Unités arithmétiques et logiques

Polytech Nice Sophia Antipolis

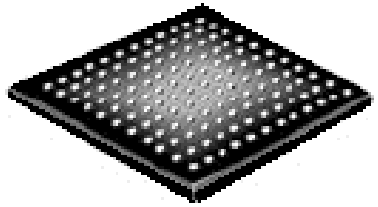
Circuits intégrés

Les types de boîtiers

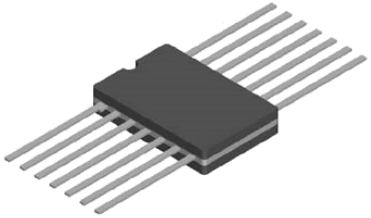
- DIP
- PGA
- Flatpack
- LCC
- QFP
- SOIC

DIP

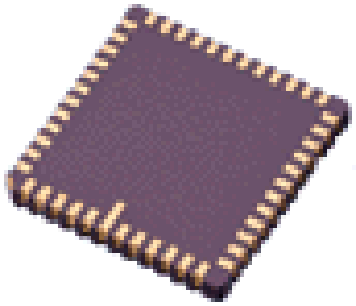
QFP



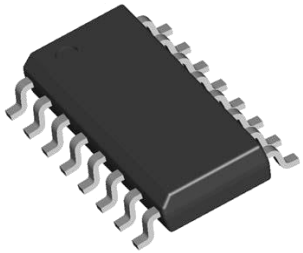
PGA



flatpack



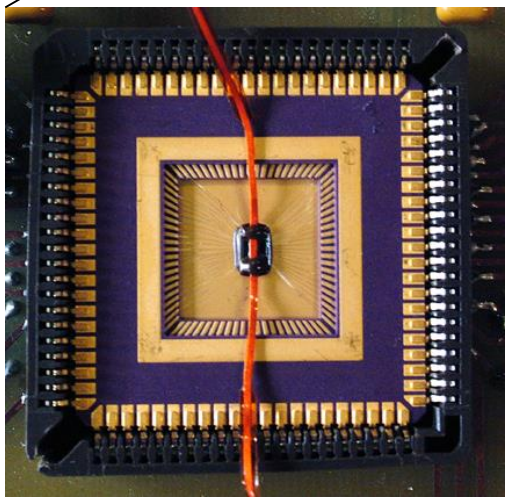
LCC



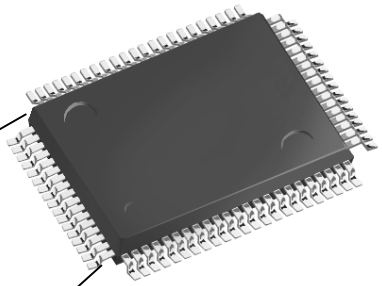
SOIC

Les types de boîtiers

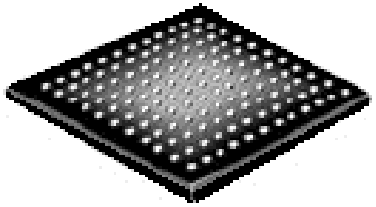
- DIP
- BGA
- Flatpack
- LCC
- QFP
- SOIC



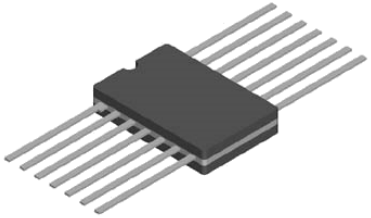
DIP



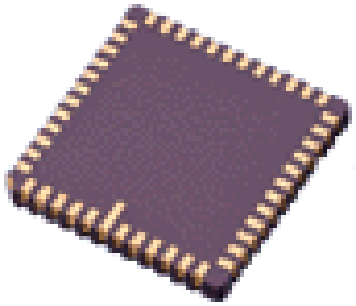
QFP



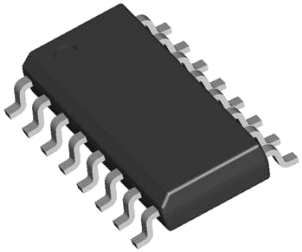
BGA



flatpack



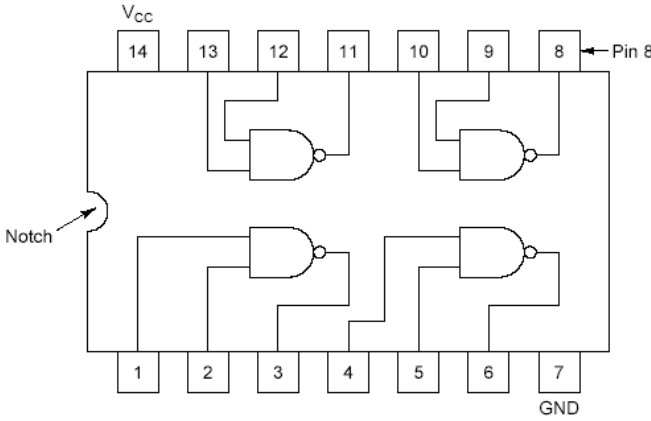
LCC



SOIC

Densité d'intégration

- SSI – Small Scale Integration (1960's)
  - 1 à 10 portes / circuit
  - Moins de 100 transistors
- MSI – Medium Scale Integration
  - 10 à 100 portes / circuit
  - Plus de 3000 transistors
- LSI – Large Scale Integration (1970's)
  - 100 à 10 000 portes / circuit
  - Plusieurs dizaines de milliers de transistors
- VLSI – Very Large Scale Integration (1980's)
  - + de 10 000 portes / circuit
  - 1 million de transistors
- ULSI – Ultra Large Scale Integration (2000's)
  - + de 100 000 porte / circuit



Les circuits logiques combinatoires

Le multiplexeur

- Un multiplexeur dispose de 2^n entrées, d'une sortie et de n lignes de sélection.
- Les sélecteurs permettent de choisir 1 entrée parmi les 2^n et de la router vers la sortie
- MUX_1_2^n, 1 parmi 2^n

- Comment faire ce circuit ?

Le multiplexeur

On commence par quoi ?

- Le circuit
- La table de vérité
- La forme algébrique
- La forme temporelle

Dans l'ordre

1.
2.
3.
4.

Le multiplexeur

On commence par quoi ?

- Le circuit
- La table de vérité
- La forme algébrique
- La forme temporelle

Dans l'ordre

1. La table de vérité
2. La forme algébrique
3. Le circuit
4. La forme temporelle

Le multiplexeur à deux entrées MUX_2_1

- Choix d'une topologie
  - 0 -> e0
  - 1 -> e1

sel	e_1	e_0	s
0	0	0	
0	1	0	
0	0	1	
0	1	1	
1	0	0	
1	1	0	
1	0	1	
1	1	1	

Le multiplexeur à deux entrées MUX_2_1

- Choix d'une topologie
  - 0 -> e0
  - 1 -> e1

$$s = \overline{sel}.e_0.\overline{e_1} + \overline{sel}.e_0.e_1 + sel.\overline{e_0}.e_1 + sel.e_0.e_1$$

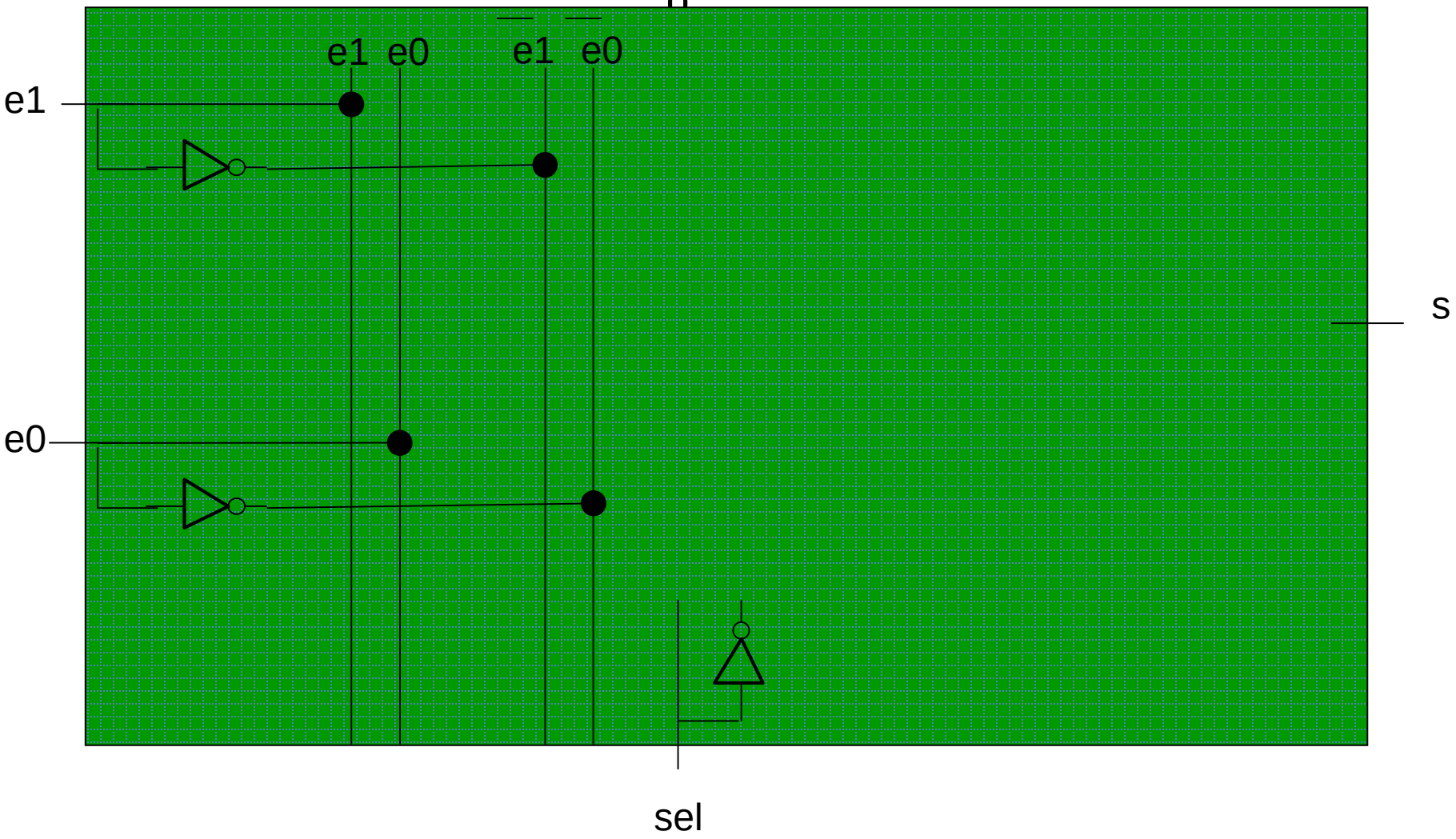
$$s = \overline{sel}.e_0(\overline{e_1} + e_1) + sel.e1(\overline{e_0} + e_0)$$

$$s = \overline{sel}.e_0 + sel.e1$$

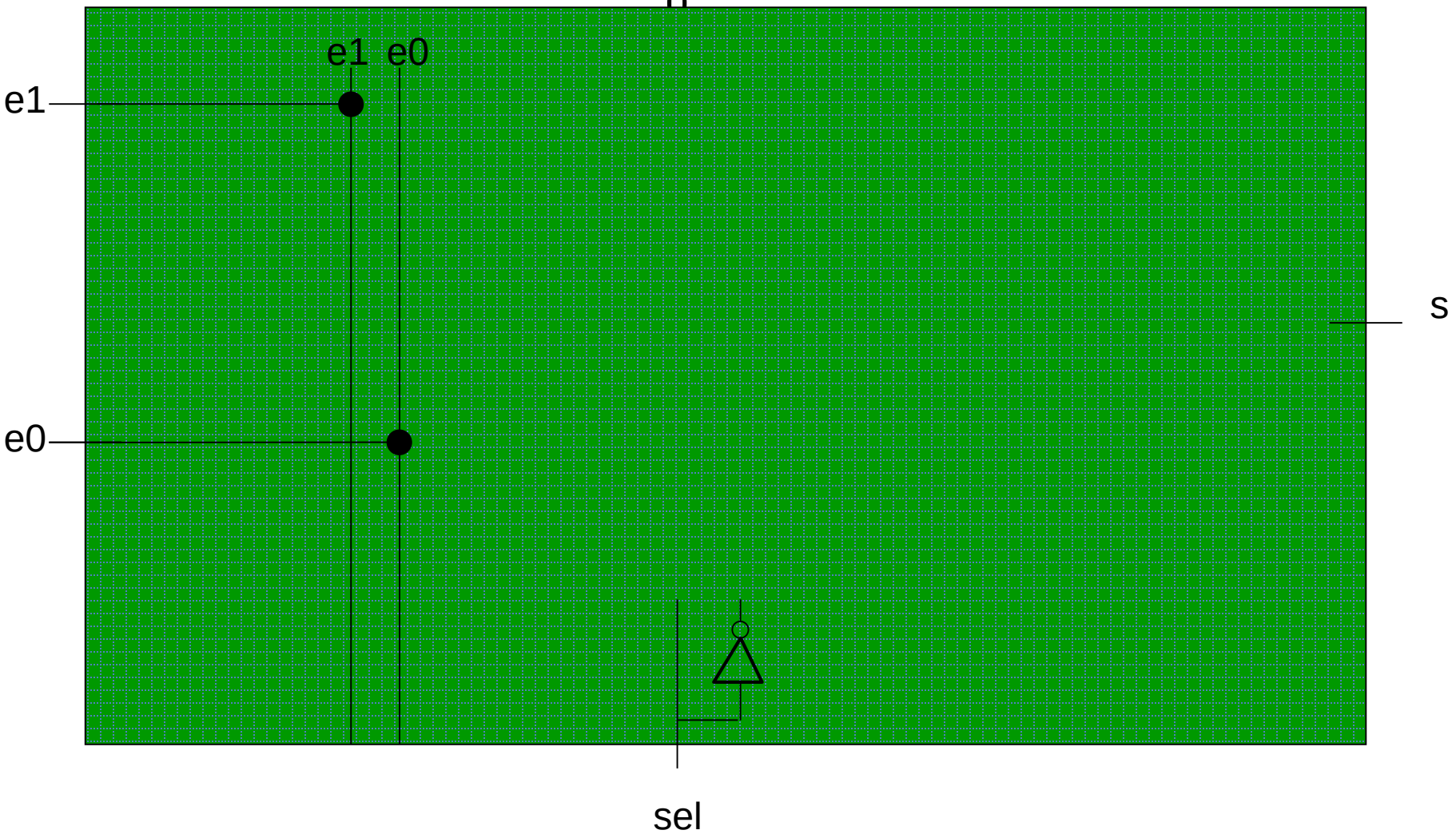
sel	e_1	e_0	s
0	0	0	0
0	1	1	1
0	0	0	0
0	1	1	1
1	0	0	0
1	1	1	1
1	0	0	0
1	1	1	1

Implantation matérielle de

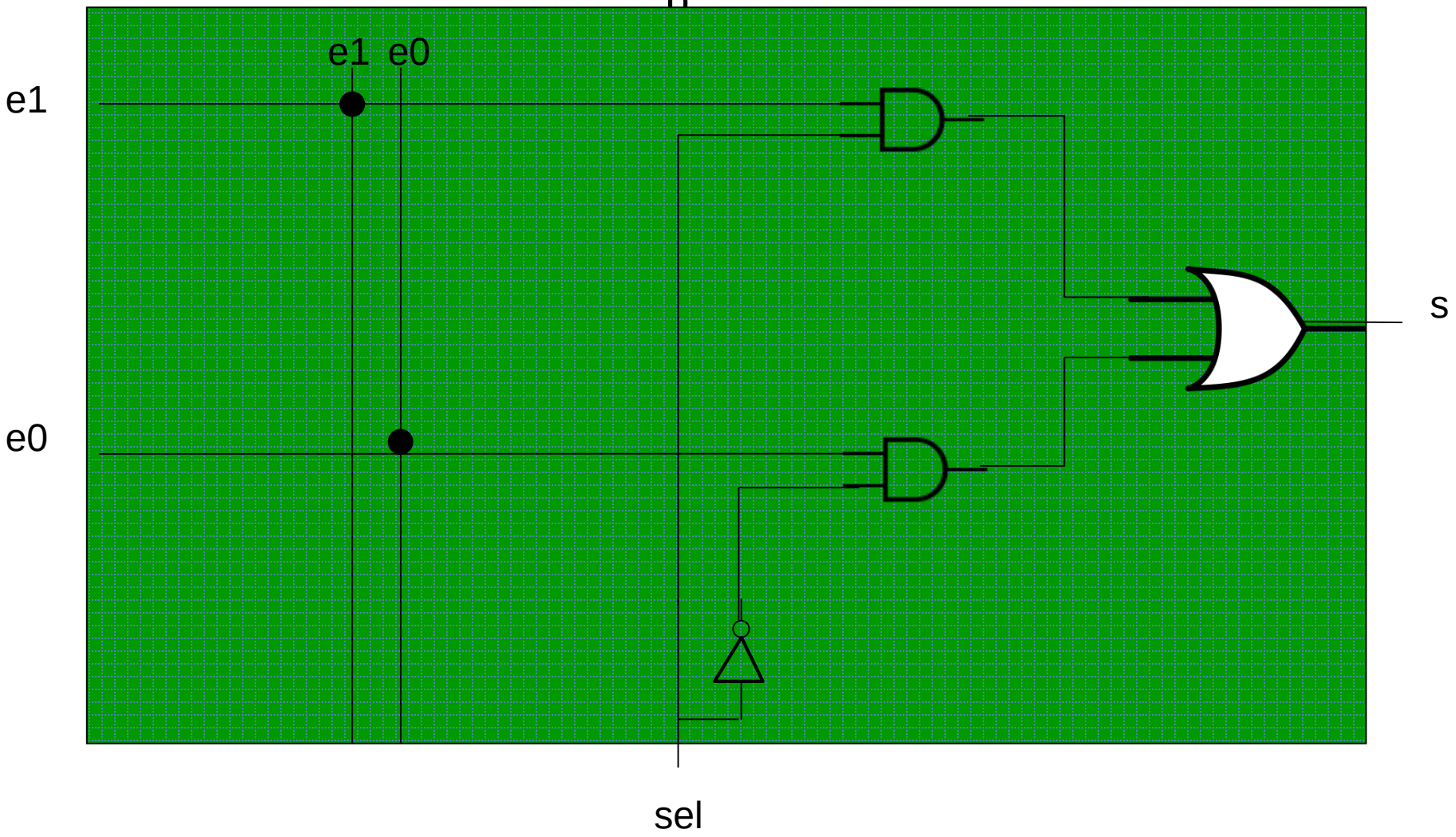
$$s = \overline{sel}.e_0 + sel.e1$$



Implantation matérielle de $s = sel.e_0 + sel.e1$

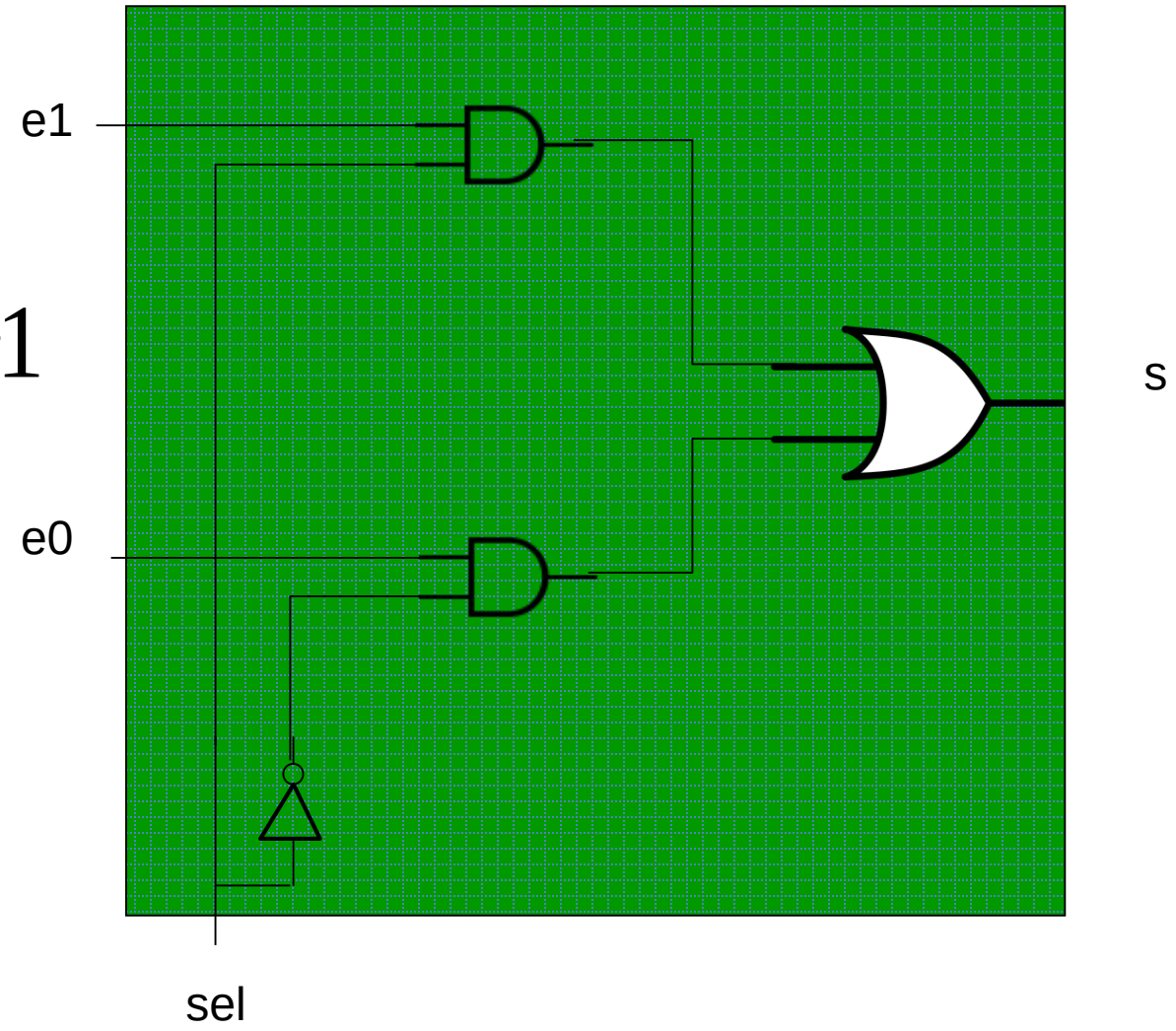


Implantation matérielle de $s = sel.e_0 + sel.e1$

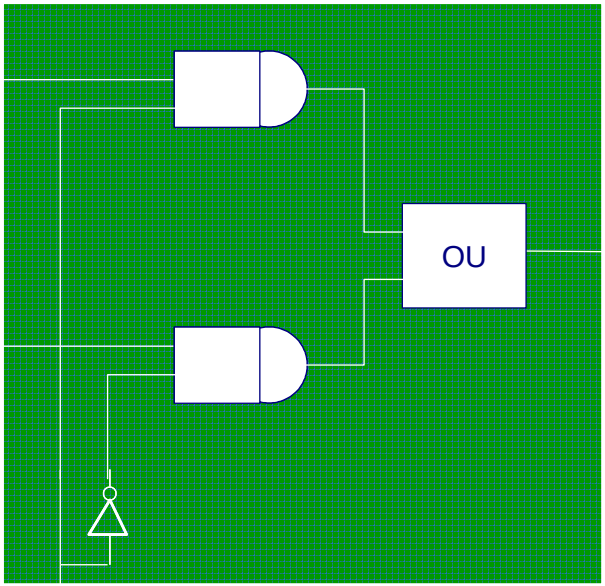


Implantation matérielle

$$s = \overline{sel}.e_0 + sel.e1$$

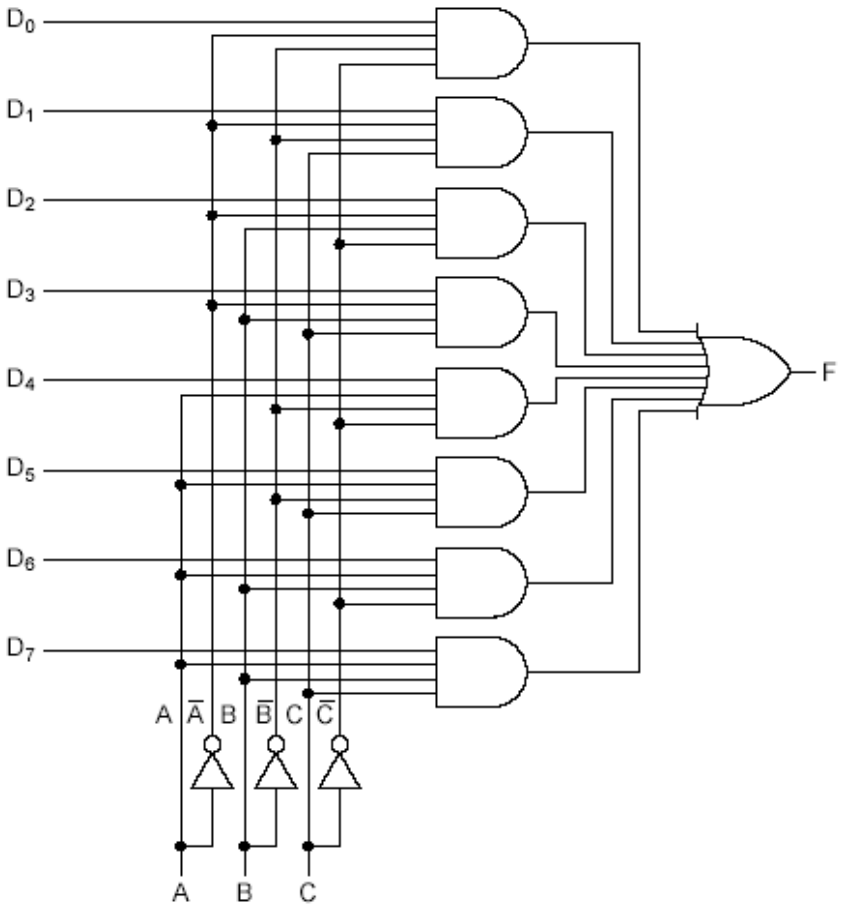


Le multiplexeur à deux entrées MUX_2_1

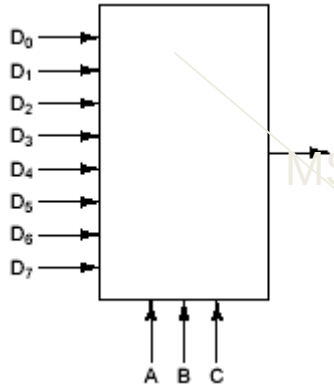


sel	e_0	e_1	s
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Le multiplexeur 8 vers 1

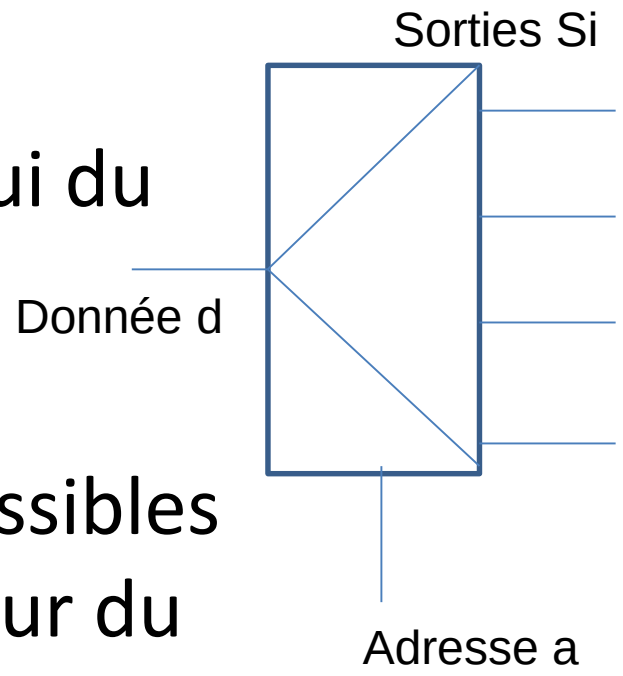


- 8 entrées = 2^3
- donc 3 sélecteurs



Le démultiplexeur

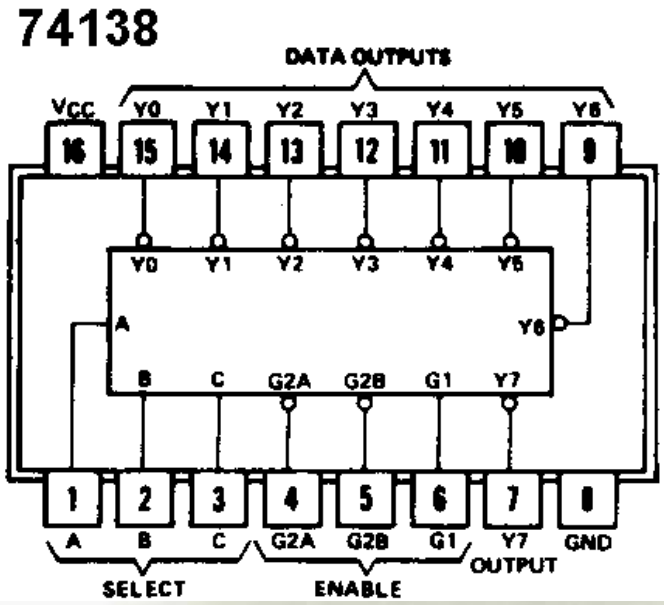
- Comme son nom l'indique, son comportement est inverse à celui du multiplexeur.
- Il autorise le routage d'un signal d'entrée vers une des sorties possibles du demux en fonction de la valeur du signal d'adresse, aussi appelé le sélecteur.

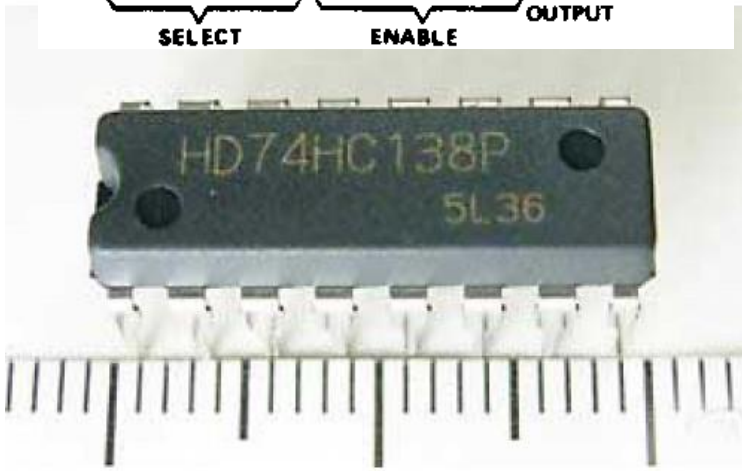


$$\begin{cases} S_0 = \bar{a}_1 \overline{a_0} d \\ S_3 = a_1 a_0 d \end{cases}$$

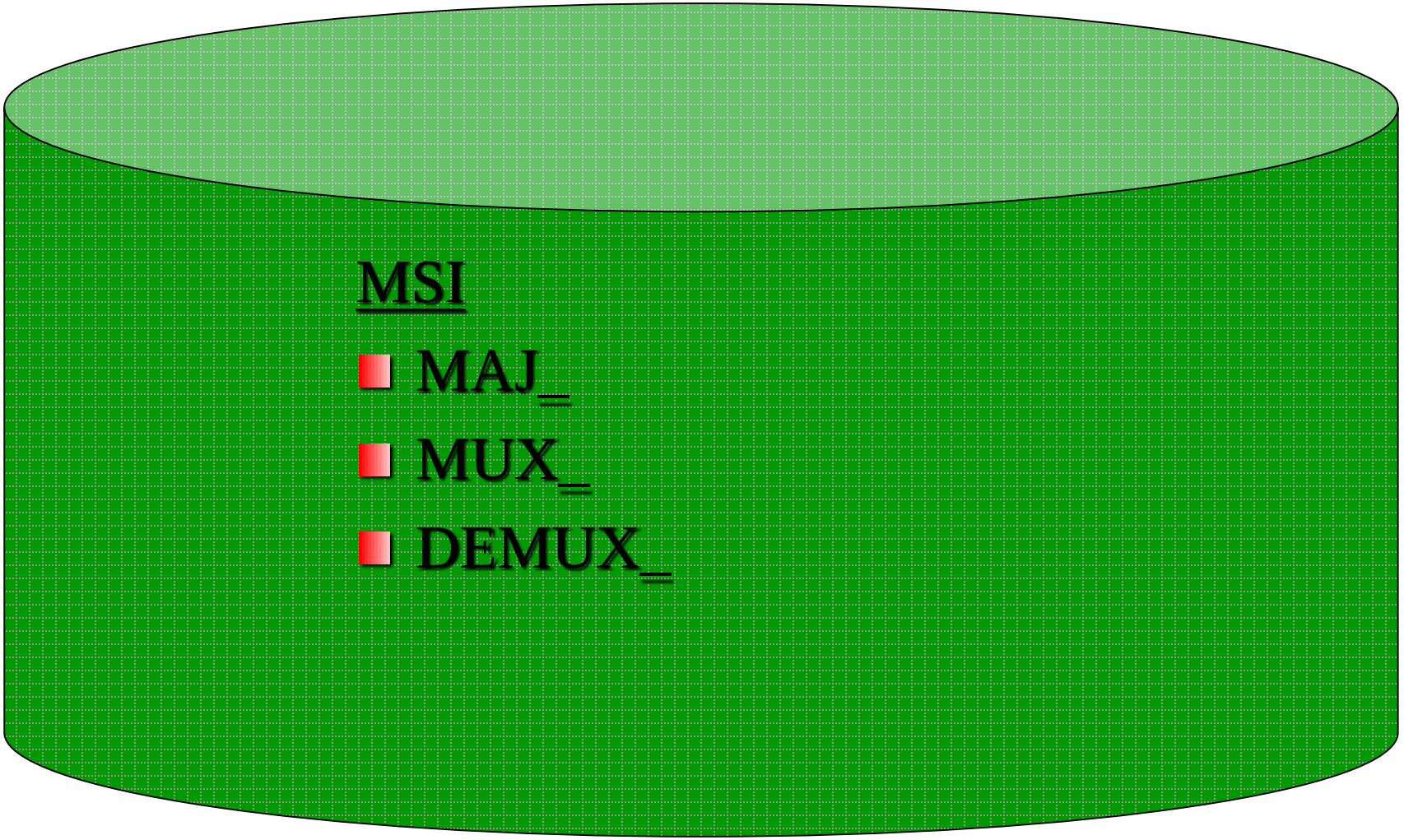
7400 Integrated Circuits

- On retrouve par exemple ce type de fonctions logiques dans les circuits de série 7400 de Texas Instrument.
- Circuits TTL : Transistor-Transistor Logic
- Exemples :
  - 74138 = demux 1:8
  - 74151A = mux 8:1
  - …





Notre bibliothèque de portes



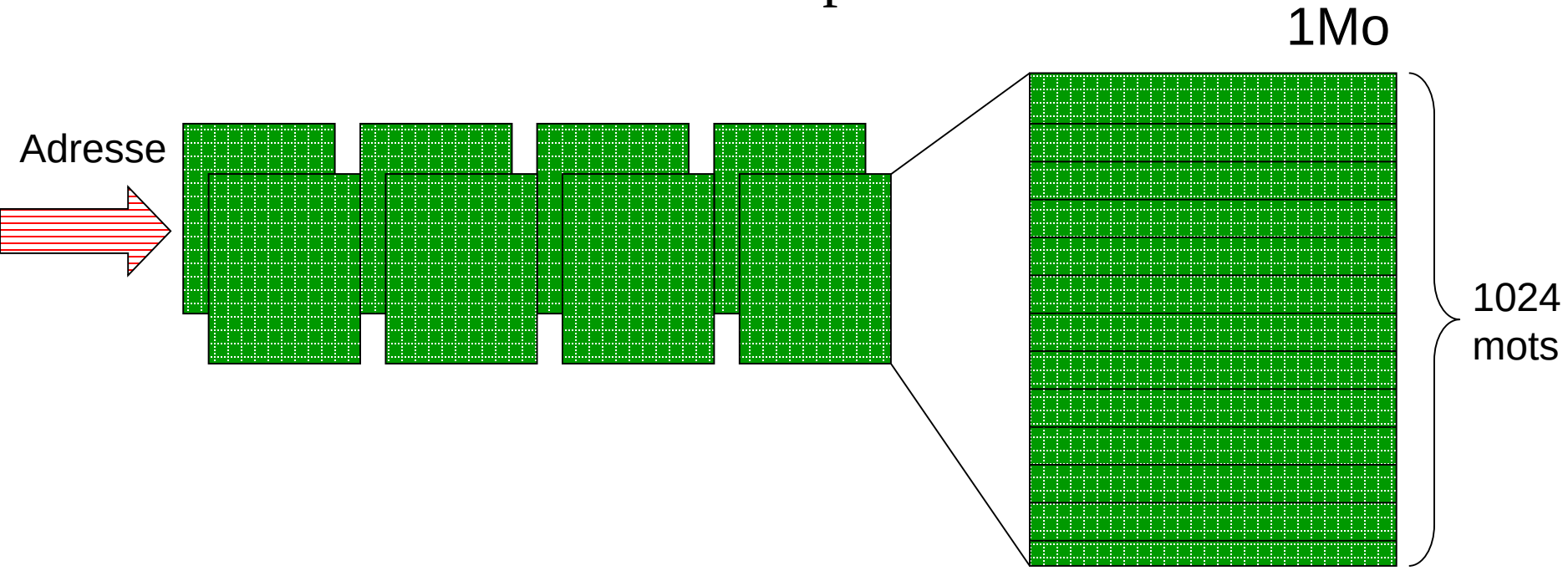
Le décodeur

- Il traduit l'information binaire en entrée pour rendre active la sortie dont le numéro correspond
- Par exemple un circuit mémoire utilise un décodeur pour aller chercher l'information présente à l'adresse n parmi ses 2^n emplacements

Le décodeur

Mémoire de 8 Moctets

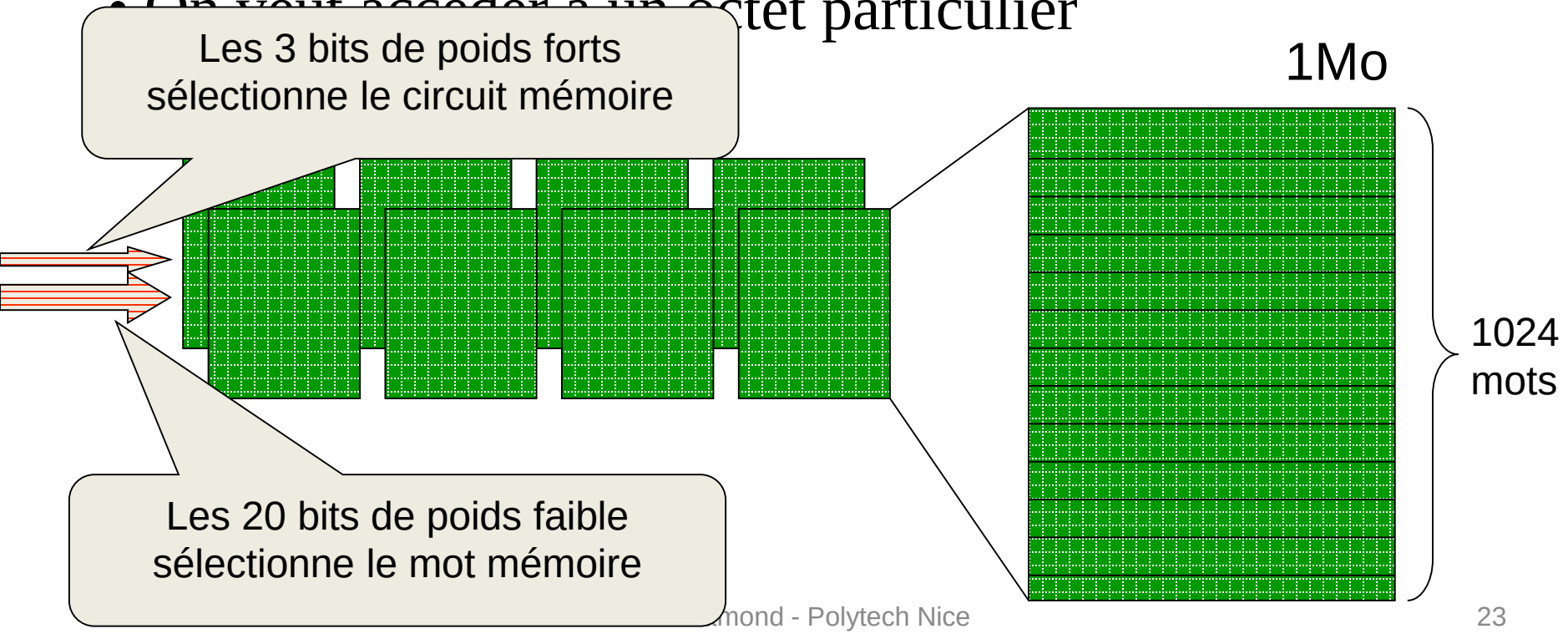
- composée de 8 circuits mémoires
- Chaque circuit contient 1 Moctet
- On veut accéder à un octet particulier



Le décodeur

Mémoire de 8 Moctets

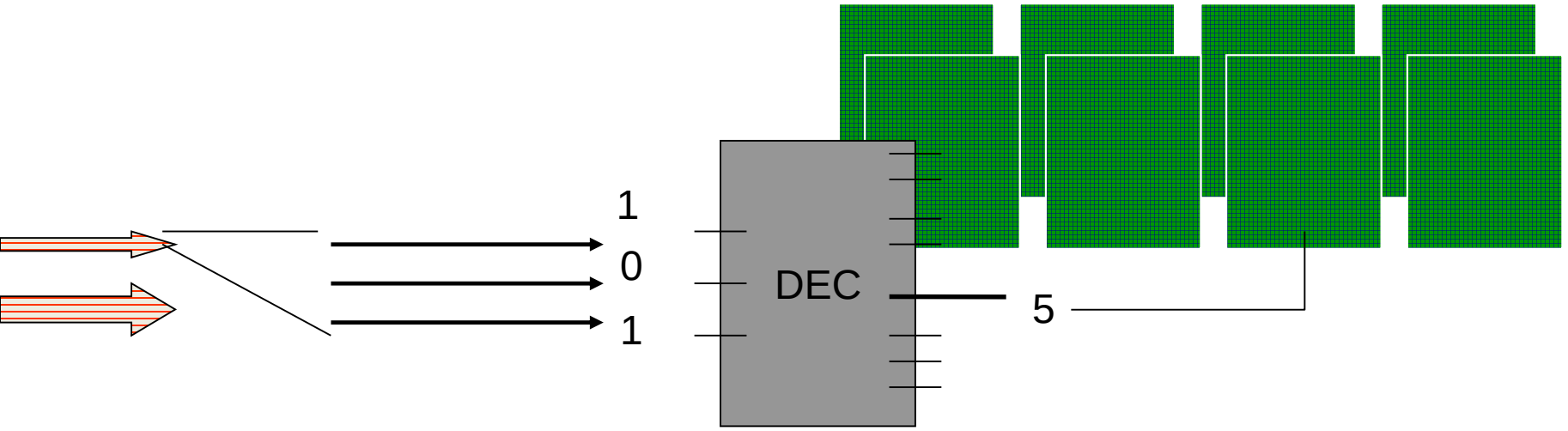
- composée de 8 circuits mémoires
- Chaque circuit contient 1 Moctet
- On veut accéder à un octet particulier



Le décodeur

- *Entrée = code 101*
- *Sortie = activer la sortie 5*

  Mettre les autres sorties à 0



Le décodeur

e_2	e_1	e_0	s_0	…	s_7
0	0	0	1		0
0	0	1	0		0
0	1	0	0		0
0	1	1	0		0
1	0	0	0		0
1	0	1	0		0
1	1	0	0		0
1	1	1	0		1

$$s_0 = \overline{e_2}\,\overline{e_1}\,\overline{e_0}$$

$$s_1 = \overline{e_2}\,\overline{e_1}\,e_0$$

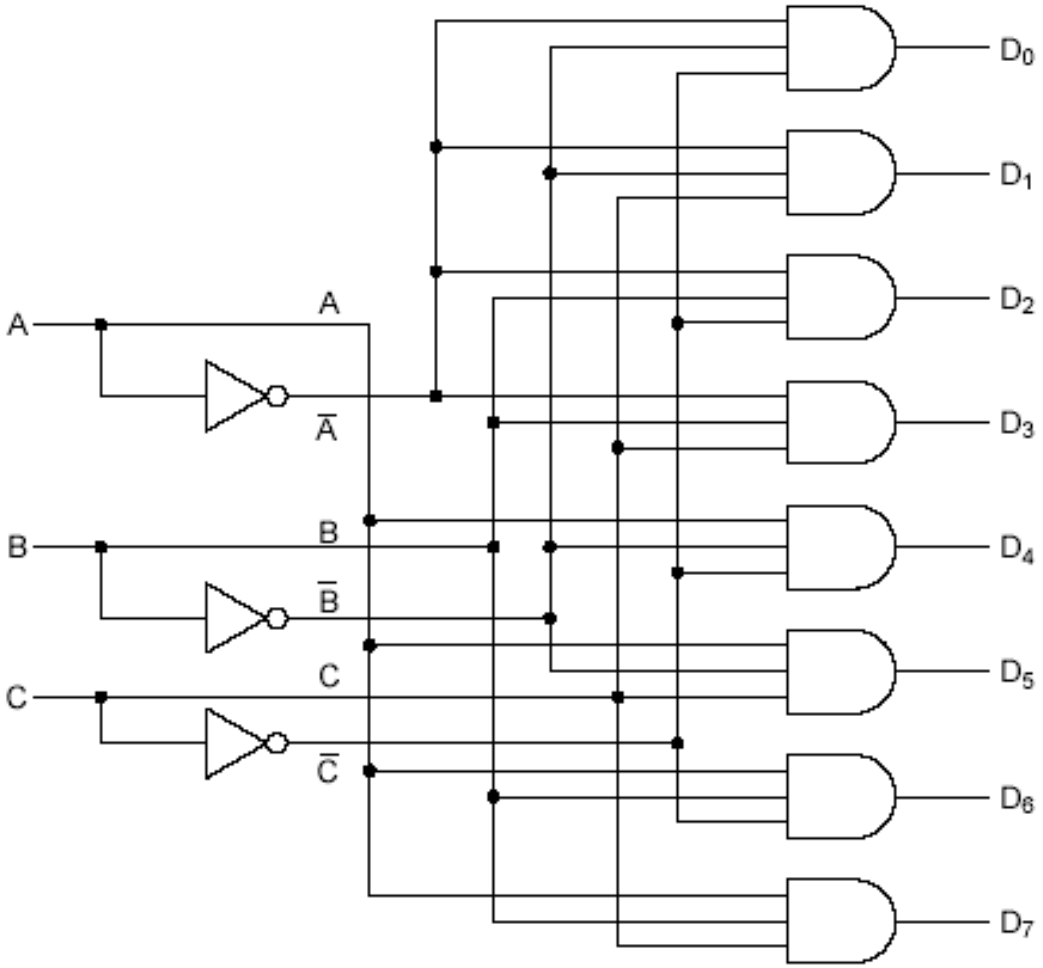
$$s_2 = \overline{e_2}\,e_1\,\overline{e_0}$$

…

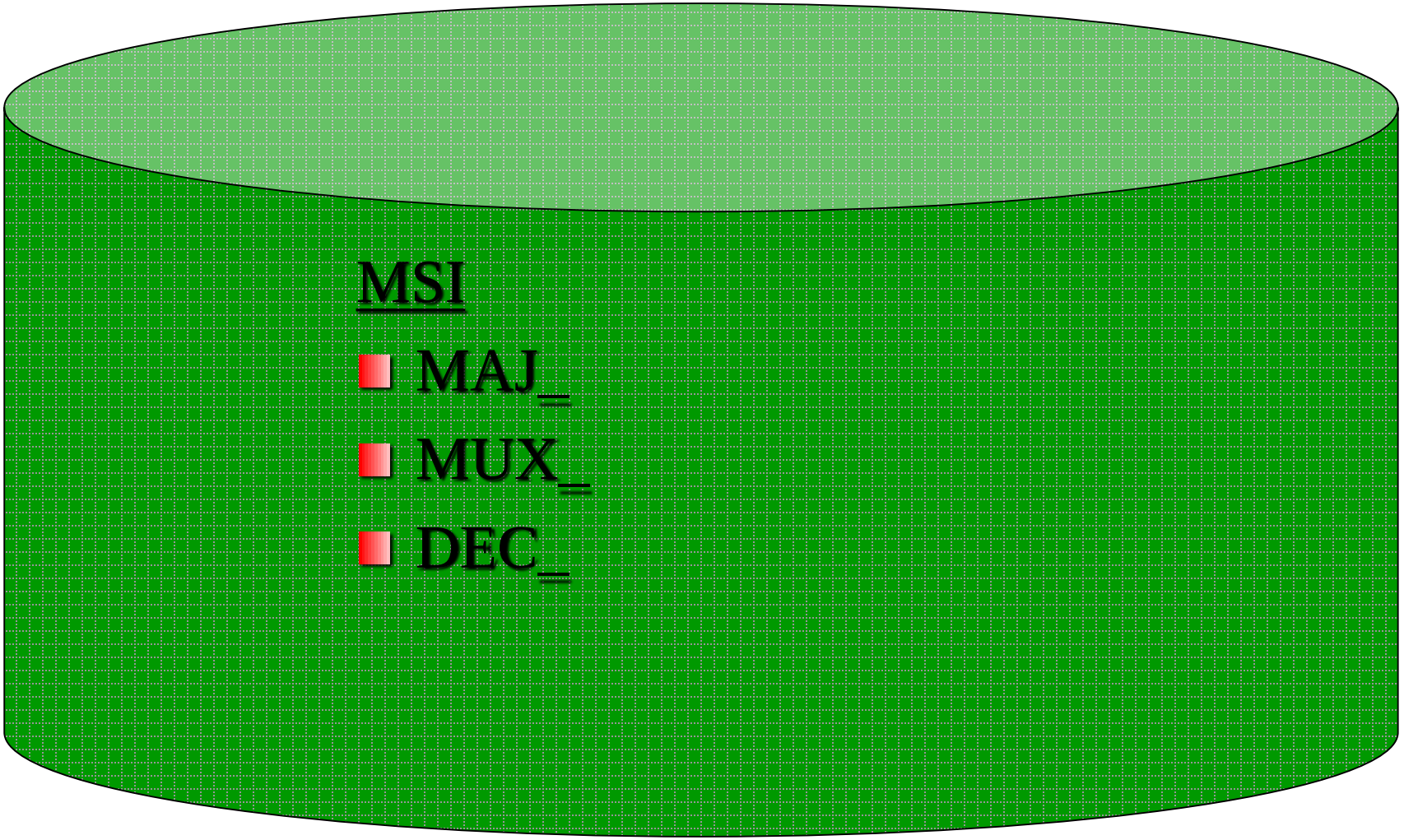
$$s_7 = e_2 e_1 e_0$$

Le décodeur

…

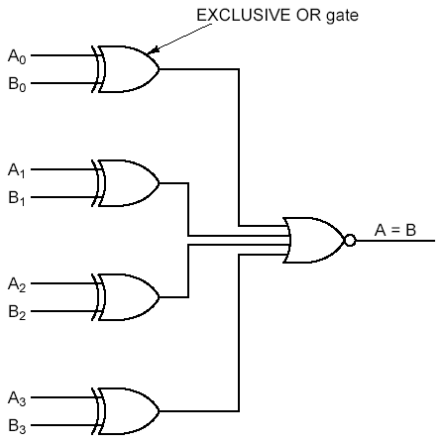


Notre bibliothèque de portes



Le comparateur

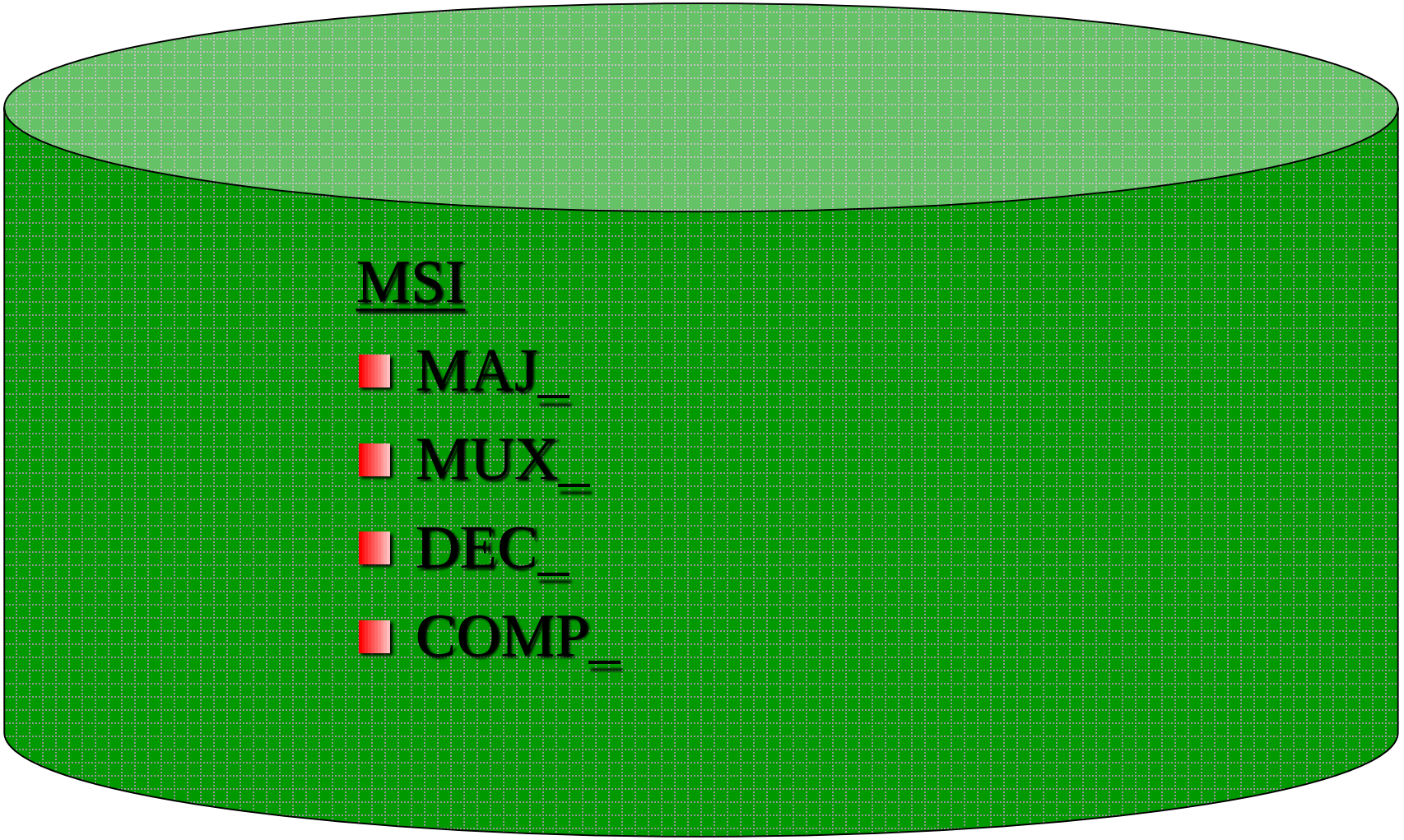
- Il effectue la comparaison de 2 mots de *n* bits A et B



- Renvoie
  - 1 si A = B,
  - 0 sinon

$$s = \overline{a_0 \oplus b_0 + a_1 \oplus b_1 + a_2 \oplus b_2 + a_3 \oplus b_3}$$

Notre bibliothèque de portes

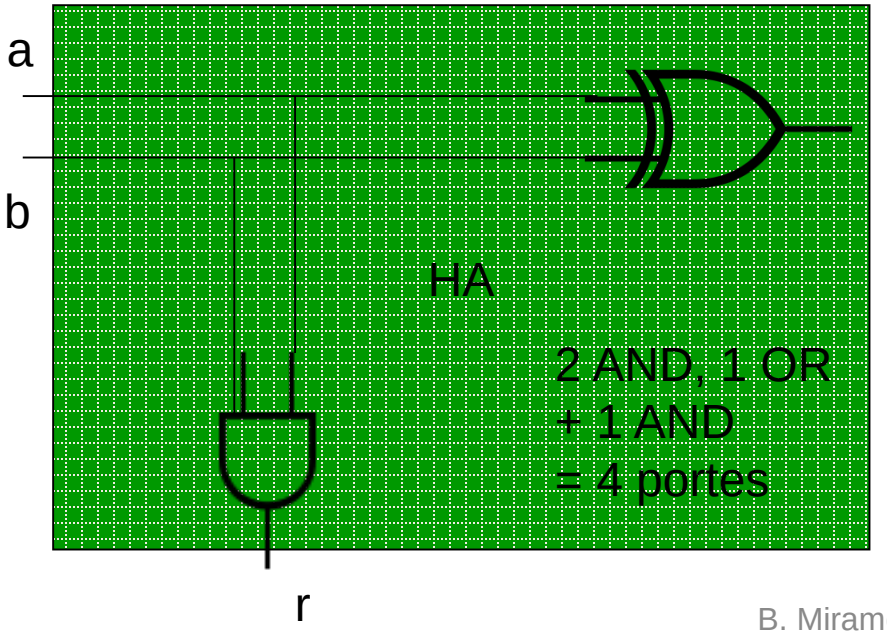


Les circuits arithmétiques

Addition & demi-additionneur (Half-Adder)

- sans retenue entrante
- s(omme) = a $\oplus$ b

- r(etenue) = a.b



a	b	s	r
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Additionneur complet

$$s = \bar{a}\bar{b}R_e + \bar{a}b\bar{R}_e + a\bar{b}\bar{R}_e + abR_e$$

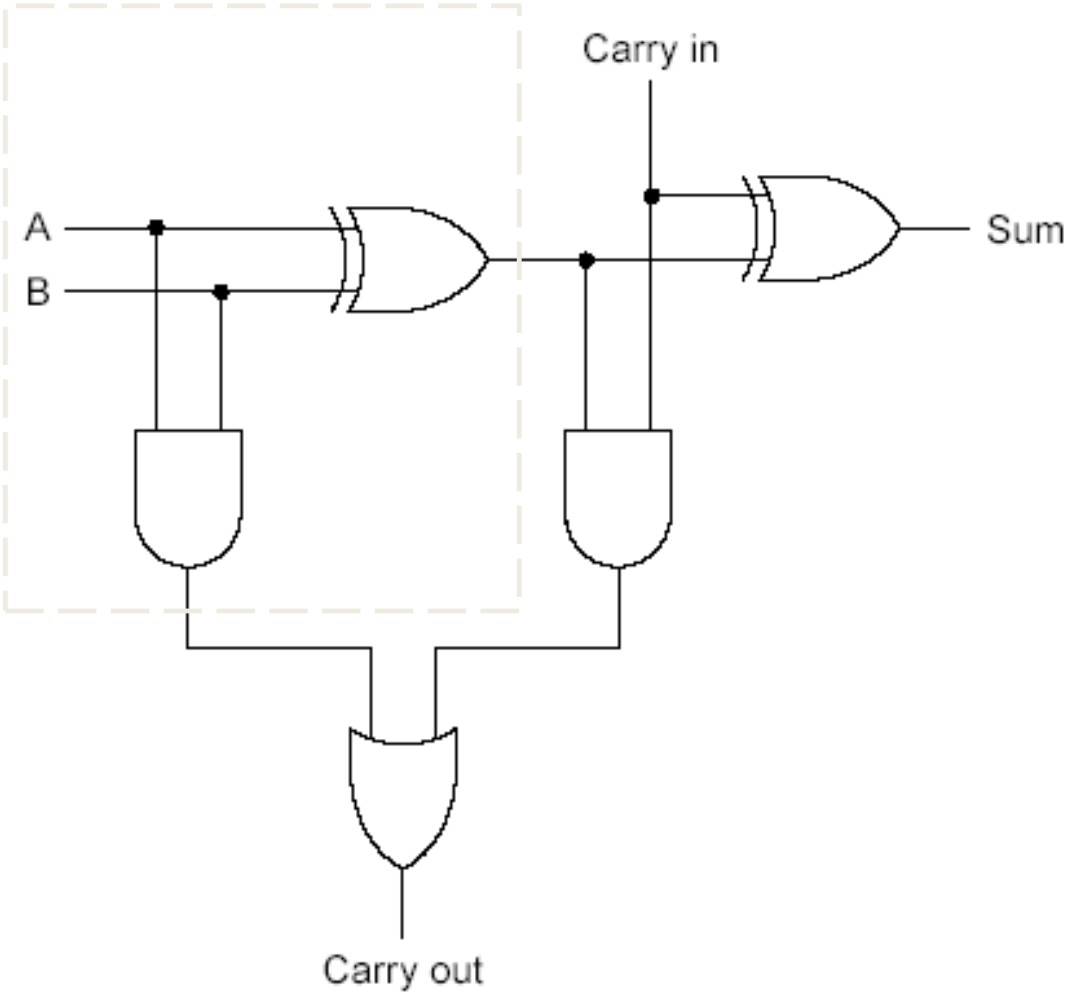
$$s = a \oplus b \oplus R_e$$

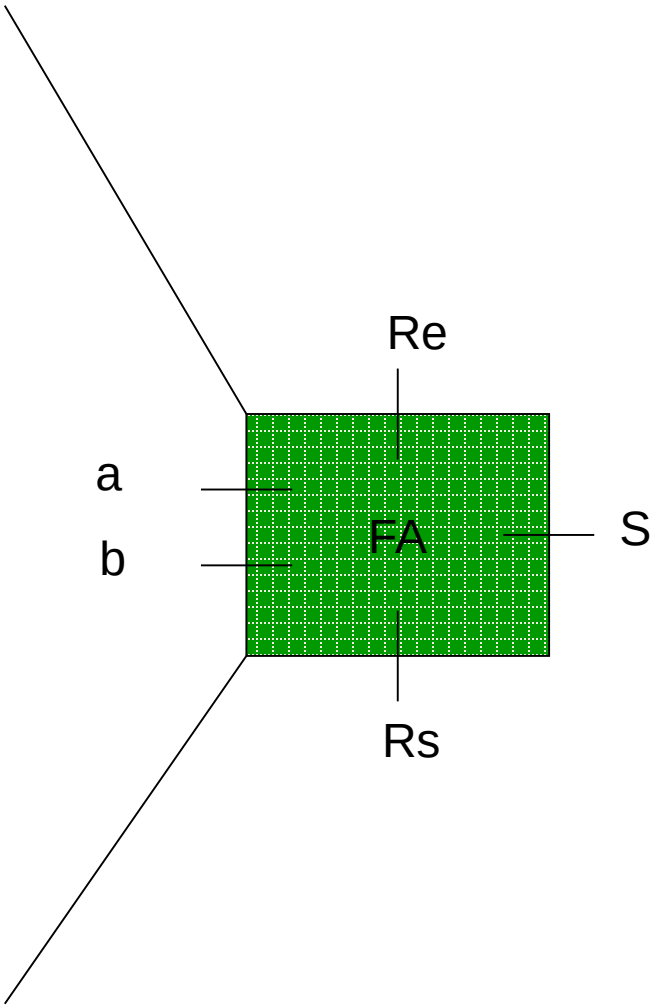
$$R_s = \bar{a}bR_e + a\bar{b}R_e + ab\bar{R}_e + abR_e$$

$$R_s = R_e(a \oplus b) + ab$$

a	b	Re	S	Rs
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adder (1 bit)

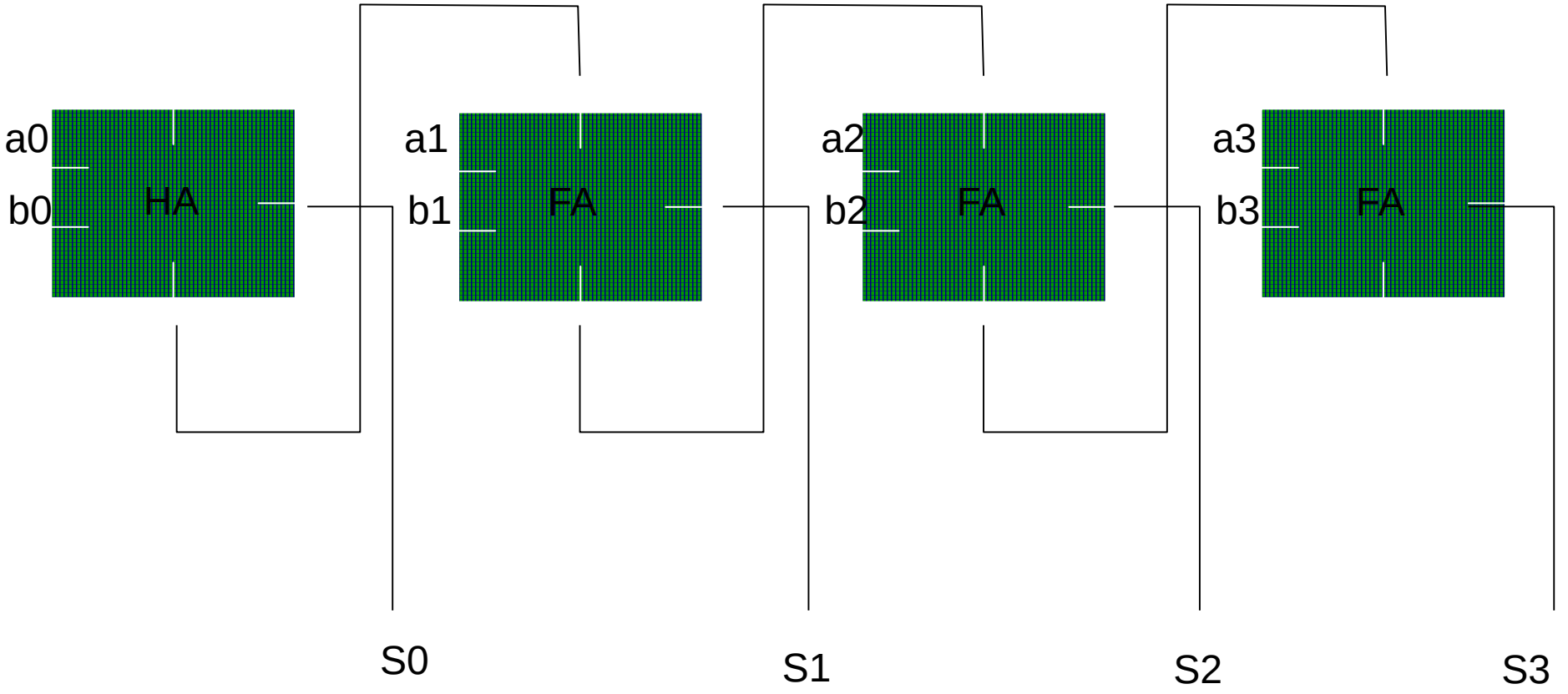




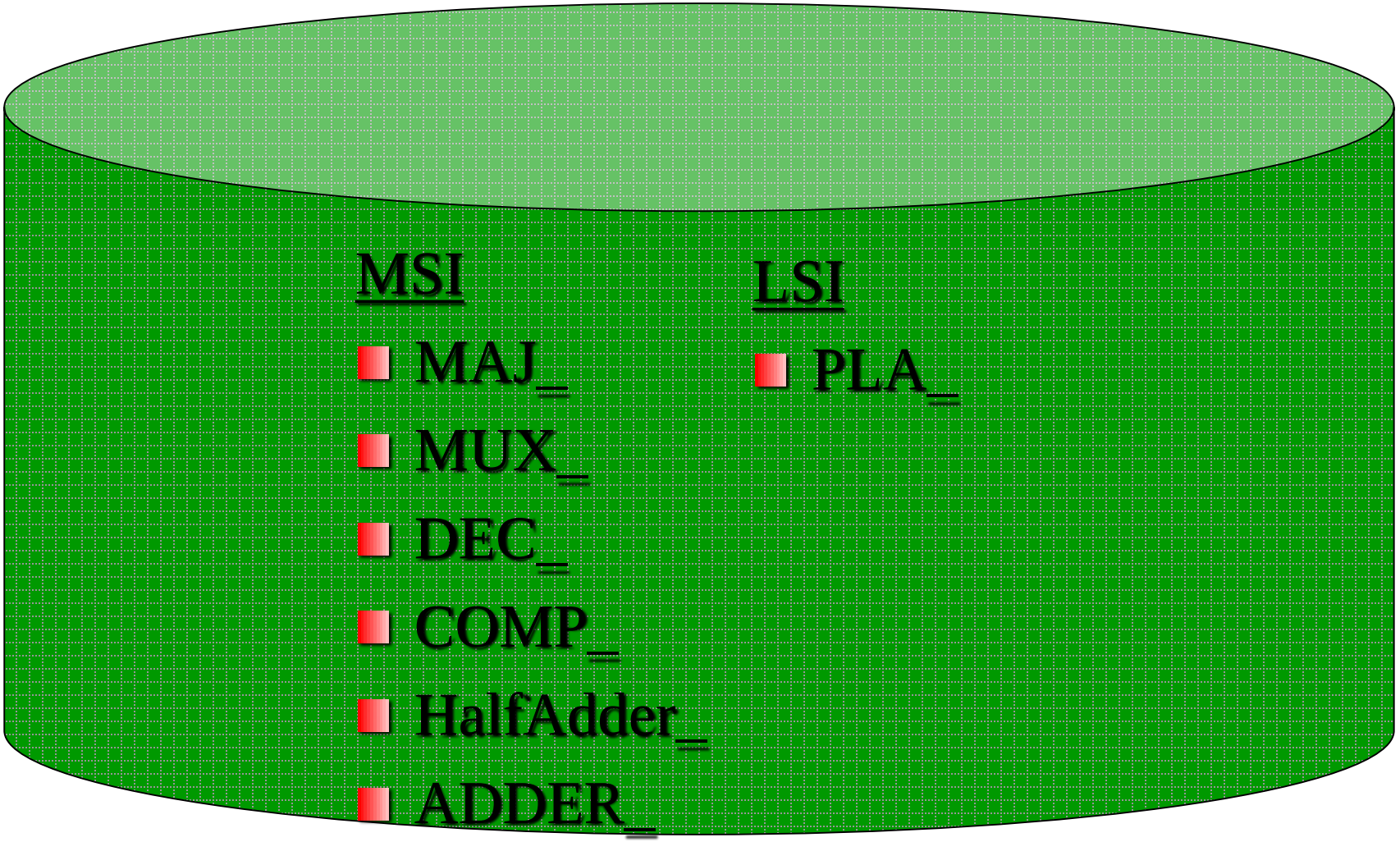
Full Adder 4 bits

additionneur à propagation de retenue

Temps de propagation très long



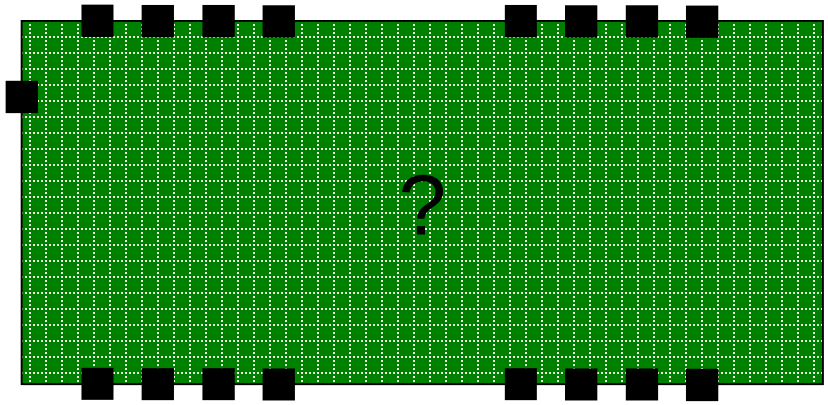
Notre bibliothèque de portes



Décaleur

- Le circuits dispose d'une entrée de donnée sur 8 bits et d'une ligne de commande '***c***' sur 1 bit

- Le circuit fournit en sortie, l'entrée décalée d'une place à gauche ($\overline{c}$) ou à droite en fonction de (*c)*



Décaleur

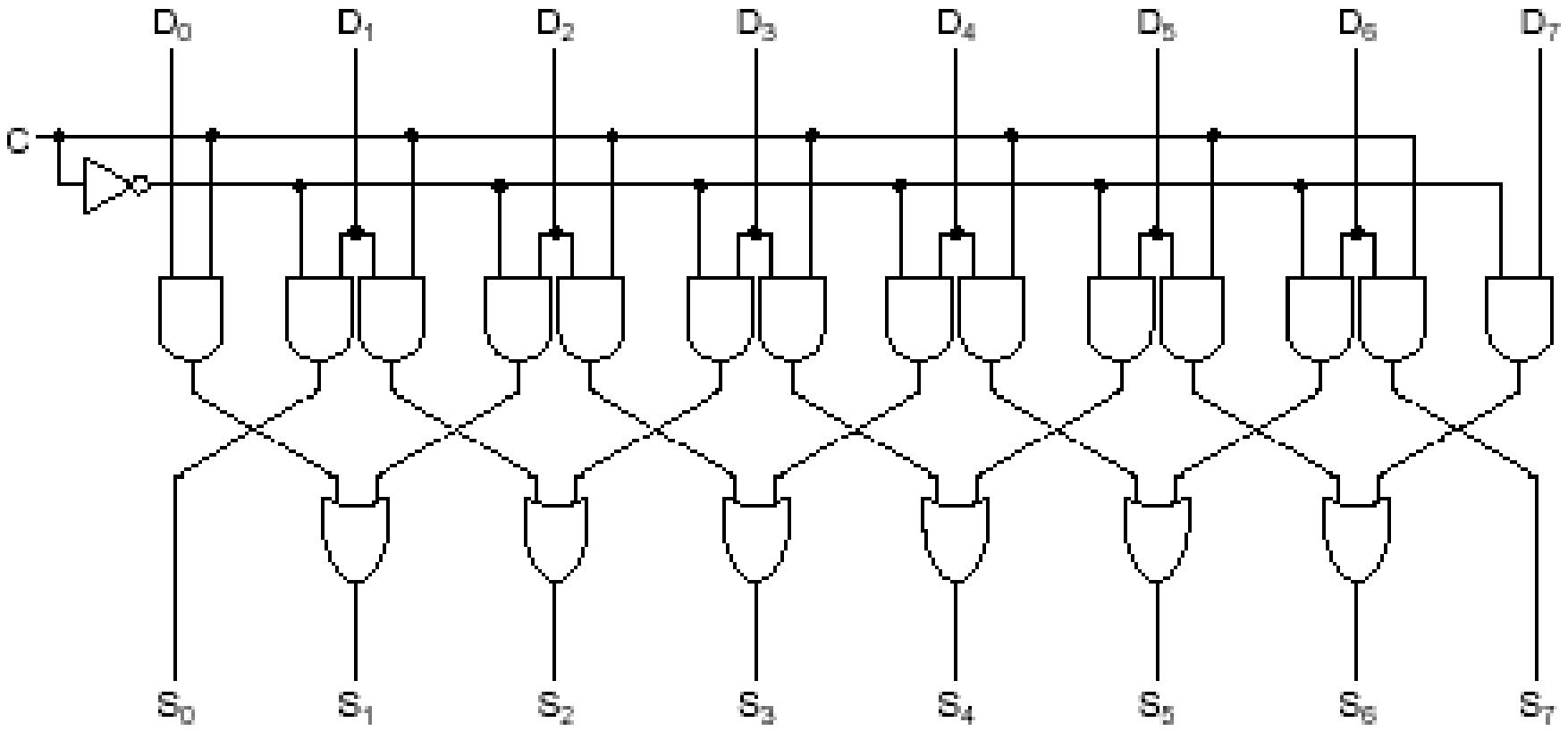
- Donner l'équation de chaque bit de sortie du décaleur 8 bits

- Dessiner ensuite le schéma en portes du décaleur d'une position à gauche ou à droite

Décaleur

- Donner l'équation de chaque bit de sortie du décaleur 8 bits
- $$\begin{cases} S_0 = e_1.c \\ S_7 = e_6.\overline{c} \\ S_i = e_{i+1}.c + e_{i-1}.\overline{c}, i \in [1,6] \end{cases}$$
- Dessiner ensuite le schéma en portes du décaleur d'une position à gauche ou à droite

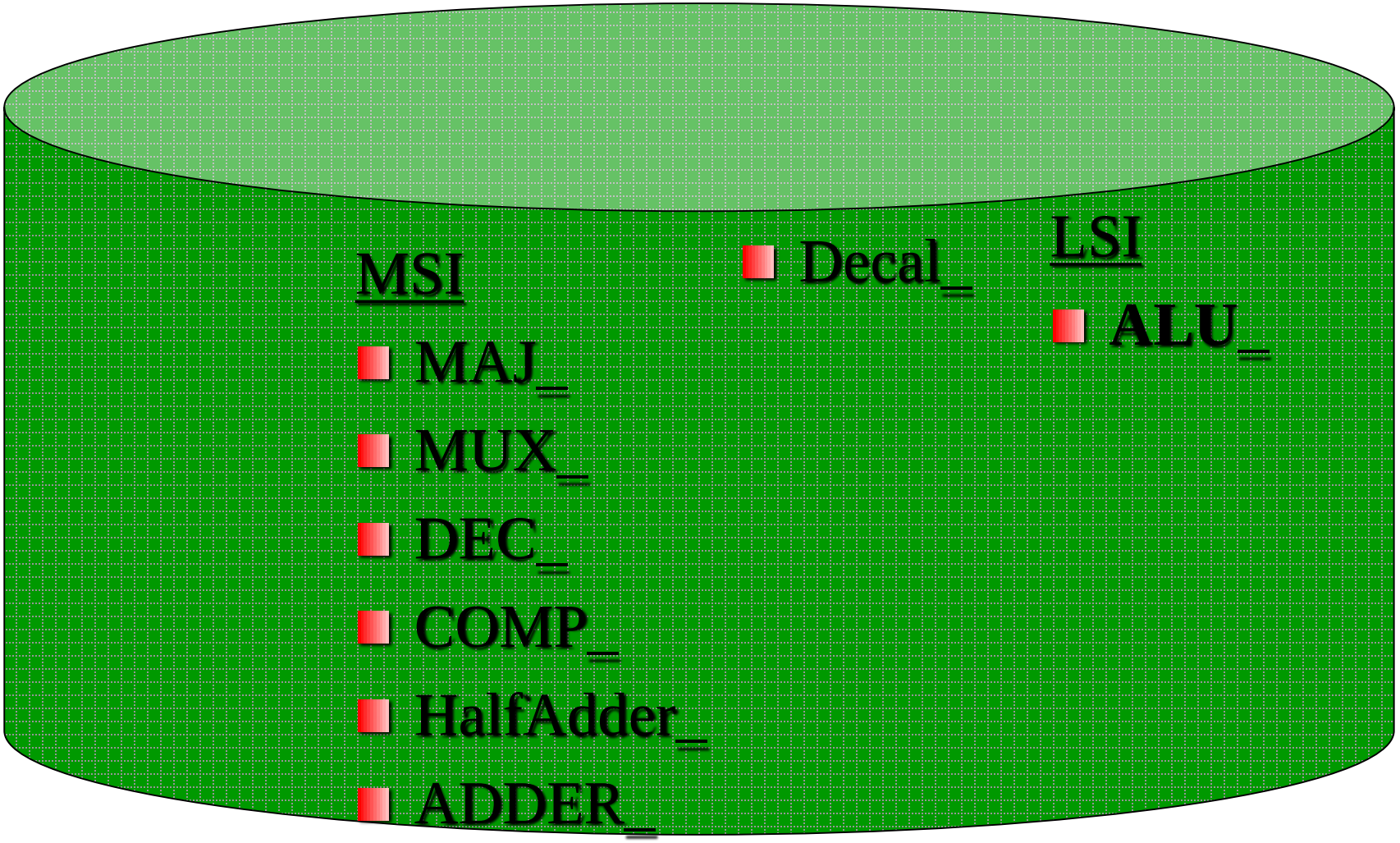
Décaleur



Décaleur logique / arithmétique

- On distingue décaleur logique et décaleur arithmétique
- Le premier considère un nombre non-signé et le deuxième un nombre signé.
- La différence intervient surtout sur le décalage à droite
  - Dans le cas logique la donnée injectée à gauche est un 0
  - Dans le cas arithmétique la donnée injectée à gauche est le bit de signe pour conserver la cohérence du complément à 2

Notre bibliothèque de portes



Unité de calcul du processeur

- Un processeur doit pouvoir réaliser toute fonction logique et arithmétique
- L'unité matérielle responsable de ces calculs est appellée Unité Arithmétique et Logique
  - UAL
  - ou ALU
- Elle dispose de deux entrées dont la taille dépend de l'architecture du processeur
- Elle fournie en sortie le résultat de l'opération choisie par la troisième entrée

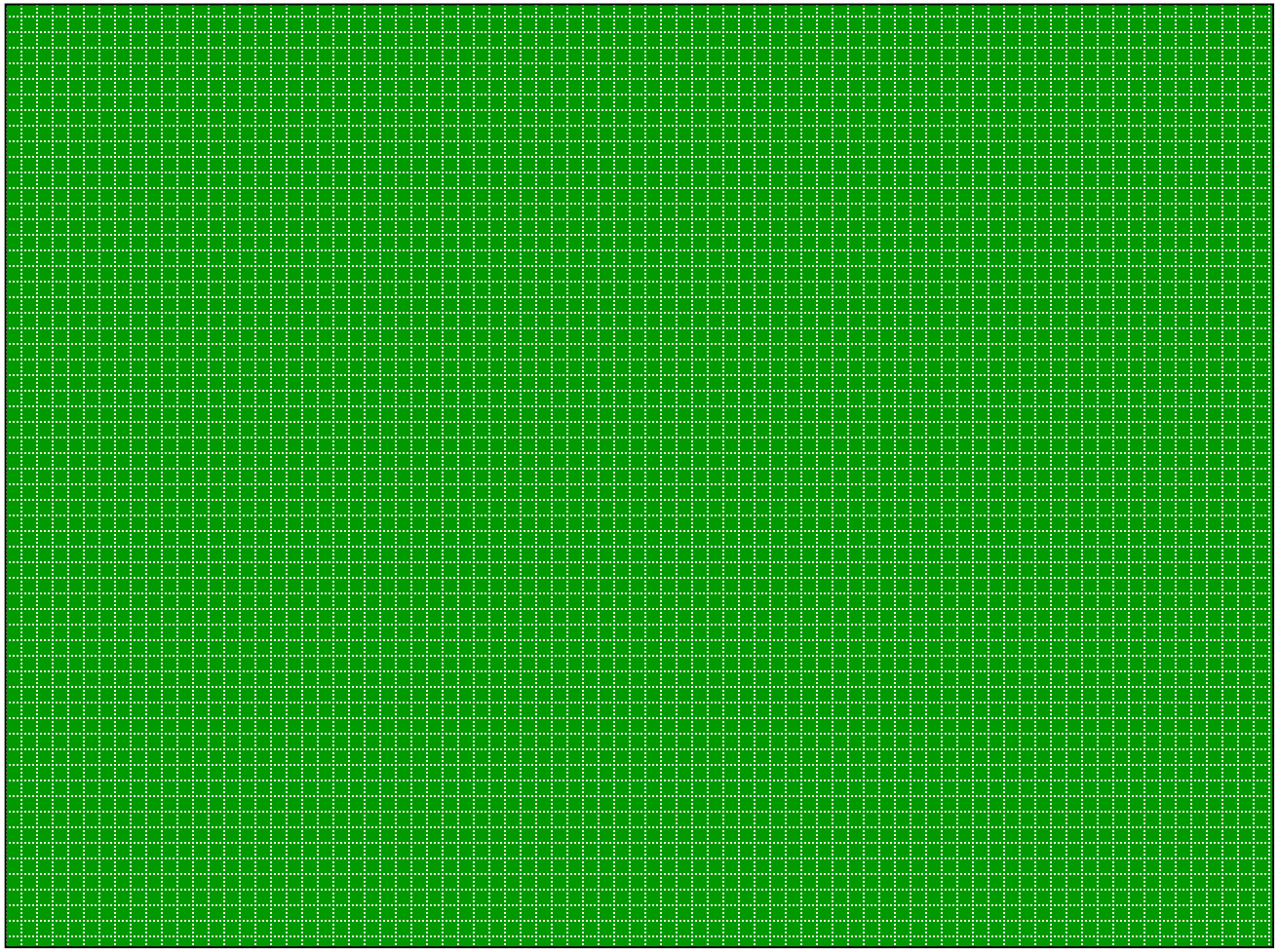
Réalisation structurelle d'une ALU

Cahier des charges :

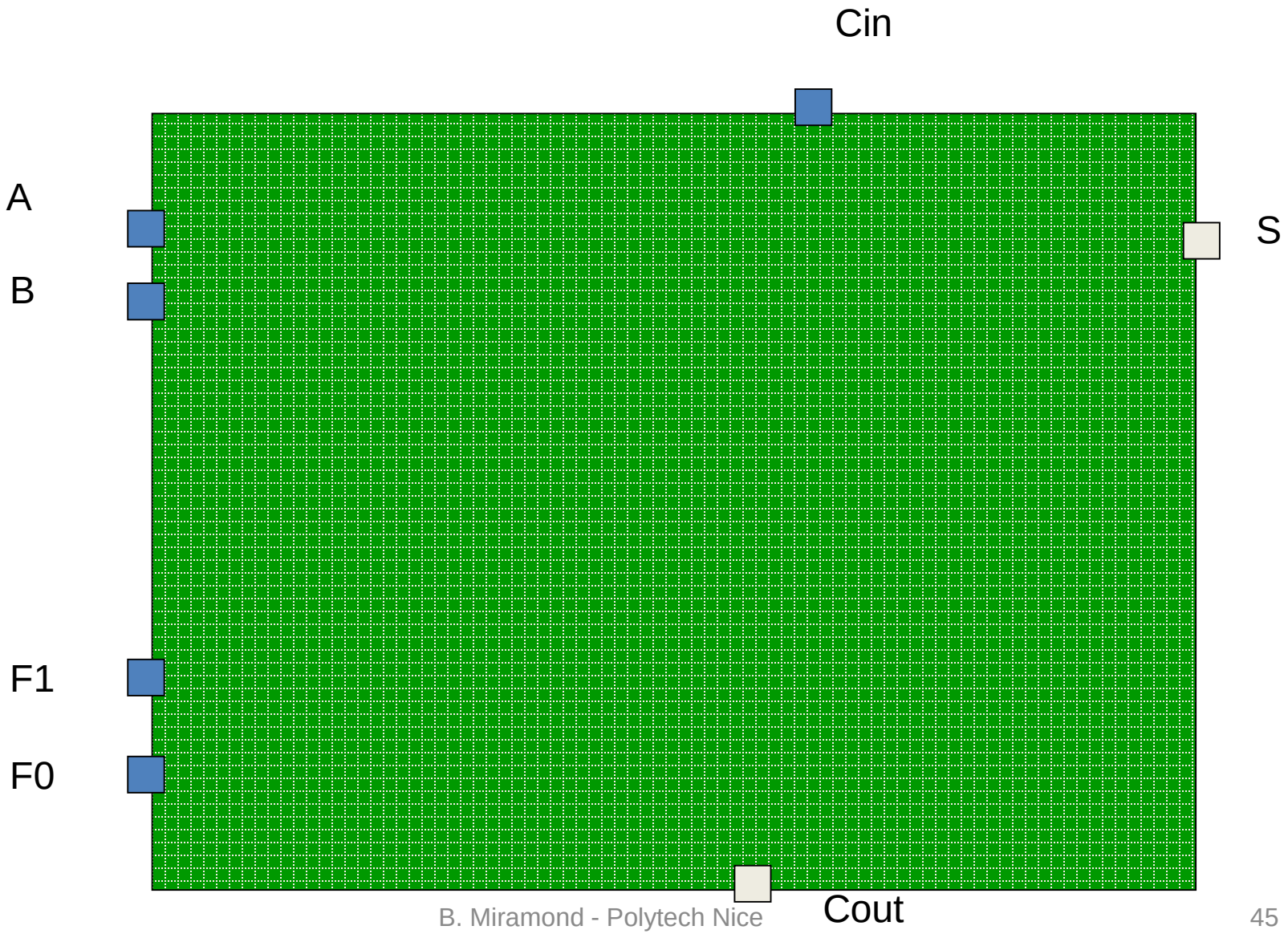
- Opérandes d'1 bit (A et B)
- Pouvant réaliser au choix les opérations
  - A or B
  - A and B
  - not B
  - A + B

Quels besoins ?

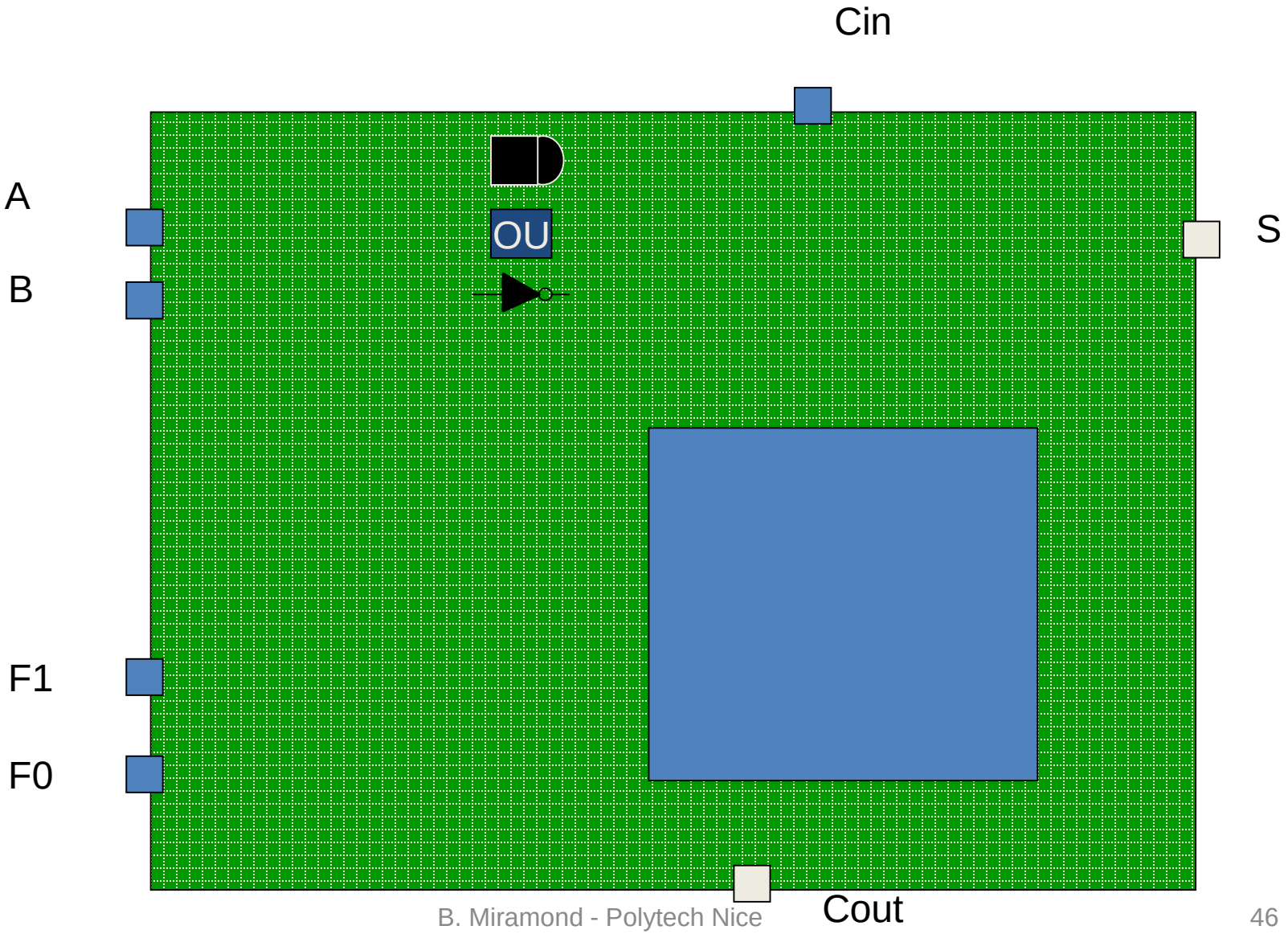
Interface ?



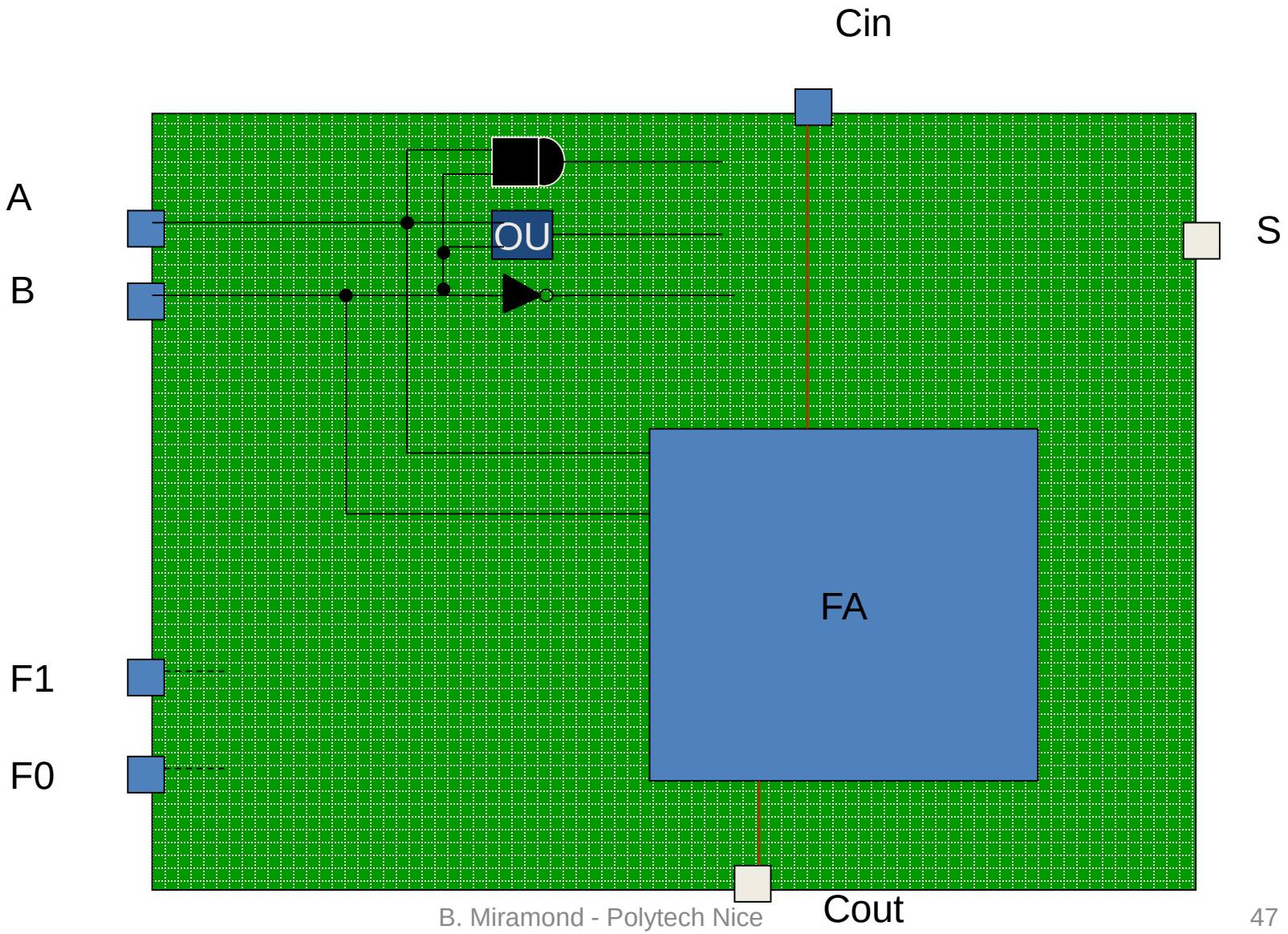
Interface ?



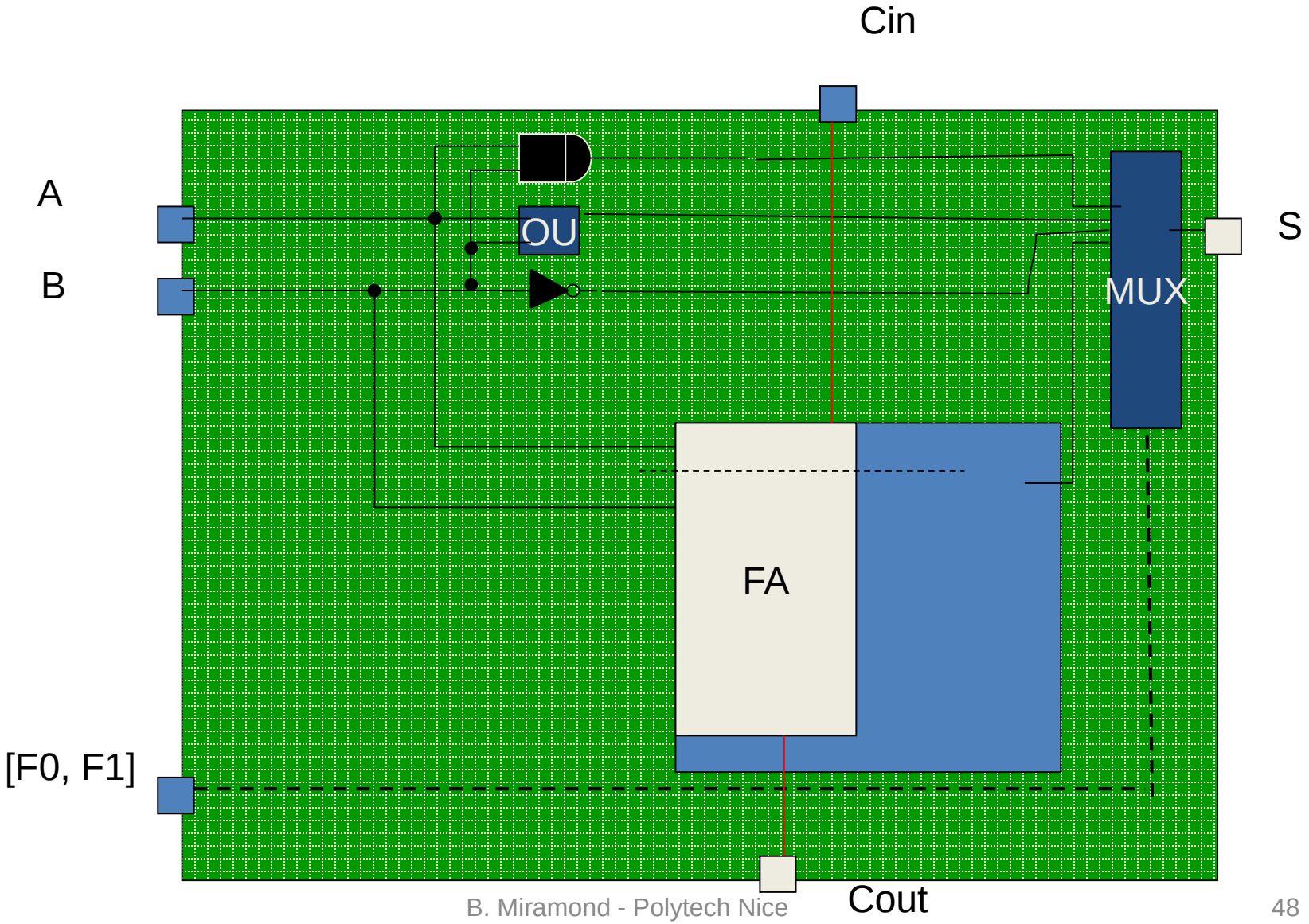
Vue structurelle



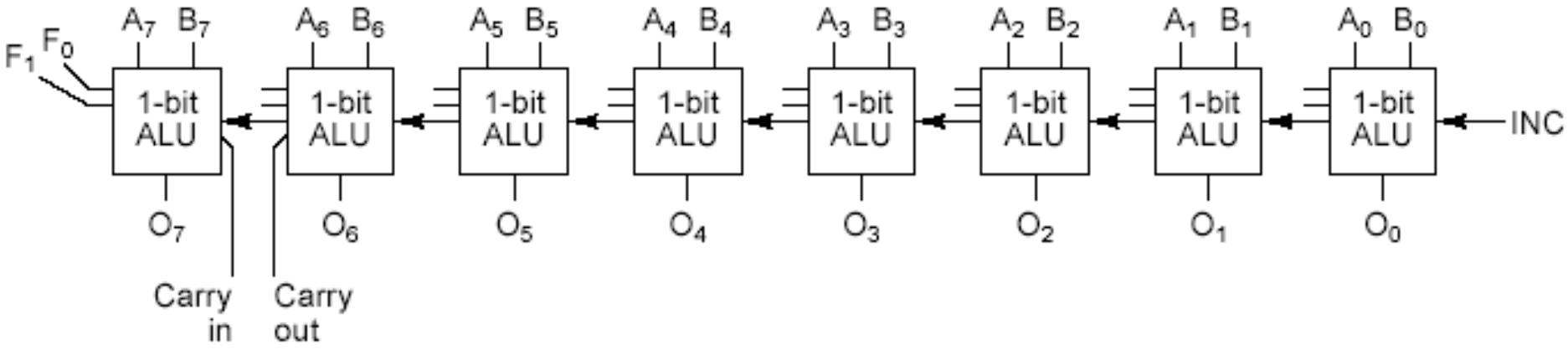
Vue structurelle

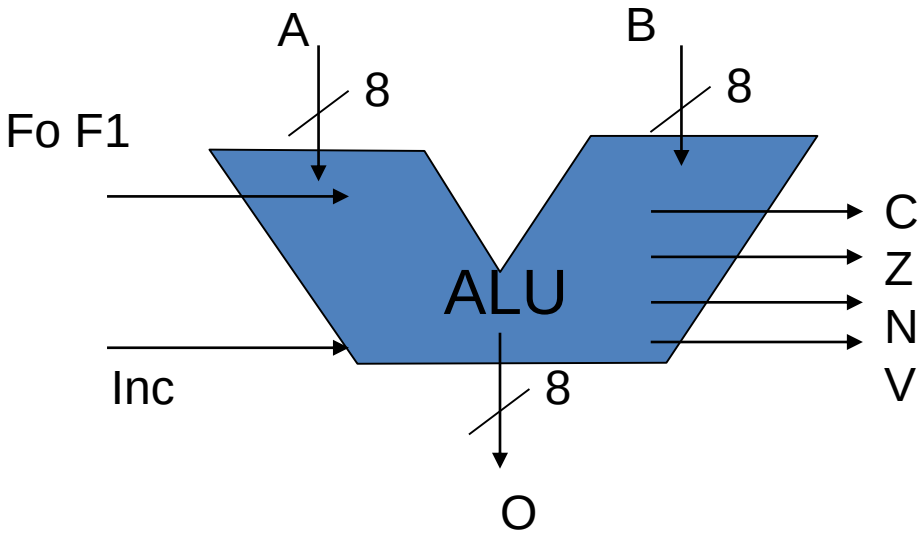


Vue structurelle



ALU_8bits





FLAGS

Carry
Zéro
Négatif
oVerflow

Les Flags

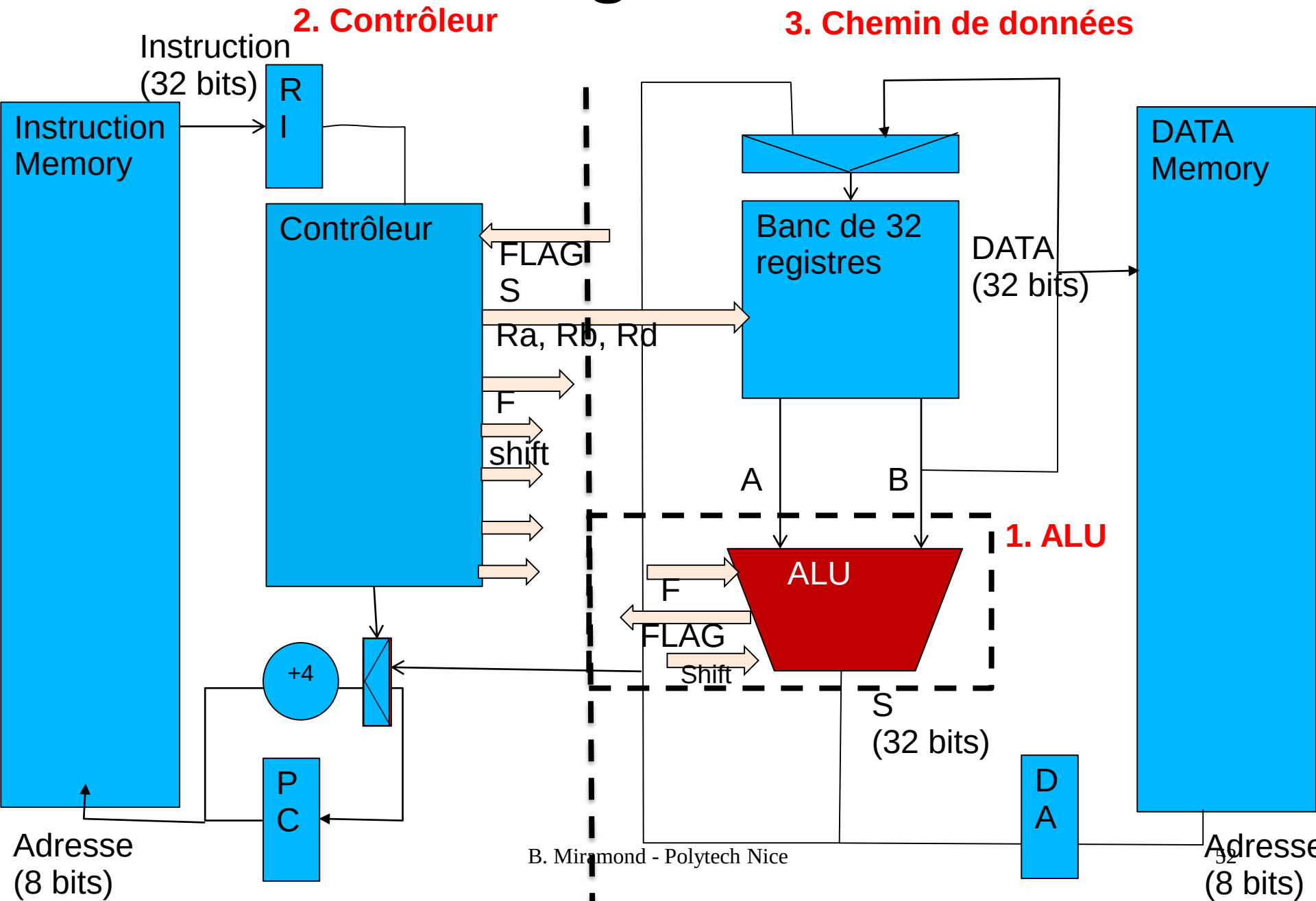
- C – Carry ou retenue sortante est générée par chaque opérateur arithmétique. Elle doit donc être sélectionnée en fonction du Codop (mux).
- Z – Zero est obtenu en testant si la sortie S de l'ALU est égale à la constante 0
- N – Negatif est obtenu en récupérant le 32^e bit de la sortie S qui en complément à 2 code le signe
- V – oVerflow détecte le dépassement de capacité qui est différent de la retenue. Elle est calculée en vérifiant le signe des opérandes et du résultat pour chaque opérateur arithmétique. Une sélection est faite en fonction du Codop.

Multiplieur

- Dans la version de Logisim que vous devez utiliser en projet, le multiplieur fournit est un multiplieur qui ne peux calculer que jusqu'à 31 bits.
- Vous devez donc trouver comment calculer une multiplication 32 bits à partir de multiplieurs plus petits.
- Exemple en base 10 sur 4 digits :
  - 1024 * 3435 = (1000 + 24) * (3000 + 435)
  - Soit 24*435 + 1000*3000 + 24*3000 + 1000*435
  - Soit 24*435 +(1*3 << 6) + (24*3 << 3) + (1*435 << 3)
  - Les décalages sont aussi en base 10 dans cet exemple
  - Les multiplieurs sont au plus sur 3 digits !
- A vous de généraliser et à adapter à la base 2

```
  1 0 2 4
* 3 4 2 5
------------
```

Architecture générale de PARM



L’interface du composant ALU dans le projet PARM

ALU (composant combinatoire)

Port	Direction	Taille	Rôle
A	In	32	Opérande A
B	In	32	Opérande B
Flags	Out	4	Drapeaux Z, N, V, C
Shifter	In	5	Nombre de décalage de 0 à 31
CarryIn	In	1	Retenue entrante
Codop	In	4	Sélection de l’opération
S	Out	32	Résultat

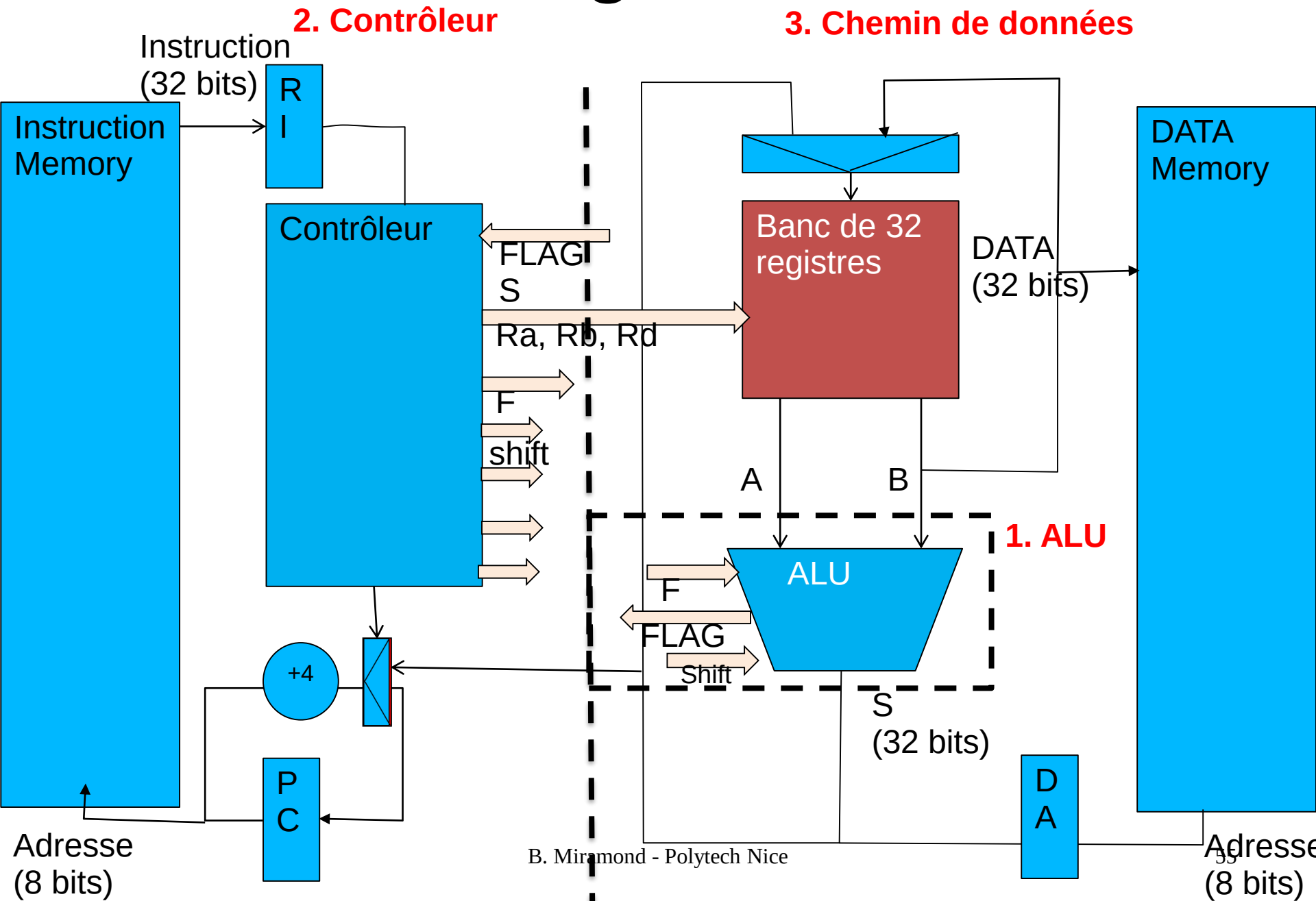
Tableau 1 - Interface de l'ALU

Les opérations supportées par l'ALU

CODOP	Opération	Instructions
0 0 0 0	A and B	AND
0 0 0 1	A xor B	EOR
0 0 1 0	B << shift	LSL
0 0 1 1	B >> shift	LSR
0 1 0 0	B >> shift (arith)	ASR
0 1 0 1	A + B + Cin	ADC
0 1 1 0	A – B + Cin – 1	SBC
0 1 1 1	B >> shift (rot)	ROR
1 0 0 0	A and B	TST
1 0 0 1	0 – B	RSB
1 0 1 0	A – B	CMP
1 0 1 1	A + B	CMN
1 1 0 0	A or B	ORR
1 1 0 1	A * B	MUL
1 1 1 0	A and not B	BIC
1 1 1 1	Not B	MVN

Tableau 2 - Opérations de l'ALU

Architecture générale de PARM



L’interface du banc de registres du projet PARM

Banc de registre (composant séquentiel)

Port	Direction	Taille	Rôle
DataIn	In	32	Données à enregistrer
Aout	Out	32	Opérande A
Bout	Out	32	Opérande B
Clk	In	1	Clock
RegDest	In	3	Sélection du registre en écriture
RegA	In	3	Sélection du registre en lecture pour la source A
RegB	In	3	Sélection du registre en lecture pour la source B
Reset	In	1	Reset des registres

Tableau 3 - Interface du Banc de registres