Electronique numérique
Logique combinatoire

Remarque : tous les exercices ne seront pas forcément traités en TD.

1 Equations logiques de circuits de base

1.1 Additionneur

1. Etablir la table de vérité d'un *demi-additionneur* logique, puis les deux équations logiques associées : ce circuit additionne simplement 2 entrées booléennes a et b (donc *sans retenue entrante*). Il calcule deux sorties : la somme et la retenue sortante. Dessiner le circuit.

Solution

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

On trouve : $\begin{cases} s = a \oplus b \\ c_o = a.b \end{cases}$

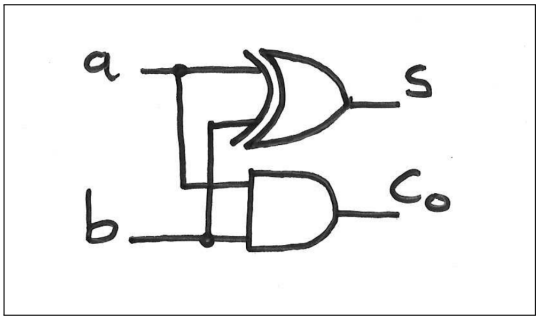


FIGURE 1 – Circuit ('netlist') du demi-additionneur 1 bit

2. Etablir la table de vérité d'un *additionneur complet* pour deux opérandes 1 bit. Ce dernier tient désormais compte d'une retenue entrante. Eta-

blir les équations simplifiées des sorties, puis dessiner le circuit correspondant. Au préalable on démontrera que :

$$\overline{x \oplus y} = x \oplus \overline{y}$$

Solution

On écrit la table de vérité, comme précédemment :

a	b	c_i	s	c_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

On trouve :

$$s = \overline{a}.\overline{b}.c_i + \overline{a}.b.\overline{c_i} + a.\overline{b}.\overline{c_i} + a.b.c_i$$
$$c_o = \overline{a}.b.c_i + a.\overline{b}.c_i + a.b + \overline{c_i} + a.b.c_i$$

On peut simplifier ces deux équations :

$$s = \overline{a}.(b \oplus c_i) + a.(b \oplus \overline{c_i})$$
$$c_o = a.b + c_i.(a \oplus b)$$

La première équation (la somme) peut s'écrire autrement, ce qui nous permettra de reconnaître les équations du demi-additionneur. Pour cela on démontre que $\overline{x \oplus y} = x \oplus \overline{y}$. On a :

$$\begin{aligned}
\overline{x \oplus y} &= \overline{x.\overline{y} + \overline{x}.y} \\
&= \overline{x.\overline{y}}.\overline{\overline{x}.y} \\
&= (\overline{x} + y).(x + \overline{y}) \\
&= \overline{x}.x + \overline{x}.\overline{y} + x.y + y.\overline{y} \\
&= 0 + \overline{x}.\overline{y} + x.y + 0 \\
&= x \oplus \overline{y}
\end{aligned}$$

On pose $\alpha = b \oplus c_i$. On trouve que :

$$\begin{aligned} s &= \overline{a}.(b \oplus c_i) + a.(b \oplus \overline{c_i}) \\ &= \overline{a}.\alpha + a.\overline{\alpha} \\ &= \overline{a} \oplus \alpha \\ &= a \oplus b \oplus c_i \end{aligned}$$

En résumé :

$$\begin{aligned} s &= a \oplus b \oplus c_i \\ c_o &= a.b + c_i.(a \oplus b) \end{aligned}$$

3. Montrer qu'un additionneur complet "1 bit" peut réutiliser le demi-additionneur de la question 1.

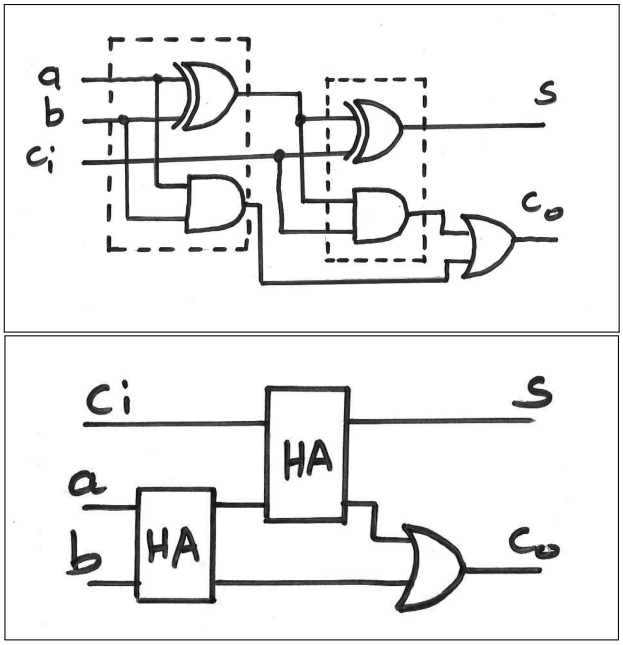


FIGURE 2 – Circuit ('netlist') de l'additionneur 1 bit complet (full-adder 1 bit)

4. Dessiner un additionneur 4 bits en réutilisant l'additionneur 1 bit.

On propose ci-dessous 2 solutions : la première ne permet pas l'utilisation de retenue entrante, tandis que la seconde utilise une telle retenue. Dans le premier cas, il n'est alors pas nécessaire de recourir à un full adder. La seconde solution est parfaitement régulière en terme d'aboutement, constituée de full-adder 1 bit.

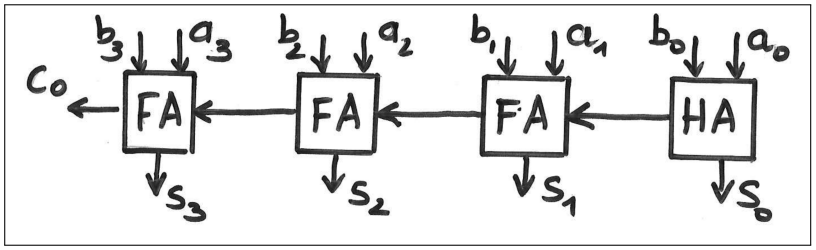


FIGURE 3 – Additionneur 4 bits sans retenue entrante

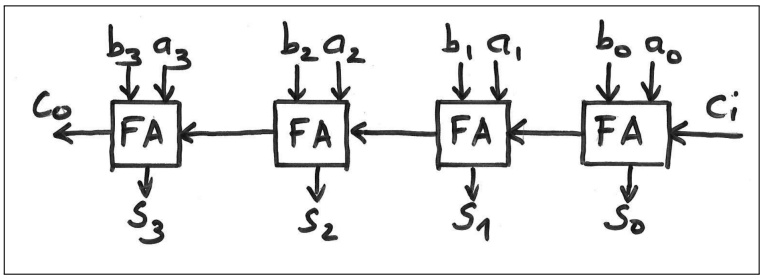


FIGURE 4 – Additionneur 4 bits avec retenue entrante

5. Quel est le chemin critique de ce dernier circuit ? On rappelle que le chemin critique est le parcours le plus long entre une entrée et une sortie combinatoire. On le mesurera en nombre de portes logiques élémentaires traversées.

Il est précisé dans l'exercice que le chemin critique peut être mesuré en nombre de portes "abstraites" traversées, donc sans unité physique. Dans la réalité, chacune des portes logiques sera pré-caractérisée par des temps de traversée donnés par le constructeur (Intel, IBM, TSMC.,..) et il faudra donc sommer des *temps*. L'exercice vous simplifie un peu la vie : des outils d'*analyse de timing* permettent d'automatiser cette tâche fastidieuse. Les *synthétiseurs logiques* contiennent également un tel algorithme.

Nommons les étages de 0 à 3. Nous devons gérer deux cas distincts correspondants aux 2 solutions précédentes : avec ou sans retenue entrante.

1. Sans retenue entrante : dans ce cas, l'additionneur de l'étage 0 a un chemin critique à partir des entrées. Ce chemin traverse 2 demi-additionneurs, suivi d'un OU logique. Ce qui fait un total de 3 portes logiques. Les autres étages de 1 à 3 ont un chemin critique qui se situe entre l'entrée C_{in} et C_{cout}, soit 1 demi-additionneur , suivi d'un OU, ce qui fait, pour chacun des étages,

un chemin critique de 2. Pour les 3 étages cela donne donc 6 portes traversées. Le résultat est donc $3 + 6 = 9$ portes pour l'ensemble.

2. Avec retenue entrante : le même raisonnement conduit à *bf 7 portes traversées*, sur le chemin critique.

remarque : si on place un 0 (définitif) en entrée du premier cas traité, le chemin réel traversé sera 7. Par contre, structurellement, c'est 9 portes qui existent. Ces 9 portes constituent un "faux chemin combinatoire critique".

1.2 Multiplexeur

La fonction (combinatoire) d'un multiplexeur est de sélectionner, grâce à un signal de contrôle, un signal d'entrée parmi plusieurs et de transmettre sa valeur sur une sortie unique.

1. Dessiner le symbole communément admis pour représenter un multiplexeur à 2 entrées binaires (chacune codée sur un 1 bit).
2. Calculer l'équation d'un multiplexeur à 2 entrées 1 bit.
3. Lorsque les données d'entrées sont codées sur $n > 1$ bits, chacun des bits du signal multiplexé suit évidemment un chemin similaire. Dessiner un tel multiplexeur à 2 entrées, pour $n = 2$.

Solution : on parle de multiplexeur 2-vers-1 pour se référer au nombre de voies en entrées. La solution est donnée sur le schéma **??**.

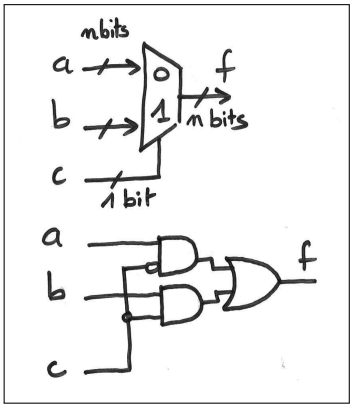


FIGURE 5 – Multiplexeur 2 vers 1, codé sur 1 bit : symbole commun et circuit

2 Formes canoniques

Toute fonction booléenne F de plusieurs variables peut s'écrire sous deux formes spéciales, appelées *formes canoniques* : il ne s'agit ni plus ni moins que de l'expression de F sous forme de :

- (**première forme**) : somme de produits de chaque variable *ou de son complément*
- (**seconde forme**) : produit de sommes de chaque variable *ou de son complément*

Pour obtenir la première forme à partir de leur table de vérité, il suffit de sommer les produits pour lesquels la fonction vaut '1'. La seconde forme s'obtient de manière duale. Notons que ces formes ne sont pas simplifiées.

Etablir les deux formes canoniques des fonctions suivantes :

1. $F_1 = xy + yz + xz$
2. $F_2 = x + yz + \overline{y}.\overline{z}.t$

solutions

1. $F_1 = XY + YZ + XZ$

X	Y	Z	F_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- Première forme canonique

  $F_1 = \overline{X}\,Y\,Z + X\,\overline{Y}\,Z + X\,Y\,\overline{Z} + X\,Y\,Z$

- Seconde forme canonique

  $F_1 = (X + Y + Z)(X + Y + \overline{Z})(X + \overline{Y} + Z)(\overline{X} + Y + Z)$

2. $F_2 = X + YZ + \overline{Y}\,\overline{Z}\,T$

X	Y	Z	T	F_2
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

- Première forme canonique

$$F_2 = \overline{X}\,\overline{Y}\,\overline{Z}\,T + \overline{X}\,YZ\overline{T} + \overline{X}\,YZT + X\,\overline{Y}\,\overline{Z}\,\overline{T} + X\,\overline{Y}\,\overline{Z}\,T + X\,\overline{Y}\,Z\overline{T} + X\overline{Y}\,ZT + XY\overline{Z}\,\overline{T} + XY\overline{Z}\,T + XYZ\overline{T} + XYZT$$

- Seconde forme canonique

$$F_2 = (X+Y+Z+T)(X+Y+\overline{Z}+T)(X+Y+\overline{Z}+\overline{T})(X+\overline{Y}+Z+T)(X+\overline{Y}+Z+\overline{T})$$

3 Quelques formulations booléennes

Ecrire sous la première forme canonique les fonctions définies par les propositions suivantes :

1. $F(a,b,c) = 1$ si et seulement si aucune des variables a, b, c ne prend la valeur 1.
2. $F(a,b,c) = 1$ si et seulement si au plus une des variables a, b, c prend la valeur 0.

solutions

1. $f(A,B,C) = \overline{A}\,\overline{B}\,\overline{C}$
2. $f(A,B,C) = \overline{A}\,BC + A\,\overline{B}\,C + AB\,\overline{C} + ABC$

4 Simplifications algébriques

Simplifier algébriquement les fonctions suivantes

1. $F = \overline{x}.\overline{y} + xy + \overline{x}y$
2. $F = (x + \overline{y})(x\overline{y} + z)z$

solutions

1. $F = \overline{x} + y$
2. $F = (x + \overline{y}).z$

5 Simplifications par tableaux de Karnaugh

Simplifier, par la méthode des tableaux de Karnaugh, les fonctions booléennes suivantes. Dessiner le circuit correspondant et déterminer son chemin critique, en supposant que toutes les portes présentent un temps de propagation de 5 ns. Tous les problèmes ne seront pas forcément résolus en TE.

5.1 K-map à 3 variables

1. $F(a,b,c) = \overline{a}\overline{b}c + \overline{a}b\overline{c} + ab\overline{c}$
2. $F(a,b,c) = \overline{a}b\overline{c} + \overline{a}bc + ab\overline{c}$
3. $F(a,b,c) = \overline{a}\overline{b}c + a\overline{b}\overline{c} + ab\overline{c}$, sachant que la valeur de F pour les combinaisons $\overline{a}bc$ et abc est indifférente.

solutions

1. $F(a,b,c) = \overline{a}\overline{b}c + b\overline{c}$
2. $F(a,b,c) = \overline{a}b + b\overline{c}$
3. $F(a,b,c) = \overline{a}c + a\overline{c} = a \oplus c$

5.2 K-map à 4 variables

$F(a,b,c,d) = \overline{a}b\overline{c}\overline{d} + \overline{a}b\overline{c}d + \overline{a}bcd + \overline{a}bc\overline{d}$
$F(a,b,c,d) = \overline{a}\overline{b}\overline{c}d + \overline{a}\overline{b}cd + a\overline{b}\overline{c}d + a\overline{b}cd$

solutions

$$F(a,b,c,d) = \overline{a}b$$

$$F(a,b,c,d) = \overline{b}d$$

5.3 K-map à 5 variables

$F(a,b,c,d,e) = \overline{a}\overline{b}\overline{c}\overline{d}\overline{e} + \overline{a}\overline{b}c\overline{d}\overline{e} + \overline{a}b\overline{c}\overline{d}\overline{e} + \overline{a}bc\overline{d}\overline{e} + ab\overline{c}\overline{d}\overline{e} + ab\overline{c}d\overline{e} + abcd\overline{e} + abc\overline{d}\overline{e} + a\overline{b}\overline{c}\overline{d}\overline{e} + a\overline{b}c\overline{d}\overline{e} + \overline{a}b\overline{c}\overline{d}e + \overline{a}bc\overline{d}e + ab\overline{c}\overline{d}e + ab\overline{c}de + abcde + abc\overline{d}e$

solution : 3 groupes de 8

$$F(a,b,c,d,e) = ab + b\overline{d} + \overline{d}.\overline{e}$$

6 Différentes utilisations du OU exclusif

1. Rappeler la table de vérité du OU exclusif, ainsi que son symbole (américain)
2. Dessiner la fonction OU-exclusif à partir de portes logiques ET, OU et NOT
3. Le OU-exclusif possède de nombreuses propriétés intéressantes. Compléter les énoncé suivants :
   (a) Sa sortie vaut 1 si exactement une des entrées vaut 1 :
   $$x \oplus y = ?? + ??$$
   (b) On peut le considérer comme un comparateur d'égalité pour 1 bit :
   $$x \oplus y = \text{SI } (x == y) \text{ ALORS } ? \text{ SINON } ?$$
   (c) On peut le considérer comme un inverseur conditionnel :
   $$x \oplus y = \text{SI } (?) \text{ ALORS } ? \text{ SINON } ?$$
   (d) On peut enfin le considérer comme un additionneur modulo 2.

solutions

1. Sa sortie vaut 1 si exactement une des entrées vaut 1 :
   $$x \oplus y = x.\overline{y} + \overline{x}.y$$
2. On peut le considérer comme un comparateur d'égalité pour 1 bit :
   $$x \oplus y = \text{SI } (x == y) \text{ ALORS } 0 \text{ SINON } 1$$
3. On peut le considérer comme un inverseur conditionnel :
   $$x \oplus y = \text{SI } (y == 1) \text{ ALORS } \overline{x} \text{ SINON x}$$
4. On peut enfin le considérer comme un additionneur modulo 2.

7 Le jeu des cambrioleurs

Le principe du jeu est le suivant : un jeton est introduit par le joueur en haut du dispositif. Le but est de faire circuler ce jeton dans le dispositif, jusqu'à ce qu'il apparaisse en sortie. dans l'intervalle, le jeton peut rester coincé à différents endroits (ou "niveaux") : seul un code secret correctement introduit à cet endroit permet au jeton de continuer son parcours. Sinon, l'utilisateur voit le code d'erreur 0xDEAD apparaître en sortie. Chaque code secret, associé au niveau i est codé sur 4 bits.

1. Imaginer un circuit combinatoire qui simule ce jeu. On utilisera pour cela différents circuits connus : multiplexeurs et comparateurs. Enrichir votre circuit avec d'autres circuits combinatoires afin de le démarrer et d'indiquer le nombre de niveaux franchis correctement.
2. Sachant que l'on peut envoyer 1 million de codes par secondes, combien de temps faudrait-il, au pire cas, pour tester toutes les combinaisons possibles, dans le cas où le nombre de niveaux est 9 ? Refaites le calcul pour 5 bits.

Cet exercice ne nécessite pas de formules

solution : Le but est de faire manipuler des circuits plus complexes que les portes logiques, afin de réaliser un "chemin de données" (au sens littéral !), notamment en utilisant des multiplexeurs.

On choisit de représenter le jeton par un vecteur de bits de taille donnée $n = 16$, du fait de la présence du code 0xDEAD... (la taille n'influe pas sur la structure de la solution, si ce n'est la propagation du code d'erreur fixé 0xDEAD). Ne pas représenter explicitement les 16 fils, mais juste un fil, annoté par "n=16". Ce jeton se présente sur un port d'entrée DIN...

Il y a $(2^4)^9 = 2^{36}$ combinaisons à tester, soit environ 19 heures. Pour le cas où les codes sont sur seulement 1 bit de plus (5 bits), il faut 405 jours.

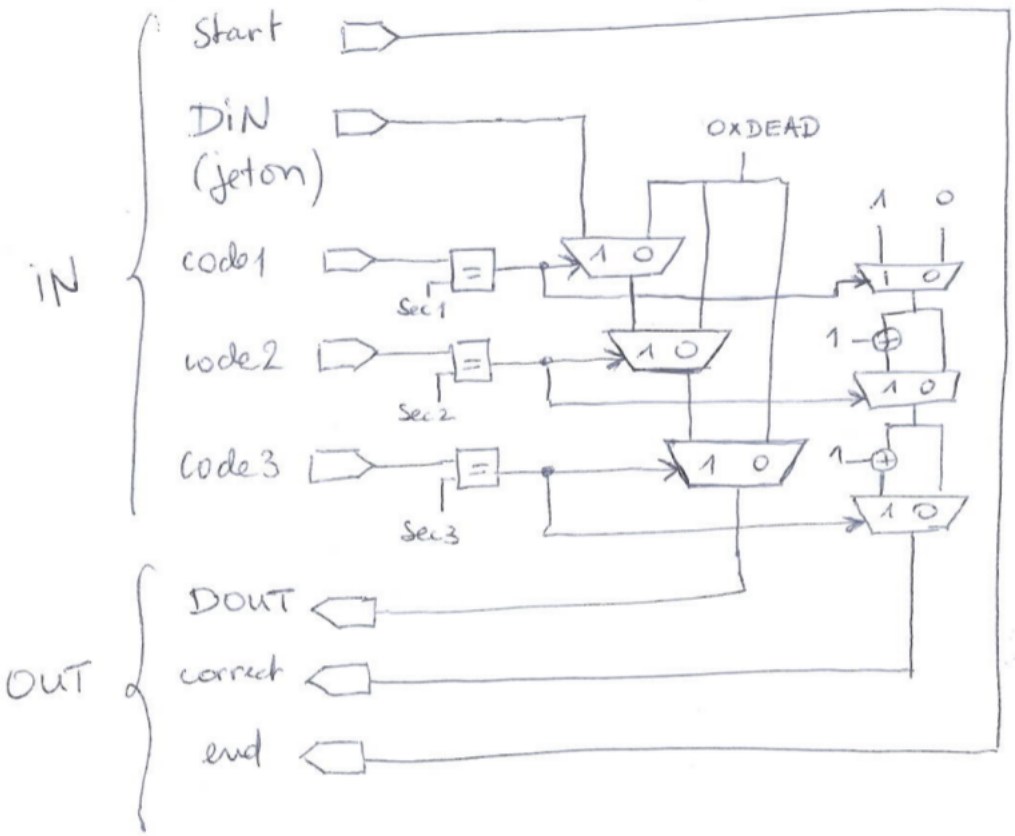
8 Vu-mètre

On se propose de réaliser un "contrôleur de vu-mètre" : c'est un dispositif qui permet de visualiser une grandeur numérique (un volume sonore par exemple) au moyen d'une barrette lumineuse qui s'étire plus ou moins un fonction de la grandeur.

1. Schématiser le principe d'ensemble dans le cas où la grandeur à visualiser est codée sur 3
2. A partir du tableau de Karnaugh, trouver une réalisation possible.
3. On supposer que l'on dispose désormais d'un composant complexe de décodage (un décodeur). Rappeler la table de vérité de ce circuit. Utiliser alors ce décodeur pour proposer une nouvelle réalisation.

solution :

Il faut faire très attention : lorsqu'on envoie un code 000 cela signifie qu'on ne veut *aucune* LED allumée. De même, quand on envoie le code 111 (7), on veut voire 7 LEDs alumées, qui sont numérotées de 0 à 6 (et



non 7).

e2	e1	e0	L0	L1	L2	L3	L4	L5	L6
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
1	0	0	1	1	1	1	0	0	0
1	0	1	1	1	1	1	1	0	0
1	1	0	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1

(Pour les premiers,partir de la sortie complémentée plutôt)

$$L_0 = e_2 + e_1 + e_0$$

$$L_1 = e_2 + e_1$$

$$L_2 = \overline{e_2 e_1} + \overline{e_2} e_1 \overline{e_0}$$

$$L_3 = e_2$$

$$L_4 = e_2.e_1 + e_2.\overline{e_1}.e_0$$

$$L_5 = e_2.e_1$$

$$L_6 = e_2.e_1.e_0$$

Passons maintenant à l'utilisation d'un encodeur. La sortie C_i d'un encodeur s'allume lorsque la *valeur i* correspondante est entrée. Attention ! Cela signifie que sur 3 bits, on a 8 sorties (de 0 à 7).

e2	e1	e0	C0	C1	C2	C3	C4	C5	C6	C7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

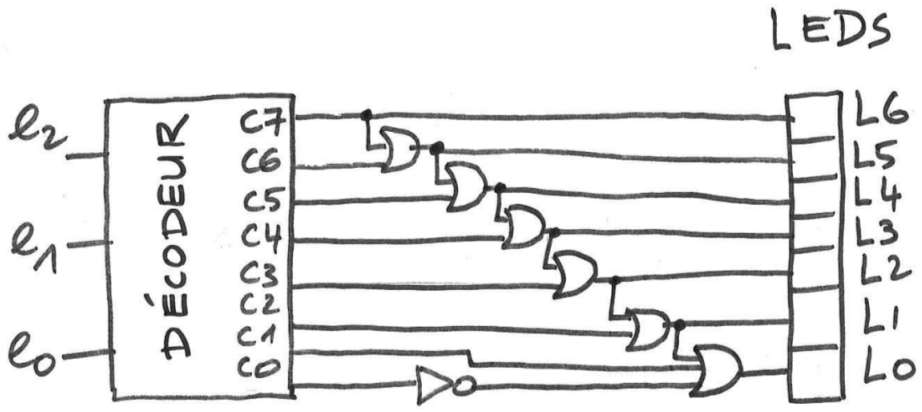
L_0 doit être éteinte quand $C_0 = 1$:

$$\overline{L_0} = C_0.\overline{C_1}.\overline{C_2}...\overline{C_7}$$

d'où :

$$L_0 = \overline{C_0} + C_1 + C_2...C_7 \quad (1)$$
$$L_1 = C_2 + C_3 + ... + C_7 \quad (2)$$
$$L_2 = C_3 + ... + C_7 \quad (3)$$
$$\dots \quad (4)$$
$$L_5 = C_6 + C_7 \quad (5)$$
$$L_6 = C_7 \quad (6)$$

On peut également remarquer que les deux tables (celle initiale et celle du décodeur) sont très proches. Il suffit de cascader des OU logiques de manière intelligente. Seul le cas du zéro est problématique.


LEDS
e_2
e_1
e_0
DÉCODEUR
C7
C6
C5
C4
C3
C2
C1
C0
L6
L5
L4
L3
L2
L1
L0