

ARCHI – Architecture des ordinateurs

*Sylvain Brandel*

2022 – 2023

[sylvain.brandel@univ-lyon1.fr](mailto:sylvain.brandel@univ-lyon1.fr)



CM 6

# CIRCUITS COMBINATOIRES

## *PARTIE 2 – LOGIQUE EN TRANCHE*

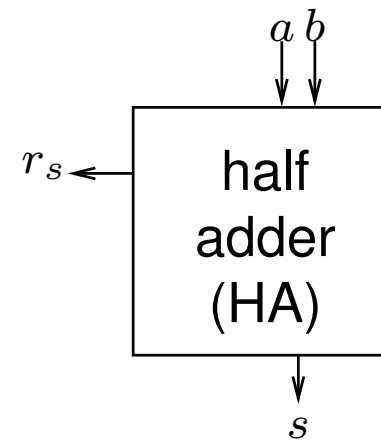
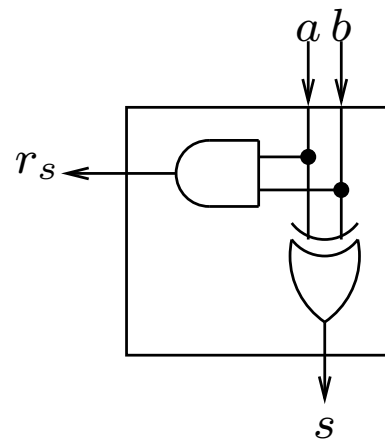
# Circuit combinatoire – Logique en tranche

- Opérateurs  $n$  bits à partir d'opérateurs 1 bit et règles d'assemblage
- Typiquement opérateurs arithmétiques :
  - Traitement par bits ou blocs (tranches) de bits
  - Propagation des retenues
- Ex : Additionneurs
  - Demi-additionneur *Half-adder*
  - Additionneur complet 1 bit *Full-adder*
  - Additionneur complet  $n$  bits
    - Propagation **simple** de retenue
    - Propagation **rapide** de retenue
      - **Sélection** de retenue
      - **Anticipation** de retenue

# Circuit combinatoire – Logique en tranche

## *Demi-additionneur*

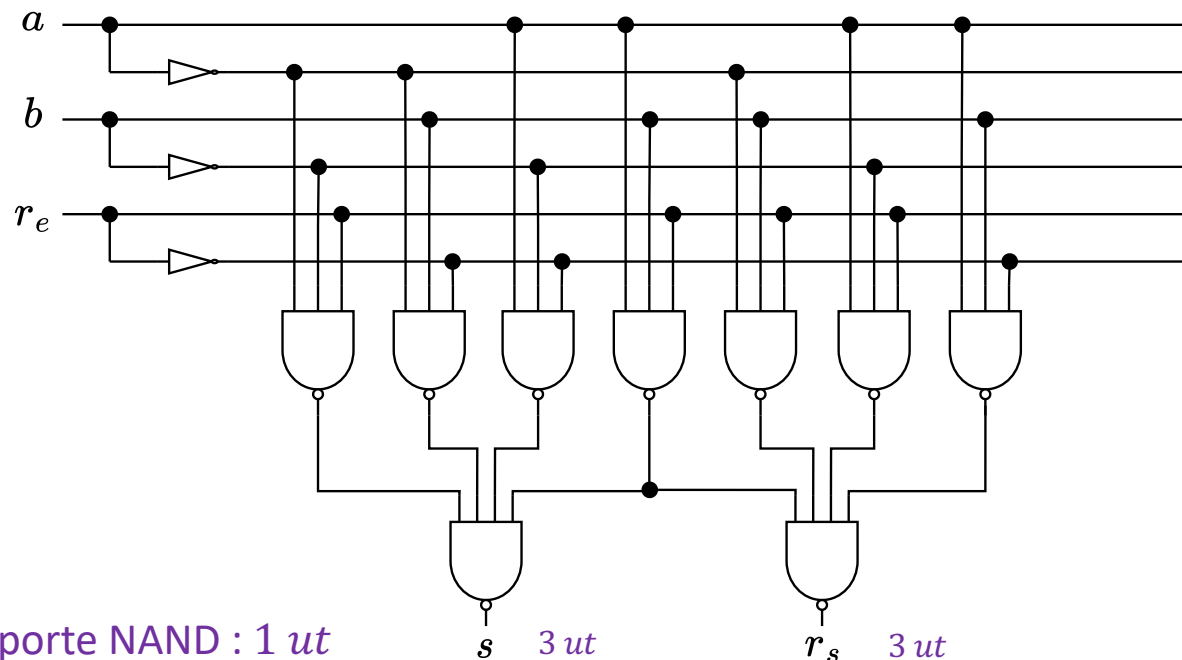
- Addition simple de deux bits
  - Entrée :  $a$  et  $b$  (opérandes)
  - Sortie :  $s$  (somme) et  $r_s$  (retenue de sortie)
- D'après la table de vérité
  - $s = \bar{a}b + a\bar{b}$
  - $r_s = ab$
- Après simplification
  - $s = a \oplus b$
  - $r_s = ab$



# Circuit combinatoire – Logique en tranche

## Additionneur complet

- Addition de deux bits avec une retenue entrante
  - Entrée :  $a$ ,  $b$  (opérandes) et  $r_e$  (retenue d'entrée)
  - Sortie :  $s$  (somme) et  $r_s$  (retenue de sortie)
- D'après la table de vérité
  - $s = \bar{a}\bar{b}r_e + \bar{a}b\bar{r}_e + a\bar{b}\bar{r}_e + abr_e$
  - $r_s = \bar{a}br_e + a\bar{b}r_e + ab\bar{r}_e + abr_e$



Temps de passage porte NAND : 1 ut

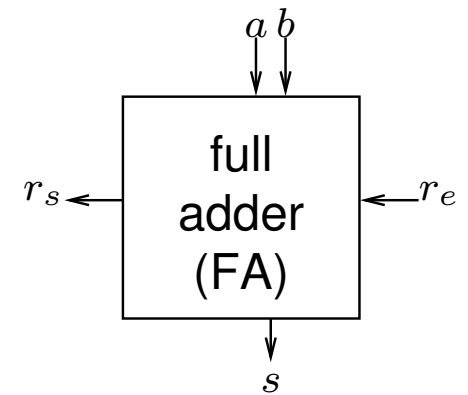
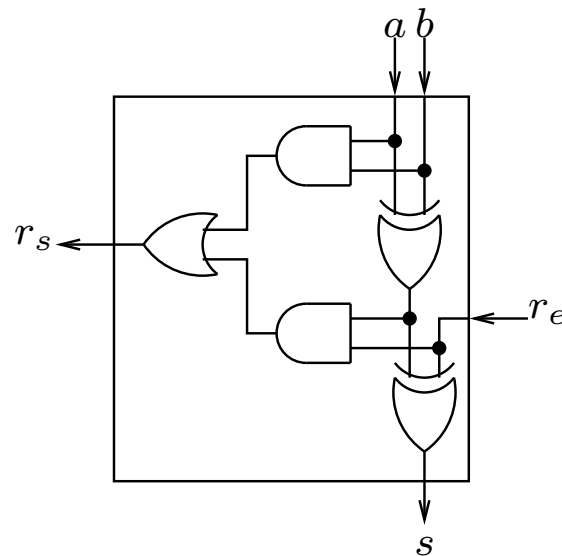
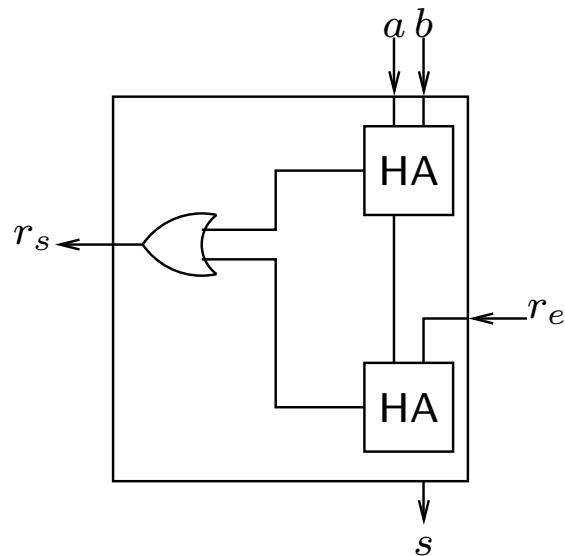
$s$  3 ut

$r_s$  3 ut

# Circuit combinatoire – Logique en tranche

## Additionneur complet

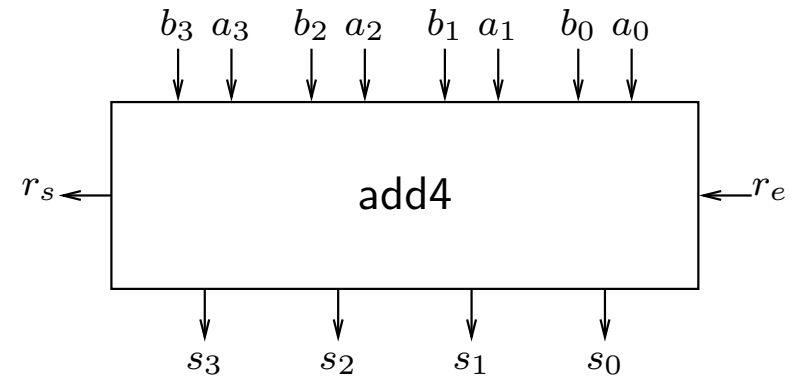
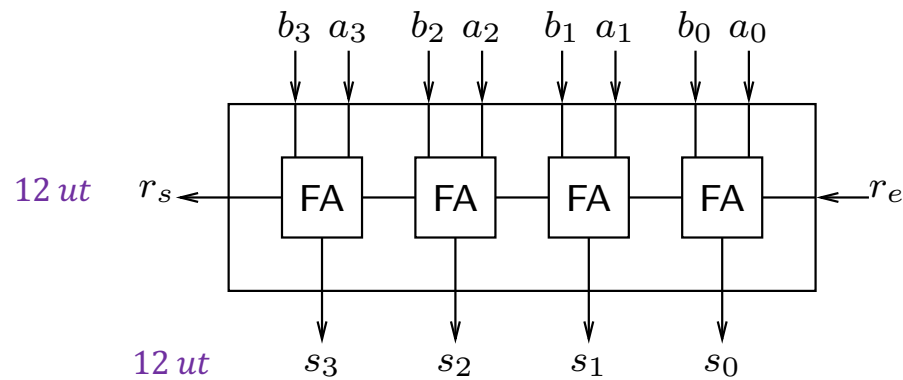
- Addition de deux bits avec une retenue entrante
  - Entrée :  $a$ ,  $b$  (opérandes) et  $r_e$  (retenue d'entrée)
  - Sortie :  $s$  (somme) et  $r_s$  (retenue de sortie)
- Après simplification
  - $s = (a \oplus b) \oplus r_e$
  - $r_s = (a \oplus b)r_e + ab$



# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

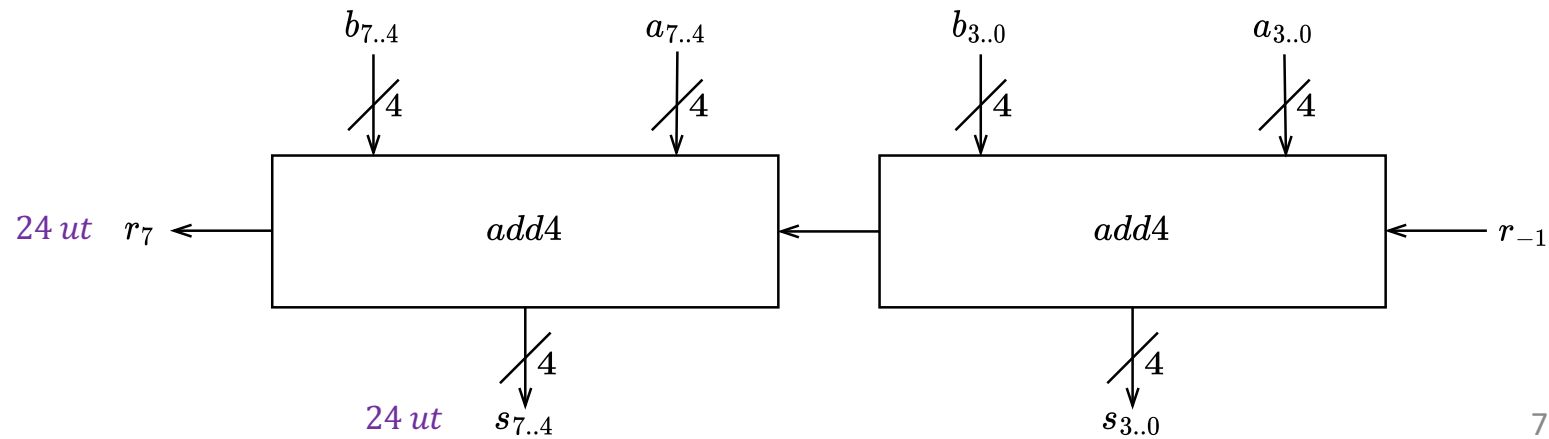
- Propagation simple de la retenue
  - Enchaînement de  $n$  FA
  - Exemple  $n = 4 \rightarrow 12\text{ ut}$



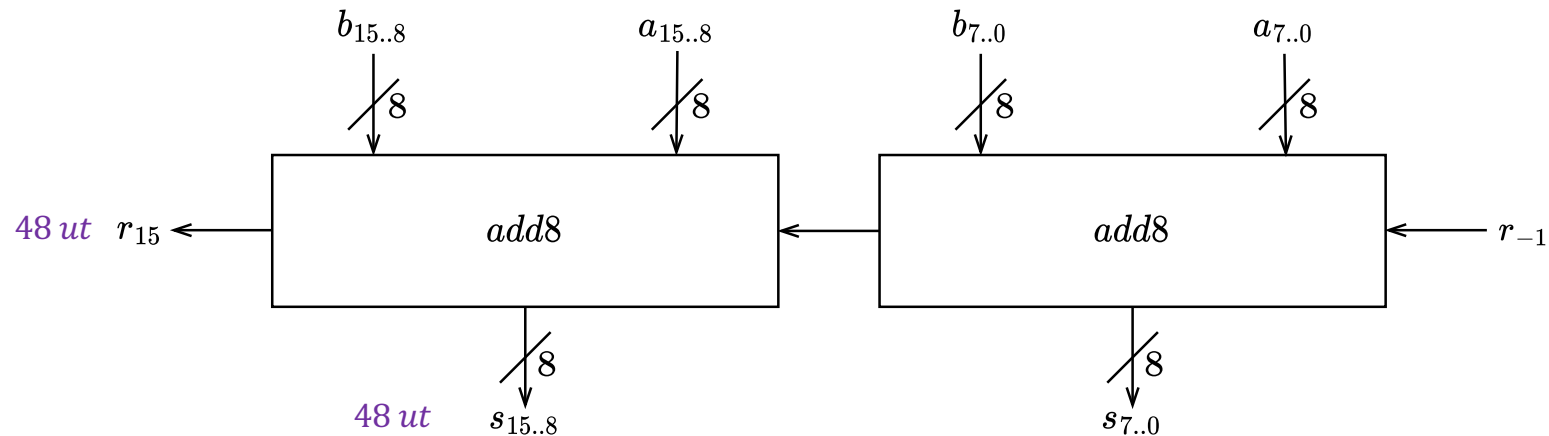
# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

- Propagation simple de la retenue
  - Exemple  $n = 8 \rightarrow 24 ut$



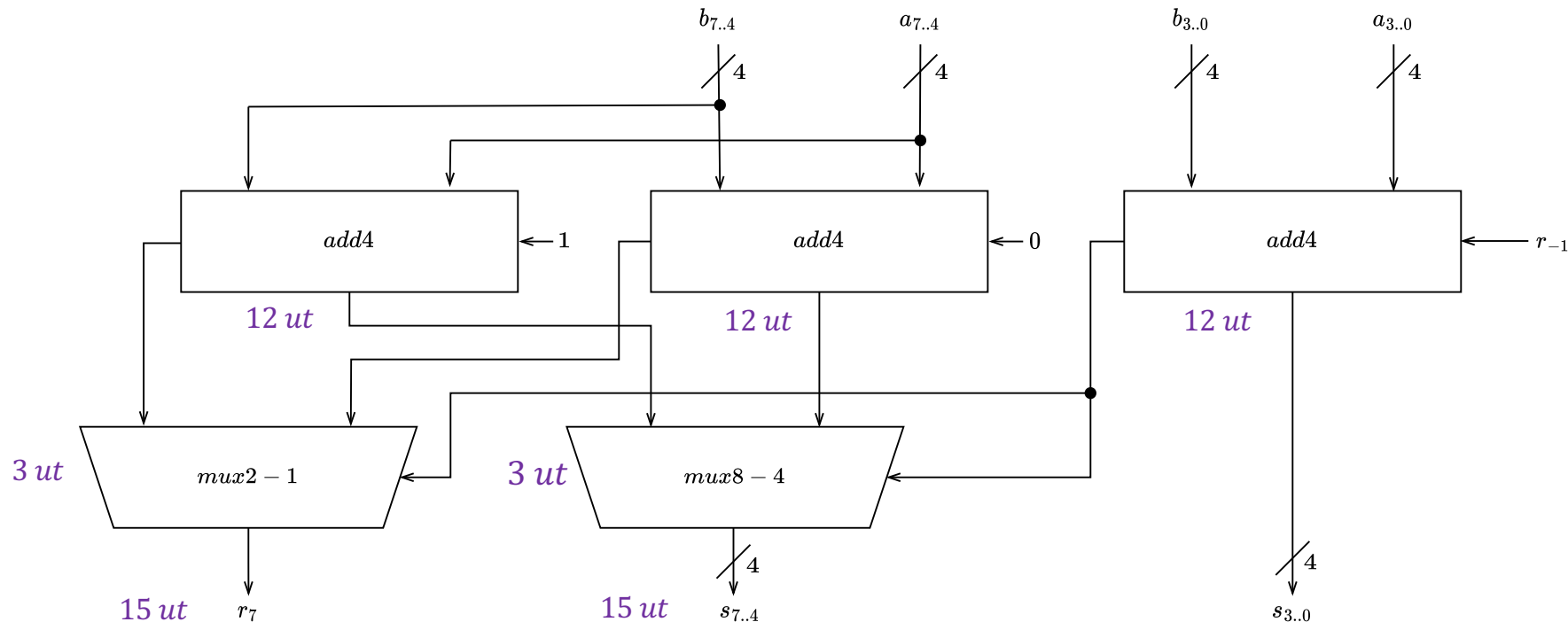
- Exemple  $n = 16 \rightarrow 48 ut$



# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

- Propagation **rapide** de la retenue par **sélection de retenue**
  - On calcule l'addition des  $p$  bits de poids forts pour les **deux** valeurs de la retenue, avec  $n = p + q$ .
  - On ne conserve que le résultat pour la retenue sortant de l'addition des  $q$  bits de poids faible (l'autre a été fait pour rien)
  - Exemple  $n = 8 \rightarrow 15 ut$

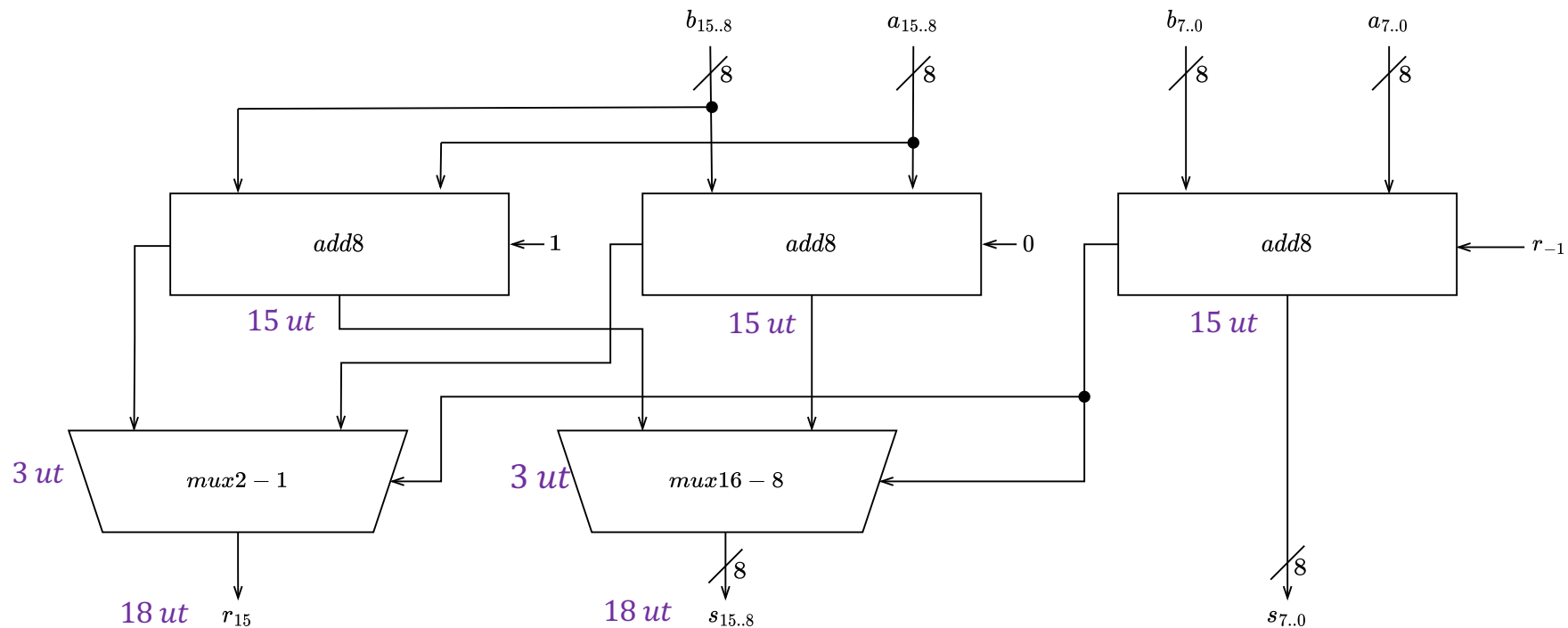




# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

- Propagation **rapide** de la retenue par **sélection de retenue**
    - Exemple  $n = 16$  avec
      - 3  $add8$  (donc 9  $add4$ , 3  $mux8-4$  et 3  $mux2-1$ )
      - 1  $mux16-8$
      - 1  $mux2-1$
- 18 ut



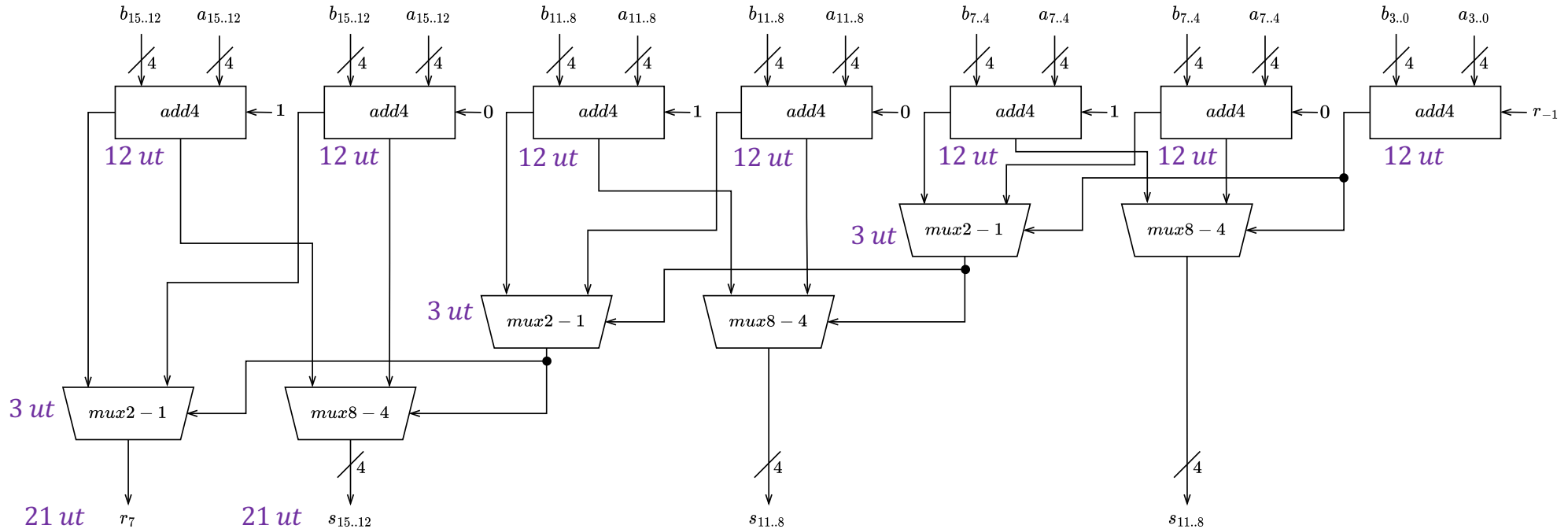
# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

- Propagation rapide de la retenue par sélection de retenue

- Exemple  $n = 16$  avec
  - 7  $add4$
  - 3  $mux8-4$
  - 3  $mux2-1$

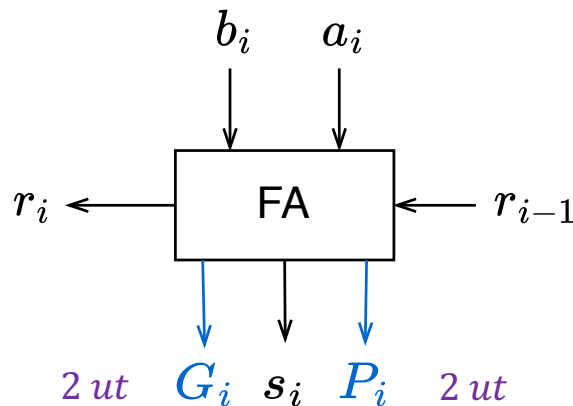
→ 21 ut



# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

- Propagation **rapide** de la retenue par **anticipation de retenue**
  - Anticipation à max  $n$  étages si on dispose de portes d'entrée  $n + 1$
  - On détermine si un FA 1 bit **génère** et/ou **propage** une retenue
    - Ne dépend que des opérandes (et pas de la retenue entrante)

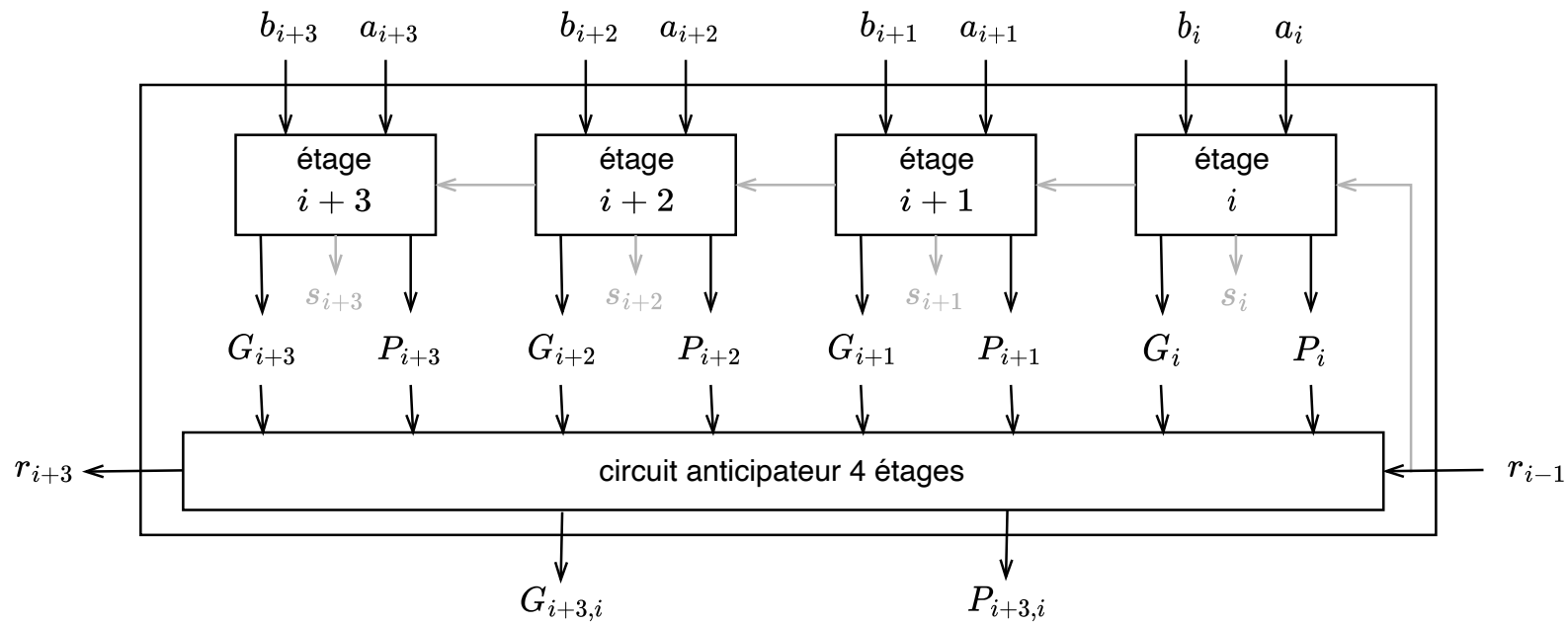


- Fonctions de génération et de propagation
  - $G_i = a_i b_i$
  - $P_i = a_i + b_i$
- En plus elles peuvent servir à calculer  $s$ 
  - $s_i = \bar{G}_i P_i \oplus r_{i-1}$

# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

- Propagation **rapide** de la retenue par **anticipation de retenue**
  - Exemple : anticipateur à 4 étages (4 bits)
    - Bloc de 4 étages avec circuit anticipateur

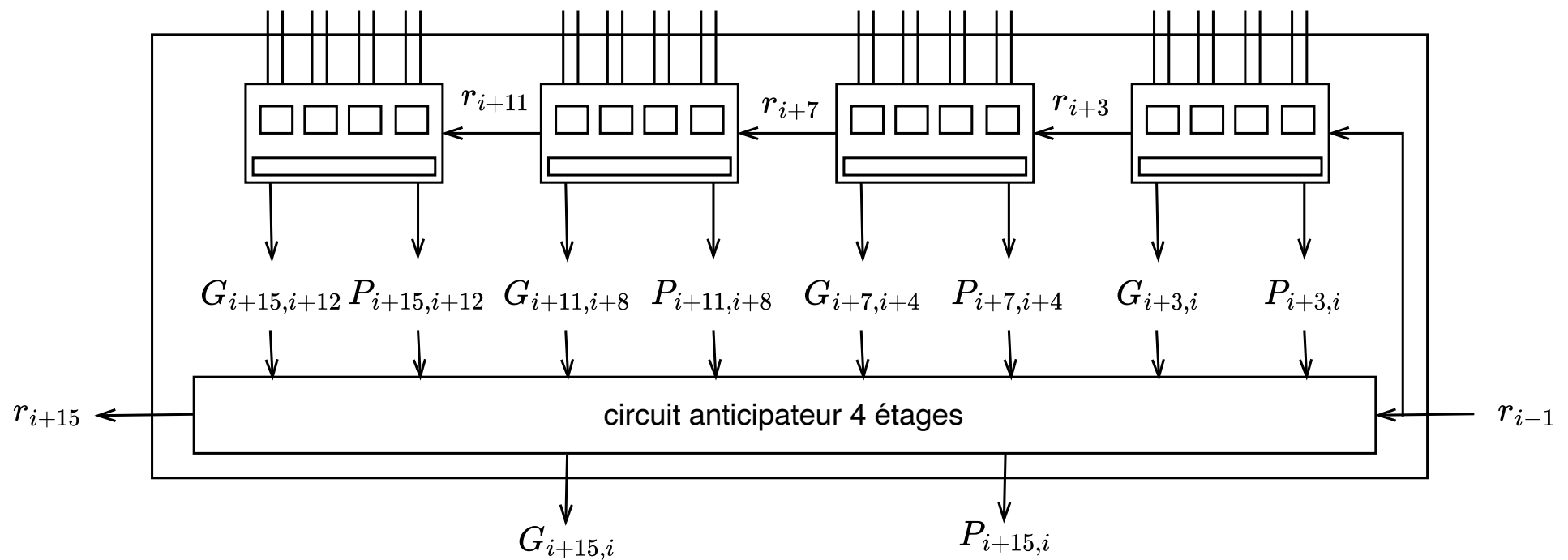


- $$r_{i+3} = G_{i+3} + P_{i+3}G_{i+2} + P_{i+3}P_{i+2}G_{i+1} + P_{i+3}P_{i+2}P_{i+1}G_i + P_{i+3}P_{i+2}P_{i+1}P_i r_{i-1}$$

# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

- Propagation **rapide** de la retenue par **anticipation de retenue**
  - Exemple : anticipateur à 4 étages (4 bits)
    - Bloc de 4x4 étages avec circuit anticipateur
    - Fonctions de génération et propagation par bloc de 4 étages :
      - $G_{i+3,i} = G_{i+3} + P_{i+3}G_{i+2} + P_{i+3}P_{i+2}G_{i+1} + P_{i+3}P_{i+2}P_{i+1}G_i$
      - $P_{i+3,i} = P_{i+3}P_{i+2}P_{i+1}P_i$

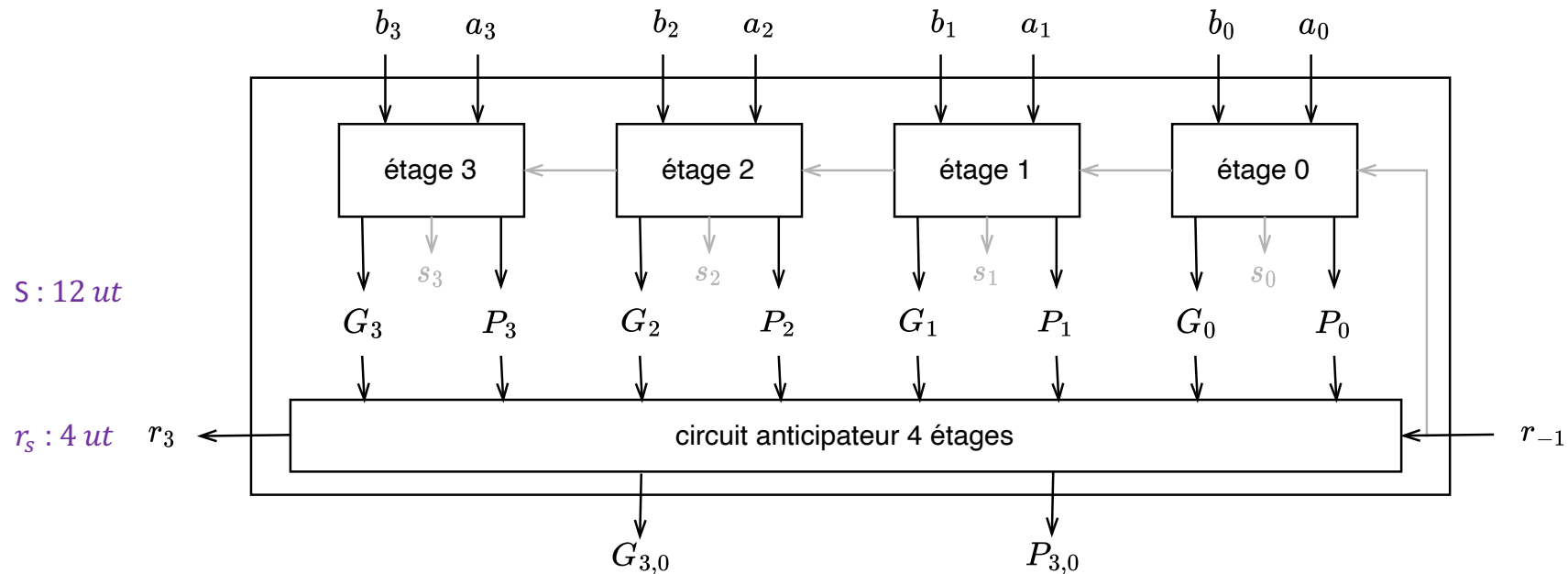


- $r_{i+15} = G_{i+15,i+12} + P_{i+15,i+12}G_{i+11,i+8} + P_{i+15,i+12}P_{i+11,i+8}G_{i+7,i+4} + P_{i+15,i+12}P_{i+11,i+8}P_{i+7,i+4}G_{i+3,i} + P_{i+15,i+12}P_{i+11,i+8}P_{i+7,i+4}P_{i+3,i}r_{i-1}$

# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

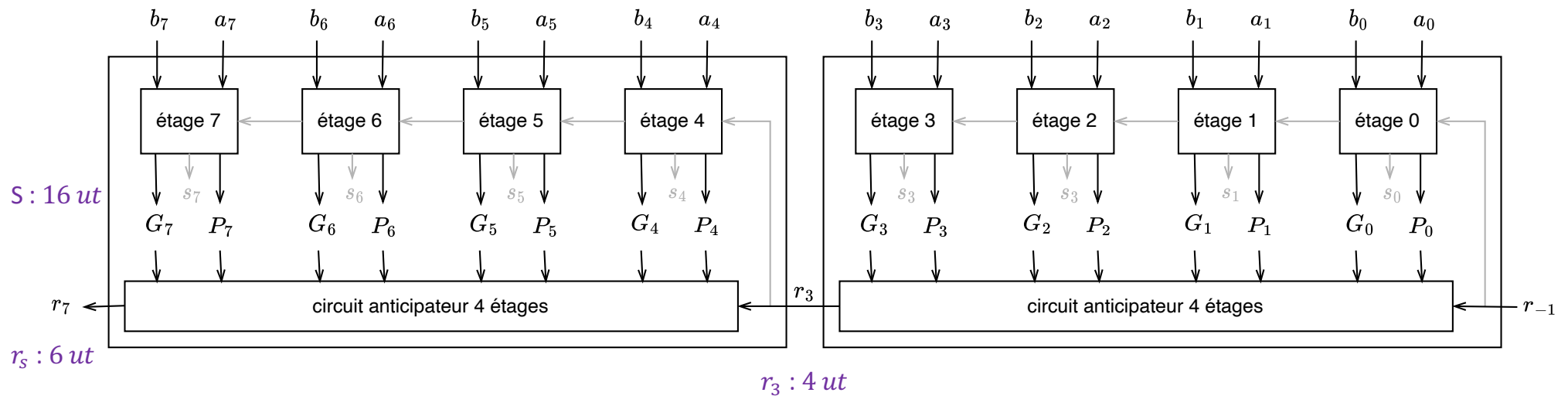
- Propagation **rapide** de la retenue par **anticipation de retenue**
  - Exemple : anticipateur à 4 étages (4 bits)
    - Exemple  $n = 4 \rightarrow S : 12 \text{ ut}$ ,  $r_s : 4 \text{ ut}$



# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

- Propagation **rapide** de la retenue par **anticipation de retenue**
  - Exemple : anticipateur à 4 étages (4 bits)
    - Exemple  $n = 8 \rightarrow S : 16 \text{ ut}$ ,  $r_s : 6 \text{ ut}$



# Circuit combinatoire – Logique en tranche

## Additionneur complet $n$ bits

- Propagation **rapide** de la retenue par **anticipation de retenue**
  - Exemple : anticipateur à 4 étages (4 bits)
    - Exemple  $n = 16 \rightarrow S : 20 \text{ ut}, r_s : 6 \text{ ut}$

