

Cours 2 Architectures & Systèmes 1

Jalil Boukhobza

LC 206

boukhobza@univ-brest.fr

1

Chemin de données

Font l'objet de ce cours:

- Représentation des nombres
- Les portes logiques et circuits combinatoires
- Le traitement de quelques opérations arithmétiques (UAL)
- Circuits séquentiels, registres et mémorisation
- l'interconnexion des différents éléments du chemin de données

Définition: ensemble des composants requis pour l'**exécution** des diverses instructions (banc de registres, PC, UAL, etc.)

2

Représentation et précision finie

- Représentation en précision finie (quantité limitée de nombres représentables)
 - Problème de représentation: ex 1/3 n'est pas représentable
 - Les propriétés sur les nombres sont différentes en précision finie

$$(A+B) - C = A + (B-C) ??$$

Exemple:

- **Loi associative:** $a+(b-c) \Leftrightarrow (a+b)-c$
Entiers sur 3 chiffres: $a=700, b=400, c=300$
- **Loi distributive:** $a*(b-c) \Leftrightarrow (a*b)-(a*c)$
Entiers sur 3 chiffres: $a=5, b=210, c=100$

Jalil Boukhobza

3

Les bases utilisées en informatique

- La base 10 bien sûr ! Sauf exception, les programmeurs comptent en base 10 comme le quidam ordinaire.
- La base 2 : les opérateurs de calcul sont souvent prévus pour calculer dans cette base, car leur réalisation en est simplifiée.
- La base 16 ou hexadécimal : elle est utilisée lorsque le programmeur doit raisonner en base 2. Les conversions de la base hexadécimal vers la base 2 se calculent très facilement de tête.

Représentation binaire des nombres

- nombre **positif** : (conversion en base 2)

$$A = \sum_{i=0}^{n-1} a_i \cdot 2^i$$

- nombre **néгатif** :

- représentation **signe-valeur absolue**:

$$A = \sum_{i=0}^{n-2} a_i \cdot 2^i \quad \text{si } a_{n-1} = 0$$

$$A = -\sum_{i=0}^{n-2} a_i \cdot 2^i \quad \text{si } a_{n-1} = 1$$

- représentation **complément à 2** :

$$A = -2^{n-1} \cdot a_{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i$$

Jalil Boukhobza

5

Conversion de la base 10 à une base B

- On divise successivement le nombre à convertir par la base B, tant que le résultat n'est pas nul.
- Les **restes** de chacune des divisions forment les chiffres du nombre converti dans la base d'arrivée, du poids faible au poids fort.
- Exemple: base 2:

$$\begin{array}{r}
 54 \mid 2 \\
 \hline
 0 \mid 27 \mid 2 \\
 \hline
 1 \mid 13 \mid 2 \\
 \hline
 1 \mid 6 \mid 2 \\
 \hline
 0 \mid 3 \mid 2 \\
 \hline
 1 \mid 1 \mid 2 \\
 \hline
 1 \mid 0
 \end{array}
 \quad (54)_{10} = (110110)_2$$

Représentation binaire (2)

Représentation décimale	Représentation signe et valeur absolue	Représentation complément à 1	Représentation complément à 2
+4	-	-	-
+3	011	011	011
+2	010	010	010
+1	001	001	001
+0	000	000	000
-0	100	111	000
-1	101	110	111
-2	110	101	110
-3	111	100	101
-4	-	-	100

7

Caractéristiques principales du complément à 2

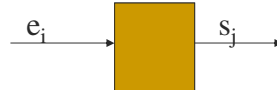
- Intervalle de représentation: $-2^{n-1} \dots 2^{n-1}-1$
- Nombre de représentation de zéro: **une seule**
- Négation: $-A = \overline{A} + 1$
- Augmentation du nombre de bits: extension par le bit du signe
- Dépassement de capacité: résultat prend le signe opposé de ce qu'il devrait prendre
- Soustraction: $A-B = A + (-B)$

Jalil Boukhobza

8

Circuits logiques

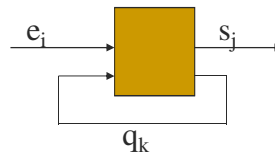
- **Circuit logique combinatoire** : l'état des sorties s_j dépend uniquement de l'état courant des entrées e_i .
 $s_j = f(\dots, e_i, \dots)$



- **Circuit logique séquentiel** : l'état des sorties s_j dépend de l'état courant des entrées e_i et de l'état (\dots, q_k, \dots) de la machine

$$s_j = f(\dots, e_i, \dots, q_k, \dots)$$

$$q_k = g(\dots, e_i, \dots, q_k, \dots)$$



9

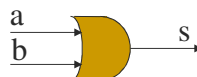
L'algèbre de Boole

Opérateur OU

- Identité : $0+a=a$
- Nullité : $1+a=1$
- Idempotence : $a+a=a$
- Commutativité : $a+b=b+a$
- Associativité : $(a+b)+c=a+(b+c)$

Porte OU (OR)

$$s = a + b$$



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

10

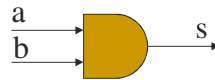
L'algèbre de Boole

Opérateur ET

- Identité : $1.a=a$
- Nullité : $0.a=0$
- Idempotence : $a.a=a$
- Commutativité : $a.b=b.a$
- Associativité : $(a.b).c=a.(b.c)$

Porte ET (AND)

$$s=a.b$$



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

11

L'algèbre de Boole

Lois de composition

- Distributivité
 $a.(b+c)=a.b+a.c$ $a+(b.c)=(a+b).(a+c)$
- Absorption
 $a.(a+b)=a$ $a+a.b=a$
- De Morgan
 $\overline{a.b}=\overline{a}+\overline{b}$ $\overline{a+b}=\overline{a}.\overline{b}$

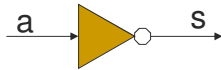
12

Logique combinatoire

Portes logiques unaires

Porte NON (NOT) ou inverseur

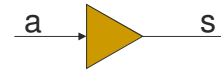
$$s = \bar{a}$$



a	s
0	1
1	0

Porte OUI ou répéteur

$$s = a$$



a	s
0	0
1	1

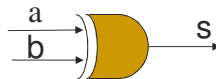
13

Logique combinatoire

Portes logiques binaires (suite)

Porte OU exclusif (XOR)

$$s = a \oplus b$$



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

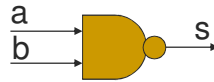
$$s = a \oplus b = \bar{a}.b + a.\bar{b}$$

14

Logique combinatoire
Portes logiques binaires (suite)

Porte NON-ET (NAND)

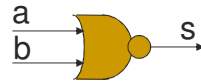
$$s = \overline{a \cdot b}$$



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

Porte NON-OU (NOR)

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



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

15

L'addition en nombre non signés

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

Retenue sortante

Étude de l'architecture de l'additionneur à retenue propagée.

16

L'additionneur à retenue propagée

- Addition sur 1 bit, somme variant de la valeur 0 à 2:

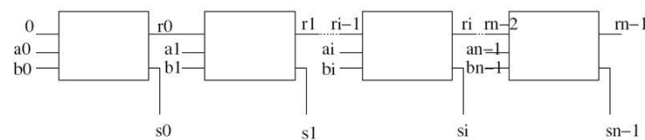
$$\begin{aligned} \text{somme}(\text{poids faible}) &= a \oplus b = s(0) \\ \text{somme}(\text{poids fort}) &= a.b = \text{retenue} = r(0) \end{aligned}$$

- Addition sur n bits:

$$\begin{aligned} s(i) &= a(i) \oplus b(i) \oplus r(i-1) \\ r(i) &= a(i).b(i) + r(i-1).(a(i) \oplus b(i)) \\ &\text{avec } i \in [0..n-1[\end{aligned}$$

17

Additionneur complet (4 bits) à retenue propagée



Additionneur à retenue propagée

Problème de **temps de propagation**: circuit lent car il dépend du temps de propagation de la retenue.

18

Addition avec des nombres signés

- En complément à 2 : pas de traitement particulier pour le bit de signe.

$$\begin{array}{r}
 1\ 1\ 0\ 0\ (-4) \\
 1\ 1\ 0\ 1\ (-3) \\
 \hline
 1\ 1\ 0\ 0\ 1\ (-7)
 \end{array}$$

La retenue du dernier étage *n'est pas synonyme de dépassement de capacité.*

$$\begin{array}{r}
 0\ 1\ 0\ 0\ (4) \\
 0\ 1\ 0\ 1\ (5) \\
 \hline
 0\ 1\ 0\ 0\ 1\ (-7)
 \end{array}$$

Pas de dépassement de capacité si $x + y \in [-2^{n-1} - 1, 2^{n-1}]$

19

Cas de débordement

- Addition de nombres positifs ($\text{res} \geq 2^{n-1}$)

$$\begin{array}{r}
 0\ 1\ 1\ 1\ (7) \\
 0\ 0\ 0\ 1\ (1) \\
 \hline
 0\ 1\ 0\ 0\ 0\ (-8)
 \end{array}$$

- Addition de nombres négatifs ($\text{res} \leq -2^{n-1}-1$)

$$\begin{array}{r}
 1\ 1\ 0\ 0\ (-4) \\
 1\ 0\ 1\ 0\ (-6) \\
 \hline
 1\ 0\ 1\ 1\ 0\ (6)
 \end{array}$$

- Mise en équation du débordement (overflow):

$$\text{overflow} = a.b.\bar{s} + \bar{a}.\bar{b}.s$$

20

Multiplication des nombres non signés

- A et B opérandes sur N bits

$$A = A_{n-1} * 2^{n-1} + A_{n-2} * 2^{n-2} + A_{n-3} * 2^{n-3} + \dots + A_1 * 2^1 + A_0 * 2^0$$

$$B = B_{n-1} * 2^{n-1} + B_{n-2} * 2^{n-2} + B_{n-3} * 2^{n-3} + \dots + B_1 * 2^1 + B_0 * 2^0$$

- Le produit A.B nécessite $2n$ bits

$$A.B = A * B_{n-1} * 2^{n-1} + A * B_{n-2} * 2^{n-2} + \dots + A * B_1 * 2^1 + A * B_0 * 2^0$$

- A.B est une somme de produits partiels P_i :

$$P_i = A.B_i * 2^i$$

P_i est obtenu par décalage si $B_i=1$.

21

Exemple

$$\begin{array}{r}
 1\ 1\ 0\ 1\ (13) \\
 1\ 0\ 0\ 1\ (9) \\
 \hline
 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ (P0) \\
 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \bullet\ (P1) \\
 0\ 0\ 0\ 0\ 0\ 0\ 0\ \bullet\ \bullet\ (P2) \\
 0\ 1\ 1\ 0\ 1\ \bullet\ \bullet\ \bullet\ (P3) \\
 \hline
 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ (P3)
 \end{array}$$

Pour des multiplications ou divisions de nombre non signés par 2^n , on procède par **décalages**;

- vers la **gauche** pour la **multiplication**
- vers la **droite** pour la **division**

22

[Multiplication de nombres signés]

- En complément à 2, B s'écrit:

$$B = -B_{n-1} * 2^{n-1} + B_{n-2} * 2^{n-2} + \dots + B_1 * 2^1 + B_0 * 2^0$$

- Le produit A.B s'écrit:

$$A.B = -A * B_{n-1} * 2^{n-1} + A * B_{n-2} * 2^{n-2} + \dots + A * B_1 * 2^1 + A * B_0 * 2^0$$

même principe mais une soustraction à la place d'une addition.

23

[Exemple]

$$\begin{array}{r}
 1 \ 1 \ 0 \ 1 \ (-3) \\
 0 \ 1 \ 0 \ 1 \ (5) \\
 \hline
 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ (P0) \\
 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \bullet \ (P1) \\
 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ \bullet \ \bullet \ (P2) \\
 0 \ 0 \ 0 \ 0 \ 0 \ \bullet \ \bullet \ \bullet \ (P3) \\
 \hline
 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ (-15)
 \end{array}$$

24

[Autre exemple]

$$\begin{array}{r}
 1\ 1\ 0\ 1\ (-3) \\
 1\ 1\ 0\ 1\ (-3) \\
 \hline
 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ (P0) \\
 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \bullet\ (P1) \\
 1\ 1\ 1\ 1\ 0\ 1\ \bullet\ \bullet\ (P2) \\
 0\ 0\ 0\ 1\ 1\ \bullet\ \bullet\ \bullet\ (-P3) \\
 \hline
 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ (9)
 \end{array}$$

25

[Conversion de nombres flottants (Exemple)]

-54,625

$$54 = (110110)_2$$

$$0,625 = (0,101)_2 \dots 0,625 * 2 = 1,25$$

$$0,25 * 2 = 0,5$$

$$0,5 * 2 = 1,0 \rightarrow \text{STOP}$$

$$54,625 = (110110,101)_2 = 110110101 * 2^5$$

$$\text{biais} = 2^{(\text{nombre de bits pour l'exposant biaisé} - 1)} - 1 = 2^{(8-1)} - 1 = 127$$

$$\text{Exposant biaisé} = \text{exposant réel} + \text{biais} = 5 + 127 = 132$$

$$= (10000100)_2$$

Résultat:

$$-54,625 = (1\ 10000100\ 101101010000000000000000)_2$$

26

[Conversion de nombres flottants]

Notation exponentielle pour la représentation des flottants

Mantisse * Base ^{Exposant}

- **Mantisse** : précision du nombre (m). La virgule se trouve par défaut après le premier bit le plus à gauche du nombre, en général pas représenté car est égal à la valeur 1 (cas des nombres normalisés).
- **Exposant** : ordre de grandeur et place de la virgule dans la mantisse (e). L'exposant est donné en représentation **baisée afin qu'il soit toujours** positif. Un exposant sur **8 bits** varie avec la représentation biaisée de 0..255. Pour obtenir la vraie valeur du biais, on doit **soustraire 127**, l'espace de variation est alors de -127 à +128.

On pose alors $e=[E]-127$ et $m = (1 + \frac{[M]}{2^{23}})$ avec E et M correspondant à la représentation du flottant (+bit du signe).

27

[Format IEEE 754]

Simple précision (double précision)

- s: 1 bit de signe
- E: 8 bits pour l'exposant (11 bits)
- M: 23 bits pour la mantisse (52 bits)

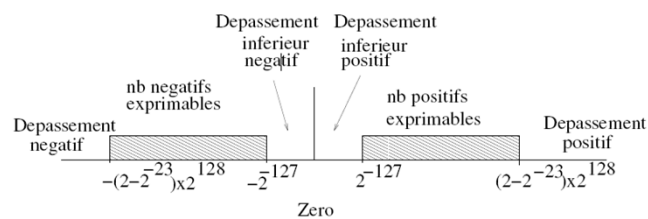
un flottant normalisé s' évalue de la manière suivante:

Par exemple: $(-1)^s * (1 + \frac{[M]}{2^{23}}) * 2^{[E]-127}$
 $(1.1010001)_2 * 2^{(10100)} = (1 + 1/2 + 1/2^3 + 1/2^7) * 2^{20} = 1.6328125 * 2^{20}$
 Biais= 20+127=147= (10010011)₂
 Est représenté par:
 0 10010011 101000100000000000000000

28

Espace de nombres représentables

- Sur 32 bits, nous avons 2^{32} valeurs représentables.
- En notation complément à 2 tous les entiers de -2^{31} à $2^{31} - 1$.
- En représentation flottante, les nombres ne sont pas régulièrement espacés



29

Codages spéciaux

- **Zéros** : pas de représentation normalisée
 $e=-127, m=0 \Rightarrow E=00000000, M=0\dots0$
 2 représentations du zéro (signe)
- **Infinis** : $-\infty$ et $+\infty$
 $e=128, m=0, \Rightarrow E=11111111, M=0\dots0$
- Représentation **dénormalisée** pour $e=-127$ et $m \neq 0$
- **NaN** (Not a Number) utilisé pour représenter les erreurs
 $e=128$ et $m \neq 0$.

30

[Les opérations en flottant]

Déroulement

1. Aligner les exposants sur l'exposant du plus grand nombre
2. Opération
3. Normalisation du résultat → pb d'arrondi

31

[Indicateurs fournis par les opérateurs]

Indicateurs supplémentaires fournis par les opérateurs:

- **I** : résultat inexact, mantisse arrondie
- **V** : opération invalide,
- **Z** : division par zéro
- **O** : Overflow dépassement de capacité, supérieur à $1,1...1 * 2^{127}$ en valeur absolue
- **U** : Underflow dépassement de capacité, inférieur à 2^{-126} en valeur absolue

32

[Unité Arithmétique et Logique (UAL)]

Une UAL peut être définie pour réaliser:

- des opérations arithmétiques sur des entiers positifs, négatifs ou flottants
- des opérations de comparaison
- de décalages
- des opérations logiques

Les drapeaux générés par l'UAL le sont généralement pour le dépassement de capacité, pour le cas de résultat nul, pour le cas de résultat négatif ou pour le cas de résultats erronés dus à des erreurs d'arrondis ou à des opérations non valides.

33

[Éléments de mémorisation]

- Les **registres** sont des éléments *séquentiels* qui sont activés par une *horloge* et permettent de mémoriser des valeurs (**état interne**).
- Les registres sont constitués d'éléments de mémorisation élémentaires généralement des **bascules D** ou des **latch D**.
- **La bascule D** est synchronisée sur **front d'horloge** tandis que **le latch D** est synchronisée sur **niveau d'horloge** (circuit plutôt asynchrone dans lequel on ne maîtrise pas complètement les instants de changement de la sortie).

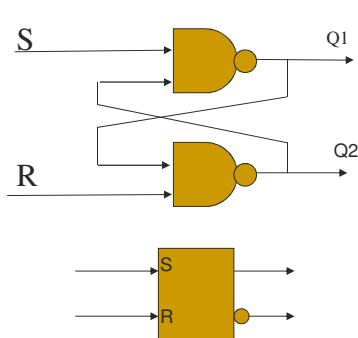
34

[Bascule RS]

- Une bascule RS (Reset-Set) est un circuit séquentiel asynchrone.

$$Q1 = S \cdot \overline{Q2}$$

$$Q2 = \overline{R} \cdot Q1$$

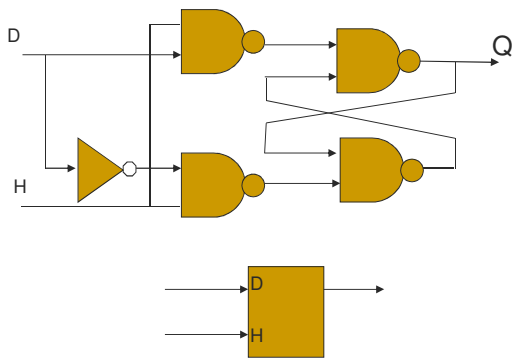


R	S	Q1 _(t-1)	Q2 _(t-1)	Q1	Q2
1	1	0	1	0	1
1	1	1	0	1	0
0	1	X	X	0	1
1	0	X	X	1	0
0	0	X	X	1	1

35

[latch D (verrou)]

- Un latch D (ou verrou) est un circuit séquentiel synchrone.
- La sortie Q recopie l'entrée D lorsque le signal d'horloge H est actif.

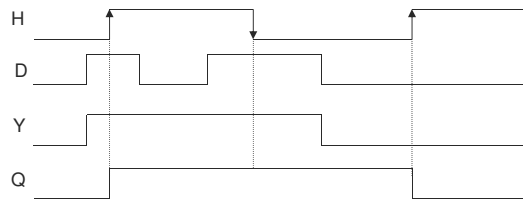
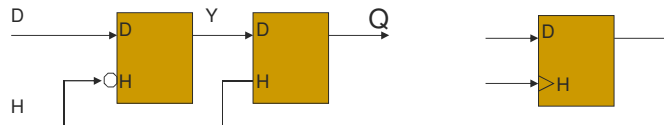


D	H	Q
0	0	Q _(t-1)
1	0	Q _(t-1)
0	1	0
1	1	1

36

Bascule D (flip-flop)

- Une bascule D (flip-flop) est un circuit séquentiel **synchrone**.
- La sortie Q recopie l'entrée D lorsque le signal d'horloge H passe de l'état 0 à l'état 1 (front montant).
- Structure maître-esclave



37

Table de vérité des bascules/latch D

Entrée D	Horloge	etat interne	sortie D flip-flop	sortie D latch
0	1	0	0 (mémorisation)	0 (mise à 0)
0	1	1	1 (mémorisation)	0 (mise à 0)
1	1	0	0 (mémorisation)	1 (mise à 1)
1	1	1	1 (mémorisation)	1 (mise à 1)
0	↑	0	0 (mise à 0)	0 (mémorisation)
0	↑	1	0 (mise à 0)	1 (mémorisation)
1	↑	0	1 (mise à 1)	0 (mémorisation)
1	↑	1	1 (mise à 1)	1 (mémorisation)
0	↓,0	0	0(mémorisation)	0 (mémorisation)
0	↓,0	1	1(mémorisation)	1 (mémorisation)
1	↓,0	0	0(mémorisation)	0 (mémorisation)
1	↓,0	1	1(mémorisation)	1 (mémorisation)

38

Quelques variantes de bascules D

- Bascule D avec autorisation de chargement **synchrone**

CLK	EN	état futur Q^+	fonction réalisée
0,1,↓	X	Q	Maintien
↑	0	Q	Maintien
↑	1	D	Recopie de D

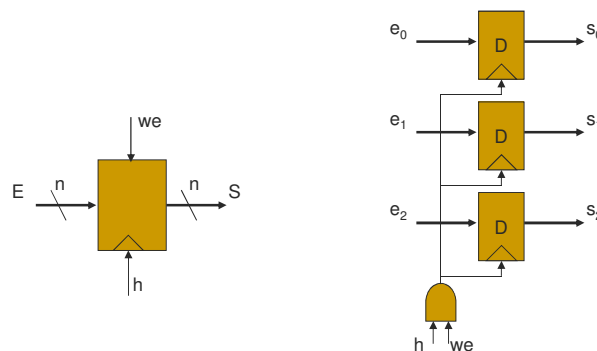
- Bascule D avec reset **asynchrone**

CLK	reset	état futur Q^+	fonction réalisée
0,1,↓	1	0	Mise à 0
↑	1	0	Mise à 0 prioritaire
↑	0	D	Recopie de D

39

Registres

- N bascules D sont associées pour enregistrer un mot (...de N bits).
- Une entrée « **write enable** », optionnelle, permet de valider ou non le signal d'horloge



40

Variantes au niveau du registre

- Synchronisation de données en mode parallèle
Après une impulsion d'horloge $Q_i^+ = D_i$
- Registre à décalage (avec bascules actives sur front uniquement)

- Décalage à droite $Q_i = D_{i+1}$

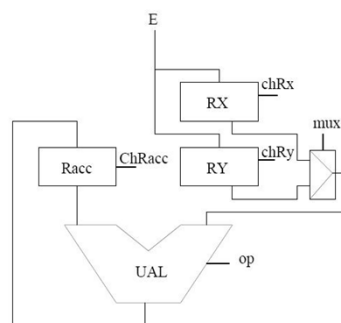
ESG	Q ₀	Q ₁	Q ₂	Q ₃	Q ₄
t	1	0	1	1	0
t+1	ESG	1	0	1	1

- Décalage à gauche $Q_{i+1} = D_i$

Q ₀	Q ₁	Q ₂	Q ₃	Q ₄	ESD
1	0	1	1	0	t
0	1	1	0	ESD	t+1

41

Éléments pour l'interconnexion des composants



- Les aiguillages peuvent être réalisés en utilisant des multiplexeurs ou des bus 3 états.

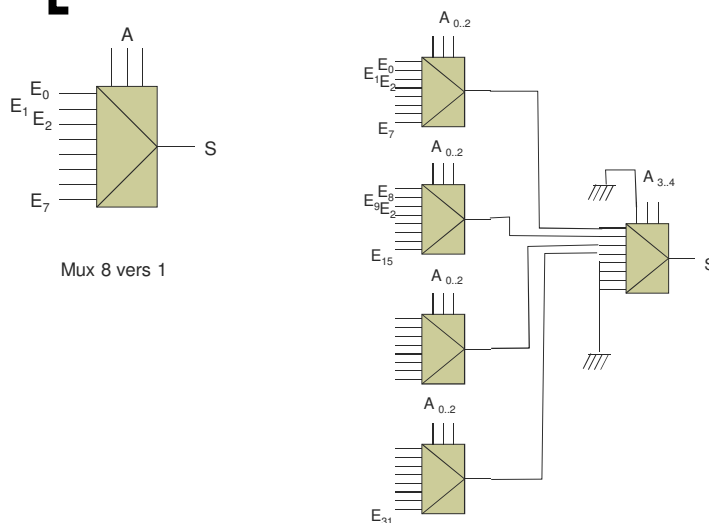
42

Les multiplexeurs

- Circuit qui accepte plusieurs signaux logiques en entrées (données) et n'autorise qu'un seul signal de sortie. L'entrée transférée en sortie est donnée par l'adressage.
- Équation générale : $S = \sum_{j=0}^{j=n} E_j \cdot (A = j)$
Possibilité d'avoir des entrées de validation.
- Exemple : Multiplexeur 8 vers 1 et réalisation d'un multiplexeur 32 vers 1 de manière hiérarchique.
- Sur le même principe, on peut définir un multiplexeur de mots.
- Le **démultiplexeur** définit le circuit réalisant l'opération inverse.

43

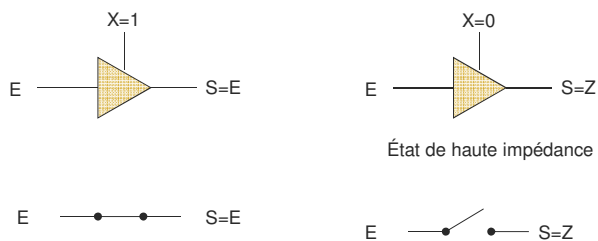
Multiplexeur (2)



44

Portes 3 états (Tristate)

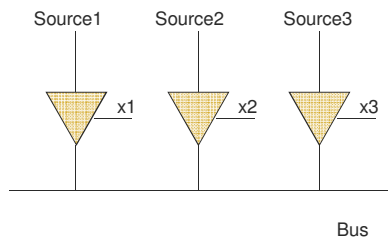
- Introduire des portes 3 états pour gérer les conflits d'écriture sur des bus.



45

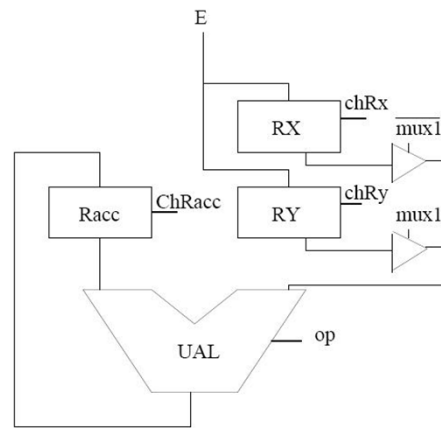
Écriture sur un bus

- Une seule source est autorisée à un instant donné à écrire sur le bus.



46

Utilisation des portes trois états



47