Automates et circuits : Circuits combinatoires et circuits séquentiels

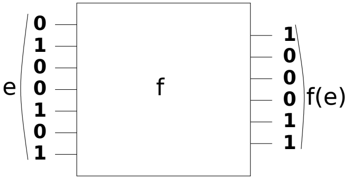
Nicolas Prcovic
Courriel : nicolas.prcovic@univ-amufr

Aix-marseille Université

Buts du chapitre (1)

Savoir réaliser n'importe quel circuit combinatoire

Etant donné la représentation d'une fonction (table de vérité ou expressions booléennes), savoir construire le circuit réalisant la fonction uniquement à l'aide de portes logiques.



Buts du chapitre (2)

Savoir réaliser des circuits séquentiels

Etant donné un calcul récursif à effectuer, construire le circuit séquentiel réalisant le calcul par étapes.

Nous aurons besoin de définir les mémoires, qui servent à stocker les résultats intermédiaires du calcul.

Circuit : fonction sur des entiers

Contexte

On souhaite réaliser un circuit qui donne le résultat d'une fonction f : A $\rightarrow$ B.

Ex : A $= \{\spadesuit, \heartsuit, \diamondsuit, \clubsuit\}$, B $=$ {rouge, noir}.

1. On code A et B sous forme d'entiers : on trouve des bijections $f_A : A \rightarrow \mathbb{N}_3$ et $f_B : B \rightarrow \mathbb{N}_1$
   Ex : $f_A(\spadesuit) = 0$, $f_A(\heartsuit) = 1$, $f_A(\diamondsuit) = 2$, $f_A(\clubsuit) = 3$, $f_B(rouge) = 0$, $f_B(noir) = 1$.
2. Mais on représente les entiers en base 2.
   Ex : $f_A(\spadesuit) = 00$, $f_A(\heartsuit) = 01$, $f_A(\diamondsuit) = 10$, $f_A(\clubsuit) = 11$, $f_B(rouge) = 0$, $f_B(noir) = 1$.

Table de vérités

A	$f_A(A)$	$f_B(B)$	B
♠	00	1	noir
♡	01	0	rouge
♢	10	0	rouge
♣	11	1	noir

On obtient une fonction booléenne b = f(a_1, a_0) :

a_1	a_0	b
0	0	1
0	1	0
1	0	0
1	1	1

Autre exemple : $f(x) = x^2$

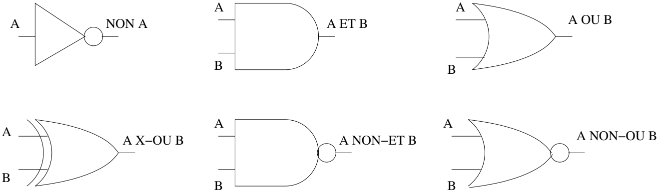
- x_1 et x_0 codent un nombre binaire $0 \leq x \leq 3$.
- y_3, y_2, y_1 et y_0 codent x^2.

x	x_1	x_0	y_3	y_2	y_1	y_0	x^2
0	0	0	0	0	0	0	0
1	0	1	0	0	0	1	1
2	1	0	0	1	0	0	4
3	1	1	1	0	0	1	9

Quatre fonctions booléennes : $y_3 = f_3(x_1, x_0)$, $y_2 = f_2(x_1, x_0)$, $y_1 = f_1(x_1, x_0)$ et $y_0 = f_0(x_1, x_0)$.

Rappels : portes logiques

- Un circuit électronique d'ordinateur est constitué d'un certain nombre d'entrées et de sorties binaires pouvant être à l'état 0 ou 1.
- N'importe lequel de ces composants peut être construit grâce à un assemblage de circuits à 1 ou 2 entrées et une sortie qu'on appelle *portes logiques*.



Rappels : table de vérité → fonction booléenne

1. A chaque combinaison de valeurs qui rend la fonction égale à 1, on associe le produit des variables en ajoutant la barre de la négation sur les variables dont la valeur est 0.

x	y	z	majorité	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{x}.y.z$
1	0	0	0	
1	0	1	1	$x.\overline{y}.z$
1	1	0	1	$x.y.\overline{z}$
1	1	1	1	$x.y.z$

Table de vérité $\longrightarrow$ fonction booléenne

2. Expression complète de f : on fait la somme des produits déterminés précédemment.
   Ex : pour la fonction majorité, on obtient
   $f(x, y, z) = \overline{x}yz \vee x\overline{y}z \vee xy\overline{z} \vee xyz$.
3. Quand c'est possible, on simplifie la formule (méthode des consensus, ...).
   Ex : $f(x, y, z) = yz \vee xz \vee xy$.

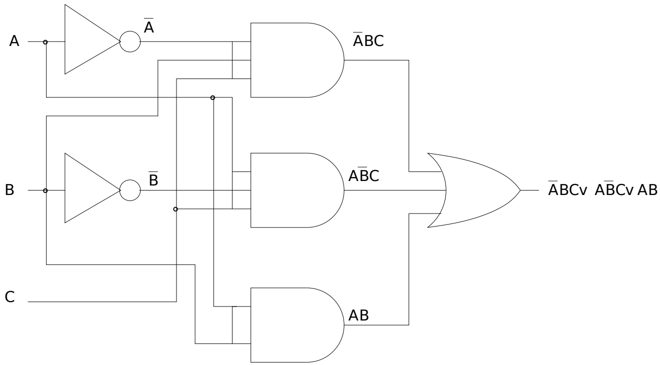
Fonctions booléennes → circuits

A partir d'une fonction booléenne sous forme normale disjonctive, on réalise ainsi le circuit correspondant :

- A chaque variable correspond une entrée du circuit.
- A toutes les variables qui apparaissent négativement on ajoute le circuit de la négation à l'entrée correspondante.
- On ajoute les circuits ET correspondant aux produits de l'expression booléenne.
- Enfin, chaque sortie des circuits OU sert d'entrée à une unique porte OU dont la sortie est celle du circuit logique entier.

Fonctions booléennes → circuits

Circuit correspondant à la formule
$f(A, B, C) = \overline{A}BC \vee A\overline{B}C \vee AB$.



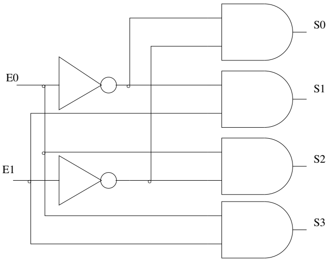
Les circuits logiques de base

Un certain nombre de circuits logiques sont très utilisés dans la conception d'une machine :

- Le décodeur
- Le multiplexeur
- Les additionneurs : demi-additionneur 1 bit, additionneur 1 bit, additionneur n bits

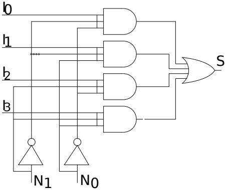
Le décodeur

- Un décodeur est un circuit a n entrées et 2^n sorties :
  - Les n entrées code un nombre i (de n bits).
  - Toutes les sorties sont à 0 sauf la i^{eme} qui est à 1.
- Utilité : chaque sortie d'un décodeur est potentiellement relié à un autre circuit. Sert donc à sélectionner un circuit à déclencher à partir de son numéro.



Le multiplexeur

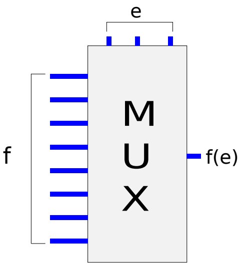
- Un *multiplexeur* a $2^n + n$ entrées et 1 sortie :
  - Les entrées sont réparties en 2 groupes :
    - le groupe de sélection qui a n entrées qui codent un nombre binaire i
    - le groupe d'information composé de 2^n entrées dont chacune code une information sur 1 bit.
  - En sortie : valeur du i^e bit du groupe d'information.
- Utilité : sélectionne le circuit dont on recueille l'info.



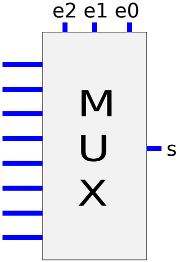
Le multiplexeur comme circuit général paramétrable

Un *multiplexeur* à $2^n + n$ entrées peut coder n'importe quelle fonction à n variables avec :

- une combinaison e de n valeurs d'entrées pour le groupe de sélection.
- une fonction caractéristique f pour le groupe d'information (2^n entrées)
- en sortie : f(e).



e2	e1	e0	s
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



Le demi-additionneur

- *Demi-additionneur* (1 bit) : consiste à additionner 2 bits a et b et à récupérer le bit de somme s et le bit de retenue r.
- Table de vérité :

a	b	r	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- Pas assez complet pour être utilisé pour faire des additions à plusieurs bits.

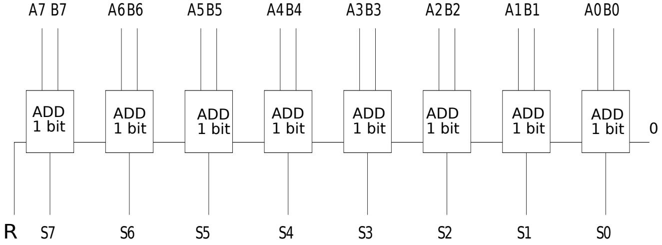
L'additionneur 1 bit

- L'*additionneur 1 bit* = demi-additionneur avec une entrée supplémentaire : la retenue r_e de l'addition précédente.
- Table de vérité :

r_e	a	b	r	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

L'additionneur n bits

- *Additionneur n bits* : un circuit à $2n$ entrées pour 2 entiers de n bits et $n+1$ sorties pour la somme des 2 entiers plus la retenue.
- Pour faire des additions sur n bits, il suffit d'assembler n additionneurs 1 bit :

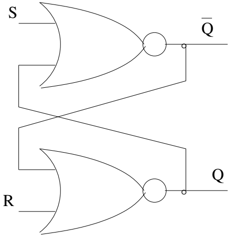


Réalisation des mémoires

- Jusqu'à présent nous avons montré des circuits dont les états internes sont uniquement fonctions des valeurs actuelles de leurs entrées.
- Les circuits mémoires possèdent des états internes qui dépendent aussi des anciennes valeurs qu'ils avaient en entrée.

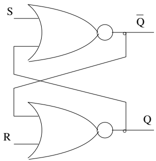
Bascule RS

- Une bascule RS (RS pour Reset/Set) est le circuit de base permettant de constituer une mémoire d'un bit.



- On note les 2 sorties Q et Q' (on note cette dernière aussi $\overline{Q}$ car sa valeur est souvent l'opposée de Q).

Bascule RS

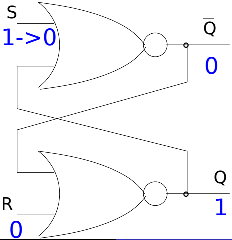


- $Q = \overline{R \vee Q'}$ et $Q' = \overline{S \vee Q}$
  donc $Q = \overline{R \vee \overline{S \vee Q}} = \overline{R}.(S \vee Q)$
  et $Q' = \overline{S \vee \overline{R \vee Q'}} = \overline{S}.(R \vee Q')$
- "Table de vérité" :

R	S	Q	Q'
0	0	"Q"	"Q'"
0	1	1	0
1	0	0	1
1	1	0	0

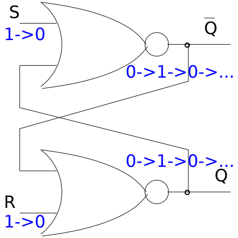
Bascule RS : quand $R = S = 0$

- Les relations $Q = \overline{R}.(S \vee Q)$ et $Q' = \overline{S}.(R \vee Q')$ deviennent $Q = Q$ et $Q' = Q'$.
  Les sorties conservent la valeur qu'elles avaient précédemment.
- Les valeurs de Q et de Q' peuvent être égales à 0 ou à 1 mais on a toujours $Q = \overline{Q'}$.



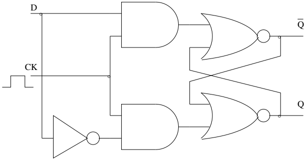
Bascule RS : cas problématique

- Si on passe directement de $R = S = 1$ (qui fait que $Q = Q' = 0$) à $R = S = 0$, les valeurs des sorties peuvent osciller avant de se fixer. On ne peut déterminer à l'avance quelles vont être les valeurs de Q et Q'.



- Il faut donc compléter ce circuit afin qu'il ne puisse y avoir $R = S = 1$ en entrée. C'est ce que va réaliser une bascule D.

Bascule D



- Si $CK = 0$ alors quelle que soit la valeur de D on a $R = S = 0$ et donc les états Q et $\overline{Q}$ gardent leurs valeurs.
- Si $CK = 1$ alors Q = D.
- On peut donc dire que cette bascule mémorise l'ancienne valeur de D tant que $CK = 0$.
(Dès que CK passe à 1, Q prend la valeur de D.)

L'horloge interne

- Habituellement, CK est relié à la sortie d'un dispositif spécial appelé *horloge interne*.
- Une horloge émet à intervalle régulier des tensions correspondant à l'état 1 pendant un court moment avant de revenir à une tension correspondant à l'état 0.
  - Toutes les entrées CK sont reliées à l'horloge interne de l'UC.
  - A chaque cycle d'horloge, l'état global du système constitué par l'ensemble des circuits change : chaque entrée D de bascule est recopiée sur sa sortie Q.

Succession d'états d'un ordinateur

- Un ordinateur en train de fonctionner peut être vu comme passant périodiquement d'un état global de ses circuits à un autre.
- La fréquence de changement est celle de l'horloge interne du micro-processeur (actuellement env 3Ghz).

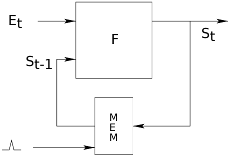
Différence circuit combinatoire/séquentiel

- Soient les vecteurs d'entrées E_t =$(e_{1,t}, \ldots e_{n,t})$ et de sorties S_t =$(s_{1,t}, \ldots s_{m,t})$ à un temps t donné.
- Un circuit combinatoire réalise une fonction F :

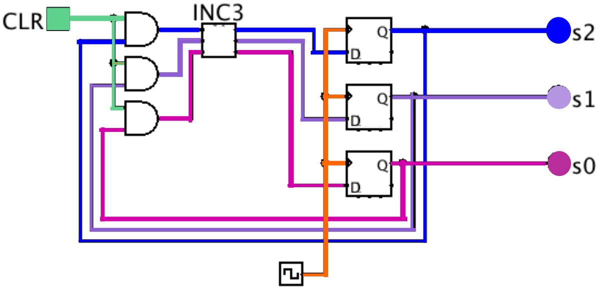
$$S_t = F(E_t)$$

- La fonction F d'un circuit séquentiel dépend aussi de S_{t-1} grâce aux résultats mémorisés :

$$S_t = F(E_t, S_{t-1})$$



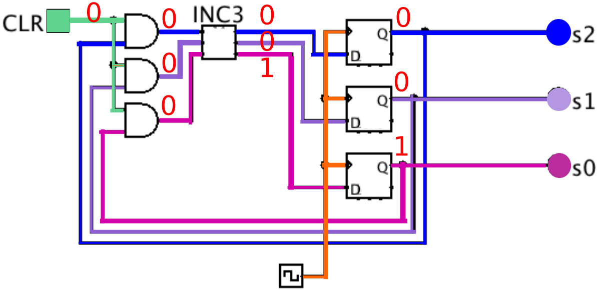
Exemple : compteur à 3 bits



Circuit énumérant cycliquement les entiers de 000 à 111.

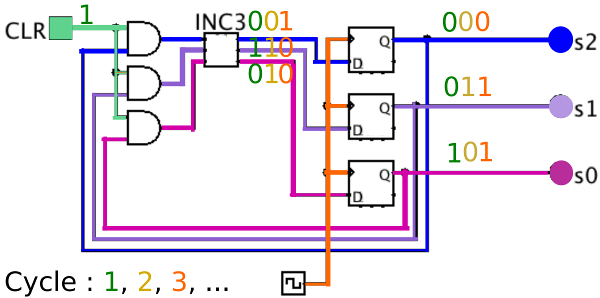
- INC3 réalise l'incrémentation sur 3 bits.
- Les 3 bascules D mémorisent l'entier sur 3 bits.

Compteur à 3 bits : initialisation



- CLR à 0 permet d'initialiser le circuit (et de bloquer les sorties à 001).
- Dès qu'on passe CLR à 1, le compteur démarre.

Compteur à 3 bits : exécution



Cycle 1 :

- 001 (sorties Q) passe les portes ET, arrive en entrée de INC3.
- 010 sort de INC3 et reste bloqué sur les entrées D jusque'à la fin du cycle d'horloge.

A chaque nouveau cycle, on a un nouveau successeur en sortie du circuit.