Conception de circuits numériques et architecture des ordinateurs

Frédéric Pétrot

Année universitaire 2022-2023

C1	Codage des nombres en base 2, logique booléenne, portes logiques, circuits combinatoires
C2	Circuits séquentiels
C3	Construction de circuits complexes
C4	Micro-architecture et fonctionnement des mémoires
C5	Machines à état
C6	Synthèse de circuits PC/PO
C7	Optimisation de circuits PC/PO
C8	Interprétation d'instructions - 1
C9	Interprétation d'instructions - 2
C10	Interprétation d'instructions - 3
C11	Introduction aux caches

Intérêt

Pour des raisons technologiques, les ordinateurs utilisent la présence ou de l'absence de courant électrique pour « travailler ».

Les données sont donc codées en base 2, et l'ensemble des opérations que l'on peut effectuer se fait par conséquence en base 2.

Les programmes informatiques se trouvent donc, par un processus complexe, ramenés à une suite d'opérations sur des données binaires.

Plan détaillé du cours d'aujourd'hui

Plan

Représentation des données

Ordinateur ≡ circuit électronique numérique
Données manipulées ≡ informations électriques discrètes

- types de données variés
  - nombres
  - caractères affichables
  - agrégats
- taille des données variée
  - chaînes de caractères
  - entiers bornés
  - ...

Tout est affaire d'interprétation !

Vecteur de bits

Élément de base : le **bit** $\in \{0, 1\}$, (*binary digit*)
Exemple : 01001101111100010001010001111111
Nombre de bits fini
l est le nombre de bits, en pratique $l = 8 \times n$
n est le nombre d'octets (*bytes*), en pratique $n = 2^p$
En programmation informatique (machine « 32 ou 64 bits ») :

- $n = 1$ est appelé un octet ou *byte*
- $n = 2$ est appelé un *short*
- $n = 4$ est appelé un entier ou *int*, ou un *float* s'il représente un nombre pseudo-réel
- $n = 8$ est appelé un *long long*, ou un *double* s'il représente un nombre pseudo-réel

Entiers naturels (*Unsigned integers*)

Soit un vecteur x de n bits :
$(x_{n-1}, x_{n-2}, \ldots, x_1, x_0)$ avec $x_i \in \{0, 1\}$
Valeur de x interprété comme entier naturel :

$$x = \sum_{i=n-1}^{0} x_i \times 2^i$$

Bit le plus à gauche (de poids 2^{n-1}) : *most significant bit (MSB)*
Bit le plus à droite (de poids 1) : *least significant bit (LSB)*

Intervalle des 2^n valeurs représentables :

$$0 \leq x \leq 2^n - 1$$

Exemples

Conversion binaire/décimale des entiers naturels

base 2 $\longrightarrow$ base 10 :
application de la formule

$$x = \sum_{i=n-1}^{0} x_i \times 2^i$$

base 10 $\longrightarrow$ base 2 :

$$\begin{array}{rcl} x & = & v_0 \times 2 + x_0 \\ v_0 & = & v_1 \times 2 + x_1 \\ v_i & = & v_{i+1} \times 2 + x_{i+1} \end{array}$$

Arrêt lorsque $v_j < 2$ et $x = (v_j, x_j, \ldots, x_1, x_0)$
Exemples

Entiers relatifs (*Signed integers*)

Principalement 2 possibilités pour représenter $-x$:

- signe et grandeur

  $$x_{n-1} = \begin{cases} 0 & \text{si } x \geq 0, \\ 1 & \text{si } x < 0. \end{cases} \qquad \text{et} \qquad (x_{n-2}, x_{n-3}, \ldots, x_1, x_0) = |x|$$

  Simple à comprendre
  Mais : deux « 0 » et implantation arithmétique signée $\neq$ non-signée
- complément à 2 (ou plus précisément à 2^{n-1})
  Moins simple à comprendre
  Mais : un seul « 0 » et numération et implantation arithmétique inchangée

Entiers relatifs (*Signed integers*)

Notation dite *en complément à 2*
Utilisée dans 99,9% des circuits intégrés
Soit un vecteur x de n bits :
$(x_{n-1}, x_{n-2}, \ldots, x_1, x_0)$ avec $x_i \in \{0, 1\}$
Valeur de x interprété comme entier relatif :

$$x = -x_{n-1} \times 2^{n-1} + \sum_{i=n-2}^{0} x_i \times 2^i$$

Exemples

Entiers relatifs

Représentation de $-x$ a partir de la représentation de $x = (x_{n-1}, x_{n-2}, \ldots, x_1, x_0)$?

Définition du *complément* de $a \in \{0, 1\}$: $\bar{a} = \begin{cases} 1 & \text{si } a = 0, \\ 0 & \text{si } a = 1. \end{cases}$

$$-x = (\overline{x_{n-1}}, \overline{x_{n-2}}, \ldots, \overline{x_1}, \overline{x_0}) + 1$$

x_{n-1} est appelé le bit de signe (*sign bit)*
Intervalle des 2^n valeurs représentables :

$$-2^{n-1} \leq x \leq 2^{n-1} - 1$$

C'est la représentation **unique** utilisée dans ce cours !

Extension de signe

Affectation d'un vecteur de taille n dans un vecteur de taille m ?

- $n > m$ troncature, perte irrémédiable des poids forts
- $n < m$ exige une *extension de signe*
  Soit $(x_{n-1}, x_{n-2}, \ldots, x_1, x_0)$ à écrire comme
  $(y_{m-1}, y_{m-2}, \ldots, y_n, y_{n-1}, y_{n-2}, \ldots, y_1, y_0)$
  La valeur est conservée avec :

$$y_{m-1} = x_{n-1}, y_{m-2} = x_{n-1}, ..., y_n = x_{n-1}, y_{n-1} = x_{n-1},$$
$$y_{n-2} = x_{n-2}, \ldots, y_1 = x_1, y_0 = x_0$$

Preuve, non totalement triviale, basée sur le lemme

$$\sum_{i=l-1}^{0} 2^i = 2^l - 1$$

démontrable par récurrence

Nombres à virgule fixe

Partie entière sur *n* bits, partie fractionnaire sur *m* bits :

$x_{n-1}x_{n-2}\dots x_1x_0, y_1y_2\dots y_{m-1}y_m$

Interprété comme :

$$x = \sum_{i=n-1}^{0} x_i \times 2^i + \sum_{i=1}^{m} y_i \times 2^{-i}$$

Exemple :

$$\begin{aligned} 110,011 &= 1\times 2^2 + 1\times 2^1 + 0\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} + 1\times 2^{-3} \\ &= 4 + 2 + 0,25 + 0,125 = 6,375 \end{aligned}$$

Nombreux nombres non représentables !

- Virgule fixe mais aussi « flottante »
- Norme IEEE 754 définie la représentation des flottants et le comportement des opérateurs associés

Autres interprétations

- Champs de bits, indicateurs, booléens, ...
- Caractères/symboles : encodage ASCII ou ISO ou UTF-8.
  Nécessaire pour afficher des messages textuels
- Flux de données encodées : musique (mp3, ogg, ...), vidéo (mpeg2, h264, ...)
- Flux de données archivées/compressées : tar, bzip, ...
- Données chiffrées : aes, rsa, ...
- Instructions : forme finale des programmes
- ...

Toute interprétation nécessite la connaissance du « type » des vecteurs de bits !

Plan

Hexadécimal et octal

Lire du binaire est une misère :
l'hexadécimal (base 16) est la règle
l'octal (base 8) est parfois utilisé

binaire	hexa	octal	binaire	hexa	octal
0000	0	0	1000	8	10
0001	1	1	1001	9	11
0010	2	2	1010	A	12
0011	3	3	1011	B	13
0100	4	4	1100	C	14
0101	5	5	1101	D	15
0110	6	6	1110	E	16
0111	7	7	1111	F	17

Préfixes de quelques langages	notation du langage C :	`0x`, nombre hexadécimal
		`0`, nombre octal
		`0b`, nombre binaire †
	notation du langage VHDL :	`X""`, nombre hexadécimal
		`O""`, nombre octal
		`B""`, nombre binaire
	notation du langage Pyhton :	`0x`, nombre hexadécimal
		`0o`, nombre octal
		`0b`, nombre binaire

†. Extension GCC

Kilo, Mega, Giga - octets

En matériel, $1K \neq 10^3 = 1000$ mais $1K = 2^{10} = 1024$!
Ainsi :

Symbole	valeur	héxa	décimale
$1K$	2^{10}	$0x400$	1024
$1M$	2^{20}	$0x100000$	1048576
$1G$	2^{30}	$0x40000000$	1073741824

Nouvelle unité SI : Kibibyte (KiB) introduite en 2000 [‡]

De même, les $\log$ sont des $\log_2$
Par extension, c'est le cas très souvent en informatique !

‡. Les dénominations KiB, MiB et GiB sont peu usitées en pratique.

Plan

Rappels sur les booléens

$\mathbb{B} = \{0, 1\}$: ensemble des booléens
$x \in \mathbb{B}$: variable booléenne
$\mathbb{B}$ doté d'un opérateur unaire et de 2 opérateurs binaires

Notations utilisées pour le cours :

opérateur	notation « info »	notation « math »
non, unaire, complément	not $x = \bar{x}$	$\neg x$
et, binaire, conjonction	x and $y = x \cdot y$	$x \wedge y$
ou, binaire, disjonction	x or $y = x + y$	$x \vee y$

Interprétation des valeurs :
$0 =$ false et $1 =$ true
Interprétation des fonctions :

 en français **ou** est souvent **ou bien**

Quelques propriétés

associativité

$a + (b + c) = (a + b) + c \quad / \quad a \cdot (b \cdot c) = (a \cdot b) \cdot c$

commutativité

$a + b = b + a \quad / \quad a \cdot b = b \cdot a$

distributivité

$a + (b \cdot c) = (a + b) \cdot (a + c) \quad / \quad a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

absorption

$a + (a \cdot b) = a \quad / \quad a \cdot (a + b) = a$

complementation

$a + \bar{a} = 1 \quad / \quad a \cdot \bar{a} = 0$

Quelques propriétés

idempotence

$a + a = a \quad / \quad a \cdot a = a$

éléments neutres

$a + 0 = a \quad / \quad a \cdot 1 = a$

$a + 1 = 1 \quad / \quad a \cdot 0 = 0$

0 et 1 sont complementaires

$\bar{0} = 1 \quad / \quad \bar{1} = 0$

involution

$\bar{\bar{a}} = a$

Quelques propriétés, suite

Autres propriétés :

Lois de De Morgan

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

$$\overline{\sum_{i=1}^{n} a_i} = \prod_{i=1}^{n} \bar{a}_i \quad / \quad \overline{\prod_{i=1}^{n} a_i} = \sum_{i=1}^{n} \bar{a}_i$$

Principe de dualité : *une loi valide peut être transformée en son dual en échangeant* 0 *avec* 1 *et* $+$ *avec* $\cdot$

Plan

Tables de vérité

Représentation sous forme de *tables de vérité*
Énumération de toutes les combinaisons possibles des entrées
$\Rightarrow$ n variables impliquent 2^n valeurs possibles
Quelques fonctions de deux variables remarquables

x	y	$\bar{x}$	$x \cdot y$	$x + y$	$x \oplus y$	$x \mathbin{\bar{\cdot}} y$	$x \mathbin{\bar{+}} y$	$x \mathbin{\bar{\oplus}} y$
		not x	x and y	x or y	x xor y	x nand y	x nor y	x xnor y
0	0	1	0	0	0	1	1	1
0	1	1	0	1	1	1	0	0
1	0	0	0	1	1	1	0	0
1	1	0	1	1	0	0	0	1

Note : il y a 2^{2^n} fonctions booléennes de n variables

Formes canoniques d'équations booléennes à partir d'une table de vérité

Somme de produits (forme normale disjonctive) :

- 1 en sortie obtenu par produit des littéraux
- fonction obtenue par somme de ces produits

Produit de sommes (forme normale conjonctive) :

- 0 en sortie obtenu par somme des littéraux complémentés
- fonction obtenue par produit de ces sommes

x	y	f	sop	pos
0	0	0		$x + y$
0	1	1	$\bar{x} \cdot y$	
1	0	0		$\bar{x} + y$
1	1	0		$\bar{x} + \bar{y}$

d'où $f = \bar{x} \cdot y = (x + y)(\bar{x} + y)(\bar{x} + \bar{y})$

```
f <= not(x) and y
f <= (x or y) and ((not(x) or y) and (not(x) and not(y)))
```

Schémas utilisés en circuiterie numérique

Porte logique (*logic gate*) : représentation schématique d'un opérateur booléen

Une porte possède :

- des connecteurs d'entrée $i_j, 0 \leq j \leq n-1$
- un connecteur de sortie o
- un comportement $o = f(i_{n-1}, \ldots, i_0)$

Syntaxe VHDL :

Définition

```
ENTITY gate IS
PORT (i0, i1, ... : IN STD_LOGIC;
      o           : OUT STD_LOGIC)
END gate;
ARCHITECTURE behavioral OF gate IS
BEGIN
o <= i0 op i1 op ... ;
END behavioral;
```

Instanciation

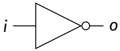
```
mygate : gate
PORT MAP (
i0 => a, i1 => b, ..., o => s
);
```

Portes de base

Portes usuelles à une ou deux entrées

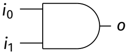
$\overline{i}$

```
o <= not i
```



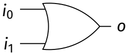
$i_0 \cdot i_1$

```
o <= i0 and i1
```



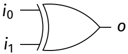
$i_0 + i_1$

```
o <= i0 or i1
```



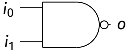
$i_0 \oplus i_1$

```
o <= i0 xor i1
```



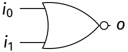
$\overline{i_0 \cdot i_1}$

```
o <= i0 nand i1
```



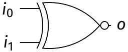
$\overline{i_0 + i_1}$

```
o <= i0 nor i1
```



$\overline{i_0 \oplus i_1}$

```
o <= i0 xnor i1
```



Portes d’arité supérieure à 2 : exemples

Aspects temporels

Porte : dispositif électronique (transistors, R, C)

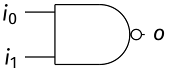
Conséquence :

un calcul logique prend du temps !

Temps $t_{p_{e}(i,o)}$ (propagation) et $t_{t_{e}(o)}$ (transition)

- e front (*egde*) de la sortie : HL, LH
- o sortie dépendante de l'entrée i si t_p
- pris entre $\frac{Vdd}{2}$ de i et $\frac{Vdd}{2}$ de o si t_p
- pris entre $0,9 \times Vdd$ et $0,1 \times Vdd$ si t_t

Pour une technologie 28nm, charge 4 inverseurs



$t_{p_{LH}(i_0,o)}$	10.0ps	$t_{t_{HL}(o)}$	2.1ps
$t_{p_{HL}(i_0,o)}$	9.8ps	$t_{t_{LH}(o)}$	1.8ps
$t_{p_{LH}(i_1,o)}$	9.3ps		
$t_{p_{HL}(i_1,o)}$	10.0ps		

Circuit combinatoire

Circuit combinatoire (*combinational circuit*) :
représentation schématique d'un ensemble de portes ou circuits interconnectés

Un circuit combinatoire possède :

- des connecteurs d'entrée $i_j, 0 \leq j \leq n-1$
- des connecteurs de sortie $o_k, 0 \leq k \leq m-1$
- un comportement $(o_{m-1}, \ldots, o_0) = f(i_{n-1}, \ldots, i_0)$

Circuit combinatoire

Interconnexions légales :

- entrée primaire sur sortie primaire
- entrée primaire sur entrée de porte
- sortie de porte sur entrée(s) de porte(s)
- sortie primaire sur sortie de porte

$n \geq 2$ sorties ensemble $\Rightarrow$ court-circuit !

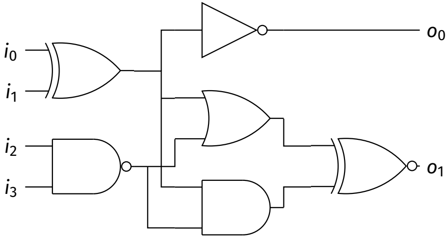
entrée(s) non connectée(s) $\Rightarrow$ valeur(s) de sortie non déterministe(s) !

Schémas utilisés en circuiterie numérique

Réalisation en portes de l'équation :

$$t = (x + \bar{y}).\bar{z}$$

Analyse du circuit :




```
o0 <= not a;
o1 <= c xnor d;
b <= i2 nand i3;
c <= a or b;
a <= i0 xor i1;
d <= a and b;
```