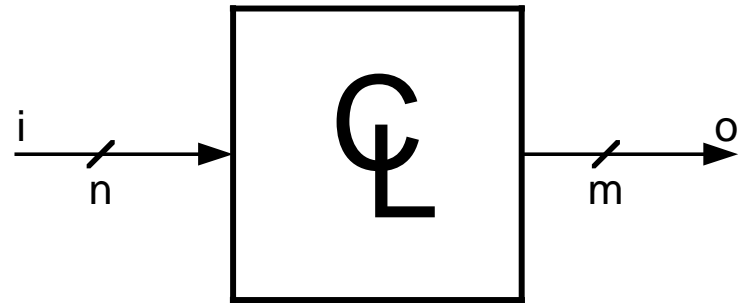
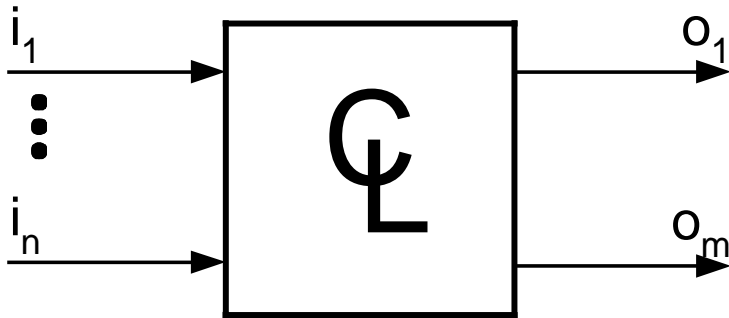


---

# Combinational Logic Design – Descriptions

# Combinational logic is memoryless

---

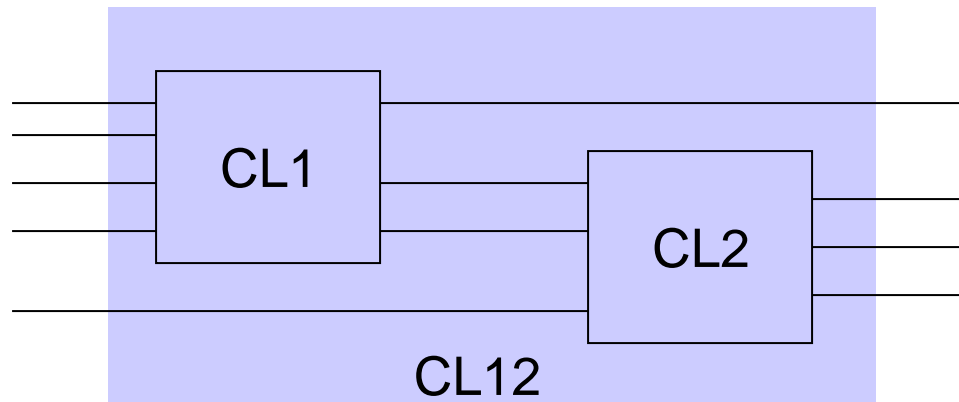


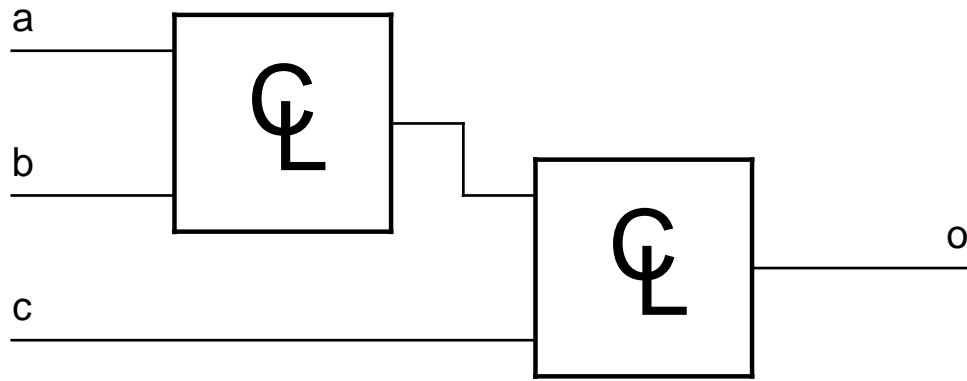
$$o = f(i)$$

# Closure

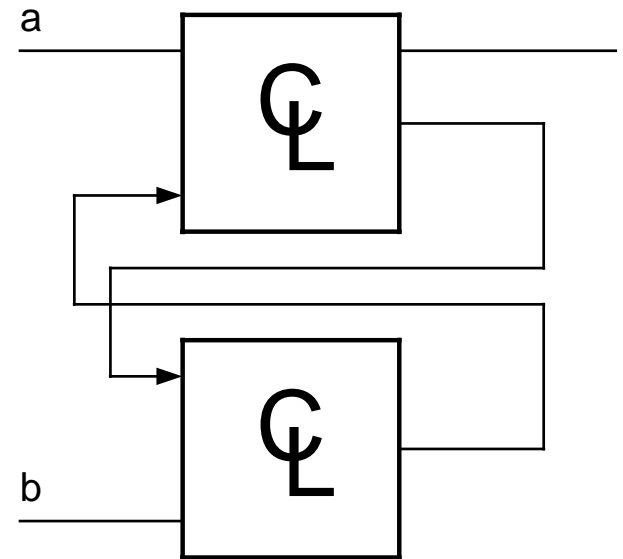
---

- Combinational logic circuits are closed under acyclic composition.





YES



NO

# Combinational not Combinatorial

---

Combinational



combinational logic circuit  
combines inputs to generate  
an output

Combinatorial



mathematics of counting

# Several ways to describe a combinational logic function

---

## Example – Majority Circuit

1. English language description
  - Outputs 1 if more inputs are 1 than are 0
2. Logic equation
  - $q = (a \wedge b) \vee (a \wedge c) \vee (b \wedge c)$
3. Truth table

abc	q
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

# English language description of a combinational logic function

---

$F(d,c,b,a)$  is true if input  $d,c,b,a$  is prime

# Truth Table

---

$F(d,c,b,a)$  is true if input  $d,c,b,a$  is prime

No	dcba	q
0	0000	0
1	0001	1
2	0010	1
3	0011	1
4	0100	0
5	0101	1
6	0110	0
7	0111	1
8	1000	0
9	1001	0
10	1010	0
11	1011	1
12	1100	0
13	1101	1
14	1110	0
15	1111	0



# Equation

$F(d,c,b,a)$  is true if input  $d,c,b,a$  is prime

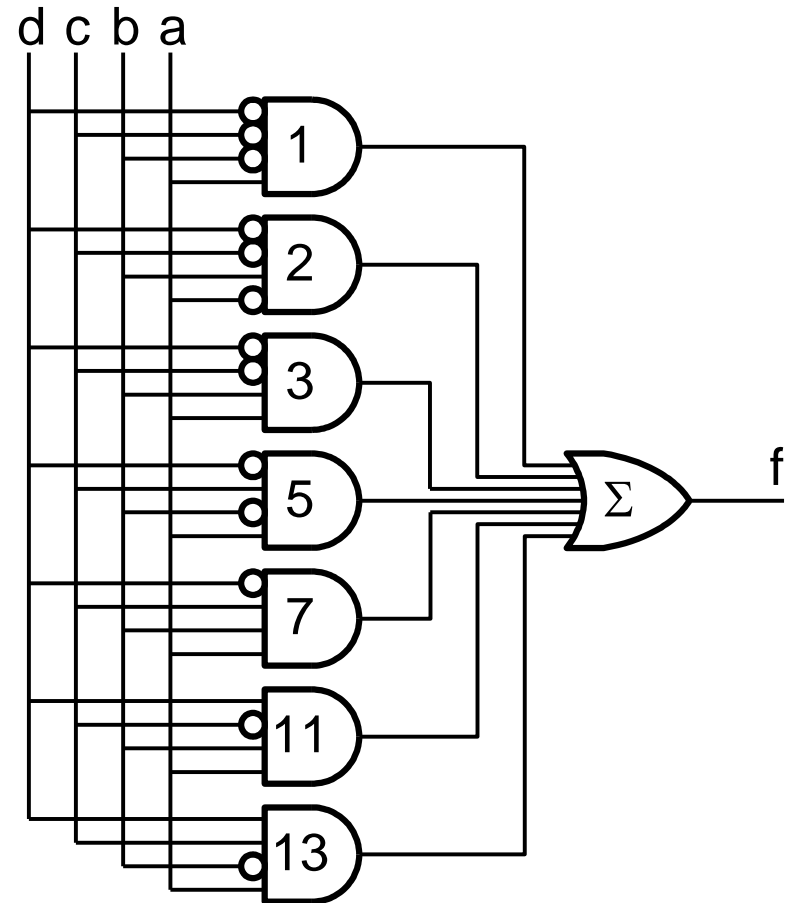
$$f = \sum_{dcba} m(1,2,3,5,7,11,13)$$

No	dcba	q
0	0000	0
1	0001	1
2	0010	1
3	0011	1
4	0100	0
5	0101	1
6	0110	0
7	0111	1
8	1000	0
9	1001	0
10	1010	0
11	1011	1
12	1100	0
13	1101	1
14	1110	0
15	1111	0

# Schematic Logic Diagram

Equation:

$$f = \sum_{dcba} m(1,2,3,5,7,11,13)$$



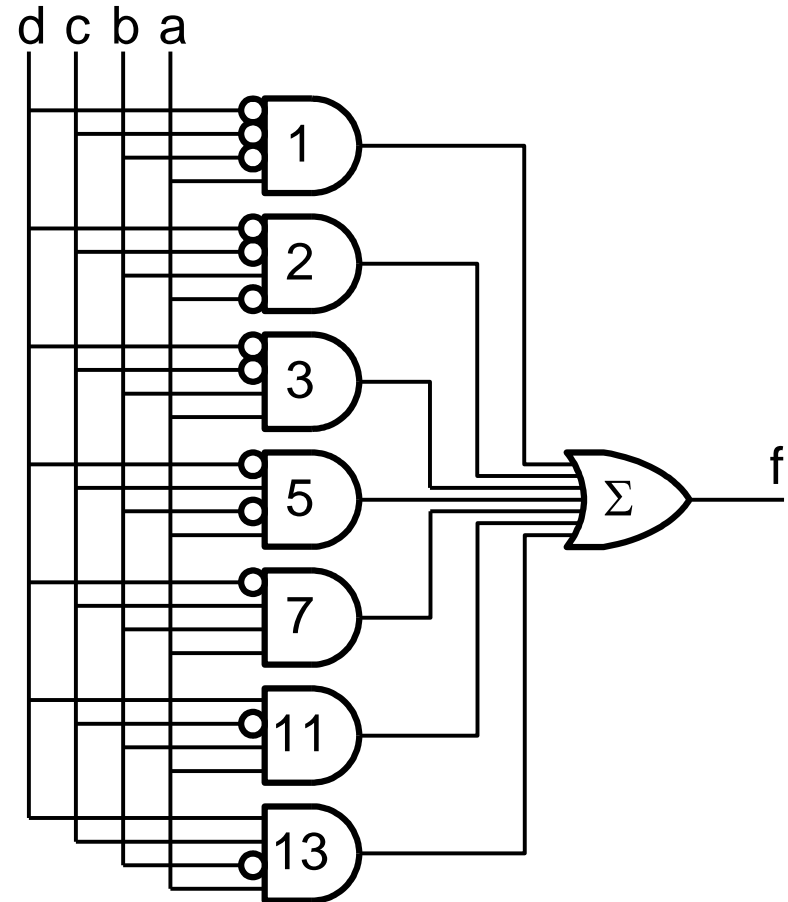
---

# Combinational Logic Design – Karnaugh map

# Schematic Logic Diagram

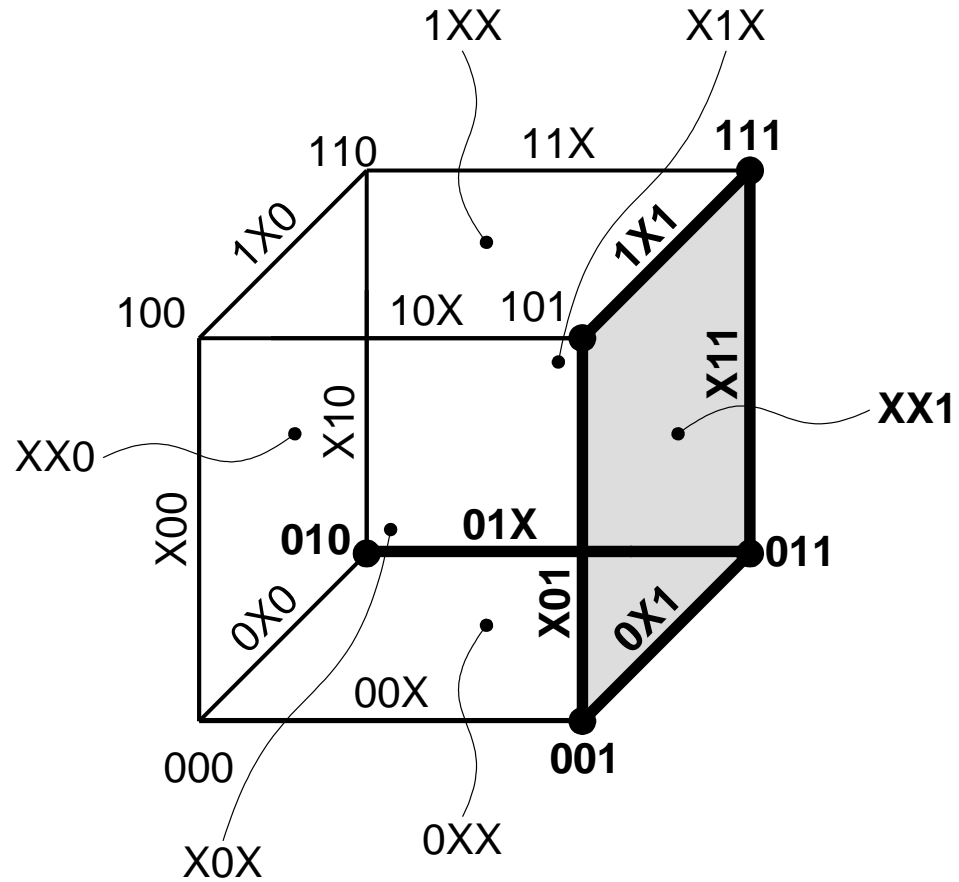
Equation:

$$f = \sum_{dcba} m(1,2,3,5,7,11,13)$$



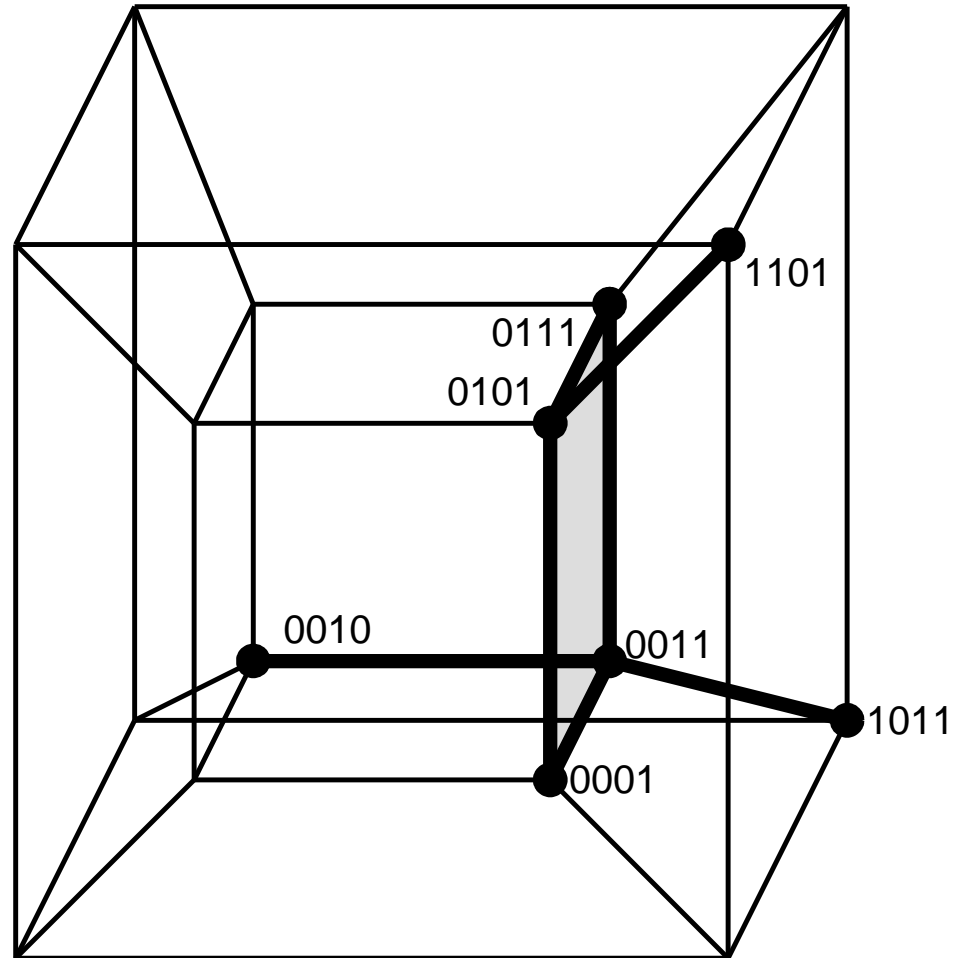
# Cube representation (3-bit prime)

$$f = \sum_{cba} m(1,2,3,5,7)$$

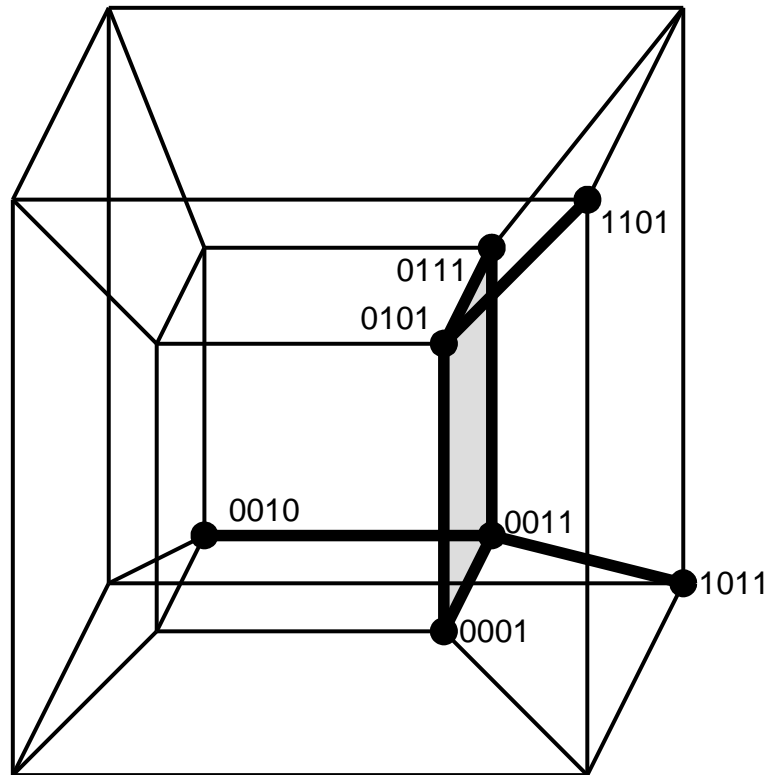


# 4-D Hypercube (4-bit prime)

$$f = \sum_{dcba} m(1,2,3,5,7,11,13)$$



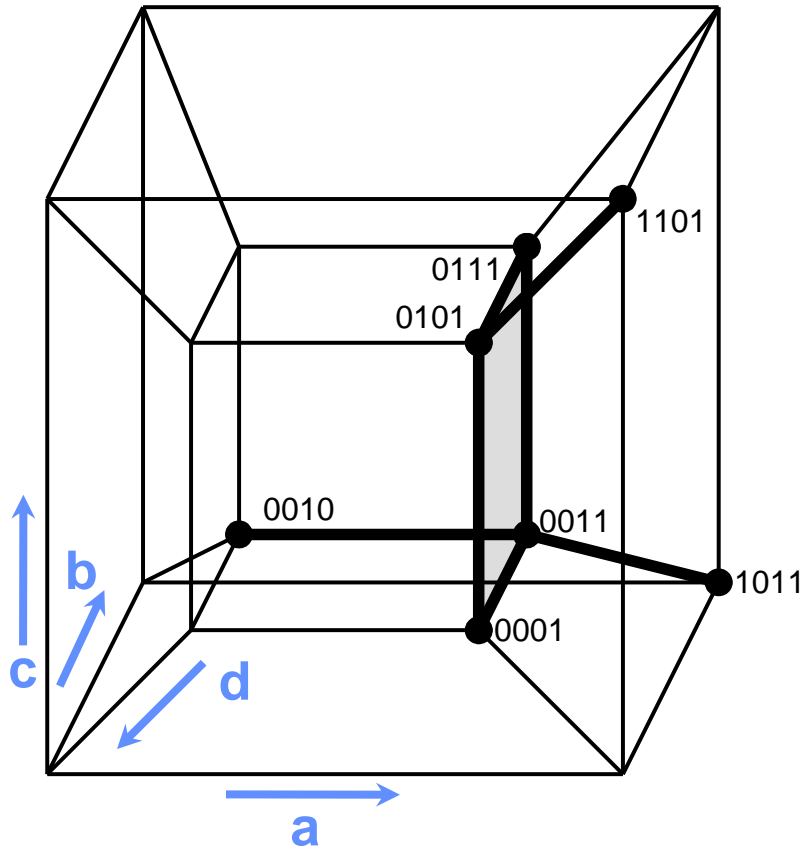
# 4-bit Prime Number Function (계속)



dcba
0xx1
001x
x101
x011

$$f = (a \wedge \bar{d}) \vee (b \wedge \bar{c} \wedge \bar{d}) \vee (a \wedge \bar{b} \wedge c) \vee (a \wedge b \wedge \bar{c})$$

# Karnaugh Map of 4-bit Prime

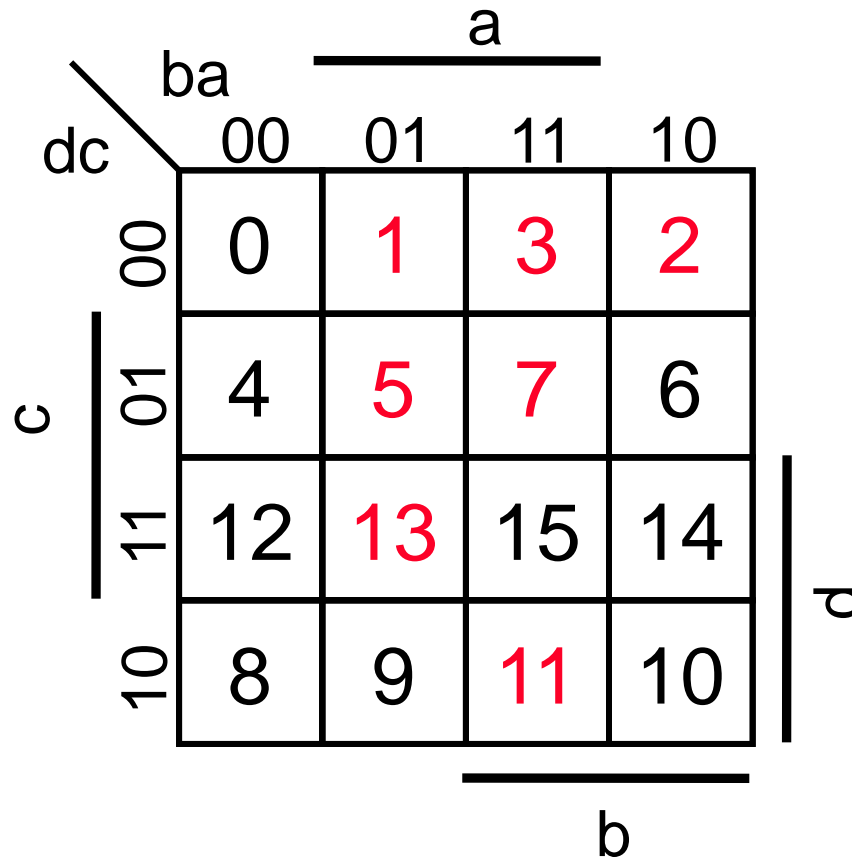


		$a$				
		$ba$	00	01	11	10
$c$	$dc$	00	0 <sub>0</sub>	1 <sub>1</sub>	1 <sub>3</sub>	1 <sub>2</sub>
	01	0 <sub>4</sub>	1 <sub>5</sub>	1 <sub>7</sub>	0 <sub>6</sub>	
	11	0 <sub>12</sub>	1 <sub>13</sub>	0 <sub>15</sub>	0 <sub>14</sub>	
	10	0 <sub>8</sub>	0 <sub>9</sub>	1 <sub>11</sub>	0 <sub>10</sub>	
		$b$				



## Karnaugh Map of 4-bit Prime: Min-terms

---



Position of min-terms on a 4-bit Karnaugh map

---

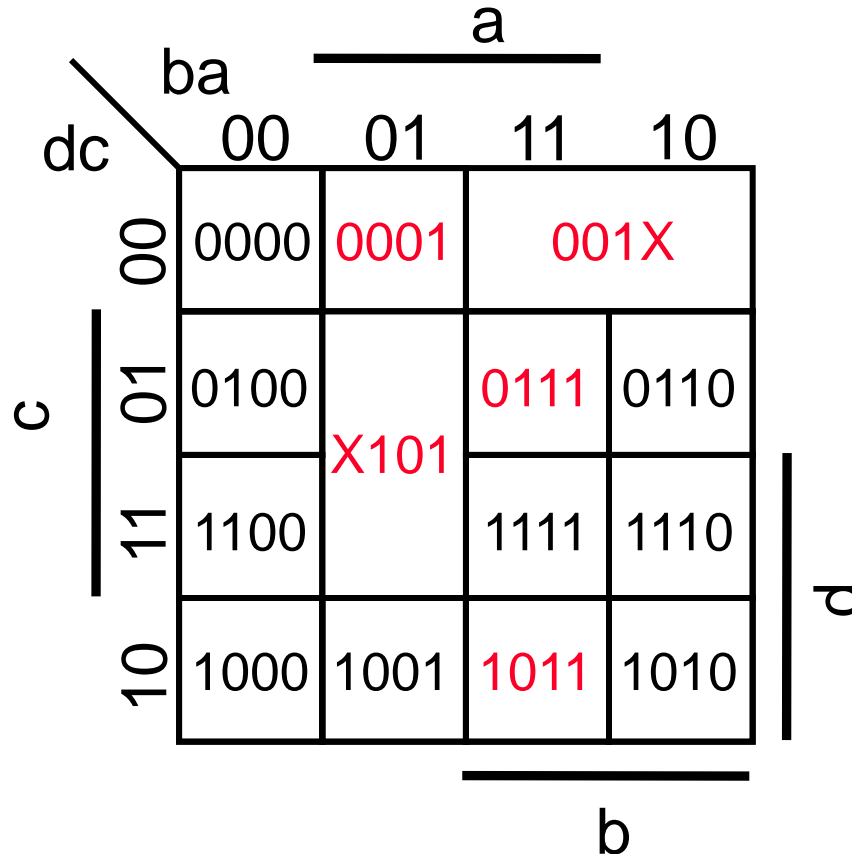
## Karnaugh Map of 4-bit Prime: Implicants

A 4x4 Karnaugh Map for a 4-bit prime function. The map is a grid with columns labeled 'ba' (00, 01, 11, 10) and rows labeled 'dc' (00, 01, 11, 10). The top row is labeled 'a' with a horizontal bar above it. The left side is labeled 'c' with a vertical bar to its left. The right side is labeled 'd' with a vertical bar to its right. The bottom is labeled 'b' with a horizontal bar below it. The cells contain 4-bit binary strings. The following cells are highlighted in red: (00,01), (00,11), (00,10), (01,01), (01,11), (11,01), (11,11), (10,01), and (10,11). These red cells represent the prime implicants of the function.

dc \ ba	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

Adjacent min-terms differ in exactly one bit  
Every positive min-term is an implicant

## Karnaugh Map of 4-bit Prime: Larger Implicants



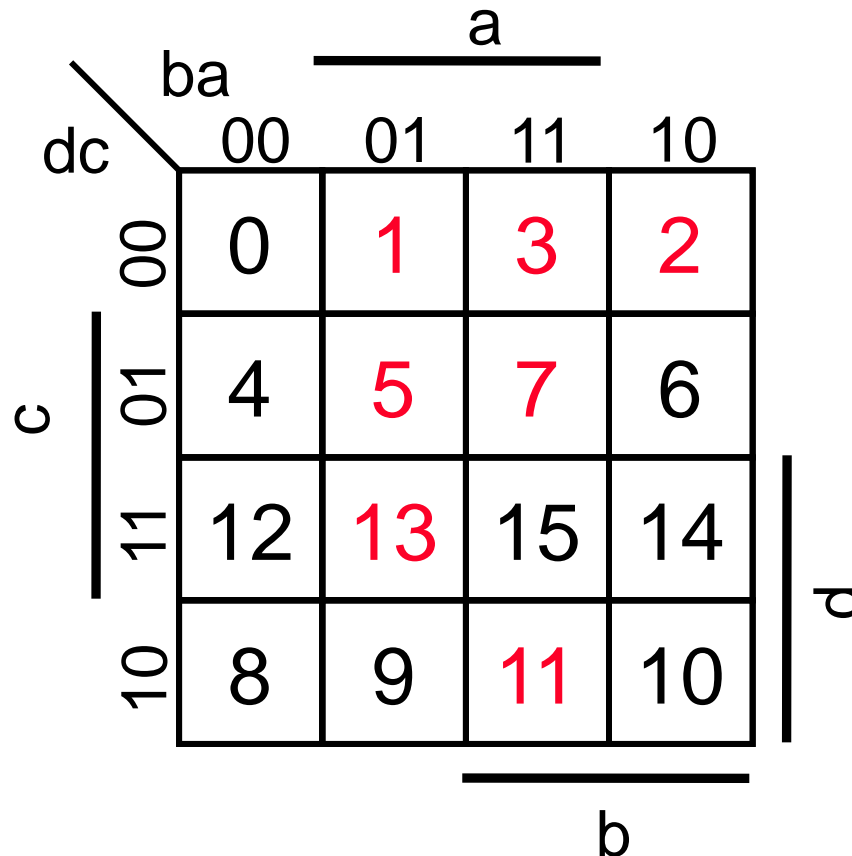
Can combine adjacent min-terms into implicants

---

# Combinational Logic Design - Covering

## Karnaugh Map of 4-bit Prime: Min-terms

---



Position of min-terms on a 4-bit Karnaugh map

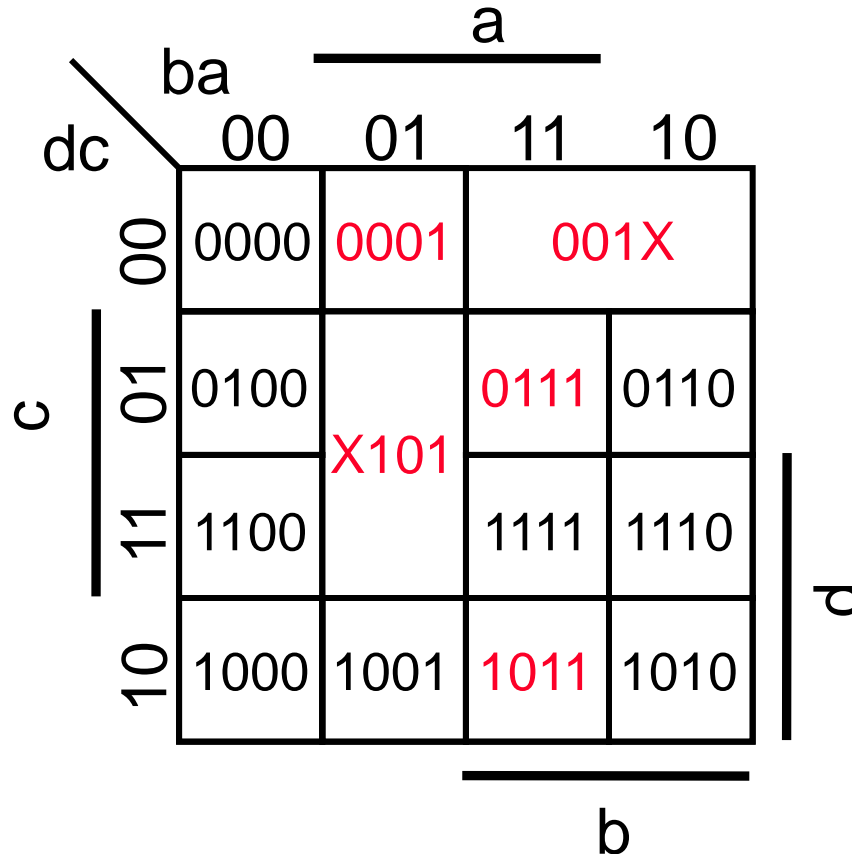
## Karnaugh Map of 4-bit Prime: Implicants

A 4x4 Karnaugh map for a 4-bit prime function. The map is a grid with columns labeled 'ba' (00, 01, 11, 10) and rows labeled 'dc' (00, 01, 11, 10). The cells contain 4-bit binary strings. The following cells are highlighted in red: (00,01), (00,11), (00,10), (01,01), (01,11), (11,01), (11,11), (10,01), and (10,11). The map is annotated with lines: a horizontal line above the top row labeled 'a', a vertical line to the left of the first column labeled 'c', a vertical line to the right of the last column labeled 'd', and a horizontal line below the bottom row labeled 'b'.

dc \ ba	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

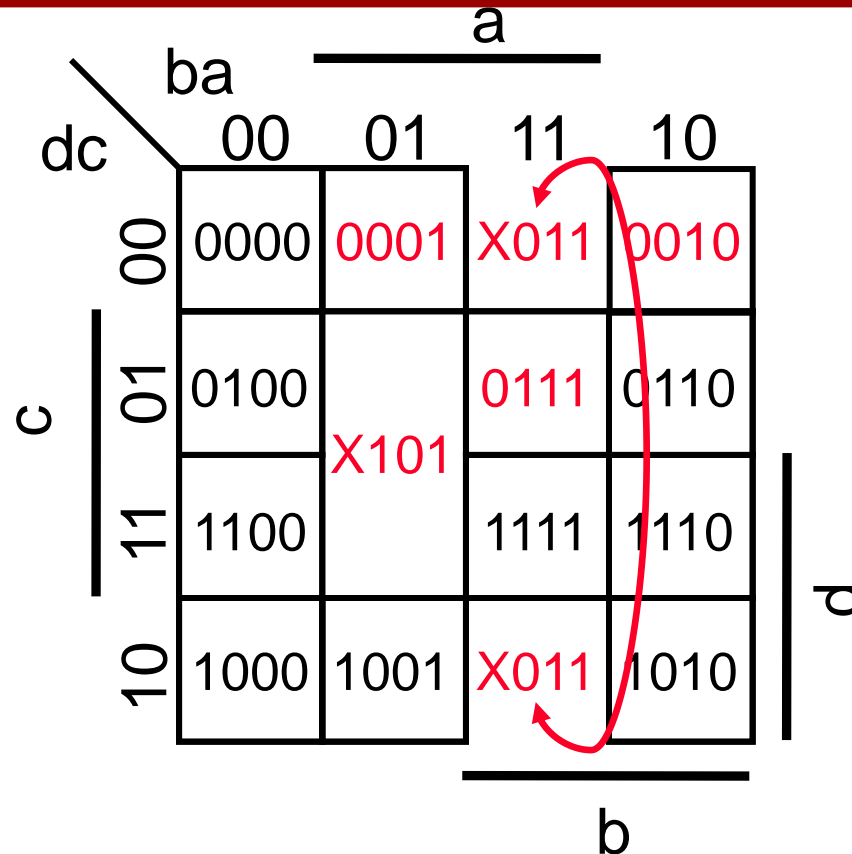
Adjacent min-terms differ in exactly one bit  
Every positive min-term is an implicant

## Karnaugh Map of 4-bit Prime: Larger Implicants



Can combine adjacent min-terms into implicants

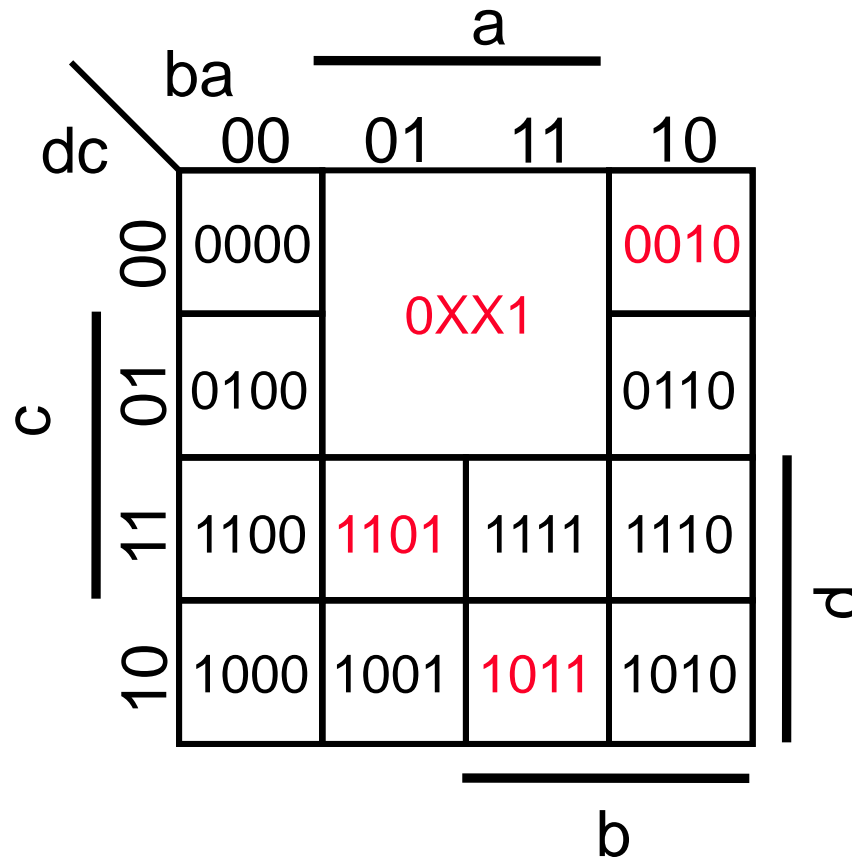
## Karnaugh Map of 4-bit Prime: Adjacency



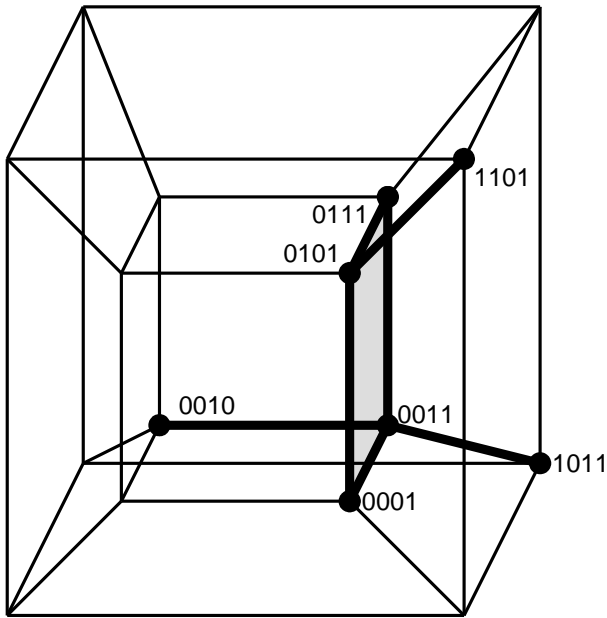
Can combine adjacent min-terms into implicants  
Note edges wrap around



# Karnaugh Map of 4-bit Prime: 0XX1



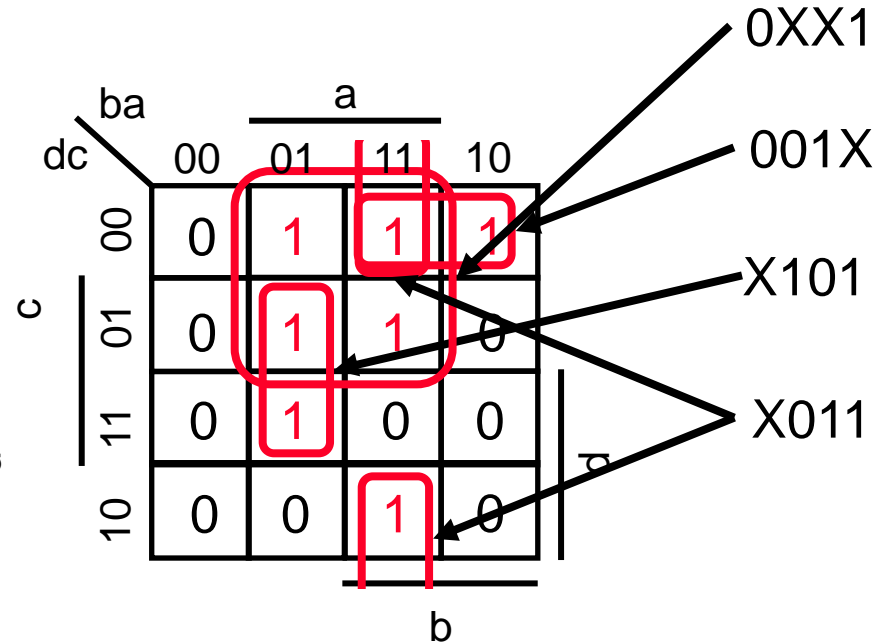
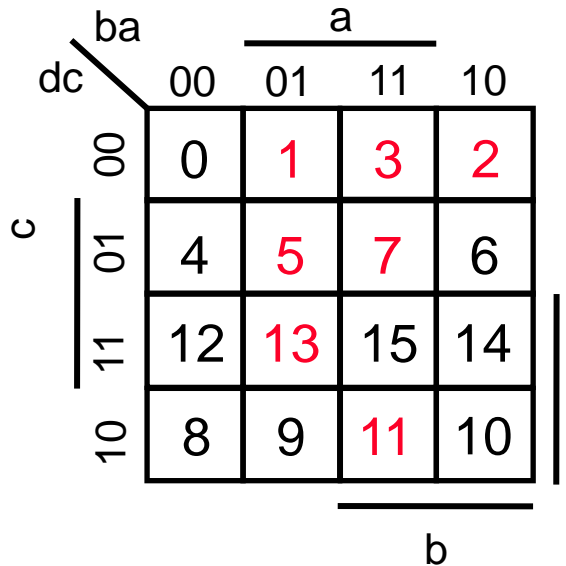
A larger implicant



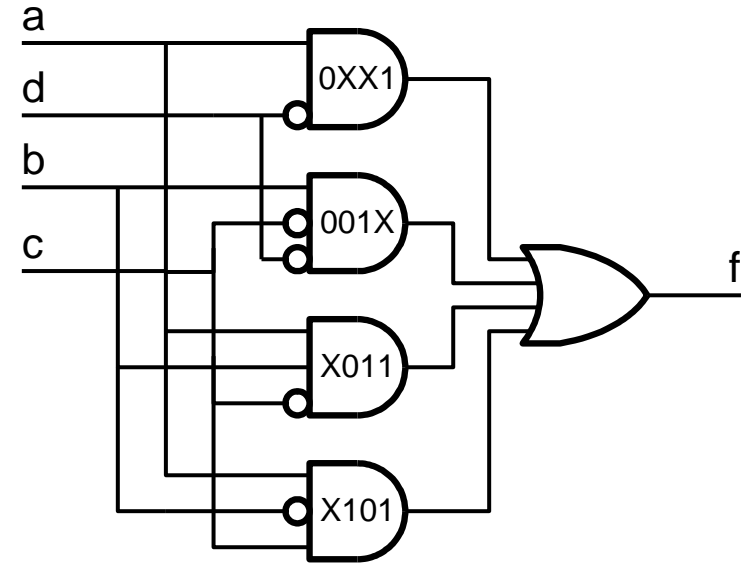
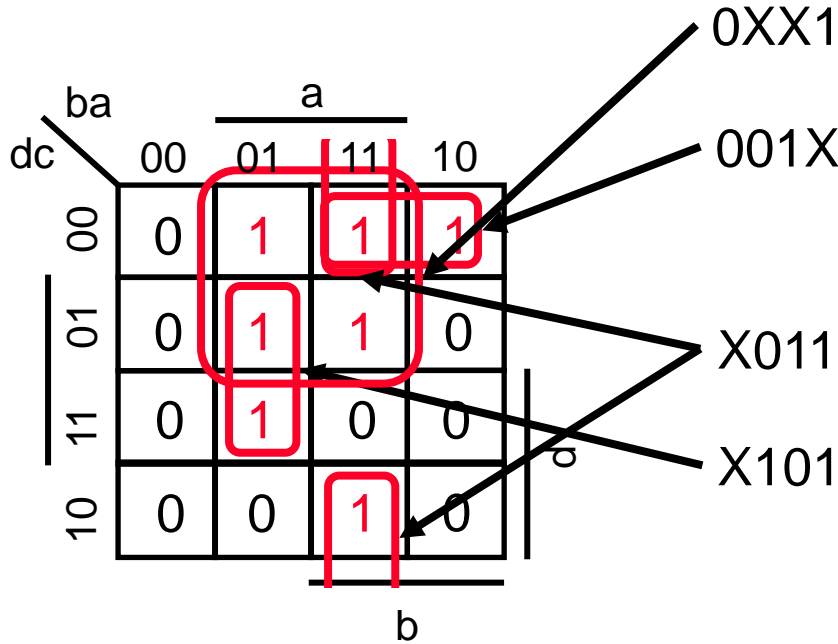
Why make implicants overlap?

Two reasons:

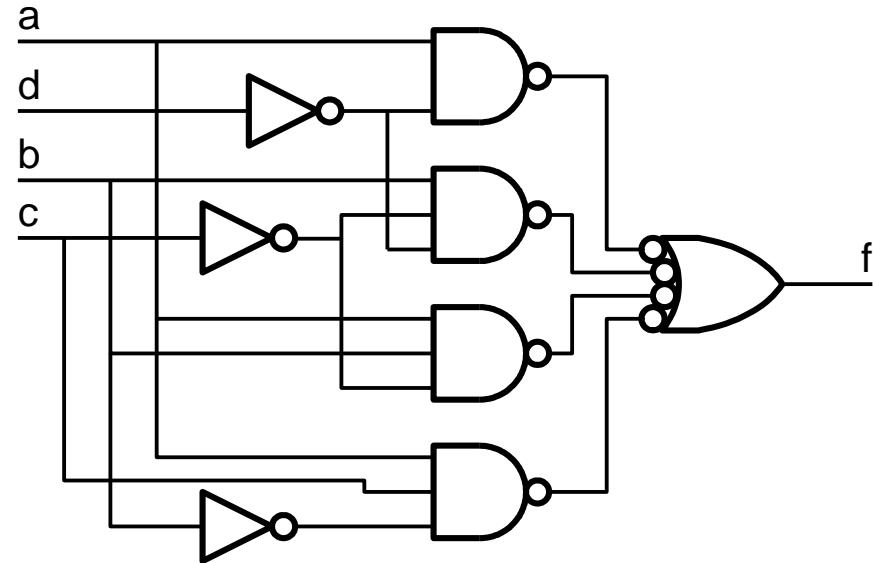
1. Larger implicants have fewer gates, or gates with fewer inputs.
2. Hazards. More on them later.



dcba
0xx1
001x
x101
x011



In practice, CMOS gates are always inverting, so the real circuit might look like this



# Summary

---

- Studied synthesizing manually a combinational logic.
  - English-like description → truth table → Karnaugh map → covering
  - Essential primes → minimal number of primes
- CAD synthesis tools produce logic circuits that are better than the ones a typical designer could generate manually.
  - The synthesis program considers multi-level circuits and implementations that make use of special cells in the library, and can try thousands of combinations before picking the best one.
  - Please spend more time on a clever high-level organization of the system.

---

# Combinational Logic Design – Incompletely specified functions

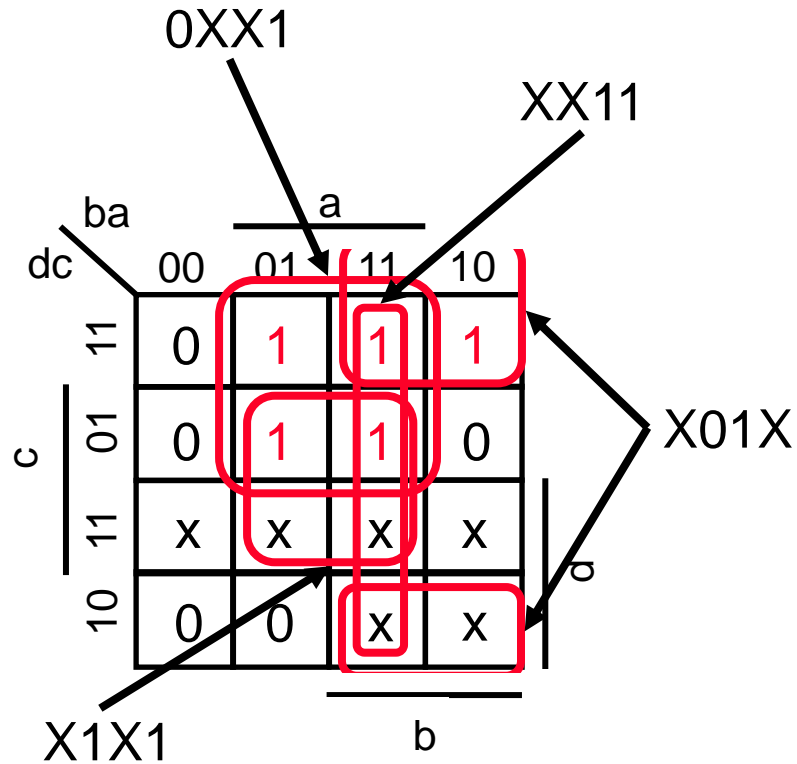
# Decimal prime number function – includes don't cares

---

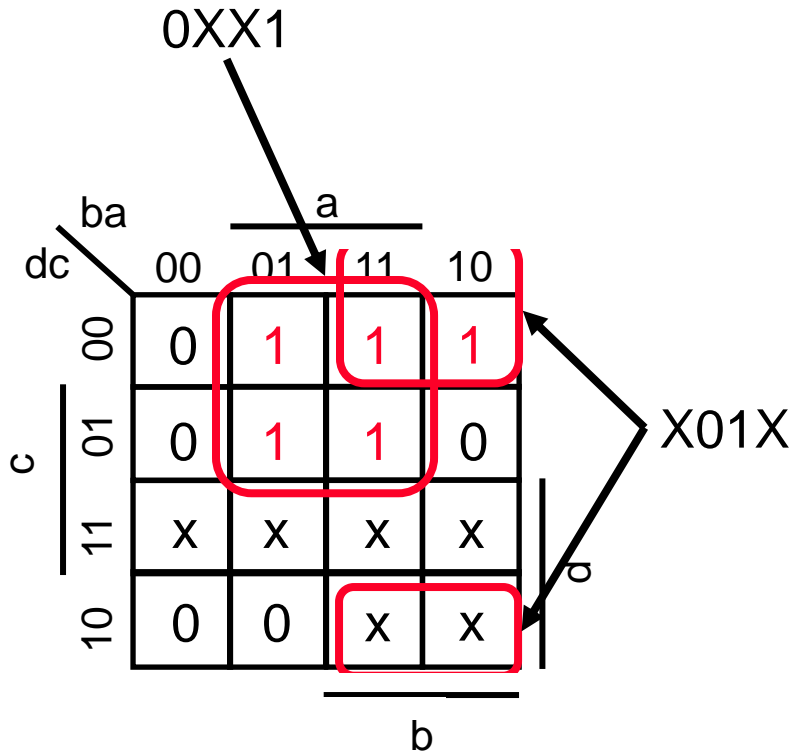
$$f = \sum_{dcba} m(1,2,3,5,7) + D(10,11,12,13,14,15)$$

		ba		a	
		00	01	11	10
c	dc	00	01	11	10
	00	0	1	1	1
	01	0	1	1	0
	11	x	x	x	x
10	0	0	x	x	
		b			

# Decimal prime number function K-Map

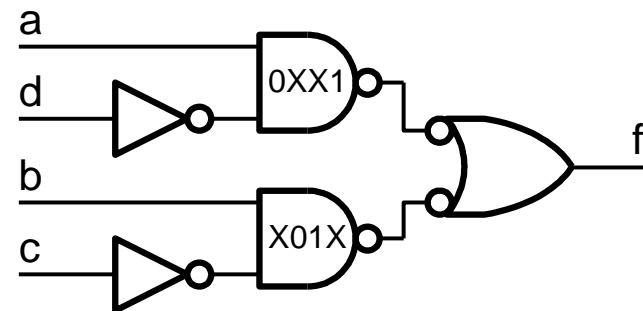


# Decimal prime number function – circuit



Cover:  
0XX1  
X01X

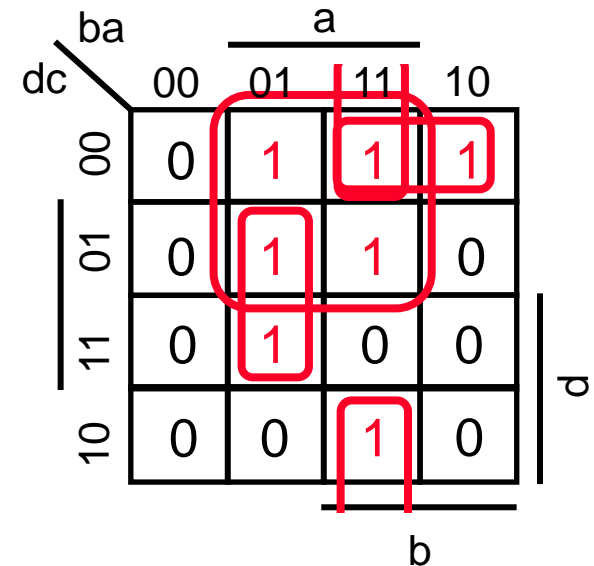
$$f = (a \wedge \bar{d}) \vee (b \wedge \bar{c})$$





# Revisiting definitions

- Min-term: a product term that includes each input of a circuit or its complement.
- Implicant: a product term that if true implies the function is true.
- Prime Implicant: is an implicant that cannot be made any larger and still be an implicant.
- Essential Prime Implicant: the only prime implicant that contains a particular min-term of the function.



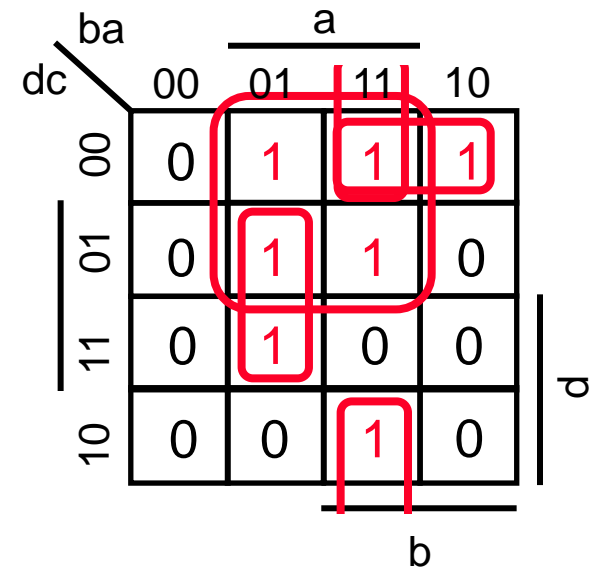
---

# Combinational Logic Design

## – Product-of-sums

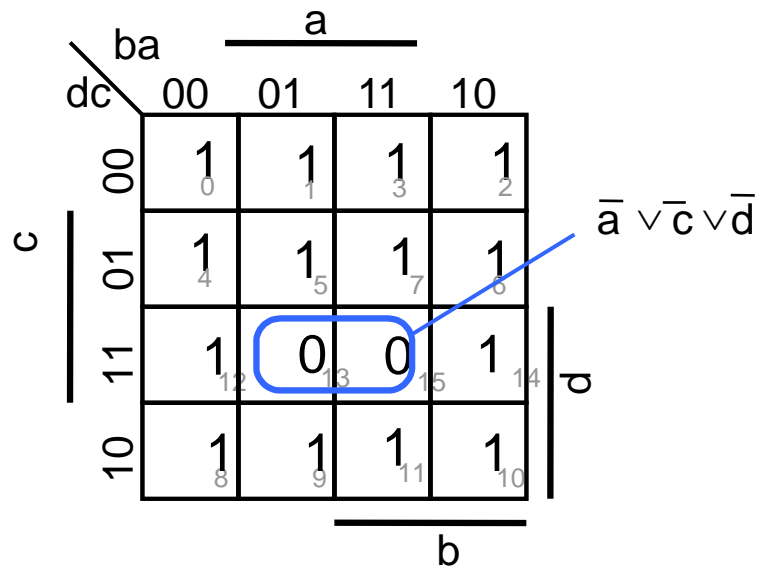
# Revisiting definitions

- Min-term: a product term that includes each input of a circuit or its complement.
- Implicant: a product term that if true implies the function is true.
- Prime Implicant: is an implicant that cannot be made any larger and still be an implicant.
- Essential Prime Implicant: the only prime implicant that contains a particular min-term of the function.

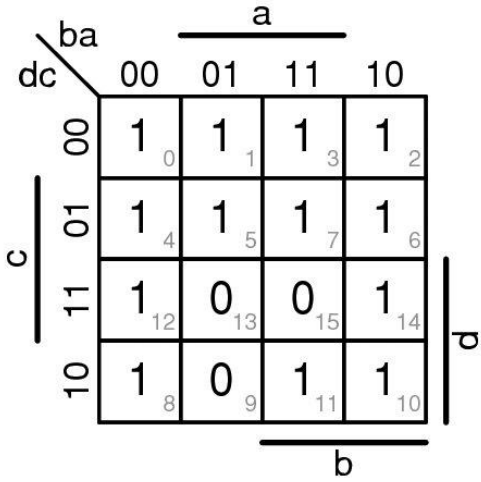


# Product-of-Sums Implementation

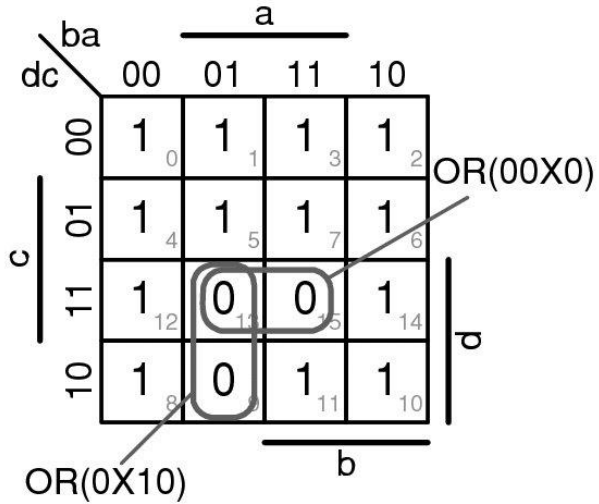
- Sum-of-Products circuit: focus on inputs states where truth table is a 1.
- Product-of-Sums: focus on input states where truth table is a 0.



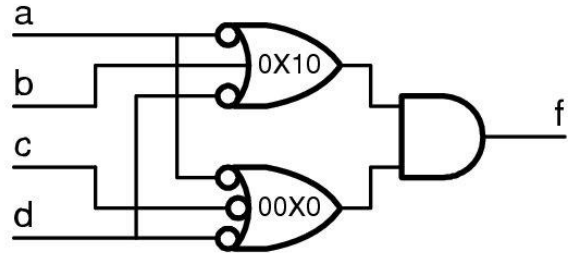
# Example 1: Prime



(a)

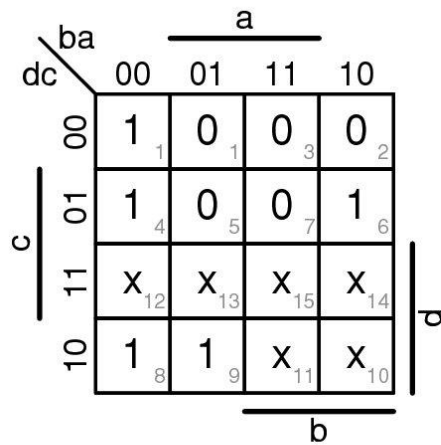


(b)

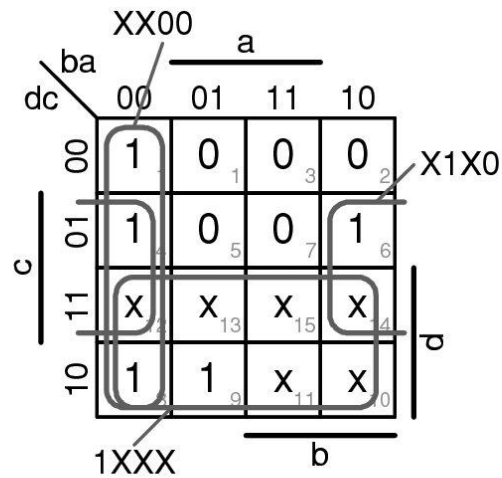


(c)

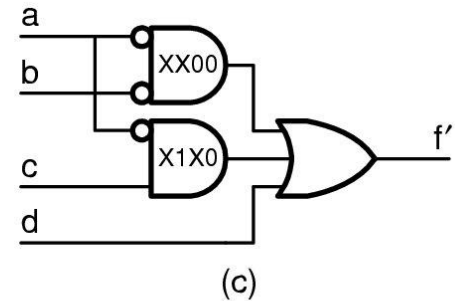
# Example 2: Decimal Prime



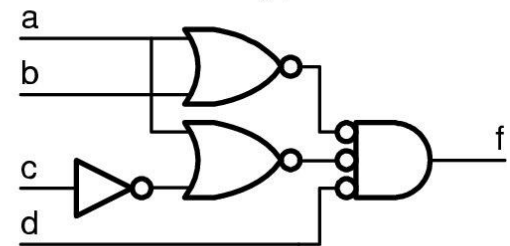
(a)



(b)



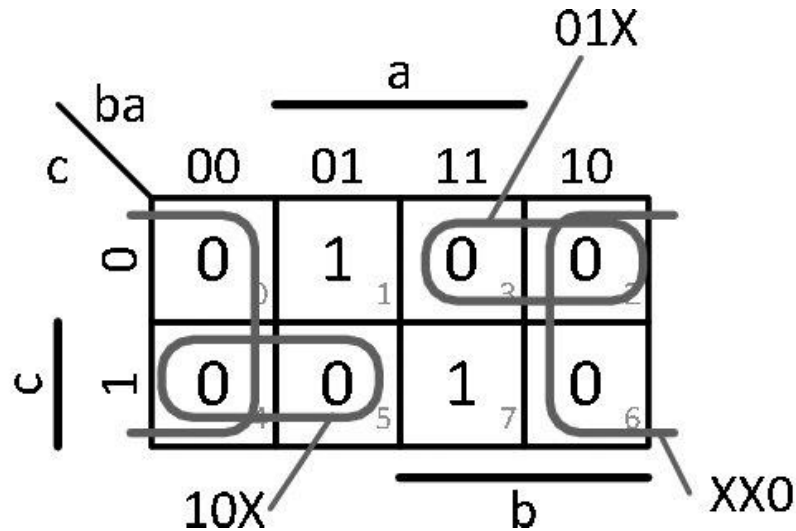
(c)



(d)

## Example 3

Express the three-input function  $f = \sum m(1,7)$  as a minimal product-of-sums expression.

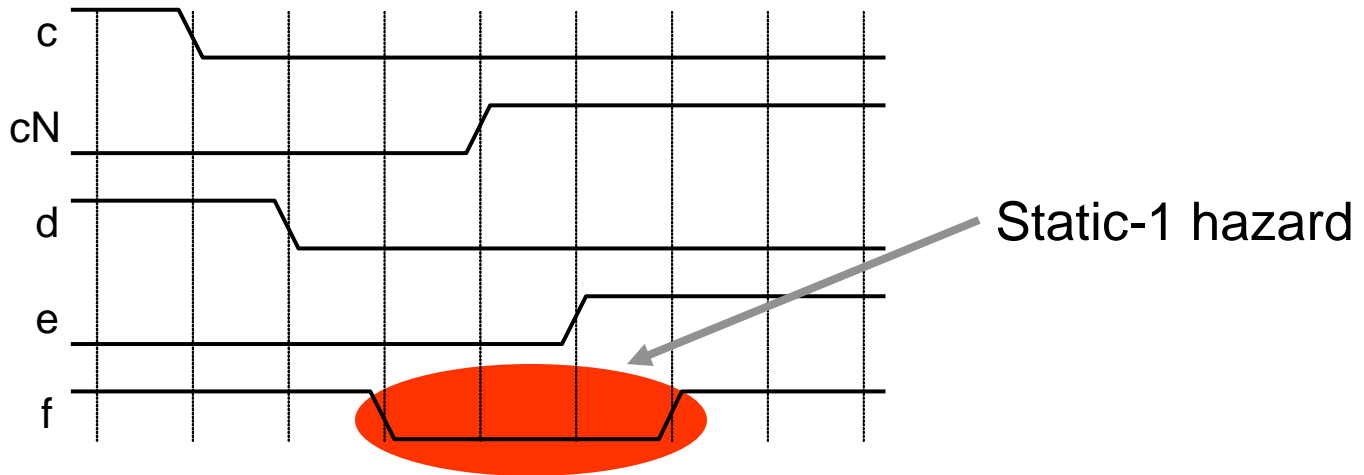
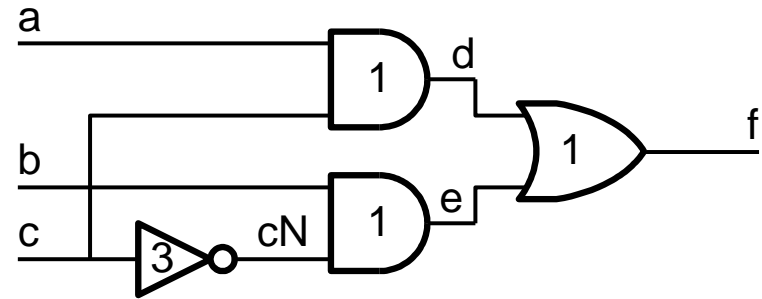
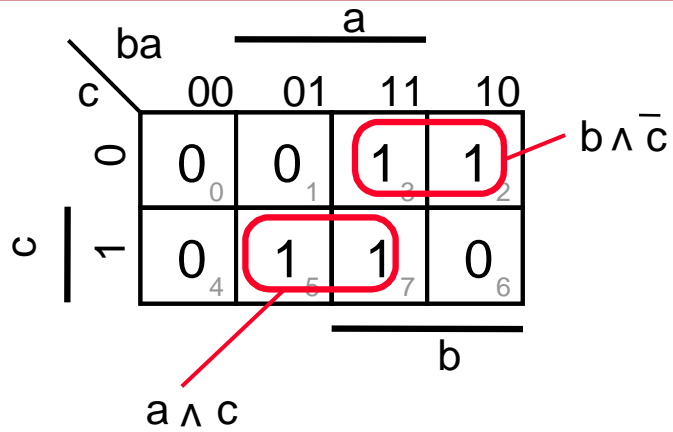


---

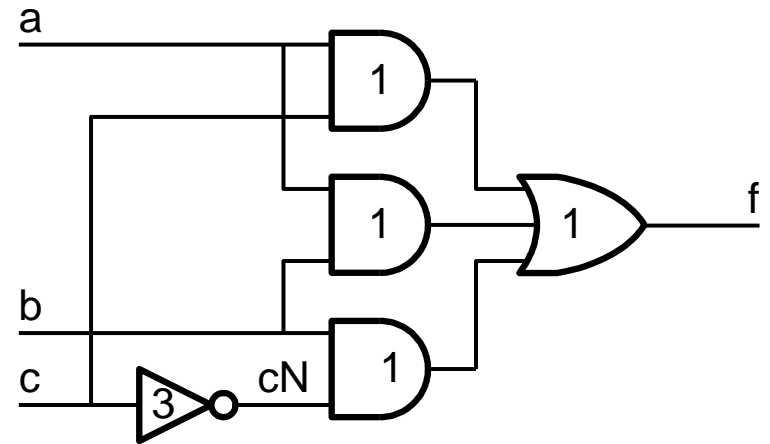
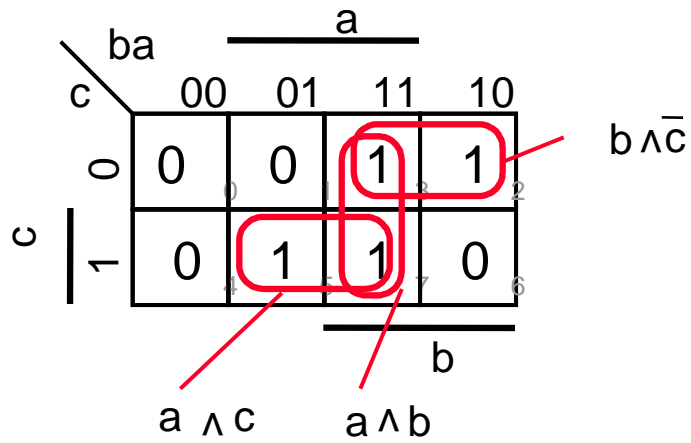
# Combinational Logic Design - Hazards



# Hazards



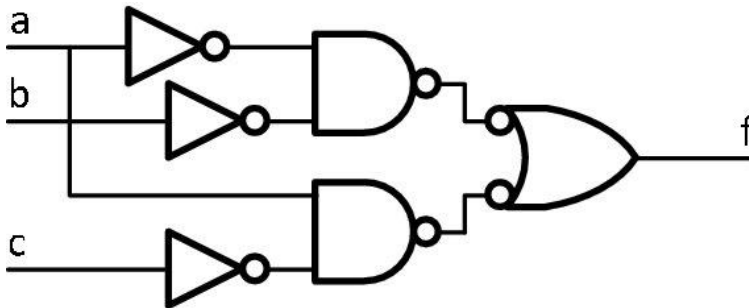
# Cover transitions to eliminate hazards



# Example

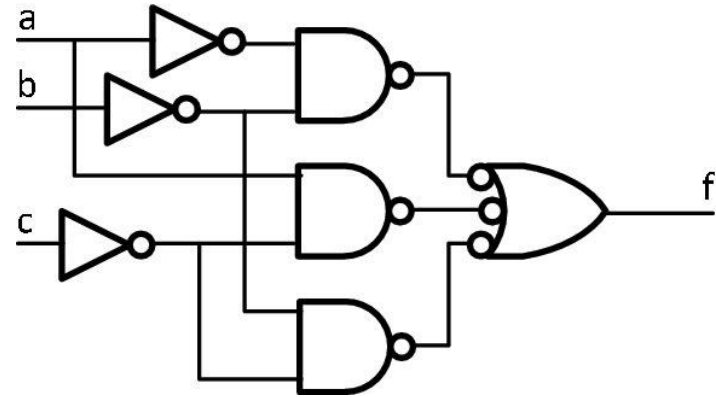
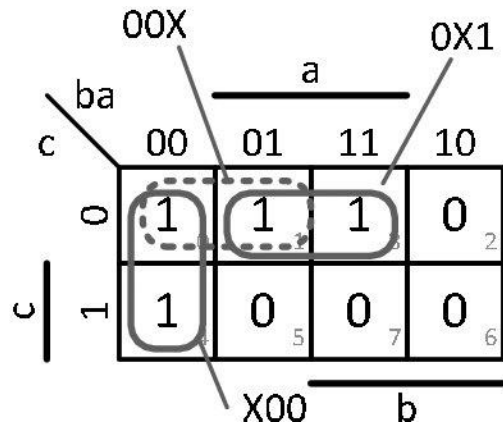
---

Fix the hazard that occurs in the circuit below. That is, preserve the logic function of the circuit while eliminating any hazards that occur during input transitions.



# Solution

---



# Summary

---

- Studied synthesizing manually a combinational logic.
  - English-like description → truth table → Karnaugh map → covering
  - Essential primes → minimal number of primes
  - PoS of a function can be derived from the SoP of its complemented function.
- CAD synthesis tools produce logic circuits that are better than the ones a typical designer could generate manually.
  - The synthesis program considers multi-level circuits and implementations that make use of special cells in the library, and can try thousands of combinations before picking the best one.
  - Please spend more time on a clever high-level organization of the system.