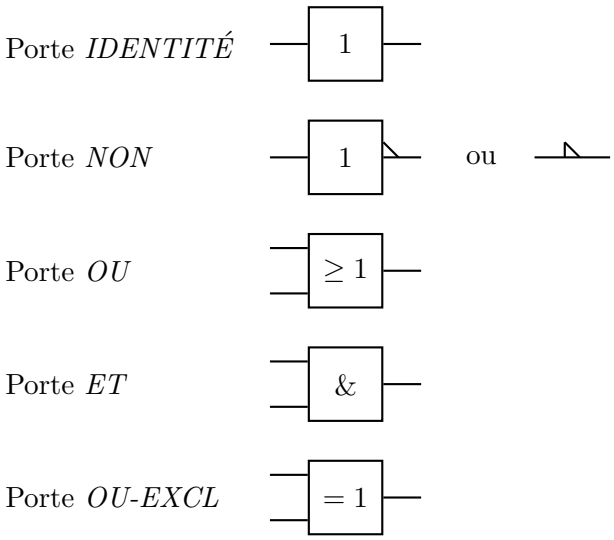
Architecture des Ordinateurs, TD 1

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.

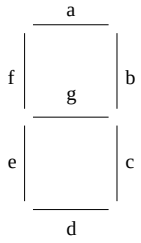
Exercice 2. On considère un ascenceur desservant un rez-de chaussée et trois étages. L'unité logique de contrôle de cet ascenceur reçoit en entrée des requêtes de déplacement d'un étage vers un autre, et génère en sortie les commandes correspondantes destinées au moteur de la cage. Il s'agit de réaliser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

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 :



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.

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 :
$$NON\text{-}OU(x, y) = \overline{x + y}$$
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.

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.