

**Institut Supérieur d'Enseignement
Technologique**

Licence Electromécanique

Cours

d'Automatismes

Partie 1 : Systèmes Combinatoires

Docteur Chighali OULD EHSSEIN

Année scolaire 2019-2020

Table des matières

Introduction.....	3
Chapitre 1 : Systèmes numériques et Information.....	4
Introduction aux Systèmes embarqués	4
Systèmes de numération	5
Le système décimal.....	5
Le système octal	6
Le système binaire	6
Le système hexadécimal	7
Virgule fixe.....	8
Conversion entre systèmes de numérations.....	8
Conversion d'un nombre de base quelconque en nombre décimal	9
Conversion d'un nombre décimal en nombre binaire	9
Conversion entre les nombres binaires et les nombres octaux	9
Conversion entre les nombres binaires et les nombres hexadécimaux.....	10
Codes	10
Code Binaire réfléchi (Gray)	10
Code binaire DCB (Décimal Codé Binaire)	11
Code ASCII	11
Chapitre 2 : Circuits de Logiques Combinatoires.....	14
Portes Logiques	14
Algèbre de Boole	15
Règles générales	15
Théorème de De Morgan.....	16
Universalité des portes NAND et NOR.....	17
Chapitre 3 : Conception de la logique combinatoire	19
Conception des circuits avec la table de vérité et simplification.....	19
Diagramme de Karnaugh.....	20
Codeur.....	22
Décodeur.....	24
Transcodeur	25
Multiplexeur	26
Demultiplexeur	27
Chapitre 4 : Fonctions Arithmétiques.....	29
Nombres signés.....	29
Notation signe/valeur absolue.....	29
Notation en complément à 1	29
Notation en complément à 2	30
Opérations Arithmétiques avec les nombres signés :	31
Additionneurs de base.....	32
Demi additionneur	33
Additionneur complet	33
Additionneur à n bits	34
Additionneur soustracteur à n bits	35
Conclusion	39

Introduction

Ce support de cours est destiné aux étudiants de l'Institut Supérieur d'Enseignement Technologique de Rosso, plus particulièrement aux étudiants de la filière Electromécanique. Il leur permettra d'avoir une vue approfondie de la logique digitale, nécessaire à la compréhension de l'architecture hardware des ordinateurs (Computer Engineering). L'apparition des composants électroniques programmables, tels que les microprocesseurs, a donnée une poussée au développement de l'électronique embarquée et des logiciels (softwares). Ceci a réduit le marché de la logique câblé et a permis le développement de plusieurs langages de programmation.

Les instructions écrites en langage de haut niveau (Java, C++, C, etc.) sont d'abord converties en langage bas niveau (assembleur). Les instructions assembleur sont ensuite converties en langage machine (binaire), seul langage compris par les ordinateurs. D'où alors la nécessité de la maîtrise de la logique digitale pour tous les futurs développeurs.

Ce support est divisé en cinq chapitres :

Le premier chapitre est dédié aux systèmes de numérations et le codage des informations.

Le deuxième chapitre est consacré aux portes logiques et à l'algèbre de Boole.

Le troisième chapitre détaille la conception des circuits logiques combinatoires. Et présente des exemples des fonctions logiques combinatoires telles que le codeur, le décodeur, le transcodeur, le multiplexeur et le démultiplexeur.

Le quatrième chapitre commence par la présentation des nombres signés (codage des nombres décimaux positifs et négatifs en binaire) suivi des opérations arithmétiques avec les nombres signés. Le chapitre présente aussi les fonctions arithmétiques permettant de faire une opération arithmétique addition ou soustraction.

Ce support est complété par un support de TD et un autre support de TP. Les trois supports vous permettront d'acquérir une maîtrise de la logique digitale combinatoire.

Chapitre 1 : Systèmes numériques et Information

Introduction aux Systèmes embarqués

Un système embarqué peut être défini comme un système électronique et informatique autonome, qui est dédié à une tâche bien précise. Ses ressources disponibles sont limitées. Cette limitation est généralement d'ordre spatial (taille limitée) et énergétique (consommation restreinte).

Les systèmes embarqués font très souvent appel à l'informatique, et notamment aux systèmes temps réel.

Le terme de système embarqué désigne aussi bien le matériel que le logiciel utilisé.

Les systèmes embarqués utilisent généralement des microprocesseurs à basse consommation d'énergie ou des microcontrôleurs, dont la partie logicielle est en partie ou entièrement programmée dans le matériel, généralement en mémoire dans une mémoire morte (ROM), EPROM, EEPROM, FLASH, etc. (on parle alors de firmware).

Voici des exemples de systèmes embarqués:

- Les ordinateurs de bord d'une automobile, d'un avion, d'une navette spatiale;
- Les radars, les sonars, les satellites;
- Les téléphones portables, les routeurs, les assistants personnels, les lecteurs mp3;
- Les robots, les automates programmables, les contrôleurs d'usine, de périphériques industriels;
- Les appareils d'imagerie et d'électrophysiologie médicale;
- Les systèmes d'alarmes, les contrôleurs de climatisation, d'ascenseurs, d'accès;
- Les guichets automatiques, les télévisions, les photocopieurs, les caméscopes;

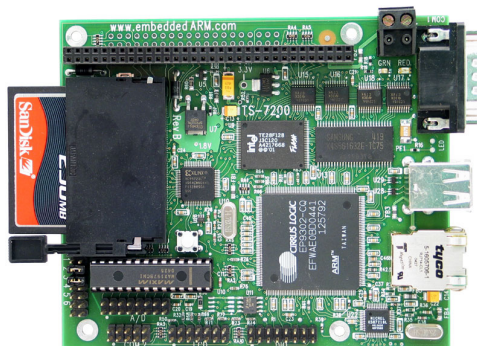


Figure °1 : Exemple d'un système embarqué

Systèmes de numération

Les systèmes de numération binaire et hexadécimale sont très utilisés dans les domaines de l'électronique et de l'informatique. Tout programmeur se doit de les connaître en plus des systèmes décimal et octal.

Principe d'une base

La base est le nombre qui sert à définir un système de numération. La base du système décimal est dix alors que celle du système octal est huit.

Quelque soit la base numérique employée, elle suit la relation suivante :

$$\sum_{i=0}^{i=n} (b_i a^i) = b_n a^n + \dots + b_3 a^3 + b_2 a^2 + b_1 a^1 + b_0 a^0$$

Ou : b_i : chiffre de la base de rang i

Et : a_i : puissance de la base a d'exposant de rang i

Exemple : base 10

$$1986 = (1 \times 10^3) + (9 \times 10^2) + (8 \times 10^1) + (6 \times 10^0)$$

Le système décimal

Le système décimal est celui dans lequel nous avons le plus l'habitude d'écrire. Chaque chiffre peut avoir 10 valeurs différentes :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, de ce fait, le système décimal a pour base 10.

Tout nombre écrit dans le système décimal vérifie la relation suivante :

$$\begin{aligned} 745 &= 7 \times 100 + 4 \times 10 + 5 \times 1 \\ 745 &= 7 \times 10 \times 10 + 4 \times 10 + 5 \times 1 \\ 745 &= 7 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 \end{aligned}$$

Chaque chiffre du nombre est à multiplier par une puissance de 10 : c'est ce que l'on nomme le poids du chiffre.

L'exposant de cette puissance est nul pour le chiffre situé le plus à droite et s'accroît d'une unité pour chaque passage à un chiffre vers la gauche.

$$12435 = 1 \times 10^4 + 2 \times 10^3 + 4 \times 10^2 + 3 \times 10^1 + 5 \times 10^0.$$

Dans notre système conventionnel, nous utilisons les puissances de 10 pour pondérer la valeur des chiffres selon leur position, cependant il est possible d'imaginer d'autres systèmes de nombres ayant comme base un nombre entier différent.

Le système octal

Le système octal utilise un système de numération ayant comme base 8. Il faut noter que dans ce système nous n'aurons plus 10 chiffres mais 8 seulement :

0, 1, 2, 3, 4, 5, 6, 7

Ainsi, un nombre exprimé en base 8 pourra se présenter de la manière suivante :

$(745)_8$

Lorsque l'on écrit un nombre, il faudra bien préciser la base dans laquelle on l'exprime pour lever les éventuelles indéterminations (745 existe aussi en base 10).

Ainsi le nombre sera mis entre parenthèses (745 dans notre exemple) et indicé d'un nombre représentant sa base (8 est mis en indice).

Cette base obéira aux mêmes règles que la base 10, vue précédemment, ainsi on peut décomposer $(745)_8$ de la façon suivante :

$$(745)_8 = 7 \times 8^2 + 4 \times 8^1 + 5 \times 8^0$$

$$(745)_8 = 7 \times 64 + 4 \times 8 + 5 \times 1$$

$$(745)_8 = 448 + 32 + 5$$

Nous venons de voir que :

$$(745)_8 = (485)_{10}$$

Le système binaire

Dans le système binaire, chaque chiffre peut avoir 2 valeurs différentes : 0, 1.

De ce fait, le système a pour base 2.

Tout nombre écrit dans ce système vérifie la relation suivante :

$$(10110)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$(10110)_2 = 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$$

Donc : $(10110)_2 = (22)_{10}$.

Le chiffre dont le poids est le plus fort, bit le plus à gauche, s'appelle le **MSB** (Most Significant Bit) et le chiffre dont le poids est le plus faible, bit le plus à droite, s'appelle le **LSB** (Least Significant Bit).

Tous les systèmes de numération de position obéissent aux règles que nous venons de voir.

Ci-dessous un tableau récapitulatif, montrant les équivalences entre les systèmes décimal, octal et binaire.

Système décimal	Système octal	Système binaire
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111
16	20	10000
17	21	10001
-	-	-
-	-	-
63	77	111111
64	100	1000000
65	101	1000001

Tableau récapitulatif

Le système hexadécimal

Le système hexadécimal utilise les 16 symboles (chiffres et lettres) suivants :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

De ce fait, le système a pour base 16.

Un nombre exprimé en base 16 pourra se présenter de la manière suivante :

$$(5AF)_{16}$$

Le nombre $(5AF)_{16}$ peut se décomposer comme suit :

$$(5AF)_{16} = 5 \times 16^2 + A \times 16^1 + F \times 16^0$$

En remplaçant A et F par leur équivalent en base 10, on obtient :

$$\begin{aligned}(5AF)_{16} &= 5 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\ (5AF)_{16} &= 5 \times 256 + 10 \times 16 + 15 \times 1\end{aligned}$$

$$\text{Donc } (5AF)_{16} = (1455)_{10}$$

La correspondance entre le système binaire (base 2), le système décimal (base 10) et le système hexadécimal (base 16) est indiquée dans le tableau ci-après :

Système décimal	Système hexadécimal	Système binaire
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
-	-	-
-	-	-
63	3F	111111
64	40	1000000
65	41	1000001

Tableau récapitulatif

Virgule fixe

Les bits à gauche de la virgule représentent la partie entière du nombre (au vrai sens du terme), c'est-à-dire l'entier se trouvant à gauche de la virgule. Chaque bit à droite de la virgule correspond à l'inverse d'une puissance de 2. Ainsi la première décimale binaire est $\frac{1}{2}$, la seconde est $\frac{1}{4}$, la troisième est $\frac{1}{8}$ et ainsi de suite.

Exemple

$$\begin{aligned}
 (11001,110)_2 &= 1x2^4 + 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2} + 0x2^{-3} \\
 (11001,110)_2 &= 1x16 + 1x8 + 0x4 + 0x2 + 1x1 + 1x0,5 + 1x0,25 + 0x0,125 \\
 (11001,110)_2 &= 16 + 8 + 0 + 0 + 1 + 0,5 + 0,25 + 0 = (25,75)_{10}
 \end{aligned}$$

Soit $(11001,110)_2 = (25,75)_{10}$

Conversion entre systèmes de numérations

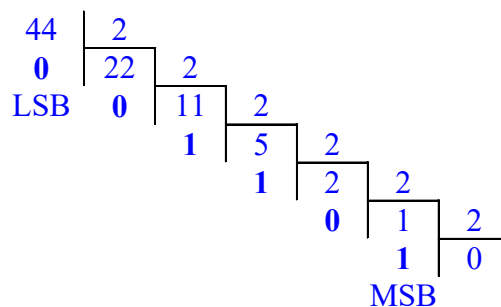
Conversion d'un nombre de base quelconque en nombre décimal

En exposant les principes des systèmes de numération de position, nous avons déjà vu comment convertir les nombres de base 8, base 2 et base 16 en nombres décimaux.

Conversion d'un nombre décimal en nombre binaire

Pour obtenir l'expression binaire d'un nombre exprimé en décimal, il suffit de diviser successivement ce nombre par 2 jusqu'à ce que le quotient obtenu soit égal à 0.

Exemple conversion de $(44)_{10}$ en binaire :



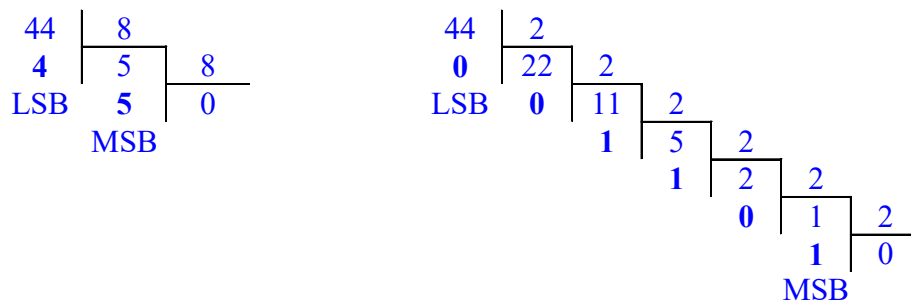
Les restes de ces divisions lus de bas en haut représentent le nombre binaire.

$$\text{Donc } (44)_{10} = (101100)_2$$

De la même manière pour convertir un nombre décimal en octal ou en hexadécimal, on divise successivement par 8 et 16 respectivement jusqu'à ce que le quotient obtenu soit égal à 0.

Conversion entre les nombres binaires et les nombres octaux

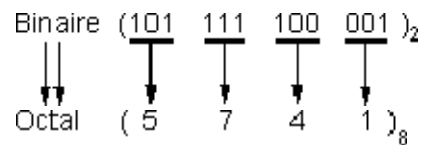
Exprimons $(47)_{10}$ dans le système octal et le système binaire. Nous obtenons :



Nous pouvons remarquer qu'après 3 divisions en binaire nous avons le même quotient qu'après une seule en octal. De plus le premier reste en octal obtenu peut être mis en relation directe avec les trois premiers restes en binaire.

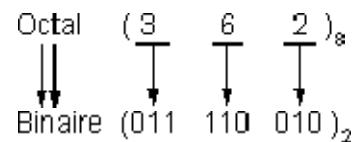
Cette propriété d'équivalence entre chaque chiffre octal et chaque groupe de 3 chiffres binaires permet de passer facilement d'un système à base 8 à un système à base 2 et vice versa.

Exemple de conversion binaire octal:



Donc $(101111100001)_2 = (5741)_8$

Exemple de conversion octal binaire :

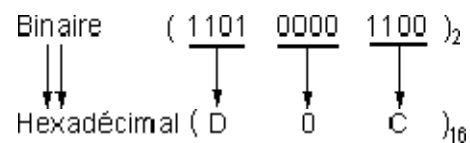


Donc $(362)_8 = (011110010)_2$

Conversion entre les nombres binaires et les nombres hexadécimaux

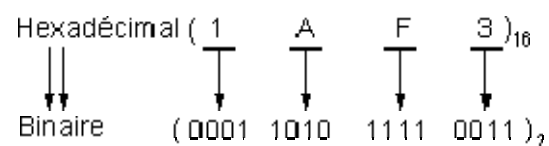
La propriété d'équivalence que nous venons de voir entre le binaire et l'octal existe entre l'hexadécimal et le binaire. La seule différence est qu'il faut exprimer chaque caractère hexadécimal à l'aide de 4 informations binaires.

Exemple de conversion binaire hexadécimal:



Donc $(110100001100)_2 = (D0C)_{16}$.

Exemple de conversion hexadécimal binaire :



Donc $(1AF3)_{16} = (0001101011110011)_2$.

Codes

Code Binaire réfléchi (Gray)

Dans ce codage, un seul bit change d'état lorsque l'on passe d'un terme au suivant. A l'apparition d'une variable supplémentaire on fait la symétrie du code déjà obtenu plus le nouveau bit à 1.

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

Intérêt: Ce codage évite les états indéterminés lors du passage d'un terme à un autre terme adjacent.

Code binaire DCB (Décimal Codé Binaire)

Dans ce codage, chaque chiffre décimal est converti en binaire, indépendamment des autres chiffres.

$$(1929)_{10} = (0001\ 1001\ 0010\ 1001)_{DCB}$$

Ce code est utilisé dans les systèmes traitant des nombres décimaux uniquement. Dans les petites calculatrices de poche par exemple.

Code ASCII

ASCII : (American Standard Code for Information Interchange) Norme d'encodage informatique des caractères alphanumériques de l'alphabet latin. La norme ASCII établit une correspondance entre une représentation binaire des caractères de l'alphabet latin et les symboles, les signes, qui constituent cet alphabet.

Par exemple, le caractère "a" est associé à "01100001" et "A" à "01000001".

La norme ASCII permet ainsi à toutes sortes de machines de stocker, analyser et communiquer de l'information textuelle. En particulier, la quasi totalité des ordinateurs personnels et des stations de travail utilisent l'encodage ASCII.

Il existe deux modes de transmission des fichiers Informatiques : le mode ASCII et le mode Binaire. Un mauvais choix dans le mode de transmission peut rendre un fichier inexploitable.

En mode ASCII, le logiciel de transmission adressera le code ASCII de chaque caractère. Ce mode est particulièrement destiné à la transmission des fichiers dits "texte" (HTML, sources, scripts).

En mode Binaire, le logiciel de transmission enverra les bits par paquets. Ce mode convient aux fichiers, comme les exécutables, les images, les sons, etc...

Décimal	Octal	Hex	Binaire	Caractère	
-----	-----	---	-----	-----	
000	000	00	00000000	NUL	(Null char.)
001	001	01	00000001	SOH	(Start of Header)
002	002	02	00000010	STX	(Start of Text)
003	003	03	00000011	ETX	(End of Text)
004	004	04	00000100	EOT	(End of Transmission)
005	005	05	00000101	ENQ	(Enquiry)
006	006	06	00000110	ACK	(Acknowledgment)
007	007	07	00000111	BEL	(Bell)
008	010	08	00001000	BS	(Backspace)
009	011	09	00001001	HT	(Horizontal Tab)
010	012	0A	00001010	LF	(Line Feed)
011	013	0B	00001011	VT	(Vertical Tab)
012	014	0C	00001100	FF	(Form Feed)
013	015	0D	00001101	CR	(Carriage Return)
014	016	0E	00001110	SO	(Shift Out)
015	017	0F	00001111	SI	(Shift In)
016	020	10	00010000	DLE	(Data Link Escape)
017	021	11	00010001	DC1	(XON) (Device Control 1)
018	022	12	00010010	DC2	(Device Control 2)
019	023	13	00010011	DC3	(XOFF) (Device Control 3)
020	024	14	00010100	DC4	(Device Control 4)
021	025	15	00010101	NAK	(Negative Acknowledgement)
022	026	16	00010110	SYN	(Synchronous Idle)
023	027	17	00010111	ETB	(End of Trans. Block)
024	030	18	00011000	CAN	(Cancel)
025	031	19	00011001	EM	(End of Medium)
026	032	1A	00011010	SUB	(Substitute)
027	033	1B	00011011	ESC	(Escape)
028	034	1C	00011100	FS	(File Separator)
029	035	1D	00011101	GS	(Group Separator)
030	036	1E	00011110	RS	(Request to Send) (Record Separator)
031	037	1F	00011111	US	(Unit Separator)
032	040	20	00100000	SP	(Space)
033	041	21	00100001	!	(exclamation mark)
034	042	22	00100010	"	(double quote)
035	043	23	00100011	#	(number sign)
036	044	24	00100100	\$	(dollar sign)
037	045	25	00100101	%	(percent)
038	046	26	00100110	&	(ampersand)
039	047	27	00100111	'	(single quote)
040	050	28	00101000	((left opening parenthesis)
041	051	29	00101001)	(right closing parenthesis)
042	052	2A	00101010	*	(asterisk)
043	053	2B	00101011	+	(plus)
044	054	2C	00101100	,	(comma)
045	055	2D	00101101	-	(minus or dash)
046	056	2E	00101110	.	(dot)
047	057	2F	00101111	/	(forward slash)
048	060	30	00110000	0	
049	061	31	00110001	1	
050	062	32	00110010	2	
051	063	33	00110011	3	
052	064	34	00110100	4	
053	065	35	00110101	5	
054	066	36	00110110	6	
055	067	37	00110111	7	
056	070	38	00111000	8	
057	071	39	00111001	9	
058	072	3A	00111010	:	(colon)
059	073	3B	00111011	;	(semi-colon)
060	074	3C	00111100	<	(less than sign)
061	075	3D	00111101	=	(equal sign)

062	076	3E	00111110	>	(greater than sign)
063	077	3F	00111111	?	(question mark)
064	100	40	01000000	@	(AT symbol)
065	101	41	01000001	A	
066	102	42	01000010	B	
067	103	43	01000011	C	
068	104	44	01000100	D	
069	105	45	01000101	E	
070	106	46	01000110	F	
071	107	47	01000111	G	
072	110	48	01001000	H	
073	111	49	01001001	I	
074	112	4A	01001010	J	
075	113	4B	01001011	K	
076	114	4C	01001100	L	
077	115	4D	01001101	M	
078	116	4E	01001110	N	
079	117	4F	01001111	O	
080	120	50	01010000	P	
081	121	51	01010001	Q	
082	122	52	01010010	R	
083	123	53	01010011	S	
084	124	54	01010100	T	
085	125	55	01010101	U	
086	126	56	01010110	V	
087	127	57	01010111	W	
088	130	58	01011000	X	
089	131	59	01011001	Y	
090	132	5A	01011010	Z	
091	133	5B	01011011	[(left opening bracket)
092	134	5C	01011100	\	(back slash)
093	135	5D	01011101]	(right closing bracket)
094	136	5E	01011110	^	(caret cirumflex)
095	137	5F	01011111	_	(underscore)
096	140	60	01100000		
097	141	61	01100001	a	
098	142	62	01100010	b	
099	143	63	01100011	c	
100	144	64	01100100	d	
101	145	65	01100101	e	
102	146	66	01100110	f	
103	147	67	01100111	g	
104	150	68	01101000	h	
105	151	69	01101001	i	
106	152	6A	01101010	j	
107	153	6B	01101011	k	
108	154	6C	01101100	l	
109	155	6D	01101101	m	
110	156	6E	01101110	n	
111	157	6F	01101111	o	
112	160	70	01110000	p	
113	161	71	01110001	q	
114	162	72	01110010	r	
115	163	73	01110011	s	
116	164	74	01110100	t	
117	165	75	01110101	u	
118	166	76	01110110	v	
119	167	77	01110111	w	
120	170	78	01111000	x	
121	171	79	01111001	y	
122	172	7A	01111010	z	
123	173	7B	01111011	{	(left opening brace)
124	174	7C	01111100		(vertical bar)
125	175	7D	01111101	}	(right closing brace)
126	176	7E	01111110	~	(tilde)
127	177	7F	01111111	DEL	(delete)


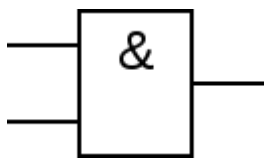
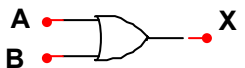
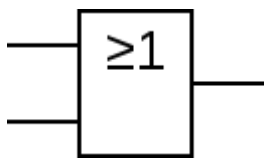
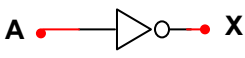
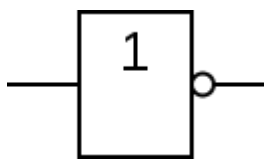
Chapitre 2 : Circuits de Logiques Combinatoires

Portes Logiques

Les fonctions logiques combinatoires directement issues des mathématiques (algèbre de Boole) sont les outils de base de l'électronique numérique donc de l'automatisme et de l'informatique. Elles sont mises en oeuvre en électronique sous forme de portes logiques. Ainsi les circuits électroniques calculent des fonctions logiques de l'algèbre de boole. Ces portes électroniques sont construites à partir de plusieurs transistors reliés entre eux.

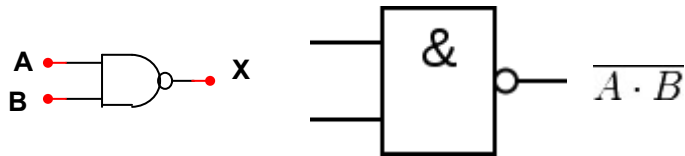
En algèbre de Boole, une donnée, qu'elle soit en entrée ou en sortie, n'a que deux niveaux possibles. Dans le cas de circuits électroniques, les deux niveaux sont représentés par deux niveaux de tension, 5v «haut» 22et 0v «bas». **Ces deux niveaux seront notés ici 1 et 0.**

Le tableau ci-dessous montre les différents types des portes logiques, leurs symboles américain et européen, ainsi que leurs tables de vérité.

Type	Symbole américain	Symbole européen	Opération booléenne entre A et B	Table de vérité																		
<u>ET (AND)</u>			$A \cdot B$	<table><tr><th colspan="2">Entrée</th><th>Sortie</th></tr><tr><th>A</th><th>B</th><th>A ET B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	Entrée		Sortie	A	B	A ET B	0	0	0	0	1	0	1	0	0	1	1	1
	Entrée		Sortie																			
	A	B	A ET B																			
	0	0	0																			
	0	1	0																			
1	0	0																				
1	1	1																				
<u>OU (OR)</u>			$A + B$	<table><tr><th colspan="2">Entrée</th><th>Sortie</th></tr><tr><th>A</th><th>B</th><th>A OU B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	Entrée		Sortie	A	B	A OU B	0	0	0	0	1	1	1	0	1	1	1	1
	Entrée		Sortie																			
	A	B	A OU B																			
	0	0	0																			
	0	1	1																			
1	0	1																				
1	1	1																				
<u>NON (NOT)</u>			\overline{A}	<table><tr><th>Entrée</th><th>Sortie</th></tr><tr><th>A</th><th>NON A</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	Entrée	Sortie	A	NON A	0	1	1	0										
	Entrée	Sortie																				
	A	NON A																				
0	1																					
1	0																					

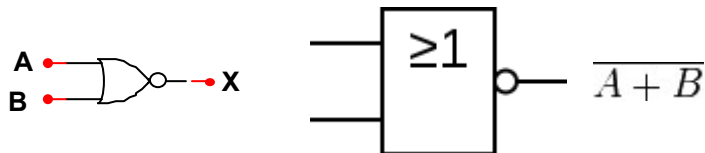
Le petit cercle utilisé sur la représentation de la porte NON est généralement utilisé dans les diagrammes pour montrer qu'une entrée ou une sortie est inversée.

NON-ET (NAND)



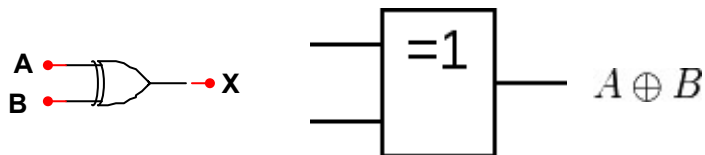
Entrée		Sortie
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

NON-OU (NOR)



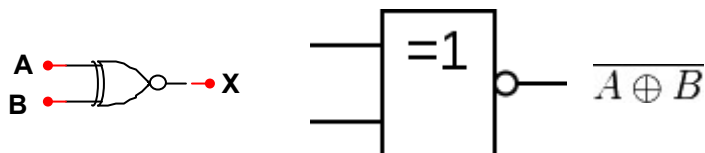
Entrée		Sortie
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

OU exclusif (XOR)



Entrée		Sortie
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

NON-OU exclusif (XNOR)



Entrée		Sortie
A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

Algèbre de Boole

Règles générales

Les lois de composition sont des règles simples d'écriture permettant de simplifier les expressions algébriques booléennes.

Commutativité

Somme logique : $a + b = b + a$

Produit logique : $a \cdot b = b \cdot a$

Associativité

Somme logique : $a + b + c = (a + b) + c = a + (b + c)$

Produit logique : $a \cdot b \cdot c = (a \cdot b) \cdot c = a \cdot (b \cdot c)$

Distributivité

Somme logique : $a \cdot (b + c) = a \cdot b + a \cdot c$

Produit logique : $a + (b \cdot c) = (a + b) \cdot (a + c)$

L'idempotence

$A \cdot A$ est équivalent à A

$A + A$ est équivalent à A

L'inversion

$A \cdot \overline{A}$ est équivalent à 0

$A + \overline{A}$ est équivalent à 1

L'identité

$A \cdot 1$ est équivalent à A

$A + 0$ est équivalent à A

$A \cdot 0$ est équivalent à 0

L'absorption

$A + \overline{A} \cdot B$ est équivalent à $A + B$

Théorème de De Morgan

Complément d'une somme logique

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

La table de vérité ci-dessous montre que $Y_1 = Y_2$.

b	a	a+b	$Y_1 = \overline{a+b}$	\overline{b}	\overline{a}	$Y_2 = \overline{a} \cdot \overline{b}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Généralisation : $\overline{X_1 + X_2 + X_3 + \dots + X_n} = \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} \cdot \dots \cdot \overline{X_n}$

Complément d'un produit logique

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

La table de vérité ci-dessous montre que $Y_1 = Y_2$.

b	a	a.b	$Y1 = \overline{a.b}$	\bar{b}	\bar{a}	$Y2 = \bar{a} + \bar{b}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Généralisation : $\overline{X_1.X_2.X_3....X_n} = \overline{X_1} + \overline{X_2} + \overline{X_3} + + \overline{X_n}$

Universalité des portes NAND et NOR

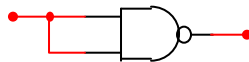
Les fonctions NAND et NOR sont dites « universelles », car elles permettent de reconstituer toutes les autres fonctions logiques.

Universalité de la porte NAND

La porte Inverseur

Théorème de Boole : $\bar{A} = \overline{A.A}$;

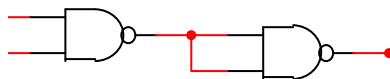
Il suffit de mettre $A = B$, et dès lors la porte NAND peut être utilisée comme porte inverseur de cette manière :



Equivalent à une porte NOT

La porte ET

Il suffit d'inverser le résultat de la porte NAND : $A.B = \overline{\overline{A.B}}$

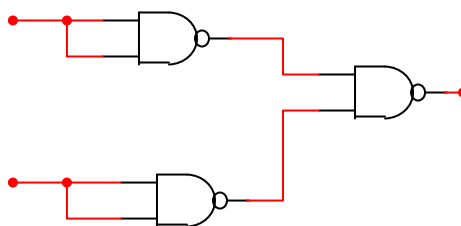


Equivalent à une porte ET

La porte OU

Théorème de De Morgan : $\overline{A + B} = \overline{A}.\overline{B}$

Et si nous inversons le tout : $A + B = \overline{\overline{A + B}}$



Equivalent à une porte OU

Universalité de la porte NOR

Fonction Inverseur

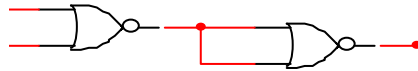
$$A = \overline{A + A}$$



Equivalent à une porte NOT

Fonction OU

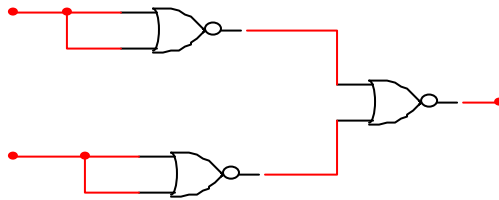
$$A + B = \overline{\overline{A} \cdot \overline{B}}$$



Equivalent à une porte OU

Fonction ET

$$A.B = \overline{\overline{A} + \overline{B}}$$



Equivalent à une porte ET

Chapitre 3 : Conception de la logique combinatoire

Conception des circuits avec la table de vérité et simplification

Une fonction X peut comporter n variables. Nous obtenons 2^n combinaisons de ces n variables. Pour chacune de ces combinaisons, la fonction peut prendre une valeur 0 ou 1. L'ensemble de ces 2^n combinaisons des variables et les valeurs associées de la fonction représente la Table de Vérité.

Exemple d'une table de vérité :

c	b	a	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Forme canonique

Pour écrire l'équation de X en fonction des 3 variables il faut dire :

$X=1$ si $c=0$ et $b=0$ et $a=1$
ou $c=0$ et $b=1$ et $a=0$
ou $c=1$ et $b=0$ et $a=1$
ou $c=1$ et $b=1$ et $a=0$
ou $c=1$ et $b=1$ et $a=1$

Autant de termes que de fois que la fonction est égale à 1.

Ce qui donne une écriture "algébrique" en notant : La variable par sa lettre si elle vaut 1 (ex : si a vaut 1 nous écrirons a) et par sa lettre surlignée si elle vaut 0 (ex : si a vaut 0 nous écrirons \bar{a} et nous lirons a barre).

Pour la table de vérité ci-dessus, cela nous donne

$$X = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + abc$$

Cette forme d'écriture est appelée FORME CANONIQUE.

Simplification

$$X = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + abc = (\bar{a}\bar{b} + \bar{a}b)\bar{c} + (\bar{a}\bar{b} + \bar{a}b)c + abc$$

D'après les règles de l'algèbre de Boole, nous obtenons :

$$X = (a\bar{b} + \bar{a}b)\bar{c} + (a\bar{b} + \bar{a}b)c + abc = abc + a \oplus b$$

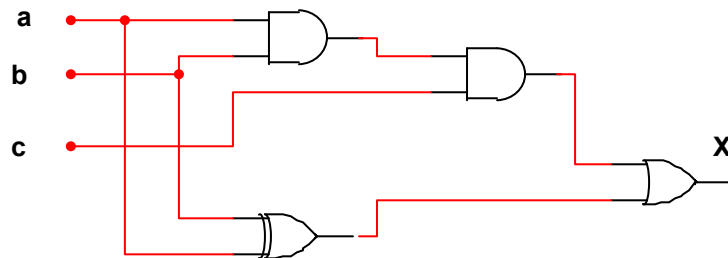


Diagramme de Karnaugh

Nous avons vu que les règles de l'algèbre de Boole permettent de simplifier les fonctions ; cette méthode est cependant relativement lourde et ne permet jamais de savoir si l'on aboutit à une expression minimale de la fonction ou pas.

Les tableaux de KARNAUGH permettent la simplification des équations logiques. Ils comportent 2^n cases, n étant le nombre de variables d'entrée, organisés selon le code GRAY. (Ex : 4 variables donnent 16 cases).

Chaque case correspond à une combinaison possible des variables d'entrée;

Chaque combinaison exprimée dans l'équation sera représentée par un « 1 » dans la case correspondante.

Tableau de Karnaugh à Trois variables

A chaque case est associé un triplet des valeurs a, b, c.

		a b			
		00	01	11	10
c	0				
	1				

Exemple : La case n° 1 représentera le triplet {0, 0,0} ou a = 0, b = 0 et c = 0.

Tableau de Karnaugh à Quatre variables

A chaque case est associé un quadruplet des valeurs a, b, c, d.

				a b			
				00	01	11	10
c d	00						
	01						
	11						
	10						

La case n° 4 représentera le quadruplet {1,0,0,0} ou a = 1, b = 0, c = 0 et d = 0

La case n° 11 représentera le quadruplet {1,1,1,1} ou a = 1, b = 1, c = 1 et d = 1

La case n° 16 représentera le quadruplet {1,0,1,0} ou a = 1, b = 0, c = 1 et d = 0

Adjacences des cases

Dans chaque cas, l'ordre d'écriture des états des variables (Code Gray) fait qu'entre deux cases voisines (en ligne ou en colonne) une seule variable change d'état ; on dit de telles cases qu'elles sont adjacentes.

		a b			
		c 0	0 1	1 1	1 0
Changement de d	0 0	1	2	3	4
	0 1	5	6	7	0
Changement de c	1 0	9	10	11	12
Changement de d	1 1	13	14	15	16

Attention à l'ordre d'écriture !

La case 2 correspond à $a = 0 ; b = 1 ; c = 0 ; d = 0$

La case 3 correspond à $a = 1 ; b = 1 ; c = 0 ; d = 0$

Lorsque nous passons de 2 à 3, seule la variable "a" change d'état : 2 et 3 sont adjacentes.

Lorsque nous passons de 2 à 1, seule la variable "b" change d'état : 2 et 1 sont adjacentes.

Lorsque nous passons de 2 à 6, seule la variable "d" change d'état : 2 et 6 sont adjacentes.

Enfin, lorsque nous passons de 2 à 14, seule la variable "c" change d'état : 2 et 14 sont adjacentes.

Nous venons de déterminer les adjacences de la case n° 2.

Règles de regroupement des cases adjacentes de 1

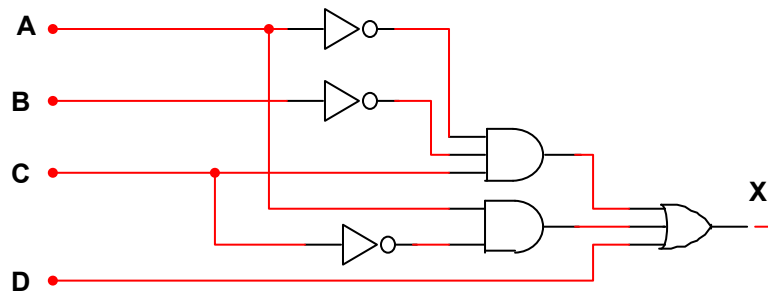
1. Un groupes doit contenir 1, 2, 4, 8 toujours 2^n cases de 1. Chaque case de 1 doit appartenir à un groupe, sinon elle constitue en soit un groupe.
2. Un groupe doit contenir le maximum possible des 1. Un 1 peut appartenir à plusieurs groupes.
3. Le nombre de groupes doit être le minimum possible.

La valeur d'un groupe

Les éléments ayant changé de valeur dans le groupe seront éliminer.

	\bar{C}	C	
\bar{A}	1	1	1
	1	1	
B			
A	1	1	1
	1	1	
\bar{B}			
	\bar{D}	D	\bar{D}

$$X = D + A\bar{C} + C\bar{A}\bar{B}$$



Nous pouvons observer les faits suivants :

- Quand un terme ne contient qu'une variable il occupe une zone de 8 cases
- Quand un terme est un produit de 2 variables il occupe une zone de 4 cases
- Quand un terme est un produit de 3 variables il occupe une zone de 2 cases
- Quand un terme est un produit de 4 variables il occupe une zone d'1 cases

Codeur

Le codeur convertit une information, comme un nombre décimal ou une lettre de l'alphabet, en un code binaire. Il dispose de N entrées, dont une seule est active à la fois, et de n sorties, tel que $N \leq 2^n$ (nombre des codes possible avec n sorties).

Un codeur à 8 entrées de E0 à E7 disposait de 3 sorties de S0 à S2. On ajoutera une sorties supplémentaire Gs d'activation d'une entrée.

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

$$\begin{cases} S0 = E1 + E3 + E5 + E7 \\ S1 = E2 + E3 + E6 + E7 \\ S2 = E4 + E5 + E6 + E7 \\ S3 = E1 + E3 + E5 + E7 \\ Gs = E1 + E2 + E3 + E4 + E5 + E6 + E7 \end{cases}$$

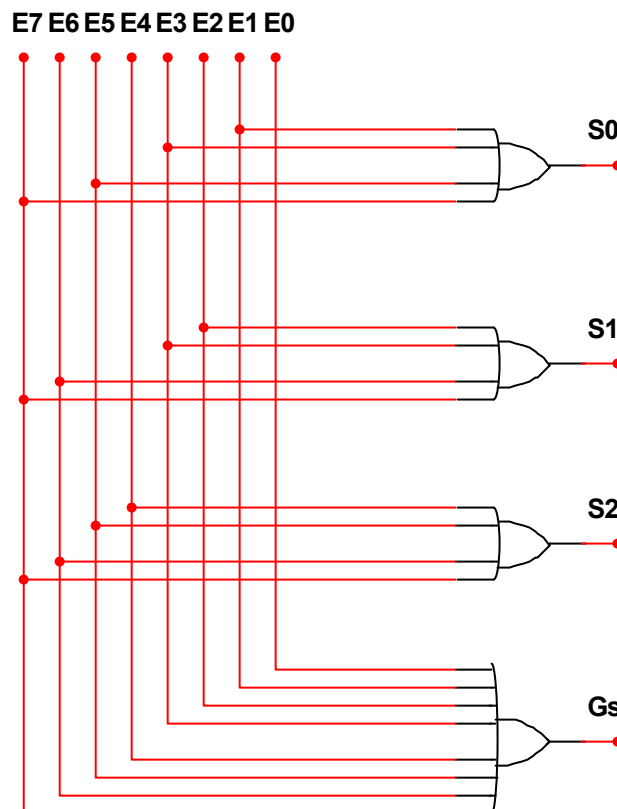


Figure : codeur 8 entrées, 3 sorties et une sortie d'activation d'une entrée

Décodeur

La fonction de base d'un décodeur est de détecter la présence d'une combinaison spécifique de bits (code). Un décodeur comporte 2^n sorties pour n entrées.

Décodeur 2 entrées vers 4 sorties

Entrées		Sorties			
E1	E0	S3	S2	S1	S0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Table de vérité

L'équation de la forme canonique :

$$\begin{cases} S0 = \overline{E0}.\overline{E1} \\ S1 = E0.\overline{E1} \\ S2 = \overline{E0}.E1 \\ S3 = E0.E1 \end{cases}$$

Ci-dessous le schéma du circuit du décodeur.

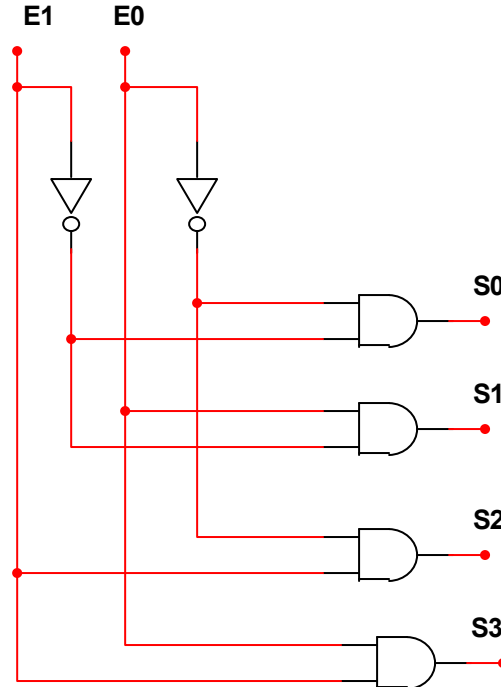


Figure : décodeur 2 entrée, 4 sorties

Transcodeur

Un transcodeur est un circuit permettant de passer d'un code à un autre. Il fait correspondre à un code en entrée sur n lignes, un autre code en sortie sur m lignes.

Afficheur DCB-7 Segments

Un afficheur DCB sept segments reçoit un code DCB à ces entrées (ABCD) et produit sept sorties (a,b, c, d, e, f, g) pour piloter l'afficheur à sept segment, afin de fournir un affichage décimal.

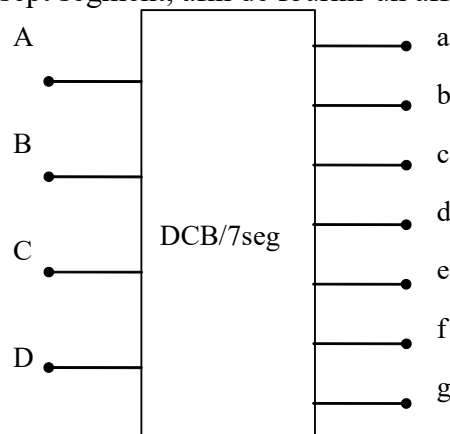


Figure : Transcodeur DCB/6segments

Chaque segment est une diode électroluminescente, qui s'allume lors ce que la sortie correspondante vaut 1 et s'éteint dans le cas contraire.

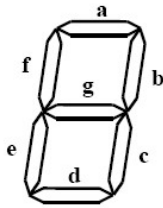


Figure : Afficheur 7 segments

En fonction du code DCB présent sur les entrées, les segments appropriés vont s'allumer et les autres segments vont s'éteindre, conformément à la figure ci-dessus. On pourra alors enduire la table de vérité du circuit.

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Table de vérité du transcodeur DCB/7segments

Multiplexeur

Un multiplexeur est un circuit permettant d'acheminer les informations numériques de plusieurs sources sur une seule ligne (une sortie), grâce à plusieurs entrées de sélection. Si N est le nombre des entrées des données, n est le nombre des entrées de sélection, tel que $2^{n-1} < N \leq 2^n$.

Multiplexeur 4 vers 1 : Un multiplexeur à 4 entrées de données possède 2 entrées de sélection.

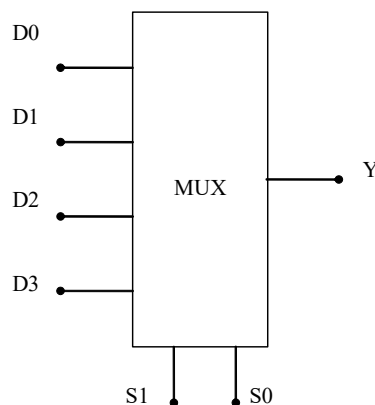


Figure : multiplexeur 4 vers 1 (entité)

L'entité d'un circuit est sa vue externe : les entrées et les sorties.

La table de vérité ci-dessous montre la valeur de sortie en fonction du code présent sur les entrées de sélection.

Entrées de sélection		Sortie
S1	S0	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

La forme canonique de la sorties en fonction des entrées : $Y = D0\overline{S0}\overline{S1} + D1S0\overline{S1} + D2\overline{S0}S1 + D3S0S1$

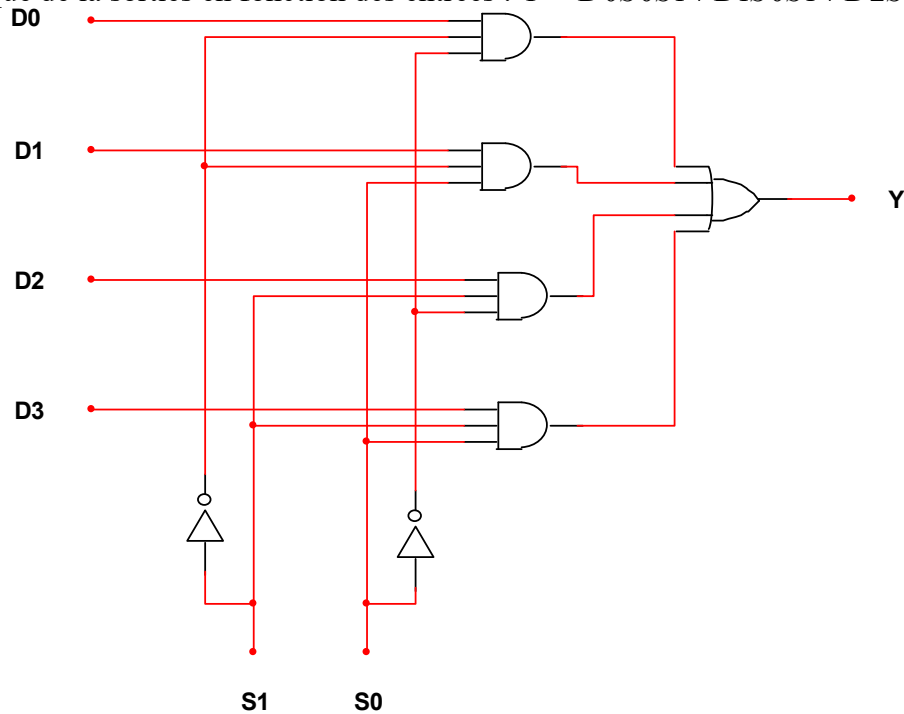


Figure : multiplexeur 4 vers 1 (architecture)

L'architecture d'un circuit est sa vue interne : la relation entre les sorties et les entrées.

Demultiplexeur

Un démultiplexeur effectue l'opération inverse d'un multiplexeur. Il prend les entrées reçues en série sur une seule ligne (une entrée) et les distribue à plusieurs lignes de sorties, en fonction des entrées de sélection. Si N est le nombre des sorties, n est le nombre des entrées de sélection, tel que $2^{n-1} < N \leq 2^n$.

Une démultiplexeur 1 vers 4, possède 1 entrée de données, 4 sorties et 2 entrées de sélection.

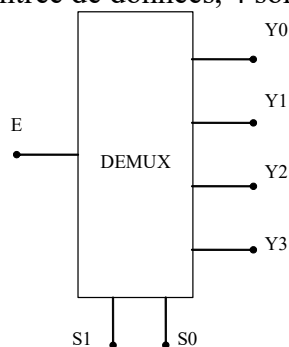


Figure : démultiplexeur (entité)

La table de vérité ci-dessous montre les valeurs de sorties en fonction du code présent sur les entrées de sélection.

Entrées de sélection		Sorties			
S1	S0	Y0	Y1	Y2	Y3
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

La forme canonique des sorties en fonction des entrées :

$$\begin{cases} Y0 = E.\overline{S0}.\overline{S1} \\ Y1 = E.S0.\overline{S1} \\ Y2 = E.\overline{S0}.S1 \\ Y3 = E.S0.S1 \end{cases}$$

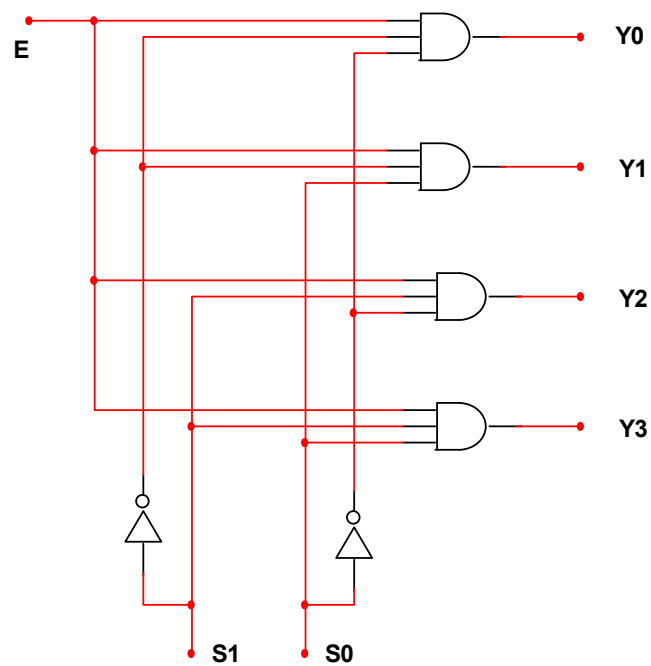


Figure : démultiplexeur 1 vers 4 (architecture)

Chapitre 4 : Fonctions Arithmétiques

Nombres signés

Un nombre signé binaire comprend un signe et une information de grandeur. Le signe détermine s'il s'agit d'un nombre positif ou négatif tandis que la grandeur détermine sa valeur. Il existe trois notation pour représenter les nombres entiers signés en binaire : notation signe/valeur absolue, notation en complément à 1 et notation en complément à 2.

Bit de Signe

Dans un nombre binaire signé, le bit situé le plus à gauche est le bit de signe, qui indique si le nombre est positif ou négatif. Un bit de signe 0 désigne un nombre positif et un bit de signe 1 désigne un nombre négatif.

Notation signe/valeur absolue

Dans cette notation un nombre négatif possède les mêmes bits de grandeur que son nombre positif correspondant mais comporte un bit de signe de 1 au lieu de 0.

Exemple : Nombres binaires signé sur 4 bits

5 = 0101 tandis que -5 = 1101 (la seule différence est le bit de signe)

Le tableau ci-dessous résume les nombres binaires signés sur 4 bits (1 bit de signe et 3 bits de grandeur).

Nombre décimal	Nombre binaire signé			
	Bit de signe	Bits de grandeur		
7	0	1	1	1
6	0	1	1	0
5	0	1	0	1
4	0	1	0	0
3	0	0	1	1
2	0	0	1	0
1	0	0	0	1
0	0	0	0	0
-1	1	0	0	1
-2	1	0	1	0
-3	1	0	1	1
-4	1	1	0	0
-5	1	1	0	1
-6	1	1	1	0
-7	1	1	1	1

Notation en complément à 1

Dans cette notation les nombres négatifs deviennent les compléments à 1 de leurs nombres positifs correspondants. Le complément à 1 est obtenu par l'inversion de tous les bits 0 vers 1 et 1 vers 0.

Exemple : Nombres binaires signé sur 4 bits

5 = 0101 tandis que -5 = 1010 (Tous les bits sont modifiés 0 vers 1 et 1 vers 0)

Nombre décimal	Nombre binaire signé			
	Bit de signe	Bits de grandeur		
7	0	1	1	1
6	0	1	1	0
5	0	1	0	1
4	0	1	0	0
3	0	0	1	1
2	0	0	1	0
1	0	0	0	1
0	0	0	0	0
-1	1	1	1	0
-2	1	1	0	1
-3	1	1	0	0
-4	1	0	1	1
-5	1	0	1	0
-6	1	0	0	1
-7	1	0	0	0

Le tableau ci-dessus résume les nombres binaires signés sur 4 bits, en notation complément à 1.

Notation en complément à 2

Dans cette notation les nombres négatifs deviennent les compléments à 2 des nombres positifs correspondants. Le complément à 2 est obtenu par l'addition de 1 au complément à 1.

Exemple : Nombres binaires signé sur 4 bits

5 = 0101 ;

complément à 1 de 5 = 1010 ;

complément à 2 de 5 égale 1011.

Donc -5 = 1011.

Nombre décimal	Nombre binaire signé			
	Bit de signe	Bits de grandeur		
7	0	1	1	1
6	0	1	1	0
5	0	1	0	1
4	0	1	0	0
3	0	0	1	1
2	0	0	1	0
1	0	0	0	1
0	0	0	0	0
-1	1	1	1	1
-2	1	1	1	0
-3	1	1	0	1
-4	1	1	0	0
-5	1	0	1	1
-6	1	0	1	0
-7	1	0	0	1
-8	1	0	0	0

Notes que dans cette notation on peut exprimer -8 sur 4 bits, ce qui n'était pas possible avec les deux précédentes notations.

Pour les nombres signés en compléments à 2, l'échelle des valeurs pour des nombres possédant une quantité de n bits est :

$$\text{De } -2^{n-1} \quad \text{À} \quad +(2^{n-1} - 1)$$

Avec 4 bits l'échelle est de - $2^3 = -8$ À $+(2^3 - 1) = +7$.

Avec 8 bits l'échelle est de $-2^7 = -128$ À $+(2^7 - 1) = +127$.

La valeur décimale d'un nombre binaire signé sur n bit, en notation complément à 2, est la somme des poids positionnels de chaque bit. Le poids du bit de signe est de -2^{n-1}

Exemples :

Convertir en décimal les nombres binaires suivants signés en notation compléments à 2.

a) 01010110

b) 10101010

Solution

$$\text{a) } 01010110 = 0 \times (-2^7) + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 0 + 64 + 0 + 16 + 0 + 4 + 2 + 0 = 86.$$

$$\text{b) } 10101010 = 1 \times (-2^7) + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = -128 + 0 + 32 + 0 + 8 + 2 = -86.$$

Opérations Arithmétiques avec les nombres signés :

L'opération de base en arithmétique est l'opération de l'addition. L'opération de la soustraction (addition d'un nombre positive et d'un autre nombre) est équivalente à l'addition du complément à 2 du nombre négatif. L'opération de la multiplication est équivalente à plusieurs opérations d'addition successives, tandis que l'opération de la division est équivalente à plusieurs opérations de soustraction successives.

Dans ce cours on va se limiter à l'opération de l'addition de deux nombres positifs ou négatives. Les opérations se font avec les nombres signés sur 8 bits en notation complément à 2.

Cas de deux nombre positifs

Nombres décimaux	Nombres binaires signés Cà2								
+7	0	0	0	0	0	1	1	1	
+4	0	0	0	0	0	1	0	0	
=11	0	0	0	0	1	0	1	1	

Le résultat est positive est exprime une valeur réelle (non complémentée).

Cas d'un nombre positif plus grand que le nombre négatif

Nombres décimaux	Nombres binaires signés Cà2								
15		0	0	0	0	1	1	1	1
-6		1	1	1	1	1	0	1	0
	retenue								
+9	1	0	0	0	0	1	0	0	1
			1	0	0	1	0	1	

La retenue finale est rejetée. La somme positive et exprime une valeur binaire réelle.

Cas d'un nombre négatif plus grand que le nombre positif

Nombres décimaux	Nombres binaires signés Cà2								
16	0	0	0	1	0	0	0	0	
-24	1	1	1	0	1	0	0	0	
-8	1	1	1	1	1	0	0	0	

La somme est négative est s'exprime sous la forme de complément à 2.

Cas de deux nombre négatifs

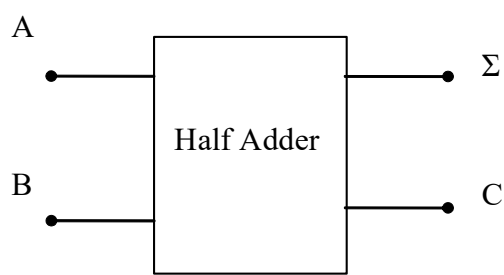
Nombres décimaux	Nombres binaires signés Cà2								
-5		1	1	1	1	1	0	1	1
-9		1	1	1	1	0	1	1	1
	retenue								
-14	1	1	1	1	1	0	0	1	0

La retenue finale est toujours rejetée. La somme est négative et s'exprime en complément à 2.

Additionneurs de base

Demi additionneur

Un demi additionneur additionne deux bits et produit une somme et une retenue à sa sortie.



Entrées		Sorties	
B	A	Σ	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table de vérité d'un demi additionneur

A partir de la table de vérité, on en déduit la forme canonique des sorties en fonction des entrées :

$$\begin{cases} \Sigma = A \oplus B \\ C = A.B \end{cases}$$

Circuit logique du demi additionneur

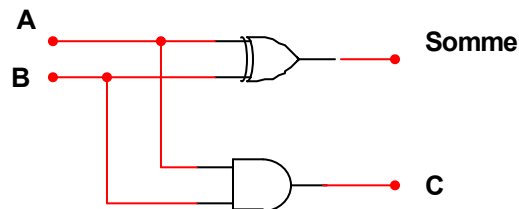
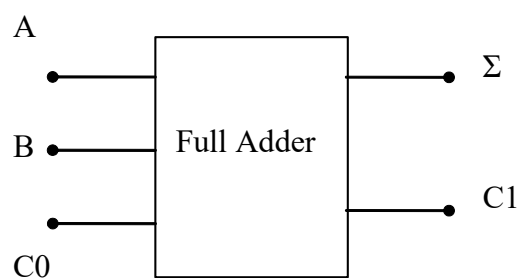


Figure : demi additionneur

Additionneur complet

Un additionneur complet prend deux bits d'entrée et une retenue d'entrée et produit une sortie de somme et une sortie de retenue.



Entité d'un additionneur complet

Entrées			Sorties	
B	A	C0	Σ	C1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table de vérité d'un additionneur complet

La forme canonique des sorties après simplification :

$$\begin{cases} \Sigma = A \oplus B \oplus C0 \\ C1 = C0.(A \oplus B) + A.B \end{cases}$$

Le circuit logique de l'additionneur complet est ci-dessous.

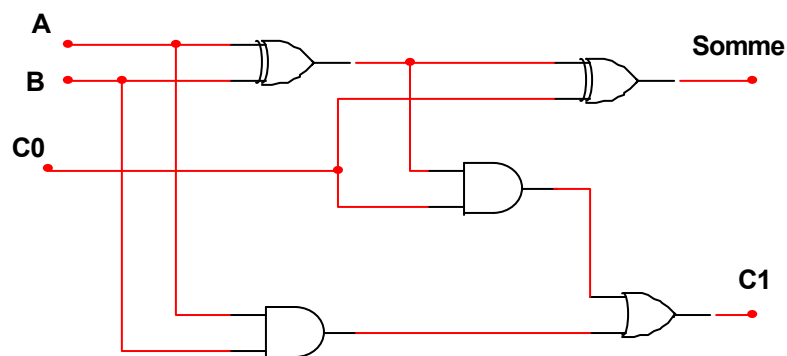


Figure : additionneur complet

Additionneur à n bits

Un additionneur à n bits, est un additionneur permettant de faire l'addition de deux nombres binaires de n bits chacun $A[0, 1, \dots, n-1]$ et $B[0, 1, \dots, n-1]$. Il produit une somme en sortie sur n bits $\Sigma[0, 1, \dots, n-1]$. L'additionneur n bits possède aussi une entrée de retenue $C0$ et une sortie de retenue Cn . La figure ci-dessous montre l'entité (vue externe) d'un additionneur 4 bits.

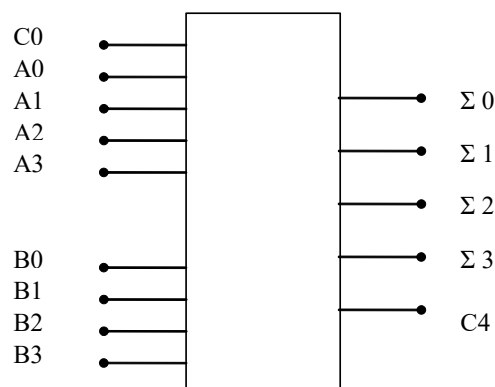


Figure : Additionneur 4 bits

Nous avons vu précédemment que l'addition de deux chiffres binaires, commence par l'addition des bits de poids faible le report est ajouté au bit du rang immédiatement à gauche. Cela pourra se faire par la mise en cascade de plusieurs additionneurs complets. La retenue d'entrée est mise à zéro ($C_0=0$).

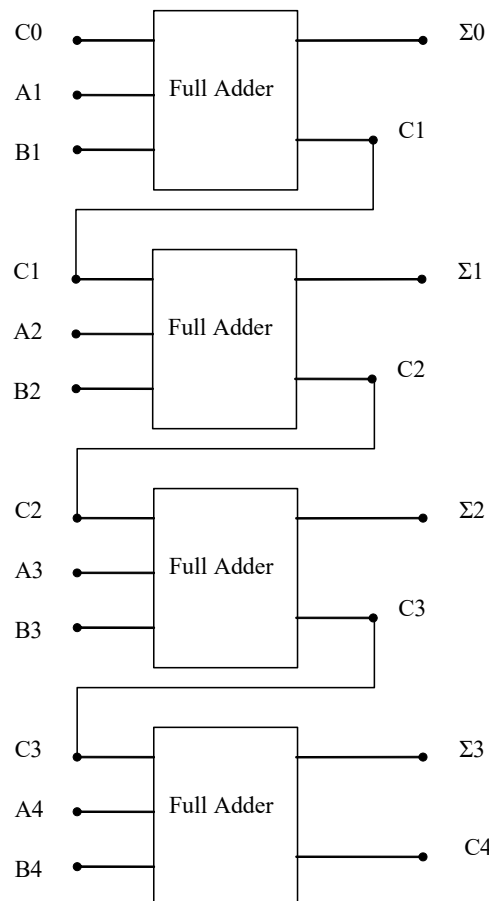


Figure : additionneur 4 bits (mise en cascade de plusieurs additionneurs complets)

Ce type d'additionneur est appelé additionneur à propagation de retenue, puis que la retenue se propage d'un étage à l'autre. La figure ci-dessus montre un additionneur à propagation de retenue de 4 bits.

Additionneur soustracteur à n bits

Un additionneur soustracteur est un circuit permettant de réaliser une addition ($A+B$) ou une soustraction ($A-B$) en fonction d'une entrée de commande C : $C=0$ pour l'addition et $C=1$ pour la soustraction.

Conformément à ce que nous avons vu précédemment, cela revient à faire l'addition de A et du complément à 2 de B (B_{C2}) quand $C=1$. Le complément à 2 est égale au complément à 1 plus 1.

La soustraction pourra alors se faire alors en utilisant un additionneur qui fait l'addition de A et du complément à 1 de B avec son entrée de retenue C_0 initialisée à 1.

On en déduit alors que la sortie :

$$\Sigma = A + \overline{B} + C_0 \text{ dans le cas de la soustraction } C=C_0=1.$$

$$\Sigma = A + B + C_0 \text{ dans le cas de l'addition } C=C_0=0.$$

On pose $Y = B$ si $C=0$ et $Y = \bar{B}$ si $C=1$. On remarque que $Y = B\bar{C} + \bar{B}C = B \oplus C$

Une porte ou-exclusif, permet de fournir B dans le cas ou $C=0$ et de fournir \bar{B} dans le cas ou $C=1$.

On trouve que dans les deux cas, la sortie est :

$$\Sigma = A + Y + C0$$

Le montage ci-dessous montre un additionneur soustracteur de 4 bits.

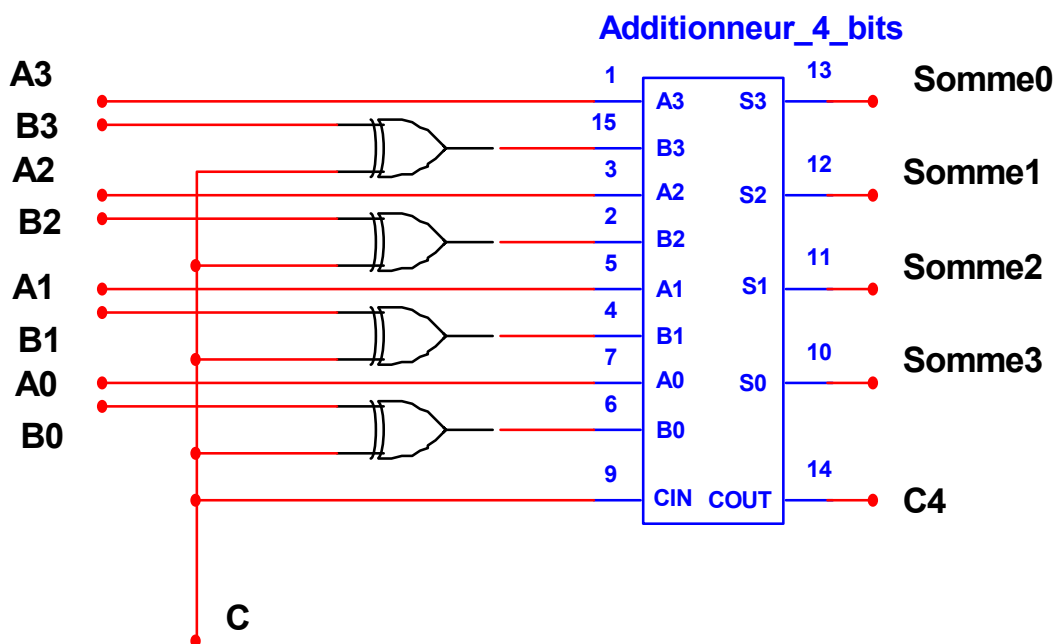


Figure : Additionneur soustracteur 4 bits

Gestion de dépassement

En additionnant deux nombres de signes différents, il ne peut pas y avoir de dépassement, par contre lorsqu'on additionne deux nombres de même signe, on a un dépassement si le signe du résultat est différent du signe des deux nombres additionnés.

Voyons Les exemples :

Nombres décimaux		Nombre binaire signé Cà2			
7		0	1	1	1
+					
7		0	1	1	1
=	retenue	0	1	1	1
14	résultat	1	1	1	0

Le bit de signe de la réponse est celui d'un nombre négatif, ce qui est manifestement une erreur.

Etant donnée que 14 en binaire signé complément à 2 nécessite plus que 4 bits, il y'a eu un dépassement (overflow) sur le rang du bit de signe.

Nombres décimaux		Nombre binaire signé Cà2			
-8		1	0	0	0
+					
-1		1	1	1	1
=	retenue	1	0	0	0
-9	résultat	0	1	1	1

On en déduit qu'il y'a dépassement (DPS) quand $A_{n-1} = B_{n-1}$ et $C_n \neq C_{n-1}$

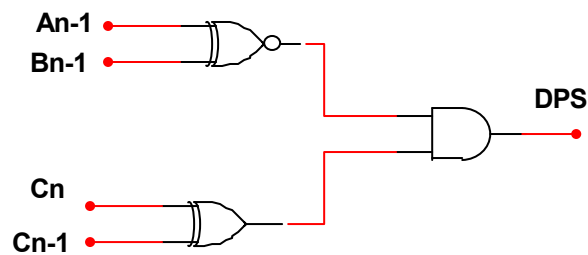


Figure : Circuit de détection de dépassement

Chapitre 5 : Unité Arithmétique et logique

Conclusion

Ce support de cours vous a fourni une vue détaillée de la conception de la logique digitale combinatoire. Vous avez vu que les sorties ne dépendent que des entrées.

Le premier chapitre vous a été utile pour comprendre les systèmes des numérations en particulier ceux utilisés dans le domaine de l'informatique (binaire, hexadécimal, octal et le décimal). Ce chapitre vous a été utile aussi pour comprendre les codes, en particulier les codes non pondérés (Gray), le code DCB adapté au codage et décodage des chiffres décimaux et le code ASCII adapté au codage des caractères (chiffres, lettres etc.).

Le deuxième chapitre vous permis de comprendre les fonctions de base de la logique digitale (portes logiques), les règles de l'algèbre permettant la simplification d'une fonction logique.

Le troisième chapitre vous ont permis de voir les outils des conceptions (table de vérité et diagramme de Karnaugh) d'un système digital combinatoire. Il vous a permis de voir en particulier les fonctions combinatoires telles que les codeurs, décodeurs (permettant le codage et le décodage des informations), le transcodeur (permettant le passage d'un code à un autre code) et aussi le multiplexeur, démultiplexeur (permettant de passer d'une communication parallèle en une communication série où l'inverse).

Le quatrième chapitre vous a fourni les outils nécessaires aux calculs arithmétiques. Dans un ordinateur toutes les opérations sont ramenées en une opération d'addition. Vous avez vu comment on peut réaliser une opération d'addition entre deux chiffres binaires de n bits chacun et aussi de voir comment on peut ramener une opération de soustraction une opération d'addition.

Dans un ordinateur toutes les opérations des calculs se font à l'aide de l'unité arithmétique et logique (circuit combinatoire). L'ordonnancement des opérations, la durée d'une opération, le stockage temporaire des données se font à l'aide des circuits séquentiels (séquenceurs, horloge, registres etc.).

La deuxième partie de ce cours sera consacrée à la logique séquentielle, dans ce cas les sorties ne dépendent pas seulement des entrées mais aussi de leurs valeurs à l'état précédent. Le système dispose d'une mémoire interne. La vitesse d'évolution des systèmes séquentiels est déterminée par un signal externe appelé signal d'horloge.