Chapitre 9

Synthèse des systèmes séquentiels synchrones

Ce chapitre est consacré à la présentation de la méthode d'Huffman-Mealy pour la synthèse des systèmes séquentiels synchrones. Cette méthode permet de passer du cahier des charges décrivant un système au circuit logique correspondant. Elle est particulièrement utilisée pour la réalisation de contrôleurs.

9.1. Définitions générales

9.1.1. Circuits séquentiels synchrones et asynchrones

Les systèmes séquentiels peuvent être différenciés en fonction de leur mode de fonctionnement qui peut être synchrone ou asynchrone. Dans le mode synchrone, les éléments de mémorisation sont des bascules. Les modifications d'état du système ne peuvent donc intervenir qu'à des instants très précis déterminés par des signaux d'horloge. Par contre, dans le mode asynchrone, la fonction de mémorisation est réalisée par de simples boucles de rétroaction. L'évolution des états ne dépend donc que des modifications intervenant sur les entrées E_i de la machine.

Comme illustré sur la Figure 9.1, ces systèmes séquentiels peuvent donc être représentés par deux modèles différents.

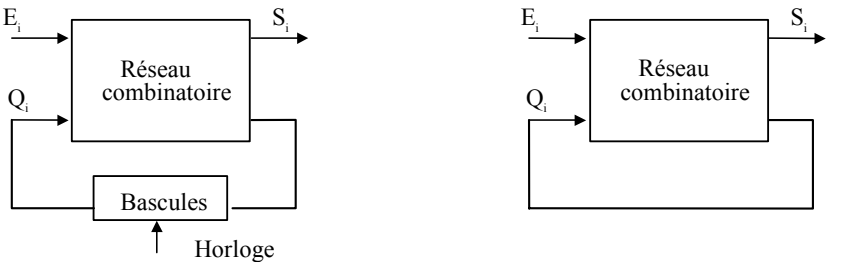

Figure 9.1. *Systèmes séquentiels synchrones et asynchrones*

Ce chapitre est consacré à l'étude des systèmes séquentiel synchrones. Les systèmes séquentiels asynchrones feront l'objet du chapitre suivant.

9.1.2. Modèle des systèmes séquentiels synchrones

Dans un système séquentiel, la présence d'un même vecteurs d'entrée n'entraîne pas nécessairement l'apparition d'une même combinaison sur les sorties. La combinaison de sortie dépend également de données internes caractérisant l'état du système.

L'état d'un système séquentiel est en fait une image des événements antérieurs s'étant produits sur les entrées du système. Cet état, qui évolue lorsque de nouveaux événements se produisent sur les entrées, est mémorisé dans des éléments internes spécifiques appelés éléments mémoires. Un circuit séquentiel contient r éléments de mémoire élémentaire q_1, q_2, ..., q_r, le vecteur $Q=(q_1, q_2, ..., q_r)$ caractérisant l'état interne du circuit séquentiel.

Les vecteurs E_i (Entrées), S_i (Sorties), Q_i (Etat) évoluent à des instants déterminés (phases) que l'on peut discrétiser en notant n l'instant présent et n+1 l'instant suivant. Le fonctionnement d'un système séquentiel peut alors être exprimé par des équations récurrentes et un état initial. Le modèle général (modèle de Mealy) de ces équations est le suivant :

$$Q_i(n+1) = F\ \{E_i(n)\ ,\ Q_i\ (n)\}$$
$$S_i\ (n) = G\ \{E_i(n)\ ,\ Q_i\ (n)\}$$

Un cas particulier concerne les systèmes dont la sortie ne dépend que de l'état interne (Modèle de Moore). Dans ce cas les équations s'expriment de la manière suivante :

$$Q_i\ (n+1) = F\ \{E_i\ (n)\ ,\ Q_i\ (n)\}$$
$$S_i\ (n) = G\ \{Q_i\ (n)\}$$

Ces équations peuvent être représentées de manière symbolique par le modèle présenté sur la Figure 9.2.

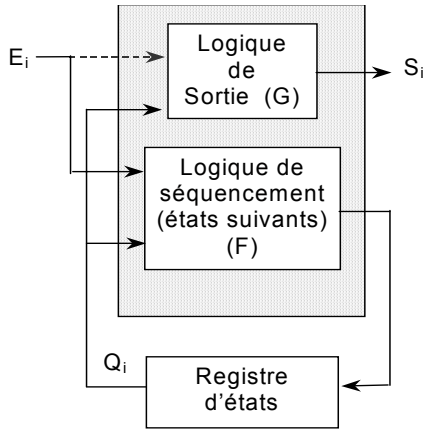

Figure 9.2. *Schéma général d'un système séquentiel synchrone*

Quel que soit le type de machine (Moore ou Mealy) l'état suivant du système $Q_i\ (n+1)$ dépend de l'état actuel $Q_i\ (n)$ et des entrées $E_i\ (n)$. La différence entre machine de Mealy et machine de Moore n'intervient que sur les sorties. Dans une machine de Mealy, les sorties $S_i\ (n)$ dépendent de l'état $Q_i\ (n)$ et des entrées $E_i\ (n)$. Par contre dans une machine de Moore, les sorties $S_i\ (n)$ ne dépendent que de l'état $Q_i\ (n)$.

Le registre d'état de ces machines synchrones est composé de bascules maître-esclaves synchronisée par le signal d'horloge. Ainsi, les modifications d'état du système c'est à dire des sorties de ces bascules ne peuvent donc intervenir qu'à des instants très précis déterminés par le signal d'horloge. Divers types de bascules peuvent être utilisés pour réaliser ce registre d'état (D, T, JK). Dans la pratique, se seront essentiellement des bascule D.

9.2. Synthèse de séquenceurs

Avant de développer la méthode générale de synthèse des systèmes séquentiels synchrone, nous nous proposons d'en aborder le principe de manière simplifiée en considérant le cas des séquenceurs.

Un séquenceur est un système séquentiel délivrant une séquence de combinaisons prédéterminées. Un tel système un en fait cas particuliers de systèmes séquentiels synchrone dans la mesure dans la mesure ou les sorties peuvent être directement portées par les sorties des bascules. Un compteur est par exemple un séquenceur particulier, mais autant la synthèse d'un compteur peut s'envisager de manière intuitive du fait des caractéristiques qui émanent de l'évolution des états (au moins dans le cas d'un compteur en binaire naturel), autant une méthode plus systématique est nécessaire lorsque la séquence à générer ne dispose pas de propriétés particulières.

A titre d'exemple nous considérerons un séquenceur délivrant une séquence de 8 états successifs correspondant à ceux d'un compteurs binaire par 8. Supposons également que ce séquenceur dispose d'une entrée C permettant à tout moment de revenir de manière synchrone dans l'état initial (000). Ce système peut se mettre sous la forme générale d'un système séquentiel synchrone pour lequel les sorties sont directement les sorties S_i sont directement les variables d'état Q_i (sorties des bascules) (Figure 9.3).

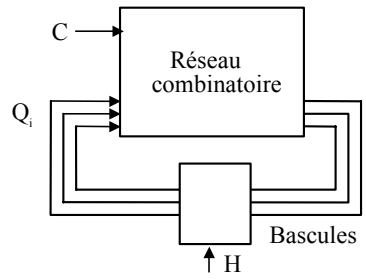

Figure 9.3. *Schéma général d'un séquenceur*

D'autre part, la sortie ne dépendant pas directement de l'entrée C, il s'agit d'une machine de Moore.

La fonction réalisée par tout système séquentiel synchrone peut être représentée par un graphe appelé graphe d'état. Le graphe du compteur est représenté sur la Figure 9.4.

Chaque noeud du graphe représente un état du compteur et chaque arc indique l'état suivant que prendra le compteur après un coup d'horloge. Cet état suivant est fonction de l'entrée C.

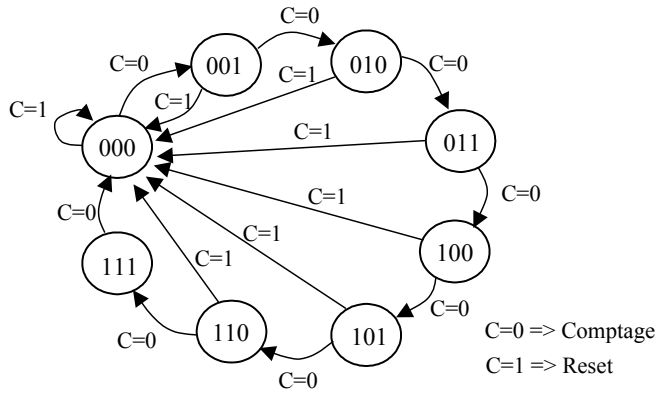

Figure 9.4. *Graphe de d'état*

Ce graphe peut également s'exprimer sous une forme tabulaire. Cette représentation, appelée table d'état (Figure 9.5).

Etats	Etats Suivants	
	C = 0	C = 1
000	001	000
001	010	000
010	011	000
011	100	000
100	101	000
101	110	000
110	111	000
111	000	000

Figure 9.5. *Table d'état*

Supposons maintenant que pour réaliser ce système nous prenions des bascules D. La table de transition d'une telle bascule est la suivante (Figure 9.6) :

D(n)	Q(n+1)
0	0
1	1

Figure 9.6. *Tables de transition des bascules D*

Pour chaque bascule i nous connaissons l'état suivant $Q_i(n+1)$ (après le coup d'horloge) en fonction de l'état présent $Q_i(n)$ et de l'entrée C. Il reste donc à déterminer les entrées D_i de chaque bascule à partir des tables de transition correspondantes (Figures 9.7).

Etats $Q_2Q_1Q_0(n)$	Etats Suivants $Q_2Q_1Q_0(n+1)$		C = 0			C = 1		
	C = 0	C = 1	$D_2(n)$	$D_1(n)$	$D_0(n)$	$D_2(n)$ $D_0(n)$		$D_1(n)$
000	001	000	0	0	1	0	0	0
001	010	000	0	1	0	0	0	0
010	011	000	0	1	1	0	0	0
011	100	000	1	0	0	0	0	0
100	101	000	1	0	1	0	0	0
101	110	000	1	1	0	0	0	0
110	111	000	1	1	1	0	0	0
111	000	000	0	0	0	0	0	0

Figure 9.7. *Entrées des bascules*

Remarque : Avec les bascules D, on a D(n) = Q(n+1). Avec de telle bascules, on peut ainsi s'affranchir de l'écriture des colonne donnant les $D_i(n)$ sachant qu'elles sont identiques à celle donnant les états suivants $Q_i(n+1)$.

Les expressions des entrées de bascules $D_i(n)$ peuvent maintenant être déterminées (Figure 9.8).

Q_0C

Q_2Q_1	00	01	11	10
00				
01				1
11	1			
10	1			1

$D_2 = C'.(Q_2' \oplus Q_1.Q_0)$

Q_0C

Q_2Q_1	00	01	11	10
00				1
01	1			
11	1			
10				1

$D_1 = C'.(Q_1 \oplus Q_0)$

Q_0C

Q_2Q_1	00	01	11	10
00	1			
01	1			
11	1			
10	1			

$D_0 = C'.Q_0'$

Figure 9.8. *Synthèse des entrées de bascules*

Ayant les équations des entrées des bascules le circuit peut être réalisé (Figure 9.9).

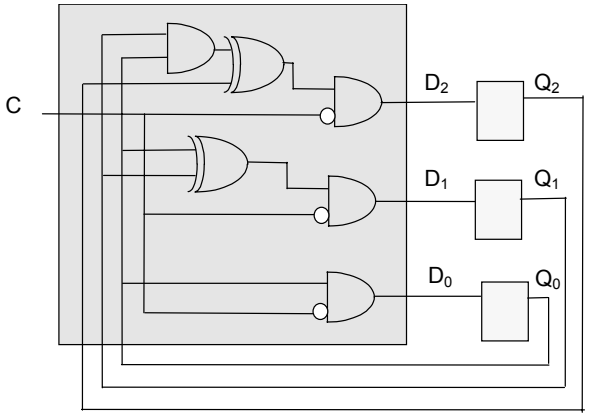

Figure 9.9. *Schéma logique*

9.3. Méthode de synthèse d'Huffman-Mealy

Dans cette partie, nous généraliserons les concepts évoqués précédemment afin de permettre le passage d'un cahier des charges quelconque au circuit correspondant. De plus nous établirons des règles de minimisation permettant d'optimiser le nombre de bascules utilisées pour la réalisation du circuit.

La méthode proposée, connue sous le nom de méthode d'Huffman-Mealy se décompose en plusieurs étapes. Ces étapes sont les suivantes :

1 : Modélisation du cahier des charges

- Graphe d'état
- Table d'état

2 : Minimisation du nombre d'états

- Règles de minimisation
- Détermination du nombre de bascules minimum

3 : Codage

- Codage des états
- Codage des entrées de bascules

4 : Synthèse

- Synthèse des entrées de bascules et des sorties de la machine
- Implantation technologique (mapping)

9.3.1. Modélisation du cahier des charges

Le cahier des charges d'un système est généralement donné en langage courant.

Exemple : Le système considéré a une entrée (E) et une sortie (S). Il reçoit sur son entrée des bits arrivant en série. La sortie (S) doit passer à 1 chaque fois qu'une séquence 010 apparaît sur l'entrée (E) puis repasser à 0 sur le bit suivant quel que soit sa valeur.

Pour faire la synthèse d'un tel cahier des charges, la première étape est de le modéliser.

9.3.1.a. Graphe d'état

Le modèle généralement utilisé pour représenter le cahier des charges d'un système est un graphe appelé graphe d'état ou graphe de fluence. Les nœuds de ce graphe représentent les états, un nom symbolique étant affecté à chacun des états. Les arcs du graphe sont orientés. Ils représentent les possibilités de passage entre états. Ces changements d'états se font sur un front d'horloge en fonction des valeurs d'entrée. La structure générale du graphe représentant l'évolution des états d'une machine ayant une entrée E est représentée sur la Figure 9.10.

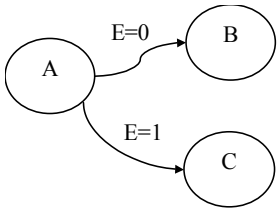

Figure 9.10. *Structure générale du graphe d'état d'un système séquentiel synchrone*

Les caractéristiques de ces graphes sont les suivantes :

- Chaque état (Q_i) est représenté par un cercle,
- A chaque état est associé un nom symbolique
- Le passage d'un état à un autre se fait au coup d'horloge,
- L'état atteint dépend de l'état de départ et de la valeur des d'entrées (E_i),
- De chaque état part au plus 2^n arcs, n étant le nombre d'entrées (E_i),
- Ce graphe est connexe.

Si l'on considère maintenant la sortie, il faut différencier les deux types de machines que sont machines de Moore et machines de Mealy. En effet, dans une machine de Moore, les sorties ne dépendent que des états et par conséquent peuvent être consignées à l'intérieur des cercles. Dans une machine de Mealy, les sorties dépendent des états mais également des entrées. Ces sorties doivent donc être consignées sur les arcs du graphe. Ainsi, selon le type de machine, un modèle différent de graphe d'état doit être considéré (Figure 9.11).

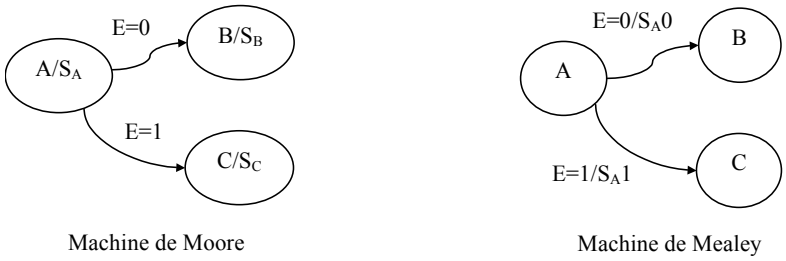

Figure 9.11. *Structure des graphes d'état de Moore et de Mealy*

Exemple : Selon que le système sera réalisé sous forme de machine de Moore ou sous forme de Machine de Mealy, le graphe d'état représentant le cahier des charges précédent est représenté sur la Figure 9.12.

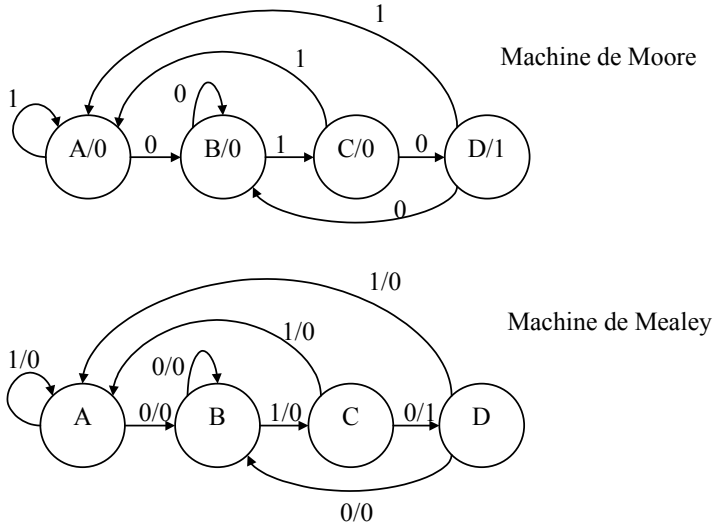

Figure 9.12. *Graphes d'état de la machine détectant les séquences 010*

Remarque : La structure des deux graphes paraît identique. En fait, il est toujours possible de passer d'un graphe de Moore à un graphe de Mealy. Pour cela il, suffit de reporter les sorties associées à chaque état (Moore) sur les arcs arrivant à chacun de ces états. Nous verrons par la suite que l'inverse, c'est à dire passer d'un graphe de Mealy à un graphe de Moore n'est pas toujours possible.

Ce modèle permet de représenter le cahier des charges sous une forme exploitable, mais permet également de figer le fonctionnement du circuit dans tous les cas particuliers non prévus dans le cahier des charges initial. Dans le cas de cet exemple, le graphe correspond à une machine qui détecte les séquences 010 en mots disjoints (Arc 1 de D à A) et non en mots imbriqués.

A ce niveau de la synthèse, ce qui est important c'est de s'assurer que le graphe correspond bien au cahier des charges. Le nombre d'état utilisés pour représenter ce cahier des charges importe peu. Il sera minimisé par la suite.

9.3.1.b. Table d'état

Le cahier des charges d'un système peut également être modélisé sous une forme tabulaire qui est plus facile à manipuler qu'une représentation sous forme de graphe. Cette représentation tabulaire, appelée table d'état, est directement déductible du graphe d'état. Elle représente les différentes possibilités d'états suivants de chacun des états du système et ceci en fonction des entrées. Les sorties associées à chaque état sont également représentées

sur cette table. Elles dépendent ou non des entrées selon qu'il s'agit d'une machine de Mealy ou d'une machine de Moore.

Exemple : Les tables d'état correspondant au graphe de la figure 1.12 sont présentées sur la Figure 9.13.

Etats	Etats Suivants		Sortie
	E=0	E=1	
A	B	A	0
B	B	C	0
C	D	A	0
D	B	A	1

Machine de Moore

Etats	Etats Suivants		Sortie	
	E=0	E=1	E=0	E=1
A	B	A	0	0
B	B	C	0	0
C	D	A	1	0
D	B	A	0	0

Machine de Mealy

Figure 9.13. *Tables d'état de la machine détectant les séquences 010*

9.3.2. Minimisation du nombre d'états

Le nombre d'états de la machine influe directement sur le nombre de bascules nécessaires pour réaliser ce système. Or, le nombre d'états utilisés pour représenter le cahier des charges, que ce soit sur le graphe d'état ou sur la table d'état, n'est pas nécessairement minimum.

9.3.2.a. Règles de minimisation

Deux règles permettent de déterminer les états équivalents et par conséquent de minimiser le nombre d'états nécessaires à la réalisation du circuit.

- Règle R1 : Deux états sont équivalents si pour chaque combinaison d'entrée, ils ont mêmes sorties et mêmes états suivants.
- Règle R2 : Les états sont regroupés en différentes classes selon les valeurs de sorties associées. Ainsi, deux états ayant mêmes sorties (pour chaque combinaison d'entrée) sont dans la même classe. Les états appartenant à une même classe sont équivalents s'il ne peuvent être séparés. Or les états appartenant à une même classe doivent être séparés si les états suivants associés à chacun d'eux sont dans des classes différentes.

Lorsque plusieurs états sont équivalents, il suffit de garder qu'un seul représentant par classe d'équivalence et de renommer les états suivants en conséquence.

Exemple 1: Les règles de minimisation appliquées à la table d'état de la machine de Mealy précédente donnent :

R1 : A et D on mêmes sorties et mêmes états suivants, ils sont donc équivalents. L'état D peut par exemple être éliminé. En renommant les états suivants en conséquence, c'est à dire en remplaçant D par A, la table d'état devient :

Etats	Etats Suivants		Sortie	
	E=0	E=1	E=0	E=1
A	B	A	0	0
B	B	C	0	0
C	A	A	1	0

Figure 9.14. *Table d'état réduite*

Au sens de la règle R1 il n'y a pas d'autres états équivalents.

R2 : les états peuvent être regroupés en deux classes (classe 1 et classe 2).

(1) (2)	Classes
(A , B) (C)	Etats
BA BC	Etats suivants
11 12	Classes des états suivants

Les états A et B doivent être séparés. Il y a maintenant qu'un seul état par classe. Il n'y a donc plus d'états équivalents. Cette machine peut être réalisée avec 3 états.

Remarque : s'il est toujours possible de passer du graphe représentant une machine de Moore à un graphe représentant la même machine en Mealy, l'exemple précédent montre que l'inverse n'est pas toujours possible. En effet, une machine de Mealy peut comporter moins d'état qu'une machine de Moore.

Le nombre d'états nécessaire à la réalisation d'une machine de Mealy pouvant être inférieur à celui nécessaire à la réalisation d'une machine de Moore, le nombre de bascules peut l'être également. D'où l'avantage qu'il peut y avoir à réaliser une machine de Mealy plutôt qu'une machine de Moore. Ceci dit, les machines de Mealy peuvent avoir des inconvénients liés au fait que les sorties dépendent directement des entrées. En effet, lors du passage d'un état à un autre, les entrées ne doivent pas varier. Il se produit donc un instant entre le changement d'état et le changement d'entrée ou le système se trouve dans le nouvel état mais en présence de l'entrée ayant conduit à cet état, c'est à dire de l'entrée précédente. Puisqu'en machine Mealy, les sorties dépendent directement de l'état et des entrées, elles peuvent donc être soumise à des commutations parasites.

9.3.2.b. Détermination du nombre de bascules minimum

Le nombre minimum d'états "q" étant déterminé, on peut en déduire le nombre minimum "n" de variables d'état et par conséquent de bascules nécessaires au codage de ces états à partir de la double inéquation suivante :

$$2^{n-1} < q < 2^n$$

Exemple : Pour la machine de Mealy précédente, le nombre minimum d'état étant de 3, le nombre de variables d'état nécessaire au codage de ces états est 2. Deux bascules sont donc nécessaires pour réaliser ces systèmes.

9.3.3. Codage

9.3.3.a. Codage des états

Chaque état peut être codé par une combinaison de ces n variables d'état. Chaque état doit avoir un code différent des autres mais le codage des états peut être quelconque. Notons toutefois que le codage influence la structure de la future machine et peut donc influencer sa complexité. Le problème de l'optimisation de la

machine résultante passe donc par un choix judicieux du codage des états. Ce problème ne sera pas développé ici.

Une fois les états codés, la table d'état peut être exprimée en fonction de ces codes.

Exemple : Nous appellerons les variables d'état (sorties des 2 bascules) de la machine détectant la séquence 010, Q_1 et Q_2. En considérant un codage donné (celui indiqué dans la colonne "Etats"), la table d'état codée de la machine de Mealy correspondant à la table d'état de la Figure 9.14 est représentée sur la Figure 9.15.

Etats $Q_1Q_2(n)$	Etats Suivants $Q_1Q_2(n+1)$		Sortie	
	E=0	E=1	E=0	E=1
00 (A)	01	00	0	0
01 (B)	01	11	0	0
11 (C)	00	00	1	0

Figure 9.15. *Tables d'état codées de la machine de Mealy*

9.3.3.b. Codage des entrées de bascules

Le code des états étant déterminé, les entrées de bascules peuvent l'être. Pour chaque bascule i nous connaissons l'état suivant $Q_i(n+1)$ (après le coup d'horloge) en fonction de l'état présent $Q_i(n)$ et des entrées. Pour réaliser ce système il reste à déterminer les entrées de chaque bascule.

Avec des bascules D, les entrées D_i peuvent être déterminée directement à partir de la relation :

$$D_i(n) = Q_i(n+1)$$

Exemple : Reprenons la machine de Mealy précédente. Les entrées D_1 et D_2 des bascules Q_1 et Q_2 sont représentées sur la Figure 9.16.

Etats $Q_1Q_2(n)$	Etats Suivants $Q_1Q_2(n+1)$		Sortie		Entrées Bascules $D_1D_2(n)$	
	E=0	E=1	E=0	E=1	E=0	E=1
00 (A)	01	00	0	0	01	00
01 (B)	01	11	0	0	01	11
11 (C)	00	00	1	0	00	00

Figure 9.16. *Détermination des entrées de bascules*

9.3.4. Synthèse

9.3.4.a. Synthèse des entrées de bascules et des sorties de la machine

Sur la table précédente on dispose des sortie et entrées de bascules exprimées en fonction des entrées et des variables d'état (sorties des bascules). Il suffit donc maintenant d'exprimer les fonction logiques relatives aux sorties et entrées de bascules.

Exemple : Reprenons la machine de Mealy précédente et déterminons les équations de la sortie S et des entrée de bascules D_1 et D_2 (Figure 9.17)

Q_1Q_2 \ E	0	1
00	0	0
01	0	1
11	0	0
10	ϕ	ϕ

$D_1 = E.Q_1' + E'.Q_2$

Q_1Q_2 \ E	0	1
00	1	0
01	1	1
11	0	0
10	ϕ	ϕ

$D_2 = E'.Q_2' + Q_1'Q_2$

Q_1Q_2 \ E	0	1
00	0	0
01	0	0
11	1	0
10	ϕ	ϕ

$S = E'.Q_1$

Figure 9.17. *Synthèse de D1 et D2*

9.3.4.b. Implantation technologique

L'implantation technologique de fonctions logiques est un problème en soit qui sera traité dans un chapitre spécifique. Nous pouvons nous contenter ici, d'une implantation à portes obtenue directement à partir de ces équations déterminée précédemment (Figure 9.18).

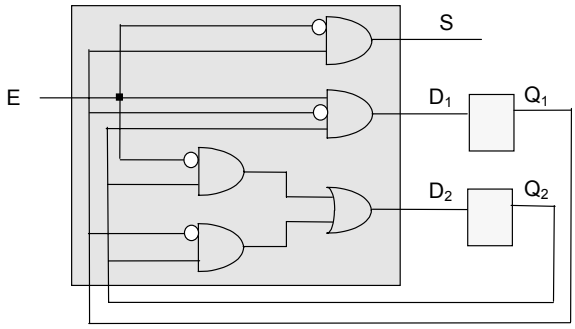

Figure 9.18. *Schéma du détecteur de séquence 010*

9.4. Traitement de « mots binaires »

Lorsque les bits d'entrée correspondent à des mots binaires, la synthèse de la machine analysant ces mots peut être réalisée par le processus décrit précédemment mais également par un processus spécifique ou les temps d'arrivée des bits sur la machine sont exprimés de manière explicite.

9.4.1. Graphes en arbre

Lorsque la machine à réaliser doit traiter des mots binaire arrivant en série sur ces entrées, le graphe d'état représentant la machine peut être réalisé sous forme d'arbre, sachant qu'à la fin de l'analyse d'un (ou plusieurs) mot(s) il faut revenir à l'état initial pour traiter le(s) suivant(s).

Exemple : On désire réaliser une machine recevant sur sont entrée E des mots de 4 bits en série (poids faible en tête). Sa sortie S doit passer à 1 chaque fois qu'un mot représente un nombre supérieur à 9 et revenir à 0 sur le premier bit du mot suivant. Ce cahier des charges peut se modéliser par le graphe en arbre représenté sur la figure 9.19.

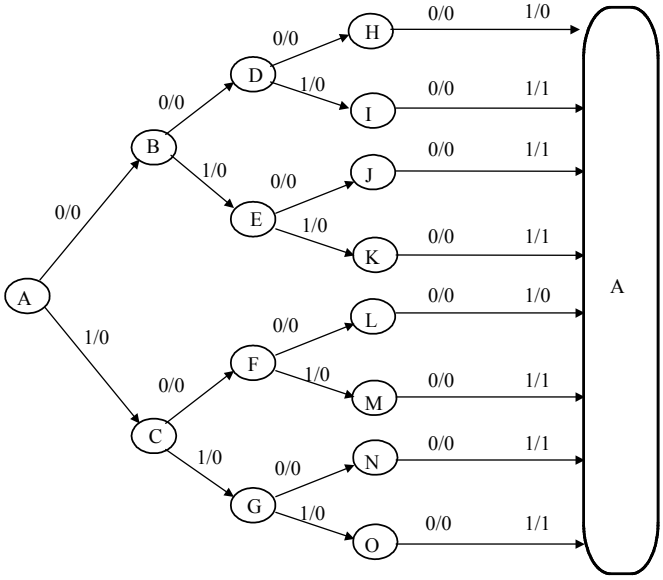

Figure 9.19. *Graphe d'état de la machine détectant les nombre supérieurs à 9*

Ce graphe n'est évidemment pas optimum d'un point de vue nombre d'état, mais à ce stade de la synthèse, ce n'est absolument pas le but. L'essentiel est que le cahier des charges soit fidèlement représenté.

La table d'état correspondant à ce graphe est la suivante (figure 9.20) :

Etats	Etats Suivants		Sortie	
	0	1	0	1
A	B	C	0	0
B	D	E	0	0
C	F	G	0	0
D	H	I	0	0
E	J	K	0	0
F	L	M	0	0
G	N	O	0	0
H	A	A	0	0
I	A	A	0	1
J	A	A	0	1
K	A	A	0	1
L	A	A	0	0
M	A	A	0	1
N	A	A	0	1
O	A	A	0	1

Figure 9.20. *Table d'état de la machine détectant les nombre supérieurs à 9*

- En appliquant la première règles de minimisation, on trouve les équivalences suivantes (I,J,K,M,N,O) (H,L). On conserve H et I et on élimine J, K, M, N, O, L (Figure 9.21.a).
- En ré-appliquant la première règle de minimisation, on trouve les équivalences suivantes (D,F) (E,G). On conserve D et E et on élimine F et G (Figure 9.21.b).
- En ré-appliquant la première règles de minimisation, on trouve les équivalences suivantes (B,C). On conservant B et on élimine C (Figure 9.21.c).

Etats	Etats Suivants		Sortie	
	0	1	0	1
A	B	C	0	0
B	D	E	0	0
C	F	G	0	0
D	H	I	0	0
E	I	I	0	0
F	H	I	0	0
G	I	I	0	0
H	A	A	0	0
I	A	A	0	1

(a)

Etats	Etats Suivants		Sortie	
	0	1	0	1
A	B	C	0	0
B	D	E	0	0
C	D	E	0	0
D	H	I	0	0
E	I	I	0	0
H	A	A	0	0
I	A	A	0	1

(b)

Etats	Etats Suivants		Sortie	
	0	1	0	1
A	B	B	0	0
B	D	E	0	0
D	H	I	0	0
E	I	I	0	0
H	A	A	0	0
I	A	A	0	1

(c)

Figure 9.21. *Réduction de la table d'état*

A ce stade, il n'y a plus de minimisation possible avec la règle R1. On peut également vérifier que la règle R2 n'amène pas de réduction supplémentaires. Le nombre d'états minimum permettant de réaliser cette machine étant de 6, le nombre de bascules nécessaires est donc 3. Nous ne développerons pas ici la suite de la synthèse de cette machine sous cette forme.

Remarque : Dans le cas ou l'on a à analyser des mots binaires, le cahier des charges peut toujours être modélisé par un graphe en arbre. Un tel graphe n'est évidemment pas optimum d'un point de vue nombre d'état, mais une telle approche facilite grandement l'interprétation du cahier des charge sachant qu'à ce stade de la synthèse, l'essentiel est que le cahier des charges soit fidèlement représenté. Réaliser directement un graphe réduit (voir optimum) est toujours possible mais peut conduire à une interprétation plus « risquée » du cahier des charges. Le graphe optimum peut d'ailleurs toujours être obtenu à l'issu du processus de minimisation du nombre d'états. A titre d'exemple et de comparaison avec le graphe en arbre, le graphe optimum décrivant le comportement de la machine précédente est présenté sur la figure 9.22. Ce graphe est issu de la table présentée sur la figure 9.21.c.

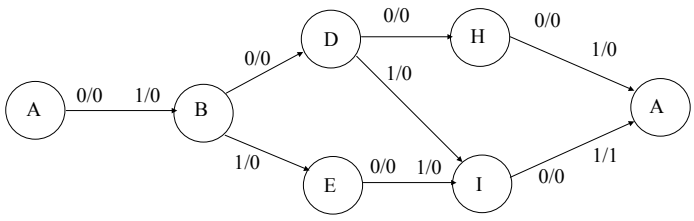

Figure 9.22. *Graphe d'état optimum de la machine détectant les nombre supérieurs à 9*

9.4.2. Machines à « temps explicite »

Supposons maintenant qu'il existe dans le circuit un dispositif (compteur) synchronisé sur la même horloge fournisse une information sur le bit présent en entrée de la machine à un instant donné (Figure 9.23).

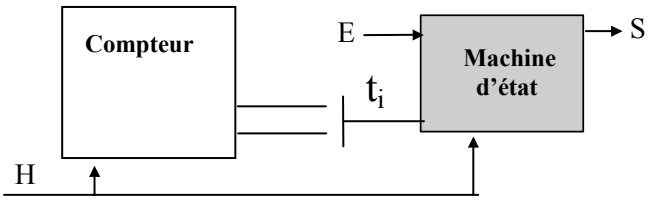

Figure 9.23. *Machine à temps explicite*

En reprenant l'exemple précédent, supposons que les quatre sorties du compteur (t1,t2,t3,t4) soient telles que :

t1 t2 t3 t4 = 1000 lorsque le premier bit est présent sur les entrées de la machine,
t1 t2 t3 t4 = 0100 lorsque le deuxième bit est présent sur les entrées de la machine,
t1 t2 t3 t4 = 0010 lorsque le troisième bit est présent sur les entrées de la machine,
t1 t2 t3 t4 = 0001 lorsque le quatrième bit est présent sur les entrées de la machine.

Ces informations complémentaires peuvent être utilisées comme entrées de la machine d'état, machine comporte maintenant, dans le cas de notre exemple, non plus une seule entrée mais 5 qui sont : E, t1, t2, t3, t4. Notons par ailleurs qu'uniquement 4 combinaisons de t1, t2, t3, t4 sont possibles.

Le graphe de la machine peut rester identique à celui réalisé précédemment. Il suffit simplement d'indiquer les valeurs de t1 t2 t3 t4.

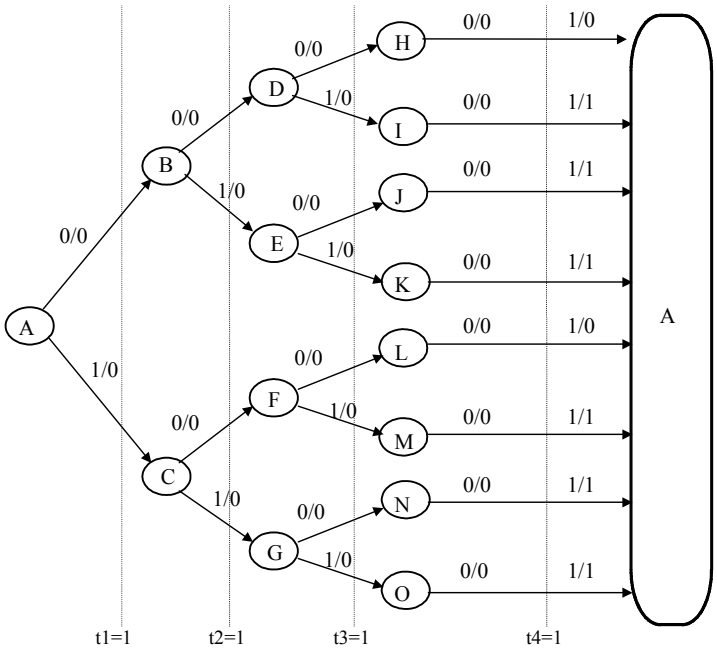

Figure 9.24. *Graphe d'état de la machine à temps explicite.*

La table d'état peut maintenant s'exprimer de la manière suivante (Figure 9.25).

Etats	Etats Suivants								Sortie			
	t1=1		t2=1 t4=1		t3=1				t1	t2	t3	t4
	0	1	0	1	0	1	0	1	01	01	01	01
A	B	C	-	-	-	-	-	-	00	--	--	--
B	-	-	D	E	-	-	-	-	--	00	--	--
C	-	-	F	G	-	-	-	-	--	00	--	--
D	-	-	-	-	H	I	-	-	--	--	00	--
E	-	-	-	-	J	K	-	-	--	--	00	--
F	-	-	-	-	L	M	-	-	--	--	00	--
G	-	-	-	-	N	O	-	-	--	--	00	--
H	-	-	-	-	-	-	A	A	--	--	--	00
I	-	-	-	-	-	-	A	A	--	--	--	01
J	-	-	-	-	-	-	A	A	--	--	--	01
K	-	-	-	-	-	-	A	A	--	--	--	01
L	-	-	-	-	-	-	A	A	--	--	--	00
M	-	-	-	-	-	-	A	A	--	--	--	01
N	-	-	-	-	-	-	A	A	--	--	--	01
O	-	-	-	-	-	-	A	A	--	--	--	01

Figure 9.25. *Table d'état de la machine à temps explicite.*

En appliquant la première règles de minimisation sans affecter les états indéterminés nous revenons à 6 états comme précédemment (Figure 9.26).

Etats	Etats Suivants								Sortie			
	t1=1		t2=1		t3=1		t4=1		t1	t2	t3	t4
	0	1	0	1	0	1	0	1	01	01	01	01
A	B	B	-	-	-	-	-	-	00	--	--	--
B	-	-	D	E	-	-	-	-	--	00	--	--
D	-	-	-	-	H	I	-	-	--	--	00	--
E	-	-	-	-	I	I	-	-	--	--	00	--
H	-	-	-	-	-	-	A	A	--	--	--	00
I	-	-	-	-	-	-	A	A	--	--	--	01

Figure 9.26. *Table d'état réduite de la machine à temps explicite.*

En ré-appliquant toujours la première règle de minimisation, on peut maintenant trouver d'autres équivalences en affectant certains états indéterminés. Ainsi on peut arriver à deux classes d'équivalence telles que par exemple (A,D,H) et (B,E,I). En conservant A et B et en éliminant D, H, E et I, la table devient celle présentée sur la Figure 9.27.

Etats	Etats Suivants								Sortie			
	t1=1		t2=1		t3=1		t4=1		t1	t2	t3	t4
	0	1	0	1	0	1	0	1	01	01	01	01
A	B	B	-	-	A	B	A	A	00	--	00	00
B	-	-	A	B	B	B	A	A	--	00	00	01

Figure 9.27. *Table d'état minimale de la machine à temps explicite.*

Une seule bascule est donc nécessaire pour réaliser cette machine. En supposant une bascule D et en codant l'état a 0 et l'état b 1 sur la sortie Q de cette bascule on obtient la table 9.28.

Etats	Etats Suivants								Sortie			
	t1=1		t2=1		t3=1		t4=1		t1	t2	t3	t4
	0	1	0	1	0	1	0	1	01	01	01	01
0	1	1	-	-	0	1	0	0	00	--	00	00
1	-	-	0	1	1	1	0	0	--	00	00	01

Figure 9.28. *Table d'état codée de la machine à temps explicite.*

Ce qui donne : $S = t4.e.Q$

$D = t1 + t2.e + t3.(e + Q)$

Bilan : Si l'on fait le bilan global du nombre de bascules, on s'aperçoit que globalement le nombre de bascules nécessaires à la réalisation de ce dispositif est identique à celui déterminé précédemment (3 bascules). En effet, il faut ajouter à la bascule de la machine ainsi réalisée, les deux bascules nécessaires à la réalisation du compteur. Concevoir une telle machine peut toutefois être intéressant dans les deux cas suivants:

1. Le dispositif (compteur) donnant les informations sur le bit présent en entrée de la machine existe déjà dans le circuit,
2. Plusieurs machines manipulant les mêmes données doivent être conçues sur le même circuit (auquel cas, le compteur n'est réalisé qu'une seule fois). Dans ce dernier cas, si n est le nombre de machines et m le nombre de bascules du compteur, le gain par rapport à une approche classique est de (n-1)*m.

9.5. Implantations partitionnées

Le premier réflexe à avoir, face à un problème, un tant soit peu complexe à résoudre, est de le scinder le problème en deux. La démarche précédente peut être répétée, pour chaque demi-problème, jusqu'à obtenir des sous-ensembles dont la réalisation tient en quelques circuits élémentaires, en quelques lignes de code source dans un langage ou dans un diagramme de transitions qui ne dépasse pas une dizaine d'états différents. L'architecture résultante sera ainsi constituée de plusieurs machines communiquant entre elles comme illustré sur la figure 9.29.

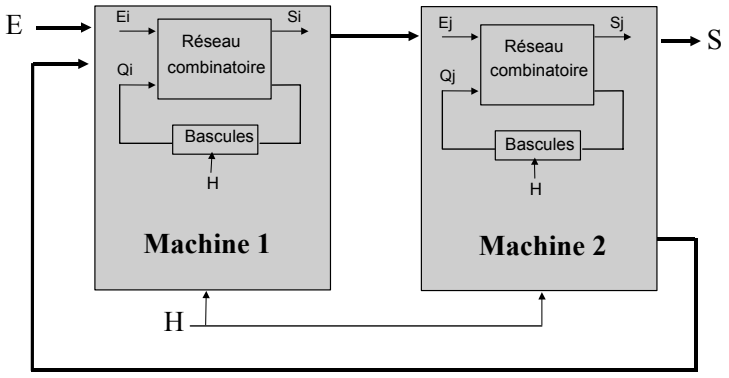

Figure 9.29. *Machine d'état partitionnée*

Exemple : On désire concevoir un séquenceur produisant la séquence suivante : 0, 1, 2, 3, …, n-1, n, n-1, n-2, …, 3, 2, 1, 0, 1, 2, 3, …n. Un tel système peut être synthétisé à partir d'un graphe comportant 2*n états. Mais la synthèse peut également être envisagée en partitionnant la machine en 2 machines d'état dont un compteur / décompteur commandé par un signal Up/Dn (Figure 9.30).

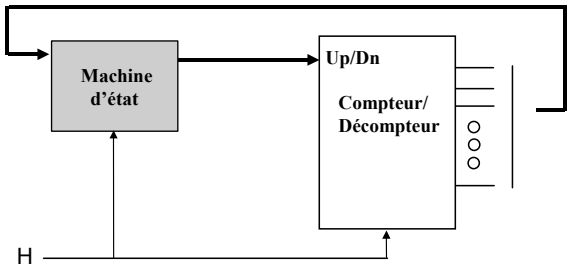

Figure 9.30. *Machine d'état partitionnée*

La synthèse se limite alors à la synthèse de la machine d'état commandant l'entrée Up/Dn du compteur / décompteur. Le graphe représentant le fonctionnement de cette machine ainsi que la structure de la machine obtenue après synthèse de ce graphe est donné sur la figure 9.31.

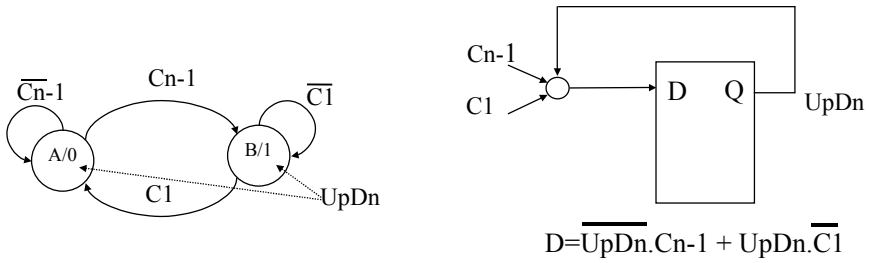

Figure 9.31. *Machine de contrôle du signal Up/Dn*

Remarque : Outre l'aspect complexité, de telle implantations peuvent avoir un intérêt certain lorsqu'on cherche à minimiser les chemins critiques pour augmenter la vitesse de fonctionnement globale du système. Cet aspect ne sera pas développé ici.