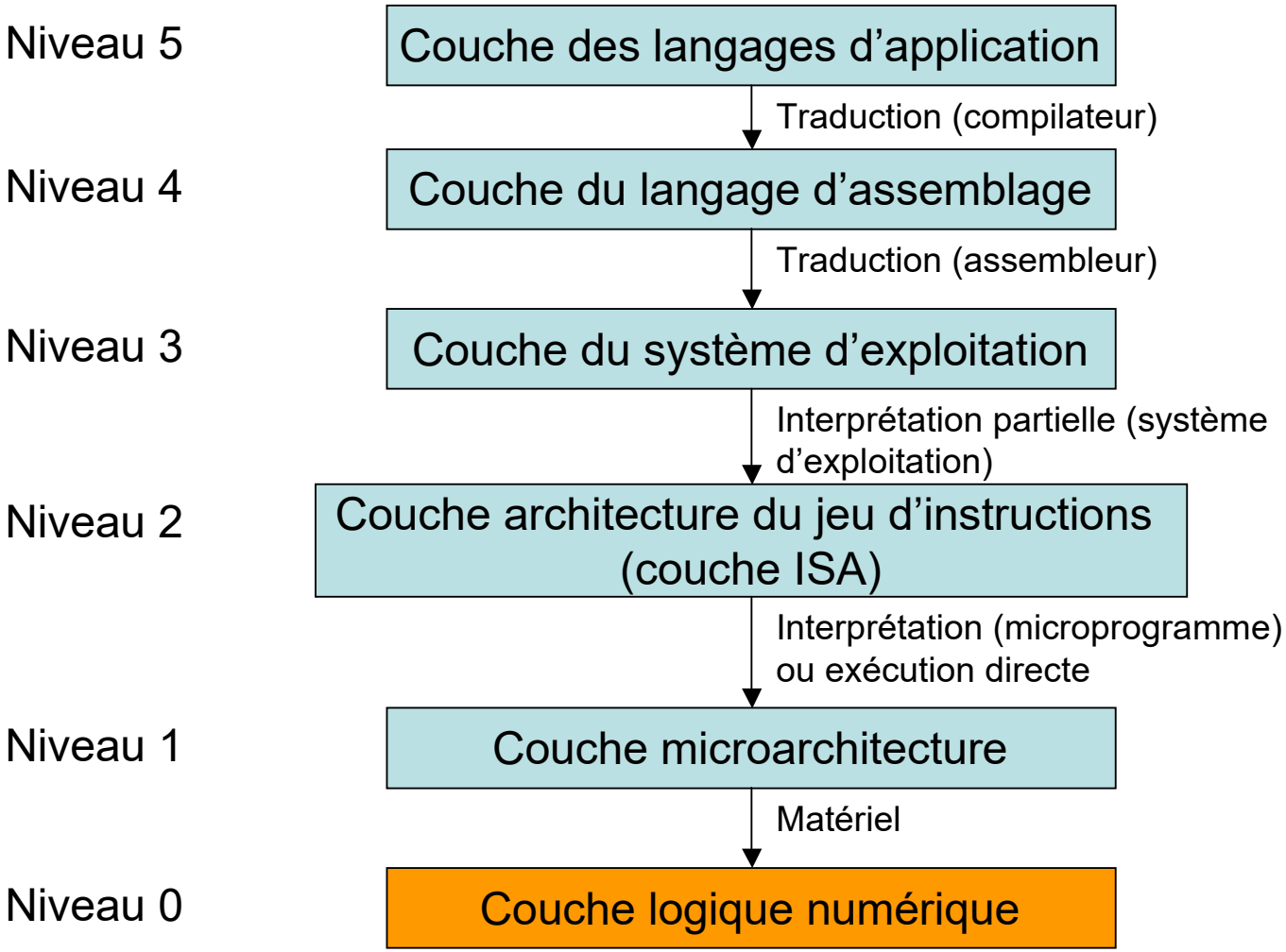
Introduction aux circuits logiques de base

Architecture en couches



Introduction

- Tout ordinateur est conçu à partir de circuits intégrés qui ont tous une fonction spécialisée (ALU, mémoire, circuit décodant les instructions etc.)
- Ces circuits sont fait à partir de circuits logiques dont le but est d'exécuter des opérations sur des variables logiques (binaires)

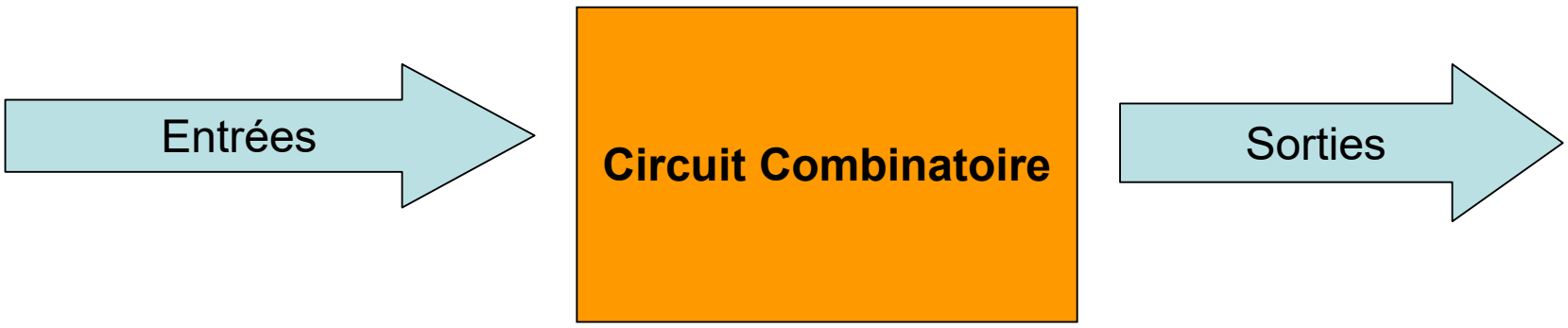
Introduction

- Les circuits logiques sont élaborés à partir de composants électroniques – transistors

- Types de circuits logiques:
  - Combinatoires
  - Séquentiels

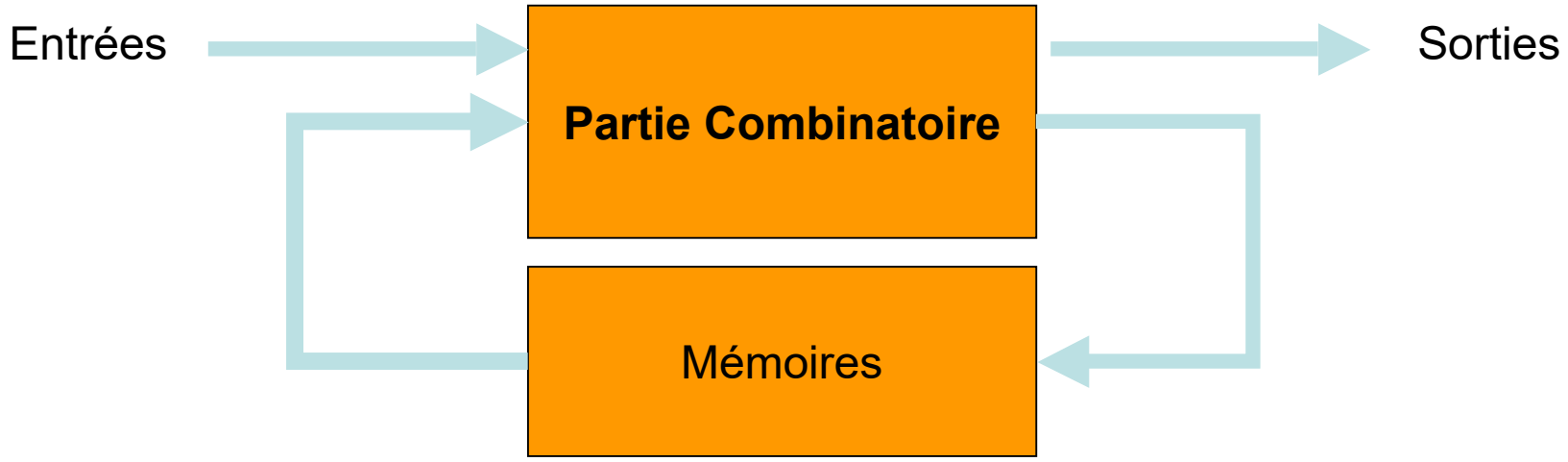
Circuits combinatoires

- Support théorique – algèbre de Boole
- Les fonctions de sortie s'expriment selon des expressions logiques des seules variables d'entrée
  - Un circuit combinatoire est défini par une ou plusieurs fonctions logiques



Circuits séquentiels ou à mémoire

- Support théorique – FSM (Finite State Machine)
- Les fonctions de sortie dépendent non seulement de l'état des variables d'entrée mais également de l'état antérieur de certaines variables de sortie (propriétés de mémorisation)



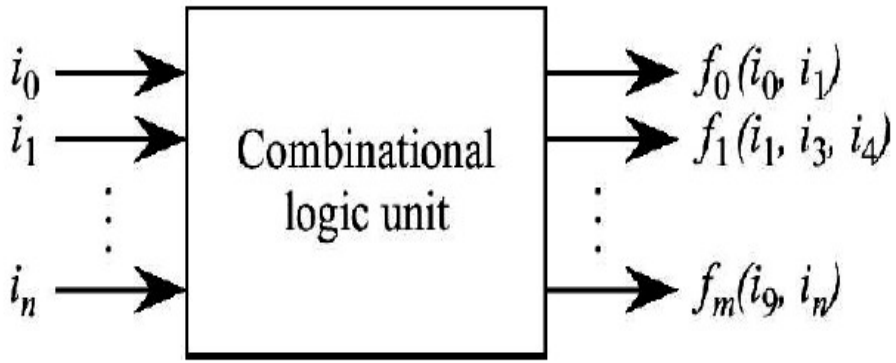
Variables booléennes

- **Un système binaire est un système qui ne peut exister que dans deux états autorisés.**
- **Diverses notations peuvent être utilisées pour représenter ces deux états :**
  - **numérique : 1 et 0**
  - **logique : vrai et faux**
  - **électronique : ON et OFF, haut et bas**
- Une variable logique est une variable qui peut prendre deux états ou valeurs: vrai (V) ou faux (F)
- En faisant correspondre V avec le chiffre binaire 1 et F – 0, ce type de variable devient une variable booléenne ou binaire

Circuits combinatoires

- Le circuit combinatoire est défini lorsque son nombre d'entrées, sont nombre de sorties ainsi que l'état de chaque sortie en fonction des entrées ont été précisés
- Ces informations sont fournies grâce à une table de vérité
- La table de vérité d'une fonction de n variables a 2^n lignes - états d'entrée
- Algèbre de Boole et les fonctions logiques sont le support théorique des circuits combinatoires

Table de vérité



i_0	i_1	$F_0(i_0, i_1)$
0	0	1
0	1	0
1	0	1
1	1	0

i_1	i_3	i_4	$F_1(i_1, i_3, i_4)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
.	.	.	0
1	1	1	1

i_0	i_1	i_2 . . .	i_n	$F_0(i_0, i_1)$	$F_1(i_1, i_3, i_4)$	. . .	$F_m(i_9, i_n)$
0	0	0 …	0				
0	0	0 …	1				
. . .					0		
.	.	.					
1	1	1…	1				

Portes logiques

- En électronique les deux états d'une variable booléenne sont associés à deux niveaux de tension: V(0) et V(1) pour les états 0 et 1 respectivement.

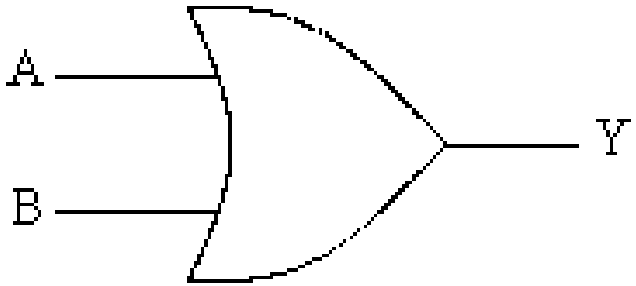
- Toute fonction logique peut être réalisée à l'aide d'un nombre de fonctions logiques de base appelées portes

- Un circuit se représente par un logigramme

Porte OU

- Au moins deux entrées
- La sortie d'une fonction OU est dans l'état 1 si au moins une de ses entrées est dans l'état 1

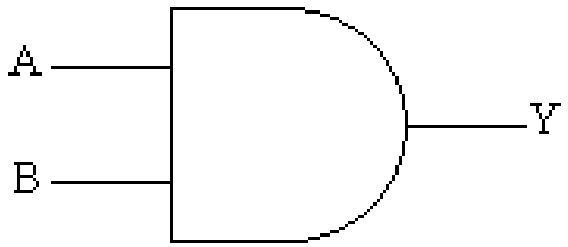
A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1



Porte ET

- Au moins deux entrées
- La sortie d'une fonction AND est dans l'état 1 si et seulement si toutes ses entrées sont dans l'état 1

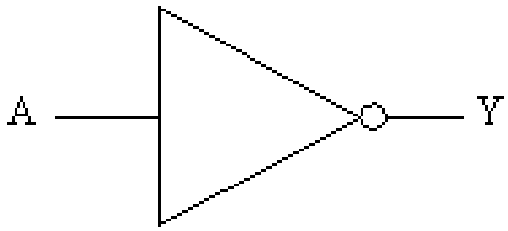
A	B	Y = A • B
0	0	0
0	1	0
1	0	0
1	1	1



Inverseur : porte NON

- Une seule entrée et une seule sortie
- La sortie d'une fonction NON prend l'état 1 si et seulement si son entrée est dans l'état 0

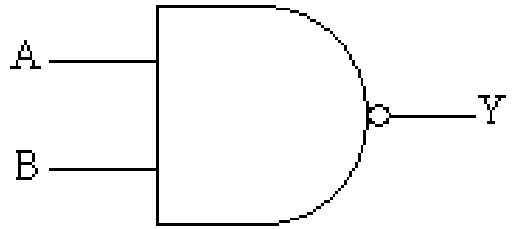
A	$Y = \overline{A}$
0	1
1	0



Porte NON ET

- Porte NON ET est constituée par un inverseur à la sortie d'une porte ET

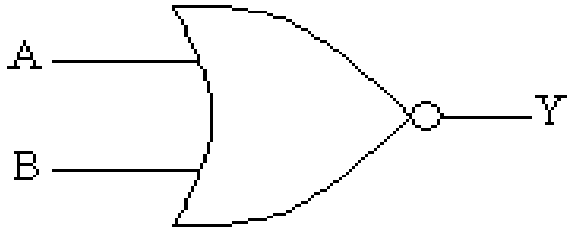
A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0



Portes NON OU

- Une négation à la sortie d'une porte OU constitue une fonction NON OU (NOR : NOT OR)

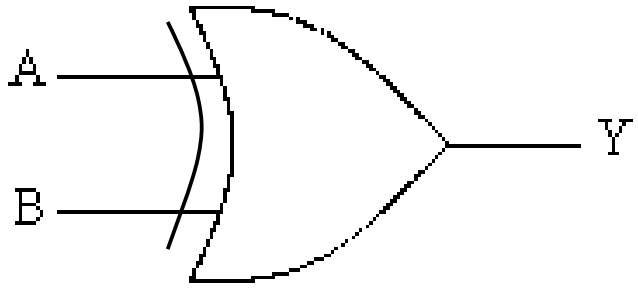
A	B	$Y = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0



Porte OU-EXCLUSIF (XOR)

- **Au moins deux entrées**
- **La sortie d'une fonction XOR est dans l'état 1 si le nombre de ses entrées à 1 est un nombre impair**

A	B	Y = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0



Réalisation des fonctions booléennes

- Toute fonction logique peut être réalisée à l'aide des portes
- Réalisation d'une fonction booléenne
  - Écrire l'équation de la fonction à partir de sa table de vérité
  - Simplifier l'équation
  - Réaliser l'équation à l'aide des portes disponibles

Comment rendre une table de vérité en une fonction booléenne

- À partir de la table de vérité, nous pouvons avoir deux formes analytiques, dénommées formes canoniques

  – **somme canonique de produits (SOP)**

  – **produit canonique de sommes (POS)**

Écritures canoniques (SOP)

- **3 variables,** terme produit, qu'on appelle minterme, égal au ET des variables qui composent cette combinaison

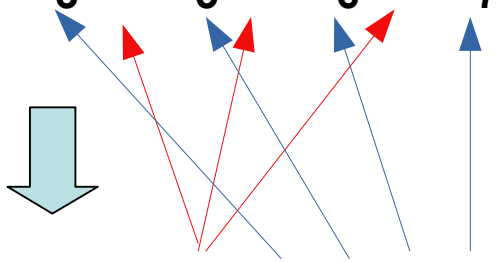
				P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
	x	y	z	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}y\bar{z}$	$\bar{x}yz$	$x\bar{y}\bar{z}$	$x\bar{y}z$	$xy\bar{z}$	xyz
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

Écritures canoniques, SOP

	A	B	C	F	$P_3 + P_5 + P_6 + P_7$
	0	0	0	0	0
	0	0	1	0	0
	0	1	0	0	0
3	0	1	1	1	1
	1	0	0	0	0
5	1	0	1	1	1
6	1	1	0	1	1
7	1	1	1	1	1

Cette façon, très générale, d'écrire une fonction booléenne est appelée somme canonique de produits (SOP)

F(A, B, C) = $P_3 + P_5 + P_6 + P_7$



$$F(A,B,C) = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC = \sum(3,5,6,7)$$

Écritures canoniques (POS)

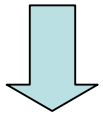
- **3 variables,** terme somme, qu'on appelle maxterme, égal au **OU** des variables qui composent cette combinaison

				S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
	X	Y	Z	$X+Y+Z$	$X+Y+\bar{Z}$	$X+\bar{Y}+Z$	$X+\bar{Y}+\bar{Z}$	$\bar{X}+Y+Z$	$\bar{X}+Y+\bar{Z}$	$\bar{X}+\bar{Y}+Z$	$\bar{X}+\bar{Y}+\bar{Z}$
0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1
2	0	1	0	1	1	0	1	1	1	1	1
3	0	1	1	1	1	1	0	1	1	1	1
4	1	0	0	1	1	1	1	0	1	1	1
5	1	0	1	1	1	1	1	1	0	1	1
6	1	1	0	1	1	1	1	1	1	0	1
7	1	1	1	1	1	1	1	1	1	1	0

Écritures canoniques, POS

X	Y	Z	F	$S_0 \cdot S_1 \cdot S_2 \cdot S_4$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

$$F(X, Y, Z) = S_0 \cdot S_1 \cdot S_2 \cdot S_4$$



$$F(X, Y, Z) = (X + Y + Z)(X + \overline{Y} + Z)(X + \overline{Y} + \overline{Z})(\overline{X} + Y + Z)$$

Cette écriture est appelée produit canonique de sommes (POS)

Écritures canoniques

- Écritures canoniques expriment une fonction booléenne à l'aide des opérateurs logiques ET, OU, NON

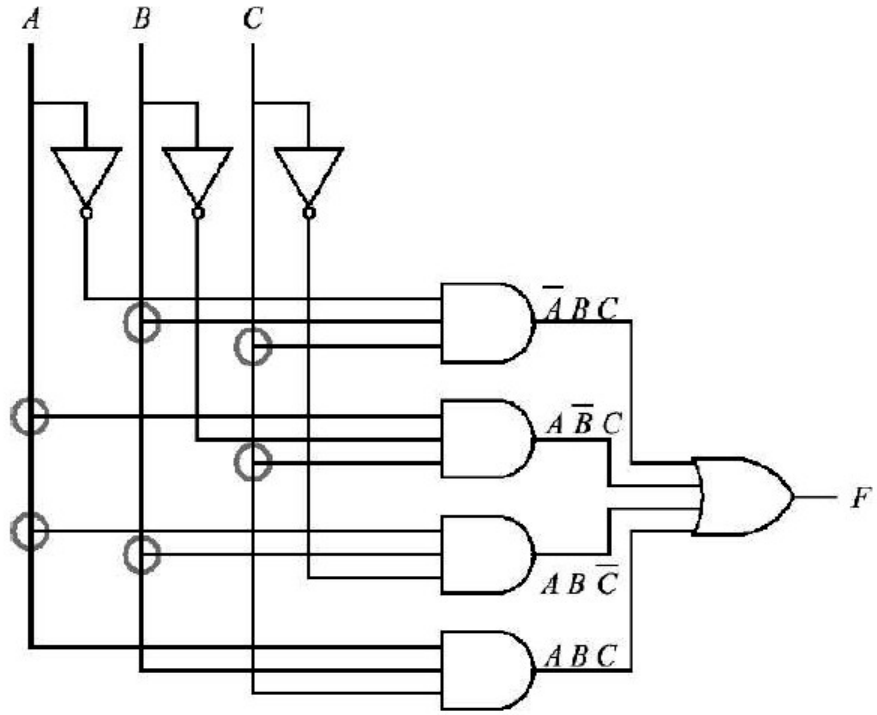
On peut réaliser une fonction à l'aide des portes

ET, OU, NON

Écritures canoniques d'une fonction logique

$\overline{A}BC$

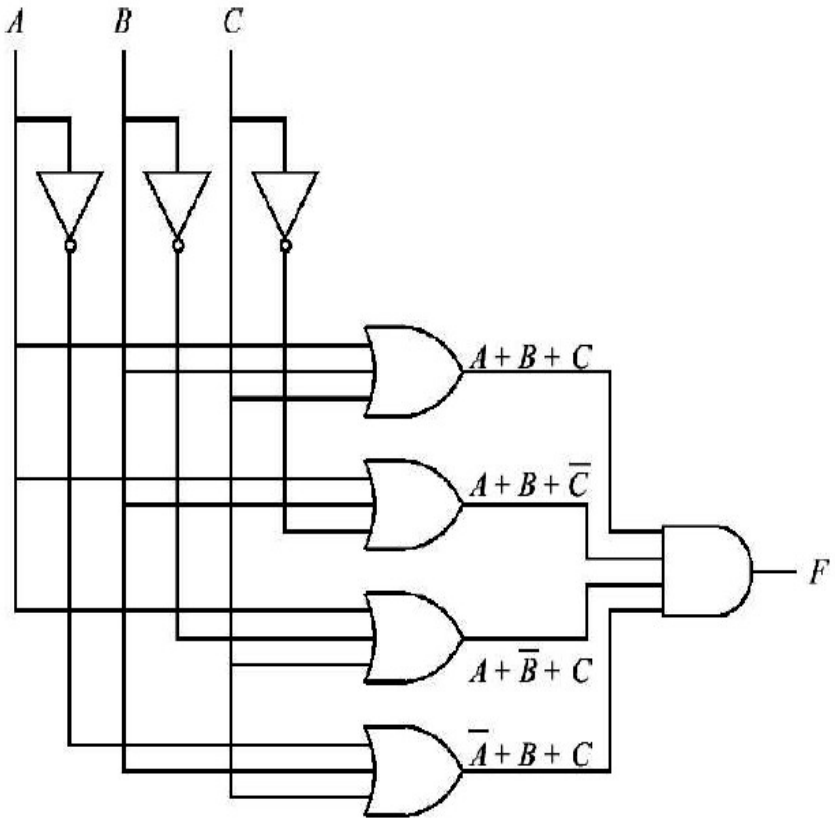
A	B	C	F	$P_3 + P_5 + P_6 + P_7$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



Écritures canoniques d'une fonction logique

A+B+C

A	B	C	F	$S_0 \cdot S_1 \cdot S_2 \cdot S_4$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



Relation d'équivalence des circuits

- Soucis majeurs des concepteurs
  - Réduire le nombre de portes nécessaires à la réalisation des systèmes
    - Minimiser le coût en nombre de boîtiers
    - La consommation électrique
  - Minimiser la complexité
    - Créer un système équivalent avec certains paramètres optimisés
  - Recherche d'équivalence
    - Utiliser les lois et théorèmes de l'algèbre de Boole

Résumé des identités booléennes de base

OU (Dualité)	**(A + B) + C = A + (B + C) = A + B + C** **A + B = B + A** **A + A = A** **A + 0 = A** **A + 1 = 1**
ET (Dualité)	**(A • B) • C = A • (B • C) = A • B • C** **A • B = B • A** **A • A = A** **A • 1 = A** **A • 0 = 0**
Distributivité	**A • (B + C) = (A • B) + (A • C)** **A + (B • C) = (A + B) • (A + C)**

Résumé des identités booléennes

NON	$\overline{\overline{A}} = A$ $\overline{A} + A = 1$ $\overline{A} \bullet A = 0$
Loi d'absorption	A + (A • B) = A A • (A + B) = A
De Morgan	$\overline{A \bullet B \bullet C \bullet \ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots$ $\overline{A + B + C + \ldots} = \overline{A} \bullet \overline{B} \bullet \overline{C} \bullet \ldots$
OU exclusif	$A \oplus B = (A + B) \bullet \overline{(A \bullet B)}$ $A \oplus B = (A \bullet B) + (B \bullet \overline{A})$ $A \oplus B = \overline{(A \bullet B) + (\overline{A} \bullet \overline{B})}$ $A \oplus B = (A + B) \bullet (\overline{A} + \overline{B})$ $A \oplus B = \overline{A} \bullet B + A \bullet \overline{B}$

Relation d'équivalence des circuits

- La manipulation algébrique

$$F(A,B,C) = \overline{A} \bullet \overline{B} \bullet C + \overline{A} \bullet B \bullet \overline{C} + A \bullet \overline{B} \bullet \overline{C} + A \bullet B \bullet C =$$

$$= C \bullet (\overline{A} \bullet \overline{B} + A \bullet B) + \overline{C} \bullet (A \bullet \overline{B} + \overline{A} \bullet B) =$$

$$= C \bullet \overline{(A \oplus B)} + \overline{C} \bullet (A \oplus B) = A \oplus B \oplus C$$

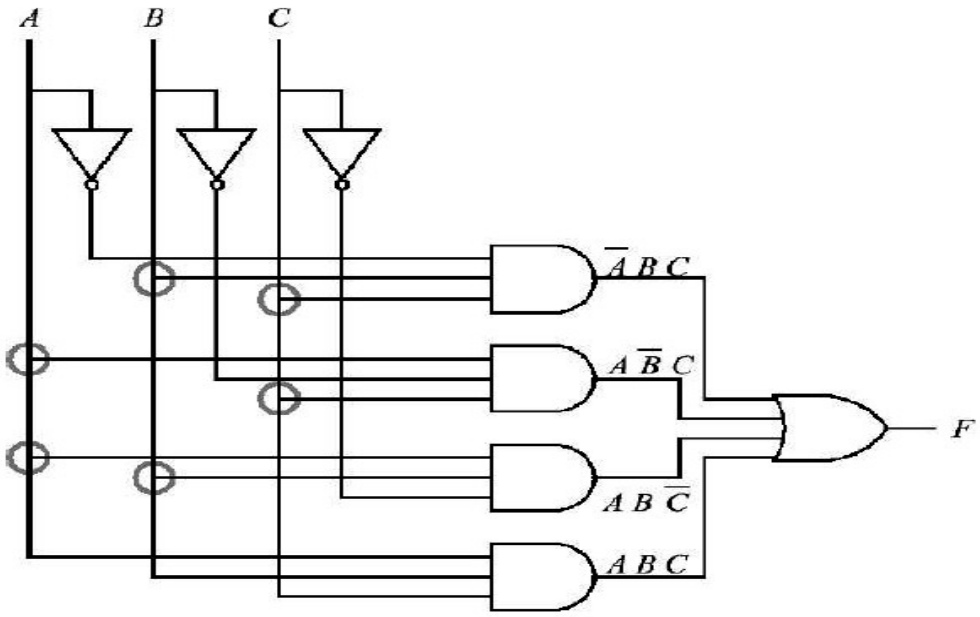
Relation d'équivalence des circuits

- Deux fonctions logiques sont équivalentes si, et seulement si, les valeurs de leurs sorties sont les mêmes pour toutes les configurations identiques de leurs variables d'entrée
  - Examen des tables de vérité respectives

Circuits combinatoires

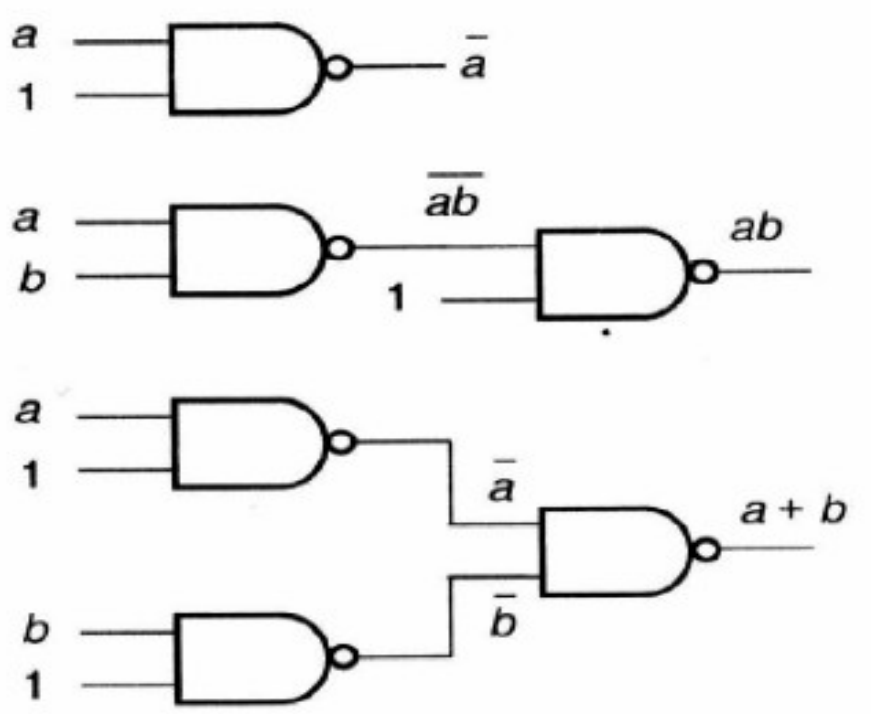
- **Toute fonction booléenne d'un nombre quelconque de variables peut s'écrire avec les trois fonctions de base ET, OU et NON**
- **L'ensemble { ET, OU, NON } est complet**

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



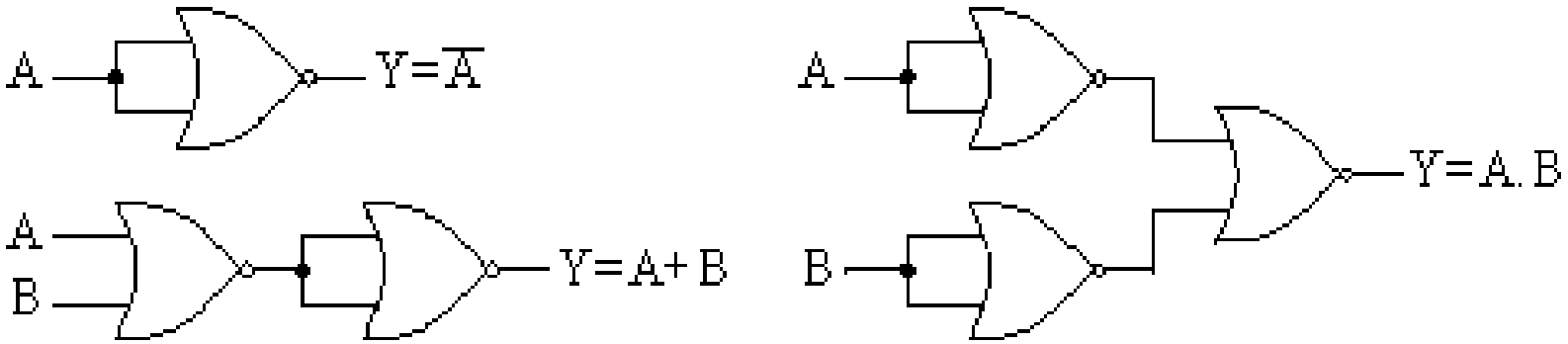
Ensemble { NON-ET (NAND) }

- { NON-ET (NAND) } est complet et minimal
- Les portes NON, OU et ET peuvent être obtenues à partir de portes NON-ET



Ensemble { NON-OU (NOR) }

- { NON-OU (NOR) } est complet et minimal
- Les portes NON, OU et ET peuvent être obtenues à partir de portes NON-OU

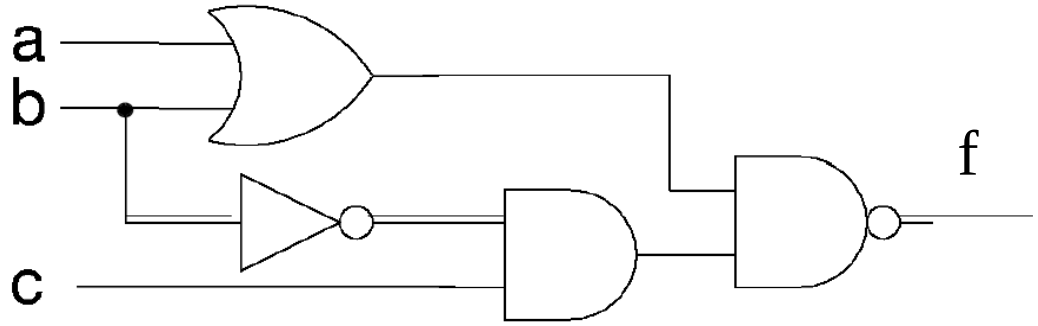


Analyse de circuit logique

- Trouver sa fonction logique
- Principe
  - Donner l'expression des sorties de chaque porte/composant en fonction des valeurs de ses entrées
  - En déduire au final la (ou les) fonction(s) logique(s) du circuit
  - On peut ensuite
    - Déterminer la table de vérité du circuit
    - Simplifier la fonction logique

Analyse de circuit logique

- 3 entrées, 1 sortie
- Composé uniquement de portes logiques OU, ET et NON

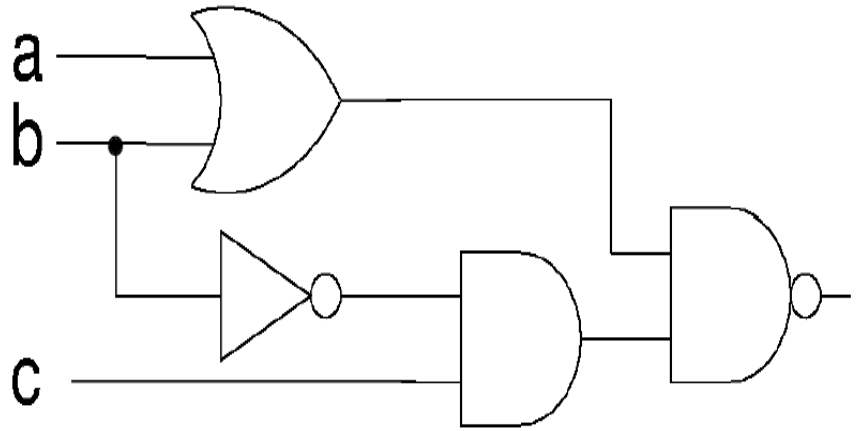


- À partir de son logigramme

$$f(a,b,c) = \overline{(a+b) \bullet (\bar{b} \bullet c)}$$

Analyse de circuit logique

$$f(a,b,c) = \overline{(a+b) \bullet (\bar{b} \bullet c)}$$



a	b	c	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Synthèse d'un circuit logique

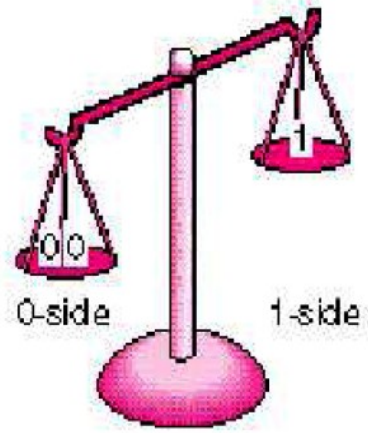
- A partir d'une fonction logique trouver le logigramme correspondant à cette fonction
- Principe
  - Simplifier la fonction logique avec 2 méthodes
    - La méthode algébrique (algèbre de Boole)
    - La méthode des tableaux de Karnaugh
  - En déduire le logigramme correspondant

Simplification d'expression booléenne

- La méthode algébrique (algèbre de Boole)
- La méthode des tableaux de Karnaugh
  - SOP, POS
- **Fonction Majorité**

$F(A, B, C) = \bar{A}BC + A\bar{B}C + AB\bar{C} +$
$ABC = \sum(3, 5, 6, 7)$

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



A balance tips to the left or right depending on whether there are more 0's or 1's.

La méthode des tableaux de Karnaugh

- Méthode graphiques de simplification

- Le diagramme de Karnaugh d'une fonction logique est une transformation graphique de la table de vérité qui permet la visualisation de tous les mintermes

Diagramme de Karnaugh

- Minterme est représenté par une case dans le diagramme de Karnaugh
  - Les cases sont placées d'une façon telle que les mintermes qui ne diffèrent que par l'état d'une seule variable ont une frontière commune sur une ligne ou sur une colonne, ou bien se trouvent aux extrémités d'une ligne ou d'une colonne

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$F(A, B, C) = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC = \sum(3, 5, 6, 7)$

C \ AB	00	01	11	10
0			1	
1		1	1	1

Méthode de Karnaugh

1. Transposition du tableau de vérité dans un tableau de Karnaugh ;
2. Réalisation des groupements de 1, 2, 4, 8 termes (une puissance de 2);
3. Minimisation des groupements (maximisation des termes dans un groupement) ;
   - si groupement d'un terme, alors on ne fait rien ;
   - on élimine les variables qui changent d'état et on conserve le produit des variables qui n'ont pas changé d'état dans le groupement;
4. L'expression logique finale est la réunion des groupements après élimination des variables

Méthode de Karnaugh

- Réalisation des groupements de 1, 2, 4, 8 termes (une puissance de 2)
- Minimisation des groupements
  - maximisation des termes dans un groupement

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

C \ AB	00	01	11	10
0			1	
1		1	1	1

Méthode de Karnaugh

- On élimine les variables qui changent d'état et on conserve le produit des variables qui n'ont pas changé d'état dans le groupement

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

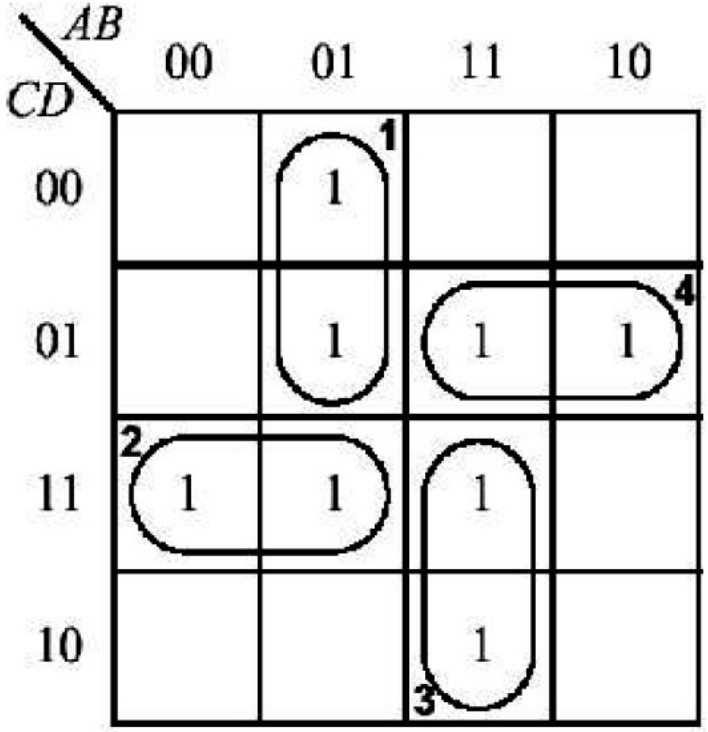
C \ AB	00	01	11	10
0			1	
1		1	1	1

B

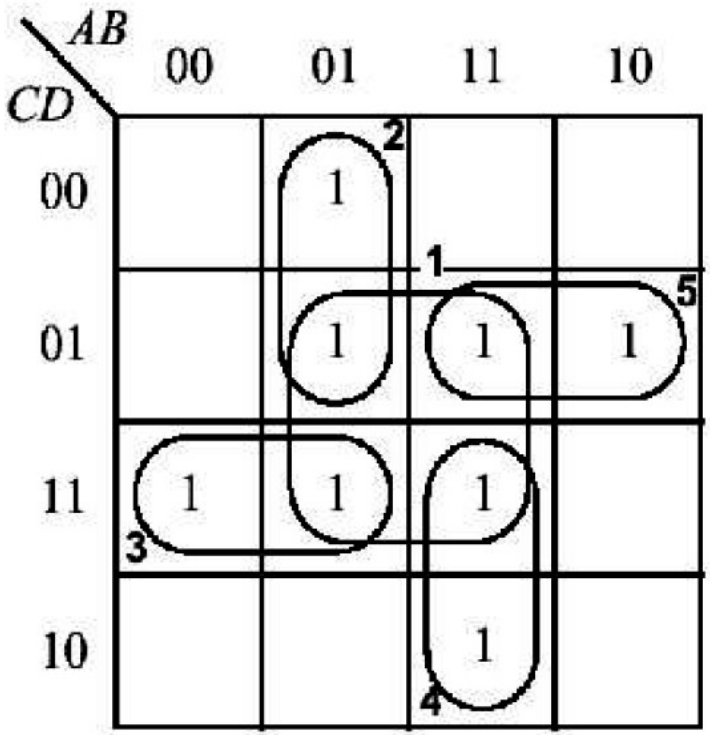
A

F = AB + BC + AC

Groupement minimal et non minimal



$$F = \overline{A}\,B\,\overline{C} + \overline{A}\,C\,D + A\,B\,C + A\,\overline{C}\,D$$



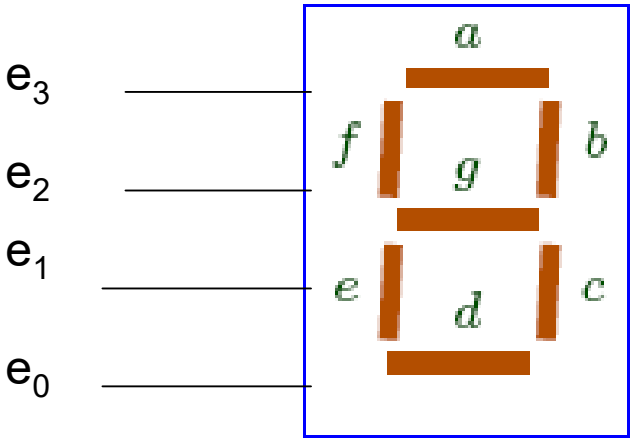
$$F = B\,D + \overline{A}\,B\,\overline{C} + \overline{A}\,C\,D + A\,B\,C + A\,\overline{C}\,D$$

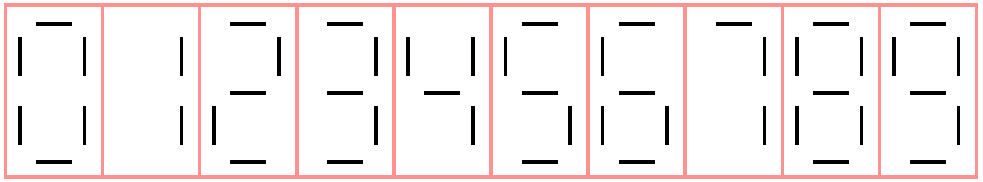
Fonctions booléennes incomplètement spécifiées

- Il existe des fonctions booléennes pour lesquelles il n'y a pas de valeurs associées à certains termes produits
- Ceux-ci ne sont jamais "sélectionnés" et la valeur qui leur est associée peut être indifféremment 0 ou 1. On note « d » *(don't care)*
- L'afficheur 7 segments est un exemple particulier de fonction booléenne incomplètement spécifiée.

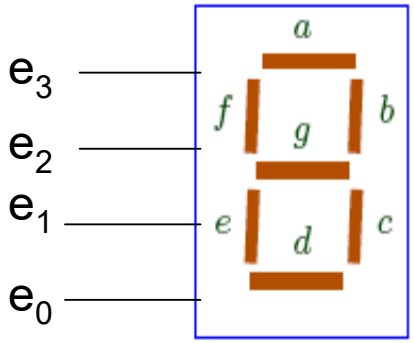
L'afficheur 7 segments

- On veut afficher les 10 chiffres décimaux à l'aide de 7 segments, notés de a à g, qui peuvent être à 0 (éteint) ou 1 (allumé). Le codage des 10 chiffres décimaux nécessite 4 bits, que l'on peut noter e_3 à e_0.





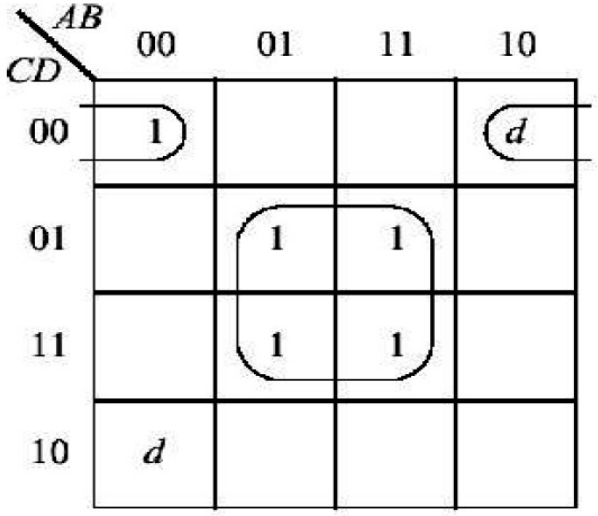
L'afficheur 7 segments



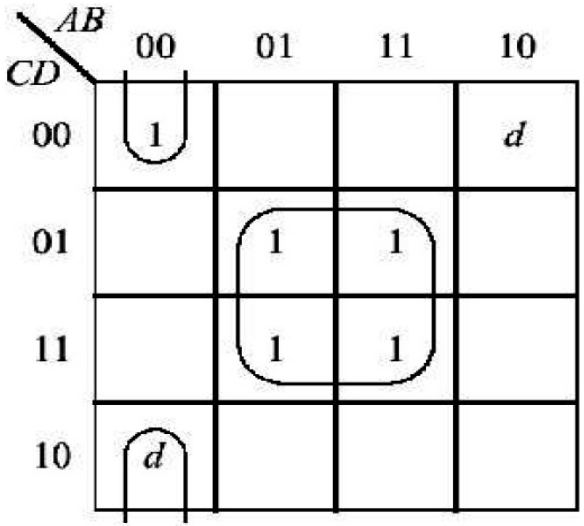
e3	e2	e1	e0		a	b	c	d	e	f	g
0	0	0	0	**0**	1	1	1	1	1	1	0
0	0	0	1	**1**	0	1	1	0	0	0	0
0	0	1	0	**2**	1	1	0	1	1	0	1
0	0	1	1	**3**	1	1	1	1	0	0	1
0	1	0	0	**4**	0	1	1	0	0	1	1
0	1	0	1	**5**	1	0	1	1	0	1	1
0	1	1	0	**6**	1	0	1	1	1	1	1
0	1	1	1	**7**	1	1	1	0	0	0	0
1	0	0	0	**8**	1	1	1	1	1	1	1
1	0	0	1	**9**	1	1	1	0	0	1	1
1	0	1	0	**10**	d	d	d	d	d	d	d
1	0	1	1	**11**	d	d	d	d	d	d	d
1	1	0	0	**12**	d	d	d	d	d	d	d
1	1	0	1	**13**	d	d	d	d	d	d	d
1	1	1	0	**14**	d	d	d	d	d	d	d
1	1	1	1	**15**	d	d	d	d	d	d	d

Tableau de Karnaugh et D(on’t care)

- Lorsqu’une variable peut être indifféremment un « 1 » ou un « 0 » symbolisé par un d (« don’t care »), il peut y avoir plus d’un groupement minimal



$F = \overline{B}\,\overline{C}\,\overline{D} + B\,D$



$F = \overline{A}\,\overline{B}\,\overline{D} + B\,D$

Synthèse d'un circuit logique

1. Identifier les entrées et les sorties (IN / OUT) du circuit.
2. Construire la table (les tables) de vérité.
3. Identifier chaque fonction à partir de la table de vérité.
4. Simplifier chaque fonction.
5. Dessiner le schéma du circuit.

Exemple

- Établissez la table de vérité d'un additionneur combinatoire de deux bits. Le circuit doit avoir quatre entrées, dont les deux bits du premier opérande et les deux bits du deuxième opérande. Il doit avoir trois sorties, deux bits qui expriment la somme et le bit de la retenue. Réalisez le circuit à l'aide des portes OU, ET et NON.

Conception d'un circuit logique

- Additionneur combinatoire de deux bits

  1. Identifier les entrées et les sorties (IN / OUT) du circuit

  – 4 entrées (2 bits du premier opérande et 2 bits du deuxième opérande)
    - A, B, C, D

  – 3 sorties (2 bits - la somme et 1 bit - la retenue)
    - S1, S2, R

Table de vérité

Additionneur combinatoire de 2 bits

Construire la table (les tables) de vérité.

A	B	C	D	S_1	S_2	R
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	1	0	1	1
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	1	0	0	1	1
1	1	1	1	1	0	1

Tableaux de Karnaugh

Simplifier chaque fonction: S1=∑(2,3,5,6,8,9,12,15)

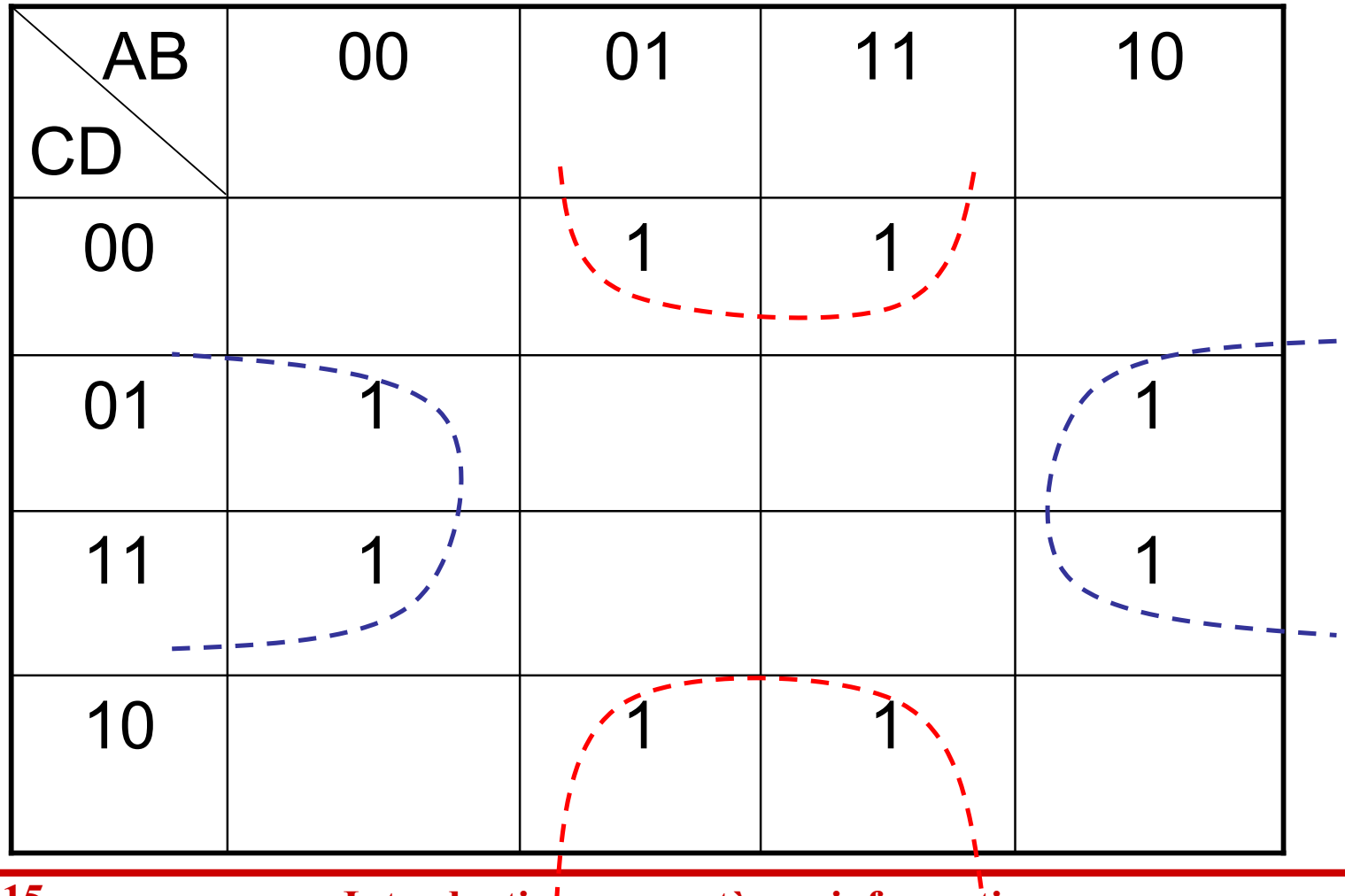
$$S_1 = \bar{A}B\bar{C}D + ABCD + \bar{A}C\bar{D} + A\bar{C}\bar{D} + A\bar{B}\bar{C} + \bar{A}\bar{B}C$$

AB / CD	00	01	11	10
00			1	1
01		1		1
11	1		1	
10	1	1		

Tableaux de Karnaugh

Simplifier chaque fonction: S2=∑(1,3,4,6,9,11,12,14)

$S_2 = \bar{B}D + B\bar{D}$



Tableaux de Karnaugh

Simplifier chaque fonction: R=∑(7,10,11,13,14,15)

R = AC + BCD + ABD

CD \ AB	00	01	11	10
00				
01			1	
11		1	1	1
10			1	1

Conception d'un circuit logique

- Additionneur combinatoire de deux bits

  5. Dessiner le schéma du circuit

$S_1 = \bar{A}B\bar{C}D + ABCD + \bar{A}C\bar{D} + A\bar{C}\bar{D} + A\bar{B}\bar{C} + \bar{A}\bar{B}C$

$S_2 = \bar{B}D + B\bar{D}$

$R = AC + BCD + ABD$

Circuits intégrés logiques

- Un circuit contenant les portes intégrées par divers procédés technologiques sur une petite plaquette de silicium
- Cette puce de silicium est enfermée dans un boîtier
- Sur les côtés sont disposées des broches (ou pattes)
  - Assurer les connexions électriques des circuits logiques internes

Circuits intégrés logiques

- 4 familles des circuits intégrés logiques (selon la densité d'intégration)
  - SSI (Small Scale Integration)
    - Circuits à faible intégration groupant de 1 à 10 portes/circuit
  - MSI (Medium Scale Integration)
    - Circuits à moyenne intégration groupant de 10 à 100 portes/circuit

Circuits intégrés logiques

- LSI (Large Scale Integration)
  - Circuits à haute intégration groupant de 100 à 100 000 portes/circuit
- VLSI (Very Large Scale Integration)
  - Circuits à très haute intégration groupant plus de 100 000 portes/circuit
  - Mémoires, microprocesseurs
    - Intel Pentium > 3 000 000 transistors

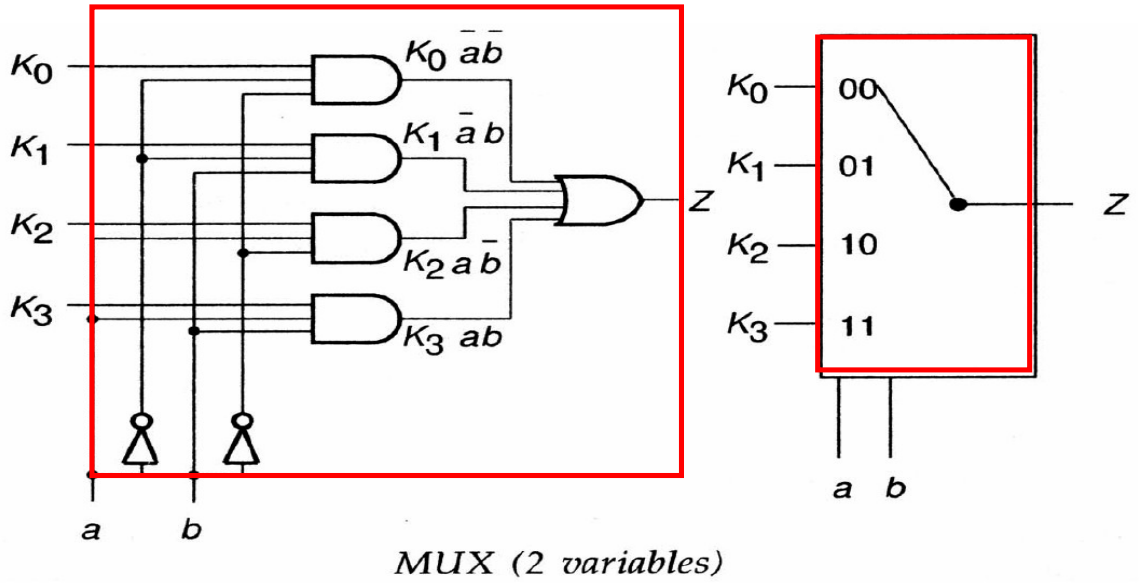
Circuits intégrés logiques

- Principaux circuits MSI combinatoires
  - Multiplexeur
  - Démultiplexeur
  - Encodeur
  - Décodeur
- Les circuits de traitement ou de calcul
  - Un circuit de décalage
  - Additionneur
  - Unité arithmétique et logique

Multiplexeur

- Le multiplexeur est un circuit combinatoire Sélecteur qui possède 2^n entrées d'information, n entrées de commande et une seule sortie. Son rôle consiste à sélectionner, à l'aide de signaux de commande, une des entrées et à la lier à la sortie.

a	b	Z
0	0	K_0
0	1	K_1
1	0	K_2
1	1	K_3

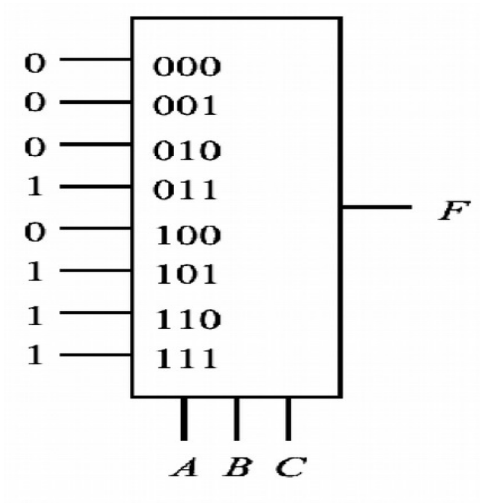


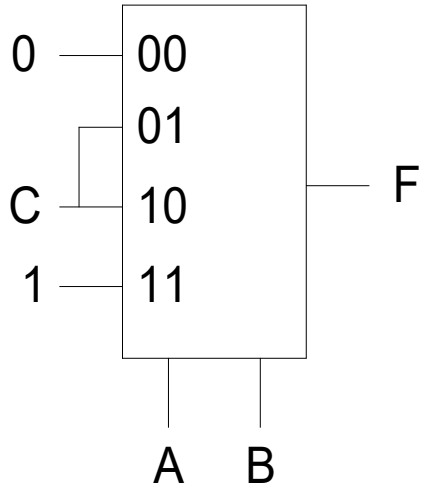
MUX (2 variables)

Multiplexeur

- Le multiplexeur peut générer une fonction booléenne si on utilise ses entrées de contrôle pour sélectionner (une à la fois) les 8 données d'entrée
  - Fonction Majorité

A	B	C	F	
0	0	0	0	0
0	0	1	0	
0	1	0	0	c
0	1	1	1	
1	0	0	0	
1	0	1	1	
1	1	0	1	1
1	1	1	1	



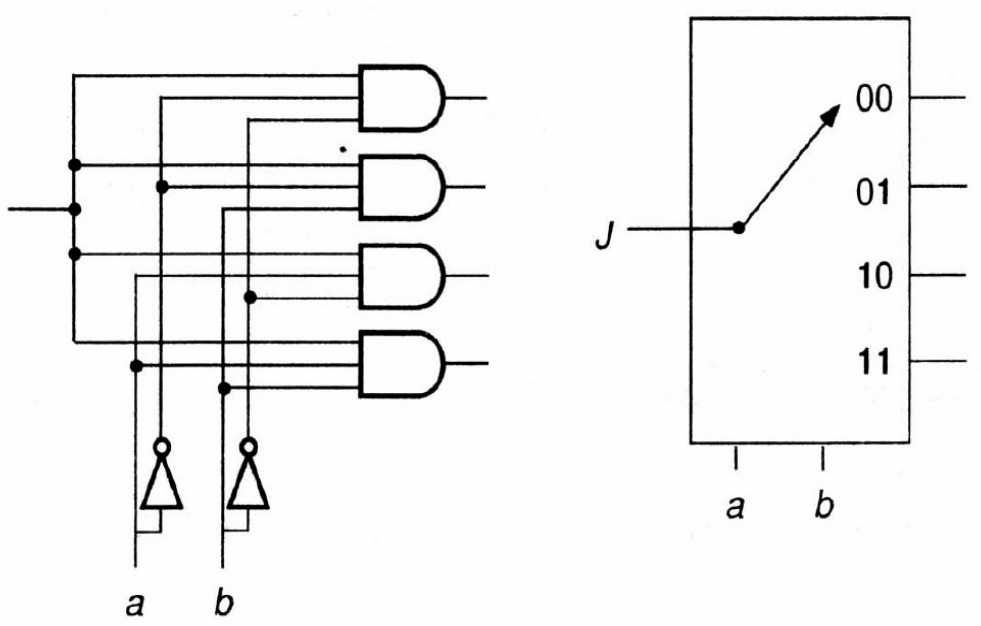


Démultiplexeur

- 1 entrée, X sorties
- Selon une adresse (n bits), une des X sorties prend la valeur de l'entrée

Valeurs des k_x sorties selon a et b

a	b	k0	k1	k2	k3
0	0	K	0	0	0
0	1	0	K	0	0
1	0	0	0	K	0
1	1	0	0	0	K



DEMUX (2 variables)

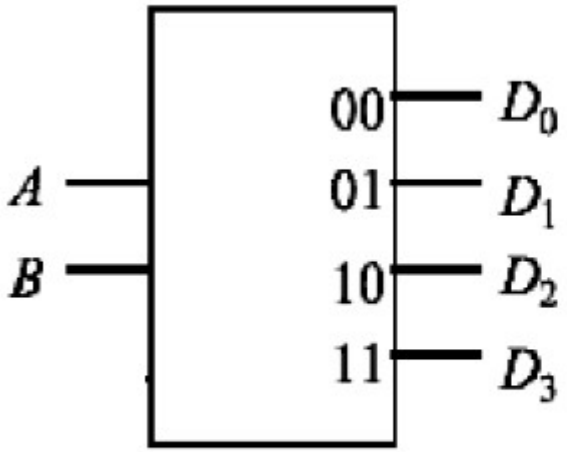
Encodeur

- Active un code selon l'une des X entrées actives
- 2^n (en général) entrées, 1 entrée active (valeur 1), les autres sont toutes désactivées (valeur 0)
- Code en sortie : sur *n* bits
- Encodeur sur 3 bits

E0	E1	E2	E3	E4	E5	E6	E7	S0	S1	S2
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Décodeur

- Active une des X sorties selon un code
- Code : sur *n* bits
- Nombre de sorties : 2^n (en général)



A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

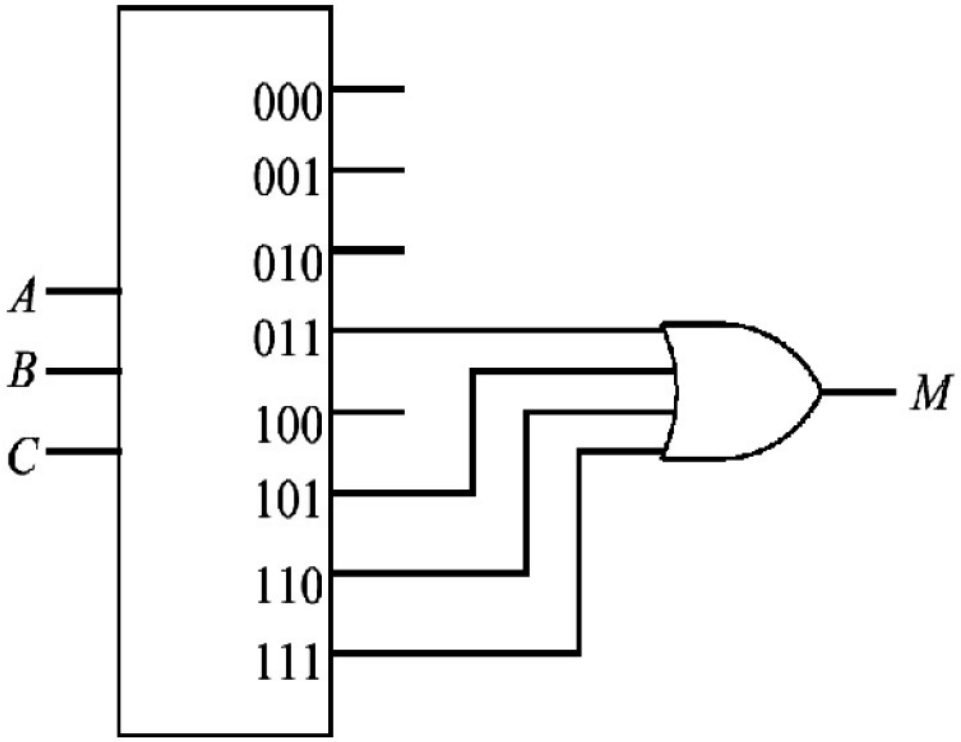
$$D_0 = \overline{A}\,\overline{B} \qquad D_1 = \overline{A}\,B \qquad D_2 = A\,\overline{B} \qquad D_3 = A\,B$$

Décodeur

- Fonction Majorité

$$M = CB\overline{A} + C\overline{B}A + \overline{C}BA + CBA$$

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

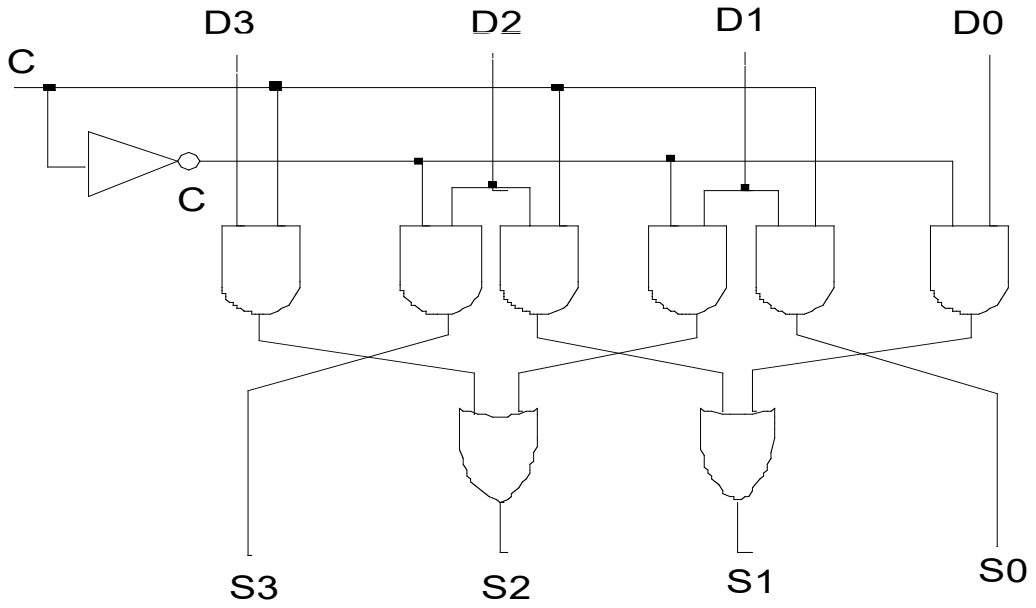


Les circuits de traitement ou de calcul

- Circuits MSI utilisés couramment dans les unités de calcul
  - Décaleur
  - Additionneur
  - Unité arithmétique et logique

Décaleur

- Un décaleur est formé de (n+1) entrées D_1, ..., D_n, C et de n sorties S_1, ..., S_n et opère un décalage de 1 bit sur les entrées D_1, ..., D_n
- Si C=1, il s'agit d'un décalage à droite et si C=0, d'un décalage à gauche



Additionneur

- Pour réaliser des additions binaire de 2 nombres A et B de n bits on décompose un circuit en deux parties
  - Un circuit correspondant à l'addition des bits de poids faible a_0 et b_0 (il n'y a pas de retenue propagée à prendre en compte)
  - Un circuit correspondant à l'addition des bits de poids supérieur a_i et b_i (prendre en compte la retenue r_{i-1} propagée depuis le rang i-1 inférieur)

Additionneur

- Additionneur 1 bit pour les bits de poids faible a_0 et b_0

a_0	b_0	Som	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

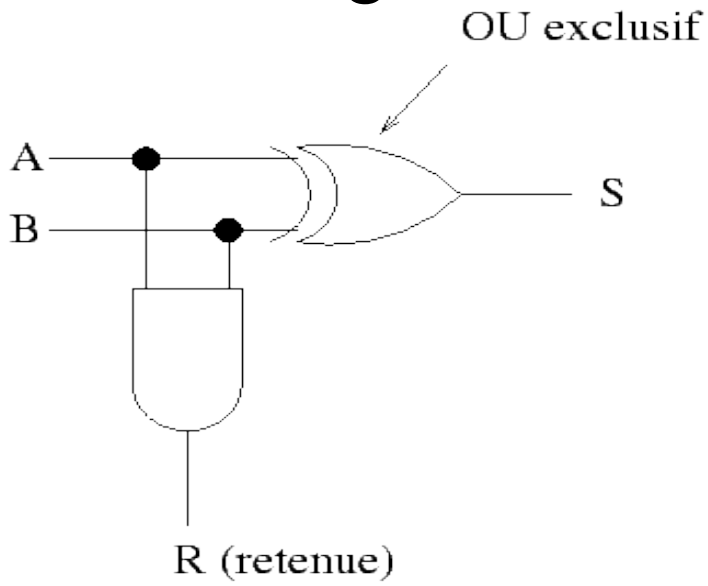
$Som = a_0 \oplus b_0$ $R = a_0 b_0$

Additionneur

- Le circuit logique associé est donc constitué de deux portes, un OU exclusif pour le résultat et un ET pour la retenue:
- Comme ce circuit ne tient pas compte d'une retenue propagée depuis le rang inférieur

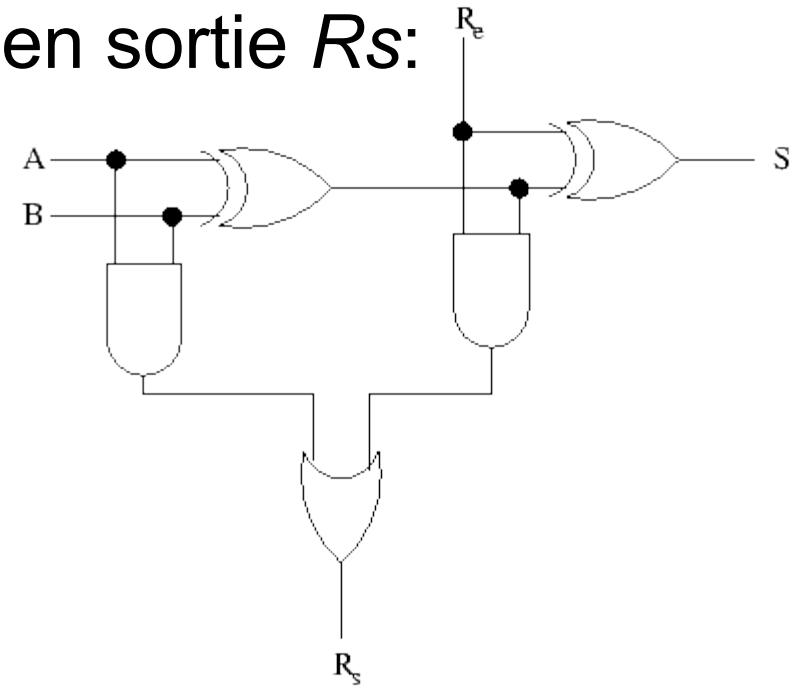
– il est qualifié de demi-additionneur

$S = A \oplus B \quad R = AB$



Additionneur

- Additionneur 1 bit pour les bits de poids fort
  - Afin de permettre une liaison de plusieurs additionneurs en série, un additionneur doit avoir une retenue en entrée *Re* en plus de la retenue en sortie *Rs*:

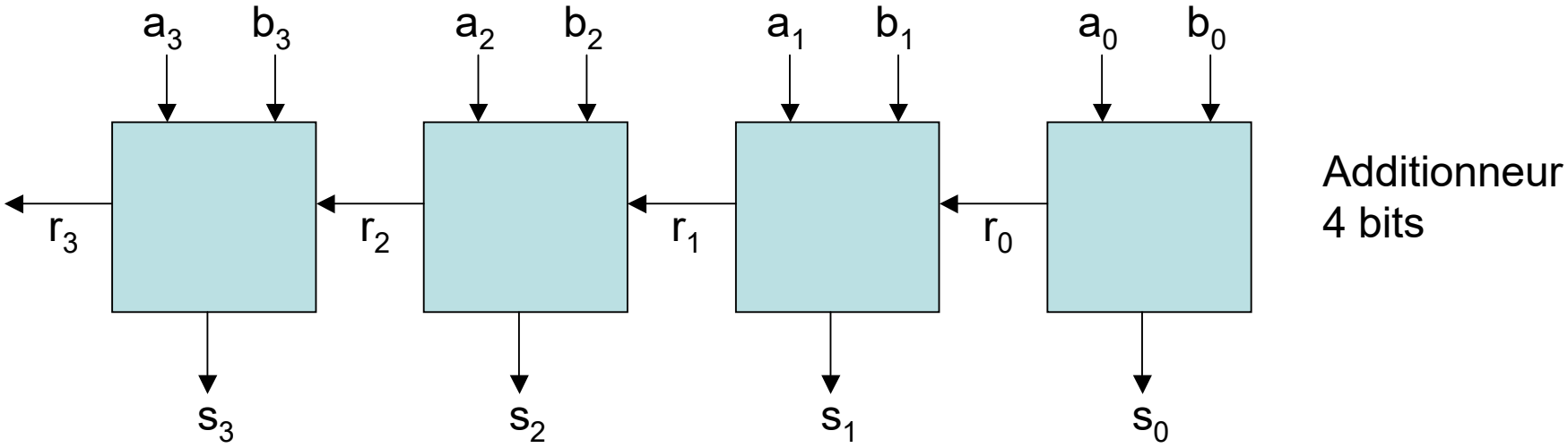


Additionneur complet

A	B	R_e	S	R_s
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Additionneur complet n bits

- L'additionneur n bits est obtenu en chaînant entre eux un demi-additionneur et n-1 additionneurs 1 bit complets
- Le chaînage s'effectue par le biais des retenues propagées

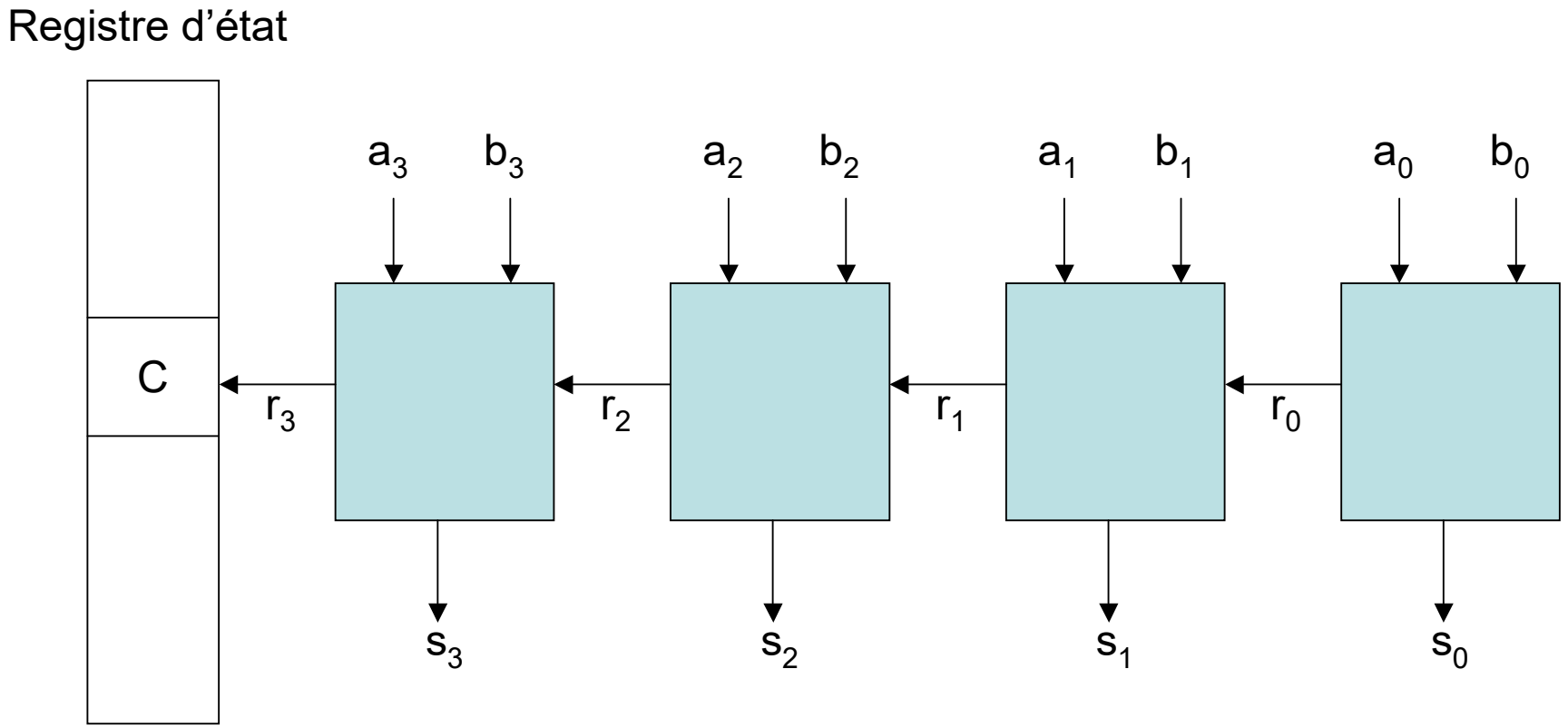


Additionneur

- Indicateur de carry
  - Lors d'une opération arithmétique effectuée sur des nombres de n bits un n+1e bit, un bit de carry peut être généré
  - Ce bit de carry mémorisé par l'indicateur C du registre d'état du processeur, PSW, correspond au niveau de l'additionneur n bits, à une retenue r à la position n-1 égale à 1

Additionneur

- Indicateur de carry pour l'additionneur 4 bits



Additionneur

- Indicateur d'overflow
  - Lors d'une opération arithmétique mettant en jeu des nombres de n bits et de même signe, le résultat peut être en dehors de l'intervalle des nombres représentables sur n bits par la convention choisie pour la représentation de ces nombres signés

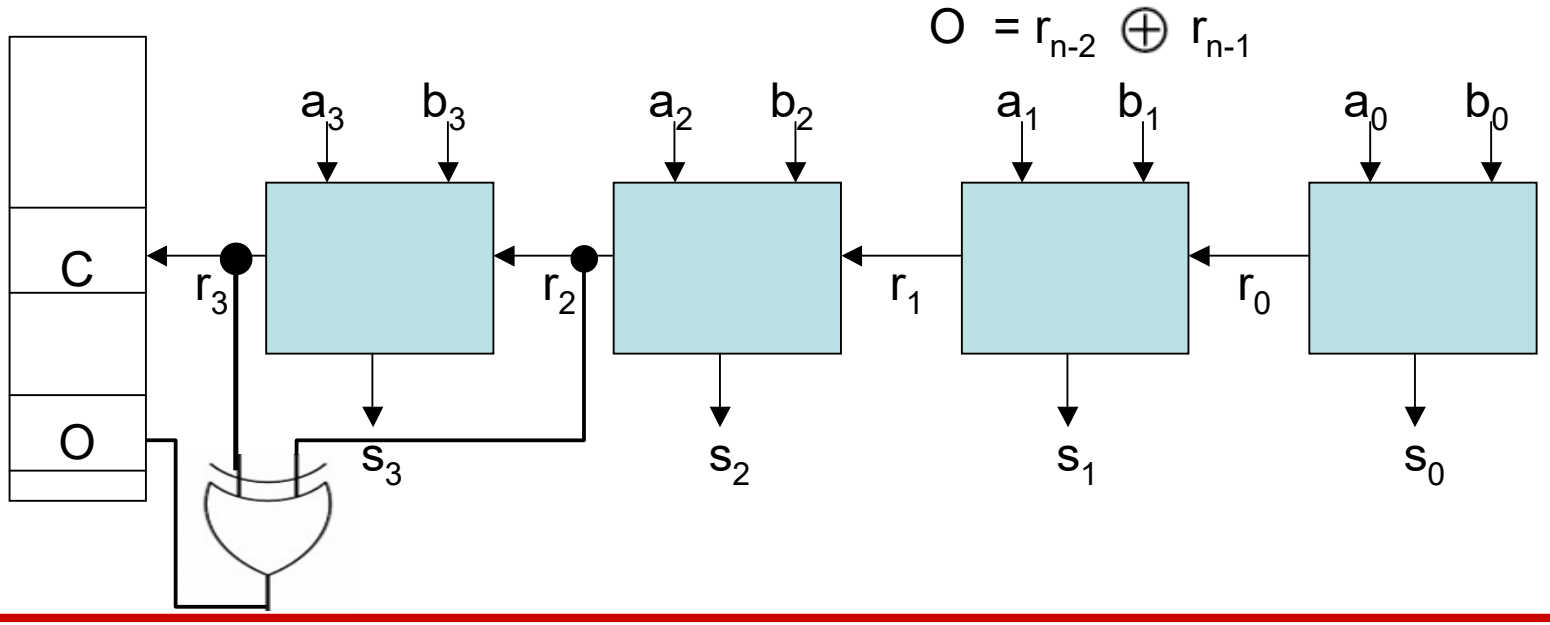
Dépassement de capacité ne peut se produire que lors de l'addition de 2 nombres de même signe

Indicateur d'overflow

- – Overflow peut être détécté en effectuant un test de comparaison entre r_{n-2} et r_{n-1}

  – Overflow <=> $r_{n-2} \neq r_{n-1}$

Registre d'état

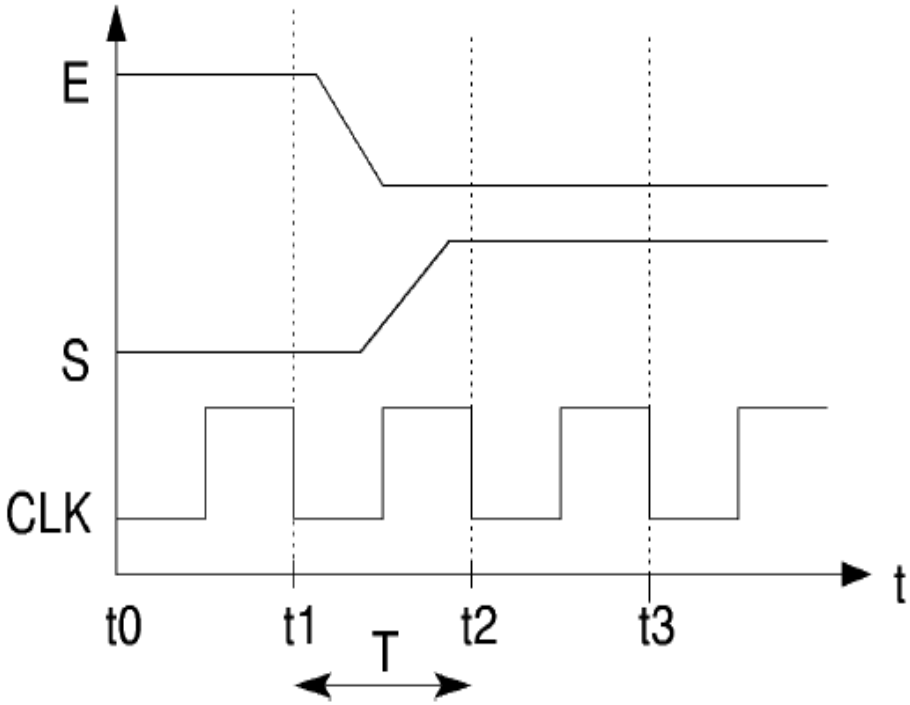


Horloge

- À cause de tous les délais existants (montée, descente, propagation) un signal n'est pas dans un état valide en permanence
- Idée : on ne lit ses valeurs qu‘à des instants précis et à des intervalles réguliers
  - Instants donnés par une horloge
- Horloge
  - Système logique qui émet régulièrement une suite d’impulsions calibrées
  - L’intervalle de temps entre 2 impulsions représente le temps de cycle ou la période de l’horloge

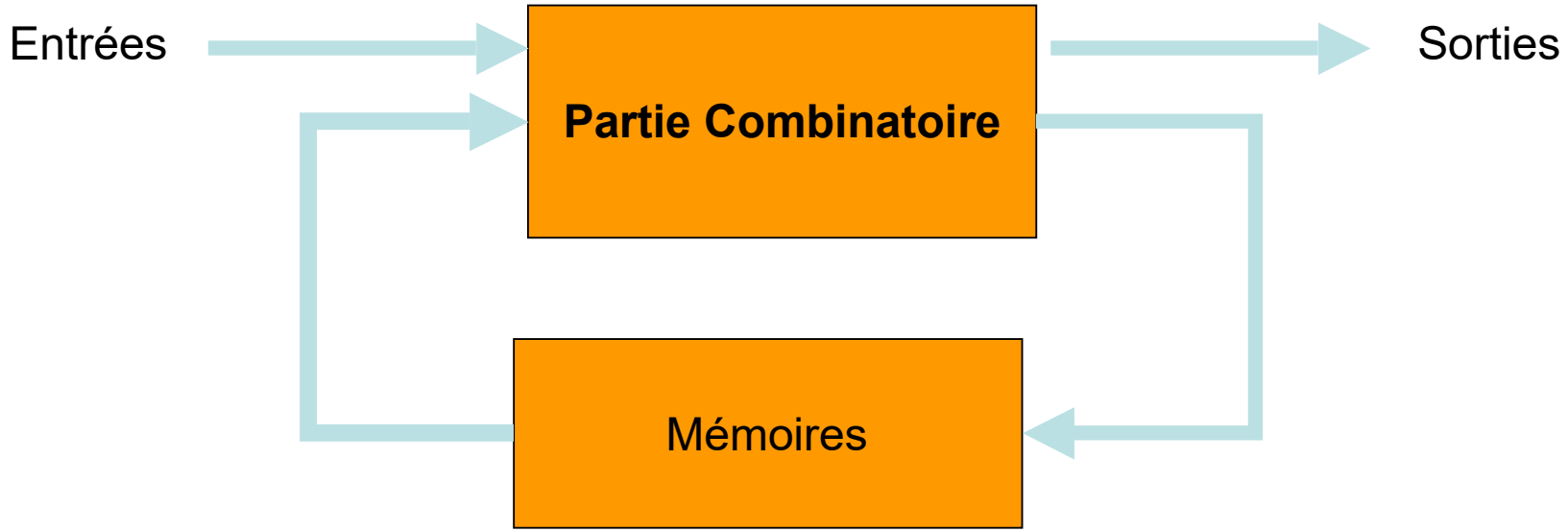
Horloge

- Signal périodique
  - une demi période à 0, l'autre à 1
- Début d'une nouvelle période : instant t_i
- Exemple
  - Instant *t1* : E = 1, S = 0
  - Instant *t2* : E = 0, S = 1
  - CLK = Clock = signal d'horloge



Circuits logiques à mémoire

- Circuits séquentiels ou à mémoire (FSM)
  - Les fonctions de sortie dépendent non seulement de l'état des variables d'entrée mais également de l'état antérieur de certaines variables de sortie (propriétés de mémorisation)



Circuits séquentiels

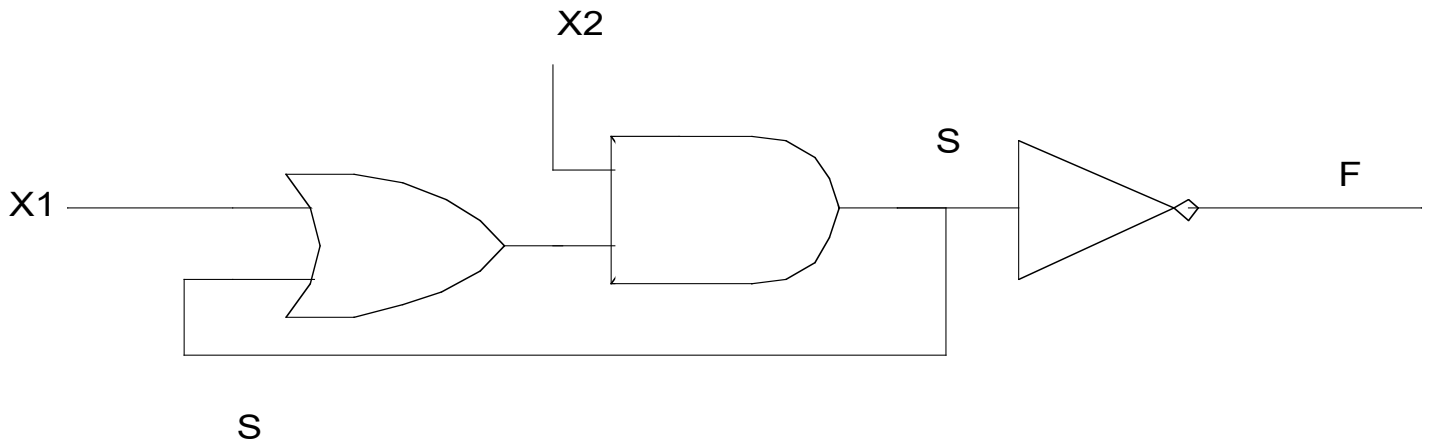
- Circuits combinatoires
  - Les sorties ne dépendent que des valeurs des entrées
- Circuits séquentiels
  - Ajout des notions d'état et de mémoire
  - Ajout de la notion de temps (horloge)

Circuits séquentiels

- Les valeurs de sorties du circuit dépendent
  - Des valeurs en entrée
  - De valeurs calculées précédemment
  - De l'état dans lequel on se trouve
- Théories utilisées pour étudier/spécifier les différents types de circuits
  - Circuits combinatoires : algèbre de Boole
  - Circuits séquentiels : théorie des automates finis

Principe de fonctionnement

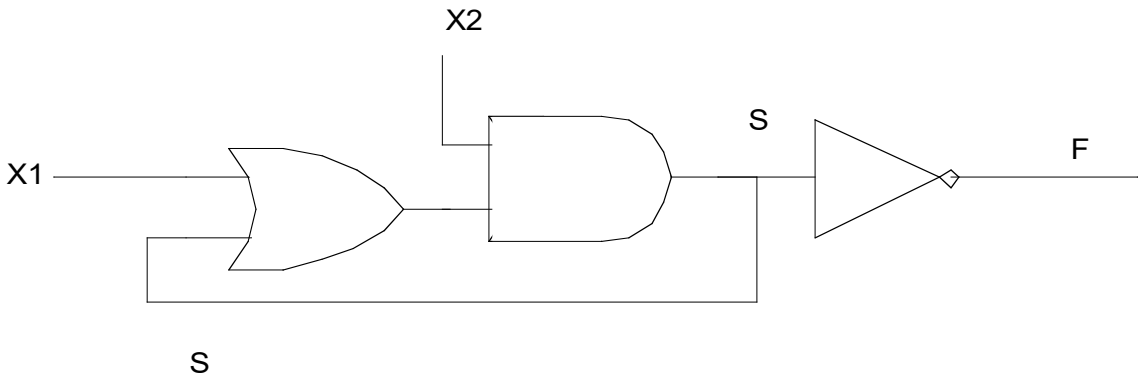
- Particularité de ce circuit
  - La sortie S du circuit est réinjectée à l'entrée du circuit
  - Rétroaction
  - L'état de sortie du circuit influencé par l'état antérieur



Principe de fonctionnement

- La table de vérité

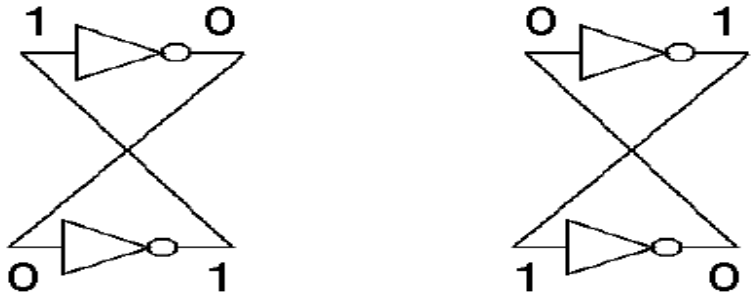
X1	*X2*	*S*	*F*
0	0	0	1
0	1	*S*	$\overline{S}$
1	0	0	1
1	1	1	0



  - L'état pour lequel X1=0 et X2=1 correspond à l'état de mémorisation du circuit séquentiel

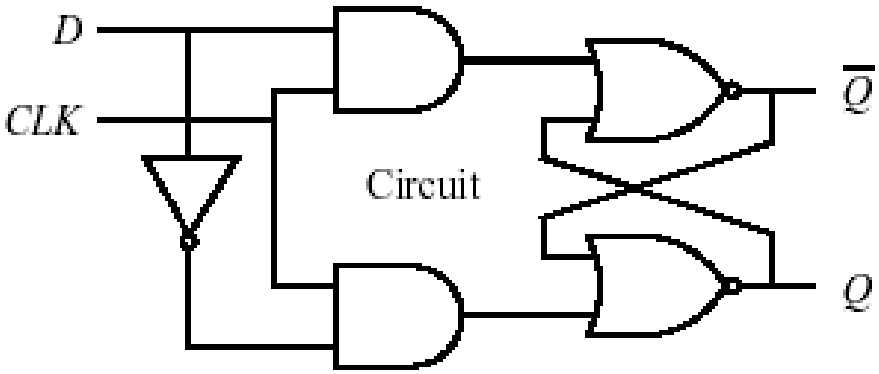
Bascules

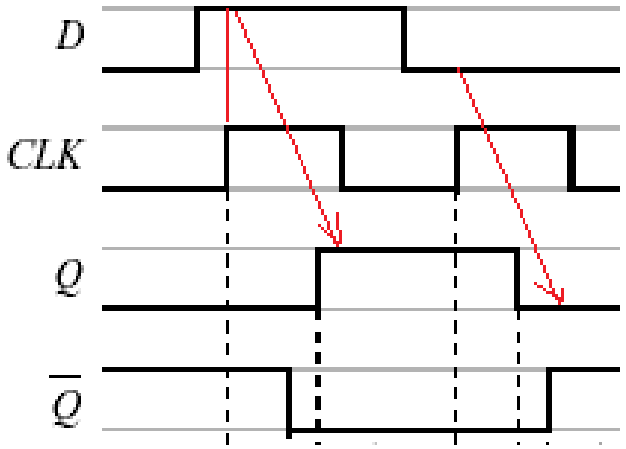
- Bistable : 2 états stables dans le temps
  - Principe général d'une bistable : 2 portes NON (inverseurs) en opposition



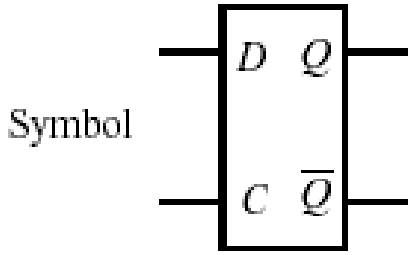
- Bascule : composant qui met en oeuvre une bistable
  - Possibilité de passer d'un état à l'autre, de changer l'état mémorisé
  - Plusieurs façons de gérer et changer l'état
  - Plusieurs types de bascules : RS, D, JK ...

Bascule D sur front d'horloge





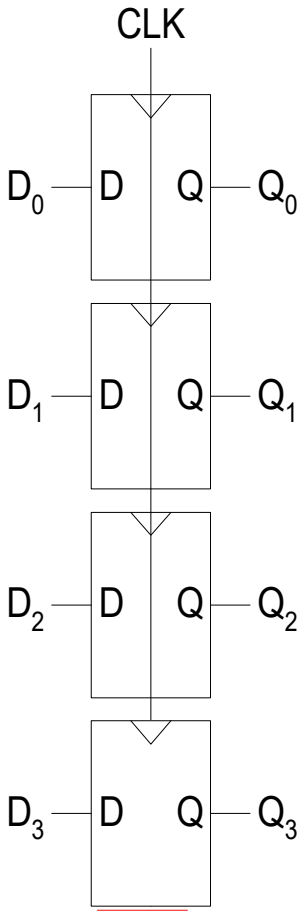
Timing Behavior

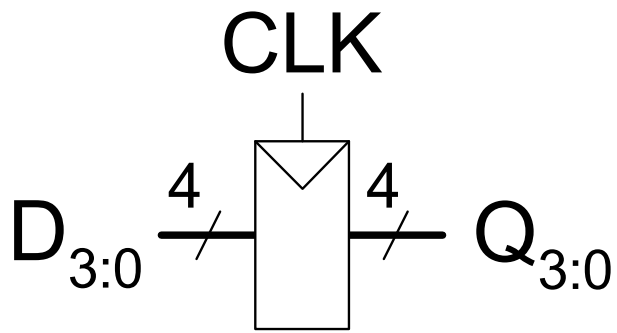


- La sortie recopie l'entrée à chaque front montant de CLK

Circuits séquentiels

En réunissant plusieurs bascules sur un même signal d'horloge, on peut fabriquer un circuit qui constitue un registre, d'où la possibilité de construire des mémoires

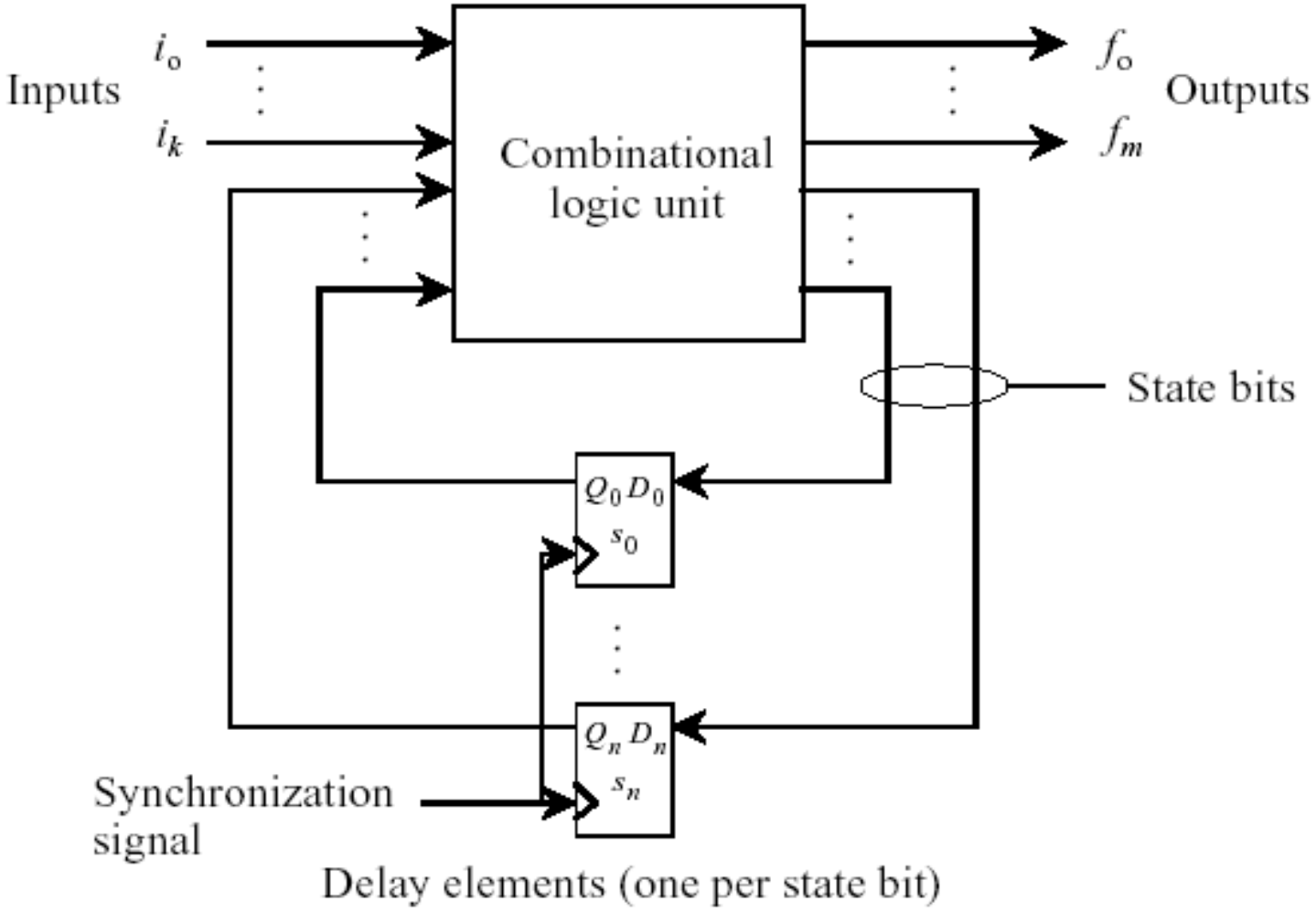




Registres

- Un registre est un dispositif qui permet de mémoriser une information et de la restituer autant de fois que désiré.
- Tout registre comporte un mécanisme de remise à zéro (RAZ), qui met tous les registres élémentaires qui le composent à zéro simultanément

Classical FSM Model



Automate fini

- Un automate fini possède un nombre fini d'états
- Il est caractérisé, pour un couple d'instants (t, t+1), par
  - Sa réponse S
  - Son entrée E
  - Son état Q

Synthèse d'un circuit séquentiel

- Pour réaliser la synthèse d'un circuit séquentiel il faut :

1. Déterminer le graphe des états (diagramme de transitions)
2. Construire la table d'états
3. Déterminer le nombre des bascules en choisissant un encodage
4. Construire la table de vérité
5. Réaliser les blocs combinatoires associés aux bits d'état future et aux sorties
6. Interconnecter les blocs registre et combinatoires

Synthèse d'un circuit séquentiel

- Diagramme d'états ou de transitions

– État : ce qu'il faut mémoriser de l'histoire du passé c-à-d
- jusqu'à l'instant t+1, pour pouvoir déterminer les sorties présentes S(t)

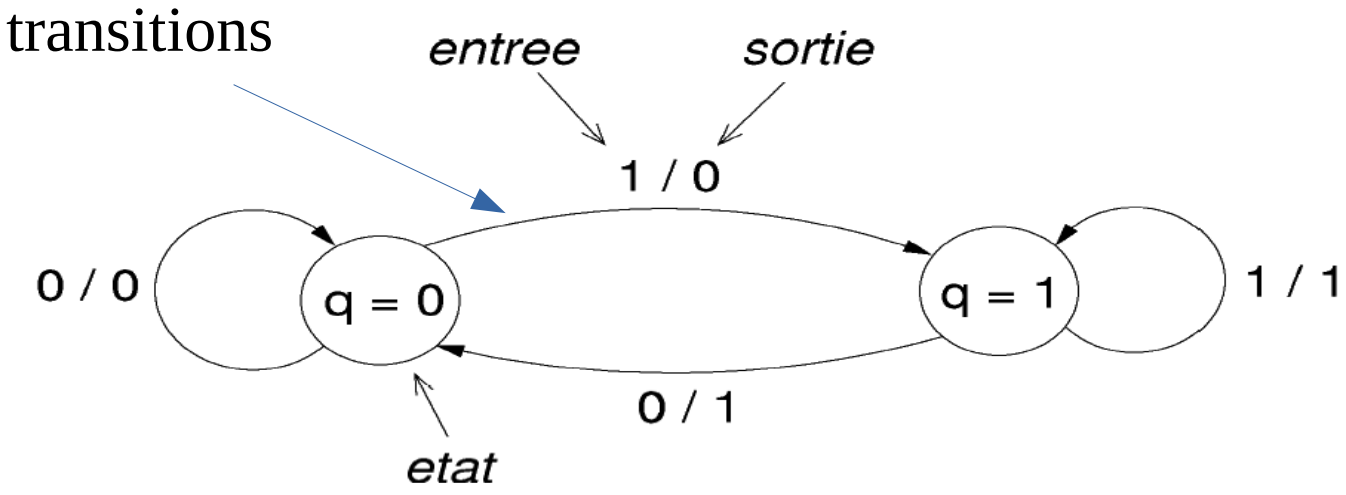
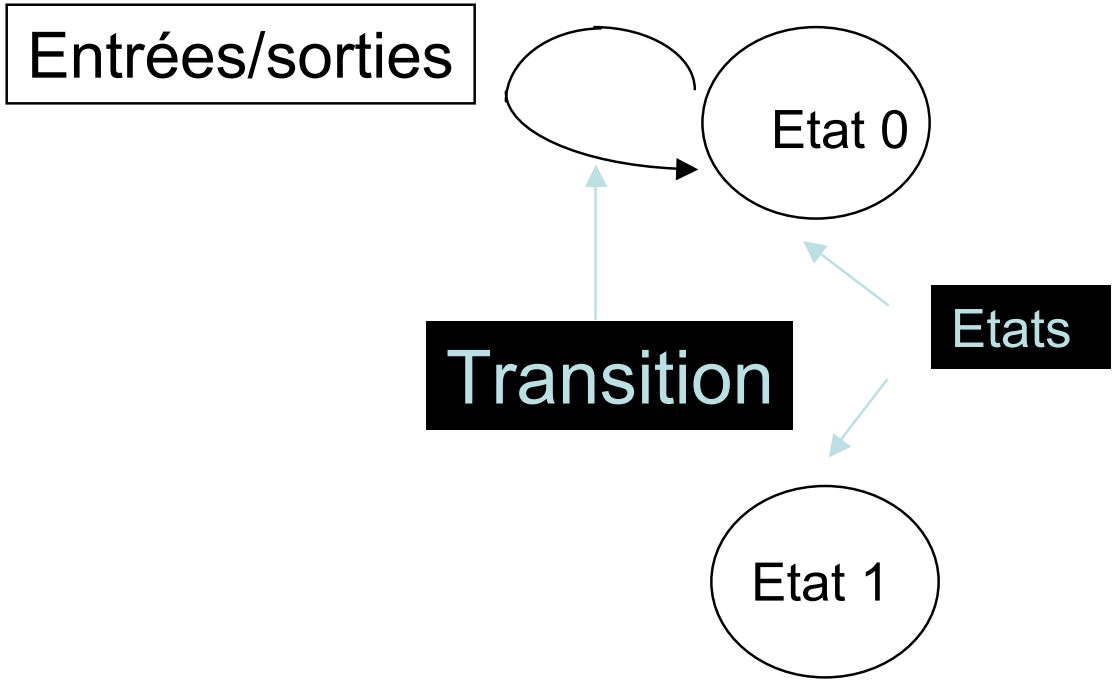


Diagramme d'états

– Après avoir défini les états, il faut compléter le graphe par les transitions du systèmes

– Une fonction de transition définit l 'évolution d 'un automate sous l'effet d 'un stimulus externe

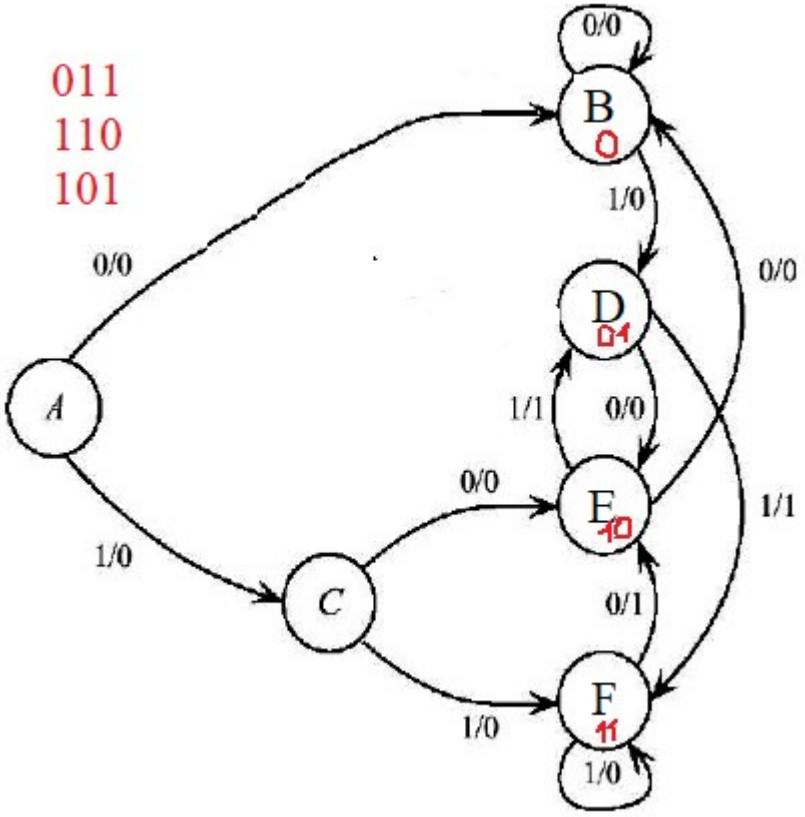


Circuits séquentiels

- ***Exemple***
- ***Détecteur de séquence***
  - ***Automate qui produit une sortie « 1 » lorsque exactement deux de ses trois entrées (1-bit série) sont à « 1 »***
  - ***Une séquence d'entrée 011011100 produira 001111010***
- Diagramme de transitions d'états
  - 1 entrée:
    - X
  - 1 sortie:
    - Z

Circuits séquentiels

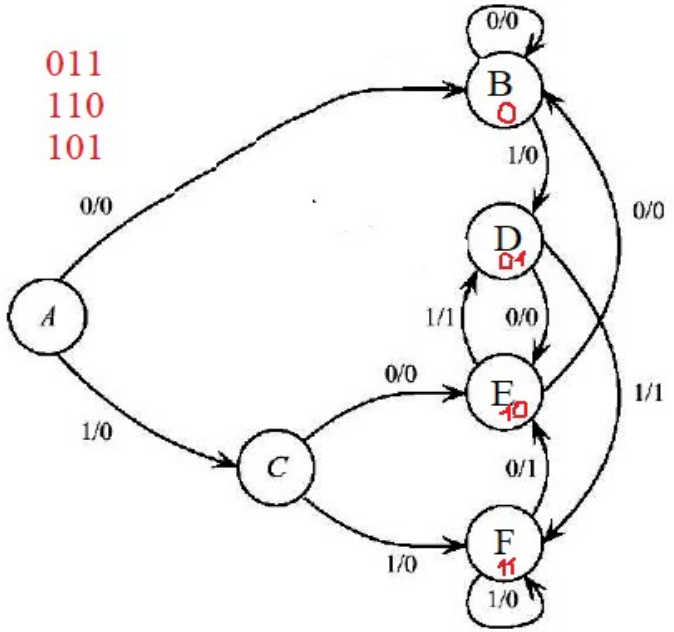
- ***Exemple***
- ***Détecteur de séquence***
  - ***Automate qui produit une sortie « 1 » lorsque exactement deux de ses trois entrées (1-bit série) sont à « 1 »***
  - ***Une séquence d'entrée 011011100 produira 001111010***
- Diagramme de transitions d'états
  - 1 entrée:
    - X
  - 1 sortie:
    - Z



Circuits séquentiels

- Table d'états
- 6 états (3 bits pour encoder en binaire)
- Choisir arbitrairement l'encodage:
  - A: 000, B:001; C:010; D:011; E:100; F:101;

Present state \ Input	X = 0	X = 1
A		
B		
C		
D		
E		
F		



Circuits séquentiels

- Table de transition d'états

Entrées / État actuel	X = 0	X = 1
A	B/0	C/0
B	B/0	D/0
C	E/0	F/0
D	E/0	F/1
E	B/0	D/1
F	E/1	F/0

Entrées / État actuel		X = 0 État future/Sorties	X = 1 État future/Sorties
A	000	B(001)/0	C(010)/0
B	001	B(001)/0	D(011)/0
C	010	E(100)/0	F(101)/0
D	011	E(100)/0	F(101)/1
E	100	B(001)/0	D(011)/1
F	101	E(100)/1	F(101)0

Circuits séquentiels

- Table de transition d'états

Entrées / État actuel		X = 0 État future/Sorties	X = 1 État future/Sorties
A	000	B(001)/0	C(010)/0
B	001	B(001)/0	D(011)/0
C	010	E(100)/0	F(101)/0
D	011	E(100)/0	F(101)/1
E	100	B(001)/0	D(011)/1
F	101	E(100)/1	F(101)0

	État à l'instant t			Entrée X	État à l'instant t+1				Sortie Z
	S2	S1	S0		S2'	S1'	S0'		
A	0	0	0	0	0	0	1	B	0
A	0	0	0	1	0	1	0	C	0
B	0	0	1	0	0	0	1	B	0
B	0	0	1	1	0	1	1	D	0
C	0	1	0	0	1	0	0	E	0
C	0	1	0	1	1	0	1	F	0
D	0	1	1	0	1	0	0	E	0
D	0	1	1	1	1	0	1	F	1
E	1	0	0	0	0	0	1	B	0
E	1	0	0	1	0	1	1	D	1
F	1	0	1	0	1	0	0	E	1
F	1	0	1	1	1	0	1	F	0
..					X	X	X		X

	État à l'instant t			Entrée X	État à l'instant t+1				Sortie Z
	S2	S1	S0		S2'	S1'	S0'		
A	0	0	0	0	0	0	1	B	0
A	0	0	0	1	0	1	0	C	0
B	0	0	1	0	0	0	1	B	0
B	0	0	1	1	0	1	1	D	0
C	0	1	0	0	1	0	0	E	0
C	0	1	0	1	1	0	1	F	0
D	0	1	1	0	1	0	0	E	0
D	0	1	1	1	1	0	1	F	1
E	1	0	0	0	0	0	1	B	0
E	1	0	0	1	0	1	1	D	1
F	1	0	1	0	1	0	0	E	1
F	1	0	1	1	1	0	1	F	0
..					X	X	X		X