10. Réalisation d'un circuit combinatoire

Principes de fonctionnement des ordinateurs

Jonas Lätt

Centre Universitaire d'Informatique

Trouvé une erreur sur un transparent? Envoyez-moi un message

- *sur Twitter @teachjl ou*
- *par e-mail jonas.latt@unige.ch*

Contenu du cours

Partie I: Introduction

Partie II: Codage de l'information

Partie III: Circuits logiques

Partie IV: Architecture des ordinateurs

1. Introduction
2. Histoire de l'informatique
3. Information digitale et codage de l'information
4. Codage des nombres entiers naturels
5. Codage des nombres entiers relatifs
6. Codage des nombres réels
7. Codage de contenu média
8. Portes logiques
9. Circuits logiques combinatoires et algèbre de Boole
10. **Réalisation d'un circuit combinatoire**
11. Circuits combinatoires importants
12. Principes de logique séquentielle
13. Réalisation de la bascule DFF
14. Architecture de von Neumann
15. Réalisation des composants
16. Code machine et langage assembleur
17. Réalisation d'un processeur
18. Performance et micro-architecture
19. Du processeur au système

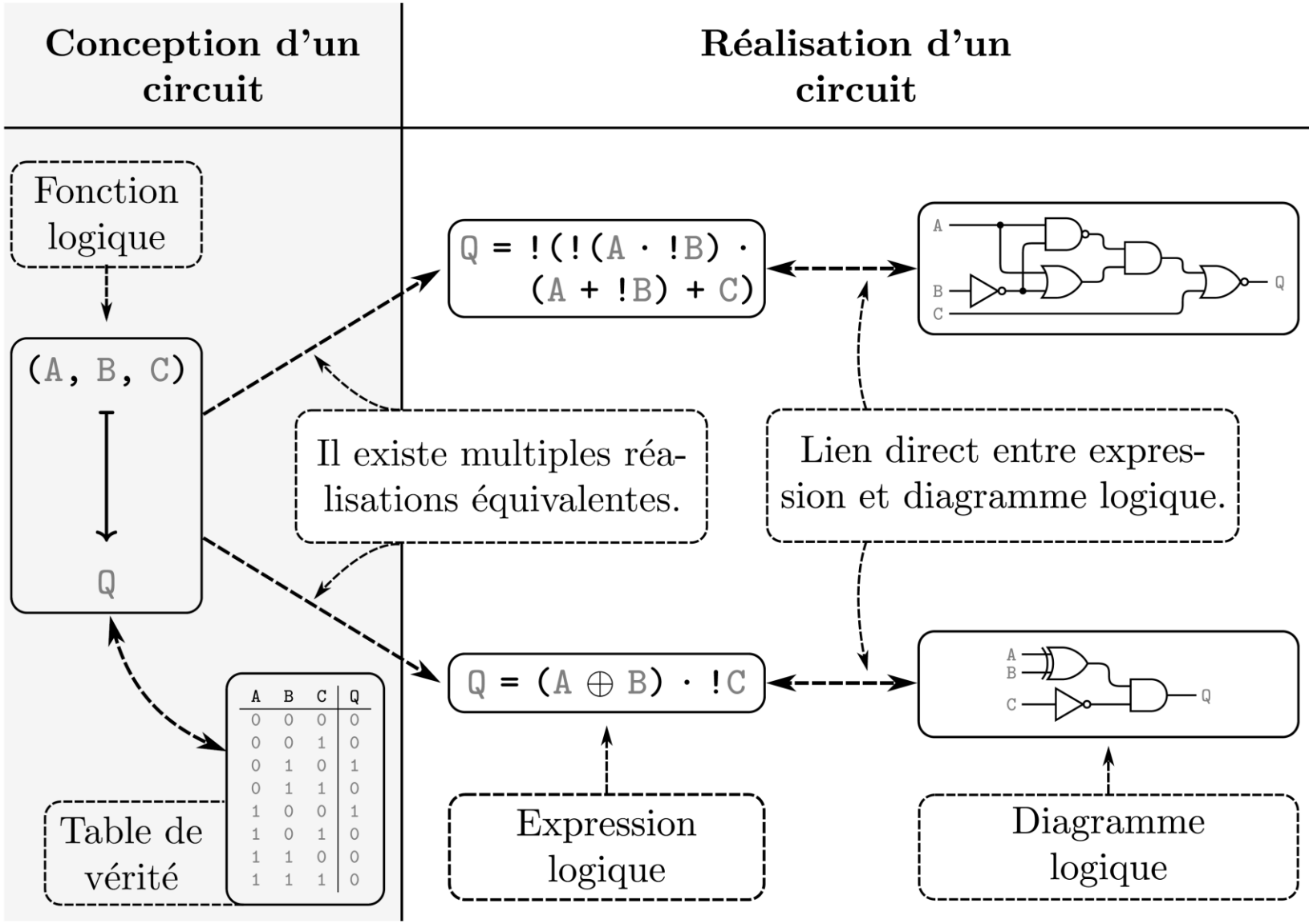
Rappel

- Un **circuit logique combinatoire** implémente une fonction logique.

- **M bits de sortie** S_j, j=1..M dépendent uniquement de **N bits d'entrée** E_i, i=1…N.

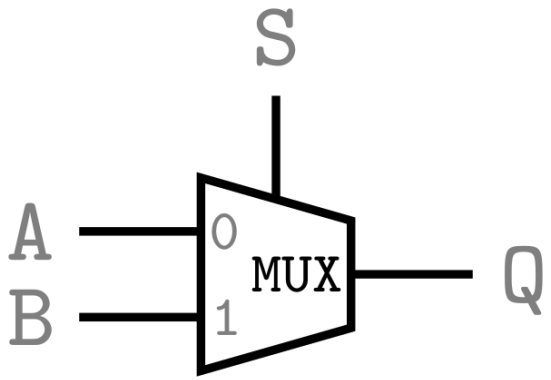
$S_j=f_j(E_1, E_2,…E_N)$ pour tout j=1...M

Conception vs Réalisation d'un circuit

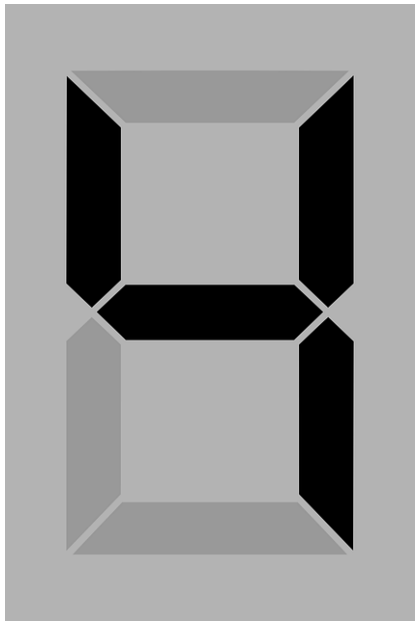


Maintenant: deux circuits concrets

Multiplexeur

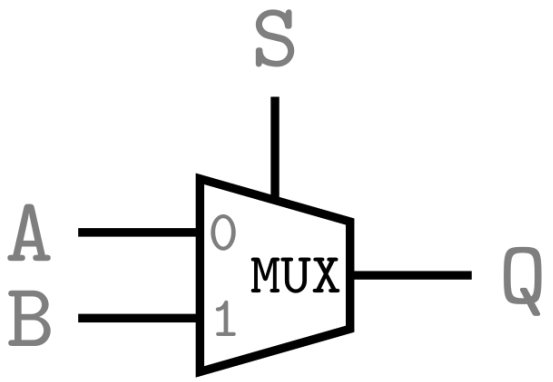


Circuit Affichage 7-Segments



1. Ecriture de la table de vérité du circuit.
2. Réalisation du circuit: construction d'une expression booléenne.

Le multiplexeur à deux voies

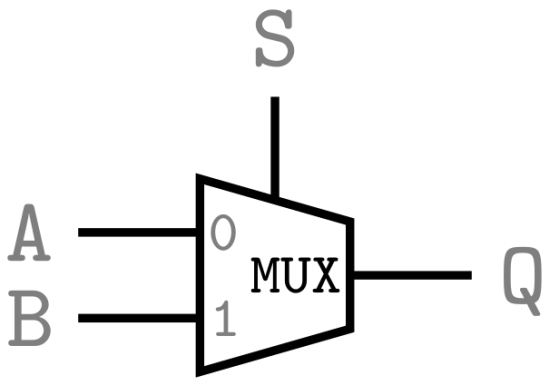


- **Q** est égal à **A** si **S** vaut **0**.
- **Q** est égal à **B** si **S** vaut **1**.

Réalisation électronique d'une sélection («*if-else*»):

```
si S vaut 0
    Q = A
autrement
    Q = B
```

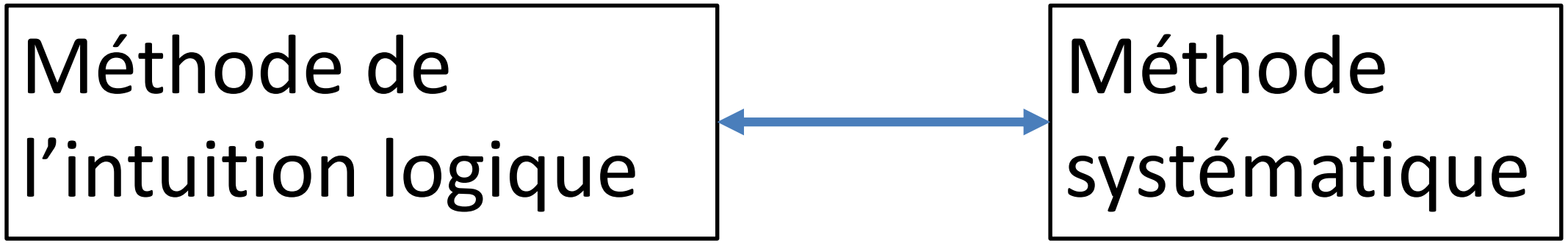
Le multiplexeur à deux voies




```
si S vaut 0
    Q = A
autrement
    Q = B
```

S	A	B	Q=MUX (S,A,B)

Comment réaliser un circuit?

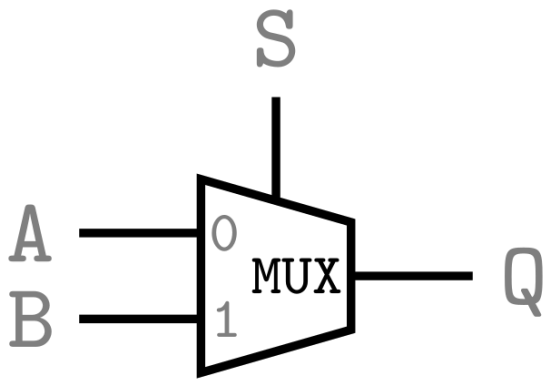


Rappel: méthode de l'intuition logique

D est vrai si (M est vrai et A est vrai) ou (M est vrai et C est vrai) ou P est vrai

$$D = (M \cdot A) + (M \cdot C) + P$$

Trouver une expression booléenne: Méthode de l'intuition logique

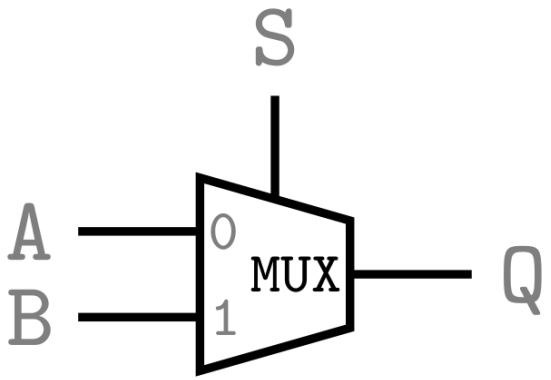


si **S** vaut **0**
Q = **A**
autrement
Q = **B**

Méthode:

On énonce une phrase à haute voix qui commence par **Q** vaut **1** si et seulement si …

Trouver une expression booléenne: Méthode de l'intuition logique



- **Q** est égal à **A** si **S** vaut **0**.
- **Q** est égal à **B** si **S** vaut **1**.

Q vaut **1** si et seulement si

- L'entrée sélectionnée par **S** vaut **1**

Q vaut **1** si et seulement si

- **S** vaut **0** ET **A** vaut **1** OU
- **S** vaut **1** ET **B** vaut **0**

Méthode systématique: La méthode des minterms et des maxterms

Idée: la méthode de l'intuition logique est appliquée à la table de vérité

Exemple: un circuit pour la porte XOR

Méthode des minterms

Problème: Table de vérité -> Expression Booléenne dans le cas général.
Solution: Méthode de l'intuition logique appliquée à la table de vérité.

Exemple: XOR

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Q vaut 1 si et seulement si…

Ce procédé s'appelle la Méthode des Minterms

Méthode des minterms

1. Rechercher les lignes dans lesquelles Q vaut 1.
2. Ecrire un minterm pour chacune de ces lignes. Un minterm contient toutes les variables d'entrée, inversées par un NOT si la variable d'entrée vaut 0 dans la ligne en question, combinées à l'aide de AND.
3. Exprimer Q comme somme logique (OR) de ces Minterms.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Q est **vrai** si et seulement si

A est **faux ET** B est **vrai**
OU
A est **vrai ET** B est **faux**

Méthode des minterms: commentaires

- Une fonction logique peut être réalisée par une infinité d'expressions, mais l'expression obtenue par minterms est unique (à part des permutations).
- Cette expression peut être vue comme une manière alternative d'écrire la table de vérité.
- Noms utilisés pour cette expression:
  - **Forme canonique** de l'expression logique
  - Forme **somme-de-produits** de l'expression logique

Equivalence de Q3 et Q5 du chapitre précédent

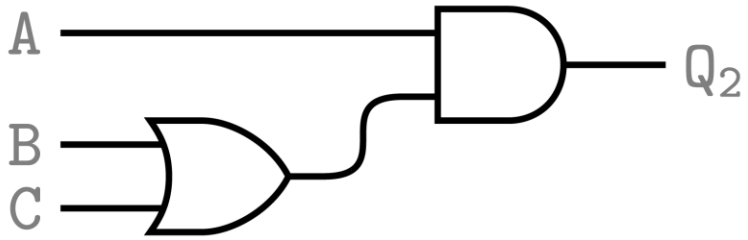
$$Q_3 = !(!(A \cdot !B) \cdot (A + !B) + C)$$

$$Q_5 = (A \oplus B) \cdot !C$$

Exemple: Ecrivons la forme canonique de Q2

A	B	C	Q_2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$Q_2 = A \cdot (B + C)$$



Alternative: Méthode des maxterms

Idée: on se concentre sur les cas Q = 0

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Q vaut 0 si et seulement si…

Méthode des maxterms

1. Rechercher les lignes dans lesquelles **Q** vaut **0**.
2. Ecrire un maxterm pour chacune de ces lignes. Un maxterm contient toutes les variables d'entrée, inversées par un `NOT` si la variable d'entrée vaut **1** dans la ligne en question, combinées à l'aide de `OR`.
3. Exprimer **Q** comme produit logique (`AND`) de ces Minterms.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Q vaut **1** si et seulement si

A vaut **1 OU** B vaut **1**
ET
A vaut **0 OU** B vaut **0**

Minterms: Somme-de-produits

Maxterms: Produit-de-sommes

Minterms vs. Maxterms

Minterms

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

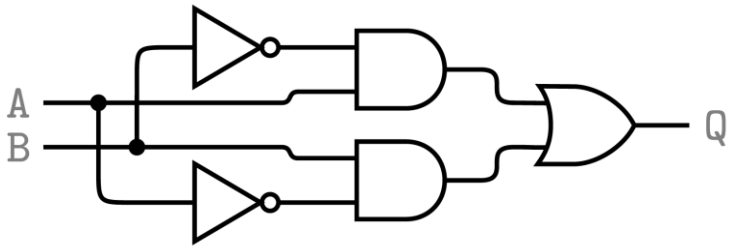
$Q = !A \cdot B + A \cdot !B$

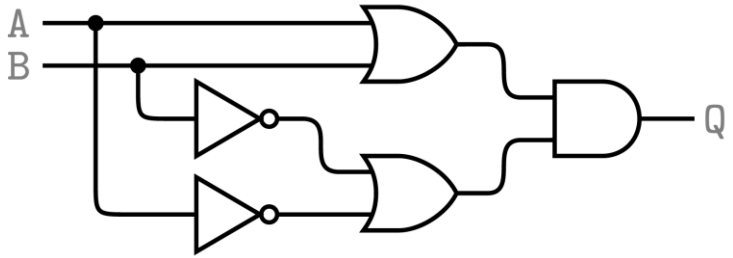
Maxterms

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

$Q = (A+B)\cdot(!A+!B)$

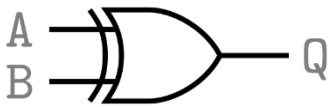
Expressions équivalentes





En général, laquelle des deux choisir? Cela dépend du nombre de 1 dans la colonne Q.

Interprétations de la porte XOR: Retour à l'intuition logique



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Interprétation:

$Q = !A \cdot B + A \cdot !B$

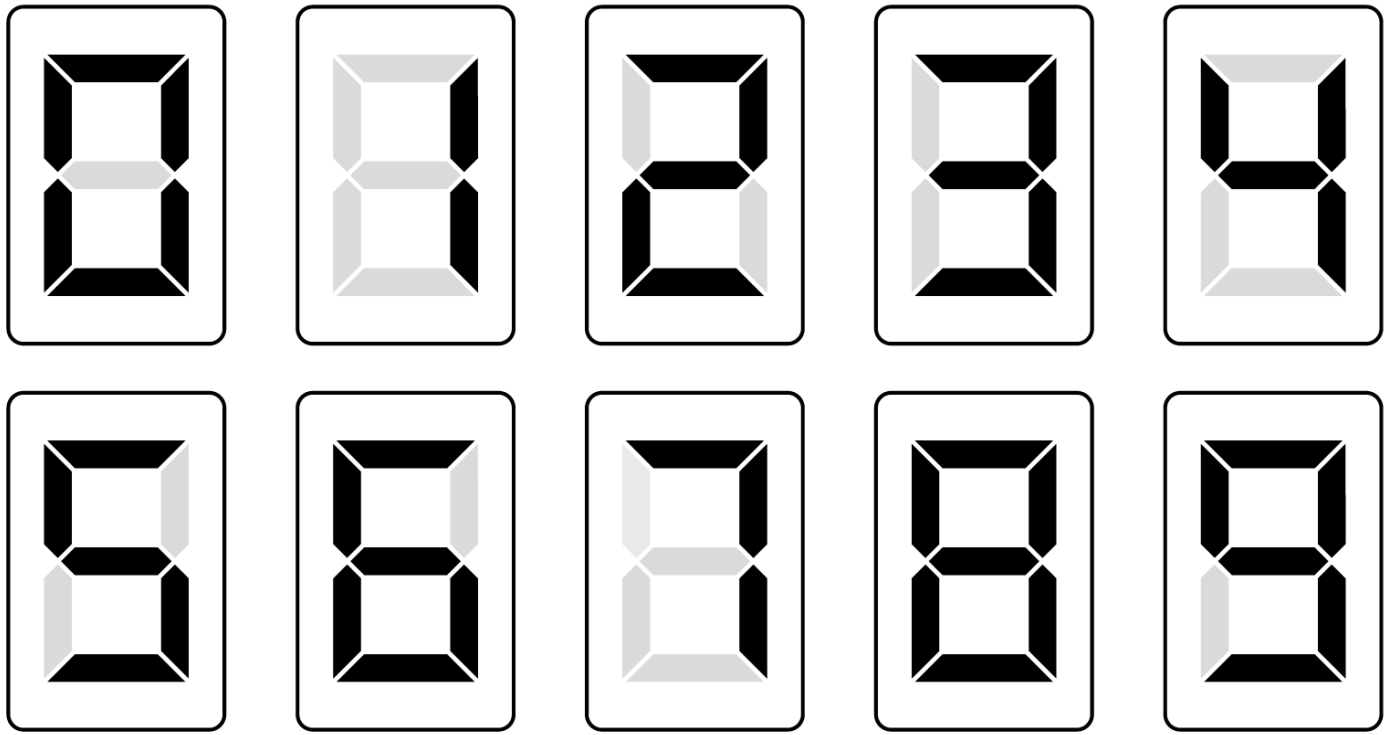
$Q = (A+B)\cdot(!A+!B)$

Equivalent dual:

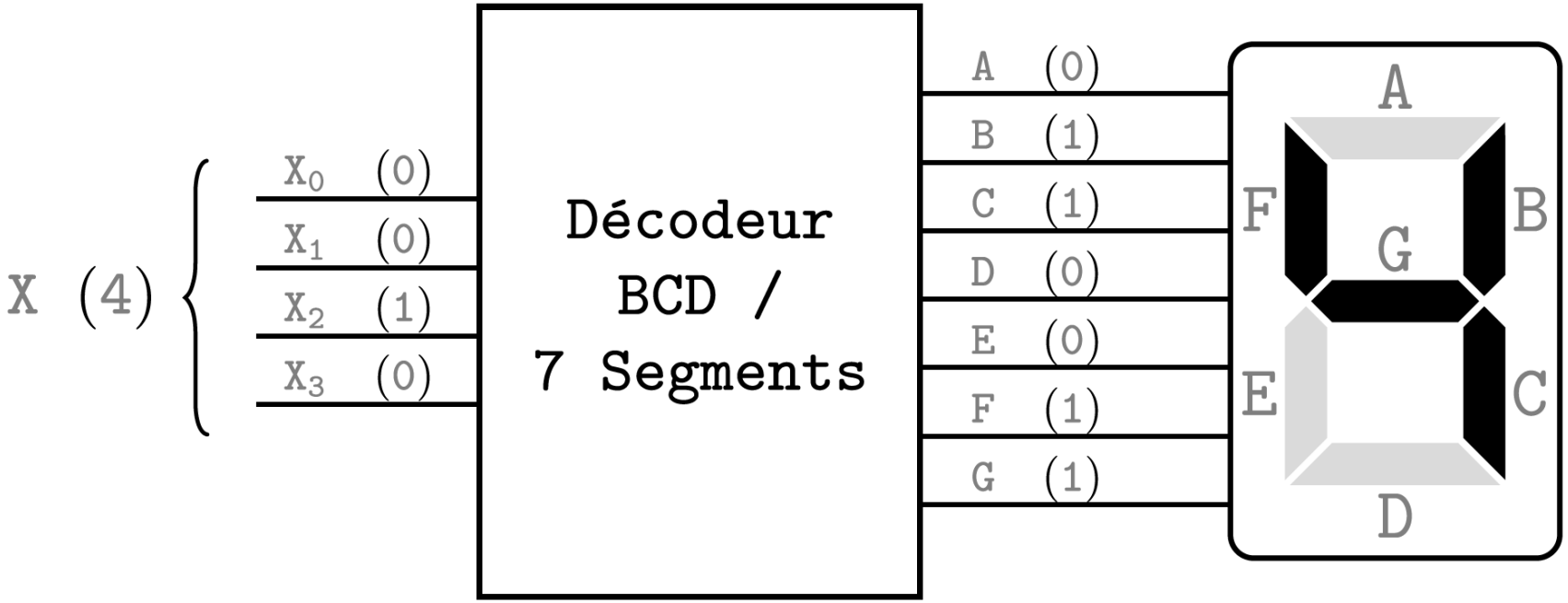
Retour à la méthode des minterms / maxterms

- Circuit de l’afficheur 7-segments: un exemple qui ne se pête pas à l’approche d’intuition logique

Circuit pour l’afficheur 7-segments



Exemple: circuit pour l'afficheur 7-segments



Ici, les entrées du circuit représentent les bits d'un mot à k bits: ce mot représente un nombre entier.
Ca sera le cas pour beaucoup de circuits dans les chapitres à venir.

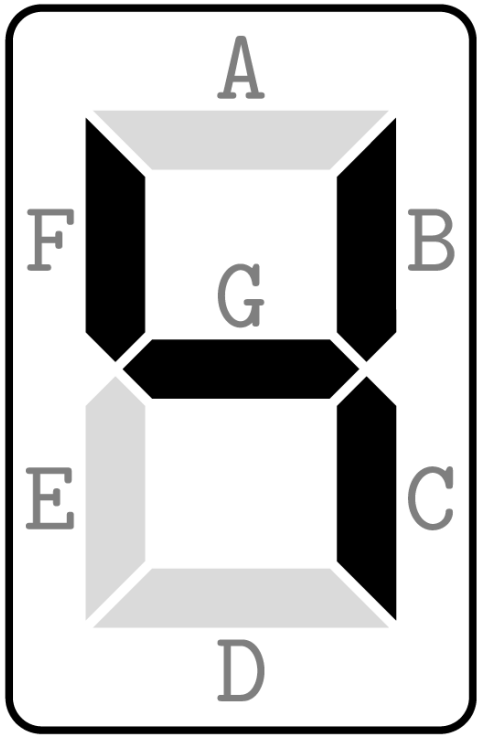
Exemple: Minterms appliqués au circuit 7-segments

- Segment “E” de l’affichage LCD.
- Application des Minterms.

Les Minterms sont:

E =

	X_3	X_2	X_1	X_0	E
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	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	



Exemple: Minterms appliqués au circuit LCD

L'expression trouvée

$E = !X_3\ !X_2\ !X_1\ !X_0 + !X_3\ !X_2\ X_1\ !X_0 + !X_3\ X_2\ X_1\ !X_0 + X_3\ !X_2\ !X_1\ !X_0$

Nécessite 26 portes logiques. Simplifions!

1. Factorisation de $!X_0$:

2. Factorisation de $!X_3\ !X_2$:

3. Complément de l'addition:

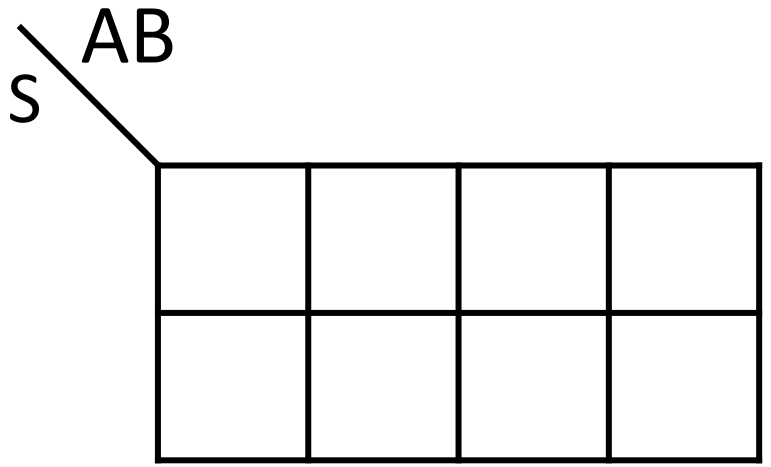
4. Factorisation de $!X_3$:

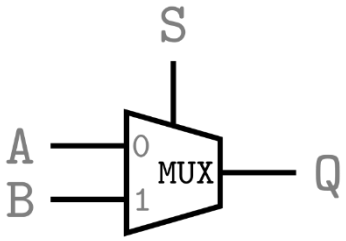
Comment simplifier un circuit de manière systématique?

- Grands circuits, cas général: il n'existe pas de méthode permettant de trouver un circuit optimal en un temps utile.
- Il existe des méthodes heuristiques permettant de trouver des circuits «raisonnablement bons».
- Pour des petits circuits, il existe des méthodes de simplification systématiques. Nous allons présenter les **tables de Karnaugh** (Maurice Karnaugh 1954 @ Laboratoires Bell).

Table de Karnaugh: Table de vérité en format compact

S	A	B	Q=MUX (S,A,B)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



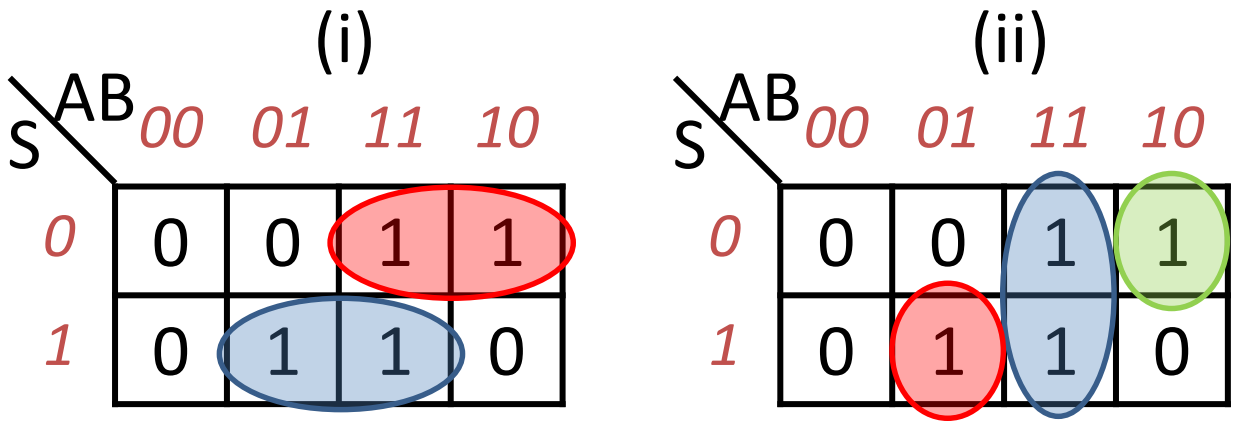


Tables de Karnaugh: Minterms

S \ AB	*00*	*01*	*11*	*10*
0	0	0	1	1
1	0	1	1	0

- On a deux groupes.
- Dans chaque groupe, une variable est éliminée (elle est redondante).
- La méthode des tables de Karnaugh permet d'identifier les variables redondantes.

Tables de Karnaugh: Idée

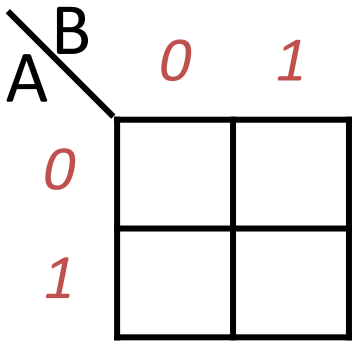


Stratégie:

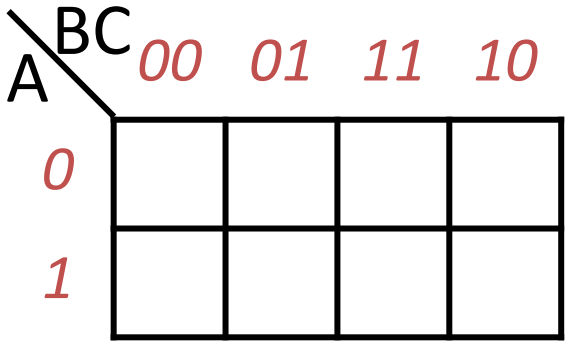
- On regroupe les 1 adjacents en carrés ou rectangles.
- Chaque rectangle donne lieu à un seul Minterm.
- Une variable qui apparait sous forme X et !X est redondante: on l'élimine.

Tables de Karnaugh: formes

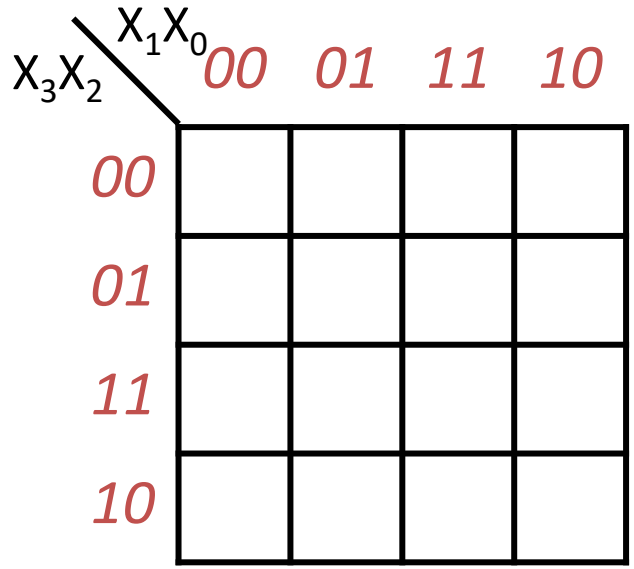
2 entrées:



3 entrées:

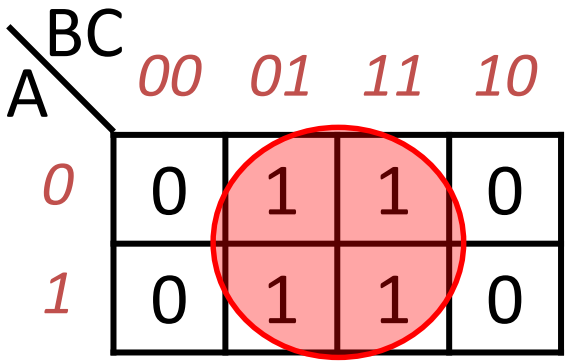


4 entrées:



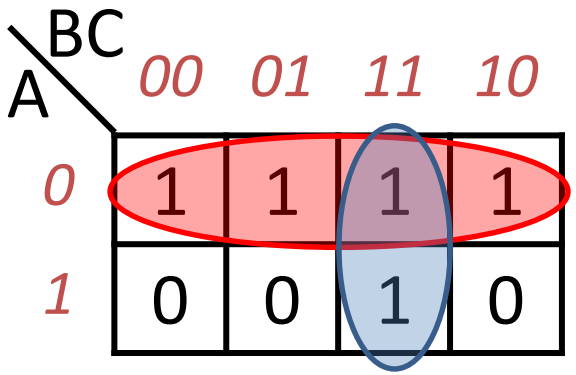
Important: les entrées ne sont pas en ordre binaire. On les ordonne de manière à ce qu'un seul bit change à la fois: le 01 est suivi du 11 et non pas du 10. Cela est aussi valable le long du bord: le 10 est suivi du 00, avec un seul changement de bit.

Tables de Karnaugh: règles



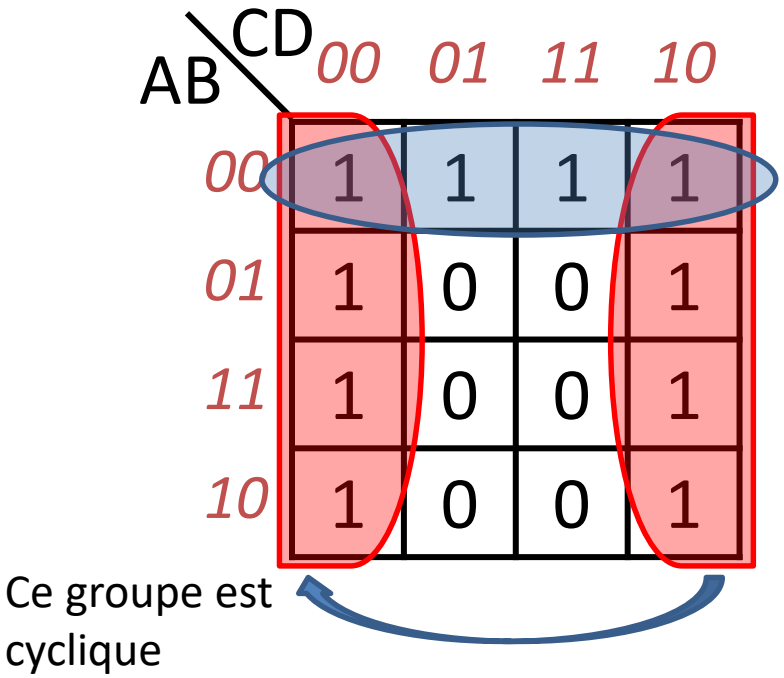
- Il ne peut y avoir que des 1 dans un groupe. Les 0 ne sont pas permis.
- Un groupe doit être rectangulaire.
- Le nombre de 1 dans le groupe doit être une puissance de 2 (1, 2, 4, 8, 16).

Tables de Karnaugh: règles



- Les groupes peuvent se chevaucher.
- Les chevauchements sont souhaitables: groupes plus grands = davantage de redondance

Tables de Karnaugh: règles



- Les groupes peuvent se construire de manière cyclique (haut/bas ou gauche/droite).

Tables de Karnaugh pour Minterms: résumé

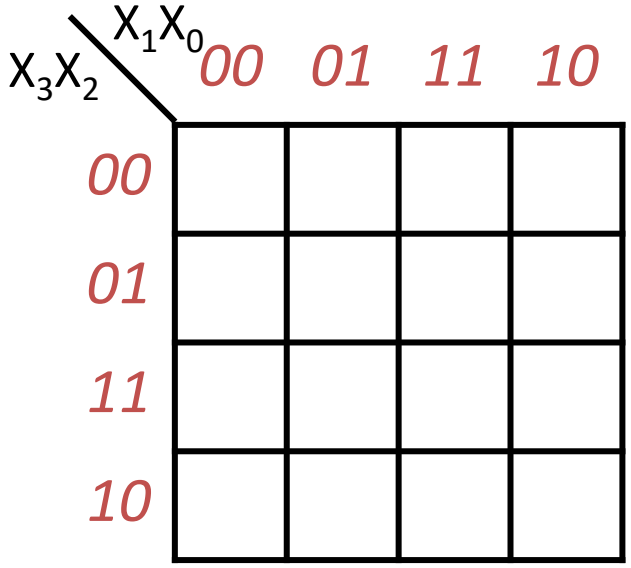
- Ce qu'on vient de faire, c'est la méthode des «**tables de Karnaugh pour les minterms**», car on encercle des groupes de «1».
- Faites des groupes aussi grands que possible!
- Faites aussi peu de groupes que possible!

Tables de Karnaugh pour Maxterms

«C'est la même chose, sauf que c'est partout l'inverse»:

- Faites des groupes de «0».
- Quand vous prenez une variable, inversez-la: X pour 0 et !X pour 1.
- A l'intérieur d'un groupe, enchaînez les variables par des «+» (et non pas par des «*», comme pour les Minterms).
- Enchaînez les groupes par des «*» (et non pas par des «+», comme pour les Minterms).

Segment E de l'affichage 7-segments: Minterms



	X_3	X_2	X_1	X_0	E
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
	1	0	1	0	*
	1	0	1	1	*
	1	1	0	0	*
	1	1	0	1	*
	1	1	1	0	*
	1	1	1	1	*

Segment E de l'affichage 7-segments: Maxterms

X_3X_2 \ X_1X_0	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	*	*	*	*
10	1	0	*	*

Dernière remarque intéressante: Universalité des portes logiques

Portes logiques nécessaires

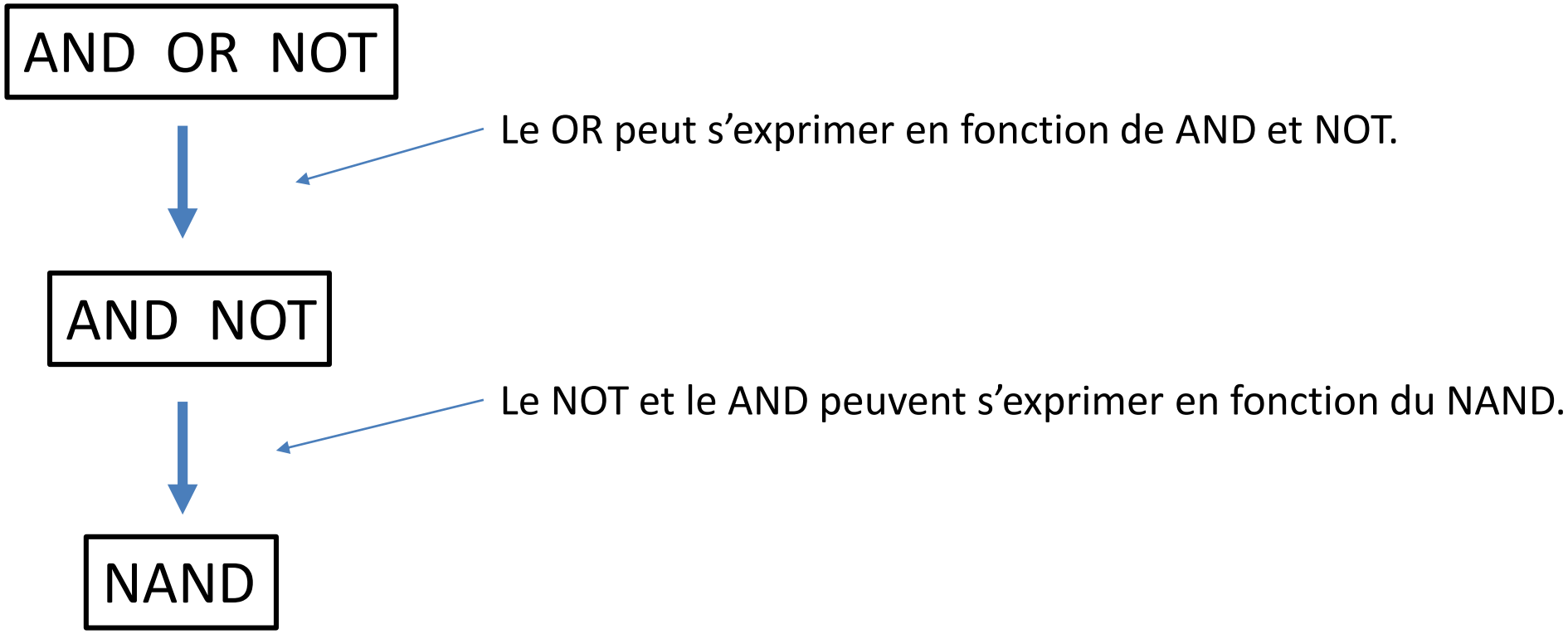
- Avant de terminer ce chapitre, nous répondons à la **Question** fondamentale: de combien de types de portes logique différents a-t-on vraiment besoin pour construire n'importe quell circuit?

- **Première réponse**: Nous savons que tout circuit peut s'exprimer en terme de

AND, **OR**, **NOT**

- En effet, il suffit d' écrire la table de vérité du circuit, puis d'appliquer la règle des Minterms ou des Maxterms.

On peut construire n'importe quel circuit à l'aide du NAND



Argument équivalent: on peut construire n'importe quel circuit à l'aide du NOR.

Exercice

Les portes suivantes sont-elles universelles?

- Le AND
- Le XOR
- Le Multiplexeur à 2 voies