Option Informatique en Spé MP

Devoir surveillé du mardi 17 mars 1998

Additionneurs, systèmes de numération,
parties reconnaissables de $\mathbb{N}^*$

Résumé

On construit un circuit additionneur pour des nombres écrits en base 2, et on montre que sa performance ne peut être améliorée, sauf peut-être en ayant recours à une autre façon de représenter les nombres.

On étudie alors une méthode redondante de représentation des nombres, proposée par Algirdas Avizienis, et permettant de réaliser un circuit additionneur fonctionnant en temps constant.

On s'intéresse ensuite aux langages associés aux parties de $\mathbb{N}^*$, lorsqu'un système de numération a été fixé : après avoir défini la *reconnaissabilité* d'une partie de $\mathbb{N}^*$, on donne quelques exemples de parties reconnaissables et de parties qui ne le sont pas.

Enfin, on étudie une autre méthode de représentation des nombres, due à Édouard Zeckendorf, utilisant la suite de Fibonacci.

Veuillez rédiger chaque partie sur une copie séparée.

Table des matières

Notations, définitions et mises en garde

▶ Soit B un naturel au moins égal à 2. L'*écriture en numération de position en base B* (ou plus simplement *écriture en base B*) du naturel non nul n est l'unique mot $d_p d_{p-1} \dots d_0$ sur l'alphabet $\{0, 1, \dots, B-1\}$ vérifiant $d_p \neq 0$ et $\sum_{0 \leqslant i \leqslant p} d_i B^i = n$.

▶ On note $\mathcal{B} = \{0, 1\}$ l'algèbre de Boole ; on note $\overline{x} = 1 - x$ pour $x \in \mathcal{B}$.

▶ lg désigne le logarithme en base 2, et $\lceil x \rceil$ le plus petit relatif au moins égal au réel x.

▶ La suite de Fibonacci est définie par les relations $F_0 = 1$, $F_1 = 2$ et $F_{n+2} = F_{n+1} + F_n$ pour tout $n \in \mathbb{N}$.

▶ Les portes logiques *élémentaires* sont la porte ET à deux entrées, la porte OU à deux entrées, et la porte NON (à une entrée) ; elles sont décrites dans la figure 1.

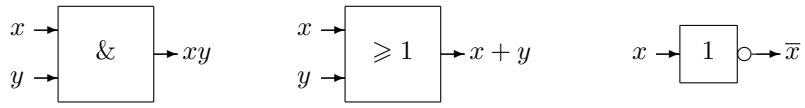


Figure 1: les trois portes logiques élémentaires

On peut admettre un résultat, à condition de le signaler clairement ; en tout état de cause, il est demandé de rédiger les questions dans l'ordre de l'énoncé. Les programmes devront être concis, et suffisamment documentés pour être compréhensibles. L'emploi de références est interdit. Les questions marquées ⋆⋆⋆ sont, à mon sens, plus délicates.

1 L'additionneur *diviser pour régner*

Terms ! names !
— Amaimon sounds well ; Lucifer, well ; Barbason, well ;
yet they are devils' additions, the names of fiends :
but cuckold ! wittol !
— Cuckold ! the devil himself hath not such a name.

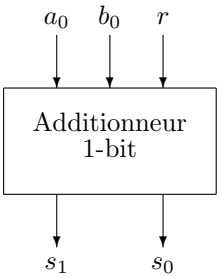
Shakespeare — Merry Wives of Windsor

▶ On étudie dans cette partie deux modèles de circuits logiques *additionneurs n-bits* ; un tel circuit comporte $2n + 1$ entrées : $a = (a_0, \dots, a_{n-1})$ est le premier opérande, $b = (b_0, \dots, b_{n-1})$ est le deuxième opérande et r est la retenue *entrante*. Il comporte $n + 1$ sorties $s_0, \dots, s_n$ et est entièrement décrit par l'équation

$$\sum_{0 \leqslant k < n} 2^k a_k + \sum_{0 \leqslant k < n} 2^k b_k + r = \sum_{0 \leqslant k \leqslant n} 2^k s_k$$

On note que s_n est la retenue *sortante*.

Question 1 • En utilisant des portes logiques élémentaires, construisez un circuit additionneur 1-bit. Vous commencerez par écrire les équations logiques exprimant s_0 et s_1 en fonction de a_0, b_0 et r.



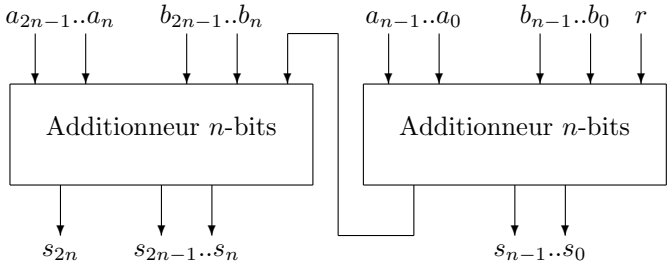
Question 2 • Expliquez comment assembler n additionneurs 1-bit pour obtenir un additionneur n-bits.

Question 3 • Déterminez le nombre $s(n)$ de portes logiques élémentaires nécessaires pour la réalisation de cet additionneur n-bits.

Question 4 • On suppose que le temps de propagation d'un signal logique à travers une porte logique élémentaire est une constante τ indépendante de la porte. Exprimez, en fonction de n et τ, le délai $t(n)$ qui s'écoule entre la présentation des données $a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}, r$ aux entrées du circuit additionneur n-bits que l'on vient de construire, et l'obtention des résultats $s_0, \ldots, s_n$ sur ses sorties.

▶ On se propose de construire un additionneur n-bits, fondé sur le principe *diviser pour régner*, pour diminuer le délai d'obtention du résultat. Bien entendu, comme on ne peut prétendre avoir à la fois le beurre et l'argent du beurre, la complexité du circuit (le nombre de portes logiques requises pour le réaliser) augmentera.

▶ L'étudiant Jean-Marcel MALHABILE propose le schéma suivant, pour réaliser un additionneur $2n$-bits à partir de deux additionneurs n-bits.



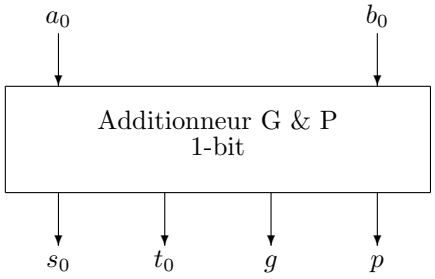
Question 5 • D'après vous, quel gain notre ami Jean-Marcel va-t-il retirer de cette savante construction ?

▶ On part alors de l'idée suivante : chaque additionneur n-bits va calculer en fonction de ses entrées $a = (a_{n-1}, \ldots, a_0)$ et $b = (b_{n-1}, \ldots, b_0)$ *deux* résultats : $s = (s_{n-1}, \ldots, s_0)$ qui correspond au cas où la retenue entrante est égale à 0, et $t = (t_{n-1}, \ldots, t_0)$ qui correspond au cas où la retenue entrante est égale à 1. Le circuit va calculer également deux indicateurs :

1. le bit de *génération* de retenue noté g, qui sera égal à 1 si et seulement si le calcul de la somme $a + b$ génère[1] une retenue ;
2. le bit de *propagation* de retenue noté p, qui sera égal à 1 si et seulement si le calcul de la somme $a + b + 1$ génère une retenue, ce qui revient à dire que le calcul de la somme $a + b$ *propage* une éventuelle retenue entrante.

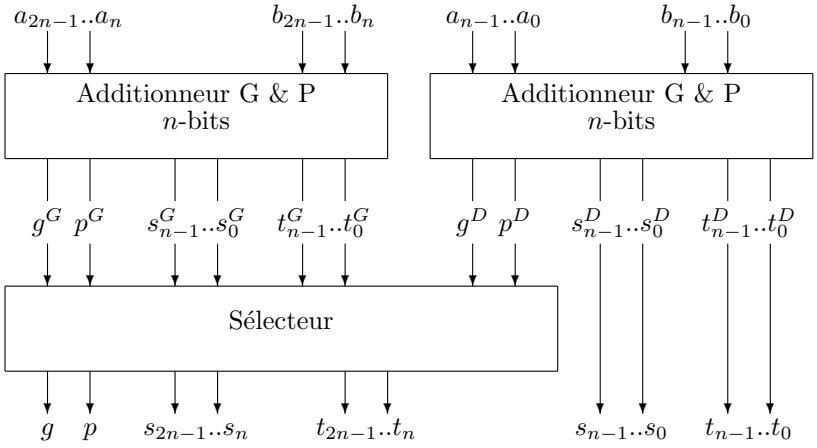
Nous dirons qu'un additionneur construit selon ce principe est un *additionneur à génération et propagation de retenue*, ou, en abrégé, *additionneur G & P*.

Question 6 • En utilisant des portes logiques élémentaires, construisez un circuit additionneur 1-bit avec génération et propagation de retenue. Vous commencerez par écrire les équations logiques exprimant s_0, t_0, g et p en fonction de a_0 et b_0.



▶ Le schéma d'ensemble qui suit décrit la réalisation de principe d'un additionneur $2n$-bits avec génération et propagation de retenue, à partir de deux additionneurs n-bits du même type et d'un *sélecteur* qui reste à décrire. Pour faciliter la lecture, les sorties des deux additionneurs n-bits sont marquées respectivement G (pour *gauche*) et D (pour *droite*).

[1] Le verbe *générer* a été préféré à *produire* pour trois raisons : il est proche du terme anglo-saxon *generate* ; son initiale n'est pas la même que celle de *produire*, ce qui permet d'adopter des notations cohérentes avec la terminologie ; enfin, il est présent dans le *Petit Robert* auquel j'ai recours.



Question 7 • Donnez des formules logiques exprimant g et p en fonction de g^G, p^G, g^D et p^D.

Question 8 • Donnez de même des formules logiques exprimant s_i et t_i en fonction de t_i^G, s_i^G, g^D et p^D, et ce pour $i \in [\![n, 2n-1]\!]$.

▶ On note $T(n)$ le délai d'obtention du résultat avec un additionneur n-bits *diviser pour régner* construit à partir de portes logiques élémentaires, et $S(n)$ le nombre de ces portes utilisées pour réaliser un tel additionneur.

Question 9 • Exprimez $T(2n)$ en fonction de $T(n)$; en déduire une expression simple de $T(2^n)$.

Question 10 • Présentez dans un tableau les valeurs de $t(2^n)$ et $T(2^n)$ pour $n \in [\![0, 6]\!]$ (on prendra $\tau = 1$ pour simplifier).

Question 11 • De la même façon, exprimez $S(2n)$ en fonction de $S(n)$, puis en déduire une expression simple de $S(2^n)$.

Question 12 • Présentez dans un tableau les valeurs de $s(2^n)$ et $S(2^n)$ pour $n \in [\![0, 6]\!]$.

▶ Une fonction f de $\mathcal{B}^n$ dans $\mathcal{B}$ est *complètement dépendante* si, pour tout indice $i \in [\![1, n]\!]$, il existe au moins un n-uplet $x = (x_1, \ldots, x_n)$ tel que

$$f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \neq f(x_1, \ldots, x_{i-1}, \overline{x_i}, x_{i+1}, \ldots, x_n)$$

On considère un circuit combinatoire construit à partir de portes logiques élémentaires, évaluant une telle fonction f complètement dépendante.

Question 13 • Montrez que le délai qui s'écoule entre la présentation des données aux entrées de ce circuit, et l'obtention du résultat à la sortie, est au moins égal à $\lceil \lg n \rceil \tau$.

Question 14 • Montrez que la fonction qui associe au $2n$-uplet

$$(a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1})$$

le bit de poids 2^n de la représentation en base 2 de $\sum_{0 \leqslant i < n} 2^i (a_i + b_i)$ est complètement dépendante.

Question 15 • En déduire qu'un circuit additionneur n-bits, construit à partir de portes logiques élémentaires, et fondé sur la représentation des opérandes en numération de position en base 2, a un temps de calcul qui ne peut être négligeable devant $\ln n$.

2 Numération d'Avizienis

Now the number is even.
Shakespeare — Love's Labour's lost

▶ En 1961, Algirdas Avizienis a proposé une représentation des nombres employant des *chiffres négatifs*. Par exemple, en base 3, il suggère d'utiliser les chiffres $\overline{2}$ (de valeur -2), $\overline{1}$ (de valeur -1), 0, 1 et 2. Ainsi, $2\overline{1}0\overline{2}$ représente $2 \cdot 3^3 - 1 \cdot 3^2 + 0 \cdot 3^1 - 2 \cdot 3^0 = 2 \cdot 27 - 9 - 2 = 43$.

▶ On fixe donc des naturels $B \geqslant 2$ (la *base* de numération) et $q > 0$. L'alphabet utilisé pour écrire les nombres comporte $2q + 1$ chiffres : zéro, les entiers positifs de 1 à q, les entiers négatifs de $\overline{1}$ à $\overline{q}$.

Question 16 • Quel est le plus grand nombre M représentable avec n chiffres ?

Question 17 ⋆⋆⋆ • Quelle relation doit-il exister entre B et q pour que l'on puisse, toujours avec n chiffres, représenter tout élément de $[\![-M, M]\!]$?

Question 18 • Une écriture d'un nombre non nul est *normalisée* lorsque le chiffre le plus à gauche n'est pas nul. Donner une CNS simple liant B et q pour que l'on puisse décider du signe d'un nombre (non nul) en examinant uniquement le chiffre de gauche d'une écriture normalisée de ce nombre.

▶ On suppose la condition précédente satisfaite, ainsi que la relation $2q > B$. On va montrer que l'on peut additionner deux nombres de n chiffres $a = a_{n-1} \dots a_0$ et $b = b_{n-1} \dots b_0$ en temps constant (indépendant de n). Pour $i \in [\![0, n-1]\!]$, on définit deux quantités t_{i+1} et w_i : $t_{i+1} = \overline{1}$ si $a_i + b_i \leqslant -q$, $t_{i+1} = 1$ si $a_i + b_i \geqslant q$, $t_{i+1} = 0$ si $|a_i + b_i| < q$; et $w_i = a_i + b_i - Bt_{i+1}$. On note également $w_n = t_0 = 0$. On définit ensuite $s_i = w_i + t_i$ pour tout $i \in [\![0, n]\!]$.

Question 19 • Montrez que s_i est compris entre $-q$ et q inclus.

Question 20 • Montrez que $\sum_{0 \leqslant i \leqslant n} s_i B^i = a + b$.

Question 21 • Montrez que le calcul de $s = (s_i)_{0 \leqslant i \leqslant n}$ en fonction des entrées $a = (a_i)_{0 \leqslant i < n}$ et $b = (b_i)_{0 \leqslant i < n}$ peut être réalisé en un temps qui ne dépend que de B et q (et ne dépend donc pas de n).

Question 22 • Peut-on employér la représentation d'Avizienis en base 2 ?

▶ Pour les deux questions suivantes, on décide de prendre comme base $B = 4$.

Question 23 • Quelle est nécessairement la valeur de q ?

Question 24 • Réalisez l'addition de $a = \overline{3}102\overline{1}$ et $b = 20\overline{1}1\overline{2}$ (sans effectuer la conversion vers la base dix usuelle).

Question 25 • L'obtention de l'écriture décimale d'un nombre est certainement plus laborieuse à partir de sa représentation d'Avizienis, qu'à partir de sa représentation en base 2. Ceci est-il réellement gênant ?

3 Parties reconnaissables de $\mathbb{N}^*$

▶ Un système de numération étant fixé, on peut associer à chaque partie X de $\mathbb{N}^*$ le langage L_X formé des écritures des éléments de X dans ce système de numération. X est dite *reconnaissable* si L_X est lui-même reconnaissable (par un automate fini). On dit que X est p-reconnaissable lorsqu'elle est reconnaissable dans le système de numération de position à base p. Par exemple, $\mathbb{N}^*$ est 2-reconnaissable puisque $L_{\mathbb{N}^*} = 1(0+1)^*$.

Question 26 • Montrez que $\{2^n \mid n \in \mathbb{N}\}$ est 2-reconnaissable ; vous donnerez une expression rationnelle décrivant le langage associé, et un automate fini le reconnaissant.

Question 27 • Montrez que $\{2^n \mid n \in \mathbb{N}\}$ est 4-reconnaissable ; ici encore, vous donnerez une expression rationnelle décrivant le langage considéré, et un automate fini le reconnaissant.

Question 28 • Montrez qu'une partie X de $\mathbb{N}^*$ est k-reconnaissable si et seulement si elle est k^2-reconnaissable.

Question 29 • Soient a et b deux naturels, $a \neq 0$. On note $E_{a,b}$ l'ensemble des naturels non nuls dont le reste dans la division par a est égal à b : $E_{a,b} = a\mathbb{Z} + b$. Montrez que $E_{a,b}$ est k-reconnaissable, et ce quel que soit $k \geqslant 2$.

▶ Une partie X de $\mathbb{N}^*$ est *ultimement périodique* s'il existe des naturels n_0 et $p > 0$ tels que, si $n \geqslant n_0$ appartient à X, alors $n + p$ appartient aussi à X.

Question 30 • Montrez que toute partie X de $\mathbb{N}^*$ ultimement périodique est k-reconnaissable, et ce quel que soit $k \geqslant 2$.

Question 31 ⋆⋆⋆ • L'ensemble des carrés parfaits est-il 2-reconnaissable ?

▶ L'ensemble de Thue-Morse est la partie $\mathcal{T}$ de $\mathbb{N}^*$ définie par les règles suivantes : $1 \in \mathcal{T}$; si $n \in \mathcal{T}$ alors $2n \in \mathcal{T}$ et $2n+1 \notin \mathcal{T}$; si $n \notin \mathcal{T}$ alors $2n \notin \mathcal{T}$ et $2n+1 \in \mathcal{T}$.

Question 32 • Déterminez les éléments de $[\![1, 10]\!]$ qui sont dans $\mathcal{T}$.

Question 33 • Prouvez que les règles précédentes permettent effectivement de décider si un naturel n non nul appartient à $\mathcal{T}$.

Question 34 • Rédigez en Caml une fonction de type `int -> bool`, qui détermine si un naturel non nul donné appartient à $\mathcal{T}$.

Question 35 • $\mathcal{T}$ est-il 2-reconnaissable ?

4 Numération en base de Fibonacci

Here are only numbers ratified.

Shakespeare — Love's Labour's lost

Question 36 • Montrez que tout naturel non nul peut se décomposer en somme de termes deux à deux distincts de la suite de FIBONACCI. Cette décomposition est-elle unique ?

▶ Soit $u = u_1u_2\dots u_p$ un mot sur l'alphabet $\{0,1\}$. On lui associe le naturel $\varphi(u) = \sum_{1\leqslant k\leqslant p} u_kF_{p-k}$; u est une *écriture en base de Fibonacci* de ce naturel.

▶ Une telle écriture u est dite *normalisée* si $u_1 = 1$ et $u_iu_{i+1} = 0$ pour tout $i \in [\![1, p-1]\!]$; cette deuxième condition revient à dire que la décomposition de n définie par cette écriture ne fait pas intervenir deux termes *consécutifs* de la suite de FIBONACCI. Par exemple, 100010101 est une écriture normalisée en base de FIBONACCI du naturel 67.

Question 37 • Montrez que tout naturel n non nul possède une et une seule écriture normalisée en base de FIBONACCI, que l'on notera $w(n)$.

Question 38 • Déterminez $w(137)$.

Question 39 • Décrivez un algorithme calculant l'écriture normalisée en base de FIBONACCI d'un naturel n non nul donné.

Question 40 • Traduisez cet algorithme par une fonction en Caml de type `int -> string`.

Question 41 • Quels sont les naturels n non nuls dont les écritures normalisées en base 2 et en base de FIBONACCI ont même longueur ?

Question 42 • Montrez que $\mathbb{N}^*$ est reconnaissable en base de FIBONACCI.

Question 43 ⋆⋆⋆ • On note L l'ensemble des mots u sur l'alphabet $\{0,1,\overline{1}\}$ vérifiant $\sum_{1\leqslant k\leqslant |u|} u_kF_k = 0$. Montrez que L est reconnu par l'automate $\mathcal{A}$ représenté figure 2.

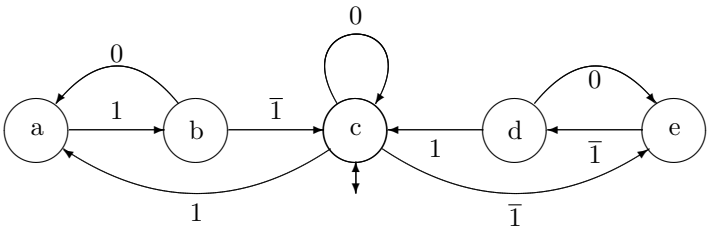


Figure 2: l'automate $\mathcal{A}$

FIN