

## Plan

- Circuits logiques
- Notions de temps et de mémorisation
- Représentation des nombres
- Unité Arithmétique et Logique
- Contrôle et jonction des composants
- Evolution des ordinateurs – Historique
- Un microprocesseur simple
- Programmation d'un microprocesseur
- Système complet
- Les microprocesseurs actuels
- Exploitation de la performance des microprocesseurs
- Evolutions prochaines des microprocesseurs
- Utilisation des processeurs/ordinateurs
- Evolutions possibles pour les processeurs/ordinateurs à long terme

---

---

---

---

---

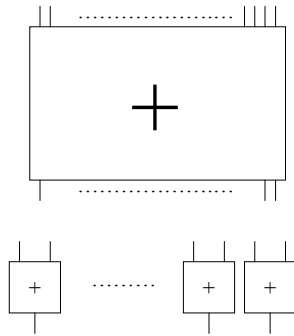
---

---

---

## Réalisation d'un Additionneur

- Conception d'un circuit: description complète avec table de vérité.
- Compromis coût/rapidité:
  - Additionneur n bits: table de  $2n$  variables,  $2^{2n}$  entrées → rapide (en théorie), mais coûteux.
  - n additionneurs 1-bit: peu coûteux mais lent.
  - Alternative: n additionneurs 1-bit et ajout de composants pour accélérer le calcul.




---

---

---

---

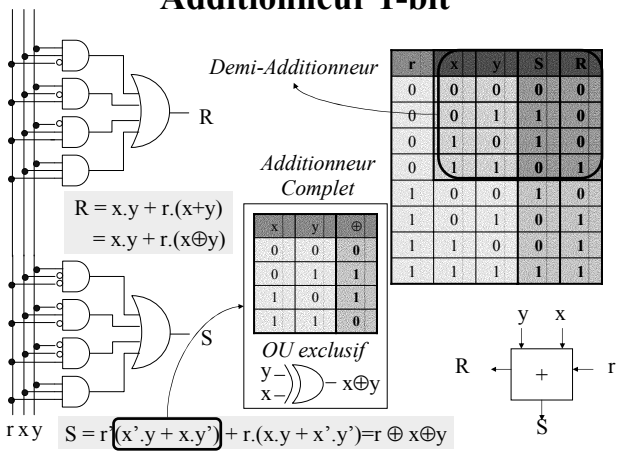
---

---

---

---

## Additionneur 1-bit




---

---

---

---

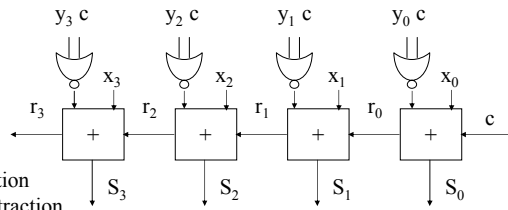
---

---

---

---

## Additionneur/Soustracteur n-bits



$c=0$ : addition  
 $c=1$ : soustraction

- Soustracteur:
  - $X - Y = X + (-Y) = X + Y' + 1$
  - $c = 1 \rightarrow$  pas d'additionneur supplémentaire
- Facteur limitatif: propagation de la retenue

|   |       |       |       |       |       |       |
|---|-------|-------|-------|-------|-------|-------|
|   | $r_3$ | $r_2$ | $r_1$ | $r_0$ |       |       |
| X |       | $x_3$ | $x_2$ | $x_1$ | $x_0$ |       |
| Y | +     | $y_3$ | $y_2$ | $y_1$ | $y_0$ |       |
| Z |       | $r_3$ | $s_3$ | $s_2$ | $s_1$ | $s_0$ |

---

---

---

---

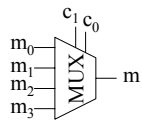
---

---

---

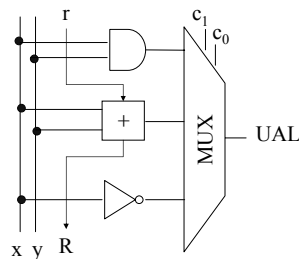
---

## Une UAL 1-bit Simple



Multiplexeur:  
 choix entre plusieurs entrées

|       |       |       |
|-------|-------|-------|
| $c_1$ | $c_0$ | $m$   |
| 0     | 0     | $m_0$ |
| 0     | 1     | $m_1$ |
| 1     | 0     | $m_2$ |
| 1     | 1     | $m_3$ |



|       |       |     |
|-------|-------|-----|
| $c_1$ | $c_0$ | UAL |
| 0     | 0     | ADD |
| 0     | 1     | ET  |
| 1     | 0     | NON |
| 1     | 1     | $d$ |

- Opérations élémentaires:
  - ADD, ET, NON
- UAL: Unité Arithmétique et Logique

---

---

---

---

---

---

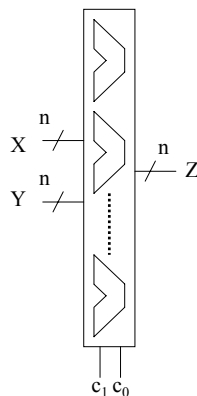
---

---

## UAL n -bits

- n ALUs 1-bit.
- Ebauche d'un jeu d'instructions:

|       |       |     |
|-------|-------|-----|
| $c_1$ | $c_0$ | UAL |
| 0     | 0     | ADD |
| 0     | 1     | ET  |
| 1     | 0     | NON |
| 1     | 1     | $d$ |




---

---

---

---

---

---

---

---

## Dépassement de Capacité

|    |   |   |   |   |   |
|----|---|---|---|---|---|
|    | 1 | 0 | 0 |   |   |
| 6  | 0 | 1 | 1 | 0 |   |
| 5  | + | 0 | 1 | 0 | 1 |
| -5 | 1 | 0 | 1 | 1 |   |

|    |   |   |   |   |   |
|----|---|---|---|---|---|
|    | 1 | 0 | 0 | 0 |   |
| -4 | 1 | 1 | 0 | 0 |   |
| -7 | + | 1 | 0 | 0 | 1 |
| 5  | 1 | 0 | 1 | 0 | 1 |

- Détecter un dépassement de capacité en complément à 2 ?

---

---

---

---

---

---

---

---

---

---

## Dépassement de Capacité

- Signal de dépassement de capacité  $S_{overflow}$  (1=overflow).

- Dépassement en complément à 2:

-  $x \leq 0, y \geq 0$  → pas de dépassement possible.

-  $x \geq 0, y \geq 0$ :

- Dépassement:  $Z=X+Y$  (complément à 2)

$Z > 2^{n-1}-1$ , et  $0 \leq X \leq 2^{n-1}-1, 0 \leq Y \leq 2^{n-1}-1$

→  $2^{n-1}-1 < Z \leq 2^n-2$

→ En complément à 2, nombres négatifs codés dans  $[2^{n-1}; 2^n-1]$

→ Z apparaît comme un nombre négatif.

-  $x \leq 0, y \leq 0$ : idem, en cas de dépassement, Z apparaît comme un nombre positif.

→ Critère de dépassement de capacité.

| $Z_{n-1}$ | $X_{n-1}$ | $Y_{n-1}$ | $S_{overflow}$ |
|-----------|-----------|-----------|----------------|
| 0         | 0         | 0         | 0              |
| 0         | 0         | 1         | 0              |
| 0         | 1         | 0         | 0              |
| 0         | 1         | 1         | 1              |
| 1         | 0         | 0         | 1              |
| 1         | 0         | 1         | 0              |
| 1         | 1         | 0         | 0              |
| 1         | 1         | 1         | 0              |

$$S_{overflow} = x_{n-1} \cdot y_{n-1} \cdot z'_{n-1} + x'_{n-1} \cdot y'_{n-1} \cdot z_{n-1}$$

---

---

---

---

---

---

---

---

---

---

## Dépassement de Capacité

- Que faire en cas de dépassement de capacité ?

- Plusieurs solutions:

- Provoquer une interruption du programme (TRAP)

- Exemple: MIPS R3000; deux instructions d'addition dont l'une peut provoquer une interruption.

- Mettre un bit de signalisation à 1 comme  $S_{overflow}$

- Exemple: Intel x86; SPARC: deux instructions, l'une met le bit à 1, l'autre pas.

- Ne rien faire

- Exemple: en C sur un SPARC 32-bits

-2147483648 ( $-2^{31}$ ) → 2147483647 ( $2^{31}-1$ )

```
int a=2000000000;
int b=2000000000;
int c;
c=a+b;
printf("%d\n",c);
```



**-294967296**

```
01110111001101011001010000000000
+01110111001101011001010000000000
-----
11110111001101011001010000000000
```

---

---

---

---

---

---

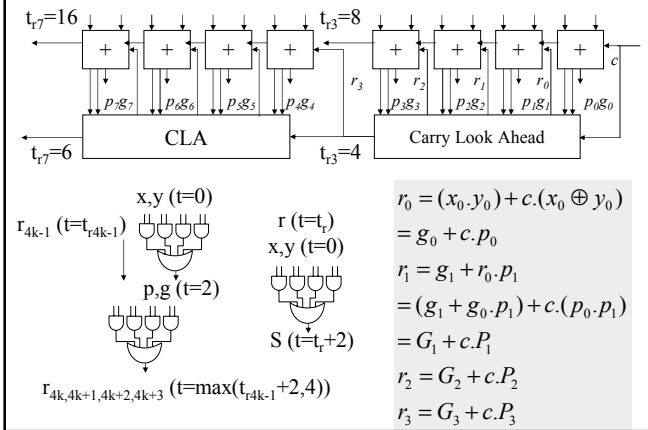
---

---

---

---

## Accélérer la Propagation de la Retenue




---

---

---

---

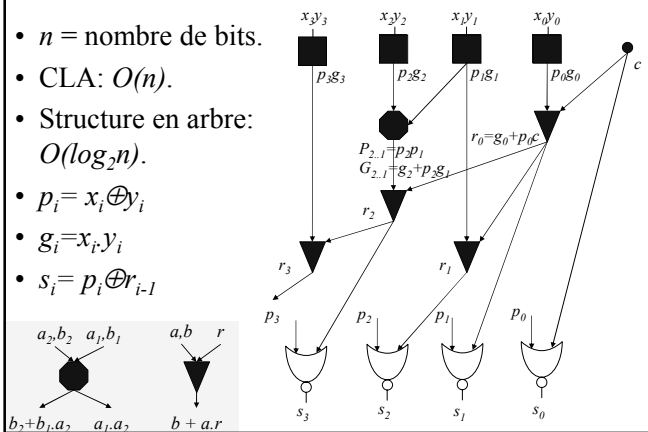
---

---

---

---

## Accélérer l'Addition




---

---

---

---

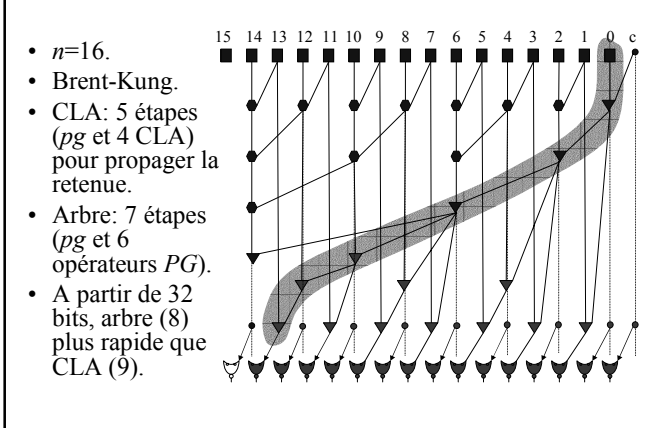
---

---

---

---

## Accélérer l'Addition




---

---

---

---

---

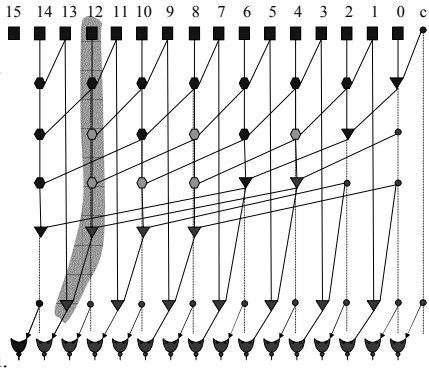
---

---

---

## Accélérer l'Addition

- $n=16$
- Han-Carlson.
- Compromis taille / performance
- Itanium:
  - 64 bits,
  - $0,18\mu\text{m}$ ,
  - 482 ps pour une addition.



---

---

---

---

---

---

---

---