Information, Calcul et Communication: Cours 12

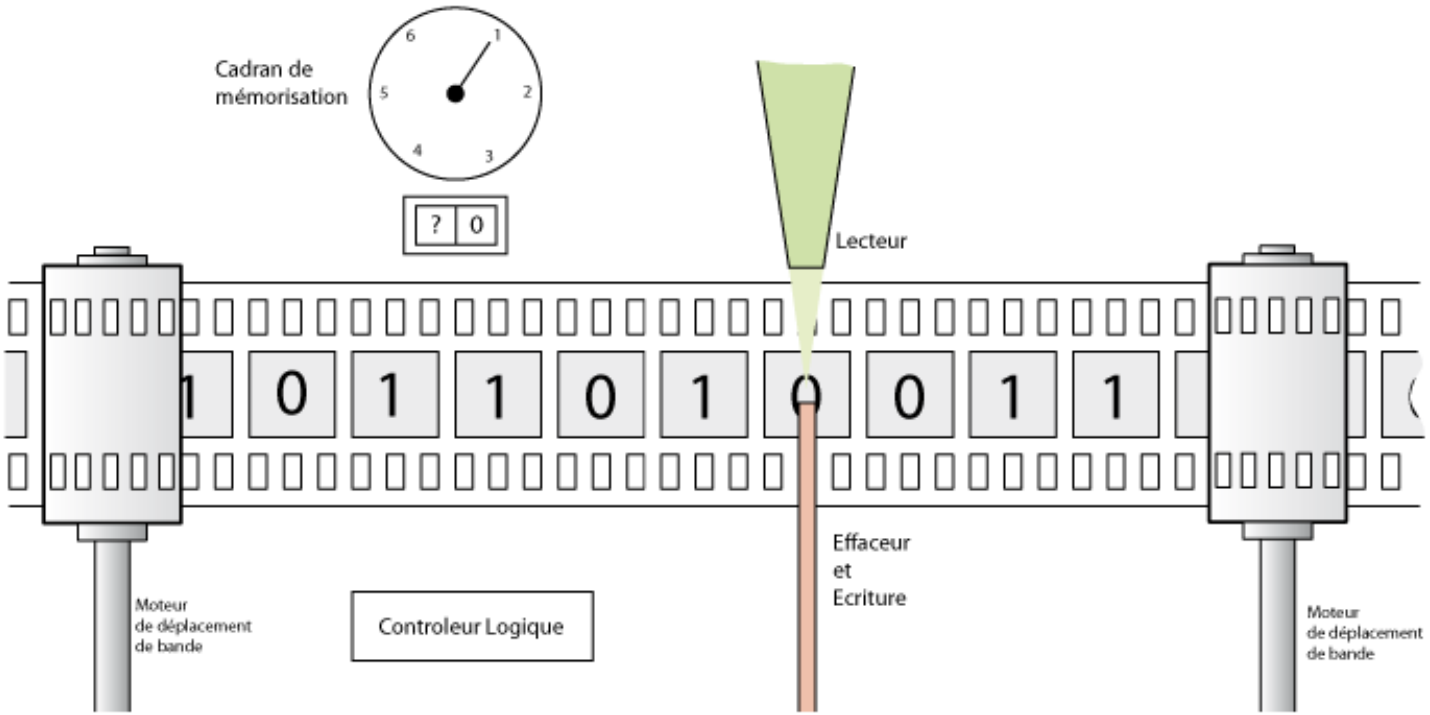
Résumé de la dernière leçon: correction d'erreurs

- principe: se protéger des erreurs en ajoutant de la redondance
- il y a des compromis à faire entre:
  - ajouter un minimum de redondance
  - transmettre un maximum d'information
  - se protéger d'un maximum d'erreurs

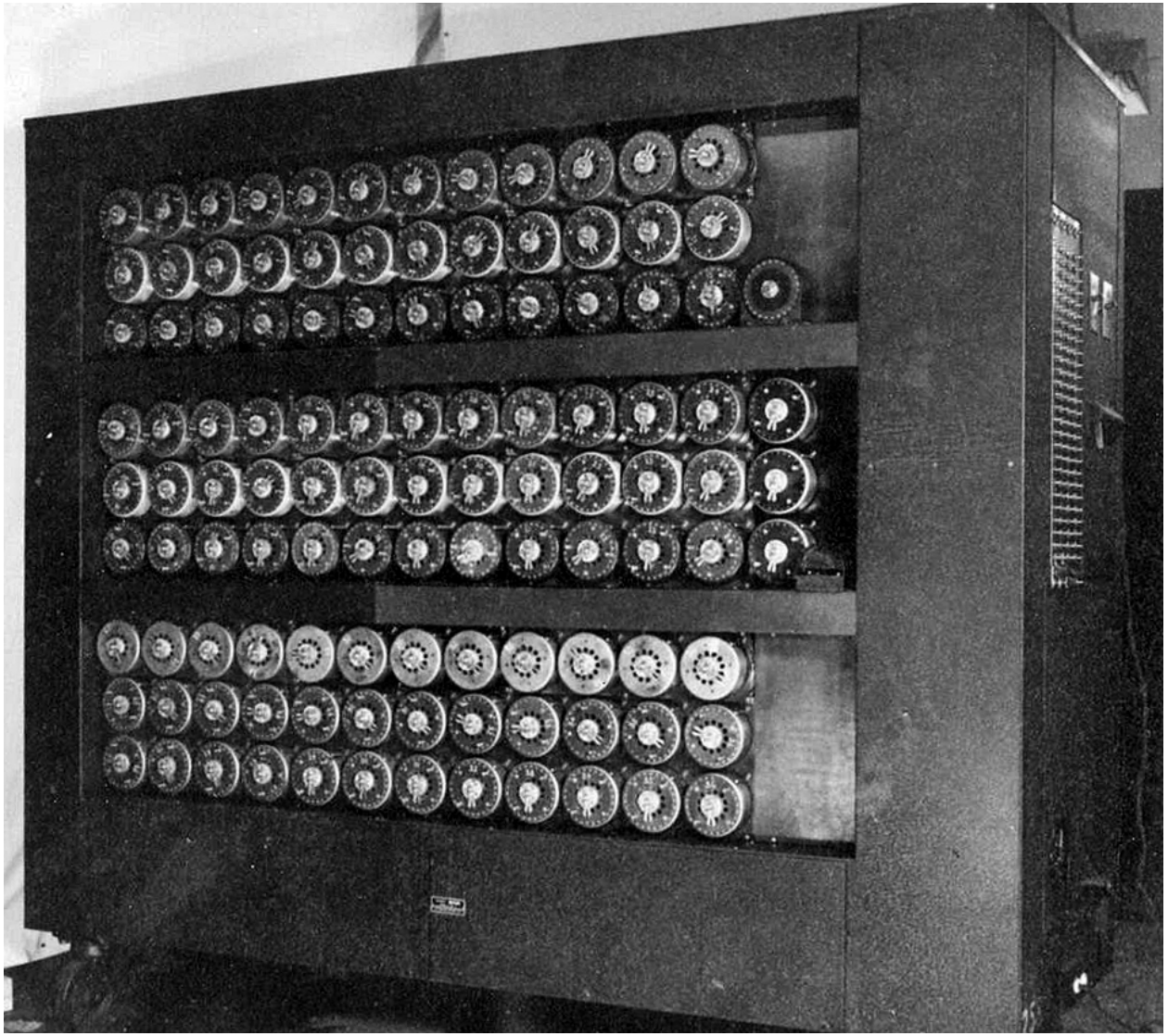
Plan de cette leçon: fonctionnement interne d'un ordinateur

- machine de Turing et architecture de Von Neumann
- enregistrement physique des 0 et des 1
- opération élémentaires et additions

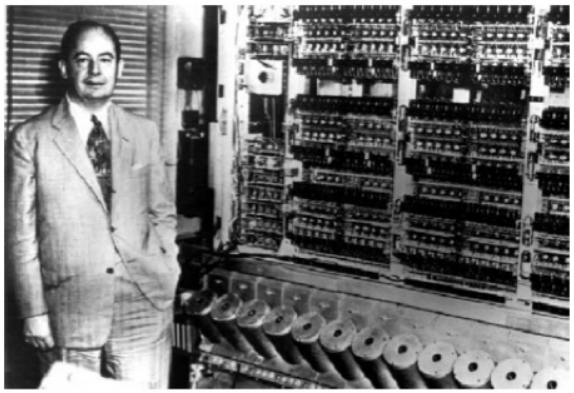
La machine de Turing

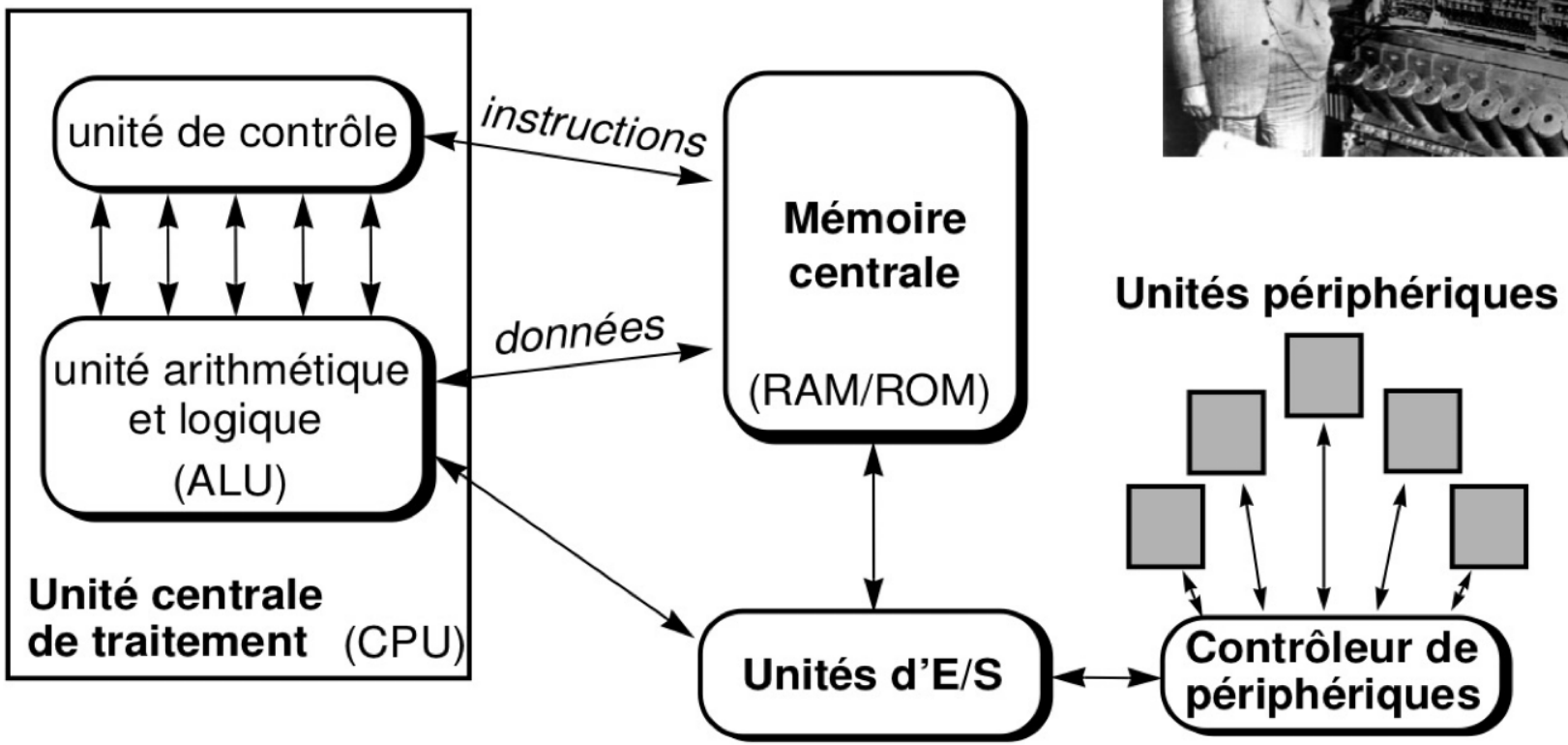


La vraie machine de Turing: la “bombe”



L'architecture de Von Neumann

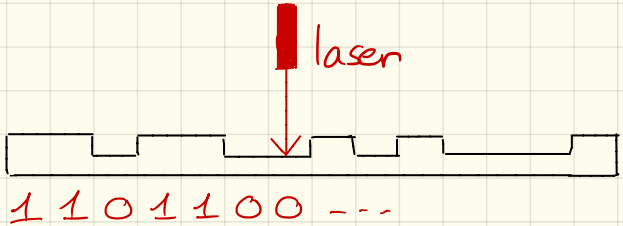




Comment enregistrer physiquement des bits 0 & 1 ?

1. Sur un CD ou un DVD

De profil :



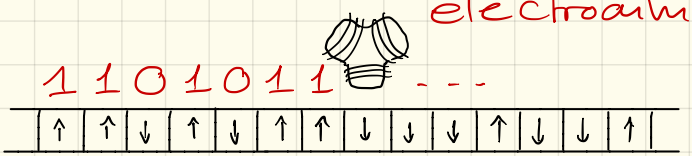
Les 0 et les 1 sont enregistrés comme des successions de différences de niveau dans la matière plastique du CD. Lorsque le laser parcourt le CD, la réflexion est différente suivant qu'il y a une dépression (0) ou non (1).

Avantage : mémoire solide !

Défaut : mémoire difficilement accessible et non modifiable

2. Dans un disque dur ou sur une bande magnétique

Ici, les 0 et les 1 sont enregistrés comme des moments magnétiques de petits composants métalliques, orientés dans un sens ou un autre :



Les valeurs (+/-) de ces moments magnétiques sont lues et/ou modifiées lors du passage d'un électroaimant placé sur une tête de lecture/écriture qui parcourt le disque ou la bande magnétique.

Avantages : { lecture/écriture
mémoire "solide"

Désavantage : accès difficile

3. Dans la mémoire vive d'un ordinateur

Les 0 et les 1 sont représentés par des tensions différentes (typiquement, $U_0 = 0\,V$ et $U_1 = 5\,V$) dans un circuit électrique.

Avantages : { lecture/écriture
accès facile

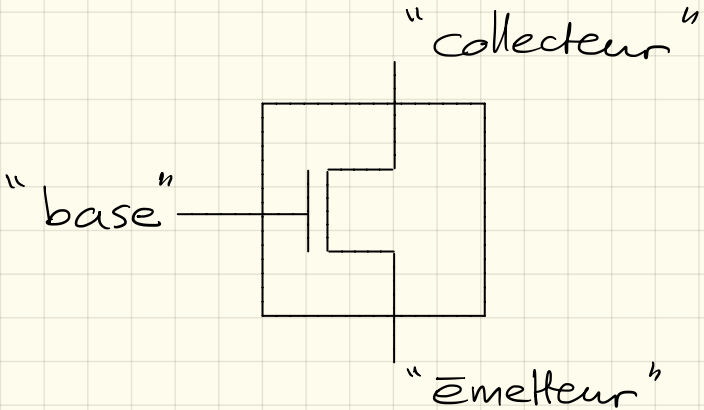
Désavantage : mémoire volatile (?)

Notre but aujourd'hui : Comprendre comment les ordinateurs manipulent des bits 0 & 1 avec des circuits électriques pour effectuer des opérations élémentaires (logiques, additions, multiplications, ...)

Le transistor (Bardeen, Brattain & Shockley, 1947)

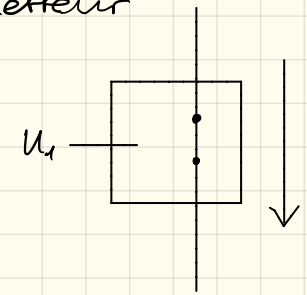
Ce composant, qui est à la base de toute l'électronique moderne, a remplacé avantageusement les relais électro-mécaniques et les tubes à vide utilisés dans les premiers ordinateurs à la même époque. (→ miniaturisation)

Schéma :

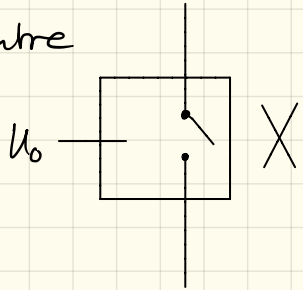


Principe (transistor n-mos, symbole )

- Si la tension à la base est haute ($U_1 = 5V$) alors le courant passe entre l'émetteur et le collecteur :



- Si la tension à la base est basse ($U_0 = 0V$), alors le courant ne passe pas entre l'émetteur est le collecteur :

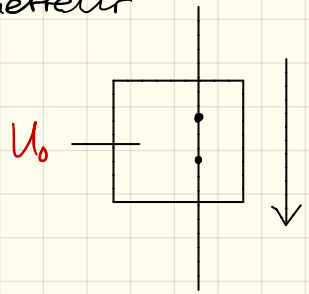


interrupteur !

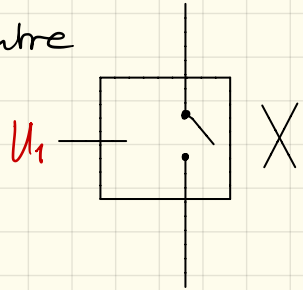
Principe (transistor p-mos, symbole )

Il se produit ici le contraire:

- Si la tension à la base est basse ($U_0 = 0V$), alors le courant passe entre l'émetteur et le collecteur:

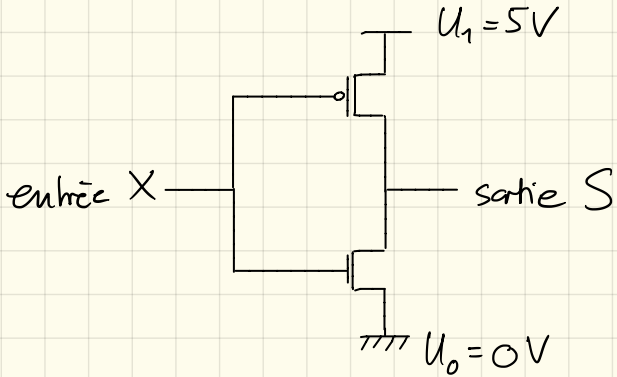


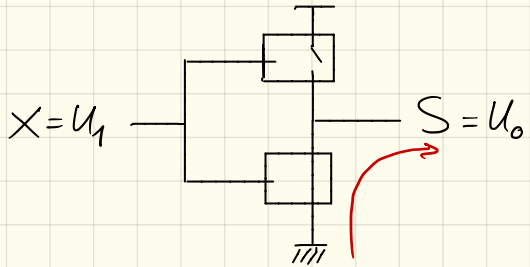
- Si la tension à la base est haute ($U_1 = 5V$) alors le courant ne passe pas entre l'émetteur est le collecteur:

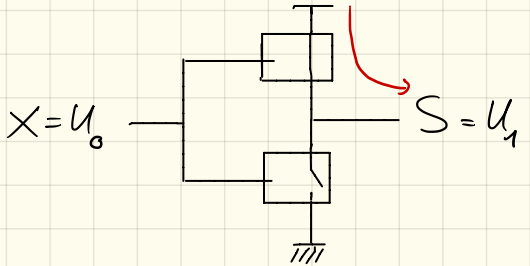


Création d'un inverseur

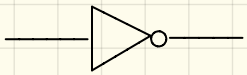
Si on identifie U_0 comme "0" et U_1 comme "1", on peut créer un inverseur à l'aide d'un transistor n-mos et d'un transistor p-mos :







Symbole :

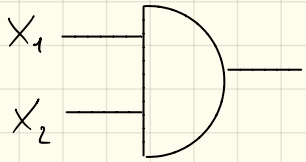


En interprétant "0" comme "faux" et "1" comme "vrai", on a crée ici une porte logique : $S = \text{NON } X$

(i.e., S est faux ssi X est vrai ; S est vrai ssi X est faux)

De manière similaire, on peut créer d'autres portes logiques :

- La porte ET :



(S est vrai ssi X_1 et X_2 sont tous les deux vrais)

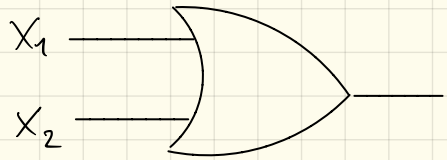
Table de vérité :

X_1	X_2	S
0	0	0
0	1	0
1	0	0
1	1	1

Note :

$S = X_1 \cdot X_2$ aussi

- La porte OU :



$S = X_1$ ou X_2 (S est vrai ssi X_1 ou X_2 est vrai (non-exclusif))

Table de vérité :

X_1	X_2	S
0	0	0
0	1	1
1	0	1
1	1	1

Note :

$S \neq X_1 + X_2$

Avec ces 3 portes de base (NON, ET, OU), on peut créer tous les circuits possibles, et donc effectuer toutes les opérations possibles !

Exemple 1 : Additionner deux bits (sans retenue)

On aimerait créer un circuit avec entrées X_1, X_2 et sortie S dont la table de vérité soit :

X_1	X_2	S
0	0	0
0	1	1
1	0	1
1	1	0

$S = X_1 + X_2$ (modulo 2)

$= X_1 \oplus X_2$

("xor" = ou exclusif)

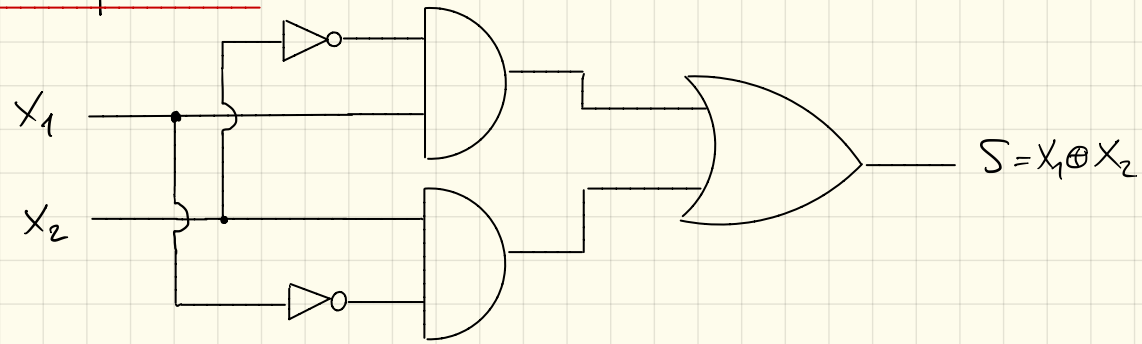
Pour créer ce circuit, remarquer que :

S est vrai ssi (X_1 est vrai et X_2 est faux)

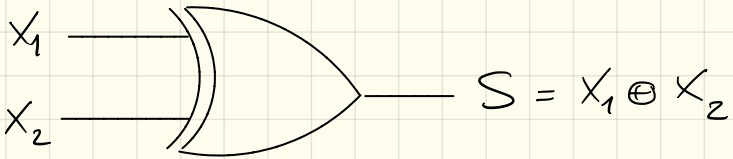
ou (X_1 est faux et X_2 est vrai)

Autrement dit : $S = (X_1$ ET NON $X_2)$ ou (NON X_1 ET $X_2)$

Circuit correspondant :

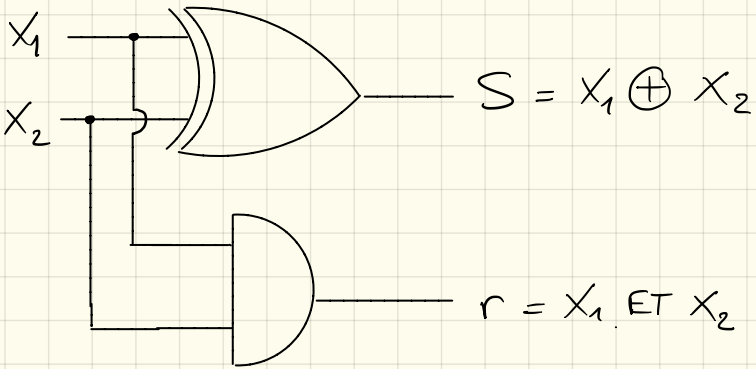


Symbole résumant ce nouveau circuit :

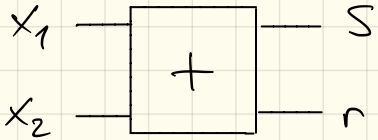


Exemple 2 : Additionner deux bits (avec retenue)

On aimerait maintenant créer un circuit avec entrées X_1, X_2 et sortie $S = X_1 \oplus X_2$, ainsi qu'une retenue $r = 1$ ssi $X_1 = 1$ et $X_2 = 1$:



Symbole :



Exercice : créer un circuit avec une retenue de plus en entrée (pourquoi donc ? → cf. pages suivantes)

Notre but : additionner des nombres !

Rappel :

avec des nombres entiers :

$$\begin{array}{r} {\scriptstyle 1\ 1\ \ } \\ 57 \\ +\ 43 \\ \hline = 100 \end{array}$$

avec des bits :

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ \ 1\ 1\ \ } \\ 111001 \\ +\ 101011 \\ \hline = 1100100 \end{array}$$

La règle d'addition est la même !

Il faut juste se rappeler que $1+1=10$

et $1+1+1=11$ en binaire.

Additionneur sur 8 bits:

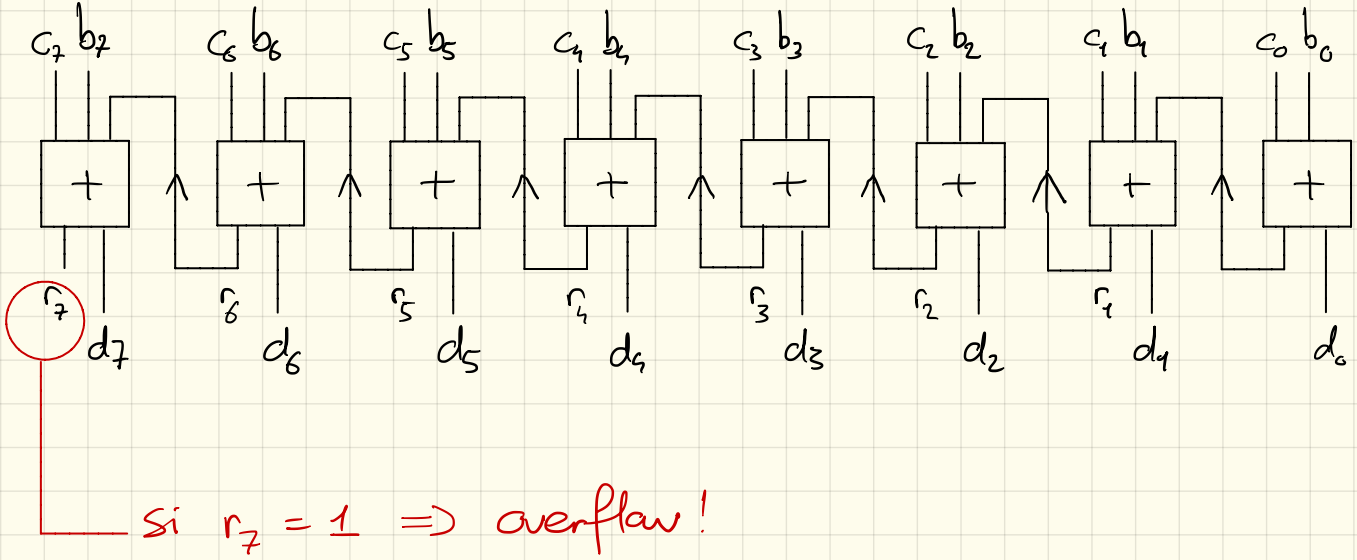
effectuer $b_7 b_6 b_5 \cdots b_1 b_0$

$+ \; c_7 c_6 c_5 \cdots c_1 c_0$

$= d_7 d_6 d_5 \cdots d_1 d_0$

Ex: 00111001 + 00101011 = 01100101

(11 11)



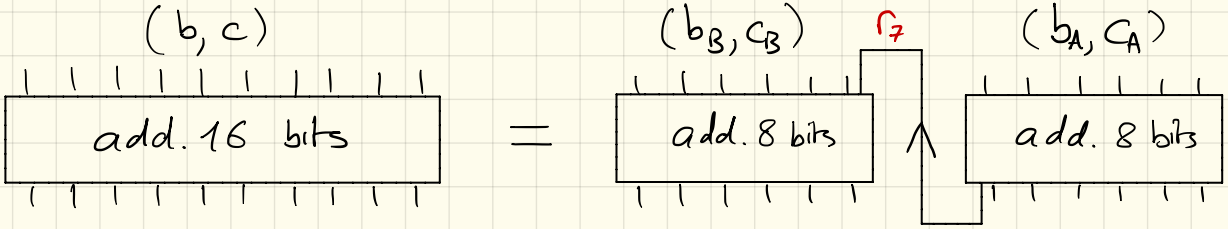
si $r_7 = 1 \Rightarrow$ overflow !

Optimisation : calcul en parallèle

La méthode décrite précédemment fonctionne bien, mais si on augmente le nombre de bits, elle est un peu lente, car les opérations d'addition doivent s'effectuer les unes à la suite des autres à cause des retenues successives ($\longrightarrow$ complexité temporelle $O(n)$ si n est le nombre de bits).

Dans les pages suivantes, on montre comment accélérer le processus avec p. ex $n = 16$ bits

1. Remarquer que :



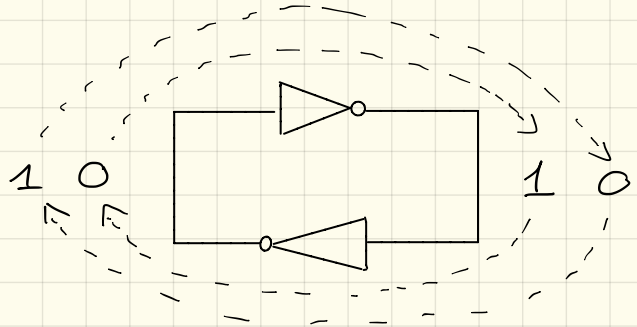
2. Au lieu d'effectuer toutes les opérations en séquence, effectuer en parallèle :

- une addition sur 8 bits de b_A et c_A
- une addition sur 8 bits de b_B et c_B avec $r_7 = 0$ en entrée
- une addition sur 8 bits de b_B et c_B avec $r_7 = 1$ en entrée

3. • Lire la valeur de la retenue r_7 résultant du calcul de l'addition de b_A et c_A ;
• ne garder que le calcul de l'addition de b_B et c_B ayant pris en compte la bonne valeur de r_7 en entrée ;
• rassembler finalement le résultat des deux additions.

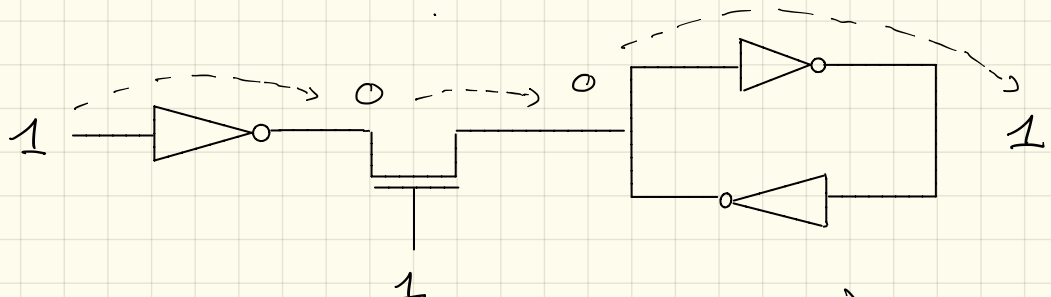
4. Ainsi, on a effectué un calcul (sur 8 bits) pour rien, mais le temps d'exécution est deux fois plus rapide !

Et pour finir : comment garder un bit 0 ou 1 en mémoire dans un circuit électrique ? (mémoire vive)



circuit stable !

Ecrire dans la mémoire :



(tension haute ⇒ "écrire")