ES102/PC1 : énoncé

1) To be or not to be… (~15')

1a) Soit deux valeurs binaires *c* et *x* et un multiplexeur 2 vers 1 laissant passer *x* si c=0 et *x'* si c=1. Dessiner le montage correspondant et sa table de Karnaugh *fonctionnelle*. Que reconnaît-on ?

1b) Quel usage de la porte reconnue cela suggère-t-il ?

1c) Compléter les équations suivantes : $0 \oplus x = ?$ $1 \oplus x = ?$ $x' \oplus y = ?$

2) Full adder (~35')

Un FA, alias *Full Adder* (additionneur binaire complet), prend en entrée 3 bits *a*, *b* et *c* et produit en sortie deux bits *s* et *cout*, tel que $2 \cdot \text{cout} \mathbf{+} s = a \mathbf{+} b \mathbf{+} c$ (le signe **+** est mis en gras pour représenter l'addition, évitant la confusion avec le OU logique +). En d'autres termes, un FA exprime en base 2 la somme arithmétique $\sum$ des 3 valeurs binaires *a*, *b* et *c*.

2a) Dresser une table de vérité 1D de $\sum$, *cout* et *s* en fonction de *a*, *b* et *c*.

2b) Donner une représention géométrique 3D de *s* et *cout* en fonction de *a*, *b* et *c*.

2c) Pourquoi *s* et *cout* sont-elles respectivement appelées fonctions *parité* et *majorité* de *a*, *b* et *c* ?

2d) Représenter *s* et *cout* sous forme de tables de Karnaugh *fonctionnelles*, en fonction de *a*, *b* et *c*.

2e) À l'aide des deux représentations précédentes, exprimer *s* et *cout* en fonction de *a*, *b* et *c*, sous forme de formule booléenne disjonctive aussi concise que possible.

2f) Repartant de la forme $\sum\prod\sum$ de *cout* évoquée en marge de la réponse précédente, réétablir la table de Karnaugh fonctionnelle lui correspondant (et constater comme cela est simple).

2g) Montrer que, en fait, $s = a \oplus b \oplus c$.

2h) Traduire directement les formules précédentes en une implantation.

2i) Simuler l'addition des entiers 5 et 7 sur un additionneur à retenue propagée comportant 4 FA.

2j) *Prérequis pour l'exercice suivant*. En tant que somme arithmétique modulo 2, la fonction *parité* se généralise à *n* variables. Montrer qu'elle est égale au OU exclusif entre ses *n* variables.

3) Codes de Gray (~40')

Les 3 premières questions sont à traiter rapidement, au profit des 4 dernières, plus importantes car concernant les concepts de complexité.

Le système de numération binaire classique sur *n* bits est une bijection de l'intervalle entier $[\![0, 2^n-1]\!]$ vers B^n (où $B = \{0, 1\}$) Une telle bijection est appelée *code* sur *n* bits. Mais sa valeur pour un entier particulier est aussi appelée *code* : ainsi, on dit que le code binaire classique sur 4 bits de l'entier 3 est 0011.

3a) Avec le code classique sur n=32 bits, quel est le nombre maximal de bits qui changent du code d'un entier *k* au code de son successeur *k*+1 ? Et en moyenne ?

Exigeant le chargement de capacités, les changements de bits consomment courant et énergie. Il peut être intéressant d'en réduire le nombre dans certaines situations. Il existe justement un code dit *de Gray*, tel qu'un seul bit change du code de chaque entier à celui de son successeur.

3b) Comptant à partir de l'entier 0, représenté par 0···0 (chaîne de *n* '0'*)*, le code de Gray sur *n* bits se construit en complémentant toujours le bit situé le plus à droite possible, sans jamais revenir sur un code (*n*-uplet de bits) déjà exploité. Le construire à la main pour *n*=2, puis 3, puis 4.

Comment passer du code classique d'un entier *k* à son code de Gray, et réciproquement ? Ce problème de codage/décodage (*codec*) n'a rien d'évident a priori. Notons b_i le bit de poids 2^i du code binaire classique de *k*, et g_i celui occupant la même position dans son code de Gray. On considèrera le cas *n*=4 ci-dessous.

3c) Exprimer algébriquement le fait qu'un seul bit change entre deux codes de Gray d'entiers successifs et, sur cette base, observer une relation entre b_0 et les g_i.

3d) Implanter cette relation à l'aide de portes XOR à 2 entrées (OU exclusif), en choisissant la solution la plus rapide. Chaque porte XOR présente un même temps de réponse (alias délai) τ_{XOR}.

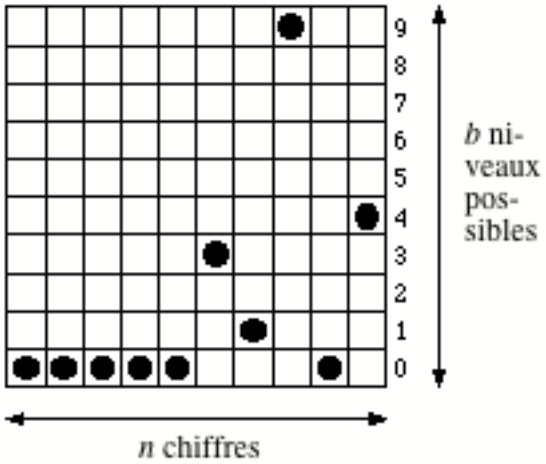
3e) Pour *n* plus grand, comment se comporterait le délai d'obtention de b_0 ?

3f) Exprimer les autres b_i en fonction des g_i. *Indice : examiner la parité des g_i hors g_0*. Implanter avec des portes XOR le décodage Gray vers classique, de façon aussi compacte que possible.

3g) Calculer $b_1 \oplus b_0$ en fonction des g_i. En déduire les relations de codage, les implanter et qualifier les complexités en temps et en espace obtenues.

4) Base optimale pour représenter les nombres (~30')

Considérons un moyen ancestral de représentation des nombres, constitué d'une tablette comme celle ci-contre. Chaque colonne comporte *b* cases, représentant de bas en haut les chiffres de 0 à *b*-1, et l'une de ces cases contient un caillou (*calculus* en latin…) qui indique le chiffre représenté. Cette tablette permet donc de manipuler des nombres de *n* chiffres en base *b*. Ci-contre, c'est donc l'entier 31904 qui est représenté, en base 10.



On mesure le niveau de performance d'une tablette par sa *portée P*, c'est-à-dire le nombre P(b, n) d'entiers qu'elle peut représenter. Par ailleurs, on mesure son *coût C* par son nombre C(b, n) de cases.

4a) Exprimer les fonctions *C* et *P* en fonction de *b* et *n*. Comparer le cas de la figure ci-dessus, où *b*=10 et *n*=10, avec le cas *b*=5 et *n*=20. Pour mémoire, $\log_{10}(5) \approx 0{,}7$.

4b) Comment choisir la base *b* pour minimiser le coût *C* à portée *P* donnée ? Pour le savoir, exprimer *C* en fonction de *P* et *b*, en faisant disparaître *n*. Finalement, que choisir ?

4c) Selon ce qui précède, la base 2 ne serait pas la meilleure. Mais le raisonnement mené ne vaut que pour un modèle de coût bilinéaire C(b,n)=b·n. Une version électronique de notre tablette comporterait typiquement *n* dispositifs présentant chacun *b* états possibles. Si la proportionnalité de *C* par rapport à *n* est plausible, elle l'est moins par rapport à *b*. Considérons donc un modèle de coût plus général C(b,n)= n·f(b), où *f* est une fonction inconnue. Pour quelle propriété de *f* la base 2 est-elle préférable à la base 3 ?