

Finite State Machines (FSM)

dekneuv@unice.fr

(c) E.Dekneuvel 2021

Automates à états finis

A. Catégories de systèmes séquentiels

B. Analyse des FSM

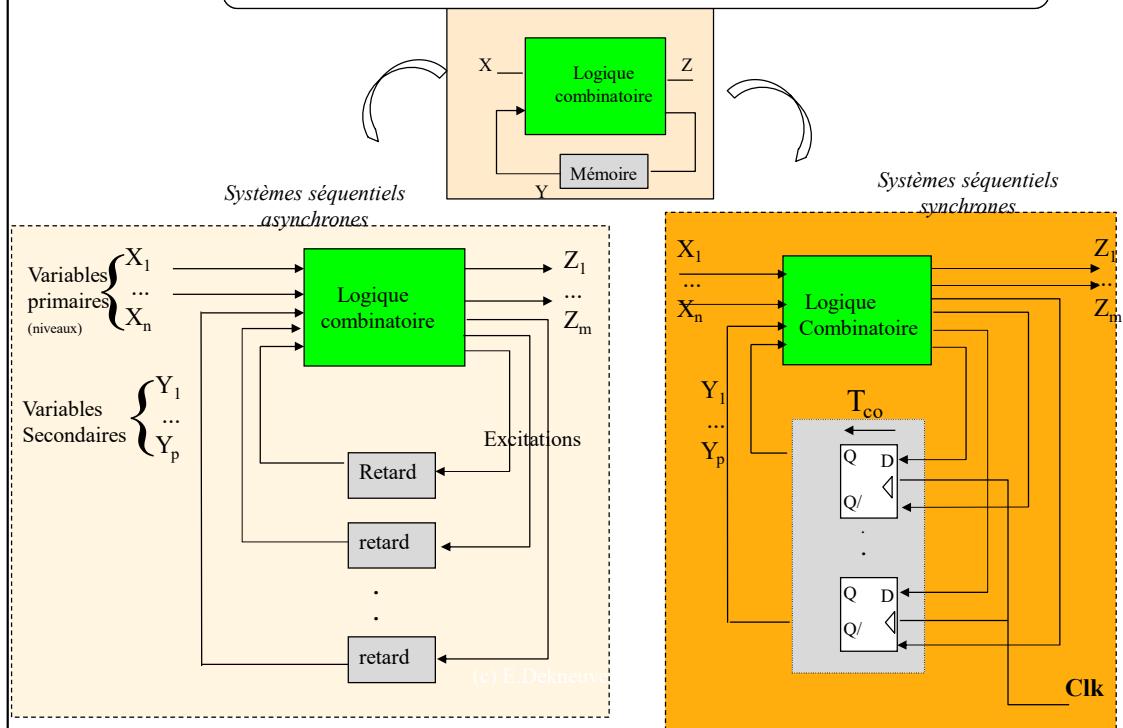
C. Synthèse des FSM

D. Codage des états

E. Réduction des machines séquentielles

(c) E.Dekneuvel 2020

Catégories de systèmes séquentiels



On distingue 2 catégories de systèmes séquentiels:

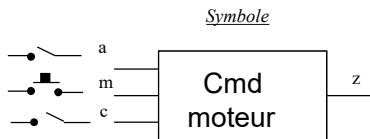
systèmes asynchrones: Toutes les entrées sont considérées de type niveau. La mémoire du système asynchrone peut être modélisée par un retard = temps de propagation des signaux dans la logique combinatoire (sur l'exemple, le système conserve la valeur 1 en mémoire pour 4.8 ns avant de basculer à 0. Si les nouvelles valeurs d'entrée des délais (excitations) sont identiques aux sorties des retards (secondaires/internes), le système est stable. Sinon, on dit qu'il est dans un état transitoire (il continue à évoluer)

Systèmes synchrones: Ils utilisent des éléments mémoires dont le fonctionnement permet de garantir que la bascule ne soit jamais entièrement transparente. La figure montre un échantillon des types de bascules disponibles. La fréquence d'horloge des bascules est fixée par le temps de réaction des bascules et par les circuits de calcul de leur entrées. Dans certaines bascules M/S construite avec un étage de RS en maître, le temps de setup est d'une demi-période d'horloge, ce qui limite le temps de réaction de la combinatoire et donc les performances globales du circuit.



Système séquentiel asynchrone

Cahier des charges



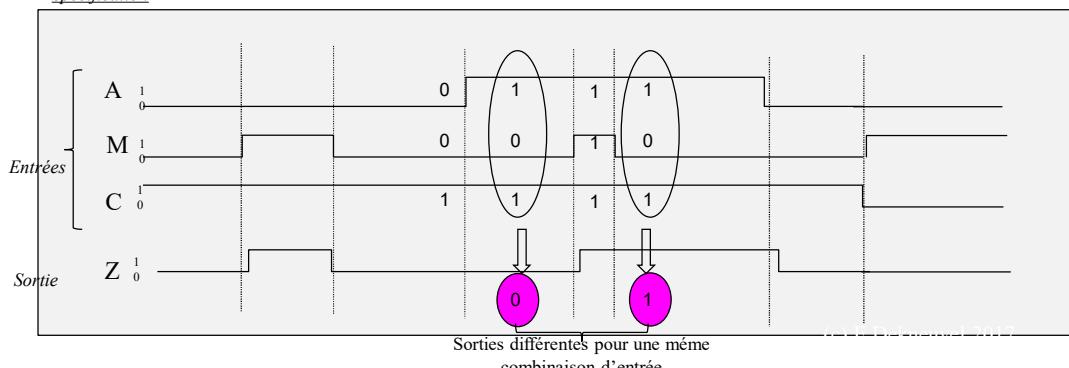
Un circuit de commande un moteur à l'aide de sortie Z. 3 entrées a,m,c permettent de commander l'évolution du système uniquement à partir de leurs niveaux.

C est un interrupteur de sécurité à deux positions. C=0 verrouille le mécanisme.

A est un interrupteur précisant le mode de fonctionnement (intermittent si $a=0$ ou continu si $a=1$)

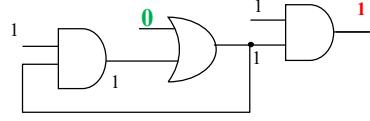
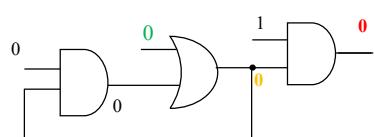
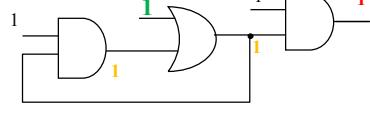
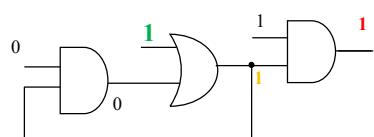
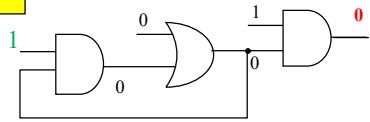
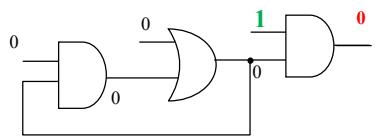
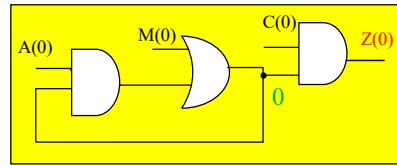
M est un bouton poussoir permettant de commander la mise en route du moteur.

Spécification



Il s'agit bien d'un système séquentiel car pour une même combinaison d'entrée, on peut observer une sortie différente.

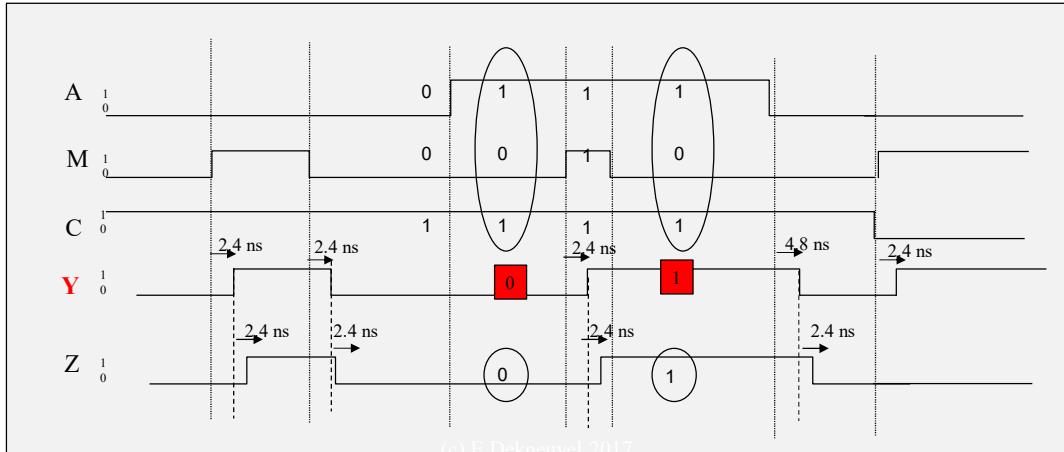
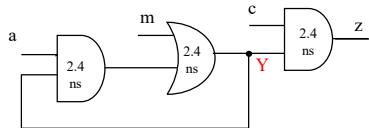
Bloc diagramme



Mode intermittent

Mode continu

Chronogramme de la FSM

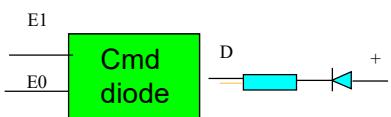


Dans ce circuit, la sortie ne peut être déterminée uniquement en fonction des entrées (ce qui serait combinatoire) puisqu'on voit qu'une même combinaison d'entrée telle que $AMC=101$ peut fournir deux sorties différentes en mode continu (0 puis 1) une fois que la modification de l'entrée m s'est propagée dans la boucle de rétro-action. C'est donc un circuit séquentiel (état interne Y) et il y a une notion d'historique. Il réagit par ailleurs immédiatement (au tps de propagation près) aux changements de niveaux d'une entrée: il est donc asynchrone.

Des temps de propagation permettent de mieux visualiser l'évolution temporelle de l'état interne et de la sortie. On notera qu'il est recommandé de ne pas changer plus d'une entrée à la fois dans un système asynchrone, les transitions simultanées d'entrées étant difficiles à garantir.

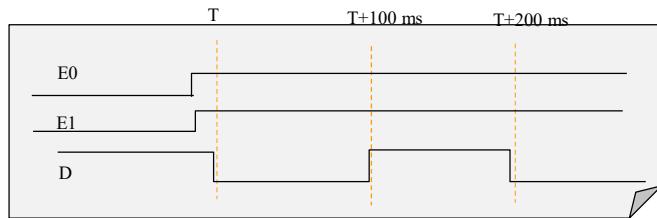
Limite des systèmes asynchrones

Cahier des charges



Un circuit numérique doit pouvoir commander une diode électroluminescente à partir d'une entrée E selon les modalités suivantes:

- Si $E_0=0$ et $E_1=0$, la diode est figée dans son état courant
- Si $E_0=1$ et $E_1=0$, la diode s'allume
- Si $E_0=0$ et $E_1=1$, la diode s'éteint
- Si $E_0=1$ et $E_1=1$, la diode clignote à une fréquence de 10 hz



Spécification

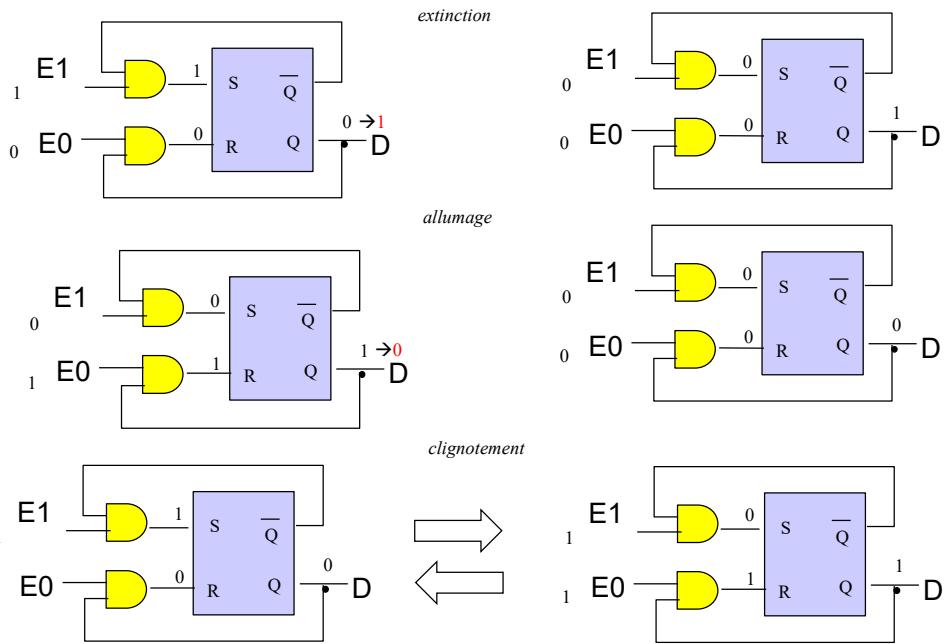
Dans ce système séquentiel, une contrainte de timing impose des évolutions de la sortie à des instants précis du temps

Solution asynchrone?



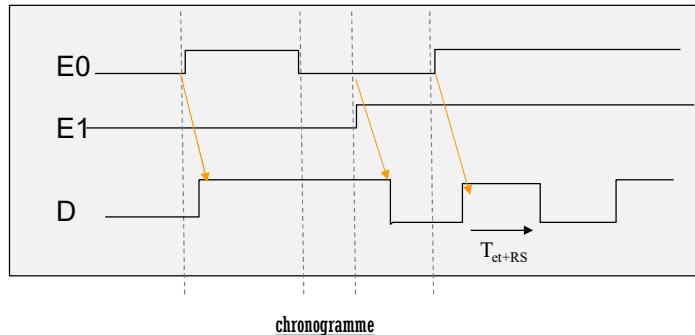
Il est possible de remplacer tout ou partie de la structure du circuit par une description de type table de transition pour simplifier l'analyse du comportement.

Validation de la solution gate-level



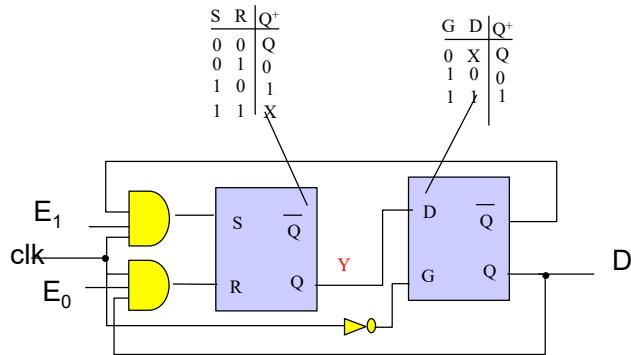
Les cas d'utilisation allumage/extinction fonctionnent proprement mais pas le clignotement

Analyse temporelle



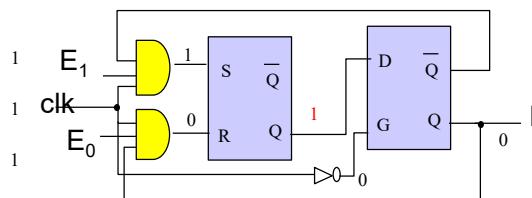
Dans ce système séquentiel, la contrainte de timing impose des évolutions de la sortie à des instants précis du temps, ce qui ne peut pas être garantit par une solution asynchrone. Ici, la modification de l'état interne s'effectue en fonction du temps de propagation dans les portes NOR.

Solution synchrone

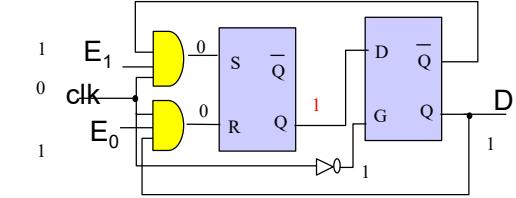


L'ajout d'un signal d'horloge permet de rythmer les évolutions du vecteur d'état à la bonne cadence en bloquant la réaction sur Y par le niveau de l'horloge

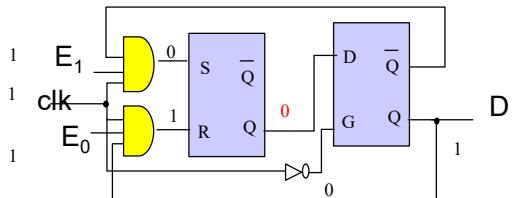
Validation de la solution gate level (cligno)



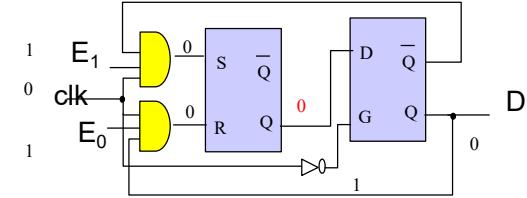
50 ms



100 ms



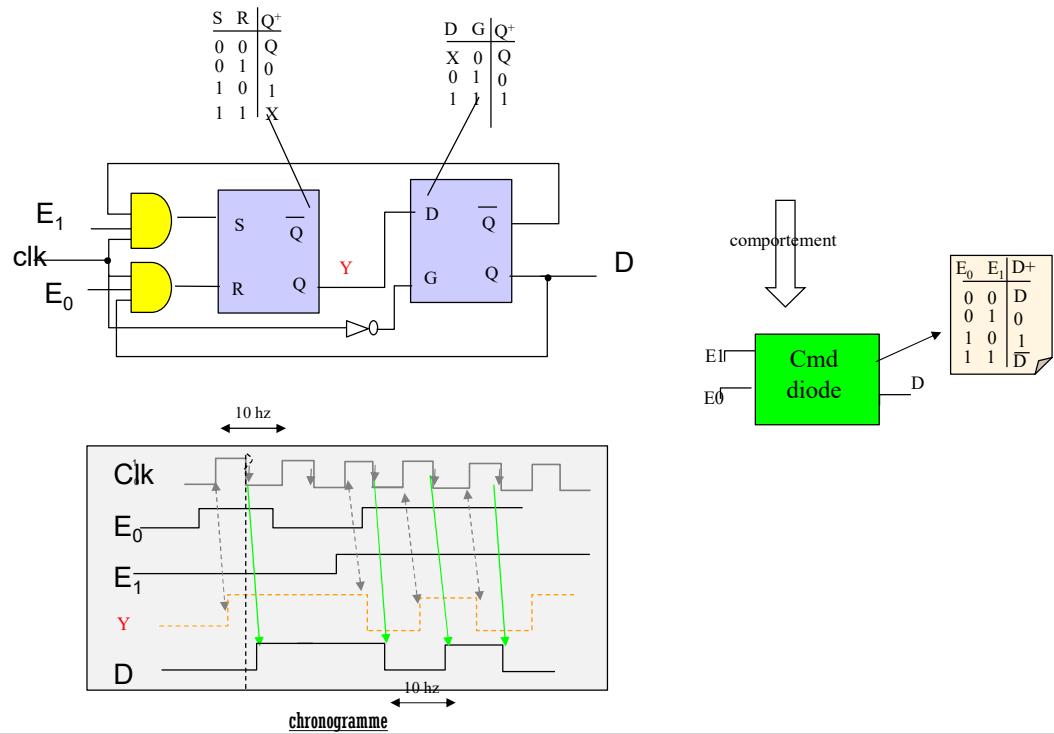
150 ms



200 ms

L'ajout d'un signal d'horloge permet de rythmer les évolutions du vecteur d'état à la bonne fréquence

Table de transition du circuit



Le chronogramme donne la réponse du circuit pour un scénario donné. La table de transition permet de décrire le comportement général du circuit

Automates à états finis

A. Catégories des FSM

B. Analyse des FSM

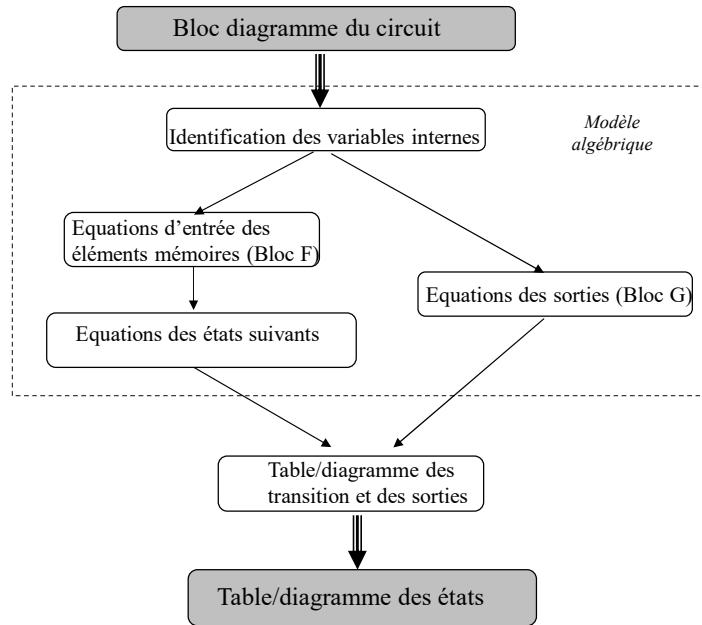
C. Synthèse des FSM

D. Codage des états

E. Réduction des machines séquentielles

(c) E.Dekneuvel 2021

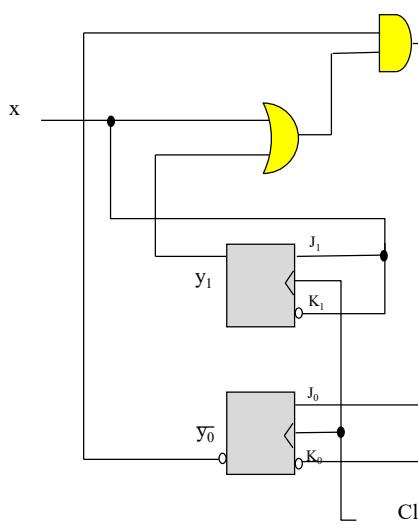
Méthode d'analyse



Contrairement à une description structurelle, un modèle comportemental permet de décrire le fonctionnement d'un circuit sans référence à une technologie particulière (portes, bascules). L'objectif de l'analyse est de dériver la fonctionnalité du circuit depuis son schéma logique, en dérivant un modèle algébrique, et éventuellement tabulaire ou graphique du circuit.

Illustration

Bloc diagramme



Fonctions d'excitation

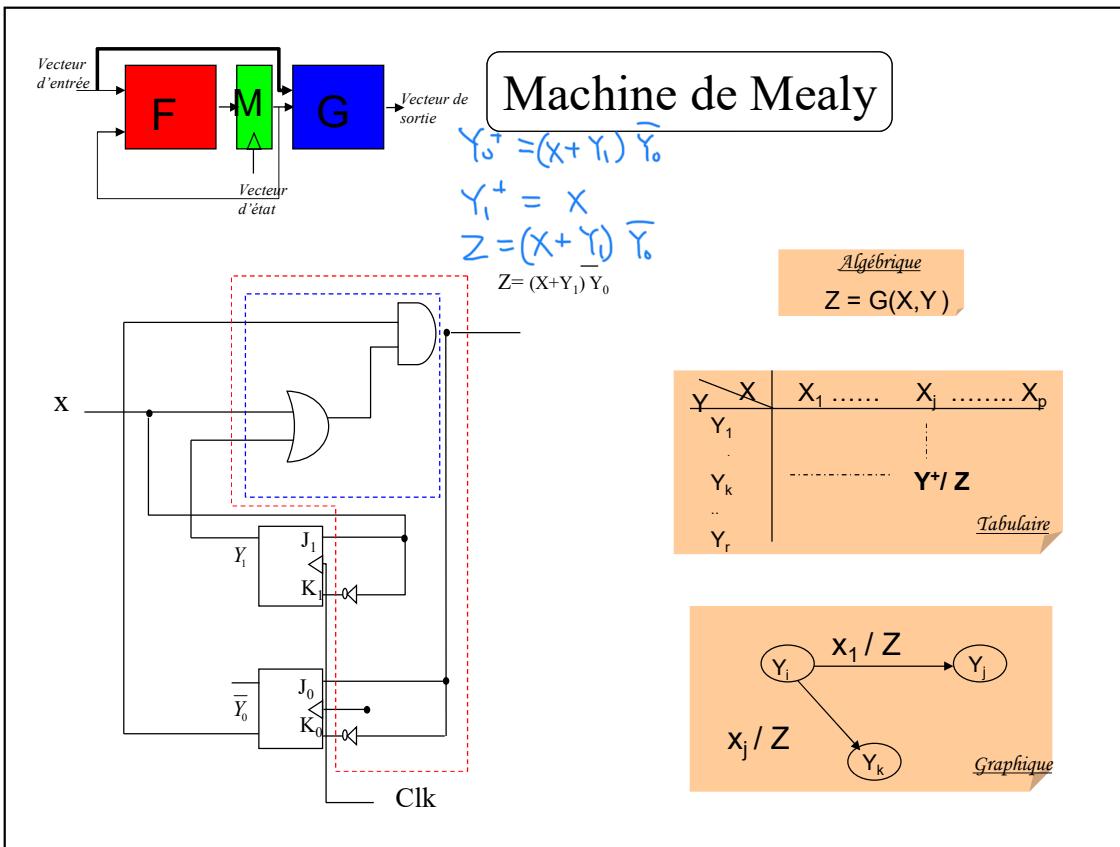
$$\begin{aligned} J_1 &= X & K_1 &= \bar{X} \\ J_0 &= (X+Y_1)\bar{Y}_0 & K_0 &= (X+Y_1)\bar{Y}_0 \end{aligned}$$

Equation de la bascule JK
 $Q^+ = J\bar{Q} + \bar{K}Q$

Modèle algébrique

$$\begin{aligned} Y_1^+ &= X \\ Y_0^+ &= (X+Y_1)\bar{Y}_0 \\ Z &= (X+Y_1)\bar{Y}_0 \end{aligned}$$

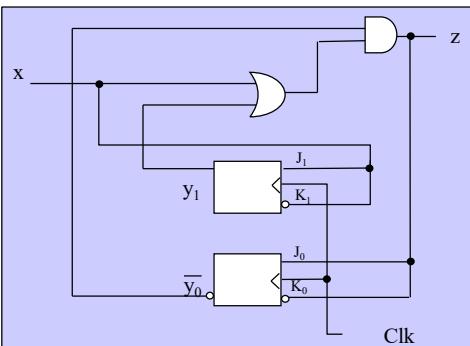
Dans ce circuit, le nouvel état calculé par la logique combinatoire (en entrée des bascules) ne peut apparaître en sortie des bascules qu'au signal d'horloge. En dehors, le système est verrouillé. Il s'agit d'une machine séquentielle synchrone. La représentation algébrique permet de représenter la fonctionnalité que le circuit implémente ou doit implémenter à l'aide d'un ensemble des fonctions booléennes (représentation formelle); Elle utilise des variables booléennes et des opérateurs logiques pour caractériser la relation Entrées Sorties et états.



Il s'agit d'une machine dont la sortie dépend d'une entrée du circuit. Une description tabulaire permet généralement une meilleure interprétation du comportement de la machine (tout comme un TV pour un circuit combinatoire). On parle de table de transition ou table d'états (adresses, états codés): elle peut être disposée en une ou deux dimensions. Dans ce dernier cas, les combinaisons du vecteur d'entrées sont disposées sur les colonnes. Les combinaisons du vecteur d'états sont disposées sur les lignes. La combinaison de l'état suivant est disposée à l'intersection de chacune des combinaisons d'entrée et d'état courant. Le diagramme des transitions est forme de représentation alternative, graphique, de la table dans laquelle les cercles entourent les combinaisons d'états, les arcs orientés relient les états courants avec leurs états suivants en fonction de la combinaison d'entrée qui sert d'étiquette au lien.

Diagramme/Table des transitions

Bloc diagramme



Modèle algébrique

$$\begin{aligned} Y_0^+ &= (X + Y_1) \bar{Y}_0 \\ Y_1^+ &= X \\ Z &= (X + Y_1) \bar{Y}_0 \end{aligned}$$

X Y ₁ Y ₀	0	1
00	00/0	11/1
01	00/0	10/0
10	01/1	11/1
11	00/0	10/0

Table des transitions $Y_1^+Y_0^+/Z$

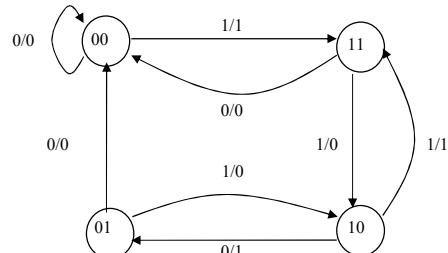
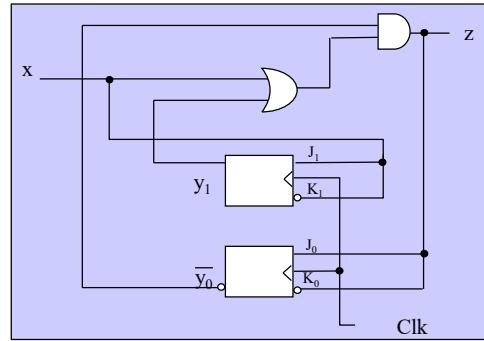


Diagramme des transitions

La table de transition s'obtient en injectant les combinaisons d'entrées possibles dans le modèle algébrique. Le diagramme n'est qu'un aperçu graphique équivalent à la table.

Diagramme/table des états



Bloc diagramme

$X \backslash Y_1 Y_0$	0	1
E0	E0/0	E3/1
E1	E0/0	E2/0
E2	E1/1	E3/1
E3	E0/0	E2/0

Table des états

$Y_1^+ Y_0^+ / Z$

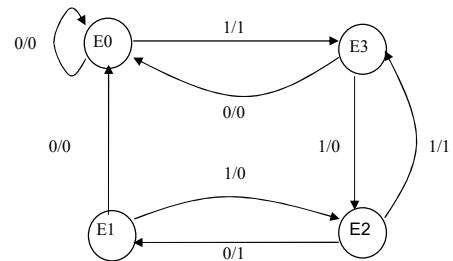
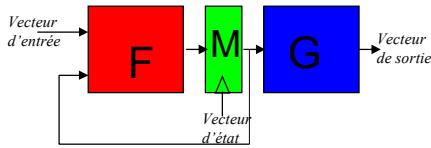


Diagramme des états

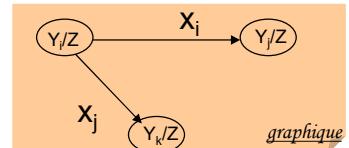
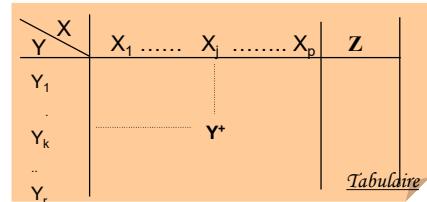
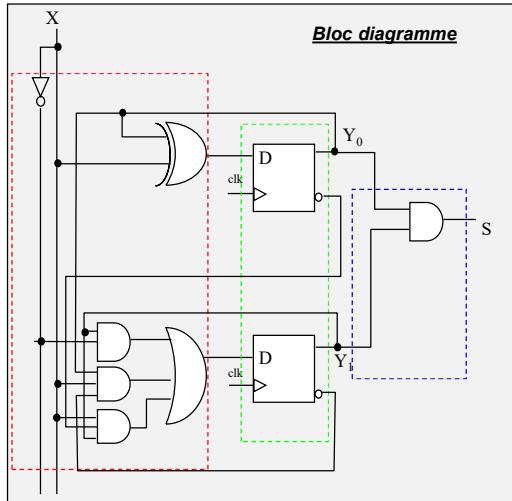
La table de d'états s'obtient en remplaçant les combinaisons numériques par des identificateurs.

Machine de MOORE

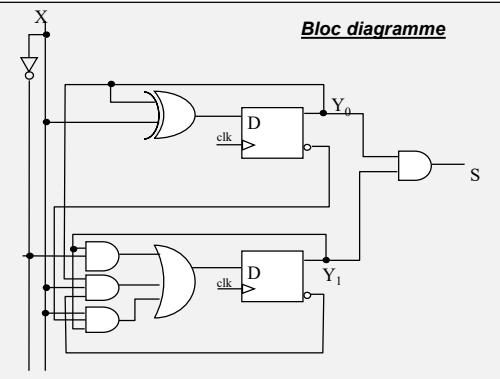


Algébrique

$$Z = G(Y)$$



Contrairement à une machine de MEALY, la sortie est isolée dans une colonne relative à l'état courant (MOORE) ou apparaît au niveau du cercle; car elle ne dépend pas des entrées. Dans ce type de machine, l'évolution de la sortie est donc, comme l'état, synchrone à l'horloge. On qualifie souvent les machines de Moore de machines séquentielles purement synchrones.



Analyse

Fonctions d'excitation

$$D_0 = Y_0 \oplus X$$

$$D_1 = \overline{X} \cdot Y_1 + X \cdot \overline{Y}_1 \cdot Y_0 + X \cdot Y_1 \cdot \overline{Y}_0$$

Modèle algébrique

$$\begin{aligned} S &= Y_1 \cdot Y_0 \\ Y_1^+ &= D_1 \\ Y_0^+ &= D_0 \end{aligned}$$

X \ Y ₁ Y ₀	0	1	S
00	00	01	0
01	01	10	0
10	10	11	0
11	11	00	1

Table des transitions

$$Y_1^+ Y_0^+ / Z$$

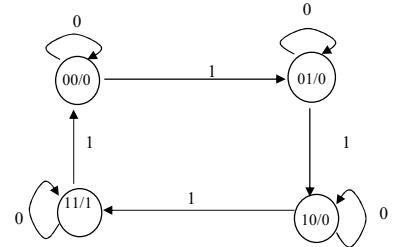


Diagramme des transitions

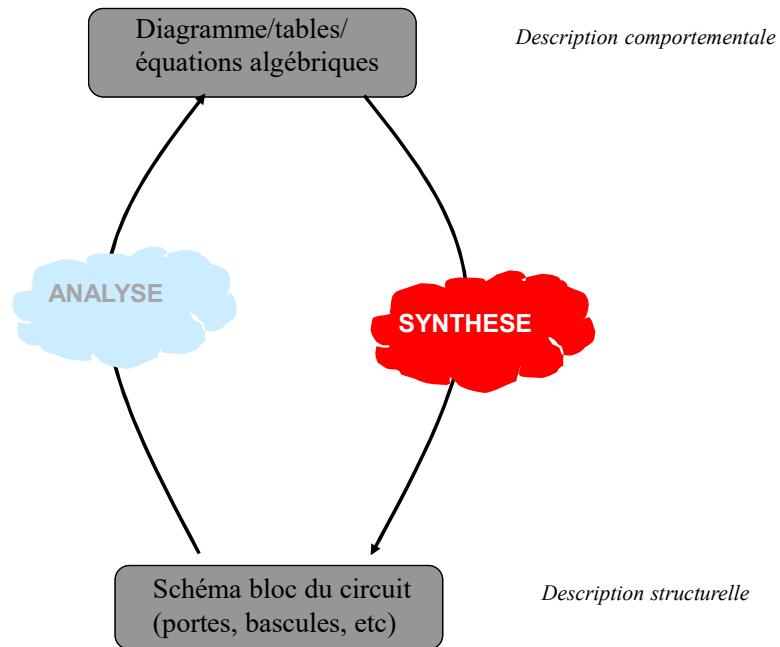
Dans le cas d'une machine de MOORE, plutot que de répéter autant fois la sortie que d'entrées, celle-ci est positionnée dans une colonne séparée de la table des transitions ou dans le cercle identifiant l'état pour le diagramme des transitions.

Automates à états finis

- A. Catégories de FSM*
- B. Analyse des FSM*
- C. Synthèse des FSM*
- D. Codage des états*
- E. Réduction des machines séquentielles*

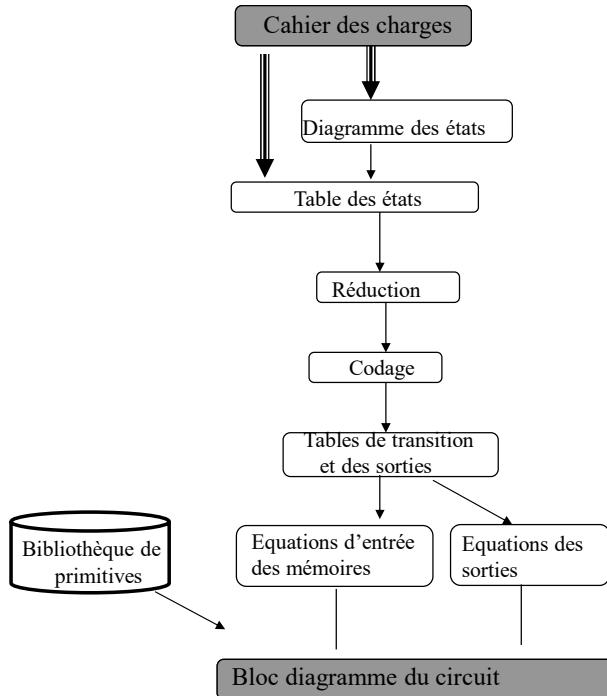
Comment formaliser le comportement de circuits séquentiels et déterminer une solution à base de composants primitifs (bascules, portes logiques)?

Synthèse des FSM



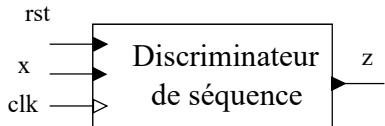
Si l'objectif de l'analyse est de dériver le modèle comportemental depuis le schéma logique, la synthèse procède à l'inverse.

Méthode de synthèse tabulaire

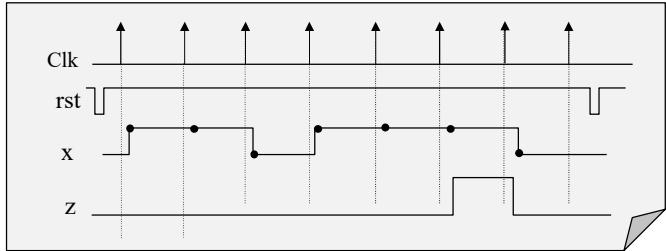


Le flot de conception procède par différentes étapes dont certaines peuvent être facultatives.

Illustration: discriminateur de séquences

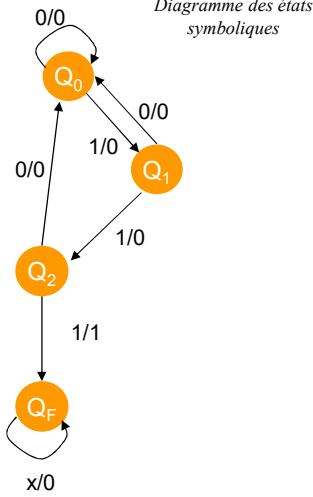


Le système à concevoir doit détecter l'apparition d'une séquence (ex: début d'un message) composé de 3 '1' consécutifs dans un signal x . Les données de la ligne sont synchronisées à partir d'une source d'impulsions d'horloge clk . Le circuit voit sa sortie z passer de 0 à 1 en coïncidence avec le front montant de l'horloge à partir du moment où la séquence est reconnue. La sortie repasse à zéro dans la période d'horloge suivante. Le circuit ne peut revenir sur son état initial que sur intervention externe. Un nouveau cycle complet ne peut donc s'effectuer que suite à un rst .



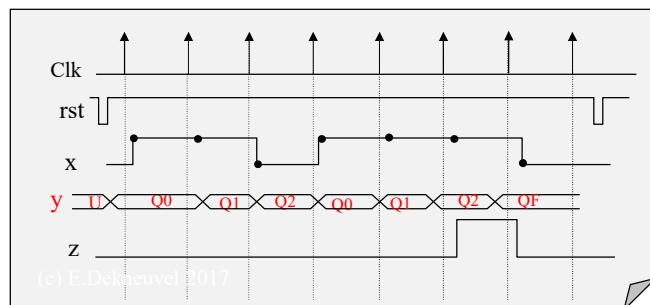
Le système à concevoir doit détecter l'apparition d'une séquence (ex: début d'un message) composé de 3 uns consécutifs dans un signal x . Les données de la ligne sont synchronisées à partir d'une source d'impulsions d'horloge clk . Le circuit voit sa sortie z passer de 0 à 1 en coïncidence avec le front montant de l'horloge à partir du moment où la séquence est reconnue. La sortie repasse à zéro dans la période d'horloge suivante. Le circuit ne peut revenir sur son état initial que sur intervention externe (rst)

Description comportementale



		X	0	1
Y	X\Y			
	Q ₀	Q ₀ /0	Q ₁ /0	
Q ₁	Q ₀ /0	Q ₂ /0		
Q ₂	Q ₀ /0	Q _F /1		
Q _F	Q _F /0	Q _F /0		

Validation fonctionnelle



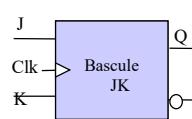
L'élément essentiel lors de la construction d'un diagramme d'état est d'identifier la nécessité de changer d'état pour mémoriser l'historique d'un certaine séquence dans le flot d'entrée. Chaque état caractérise donc un historique d'entrée jugé essentiel pour la reconnaissance de la séquence et doit posséder autant de successeurs que de combinaisons d'entrée. Par exemple ici, avec q1 on mémorise l'apparition d'un premier 1 après une suite de longueur indéterminée de 0. (1, 01, 001, etc). q2 pour 11 ou 011 ou 0011, etc. La validation fonctionnelle permet de vérifier si la description comportementale est conforme à la spécification

Codage binaire séquentiel/FF JK

Table des transitions

$Y_1 Y_0 \backslash X$	0	1
00	00/0	01/0
01	00/0	10/0
10	00/0	11/1
11	11/0	11/0

$Y_1 + Y_0 + Z$



Transition	J	K
0 → 0	0	X
1 → 1	X	0
1 → 0	X	1
0 → 1	1	X

Table des excitations

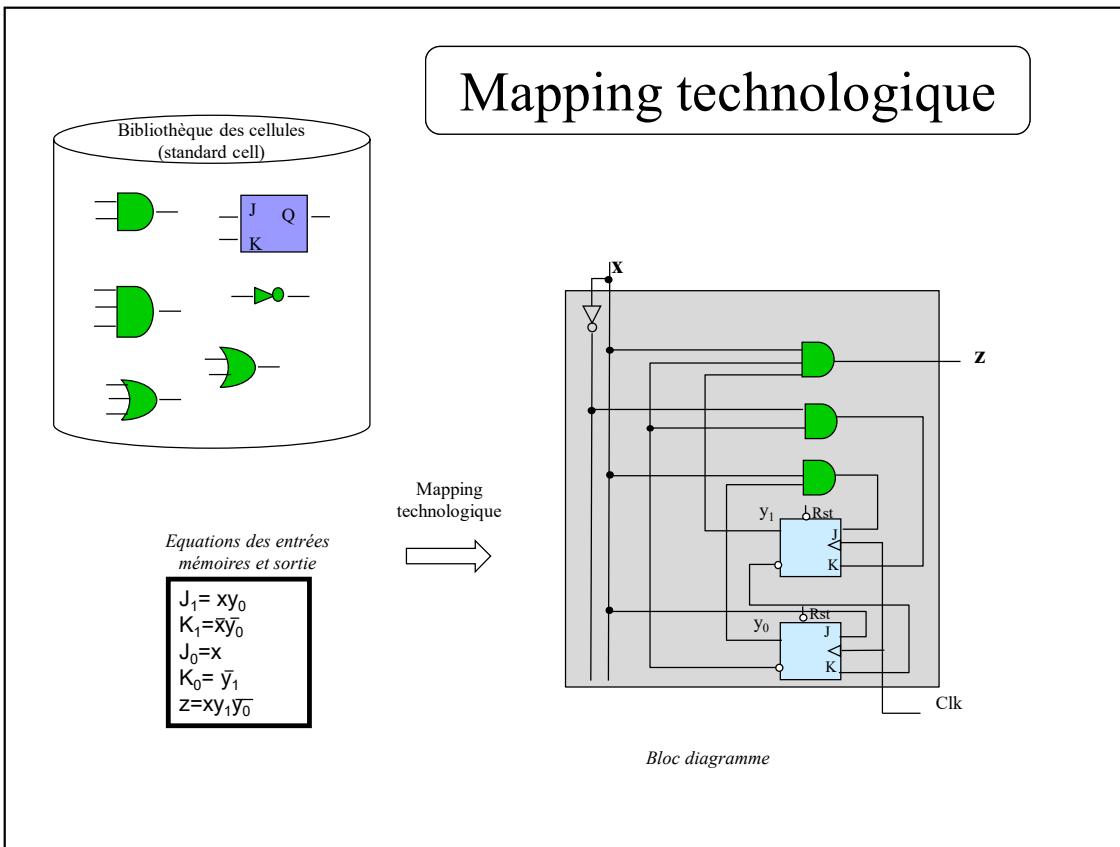
0 $J_1 K_1 J_0 K_0$	1 $J_1 K_1 J_0 K_0$
0X 0X	0X 1X
0X X1	1X X1
X1 0X	X0 1X
X0 X0	X0 X0

Equations des entrées mémoires et sortie

$$\begin{aligned} J_1 &= XY_0 \\ K_1 &= \bar{X}\bar{Y}_0 \\ J_0 &= X \\ K_0 &= \bar{Y}_1 \\ z &= XY_1\bar{Y}_0 \end{aligned}$$

La table d'excitation des entrées des éléments mémoires s'obtient à partir de la table des transitions en déterminant les valeurs à appliquer aux différentes entrées pour réaliser la transition demandée en fonction du comportement de la bascule JK:

Mapping technologique



L'ultime étape consiste à allouer les cellules d'une bibliothèque technologique aux expressions booléennes. Ici , on se contente d'utiliser des portes élémentaires. Une simulation du niveau gate level peut être réalisée pour vérifier la conformité de l'évolution de l'état interne et de la sortie

Codage binaire séquentiel/FF D

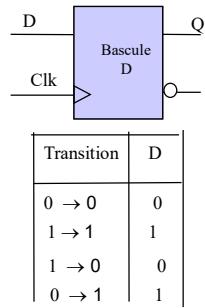
Table des transitions

X Y ₁ Y ₀	0	1
00	00/0	01/0
01	00/0	10/0
10	00/0	11/1
11	11/0	11/0

Y₁+Y₀+Z

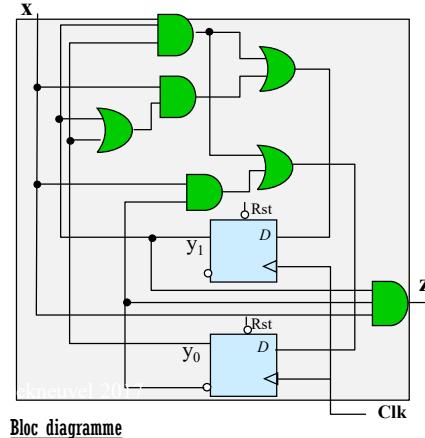
Table des excitations

0 D ₁ D ₀	1 D ₁ D ₀
00	01
00	10
00	11
11	11



Equations des entrées mémoires et sortie

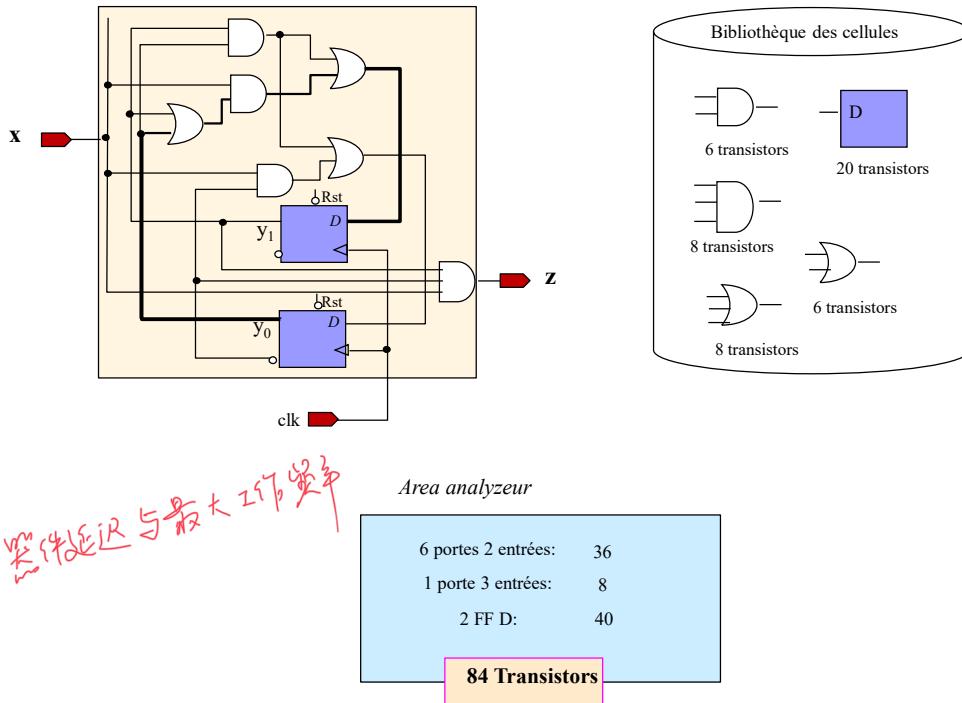
$$\begin{aligned} D_1 &= y_0 y_1 + (y_0 + y_1)x \\ D_0 &= xy_0 + y_1 y_0 \\ z &= xy_1 y_0 \end{aligned}$$



Bloc diagramme

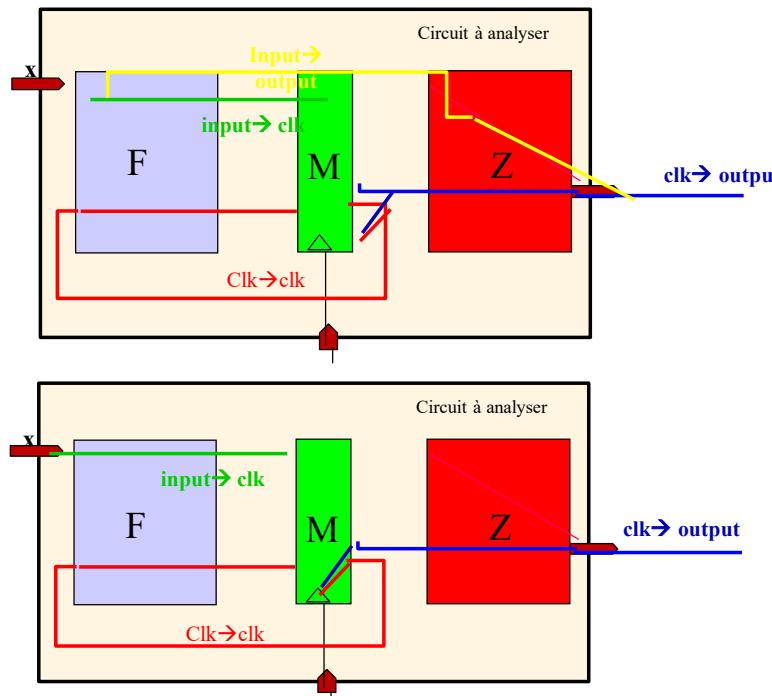
La méthode de synthèse est indépendante du type d'élément mémoire utilisé mais l'obtention de la table des excitations dans une synthèse avec des bascules D est quasi-directe. En plus, il n'y a que 2 fonctions booléennes à déterminer ici. On peut également noter que la fonction combinatoire du générateur d'état a été optimisée pour la surface par factorisation de termes communs. Ressources nécessaires: 3 OU2 et 3 ET2 pour matérialiser la fonction générateur d'état et 1 ET3 pour la fonction de génération de la sortie. En logique 2 niveaux, il faudrait 3 ET2 et un OU3 pour le générateur des états suivants

Evaluation de Surface(CMOS)



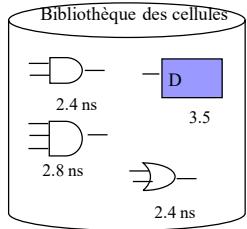
On suppose que les bascules sont forcées à zéro grâce à des entrées asynchrones au début de la simulation. S'agissant d'une machine de Mealy, on voit qu'une modification de l'entrée X (fournie par exemple par un récepteur série) peut se répercuter sur la sortie Z en 2.8 ns. Remarque: dans le calcul des délais, on ne tient pas compte du tsetup. Par contre, ce temps est pris en compte dans le calcul de la Fréquence maximale de fonctionnement du circuit.

Table des délais internes

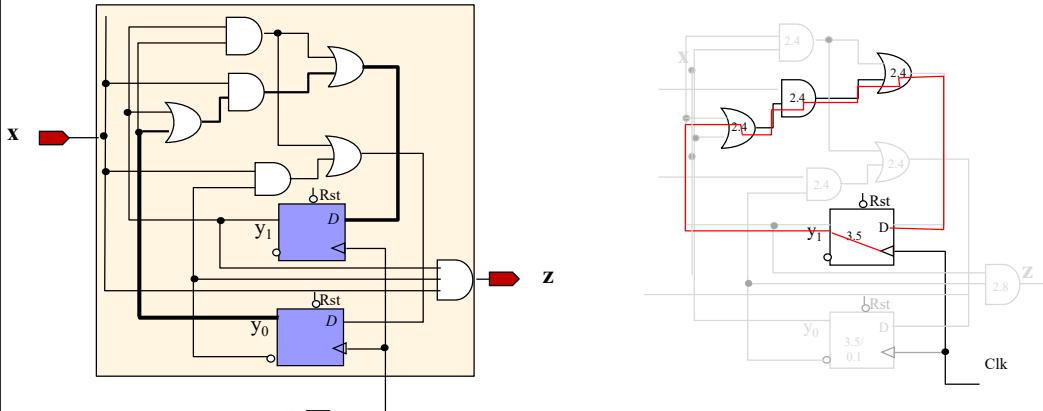


Quatre chemins doivent être évalués en interne du circuit pour déterminer les performances en terme de timing s'il s'agit d'une machine de Mealy. Trois seulement (pas de input to output) s'il s'agit d'une machine de Moore.

Délai clk to clk

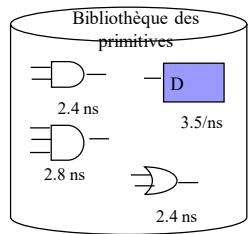


chemin	Délai critique
clk to clk	10.7 ns
Input to clk	
clk to output	
Input to output	

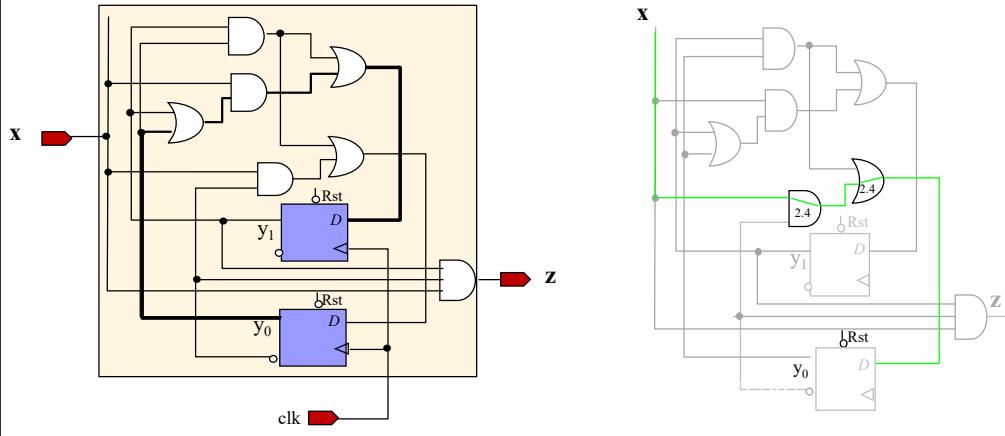


On détermine ici le délai maximal de propagation dans la logique séparant une sortie de FF à une entrée de FF. On prend le délai le plus long (chemin critique)

Délai input to clk

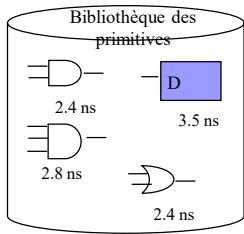


chemin	Délai critique
clk to clk	
Input to clk	4.8 ns
clk to output	
Input to output	

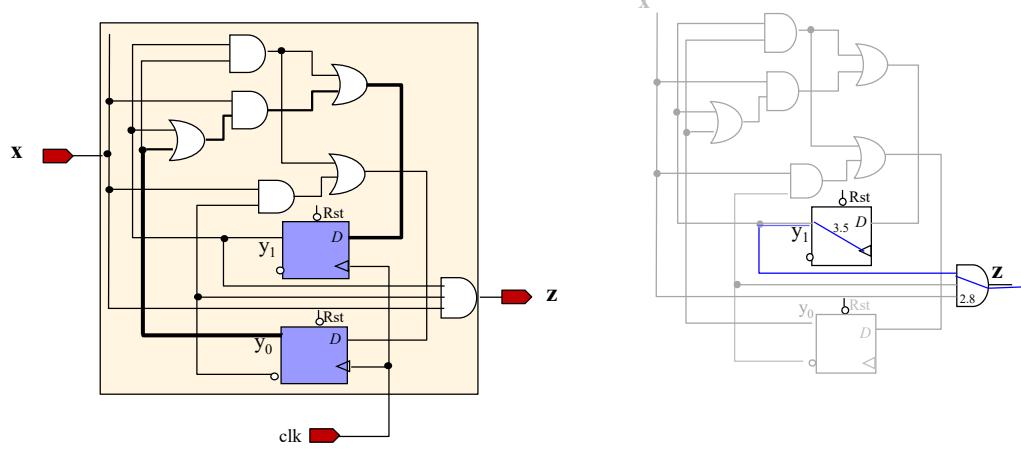


On détermine ici le délai maximale de propagation depuis une entrée du circuit jusqu'à l'entrée d'une FF

Délai clk to output

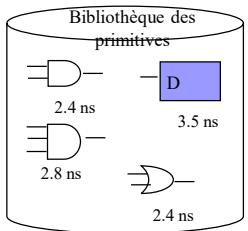


chemin	Délai critique
clk to clk	
Input to clk	
clk to output	6.3 ns
Input to output	

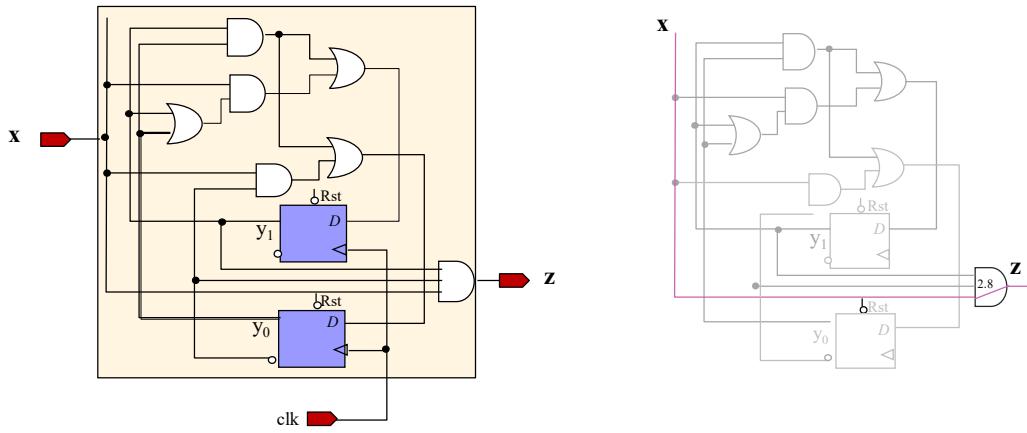


On détermine ici le délai maximal de propagation du signal depuis une des FF jusqu'à la sortie

Délai input to output(mealy)

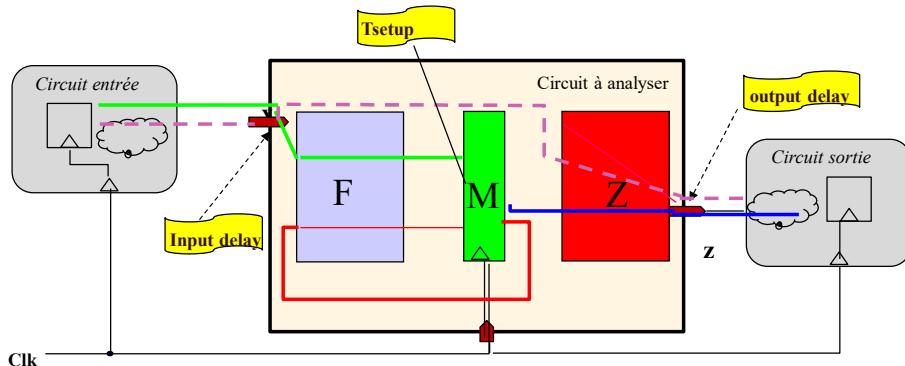


chemin	Délai critique
clk to clk	
Input to clk	
clk to output	
Input to output	2.8 ns



pour une machine de Mealy, il s'agit du délai maximal de propagation depuis une entrée jusqu'à une sortie de circuit (ici de X à Z)

Fréquence maximale du circuit



Periode min clk=>Max

- Clk to clk+Tsetup
- Input_delay+input to clk+tsetup
- Output to clk+output_delay
- Input_delay+input to output+output_delay (Mealy)

La fréquence maximale/période minimale de l'horloge se calcule sur le plus long chemin parmi 3(moore) ou 4(mealy) chemins possibles dans le circuit:

Clk to clk: temps de propagation dans la logique séparant une sortie de FF à une entrée de FF

Clk to output: temps de propagation jusqu'à la sortie

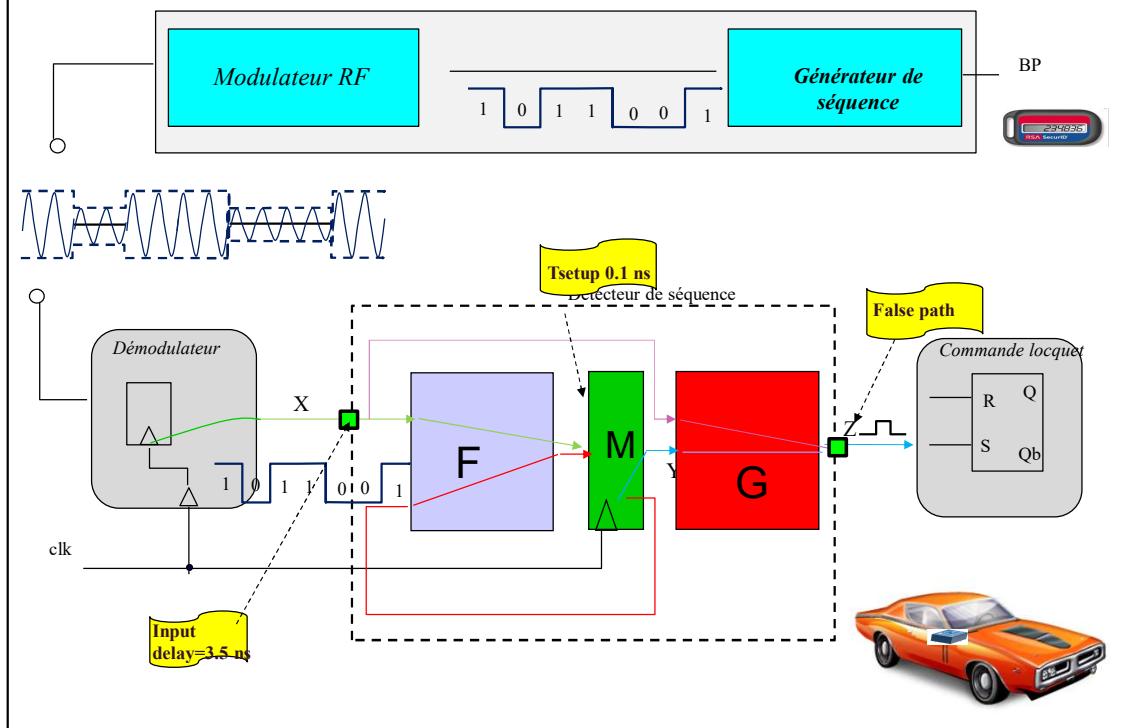
Input to clk : temps de propagation depuis une entrée du circuit jusqu'à l'entrée d'une FF

Input to output(mealy): pour une machine de Mealy, temps de propagation depuis une entrée jusqu'à une sortie de circuit.

Pour chaque chemin, on ne considère que le délai le plus long (chemin critique)

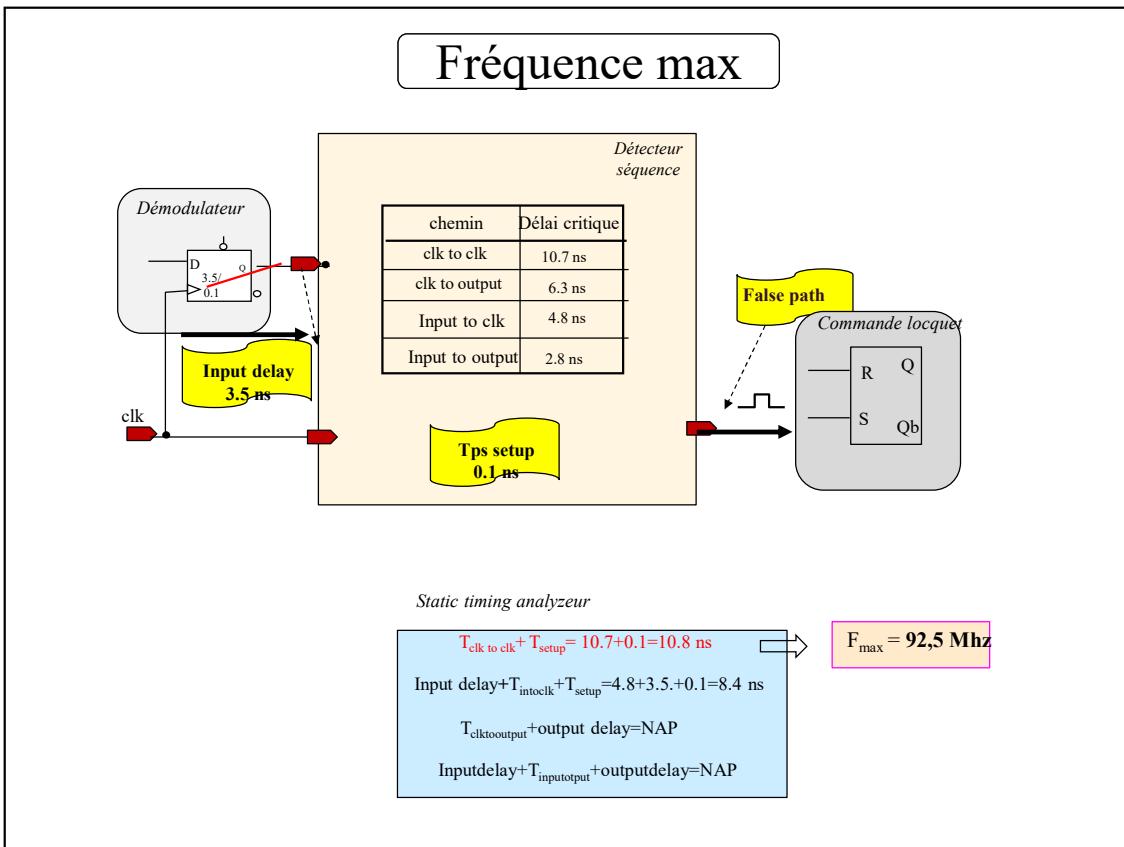
Cette période prend en compte la table des délais internes au circuit mais également les contraintes en entrée(input delay) de sortie(output delay) imposés par l'environnement; si ces circuits sont synchrones entre eux ou encore interne (tsetup de la technologie)

Illustration



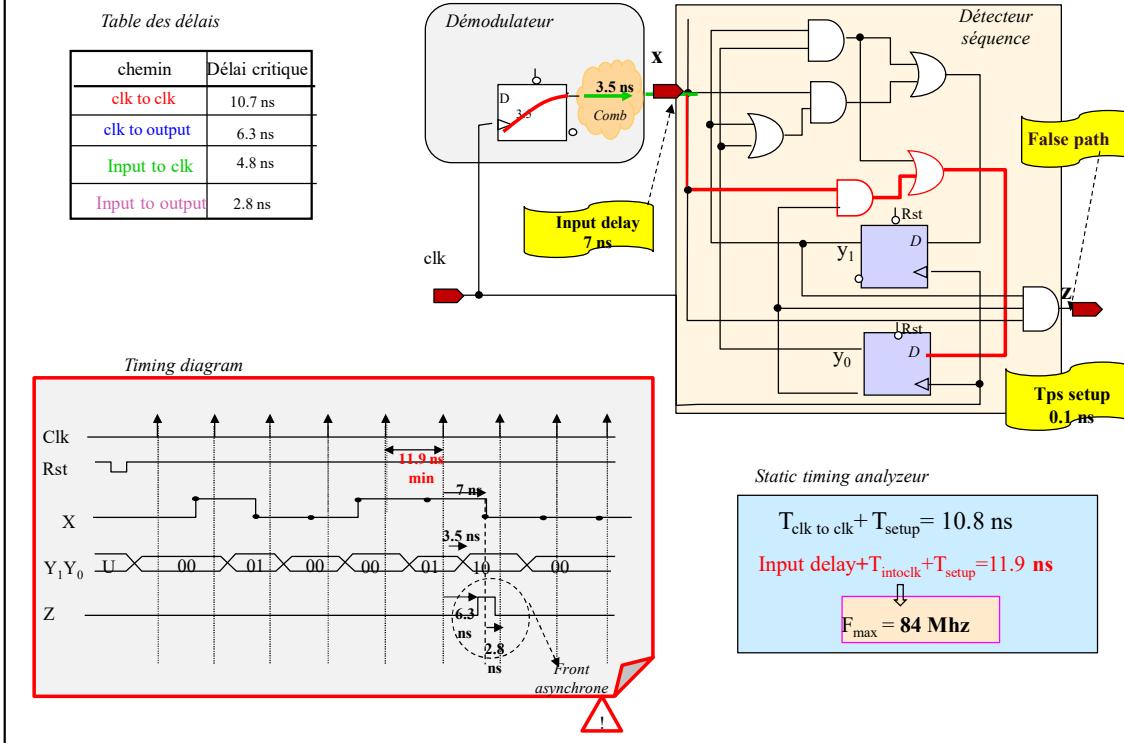
Une fois le circuit élaboré, il importe de vérifier la qualité de la solution vis-à-vis des critères de conception privilégiés; surface, temps, consommation, etc. En ce qui concerne les contraintes de timing, différents chemins doivent être analysés et validés vis-à-vis d'un certain nombres de paramètres tel que le débit des entrées(débit binaire ici) , le input delay, ...

Fréquence max



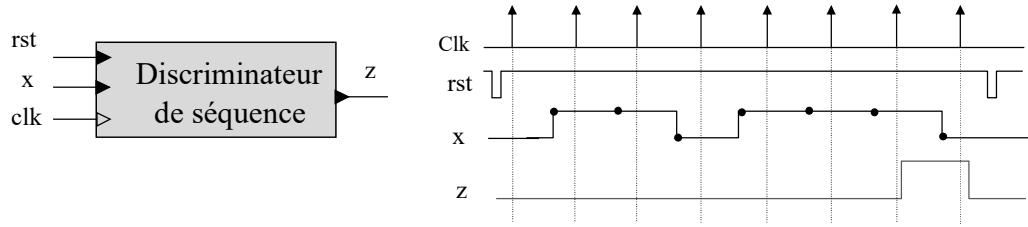
On suppose que les bascules sont forcées à zéro grâce à des entrées asynchrones au début de la simulation. S'agissant d'une machine de Mealy, on voit qu'une modification de l'entrée X (fournie par exemple par un récepteur série) peut se répercuter sur la sortie Z en 2.8 ns. Remarque: dans le calcul des délais, on ne tient pas compte du tsetup. Ce temps est pris en compte dans le calcul de la Fréquence maximale de fonctionnement du circuit.

Limite des machines de MEALY



Si l'entrée est asynchrone à l'horloge (modifiée peu avant le front d'horloge), en provenance par exemple d'un circuit de décodage du signal, la sortie obtenue n'est plus synchrone (3.5 ns après le clk qui est le délai tco). Autrement dit, l'entrée n'est plus modifiée en concordance avec le changement d'état mais après ce changement d'état. Dans le scénario ci-dessus, il n'y a en réalité que 2 '1 consécutifs et on observe néanmoins une impulsion (changement d'état fugitif) sur la sortie. Une impulsion parasite (bruit) sur l'entrée aurait également le même effet. Ceci vient du fait que la machine de Mealy n'est pas une machine purement synchrone.

Discriminateur de séquences(Moore)



Le système à concevoir doit détecter l'apparition d'une séquence (ex: début d'un message) composé de 3 un consécutifs dans un signal x . Les données de la ligne sont synchronisées à partir d'une source d'impulsions d'horloge clk . Le circuit voit sa sortie z passer de 0 à 1 **sur le front montant de l'horloge suivant l'échantillonnage du 3^{ème} '1' de la séquence**. La sortie repasse à zéro dans la période d'horloge suivante. Le circuit ne peut revenir sur son état initial que sur intervention externe. Un cycle complet de détection ne peut donc s'effectuer que suite à un rst .



Si les entrées ne sont plus asynchrones mais assujetties à l'horloge, il est possible de concevoir le circuit sous la forme d'une machine de Moore.

Synthèse logique

Diagramme des états

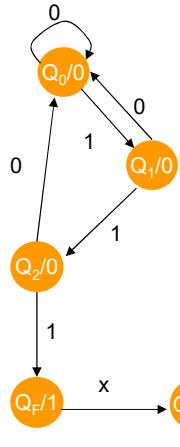


Table des états

X Y	0	1	Z
Q0	Q0	Q1	0
Q1	Q0	Q2	0
Q2	Q0	QF	0
Qv	QG	QG	1
QG	QG	QG	0

Table des transitions(codage séquentiel)

X Y2 Y1 Y0	0	1	Z
000	000	001	0
001	000	010	0
010	000	011	0
011	100	100	1
100	100	100	0

Équations des entrées
mémoires et sortie

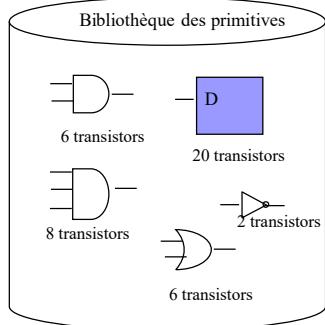
$$\begin{aligned}
 D_2 &= Y_2 Y_1 Y_0 \\
 D_1 &= Y_1 \bar{Y}_0 x + Y_0 \bar{Y}_1 x \\
 D_0 &= \bar{Y}_2 \bar{Y}_0 x \\
 Z &= Y_1 Y_0
 \end{aligned}$$

On peut toujours passer d'une machine de Moore à une machine de Mealy et vice versa. Une machine de Moore sera toujours plus complexe qu'une machine de Mealy en terme de logique séquentielle (nombre d'état m de Moore sera toujours borné par la formule: $qn+1$ où q=nbre de sorties et n=nbre d'états en Mealy).

La fonction générateur des sorties de Moore sera par contre généralement moins complexe car l'équation du vecteur de sortie ne dépend plus du vecteur d'entrée

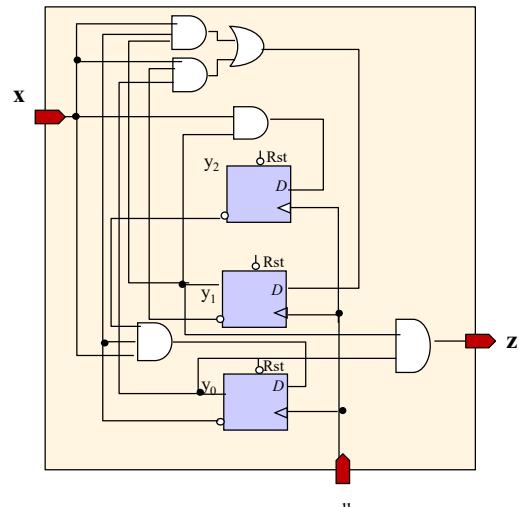
On vérifie ici que $5 \leq 4*1+1$.

Mapping



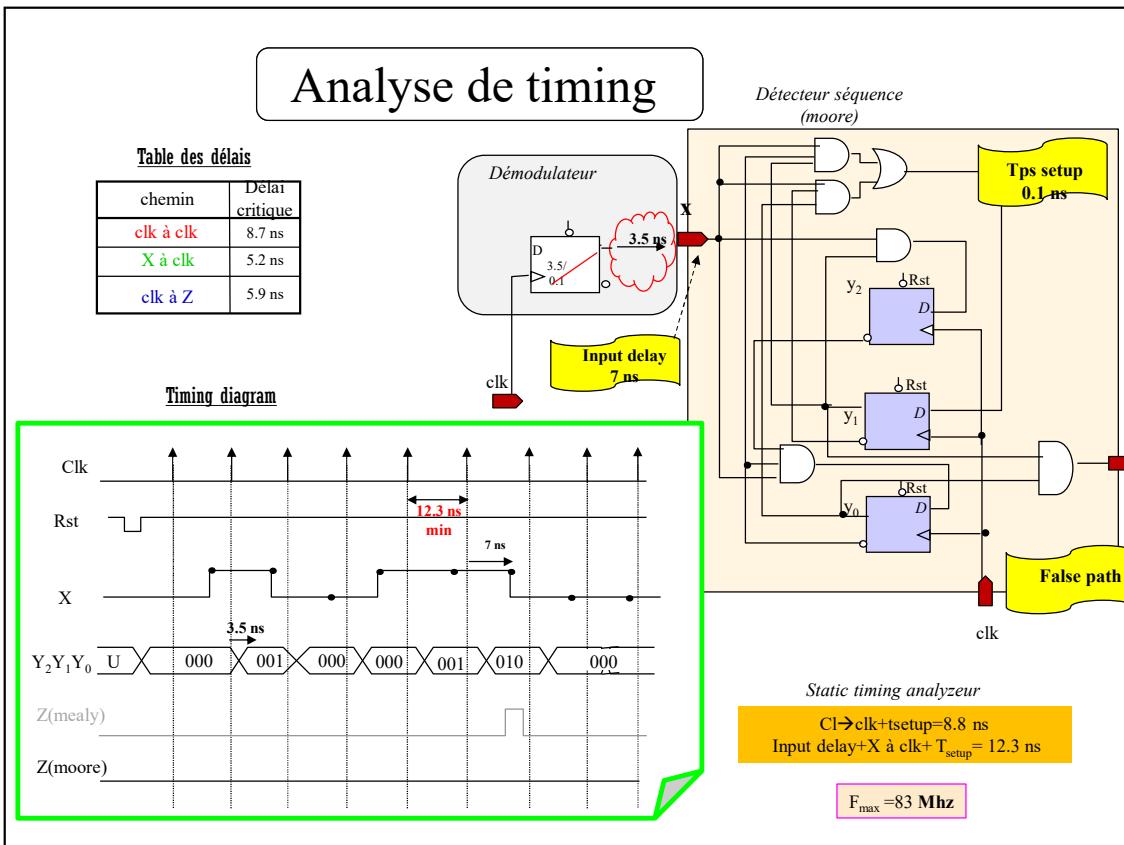
$$\begin{aligned}
 D_2 &= Y_2 Y_1 Y_0 \\
 D_1 &= Y_1 \bar{Y}_0 x + Y_0 \bar{Y}_1 x \\
 D_0 &= \bar{Y}_2 \bar{Y}_0 x \\
 Z &= Y_1 Y_0
 \end{aligned}$$

Bloc diagramme



Aera report 102 T

Analyse de timing



La sortie ne peut plus anticiper l'évolution de X. La sortie de Moore ne passe pas à 1 car on n'échantillonne que deux '1' consécutifs. On dit que les machines de Moore sont purement synchrones (évolution de l'état ET des sorties liées à l'horloge)

Automates à états finis

A. Catégories de FSM

B. Analyse des FSM

C. Synthèse des FSM

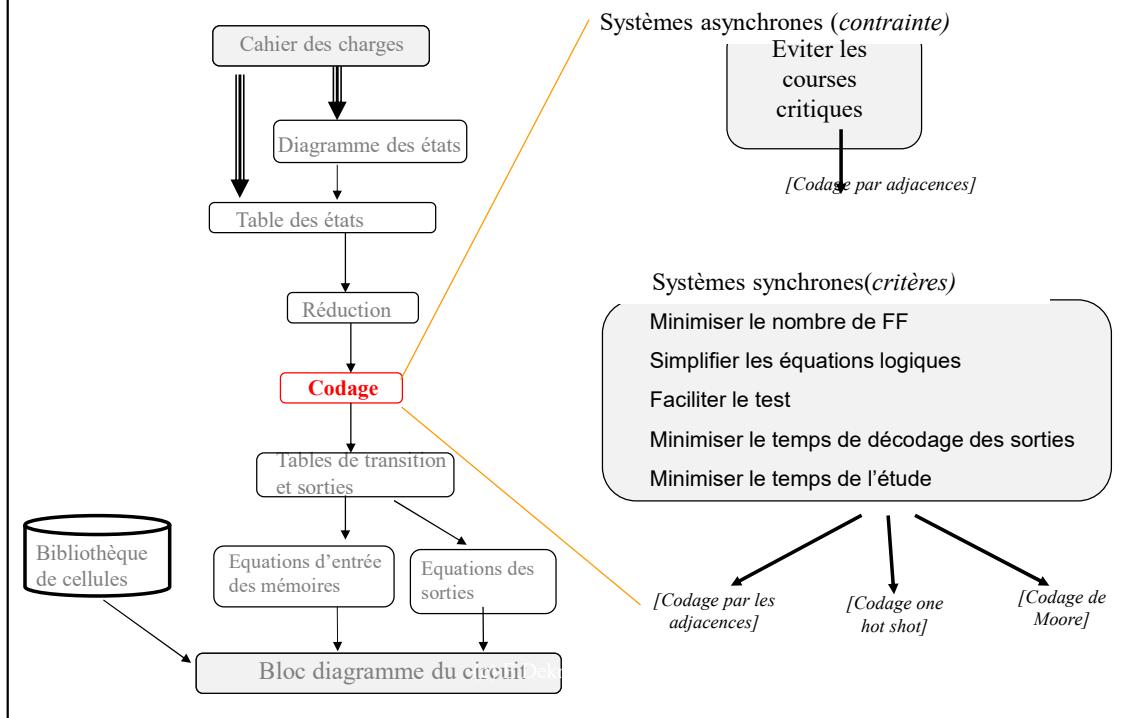
D. Codage des états

E. Réduction des machines séquentielles

(c) E.Dekneuvel 2021

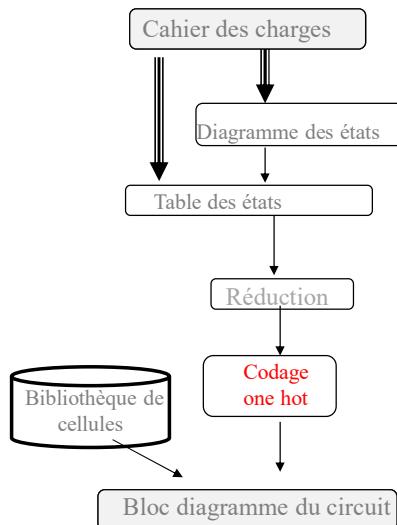
comment affecter à chaque ligne de la table des états, un nombre binaire comprenant un certain nombre de bits à 1 et 0?

Contraintes et critères



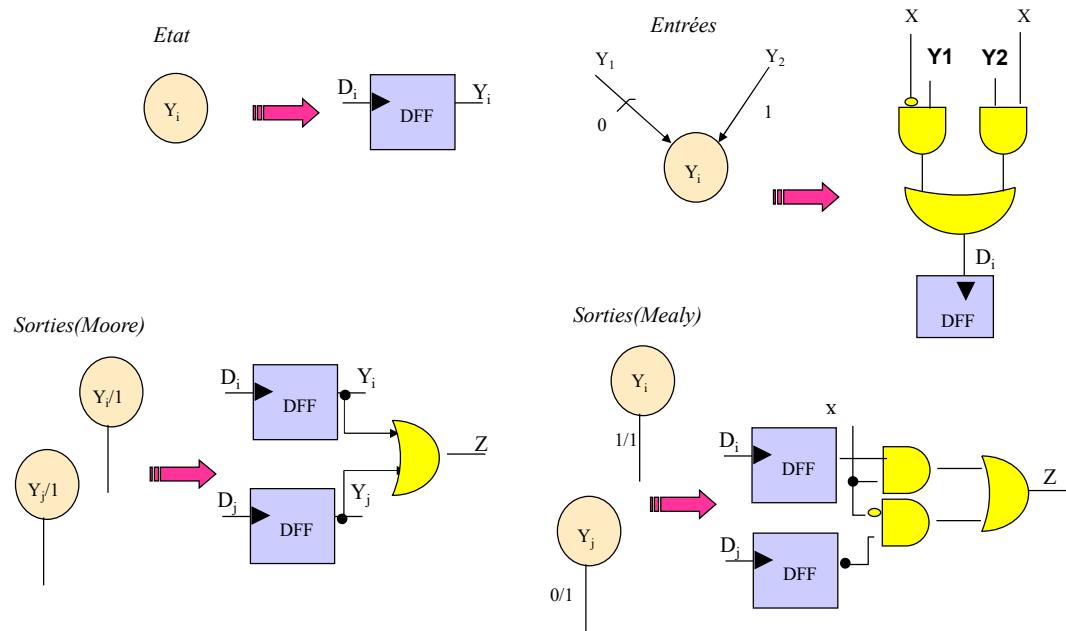
Le codage sert à matérialiser le système séquentiel à partir de la table et représente donc une étape importante de la réalisation. Pour les systèmes asynchrones, une contrainte est d'affecter des codes respectant impérativement des règles d'adjacence pour éviter les phénomènes de courses critiques. Pour les systèmes synchrones, le codage dépend de critères conduisant à des avantages en termes de réalisation (surface, performances, robustesse, etc)

Synthèse directe

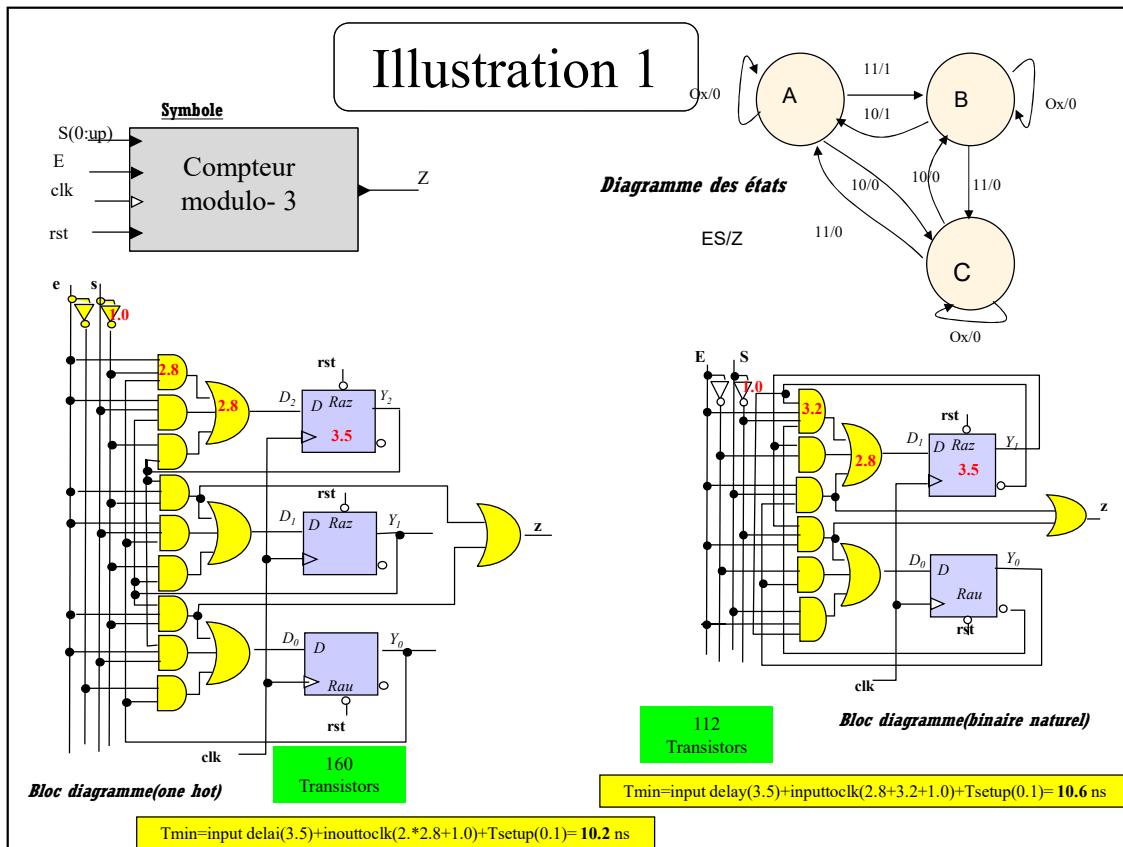


La synthèse logique est directe: plus besoin d'établir une table de transition et de rechercher les équations logiques comme dans la méthode de synthèse traditionnelle

Codage one hot (système synchrone)

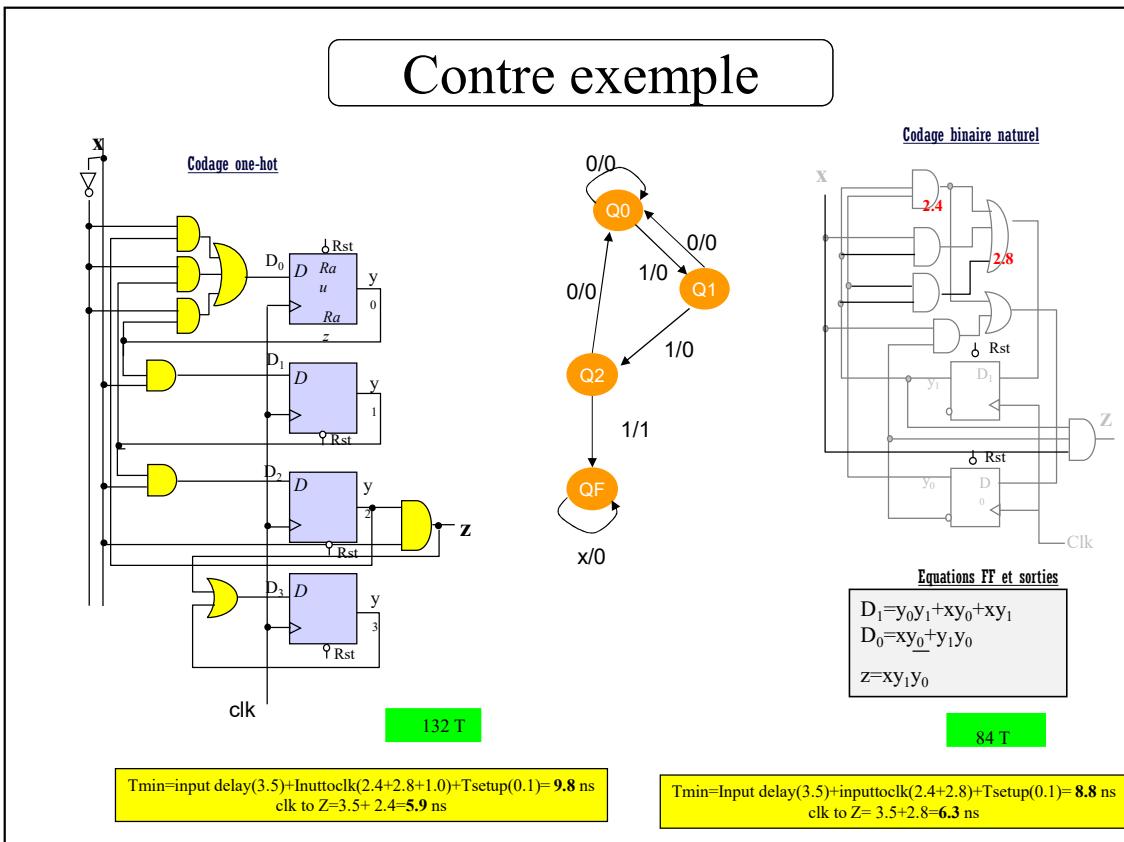


Méthode de conception considérant, pour une machine à n états, un code formé de n bits avec un seul bit à un. Seules n combinaisons parmi les 2^n sont utilisées. Cette méthode autorise une déduction directe du bloc diagramme par transposition des nœuds et flèches du diagramme sous forme matérielle.



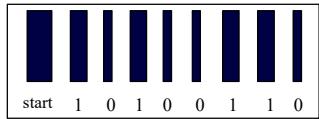
Pour ce circuit, le nombre d'éléments séquentiels est plus important en codage one hot qu'en codage binaire mais la complexité de la logique combinatoire est plus faible (On peut noter que l'écart de complexité croît avec le nombre d'états). Le nombre d'entrée étant généralement plus faible, les performances du générateur (F_{max} de l'horloge) sont donc généralement meilleures en one hot. D'autre part, et quelque soit le nombre d'états du circuit, 2 FF au maximum devant modifier leur valeur à chaque cycle d'horloge (la FF qui revient à '0' et celle qui passe à '1'), la consommation est également généralement plus faible.

Contre exemple



On voit dans cet exemple que le codage one hot pénalise ici légèrement les performances du générateur des états suivant. Ceci est essentiellement du au fait que 3 flèches amènent sur l'état E0. Par contre, il améliore légèrement celui du générateur des sorties (porte OU2 contre une porte OU3) dont le décodage est plus court

Codage de Moore



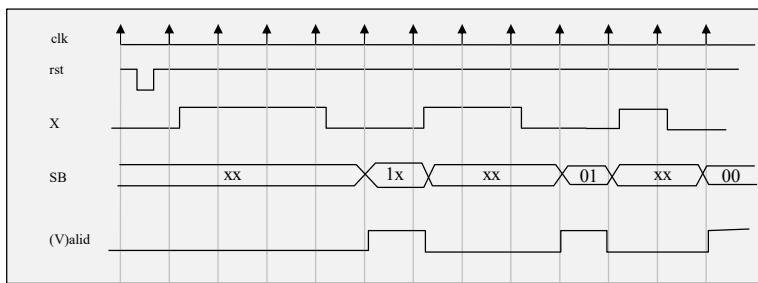
Cahier des charges

On souhaite réaliser un circuit capable de mesurer la largeur des impulsions issues d'un faisceau lumineux en entrée synchronisé par l'horloge clk. Le rôle du circuit est de produire des sorties S_1S_0 dont la valeur détermine si l'impulsion est :

- courte (sorties SB = 00) : l'impulsion dure une période d'horloge: on est en présence d'un bit de donnée à 0
- moyenne (sorties SB = 01) : l'impulsion dure deux périodes d'horloge: on est en présence d'un bit de donnée à 1
- longue (sorties SB= 1x) : l'impulsion dure trois périodes d'horloge.: on est en présence d'une barre de start

On notera qu'une impulsion de x dure au maximum 3 périodes d'horloge. Il est donc **impossible** d'avoir à mesurer une impulsion sur 4 périodes d'horloge ou plus. Un signal *valid* permet de préciser l'intervalle de validation des sorties.

Les différentes impulsions sont **obligatoirement** séparées d'au **minimum** deux périodes d'horloge durant lesquelles le signal d'entrée est maintenu à zéro.



Spécification

Le signal V permet d'indiquer la validité de la donnée car le calcul peut prendre 3,4 ou 5 cycles d'horloges selon le type de barre.

Description MOORE

Diagramme des états

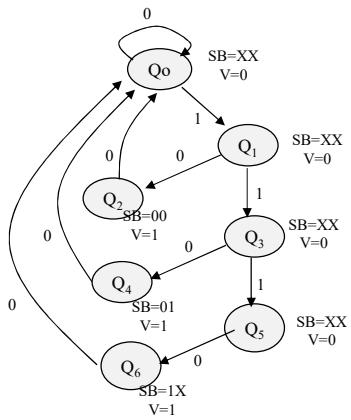
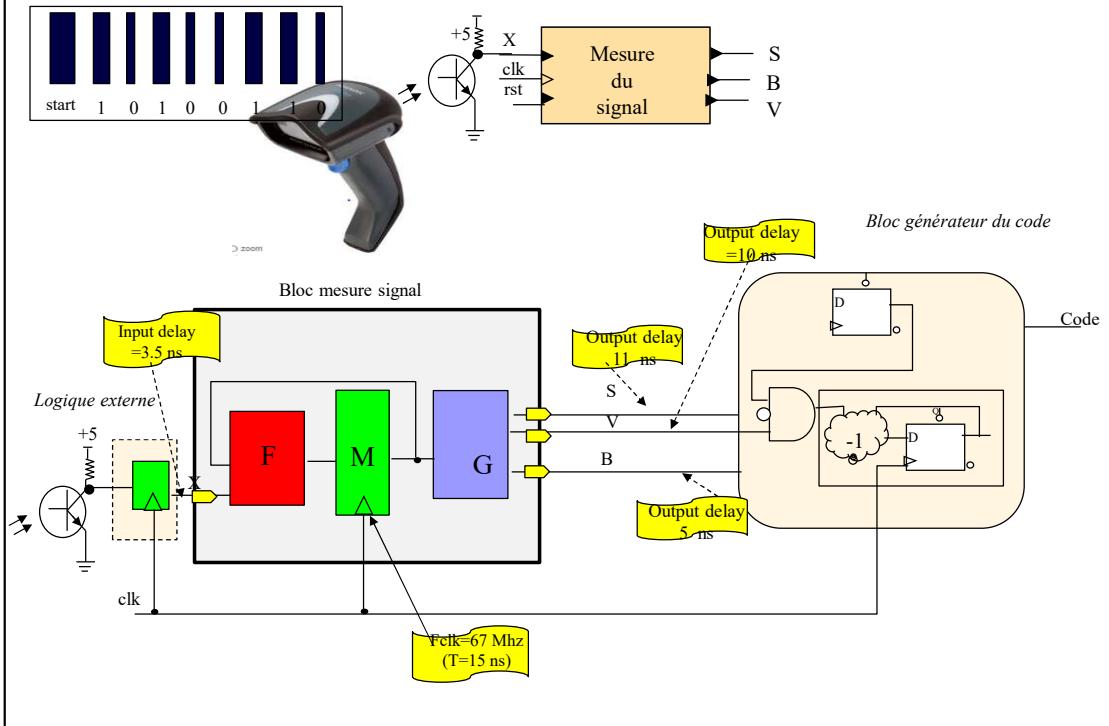


Table des états

Y	X=0	X=1	V S B
Q ₀	Q ₀	Q ₁	0 - -
:Q ₁	Q ₂	Q ₃	0 - -
Q ₂	Q ₀	-	1 0 0
Q ₃	Q ₄	Q ₅	0 - -
Q ₄	Q ₀	-	1 0 1
Q ₅	Q ₆	-	0 - -
Q ₆	Q ₀	-	1 1 x

Une description comportementale à partir du modèle de Moore est privilégiée, machine pour laquelle les sorties sont définies au niveau des états.

Contraintes de l'environnement



Le output delay représente le temps nécessaire au traitement de V additionné au temps de setup du registre. Le bloc exerne effectue la génération du code (contient notamment des registres tels que un compteur et un registre à décalage pour stocker le code final. Ce bloc est optimisé pour la fréquence de 20 Mhz et ne peut être modifié. Par conséquent, le temps de propagation dans le registre d'état et la logique menant à B ne doit pas dépasser 5 ns.

Codage binaire naturel?

Table des états

Y	X=0	X=1	V S B
Q ₀	Q ₀	Q ₁	0 - -
Q ₁	Q ₂	Q ₃	0 - -
Q ₂	Q ₀	-	1 0 0
Q ₃	Q ₄	Q ₅	0 - -
Q ₄	Q ₀	-	1 0 1
Q ₅	Q ₆	-	0 - -
Q ₆	Q ₀	-	1 1 -

Affectations

Q₀:011
Q₁:111
Q₂:000
Q₃:100
Q₄:001
Q₅:101
Q₆:010

Table des transitions

Y ₂ Y ₁ Y ₀	X=0	X=1	V S B
011	011	111	0 - -
111	000	100	0 - -
000	011	-	1 0 0
100	001	101	0 - -
001	011	-	1 0 1
101	010	-	0 - -
010	011	-	1 1 -
110	-	-	xxx



Équations des entrées FF

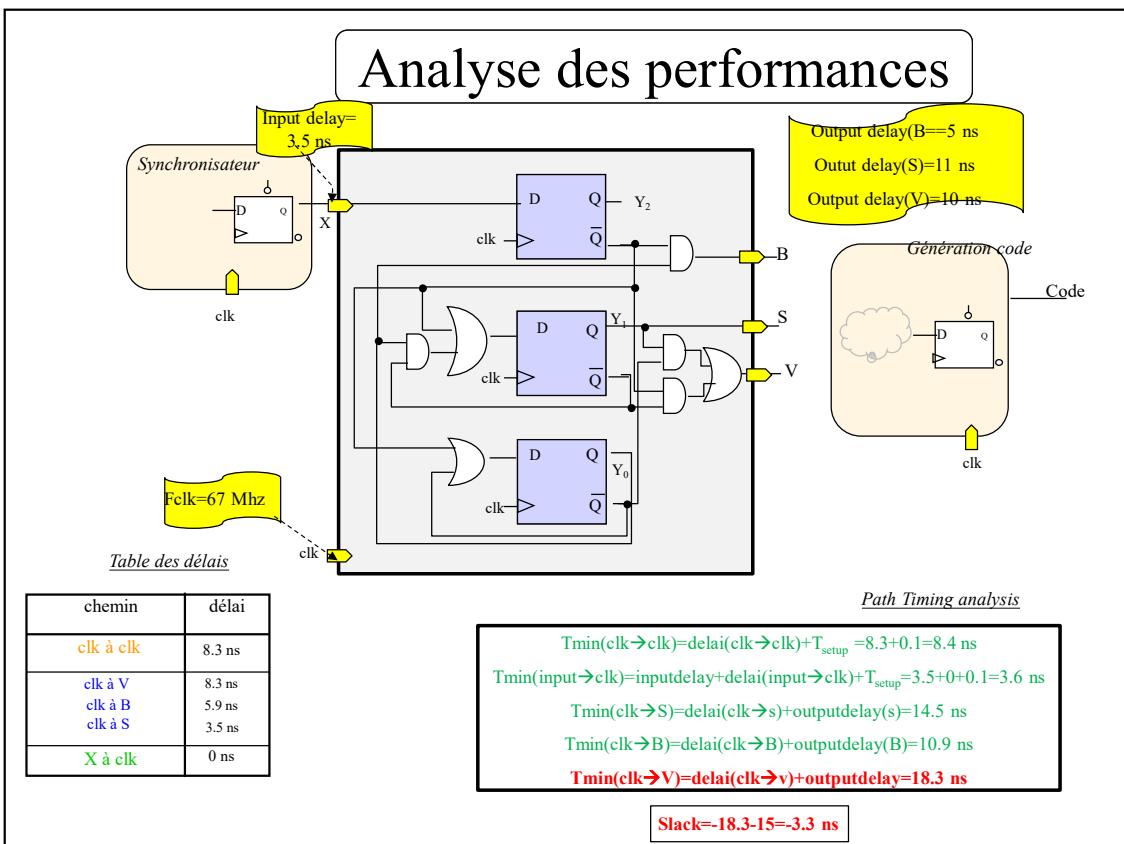
$$\begin{aligned} D_0 &= \bar{Y}_2 + \bar{Y}_0 \\ D_1 &= \bar{Y}_2 + \bar{Y}_1 Y_0 \\ D_2 &= x \end{aligned}$$

Équations des sorties

$$\begin{aligned} V &= Y_1 \bar{Y}_0 + \bar{Y}_2 Y_1 \\ S &= Y_1 \\ B &= \bar{Y}_2 Y_0 \end{aligned}$$

La méthode traditionnelle de synthèse mène aux modèle algébrique à mapper sur les primitives logiques.

Analyse des performances

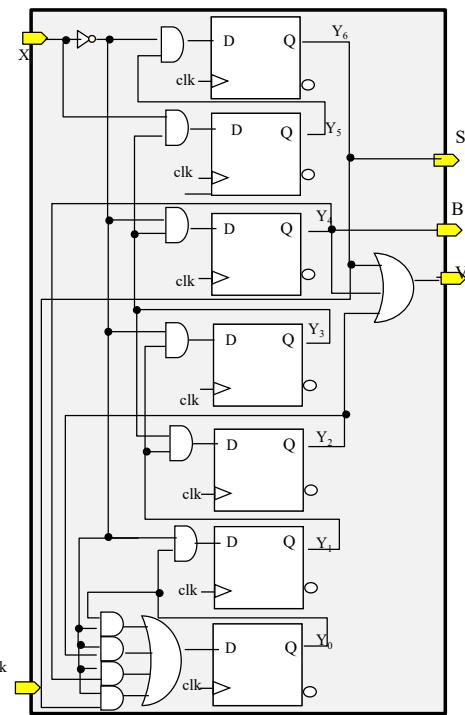
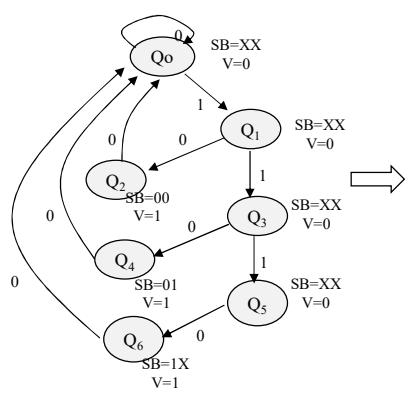


une tentative de synthèse de la FSM à l'aide d'un codage binaire naturel se révèle infructueux car le temps du chemin en sortie est supérieur à la contrainte d'horloge

Codage one hot?

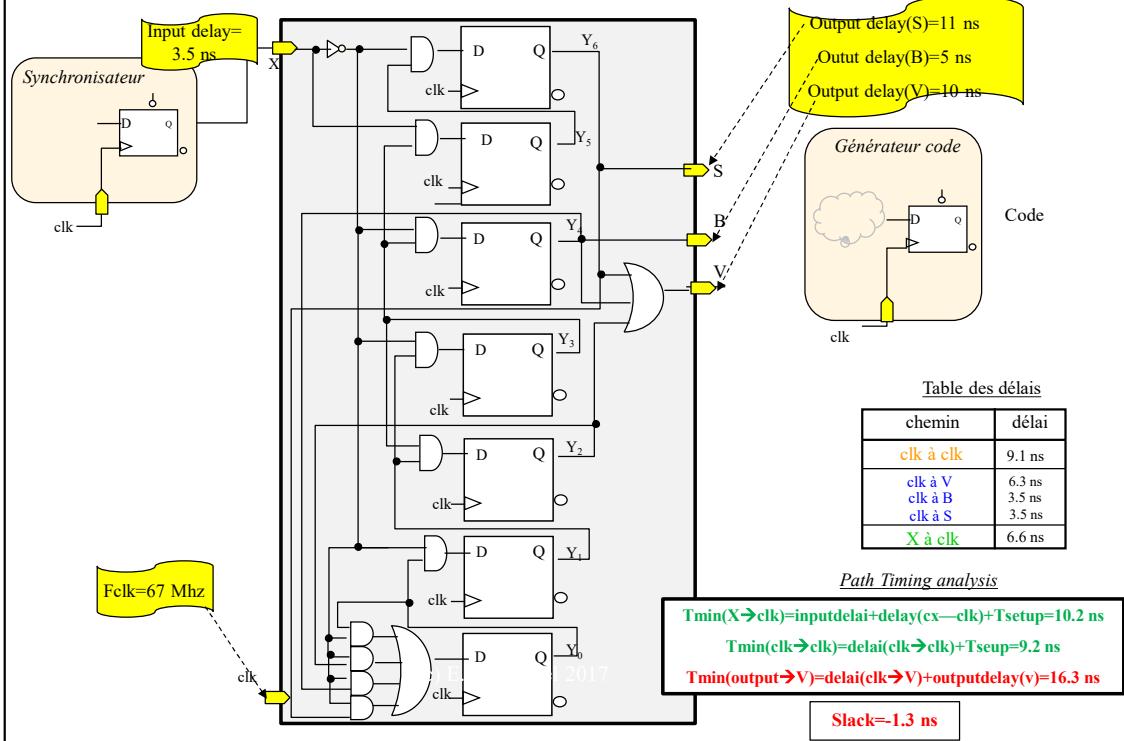
Bloc diagramme

Diagramme des états



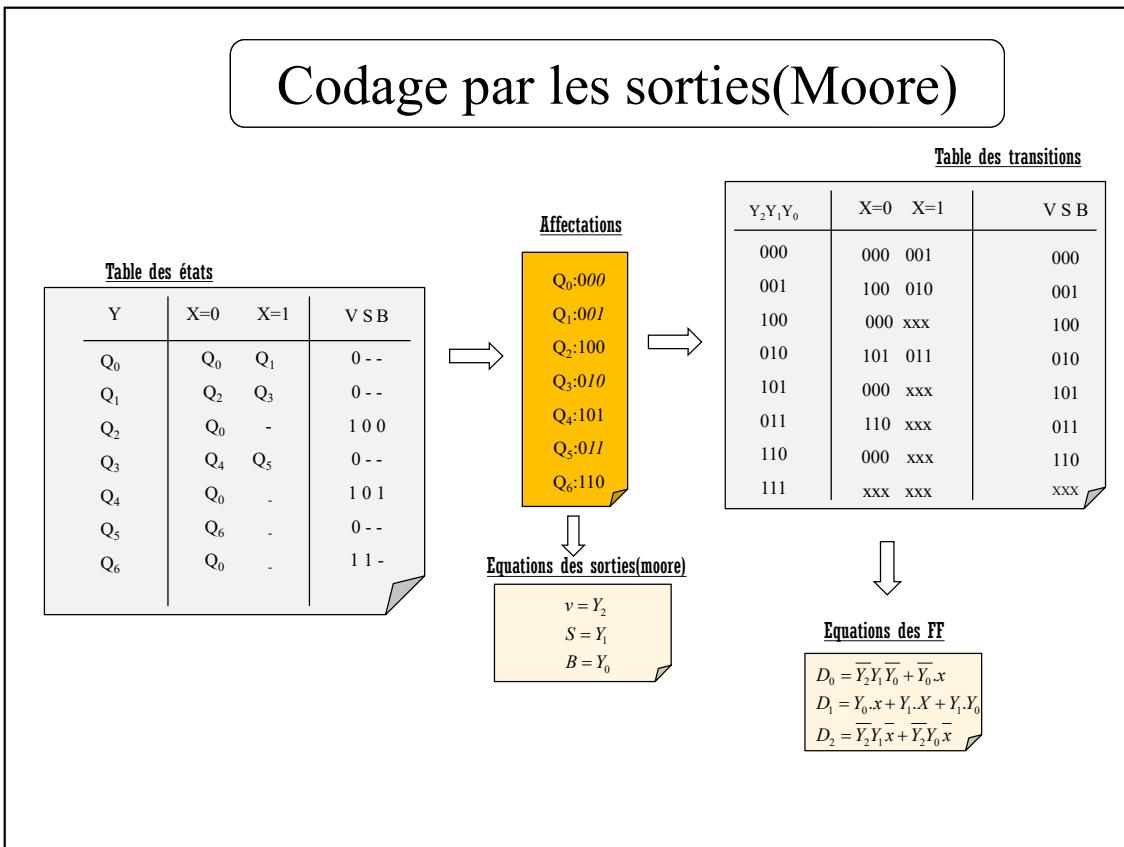
Le codage one hot permet de réduire le temps de décodage de la sortie V mais sans satisfaire la contrainte et au prix d'un accroissement conséquent de la surface.

Analyse des performances



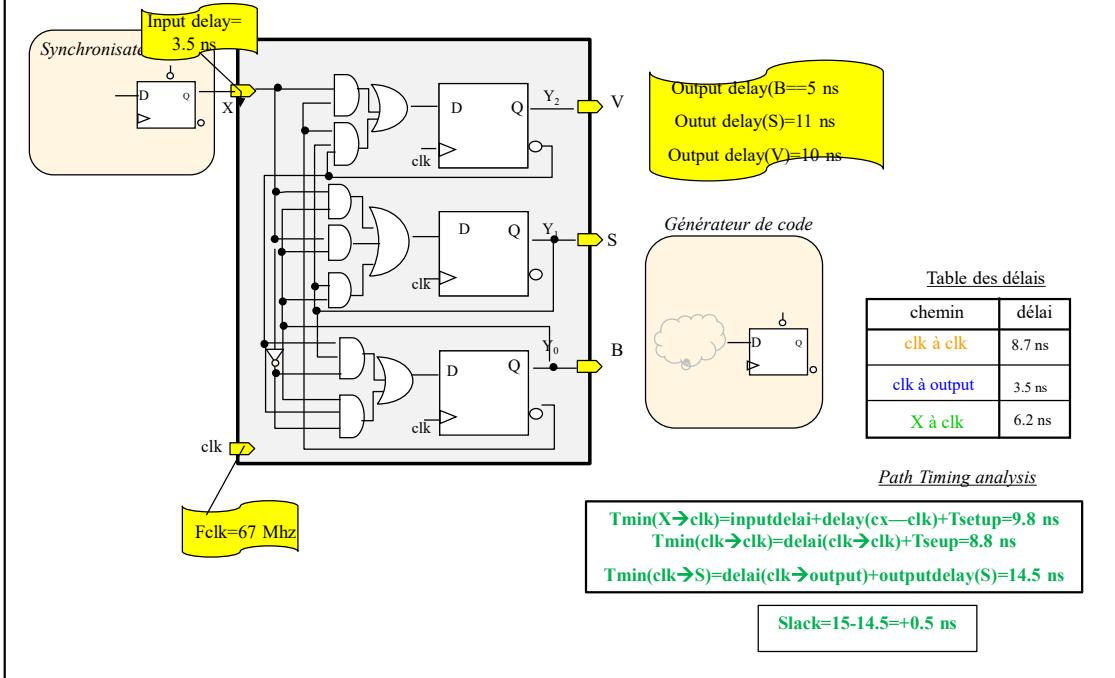
Le codage one hot permet de satisfaire la contrainte sur B mais au prix d'un accroissement conséquent de la surface.

Codage par les sorties(Moore)



On recherche un codage des sorties permettant de faire correspondre une sortie à un bit du vecteur d'état. Ici; tous les sorties peuvent être différencierées grâce aux valeurs non significatives.

Analyse des performances



Le codage de moore permet de résoudre le problème de timing sur la sortie.

Impact du codage binaire naturel

Table d'états

	X	0	1
Y			
Q ₀	Q ₀ Q ₁		
Q ₁	Q ₀ Q ₂		
Q ₂	Q ₂ Q ₃		
Q ₃	Q ₂ Q ₃		

Tables de transition

Y	Y ₁ Y ₀	X		D ₁ D ₀	
		0	1	0	1
0 0	00	11	0 0	1 1	
1 1	00	01	0 0	0 1	
0 1	01	1 0	0 1	1 0	
1 0	01	1 0	0 1	1 0	

Y	Y ₁ Y ₀	X		D ₁ D ₀	
		0	1	0	1
0 0	00	0 0	0 1	0 0	0 1
0 1	01	0 0	1 1	0 0	1 1
1 1	1 1	1 1	1 0	1 1	1 0
1 0	1 0	1 1	1 0	1 1	1 0

Equations

D ₁	D ₀
$D_1 = \bar{Y}_1 x + \bar{Y}_0 x$	$D_0 = \bar{Y}_1 \bar{Y}_0 x + \bar{Y}_1 Y_0 x + Y_1 Y_0 x + Y_1 \bar{Y}_0 x$
10 0 1	10 1 0

D ₁	D ₀
$D_1 = Y_0 x + Y_1$	$D_0 = \bar{Y}_1 x + \bar{Y}_0 x$
0 0 1	0 1 1
0 1 0	1 0 0
1 1 1	1 0 1
1 0 0	0 1 0

Comme le montre l'exemple ci-dessus, le choix du code binaire naturel a un impact important sur l'implémentation finale du système séquentiel. Les états sont définis initialement de manière symbolique, l'affectation d'une combinaison d'état dépendant des contraintes économiques, robustesse, performances pesant sur la réalisation du circuit, cette décision est différée (plusieurs solutions peuvent être expérimentée). Dans l'exemple ci-dessus, les codes étant forcément les mêmes sur les deux colonnes des états Q2 et Q3, en codant ces états courants par des codes adjacents, on a forcément que des 0 ou que des 1 sur les colonnes. On peut alors faire des paquets plus facilement et simplifier en éliminant une ou plusieurs variables d'état dans les termes produits. Un principe directeur semble consister à rechercher un code permettant de regrouper au maximum les 1, 0 et X dans les tables d'excitation pour pouvoir faire des paquets importants de 1 ou 0 et simplifier au maximum les équations.

Nombres de codes binaire naturel

N	n	Nbre de codes total $A_{2^n}^N = \frac{(2^n)!}{(2^n-N)!}$	Nbre de codes ≠ $\frac{A_{2^n}^N}{n! 2^n} = \frac{(2^n-1)!}{n!(2^n-N)!}$
2	1	2	1
3	2	24	3
4	2	24	3
5	3	6720	140
6	3	20160	420
7	3	40320	840
8	3	40320	840
9	4	$4 \cdot 10^9$	10.810.800
..
16	5	$2 \cdot 10^{13}$	$5 \cdot 10^{10}$

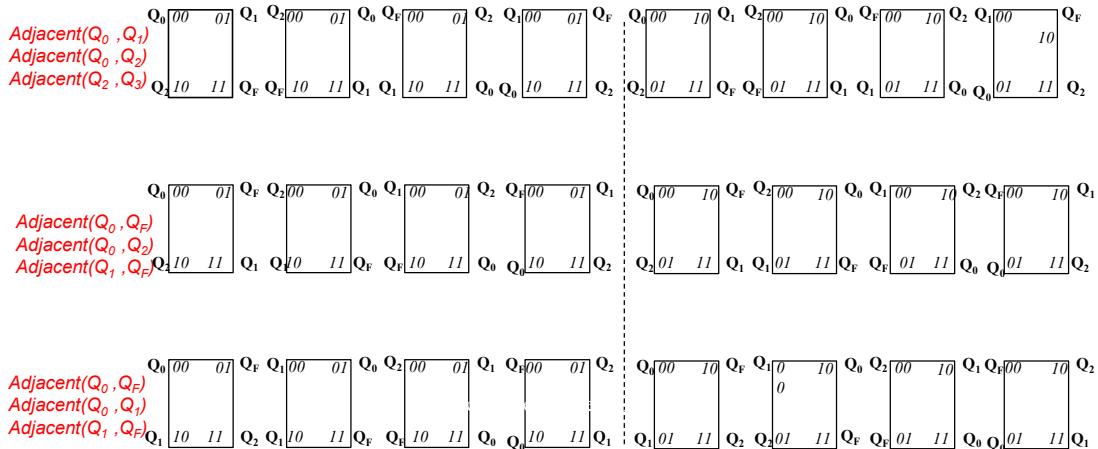
Si N=nombre d'états, n=nombre de bascules, on voit que le nombre total d'affectations possibles croît de manière exponentielle même si le nombre de solution de complexité différente est inférieur à ce nombre. Question: comment déterminer les affectations produisant la solution optimale sans faire une exploration systématique de l'espace des codages possibles?

Illustration

Table du détecteur de séquences

X Y \	0	1
Q₀	Q ₀ /0	Q ₁ /0
Q₁	Q ₀ /0	Q ₂ /0
Q₂	Q ₀ /0	Q _F /1
Q_F	Q _F /0	Q _F /0

24 affectations
3 codes différents



le codage binaire naturel répond essentiellement à un critère de minimisation de la logique séquentielle (nombre de bascules). 4 états \Rightarrow 2 bascules. Il y a 24 codages possibles pour cette machine d'état. En réalité, la complexité de l'implémentation va dépendre de l'adjacence des états. Ici, il a 3 possibilités d'adjacences (codes pour lesquels un seul bit change) et 8 combinaisons (permutations) pour chaque codage adjacent ($00 \rightarrow 01, 00 \rightarrow 10, 11 \rightarrow 01, 11 \rightarrow 10$)

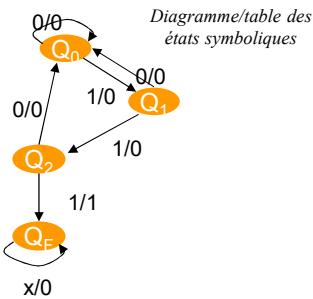
Règles heuristiques

-
- I.** (a) Chercher les lignes de la table d'états qui ont des états suivants identiques.
Coder ces lignes par des codes adjacents. Si possible, les états suivants de ces lignes recevront des codes adjacents
- (b) Chercher les lignes qui ont les mêmes états suivants mais dans un ordre différent. Choisir des codes adjacents pour ces lignes si on peut coder les états suivants par des codes adjacents.
- (c) Des lignes avec quelques états suivants identiques recevront des codes adjacents.
On considère tout d'abord les lignes ayant le plus de colonnes identiques.
- II.** Les états suivants d'une ligne recevront des codes adjacents.
- III.** Les codages sont tels qu'ils simplifient les tables de sortie.

ce sont des règles (sorte de manuel d'emploi) qui, lorsque plusieurs affectations sont possibles (dénommé ensemble de conflits), permettent de sélectionner une affectation de codes aboutissant à une bonne solution, proche de la solution optimale en terme de ressources pour un temps de recherche raisonnable (Compromis qualité/temps de recherche) Le principe du codage est d'essayer de rendre adjacent au maximum des états courant qui évoluent vers des états suivants identiques ou partiellement identiques, ces adjacences étant de nature à engendrer par la suite des simplifications au niveau des équations d'entrée des éléments mémoires.

Les règles s'appliquent avec une priorité décroissante en recherchant d'abord des états suivant strictement identiques quelque soit l'entrée, puis partiellement identiques, etc. Les règles II et III peuvent ensuite souvent s'appliquer sur les résultats de la règle I pour régler les choix dans l'ensemble de conflits d'affectation trouvés précédemment.

Application règle I au détecteur



REGLE Ia Chercher les lignes de la table d'états qui ont des états suivants identiques. Coder ces lignes par des codes adjacents. Si possible, les états suivants de ces lignes recevront des codes adjacents

Non applicable

	X	0	1
Y			
Q_0		$Q_0/0$	$Q_1/0$
Q_1		$Q_0/0$	$Q_2/0$
Q_2		$Q_0/0$	$Q_F/1$
Q_F		$Q_F/0$	$Q_F/0$

REGLE Ib Chercher les lignes qui ont les mêmes états suivants mais dans un ordre différent. Choisir des codes adjacents pour ces lignes si on peut coder les états suivants par des codes adjacents.

Non applicable

Adjacences

REGLE Ic Des lignes avec quelques états suivants identiques recevront des codes adjacents. On considère tout d'abord les lignes ayant le plus de colonnes identiques.

Q_0 et Q_1
 Q_0 et Q_2
 Q_1 et Q_2
 Q_2 et Q_f

Application au détecteur: L'affectation utilisée lors de l'étape de synthèse (codage séquentiel) se révèle être désigné par les règles heuristiques comme étant la meilleure.

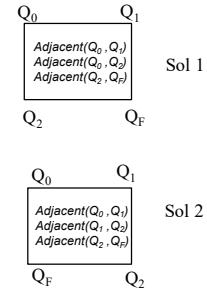
Application règle II au détecteur

Table des états symboliques

	X	0	1
Y			
Q ₀	Q ₀ /0	Q ₁ /0	
Q ₁	Q ₀ /0	Q ₂ /0	
Q ₂	Q ₀ /0	Q _F /1	
Q _F	Q _F /0	Q _F /0	

Adjacences souhaitées
(issues règle I)

Q₀ et Q₁
Q₀ et Q₂
Q₁ et Q₂
Q₂ et Q_f



REGLE II Les états suivants d'une ligne recevront des codes adjacents.

Adjacences (règles II)

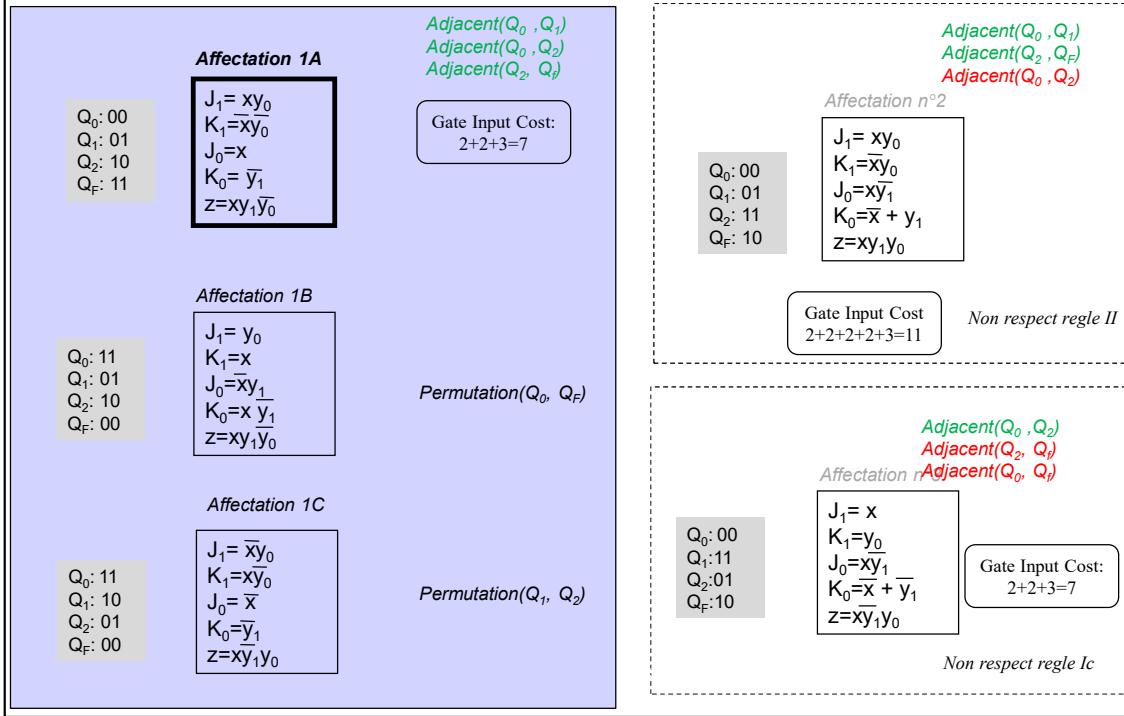
Q₀ et Q₁
Q₀ et Q₂

Affectations finales

Q₀: 00
Q₁: 01
Q₂: 10
Q_F: 11

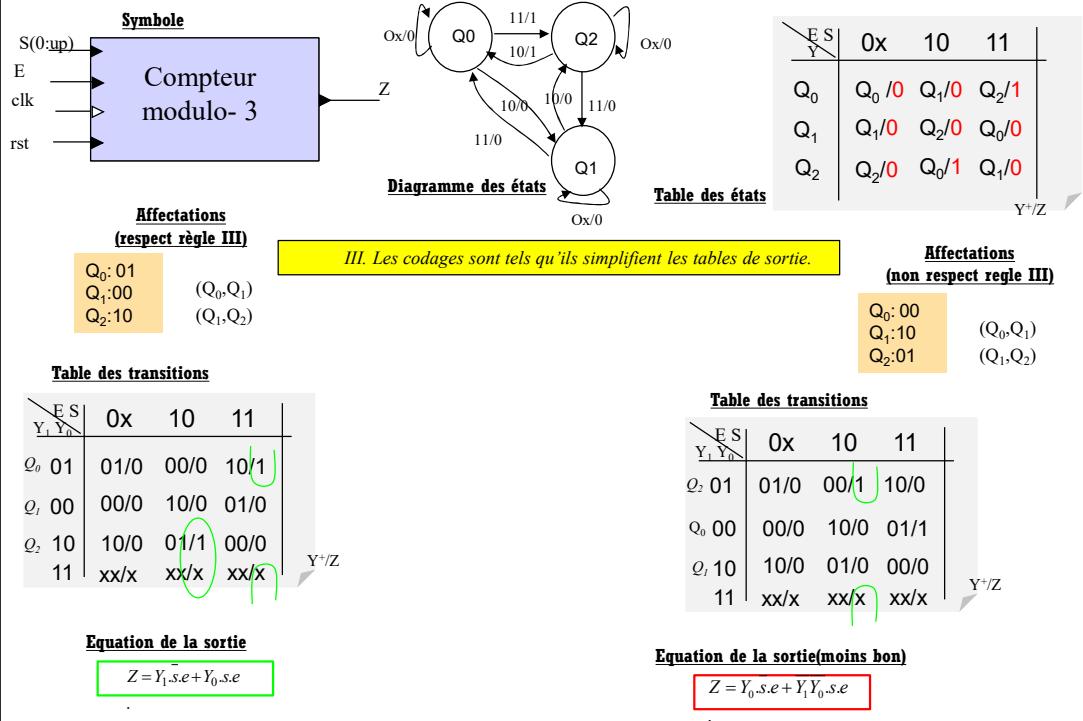
En présence de conflits ou de degrés de libertés restant dans les affectations, la règle prend en compte les états suivants et demande de coder ces états suivants avec un code adjacent si possible lorsqu'ils sont état suivants d'un même état courant. On retient finalement la solution 1

Equations du détecteur de séquence



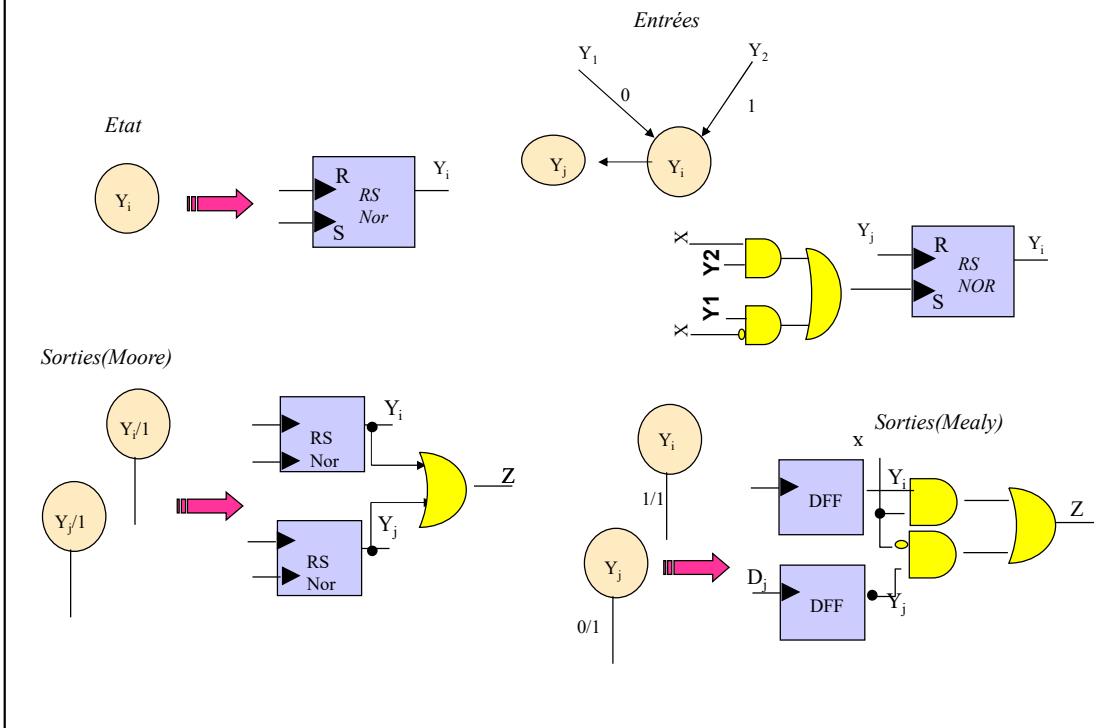
Si on observe 3 affectations possibles, on peut noter que les solutions 1 et 2 ont a priori le même coût en portes (selon la métrique GIC). La solution 1 a l'avantage de ne requérir que des ET. La solution 3 est la plus coûteuse. le choix de l'affectation est indépendant du codage particulier de l'état et des permutations donnent des résultats de complexité comparable. Par conséquent, seul compte le respect des contraintes d'adjacence.

Illustration de la règle III



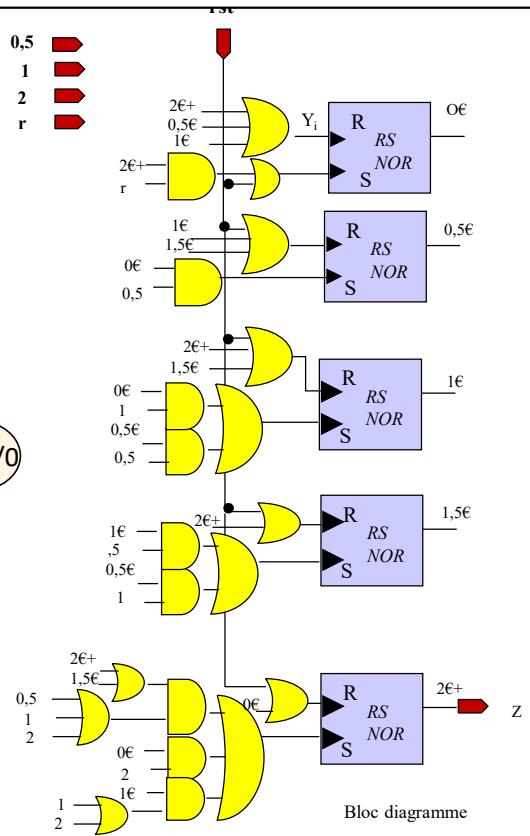
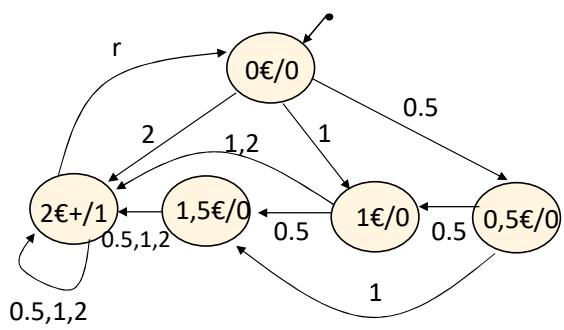
Dans ce circuit, la règle Ib s'applique pour tous les états (et donc la règle II) avec un conflit en final car les 3 états ne peuvent pas être tous adjacents). On recherche donc, conformément à la règle III, les états qui possèdent le plus de sorties communes pour des mêmes combinaisons d'entrée, ce qui donne les paires Q0,Q1 (entrées 0x et 10) et Q1,Q2 (entrées 0x et 11). Si on avait mis, Q2 en adjacence avec Q0, la solution aurait été plus complexe car la sortie à 1 pour la colonne 10 n'aurait pas pu se combiner avec le x de l'état indéterminé.

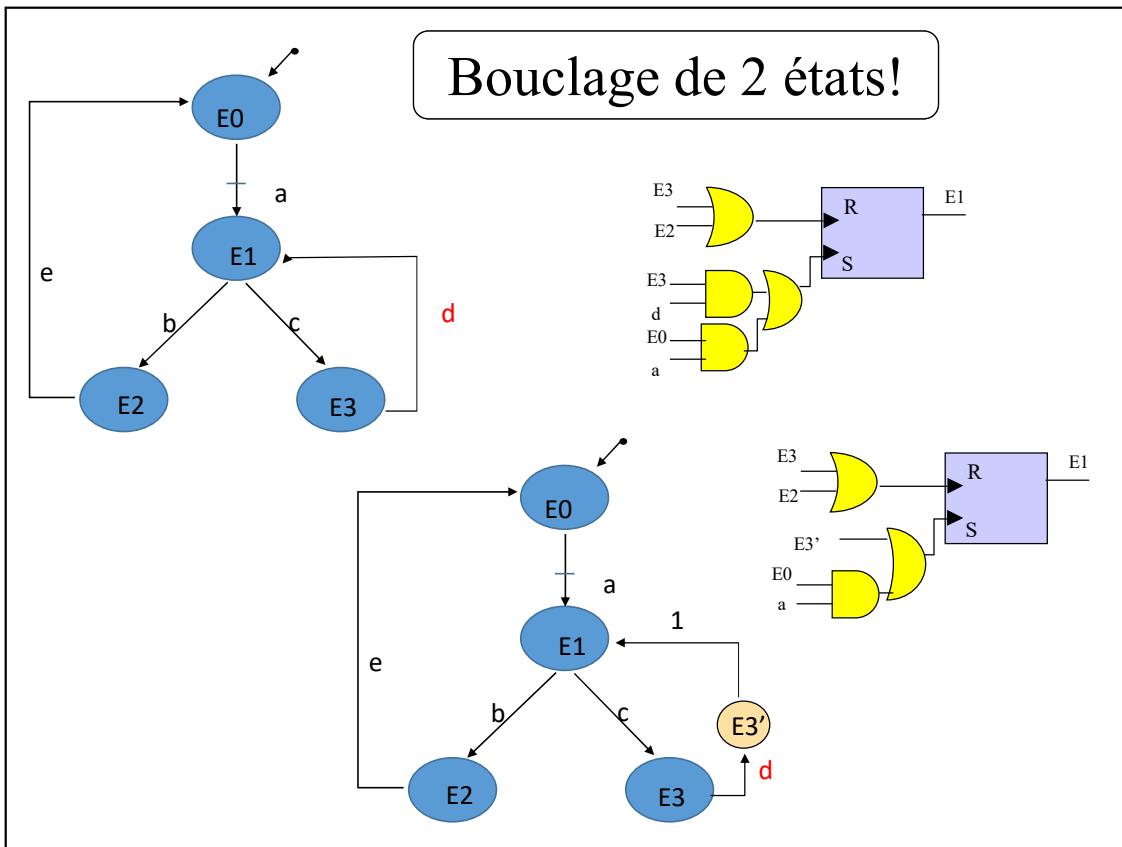
Codage one hot (Système asynchrone pulsé)



Attention à ne pas mettre les conditions de transition sur l'entrée R. Elles ne doivent figurer que sur l'entrée S

Illustration





Dans le cas d'un bouclage de 2 états, on peut se retrouver avec une FF RS dont les 2 entrées sont à 1 simultanément (strictement interdit). La solution consiste à introduire un état intermédiaire avant de repasser sur E1.

Automates à états finis

A. Catégorie de FSM

B. Analyse des FSM

C. Synthèse des FSM

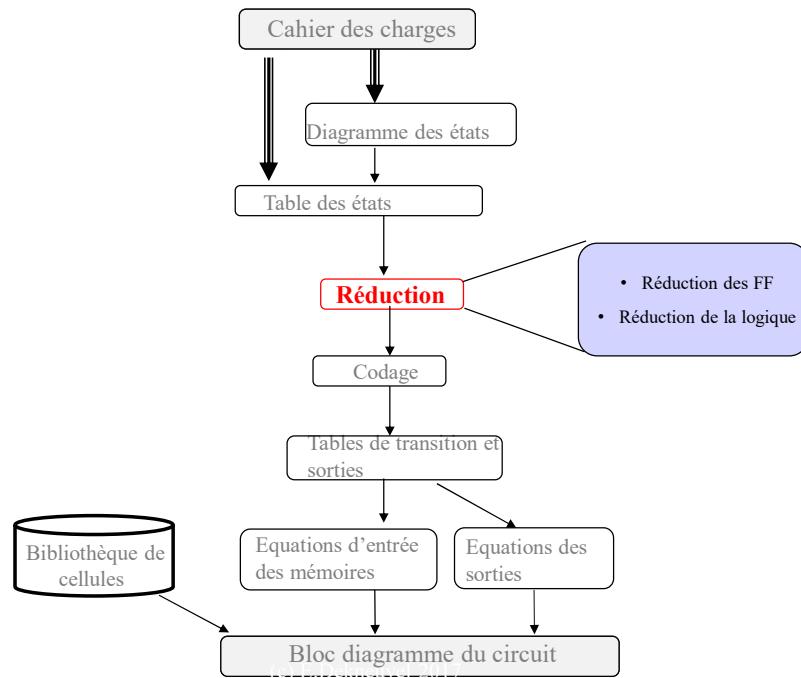
D. Codage des états

E. Réduction des machines séquentielles

(c) E.Dekneuvel 2021

Soit une machine séquentielle S définie par sa table ou son diagramme d'état), peut-on trouver une machine séquentielle T plus simple i.e avec un nombre d'état réduit?

Réduction des systèmes séquentiels

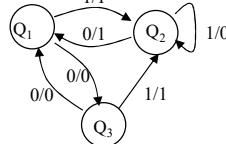


Deux méthodes possibles:

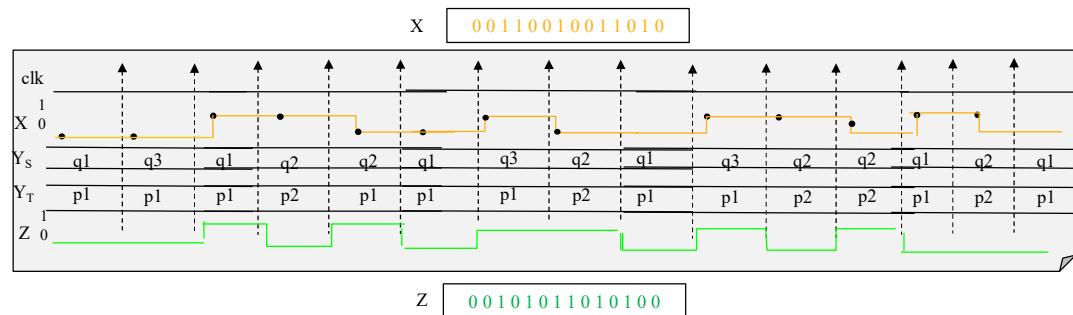
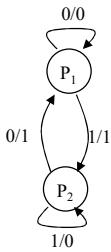
1. Approche des classes d'équivalence: ne fonctionne que sur les systèmes complètement spécifiés
2. Approche par la table des implications: fonctionne sur les systèmes complètement spécifiés et incomplètement spécifiés.

Machines équivalentes

<i>Machine S</i>	X	0	1
Y	q ₁	q ₃ /0 q ₂ /1	q ₂ /0
	q ₂	q ₁ /1 q ₂ /0	q ₁ /0 q ₂ /1
	q ₃	q ₁ /0 q ₂ /1	



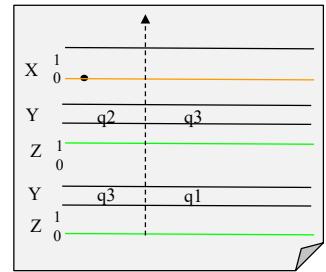
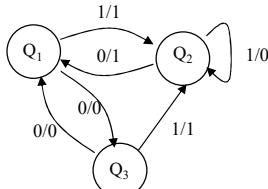
<i>Machine T</i>	X	0	1
Y	p ₁	p ₁ /0 p ₂ /1	p ₂ /0
	p ₂	p ₁ /1 p ₂ /0	



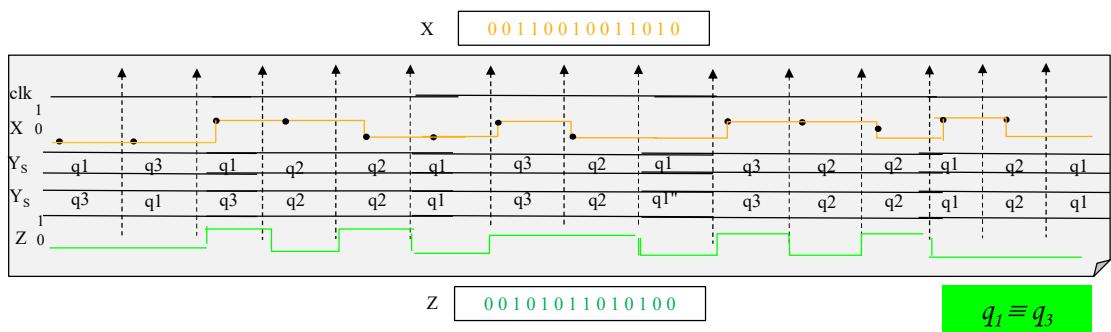
Quelque soit la séquence d'entrée appliquée, les sorties des machines évoluent de manière identiques. Les machines sont donc équivalentes mais la machine T est plus économique à matérialiser.

Etats indissociables

	X	0	1
Y	q_1	$q_3/0$	$q_2/1$
q_1	$q_1/1$	$q_1/0$	$q_2/0$
q_3	$q_1/0$	$q_2/1$	



$$q_2 \not\equiv q_3$$



On voit ici que q_2 et q_3 sont 1-dissociables alors que q_1 et q_3 sont indissociables.

Relations états indissociables

Deux états p et q d'une machine séquentielle sont dits *états indissociables* ou *états équivalents* ($p \equiv q$), si la séquence de sortie est identique pour toute séquence d'entrée appliquée à p et q .

Réflexive

$$\forall x \in E, x \mathcal{R} x$$

\leq
Est-de-la-même-casse-que (minuscule ou majuscule)
a-meme-couleur-d'yeux-ou-de-cheveux-que

Symétrique

$$\forall x, y \in E, x \mathcal{R} y \Rightarrow y \mathcal{R} x$$

$\left\{ \begin{array}{l} \text{i} \text{st-de-la-même-casse-que} \\ \text{a-meme-couleur-d'yeux-ou-de-cheveux-que} \end{array} \right.$

Transitive

$$\forall x, y, z \in E, x \mathcal{R} y \text{ et } y \mathcal{R} z \Rightarrow x \mathcal{R} z$$

$\left\{ \begin{array}{l} \text{i} \text{st-de-la-même-casse-que} \\ \downarrow \\ \text{Classes d'équivalence} \\ \mathcal{P}_1 = \{a, b, c, d\} \quad \mathcal{P}_2 = \{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}\} \end{array} \right.$

La relation « est indissociable » est une relation d'équivalence. Elle est définie sur un ensemble d'éléments E (les états de la machine) avec les propriétés de réflexivité, symétrie, transitivité. Les classes d'équivalence permettent ensuite de partitionner l'ensemble des éléments en sous-ensembles d'éléments disjoints

Théorème du partitionnement en classes d'équivalences

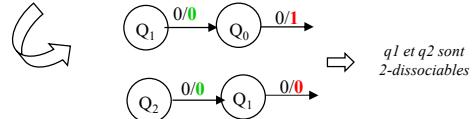
$$\begin{aligned} \forall \chi \\ 1. \mathcal{G}(p,\chi) &= \mathcal{G}(q,\chi) \\ 2. \mathcal{F}(p,\chi) &\stackrel{\Delta}{=} \mathcal{F}(q,\chi) \end{aligned}$$

Table d'états

$Q \setminus X$	0	1
Q_0	$Q_0/1$	$Q_4/0$
Q_1	$Q_0/0$	$Q_4/0$
Q_2	$Q_1/0$	$Q_5/0$
Q_3	$Q_1/0$	$Q_5/0$
Q_4	$Q_2/0$	$Q_6/1$
Q_5	$Q_2/0$	$Q_6/1$
Q_6	$Q_3/0$	$Q_7/1$
Q_7	$Q_3/0$	$Q_7/1$

Partition initiale

Classes	a	b			c			
Etats	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
Cl. Suivante $x=0 \quad x=1$	a c	a c (b)	c b	c b	c b	c b	c b	c

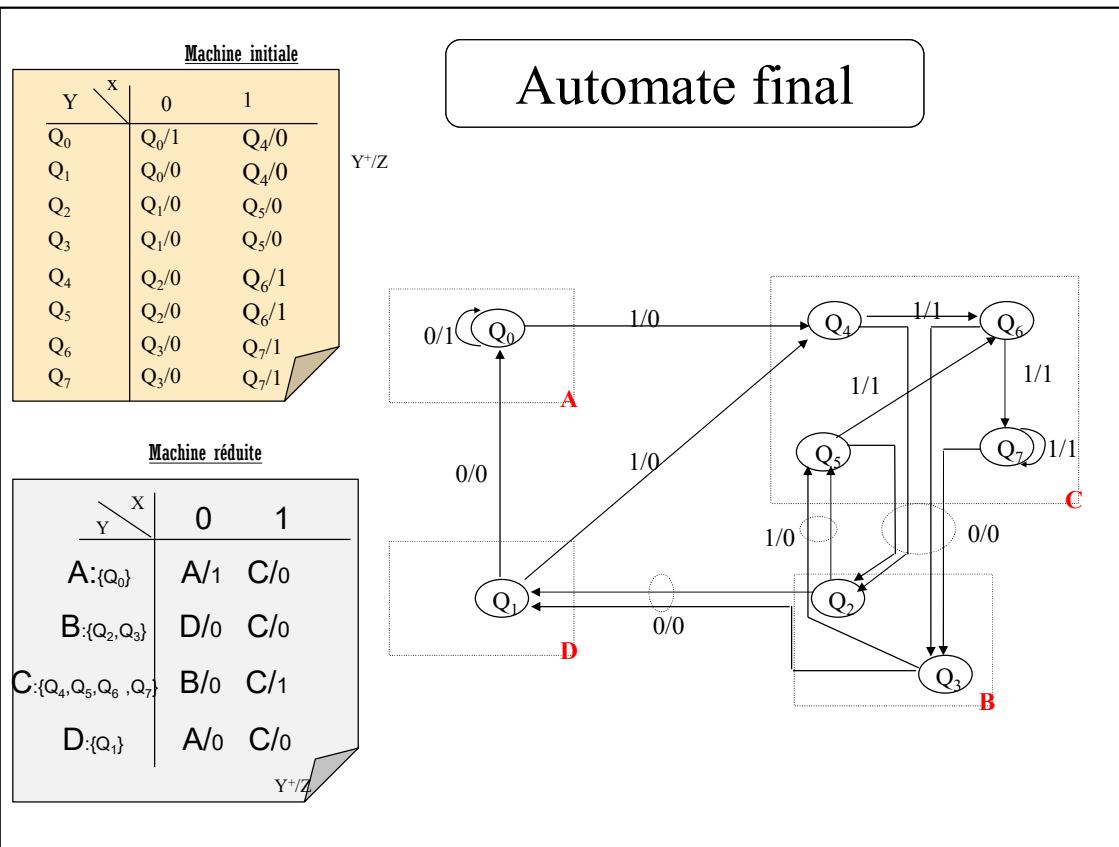


q_1 et q_2 sont
2-dissociables

Repartitionnement

Classes	a	b		c			d
Etats	Q_0	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
Cl. Suiv.	a c	d c	d e	b c	b c	b c	a c

Le théorème fournit un moyen de partitionner la table des états en classes d'équivalences, plutôt que de vérifier l'équivalence des états 2 par 2 comme précédemment. Le principe consiste à former une partition de l'ensemble initial des états telle que les sorties soient identiques pour les combinaisons d'états courants (et d'entrées s'il s'agit d'une machine de Mealy). On vérifie ensuite si les états suivants de chaque élément de la partition tombent tous dans la même partition. Si tel n'est pas le cas, les états ne sont pas indissociables (équivalents) et ne peuvent figurer dans la même partition. On réitère jusqu'à ce que la relation n°2 soit satisfaite. Ici, on voit que Q_1 et Q_2 sont 2-dissociables (séquence 00).



On notera que la machine minimale obtenue est unique. On voit bien qu'à chaque fois qu'on quitte un état de la classe, on va toujours vers un état suivant appartenant à la même classe, quelque soit l'état.