Institut
Mines-Télécom

Eléments de base des Architectures Numériques

De l'architecture aux porte logiques

Jean-Luc Danger

Plan

Variables et fonctions logiques

Variables logiques

- élément binaire qui appartient à l'ensemble $E = \{0, 1\}$
- Implanté par des grandeurs physiques :
  - tension, courant, photon sur fibre, champ EM,...
- **bit** d'une représentation d'un nombre en base 2
- Transmission des bits : parallèle ou série

Fonctions logiques

- Fonction d'une ou plusieurs variables logiques.

$$\begin{cases} E \times E \ldots \times E \to E \\ e_0, e_1, \ldots, e_n \to s = F(e_0, e_1, \ldots, e_n) \end{cases}$$

Types de Fonctions logiques

Fonctions combinatoires

- La sortie ne dépend que des entrées

$$\forall t, s(t) = F(e_0(t), e_1(t), \ldots, e_n(t))$$

Fonctions séquentielles

- La sortie dépend de l'état actuel des entrées et de leur passé

$$s(t) = F(e_0(t), e_1(t), \ldots, e_n(t), e_0(t - t_1), e_1(t - t_1) \ldots)$$

Implantation d'une fonction combinatoire

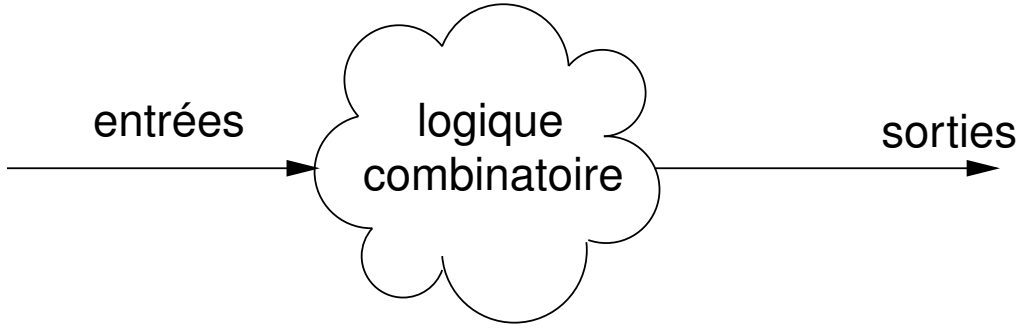


FIGURE : Fonction combinatoire

Implantation physique d'une fonction combinatoire

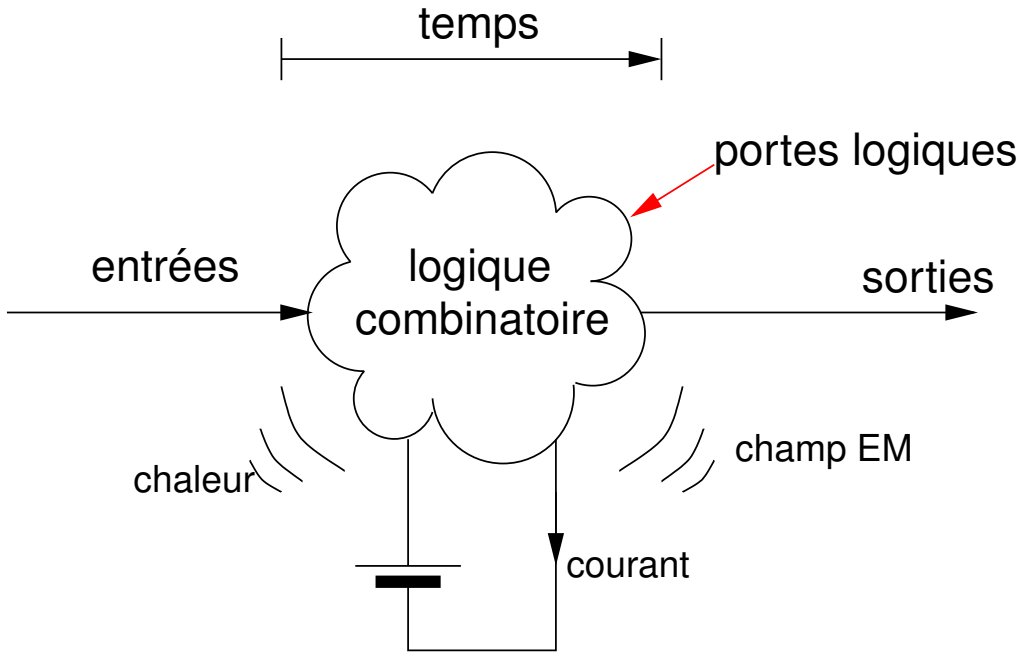


FIGURE : Fonction combinatoire réelle

Implantation d'une fonction séquentielle

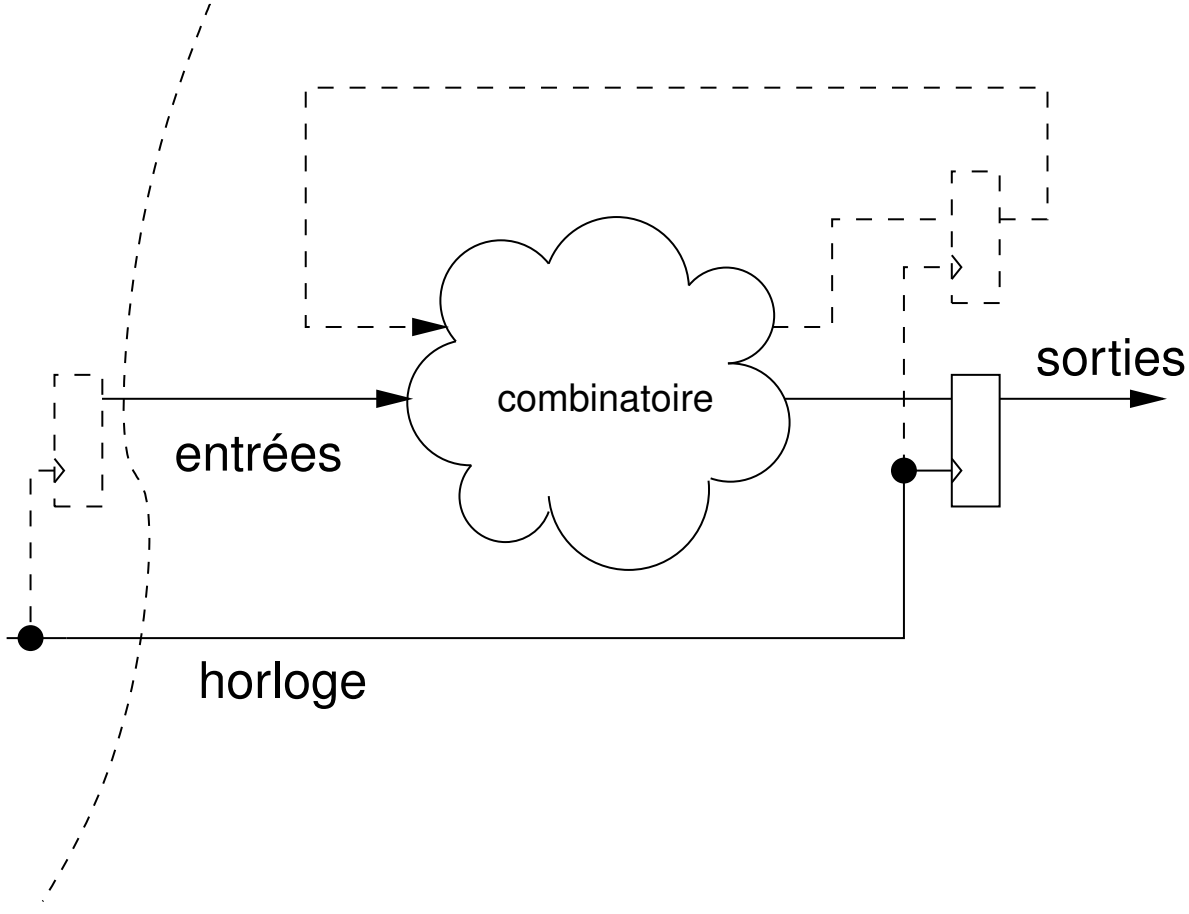


FIGURE : Fonction séquentielle

Plan

Fonctions logiques

Représentations

3 représentations possibles :

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

Analytique : En donnant l'équation analytique

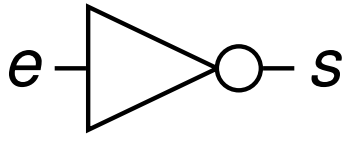
Graphique : En utilisant les symboles de fonctions de base

Fonctions élémentaires

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

$$s = \overline{e}$$

Table de vérité

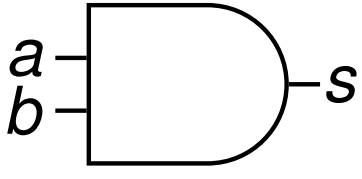
e	s
0	1
1	0

Fonctions élémentaires

Le “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

$$s = a \cdot b$$

Table de vérité

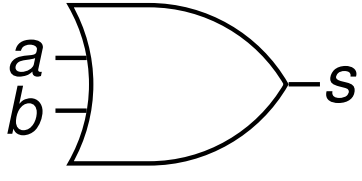
a	b	s
0	0	0
0	1	0
1	0	0
1	1	1

Fonctions élémentaires

Le “ou” (Or)

- 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

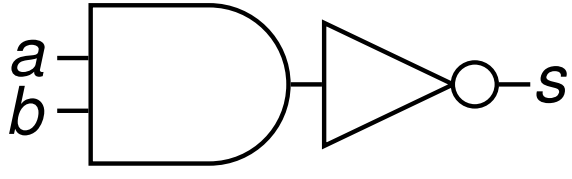
a	b	s
0	0	0
0	1	1
1	0	1
1	1	1

Fonctions de base

Le “non et” (Nand)

- La fonction complémentaire du And

Symbole



Équation

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

Table de vérité

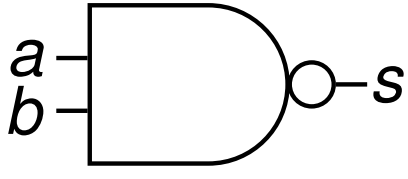
a	b	s
0	0	1
0	1	1
1	0	1
1	1	0

Fonctions de base

Le “non et” (Nand)

- La fonction complémentaire du And

Symbole



Équation

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

Table de vérité

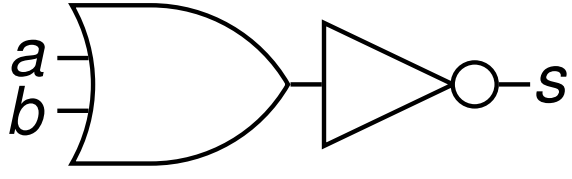
a	b	s
0	0	1
0	1	1
1	0	1
1	1	0

Fonctions de base

Le “non ou” (Nor)

- La fonction complémentaire du Or

Symbole



Équation

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

Table de vérité

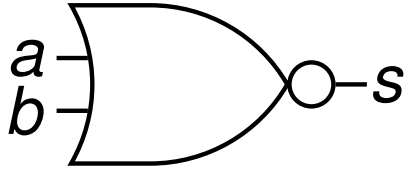
a	b	s
0	0	1
0	1	0
1	0	0
1	1	0

Fonctions de base

Le “non ou” (Nor)

- La fonction complémentaire du Or

Symbole



Équation

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

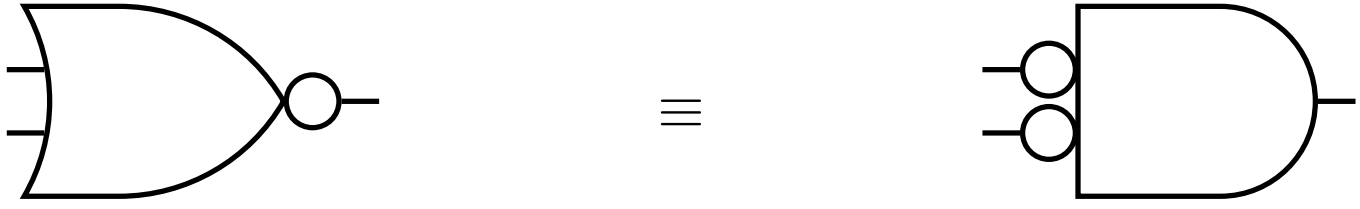
Table de vérité

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

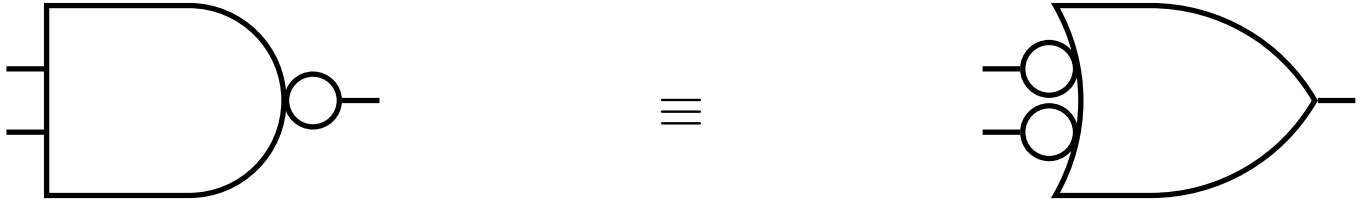
Équivalence And/Or

Théorème de De Morgan

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



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



Exercice

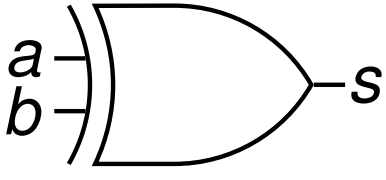
- Comment réaliser une porte à deux entrées, dont la sortie vaut ’1’ si et seulement si les deux entrées sont différentes ?
- Comment réaliser une porte à deux entrées, dont la sortie vaut ’1’ si et seulement si les deux entrées sont identiques ?

Fonctions de base

Le “Ou exclusif” (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

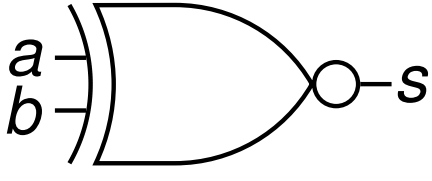
a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

Fonctions de base

Le “Non Ou exclusif” (Xnor)

- La sortie vaut 1 si les deux entrées sont identiques
- C’est la fonction complémentaire du xor

Symbole



Équation

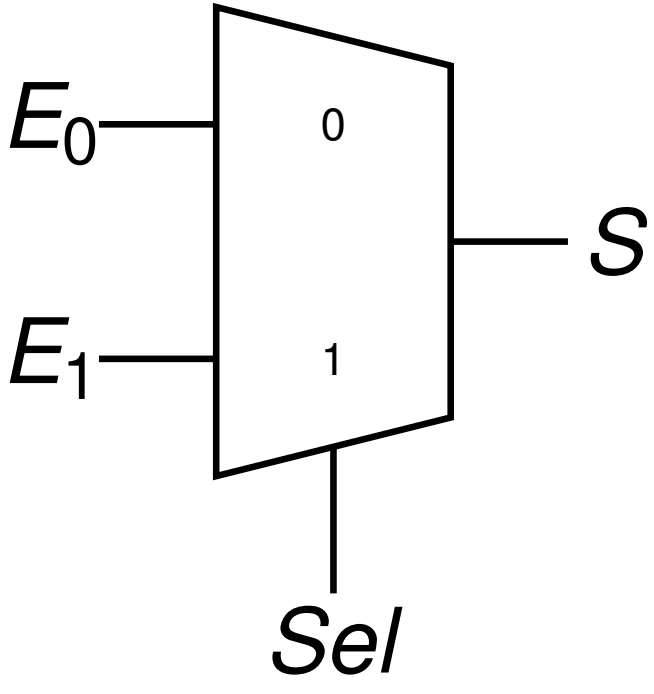
$$s = \overline{a \oplus b}$$
$$s = a \cdot b + \overline{a} \cdot \overline{b}$$

Table de vérité

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

Le multiplexeur

La fonction d'aiguillage

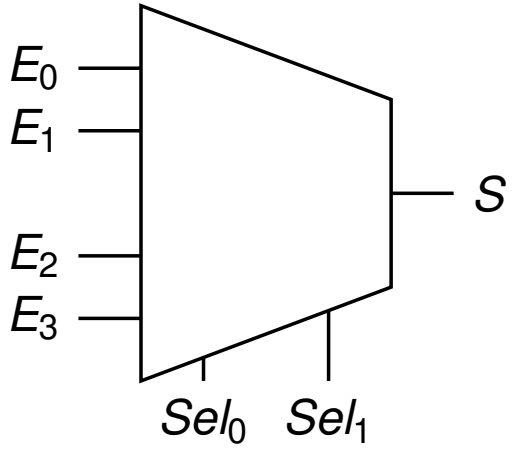


Sel	E_1	E_0	S
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

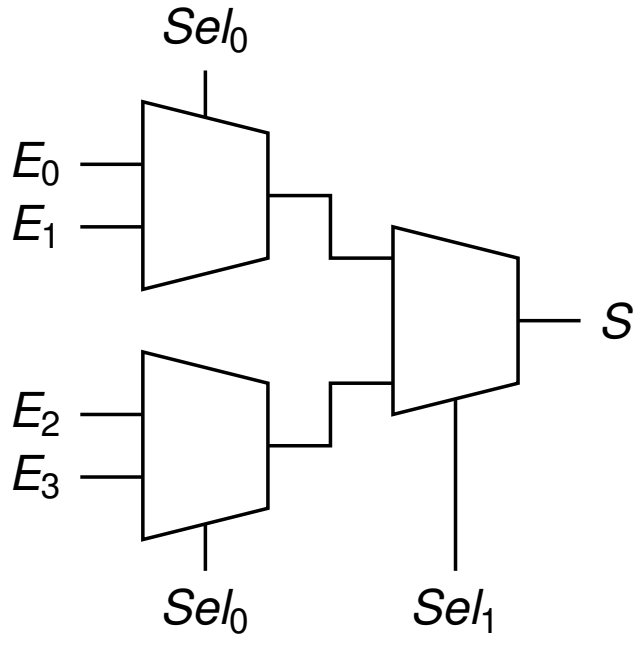
$$S = \overline{Sel} \cdot E_0 + Sel \cdot E_1$$

Le multiplexeur

Fonction d'aiguillage à quatre entrées



4 entrées $\Rightarrow$ 2 entrées de sélection



Peut être réalisé à partir de 3 mux 2 vers 1

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

Le multiplexeur

Généralisation

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

Plan

Représentation des nombres

Les nombres entiers

Un entier positif N dans une base b se représente par un vecteur $(a_{n-1}, a_{n-2}, \ldots, a_1, a_0)$ tel que :

$$N = a_{n-1} \cdot b^{n-1} + a_{n-2} \cdot b^{n-2} + \ldots + a_1 \cdot b^1 + a_0 \cdot b^0$$

Où :

- a_{n-1} est le chiffre le plus significatif
- a_0 est le chiffre le moins significatif
- a_i appartient à un ensemble de b symboles valant de 0 à $b-1$

Bases souvent utilisées

- $b = 10$
  - Représentation Décimale
  - $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $b = 16$
  - Représentation Hexadécimale
  - $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
- $b = 8$
  - Représentation Octale
  - $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$
- $b = 2$
  - Représentation Binaire
  - $a_i \in \{0, 1\}$ est un bit (binary digit)

Représentation binaire

Les nombres entiers

Un entier positif N en base 2 se représente par un vecteur $(a_{n-1}, a_{n-2}, \ldots, a_1, a_0)$ tel que :

$$N = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \ldots + a_1 \cdot 2^1 + a_0 \cdot 2^0$$

Où :

- a_i est un bit
- a_i appartient à un ensemble de 2 symboles valant 0 ou 1

Représentation binaire

Physiquement, le nombre de bits ne peut être infini !

- En utilisant n bits on ne peut représenter que les nombres entre 0 et $2^n - 1$.
- Il y a 2^n valeurs représentables
- L'arithmétique sur ces représentations est faite modulo 2^n

Représentation binaire

Exemple sur 4 bits

- Avec 4 bits on peut représenter les nombres allant de 0 à $15 = 2^4 - 1$
- L'arithmétique est alors modulo $2^4 = 16$
- Par exemple :
  - $15 + 1 = 0$
  - $0 - 1 = 15$

Décimal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Représentation binaire

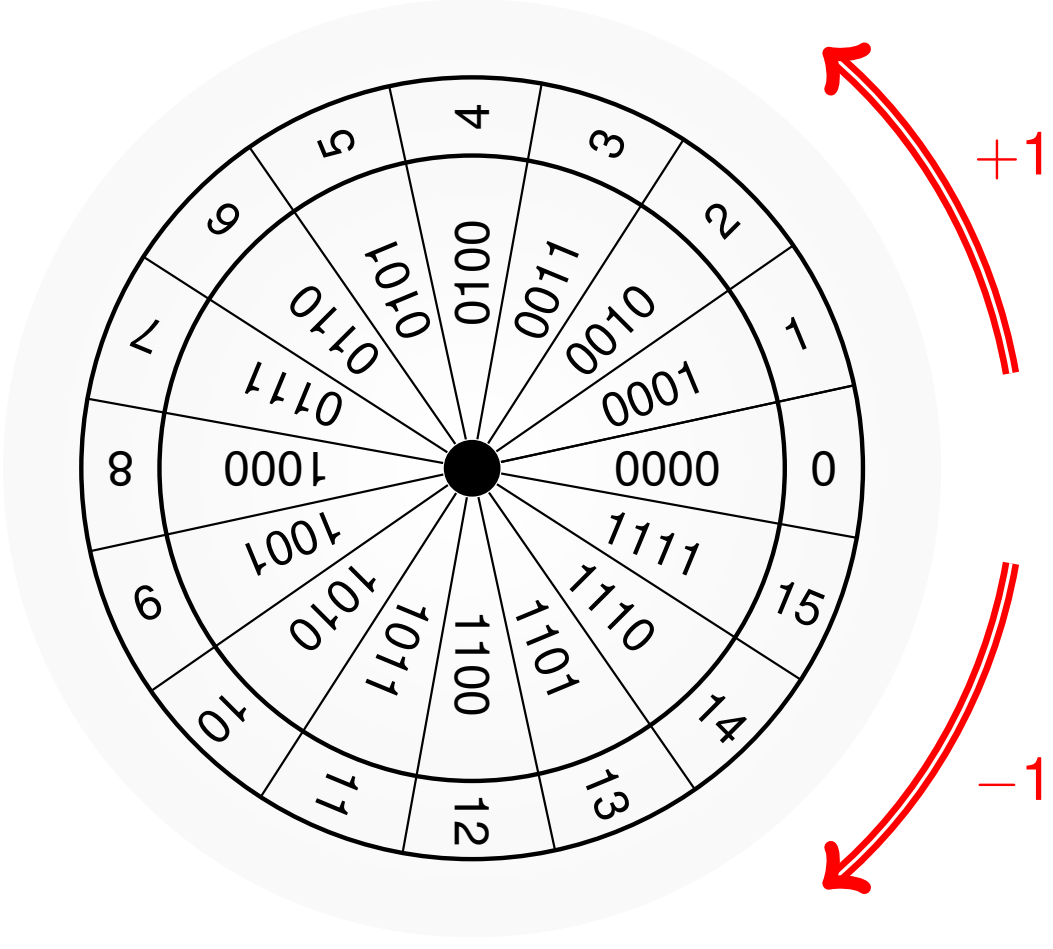
Exemple sur 4 bits

- Avec 4 bits on peut représenter les nombres allant de 0 à $15 = 2^4 - 1$
- L'arithmétique est alors modulo $2^4 = 16$
- Par exemple :
  - $15 + 1 = 0$
  - $0 - 1 = 15$
- Comment conserver le même comportement avec des nombres signés ?

Décimal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

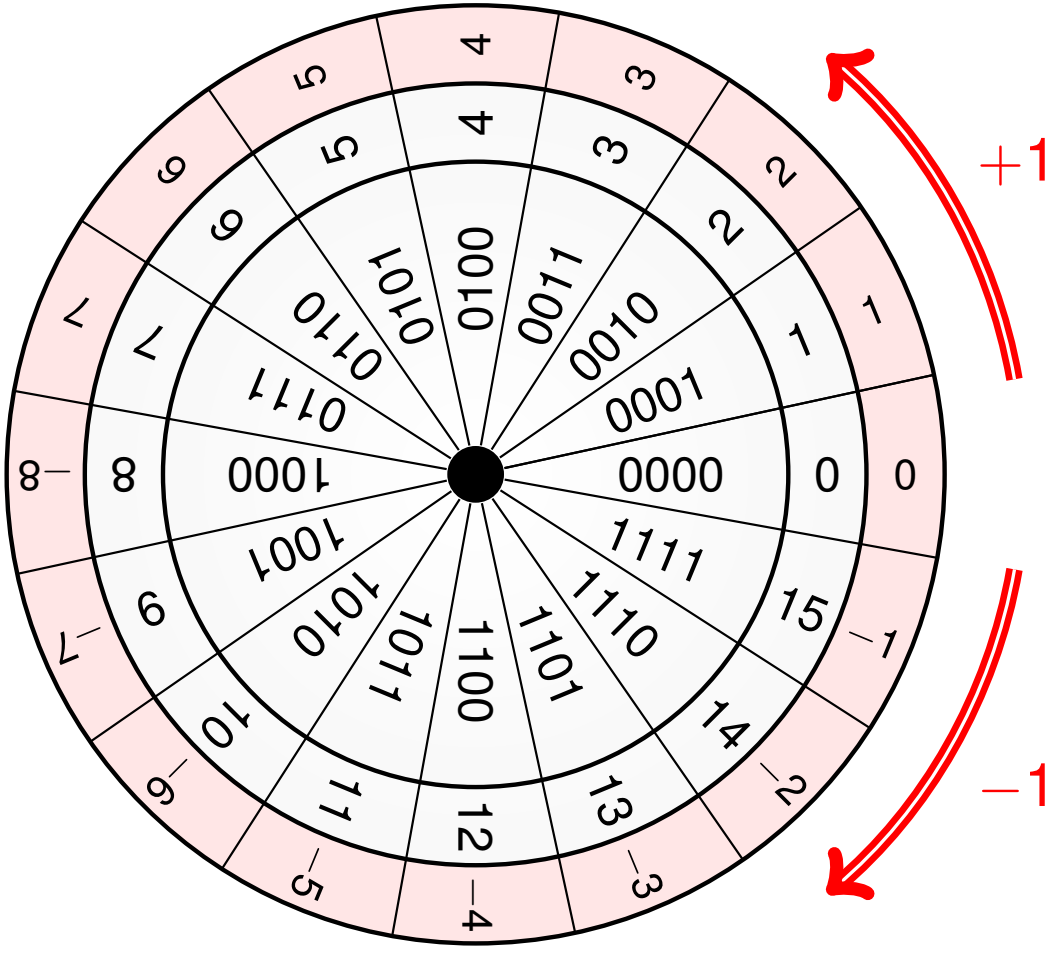
Représentation binaire des nombres

Les nombres entiers signés : exemple sur 4 bits



Représentation binaire des nombres

Les nombres entiers signés : exemple sur 4 bits



Représentation binaire des nombres

Représentation en complément à 2

Un nombre signé en CA2 sur n bits $A = a_{n-1}a_{n-2}\ldots a_0$

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

- C'est une interprétation du mot binaire
- Le bit de poids fort représente le signe
  - $0 \rightarrow$ nombre positif
  - $1 \rightarrow$ nombre négatif
- Permet de représenter sur n bits les nombres signés de -2^{n-1} à $2^{n-1} - 1$

Exemples

- Représentez -8 et +8
  - en utilisant 4 bits
  - en utilisant 5 bits

Exemples

- Représentez -8 et +8
  - en utilisant 4 bits
  - en utilisant 5 bits
- Représentez -1
  - en utilisant 1 bit
  - en utilisant 2 bits
  - en utilisant 3 bits . . .

Représentation binaire des nombres

Extension de signe pour le CA2

Soit $N = (a_{n-1}, a_{n-2}, \ldots, a_0)_{CA2/n}$ un entier relatif représenté en CA2 sur n bits.
Comment représenter N sur un $n+1$ bits.

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

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

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

D'où $N = (a_{n-1}, a_{n-1}, a_{n-2}, \ldots, a_0)_{CA2/n+1}$
On a dupliqué le bit de poids fort, on parle d'extension de signe.

Représentation binaire des nombres

Les nombres en virgule fixe

Un nombre décimal D en base 2 peut être approximé un par vecteur $(a_{n-1}, a_{n-2}, \ldots, a_1, a_0, a_{-m} \ldots a_{-m})$ tel que :

$$D = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \ldots + a_1 \cdot 2^1 + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + \ldots a_{-m} \cdot 2^{-m}$$

Où :

- $(a_{n-1}, \ldots a_0)$ est la partie entière
- $(a_{-1}, \ldots a_{-m})$ est la partie fractionnaire
- 2^{-m} représente la précision de cette approximation

Exemples

- Représentez 0.5, 3.625

Exemples

- Représentez 0.5, 3.625
- Représentez 0.6
  - en utilisant 1 bit
  - en utilisant 3 bits
  - en utilisant 5 bits

Addition

Exemple

Faire une addition en binaire sur 4 bits...

Addition

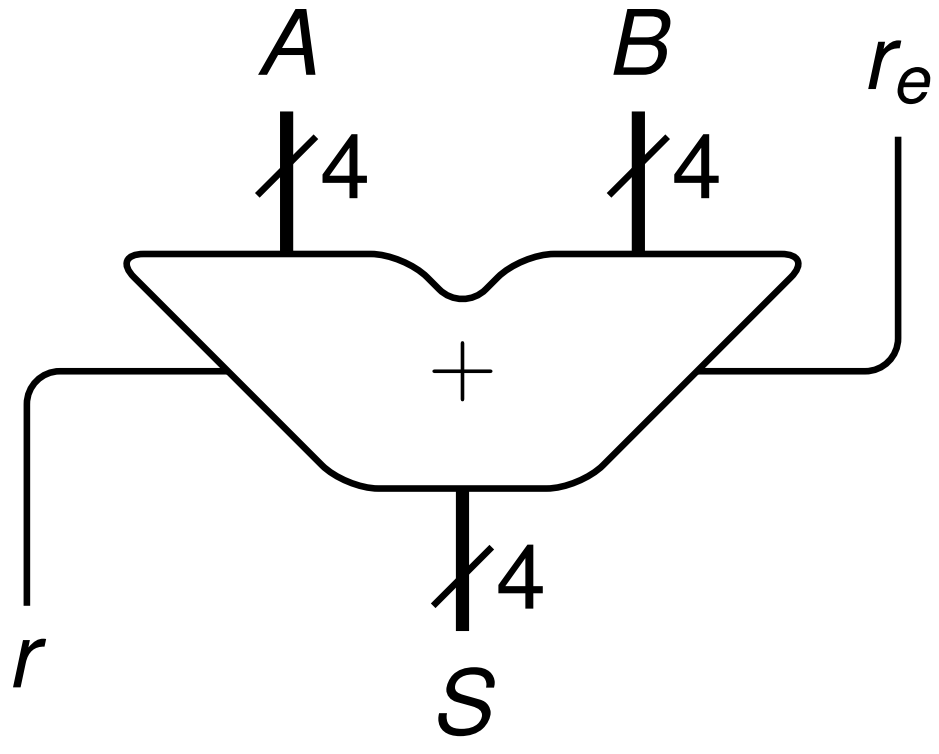
Exemple

Faire une addition en binaire sur 4 bits...

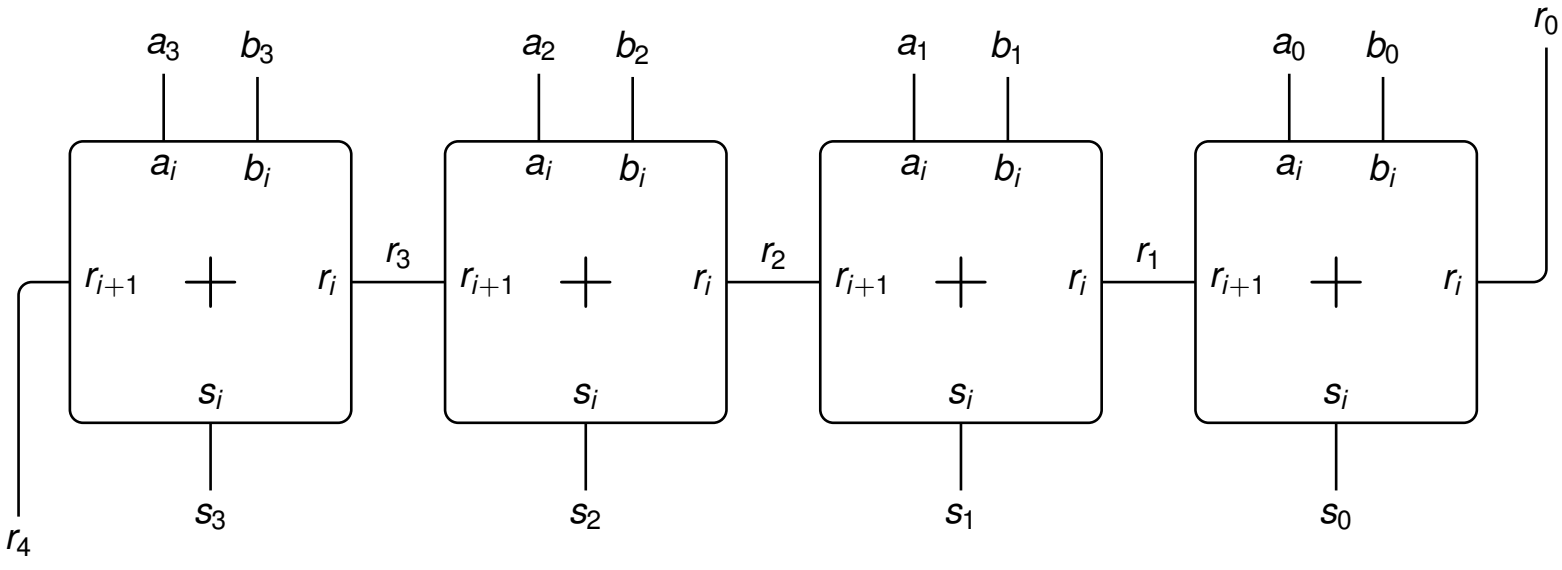
Décomposition de l'addition

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

Additionneur à propagation de retenue



Additionneur à propagation de retenue



Additionneur complet sur 1 bit

Arithmétiquement

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

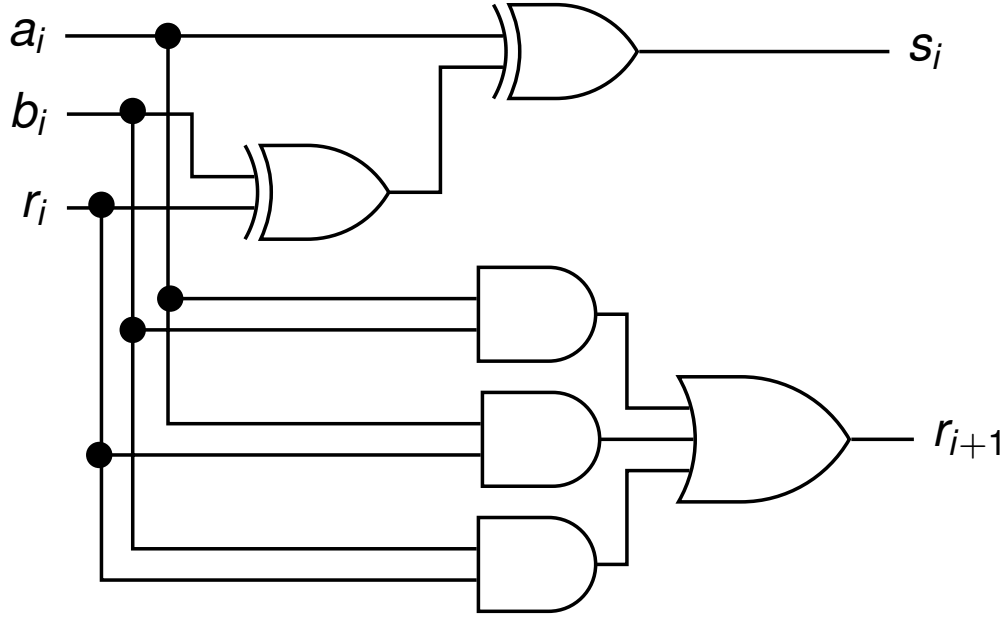
Table de vérité

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

Additionneur complet sur 1 bit

$$s_i = a_i \oplus b_i \oplus r_i$$

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



Soustracteur complet sur 1 bit

Table de vérité

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

Soustracteur complet sur 1 bit

Arithmétiquement

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

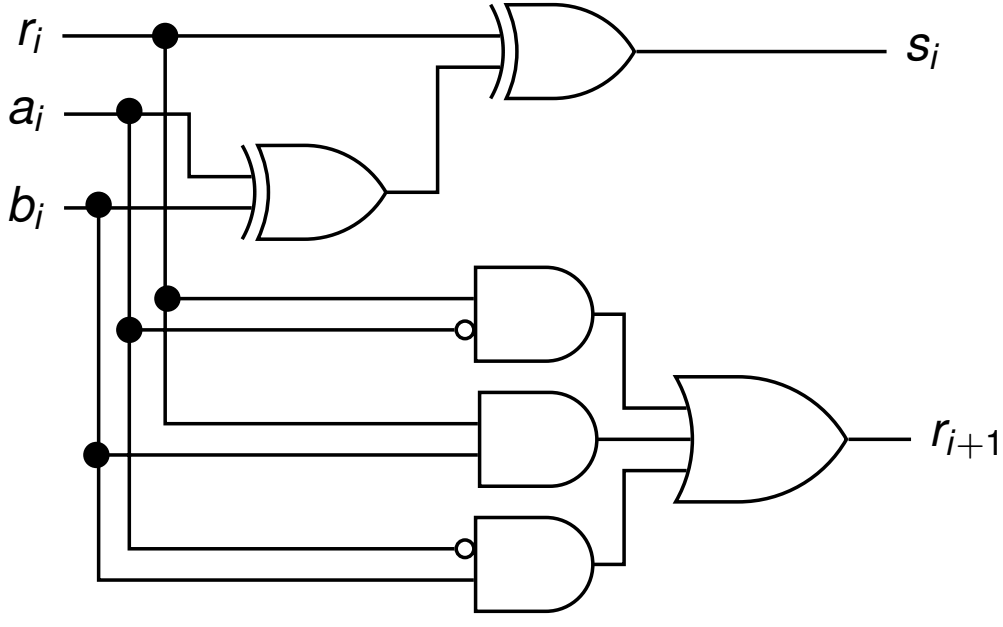
Table de vérité

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

Soustracteur complet sur 1 bit

$$s_i = a_i \oplus b_i \oplus r_i$$

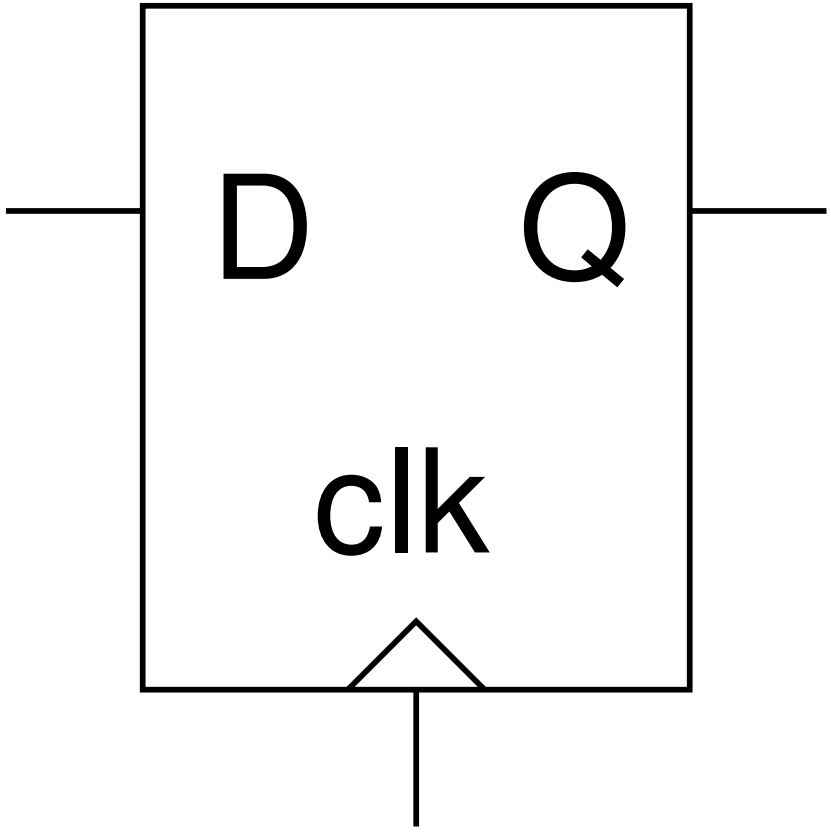
$$r_{i+1} = \overline{a_i} \cdot b_i + \overline{a_i} \cdot r_i + b_i \cdot r_i$$



Plan

La bascule D

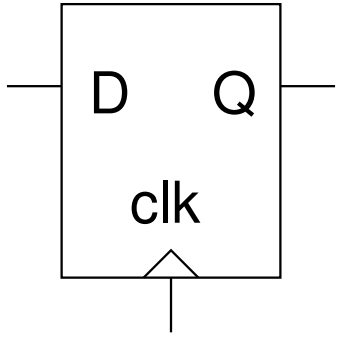
Élément de base de la logique séquentielle



La bascule D

Élément de base de la logique séquentielle

- bascule D, flipflop, dff, registre ...
- Une entrée particulière ! l'horloge **clk**
- L'horloge est symbolisée par un triangle



Fonctionnement :

- A chaque front montant de l'horloge **clk** (passage de $0 \rightarrow 1$) l'entrée **D** est copiée (échantillonnée, mémorisée) sur la sortie **Q**.
- Entre deux fronts d'horloge, la sortie **Q** ne change pas.

La bascule D

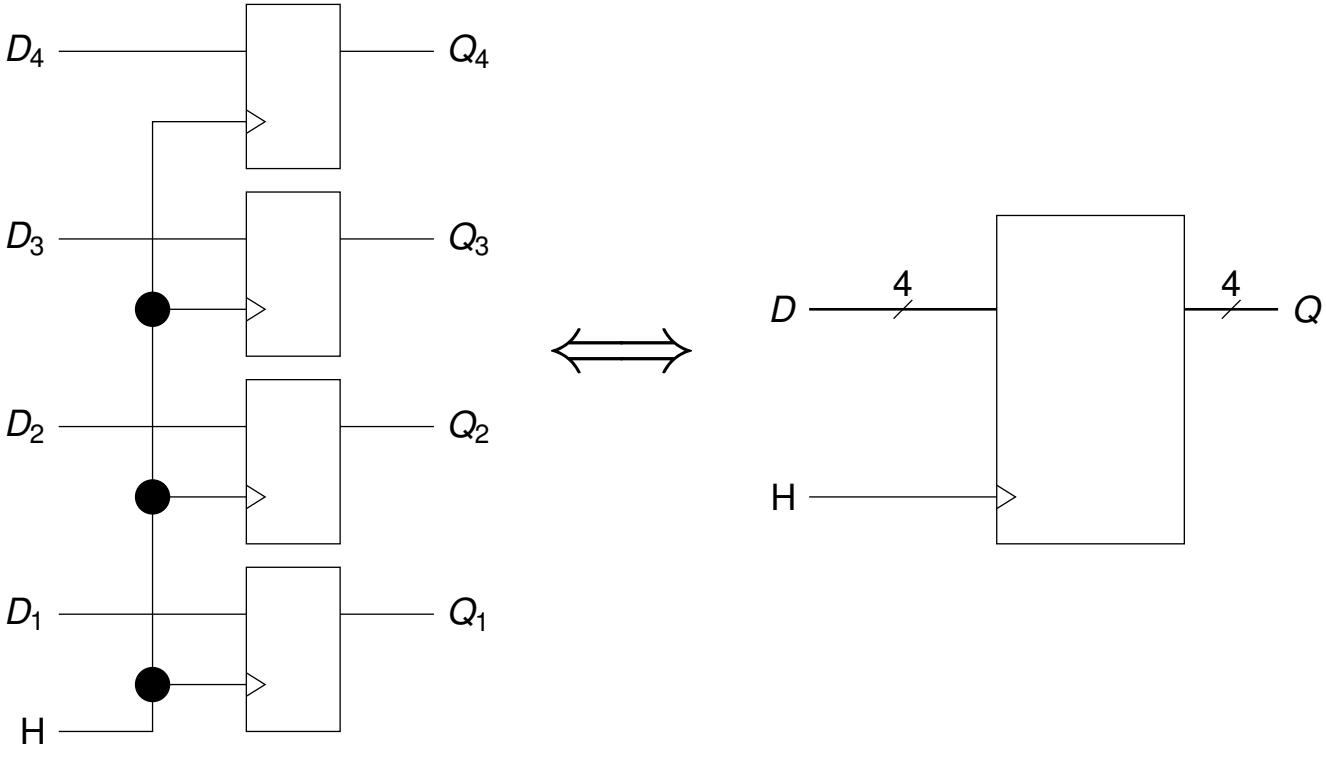
Table de vérité de la bascule D

D	clk	Q	
0	$\uparrow$	0	copie de D sur Q
1	$\uparrow$	1	copie de D sur Q
$\times$	0	Q	Q conserve sa valeur
$\times$	1	Q	Q conserve sa valeur
$\times$	$\downarrow$	Q	Q conserve sa valeur

La bascule D

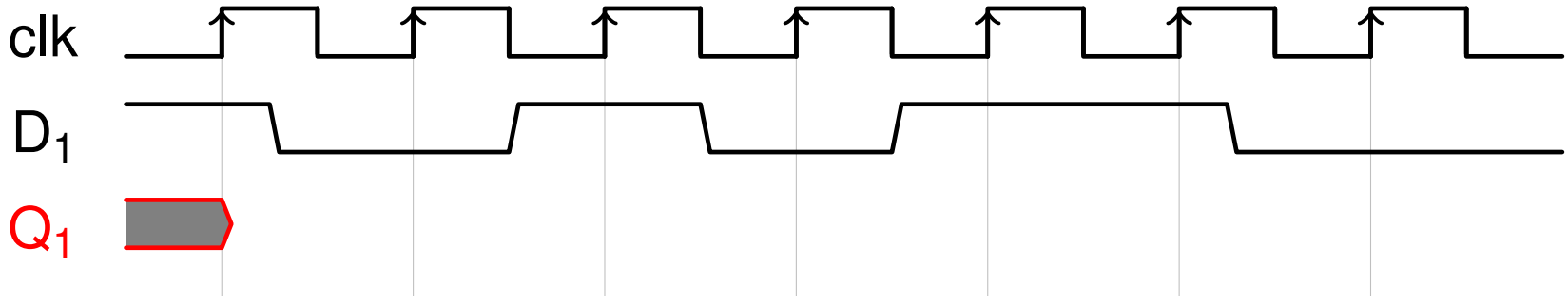
Le registre

- Un registre est un ensemble de bascules fonctionnant en parallèle.
  - Exemple un registre 4bits



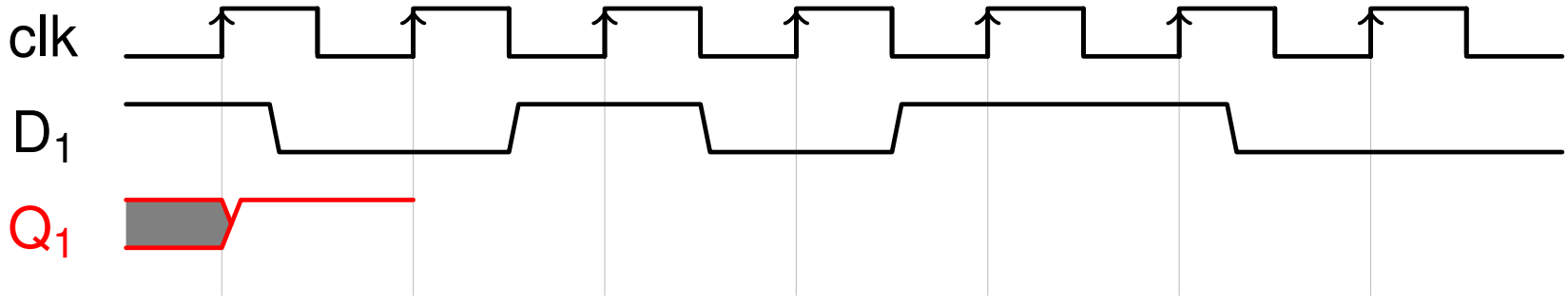
La bascule D

Exemple



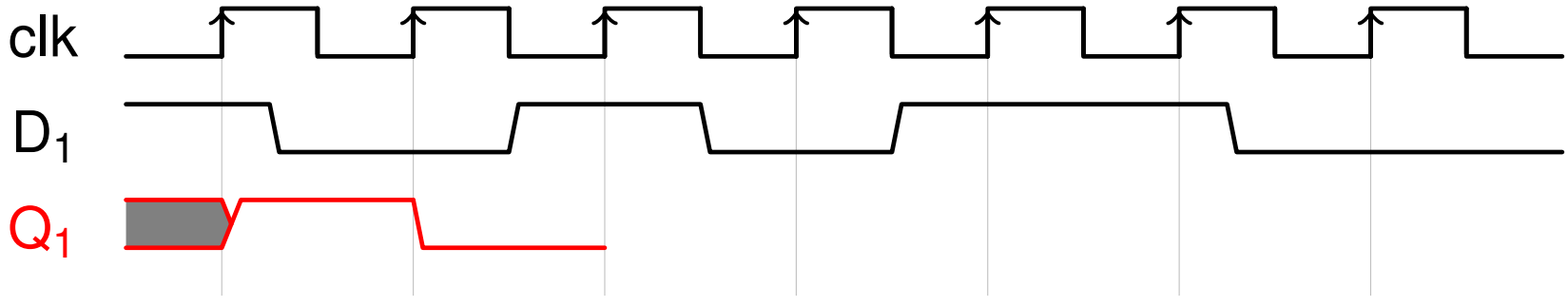
La bascule D

Exemple



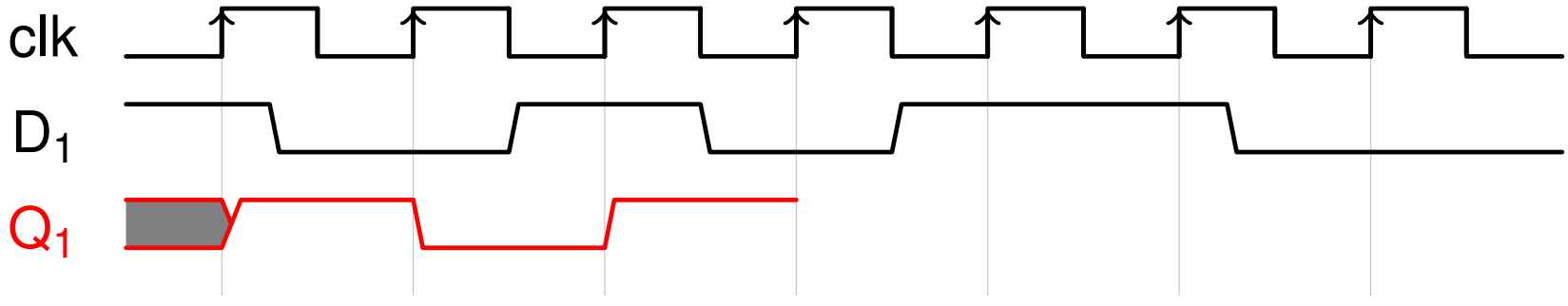
La bascule D

Exemple



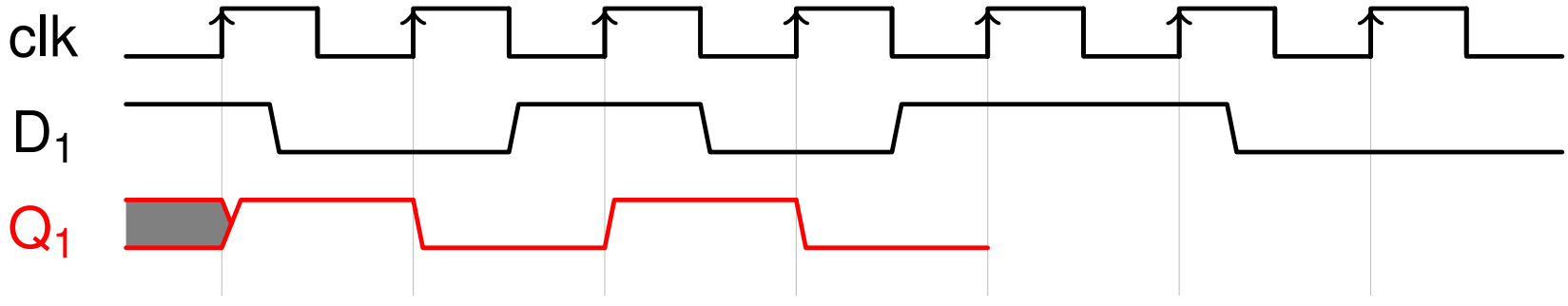
La bascule D

Exemple



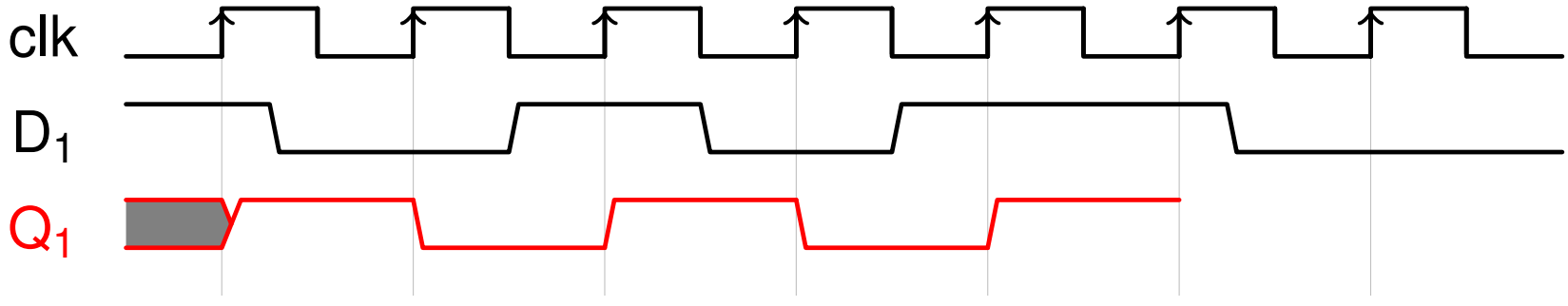
La bascule D

Exemple



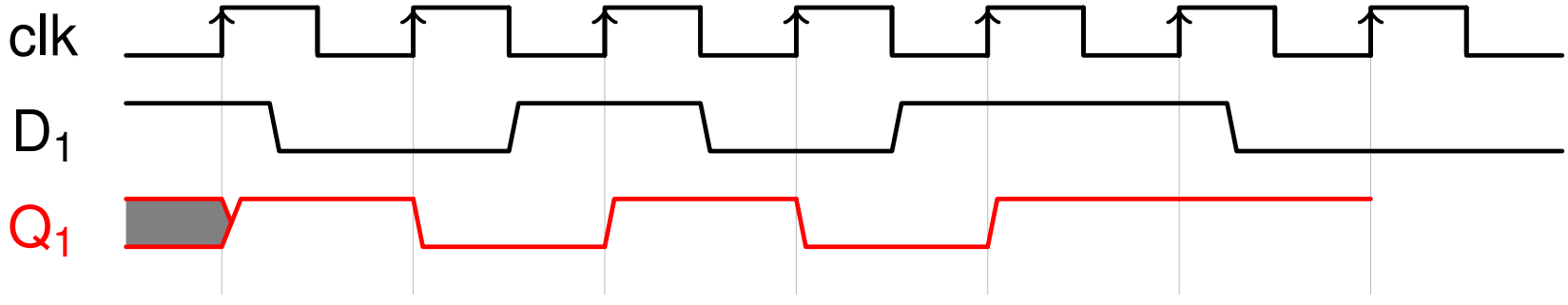
La bascule D

Exemple



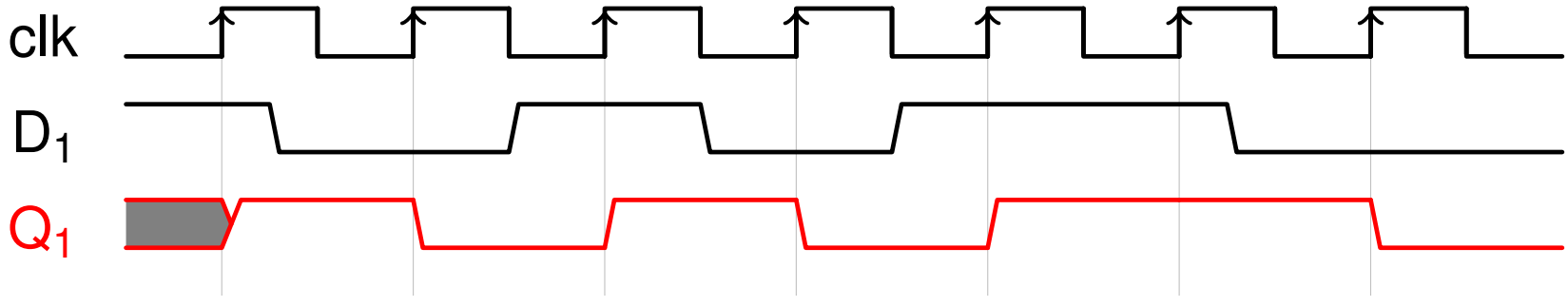
La bascule D

Exemple



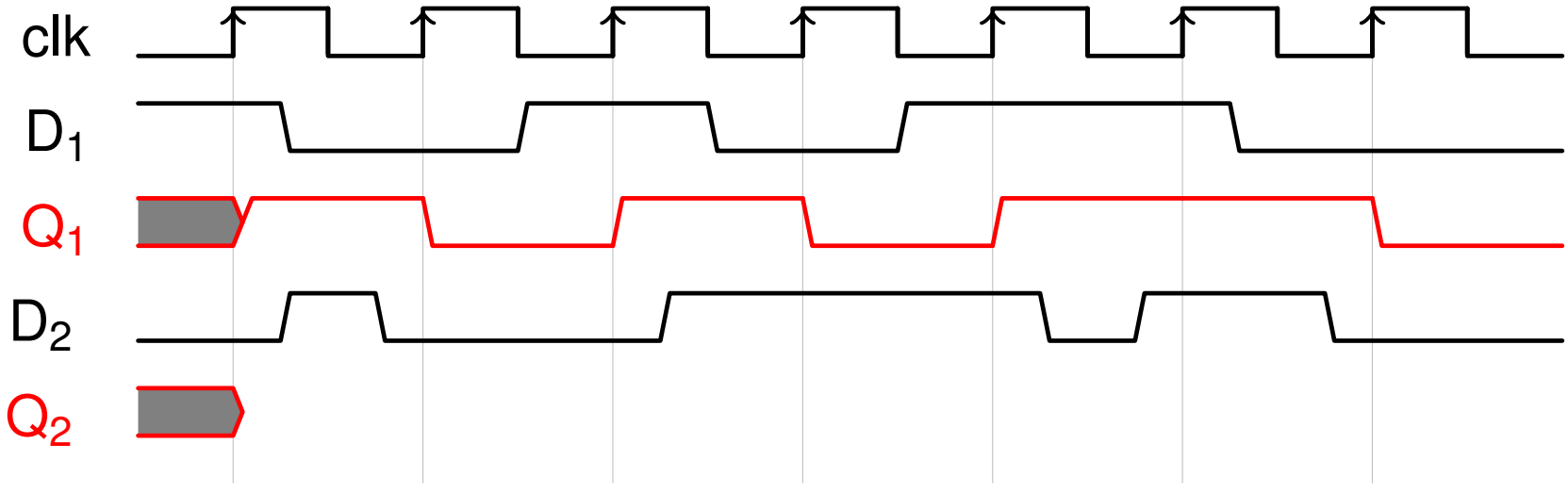
La bascule D

Exemple



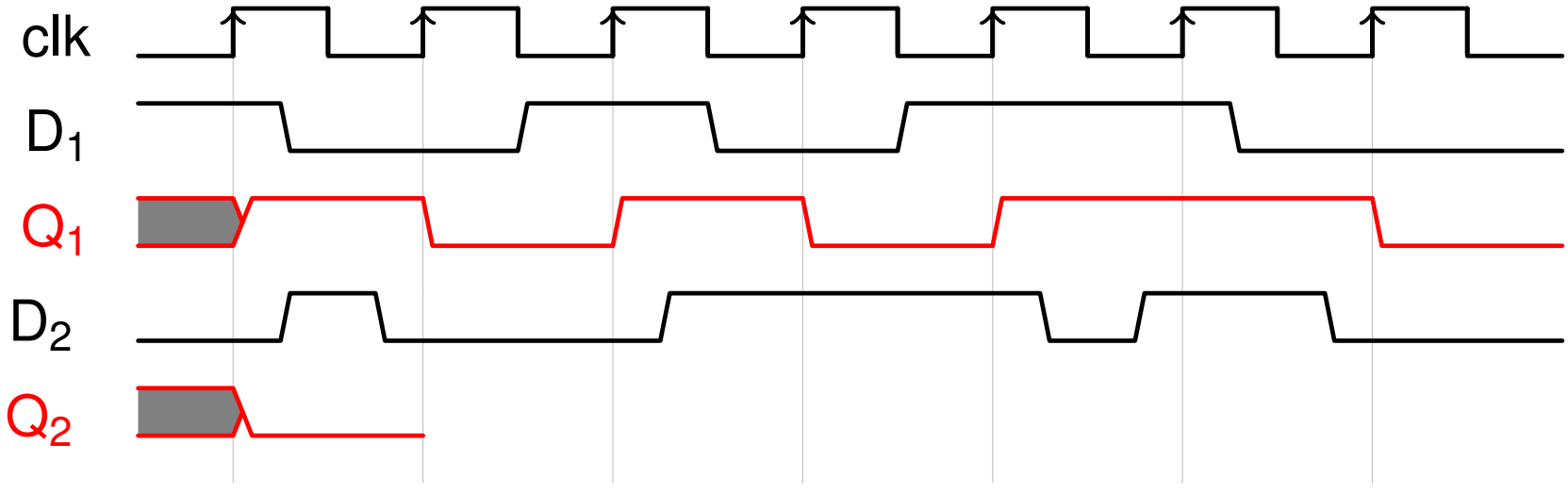
La bascule D

Exemple



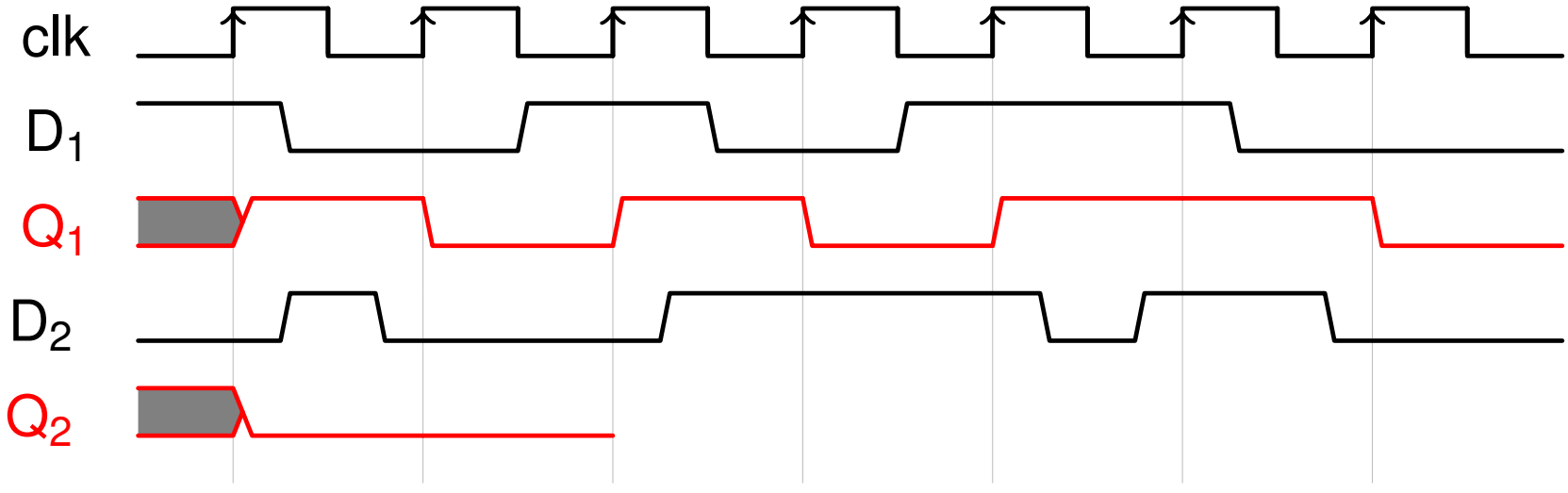
La bascule D

Exemple



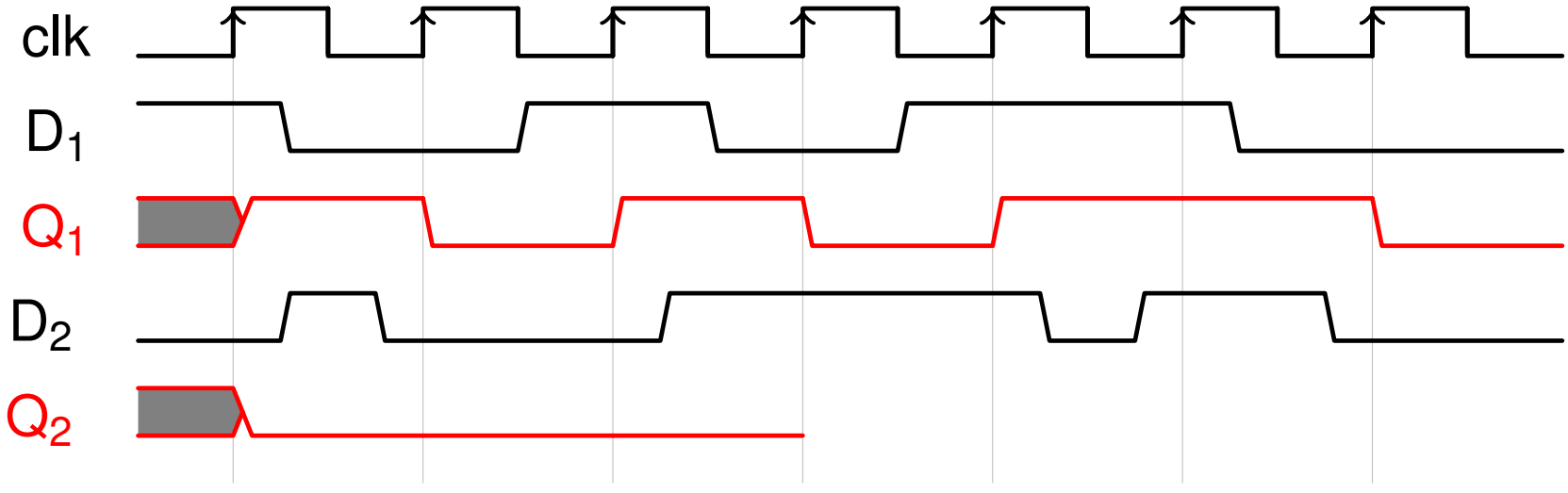
La bascule D

Exemple



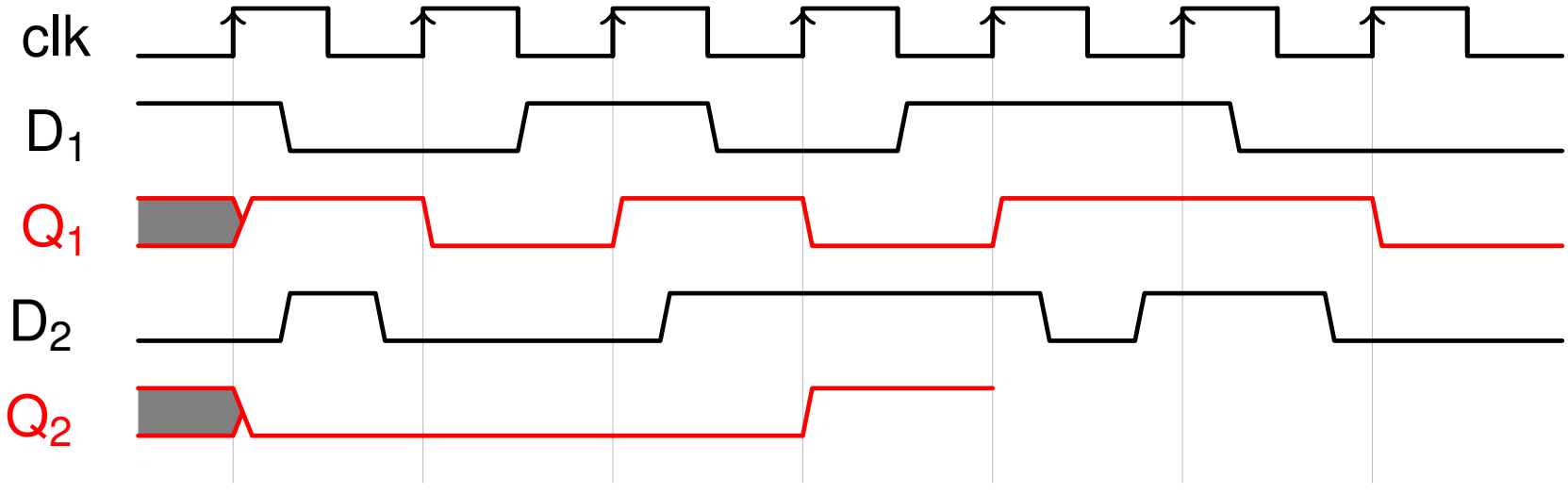
La bascule D

Exemple



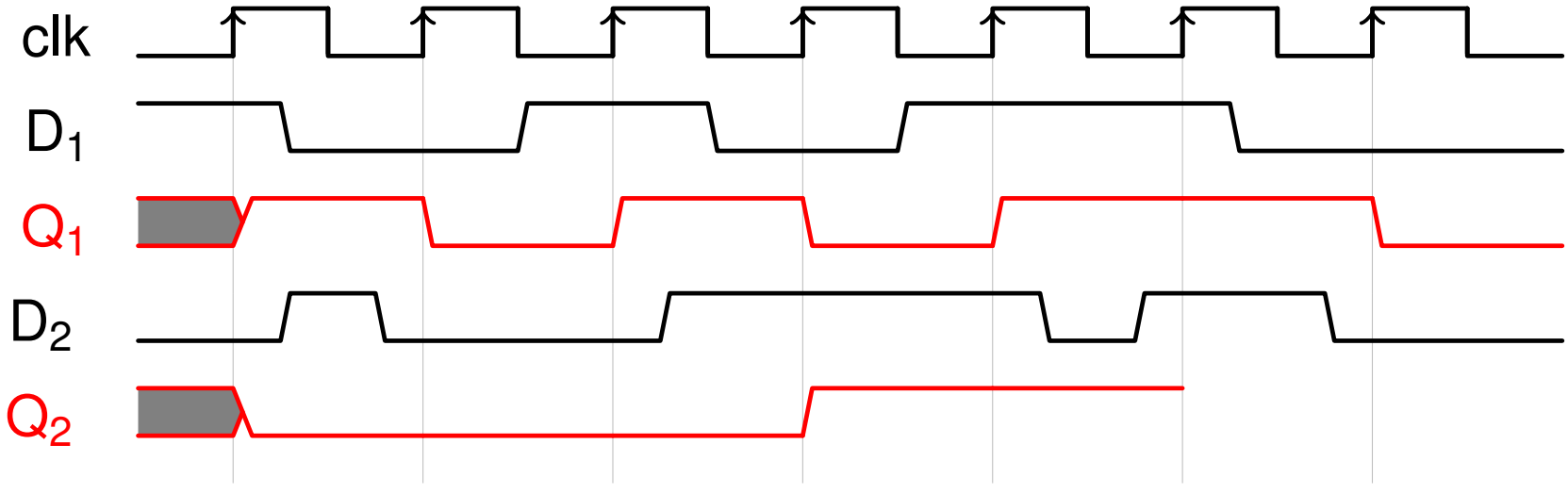
La bascule D

Exemple



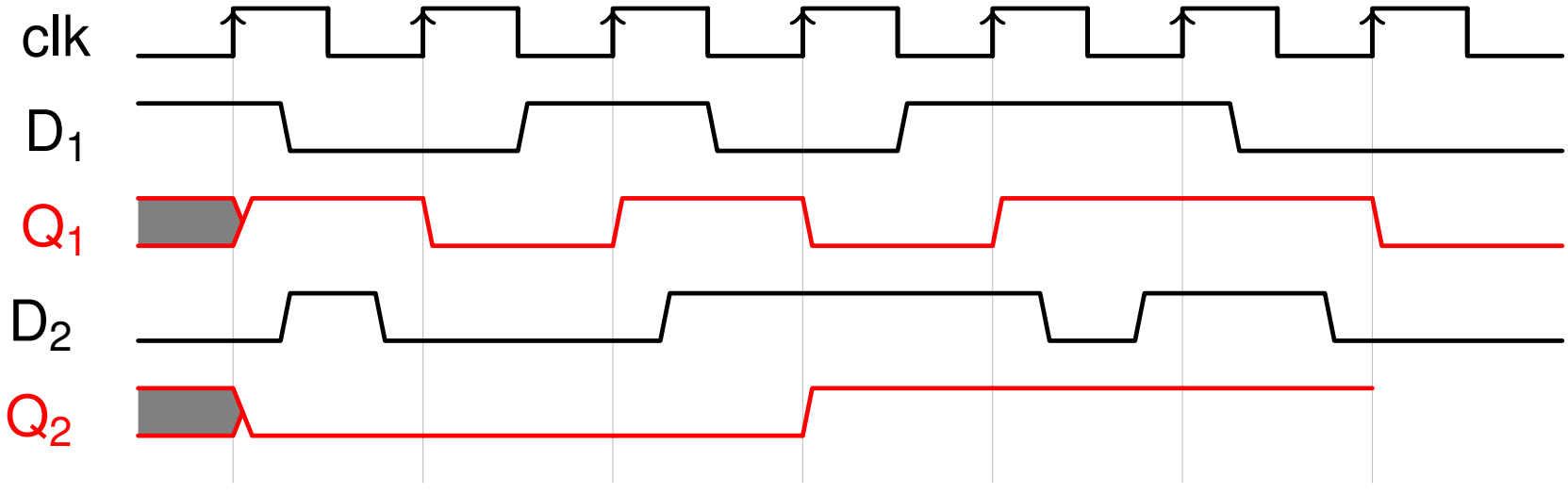
La bascule D

Exemple



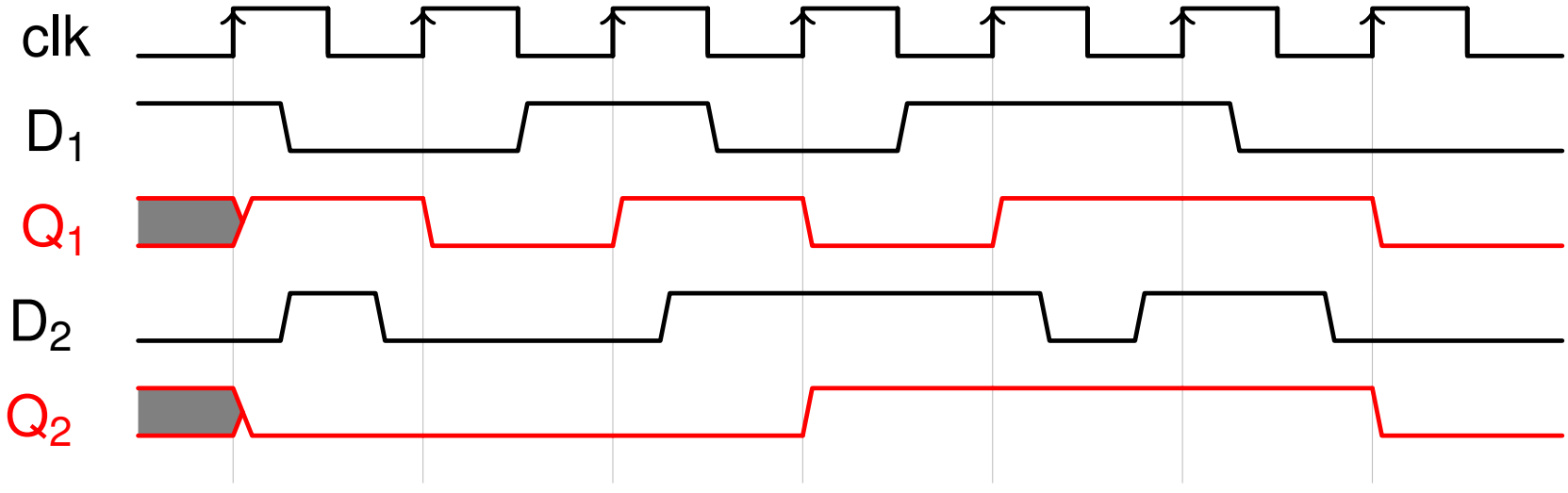
La bascule D

Exemple



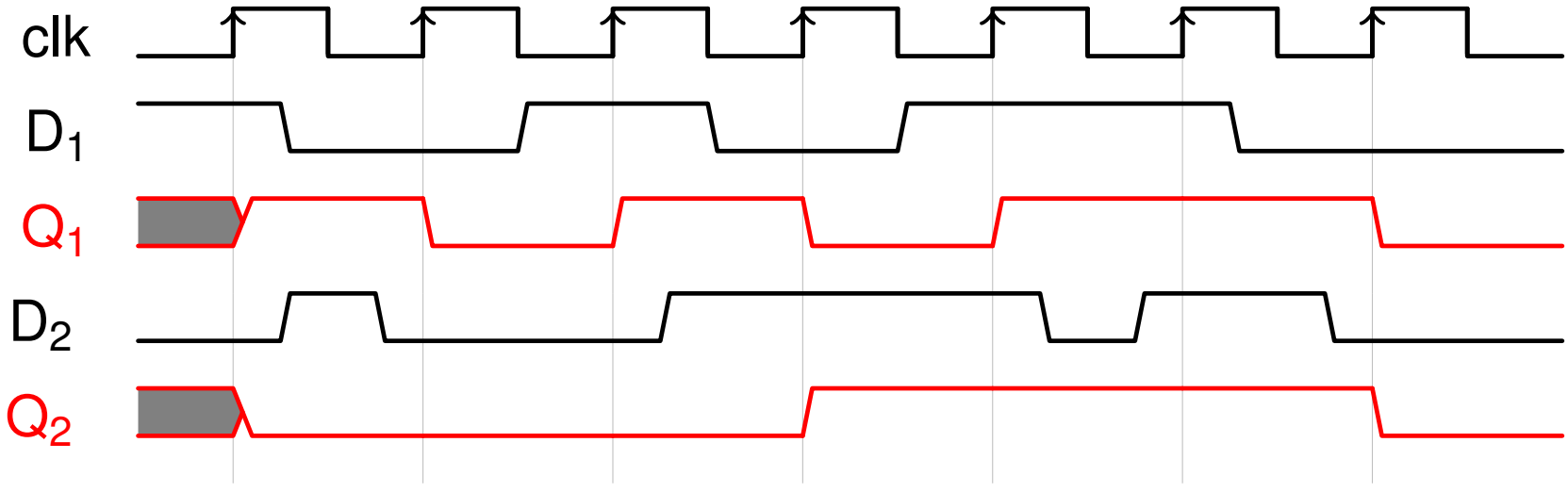
La bascule D

Exemple



La bascule D

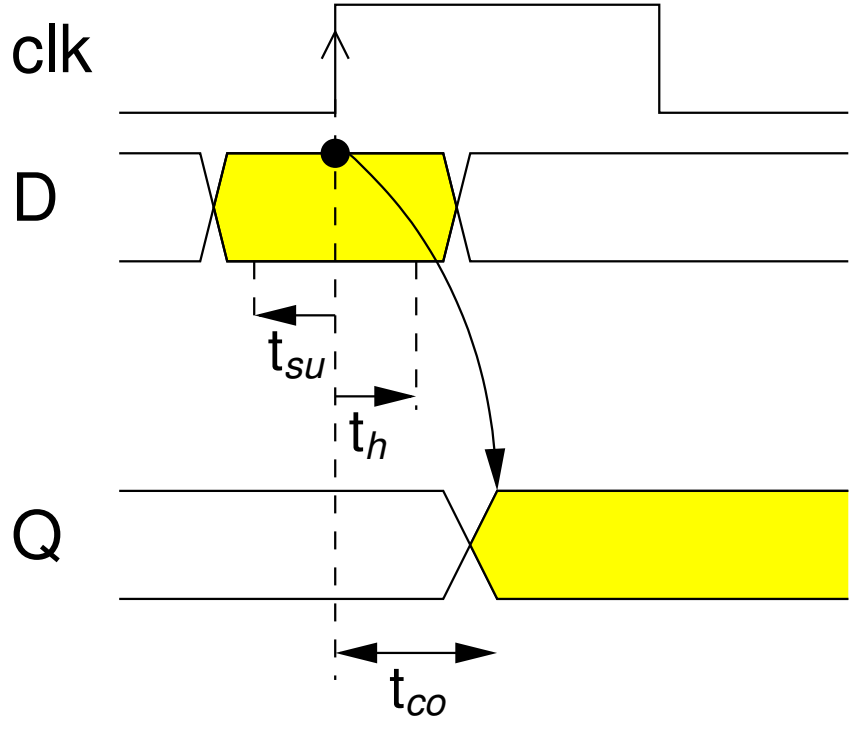
Exemple



- Q_1 recopie D_1 avec un cycle de retard.
- Q_2 recopie D_2 en filtrant les impulsions de durée inférieure à la période de l'horloge.

La bascule D

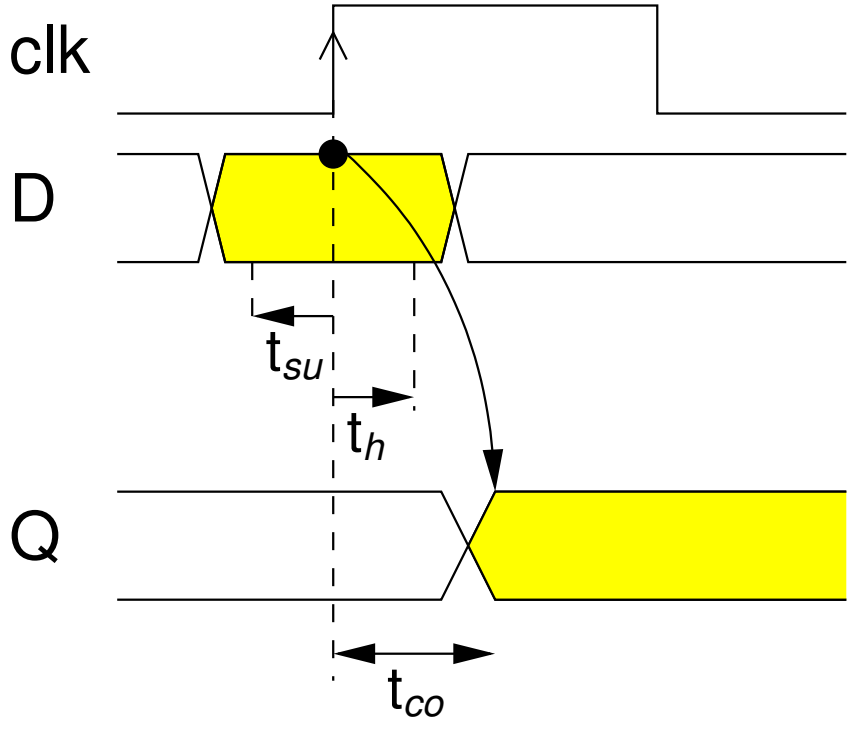
Contraintes temporelles



- La donnée doit être stable au front d'horloge :
  - avoir atteint sa valeur t_{su} avant le front d'horloge (setup).
  - cette valeur doit être maintenue t_h après le front d'horloge (hold).
- La copie de l'entrée sur la sortie se fait avec un retard de t_{co} (clock to output).

La bascule D

Contraintes temporelles



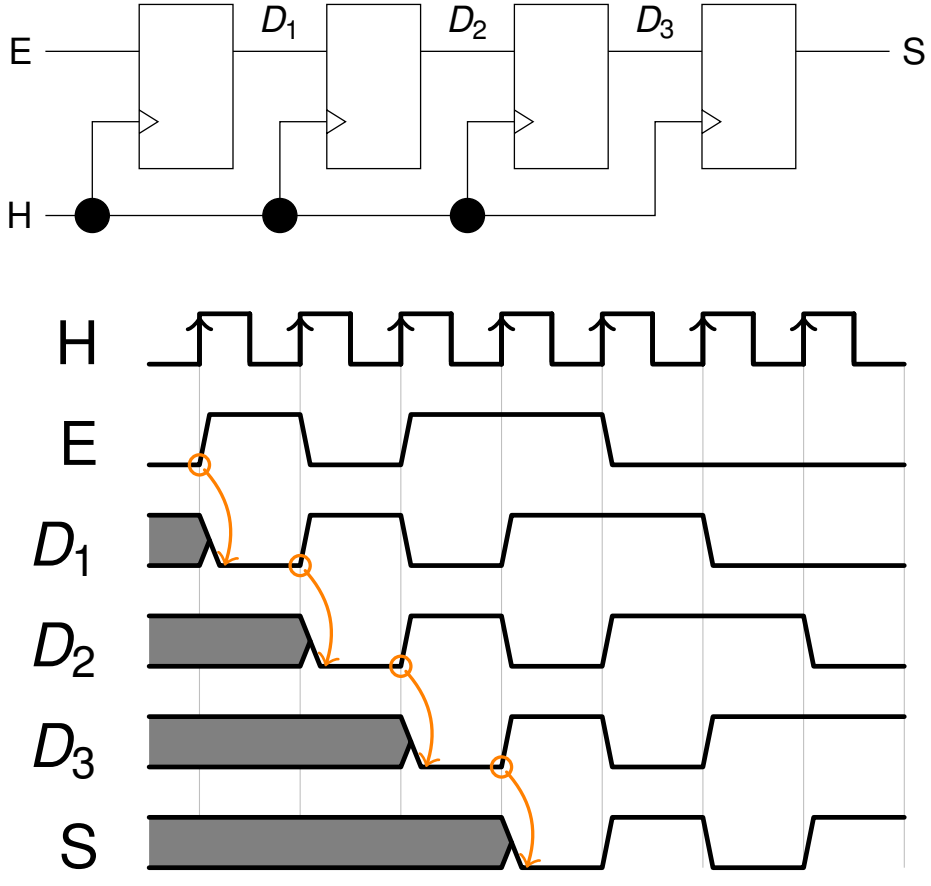
t_{su} : temps de pré–positionnement

t_h : temps de maintien

t_{co} : temps de propagation

Exemples d'applications

Registre à décalage



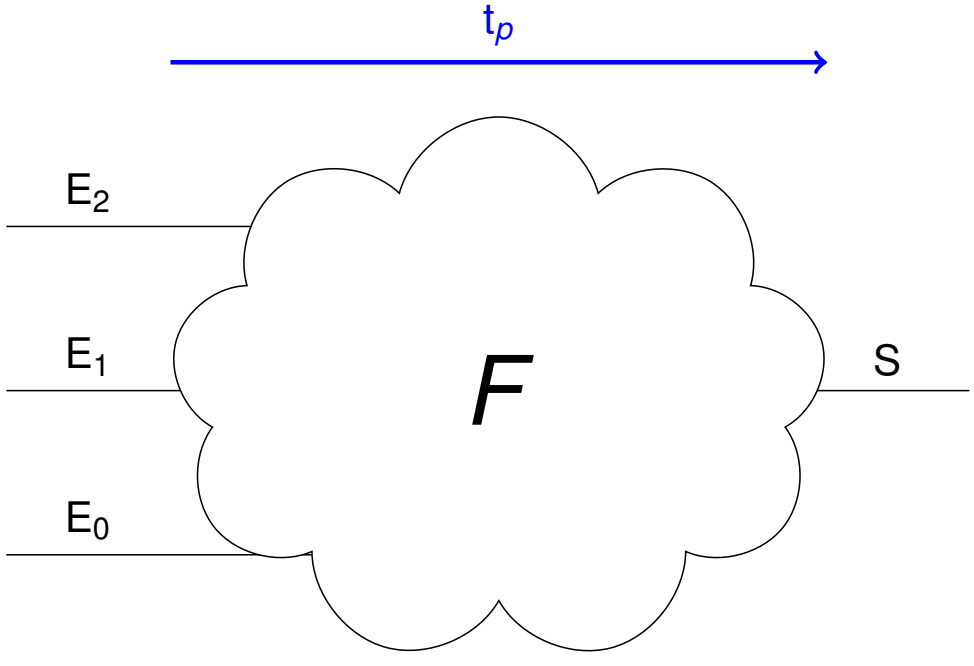
Plan

Rappel

- La sortie d'une fonction ne dépend que de ses entrées
  - Pour les mêmes entrées on obtient toujours les mêmes sorties.
- Permet de construire des opérateurs :
  - Logiques
  - Arithmétiques

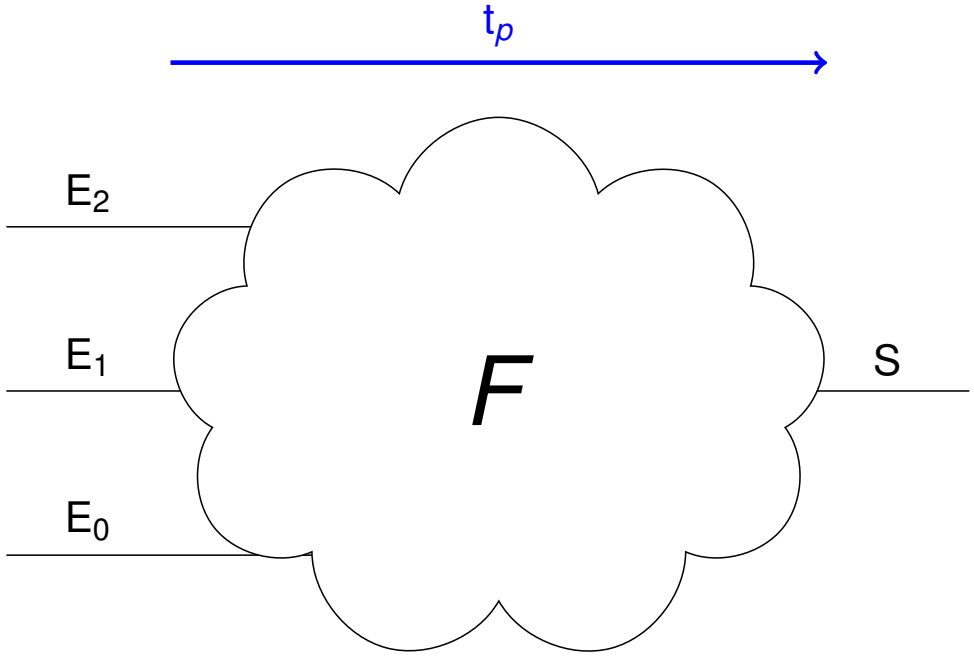
La logique séquentielle

- Le temps de propagation t_p est non nul. Durant ce temps :
  - La sortie n'est pas valide !
  - On ne peut pas modifier les entrées !
- Comment enchaîner plusieurs calculs ?



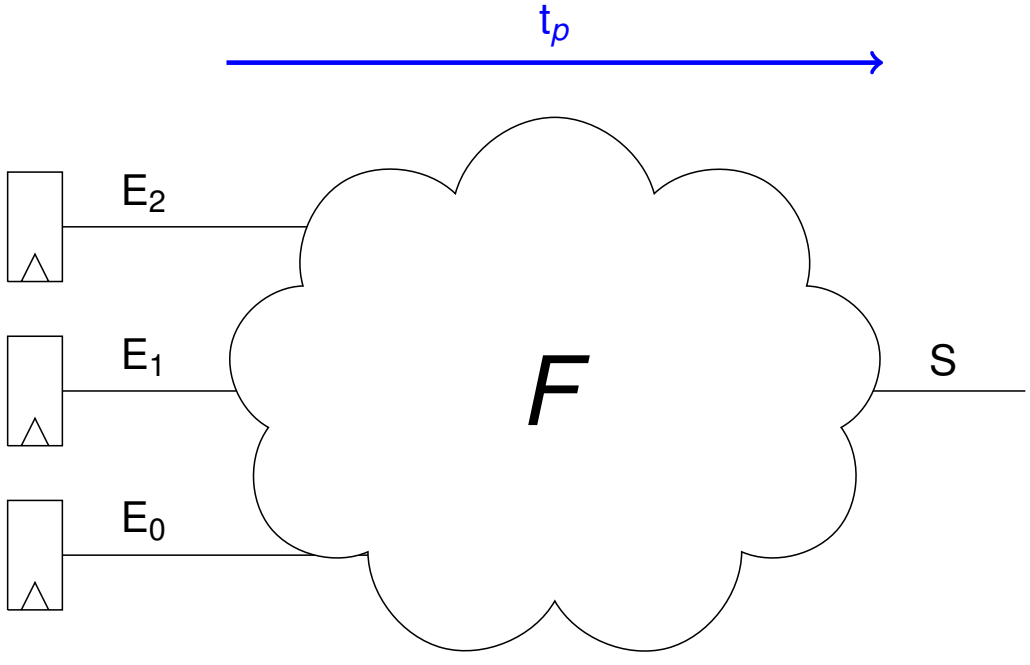
La logique séquentielle

- On maintient les entrées stables au moins pendant t_p



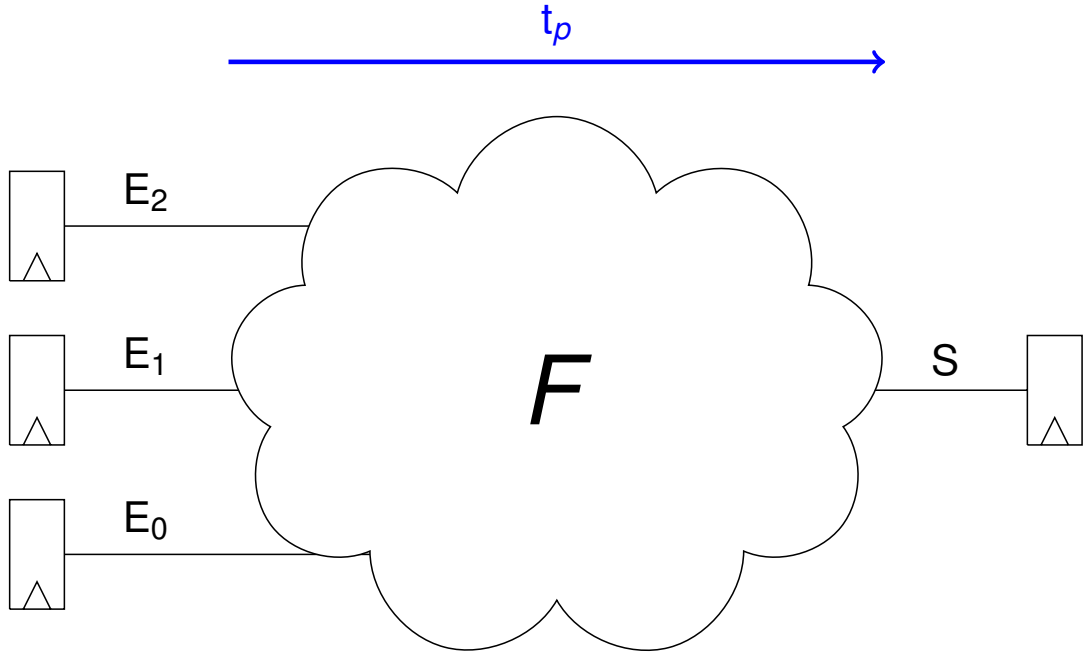
La logique séquentielle

- On maintient les entrées stables au moins pendant t_p
- En échantillonnant les entrées.



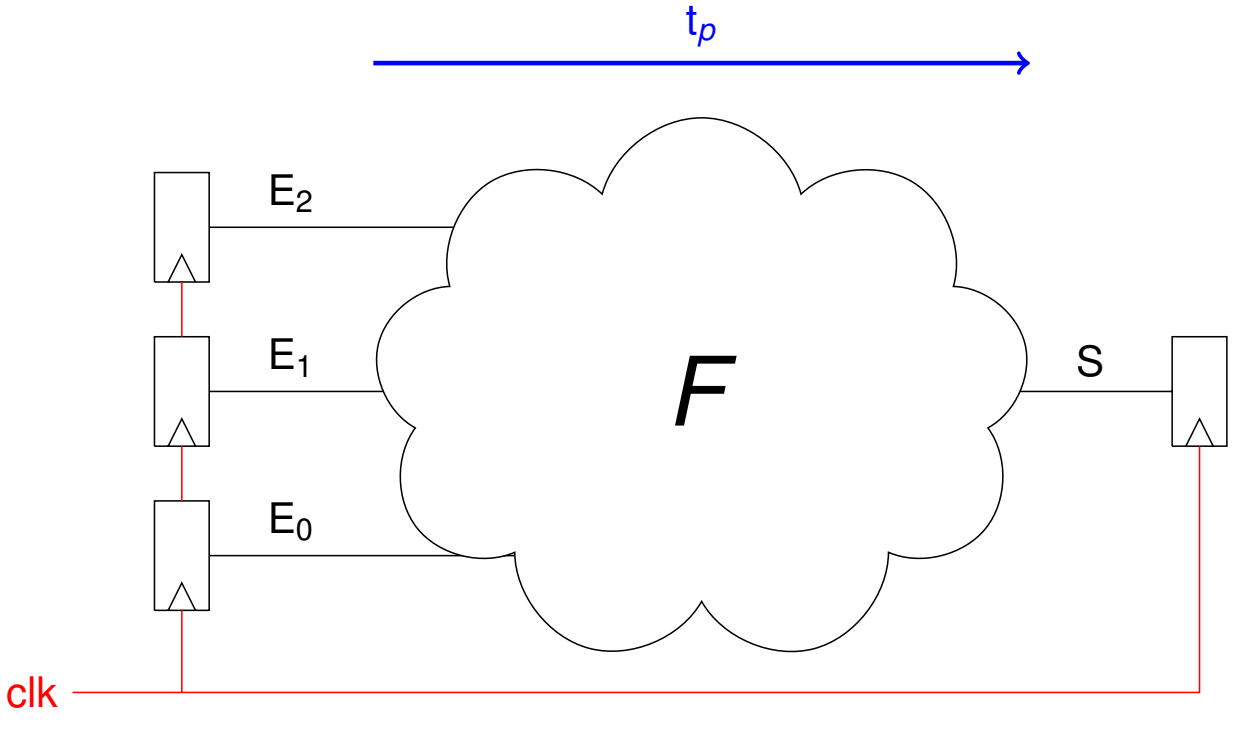
La logique séquentielle

- Dès que le résultat est prêt :
  - La sortie est échantillonnée
  - On peut au même instant modifier les entrées



La logique séquentielle

- On utilise donc le même signal de synchronisation.
  - La même horloge



La logique séquentielle

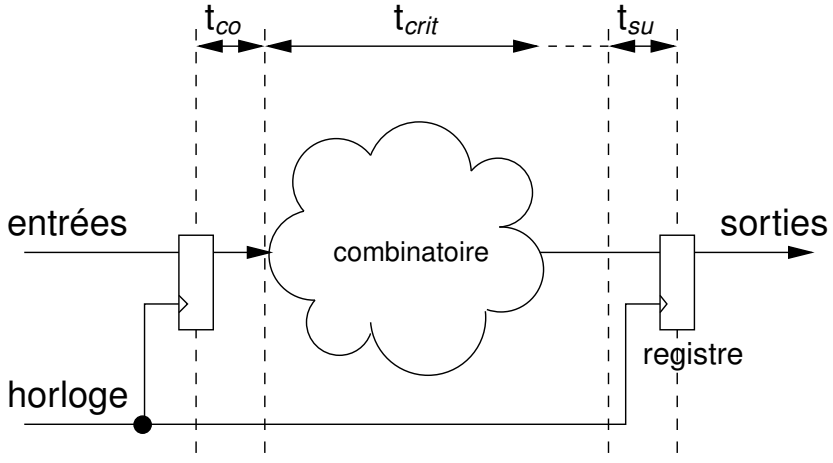
Règles

La logique séquentielle “synchrone”

- Tout bloc combinatoire est entouré par des registres (bascules D).
- Les registres sont synchrones
  - Ils utilisent le même signal d’horloge
  - L’horloge doit arriver en même temps
  - pas de combinatoire sur l’horloge

La logique séquentielle

Fréquence de fonctionnement

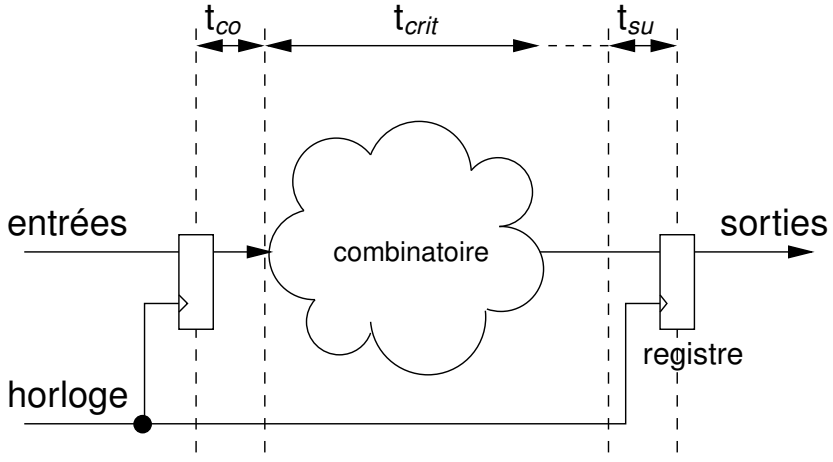


$$T_{clk} > t_{co} + t_{crit} + t_{su}$$

- t_{crit} est le temps du chemin critique càd le temps de propagation du chemin combinatoire le plus long

La logique séquentielle

Fréquence de fonctionnement

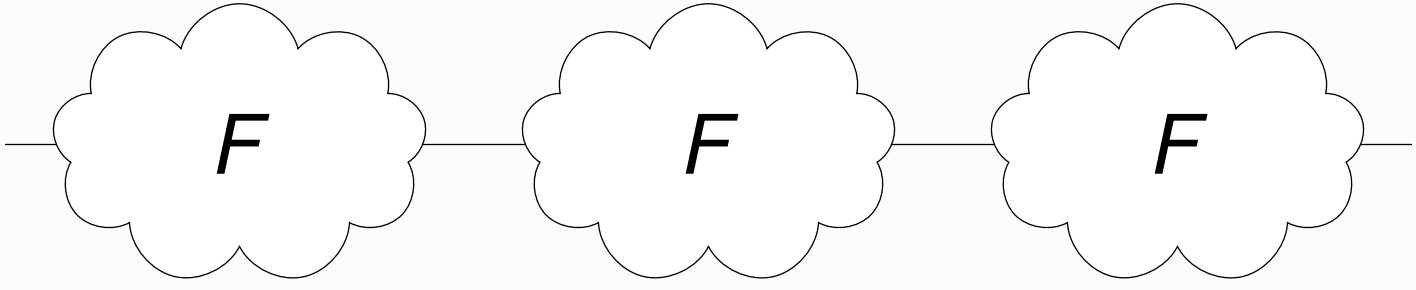


$$F_{max} = \frac{1}{t_{co} + t_{crit} + t_{su}}$$

- t_{crit} est le temps du chemin critique càd le temps de propagation du chemin combinatoire le plus long

Séquencer les traitements

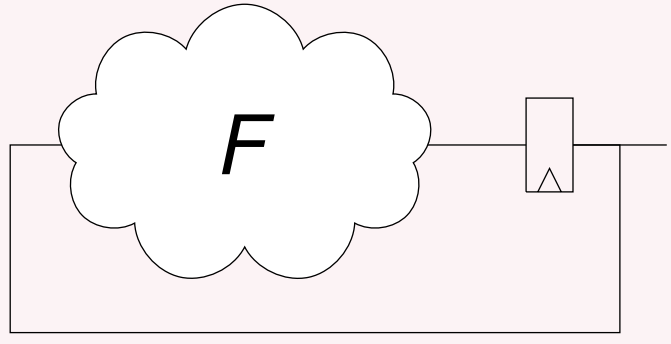
Répéter plusieurs fois le même calcul



- Comment effectuer plusieurs fois le même calcul sans dupliquer les opérateurs ?

Séquencer les traitements

Mémoriser et réutiliser le même bloc combinatoire



- Mémoriser les résultats intermédiaires.
- Reboucler sur la partie combinatoire.

Séquencer les traitements

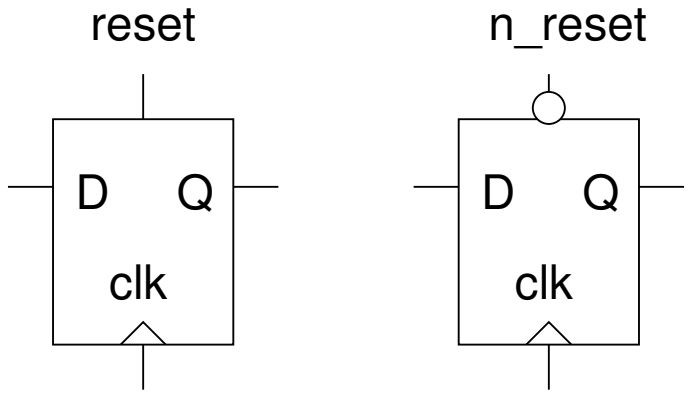
La valeur initiale ?

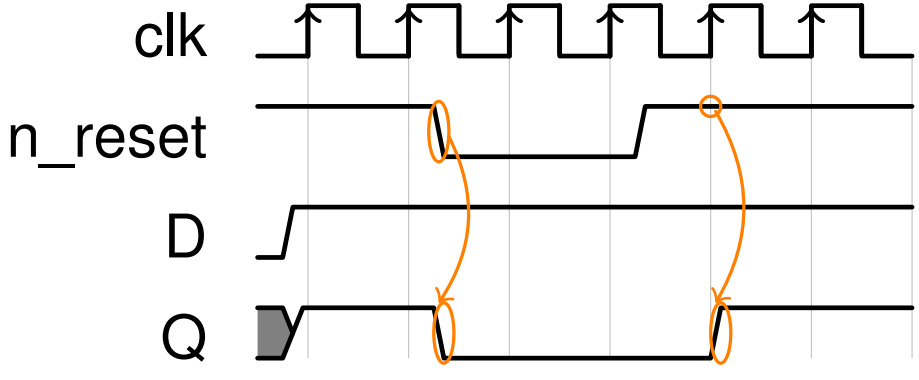
- Un signal extérieur est utilisé pour forcer l'état des bascules.
- On parle de "reset" (remise à zéro).
- Il peut s'agir d'une mise à "1", on parle alors de "preset".

Séquencer les traitements

Le reset asynchrone

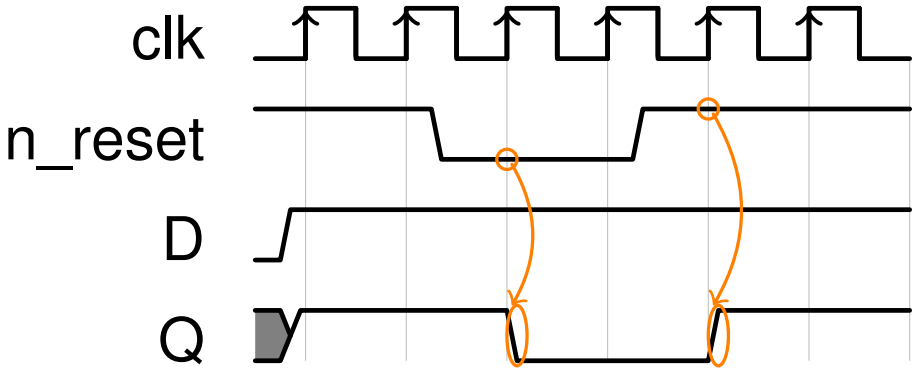
- Un signal connecté directement aux bascules.
- Son action est indépendante de l'horloge.
- En général global pour tout le circuit.
- Peut être positif ou négatif.





Séquencer les traitements

Le reset synchrone



- Vient de la logique qui précède les bascules.
- Son action n'est effective qu'au front de l'horloge.
- C'est une remise à zéro fonctionnelle.

Séquencer les traitements

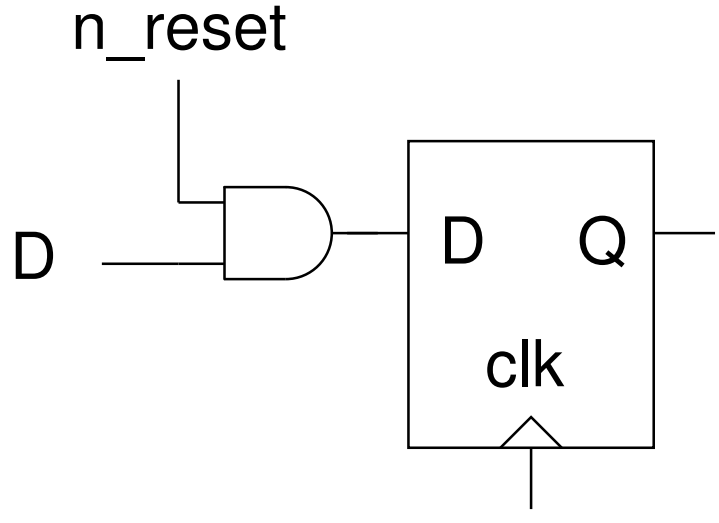
Le reset synchrone

- Comment construire un reset synchrone en utilisant les portes logiques de base ?

Séquencer les traitements

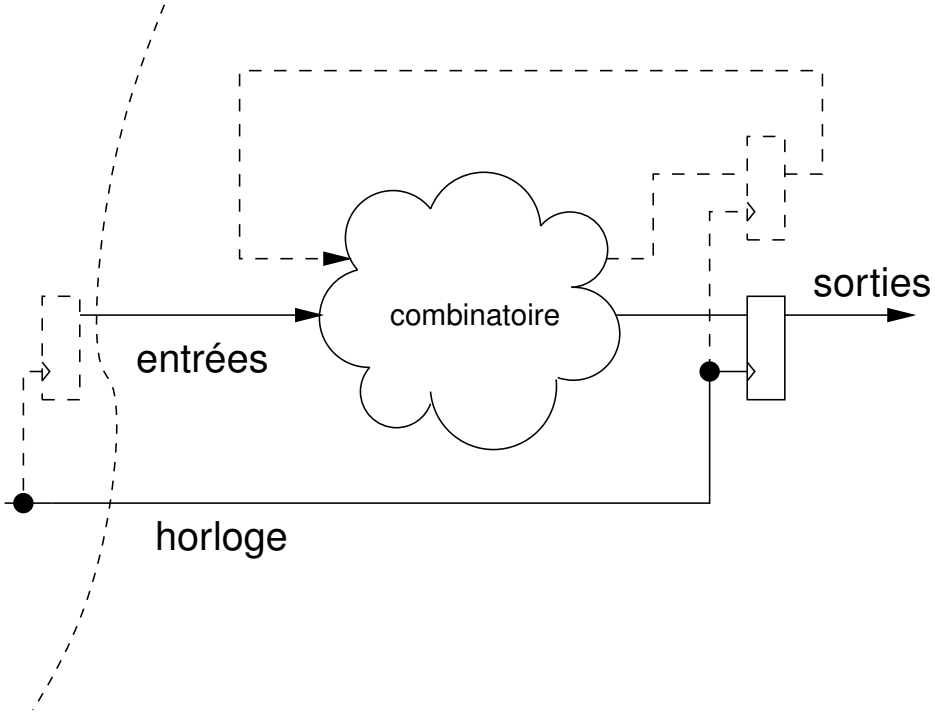
Le reset synchrone

- Comment construire un reset synchrone en utilisant les portes logiques de base ?



Logique séquentielle synchrone

Résumons



- Utilisation de bascules D synchrones pour mémoriser les variables internes et les sorties
- La sortie d'une fonction séquentielle dépend de ses entrées et de son état précédent.
- L'état initial doit être forcé par un signal extérieur.

Plan

Exemples d'applications

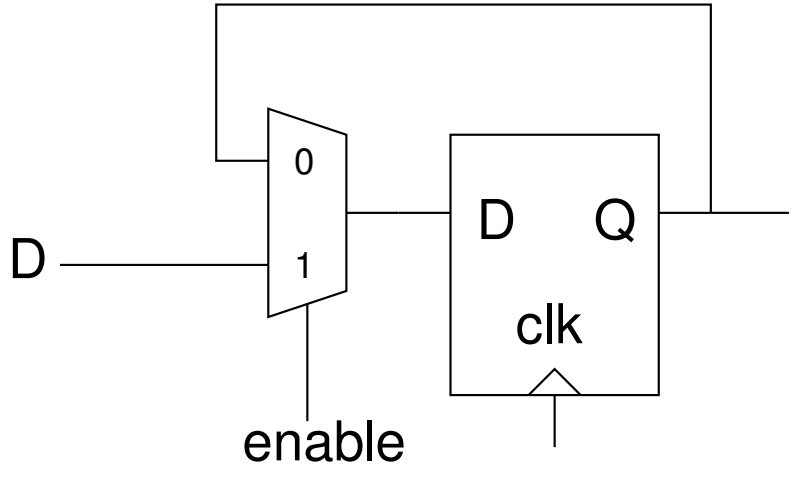
L'enable

- Construire une bascule D avec une entrée d'activation (enable).
  - enable vaut 1 la bascule fonctionne normalement.
  - enable vaut 0 la bascule ne change pas d'état.

Exemples d'applications

L'enable

- Construire une bascule D avec une entrée d'activation (enable).
  - enable vaut 1 la bascule fonctionne normalement.
  - enable vaut 0 la bascule ne change pas d'état.



Exemples d'applications

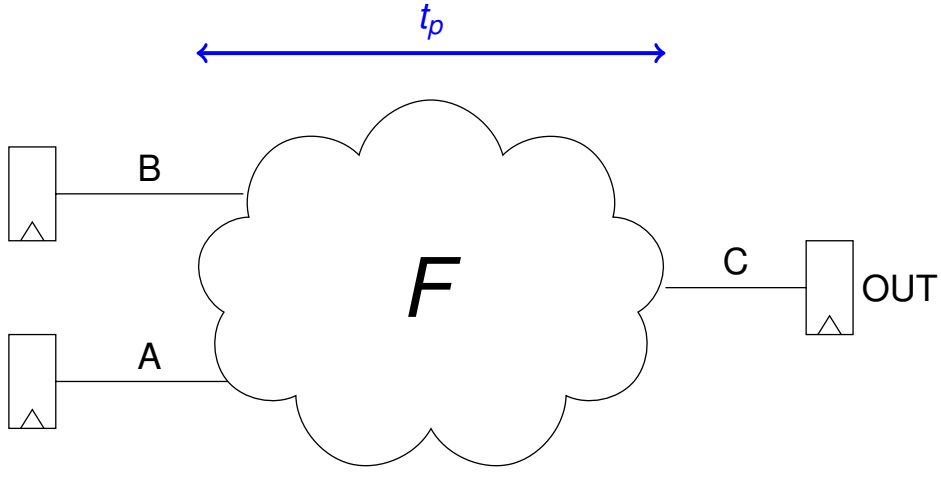
Un compteur

- Construisez un compteur ($+1$ à chaque coup d'horloge) de 0 à 255.
- Ajoutez la possibilité de le remettre à zéro
  - de façon asynchrone
  - de façon synchrone
- Ajoutez la possibilité d'interrompre/reprendre le comptage.
- Ajoutez la possibilité qu'il compte ou décompte.

Exemples d'applications

Le Pipeline

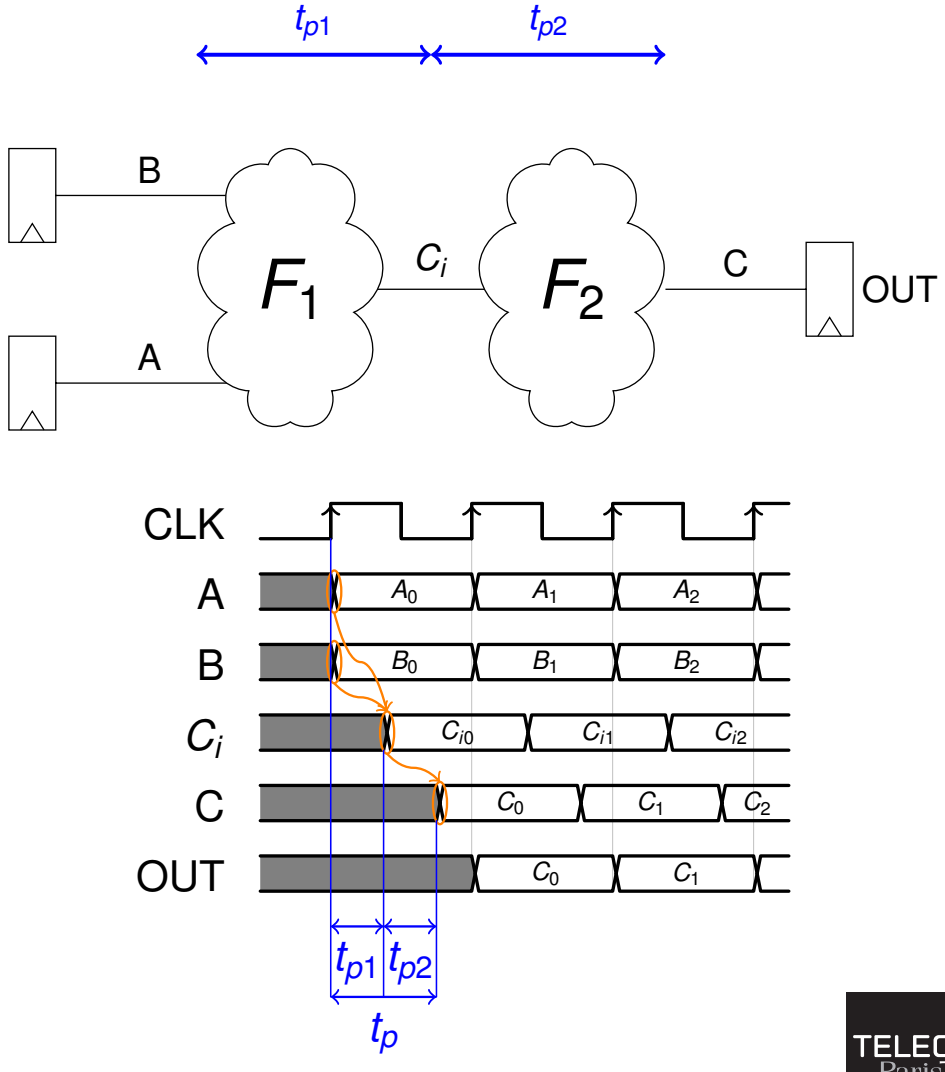
- Une fonction combinatoire F de temps de propagation t_p
- Les entrées sorties peuvent être synchrones tant que $t_p < T_{CLK}$
  - Où T_{CLK} est la période d'horloge



Exemples d'applications

Le Pipeline

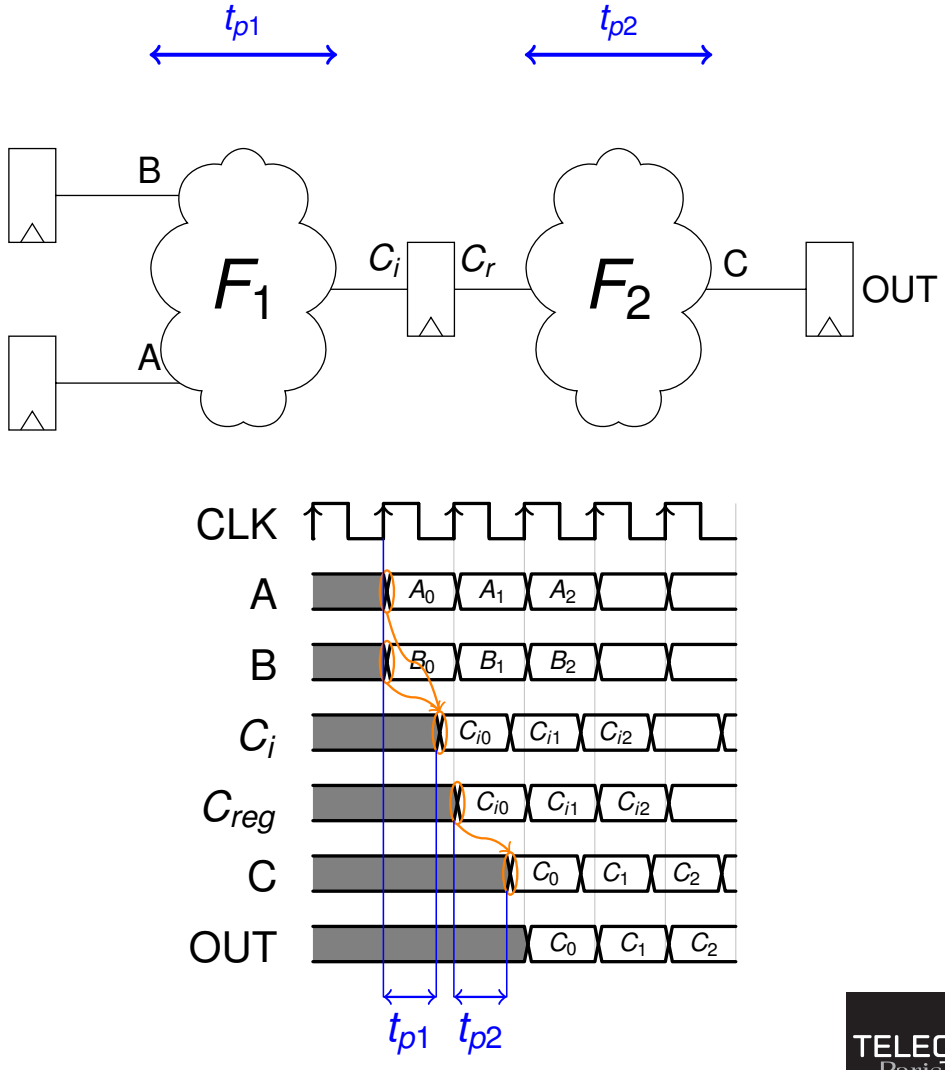
- On décompose F en deux fonctions combinatoires F_1 et F_2 de temps de propagation respectifs t_{p1} et t_{p2}
  - On s'arrange pour avoir $t_{p1} < t_p$ et $t_{p2} < t_p$
  - Dans cet exemple $t_{p1} + t_{p2} = t_p$
- Les entrées sorties peuvent être synchrones tant que $t_{p1} + t_{p2} < T_{CLK}$



Exemples d'applications

Le Pipeline

- On peut échantillonner la sortie de la fonction F_1
- Les entrées sorties peuvent être synchrones tant que $t_{p1} < T_{CLK}$ et $t_{p2} < T_{CLK}$
  - On peut donc réduire la période de l'horloge
  - Ou augmenter la fréquence



Exemples d'applications

Le Pipeline

- Le pipeline permet d'augmenter la fréquence de fonctionnement.
- On a augmenté la taille du circuit (Ajout des bascules, modification de la partie combinatoire).
- On a augmenté la latence initiale.