TELECOM ParisTech

INSTITUT Mines-Télécom

ELECINF 102 : Processeurs et Architectures Numériques

Logique séquentielle

Yves Mathieu
yves.mathieu@telecom-paristech.fr

Plan

La logique combinatoire

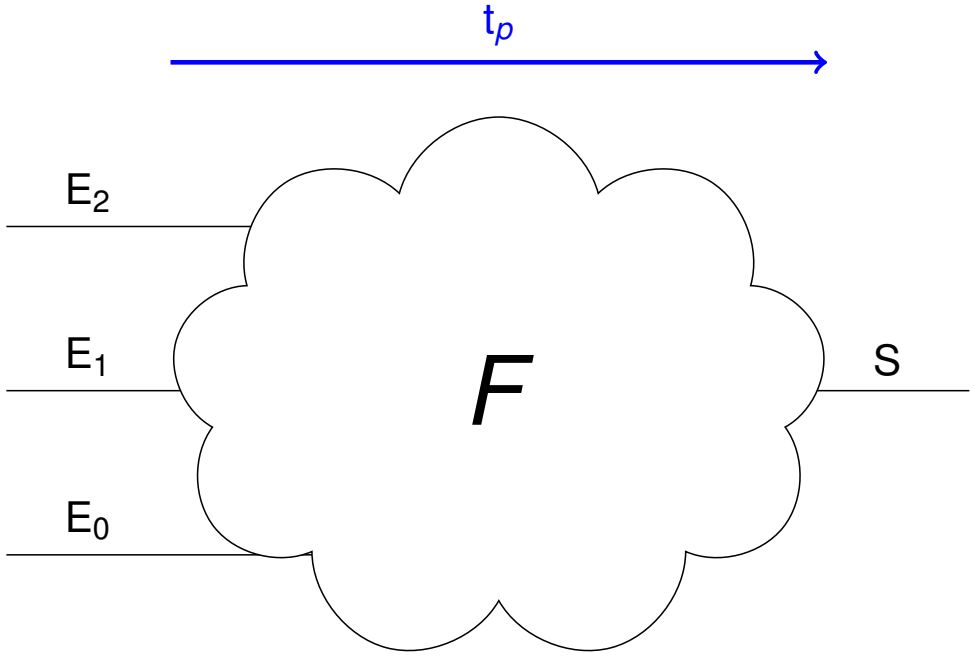
Rappel

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

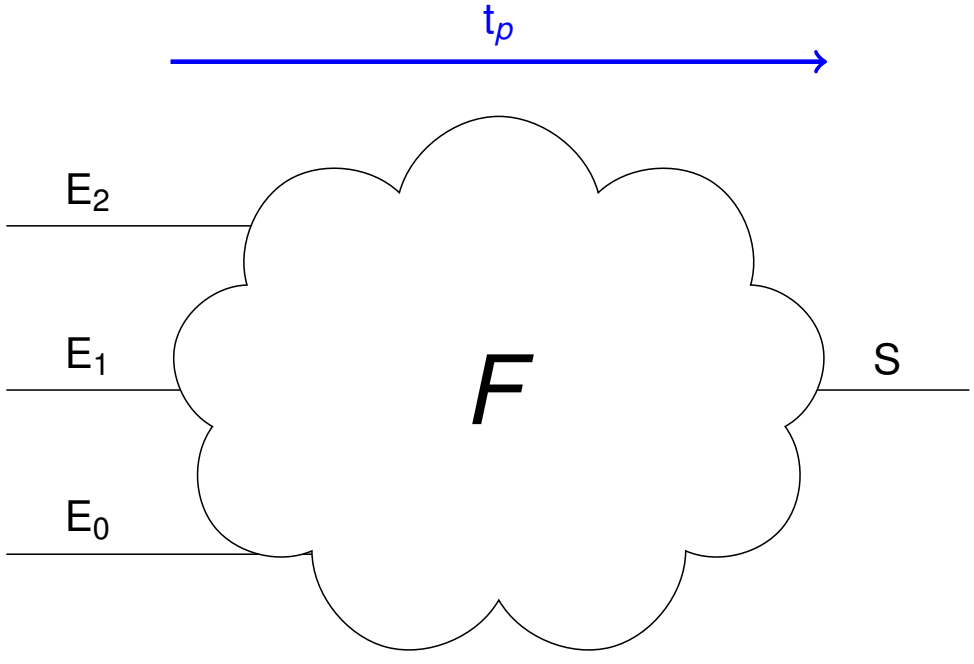
La logique séquentielle

- 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 ?



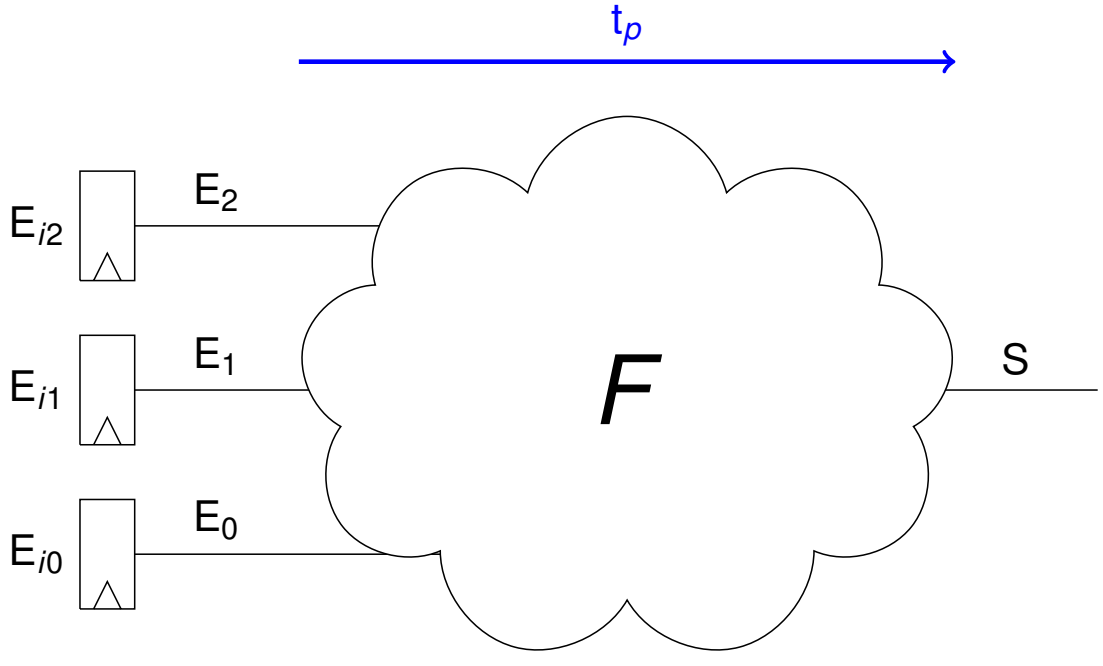
La logique séquentielle

- On maintient les entrées stables au moins pendant t_p



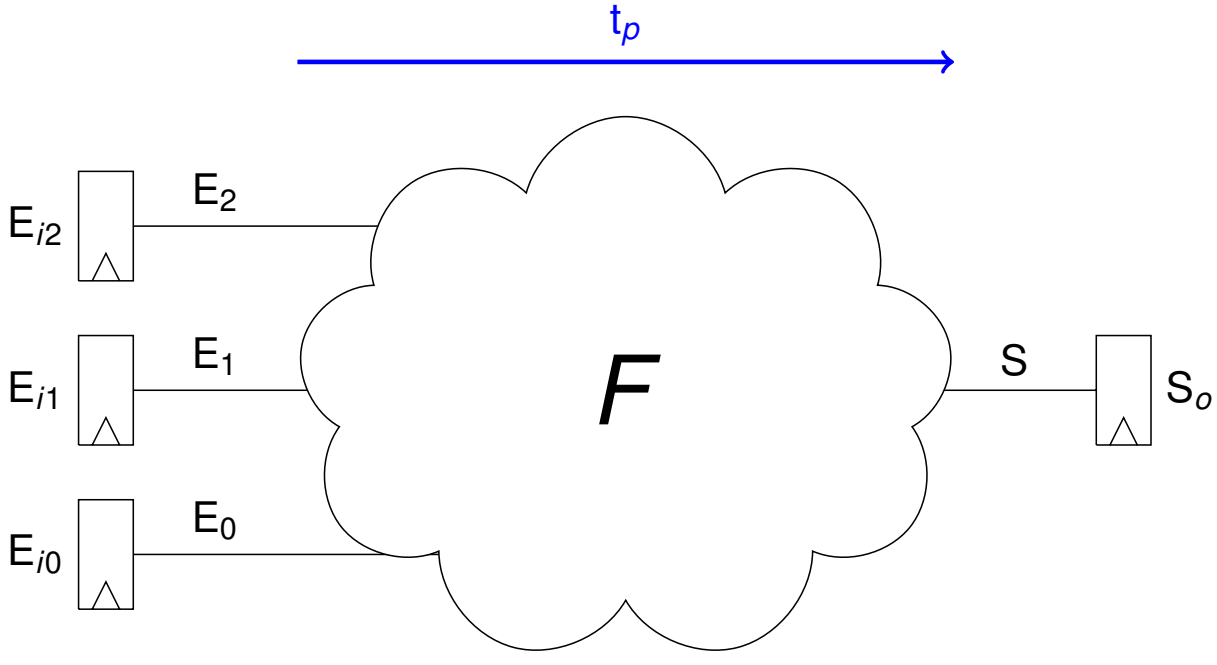
La logique séquentielle

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



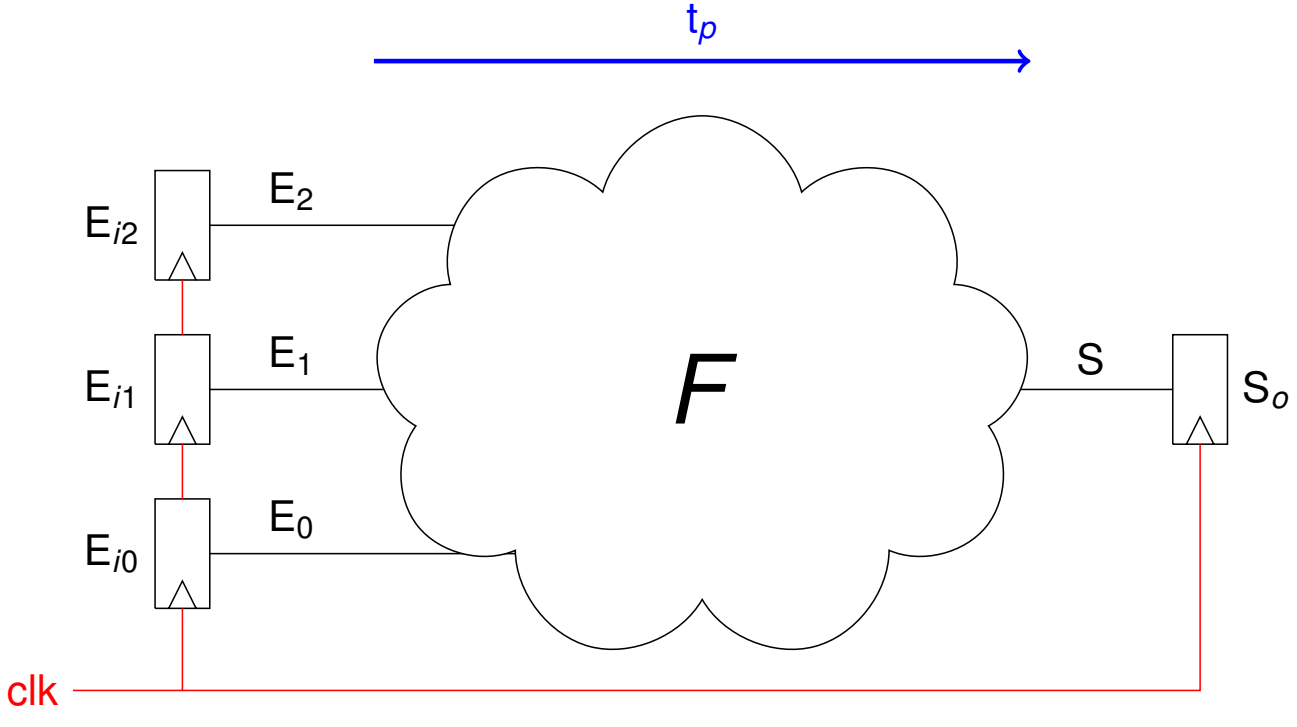
La logique séquentielle

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

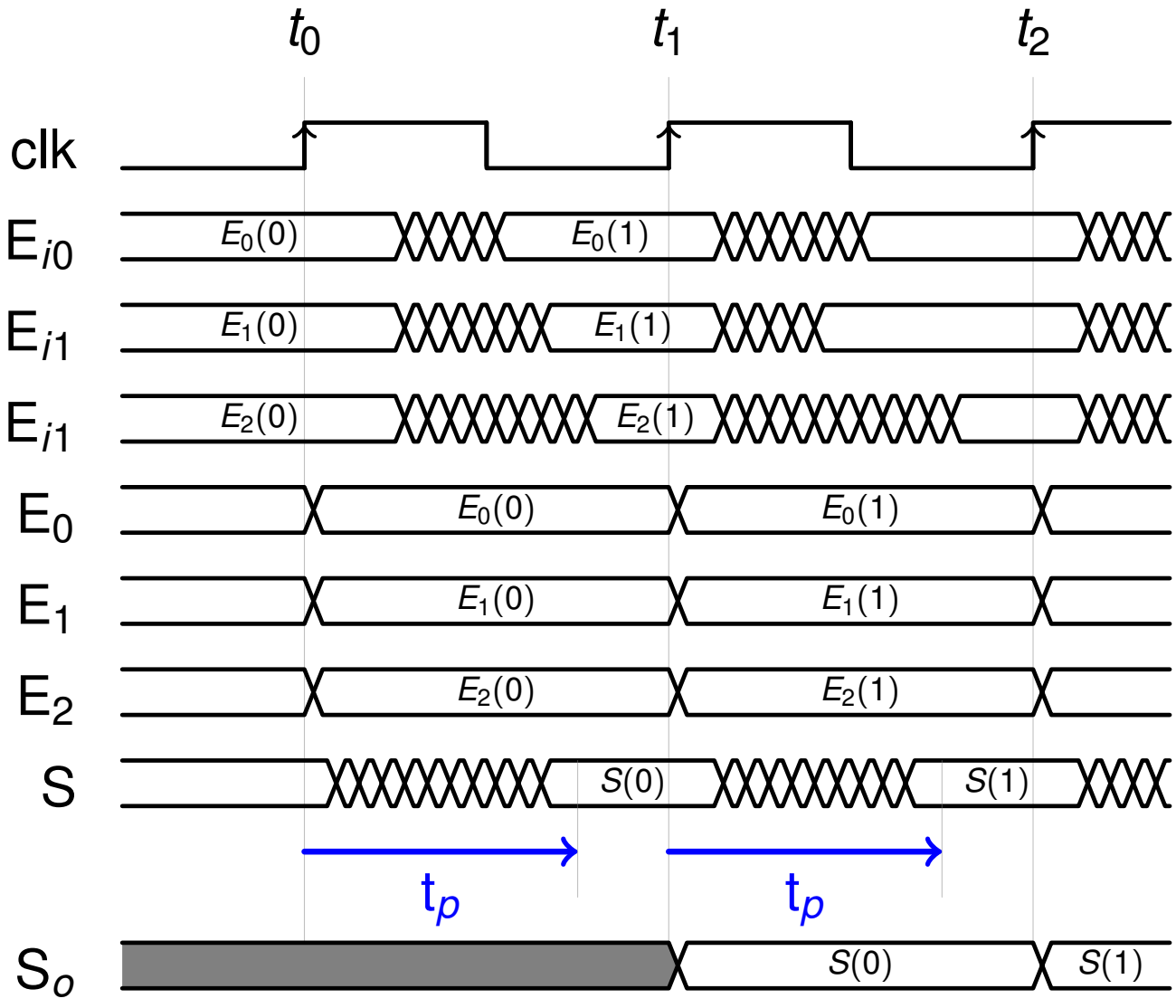


La logique séquentielle

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



La logique séquentielle



La logique séquentielle

Il manque quelque chose !

Nous avons besoin d'un composant mémorisant sur le front d'un signal

La logique séquentielle

Il manque quelque chose !

Nous avons besoin d'un composant mémorisant sur le front d'un signal

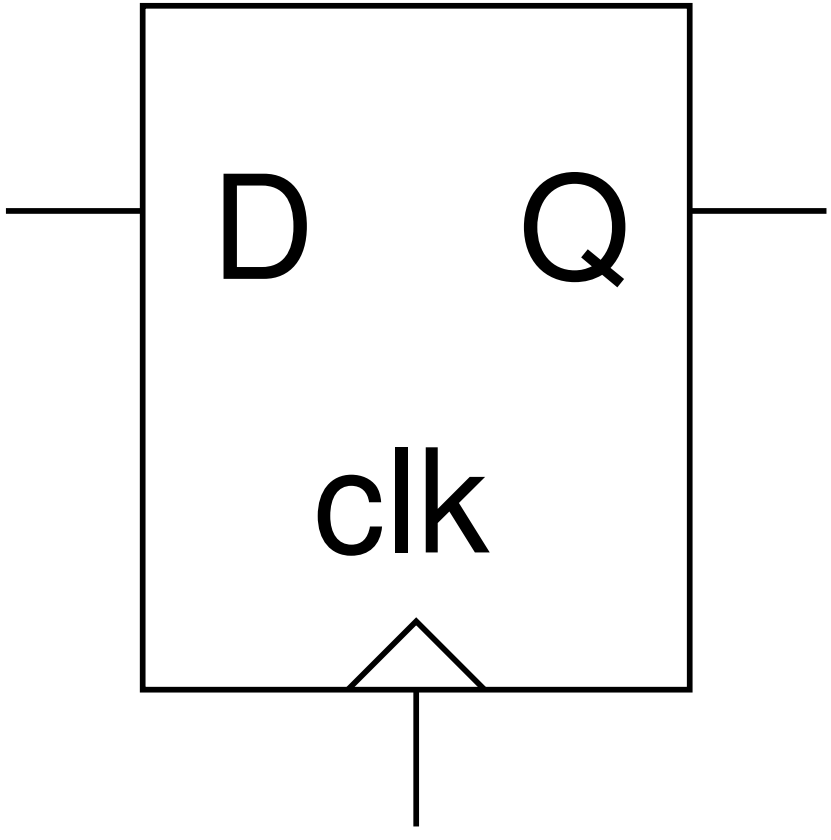
- Ce composant s'appelle une bascule D (*D flip-flop*)

C'est ce que nous verrons dans ce qui suit...

Plan

La bascule D

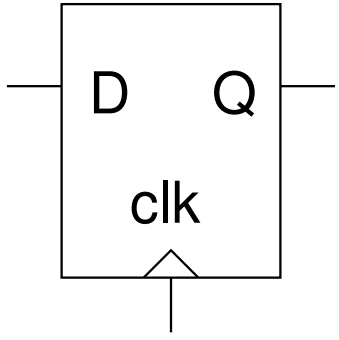
Élément de base de la logique séquentielle



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

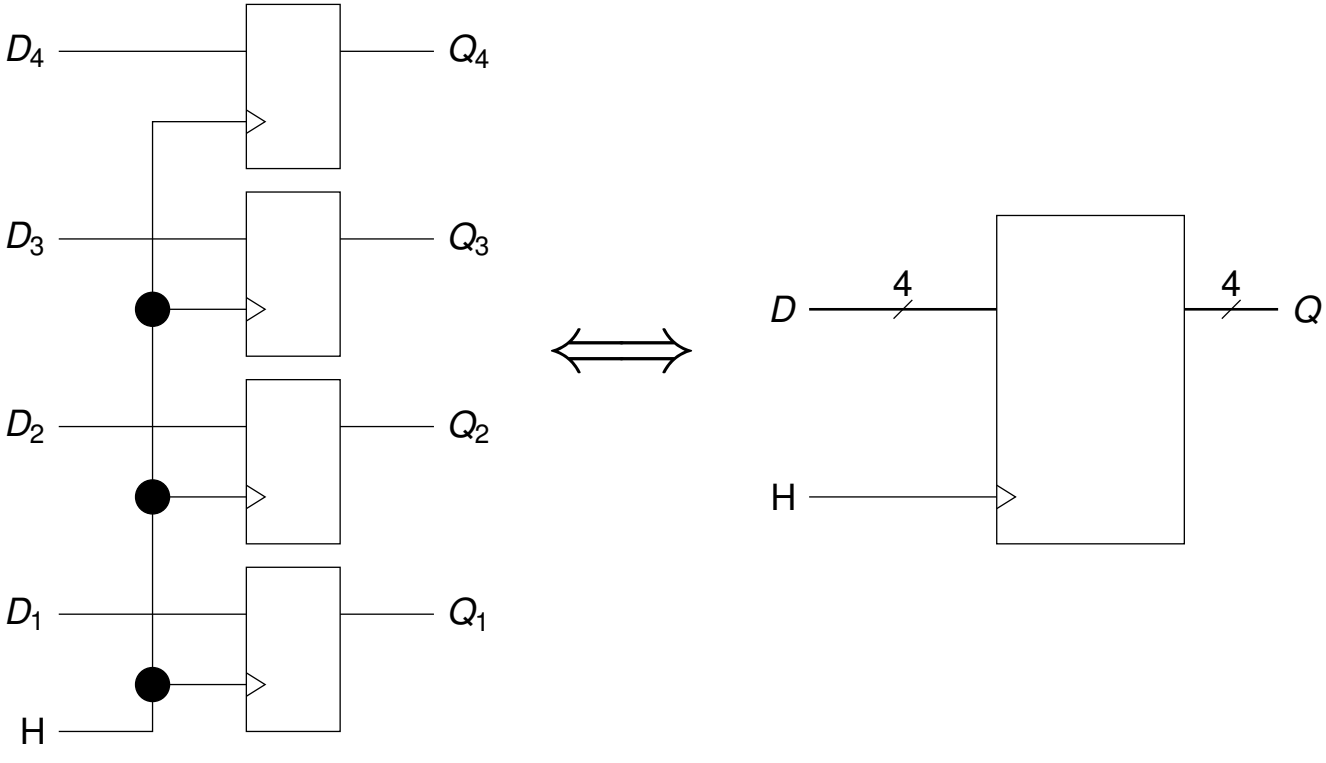
Table de vérité de la bascule D

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

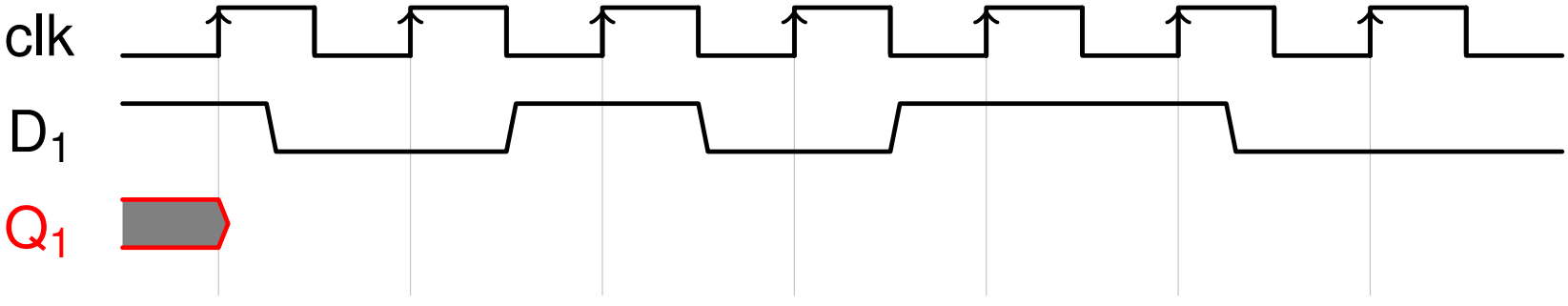
Le registre

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



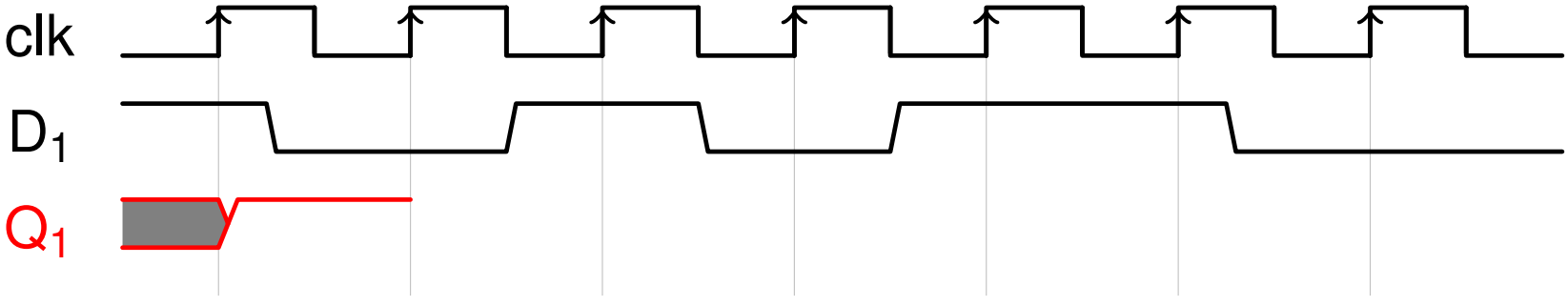
La bascule D

Exemple



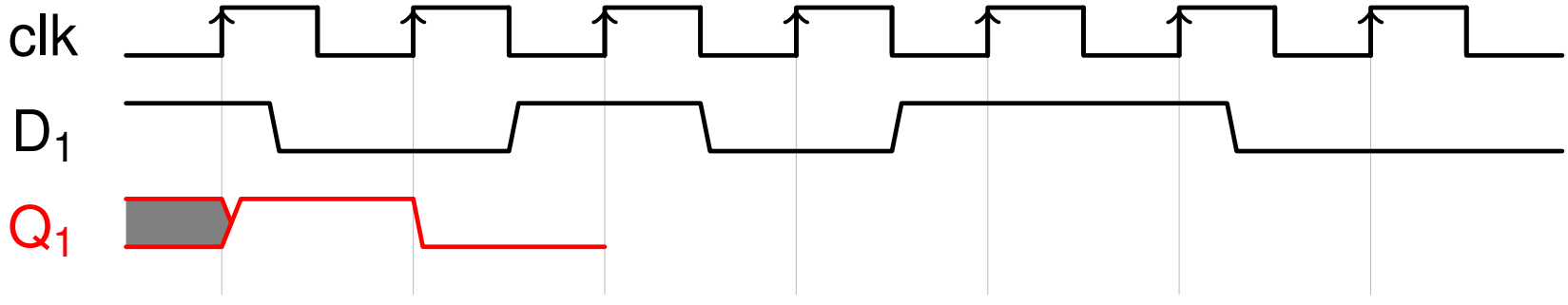
La bascule D

Exemple



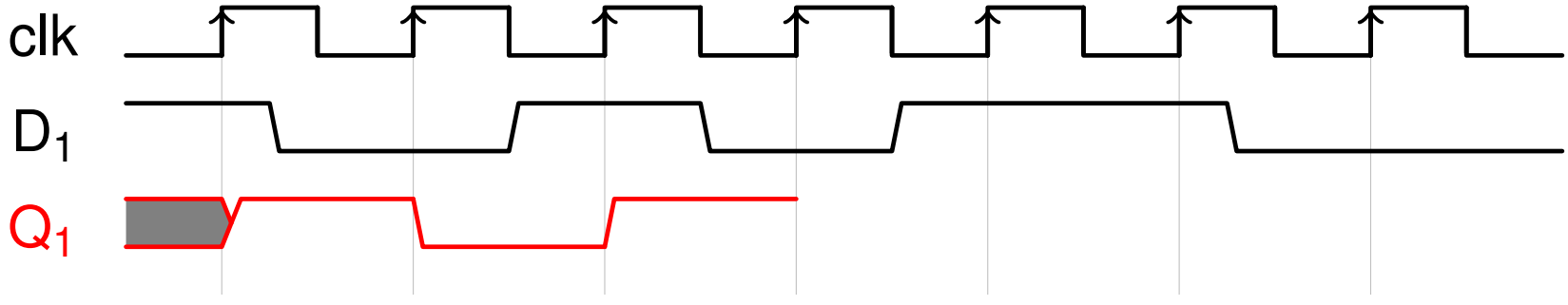
La bascule D

Exemple



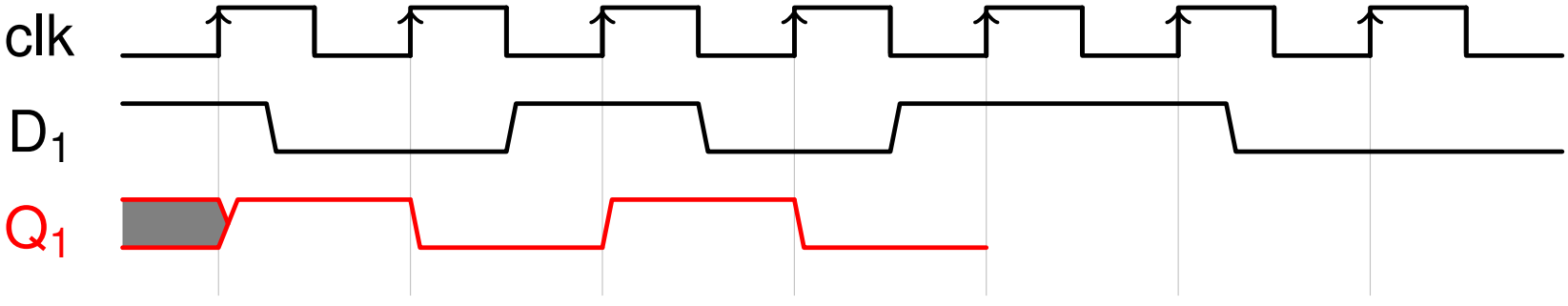
La bascule D

Exemple



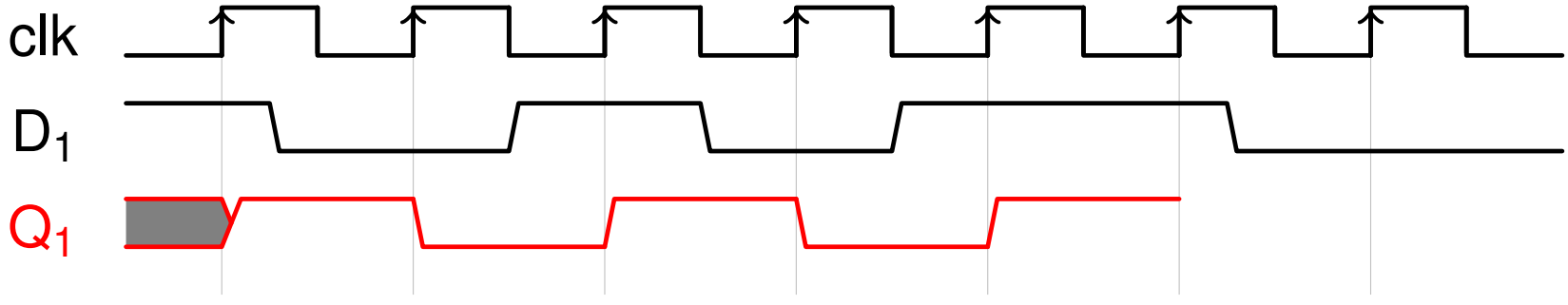
La bascule D

Exemple



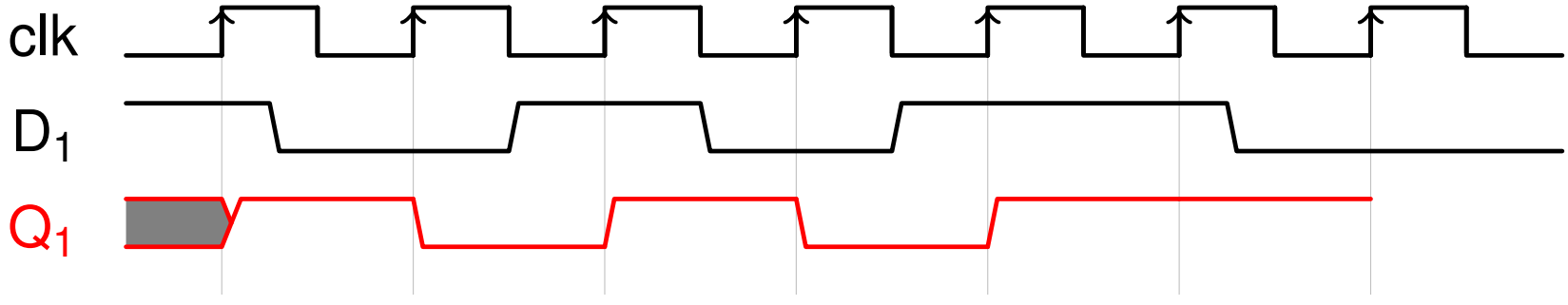
La bascule D

Exemple



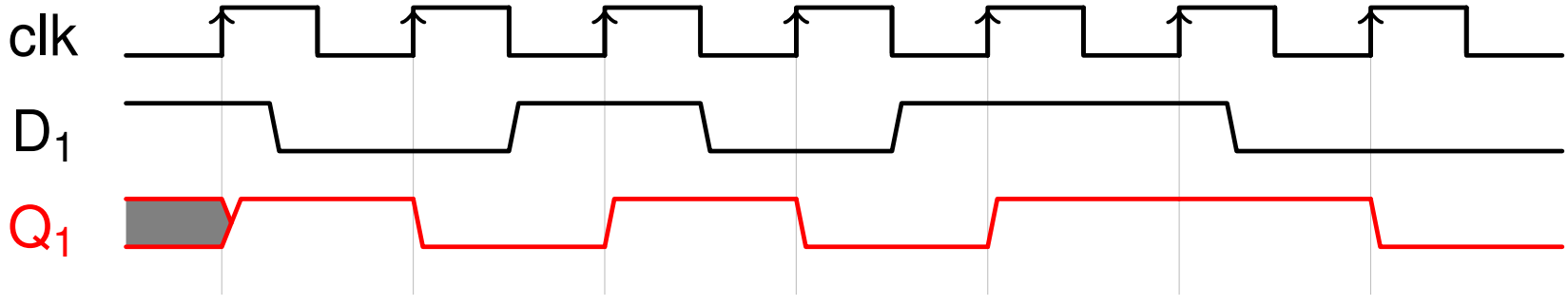
La bascule D

Exemple



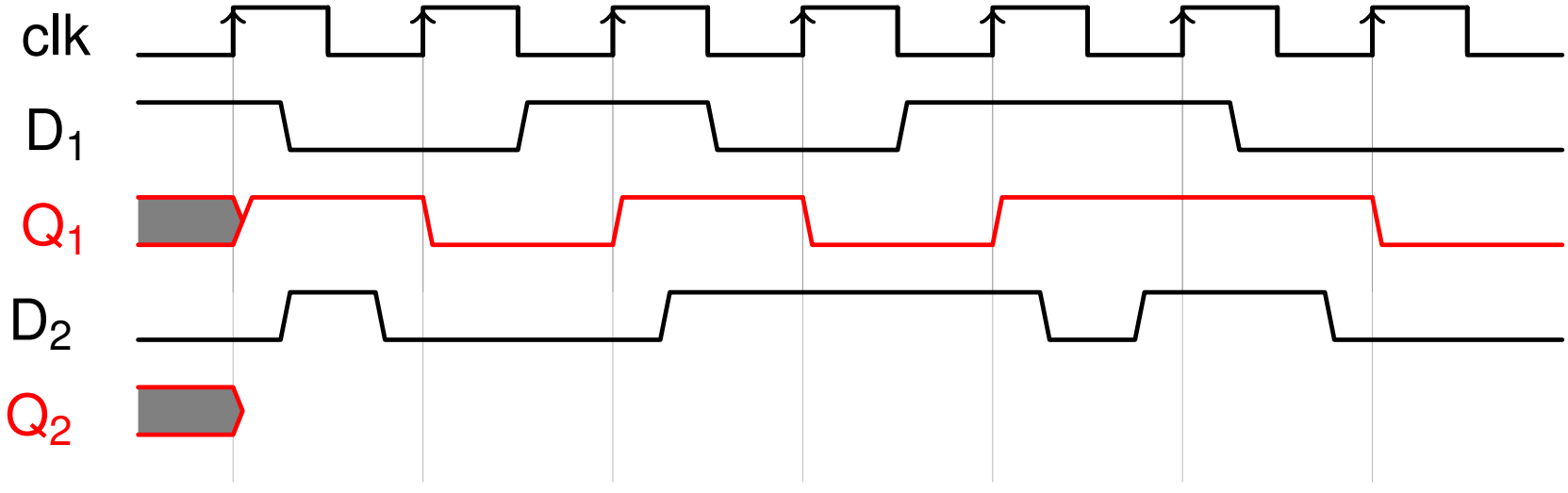
La bascule D

Exemple



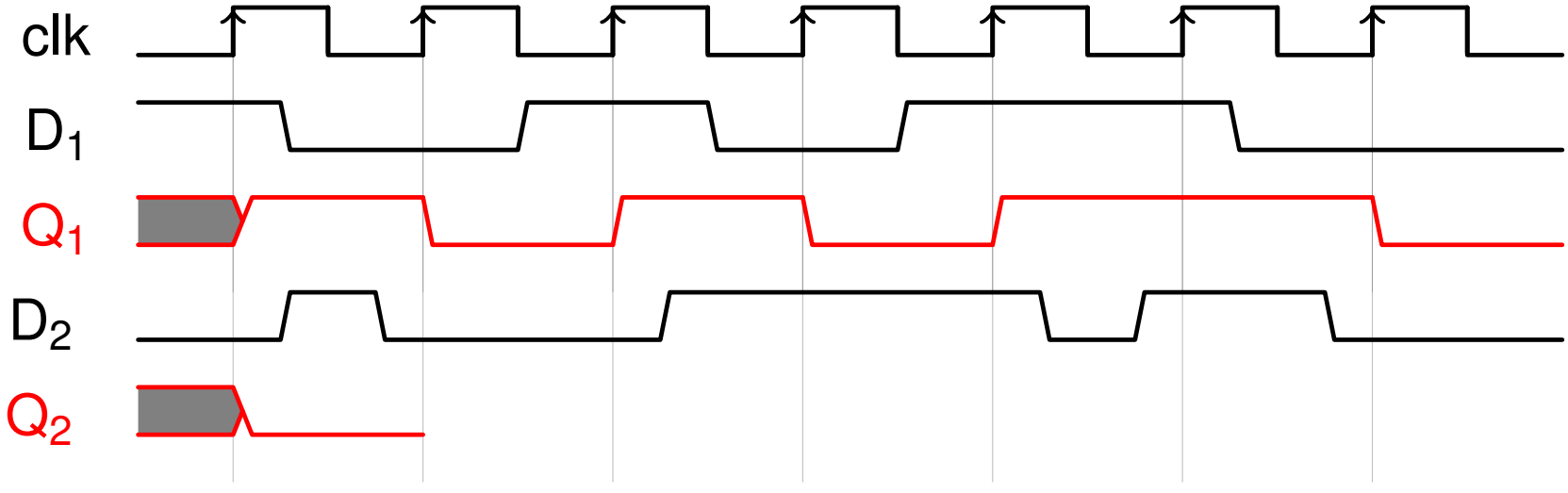
La bascule D

Exemple



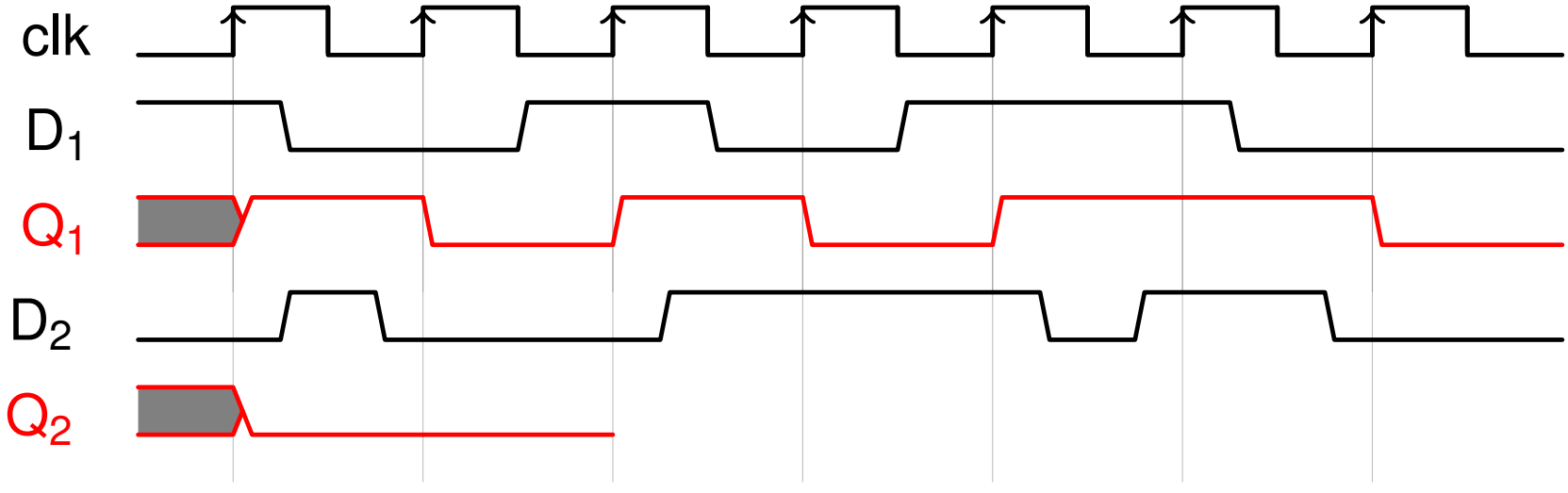
La bascule D

Exemple



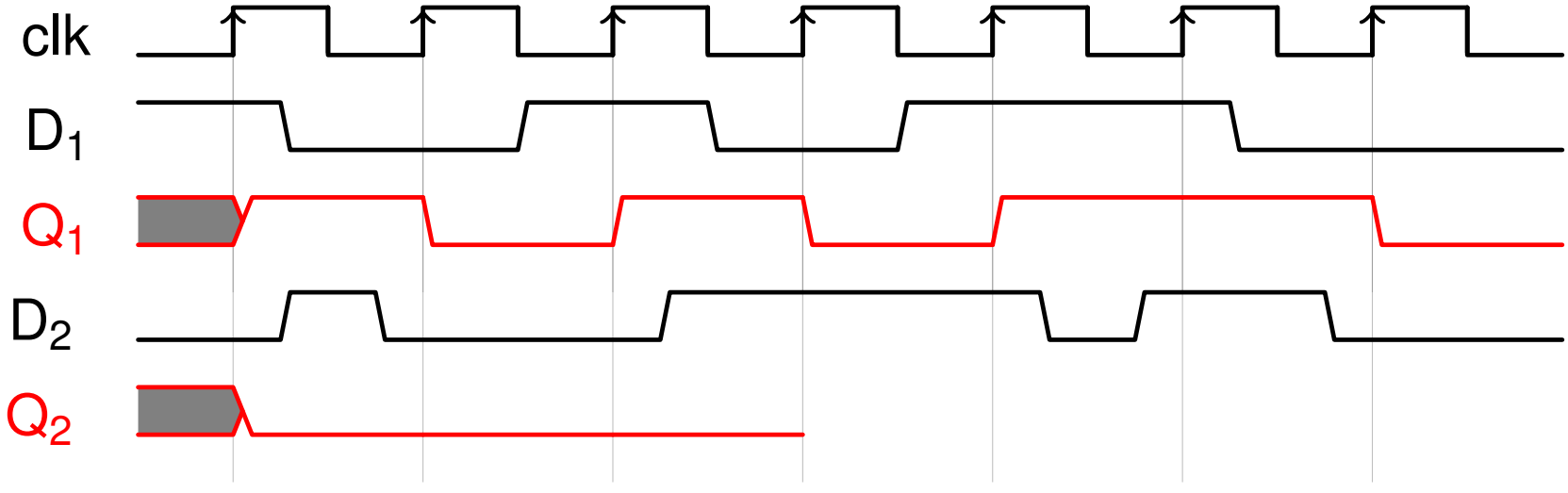
La bascule D

Exemple



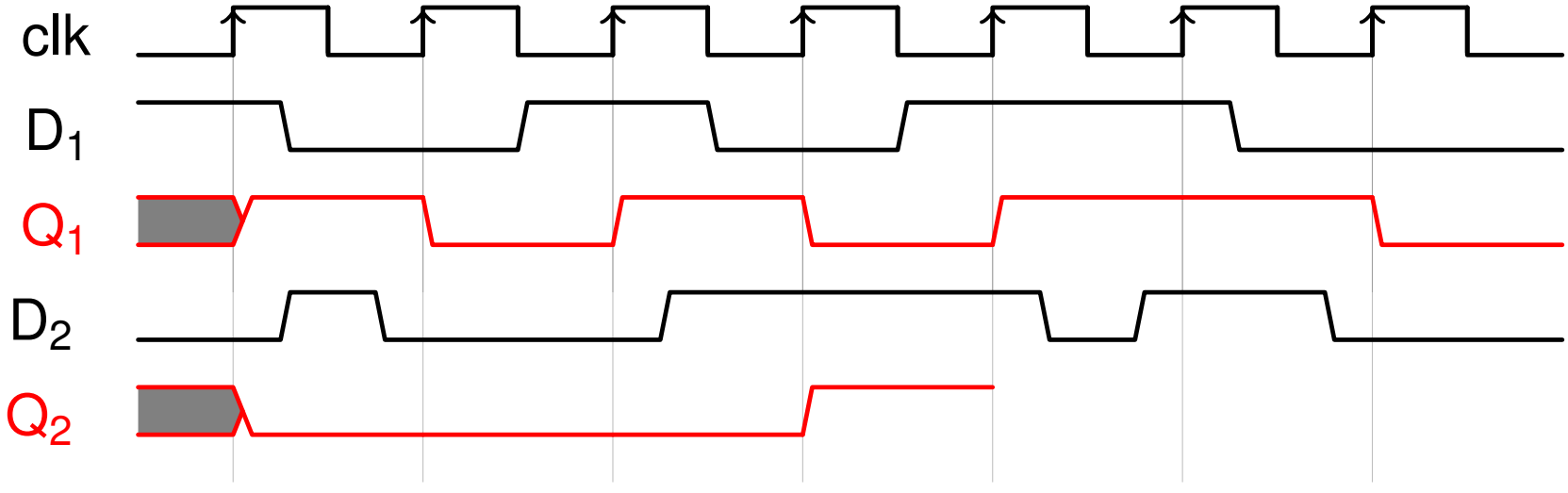
La bascule D

Exemple



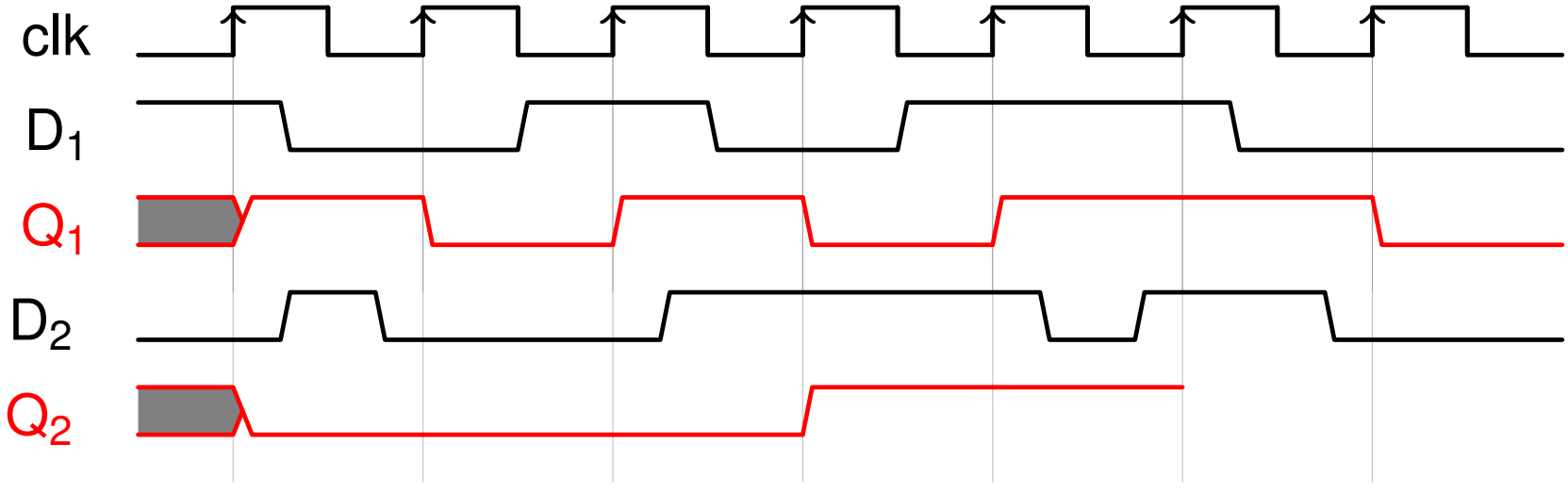
La bascule D

Exemple



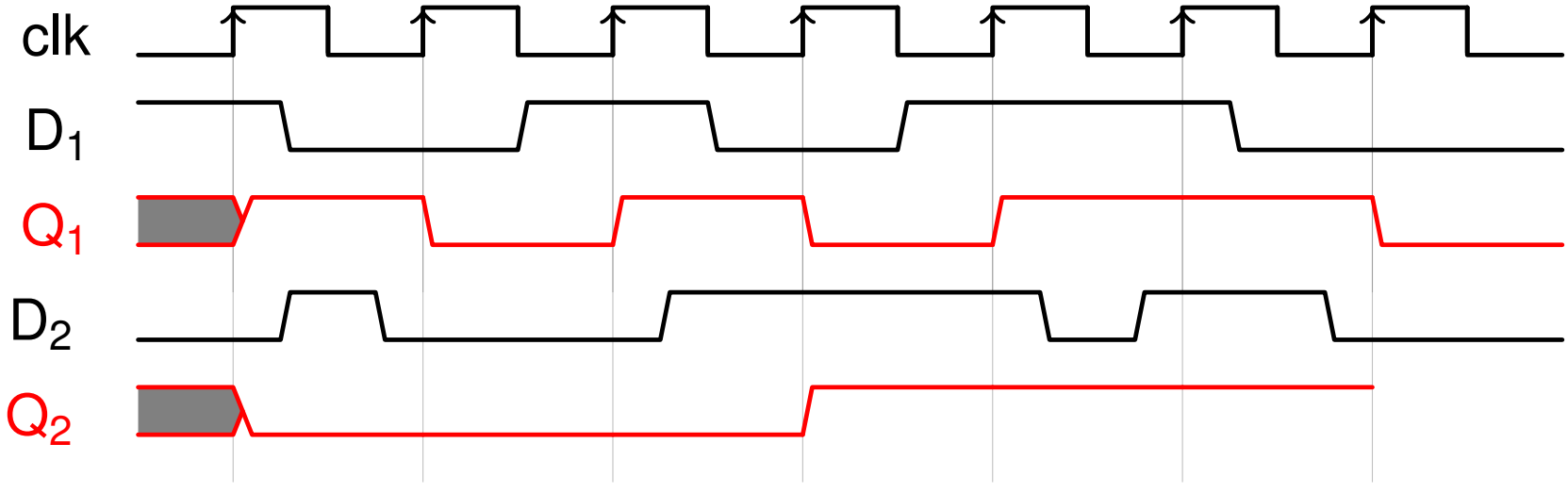
La bascule D

Exemple



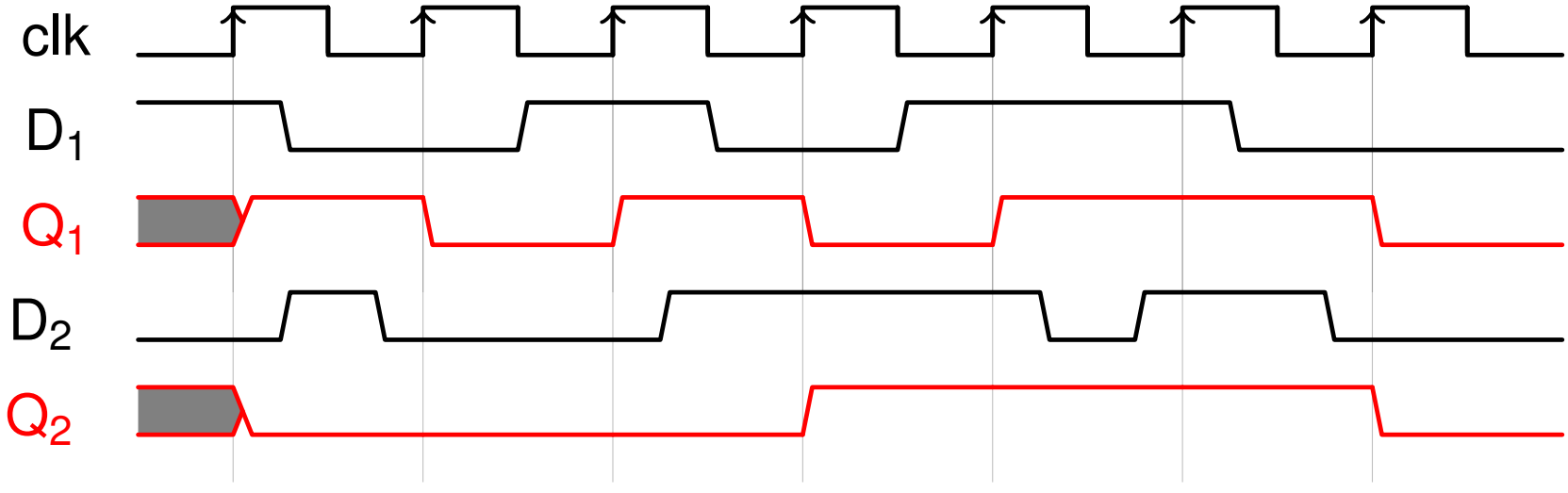
La bascule D

Exemple



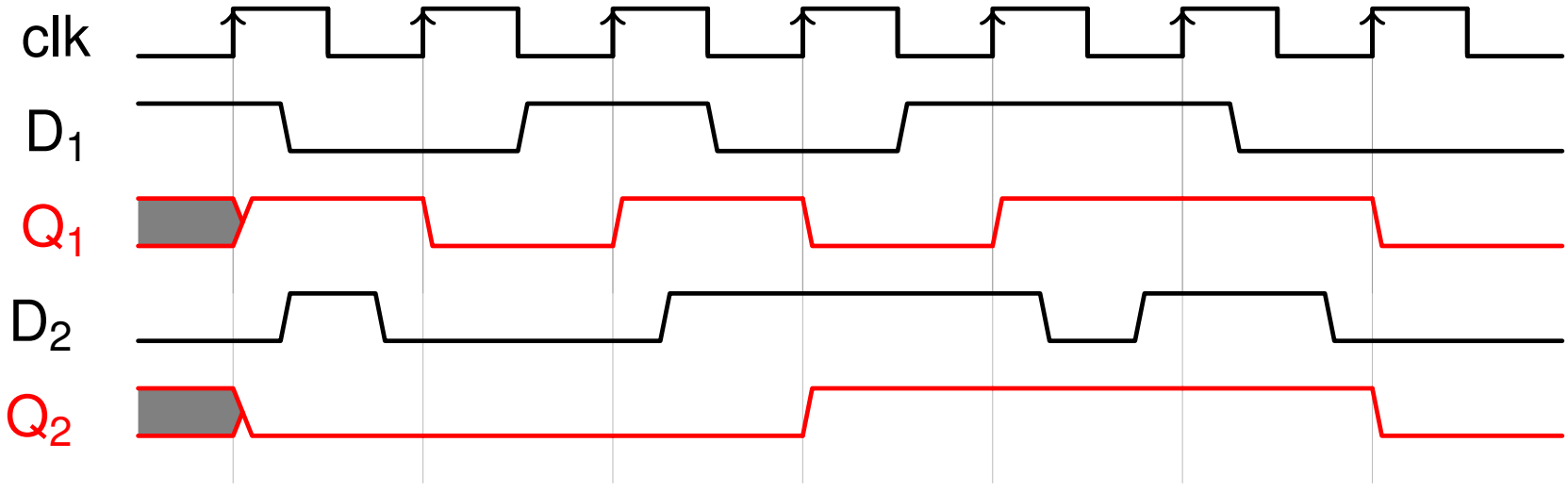
La bascule D

Exemple



La bascule D

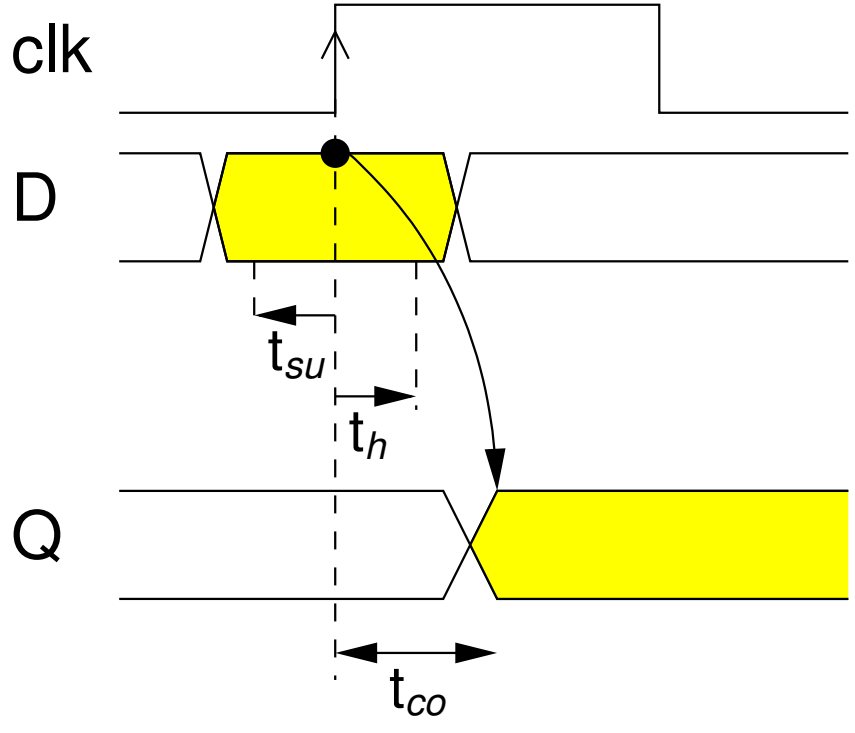
Exemple



- 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

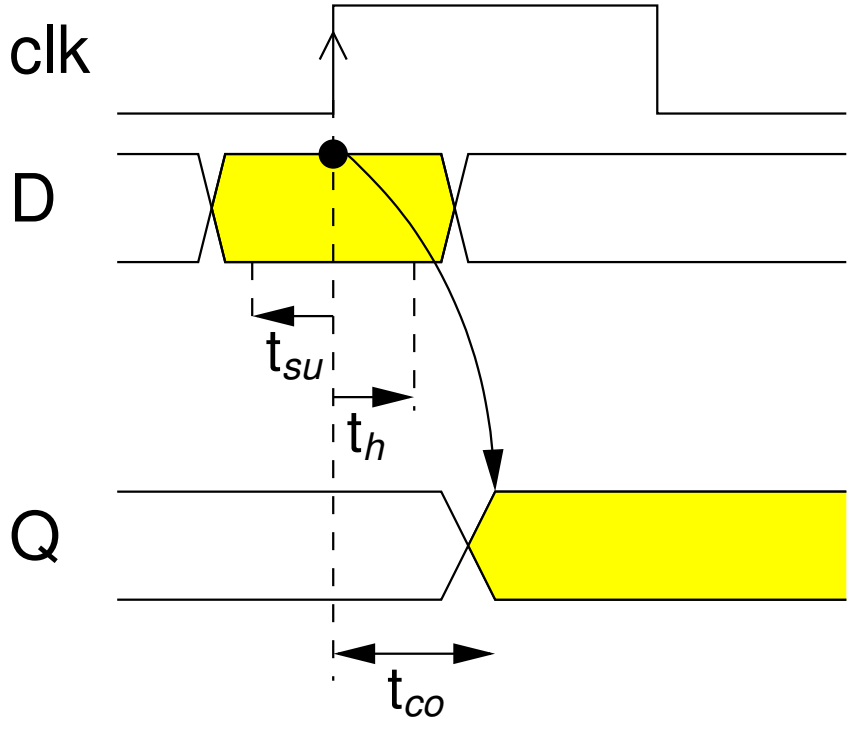
Contraintes temporelles



- 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



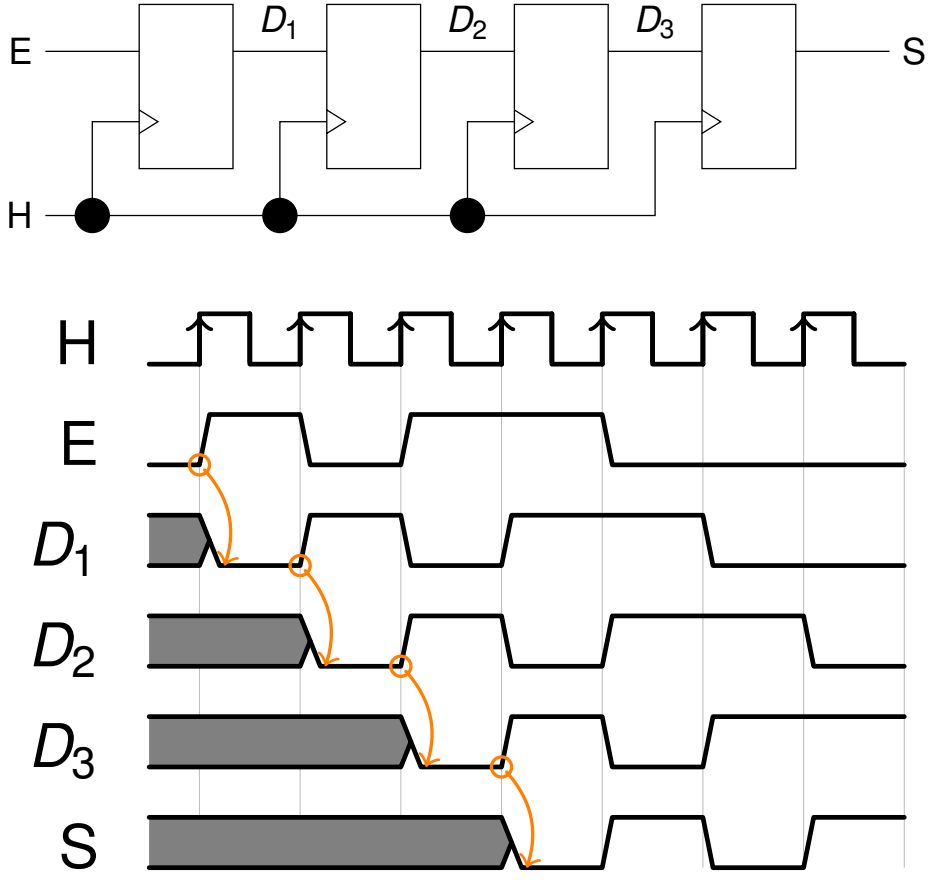
t_{su} : temps de pré–positionnement

t_h : temps de maintien

t_{co} : temps de propagation

Exemples d'applications

Registre à décalage



Plan

La logique séquentielle

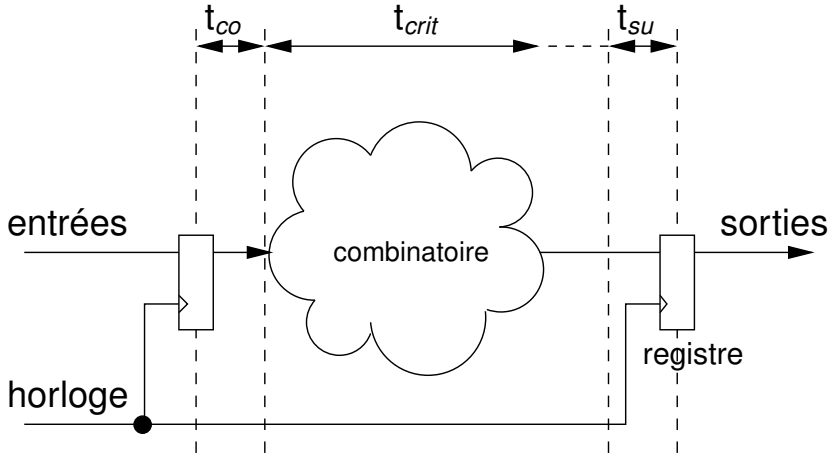
Règles

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 période de l’horloge dont être compatible avec le temps de propagation de la logique combinatoire

La logique séquentielle

Fréquence de fonctionnement

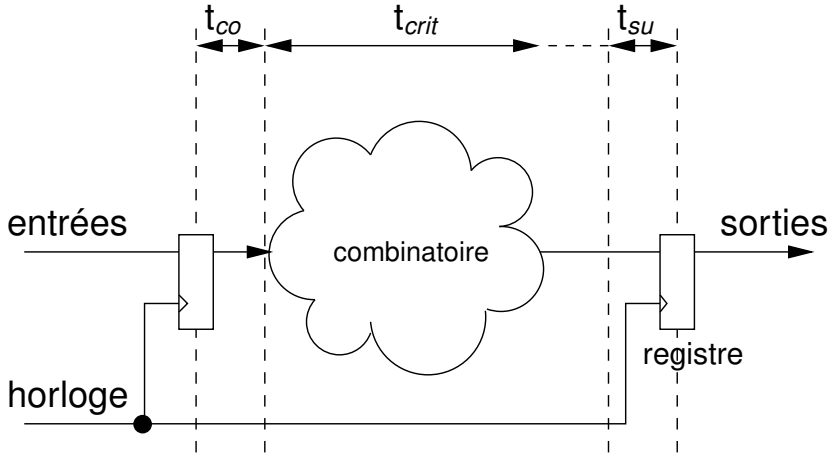


$$T_{clk} > t_{co} + t_{crit} + t_{su}$$

- 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

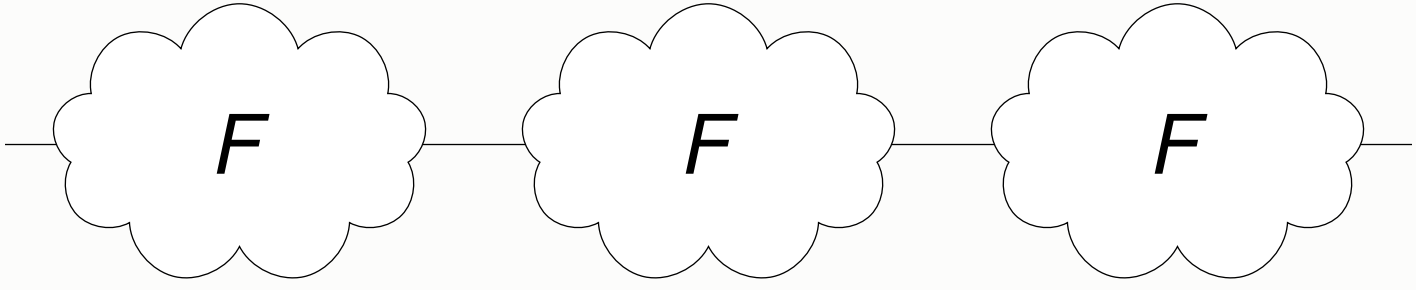


$$F_{max} = \frac{1}{t_{co} + t_{crit} + t_{su}}$$

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

Séquencer les traitements

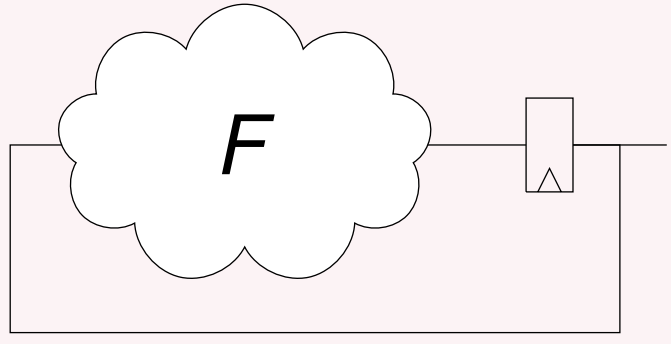
Répéter plusieurs fois le même calcul



- 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

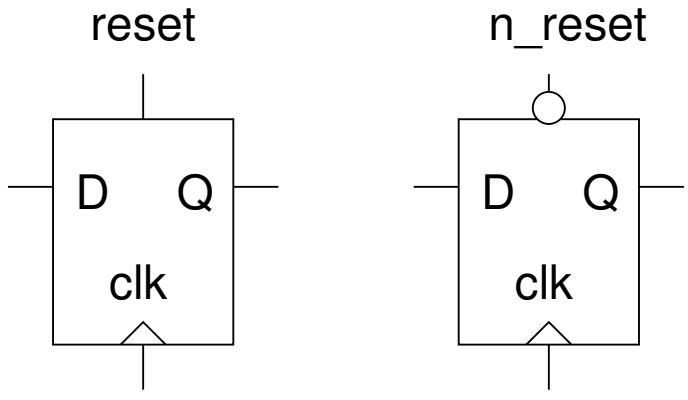
La valeur initiale ?

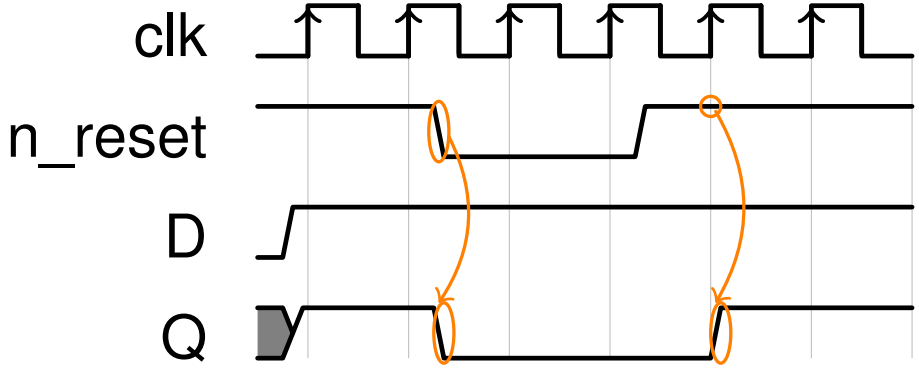
- Un signal extérieur est utilisé pour forcer l'état des bascules.
- On parle de "reset" (remise à zéro).
- Il peut s'agir d'une mise à "1", on parle alors de "preset".

Séquencer les traitements

Le reset asynchrone

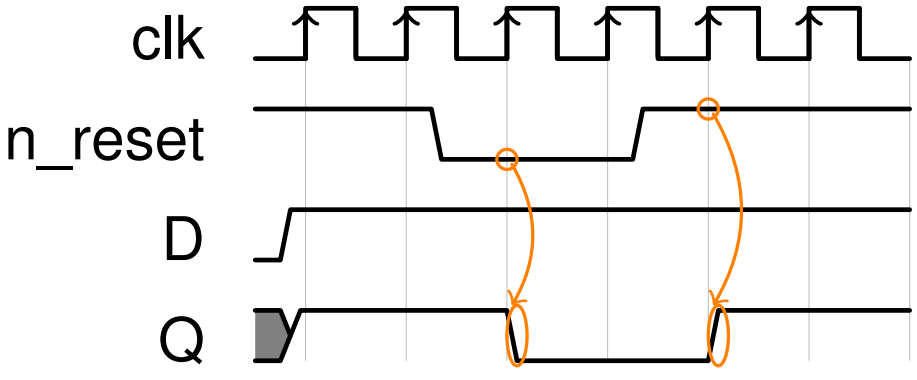
- Un signal connecté directement aux bascules.
- Son action est indépendante de l'horloge.
- En général global pour tout le circuit.
- Peut être positif ou négatif.





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

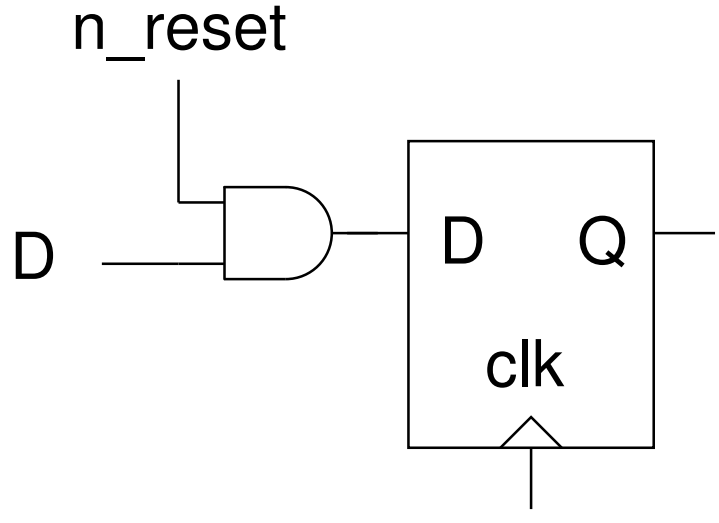
Le reset synchrone

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

Séquencer les traitements

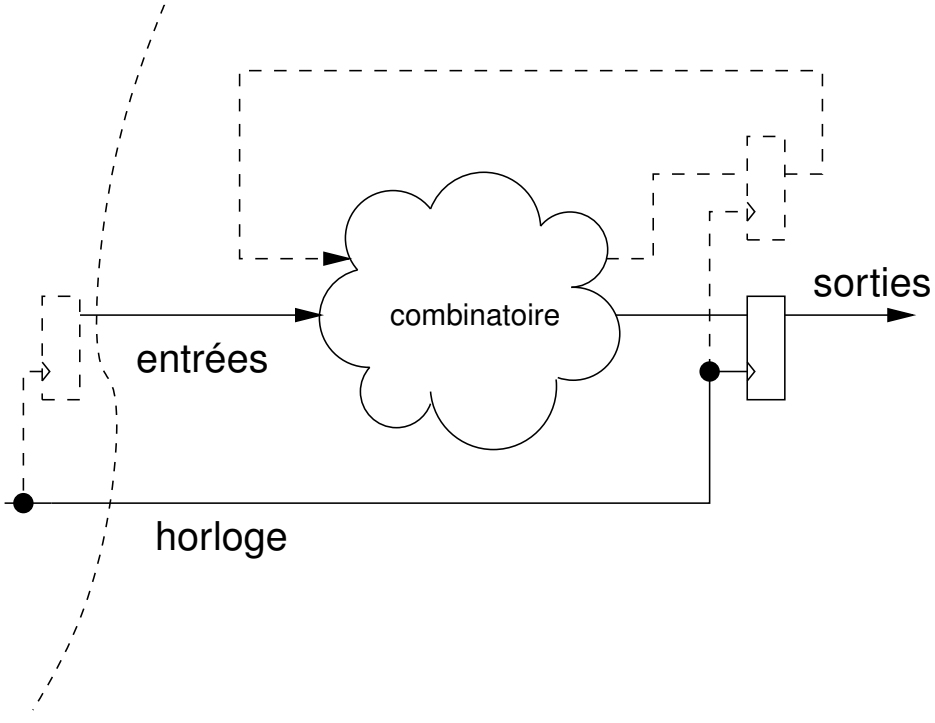
Le reset synchrone

- 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

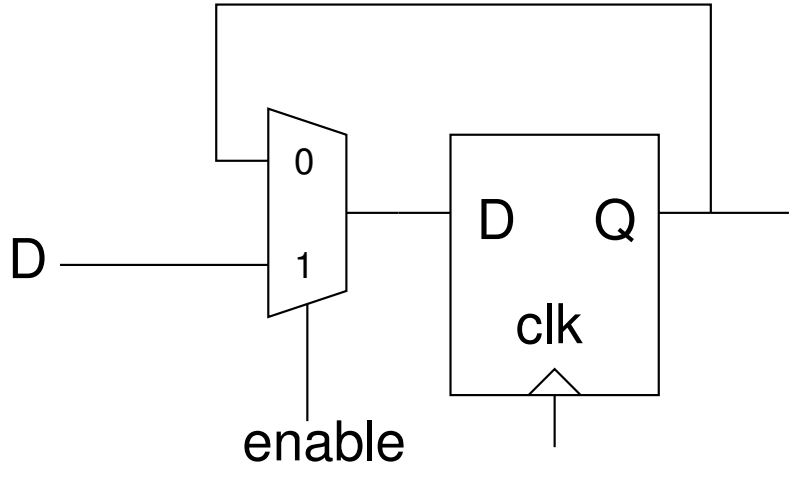
L'enable

- 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

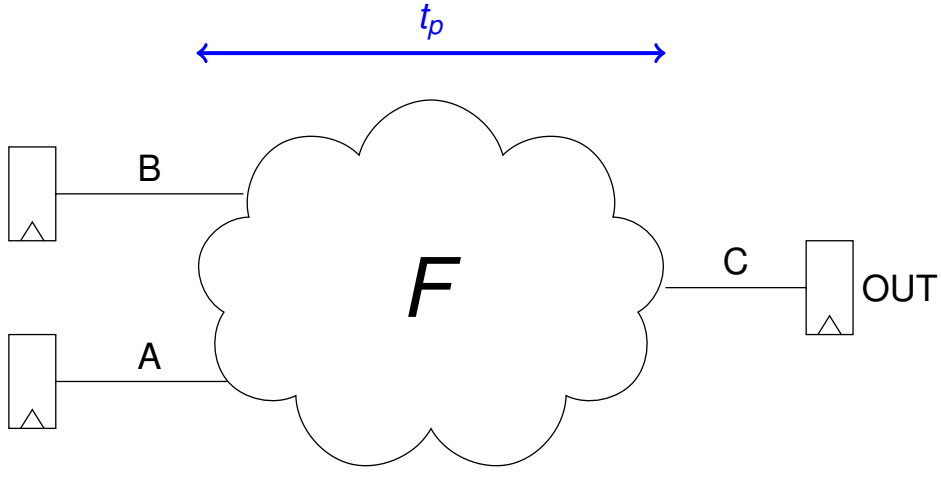
Un compteur

- 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

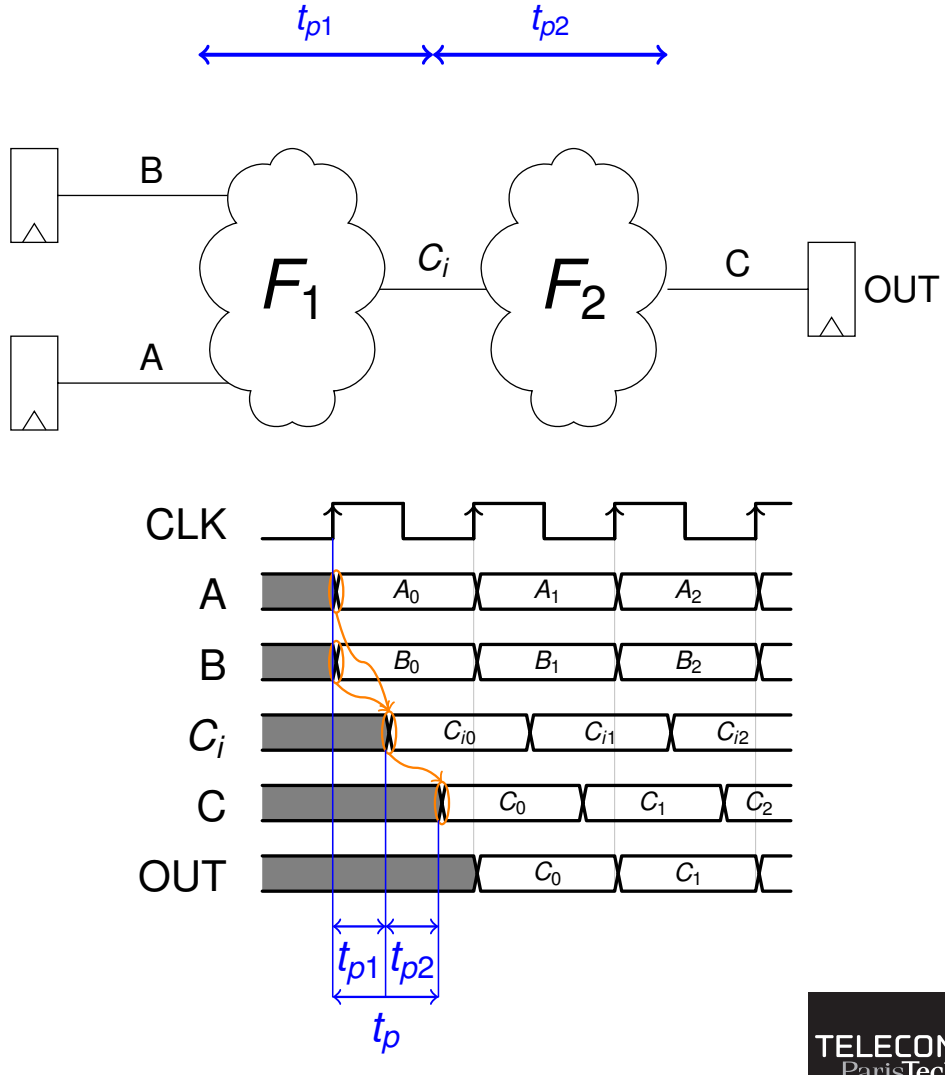
- Une fonction combinatoire F de temps de propagation t_p
- Les entrées sorties peuvent être synchrones tant que $t_p < T_{CLK}$
  - Où T_{CLK} est la période d'horloge



Exemples d'applications

Le Pipeline

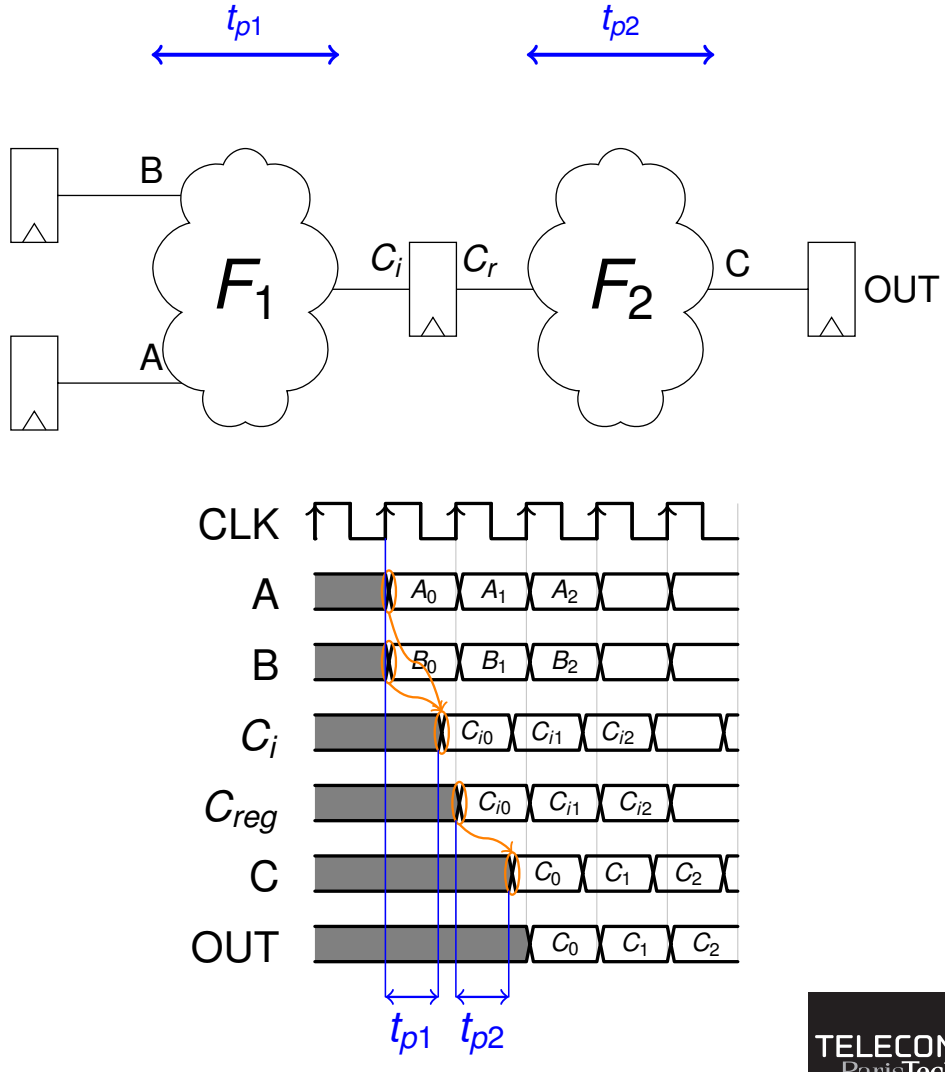
- On décompose F en deux fonctions combinatoires F_1 et F_2 de temps de propagation respectifs t_{p1} et t_{p2}
  - On s'arrange pour avoir $t_{p1} < t_p$ et $t_{p2} < t_p$
  - Dans cet exemple $t_{p1} + t_{p2} = t_p$
- Les entrées sorties peuvent être synchrones tant que $t_{p1} + t_{p2} < T_{CLK}$



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.

Plan

La bascule D

Toujours, à chaque front d'horloge, copier l'entrée sur la sortie.
Sinon, la sortie ne change pas.

La bascule D

Toujours, à chaque front d'horloge, copier l'entrée sur la sortie.
Sinon, la sortie ne change pas.(implicite)

En SystemVerilog

```
always @(posedge clk)
   q <= d;
```

La bascule D

La remise à zéro asynchrone

. . . sur niveau haut

```
always @(posedge clk or posedge reset)
   if(reset)
      q <= 1'b0;
   else
      q <= d;
```

La bascule D

La remise à zéro asynchrone

. . . sur niveau bas

```
always @(posedge clk or negedge nreset)
   if(!nreset)
      q <= 1'b0;
   else
      q <= d;
```

La bascule D

La remise à zéro synchrone

. . . sur niveau haut

```
always @(posedge clk)
   if(reset)
      q <= 1'b0;
   else
      q <= d;
```

La bascule D

La remise à zéro synchrone

. . . sur niveau bas

```
always @(posedge clk)
   if(!nreset)
      q <= 1'b0;
   else
      q <= d;
```

La bascule D

l'enable

sans reset

```
always @(posedge clk)
   if (en)
      q <= d;
```

La bascule D

l'enable

avec reset asynchrone

```
always @(posedge clk or negedge nreset)
   if(!nreset)
      q <= 1'b0;
   else if (en)
      q <= d;
```

La bascule D

l'enable

avec reset synchrone

```
always @(posedge clk)
   if(!nreset)
      q <= 1'b0;
   else if (en)
      q <= d;
```

Un registre

Un registre de 4 bits

```
logic[3:0] d;
...
logic[3:0] q;

always @(posedge clk)
   q <= d;
```

Un compteur

Un compteur modulo 16

```
logic[3:0] q;

always @(posedge clk)
   q <= q + 1;
```

Dont on ne connait pas l'état initial !

Un registre à décalage

Affectations simultanées

```
logic a, b, c, d;

always @(posedge clk)
begin
   b <= a;
   c <= b;
   d <= c;
end
```

```
logic a, b, c, d;

always @(posedge clk)
begin
   d <= c;
   c <= b;
   b <= a;
end
```

Dans un bloc, entre begin et end, les affectations sont faites simultanément.
L'ordre n'a donc pas d'importance et les deux codes sont équivalents.

- À **droite** d'une affectation <=, l'état **avant** le front.
- À **gauche** d'une affectation <=, l'état **après** le front.