ELEC-H-305

Circuits logiques et numériques

2011-2012

Cours 9

Dragomir Milojevic

dmilojev@ulb.ac.be

Systèmes séquentiels

Rappel — Synthèse des systèmes séquentiels

1. Systèmes séquentiels: notion de l'état (machine à état, dans littérature anglo-saxonne: *Finite State Machine* - FSM)
2. Représentation:
   - **Table d'état primitive —** un état stable par ligne de la table d'état,
   - **Graphe d'état —** représentation graphique, pas pour la synthèse

   Passage d'un cahier de charges verbal vers une table d'état.
3. Réduction de nombre d'états à l'aide de **Tableau de condition d'équivalences** (simplification de la table d'état : les équivalences et les fusionnements). Passage de la table d'états primitive à la table d'état réduite.

Systèmes séquentiels

Rappel — Synthèse des systèmes séquentiels

4. Codage des états (à chaque état on attribue un code binaire distinct) afin de permettre une réalisation physique à l'aide des portes logiques.
5. Après codage : la **Table d'état** devient la **Table d'état codée**.
6. A partir de la **Table d'état codée** on dérive les K-Maps pour chaque fonction de rétroaction (variable d'état). Pour un système à N états il y a $\log_2 N$ **Fonctions de rétroaction**.
7. K-Map pour la sortie. Résolution des transitions afin d'éviter les *glitchs* à la sortie du système (il ne peut y avoir qu'un seul changement de la valeur de sortie lors des transitions). Fonction logique de la sortie.

Systèmes séquentiels

Aujourd'hui

- Problèmes des **courses critiques** dans un système séquentiel (problème fondamental des circuits logiques à rétroaction)
- Résolution du problème des courses critiques en vue de synthèse (en général):
  I. Action sur le ***codage des états*** et les transitions (passage entre deux états stables):
  Conception des systèmes séquentiels asynchrones
  II. Action sur la ***mise à jour*** des variables d'état :
  Conception des systèmes séquentiels synchrones
- Ces deux façons de résoudre les courses critiques impliquent deux classes des circuits logiques.

Circuits asynchrones

Problème des courses critiques — Source du problème

Prenons l'exemple de la **Table d'état réduite** suivante:

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

Considérons le codage des états suivants (rappel : jusqu'à présent le choix de codage était arbitraire):

1→00, 2→01, 3→11, 4→10

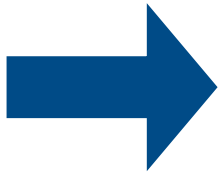
Circuits asynchrones

Problème des courses critiques — Source du problème

On obtient alors la **Table d'état codée** suivante:
(avec 1→00, 2→01, 3→11, 4→10)

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

Table d'état réduite



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

y_1y_2

Table d'état (réduite) codée

Circuits asynchrones

Problème des courses critiques — Source du problème

On en déduit les deux K-Maps et les fonctions de rétroactions:

Y_1Y_2 \ ab	00	01	11	10
0	00	01	11	01
1	00	01	10	01
11	00	11	11	10
10	00	11	10	10

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_1y_2

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

y_1y_2

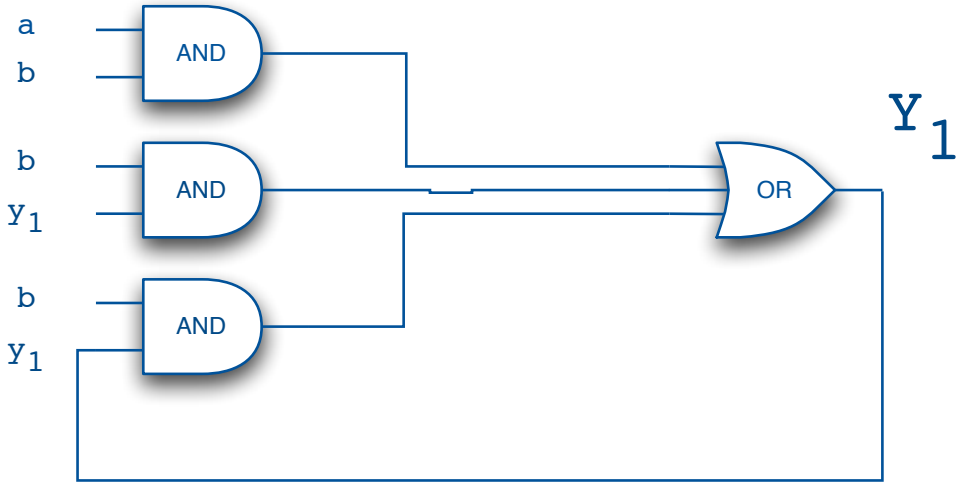
$Y_1 = ab + y_1b + y_1a$

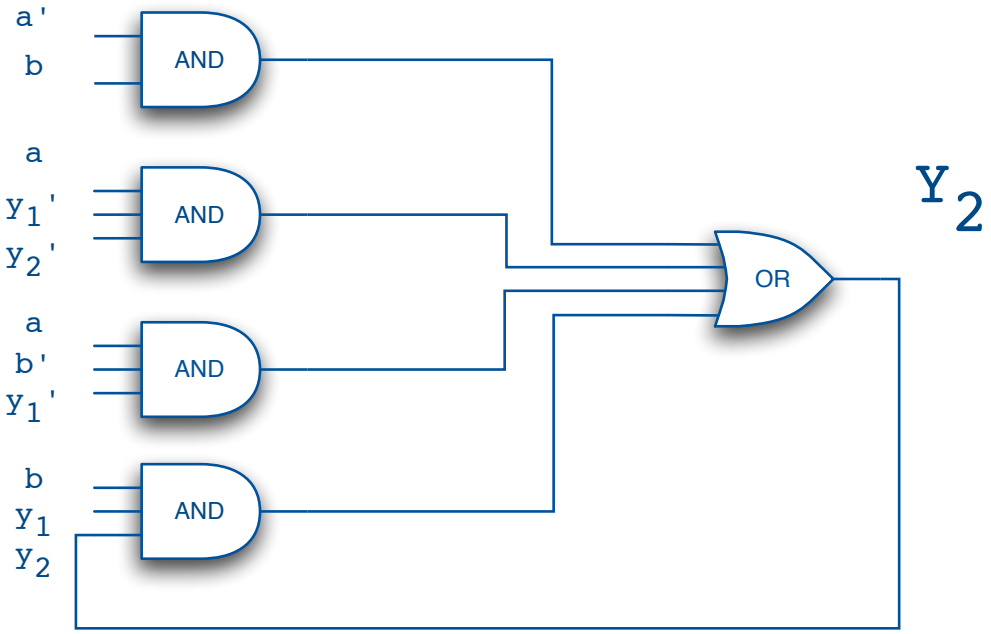
$Y_2 = a'b + y_1'y_2'a + ab'y_1' + y_1y_2b$

Circuits asynchrones

Problème des courses critiques — Source du problème

Ce qui correspond aux logigrammes suivants : les **rétroactions expriment la mise à jour des variables d'état** (les états présent deviennent les états futurs: affectation $y_i \leftarrow Y_i$).





$Y_1 = ab + y_1 b + y_1 a$

$Y_2 = a'b + y_1' y_2' a + ab' y_1' + y_1 y_2 b$

Circuits asynchrones

Problème des courses critiques — Source du problème

Imaginons le passage entre deux états stables suivant:

- Le système se trouve à l'état `1` (l'entrée vaut `ab=00`)
- L'entrée change de `00` à `11`...
- Donc le système va passer de l'état `1` à l'état `3` (la transition indiquée nous envoie en `3`).

... dans la **Table d'état réduite**.

ab

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

Table d'état réduite

Circuits asynchrones

Problème des courses critiques — Source du problème

Observons la même transition dans la **Table d'états réduite codée**:

Le système se trouve en état codée 00 (les entrées ab=00).

Les entrées changent (ab=11) la nouvelle destination pour le système est l'état 11.

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

y_1y_2

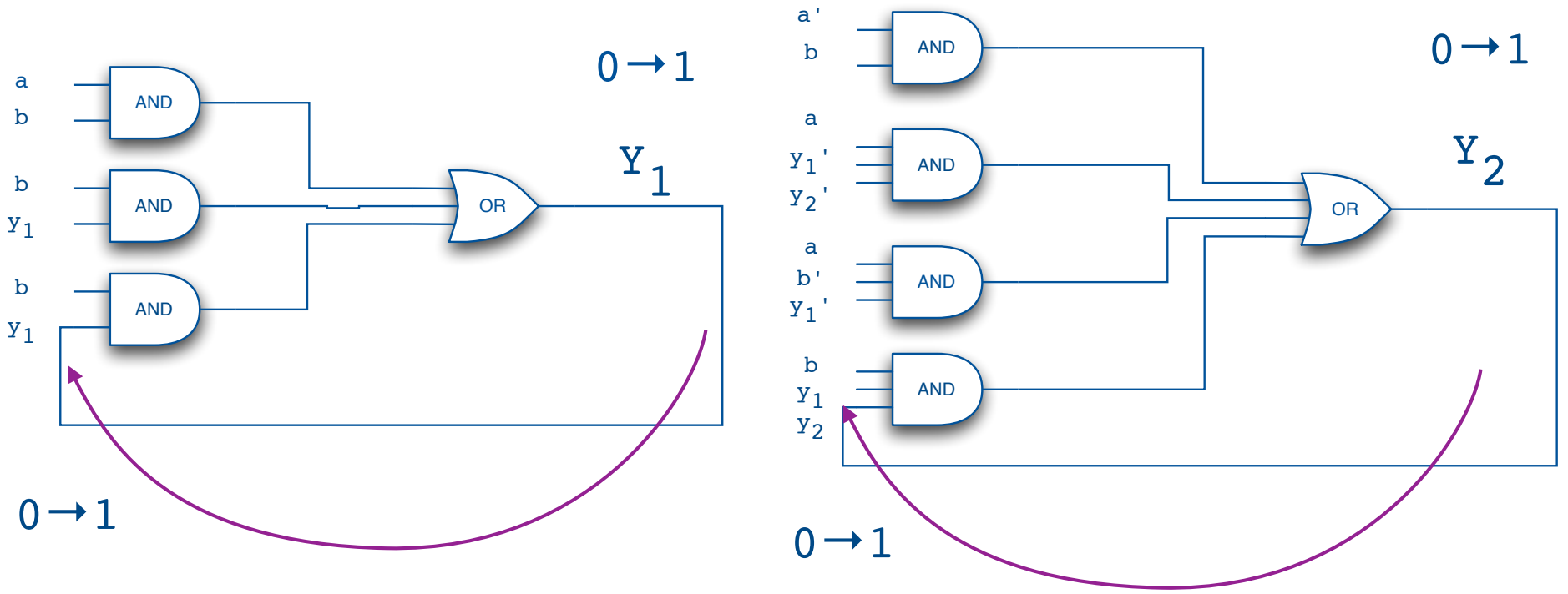
Table d'état codée

Les variables d'état passent de 00 à 11...

Circuits asynchrones

Problème des courses critiques — Source du problème

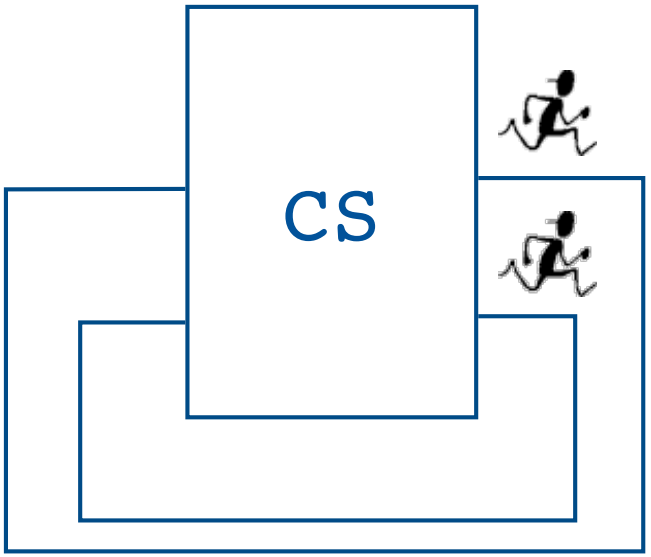
Et finalement observons ce qui se passe dans le circuit logique correspondant au système:
Lors de la mise à jour $(y_i \leftarrow Y_i)$, les deux variables d'état passent de 0 à 1.



Circuits asynchrones

Problème des courses critiques — Source du problème

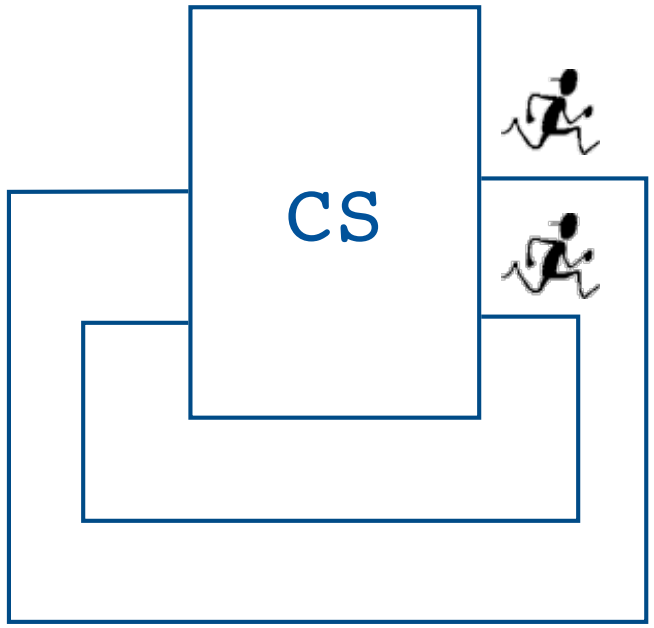
- Imaginons le circuit précèdent comme une boite noire
- C'est un circuit logique à deux variables d'état (deux rétroactions).
- Analogie avec une piste de course et deux coureurs:



Circuits asynchrones

Problème des courses critiques — Source du problème

Le changement à l'entrée provoque la transition du système (entre deux états stables), qui dans ce cas précis, implique le changement **simultané** des **deux variables d'état** dans le circuit (les deux coureurs devrait partir en même temps).

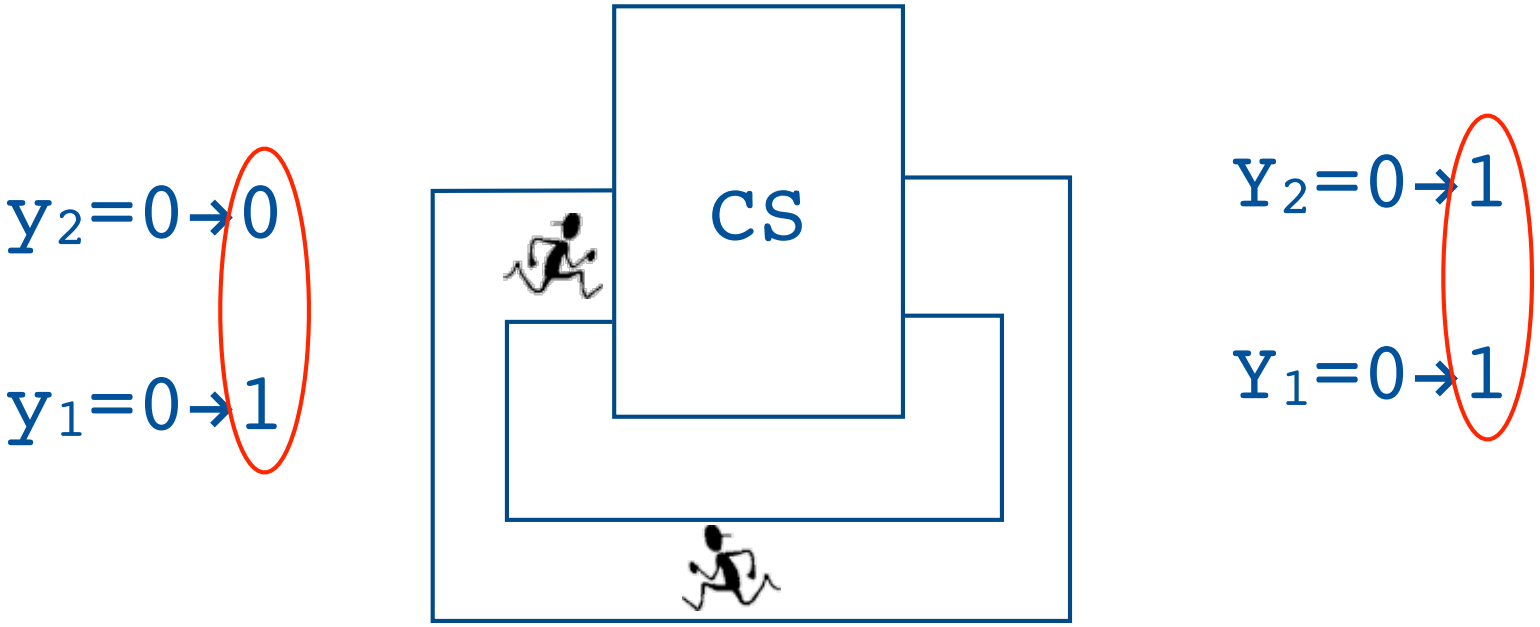


$Y_2=0\rightarrow1$

$Y_1=0\rightarrow1$

Circuits asynchrones

Problème des courses critiques — Source du problème



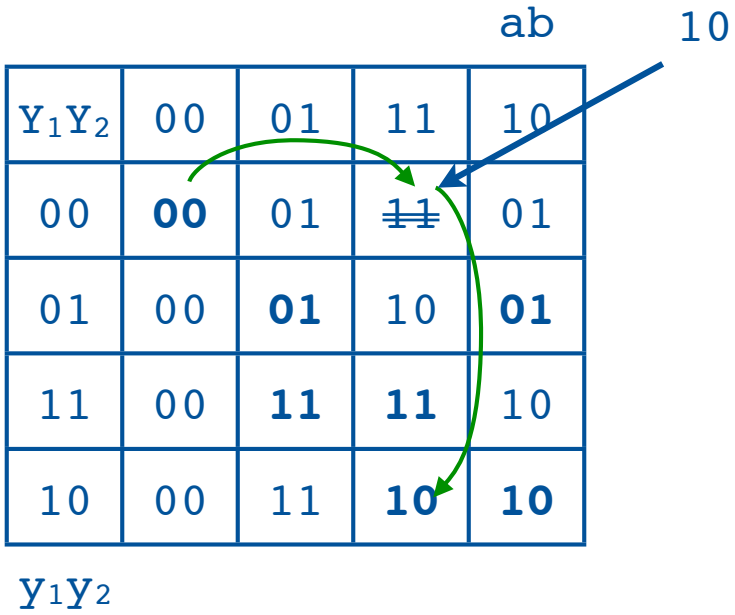
Les deux pistes ne sont pas de la même longueur, la vitesse de deux coureurs n'est jamais la même ... implique que l'un des deux coureurs va arriver **avant** l'autre.

Si au lieu de `11`, la nouvelle destination devient `10` (ou `01`), le système ne respecte plus le cahier de charges initial.

Problème des courses critiques — Source du problème

Le système va donc considérer comme nouvel état de destination `10` au lieu de `11`.

Comme l'état `10` est un état stable, le système va y rester !
(Dans notre cas la transition vers `01` aura le même effet).



Circuits asynchrones

Problème des courses critiques — Table d'état

Initialement la spécification dit:

> *Lorsque le système en état* `00` *et lorsque les entrées* `11`*, le nouvel état devrait être* `11`.

Or à cause des courses :
le système va se retrouver dans l'état `10` !

Le cahier de charges n'est pas respecté.

Le système présente un comportement non voulu ... (et non-déterministe)

Circuits asynchrones

Problème des courses critiques

Def. *Changement instantané de deux (ou plus) variables d'états.*

Dans un système séquentiel il est **impératif** de ne pas avoir des courses critiques afin de garantir le fonctionnement correct de l'automate !!!

Résolution de problème des courses critiques

a) Soit on résout le problème des courses critiques (en les éliminant) → **Circuits asynchrones**

b) Soit on utilise la synchronisation pour mettre à jour le "futur" et donc attendre les variables d'état les plus lentes (il n'est alors plus nécessaire de résoudre les courses) → **Circuits synchrones**

Circuits asynchrones

Problème des courses critiques

Attention !!!

- Ne pas confondre les variables d'états et les entrées : le changement des variables d'état n'a rien avoir avec le changement des entrées (ceci est souvent mal compris).
- On considère que le changement simultané des variables à l'entrée est possible.
- Dans le cahier de charges on spécifie le comportement de l'automate lorsque on a des variations simultanées à l'entrée. On peut aussi dire que les variations simultanées sont impossibles (exemple du réservoir).

Circuits asynchrones

Problème des courses critiques — Exemple

Changement instantané de deux (ou plus) de variables d'états.

Donc, dans la table suivante, toutes les transitions marquées en rouge représentent des courses critiques et doivent être résolues.

ab

Y_1Y_2	00	01	11	10
00	**00**		11	
01		**01**		10
11	00		**11**	
10		01		**10**

y_1y_2

Circuits asynchrones

Problème des courses critiques

Changement de plus de 2 variables à l'entrée est possible, mais pas le changement des variables d'état.

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

y_1y_2

Ceci n'est pas une course !

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

y_1y_2

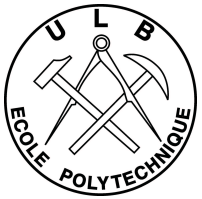
Ceci est une course !

Circuits asynchrones

Problème des courses critiques

- Résolution du problème des courses critiques par manipulation des transitions? on dispose de trois méthodes différentes: **Méthode 1, Méthode 2, Méthode 3**.
- Afin de rendre le circuit final le plus optimal possible, on appliquera les différentes méthodes l'une après l'autre:
  - **Méthode 2** seulement si **Méthode 1** ne donne pas de résultats et
  - **Méthode 3** seulement si les **Méthodes 1** et **2** ne donnent pas de résultats).
- Résolution des courses : uniquement lorsque l'on demande la synthèse des systèmes **asynchrones !**

Circuits asynchrones



Problème des courses critiques

Méthode 1 (M1) — Le codage des états

- Le choix des codes pour les états est arbitraire. Nous pouvons donc tenter de choisir un codage qui ne présente pas de problème de courses critiques.
- *Ceci n'est pas toujours possible.*
- **Méthode graphique** permettant de trouver le codage ne présentant le problème des courses. Basée sur le **Graphe de codage des états** (on peut montrer la non-existence d'un codage sans courses critiques).

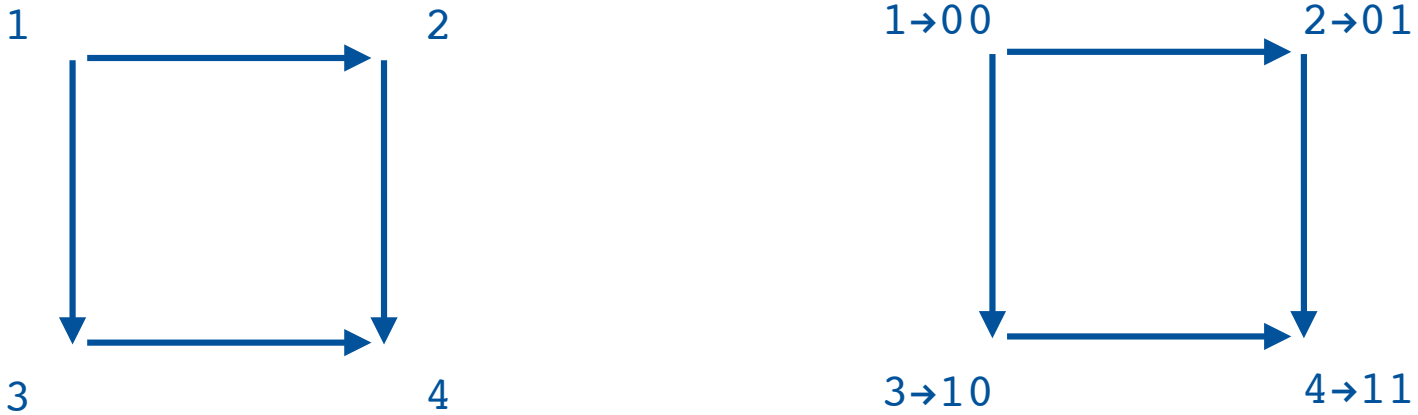
Circuits asynchrones

Problème des courses critiques

Graphe de codage des états

Imaginons la représentation suivante de la table d'états (similaire au Graphe d'état):

- Un cube, chaque sommet est un état stable, transitions représentées par des arc orientés.
- On note les états futurs possibles pour chaque état.
- On attribue les codes de façon à ne pas avoir des courses critiques (les **diagonales** représentent le changement simultané des deux variables).

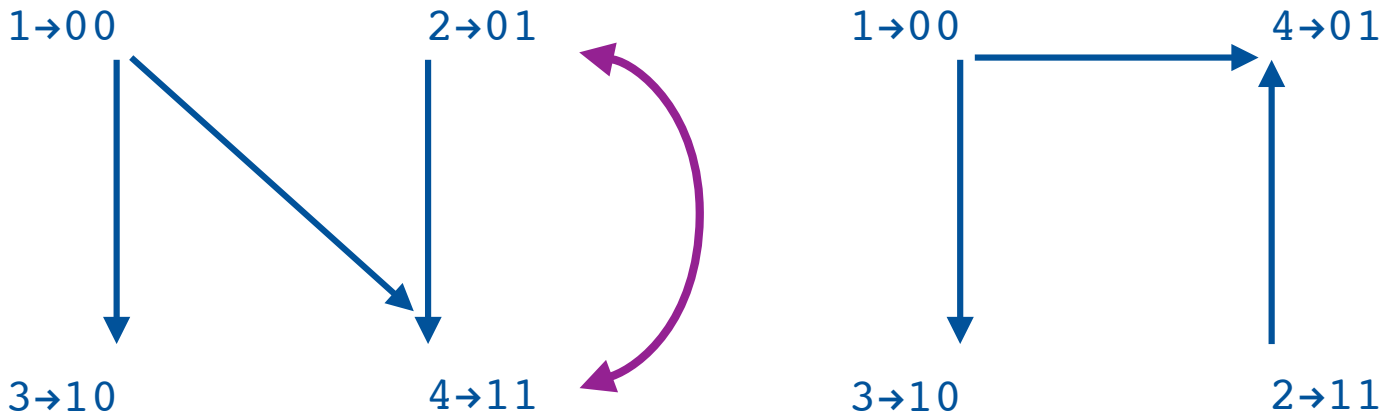


Circuits asynchrones

Problème des courses critiques

Graphe de codage des états

Imaginons la **Graphe de codage des états** suivant :



Une course critique : 1→4

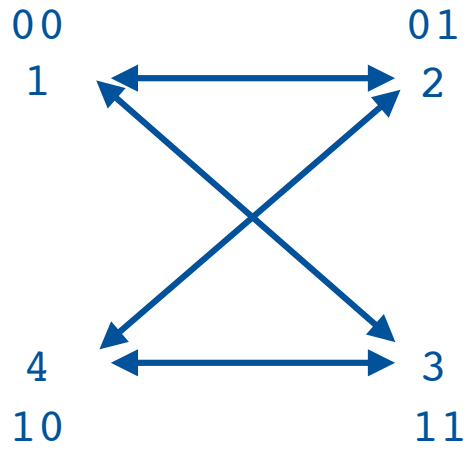
On peut inverser les codes pour 2 et 4 et ainsi éviter le changement simultané de deux variables !

Circuits asynchrones

Problème des courses critiques — Exemple M1

Pour la table d'état suivante et le codage :

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



y_1y_2	ab 00	01	11	10
00	**00**	01	11	01
01	**01**	**01**	10	**01**
11	00	**11**	**11**	10
10	01	11	**10**	**10**

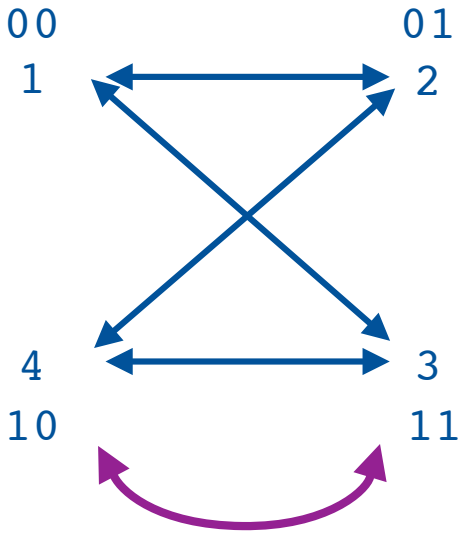
4 courses critiques!

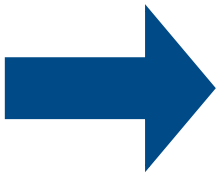
Circuits asynchrones

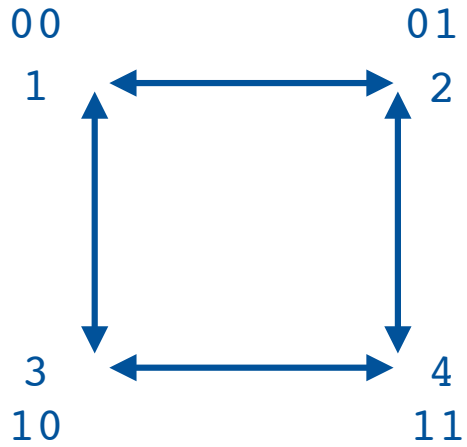
Problème des courses critiques — Exemple M1

On inverse les codes pour 3 et 4 :

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





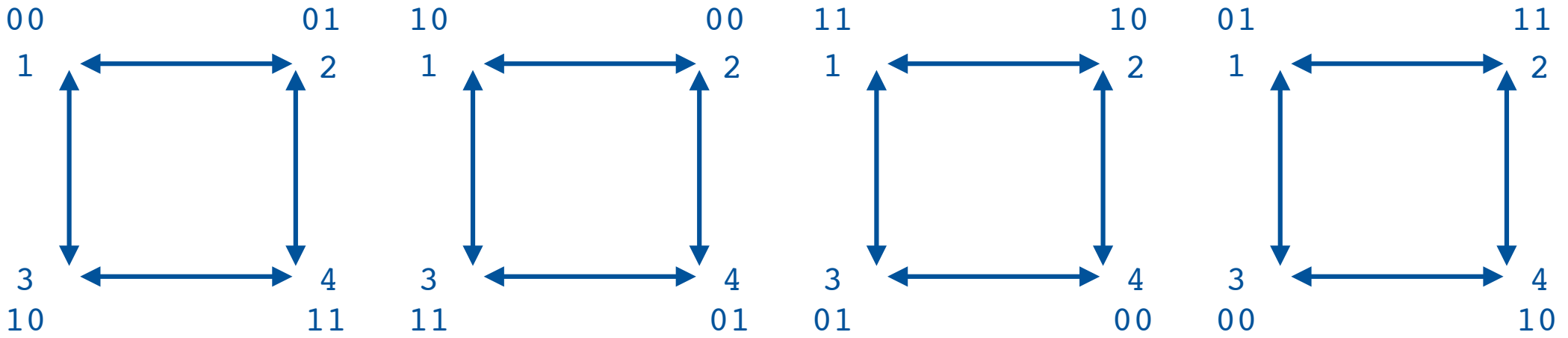


Circuits asynchrones

Problème des courses critiques — Exemple M1

Cette solution n'est pas unique. Pour la table d'état suivante, on a aussi :

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

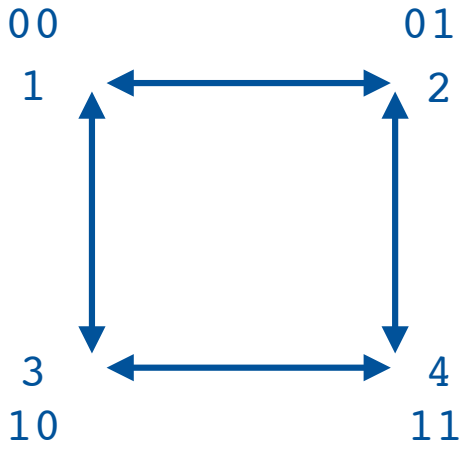


Circuits asynchrones

Problème des courses critiques — Exemple M1

En prenant la table d'état et le codage trouvé on a :

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



La table d'état codée est sans courses critiques !

Mais ceci n'est pas une K-Map

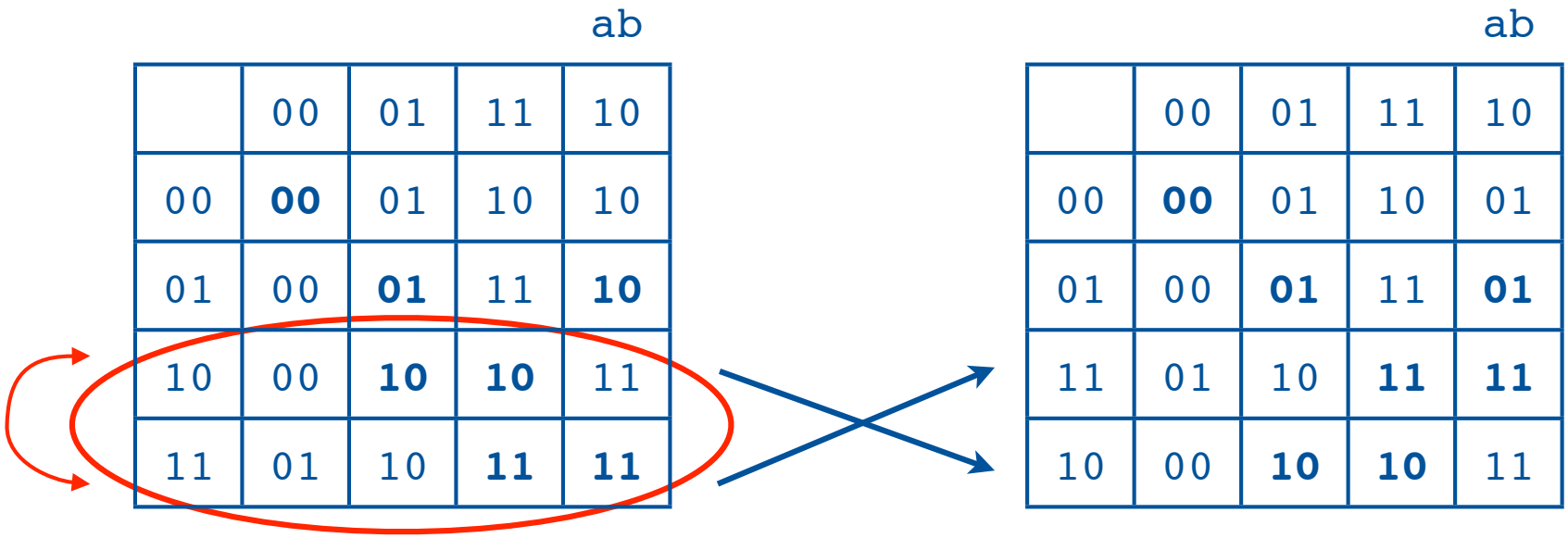
	ab 00	01	11	10
00	**00**	01	10	01
01	00	**01**	11	**01**
10	00	**10**	**10**	11
11	01	10	**11**	**11**

y_1y_2

Circuits asynchrones

Problème des courses critiques — Exemple M1

Attention à la K-Map !!! Le codage des états implique que l'on doit inverser les deux dernières lignes pour obtenir une K-Map pour dériver les équations (correctes) de rétroaction (en respectant la distance d'*Hamming*).



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

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

Circuits asynchrones

Problème des courses critiques

- Résolution du problème des courses critiques permet la réalisation physique d'un système séquentiel **(asynchrone)** correct (son implémentation physique sera opérationnelle).
- Vu les possibilités de codage : plusieurs solutions (correctes!!!) sont possibles pour un circuit **asynchrone**.
- Le choix de codage peut influencer la complexité du circuit résultant, mais vous ne devriez pas en tenir compte (tout codage sans courses sera considéré comme bon).

Circuits asynchrones

Problème des courses critiques — Exemple

Il est maintenant possible de dériver les expressions pour les fonctions de rétroaction en divisant la **Table d'état codée** en 2 **K-Maps**.

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

$Y_1 = ab + y_1 b + y_1 a$

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

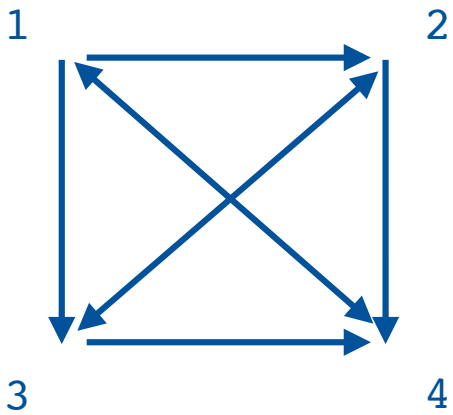
Y_2 \ ab	00	01	11	10
00	0	1	0	1
01	0	1	1	1
11	1	0	1	1
10	0	0	0	1

$Y_2 = ab' + ay_2 + a'b + b'y_1y_2$

Circuits asynchrones

Problème des courses critiques

- Le choix de codage ne peut pas toujours résoudre le problème des courses critiques
- Imaginons le graphe de codage des états suivant :



- Quelque soit le codage choisi, il y aura toujours deux variables d'état qui vont changer en même temps (présence simultanée des diagonales et des cotés)

Circuits asynchrones

Problème des courses critiques

Méthode 2 (M2) — En utilisant (modifiant) les transitions

a) En utilisant les *don't cares*

- Le changement entre deux états stables se fait via les transitions
- Dans une machine d'état, les transitions désignent l'évolution interne de l'automate
- Ainsi, si une case dans la table d'état est libre (un *don't care*) on peut la transformer en une transition et ainsi éviter la course critique !

Circuits asynchrones

Problème des courses critiques — Exemple M2 a)

Dans la table d'état suivante :

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

	ab 00	01	11	10
00	**00**	01	~~11~~ 01	
01		**01**	-- 11	
11			**11**	
10				

Codage 1→00, 2→01, 3→11 présente une course critique

Dans la même colonne des entrées (ab=11), il existe un état *don't care* que l'on peut l'utiliser pour la transition.

Circuits asynchrones

Problème des courses critiques

Méthode 2 (M2) — En utilisant les transitions

b) En modifiant les transitions

Le changement des états se fait via transitions. Une transition dans la table d'état peut être **transformée** pour éviter la course critique. Le chemin emprunté entre deux états stables n'a pas d'importance sur le fonctionnement de l'automate, seul le point d'arrivé compte.

Ainsi, pour passer de **1** à **3** on peut avoir le choix entre:

1→2→3→**3**

=

1→4→3→**3**

Circuits asynchrones

Problème des courses critiques — Exemple M2 b)

Dans la table d'état suivante :

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

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

Codage 1→00, 2→01, 3→11
présente une course critique

Dans la colonne des entrées ab=11 on transforme la transition 11 en 10 afin de permettre le changement d'une variable d'état à la fois.

Circuits asynchrones

Problème des courses critiques

Méthode 3 (M3) Ajout d'une variable d'état supplémentaire

- Lorsque les deux méthodes précédentes ne permettent pas de résoudre le problème des courses critiques, on est obligé d'ajouter une variable d'état en plus
- En ajoutant une variable supplémentaire on dispose de 2^n (n - nombre de variables d'état au départ) transitions en plus. Ces transitions sont ensuite utilisées pour résoudre les courses
- Avec une variable d'état en plus, on ne doit traiter que des courses critiques que l'on ne peut pas résoudre autrement (pour avoir une expression la plus simple possible)

Circuits asynchrones

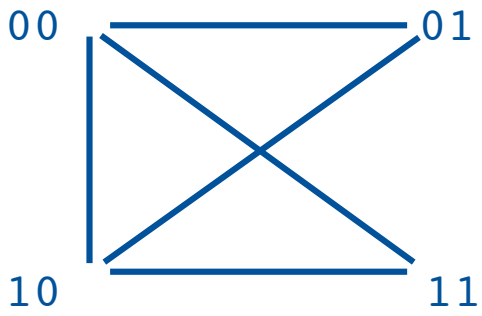
Problème des courses critiques — Exemple M3

Pour la table d'état codé: colonnes `ab=11` et `ab=10` une seule transition!

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

Toutes les autres - états stables!

M1. Pas de codage sans courses:



M2. Colonnes ab=11: une seule transition - on ne peut donc pas modifier les transitions.

Ajout d'une variable en plus !

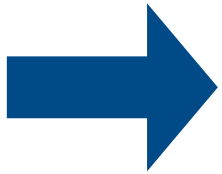
Circuits asynchrones

Problème des courses critiques — Exemple M3

Ajout d'une variable d'état: nouvelle table d'état

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

y_1y_2



$Y_1Y_2Y_3$ \ ab	00	01	11	10
000	**00**	01	11	10
010	00	**01**	**01**	10
110	01	10	**11**	**11**
100	00	**10**	**10**	**10**
001	–	–	–	–
011	–	–	–	–
111	–	–	–	–
101	–	–	–	–

$y_1y_2y_3$

Attention ceci n'est pas une Table d'état correcte. Pourquoi?

Circuits asynchrones

Problème des courses critiques — Exemple M3

ab				
$Y_1Y_2Y_3$	00	01	11	10
000	**000**	010	011	100
010	000	**010**	**010**	100
110	010	100	**110**	**110**
100	000	**100**	**100**	**100**
001	–	–	–	–
011	–	–	–	–
111	–	–	–	–
101	–	–	–	–

$y_1y_2y_3$

ab				
$Y_1Y_2Y_3$	00	01	11	10
000	**000**	010	001	100
010	000	**010**	**010**	011
110	010	100	**110**	**110**
100	000	**100**	**100**	**100**
001	–	–	011	–
011	–	–	111	111
111	–	–	110	101
101	–	–	–	100

$y_1y_2y_3$

Est-ce nécessaire?

Circuits asynchrones

Problème des courses critiques — Exemple M3

Y_1 \ ab	00	01	11	10
000	0	0	0	1
010	0	0	0	0
110	0	1	1	1
100	0	1	1	1
001	–	–	0	–
011	–	–	1	1
111	–	–	1	1
101	–	–	–	1

$y_1y_2y_3$

Y_2 \ ab	00	01	11	10
000	0	1	0	0
010	0	1	1	1
110	1	0	1	1
100	0	0	0	0
001	–	–	1	–
011	–	–	1	1
111	–	–	1	0
101	–	–	–	0

$y_1y_2y_3$

Y_3 \ ab	00	01	11	10
000	0	0	1	0
010	0	0	0	1
110	0	0	0	0
100	0	0	0	0
001	–	–	1	–
011	–	–	1	1
111	–	–	0	1
101	–	–	–	0

$y_1y_2y_3$

Circuits asynchrones

Synthèse des circuits logiques séquentiels asynchrones

Algorithme

1. **Table d'état primitive** à partir du cahier de charges
2. **Réduction** de nombre d'états à l'aide de la table d'équivalence des états
3. **Codage** des états
4. Résolution de problème des **courses critiques** en appliquant les méthodes M1, M2 ou M3 (dans l'ordre!)
5. Dérivation des **fonctions de rétroaction** et **fonction de sortie** → implémentation du circuit

Circuits asynchrones

Fautes graves

1. Laisser une course critique lorsque l'on demande la synthèse d'un circuit asynchrone
2. Ne pas appliquer les méthodes de résolution dans le bon ordre: ajouter une variable d'état en plus, lorsqu'il existe un codage sans courses critiques ou lorsqu'il est possible de s'en sortir avec des transitions
3. Effectuer le codage et dériver une K-Map de fonction de rétroaction qui ne respecte pas les règles de construction d'une K-Map: distance de Hamming > 1 entre les lignes et les colonnes adjacentes