Karnaugh Map

A graphical systematic representation of truth table. It enables to find minimal representations of Boolean functions.

- $F(x_1, \ldots, x_n)$, 2^n cells representing minterms or maxterms. Two cells are adjacent iff the corresponding binary codes differ in one bit

SOP representation: cell = 1 where $f = 1$

POS representation: cell = 0 where $f = 0$

Conclusion: If two 1-cells (0-cells) are adjacent in SOP (POS) K-map, they represent redundant variable, and the corresponding minterms (maxterms) can be merged

Generalization: a rectangle of 2^m 1-cell whose corresponding minterms differ in exactly m bits, can be merged into single term comprising n-m literals.

Homework: prove the above statement by induction. Hint: divide k-map into two equal size k-maps

One-Variable $F(x_1)$

x_1	0	1
	m_0	m_1
	$F(0)$	$F(1)$

two-Variables $F(x_1, x_2)$

x_2 \ x_1	0	1
0	m_0 $F(0,0)$	m_1 $F(0,1)$
1	m_2 $F(0,1)$	m_3 $F(1,1)$

3-Variables

$x_3 = 1$

x_1 \ x_2x_3	00	01	11	10
0	m_0 $F(0,0,0)$	m_1 $F(0,0,1)$	m_3 $F(0,1,1)$	m_2 $F(0,1,0)$
$x_1 = 1$ → 1	m_4 $F(1,0,0)$	m_5 $F(1,0,1)$	m_7 $F(1,1,1)$	m_6 $F(1,1,0)$

$x_2 = 1$

Notice that all adjacent cells differ in 1-bit

map planar K-map into doughnut and

derive cross-border adjacencies

Example: $F(x_1, x_2, x_3) = \sum(0, 3, 5, 7)$

x_1 \ x_2x_3	00	01	11	10
0	1	0	(1	0
1	0	(1	1)	0

Encircle adjacent cells (rectangles of

2^m cells).

4 Variables

$x_4 = 1$

x_1x_2 \ x_3x_4	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

$x_2 = 1$

$x_1 = 1$

$x_3 = 1$

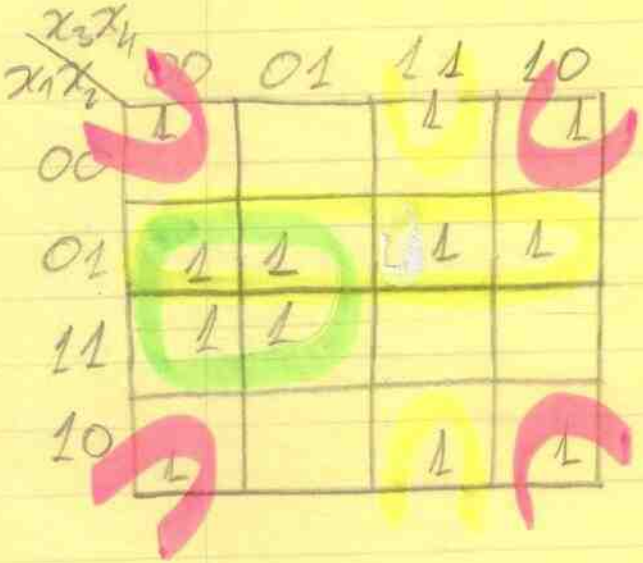
Example: $F(x_1, x_2, x_3, x_4) = \sum(1, 6, 7, 9, 12, 15)$

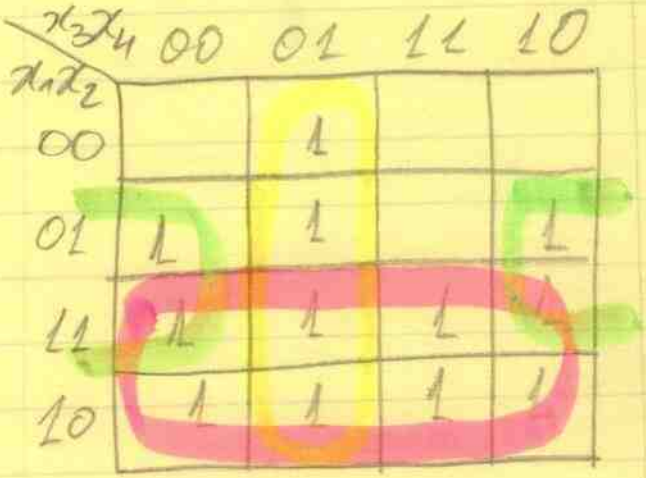
x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	0	0
01	0	0	1	1
11	1	0	1	0
10	0	1	0	0

Prime Implicants and Minimization

Find clusters of 2^m 1-cells where every 1-cell is adjacent to **m-1** other 1-cells. The cluster covers its constituent 1-cells.

The cluster represents a product of n-m literals





A cluster of k-map produces implicant

Maximal implicant (not contained in any other) is called prime implicant

Example: $m_4 + m_5 = x_1'x_2x_3'$ is an implicant but not prime. $m_4 + m_5 + m_{12} + m_{13} = x_2x_3'$ is prime implicant

Prime implicant is a product with minimal number of literals.

Theorem: Any minimal SOP expression of Boolean function F is equivalent to a Sum of prime implicants.

Proof: Let $F=P+R$ where P is an implicant (Product) but not prime and R is sum of rest products.

$P=1 \Rightarrow F=1$ but P is not prime, hence there is a literal that can be removed from P without changing F. Let Q be the new product $Q=1 \Rightarrow F=1$.

Consequently $F=Q+R$ but this is SOP with same # products as before but less literals, hence $F=P+R$ is not minimal. ▨

Corollary: A Boolean function F is equivalent to the Sum of <u>all</u> prime implicants

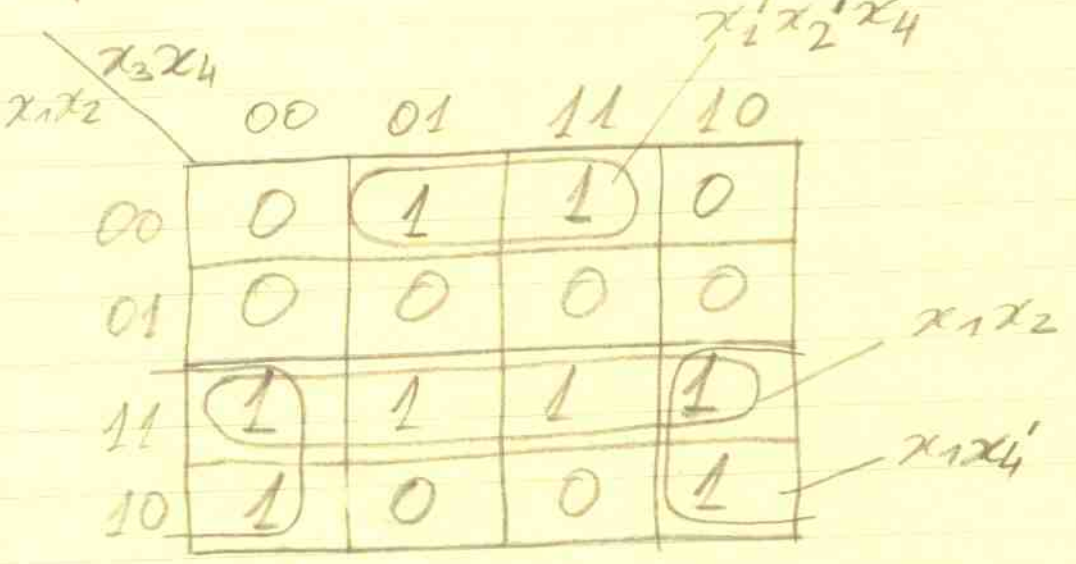
Proof: By theorem there exists G sum of prime implicants implying F. Let H be the sum of F's rest prime implicants. $H=1 \Rightarrow F=1$ Hence $F=G+H$ - sum of all prime implicants. ▨

of Prime implicants

As canonical SOP, complete SOP is unique, but minimal SOP may not be unique and also not easy to be found

Derivation of minimal SOP is 2-step
1- Determine all prime implicants
2- Select subset that covers the function and is minimal.

Example: $F(x_1, x_2, x_3, x_4) = \Sigma(1,3,8,10,12,13,14,15)$

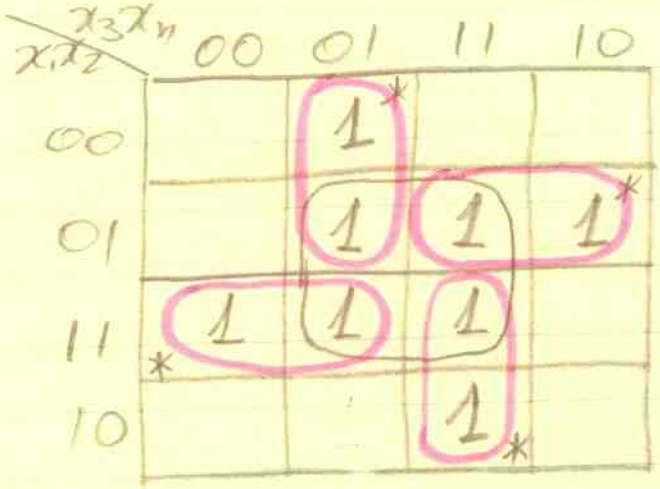


$$F = x_1'x_2'x_4 + x_1x_2 + x_1x_4'$$

Minimal Sums and Essential Prime Implicants

Observation: a prime implicant contained (covered) by union of other prime implicants is redundant

$F = \Sigma(1,5,6,7,11,12,13,15)$

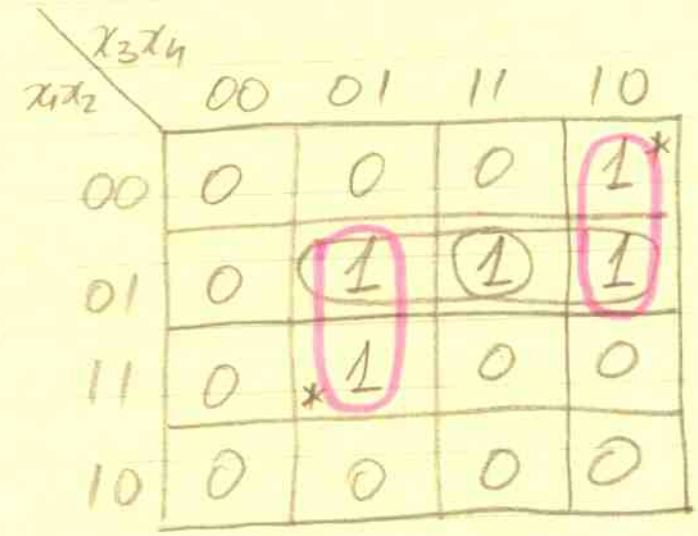


m_2, m_6, m_{12}, m_{11} each covered by unique prime implicant. These are called **essential** as by the very definition of prime implicant they must be used by any minimal SOP.

Once all essential prime implicants are chosen, minterm x_2x_4 is **redundant**

$$F = x_1x_2x_3' + x_1'x_3'x_4 + x_1'x_2x_3 + x_1x_3x_4$$

Example



$F = \Sigma(2,5,6,7,13)$

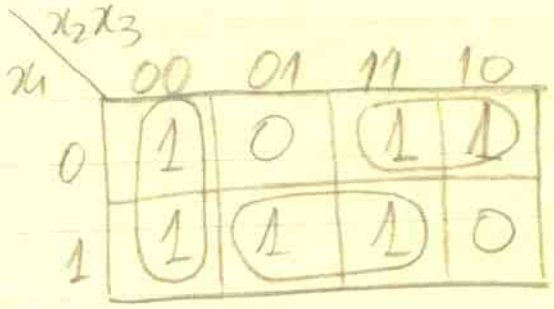
essential essential

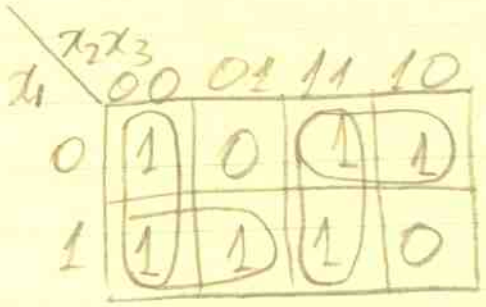
$F = x_2x_3'x_4 + x_1'x_3x_4' +$

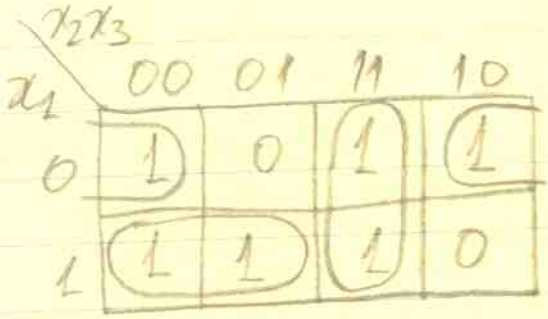
$+ \; x_1'x_2x_4$ ← non essential

$\quad\; x_1'x_2x_3$

Example: cyclic prime implicants - none essential







Every SOP in the above is non redundant

SOP is **irredundant** iff deletion of any product or literal results in a function not **equivalent** to the original

In other words no Prime implicant of the Sum is **implied** by the union of other prime implicants (clusters) of the Sum.

Example: $F(x_1, x_2, x_3, x_4) = \sum(1, 3, 4, 5, 7, 10, 11, 12, 14, 15)$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	*1	1	0
01	1	1	1	0
11	1	0	1	1
10	0	0	1	1*

x_1x_2 \ x_3x_4	00	01	11	10
00	0	1*	1	0
01	1	1	1	0
11	1	0	1	1
10	0	0	1	1*

$F = x_1'x_4 + x_1x_3 + x_1'x_2x_3' + x_1x_2x_4'$ ↑ irredundant sum

$F = x_1'x_4 + x_1x_3 + x_2x_3'x_4'$ ↑ minimal sum

5 and 6-Variable K-Maps

1-D enabled drawing 2-Variable maps

2-D enabled 3 and 4-Variable maps.

3-D enables "Volume" K-map by adding 1 or 2 Variables. Maps are built by layering two 4-Variable map (32 cells) or 4 4-Variable maps (64 cells)

Cells adjacencies are detected in 3-D

Incompletely specified functions - Don't care

It happens that some input combinations do not exist (not allowed, never occur, cannot occur). For those the output is said to be **don't care**. Unlike the case called **completely specified** where 2^n input combinations F are defined.

Don't care has big advantage in **simplification**. It is used whenever it can increase prime implicants.

Example: $F(x_1, x_2, x_3, x_4) = \sum(1,3,7,8,9,10,12,13,14,15) + \sum_d(4,5,11)$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	1	0
01	X	X	1	0
11	1	1	1	1
10	1	1	X	1

$F = x_1x_2 + x_1x_3' + x_1x_4' + x_1'x_2'x_4 + x_4'x_3x_4$

Using Don't care

$F = x_1 + x_4$

PoS Simplification

K-map Simplification clusters 0-cells and create Prime implicate. All SOP conclusions hold with appropriate modifications.

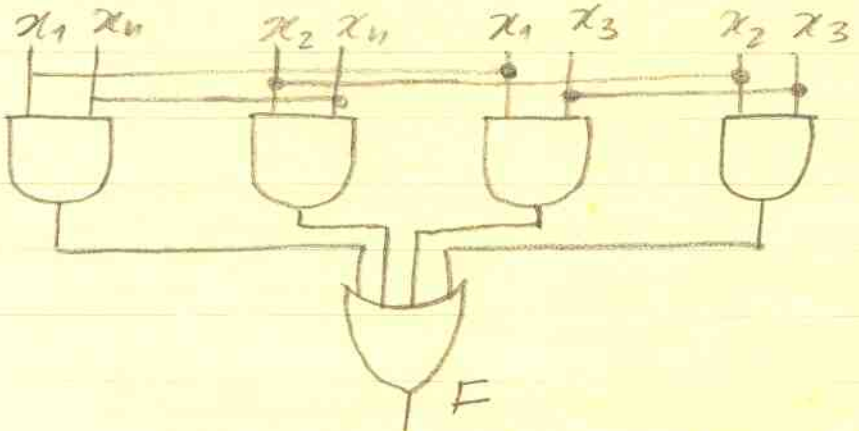
Example: $F(x_1, x_2, x_3, x_4) = \sum(5, 6, 7, 9, 10, 11, 13, 14, 15)$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

SOP:

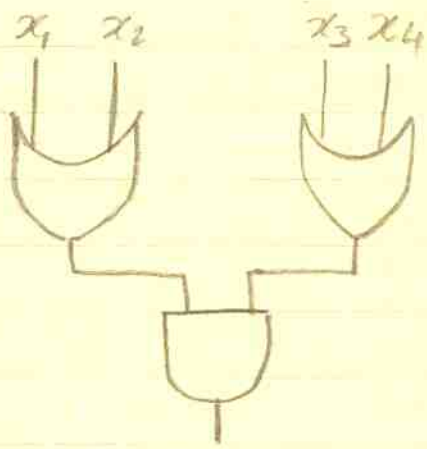
algebraic simplification ↓

$$F = x_1x_4 + x_2x_4 + x_1x_3 + x_2x_3 = (x_1 + x_2)x_4 + (x_1 + x_2)x_3 = (x_1 + x_2)(x_3 + x_4)$$



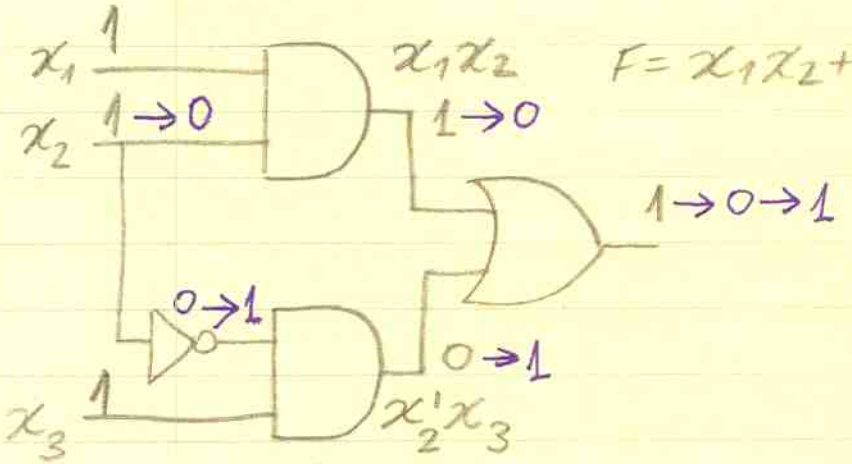
POS:

$$F = (x_1 + x_2)(x_3 + x_4)$$



Hazards in Combinational Circuits

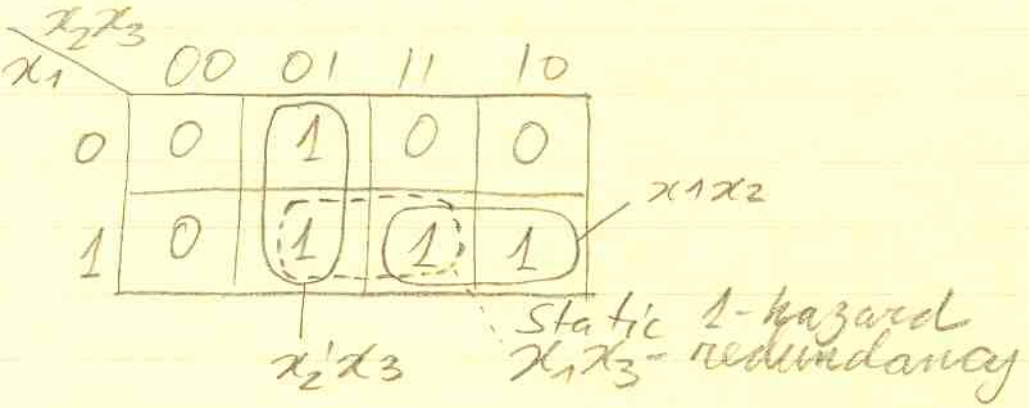
Propagation delay may cause erratic output



If as a response to input change the output goes momentarily to 0, where it should stay 1 this is called static 1-hazard.

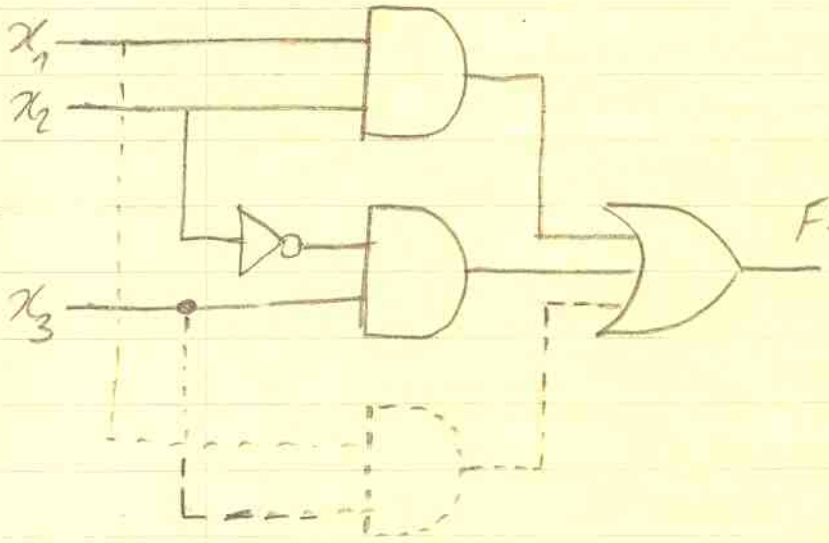
The opposite is called static 0-hazard

Consider K-map of Implementation



The hazard occurs because switching one input results in transition from one implicant to another one and as a result of delay the input of OR is not well defined

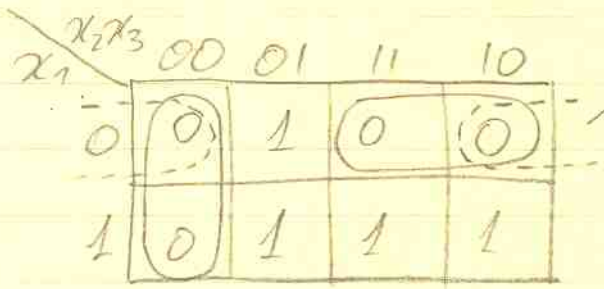
Solution: Introduce redundancy!



Concensus theorem

$F = x_1x_2 + x_2'x_3 + x_1x_3$

Static 0-hazard occurs in POS implementation



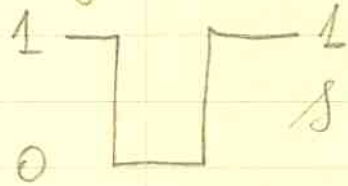
static 0-hazard

$F = (x_2 + x_3)(x_1 + x_2')$ concensus

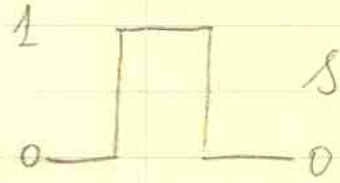
resolve 0-hazard by redundancy

$F = (x_2 + x_3)(x_1 + x_2')(x_1 + x_3)$

Hazards have reliability implications. Minimality and reliability are conflicting

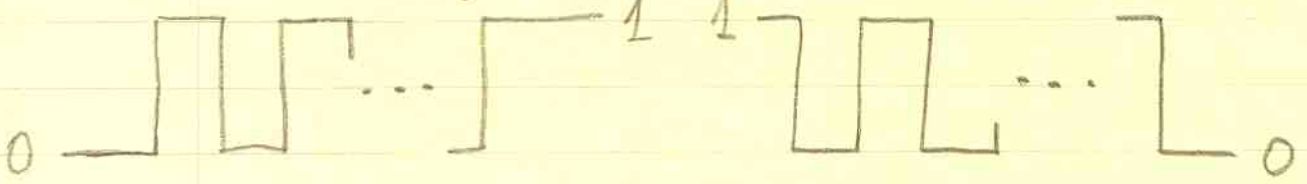


static 1-hazard



static 0-hazard

Dynamic hazards



output changes 3 or more (odd) times. This occurs due to a change of 3 inputs reaching to output at 3 different paths

Quine - McCluskey Tabulation Method

Limitation of k-map: Visualization, 6-Variable
Selection of minimal Sum

① generate all prime implicants

② select those that will appear in minimal SOP

Two adjacent 1-cell in k-map can be merged. They differ in 1bit

Example: m_7: 00111 $(x_1'x_2'x_3x_4x_5)$
m_{23}: 10111 $(x_1x_2'x_3x_4x_5)$

$m_7 + m_{23}$: -0111 $(x_2'x_3x_4x_5)$

Binary strings p and q differ in one bit

$$\left.\begin{array}{l} p = k_1 \cdots k_{i-1} k_i k_{i+1} \cdots k_n \\ q = k_1 \cdots k_{i-1} k_i' k_{i+1} \cdots k_n \end{array}\right\} p + q = k_1 \cdots k_{i-1} (-) k_{i+1} \cdots k_n$$

$k_j \in \{0, 1, -\}$ $j \neq i$, $k_i \in \{0, 1\}$

Example: $F(x_1, x_2, x_3, x_4) = \sum(0, 1, 2, 3, 5, 6, 8, 11)$

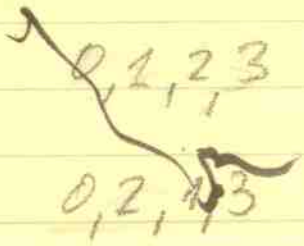
① convert all minterms to {0,1} bit strings. group according to # of 1's in ascending order

# 1's	Decimal Label	Product String ($x_1x_2x_3x_4$)	
0	0	0000	✓
1	1	0001	✓
	2	0010	✓
	8	1000	✓
2	3	0011	✓
	5	0101	✓
	6	0110	✓
3	11	1011	✓

② Iterate over successive groups and detect for merging. If a string has merged, check in above table

0	0,1	000-	✓
	0,2	00-0	✓
	0,8	-000	
1	1,3	00-1	✓
	1,5	0-01	
	2,3	001-	✓
	2,6	0-10	
2	3,11	-011	

③ Iterate ② untill no more checks

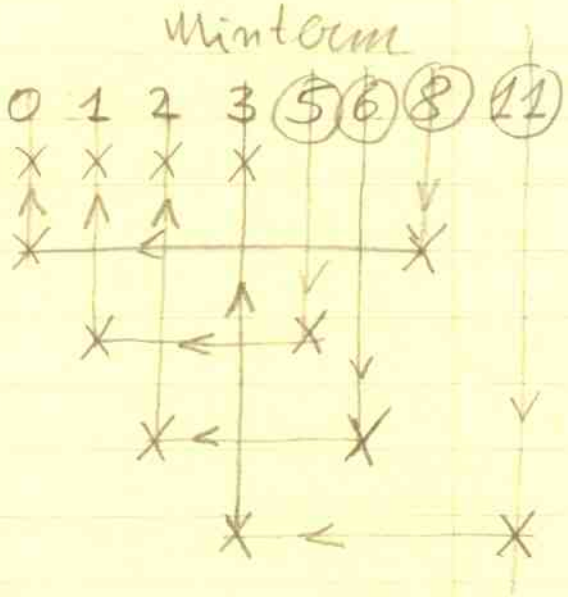


00--

00-- duplicate

We cannot apply step 2 any more and the process terminates. Unchecked strings are all the prime implicants

Decimal Label	String	Prime Implicant
0,1,2,3	0 0 - -	$x_1'x_2'$
0,8	- 0 0 0	$x_2'x_3'x_4'$
1,5	0 - 0 1	$x_1'x_3'x_4$
2,6	0 - 1 0	$x_1'x_3x_4'$
3,11	- 0 1 1	$x_2'x_3x_4$



Minterms 5, 6, 8, 11 are covered by single prime implicant (essential), hence must participate in minimal SOP. Their selection covers automatically all other minterms, hence we are done.

$$F = x_2'x_3'x_4' + x_1'x_3'x_4 + x_1'x_3x_4' + x_2'x_3x_4$$

Example: Full coverage of F by essential prime implicants may not happen

$$F(x_1, x_2, x_3, x_4) = \sum(0, 1, 3, 6, 7, 14, 15)$$

# 1's	Decimal Label	String	
0	0	0 0 0 0	✓
1	1	0 0 0 1	✓
2	3	0 0 1 1	✓
	6	0 1 1 0	✓
3	7	0 1 1 1	✓
	14	1 1 1 0	✓
4	15	1 1 1 1	✓

0	0,1	0 0 0 -	
1	1,3	0 0 - 1	
2	3,7	0 - 1 1	
	6,7	0 1 1 -	✓
	6,14	- 1 1 0	✓
3	7,15	- 1 1 1	✓
	14,15	1 1 1 -	✓

2	6,14,7,15	- 1 1 -

minterms

Decimal label	String	0	1	3	6	7	14	15
6,7,14,15	- 1 1 -				X	X	X	X
0,1	0 0 0 -	X	X					
1,3	0 0 - 1		X	X				
3,7	0 - 1 1			X		X		

$$F = x_2x_3 + x_1'x_2'x_3' + \begin{cases} x_1'x_2'x_4 \\ \text{OR} \\ x_1'x_3x_4 \end{cases}$$ are both minimal

Dominant Row Removal

A row P_i (Prime implicant) dominates another row P_j if P_i covers all minterms which P_j does and #literals in P_i is not greater than P_j. P_j can then be eliminated. (why?)

A column C_i dominates C_j if C_i has X wherever C_j does. C_i can then be eliminated.

Any prime implicant that will be chosen to realize C_j will automatically realize C_i, hence C_i is redundant.

Example:

Decimal	String	0	2	(5)	6	7	8	(9)	(12)	13	(15)
8,9,12,13	1-0-						X	X	X	X	
5,7,13,15	-1-1			X		X				X	X
0,2	00-0	X	X								
0,8	-000	X					X				
2,6	0-10		X		X						
6,7	011-				X	X					

① Minterms (Columns 5, 9, 12, 15) are covered by Single minterms, hence 1-0-(x_1x_3') and -1-1 (x_2x_4) must be selected

		minterm		
	string	0	2	6
0,2	00-0	X	X	
0,8	-000	X		
2,6	0-10		X	X
6,7	011-			X

Row 0,2 dominates 0,8 and 2,6 dominates 6,7. But then column C_2 is dominated since it dominates 0 and 6 so

$F = x_1x_3' + x_2x_4 + x_1'x_2'x_4' + x_1'x_3x_4'$