



IP PARIS

ELECINF102 : Processeurs et Architectures Numériques

Introduction et logique combinatoire

Guillaume Duc

`guillaume.duc@telecom-paris.fr`



Plan

Introduction

Logique booléenne

Représentation des nombres

Opérateurs arithmétiques

Notion de temps de propagation

Exercices

Informations utiles

- Site pédagogique sur eCampus
- Équipe enseignante : Laurent Sauvage (responsable du module), Sumanta Chaudhuri, Jean-Luc Danger, Tarik Graba, Ulrich Kühne, Yves Mathieu, Guillaume Duc
- Les supports de cours sont disponibles à l'adresse : <https://perso.telecom-paris.fr/duc/cours/elecinf102>
- Le module est progressif, chaque séance (cours, TD et TP) s'appuie sur les notions abordées lors des séances précédentes



Consignes

- Évaluation : examen individuel + QCM au début de chaque séance de TP

Objectifs du module

- Comprendre les grands principes de la logique combinatoire et séquentielle synchrone
- Comprendre les architectures d'opérateurs de traitement numériques
- Comprendre comment concevoir un processeur
- Découvrir certains enjeux de l'industrie micro-électronique



Plan

Introduction

Logique booléenne

Représentation des nombres

Opérateurs arithmétiques

Notion de temps de propagation

Exercices

Variables et fonctions logiques

Variables logiques

- Une variable logique est un élément qui appartient à l'ensemble $E = \{0, 1\}$
- Ne possède que deux états possibles : 0 ou 1

Fonctions logiques

- Fonction d'une ou plusieurs variables logiques

$$\begin{cases} E \times E \dots \times E \rightarrow E \\ e_0, e_1, \dots, e_n \rightarrow s = F(e_0, e_1, \dots, e_n) \end{cases}$$

Le bit

■ Représentation physique

- 2 niveaux de tension (0/5V ou -12/12V ou ...)
- 2 niveaux de courant électrique
- Absence/présence de lumière sur une fibre
- ...

■ Interprétations

- Vrai/Faux pour de la logique et le contrôle
- 1/0 pour du calcul

Fonctions logiques — Deux catégories

Fonctions combinatoires

La sortie ne dépend que de l'état actuel des entrées

$$\forall t, s(t) = F(e_0(t), e_1(t), \dots, e_n(t))$$

Fonctions séquentielles

La sortie dépend de l'état actuel des entrées et de leur passé

$$s(t) = F(e_0(t), e_1(t), \dots, e_n(t), e_0(t - t_1), e_1(t - t_1) \dots)$$

Fonctions logiques — Représentations

Plusieurs représentations équivalentes

Table de vérité : Liste de toutes les valeurs possibles pour toutes les entrées possibles

Analytique : Équation analytique

Graphique : Symboles normalisés représentant les fonctions élémentaires

Fonctionnelle : Description de la fonction réalisée en langage naturel (ex. *la sortie S vaut 1 si, et seulement si, les deux entrées A et B valent 1*)

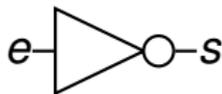
HDL : Description dans un langage dit de description matériel (*Hardware Description Language*) facilement interprétable par un logiciel (ex. SystemVerilog que vous verrez à partir du prochain cours)

Fonctions élémentaires

L'inverseur (NOT)

- La sortie est le complément de l'entrée
- La sortie vaut 1 si et seulement si l'entrée vaut 0

Symbole



Équation

$$s = \bar{e}$$

Table de vérité

e	s
0	1
1	0

Fonctions élémentaires

ET (AND)

- La sortie vaut 1 si et seulement si les deux entrées valent 1
- Forçage à 0 : Si $a = 0$, $s = 0$; si $a = 1$, $s = b$

Symbole



Équation

$$s = a \cdot b$$

Table de vérité

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

Fonctions élémentaires

OU (OR)

- La sortie vaut 0 si et seulement si les deux entrées valent 0
- Forçage à 1 : Si $a = 1$, $s = 1$; si $a = 0$, $s = b$

Symbole



Équation

$$s = a + b$$

Table de vérité

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

Fonctions élémentaires

NON ET (NAND)

- La fonction complémentaire du ET (AND)

Symbole



Équation

$$s = \overline{a \cdot b}$$

Table de vérité

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

Fonctions élémentaires

NON ET (NAND)

- La fonction complémentaire du ET (AND)

Symbole



Équation

$$s = \overline{a \cdot b}$$

Table de vérité

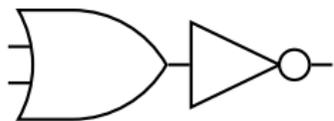
<i>a</i>	<i>b</i>	<i>s</i>
0	0	1
0	1	1
1	0	1
1	1	0

Fonctions élémentaires

NON OU (NOR)

- La fonction complémentaire du OU (OR)

Symbole



Équation

$$s = \overline{a + b}$$

Table de vérité

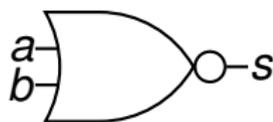
<i>a</i>	<i>b</i>	<i>s</i>
0	0	1
0	1	0
1	0	0
1	1	0

Fonctions élémentaires

NON OU (NOR)

- La fonction complémentaire du OU (OR)

Symbole



Équation

$$s = \overline{a + b}$$

Table de vérité

<i>a</i>	<i>b</i>	<i>s</i>
0	0	1
0	1	0
1	0	0
1	1	0

Équivalence AND/OR

Théorème de De Morgan

$$\overline{a + b} = \bar{a} \cdot \bar{b}$$



$$\overline{a \cdot b} = \bar{a} + \bar{b}$$



Comment résoudre les exercices ?

- Plusieurs formes possibles pour une même fonction
- Méthode 1 : à partir de la table de vérité
 - Trouver quand la fonction vaut 1
 - Trouver quand la fonction vaut 0
- Méthode 2 : à partir des spécifications
 - Reformuler pour faire apparaître des mots-clés (si, et, ou...)

Comment résoudre les exercices ?

Exemple

- Égalité : on veut une porte dont la sortie s vaut 1 si et seulement si les deux entrées (a et b) valent la même chose

Comment résoudre les exercices ?

Exemple

- Égalité : on veut une porte dont la sortie s vaut 1 si et seulement si les deux entrées (a et b) valent la même chose
- Reformulation : $s=1$ si ($(a=1$ et $b=1)$ ou $(a=0$ et $b=0)$)

Comment résoudre les exercices ?

Exemple

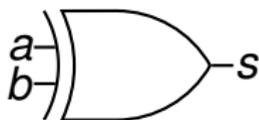
- Égalité : on veut une porte dont la sortie s vaut 1 si et seulement si les deux entrées (a et b) valent la même chose
- Reformulation : $s=1$ si ($a=1$ et $b=1$) ou ($a=0$ et $b=0$))
- $s = a \cdot b + \bar{a} \cdot \bar{b}$

Fonctions élémentaires

OU exclusif (XOR)

- La sortie vaut 1 si une et seulement une des entrées est à 1
- Forçage au complémentaire : si $a = 1$, $s = \bar{b}$; si $a = 0$, $s = b$

Symbole



Équation

$$s = a \oplus b$$
$$s = a \cdot \bar{b} + \bar{a} \cdot b$$

Table de vérité

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

Fonctions élémentaires

NON OU exclusif (XNOR)

- La sortie vaut 1 si les deux entrées sont identiques
- C'est la fonction complémentaire du XOR

Symbole



Équation

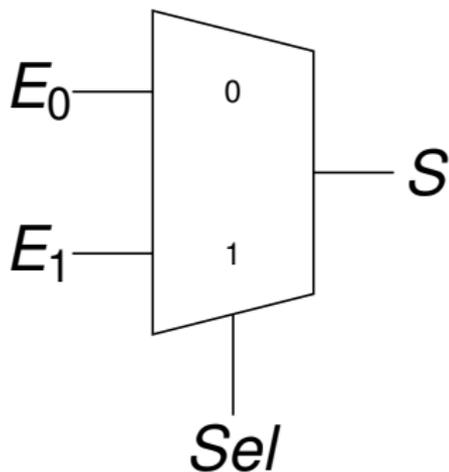
$$s = \overline{a \oplus b}$$
$$s = a \cdot b + \bar{a} \cdot \bar{b}$$

Table de vérité

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

Le multiplexeur

La fonction d'aiguillage

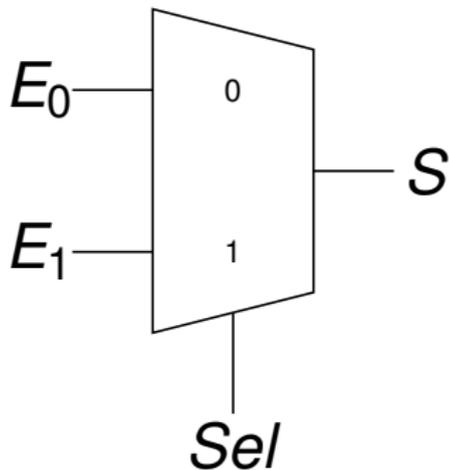


Sel	E_1	E_0	S
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Quelle est l'équation de cette fonction ?

Le multiplexeur

La fonction d'aiguillage

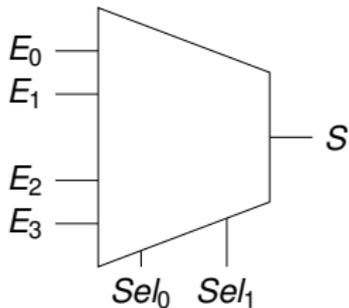


<i>Sel</i>	E_1	E_0	S
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$S = Sel \cdot E_1 + \overline{Sel} \cdot E_0$$

Le multiplexeur

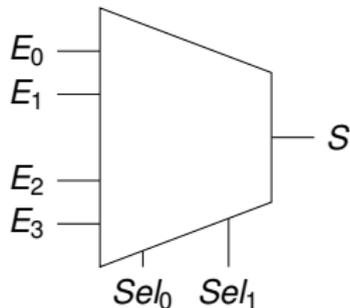
Fonction d'aiguillage à quatre entrées



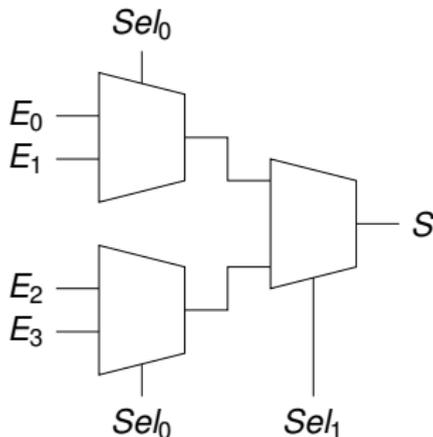
4 entrées \Rightarrow 2 entrées de sélection

Le multiplexeur

Fonction d'aiguillage à quatre entrées



4 entrées \Rightarrow 2 entrées de sélection



Peut être réalisé à partir de 3 multiplexeurs 2 vers 1

$$S = \overline{Sel_1} \cdot \overline{Sel_0} \cdot E_0 + \overline{Sel_1} \cdot Sel_0 \cdot E_1 + Sel_1 \cdot \overline{Sel_0} \cdot E_2 + Sel_0 \cdot Sel_1 \cdot E_3$$

Le multiplexeur

Généralisation

Pour un multiplexeur à n entrées ($n = 2^p$ étant une puissance de 2) :

- Il faut $p = \log_2(n)$ entrées de sélection
- Il peut être réalisé avec $n - 1$ multiplexeurs à 2 entrées organisés en p couches

Le décodeur

- n entrées et 2^n sorties
- Toutes les sorties valent 0 sauf une
- Le numéro de la sortie valant 1 est sélectionné par les entrées
- Exemple décodeur 2 vers 4 :

E_1	E_0	S_0	S_1	S_2	S_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Plan

Introduction

Logique booléenne

Représentation des nombres

Opérateurs arithmétiques

Notion de temps de propagation

Exercices

Les nombres entiers naturels

Un entier positif N dans une base b se représente par un vecteur $(a_{n-1}, a_{n-2}, \dots, a_1, a_0)$ tel que :

$$N = a_{n-1} \cdot b^{n-1} + a_{n-2} \cdot b^{n-2} + \dots + a_1 \cdot b^1 + a_0 \cdot b^0$$

Où :

- a_i appartient à un ensemble de b symboles valant de 0 à $b - 1$
- a_{n-1} est le chiffre le plus significatif
- a_0 est le chiffre le moins significatif

Les nombres entiers naturels

Bases rencontrées couramment

- $b = 10$
 - Représentation *Décimale*
 - $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $b = 16$
 - Représentation *Hexadécimale*
 - $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
- $b = 8$
 - Représentation *Octale*
 - $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$
- $b = 2$
 - Représentation *Binaire*
 - $a_i \in \{0, 1\}$ est un bit (binary digit)

Les nombres entiers naturels

En binaire

Un entier positif N en base 2 se représente par un vecteur $(a_{n-1}, a_{n-2}, \dots, a_1, a_0)$ tel que :

$$N = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_1 \cdot 2^1 + a_0 \cdot 2^0$$

Où :

- a_i est un bit et appartient à un ensemble de 2 symboles valant 0 ou 1
- a_{n-1} est le bit le plus significatif (MSB : *Most Significant Bit*)
- a_0 est le bit le moins significatif (LSB : *Least Significant Bit*)

Les nombres entiers naturels

Exemples

- Représentez $54_{(10)}$ en binaire et en hexadécimal

Les nombres entiers naturels

Exemples

- Représentez $54_{(10)}$ en binaire et en hexadécimal
- $54_{(10)} = 3 \cdot 16^1 + 6 \cdot 16^0 = 36_{(16)}$
- $54_{(10)} = 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 110110_{(2)}$

Les nombres entiers naturels

Conversion binaire \leftrightarrow hexadécimal

$$N = \sum_{i=0}^{n-1} a_i \cdot 2^i$$

Profitons du fait que $2^4 = 16$ (on suppose également que N est représenté sur un nombre de bits multiple de 4)

$$N = \sum_{k=0}^{n/4-1} (a_{4k+3} \cdot 8 + a_{4k+2} \cdot 4 + a_{4k+1} \cdot 2 + a_{4k}) \cdot 16^k$$

Passer à une représentation hexadécimale permet d'avoir une représentation compacte

Exemples binaire ↔ hexadécimal

- Représentez $254_{(10)}$ en hexadécimal
 - En déduire la représentation en binaire
- Représentez $72_{(10)}$ en binaire
 - En déduire la représentation en hexadécimal

Représentation binaire

Physiquement, le nombre de bits utilisé pour représenter un nombre ne peut être infini

En utilisant n bits :

- Il y a 2^n valeurs représentables
- On ne peut représenter que les nombres dans l'intervalle $[0, 2^n - 1]$
- L'arithmétique sur ces représentations est faite modulo 2^n

Les nombres entiers relatifs

Exemple sur 4 bits

- Avec 4 bits on peut représenter les nombres allant de 0 à $15 = 2^4 - 1$
- L'arithmétique se fait alors modulo $2^4 = 16$
- Par exemple :
 - $15 + 1 = 0$
 - $0 - 1 = 15$

Décimal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Les nombres entiers relatifs

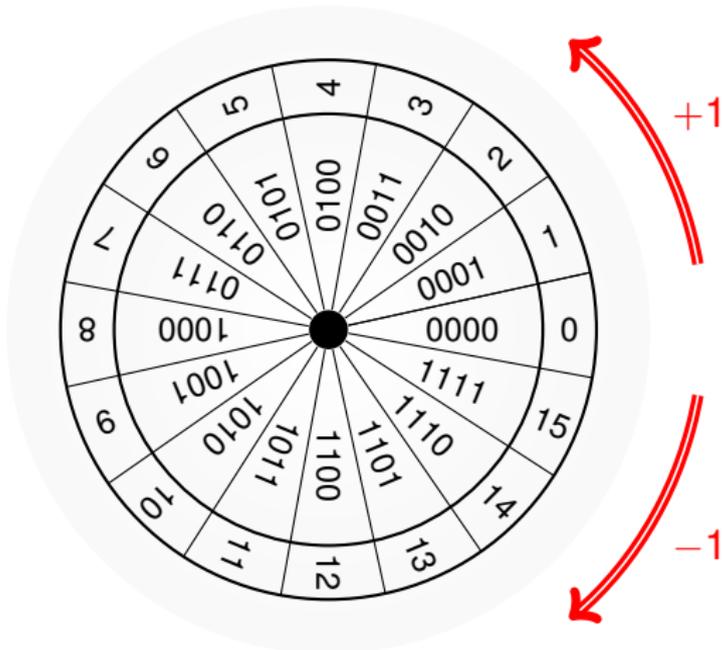
Exemple sur 4 bits

- Avec 4 bits on peut représenter les nombres allant de 0 à $15 = 2^4 - 1$
- L'arithmétique se fait alors modulo $2^4 = 16$
- Par exemple :
 - $15 + 1 = 0$
 - $0 - 1 = 15$
- Comment représenter les nombres signés tout en conservant le même comportement ?

Décimal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

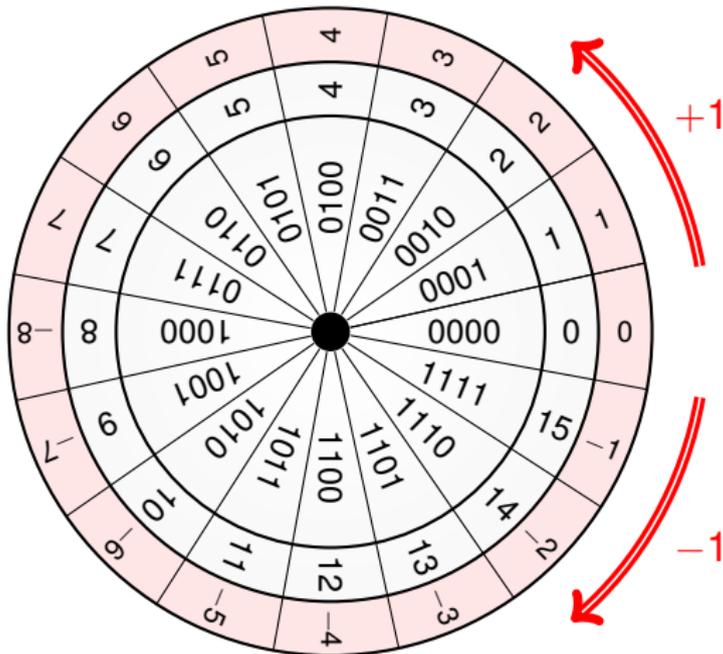
Les nombres entiers relatifs

Exemple sur 4 bits



Les nombres entiers relatifs

Exemple sur 4 bits



Les nombres entiers relatifs

Complément à 2 (CA2)

- C'est une **interprétation** du mot binaire
- Le bit de poids fort représente le signe (0 = nombre positif, 1 = nombre négatif)
- Calcul modulo 2^n , c'est-à-dire $0 \equiv 2^n \Rightarrow -1 \equiv 2^n - 1$
- Permet de représenter les nombres dans l'intervalle $[-2^{n-1}, 2^{n-1} - 1]$
- Un nombre $A = a_{n-1}a_{n-2} \dots a_0$ en CA2 sur n bits :

$$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$

Les nombres entiers relatifs

Exemple sur 4 bits

- Avec 4 bits on peut représenter les nombres allant de -8 à 7 ($[-2^3, 2^3 - 1]$)

Décimal	Binaire	Signé
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	-8
9	1001	-7
10	1010	-6
11	1011	-5
12	1100	-4
13	1101	-3
14	1110	-2
15	1111	-1

Les nombres entiers signés

Exemples

- $-3_{(10)}$ en CA2 =?
- $32_{(10)}$ en non signé =?
- $32_{(10)}$ en CA2 =?

Les nombres entiers signés

Exemples — Extension de signe

- $7_{(10)}$ en CA2 sur 4 bits =?
- $7_{(10)}$ en CA2 sur 5 bits =?
- $7_{(10)}$ en CA2 sur 6 bits =?
- $-7_{(10)}$ en CA2 sur 4 bits =?
- $-7_{(10)}$ en CA2 sur 5 bits =?
- $-7_{(10)}$ en CA2 sur 6 bits =?
- En déduire la règle d'extension

Les nombres entiers signés

Extension de signe pour le CA2

Soit $N = (a_{n-1}, a_{n-2}, \dots, a_0)_{\text{CA2}/n}$ un entier relatif représenté en CA2 sur n bits. Comment représenter N sur un $n + 1$ bits ?

$$\begin{aligned} N &= -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i \\ &= -a_{n-1}2^{n-1} \cdot (2 - 1) + \sum_{i=0}^{n-2} a_i 2^i \\ N &= -a_{n-1}2^n + \sum_{i=0}^{n-1} a_i 2^i \end{aligned}$$

D'où $N = (a_{n-1}, a_{n-1}, a_{n-2}, \dots, a_0)_{\text{CA2}/n+1}$
On a dupliqué le bit de poids fort, on parle d'extension de signe

Les nombres décimaux en virgule fixe

Un nombre décimal D en base 2 peut être approximé un par vecteur $(a_{n-1}, a_{n-2}, \dots, a_1, a_0, a_{-1}, \dots, a_{-m})$ tel que :

$$D = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_1 \cdot 2^1 + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + \dots + a_{-m} \cdot 2^{-m}$$

Où :

- (a_{n-1}, \dots, a_0) est la partie entière
- (a_{-1}, \dots, a_{-m}) est la partie fractionnaire
- 2^{-m} représente la précision de cette approximation
- Les mêmes opérateurs arithmétiques que ceux des nombres entiers

Les nombres décimaux en virgule fixe

Exemples

- Représentez 0.5, 3.625

Les nombres décimaux en virgule fixe

Exemples

- Représentez 0.5, 3.625
- Représentez 0.6
 - en utilisant 1 bit
 - en utilisant 3 bits
 - en utilisant 5 bits



Plan

Introduction

Logique booléenne

Représentation des nombres

Opérateurs arithmétiques

Notion de temps de propagation

Exercices



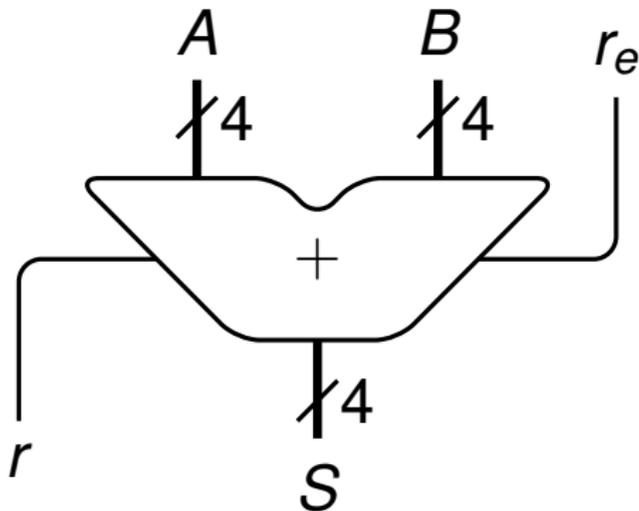
Addition

- Faire une addition en binaire sur 4 bits...

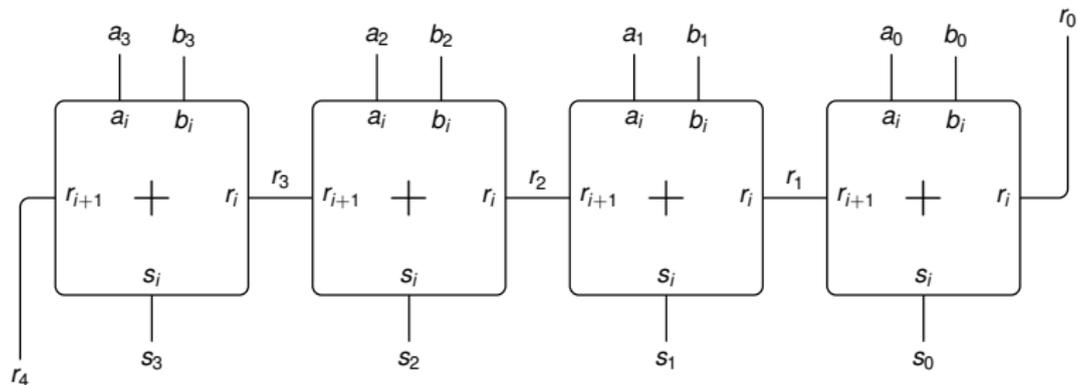
Addition

- Faire une addition en binaire sur 4 bits...
- L'addition peut être décomposée en plusieurs additions élémentaires sur 1 bit

Additionneur à propagation de retenue



Additionneur à propagation de retenue



Additionneur complet sur 1 bit

- Arithmétiquement

$$a_i + b_i + r_i = 2 \cdot r_{i+1} + s_i$$

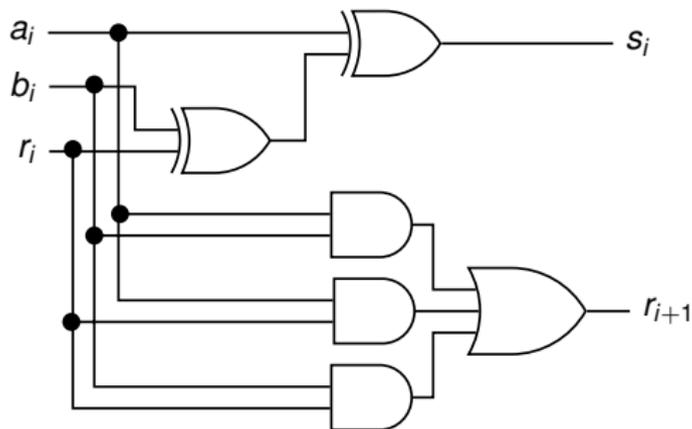
- Table de vérité

a_i	b_i	r_i	r_{i+1}	s_i	Décimal
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	2
1	0	0	0	1	1
1	0	1	1	0	2
1	1	0	1	0	2
1	1	1	1	1	3

Additionneur complet sur 1 bit

$$s_i = a_i \oplus b_i \oplus r_i$$

$$r_{i+1} = a_i \cdot b_i + a_i \cdot r_i + b_i \cdot r_i$$



Dynamique de l'addition

- Sur combien de bit doit être codé le résultat de la somme de deux nombres non signés codés sur n bits ?
- Même question si les nombres sont signés (représentation en complément à 2) ?
- Réaliser en binaire CA2 les additions suivantes :
 - $30 + 8$
 - $30 + (-8)$
 - $(-30) + 8$
 - $(-30) + (-8)$

Dynamique de l'addition

Nombres positifs

- Pour deux nombres A et B représentés sur n bits nous avons :

$$\begin{aligned}A &\leq 2^n - 1 \\B &\leq 2^n - 1 \\A + B &\leq 2^{n+1} - 2 < 2^{n+1}\end{aligned}$$

- Donc $A + B$ est toujours représentable sur $n + 1$ bits
- Exemple

$$\begin{array}{r} \\ \\ + \\ \hline = 1 \end{array} \begin{array}{l} (7) \\ (1) \\ (8) \end{array}$$

Dynamique de l'addition

Nombres en CA2

- Pour deux nombres A et B représentés en CA2 sur n bits nous avons :

$$\begin{aligned} -2^{n-1} &\leq A \leq 2^{n-1} - 1 \\ -2^{n-1} &\leq B \leq 2^{n-1} - 1 \\ -2^n &\leq A + B \leq 2^n - 2 < 2^n \end{aligned}$$

- Donc $A + B$ est toujours représentable sur $n + 1$ bits

Dynamique de l'addition

Exemples en CA2

				non signé	CA2
	1	1	1	7	-1
+	0	0	1	1	1
=	1	0	0	8	0 ou -8?

				non signé	CA2
	0	1	1	3	3
+	0	0	1	1	1
=	1	0	0	4	-4

				non signé	CA2
	1	1	1	7	-1
+	1	0	0	4	-4
=	1	0	1	11	+3 + retenue?

- En CA2 l'interprétation de la retenue n'est pas la même que pour les nombres non signés

Dynamique de l'addition

Nombre en CA2

- Une solution simple pour toujours avoir le bon résultat est de d'abord étendre les nombres sur un bit de plus puis faire la somme
- La retenue produite au delà du bit ajouté n'est pas prise en compte

$$\begin{array}{rcccc|c} & 1 & 1 & 1 & 1 & -1 \\ + & 1 & 1 & 0 & 0 & -4 \\ \hline = & 1 & 0 & 1 & 1 & -5 \end{array}$$

$$\begin{array}{rcccc|c} & 1 & 1 & 1 & 1 & -1 \\ + & 0 & 0 & 0 & 1 & 1 \\ \hline = & 0 & 0 & 0 & 0 & 0 \end{array}$$

Exercice

- Pour un bit b exprimez “logiquement” en fonction de b le résultat du calcul arithmétique $1 - b$
- A est un nombre signé codé en CA2 sur n bits. Montrez que $-A = \overline{A} + 1$ (ici $+$ est l'addition arithmétique)



Plan

Introduction

Logique booléenne

Représentation des nombres

Opérateurs arithmétiques

Notion de temps de propagation

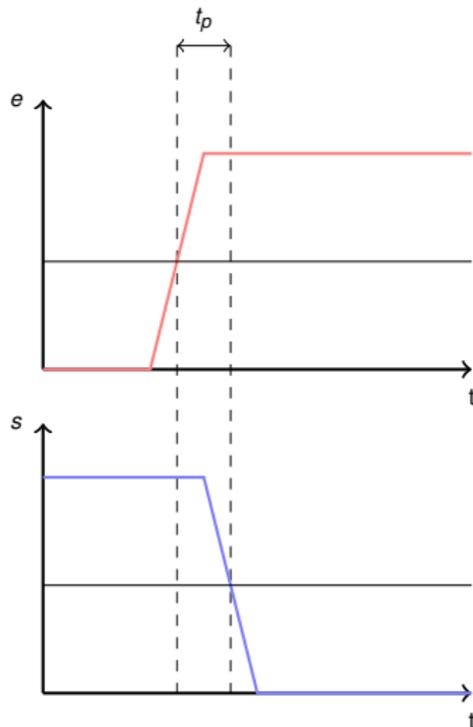
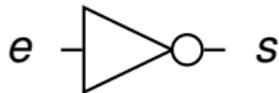
Exercices

Temps de propagation d'une porte

- Les composants logiques respectent les lois de la physique. Les changements d'état ne peuvent pas être instantanés
- Le **temps de propagation** est le temps entre le changement des entrées d'une porte et la stabilisation de la valeur de sa sortie
- La valeur de la sortie ne doit pas être prise en compte durant cette période

Temps de propagation d'une porte

Exemple : Un inverseur



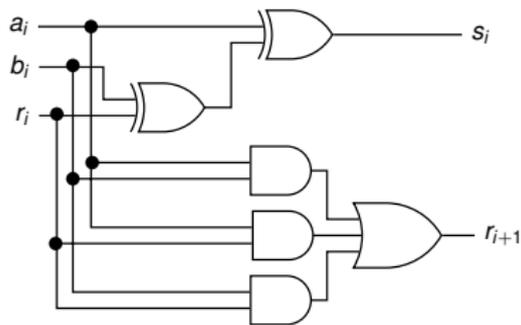
Temps de propagation d'une porte

Règles pour les portes complexes

- Les portes logiques de base sont pré-caractérisées
- Pour une technologie particulière, on connaît le temps de propagation des portes de base
- À partir de ces tables on calcule le temps de propagation des portes complexes en faisant la somme des temps individuels des portes qui se suivent
- Le temps de propagation d'une porte complexe est le temps de propagation du chemin le plus long

Temps de propagation d'une porte

Exemple : Full adder



- Considérons que les portes élémentaires ont toutes un temps de propagation $t_p = 1\text{ ns}$
- Quel est le temps de propagation des entrées vers les 2 sorties ?



Plan

Introduction

Logique booléenne

Représentation des nombres

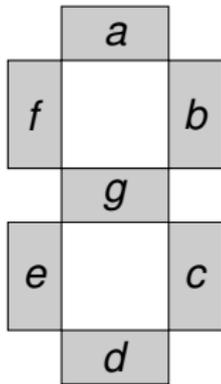
Opérateurs arithmétiques

Notion de temps de propagation

Exercices

Préparation de la prochaine séance

Décodeur 7 segments



- Permet d'afficher de l'hexadécimal (0,1,...,9,A,B...,F)
- L'entrée est sur 4 bits $((i3, i2, i1, i0), 0 \rightarrow F)$
- La sortie est sur 7 bits
 - Chaque bit contrôle un segment
 - Si le bit est à 0 le segment est allumé
- **Donnez les équations de chaque sortie : a à g**