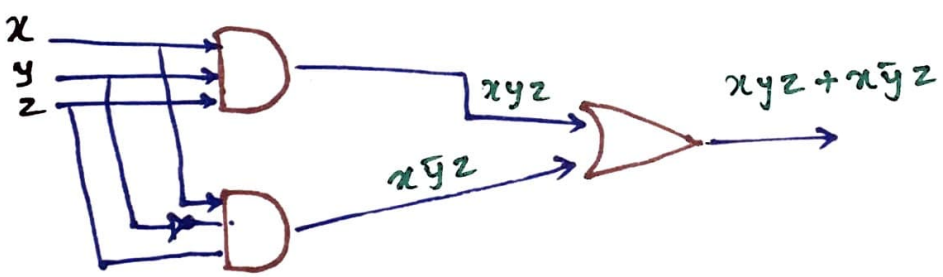
Week IX: Lecture Notes

Minimization of circuits.

- The efficiency of a combinational circuit depends on the number and arrangements of its gates.
- The process of designing a combinational circuit:
  - Begin with the table specifying the output for each combination of input values.
  - May use the sum-of-product expansion of the circuit and find a set of logic gates.
- The sum-of-product expansion may contain many more terms than are necessary.

Example:



2 AND gate
1 OR gate
1 Inverter

Sum of products expansion is $xyz + x\bar{y}z$

these terms differ only in one variable, and so can be combined

$xyz + x\bar{y}z = x(y+\bar{y})z = x \cdot 1 \cdot z = xz$ → a Boolean expression with few operators representing the circuit

only one AND gate

- Combining terms in the sum of products expansion leads to a simpler expansion for the circuit.

Minimization of the Boolean Function

- produces Boolean sums of Boolean products that represent a Boolean function with:
  a). fewest products of literals
  b). each product contains the fewest literals possible.

  among all sums of products that represent the Boolean function.

- makes it possible to construct a circuit for this Boolean function that uses:
  a). fewest gates
  b). fewest inputs to the AND and OR gates.

  in the circuit, among all the Boolean expressions we are minimizing.

- Karnaugh Maps (or K-Maps), 1950
  - useful in minimizing circuits upto six variables

- Quine. Mc Cluskey method, 1960
  - automates the process of minimizing combinational circuits, can be implemented on a computer program.

K-map Method (Graphical), 1953

- visual method for simplifying sum-of-products expansion
- applied only when the function involves six or fewer variables.

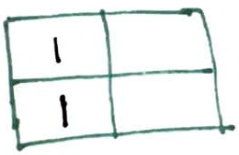
K-map with 2 variables → a square of 4 cells

	y	$\bar{y}$
x	xy	$x\bar{y}$
$\bar{x}$	$\bar{x}y$	$\bar{x}\bar{y}$

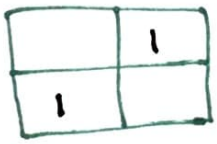
- 4 possible minterms
- 4 cells
- place 1 in a cell if the corresponding minterm representing the cell is in the given Boolean expression
- adjacent cells have minterm that differ in only one literal.

Example: Find the K-map for

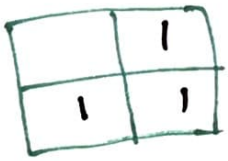
a. $xy + \bar{x}y$



b. $x\bar{y} + \bar{x}y$



c. $x\bar{y} + xy + \bar{x}\bar{y}$



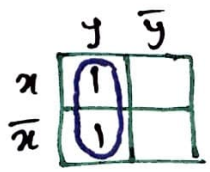
- identify minterms that can be combined from the K-map
- is in two adjacent cells in the K-map, the minterms can be combined to produce a product involving only one variable.

- 1's in all four cells, the four min-terms can be combined into one term, namely the Boolean expression 1 that involves none of the variables.
- circle blocks of cells in the K-map that represents minterm that can be combined and then find the corresponding sum of products.

- goal is to identify the largest possible blocks and cover all the 1s with fewest blocks using the largest blocks first and always using the largest possible blocks.

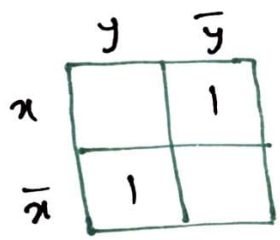
Example:

(a) $xy + \bar{x}y$

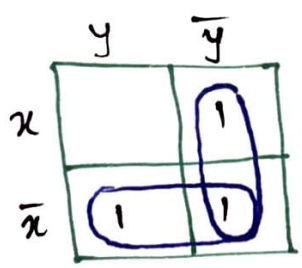


$xy + \bar{x}y = y$

(b) $x\bar{y} + \bar{x}y$

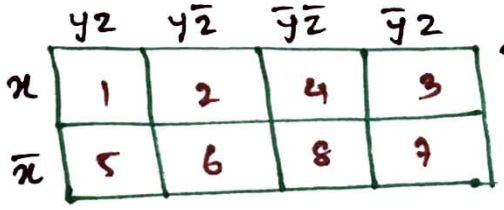


(c) $x\bar{y} + \bar{x}y + \bar{x}\bar{y}$



$x\bar{y} + \bar{x}y + \bar{x}\bar{y} = \bar{x} + \bar{y}$

K-map with three variables



← one way shown

- think of it as lying as a cylinder
- cell 1, cell 3 adjacent
- cell 5, cell 7 adjacent

To simplify a sum-of-product expansion in 3-variables

- We K-map to identify the blocks of minterms that can be combined
- blocks of two adjacent cells represent pairs of minterms that can be combined a product of literal.
- blocks of 2x2 and 1x1 cell represent minterms that can be combined into a single literal.
- blocks of all 8 cells represents a product of no literals.

Implicant: The product of literals corresponding to a block of 1s in the K-map is called an implicant of the function to be minimized.

Prime Implicant: It is called prime implicant if this block of 1's is not contained in a larger block of 1's representing the product of fewer literals than in this product.

- Goal is to find the largest possible blocks in the map and cover all the 1's in the map with least number of blocks using largest blocks first

Essential Prime Implicant: The largest blocks are always chosen, but we must always choose a block if it is the only block of 1's, covering a 1 in the K-map. Such a block is called an **essential prime implicant.**

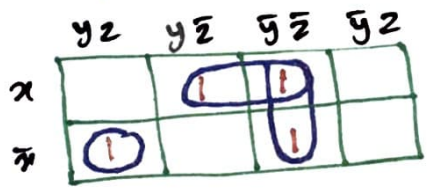
- Covering all the 1s in the map with blocks corresponding to prime implicants, we can express the sum-of-products as a sum of prime implicants.

Note: There may be more than one way to cover all the 1s using the least number of blocks.

Example:

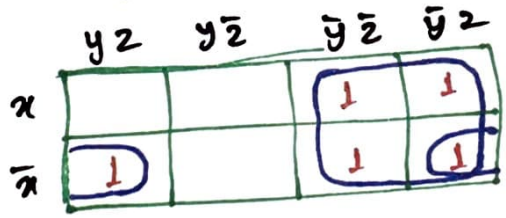
Use K-maps to minimize the sum of product expansions.

(a) $xy\bar{z} + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}\bar{y}\bar{z}$



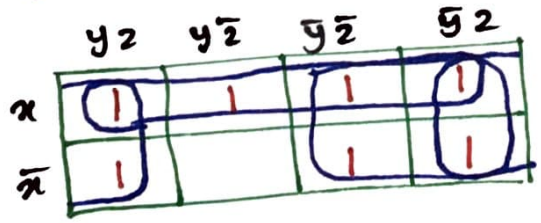
$= x\bar{z} + \bar{x}yz + \bar{y}\bar{z}$

(b) $x\bar{y}z + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$



∴ minimal expansion into Boolean sums of Boolean products is $\bar{y} + \bar{x}z$

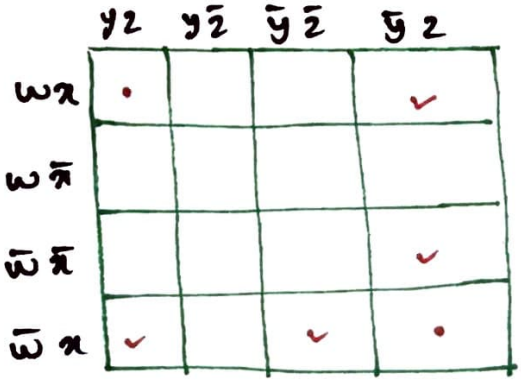
(c) $x\bar{y}z + xyz + xy\bar{z} + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$



∴ simplified minimal form is $z + \bar{y} + x$

K-map with four variables : a square divided into 16 cells, representing 16 possible minterms.

One of the ways to form a K-map in 4 variables w, x, y, z



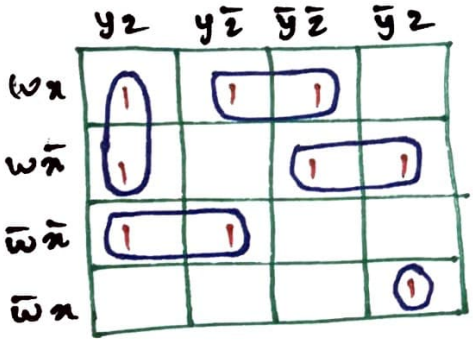
- two cells are adjacent iff the minterms the represent differ in one literal
- each cell is adjacent to 4 other cells
- This K-map can be thought of as lying on a torus so that adjacent cells have common boundary.

Simplication Procedure

- Identify those blocks of 2, 4, 8 or 16 cells that represent minterms that can be combined.
- each cell representing a minterm must either be used to form a product using fewer literals or be included in the expansion.

Example

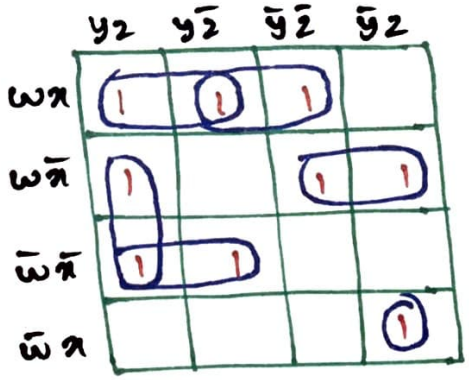
(a) $wxyz + wxy\bar{z} + wx\bar{y}\bar{z} + w\bar{x}yz + w\bar{x}\bar{y}z + w\bar{x}\bar{y}\bar{z} + \bar{w}x\bar{y}z + \bar{w}\bar{x}yz + \bar{w}\bar{x}y\bar{z}$



$= wx\bar{z} + w\bar{x}\bar{y} + \bar{w}\bar{x}y + wyz + \bar{w}x\bar{y}z$

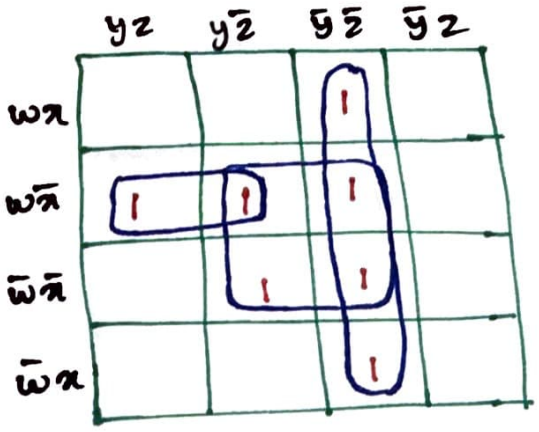
5 implicants, least no. of blocks covering all 1's

OR



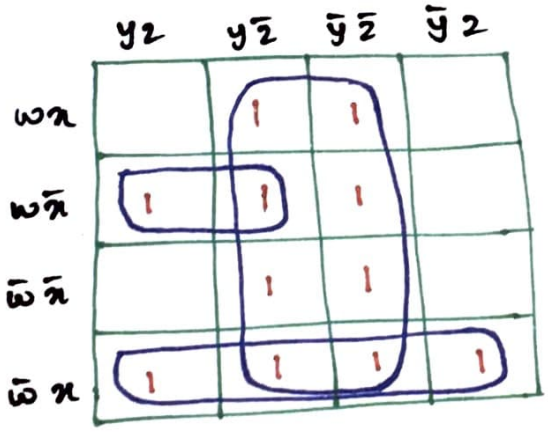
6 implicants, NOT least no. of blocks covering all 1's

(b) $wx\bar{y}\bar{z} + w\bar{x}yz + w\bar{x}y\bar{z} + w\bar{x}\bar{y}\bar{z} + \bar{w}x\bar{y}\bar{z} + \bar{w}\bar{x}yz + \bar{w}\bar{x}\bar{y}\bar{z}$



$= \bar{x}\bar{z} + \bar{y}\bar{z} + w\bar{x}y$

(c) $wxy\bar{z} + wx\bar{y}\bar{z} + w\bar{x}yz + w\bar{x}\bar{y}\bar{z} + w\bar{x}y\bar{z} + \bar{w}xyz + \bar{w}xy\bar{z}$
$+ \bar{w}x\bar{y}\bar{z} + \bar{w}x\bar{y}z + \bar{w}\bar{x}y\bar{z} + \bar{w}\bar{x}\bar{y}\bar{z}$



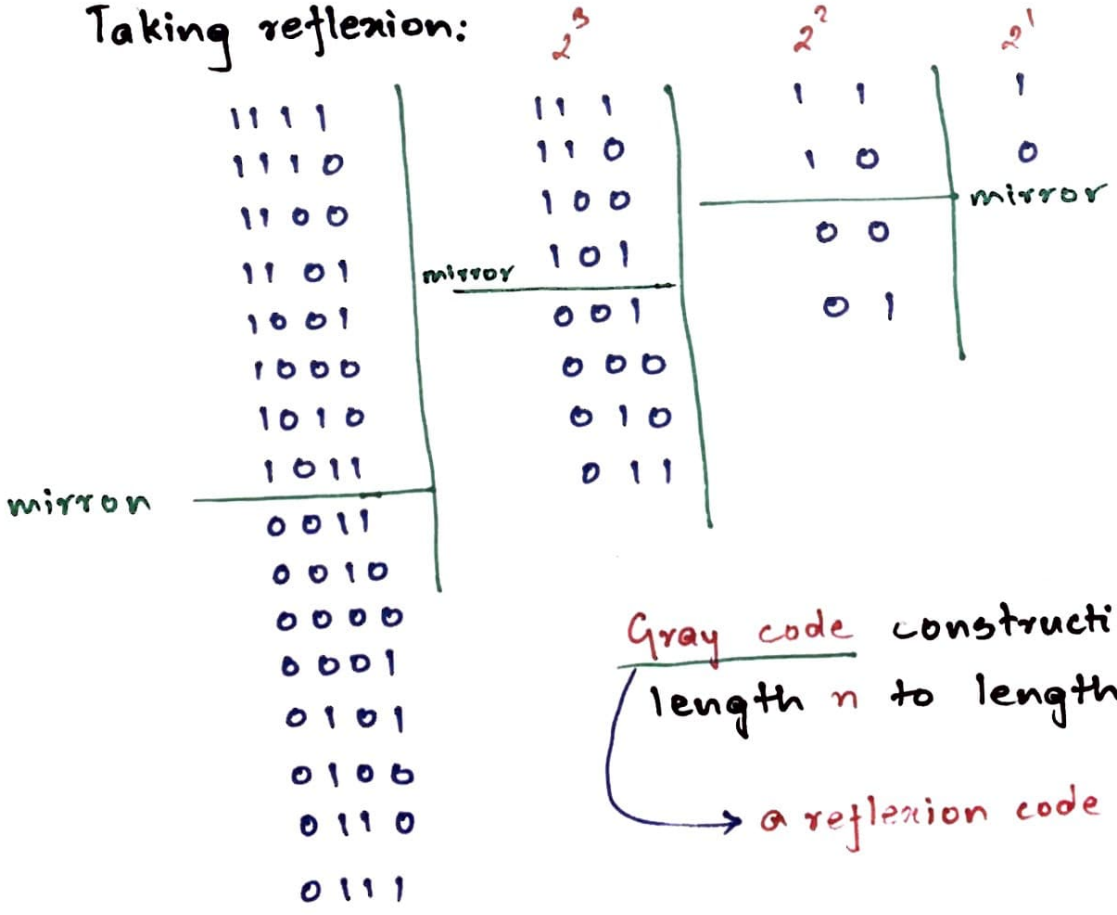
$= \bar{z} + \bar{w}x + w\bar{x}y$

- Realistic with five or six variables
- Beyond five or six variables, rarely used as K-maps become extremely complicated.
- Concept is very useful to devise new algorithms to minimize Boolean functions implemented in the computer aided design (CAD) program.

- Two variable $\xrightarrow{use}$ 2 x 2 rectangle
  Three variables $\longrightarrow$ 2 x 4 rectangle
  four variables $\longrightarrow$ 4 x 4 rectangle
  n- variables $\longrightarrow$ $2^{\lfloor n/2 \rfloor} \times 2^{\lceil n/2 \rceil}$ rectangle

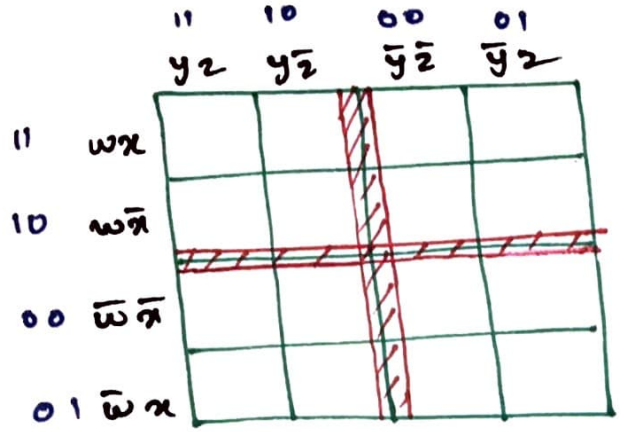
- Gray code used to assign a K-map to the minterms

Taking reflexion:



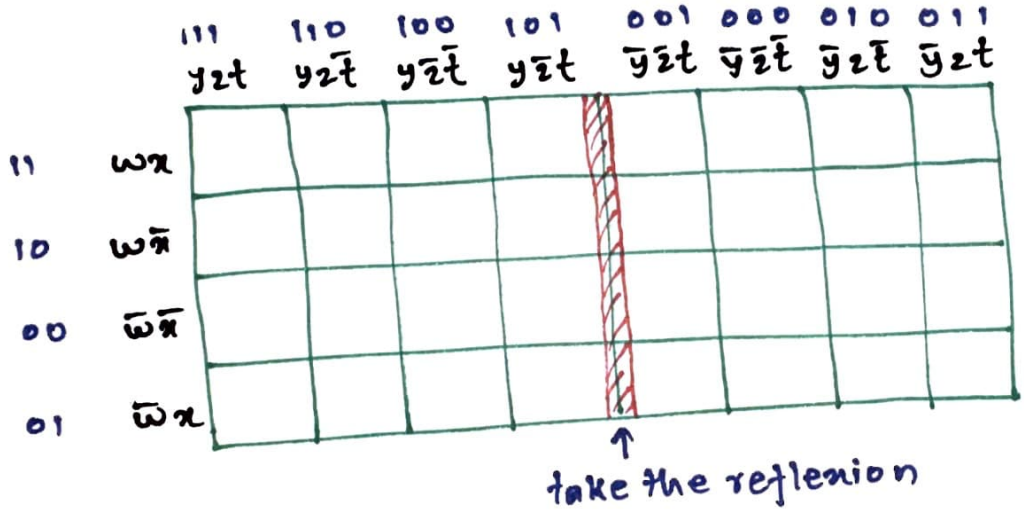
Gray code construction from length n to length n+1 → a reflexion code

4-variables 4x4, w, x, y, z



Considering 1st row and last row cells adjacent and 1st column and last column adjacent and using Gray code to labels the rows and columns ensures minterms that differ in only one variable are always adjacent.

5-variables 4x8, w, x, y, z, t



- To ensure all cells representing products that differ in only one variable are considered adjacent, verify
  - cells in top and bottom cells are adjacent
  - 1st and 8th column are adjacent
  - 1st and 4th column are adjacent
  - 5th and 8th column are adjacent
  - 2nd and 7th, 3rd and 6th are adjacent.

Use of k-maps to minimize a Boolean function in n-variables

- Draw a k-map of the appropriate size
  - Place 1s in all cells corresponding to minterm in the sum of product expansion of this function
- Identify all prime implicants of the Boolean function
  - look for the blocks consisting of 2^k clustered cells containing a 1, where $1 \leq k \leq n$
  - these blocks correspond to the product of $n-k$ literals.
  - a block of 2^k cells each containing a 1 not contained in a block of 2^{k+1} cells each containing 1 is a prime implicant as no product obtained by deleting a literal is also represented by a block of cells all containing 1

- k-map containing 2^n cells (i.e. n variables)

  Use a rectangle containing $2^{\lfloor n/2 \rfloor}$ rows and $2^{\lceil n/2 \rceil}$ columns

  $$\left(\lfloor n/2 \rfloor + \lceil n/2 \rceil = n \right)$$

- Once all prime implicants have been identified, the goal is to find the smallest possible subset of these prime implicants, with the property that: the cells representing these prime implicants cover all the cells containg a 1 in the K-map
  - first select the essential prime implicants, as each of these is represented by a block that covers a cell containing a 1 that is not covered by any other prime implicant
  - add additional prime implicants to ensure that all 1s in the K-map are covered. (This step can become exceedingly complicated for larger number of variables)

Don't Care Conditions

- some circuits care about only output for some combinations of input values
- Other combinations of input values are not possible or never occur
- This gives us freedom in producing a simple circuit with the desired output
- The output values for all those combinations that never occur can be arbitrarily chosen
- The values of functions for these combinations are called don't care conditions.

- A d is used in K-map to mark those combinations of values of the variables for which the function can be arbitrarily assigned.
- In the minimization process, we can assign 1 as values of those combinations of input values that lead to the largest blocks in the K-map.

Example: BCD (Binary Coded Decimal)

Digit	BCD codeword	
0	0 0 0 0	0
1	0 0 0 1	0
2	0 0 1 0	0
→ 3	0 0 1 1	0
4	0 1 0 0	0
5	0 1 0 1	1
6	0 1 1 0	1
→ 7	0 1 1 1	1
→ 8	1 0 0 0	1
9	1 0 0 1	1
	w x y z	f

873 → 1000 0111 0011

6 combinations out of 16 never used to encode digits

Question:

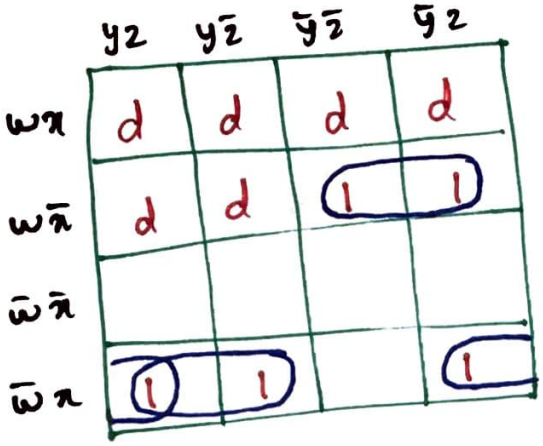
Suppose that a circuit is built that produces an output of

$$\begin{cases} 1 & \text{if the decimal digit is } \geq 5 \\ 0 & \text{if the decimal digit is } < 5 \end{cases}$$

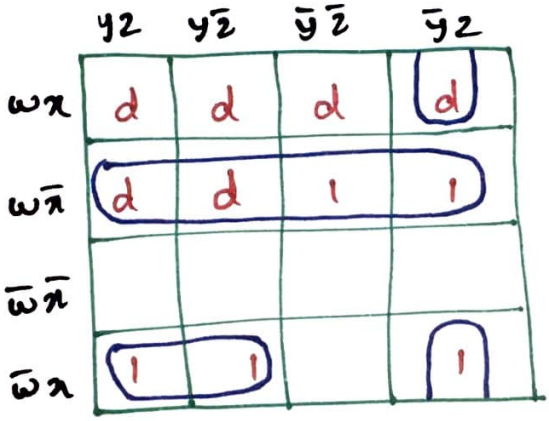
How can this circuit be simply built using OR gate, AND gate and inverters ?

$$f = \bar{w}x\bar{y}z + \bar{w}xy\bar{z} + \bar{w}xyz + w\bar{x}\bar{y}\bar{z} + w\bar{x}\bar{y}z$$

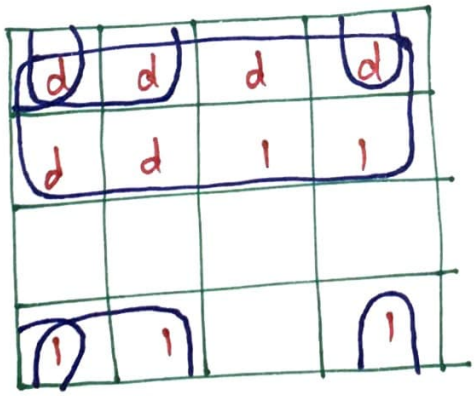
The K-map for f, with d in the don't care positions.



$f = w\bar{x}\bar{y} + \bar{w}xy + \bar{w}xz$



$f = w\bar{x} + \bar{w}xy + x\bar{y}z$



$f = w + xy + xz$

simplest possible sum of product expansion

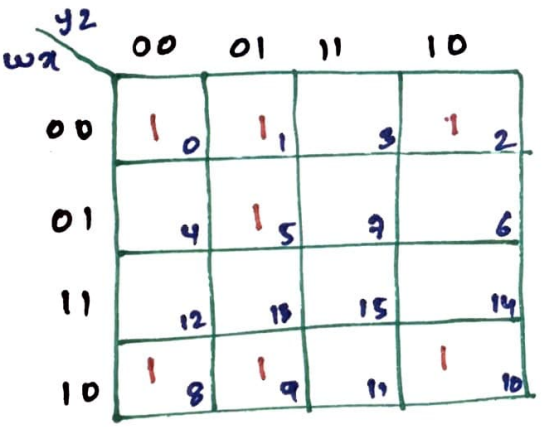
Exercise:

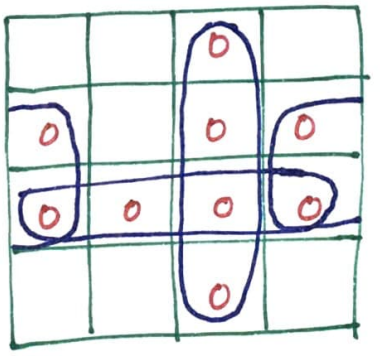
Simplify the Boolean function $f(w,x,y,z) = \Sigma(1,3,7,11,13)$ and the dont care condition $d(w,x,y,z) = \Sigma(0,2,5)$

- 1s placed in the sequences of the map represents the minterm of f
- the minterms not included in f, denote the compliment $\bar{f}$ of f
- complement $\bar{f}$ of f is represented in the map by a square not marked 1
- mark empty squares by 0s and combine them into valid adjacent squares

  → Get a simplified expression of $\bar{f}$ of f
- Then apply DeMorgan's laws to $\bar{f}$ to get f back

Example: $f = \Sigma\,(0,1,2,5,8,9,10)$





$\bar{f} = \bar{y}\bar{z} + \bar{w}\bar{x} + \bar{x}\bar{z}$

$f = (y+z)(w+x)(x+z)$

- Unfortunately, minimizing Boolean function with many variables is a computationally intensive problem
  - → NP- complete problem
  - → No polynomial time algorithm for minimizing Boolean circuits so far

- Quin - McCluskey's method → exponential complexity and only used when # literals ≤ 10

- So far, the best algorithm minimizes only circuits with no more than 25 variables

- Heuristic methods → used to substantially simplify Boolean expressions with a large number of literals.
  - → not necessarily minimize

The Quine- Mc Clusky Method (1950)

- can be used for Boolean functions in any number of variables
- Two points
  (i) find those terms that are candidates for inclusion in a minimal expansion as a Boolean sum of Boolean products
  (ii) determine which of these terms to actually use

Example: Show how the Quine- Mc Cluskey method can be used to find a minimal expansion equivalent to

$$xyz + x\bar{y}z + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$$

(i)

minterms	bitstring	# of 1's	
1. xyz	111	3	(1,2)
2. $x\bar{y}z$	101	2	(1,3)
3. $\bar{x}yz$	011	2	(2,4)
4. $\bar{x}\bar{y}z$	001	1	(3,4)
5. $\bar{x}\bar{y}\bar{z}$	000	0	(4,5)

- minterms that can be combined are those that ~~can be~~ differ in exactly one literal
- Hence, two terms that can be combined differ by exactly one in the # of 1s in the bitstring.

- When two minterms are combined into a product, this product contains two literals

- A product in two literals is represented using a dash to denote the variable that doesnot occur.

Step 0		Step 1		Step 2	
Terms	Bit strings	Term	Bit strings	Term	Bit strings.
xyz	111	xz	1_1	z	__1
$x\bar{y}z$	101	yz	_11		
$\bar{x}yz$	011	$\bar{y}z$	_01		
$\bar{x}\bar{y}z$	001	$\bar{x}z$	0_1		
$\bar{x}\bar{y}\bar{z}$	000	$\bar{x}\bar{y}$	00_		

- Next, all pairs of products of two literals that can be combined are combined into one literal

Two such products can be combined if they contain literals for the same two variables, and literals for only one of the two variables differ.

(ii) To identify a minimal set of products needed to represent the Boolean function.

- Note that the above table indicate which terms have been used to form products with fewer literals (1,2,3,4). These terms will not be needed in a minimal expansion

- Begin with all those products that were not used to construct products with fewer literals.

Cover finding Table

candidate product	1 $x y \bar{z}$	2 $x \bar{y} z$	3 $\bar{x} y z$	4 $\bar{x} \bar{y} z$	5 $\bar{x} \bar{y} \bar{z}$
z	X	X	X	X	
$\bar{x} \bar{y}$				X	X

- Form a table which has a row for each candidate product formed by combining original terms of a column for each original term.
- Put an X in a position if the original term in the sum-of-product expansion was used to form the candidate product
- In this case, the candidate product is said to cover the original minterm, (eg. $\bar{x} \bar{z}$ covers $\bar{x} \bar{y} z$ and $\bar{x} \bar{y} \bar{z}$)
- need to include atleast one product that covers each of the original minterms
- Only one X in a column in the table → corresponding product in the row of this X must be used.
- From the above table, see that both z and $\bar{x} \bar{y}$ are needed

Hence, the final answer is $z + \bar{x} \bar{y}$

Sequence of steps to simplify a sum-of-products expression in the Quine-McCluskey method

1. express each minterm in n variables by a bit string of length n while
   - 1 in the i^{th} position if x_i occurs
   - 0 in the i^{th} position if $\bar{x}_i$ occurs
2. group the bit strings accordingly to the # of 1's in them
3. determine all products in $n-1$ variables that can be formed by taking the Boolean sum of minterms in the expansion
   - minterms that can be combined are represented by bit strings that differ in exactly one positions
   - represent these products in $n-1$ variables with strings that have
     - 1 in the i^{th} position if x_i occurs in the product
     - 0 in the i^{th} position if $\bar{x}_i$ occurs in the product
     - - (dash) in the i^{th} position if there is no literal involving of x_i in the product
4. Determine all products in $n-2$ variables that can be formed by taking the Boolean sum of products in $n-1$ variables found in the previous step.
   - products in $n-1$ variables that can be combined are represented by bit strings that have a dash in the same position and differ in exactly one position

5. Continue combining Boolean products into products with fewer variables as long as possible

6. Find all the Boolean products that arose that were not used to form a Boolean product in one fewer literal

7. Find the smallest set of these Boolean products such that the sum of these products represents the Boolean function
   - done by forming a table showing which minterms are covered by which products
   - every minterm must be covered by atleast one product
   - first find all essential prime implicants
   - An essential prime implicants is the only prime implicant that covers one of the minterms
   - Then simplify the table by eliminating the columns for minterms covered by this prime implicant
   - eliminate any prime implicants that cover a subset of minterms covered by another prime implicant.

     (e.g. $\bar{x}yz$ can be ignored while $w\bar{x}y$ is included together with $\bar{w}z$ and $wy\bar{z}$ as:

     $\bar{x}yz$ covers $\{w\bar{x}yz, w\bar{x}y\bar{z}\}$

- eliminate column for a minterm if there is another minterm that is covered by a subset of the prime implicants that cover this minterm

  (e.g. minterm $\bar{w}\bar{x}yz$ can be ignored, $\bar{w}\bar{x}y$ is covered by prime implicants $\bar{w}z$ and $\bar{x}yz$

  Let 'm' be a minterm covered by 'c', then if $\exists$ another minterm 'm' with $c' \subseteq c$, delete column for m.)

- This process of
  (a) identifying essential prime implicants that must be included,
  (b) eliminating redundant prime implicants (i.e. eliminating dominated rows), and
  (c) identifying minterm that can be ignored (i.e. eliminating dominating columns)

  is iterated untill the table doesnot change

- finally, use a backtracking procedure to find the optimal solution, when we add prime implicants to the cover to find possible solutions.

- finding cover is NP-complete problem

Example

$w\bar{x}y\bar{z} + w\bar{x}yz + w\bar{x}\bar{y}\bar{z} + \bar{w}xyz + \bar{w}x\bar{y}z + \bar{w}\bar{x}yz + \bar{w}\bar{x}\bar{y}z$

Step 0

	Terms	Bit strings		
1.	$w\bar{x}y\bar{z}$	1110	(3)	(1,3)
2.	$w\bar{x}yz$	1011	(3)	(1,5)
3.	$w\bar{x}\bar{y}\bar{z}$	1010	(2)	(4,6)
4.	$\bar{w}xyz$	0111	(3)	(2,3)
5.	$\bar{w}x\bar{y}z$	0101	(2)	(2,5)
6.	$\bar{w}\bar{x}yz$	0011	(2)	(2,6)
7.	$\bar{w}\bar{x}\bar{y}z$	0001	(1)	(6,7)
				(4,5)
				(4,6)
				(5,7)

Step 1

Terms	Bit strings		
$wy\bar{z}$	1-10	(6,7,4,5)	wz
$w\bar{x}y$	101-		
$\bar{x}yz$	-011		
$\bar{w}\bar{x}z$	00-1		
$\bar{w}xz$	01-1		
$\bar{w}yz$	0-11		
$\bar{w}\bar{y}z$	0-01		

Step 2

wz	0--1

	1 $w\bar{x}\bar{y}z$	2 $w\bar{x}yz$	3 $w\bar{x}y\bar{z}$	4 $\bar{w}xyz$	5 $\bar{w}x\bar{y}z$	6 $\bar{w}\bar{x}yz$	7 $\bar{w}\bar{x}\bar{y}z$
$\bar{w}z$				(x)	x	x	x
$wy\bar{z}$	(x)		x				
$w\bar{x}y$		x	x				
$\bar{x}yz$		x				(x)	

Final answer: $\bar{w}z + wy\bar{z} + w\bar{x}y$

or

$\bar{w}z + wy\bar{z} + \bar{x}yz$