Composants combinatoires

Formalisme graphique, Logique négative/positive

Portes intégrées,Multiplexeurs, Demultiplexeurs

Codeurs, Décodeurs, Transcodeurs

Comparateurs, ALU

Mémoire, Composants programmables (principes)

Réalisation en électronique

0/1 représentés par des tensions, courants,
charges, fréquences, ...

Classiquement TENSIONS : Niveau haut = H (le plus positif)
Niveau bas = L (B) (le plus négatif)

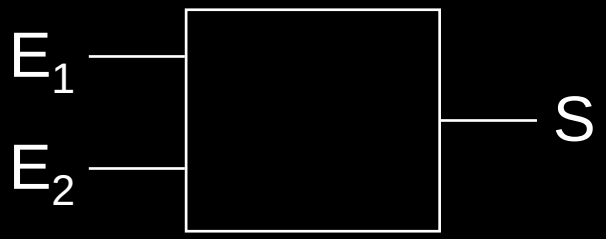
Association d'une information binaire à un niveau :

Convention positive	H ⟶ 1
(ou logique positive)	L ⟶ 0
Convention négative	H ⟶ 0
(ou logique négative)	L ⟶ 1

Attention : convention utilisée

La fonction logique réalisée par un circuit dépend de la convention utilisée.

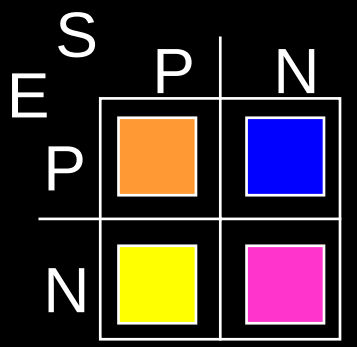
Exemple : Soit le circuit physique suivant



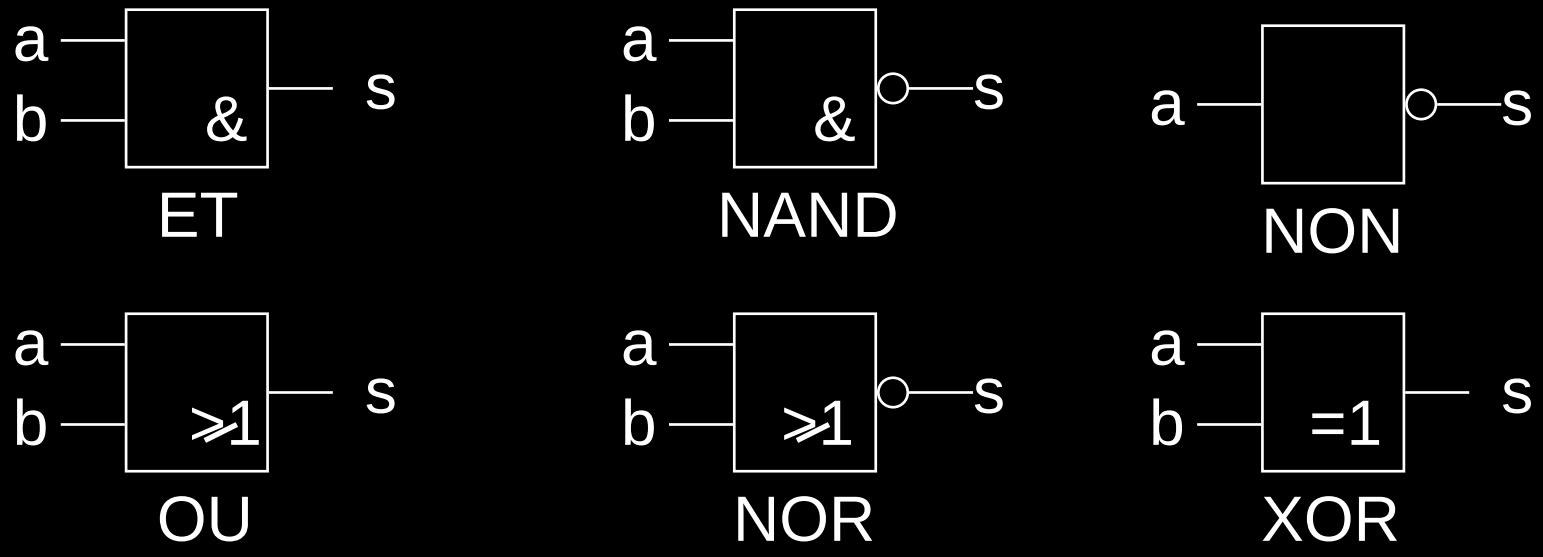
E_1 \ E_2	L	H
L	L	L
H	L	H

S

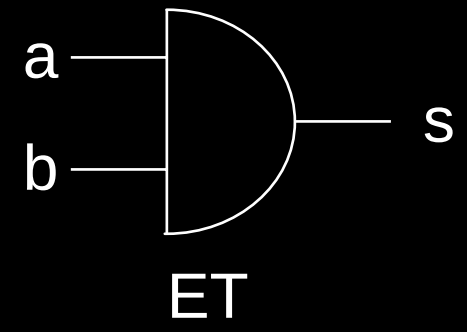
Table physique



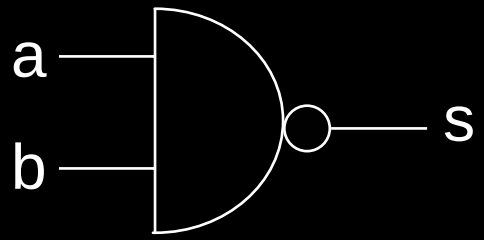
Représentation graphique : Norme française



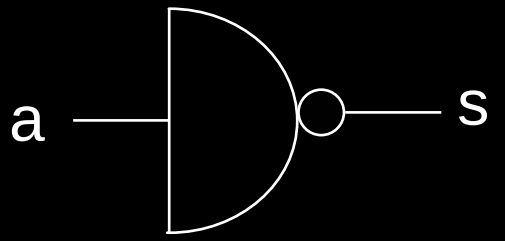
Représentation graphique : Norme américaine



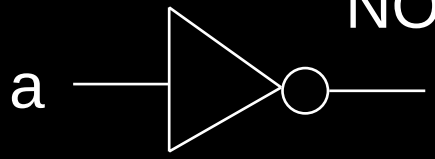
ET

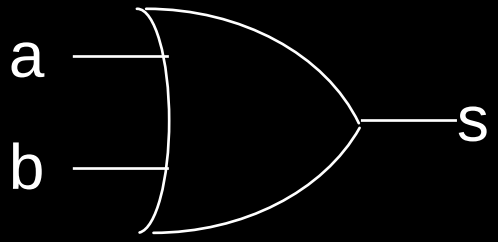


NAND

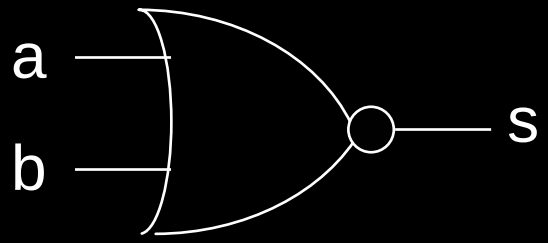


NON

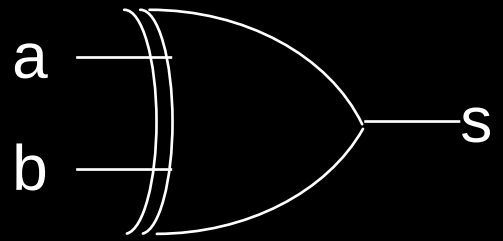




OU

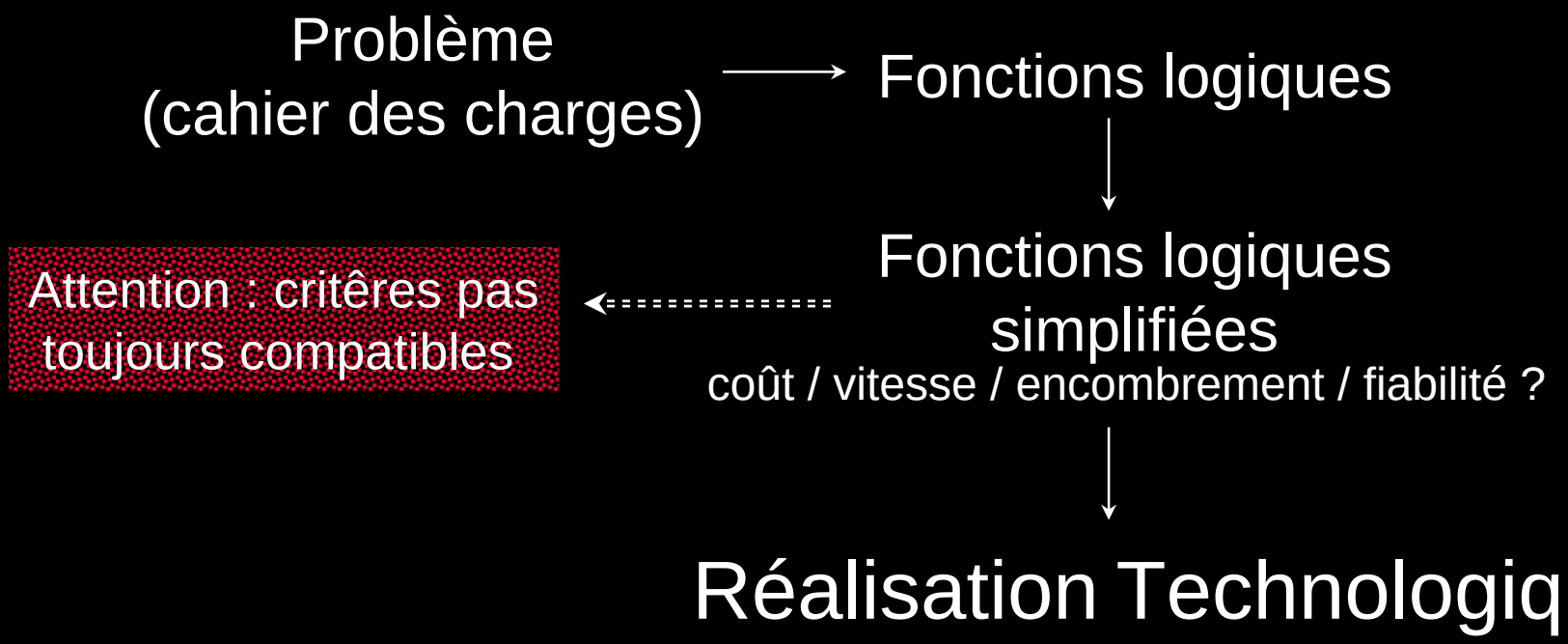


NOR



XOR

Moyens physiques de réalisation des fonctions logiques



Simplification /optimisation ?

Méthodes «classiques» de simplifications :

- pas de solution unique
- indépendant de la technologie
- le temps n'est pas pris en compte

La simplification «mathématique» n'est pas toujours optimale en regard des critêres d'optimisation technologiques.

Combinatoire : définition
(définition sommaire)

Combinatoire : $S = f(a,b,c,..,n)$

idem Boucle ouverte

Séquentiel : $S = f(a,b,c,..,n,S)$

idem Systèmes bouclés

Composants combinatoires

(au catalogues des fournisseurs)

- Portes intégrés et inverseurs (Gates)
- Portes cascadable (Expanders)
- Multiplexeur / demultiplexeur
- Codeurs / Decodeurs
- Transcodeurs
- Comparateurs / Detection d'erreurs
- Circuits arithmétiques (add, ALU, mult)

! →
- Mémoires
- Composants programmables (PLD)

Portes intégrées (glue logic)

Porte de base la plus fabriquée : NAND (GOC)

		TTL	CMOS
CI =	4 NAND à 2 entrées	7400	4011
	3 « 3 «	7410	4023
	2 « 4 «	7420	4012
	1 « 8 «	7430	4068

Aussi : 6 inverseurs

Remarque : 4 NAND à 2 entrées
2*4 entrées + 1*4 sorties + Vcc + Gnd
= 14 pattes

Portes intégrées (2)

Options technologiques : familles logiques
(TTL,CMOS, BiCMOS, ECL ..)
Entrées : classique, triggée
Sorties : collecteur (drain) ouvert, sortie 3 états ...

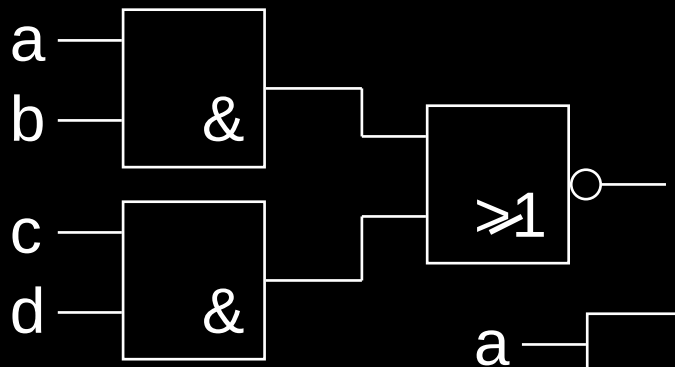
Variété moindre : ET, OU, XOR ...

Problème du nombre de boitier pour réaliser
une fonction logique $\longrightarrow$ INTEGRATION (VLSI/ULSI)

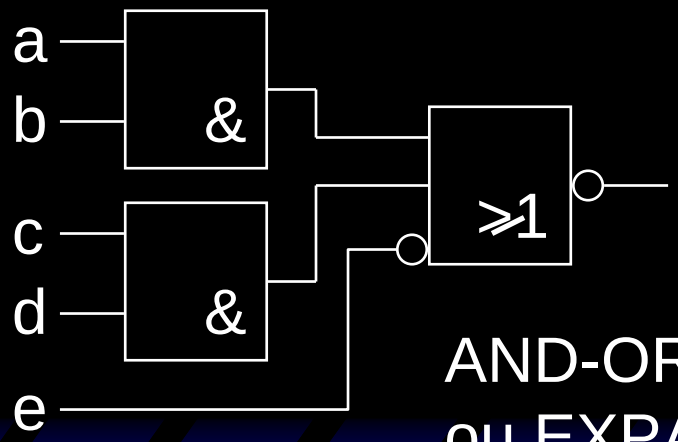
10 entrées = $2^{2^{10}}$ fonctions possibles
$\longrightarrow$ Choix des meilleures fonctions

Expander (historique)

Les premières portes complexes intégrées :
AND-OR-INVERT (exemple AOI 2-2-1)



$$s = \overline{a.b + c.d}$$

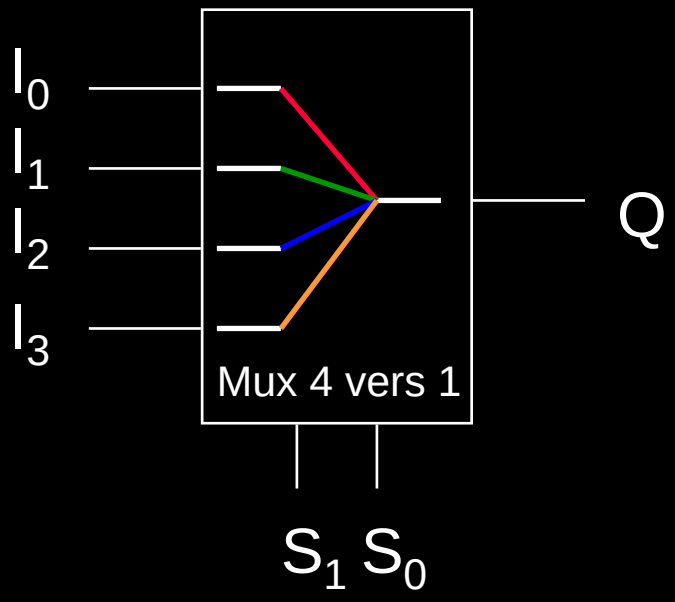


$$s = \overline{a.b + c.d + \overline{e}}$$

AND-OR-INVERT expandable
ou EXPANDER

Multiplexeur

Sélection d'une voie parmi 2^N par N bits de commande



Si $(S_1S_0)_2 = 0$ alors $Q = I_0$

$$Q = \overline{S_0}.\overline{S_1}.I_O$$

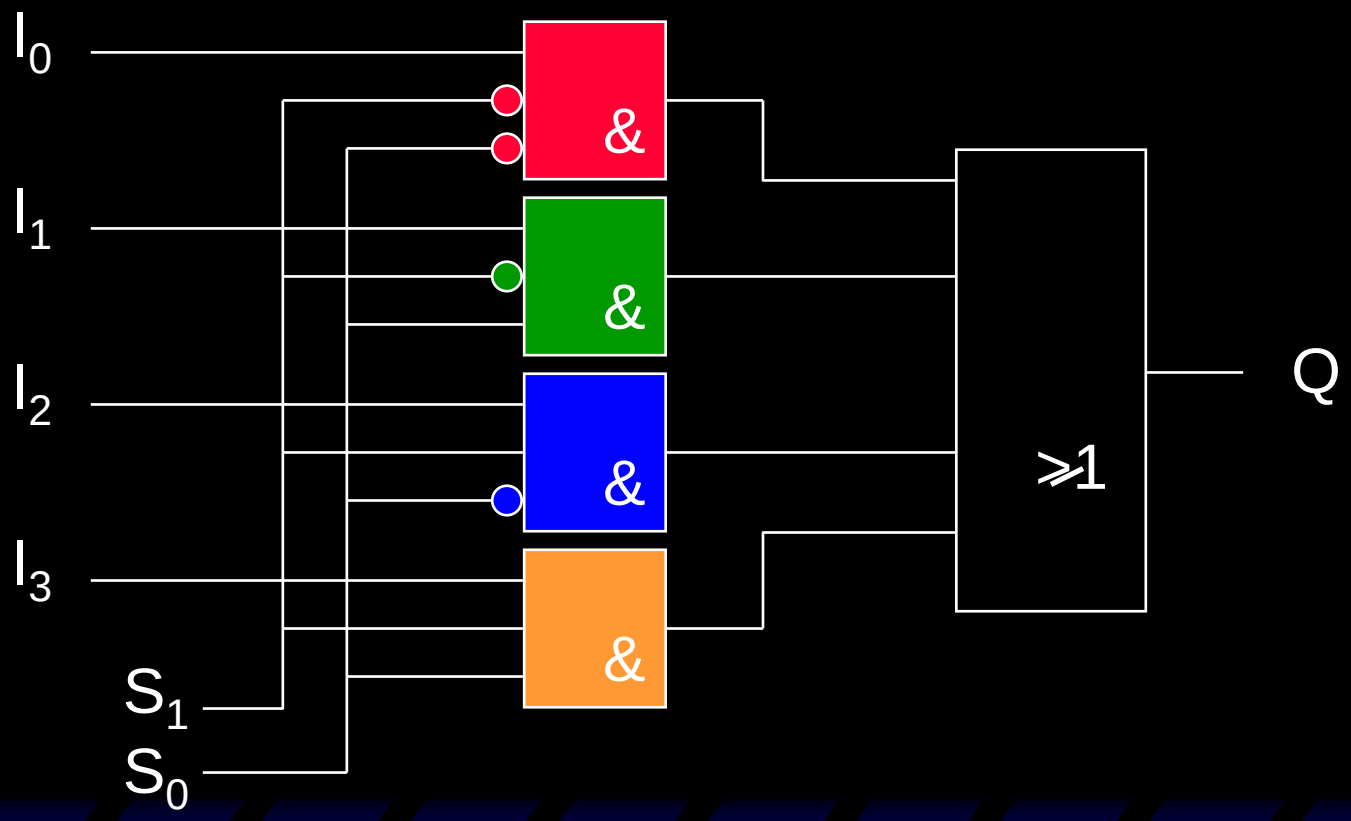
Si $(S_1S_0)_2 = 1$ alors $Q = I_1$

$$Q = S_0.\overline{S_1}.I_1$$

$$Q = \overline{S_1}.\overline{S_0}.I_0 + \overline{S_1}.S_0.I_1 + S_1.\overline{S_0}.I_2 + S_1.S_0.I_3$$

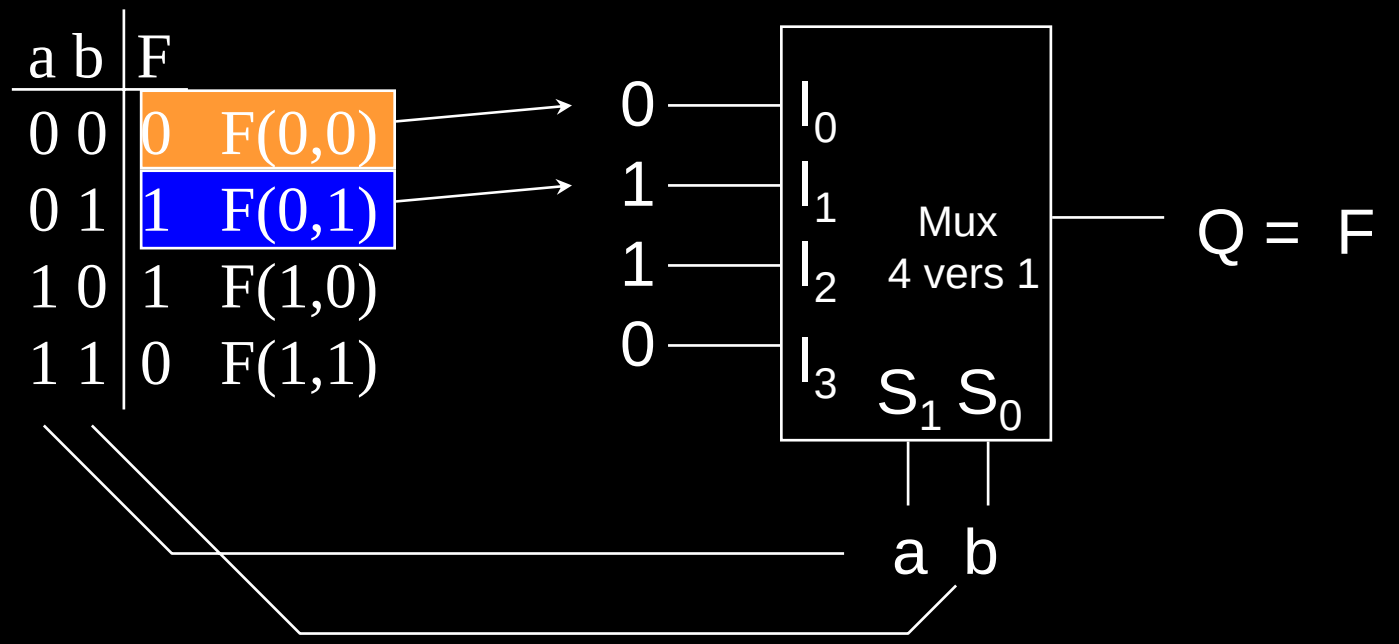
Multiplexeur (constitution)

$$Q = \overline{S_1}.\overline{S_0}.I_0 + \overline{S_1}.S_0.I_1 + S_1.\overline{S_0}.I_2 + S_1.S_0.I_3$$



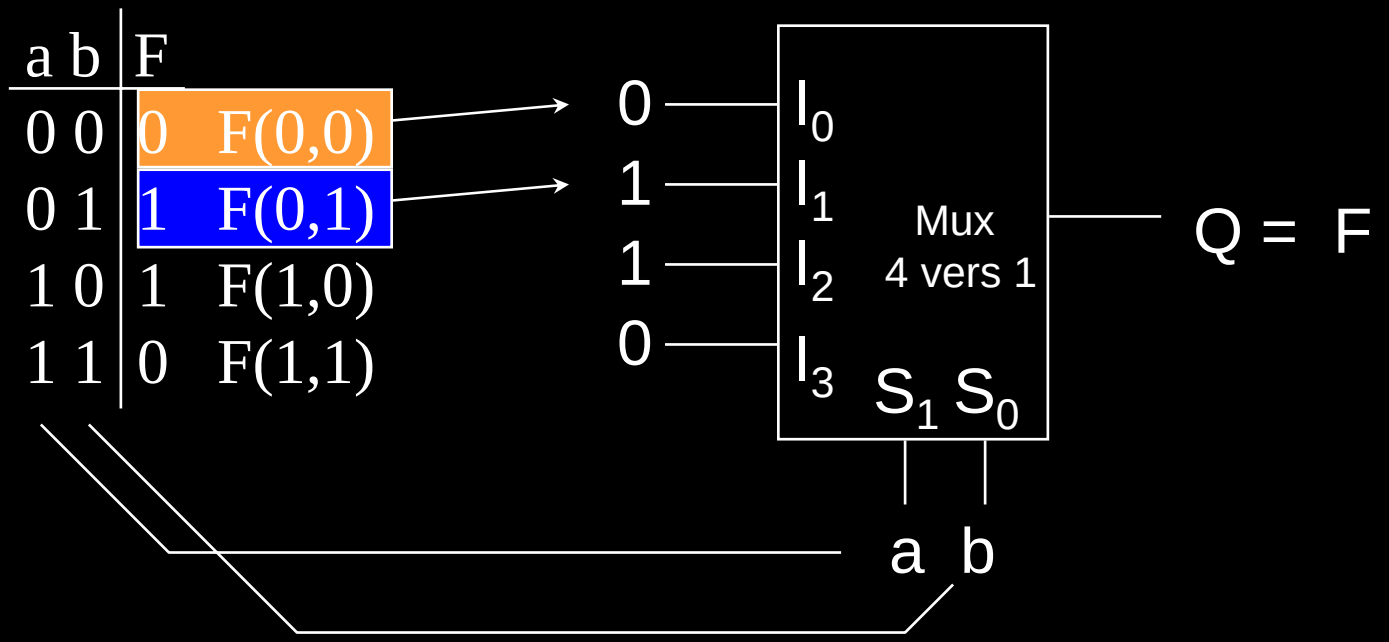
Multiplexeur : réalisation de fonctions

Utilisation de la première forme canonique



Multiplexeur : réalisation de fonctions

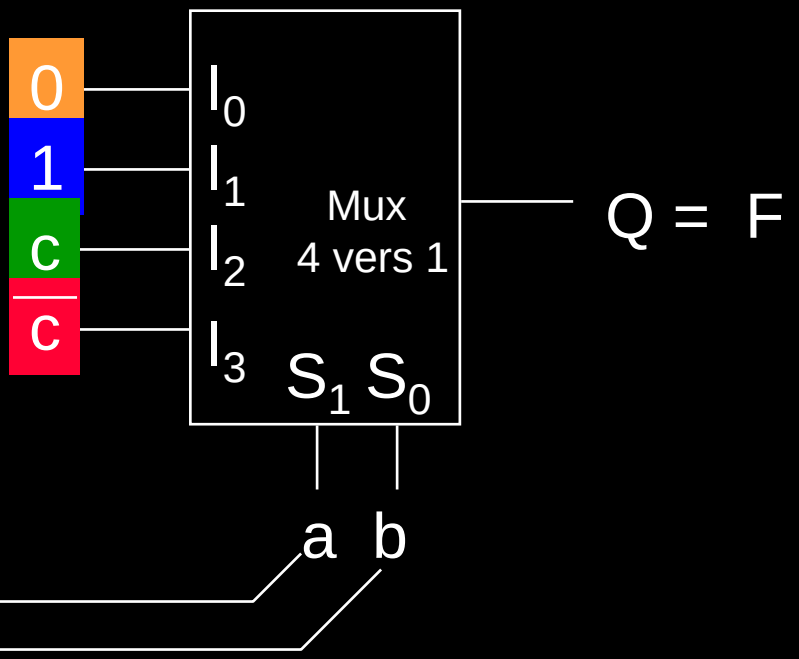
Utilisation de la première forme canonique



Toute fonction logique de N variables est réalisable avec un multiplexeur de 2^N vers 1

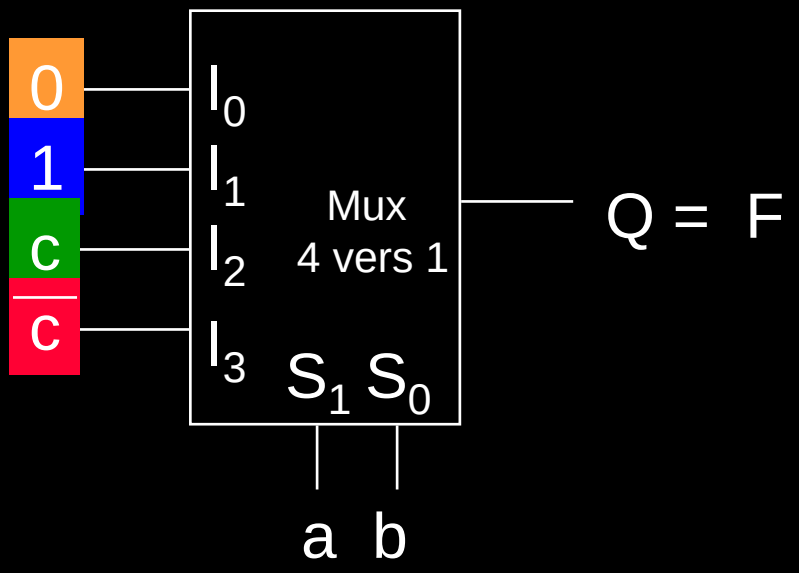
Multiplexeur : réalisation de fonctions (2)

a b c	F	
0 0 0	0	$(ab)_2 = 0$
0 0 1	0	F = 0
0 1 0	1	$(ab)_2 = 1$
0 1 1	1	F = 1
1 0 0	0	$(ab)_2 = 2$
1 0 1	1	F = c
1 1 0	1	$(ab)_2 = 3$
1 1 1	0	F = $\overline{c}$



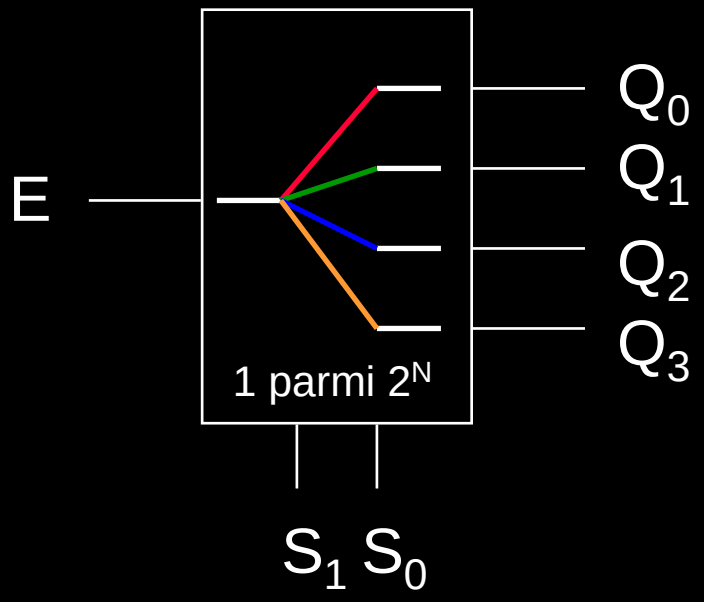
Multiplexeur : réalisation de fonctions (2)

a b c	F	
0 0 0	0	$(ab)_2 = 0$
0 0 1	0	F = 0
0 1 0	1	$(ab)_2 = 1$
0 1 1	1	F = 1
1 0 0	0	$(ab)_2 = 2$
1 0 1	1	F = c
1 1 0	1	$(ab)_2 = 3$
1 1 1	0	F = $\overline{c}$



Toute fonction logique de N variables est réalisable avec un multiplexeur de 2^{N-1} vers 1 et un inverseur

Démultiplexeur : 1 parmi 2^n

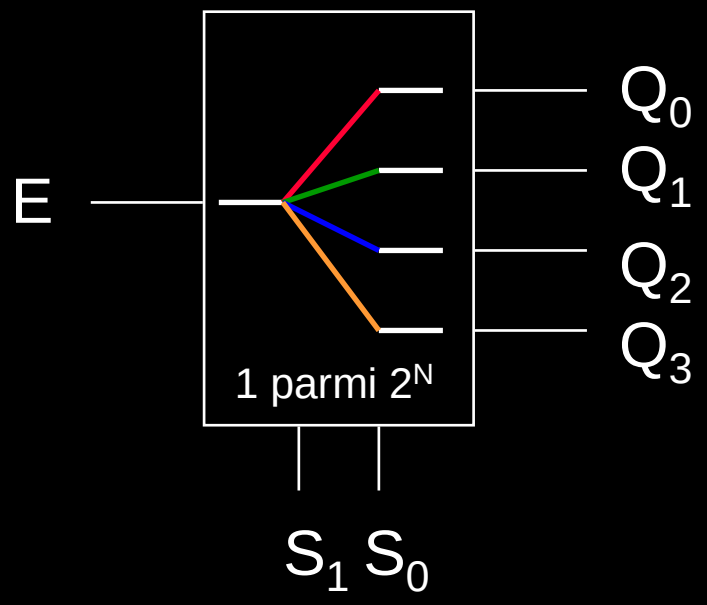


Q_0 = $\overline{E}$ si $(S_1S_0)_2$=0
E sinon

Q_1 = $\overline{E}$ si $(S_1S_0)_2$=1
E sinon

Remarque : E peut ne pas être «disponible»
Sortie sélectionnée = 1 les autres 0
ou Sortie sélectionnée = 0 les autres 1

Démultiplexeur : 1 parmi 2^n



E=1

$$Q_0 = E.\overline{S_1}.\overline{S_0} + \overline{E}.\overline{\overline{S_1}.\overline{S_0}}$$
$$= E.\overline{S_1}.\overline{S_0} + \overline{E}.(S_1 + S_0)$$

$$Q_0 = \overline{S_1}.\overline{S_0}$$

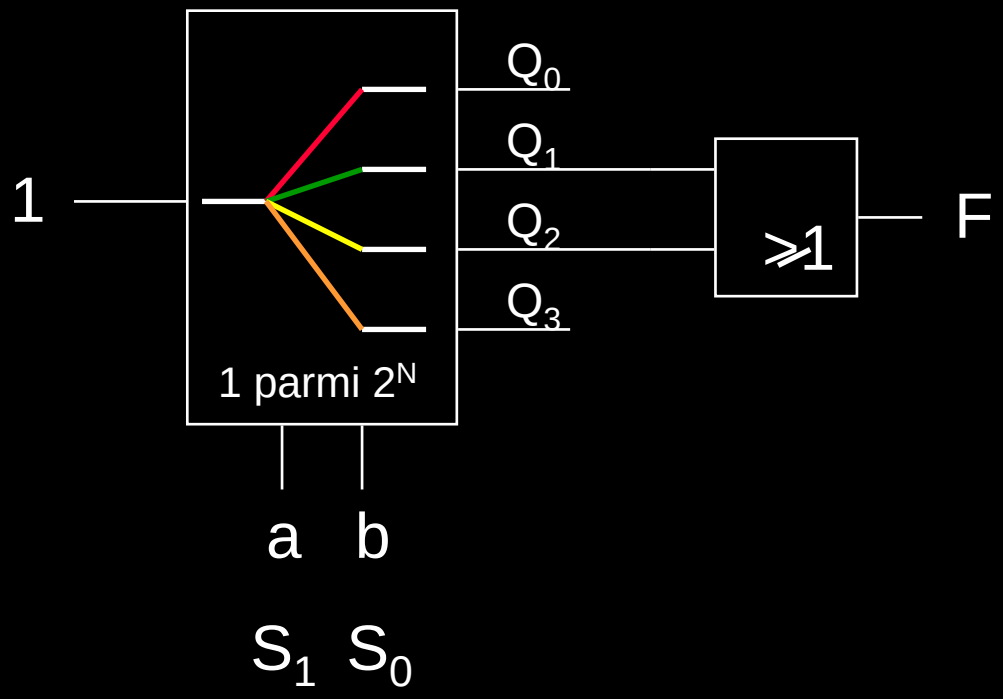
$$Q_1 = E.\overline{S_1}.S_0 + \overline{E}.\overline{\overline{S_1}.S_0}$$
$$= E.\overline{S_1}.S_0 + \overline{E}.(S_1 + \overline{S_0})$$

$$Q_1 = \overline{S_1}.S_0$$

Qi=$(i)_2$

Démultiplexeur: réalisation de fonctions

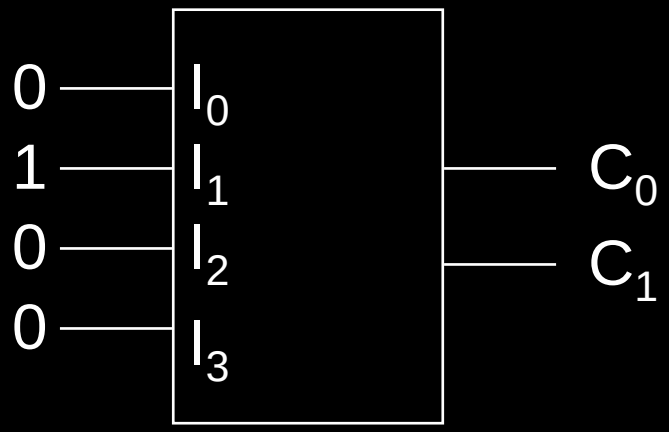
a	b	F	
0	0	0	F(0,0)
0	1	1	F(0,1)
1	0	1	F(1,0)
1	1	0	F(1,1)



Codeur (ou Encodeur)

Faire correspondre un mot code à un symbole

1 entrée parmi N



Le code de l'entrée

Exemple : Clavier / Scan code
Caractère / Code ASCII

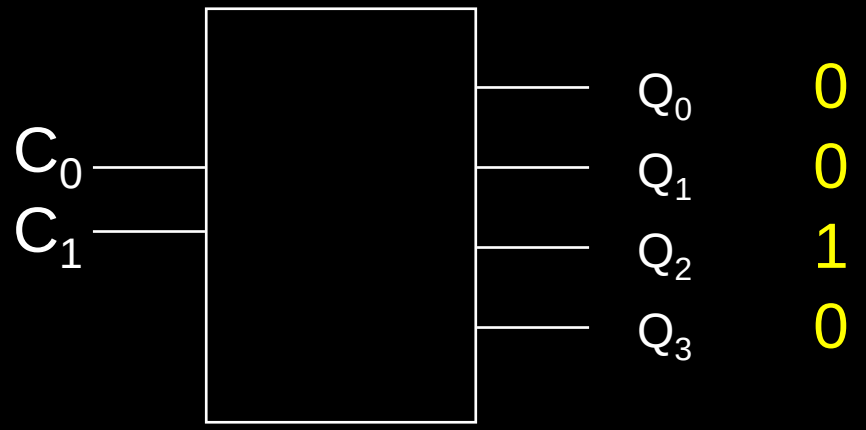
Décodeur

Remarque : Multiplexeur ⟷ Démultiplexeur

Codeur ⟷ Décodeur

Décodeur = Démultiplexeur (à E fixe)

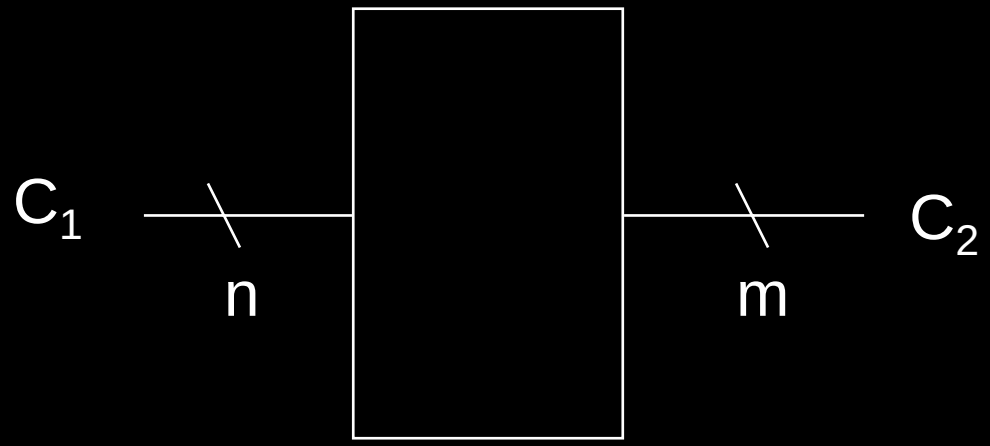
Exemple 0 1



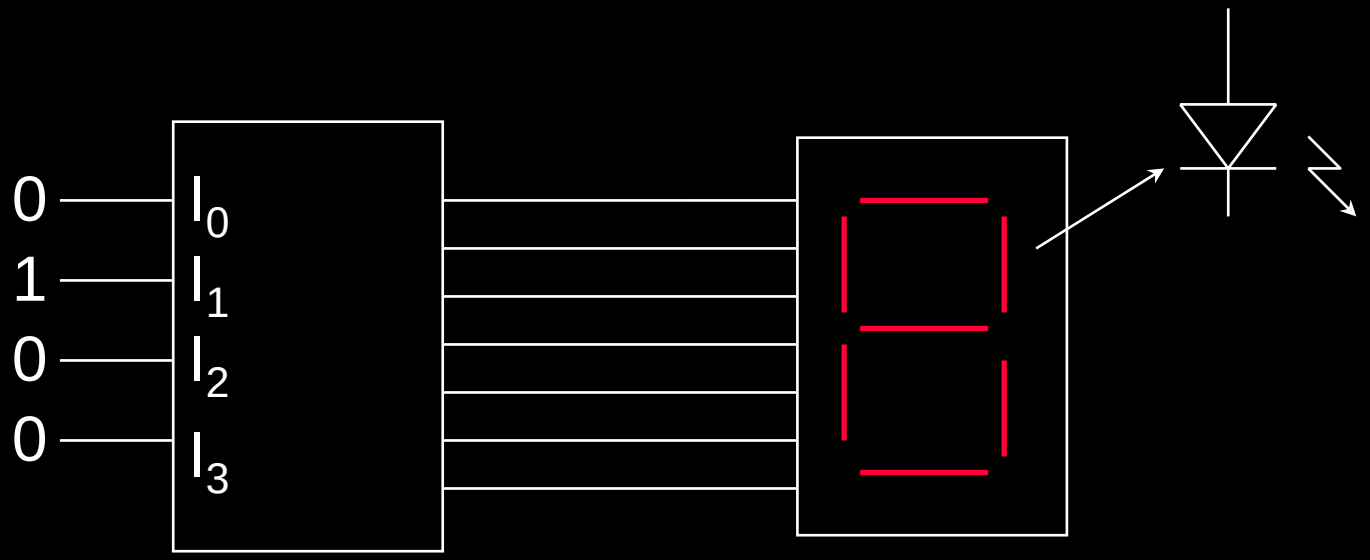
Exemple : adresses pixel / position effective pixel

Transcodeur

Passage d'un code C_1 à un code C_2



Transcodeur : exemple



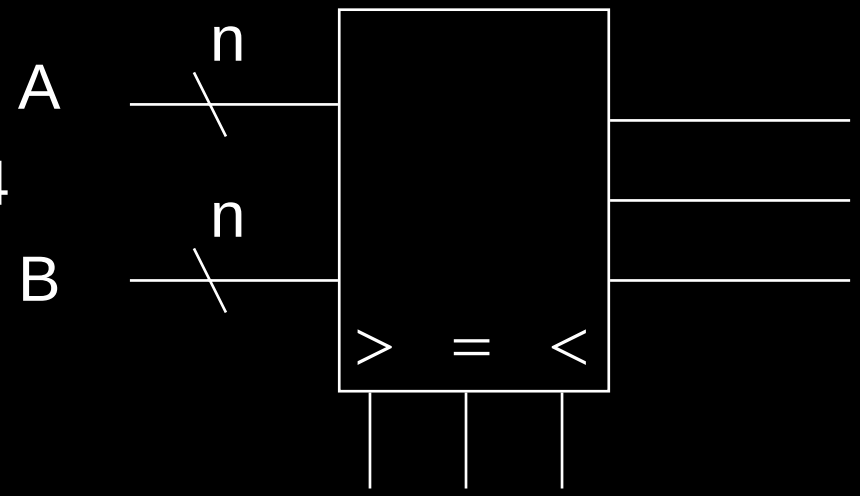
Code binaire 0 à 9

Configuration alimentation des diodes (ou LCD)

Comparateur binaire

Remarque : architecture interne plus tard

classiqu. n=4



S = 1 si A > B

E = 1 si A = B

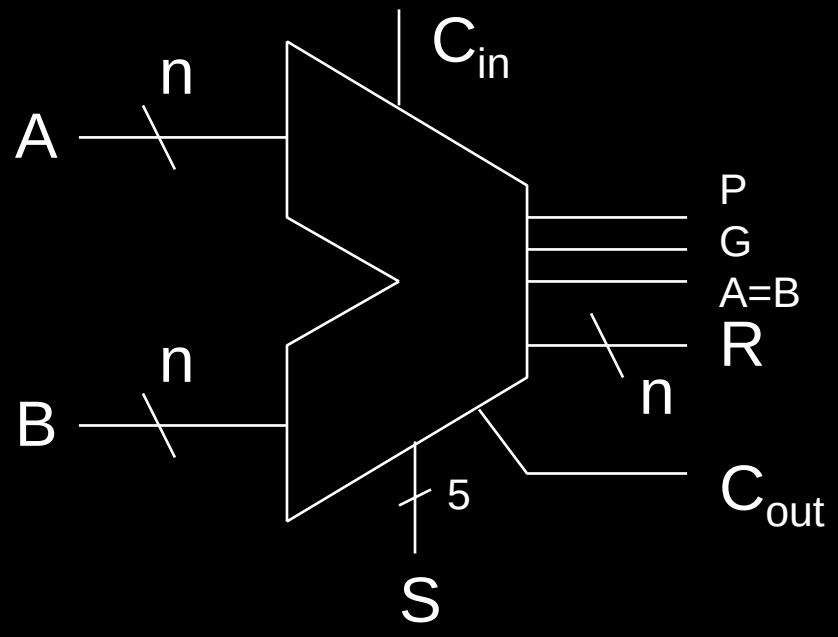
I = 1 si A < B

Entrées de cascadage

ALU (ou UAL)
Unité Arithmétique et Logique

Remarque : architecture interne plus tard

classiqu. n=4



Choix de la fonction (32 cas)

Exemple :

$R = A + \overline{B}$

R = A + B

R = A + B + 1

...

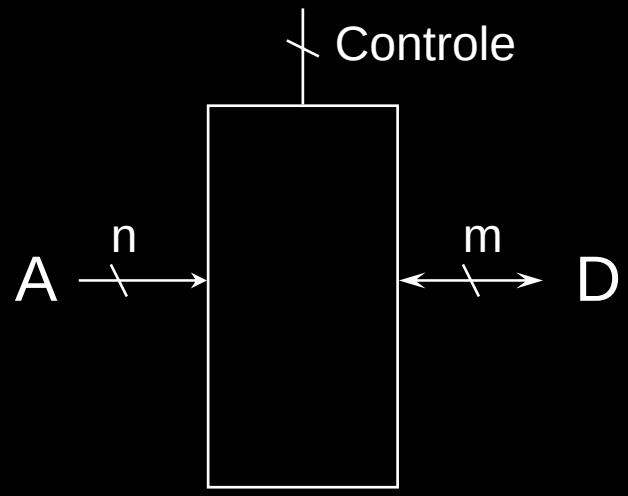
R = A ou B

R = A nand B

...

Mémoire

Remarque (1) : architecture interne en 2ème année
Remarque (2) : la brique de base n'est pas combinatoire !



2^n mots de m bits

Les mémoires peuvent être :
- lecture seule
- écriture une fois/lecture
- écriture lente/lecture
- écriture/lecture

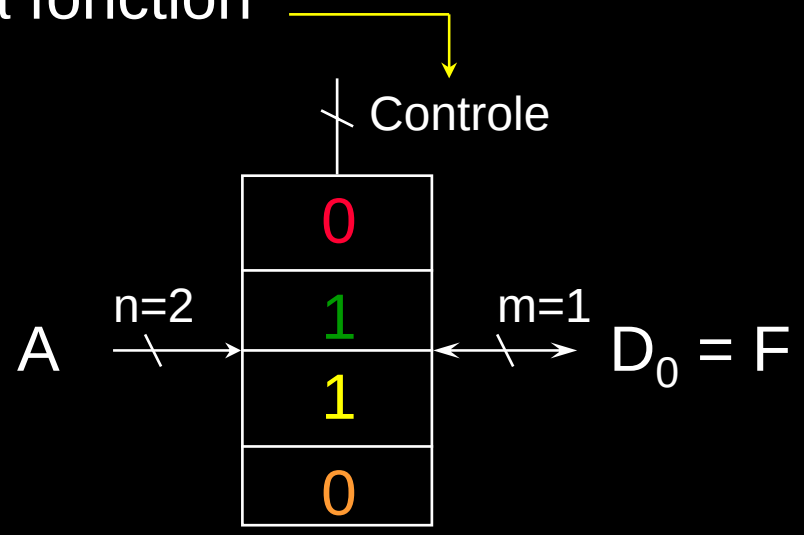
write : D stockée dans M(A)
read : M(A) positionne les fils D

Mémoire : réalisation de fonction

write : stockage de la fonction
read : utilisation

a b	F	
0 0	0	M(00)=0
0 1	1	M(01)=1
1 0	1	M(10)=1
1 1	0	M(11)=0

Il faut
4 mots
de 1 bits

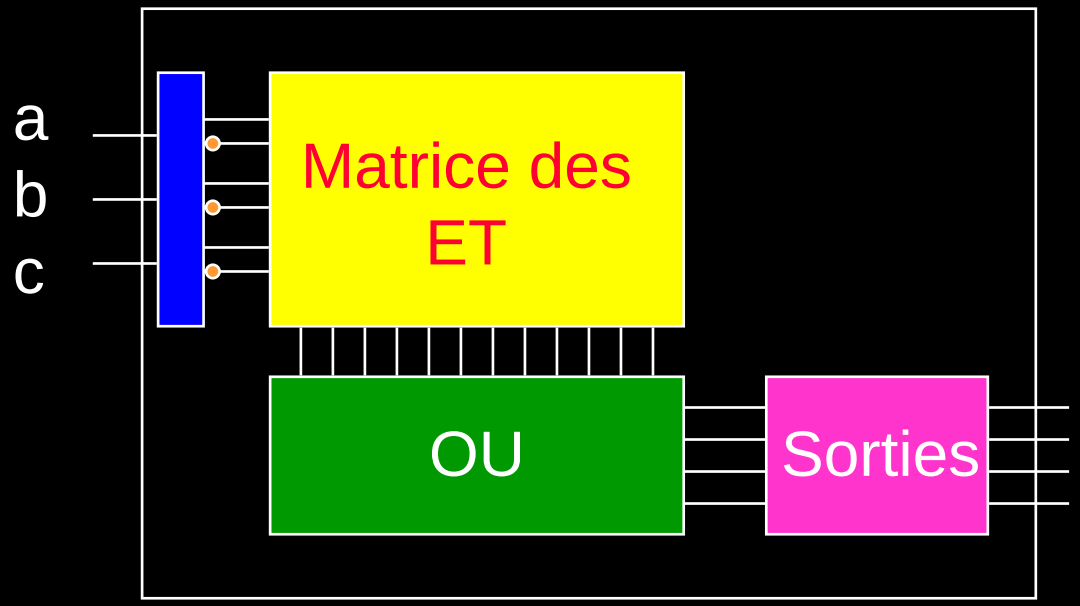


La mémoire est utilisée en LUT (Look Up Table)
C'est souvent une méthode très performante
en vitesse/surface (8 fonctions de 20 variables)

PLD (Composants programmables)

Remarque : architecture interne en 2ème année

Composant acheté sous forme générique et spécialisé par l'utilisateur. Exemple d'une structure 2 couches

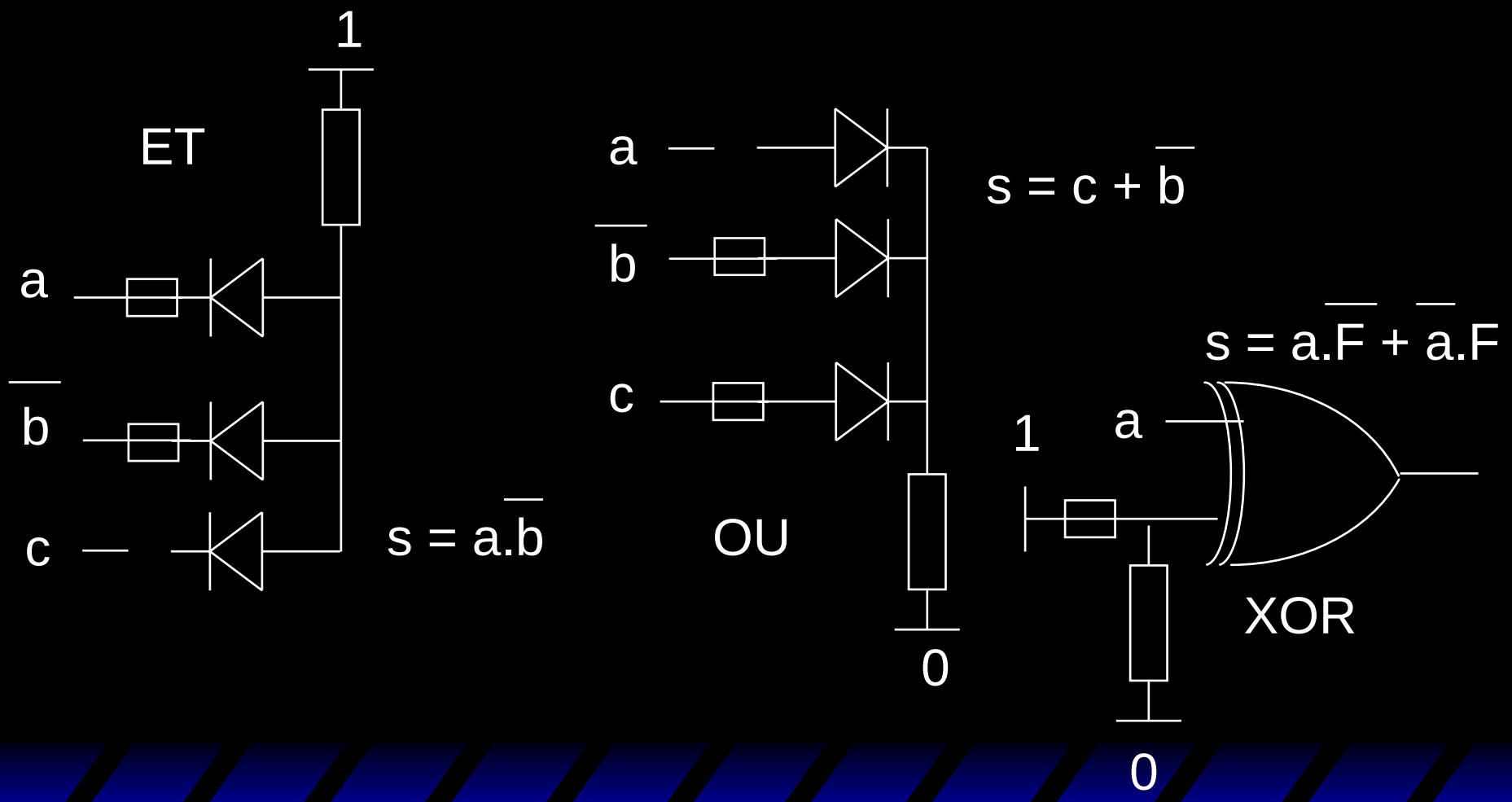


Actuellement :

maxi 200 k portes
qq ns de tps de calcul
qq Ko de RAM
#200 IO

(pas conjointement)

PLD (illustration fonct. interne)



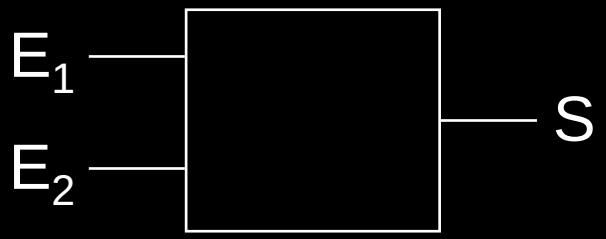
Exercice de cours

Soit deux nombres binaires sur 2 bits A et B, on veut effectuer R = A+B (arithmétique). R est sur 3 bits

- Donner la table de verité de $R_i = F_i (A_1,A_0,B_1,B_0)$
- Donner les formes canoniques de F_i
- Simplifier les équations
- Proposer des chémas à base de portes OU, ET,NON
- Idem avec des NAND seulement
- Idem avec 3 NON et 3 MUX 8 vers 1

Attention : convention utilisée

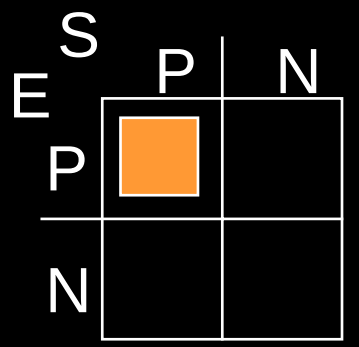
La fonction logique réalisée par un circuit dépend de la convention utilisée.



E_1 \ E_2	0	1
0	0	0
1	0	1

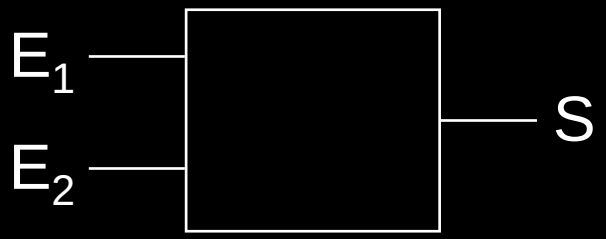
S

Table logique ET



Attention : convention utilisée

La fonction logique réalisée par un circuit dépend de la convention utilisée.



E_1 \ E_2	1	0
1	1	1
0	1	0

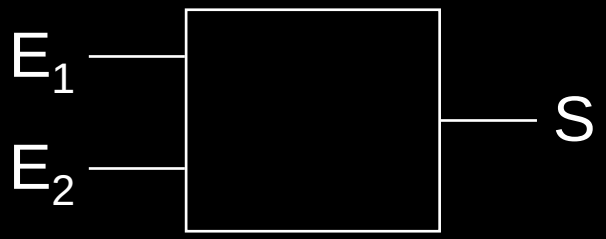
S

Table logique OU

E \ S	P	N
P		
N		■

Attention : convention utilisée

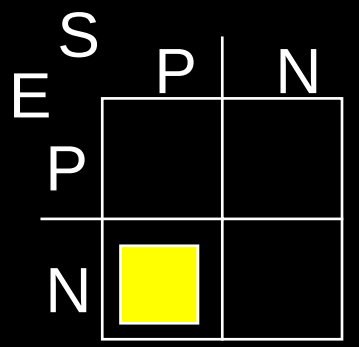
La fonction logique réalisée par un circuit dépend de la convention utilisée.



E_1 \ E_2	1	0
1	0	0
0	0	1

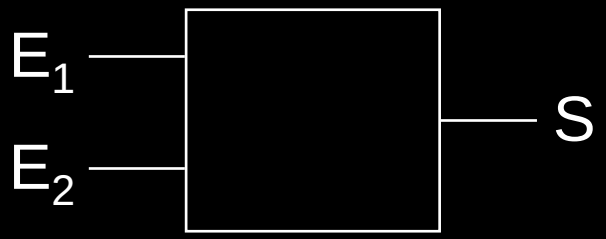
S

Table logique NOR



Attention : convention utilisée

La fonction logique réalisée par un circuit dépend de la convention utilisée.



E_1 \ E_2	0	1
0	1	1
1	1	0

S

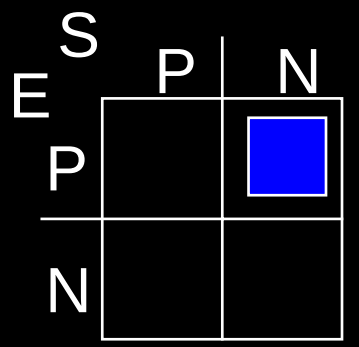


Table logique NAND