ELECINF 102 : Processeurs et Architectures Numériques

Automates finis: Réalisation matérielle de séquenceurs

Sumanta Chaudhuri, Jean-Luc Danger, Guillaume Duc,
Tarik Graba, Ulrich Kühne, Yves Mathieu
Alexis Polti, Laurent Sauvage

Plan

Automates materiels

Méthodologie

Mise en œuvre en SystemVerilog

Encodage des états

Temps de réaction

TELECOM Paris

Automates finis

Vocabulaire

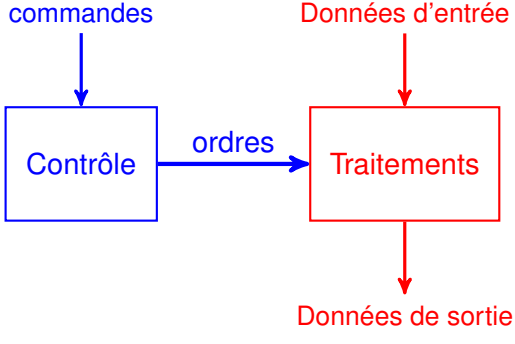
- Automates finis
- Machines à états finis (MAE)
- Finite–State Machines (FSM)
- Ce qui est fini, c'est le nombre d'états

Automates finis

Pourquoi ?

Dans une architecture, on distingue traditionellement :

- le **traitement** des données,
- le **contrôle** de ce traitement.



Automates finis

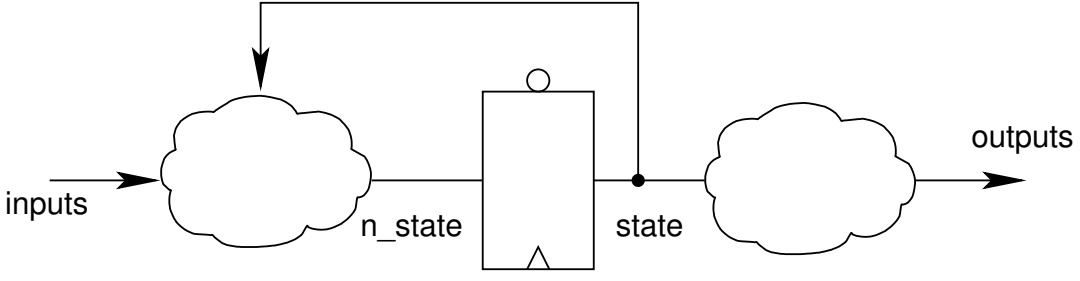
Les automates sont une solution à la réalisation des blocs de **contrôle**.

- Dans un automate fini on définit :
  - l'**état** d'un système,
  - les actions effectuées dans un état déterminé,
  - le passage d'un état à l'autre (les **transitions**).

Automates finis en matériel

- Mémoriser l'état $\rightarrow$ **logique synchrone** :
  - un **registre** pour mémoriser l'état :
    - La taille du registre est liée au **nombre** d'états.
    - La réinitialisation du registre (**reset**) permet de maitriser l'état de départ.
- En fonction de l'**état courant** calculer combinatoirement les sorties (les **ordres**)
- En fonction de l'**état courant** et des entrées (les **commandes**) calculer combinatoirement l'**état futur**

Automates finis en matériel



- La sortie du registre représente l'état courant.
- L'entrée du registre représente l'état futur.
- Au front d'horloge, l'état courant est mis à jour.

Cette architecture est appelée **machine à états de Moore**

Plan

Automates materiels

Méthodologie

Mise en œuvre en SystemVerilog

Encodage des états

Temps de réaction

Avant de coder...

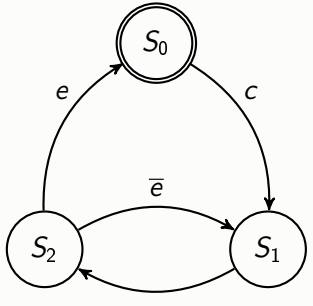
Une méthode générique de conception

- On trace un graphe :
  - On construit un graphe orienté.
  - Chaque état est représenté par un cercle.
  - On précise l'état initial par un double cercle (après un reset).
  - On précise pour chaque état la valeur des sorties.
  - Les états sont liés par des flèches qui représentent les transitions.
  - La condition de chaque transition doit être précisée.
  - Ces conditions dépendent des entrées.

Automates finis

Une méthode générique de conception

Exemple



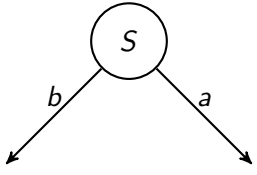
- L'état initial est S_0
- Le passage de S_0 à S_1 ne se fait que si la condition c est vraie
- Le transition de S_1 à S_2 est inconditionnelle (toujours vraie)
- À partir de S_2 on passe à S_0 ou S_1 en fonction de e
- Si un transition n'est pas prise, on reste dans l'état courant.

REMARQUE : Pour ne pas encombrer le schéma, nous ne représentons pas le fait de rester dans l'état si aucune transition n'est prise.

Le style de codage en SystemVerilog que nous utiliserons garantira cela.

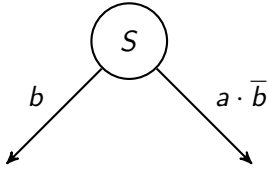
Graphe non contradictoire

Deux transitions avec des conditions contradictoires ne doivent pas partir du même état.



Graphe non contradictoire

Deux transitions avec des conditions contradictoires ne doivent pas partir du même état.



Plan

Automates materiels

Méthodologie

Mise en œuvre en SystemVerilog

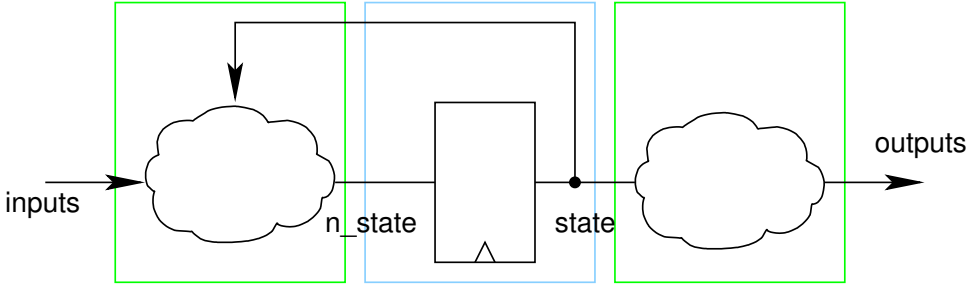
Encodage des états

Temps de réaction

Codage en SystemVerilog

Trois blocs

- Un bloc séquentiel pour sauvegarder l'état.
- Deux blocs combinatoires : calcul de l'état futur ; calcul des sorties.



Représentation en SystemVerilog

Style de codage des 3 blocs

Déclaration des signaux du registre d'états

```
enum logic[1:0] {INIT, S0, S1} state, n_state ;
```

Calcul de l'état futur

```
always @(*)
begin
   // par défaut on reste
   // dans l'état courant
  n_state <= state ;
  case (state)
   INIT: if (cond0)
           n_state <= S0;
   S0  : if (cond1)
           n_state <= S1;
   S1  : if (cond2)
           n_state <= INIT;
  endcase
end
```

Stockage de l'état courant

```
always @(posedge clk)
if (reset)
    state <= INIT ;
else
    state <= n_state ;
```

Calcul des sorties

```
always @(*)
    output1 <= f(state);

always @(*)
    output2 <= g(state);

always @(*)
    output3 <= ...
...
```

Représentation en SystemVerilog
Déclaration des signaux du registre d'états

```
enum logic[1:0] {INIT, S0, S1} state, n_state ;
```

- Pour la lisibilité utilisation d'un type énuméré
  - mot clé **enum**
  - permet de nommer les états
- on précise la taille du registe
- on déclare en même temps les signaux représentant l'état et l'état futur

Représentation en SystemVerilog
Le registre d'état

```
always @(posedge clk)
if (reset)
    state <= INIT ;
else
    state <= n_state ;
```

- Le reset met l'automate dans son état initial
- À chaque front d'horloge on change l'état
  - l'état futur remplace l'état actuel

Représentation en SystemVerilog
Transitions d'états

```
always @(*)
begin
   // par défaut on reste
   // dans l'état courant
  n_state <= state ;
  case (state)
   INIT: if (cond0)
            n_state <= S0;
   S0  : if (cond1)
            n_state <= S1;
   S1  : if (cond2)
            n_state <= INIT;
  endcase
end
```

- Par défaut l'état ne change pas
  - l'état futur reste à l'état courant
- Pour chaque état
  - n'exprimer que ce qui entraine un changement d'état
- Le case peut être incomplet car il y a un cas par défaut

Représentation en SystemVerilog
Calcul des sorties

- Les sorties sont calculées combinatoirement
  - elles ne dépendent que de l'état courant
- Un style de codage, dans lequel on part de la sortie est conseillé.

```
always @(*)
    output1 <= f(state);

always @(*)
    output2 <= g(state);

always @(*)
    output3 <= ...
...
```

Plan

Automates materiels

Méthodologie

Mise en œuvre en SystemVerilog

Encodage des états

Temps de réaction

Encodage binaire naturel

- Chaque état est représenté par une valeur du registre d'état
- La taille du registre évolue en $\log_2$ du nombre d'états
- C'est comme ça que fonctionnent les **`enum`** de SystemVerilog

Dans notre exemple :

```
enum logic[1:0] {INIT, S0, S1} state, n_state ;
```

- 3 états $\rightarrow$ 2 bits.
  - `INIT` $\rightarrow$ `2'b00`
  - `S0` $\rightarrow$ `2'b01`
  - `S1` $\rightarrow$ `2'b10`

Encodage OneHot

- Chaque état est représenté par un bit du registre d'états
  - Il y a systématiquement qu'un seul bit à 1
- La taille du registre est égale au nombre d'états

Dans notre exemple :

- 3 états $\rightarrow$ 3 bits.
  - `INIT` $\rightarrow$ `2'b001`
  - `S0` $\rightarrow$ `2'b010`
  - `S1` $\rightarrow$ `2'b100`

Encodage OneHot
Avantages/Inconvénients

- Le décodage des états est simplifié au maximum
  - Pour déterminer si on est dans un état particulier, il suffit de regarder la valeur du bit correspondant.
- Simplification de la logique combinatoire.

Par contre :

- On augmente la taille du registre d'état.

Il y a donc un compromis entre la taille de la logique combinatoire et la taille de la logique séquentielle.

Encodage OneHot
Et en SystemVerilog ?

Sauf cas particuliers, il faut laisser les outils informatiques décider si le passage en encodage OneHot est opportun.

- On garde le style de codage précédemment présenté
  - avec une taille de registre compatible avec l'encodage binaire naturel

Les outils de synthèse détectent les registres d'états et peuvent changer l'encodage

Plan

Automates materiels

Méthodologie

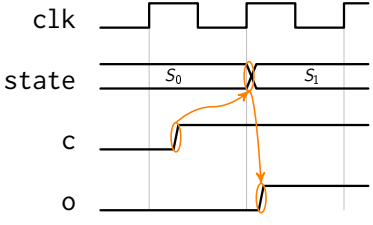
Mise en œuvre en SystemVerilog

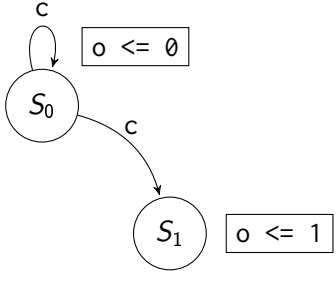
Encodage des états

Temps de réaction

Latence de réaction pour une machine de Moore

Le changement de la valeur d'une sortie nécessite un changement d'état et donc au moins un cycle de latence.

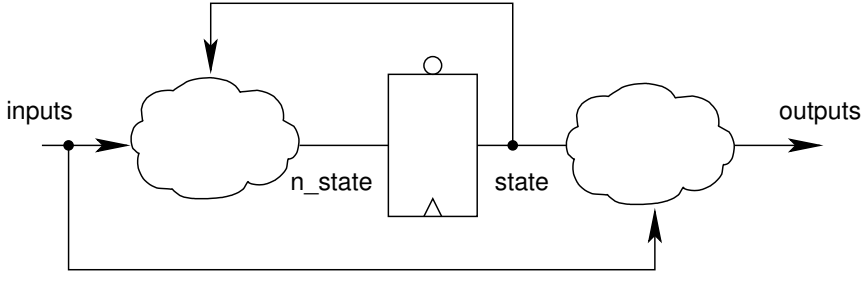




Une variante de la Machine de Moore

La machine de Mealy

- La sortie dépend de l'état mais aussi directement des entrées

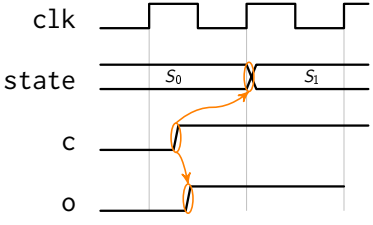


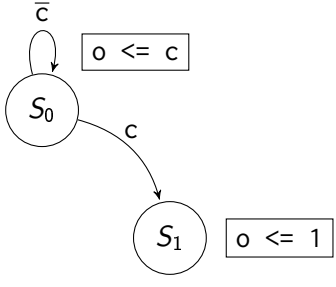
Ceci permet de :

- changer la sortie sans changer d'état,
- changer la sortie en même temps qu'on change d'état.

Latence de réaction pour une machine de Mealy

Un changement d'entrée peut être propagé immédiatement sur une sortie.





Moore *vs* Mealy

Faut-il faire du Mealy systématiquement ?

- Avec une machine de Mealy on peut réagir plus vite.
- Mais on a créé un chemin combinatoire entre l'entrée et la sortie
  - la maitrise du chemin critique plus complexe.