

ARCHI – Architecture des ordinateurs

Sylvain Brandel

2025 – 2026

sylvain.brandel@univ-lyon1.fr

Partie 4

CIRCUITS COMBINATOIRES

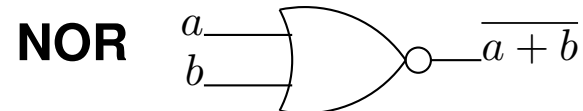
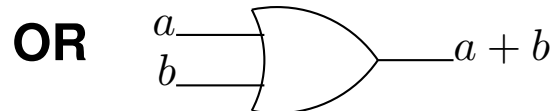
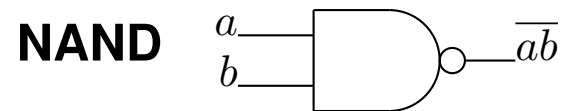
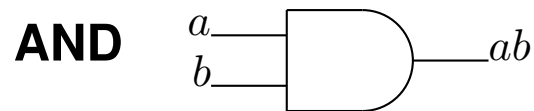
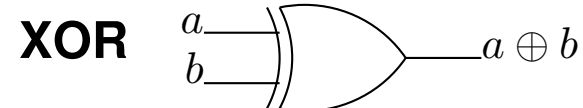
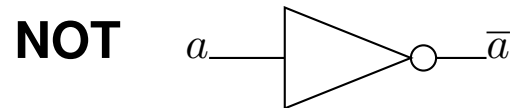
Logique anarchique

Logique structurée

Logique en tranche

Algèbre de Boole

- Rappel



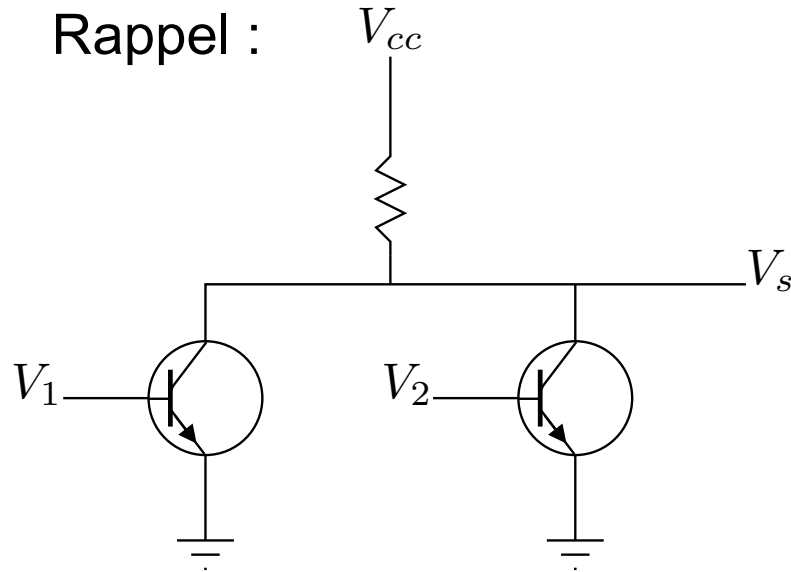
- Actuellement : transistors, électricité ...
 - Mais d'autres possibilités existent

<https://www.youtube.com/watch?v=CNbScb8v-MI>

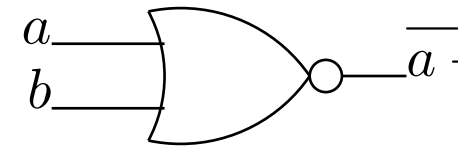
Niveau 0

Portes logiques

- Rappel :



V1	V2	Vs
0	0	1
0	1	0
1	0	0
1	1	0



- Porte NAND et NOR : on sait faire
 - nMOS : passant haut
 - pMOS : passant bas
 - CMOS
- Inverseurs → retard

Circuit combinatoire

- Circuit combinatoire bien formé (CCBF) :
 - Une porte de base est un CCBF
 - Un fil est un CCBF
 - Deux CCBF disjoints forment un CCBF
 - Un CCBF dont on a connecté les sorties aux entrées d'un autre CCBF est un CCBF
 - Deux CCBF dont on a connecté les entrées est un CCBF
- Pas de cycle
- Pas de connexion des sorties entre elles

Circuit combinatoire – Logique anarchique

- Nombre minimal de portes de base
- Moins de portes et moins d'entrées → moins de puissance
- Simplification d'une DNF : Karnaugh
 - Seulement pour peu d'entrées
 - Basé sur un code de Gray
 - Tableau des m_i : un seul changement entre deux cases adjacentes
 - Groupes de 1 :
 - Tous les 1 → dans un groupe
 - Groupes les plus gros possibles
 - Limiter les redondances
 - Profiter des entrées incomplètes
 - Complicé pour plus de 6 entrées
- Simplification d'une DNF : Quine – Mc Cluskey (Espresso, Mc Boole ...)



Circuit combinatoire – Logique structurée

- Minimisation de la surface
- Utilisation de structures régulières
- Décodeur
- ROM : *Read Only Memory*
- PLA : *Programmable Logic Array*
- LUT : *Look-Up Table*
- Multiplexeur / Démultiplexeur

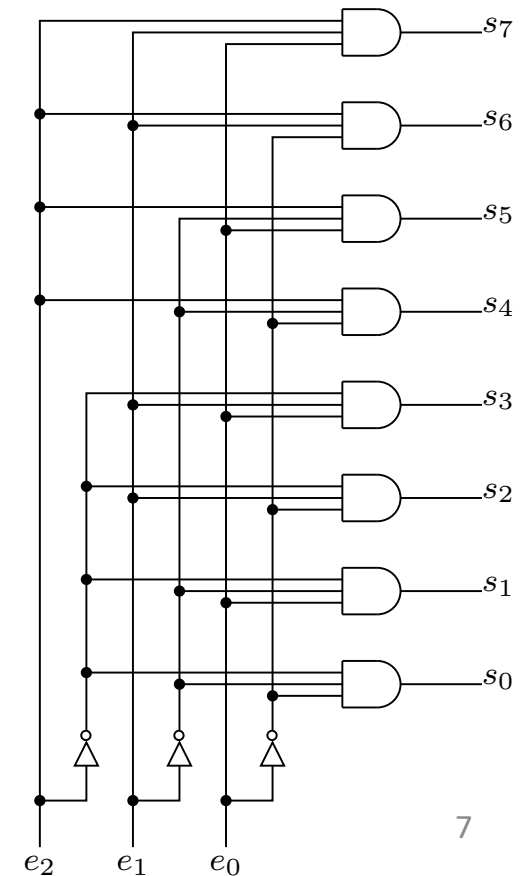
Circuit combinatoire – Logique structurée

Décodeur

- Décodeur n vers 2^n
 - n entrées e_i : l'entier $(e_{n-1} \dots e_0)_2$
 - 2^n sorties s_i indicées de 0 à $2^n - 1$
 - Unique ligne de sortie active : ligne $s_{(e_{n-1} \dots e_0)_2}$

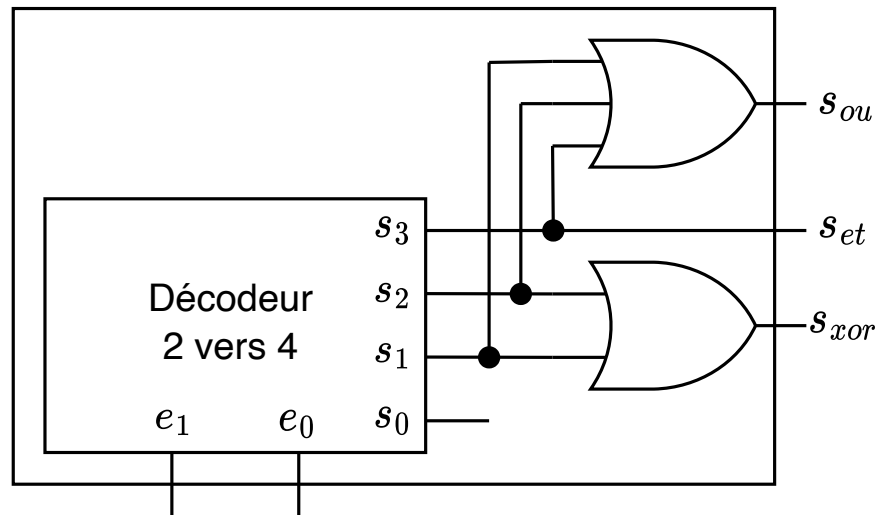
(démultiplexeur)

- Ex : Décodeur 3 vers 8



Circuit combinatoire – Logique structurée

- ROM : *Read Only Memory*
 - Décodeur complet + union des termes produit à 1
- Exemple : ROM avec
 - 2 entrées (donc décodeur complet 2 vers 4)
 - 3 sorties : e_0 et e_1 , e_0 ou e_1 , e_0 xor e_1

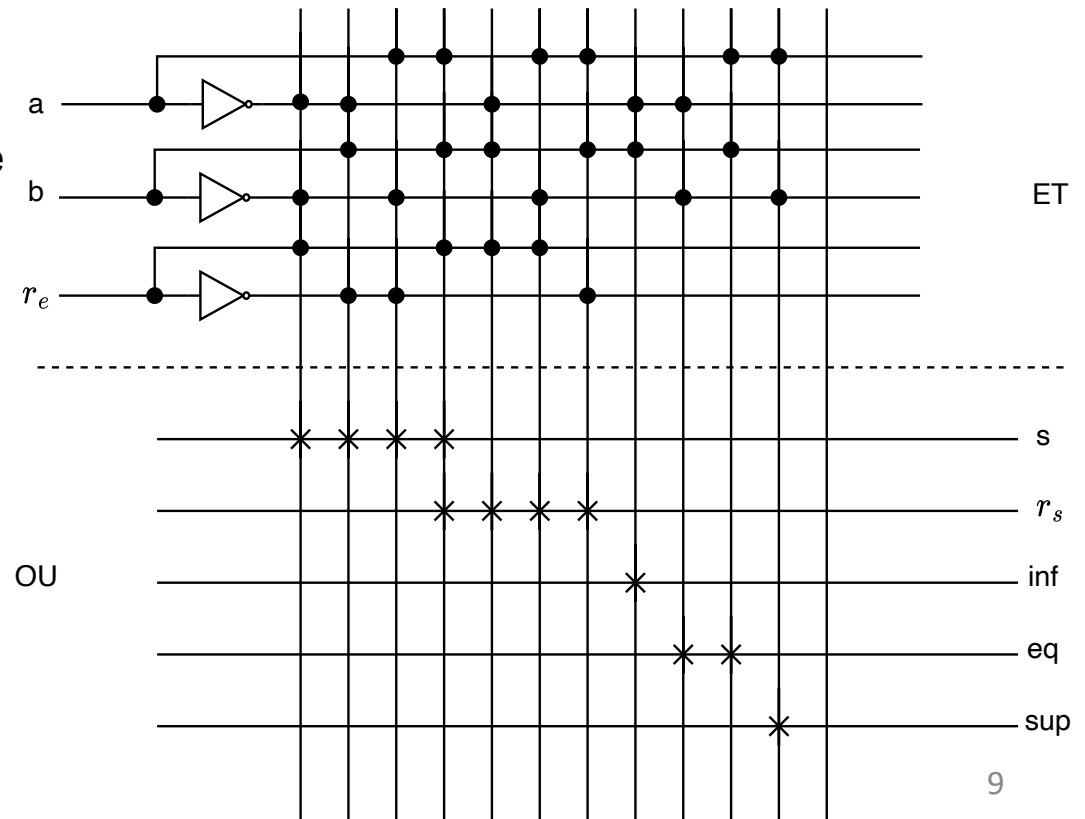


Circuit combinatoire – Logique structurée

- PLA : *Programmable Logic Array*
 - Décodeur partiel programmable + union programmable
 - Constitué de
 - Un demi-PLA ET : générateur partiel de termes produits
 - Un demi-PLA OU : union des termes produits pour lesquels la fonction = 1

- PAL : PLA simplifié
 - La partie ET est programmable
 - La partie OU est pré-câblée

- Exemple de PLA

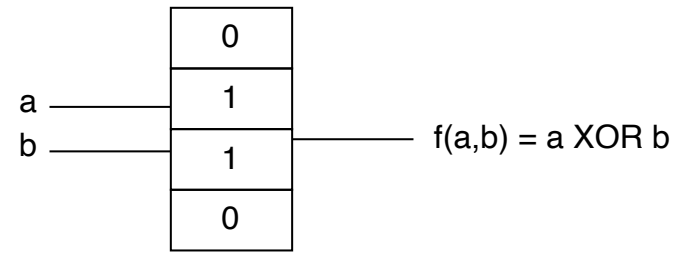


Circuit combinatoire – Logique structurée

- LUT : *Look-Up Table*

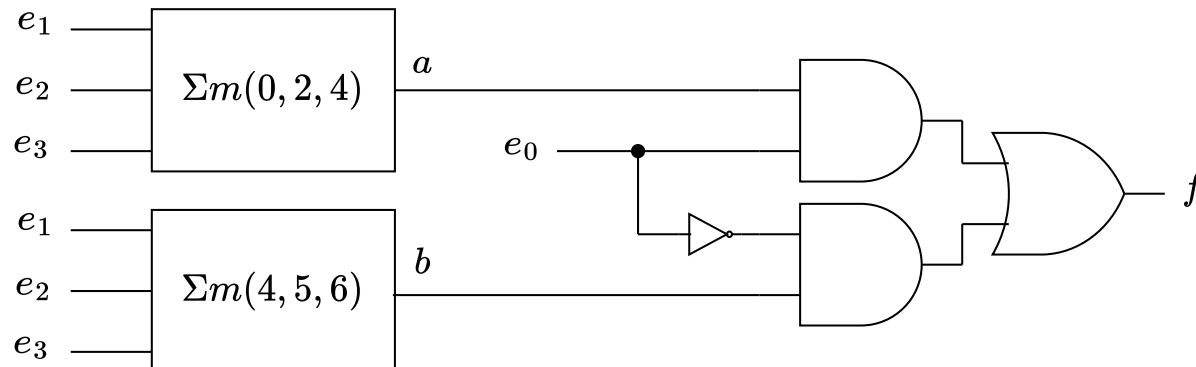
- Tables de vérité 2, 3 ou 4 entrées stockées dans une RAM → FPGA
- Une LUT-2 peut implémenter n'importe laquelle des 16 fonctions à 2 entrées

- Exemple : XOR avec LUT-2



- Exemple : $f(e_0, e_1, e_2, e_3) = \Sigma m(1,5,8,9,10,12)$ avec LUT-3

$$\begin{aligned} f &= \bar{e}_3 \bar{e}_2 \bar{e}_1 e_0 + \bar{e}_3 e_2 \bar{e}_1 e_0 + e_3 \bar{e}_2 \bar{e}_1 \bar{e}_0 + e_3 \bar{e}_2 \bar{e}_1 e_0 + e_3 \bar{e}_2 e_1 \bar{e}_0 + e_3 e_2 \bar{e}_1 \bar{e}_0 \\ &= (\bar{e}_3 \bar{e}_2 \bar{e}_1 + \bar{e}_3 e_2 \bar{e}_1 + e_3 \bar{e}_2 \bar{e}_1) e_0 + (e_3 \bar{e}_2 \bar{e}_1 + e_3 \bar{e}_2 e_1 + e_3 e_2 \bar{e}_1) \bar{e}_0 \\ &= a e_0 + b \bar{e}_0 \quad \text{avec } a = \Sigma m(0,2,4) \text{ et } b = \Sigma m(4,5,6) \end{aligned}$$

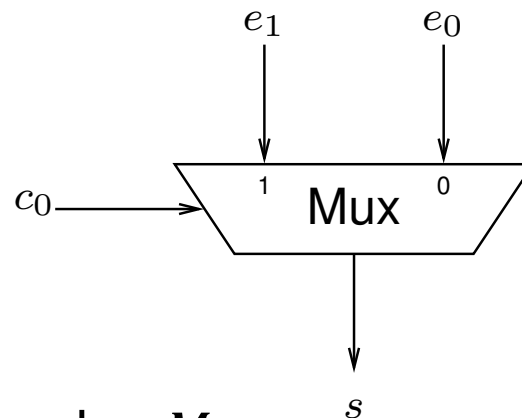


Circuit combinatoire – Logique structurée

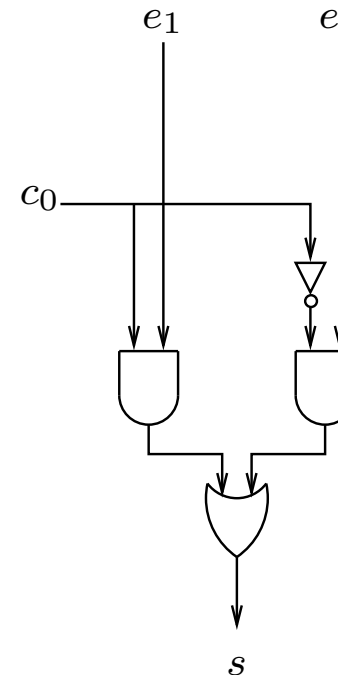
Multiplexeur

- Multiplexeur 2^n vers 1
 - 2^n entrées e_i indicées de 0 à $2^n - 1$
 - n lignes de sélection : l'entier $(c_{n-1} \dots c_0)_2$
 - 1 sortie s
 - Une des entrées est **sélectionnée** en fonction des lignes de sélection
 - Sortie : $s = e_{(c_{n-1} \dots c_0)_2}$

- Ex : Mux_2



- Ex : Mux_4
- Ex : Mux_4 avec des Mux_2
- Ex : Implémentation de DNF

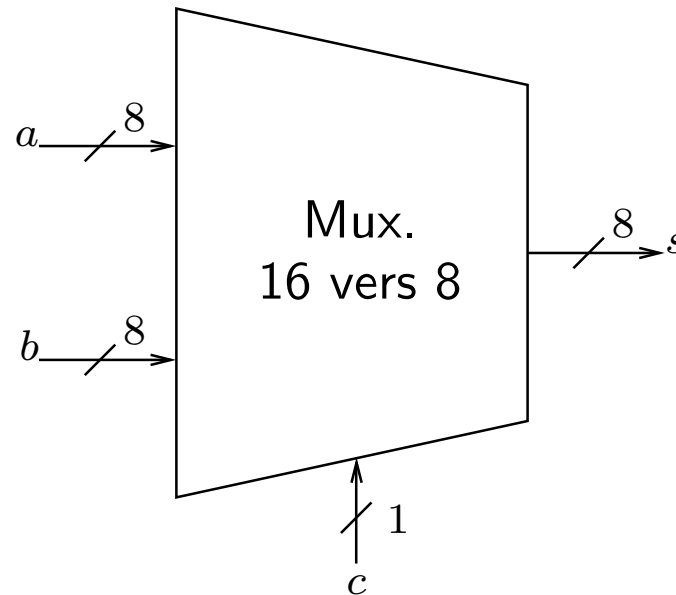


Circuit combinatoire – Logique structurée

Multiplexeur

- Multiplexeur $k \cdot 2^n$ vers k
 - $k \cdot 2^n$ entrées
 - n lignes de sélection
 - k sorties
 - k entrées sont sélectionnées en fonction des lignes de sélection

- Ex : Multiplexeur 16 vers 8



- Ex : Multiplexeur 8 vers 4 avec des Multiplexeurs 2 vers 1

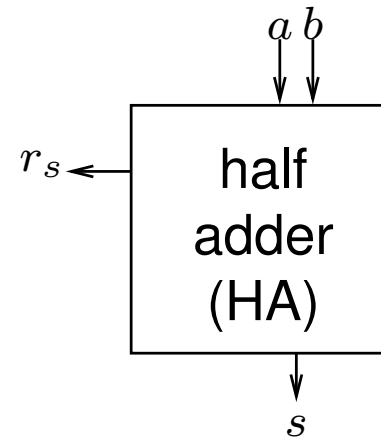
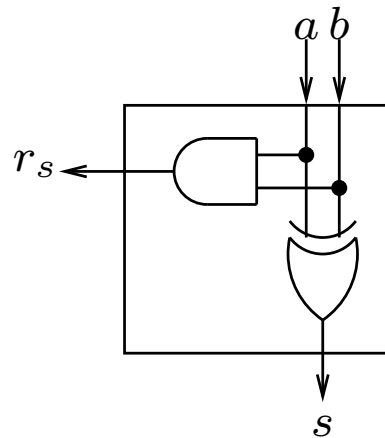
Circuit combinatoire – Logique en tranche

- 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

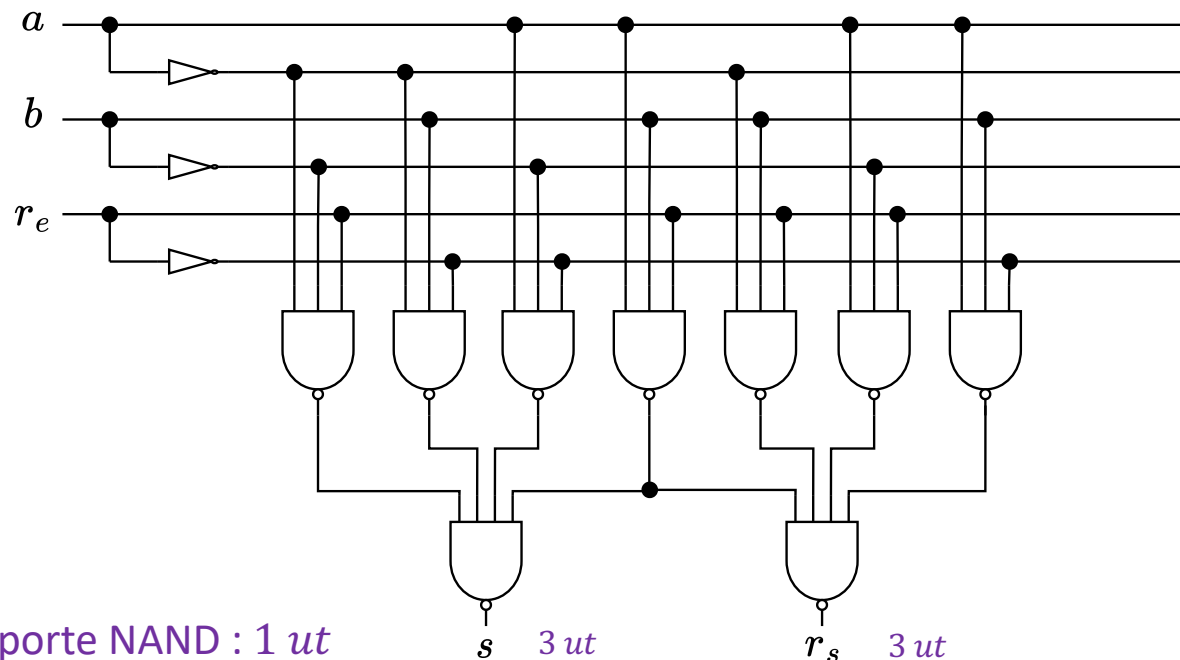
- Addition simple de deux bits
 - Entrée : a et b (opérandes)
 - Sortie : s (somme) et r_s (retenue de sortie)
- D'après la table de vérité
 - $s = \bar{a}b + a\bar{b}$
 - $r_s = ab$
- Après simplification
 - $s = a \oplus b$
 - $r_s = ab$



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
 - $s = \bar{a}\bar{b}r_e + \bar{a}b\bar{r}_e + a\bar{b}\bar{r}_e + ab r_e$
 - $r_s = \bar{a}b r_e + a\bar{b} r_e + ab\bar{r}_e + ab r_e$

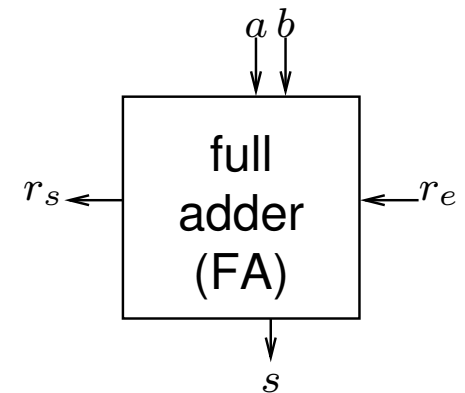
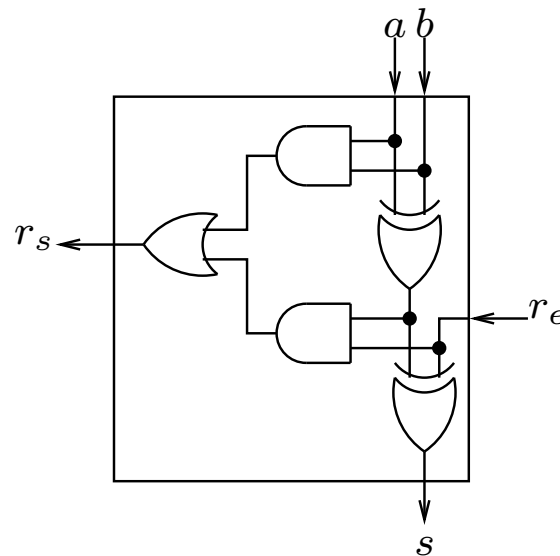
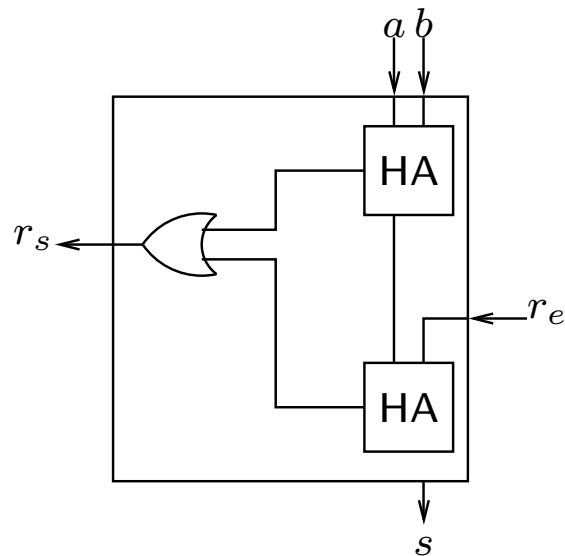


Temps de passage porte NAND : 1 ut

Circuit combinatoire – Logique en tranche

Additionneur complet

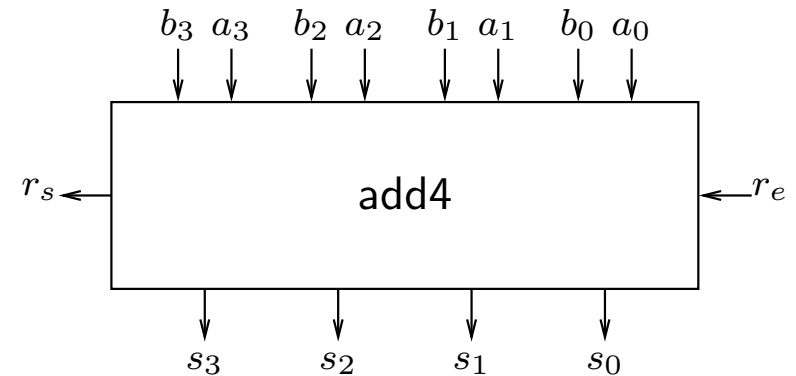
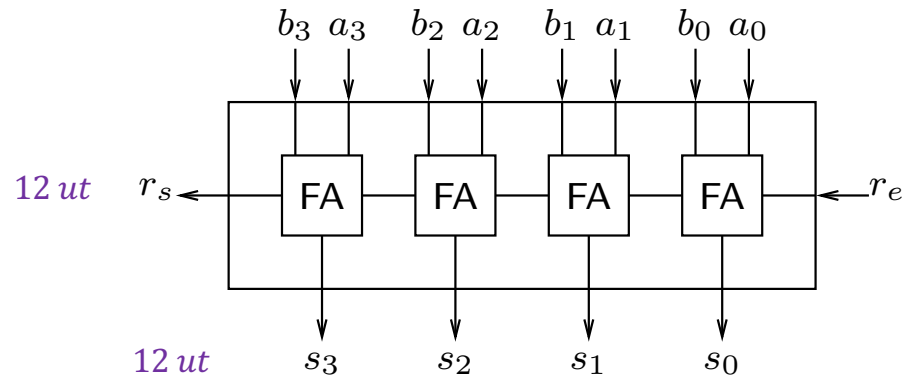
- Addition de deux bits avec une retenue entrante
 - Entrée : a , b (opérandes) et r_e (retenue d'entrée)
 - Sortie : s (somme) et r_s (retenue de sortie)
- Après simplification
 - $s = (a \oplus b) \oplus r_e$
 - $r_s = (a \oplus b)r_e + ab$



Circuit combinatoire – Logique en tranche

Additionneur complet n bits

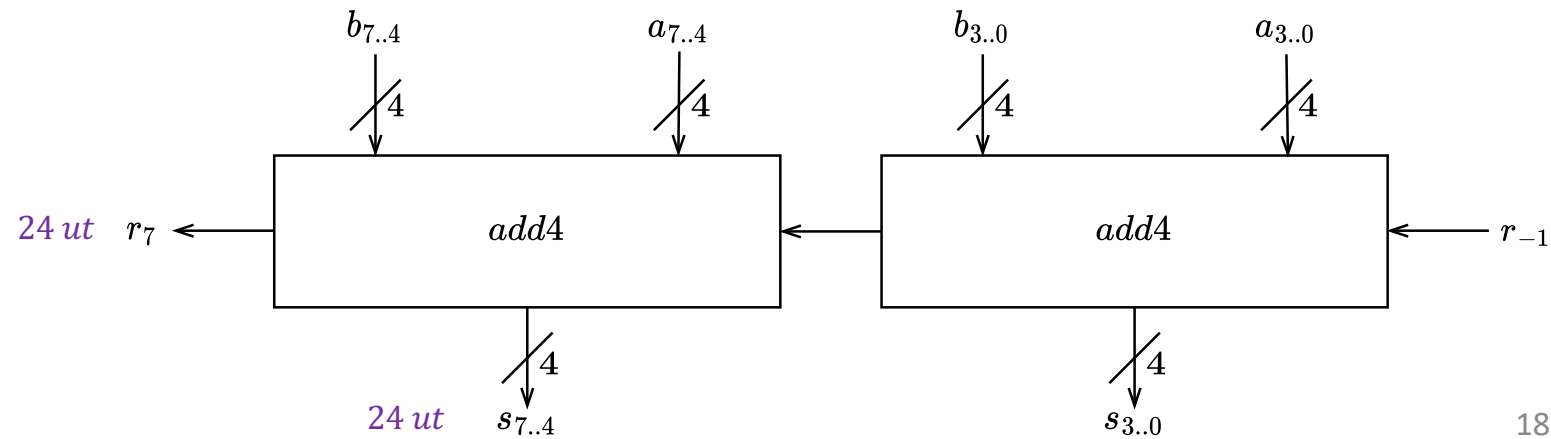
- Propagation **simple** de la retenue
 - Enchainement de n FA
 - Exemple $n = 4 \rightarrow 12\text{ ut}$



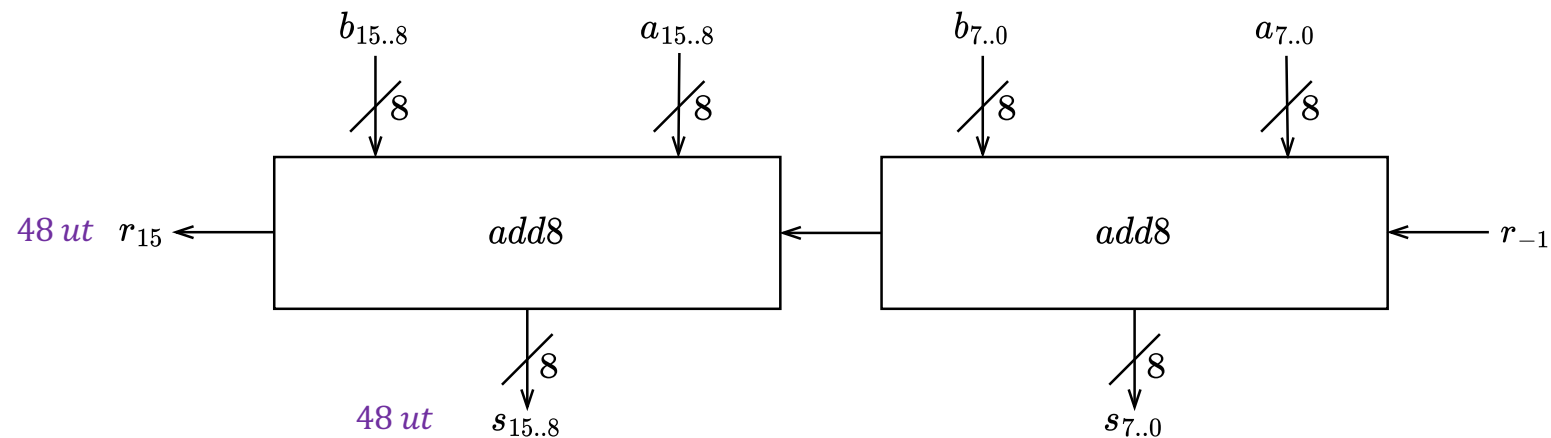
Circuit combinatoire – Logique en tranche

Additionneur complet n bits

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



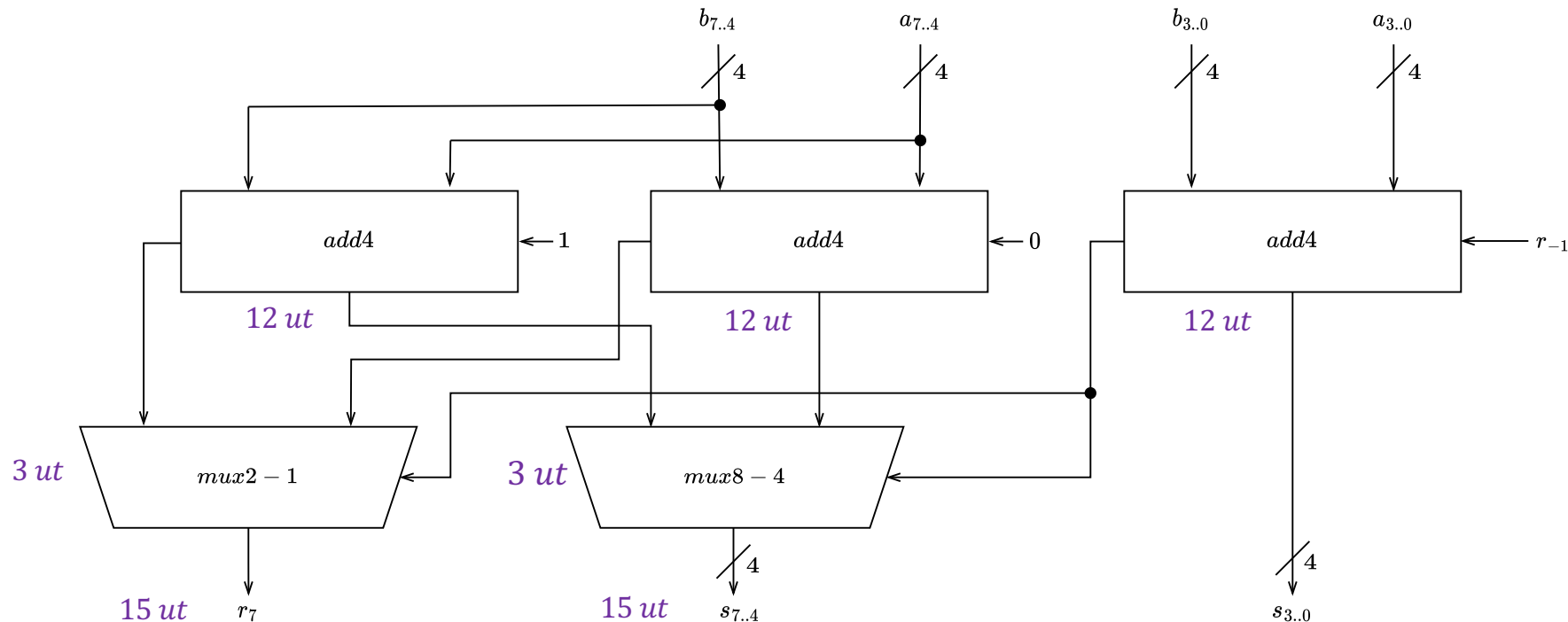
- Exemple $n = 16 \rightarrow 48 \text{ ut}$



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 $n = 8 \rightarrow 15\text{ ut}$

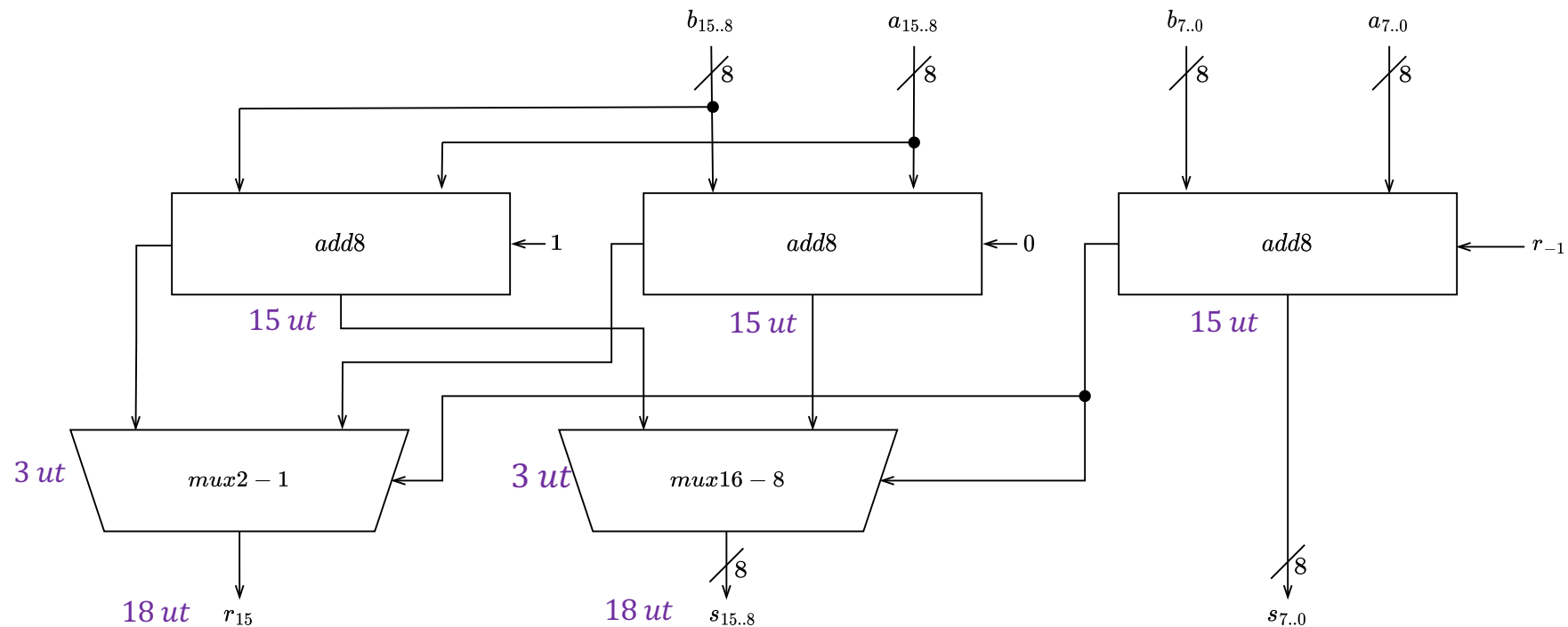


Circuit combinatoire – Logique en tranche

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

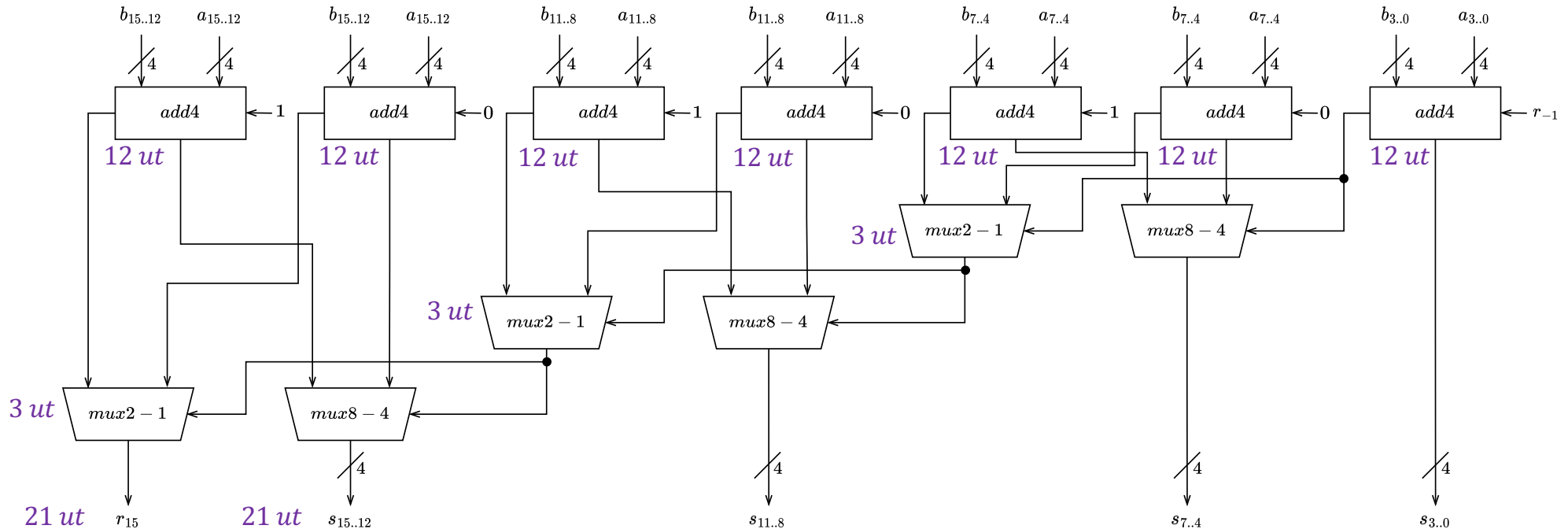
Additionneur complet n bits

- Propagation rapide de la retenue par sélection de retenue

– Exemple $n = 16$ avec

- 7 $add4$
- 3 $mux8-4$
- 3 $mux2-1$

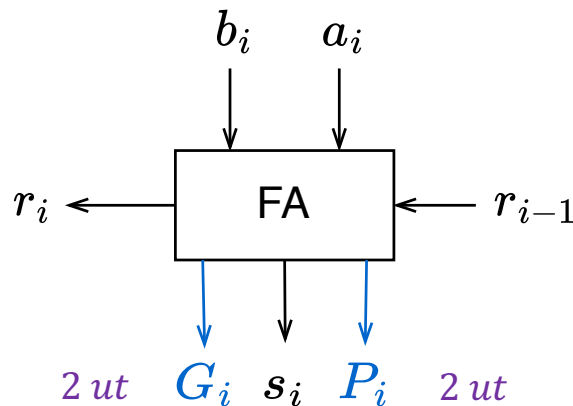
→ 21 ut



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)

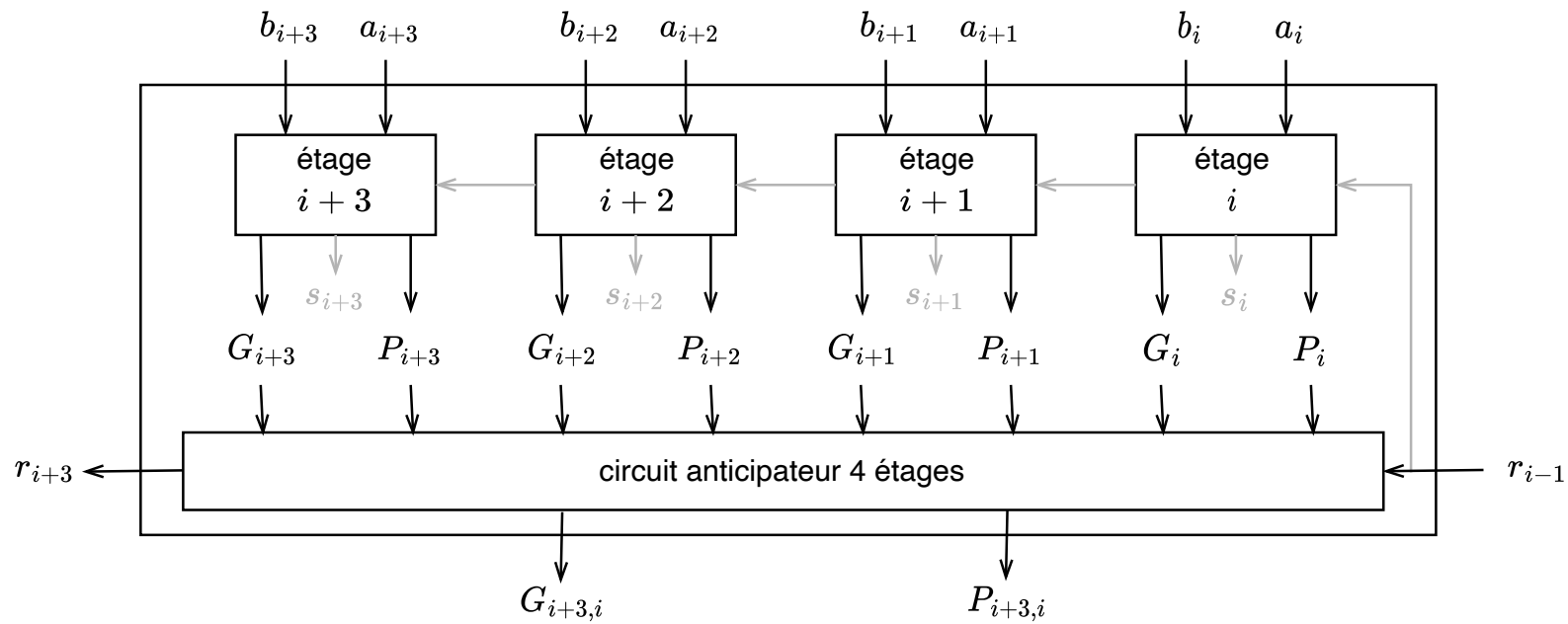


- Fonctions de génération et de propagation
 - $G_i = a_i b_i$
 - $P_i = a_i + b_i$
- En plus elles peuvent servir à calculer s
 - $s_i = \bar{G}_i P_i \oplus r_{i-1}$

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

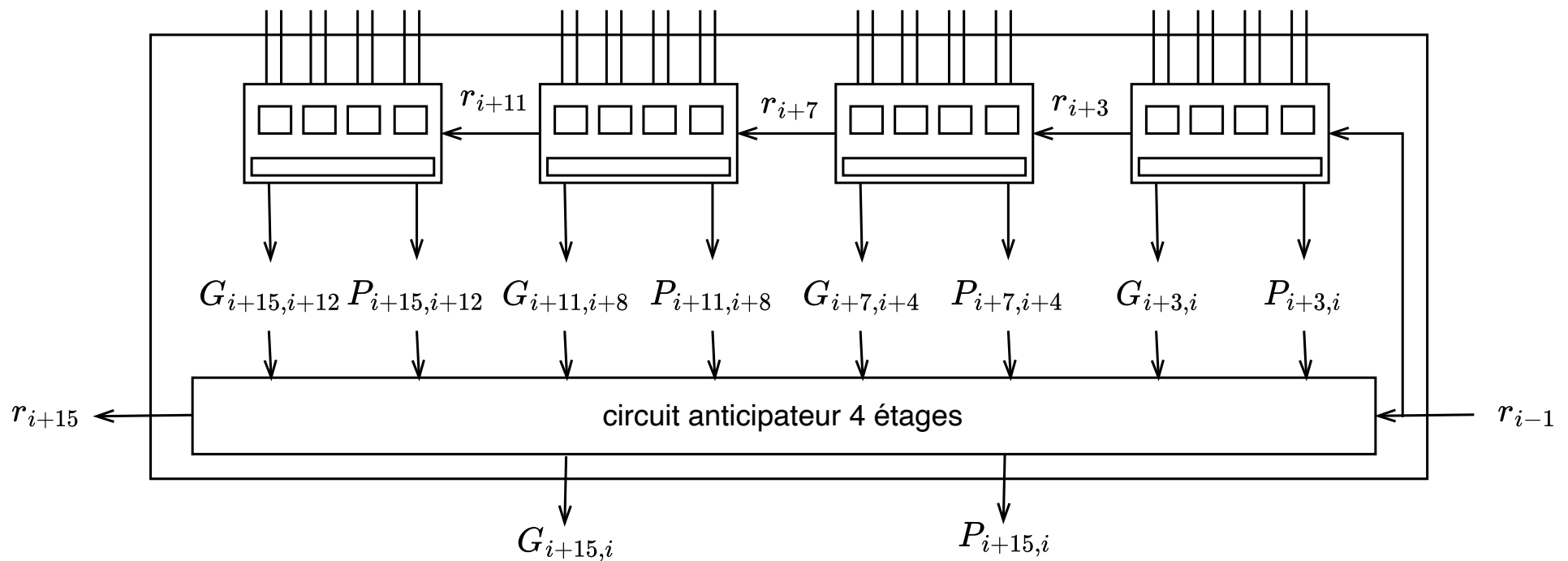


- $$r_{i+3} = G_{i+3} + P_{i+3}G_{i+2} + P_{i+3}P_{i+2}G_{i+1} + P_{i+3}P_{i+2}P_{i+1}G_i + P_{i+3}P_{i+2}P_{i+1}P_i r_{i-1}$$

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

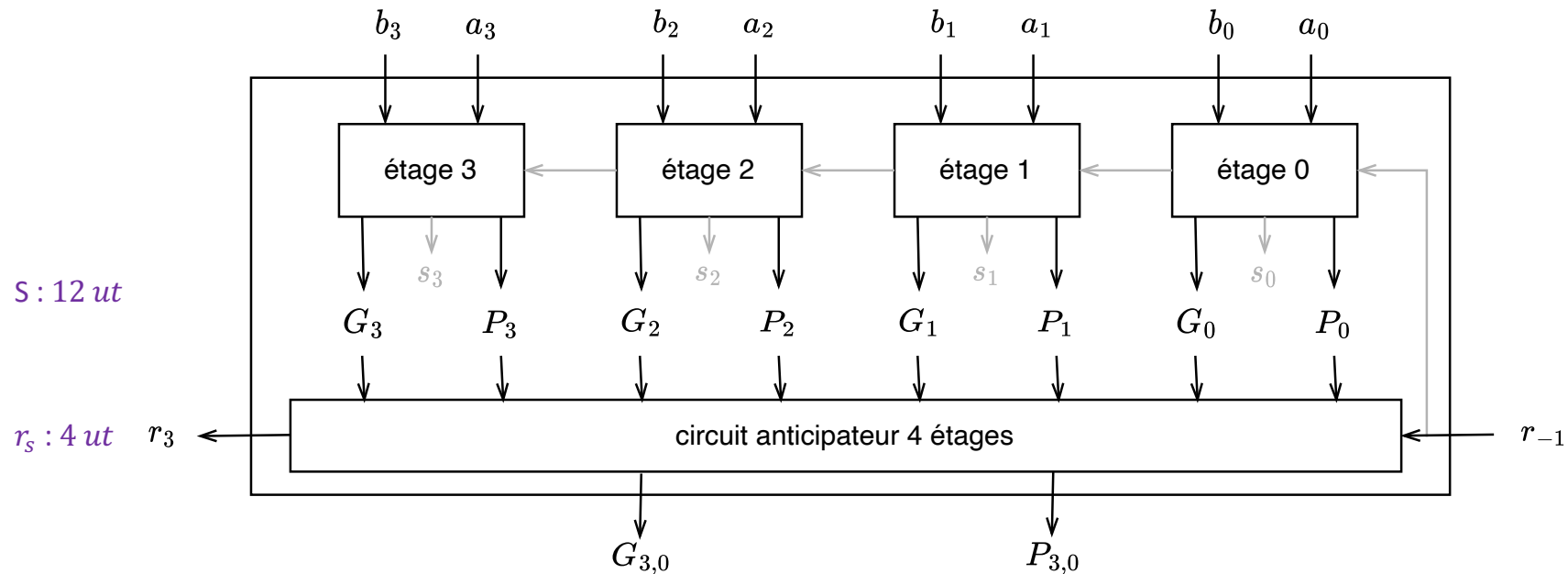


$$r_{i+15} = G_{i+15,i+12} + P_{i+15,i+12}G_{i+11,i+8} + P_{i+15,i+12}P_{i+11,i+8}G_{i+7,i+4} + P_{i+15,i+12}P_{i+11,i+8}P_{i+7,i+4}G_{i+3,i} + P_{i+15,i+12}P_{i+11,i+8}P_{i+7,i+4}P_{i+3,i}r_{i-1}$$

Circuit combinatoire – Logique en tranche

Additionneur complet n bits

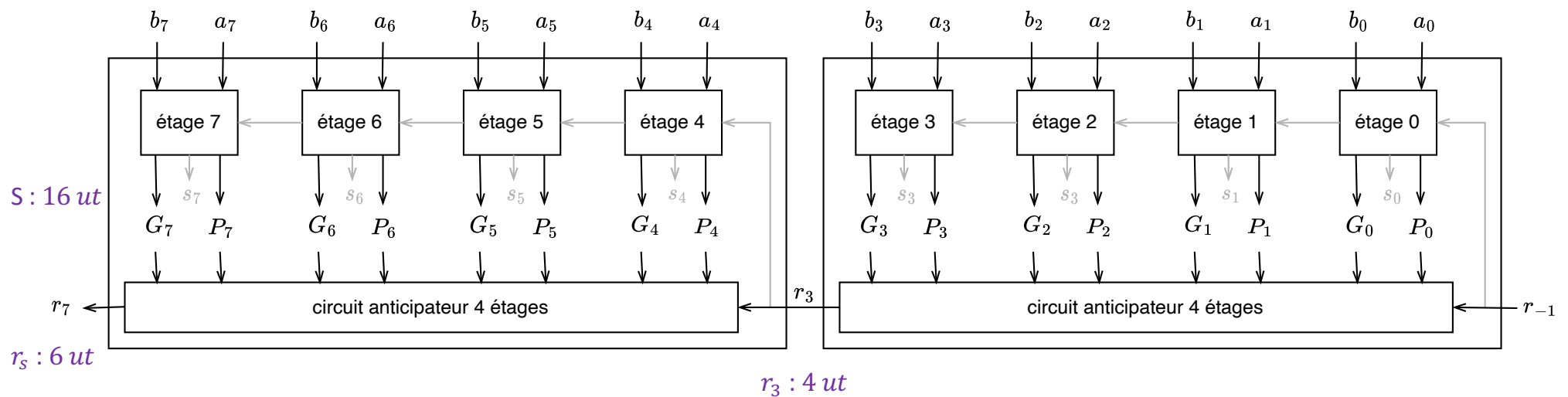
- Propagation **rapide** de la retenue par **anticipation de retenue**
 - Exemple : anticipateur à 4 étages (4 bits)
 - Exemple $n = 4 \rightarrow S : 12\text{ ut}$, $r_s : 4\text{ ut}$



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 $n = 16 \rightarrow S : 20 \text{ ut}, r_s : 6 \text{ ut}$

