Principes de fonctionnement des machines binaires

2019/2020

Pierluigi Crescenzi
Université de Paris, IRIF

Université de Paris

INSTITUT DE RECHERCHE EN INFORMATIQUE FONDAMENTALE

cnrs dépasser les frontières

- Tests et examens
  - CC : résultat des tests en TD / TP (semaine 4 et **cette semaine ou la prochaine**)
  - E0 : partiel (samedi 26 octobre)
  - E1 : examen (19 décembre de 8h30 à 11h:30)
    - Examen écrit
  - E2 : examen fin juin
- Notes finales
  - Note session 1 : 25% CC + 25% E0 + 50% E1
  - Note session 2 : max(E2, 33% CC + 67% E2)
- Rappel
  - Pas de note $\Rightarrow$ pas de moyenne $\Rightarrow$ pas de semestre
- Site web
  - `moodlesupd.script.univ-paris-diderot.fr`

- Numération et arithmétique
- Numération et arithmétique en machine
- Numérisation et codage (texte, images)
- Compression, cryptographie, contrôle d'erreur
- Logique et calcul propositionnel
- **Circuits numériques**

- Questions
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes et et non

- Questions
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes et et non
    - $a \vee b \Leftrightarrow \neg(\neg a \wedge \neg b)$ (De Morgan)

a	b	a	$\vee$	b	$\neg$	$\neg a$	$\wedge$	$\neg b$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$
$\top$	$\bot$	$\top$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$

- Questions
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes et et non
    - $a \vee b \Leftrightarrow \neg(\neg a \wedge \neg b)$ (De Morgan)
    - $\neg((a \vee b) \wedge c) \Leftrightarrow \neg(\neg(\neg a \wedge \neg b) \wedge c)$

- Questions
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes et et non
    - $a \vee b \Leftrightarrow \neg(\neg a \wedge \neg b)$ (De Morgan)
    - $\neg((a \vee b) \wedge c) \Leftrightarrow \neg(\neg(\neg a \wedge \neg b) \wedge c)$
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes non-ou ($p \downarrow q = \top$ ssi $p = \bot$ et $q = \bot$)

- Questions
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes et et non
    - $a \vee b \Leftrightarrow \neg(\neg a \wedge \neg b)$ (De Morgan)
    - $\neg((a \vee b) \wedge c) \Leftrightarrow \neg(\neg(\neg a \wedge \neg b) \wedge c)$
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes non-ou ($p \downarrow q = \top$ ssi $p = \bot$ et $q = \bot$)
    - $\neg a \Leftrightarrow a \downarrow a$

a	$\neg a$	a	$\downarrow$	a
$\bot$	$\top$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

- Questions
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes et et non
    - $a \vee b \Leftrightarrow \neg(\neg a \wedge \neg b)$ (De Morgan)
    - $\neg((a \vee b) \wedge c) \Leftrightarrow \neg(\neg(\neg a \wedge \neg b) \wedge c)$
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes non-ou ($p \downarrow q = \top$ ssi $p = \bot$ et $q = \bot$)
    - $\neg a \Leftrightarrow a \downarrow a$
    - $a \vee b \Leftrightarrow (a \downarrow b) \downarrow (a \downarrow b)$

a	b	a	$\vee$	b	a	$\downarrow$	b	$\downarrow$	a	$\downarrow$	b
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\bot$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\top$

- Questions
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes et et non
    - $a \vee b \Leftrightarrow \neg(\neg a \wedge \neg b)$ (De Morgan)
    - $\neg((a \vee b) \wedge c) \Leftrightarrow \neg(\neg(\neg a \wedge \neg b) \wedge c)$
  - Réaliser le circuit de la fonction $\neg((a \vee b) \wedge c)$ en utilisant que des portes non-ou ($p \downarrow q = \top$ ssi $p = \bot$ et $q = \bot$)
    - $\neg a \Leftrightarrow a \downarrow a$
    - $a \vee b \Leftrightarrow (a \downarrow b) \downarrow (a \downarrow b)$
    - $a \wedge b \Leftrightarrow (a \downarrow a) \downarrow (b \downarrow b)$

a	b	a	$\wedge$	b	a	$\downarrow$	a	$\downarrow$	b	$\downarrow$	b
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\bot$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\top$

- Question
  - $\neg((a \vee b) \wedge c)$ en utilisant que des portes non-ou

- $\neg a \Leftrightarrow a \downarrow a$
- $a \vee b \Leftrightarrow (a \downarrow b) \downarrow (a \downarrow b)$
- $a \wedge b \Leftrightarrow (a \downarrow a) \downarrow (b \downarrow b)$

- Question
  - $\neg((a \vee b) \wedge c)$ en utilisant que des portes non-ou

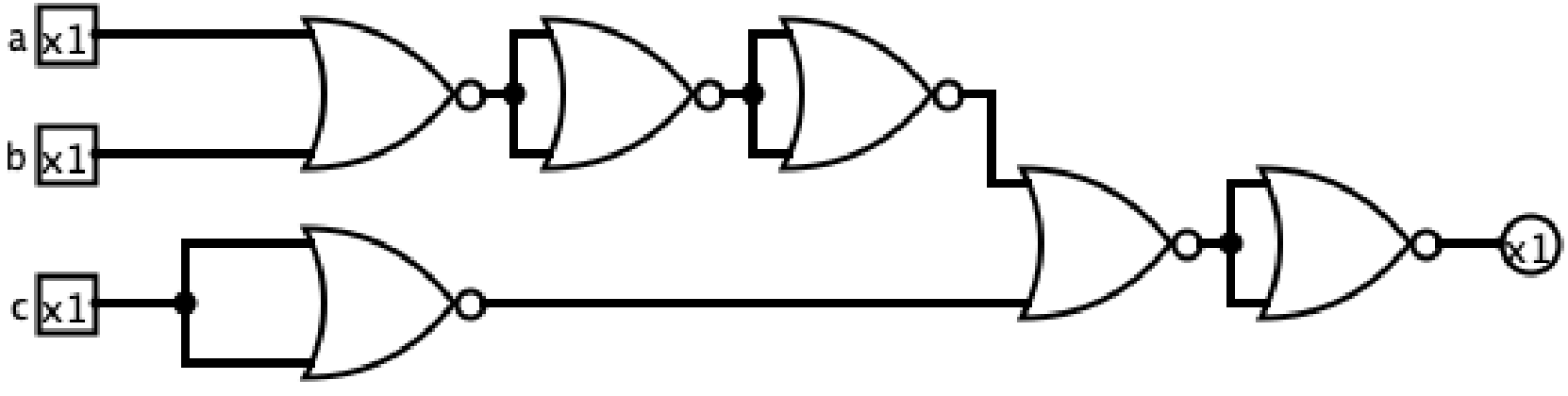
- $\neg a \Leftrightarrow a \downarrow a$
- $a \vee b \Leftrightarrow (a \downarrow b) \downarrow (a \downarrow b)$
- $a \wedge b \Leftrightarrow (a \downarrow a) \downarrow (b \downarrow b)$

$\neg((a \vee b) \wedge c) \Leftrightarrow$
$\neg(((a \downarrow b) \downarrow (a \downarrow b)) \wedge c) \Leftrightarrow$
$\neg((((a \downarrow b) \downarrow (a \downarrow b)) \downarrow ((a \downarrow b) \downarrow (a \downarrow b))) \downarrow (c \downarrow c)) \Leftrightarrow$
$((((a \downarrow b) \downarrow (a \downarrow b)) \downarrow ((a \downarrow b) \downarrow (a \downarrow b))) \downarrow (c \downarrow c)) \downarrow ((((a \downarrow b) \downarrow (a \downarrow b)) \downarrow ((a \downarrow b) \downarrow (a \downarrow b))) \downarrow (c \downarrow c))$

- Question
  - $\neg((a \vee b) \wedge c)$ en utilisant que des portes non-ou

- $\neg a \Leftrightarrow a \downarrow a$
- $a \vee b \Leftrightarrow (a \downarrow b) \downarrow (a \downarrow b)$
- $a \wedge b \Leftrightarrow (a \downarrow a) \downarrow (b \downarrow b)$

$\neg((a \vee b) \wedge c) \Leftrightarrow$
$\neg(((a \downarrow b) \downarrow (a \downarrow b)) \wedge c) \Leftrightarrow$
$\neg((((a \downarrow b) \downarrow (a \downarrow b)) \downarrow ((a \downarrow b) \downarrow (a \downarrow b))) \downarrow (c \downarrow c)) \Leftrightarrow$
$((((a \downarrow b) \downarrow (a \downarrow b)) \downarrow ((a \downarrow b) \downarrow (a \downarrow b))) \downarrow (c \downarrow c)) \downarrow ((((a \downarrow b) \downarrow (a \downarrow b)) \downarrow ((a \downarrow b) \downarrow (a \downarrow b))) \downarrow (c \downarrow c))$

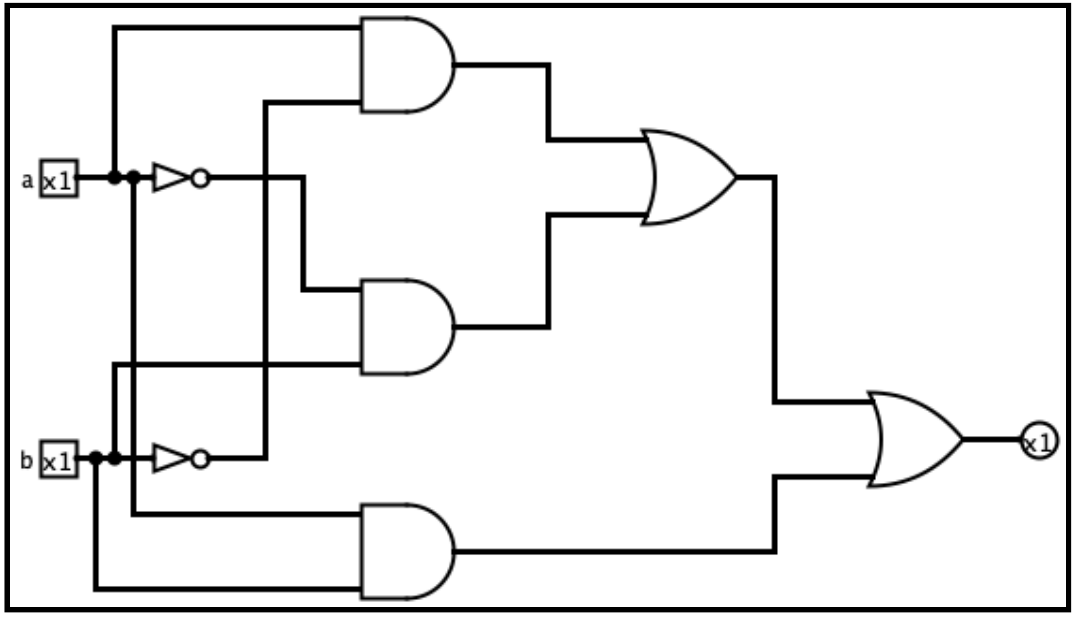


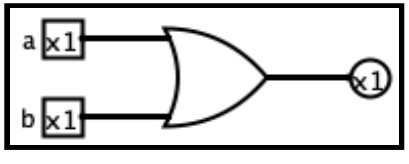
- Puisqu'il existe différentes réalisations, on peut se poser la question du coût ?

- Puisqu'il existe différentes réalisations, on peut se poser la question du coût ?
  - Par exemple, peut-on minimiser le nombre de portes ?

- Puisqu'il existe différentes réalisations, on peut se poser la question du coût ?
  - Par exemple, peut-on minimiser le nombre de portes ?
  - Nous avons déjà donné quelques propriétés des connecteurs

- Puisqu'il existe différentes réalisations, on peut se poser la question du coût ?
  - Par exemple, peut-on minimiser le nombre de portes ?
  - Nous avons déjà donné quelques propriétés des connecteurs
  - Ces propriétés permet de simplifier un circuit
    - Comparons les deux





- En appliquant les propriétés déjà données on peut donc simplifier une expression (et donc un circuit)
  - Mais dans quel ordre doit-on appliquer les règles ?
    - La **méthode de Karnaugh** permet de simplifier les expressions lorsque la fonction ne possède pas trop de variables
    - C'est une méthode visuelle/graphique
    - L'idée de base est d'utiliser dans les mintermes de la FND les propriétés
      - $a \vee \neg a \Leftrightarrow \top$
      - $(a \wedge b_1 \wedge \cdots b_k) \vee (\neg a \wedge b_1 \wedge \cdots \wedge b_k) \Leftrightarrow (b_1 \wedge \cdots \wedge b_k)$
    - Cela conduit à éliminer ces instances de la variable a dans l'expression et donc diminuer la taille du circuit

- Il faut fabriquer un tableau de Karnaugh
  - C'est la représentation d'une fonction booléenne dans laquelle les valuations sont ordonnées de sorte que d'une ligne/colonne à l'autre une seule variable change de valeur
  - C'est possible ça ?
    - Oui, c'est le **code de Gray** ou *code binaire réfléchi*

Code 1b

0

1

Code 1b	Symétrise
0	0
1	1
	1
	0

Code 1b	Symétrise	Rajoute 0/1 Code 2b
0	0	**00**
1	1	**01**
	1	**11**
	0	**10**

Rajoute 0/1

Code 1b	Symétrise	Code 2b	Symétrise
0	0	**00**	00
1	1	**01**	01
	1	**11**	11
	0	**10**	10
			10
			11
			01
			00

Code 1b	Symétrise	Rajoute 0/1 Code 2b	Symétrise	Rajoute 0/1 Code 3b
0	0	**00**	00	**000**
1	1	**01**	01	**001**
	1	**11**	11	**011**
	0	**10**	10	**010**
			10	**110**
			11	**111**
			01	**101**
			00	**100**

Code 1b	Symétrise	Rajoute 0/1 Code 2b	Symétrise	Rajoute 0/1 Code 3b	Symétrise
0	0	**00**	00	**000**	000
1	1	**01**	01	**001**	001
	1	**11**	11	**011**	011
	0	**10**	10	**010**	010
			10	**110**	110
			11	**111**	111
			01	**101**	101
			00	**100**	100
					100
					101
					111
					110
					010
					011
					001
					000

Code 1b	Symétrise	Rajoute 0/1 Code 2b	Symétrise	Rajoute 0/1 Code 3b	Symétrise	Rajoute 0/1 Code 4b
0	0	**00**	00	**000**	000	**0000**
1	1	**01**	01	**001**	001	**0001**
	1	**11**	11	**011**	011	**0011**
	0	**10**	10	**010**	010	**0010**
			10	**110**	110	**0110**
			11	**111**	111	**0111**
			01	**101**	101	**0101**
			00	**100**	100	**0100**
					100	**1100**
					101	**1101**
					111	**1111**
					110	**1110**
					010	**1010**
					011	**1011**
					001	**1001**
					000	**1000**

Code 1b	Symétrise	Rajoute 0/1 Code 2b	Symétrise	Rajoute 0/1 Code 3b	Symétrise	Rajoute 0/1 Code 4b
0	0	**00**	00	**000**	000	**0000**
1	1	**01**	01	**001**	001	**0001**
	1	**11**	11	**011**	011	**0011**
	0	**10**	10	**010**	010	**0010**
			10	**110**	110	**0110**
			11	**111**	111	**0111**
			01	**101**	101	**0101**
			00	**100**	100	**0100**
					100	**1100**
					101	**1101**
					111	**1111**
					110	**1110**
					010	**1010**
					011	**1011**
					001	**1001**
					000	**1000**

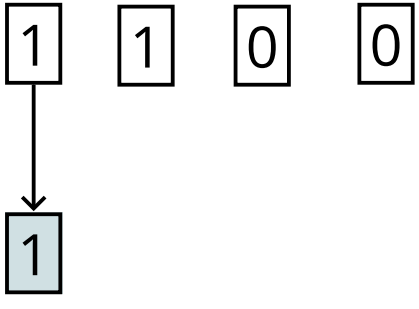
- On remarque que d'une ligne à l'autre un seul bit est modifié et ceci est valable aussi en passant de la dernière ligne à la première
  - Même de la dernière à la premiere
    - Le code de Gray est un tore

- Comment passer de la représentation binaire à celle de Gray ?
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Règle
    - $g_{n-1} = b_{n-1}$
    - $g_i = b_{i+1} \oplus b_i$
  - Exemple : 1100

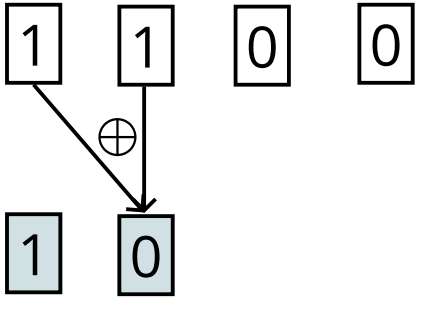
- Comment passer de la représentation binaire à celle de Gray ?
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Règle
    - $g_{n-1} = b_{n-1}$
    - $g_i = b_{i+1} \oplus b_i$
  - Exemple : 1100

1	1	0	0

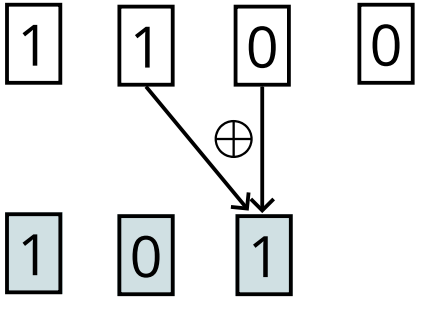
- Comment passer de la représentation binaire à celle de Gray ?
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Règle
    - $g_{n-1} = b_{n-1}$
    - $g_i = b_{i+1} \oplus b_i$
  - Exemple : 1100



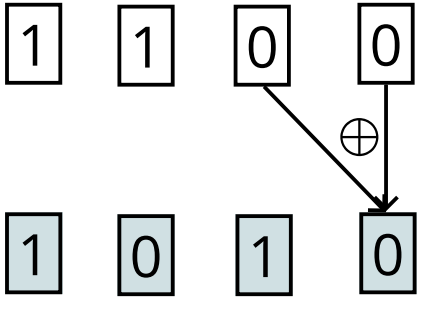
- Comment passer de la représentation binaire à celle de Gray ?
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Règle
    - $g_{n-1} = b_{n-1}$
    - $g_i = b_{i+1} \oplus b_i$
  - Exemple : 1100



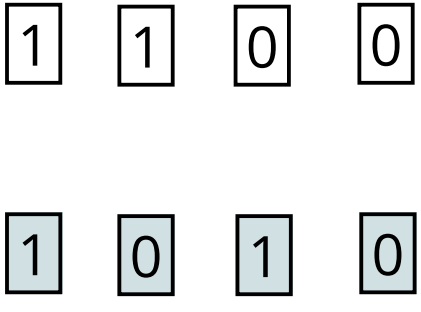
- Comment passer de la représentation binaire à celle de Gray ?
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Règle
    - $g_{n-1} = b_{n-1}$
    - $g_i = b_{i+1} \oplus b_i$
  - Exemple : 1100



- Comment passer de la représentation binaire à celle de Gray ?
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Règle
    - $g_{n-1} = b_{n-1}$
    - $g_i = b_{i+1} \oplus b_i$
  - Exemple : 1100



- Comment passer de la représentation binaire à celle de Gray ?
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Règle
    - $g_{n-1} = b_{n-1}$
    - $g_i = b_{i+1} \oplus b_i$
  - Exemple : 1100



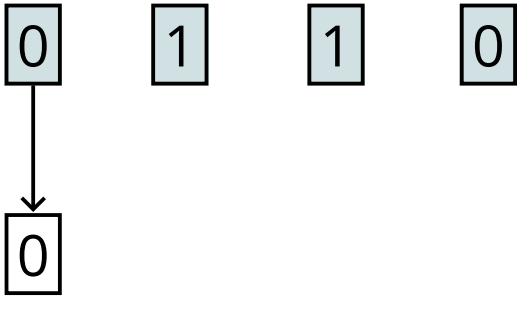
Code
0000
0001
0011
0010
0110
0111
0101
0100
1100
1101
1111
1110
1010
1011
1001
1000

treizième code de Gray

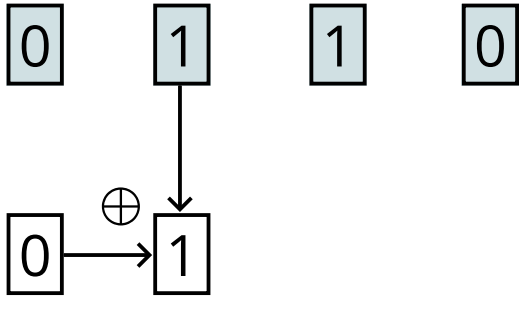
- Comment passer de la représentation de Gray à celle binaire ?
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Règle
    - $b_{n-1} = g_{n-1}$
    - $b_i = g_i \oplus b_{i+1}$
  - Exemple : 0110

- Comment passer de la représentation de Gray à celle binaire ?
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Règle
    - $b_{n-1} = g_{n-1}$
    - $b_i = g_i \oplus b_{i+1}$
  - Exemple : 0110

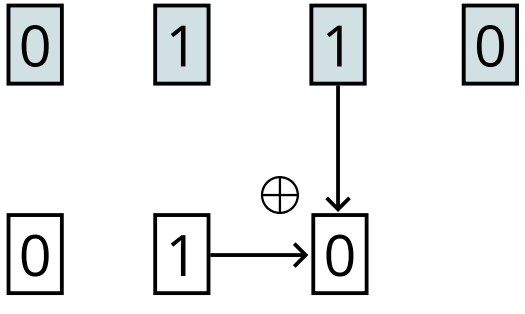
- Comment passer de la représentation de Gray à celle binaire ?
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Règle
    - $b_{n-1} = g_{n-1}$
    - $b_i = g_i \oplus b_{i+1}$
  - Exemple : 0110



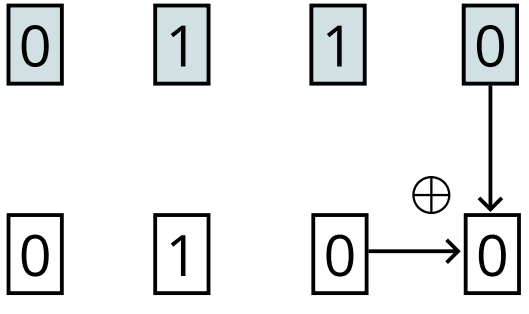
- Comment passer de la représentation de Gray à celle binaire ?
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Règle
    - $b_{n-1} = g_{n-1}$
    - $b_i = g_i \oplus b_{i+1}$
  - Exemple : 0110

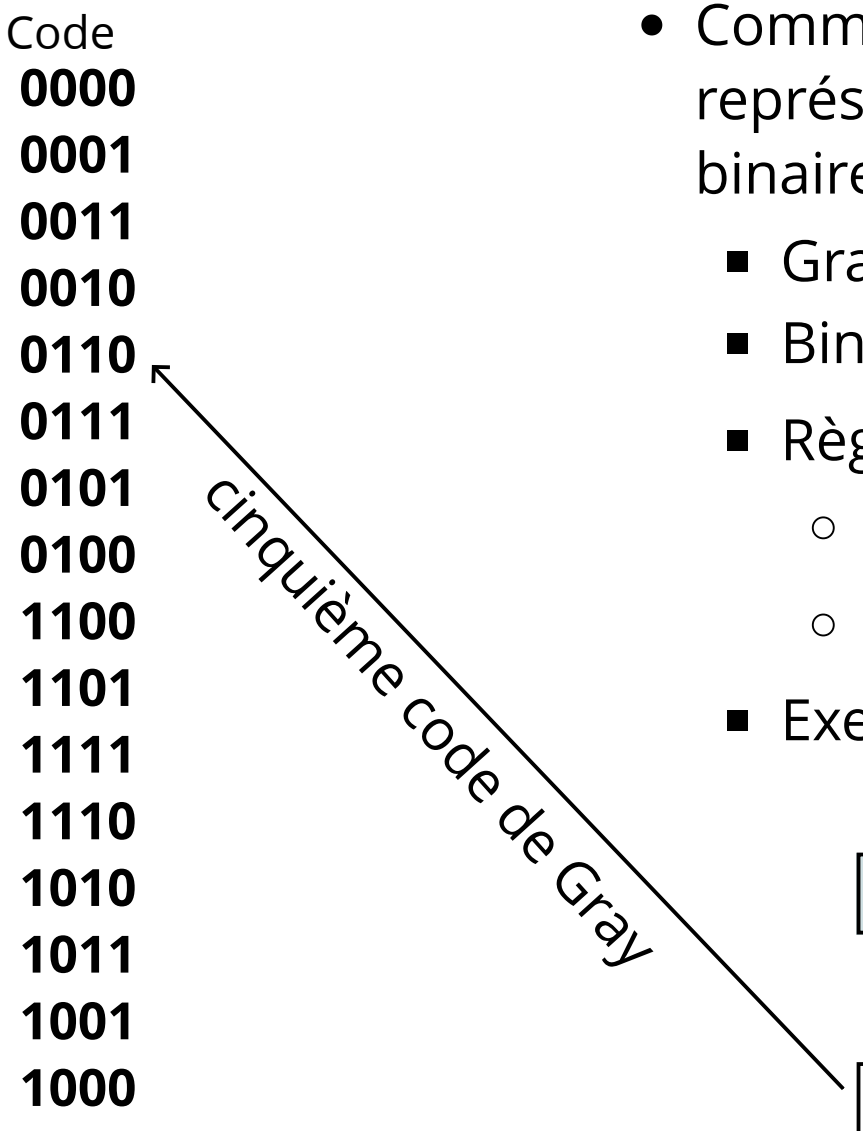


- Comment passer de la représentation de Gray à celle binaire ?
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Règle
    - $b_{n-1} = g_{n-1}$
    - $b_i = g_i \oplus b_{i+1}$
  - Exemple : 0110

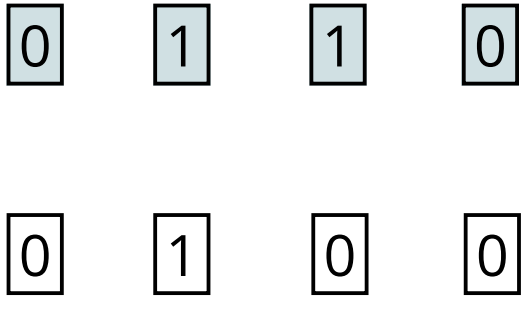


- Comment passer de la représentation de Gray à celle binaire ?
  - Gray $g_{n-1}g_{n-2}\cdots g_1g_0$
  - Binaire $b_{n-1}b_{n-2}\cdots b_1b_0$
  - Règle
    - $b_{n-1} = g_{n-1}$
    - $b_i = g_i \oplus b_{i+1}$
  - Exemple : 0110





- Comment passer de la représentation de Gray à celle binaire ?
  - Gray $g_{n-1}g_{n-2}\cdots g_1 g_0$
  - Binaire $b_{n-1}b_{n-2}\cdots b_1 b_0$
  - Règle
    - $b_{n-1} = g_{n-1}$
    - $b_i = g_i \oplus b_{i+1}$
  - Exemple : 0110



- Un **tableau de Karnaugh** consiste à exprimer la valeur de la fonction sous la forme d'un tableau *torique* dans lequel les variables sont également réparties en entrées des lignes et colonnes et listées selon le code de Gray
- Exemple : pour une fonction a 5 variables

abc de	000	001	011	010	110	111	101	100
00								
01								
11								
10								

- Dans un tel tableau la méthode de Karnaugh consiste à rechercher des rectangles dont les dimensions sont de la forme 2^p (1, 2, 4, 8, 16, etc), dans lesquels la fonction vaut 1 partout et dont p variables sont non-constantes les autres étant fixées
- Exemple

abc de	000	001	011	010	110	111	101	100
00	0	1	1	0	0	0	1	1
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

- Dans un tel tableau la méthode d
  rechercher des rectangles dont le
  2^p (1, 2, 4, 8, 16, etc), dans lesquel
  dont p variables sont non-constar
- Exemple

abc de	000	001	011	010				
00	0	1	1	0				
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

- Longueur : $4 = 2^2$
  - 2 variables non constantes (b et c)
  - 1 variable constante (a)
- Hauteur : $2 = 2^1$
  - 1 variable non constante (e)
  - 1 variable constante (d)

- Dans un tel tableau la méthode de Karnaugh consiste à rechercher des rectangles dont les dimensions sont de la forme 2^p (1, 2, 4, 8, 16, etc), dans lesquels la fonction vaut 1 partout et dont p variables sont non-constantes les autres étant fixées
- Exemple

abc de	000	001	011	010	110	111	101	100
00	0	1	1					
01	0	0	0					
11	1	1	1					
10	1	1	1					

- Longueur : $2 = 2^1$
  - 1 variable non constante (b)
  - 2 variable constantes (a et c)
- Hauteur : $1 = 2^0$
  - Aucune variable non constante
  - 2 variables constantes (d et e)

- Dans un consiste à recherch ns sont de la forme 2^p (1, 2, 4 vaut 1 partout et dont p va es étant fixées
- Exemple

- Longueur : $2 = 2^1$
  - 1 variable non constante (c)
  - 2 variable constantes (a et b)
- Hauteur : $2 = 2^1$
  - 1 variable non constante (d)
  - 1 variable constante (e)

abc / de							101	100
00	0	1	1	0	0	0	1	1
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

- On ne peut pas regrouper ainsi

abc de	000	001	011	010	110	111	101	100
00	0	1	1	1	1	0	0	0
01	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0

  - Le paquet est de longueur 2^2, mais a, b, c changent toutes les trois, c'est donc un mauvais regroupement

- On ne peut que regrouper de la façon suivante

abc de	000	001	011	010	110	111	101	100
00	0	1	1	1	1	0	0	0
01	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0

abc / de	000	001	011	010	110	111	101	100
00	0	1	1	0	0	0	1	1
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1				

- La variable b prend les valeurs 0 et 1 alors que les autres variables sont fixes
  - Les mintermes correspondants peuvent être simplifiés en le simple terme $\neg a \wedge c \wedge \neg d \wedge \neg e$
    - 2 mintermes à 5 variables remplacés par un terme a 4 variables

- Les variables b,c,e prennent les valeurs 0 et 1 alors que les variables a et d sont fixes
  - Les mintermes correspondants peuvent être simplifiés en le simple terme $\neg a \wedge d$
    - 8 mintermes à 5 variables remplacés par un simple terme à 2 variables

abc de	000	001	011	010	110	111	101	100
00	0	1	1	0	0	0	1	1
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

- Il faut ne pas oublier que le tableau est torique
  - Les variables c et d prennent toutes les valeurs possibles alors que a,b,e sont fixes
    - Les mintermes correspondants peuvent être simplifiés en le simple terme $a \wedge \neg b \wedge \neg e$
      - 4 mintermes à 5 variables remplacés par un terme à 3 variables

abc / de								
						11	101	100
00						0	1	1
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

abc / de	000	001	011	010	110	111	101	100
00	0	1	1	0	0	0	1	1
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

- Ces rectangles ne sont pas simplifiables
  - On conserve les mintermes initiaux $a \wedge b \wedge \neg c \wedge \neg d \wedge e$ et $a \wedge b \wedge c \wedge d \wedge e$

abc de	000	001	011	010	110	111	101	100
00	0	1	1	0	0	0	1	1
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

abc de	000	001	011	010	110	111	101	100
00	0	1	1	0	0	0	1	1
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

- Fonction original
  - 16 mintermes à 5 variables

abc / de	000	001	011	010	110	111	101	100
00	0	1	1	0	0	0	1	1
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

- Fonction original
  - 16 mintermes à 5 variables
- Fonction simplifiée
  $(\neg a \wedge c \wedge \neg d \wedge \neg e) \vee (\neg a \wedge d) \vee (a \wedge \neg b \wedge \neg e) \vee (a \wedge b \wedge \neg c \neg d \wedge e) \vee (a \wedge b \wedge c \wedge d \wedge e)$

abc / de	000	001	011	010	110	111	101	100
00	0	1	1	0	0			
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

- On aurait plus simplifier encore un peu plus
  - $\neg a \wedge c \wedge \neg d \wedge \neg e$ remplacé par $\neg a \wedge \neg c \wedge \neg e$

abc / de	000	001	011	010	110	111	101	100
00	0	1	1	0	0			
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

- On aurait plus simplifier encore un peu plus
  - $\neg a \wedge c \wedge \neg d \wedge \neg e$ remplacé par $\neg a \wedge \neg c \wedge \neg e$

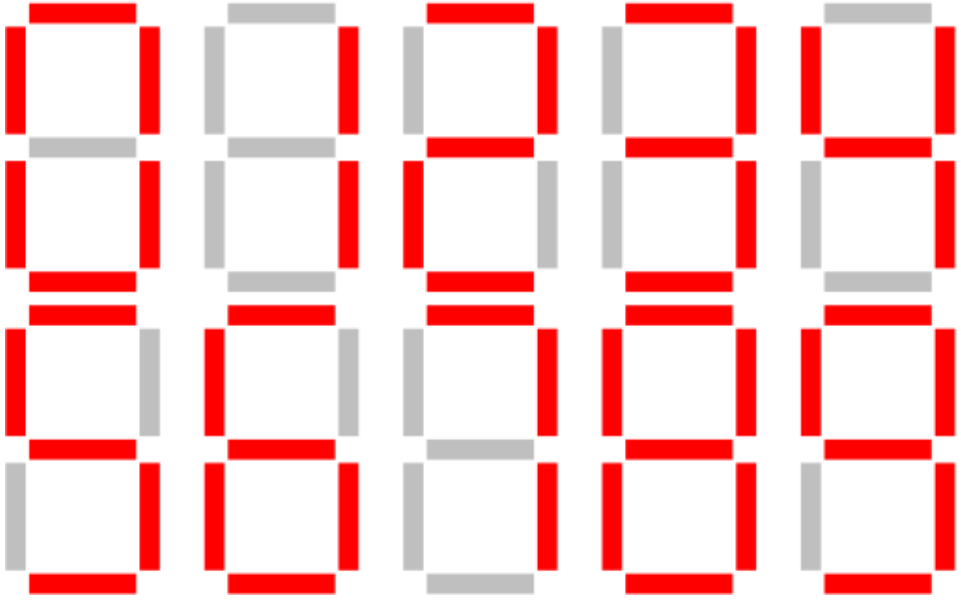
- Fonction simplifiée
  $(\neg a \wedge c \wedge \neg e) \vee (\neg a \wedge d) \vee (a \wedge \neg b \wedge \neg e) \vee (a \wedge b \wedge \neg c \neg d \wedge e) \vee (a \wedge b \wedge c \wedge d \wedge e)$

abc / de	000	001	011	010	110	111	101	100
00	0	1	1	0	0			
01	0	0	0	0	1	0	0	0
11	1	1	1	1	0	1	0	0
10	1	1	1	1	0	0	1	1

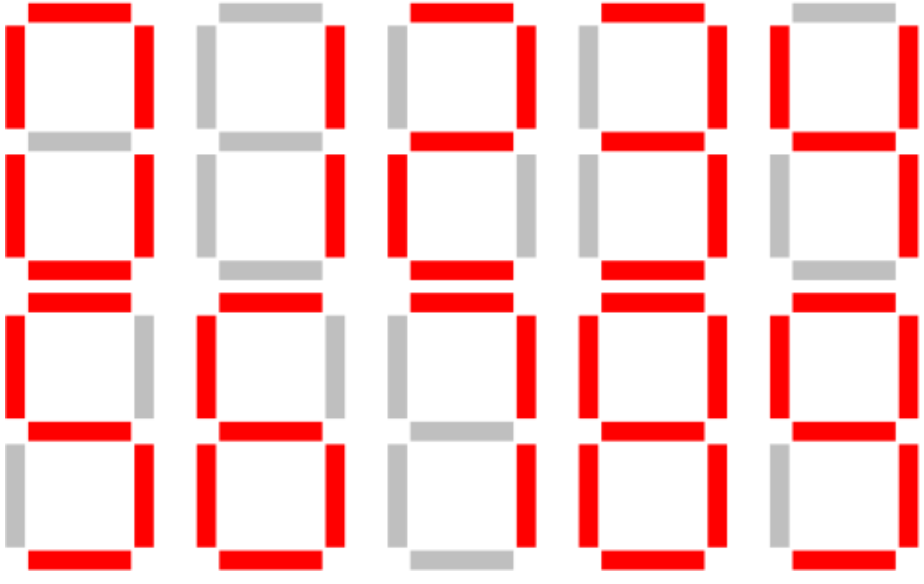
- On aurait plus simplifier encore un peu plus
  - $\neg a \wedge c \wedge \neg d \wedge \neg e$ remplacé par $\neg a \wedge \neg c \wedge \neg e$

- Fonction simplifiée
  $(\neg a \wedge c \wedge \neg e) \vee (\neg a \wedge d) \vee (a \wedge \neg b \wedge \neg e) \vee (a \wedge b \wedge \neg c \neg d \wedge e) \vee (a \wedge b \wedge c \wedge d \wedge e)$
- **Il faut toujours essayer de combiner les 1 dans des rectangles aussi grands que possible**

- L'afficheur à 7 segments pour les chiffres décimaux codés en binaire



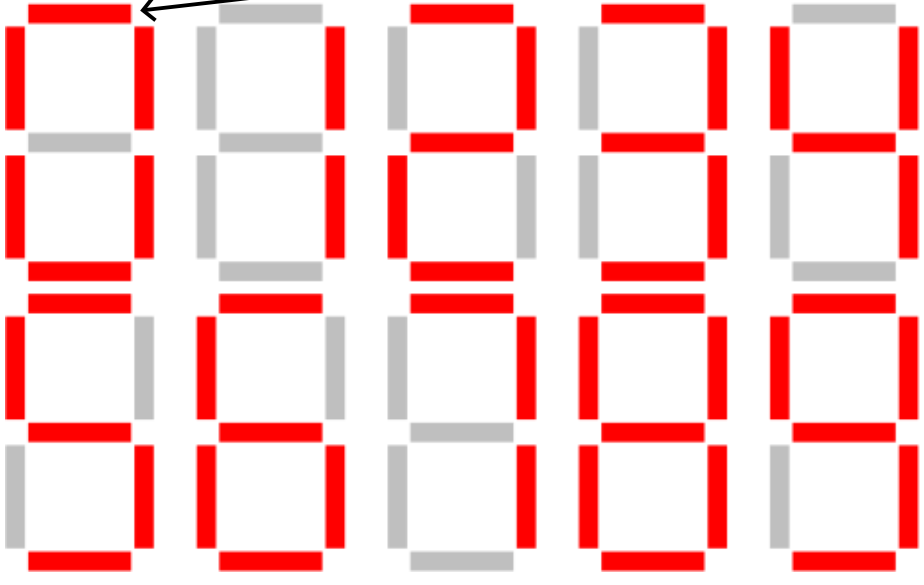
- Écrire les fonctions associés à chaque segment sous la FND
- Simplifier les fonctions par la méthode de Karnaugh



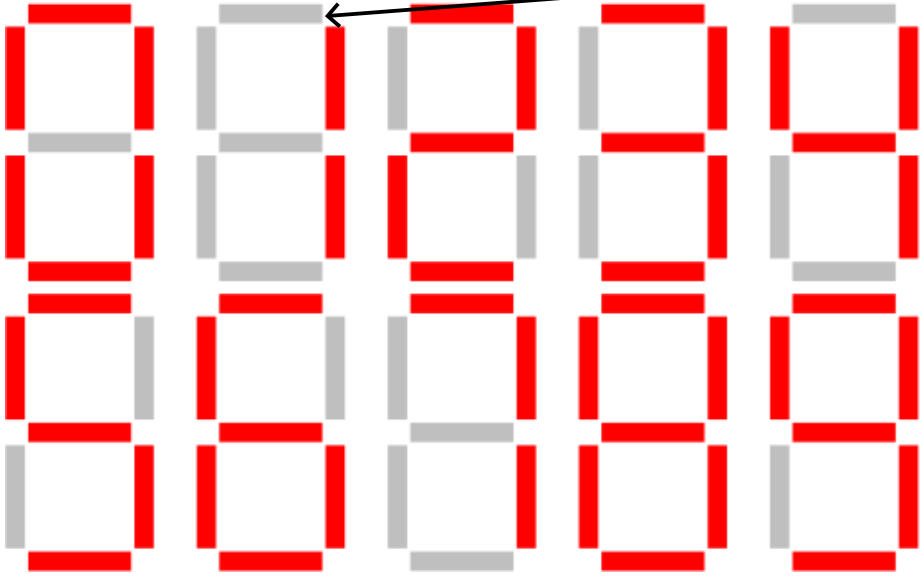
- Table de vérité du segment en haut

n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
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
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

- Table de vérité du segment en haut

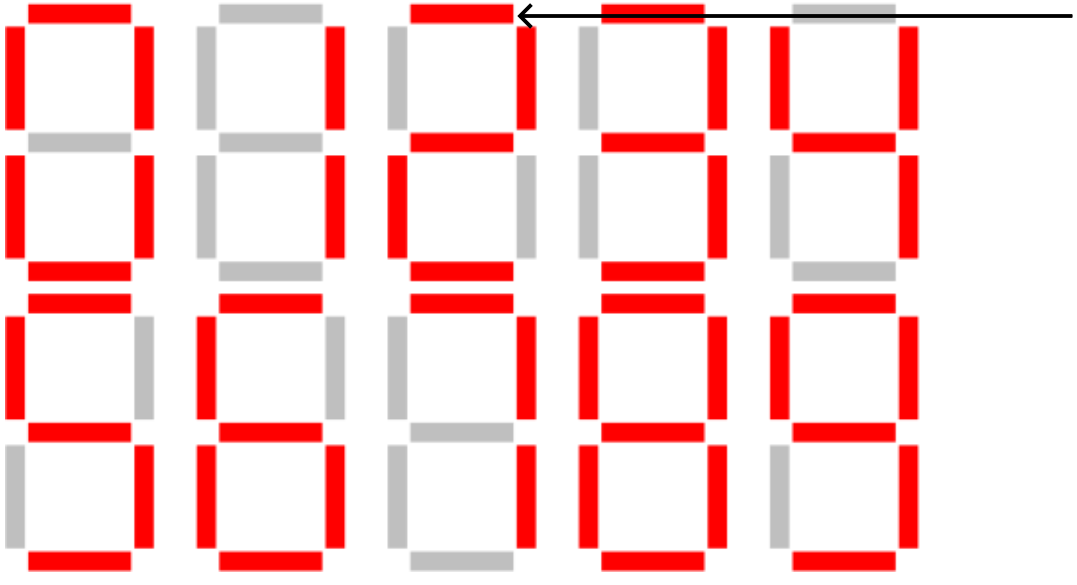


n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0



- Table de vérité du segment en haut

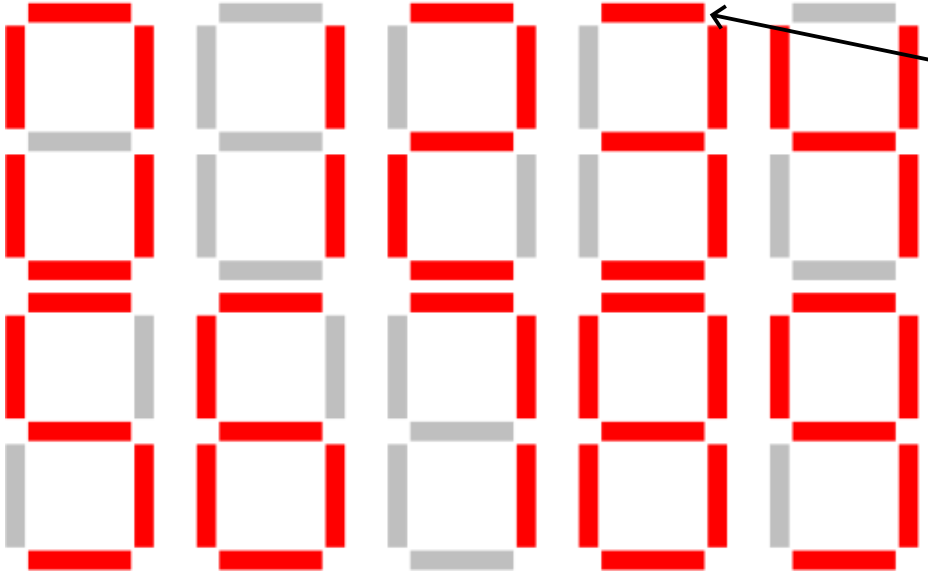
n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0



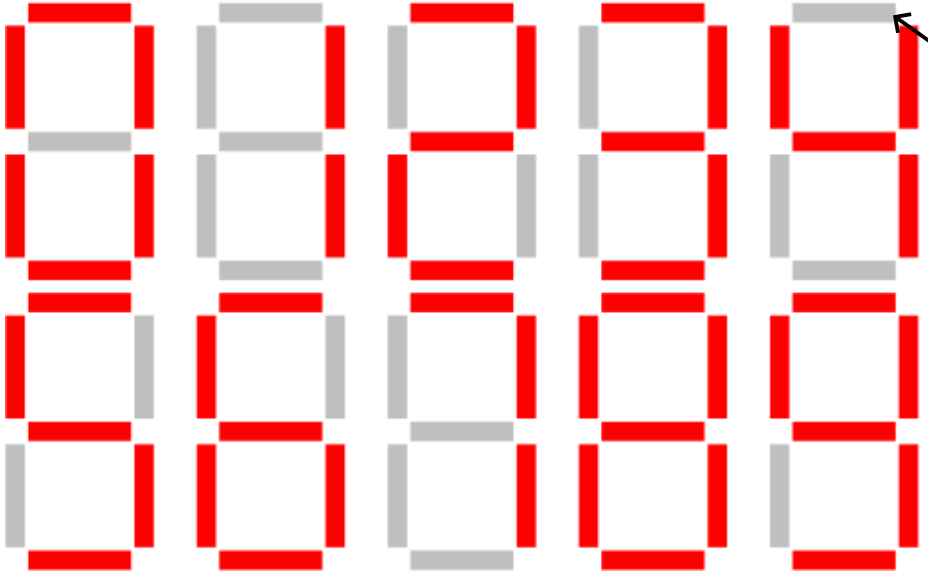
- Table de vérité du segment en haut

n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
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
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

- Table de vérité du segment en haut

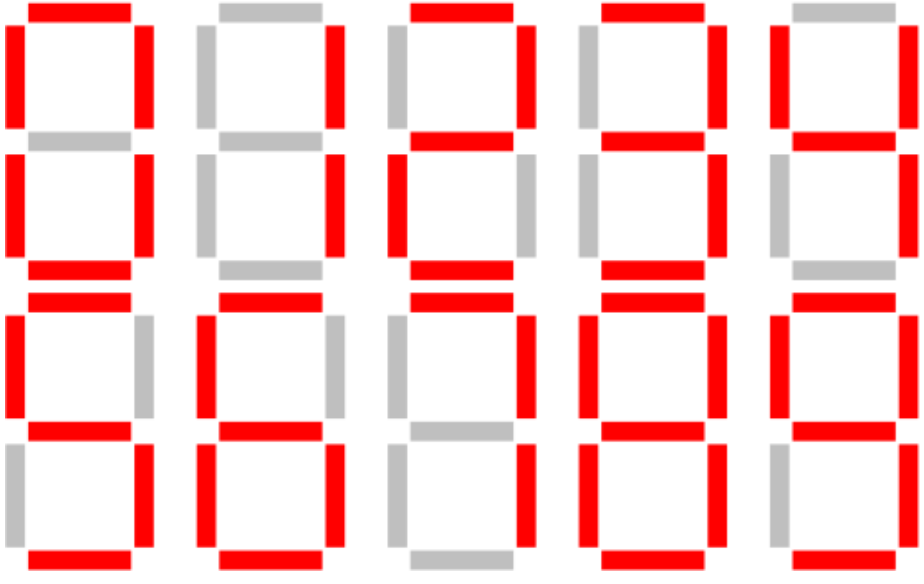


n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0



- Table de vérité du segment en haut

n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
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
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0



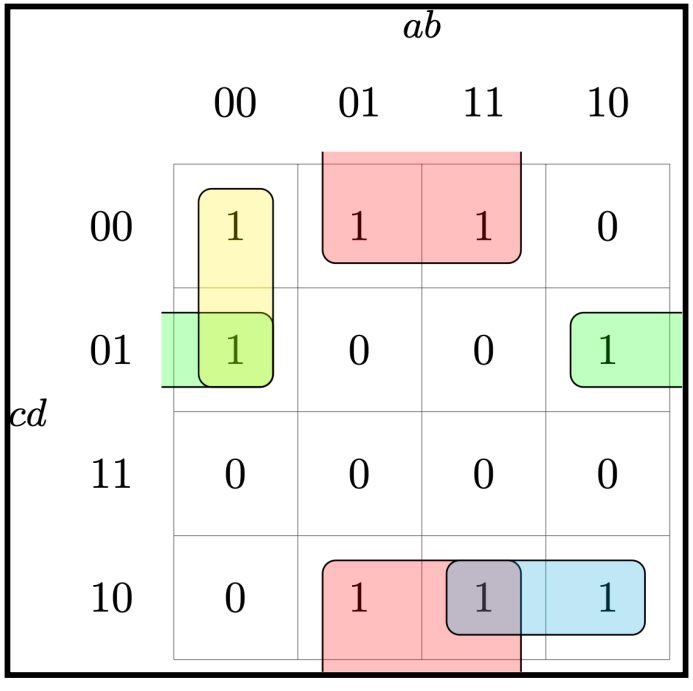
- 8 mintermes à 4 variables

- Table de vérité du segment en haut

n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
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
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

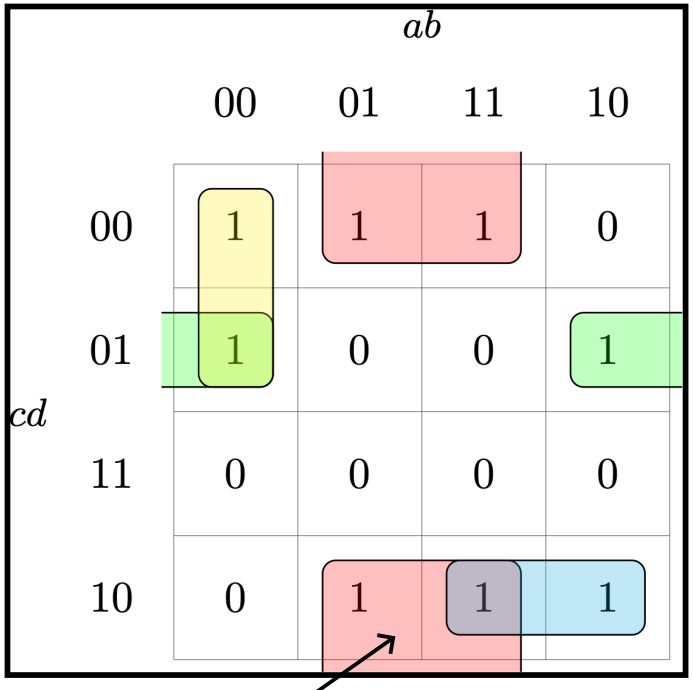
n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

- Tableau de Karnaugh



n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
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
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

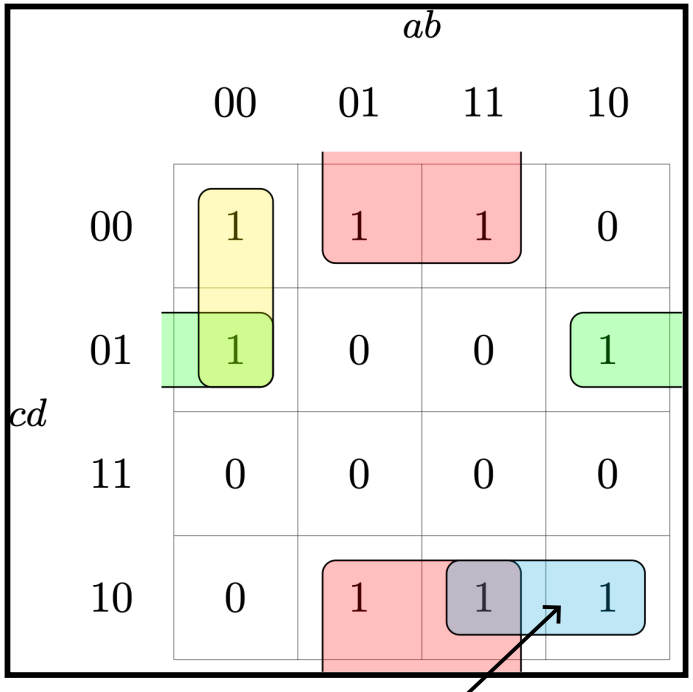
- Tableau de Karnaugh



$(b \wedge \neg d)$

n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
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
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

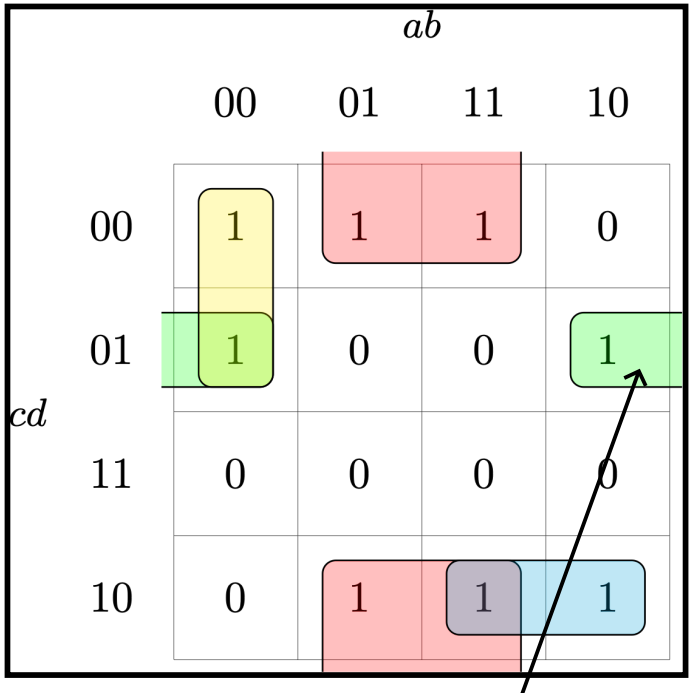
- Tableau de Karnaugh



$$(b \wedge \neg d) \vee (a \wedge c \wedge \neg d)$$

n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
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
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

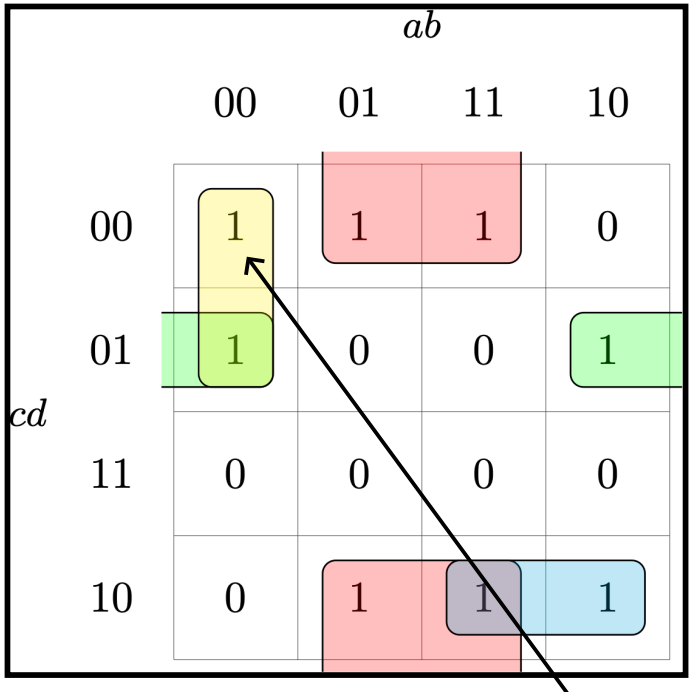
- Tableau de Karnaugh



$$(b \wedge \neg d) \vee (a \wedge c \wedge \neg d) \vee (\neg b \wedge \neg c \wedge d)$$

n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
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
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

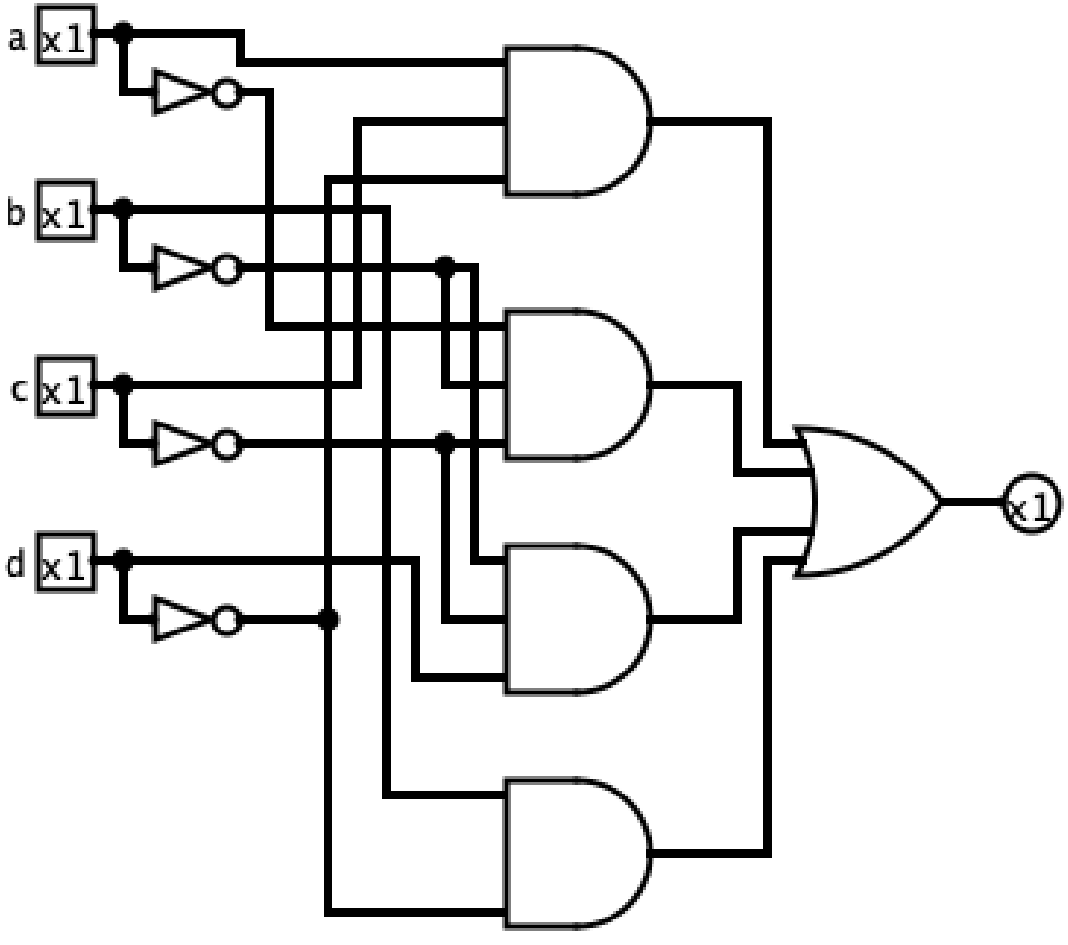
- Tableau de Karnaugh



$$(b \wedge \neg d) \vee (a \wedge c \wedge \neg d) \vee (\neg b \wedge \neg c \wedge d) \vee (\neg a \wedge \neg b \wedge \neg c)$$

n	d	c	b	a	s
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
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
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

- Tableau de Karnaugh



$$(b \wedge \neg d) \vee (a \wedge c \wedge \neg d) \vee (\neg b \wedge \neg c \wedge d) \vee (\neg a \wedge \neg b \wedge \neg c)$$