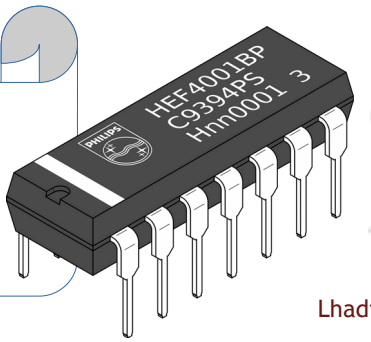
Structure Machine 2

A la découverte des circuits logiques

Avril à Juin 2020



Lhadi BOUZIDI
Lhadi.bouzidi@gmail.com

Ce cours intitulé structure machine 2 complète le cours de structure machine 1 que vous avez suivi lors du premier semestre. Je rappel que durant ce 1er semestre vous aviez découvert les notions de base vous permettant de cerner les connaissance de base permettant la conception des composants des systèmes numériques (digitaux). Ainsi, vous aviez découvert les système de numération, le codage de l'information et l'algèbre de Boole. Durant ce second semestre, vous allez approfondir vos connaissance en abordant la conception des circuits logiques combinatoires, la conception des circuits logiques séquentiels et les caractéristiques des circuits intégrés.

Le cours est proposé sous forme d'une séance de cours et d'une séance de travaux dirigés par semaine durant un semestre.

Public : Étudiants de 1ère année MI	**Responsables de matière** : Mr BOUZIDI Lhadi
Année universitaire : 2019/2020	**Chargé de cours** : Mr BOUZIDI Lhadi
Crédits : 4	**Coefficient** : 2
Cours : 1h30	**TD** : 1h30
Code de cette unité d'enseignement fondamentale : UEF22	

Chargés de TD	Nombre Groupes	Groupes

Planning pédagogique :

Non encore défini vu la situation liée à la pandémie du corona-virus

Évaluation des apprentissages :

- Les apprentissages qui seront acquis à l'issue de ce cours seront évalués au travers :
    - Un examen final (EMD) en fin de cours.
    - Une évaluation continue basée sur deux interrogations et le suivi de la participation et de l'assiduité.

- La première interrogation doit avoir lieu avant une date qui sera définie ultérieurement (après la première série de TD). Sa durée doit être de **45 minutes**. Elle doit porter sur le **chapitre I (circuits logiques combinatoires)**.

- La seconde interrogation doit avoir lieu durant le déroulement de la troisième série de TD mais impérativement avant une date qui sera définie ultérieurement . Elle portera sur les **chapitres II et III**. Sa durée est de **45 minutes**.

- Le calcul de la note de l'évaluation continue (TD) sera comme suit: Note de TD sur 20 points réparties comme suit :

Interrogation 1	Interrogation 2	Assiduité	Participation
7 points	8 points	2 points	3 points

- Les modalité de l'évaluation de l'assiduité seront définies ultérieurement (vue la situation de la pandémie)
- L'évaluation de la participation consiste à vérifier la préparation des exercices et la participation en classe.

- Les enseignants sont tenus d'organiser une séance de consultation pour l'examen final.

Plan du cours

Introduction

Chapitre 1 : Les circuits logiques combinatoires (CLC)
- définition
- étapes de conception
- Étude de quelques CLC (Additionneur, décodeur, codeur, multiplexeur, démultiplexeur, comparateur, etc.)
- logique structurée (ROM, PAL, FPGA, etc.)

Chapitre 2 : Les circuits logiques séquentiels
- Définition
- Les bascules RS, JK et D
- Les registres
- Les mémoires
- Synthèse d'un circuit logique séquentiel (automates)
- Réalisation d'automates (compteur / décompteur)

Chapitre 3 : Les circuits intégrés (CI)
- Définition
- caractéristiques
- Exemple d'un montage d'un circuit combinatoire simple utilisant des CI

Important : Vue la pandémie du au corona-virus,
certaines informations ci-dessus sont susceptibles d'être modifiées.

Introduction

Je ne sais pas si vous vous rappeler lors du semestre passé, je vous ai dit que les systèmes électroniques que nous utilisons sont construits à base circuits analogiques et numériques. J'ai expliqué que le transfert de certains type d'information comme le son ou l'image nécessite l'utilisation de composants électroniques analogiques comme ceux utilisés dans des baffles ou des microphones. En réalité, depuis quelques décennies, pour le stockage, le traitement et même le transfert de l'information, des composant numériques sont de plus en plus utilisés. Ceci montre bien que deux types de composants électroniques coexistent dans nos systèmes numériques comme les ordinateurs ou les smartphones. Dans ce cours, nous nous intéressons uniquement aux composants relevant de l'électronique numérique ou digitale (composant numériques) qui sont conçus à base de circuit logiques.

Les circuits logiques sont utilisés presque à tous les niveaux de notre vie quotidienne. Ils constituent la base de la technologie numérique actuelle qui a permis un développement extraordinaire des moyens de communication, de transport et des processus industriels...

On définie un circuit logique comme un circuit électronique réalisant une ou plusieurs fonction(s) logique(s). Il est définit par l'interconnexion d'un ensemble de portes logiques mettant en relation des sorties avec des entrées. Ainsi, globalement, on peut schématiser un circuit logique sous la forme d'une boite noire ayant des entrées (binaires) et des sorties binaires (les fonctions logiques). Bien évidemment, la boite noire est composée d'un ensemble de portes logiques interconnectées pouvant être schématisées par un schéma que l'on nomme par logigramme. Deux grandes familles de circuits logiques existent : les circuits logiques combinatoires et les circuits logiques séquentiels.

D'un point de vue conceptuel, nous avons déjà vue beaucoup de choses concernant les circuits logiques combinatoires. En générale ; il sont utilisés pour faire des calculs ou des traitement et des transferts d'information. Malheureusement, ces circuits, seuls, ne peuvent pas assurer la fonction de mémorisation de l'information. Leur base conceptuelle ne prévoit pas la prise en compte de l'état passé de leurs sorties. Donc ça ne mémorise pas l'information. C'est pour cela qu'une autre base conceptuelle est utilisée pour concevoir des circuits permettant de remplir cette fonction de mémorisation de l'information : C'est la méthode de conception de circuits séquentiels.

Dans ce qui suit, nous reviendrons, bien évidement sur la conception de circuits logiques combinatoires, et nous ferons la synthèse de quelques uns de ces circuits comme l'additionneur, le décodeur ou le comparateur. Ensuite, nous présenterons les composants qui sont à la base de la fonction « mémoire ». Ces composant sont appelés « bascules ». Par la suite, nous expliquerons quelques circuits séquentiels très importants comme la mémoire, le registre ou le compteur. Nos terminerons les circuits logiques séquentiel par présenter une méthode de synthèse en l'illustrant par la synthèse d'un compteur.

Ce que je viens d'évoquer est de la théorie, mais dans la pratiques, les circuits logiques sont, en générale, utilisés sous forme de circuits intégrés. Nous présenterons, donc, les caractéristiques de ces circuits intégrer en présentant un montage simple d'un circuits combinatoire à base de circuits intégrés.

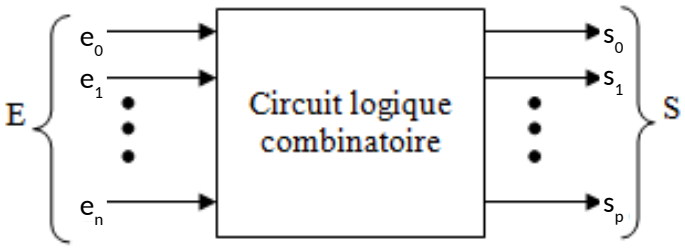
Chapitre 1 : Les circuits logiques combinatoires (CLC)

2.1 Introduction

Je vous rappel que l'on distingue deux types de circuits logiques : les circuits combinatoires et les circuits logiques séquentiels. Commençons d'abord par les circuits combinatoires. Mathématiquement parlant, on peut les définir par une équation reliant ses sorties à ses entrées. En considérant $E = (e_0, e_1, \ldots, e_n)$ comme les entrées de notre circuit et $S = (s_0, s_1, \ldots, s_p)$ comme ses sorties alors, on peut écrire :

$S = F(E)$ où $s_i = f_i(e_0, e_1, \ldots, e_n)$ avec i allant de 0 à p.

D'un point vue schématique on aura :

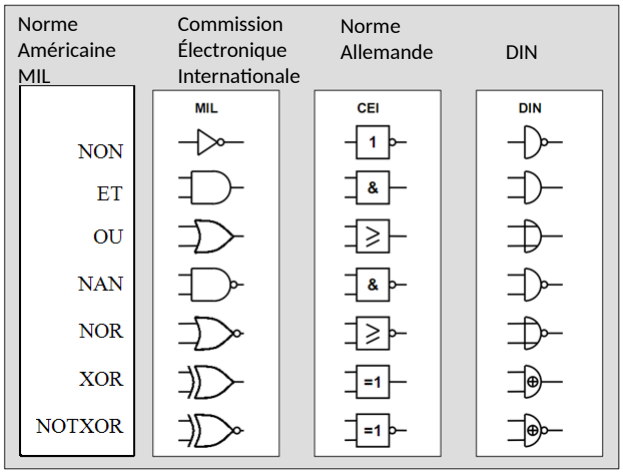


Dans ce qui suit, nous allons définir ce que c'est que la synthèse et l'analyse de circuits logiques combinatoires et nous allons présenter la synthèse que quelques circuits courants.

Remarque :

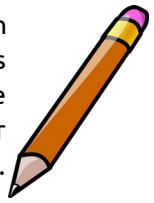
- Dans le chapitre précédent, nous parlions d'opérateurs logiques de base. Dans ce chapitre nous parleront plutôt de porte logiques (ET, OU, NON, NAND, NOR et XOR). Pour traiter ce chapitre, vous devez impérativement connaître l'algèbre de Boole.

- Nous allons nous servir de symboles pour représenter les portes logiques. Vous allez vous rendre compte que dans la littérature, on utilise des symboles différents pour représenter les mêmes portes logiques. En fait, il existe plusieurs normes de représentations :



Dans ce qui suit, nous utiliserons la norme MIL.

2.2 Analyse et synthèse de circuit logiques combinatoires



La synthèse d'un circuit logique combinatoire a pour but la réalisation d'une fonction logique qui remplit un cahier des charges et qui satisfait également à d'autres critères tels que le coût et l'encombrement minimum par exemple. Le nombre de circuits à produire, le matériel à disposition, le délai de réalisation, etc. sont d'autres paramètres dont il faut tenir compte lors de la synthèse. De façon générale, la simplification d'un circuit est toujours utile.

Concrètement, il s'agit de déterminer le logigramme associé à une fonction logique connaissant la définition de cette dernière.

Voici les étapes à suivre pour réaliser la synthèse d'un circuit logique combinatoire :

1. Établir la table de vérité de chacune des fonctions impliquées dans le problème à traiter
2. Établir les équations logiques ou la table de Karnaugh
3. Simplifier les équations de chacune des fonctions logiques
4. Établir le logigramme du circuit logique
5. Réaliser le circuit logique

L'analyse d'un circuit logique consiste à trouver à partir d'un logigramme ses fonctions logiques. Le principe de l'analyse d'un circuit logique consiste à :

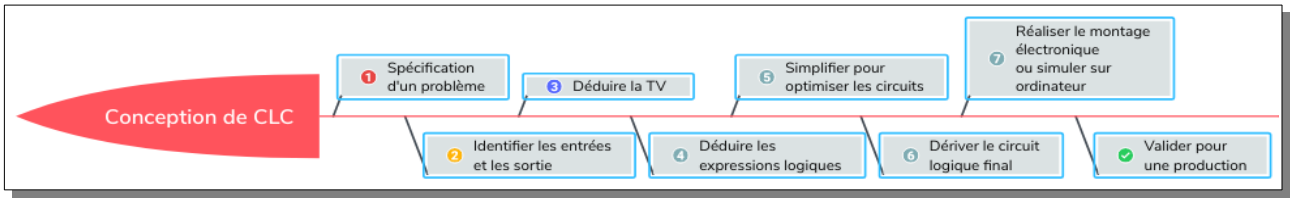
- Donnez l'expression de chaque porte en fonction des valeurs de ses entrées.
- En déduire au final la ou les fonction(s) logique(s) du circuit analysé.

On peut ensuite établir la table de vérité du circuit analysé et opéré à une simplification à 'laide des propriétés de l'algèbre de Boole de la table de Karnaugh.

2.3 Conception d'un CLC

La procédure de conception des circuits logiques combinatoires commence par la spécification du problème et comprend les étapes suivantes :

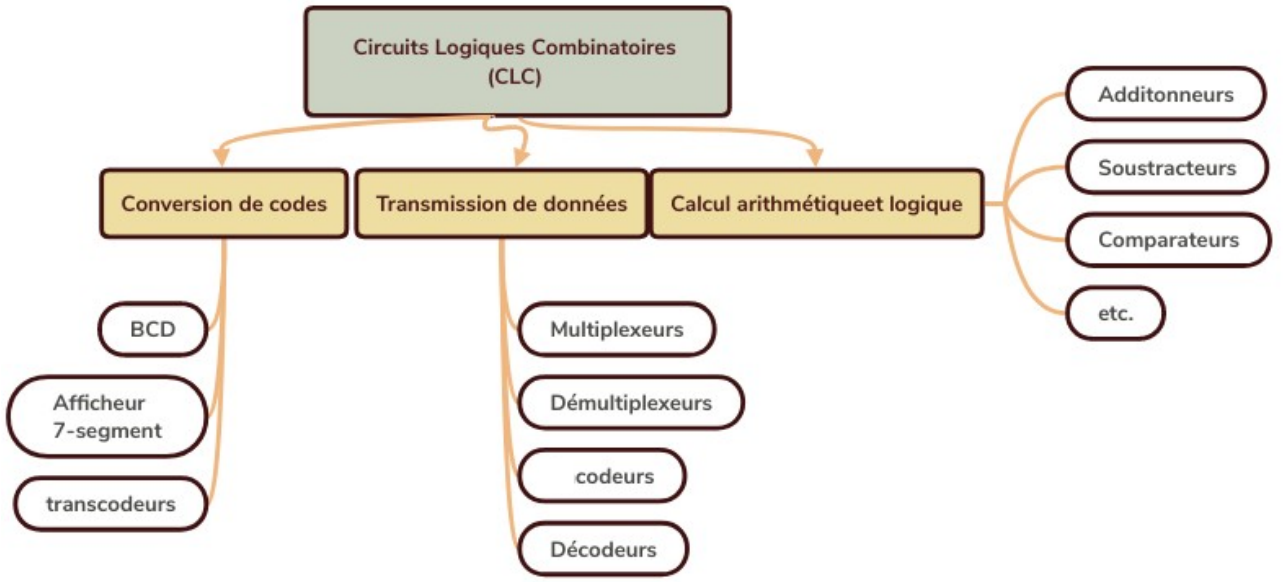
1. Déterminer le nombre requis d'entrées et de sorties à partir des spécifications du problème.
2. Déterminer la table de vérité pour chacune des sorties en fonction de leurs liens avec les entrées.
3. Simplifier l'expression booléenne pour chaque sortie. Utilisez les tableaux de Karnaugh ou l'algèbre booléenne.
4. Dessinez un diagramme logique (logigramme) qui représente l'expression booléenne simplifiée. Vérifier la conception en analysant ou en simulant le circuit.



2.4 Classification

On distingue au moins 3 classes de circuits logiques combinatoires :

- Les circuits de calcul arithmétiques et logiques,
- les circuits d'aiguillage et de transmission de données,
- les convertisseurs de codes.



Les circuits de calcul arithmétiques et logiques

Ce sont généralement des circuits logiques combinatoires permettant d'effectuer des calculs arithmétiques (addition, soustraction, multiplication) sur des entiers ou des nombre en virgule flottantes et des opérations logiques comme des négations, des ET, des OU ou des OU Exclusifs. On les trouve le plus souvent dans les unité de calculs des ordinateur communément appelées **UAL** ou unité arithmétique et logique (*ALU* en anglais!).

Les circuits de transmission de données

C'est un groupe de circuits permettant d'aiguiller les informations (données) binaires à travers des lignes électriques (souvent appelé BUS) d'une source (une petite mémoire appelée registre ou des capteurs, interrupteurs ou boutons poussoirs) vers une destination (registre ou un afficheur par exemple). Le décodeur, le multiplexeur en sont des exemples.

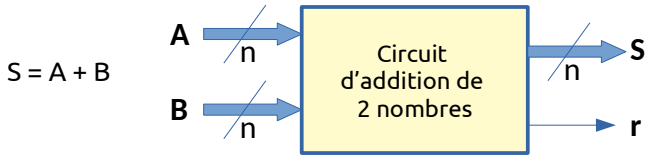
Les convertisseurs de code

Les nombres sont habituellement codés sous une forme ou une autre afin de les représenter ou de les utiliser au besoin. Par exemple, un nombre 'sept' est codé en décimal à l'aide du symbole $(7)_{10}$. Ce nombre est affiché sur votre calculatrice en se servant du codage 7 segments, mais au sein de l'unité de calcul de votre calculatrice, ce même nombre est codé en général en complément à 2. Bien que les ordinateurs numériques traitent tous des nombres binaires, il y a des situations où la représentation binaire naturelle des nombres n'est pas pratiques ou est inefficace ce qui nécessite des codes plus appropriés. Cette situation fait cohabiter, dans une même machine, diverses codes pour représenter une même information. Des circuits de conversion d'un code vers un autres sont donc utiliser.

2.3 Étude de quelques CLC

Additionneur

Il s'agit de faire la synthèse d'un circuit d'addition de 2 nombres ***A*** et ***B*** sur ***n bits*** chacun. Ce circuit doit avoir en entrée 2n variables (ce qui correspondant aux nombres A et B qui sont sur n bits chacun) et en sortie n bits qui correspondront à la sortie S et un seul bit qui correspond à la retenue finale du calcul.

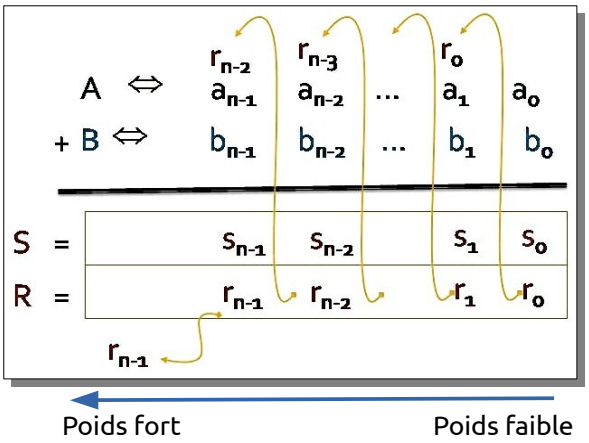


L'analyse rapide de ce problème nous apprend que nous avons ***2n*** entrées et ***n+1*** sorties au moins. Maintenant essayons de comprendre le lien entre les sorties et les entrées. Ce lien (ou cette relation) est déduite, bien évidemment des règles d'addition que nous avons déjà vue dans le chapitre sur les systèmes de numération (rappelez-vous l'arithmétique binaire!).

Si je me rappel bien, j'avais présenté 4 règles que voici :

0+0 = 0	retenue 0
0+1 = 1 + 0 = 1	retenue 0
1 + 1 = 0	retenue 1
1+1+1 = 1	retenue 1

Ces règles sont applicables pour chaque niveau des bits des deux nombres A et B. Ainsi le calcul se fait en allant du bit de poids faible vers le bits de poids fort.



Comme je l'ai dit ci-dessus, nous avons ***n*** sorties pour constituer la somme ***S*** et une sortie supplémentaire pour représenter la dernière retenue. Ainsi, nous pourrions être tentés de penser que ces ***n+1*** sorites dépendent des ***2n*** variables en entrées (rappelez-vous, n entrées pour constituer A et n entrées pour constituer B!). Mais l'analyse plus fine des règles de l'addition nous apprend que ce n'est pas vraie. En effet, une sortie s_i ne dépend pas de toutes les entrées a_j (j allant de 0 à n-1) du nombre ***A*** et b_k (k allant de 0 à n-1) du nombre **B**. En fait, le calcul de l'addition se fait étage par étage en allant de la gauche vers la droite (du poids faible vers le poids forts de nos deux nombres A et B).

Pour le premier étage nous avons 2 sorites : s_0 et r_0. Ces sorties sont des fonctions f_0 et g_0 de 2 entrées uniquement a_0 et b_0.

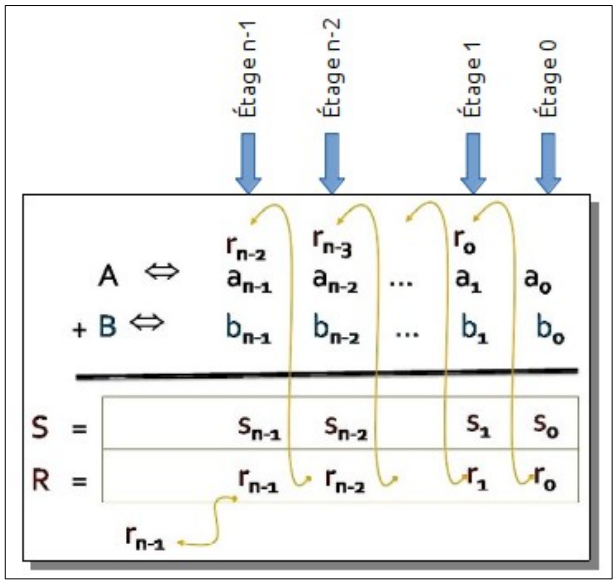
On écrira donc :

- ✔ $s_0 = f_0(a_0, b_0)$
- ✔ $r_0 = g_0(a_0, b_0)$.

Pour les autres étages, on voit aussi que les sorties s_i et r_i ne dépendent que des entrées a_i, b_i et r_{i-1}. Pour chacun de ces étage, on utilisera les mêmes règles de calcul (l'addition binaire).

On écrira donc :

- ✔ $s_i = f_1(a_i, b_0, r_{i-1})$
- ✔ $ri = g_1(a_0, b_0, r_{i-1})$.



En appliquant les règles de l'addition binaire, trouvons maintenant les tables de vérité des fonctions f_0 , g_0 , f_1 et g_1 en d'autres termes s_0 , r_0 , s_i et r_i

	a_0	b_0	s_0	r_0
m_0	0	0	0	0
m_1	0	1	1	0
m_2	1	0	1	0
m_3	1	1	0	1

	a_i	b_i	r_{i-1}	s_0	r_0
m_0	0	0	0	0	0
m_1	0	0	1	1	0
m_2	0	1	0	1	0
m_3	0	1	1	0	1
m_4	1	0	0	1	0
m_5	1	0	1	0	1
m_6	1	1	0	0	1
m_7	1	1	1	1	1

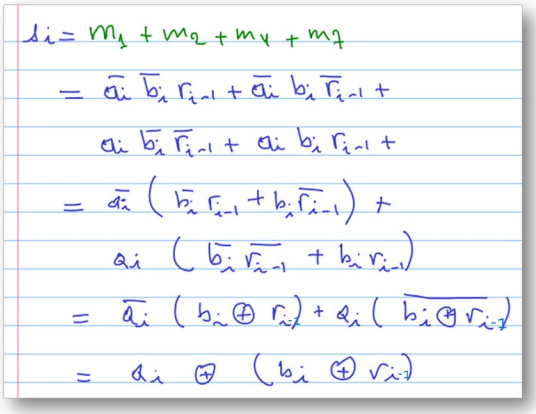
La détermination des équations de s_0 et r_0 est très simple :

- $s_0 = m_1 + m_2 = \quad \bar{a}_0 . b_0 + a_0 . \bar{b}_0 = a_0 \oplus b_0$
- $r_0 = a_0 . b_0$

La détermination des équations de s_i et r_i n'est pas trop compliqué ! :

- $s_i = m_1 + m_2 + m_4 + m_7$
- $s_i = m_3 + m_5 + m_6 + m_7$

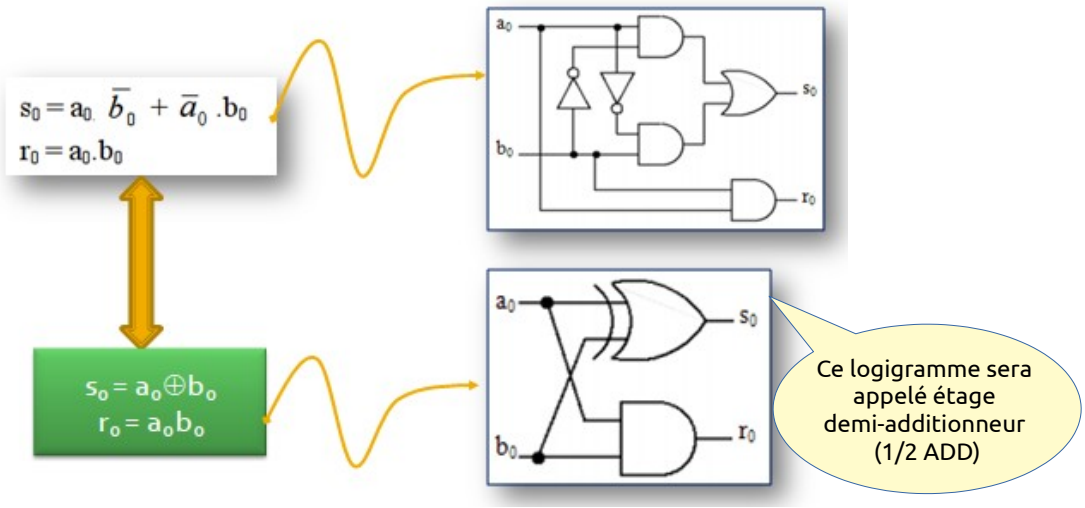
Essayons de simplifier la somme s_i:



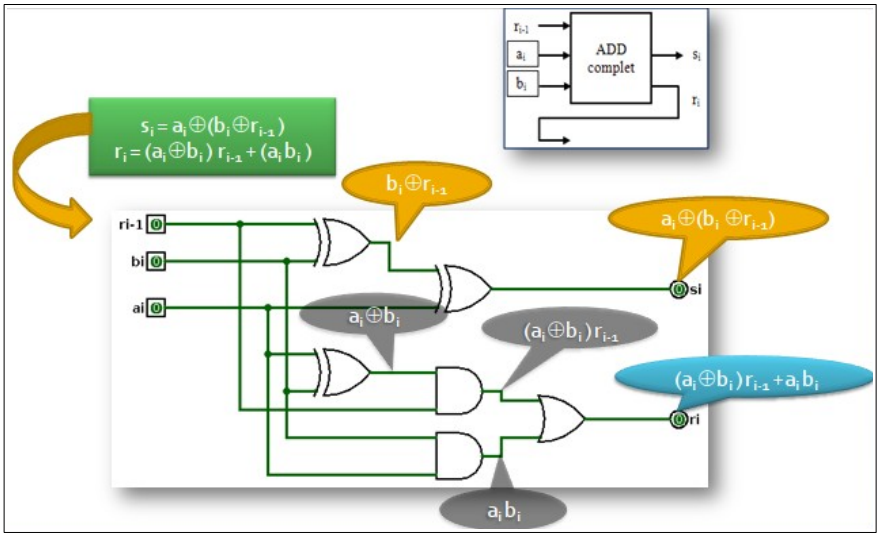
Simplifions maintenant la retenue r_i :

$$r_i = m_3 + m_5 + m_6 + m_7$$
$$= \bar{a_i} b_i r_{i-1} + a_i \bar{b_i} r_{i-1} +$$
$$a_i b_i \overline{r_{i-1}} + a_i b_i r_{i-1}$$
$$= \bar{a_i} b_i r_{i-1} + a_i \bar{b_i} r_{i-1} + a_i b_i (\overline{r_{i-1}} + r_{i-1})$$
$$= \bar{a_i} b_i r_{i-1} + a_i \bar{b_i} r_{i-1} + a_i b_i$$
$$= (\bar{a_i} b_i + a_i \bar{b_i}) r_{i-1} + a_i b_i$$
$$= (a_i \oplus b_i) r_{i-1} + a_i b_i$$

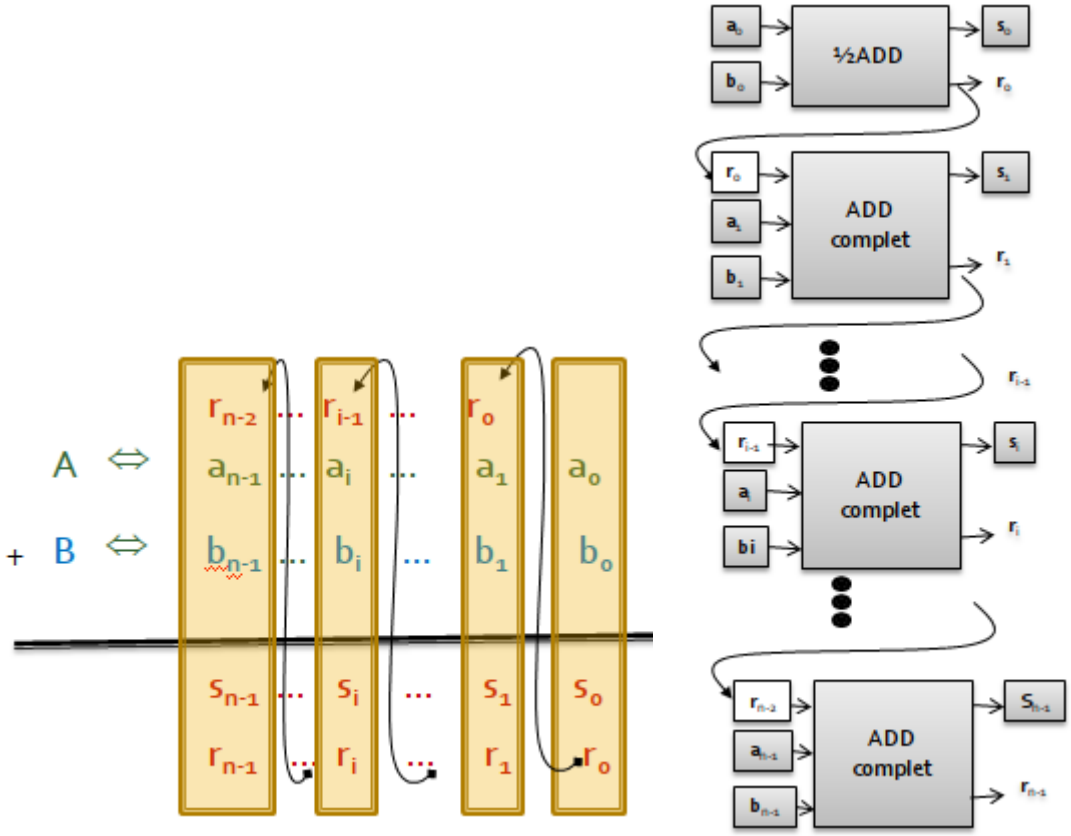
Au finale nous obtenons les logigrammes pour le premier étage de notre circuit d'addition:



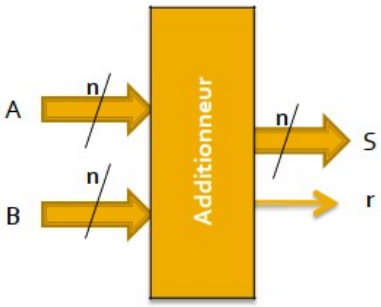
Nous obtenons le circuit logique suivant pour les sommes s_i et retenues r_i (i allant de 1 à n-1) :

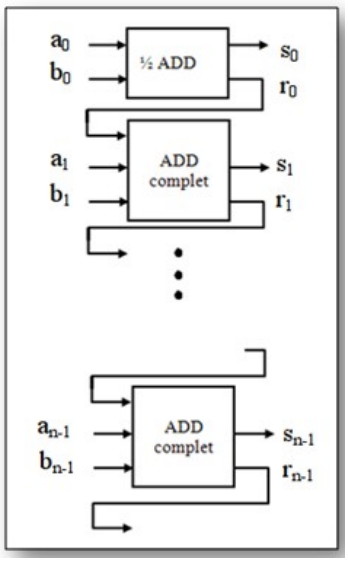


En utilisant le 1/2ADD pour le premier étage de l'additionneur et l'ADD complet pour les étages suivants on obtient le circuit suivant avec un montage en cascade :



Ce qui donne en définitif le circuit suivant :

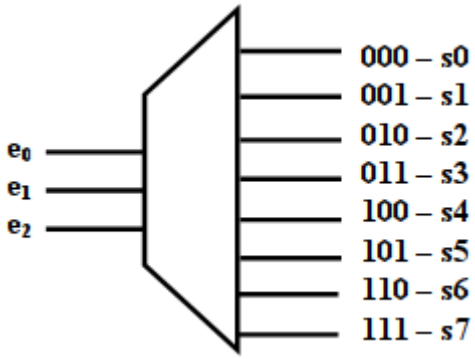




Il faut noter que le circuit additionneur fait partie d'une unité dite Unité Arithmétique et Logique (**UAL**) des ordinateurs. Il peut aussi être vendu sous forme d'un circuit intégré. Nous le verront dans le chapitre 3.

Décodeur

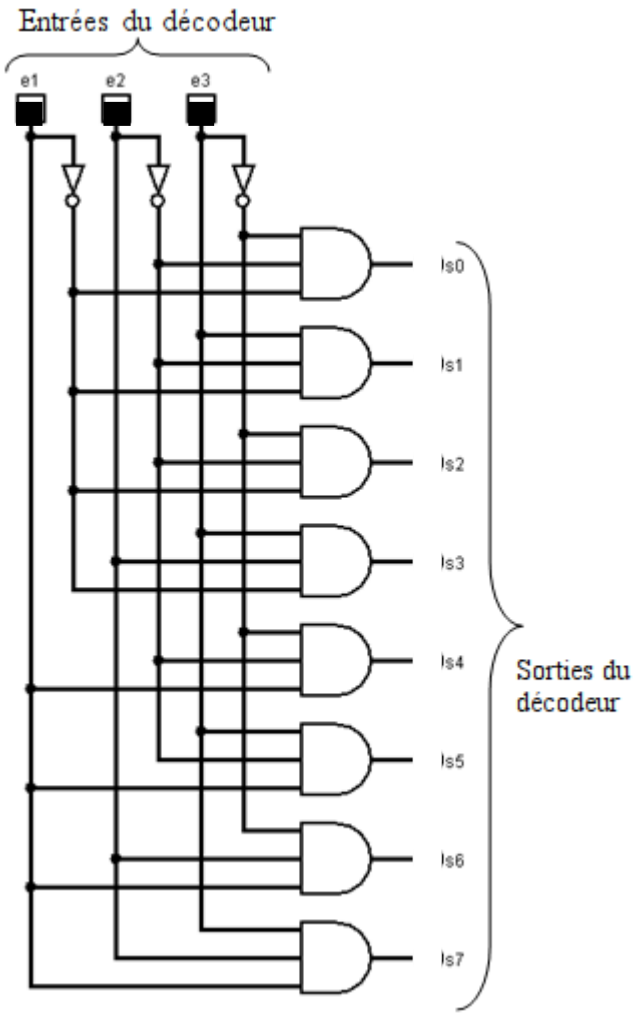
Un décodeur dispose de ***n*** entrées et de 2^n sorties. Il permet d'activer une ligne de sortie (sélection) correspondante à la configuration présentée en entrée. Le schéma suivant illustre un exemple de décodeur à trois entrées:



Décodeur 3:8

Afin de trouver les équations des sorties s_i du décodeur, il suffit de constater que $s_i = 1$ si le code représenté par les variables est ***i*** , c'est à dire que $m_i = 1$. On déduit donc que $s_i = m_i$ pour ***i*** allant de ***0*** à 2^n. Voici par exemple le décodeur 3→8 avec indication de l'équation de chacune de ses sorties.

Il faut noter, par ailleurs, que le plus souvent, une entrée de validation ***E*** (pour dire *Enable*) est utiliser pour valider l'utilisation du décodeur. Ainsi, lorsque cette entrée est à « ***0*** », toutes les sorties si sont à zéro (c'est à dire qu'aucune n'est sélectionnée). Par contre lorsque cette entrée est à « 1 », notre décodeur va fonctionner normalement, c'est à dire la sortie s_i (***i*** allant de ***0*** à 2^n , ***n*** étant le nombre d'entrées) sera à « 1 » (donc sélectionnée) si le code indiquée par les entrées est ***i***.



On aura donc les équations suivantes :

$s_i = E.m_i$, avec ***i*** allant de ***0*** à 2^n.

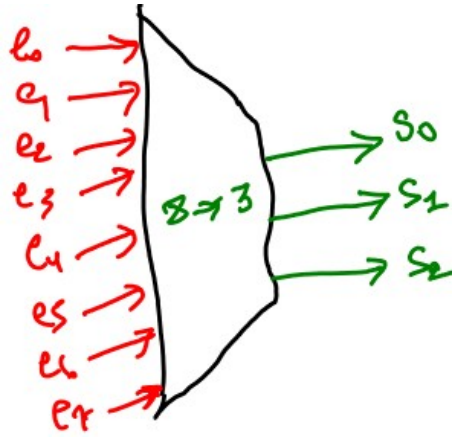
Voici le logigramme de notre décodeur 3→8 :

Codeur

Le codeur (ou encodeur) est un circuit qui possède (en général) ***n*** sorties et $N=2^n$ entrées. Son principe de fonctionnement repose sur le fait que la configuration indiquée par les ***n*** sorites correspond au numéro de la ligne d'entrée à « 1 ». Cela suppose donc qu'à un moment donnée, une seule entrée à « 1 » !

Lorsque plusieurs entrées peuvent être à « 1 » au même temps, on défit un ordre de priorité entre ces entrées de sorte à indiquer en sorite l'entrée prioritaire.

Dans ce qui suit, nous présenterons, à titre d'exemple, un codeur 8→3 (8 entrées et 3 sorties) :



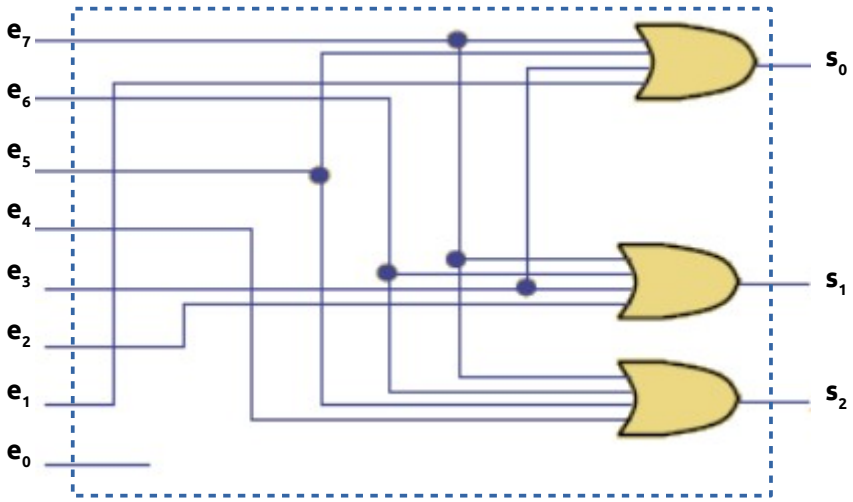
Voici la table de vérité que nous pouvons déduire du principe de fonction de l'encodeur 8→3 :

Entrées	**s_2**	**s_1**	**s_0**	**L'entrée indiqué par les 3 sorites**
e_0	0	0	0	l'entrée 0 (e_0)
e_1	0	0	1	l'entrée 1 (e_1)
e_2	0	1	0	l'entrée 2 (e_2)
e_3	0	1	1	l'entrée 3 (e_3)
e_4	1	0	0	l'entrée 4 (e_4)
e_5	1	0	1	l'entrée 5 (e_5)
e_6	1	1	0	l'entrée 6 (e_6)
e_7	1	1	1	l'entrée 7 (e_7)

Nous constatons ceci :

- ***s_2*** est à « **1** » lorsque ***e_4*** **OU** e_5 **OU** e_6 **OU** ***e_7*** sont à « **1** », ce qui donne $s_2 = e_4+e_5+e_6+e_7$
- ***s_1*** est à « **1** » lorsque ***e_2*** **OU** e_3 **OU** e_6 **OU** ***e_7*** sont à « **1** », ce qui donne $s_1 = e_2+e_3+e_6+e_7$
- ***s_0*** est à « **1** » lorsque ***e_1*** **OU** e_3 **OU** e_5 **OU** ***e_7*** sont à « **1** », ce qui donne $s_0 = e_1+e_3+e_5+e_7$

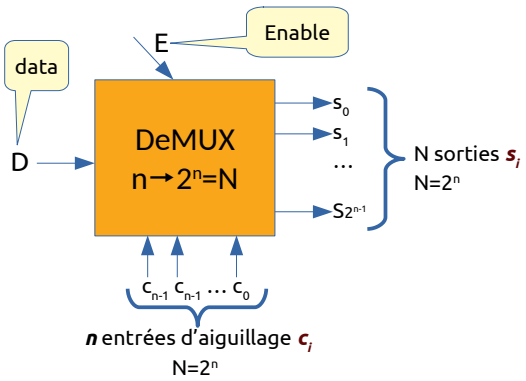
Ce qui nous donne le logigramme suivant :



Démultiplexeur

Un démultiplexeur (**DeMUX**) est un circuit comptant :

- une entrée (de données ou d'information)
- ***N*** sorties (destinations possibles)
- ***n*** entrée de commande (ou de sélection)
- une entrée ***E*** de validation (*Enable*)



Le principe de fonctionnement de ce circuit repose sur l'idée de permettre d'aiguiller (de faire passer) l'entrée ***D*** vers une seule sortie S_i selon le code (ou adresse) indiquée par les entrées de commandes c_i. Ceci fait que les ***n*** entrées de commande (sélection ou aiguillage) doivent être suffisantes pour coder ***N*** valeurs chacune associées aux ***N*** sorties du DeMUX. Ainsi, nous avons cette règle qui doit être respectée : $N=2^n$.

Voici la table de vérité qui découle de ce principe (sans prise en comptes de l'entrée de validation E) :

Entrée de commande (c_{n-1}, …, c_1, c_0)		s_0	s_1	…	s_i	…	S_N N=2^n
m_0	0… 00	D	0		0	…	0
m_1	0…01	0	D		0	…	0
. . .	. . .	0 . . . 0	0 . . . 0		0 . . . 0	…	0 . . . 0
m_i		0	0		D	…	0
. . .	. . .	0 . . . 0	0 . . . 0		0 . . . 0	…	0 . . . 0
m_N	1…11	0	0		0	…	D

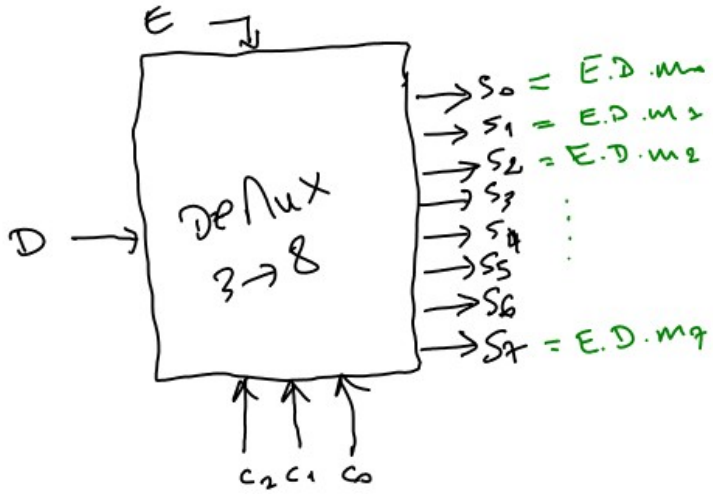
En se référant à la table de vérité ci-dessus, on déduit les équation des sorties :

$s_i = D \cdot m_i$ pour ***i*** allant de ***0*** à $N=2^{n-1}$.

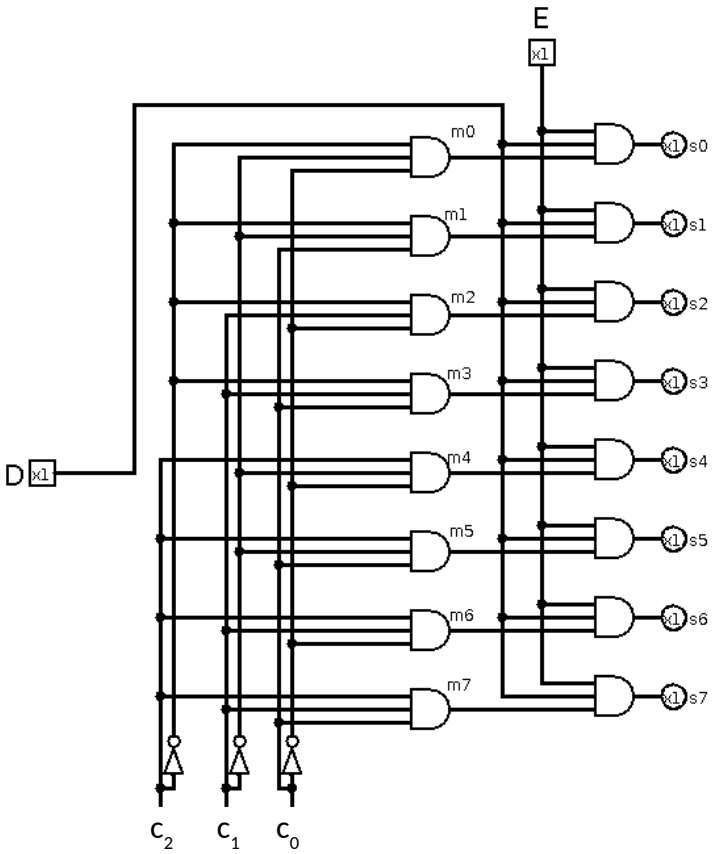
En prenant en compte l'entrée de validation ***E*** qui fait que si elle est à « 0 » aucune sortie ne sera sélectionnée et si elle est à « 1 » la sortie ***i*** sera sélectionnée si le minterme m_i est réalisé alors on aura cette équation :

$s_i = E. D \cdot m_i$

A titre d'exemple, prenons un DeMUX 3→8 :



Voici le logigramme qui en découle :

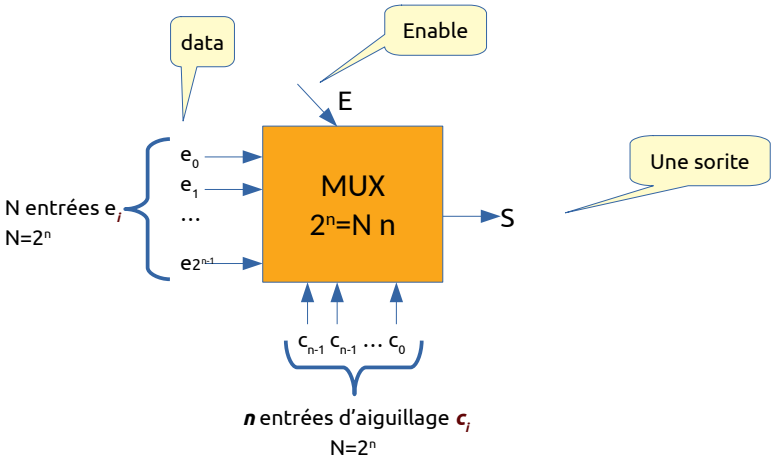


Multiplexeur

Un multiplexeur, réalise l'opération inverse du démultiplexeur. Il dispose de :

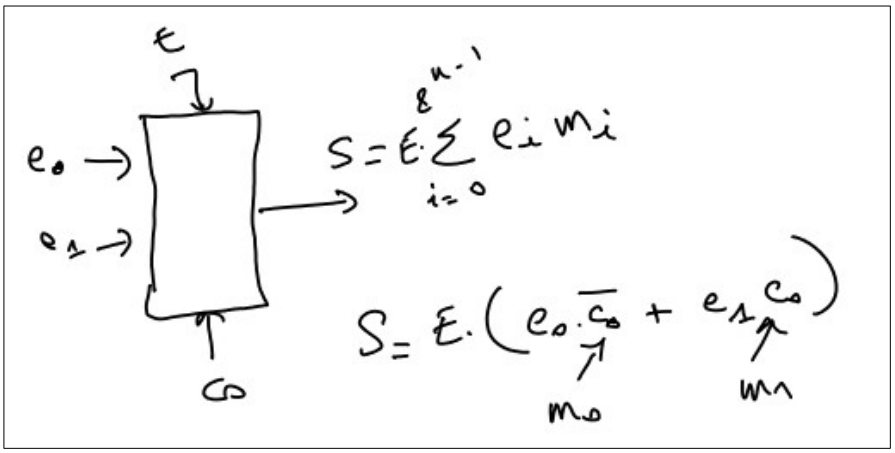
- ***n*** entrées de commande (sélection)
- ***N=2^n^*** entrées de données e_i
- une seule sortie ***S***
- une entrées de validation ***E***

Son principe de fonctionnement repose sur l'idée de sélectionner une entrée e_i parmi les 2^n en se servant des entrées de commandes ($c_{n-1}..c_1c_0$).

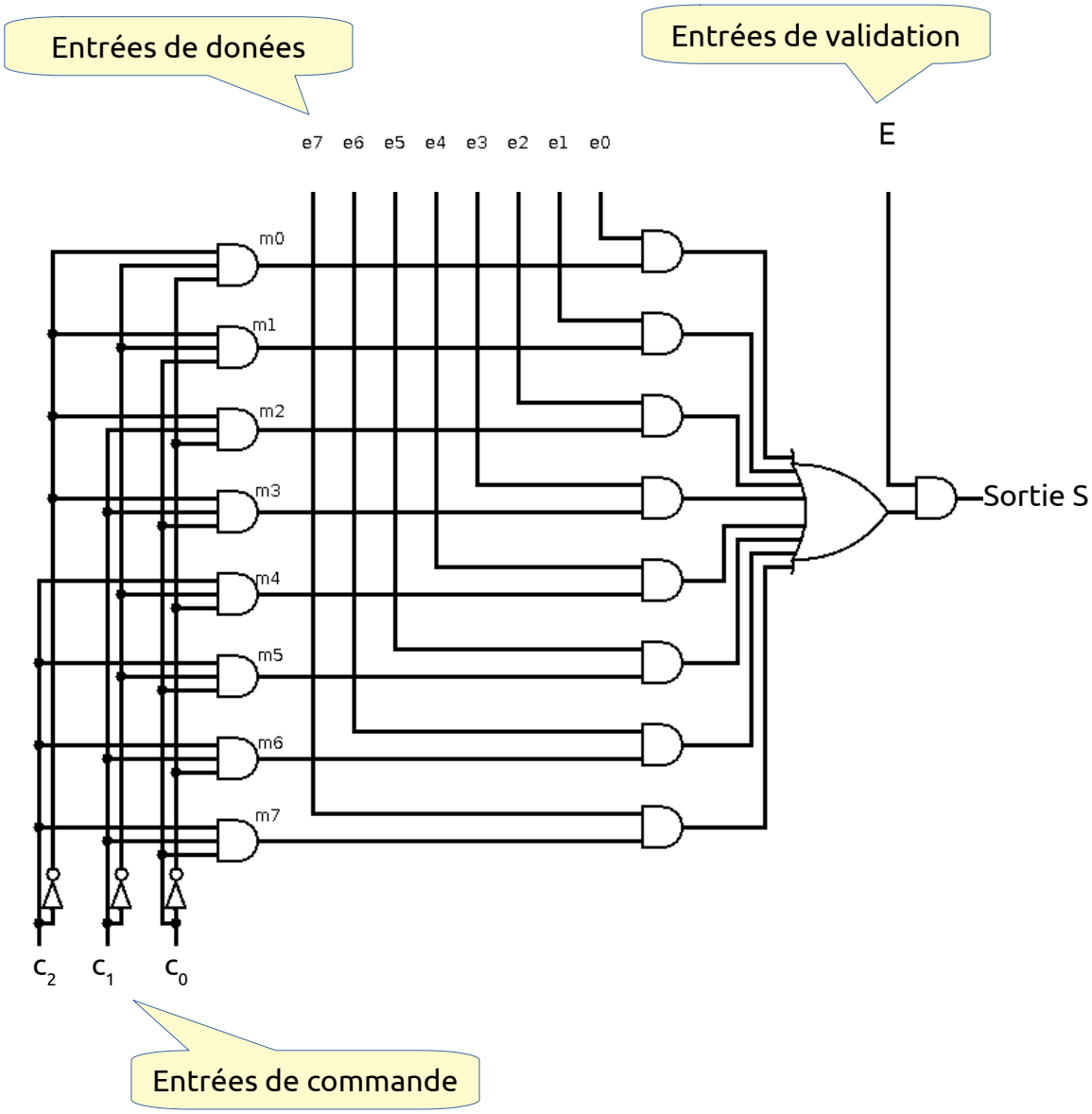


L'équation de la sortie S est données par : $S = E.\sum_{i=0}^{2^{n-1}} e_i.m_i$

Voici un exemple de MUX 2-1 (2 entrées de données et 1 entrées de commande)

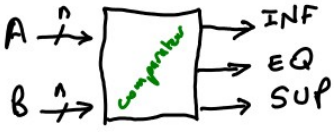


Voici un exemple de MUX 4-2 (4 entrées de données et 2 entrées de commande)



Comparateur

Un comparateur est un circuit logique combinatoire (en général) permettant de comparer entre deux valeurs *A* et *B* (codées, en général, en binaire pure) et possédant généralement 3 sorties *EQ*, *SUP* et *INF* indiquant respectivement si A=B ou A>B ou A<B.



Avant de procéder à la synthèse de ce circuit, il est utile d'étudier les règles (fonctions) de comparaison entre 2 bits ***a*** et ***b***. Nous avons 3 fonctions ***inf***, ***sup*** et ***eq*** qui correspondent respectivement à ***a<b***, ***a>b*** et ***a=b***. Voici leur table de vérité :

a b	Eq = (a=b)	Inf = (a<b)	Sup = (a>b)
0 0	1	0	0
0 1	0	1	0
1 0	0	0	1
1 1	1	0	0
Ce qui donne	$Eq=\overline{a \oplus b}$	$inf=\overline{a}.b$	$sup= a.\overline{b}$

Passant maintenant à la synthèse de notre circuit. Nous pouvons écrire mathématiquement nos nombres comme suit :

- ***A = (a_{n-1} a_{n-2} … a_1 a_0)***
- et ***B = (b_{n-1} b_{n-2} … b_1 b_0)***
- avec a_i et b_i les chiffres des nombres ***A*** et ***B*** pour ***i*** allant de ***0*** à ***n-1***.

A première vue, notre système dispose de **2n** entrées et de **3** sorites. Bien évidement, il est nécessaire de trouver des astuces pour simplifier notre problème et notamment identifier une relation de récurrence.

Pour cela nous constatons 3 choses :

- **Constatation 1** : Pour vérifier si le nombre ***A*** est égale au nombre ***B*** *(fonction **EQ**)* , il suffit de vérifier l'égalité au rang ***n-1***, c'est à dire que (a_{n-1}=b_{n-1}) et l'égalité au rang inférieur (n-2).

  On écrira donc :

  - ***EQ = eq_{n-1} = (a_{n-1}=b_{n-1}) ET eq_{n-2}***
    ce qui donne plus formellement : $EQ=eq_{n-1}=(\overline{a_{n-1} \oplus b_{n-1}}).eq_{n-2}$
  - et au rang i : $eq_i=(\overline{a_i \oplus b_i}).eq_{i-1}$
  - et au rang 0 : $eq_0=(\overline{a_0 \oplus b_0})$ *(je vous rappelle que le non ou exclusif exprime l'égalité !)*
- **Constatation 2** : Pour vérifier si le nombre ***A*** est supérieur au nombre ***B***, il suffit de vérifier si le bit de poids fort de ***A*** (a_{n-1}) est supérieur au bit lui correspondant dans ***B*** (b_{n-1}). 2 possibilités se présentent :
  - Possibilité 1 : Si a_{n-1} est différent de b_{n-1} alors on peut dire que ***A*** est supérieur ou non à ***B***. En effet si $a_{n-1}>b_{n-1}$ alors on peut déduire que ***A>B,*** donc la fonction ***SUP=Sup_{n-1} = (a_{n-1}>b_{n-1}) =*** $(a_{n-1}.\overline{b_{n-1}})$
  - Possibilité 2 : Si $a_{n-1} = b_{n-1}$ alors on ne peut pas dire si A est supérieur (ou inférieur) ou non à B. Il faut vérifier (récursivement) exactement de la même façon les 2 possibilités (1 et 2) pour les bits qui vient juste avant (n-2, n-3 et ainsi de suite). On peut écrire ceci
    - ***SUP=Sup_{n-1} =(a_{n-1}=b_{n-1}) ET sup_{n-2}*** ce qui donne :

      $(\overline{a_{n-1} \oplus b_{n-1}})$. sup_{n-2}
    - *au rang i nous aurons :* $sup_i =$ $(\overline{a_i \oplus b_i})$. sup_{i-1}
    - *au rang 0 :* $sup_0 =$ $(a_0\overline{b_0})$

En résumé : on aura en prenant en compte les 2 possibilités :

pour i allant de n-1 à 1 , on a : ***sup_i =*** $a_i . \overline{b_i} + (\overline{a_i \oplus b_i})$ ***. sup_{i-1}***

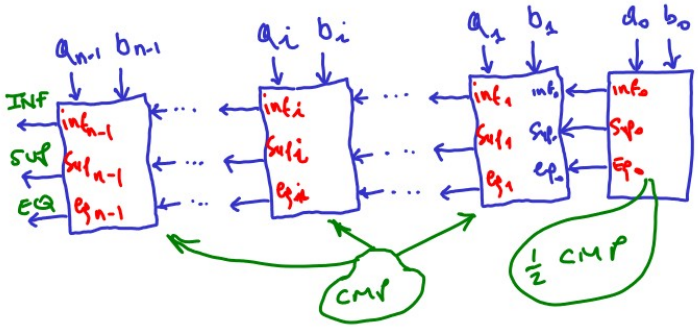
pour i = 0, on a : ***sup_0 =*** $a_0 . \overline{b_0}$

- **Constatation 3** : La fonction INF (A<B) peut être générée comme on a fait pour la fonction SUP. Nous trouverons les équations suivantes :

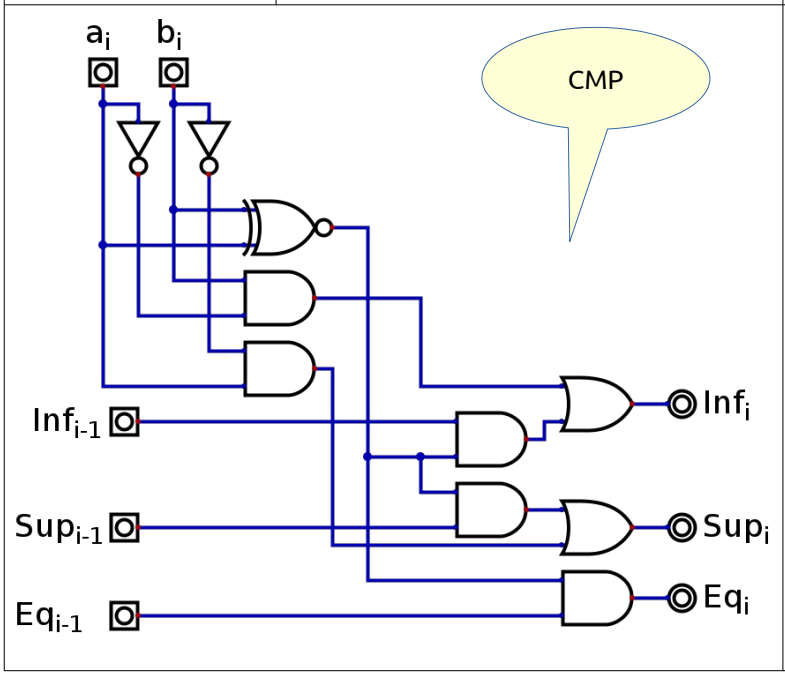
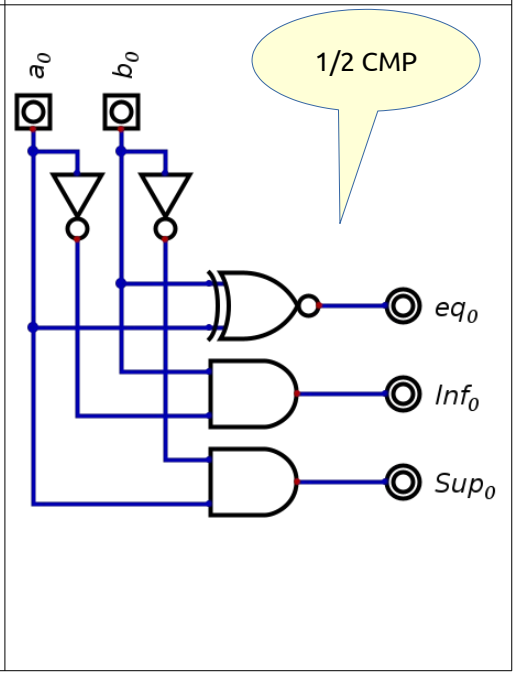
pour i allant de n-1 à 1, on a : ***inf_i =*** $\overline{a_i} . b_i + (\overline{a_i \oplus b_i})$ ***. inf_{i-1}***

pour i = 0, on a : ***inf_0 =*** $\overline{a_0} . b_0$

Au final, nous allons concevoir un demi comparateur pour le première étage (rang 0) et un comparateur complet pour les étages (rangs) i allant de 1 à n-1. Ceci nous donne un montage en cascade comme suit :



Je rappelle les équations :

Fonctions	Étage i (Comparateur complet CMP)	Étage 0 (demi comparateur 1/2 CMP)
A<B : *INF = inf_{n-1}*	*inf_i =* $\overline{a_i} . b_i + (\overline{a_i \oplus b_i})$ *. inf_{i-1}*	*inf_0 =* $\overline{a_0} . b_0$
A>B : *SUP = Sup_{n-1}*	*sup_i =* $a_i . \overline{b_i} + (\overline{a_i \oplus b_i})$ *. sup_{i-1}*	*sup_0 =* $a_0 . \overline{b_0}$
A=B : *EQ= Eq_{n-1}*	$eq_i = (\overline{a_i \oplus b_i}) . eq_{i-1}$	$eq_0 = (\overline{a_0 \oplus b_0})$
		

Génération de fonctions

Il est tout à fait possible de générer des fonction de façon extrêmement simple en se servant de circuits comme le décodeur (DEC) ou le démultiplexeur (DeMUX)

Générer une fonction en se servant d'un décodeur :

Nous savons que toute fonction booléenne peut être écrite sous la forme :

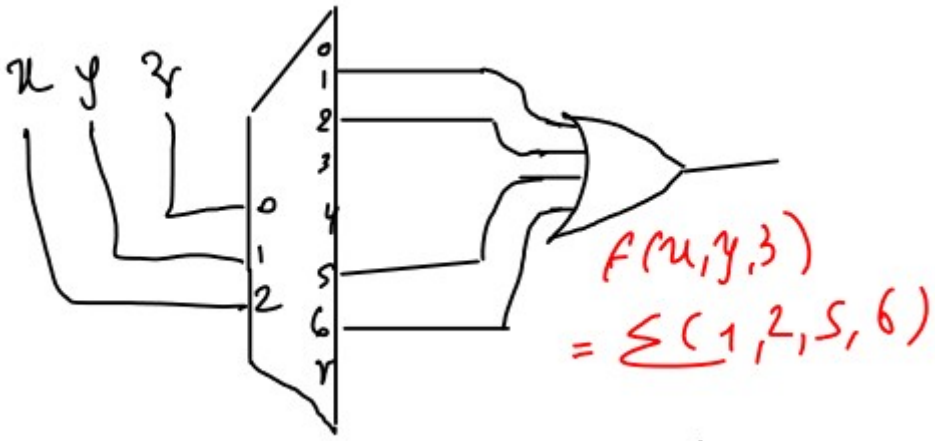
$$f(e_{n-1}, e_{n-2}, \dots, e_0) = \sum_{i=0}^{2^{n-1}} v_i . m_i$$

Nous savons par ailleurs que les sorties d'un décodeur correspondent aux mintermes composés des variables d'entrée (e_{n-1}, e_{n-2}, …, e_0).

En considérant les variables de commandes comme les variables de notre fonction, il suffit de faire un OU Logique entre les sorties si du décodeur qui correspondent aux mintermes pour lesquels la fonction est à « 1 ».

Exemple : $f(x,y,z) = \Sigma(1,2,5,6)$

Nous avons ici 3 variables pour notre fonction ***f***. Nous allons donc nous servir de 3 variables d'entrée du décodeur comme suit : ce qui nous donne le circuit suivant :

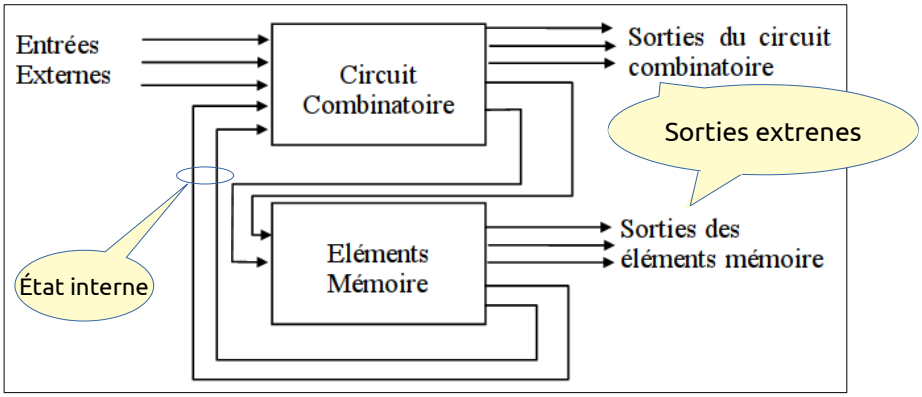


Chapitre 2 : Les circuits logiques séquentiels

La logique combinatoire permet certes de synthétiser des circuits pour réaliser des fonctions comme des additions ou des comparaison, mais reste insuffisants pour concevoir des fonctions comme la mémorisation ou le comptage. Nous allons découvrir, dans ce chapitre, ce que c'est qu'un circuits logiques séquentiel, l'élément fondamental qui est à la base de sa conception (la bascule) et nous présenterons quelques circuits séquentiels : le compteur, le registres et la mémoire.

2.1 Définition

Un circuit séquentiel peut être défini comme un circuit combinatoire englobant des éléments de mémoire. Son schéma général est comme suit:



La partie combinatoire (circuit combinatoire) accepte des signaux binaires en provenance des entrées externes mais aussi en provenance des éléments de mémoire.

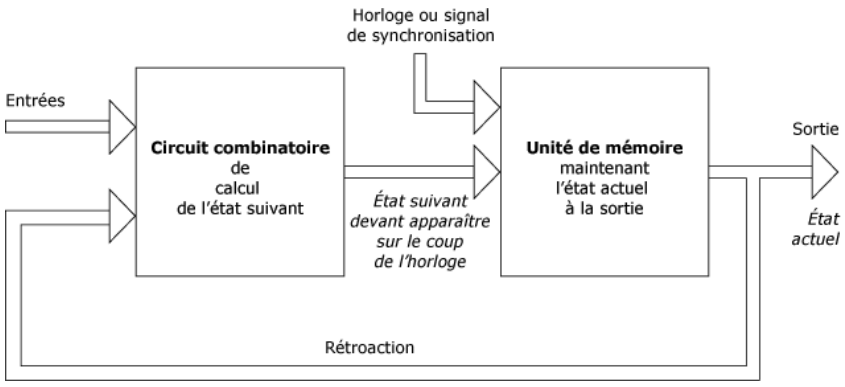
Un élément de mémoire est un dispositif capable d'emmagasiner un bit d'information. L'information mémorisée par les éléments de mémoire du circuit séquentiel peut être modifiée par le circuit combinatoire (ce qui défini l'état « ***présent*** » du circuit). Ceci montre que les ***sorties externes*** du circuit séquentiel sont non seulement fonction des entrées du circuit combinatoire mais aussi de l'état présent des éléments mémoire. Un circuit séquentiel est spécifié par une séquence dans le temps, des entrées, des sorties et des états externes.

Il existe deux types de circuits séquentiels:

- Circuits séquentiel synchrones
- Circuits séquentiel asynchrones

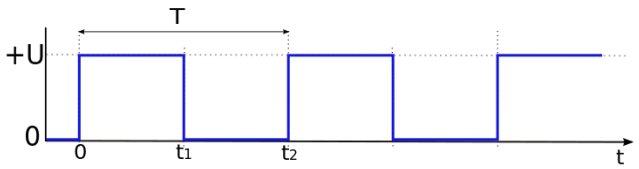
Un circuit séquentiel ***asynchrone*** est caractérisé par le fait que l'allure du changement de ses états de sortie dépend de l'ordre selon lequel les signaux apparaissent en entrée. Autrement dit, seule le changement des entrées externes ou de l'état interne (mémoire / rétroaction) peut provoquer le changement des sorties externes du circuit.

Un circuit séquentiel ***synchrone*** est caractérisé par le fait que l'allure du changement de ses états de sortie, en plus de dépendre des entrées externe et des entrées de rétroaction (mémoire), il est défini en fonction de signaux apparaissant à des ***périodes de temps réguliers***. Ces périodes de temps sont définie par des ***signaux d'horloge*** qui agencent le fonctionnement du circuit logique séquentiel.



Une ***horloge*** (ou signal d'horloge) est un signal électrique oscillant qui rythme les actions d'un circuit. Sa période est appelée ***cycle d'horloge***.

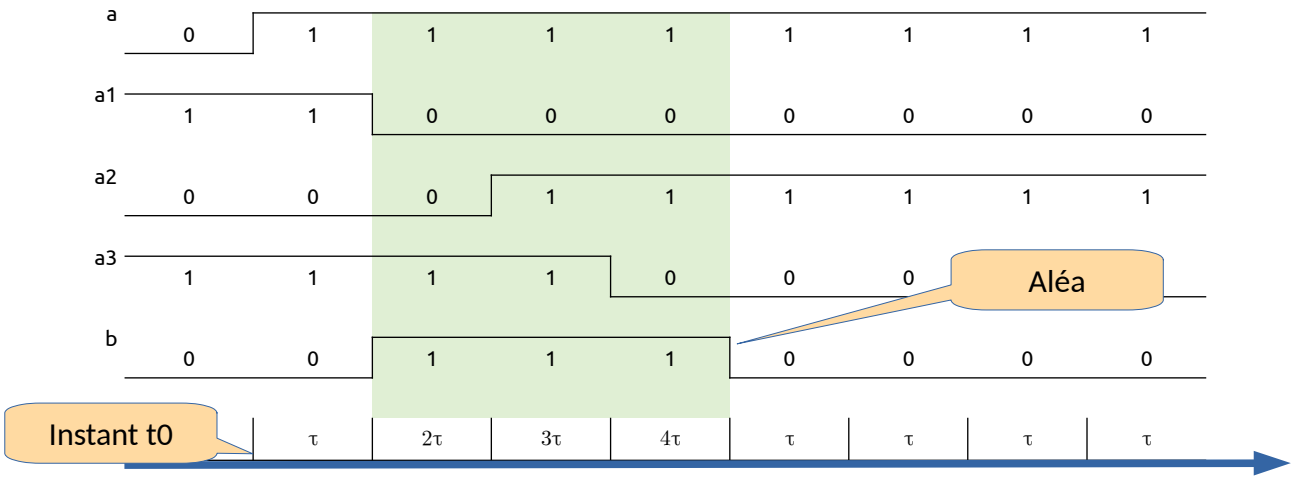
À chaque cycle d'horloge, des calculs peuvent être effectués en utilisant les sorties de bascules. L'horloge permet d'assurer que les données sont valides au cycle d'horloge suivant, c'est-à-dire que les calculs sont terminés et les résultats stabilisés. La durée du cycle doit donc être choisie en fonction de la durée maximale possible de chacun des calculs.



Dans la figure suivante, nous illustrons un signal d'horloge carré ayant une période (ou un cycle d'horloge) égale à T. Son niveau montant dure ***t1*** et son niveau descendant dure ***t2***. Ici notre signal est carré car ***t1***=***t2***. On peut très bien avoir des signaux d'horloge non carré (***t1≠t2***).

Délai de propagation des signaux : La conception des circuits logiques séquentiels prend en compte une propriété que nous avons ignorée dans les circuits combinatoires. Il s'agit du délai de propagation des signaux électriques à travers les portes logiques. En effet, nous avons supposé que dès qu'un changement se produits dans les entrées d'un circuit combinatoire, immédiatement les sorties sont affectées. En réalité c'est faux, chaque porte logiques (selon son procédé de fabrication) mettra toujours un certain temps (dit temps de propagation) pour fait passer un signal électrique depuis son entrée vers sa sortie. Vérifions cela à travers l'exemple suivant :

En principe, en se basant sur ce que nous avions vue pour les circuits logiques combinatoires, la sortie de ce circuit devrait être toujours égale à **0**. Vous allez voir, en réalité qu'il y a un petit ***aléa*** qui se produit lorsque qu'on change la valeur de l'entrée « a ». Supposons que chacune des portes (ici « **NOT** » et « **AND** ») mette un temps égale à τ. Traçons dans un graphique temporelle (que l'on appel chronogramme) l'évolution de nos signaux (a, a1, a2, a3 et b).



On suppose que « **a** » passe de « 0 » vers « 1 » à l'instant t_0.

- Ce n'est qu'à l'instant ($t_0+\tau$) que « a1 » passe à 0 (puis « a1 » est l'inverse de « a »).
- à l'instant ($t_0+2\tau$) que « a2 » passe à 1 (puis « a2 » est l'inverse de « a1 »).
- à l'instant ($t_0+3\tau$) que « a3 » passe à 0 (puis « a3 » est l'inverse de « a2 »).
- Il faut remarquer qu'entre (t_0) et ($t_0+3\tau$) « a » et « a3 » sont tous les 2 à « 1 » au même temps ce qui donne une situations particulière pour « b » égale à « 1 ».

Remarque : Le schéma temporelle indiqué ci-dessus est appelée « **chronogramme** ». Il est souvent utilisé pour montrer les fonctions des circuits logiques séquentiels.

2.2 Les bascules

Une bascule est un circuit logique capable, dans certaines circonstances, de maintenir les valeurs de ses sorties malgré les changements de valeurs d'entrées, c'est-à-dire de mémoriser sont état « mémoire ». Il s'agit de l'élément de mémorisation à la de la logique séquentielle.

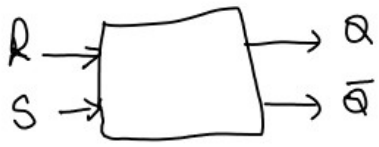
On distingue deux catégories principales de bascules :

- les ***bascules asynchrones*** que l'on nomme ***verrous*** (ou *latch* en anglais)
- et les ***bascules synchrones*** (dépendant d'un signal d'horloge) que l'on nomme simplement bascules (ou *flip-flop* en anglais).

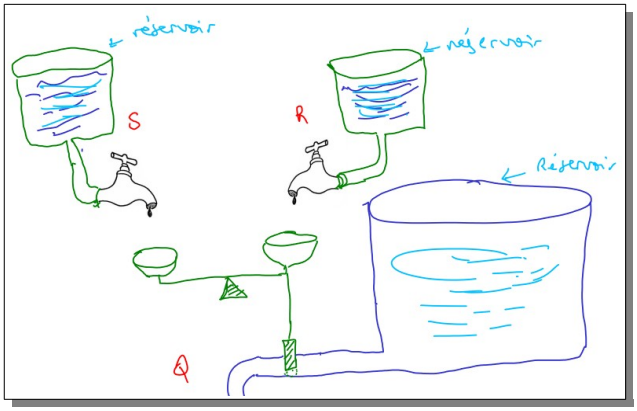
Dans ce qui suit, nous allons d'abord découvrir le principe des bascules en présentant ce que j'appellerai ici la **bascule fondamentale RS**. Vous allez voir que je l'ai appelé ainsi, car elle sera la base de constructions de tous les autres types de bascules (D, T, JK, etc.). Je présenterai aussi les **bascules D et JK** largement utilisées pour réaliser des ***registres*** (groupes de bascules mémorisant un mots d'information) et des ***compteurs***.

Bascule RS

Dans le nom de cette bascules vous remarquez 2 lettres : « **R »** et « **S »**. Ce sont les entrées de cette bascule (« **R »** pour ***reset*** ou mise à zéro et « **S »** pour ***set*** ou mise à « 1 »). Bien évidement, cette bascule va avoir un sortie que l'on appellera **Q**. En réalité, nous aurons toujours 2 sorties : **Q** et $\bar{Q}$.



Je vous propose d'expliquer le principe de fonctionnement d'une bascule RS en me basant l'analogie suivante :



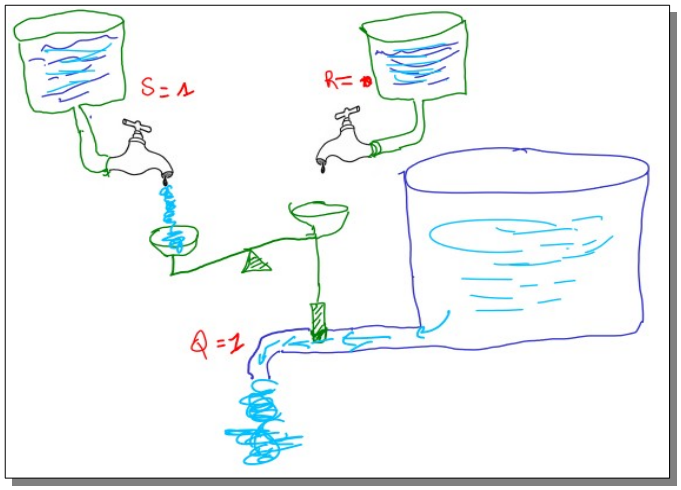
- deux vannes R et S
- une bascules
- une sortie Q qui est un tuyaux
- R = 1 signifie que le robinet R est ouvert (idem pour le robinet S)
- R = 0 signifie que le robinet R est fermé (idem pour le robinet S)
- la sortie Q (tuyau de sortie) est à « 1 » veut dire qu'il fait sortir de l'eau
- la sortie Q (tuyau de sortie) est à « 0 » veut dire qu'il ne fait pas sortir de l'eau
- la partie droite de la bascule est liée à une tige qui actionne une vanne pour fermer ou ouvrir le tuyau de sortie Q.

Analysons le fonctionnement de notre système :

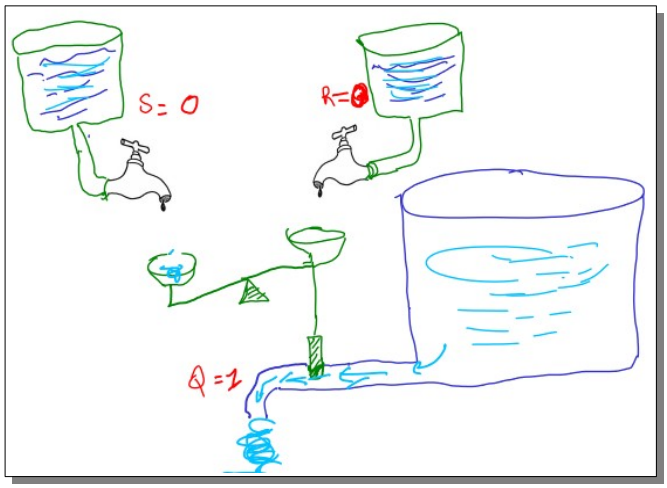
Situation 1 : les deux vannes R et S sont fermées (R=0 et S=0).

- Si la vanne Q était fermée elle restera fermée:
- si elle était ouverte elle restera ouverte
- On dira qualifiera cet état de mémorisation

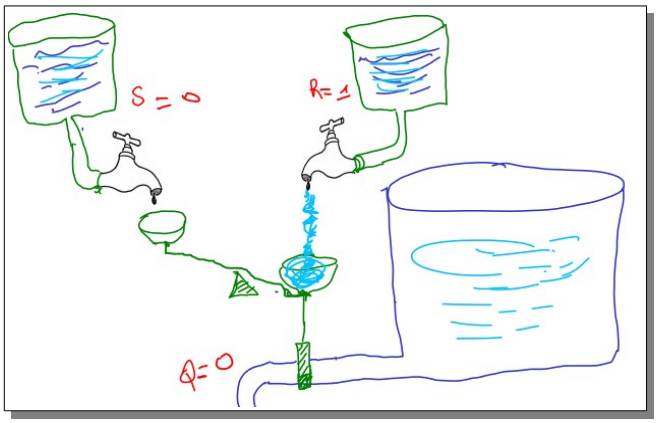
Situation 2 : j'ouvre la vanne « S » en laissant la vanne R fermée (donc S=1 et R=0) : la bascule penche vers la gauche soulevant la vanne Q qui permet à l'eau de couler en sortie. Nous aurons donc Q=1. On dira qu'on a mis à « 1 » notre bascule.



Situation 3 : je ferme la vanne « S » sachant que la vanne R est déjà fermée (donc S=0 et R=0) : la bascule reste penché vers la gauche soulevant la vanne Q qui permet à l'eau de couler en sortie. Nous aurons donc Q=1. On dira qu'on a mémorisé l'état précédent (ici l'état « 1 ») de notre bascule.

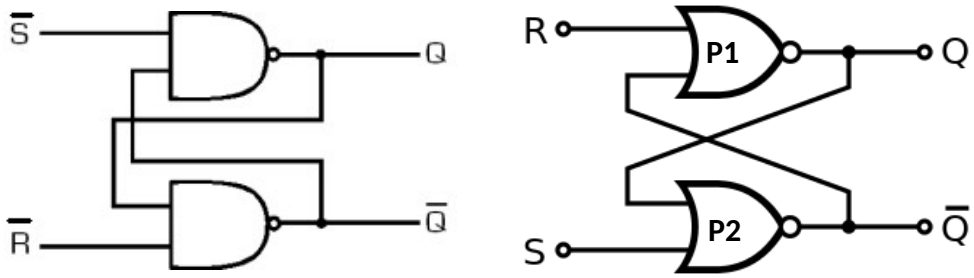


Situation 4 : j'ouvre la vanne « R » sachant que la vanne S est déjà fermée (donc S=0 et R=1) : la bascule va pencher vers la droite fermant la vanne Q qui arrête l'eau de couler en sortie. Nous aurons donc Q=0. On dira qu'on remis à zéro la bascule (pas d'eau en sortie).



Remarque : Il n'est pas du recommandé d'ouvrir au même temps les deux vannes R et S, car on ne saura pas vers quel coté la bascule va se pencher, donc l'état de la vanne Q sera indéterminé !

Voyons maintenant une véritable bascule RS. On peut la construire en se basant sur des portes NANDs ou des portes NORs. La Figure suivante en est une illustration:



Rappelez-vous que le but d'une bascule est de remplir la fonction de mémorisation. Nous utiliserons les 2 entrées R et S pour mettre à « 1 » ou mettre à « 0 » cette bascule. Lorsque R et S sont à « 0 » toutes les deux, la bascule doit mémoriser sont état (précédent). Vous allez voir qu'on va éviter de mettre R et S au même temps à « 1 » (mettre à « 1 » et mettre à « 0 » au même temps n'a pas de sens!). Vérifions ce fonctionnement en analysant le circuit de la base RS à base des portes NOR ci-dessus :

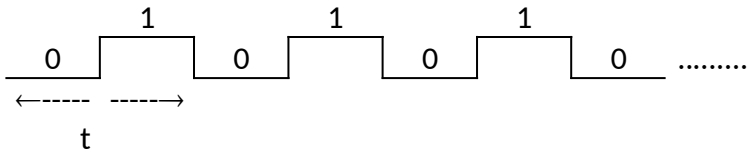
On peut représenter la fonction Q par deux états : Q^+ et Q. Q^+ indiquant l'état future de la bascule et Q indiquant son état présent. Ainsi, nous pourrons écrire $Q^+ = f(R, S\ Q)$. Nous pourront donc représenter la fonction Q de notre bascule par la table caractéristique suivantes :

A compléter... 

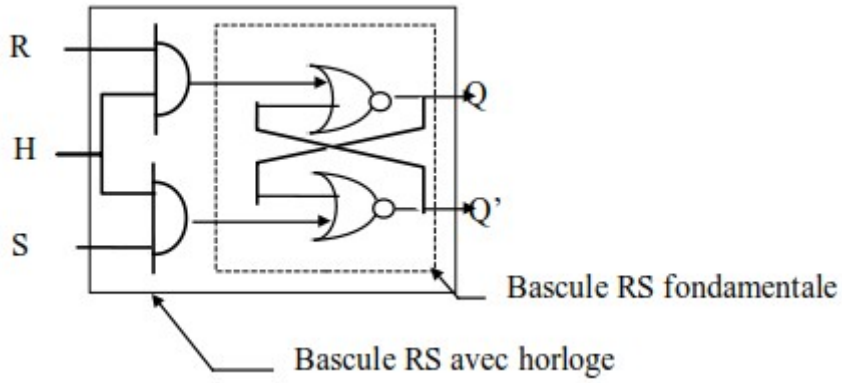
Bascule RSH

Il peut être très intéressant de pouvoir cadencer les changements d'états d'une bascule avec un signal d'horloge.

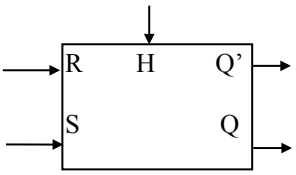
Un signal d'horloge est un signal basculant d'un état à son inverse périodiquement. La caractéristique fondamentale de ce signal est sa fréquence ou sa période. Schématiquement le signal d'horloge est représenté comme suit :



Sachant que l'état mémorisation coïncide avec les entrées R et S à zéro, il suffit de faire un ET logique entre ses entrées et le signal d'horloge, pour imposer à ce que la prise en considération de ses entrées soit rythmée par le signal d'horloge.

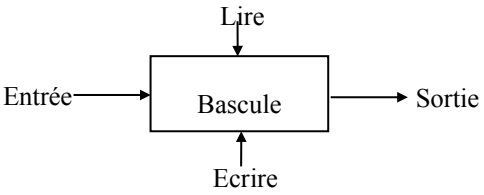


Schématiquement, on représente la bascule RS comme suit:



Autres bascules

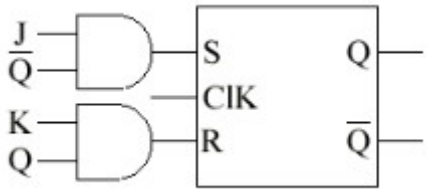
Il existe plusieurs autres types de bascules. On peut citer en particulier, la bascule D, la bascule T et la bascule JK qui sont souvent utilisées pour la construction de circuits logiques séquentiels tel que les compteurs et les registres. En général, on peut dire que la bascule est un circuit logique (constitué de portes logiques) permettant la mémorisation d'une information constituée d'un bit. On peut écrire une information ("0" ou "1") dans cette bascule grâce à un signal d'écriture. Le schéma suivant en est une illustration:



La bascule est l'élément de base permettant la construction de circuits séquentiels (mémoire, registre, compteur, etc.).

Bascule JK

La bascule J-K permet de lever l'ambiguïté qui existe lorsque R=S=1 dans le cas de la bascule RS. Ceci peut être obtenu en asservissant les entrées R et S aux sorties Q et selon le schéma logique indiqué sur la figure suivante.



Nous avons alors pour les signaux R et S : $S = J.Q_{barre}$ et $R = K \,.\, Q$

Ce qui nous permet de construire la table de vérité de la bascule J-K.

Jn	Kn	Q_n	Q_n	S	R	Qn+1
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	1	0	1	0	0	0
0	1	1	0	0	1	0
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	1	0	1	1	0	1
1	0	1	0	0	1	0

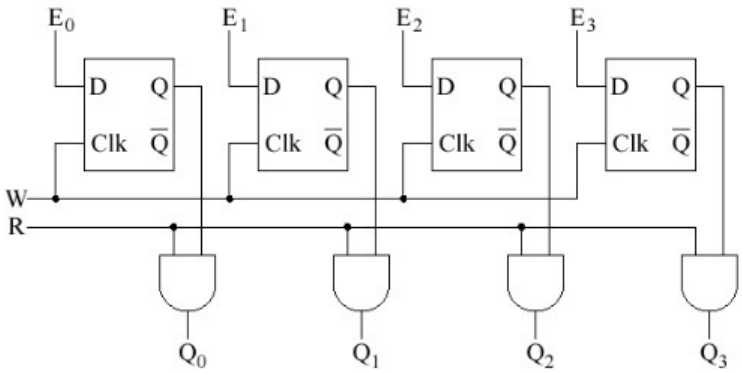
Nous constatons que nous ne rencontrons jamais la combinaison R = S = 1. Cette table peut se résumer sous la forme suivante :

Jn	Kn	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	$\overline{Qn}$

Bascule D

2.3 Les registres

Un registre permet la mémorisation de n bits. Il est donc constitué de n bascules, mémorisant chacune un bit. L'information est emmagasinée sur un signal de commande et ensuite conservée et disponible en lecture. La suivante donne un exemple de registre 4 bits réalisé avec quatre bascules D.

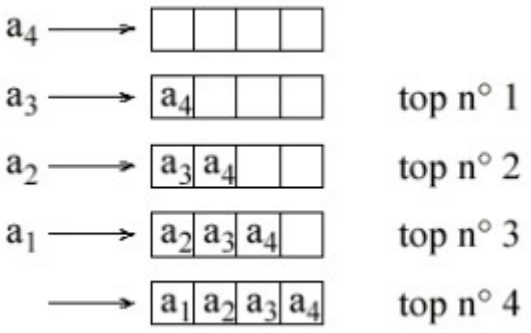


En synchronisme avec le signal d'écriture W le registre mémorise les états des entrées E0 , E1 , E2 et E3 . Ils sont conservés jusqu'au prochain signal de commande W. Dans cet exemple les états mémorisés peuvent être lus sur les sorties Q0 , Q1 , Q2 et Q3 en coïncidence avec un signal de validation R.

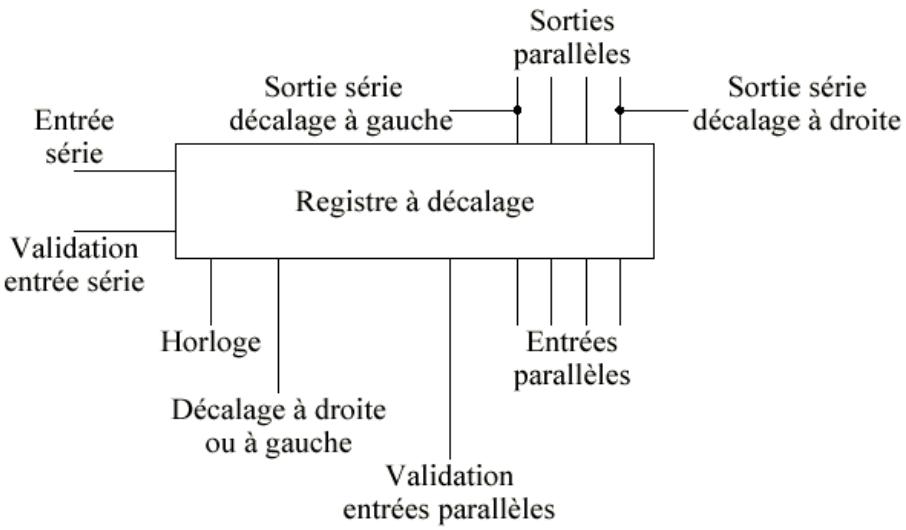
Registre à décalages

Dans un registre à décalage, les bascules sont interconnectées de façon à ce que l'état logique de la bascule de rang i puisse être transmis à la bascule de rang i+1 quand un signal d'horloge est appliqué à l'ensemble des bascules. L'information peut être chargée de deux manières dans ce type de registre.

- Entrée parallèle : comme dans le cas d'un registre de mémorisation. En général une porte d'inhibition est nécessaire pour éviter tout risque de décalage pendant le chargement parallèle.
- Entrée série : l'information est présentée séquentiellement bit après bit à l'entrée de la première bascule. A chaque signal d'horloge un nouveau bit est introduit pendant que ceux déjà mémorisés sont décalés d'un niveau dans le registre. La Figure 54 schématise le chargement d'un registre 4 bits en quatre coups d'horloge.



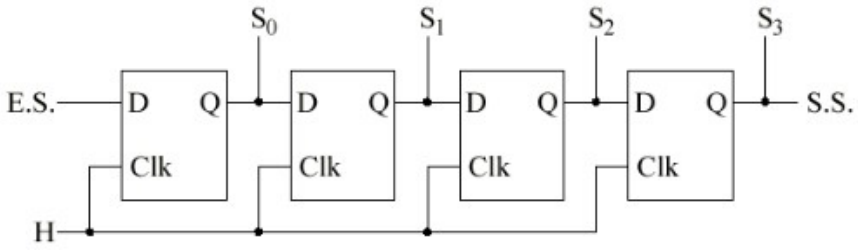
De même l'information peut être lue en série ou en parallèle. D'autre part, certains registres peuvent être capables de décaler à gauche et à droite. Un registre à décalage universel serait donc constitué des entrées, des sorties et des commandes suivantes :



Généralement on utilise des bascules du type maître-esclave D ou R-S.

Entrée série - Sortie parallèle

La figure suivante donne un exemple de registre de 4 bits à entrée série et sortie parallèle réalisé avec des bascules D.

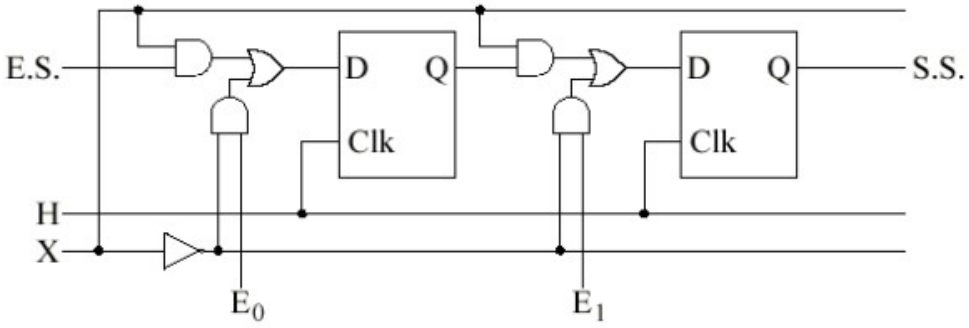


Ce type de registre permet de transformer un codage temporel (succession des bits dans le temps) en un codage spatial (information stockée en mémoire statique).

La sortie série peut également être utilisée. L'intérêt d'utilisation d'un registre à décalage en chargement et lecture série réside dans la possibilité d'avoir des fréquences d'horloge différentes au chargement et à la lecture. Le registre constitue alors un tampon.

Entrée parallèle - sortie série

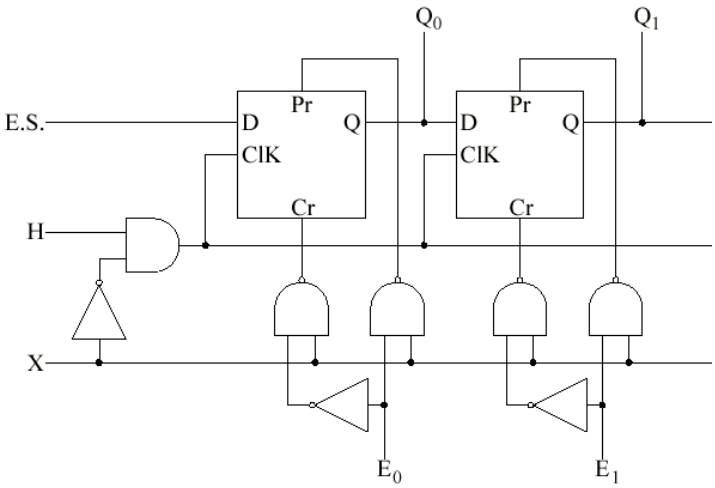
La suivante présente un exemple de registre à décalage à entrée parallèle ou série et sortie série. Si X = 1 l'entrée parallèle est inhibée et l'entrée série est validée. Si X = 0 l'entrée série est bloquée par contre le chargement par l'entrée parallèle est autorisé.



Un registre à décalage à entrée parallèle et sortie série transforme un codage spatial en codage temporel.

Entrée parallèle - Sortie parallèle

La suivante présente un exemple de registre à décalage avec entrées série et parallèle et sorties série et parallèle réalisé avec des bascules de type D.



La commande permet de sélectionner le mode de chargement et d'inhiber le signal d'horloge en cas de chargement parallèle. Si X = 0 nous avons Pr = Cr = 1, ce qui garantit le fonctionnement normal des bascules. Si X = 1 alors selon l'état de chacune des entrées nous avons :

$$\left.\begin{array}{l} Ei = 1 \Rightarrow (Pr = 0, Cr = 1) \Rightarrow Qi = 1 \\ Ei = 0 \Rightarrow (Pr = 1, Cr = 0) \Rightarrow Qi = 0 \end{array}\right\} \Rightarrow Qi = Ei$$

2.4 Les compteurs

Un compteur est un ensemble de n bascules interconnectées par des portes logiques. Il peut donc mémoriser des mots de n bits. Au rythme d'une horloge il peut décrire une séquence déterminée c'est-à-dire occuper une suite d'états binaires. Il ne peut y avoir au maximum que 2^n combinaisons. Ces états restent stables et accessibles entre les impulsions d'horloge. Le nombre total N des combinaisons successives est appelé le modulo du compteur.

On a $N \leq 2^n$. Si $N < 2^n$ un certain nombre d'états ne sont jamais utilisés.

Les compteurs binaires peuvent être classés en deux catégories :

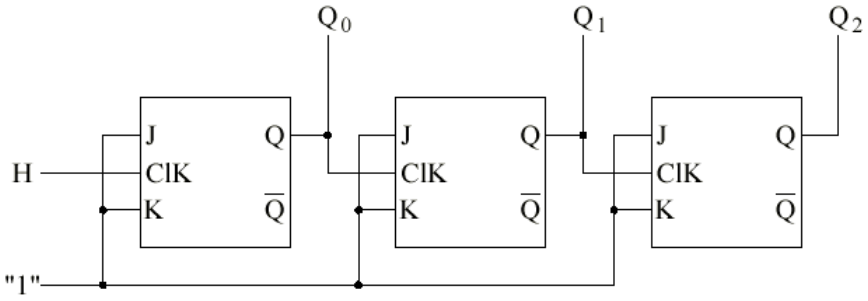
- les compteurs asynchrones;
- les compteurs synchrones.

De plus on distingue les compteurs réversibles ou compteurs-décompteurs.

2.4.1 Compteurs asynchrones

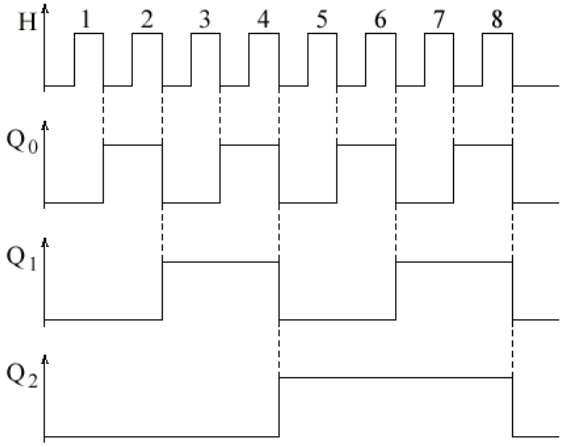
Un compteur asynchrone est constitué de n bascules J-K fonctionnant en mode T (c'est à dire que les entrées J et K sont reliées au même signal). Le signal d'horloge n'est reçu que par le premier étage (bascule LSB : *Least Significant Bit*). Pour chacune des autres bascules le signal d'horloge est fourni par une sortie de la bascule de rang immédiatement inférieur.

Considérons par exemple (figure suivante) un compteur modulo 8 suivant le code binaire pur constitué de trois bascules J-K maîtres-esclaves.



Supposons les trois bascules à zéro à l'instant t = 0. Nous avons vu que pour une bascule maître-esclave la sortie change d'état juste après le passage du signal d'horloge de l'état 1 à l'état 0 (front descendant). L'évolution temporelle des trois sorties Q0 , Q1 et Q2 par rapport aux impulsions d'horloge est représentée sur la suivante. La sortie Q0 bascule sur chaque front descendant du signal d'horloge. La sortie Q1 change d'état à chaque transition 1 vers 0 de la sortie Q0.

De même le basculement de la sortie Q2 est déclenché par une transition 1 vers 0 de la sortie Q1.



A partir de ce chronogramme nous pouvons écrire la liste des états successifs des trois sorties :

Impulsion	Q2	Q1	Q0
État initial	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
....	0	0	0

Nous avons réalisé un compteur s'incrémentant d'une unité à chaque top d'horloge, avec un cycle de huit valeurs de 0 à 7 (modulo 8).

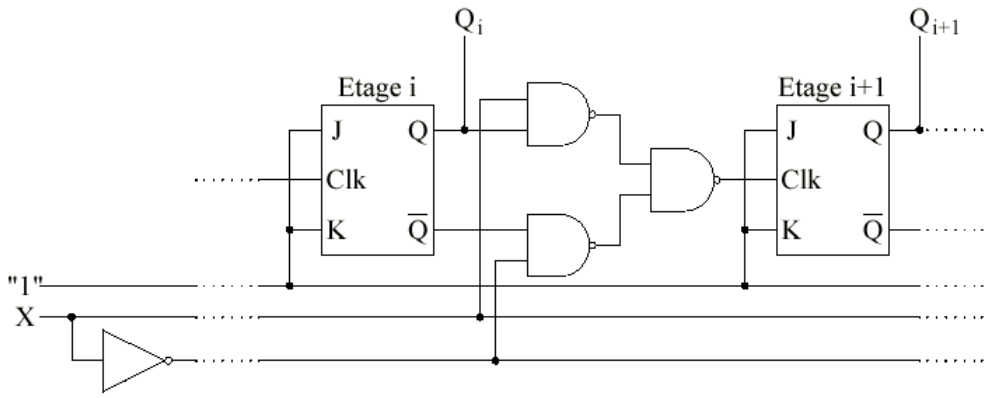
Nous constatons que les sorties Q0, Q1 et Q2 fournissent des signaux périodiques de fréquences respectivement 2, 4 et 8 plus faibles. La division de fréquence est une des applications des compteurs.

2.4.2 Compteur-décompteur asynchrone

Nous obtenons un compteur en déclenchant chaque bascule lorsque celle de rang immédiatement inférieur passe de l'état 1 à 0. Pour réaliser un décompteur il faut que le changement d'état d'une bascule intervienne lorsque la bascule de rang immédiatement inférieur passe de l'état 0 à 1. Pour cela il suffit d'utiliser la sortie Q de chaque bascule pour déclencher la suivante.

On réalise un compteur-décompteur en utilisant un multiplexeur 2 entrées - 1 sortie entre chaque étage pour sélectionner la sortie à utiliser. Pour l'exemple présenté sur la Figure 62, selon l'état de la ligne de commande X nous pouvons sélectionner le mode de comptage :

- X = 1 ⇒ compteur;
- X = 0 ⇒ décompteur.



2.4.3 Compteurs synchrones

Dans un compteur synchrone, toutes les bascules reçoivent, en parallèle, le même signal d'horloge. Pour faire décrire au compteur une séquence déterminée, il faut, à chaque impulsion d'horloge, définir les entrées synchrones J et K. Pour cela, on utilise la table de transition de la bascule J-K (Tableau ci-dessous). Nous avons déjà remarqué que cette table peut se simplifier. En effet, pour chacune des quatre transitions possibles, une seule des entrées J ou K est définie. Rien ne nous interdit donc de les mettre dans le même état, c'est-à-dire J = K, comme dans une bascule T.

Prenons l'exemple d'un compteur synchrone 3 bits, fonctionnant selon le code binaire pur. Nous pouvons dresser un tableau précisant les valeurs des entrées J et K permettant d'obtenir chaque transition (passage d'une ligne à la suivante). Pour qu'une bascule change d'état il faut que ses deux entrées soient à 1.

#Top	Q2	Q1	Q0	J2=K2	J1=K1	J0=K0
0	0	0	0	0	0	1
1	0	0	1	0	1	1
2	0	1	0	0	0	1
3	0	1	1	1	1	1
4	1	0	0	0	0	1
5	1	0	1	0	1	1
6	1	1	0	0	0	1
7	1	1	1	1	1	1
8	0	0	0			

Chaque ligne de cette table correspond à une même tranche de temps. Il est assez facile d'en déduire les expressions logiques reliant les entrées aux sorties :

$$\left\{ \begin{array}{l} J0 = K0 = 1 \\ J1 = K1 = Q0 \\ J2 = K2 = Q0.\ Q1 \end{array} \right\}$$

De manière générale nous pouvons vérifier que les équations de commutation satisfont les relations de récurrence suivantes :

$$\left\{ \begin{array}{l} J0 = K0 = 1 \\ Ji = Ki = Q0.\ Q1\ \ldots.\ Qi\text{-}1 \end{array} \right\}$$

ou encore :

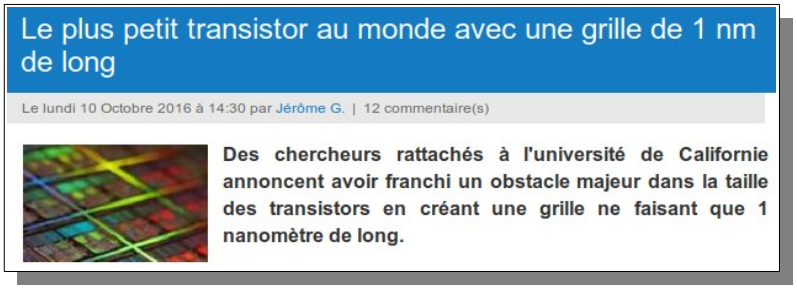
$$\left\{ \begin{array}{l} J0 = K0 = 1 \\ Ji = Ki = Ji\text{-}1.\ Qi\text{-}1 \end{array} \right\}$$

2.5 Les mémoires

A compléter...

2.6 Synthèse d’un circuit logique séquentiel (automates)

A compléter…



Le plus petit transistor au monde avec une grille de 1 nm de long

Le lundi 10 Octobre 2016 à 14:30 par Jérôme G. | 12 commentaire(s)

Des chercheurs rattachés à l'université de Californie annoncent avoir franchi un obstacle majeur dans la taille des transistors en créant une grille ne faisant que 1 nanomètre de long.

2.7 Réalisation d’automates (compteur / décompteur)

A compléter…

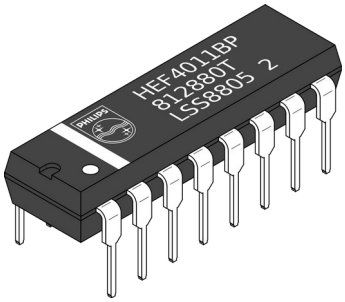
Chapitre 3 : Les circuits intégrés (CI)

3.1 Définition

Voici ce que nous donne Wikipédia comme définition pour un circuit intégré :

> **Circuit intégré**
>
> *Pour les articles homonymes, voir CI et Intégration.*
>
> Le **circuit intégré** (**CI**), aussi appelé **puce électronique**, est un composant électronique, basé sur un semi-conducteur, reproduisant une, ou plusieurs, fonction(s) électronique(s) plus ou moins complexe(s), intégrant souvent plusieurs types de composants électroniques de base dans un volume réduit (sur une petite plaque), rendant le circuit facile à mettre en œuvre[1].
>
> Il existe une très grande variété de ces composants divisés en deux grandes catégories : analogique et numérique.

On peut tirer de cette définition plusieurs concepts :



- puce électronique
- composants électroniques
- semi-conducteur
- fonctions électroniques
- complexité
- intégration de composants électroniques de base
- facilité de mise en œuvre de circuits
- deux variétés de composants électroniques : numériques et analogiques

Puce électronique : Ces juste un nom familier utiliser pour qualifier le circuit intégré

Composants électroniques : On en distingues deux types : les numériques et les analogiques. Les numériques correspondent aux différentes portes et circuits logiques (combinatoires et séquentiels qui sont tous construits à base des transistors. Les composants analogique correspondent aux éléments comme les diodes, les condensateurs, les bobines, les transistors (utilisé en mode analogique) etc.

Semi-conducteurs : Ce nom me rappel la chimie. En fait il s'agit d'un groupe d'éléments chimiques composant la matière et caractérisés par le fait qu'il soit non métallique et qu'il conduit imparfaitement l'électricité. En générale, on utilise le silicium comme base sur laquelle on grave des circuits (transistors notamment) pour réaliser des circuits intégrés.

Fonctions électroniques : C'est ce qui permet de répondre à des besoins réels en utilisant des composants électroniques. Elle peuvent consister en des calculs, des communication, de la mémorisation, de l'affichage, de l'impression, du traitement du signal etc.

Complexité : Se mesure en nombre de transistors intégrés au sein d'une seul puce (substrat de silicium). Elle varie de quelques centaines à des milliards de transistors. Un autre facteur influent sur la complexité est le nombre de liaisons entre les portes logiques et la façon dont est réalisée ces connexion. Par exemple, elles sont plus régulières (donc plus simple) dans le cas d'une mémoire que dans le cas d'un microprocesseur.

Intégration de composants électroniques de base : Il s'agit du processus de rassembler de plusieurs composants aux sein d'une même substrat de silicium (puce). On parle aussi de **miniaturisation** car, on essai au maximum d'intégrer le maximum de composants dans une puce.

Miniaturisation : Sur les substrat de silicium des CI on grave des traits (ligne conductrice de courant). Plus ces traits sont fin, plus on pourra mettre de transistors ! L'industrie est passée d'une largeur de traits de 8 micromètres (µm) en 1970 à 2 µm en 1980, puis à 0,10 µm en 2004. L'intégration des circuits a augmenté très car la surface disponible de silicium a été mieux utilisée et, d'autre part, la taille des puces a crû aussi, dans l'intervalle, de 10 à 200 mm^2 environ. En 2016, on a pu réduire la taille des gravures à 1nm.

À titre de comparaison, un cheveu humain mesure en moyenne 50 000 nanomètres d'épaisseur, ce témoigne bien du progrès technologie dans le domaine de ce domaine. Mais, la miniaturisation a ses limites qui sont principalement liés aux propriétés physico-chimiques de la matières.

En fonction du niveau de miniaturisation des CI, on a définie plusieurs catégories de technologies qui sont résumées dans le tableau ci-contre.

Catégorie	Nombre de portes (n)
SSI : Small Scale Integration (intégration à petite échelle)	$n < 10$
MSI : Medium Scale Integration (intégration à échelle moyenne)	$10 < n < 100$
LSI : Large Scale Integration (intégration à grande échelle)	$100 < n < 1000$
VLSI : Very Large Scale Integration (intégration à très grande échelle)	$n > 1000$

Facilité de mise en œuvre : Les circuits intégrés ont révolutionné les technologies en permettant plus de facilités dans la réalisation des fonctions de différents appareils électroniques. La possibilité d'intégrer des milliers pour ne pas dire des millions de fonctions logiques aux sein d'une seule puce permet de réaliser des circuits électroniques très simples impactant ainsi sur les circuits imprimés.

Circuits intégrés analogiques : Ce sont les plus simples. Il regroupe, en générale, de simples transistors encapsulés les uns à cotés des autres sans liaisons entre eux, ou des assemblages de composants réalisant les fonctions requise par un appareil électronique.

Circuits intégrés numériques : Ils vont des plus simples aux plus complexes. Les plus simples peuvent englober quelques portes logiques de base. Les plus complexes peuvent englober toutes les fonctions booléennes d'un microprocesseur ou d'une mémoire.

3.2 Caractéristiques d'un CI

Les circuits intégrés possèdent plusieurs caractéristiques à prendre en compte lors de la conception d'un dispositif électronique. On distingue les caractéristiques comportementales et les caractéristiques liées aux performances et d'autres liés à la compatibilités entre plusieurs familles de CI.

Caractéristiques comportementales :

- Délai entre changement des entrées et mise à jour de la sortie (délai de propagation)
- taux de sensibilité aux bruits (parasites) peut réduire la fiabilité
- Limites de puissance : Une sortie ne peut fournir de signal à plus de N entrées

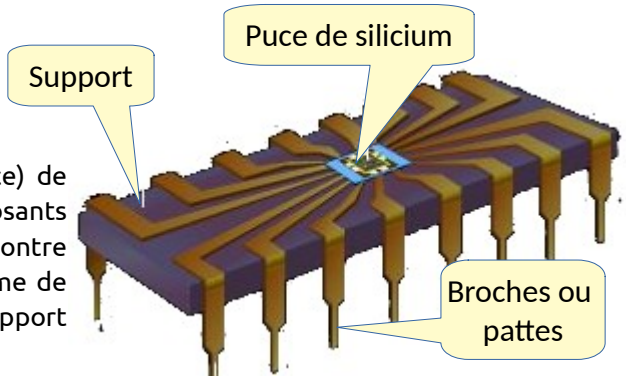
Caractéristiques de performances :

- consommation électrique
- vitesse de propagation des signaux

caractéristiques de compatibilité entre famille de CI :

- Gammes différentes de tension d'alimentation
- Codage incompatible des niveau de tension H (height ou haut) et L (low ou bas)

3.3 Composition d'un CI

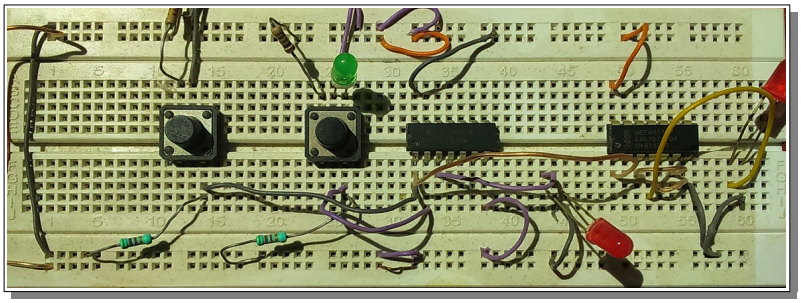


Un CI est généralement composé d'une surface (puce) de silicium sur laquelle sont gravés les composants électroniques (transistors). Cette puce est reliée au montre extérieur par des connecteurs se traduisant sous forme de broches ou de pattes. Le tous est maintenu par un support isolant.

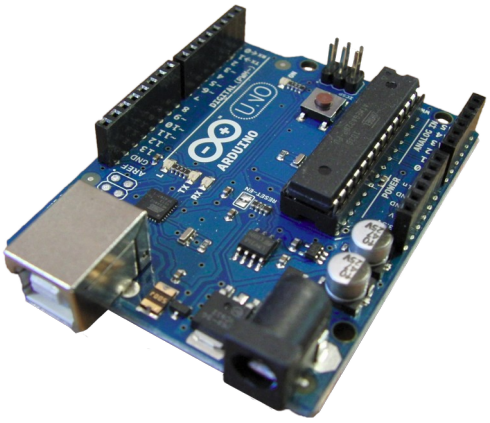
3.4 Montage d'un circuit utilisant des CI

Les montages électroniques se font par la mise en liaisons de plusieurs composants de bases (circuits intégrés ou composants discrets comme les transistors, les diodes, les résistances, les bobines et les capacités) soudés ou implantés sur des supports isolants. Pour maintenir l'ensemble on utiliser soit un circuits imprimé oit une maquette d'essai.

Voici un exemple de montage électronique que j'ai réalisé sur une maquette d'essai :

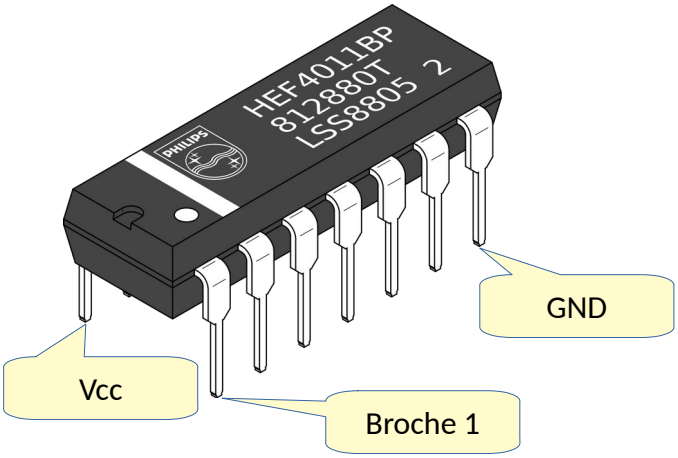


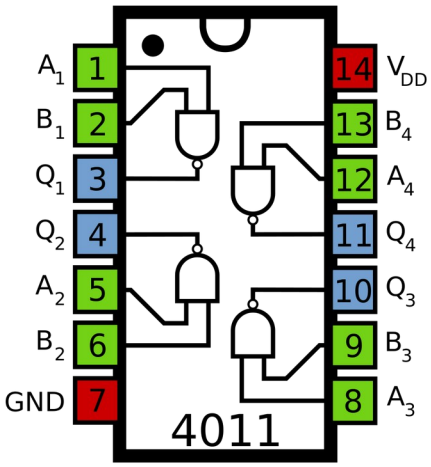
Voici un exemple de montage réalisé à base d'un circuit imprimé :



Exemples de circuits intégrés

Voici un exemple de d'un CI est représentant 4 portes NAND. Il s'agit du HEF 4011. Il est composé de 14 broches. Il faut noter que tous les CI possèdent une broche pour l'alimentation (Vcc ou Vdd) et une broche pour la masse (GND) Comme toutes les broches se ressembles, on utilise un repère (en général un petit creu dans un des 4 bord du CI et un petit rond à coté. Ce petit identifier la broche numéro 1. Ainsi, on pourra identifier toutes les autres broche en se référant au *data-sheet* (descriptif) du CI. Dans le cas du FEF 4011 (voir schéma ci-dessous), on peut facilement identifier les entrées de la première porte NAND (broches 1 et 2) et sa sortie (broche 3). Bien évidement, nous identifions facilement les broche Vdd et GND qui son, en générale sur les bords.





Voici un autre exemple de CI: Le 7486 qui représente 4 portes logiques XOR :