Algèbre de Boole et circuits logiques

Architecture des ordinateurs

Guillaume Blin

IGM-LabInfo UMR 8049,
Bureau 4B066
Université de Marne La Vallée
gblin@univ-mlv.fr
http://igm.univ-mlv.fr/∼gblin

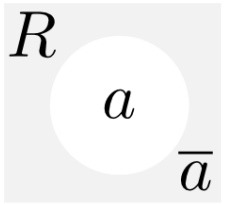
Références

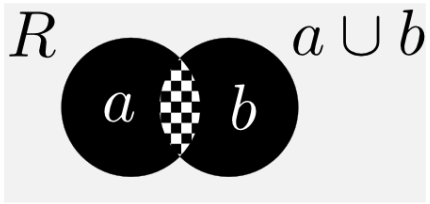
- Cours de Frédéric Goualard de l'Université de Nantes http ://supports.goualard.free.fr/
- Cours de Eric Garcia de l'IUT GTR, Montbéliard
- Emmanuel Viennet de l'IUT de Viltaneuse

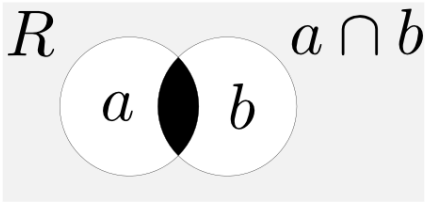
Plan

Plan

Notion ensembliste







Logique des propositions

- Proposition = énoncé vrai ou faux
  - a : les élèves sont présents
  - b : le professeur est présent
  - cours si a et b
- Combinaison par des connecteurs
  - $\neg$ (non logique) $\wedge$ (et logique) $\vee$ (ou logique non exclusif)

a	b	$\neg a$	$a \wedge b$	$a \vee b$
V	V	F	V	V
V	F	F	V	F
F	V	V	V	F
F	F	V	F	F

Passage à l'algèbre de Boole

- Il y a d'autres connecteurs et quelques lois dans la logique des propositions : $\neg a \wedge \neg b = \neg(a \vee b) \ldots$

a	b	$\neg a$	$\neg b$	$\neg a \wedge \neg b$	$a \vee b$	$\neg(a \vee b)$
V	V	F	F	F	V	F
V	F	F	V	F	V	F
F	V	V	F	F	V	F
F	F	V	V	V	F	V

- Georges Boole a exprimé la logique des propositions en termes algébriques
  - $a \vee b \Rightarrow a + b$
  - $a \wedge b \Rightarrow a.b$
  - $\neg a \Rightarrow \overline{a} = 1 - a$
- Claude Shannon a proposé : $V = 1$ et $F = 0$

Algèbre de Boole

- Il existes 16 fonctions à deux variables : toutes ne sont pas intéressantes

a	b	$a+b$	$\overline{a+b}$	ab	$\overline{ab}$	$a \oplus b$	$\overline{a \oplus b}$
0	0	0	1	0	1	0	1
0	1	1	0	0	1	1	0
1	0	1	0	0	1	1	0
1	1	1	0	1	0	0	1

- Ou exclusif (xor) $a \oplus b = a\overline{b} + \overline{a}b$
- Non ou (nor) $\overline{a+b}$
- Non et (nand) $\overline{ab}$

Opérations de base

- Toutes les fonctions peuvent s'exprimer à l'aide des trois opérations logiques et, ou, non
  - Avec n variables on peut construire 2^n fonctions
  - Une fonction à trois variables peut se décomposée en deux
    - $f(a,b,c) = f(0,b,c)$ si $a = 0$ et $f(a,b,c) = f(1,b,c)$ si $a = 1$
    - $f(a,b,c) = a.f(1,b,c) + \overline{a}.f(0,b,c)$
  - groupe logique complet = ensemble de fonction à partir desquels il est possible de réaliser toutes les fonctions {et,ou,non}...
  - Fonctions à n variables décomposables et fonctions à 2 variables exprimable avec {et,ou,non} $\Rightarrow$ récurrence ...
- L'algèbre de Boole se construit sur les booléens à partir des trois opérations internes : +, * et

Théorèmes et axiomes

- Théorème des constantes

$$a+0=a \quad a.0=0 \quad a \oplus 0 = a$$
$$a+1=1 \quad a.1=a \quad a \oplus 1 = \overline{a}$$

- Idempotence

$$a+a=a \quad a.a=a \quad a \oplus a = 0$$

- Complémentation

$$a+\overline{a}=1 \quad a.\overline{a}=0 \quad a \oplus \overline{a} = 1$$

- Commutativité

$$a+b=b+a \quad a.b=b.a \quad a \oplus b = b \oplus a$$

Théorèmes et axiomes

- Distributivité

$$a+(bc) = (a+b)(a+c)$$
$$a(b+c) = ab+ac$$

- Associativité

$$a+(b+c) = (a+b)+c = a+b+c$$
$$a(bc) = (ab)c = abc$$

- Théorème de De Morgan

$$\overline{ab} = \overline{a}+\overline{b}$$
$$\overline{a+b} = \overline{a}.\overline{b}$$

Table de vérité et équation

- Pour concevoir un circuit, il faut le modéliser.
  - on utilise une table de vérité
  - on obtient une équation qu'on simplifie (algébrique ou Karnaugh)
  - on trace le schéma électrique du circuit
  - éléments constitutifs d'un ordinateur : exclusivement portes nand ou bien nor (système complet)
- Table de vérité : présente toutes les combinaisons possibles des n entrés (2^n lignes) et les états de sortie correspondant
- Une fois la table de vérité écrite, il faut la transformer en équation logique
  - somme (fonction ou) de produit (fonction et)

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = \overline{a}.\overline{b} + a.(\overline{b} + b)$

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = \overline{a}.\overline{b} + a.(\overline{b} + b)$
$F(a,b) = \overline{\overline{a}.\overline{b} + a}$

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = \overline{a}.\overline{b} + a.(\overline{b} + b)$
$F(a,b) = \overline{\overline{a}.\overline{b} + a}$
$F(a,b) = (\overline{\overline{a}.\overline{b}}.\overline{a}))$

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

$$F(a,b) = \overline{\overline{\overline{a}.\overline{b} + a.(\overline{b} + b)}}$$
$$F(a,b) = \overline{\overline{\overline{a}.\overline{b}} + a}$$
$$F(a,b) = \overline{((\overline{\overline{a}.\overline{\overline{b}}.\overline{a}))}}$$
$$F(a,b) = \overline{(a+b).\overline{a}}$$

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

$$F(a,b) = \overline{\overline{\overline{a}.\overline{b} + a.(\overline{b} + b)}}$$
$$F(a,b) = \overline{\overline{\overline{a}.\overline{b}} + a}$$
$$F(a,b) = \overline{((\overline{\overline{a}.\overline{\overline{b}}.\overline{a}))}}$$
$$F(a,b) = \overline{(a+b).\overline{a}}$$
$$F(a,b) = \overline{a.\overline{a} + b.\overline{a}}$$

G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

$$F(a,b) = \overline{\overline{\overline{a}.\overline{b} + a.(\overline{b} + b)}}$$
$$F(a,b) = \overline{\overline{\overline{a}.\overline{b}} + a}$$
$$F(a,b) = \overline{((\overline{\overline{a}.\overline{\overline{b}}.\overline{a}))}}$$
$$F(a,b) = \overline{(a+b).\overline{a}}$$
$$F(a,b) = \overline{a.\overline{a} + b.\overline{a}}$$
$$F(a,b) = \overline{b.\overline{a}}$$

G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

$$F(a,b) = \overline{\overline{\overline{a}.\overline{b} + a.(\overline{b} + b)}}$$
$$F(a,b) = \overline{\overline{\overline{a}.\overline{b}} + a}$$
$$F(a,b) = \overline{((\overline{\overline{a}.\overline{\overline{b}}.\overline{a}))}}$$
$$F(a,b) = \overline{(a+b).\overline{a}}$$
$$F(a,b) = \overline{a.\overline{a} + b.\overline{a}}$$
$$F(a,b) = \overline{b.\overline{a}}$$
$$F(a,b) = \overline{b} + \overline{\overline{a}}$$

G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Équation : exemple

a	b	F(a,b)
0	0	1
0	1	0
1	0	1
1	1	1

$F(a,b) = 1$ si
$a = 0$ et $b = 0$ soit $\overline{a} = 1$ et
$\overline{b} = 1$ donc si $\overline{a}.\overline{b} = 1$
$a = 1$ et $b = 0$ soit $a = 1$ et
$\overline{b} = 1$ donc si $a.\overline{b} = 1$
$a = 1$ et $b = 1$ donc si $a.b = 1$
$\Rightarrow F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$

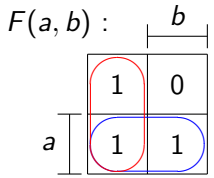
$$F(a,b) = \overline{a}.\overline{b} + a.(\overline{b} + b)$$
$$F(a,b) = \overline{\overline{a}.\overline{b} + a}$$
$$F(a,b) = \overline{(\overline{\overline{a}.\overline{\overline{b}}.\overline{a}})}$$
$$F(a,b) = \overline{(a+b).\overline{a}}$$
$$F(a,b) = \overline{a.\overline{a} + b.\overline{a}}$$
$$F(a,b) = \overline{b.\overline{a}}$$
$$F(a,b) = \overline{b} + \overline{\overline{a}}$$
$$F(a,b) = \overline{b} + a$$

Diagrammes de Karnaugh

- Table de Karnaugh : principe
  - départ : somme de produits où chaque produit doit contenir toutes les variables (pour simplifier) : $(a + \overline{a})b = b$
  - chaque colonne diffère de sa voisine d'un seul littéral
  - les tables de Karnaugh sont à deux dimensions : on regroupe des variables
  - on regroupe les 1 en morceaux rectangulaires
    - plus grands morceaux possibles
    - moins de morceaux possibles
    - nouveau morceau que s'il permet de regrouper des 1 non encore regroupés
    - la ligne du haut et du bas ainsi que colonne de droite et de gauche sont adjacentes
  - $\Rightarrow$ Morceau = produit de variable : variable et son inverse dans le même morceau = élimination de la variable

Karnaugh : Exemple 1

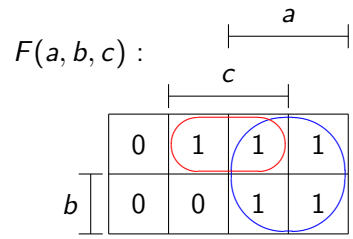
- $F(a,b) = \overline{a}.\overline{b} + a.\overline{b} + a.b$



- $a + \overline{b}$

Karnaugh : Exemple 2

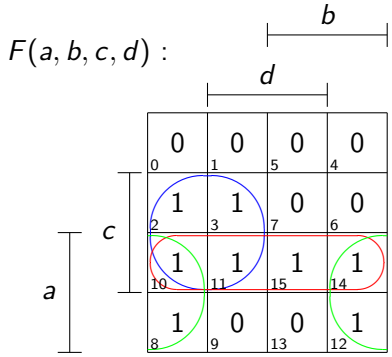
- $F(a,b,c) = \overline{a}.\overline{b}.c + a.\overline{b} + a.c$
  $F(a,b,c) = \overline{a}.\overline{b}.c + a.\overline{b}.(c + \overline{c}) + a.c.(b + \overline{b})$
  $F(a,b,c) = \overline{a}.\overline{b}.c + a.\overline{b}.c + a.\overline{b}.\overline{c} + a.b.c + a.\overline{b}.c$



- $a + \overline{b}.c$

Karnaugh : Exemple 3

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

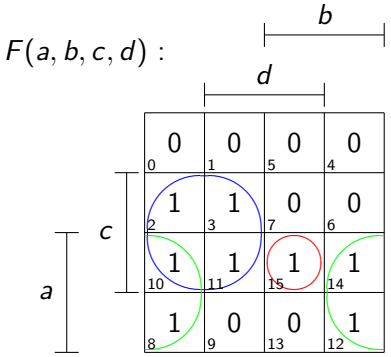


- $\overline{b}.c + a.b.c.d + a.\overline{d}$!!!!!!!
- PAS BON !

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules

Karnaugh : Exemple 3

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



- $\overline{b}.c + a.b.c.d + a.\overline{d}$!!!!!!!
- PAS BON !

G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules

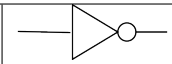
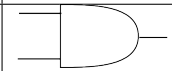
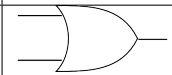
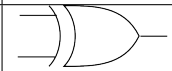
Portes et circuits logiques

- Un ordinateur travaille en base 2
  - Électroniquement 0 correspondait à une tension de 0 à 0,8V et 1 à une tension de 2,8 à 2,5V (tensions données par les constructeurs de composants)
  - Toute fonction binaire peut être représentée par une expression booléenne
  - Tout circuit électrique ou électronique à deux valeurs de tension peut être représenté par une expression booléenne
- Préférable de représenter les circuits par des symboles logiques et non par des expressions booléennes
  - Correspondance entre les différentes fonctions logiques $(+, *\ldots)$ et des symboles appelés portes logiques

G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

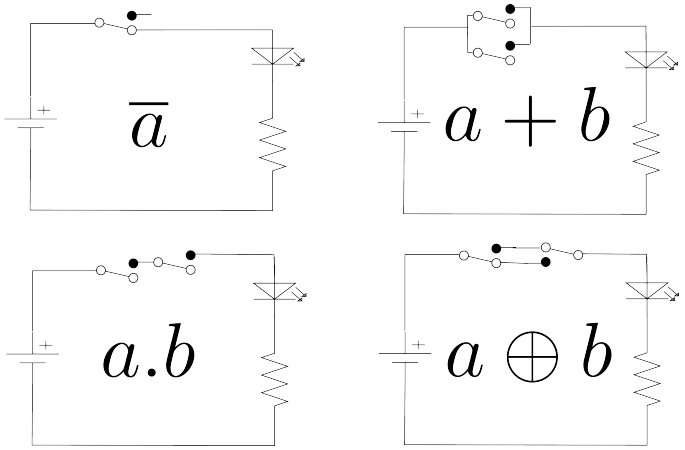
Algèbre de Boole Electronique Circuits combinatoires UAL Circuits séquentiels : bascules

Portes logiques

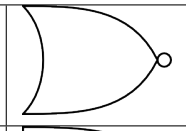
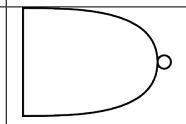
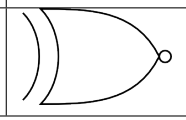
NON(a)		$\overline{a}$
AND(a,b)		$a+b$
OR(a,b)		$a.b$
XOR(a,b)		$a \oplus b$

G. Blin - Architecture des ordinateurs Algèbre de Boole et circuits logiques

Portes logiques et circuits

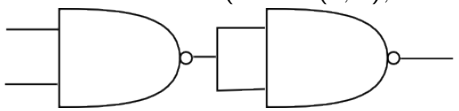


Portes logiques : mise en série

NOR(a, b)		$\overline{a + b}$
NAND(a, b)		$\overline{a.b}$
ID(a, b)		$\overline{a \oplus b}$

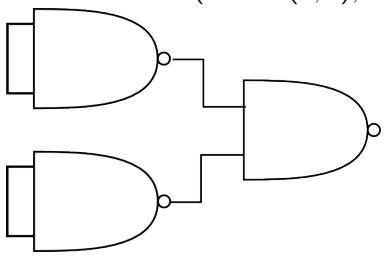
Système complet

- Porte NAND = la plus simple à réaliser du point de vue technologique.
- Possible de réaliser toutes les fonctions logiques en utilisant uniquement NAND
- AND : $a.b = \overline{\overline{a.b}.\overline{a.b}} = NAND(NAND(a, b), NAND(a, b))$



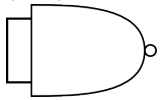
Système complet

- Porte NAND = la plus simple à réaliser du point de vue technologique.
- Possible de réaliser toutes les fonctions logiques en utilisant uniquement NAND
- OR : $a + b = \overline{\overline{a.a}.\overline{b.b}} = NAND(NAND(a, a), NAND(b, b))$



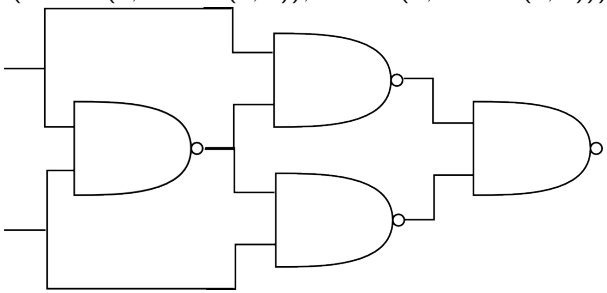
Système complet

- Porte NAND = la plus simple à réaliser du point de vue technologique.
- Possible de réaliser toutes les fonctions logiques en utilisant uniquement NAND
- NOT : $\overline{a} = \overline{a.a} = NAND(a, a)$



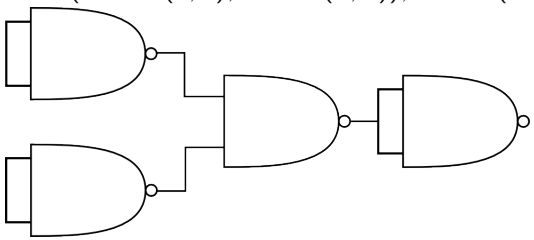
Système complet

- Porte NAND = la plus simple à réaliser du point de vue technologique.
- Possible de réaliser toutes les fonctions logiques en utilisant uniquement NAND
- XOR : $a \oplus b = \overline{\overline{a.\overline{a.b}}.\overline{b.\overline{a.b}}} = NAND(NAND(a, NAND(a, b)), NAND(b, NAND(a, b)))$



Système complet

- Porte NAND = la plus simple à réaliser du point de vue technologique.
- Possible de réaliser toutes les fonctions logiques en utilisant uniquement NAND
- NOR : $\overline{a + b} = \overline{\overline{a.\overline{a}.\overline{b}.\overline{b}}.\overline{\overline{a.\overline{a}.\overline{b}.\overline{b}}}} = NAND(NAND(NAND(a, a), NAND(b, b)), NAND(NAND(a, a), NAND(b, b)))$



Plan

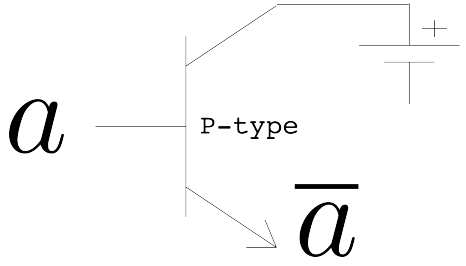
Transistor

- Un transistor est un interrupteur qui permet de réunir ou de séparer deux fils.
- L'état d'un transistor (passant ou non), est commandé par un troisième fil.
- Il existe deux types de transistors :
  - les transistors de type N ("non-passant par défaut"), pour lesquels le transistor est non passant quand le fil de commande est dans l'état 0,

  - et les transistors de type P ("passant par défaut"), pour lesquels le transistor est passant quand le fil de commande est dans l'état 0.

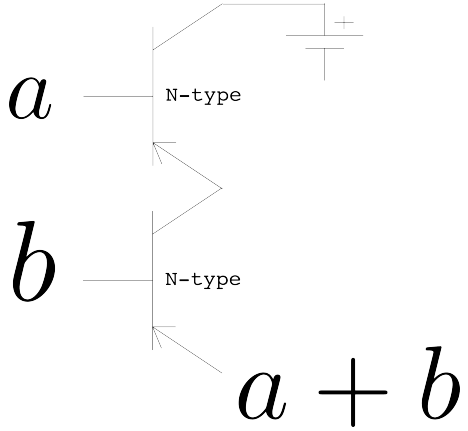
Circuits intégrés

- A partir de ce dispositif on peut réaliser des portes de base (NON, ET, OU ...)



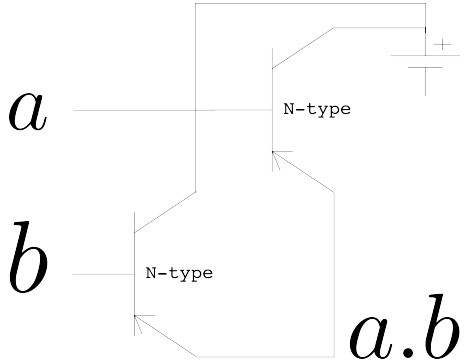
Circuits intégrés

- A partir de ce dispositif on peut réaliser des portes de base (NON, ET, OU ...)



Circuits intégrés

- A partir de ce dispositif on peut réaliser des portes de base (NON, ET, OU ...)



Circuits intégrés

- Les circuits intégrés (chip) peuvent être classifiés
  - SSI (Small Scale Integration) contenant moins de 100 portes
  - MSI (Medium) : entre 100 et 1000 portes
  - LSI (Large) : entre 1000 et 10^5 portes
  - VLSI (Very Large) : entre 10^5 et 10^7 portes
  - ULSI (Ultra Large) : plus de 10^7 portes
- Portes logiques : réalisées électroniquement par un ou deux transistors
- Plusieurs portes logiques forment un circuit logique = circuit intégré
  - circuits combinatoires : ne font que combiner les variables d'entrée selon une table de vérité
  - circuits séquentiels : construits à partir de circuits combinatoires + capacité de mémorisation
  - La réalisation d'un circuit passe par la recherche des expressions booléennes, puis par leur simplification (règles ou tableau de Karnaugh)

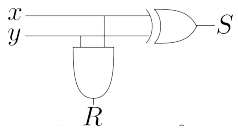
Plan

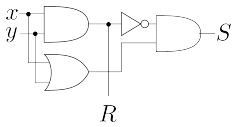
Demi-additionneur

- Addition de deux bits x et y

x	y	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

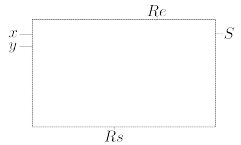
$R = xy$ $S = x \oplus y = (x + y).xy$





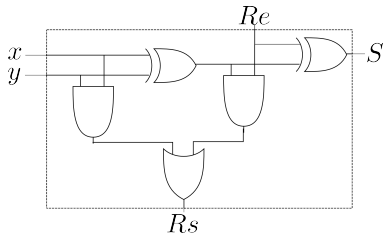
Additionneur complet

- Addition de deux nombres
  - addition bit à bit
  - considération de la retenue précédente
  - somme de 3 bits = additionneur complet
- Somme S
  - vaut 1 si entre x, y et re (retenue d'entrée) : le nombre de bits à 1 est impaire
- Retenue de sortie rs
  - vaut 1 si x et y valent 1, ou si l'un des deux vaut 1 alors que revaut 1



Additionneur complet

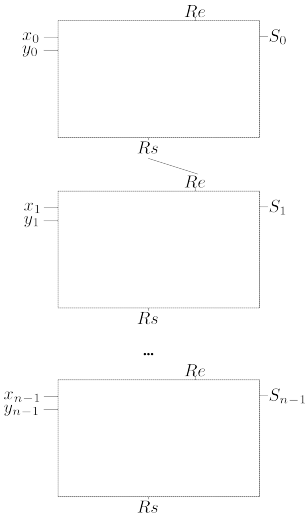
x	y	re	S	rs
1	1	0	0	1
1	1	1	1	1
1	0	0	1	0
1	0	1	0	1
0	1	0	1	0
0	1	1	0	1
0	0	0	0	0
0	0	1	1	0



- $S = re \oplus x \oplus y$
- $rs = xy + x.re + y.re$
- $rs = xy + re.(x \oplus y)$

Additionneur complet : propagation

- Forme la plus simple : propagation de retenue
  - le calcul sur chaque bit se fait de façon différée
  - rang 0 en premier puis rang 1 avec la retenue du rang 0
- Temps de calcul
  - si un additionneur 1 bit met 1T pour calculer la retenue et 1,5T pour le résultat
  - rs_0 à T et S_0 à 1,5T
  - rs_1 à 2T et S_1 à 2,5T
  - rs_{n-1} à nT et S_{n-1} à (n+0,5)T
  - Temps : (n+0,5)*T = linéaire



Additionneur complet : améliorations

- La lenteur de l'additionneur par propagation de retenue impose d'utiliser d'autres techniques pour des additionneurs ayant un nombre important de bits. Comme cette lenteur est due au temps nécessaire à la propagation de la retenue, toutes les techniques ont pour but d'accélérer le calcul des retenues. La première technique appelée anticipation de retenue consiste à faire calculer les retenues par un circuit extérieur.

Additionneur complet : améliorations

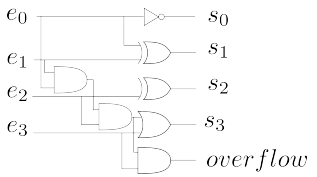
- Additionneur par anticipation de retenue
  - Afin de faciliter le calcul des retenues, on introduit deux quantités appelées G (pour Generate en anglais) et P (pour Propagate en anglais).
  - Soient $A = A_{n-1} \dots A_0$ et $B = B_{n-1} \dots B_0$ deux entrées de n bits. On note C_i la retenue de l'addition des i bits de poids faible de A et B. Pour accélérer le calcul des C_i, on introduit les deux quantités G_i et P_i associées aux entrées A_i et B_i par les formules suivantes.
    $G_i = A_i B_i$ et $P_i = A_i + B_i$
  - $C_{i+1} = G_i + P_i C_i$
  - $C_0 = G_0 + Re$
  - Par récursivité on peut calculer toutes les C_i en traversant seulement 3 portes logiques
  - Pb : il faut des portes logiques avec de nombreuses entrées

Additionneur complet : améliorations

- Additionneur par sélection de retenue
  - L'idée générale de l'additioneur par sélection de retenue est d'effectuer en parallèle le calcul avec une retenue de 0 et le calcul avec une retenue de 1 puis de sélectionner le bon résultat lorsque la retenue est enfin connue.
  - Le problème est que ca double le nombre de portes

Incrémenteur

- Ajouter ou retrancher 1 : opération fréquente d'un processeur
  - utiliser l'addition : utilisation non optimale
  - Exemple : incrémenteur 4 bits



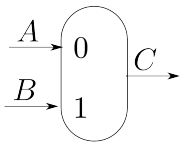
$s_0 = \overline{e_0}$
$s_i = \overline{e_i}$ *si* $e_{i-1} \dots e_0 = 1$; e_i *sinon*
donc $s_i = e_i \oplus (e_{i-1} x \dots x e_0)$

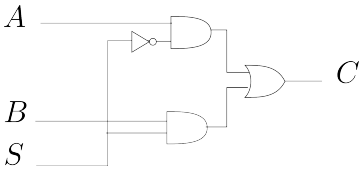
Multiplexeur

- Permet d'envoyer sur la sortie (C) l'état d'une entrée (A) ou de l'autre (B) en fonction d'un signal de sélection (S)

A	B	S	C
0	X	0	0
1	X	0	1
X	0	1	0
X	1	1	1

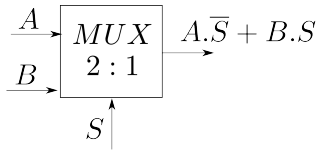
$C = A.(B + \overline{B}).S + (A + \overline{A}).B.S$
$C = A.\overline{S} + B.S$

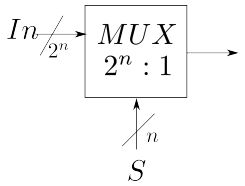




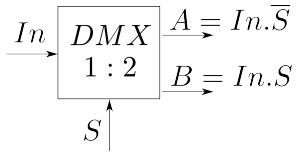
Multiplexeur/Demultiplexeur

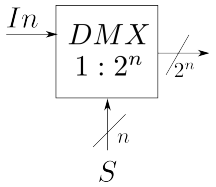
- Multiplexeur à deux et 2^n entrées (n bits pour coder 2^n val $\neq$)





- Demultiplexeur : aiguille l'entrée sur la sortie num S



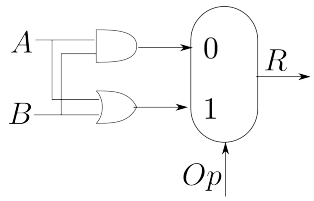


Plan

UAL : principe

- UAL : Unité Arithmétique et logique
  - effectue les opérations de bases (arithmétiques et logiques)
  - un code d'entrée détermine la partie du circuit qui va effectuer les opérations
- UAL 1 bit : opération ET / OU
  - en fonction d'un signal Op le circuit calcul a ET b (Op=0) ou bien a OU b (Op=1)

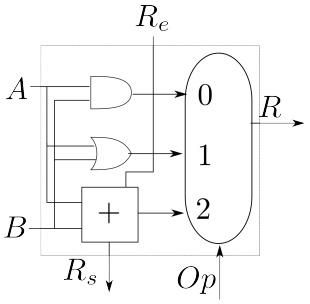
$S = a.b.\overline{Op} + (a + b).Op$
$\Rightarrow$ Multiplexeur



UAL : ET, OU, +

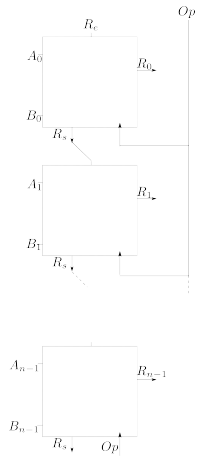
- UAL 2 bits : opération ET / OU / +
  - on ajoute un additionneur

Op1	Op0	S
0	0	a et b
0	1	a ou b
1	0	a+b
1	1	libre



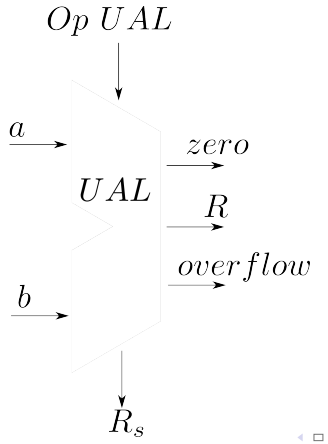
UAL n bits

- Traitement de données codées sur n bits



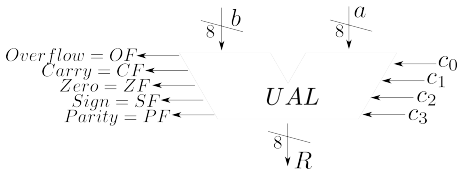
UAL : représentation

- UAL chargée des opérations
  - logiques : AND, OR, XOR, NOT, CMP, LSL, LSR, ASR (décalages)
  - arithmétiques : ADD, SUB, MUL, DIV, INC (+1), DEC (-1)



UAL : autre représentation

- UAL 8 bits
  - c_0, c_1, c_2, c_3 = sélection de l'opération
  - R = résultat
  - OF, CF, ZF, SF, PF = Drapeau (booléen) décrivant l'état du résultat



Plan

Circuits séquentiels : définition

- Définition
  - circuit séquentiel : pour la même combinaison des données, la sortie peut prendre diverses valeurs en fonction du temps
  - mémorisation des états passés
  - table de vérité : on trouve en plus des entrées, la valeur de sortie à l'état précédent
- Les circuits séquentiels de base sont les bascules
  - particularité : deux états stables = conservation de l'état de leur sortie même si la combinaison des signaux d'entrée l'ayant provoquée disparaît
- Horloge : composant passant d'un niveau haut à bas (0101010...)
  - bascules synchrones (avec horloge) insensibles aux bruits entre deux tops ou asynchrone (sans)

Bascules RS

- Circuit le plus simple
  - une entrée S (Set – mise à un) et une sortie R (Reset – remise à 0) qui permettent de changer l'état de la bascule
  - bascule asynchrone : pas d'horloge

R	S	Q'	Q	
0	0	0	0	Q=Q'
0	0	1	1	Q=Q'
0	1	X	1	Q=1
1	0	X	0	Q=0
1	1	X	?	interdit

Mise à 1 et 0 simutanément est impossible

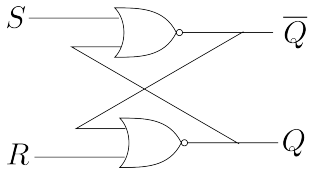
$$Q = \overline{R}.S + \overline{R}.\overline{S}.Q'$$
$$Q = \overline{R}.(S + \overline{S}.Q')$$
$$Q = \overline{R}.(S + Q')$$
$$Q = \overline{\overline{R}.(S + Q')}$$
$$Q = \overline{R + \overline{(S + Q')}}$$

Bascules RS

- Effet mémoire représenté par le retour des sorties
- Bascules RS fréquemment utilisées dans les circuits antirebond.

R	S	Q'	Q	
0	0	0	0	Q=Q'
0	0	1	1	Q=Q'
0	1	X	1	Q=1
1	0	X	0	Q=0
1	1	X	?	interdit

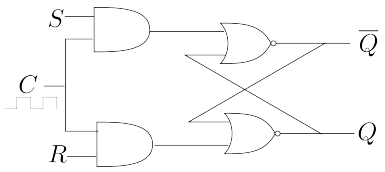
$$\overline{Q} = R.\overline{S} + \overline{R}.\overline{S}.\overline{Q'}$$
$$\overline{Q} = \overline{S}(R + \overline{R}.\overline{Q'}) = \overline{S}(R + \overline{Q'})$$
$$\overline{Q} = \overline{\overline{S}(R + \overline{Q'})} =$$
$$\overline{S + \overline{(R + \overline{Q'})}}$$



Bascules RSC ou RSH

- Bascule RS asynchrone : deux problèmes
  - sensibilité aux bruits parasites (prise en compte des entrées à tout moment)
  - synchronisation (quand peut-on prendre les données)
- Solution : R et S pris en compte à des instants déterminés à l'aide d'une horloge (période plus grande que le temps de stabilisation)

C	R	S	Q
0	X	X	Q'
1	0	0	Q'
1	0	1	1
1	1	0	0
1	1	1	?

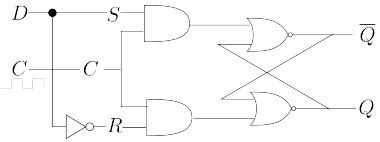


Bascules DH

- Pour supprimer l'état indéterminé (Horloge ou Clock)
  - on utilise un signal D qui va être mémorisé dans la bascule RS
  - D permet de fabriquer deux signaux R et S (différents de 11)
  - C : signal d'horloge permet de mémoriser le signal dans une plage de temps fixée (quand C vaut 1)

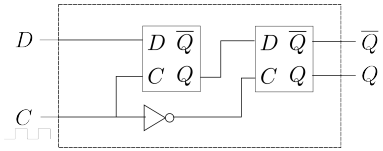
C	D	Q'	Q	
0	X	0	0	Q=Q'
1	X	1	1	Q=Q'
1	1	X	1	S=1 ;Q=1
1	0	X	0	R=1 ;Q=0

$$R = C.\overline{D}$$
$$S = C.D$$



Bascules DH sur front

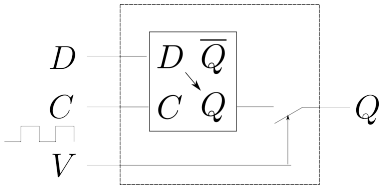
- On mémorise à un instant précis
  - front descendant : signal d'horloge passe de 1 à 0
  - fonctionnement de bascule synchrone



- Signal d'horloge = 1 : D est stocké dans la 1ère bascule
- Passage à 0 : le signal issu de la 1ère bascule est stockée dans la 2ème
- DH sur front montant ?

Bascules D avec validation

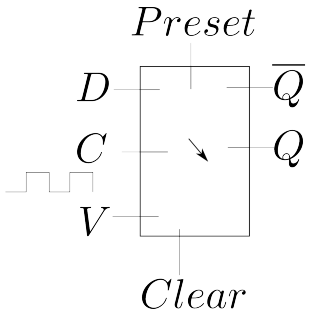
- Branchement de plusieurs bascules sur un bus de données
  - attention : une seule sortie validée à un instant donnée
  - on ajoute, en sortie de bascule, un interrupteur commandé par un signal de validation
  - la sortie de bascule est reliée à l'extérieur si Val = 1



V	C	D	Q
1	X	X	Q'
1	↘	1	1
1	↘	0	0
0	X	X	?

Bascules complètes

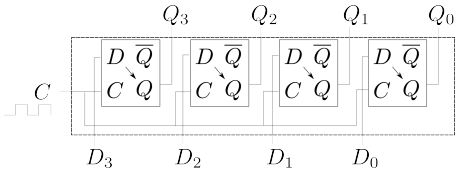
- Elles possèdent à la fois des entrées
  - asynchrones : PRESET (=0 mise à 1) et CLEAR (=0 mise à 0)
  - synchrones : changement de sortie qu'à des moments précis (H,D)
- Elles constituent les mémoires statiques de l'ordinateur



V	C	D	Preset	Clear	Q
1	X	X	1	1	Q'
1	↘	1	1	1	1
1	↘	0	1	1	0
1	X	X	0	1	1
1	X	X	1	0	0
1	X	X	0	0	?
0	X	X	X	X	?

Registres

- Application des bascules : utilisées pour leurs effets mémoire, chacune permet de stocker un bit
  - Registres
    - parallèles synchrones (bascules DH sur front montant)
    - parallèles asynchrones (PRESET et CLEAR)
    - Série (bascules connectées en séries) : pour transmettre des informations n bit sur un seul fil : modem, imprimante série...
  - Compteurs binaires
  - Décalage
  - ...



Registres

- Application des bascules : utilisées pour leurs effets mémoire, chacune permet de stocker un bit
  - Registres
    - parallèles synchrones (bascules DH sur front montant)
    - parallèles asynchrones (PRESET et CLEAR)
    - Série (bascules connectées en séries) : pour transmettre des informations n bit sur un seul fil : modem, imprimante série...
  - Compteurs binaires
  - Décalage
  - ...