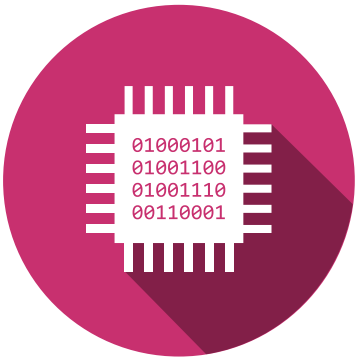


Machines d'état (FSM)

Cours Conception Numérique

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Orientation : Énergie et techniques environmentales (ETE)
Cours : Conception Numérique (Cnum)
Auteur : Christophe Bianchi, François Corthay, Pierre Pompili, Silvan Zahno
Date : 25 août 2022
Version : v2.1

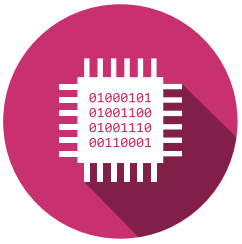
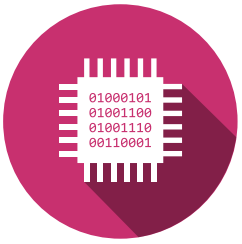


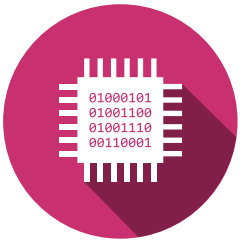
Table des matières



1 Introduction

Les circuits de commande se font généralement à l'aide de **machines d'états**.

On distingue les machines de Moore, où les sorties dépendent uniquement des états internes, et les machines de Mealy, où les sorties dépendent des états internes et des entrées.



2 Systèmes logiques synchrones

2.1 Définition : système logique synchrone

Un système logique synchrone est caractérisé par le fait que toutes les sorties des bascules peuvent uniquement changer au moment d'un flanc de l'horloge maîtresse du système. Ceci implique que

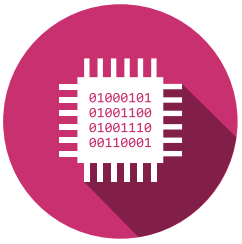
- toutes les bascules ont le même signal d'horloge,
- aucune bascule n'a d'entrées asynchrones.

Pour des raisons d'initialisation et de testabilité, il est usuel de connecter une commande de mise à zéro à toutes les bascules. Dans le cas où la commande de remise à zéro provient de l'extérieur du circuit et est identique pour toutes les bascules, on considère tout de même le système comme synchrone.

2.2 Signal d'horloge

Le signal d'horloge provient d'un oscillateur. Il a une fréquence régulière et n'est pas à bloquer par des portes logiques dans le but d'arrêter le circuit pendant un certain temps.

Généralement, le signal d'horloge a une fréquence bien plus élevée que celle des signaux d'entrée du système logique synchrone.



3 Machines de Moore

3.1 Architecture

Une Finite State Machine, FSM (machine d'état) de Moore est un système logique synchrone comprenant

- des bascules qui mémorisent un état interne,
- un bloc logique combinatoire qui détermine l'état interne futur en fonction de l'état interne présent et des entrées,
- un bloc logique combinatoire qui détermine la valeur des sorties en fonction de l'état interne présent.

La figure 1 représente un exemple de machine d'état de Moore.

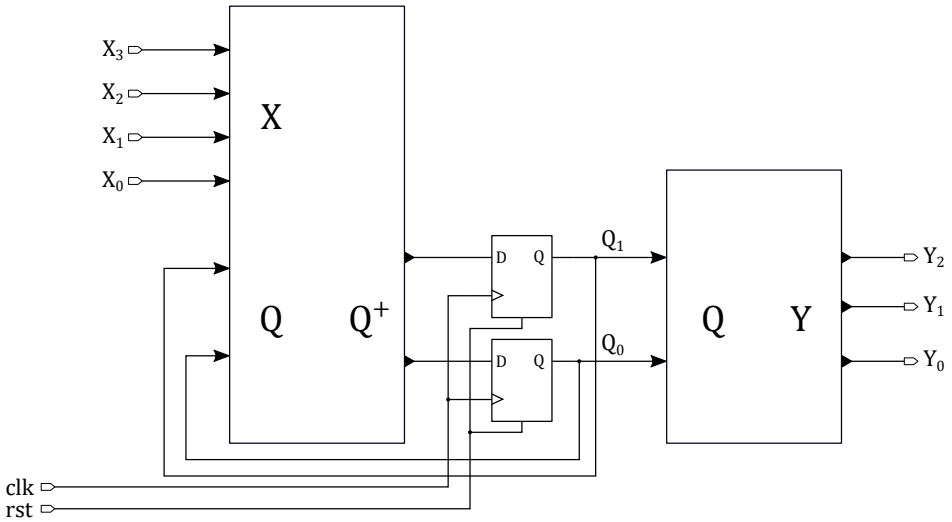


FIGURE 1 – Architecture d'une machine d'état de Moore

3.1.1 Définition : état d'entrée

Un input state (état d'entrée) est une combinaison de valeurs des signaux d'entrée.

Pour un système à n entrées, il y a 2^n états d'entrée possibles.

3.1.2 Définition : état interne

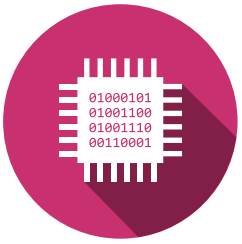
Un internal state (état interne) est une combinaison de valeurs des sorties des bascules.

Pour un système à m bascules, il y a 2^m états internes possibles.

3.1.3 Définition : état de sortie

Un output state (état de sortie) est une combinaison de valeurs des signaux de sortie.

Pour un système à l sorties, il y a 2^l états de sortie possibles.



3.2 Table de vérité des blocs combinatoires

Le fonctionnement d'une machine d'état peut s'étudier à l'aide d'une table de vérité qui donne, pour chaque état d'entrée et chaque état interne présent, l'état interne futur et l'état de sortie.

Exemple : compteur cascadable

La figure 2 présente le schéma d'un compteur cascadable.

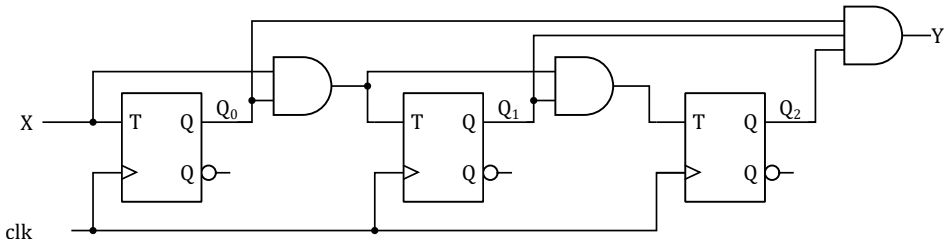


FIGURE 2 – Compteur cascadable

La table 1 est la table de vérité correspondante.

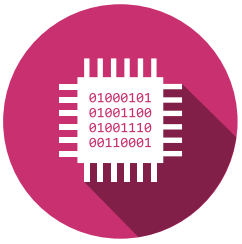
X	$Q_{2...0}$	$T_{2...0}$	$Q^+{}_{2...0}$	Y
0	000	000	000	0
0	001	000	001	0
0	010	000	010	0
0	011	000	011	0
0	100	000	100	0
0	101	000	101	0
0	110	000	110	0
0	111	000	111	1
1	000	001	001	0
1	001	011	010	0
1	010	001	011	0
1	011	111	100	0
1	100	001	101	0
1	101	011	110	0
1	110	001	111	0
1	111	111	000	1

TABLE 1 – Table de vérité des blocs combinatoires du compteur cascadable

3.3 Graphe des états

La table de vérité des blocs combinatoires permet de dessiner le graphe des états. Celui-ci donne une représentation plus compréhensible du fonctionnement de la machine d'états.

Une machine d'états à m bascules comprend 2^m états. Chaque état est représenté par un cercle. Si la machine possède n entrées, il y a 2^n cas à spécifier : de chaque état partent 2^n flèches. Pour une machine de Moore, les sorties dépendent uniquement des états. Le plus simple est donc de représenter la valeur que prennent les sorties dans le cercle correspondant à l'état.



Exemple : compteur cascadable

Le graphe des états du compteur cascadable, donné à la figure 3, se développe à partir de la table de vérité des blocs combinatoires donnée à la table 1.

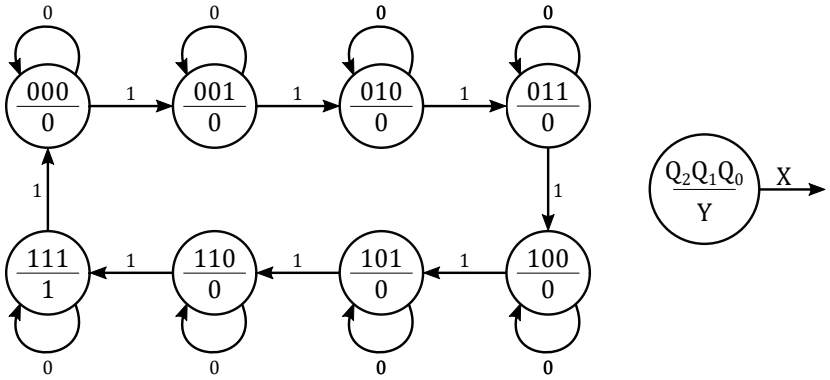


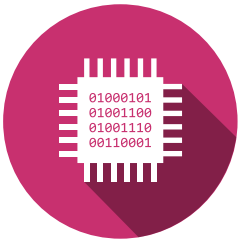
FIGURE 3 – Graphe des états du compteur cascadable

On observe qu'avec $X =$ '0', le compteur reste arrêté et qu'avec $X =$ '1' il tourne régulièrement. Lorsqu'il arrive à la valeur maximale, le compteur fait un report $Y =$ '1' pour la mise en cascade.

Pour la représentation du graphe des états, il est important de spécifier l'ordre dans lequel les signaux sont annotés.

3.4 Définition : état total stable

Un état total stable est un état dans lequel la machine reste arrêtée, pour un des états d'entrée au moins. Dans le graphe des états, d'un état total stable part une flèche qui revient sur lui-même.



4 Machines de Mealy

4.1 Architecture

A la différence des machines de Moore, les machines de Mealy ont des sorties qui peuvent dépendre aussi des entrées. La figure 4 représente un exemple de machine d'état de Mealy.

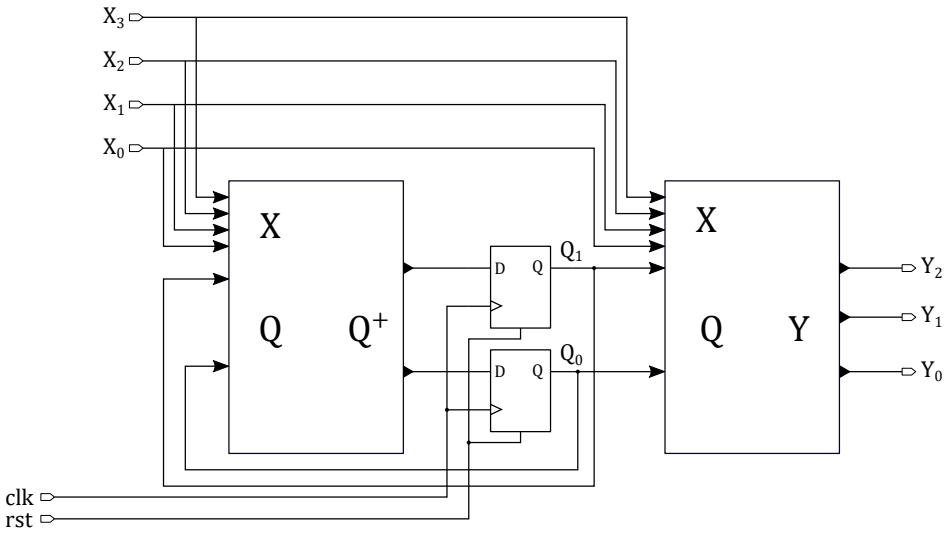


FIGURE 4 – Architecture d'une machine d'état de Mealy

4.2 Comportement temporel

La figure 5 présente le comportement temporel des signaux de sortie d'une machine de Mealy.

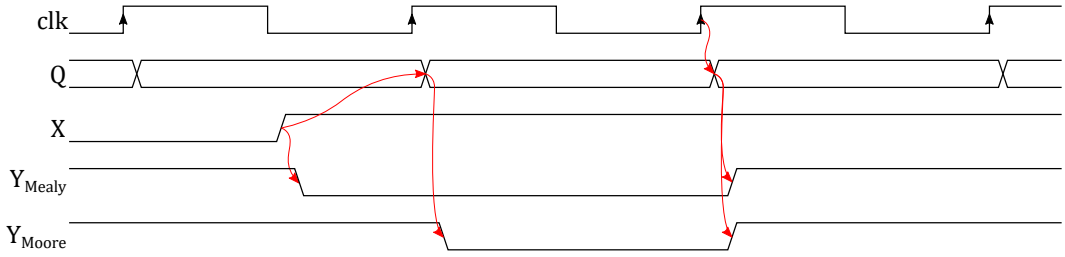


FIGURE 5 – Comportement temporel d'une machine de Mealy

Avec une machine de Mealy, les sorties peuvent réagir directement à la variation d'une entrée, alors qu'avec une machine de Moore elles ne changent qu'après un flanc actif de l'horloge.

4.3 Graphe des états

Le graphe des états d'une machine de Mealy doit montrer comment les sorties dépendent aussi des entrées. On y représente la valeur des sorties à côté de celle des entrées.

Exemple : compteur cascadable avec inhibition du report

La figure 6 présente un compteur cascadable où le signal de report est mis à Y = '0' lorsque l'entrée est à X = '0'.

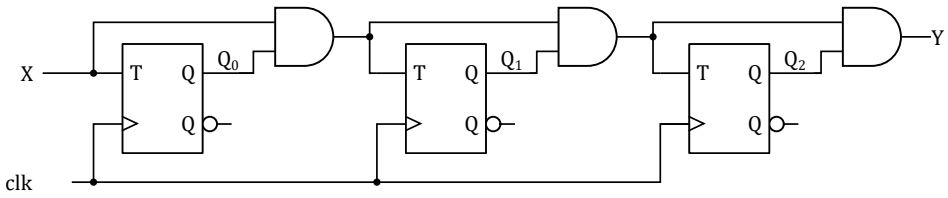


FIGURE 6 – Compteur cascadable avec inhibition du report

Le graphe des états correspondant est donné à la figure 7.

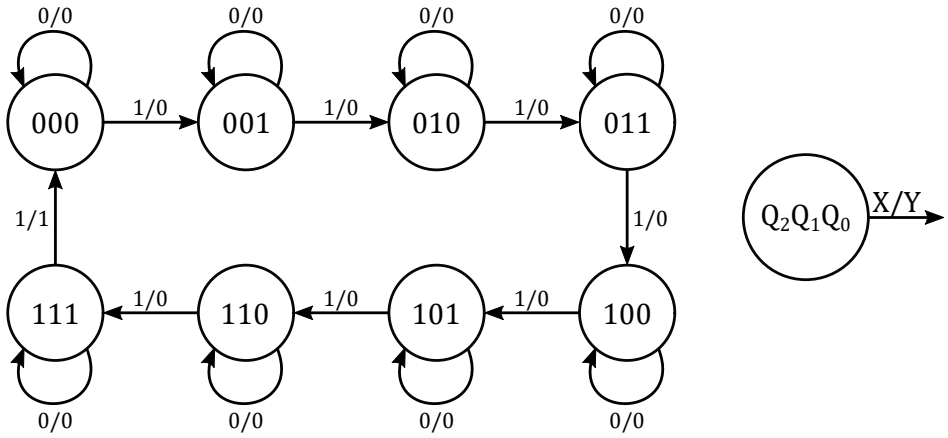
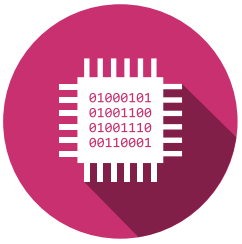


FIGURE 7 – Graphe des états du compteur cascadable avec inhibition du report

La valeur des sorties indiquée à côté d'une flèche est valable dans l'état d'où part la flèche et non dans celui où elle va.



5 Établissement du graphe des états

5.1 Conception de machine d'état

La conception de machine d'état peut se faire comme suit :

- expression du cahier des charges par un graphe d'états,
- remplissage de la table des états,
- réduction du nombre d'états,
- codage des états,
- réalisation du circuit logique.

Le point le plus délicat consiste à développer un graphe d'états à partir du cahier des charges du système. A cet effet, 3 méthodes sont présentées ci-après.

5.2 Développement à partir d'un état quelconque

La première manière de développer un graphe d'états se fait en procédant état par état. Pour cela, il faut

- Spécifier un état déterminé de la machine.
- Pour cet état, analyser toutes les conditions d'entrées possibles (2^n cas pour n entrées) et dessiner les flèches correspondantes. Selon les cas, de nouveaux états apparaissent.
- Prendre un par un les états non encore traités, analyser toutes les conditions d'entrées possibles (2^n cas pour n entrées) et dessiner les flèches correspondantes. Chaque fois qu'un nouvel état apparaît, vérifier qu'il n'existe pas déjà dans le graphe en cours de construction. Répéter cette étape jusqu'à ce que le graphe soit complet.

Il peut exister des cas où certaines conditions d'entrée ne peuvent se produire. Ceci est typiquement le cas si, par construction, plusieurs signaux d'entrée ne peuvent changer simultanément. Dans ce cas, on a moins 2^n de flèches qui partent de chaque état.

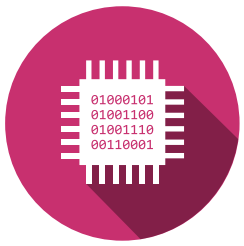
Exemple : circuit anti-rebonds

Le circuit anti-rebonds à concevoir est donné par le cahier des charges suivant : le circuit ne transmet le signal d'entrée que lorsque celui-ci n'a pas varié pendant les derniers trois coups d'horloge, sinon le circuit conserve la même valeur de sortie que précédemment.

Clairement, ceci est une machine de Moore : une variation a court terme de l'entrée ne doit aucunement affecter la sortie.

Spécifions un état déterminé de la machine : le signal d'entrée était à '0' pendant suffisamment longtemps, et la sortie est à '0'.

FIGURE 8 – Développement du graphe : état initial



L'analyse de toutes les conditions d'entrée possibles donne 2 cas : soit le signal d'entrée est à '0', et on reste dans le même état, soit le signal est à '1', et on change d'état pour se rapprocher de l'état où le signal de sortie pourra basculer à '1'.

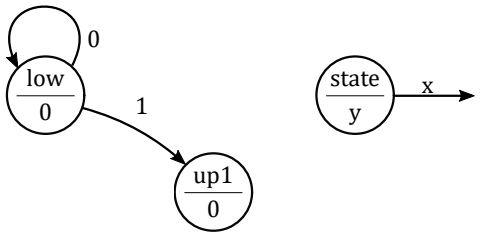


FIGURE 9 – Développement du graphe : analyse de toutes les conditions d'entrée

Un nouvel état est apparu. Il faut donc à nouveau analyser toutes les conditions d'entrée possibles pour ce nouvel état.

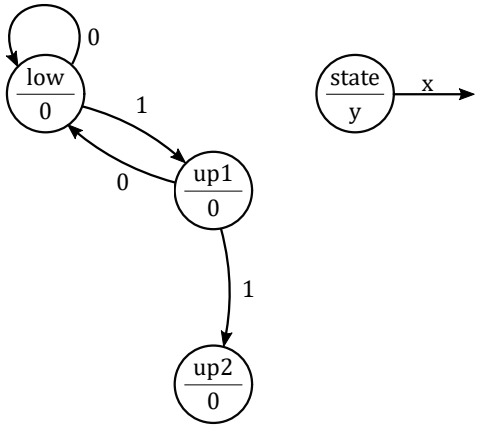
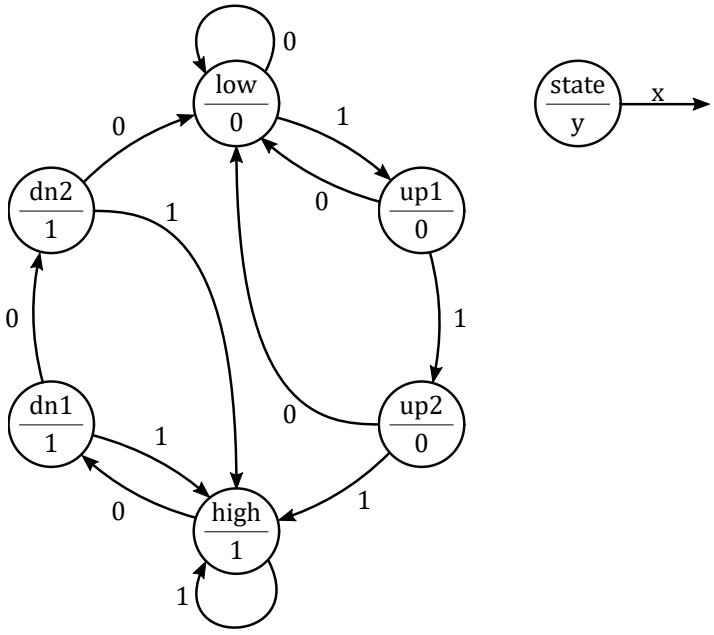


FIGURE 10 – Développement du graphe, suite

Par le développement successif des différents états avec l'analyse de toutes les conditions d'entrée possibles, on obtient le graphe de la figure 11



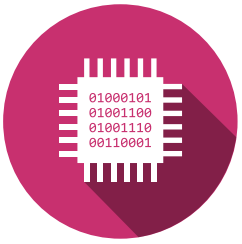


FIGURE 11 – Graphe des états du circuit anti-rebonds

5.3 Développement à partir d'un scénario

La deuxième manière de développer un graphe d'états se base sur la définition d'un scénario. Pour cela, il faut

- Définir un scénario couvrant une large part du cahier des charges.
- Développer le scénario sous forme de graphe, avec les états correspondant aux étapes du scénario et les flèches correspondant aux actions.
- Pour chaque état ainsi défini, analyser toutes les conditions d'entrées possibles (2^n cas pour n entrées) et dessiner les flèches non encore présentes. Parfois, de nouveaux états apparaissent. Il faut alors aussi faire l'analyse complète de ces nouveaux états.

Exemple : circuit anti-rebonds

Un scénario simple pour le développement du graphe du circuit anti-rebonds est le suivant : le signal d'entrée reste à '0' pendant suffisamment longtemps pour que la sortie soit à '0', le signal passe à '1', y reste suffisamment longtemps pour que la sortie le suive, puis il retombe à '0'. Ceci s'exprime par le graphe de la figure 12.

Le graphe n'est pas encore complet. Il reste à analyser pour chacun des états toutes les conditions d'entrées possibles et à compléter le graphe en conséquence. On obtient alors le graphe de la figure 11.

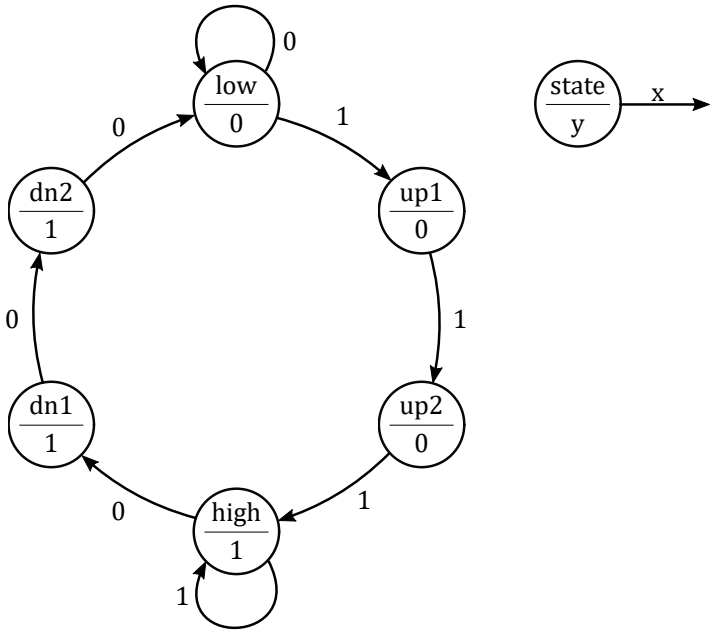
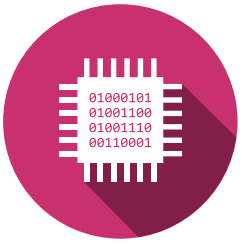


FIGURE 12 – Scénario de fonctionnement du circuit anti-rebonds

5.4 Développement à partir de la liste des états

La troisième manière de développer un graphe d'états se base sur la connaissance préalable de ses états. Pour cela, il faut

- Déterminer les événements ou les états à mémoriser.



- Pour chaque événement à mémoriser, dessiner un état.
- Pour chaque état ainsi défini, analyser toutes les conditions d'entrées possibles (2^n cas pour n entrées) et dessiner les flèches correspondantes.

5.4.1 Exemple : circuit anti-rebonds

Déterminons les choses à mémoriser : pour savoir si le signal n'a pas varié, il faut connaître les 2 dernières valeurs du signal d'entrée en plus de sa valeur présente ; aussi, si le signal a varié, il faut se rappeler de la valeur précedente de la sortie. Il faut donc se souvenir de 3 valeurs binaires, ce qui nécessite 8 états, représentés à la figure 13.

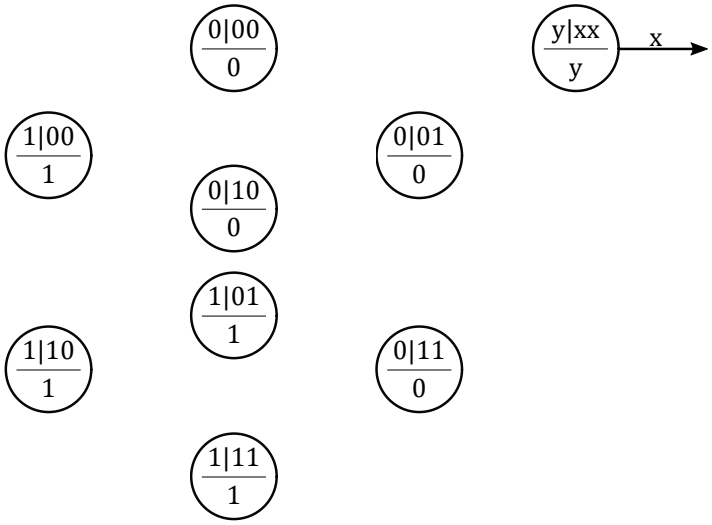


FIGURE 13 – Evénements à mémoriser

Pour chaque état ainsi défini, il faut analyser toutes les conditions d'entrées possibles et dessiner les flèches correspondantes. Ceci donne le graphe de la figure 14.

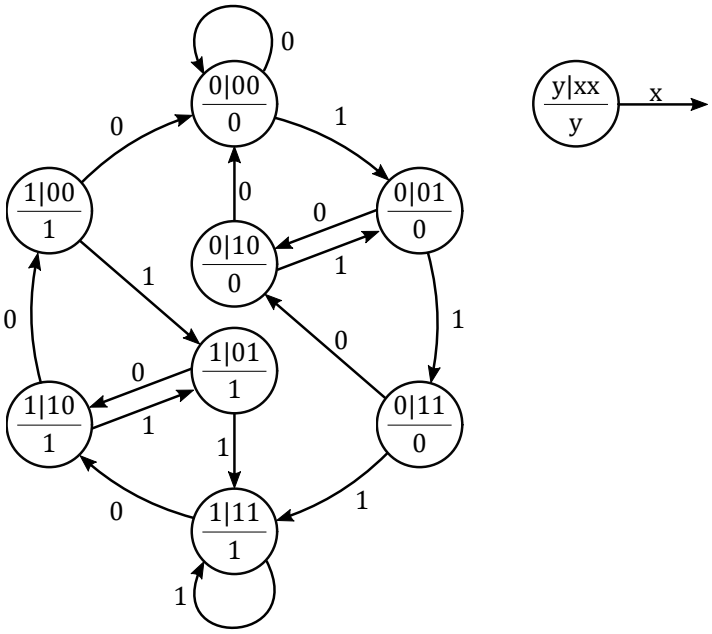
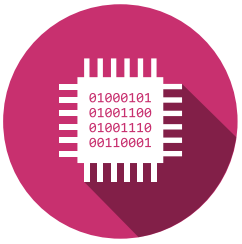
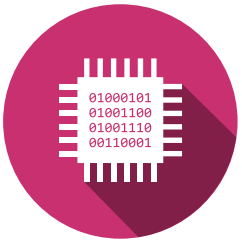


FIGURE 14 – Graphe des états à partir des événements à mémoriser



Les graphes développés à partir de différentes méthodes n'ont pas nécessairement le même nombre d'états, même si leur fonctionnement est identique.



6 Codage des états

Après réduction du graphe des états, il faut encore attribuer un code binaire aux différents états pour pouvoir remplir la table de vérité des blocs combinatoires et déterminer le schéma de la machine d'états.

6.1 Codage minimal

Le nombre minimal de bits, m, nécessaire pour coder les M états d'une machine est donné par

$$2^{m-1} < M < 2^m \tag{1}$$

Cette inéquation donne le nombre minimal de bascules nécessaire à la réalisation de la machine d'états.

L'assignement des codes peut se faire de manière aléatoire, en assignant le premier code binaire au premier état venu, le deuxième au suivant, etc.

Exemple : circuit anti-rebonds

La table 2 donne un exemple de codage des états pour le circuits anti-rebonds.

$state$	Q
low	000
$up1$	001
$up2$	010
$high$	111
$dn1$	110
$dn2$	011

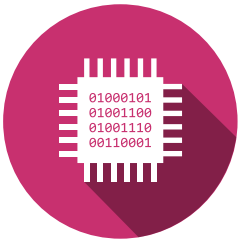
TABLE 2 – Codage binaire des états

Avec ce codage, on obtient les équations suivantes pour les blocs logiques combinatoires de la machine d'états :

$$\begin{cases} D_2 &= xQ_1 + Q_2Q_0 \\ D_1 &= xQ_1 + x\overline{Q_0} + Q_2 \\ D_0 &= xQ_1 + x\overline{Q_0} + Q_2\overline{Q_0} \\ Y &= Q_2 + Q_1Q_0 \end{cases} \tag{2}$$

6.2 Codage 1 parmi m

Une autre technique de codage consiste à utiliser M bits, et donc M bascules, pour coder les M états d'une machine. Le codage est fait de sorte que pour chaque état un seul des bits soit à '1'.



Ce codage est gourmand en bascules, mais les fonctions combinatoires qui y sont associées sont généralement très simples.
Surtout, il permet un décodage des états sans aléa. Ce point est d'une importance particulières pour des machines d'états qui régissent un séquencement d'opérations.
Dans un code 1 parmi m, la majorité des états internes possibles ne sont pas utilisés. La probabilité de voir apparaître des boucles indésirables en plus de la boucle principale du graphe est donc élevée. Il est donc important de prévoir une logique pour mettre la machine dans un des états de la boucle principale.

Exemple : circuit anti-rebonds

La table 3 donne un exemple de codage 1 parmi m pour les états du graphe de la figure 11.

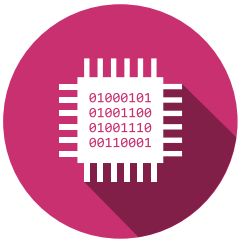
$state$	$Q_5 \dots Q_0$
low	000001
$up1$	000010
$up2$	000100
$high$	001000
$dn1$	010000
$dn2$	100000

TABLE 3 – Codage 1 parmi m

On peut alors directement déduire les équations du graphe des états. En effet, chaque état est associé à un bit d'état. La condition «l'état futur est $up1$ si l'état présent est low et l'entrée vaut '1'» s'écrit $D_1 = Q_0 x$. Les équations pour les blocs logiques combinatoires de la machine d'états sont donc donnés par

$$\begin{cases} D_0 &= \bar{x}Q_0 + \bar{x}Q_1 + \bar{x}Q_2 + \bar{x}Q_5 \\ D_1 &= xQ_0 \\ D_2 &= xQ_1 \\ D_3 &= xQ_2 + xQ_3 + xQ_4 + xQ_5 \\ D_4 &= \bar{x}Q_3 \\ D_5 &= \bar{x}Q_4 \\ Y &= Q_3 + Q_4 + Q_5 \end{cases} \tag{3}$$

Pour l'initialisation, on doit prévoir de mettre toutes les bascules à '0' sauf une qui est à mettre à '1'.



Références

[1] Suhail Almani. *Electronic Logic Systems*. second edition. New-Jersey : Prentice-Hall, 1989.

[2] Michael D. Ciletti et M. Morris Mano. *Digital Design*. second edition. New-Jersey : Prentice-Hall, 2007.

[3] David J. Comer. *Digital Logic and State Machine Design*. Saunders College Publishing, 1995.

[4] Randy H. Katz et Gaetano Borriello. *Contemporary Logic Design*. California : The Benjamin/Cummings Publishing Company Inc, 2005.

[5] Martin V. Künzli et Marcel Meli. *Vom Gatter Zu VHDL : Eine Einführung in Die Digitaltechnik*. vdf Hochschulverlag AG, 2007. isbn : 3 7281 2472 9.

[6] David Lewin et Douglas Protheroe. *Design of Logic Systems*. second edition. Hong Kong : Springer, 2013.

[7] Daniel Mange. *Analyse et synthèse des systèmes logiques*. Editions Géorgi. T. Traité d'électricité, volume V. St Saphorin : PPUR presses polytechniques, 1995. 362 p. isbn : 978-2-88074-045-0. Google Books : 5NSdD4GRl3cC.

[8] John F. Wakerly. *Digital Design : Principles And Practices*. 3rd edition. Prentice-Hall, 2008. isbn : 0-13-082599-9.

Acronymes

Ausgangszustand output state. 1

Eingangszustand input state. 1

interner Zustand internal state. 1

machine d'état Finite State Machine, FSM. 4, 5, 7, 9

Zustandsmaschine Finite State Machine, FSM. 1

état de sortie output state. 4, 5

état d'entrée input state. 4, 5

état interne internal state. 4, 5