Conception de circuits numériques et architecture des ordinateurs

Équipe pédagogique :
Marie Badaroux (TD), Julie Dumas (TD+TP), Florence Maraninchi (TD+TP),
Olivier Muller (TD+TP), **Frédéric Pétrot** (CM+TD), Pierre Ravenel (TD+TP)
Eduardo Tomasi Ribeiro (TP), Lionel Rieg (TD+TP),
Sébastien Viardot (TD+TP Apprentis)

Année universitaire 2024-2025

C1	Codage des nombres en base 2, logique booléenne, portes logiques, circuits combinatoires
C2	Circuits séquentiels
C3	Construction de circuits complexes
C4	Micro-architecture et fonctionnement des mémoires
C5	Machines à état
C6	Synthèse de circuits PC/PO
C7	Optimisation de circuits PC/PO
C8	Interprétation d'instructions - 1
C9	Interprétation d'instructions - 2
C10	Interprétation d'instructions - 3
C11	Introduction aux caches

Intérêt

Mémoriser l'information dans des variables est une nécessité des programmes
Les éléments séquentiels servent à cela.

Plan détaillé du cours d'aujourd'hui

Plan

Utilité des circuits séquentiels

Un circuit combinatoire implémente une fonction

- pour une valeur donnée de ses entrées, il produit *toujours* la même sortie, indépendamment de la séquence d'entrées lui ayant été préalablement appliqué
- n entrées, 2^n cas

Question :

Comment conserver une information binaire dans un circuit si ses entrées changent ?

Utilité des circuits séquentiels

Un élément séquentiel conserve *un état*, ainsi un circuit séquentiel n'est pas une fonction

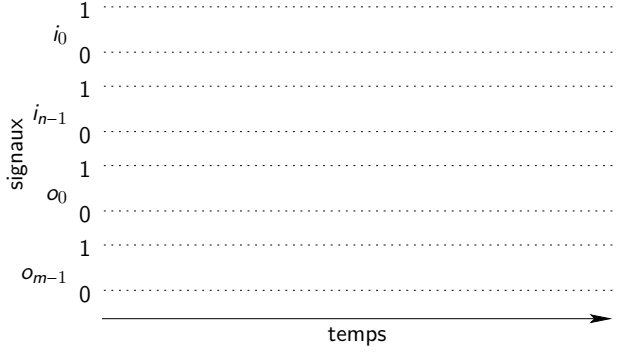
- pour une valeur donnée de ses entrées, la sortie dépend non seulement des entrées, mais aussi de la séquence d'entrées qui à été appliquée jusqu'à l'instant présent
- impossible à représenter sous forme de tables de vérité
- n entrées, m éléments séquentiels, 2^{n+m} cas

Éléments séquentiels contiennent les variables
La séquentialité est importante :
$a \Leftarrow 1$ seq $b \Leftarrow a$ différent de $b \Leftarrow a$ seq $a \Leftarrow 1$

Représentation des séquences

On utilise classiquement :

- des diagrammes de temps ou chronogrammes
  décrivent graphiquement une séquence particulière du fonctionnement

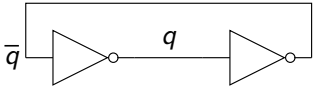


- des automates à états : cours n° 5
  décrivent formellement le fonctionnement « complet »

Plan

Bistable fonctionnellement inutile

Élement mémorisant le plus simple :
inverseurs couplés



Prend sa valeur à la mise sous tension et n'en change pas
Mémorisation ⇒ dépendance de l'entrée d'une porte sur sa sortie †

⚠ Que se passerait-il s'il y a avait 3 inverseurs ?

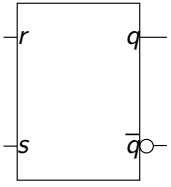
Tableau

†. Sauf dans le cas des mémoires dynamiques, mais c'est une autre histoire, c.f. cours n° 4

Verrou Reset/Set

Permet de forcer une valeur et de la conserver

- $r \leftarrow 1$ mise à 0 du verrou (*reset*)
- $s \leftarrow 1$ mise à 1 du verrou (*set*)
- r et s jamais à 1 simultanément

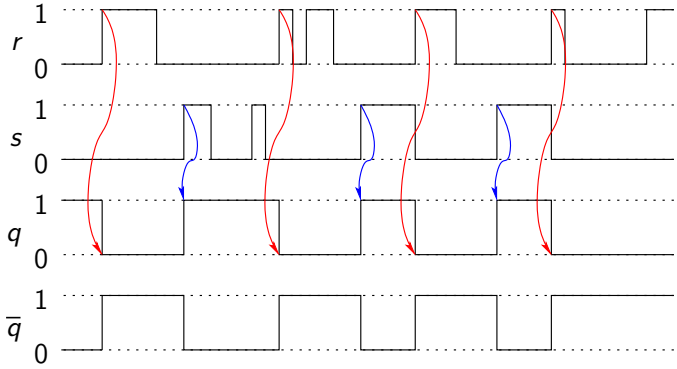


r	s	q	$\overline{q}$
0	0	q'	$\overline{q}'$
0	1	1	0
1	0	0	1

Changement de valeur de la sortie sur *l'état* des entrées

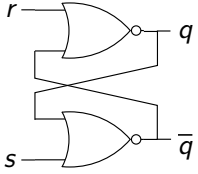
Verrou Reset/Set : comportement temporel

Chronogramme du comportement :



Verrou Reset/Set : implantation

Portes *nor* couplées



r	s	q	$\overline{q}$
0	0	q'	$\overline{q'}$
0	1	1	0
1	0	0	1
1	1	0	0

$\overline{q}$ porte bien mal son nom !

Stocker une information

RS :

- force à 0 ou à 1 une sortie
- utile pour contrôler une sortie
- mais repose sur la connaissance de r et s

Comment stocker une information ?

Utiliser un verrou de type D
Conserve une information dont la valeur (0 ou 1) est inconnue

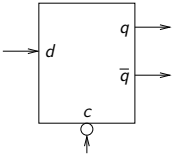
Verrou D (*D latch*)

Échantillonne entrée de donnée sur **niveau** de commande

- d entrée de donnée
- q sortie de donnée
- c entrée de commande (aussi appelée e pour *enable* ou g pour *guard*)

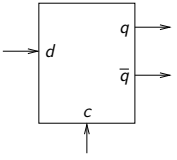
1. si active à l'état bas :

$$q = \begin{cases} d & \text{si } c = 0, \\ q' & \text{si } c = 1. \end{cases}$$



2. si active à l'état haut :

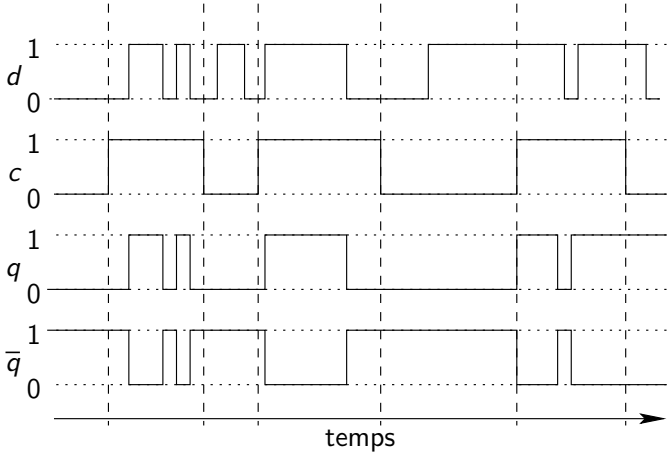
$$q = \begin{cases} d & \text{si } c = 1, \\ q' & \text{si } c = 0. \end{cases}$$



Changement de valeur de la sortie sur *l'état* de c

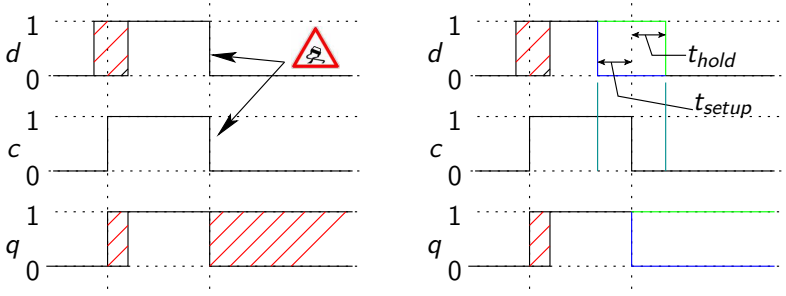
Verrou D : comportement temporel

Chronogramme du comportement pour un *latch* actif à 1 :



Verrou D : comportement temporel

Précisions très importantes :



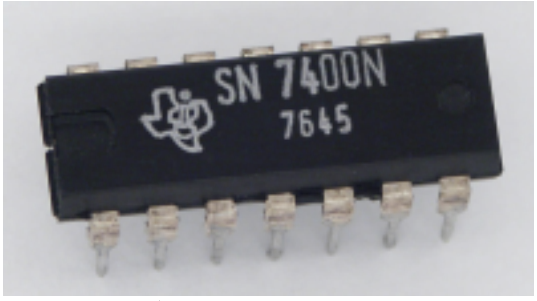
Conséquences pratiques

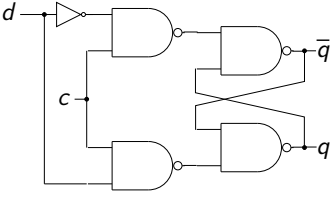
d doit être stable lorsque $c \leftarrow 0$

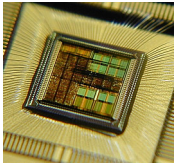
$c \leftarrow 0 \Rightarrow q$ prend la valeur que d possède *avant* le front

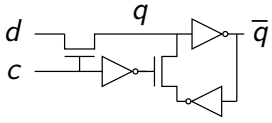
chronogrammes montrent c et q changent en même temps

Verrou D : implantations







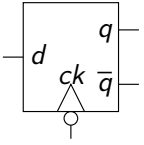


Bascule D (*D flip-flop*)

Échantillonne entrée de donnée sur **évènement** de commande

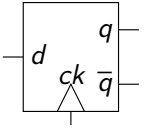
- d entrée de donnée
- q sortie de donnée
- ck entrée d'horloge (aussi appellée *clk*)
  1. si active sur front descendant :

$$q = \begin{cases} d & \text{si } ck \text{ subit } 1 \to 0, \\ q' & \text{sinon.} \end{cases}$$



  2. si active sur front montant :

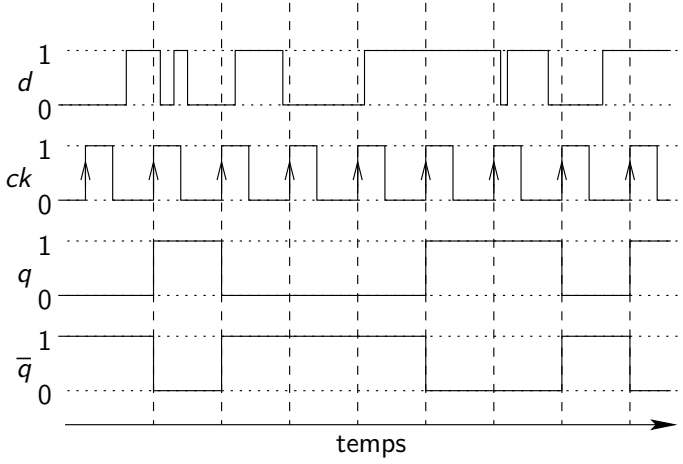
$$q = \begin{cases} d & \text{si } ck \text{ subit } 0 \to 1, \\ q' & \text{sinon.} \end{cases}$$



Changement de valeur de la sortie uniquement sur *un front* de ck

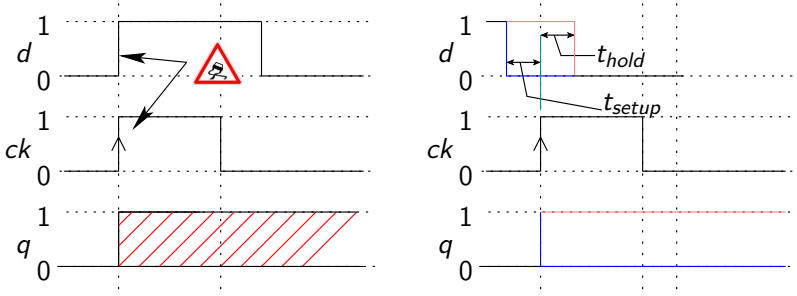
Bascule D : comportement temporel

Chronogramme du comportement pour une bascule active sur front montant :



Bascule D : comportement temporel

Précisions très importantes :



Conséquences pratiques

d doit être stable lorsque *ck* effectue ${}_0\nearrow^1$

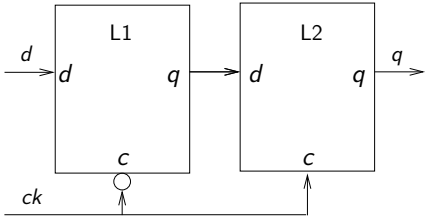
ck effectue ${}_0\nearrow^1 \Rightarrow$ *q* prend la valeur que *d* possède *avant* le front

chronogrammes montrent *ck* et *q* changent en même temps

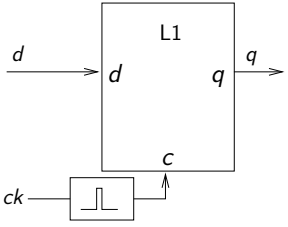
$t_{p(ck,q)} > t_{hold}$ pour échantillonnage correct

Bascule D : principes d'implantations

Master-Slave flip-flop

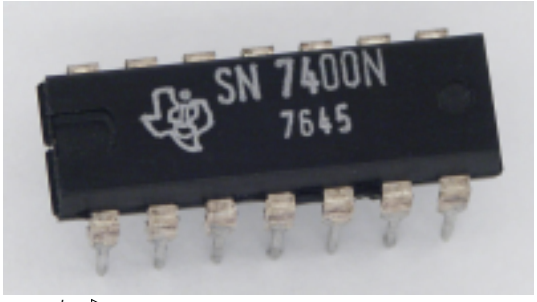


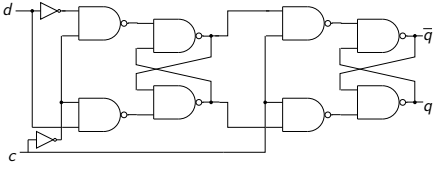
Pulse-Triggered flip-flop

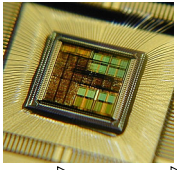


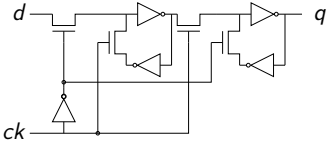
- Maître-Esclave : attention à c_{L1} et c_{L2}
- Impulsion : décalage de c_{L1} par rapport à $ck_{0\nearrow 1}$

Bascule D : implantations









Autres entrés

Verrous et bascules D peuvent avoir :

- une entréé de *reset* (ou *clear*)
- une entréé de *set* (ou *preset*)

Sur les bascules D, ces entrées peuvent avoir un effet :

- immédiat : asynchrones
- lié au front sensible suivant : synchrones

Et peuvent être actives à 0 ou à 1

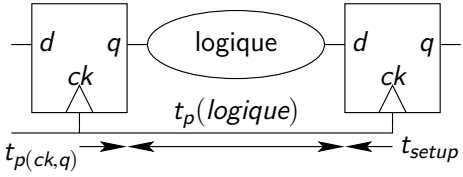
Les bascules D peuvent avoir aussi :

- une entrée d'autorisation d'écriture :
  WE pour *write enable* ou *E* pour *enable* ou *L* pour *load*
- cette entrée est toujours lié à l'horloge

Plan

Principes

Intègrent des éléments mémorisants et des portes logiques !



Période minimale de l'horloge (entre 2 ${}_0\nearrow^1$) :

$$t_{ck_{min}} = t_{p(ck,q)} + t_{logique_{max}} + t_{setup} + t_{skew}$$

avec
t_{skew} : décalage temporel entre les 2 entrées *ck*
et $t_{p(ck,q)} + t_{logique_{min}} - t_{skew} > t_{hold}$

Pour info : CPU à 4 GHz $\Rightarrow t_{ck_{min}} = 250ps$
Combien de cm à la vitesse de la lumière ?