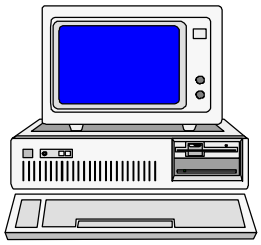


# Logique séquentielle
# La méthode d'huffman

LogHuff

## A) Introduction.

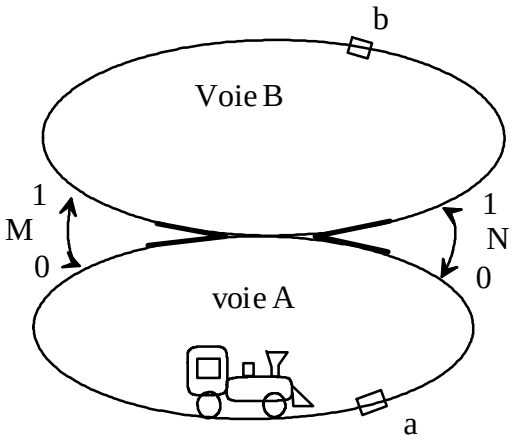
La méthode d'huffman est une méthode de synthèse des systèmes séquentiels qui oblige à faire une étude complète du système à réaliser et fournit un moyen systématique de réalisation avec un minimum de variables internes (bascules).

Cette méthode a beaucoup perdu de son importance depuis l'apparition des microprocesseurs et d'autre méthode d'analyse (GRAFCET). Cependant pour les systèmes séquentiels rapides (les systèmes micro sont lents), elle est utile pour la conception de petits sous ensembles, lorsque la solution n'est pas apparente, et que l'emploi des autres méthodes serait trop lourd.

## B) Analyse et synthèse à partir d'un exemple.

Recherche d'un système séquentiel pour automatisme ferroviaire (modèle réduit).

On désire que la locomotive passe 2 fois sur la voie A puis 1 fois sur la voie B, et recommence (2 fois sur la voie A etc. ...).




2 aiguillages M et N permettent de passer sur les voies A ou B

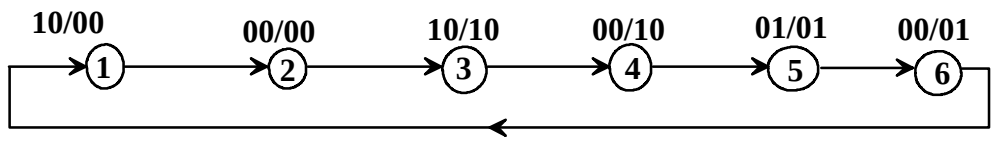
Si les actionneurs M et N sont à 1, les aiguillages s'orientent vers la voie B. Si M = N= 0 => voie A.

2 capteurs a et b délivrent un 1 lorsque la locomotive passe dessus.

### B-1) Représentation par un diagramme de fluence.

On représente sur un diagramme orienté les étapes numérotées de la séquence en indiquant les états logiques des entrées et sorties. Ici ab/MN.

**Diagramme de fluence.**




**Remarque importante:** Dans la méthode d'huffman, on suppose que 2 entrées ne peuvent varier simultanément (les transitions d'entrée (01)->(10) sont interdites).

Dans l'exemple présent, cette condition est toujours remplie (s'il n'y a qu'une locomotive).

**B-2) Ecriture de la matrice des phases.**

La méthode d'Huffman commence par l'écriture d'un tableau appelé matrice des phases, équivalent au diagramme de fluence. Il comporte une colonne pour chaque combinaison des entrées (codage en binaire réfléchi) et une colonne pour chaque sortie.

On place un état stable par ligne (phase stable = étape), avec les valeurs des sorties pour cette étape. L'état stable est représenté par le numéro de l'étape entouré. On indique alors sur la ligne, les transitions possibles par le numéro de la phase stable d'arrivée (numéro de l'étape non entouré).

**Matrice des phases**

| a b 00 | 01 | 11 | 10 | M | N |
|---|---|---|---|---|---|
| 2 | | | (1) | 0 | 0 |
| (2) | | | 3 | 0 | 0 |
| 4 | | | (3) | 1 | 0 |
| (4) | 5 | | | 1 | 0 |
| 6 | (5) | | | 0 | 1 |
| (6) | | | 1 | 0 | 1 |

**Explication:**

**la 1ere ligne indique l'étape 1 ab=10 (locomotive en a), les sorties sont MN=00, et la seule transition possible est ab=00 donc passage à l'étape 2**

**états stables (entourés)**

**états instables (non entourés) ils indiquent l'étape suivante.**

**Chaque ligne correspond à une étape (état stable).**

On constate que le système est bien séquentiel (et non combinatoire) à la présence dans une même colonne, de plusieurs phases stables (ex: pour ab=00, on trouve les étapes N°2, N°4 et N°6, alors que l'état des sorties M et N diffèrent. Il est donc impossible d'écrire une équation logique de M et N en fonction de a et b).

**Remarque:** Il peut se produire que l'on ait introduit plusieurs phases stables équivalentes, sans s'en rendre compte au cours de l'analyse. Deux phases seront équivalentes si:

- Elles sont dans la même colonne (même vecteur d'entrée).
- Les états des sorties sont les mêmes.
- Les mêmes transitions d'entrées mènent aux mêmes phases ou à des phases équivalentes.

Il faut détecter ces phases équivalentes afin de n'en garder qu'une (cela permet de limiter le nombre d'étapes et donc de simplifier d'autant le système séquentiel). Dans l'exemple proposé, il n'y a pas de phases équivalentes (les étapes dans les mêmes colonnes n'ont pas les mêmes états en sortie.).

**B-3) Ecriture de la matrice réduite.**

L'écriture de la matrice des phases montre que les phases stables dans une même colonne ne peuvent pas être différenciées par les variables d'entrées a et b. Il faudra donc ajouter d'autres variables appelées variables internes, pour pouvoir coder les lignes et donc indiquer l'étape de la séquence (étape = ligne de la matrice des phases). Les variables internes seront réalisées à l'aide d'éléments comportant un état mémoire, afin de garder le numéro de l'étape dans laquelle on se trouve à un instant donné. Il est donc possible d'utiliser les bascules RS, D ou JK. Si l'on prend n bascules, il est donc possible de coder une matrice de $2^n$ lignes. On a donc intérêt à limiter le nombre de ligne.

Il est possible de réduire le nombre de lignes par fusion.

**Conditions de fusionnement de 2 ou plusieurs lignes:**

- Les états stables doivent être dans des colonnes différentes.
- Les transitions doivent être compatibles (on ne peut mettre deux numéros différents dans une même case).
- On ne tient pas compte de l'état des sorties (elles peuvent être différentes).

Ici il est possible de fusionner la ligne 3 avec la ligne 4, et de fusionner les lignes 5 et 6 (pas de conflit entre des chiffres différents dans chaque colonne).

**Matrice réduite**

| a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| | 2 | | | (1) |
| | (2) | | | 3 |
| | (4) | 5 | | (3) |
| | (6) | (5) | | 1 |

**Dans cette matrice les sorties ne doivent pas être représentées**

**lignes 3 et 4 fusionnées**

**lignes 5 et 6 fusionnées**

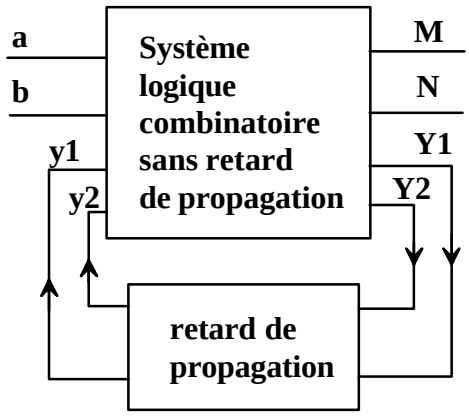
Lorsque deux lignes fusionnent, les états stables sont représentés en priorité, certains états instables ne sont donc plus nécessaires, car ils correspondent à l'état stable de même numéro.

**B-4) Codage des lignes.**

Il est indispensable d'ajouter des variables internes pour différentier les lignes. Ici la matrice réduite comporte 4 lignes, il faut donc deux variables internes pour coder ces 4 lignes. Appelons y1 et y2 les deux variables internes supplémentaires. Il est alors possible en attribuant à chaque ligne, une combinaison de y1 et y2, de traduire chaque étape par une relation en logique combinatoire de a, b, y1 et y2.

**Codage des lignes**

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | 2 | | | (1) |
| 0 1 | (2) | | | 3 |
| 1 1 | (4) | 5 | | (3) |
| 1 0 | (6) | (5) | | 1 |




On utilisera des lettres minuscules pour les variables d'entrée, et des majuscules pour les grandeurs de sortie. Nous aurons donc ici 4 variables d'entrée a, b, y1, y2, et il faudra réaliser les 4 grandeurs de sortie M, N, Y1 et Y2.

**Remarque:** y1 et y2 correspondent respectivement à Y1 et Y2, mais aux temps de propagations près des portes logiques et/ou des bascules.

**B-5) Ecriture de la matrice d'excitation et de la matrice de sortie.**

Il est alors indispensable de choisir le mode de réalisation de l'ensemble du système.

- a) Réalisation totale en logique combinatoire (possible s'il n'y a que quelques variables internes, en principe 3 maximum).
- b) Réalisation à l'aide de bascules RS (si asynchrone), D ou JK (si système synchrone ou asynchrone synchronisé).
- c) Réalisation à l'aide de composants programmables (PROMs, ROMs, EPROMs, PALs combinatoire, pour les systèmes asynchrones, ou les mêmes composants associés à des bascules pour les systèmes asynchrones synchronisés, ou même des PALs séquentiels).

**B-5-a) Réalisation en logique combinatoire.**

**Matrice réduite**

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | 2 | | | (1) |
| 0 1 | (2) | | | 3 |
| 1 1 | (4) | 5 | | (3) |
| 1 0 | (6) | (5) | | 1 |

**Pour la construction des matrices d'excitation et des matrices de sortie, voir Explication**

**Matrices d'excitation de Y1 et Y2**

Y1

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | 0 | | | 0 |
| 0 1 | 0 | | | 1 |
| 1 1 | 1 | 1 | | 1 |
| 1 0 | 1 | 1 | | 0 |

Y2

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | 1 | | | 0 |
| 0 1 | 1 | | | 1 |
| 1 1 | 1 | 0 | | 1 |
| 1 0 | 0 | 0 | | 0 |

**Matrices des sorties**

M

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | X | | | 0 |
| 0 1 | 0 | | | X |
| 1 1 | 1 | X | | 1 |
| 1 0 | 0 | 0 | | X |

N

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | X | | | 0 |
| 0 1 | 0 | | | X |
| 1 1 | 0 | X | | 0 |
| 1 0 | 1 | 1 | | X |

**Explication sur la construction des matrices d'excitation:**

Pour construire la matrice d'excitation de Y1 (tableau de Karnaugh), il faut utiliser la matrice réduite. On procédera de même pour Y2 en remplaçant par les indices 2.

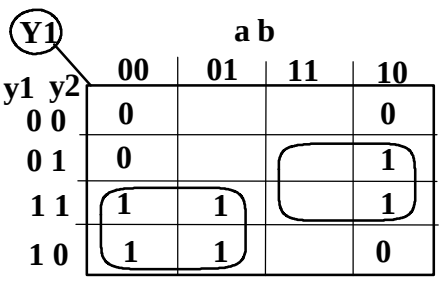
❑ Pour chaque état stable Y1 aura la même valeur que la variable d'entrée y1 que l'on voit sur la ligne de l'état stable (ex: pour l'état stable 1 {1ère ligne}, y1 vaut 0, donc Y1 vaudra également 0).

❑ **Pour les états instables** (états transitoires), on prend comme valeur de Y1, la valeur de l'entrée y1 **de l'état stable .(Ex**: pour l'état instable 3 sur la 2 ième ligne, y1 vaut 1 pour l'état stable 3, donc l'état instable prend la même valeur, soit 1.

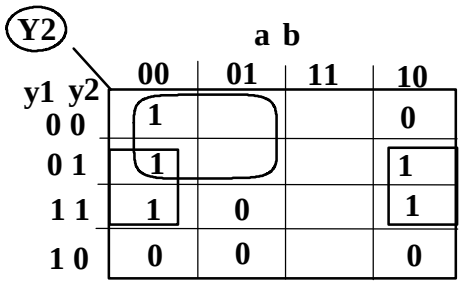
Il ne reste plus qu'à déterminer les équations de Y1 et Y2 (tableau de Karnaugh).

**Explication sur la construction des matrices de sortie:**

**Equations de Y1 et Y2**




**Y1=y1.$\overline{a}$ +y2.a**



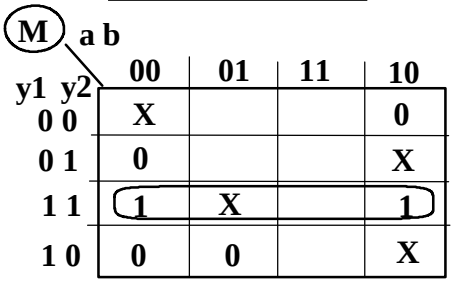

**Y2=$\overline{y1}$.$\overline{a}$ + y2.$\overline{b}$**

Les matrices de sortie sont construites à partir de la matrice des phases et de la matrice réduite. On écrit une matrice (tableau de Karnaugh) pour chaque sortie. Prenons le cas de la sortie M en exemple.

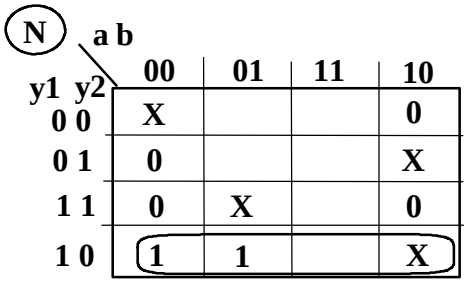
❑ Pour les états stables il faut prendre l'état spécifié dans la matrice des phases.

❑ Pour les états instables (ils ne durent que pendant des temps très brefs = temps de propagations) l'état peut en général être quelconque, mais parfois pour éviter des problèmes d'aléas ou d'états transitoires parasites, on préfère imposer soit l'état de l'étape stable de départ, ou celui de l'étape stable d'arrivée.

**Matrices de sortie**




**M=y1.y2**

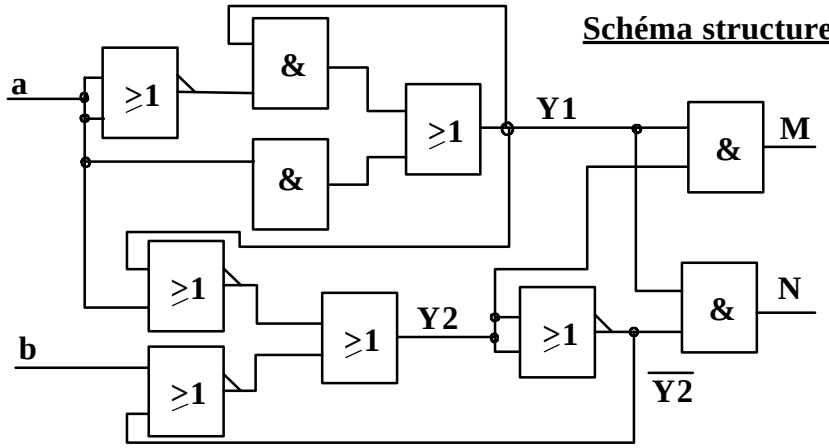



**N=y1.$\overline{y2}$**

Le système peut donc être réalisé après avoir simplifié les équations à l'aide des règles de l'algèbre de Boole. Ex: en transformant l'équation de Y2 de la façon suivante.

$Y2 = \overline{(y1 + a)} + \overline{(\overline{y2} + b)}$ Il faut alors 4 portes NOR à 2 entrées, 4 portes ET à 2 entrées et 2 portes OU à 2 entrées, soit 3 boîtiers au total.

**Schéma structurel**




**B-5-a) Réalisation à l'aide de bascules.**

On écrit à partir de la matrice réduite, une matrice de transition (TY1 et TY2) pour chaque variable interne (Y1 et Y2).

**Réalisation à l'aide de bascules**

A partir de la matrice réduite pour chaque état (stable et instable) on détermine les transitions que doivent subir Y1 et Y2 pour arriver à la phase suivante.

**Matrice réduite**

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | 2 | | | (1) |
| 0 1 | (2) | | | 3 |
| 1 1 | (4) | 5 | | (3) |
| 1 0 | (6) | (5) | | 1 |

**Matrices de transition de Y1 et Y2**

**TY1**

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | t00 | | | t00 |
| 0 1 | t00 | | | t01 |
| 1 1 | t11 | t11 | | t11 |
| 1 0 | t11 | t11 | | t10 |

**TY2**

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | t01 | | | t00 |
| 0 1 | t11 | | | t11 |
| 1 1 | t11 | t10 | | t11 |
| 1 0 | t00 | t00 | | t00 |

En utilisant les tables de transitions des bascules, on obtient alors une matrice d'excitation et les équations des entrées de la bascule. Dans l'exemple, le système étant asynchrone, on choisit d'utiliser des bascules RS. Il suffit alors d'utiliser la table des transitions de la bascule RS, pour obtenir le tableau de Karnaugh des entrées R et S (voir tableau de Karnaugh de R1, S1, R2, S2).

Les équations des sorties sont obtenues comme dans le cas de la solution en logique combinatoire à partir des matrices de sortie .

**Rappel de la table de transition de la bascule RS**

| RS | | | |
|---|---|---|---|
| --0 | 01 | 10 | 0-- |
| t00 | t01 | t10 | t11 |

**Tableaux de Karnaugh des entrées des bascules**

**Matrices d' excitation de Y1**

R1

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | X | | | X |
| 0 1 | X | | | 0 |
| 1 1 | 0 | 0 | | 0 |
| 1 0 | 0 | 0 | | 1 |

$R1=\overline{y2}.a$

S1

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | 0 | | | 0 |
| 0 1 | 0 | | | 1 |
| 1 1 | X | X | | X |
| 1 0 | X | X | | 0 |

S1=y2.a

**Matrices d'excitation de Y2**

R2

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | 0 | | | X |
| 0 1 | 0 | | | 0 |
| 1 1 | 0 | 1 | | 0 |
| 1 0 | X | X | | X |

R2=b

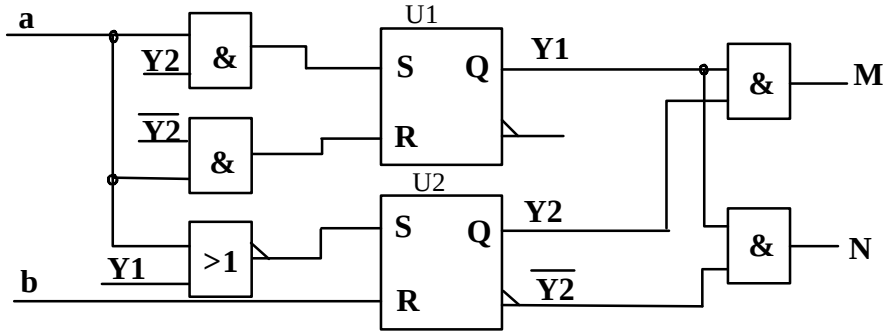
S2

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | 1 | | | 0 |
| 0 1 | X | | | X |
| 1 1 | X | 0 | | X |
| 1 0 | 0 | 0 | | 0 |

$S2=\overline{y1}.\overline{a}$

Les équations sont alors plus simples. Ici il y aura toujours 3 boîtiers (1 pour les bascules, 1 boîtier de 4 portes ET à deux entrées et 1 boîtier de NOR en transformant l'équation de S2). Remarque: pour les bascules RS, il faut disposer des sorties complémentées, on prend alors souvent des bascules D ou JK avec entrées asynchrones Set et Reset (attention les autres entrées doivent être invalidées et non pas laissées en l'air).

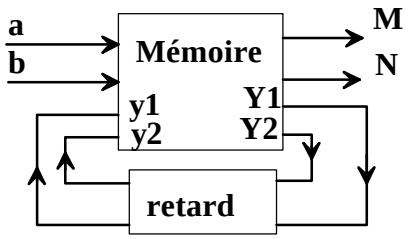
**Schéma structurel à l'aide de bascules RS.**




**B-5-a) Réalisation en logique programmée.**

Cette méthode permet, à partir de la matrice réduite, une réalisation à l'aide de composants programmables (mémoires ROMs, PROMs, EPROMs). Il est alors possible de réaliser des systèmes séquentiels rapides, avec un grand nombre d'étapes. Les variables d'entrées et les variables internes (y1, y2...) donnent l'adresse de la case mémoire, les sorties de la mémoire permettent de réaliser les sorties du système et de fabriquer les variables internes (Y1, Y2....). Les sorties Y1, Y2 ..Yn sont reliées aux entrées y1, y2,...yn (appelé adressage composite) soit par l'intermédiaire de circuit de retard (RC), le système est alors asynchrone (les constantes de temps des circuits RC doivent être supérieures aux temps de propagation et d'accès de la mémoire), soit par des bascules de type D ce qui permet d'avoir un système asynchrone synchronisé (la fréquence d'horloge des bascules doit être inférieure aux temps de propagation et d'accès de la mémoire). La résolution se ramène simplement à l'écriture d'une table ou les colonnes correspondent aux entrées et sorties (variables internes comprises), avec autant de lignes que d'états stables et instables.

**Solution en logique programmée**




**Matrice réduite**

| y1 y2 \ a b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 0 | 2 | | | ① |
| 0 1 | ② | | | 3 |
| 1 1 | ④ | 5 | | ③ |
| 1 0 | ⑥ | ⑤ | | 1 |

| Adr | N°étape | y1 | y2 | a | b | Y1 | Y2 | M | N | Data |
|---|---|---|---|---|---|---|---|---|---|---|
| $2 | ① | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | $0 |
| $4 | ② | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | $4 |
| $E | ③ | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | $E |
| $C | ④ | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | $E |
| $9 | ⑤ | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | $9 |
| $8 | ⑥ | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | $9 |
| $A | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | $0 |
| $0 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | $4 |
| $6 | 3 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | $E |
| $D | 5 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | $9 |

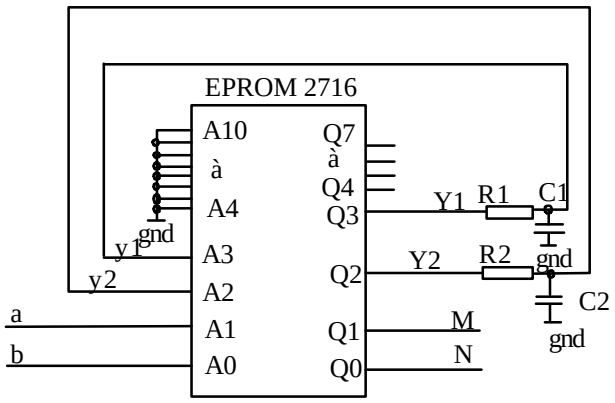
Il suffit alors d'affecter chaque variable d'entrée à un fil d'adresse (ici y1=A3, y2= A2, a= A1, b= A0), et d'affecter chaque grandeur de sortie à une sortie de la mémoire (ici Q3 = Y1, Q2 = Y1, Q1= M et Q0 = N). Après avoir écrit les adresses obtenues en hexadécimal dans l'ordre, ainsi que les données, il n'y a plus qu'à programmer la mémoire.

Ce mode de réalisation permet d'utiliser un grand nombre de variables d'entrée (ex: 11 entrées pour une EPROM 2716, y compris les variables internes), mais les sorties sont limitées au nombre de sortie de la mémoire (souvent 8, y compris les variables internes).

La réalisation de la carte (typon) a l'avantage d'être très simple par rapport à son équivalent en logique combinatoire ou même en logique séquentielle.

**Réalisation en logique programmée**

**(Système asynchrone)**




Remarque: Les circuits RC (R1 C1 et R2 C2) permettent de maîtriser les retards, sinon seuls les temps de propagations limitent la vitesse d'évolution de la séquence. Il faut alors vérifier la bonne évolution de la séquence.

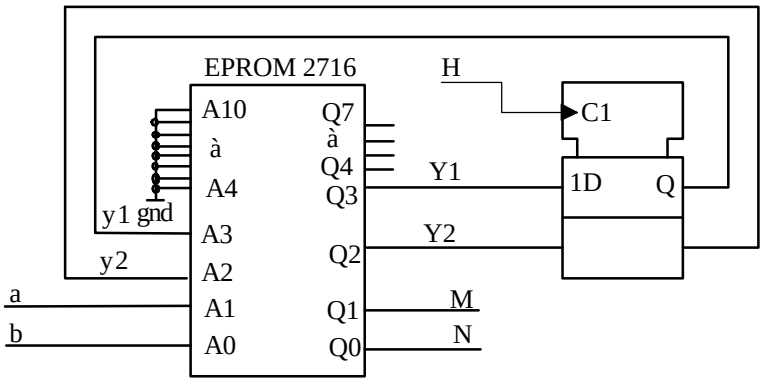
**Systèmes asynchrones synchronisés.**

Pour éviter tout problème de temps de propagation, on préfère imposer la vitesse maximale d'évolution de la séquence en n'autorisant des évolutions que sur les fronts d'une horloge de synchronisation. La vitesse de cette horloge est choisie en fonction de la vitesse maximale d'évolution de la séquence. Ces systèmes sont appelés: systèmes asynchrones synchronisés (voir schéma ci dessous).

Principe: Il suffit de verrouiller à l'aide de registre de type D ( bascule D), les variables internes (et si besoin les variables d'entrées). Il ne peut alors y avoir qu'une étape maximale à chaque période de l'horloge.

**Réalisation en logique programmée**

**(Système asynchrone synchronisé)**



**C) Problème des courses critiques.**

**C-1) Les courses.**

**C-2) Les courses critiques.**

**D) Problème des aléas.**

**E) Conclusion: systèmes asynchrones et systèmes asynchrone synchronisés.**