Architecture des ordinateurs
IFT1227
Introduction

Franz Girardin

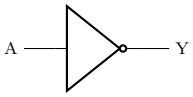
17 février 2024

Section 1

Circuits logiques

Couche logique numérique Constituées de *portes logiques* construite à partir de **transisteurs** qui prennent un **signal 0** ou **1** et calcule une **fonction logique** ET, OU et NON, etc.

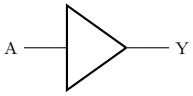
Porte NON



A	Y
1	0
0	1

$Y = \overline{A}$

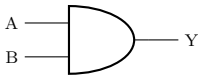
Porte BUF



A	Y
0	0
1	1

$Y = A$

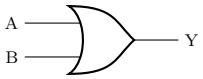
Porte ET



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

$Y = AB$

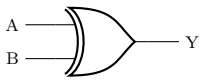
Porte OU



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = A + B$

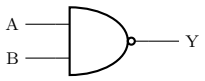
Porte XOR



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

$Y = A \oplus B$

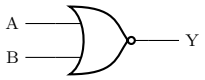
Porte NAND



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

$Y = \overline{AB}$

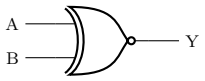
Porte NOR



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

$Y = \overline{A + B}$

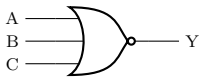
Porte XNOR



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

$Y = \overline{A \oplus B}$

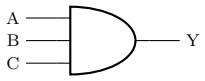
Porte NOR3



A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

$Y = \overline{A + B + C}$

Porte AND3



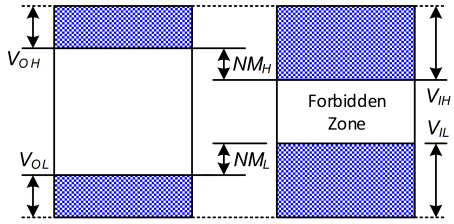
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$Y = ABC$

Définition de la marge de bruit Tolérance d'un circuit aux **perturbations** pouvant fausser l'interprétation du signal.

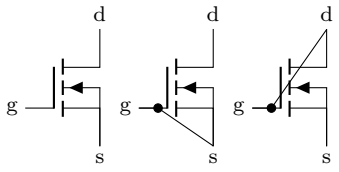
- ▷ V_{IH} : $\min(V | V_{in} ::= \mathbf{1})$
- ▷ V_{IL} : $\max(V | V_{in} ::= \mathbf{0})$
- ▶ Signal reçu **min** ou **max** est être interprété comme **1** ou **0**.
- ▷ V_{OH} : $\min(V | V_{out} ::= \mathbf{1})$
- ▷ V_{OL} : $\max(V | V_{out} ::= \mathbf{0})$
- ▶ Signal **min** ou **max** que l'émetteur s'engage à fournir pour être interprété comme **1** ou **0**.

$$NM_H = V_{OH}\ (\acute{e}m.) - V_{IH}\ (src.)$$
$$NM_L = V_{IL}(src.) - V_{OL}(\acute{e}m.)$$



Transistors

Éléments de base des circuits électroniques. Ils sont composés de trois broches ; le drain, la source, et la **grille** qui contrôle les deux autres comme un interrupteur. La figure suivante représente un transistor hors tension *off*, un transistor hors tension mais polarisé *off* et un transistor sous tension *on*.



Composition d'un circuit

Circuit::= E., S., spec. *fonct.*, spec. *temp.*

Propriétés d'un circ. combinatoire

- ▷ **Noeud** $\Longrightarrow$ *In* ou **connexion à** un *Out*.
- ▷ **Aucun** chemin cyclique.

Sommes de produits SOP En considérant les variables *In* d'une ligne de la table de vérité, il faut identifier la **conjonction** (produit) nécessaire pour engendrer un **1** logique. Un **minterm** est une représentation du produit engendrant un 1 logique.

A	B	Y	*minterm*
V	F	0	$\overline{A}\overline{A}$
0	1	1	$\overline{A}B$
1	0	0	$A\overline{B}$
1	1	1	AB

$$Y(A,B) = \overline{A}B + AB \leftrightarrow Y(A,B) = \sum(1,3)$$

Produits de sommes En considérant les variables *In* d'une ligne de la table de vérité, il faut identifier la **somme** nécessaire pour engendrer un **0** logique. Un **maxterm** est une représentation de la somme engendrant un 0 logique.

A	B	Y	*maxterm*
V	F	0	$A + B$
0	1	1	$A + \overline{B}$
1	0	0	$\overline{A} + B$
1	1	1	$\overline{A} + \overline{B}$

$$Y(A,B) = (A+B)(A+\overline{B}) = \prod(0,2)$$

Axiome	Dual	Nom
$B = 0$ if $B \neq 1$	$B = 1$ if $B \neq 0$	Bin. field
$\overline{0} = 1$	$\overline{1} = 0$	NOT
$0 \cdot 0 = 0$	$1 + 1 = 1$	AND/OR
$1 \cdot 1 = 1$	$0 + 0 = 0$	AND/OR
$0 \cdot 1 \cdot 0 = 0$	$1 + 0 = 1 + 1$	AND/OR

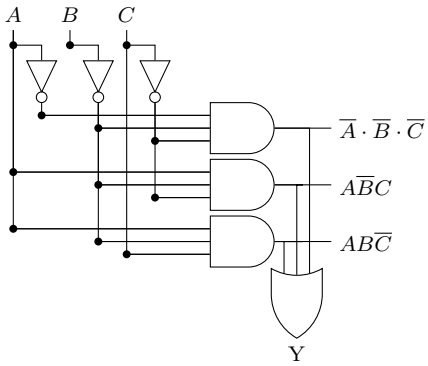
Théorème	Dual	Nom
$B \cdot 1 = B$	$B + 0 = B$	Identité
$B \cdot 0 = 0$	$B + 1 = 1$	Élément nul
$B \cdot B = B$	$B + B = B$	Indépotence
	$\overline{\overline{B}} = B$	Involution
$B \cdot \overline{B} = 0$	$B + \overline{B} = 1$	Complément

De Morgan

$$\neg(A + B) = \neg A + \neg B$$

$$\neg(A \cdot B) = \neg A \cdot \neg B$$

Exemple simple de circuit combinatoire



$$Y = \overline{A} \cdot \overline{B} \cdot \overline{C} + A\overline{B}\overline{C} + A\overline{B}C$$

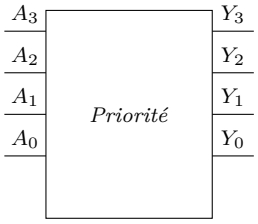
Simple règles de schématisation Les **E.** sont en haut à gauche et les **S.** sont en bas à droite ; on utilise des **fils droits**, préférablement.

(a) Fils engendrant une jonction T

(b) Connexion explicitée par un point

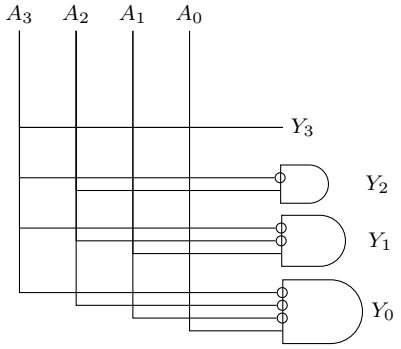
(c) Fils croisés sans point (∴ non connectés)

Circuit de priorité



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

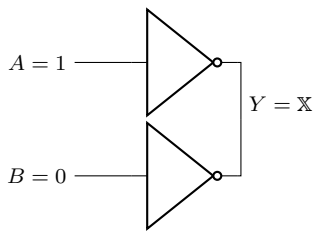
Représentation d'un circuit de priorité



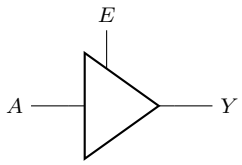
Entrée don't care ou X Ces entrés sont utilisées pour spécifier que la variables possédant le *don't care* n'affecte pas le résultat de la fonction logique. Une file de priorité peut être résumée par la table suivante.

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	*d*	0	0	1	0
0	1	*d*	*d*	0	1	0	0
1	*d*	*d*	*d*	1	0	0	0

Contention : signal X Se produit lorsque les portes logiques et les entrées sont telles que la sortie à générer est **contradictoire**.



Tampon à trois états : signal $\mathbb{Z}$ Circuit dans lequel une entrée $\mathbb{E}$ est connecté à une porte tampon et contrôle la **propagation du signal**. Lorsque l'entrée $\mathbb{E}$ est sous tension haute, la porte agit comme un tampon normal ; lorsque l'entrée $\mathbb{E}$ est sous tension basse, la porte produit le signal $\mathbb{Z}$ qui indique que A est *contrôlé*.



Méthodes de Karnaugh Méthode **graphique** permettant de simplifier les formules de circuits

- ▷ **Organiser** les éléments en grille de façon à ce que chaque cellule ne diffère d'une cellule voisine que par **1 bit**.
- ▷ Remplir la grille de façon à refléter le **tableau d'origine**.
- ▷ **Grouper** ou entourer les cellules adjacentes qui possèdent un **1**.

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

AB

C \ AB	00	01	11	10
0	1	0	0	0
1	1	0	0	0

AB

C \ AB	00	01	11	10
0	$\overline{A}\cdot\overline{B}\cdot\overline{C}$	0	0	0
1	$\overline{A}\cdot\overline{B}C$	0	0	0

Solution : considérer les variables qui ne changent pas leurs valeurs entre les cases **groupés**. Ici, la variable C change de valeur entre le cellule 1 et la cellule 2 ; elle n'est donc pas considérée dans **l'équation simplifiée**. Nous avons alors :

$$Y = \overline{AB}$$

Autre exemple de Karnaugh-map de 3 entrées Parfois, il y a plusieurs **implicants**.

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

AB

C \ AB	00	01	11	10
0	0	1	1	0
1	0	1	0	0

$$Y = \overline{A}B + B\overline{C}$$

Autre exemple de Karnaugh-map de 3 entrées Pour les tables de Karnaugh à 4 entrées et plus, les implicants devinnent plus complexes.

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

AB

CD \ AB	00	01	11	10
00	1			1
01			1	1
11	1		1	1
10		1		1

Karnaugh-map avec Don't Cares Les *don't care* peuvent complexifier la simplifiation de la fontion à cause du plus grand nombre de **cas possibles** lors du regroupement.

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

AB

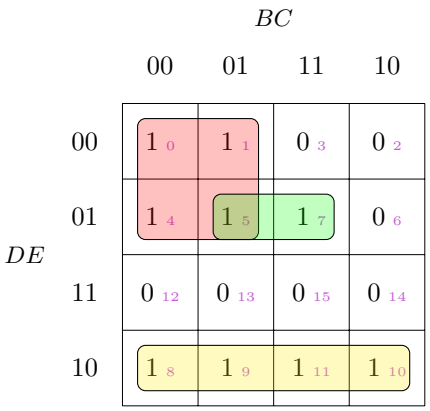
CD \ AB	00	01	11	10
00	1		X	1
01		X	X	1
11	1	X	X	X
10	1	1		X

$$Y = A + \overline{B}\cdot\overline{D} + C$$

Karnaugh-map de 5 entrées Il faut considérer **deux** K-map de **quatre** variables. Par convention, on peut omettre la première variable dans les deux K-map ; on considère que la dans la première K-map, la variable irgnorée a une valeur de **0** et, dans la 2e K-map, elle a une valeur de **1**. `Exemple sur Youtube`. Soit la fonction et K-maps correspondantes :

$$f(A,B,C,D,E) = \sum(0,1,2,4,5,6,10,13,1418,21,22,24,26,29,30)$$

K-map de $BCDE$ en considérant $A = 0$



$$D\overline{E} + C\overline{D} \cdot E + \overline{A} \cdot \overline{B} \cdot \overline{D}$$

K-map de $BCDE$ en considérant $A = 1$

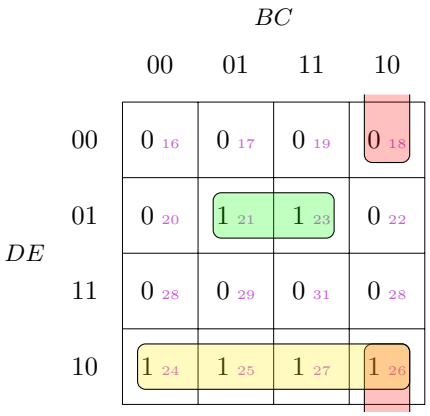


Figure 1.1

$$D\overline{E} + C\overline{D} \cdot E + AB \cdot \overline{C}$$

Karnaugh-map de 6 entrées

Note :

On peut utiliser la même approches que la méthode pour entrée, cette fois en considérant **4** K-map de 4 variables superposées dans un cube tridimensionnel Exemple sur `Youtube`

Multiplexeur

- ▷ 2^n lignes d'entrées
- ▷ 2^n N lignes de sélections
- ▷ 2^n Une seule sorties Y

Possède deux implémentations secondaires, soit (1) **portes logiques** et (2) **tampons** à trois états.

Quine-McCluskey Method

Méthode tabulaire qui permet de réduire les expression **SOP**

0	0	0	0	0	d
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	d
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	d

$$\sum(1, 3, 5, 6, 7, 10, 13) + \sum_{d}(0, 11, 15)$$

Formation du 1er tableau

- ▷ Considérer les mintermes tels que $F \neq 0$
- ▷ Grouper les mintermes :
  1. Minterme contiennent aucun 1
  2. Mintermes contiennent un 1

     ...
  4. Minterme contient quatre 1

0	0	0	0
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	1	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

Formation de la première tablea réduite

- ▷ Comparer les groupes adjacents.
- ▷ Trouver les termes qui diffèrent d'une valeur booléenne.
- ▷ Lorsqu'un différence de 1 seul terme est trouvée, placer le terme réduit dans la nouvelle terme avec un **underscore** `"-"` à la position de la différence.
- ▷ Comparer chaque terme d'un groupe à chaque terme de son groupe adjacent.

0	0	0	_	*
0	0	_	1	
0	_	0	1	
0	_	1	1	
_	0	1	1	
0	1	_	1	
_	1	0	1	
0	1	1	_	*
1	0	1	_	*
_	1	1	1	
1	_	1	1	
1	1	_	1	

Formation de 2e tableau

On continue la comparaison. Si pour un groupe donné, un terme ne peut pas être comparé avec *aucun groupe adjacent*, il s'agit d'un implicant premier. Par exemple, le premier terme du tableau précédent ne peut être comparé avec aucun terme de du groupe qui lui est adjacent.

0	_	_	1	*
_	_	1	1	*
_	1	_	1	*

Formation de la table de choix

IP	Minterms						
	0001	0011	0101	0110	0111	1010	1101
000-	✓						
011-				✓	✓		
101-						✓	
0–1	✓	✓	✓		✓		
–11		✓			✓		
-1-1			✓		✓		✓

Faire le choix

Considérer un implicant premier à la fois et cocher les mintermes qui sont couverts par l'implicant premier. Chercher ensuite les **colonnes** où il y a un seul cochet ; l'implicant premier à cet endroit est donc un **implicant obligatoire**

IP	Minterms						
	0001	0011	0101	0110	0111	1010	1101
000_	✓						
011_ *				✓	✓		
101_ *						✓	
0 __1	✓	✓	✓		✓		
__11		✓			✓		
_1_1 *			✓		✓		✓

Regarder ensuite quels mintermes sont couvert pas les **implicant obligatoire**. Dans l'exemple donnée les 3 implicants obligatoire couvrent 5 des 7 mintermes.

Parmi les implicants restants, il faut choisir la quantité minimale d'implicants premiers qui permettent de couvrir les mintermes restants.

IP	Minterms	
	0001	0011
000_	✓	
0 __1	✓	✓
101_		✓

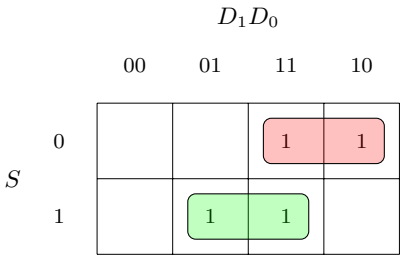
Écrire l'équation finale Considérer les implicants obligatoires trouvés et déduire la valeur des entrées formant la SOP en fonction du code de l'implicant obligatoire.

$$f(A, B, C, D) = \texttt{011\_ + 101\_ + \_1\_1 + \_1\_1}$$
$$= BC + A\overline{B}C + BD + \overline{A}D$$

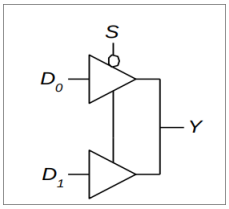
Multiplexeur MUX

- ▷ Possède 2^n lignes d'entrée : $D_0, D_1, \ldots, D_{2^n}$
- ▷ Une seule sortie Y
- ▷ **Rôle** : propager sur la sortie la valeur de l'une des 2^n entrées.

S	D_0	D_1	Y
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



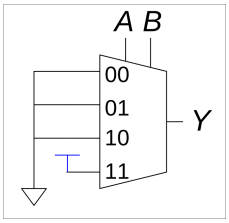
$$Y = D_0\overline{S} + D_1S$$



Logique en utilisant MUX

Soit une *fonction logique* $f(A, B...)$ dépendant de 2^n entrées, on peut utiliser un MUX pour repréter cette fonction.

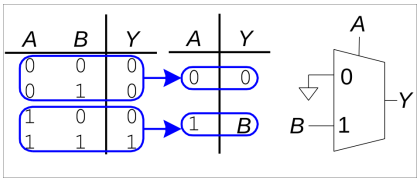
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



Note :

Sur les schéma de circuit, le **triangle** ▷ représent la valeur source 0 et la **barre** est 1

Réduction de la taile de MUX

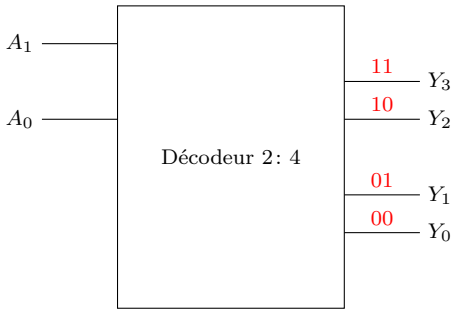


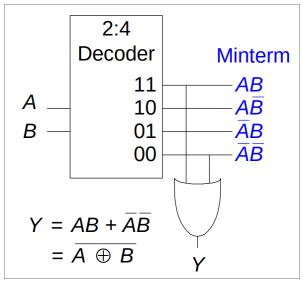
Décodeur

- ▷ Possède N entrées et 2^N sorties
- ▷ Les sorties expriment *combinaisons binaires* possibles des E.

A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Implémentation de décodeur





Caractéristiques électriques et tempo.

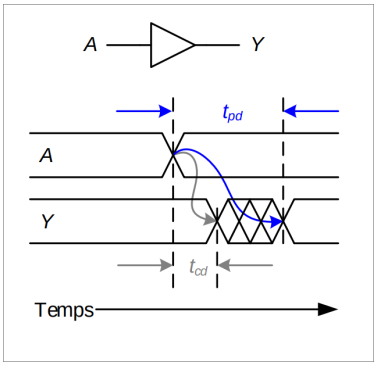
Les changements $0 \to 1$ ou $1 \to 0$ **ne sont pas** immédiat. Par exemple, s'il y a une porte tampons qui propage la valeur d'une entrée A, il y a un délais pour que Y passe de l'état précédent au nouvel état acuel de A.

Détails de propagation t_{pd} Délais maximal ; il est évalué du premier changement de l'entrée jusqu'à la *stabilisation* de la sortie.

Détails de contaminationt_{cd} Délais minimum ; il provient du premier changement de l'entrée jusqu'au premier changement de la sortie.

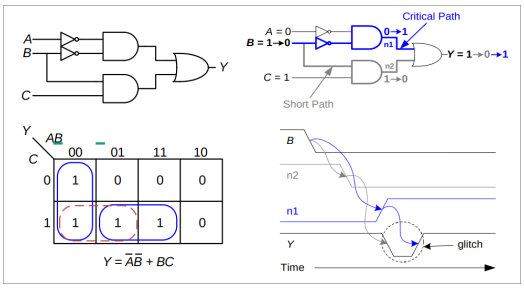
Chemin critique Chemin le plus long pour que la propagation **totale** s'effectue. Pour le calculer, il faut additionner les t_{pd} des portes impliquant le plus long chemin.

Chemin court Chemin le plus court engendrant une contamination. Pour le calculer, il faut additionner les t_{cd} des portes impliquant le plus court chemin.



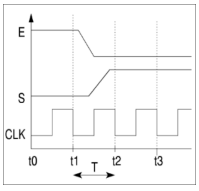
Aléas Se produit lorsque des transitions *non prévues* par la fonction booléenne apparaissent suite à une variation du vecteur d'E.

Exemple de glitch Puisque le chemin le plus court pour l'entrée B est la 2e porte $\mathbb{ET}$, lorsque B transitionne, on a B: $1 \to 0$, $1(C)^0(B) \to 0$. Lors que la porte ou est temporairement dans l'état 0, on a $0^0 \to 0$.



Horloge Les signaux ne se trouve pas toujours dans leur *état valide*, à cause des délais de montée, descente et propagation. Pour éviter les aléas, on impose au système de lire les valeurs à des **instants précis** et à des intervales réguliers.

L'intervale de temps entre deux impulsions régulières est le *temps de cycle* ou *période d'horloge*



Section 2

Circuits séquentiels

Principe En plus de l'état des entrées, les circuits logiques considèrent aussi les *information précédemment traitées* ; ils possèdent donc une mémoire.

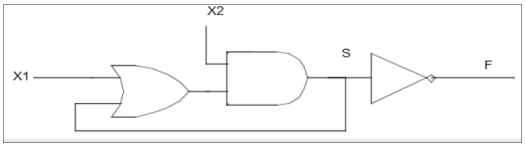
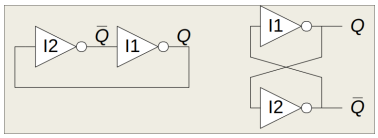


Figure 2.1 – Sortie réinjectée dans porte entrée

Circuit bistable Possède deux inverseurs et **aucune entrée**



Latch S/R Application d'un circuit bistable contenant additionnelement deux portes $\mathbb{OU}$ et deux entrées R et S. En pratique, le Latch *Set Reset* stocke un bit d'état Q et contrôle ce bit par les entrée S. Lors de l'étape *Set*, on a $S = 1, R = 0, Q = 1$. Lors de l'étape *Reset*, on a $S = 0, R = 1, Q = 0$.

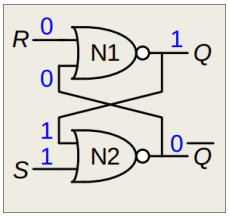


Figure 2.2 – $(R = 0) \wedge (S = 1) \implies (Q = 1) \wedge (\overline{Q} = 0)$

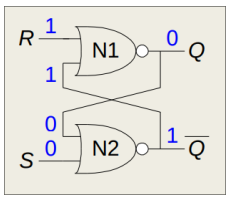


Figure 2.3 – $(R = 1) \wedge (S = 0) \implies (Q = 0) \wedge (\overline{Q} = 1)$

Lorsque S et R ont pour valeur **0**, il y a formation de mémoire. Pour un cycle on obtient $Q_{prev} = 0$, pour le cycle suivant on obtient $Q_{prev} = 1$

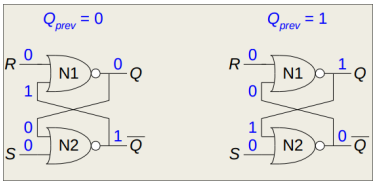


Figure 2.4 – $(R = 0) \wedge (S = 0) \implies (Q_{prev} = 0)$

Lorsque R et S on pour valeur **1**, il y a une incohérence ; $Q = \overline{Q}$.

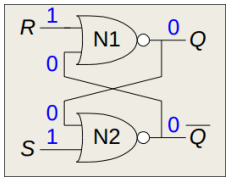


Figure 2.5 – $(R = 0) \wedge (S = 0) \implies (Q_{prev} = 0)$

Verrou D Circuit dans lequel la valeur de CLK contrôle la valeur propagée par l'entrée D. Lorsque le CLK a une valeur de **0** la valeur de Q varie, indépendamment de D. Lorsque la valeur de CLK est **1** la valeur de D est propagée. okkqwkboo

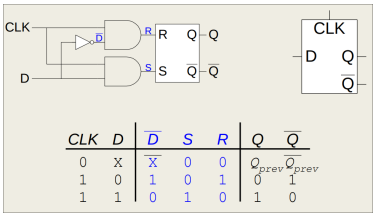
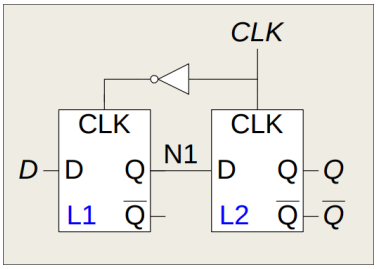


Figure 2.6

Bascule D Circuit dans lequel D est propagé sur le front montant de l'horloge CLK. Lorqque CLK passe de **0** à **1**, D est propagé vers Q. Autrement, Q préserve sa valeur précédente.



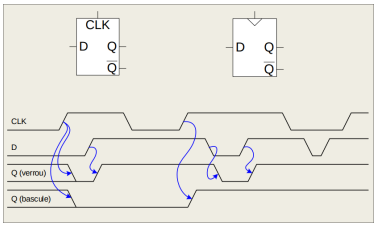
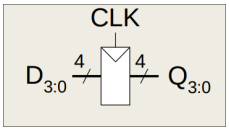
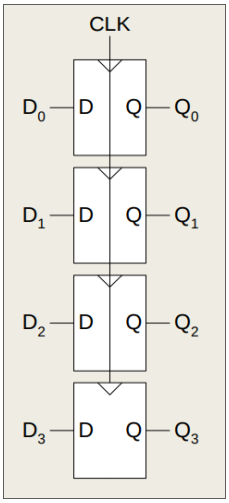


Figure 2.7 – Comparaison Verrou D et Bascule D

Registre Ils sont composés de bascules. Chaque bascule a un index et la taille du bus de bascule est spécifiée.

- ▷ Registre 3 : 0 possède 4 bascules.

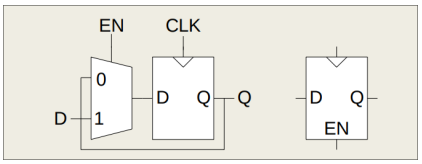




Bascule D **avec** `Enable` On ajout un signal `enable`. Est implémenté en ajoutant un MUX 2 :1

- ▷ `enable` $= 1$: D passe vers Q au fron montant CLK
- ▷ `enable` $= 0$: Bascule retient sa valeur précédente ; l'ancien Q deeure.

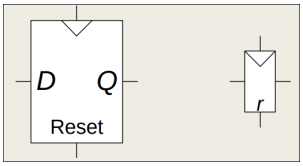
Sert à mémoriser l'information de Q durant plusieurs cyles d'horloge.



Bascule D avec `reset` On ajout un signal `reset`.

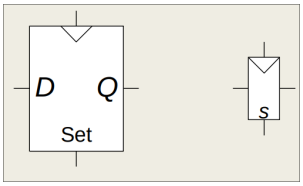
- ▷ `reset` $= 1$: Q est fixé à 0
- ▷ `reset` $= 1$: Fonctionne comme une bascule D normale

Peut être synchrone (fixe $Q = 0$ à condition que `reset = 1` et $CLK = 1$ front montant) ou asynchrone (fixe $Q = 0$ pour `reset` $= 1$, indépendamment de CLK.



Bascule D avec `set` On ajout un signal `set`.

- ▷ `set` $= 1$: Q est fixé à 1
- ▷ `set` $= 0$: Fonctionne comme une bascule D normale



Automate fini Il s'agit de la **base théorique** des circuits séquentiels Possède un nombre fini d'**états**. Pour u coupe d'instants (`t`, `t+1`), il possède une réponse S, un entre E et un état Q.