Chapitre

6

Circuits logiques séquentiels

Les circuits logiques combinatoires ne suffisent pas à eux seuls à la manipulation de l'information comme cela se fait dans les systèmes numériques modernes. Le caractère figé des circuits combinatoires — que traduit la correspondance stricte entre les entrées et les sorties — limite considérablement le champ de leurs applications. C'est là que les circuits séquentiels prennent toute leur importance. Ces derniers permettent la mise au point de systèmes dont le fonctionnement dépend non plus seulement des entrées reçues, mais également des informations traitées précédemment dans le cours de leur fonctionnement. On comprend alors qu'une forme de mémoire est mise en œuvre, une mémoire permettant au circuit de **se souvenir** des évènements passés et de traiter l'information plus adéquatement. Dans un premier temps, nous allons tâcher de définir les outils nous permettant de construire de tels circuits. Par la suite, nous considérons les techniques de conception permettant d'en tirer toute la puissance calculatoire.

6.1 Circuits logiques séquentiels

Un circuit logique séquentiel est un circuit logique possédant des entrées et des sorties et présentant un comportement où les sorties ne dépendent pas seulement des entrées, mais également des séquences des entrées passées. Pour ce faire, le circuit utilise une partie mémoire qui va lui permettre de retrouver l'**état** induit par les entrées passées. La sortie est par conséquent calculée en fonction de l'état présent et des entrées qui arrivent au système. Ce concept d'état sera largement exploré dans le reste de ce chapitre ; aussi nous concentrerons-nous d'abord sur les composants de type mémoire que nous aurons à utiliser.

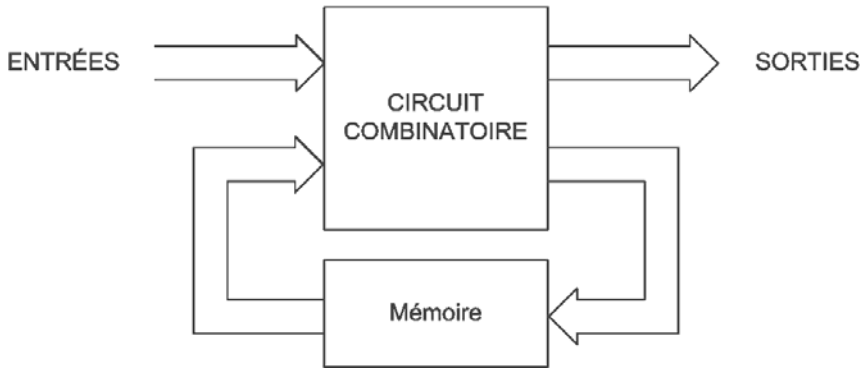


Figure 6.1 Schéma générique d'un circuit séquentiel

6.2 Éléments de mémoire

Au chapitre 4, nous avions vu que la ROM se comportait comme une mémoire adressable. Les éléments de mémoire considérés ici diffèrent de la ROM en plusieurs points, notamment en ce qu'ils sont d'un type dit séquentiel. Le caractère séquentiel de ces nouveaux éléments de mémoire renvoie au fait que le temps de consultation des données dans le circuit joue tout un rôle dans son fonctionnement. Pour parvenir à ce résultat, il faut jouer sur les délais de propagation des signaux dans les portes logiques ou introduire une rétroaction.

6.2.1 Délai de propagation des portes logiques

Jusqu'à présent, nous avions considéré que les circuits combinatoires agissaient de manière instantanée, associant dans l'immédiateté des sorties à une combinaison d'entrées donnée. Il n'en va pas ainsi des circuits combinatoires en réalité. Chaque porte logique possède des propriétés physiques qui font que les signaux logiques à l'entrée mettent du temps avant qu'il n'y ait d'incidence sur la valeur logique à la sortie. Considérons par exemple le circuit combinatoire suivant :

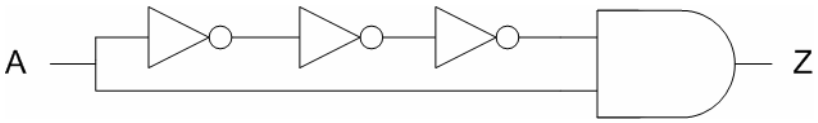


Figure 6.2 Circuit combinatoire avec chemins de propagation déséquilibrés

L'équation algébrique donnant Z en fonction de A nous dit que Z devrait est constamment nul, peu importe la valeur de A :

$$Z(A) = A \cdot \overline{\overline{\overline{A}}} = A \cdot \overline{A} = 0$$

Cependant, si on considère le chronogramme des signaux A et Z lors d'une transition dite de front montant (transition de $0 \rightarrow 1$) sur A, il apparaît que le signal Z prend la valeur 1 durant un très court laps de temps. On nomme cette valeur transitoire un aléa, *hazard* en anglais. Dans la pratique, un tel phénomène est également désigné par son nom anglais moins formel : *glitch*.

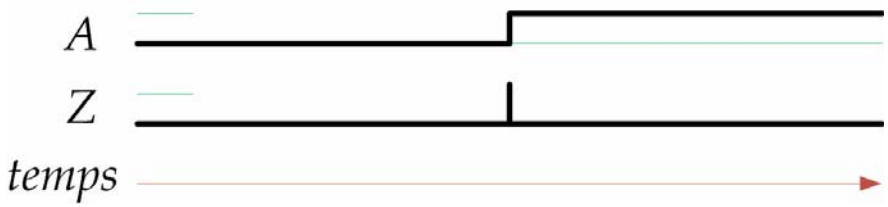


Figure 6.3 Chronogramme du front montant sur l'entrée A

En considérant le temps de propagation du signal A au travers des différentes portes, l'aléa observé sur le signal Z peut être expliqué facilement. Supposons que le temps de propagation soit égal pour toutes les portes. Appelons ce délai τ. Cette hypothèse est raisonnable dans un premier temps d'analyse et sera admise dans le restant de ce chapitre. En vérité, chaque porte logique possède un temps de propagation propre résultant du processus de fabrication. Ces variabilités compliquent d'autant plus l'analyse et demandent dans la pratique de sonder le circuit en laboratoire avec des instruments très précis.

Afin de mieux comprendre l'aléa observé sur le signal Z, supposons que nous disposions de sondes numériques nous permettant de suivre l'évolution des signaux au travers du circuit à différents endroits, comme indiqué à la figure 6.4 :

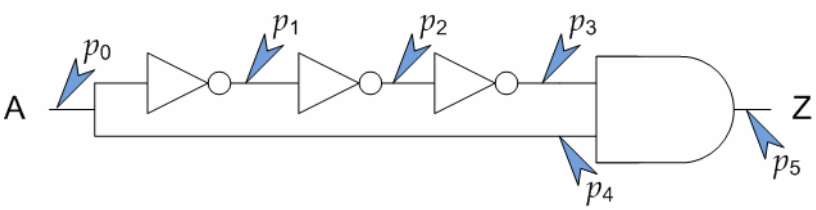


Figure 6.4 Sondes sur le circuit déséquilibré

La figure 6.5 présente le chronogramme où a été enregistré un front montant sur A. La sonde p_0 nous permet d'observer A. Le signal sur la sonde p_4 est synchrone avec p_0 du fait que A ne traverse aucune porte du point p_0 au point p_4. La sortie de l'inverseur entre les points p_0 et p_1 prend un temps τ avant de réagir au front montant sur A. Ce délai peut être observé sur le signal p_1. Le même phénomène est observé sur les inverseurs subséquents, respectivement entre les points p_1 et p_2 et les points p_2 et p_3.

Les sondes p_3 et p_4 nous donnent l'enregistrement des signaux à l'entrée de la porte ET dont la sortie (p_5) passe à 1 quand les entrées valent 1 en même temps, ce qui est le cas durant un temps égale à 3τ. Ce phénomène a lieu après un temps de retard τ égal au temps de propagation de la porte ET.

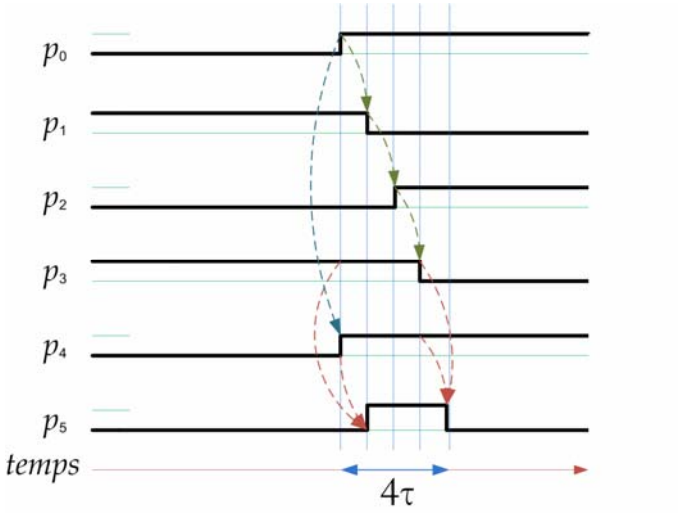


Figure 6.5 Chronogramme de front montant sur A

6.2.2 Phénomène de mémoire

Plutôt que de constituer un problème, les aléas peuvent être exploités pour créer un phénomène de mémoire. Pour ce faire, il suffit d'introduire judicieusement un chemin de rétroaction dans les circuits combinatoires entre ses sorties et ses entrées. Cette technique va nous permettre de construire une grande variété de composants numériques que nous allons découvrir au cours de cette section.

Afin de mieux comprendre cette technique, considérons le circuit simple de la figure 6.6. Il s'agit d'une simple porte XOR dont une des entrées est alimentée par la sortie du XOR. Trois sondes p_0, p_1 et p_2 sont

posées le long des chemins de propagation des signaux, respectivement aux deux entrées et à la sortie. Le chemin de rétroaction allant de p_2 à p_1 ne subit aucun délai étant donné l'absence de porte logique et les signaux sont synchrones. L'entrée où est posée la sonde p_0 constitue l'unique entrée A du circuit rétroactif tandis que la sortie où est posée la sonde p_2 forme son unique sortie Z.

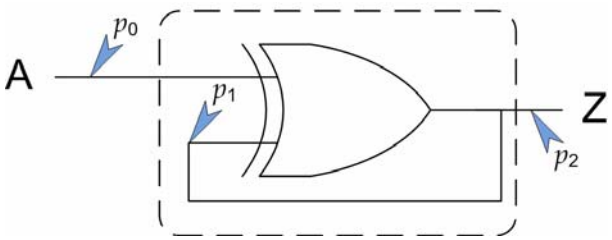


Figure 6.6 XOR avec chemin de rétroaction

Supposons que le circuit de la figure 6.6 se trouve dans un *état* initial où l'entrée et la sortie sont à 0. Considérons maintenant le chronogramme de la figure 6.7 présentant le comportement du circuit depuis son état initial jusqu'au moment où un front montant apparaît sur l'entrée A. Au moment où le signal passe à 1 sur le point p_0, les deux entrées du XOR sont l'inverse l'une de l'autre et la sortie Z (p_2) passe à 1 avec un délai τ. À ce moment, les deux entrées ont la même valeur 1 et la sortie Z passe à 0 après un autre délai τ.

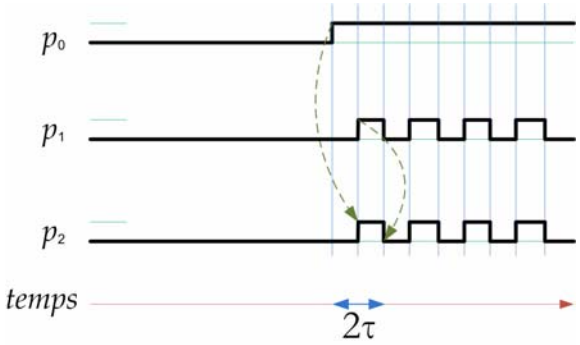


Figure 6.7 Chronogramme de la technique de rétroaction sur un XOR

La sortie Z devient alors instable et continue à osciller entre ces deux valeurs tant que l'entrée A ne repasse pas à 0. À ce moment, la sortie Z peut autant valoir 0 que 1 selon la dernière valeur de Z. La figure 6.8 illustre ces deux cas de figure.

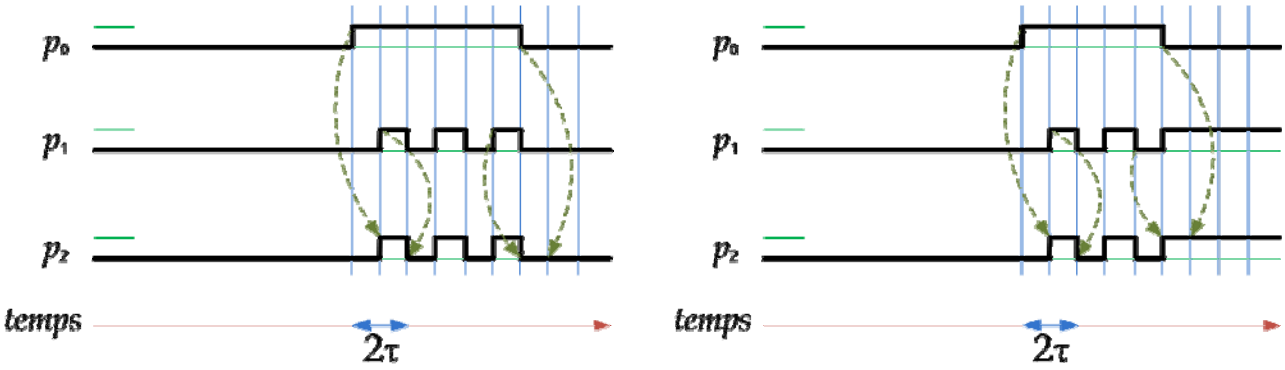


Figure 6.8 Variabilité de l'état final de Z après le pulse sur A

Si la durée du pulse sur A était connue à l'avance, il serait éventuellement possible d'utiliser le circuit de la figure 6.6 pour des tâches de mémorisation. Les pulses à durée $n \cdot \tau$, à n impair, laisse la sortie Z à 1. En combinant le circuit de la figure 6.6 avec un circuit provoquant à chaque front (montant ou descendant) un aléa de $1 \cdot \tau$, comme l'indique la figure 6.9, on peut construire un circuit permettant de détecter la présence d'un front montant ou d'un front descendant sur un signal d'entrée x. Il est laissé au lecteur le détail de l'exercice aboutissant aux phases transitoires.

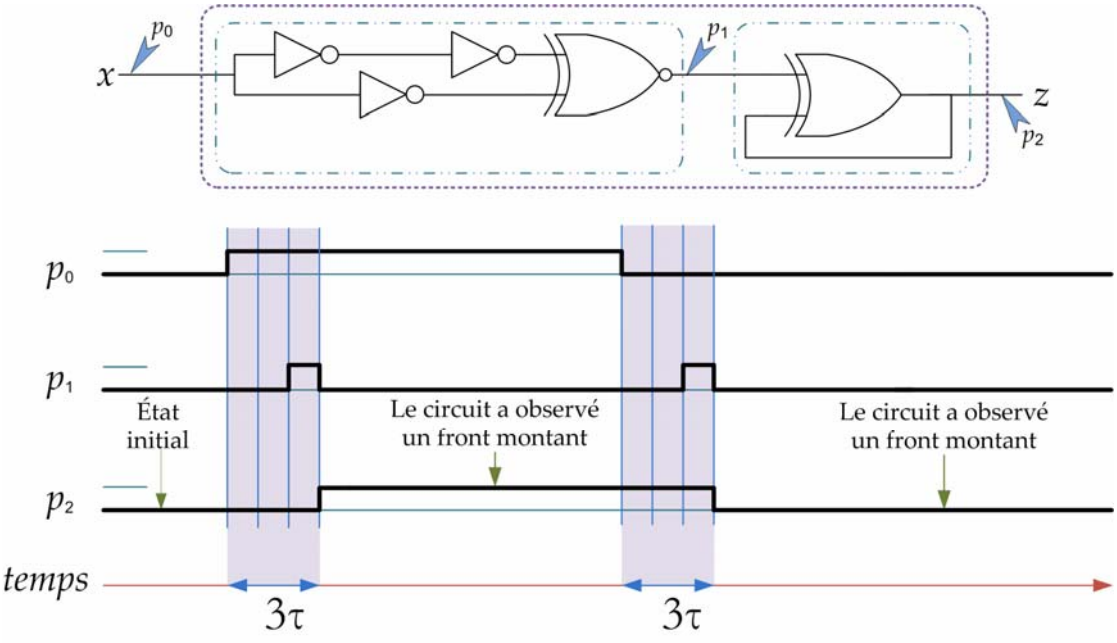


Figure 6.9 Circuit de détection des fronts montant et descendant. En excluant l'état initial, la sortie z indique l'observation d'un front montant si elle vaut 1, un front descendant si elle vaut 0. Les zones ombragées indiquent les phases transitoires du circuit.

Le comportement du circuit peut être schématisé par un diagramme d'états où les états du circuit sont indiqués par des cercles, les transitions par des flèches sur lesquelles sont portées les conditions desdites transitions. L'état initial est indiqué par deux cercles concentriques qui correspondent aux configurations initiales hypothétiques du circuit (l'entrée et la sortie du circuit sont à 0).

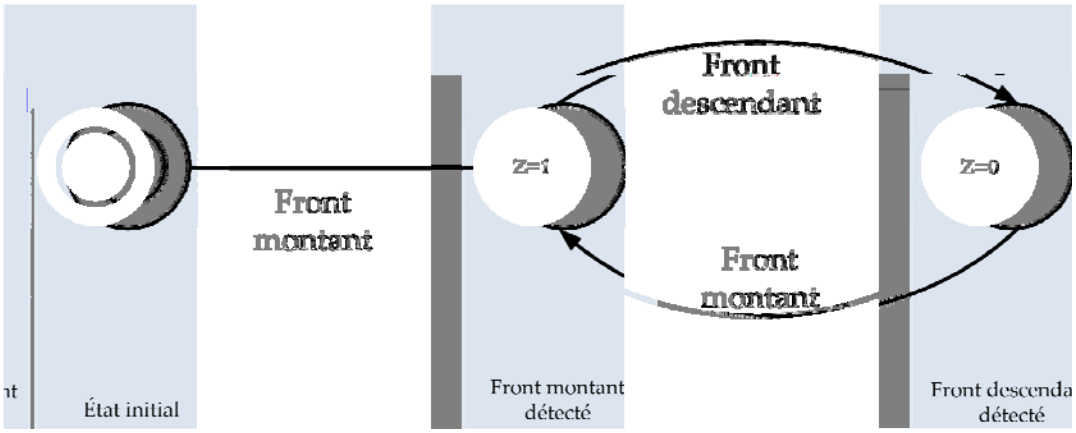


Figure 6.10 Diagramme d'état de détection des fronts montant et descendants.

6.2.3 Circuit bistable SR

Le circuit de la figure 6.9 nous a permis d'observer un exemple de phénomène de mémoire dû aux chemins de rétroaction dans un circuit logique. Cependant, le diagramme d'état de la figure 6.10 peut être modifié pour témoigner des limitations du système.

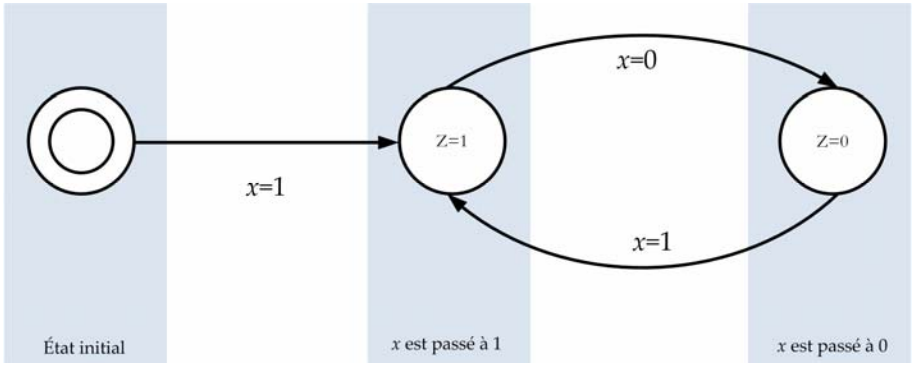
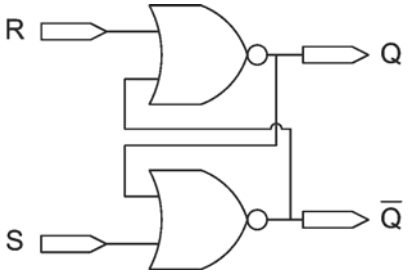


Figure 6.11 Diagramme d'état modifié de la détection des fronts montant et descendants.

À la lumière de cette nouvelle interprétation, il est clair le signal $z = x$ après un délai 3τ. Aussi, s'il est vrai que le circuit de la figure 6.9 a démontré une forme de mémoire, sont utilisation demeure fort limité et de peu d'applicabilité. Nous allons donc considérer un circuit plus évolué illustrant mieux les propriétés mnémoniques des circuits logiques à rétroaction.

Le circuit bistable SR est un circuit logique à rétroaction simple permettant d'enregistrer un bit. Le nom SR vient du fait que la bistable possède deux entrées, S et R, renvoyant respectivement à Set et à Reset. Lorsque l'entrée S est à 1, le circuit enregistre un 1 à sa sortie Q. Lorsque l'entrée R passe à 1, le circuit est réinitialisé et il enregistre 0 à sa sortie Q. Cela n'est possible que si S et R ne valent pas 1 en même temps. Si S et R valent tous deux 0, le système est stable et mémorise la dernière valeur enregistrée.



Bistable SR

S	R	Q^+	$\overline{Q}^+$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	0	0

Table de vérité

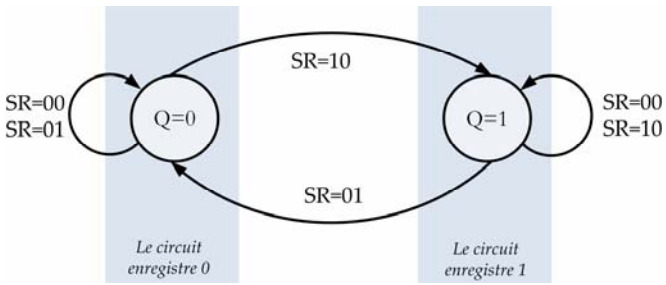


Figure 6.12 Bistable SR, sa table de vérité et son diagramme d'états.

Contrairement au diagramme d'états de la figure 6.11, celui de la figure 6.12 indique un fonctionnement du système plus intéressant qu'un délai. Le système possède deux états stables, d'où la désignation du circuit SR par le nom de bistable SR. La table de vérité de la figure 6.12 dernière colonne est notée Q^+, le + en exposant est là pour indiquer le fait qu'il s'agit de l'état futur de la sortie (par opposition à l'état actuel de la sortie : Q).

Afin d'illustrer le fonctionnement de la bistable, supposons qu'elle soit à un état quelconque et que les entrées SR=00. Nous allons suivre le fonctionnement du circuit lors de diverses excitations sur S et R. On suppose que les signaux S et R varient assez lentement et que S et R ne valent pas 1 en même temps. Pour ce faire, il faut remettre le signal activé (S ou R) à 0 avant d'activer le second.

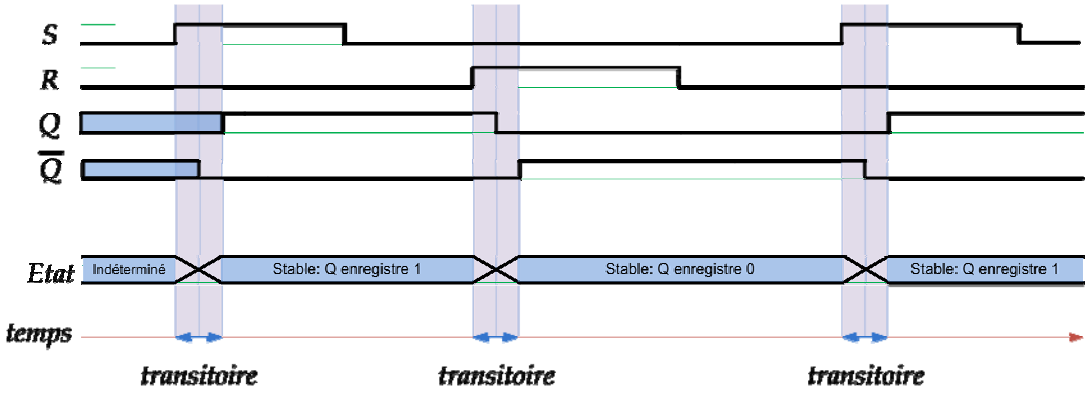
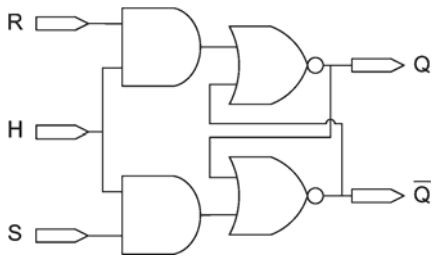


Figure 6.13 Chronogramme de la Bistable SR.

Au départ, la bistable se trouve dans un état inconnu. Lorsque l'entrée S passe à 1, la sortie !Q passe à 0 après un temps de propagation τ. À ce moment, les signaux R et !Q valent 0 et la sortie Q passe à 1 après un temps τ. Le système se trouve alors dans un état stable qui se prolonge lorsque S retombe à 0. La période transitoire dure 2 τ. Lorsque R passe à 1, la sortie Q passe à 0 après un temps τ . L'entrée S valant 0, la sortie !Q passe à 1 après un délai τ et le système est stable. Lorsque R retombe à 0, le système demeure stable et la sortie ne change pas tant et autant que ces opérations d'activation ou de réinitialisations ne sont pas effectuées de nouveau.

6.2.4 Bistable SR avec signal d'activation

Il peut arriver que l'on veuille commander le fonctionnement de la bistable SR avec un signal d'activation périodique H (appelé horloge). La bistable fonctionne alors normalement quand H est actif (H=1) et mémorise la dernière valeur enregistrée lorsque H est inactif (H=0). Il suffit pour cela d'introduire des portes ET aux entrées, ce qui nous permet d'obtenir :



Circuit de la bistable SR avec signal d'activation H

H	S	R	Q^+
0	-	-	Q
1	0	0	Q
1	0	1	0
1	1	0	1
1	1	1	1

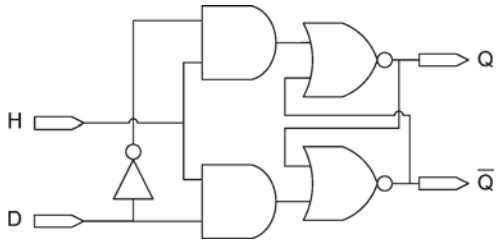
Table de vérité rattachée

Figure 6.14 Bistable SR avec signal d'activation H et sa table de vérité.

Ici encore, il faut éviter SR=11, mais cela uniquement dans le cas où H vaut 1. L'utilisation du signal H peut être imaginée dans le cas d'un grand système où plusieurs bistables travaillent en interaction. On veut figer le système dans l'état où il se trouve. On pose alors H=0, et toutes les mémoires demeurent au dernier état enregistré.

6.2.5 Bistable D

La bistable D dérive de la SR avec signal d'activation H. Le principe est de réunir les signaux SR par un seul signal D qui correspond à la donnée que l'on veut écrire dans la mémoire de la bistable. On part du constat que dans le cas de la bistable SR avec signal d'activation H, le cas HSR=100 est redondant avec celui où H=0. Il devient clair alors que les signaux S et R sont constamment l'inverses l'un de l'autre. On peut alors construire le circuit suivant :



Circuit de la bistable SR avec signal d'activation H

H	D	Q^+
0	-	Q
1	0	0
1	1	1

Table de vérité rattachée

Figure 6.15 Bistable D avec signal d'activation H et sa table de vérité.

La bascule conserve sa valeur (sur le signal Q) lorsque H veut 0, elle prend celle de D si H vaut 1. On dit d'une telle bistable qu'elle est transparente car elle correspond à un fil liant D à Q lorsque H vaut 1 (les délais de propagations mis à part).

6.2.6 Bascule D de type maître esclave

On pourrait vouloir disposer d'un circuit de mémoire qui fonctionne non pas sur la durée d'activation (fonctionnement transparent) du signal H mais à un moment précis : les fronts montant 0⟶1 ou descendant 1⟶0 du signal d'horloge H pourraient alors être utilisés pour indiquer ce changement. La bascule D de type maître esclave utilise une bistable D pour réussir cette performance. L'idée est de cascader deux bistables qui vont fonctionner en conjonction. Pour ce faire, les deux bistables partagent le signal de commande H, à cette différence que la seconde (appelée esclave) reçoit l'inverse du signal H.

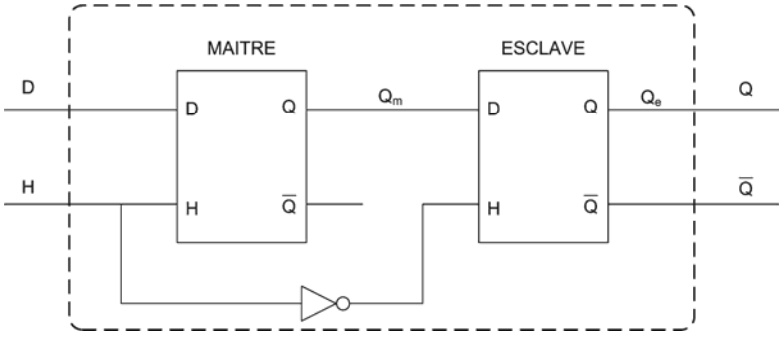
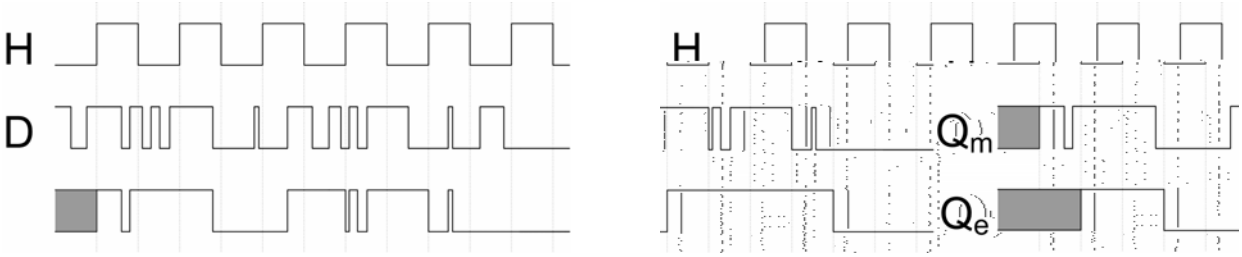


Figure 6.16 Bascule D de type maître esclave.

Le système va passer par deux phases de transition dans une période de H. Pour nous en convaincre, regardons les chronogrammes suivants :



Chronogramme relatif au maître **Chronogramme relatif à l'esclave**

Figure 6.17 Chronogrammes de la bascule D de type maître esclave.
Les zones ombragées sont des états non connus.

Ce que nous montrent ces deux chronogrammes, c'est le comportement de la bistable D. Dans le premier chronogramme, Q_m suit exactement le comportement de D lorsque H est à 1. Si H est à 0, Q_m conserve la dernière valeur retenue (Celle de D précédant le front descendant). Notons également qu'une partie du chronogramme de Q_m est surlignée en gris pour indiquer que la valeur du signal Q_m n'est pas connue durant cette période. Dans le second chronogramme, Q_e suit exactement le comportement de Q_m lorsque H est à 0 à cause de l'inversion du signal d'horloge à l'entrée. Lorsque H est à 1, la dernière valeur retenue est maintenue. Or rappelons que durant ce temps, le maître conserve la dernière valeur retenue sur le front descendant. Ainsi, le système (la bascule) est sûr de conserver à sa sortie (et durant toute une période de H) la valeur du front descendant comme l'illustre le chronogramme global suivant :

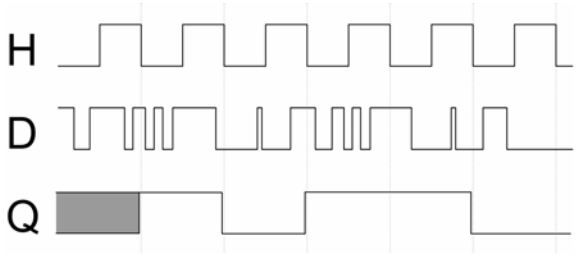
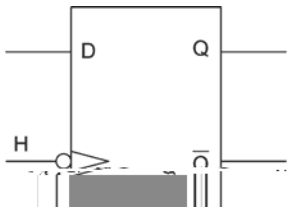


Figure 6.18 Chronogramme global de la bascule D de type maître esclave.

Ce dernier chronogramme nous montre que la sortie Q (équivalente de Q_e) prend la valeur de D sur le front descendant (indiqué par les traits verticaux en pointillés). Notons que le nom même de bascule fait référence au basculement de la valeur de la sortie Q au moment précis où le signal d'horloge H passe de 1 à 0 (front descendant).



Bascule D de type maître esclave

H	D	Q^+
0	-	Q
1	-	Q
↓	0	0
↓	1	1

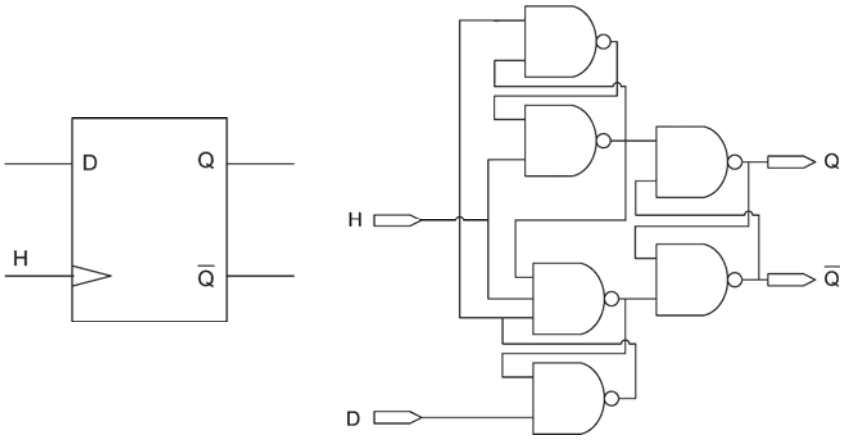
Table de vérité rattachée

Figure 6.19 Bascule D de type maître esclave et sa table de vérité.

La figure 6.19 présente la bascule D de type maître esclave et sa table de vérité. La flèche orientée vers le bas dans la table de vérité rend compte du front descendant de l'horloge qui provoque le changement de valeur à la sortie. Sur le symbole de la bascule D de type maître esclave, c'est le triangle à l'entrée qui représente la sensibilité au front — le cercle d'inversion indique qu'il s'agit d'un front descendant.

6.2.7 Bascule D

Il existe un autre modèle de bascule D qui n'est pas réalisé sur la base de la bistable D. Cette bascule réagit au front montant et permet une économie de ressources, comme le montre la figure suivante :



H	D	Q^+
0	-	Q
1	-	Q
↑	0	0
↑	1	1

Bascule D **Implémentation de la bascule D** **Table de vérité**

Figure 6.20 Bascule D, son implémentation et sa table de vérité.

Comme l'illustre le symbole et la table de vérité correspondante, cette bascule fonctionne exactement de la même façon qu'une bascule D de type maître esclave, à cette différence près que la transition est faite selon le front montant de l'horloge, et non pas sur le front descendant.

À titre d'illustration du fonctionnement, considérons le chronogramme suivant :

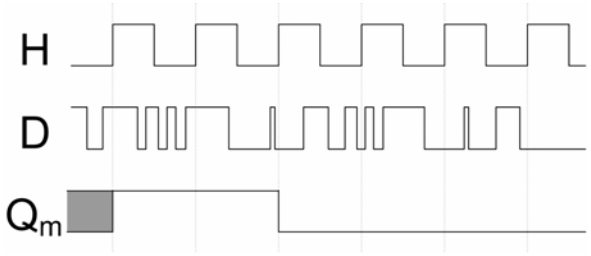


Figure 6.21 Chronogramme pour la bascule D.

Comme on le voit, la sortie Q prend la valeur enregistrée de D lorsque l'horloge H bascule de 0 à 1 (front montant), et cette valeur est conservée tout le temps de la période de l'horloge.

Afin de mieux comprendre le comportement de la bascule D, considérons d'abord la relation entre les entrées et les sorties pour les deux combinaisons possibles de D où H est à 0.

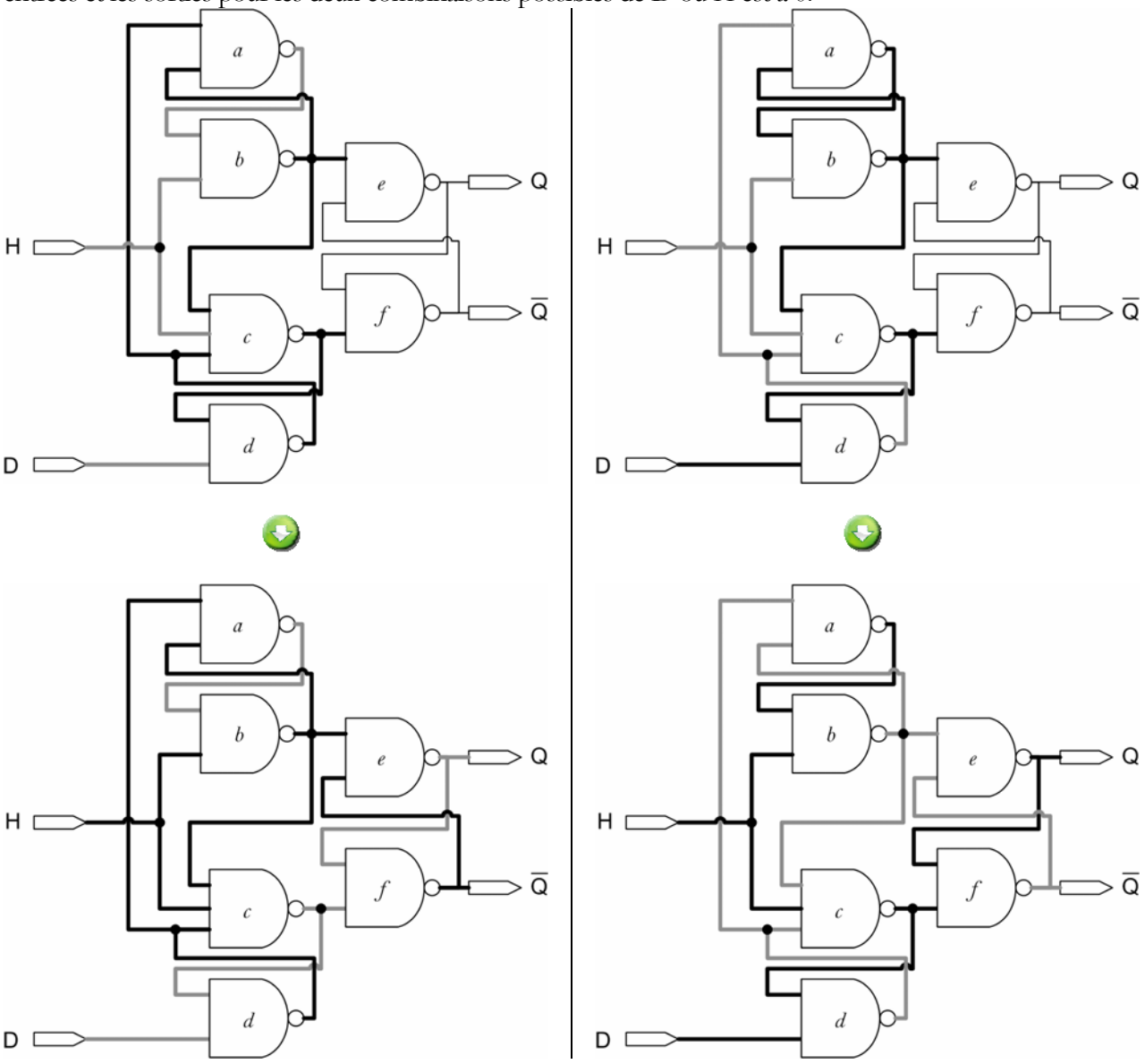


Figure 6.22 Front montant sur l'horloge de la bascule D.

À la première ligne, la bascule est donnée à son état initial. On signale les signaux à 0 par la couleur gris foncé et un trait gras. Un trait gras noir est utilisé pour les signaux à 1. Les sorties des bascules peuvent dans ce cas prendre n'importe quelles valeurs en autant que l'une soit l'inverse de l'autre. Quand le signal H passe à 1 (seconde ligne), la valeur de D est reportée sur Q, son inverse sur la seconde sortie.

En ce qui a trait au cas de gauche, le passage de H de 0 à 1 se traduit par le changement de la sortie de la porte *c* de 1 vers 0, la porte *f* trouvant l'une de ses entrée à 0, sa sortie est forcée à 1, ce qui impose un 0 à la sortie de la porte *e* (la sortie Q). Pour le cas de droite, le passage de H de 0 à 1 se traduit par le changement de la sortie de la porte *b* de 1 vers 0, la porte *e* trouvant l'une de ses entrée à 0, sa sortie est forcée à 1 (la sortie Q), ce qui impose un 0 à la sortie de la porte *e*.

Le chronogramme de la figure 6.23 présente le détail de ces opérations et illustre également la stabilité de la bascule sur les fronts descendants et les variations de l'entrée D :

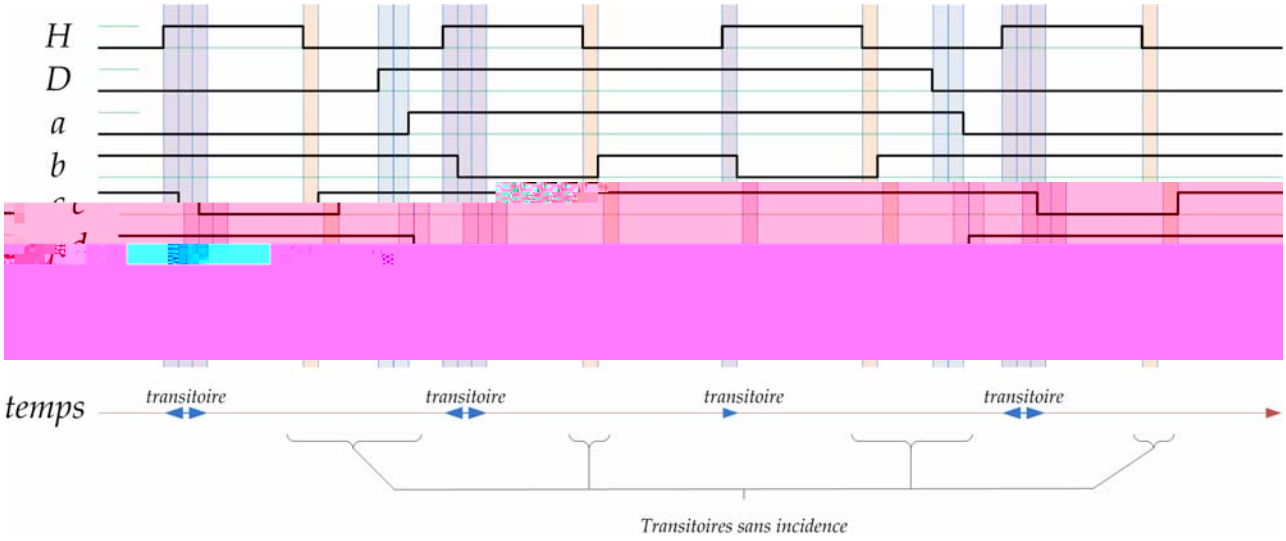
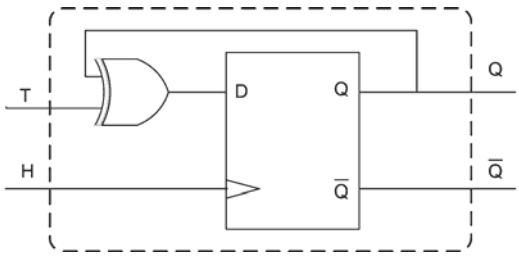


Figure 6.23 Chronogramme détaillé de la bascule D.

6.2.8 Bascule T

D'autres types de bascules existent. Ces bascules sont utiles pour la conception de certains circuits séquentiels, comme nous aurons le loisir de le constater plus bas. La première de ces bascules que nous aurons à considérer ici est la bascule T (T pour *toggle*). La bascule T est une bascule D à laquelle on ajoute une rétroaction depuis la sortie Q vers un XOR :



Circuit de la bascule T

H	T	Q^+
0	-	Q
1	-	Q
↑	0	Q
↑	1	$\overline{Q}$

Table de vérité rattachée

Figure 6.24 Bascule T, implémentation et table de vérité.

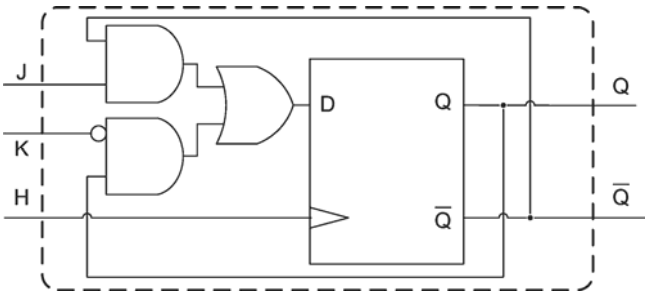
Comme l'indique la table de vérité associée à la bascule T, l'entrée T permet de changer ou de conserver la valeur mémorisée par la bascule. Si T vaut 0, la valeur Q est maintenue au front montant. Si l'entrée T vaut 1, la valeur de Q est inversée au front montant.

6.2.9 Bascule JK

La bascule JK est une bascule relativement complexe. Contrairement aux bascules D et T, la bascule JK possède deux entrées. Son fonctionnement est à mi-chemin entre celui de la bistable SR et de la bascule T.

La bascule JK possède un comportement proche de celui de la bascule T en cela qu'elle permet de conserver ou d'inverser la valeur de Q. Pour ce faire, il faut garder les deux entrées JK à la même valeur. Si cette valeur est 0, la valeur de Q est conservée au front montant. Si les entrées JK sont fixées à 1, l'entrée est inversée au front montant.

La bascule KJ possède un comportement proche de celui de la bistable SR. Si on fixe les entrées JK à 10, on peut écrire 1 dans Q au front montant. À l'inverse, si on fixe les entrées JK à 01, on peut écrire 0 dans Q. Tout cela est résumé dans la table de vérité suivante, et le circuit correspondant est présenté :



Circuit de la bascule JK

H	J	K	Q^+
0	-	-	Q
1	-	-	Q
↑	0	0	Q
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q}$

Table de vérité rattachée

Figure 6.25 Bascule T, implémentation et table de vérité.

Une façon simple de prédire le comportement de la bascule JK est d'écrire son équation :

$$Q^+ = \overline{Q}\,J + Q\,\overline{K}$$

Ce qui peut s'exprimer ainsi : Si Q vaut 0, la sortie prendra la valeur de J au front montant ; si Q vaut 1, la sortie prendra la valeur de l'inverse de K au front montant.

6.3 Machine à états finis

Une famille importante de circuits séquentiels auxquels nous allons nous intéresser est celle des machines à états finis. En dehors de leur implémentation par circuits logiques, les machines à états forment un modèle théorique avec ses subtilités qu'il importe de connaître.

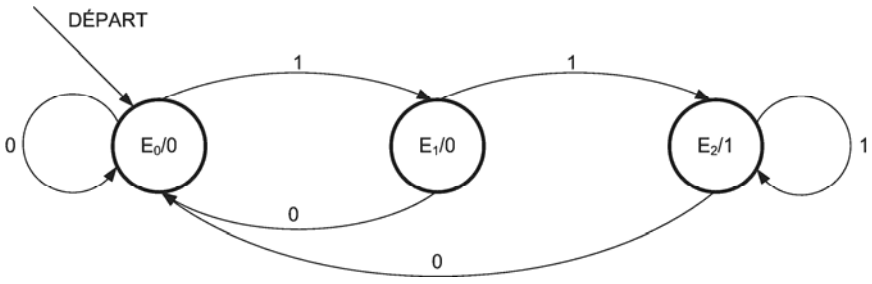
6.3.1 Définition

Une machine à état finis est une machine séquentielle algorithmique caractérisée par un vecteur d'entrées, un vecteur de sorties, et une séquence d'états définissant son comportement. La machine (également appelée automate) va passer d'un état à l'autre suivant les séquences d'entrée qu'elle reçoit. On attribue généralement à la machine un état de départ lui permettant de débuter son fonctionnement à partir d'un point fixe.

Les machines à états finis sont généralement représentées par un diagramme des états permettant de visualiser les transitions entre les états selon le patron de stimulation reçu par le vecteur d'entrées. Les états sont alors représentés par des cercles ; le nom rattaché à chaque état à l'intérieur de chaque cercle ; les transitions entre les états sont représentés par des arcs dirigés reliant les cercles ; les conditions (valeurs du vecteur d'entrée) enclenchant ces transitions sont notées sur les arcs ; et finalement la valeur des sorties est

généralement indiquée soit sur l'arc (séparée des entrées par un trait oblique : /) ou à l'intérieur du cercle (séparée du nom de l'état par un trait oblique : /).

Voici un exemple de machine à états finis où les sorties sont indiquées à l'intérieur des cercles :



Voici les éléments que l'on y trouve :

- trois états notés E_0, E_1 et E_2 ;
- un état de départ E_0 ;
- une entrée (notée sur les arcs) ;
- une sortie associé à chacun des trois états, respectivement 0, 0 et 1.

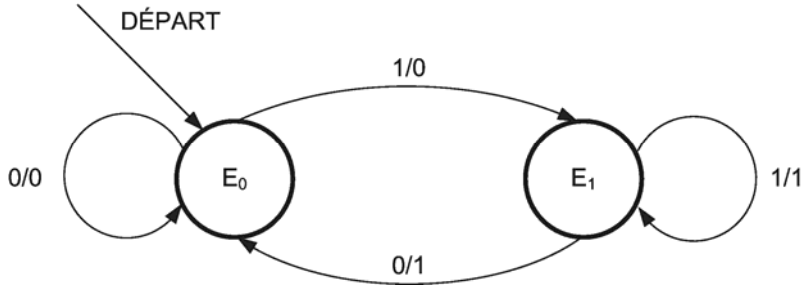
Cette machine permet de détecter une séquence de deux 1 consécutifs en entrée. Considérons en effet son fonctionnement :

- au départ, la machine est à E_0 et sa sortie à 0 ;
- tant que la machine ne reçoit pas 1, elle demeure à E_0 et sa sortie à 0 ;
- si la machine reçoit 1, elle change d'état et part vers E_1 : le début de la séquence 11 est détecté. La sortie demeure à 0 ;
- si à cette étape, la machine reçoit en entrée 0, la séquence est brisée. La machine revient à E_0 et tout reprend depuis le début ;
- si la machine reçoit 1 alors qu'elle se trouve en E_1, elle transite vers E_2 et sa sortie est alors à 1 (pour indiquer qu'elle a reçu la séquence 11) ;
- une fois en E_2, si la machine reçoit 1, elle demeure dans le même état car la séquence (entrée précédente - entrée actuelle) est encore 11 : la sortie est à 1 ;
- autrement elle part en E_0 et tout recommence depuis le début.

Dans la pratique, ce type de diagramme est construit par le concepteur pour définir une machine à états finis. Il importe donc de bien comprendre ses règles et user d'imagination et de créativité pour aboutir à un tel résultat. Le concepteur qui propose cette machine n'avait au départ qu'une indication concise :

Concevoir une machine qui prend une entrée et une sortie. La sortie vaut 1 si la machine reçoit en entrée la séquence 11. Autrement, la sortie vaut 0.

Maintenant considérons le second cas. Voici un exemple de machine à états finis où les sorties sont indiquées sur les arcs :



Voici les éléments que l'on y trouve :

- deux états notés E_0 et E_1 ;
- un état de départ E_0 ;
- une entrée (notée sur les arcs) ;
- des sorties associées aux transitions, 0 pour les transitions de E_0 à E_0 et de E_0 à E_1, 1 pour les transitions de E_1 à E_1 et de E_1 à E_0.

Cette machine (aussi) permet de détecter une séquence de deux 1 consécutifs en entrée. Considérons son fonctionnement :

- Au départ, la machine est à E_0;
- Tant que la machine ne reçoit pas 1, elle demeure à E_0 et sa sortie à 0 ;
- Si la machine reçoit 1, elle change d'état et part vers E_1 : le début de la séquence 11 est détecté. La sortie demeure à 0 mais elle est ici associée à la transition;
- Si à cette étape, la machine reçoit en entrée 0, la séquence est brisée. La machine revient à E_0 et tout reprend depuis le début ;
- Si la machine reçoit 1 et qu'elle se trouve en E_1, elle demeure en E_1 et sa sortie est alors à 1 (pour indiquer qu'elle a reçu la séquence 11). Elle reste à cet état tant et aussi longtemps que l'entrée est 1;
- Autrement elle part en E_0 et tout recommence depuis le début.

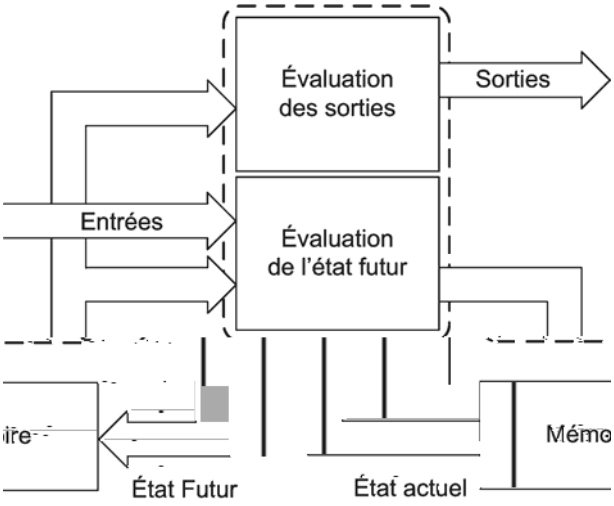
Comme on le voit, dans sa conception, cette machine est différente de la première, mais dans son fonctionnement global, elle lui est identique.

6.3.2 Machine de Moore et Mealy

Nous avons pu voir que lors de la représentation d'une machine à états finis par un diagramme, il est possible d'indiquer les sorties sur les arcs ou à l'intérieur de l'état. Cette distinction n'est pas purement graphique. Comme on a pu s'en rendre compte en réalisant deux machines selon les deux conventions, les machines sont différentes.

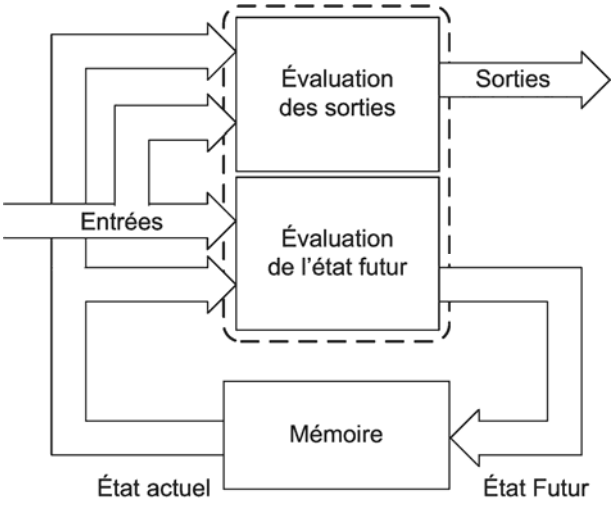
Cette différence réside dans le fait que dans le premier cas, la sortie dépend uniquement de l'état, tandis que dans le second cas, elle dépend de l'état et de l'entrée.

Lorsque la sortie dépend uniquement de l'état courant, on dit que la machine est une machine de Moore. Le nom de machine de Moore fait référence au professeur Edward F. Moore de l'université de Winsconsin-Madison aux États-unis qui en est l'inventeur.



Modèle de machine de Moore

Lorsque la sortie dépend de l'état courant et de l'entrée, on dit que la machine est une machine de Mealy, en référence à G.H. Mealy qui les fit connaître dans un article au Bell System Tech en 1955.



Modèle de machine de Mealy

6.3.3 Convertir une Machine de Moore en Mealy

Étant donné la flexibilité des machines de Mealy par rapport aux machines de Moore qui n'évaluent les sorties qu'en fonction des états, ces premières s'avèrent souvent plus économiques à implémenter en circuits logiques. Mais cette économie se paie souvent au prix d'une difficulté de conception. Les machines de Mealy sont donc plus économiques que les machines de Moore, tandis que les machines de Moore sont plus simples à concevoir.

Il est cependant facile de passer d'une machine de Moore à une machine de Mealy en respectant quelques règles simples :

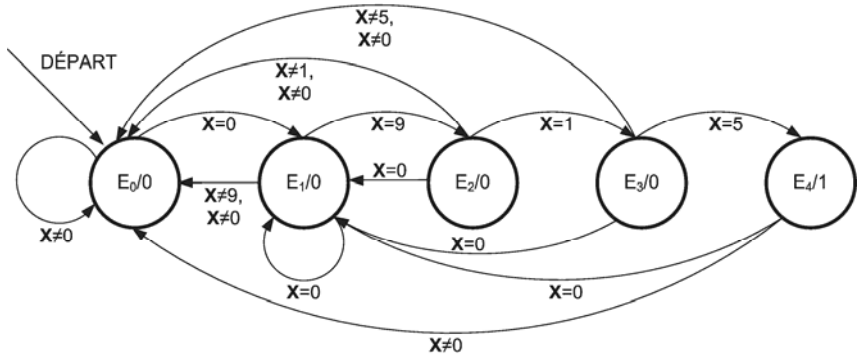
1. Reporter les sorties sur les arcs arrivant vers cet état
2. Simplifier le circuit si la simplification est possible

Considérons pour cela l'exemple suivant :

Exemple :

On veut concevoir le diagramme d'états d'un système d'ouverture de porte avec code d'accès. La machine reçoit à son entrée **X** une série de chiffres tapée sur un clavier numérique. Si la machine reçoit la bonne séquence de chiffres (0,9,1,5) la porte est ouverte grâce au signal de sortie.

On va d'abord concevoir le diagramme selon une forme de machine de Moore :



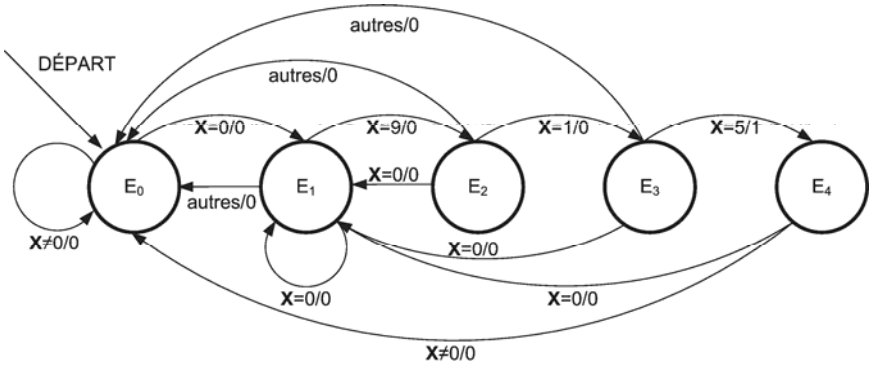
Ça a l'air un peu complexe, mais il suffit de suivre les états pour bien comprendre. Reprenons l'analyse :

- cinq états notés E_0, E_1, E_2, E_3 et E_4 ;
- un état de départ E_0 ;
- un vecteur d'entrée **X** ;
- les sorties associées aux états : 0 pour les transitions de E_0 à E_3 ; 1 pour E_4.

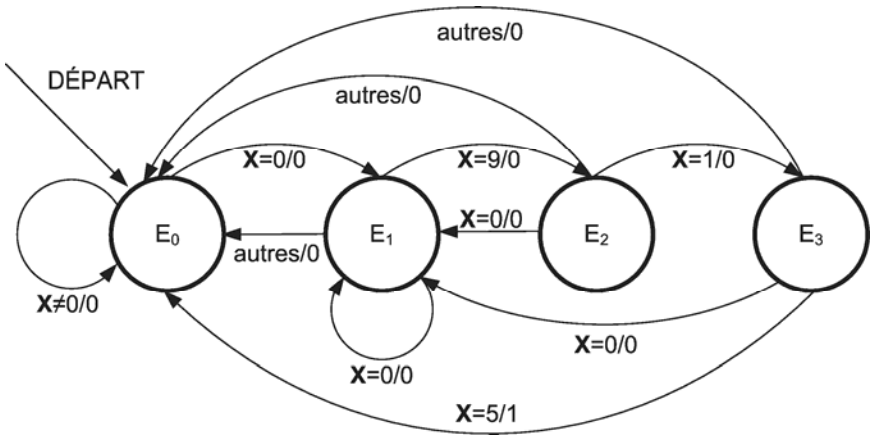
Considérons son fonctionnement :

- tant que la bonne séquence est donnée en entrée, la machine va de E_0 à E_4 ;
- à chaque état, si l'entrée est 0, on revient à E_1 ;
- Autrement, la machine revient à son état initial E_0

C'est tout simple finalement ! Transformons maintenant cette machine en une machine de Mealy :



Tous les arcs ont pour sortie 0, sauf celui de la transition de E_3 à E_4 où l'arc porte une sortie qui vaut 1. Est-ce qu'il y a moyen de simplifier cette machine ? Affirmatif. Il existe une méthode systématique qui sera expliquée plus en détails plus bas. En attendant, regardant cette solution :



Comme on le voit, cette nouvelle version permet d'économiser un état. On verra plus bas que cette économie d'état peut se traduire en économie matérielle en termes de composants électroniques. Nous allons maintenant considérer les différentes façons d'implémenter de telles machines, et approfondirons alors les techniques rattachées à leur conception…

6.4 Circuits séquentiels synchrones

Nous allons considérer l'implémenter matérielle des machines à états finis dites circuits séquentiels synchrones. Les circuits séquentiels synchrones se différencient des circuits séquentiels asynchrones que nous avons pu voir au début du chapitre en cela que les mémoires utilisées sont synchrones, synchronisées pas un signal d'horloge . On se rappellera à cet effet que les bascules (D maître esclave, D, T ou JK) sont des mémoires qui enregistrent une valeur à un instant précis (front montant ou front descendant de l'horloge). Ce type de mémoires sera mis en œuvre dans cette section.

6.4.1 Conception d'une machine de Moore

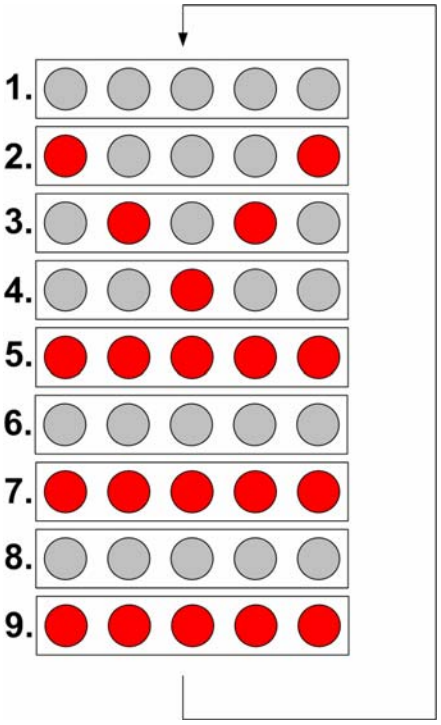
Les étapes à suivre pour concevoir une machine à états finis selon le modèle de la machine de Moore à partir d'un cahier des charges sont :

1. Dessiner le diagramme des états
2. Poser la table des états
3. Définir la table de transition
4. Déterminer les expressions des entrées des bascules
5. Déterminer les expressions des sorties
6. Faire le schéma

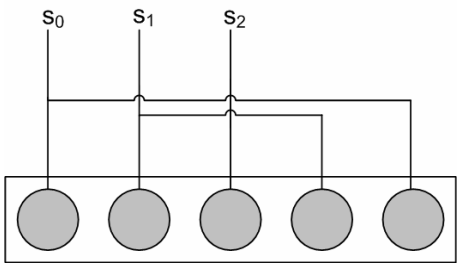
Pour illustrer ce procédé, nous allons considérer un exemple :

Exemple :

Le jeune étudiant de Polytechnique, Jean Némard, fraîchement sorti de sa session et de son cours de circuits logiques, pense à mettre en pratique ses compétences pour amuser des amis dans son *mega party* de fin de session. Jean Némard achète 5 lampes de couleur avec lesquelles il veut réaliser un effet Disco. Il pose les cinq lampes les unes à coté des autres, et veut obtenir le résultat suivant :

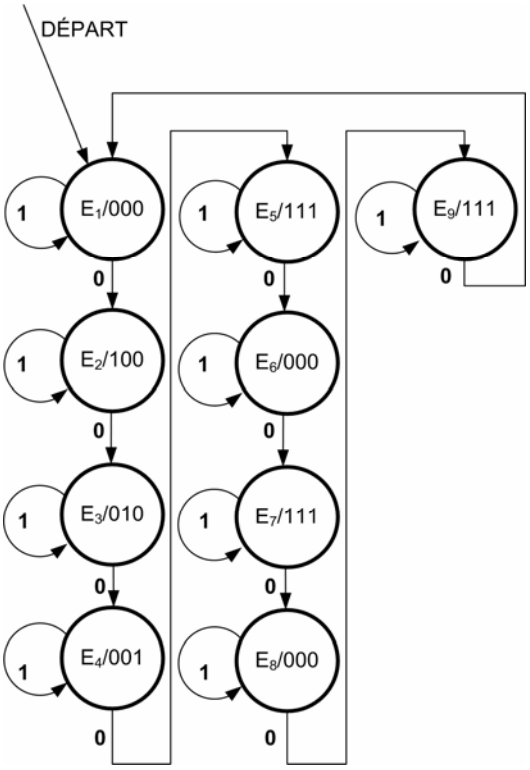


En plus de cette séquence, Jean Némard ajoute un bouton à son système qui lui permet d'arrêter le système. Lorsque Jean appuie sur le bouton, les lampes se figent au dernier état rencontré. Finalement, il se dit que son système peut utiliser 3 sorties seulement étant donnée la façon de les allumer. Il dessine le schéma suivant :



Jean Némard décide de suivre scrupuleusement la méthode enseignée dans son cours pour concevoir une machine séquentielle de Moore.

1. Dessiner le diagramme des états



2. Poser la table des états

La table des états consiste à traduire le diagramme des états présenté ci-contre :

État actuel	État futur		Sorties
	Entrée (b)		
	0	1	$(s_0s_1s_2)$
E_1	E_2	E_1	000
E_2	E_3	E_2	100
E_3	E_4	E_3	010
E_4	E_5	E_4	001
E_5	E_6	E_5	111
E_6	E_7	E_6	000
E_7	E_8	E_7	111
E_8	E_9	E_8	000
E_9	E_1	E_9	111

Il y a trois colonnes principales. La première correspond à l'état initial ; la seconde à l'état futur, et elle est subdivisée en colonnes selon les valeurs des entrées ; la troisième correspond à la sortie — comme nous avons affaire à une machine de Moore, la sortie de dépend que de l'état actuel.

3. Définir le table de transition

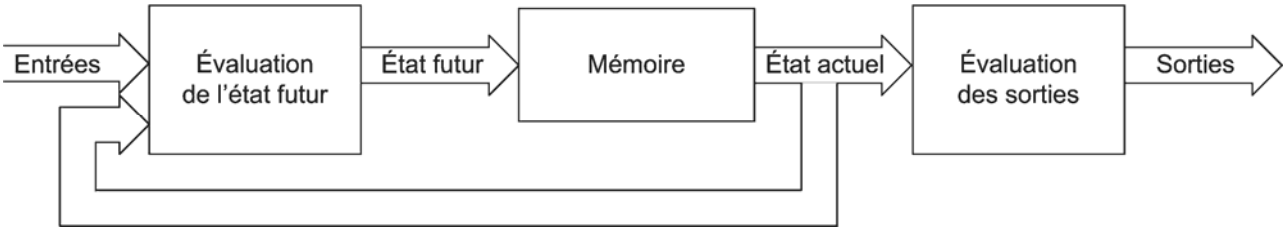
Le table de transition est l'équivalent de la table des états, à cette différence près que les états sont encodés sous une forme binaire. Dans le cas qui nous concerne, nous avons 9 états. Il nous faut donc encoder ces états avec 4 bits au minimum, que l'on notera $e_3e_2e_1e_0$.

Voici l'encodage code choisi :

État	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9
Code : $e_3e_2e_1e_0$	0000	0001	0010	0011	0100	0101	0110	0111	1000

État actuel $(e_3e_2e_1e_0)$	État futur $(e_3^+e_2^+e_1^+e_0^+)$		Sorties
	Entrée (b)		
	0	1	$(s_0s_1s_2)$
0000	0001	0000	000
0001	0010	0001	100
0010	0011	0010	010
0011	0100	0011	001
0100	0101	0100	111
0101	0110	0101	000
0110	0111	0110	111
0111	1000	0111	000
1000	0000	1000	111

Cette table va nous permettre de procéder aux deux étapes suivantes. Mais avant de continuer, considérons le schéma suivant: :

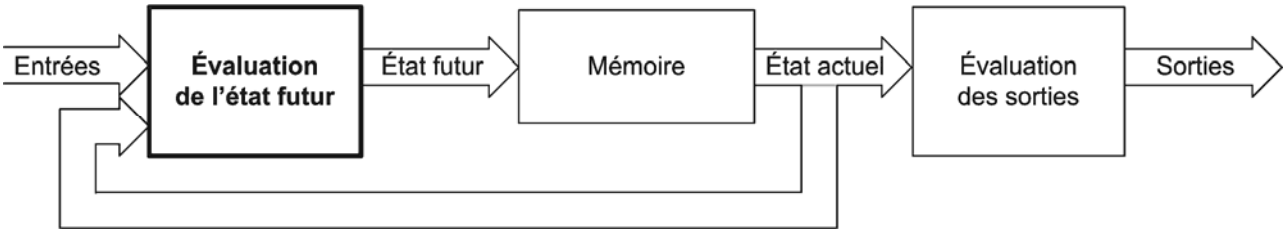


Ce schéma est exactement le même que celui que nous avons vu précédemment, mais légèrement modifié pour mettre en évidence les circuits que nous allons implémenter.

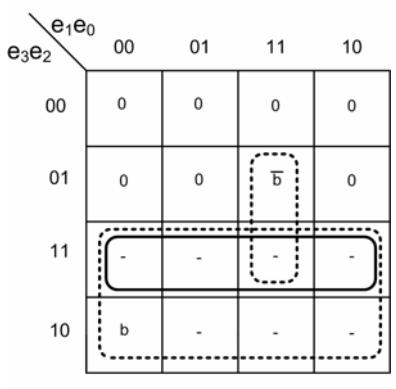
Pour ce qui est de la mémoire, nous avons 4 bits à enregistrer ($e_3e_2e_1e_0$), nous allons par conséquent utiliser quatre bascules D.

4. Déterminer les expressions des entrées des bascules

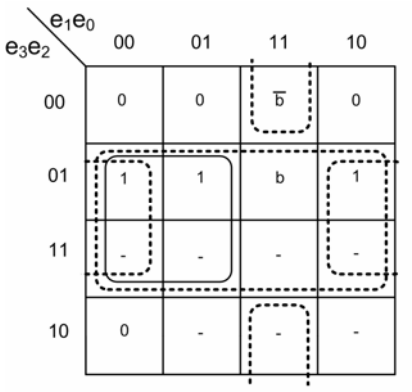
Cette étape consiste à trouver le circuit combinatoire permettant d'évaluer l'état futur en fonction de l'état actuel et des entrées :



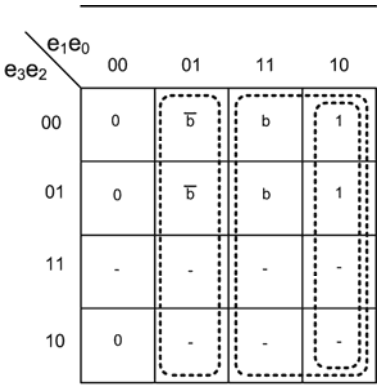
Nous avons donc 5 entrées $e_3e_2e_1e_0$ et b pour le bouton et 4 sorties $e_3^+e_2^+e_1^+e_0^+$:



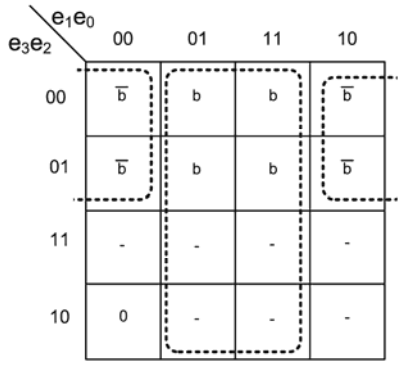
Karnaugh pour e_3^+



Karnaugh pour e_2^+



Karnaugh pour e_1^+



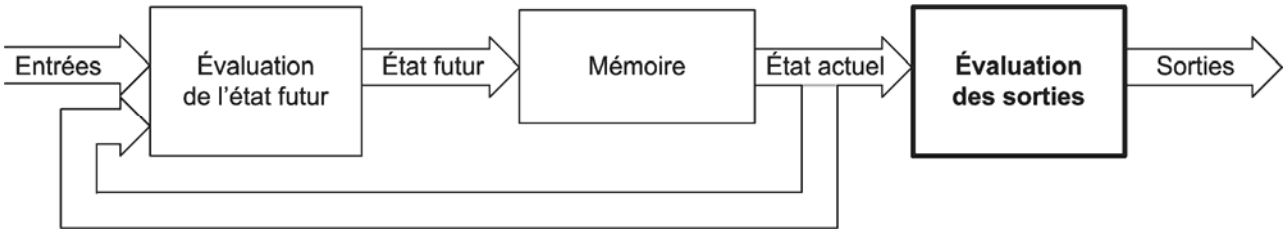
Karnaugh pour e_0^+

Ce qui nous permet d'écrire :

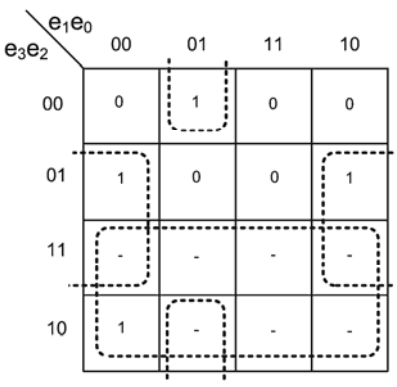
- $e_3^+ = be_3 + \overline{b}\, e_2e_1e_0 + e_3e_2$
- $e_2^+ = be_2 + \overline{b}\ \overline{e_2}\ e_1e_0 + e_2\overline{e_1} + e_2\overline{e_0}$
- $e_1^+ = \overline{b}\ \overline{e_1}\ e_0 + be_1 + e_1\overline{e_0} = \overline{b}\,(e_0 \oplus e_1) + be_1$
- $e_0^+ = \overline{b}\ \overline{e_3}\ \overline{e_0} + be_0$

5. Déterminer les expressions des sorties

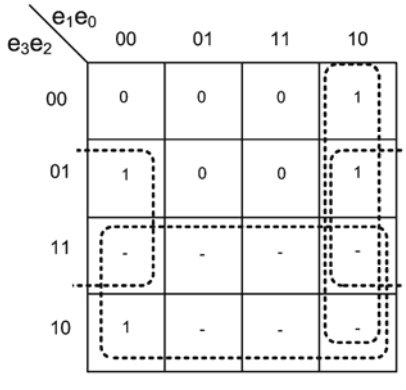
Cette étape consiste à trouver le circuit combinatoire permettant d'évaluer les sorties en fonction de l'état actuel :



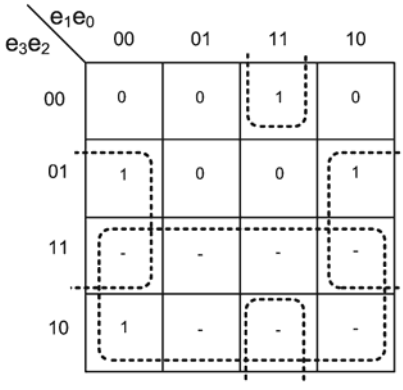
Nous avons donc 4 entrées $e_3e_2e_1e_0$ et 3 sorties $s_0s_1s_2$:



Karnaugh pour s_0



Karnaugh pour s_1



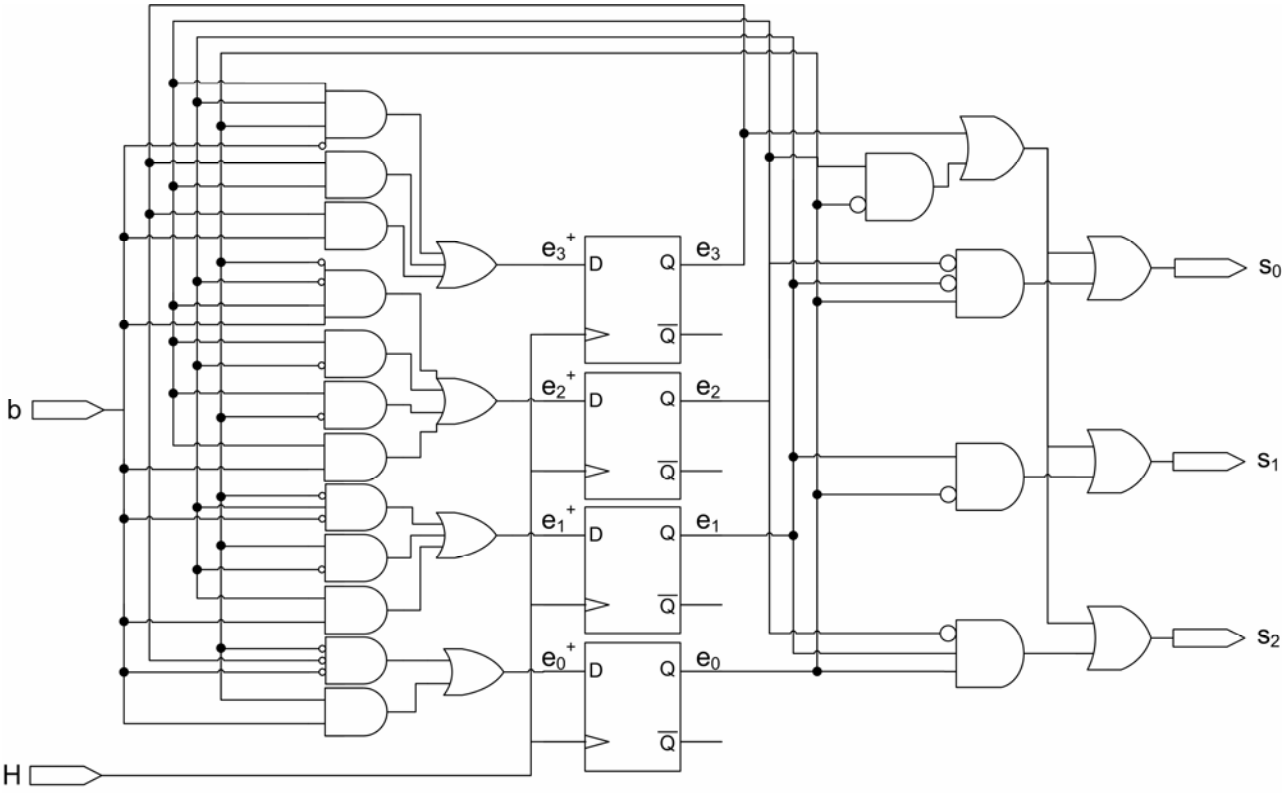
Karnaugh pour s_2

Ce qui nous permet d'écrire :

- $s_0 = e_3 + e_2 \overline{e_0} + \overline{e_2}\ \overline{e_1}\ e_0$
- $s_1 = e_3 + e_2 \overline{e_0} + e_1 \overline{e_0}$
- $s_2 = e_3 + e_2 \overline{e_0} + \overline{e_2}\ e_1 e_0$

6. Faire le schéma

L'étape finale ! Il reste plus qu'à dessiner :



Jean Némard est maintenant prêt à faire son *mega party*.

6.4.2 Conception d'une machine de Mealy

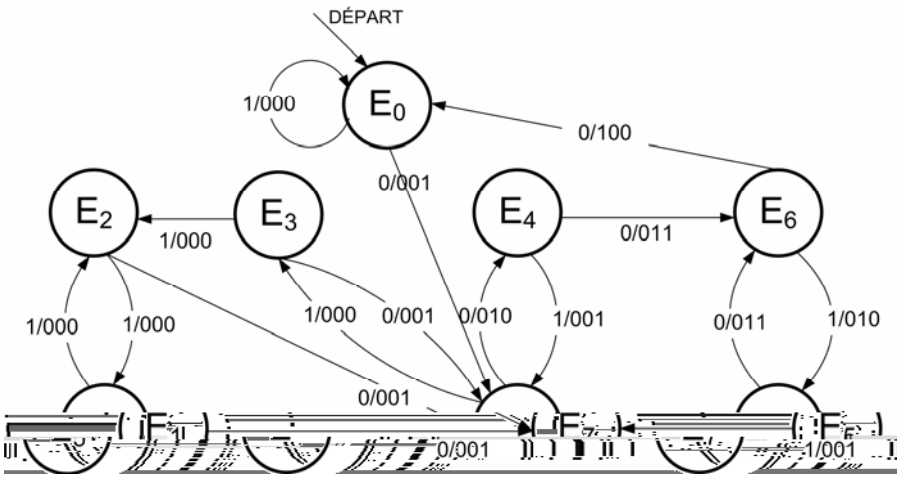
Les étapes à suivre pour concevoir une machine à états finis selon le modèle de la machine de Mealy est sensiblement identique à celui de la machine de Moore. Pour illustrer le procédé, nous allons considérer un exemple :

Exemple :

On veut concevoir une machine de Mealy qui reçoit une entrée x et qui affiche à sa sortie ($s_2s_1s_0$) le nombre de fois qu'un 0 apparaît à cette entrée. Le compte est décrémenté à chaque fois qu'un 1 apparaît à l'entrée. Si une séquence de quatre 0 apparaît à l'entrée, le compte est remis à zéro. Un étudiant jeune et maladroit propose une solution non optimale que nous allons explorer. Cette conception maladroite sera améliorée plus tard, nous permettant du même coup d'introduire les outils d'optimisation…

1. Dessiner le diagramme des états

Voici le diagramme des états proposé par notre étudiant :



2. Poser la table des états

Comme nous avons affaire à une machine de Mealy, les sorties dépendent des entrées. La colonne de sorties est donc subdivisée selon les valeurs des entrées :

État actuel	État futur		Sorties ($s_2s_1s_0$)	
	Entrée (x)		Entrée (x)	
	0	1	0	1
E_0	E_7	E_0	001	000
E_1	E_7	E_2	001	000
E_2	E_7	E_1	001	000
E_3	E_7	E_2	001	000
E_4	E_6	E_7	011	001
E_5	E_6	E_7	011	001
E_6	E_0	E_5	100	010
E_7	E_4	E_3	010	000

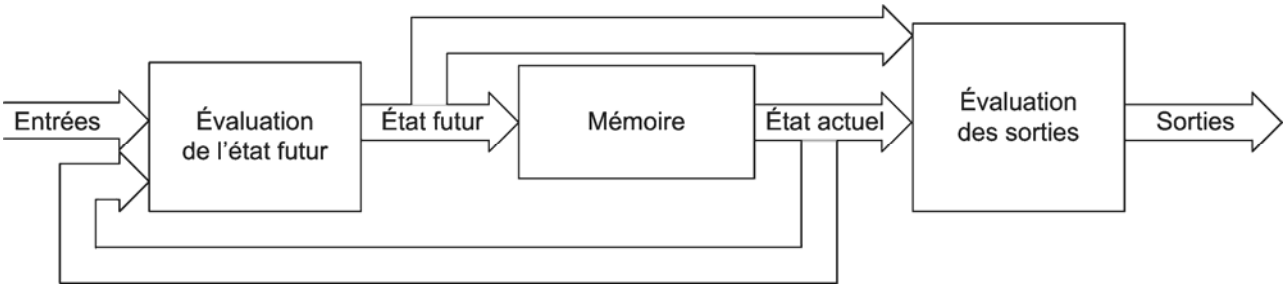
3. Définir la table de transition

Dans le cas qui nous concerne, nous avons 8 états. Il nous faut donc encoder ces états avec 3 bits, que l'on notera $e_2e_1e_0$. Voici l'encodage code choisi :

État	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7
Code : $e_2e_1e_0$	000	001	010	011	100	101	110	111

État actuel $e_2e_1e_0$	État futur ($e_2^+e_1^+e_0^+$)		Sorties ($s_2s_1s_0$)	
	Entrée (x)		Entrée (x)	
	0	1	0	1
000	111	000	001	000
001	111	010	001	000
010	111	001	001	000
011	111	010	001	000
100	110	111	011	001
101	110	111	011	001
110	000	101	100	010
111	100	101	010	000

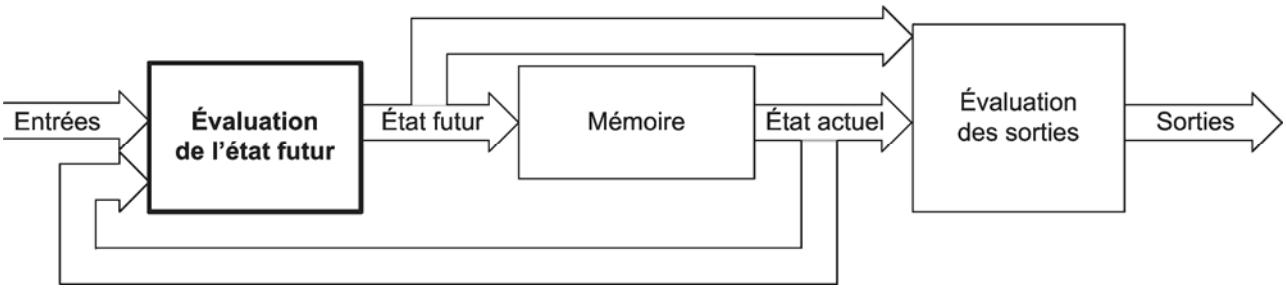
Considérons le schéma suivant :



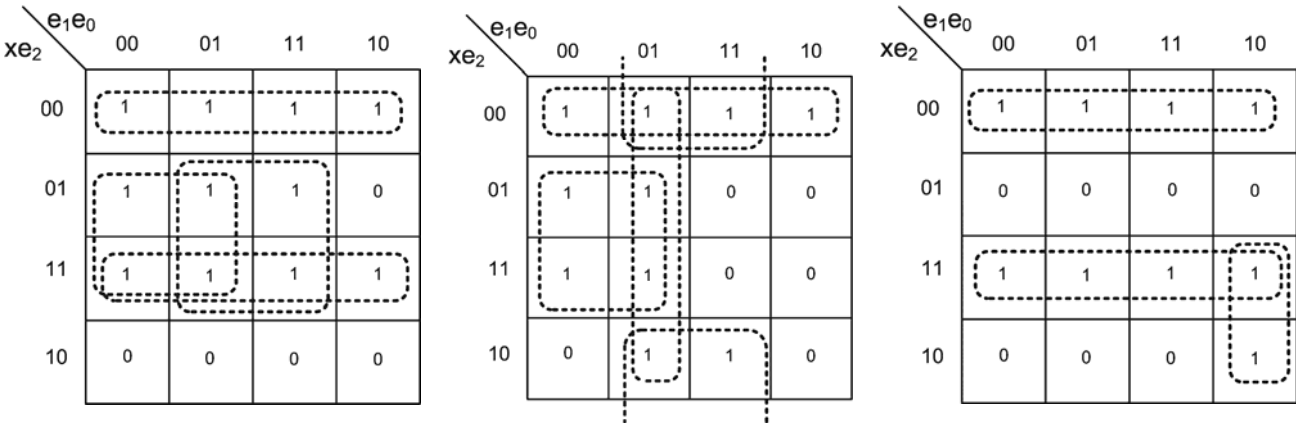
Ce schéma est exactement le même que celui que nous avons vu précédemment, mais légèrement modifié pour mettre en évidence les circuits que nous allons implémenter.

4. Déterminer les expressions des entrées des bascules

Cette étape consiste à trouver le circuit combinatoire permettant d'évaluer l'état futur en fonction de l'état actuel et des entrées :



Nous avons donc 4 entrées $e_2e_1e_0$ et x pour l'entrée, ainsi que 3 sorties $e_2^+e_1^+e_0^+$:



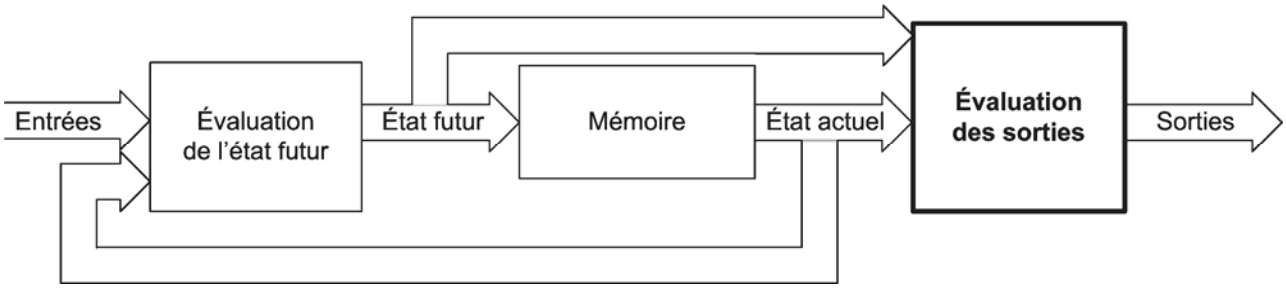
Karnaugh pour e_2^+ **Karnaugh pour e_1^+** **Karnaugh pour e_0^+**

Ce qui nous permet d'écrire :

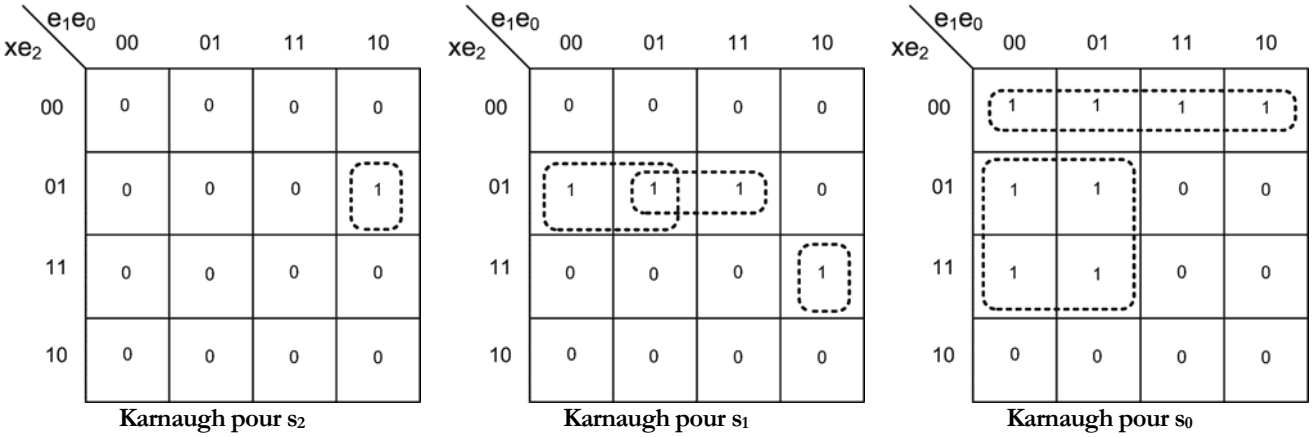
- $e_2^+ = \overline{x}\,\overline{e_2} + xe_2 + e_2\overline{e_1} + e_2e_0$
- $e_1^+ = \overline{x}\,\overline{e_2} + \overline{e_1}\,e_0 + e_2\overline{e_1} + \overline{e_2}\,e_0$
- $e_0^+ = \overline{x}\,\overline{e_2} + xe_2 + xe_1\overline{e_0}$

5. Déterminer les expressions des sorties

Cette étape consiste à trouver le circuit combinatoire permettant d'évaluer les sorties en fonction de l'état actuel et de l'entrée :



Nous avons donc 4 entrées $e_2e_1e_0$ et l'entrée x, et trois sorties $s_2s_1s_0$:

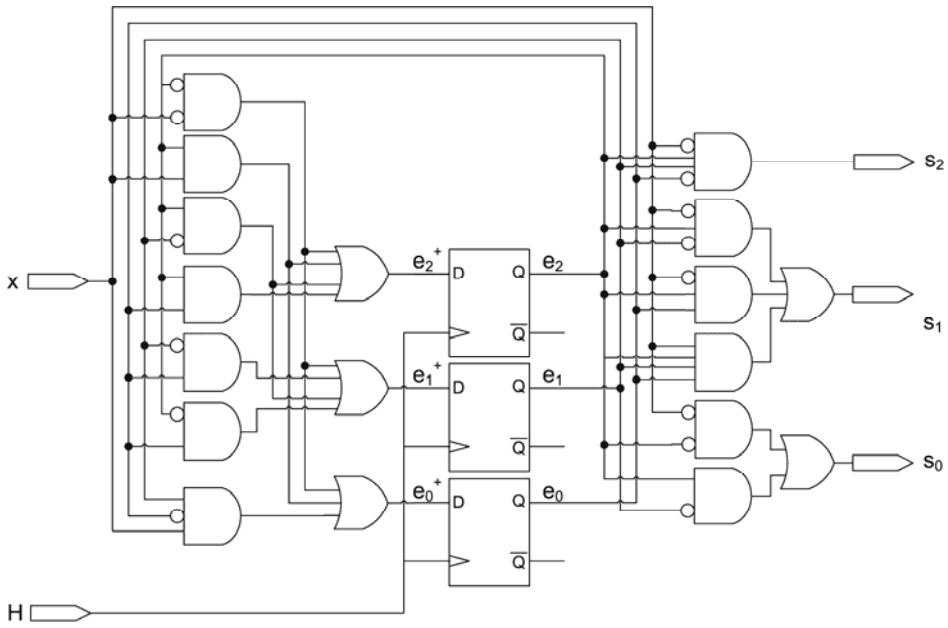


Karnaugh pour s_2 **Karnaugh pour s_1** **Karnaugh pour s_0**

Ce qui nous permet d'écrire :

- $s_2 = \overline{x}\, e_2 e_1\, \overline{e_0}$
- $s_1 = \overline{x}\, e_2\, \overline{e_1} + \overline{x}\, e_2 e_0 + x e_2 e_1 e_0$
- $s_0 = \overline{x}\, \overline{e_2} + e_2\, \overline{e_1}$

6. Faire le schéma :



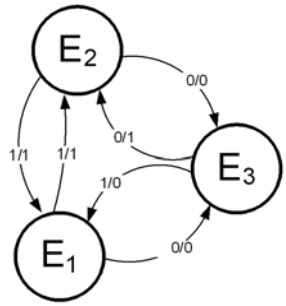
6.4.3 États redondants

Lors de la conception d'une machine à états finis, il arrive souvent que le concepteur crée deux états distincts pour une machine sans que cela soit requis car les deux états de la machines sont strictement l'équivalent l'un de l'autre. Cette erreur est fréquente lors de la conception de machines complexes.

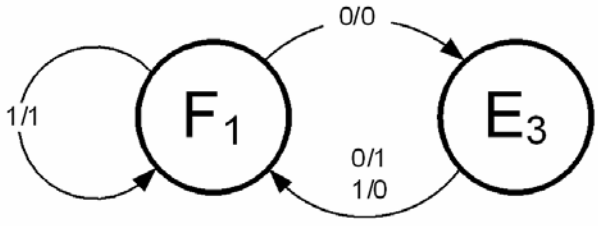
Deux états sont considérés comme strictement équivalent si :

1. Leurs sorties (qu'elles dépendent des entrées ou non) sont identiques.
2. Pour chaque condition dictée par le vecteur d'entrées, les états suivants des deux états considérés sont identiques.

Considérons l'exemple suivant :



Machine à états avec états redondants



Machine équivalente avec F_1 remplaçant E_1-E_2

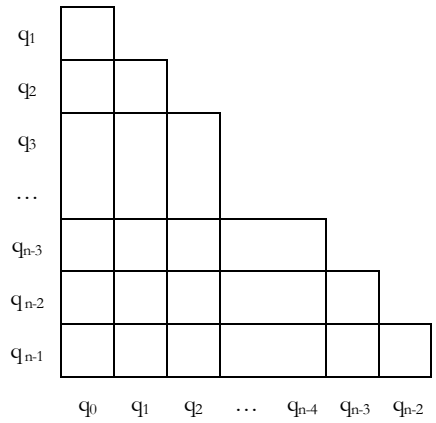
- L'état E_3 ici est différent des deux états E_1 et E_2, car ses sorties conditionnelles (0/1, 1/0) sont différentes de celles de E_1 et E_2 (0/0, 1/1).

- Considérons maintenant Les états E_1 et E_2.
  - Les sorties : Les deux états possèdent les mêmes sorties. La première condition pour leur union est remplie.
  - Les états suivants : Si l'entrée vaut 0, l'état suivant est E_3 dans les deux cas. Si l'entrée vaut 1, l'état suivant est E_1 ou E_2, ce qui revient au même. La deuxième condition est remplie.

⇨ Ces deux états n'en font qu'un, que nous remplaçons par l'état F_1.

Une méthode systématique :

Nous allons présenter une méthode systématique permettant de trouver les états redondants. Il faut pour cela pouvoir comparer deux états deux à deux. La matrice suivante permet de comparer les états deux à deux.



Chaque cellule (q_i,q_j) permet de comparer les deux états concerner. Pour ce faire, on procède comme suit :

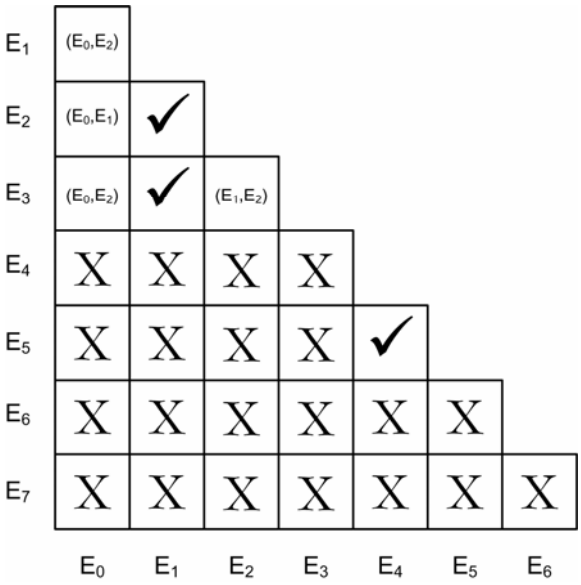
1. Si les deux états de la paire ont des sorties différentes, on place un (X) dans la case ;
2. Si deux états possèdent des états ou des sorties facultatifs non alignés, on place un (X) dans la case ;
3. Pour les cellules qui ne contiennent pas de (X), on note les paires d'états suivants et indiquer toutes les combinaisons possibles. Il faut exclure les paires identiques (ex : (A,A)) et les paires correspondants à la cellule courante ;
4. On place un crochet (✓) dans les cellules vides
5. Pour toutes les cellules qui n'ont pas un crochet ou un X, prendre chaque paire d'état énumérée dans la cellule à l'étape (3) et examiner le contenu des cellules correspondant à ces paires. Si l'une d'elle comprend un (X), placer un (X) dans la cellule courante ;
6. Répéter l'étape (5) tant et aussi longtemps que possible.

Toutes les cellules qui n'ont pas un (X) lorsque ces étapes sont terminées représente une paire d'états équivalents. Pour illustrer cette procédure, nous allons reprendre l'exemple de la machine de Mealy conçue plus haut.

Reprenons ici le tableau d'états correspondant :

État actuel	État futur		Sorties ($s_2s_1s_0$)	
	Entrée (x)		Entrée (x)	
	0	1	0	1
E_0	E_7	E_0	001	000
E_1	E_7	E_2	001	000
E_2	E_7	E_1	001	000
E_3	E_7	E_2	001	000
E_4	E_6	E_7	011	001
E_5	E_6	E_7	011	001
E_6	E_0	E_5	100	010
E_7	E_4	E_3	010	000

On applique les étapes (1) (2) (3) et (4).



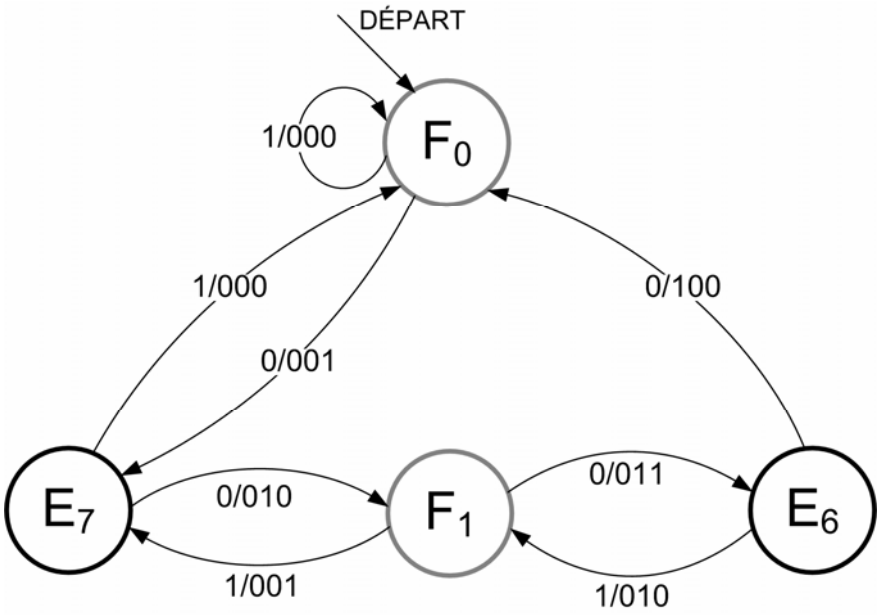
Il reste alors quatre cases contenant une paire d'état. Pour chacune de ces cases, on va appliquer l'étape (5), C'est-à-dire consulter la case correspondante. 1. Les cases E_0E_1 et E_0E_3 renvoient vers E_0E_2. La case E_0E_2 renvoie vers E_0E_1 qui contient un crochet (✓). Aucune case ne renvoie vers une case où il y a un (X). On peut donc s'arrêter là.

Les cases où il n'y a pas de (X) correspondent à des redondant. Voici la liste complète :

$E_0 \equiv E_1$; $E_0 \equiv E_2$;
$E_0 \equiv E_3$; $E_1 \equiv E_3$;
$E_1 \equiv E_3$; $E_2 \equiv E_3$; $E_4 \equiv E_5$.

Ce qui signifie :

- $E_0 \equiv E_1 \equiv E_2 \equiv E_3$ et seront remplacés par F_0 ;
- $E_4 \equiv E_5$ et seront remplacés par F_1 ;



Comme on le voit, cette procédure nous a permis de passer de huit états à quatre, ce qui économise une bascule puisque deux suffisent. Le circuit résultant est présenté ci-dessous. Les étapes de conception sont laissées en exercice. Notons cependant que nous encodons les états ainsi :

État	F_0	E_7	F_1	E_6
Code : e_1e_0	00	01	10	11

6.4.4 Implémentation avec bascule JK et T

L'utilisation de bascules JK ou T peut s'avérer plus économique qu'une implémentation en bascules D. Nous allons montrer comment obtenir les équations de l'étage d'entrée (calcul des états futurs) avec des bascules JK et T. Revenons donc aux équations.

Pour une bascule D, on a :

$$Q^+=D$$

On a (dans le cas d'une bascule T) :

$$Q^+=T\overline{Q}+\overline{T}Q$$

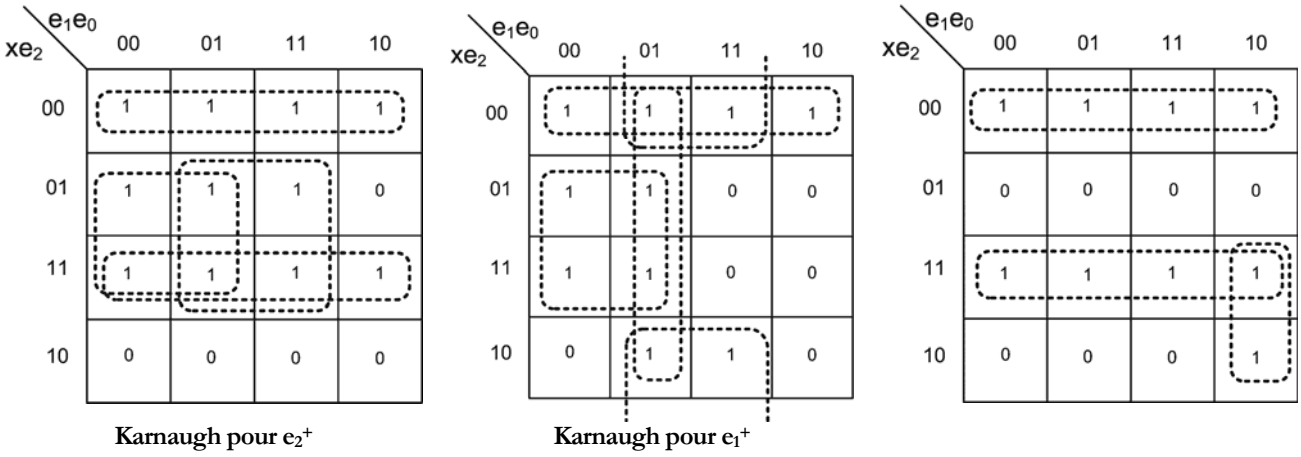
Ainsi, si Q=0, les bascules T et D sont équivalentes :

$$Q=0 \Rightarrow T=D$$

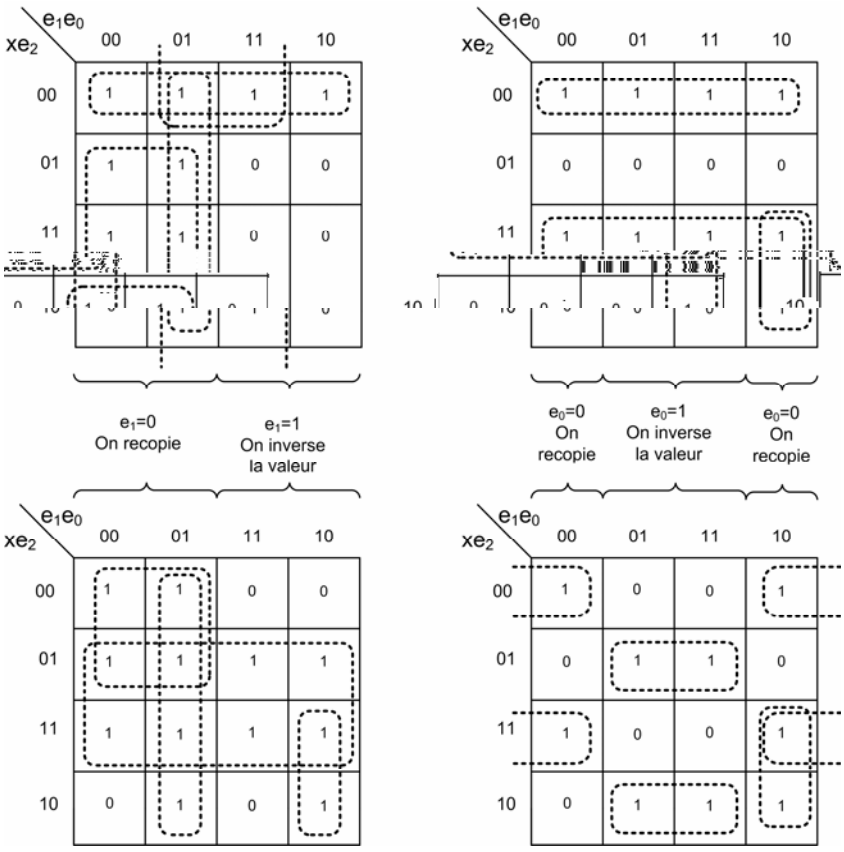
Si Q=1, T est l'inverse de D :

$$Q=1 \Rightarrow T=\overline{D}$$

Ainsi, lorsque l'on est à l'étape (4) de conception d'une machine séquentielle synchrone pour déterminer les expressions des entrées des bascules, il est possible d'utiliser les tables de Karnaugh pour les entrées D et d'inverser la valeur lorsque Q vaut 1. Reprenons l'exemple de la machine de Mealy à huit états. Les tables de Karnaugh obtenues étaient :



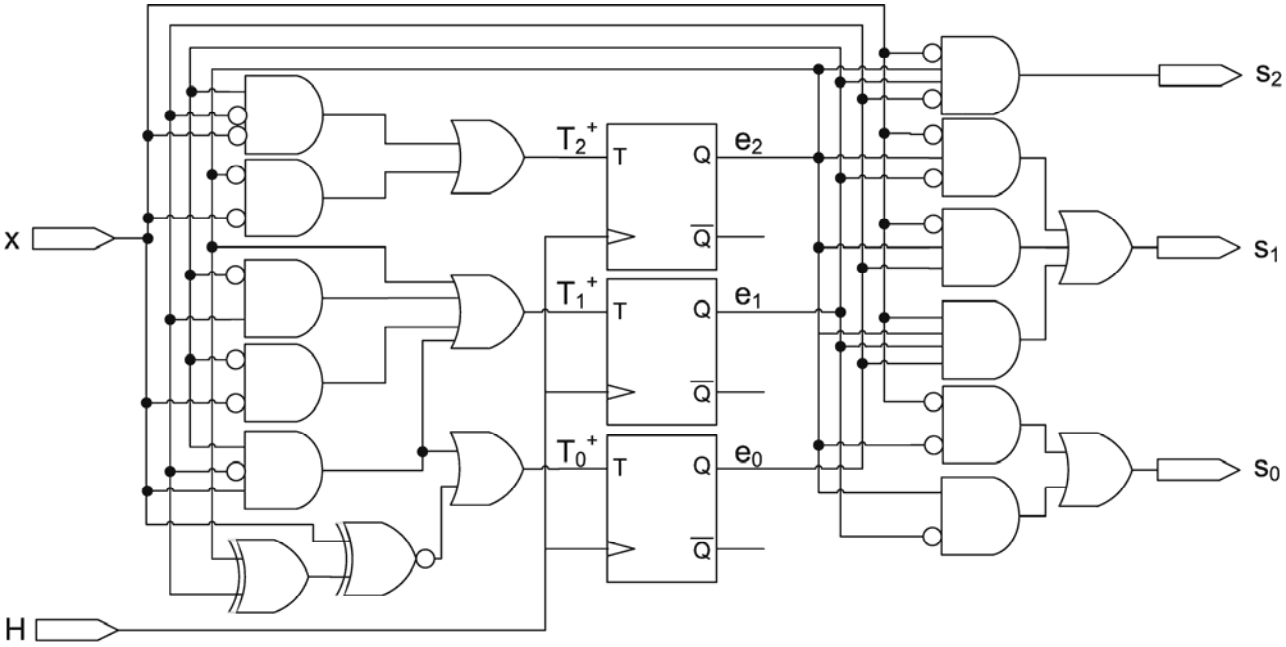
Karnaugh pour e_2^+ **Karnaugh pour e_1^+**



Ce qui nous permet d'écrire :

$T_2^+ = \overline{x}\ \overline{e_2} + \overline{x}\ e_1 \overline{e_0}$ $\quad T_1^+ = \overline{e_1}\ e_0 + \overline{x}\ \overline{e_1} + x e_1 \overline{e_0} + e_2$ $\quad T_0^+ = \left[x \otimes (e_2 \oplus e_0)\right] + x e_1 \overline{e_0}$

Et on trouve :



Dans le cas des bascules JK, le raisonnement est relativement semblable. Reprenons les équations. Pour une bascule D, on a :

$$Q^+=D$$

Pour une bascule JK :

$$Q^+=J\,\overline{Q}+\overline{K}\;Q$$

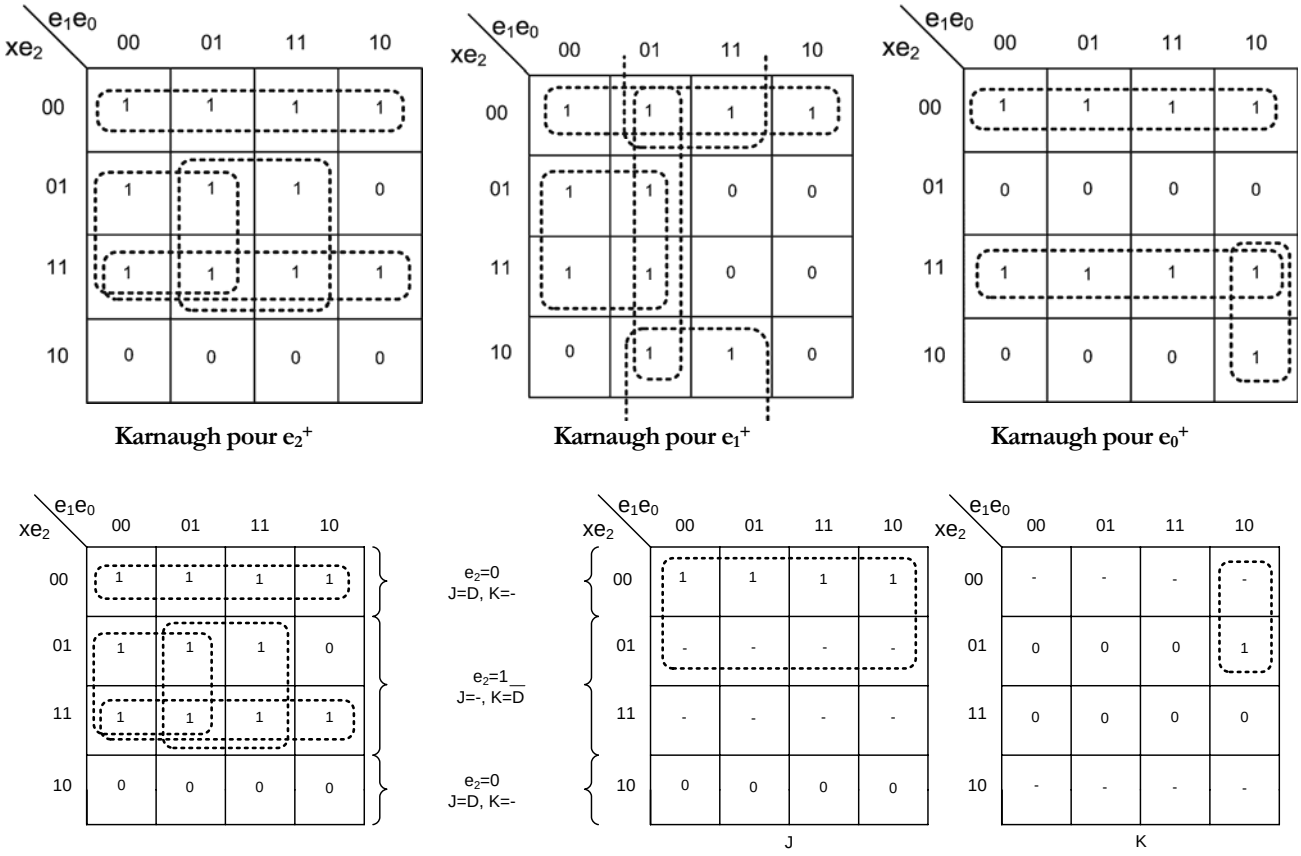
Ainsi, si Q=0, les bascules JK et D sont équivalentes selon l'entrée J (K peut être quelconque) :

$$Q=0 \Rightarrow J=D,\ K=-$$

Si Q=1, K est l'inverse de D (J quelconque) :

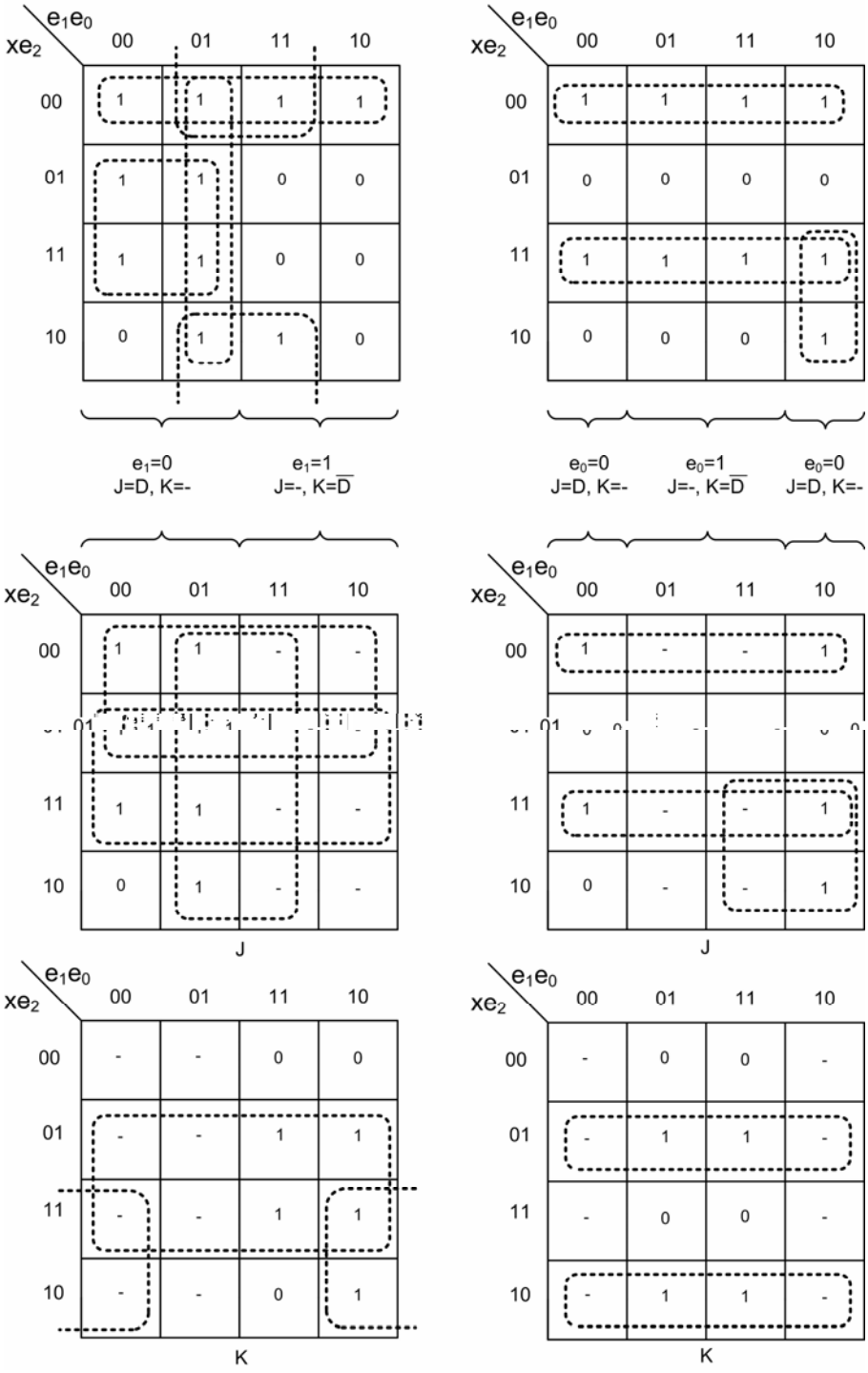
$$Q=1 \Rightarrow K=\overline{D},\ J=-$$

Reprenons encore l'exemple de la machine de Mealy à huit états :



Comme on le voit, il faut cette fois construire deux table de Karnaugh, la première pour J, la seconde pour K. Étant donné la grande présence des cas facultatifs, les équations sont généralement fortement simplifiée :

- $J_2^+=\overline{x}$
- $K_2^+=\overline{x}\,e_1\overline{e_0}$



- $J_1^+ = e_2 + \overline{x} + e_0$
- $K_1^+ = e_2 + x\,\overline{e_0}$
- $J_0^+ = \overline{x \oplus e_2} + xe_1$
- $K_0^+ = x \oplus e_2$

Ce qui nous donne le circuit suivant :