Architecture des ordinateurs : TD2

Université de Tours

`Département informatique de Blois`

Logique booléenne et circuits combinatoires

*
* *

Problème 1

On veut concevoir un circuit combinatoire permettant de comparer deux nombres A et B de 4 bits, $A = \langle a_3 a_2 a_1 a_0 \rangle_2$ et $B = \langle b_3 b_2 b_1 b_0 \rangle_2$.
Le circuit possède deux sorties :

$$G(A,B) = \begin{cases} 1 & \text{Si } A > B \\ 0 & \text{Sinon} \end{cases} \quad \text{et} \quad E(A,B) = \begin{cases} 1 & \text{Si } A = B \\ 0 & \text{Sinon} \end{cases}$$

1. Soient a et b deux bits. Soit un circuit à deux sorties :

$$g(a,b) = \begin{cases} 1 & \text{Si } a > b \\ 0 & \text{Sinon} \end{cases} \quad \text{et} \quad e(a,b) = \begin{cases} 1 & \text{Si } a = b \\ 0 & \text{Sinon} \end{cases}.$$

   Donner l'expression logique de g et e puis dessiner le circuit correspondant à un comparateur 1 bit.

2. En utilisant le circuit de la question précédente ainsi que des portes logiques $\vee, \wedge$ (éventuellement à entrées ≥ 3) et $\neg$, proposer un circuit permettant de comparer deux nombres de 4 bits.

Problème 2

On cherche à représenter la fonction implémentant un *hidden bit*.
Cette fonction prend en entrée k valeurs booléennes et retourne une valeur booléenne. Soient k entrées binaires $a_1, a_2, ..., a_k$ et soit $s = \text{card}(\{i | a_i = 1\})$.
La sortie S est alors égale à 0 si $s = 0$ et elle est égale à a_s si $s \in [\![1, k]\!]$.

1. Dresser la table de vérité de S pour $k = 3$.
2. Simplifier S à l'aide des tableaux de Karnaugh.
3. Dessiner le circuit combinatoire correspondant. À quel autre circuit celui-ci correspond-il ?

Problème 3

L'opérateur $Nand$ noté $\uparrow$ est un opérateur très utilisé en électronique et dans la réalisation des microprocesseurs car il forme un système complet de connecteurs à lui seul.

1. Montrer que $x \oplus y = [(x \uparrow y) \uparrow x] \uparrow [y \uparrow (x \uparrow y)]$. On rappelle que l'opérateur $\oplus$ désigne le $OU\ exclusif$ (ou Xor).

2. Sur la modélisation de $\oplus$:
   (a) Proposer un circuit bien modélisé de l'opérateur $\oplus$ à l'aide du système d'opérateurs $\{\vee, \wedge, \neg\}$.
   (b) Expliquer pourquoi la modélisation 2.(a) n'est pas satisfaisante. Proposer un circuit logique à l'aide de l'opérateur Nand. Pourquoi cette modélisation est meilleure ?

Problème 4

Soit la fonction booléenne P de n variables booléennes définie telle que :

$$P(x_1, ..., x_n) = \bigoplus_{i=1}^{n} x_i$$

Où $\oplus$ désigne l'opérateur Ou exclusif (XOR). On rappelle que l'opérateur $\oplus$ est associatif et commutatif, ainsi P s'écrit aussi comme $P(x_1, ..., x_n) = x_1 \oplus x_2 \oplus ... \oplus x_n$.
Cette fonction est appelée fonction de *clé de parité*.

1. Montrer que la valeur de la fonction P est 1 si et seulement s'il y'a, parmi $x_1, ..., x_n$, un nombre impair de variables valant 1.
2. On considère x, un vecteur binaire tel que $x = \langle x_1, ..., x_n \rangle$. Par simplicité, on supposera que $n = 2^k$.
   (a) Écrire une fonction `cle_parite` en C ou en Java utilisant les opérateurs ^ (ou exclusif) et $\gg$ (décalage à droite) qui implémente la fonction $P(x)$.
   (b) Écrire une fonction `cle_parite_log` en C ou en Java, plus efficace, permettant de calculer $P(x)$ s'appuyant sur l'exemple ci-dessous.
   Exemple : pour 8 bits stockés dans la donnée x, on calcule :
   $y = 4$ forts $\oplus$ 4 faibles de $x = \langle x_7 \oplus x_3, x_6 \oplus x_2, x_5 \oplus x_1, x_4 \oplus x_0 \rangle$
   $z = 2$ forts $\oplus$ 2 faibles de $y = \langle y_3 \oplus y_1, y_2 \oplus y_0 \rangle$
   $t = z_1 \oplus z_0$ puis on retourne t.
3. Combien d'étapes de calcul sont nécessaires pour calculer $P(x)$ avec la méthode 2.b si x est codé sur 16 bits ? Sur 2^k bits ? Combien d'étapes avec la méthode 2.a ? Pourquoi qualifie t-on la méthode 2.b de logarithmique ?
4. Appliquer la méthode de la question 2.b pour les valeurs $x = 2^8 - 1$ et $x = -15$.
5. Traduire sur papier les méthodes 2.a et 2.b du calcul de la clé de parité $P(x)$ sous forme de circuits logiques à 8 entrée et 1 sortie, avec des portes XOR.

Problème 5

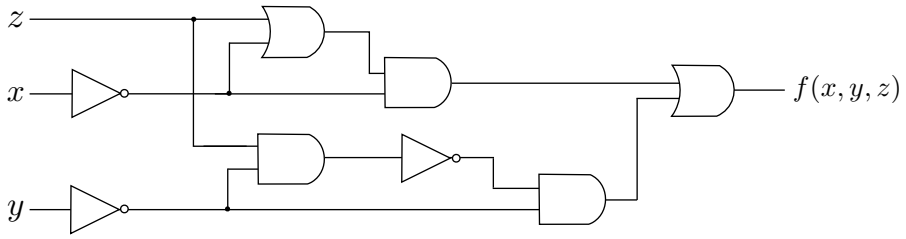
On dit qu'une fonction booléenne est sous *forme minimale* si elle se réduit au système de connecteurs $\{\Rightarrow, 0\}$. Où 0 représente la valeur FAUX d'arité 0.
On veut montrer que toute formule $f \in \mathcal{B}$ admet une forme minimale équivalente.

1. Montrer que les opérateurs $\neg$ et $\vee$ peuvent être exprimés à l'aide du système $\{\Rightarrow, 0\}$.
2. Donner la table de vérité de la formule $(p \Rightarrow (q \Rightarrow 0)) \Rightarrow 0$ et la comparer à $p \wedge q$.
3. Que pouvez-vous déduire à l'aide des questions précédentes ?
4. Déduire des questions précédentes une fonction `min` qui transforme toute fonction f de la logique booléenne en une fonction équivalente sous forme minimale.

Problème 6

Démontrer les assertions vraies. Donner un contre-exemple ou justifier soigneusement les assertions fausses.

1. L'unique connecteur unaire existant en logique booléenne est la négation $\neg$.
2. Le système de connecteurs $\{\oplus, \vee, 1\}$ est complet. On précise que 1 est la constante VRAI d'arité 0.
3. Il est vrai que $x \uparrow y = [(x \downarrow x) \downarrow (y \downarrow y)] \downarrow [(x \downarrow x) \downarrow (y \downarrow y)]$.
4. La fonction logique $f(x, y, z) = \neg x \wedge y$ est identique au circuit logique ci-dessous.



5. Soient x_3, x_2, x_1 et x_0 quatre variables booléennes. On note $x = \langle x_3 x_2 x_1 x_0 \rangle_2$. On pose la fonction booléenne $\varphi(x) = \begin{cases} 1 & \text{si } x < \langle 1001 \rangle_2 \\ 0 & \text{sinon} \end{cases}$.
   Il est vrai que $\varphi(x) = \neg x_3 \vee (\neg x_2 \wedge \neg x_1)$.