Automates : quelques compléments

Jean-Christophe Le Lann

ENSTA Bretagne

23 octobre 2018

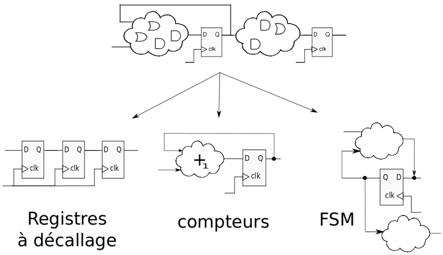
Rappels

L'Electronique numérique peut se définir comme la construction progressive de dispositifs complexes à partir d'élements simples :

- Eléments **combinatoires** : portes logiques de base (NOT,AND,OR etc voire seulement NAND ou NOR)
- Eléments **séquentiels** : bascules D et mémoires

Patterns de construction

Il existe un grand nombre de *patterns* de construction :



Toutefois, pour l'Informaticien théoricien, ces éléments séquentiels se définissent tous comme des **automates**.
Les **automates** ont donc une importance particulière dans le cadre étudié.

Définition formelle d'un **automate de Moore**

Une automate de Moore est un sextuplet $(Q, \Sigma, \Delta, \sigma, \lambda, q_0)$:

- Q est un ensemble fini d'états, q_0 est l'état initial
- Σ est l'alphabet d'entrée, Δ est l'alphabet de sortie
- δ est une application de $Q \times \Sigma$ dans Q : *fonction de transition*
- λ est une application de Q dans Δ, donnant la sortie associée à chaque état

La sortie de l'automate de Moore en réponse à une entrée $a_1 a_2 \ldots a_n$, $n \geq 0$ est $\lambda(q_0), \lambda(q_1) \ldots \lambda(q_n)$ où $q_0, \ldots, q_n$ est la séquence d'états tels que $\lambda(q_{i-1}, a_i) = q_i$ pour $1 \leq i \leq n$.
Remarque : Un automate de Moore retourne la sortie $\lambda(q_0)$ pour toute entrée.

Définition formelle d'un **automate de Mealy**

Une automate de Mealy est un sextuplet $(Q, \Sigma, \Delta, \sigma, \lambda, q_0)$:

- Q est un ensemble fini d'états, q_0 est l'état initial
- Σ est l'alphabet d'entrée, Δ est l'alphabet de sortie
- δ est une application de $Q \times \Sigma$ dans Q : *fonction de transition*
- λ est une application de $Q \times \Sigma$ dans Δ, donnant la sortie associée à chaque état

$\lambda(q, a)$ donne la sortie associée à une transition d'un état q sur l'entrée a. La sortie de l'automate de Mealy,en réponse à une séquence d'entrées $a_1, \ldots a_n$ est $\lambda(q_0, a_1)\lambda(q_1, a_2)\ldots\lambda(q_{n-1}, a_n)$ où $q_0, q_1, \ldots, q_n$ est la séquence des états tels que $\lambda(q_{i-1}, a_i) = q_i$ pour $1 \leq i \leq n$.

Représentations proches de la réalisation électronique

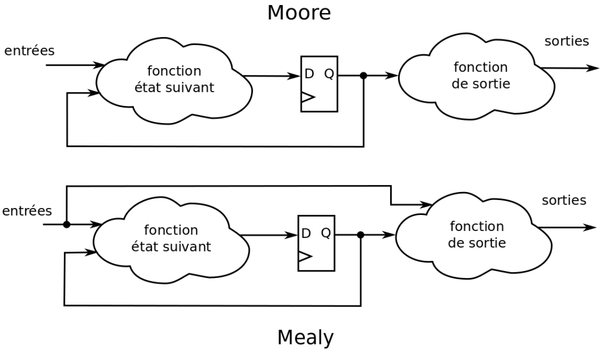


Figure – Automates de Moore et de Mealy

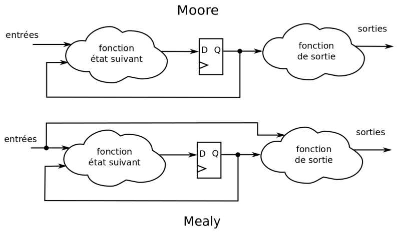
Concevoir un automate

Un problème de conception d'automate peut se résumer de la manière suivante :

1. Formulation du problème → diagramme états-transitions.
2. Vérification de la *causalité* de l'automate décrit.
3. Encodage des états.
4. Construction du tableau de séquencement (si nécessaire)
5. Détermination des fonctions d'état suivant (fonction de transition) et sorties.

$$D_i = f(Q_j, E_k)$$

$$S_l = g(Q_j, (E_k))$$



Vérification de causalité d'automate

Pour qu'un automate soit considéré comme consistant ou *causal*, on doit examiner les conditions $c_i(s)$ des transitions partant de chaque état $s \in Q$.

- **Réactivité** Les conditions sur les transitions doivent être *collectivement exhaustives* : il existe forcément une transition à '1'.

$$\forall s \in Q : \sum_i c_i(s) = 1$$

- **Déterminisme** Les conditions sur les transitions doivent être *mutuellement exclusives* : on ne peut pas avoir deux transitions à '1' en même tem ps.

$$\forall s \in Q, \forall (i,j), i \neq j : c_i(s).c_j(s) = 0$$

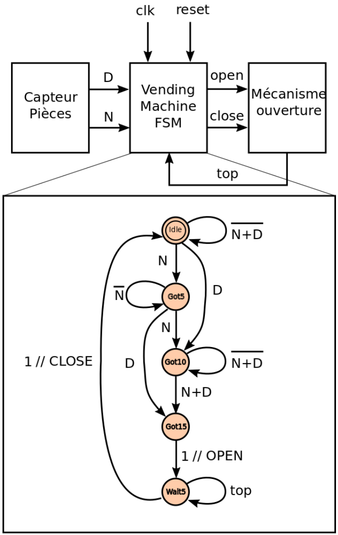
Vérification de causalité d'automate (suite)

Cela signifie simplement que dans un état, il existe un, et un seul, état suivant.
Notons qu'un état peut avoir comme état suivant lui-même : dans ce cas, le système décrit ne change pas d'état, tout simplement. En terme de diagramme états-transitions, cela revient à dessiner une boucle sur l'état (la "bulle").

Etude de cas : vending machine (distributeur)

On cherche à réaliser un dispositif qui délivre un objet (cannette etc), après que l'on y a introduit 15 centimes. Ces 15 centimes peuvent être introduits à l'aide de pièces de 5 ou 10 centimes. La machine ne rend pas la monnaie. Concevoir cette machine.

Etude de cas : diagramme états-transitions

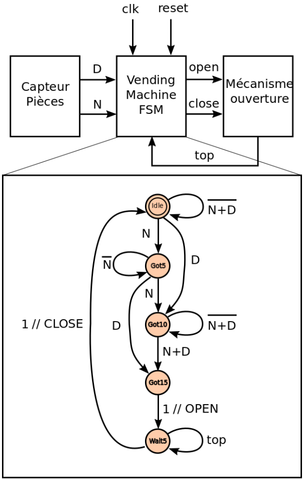


Etude de cas : encodage des états

On rappelle que l'encodage des états consiste à associer un code binaire à chaque nom symbolique des états.

- Plusieurs encodages sont possibles.
- Chaque encodage conduit évidemment à des circuits différents structurellement, mais leur comportement aux entrées-sorties est identique.
- On retiendra deux encodages particuliers :
  1. Encodage dense : numérotation binaire classique.
  2. Encodae "one-hot" : 1 bit par état. $1, 2, 4, 8, 16, 32, \ldots$
- L'encodage "one-hot" permet d'éviter les tableaux de Karnaugh.

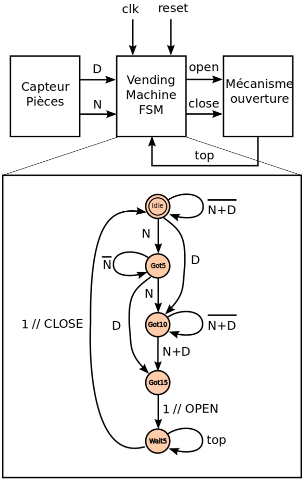
Etude de cas : Encodage dense



Encodage dense :

état	numéro	code binaire
Idle	0	000
Got5	1	001
Got10	2	010
Got15	3	011
Wait5	4	100

Etude de cas : encodage "one-hot"



Encodage "one-hot" :

état	numéro	code binaire
Idle	1	00001
Got5	2	00010
Got10	4	00100
Got15	8	01000
Wait5	16	10000

Etude de cas : table de séquencement

Le but est de trouver les équations :

$$D_i = f(Q_j, E_k)$$

$$S_l = g(Q_j, (E_k))$$

Dans un premier temps on conserve les noms des états symboliques.

état courant	e_1	...	e_n	état suivant	s_1	...	s_n
IDLE							
...							
GOT5							
...							

Puis on utilise l'encodage explicitant les Q_i (état courant) et D_i (état futur).

Q_n	...	Q_0	e_1	...	e_n	D_1	...	D_0	s_1	...	s_n

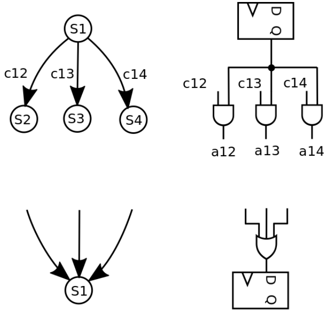
Etude de cas : équations finales

Il reste à observer les colonnes des D_i et s_j et déduire les équations logiques

- Déduction immédiate dans certains cas.
- Construction des tableaux de Karnaugh pour chaque D_i et s_j

Cas de l'encodage One-Hot

On peut *bypasser* les étapes précédentes dans le cas d'un automate dont les états sont encodés One-Hot, et déduire les équations par simple observation du diagramme états-transitions et application du schéma de traduction suivant :



Cas de l'encodage One-Hot

- On dessine une bascule D pour un état S
- Pour chacune des transitions sortantes d'un état S_i vers un état S_j, on dessine une porte AND connectée d'une part à la sortie Q de la bascule de l'état S_i, et d'autre part la condition booléenne c de la transition $S_i \rightarrow_c S_j$. Appelons $c_{i,j}$ la sortie de cette porte AND.
- Pour chacune des transitions entrantes d'un état S_j on dessine une porte OR connectée à l'entrée D_j et possédant autant d'entrées qu'il existe de transitions entrantes. A partir des états précedents S_i, on connecte le signal $c_{i,j}$ sur une des entrées de cette porte OR.

Dessin du circuit

Dans certains cas simples, il reste possible de dessiner le circuit.