Chapitre 10

Synthèse des systèmes séquentiels asynchrones

Dans beaucoup de situation pratiques, les entrées du système à réaliser ne sont pas gérées par une horloge et sont donc sujettes à modification à des instants quelconques. Dans ce cas il n'est pas possible de concevoir un système synchrone. Dans ce chapitre nous nous intéresserons donc à la synthèse des systèmes séquentiels asynchrones.

10.1. Structure des systèmes séquentiels asynchrones

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 primaires (E_i) de la machine. La représentation la plus générale d'un système asynchrone est donné sur la Figure 10.1.

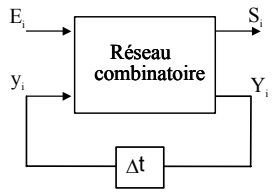


Figure 10.1. *Modèle général d'un systèmes séquentiel asynchrone*

Remarque : L'élément retard Δt est un élément virtuel permettant de modéliser les retards des signaux dans la logique combinatoire.

Le circuit est dit dans un état stable si et seulement si $y_i=Y_i$ pour tout i= 1, 2, ..., k. En réponse à un changement d'entrée, la logique combinatoire produit un nouvel ensemble de valeurs sur Y_i. Si ces valeurs sont différentes de celles de y_i ce circuit entre dans un état instable. On obtiendra un nouvel état stable lorsqu'à nouveau on aura $y_i=Y_i$. Si aucun un état stable n'est atteint le système oscille.

Une variation d'entrée est obligatoire pour que le système évolue d'un état stable vers un autre état stable. D'autre part, vu les délais inhérents aux structures, il est impossible de garantir le changement de deux variables simultanément. La synthèse d'un système asynchrone doit interdire de telles situations. Cette restriction signifie en effet qu'une seule entrée peut changer à un instant donné, et que le temps écoulé entre deux variations est plus grand que le délai interne de la structure. Lorsqu'un changement apparaît sur une entrée, nous ferons donc l'hypothèse qu'aucune variation sur toutes entrées ne peut intervenir avant que le circuit soit dans un état stable.

Un tel mode de fonctionnement est appelé mode fondamental. Notons qu'un mode de fonctionnement différent appelé mode pulsé est aussi possible mais ne sera pas étudié ici.

10.2. Méthode de synthèse

La méthode de synthèse de systèmes séquentiels asynchrones proposée a pour but de minimiser le nombre de variables secondaires (Y_i). Ceci a pour effet direct de minimiser le nombre de fonctions du réseau combinatoire et par conséquent le nombre de portes du circuit. Cette méthode se décompose en plusieurs étapes. Ces étapes sont les suivantes:

1 : Modélisation du cahier des charges

- Graphe d'état
- Table d'état (ou des phases) primitive

2: Minimisation du nombre d'états

- Elimination des états équivalents
- Fusionnement d'états

3: Codage des états

- Graphe d'adjacence
- Assignation des états

4: Synthèse

- Synthèse des variables secondaires
- Synthèse des sorties

10.2.1. Modélisation du cahier des charges

Tout comme pour les systèmes séquentiels synchrones, le cahier des charges d'un système est généralement donné en langage courant. Pour faire la synthèse d'un tel cahier des charges, la première étape est de l'exprimer dans un modèle mathématique. Le modèle généralement utilisé pour représenter le cahier des charges d'un système asynchrone est également un graphe appelé graphe d'état ou graphe de fluence.

10.2.1.a. Graphe d'état

Tout comme pour les systèmes séquentiels synchrones, les nœuds du graphe d'état représentent les états du système et les arcs orientés, les possibilités de passage d'un état à un autre. Par contre, le fait qu'en asynchrone, il n'y ait plus d'horloge introduit une différence notable dans l'interprétation du graphe. En effet, le passage d'un état à un autre ne se fait non plus sur un coup d'horloge mais directement lorsqu'une entrée varie. Cette différence de fonctionnement peut se traduire par une différence de structure du graphe d'état. En effet, chaque état étant stable tant qu'il n'y a pas de variation d'entrée, les entrées peuvent être portées sur les cercles représentant les états. Quant aux sorties, elles peuvent également être portées sur les cercles représentant les états puisqu'elles sont fonction des variables secondaires caractérisant ces états et des entrées qui sont maintenant portées sur les états (Figure 10.2.).

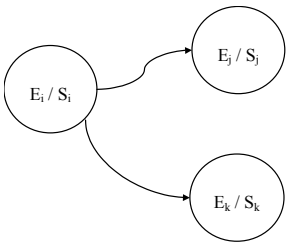


Figure 10.2. *Structure générale du graphe d'état d'un système séquentiel asynchrone*

Remarque : Dans le cas des systèmes séquentiels asynchrones, la notion de machine de Moore ou Mealey n'existe plus. En effet, les états dépendant directement des entrées (de manière combinatoire), les sorties dépendent donc toujours des entrées, que ce soit directement ou indirectement au travers de variables secondaires.

Exemple : Le système considéré a deux entrées e_1 et e_2 et une sortie S. La sortie S doit passer à 1 chaque fois que la séquence e_1e_2 = 10, 11, 01, 00 intervient sur les entrées. Quand S est à 1, S doit repasser à 0 dès la première variation d'une des deux entrées.

Le graphe d'état représentant ce cahier des charges est représenté sur la figure 10.3.

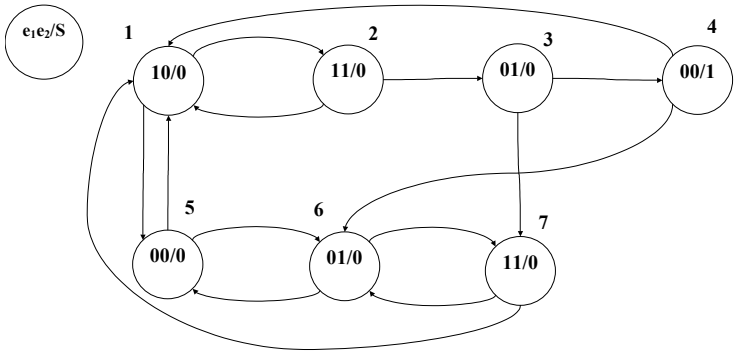


Figure 10.3. *Graphe de machine détectant les séquences 10/11/01/00*

10.2.1.b. Table d'état primitive

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 ou table des phases primitive, 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. De manière à repérer plus facilement les états stables et instables, les états stables seront surlignés (grisés). Outre le repérage des états stables, la seule différence avec les systèmes séquentiels synchrones est qu'il existe des états non-atteignables du fait de l'impossibilité de variation simultanée des entrées. Les sorties associées à chaque état sont également représentées sur cette table.

La table d'état correspondant au graphe d'état précédent est représentée sur la figure 10.4.

Etats	Etats Suivants 00	01	11	10	Sortie
1	5	-	2	1	0
2	-	3	2	1	0
3	4	3	7	-	0
4	4	6	-	1	1
5	5	6	-	1	0
6	5	6	7	-	0
7	-	6	7	1	0

Figure 10.4. *Table d'état de la machine détectant les séquences 10/11/01/00*

10.2.2. Minimisation du nombre d'états

10.2.2.a. Elimination des états équivalents

Le nombre d'états de la machine influe directement sur le nombre de variables secondaires et donc sur le nombre de portes nécessaires à la réalisation du circuit. Or, le nombre d'états utilisés pour représenter le cahier des charges, que ce soit sur le graphe des phases primitives ou sur la table d'état, n'est pas nécessairement minimum. Deux règles permettent de déterminer les états équivalents et par conséquent de minimiser le nombre d'états nécessaires.

- Règle R1 : Deux états sont équivalents s'il sont stables pour les mêmes valeurs d'entrée (état stable dans la même colonne), et si pour chaque combinaison d'entrée, ils ont même sorties et même états suivants.
- Règle R2 : Les états sont regroupés en différentes classes selon la position de l'état stable et les valeurs de sorties associées. Ainsi, deux états étant stables pour les mêmes valeurs d'entrée (état stable dans la même colonne) et ayant même 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 : Les règles de minimisation appliquées à la table des phases primitive de la Figure 10.4 donnent:

R1: Aucun état n'est stable pour les mêmes valeurs d'entrée, n'a la même sortie et les mêmes états suivants.

R2: les états peuvent être regroupés en cinq classes.

(1) (2) (3)	(4)	(5)	Classes
(1) (4) (5)	(3 , 6)	(2 , 7)	Etats
	437-/567-	-321/-671	Etats suivants
	245-/345-	-451/-451	Classes des états suiv.

Les états 3 et 6 doivent être dissociés et le processus réitéré.

(1) (2) (3) (4) (5)	(6)	Classes
(1) (4) (5) (3) (6)	(2 ,7)	Etats
	-321/-671	Etats suivants
	-461/-561	Classes des états suiv.

Les états 2 et 7 doivent être séparés. Il y a maintenant qu'un seul état par classe. Il n'y a donc pas d'états équivalents. Pour réaliser cette machine, le nombre minimum d'états est de 7.

Remarque : Il est beaucoup plus rare de trouver des états équivalents en asynchrone qu'en synchrone. Cela provient essentiellement du fait que la structure même du graphe (entrées sur les états) conduit à repérer beaucoup plus facilement les états équivalents et donc à éviter les duplications.

10.2.2.b. Fusionnement d'états

A priori, le nombre d'états obtenu après minimisation devrait nous donner le nombre de variables secondaires nécessaires à la réalisation de la machine. En fait, chaque état étant caractérisé par les valeurs d'entrées pour lequel il est stable, des états différents, stables pour des valeurs d'entrées différentes, peuvent très bien être codés de manières identiques. L'étape suivante est donc essayer de fusionner certains états pour, en leur attribuant le même code, diminuer le nombre de variables secondaires nécessaires à la réalisation de la machine.

Règle de fusionnement : Deux états sont fusionnables uniquement s'il ont mêmes état suivants (pas d'incompatibilité sur les états suivants compte tenu des états indéterminés).

Afin de déterminer les fusionnements qui conduisent à un nombre d'états minimum, on réalise un graphe appelé graphe de fusionnement. Les nœuds de ce graphe sont les états et il y a présence d'une arête entre deux nœuds si et seulement si les deux états correspondants sont fusionnables.

Le graphe de fusionnement correspondant à la table d'état de la Figure 10.4 et donné sur la Figure 10.5.

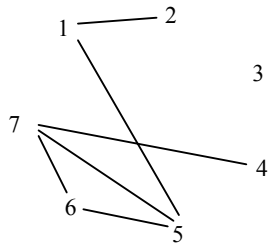


Figure 10.5. *Graphe de fusionnement*

Ce graphe donne les possibilités de regroupement d'états. Tous les états appartenant à un même groupement doivent être fusionnables entre eux. La recherche du nombre de groupement minimum est un problème classique (recherche de cliques dans un graphe) que nous ne détaillerons pas ici. Dans l'exemple précédent le nombre de groupements minimum est 4 mais comme le montre la Figure 10.6 il peut y avoir plusieurs solutions.

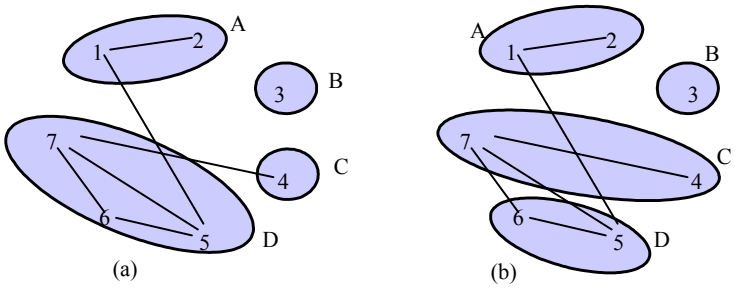


Figure 10.6. *Fusionnement d'états*

Remarque : Certaines solutions peuvent être "meilleures" que d'autres. Le choix d'une solution plutôt qu'une autre pourra s'appuyer sur des critères dont certains apparaîtront clairement par la suite. Notons dors et déjà que la solution (a) est à priori meilleure que la solution (b).

Une fois, les groupements déterminés, on peut réaliser le fusionnement des états composant un même groupement. Ceci conduit à une nouvelle table d'état appelée table des phases réduite. Lors de la fusion d'états, plusieurs situation peuvent se présenter. On peut avoir à fusionner des états stables, des états instables et des états indéterminés. La règle est la suivante:

- Un état stable et un état instable donnent un état stable.
- Un état stable et un état indéterminé donnent un état stable.
- Un état instable et un état indéterminé donnent un état instable

Exemple : En reprenant les groupements de la Figure 10.6.a on obtient la table. des phases réduite représentée sur la Figure 10.7.

Etats	Etats Suivants			
	00	01	11	10
A (1,2)	5	3	2	1
B (3)	4	3	7	-
C (4)	4	6	-	1
D (5,6,7)	5	6	7	1

Figure 10.7. *Table des phases réduite.*

10.2.3. Codage des états

Précédemment, nous avons noté qu'il n'est pas possible de prévoir un changement simultané des entrées. Pour les variables secondaires il en est de même. Si deux ou plusieurs variables secondaires nécessitent des changements de valeurs simultanés, il est impossible de garantir le fonctionnement de la machine. En conséquence, l'assignation des variables secondaires aux états d'une machine asynchrone réduite doit tenir compte de cette contrainte.

10.2.3.a. Courses et cycles

Ces notions seront abordées à partir de l'exemple de la table réduite de la Figure 10.10.

Sur cet exemple, lorsque l'état est y_1y_2=00 et que les entrées sont e_1e_2=00, l'état suivant est y_1y_2=11. La transition entre l'état 00 et l'état 11 introduit un changement des deux variables secondaires. La pratique impose qu'une seule variable y_1 ou y_2 commute à la fois. Selon le cas le système passera au préalable par l'état 01 ou 10. Ces états comportent dans l'exemple le même état instable 11 que l'état stable d'arrivée désiré.

Etats y_1y_2	Etats Suivants			
	00	01	11	10
00	11	00	10	01
01	11	00	11	01
11	11	00	10	11
10	11	10	10	11

Figure 10.8. *Table des phases réduite codée*

Définitions : Une situation nécessitant la variation de plus d'une variable secondaire est appelée une ***course***. Si l'état final que le système atteint ne dépend pas de l'ordre dans lequel les variables changent, alors la course est ***non critique***, dans le cas contraire la course est ***critique***.

Supposons maintenant que le système soit dans l'état y_1y_2=11 et que les entrées passent de 00 à 01. On désire par conséquent aller vers l'état stable 00. Si y_1 change plus vite que y_2, le circuit ira dans l'état 01 et atteindra finalement l'état stable 00. Par contre, si y_2 change plus vite que y_1, le système ira vers l'état stable 10 et s'y maintiendra. Nous avons une course critique et le fonctionnement du système sera incorrect.

Les courses critiques peuvent selon les cas être évitées, en dirigeant le système vers des états instables intermédiaires. Par exemple, lorsque le système est dans l'état y_1y_2=01 et e_1e_2=11, l'état d'arrivée voulu est 10. Mais puisque cette transition nécessite le changement simultané des deux variables secondaires y_1 et y_2, l'état instable est codé 11, dirigeant ainsi le système vers l'état 11, puis vers l'état 10.

Définition : La situation, pour laquelle le système passe par une séquence d'états instables est appelée ***cycle***. Lors de l'assignation des états, il est important de vérifier que chaque cycle se termine sur un état stable. Tout cycle ne comportant pas d'état stable appelé ***cycle instable*** doit absolument être évité.

10.2.3.b. Elimination des courses critiques

Sur l'exemple précédent afin d'éliminer la course critique de la colonne 01, on peut sélectionner un autre codage des variables secondaires (Figure 10.9).

Etats	Etats Suivants			
y_1y_2	00	01	11	10
00	10	00	10	01
01	10	00	11	01
10	10	00	11	10
11	10	11	11	10

Figure 10.9. *Assignation valide*

Définition : Une assignation qui ne contient ni course critique ni cycle instable est appelé une ***assignation valide***.

Remarque : Lorsqu'une colonne d'une table des transitions ne comporte qu'un seul état stable, il ne peut exister de course critique sur cet état. Les adjacences ne doivent pas être considérées sur cet état, toutefois il est dangereux de laisser libre les cases non spécifiées de la colonne, car au moment de l'écriture des fonctions logiques ces cases pourront être affectées à des valeurs conduisant à des cycles instables.

10.2.3.c. Méthode d'assignation

Dans de nombreuses situations, une assignation valide ne peut être obtenue par simple permutation des codes des états. Pour déterminer une assignation valide (pas de courses critiques ni de cycles instables) on doit étudier tous les passages d'états instables à états stables. Le passage d'un état instable à l'état stable correspondant peut se faire soit directement soit par l'intermédiaire d'états instables identiques ou d'états indéterminés.

La détermination d'une assignation valide peut être facilitée par la réalisation d'un graphe ou chaque état représente un sommet du graphe et chaque arc représente le fait que les deux états reliés doivent avoir un code adjacent. Ce graphe est appelé graphe d'adjacence. Afin de construire ce graphe, il est nécessaire d'introduire la notion de liaison essentielle et de liaison libre.

Définition : Lorsque pour aller d'un état instable à l'état stable correspondant, il n'y a pas d'autre solution que d'y aller directement, nous dirons qu'il y a une ***liaison essentielle*** entre les deux états. Lorsque pour aller d'un état instable à l'état stable correspondant, il y a plusieurs solutions nous dirons qu'il y a une ***liaison libre*** entre les deux états.

Toute liaison essentielle impose un arc sur le graphe d'adjacence. Toute liaison libre conduit à un ou plusieurs arcs mais le fait que la liaison soit libre laisse une certaine liberté quant à l'affectation de ces arcs. Une assignation valide, peut être déterminée en suivant la procédure suivante.

1. Analyser la table des phases réduite par colonnes et reporter sur le graphe d'adjacence les liaisons essentielles.
2. S'il n'existe pas de codage respectant les adjacences nécessaires, aller en 4.
3. Reporter sur le graphe d'adjacence les liaisons libres en choisissant une solution qui permette de coder les états en respectant les adjacences indiquées. Si une solution existe déterminer un codage respectant les adjacences indiquées par le graphe. Sinon aller en 4.

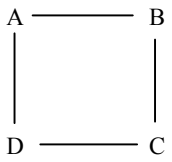
4. Rajouter une variable secondaire est recommencer en 3 (il n'existe plus de liaisons essentielles).

Remarque : S'il existe un état tel que tous les états suivants soientt indéterminés ou si le nombre d'états de la table des phases réduite n'est pas une puissance de 2, il n'existe pas de liaison essentielle.

Exemple : Reprenons la table des phases réduite de la Figure 10.7. Il existe des liaisons essentielles dans les colonnes e_1e_2=00 et e_1e_2=01 (Figure 10.10.a). Ces liaisons essentielles conduisent au graphe d'adjacence de la Figure 10.10.b .

Etats	Etats Suivants			
	00	01	11	10
A (1,2)	5	3	2	1
B (3)	4	3	7	-
C (4)	4	6	-	1
D (5,6,7)	5	6	7	1

(a)



(b)

Figure 10.10. *Liaisons essentielles*

On remarquera que sur ce graphe, si on doit rajouter un arc, il n'est plus possible de trouver un code respectant les adjacences indiquées avec deux variables. Il faut donc prendre en compte les liaisons libres en évitant de rajouter un arc. Ceci peut être fait en incluant les cycles présentés sur la Figure 10.11.a. Le graphe d'adjacence correspondant nous conduit à définir un codage adapté (Figure 10.11.b).

Etats	Etats Suivants			
	00	01	11	10
A (1,2)	5	3	2	1
B (3)	4	3	7	-
C (4)	4	6	-	1
D (5,6,7)	5	6	7	1

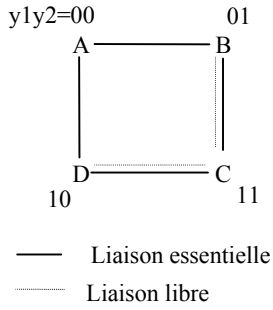


Figure 10.11. Liaisons essentielles et liaisons libres

Le codage de la table des phases réduite peut maintenant être réalisé (Figure 10.12). On remarquera le codage correspondant aux états (y_1y_2=01, e_1e_2=11) (y_1y_2=11, e_1e_2=11) (y_1y_2=11, e_1e_2=10). Ce codage est impératif pour que l'assignation soit valide.

Etats	Etats Suivants			
y_1y_2	00	01	11	10
00	10	01	00	00
01	11	01	11	--
11	11	10	10	10
10	10	10	10	00

Figure 10.12. *Assignation valide*

Remarque : Dans la quatrième colonne, il n'y a pas de possibilité d'avoir de course critique. L'affectation de la valeur 10 à l'état (y_1y_2=11, e_1e_2=10) n'est à priori pas obligatoire. Toutefois l'affectation de la valeur 00 à ce même état aurait entraîné l'affectation à 00 de l'état (y_1y_2=01, e_1e_2=10) afin d'éviter toute possibilité d'obtention d'un cycle instable.

10.2.3.d. Augmentation du nombre de variables secondaires

Il est souvent nécessaire d'augmenter le nombre de variables secondaires afin de résoudre le problème des courses critiques. Ce problème va être illustré sur l'exemple de la Figure 10.13.

Etats	Etats Suivants			
	00	01	11	10
A (2,6)	1	2	4	6
B (4,7)	1	3	4	7
C (5,8)	1	2	5	8
D (1,3)	1	3	5	6

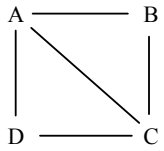


Figure 10.13. *Impossibilité d'assignation avec 2 variables*

Nous observons que la ligne a doit être adjacente aux trois autres lignes b, c et d. Bien évidemment, il est impossible d'effectuer des assignations adjacentes pour tous les états avec seulement deux variables d'états. Par conséquent, une troisième variable doit être ajoutée. L'ajout de cette variable à pour effet de créer des états supplémentaire (E,F,G,H) qui permettrons d'assurer des passages par code adjacent (Figure 10.14). Dans le cas général, pour une assignation comportant p variables d'états secondaires, chaque état peut être adjacent à p autres états au maximum

Etats	Etats Suivants			
	00	01	11	10
A (2,6)	1	2	4	6
B (4,7)	1	3	4	7
C (5,8)	1	2	5	8
D (1,3)	1	3	5	6
E	-	-	-	-
F	-	-	-	-
G	-	-	-	-
H	-	-	-	-

Figure 10.14. *Introduction d'une variable supplémentaire*

Les huit combinaisons de trois variables d'états sont représentées par les cases de la Figure 10.15. Pour déterminer une assignation valide, nous commençons par placer un état stable (A) dans la case $y_1y_2y_3$=000 pour indiquer que la ligne A sera assignée à l'état secondaire 000. De façon similaire, nous plaçons les états (B), (C) et (D) dans les trois cases adjacentes à la case (A). Cela implique toutefois que chacune des transitions des lignes B vers D et D vers C nécessite le changement de deux variables secondaires. Ces changements multiples peuvent être accomplis en amenant le système vers les destinations finales par l'intermédiaire d'états instables (Figure 10.15).

y_3 \ y_1y_2	00	01	11	10
0	A	C	F	D
1	B			E

Figure 10.15. *Assignation d'états*

Ce problème d'assignation est un problème classique d'algorithmique qui ne sera pas détaillé ici.

10.2.4. Synthèse

Cette dernière étape de la synthèse des machines asynchrones consiste à établir les fonctions logiques des variables secondaires ainsi que celles liées aux sorties de la machine.

10.2.4.a. Synthèse des variables secondaires

Une fois les assignations d'états réalisées sur la table des phases réduite nous avons toutes les information pour faire la synthèse des variables secondaires.

Exemple : En reprenant l'exemple précédent, à partir de la table de la Figure 10.12 on peut réaliser les tables de Karnaugh qui nous permettrons d'obtenir les relations de Y_1 et Y_2 (Figure 10.16).

y_1y_2 \ e_1e_2	00	01	11	10
00	1	0	0	0
01	1	0	1	-
11	1	1	1	1
10	1	1	1	0

$Y_1 = e_1'.e_2' + y_1.e_2 + y_2.e_1$

y_1y_2 \ e_1e_2	00	01	11	10
00	0	1	0	0
01	1	1	1	-
11	1	0	0	0
10	0	0	0	0

$Y_2 = y_1'.y_2 + y_2.e_1'.e_2' + y_1'.e_1'.e_2$

Figure 10.16. *Synthèse des variables secondaires*

Remarque : Rappelons que plusieurs outils de minimisation de fonctions logiques booléennes peuvent être utilisés: tableaux de Karnaugh, McCluskey, consensus ...

10.2.4.b. Synthèse des sorties

Pour l'instant, seuls les états stables possèdent des sorties spécifiées. Pour retrouver ces sorties, il suffit de se ramener à la table des phases réduite et à la table des phases primitive. Ces sorties peuvent également être présentées sous forme de table.

Exemple : La table de sortie présentée sur la Figure 10.17 est celle déduite de la table d'état réduite (Figure 10.7) et de la table d'état primitive (Figure 10.4).

Etats y_1y_2	Sorties 00	01	11	10
00	-	-	0	0
01	-	0	-	-
11	1	-	-	-
10	0	0	0	-

Figure 10.17. *Synthèse des sorties*

Notre but ici est de considérer l'assignation des sorties pour les états instables de la machine réduite. Cette assignation dépend des changements des sorties déjà spécifiées ainsi que des objectifs de conception (machine lente ou rapide).

Supposons que la valeur des sorties pour les états indéterminés nous soit indifférente. Dans ce cas, la valeur des sorties peut rester non spécifiée afin d'optimiser postérieurement la fonction logique de sortie. Mais dans ce cas, on s'expose à des commutations transitoires des sorties. En effet, supposons que le système doive aller d'un état stable vers un autre état stable pour lequel la sortie est identique. Si la sortie affectée à un des états instables intermédiaires est différente, la sortie présentera une commutation transitoire lors du passage entre ces deux états.

Supposons maintenant que le système doive aller d'un état stable vers un autre état stable pour lequel la sortie est différente. La valeur de sortie attribuée à l'état transitoire peut alors correspondre soit à l'état de départ soit à l'état d'arrivée. Ce choix doit être fait selon l'objectif désiré, à savoir si l'on désire un système commutant au plus tôt ou au plus tard. On parlera alors de machine lente et de machine rapide.

Exemple : En reprenant l'exemple précédant, les tables de sorties en machine rapide (a) et en machine lente sont données sur la Figure 10.18. Ces tables sont établies à partir de la connaissance des passages d'états instables à états stables (Figure 10.11).

Etats y_1y_2	Sorties 00	01	11	10
00	0	0	0	0
01	1	0	0	-
11	1	0	0	0
10	0	0	0	0

(a) Srapide = $y_2.e_1'.e_2'$

Etats y_1y_2	Sorties 00	01	11	10
00	0	0	0	0
01	0	0	0	-
11	1	1	0	1
10	0	0	0	0

(b) Slente = $y_1.y_2.(e_1' + e_2')$

Figure 10.18. *Machine rapide et machine lente*

10.3. Synthèse de dispositifs synchrones élémentaires

Un dispositif synchrone élémentaire est un composant synchronisé par une horloge. Une bascule est par exemple un dispositif synchrone élémentaire. Pour concevoir un tel système plusieurs démarches peuvent être envisagées. Il en est une qui est de considérer l'horloge H comme une entrée banalisée est ainsi de considérer le système global comme étant un système asynchrone. Dans ce chapitre nous réaliserons la synthèse des 3 bascules suivantes :

- La bascule D-Latch,
- La bascule D,

- La bascule D avec signaux de forçage « Clear » et « Preset »

10.3.a. Synthèse d'une D-Latch

Une bascule D-Latch est un dispositif comportant 2 entrées D et H et une sortie Q telles que :

- Si H = 1 (niveau 1) Q = D
- Sinon mémoire

Le graphe d'état d'une bascule D-Latch considéré comme un système asynchrone est donné sur la figure 10.19.

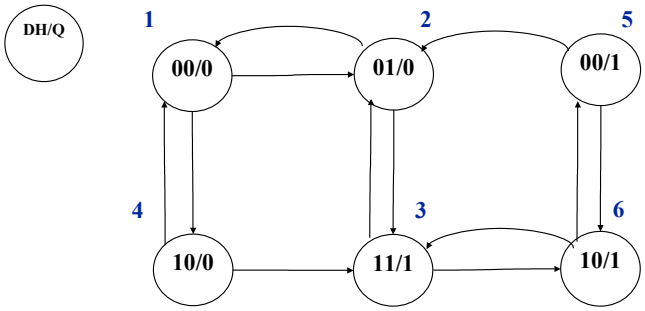


Figure 10.19. *Graphe d'état d'une D-Latch*

La table d'état correspondante est représentée sur la figure 10.20.

Etats	Etats Suivants				Sortie
	00	01	11	10	
1	1	2	-	4	0
2	1	2	3	-	0
3	-	2	3	6	1
4	1	-	3	4	0
5	5	2	-	6	1
6	5	-	3	6	1

Figure 10.20. *Table d'état d'une D-Latch*

L'application des deux règles de minimisation du nombre d'état ne donnant rien, il n'y a pas, parmi ces états, d'états équivalents. Le graphe de fusionnement est représenté sur la figure 10.21.

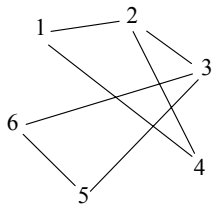


Figure 10.21. *Graphe de fusionnement*

En fusionnant les états 1,2,4 et 3,5,6, la table d'état devient (Figure 10.22)

Etats	Etats Suivants			
	00	01	11	10
A(1,2,4)	1	2	3	4
B(3,5,6)	5	2	3	6

Figure 10.22. *Table d'état réduite d'une D-Latch*

En codant 0 l'état A et 1 l'état B les tables d'état et de sortie sont les suivantes (Figure 10.23):

Etats y	Etats Suivants 00	01	11	10
0	0	0	1	0
1	1	0	1	1

Etats y	Sortie 00	01	11	10
0	0	0	1	0
1	1	0	1	1

Figure 10.23. *Table d'état codée et table de sortie d'une D-Latch*

La structure du système (D-Latch) ainsi obtenu est présenté sur la figure 10.24.

y = H'y + H.D

Q = y

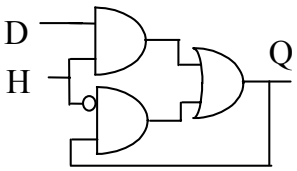


Figure 10.24. *Bascule D-Latch*

10.3.b. Synthèse d'une bascule D

Une bascule bascule D est un dispositif comportant 2 entrées D et H et une sortie Q telles que :

- Si H passe de 0 à 1 (front montant) Q = D
- Sinon mémoire

Le graphe d'état d'une bascule D considéré comme un système asynchrone est donné sur la figure 10.25.

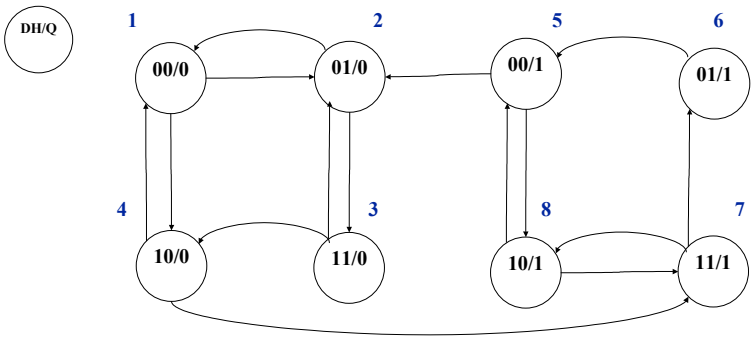


Figure 10.25. *Graphe d'état d'une bascule D*

La table d'état correspondante est représentée sur la figure 10.26.

Etats	Etats Suivants				Sortie
	00	01	11	10	
1	1	2	-	4	0
2	1	2	3	-	0
3	-	2	3	4	0
4	1	-	7	4	0
5	5	2	-	8	1
6	5	6	7	-	1
7	-	6	7	8	1
8	5	-	7	8	1

Figure 10.26. *Table d'état d'une bascule D*

L'application des deux règles de minimisation du nombre d'état ne donnant rien, il n'y a pas, parmi ces états, d'états équivalents. Le graphe de fusionnement est représenté sur la figure 10.27.

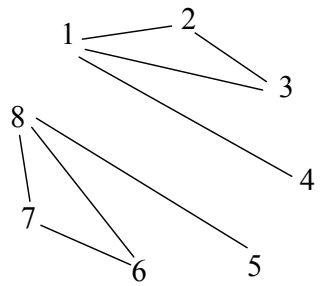


Figure 10.27. *Graphe de fusionnement*

En fusionnant les états 1,2,3 et 6,7,8, la table d'état devient (Figure 10.28)

Etats	Etats Suivants			
	00	01	11	10
A(1,2,3)	1	2	3	4
B(4)	1	-	7	4
C(5)	5	2	-	8
D(6,7,8)	5	6	7	8

Figure 10.28. *Table d'état réduite d'une bascule D*

Le graphe d'adjacence est présenté sur la figure 10.29.

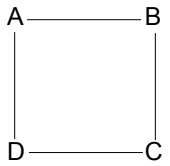


Figure 10.29. *Graphe d'adjacence*

En codant 00 l'état A, 01 l'état B, 11 l'état C et 10 l'état D, les tables d'état et de sortie sont les suivantes (Figure 10.30):

Etats	Etats Suivants			
y_1y_2	00	01	11	10
00	00	00	00	01
01	00	00	11	01
11	11	01	10	10
10	11	10	10	10

Etats	Sorties			
y_1y_2	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	0	1	1
10	1	1	1	1

Figure 10.30. *Table d'état codée et table de sortie d'une bascule D*

La structure d'une bascule D peut ainsi être obtenue à partir des équations suivantes :

$y_1 = y_1.y_2' + y_1.H' + y_2.D.H$

$y_2 = y_1.D'.H' + y_1.y_2.D' + y_1'.y_2.D + y_1'.D.H'$

$Q = y_1$

10.3.c. Synthèse d'une bascule D avec signaux de forçage « Clear » et « Preset »

Une bascule bascule D avec signaux de forçage est un dispositif comportant 2 entrées D et H, 2entrées de forçage Clear et Preset et une sortie Q telles que :

- Si Clear=Preset = 0
  - Si H passe de 0 à 1 (front montant) Q = D
  - Sinon mémoire
- Si Clear = 1 (Preset = 0), Q = 0
- Si Preset = 1 (Clear = 0), Q = 1

Le graphe d'état d'une bascule D considéré comme un système asynchrone est donné sur la figure 10.31.

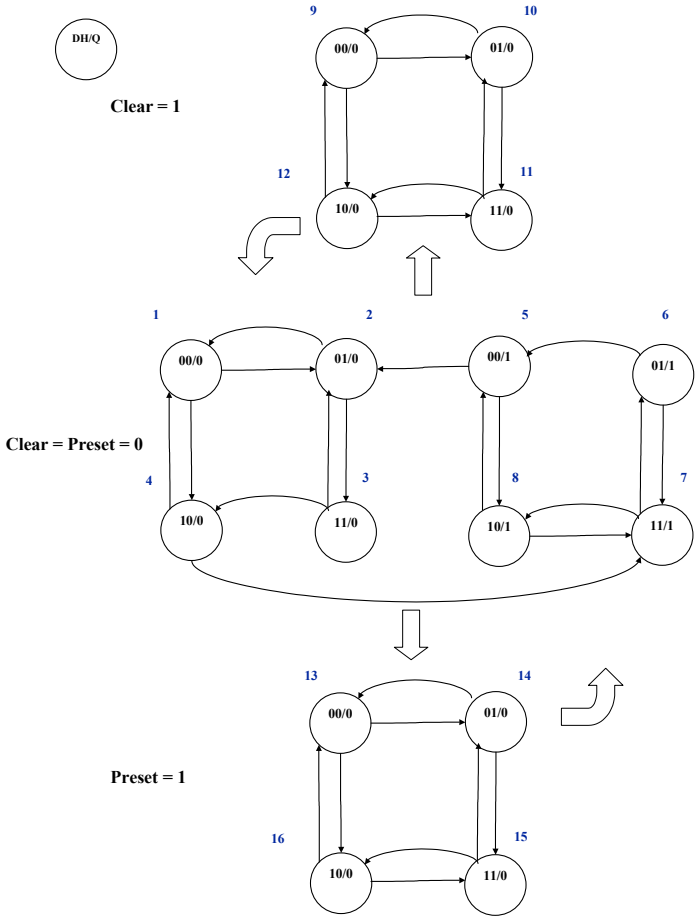


Figure 10.31. *Graphe d'état d'une bascule D avec Clear et Preset*

La table d'état correspondante est représentée sur la figure 10.32.

Etats	Etats Suivants												Sortie
	Clear=Preset=0				Clear = 1				Preset = 1				Q
	00	01	11	10	00	01	11	10	00	01	11	10	
1	1	2	-	4	9	-	-	-	13	-	-	-	0
2	1	2	3	-	-	10	-	-	-	14	-	-	0
3	-	2	3	4	-	-	11	-	-	-	15	-	0
4	1	-	7	4	-	-	-	12	-	-	-	16	0
5	5	2	-	8	9	-	-	-	13	-	-	-	1
6	5	6	7	-	-	10	-	-	-	14	-	-	1
7	-	6	7	8	-	-	11	-	-	-	15	-	1
8	5	-	7	8	-	-	-	12	-	-	-	16	1
9	1	-	-	-	9	10	-	12	-	-	-	-	0
10	-	2	-	-	9	10	11	-	-	-	-	-	0
11	-	-	3	-	-	10	11	12	-	-	-	-	0
12	-	-	-	4	9	-	11	12	-	-	-	-	0
13	5	-	-	-	-	-	-	-	13	14	-	16	1
14	-	6	-	-	-	-	-	-	13	14	15	-	1
15	-	-	7	-	-	-	-	-	-	14	15	16	1
16	-	-	-	8	-	-	-	-	13	-	15	16	1

Figure 10.32. *Table d'état d'une bascule D avec Clear et Preset*

L'application des deux règles de minimisation du nombre d'état ne donnant rien, il n'y a pas, parmi ces états, d'états équivalents. Le graphe de fusionnement est représenté sur la figure 10.33.

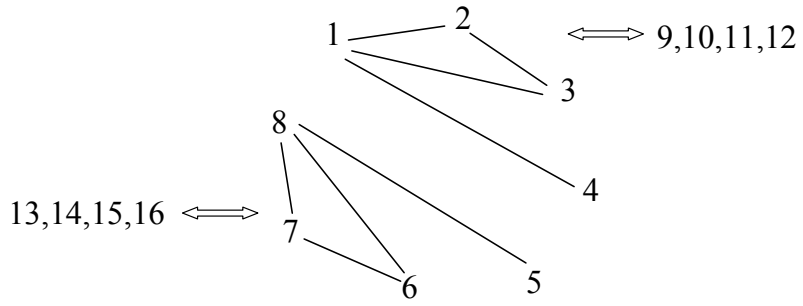


Figure 10.33. *Graphe de fusionnement*

En fusionnant les états 1,2,3,9,10,11,12 et 6,7,8,13,14,15,16, la table d'état devient (Figure 10.34)

Etats	Etats Suivants											
	Clear=Preset=0				Clear = 1				Preset = 1			
	00	01	11	10	00	01	11	10	00	01	11	10
A	1	2	3	4	9	10	11	12	13	14	15	-
B	1	-	7	4	-	-	-	12	-	-	-	16
C	5	2	-	8	9	-	-	-	13	-	-	-
D	5	6	7	8	-	10	11	12	13	14	15	16

Figure 10.34. *Table d'état réduite d'une bascule D avec Clear et Preset*

Le graphe d'adjacence est présenté sur la figure 10.35

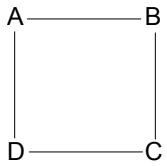


Figure 10.35. *Graphe d'adjacence*

En codant 00 l'état A, 01 l'état B, 11 l'état C et 10 l'état D, la table d'état codée (Figure 10.36) et la table de sortie (Figure 10.37) sont les suivantes :

Etats y_1y_2	Etats Suivants											
	Clear=Preset=0				Clear = 1				Preset = 1			
	00	01	11	10	00	01	11	10	00	01	11	10
00	00	00	00	01	00	00	00	00	10	10	10	-
01	00	00	11	01	00	-	-	00	-	-	-	11
11	11	01	10	10	01	-	-	-	10	-	-	10
10	11	10	10	10	-	00	00	00	10	10	10	10

Figure 10.36. *Table d'état codée d'une bascule D avec Clear et Preset*

Etats y_1y_2	Sortie											
	Clear=Preset=0				Clear = 1				Preset = 1			
	00	01	11	10	00	01	11	10	00	01	11	10
00	0	0	0	0	0	0	0	0	1	1	1	-
01	0	0	1	0	0	-	-	0	-	-	-	1
11	1	0	1	1	0	-	-	-	1	-	-	1
10	1	1	1	1	-	0	0	0	1	1	1	1

Figure 10.37. *Table d'état codée d'une bascule D avec Clear et Preset*

La structure d'une bascule D avec signaux de forçage Clear et Preset peut ainsi être obtenue à partir des équations suivantes :

$y_1 = Clear'.Preset'.[y_1.y_2' + y_1.H' + y_2.D.H] + Preset$

$y_2 = Clear'.Preset'.[y_1.D'.H' + y_1.y_2.D' + y_1'.y_2.D + y_1'.D.H'] + Clear.y_1.y_2 +$
$Preset.y_1'.y_2$

$Q = y_1$