Introduction à l'informatique

pour les mathématiques, la physique et les sciences computationnelles

Yann Thorimbert

Chapitre 4
Algèbre de Boole et circuits logiques (partie 2)

Yann Thorimbert

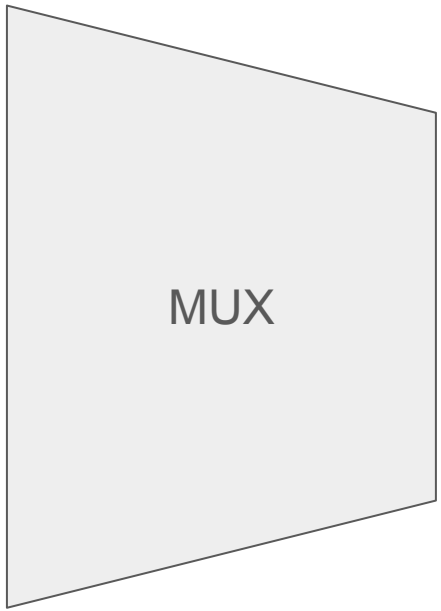
Chapitres du cours

1. Origines des ordinateurs et des réseaux informatiques
2. Codage des nombres
3. Codage des médias
4. **Circuits logiques** ←
5. Architecture des ordinateurs
6. Conception et exécution de programmes
7. Algorithmique, programmation et structures de données

Circuits combinatoires

- Circuits combinant plusieurs portes logiques (déjà vu !)
- Limite entre porte logique et circuit logique dépend des auteurs.
- Nous allons ici voir deux circuits combinatoires importants :
  - Le multiplexeur
  - L’additionneur

Le multiplexeur



Multiplexeur à 2 voies

- Il est courant de vouloir exprimer des **conditions** du type : "*Si S est faux, alors la valeur de sortie est A, sinon la valeur de sortie est B.*"

- Permet d'effectuer un **choix** entre A et B grâce à S.

- Équivalent électronique du if/else de programmation :

```
if S == 0:
    Out = A
else:
    Out = B
```

Multiplexeur à 2 voies

- Il est courant de vouloir exprimer des conditions du type : "*Si S est faux, alors la valeur de sortie est A, sinon la valeur de sortie est B.*"

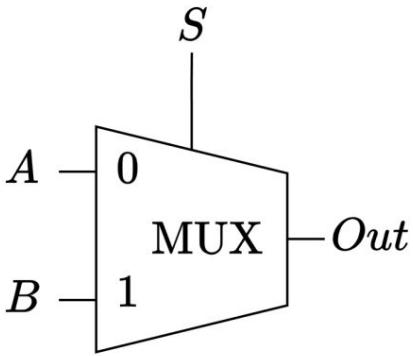
- Permet d'effectuer un choix entre A et B grâce à S.

- Bien entendu, on peut exprimer cela au travers d'une table de vérité.
  Trois entrées : S, A, B.
  Une sortie : Out.

- Vocabulaire : S est la **ligne de contrôle**, A et B sont les **lignes de données**.

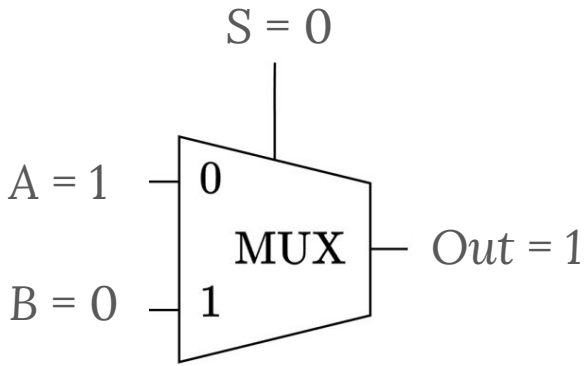
Multiplexeur à 2 voies

- On décide que $S=0$ correspond à la sélection de A et $S=1$ correspond à la sélection de B.
- On obtient en sortie une copie de l'entrée sélectionnée.
- Symbole :

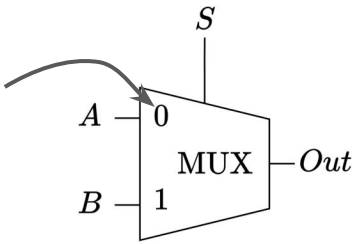


S	A	B	Out

Multiplexeur à 2 voies | **Exemple**



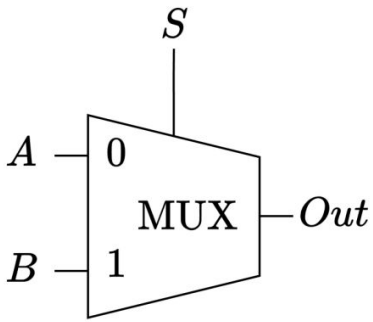
Ne veut **pas** dire que A = 0



S	A	B	Out
0	1	0	1

Multiplexeur à 2 voies

$Out = SB + !SA$

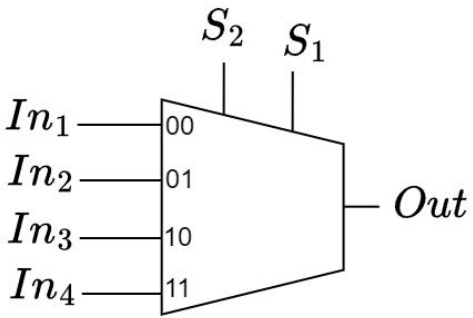


S	A	B	Out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Multiplexeur à *k* voies

Exemple pour *k* = 4 lignes de données.

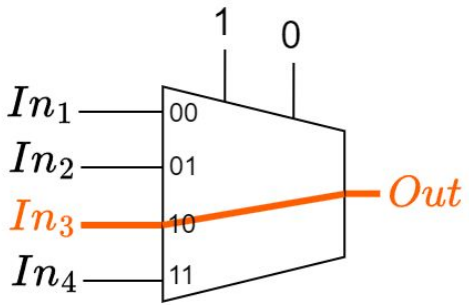
S	Out
00	In_1
01	In_2
10	In_3
11	In_4



Multiplexeur à *k* voies

Exemple pour *k* = 4 lignes de données.

S	Out
00	In_1
01	In_2
10	In_3
11	In_4

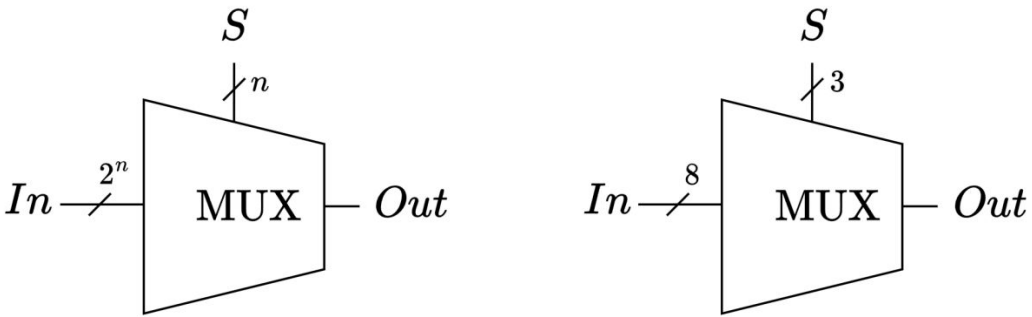


Multiplexeur à *k* voies | **Taille de la ligne de contrôle**

- Sélectionne l'une des *k* entrées, selon le signal de contrôle *S*.
- Question : quelle doit être la taille (nombre de bits) de *S* afin de pouvoir sélectionner n'importe laquelle des *k* entrées ?
- Réponse :
- Exemple :

Multiplexeur à *k* voies | **Symboles**

Plutôt que de dessiner toutes les voies sur les schémas, on indique le nombre de bits concernés avec une barre oblique. Exemples :



Multiplexeur à *k* voies | **Parcours du signal**

Question : combien de portes logiques le signal doit-il traverser pour générer un output ?

Multiplexeur à *k* voies | **Parcours du signal**

- Question : combien de portes logiques le signal doit-il traverser pour générer un output ?

- Réponse : ce nombre est proportionnel au nombre de bits de *S*.

- Raison : multiplexeur à *k* voies peut être vu comme un arbre de multiplexeurs à 2 voies. Le nombre d'étages de cet arbre est de $\log_2(k)$.

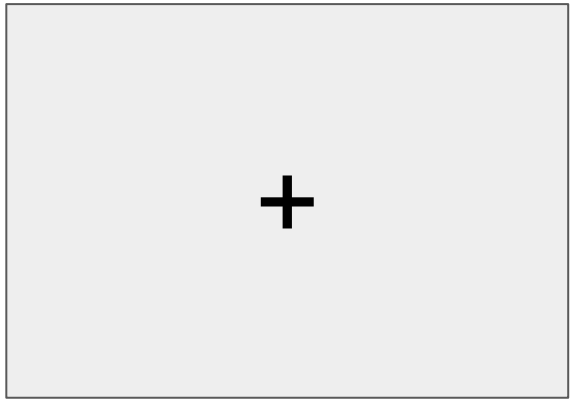
Multiplexeur à *k* voies | **Parcours du signal**

- Le signal doit traverser un nombre de portes proportionnel à log(*k*) ou encore au nombre de bits de *S*.

- Ce fait trouvera son importance plus tard dans le cours, car des multiplexeurs sont utilisés pour sélectionner des données dans la mémoire de l'ordinateur.

  La vitesse d'accès aux éléments de la mémoire est d'autant plus lente que les adresses sont longues.

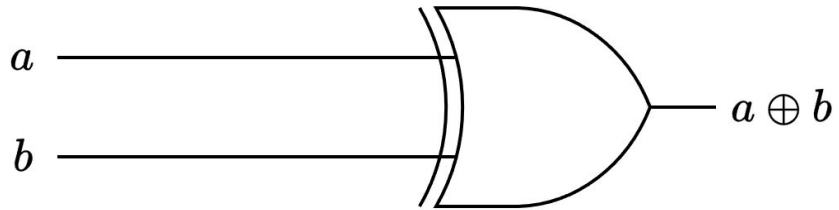
L’additionneur



Addition de deux bits

Ne prend pas en compte la retenue. Comment modifier ce circuit pour également avoir un output de retenue ?

a	b	a ⊕ b
0	0	0
0	1	1
1	0	1
1	1	0

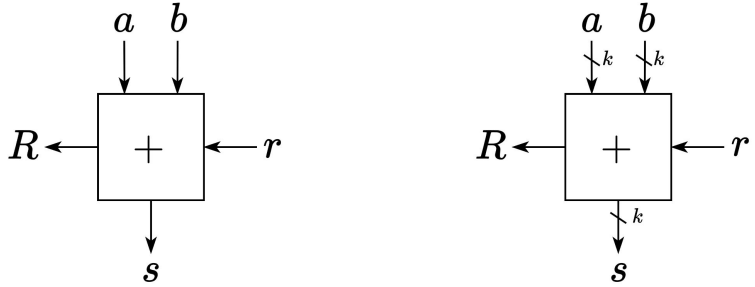


Additionneur | **Cas général (additionneur complet)**

- En plus de a_i et b_i, Le circuit associé à chaque colonne reçoit en input le reste de la colonne de droite, $r_i = R_{i-1}$.

- Le circuit de chaque colonne output la somme s_i et transmet le reste R_i à la colonne de gauche.

- Symbole de l'additionneur complet :

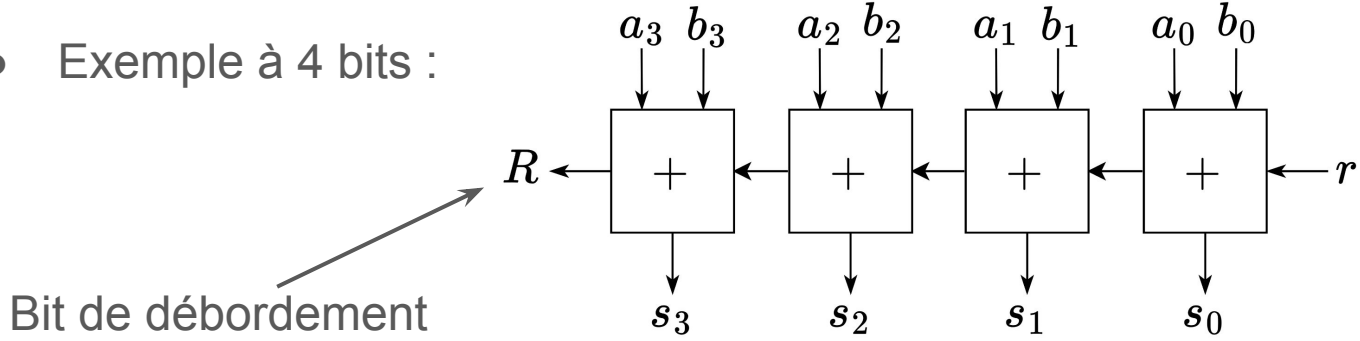


Additionneur | **Cas général**

- En plus de a_i et b_i, Le circuit associé à chaque colonne reçoit en input le reste de la colonne de droite, $r_i = R_{i-1}$.

- Le circuit de chaque colonne output la somme s_i et transmet le reste R_i à la colonne de gauche.

- Exemple à 4 bits :

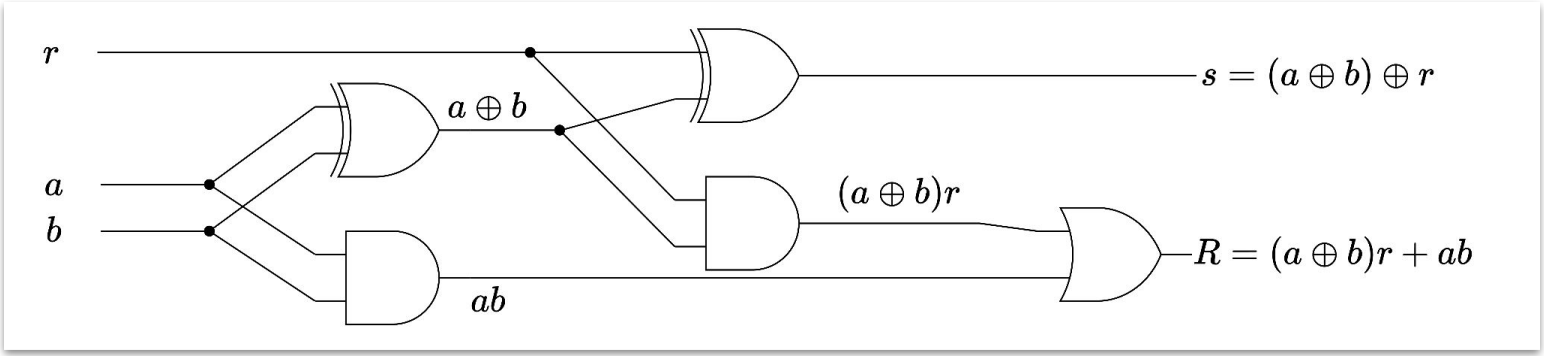


Additionneur | **Table de vérité pour le cas général 1 bit**

r	a	b	s	R
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Additionneur | **Additionneur complet**

- R = ab + r(a ⊕ b)
- s = r ⊕ (a ⊕ b)
- Additionneur complet :



Soustracteur | **Rappel sur le complément à 2**

- a - b = a + (-b).
- Profite des avantages du complément à 2.
- Le codage de (-a) est obtenu comme flip(a) + 1.

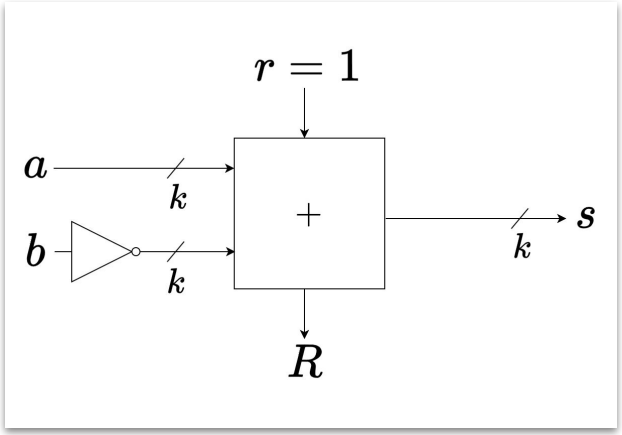
Soustracteur

- a - b = a + (-b) = a + flip(b) + 1

Fourni au circuit comme un "reste initial".

flip(b) = NOT(b).

- Soustracteur :



Note sur les optimisations

- Dans notre additionneur complet, chaque additionneur 1-bit doit attendre le résultat de son voisin avant d'effectuer son calcul, car il a besoin de la retenue.

- Dans les ordinateurs modernes, les additionneurs utilisent des techniques avancées pour anticiper les retenues et gagner du temps.

Logique séquentielle et éléments de mémoire

- Nous reviendrons en fin de semestre sur la façon de réaliser des circuits capables de **mémoriser des valeurs**.

- Cela se fera grâce à des circuits dits "**séquentiels**", où une notion de temps intervient.