

2 L'information binarisée

Peter Schlagheck

Université de Liège

Ces notes ont pour seule vocation d'être utilisées par les étudiants dans le cadre de leur cursus au sein de l'Université de Liège. Aucun autre usage ni diffusion n'est autorisé, sous peine de constituer une violation de la Loi du 30 juin 1994 relative au droit d'auteur.

2 L'information binarisée

2.1 Le système décimal et le système binaire

2.2 La réalisation du bit avec des circuits intégrés

2.3 L'octet

2.1 Le système décimal et le système binaire

Dans le système décimal habituel, les nombres sont encodés avec les chiffres 0, 1, 2, ..., 9

Exemple:

$$\begin{aligned} 325 &= 3 \times 100 + 2 \times 10 + 5 \times 1 \\ &= 3 \times 10^2 + 2 \times 10^1 + 5 \times 10^0 \end{aligned}$$

2.1 Le système décimal et le système binaire

Dans le système décimal habituel, les nombres sont encodés avec les chiffres 0, 1, 2, ..., 9

Exemple:

$$\begin{aligned} 325,172 = & 3 \times 10^2 + 2 \times 10^1 + 5 \times 10^0 \\ & + 1 \times 10^{-1} + 7 \times 10^{-2} + 2 \times 10^{-3} \end{aligned}$$

2.1 Le système décimal et le système binaire

Dans le système décimal habituel, les nombres sont encodés avec les chiffres 0, 1, 2, ..., 9

Exemple:

$$\begin{aligned} 325,172 = & 3 \times 10^2 + 2 \times 10^1 + 5 \times 10^0 \\ & + 1 \times 10^{-1} + 7 \times 10^{-2} + 2 \times 10^{-3} \end{aligned}$$

Représentation “scientifique” sur l’écran d’une calculatrice:

$$3,25172 \ 02 \equiv 3,25172 \times 10^2$$

→ pratique pour des très grands nombres ...

$$3,25172 \ 20 = 3251720000000000000000$$

2.1 Le système décimal et le système binaire

Dans le système décimal habituel, les nombres sont encodés avec les chiffres 0, 1, 2, ..., 9

Exemple:

$$\begin{aligned} 325,172 = & 3 \times 10^2 + 2 \times 10^1 + 5 \times 10^0 \\ & + 1 \times 10^{-1} + 7 \times 10^{-2} + 2 \times 10^{-3} \end{aligned}$$

Représentation “scientifique” sur l'écran d'une calculatrice:

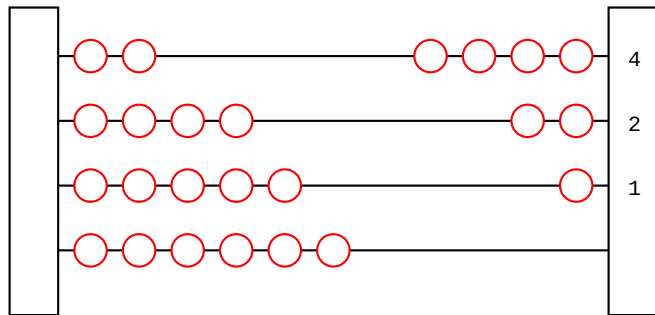
$$3,25172 \ 02 \equiv 3,25172 \times 10^2$$

→ pratique pour des très grands nombres ...
et des très petits nombres:

$$3,25172 \ -20 = 0,00000000000000000000325172$$

Le système décimal et l'abaque

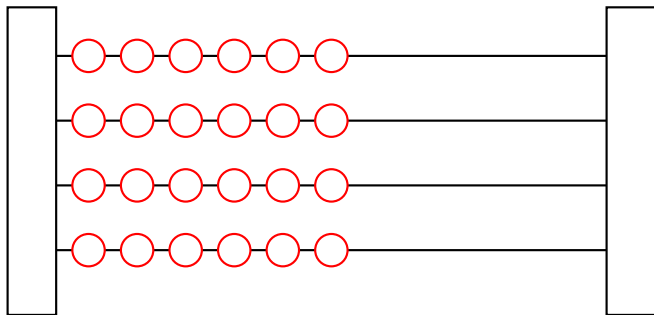
L'abaque à six boules par ligne ?



$$124 = 1 \times 6^2 + 2 \times 6^1 + 4 \times 6^0 = 52$$

Le système hexal

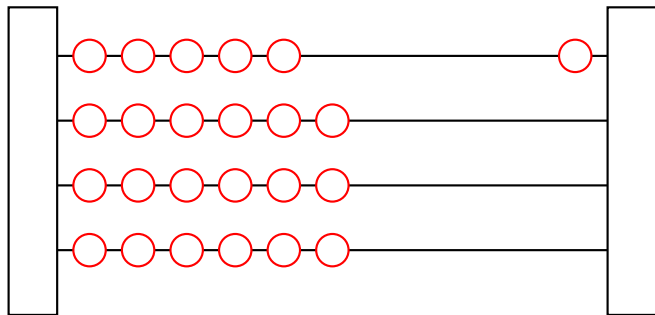
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



0

Le système hexal

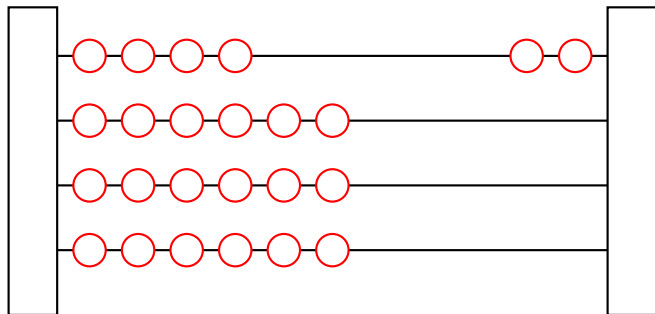
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



1

Le système hexal

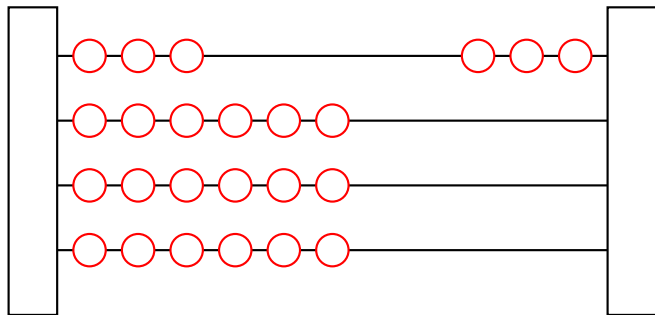
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



2

Le système hexal

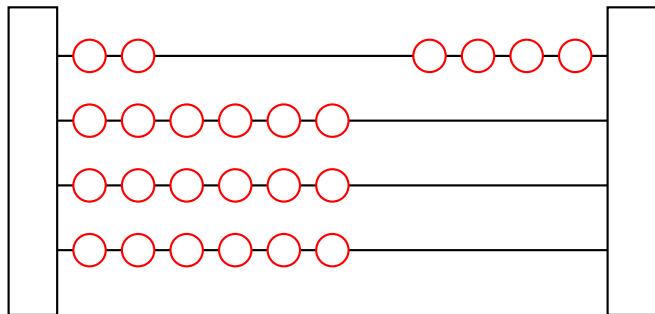
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



3

Le système hexal

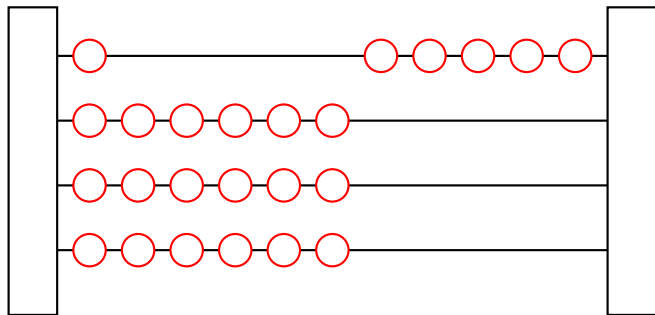
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



4

Le système hexal

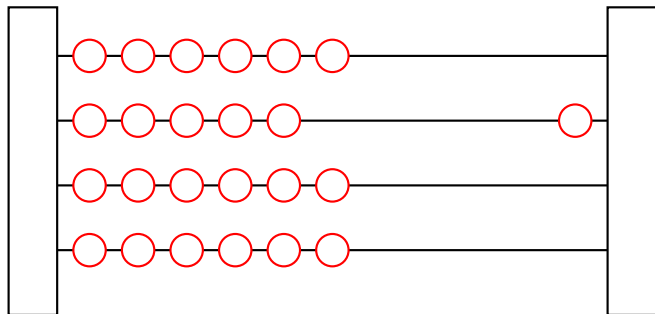
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



5

Le système hexal

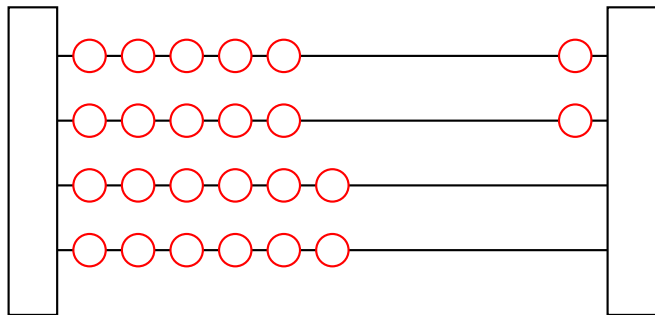
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



10 (6)

Le système hexal

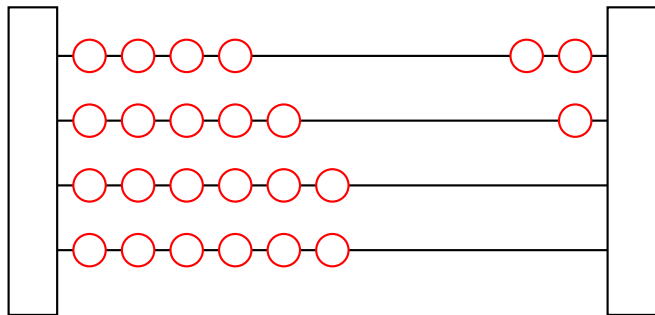
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



11 (7)

Le système hexal

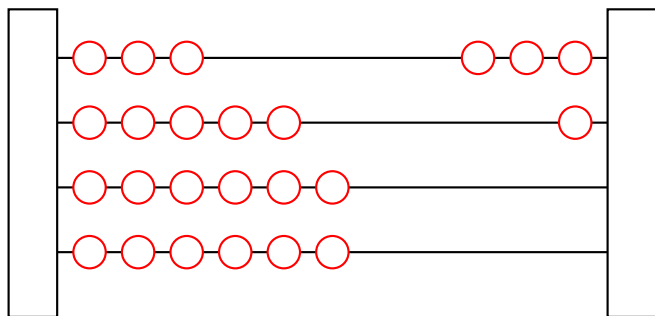
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



12 (8)

Le système hexal

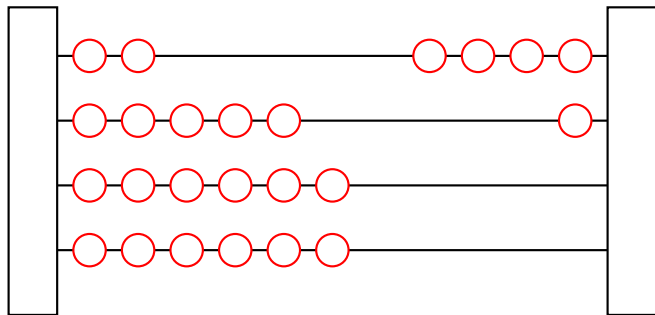
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



13 (9)

Le système hexal

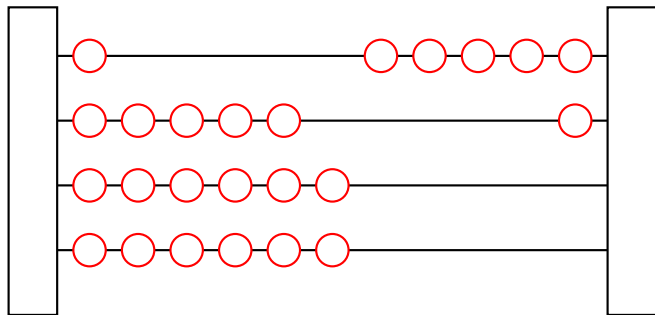
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



14 (10)

Le système hexal

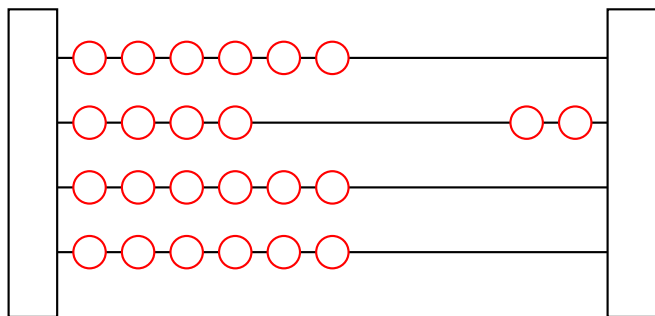
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



15 (11)

Le système hexal

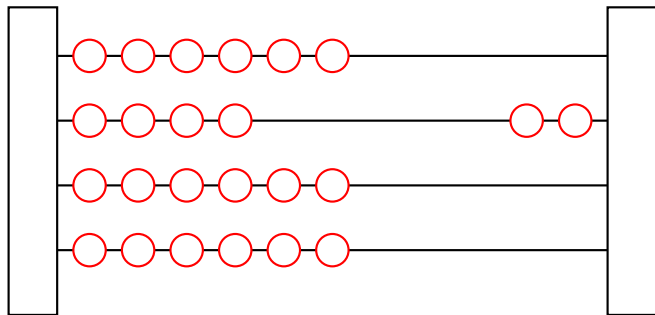
→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:



20 (12)

Le système hexal

→ encoder des nombres avec les chiffres 0, 1, 2, 3, 4, 5:

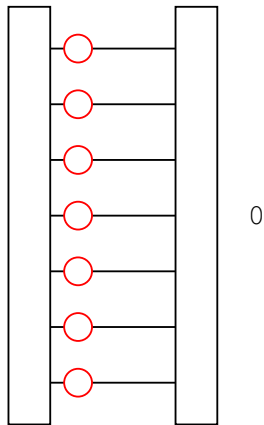


20 (12)

→ ca marcherait aussi bien que le système décimal

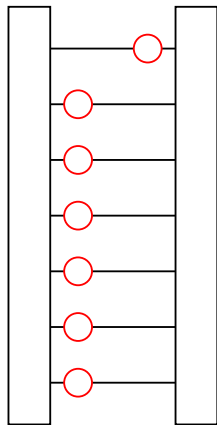
Le système binaire

→ encoder des nombres avec les chiffres 0 et 1



Le système binaire

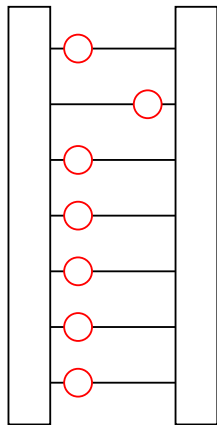
→ encoder des nombres avec les chiffres 0 et 1



1 (1)

Le système binaire

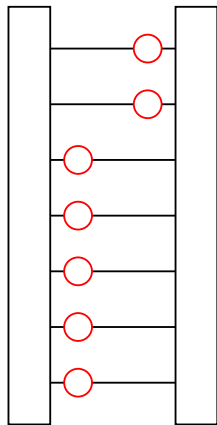
→ encoder des nombres avec les chiffres 0 et 1



1010101 (2)

Le système binaire

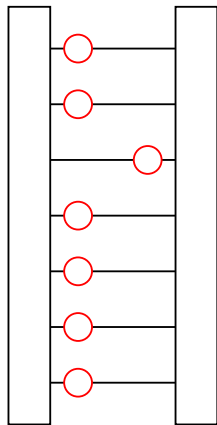
→ encoder des nombres avec les chiffres 0 et 1



11 (3)

Le système binaire

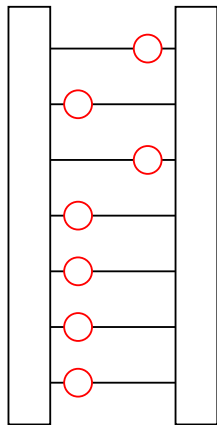
→ encoder des nombres avec les chiffres 0 et 1



100 (4)

Le système binaire

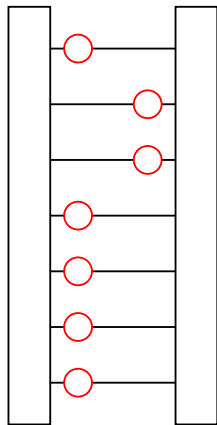
→ encoder des nombres avec les chiffres 0 et 1



101 (5)

Le système binaire

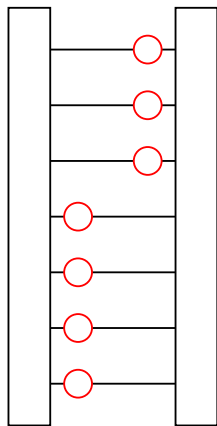
→ encoder des nombres avec les chiffres 0 et 1



110 (6)

Le système binaire

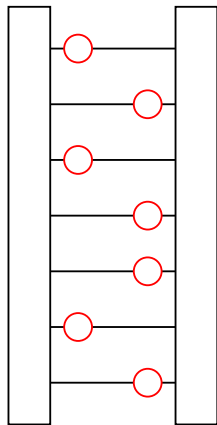
→ encoder des nombres avec les chiffres 0 et 1



111 (7)

Le système binaire

→ encoder des nombres avec les chiffres 0 et 1



$$1011010 = 2^6 + 2^4 + 2^3 + 2^1 = 90$$

Opérations arithmétiques dans le système binaire

Addition

$$\begin{array}{r} 1 \quad 0 \quad 1 \quad 1 \\ + \quad 0 \quad 1 \quad 1 \quad 0 \\ \hline \end{array}$$

$$\begin{array}{r} 11 \\ + \quad 6 \\ \hline 17 \end{array}$$

Opérations arithmétiques dans le système binaire

Addition

$$\begin{array}{rcccc} & 1 & 0 & 1 & 1 \\ + & 0 & 1 & 1 & 0 \\ \hline & & & & 1 \end{array}$$

$$\begin{array}{r} 11 \\ + 6 \\ \hline 17 \end{array}$$

Opérations arithmétiques dans le système binaire

Addition

$$\begin{array}{rcccc} & 1 & 0 & 1 & 1 \\ + & 0 & 1_1 & 1 & 0 \\ \hline & & & \cancel{2} & 1 \\ & & & 0 & \end{array}$$

$$\begin{array}{r} 11 \\ + 6 \\ \hline 17 \end{array}$$

Opérations arithmétiques dans le système binaire

Addition

$$\begin{array}{rcccc} & 1 & 0 & 1 & 1 \\ + & 0_1 & 1 & 1 & 0 \\ \hline & \cancel{2} & 0 & 1 & \\ & 0 & & & \end{array}$$

$$\begin{array}{r} 11 \\ + 6 \\ \hline 17 \end{array}$$

Opérations arithmétiques dans le système binaire

Addition

$$\begin{array}{r} 1011 \\ + 0110 \\ \hline 10001 \end{array}$$

$$\begin{array}{r} 11 \\ + 6 \\ \hline 17 \end{array}$$

Opérations arithmétiques dans le système binaire

Soustraction

$$\begin{array}{r} 1 \quad 0 \quad 1 \quad 1 \\ - \quad 0 \quad 1 \quad 1 \quad 0 \\ \hline 1 \end{array}$$

Opérations arithmétiques dans le système binaire

Soustraction

$$\begin{array}{r} 1 \quad 0 \quad 1 \quad 1 \\ - \quad 0 \quad 1 \quad 1 \quad 0 \\ \hline 0 \quad 1 \end{array}$$

Opérations arithmétiques dans le système binaire

Soustraction

$$\begin{array}{r} 1 \quad 0 \quad 1 \quad 1 \\ - \quad 0_1 \quad 1 \quad 1 \quad 0 \\ \hline \quad 1 \quad 0 \quad 1 \end{array}$$

Opérations arithmétiques dans le système binaire

Soustraction

$$\begin{array}{r} 1011 \\ - 0110 \\ \hline 101 \end{array}$$

$$\begin{array}{r} 11 \\ - 6 \\ \hline 5 \end{array}$$

Opérations arithmétiques dans le système binaire

Multiplication

$$1\ 0\ 1\ 1 \quad \times \quad 0\ 1\ 1\ 0 \quad =$$

Opérations arithmétiques dans le système binaire

Multiplication

$$\begin{array}{cccc} 1 & 0 & 1 & 1 \\ \times & 0 & 1 & 1 & 0 \\ \hline & & & & 0 & 0 & 0 & 0 \\ & & & & 1 & 0 & 1 & 1 \\ & & & 1 & 0 & 1 & 1 & \\ & & 0 & 0 & 0 & 0 & & \end{array}$$

Opérations arithmétiques dans le système binaire

Multiplication

[illegible]

Opérations arithmétiques dans le système binaire

Multiplication

[illegible]

Opérations arithmétiques dans le système binaire

Multiplication

$$\begin{array}{r} 1011 \times 0110 = \\ 0000 \\ 1011 \\ 1011 \\ 000101 \\ \hline 0010 \end{array}$$

Opérations arithmétiques dans le système binaire

Multiplication

$$\begin{array}{r} 1011 \times 0110 = \\ 0000 \\ 1011 \\ 1011 \\ 00_10_10_1 \\ \hline 00010 \end{array}$$

Opérations arithmétiques dans le système binaire

Multiplication

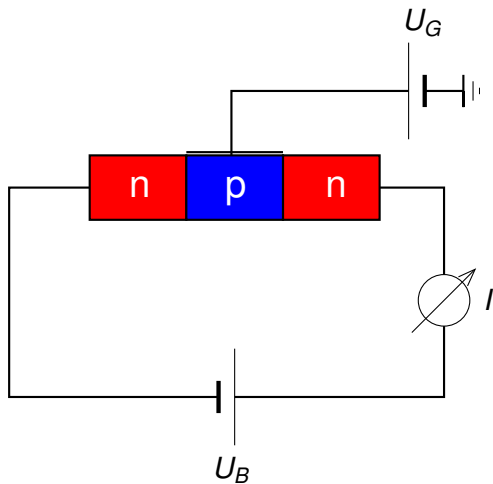
$$\begin{array}{r} 1011 \times 0110 = \\ 0000 \\ 1011 \\ 1011 \\ 0_10_10_10_1 \\ \hline 0000100 \end{array}$$

Pourquoi le système binaire ?

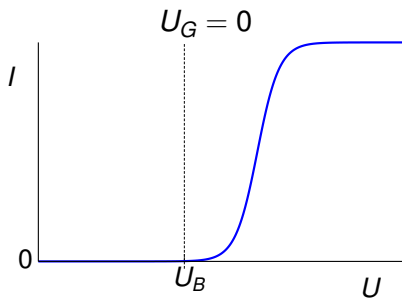
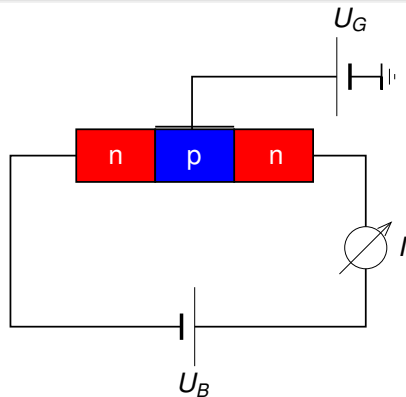
... et non pas un système ternaire, hexal, décimal ... ?

- plus facile à réaliser
- facile à encoder des informations plus générales
- plus robuste que des systèmes plus compliqués
- longue histoire :
télégraphie (code de Morse), cartes perforés, CDs ...

2.2 La réalisation du bit (binary digit)

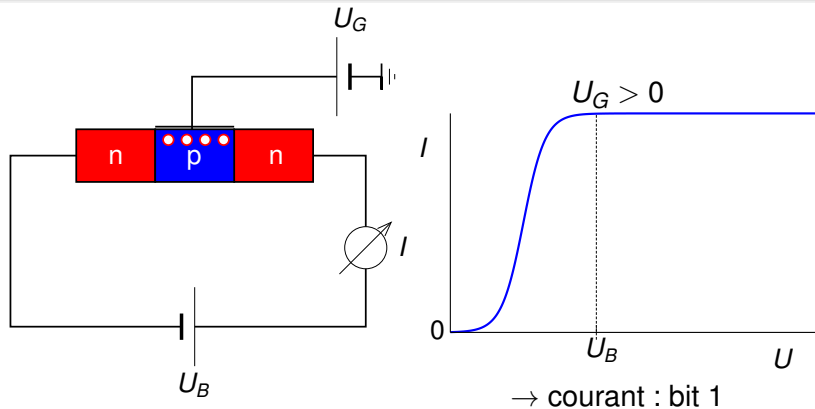


2.2 La réalisation du bit (binary digit)

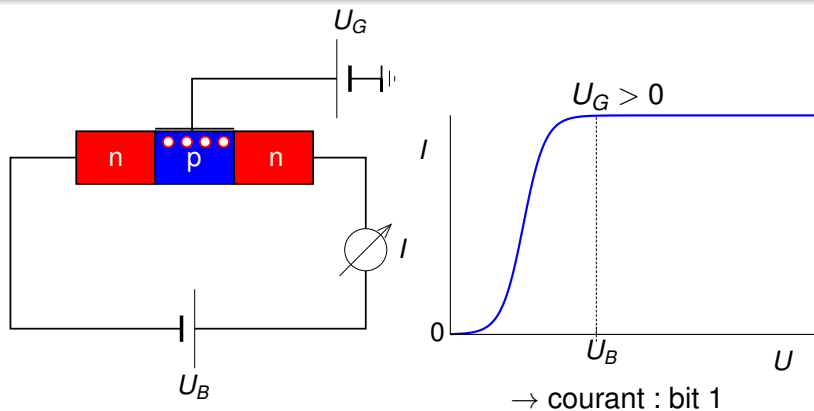


→ pas de courant : bit 0

2.2 La réalisation du bit (binary digit)



2.2 La réalisation du bit (binary digit)

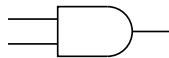


→ comment garder de l'information binaire sur un circuit ?

Les mémoires binaires sur les circuits intégrés

Rappel: les opérations logiques élémentaires

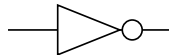
► AND : $A = B \wedge C$



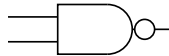
► OR : $A = B \vee C$



► NOT : $A = \overline{B}$



► NAND : $A = \overline{B \wedge C}$



► NOR : $A = \overline{B \vee C}$



► XOR : $A = (B \wedge \overline{C}) \vee (C \wedge \overline{B})$



► XNOR : $A = (B \wedge C) \vee (\overline{B} \wedge \overline{C})$



Les mémoires binaires sur des circuits intégrés

→ les circuits bistables (flip-flop)

On utilise deux éléments NOR



Les mémoires binaires sur des circuits intégrés

→ les circuits bistables (flip-flop)

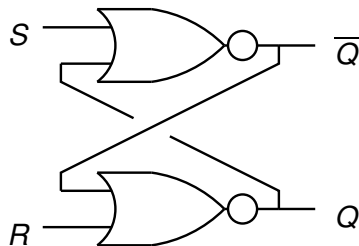
On utilise deux éléments NOR

input S : activation (set)

input R : désactivation (reset)

output Q : lire le contenu

output $\overline{Q}$: le contraire de Q

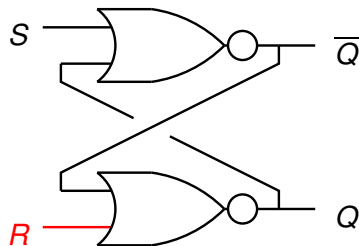


Les mémoires binaires sur des circuits intégrés

→ les circuits bistables (flip-flop)

On utilise deux éléments NOR

désactivation avec le contact R



Les mémoires binaires sur des circuits intégrés

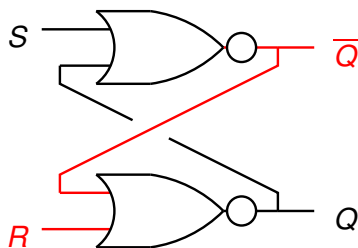
→ les circuits bistables (flip-flop)

On utilise deux éléments NOR

désactivation avec le contact R

→ Q "est faux"

→ $\overline{Q}$ "est vrai"



Les mémoires binaires sur des circuits intégrés

→ les circuits bistables (flip-flop)

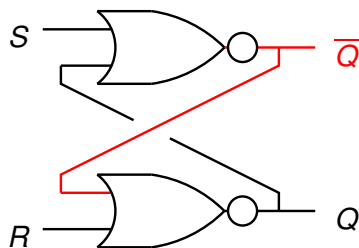
On utilise deux éléments NOR

désactivation avec le contact R

→ Q "est faux"

→ $\overline{Q}$ "est vrai"

→ ça reste comme ça
même si R est découpé

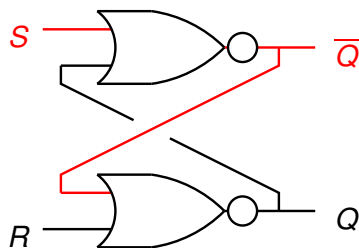


Les mémoires binaires sur des circuits intégrés

→ les circuits bistables (flip-flop)

On utilise deux éléments NOR

activation avec le contact S



Les mémoires binaires sur des circuits intégrés

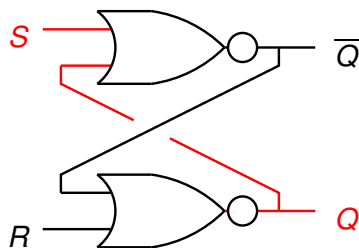
→ les circuits bistables (flip-flop)

On utilise deux éléments NOR

activation avec le contact S

→ Q "est vrai"

→ $\overline{Q}$ "est faux"



Les mémoires binaires sur des circuits intégrés

→ les circuits bistables (flip-flop)

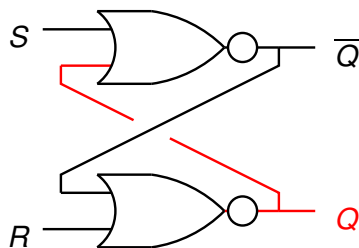
On utilise deux éléments NOR

activation avec le contact S

→ Q "est vrai"

→ $\overline{Q}$ "est faux"

→ ça reste comme ça
même si S est découpé

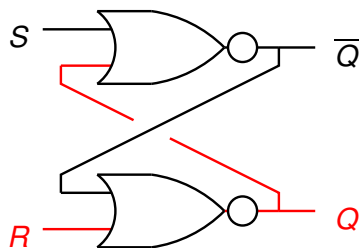


Les mémoires binaires sur des circuits intégrés

→ les circuits bistables (flip-flop)

On utilise deux éléments NOR

désactivation avec le contact R



Les mémoires binaires sur des circuits intégrés

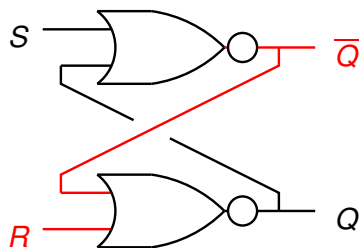
→ les circuits bistables (flip-flop)

On utilise deux éléments NOR

désactivation avec le contact R

→ Q "est faux"

→ $\overline{Q}$ "est vrai"



Les mémoires binaires sur des circuits intégrés

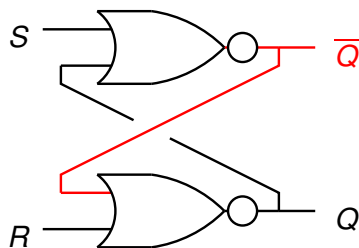
→ les circuits bistables (flip-flop)

On utilise deux éléments NOR

désactivation avec le contact R

→ Q "est faux"

→ $\overline{Q}$ "est vrai"

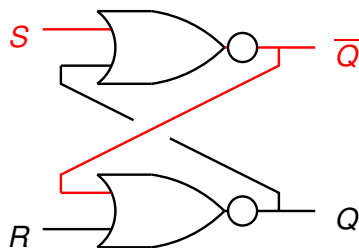


Les mémoires binaires sur des circuits intégrés

→ les circuits bistables (flip-flop)

On utilise deux éléments NOR

activation avec le contact S



Les mémoires binaires sur des circuits intégrés

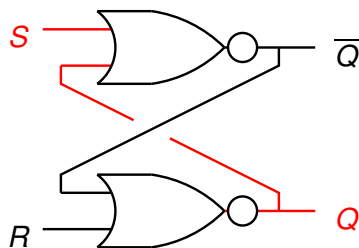
→ les circuits bistables (flip-flop)

On utilise deux éléments NOR

activation avec le contact S

→ Q "est vrai"

→ $\overline{Q}$ "est faux"



Les mémoires binaires sur des circuits intégrés

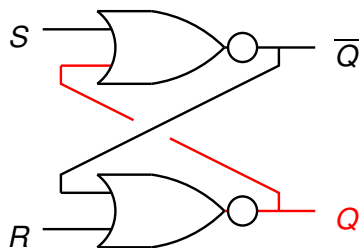
→ les circuits bistables (flip-flop)

On utilise deux éléments NOR

activation avec le contact S

→ Q "est vrai"

→ $\overline{Q}$ "est faux"



→ rapidité et stabilité

(tant que le circuit est connecté à la source de tension)

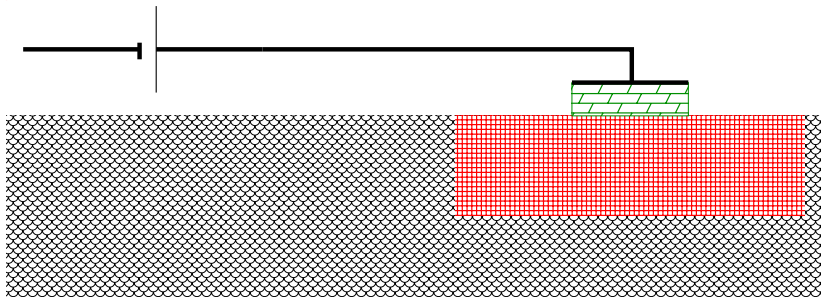
→ six transistors nécessaires pour maintenir un bit

→ SRAM (Static Random Access Memory)

Les mémoires binaires sur des circuits intégrés

→ les mémoires DRAM (Dynamic Random Access Memory)

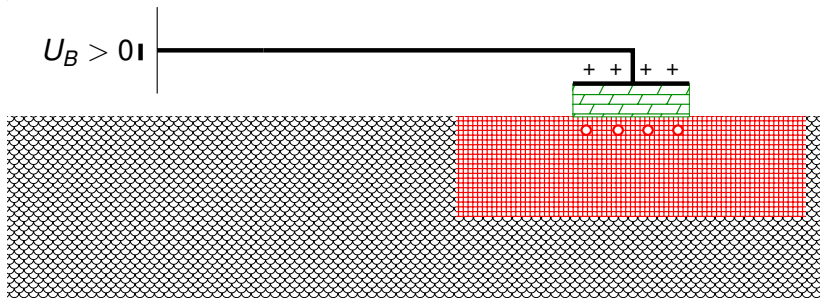
On utilise des **capacitances** pour sauvegarder des bits



Les mémoires binaires sur des circuits intégrés

→ les mémoires DRAM (Dynamic Random Access Memory)

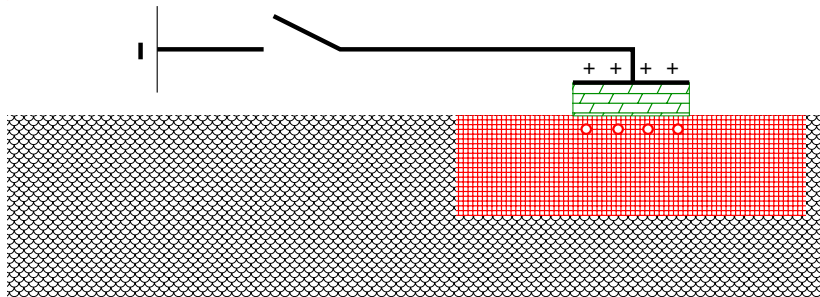
On utilise des **capacitances** pour sauvegarder des bits



Les mémoires binaires sur des circuits intégrés

→ les mémoires DRAM (Dynamic Random Access Memory)

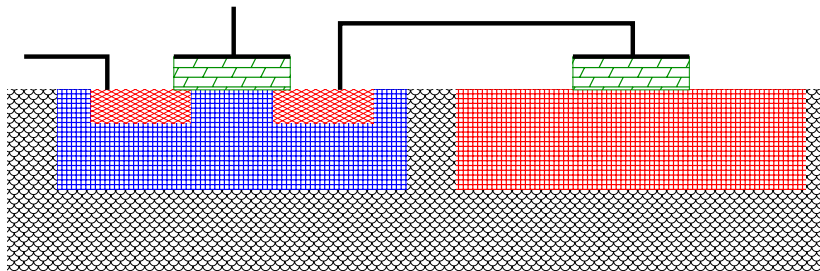
On utilise des **capacitances** pour sauvegarder des bits



Les mémoires binaires sur des circuits intégrés

→ les mémoires DRAM (Dynamic Random Access Memory)

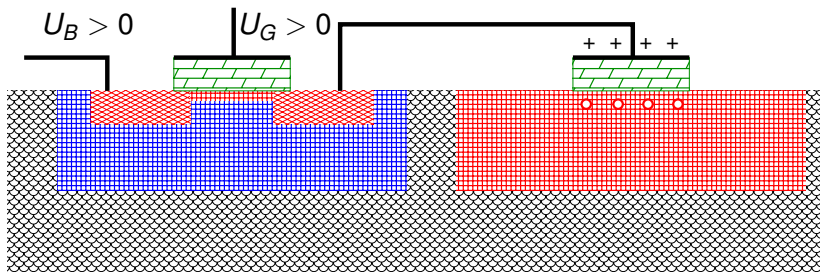
On utilise des **capacitances** pour sauvegarder des bits



Les mémoires binaires sur des circuits intégrés

→ les mémoires DRAM (Dynamic Random Access Memory)

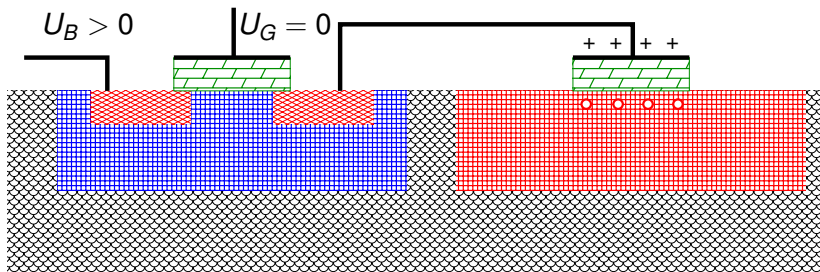
On utilise des **capacitances** pour sauvegarder des bits



Les mémoires binaires sur des circuits intégrés

→ les mémoires DRAM (Dynamic Random Access Memory)

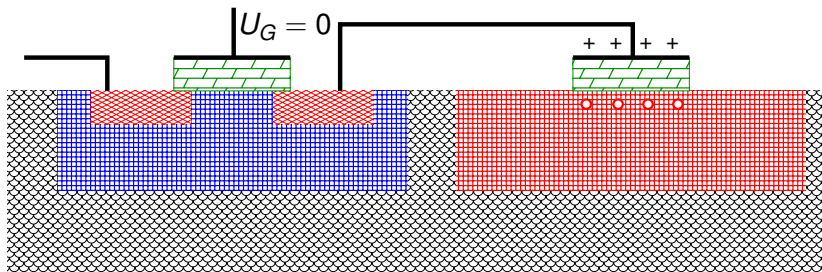
On utilise des **capacitances** pour sauvegarder des bits



Les mémoires binaires sur des circuits intégrés

→ les mémoires DRAM (Dynamic Random Access Memory)

On utilise des **capacitances** pour sauvegarder des bits



→ deux transistors nécessaires pour maintenir un bit

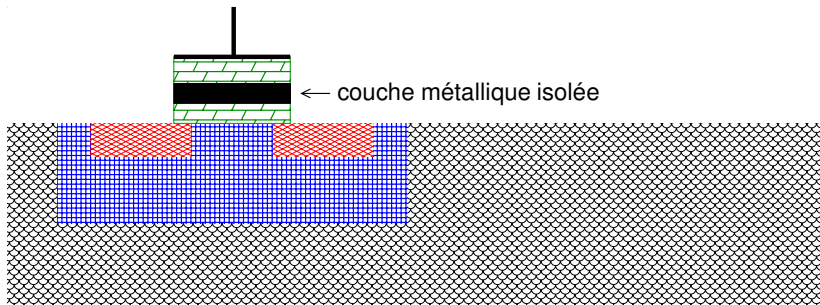
→ pas de stabilité, à cause de courants de fuite

→ circuits de rafraîchissement nécessaires

Les mémoires binaires sur des circuits intégrés

→ les mémoires Flash (Toshiba, 1980)

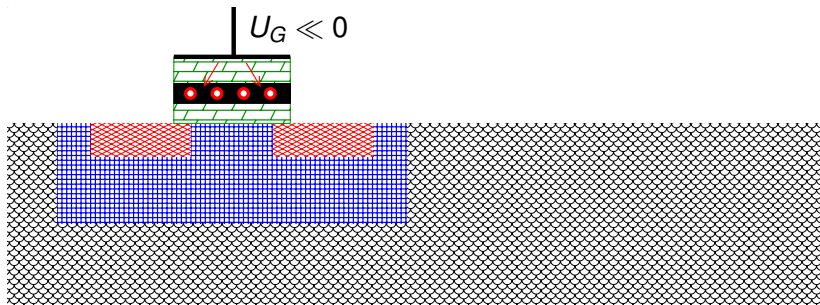
On utilise des capacitances **isolées** pour sauvegarder des bits



Les mémoires binaires sur des circuits intégrés

→ les mémoires Flash (Toshiba, 1980)

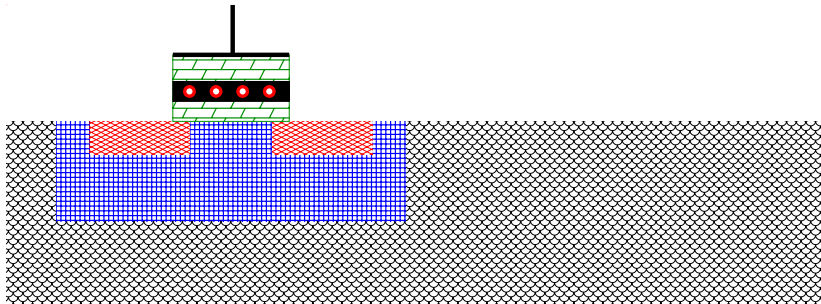
On utilise des capacitances **isolées** pour sauvegarder des bits



Les mémoires binaires sur des circuits intégrés

→ les mémoires Flash (Toshiba, 1980)

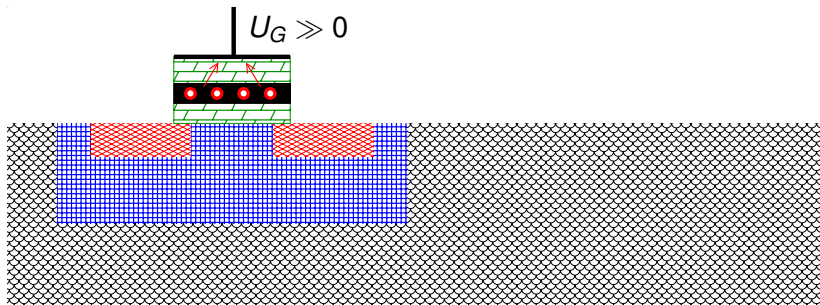
On utilise des capacitances **isolées** pour sauvegarder des bits



Les mémoires binaires sur des circuits intégrés

→ les mémoires Flash (Toshiba, 1980)

On utilise des capacitances **isolées** pour sauvegarder des bits

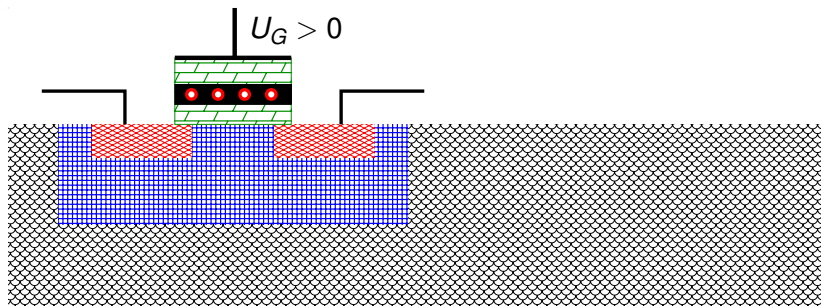


→ charger et décharger avec des tensions U_G très fortes (effet tunnel quantique)

Les mémoires binaires sur des circuits intégrés

→ les mémoires Flash (Toshiba, 1980)

On utilise des capacitances **isolées** pour sauvegarder des bits

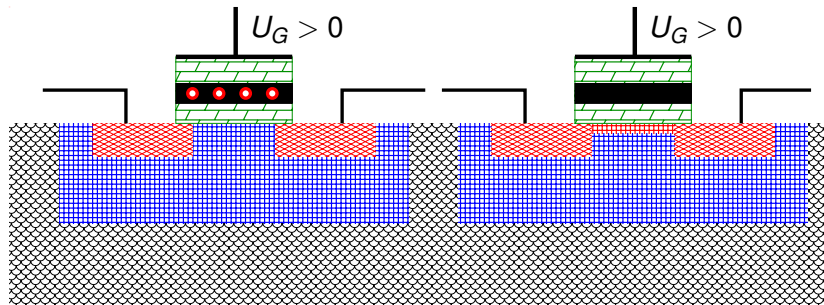


→ pas de courant au travers du transistor
si la capacitance est chargée (effet de screening)

Les mémoires binaires sur des circuits intégrés

→ les mémoires Flash (Toshiba, 1980)

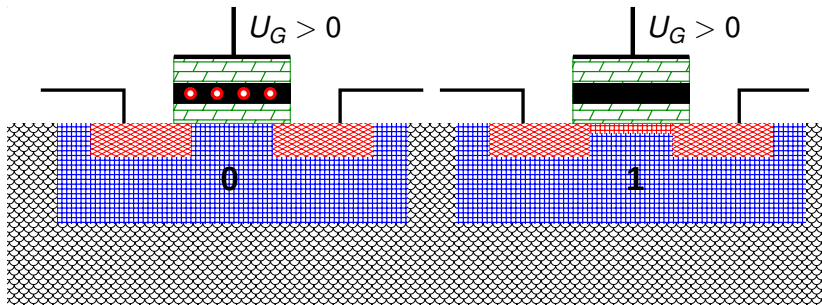
On utilise des capacitances **isolées** pour sauvegarder des bits



Les mémoires binaires sur des circuits intégrés

→ les mémoires Flash (Toshiba, 1980)

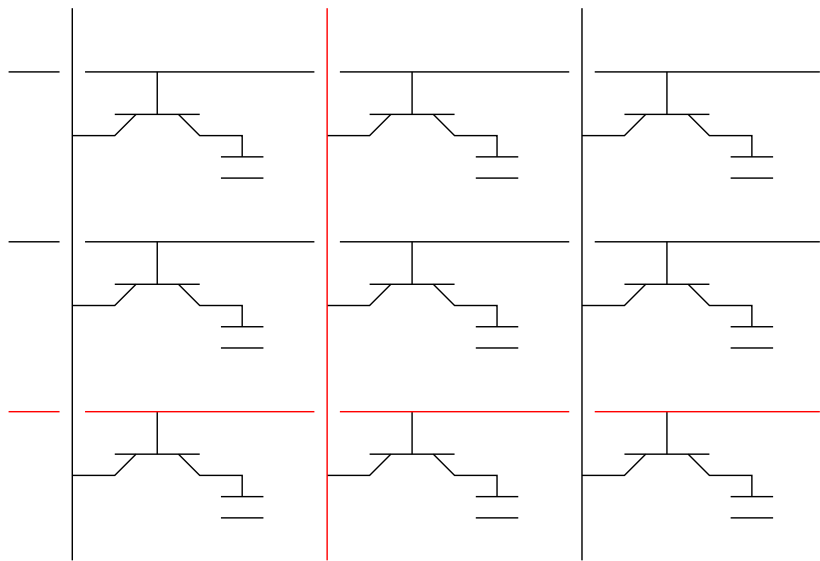
On utilise des capacitances **isolées** pour sauvegarder des bits



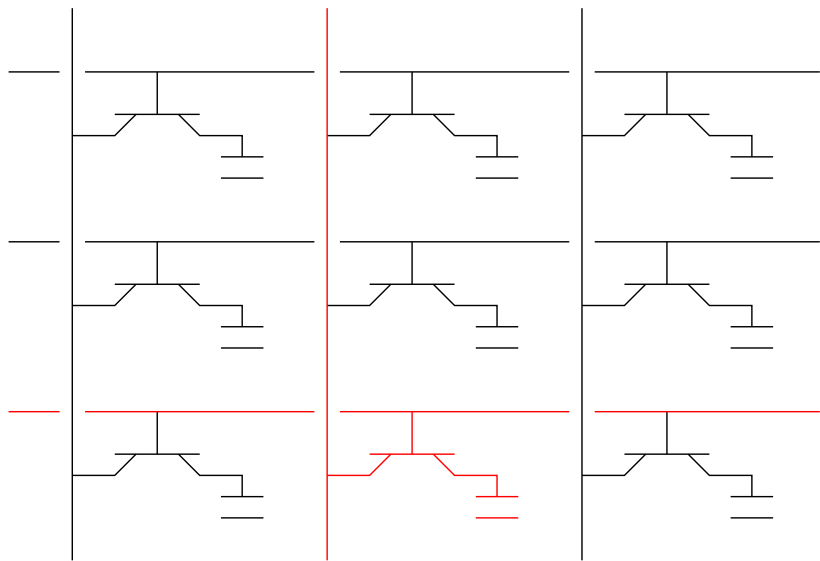
→ très longue durée de vie de la mémoire
même si le circuit est découpé de la source de tension

→ utilisé dans les GSM, les clés USB, les caméras ...

Les mémoires binaires sur des circuits intégrés



Les mémoires binaires sur des circuits intégrés



2.3 L'octet (byte)

= la combinaison de 8 bits

0 1 0 1 1 0 1 1

→ l'unité de base pour des mémoires informatiques:

1 ko (kB) = 1 kilooctet = 10^3 octets = 1000 octets

1 Mo (MB) = 1 mégaoctet = 10^6 octets = 1 000 000 octets

1 Go (GB) = 1 gigaoctet = 10^9 octets = 1 000 000 000 octets

1 To (TB) = 1 téraoctet = 10^{12} octets = 1 000 000 000 000 octets

2.3 L'octet (byte)

= la combinaison de 8 bits

0 0 0 0 0 0 0 0

0

2.3 L'octet (byte)

= la combinaison de 8 bits

0 0 0 0 0 0 0 1

1

2.3 L'octet (byte)

= la combinaison de 8 bits

0 0 0 0 0 0 1 0

2

2.3 L'octet (byte)

= la combinaison de 8 bits

0 0 0 0 0 0 1 1

3

2.3 L'octet (byte)

= la combinaison de 8 bits

0 0 0 0 0 1 0 0

4

2.3 L'octet (byte)

= la combinaison de 8 bits

0 0 0 0 1 0 0 0

8

2.3 L'octet (byte)

= la combinaison de 8 bits

0 0 0 1 0 0 0 0

16

2.3 L'octet (byte)

= la combinaison de 8 bits

0 0 1 0 0 0 0 0

32

2.3 L'octet (byte)

= la combinaison de 8 bits

0 1 0 0 0 0 0 0

64

2.3 L'octet (byte)

= la combinaison de 8 bits

1 0 0 0 0 0 0 0

128

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 1 1 1 1 1

→ codage des nombres entiers entre 0 et $2^8 - 1 = 255$

2.3 L'octet (byte)

= la combinaison de 8 bits

0 1 1 1 1 1 1 1

→ codage des nombres entiers entre 0 et $2^8 - 1 = 255$

... ou des nombres entiers, positifs et négatifs, entre

$-2^7 = -128$ et $2^7 - 1 = \mathbf{127}$

2.3 L'octet (byte)

= la combinaison de 8 bits

1 0 0 0 0 0 0 0

→ codage des nombres entiers entre 0 et $2^8 - 1 = 255$

... ou des nombres entiers, positifs et négatifs, entre

$-2^7 = -\mathbf{128}$ et $2^7 - 1 = 127$

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 1 1 1 1 1

−1

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 1 1 1 1 0

-2

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 1 1 1 0 1

−3

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 1 1 1 0 0

−4

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 1 1 0 1 1

−5

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 1 1 0 1 0

—6

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 1 1 0 0 1

-7

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 1 1 0 0 0

—8

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 1 0 0 0 0

—16

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 1 0 0 0 0 0

—32

2.3 L'octet (byte)

= la combinaison de 8 bits

1 1 0 0 0 0 0 0

—64

2.3 L'octet (byte)

= la combinaison de 8 bits

1 0 0 0 0 0 0 0

–128

2.3 L'octet (byte)

= la combinaison de 8 bits

1 0 1 1 0 1 0 1

Détermination des nombres négatifs:

→ négation de tous les bits

2.3 L'octet (byte)

= la combinaison de 8 bits

0 1 0 0 1 0 1 0

Détermination des nombres négatifs:

→ négation de tous les bits

→ addition de 1

2.3 L'octet (byte)

= la combinaison de 8 bits

0 1 0 0 1 0 1 1

Détermination des nombres négatifs:

→ négation de tous les bits

→ addition de 1

75

2.3 L'octet (byte)

= la combinaison de 8 bits

1 0 1 1 0 1 0 1

Détermination des nombres négatifs:

→ négation de tous les bits

→ addition de 1

−75

Addition des octets

$$\begin{array}{r} 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1 \\ +\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0 \\ \hline 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1 \end{array}$$

$$\begin{array}{r} 91 \\ +\ -106 \\ \hline -15 \end{array}$$

Addition des octets

$$\begin{array}{r} 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0 \\ \hline 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1 \end{array}$$

$$\begin{array}{r} 91 \\ +\ 86 \\ \hline -79\ \text{?!?} \end{array}$$

... c'est exact à l'exception d' **erreurs d'overflow**

Addition des octets

$$\begin{array}{r} 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1 \\ +\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0 \\ \hline 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1 \end{array}$$

$$\begin{array}{r} 91 \\ +\ -42 \\ \hline 49\ \text{o.k.} \end{array}$$

Addition des octets

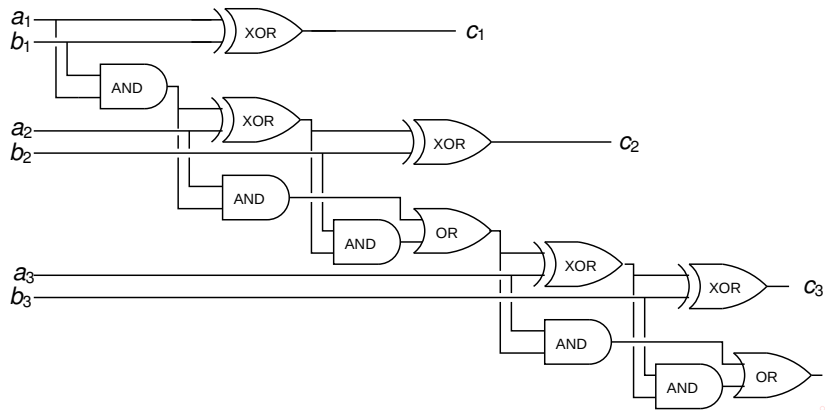
Implémentation sur un circuit intégré:

$$\begin{array}{rcccccccc} & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 \\ + & b_8 & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 \\ \hline c_8 & c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & \end{array}$$

- opération XOR pour l'addition de a_1 et b_1 , a_2 et b_2 , ...
- opération AND pour ajouter 1 à la colonne suivante si $a_1 = 1$ et $b_1 = 1$ (pareil pour a_2 et b_2 , ...)
- opération OR pour faire la même chose dans les colonnes suivantes

Addition des octets

	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1
+	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1
	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1



Soustraction des octets

→ circuit similaire à celui pour les additions ...

ou

→ multiplication du deuxième octet par -1

→ négation de tous les bits (opération NOT)

→ addition de 1

→ addition ordinaire des deux octets

Représentation des nombres entiers plus grands

→ combinaison de **plusieurs octets**

1 0 0 1 1 0 1 0 0 1 0 0 1 1 1 1

2 octets: représentation des nombres entre -2^{15} et $2^{15} - 1$
c-à-d entre $-32\,768$ et $32\,767$

4 octets: représentation des nombres entre -2^{31} et $2^{31} - 1$
c-à-d entre $-2\,147\,483\,648$ et $2\,147\,483\,647$

Représentation des nombres réels

Représentation “scientifique” du nombre 325 dans le système décimal :

3,25 02

Représentation des nombres réels

Représentation “scientifique” du nombre 325 dans le système binaire :

1,01000101	1000
mantisse	exposant (puissance de 2)

→ nombres “flottants” (floating point numbers)

Représentation des nombres réels

Représentation “scientifique” du nombre 325 dans le système binaire :

1,01000101	1000
mantisse	exposant (puissance de 2)

→ nombres “flottants” (floating point numbers)

Exemple:

3 octets pour la mantisse et 1 octet pour l'exposant
(type “float” en C/C++)

→ représentation des nombres réels entre

$$\pm 1,2 \times 10^{-38} \text{ et } \pm 3,4 \times 10^{38}$$

avec 7 chiffres significatifs

Représentation des nombres réels

Représentation “scientifique” du nombre 325 dans le système binaire :

1,01000101	1000
mantisse	exposant (puissance de 2)

→ nombres “flottants” (floating point numbers)

Exemple:

6,5 octets pour la mantisse et 1,5 octets pour l'exposant
(type “double” en C/C++)

→ représentation des nombres réels entre

$$\pm 2,2 \times 10^{-308} \text{ et } \pm 1,8 \times 10^{308}$$

avec 14 chiffres significatifs

Addition des nombres réels

Illustration dans le système décimal:

$$1435,12 + 0,119421 = 1435,239421$$

Calcul “numérique” avec 6 chiffres significatifs:

$$1,43512 \times 10^3 + 1,19421 \times 10^{-1} = 1,43523 \times 10^3 = 1435,23$$

→ attention: les derniers chiffres sont coupés !

→ **erreur numérique**

peut-être pas si grave ?

Addition des nombres réels

Illustration dans le système décimal:

$$1435,12 + 0,119421 = 1435,239421$$

Calcul “numérique” avec 6 chiffres significatifs:

$$1,43512 \times 10^3 + 1,19421 \times 10^{-1} = 1,43523 \times 10^3 = 1435,23$$

→ attention: les derniers chiffres sont coupés !

→ **erreur numérique**

$$[(1 + 10^{-3}) - 1] \times 10^3 \Rightarrow 1$$

Addition des nombres réels

Illustration dans le système décimal:

$$1435,12 + 0,119421 = 1435,239421$$

Calcul “numérique” avec 6 chiffres significatifs:

$$1,43512 \times 10^3 + 1,19421 \times 10^{-1} = 1,43523 \times 10^3 = 1435,23$$

→ attention: les derniers chiffres sont coupés !

→ **erreur numérique**

$$[(1 + 10^{-5}) - 1] \times 10^5 \Rightarrow 1$$

Addition des nombres réels

Illustration dans le système décimal:

$$1435,12 + 0,119421 = 1435,239421$$

Calcul “numérique” avec 6 chiffres significatifs:

$$1,43512 \times 10^3 + 1,19421 \times 10^{-1} = 1,43523 \times 10^3 = 1435,23$$

→ attention: les derniers chiffres sont coupés !

→ **erreur numérique**

$$[(1 + 10^{-7}) - 1] \times 10^7 \Rightarrow 0$$

→ **attention avec des différences !**