INF1500 : Logique des systèmes numériques

- Cours 10: Conception (synthèse) de circuits séquentiels synchrones (Finite State Machines – FSM)

Procédure

- Voici la procédure pour concevoir un circuit séquentiel synchrone :
  - Bâtir un diagramme d'état à partir des données du problème;
  - Bâtir le tableau d'état à partir du diagramme d'état, en identifiant les états par des symboles;
  - Réduire le nombre d'états nécessaires en éliminant les états équivalents;
  - Assigner un code binaire à chaque état, et ajouter cette information au tableau d'état;
  - À partir du tableau d'état complet, obtenir les équations booléennes d'entrée des bascules ainsi que les équations booléennes des sorties du système, en simplifiant si possible;
  - Donner le diagramme et/ou construire le circuit; et,
  - Vérifier, vérifier, vérifier.

Bâtir un diagramme et un tableau d'état à partir des données du problème

- La première étape de conception consiste à obtenir un tableau d'état à partir des données du problème. En règle générale, il est beaucoup plus facile d'obtenir tout d'abord un diagramme d'état.

- Les principes suivants peuvent grandement aider à obtenir le diagramme d'état.

  1. À partir des données du problème, simuler certaines combinaisons d'entrée et de sortie pour bien comprendre la nature du problème.
  2. Commencer par construire un diagramme partiel menant à une sortie désirée du système.
  3. Ajouter au diagramme les autres chemins menant aux sorties désirées du système.
  4. Vérifier le diagramme pour éviter les états inutiles.
  5. Compléter le diagramme en ajoutant des transitions pour toutes les entrées possibles à partir de chaque état.
  6. Identifier toute condition où le circuit doit être réinitialisé à un état de départ, et annoter le diagramme avec cette information.
  7. Vérifier le diagramme en appliquant les combinaisons d'entrées et de sorties identifiées au début.

Eliminer les états équivalents

- Deux états sont équivalents si et seulement si, pour chaque entrée, les sorties sont les mêmes et les prochains états sont les mêmes. On peut utiliser à profit les valeurs « peu importe » (*don't care*).

- Une procédure simple pour éliminer les états équivalents consiste à dresser un tableau d'état où les états sont représentés par des symboles. En faisant des passes successives à travers le tableau, on combine les états équivalents selon la définition donnée plus haut.

- Cette procédure ne permet pas toujours d'obtenir la solution la plus simple possible. Une autre technique, utilisant une table d'implication, garantie une solution optimale.

Assigner un code binaire à chaque état

- Il faut éventuellement assigner un code binaire à chaque état pour en venir à une réalisation du circuit. Par exemple, si on utilise deux bascules, il y a 4 combinaisons possibles : {00, 01, 10, 11}.

- Le problème de design consiste donc à choisir le « meilleur » arrangement, c'est-à-dire celui qui résultera en un circuit le plus simple possible. L'assignation d'un code binaire à chaque état relève beaucoup de l'expérience. On peut optimiser le choix en exploitant les adjacences entre états:

  - Etats qui ont le même prochain état;
  - Etats qui sont les prochains états d'un même état; et
  - Etats qui ont la même sortie.

Assigner un code binaire à chaque état - Suite

- Idéalement, une seule variable d'état devrait changer entre deux états adjacents. Cela simplifierait grandement la logique nécessaire pour implémenter chaque équation d'état. Ce n'est pas possible en pratique. Il est en général difficile de déterminer la « meilleure » façon d'assigner un code à chaque état.

- Dans la méthode d'assignation « one hot », on alloue une bascule par état. Une seule bascule peut donc avoir une sortie égale à 1 à la fois. L'avantage de cette assignation, c'est que les équations d'état sont très simples. Un premier désavantage est qu'il faut beaucoup de bascules quand on a plusieurs états à représenter. Un second désavantage est que beaucoup de combinaisons des variables d'état ne représentent pas un état valable (du moment que plus d'une variable d'état vaut 1 à la fois).

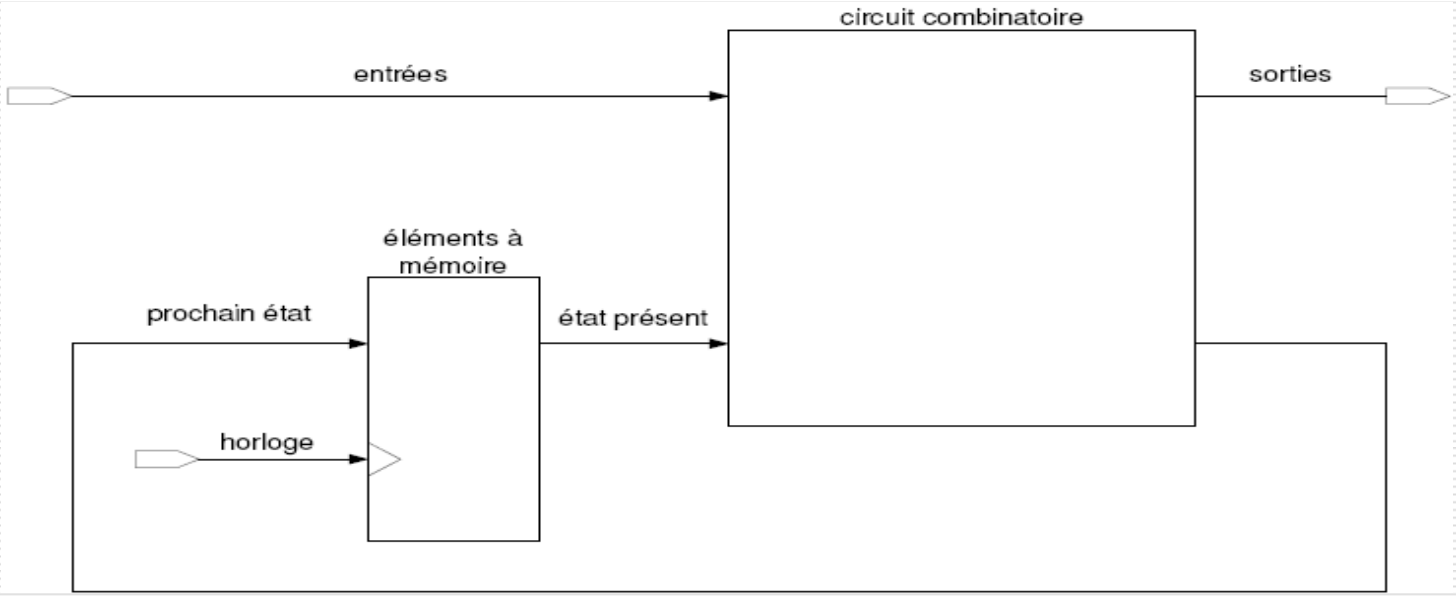
Conception de machines de Moore et de machines de Mealy

- En général, une machine de Moore contient plus d'états et les variables d'état sont plus complexes. Cependant, la sortie d'une machine de Moore est plus fiable puisqu'elle est constante pendant toute une période d'horloge, indépendamment des entrées du système.

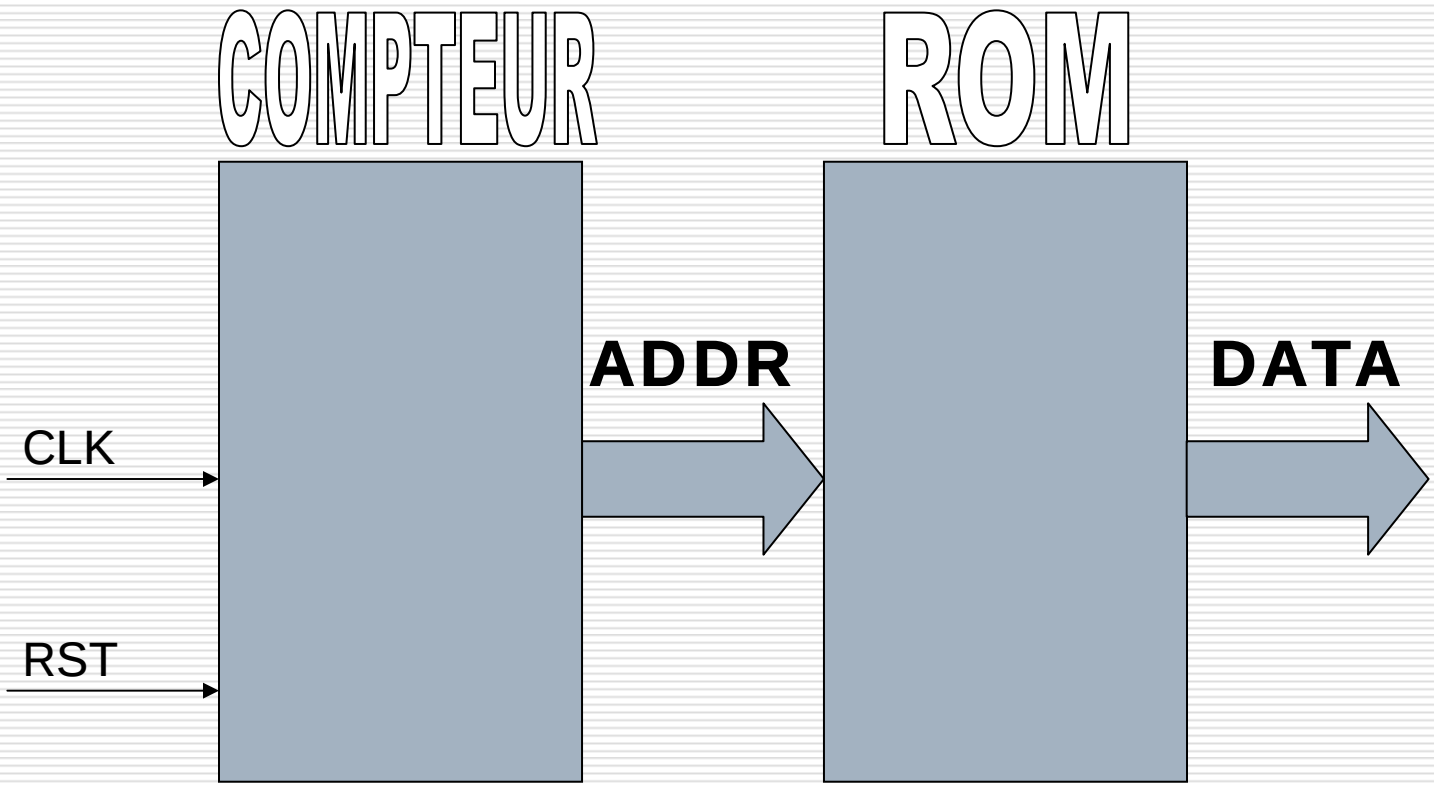
Conception avec ROM, PLA et PAL

- Rien n'empêche l'utilisation d'une mémoire ROM pour réaliser le circuit combinatoire directement.

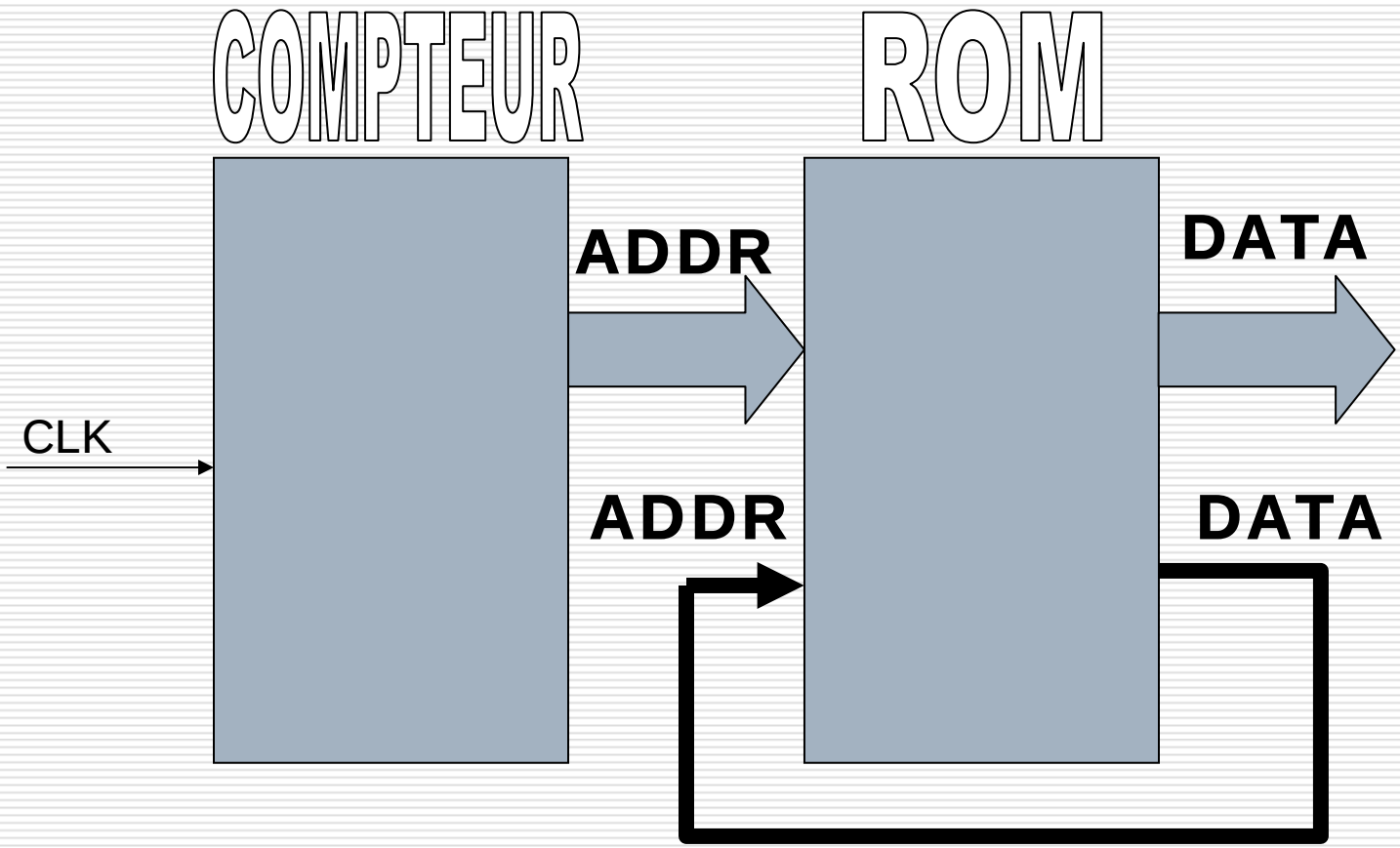
- De façon similaire, les PALs peuvent inclure des éléments à mémoire. On peut donc réaliser un circuit séquentiel complet avec une seule puce.



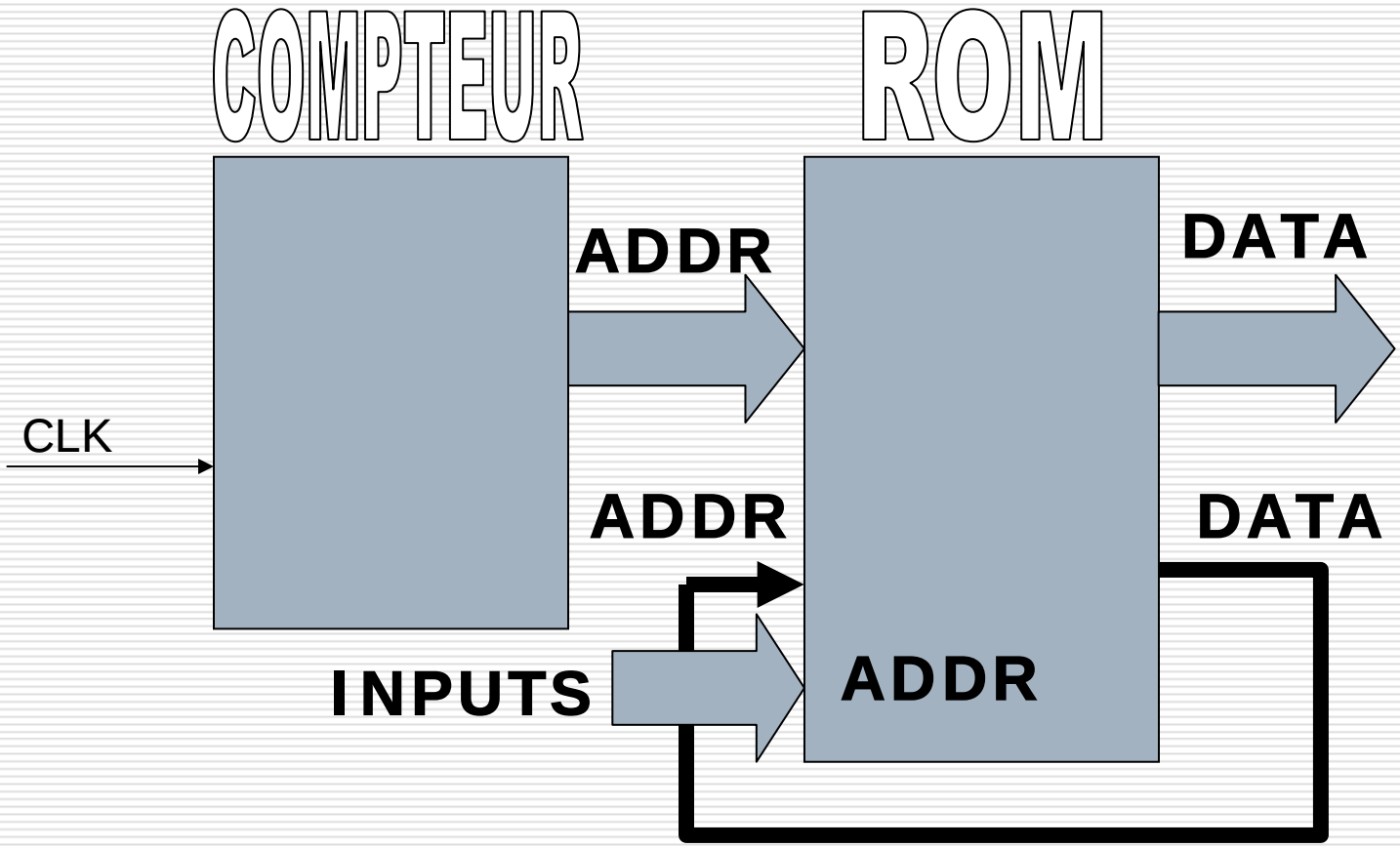
Exemple: Séquence utilisant un ROM avec sorties (DATA) utilisant un compteur



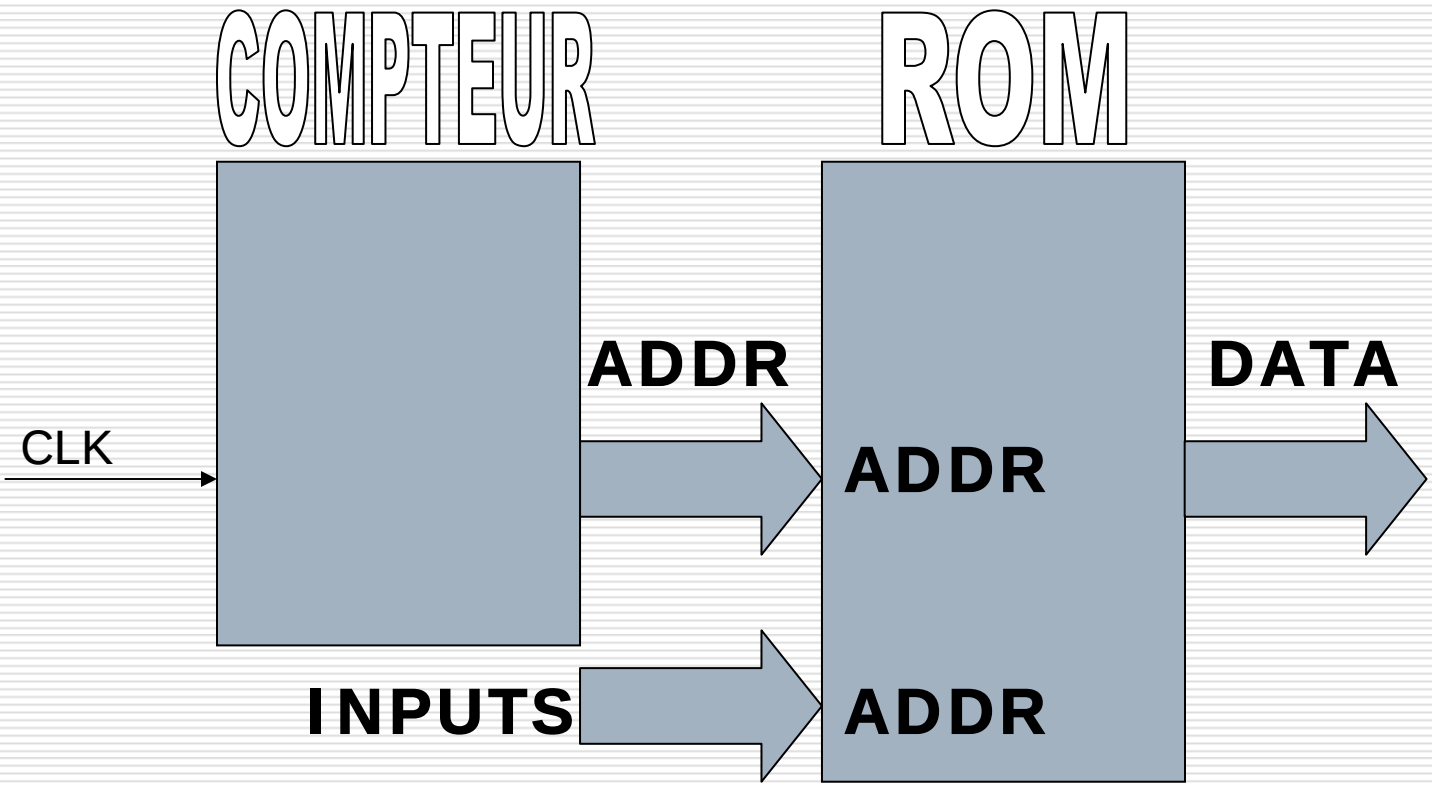
Exemple: Séquence utilisant un ROM avec sorties (DATA) utilisant un compteur et un parties de DATA comme partie de l'adresse pour la prochaine séquence (MOORE).



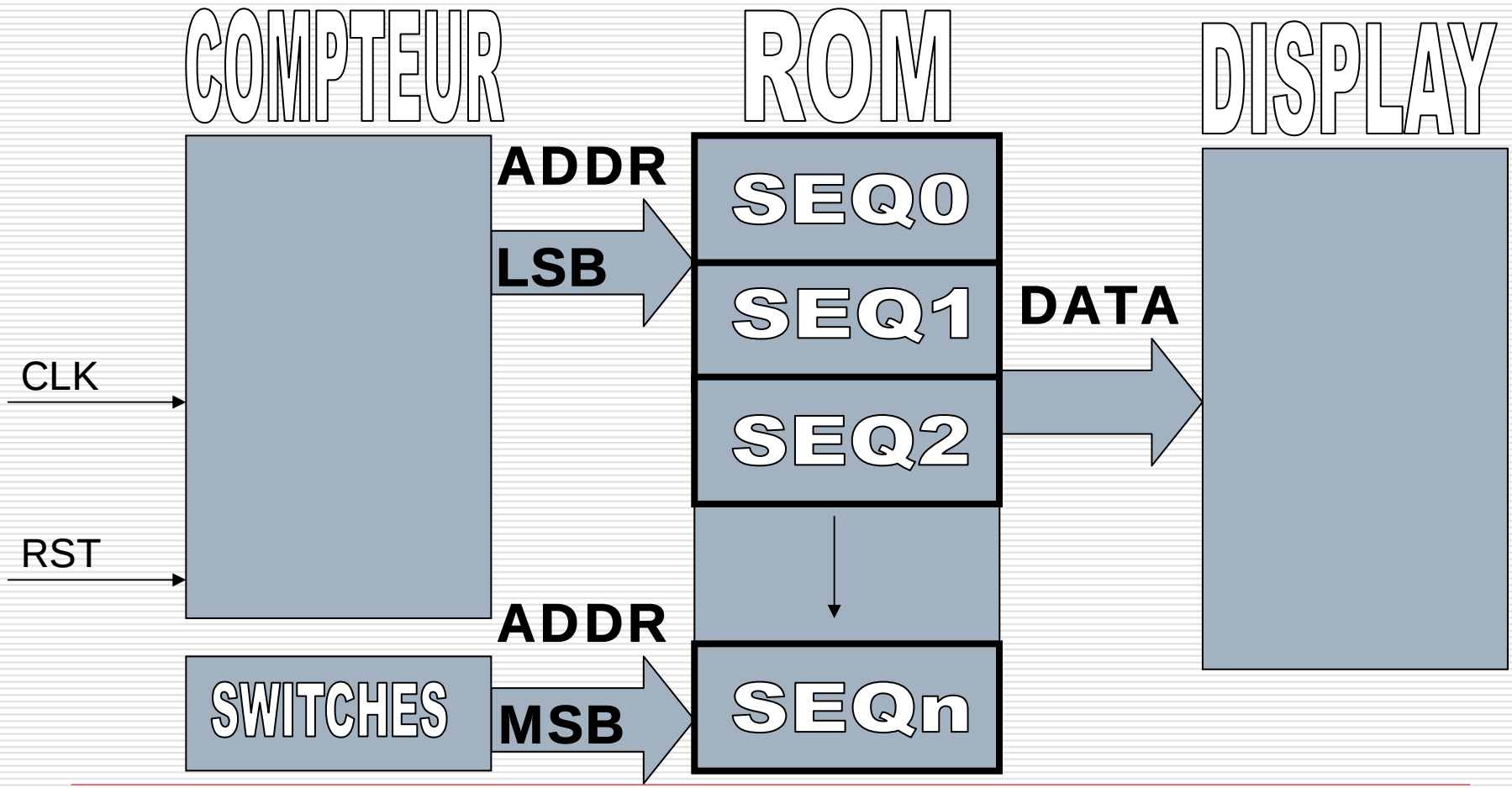
Exemple: Séquence utilisant un ROM avec sorties (DATA) utilisant un compteur et un parties de DATA comme partie de l'adresse pour la prochaine séquence (MEALY).



Exemple: Séquence utilisant un ROM avec sorties (DATA) utilisant un compteur (MEALY)



ROM – Exemple: Séquences d' effets spéciaux



Equations pour bascules D, JK et T

- Dans tous les problèmes de conception de circuit séquentiel, il faut déterminer les équations des entrées des bascules. Dans chaque cas, il faut déterminer quelle valeur appliquer aux entrées en fonction de l'état présent de la bascule et de l'état désiré de la bascule.

  - Pour une bascule D (Q+ = D), c'est très simple. Comme la bascule D n'a que le mode « saisie de donnée », la valeur à appliquer à l'entrée D est la même que celle qu'on désire voir comme état.

  - Pour une bascule JK (Q+ = JQ' + K'Q), il y a deux entrées et quatre modes possibles : mémoire (00), reset (01), set (10) et inverse (11). Par exemple, si l'état actuel de la bascule est 0 et qu'on désire le faire passer à 1, il y a deux possibilités : soit set (10) ou inverse (11). Dans ce cas, l'entrée K est sans importance.

  - Pour une bascule T (Q+ = T ou Q), il n'y a qu'une seule entrée et deux modes : mémoire et inverse. Il faut déterminer si on veut inverser ou non l'état présent de la bascule.

Equations pour bascules D, JK et T - Suite

Le tableau suivant résume la situation.

Q actuel	Q désiré	bascule D		bascule JK			bascule T	
		mode	entrée D	mode	entrée J	entrée K	mode	entrée T
0	0	saisie	0	reset ou mémoire	0	-	mémoire	0
0	1	saisie	1	set ou inverse	1	-	inverse	1
1	0	saisie	0	reset ou inverse	-	1	inverse	1
1	1	saisie	1	set ou mémoire	-	0	mémoire	0

Exemple 1: Cadenas à combinaisons numériques

- Spécifications:
  - Une entrée IN (“0” ou “1”)
  - Une sortie U (Unlock) (signal “débarré”)
  - DEBARRE est 1 ssi:
    - Les dernières entrées étaient la combinaison: 0110

Implémentation “One Hot”

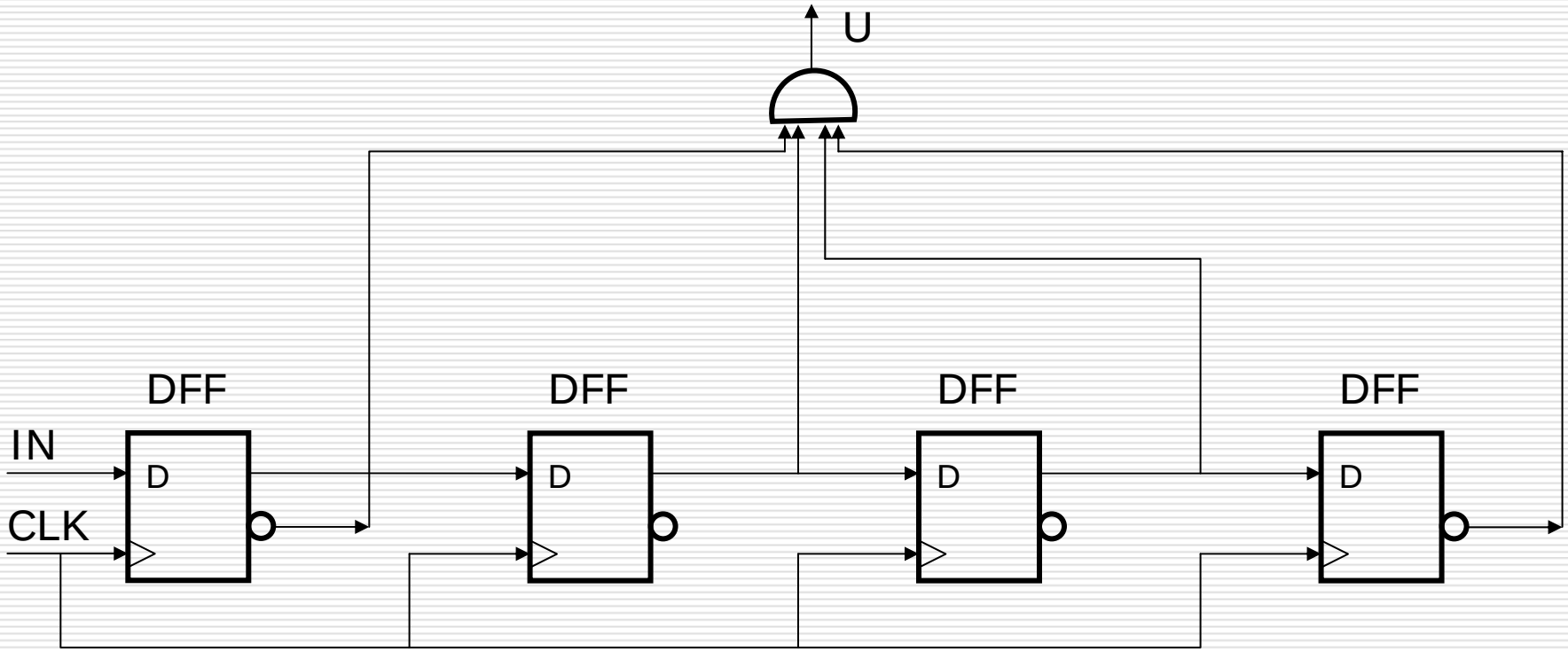


Diagramme d'état (State Transition Diagram)

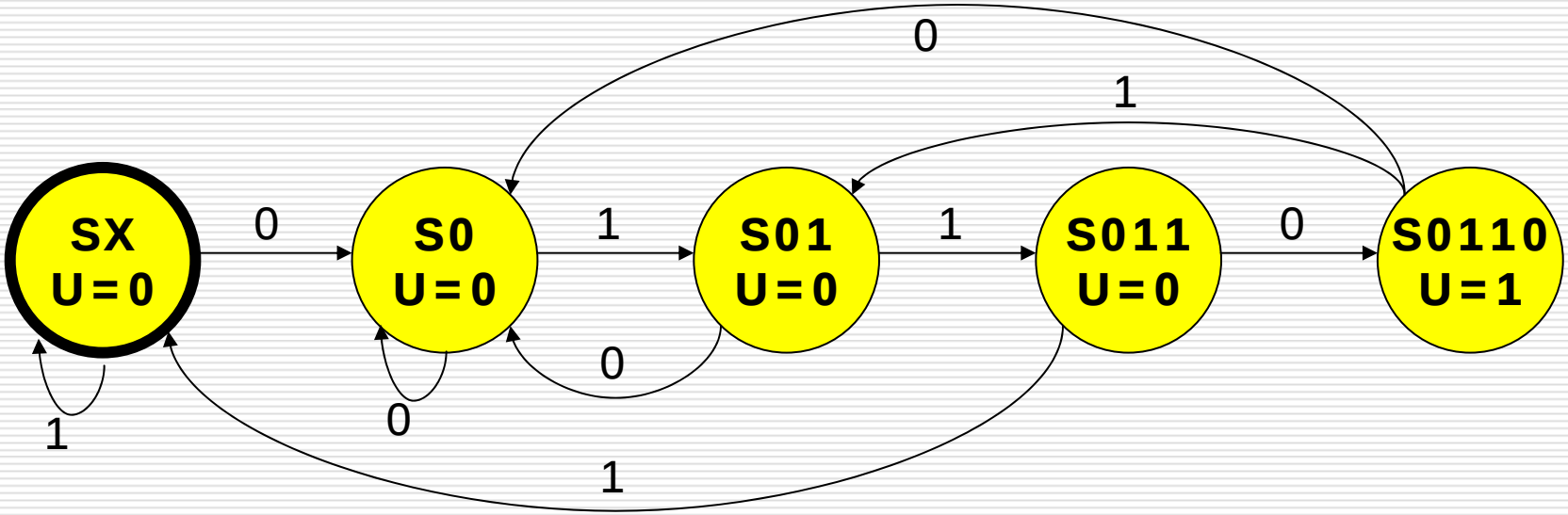
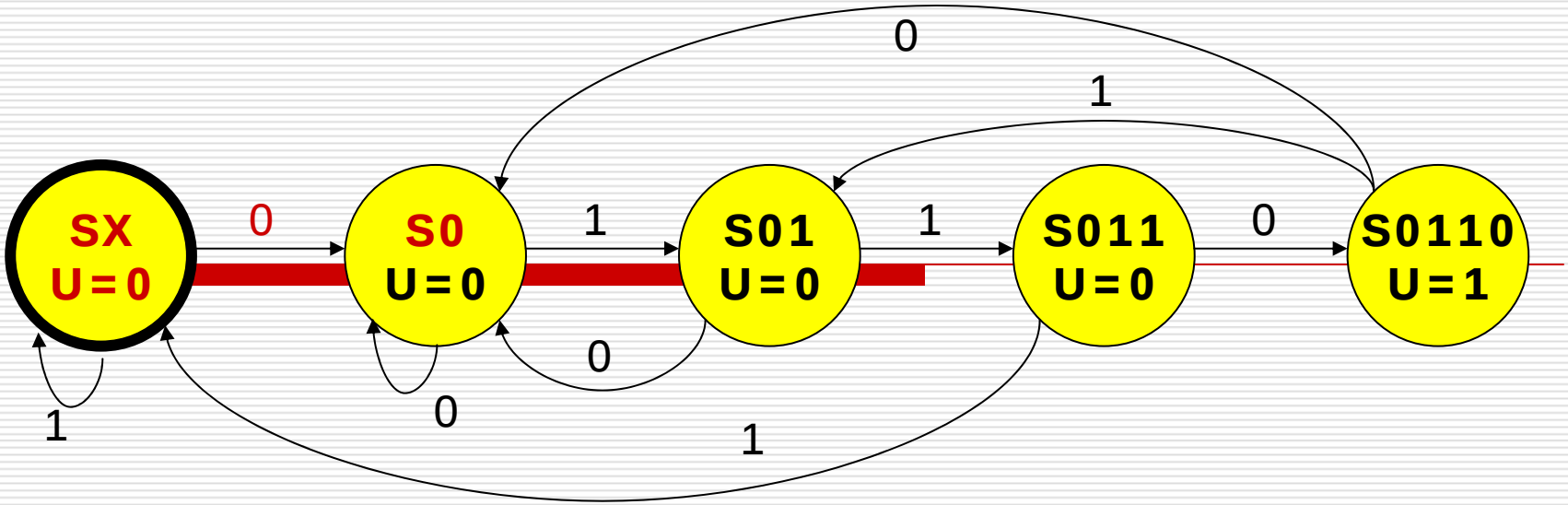


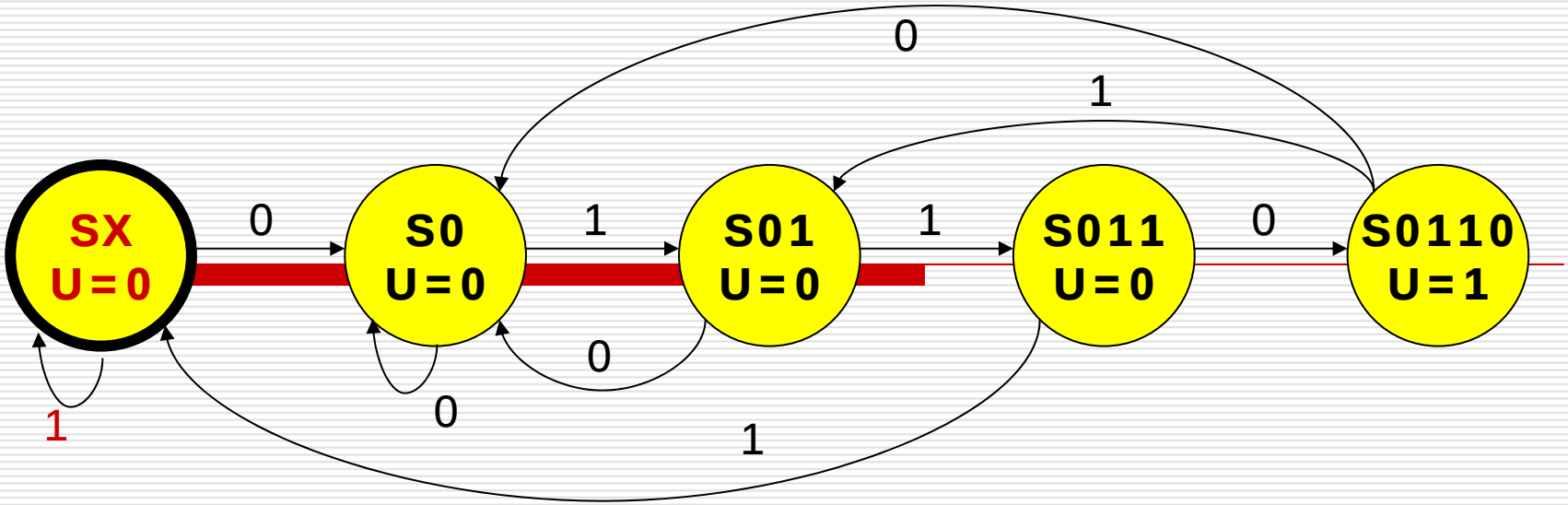
Tableau d'état (State Transition Table)

IN	PRESENT ETAT		PROCHAIN ETAT		U
0	SX	000	S0	001	0
1	SX	000	SX	000	0
0	S0	001	S0	001	0
1	S0	001	S01	011	0
0	S01	011	S0	001	0
1	S01	011	S011	010	0
0	S011	010	S0110	100	0
1	S011	010	SX	000	0
0	S0110	100	**S0**	001	1
1	S0110	100	**S01**	011	1

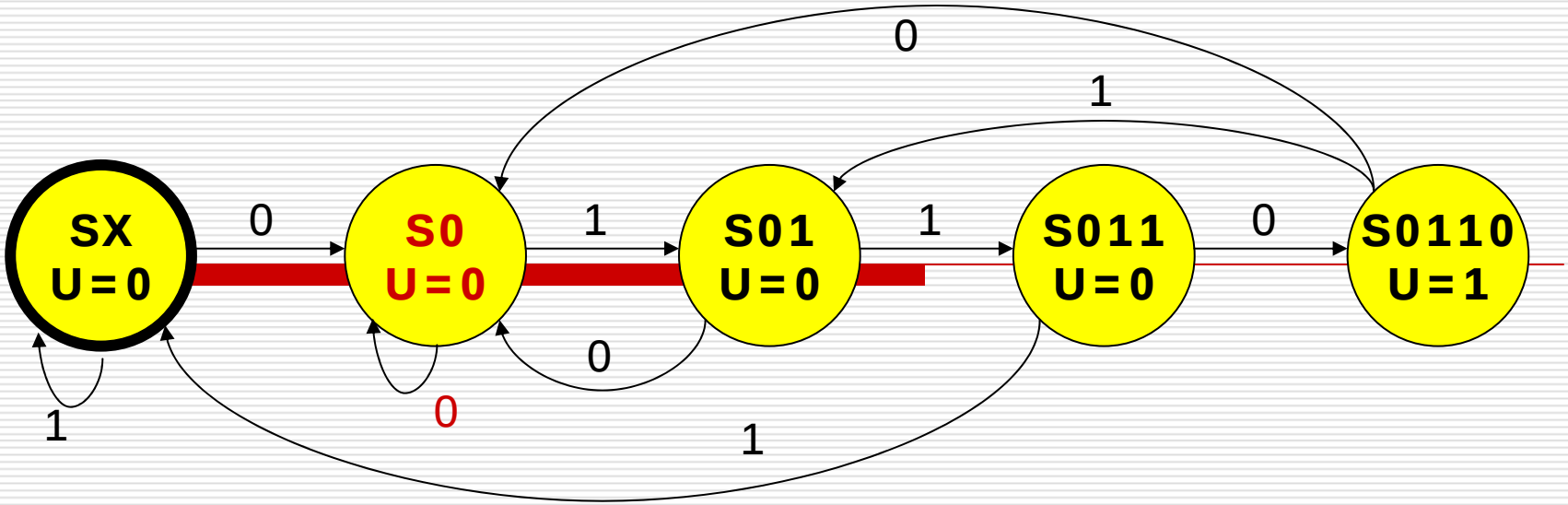
arbitraire



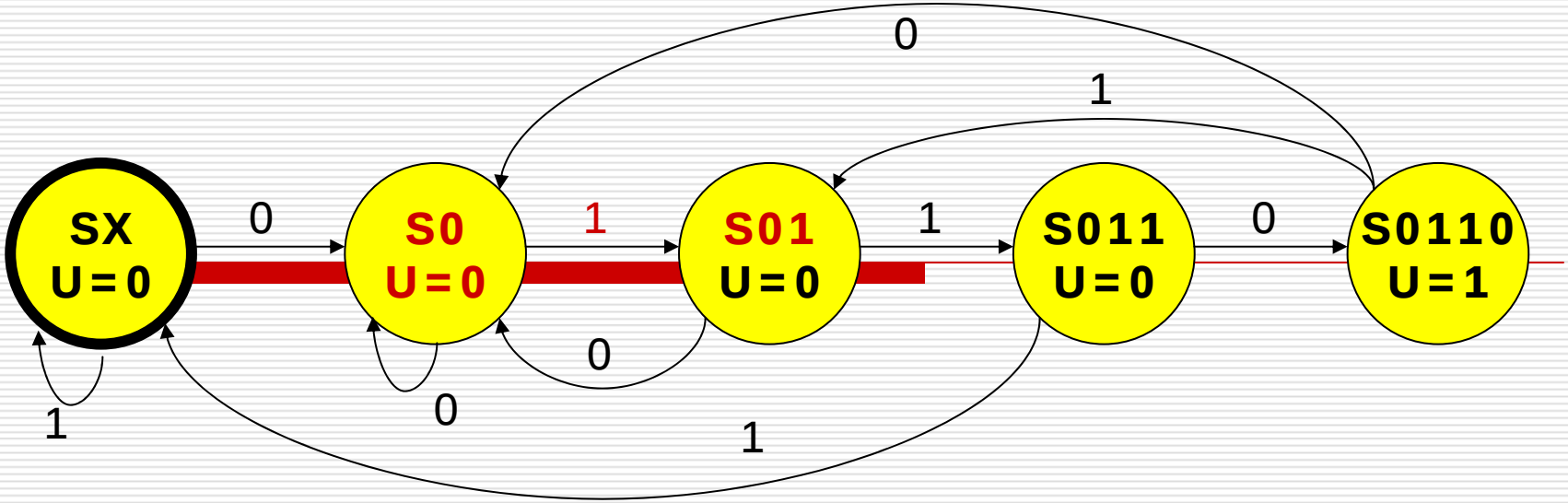
IN	PRESENT ETAT		PROCHAIN ETAT		U
0	SX	000	S0	001	0
1	SX	000	SX	000	0
0	S0	001	S0	001	0
1	S0	001	S01	011	0
0	S01	011	S0	001	0
1	S01	011	S011	010	0
0	S011	010	S0110	100	0
1	S011	010	SX	000	0
0	S0110	100	S0	001	1
1	S0110	100	S01	011	1



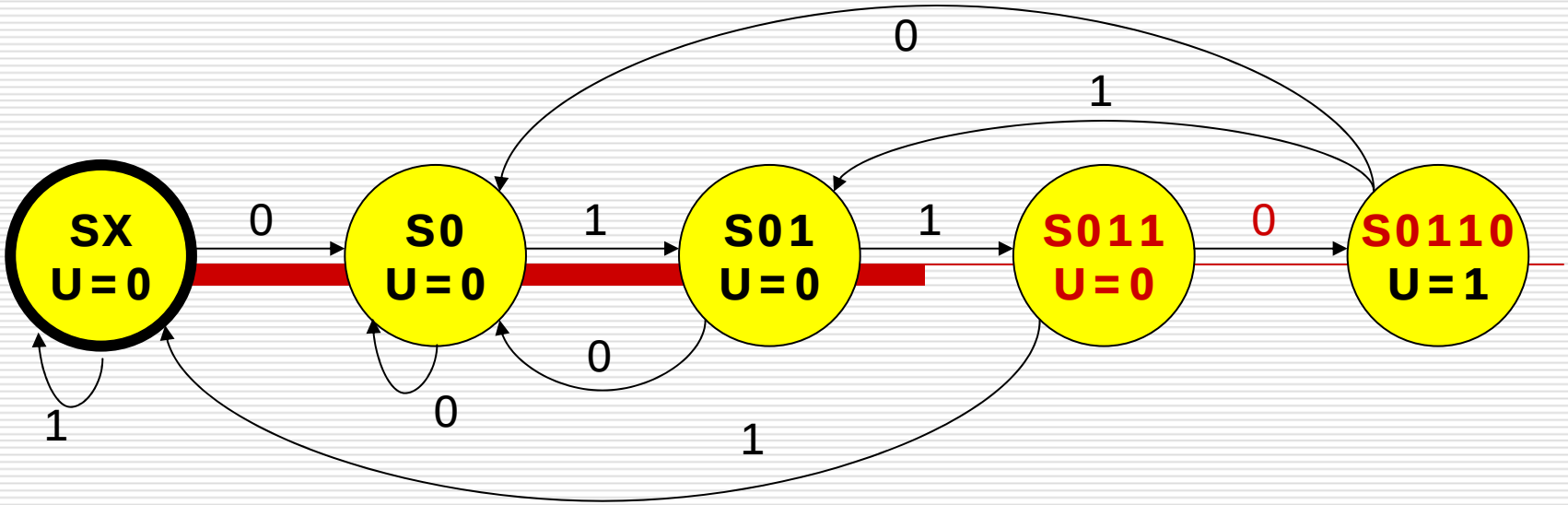
IN	PRESENT ETAT		PROCHAIN ETAT		U
0	SX	000	S0	001	0
1	SX	000	SX	000	0
0	S0	001	S0	001	0
1	S0	001	S01	011	0
0	S01	011	S0	001	0
1	S01	011	S011	010	0
0	S011	010	S0110	100	0
1	S011	010	SX	000	0
0	S0110	100	S0	001	1
1	S0110	100	S01	011	1



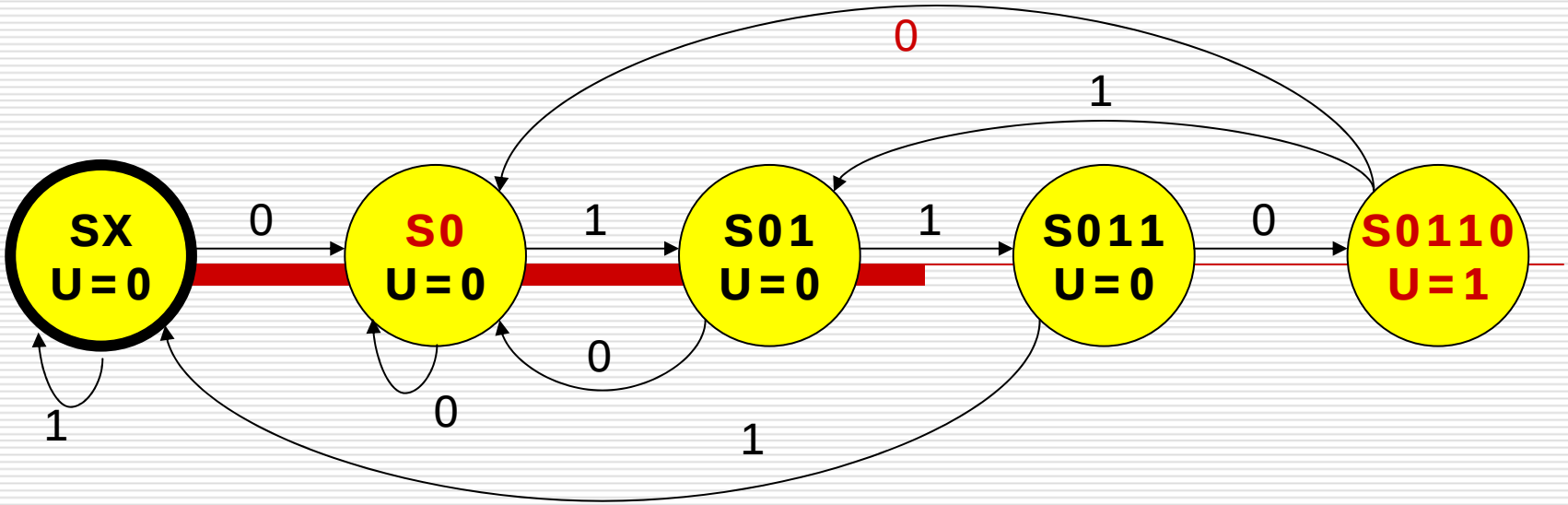
IN	PRESENT ETAT		PROCHAIN ETAT		U
0	SX	000	S0	001	0
1	SX	000	SX	000	0
0	S0	001	S0	001	0
1	S0	001	S01	011	0
0	S01	011	S0	001	0
1	S01	011	S011	010	0
0	S011	010	S0110	100	0
1	S011	010	SX	000	0
0	S0110	100	S0	001	1
1	S0110	100	S01	011	1



IN	PRESENT ETAT		PROCHAIN ETAT		U
0	SX	000	S0	001	0
1	SX	000	SX	000	0
0	S0	001	S0	001	0
1	S0	001	S01	011	0
0	S01	011	S0	001	0
1	S01	011	S011	010	0
0	S011	010	S0110	100	0
1	S011	010	SX	000	0
0	S0110	100	S0	001	1
1	S0110	100	S01	011	1

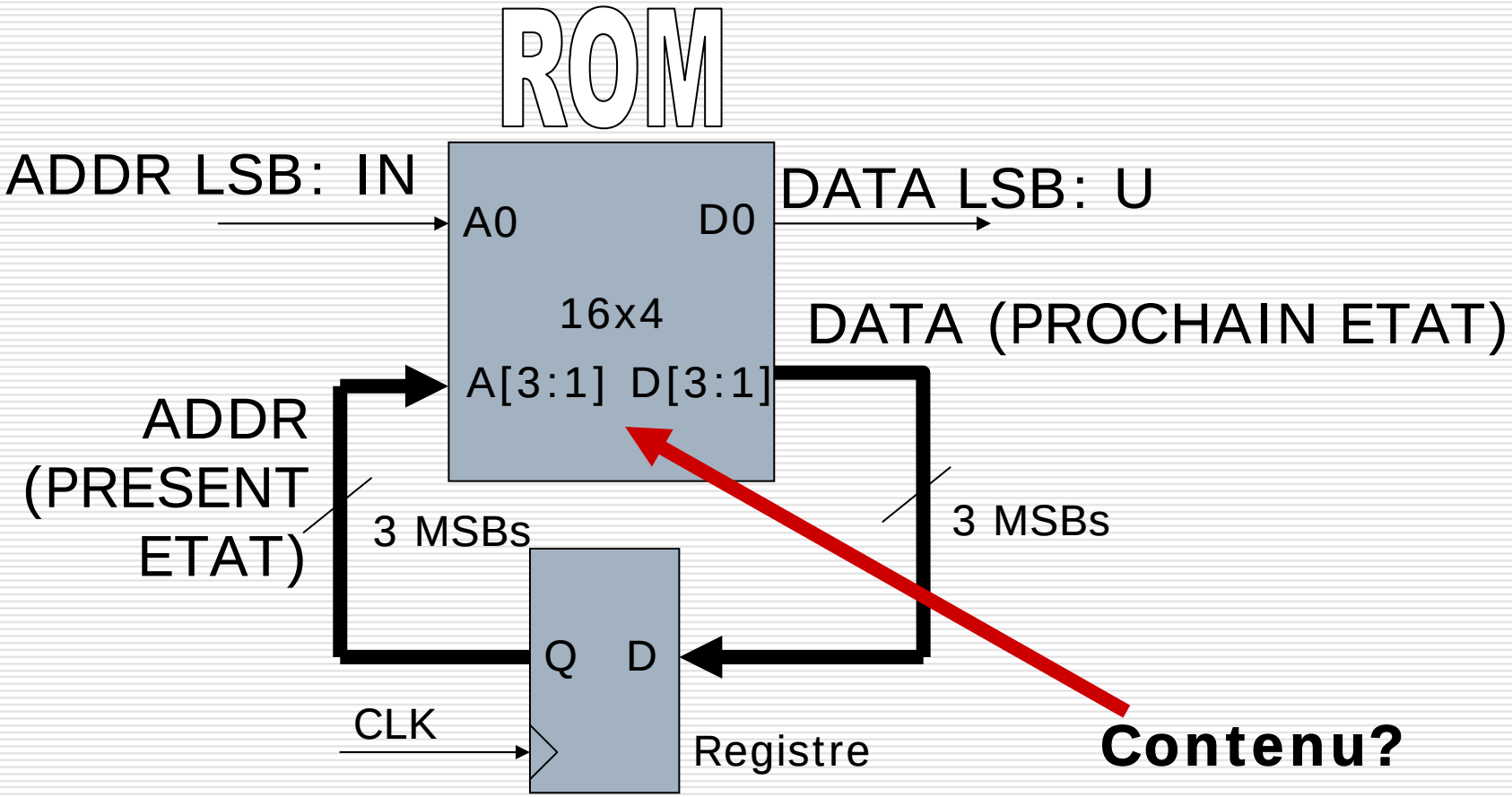


IN	PRESENT ETAT		PROCHAIN ETAT		U
0	SX	000	S0	001	0
1	SX	000	SX	000	0
0	S0	001	S0	001	0
1	S0	001	S01	011	0
0	S01	011	S0	001	0
1	S01	011	S011	010	0
0	S011	010	S0110	100	0
1	S011	010	SX	000	0
0	S0110	100	S0	001	1
1	S0110	100	S01	011	1



IN	PRESENT ETAT		PROCHAIN ETAT		U
0	SX	000	S0	001	0
1	SX	000	SX	000	0
0	S0	001	S0	001	0
1	S0	001	S01	011	0
0	S01	011	S0	001	0
1	S01	011	S011	010	0
0	S011	010	S0110	100	0
1	S011	010	SX	000	0
0	S0110	100	S0	001	1
1	S0110	100	S01	011	1

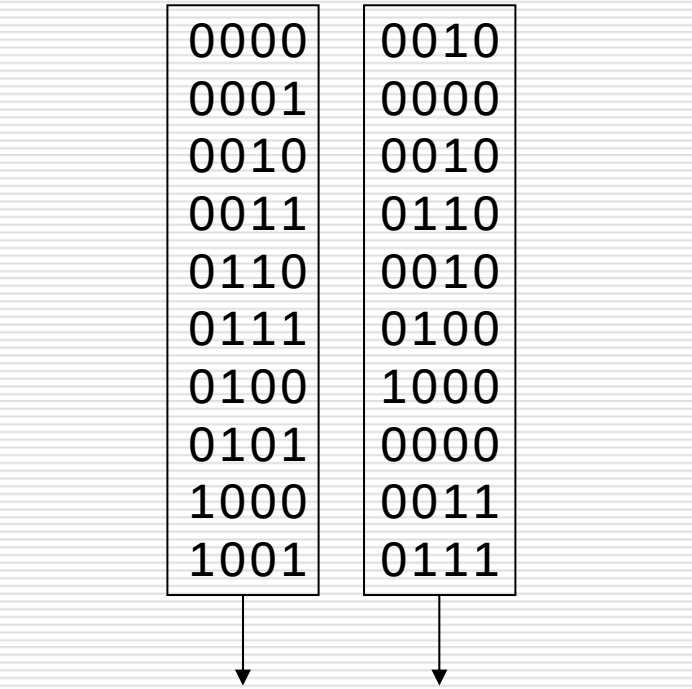
Implémentation utilisant un ROM



Contenu du ROM

IN	PRESENT ETAT	PROCHAIN ETAT		U
0	SX	000	S0	001	0
1	SX	000	SX	000	0
0	S0	001	S0	001	0
1	S0	001	S01	011	0
0	S01	011	S0	001	0
1	S01	011	S011	010	0
0	S011	010	S0110	100	0
1	S011	010	SX	000	0
0	S0110	100	S0	001	1
1	S0110	100	S01	011	1
↓		↓		↓	
A0		A3 A2 A1		D3 D2 D1	D0

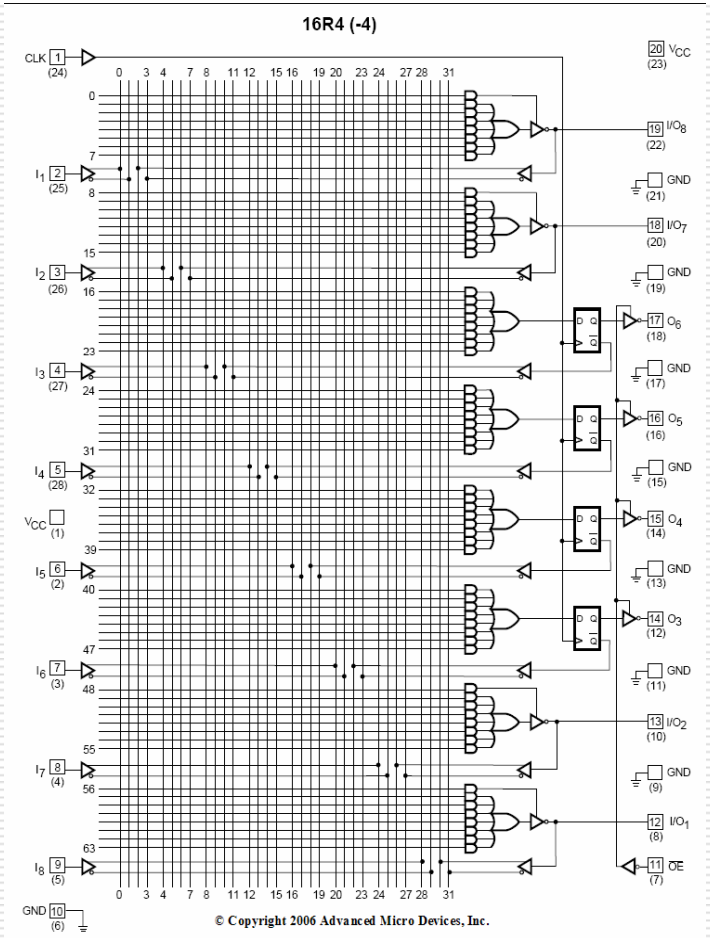
Contenu du ROM - Suite



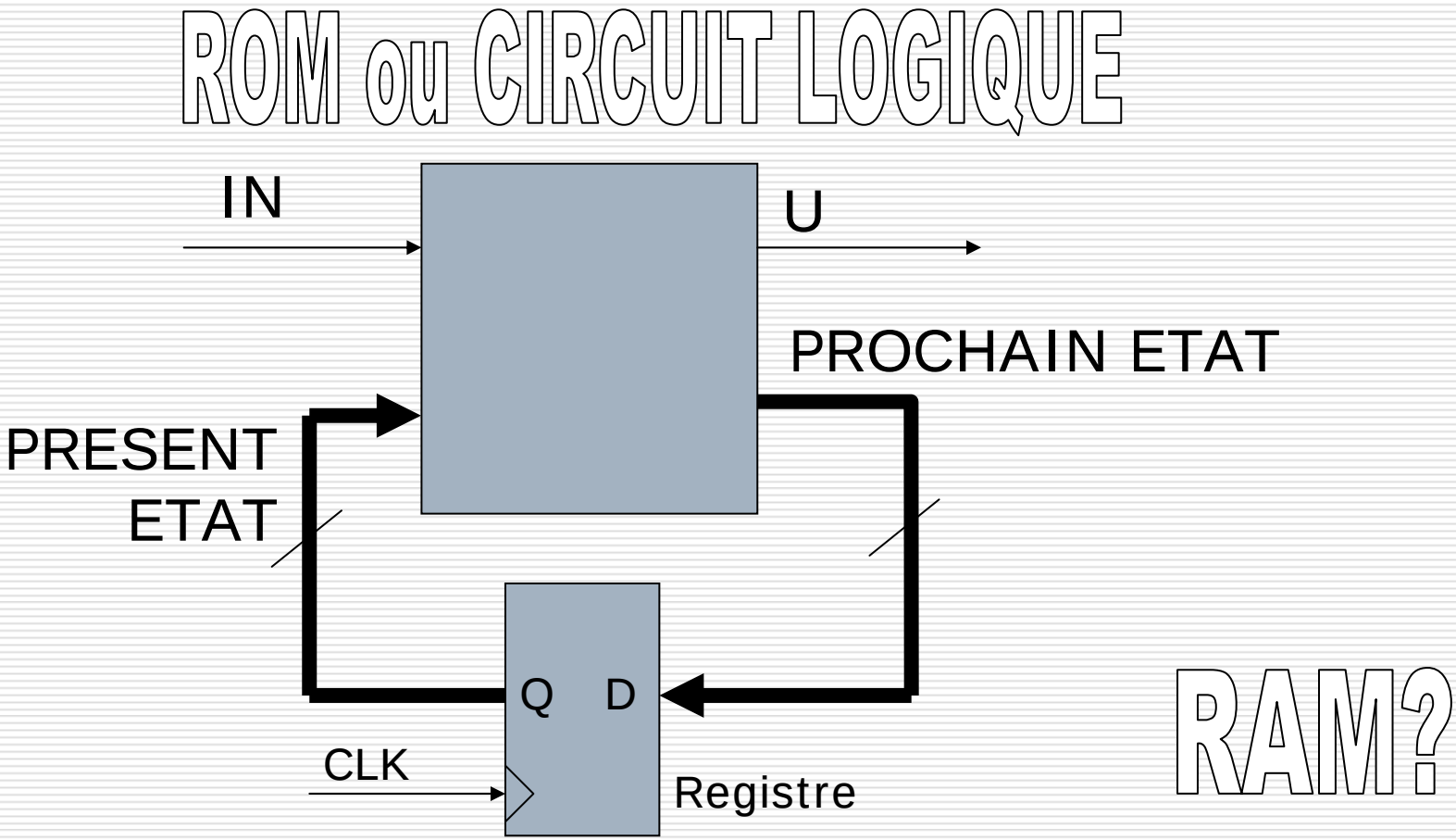
ROM

ADDR	DATA
0000	0010
0001	0000
0010	0010
0011	0110
0100	1000
0101	0000
0110	0010
0111	0100
1000	0011
1001	0111
1010	XXXX
1011	XXXX
1100	XXXX
1101	XXXX
1110	XXXX
1111	XXXX

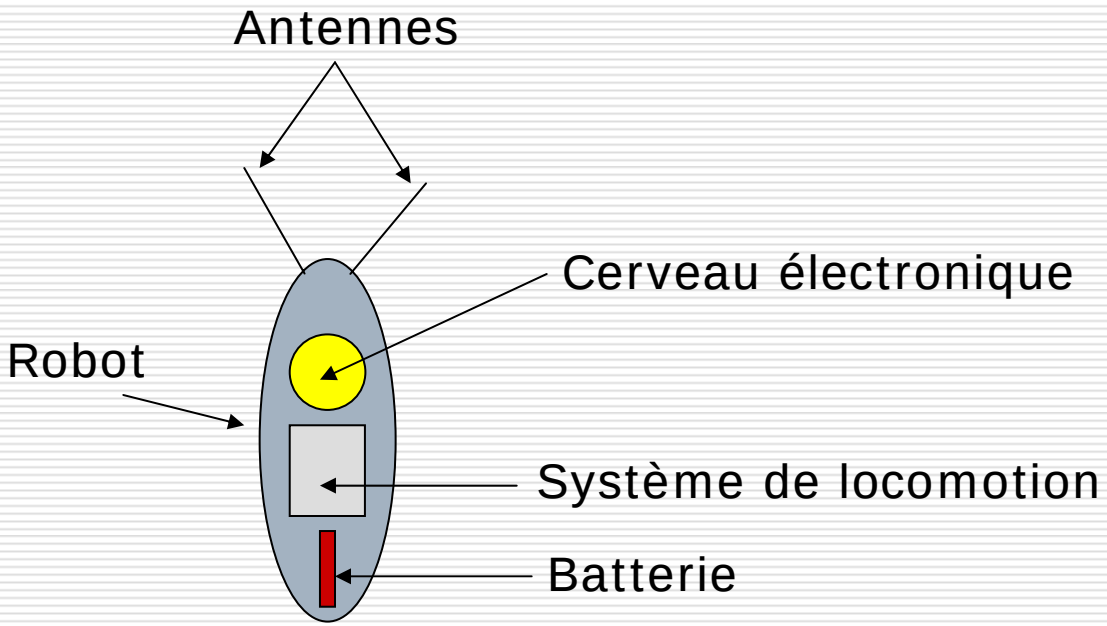
Autre alternative: PAL



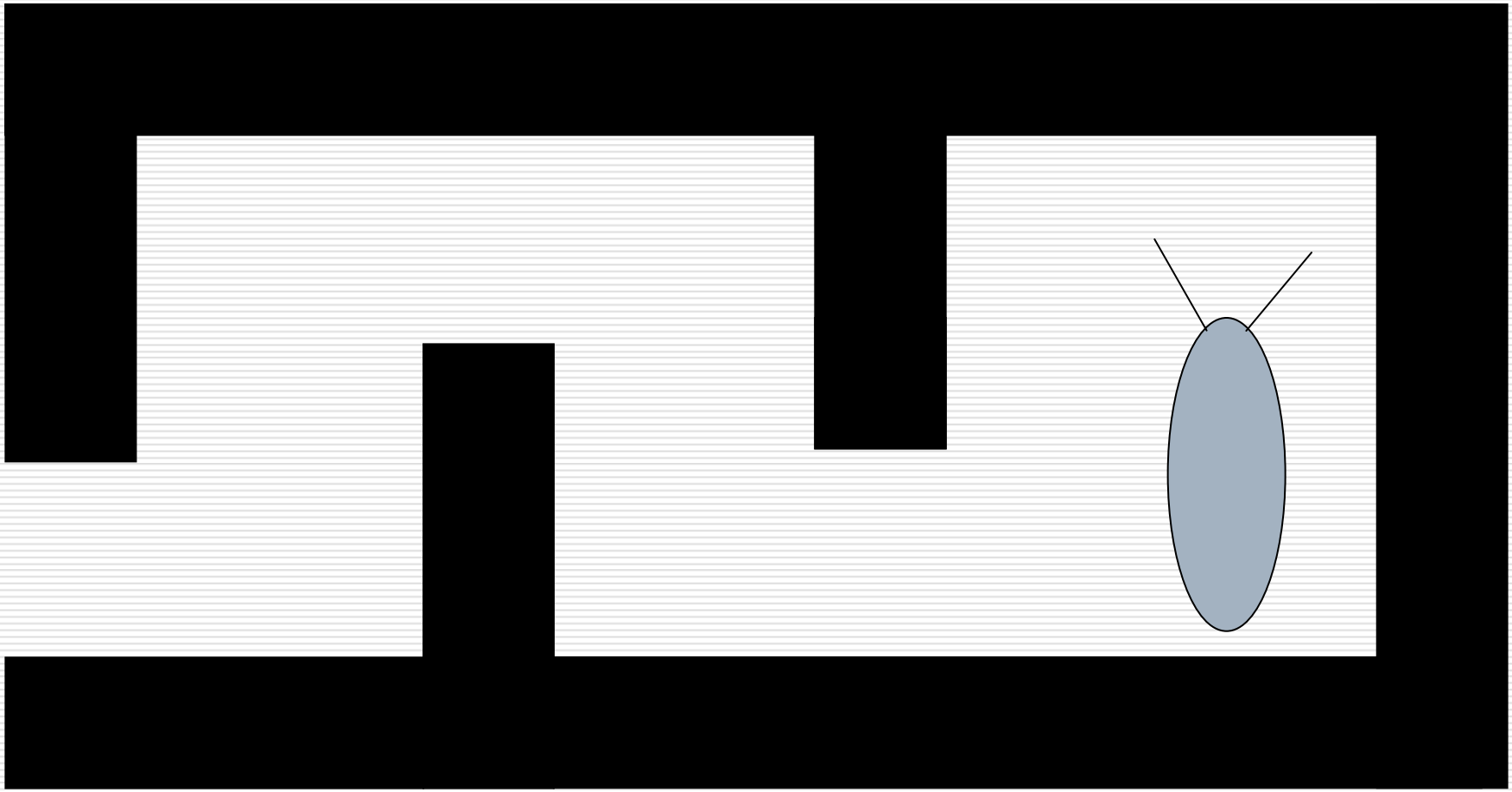
Minimiser l'électronique requise



Exemple 2: Robot insecte



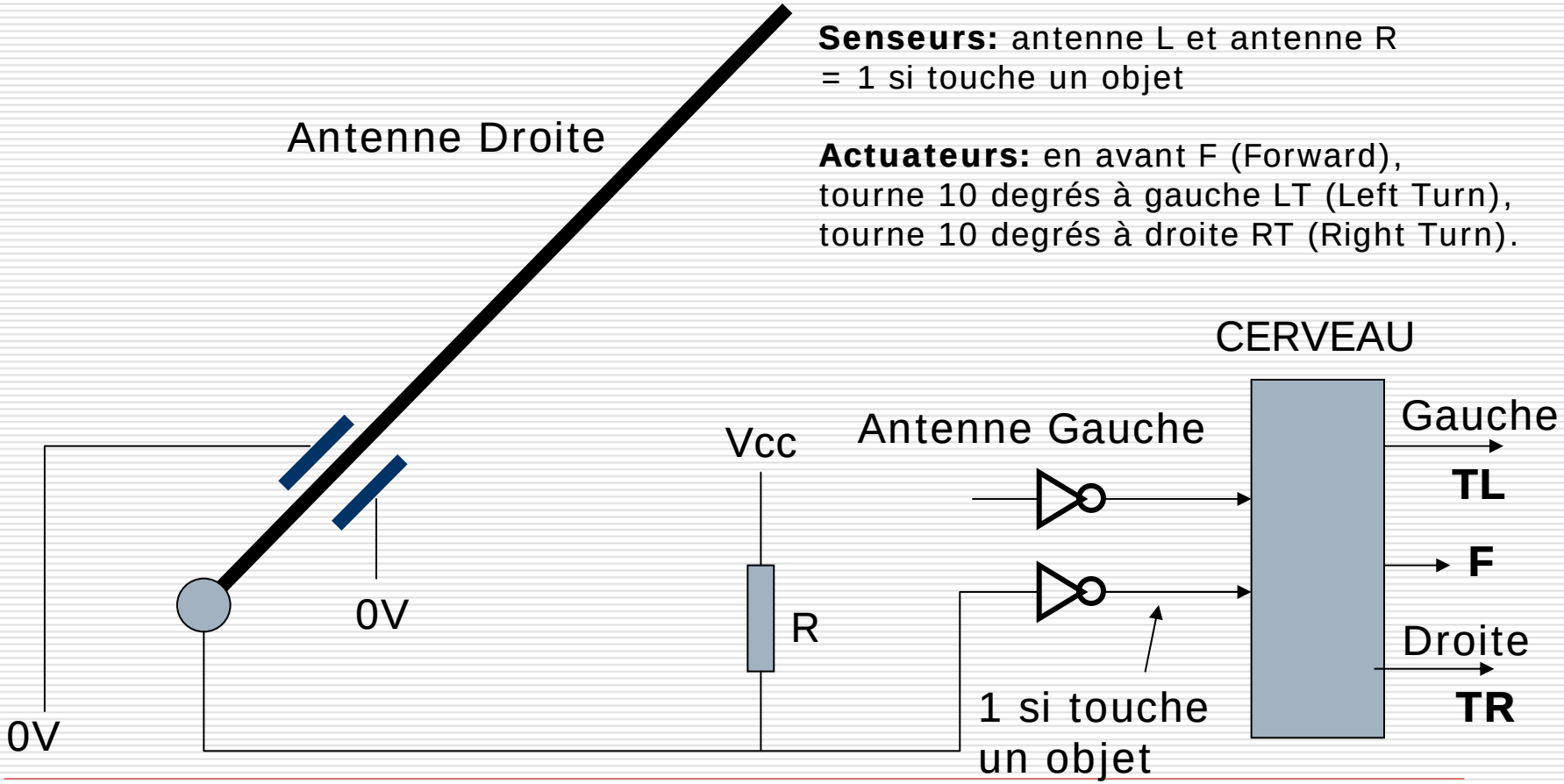
Objectif: implanter le minimum d'intelligence (limite d'espace et de puissance) pour que le robot insecte sorte du labyrinthe



Antenne

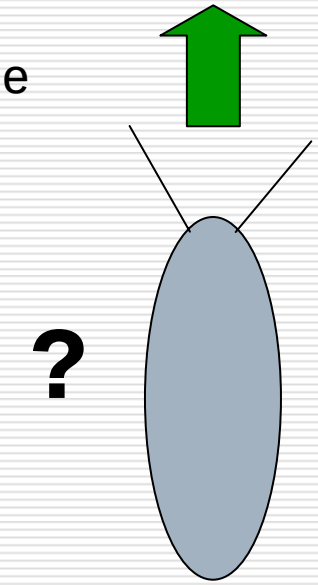
Senseurs: antenne L et antenne R
= 1 si touche un objet

Actuateurs: en avant F (Forward),
tourne 10 degrés à gauche LT (Left Turn),
tourne 10 degrés à droite RT (Right Turn).

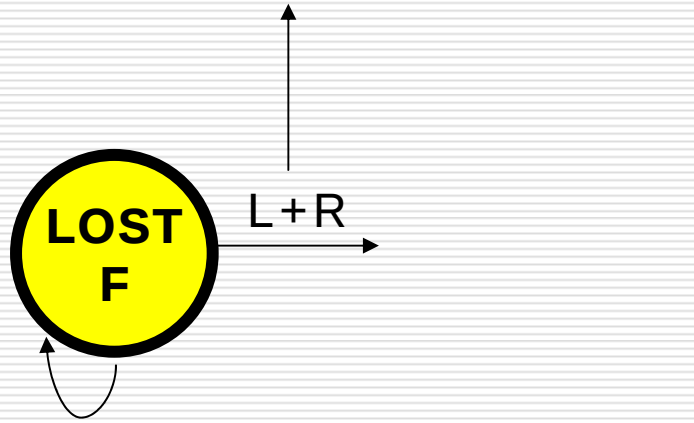


Perdu (LOST) dans l'espace

Action: aller en avant jusqu'au prochain obstacle

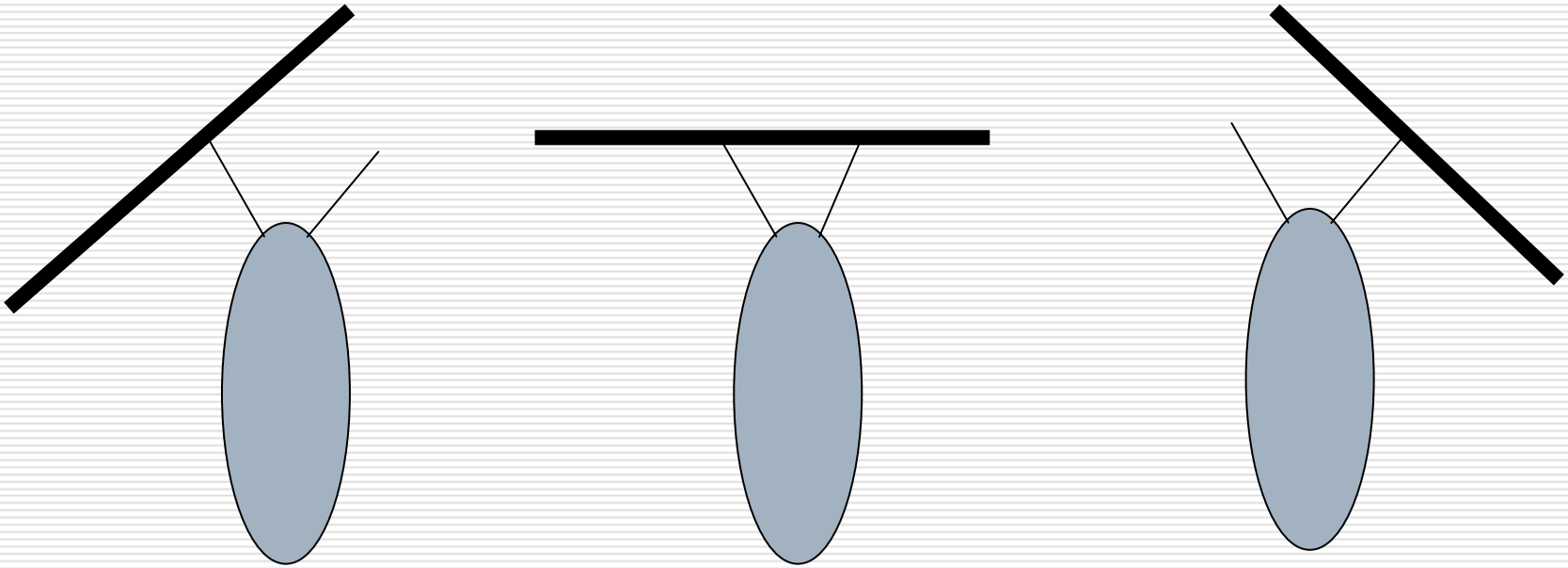


Antenne L ou antenne R = 1



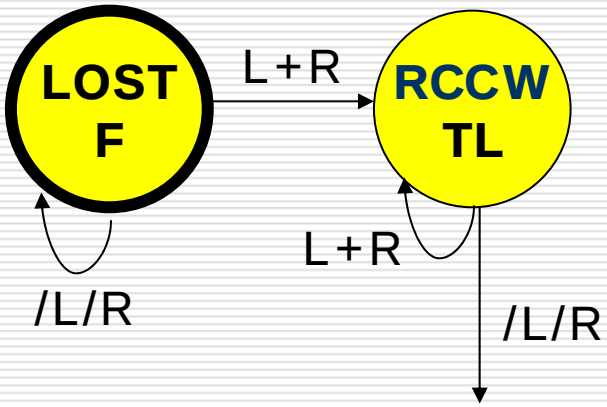
LOST est l'état initial

Obstacles



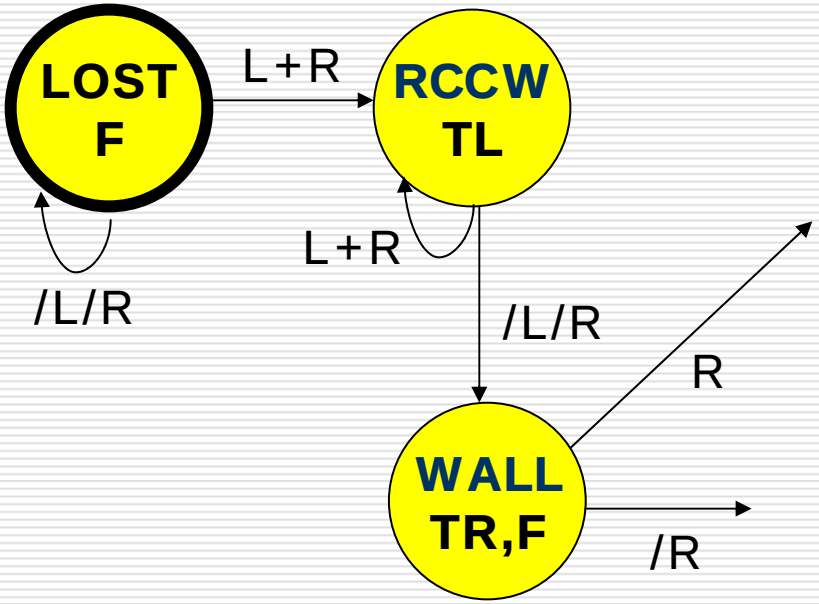
ACTION: tourner à gauche jusqu'à ce que les antennes ne touchent pas un objet

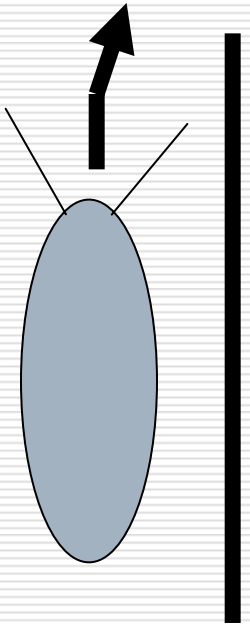
Obstacles - Suite



TCCW: Rotate Counter-ClockWise

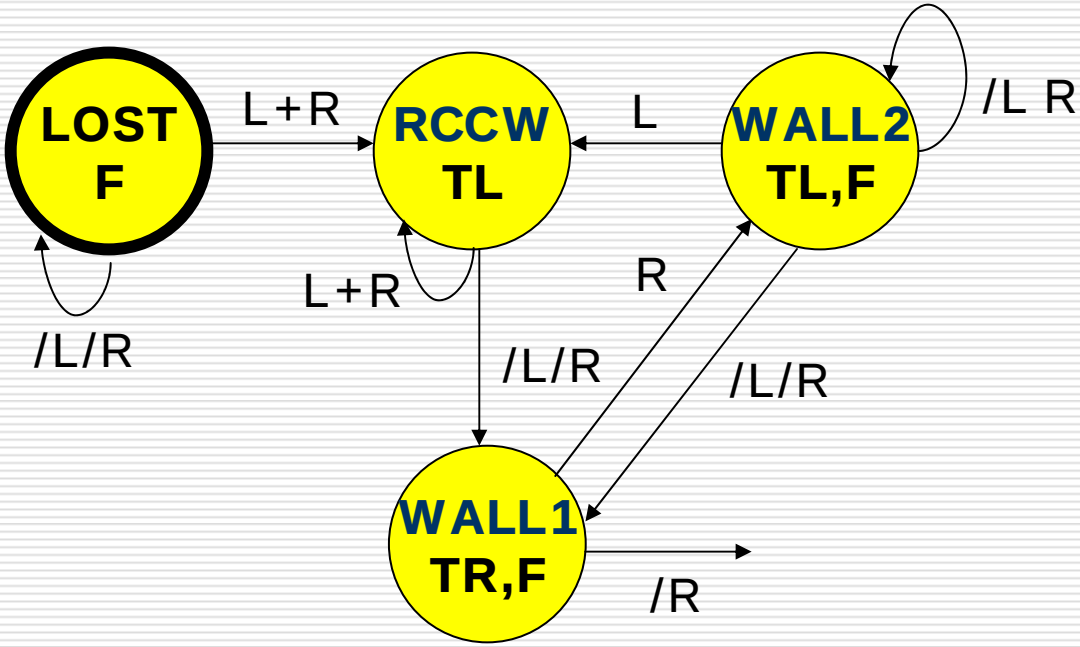
Un peu à droite

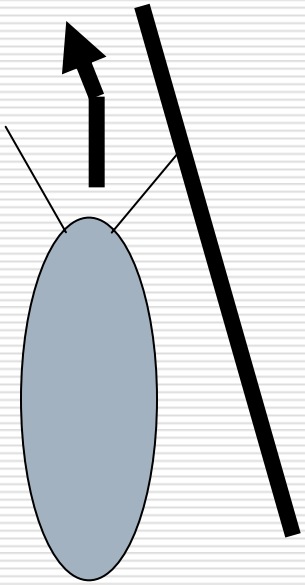




ACTION: Avance et tourne un peu à droite pour chercher le mur (wall)

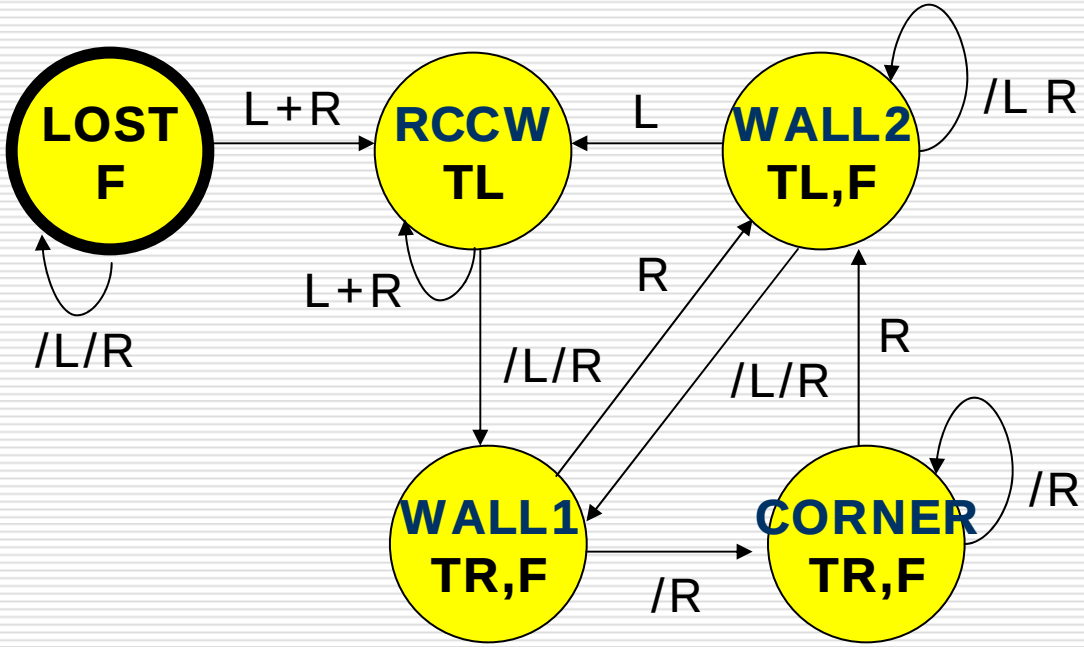
Puis un peu à gauche

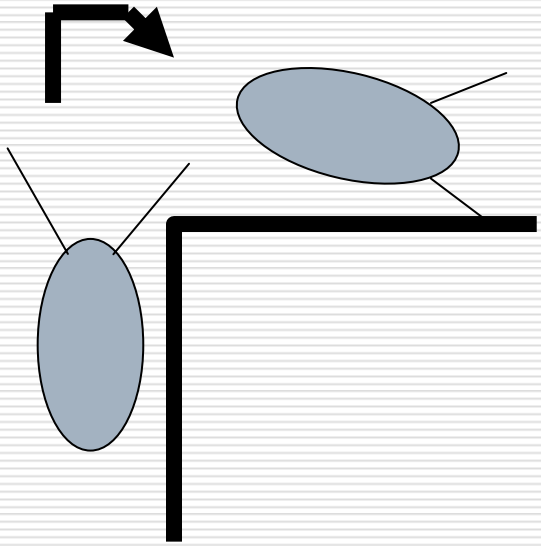




ACTION: avance et tourne un peu a gauche jusqu'à ce que l'antenne ne touche plus au mur

Pour les coins (corners)





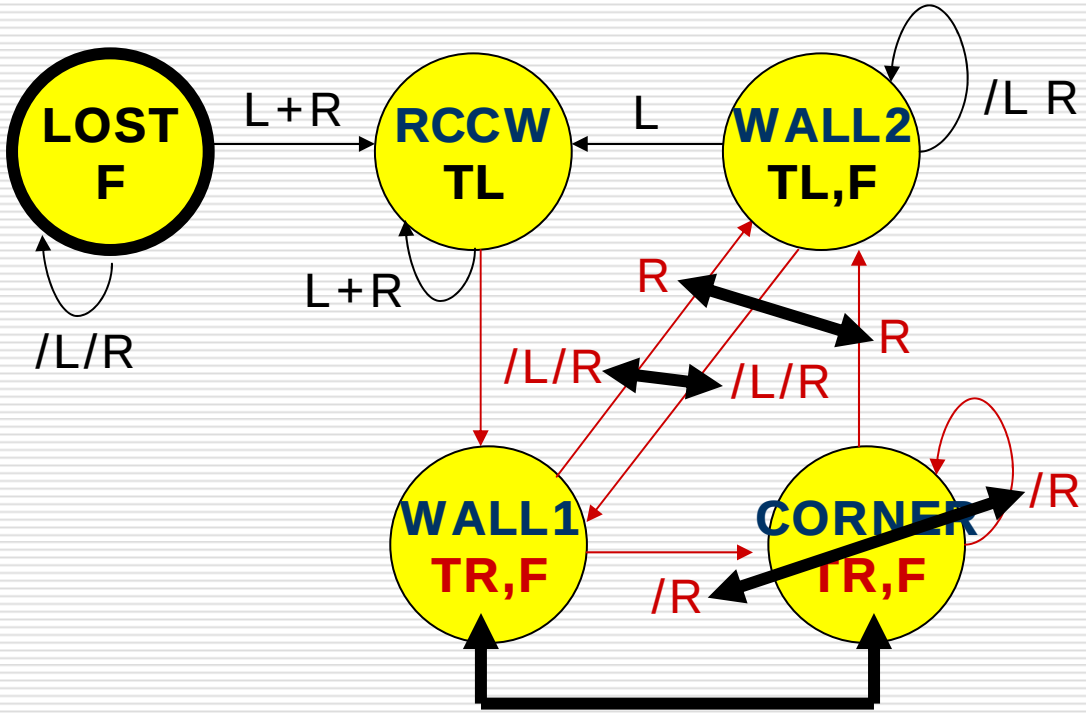
ACTION: avance et tourne à droite jusqu'à ce qu'il détecte un mur

Réduire le nombre d'états nécessaires en éliminant les états équivalents

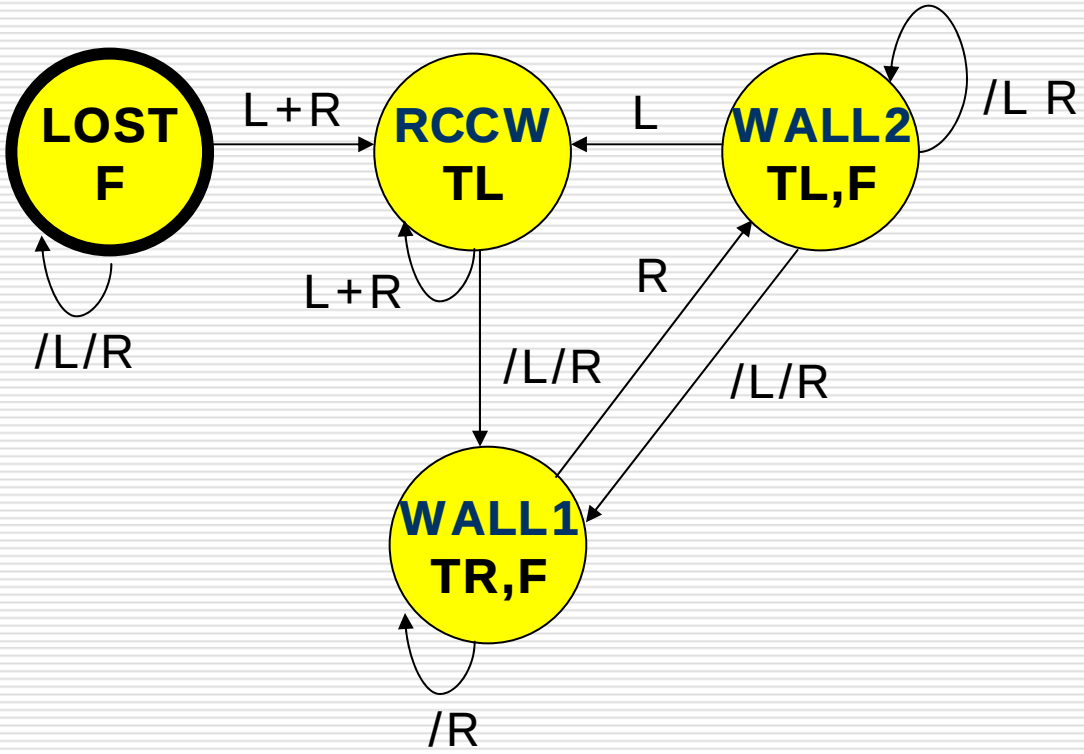
- Observation: $S_i \equiv S_j$ si
  - les états ont des sorties identiques; **ET**
  - Pour chaque entrée -> états équivalents.

- Stratégie de réduction:
  - Trouver les pairs d' états équivalents, puis les combiner.

Réduire le nombre d'états nécessaires en éliminant les états équivalents - Suite

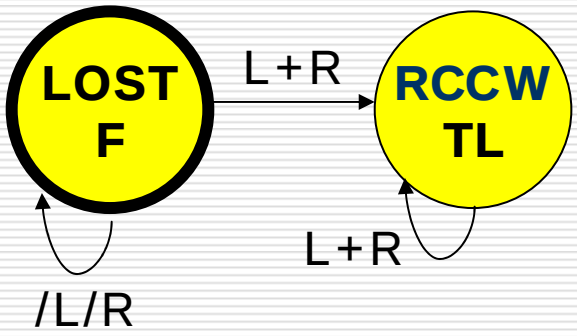


Combiner WALL1 et CORNER en un seul état nouveau WALL1



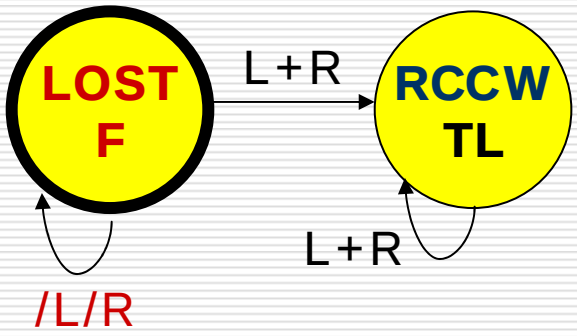
Note: Se comporte exactement comme le modèle précédent à 5 états mais a besoin de seulement la moitié du ROM pour l'implémentation.

Table d'états



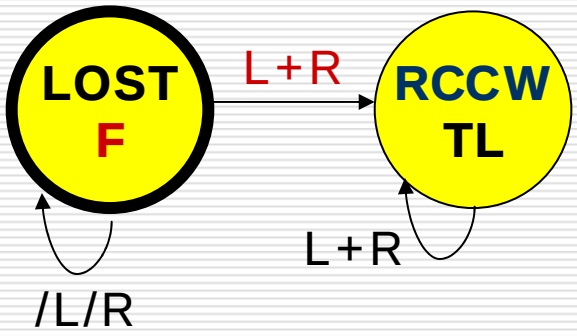
S	*L R*	*S'*	*TR*	*TL*	*F*
00	*0 0*	*00*	*0*	*0*	*1*
00	*1 -*	*01*	*0*	*0*	*1*
00	*0 1*	*01*	*0*	*0*	*1*
01	*1 -*	*01*	*0*	*1*	*0*
01	*0 1*	*01*	*0*	*1*	*0*

Table d'états



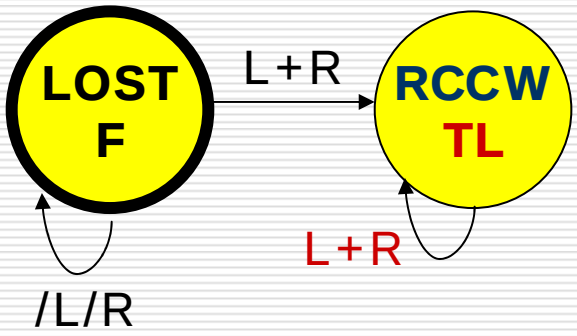
S	L R	S'	TR	TL	F
00	0 0	00	0	0	1
00	1 -	01	0	0	1
00	0 1	01	0	0	1
01	1 -	01	0	1	0
01	0 1	01	0	1	0

Table d'états



S	L R	S'	TR	TL	F
00	0 0	00	0	0	1
00	1 -	01	0	0	1
00	0 1	01	0	0	1
01	1 -	01	0	1	0
01	0 1	01	0	1	0

Table d'états



S	L R	S'	TR	TL	F
00	0 0	00	0	0	1
00	1 -	01	0	0	1
00	0 1	01	0	0	1
01	1 -	01	0	1	0
01	0 1	01	0	1	0

Implémentation

	S	L	R	S′	TR	TL	F
LOST	00	0	0	00	0	0	1
	00	1	-	01	0	0	1
	00	0	1	01	0	0	1
RCCW	01	1	-	01	0	1	0
	01	0	1	01	0	1	0
	01	0	0	10	0	1	0
WALL1	10	-	0	10	1	0	1
	10	-	1	11	1	0	1
WALL2	11	1	-	01	0	1	1
	11	0	0	10	0	1	1
	11	0	1	11	0	1	1

S1′

LR \ S_1S_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	1
10	0	0	0	1

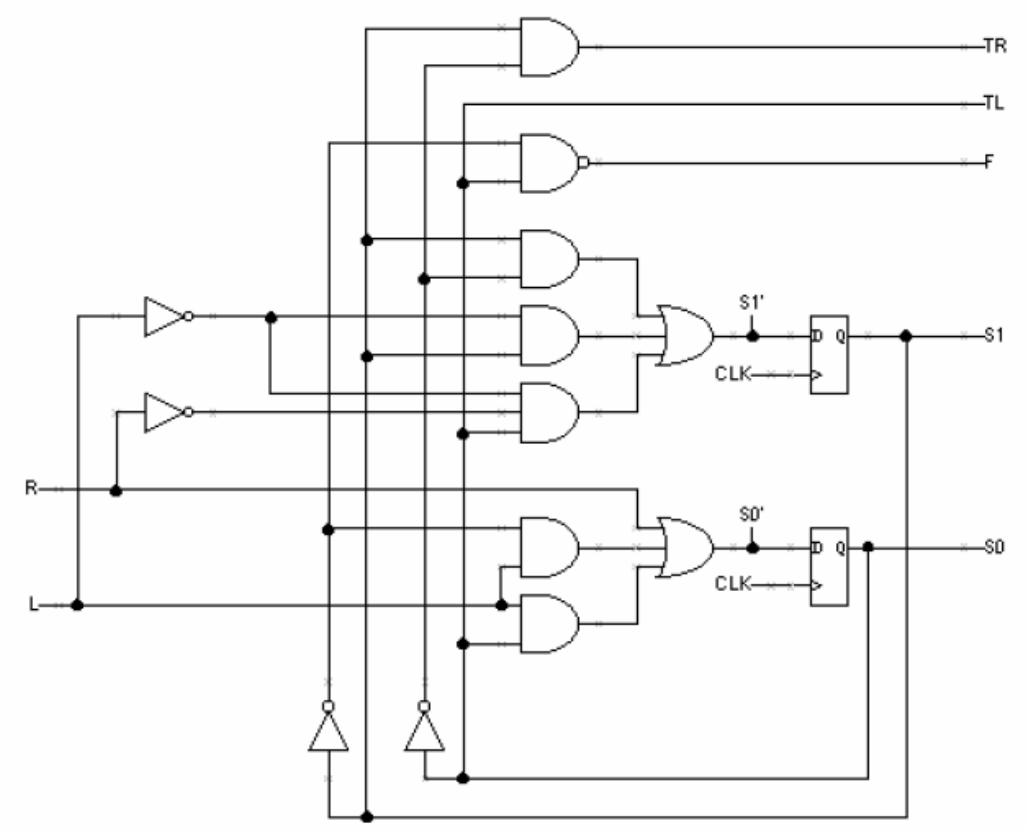
$$S_1' = S_1\overline{S_0} + \overline{L}S_1 + \overline{LR}S_0$$

S0′

LR \ S_1S_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	0

$$S_0' = R + L\overline{S_1} + LS_0$$

Schéma



Exercices

- Implanter avec un ROM et one hot
- Feux de circulation, ROM, one hot et « gates ».