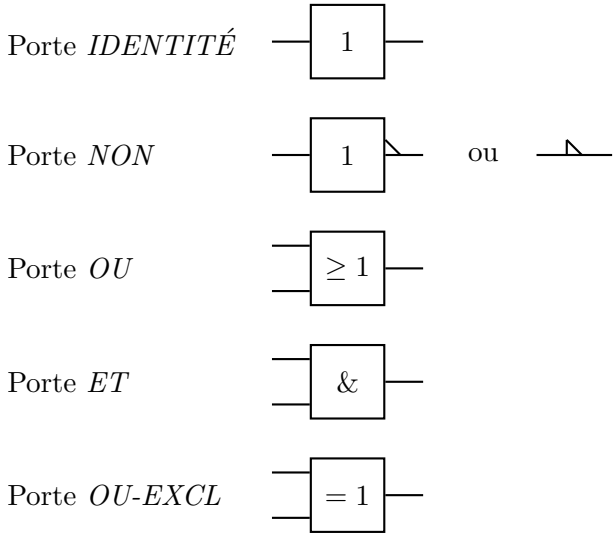
Architecture des Ordinateurs, corrigé TD 1

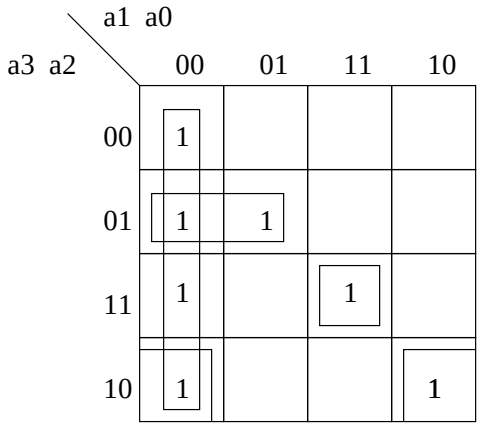
Fonctions booléennes

Remarque : Les schémas des circuits logiques sont réalisés à partir de la notation IEEE :



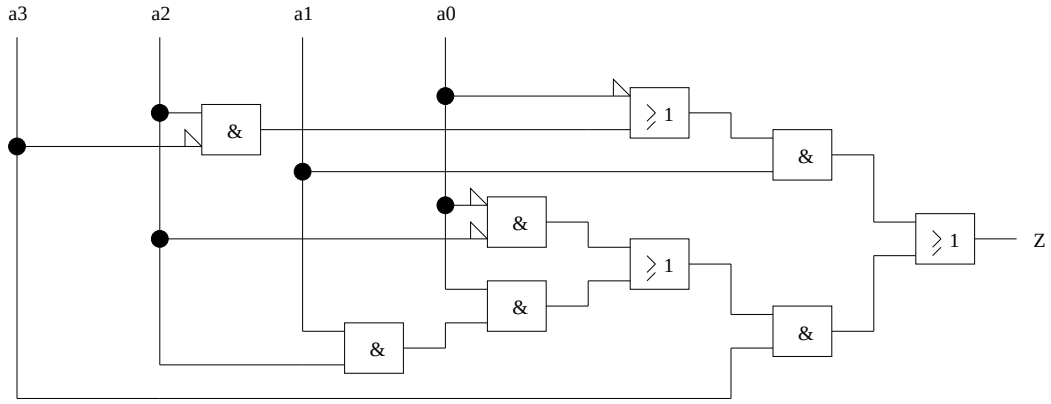
Exercice 1. Etudier un circuit combinatoire à quatre entrées a0, a1, a2, a3, et une sortie Z tel que Z = 1 chaque fois que le numéro codé par l'entier a3a2a1a0 est divisible entièrement par 4 ou 5.

Correction.



$$\begin{aligned} Z &= \overline{a1}.\overline{a0} + \overline{a3}.a2.\overline{a1} + a3.\overline{a2}.\overline{a0} + a3.a2.a1.a0 \\ &= \overline{a1}.(\overline{a0} + \overline{a3}.a2) + a3.(\overline{a2}.\overline{a0} + a2.a1.a0) \end{aligned}$$

d'où le schéma :



Exercice 2. On considère un ascenceur desservant un rez-de chaussée et trois étages. L'unité logique de contrôle de cet ascenceur reçoit en entrée des requêtes de déplacement d'un étage vers un autre, et génère en sortie les commandes correspondantes destinées au moteur de la cage. Il s'agit de réaliser les circuits qui composent l'unité de contrôle. Pour cela :

- La valeur de l'étage d'où part la requête (de 0 à 3) est codée par deux variables booléennes $x1$ et $x2$. La valeur courante de l'étage où se trouve la cage de l'ascenceur est codée de façon similaire par deux variables booléennes $x3$ et $x4$.
- La sortie de l'unité de contrôle est constituée de trois fonctions booléennes *Haut, Bas, Stop* des quatre variables d'entrée $x1$, $x2$, $x3$, et $x4$. $Haut(x1, x2, x3, x4)$ est vraie quand la position courante de la cage est *au-dessous* de l'étage d'où part la requête et fausse autrement. Inversement, $Bas(x1, x2, x3, x4)$ est vraie quand la position courante de la cage est *au-dessus* de l'étage d'où part la requête et fausse autrement. Finalement, $Stop(x1, x2, x3, x4)$ est vraie quand l'étage d'où part la requête et la position courante de l'ascenceur sont les *mêmes,* et fausse autrement.

On vous demande de :

1. représenter chacune des fonctions de l'unité de contrôle par une table de Karnaugh ;
2. simplifier ces fonctions ;
3. dessiner les schémas des circuits correspondant à chacune des fonctions simplifiées, en utilisant *le plus petit nombre possible* de portes.

Correction. *La valeur de l'étage d'où part la requête est représentée par la table binaire suivante :*

$x1$	$x2$	*Etage*
0	*0*	*Rez-de-chaussée*
0	*1*	*1er*
1	*0*	*2d*
1	*1*	*3ème*

De même, la valeur courante de l'étage où se trouve la cage de l'ascenceur est représenté par la table binaire :

$x3$	$x4$	*Etage*
0	*0*	*Rez-de-chaussée*
0	*1*	*1er*
1	*0*	*2d*
1	*1*	*3ème*

Pour la fonction booléenne $\text{Haut}(x1, x2, x3, x4)$ *par exemple, il suffit de remplir une table de Karnaugh à quatre variables de telle façon que pour chaque case un "1" y est placé si la valeur courante de l'étage où se trouve la cage de l'ascenceur est plus basse que la valeur de l'étage d'où part la requête, et un "0" y est placé sinon :*

x1 x2 \ x3 x4	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

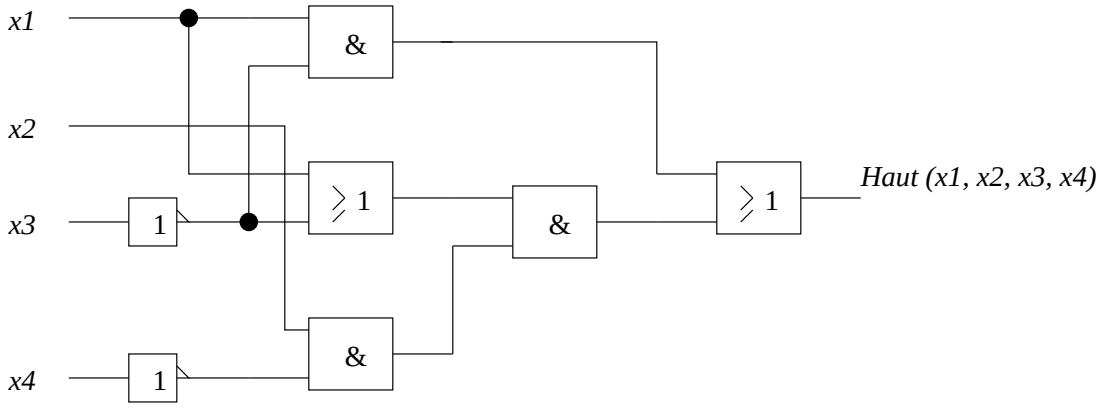
L'expression booléenne résultante après simplification est :

$$\text{Haut}(x1, x2, x3, x4) = x1.\overline{x3} + x2.\overline{x3}.\overline{x4} + x1.x2.\overline{x4}$$

Bien que cette expression soit une forme normale disjonctive minimale, il est possible de pousser la simplification plus loin :

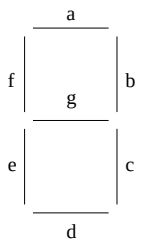
$$x1.\overline{x3} + x2.\overline{x3}.\overline{x4} + x1.x2.\overline{x4} = x1.\overline{x3} + x2.\overline{x4}.(\overline{x3} + x1)$$

Le circuit obtenu a exactement sept portes logiques :



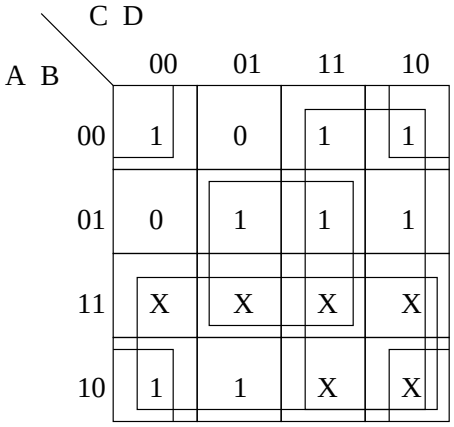
Les autres fonctions booléennes sont établies de la même manière.

Exercice 3. On désire réaliser le système inclus dans une calculette qui à un digit décimal fait correspondre l'allumage de segments représentant ce digit. Le digit étant codé sur quatre valeurs A, B, C, D, calculer les sept fonctions a, b, c, d, e, f, g commandant l'éclairement des segments correspondants :



Correction.

A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x



$$
\begin{aligned}
a &= A + C + B.D + \overline{B}.\overline{D} \\
. &\ . \ . \\
. &\ . \ . \\
\\
g &= A + B.\overline{C} + \overline{B}.C + C.\overline{D}
\end{aligned}
$$

Exercice 4. Réalisation d'un additionneur/soustracteur (portes logiques disponibles : *ET*, *OU*, *NON*, *OU EXCL*)

1. Réaliser un demi-soustracteur (1 bit A avec 1 bit B sans retenue d'entrée) :
   — Ecrire la table de vérité.
   — Donner les équations de sortie.
   — Etablir le schéma logique.
2. En comparant le circuit du demi-soustracteur avec celui d'un demi-additionneur, concevoir le plus simplement possible un circuit, appelé demi-additionneur/soustracteur, qui à partir d'un signal de commande C et des entrées A et B, simule le demi-additionneur sur A et B lorsque la commande C est à 0, et le demi-soustracteur sur A et B lorsque la commande C est à 1 (suggestion : appliquer le signal de commande à une des entrées d'une porte *OU EXCL*).
3. A partir du demi-additionneur/soustracteur qui vient d'être réalisé, concevoir un additionneur/soustracteur complet (1 bit A avec un bit B avec retenue d'entrée).
4. Donner le schéma d'un additionneur/soustracteur quatre bits par quatre bits.

Correction.

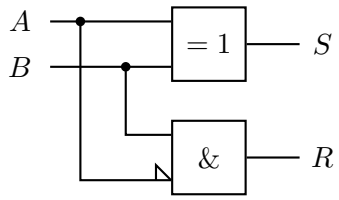
1. *Un* demi-soustracteur *est un circuit qui soustrait simplement un bit d'un autre. Le résultat est obtenu sur deux bits, S pour le poids faible (la différence), R pour le poids fort (la retenue). A partir de la table de vérité suivante :*

A	B	R	S
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

on obtient :

$$
\begin{aligned}
S &= \overline{A}.B + A.\overline{B} \\
&= A \oplus B \\
\\
R &= \overline{A}.B
\end{aligned}
$$

Le demi-soustracteur est réalisé par le circuit suivant :



2. *On constate que la seule différence entre un demi-soustracteur et un demi-additionneur tient à la présence d'une négation sur l'entrée A de la porte ET qui génère la retenue. L'ajout d'une porte OU EXCL dont une des entrées est la commande C doit donc permettre de simuler la négation présente dans le demi-soustracteur lorsque $C = 1$, et l'absence de cette négation dans le demi-additionneur lorsque $C = 0$. Il est aisé de vérifier partir de la table de vérité de OU EXCL qu'il suffit de prendre directement A pour l'autre entrée de la porte OU EXCL : lorsque $C = 0$, la sortie de la porte est équivalente à A, lorsque $C = 1$ cette sortie prend la valeur de $\overline{A}$.*

C	A	$C \oplus A$
0	0	0
0	1	1
1	0	1
1	1	0

Bien entendu, une étude systématique des sorties à partir de toutes les entrées permet de retrouver les équations booléennes attendues :

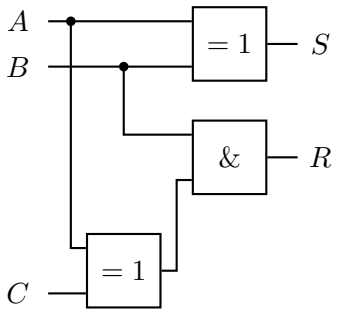
C	A	B	R	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	0	0

on obtient :

$$\begin{aligned} S &= \overline{C}.\overline{A}.B + \overline{C}.A.\overline{B} + C.\overline{A}.B + C.A.\overline{B} \\ &= \overline{C}.(A \oplus B) + C.(A \oplus B) \\ &= (\overline{C} + C).(A \oplus B) \\ &= A \oplus B \end{aligned}$$

$$\begin{aligned} R &= \overline{C}.A.B + C.\overline{A}.B \\ &= (\overline{C}.A + C.\overline{A}).B \\ &= (C \oplus A).B \end{aligned}$$

Le schéma d'un demi-additionneur/soustracteur est donc :



3. *La table de vérité de l'additionneur/soustracteur est (avec R_e, retenue en entrée et R_s retenue en sortie) :*

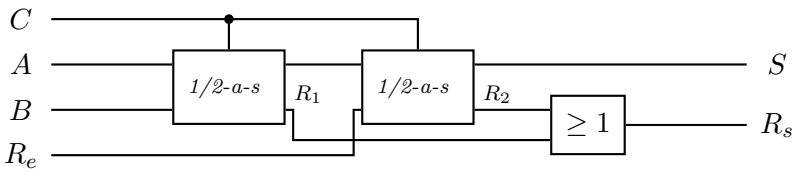
C	R_e	A	B	R_s	S
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	0	0
1	1	0	0	1	1
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	1	1	1

on obtient :

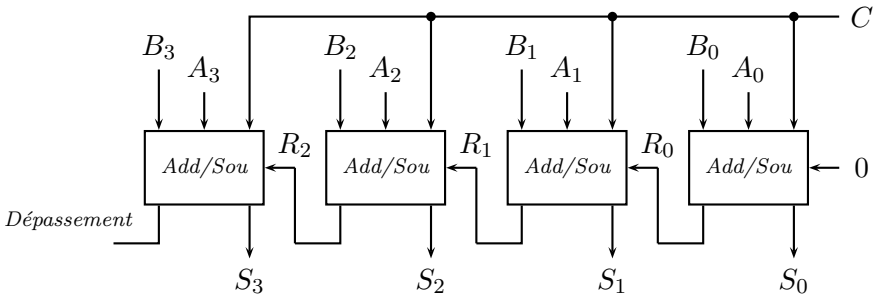
$$\begin{aligned}
S &= \overline{C}.\overline{R_e}.\overline{A}.B + \overline{C}.\overline{R_e}.A.\overline{B} + \overline{C}.R_e.\overline{A}.\overline{B} + \overline{C}.R_e.A.B + \\
&\quad C.\overline{R_e}.\overline{A}.B + C.\overline{R_e}.A.\overline{B} + C.R_e.\overline{A}.\overline{B} + C.R_e.A.B \\
&= (\overline{C}+C).\overline{R_e}.\overline{A}.B + (\overline{C}+C).\overline{R_e}.A.\overline{B} + (\overline{C}+C).R_e.\overline{A}.\overline{B} + (\overline{C}+C).R_e.A.B \\
&= \overline{R_e}.(\overline{A}.B + A.\overline{B}) + R_e.(\overline{A}.\overline{B} + A.B) \\
&= \overline{R_e}.(A \oplus B) + R_e.\overline{A \oplus B} \\
&= R_e \oplus A \oplus B
\end{aligned}$$

$$\begin{aligned}
R &= \overline{C}.\overline{R_e}.A.B + \overline{C}.R_e.\overline{A}.B + \overline{C}.R_e.A.\overline{B} + \overline{C}.R_e.A.B + \\
&\quad C.\overline{R_e}.\overline{A}.B + C.R_e.\overline{A}.\overline{B} + C.R_e.\overline{A}.B + C.R_e.A.B \\
&= \overline{C}.R_e.(\overline{A}.B + A.\overline{B}) + C.R_e.(\overline{A}.\overline{B} + A.B) + \overline{C}.(\overline{R_e} + R_e).A.B + C.(\overline{R_e} + R_e).\overline{A}.B \\
&= (\overline{C}.(A \oplus B) + C.(\overline{A \oplus B})).R_e + (\overline{C}.A + C.\overline{A}).B \\
&= \underbrace{(C \oplus A \oplus B).R_e}_{R_2} + \underbrace{(C \oplus A).B}_{R_1}
\end{aligned}$$

Le schéma de l'additionneur/soustracteur est immédiat :



4. *Le schéma d'un additionneur/soustracteur quatre bits par quatre bits s'obtient de façon toute aussi immédiate :*



Exos supplémentaires

Exercice 5.

1. Exprimer les opérateurs *NON*, *ET*, *OU* en fonction de l'unique opérateur *NON-OU*, résultant de la composition de *NON* et *OU*, et vu comme un opérateur booléen à part entière défini par :
$$NON - OU(x, y) = \overline{x + y}$$
2. Exprimer avec des *NON-OU* uniquement l'expression booléenne suivante :
$$((x_1 + x_2).(x_3 + x_4.x_5)).\overline{x_6}$$
3. Dessiner ensuite le schéma logique correspondant.

Correction.

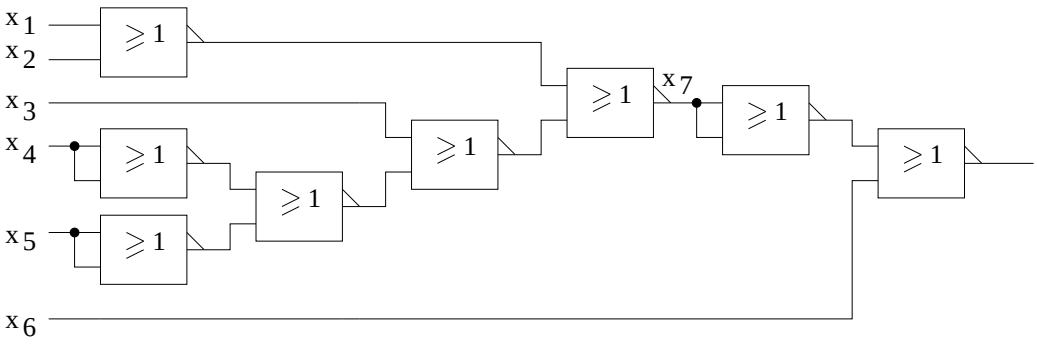
$$\begin{array}{ll} NON & \overline{x} = \overline{x + x} \\ ET & x.y = \overline{\overline{x}.\overline{y}} = \overline{\overline{x} + \overline{y}} = \overline{\overline{x + x} + \overline{y + y}} \\ OU & x + y = \overline{\overline{x + y}} = \overline{\overline{x + y} + \overline{x + y}} \end{array}$$

$$\begin{array}{rcl} f(x_1, x_2, x_3, x_4, x_5, x_6) & = & \underbrace{((x_1 + x_2).(x_3 + x_4.x_5))}_{x_7}.\overline{x_6} \\ & = & \overline{\overline{x_7 + x_7} + \overline{\overline{x_6}}} \\ & = & \overline{\overline{x_7 + x_7} + x_6} \end{array}$$

$$\begin{array}{rcl} x_7 & = & (x_1 + x_2).(x_3 + x_4.x_5)) \\ & = & \overline{\overline{(x_1 + x_2) + (x_1 + x_2)} + \overline{(x_3 + x_4.x_5) + (x_3 + x_4.x_5)}} \\ & = & \overline{\overline{(x_1 + x_2)} + \overline{(x_3 + x_4.x_5)}} \\ x_4.x_5 & = & \overline{\overline{x_4 + x_4} + \overline{x_5 + x_5}} \end{array}$$

d'où :

$$x_7 = \overline{\overline{x_1 + x_2} + \overline{x_3 + \overline{\overline{x_4 + x_4} + \overline{x_5 + x_5}}}}$$



Exercice 6. Réalisation d'un multiplicateur 2 bits par 2 bits :

- Réaliser un circuit qui effectue la multiplication 1 bit par 1 bit.
- Réaliser un multiplicateur 2 bits par 2 bits
  - directement à l'aide de portes *ET*, *OU*, *NON*, *NON–ET*, *NON–OU*...
  - alternativement, à l'aide du multiplicateur 1 bit par 1 bit réalisé ci-dessus et de demi-additionneurs.

Correction.

1. *Circuit qui effectue la multiplication 1 bit par 1 bit : a et b étant les deux bits à multiplier, la fonction booléenne $P = a.b$*
2. *On note $a1a0$ les deux bits du premier nombre, et $b1b0$ les deux bits du deuxième nombre ; le résultat s'écrit sur 4 bits $p3p2p1p0$.*
   (a)

$b1$	$b0$	$a1$	$a0$	$p3$	$p2$	$p1$	$p0$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

$$
\begin{aligned}
p3 &= b1.b0.a1.a0 \\
p2 &= b1.\overline{b0}.a1.\overline{a0} + b1.\overline{b0}.a1.a0 + b1.b0.a1.\overline{a0} \\
&= b1.\overline{b0}.a1.\overline{a0} + b1.\overline{b0}.a1.\overline{a0} + b1.\overline{b0}.a1.a0 + b1.b0.a1.\overline{a0} \\
&= b1.\overline{b0}.a1 + b1.a1.\overline{a0} \\
&= b1.a1.(\overline{b0} + \overline{a0}) \\
&= b1.a1.\overline{b0.a0} \\
p1 &= \overline{b1}.b0.a1.\overline{a0} + \overline{b1}.b0.a1.a0 + b1.\overline{b0}.\overline{a1}.a0 + b1.\overline{b0}.a1.a0 + b1.b0.\overline{a1}.a0 + b1.b0.a1.\overline{a0} \\
&= \overline{b1}.b0.a1 + b1.\overline{b0}.a0 + b1.b0.(a1 \oplus a0) \\
&= \overline{b1}.b0.a1 + b1.(\overline{b0}.a0 + b0.(a1 \oplus a0)) \\
p0 &= \overline{b1}.b0.\overline{a1}.a0 + \overline{b1}.b0.a1.a0 + b1.b0.\overline{a1}.a0 + b1.b0.a1.a0 \\
&= \overline{b1}.b0.a0 + b1.b0.a0 \\
&= b0.a0
\end{aligned}
$$

(b) Remarquons que (développement polynômial d'un nombre en base 2) :

$$\begin{aligned} a1a0 &= a1 \times 2^1 + a0 \times 2^0 \\ b1b0 &= b1 \times 2^1 + b0 \times 2^0 \end{aligned}$$

Rappelons que l'exposant associé à la base correspond au rang de chacun des bits qui codent le nombre. On a alors :

$$\begin{aligned} a1a0 \times b1b0 &= (a1 \times 2^1 + a0 \times 2^0) \times (b1 \times 2^1 + b0 \times 2^0) \\ &= a1 \times b1 \times 2^2 + (a1 \times b0 + a0 \times b1) \times 2^1 + a0 \times b0 \times 2^0 \end{aligned}$$

Dans cette équation

$p0 = a0 \times b0$

$p1 = a1 \times b0 + a0 \times b1$ *qui représente une somme de deux bits réalisée avec un demi–additionneur, et donc génère une retenue* R_1 *de rang supérieur d'où :*

$p2 = a1 \times b1 + R_1$ *qui représente une somme de deux bits réalisée avec un demi–additionneur, et donc génère une retenue* R_2 *de rang supérieur d'où :*

$p3 = R_2$

Autrement dit,

$p3p2p1p0 = R_2 \times 2^3 + (a1 \times b1 + R_1) \times 2^2 + (a1 \times b0 + a0 \times b1) \times 2^1 + (a0 \times b0) \times 2^0$

Exemple :

```
  11          a1a0 = 11
x 11          b1b0 = 11
----
  11
 11
----
1001          p3 = 1, p2 = 0, p1 = 0, p0 = 1
```

A partir des équation de $p3, p2, p1, p0$ *obtenues ci-dessus, on obtient le schéma :*