ELEC-H-305
Circuits logiques et numériques
2011-2012

Cours 11

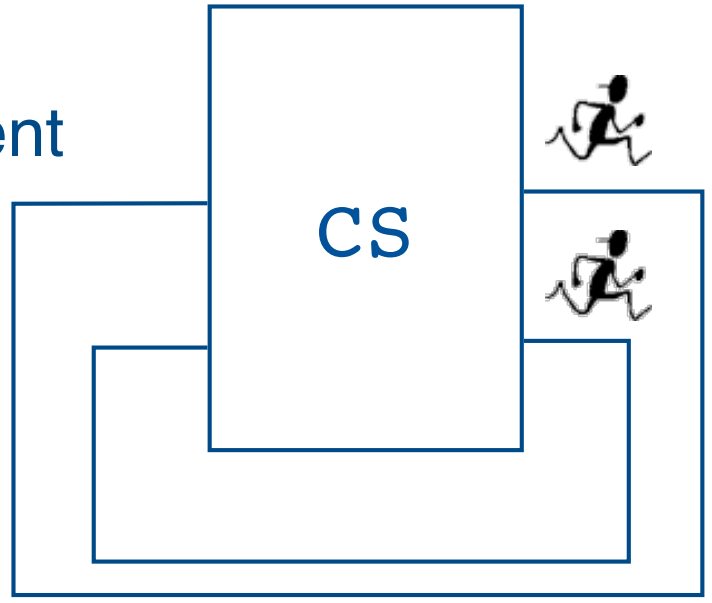
Dragomir Milojevic
dmilojev@ulb.ac.be

Aujourd'hui

1. Rappel sur les circuits logiques asynchrones et les problèmes des courses critiques
2. Résolution des courses critiques à l'aide de la **logique synchrone**; Notions de la **performance** d'un automate séquentiel asynchrone et synchrone
3. Organes de mémoires élémentaires (**SR, JK, D, T**); Différentes modes de représentation (rappel) : tables d'excitation.
4. Algorithme de synthèse des circuits logiques synchrones
5. **Exemple complet** de synthèse : Détecteur de sens de rotation (la réalisation asynchrone et synchrone)

Circuits asynchrones

- Le problème des courses critiques = le changement **simultané** de plus qu'une variable d'état (distance de Hamming entre y_i et $Y_i > 1$).
- Les courses critiques : une conséquence de l'implémentation physique de circuit → les distances impliquent les délais de propagation dans les fils
- Dans un automate **asynchrone** les **courses critiques non-résolues** peuvent causer un comportement non désiré du système → une action sur des entrées peut provoquer l'aboutissement de l'automate dans un état autre que celui prévu initialement



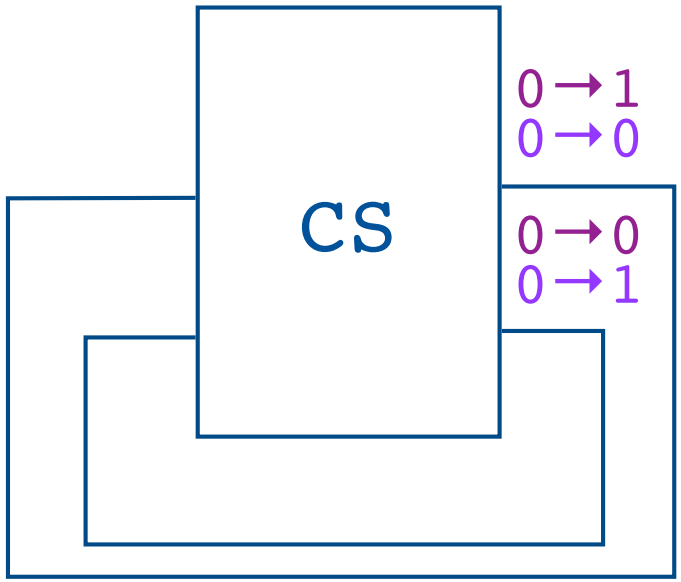
Circuits asynchrones

- La réalisation des circuits asynchrones : la Table d'état codée ne peut présenter des courses critiques.
- Résolution du problème des courses critiques à l'aide des **trois méthodes** appliquées dans l'ordre :
  1. Le codage sans courses
  2. En agissant sur les transitions
  3. En ajoutant une variable supplémentaire
- Synthèse de l'automate s'achève avec la synthèse de la **fonction de sortie** (en précisant la sortie pour toutes les transitions possibles afin d'éviter des multiples variations à la sortie)

Circuits asynchrones

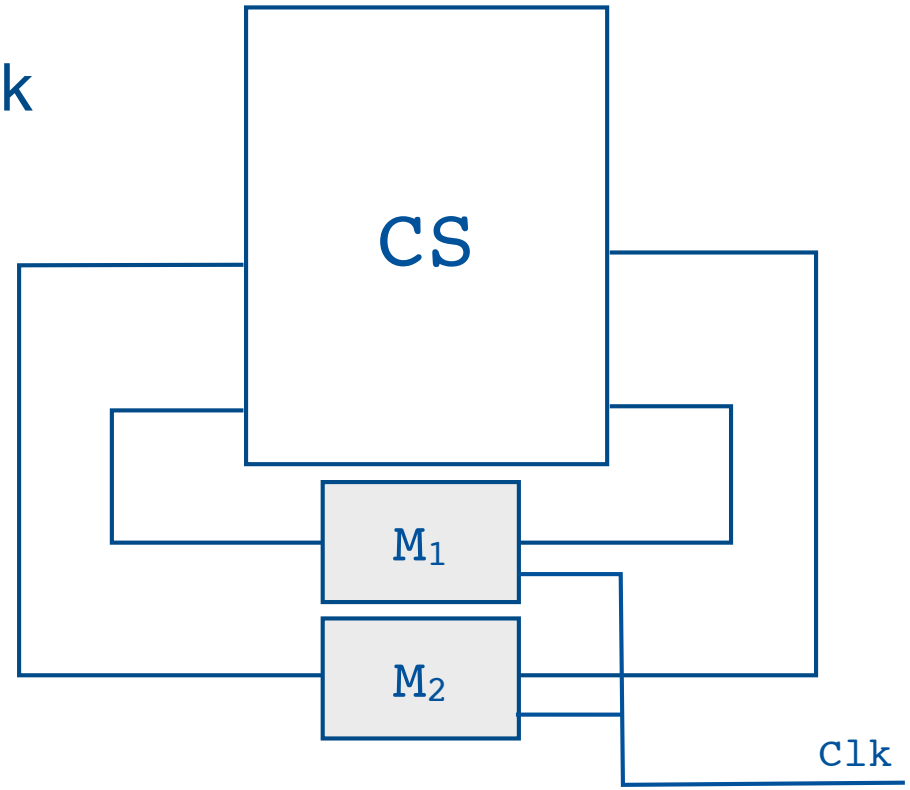
- La performance d'un automate asynchrone : le changement d'état du système est provoqué par le changement de **LA** variable d'état (puisque il n'y a qu'une seule variable qui peut changer de valeur à la fois)
- Le système se trouve dans un nouvel état dès que la variable d'état est "arrivée" à l'entrée
- La **vitesse** de l'évolution de l'automate (passage d'un état à l'autre) dépend de quel état présent nous partons et vers quel état futur on va

  Ainsi dans l'exemple, le système "passe plus vite" de l'état initial 00 à l'état 01, qu'à l'état 10. Pq?



Circuits synchrones

- Une autre façon de résoudre les courses critiques serait de **mémoriser** les variables d'état
- La **mise à jour** des variables d'état se fait alors à des **intervalles réguliers**, imposée par un signal externe **commun** à tous les organes de mémoire du système → Horloge, Clock, Clk
- La **période** avec laquelle les données seront mis à jour (la période du signal de synchronisation) sera suffisamment longue pour permettre la stabilisation des fonctions logiques de rétroaction



Circuits synchrones

- Fonctionnement **synchrone** : les variables d'état changent leur valeur au **même moment**
- La mise à jour des variables d'état matérialise le concept de la mise à jour de l'état du système : à **un** moment donné le futur devient présent (indiquer ce moment dans la Table d'état)
- Nous avons l'habitude de noter cela :

$$Y_i = f(E, y_i)$$

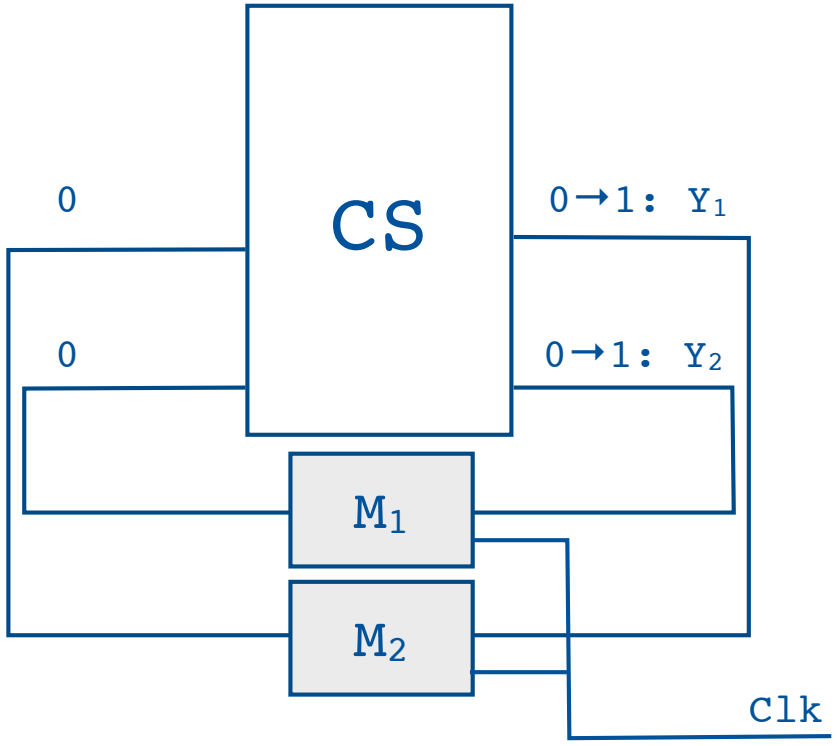
mais il ne s'agit pas d'une égalité mais d'une **affectation** (en fonction des entrées et de l'état présent, le système évolue vers le nouvel état):

$$NextState \leftarrow CurrentState$$

Circuits synchrones

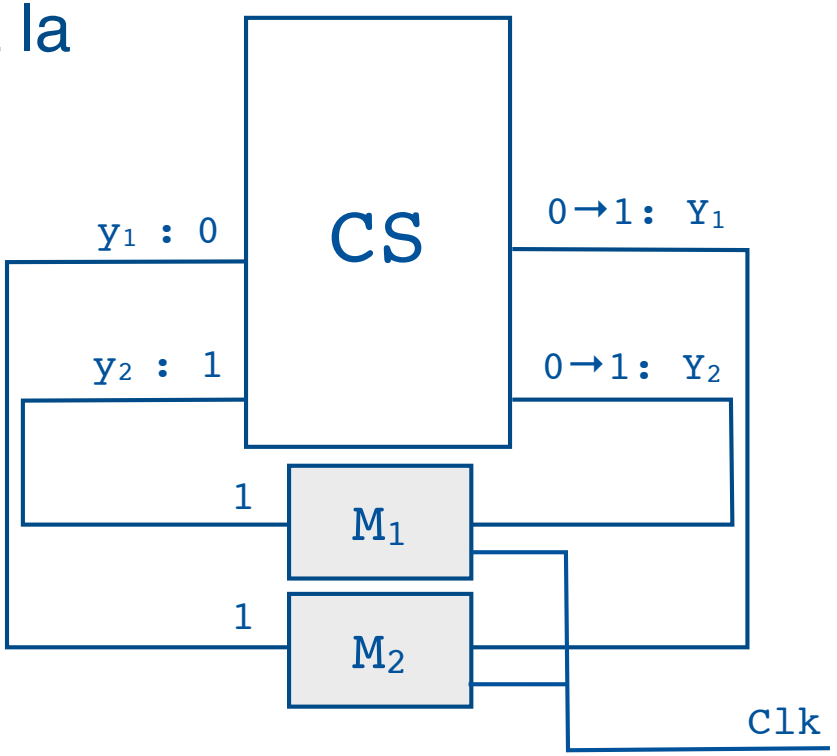
Considérons l'exemple suivant :

- On est dans l'état `00` et on doit passer à l'état `11` (il y a donc une course critique)
- Le système voit à son entrée : les entrées du système et les variables d'état présent. Les variables d'état présent gardent leur valeur tant que la mise à jour des états n'a pas eu lieu (le signal de synchronisation est à 0)
- Le système doit avoir suffisamment de temps pour faire une transition



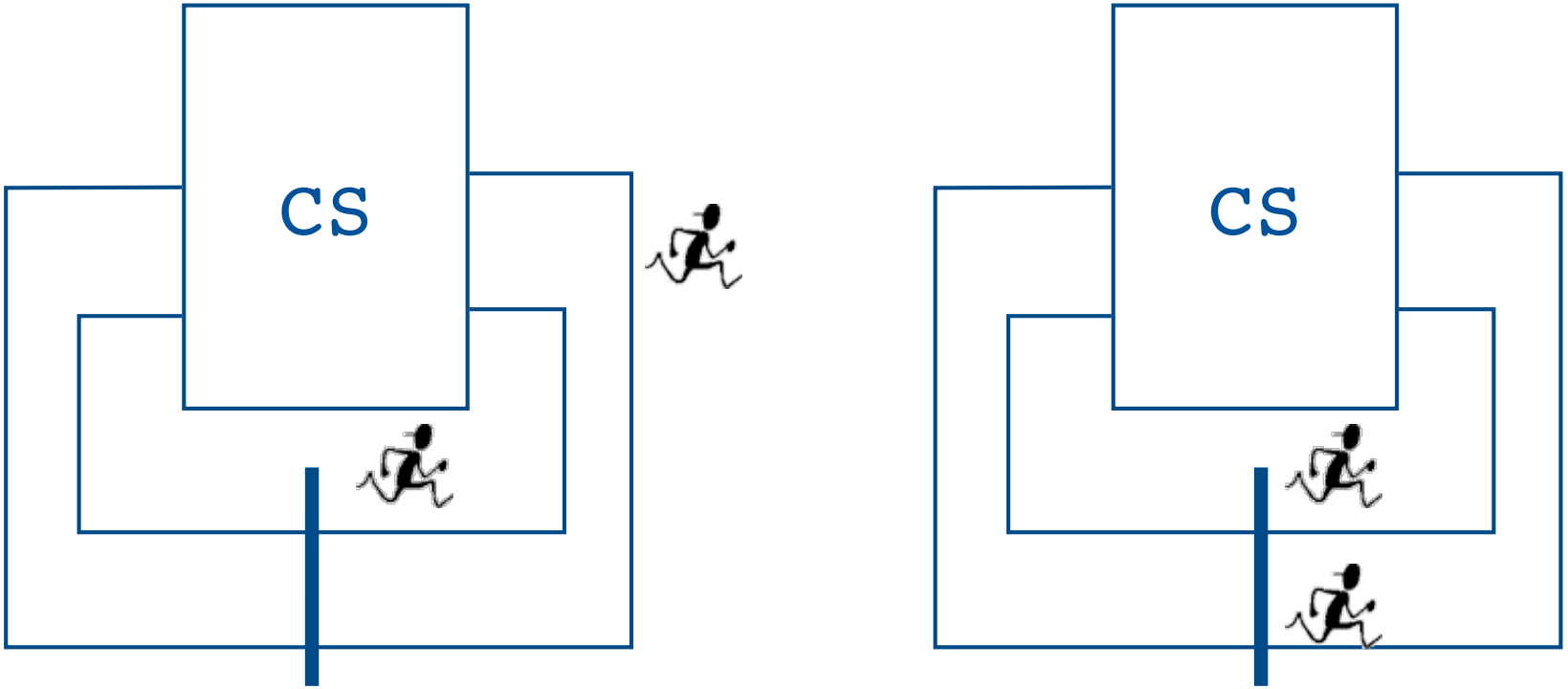
Circuits synchrones

- Lorsque le signal de synchronisation est à `1` : la mise à jour des deux organes de mémoire (ils vont avoir la valeur `11`); Ensuite le signal de synchronisation passe à `0`; Le futur devient présent.
- Le problème des courses depuis des mémoires jusqu'aux portes associées aux fonctions de rétroaction n'a pas d'influence sur le fonctionnement, car tout transitoire à la sortie de CS sera ignorée
- Les variations transitoires de *Yi* à l'entrée des mémoires seront **ignorées** par le système (le signal de synchronisation l'interdit) → y_1y_2=01 provoque un transitoire pour Y_1Y_2 qui sera ignoré ...



Circuits synchrones

La période du signal de synchronisation est déterminée par l'élément le plus lent car il faut donner suffisamment du temps pour que le système puisse se stabiliser. C'est comme s'il y avait une barrière qui va faire attendre le coureur ayant le plus long chemin à parcourir.



Circuits synchrones

Performance d'un système séquentiel synchrone

- C'est le nivellement par le bas:
  - La vitesse d'évolution de l'automate est fixée par la variable d'état la plus "lente"
  - A cause de la synchronisation, la variable la plus "rapide" va devoir attendre la variable la plus "lente"
- A priori : le système synchrone est moins performant par rapport à un système asynchrone → **Expliquer pourquoi ?**
- Les systèmes asynchrones ont été évités jusqu'à présent (la complexité de réalisation pour un grand nombre d'états) ...

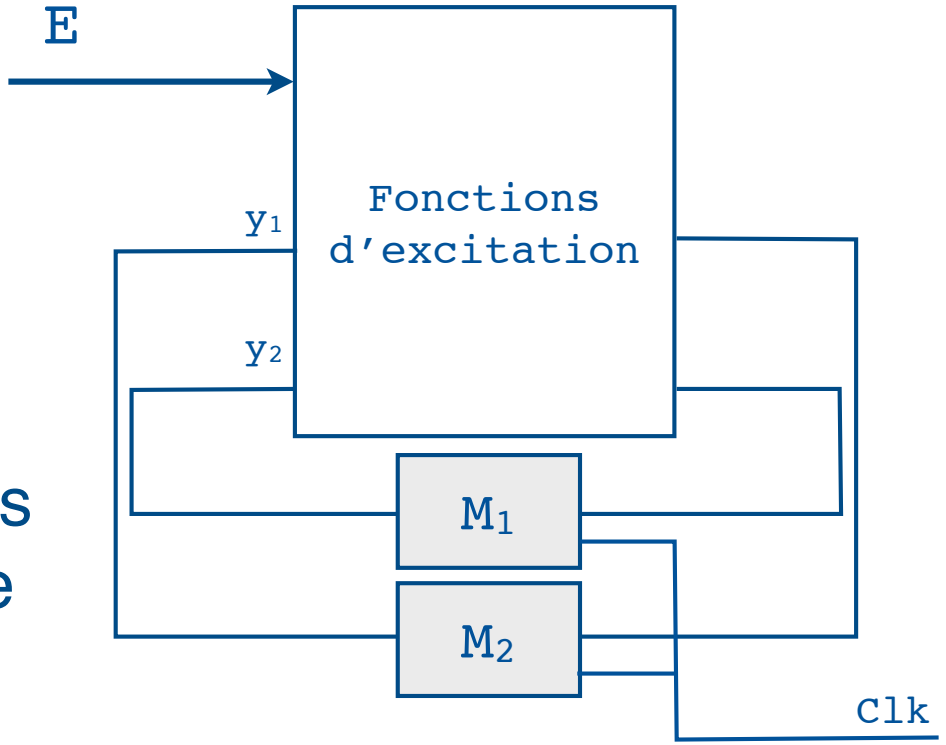
  Mais aujourd'hui on y revient pour des raisons de performance.

Circuits synchrones

- Dans l'exemple précédent chaque variable d'état est calculée à l'aide de la logique combinatoire et de la valeur sauvegardée dans une mémoire, piloté par un signal de synchronisation (l'horloge)
- Mais nous pouvons aussi utiliser des organes de mémoire standards (flip-flops : SR, JK, D et T), commandées par des fonctions logiques dérivées de la Table d'état codée
  → Les fonctions logiques de commande des flip-flops : **les fonctions d'excitation des flip-flops**.
- Synthèse d'un circuit séquentiel synchrone revient donc à **la synthèse des fonctions d'excitation pour un organe de mémoire** standard particulier

Circuits synchrones

- *Mi* sont des **organes de mémoire standard** (donc les flip-flops SR, JK, D, T)
- Les **fonctions d'excitation** sont utilisées pour piloter le contenu de ces mémoires
- Les contenues des mémoires représentent les états présents du système
- Ainsi si *Mi* est un flip-flop JK, nous allons dériver les fonctions pour J et K pour chaque organe de mémoire *i*

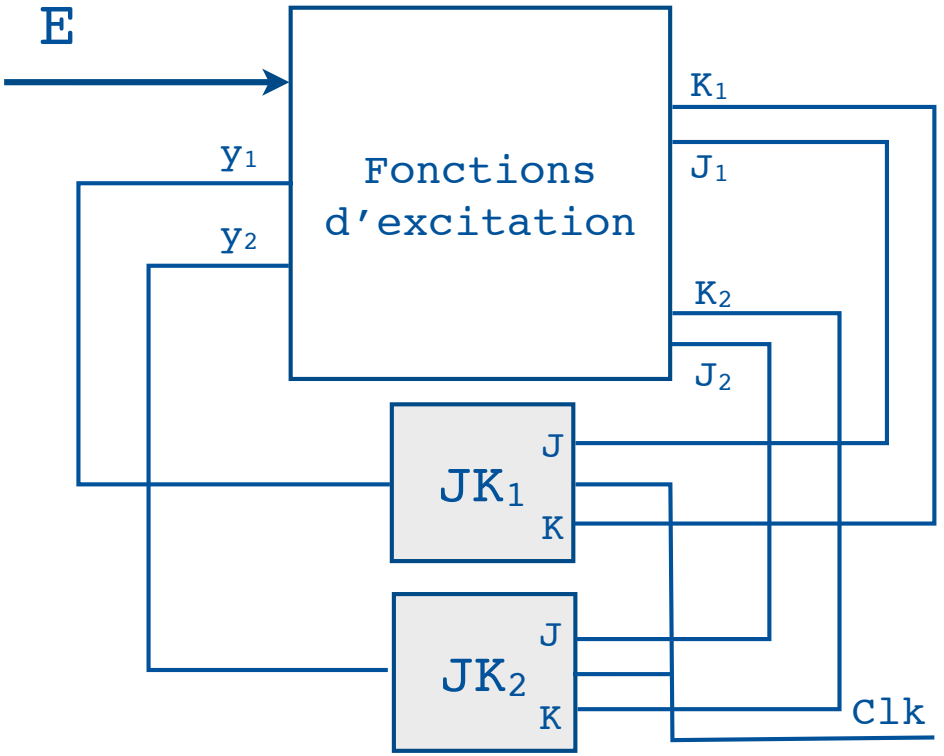


- Il y aura donc 4 fonctions d'excitation qui contrôlent le contenu des deux mémoires

Circuits synchrones

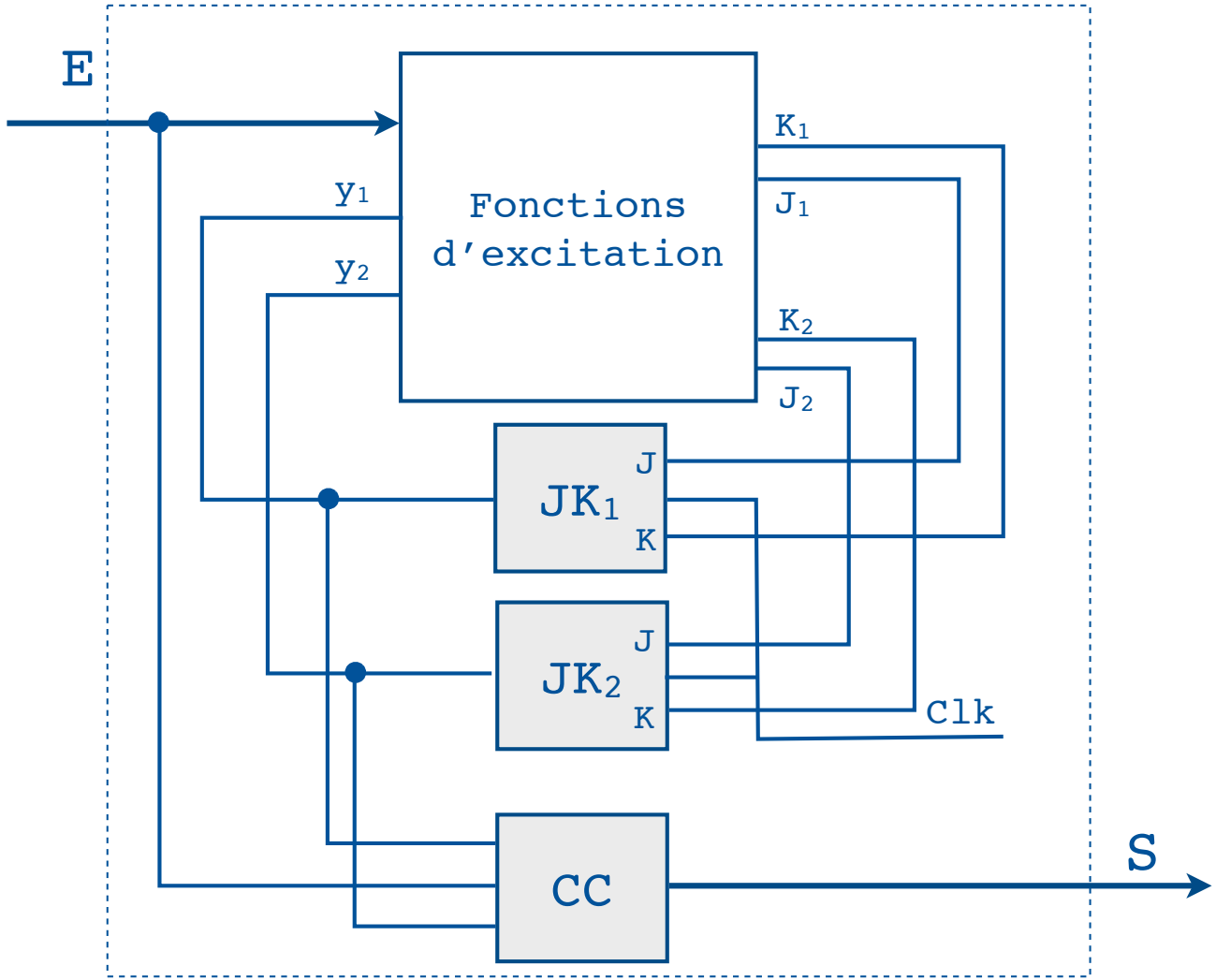
Ainsi pour un système à deux variables et les états mémorisés à l'aide des deux flip-flops JK :

$M_i = f(E, y_i)$



Circuits synchrones

... et la sortie de système, calculée en fonction de l'état (et des entrées). Le système complet devient alors:



Organes de mémoire

Les états : Q — présent, Q^+ — futur.

Les tables d'états pour des organes de mémoire SR, JK, D et T:

SR

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

D

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

JK

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

T

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

... il ne faut pas les connaître par coeur.

Organes de mémoire

- Lors de la synthèse, à la place des tables d'états codés, on utilise les **Tables d'excitation**
- Elles reflètent les changements des variables d'états du système
- Nous avons quatre possibilités pour : *CurrentState*, *NextState* (pour chaque variable d'état du système)

Présent → *Futur*		
0 → 0	Maintient à 0	μ_0
0 → 1	Enclenchement	ε
1 → 0	Déclenchement	δ
1 → 1	Maintient à 1	μ_1

Organes de mémoire

On construit la **Table d'excitation** à partir de la **Table d'état**.
Pour les flip-flops SR et JK :

SR

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

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

JK

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

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

Organes de mémoire

... et pour les flip-flops T et D:

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

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

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

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

Circuits synchrones

Algorithme de synthèse des circuits synchrones

1. Etablissement de la **Table d'état primitive d'état** à partir de cahier de charges verbal du problème
2. En fonction du modèle de la machine à synthétiser (Moore ou Meally) **réduction de nombre d'états** (obtention de la **Table d'état réduite**)
3. Codage des états (on attribue n'importe quel code)
4. Etablissement de la **Table d'excitation** de l'automate
5. En fonction de type des organes de mémoire (SR, JK, D et T) établissement des K-Maps correspondants aux **fonctions d'excitation**
6. Etablissement des fonctions d'excitation
7. Etablissement de la fonction de sortie

Circuits synchrones

Synthèse des circuits synchrones — différence par rapport aux système asynchrones :

Algorithme est le même, sauf

que il n'y ait pas d'étape de résolution

des courses critiques.

La Table d'états (réduite) codée **PEUT** présenter des courses critiques!

Synthèse synchrone

- Le choix de type d'organe de mémoire influence la complexité finale du circuit (le nombre de portes logiques nécessaires à la fabrication de circuit — espace de circuit intégré)
- A ce stade on ne cherche pas l'implémentation optimale : l'organe de mémoire est imposé
- On fourni la table d'état de l'organe de mémoire, mais il faut savoir en dériver la table d'excitation

Vous disposez à présent de tous les outils de synthèse des circuits logiques combinatoires/séquentiels asynchrones et synchrones.

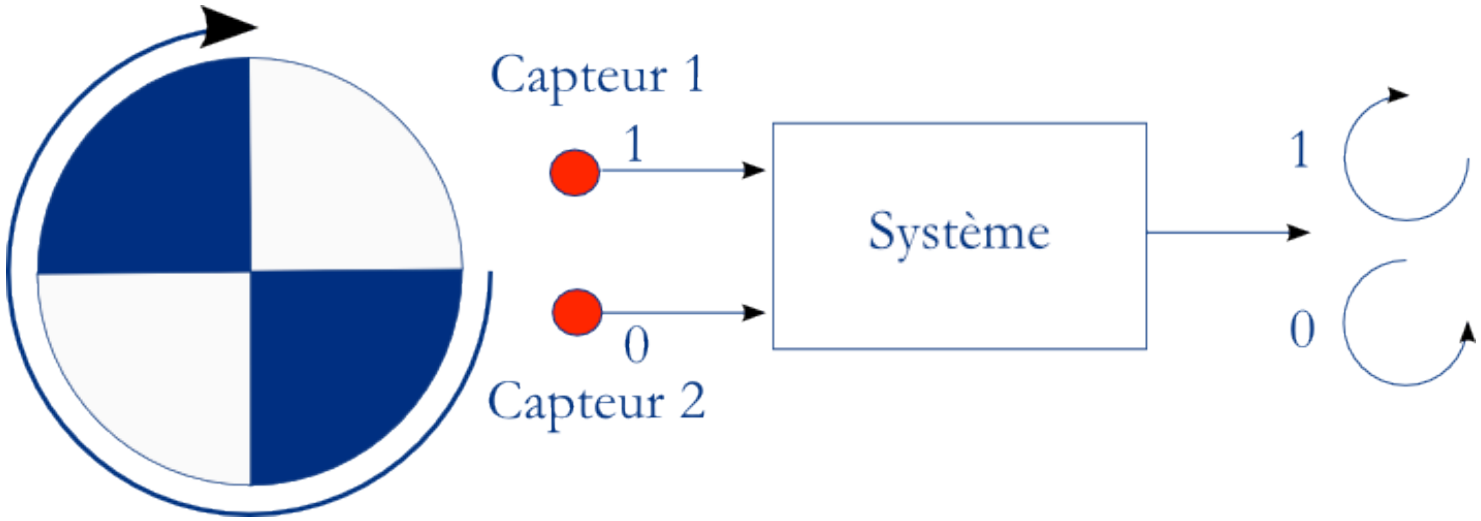
Mais il faut voir en pratique les étapes 5 et 6,
donc un exemple complet!

Exemple complet

Synthétiser le circuit logique répondant au cahier de charges suivant :

Un disque divisé en quatre secteurs coloriés alternativement en blanc et noir est fixé sur l'arbre d'un moteur. Devant ce disque se trouvent deux capteurs optiques. Chaque capteur fourni un signal logique 1 lorsqu'il est sur la partie blanche du disque (0 pour le noir).

On souhaite déterminer le sens de rotation du moteur.



Exemple complet

- Pour le problème : on indique le sens des aiguilles d'une montre avec un 1 à la sortie (0 dans le sens contraire)
- On peut utiliser :

  A. **Machine de Moore** (en respectant la valeur de sortie)

  B. **Machine de Meally** (sans respecter la valeur de sortie).

- Nous allons faire la synthèse :

  I. **Système asynchrone**

  II. **Système synchrone**

  On peut ensuite comparer les deux implémentations

Exemple complet

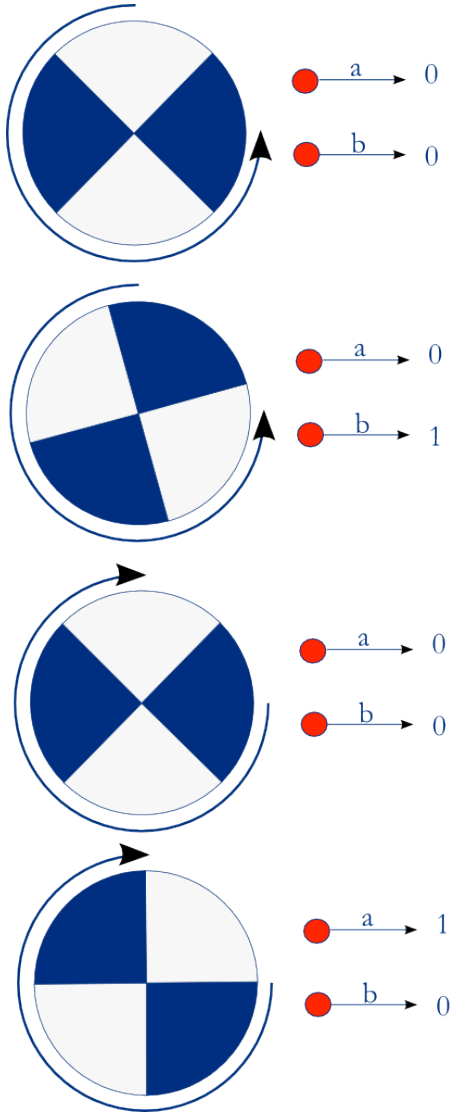
Analyse préalable de problème

- Variations simultanées ne sont pas possibles. Donc toute transition de `00` à `11`, de `01` à `10`, de `10` à `01` et de `11` à `00` va être remplacée par un *don't care* dans la **Table d'état primitive**
- Pour l'état initial on fait une hypothèse : on imagine que l'entrées valent `00` et que le système tourne dans le sens des aiguilles d'une montre
- Pour le prochain état des entrées nous avons deux possibilités :
  - Système change de sens (1 vers 2)
  - Système garde le même sens (1 vers 3)

Exemple complet

Construction de la table primitive des états:

	ab 00	01	11	10	z
1	**1**	2	–	3	1
2		**2**	4	–	0
3				**3**	1
4			**4**		0
5					
6					



- **Etat 1**: l'état initial (entrées: 00) et changement de sens de rotation **Etat 2**, puis **Etat 4**

Transition 1 à 2: changement de sens initial

- **Etat 3**: l'état initial (entrées: 00) et le même sens de rotation

Transition 1 à 3: même sens initial

Exemple complet

La **Table primitive des états** complète :

	ab 00	01	11	10	z
1	**1**	2	–	3	1
2	1	**2**	4	–	0
3	5	–	6	**3**	1
4	–	7	**4**	8	0
5	**5**	2	–	3	0
6	–	7	**6**	8	1
7	1	**7**	4	–	1
8	5	–	6	**8**	0

- A partir de l'**Etat 3** :
  - **Etat 5** : changement de sens (entrées: 00,10,00)
  - **Etat 6** : on continue de tourner dans le même sens (entrées: 00,10,11)
- A partir de l'**Etat 4** :
  - **Etat 7** : changement de sens
  - **Etat 8** : même sens

Exemple complet

A. Machine de Moore — Equivalences et fusionnements → graphe de condition d'équivalences en respectant la valeur de la sortie. Première itération :

	ab 00	01	11	10	z
1	**1**	2	–	3	1
2	1	**2**	4	–	0
3	5	–	6	**3**	1
4	–	7	**4**	8	0
5	**5**	2	–	3	0
6	–	7	**6**	8	1
7	1	**7**	4	–	1
8	5	–	6	**8**	0

2	X						
3	1–5	X					
4	X	1–7	X				
5	X	1–5	X	2–7 3–8			
6	2–7 3–8	X	3–8	X	X		
7	2–7	X	1–5	X	X	4–6	
8	X	1–5	X	4–6	3–8	X	X
	1	2	3	4	5	6	7

Exemple complet

A. Machine de Moore — Equivalences et fusionnements

Deuxième itération

2	X						
3	1x5	X					
4	X	1x7	X				
5	X	1x5	X	2-7 3-8			
6	2-7 3-8	X	3x8	X	X		
7	2x7	X	1x5	X	X	4x6	
8	X	1x5	X	4x6	3x8	X	X
	1	2	3	4	5	6	7

1-3 si 1-5; 1-5 non-équivalents : donc 2-5, 2-8, 3-7 ne sont pas équivalents non plus

1-6 si 2-7 et 3-8;
2-7 non-équivalents: donc 4-5, non plus

Pas de simplifications possibles!!!

Exemple complet

B. Machine de Meally — Equivalences et fusionnements

Graphe de condition d'équivalences sans respecter la valeur de la sortie. Les états 1–5, 2–7, 3–8 et 4–6 ne sont pas équivalents (les états stables à sortie différentes sont dans la même colonne).

	ab 00	01	11	10	Z
1	**1**	2	–	3	1
2	1	**2**	4	–	0
3	5	–	6	**3**	1
4	–	7	**4**	8	0
5	**5**	2	–	3	0
6	–	7	**6**	8	1
7	1	**7**	4	–	1
8	5	–	6	**8**	0

2	OK						
3	1–5	1–5					
4	3–8	2–7	4–6 3–8				
5	X	1–5	OK	2–7 3–8			
6	2–7 3–8	2–7	3–8	X	2–7		
7	2–7	X	1–5	OK	2–7	4–6	
8	1–5 3–8	1–5	X	4–6	3–8	OK	1–5
	1	2	3	4	5	6	7

Exemple complet

B. Machine de Meally — Equivalences et fusionnements

Deuxième itération:

2	OK						
3	1≠5	1≠5					
4	3≠8	2≠7	4–6 X 3–8				
5	X	1≠5	OK	2–7 X 3–8			
6	2–7 X 3–8	2≠7	3≠8	X	2≠7		
7	2≠7	X	1≠5	OK	2≠7	4≠6	
8	1–5 X 3–8	1≠5	X	4≠6	3≠8	OK	1≠5
	1	2	3	4	5	6	7

1 —— 2 **1**

3 —— 5 **2**

4 —— 7 **3**

6 —— 8 **4**

Exemple complet

B. Machine de Meally — Equivalences et fusionnements

Fusionnements : **Table d'états réduite**

	ab 00	01	11	10	z
1	**1**	2	–	3	1
2	1	**2**	4	–	0
3	5	–	6	**3**	1
4	–	7	**4**	8	0
5	**5**	2	–	3	0
6	–	7	**6**	8	1
7	1	**7**	4	–	1
8	5	–	6	**8**	0

1 —— 2 → 1

3 —— 5 → 2

4 —— 7 → 3

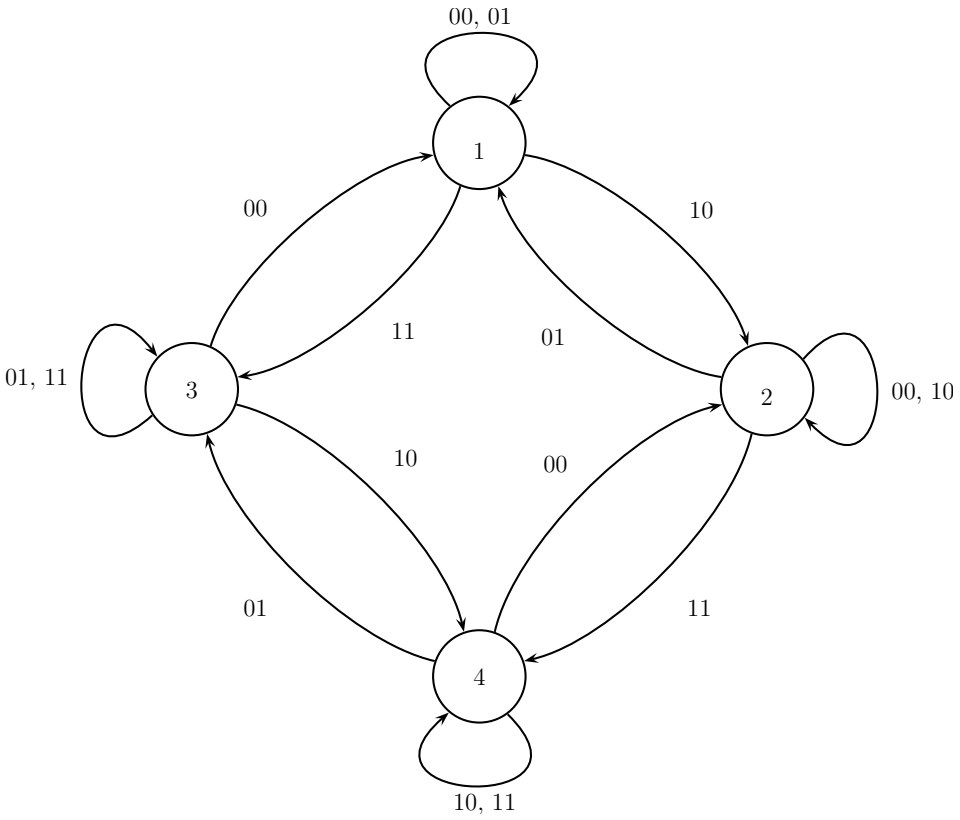
6 —— 8 → 4

	ab 00	01	11	10
1	**1/1**	**1/0**	3	2
2	**2/0**	1	4	**2/1**
3	1	**3/1**	**3/0**	4
4	2	3	**4/1**	**4/0**

Exemple complet

Détecteur de sens de rotation : la **Table d'états réduite** et le **Graphe d'états** (pour information uniquement)

	ab 00	01	11	10
1	**1/1**	**1/0**	3	2
2	**2/0**	1	4	**2/1**
3	1	**3/1**	**3/0**	4
4	2	3	**4/1**	**4/0**

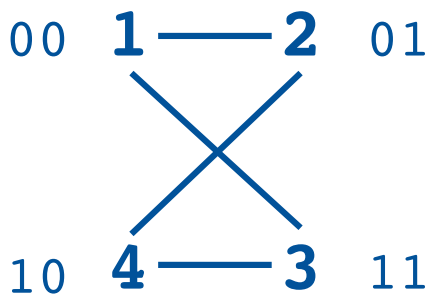


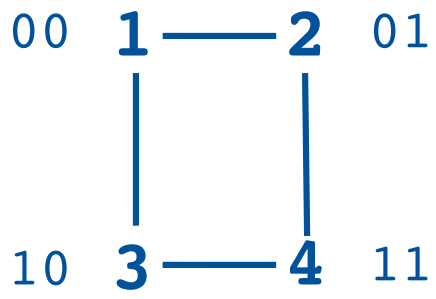
Exemple complet

I. Détecteur de sens de rotation : réalisation asynchrone

Graphe de codage des états, le codage et la **Table d'état codé**

	ab 00	01	11	10
1	**1/1**	**1/0**	3	2
2	**2/0**	1	4	**2/1**
3	1	**3/1**	**3/0**	4
4	2	3	**4/1**	**4/0**





	ab 00	01	11	10
00	**00**	**00**	10	01
01	**01**	00	11	**01**
11	01	10	**11**	**11**
10	00	**10**	**10**	11

Exemple complet

I. Détecteur de sens de rotation : réalisation asynchrone

Fonctions de rétroaction

Y_1Y_2 \ ab	00	01	11	10
00	**00**	**00**	10	01
01	**01**	00	11	**01**
11	01	10	**11**	**11**
10	00	**10**	**10**	11

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

$$Y_1 = ab + y_1(a+b)$$

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

$$Y_2 = ab' + y_2(a+b')$$

Exemple complet

I. Détecteur de sens de rotation : réalisation asynchrone

Fonction de sortie — résolution des transitions. Attention c'est une Machine de Meally, une transition est partagée par deux états stables ayant la sortie différente : p.e. de **1** à **3**, de **2** à **1**, ...

	ab 00	01	11	10
1	**1/1**	**1/0**	3	2
2	**2/0**	1	4	**2/1**
3	1	**3/1**	**3/0**	4
4	2	3	**4/1**	**4/0**

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

$$Z = b' y'_1 y'_2 + a\, y_1 'y_2 + a'b y_1 + b y_1 y_2'$$

Exemple complet

II. Détecteur de sens de rotation : réalisation synchrone

1. On part avec la table d'état réduite et on attribue un codage **SANS** considérer les courses critiques (marquées en rouge)

	ab			
	00	01	11	10
1	**1/1**	**1/0**	3	2
2	**2/0**	1	4	**2/1**
3	1	**3/1**	**3/0**	4
4	2	3	**4/1**	**4/0**

	ab			
	00	01	11	10
00	**00/1**	**00/0**	11	01
01	**01/0**	00/0	10	**01/1**
11	00	**11/1**	**11/0**	10
10	01	11	**10/1**	**10/0**

Exemple complet

2. Réalisation de la table d'excitation pour l'automate (ceci est indépendant de type de flip-flop utilisé) en partant de la Table d'états réduite ... (partie commune avec la réalisation asynchrone)

	ab 00	01	11	10
00	**00/1**	**00/0**	11	01
01	**01/0**	00	10	**01/1**
11	00	**11/1**	**11/0**	10
10	01	11	**10/1**	**10/0**

	ab 00	01	11	10
00	$\mu_0\mu_0$	$\mu_0\mu_0$	$\varepsilon\varepsilon$	$\mu_0\varepsilon$
01	$\mu_0\mu_1$	$\mu_0\delta$	$\varepsilon\delta$	$\mu_0\mu_1$
11	$\delta\delta$	$\mu_1\mu_1$	$\mu_1\mu_1$	$\mu_1\delta$
10	$\delta\varepsilon$	$\mu_1\varepsilon$	$\mu_1\mu_0$	$\mu_1\mu_0$

Exemple complet

3. Séparation de la table de l'enclenchement en tables d'enclenchement pour chaque organe de mémoire dans le système:

	ab 00	01	11	10
00	$\mu_0\mu_0$	$\mu_0\mu_0$	$\varepsilon\varepsilon$	$\mu_0\varepsilon$
01	$\mu_0\mu_1$	$\mu_0\delta$	$\varepsilon\delta$	$\mu_0\mu_1$
11	$\delta\delta$	$\mu_1\mu_1$	$\mu_1\mu_1$	$\mu_1\delta$
10	$\delta\varepsilon$	$\mu_1\varepsilon$	$\mu_1\mu_0$	$\mu_1\mu_0$

M_1	ab 00	01	11	10
00	μ_0	μ_0	ε	μ_0
01	μ_0	μ_0	ε	μ_0
11	δ	μ_1	μ_1	μ_1
10	δ	μ_1	μ_1	μ_1

M_2	ab 00	01	11	10
00	μ_0	μ_0	ε	ε
01	μ_1	δ	δ	μ_1
11	δ	μ_1	μ_1	δ
10	ε	ε	μ_0	μ_0

Exemple complet

4. K-Maps des fonctions d'excitation pour des organes de mémoire choisi (JK en l'occurrence)

M_1 \ ab	00	01	11	10
00	μ_0	μ_0	ε	μ_0
01	μ_0	μ_0	ε	μ_0
11	δ	μ_1	μ_1	μ_1
10	δ	μ_1	μ_1	μ_1

	J	K
μ_0	0	–
ε	1	–
δ	–	1
μ_1	–	0

M_2 \ ab	00	01	11	10
00	μ_0	μ_0	ε	ε
01	μ_1	δ	δ	μ_1
11	δ	μ_1	μ_1	δ
10	ε	ε	μ_0	μ_0

J_1K_1 \ ab	00	01	11	10
00	0–	0–	1–	0–
01	0–	0–	1–	0–
11	–1	0	0	0
10	–1	0	0	0

J_2K_2 \ ab	00	01	11	10
00	0–	0–	1–	1–
01	0	–1	–1	0
11	–1	0	0	–1
10	1–	1–	0–	0–

Exemple complet

5. K-Maps des fonctions d'excitation individuelles

J_1K_1 \ ab	00	01	11	10
00	0–	0–	1–	0–
01	0–	0–	1–	0–
11	–1	0	0	0
10	–1	0	0	0

J_2K_2 \ ab	00	01	11	10
00	0–	0–	1–	1–
01	0	–1	–1	0
11	–1	0	0	–1
10	1–	1–	0–	0–

J_1 \ ab	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	–	–	–	–
10	–	–	–	–

K_1 \ ab	00	01	11	10
00	–	–	–	–
01	–	–	–	–
11	1	0	0	0
10	1	0	0	0

J_2 \ ab	00	01	11	10
00	0	0	1	1
01	–	–	–	–
11	–	–	–	–
10	1	1	0	0

K_2 \ ab	00	01	11	10
00	–	–	–	–
01	0	1	1	0
11	1	0	0	1
10	–	–	–	–

$J_1 = ab$ $K_1 = a'b'$ $J_2 = y_1a'+y_1'a$ $K_2 = y_1b'+y_1'b$

Exemple complet

6. Fonction de sortie (la même comme pour la réalisation asynchrone)

	ab 00	01	11	10
00	**1/1**	**1/0**	3	2
01	**2/0**	1	4	**2/1**
11	1	**3/1**	**3/0**	4
10	2	3	**4/1**	**4/0**

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

$$Z = b' y'_1 y'_2 + a\, y_1 'y_2$$

$$a'by_1 + by_1y_2'$$

Exemple complet

Circuit complet
Logigramme

$J_1 = ab$

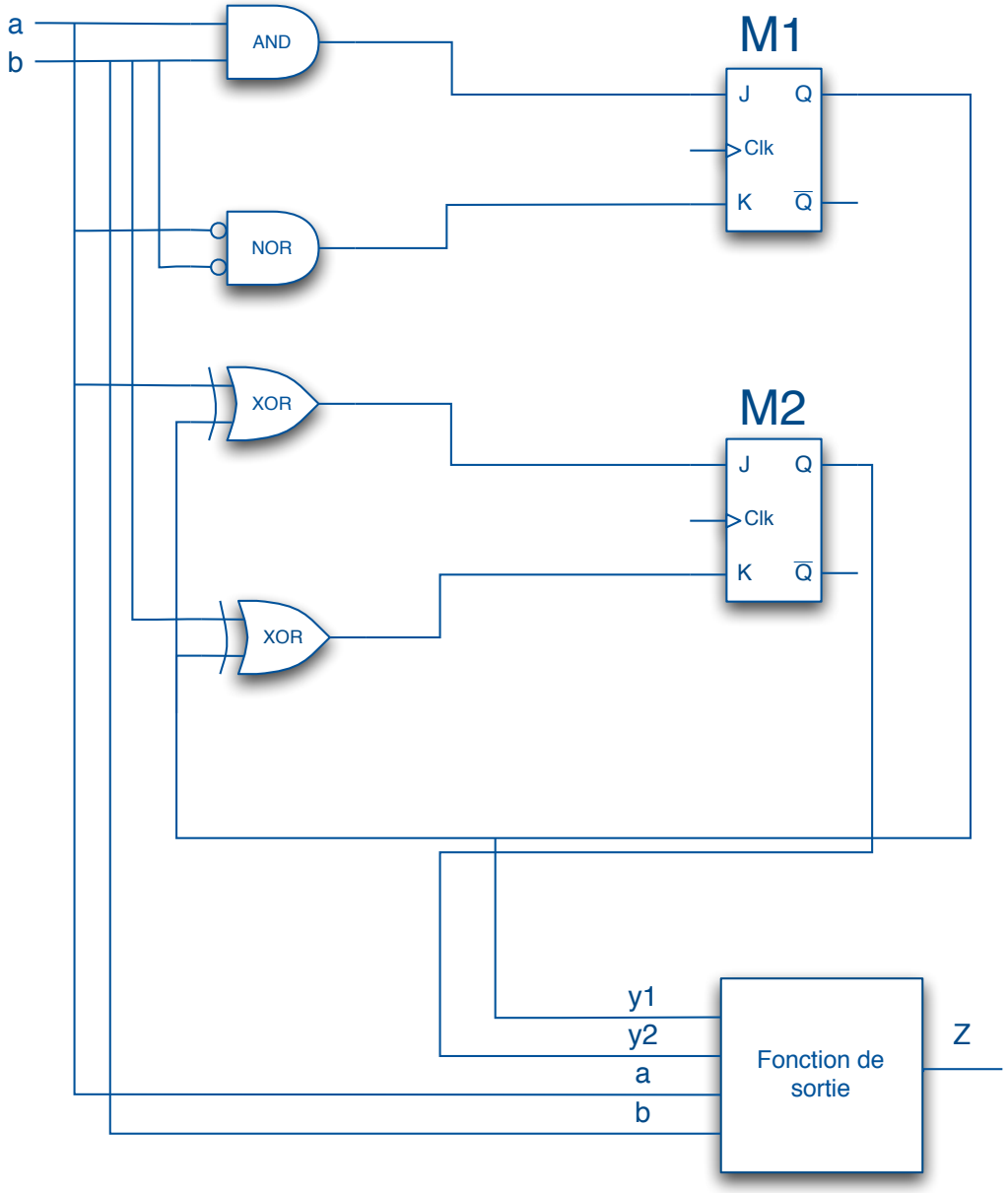
$K_1 = a'b'$

$J_2 = y_1a'+y_1'a$

$K_2 = y_1b'+y_1'b$

$Z = b'\, y_1y_2 + a\, y_1'y_2$

$+ \;\; a'by_1 + by_1y_2'$



Exemple complet

Variante de ce même automatisme avec un D flip-flop

M_1 \ ab	00	01	11	10
00	μ_0	μ_0	ε	μ_0
01	μ_0	μ_0	ε	μ_0
11	δ	μ_1	μ_1	μ_1
10	δ	μ_1	μ_1	μ_1

	D
μ_0	0
ε	1
δ	0
μ_1	1

M_2 \ ab	00	01	11	10
00	μ_0	μ_0	ε	ε
01	μ_1	δ	δ	μ_1
11	δ	μ_1	μ_1	δ
10	ε	ε	μ_0	μ_0

D_1 \ ab	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	1	1	1
10	0	1	1	1

$D_1 = ab+y_1(b+a)$

D_2 \ ab	00	01	11	10
00	0	0	1	1
01	1	0	0	1
11	0	1	1	0
10	1	1	0	0

$D_2 = y_1y_2'a'+ y_1y_2b+ y_1'y_2b'+y_1'y_2'a$

Exemple complet

Fonction de sortie (la même comme pour la réalisation asynchrone)

	ab 00	01	11	10
00	**1/1**	**1/0**	3	2
01	**2/0**	1	4	**2/1**
11	1	**3/1**	**3/0**	4
10	2	3	**4/1**	**4/0**

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

$$Z = b' y'_1 y'_2 + a\, y_1 'y_2 + a'b y_1 + b y_1 y_2'$$

Exemple complet

Comparaison des réalisations avec les flip-flops JK et D

JK

$J_1 = ab$

$K_1 = a'b'$

$J_2 = y_1a'+y_1'a$

$K_2 = y_1b'+y_1'b$

$Z = b' y'_1y'_2 + a y_1'y_2 + a'by_1 + by_1y_2'$

D

$D_1 = ab+y_1(b+a)$

$D_2 = y_1y_2'a'+ y_1y_2b + y_1'y_2b'+ y_1'y_2'a$

$Z = b'y'_1y'_2 + ay_1'y_2 + a'by_1 + by_1y_2'$

Fautes graves

- Ne pas savoir dériver (ou dériver mal) les tables d'excitation pour les organes de mémoire
- Résoudre les courses critiques avant de dériver la table d'excitation de l'automate
- Etablir la table d'excitation par rapport aux entrées! et pas par rapport aux variables d'état → exemple :

	ab			
	00	01	11	10
00	**00/1**	**00/0**	11	01
01	**01/0**	00	10	**01/1**
11	00	**11/1**	**11/0**	10
10	01	11	**10/1**	**10/0**

	ab			
	00	01	11	10
00	$\mu_0\mu_0$	$\mu_0\varepsilon$	$\varepsilon\varepsilon$	$\varepsilon\mu_0$
01				
11				
10				