ELEC-H-305

Circuits logiques et numériques

2011-2012

Cours 7

Dragomir Milojevic

dmilojev@ulb.ac.be

Introduction

Systèmes séquentiels

- Systèmes combinatoires ne peuvent être utilisés pour le contrôle de certains processus (c.f. exemple de la cuve)
- Calculer les sorties uniquement en fonction des entrées peut s'avérer limitatif
- Les systèmes séquentiels introduisent la notion de **l'état** (représentée par la **mémoire** du système)
- Représentation d'un système séquentiel:
  a) Graphe d'état, b) Table d'état, c) Equations logiques
- Spécification formelle des règles d'évolution de l'automatisme

Introduction

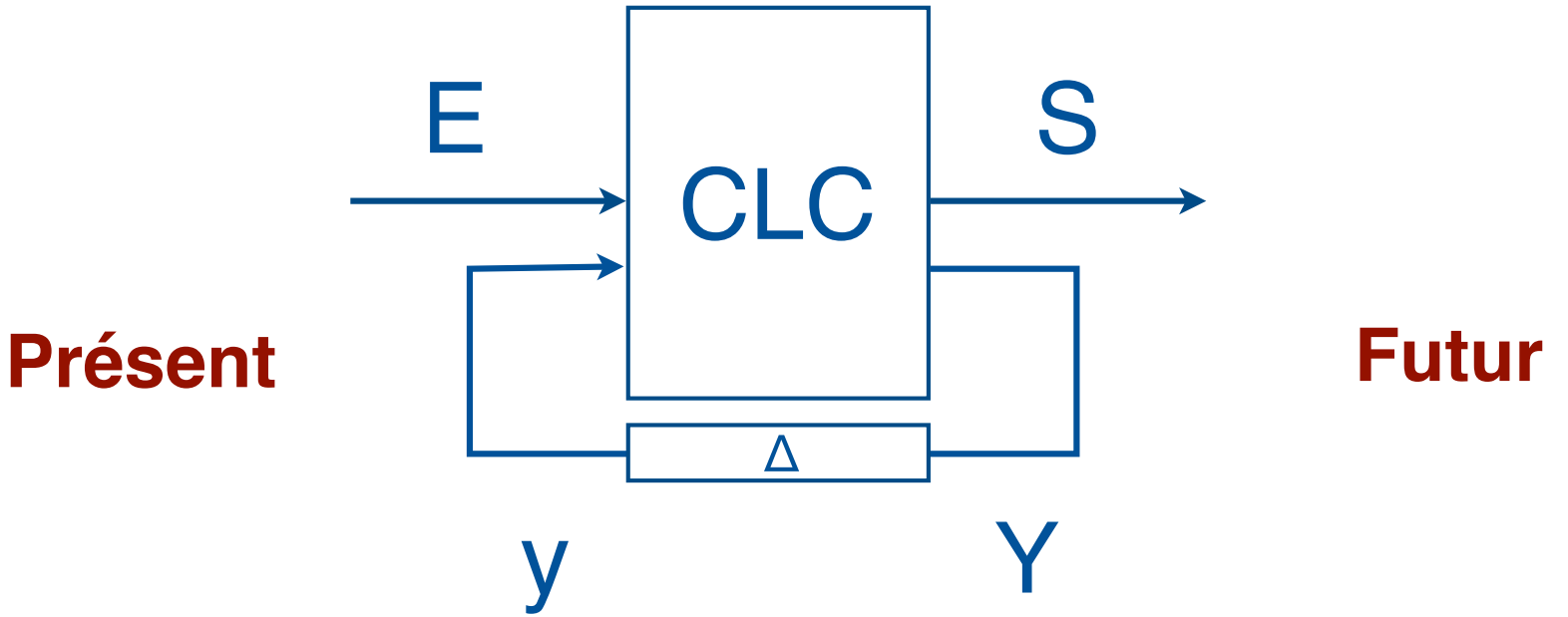
Systèmes séquentiels

- L'état du système (l'état interne) n'est pas forcement visible par le monde extérieur
- Ce qui nous intéresse c'est la sortie !
- La sortie d'un système est calculé **en fonction de l'état** dans lequel le système se trouve **et éventuellement des entrées**
- **Eventuellement** implique deux classes des circuit logiques séquentiels : la **Machine de Moore** et la **Machine de Meally**.
- Dans un premier temps il faut percevoir la différence entre les deux, l'emploi de l'une ou de l'autre sera explicité plus tard

Introduction

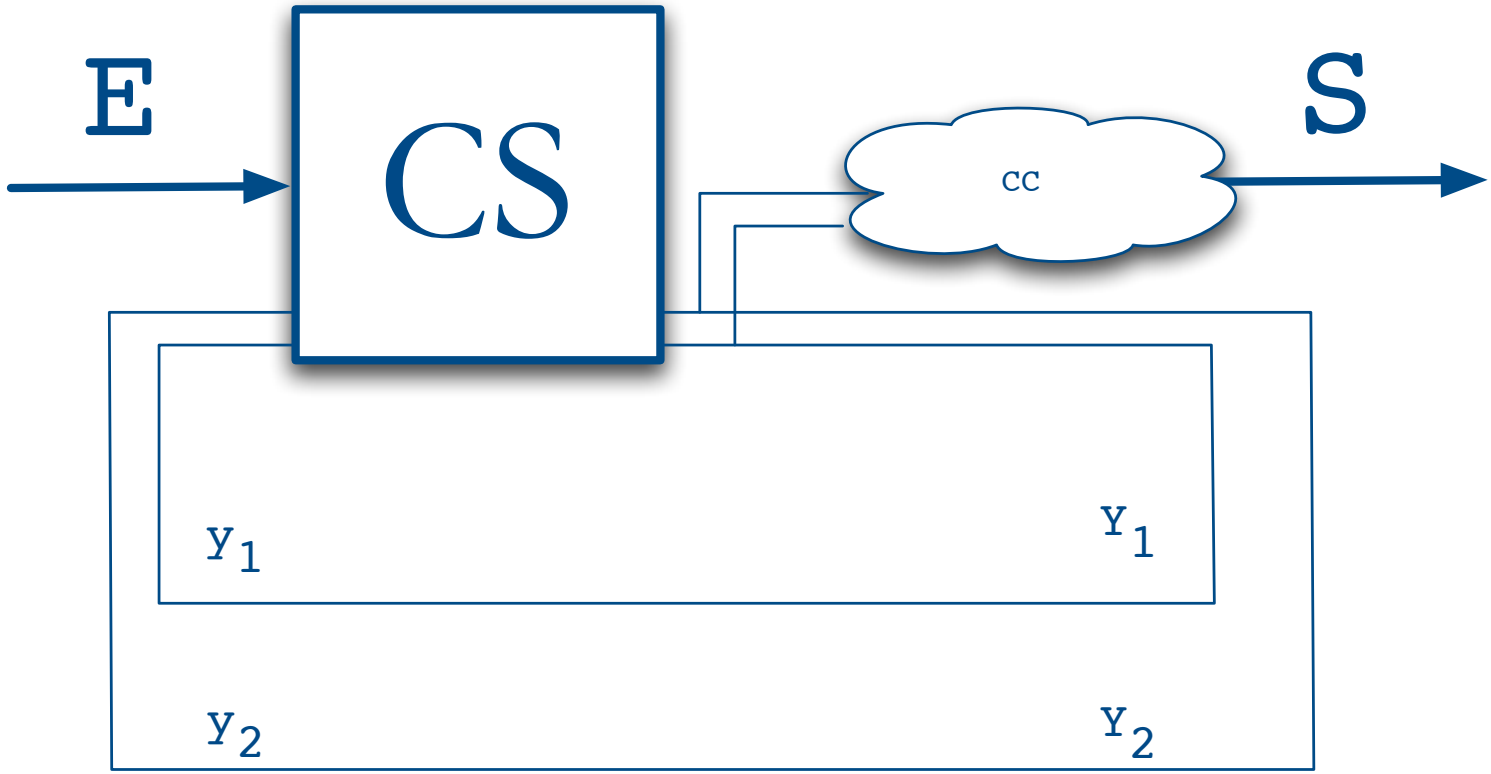
Systèmes séquentiels — schématiquement

Notion d'état — la **rétroaction** — distinction entre le **passé** et le **présent** (présent/futur) → obtenue à l'aide d'un délais (délais dans un fil ou une **mémoire)**. Pour le moment ce délais est modélisé par le délais dans un fil → il existe un temps nécessaire au signal de parcourir la rétroaction.



Machine de Moore

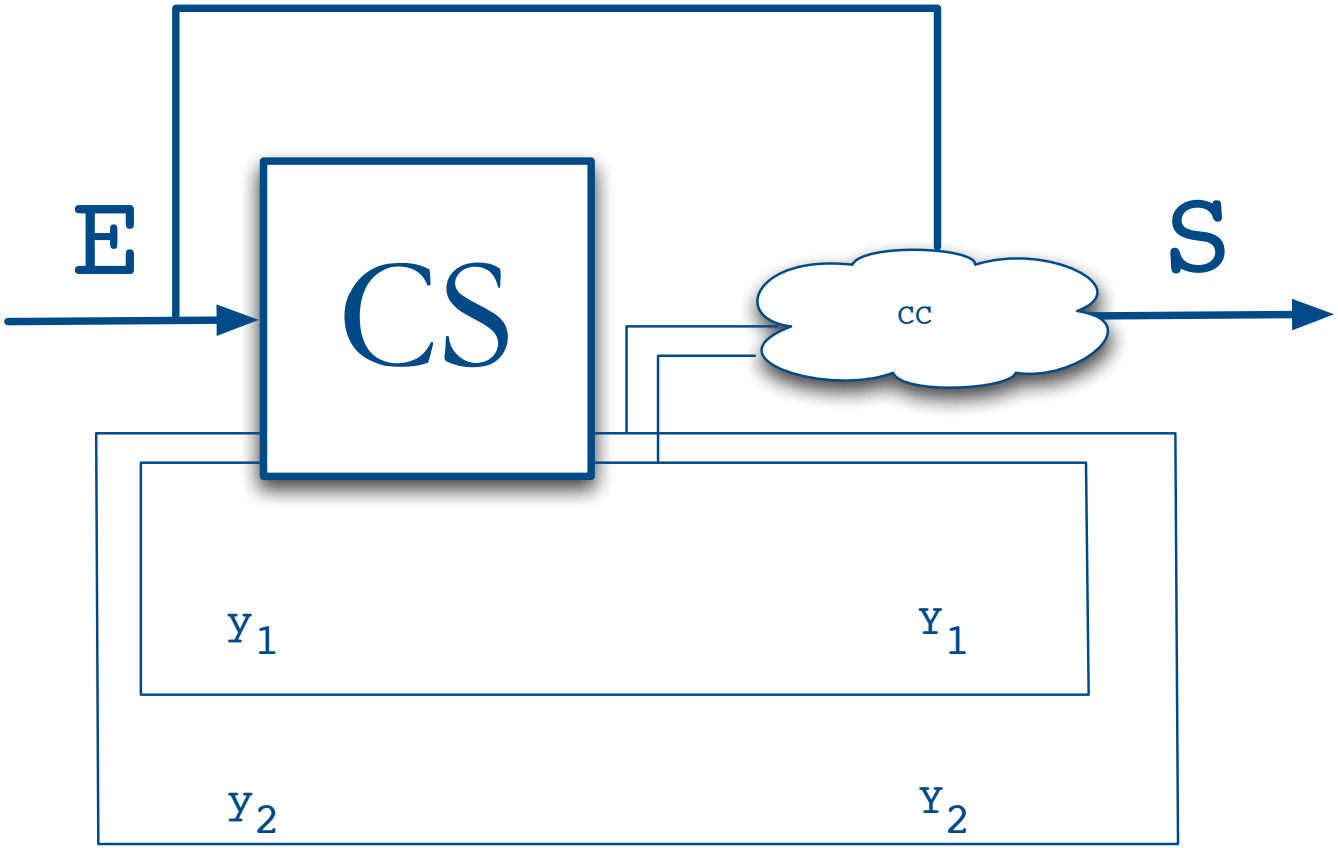
Sortie est uniquement fonction (combinatoire) des **variables d'état**. Pour le moment nous allons travailler uniquement avec ce type de machines.



Introduction

Machine de Meally

Sortie est fonction (combinatoire) des **variables d'état et des entrées**. On verra leur utilité plus tard.



Introduction

Représentation

- Dans l'ordre, pour une **synthèse** à partir de cahier de charges verbal: a) Table d'état, b) graphe d'état (optionnel), c) **équations logiques** et le circuit logique (obligatoire).
- → Notion de la **Synthèse de systèmes séquentiels** (3ème et dernière grosse partie du cours). Votre travail :
  - Dériver la table d'état,
  - Les équations logiques correspondantes et
  - Le logigramme (éventuellement)
- Plan pour aujourd'hui : représentation avec les équations logiques, exemple concret, simplification de la table d'état (réduction de nombre d'états) et la réalisation d'organes de mémoire.

Systèmes séquentiels

Représentation avec les équations logiques d'état

- On utilise les nombres décimaux pour représenter les états (mais cela peut être n'importe quoi d'autre: lettres, symboles, ...).
- Circuits logiques: on souhaite représenter les systèmes séquentiels à l'aide des codes binaires {0,1,–}.
- On appelle le processus d'attribution de codes binaires aux états codés **le codage des états**.

Le nombre de bits nécessaires (**variables d'état**) pour coder *n* états:

$$\log_2 n$$

Systèmes séquentiels

Représentation avec les équations logiques d'état

- Dans l'exemple considéré: le système à 4 états
- A chaque état on attribue un code binaire
- Pour le moment nous allons faire l'attribution des codes de façon arbitraire:

1 → 00 ; **2** → 01 ; **3** → 11 ; **4** → 10

Table d'état

	ab 00	01	11	10	z
1	**1**	2	**1**	–	0
2	1	**2**	3	–	0
3	–	2	**3**	4	1
4	1	–	–	**4**	1

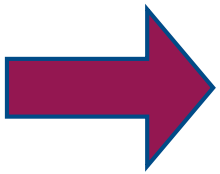


Table d'état codé

	ab 00	01	11	10	z
00	**00**	01	**00**	–	0
01	00	**01**	11	–	0
11	–	01	**11**	10	1
10	00	–	–	**10**	1

Systèmes séquentiels

Représentation avec les équations logiques d'état

	ab				
Y_2Y_1	00	01	11	10	Z
00	**00**	01	**00**	–	0
01	00	**01**	11	–	0
11	–	01	**11**	10	1
10	00	–	–	**10**	1

y_2y_1

- Les états sont codés : à chaque bit du **code** correspond **une fonction logique**
- Pour l'exemple donné : 4 états, deux variables d'état (y_2y_1) → deux fonctions logiques.
- Noter la différence entre y_2y_1 et Y_2Y_1: nécessaire pour faire la distinction entre les valeur des variables d'états se référant au **présent** (y_2y_1) et les valeurs de variables d'état se référant au **futur** (Y_2Y_1).

Systèmes séquentiels

Représentation avec les équations logiques d'état K-Maps

ab

Y_2Y_1	00	01	11	10	Z
00	**00**	01	**00**	–	0
01	00	**01**	11	–	0
11	–	01	**11**	10	1
10	00	–	–	**10**	1

Y_2Y_1

Y_2	00	01	11	10
00	**0**	0	**0**	–
01	0	**0**	1	–
11	–	0	**1**	1
10	0	–	–	**1**

Y_1	00	01	11	10
00	**0**	1	**0**	–
01	0	**1**	1	–
11	–	1	**1**	0
10	0	–	–	**0**

Systèmes séquentiels

Représentation avec les équations logiques d'état

Fonctions Logiques

Y_2 \ ab	00	01	11	10
00	**0**	0	**0**	–
01	0	**0**	1	–
11	–	0	**1**	1
10	0	–	–	**1**

y_2y_1

$$Y_2 = ab' + y_1a$$

Y_1 \ ab	00	01	11	10
00	**0**	1	**0**	–
01	0	**1**	1	–
11	–	1	**1**	0
10	0	–	–	**0**

y_2y_1

$$Y_1 = a'b + y_1b$$

Systèmes séquentiels

Représentation avec les équations logiques d'état

Fonction de sortie → Nous disposons de l'information sur la valeur de la sortie pour des états stables (à partir de la table d'états). On commence par remplir les cases correspondantes aux états stables. Pour les transitions (marquées en gris dans la K-Map de sortie) : on les fait après (voir le transparent suivant).

	ab 00	01	11	10	z
00	**00**	01	**00**	–	0
01	00	**01**	11	–	0
11	–	01	**11**	10	1
10	00	–	–	**10**	1

z \ ab	00	01	11	10
00	**0**		**0**	–
01		**0**		–
11	–		**1**	
10		–	–	**1**

y_2y_1

Systèmes séquentiels

Représentation avec les équations logiques d'état

Fonction de sortie

Pour les transitions nous allons adopter la règle suivante :
Lors des transitions, si la sortie doit changer, elle ne devrait changer qu'une seule fois.

Toutes les possibilités :

(pour une seule transition entre 2 états stables)

Etat de départ	0	1	0	1
Transition	0	1	–	–
Etat d'arrivé	0	1	1	0

Mêmes:
Il faut la maintenir lors de la transition

Différentes:
On peut mettre un *don't care*

Systèmes séquentiels

Représentation avec les équations logiques d'état

Fonction de sortie pour l'exemple : transitions permettent la simplification de la fonction logique de sortie. En gris les sorties pour les transitions, déterminées en appliquant la règle du transparent précédent.

	ab 00	01	11	10	Z
00	**00**	01	**00**	–	0
01	00	**01**	11	–	0
11	–	01	**11**	10	1
10	00	–	–	**10**	1

Z	ab 00	01	11	10
00	**0**	0	**0**	–
01	0	**0**	–	–
11	–	–	**1**	1
10	–	–	–	**1**

y_2y_1

$Z=y_2$

Systèmes séquentiels

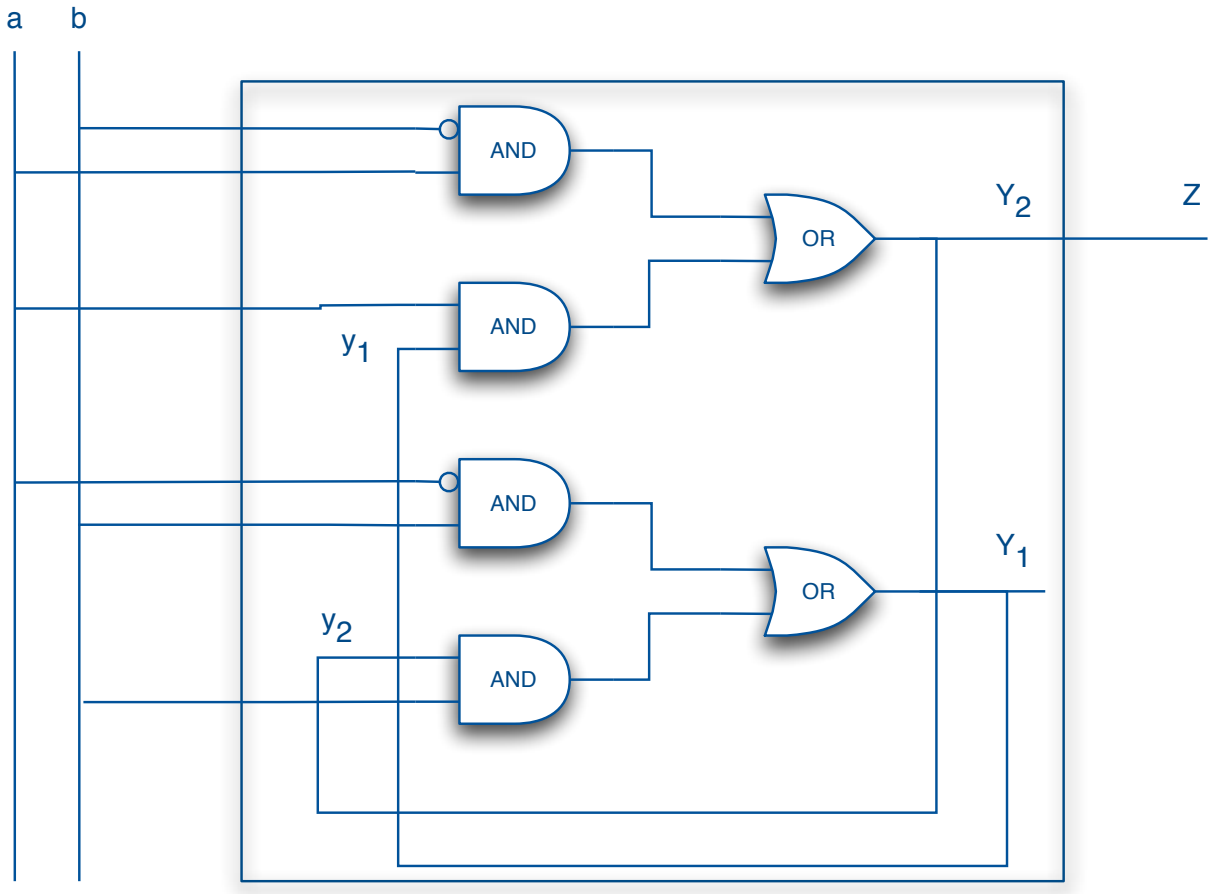
Logigramme de système séquentiel

Attention aux fonctions de rétroaction : faire la distinction entre les états présents (y_i) et les états futurs (Y_i).

$Y_2 = ab' + y_1 a$

$Y_1 = a'b + y_2 b$

$Z = y_2$



Systèmes séquentiels

Exemple complet

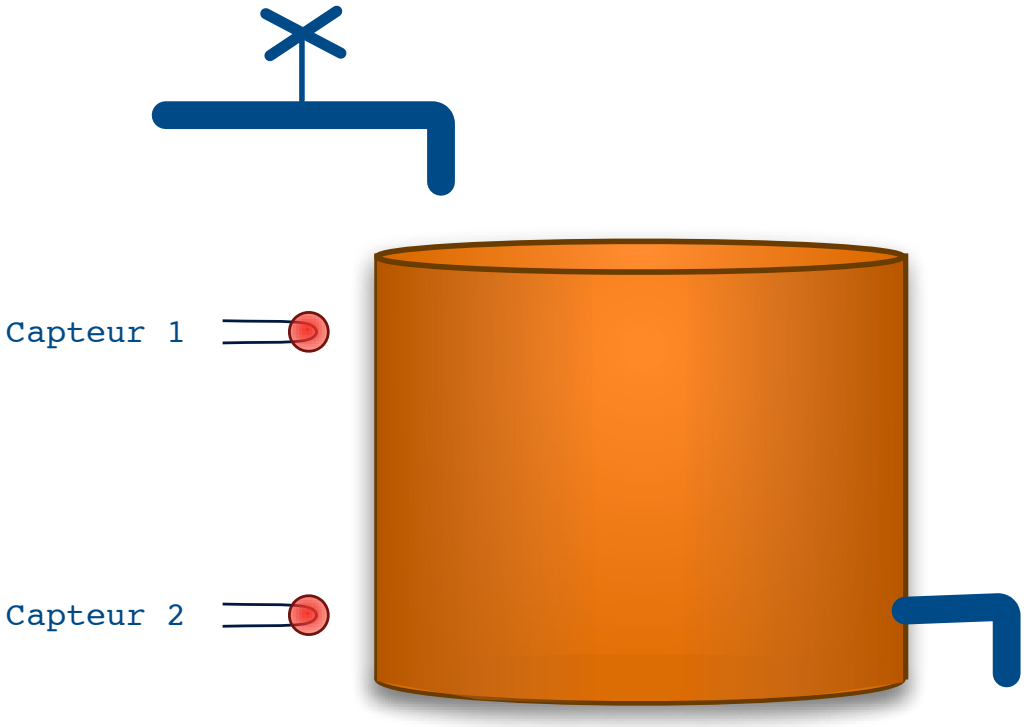
Pour l'exemple de la cuve:

1. Etablir la table d'état
2. Dessiner le graphe d'état
3. Coder les états
4. Déduire les K-Maps correspondants aux fonctions de rétroaction
5. Déterminer la valeur de la sortie pour les transitions
6. Dériver les expressions logiques
7. Dessiner le logigramme de système complet

Systèmes séquentiels

Exemple complet

1. La table d'état (on suppose l'état initial = cuve pleine)



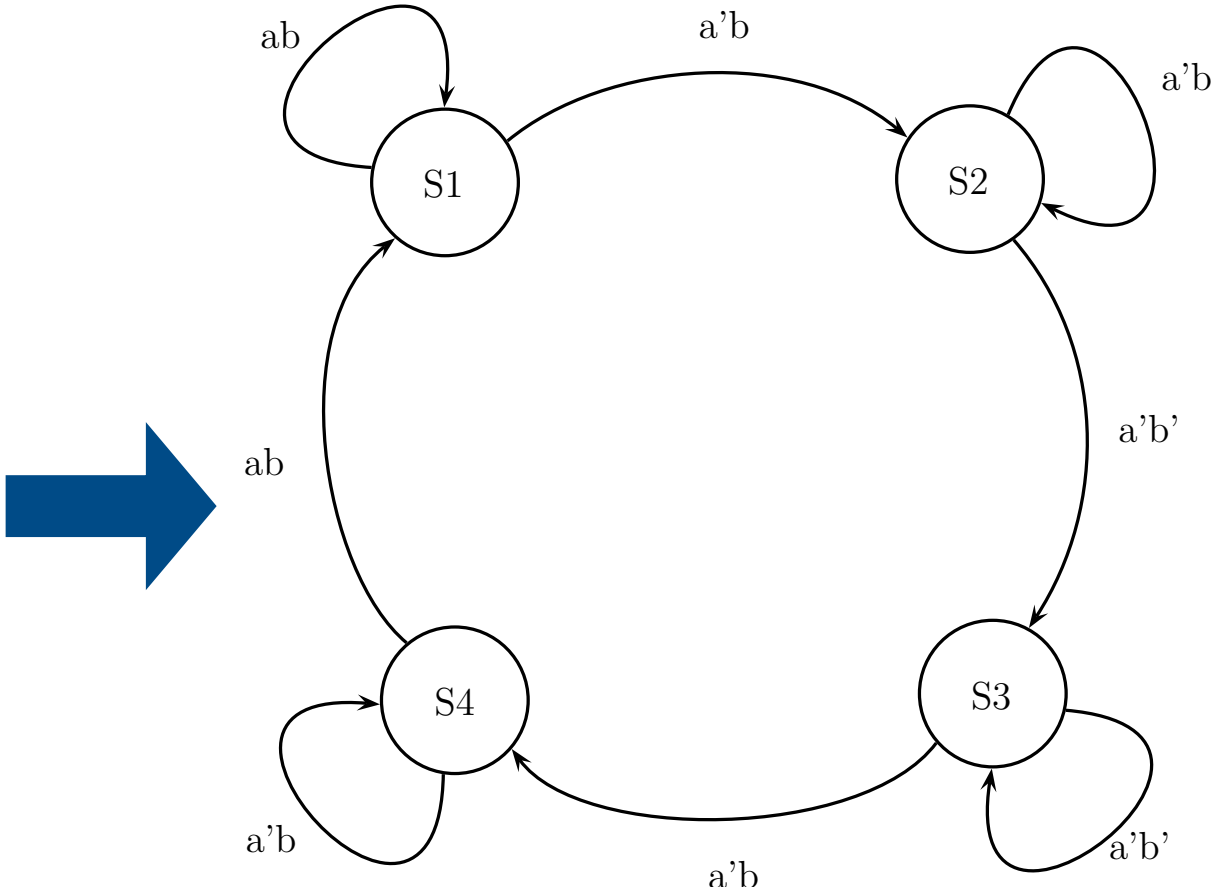
	ab 00	01	11	10	z
1	–	2	**1**	–	0
2	3	**2**	1	–	0
3	**3**	4	–	–	1
4	3	**4**	1	–	1

Systèmes séquentiels

Exemple complet

2. Graphe d'état

	ab 00	01	11	10	z
1	-	2	**1**	-	0
2	3	**2**	1	-	0
3	**3**	4	-	-	1
4	3	**4**	1	-	1

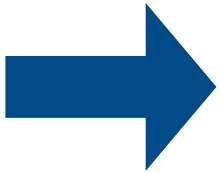


Systèmes séquentiels

Exemple complet

3. Codage d'états (**1**→00;**2**→01;**3**→11;**4**→10)

	ab 00	01	11	10	z
1	–	2	**1**	–	0
2	3	**2**	1	–	0
3	**3**	4	–	–	1
4	3	**4**	1	–	1



	ab 00	01	11	10	z
00	–	01	**00**	–	0
01	11	**01**	00	–	0
11	**11**	10	–	–	1
10	11	**10**	00	–	1

Systèmes séquentiels

Exemple complet

4. Fonctions de rétroaction

	ab 00	01	11	10	Z
00	–	01	**00**	–	0
01	11	**01**	00	–	0
11	**11**	10	–	–	1
10	11	**10**	0	–	1

Y_2 \ ab	00	01	11	10
00	–	0	**0**	–
01	1	**0**	0	–
11	**1**	1	–	–
10	1	**1**	0	–

$Y_2 = a'b' + y_2 a'$

Y_1 \ ab	00	01	11	10
00	–	1	**0**	–
01	1	**1**	0	–
11	**1**	0	–	–
10	1	**0**	0	–

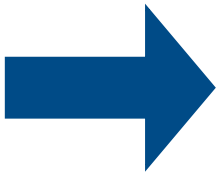
$Y_1 = a'b' + y_2' a'$

Systèmes séquentiels

Exemple complet

5. Valeurs de sortie

	ab				
	00	01	11	10	Z
1	–	2	**1**	–	0
2	3	**2**	1	–	0
3	**3**	4	–	–	1
4	3	**4**	1	–	1



Z	ab			
	00	01	11	10
0	–	0	**0**	–
1	–	**0**	0	–
11	**1**	1	–	–
10	1	**1**	–	–

Systèmes séquentiels

Exemple complet

6. Fonctions logiques (fonctions de rétroaction + sortie)

Y_2 \ ab	00	01	11	10
00	–	0	**0**	–
01	1	**0**	0	–
11	**1**	1	–	–
10	1	**1**	0	–

$Y_2 = a'b' + y_2a'$

Y_1 \ ab	00	01	11	10
00	–	1	**0**	–
01	1	**1**	0	–
11	**1**	0	–	–
10	1	**0**	0	–

$Y_1 = a'b' + y_2'a'$

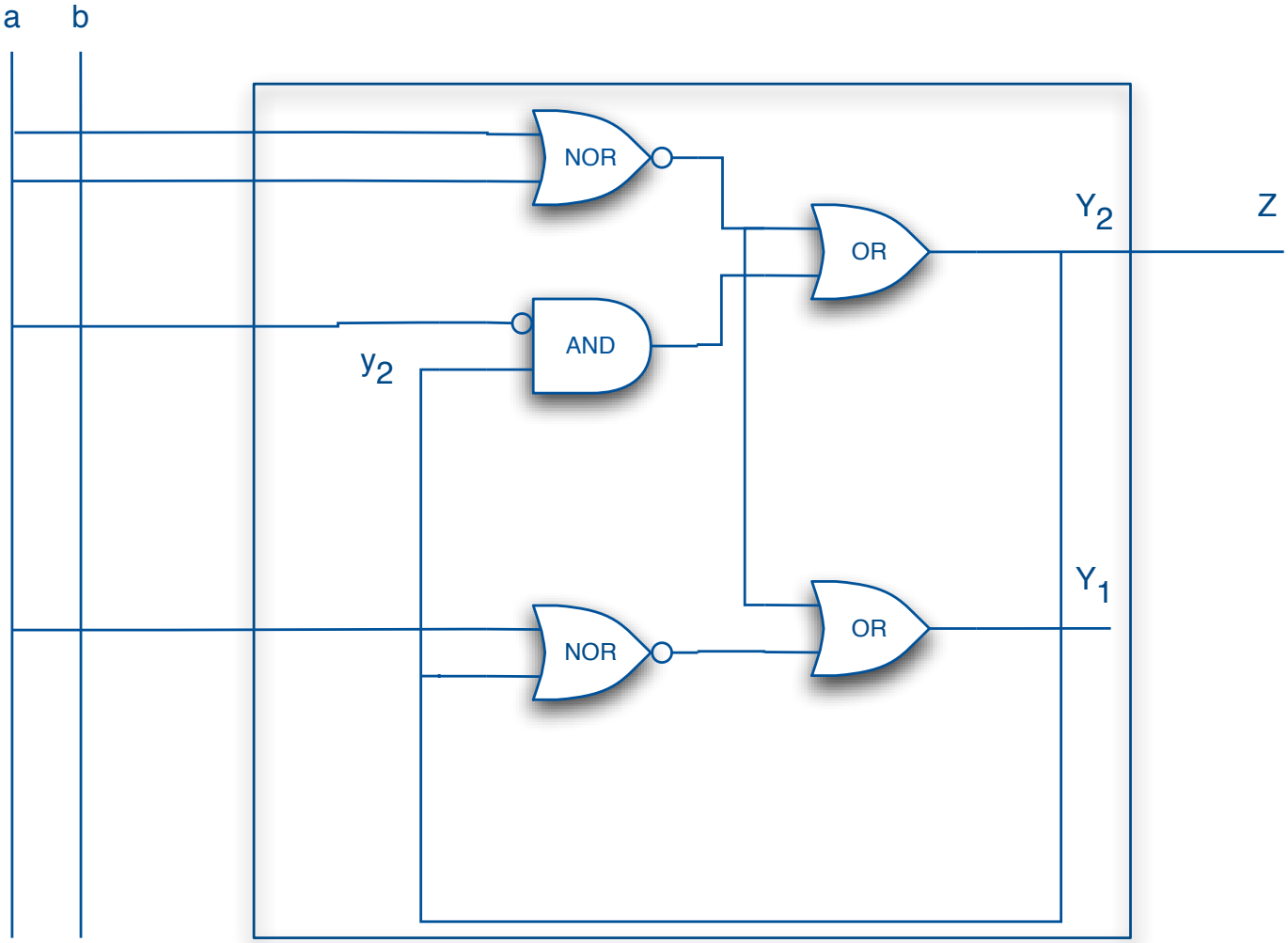
Z \ ab	00	01	11	10
00	–	0	**0**	–
01	–	**0**	0	–
11	**1**	1	–	–
10	1	**1**	–	–

$Z = y_2$

Systèmes séquentiels

Exemple complet

7. Logigramme



Simplification et réalisation

Synthèse des systèmes séquentiels

Algorithme de synthèse en trois étapes:

1. **Table primitive d'état (Table de Huffman)**
   A partir d'un cahier de charges (et/ou éventuellement le graphe d'état) dériver la **Table primitive d'état**.
   La première table construite à partir de cahier de charges:
   Un seul état stable par ligne de la table d'état.

2. **Codage des états**
   Attribution de codes binaires — on a le choix (pour le moment arbitraire)

3. **Equations logiques**
   A partir de la table d'état codé on dérive les K-Maps et les fonctions de rétroaction.

Simplification et réalisation

Synthèse des systèmes séquentiels

- La complexité de circuit obtenu lors de la synthèse est influencé par la complexité (la taille) de la table d'état.
- Le codage: attribution de $\log_2 n$ bits de code aux n états. Chaque bit de code représente:
  - une **fonction logique**,
  - et un **organe de mémoire** (délais).
- En **réduisant le nombre d'états** on **simplifie le circuit** correspondant (similaire au problème de simplification des fonctions logiques pour la réalisation des systèmes combinatoires).

Simplification et réalisation

Synthèse des systèmes séquentiels

- **Construction de la Table primitive d'états**
  (Table de Huffman)
  La partie la plus difficile car on passe d'une spécification non-formelle (cahier de charges) à une spécification formelle de système (analyse de toutes les évolutions possibles)
- Décomposition du fonctionnement de système en étapes les plus élémentaires de façon à avoir un cahier de charges complet et sans ambiguïtés.
- Les autres opérations liées à la synthèse peuvent être automatisé (ou faites à la main — si vous connaissez le mécanisme c'est simple).

Simplification et réalisation

Réduction de nombre d'état

Simplification de la table d'état avant le codage des états. Lié à la notion d'équivalence possible entre les états.

Notion de l'équivalence de deux états

Deux états sont équivalents si (on parle de l'équivalence de deux états stables):

1. Ils produisent la **même sortie** (nous sommes bien dans le cas de la machine de Moore — comme la sortie ne dépend que de l'état du système, elle doit être la même) **et**
2. Pour **toutes** les combinaisons des variables à l'entrée, **les futurs** états sont soit les **mêmes** soit **équivalents** (les deux états ont le même futur).

Simplification et réalisation

Notion de l'équivalence/fusionnement de deux états

- **Equivalences** — lorsque les états stables sont au même endroit (même combinaison des entrées).

	ab				
	00	01	11	10	
1	**1**	2	3	5	0
6	**6**	2	3	5	0

- **Fusionnements** — lorsque les état stables sont différents (ils se trouvent aux endroits différents dans la table d'état)

	ab				
	00	01	11	10	
1	**1**	2	3	7	1
7	1	2	3	**7**	1

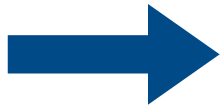
Simplification et réalisation

Notion de l'équivalence /fusionnement de deux états

Réécriture de la table d'état:

Equivalence

	ab 00	01	11	10	
1	**1**	2	3	5	0
6	**6**	2	3	5	0



	ab 00	01	11	10	
1	**1**	2	3	5	0

Fusionnement

	ab 00	01	11	10	
1	**1**	2	3	7	1
7	1	2	3	**7**	1

	ab 00	01	11	10	
1	**1**	2	3	**1**	1

Trouver l'équivalence ou fusionner deux états dans une table primitive d'état entraîne sa **SIMPLIFICATION**.

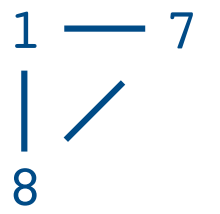
Simplification et réalisation

Notion de l'équivalence/fusionnement pour *n* états

Il est possible de généraliser la notion d'équivalence (et de fusionnement) à *n*-états:
Pour que *n*-états soient équivalents il faut que **tous** les états, **deux à deux soient équivalents**.

Ainsi pour l'exemple suivant:

	ab 00	01	11	10	
1	**1**	2	3	7	1
7	1	2	3	**7**	1
8	1	5	**8**	8	1



2=5
3=8
7=8

il faut que 2-5, 3-8 et 7-8 soient fusionnés, pour que nous puissions fusionner 1,7 et 8.

Simplification et réalisation

Recherche systématique de toutes les équivalences

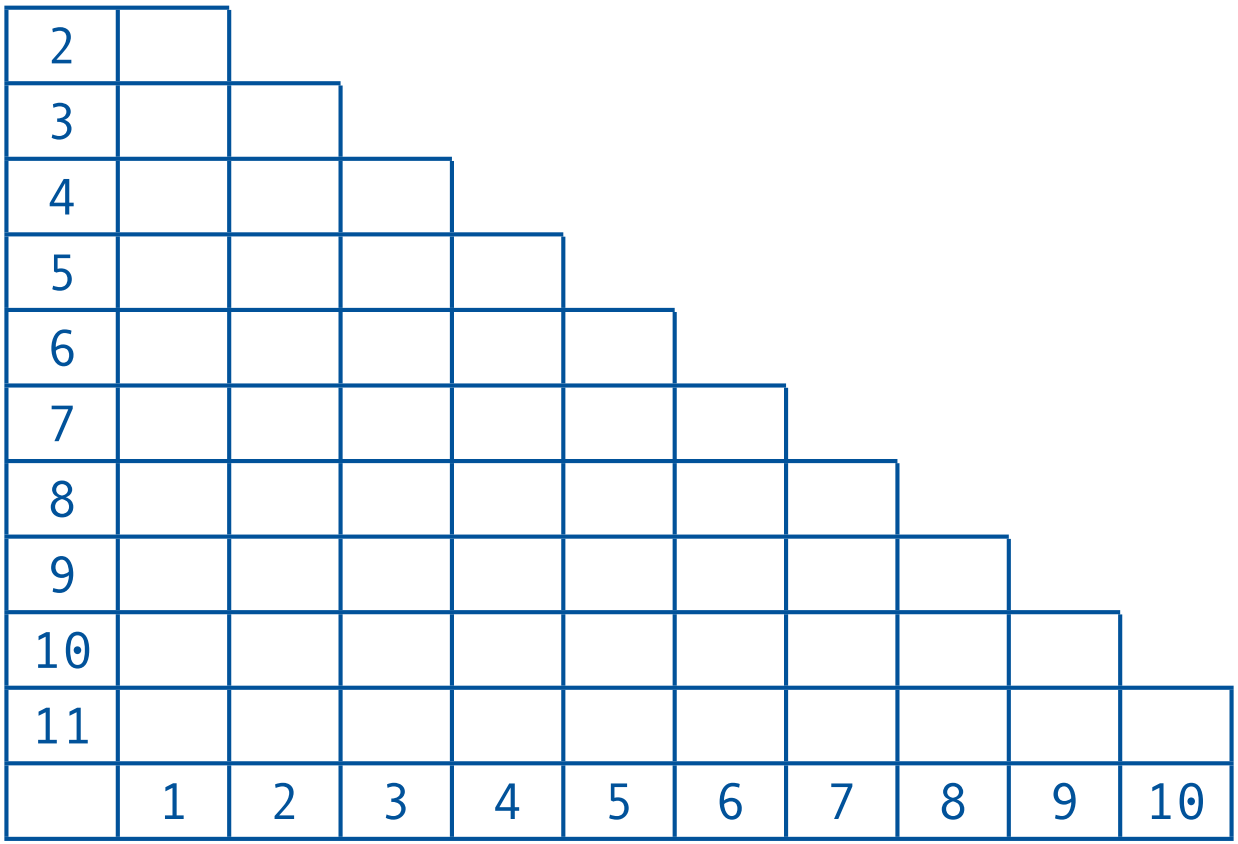
- On dresse un tableau de `(n)(n-1)/2` cases
  (on considère que chaque état est équivalant à lui même — combinaison de *n* éléments sans répétitions).
  On appellera cette table : **La table des conditions d'équivalences**.
- Chaque case de la table représente la possibilité d'équivalence et/ou de fusionnement de deux états.
- Comme nous considérons le cas de la Machine de Moore, toute paire d'états ayant la sortie différente peut être exclue.
  On note cela avec une croix (**1ère passe**).

Simplification et réalisation

Recherche systématique de toutes les équivalences

Pour une table primitive de 11 états,

la table des conditions d'équivalences :



Son application, meilleure illustration → un exemple.

Organes de mémoire

Synthèse d'un organe de mémoire

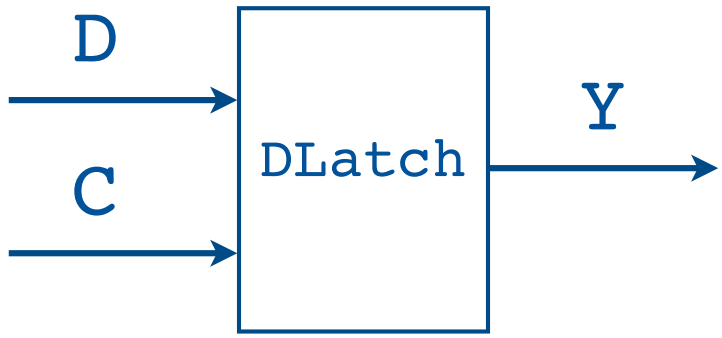
Table primitive d'état, recherche des équivalences/fusionnements, codage et dérivation d'équations logiques.

D-Latch

Mémoire Suiveuse-Bloqueuse (*Sample and Hold*)

Spécification:

- Deux entrées: D (*data*) et C (*control*)
- Une sortie Y.
- La sortie prend la valeur d'entrée lorsque C=1.



Organes de mémoire

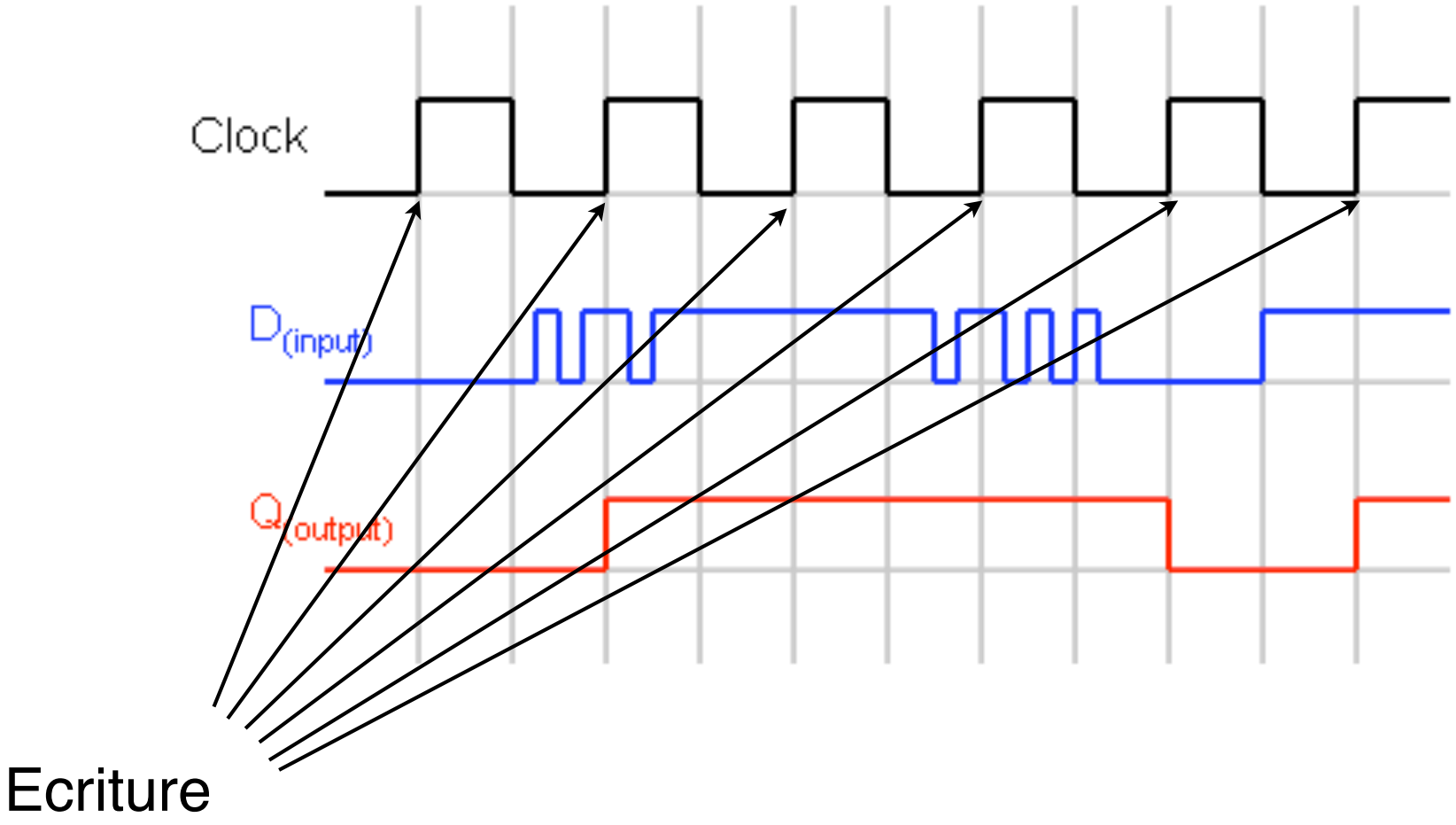
Synthèse de D Flip-Flop (*edge triggered*)

Spécification (avec un signal d'horloge C)

- La sortie `Z` prend la valeur de `D` **uniquement** lorsque **le flanc montant** de `C` (flanc montant → le passage de `0` à `1`).
- Entre deux flancs montants, l'ancienne valeur de D est maintenue (tout variation à l'entrée est ignorée).
- Dans le cas d'une variation simultanée de `C` et de `D` (nous avons le flanc montant de `C` et la valeur de `D` qui passent de 0 à 1 ou inversement), `Z` prend l'ancienne valeur de `D`, celle juste avant le flanc montant.
- Tout le reste ne donne lieu qu'à des maintiens.

Organes de mémoire

Comportement de D Flip-Flop — chronogramme



Organes de mémoire

Synthèse de D Flip-Flop — Table primitive d'état

On considère l'évolution qui mettra la sortie à 1

1. Initialement on considère une situation de départ → on a déjà retenu un 0 (Q=0) et puis aucun changement n'est pas enregistré à l'entre.
   Le CD=00 et ne change pas — c'est l'état **1**

2. Ensuite on considère l'enclenchement de D (D passe de 0 à 1) avant C — l'état **2**

3. Puis c'est le C qui enclenche — l'état **4**

Donc, la sortie est mis à 1 lorsque CD (les entrées) suivent la séquence suivante: CD=00,01,11. La machine passe par états **1**, **2** et **4**.

	CD 00	01	11	10	Q
1	**1**	2			0
2	1	**2**	4		0
3					0
4			**4**		1
5					0
					0

Organes de mémoire

Synthèse de D Flip-Flop — Table primitive d'état

On considère une autre évolution en partant de l'état **1**

1. Admettons que le C enclenche avant D en partant de l'état **1** (on a CD=00,10) → nous avons un nouvel état **3**

2. Si maintenant D enclenche après — notre filp-flop devrait ignorer ce changement à l'entrée (puisque survenu après le flanc montant de C)

3. On crée un nouvel état **5**, cette état diffère de l'état **4** par sa sortie → on doit avoir un 0 à la sortie

	CD 00	01	11	10	Q
1	**1**	2		3	0
2	1	**2**	4	3	0
3			5	**3**	0
4			**4**		1
5			**5**		0
					0

Organes de mémoire

Synthèse de D Flip-Flop

Table primitive d'état complète

Principe de mémorisation d'un 1

1. On a initialement CD=00
2. CD suit la séquence: CD=00,01,11
3. La sortie est mis à 1 (le système se trouve dans l'état **4**)
4. Ensuite imaginons que C=0
5. Par la suite quelque soit la valeur de D, la sortie gardera la valeur **1** — la succession des états 7,9,7,9,...

	CD 00	01	11	10	Q
1	**1**	2	5	3	0
2	1	**2**	4	3	0
3	6	2	5	**3**	0
4	9	7	**4**	8	1
5	6	2	**5**	3	0
6	**6**	2	5	3	0
7	9	**7**	4	8	1
8	9	7	4	**8**	1
9	**9**	7	5	10	1
10	6	2	5	**10**	0

Organes de mémoire

D Flip-Flop (Table de conditions d'équivalences)

Simplification de la table d'état primitive — Machine de Moore — conditions sur les sorties.

	CD 00	01	11	10	Q
1	**1**	2	5	3	0
2	1	**2**	4	3	0
3	6	2	5	**3**	0
4	9	7	**4**	8	1
5	6	2	**5**	3	0
6	**6**	2	5	3	0
7	9	**7**	4	8	1
8	9	7	4	**8**	1
9	**9**	7	5	10	1
10	6	2	5	**10**	0

2									
3									
4	X	X	X						
5				X					
6				X					
7	X	X	X		X	X			
8	X	X	X		X	X			
9	X	X	X		X	X			
10				X			X	X	X
	1	2	3	4	5	6	7	8	9

1ère passe

Sorties différentes

Organes de mémoire

Synthèse de D Flip-Flop

Table de conditions d'équivalences — conditions

	CD 00	01	11	10	Q
1	**1**	2	5	3	0
2	1	**2**	4	8	0
3	6	2	5	**3**	0
4	9	7	**4**	8	1
5	6	2	**5**	3	0
6	**6**	2	5	3	0
7	9	**7**	4	8	1
8	9	7	4	**8**	1
9	**9**	7	5	10	1
10	6	2	5	**10**	0

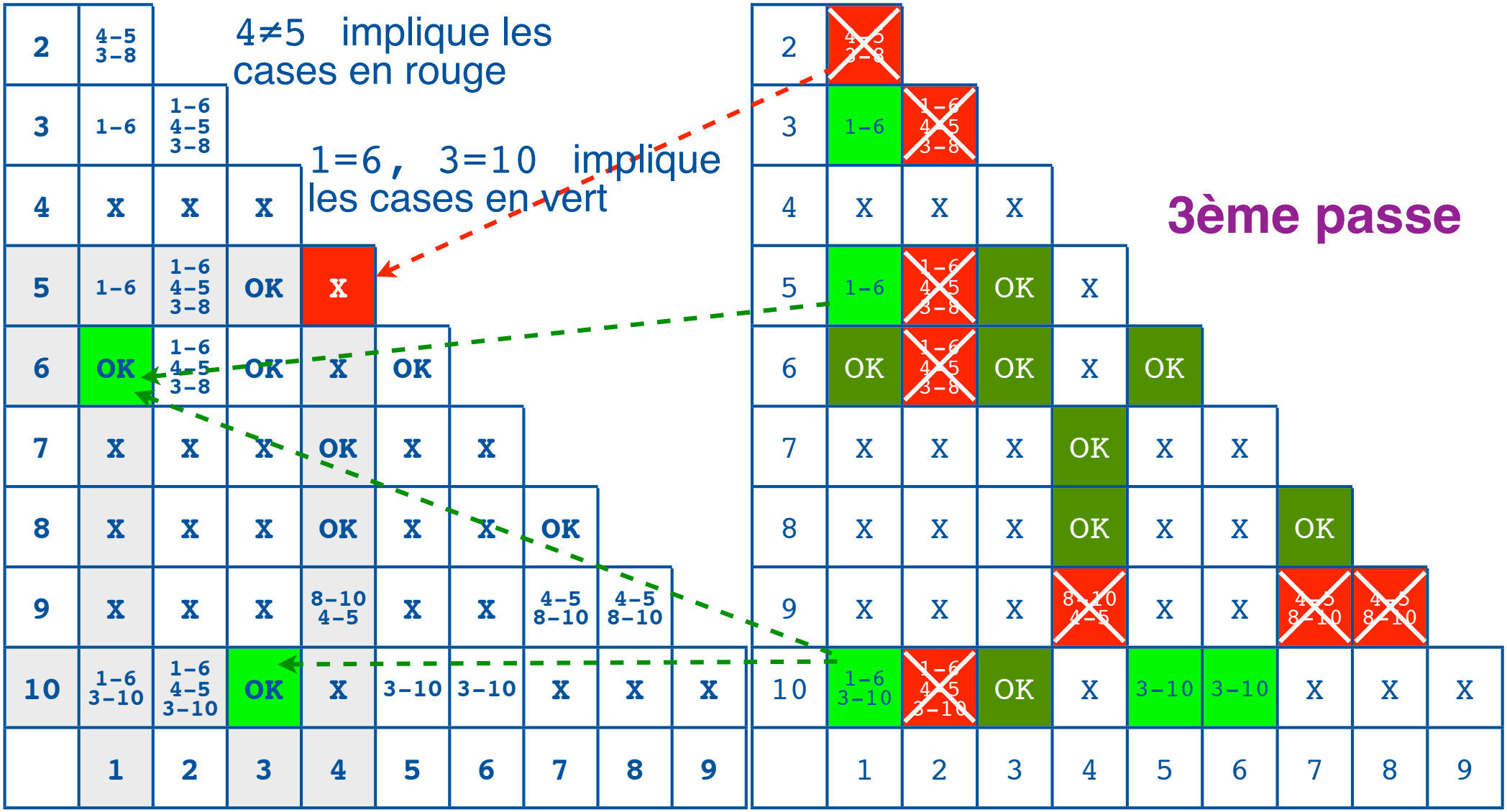
2ème passe

Conditions de fusionnements

2	4-5 3-8								
3	1-6	1-6 4-5 3-8							
4	X	X	X						
5	1-6	1-6 4-5 3-8	OK	X					
6	OK	1-6 4-5 3-8	OK	X	OK				
7	X	X	X	OK	X	X			
8	X	X	X	OK	X	X	OK		
9	X	X	X	8-10 4-5	X	X	4-5 8-10	4-5 8-10	
10	1-6 3-10	1-6 4-5 3-10	OK	X	3-10	3-10	X	X	X
	1	2	3	4	5	6	7	8	9

Organes de mémoire

Synthèse de D Flip-Flop — Table de conditions d'équivalences



Organes de mémoire

Synthèse de D Flip-Flop — Table de conditions d'équivalences

2	4-5 3-8							
3	1-6	1-6 4-5 3-8						
4	X	X	X					
5	1-6	1-6 4-5 3-8	OK	X				
6	OK	1-6 4-5 3-8	OK	X	OK			
7	X	X	X	OK	X	X		
8	X	X	X	OK	X	X	OK	
9	X	X	X	8-10 4-5	X	X	4-5 8-10	4-5 8-10
10	1-6 3-10	1-6 4-5 3-10	OK	X	3-10	3-10	X	X	X
	1	2	3	4	5	6	7	8	9

Conditions d'équivalences

```
1-3   si 1-6
1-5   si 1-6
1-10  si 1-6 et 3-10
1-6   OK équivalence
3-5   OK
3-6   OK
3-10  OK équivalence
4-7   OK
4-8   OK
5-6   OK
5-10  si 3-10
6-10  si 3-10
7-10  OK
```

Organes de mémoire

Synthèse de D Flip-Flop

Table de conditions d'équivalences

```
1-3  si 1-6 OK
1-5  si 1-6 OK
1-10 si 1-6 et 3-10
1-6  OK équivalence
3-5  OK
3-6  OK
3-10 OK équivalence
4-7  OK
4-8  OK
5-6  OK
5-10 si 3-10
6-10 si 3-10
7-8 OK
7-10 OK
```

```
1-6  → 1

3-10 → 3
```

```
1-3  OK
1-5  OK
1-10 OK
1-6  OK équivalence
3-5  OK
3-6  OK
3-10 OK équivalence
4-7  OK
4-8  OK
5-6  OK
5-10 OK
6-10 OK
7-8  OK
7-10 OK
```

Organes de mémoire

Synthèse de D Flip-Flop

Table de conditions d'équivalences: fusionnements. **Graphe de fusionnement.**

```
1-3   OK        1-3 OK                    1 ── 3
1-5   OK        1-5 OK    3-5 OK          │ ／              → 1
1-10  OK                                  5
1-6   OK
3-5   OK
3-6   OK                                  2                 → 2
3-10  OK
4-7   OK                                  4 ── 7
4-8   OK        4-7 OK                    │ ／              → 4
5-6   OK        4-8 OK      7-8 OK        8
5-10  OK
6-10  OK                                  9                 → 9
7-8   OK
7-10  OK
```

Organes de mémoire

Synthèse de D Flip-Flop

Réécriture de la table d'état après **équivalences** et **fusionnement**

Equivalences

1,6 → 1

3,10 → 3

Fusionnements

1,3,5 → 1

2 → 2

4,7,8 → 4

9 → 9

CD

	00	01	11	10	Q
1	**1**	2	5	3	0
2	1	**2**	4	8	0
3	6	2	5	**3**	0
4	9	7	**4**	8	1
5	6	2	**5**	3	0
6	**6**	2	5	3	0
7	9	**7**	4	8	1
8	9	7	4	**8**	1
9	**9**	7	5	10	1
10	6	2	5	**10**	0

CD

	00	01	11	10	Q
1	**1**	2	5	3	0
2	1	**2**	4	8	0
3	1	2	5	**3**	0
4	9	7	**4**	8	1
5	1	2	**5**	3	0
7	9	**7**	4	8	1
8	9	7	4	**8**	1
9	**9**	7	5	3	1

Organes de mémoire

Synthèse de D Flip-Flop

Réécriture de la table d'état après **équivalences** et **fusionnement**

Equivalences

1,6 → 1

3,10 → 3

Fusionnements

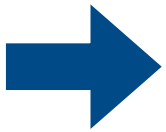
1,3,5 → 1

2 → 2

4,7,8 → 4

9 → 9

	CD 00	01	11	10	Q
1	**1**	2	5	3	0
2	1	**2**	4	8	0
3	1	2	5	**3**	0
4	9	7	**4**	8	1
5	1	2	**5**	3	0
7	9	**7**	4	8	1
8	9	7	4	**8**	1
9	**9**	7	5	3	1



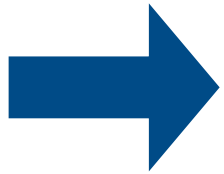
	CD 00	01	11	10	Q
1	**1**	2	**1**	**1**	0
2	1	**2**	4	4	0
4	9	**4**	**4**	**4**	1
9	**9**	4	1	1	1

Organes de mémoire

Synthèse de D Flip-Flop

Réécriture de la table d'état finale (graphe d'état correspondant plus simple...) — code décimaux ordonnés (1,2,3 et 4).

	CD 00	01	11	10	Q
1	**1**	2	**1**	**1**	0
2	1	**2**	4	4	0
4	9	**4**	**4**	**4**	1
9	**9**	4	1	1	1



	CD 00	01	11	10	Q
1	**1**	2	**1**	**1**	0
2	1	**2**	3	3	0
3	4	**3**	**3**	**3**	1
4	**4**	3	1	1	1

Simplification: on passe de 10 à 4 états; 2 variables au lieu de 4
Une importante économie de ressources pour faire la même chose !

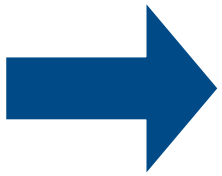
Organes de mémoire

Synthèse de D Flip-Flop

Codage des états et réécriture de la table codé (choix arbitraire):

1→00, 2→01, 3→11, 4→10

	CD				
	00	01	11	10	Q
1	**1**	2	**1**	**1**	0
2	1	**2**	3	3	0
3	4	**3**	**3**	**3**	1
4	**4**	3	1	1	1



	CD				
	00	01	11	10	Q
00	**00**	01	**00**	**00**	0
01	00	**01**	11	11	0
11	10	**11**	**11**	**11**	1
10	**10**	11	00	00	1

Organes de mémoire

Synthèse de D Flip-Flop

K-Maps et les équations logiques

	CD 00	01	11	10
00	**0**	0	**0**	**0**
01	0	**0**	1	1
11	1	**1**	**1**	**1**
10	**1**	1	0	0
y_2y_1				

	CD 00	01	11	10
00	**00**	01	**00**	**00**
01	00	**1**	11	11
11	10	**11**	**11**	**11**
10	**10**	11	00	00

	CD 00	01	11	10
00	**0**	1	**0**	**0**
01	0	**1**	1	1
11	0	**1**	**1**	**1**
10	**0**	1	0	0
y_2y_1				

Organes de mémoire

Flip-flops - Organes de mémoire

- **SR**
  Lorsque S =1 alors Q=1. La combinaison SR=11 est interdite.
  On verra la synthèse complète de SR la prochaine fois...
- **JK**
  Même chose que SR sauf que la combinaison interdite a pour effet de changer l'état.
- **D**
  La sortie suit l'entrée.
- **T**
  Changement d'état lorsque l'entrée T=1.

Organes de mémoire

Description des flip-flops

- **Table de fonctionnement** (une variante de la table d'état)
- **Equations caractéristiques**
- **Table d'excitation**
  Un flip-flop : une sortie (Q) et la sortie inversée (Q')
  On distingue l'état présent (Q) et l'état futur (Q^+) du flip-flop, nous avons donc quatre possibilités pour le couple présent-futur

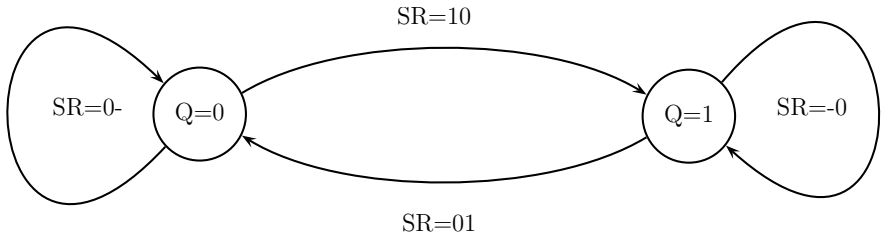
Q	Q^+		
0	0	Maintien à 0	μ_0
0	1	Enclenchement	ε
1	0	Déclenchement	δ
1	1	Maintien à 1	μ_1

Organes de mémoire

Flip-flop SR

	SR			
Q^+	00	01	11	10
0	**0**	**0**	–	1
1	**1**	0	–	**1**

Table d'état



Graphe d'état

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	–

Table de fonctionnement

	Q	Q^+	S	R
μ_0	0	0	0	–
ε	0	1	1	0
δ	1	0	0	1
μ_1	1	1	–	0

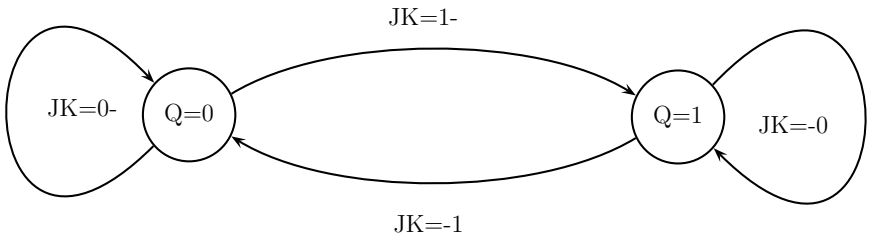
Table d'excitation

Organes de mémoire

Flip-flop JK

	JK			
Q^+	00	01	11	10
0	**0**	**0**	1	1
1	**1**	0	0	**1**

Table d'état



Graphe d'état

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Q'

Table de fonctionnement

	Q	Q^+	J	K
μ_0	0	0	0	-
ε	0	1	1	-
δ	1	0	0	1
μ_1	1	1	-	0

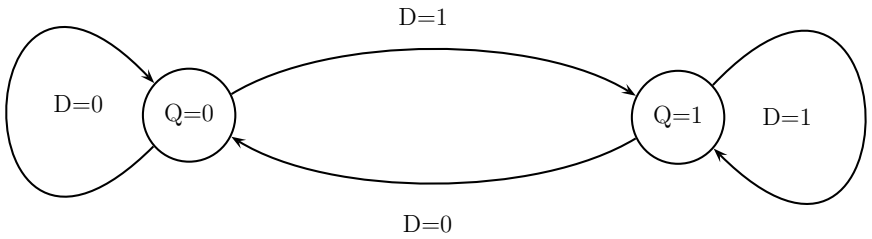
Table d'excitation

Organes de mémoire

Flip-flop D

	D	
Q^+	0	1
0	**0**	1
1	0	**1**

Table d'état



Graphe d'état

D	Q^+
0	0
1	1

Table de fonctionnement

	Q	Q^+	D
μ_0	0	0	0
ε	0	1	1
δ	1	0	0
μ_1	1	1	1

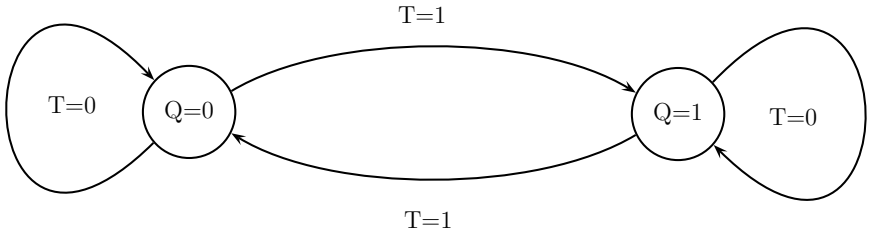
Table d'excitation

Organes de mémoire

Flip-flop T

	T	
Q^+	0	1
0	**0**	1
1	**1**	0

Table d'état



Graphe d'état

T	Q^+
0	0
1	1

Table de fonctionnement

	Q	Q^+	T
μ_0	0	0	0
ε	0	1	1
δ	1	0	1
μ_1	1	1	0

Table d'excitation

Organes de mémoire

Représentation graphique des mémoires

Lorsque l'on utilise un signal de contrôle (signal horloge — Clk — le triangle indique qu'il s'agit d'un flip-flop) ou pas: