Arithmétique binaire

M. Combacau
combacau@laas.fr

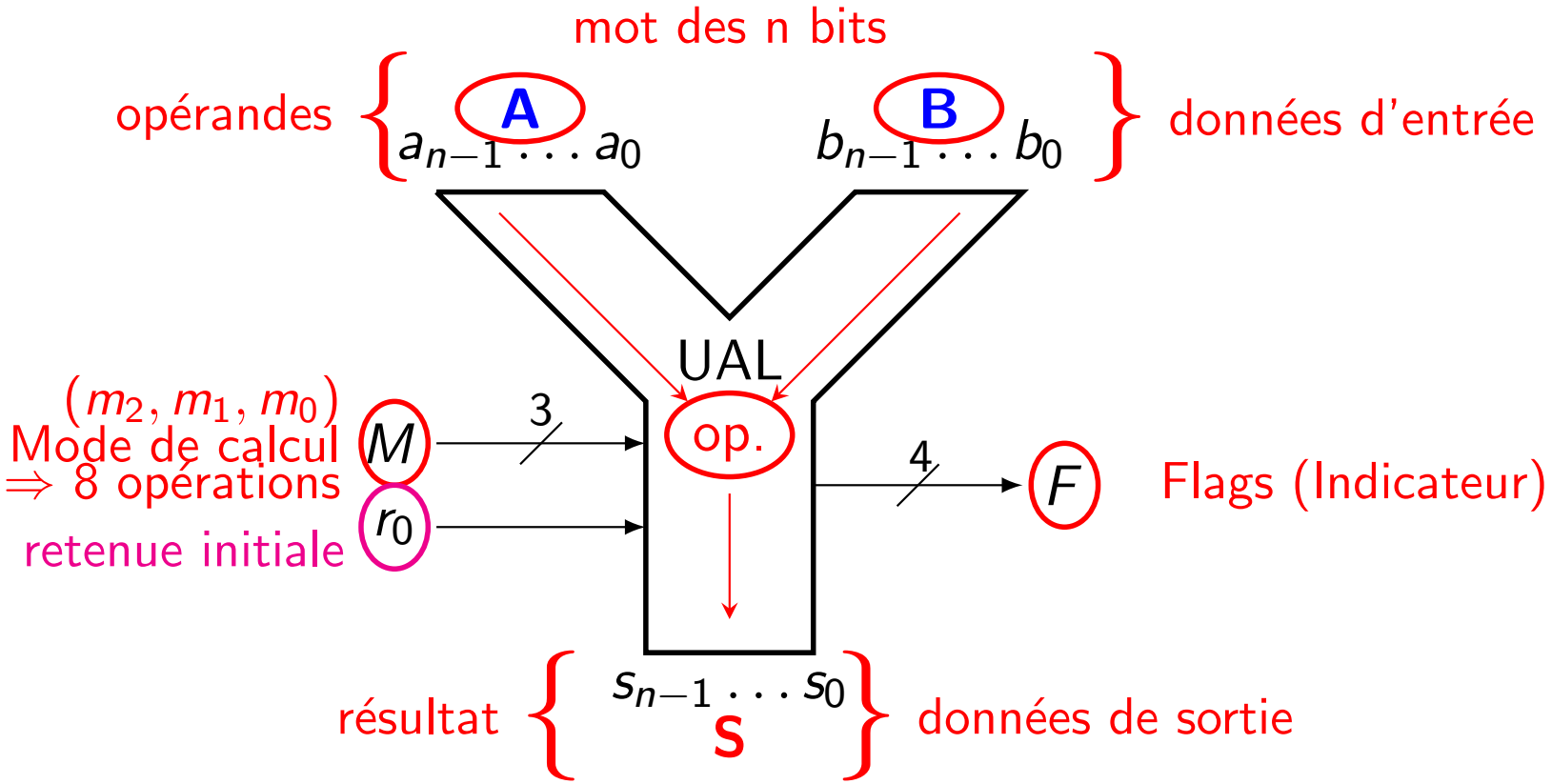
Université Paul Sabatier
LAAS-CNRS

10 novembre 2024

Objectif

Construire une unité arithmétique et logique minimale

Vue systémique d'une UAL (entrées/sorties)



Mode de fonctionnement de l'UAL

M :($\underbrace{m_2, m_1, m_0}_{\text{8 combinaisons}}$) $\Rightarrow$ sélection de l'opération à effectuer (Mode)

m_2	m_1	m_0	opération sélectionnée	
0	0	0	logique NON (A)	logique
0	0	1	logique A ET B	logique
0	1	0	logique A OU B	logique
0	1	1	logique A XOR B	logique
1	0	0	logique déclage à gauche	décalage
1	0	1	logique déclage à droite	décalage
1	1	0	arithmétique A+B	arithmétique
1	1	1	arithmétique A-B	arithmétique

Ce tableau résulte du **choix** du concepteur

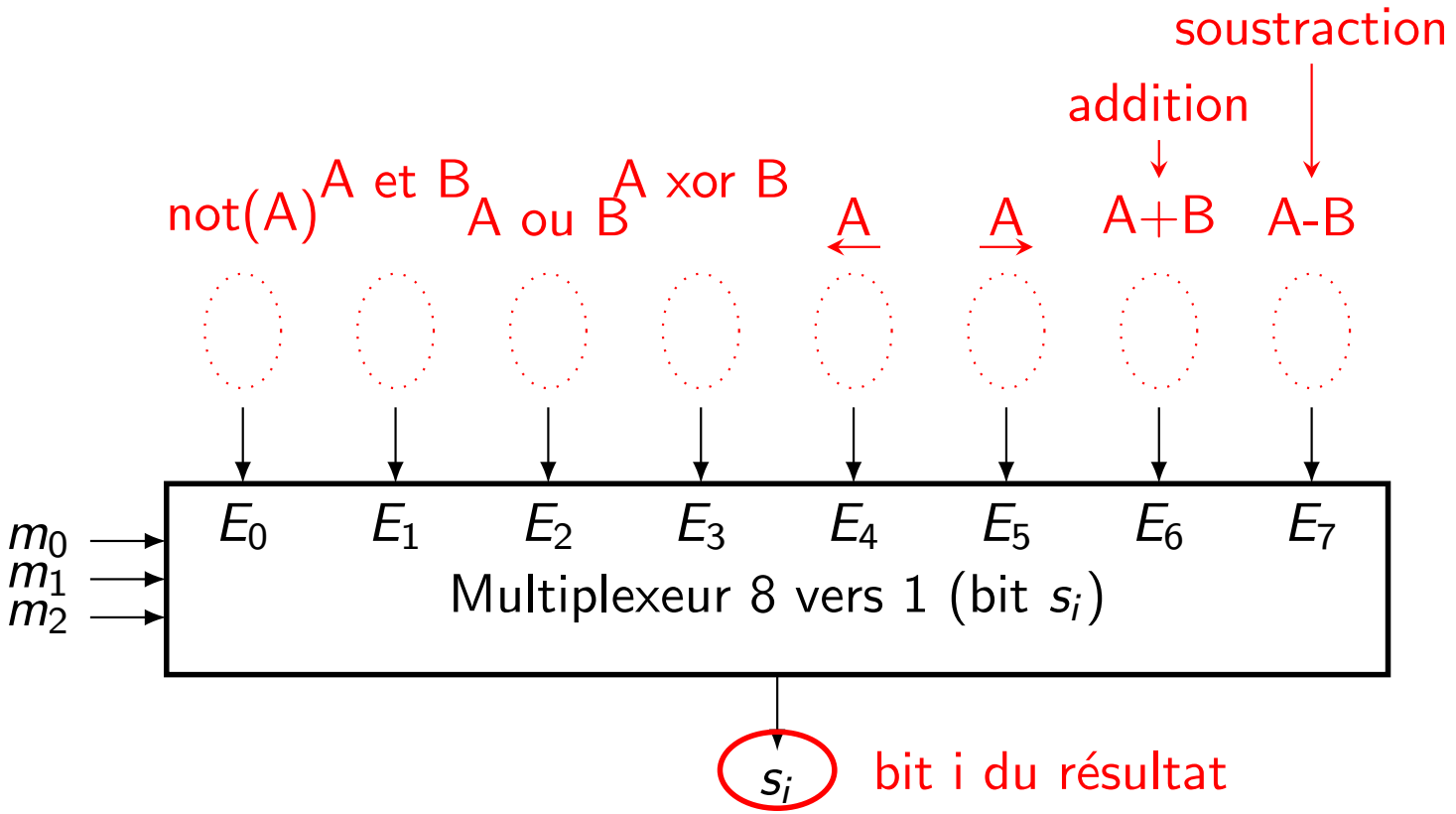
Flags de l'UAL

- Z (Zero) vaut 1 quand le résultat $S = 0$ (tous les bits sont nuls)
- N (Negative) vaut 1 quand le résultat S est un nombre négatif
- C (Carry) vaut 1 quand un débordement a lieu sur S
- V (oVerflow) vaut 1 si dépassement de capacité pour S

Z et N permettent des instructions conditionnelles comme :
`if (x==0) en c`
C et V protègent contre les erreurs de calcul

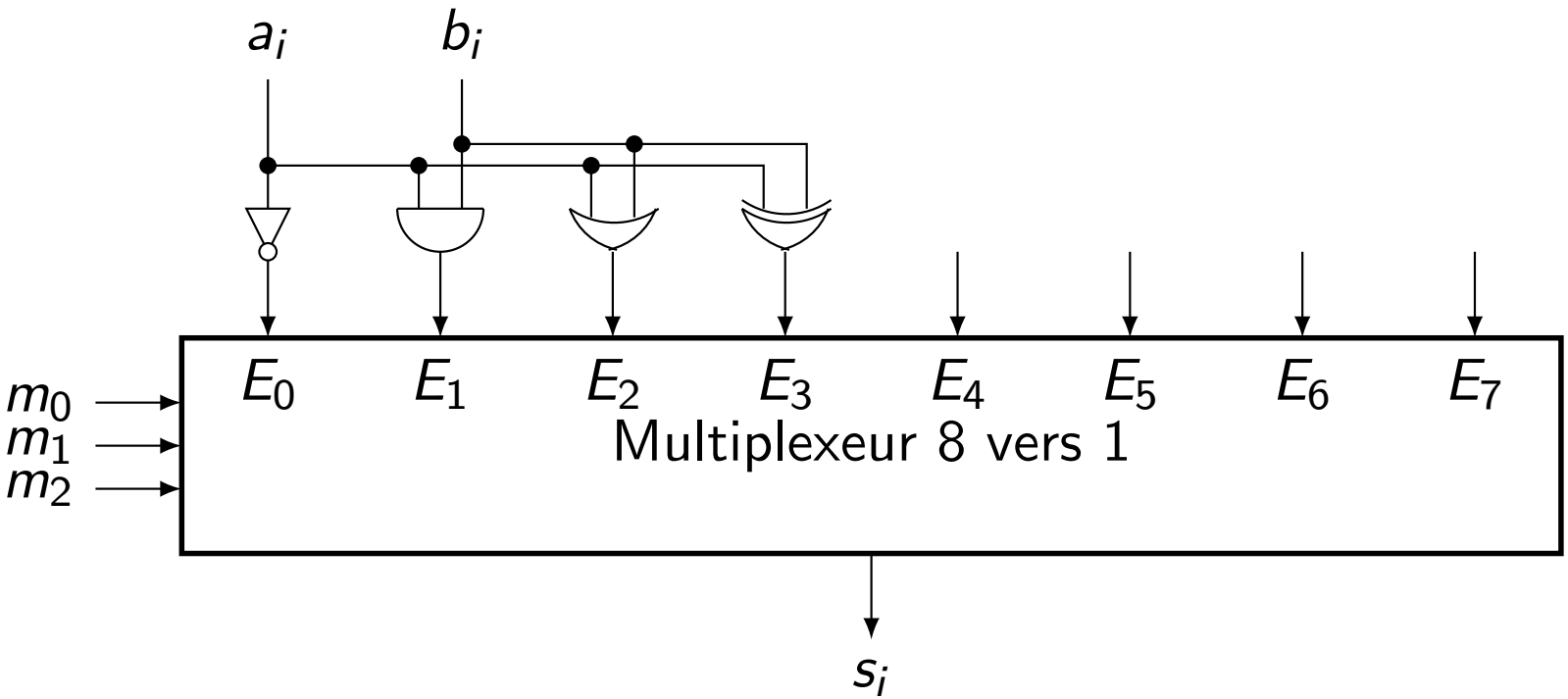
Constitution générale de l'UAL

Basée sur un multiplexeur 8 vers 1 à n bits $\equiv$ n mpx 8 $\rightarrow$ 1 à 1 bit



$$S_i = \sum_{i=0}^{7} E_i.(M = i) = \overline{m_2}.\overline{m_1}.\overline{m_0}.E_0 + \ldots + m_2.m_1.m_0.E_7$$

Constitution de la partie logique de l'UAL

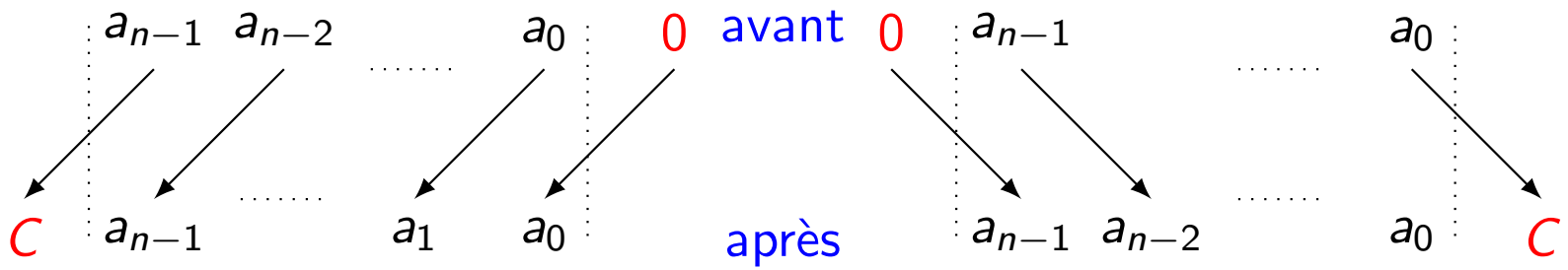


$$S_i = \overline{m_2}.\overline{m_1}.\overline{m_0}.a_i + \overline{m_2}.\overline{m_1}.m_0.a_i.b_i + \overline{m_2}.m_1.\overline{m_0}.(a_i + b_i) + \overline{m_2}.m_1.m_0.a_i.(a_i \oplus b_i) + \ldots$$

Partie décalage de l'UAL

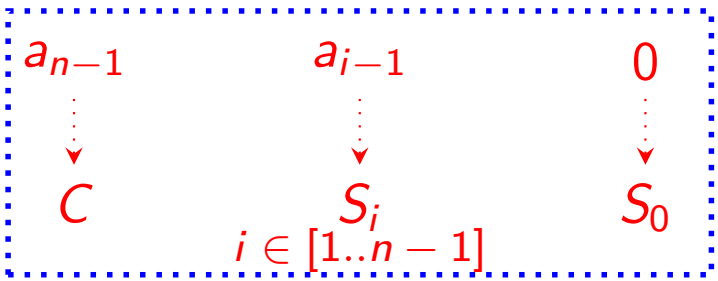
dans l'UAL - mot initial : A
- mot décalé : S

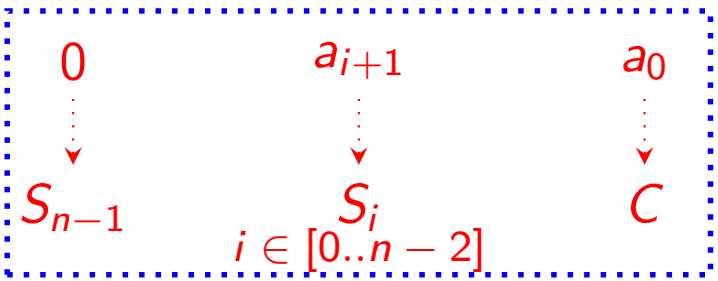
décalage de A (pour explication)



décalage à gauche

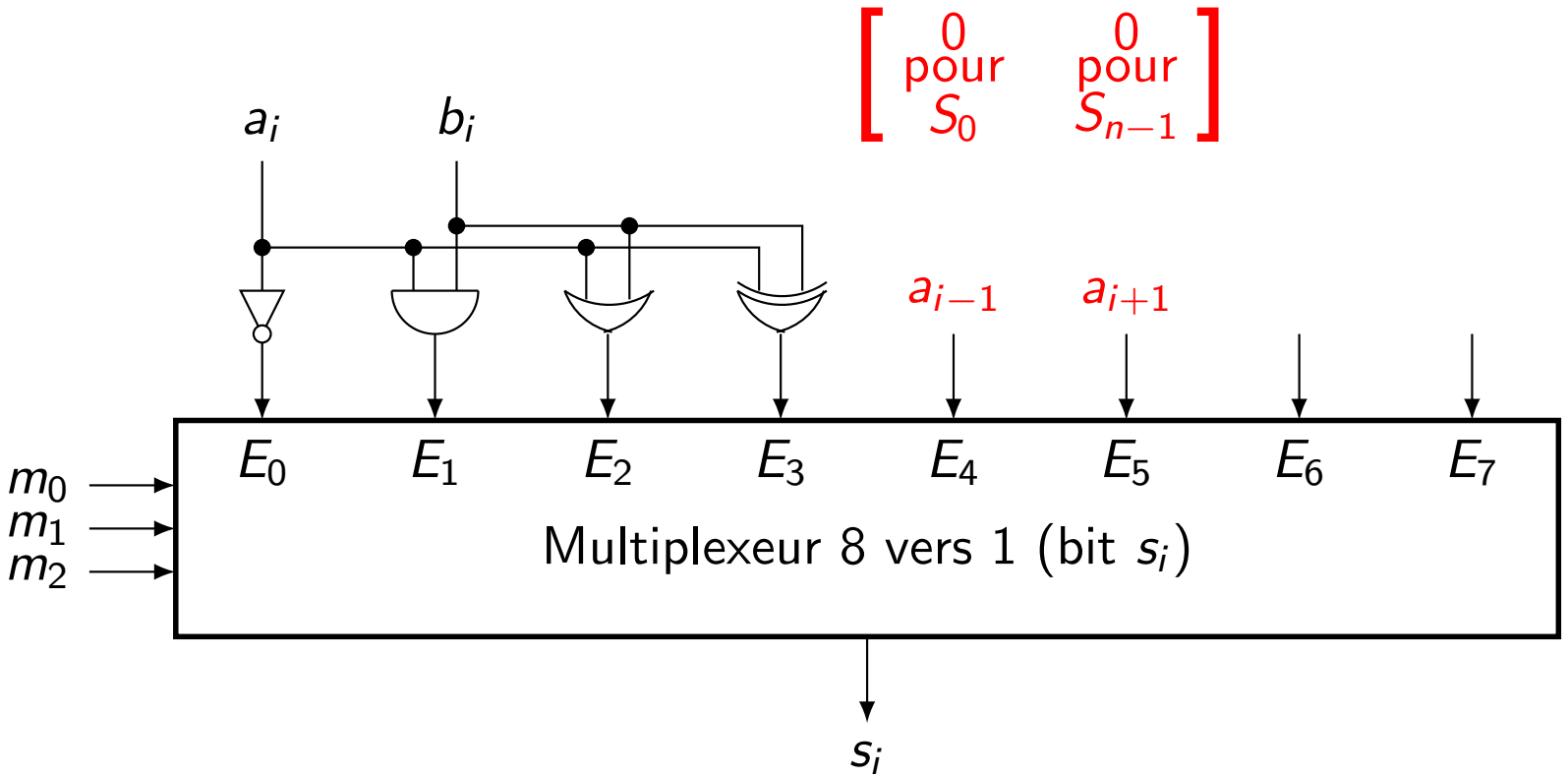
décalage à droite





Rectangles en pointillés bleus : bilan pour la suite de la synthèse

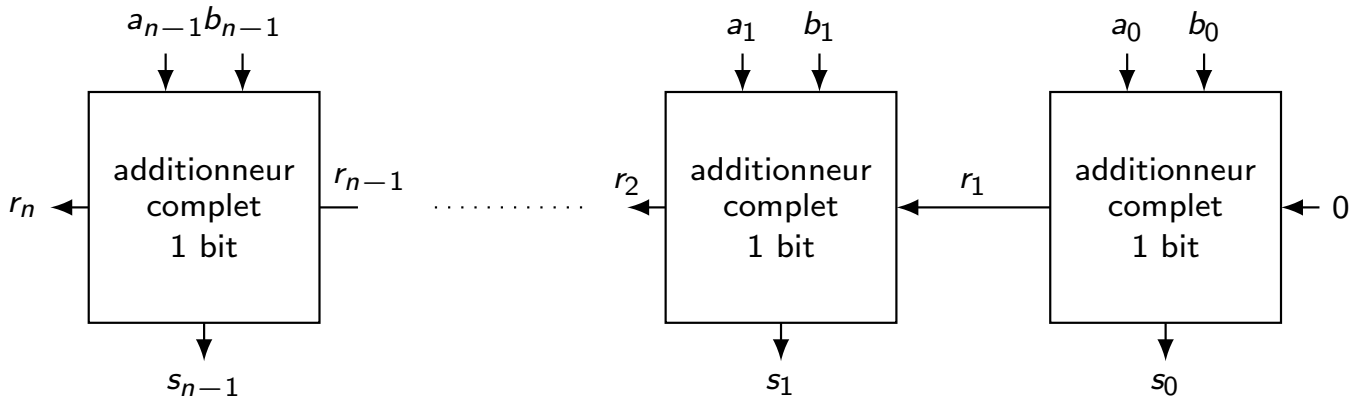
Constitution de la partie logique (complète)



$$S_0 = (E_0 \ldots E_3) + m_2.\overline{m_1}.\overline{m_0}.0 + m_2.\overline{m_1}.m_0.a_1 + (E_6, E_7)$$
$$S_i = (E_0 \ldots E_3) + m_2.\overline{m_1}.\overline{m_0}.a_{i-1} + m_2.\overline{m_1}.m_0.a_{i+1} + (E_6, E_7)$$
$$S_{n-1} = (E_0 \ldots E_3) + m_2.\overline{m_1}.\overline{m_0}.a_{n-2} + m_2.\overline{m_1}.m_0.0 + (E_6, E_7)$$

Addition binaire de n bits

Elle repose sur la mise en parallèle de n additionneurs complets 1 bit



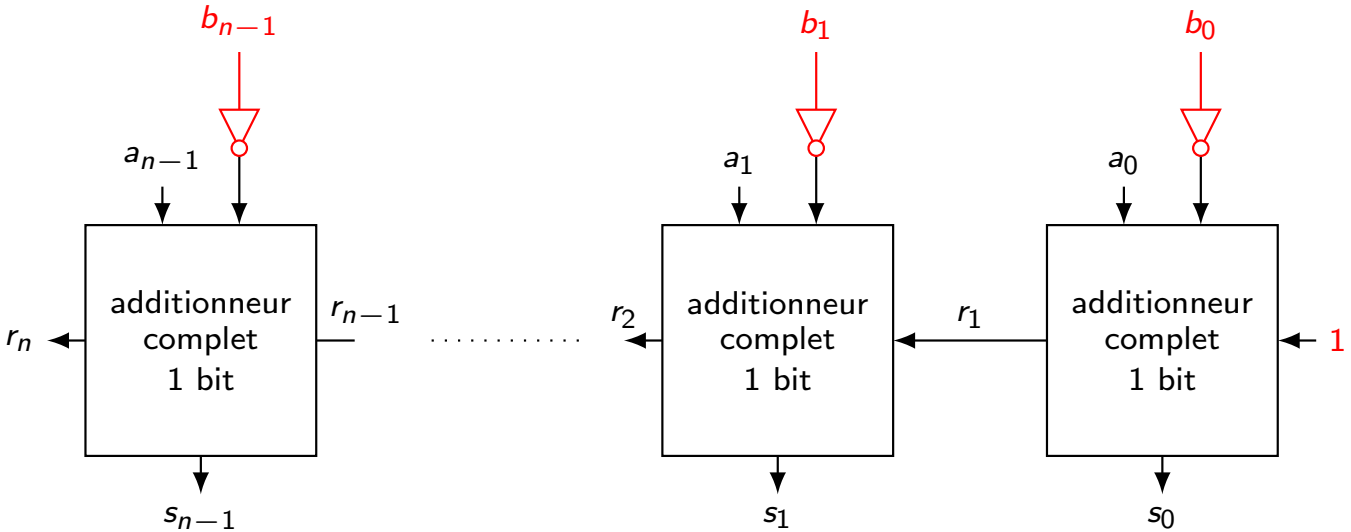
Additionneur conçu pour les entiers positifs

On a vu (Video du cours) que :

1. $B = |B|$ codé par $|B| \Rightarrow -B$ est codé par $2^n - |B|$
2. $B = -|B|$ codé par $2^n - |B| \Rightarrow -B$ est codé par $2^n - (2^n - |B|) = |B|$
3. Ainsi, prendre le Ca2 de (x) c'est coder $-x$
4. Permet de transformer $A - B$ en $A + Ca2(B)$

Calcul du complément à deux avec l'additionneur complet

Rappel : $Ca2(x) = \overline{Cb(x)} + 1$



Ainsi câblé, l'additionneur réalise en fait le calcul $A + \overline{Cb(B)} + 1$ c'est à dire l'addition $A + Ca2(B)$ soit encore la soustraction $A - B$.

Il ne reste qu'à concevoir un circuit appliquant

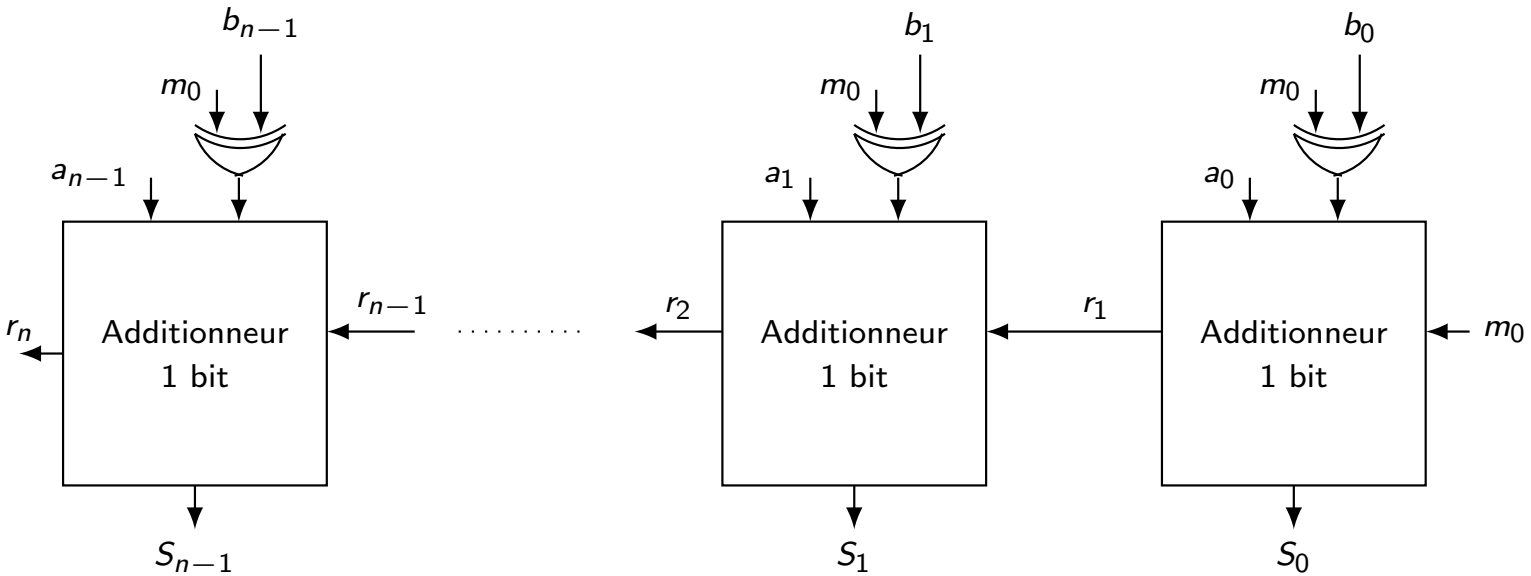
1. le complément à l'opérande B sur les entrées du multiplexeurs
2. la valeur 1 sur l'entrée r_0

uniquement lorsque l'opération effectuée est la soustraction.

Calcul du complément à deux avec l'additionneur complet

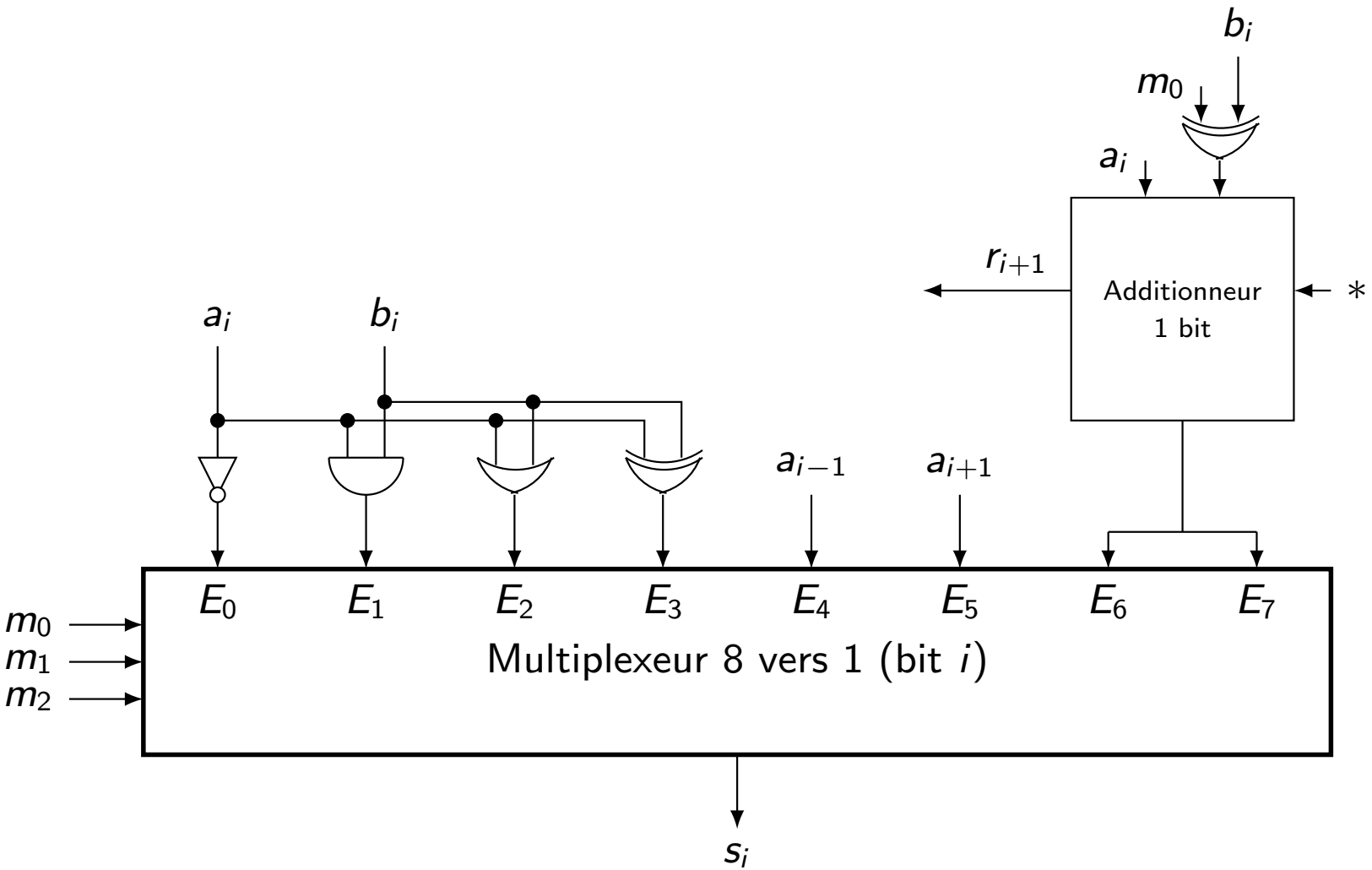
Rappel : $Ca2(x) = \overline{Cb(x)} + 1$

$m_0 = 0 \left\| \begin{array}{l} b_i \\ r_0 = 0 \end{array} \right.$ $m_0 = 1 \left\| \begin{array}{l} \overline{b_i} \\ r_0 = 1 \end{array} \right.$ et $\begin{array}{l} m_0 = 0 \text{ (addition)} \\ m_0 = 1 \text{ (soustraction)} \end{array}$



Ainsi câblé le circuit réalise addition ou soustraction en fonction de m_0

UAL 1 bit complète (sans Flags)



Questions de mise en œuvre

- Nous avons fait l'étude de chaque sous fonction de cette UAL
  1. additionneur complet n bits
  2. multiplexeur 8 vers 1 (Pr Sainrat)
  3. opérateurs logiques : NON, ET, OU, XOR (Pr Sainrat)
  4. décalage droite et gauche
- Il est donc possible de faire simplifier les différentes expressions à un outil de conception (pas à la main !)
- Rien ne s'oppose à câbler cette UAL dans un ciruit logique programmable (FPGA)
- Utilisation du langage VHDL (hors programme ici !)

Manque la définition des flags Z, N, V, C

Synthèse des indicateurs (Flags) : Z(ero)

- $Z = 1$ quand $S = 0$ (tous les bits à 0)

$$Z = \overline{S_{n-1}}.\overline{S_{n-2}} \ldots \overline{S_0} \quad \text{(minterme)}$$

$\Rightarrow$ très simple à réaliser

Remarque : indicateur valide quelle que soit l'opération réalisée par l'UAL (sa valeur ne dépend pas de m_2, m_1 ou m_0)

Par exemple, on peut tester $S = 0$ après l'opération "A *et* B"

Synthèse des indicateurs (Flags) : N(egative)

- $N = 1$ quand $s_{n-1} = 1$
  Uniquement quand l'UAL manipule des codes en en Ca2

C'est le cas pour l'addition et la soustraction
Pour les autres opérations choisissons $\rightarrow$ N=0

Rappel : en Ca2, $S < 0 \Rightarrow s_{n-1} = 1$, d'où :

m_2	m_1	m_0	N
0	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	0	1	0
1	1	0	s_{n-1}
1	1	1	s_{n-1}

$$\begin{aligned} N &= m_2.m_1.\overline{m_0}.s_{n-1} + m_2.m_1.m_0.s_{n-1} \\ &= m_2.m_1.s_{n-1} \end{aligned}$$

Synthèse des indicateurs (Flags) : C(arry)

$C = 1$ quand le résultat d'une opération déborde des n bis de S

- décalages
- $+$ et -

Pour les 4 autres opérations (logiques) notre choix $C = 0$ car il s'agit d'opérations bit à bit dans lesquelles C n'intervient jamais

Synthèse

m_2	m_1	m_0	C
0	$*$	$*$	0
1	0	0	a_{n-1}
1	0	1	a_0
1	1	0	r_n
1	1	1	r_n

d'où

$$\begin{aligned} C = \; & m_2.\overline{m_1}.\overline{m_0}.a_{n-1} \\ & + m_2.\overline{m_1}.m_0.a_0 \\ & + m_2.m_1.r_n \end{aligned}$$

Synthèse des indicateurs (Flags) : (o)V(erflow)

$V = 1$ quand un résultat est faux (opérations + et -)
$\Rightarrow$ Seules les deux opérations arithmétiques positionnent V

- $A \geq 0, B < 0 : A \in [0, 2^{n-1} - 1], b \in [-2^{n-1}, 0]$
  $A + B \in [-2^{n-1}, 2^{n-1} - 1]$
  $\Rightarrow A + B$ est toujours dans l'intervalle du Ca2 (jamais d'erreur)
- $A < 0, B < 0,\ A \in [-2^{n-1}, 0], b \in [-2^{n-1}, 0]$
  $A + B \in [-2^n, 0]$ et peut donc être hors de l'intervalle du Ca2
  Overflow si $A + B$ apparaît comme ≥ 0 ($V_1 = a_{n-1}.b_{n-1}.\overline{s_{n-1}}$)
- $A \geq 0, B \geq 0,\ A \in [0, 2^{n-1} - 1], b \in [0, 2^{n-1} - 1]$
  $A + B \in [0, 2^n - 2]$ et peut donc être hors de l'intervalle du Ca2
  Overflow si $A + B$ apparaît comme < 0 ($V_2 = \overline{a_{n-1}}.\overline{b_{n-1}}.s_{n-1}$)

D'où finalement : $V = V_1 + V_2 = a_{n-1}.b_{n-1}.\overline{s_{n-1}} + \overline{a_{n-1}}.\overline{b_{n-1}}.s_{n-1}$