## → COURS 2 : LOGIQUE COMBINATOIRE

- Sortie = combinaison « instantanée » des entrées.
- Portes + générales : NAND, NOR, XOR, multiples.
- Fonctions générales du numérique
  - → transcodage (bin → Gray)
  - → MUX N:1
  - → DEMUX 1:N
  - → ARITHMETIQUE (adder, carry)
  - → Priority encoder / Address decoder
- Les délais suivant l'implémentation de la fonction
- Applications
  - → clavier ?
  - → BCD → 7 segment ?



## LOGIQUE COMBINATOIRE VS. SEQUENTIELLE

**►COMBINATOIRE :**

**COMBINAISON DES ENTREES → SORTIE**
**(ce cours)**

**►SEQUENTIEL**

**COMBINAISON DES ENTREES**
**&**
**ENTREE (/ SORTIE / SIGNAUX INTERNES) PRECEDENTS**
**(prochain cours)**
**→ SORTIE**



## FONCTIONS DE BASE

► 3 représentations possibles :

Table de vérité : En donnant toutes les valeurs possibles pour toutes les entrées possibles.

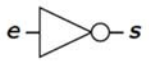
Analytique : En donnant l'équation analytique

Graphique : En utilisant les symboles de fonctions de base

►La plus simple:

L'inverseur (Not)

- La sortie est le complément de l'entrée
- La sortie vaut 1 si et seulement si l'entrée vaut 0

| Symbole | Équation | Table de vérité |
|---|---|---|
|  | $s = \overline{e}$ | |

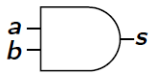
| $e$ | $s$ |
|---|---|
| 0 | 1 |
| 1 | 0 |



## FONCTIONS DE BASE : « ET » (« AND », '&')

- La sortie vaut 1 si et seulement si les deux entrées valent 1
- Si l'une des entrées vaut 0 alors la sortie vaut 0

| Symbole | Équation | Table de vérité |
|---|---|---|
|  | $s = a \cdot b$ | |

| $a$ | $b$ | $s$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

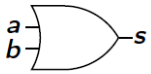


## FONCTIONS DE BASE : « OU » (« OR », '≥1')

- Si l'une des entrées vaut 1 alors la sortie vaut 1
- La sortie vaut 0 si et seulement si les deux entrées valent 0

**Symbole**




**Équation**

$s = a + b$

**Table de vérité**

| $a$ | $b$ | $s$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



## FONCTIONS DE BASE : NAND et NOR

**« NAND »**

- La fonction complémentaire du And

**Symbole court**


**Symbole**


**Équation**

$s = \overline{a \cdot b}$

**Table de vérité**

| $a$ | $b$ | $s$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**« NOR »**

**Symbole court**

- La fonction complémentaire du Or

**Symbole**


**Équation**

$s = \overline{a + b}$

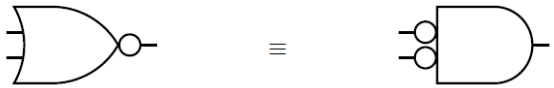
**Table de vérité**

| $a$ | $b$ | $s$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |



## EQUIVALENCE [N]AND / [N]OR

$$\overline{a + b} = \overline{a} \cdot \overline{b}$$



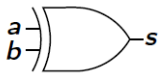
$$\overline{a \cdot b} = \overline{a} + \overline{b}$$



## PORTES PLUS GENERALES : XOR

- La sortie vaut 1 si une et seulement une des entrées est à 1
- La sortie vaut 1 si les deux entrées sont différentes

**Symbole**




**Équation**

$s = a \oplus b$

$s = a \cdot \overline{b} + \overline{a} \cdot b$

**Table de vérité**

| $a$ | $b$ | $s$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## PORTES PLUS GENERALES : PORTES MULTIPLES

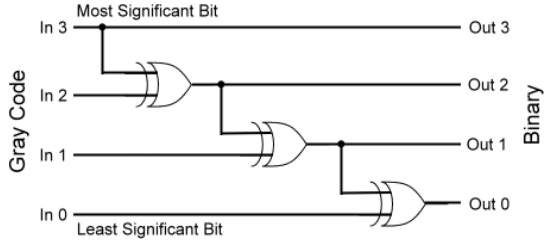
### PORTES N-UPLES (ici porte triple NOR)

*Exo : former un « quad-NAND » avec des NAND élémentaires ?*



## CIRCUITS : TRANSCODAGE

- **Exemple**




| valeur décimale | code binaire réfléchi |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0011 |
| 3 | 0010 |
| 4 | 0110 |
| 5 | 0111 |
| 6 | 0101 |
| 7 | 0100 |
| 8 | 1100 |
| 9 | 1101 |
| 10 | 1111 |
| 11 | 1110 |
| 12 | 1010 |
| 13 | 1011 |
| 14 | 1001 |
| 15 | 1000 |

- **Notez : « LSB », « MSB » (cf. Cours1)**



## CIRCUITS : TRANSCODAGE (2)

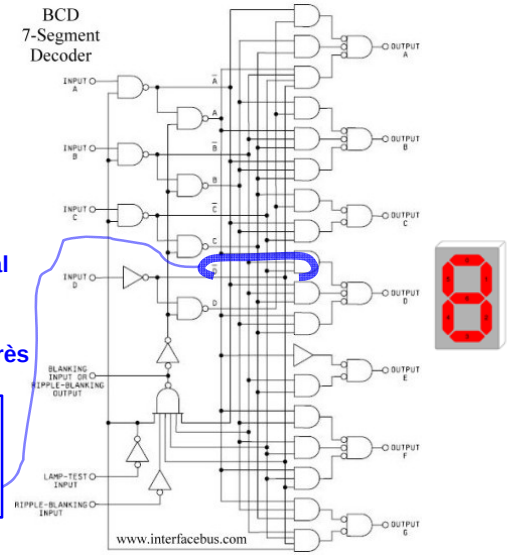
- **En principe : simplification du type diagramme de Karnaugh**

- **Exemple de complexité « moyenne »**

  - **Binaire-Codé-décimal « BCD » → 7-seg**

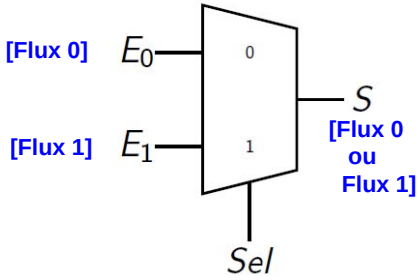
  - **Architecture à peu près lisible :**
    **Explicitation des 8 possibilités intermédiaires (A, $\overline{A}$, B, $\overline{B}$, C, $\overline{C}$, D, $\overline{D}$)**

  - **temps de propag ?**






## CIRCUITS : MULTIPLEXAGE

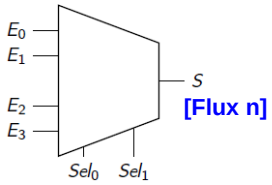



| $Sel$ | $E_1$ | $E_0$ | $S$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

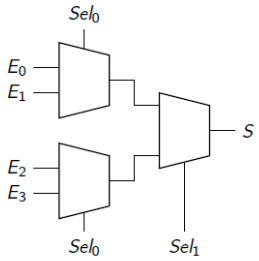
$$S = Sel \cdot E_0 + \overline{Sel} \cdot E_1$$

## CIRCUITS : MULTIPLEXAGE (2)

[Flux 0]
...
...
[Flux 3]




4 entrées ⇒ 2 entrées de sélection




Peut être réalisé à partir de 3 mux 2 vers 1

$$S = \overline{Sel_1} \cdot \overline{Sel_0} \cdot E_0 + \overline{Sel_1} \cdot Sel_0 \cdot E_1 + Sel_1 \cdot \overline{Sel_0} \cdot E_2 + Sel_0 \cdot Sel_1 \cdot E_3;$$



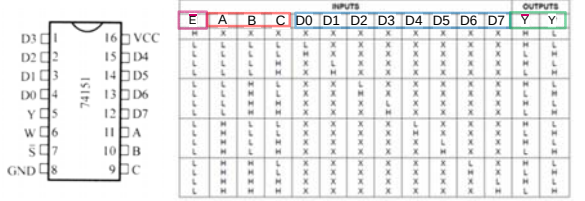
## GENERALISATION / EXEMPLE

Pour un multiplexeur à $n$ entrées ($n = 2^p$ étant une puissance de 2) :

- Il faut $p = \log_2(n)$ entrées de sélection
- Il peut être réalisé avec $n - 1$ multiplexeurs à 2 entrées organisés en $p$ couches

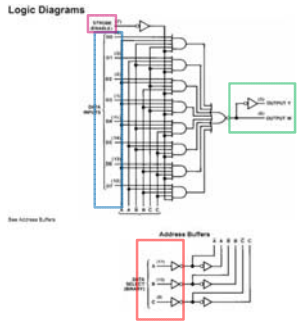


→ **74LS151 1-TO-8 MULTIPLEXER**




FUNCTION TABLE

| INPUTS | | | | | | | | | | | OUTPUTS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E | A | B | C | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | Y | Ȳ |
| H | X | X | X | X | X | X | X | X | X | X | X | L | H |
| L | L | L | L | L | X | X | X | X | X | X | X | L | H |
| L | L | L | L | H | X | X | X | X | X | X | X | H | L |
| L | L | L | H | X | L | X | X | X | X | X | X | L | H |
| L | L | L | H | X | H | X | X | X | X | X | X | H | L |
| L | L | H | L | X | X | L | X | X | X | X | X | L | H |
| L | L | H | L | X | X | H | X | X | X | X | X | H | L |
| L | L | H | H | X | X | X | L | X | X | X | X | L | H |
| L | L | H | H | X | X | X | H | X | X | X | X | H | L |
| L | H | L | L | X | X | X | X | L | X | X | X | L | H |
| L | H | L | L | X | X | X | X | H | X | X | X | H | L |
| L | H | L | H | X | X | X | X | X | L | X | X | L | H |
| L | H | L | H | X | X | X | X | X | H | X | X | H | L |
| L | H | H | L | X | X | X | X | X | X | L | X | L | H |
| L | H | H | L | X | X | X | X | X | X | H | X | H | L |
| L | H | H | H | X | X | X | X | X | X | X | L | L | H |
| L | H | H | H | X | X | X | X | X | X | X | H | H | L |

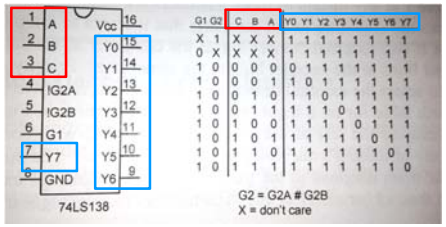


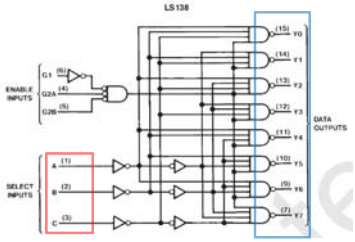



## DEMULTIPLEXEUR

→ **MISE A 0 D'1 BIT PARMI $2^n$**
→ **COMMANDE PAR n bits (allant du LSB au MSB)**

→ **74LS138 1-TO-8 MULTIPLEXER**






## ARITHMETIQUE (1) : ADDITION … binaire 1 bit !

« 1+1=2 »

(« je pose 0 et je retiens 1 »)

$$\begin{array}{r} 0\ 1 \\ +\ 0\ 1 \\ \hline 1\ 0 \end{array}$$

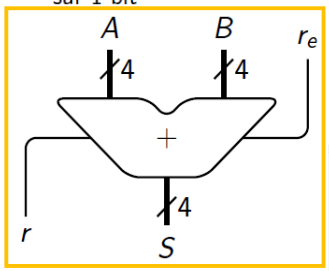
## ARITHMETIQUE (2) : ADDITIONNEUR 4 bits

### Exemple

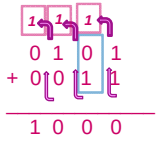
Faire une addition en binaire sur 4 bits... [ *avec retenue et report* ]

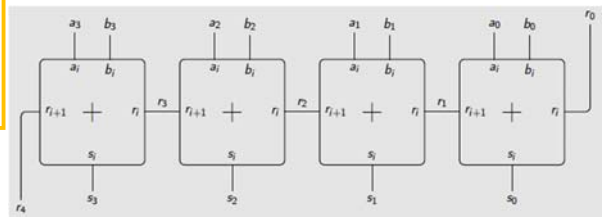
### Décomposition de l'addition

L'addition peut être décomposée en plusieurs additions élémentaires sur 1 bit




5+3=8



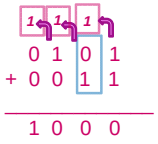


## ARITHMETIQUE (3) : ADDITION 1 bit = logique

### Arithmétiquement

$$a_i + b_i + r_i = 2 \cdot r_{i+1} + s_i$$

Retenues (« carry »)




### Table de vérité

| $a_i$ | $b_i$ | $r_i$ | $r_{i+1}$ | $s_i$ | Décimal |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 2 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 2 |
| 1 | 1 | 0 | 1 | 0 | 2 |
| 1 | 1 | 1 | 1 | 1 | 3 |

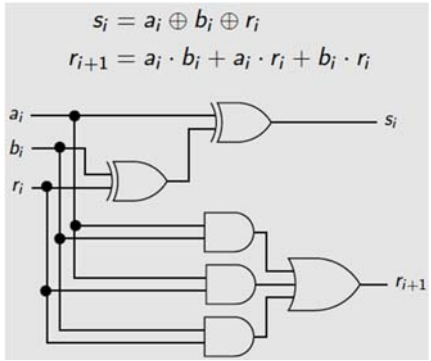




## ARITHMETIQUE (4) : ADDITION 1 bit explicite

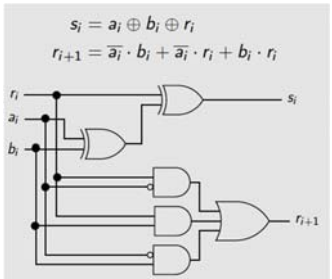
→ ADDITIONNEUR 1 BIT

$$s_i = a_i \oplus b_i \oplus r_i$$
$$r_{i+1} = a_i \cdot b_i + a_i \cdot r_i + b_i \cdot r_i$$




→ QUE FAIT CECI A VOTRE AVIS ?

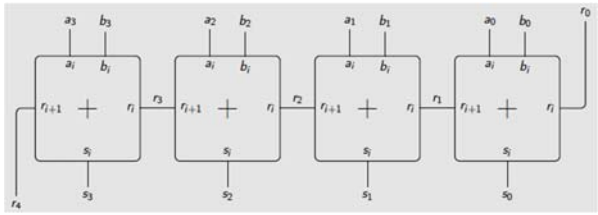
$$s_i = a_i \oplus b_i \oplus r_i$$
$$r_{i+1} = \overline{a_i} \cdot b_i + \overline{a_i} \cdot r_i + b_i \cdot r_i$$





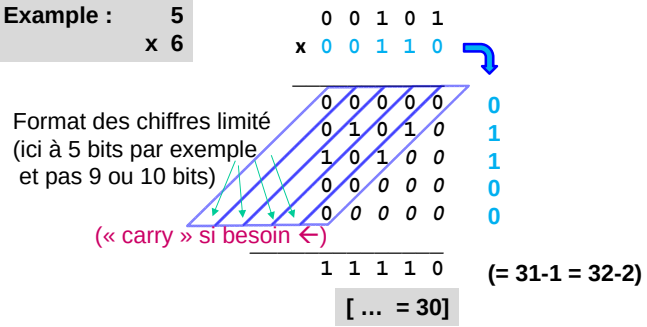

## ARITHMETIQUE (5) : TEMPS DE PROPAGATION

→ Quelle est la loi d'échelle du temps de propagation ?



## ARITHMETIQUE (6) : MULTIPLICATION ?

**Example : 5 x 6**




**(= 31-1 = 32-2)**

**[ … = 30]**

→ N x N se fait avec N décalages et {0 à N} additions,
suivant les bits à 1 de l'opérande



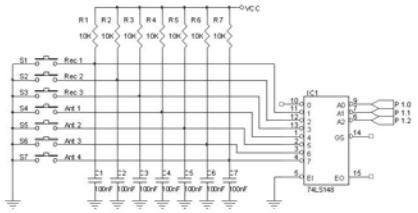
## COMBINATOIRE « UTILE »

### PRIORITY ENCODER (74LS147/148)

FUNCTION TABLE – '147, 'LS147

| INPUTS | | | | | | | | | OUTPUTS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | D | C | B | A |
| H | H | H | H | H | H | H | H | H | H | H | H | H |
| X | X | X | X | X | X | X | X | L | L | H | H | L |
| X | X | X | X | X | X | X | L | H | L | H | H | H |
| X | X | X | X | X | X | L | H | H | H | L | L | L |
| X | X | X | X | X | L | H | H | H | H | L | L | H |
| X | X | X | X | L | H | H | H | H | H | L | H | L |
| X | X | X | L | H | H | H | H | H | H | L | H | H |
| X | X | L | H | H | H | H | H | H | H | H | L | L |
| X | L | H | H | H | H | H | H | H | H | H | L | H |
| L | H | H | H | H | H | H | H | H | H | H | H | L |

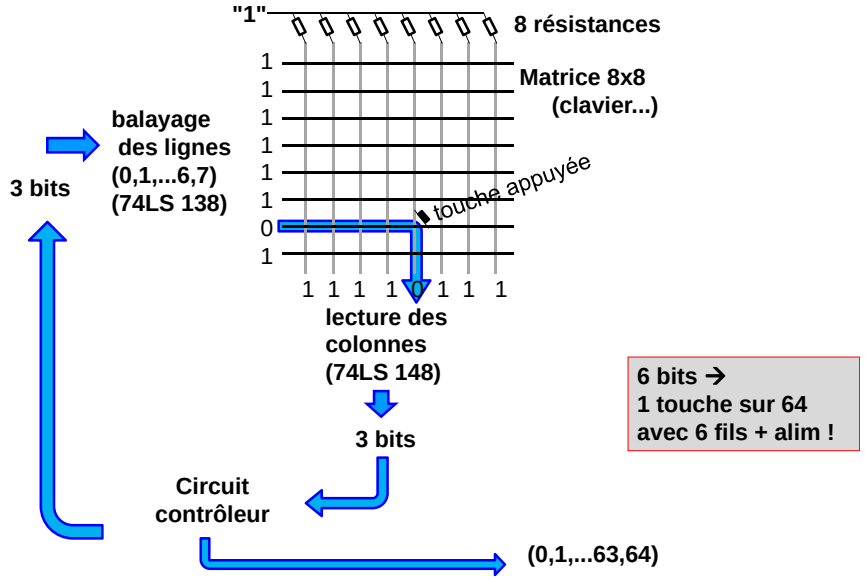
H = high logic level, L = low logic level, X = irrelevant



**Clavier élémentaire**



## COMBINATOIRE « UTILE » : clavier matricé
### avec un peu de séquentiel caché




**6 bits →**
**1 touche sur 64**
**avec 6 fils + alim !**



## Exercices possibles

→ Analyse du décodeur « BCD » → 7 segment

→ Décodeur code « ASCII-ADN » {ACGT} → binaire
→ test de panne sur liaison parallèle par le « U »
→ Architectures d'additionneurs avec court t_prop

→ circuit d'arrêt d'une machine (à café)
en fonction de la « logique » positive ou négative des N capteurs
→ code « Gray inverse » (2 bits sur 3 changent)

# CE QU'IL FAUT RETENIR

**Cours 2 (LOGIQUE COMBINATOIRE):**

→ Fonctions logiques: (**N**)**AND** (0 est roi), (**N**)**OR** (1 est roi), (**N**)**XOR** (sym.)
**CONNAITRE LES TABLES DE VERITE**

→ Portes N-uples (« quadruple NAND »)

→ Transcodage: possibilité d'expliciter les états intermédiaires

→ Comprendre ce que fait MUX [@q bits](N → 1) et DEMUX [@q bits](1→N)
[ $N=2^q$ ]

→ L'arithmétique se réduit à de la logique
additionneur 1 bit + carry → additionneur quelconque
→ Multiplication = qqs additions
→ Temps de propag … proportionnel à q ? [ $N=2^q$ ]

→ Fonctions « utiles » : détection du +bas bit à 1 parmi N (priority encoder)



# AUTO-TESTS

→ Table de vérité de AND OR XOR, NAND NOR.
Lesquelles ont plus de 0, plus de 1, autant de 0 que de 1

→ NAND et NOR multiples (détecter un sous-ensemble de caractères ASCII p ex)
{Reconnaitre des sous-ensembles de caractères avec 1 à 5 portes (NAND, NOR).
(Cours 1 et Cours 2)}

→Multiplexeur N → 1, Demux 1→ N. Combien de bits d'adresse ?

→ Arithmétique : additionneur avec retenue (table): table de vérité de la retenue ?
multiplication binaire (principe): expliquer taille registres ?

→ Encodeur de priorité : quel rapport avec le multiplexeur ?

→ *Calcul global* : combien d'addition binaires paar seconde pour gérer un flux video
{{ 1 Mpix/s, 3 couleurs(RVB), 8 bits/couleur, }}
si l'ajustement de couleur est fait sous la forme d'une multiplication d'un facteur au choix par canal (sans pb de saturation)