring (X).
d entries correspond to “don’t care”. Strictly speaking the input Boolean values are OK but since they should not be occurring, the corresponding output have no rational and could be assigned any value 0 or 1; d correspond to a “wild card” (carte joker).

OPTIMIZING BOOLEAN EXPRESSION FOR b_0 (1)

a_1a_0 \ a_3a_2	00	01	11	10
00	1	1	d	1
01			d	
11			d	d
10	1	1	d	d

$b_0 = a_3'a_0' + a_2'a_1'a_0'$

First optimization: without using don't care.

Now let us use don't care to generate larger tiles: assign 0 and 1 values to d's to maximize tile size.

OPTIMIZING BOOLEAN EXPRESSION FOR b_0 (1)

a_1a_0 \ a_3a_2	00	01	11	10
00	1	1	1	1
01			0	
11			0	0
10	1	1	1	1

$b_0 = a_0'$

When properly using the d's, the resulting Boolean expression is much simpler.

Acknowledgements

A lot of material was found in Wikipedia.

Some of these slides were inspired by slides developed by:

- University of Washington Computer Science & Engineering (CSE 370)
- Y.N. Patt (Univ of Texas Austin)
- S. J. Patel (Univ of Illinois Urbana Champaign)
- Walid A. Najjar (Univ California Riverside)
- Brian Linard (Univ California Riverside)
- G.T. Byrd (Univ North Carolina)

	0	1
0		
1		

ab \ c	0	1
00		
01		
11		
00		

ab \ c	0	1
00		
01		
11		
00		

ab	
00	
01	
11	
00	

ab \ cd	00	01	11	10
00				
01				
11				
10				