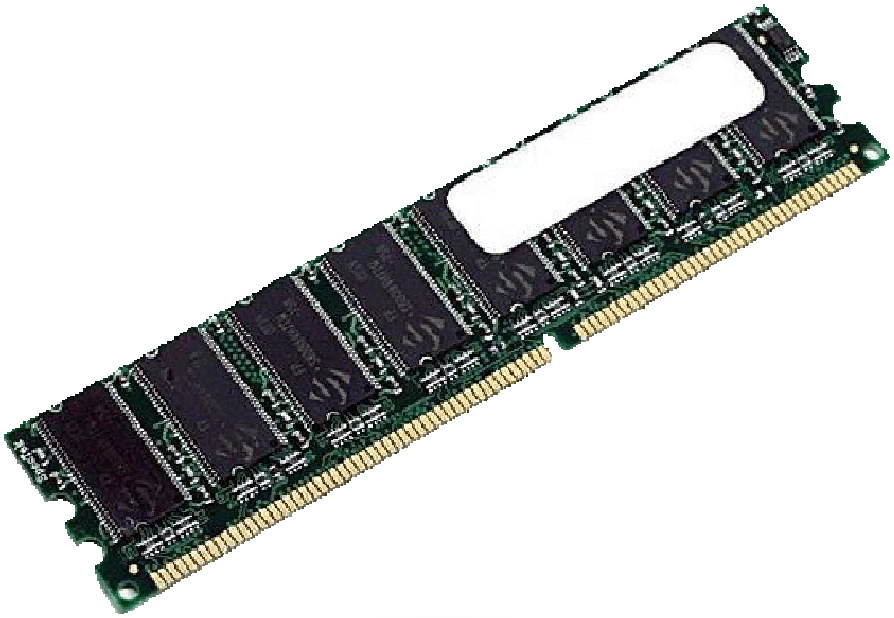


Chapitre III - Circuits Logiques Usuels

Sommaire

Support de cours élaboré par Mr BOUZIDI L'hadi
Mars 2014

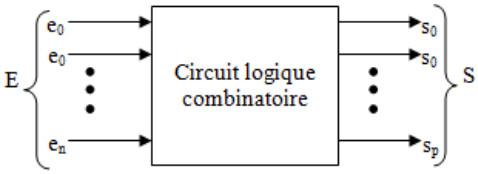
1 – Introduction

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.

On distingue deux types de circuits logiques : les circuits combinatoires et les circuits logiques séquentiels.
Mathématiquement parlant, on peut définir un circuit logique combinatoire 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 sortie alors, on peut écrire : $S = F(E)$ où $s_i = f_i(e_0, e_1, \ldots, e_n)$ avec i allant de 1 à p.

D'un point vue schématique on aura :

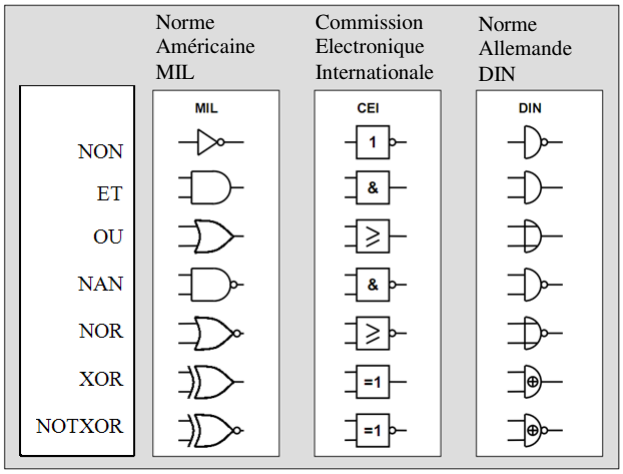


Dans ce qui suit, nous allons définir ce que c'est que la synthèse et l'analyse de circuits logiques combinatoires et nous allons présenter la synthèse que quelques circuits courants.

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 connaitre 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 – 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. Etablir la table de vérité de chacune des fonctions impliquées dans le problème à traiter
2. Etablir les équations logiques ou la table de Karnaugh
3. Simplifier les équations de chacune des fonctions logiques
4. Etablir 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.

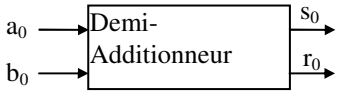
On peut ensuite établir la table de vérité du circuit analysé et opéré à une simplification à 'laide des propriétés de l'algèbre de Boole de la table de Karnaugh.

3 – Synthèse de quelques circuits logiques combinatoires

3.1 – Additionneur

A - Demi-additionneur

1 - Enoncé du problème : Il s'agit de faire la synthèse d'un circuit additionneur simple. Cet additionneur va avoir deux entrées et deux sorties. L'addition va se faire sur deux bits a_0 et b_0 qui vont donner une somme s_0 et une retenue r_0.



2 - Table de vérité : Les règles de l'addition binaire entre deux bits sont définies par la table de vérité suivante:

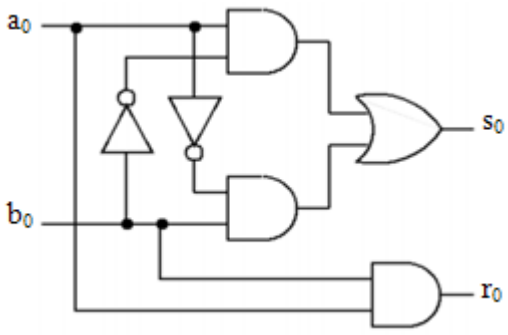
a_0	b_0	s_0	r_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

3 - Equations logiques

$s_0 = a_0 . \overline{b_0} + \overline{a_0} . b_0$

$r_0 = a_0 . b_0$

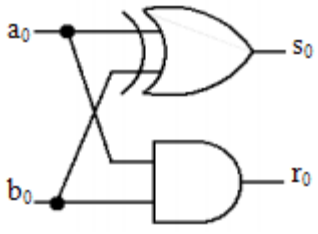
4 - Logigramme :



Il faut noter qu'il existe une fonction de base appelée « OU Exclusif » qui représente exactement la fonction s_0. L'équation va s'écrire, de ce fait, comme suit:

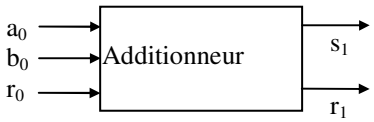
$s_0 = a_0 \oplus b_0$

Le logigramme qui va lui correspondre est:



B - Le circuit additionneur avec prise en considération de la retenue en entrée

1 - Enoncé du problème : L'additionneur que l'on veut concevoir ici va avoir trois entrées et deux sorties. L'addition va se faire sur trois bits a_0 et b_0 et un bit retenu r_0. Ces trois bits vont donner une somme s_1 et une retenue r_1.



2 - Table de vérité : Les règles de l'addition binaire entre deux bits sont définies par la table de vérité suivante:

	a_0	b_0	r_0	s_1	r_1
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

3 - Equations logiques

Considération de la sortie s_1 : Ici le nombre des « 1 » est le même que celui des zéros « 0 » de la colonne de « s_1 ». Considérons de, ce fait, uniquement les « 1 ». Nous obtenons l'équation suivante:

$r_1 = m_1 + m_2 + m_4 + m_7$

$s_1 = \overline{a_0} . \overline{b_0} . r_0 + \overline{a_0} . b_0 . \overline{r_0} + a_0 . \overline{b_0} . \overline{r_0} + a_0 . b_0 . r_0$

Nous pouvant simplifier cette équation en utilisant la fonction logique de base « Ou Exclusif » (noté $\oplus$) et en faisant des manipulations analytiques.

$s_1 = \bar{a}_0 .(\bar{b}_0 .r_0 + b_0 . \bar{r}_0) + a_0.(\bar{b}_0 . \bar{r}_0 + b_0.r_0)$

$s_1 = \bar{a}_0(\bar{b}_0 r_0 + b_0 \bar{r}_0) + a_0(\bar{b}_0 \bar{r}_0 + b_0 r_0)$

$s_1 = \bar{a}_0(b_0 \oplus r_0) + a_0(\overline{b_0 \oplus r_0})$

Posons : $b_0 \oplus r_0 = A$

$s_1 = \bar{a}_0(A) + a_0(\bar{A})$

$s_1 = a_0 \oplus A$

$s_0 = a_0 \oplus (b_0 \oplus r_0)$

Considération de la sortie r_0. : En appliquant la même méthode que pour s_0. L'équation de r_0 sera comme suit:

$r_1 = m_3 + m_5 + m_6 + m_7$

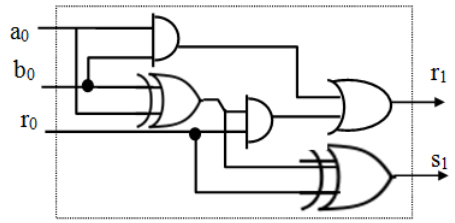
$r_1 = \bar{a}_0 .b_0.r_0 + a_0. \bar{b}_0 .r_0 + a_0.b_0. \bar{r}_0 + a_0.b_0.r_0$

$r_1 = \bar{a}_0 .b_0.r_0 + a_0. \bar{b}_0 .r_0 + a_0.b_0.(\bar{r}_0 + r_0)$

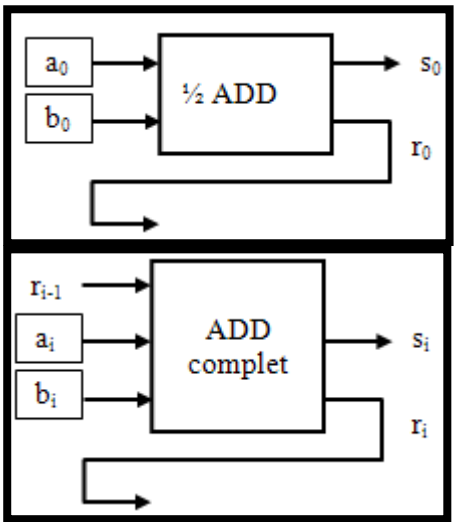
$r_1 = (\bar{a}_0 .b_0. + a_0. \bar{b}_0).r_0 + a_0.b_0$

$r_1 = (a_0 \oplus b_0).r_0 + a_0.b_0$

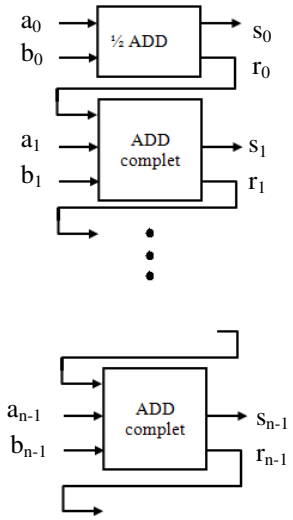
4) Logigramme



D'un point de vue schématique, nous pouvant représente le demi-additionneur et l'additionneur complet comme suit :



Nous pouvant construire avec ces deux circuits des additionneurs de deux mots de *n* bits chacun.



3.2 – 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:

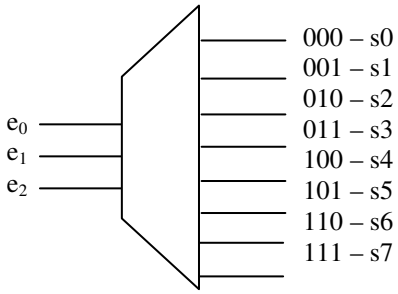


Table de vérité (pour un décodeur 3→8)

	e_1	e_2	e_3	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7		S
$\bar{e_1}\bar{e_2}\bar{e_3}$	0	0	0	1	0	0	0	0	0	0	0	→	s_0
$\bar{e_1}\bar{e_2}e_3$	0	0	1	0	1	0	0	0	0	0	0	→	s_1
$\bar{e_1}e_2\bar{e_3}$	0	1	0	0	0	1	0	0	0	0	0	→	s_2
$\bar{e_1}e_2e_3$	0	1	1	0	0	0	1	0	0	0	0	→	s_3
$e_1\bar{e_2}\bar{e_3}$	1	0	0	0	0	0	0	1	0	0	0	→	s_4
$e_1\bar{e_2}e_3$	1	0	1	0	0	0	0	0	1	0	0	→	s_5
$e_1e_2\bar{e_3}$	1	1	0	0	0	0	0	0	0	1	0	→	s_6
$e_1e_2e_3$	1	1	1	0	0	0	0	0	0	0	1	→	s_7

Equations logiques : Les équations logiques des sorties « s_0, ..., s_7 » sont très simples:

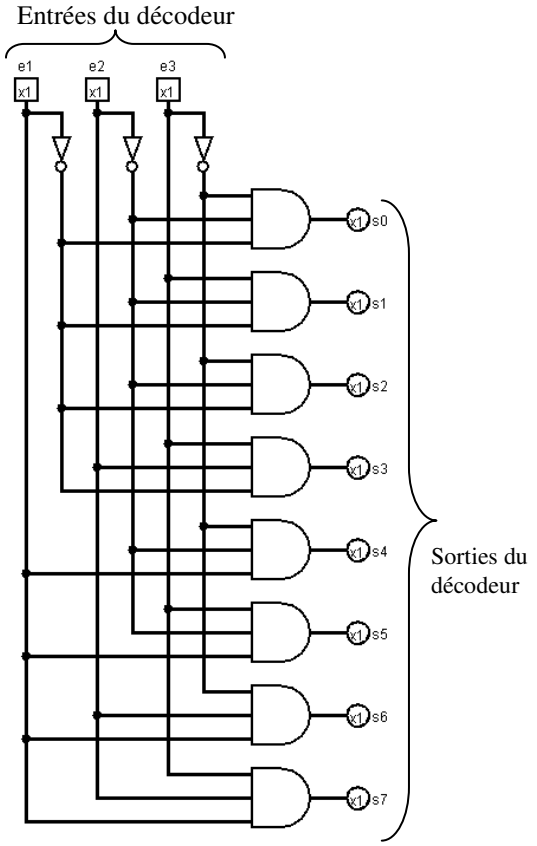
$s_0 = \bar{e_1}\bar{e_2}\bar{e_3}$ $\quad$ $s_1 = \bar{e_1}\bar{e_2}e_3$

$s_2 = \bar{e_1}e_2\bar{e_3}$ $\quad$ $s_3 = \bar{e_1}e_2e_3$

$s_4 = e_1\bar{e_2}\bar{e_3}$ $\quad$ $s_5 = e_1\bar{e_2}e_3$

$s_6 = e_1e_2\bar{e_3}$ $\quad$ $s_7 = e_1e_2e_3$

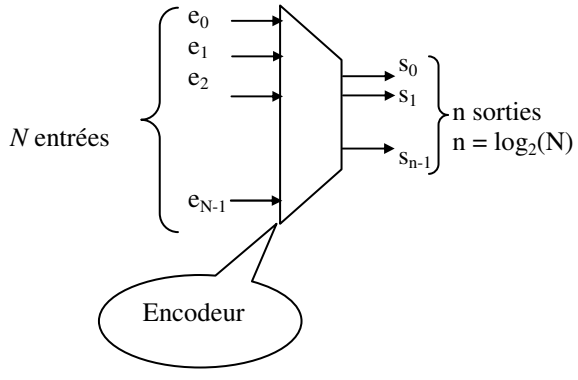
Logigramme :



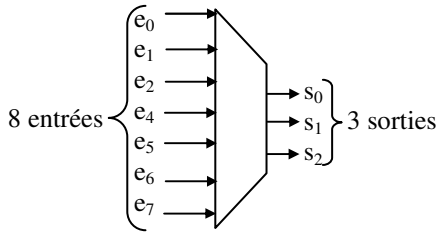
Remarque : Les décodeurs sont souvent munis d'une entrée de validation permettant de prendre en compte les entrées ou non. Dans ce cas, un ET logique avec cette entrée de validation est rajouté pour chaque sortie.

3.3 – Encodage

L'encodeur est un circuit qui réalise la fonction inverse du décodage. Il dispose de $N = 2^n$ entrées et de n sorties. On suppose que seule une entrée peut être à 1 à un moment donné. Toutes les entrées sont numérotées de 0 à *N-1*. A chaque fois qu'une entrée est à 1, on obtient en sortie le code binaire qui correspond au numéro de l'entrée mise à 1.



Exemple : Codeur 8 → 3



Correspondance entre les valeurs des sorties et les numéros des entrées

S2	S1	S0	Numéro
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Equations :

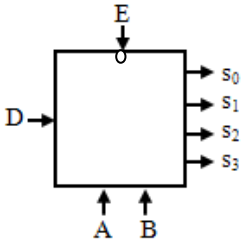
La sortie s_0 est à 1 lorsque le numéro de l'entrée activée est impair.
De ce fait : $s_0 = e_1 + e_3 + e_5 + e_7$

La sortie s_1 est à 1 lorsque le numéro de l'entrée activée est 2,3,6 et 7.
De ce fait : $s_1 = e_2 + e_3 + e_6 + e_7$

La sortie s_2 est à 1 lorsque le numéro de l'entrée activée est 4,5,6 et 7 .
De ce fait : $s_1 = e_4 + e_5 + e_6 + e_7$

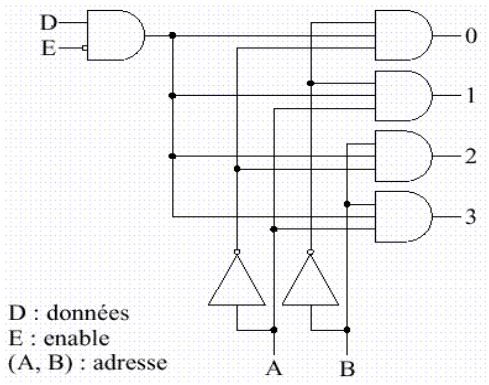
3.4 – Démultiplexeur

Un *démultiplexeur* est un circuit comptant une entrée et *N* sorties et qui met en relation cette entrée avec une sortie et une seule. Pour pouvoir sélectionner cette sortie il faut également des lignes d'adressage (ou de commande) : le code porté par ces lignes identifie la ligne de sortie à utiliser. Ce circuit est très proche d'un décodeur. Considérons un démultiplexeur avec quatre lignes de sortie. Il faut deux lignes d'adresse. Supposons que nous souhaitons également valider les données avec un signal de contrôle E (pour laisser par exemple le temps aux niveaux d'entrée de se stabiliser). Par convention nous choisissons de prendre en compte les données pour E = 0.

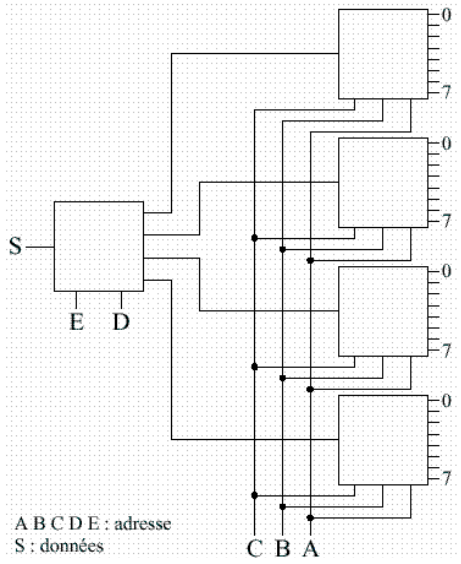


E	B	A	S_0	S_1	S_2	S_3	Produit
0	0	0	D	0	0	0	$\bar{A}.\bar{B}.\bar{C}.D$
0	0	1	0	D	0	0	$\bar{A}.\bar{B}.C.D$
0	1	0	0	0	D	0	$\bar{A}.B.\bar{C}.D$
0	1	1	0	0	0	D	$\bar{A}.B.C.D$
1	0	0	0	0	0	0	
1	0	1	0	0	0	0	
1	1	0	0	0	0	0	
1	1	1	0	0	0	0	

De cette table nous déduisons le logigramme suivant :



Il existe sous forme de circuits intégrés des démultiplexeurs avec 2, 4 ou 16 lignes de sortie. Pour constituer des démultiplexeurs d'ordre supérieur on peut être amené à cascader des démultiplexeurs. Par exemple un démultiplexeur avec 32 sorties peut être réalisé avec un "tronc" de 4 sorties et 4 "branches" de 8 sorties :



3.5 – Multiplexeur

Un multiplexeur, réalise l'opération inverse du démultiplexeur. Il sélectionne une entrée parmi *N* et transmet l'information portée par cette ligne à un seul canal de sortie. Considérons un multiplexeur à quatre entrées, donc deux lignes d'adressage, et une ligne de validation. La table de vérité de ce circuit est donnée par la table 9. De cette table nous déduisons une expression logique pour la sortie :

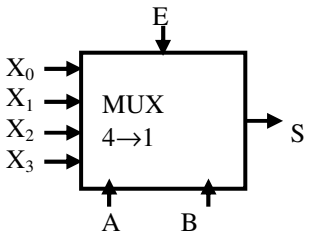
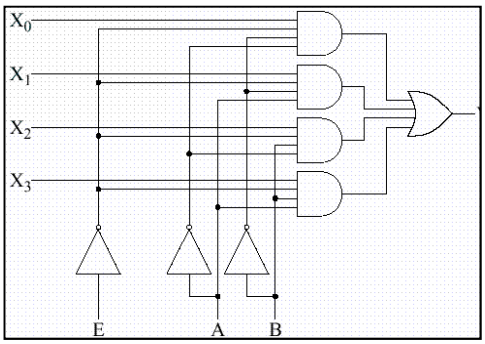


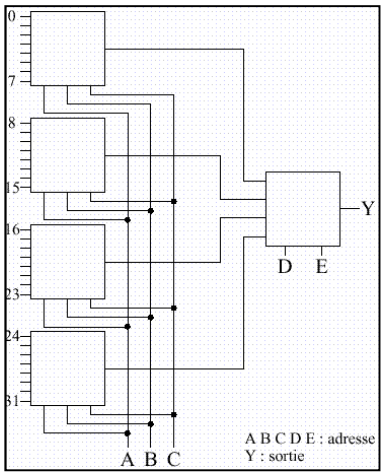
Table de vérité :

E	B	A	S
0	0	0	X_0
0	0	1	X_1
0	1	0	X_2
0	1	1	X_3
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Ce qui nous donne : $S = \bar{E}\bar{B}\bar{A}X_0 + \bar{E}\bar{B}AX_1 + \bar{E}B\bar{A}X_2 + \bar{E}BAX_1$



Tout comme pour les démultiplexeurs on peut cascader plusieurs multiplexeurs pour obtenir un multiplexeur d'ordre supérieur. La figure suivante montre comment un multiplexeur à 32 entrées peut être réalisé à partir de quatre multiplexeurs à 8 entrées et d'un multiplexeur à 4 entrées.

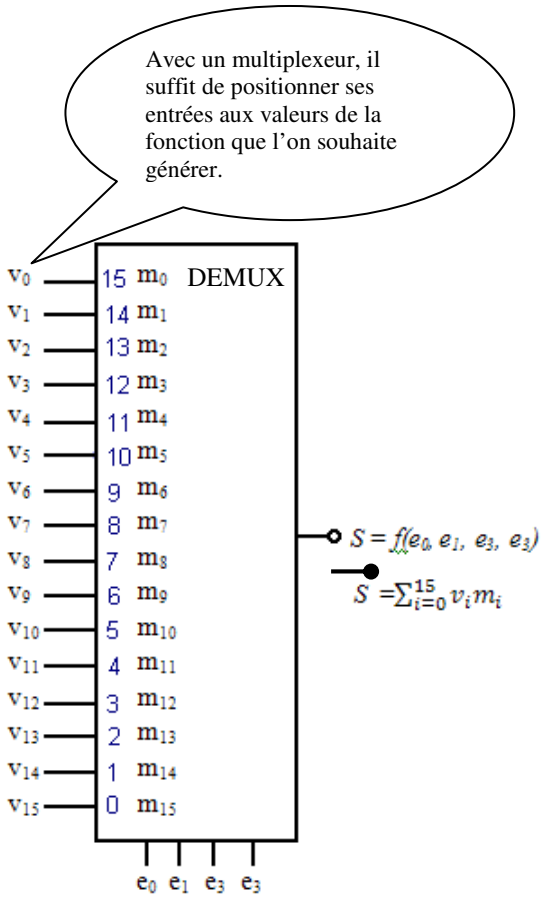


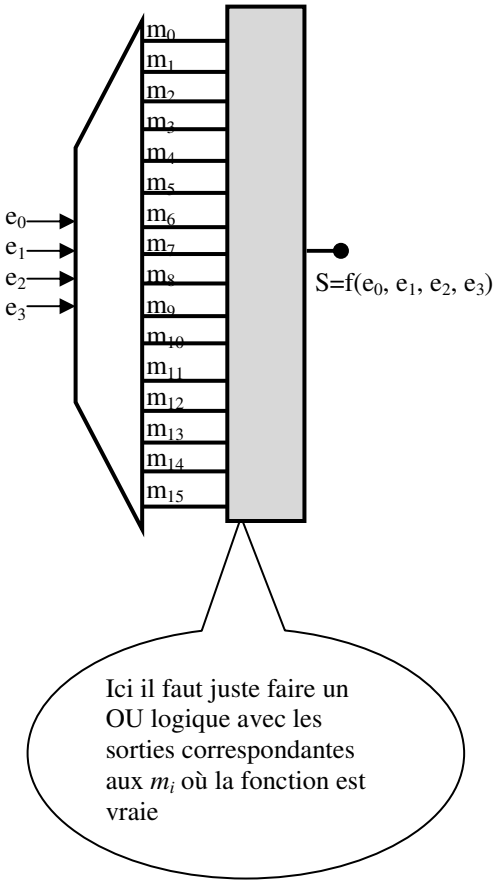
3.6 – Utilisation de décodeurs et de multiplexeurs pour générer des fonctions

Les circuits comme les décodeurs et les multiplexeurs peuvent être utilisés pour générer n'importe quelle fonction logique à l'image des ports NAND et NOR. En effet, toute fonction peut être écrite sous la formule générale suivante :

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

Où :
m_i sont les mintermes et v_i sont les valeurs de la fonction prises pour les mintermes correspondants.

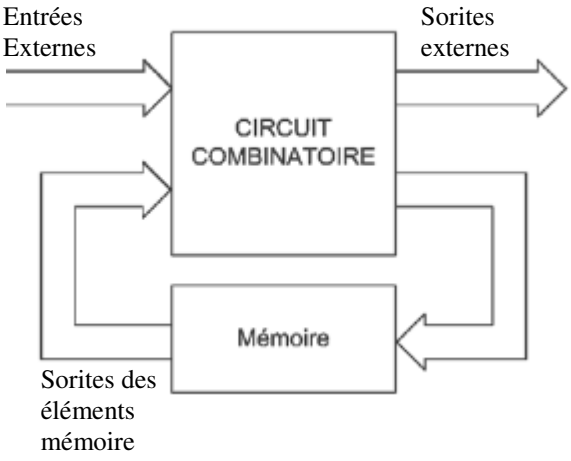




4 - Circuits logiques séquentiels

4.1 Principe

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



Le circuit combinatoire accepte des signaux binaires en provenance des entrées externes et aussi des éléments de mémoire.

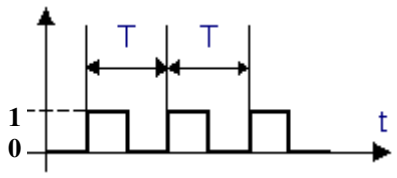
Un élément de mémoire est un dispositif capable d'emmagasiner un bit d'information. L'information mémorisée par les éléments de mémoire du circuit séquentiel peut être modifiée par le circuit combinatoire. Ceci montre que les sorties externes du circuit séquentiel sont non seulement fonction des entrées 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 synchrone est caractérisé par le fait que son état est défini en fonction de signaux apparaissant en des périodes de temps réguliers. Les circuits séquentiels synchrones utilisent, généralement, des horloges qui agencent leur fonctionnement.

⇒ Un signal d'horloge est un signal qui change d'état périodiquement. Il est donc caractérisé par une période (temps mis pour qu'il revienne au même état). Le plus souvent on utilise la fréquence au lieu de la période. La fréquence est l'inverse de la période, elle est mesuré en Hertz.



⇒ 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.

Les unités utilisées dans les circuits séquentiels pour constituer des éléments de mémoire sont appelées ***bascules***.

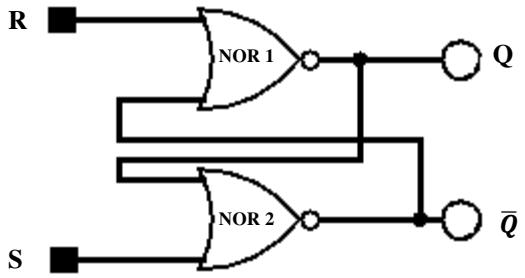
En général, une bascule emmagasine une information correspondant à un bit. Elle comporte des entrées pour agir sur son état et deux sorties, l'une représentant un état, l'autre représentant son complément.

4.2 Bascules

C'est l'élément de base de la construction de circuits logiques séquentiels comme les registres, les mémoires ou les compteurs. C'est un circuit logique capable, dans certaines circonstances, de maintenir les valeurs de ses sorties malgré les changements de valeurs d'entrées.
Nous distinguons les bascules asynchrones (verrous ou *latch*) des bascules synchrones (*flip-flop*).

La bascule fondamentale : La bascule RS (*Reset* pour le R et *Set* pour le S) constitue une bascule fondamentale. C'est de cette bascule que découlent toutes les autres (bascules D, JK et T notamment).

La bascule RS peut être construite à base de deux porte NAND ou NOR comme suit:



Fonctionnement :

Situations	R	S	Q_t	Q_{t+1}	
1	0	0	0	0	mémorisation
2	0	0	1	1	mémorisation
3	0	1	X	1	Mise à 1
4	1	0	X	0	Mise à 0
5	1	1	X	X	Interdit

Explication

Situation 1: Nous avons « R, S » à « 0, 0 » et Q_t à 0. Ceci fait que la porte NOR2 donne un 1 en sa sortie. Puis que cette sortie est réinjectée en entrée de la porte NOR1. Nous aurons donc à l'entrée de cette porte (NOR1) une entrée à 0 (c'est R) et une entrée à 1 (c'est $\bar{Q}_t$). Nous aurons donc $Q_{t+1} = 0$.

Situation 2: Nous avons « R, S » à « 0, 0 » et Q_t à 1. Ceci fait que la porte NOR2 donne un 0 en sa sortie. Puis que cette sortie est réinjectée en entrée de la porte NOR1. Nous aurons donc à l'entrée de cette porte (NOR1) une entrée à 0 (c'est R) et une entrée à 0 (c'est $\bar{Q}_t$). Nous aurons donc $Q_{t+1} = 1$.

Nous voyons que lorsque les entrées R et S sont à 0, la bascule conserve son état précédent. On dit qu'elle est en état de mémorisation.

Situation 3: Nous avons R à 0, S à 1 et Q_t à un état quelconque (X = 0 ou X = 1). Nous avons donc à

l'entrée de la porte NOR2 une entrée à 1 (c'est S) qui va forcé la sortie $\bar{Q}$ à 0. La porte NOR1 aura donc à son entrée 0 et 0, ce qui va forcé sa sortie à 1.

Nous voyons donc que quelque soit la valeur des sorties de la bascule, si on met R à 0 et S à 1 on va mettre la bascule à 1 (Q=1). On parle de mise à 1 de la bascule.

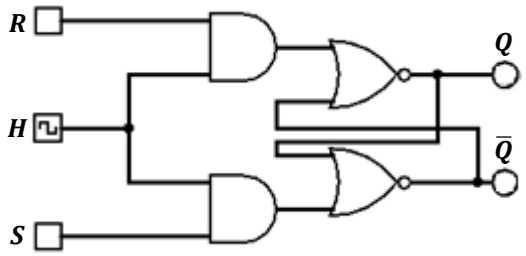
Situation 4: Nous avons R à 1, S à 0 et Q_t à un état quelconque (X = 0 ou X = 1). Nous avons donc à l'entrée de la porte NOR1 une entrée à 1 (c'est R) qui va forcé la sortie Q à 0.

Nous voyons donc que quelque soit la valeur des sorties de la bascule, si on met R à 0 et S à 1 on va mettre la bascule à 1 (Q=1). On parle de mise à 1 de la bascule.

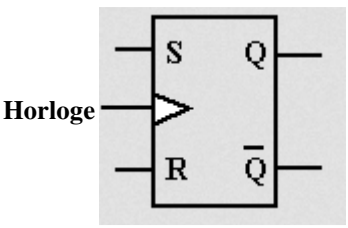
Situation 5: Les deux entrées R et S sont toutes les deux à 1 d'où les deux sorties Q et Q' à zéro. En général, cet état n'est pas utilisé.

Remarque: Le circuit de la bascule fondamentale est asynchrone puisque l'état des sortie change en fonction de rythme de changement des variables d'entrées.

En ajoutant un entrée d'horloge H on peut contrôler les moments où la bascule va changer d'état. Ainsi les entrées R et S de la bascule ci-dessous ne seront prises en compte que si H vaut 1, sinon la bascule sera en état de mémorisation.

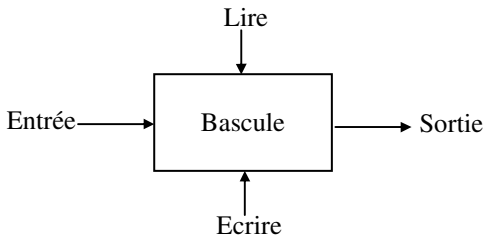


En générale on schématise la bascule sous une forme simplifiée ci-dessous



Les autres bascules : D, T, JK, ...

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:



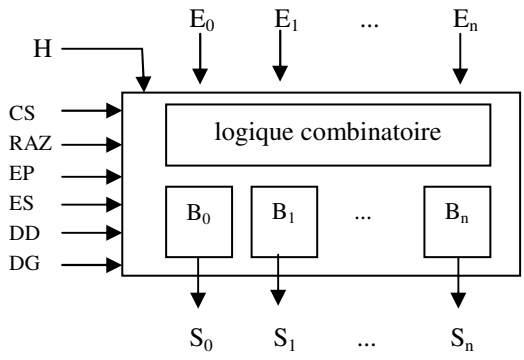
4.3 Circuits séquentiels usuels

On distingue plusieurs type de composants séquentiels construits à partir de bascules RS, D, T, JK, etc. Parmi eux on peut citer:

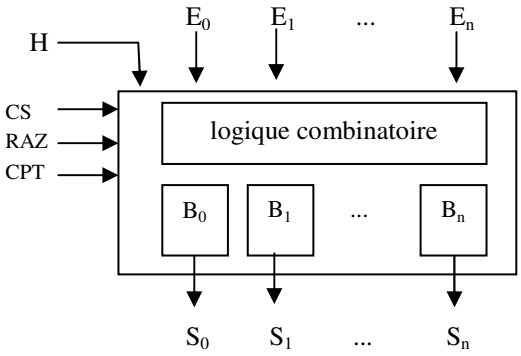
- les registres
- les compteurs
- les mémoires

les registres sont composés d'un ensemble de bascules mémorisant *n* bits d'information. On les utilisent donc pour stocker un nombre limité de bits (8, 16, 32 et 64 bits sont souvent utilisés). Ils peuvent être dotés de plusieurs fonctions (écriture, lecture, entrée série, sortie série, entrée parallèle sortie parallèle, entré de mise à zéro, décalage à gauche ou à droite, etc.).

Exemple de registre



Exemple de compteur *n* bits:

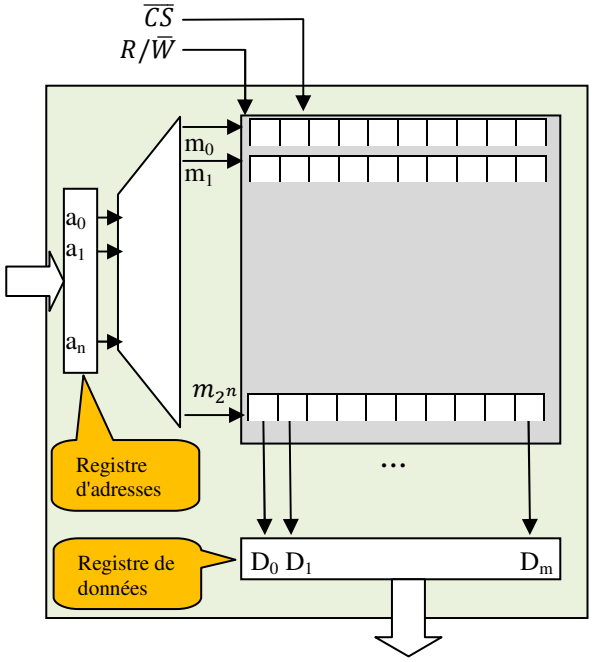


Légende :

- CS : Chip Select (ou sélection du circuit)
- RAZ : Remise à zéro
- CPT : comptage
- EP : entrée parallèle
- ES : Entrée série
- DD : Décalage à droite
- DG : Décalage à gauche
- H : Horloge
- B_i : Bascules
- E_i : Entrées
- S_i : Sorties

Mémoire : Une mémoire peut être considérée comme composée de plusieurs rangés d'éléments de mémorisation (bascules par exemple) et d'une logique permettant de sélectionner des rangés, d'écrire ou de lire des mots d'information. La logique de sélection est en général composé d'un décodeur. Chaque rangé (ou ligne) est appelée un mot mémoire et est associée à une seule adresse mémoire. Cette adresse correspond à une seule sortie du décodeur. Ainsi en introduisant un code (une adresse) à l'entrée du décodeur sur *n* bits, on est capable de sélectionner une seule ligne (une rangée) m_i de la mémoire parmi 2^n possibles. Pour lire ou écrire sur cette ligne on se sert d'une entrée de commande $R/\overline{W}$ (à 0 cette commande indique une lecture, à 1 elle indique une écriture). A l'entrée du décodeur, on peut avoir un registre qui va mémoriser (temporairement) l'adresse du mot mémoire à lire ou à écrire. En sortie de la mémoire (en bas des différentes rangées) on peut prévoir un registre de données permettant de récupérer la données (mot mémoire) lue ou la données à écrire en mémoire.

Voici un exemple de schéma d'une mémoire simple:



Remarque : l'entrée de commande $\overline{CS}$ permet de sélectionner le circuit mémoire. Si cette entrée de commande est à 1 aucune lecture ni écriture ne pourra se faire sur cette mémoire. Si par contre cette entrée de commande est à 0, le circuit est sélectionnée et les autres commandes seront prises en compte (notamment $R/\overline{W}$)