# 1 Problématique

Un ordinateur manipule des données codées sous forme de *bits* en mémoire. Des combinaisons de transistors assurent des opérations booléennes élémentaires.

À partir de ces éléments, comment réaliser une addition de nombres binaires ?

Comment traduire le programme assembleur de l'addition (vu précédemment) en circuit électronique ?

# 2 Fonctions logiques

## 2.1 Notations booléennes

Les portes logiques évoquées dans les cours précédents peuvent être vues comme des fonctions booléennes élémentaires. On note ainsi les fonctions associées à chaque porte :

— $\neg(x)$ pour NOT
— $\wedge(x)$ pour AND
— $\vee(x)$ pour OR

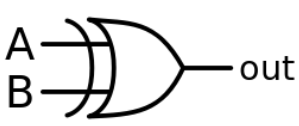
## 2.2 Nouvelle porte logique

Une porte logique élémentaire est également fréquemment utilisée dans les circuits électroniques : le **ou exclusif XOR**. Sa fonction booléenne associée se note $\oplus$.

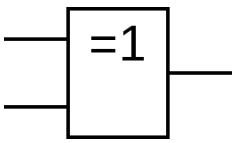
fromage ou dessert

| x | y | $x \oplus y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Tableau 1 – Fonction XOR




Symbole américain




Symbole européen

## 2.3 Opérations booléennes

Nous pouvons imaginer des fonctions booléennes complexes et déterminer leurs tables de vérités. Par exemple la fonction suivante $f$ et définit par :

$$f(x, y, z) = (x \wedge y) \oplus (\neg y \vee z)$$



Christophe Viroulaud

Nous pourrions écrire :

(x AND y) XOR (NOT y OR z)

**Activité 1 :**

1. La fonction $f$ a trois paramètres. Combien de combinaisons possibles peut-on réaliser avec ces paramètres ?
2. Établir la table de vérité des différentes expressions ci-après (tableau 2).
3. En déduire la table de vérité de $f$ (tableau 3).

| x | y | z | $(x \wedge y)$ | $\neg y$ | $(\neg y \vee z)$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | | | |
| 0 | 0 | 1 | | | |
| ... | ... | ... | | | |

Tableau 2 – Table de vérité de plusieurs expressions

| x | y | z | $(x \wedge y) \oplus (\neg y \vee z)$ |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| ... | ... | ... | |

Tableau 3 – Table de vérité de $f$

# 3 Réaliser un additionneur

## 3.1 Décomposition

Pour additionner deux nombres de $n$ *bits* il faut additionner bit à bit et prendre en compte une éventuelle retenue.
La première étape consiste à construire un additionneur 1 bit. Il est lui-même construit à partir de circuit plus simple : *le demi-additionneur*.

Refaire un exemple

## 3.2 Demi-additionneur

Un demi-additionneur prend deux bits en entrée $e_0$ et $e_1$ et renvoie la somme $e_0 + e_1$ en sortie $s$. Il faut prendre en compte une éventuelle retenue $c$. La table de vérité correspondante est (Tableau 4) :

**Activité 2 :**

1. Quelles fonctions logiques reconnaît-on en $s$ et $c$ ?
2. En déduire le schéma du demi-additionneur 1 bit.



Christophe Viroulaud

| $e_0$ | $e_1$ | s | c |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Tableau 4 – Table de vérité du demi-additionneur

$s = e_0 \oplus e_1$
$c = e_0 \wedge e_1$

### 3.3 Additionneur 1 bit

Dans une addition bit à bit il faut prendre en compte l'éventuelle retenue de l'addition précédente. Ainsi un additionneur 1 bit prend trois entrées $e_0$, $e_1$ et la retenue précédente $c_0$. Il renvoie une sortie $s = e_0 + e_1 + c_0$ et une retenue éventuelle $c$.
En combinant deux demi-additionneurs nous pouvons construire un additionneur 1 bit.

**Activité 3 :**

1. Établir la table de vérité de l'additionneur 1 bit.
2. Sur la figure 1 placer les entrées $e_0$, $e_1$, $c_0$ et les sorties $s$ et $c$.

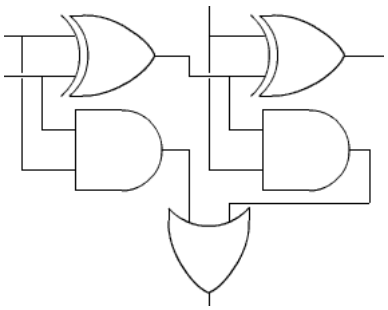


FIGURE 1 – Additionneur 1 bit

logiciel logisim ? http://ww2.ac-poitiers.fr/techno-si/spip.php?article348


Christophe Viroulaud