Unité : Sytémes logiques 1 (SysLog1)

Portes logiques et algèbre de Boole

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

E. Messerli & R. Mosqueron

Septembre 2021

Polycopié : Electronique numérique

- Portes logiques et algèbre de Boole

  chapitre 4, pages 35 à 54

- Circuits logiques combinatoires
  - Simplification, tables de Karnaugh

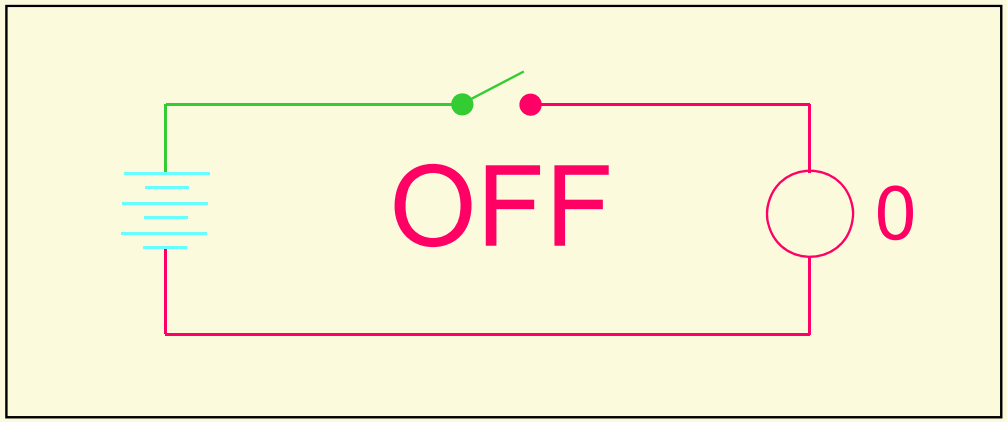
    chapitres 5-1 à 5-6, pages 55 à 64

- Symboles utilisés

  chapitre 4-9, pages 46 et 47

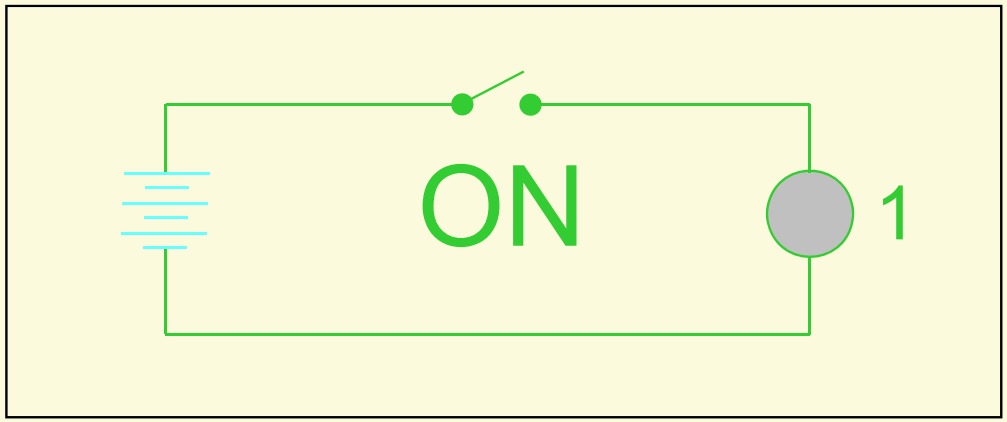
Principe de la logique (postulat) …

- Être logique, c'est :
  avoir une réponse unique sans contradiction
- Pas d'Affirmation et de Négation en même temps !!!
  - Une lampe ne peut jamais être Allumée (ON) et Eteinte (OFF) en même temps



… principe de la logique (postulat) …

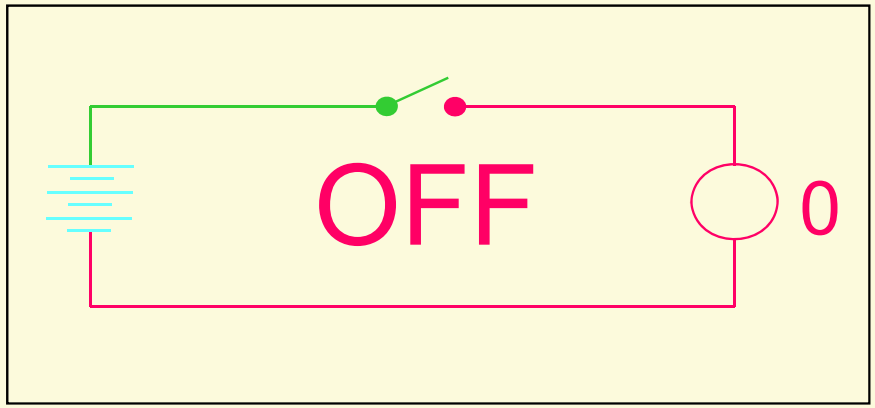
- Être logique, c'est
  avoir une réponse unique sans contradiction
- Pas d'Affirmation et de Négation en même temps !!!
  - Une lampe ne peut jamais être Allumée (ON) et Eteinte (OFF) en même temps



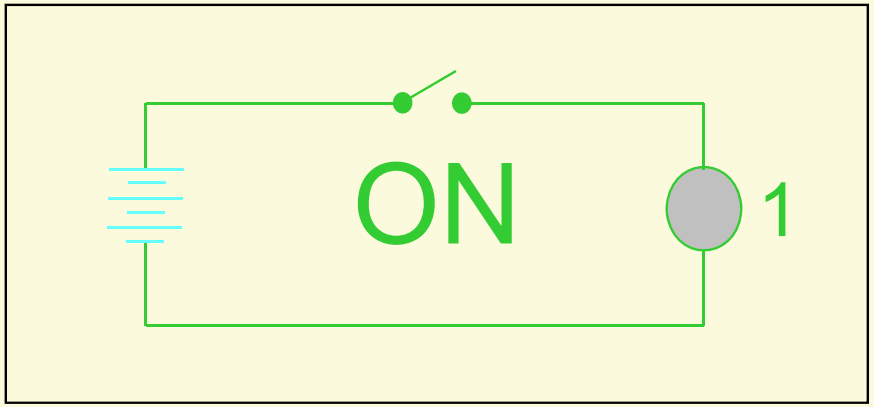
… principe de la logique (postulat)

- On voit clairement une variable binaire :
  - symbolisé par les états **'0'** et **'1'**

0 ➔ OFF



1 ➔ ON



"La Nature" versus "Monde binaire"

- La nature n'est jamais binaire !
  - Début du jour ou levé du soleil ?
  - Heure exacte: 12h00, 12h00'00", …
  - Impression de froid ou de chaud ?

- Le binaire est utile, mais il ne représente pas

  toute "la nature"

Ce paysage est vert !

Système logique

- C'est un système qui traite l'information de façon logique
- Pour étudier un système logique, il faut connaître les fonctions de base (les composants) et le langage mathématique qui permet de décrire un comportement sous forme d'équations
- Pour un additionneur 1 bit :

$$Z = f\,(X, Y)$$

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Système binaire

- Système logique qui emploie des informations élémentaires (signaux) à deux états
- Avantages :
  - on peut utiliser des interrupteurs comme éléments de base du système
  - un signal binaire est plus fiable (détection de 2 états)
  - les décisions prises dans un système digital sont très souvent binaires
- En général, les 2 états sont représentées par les chiffres 0 et 1

Quelques définitions

- Etat logique :
  - chacune des 2 valeurs que peut prendre une variable logique
- Variable logique :
  - grandeur qui ne peut prendre que les 2 états logiques
- Variable d'entrée (ou simplement entrée) :
  - information à 2 états reçue par un système logique
- Variable de sortie (ou simplement sortie) :
  - information à 2 états générée par un système logique
- Fonction logique :
  - relation logique entre une sortie et une ou plusieurs entrées

Types de systèmes logiques

- Système combinatoire :
  - la valeur des sorties à un moment donné dépend uniquement des valeurs des entrées à cet instant

    => système univoque
  - le comportement est entièrement spécifié par une table, nommée table de vérité, qui pour chaque combinaison des entrées donne la valeur des sorties (n entrées=> table de vérité comporte 2^n lignes)
  - la sortie est immédiate
- Système séquentiel :
  - la valeur des sorties dépend de la valeur des entrées au cours du temps: il faut une ***mémoire***

    => dépend de l'historique
  - l'obtention d'un résultat peut demander plusieurs étapes, avec mémorisation de résultats intermédiaires

Les portes logiques de bases

- Etude des fonctions d'une variable
  - Nous verrons la porte logique de base : NON
- Etude des fonctions de deux variables
  - Nous verrons les portes logiques de base : ET, OU
- Nous verrons ensuite des combinaisons de portes de bases :
  - Fonction : NON-ET, NON-OU
  - Fonction : OU-Exclusif

dia volontairement vide

Fonctions d'une variable

Table des fonctions d'une variable :

Variable	Fonctions F1.x			
A	F1.0	F1.1	F1.2	F1.3
0	0	0	1	1
1	0	1	0	1
			not	

F1.0 = 0 constante 0

F1.1 = A transmission

F1.2 = not A = /A fonction utile à seule variable : **NON** (inversion logique)

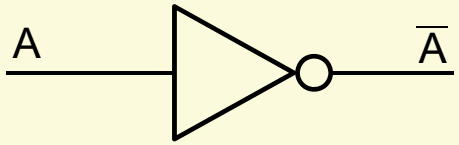
F1.3 = 1 constante 1

Fonction NON (not)

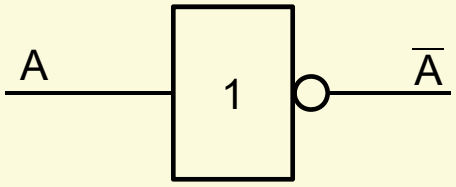
- F1.2 = not A = /A
  - La sortie du circuit est à l'état logique inverse de son entrée

A	F1.2
0	1
1	0

Symbole MIL (US)



Symbole IEEE



Fonctions de deux variables …

Table des 16 fonctions de deux variables (fcts 0 à 7)

Variables		Fonctions F2.x							
B	A	F2.0	F2.1	F2.2	F2.3	F2.4	F2.5	F2.6	F2.7
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1
			and					**xor**	**or**

F2.0 = 0

F2.1 = B and A = B • A

F2.2 = B and not A = B • /A

F2.3 = B

F2.4 = not B and A = /B • A

F2.5 = A

F2.6 = B xor A = B ⊕ A

F2.7 = B or A = B + A

… fonctions de deux variables …

- Nous pouvons repérer les fonctions :
  - ET (AND, intersection) = F2.1
  - OU (OR, réunion) = F2.7

- Autre fonction logique intéressante :
  - OU-EXCLUSIF (XOR) = F2.6 fonction différence

… fonctions de deux variables …

Table des 16 fonctions de deux variables: (fcts 8 à 15)

Variables		Fonctions F2.x							
B	A	F2.8	F2.9	F2.A	F2.B	F2.C	F2.D	F2.E	F2.F
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1
		nor	xnor					nand	

F2.8 = /F2.7 = /(B + A)

F2.9 = /F2.6 = /(B ⊕ A)

F2.A = /F2.5 = /A

F2.B = /F2.4

xnor

F2.C = /F2.3 = /B

F2.D = /F2.2

F2.E = /F2.1 = /(B • A)

F2.F = /F2.0 = 1

…. fonctions de deux variables

- Les fonctions F2.8 à F2.F sont respectivement les inverses des fonctions F2.7 à F2.0

- Parmi ces fonctions 3 sont intéressantes :
  - NON-ET (NAND) fonction universelle
  - NON-OU (NOR) fonction universelle
  - NON-OU-EXCLUSIF (XNOR) fonction égalité

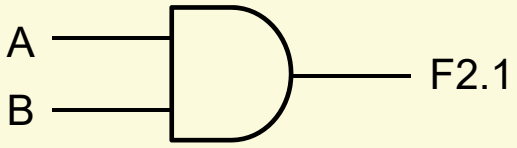
Fonction ET (and)

- F2.1 = B . A = B and A

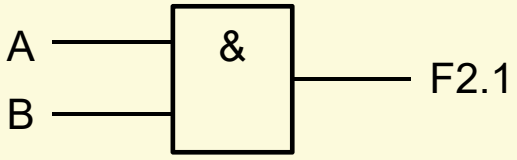
  La sortie de la porte « ET » est à 1 si l'entrée A et l'entrée B sont à 1

B	A	F2.1
0	0	0
0	1	0
1	0	0
1	1	1

Symbole MIL (US)

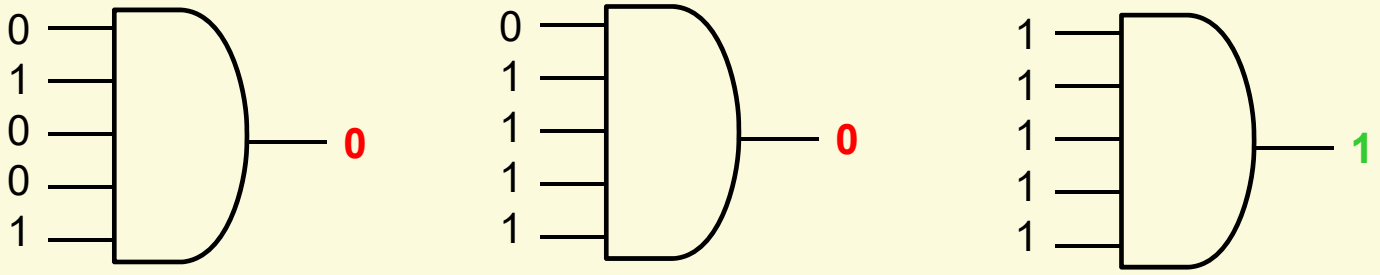


Symbole IEEE



Fonction ET à n entrées

- La définition peut-être étendue à n entrées:
  - La sortie de la porte « ET » est à 1 si toutes les entrées sont à 1.

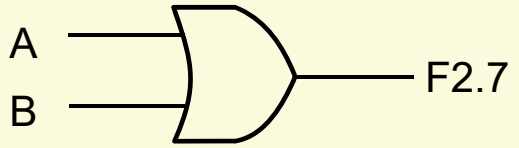


Fonction OU (or)

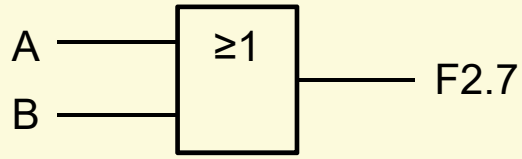
- F2.7 = B + A = B or A
  - La sortie de la porte « OU » est à 1 si l'entrée A ou B valent 1 (l'une ou l'autre ou les deux)

B	A	F2.7
0	0	0
0	1	1
1	0	1
1	1	1

Symbole MIL (US)

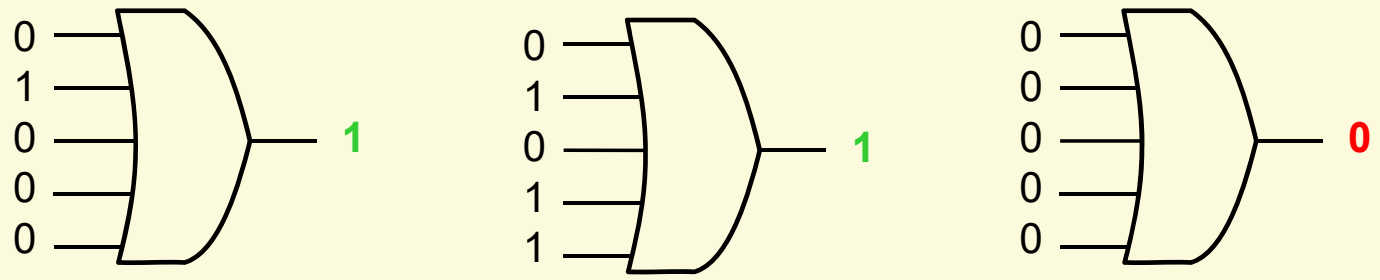


Symbole IEEE



Fonction OU à n entrées

- La définition peut-être étendue à n entrées:
  - La sortie du circuit «OU» est à 1 si une entrée est à 1.

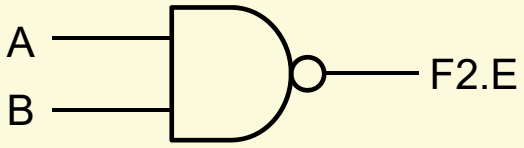


Fonction NON-ET (nand)

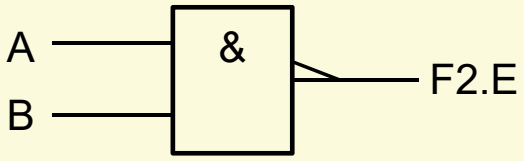
- F2.E = /F2.1 = /(B . A) = B nand A
  - Fonction inverse du ET, soit : la fonction est à 1 si une entrée est à 0 .
  - Fonction universelle

B	A	F2.1	F2.E
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Symbole MIL (US)



Symbole IEEE

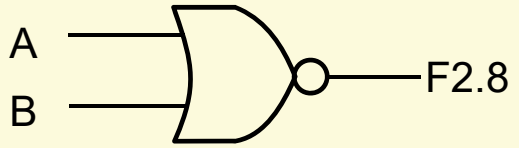


Fonction NON-OU (nor)

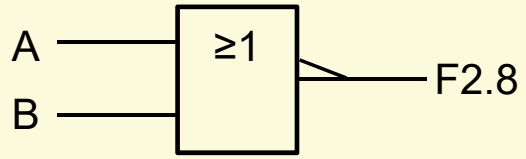
- F2.8 = /F2.7 = /(B + A) = B nor A
  - Fonction inverse du OU, soit: la fonction est à 1 si toutes les entrées sont à 0
  - Fonction universelle

B	A	F2.7	F2.8
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Symbole MIL (US)



Symbole IEEE

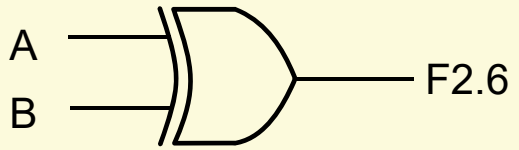


Fonction OU-Exclusif (xor)

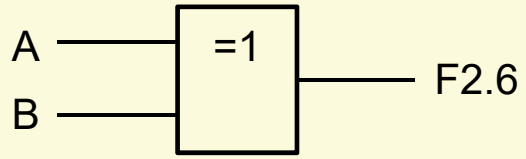
- F2.6 = B ⊕ A = B xor A
  - La sortie de la porte « OU-Exclusif » est à 1 si A ou B valent 1, mais pas les 2 (détecte la différence)
  - Pas une fonction de base!

B	A	F2.6
0	0	0
0	1	1
1	0	1
1	1	0

Symbole MIL (US)



Symbole IEEE



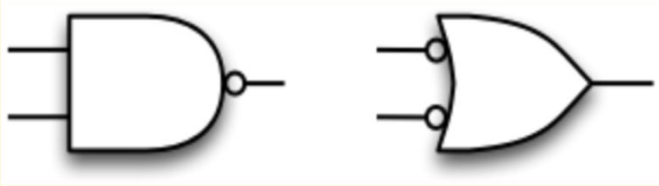
Réalisation d'un système combinatoire

- Tout système combinatoire, quelque soit le nombre d'entrées, peut être décrit avec les 3 fonctions de base :
  - NON
  - ET
  - OU
- ou seulement avec la fonction universelle NAND à 2 entrées
- ou seulement avec la fonction universelle NOR à 2 entrées

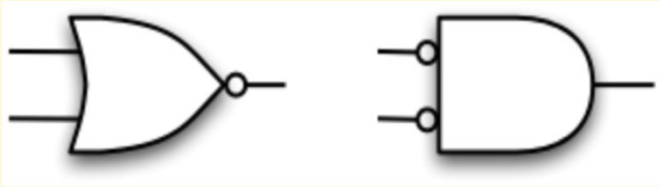
Fonctions universelles

- Un fonction est universelle lorsqu'elle permet la réalisation des trois fonctions logiques de base (NON, ET, OU)

- NAND



- NOR



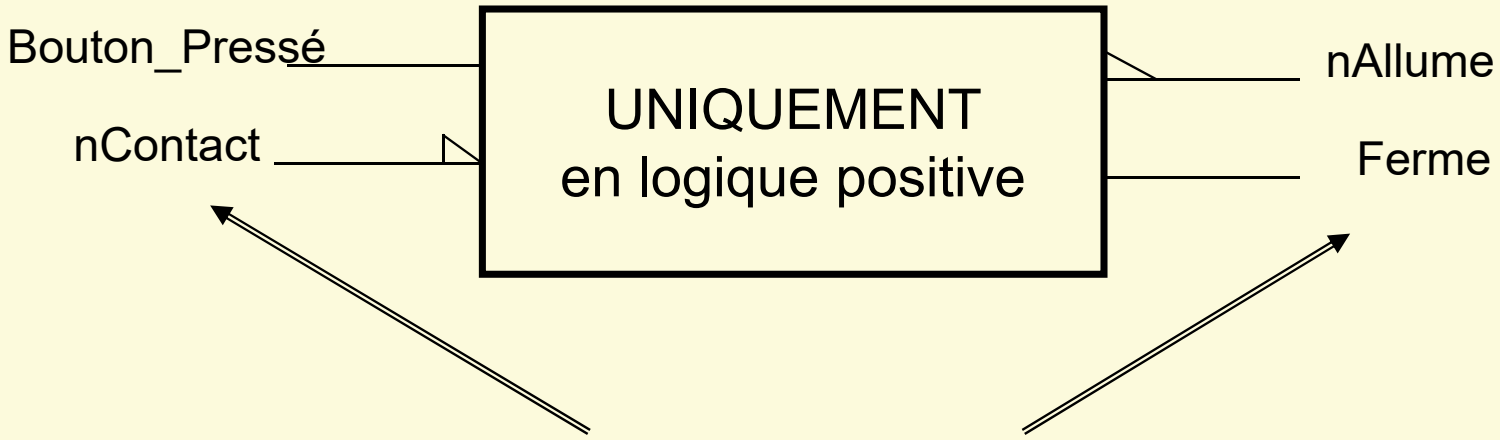
Logigramme

- Logigramme = schéma logique
- Utilise les symboles graphiques des fonctions usuelles (ET, OU, NON, …)
- Montre les liaisons entre les entrées, les fonctions utilisées et les sorties
- Par convention, les signaux vont de gauche à droite (entrées à gauche, sorties à droite)

Convention

A l'intérieur d'un bloc : **logique positive** (état actif = 1)

A l'extérieur : logique mixte actif haut ou bas
si actif bas => cela doit être indiqué (nom signal)



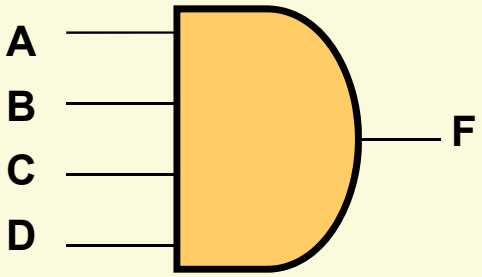
Décomposition fonctions de base …

- Les fonctions ET, OU à plus de 2 entrées peuvent être réalisées à l'aide des fonctions correspondantes à 2 entrées seulement
- 3 formes :
  - Parallèle (non décomposée)
  - Décomposition pyramidale
  - Décomposition en cascade

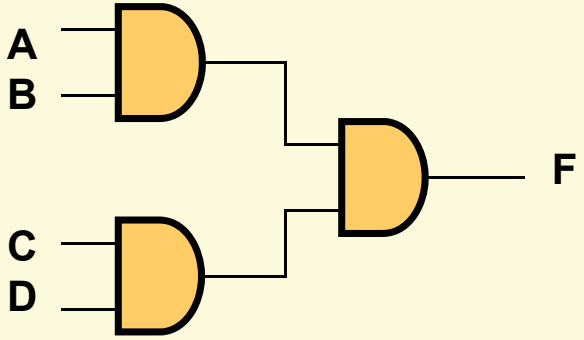
… décomposition fonctions de base

- Exemple : fonction ET à 4 entrées

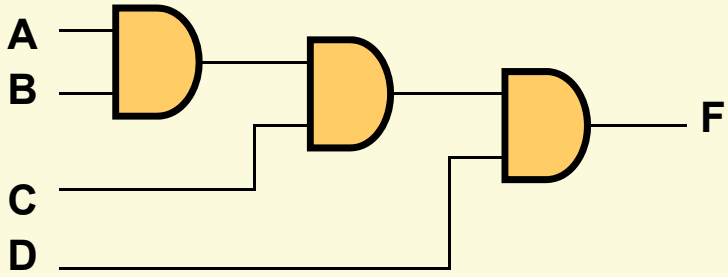
Parallèle



Pyramidale



Cascadée



Exercices I

a. Ecrivez l'équation logique de la sortie Z de l'additionneur 2 bits de la page 7.
b. Exprimez la fonction OU-EXCLUSIF (xor) à 2 entrées à l'aide des fonctions de base uniquement, dessinez le logigramme.
c. La fonction NON OU-EXCLUSIF (xnor) est appelée « égalité ». Pourquoi ?
d. Réalisez la fonction « impair » à 3 entrées, en utilisant des fonctions OU-EXCLUSIF (xor) à 2 entrées.
   - La sortie "Impair" est active si le nombre de '1' parmi les 3 entrées est impair.
e. Démontrez que les fonctions NAND et NOR sont universelles.

Algèbre de Boole : postulats

- Postulats de l'algèbre de Boole

  A + /A = 1 (A or /A = 1)

  A • /A = 0 (A and /A = 0)

- découlent de l'hypothèse :

  l'inverse d'une variable ne peut jamais avoir la même valeur que la variable

Les postulats seront violés dans les circuits réels!

Algèbre de Boole : théorèmes …

I	/(/A)	= A	théorème de l'involution not(notA) = A
II	A + 0	= A	
III	A • 0	= 0	
IV	A + 1	= 1	
V	A • 1	= A	
VI	A + A	= A	idempotence de l'opérateur OU
VII	A • A	= A	idempotence de l'opérateur ET

… théorèmes …

VIII	$A + B = B + A$	commutativité
IX	$A \cdot B = B \cdot A$	commutativité
X	$A + (B + C) = (A + B) + C = A + B + C$	associativité
XI	$A \cdot (B \cdot C) = (A \cdot B) \cdot C = A \cdot B \cdot C$	associativité
XII	$A + B \cdot C = (A + B) \cdot (A + C)$	distributivité
XIII	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	distributivité

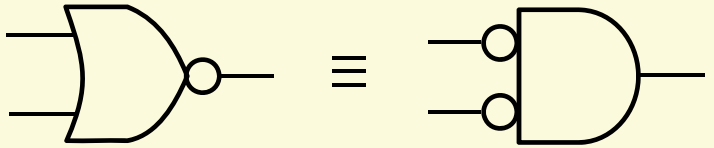
… théorèmes

Théorèmes de De Morgan (1806-1871)

XIV

$$/(B + A) = /B \bullet /A$$

B	A	/B	/A	NOR
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0



XV

$$/(B \bullet A) = /B + /A$$

B	A	/B	/A	NAND
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

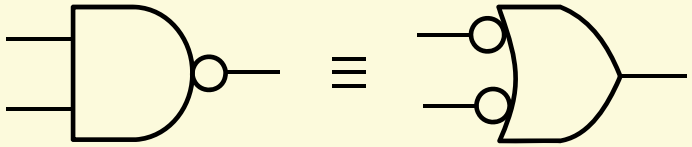
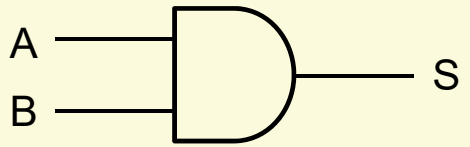


Table de vérité (TDV)

- Liste des valeurs de sortie en fonction des combinaisons des entrées

no	B	A	S
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1



- Permet de spécifier **toutes** les combinaisons possible d'une fonction logique

Mintermes

- On appelle minterme, ou fonction unité de deux variables chacun des quatre monômes de ces deux variables:

  minterme 0 = /B • /A

  minterme 1 = /B • A

  minterme 2 = B • /A

  minterme 3 = B • A

- Ceci se généralise à n variables

Construction TDV

TDV : Table de vérité

- Table avec la liste de toutes les combinaisons des entrées
  - N entrées => 2^N lignes dans la table

Exemple de table à 4 entrées

No minterme	D	C	B	A	F
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
..	..	..	..	..	..
14	1	1	1	0	1
15	1	1	1	1	0

Liste des mintermes d'une fonction

- Soit la TDV d'une fonction :

No	C B A	F
0	0 0 0	1
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	1
6	1 1 0	0
7	1 1 1	1

Nous pouvons résumer la TDV en donnant la liste des mintermes vrai ('1'):

$$F(C,B,A) = \Sigma\ 0, 3, 5, 7$$

Forme canonique décimale

Equation logique

- L'équation canonique découle directement de la TDV. Mais il peut exister des solutions équivalentes.
- Exemple: la fonction OU

B	A	Z
0	0	0
0	1	1
1	0	1
1	1	1

équation canonique:

$$Z(B,A) = \overline{B}A + B\overline{A} + BA$$

équation simplifiée:

$$Z(B,A) = B + A$$

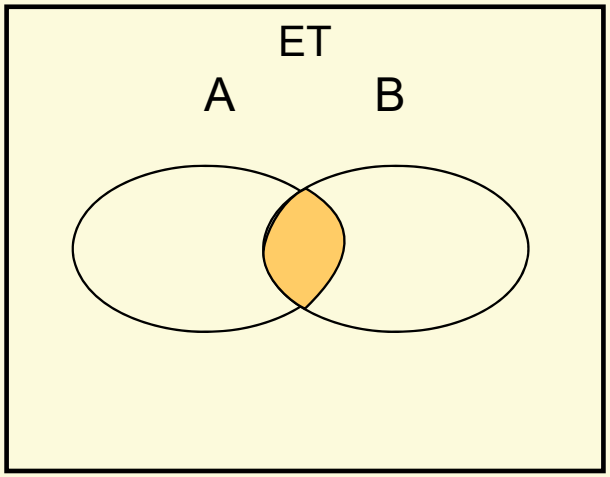
- Comment simplifié la fonction => deux méthodes:
  - algébrique (algèbre de Boole)
  - graphique (table de Karnaugh)

Diagramme de Venn

Représentation graphique d'une fonction logique

ET: intersection

OU: réunion



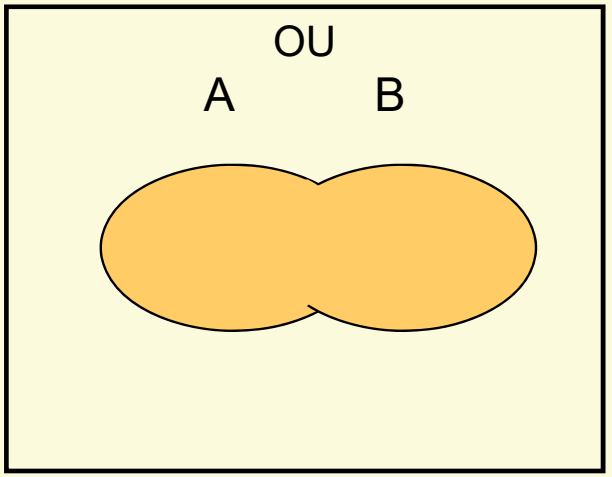


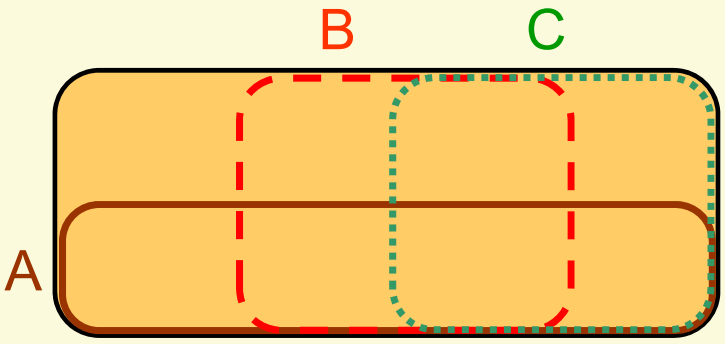
Table de Karnaugh …

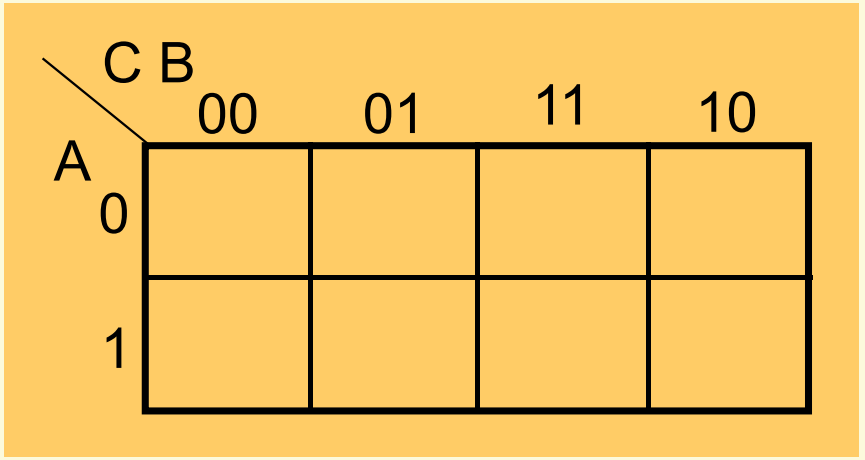
Objectif: représentation graphique d'une TDV

- Forme stylisée d'un diagramme de Venn
- Chaque case correspond à
  - une combinaison des valeurs des entrées, soit à un minterme
  - donc à **une ligne** de la table de vérité
- N entrées $\rightarrow 2^N$ cases
- Dans chaque case on indique la valeur de la fonction, soit
  - 0 ou 1
  - - *ou* Ø => *état indifférent*

… table de Karnaugh …

- Variables d'entrée indépendantes : chaque variable a une intersection avec
  - toutes les autres variables
  - et avec toutes les intersections des autres variables
- Fonction de 3 variables : 2^3 cases = 8





… table de Karnaugh

Autre notation de la table

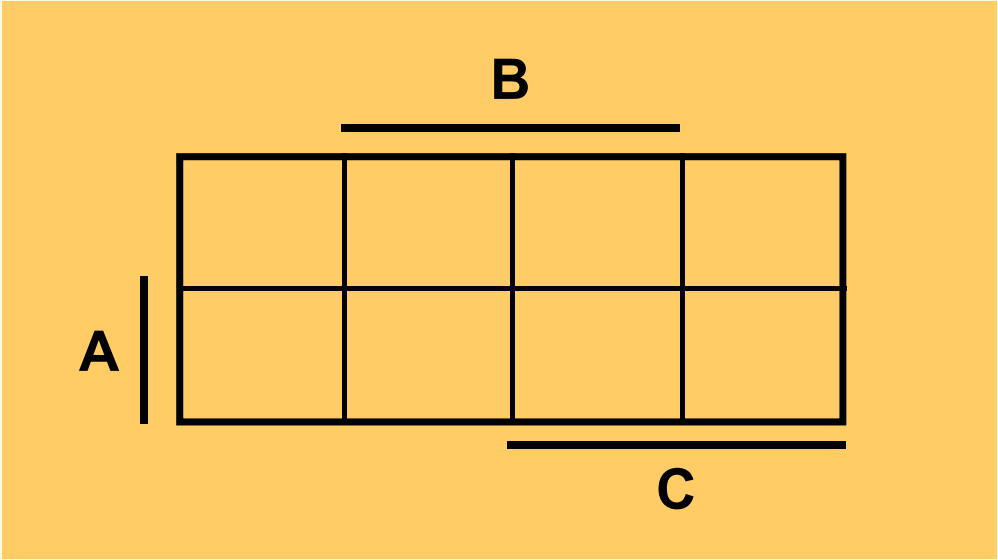
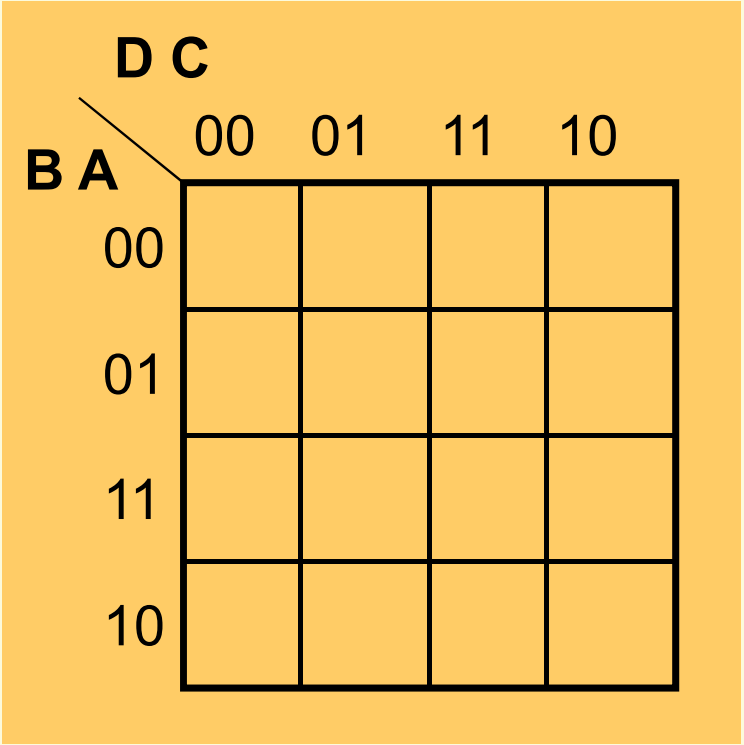


Table de Karnaugh à 4 variables

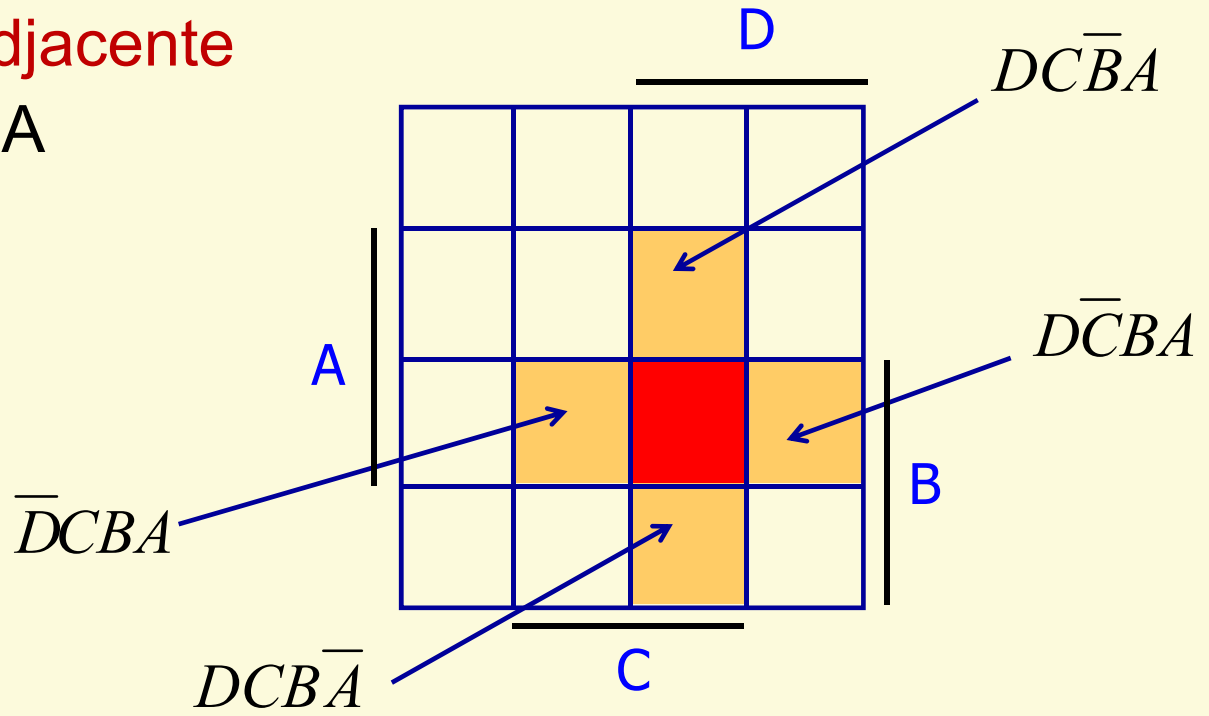


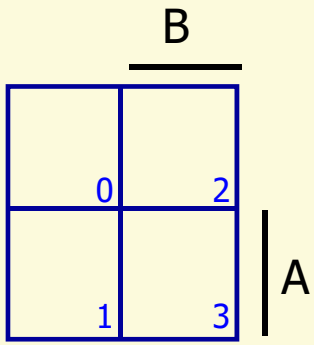
Structure de la table de Karnaugh

- Une table de Karnaugh est similaire à une table de vérité :
  - TDV : chaque ligne correspond à une combinaison des entrées
  - Karnaugh : chaque ligne correspond à une combinaison des entrées
- Le nombre de cases d'une table de Karnaugh d'une fonction à n variables est donc égal à 2^n
- Les cases de la table de Karnaugh sont arrangées de telle façon qu'une seule variable change entre deux cases contiguës
- Toute case d'une table de Karnaugh à n variables est contiguë à n autres cases

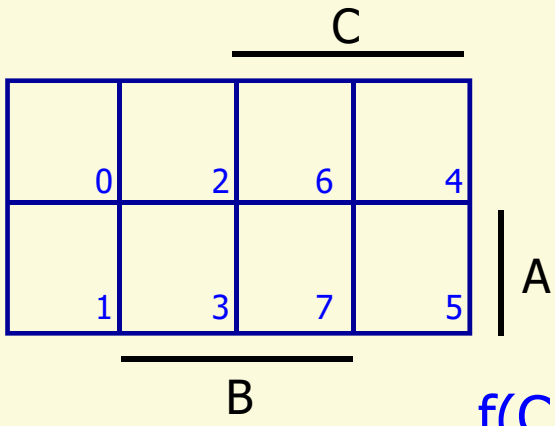
Exemple table de Karnaugh 4 variables

- 4 variables, donc 4 cases contiguës avec changement d'une seule variable
- On parle de case adjacente
- Exemple case DCBA

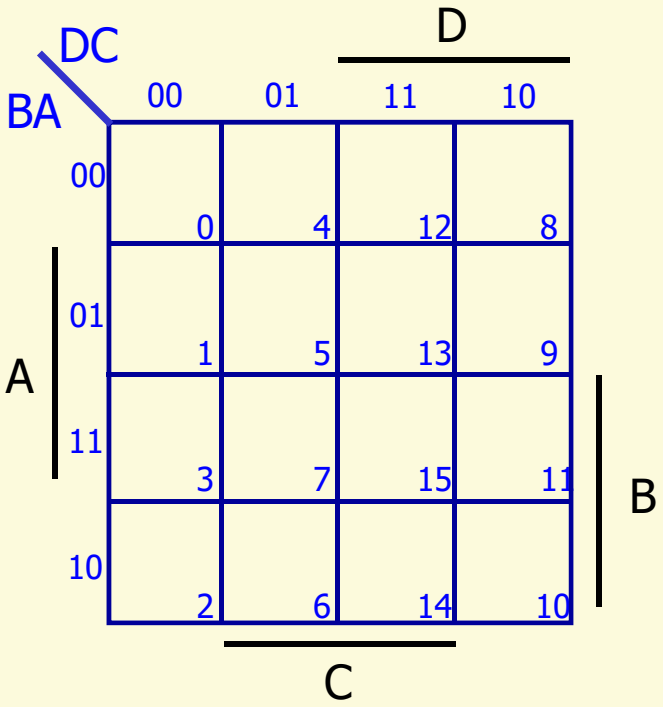




f(B,A)

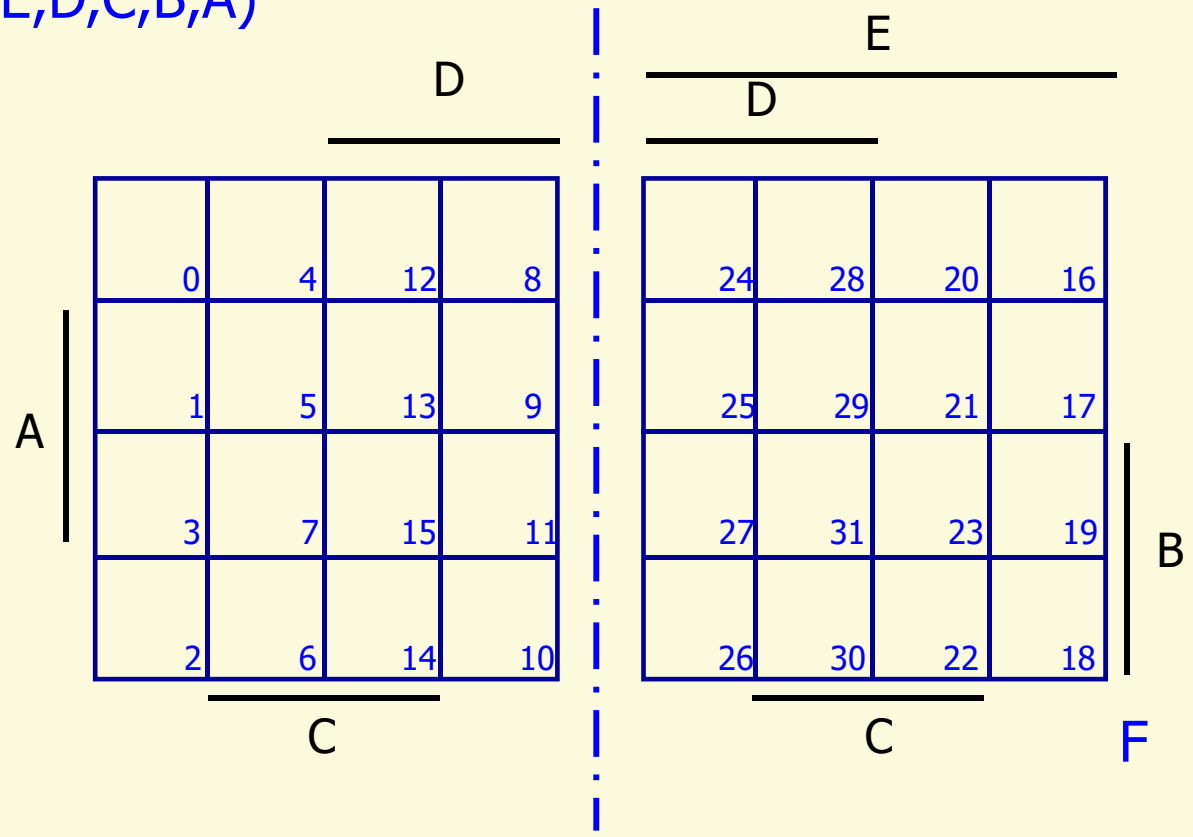


f(C,B,A)



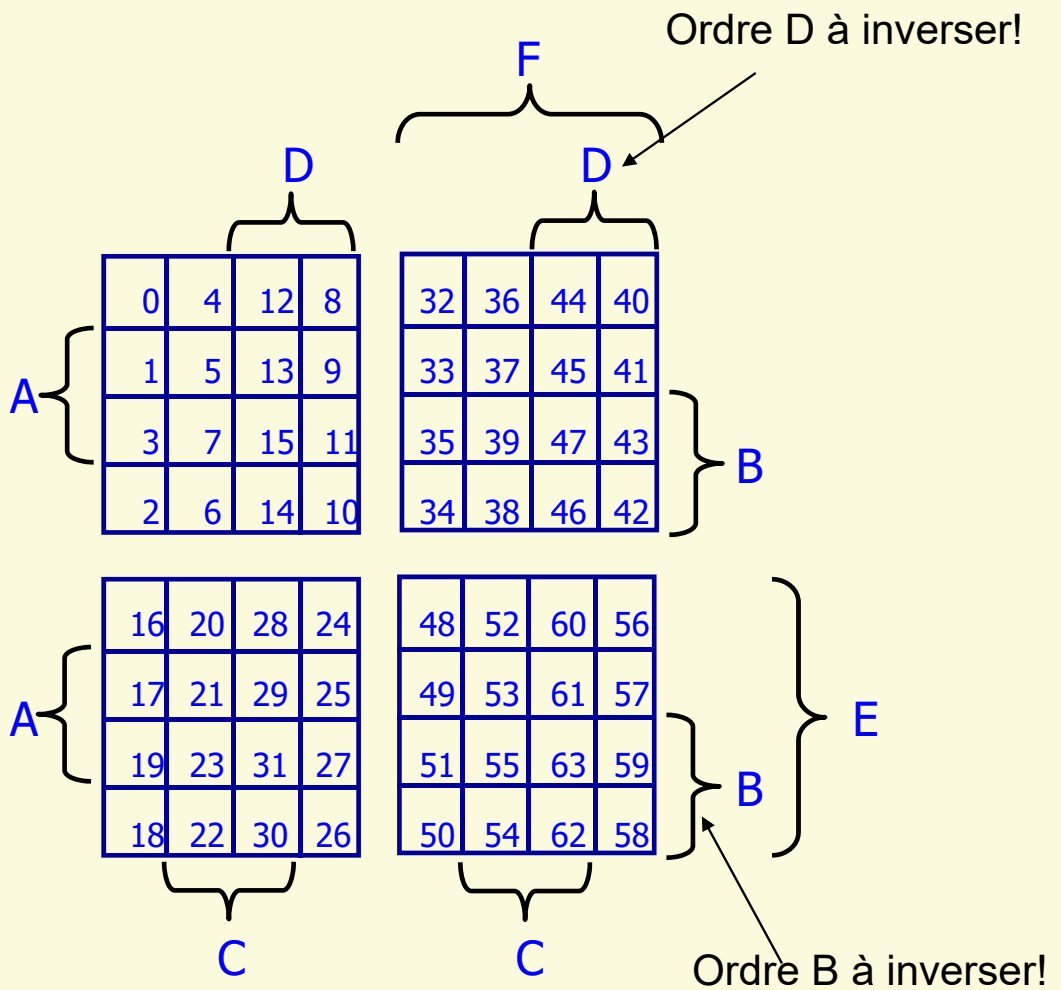
f(D,C,B,A)

F(E,D,C,B,A)



Pour table de Karnaugh à 5 et plus:

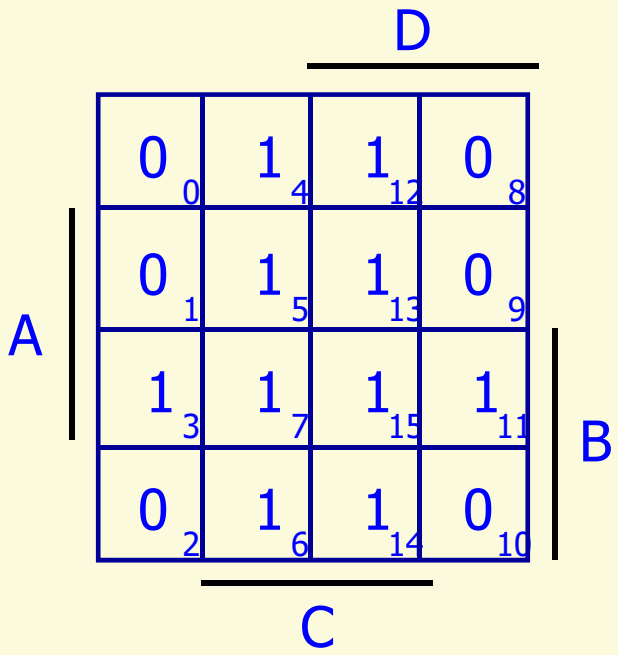
Simplification automatisée: http://www.32x8.com/


Ordre D à inverser!
F
D
D
A
B
A
B
E
C
C
Ordre B à inverser!
0 4 12 8
1 5 13 9
3 7 15 11
2 6 14 10
32 36 44 40
33 37 45 41
35 39 47 43
34 38 46 42
16 20 28 24
17 21 29 25
19 23 31 27
18 22 30 26
48 52 60 56
49 53 61 57
51 55 63 59
50 54 62 58

f(F,E,D,C,B,A)

Représentation d'une fonction …

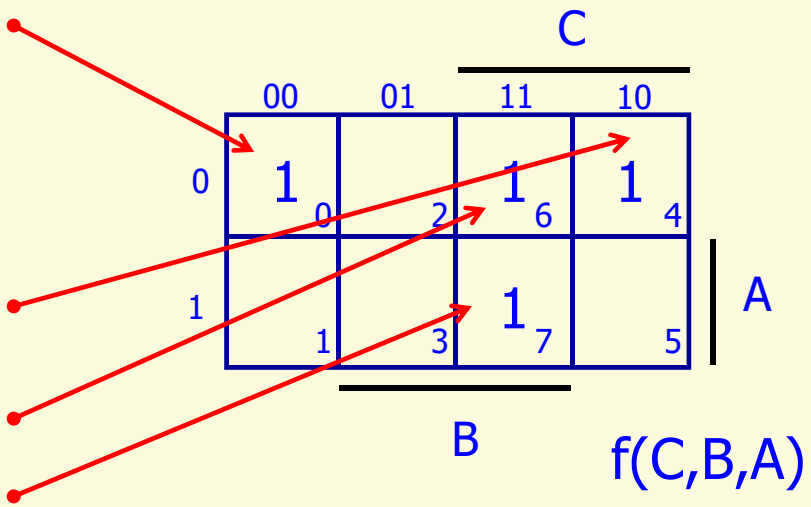
- Une fonction est représentée à l'aide d'une table de Karnaugh en mettant la valeur de la fonction à l'intérieur de chaque case
- Exemple: Z(D,C,B,A) = ∑(3,4,5,6,7,11,12,13,14,15)



… représentation d'une fonction

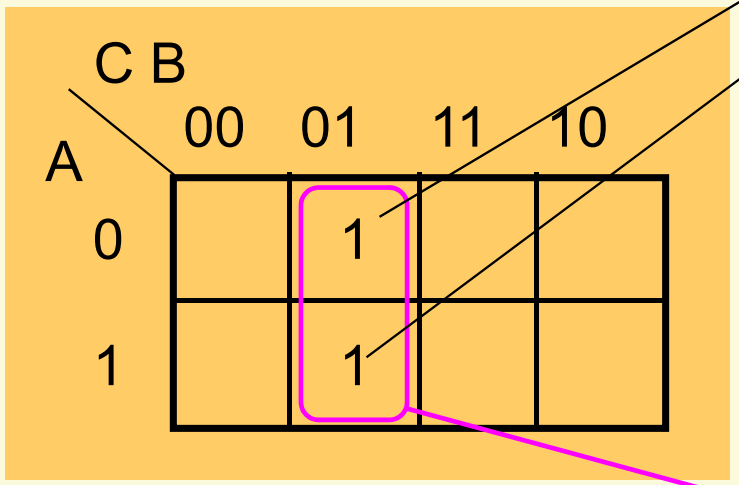
- Exemple à partir d'une table de vérité:

C	B	A	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



Simplification par Karnaugh …

- Simplification basée sur l'application graphique de l'algèbre de Boole



terme /C • B • /A

terme /C • B • A

Démonstration par Boole :

somme des 2 termes:

/C • B • /A + /C • B • A =

/C • B • (/A + A) = /C • B

groupe /C • B

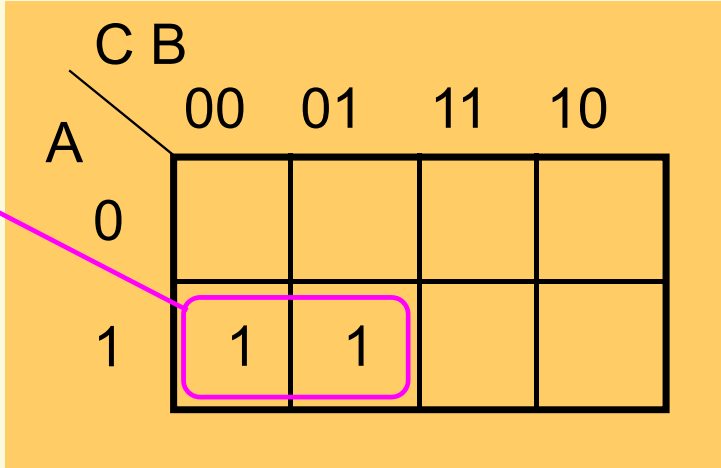
… simplification par Karnaugh …

- Un groupe de deux '1' adjacents (ayant une frontière commune non réduite à un point) s'exprime sous la forme d'un produit comportant N-1 variables
- Par analogie :
  - le groupement de deux groupes adjacents de deux '1' s'exprime sous la forme d'un produit comportant N-2 variables, etc…

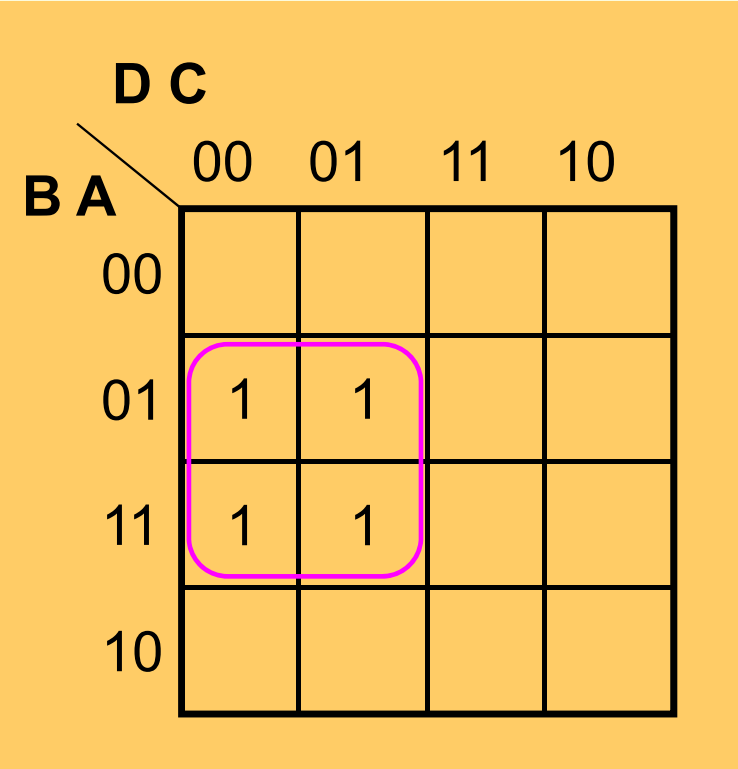
… simplification par Karnaugh …

- L'expression d'un groupe donne ses caractéristiques
- Le groupe ci-dessous est caractérisé par le fait qu'il est contenu dans A ('1') mais à l'extérieur de C ('0'). Donc il s'exprime par

/C • A



… simplification par Karnaugh …



- Expression :
  - groupe de quatre '1'
    → produit de N-2 variables
    (4-2=2)
  - caractéristiques :
    dans A et à l'extérieur de D
    → /D • A

… simplification par Karnaugh …

- Résumé :
  - groupe de deux '1' => produit de N - 1 variables
  - groupe de quatre '1' => produit de N - 2 variables
  - groupe de huit '1' => produit de N - 3 variables

  finalement :
  - groupe de 2^N '1' => 1

- un groupe est nommé: Impliquant d'une fonction
  - il s'agit du produit des variables de la fonction.
    Chaque impliquant est représenté dans la table de Karnaugh par un groupe de cases contiguës, en un nombre égal à une puissance de deux

… simplification par Karnaugh

- La table de Karnaugh permet l'obtention de l'équation minimale d'une fonction logique, sous la forme d'une somme de produits
- En effet, tout produit logique correspond à un ensemble de cases contiguës dont le nombre est une puissance de 2
- Si le nombre de variables de la fonction est égal à n et le nombre de cases contiguës est égal à 2^m, le produit correspondant aura seulement n-m variables
- La somme de produits minimale d'une fonction correspond donc à l'ensemble minimal de groupes de cases de la table de Karnaugh où la fonction est égale à 1. Le nombre de cases de chaque groupe doit être une puissance de 2

Marche à suivre avec Karnaugh …

- Impliquant premier:
  impliquant qui n'est pas inclus dans un autre impliquant plus grand. La solution minimale d'une fonction est formée seulement d'impliquants premiers. Mais tous les impliquants premiers ne font nécessairement pas partie de la solution minimale.
- Impliquant premier essentiel:
  impliquant premier qui contient au moins un minterme qui n'est pas inclus dans un autre impliquant premier. Un impliquant premier essentiel fait toujours partie de la solution minimale.

… marche à suivre avec Karnaugh

Méthode de minimisation

- Dresser la liste de tous les impliquants premiers de la fonction
- Dresser la liste de tous les impliquants premiers **essentiels**
- Tous les impliquants premiers **essentiels** font partie de la solution minimale
- Couvrir les mintermes restants avec un nombre minimal d'impliquants premiers

Résumé marche à suivre avec Karnaugh

- Pour trouver l'expression en somme de produits la plus simple
  - rechercher les plus grands groupes possibles (premiers et premiers essentiels)
  - prendre tous les groupes contenant des '1' isolés et non groupé
    - '1' isolé = '1' qui n'entre que dans un seul groupe
  - ajouter les groupes les plus grands qui incluent des '1' n'ayant pas encore été pris
  - jusqu'à ce que tous les '1' aient été pris
  - un '1' peut faire partie de plusieurs groupes (1 ou 1 = 1), mais il suffit qu'il soit pris une seule fois

Exercices II

a. Ecrivez la table de vérité, cherchez l'expression simplifiée en somme de produits puis dessinez le logigramme de la fonction majorité à 3 entrées.
   MAJ = 1 si la majorité des 3 entrées est à 1.

b. Dessinez le logigramme ci-dessus en utilisant un nombre minimum de NAND à 2 entrées (à l'exclusion de toute autre fonction).

c. Dessinez le logigramme ci-dessus en utilisant un nombre minimum de NOR à 2 entrées (à l'exclusion de toute autre fonction).

… exercices II …

- Donner l'équation la plus simple pour la fonction Z donnée dans la table de Karnaugh ci-dessous:

B A \ D C	00	01	11	10
00	1	0	1	1
01	0	0	1	0
11	1	0	1	0
10	0	1	1	0

Z

… exercices II …

- Donner l'équation la plus simple pour la fonction S donnée dans la table de Karnaugh ci-dessous:

B A \ D C	00	01	11	10
00	1	0	0	1
01	0	0	1	1
11	1	0	1	1
10	1	1	1	0

S

… exercices II

- Donner l'équation la plus simple pour la fonction T donnée dans la table de Karnaugh ci-dessous:

B A \ D C	00	01	11	10
00	1	0	1	1
01	1	1	0	0
11	0	1	1	0
10	1	1	1	1

T

Fonction incomplètement définie

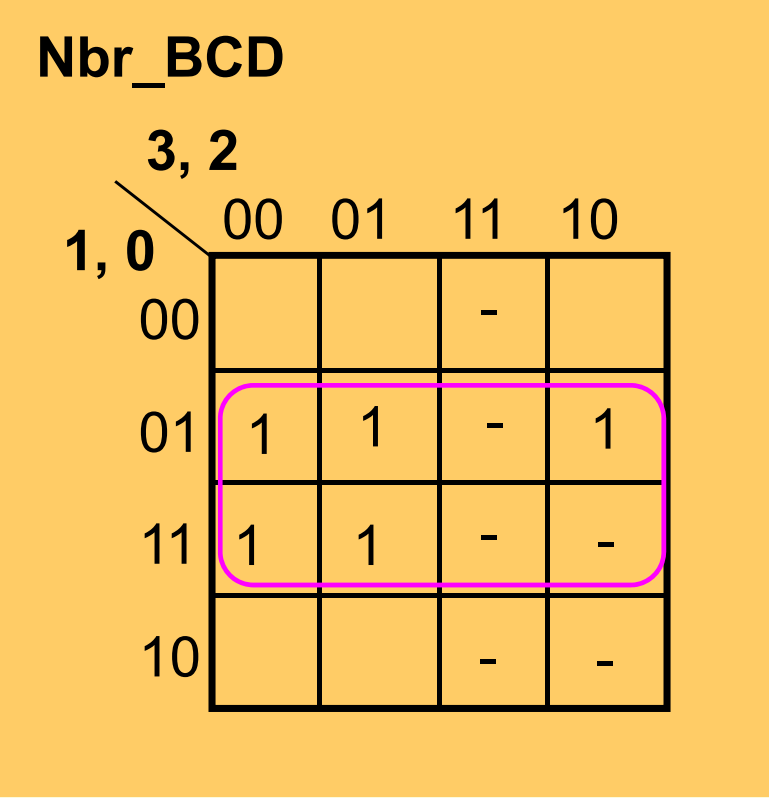
- Fréquemment, il y a des combinaisons des entrées qui n'existent pas. Dès lors l'état de sortie n'est pas défini.
- Nous pouvons alors choisir l'état de la fonction pour les combinaisons d'entrées correspondantes
- Nous l'indiquerons par : - ou Ø (état indifférent)
- Dans la table de Karnaugh, l'état indifférent pourra être considéré comme '0' ou '1'
  - Sert à créer des impliquants les plus grands possibles

Valeur impaire d'un nombre BCD

- La fonction de sortie Impaire n'est définie que pour les combinaisons BCD
- Il y a 6 combinaisons où la fonction n'est pas spécifiée
  => choix état de sortie => -
  pour simplifier celle-ci
- Déterminer l'équation simplifiée de la fonction Impaire

Nombre BCD	Impaire
0 0 0 0	0
0 0 0 1	1
0 0 1 0	0
0 0 1 1	1
0 1 0 0	0
0 1 0 1	1
0 1 1 0	0
0 1 1 1	1
1 0 0 0	0
1 0 0 1	1
1 0 1 0	-
1 0 1 1	-
1 1 0 0	-
1 1 0 1	-
1 1 1 0	-
1 1 1 1	-

… valeur impaire d'un nombre BCD

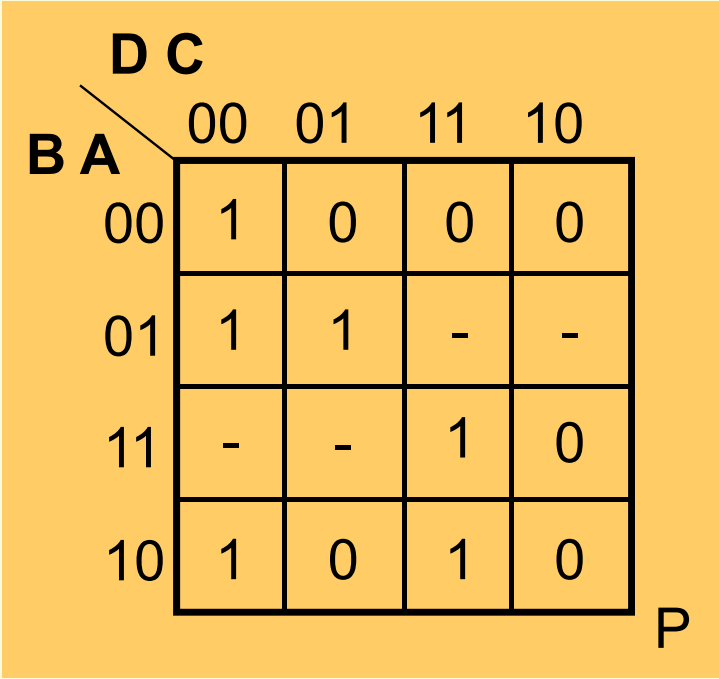


Expression :

Impaire = Nbr_BCD(0)

Exercices III

- Donner l'équation la plus simple pour la fonction P donnée dans la table de Karnaugh ci-dessous:



… exercices III …

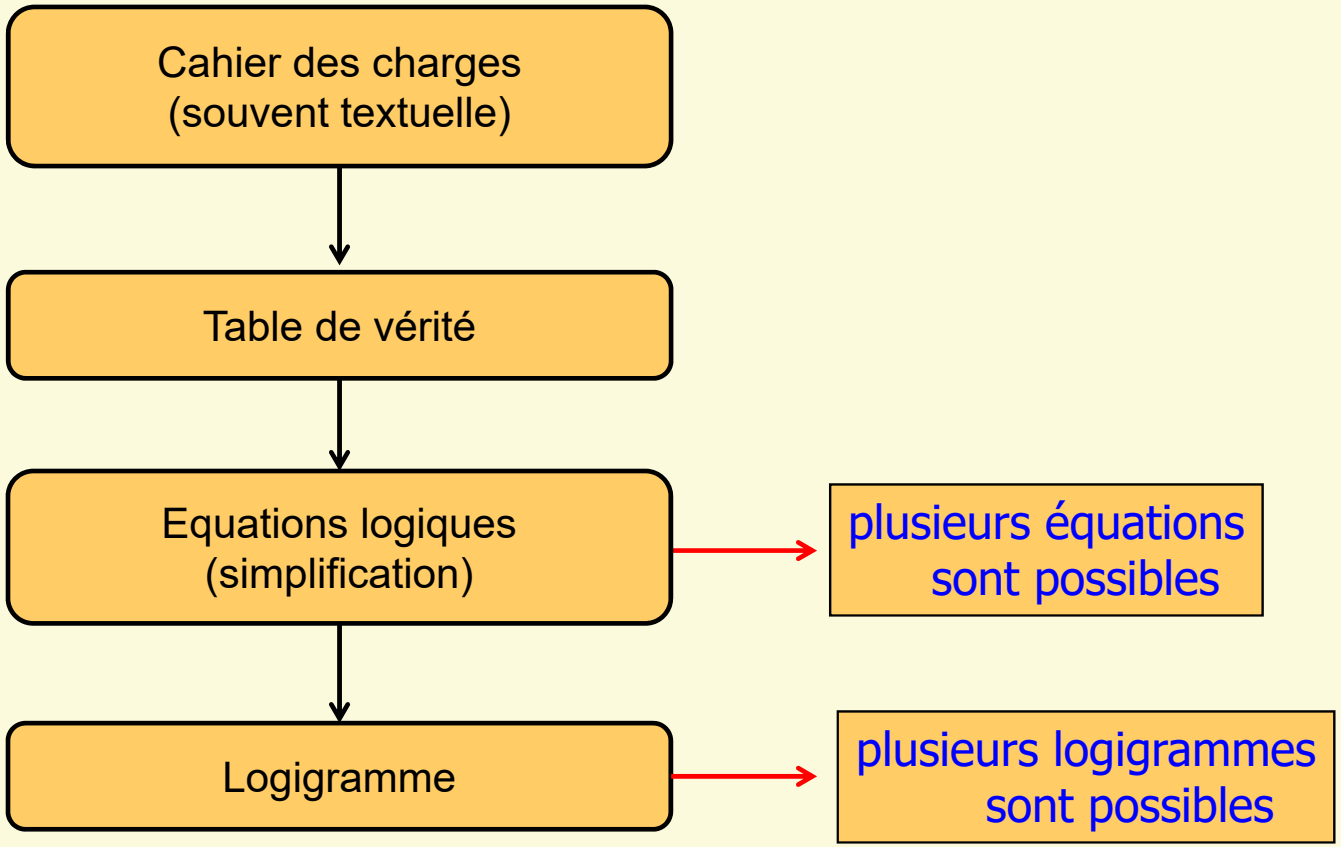
- Donner l'équation la plus simple pour la fonction F donnée dans la table de Karnaugh ci-dessous:

B A \ D C	00	01	11	10
00	1	0	0	1
01	0	0	0	-
11	-	0	1	1
10	1	-	1	-

F

dia volontairement vide

Conception de systèmes logiques combinatoires

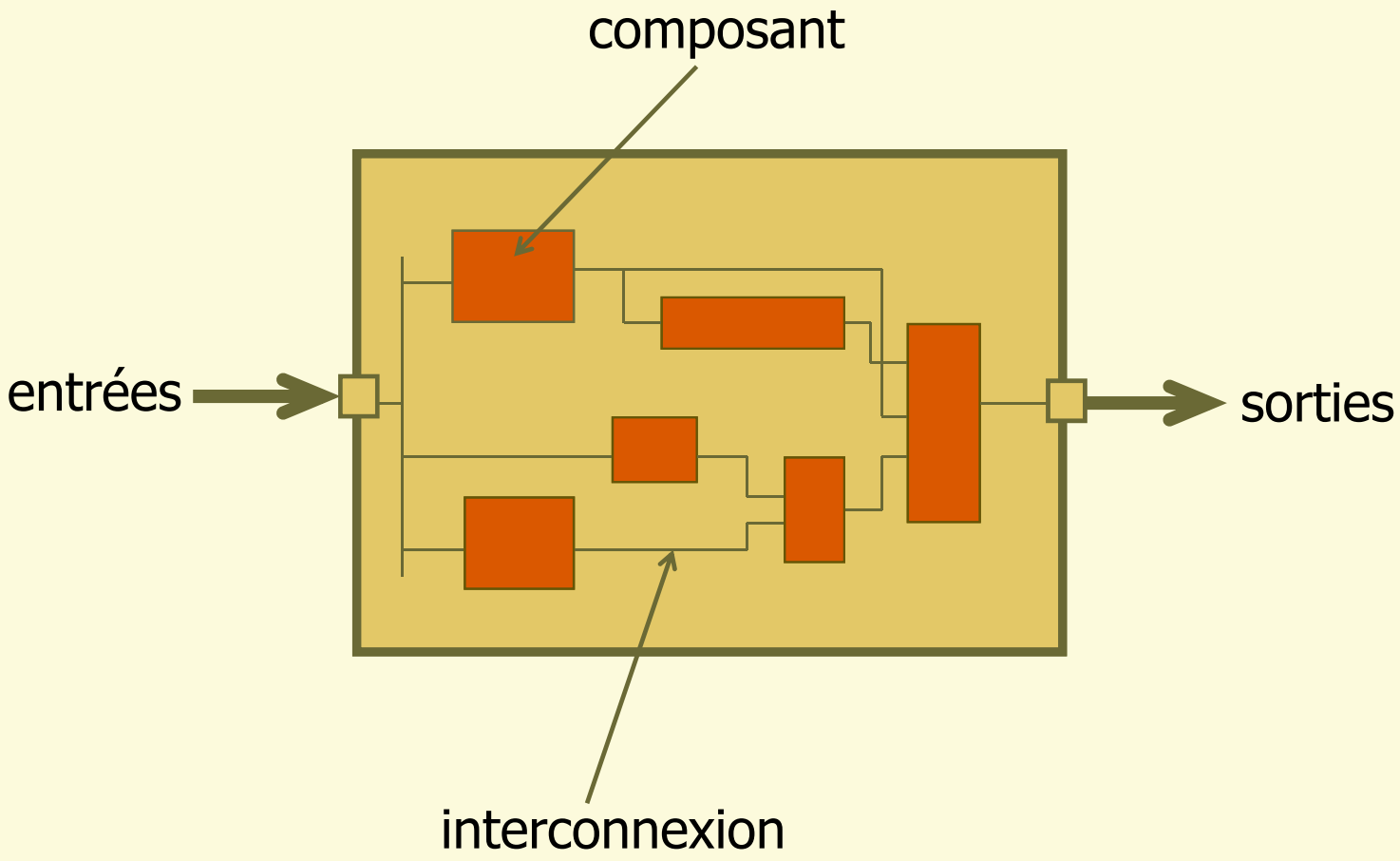


Notion de Système

Un système est une collection organisée d'objets qui interagissent pour former un tout :

- Objets = composants du système
- Interconnexions = liens entre les objets nécessaires pour les interactions
- Structure = organisation du système (composants, interconnexions)
- Comportement = fonctionnement du système (entrées, sorties)

Vue d'un système



Réalisation d'un système

- Analyse:
  - Déterminer le comportement d'un système à partir d'une description textuelle
  - Déterminer les entrées/sorties
- Conception:
  - Déterminer la structure nécessaire qui produit un comportement donné.
  - Plusieurs structures sont possibles pour obtenir un même comportement (entrées – sorties)

Solution logicielle ou matérielle

- Tout traitement réalisé par un programme peut être réalisé par un composant matériel
- Solution logicielle
  - souplesse
  - grande capacité de traitement (beaucoup de données)
  - séquentiel, peu rapide
- Solution matérielle
  - très rapide
  - nécessite beaucoup de matériel (composants logiques)
  - temps de développement important

Les niveaux d'abstraction

- L'abstraction se réfère à la distinction entre les propriétés externes d'un système et les détails de sa composition interne
- L'abstraction d'un système comprend:
  - la suppression de certains détails pour montrer seulement l'essence du sujet (pour chaque niveau d'abstraction il faut pouvoir différencier ce qui est essentiel des détails superflus)
  - une structure
  - une division en sous-systèmes (composants)

Réalité versus Abstraction

- L'abstraction est utile et nécessaire, mais il ne faut pas oublier la réalité
- Les abstractions possèdent toujours des limites qu'il faut connaître

Exemples

Les **`int (integer)`** ne sont pas des entiers et les **`float`** ne sont pas des réels

- Est-ce que $x^2 \geq 0$ est toujours vrai?
  - pour les **`float`**, oui
  - pour les **`int`** (32 bits signé en C2), pas toujours:
    - **`40'000 * 40'000`** → **`1'600'000'000`**
    - **`50'000 * 50'000`** → ??
- Est-ce que **`(x+y)+z = x+(y+z)`** est toujours vrai?
  - pour les **`integer`**, oui
  - pour les **`float`**, pas toujours:
    - **`(1e20 + -1e20) + 3.14`** → **`3.14`**
    - **`1e20 + (-1e20 + 3.14)`** → ??