Architecture des ordinateurs

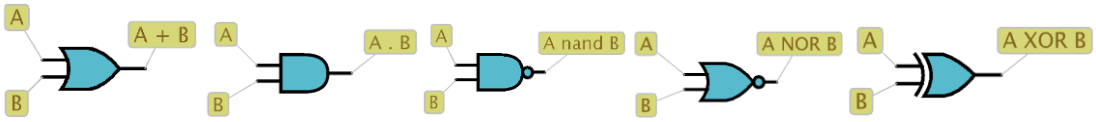
TD1 : Algèbre de Boole ; portes et fonctions combinatoires

Rappel des notions abordées (polycopié, chapitre II, pages 9 à 31)

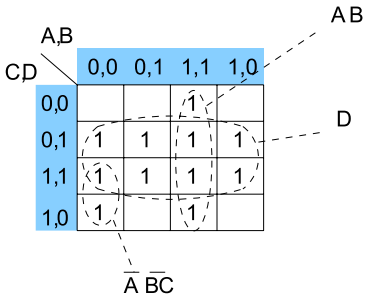
- Algèbre de Boole et ses théorèmes ;

relation	relation duale	Propriété
$AB = BA$	$A + B = B + A$	Commutativité
$A.(B + C) = A.B + A.C$	$A + B.C = (A + B).(A + C)$	Distributivité
$1.A = A$	$0 + A = A$	Identité
$A.\overline{A} = 0$	$A + \overline{A} = 1$	Complément
$0.A = 0$	$1 + A = 1$	Zéro et un
$A.A = A$	$A + A = A$	Idempotence
$A.(B.C) = (A.B).C$	$A + (B + C) = (A + B) + C$	Associativité
$\overline{\overline{A}} = A$		Involution
$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}.\overline{B}$	Théorèmes de De Morgan
$A.(A + B) = A$	$A + A.B = A$	Théorèmes d'absorption
$A + \overline{A}.B = A + B$	$A.(\overline{A} + B) = A.B$	Théorèmes d'absorption

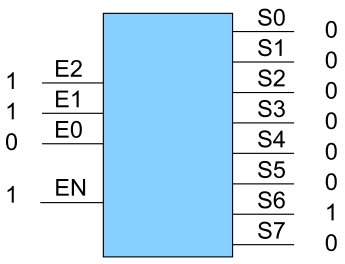
- fonctions combinatoires, portes



- simplification des sommes de minterms par tables de Karnaugh

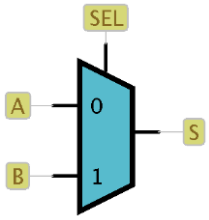


- décodeurs, utilisation pour produire des fonctions combinatoires. Exemple de décodeur 3 vers 8 avec entrée de validation :




```
module decodeur(e[2..0], en : s[7..0])
    s[7] = en*e[2]*e[1]*e[0] ;
    s[6] = en*e[2]*e[1]*/e[0] ;
    s[5] = en*e[2]*/e[1]*e[0] ;
    s[4] = en*e[2]*/e[1]*/e[0] ;
    s[3] = en*/e[2]*e[1]*e[0] ;
    s[2] = en*/e[2]*e[1]*/e[0] ;
    s[1] = en*/e[2]*/e[1]*e[0] ;
    s[1] = en*/e[2]*/e[1]*/e[0] ;
end module
```

- le multiplexeur : réalise la fonction si-alors-sinon

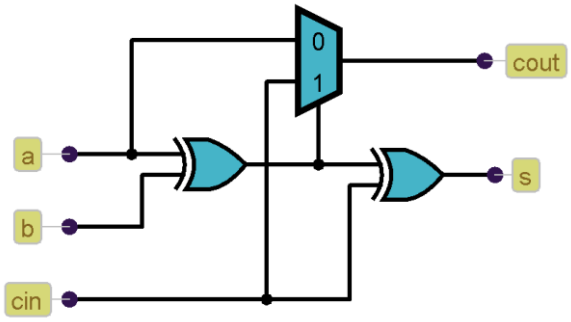


SEL	S
0	A
1	B

- XOR a plusieurs fonctions très utiles :
  - XOR2 est un inverseur commandé et un opérateur de différence
  - XORn est un indicateur d'imparité sur n bits
  - XORn est un opérateur d'addition sur n bits
- le langage SHDL (voir syntaxe en fin de sujet). Il s'agit d'un langage structurel (et non fonctionnel), dans lequel la structure de chaque objet doit être décrite. La modularité permet de construire des bibliothèques de composants réutilisables.

1. Simplification des fonctions combinatoires

On considère le schéma suivant :



a- Ecrire les équations algébriques des sorties `cout` et `s` à l'aide des opérateurs et, ou, xor, non.

b- Ecrire la table de vérité de `cout`.

c- Simplifier `cout` en utilisant les théorèmes de l'algèbre de Boole.

d- Simplifier `cout` en utilisant une table de Karnaugh. Quelles fonctions réalisent `s` et `cout` ?

2. Décodeurs ; conception fonctionnelle

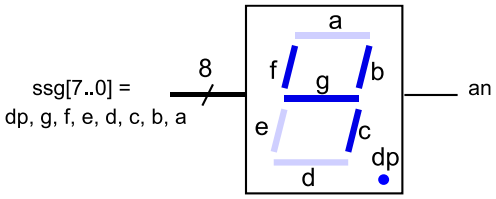
Concevoir un décodeur 2 : 4 sans entrée de validation. On appellera ses entrées e1, e0 et ses sorties s3, s2, s1, s0. On le décrira par un schéma ou des équations.

Concevoir ensuite un décodeur 3 : 8 en réutilisant un décodeur 2 : 4 et en ajoutant un peu de logique combinatoire. On appellera ses entrées e2, e1, e0 et ses sorties s7, s6,…, s0.

Cette méthode est-elle généralisable à des décodeurs de plus grandes tailles ?

3. Décodeur 7 segments

Un afficheur 7 segments est un ensemble de 7 leds organisées en segments (et un point décimal), qui peuvent représenter des chiffres décimaux ou hexadécimaux. Les cathodes des leds des segments sont appelées a, b,…, e ; les anodes de tous les segments sont reliées entre elles au signal `an` qui permet l'activation ou l'extinction de tout l'affichage:



a. Pour un décodeur hexadécimal, écrire la table de vérité du segment a ; la simplifier à l'aide d'une table de Karnaugh
b. Pour un décodeur décimal, écrire la table de vérité du segment a ; la simplifier à l'aide d'une table de Karnaugh
c. Concevoir un décodeur 7 segments hexadécimal complet, en utilisant un décodeur 4 : 16 et un peu de logique combinatoire. On appellera les entrées x3, x2, x1, x0 et les sorties a, b, c, d, e, f, g

4. Transcodeurs

Ecrire la table de vérité d'un circuit de transcodage du binaire pur vers le code de Gray réfléchi sur 4 bits.

On appellera a,b,c,d les entrées en binaire pur et x,y,z,t les sorties en code de Gray.

Trouver la forme algébrique la plus simple du résultat, par la méthode de votre choix.