

Organisation du module +
**Chapitre 1: Introduction aux machines
séquentielles**

CHIRAZ TRABELSI

trabelsi@esiea.fr

Et

ALEXANDRE BRIERE

briere@esiea.fr

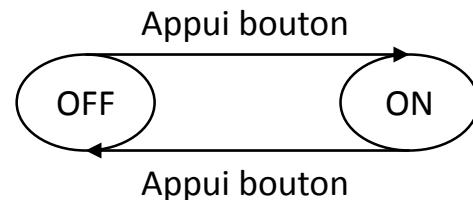
- **Cours ⇒ 1h30 / Cours (18 h au total, soit 12 séances)**
 - Systèmes à Evénements Discrets
 - Machines séquentielles (Machines de Moore et Mealy)
 - Réseaux de Pétri
 - Introduction aux systèmes micro-programmés
 - Les mémoires (les différents types, composition, association de boîtiers mémoires)
 - Interfaçage entre mémoires et micro-processeur (cartographie mémoire, décodage d'adresses)
 - Introduction au microprocesseur (introduction au module SYS3041 en 3A S1)
- **TDs ⇒ 1h30 / TD (9h au total, soit 6 séances)**
 - TD sur papier: Application du cours
- **TPs ⇒ 1h30 / TP (9h au total, soit 6 séances)**
 - TD sur les cartes Altera en fin de semestre

- Machines séquentielles (Machines de Moore et Mealy)
- Réseaux de Pétri
- Objectif:
 - Savoir **analyser un système** ou un cahier des charges afin de le **modéliser** et le **réaliser** en utilisant différentes méthodes (machines séquentielles, réseaux de Pétri)

- **Définitions**

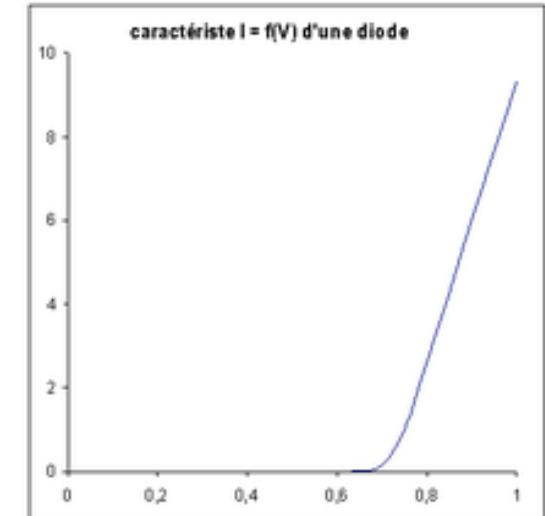
- **Systèmes à événements discrets**

- Systèmes dont l'ensemble d'états possibles est discret (un nombre bien défini d'états)
- Les changements d'état sont déclenchés par des événements ponctuels
 - Exemples d'événements discrets
 - Appui sur un bouton, arrivée d'un client, d'un signal ou l'achèvement d'une tâche



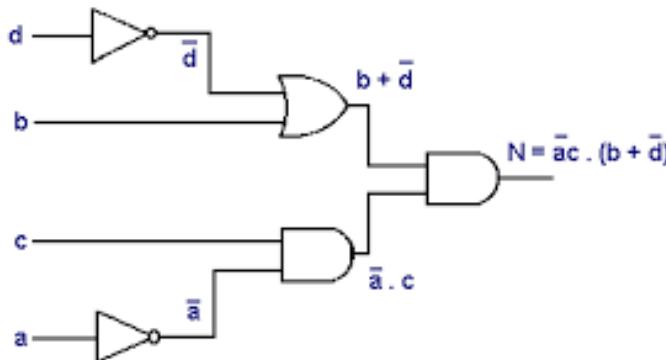
- **Systèmes à variable continue**

- L'espace d'états possibles est continu
- Les variables d'entrée sont des variables continues



Système combinatoire

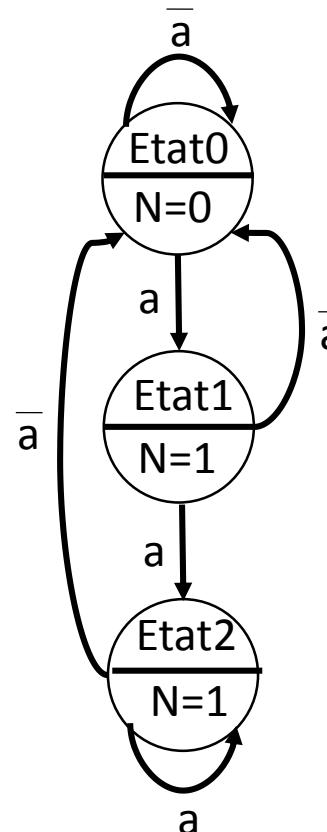
- Entrées: a, b, c,d
- Sortie: N



- La sortie du système à l'instant t ne dépend que des valeurs des entrées à cet instant
- La sortie s'écrit comme une fonction des entrées courantes.

Système séquentiel

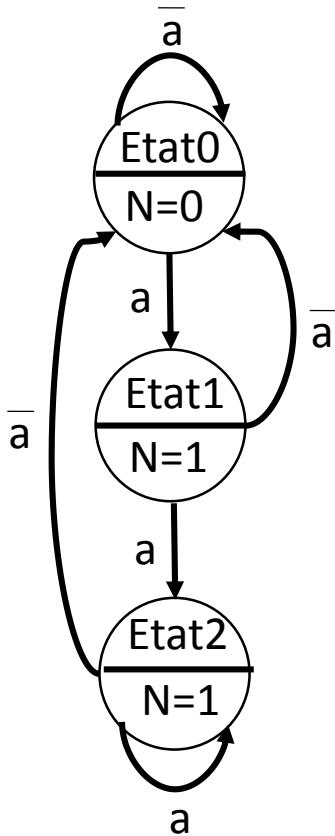
- Entrée: a
- Sortie: N



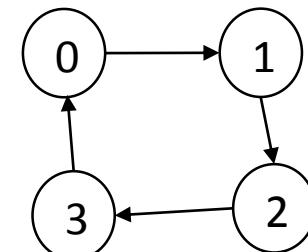
- La sortie du système à l'instant t dépend de la **séquence des entrées dans le temps**
- Le système évolue d'un état à un autre selon les valeurs des entrées
 - ➔ Pour connaître la sortie d'un système séquentiel, il faut connaître son état courant
 - ➔ Il faut sauvegarder l'état du système
 - ➔ Utilisation des **bascules** pour la mémorisation

Les systèmes séquentiels

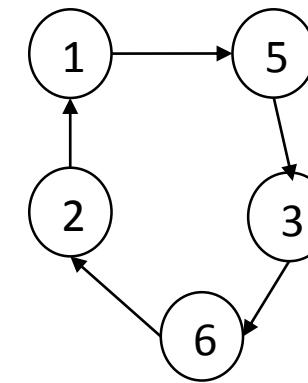
- Les compteurs et les cycles vus en 1^{er} semestre sont un cas particulier des systèmes séquentiels, où il n'y a pas d'entrées et de sorties en dehors de celles des bascules



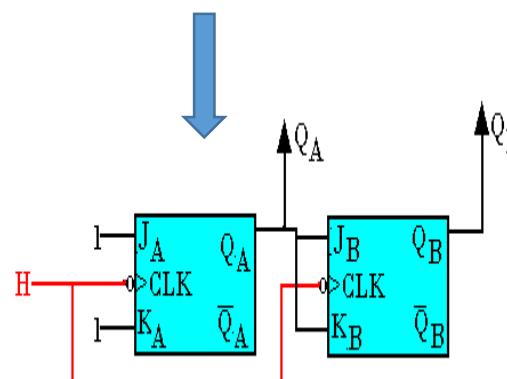
Cas général d'un système séquentiel
(avec des entrées)



Cas particulier: compteur
Compteur modulo 4

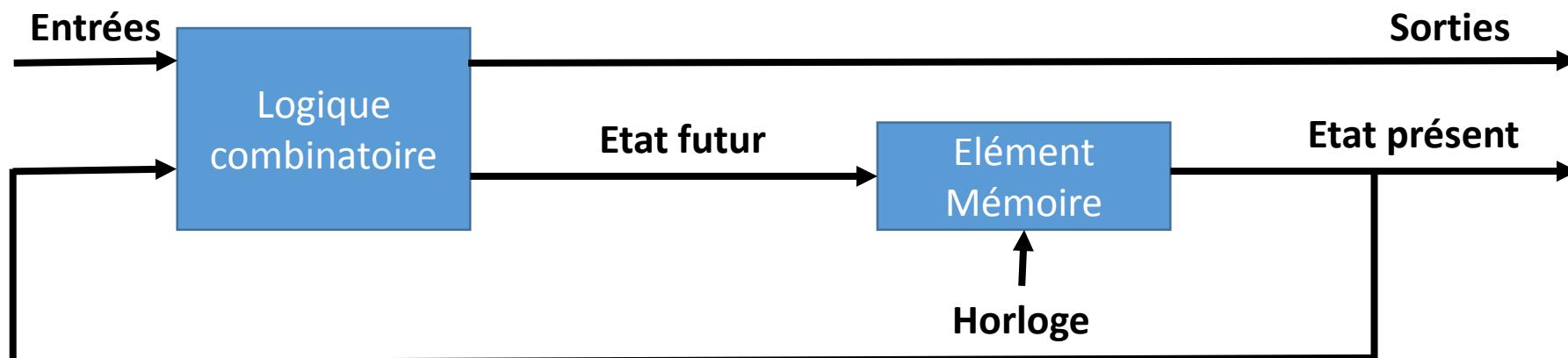


Cas particulier: cycle
Cycle 1-5-3-6-2



Synthèse en bascules JK

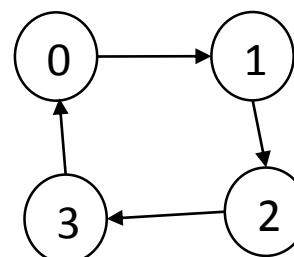
- Deux types de systèmes séquentiels
 - **Asynchrone**: peut changer ses valeurs de sortie et d'état à chaque fois qu'il y a un changement des entrées
 - **Synchrone**: change d'état à des moments fixes, généralement définis par le **front montant ou descendant** d'un signal d'**horloge**



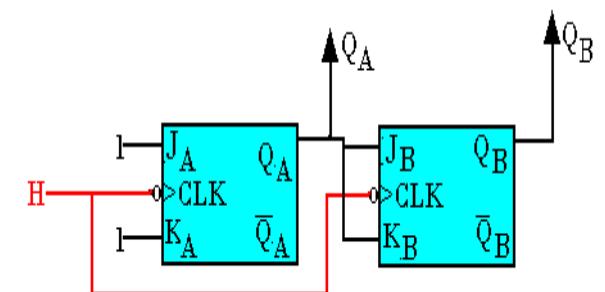
Structure d'un système séquentiel synchrone

- L'état courant est déterminé par les sorties des bascules
- Les bascules permettent de conserver l'état courant (mémorisation) pour un cycle d'horloge et de le mettre à jour au prochain front d'horloge
- Exemple de système synchrone: compteur synchrone modulo 4

- 1^{er} cycle d'horloge
Etat 0: QB=0 QA=0
- 2^{ème} cycle d'horloge
Etat 1: QB=0 QA=1
- 3^{ème} cycle d'horloge
Etat 2: QB=1 QA=0
- 4^{ème} cycle d'horloge
Etat 3: QB=1 QA=1
- 5^{ème} cycle d'horloge
Etat 0: QB=0 QA=0

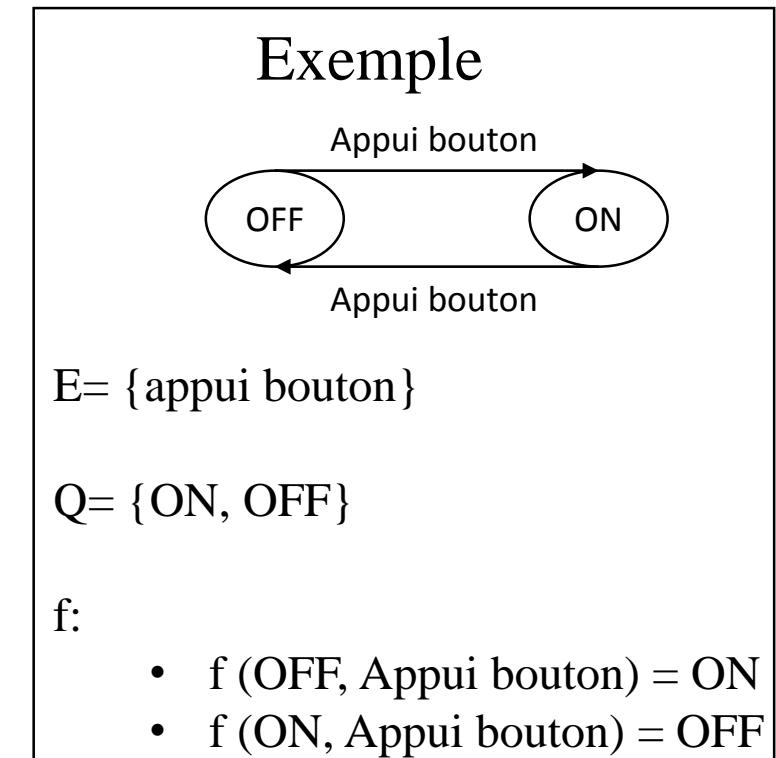


Compteur modulo 4

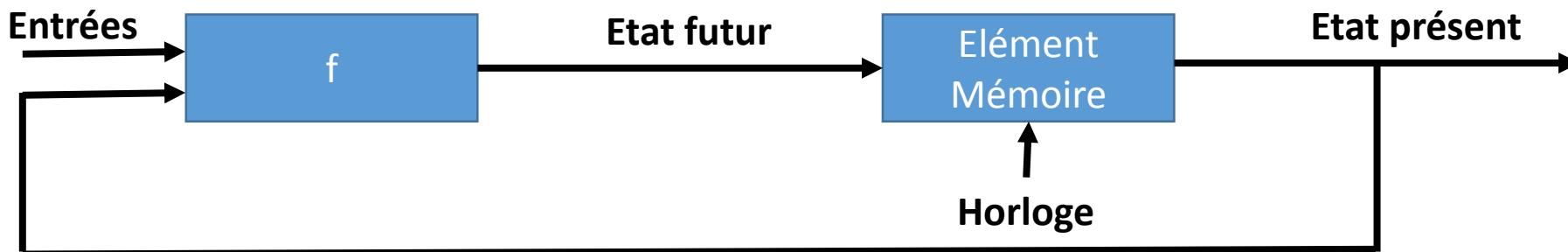


Synthèse en bascules JK

- Un système séquentiel peut être représenté par un automate fini
- Automate : $A = (E, Q, f)$
 - E : ensemble fini, non vide des symboles d'entrée
 - Q : ensemble fini, non vide des états
 - f : fonction état suivant ou état futur
 - $f: Q \times E \rightarrow Q$



- Structure de base d'un automate fini



- État : ensemble d'informations correspondant à des valeurs physiques (caractéristiques du processus à un instant).
- f est une fonction à logique combinatoire
 - état futur = f (état présent, entrées)
- Dans les systèmes séquentiels synchrones le changement d'état est synchronisé avec un signal d'horloge H

- Deux représentations possibles
 - Une table d'états
 - Un graphe d'états

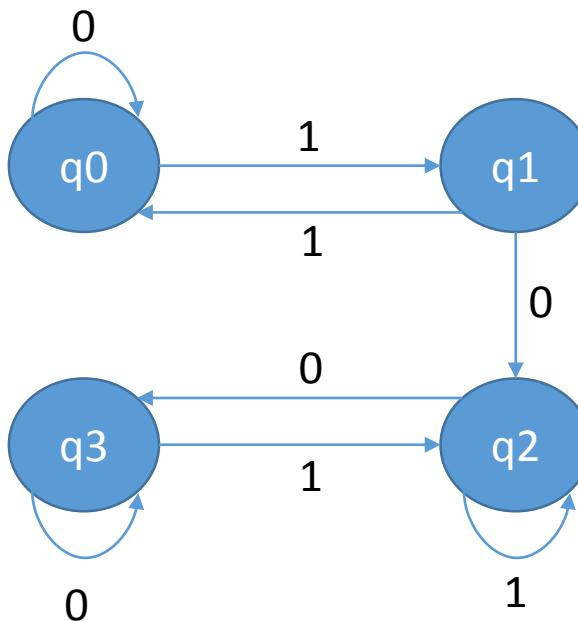
- Représentation tabulaire, où chaque ligne représente un état présent et chaque colonne représente une combinaison des entrées.
 - A chaque case de la table on introduit l'état futur.
-
- Exemple
 - Soit l 'Automate fini A tel que $E=\{e0\}$ et $Q=\{q0,q1,q2,q3\}$

$Q \setminus e0$	0	1	Entrée
q0	q0	q1	
q1	q2	q0	
q2	q3	q2	
q3	q3	q2	

Etat présent

Etat futur

- Représentation graphique où chaque état est représenté par un cercle
- Exemple
 - Soit l 'Automate fini A tel que $E=\{e0\}$ et $Q=\{q0,q1,q2,q3\}$



$Q \setminus e_0$	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_3	q_2
q_3	q_3	q_2

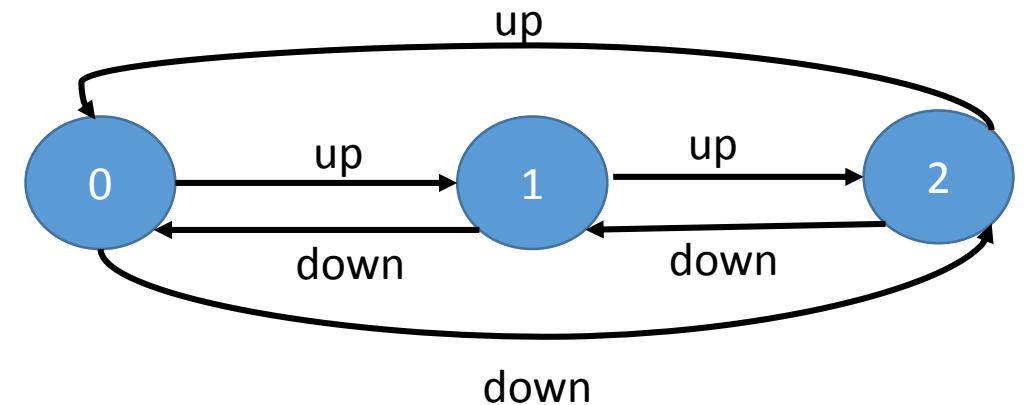
- Soit un COMPTEUR – DECOMPTEUR modulo 3
 - 1 entrée e0 avec deux valeurs possibles: Up et Down
 - Etats: 0, 1 et 2
- 1) Dessiner sa table d'états
 - 2) Dessiner son graphe d'états

- COMPTEUR – DECOMPTEUR modulo 3
 - 1 entrée Up/Down

1) Dessiner sa table d'états

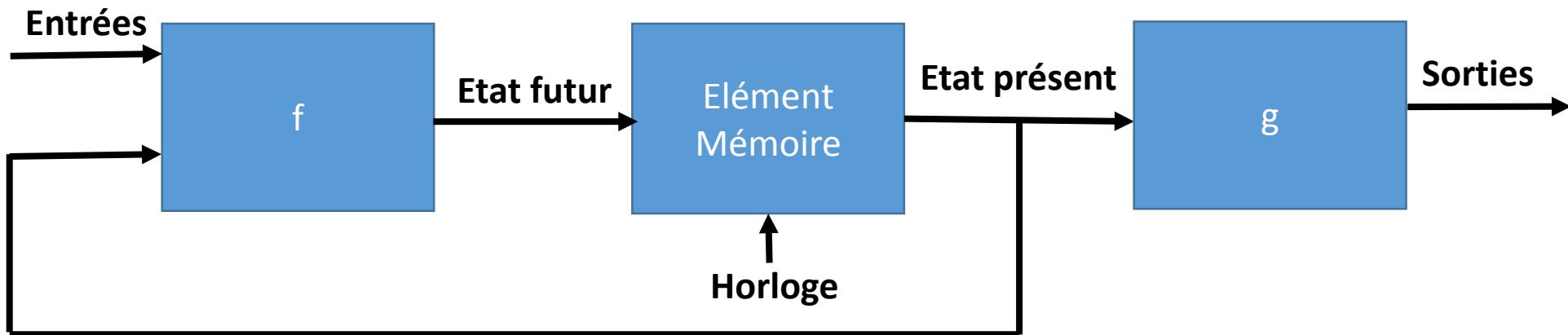
Q \ e0	up	down
0	1	2
1	2	0
2	0	1

2) Dessiner son graphe d'états



- Définition: $M = (A, S, g)$
 - A: automate fini
 - S: ensemble non vide des symboles de sortie
 - g: fonction de sortie
- c'est un automate fini qui présente des sorties
- Il existe 2 types de machines séquentielles: **Moore et Mealy**

- Les variables de sortie dépendent de l'état présent.
- Le changement des sorties est synchronisé par l'horloge.
 - Sortie = g (état présent)

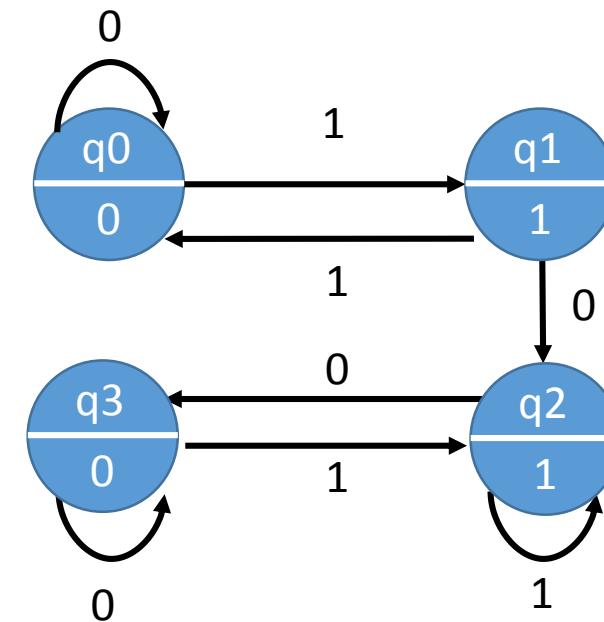


Représentations

Table d'états

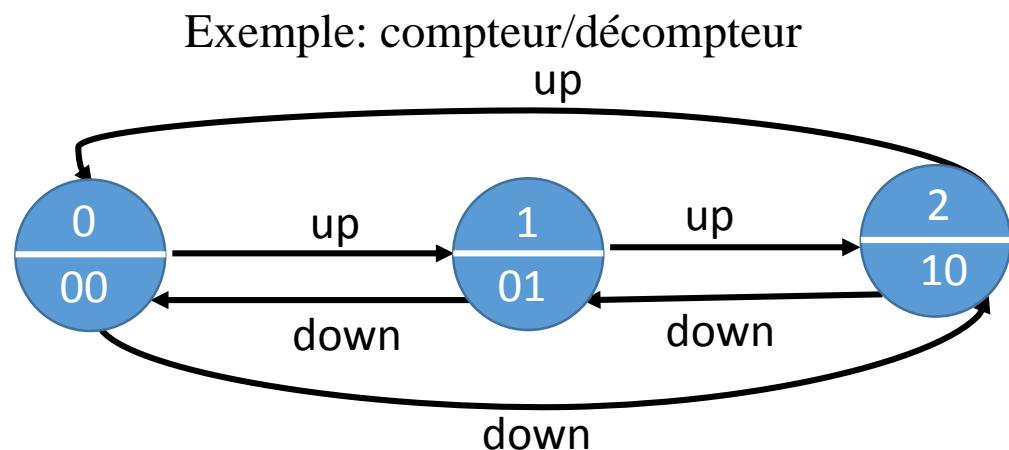
Q \ e0	0	1	S
q0	q0	q1	0
q1	q2	q0	1
q2	q3	q2	1
q3	q3	q2	0

Graphe d'états



Attention: la sortie est une fonction de l'état présent pas de l'état futur/suivant

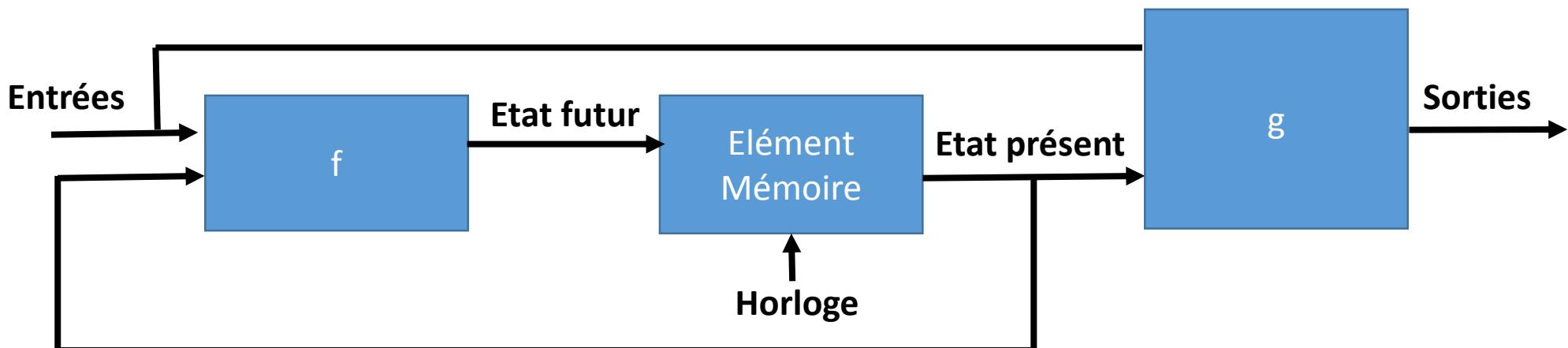
- Les variables de sortie dépendent de l'état présent.
- Le changement des sorties est synchronisé par l'horloge.



- Entrées= 1 bouton avec deux valeurs possibles (up et down)
- Sortie= deux leds qui affichent le nombre courant en binaire
(0 → led éteinte, 1 → led allumée)

Etat	Sorties
0	● ●
1	● ○
2	○ ●

- Les valeurs des variables de sortie dépendent de l 'état présent **et** des variables d 'entrée.
- le changement des sorties n'est pas synchrone, il se fait avec le changement des entrées.
 - Sortie = **g** (état présent, entrées)

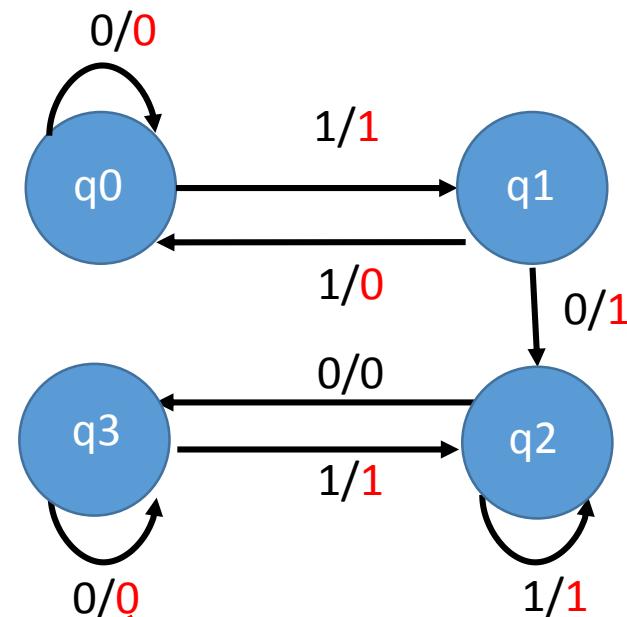


Représentations

Table d'états

$Q \setminus e_0$	0	1
q_0	$q_0/0$	$q_1/1$
q_1	$q_2/1$	$q_0/0$
q_2	$q_3/0$	$q_2/1$
q_3	$q_3/0$	$q_2/1$

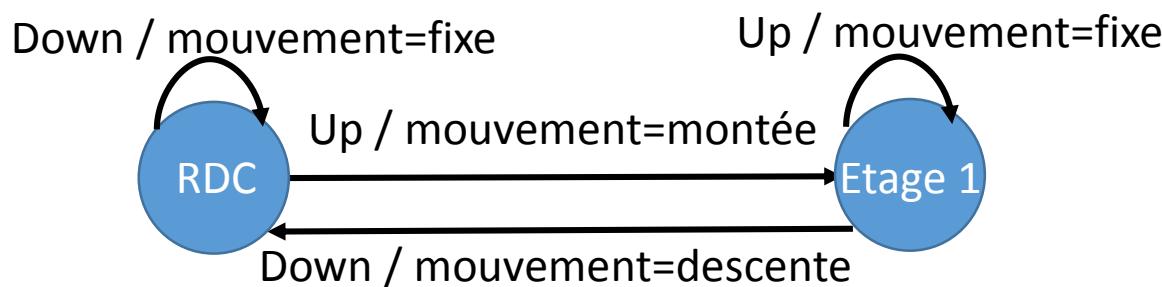
Graphe d'états



sortie

- Les valeurs des variables de sortie dépendent de l état présent **et** des variables d entrée.
- Le changement des sorties n'est pas synchrone, il se fait avec le changement des entrées.

Exemple: Ascenseur

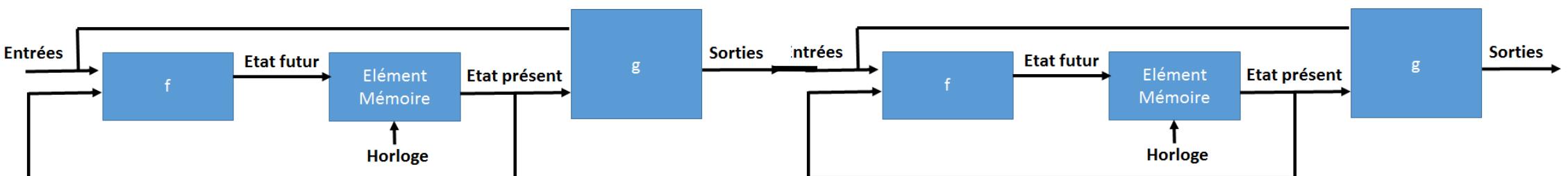


- Entrées= 1 bouton avec deux valeurs possibles (up et down)
- Sortie= le mouvement du mouvement de l'ascenseur

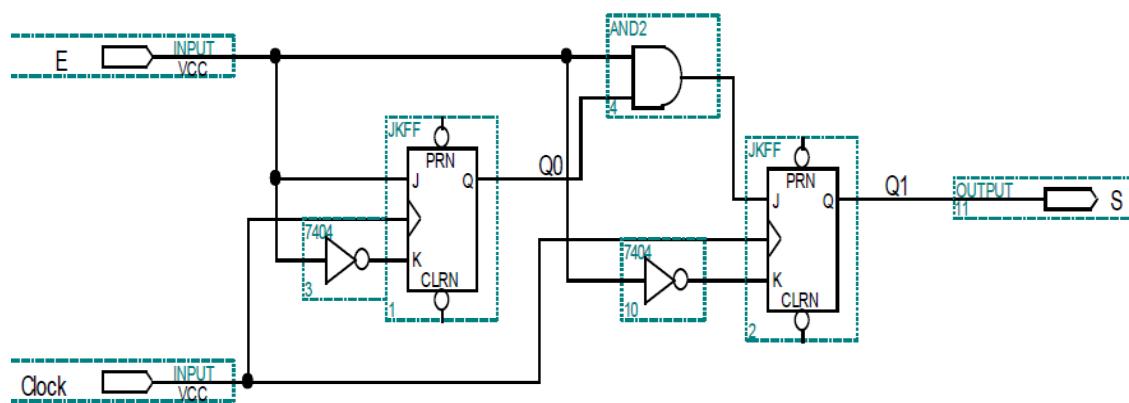
Comparaison machines de Moore / Mealy

Moore	Mealy
Les sorties dépendent de l'état courant	Les sorties dépendent de l'état courant et des entrées
Les sorties ne réagissent pas immédiatement aux entrées	Les sorties réagissent immédiatement aux entrées
L'état suivant est effectif au front d'horloge	L'état suivant est effectif au front d'horloge
Le changement de la sortie est synchrone par rapport au changement de l'état courant	Le changement de la sortie est asynchrone par rapport au changement de l'état courant

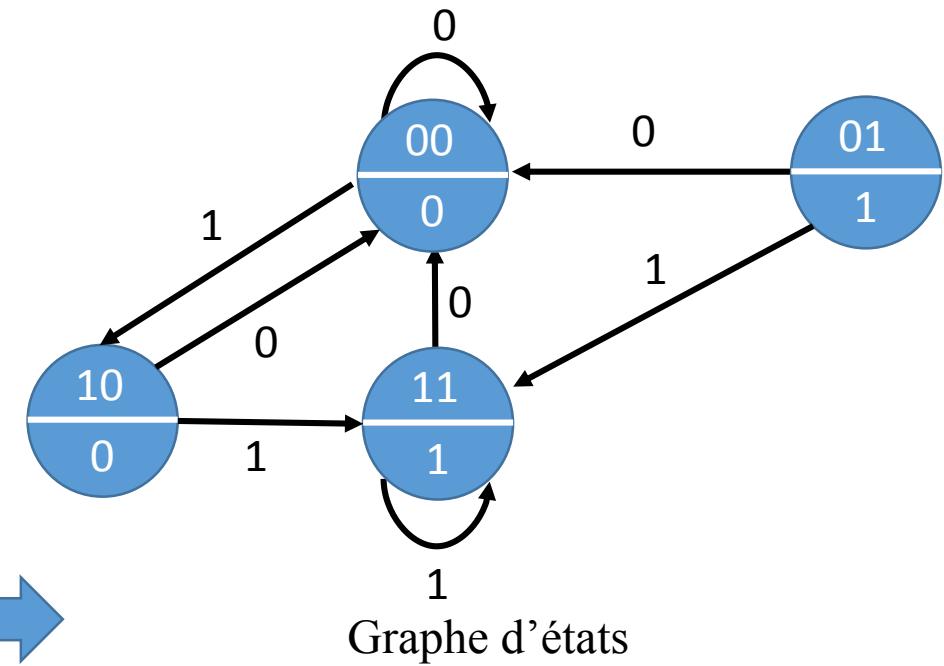
- Laquelle choisir?
 - Il faut se poser la question, est-ce qu'on a besoin que les sorties soient synchrones avec les états ou pas?
 - La machine de **Mealy** réagit plus rapidement aux entrées et n'a pas besoin d'attendre le prochain front d'horloge
 - Une implémentation à base de machine de **Mealy** a tendance à avoir moins d'états que la machine de Moore → moins de bascules → moins de ressources
 - La machine de **Moore** est plus sûre, puisque les sorties sont synchrones avec l'horloge et ne changent pas à tout moment. En effet, la machine de Mealy pourrait créer des problèmes si on a deux machines interconnectées et que l'une prend comme entrées les sorties de l'autre. Dans ce cas, si les entrées de la première sont instables entre deux fronts d'horloge, cela aura le même effet, sur la deuxième machine



- Objectif
 - A partir d'un schéma électrique :
 - Etre capable de déterminer un graphe d'états
 - Obtenir **une aide visuelle permettant de comprendre le fonctionnement du système**



Analyse du système séquentiel



- Méthode

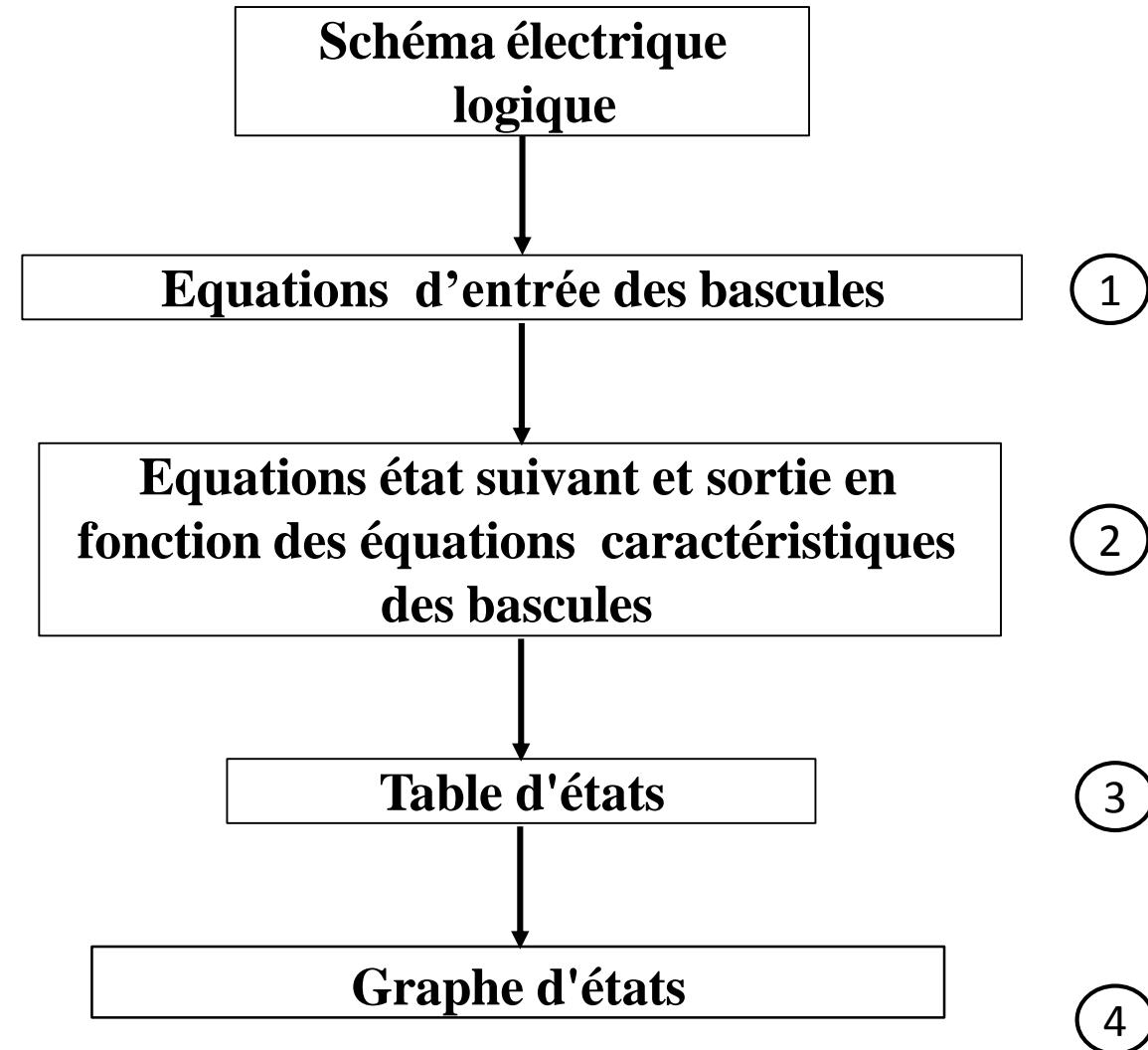
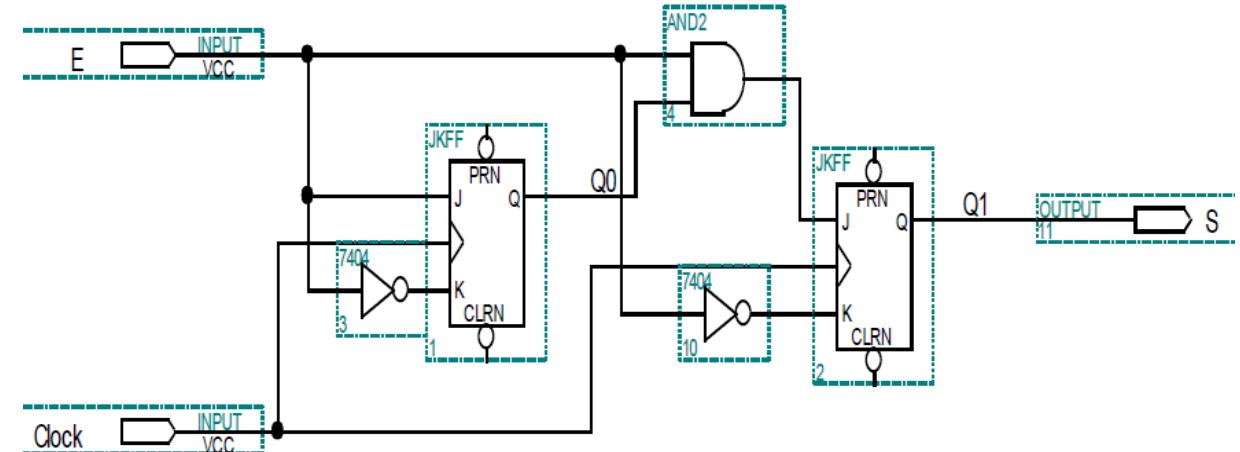


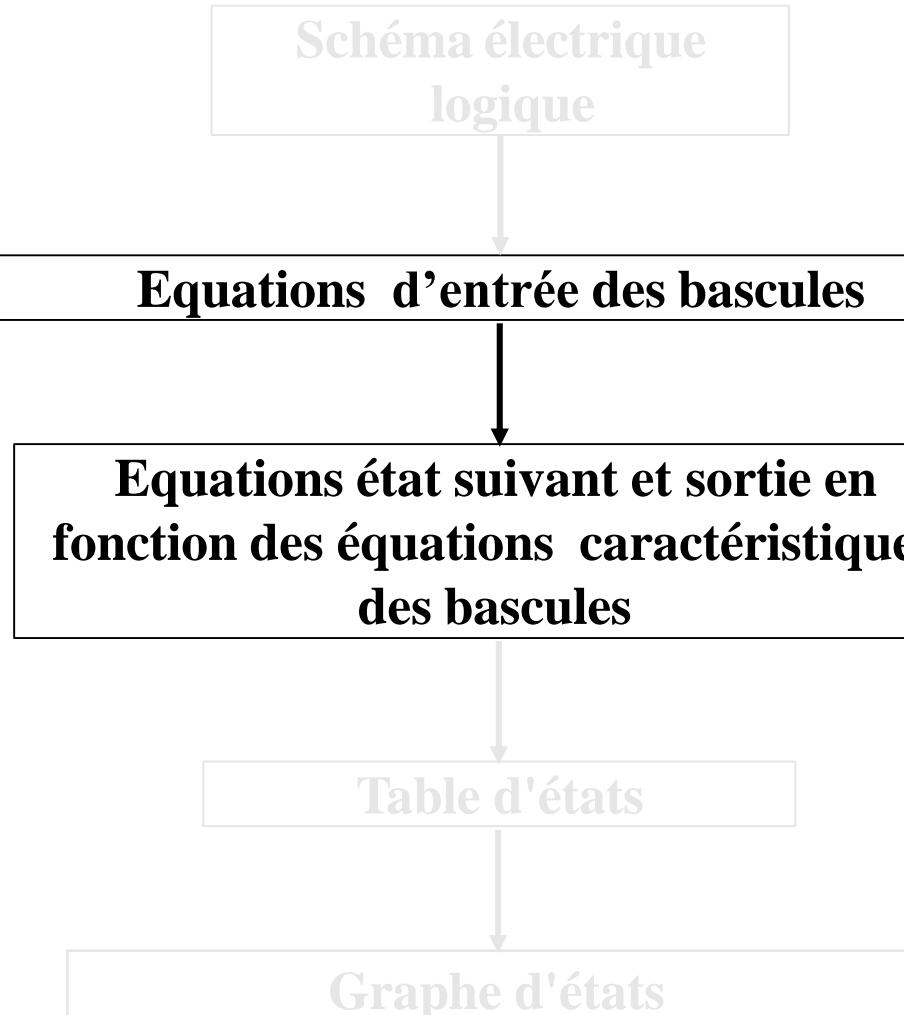
Schéma électrique logique**Equations d'entrée des bascules**

①

Equations état suivant et sortie en fonction des équations caractéristiques des bascules

Table d'états**Graphe d'états**

- Les équations des entrées des bascules sont:
 - $J_0 = E$
 - $K_0 = \overline{E}$
 - $J_1 = E \cdot Q_0$
 - $K_1 = \overline{E}$



Analyse de ce système séquentiel:

Etape 2: Les équations état suivant et sortie en fonction des équations caractéristiques des bascules

- $J_0 = E$
- $K_0 = \bar{E}$
- $J_1 = E \cdot Q_0$
- $K_1 = \bar{E}$

E	Q ₀	Q ₁	J ₀	K ₀	J ₁	K ₁	Q ₀₊	Q ₁₊
0	0	0	0	1	0	1	0	0
0	0	1	0	1	0	1	0	0
0	1	0	0	1	0	1	0	0
0	1	1	0	1	0	1	0	0
1	0	0	1	0	0	0	1	0
1	0	1	1	0	0	0	1	1
1	1	0	1	0	1	0	1	1
1	1	1	1	0	1	0	1	1

Schéma électrique logique

Equations d'entrée des bascules

Equations état suivant et sortie en fonction des équations caractéristiques des bascules

Analyse de ce système séquentiel:
Etape 2: Les équations état suivant et sortie en fonction des équations caractéristiques des bascules

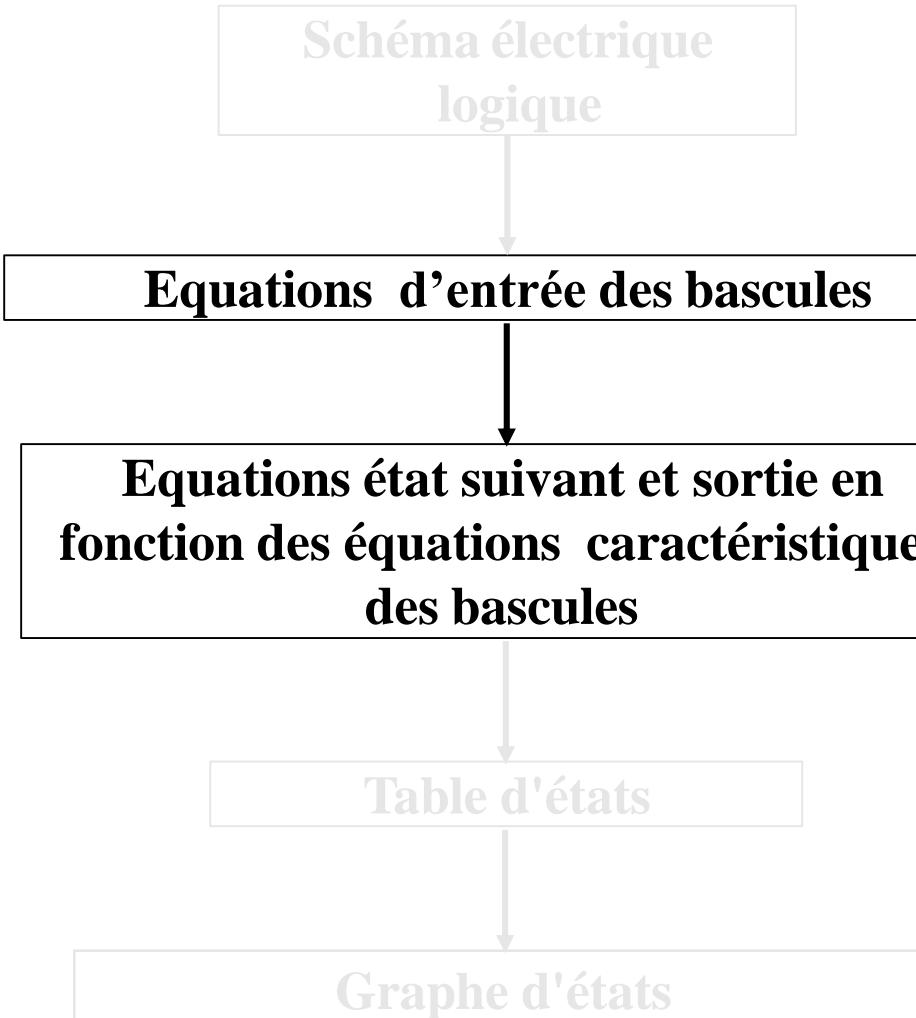
E	Q0	Q1	J0	K0	J1	K1	Q0+	Q1+
0	0	0	0	1	0	1	0	0
0	0	1	0	1	0	1	0	0
0	1	0	0	1	0	1	0	0
0	1	1	0	1	0	1	0	0
1	0	0	1	0	0	0	1	0
1	0	1	1	0	0	0	1	1
1	1	0	1	0	1	0	1	1
1	1	1	1	0	1	0	1	1

$Q0+ = E$ (directement de la table de vérité car $Q0+$ est à 1 pour toutes les combinaisons possibles entre $Q0$ et $Q1$ à condition d'avoir $E=1$)

$Q1+ = E \cdot Q1 + E \cdot Q0$ (passage par le tableau de Karnaugh)

E\Q0Q1	00	01	11	10
0	0	0	0	0
1	0	1	1	1

Tableau de Karnaugh pour $Q1+$



Analyse de ce système séquentiel:
Etape 2: Les équations état suivant et sortie en fonction des équations caractéristiques des bascules

$$S = Q_1 \text{ (directement du schéma électrique)}$$

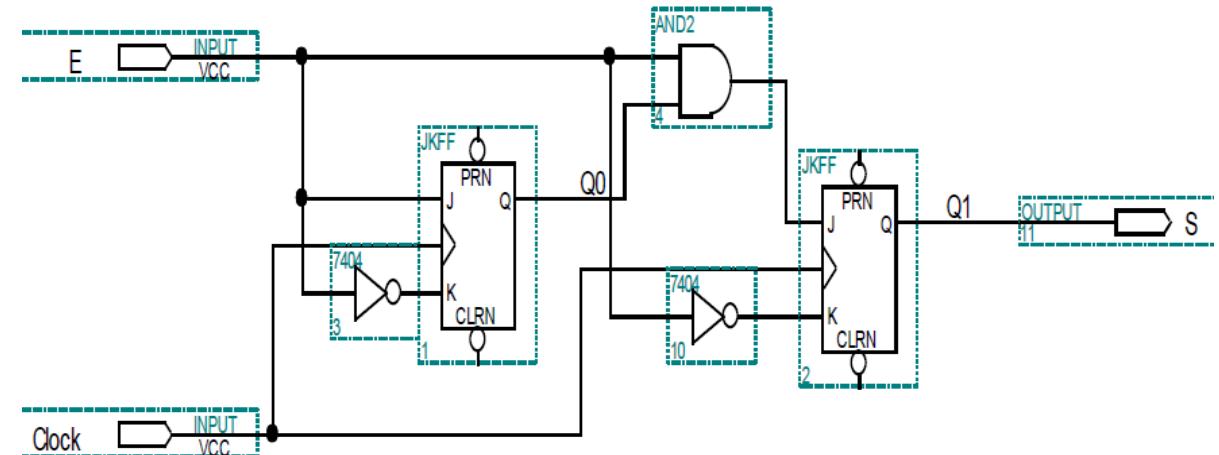


Schéma électrique logique

Equations d'entrée des bascules

Equations état suivant et sortie en fonction des équations caractéristiques des bascules

Table d'états

(3)

Graphe d'états

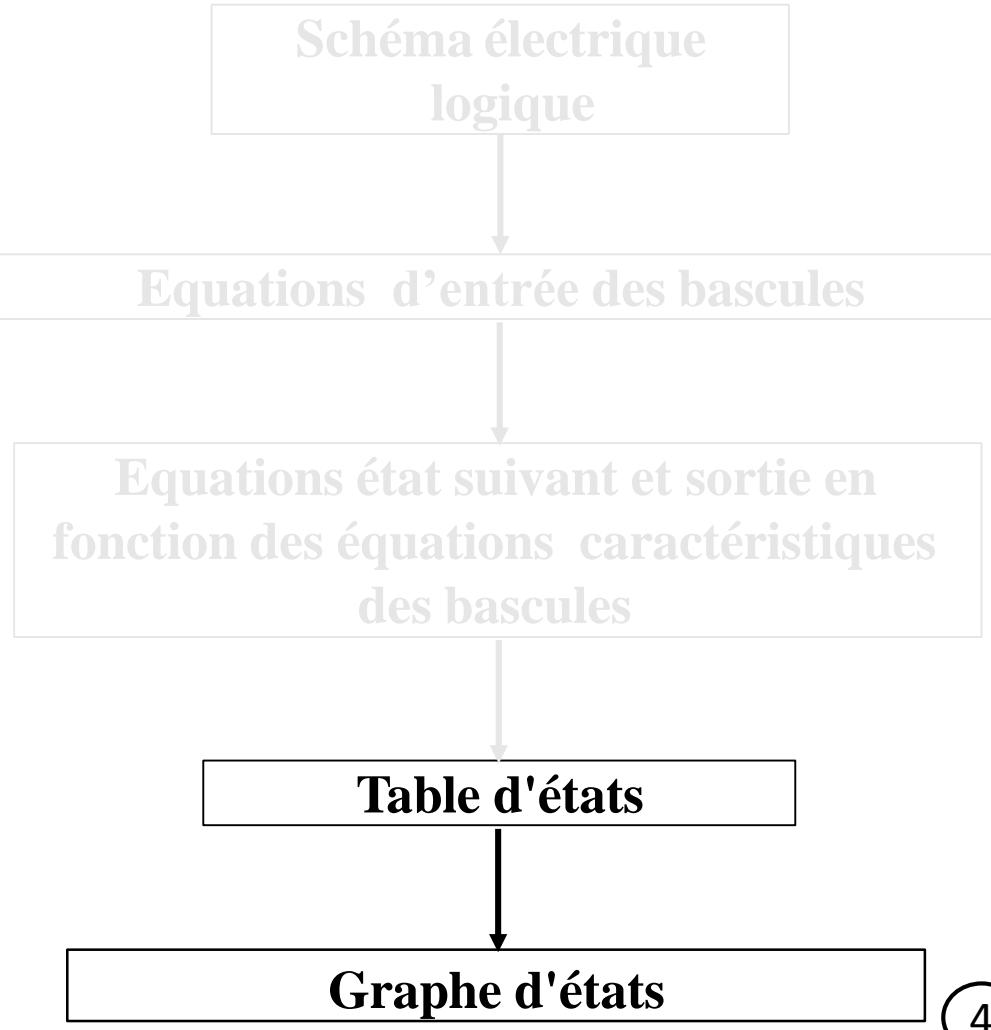
Analyse de ce système séquentiel:
Etape 3: la table d'états
Q0 représente le bit de poids fort de l'état

$$S = Q_1$$

E	Q0	Q1	Q0+	Q1+
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

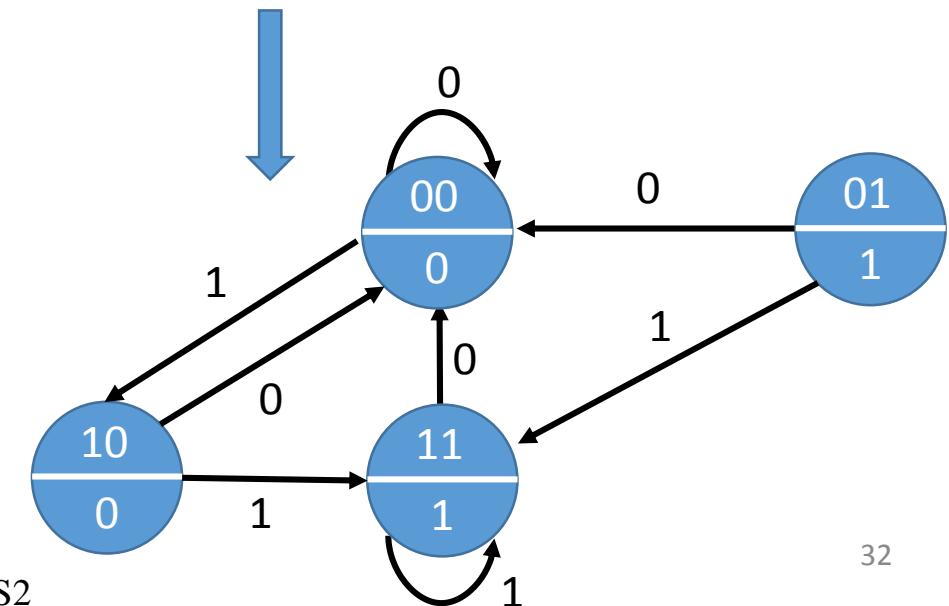


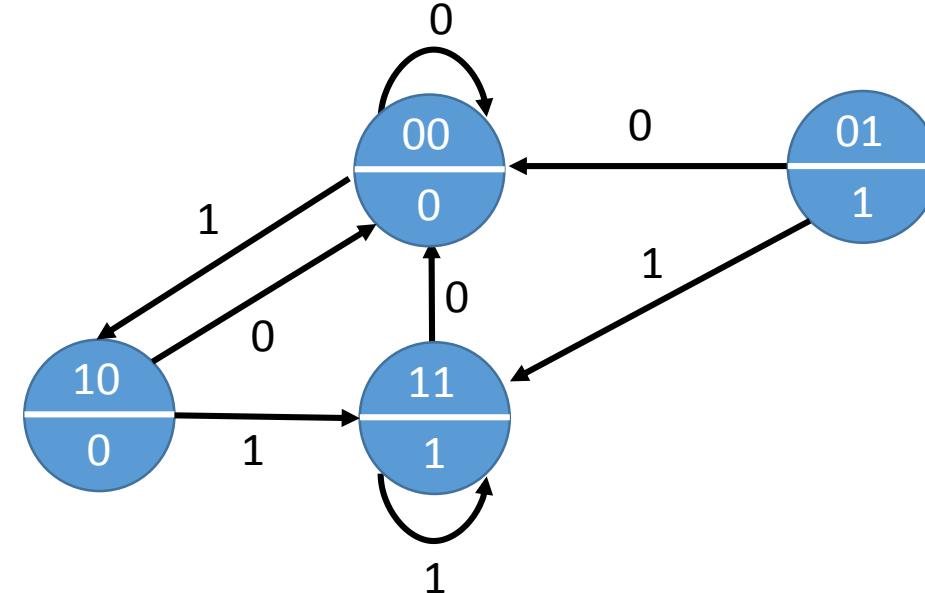
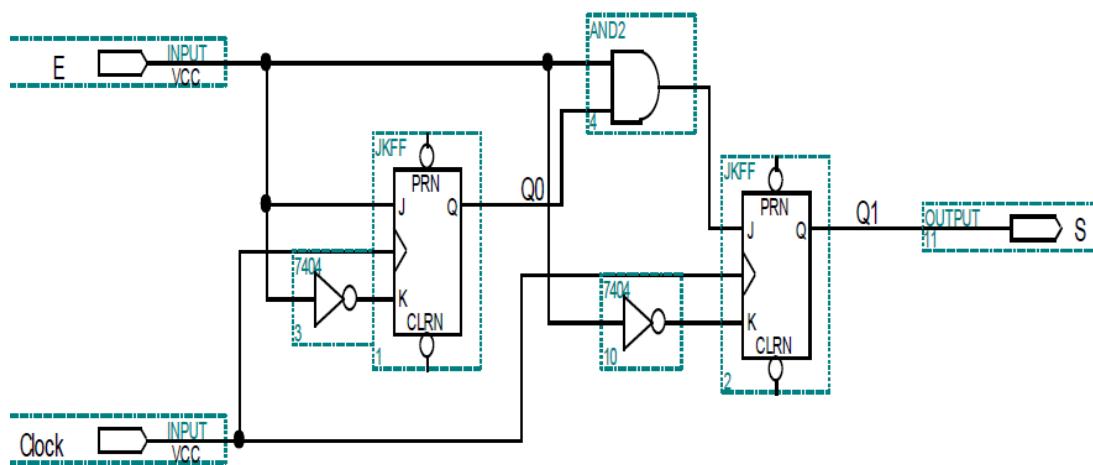
E \ Q0Q1	0	1	S
00	00	10	0
01	00	11	1
10	00	11	0
11	00	11	1



Analyse de ce système séquentiel:
Etape 4: Graphe d'états

E \ Q0Q1	0	1	S
00	00	10	0
01	00	11	1
10	00	11	0
11	00	11	1





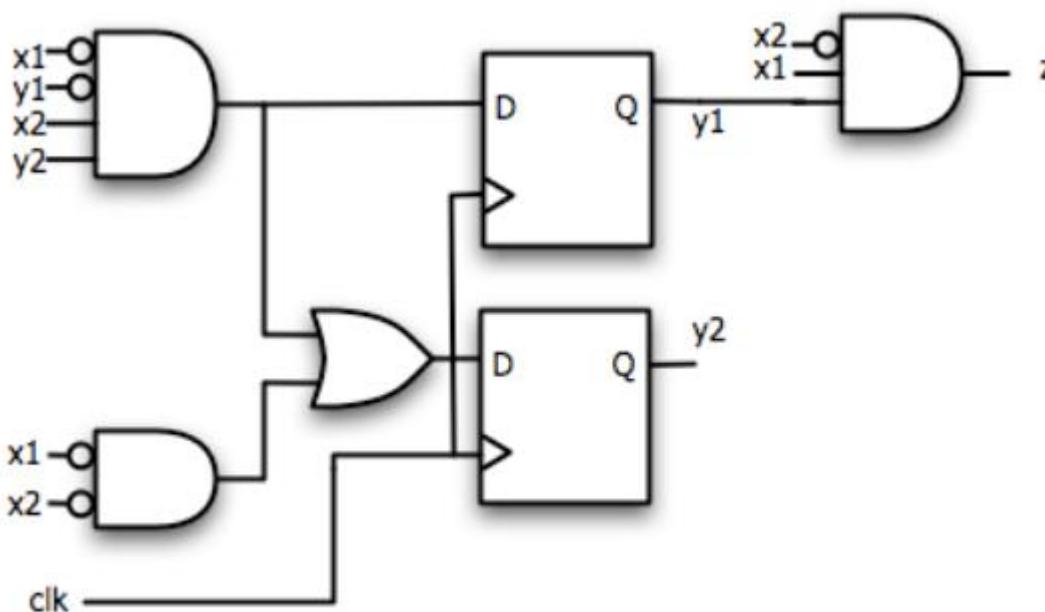
En appliquant l'analyse sur ce schéma électrique, on a déduit son fonctionnement:

Au démarrage, le système se trouve à l'état 00 et sa sortie est 0. Il y reste tant que la valeur de E est à 0. Quand la valeur de E passe à 1, il passe à l'état 10 mais sa sortie reste à 0, etc.

L'état 01 n'est pas accessible à partir des autres états, si par hasard (problème de tension) on a $Q_0=0$ et $Q_1=1$, le système passe selon la valeur de E soit en 00 soit en 11.

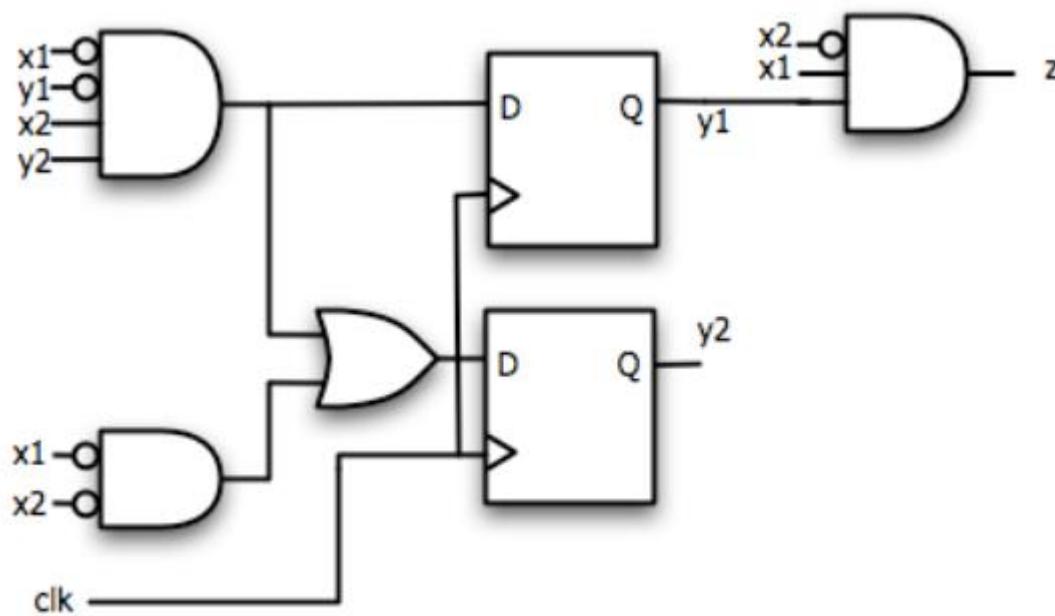
→ C'est une machine séquentielle qui présente un niveau logique 1 sur sa sortie lorsqu'elle reçoit au moins deux « 1 » consécutifs sur son entrée série.

Exemple2: avec des bascules D



- 1) Quel est le type de cette machine séquentielle?
- 2) Faire l'analyse de cette machine séquentielle:
 - Etape 1: Les équation d'entrée des bascules
 - Etape 2: Les équations état suivant et sortie en fonction des équations caractéristiques des bascules
 - Etape 3: la table d'états
 - Etape 4: graphe d'états

Exemple2: avec des bascules D



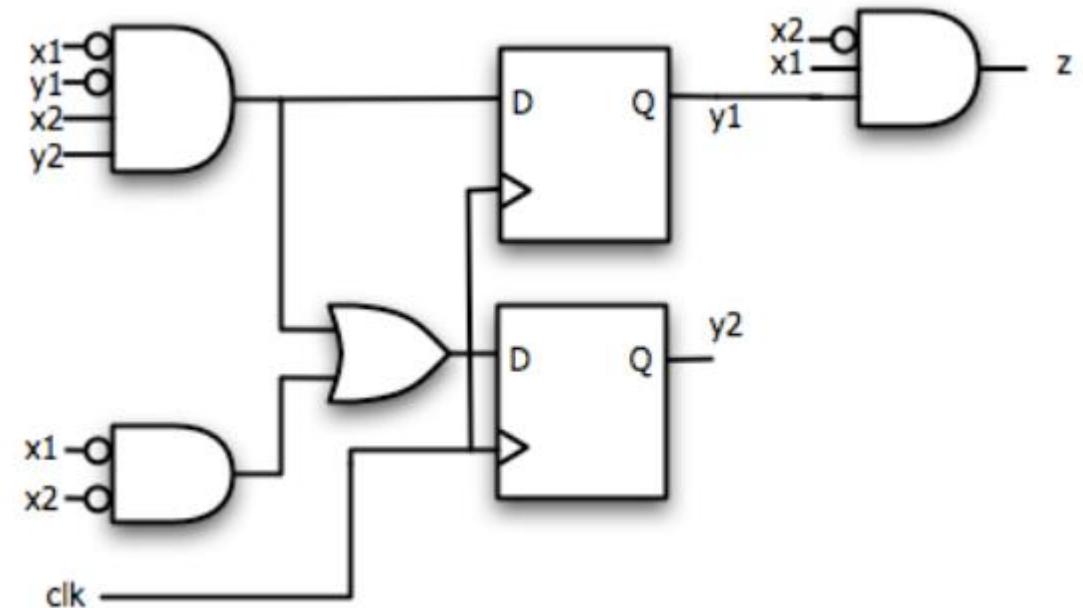
- 1) C'est une machine de Mealy puisque la sortie Z dépend des entrées

Exemple2: avec des bascules D

Etape 1: Les équation d'entrée des bascules

$$\overline{D1} = \overline{X1} \cdot \overline{Y1} \cdot X2 \cdot Y2$$

$$\overline{D2} = \overline{X1} \cdot \overline{Y1} \cdot \overline{X2} \cdot \overline{Y2} + \overline{X1} \cdot \overline{X2}$$



Exemple2: avec des bascules D

Etape 2: Les équations état suivant et sortie en fonction des équations caractéristiques des bascules

- Pour une bascule D: $Q+=D$ (ce qu'on a sur l'entrée D, on l'aura sur la sortie Q au prochain cycle d'horloge)

$$D_1 = \overline{X_1} \cdot \overline{Y_1} \cdot X_2 \cdot Y_2$$

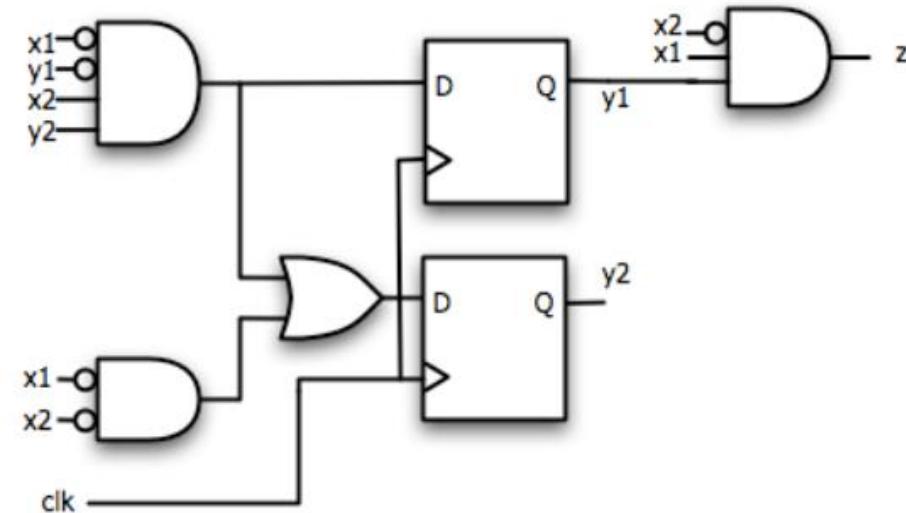
$$D_2 = \overline{X_1} \cdot \overline{Y_1} \cdot X_2 \cdot Y_2 + \overline{X_1} \cdot \overline{X_2}$$

$$Z = \overline{X_2} \cdot X_1 \cdot Y_1$$



$$Y_1+ = \overline{X_1} \cdot \overline{Y_1} \cdot X_2 \cdot Y_2$$

$$Y_2+ = \overline{X_1} \cdot \overline{Y_1} \cdot X_2 \cdot Y_2 + \overline{X_1} \cdot \overline{X_2}$$



Exemple2: avec des bascules D

Etape 3: la table d'états

Table de vérité

X1	X2	Y1	Y2	Y1+	Y2+	Z
0	0	0	0	0	1	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	1	1	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	1
1	0	1	1	0	0	1
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0

$$Y1+ = \overline{X1} \cdot \overline{Y1} \cdot X2 \cdot Y2$$

$$Y2+ = \overline{X1} \cdot \overline{Y1} \cdot X2 \cdot Y2 + \overline{X1} \cdot \overline{X2}$$

$$Z = \overline{X2} \cdot X1 \cdot Y1$$

Table d'états

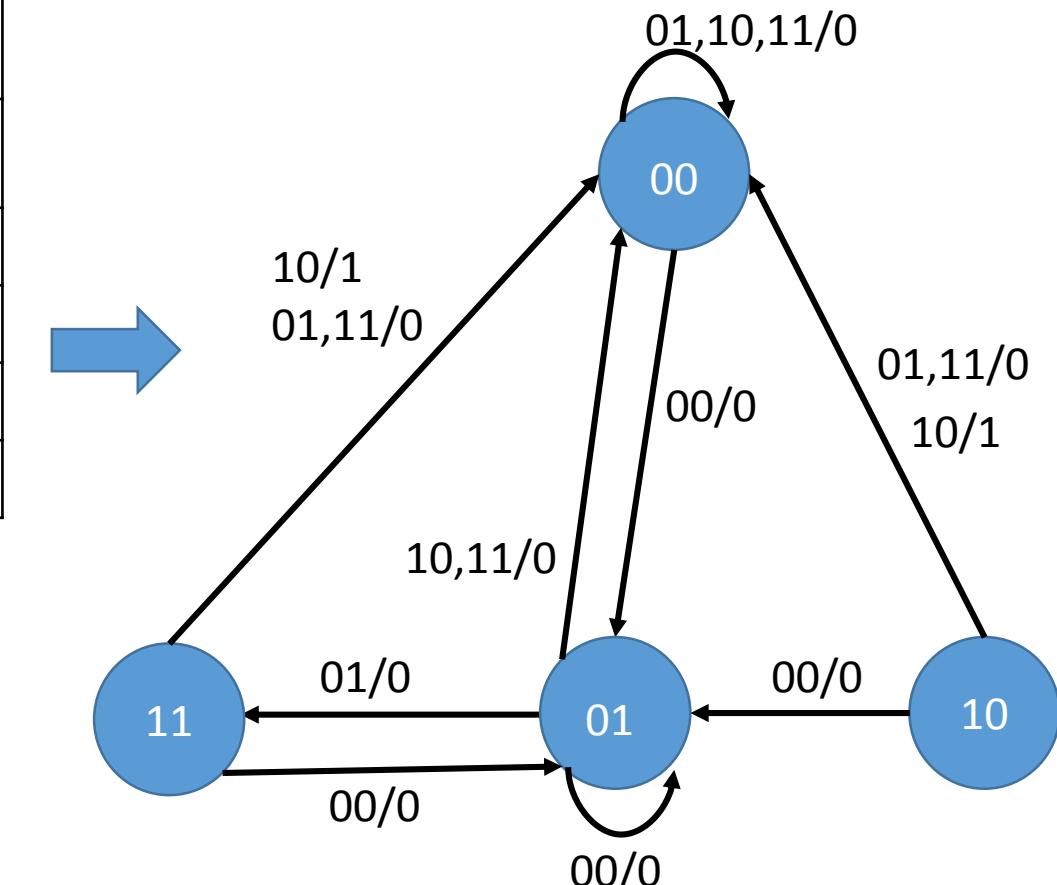
Etat présent	Etat suivant				Sortie				
	Y1Y2	X1X2= 00	X1X2= 01	X1X2= 10	X1X2= 11	X1X2= 00	X1X2= 01	X1X2= 10	X1X2= 11
00	01	00	00	00	00	0	0	0	0
01	01	11	00	00	00	0	0	0	0
10	01	00	00	00	00	0	0	1	0
11	01	00	00	00	00	0	0	1	0

Exemple2: avec des bascules D

Etape 4: graphe d'états

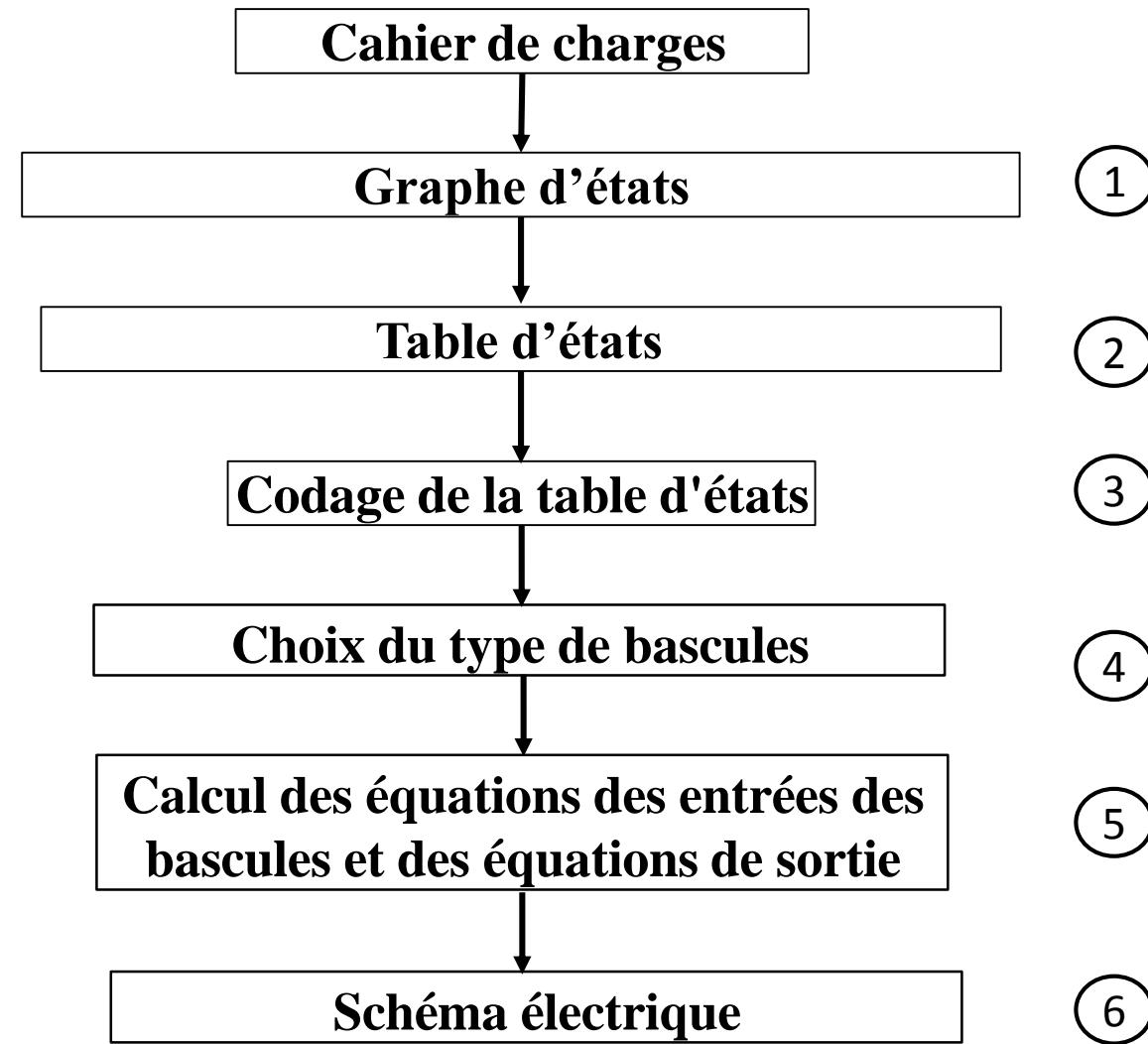
Etat présent	Etat suivant				Sortie			
	X1X2= 00	X1X2= 01	X1X2= 10	X1X2= 11	X1X2= 00	X1X2= 01	X1X2= 10	X1X2= 11
00	01	00	00	00	0	0	0	0
01	01	11	00	00	0	0	0	0
10	01	00	00	00	0	0	1	0
11	01	00	00	00	0	0	1	0

Table d'états



Graphe d'états

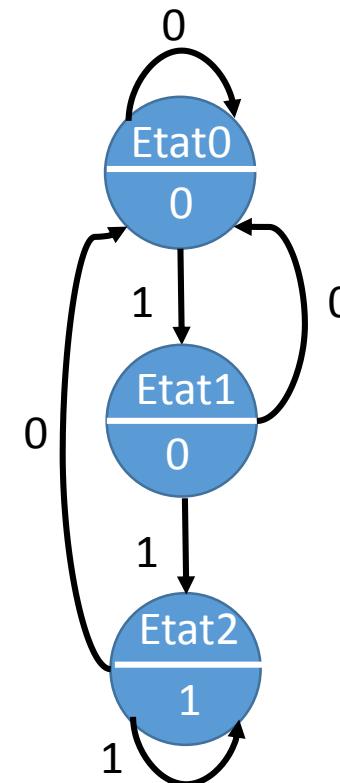
• Méthode



- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.

Exemple avec Machine de Moore

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 1: Graphe d'états



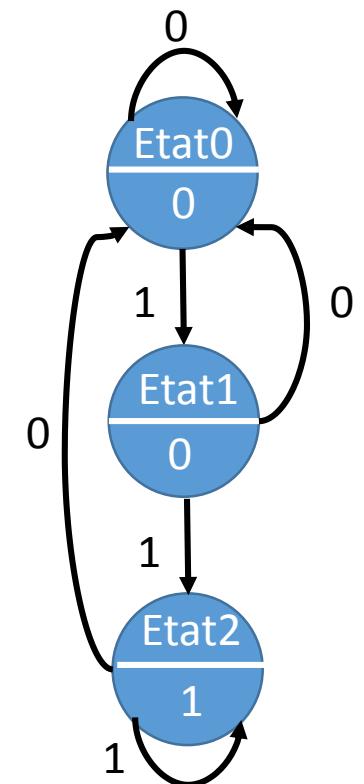
Machine de Moore

Exemple avec Machine de Moore

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 2: Table d'états

Etat présent	Etat suivant		Sortie
	E=0	E=1	
Etat0	Etat0	Etat1	0
Etat1	Etat0	Etat2	0
Etat2	Etat0	Etat2	1

- L'état Etat0 est l'état où le dernier caractère lu est 0.
- L'état Etat1 est l'état où le dernier caractère lu est 1. Ce 1 peut être soit le tout premier caractère lu ou bien précédé par un 0.
- L'état Etat2 est l'état où au moins les deux derniers caractères lus sont des 1.



Exemple avec Machine de Moore

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 3: Codage de la table d'états

Nom état	Code (Q1Q0)
Etat0	00
Etat1	01
Etat2	10

- Les états seront représentés par les sorties des bascules (voir cours précédent)
- On a choisi de coder les états sur 2 bits (le nombre minimum de bits nécessaires), on a donc besoin de 2 bascules. Le bit de poids fort est donné par la sortie Q1 et le bit de poids faible par la sortie Q0.

Etat présent	Etat suivant		Sortie
	E=0	E=1	
Etat0	Etat0	Etat1	0
Etat1	Etat0	Etat2	0
Etat2	Etat0	Etat2	1

↓
Codage

Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1

Exemple avec Machine de Moore

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 4: Choix du type des bascules
 - 2 bascules D

Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1

Exemple avec Machine de Moore

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 5: Calcul des équations des entrées des bascules et des équations de sortie
 - Pour une bascule D, l'équation de l'entrée est $D=Q+$, avec $Q+$ la sortie au prochain front d'horloge
 ➔ les équations D_1 et D_0 sont données par les sorties Q_1+ et Q_0+ (qui représentent l'état suivant)

Q1	Q0	E	D1	D0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1

Exemple avec Machine de Moore

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 5: Calcul des équations des entrées des bascules et des équations de sortie
 - $D = Q +$

Q1	Q0	E	D1	D0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1

NB: On a 3 états, qui sont codés par 00, 01 et 10. Il reste donc la combinaison 11 qui n'est pas gérée par le système. Il faut donc prévoir ce cas où les sorties de bascules se trouvent toutes les deux à 1 (si jamais cela se produit à cause d'un problème d'initialisation des bascules).

Dans ce cas, si le système se trouve au départ à l'état 11 et qu'on reçoit 0, on passe à l'état 00 (rappel: l'état 00 correspond à l'état où le dernier caractère lu est 0) et si on reçoit 1, on passe à l'état 01.

Il faut noter que se trouver à l'état 11 ne pourrait se produire qu'au départ (suite à un problème d'initialisation) et si on quitte cet état on y retournera jamais.

Exemple avec Machine de Moore

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 5: Calcul des équations des entrées des bascules et des équations de sortie
 - $D = Q +$

Q1	Q0	E	D1	D0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

E\Q1Q0	00	01	11	10
0	0	0	0	0
1	0	1	0	1

$$D1 = E \cdot (Q0 \oplus Q1)$$

E\Q1Q0	00	01	11	10
0	0	0	0	0
1	1	0	1	0

$$\begin{aligned}
 D0 &= E \cdot \overline{Q0} \cdot \overline{Q1} + E \cdot Q0 \cdot Q1 \\
 &= E \cdot (Q0 \cdot Q1 + Q0 \cdot Q1) \\
 &= E \cdot (Q0 \oplus Q1)
 \end{aligned}$$

Exemple avec Machine de Moore

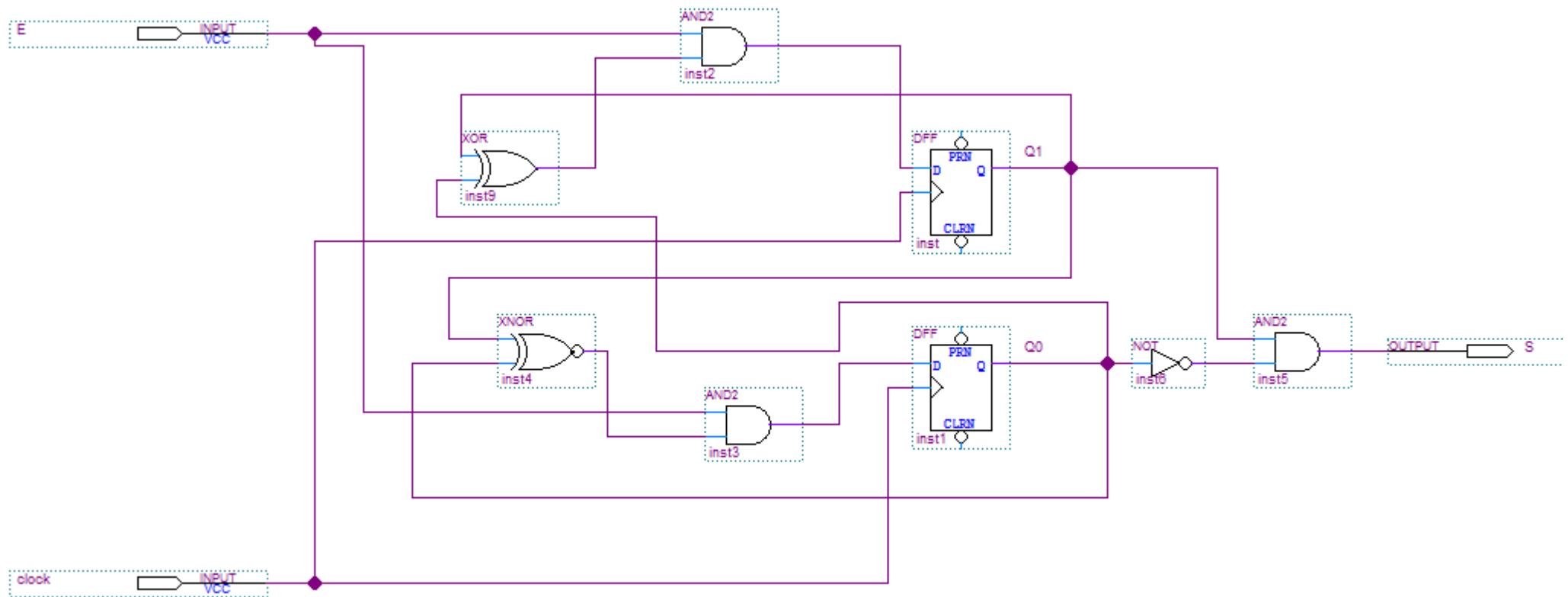
- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 5: Calcul des équations des entrées des bascules et des équations de sortie

Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1

$$S = Q_1 \cdot \overline{Q_0}$$

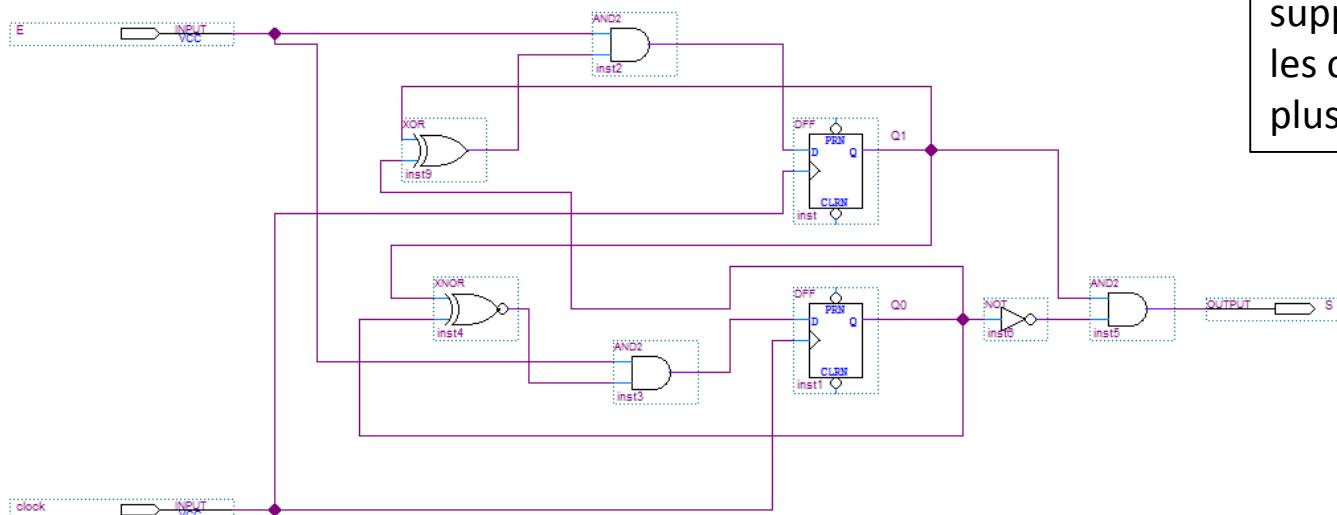
Exemple avec Machine de Moore

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 6: Schéma électrique

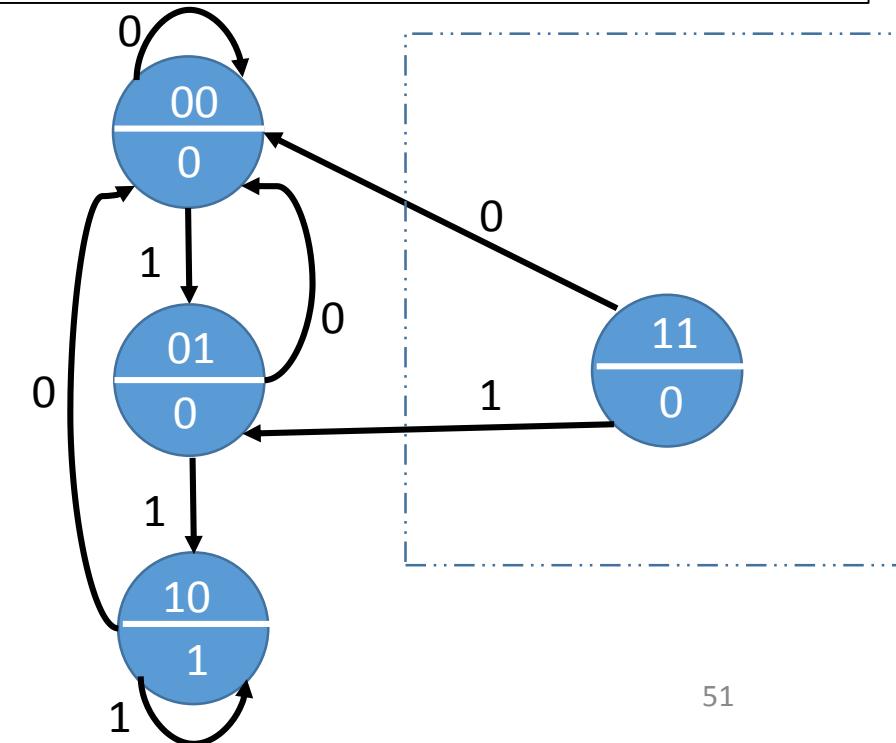


Exemple avec Machine de Moore

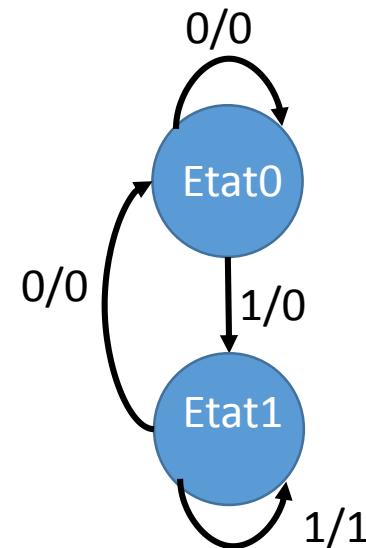
- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 6: Schéma électrique



NB: Si on part de ce schéma électrique pour avoir le graphe d'états (analyse du système séquentiel), on aura le graphe d'états suivant: le même graphe de départ + l'état 11 (l'état supplémentaire est dû au fait que l'analyse prend en compte toutes les combinaisons possibles des sorties des bascules, on a donc en plus des états 00, 01 et 10, l'état 11).



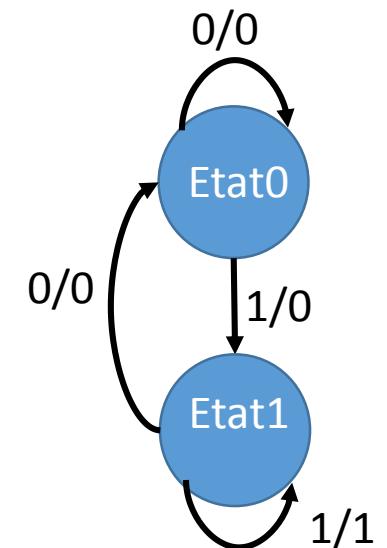
- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 1: Graphe d'états



Exemple avec Machine de Mealy

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 2: Table d'états

Etat présent	Etat suivant/Sortie	
	E=0	E=1
Etat0	Etat0/0	Etat1/0
Etat1	Etat0/0	Etat1/1



Exemple avec Machine de Mealy

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 3: Codage de la table d'états

Nom état	Q0
Etat0	0
Etat1	1

Etat présent	Etat suivant/Sortie	
	E=0	E=1
Etat0	Etat0/0	Etat1/0
Etat1	Etat0/0	Etat1/1

Etat présent	Etat suivant/Sortie	
	E=0	E=1
0	0/0	1/0
1	0/0	1/1

Table d'états codée

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 4: Choix du type des bascules
 - 1 bascule JK

Exemple avec Machine de Mealy

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 5: Calcul des équations des entrées des bascules et des équations de sortie

Etat présent	Etat suivant/Sortie	
	E=0	E=1
0	0/0	1/0
1	0/0	1/1

Q0	E	Q0+	J 0	K0	S
0	0	0	0	X	0
0	1	1	1	X	0
1	0	0	X	1	0
1	1	1	X	0	1

Q0\E	0	1
0	0	1
1	X	X

$$J_0 = E$$

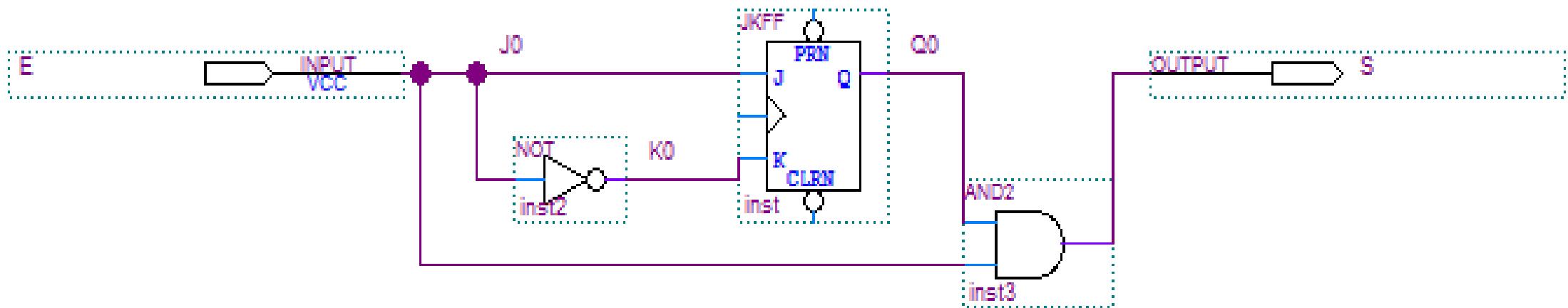
Q0\E	0	1
0	X	X
1	1	0

$$K_0 = \underline{\hspace{1cm}}$$

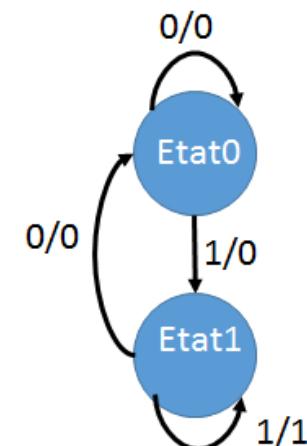
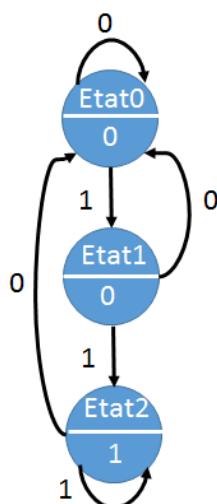
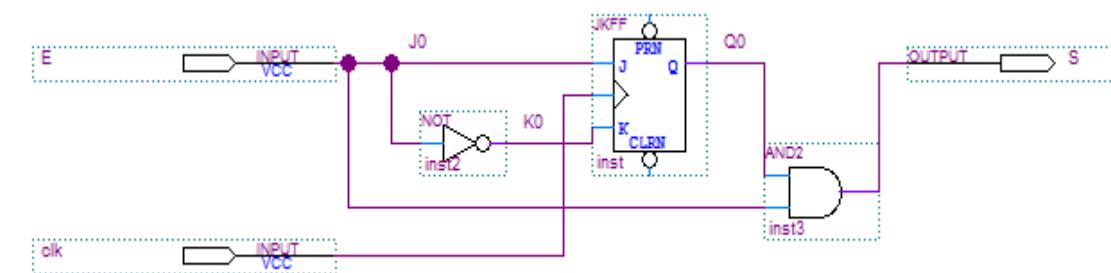
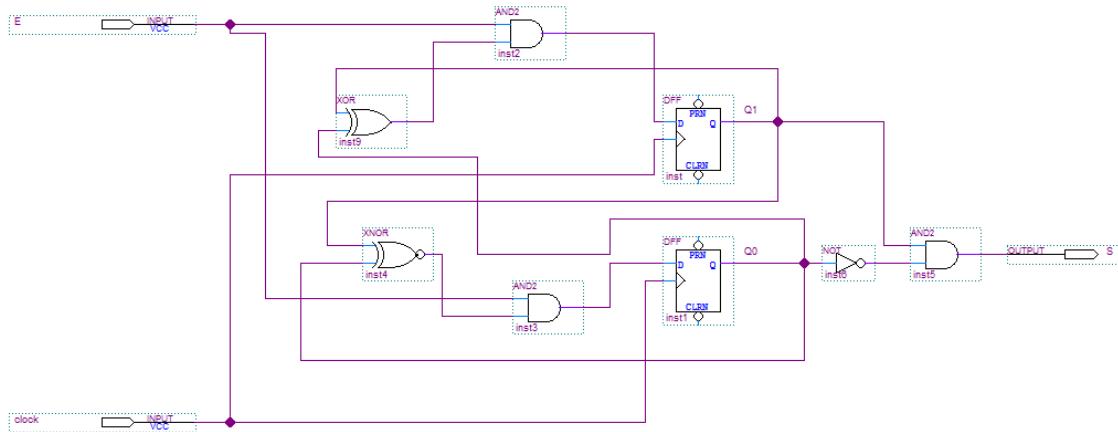
$$S = E \cdot Q_0$$

Exemple avec Machine de Mealy

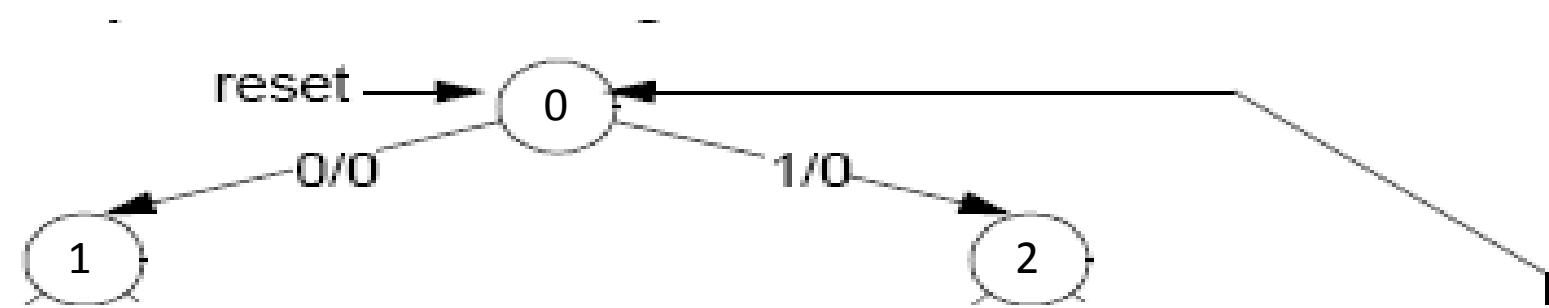
- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.
- Etape 6: Schéma électrique



- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.

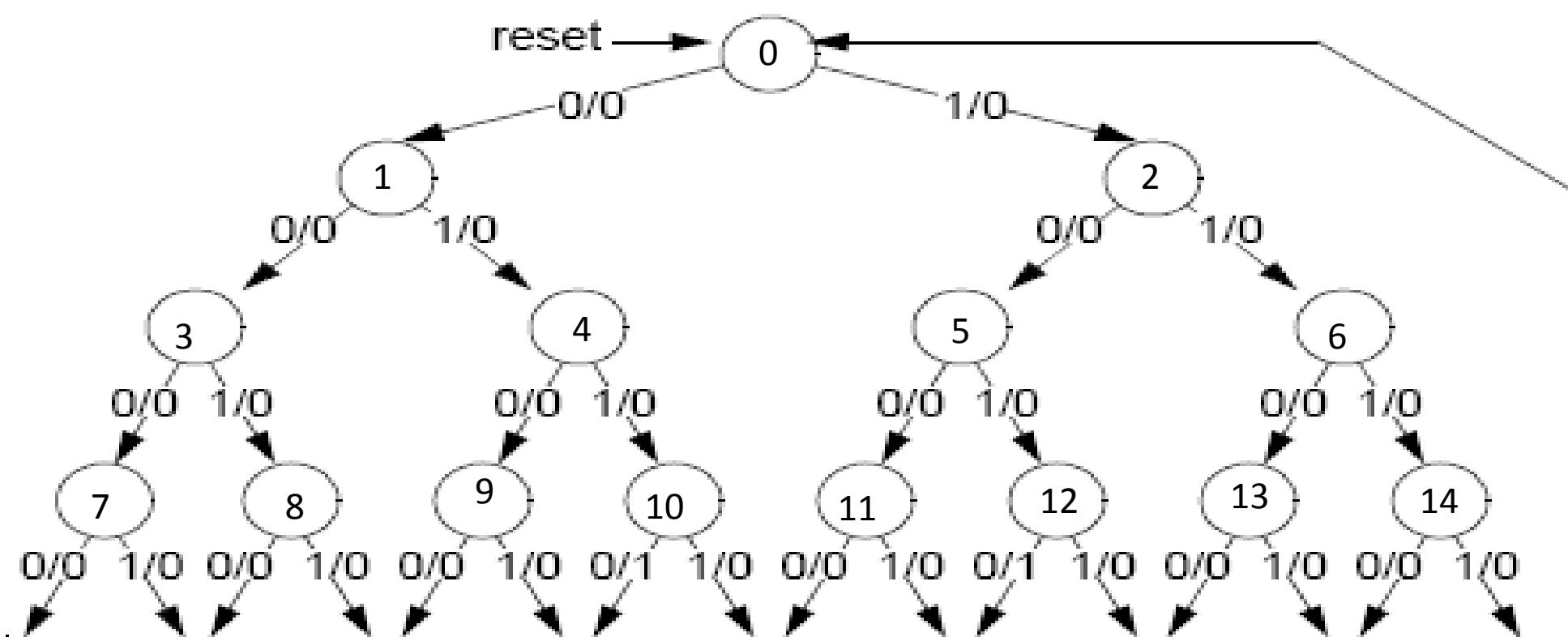


- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie dès qu'elle aura reçu le code '0110' ou '1010'.
- La machine sera resetée après chaque séquence de 4 bits.



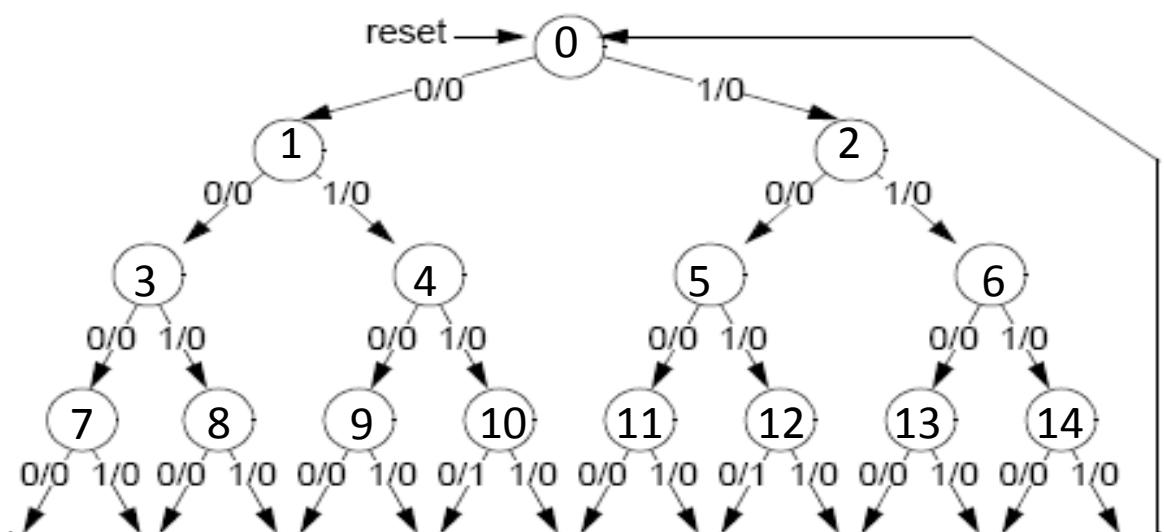
Codage de table d'états: Exemple de codage pas simple

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie dès qu'elle aura reçu le code '0110' ou '1010' .
 - La machine sera resetée après chaque séquence de 4 bits.
- Si vous voyez que la machine optimale (avec le minimum d'états possible) est compliquée à avoir dès le premier coup, commencez par la méthode naïve qui implémente toutes les combinaisons d'entrées



Codage de table d'états: Exemple de codage pas simple

- Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie dès qu'elle aura reçu le code '0110' ou '1010'.
- La machine sera resetée après chaque séquence de 4 bits.



Les deux combinaisons qui donnent une sortie à 1

Séquence d'entrée	Etat présent	Etat suivant/Sortie	
		x=0	x=1
reset	Etat0	Etat1/0	Etat2/0
0	Etat1	Etat3/0	Etat4/0
1	Etat2	Etat5/0	Etat6/0
00	Etat3	Etat7/0	Etat8/0
01	Etat4	Etat9/0	Etat10/0
10	Etat5	Etat11/0	Etat12/0
11	Etat6	Etat13/0	Etat14/0
000	Etat7	Etat0/0	Etat0/0
001	Etat8	Etat0/0	Etat0/0
010	Etat9	Etat0/0	Etat0/0
011	Etat10	Etat0/1	Etat0/0
100	Etat11	Etat0/0	Etat0/0
101	Etat12	Etat0/1	Etat0/0
110	Etat13	Etat0/0	Etat0/0
111	Etat14	Etat0/0	Etat0/0

- Problème
 - 1 état induit un coût au niveau hardware
 - Plus le nombre d'états augmente
 - plus le **nombre de bascules** nécessaires pour le codage augmente,
 - plus le **nombre de circuit logiques combinatoires**, pour prendre en charge le grand nombre de transitions, augmente.
- Solution
 - Réduire le nombre d'états sans changer le fonctionnement global du système
 - Moins coûteux
- Règles
 - On peut éliminer un état s'il est équivalent à un ou plusieurs.
 - Deux états sont équivalents si :
 - Les 2 produisent une même sortie (pour les mêmes valeurs d'entrées pour une machine de Mealy)
 - Les deux conduisent vers de mêmes états.

- Méthode
 - Écrire la table d'états d'origine
 - Pour Mealy : séparer la sortie des cases état suivant → ajouter des colonnes
 - Regrouper les lignes identiques (mêmes sorties et mêmes états suivants) en un nouvel état.
 - Réécrire la table en remplaçant les anciens états équivalents par le nouvel état.
 - Recommencer jusqu'à minimisation finale.

Minimisation du nombre d'états

Séquence d'entrée	Etat présent	Etat suivant		Sortie	
		x=0	x=1	x=0	x=1
reset	Etat0	Etat1	Etat2	0	0
0	Etat1	Etat3	Etat4	0	0
1	Etat2	Etat5	Etat6	0	0
00	Etat3	Etat7	Etat8	0	0
01	Etat4	Etat9	Etat10	0	0
10	Etat5	Etat11	Etat12	0	0
11	Etat6	Etat13	Etat14	0	0
000	Etat7	Etat0	Etat0	0	0
001	Etat8	Etat0	Etat0	0	0
010	Etat9	Etat0	Etat0	0	0
011	Etat10	Etat0	Etat0	1	0
100	Etat11	Etat0	Etat0	0	0
101	Etat12	Etat0	Etat0	1	0
110	Etat13	Etat0	Etat0	0	0
111	Etat14	Etat0	Etat0	0	0

Séquence d'entrée	Etat présent	Etat suivant		Sortie	
		x=0	x=1	x=0	x=1
reset	Etat0	Etat1	Etat2	0	0
0	Etat1	Etat3	Etat4	0	0
1	Etat2	Etat5	Etat6	0	0
00	Etat3	Etat7	Etat8	0	0
01	Etat4	Etat9	Etat10'	0	0
10	Etat5	Etat11	Etat10'	0	0
11	Etat6	Etat13	Etat14	0	0
000	Etat7	Etat0	Etat0	0	0
001	Etat8	Etat0	Etat0	0	0
010	Etat9	Etat0	Etat0	0	0
011/101	Etat10'	Etat0	Etat0	1	0
100	Etat11	Etat0	Etat0	0	0
110	Etat13	Etat0	Etat0	0	0
111	Etat14	Etat0	Etat0	0	0

Minimisation du nombre d'états

Séquence d'entrée	Etat présent	Etat suivant		Sortie	
		x=0	x=1	x=0	x=1
reset	Etat0	Etat1	Etat2	0	0
0	Etat1	Etat3	Etat4	0	0
1	Etat2	Etat5	Etat6	0	0
00	Etat3	Etat7	Etat8	0	0
01	Etat4	Etat9	Etat10'	0	0
10	Etat5	Etat11	Etat10'	0	0
11	Etat6	Etat13	Etat14	0	0
000	Etat7	Etat0	Etat0	0	0
001	Etat8	Etat0	Etat0	0	0
010	Etat9	Etat0	Etat0	0	0
011/101	Etat10'	Etat0	Etat0	1	0
100	Etat11	Etat0	Etat0	0	0
110	Etat13	Etat0	Etat0	0	0
111	Etat14	Etat0	Etat0	0	0

Séquence d'entrée	Etat présent	Etat suivant		Sortie	
		x=0	x=1	x=0	x=1
reset	Etat0	Etat1	Etat2	0	0
0	Etat1	Etat3	Etat4	0	0
1	Etat2	Etat5	Etat6	0	0
00	Etat3	Etat7'	Etat7'	0	0
01	Etat4	Etat7'	Etat10'	0	0
10	Etat5	Etat7'	Etat10'	0	0
11	Etat6	Etat7'	Etat7'	0	0
!(011/101)	Etat7'	Etat0	Etat0	0	0
011/101	Etat10'	Etat0	Etat0	1	0

Minimisation du nombre d'états

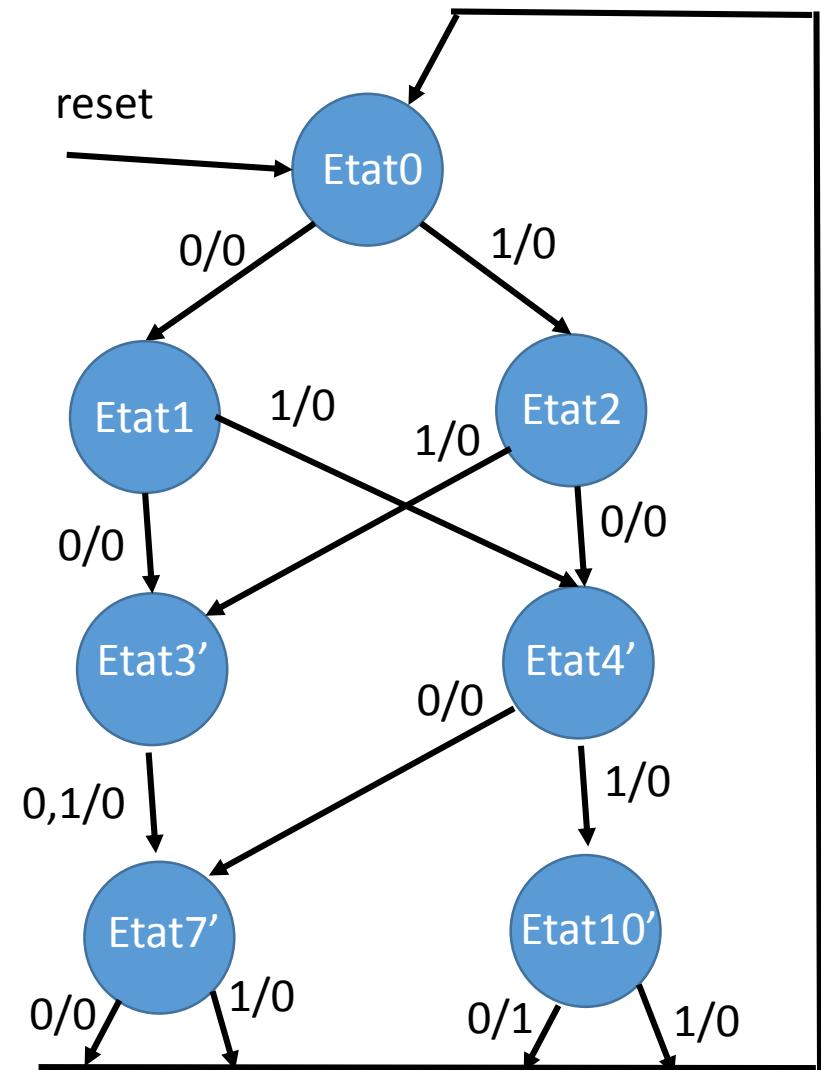
Séquence d'entrée	Etat présent	Etat suivant		Sortie	
		x=0	x=1	x=0	x=1
reset	Etat0	Etat1	Etat2	0	0
0	Etat1	Etat3	Etat4	0	0
1	Etat2	Etat5	Etat6	0	0
00	Etat3	Etat7'	Etat7'	0	0
01	Etat4	Etat7'	Etat10'	0	0
10	Etat5	Etat7'	Etat10'	0	0
11	Etat6	Etat7'	Etat7'	0	0
!(011/101)	Etat7'	Etat0	Etat0	0	0
011/101	Etat10'	Etat0	Etat0	1	0

Séquence d'entrée	Etat présent	Etat suivant		Sortie	
		x=0	x=1	x=0	x=1
reset	Etat0	Etat1	Etat2	0	0
0	Etat1	Etat3'	Etat4'	0	0
1	Etat2	Etat4'	Etat3'	0	0
00/11	Etat3'	Etat7'	Etat7'	0	0
01/10	Etat4'	Etat7'	Etat10'	0	0
!(011/101)	Etat7'	Etat0	Etat0	0	0
011/101	Etat10'	Etat0	Etat0	1	0

Minimisation du nombre d'états

Séquence d'entrée	Etat présent	Etat suivant		Sortie	
		x=0	x=1	x=0	x=1
reset	Etat0	Etat1	Etat2	0	0
0	Etat1	Etat3'	Etat4'	0	0
1	Etat2	Etat4'	Etat3'	0	0
00/11	Etat3'	Etat7'	Etat7'	0	0
01/10	Etat4'	Etat7'	Etat10'	0	0
!(011/101)	Etat7'	Etat0	Etat0	0	0
011/101	Etat10'	Etat0	Etat0	1	0

Concevez une machine séquentielle qui présentera un niveau logique 1 sur sa sortie dès qu'elle aura reçu le code '0110' ou '1010'. La machine sera resetée après chaque séquence de 4 bits.



- Le plus simple (codage binaire):
 - Pour s états, le nombre de bits est $\log_2(s)$.
 - Les s états sont codés par les s premières valeurs binaires dans l'ordre (000, 001, 010, 011, 100,).
 - Le codage le plus simple ne donne pas forcément les équations d'entrées de bascules, de sorties ou les circuits logiques les plus simples
- ➔ Le codage a un impact sur le nombre de ressources utilisées

- Utilisation d'un nombre de bits minimum (comme le codage binaire)
- Le passage d'un état à un autre ne modifie qu'un seul bit du code.
- Résultats comparables par rapport au codage binaire en termes de ressources et de vitesse
- Avantages
 - Le codage Gray est intéressant pour éviter les états temporaires indésirables si les bascules ne mettent pas à jour leurs sorties au même moment. Exemple: le passage de l'état « 01 » à l'état « 10 » dans le codage binaire peut entraîner des états temporaires erronés (« 00 » et « 11 »). Dans ce cas, si les sorties du système utilisent la valeur de l'état d'une manière asynchrone (machine de Mealy ou Moore asynchrone), on risque d'avoir des valeurs de sortie erronées.

- Le codage Gray
 - Règle: Pour avoir les codes Gray sur n bits:
 - 1) Pour avoir la première moitié de ces codes:
 - Ecrire les codes sur n-1 bits puis ajouter des 0 à gauche de ces codes
 - 2) Pour avoir la deuxième moitié de ces codes:
 - Ecrire les codes sur n-1 bits dans le sens inverse puis ajouter des 1 à gauche de ces codes

Codage sur 1 bit

Nombre décimal	Code Gray
0	0
1	1

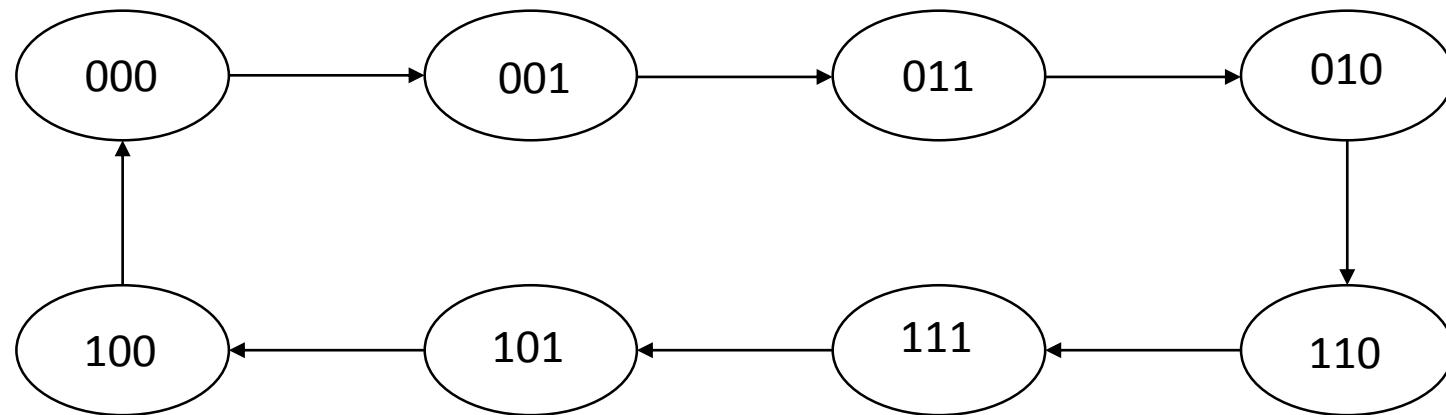
Codage sur 2 bits

Nombre décimal	Code Gray
0	00
1	01
2	11
3	10

Codage sur 3 bits

Nombre décimal	Code Gray
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

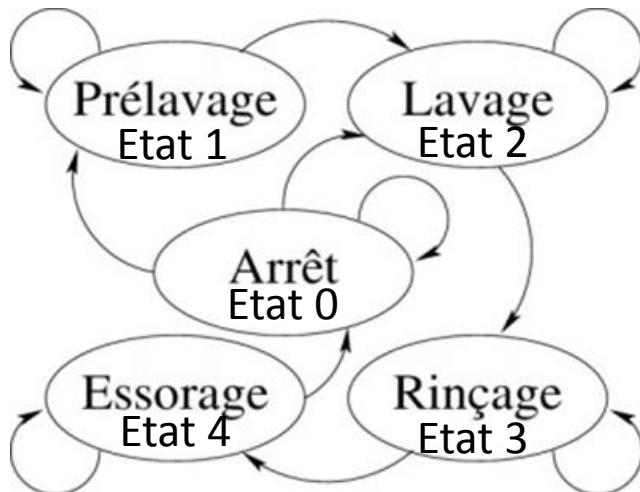
- Exemple: compteur modulo 8



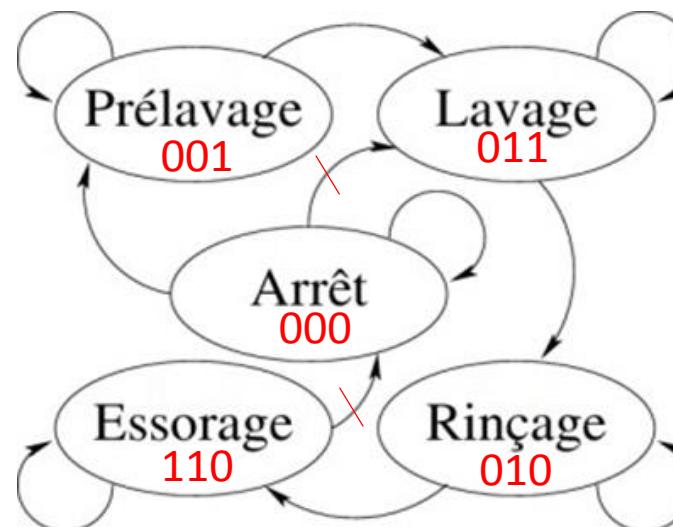
Chaque transition implique le changement d'un seul bit → aucun risque d'état erroné/transitoire

Le codage adjacent (Gray)

- Exemple machine à laver:

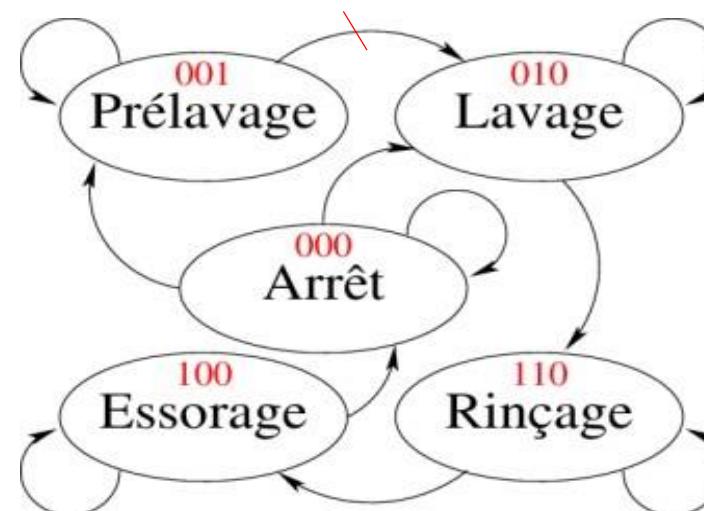


Solution 1



2 transitions impliquent le changement de 2 bits
→ Risque d'états erronés

Solution 2

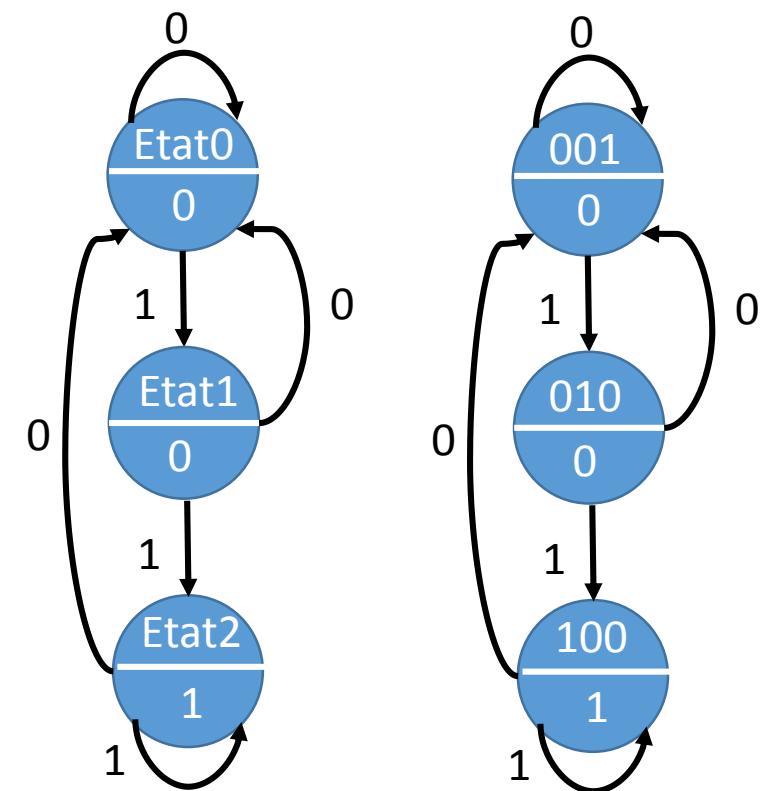


On a choisi une séquence d'états différente de la séquence décimale (on a sauté les valeurs 2, 5 et 6). Cela nous a permis d'avoir une seule transition impliquant le changement de 2 bits
→ Moins de risque

- Le codage Gray a le même coût en ressources que le codage binaire
- Il peut avoir des avantages en évitant des états erronés dans des systèmes asynchrones
 - Cet avantage se voit bien pour les compteurs puisque c'est simple d'associer des valeurs successives aux états (changement d'un seul bit d'état à la fois)
 - Pour les machines séquentielles en général, il est difficile de trouver une séquence d'états qui évitent le changement de plusieurs bits d'états à la fois

- Utilisation d'un nombre de bits égal au nombre d'états.
 - Chaque état est représenté par un mot binaire dont tous les bits sauf un valent 0.
- Donne souvent **les machines les plus simples** à concevoir, mais **plus de bascules** que les codages binaire et gray

- Souvent des équations d'entrées de bascules plus simples
 - Les équations ne dépendent pas de tous les bits de l'état présent
- Parfois plus intéressant en termes de ressources
 - Si les équations des entrées des bascules nécessitent beaucoup moins de portes logiques que celles des codages binaire et gray



Codage ‘one-hot’

Etat présent	Etat suivant		Sortie
	E=0	E=1	
001	001	010	0
010	001	100	0
100	001	100	1



Q2	Q1	Q0	E	D2	D1	D0
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	1	0	0

Étape qui peut être évitée

Equations des entrées des bascules

- Peuvent être obtenues directement à partir du graphe/table d'états

$$D2 = Q2+ = Q1 \cdot E + Q2 \cdot E = E \cdot (Q1 + Q2)$$

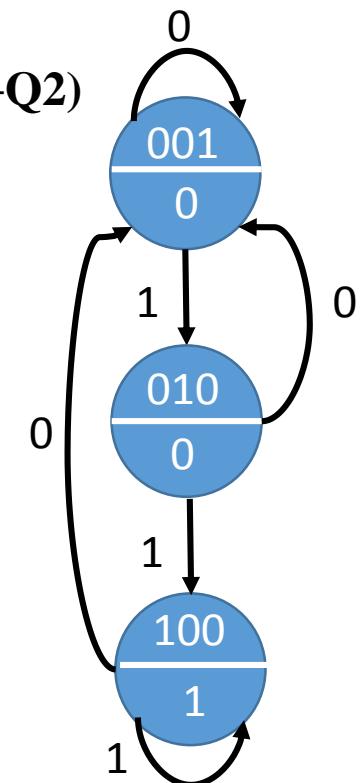
$$D1 = Q1+ = Q0 \cdot E$$

$$D0 = Q0+ = Q0 \cdot \bar{E} + Q1 \cdot \bar{E} + Q2 \cdot \bar{E} = \bar{E} \cdot (Q0 + Q1 + Q2)$$

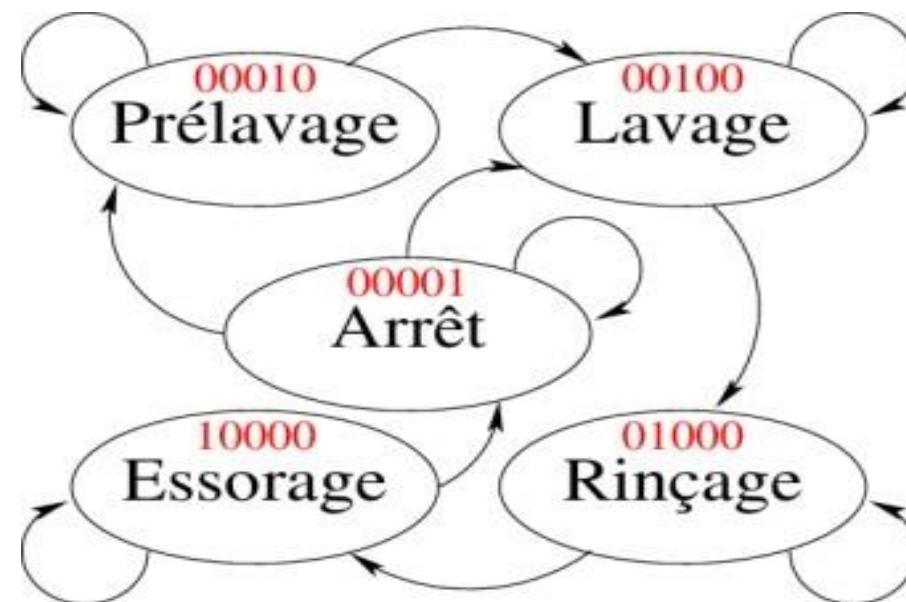
L'équation de la sortie

$$S = Q2$$

- Des équations plus simples
 → moins de ressources matérielles
 → moins de temps de propagation du signal jusqu'aux entrées des bascules → **système plus rapide**



- Exemple machine à laver

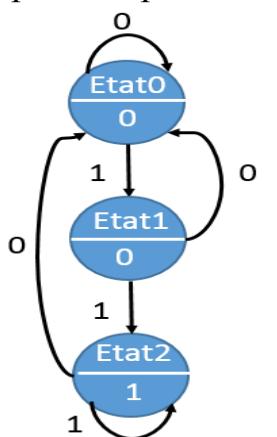


Comparaison entre différents types de codage

	Binaire	Gray	One-hot
Nombre de bascules	$\text{Log}_2(\text{nombre d'états})$	$\text{Log}_2(\text{nombre d'états})$	Nombre d'états
Equations des entrées de bascules			Plus simples car ne dépendent pas de tous les états
Equations des sorties			Plus simples
Rapidité			Plus rapide en raison de la simplicité des équations (moins de portes → moins de temps de propagation)
Possibilité d'états erronés/transitoires	Possible	Plus efficace	possible
Commutation des transistors et consommation d'énergie	Elevée (une transition peut impliquer la commutation de plusieurs bits d'états) + équations d'entrées assez complexes	Moyenne (plusieurs transitions impliquent la commutation d'un seul bit) + équations d'entrées assez complexes	Réduite (tous les transitions impliquent la commutation de deux bits) + équations d'entrées simples
Modification du design (ajout et suppression d'états)	Compliquée puisque chaque état implique plusieurs bascules	Compliquée puisque chaque état implique plusieurs bascules	Plus simple puisque l'ajout ou la suppression d'un état n'implique qu'une seule bascule

- Exemple: le système présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.

Etape1: Graphe d'états



Machine de Moore

Etape2: Table d'états

Etat présent	Etat suivant		Sortie
	E=0	E=1	
Etat0	Etat0	Etat1	0
Etat1	Etat0	Etat2	0
Etat2	Etat0	Etat2	1

Etape3: Codage de la table d'états

Nom état	Code (Q1Q0)
Etat0	00
Etat1	01
Etat2	10

Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1

Sens1: Conception

Etape5: Equations des entrées des bascules et des sorties

Q1	Q0	E	D1	D0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	?	?
1	1	1	?	?

Table de vérité



$$D1 = ?$$

$$D0 = ?$$

$$S = Q1 \cdot \overline{Q0}$$

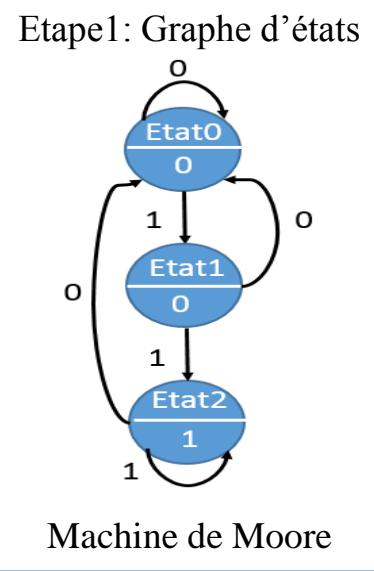
- Comment on gère un état non utilisé (ici l'état 11)?
- Dans le cas de problème où le système se retrouve dans un état inutilisé, dans quel(s) état(s) il doit passer?
- Quel sera l'impact sur les équations des entrées des bascules)?

Etape6: Schéma électrique



Etape4: Choix du type des bascules
On choisit souvent les bascules D
Elles sont souvent celles implémentées dans la plupart des circuits modernes

- Exemple: le système présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.



Etape2: Table d'états

Etat présent	Etat suivant		Sortie
	E=0	E=1	
Etat0	Etat0	Etat1	0
Etat1	Etat0	Etat2	0
Etat2	Etat0	Etat2	1

Etape5: Equations des entrées des bascules et des sorties

Q1	Q0	E	D1	D0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

Table de vérité

Q1Q0		Q1Q0						
E	00	01	11	10	00	01	11	10
0	0	0	0	0	0	0	0	0
1	0	1	1	0	1	0	1	0
1	1	0	0	0	0	0	0	0
1	1	1	0	1	0	1	0	1

$D1 = E \cdot (Q0 \oplus Q1)$

$D0 = E \cdot \overline{Q0} \cdot \overline{Q1}$
 $+ E \cdot Q0 \cdot \overline{Q1}$
 $= E \cdot (Q0 \cdot Q1)$
 $+ Q0 \cdot Q1)$
 $= E \cdot (Q0 \oplus Q1)$

$S = Q1 \cdot \overline{Q0}$

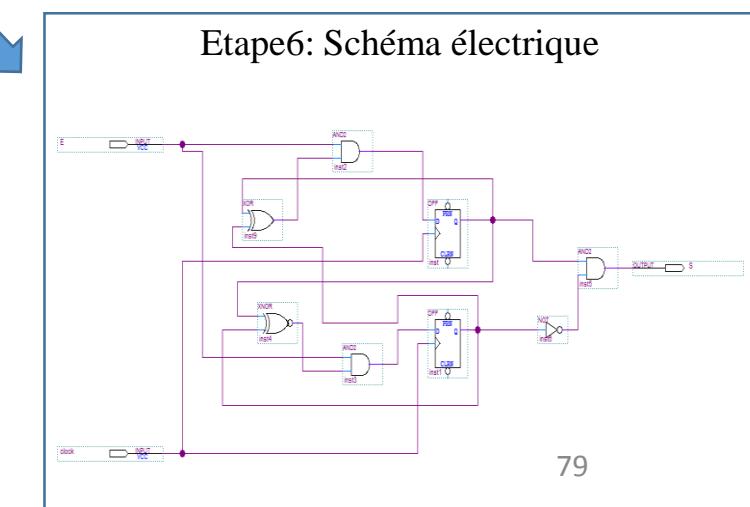
Choix de l'état suivant selon la valeur de l'entrée

Etape3: Codage de la table d'états

Nom état	Code (Q1Q0)
Etat0	00
Etat1	01
Etat2	10

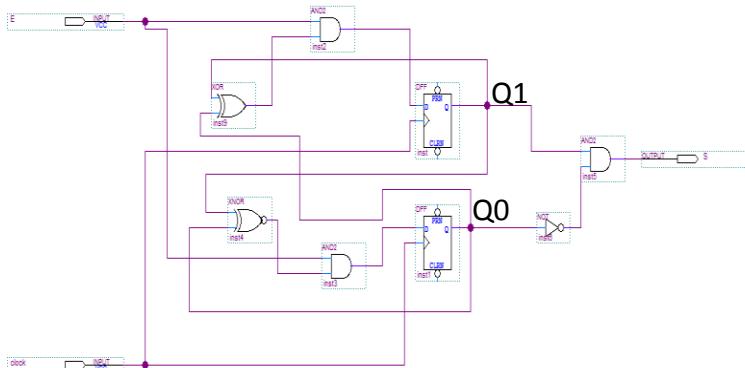
Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1

Etape4: Choix du type des bascules
 On choisit souvent les bascules D
 Elles sont souvent celles implémentées dans la plupart des circuits modernes



- Exemple: le système présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.

Etape1: Schéma électrique



Etape2: Equations des entrées des bascules

$$D_1 = E \cdot (Q_0 \oplus Q_1)$$

$$D_0 = E \cdot \overline{(Q_0 \oplus Q_1)}$$

Etape3: Equations des états suivants et des sorties

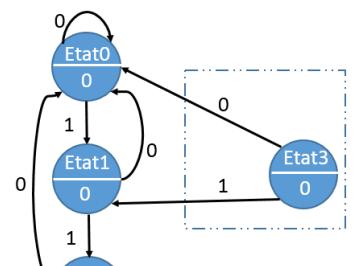
- $Q_+ = D$ (équation caractéristique d'une bascule D)

$$\rightarrow Q_1+ = E \cdot (Q_0 \oplus Q_1)$$

$$Q_0+ = E \cdot \overline{(Q_0 \oplus Q_1)}$$

- $S = Q_1 \cdot \overline{Q_0}$ (à partir du schéma électrique)

Etape 6: Graphe d'états



On obtient le même graphe d'états de l'étape 1 de la conception + les états inutilisés

Etape 5: Nommer les états (et éventuellement les entrées/sorties)

Etat présent	Etat suivant		Sorties
	E=0	E=1	
Etat0	Etat0	Etat1	0
Etat1	Etat0	Etat2	0
Etat2	Etat0	Etat2	1
Etat3	Etat0	Etat1	0

Machine de Moore

Sens2 (inverse): Analyse

Nom état	Code (Q1Q0)
Etat0	00
Etat1	01
Etat2	10
Etat3	11

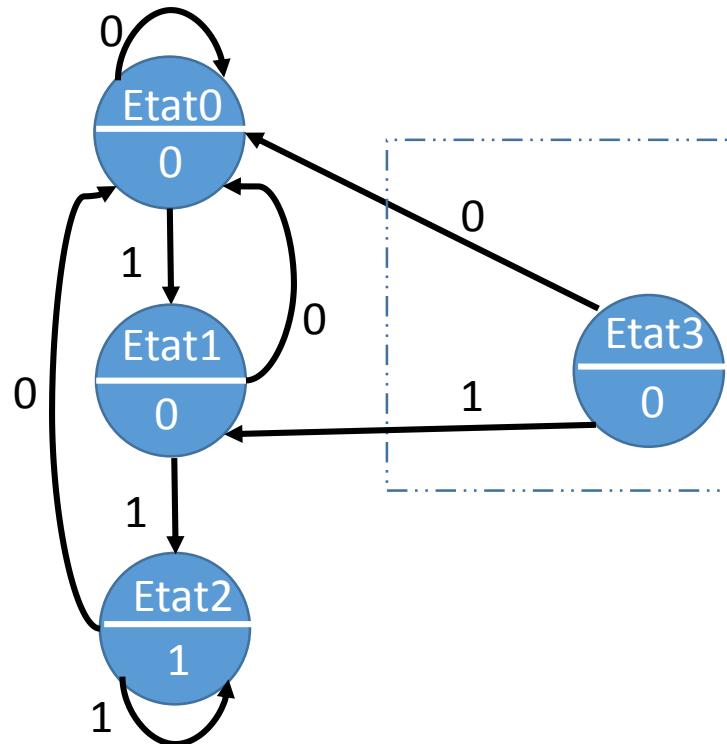
Etape4: Table d'états

Q1	Q0	E	Q1+	Q0+
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1
11	00	01	0

Table de vérité

- Exemple: le système présentera un niveau logique 1 sur sa sortie lorsqu'elle aura reçu au moins deux « 1 » consécutifs sur son entrée série.

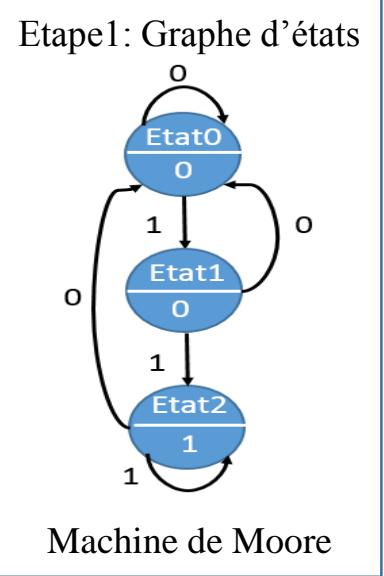


- Dans ce cas, on a choisi de **forcer le retour** dans deux états différents **selon la valeur de l'entrée**
- Mais, ce n'est pas toujours facile de faire le choix si le nombre d'états non utilisés est grand

→ Il y a généralement deux manières de faire:

- 1) Risque minimum: Forcer le retour à l'état initial
- 2) Coût minimum: Les états non utilisés sont marqués « X »

1) Risque minimum: Forcer le retour à l'état initial



Etape2: Table d'états

Etat présent	Etat suivant		Sortie
	E=0	E=1	
Etat0	Etat0	Etat1	0
Etat1	Etat0	Etat2	0
Etat2	Etat0	Etat2	1

Etape5: Equations des entrées des bascules et des sorties

Table de vérité

Q1	Q0	E	D1	D0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	0
1	1	1	0	0

$D1 = E \cdot (Q0 \oplus Q1)$

E	Q1Q0			
	00	01	11	10
0	0	0	0	0
1	0	1	0	1

$D0 = E \cdot \overline{Q0} \cdot \overline{Q1}$

E	Q1Q0			
	00	01	11	10
0	0	0	0	0
1	1	0	0	0

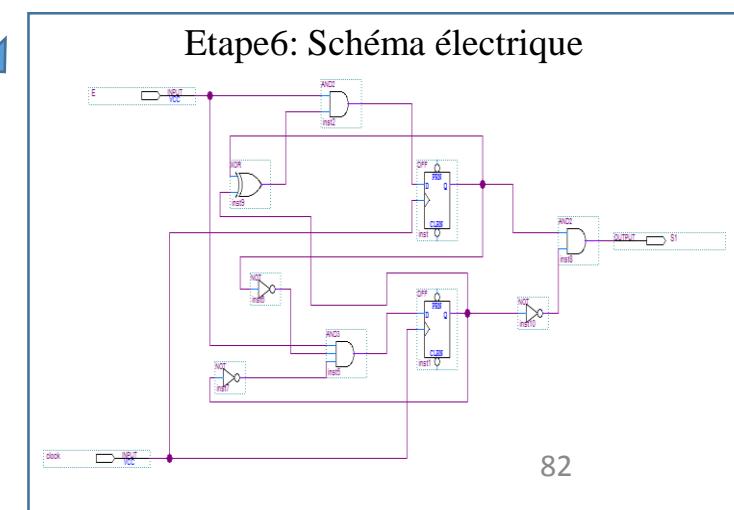
$S = Q1 \cdot \overline{Q0}$

Etape3: Codage de la table d'états

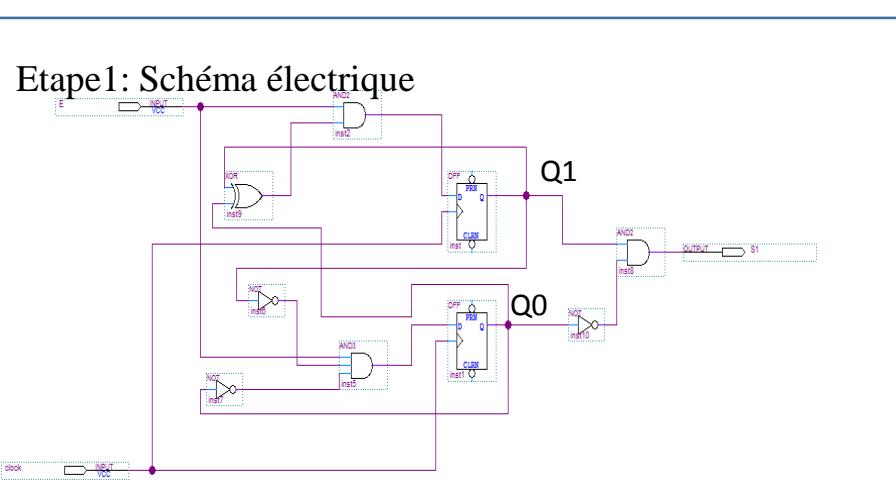
Nom état	Code (Q1Q0)
Etat0	00
Etat1	01
Etat2	10

Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1

Etape4: Choix du type des bascules
On choisit souvent les bascules D
Elles sont souvent celles implémentées dans la plupart des circuits modernes



1) Risque minimum: Forcer le retour à l'état initial



Etape2: Equations des entrées des bascules

$$D_1 = E \cdot (Q_0 \oplus Q_1)$$

$$D_0 = E \cdot \overline{Q_0} \cdot \overline{Q_1}$$

Etape3: Equations des états suivants et des sorties

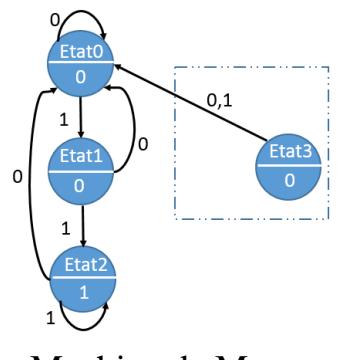
- $Q_+ = D$ (équation caractéristique d'une bascule D)

$$\rightarrow Q_1+ = E \cdot (Q_0 \oplus Q_1)$$

$$Q_0+ = E \cdot \overline{Q_0} \cdot \overline{Q_1}$$

- $S = Q_1 \cdot \overline{Q_0}$ (à partir du schéma électrique)

Etape 6: Graphe d'états



On obtient le même graphe d'états de l'étape 1 de la conception + les états inutilisés

Etape 5: Nommer les états (et éventuellement les entrées/sorties)

Etat présent	Etat suivant		Sorties
	E=0	E=1	
Etat0	Etat0	Etat1	0
Etat1	Etat0	Etat2	0
Etat2	Etat0	Etat2	1
Etat3	Etat0	Etat0	0

Nom état	Code (Q1Q0)
Etat0	00
Etat1	01
Etat2	10
Etat3	11

Etape4: Table d'états

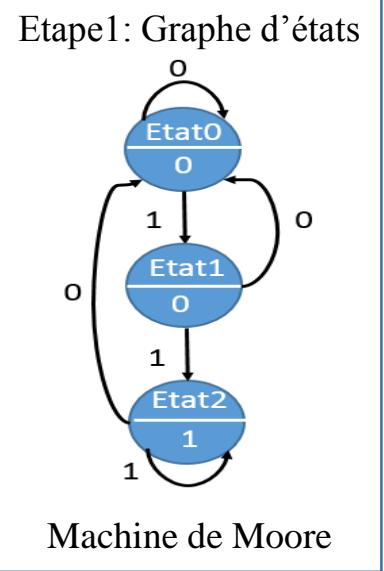
Q1	Q0	E	Q1+	Q0+
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	0
1	1	1	0	0

Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1
11	00	00	0

Table de vérité

Sens2 (inverse): Analyse

2) Coût minimum: Les états non utilisés sont marqués « X »



Etape2: Table d'états

Etat présent	Etat suivant		Sortie
	E=0	E=1	
Etat0	Etat0	Etat1	0
Etat1	Etat0	Etat2	0
Etat2	Etat0	Etat2	1

Etape5: Equations des entrées des bascules et des sorties

Table de vérité

Q1	Q0	E	D1	D0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	X	X
1	1	1	X	X

$D1 = E \cdot (Q0 + Q1)$

Equations plus simples
→ Diminuer le coût

E	Q1Q0			
	00	01	11	10
0	0	0	x	0
1	0	1	x	1

$D0 = E \cdot \overline{Q0} \cdot \overline{Q1}$

E	Q1Q0			
	00	01	11	10
0	0	0	x	0
1	1	0	x	0

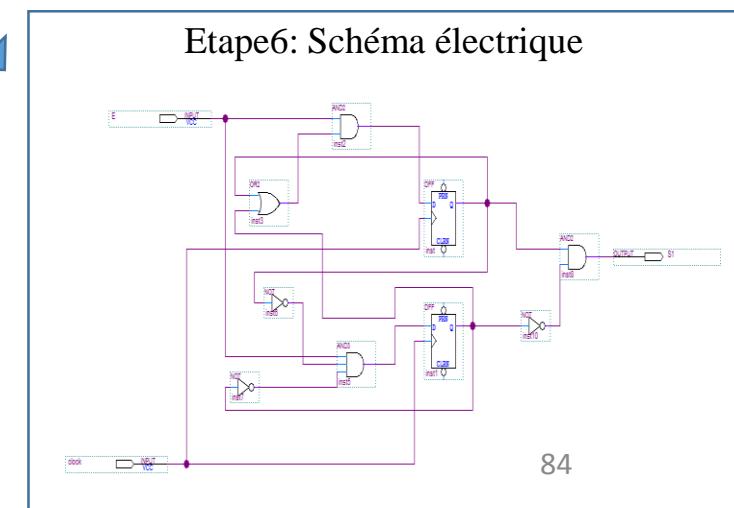
$S = Q1 \cdot \overline{Q0}$

Etape3: Codage de la table d'états

Nom état	Code (Q1Q0)
Etat0	00
Etat1	01
Etat2	10

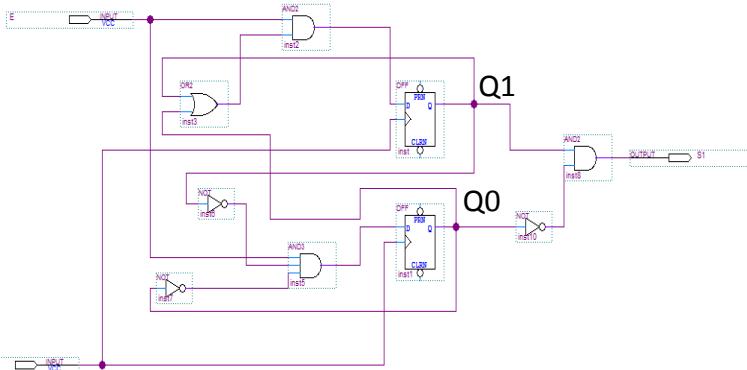
Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1

Etape4: Choix du type des bascules
On choisit souvent les bascules D
Elles sont souvent celles implémentées dans la plupart des circuits modernes



2) Coût minimum: Les états non utilisés sont marqués « X »

Etape1: Schéma électrique



Etape2: Equations des entrées des bascules

$$D1 = E \cdot (Q0 + Q1)$$

$$D0 = E \cdot \overline{Q0} \cdot \overline{Q1}$$

Etape3: Equations des états suivants et des sorties

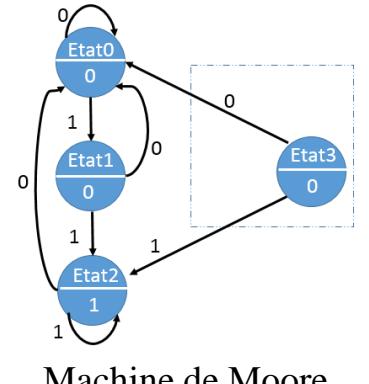
- $Q+ = D$ (équation caractéristique d'une bascule D)

$$\rightarrow Q1+ = E \cdot (Q0 + Q1)$$

$$Q0+ = E \cdot \overline{Q0} \cdot \overline{Q1}$$

- $S = Q1 \cdot \overline{Q0}$ (à partir du schéma électrique)

Etape 6: Graphe d'états



On obtient le même graphe d'états de l'étape 1 de la conception + les états inutilisés, parfois cela entraîne un comportement bizarre (passer de l'état3 à l'état2)

Etape 5: Nommer les états (et éventuellement les entrées/sorties)

Etat présent	Etat suivant		Sorties
	E=0	E=1	
Etat0	Etat0	Etat1	0
Etat1	Etat0	Etat2	0
Etat2	Etat0	Etat2	1
Etat3	Etat0	Etat2	0

Nom état	Code (Q1Q0)
Etat0	00
Etat1	01
Etat2	10
Etat3	11

Sens2 (inverse): Analyse

Etape4: Table d'états

Q1	Q0	E	Q1+ Q0+
0	0	0	0 0
0	0	1	0 1
0	1	0	0 0
0	1	1	1 0
1	0	0	0 0
1	0	1	1 0
1	1	0	0 0
1	1	1	1 1

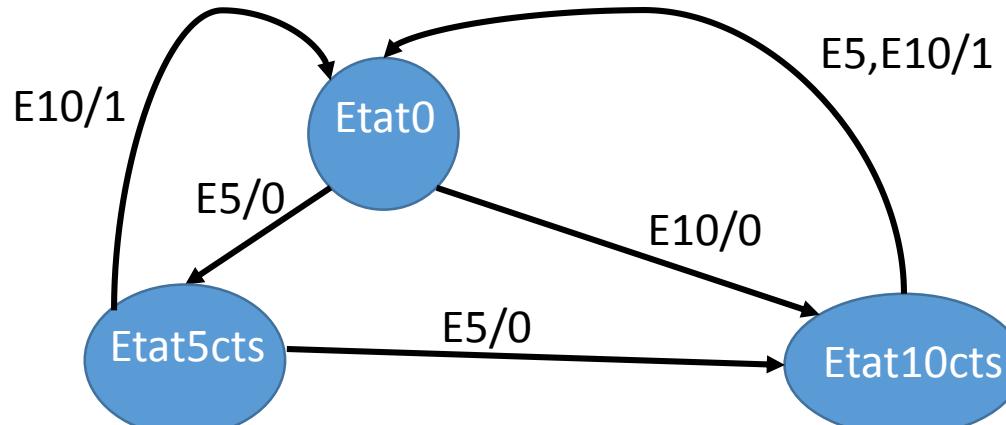
Etat présent	Etat suivant		Sortie
	E=0	E=1	
00	00	01	0
01	00	10	0
10	00	10	1
11	00	10	0

Table de vérité

- Coût minimum: Les états non utilisés sont marqués « X »
 - Risques: 1) Un comportement qui ne correspond pas aux fonctionnalités du système
 - 2) Les états non utilisés risquent de former un cycle duquel on peut pas sortir
- Dans ce module, on ne considère que la solution qui force le retour à l'état initial

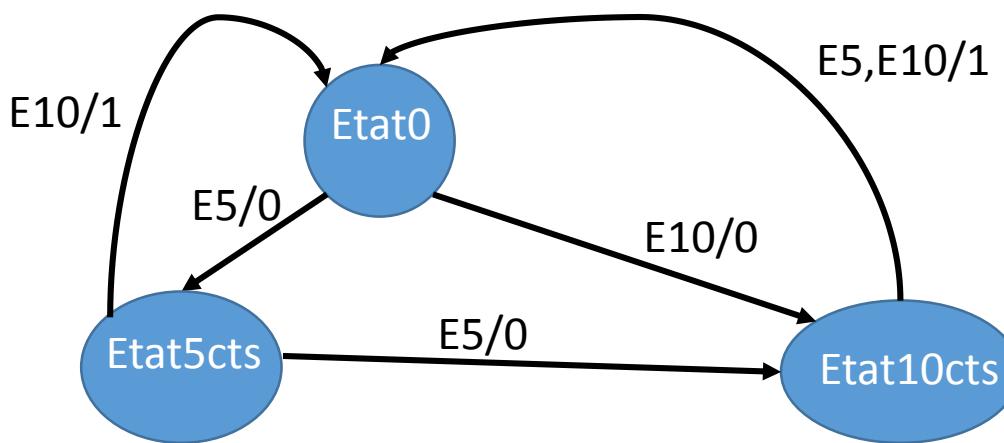
- Concevez une machine séquentielle (Mealy) qui doit automatiser un distributeur de café à 15cts . Les pièces acceptées sont de 5 cts (E5) et 10 cts (E10). Les autres pièces sont retournées. La sortie « S » passera à 1 pour lancer la distribution du café, dès que l'utilisateur aura inséré une somme \geq à 15 cts.
- Le système de rendu de monnaie n'est pas à gérer.
- Il y a deux valeurs de l'entrée « E »: E5 (5cts) et E10 (10cts)

- Concevez une machine séquentielle (Mealy) qui doit automatiser un distributeur de café à 15cts . Les pièces acceptées sont de 5 cts (E5) et 10 cts (E10). Les autres pièces sont retournées . La sortie « S » passera à 1 pour lancer la distribution du café, dès que l'utilisateur aura inséré une somme \geq à 15 cts.
- Le système de rendu de monnaie n'est pas à gérer.
- Il y a deux valeurs de l'entrée « E »: E5 (5cts) et E10 (10cts)



Etape 1: Graphe d'états

- Concevez une machine séquentielle (Mealy) qui doit automatiser un distributeur de café à 15cts . Les pièces acceptées sont de 5 cts (E5) et 10 cts (E10). Les autres pièces sont retournées . La sortie « S » passera à 1 pour lancer la distribution du café , dès que l'utilisateur aura inséré une somme ≥ 15 cts.
- Le système de rendu de monnaie n'est pas à gérer.



Etape1: Graphe d'états

Etat présent	Etat suivant		Sortie	
	E=E5	E=E10	E=E5	E=E10
Etat0	Etat5cts	Etat10cts	0	0
Etat5cts	Etat10cts	Etat0	0	1
Etat10cts	Etat0	Etat0	1	1

Etape2: Table d'états

- Concevez une machine séquentielle (Mealy) qui doit automatiser un distributeur de café à 15cts . Les pièces acceptées sont de 5 cts (E5) et 10 cts (E10). Les autres pièces sont retournées . La sortie s passera à 1 pour lancer la distribution du café , dès que l'utilisateur aura inséré une somme \geq à 15 cts.
- Le système de rendu de monnaie n'est pas à gérer.

Etat présent	Etat suivant		Sortie	
	E=E5	E=E10	E=E5	E=E10
Etat0	Etat5cts	Etat10cts	0	0
Etat5cts	Etat10cts	Etat0	0	1
Etat10cts	Etat0	Etat0	1	1

Etape2: Table d'états

Nom état	Code (Q1Q0)
Etat0	00
Etat5cts	01
Etat10cts	10

Codage des états

Nom de l'entrée	Code (E)
E5	0
E10	1

Codage des entrées

Etat présent	Etat suivant		Sortie	
	E=0	E=1	E=0	E=1
00	01	10	0	0
01	10	00	0	1
10	00	00	1	1

Etape3: Codage de la table d'états

- Concevez une machine séquentielle (Mealy) qui doit automatiser un distributeur de café à 15cts . Les pièces acceptées sont de 5 cts (E5) et 10 cts (E10). Les autres pièces sont retournées . La sortie s passera à 1 pour lancer la distribution du café , dès que l'utilisateur aura inséré une somme \geq à 15 cts.
- Le système de rendu de monnaie n'est pas à gérer.

Etat présent	Etat suivant		Sortie		
	E=0	E=1	E=0	E=1	
00	01	10	0	0	
01	10	00	0	1	
10	00	00	1	1	

Etape3: Codage de la table d'états

- $Q+ = D$ (équation caractéristique d'une bascule D)
- $\rightarrow D1 = Q1+$ et $D0 = Q0+$, $Q1+$ et $Q0+$ sont donnés par l'état suivant dans la table d'états

Q1	Q0	E	D1	D0	S
0	0	0	0	1	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	0	0	1
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	0	0	0
1	1	1	0	0	0

Table de vérité

$$D1 = \overline{Q1} \cdot (Q0 \oplus E)$$

$$D0 = \overline{Q1} \cdot \overline{Q0} \cdot \overline{E}$$

E	Q1Q0			
	00	01	11	10
0	0	0	0	1
1	0	1	0	1

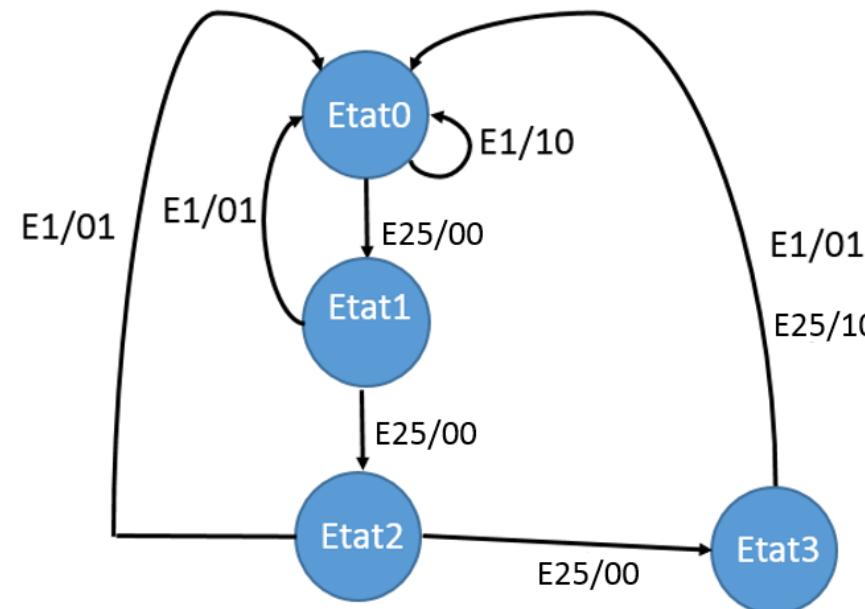
Tableau de karnaugh pour S

$$S = Q1 \cdot \overline{Q0} + E \cdot \overline{Q1} \cdot Q0$$

Etape4-5: Choix du type des bascules + Equations des entrées des bascules et des sorties

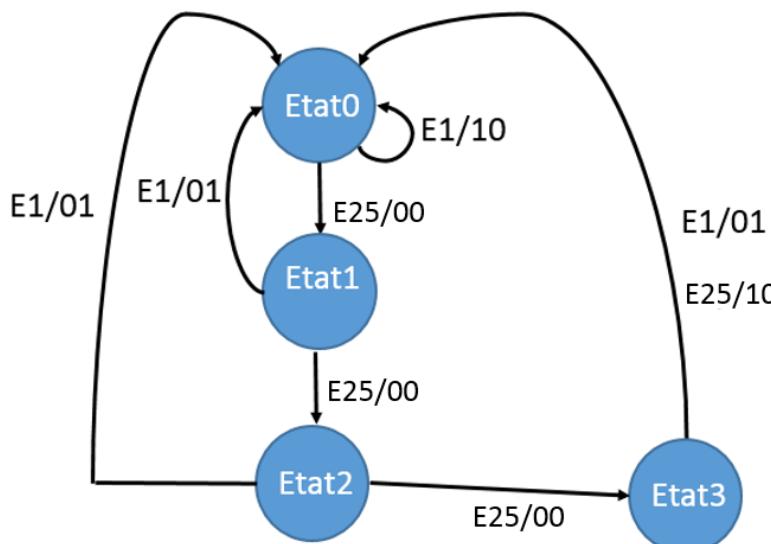
- Concevez une machine séquentielle de Mealy qui distribue une boisson quelconque.
- Une bouteille de cette boisson coûte 1\$. La machine accepte des 25¢ et des 1\$.
- Dès que le client entre exactement 1\$, une bouteille est lâchée.
- Si plus que 1\$ est inséré dans la machine, la machine ne donne pas la boisson, relâche tout l'argent (la machine est incapable de rendre la monnaie) et retourne à son état initial.
- Il y a deux valeurs de l'entrée « E »: E25 (25¢) et E1 (1\$).
- Il y a deux sorties : S_b (Lâcher une bouteille) et S_a (lâcher l'argent) .

- Concevez une machine séquentielle de Mealy qui distribue une boisson quelconque.
- Une bouteille de cette boisson coutre 1\$. La machine accepte des 25¢ et des 1\$.
- Dès que le client entre exactement 1\$, une bouteille est lâchée
- Si plus que 1\$ est inséré dans la machine , la machine relâche cet argent et retourne à son état initial.
- Il y a deux valeurs de l'entrée « E »: E25 (25¢) et E1 (1\$).
- Il y a deux sorties : Sb (Lâcher une bouteille) et Sa (lâcher l'argent) .



Etape 1: Graphe d'états

- Concevez une machine séquentielle qui distribue une boisson quelconque.
- Une bouteille de cette boisson coute 1\$. La machine accepte des 25¢ et des 1\$.
- Dès que le client entre exactement 1\$, une bouteille est lâchée
- Si plus que 1\$ est inséré dans la machine , la machine relâche cet argent et retourne à son état initial.
- Il y a deux entrées : E25 (25¢) et E1 (1\$).
- Il y a deux sorties : Sb (Lâcher une bouteille) et Sa (lâcher l'argent) .



Etape1: Graphe d'états

Etat présent	Etat suivant		Sorties			
	E=E25	E=E1	E=E25		E=E1	
			Sb	Sa	Sb	Sa
Etat0	Etat1	Etat0	0	0	1	0
Etat1	Etat2	Etat0	0	0	0	1
Etat2	Etat3	Etat0	0	0	0	1
Etat3	Etat0	Etat0	1	0	0	1

Etape2: Table d'états

- Concevez une machine séquentielle qui distribue une boisson quelconque.
- Une bouteille de cette boisson coute 1\$. La machine accepte des 25¢ et des 1\$.
- Dès que le client entre exactement 1\$, une bouteille est lâchée
- Si plus que 1\$ est inséré dans la machine , la machine relâche cet argent et retourne à son état initial.
- Il y a deux entrées : E25 (25¢) et E1 (1\$).
- Il y a deux sorties : Sb (Lâcher une bouteille) et Sa (lâcher l'argent) .

Etat présent	Etat suivant		Sorties			
	E=E25	E=E1	E=E25		E=E1	
			Sb	Sa	Sb	Sa
Etat0	Etat1	Etat0	0	0	1	0
Etat1	Etat2	Etat0	0	0	0	1
Etat2	Etat3	Etat0	0	0	0	1
Etat3	Etat0	Etat0	1	0	0	1

Etape2: Table d'états

SYS2044 Systèmes – 2A-S2

Etat présent	Etat suivant		Sorties			
	E=0	E=1	E=0		E=1	
			Sb	Sa	Sb	Sa
00	01	00	0	0	1	0
01	10	00	0	0	0	1
10	11	00	0	0	0	1
11	00	00	1	0	0	1

Etape3: Codage de la table d'états

- Concevez une machine séquentielle qui distribue une boisson quelconque.
- Une bouteille de cette boisson cout 1\$. La machine accepte des 25¢ et des 1\$.
- Dès que le client entre exactement 1\$, une bouteille est lâchée
- Si plus que 1\$ est inséré dans la machine , la machine relâche cet argent et retourne à son état initial.
- Il y a deux entrées : E25 (25¢) et E1 (1\$).
- Il y a deux sorties : Sb (Lâcher une bouteille) et Sa (lâcher l'argent) .

Etat présent	Etat suivant		Sorties			
	E=0	E=1	E=0		E=1	
			Sb	Sa	Sb	Sa
00	01	00	0	0	1	0
01	10	00	0	0	0	1
10	11	00	0	0	0	1
11	00	00	1	0	0	1

Etape3: Codage de la table d'états

- $Q+ = D$ (équation caractéristique d'une bascule D)
- $\rightarrow D1 = Q1 +$ et $D0 = Q0 +$

Q1	Q0	E	D1	D0	Sb	Sa
0	0	0	0	1	0	0
0	0	1	0	0	1	0
0	1	0	1	0	0	0
0	1	1	0	0	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$$D1 = \overline{E} \cdot (Q1 \oplus Q0)$$

$$D0 = \overline{E} \cdot \overline{Q0}$$

		Q1Q0			
E	00	01	11	10	
0	0	0	1	0	
1	1	0	0	0	

Tableau de karnaugh pour Sb

$$Sb = E \cdot Q1 \cdot Q0 + E \cdot \overline{Q1} \cdot \overline{Q0}$$

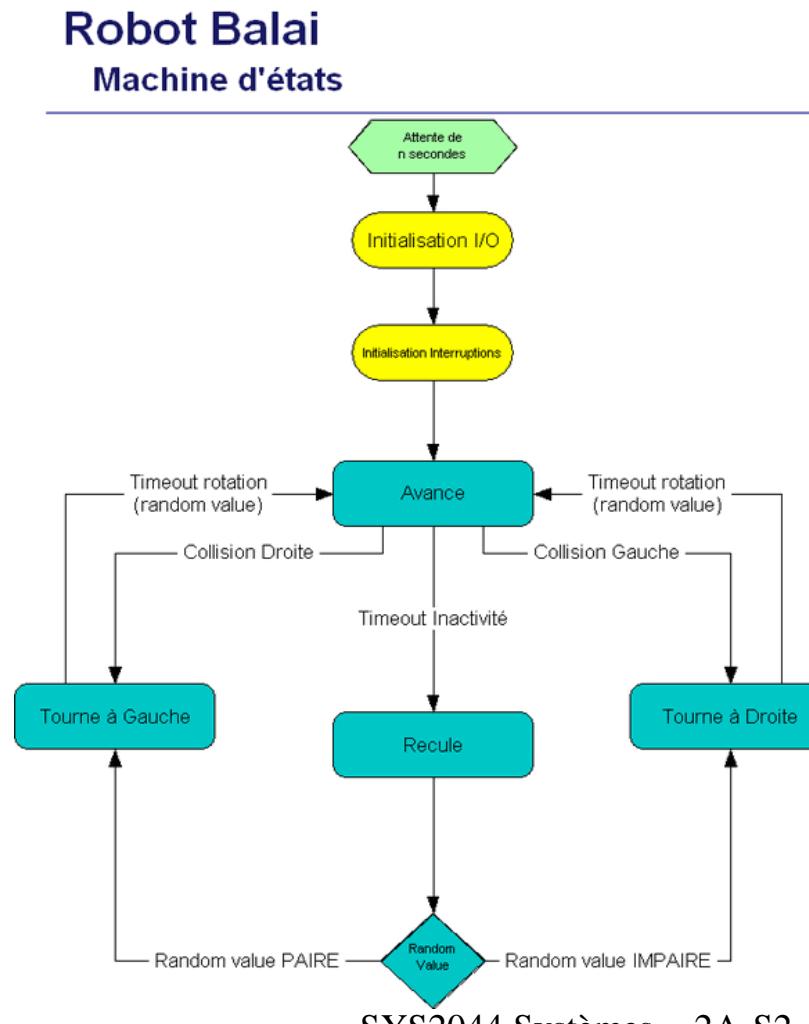
		Q1Q0			
E	00	01	11	10	
0	0	0	0	0	
1	0	1	1	1	

Tableau de karnaugh pour Sa

$$Sa = E \cdot (Q1 + Q0)$$

Etape4-5: Choix du type des bascules + Equations des entrées des bascules et des sorties

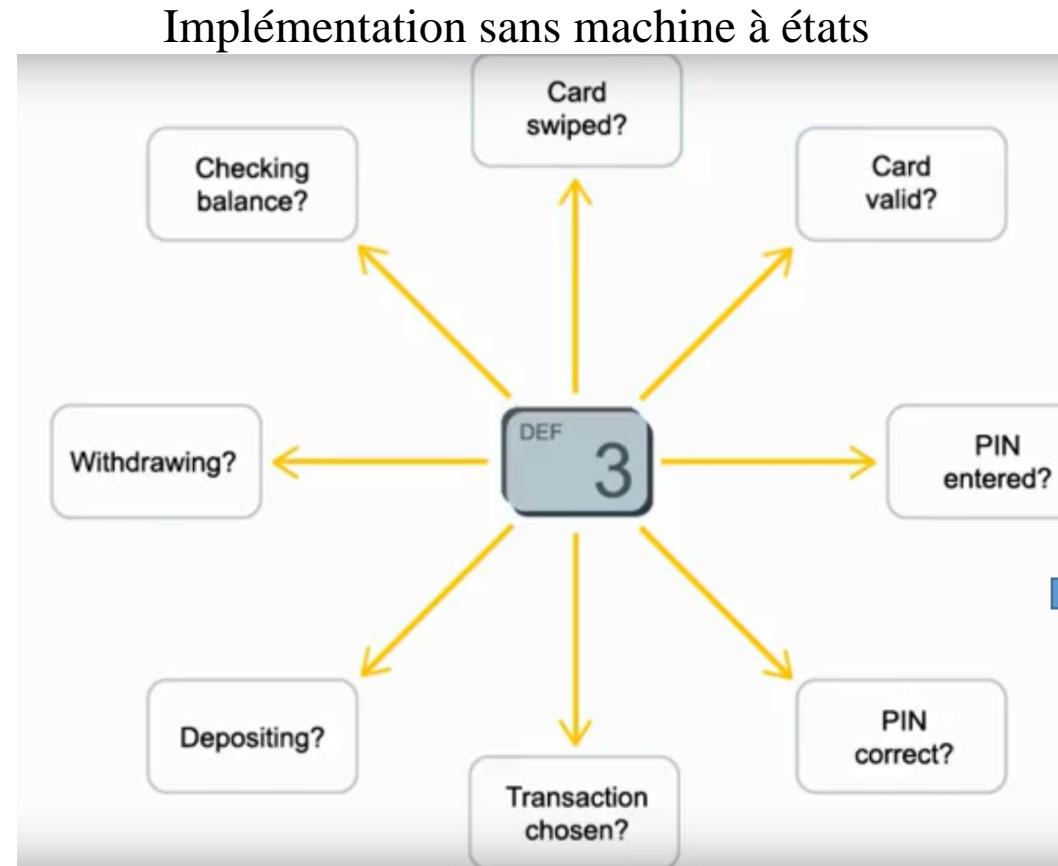
- 1) Modélisation graphique d'un système ayant un nombre défini d'états ou de modes de fonctionnement
 - Permet d'avoir une version plus lisible du cahier de charges du système



2) Une implémentation plus efficace du système



Exemple: Un distributeur de billets



A chaque appui sur un bouton, on vérifie plusieurs possibilités qui mènent à d'autres vérifications imbriquées

```
if (card_swiped){  
    if (pin_entered){  
        if (pin_validated){  
            if (transaction_selected){  
                if(cash_withdrawal){  
                    if(amount_selected){  
                        confirm_amount(a,b,c);  
                    }  
                    else {  
                        continue_entry(d,e,f);  
                    }  
                }  
                else if (depositing_cash){  
                    if(amount_selected){  
                        confirm_amount(g,h,i);  
                    }  
                    else {  
                        continue_entry(j,k,l);  
                    }  
                }  
                else if (depositing_check){  
                    if(amount_selected){  
                        confirm_amount(m,n,o);  
                    }  
                }  
            }  
        }  
    }  
}
```

Un code avec plusieurs if imbriqués → un code complexe et un temps d'exécution élevé

2) Une implémentation plus efficace du système

- Solution: organiser le code selon les états/modes possibles



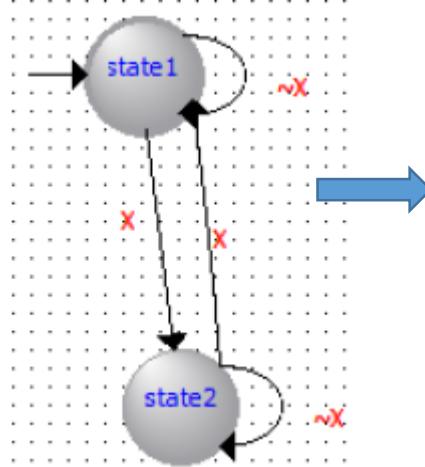
Exemple: Un distributeur de billets



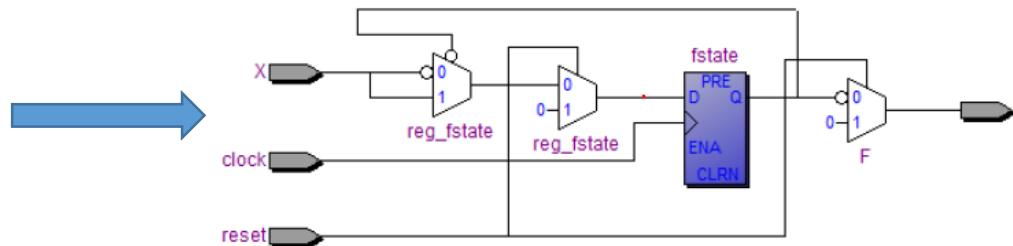
3) Une génération automatique du code (logiciel ou matériel)

- Le formalisme mathématique derrière les machines à états facilite la génération du code

- Exemple 1: génération du code matériel VHDL à partir d'une machine à états dans le logiciel Quartus II d'Altera
 - Cette option sera testée dans les TPs de fin de semestre

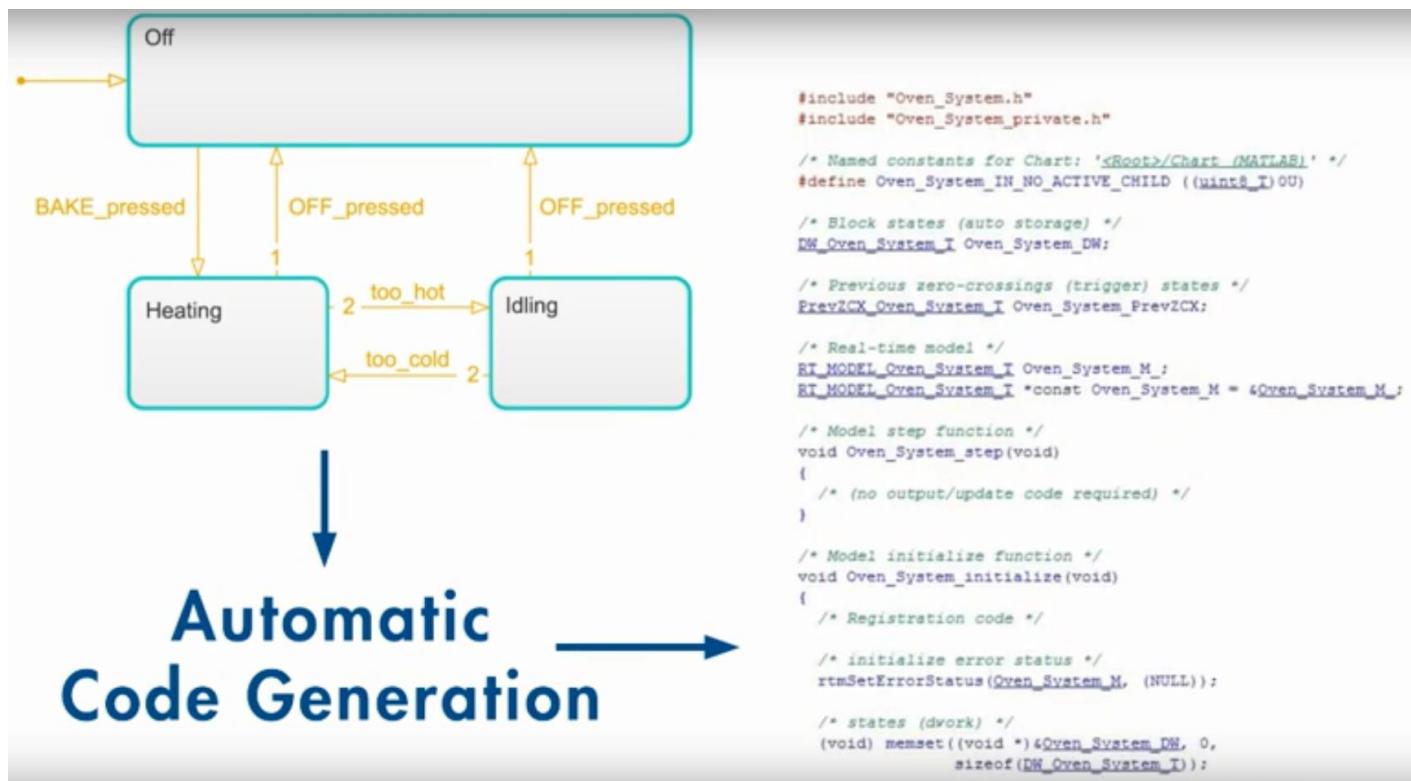


```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY SM3 IS
PORT (
reset : IN STD_LOGIC := '0';
clock : IN STD_LOGIC;
X : IN STD_LOGIC := '0';
F : OUT STD_LOGIC
);
END SM3;
ARCHITECTURE BEHAVIOR OF
SM3 IS
TYPE type_fstate IS (state1,state2);
SIGNAL fstate : type_fstate;
SIGNAL reg_fstate : type_fstate;
BEGIN
PROCESS (clock,reg_fstate)
BEGIN
IF (clock='1' AND clock'event)
THEN
fstate <= reg_fstate;
END IF;
END PROCESS;
PROCESS (fstate,reset,X)
END BEHAVIOR;
```



3) Une génération automatique du code (logiciel ou matériel)

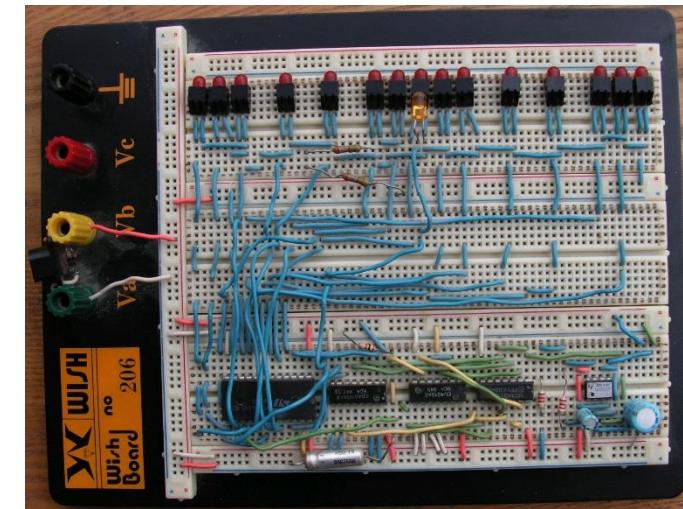
- Le formalisme mathématique derrière les machines à états facilite la génération du code
 - Exemple2: génération du code logiciel en langage C à partir d'une machine à états dans le logiciel Embedded Coder de Matlab



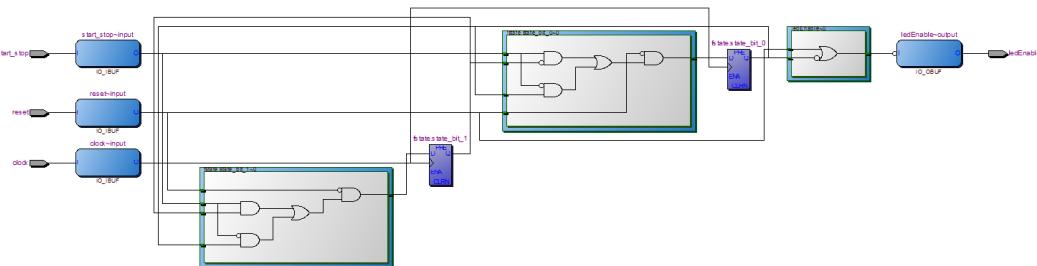
- Différentes possibilités d'utilisation
 - I. Implémentation matérielle (portes logiques et bascules)
 - En faisant un montage électronique
 - En utilisant une carte FPGA (charger un schéma électrique ou un code matériel sur la carte)
 - II. Implémentation logicielle:
 - Sur un processeur (exemple: gestion des modes d'un jeu sur ordinateur)
 - Sur un microcontrôleur (exemple: gestion des modes d'un robot implémenté sur Arduino)

I. Implémentation matérielle (portes logiques et bascules)

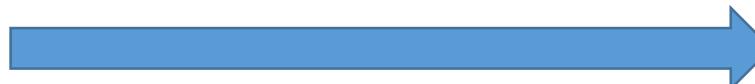
- En faisant un montage électronique
 - Beaucoup de branchements à la main
 - Non adapté pour les grands circuits



- En utilisant une carte FPGA (charger un schéma électrique ou un code matériel sur la carte)
 - C'est la méthode qui sera utilisée dans le TP

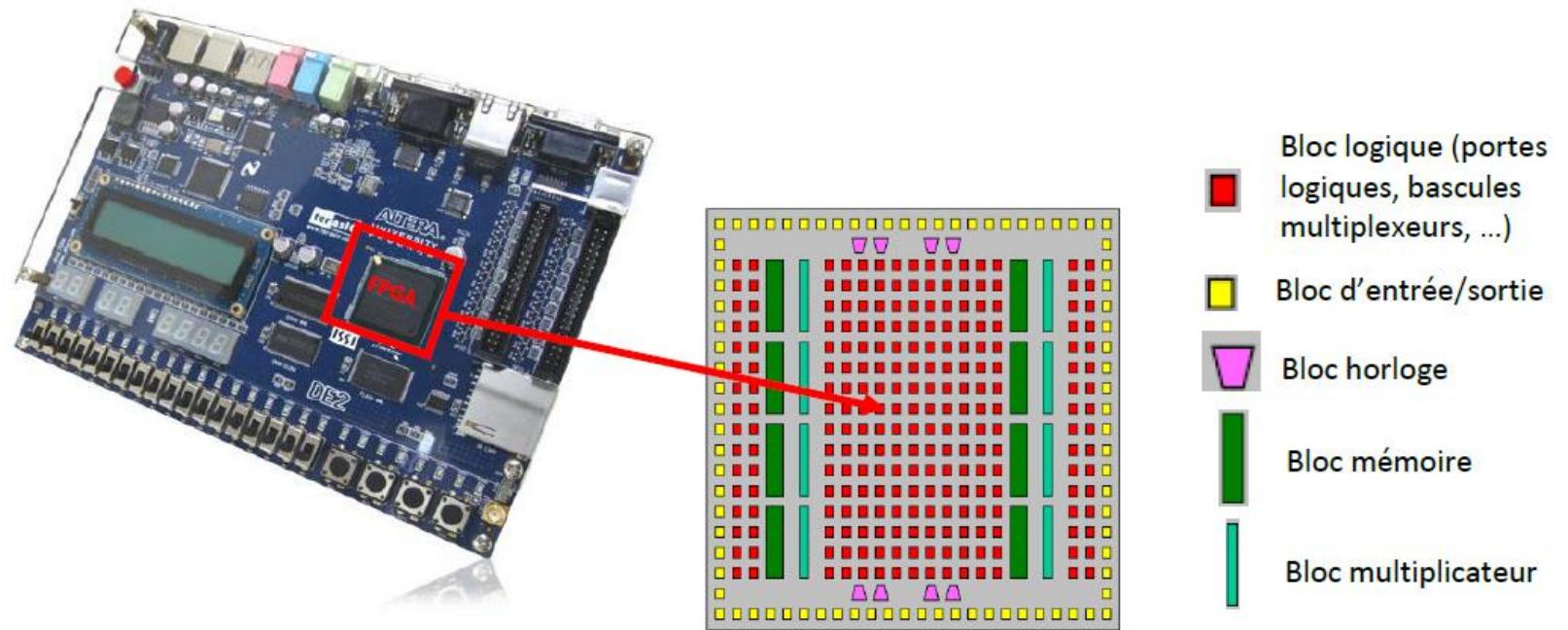


Charger par un câble USB sur la carte



I. Implémentation matérielle

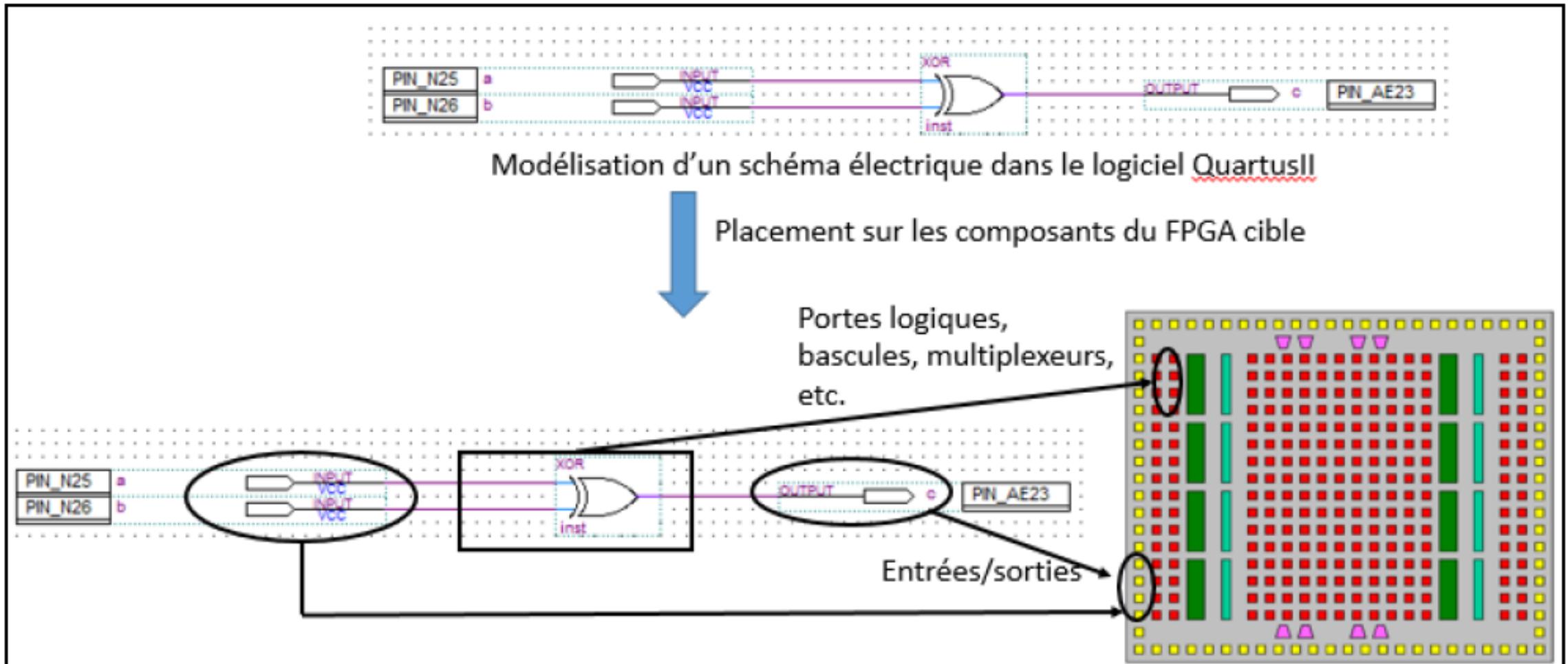
- Circuits programmables (Exemple FPGA)



Le FPGA (Field-Programmable Gate Array) est une puce contenant une matrice de blocs logiques (portes logiques, bascules, multiplexeurs, etc.), mémoires, entrées/sorties et autres. C'est pour cette raison on utilise le terme « Gate Array ». Grâce au grand nombre de blocs contenus dans les FPGAs modernes (des millions de blocs), ils peuvent être utilisés pour implémenter un grand nombre de fonctionnalités. Cette implémentation se fait à l'aide d'un programme logiciel qui permet de charger une configuration donnée sur la puce FPGA. Grâce à cette méthode, on n'a plus besoin d'utiliser des composants logiques et mémoires discrets avec des câblages compliqués, tout est contenu dans une puce de quelque cm².

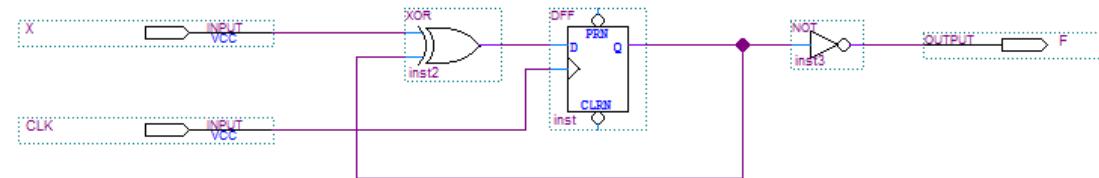
I. Implémentation matérielle

- Circuits programmables (Exemple FPGA)



I. Implémentation matérielle

- 1) Elaboration du schéma électrique dans un logiciel qui permet l'implémentation sur une carte électronique



- 2) Ecriture des équations des bascules et des sorties dans un logiciel qui permet l'implémentation sur une carte électronique

```
.....
PROCESS (CLK)
BEGIN
IF (RISING_EDGE (CLK)) THEN
    Q <= D;
END IF;
END PROCESS;

D <= X XOR Q;

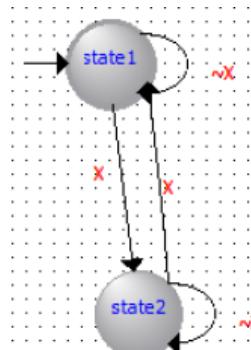
F <= NOT (Q);
.....
```

Sur front de l'horloge
Q reçoit la valeur de D

L'équation de $D=X \oplus Q$

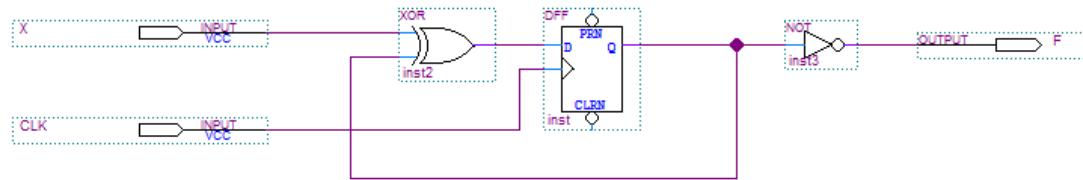
L'équation de la sortie $F=\overline{Q}$

- 3) Modélisation de la machine d'états dans un logiciel qui permet l'implémentation sur une carte électronique



I. Implémentation matérielle

- 1) Elaboration du schéma électrique dans un logiciel qui permet l'implémentation sur une carte électronique



Charger par un câble USB sur la carte



I. Implémentation matérielle

- 2) Ecriture des équations des bascules et des sorties dans un logiciel qui permet l'implémentation sur une carte électronique

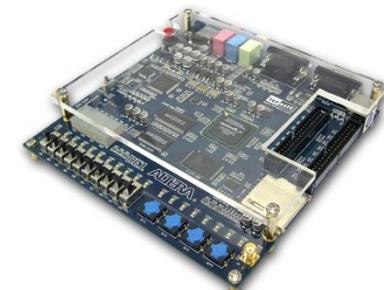
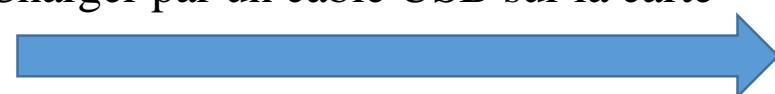
```
.....  
  
PROCESS (CLK)  
BEGIN  
IF (RISING_EDGE(CLK)) THEN  
    Q <= D;  
END IF;  
END PROCESS;  
  
D <= X XOR Q;  
  
F <= NOT(Q);  
.....
```

Sur front de l'horloge
Q reçoit la valeur de D

L'équation de $D=X \oplus Q$

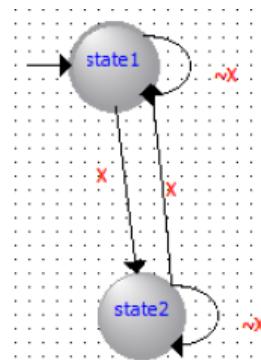
L'équation de la sortie $F=\bar{Q}$

Charger par un câble USB sur la carte

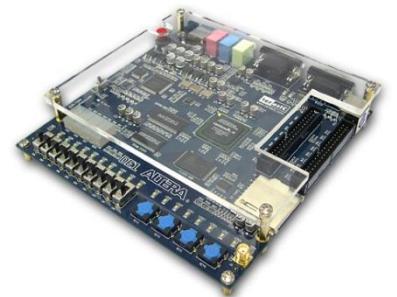


I. Implémentation matérielle

- 3) Modélisation de la machine d'états dans un logiciel qui permet l'implémentation sur une carte électronique



Charger par un câble USB sur la carte



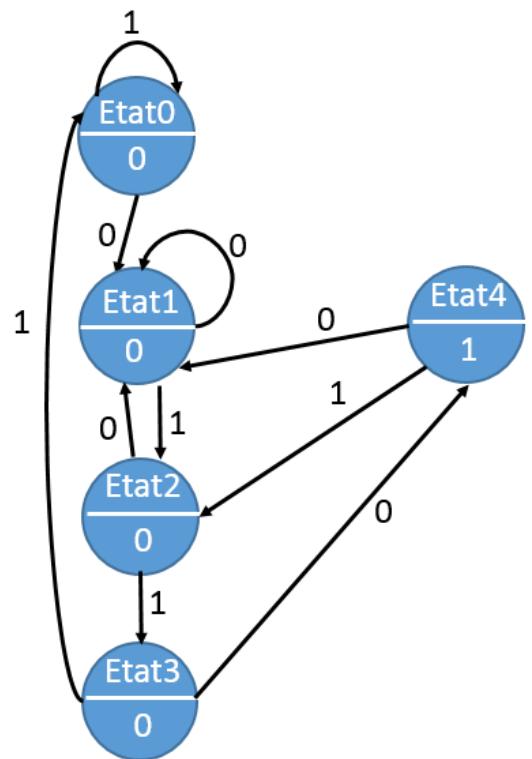
II. Implémentation logicielle

- 1) Ecriture à la main du code de la machine à états
- 2) Modélisation de la machine à états dans un logiciel qui permet la génération automatique de son code

II. Implémentation logicielle

1) Ecriture à la main du code de la machine à états

- Exemple: Concevez un système capable de détecter la séquence 0-1-1-0 sur son unique entrée (x). Quand la séquence se produit, il devrait émettre un 1 sur son unique sortie (y). Notez que dans la séquence 0-1-1-0-1-1-0, la séquence se produit deux fois.



```
.....
enum Etat {Etat0, Etat1, Etat2, Etat3, Etat4};
enum Etat etat;
char x,y;
etat=Etat0;
while(scanf("%c",&x)!=EOF){
switch(etat){
    case (Etat0):
        y='0';
        if(x=='1') etat=Etat0;
        if(x=='0') etat=Etat1;
        break;
    case (Etat1):
        y='0';
        if(x=='1') etat=Etat2;
        if(x=='0') etat=Etat1;
        break;
    ....
```

```
eisea@VM-Linux:~/Systemes$ ./FSM<entree.txt
00000100001eisea@VM-Linux:~/Systemes$
```

Fichier d'entrée

1011000110

entree.txt x

1011000110

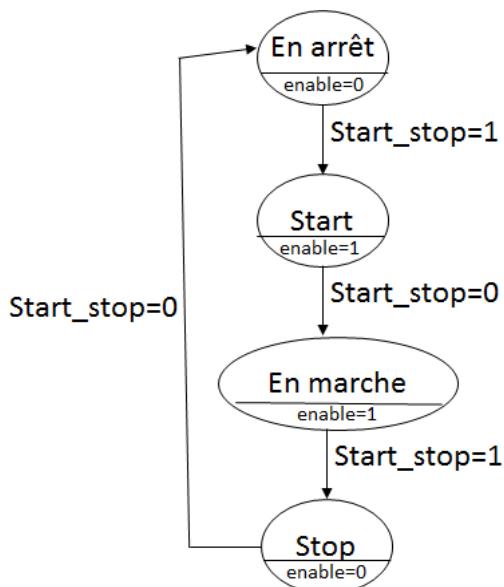
Fichier de sortie

00000100001

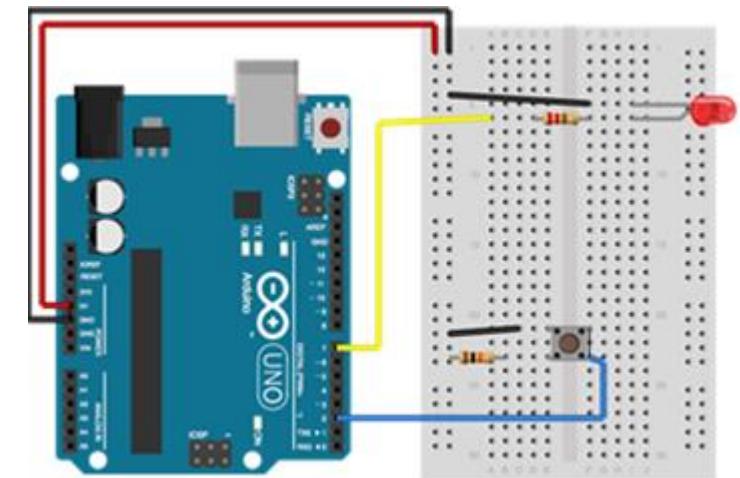
II. Implémentation logicielle

1) Ecriture à la main du code de la machine à états

- Exemple: à chaque fois qu'on appuie sur le bouton, la led change d'état

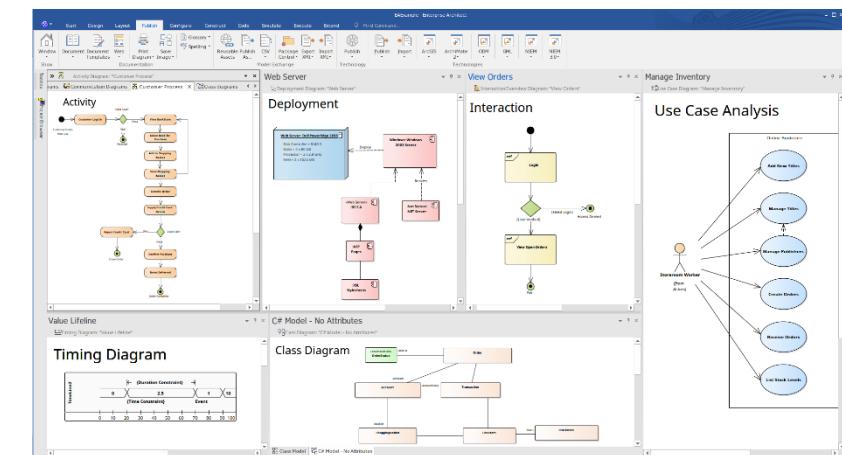
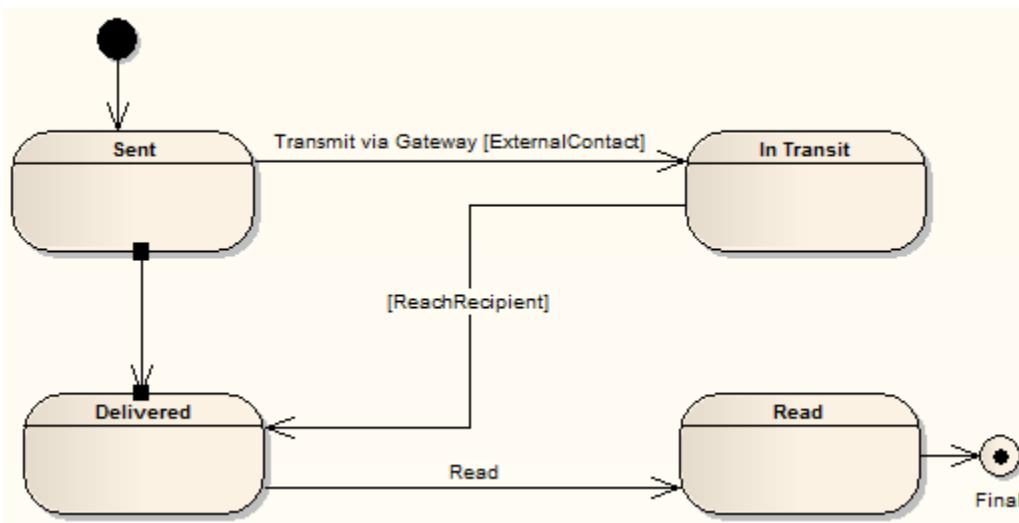


```
enum Etat {En_arret, Start, En_marche, Stop};  
enum Etat etat;  
const int buttonPin = 2;  
const int ledEnable = 7;  
int start_stop = 0;  
  
void setup() {  
    etat=En_arret;  
    pinMode(ledEnable, OUTPUT);  
    digitalWrite(ledEnable, LOW);  
    pinMode(buttonPin, INPUT);  
}  
  
void loop() {  
  
    start_stop = digitalRead(buttonPin);  
  
    switch(etat){  
        case (En_arret):  
            digitalWrite(ledEnable, LOW);  
            if(start_stop==HIGH) etat=Start;  
            else etat=En_arret;  
            break;  
  
        case (Start):  
            digitalWrite(ledEnable, HIGH);  
            if(start_stop==LOW) etat=En_marche;  
            else etat=Start;  
            break;  
  
        case (En_marche):  
            digitalWrite(ledEnable, HIGH);  
            if(start_stop==HIGH) etat=Stop;  
            else etat=En_marche;  
            break;  
  
        case (Stop):  
            digitalWrite(ledEnable, LOW);  
            if(start_stop==LOW) etat=En_arret;  
            else etat=Stop;  
            break;  
    }  
}
```



II. Implémentation logicielle

- 2) Modélisation de la machine à états dans un logiciel qui permet la génération automatique de son code



Enterprise Architect

