LOGIQUE SEQUENTIELLE (1)

ES102 / CM6

SÉQUENTIEL versus COMBINATOIRE (1)

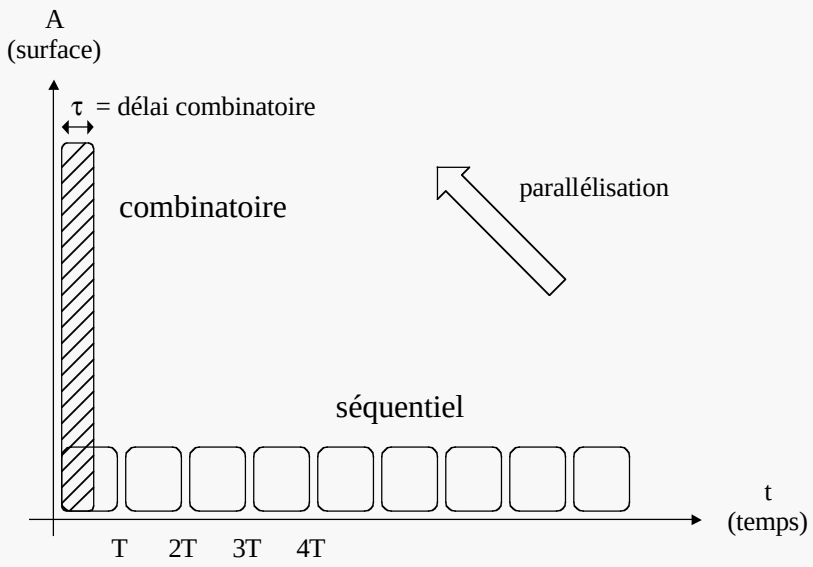
En logique combinatoire :

- Chaque opération booléenne élémentaire requiert des transistors pour elle seule.
- Des fonctions booléennes sont calculées au moyen d'un graphe acyclique de portes (c'est une situation en « boucle ouverte »).
- le délai combinatoire total de calcul est un cumul de constantes RC, dépendant de la topologie du graphe (les graphes les plus rapides sont les arbres équilibrés)
- Plus les fonctions booléennes à calculer sont complexes, plus la structure électronique nécessaire s'étale spatialement.

En logique séquentielle :

- On recourt à des « blocs » combinatoires mais ils sont réutilisés plusieurs fois, pour effectuer successivement la même opération sur des entrées distinctes.
- La dimension temporelle est occupée par la réutilisation itérée des blocs combinatoires. La durée de chaque réutilisation doit excéder les délais combinatoires des blocs utilisés.
- Certaines sorties sont rebouclées sur les entrées (c'est une situation en « boucle fermée »), d'où une aptitude à mémoriser le passé. Il en découle un comportement.
- De la complexité du comportement et de l'ingéniosité de sa réalisation dépend la complexité spatiale du système séquentiel.

SÉQUENTIEL versus COMBINATOIRE (2)

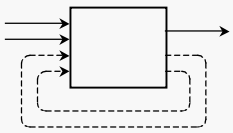


LOGIQUE SEQUENTIELLE : SUBIE OU VOULUE ?

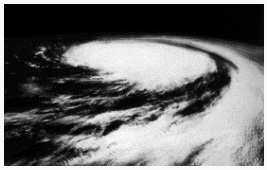
- Subie :données non disponibles simultanément
  - observation/contrôle de systèmes réels dynamiques : signaux, processus industriels, robotique, …
  - restriction d'accès aux données : mémoire, canal de communication
  - données trop nombreuses
- Voulue : valorisation des investissements matériels :
  - taux d'utilisation maximal par gestion optimisée des tâches
  - en se contentant d'un jeu de ressources universelles
    → microprocesseur et approche RISC (cf. CM9/10)
  - en profitant de la dépendance des tâches vis-à-vis des données
    → évaluation paresseuse (a&&b), algorithmie récursive, ⋯

BOUCLE FERMÉE

La logique séquentielle

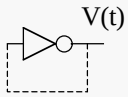


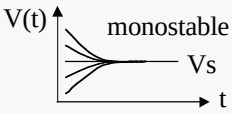
- ne se limite pas à réutiliser des ressources de calcul plusieurs fois avec des entrées différentes.
- C'est aussi réinjecter certains résultats de calcul en entrée !
- On passe donc en « boucle fermée », avec une puissance expressive bien supérieure :
  - de l'itératif au récursif
  - Boucle fermée en automatique, systèmes dynamiques, systèmes chaotiques ⟶



- Mais peut-on reboucler n'importe quoi n'importe comment ?

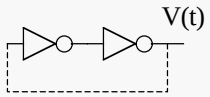
CHAINES D'INVERSEURS EN BOUCLE FERMÉE

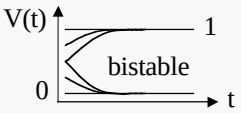




Inverseur rebouclé

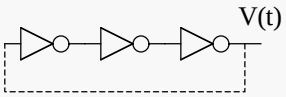
Etat stable analogique à la tension Vs : il s'agit de la frontière entre les 0 et les 1 logiques





Cellule mémoire

Intéressant, mais comment écrire une donnée binaire dedans ? En réouvrant la boucle ?



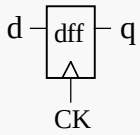
Oscillateur en anneau

Inverseur CMOS = amplificateur

Instabilité ⇒ oscillation

LOGIQUE SÉQUENTIELLE SYNCHRONE (1)

- Comment avoir les avantages de la boucle fermée (récursion) sans ses inconvénients (instabilité) ?
- En rebouclant à travers un **sas** :
  - empêchant un retour direct de sorties vers entrées
  - nommé bascule D et souvent noté « dff » (pour « D-flip-flop »)
  - piloté par un signal spécial appelé « horloge », indépendant des entrées et des sorties, souvent noté CK (pour « clock »)
- Présentation de la bascule D :

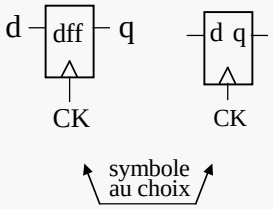


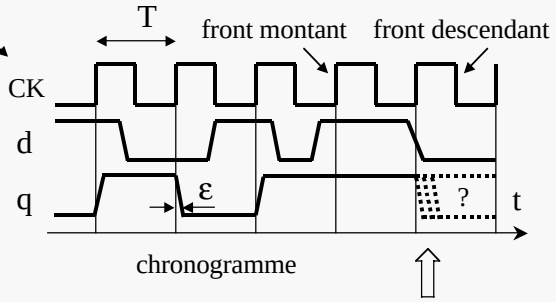
  - L'horloge CK fournit des « tops », à intervalles réguliers de durée T
  - A chaque top, la bascule D « échantillonne » la valeur de son entrée d, puis place (bloque) cette valeur sur sa sortie q jusqu'au prochain top, après un délai combinatoire petit devant T : la bascule D est un « échantillonneur-bloqueur »

FONCTIONNEMENT D'UNE BASCULE D

Horloge = signal binaire périodique à transitions brèves, appelées fronts

Les « tops » d'horloge seront les fronts montants du signal d'horloge

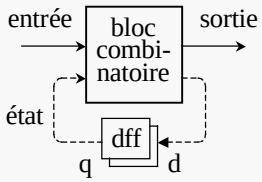




La valeur de d doit être stable de part et d'autre du top d'horloge, avant (temps de pré-positionnement = « setup time ») et après (temps de maintien = « hold time »), sans quoi q n'est binaire qu'au bout d'un délai incertain (métastabilité) et de valeur indéterminée

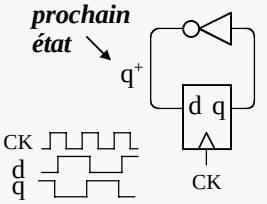
LOGIQUE SÉQUENTIELLE SYNCHRONE (2)

- Le temps, découpé en intervalles réguliers, est discrétisé.
- Des résultats de calcul produits lors d'un intervalle de temps deviennent des données de calcul lors de l'intervalle suivant.
- Grâce à ce découpage du temps, on peut reboucler n'importe quel bloc combinatoire à travers des bascules D !
- Il est possible de faire de la logique séquentielle sans horloge. Cela s'appelle de la logique séquentielle asynchrone (hors ES102).
- Dorénavant, les termes « entrée » et « sortie » ne concerneront plus ce qui est rebouclé par des bascules D, mais uniquement ce qui entre ou qui sort en dehors de cette boucle. Les valeurs q des bascules D seront appelées « état ».



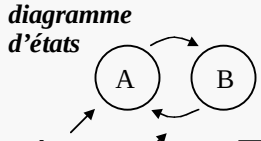
AUTOMATES AVEC UNE SEULE BASCULE D

automate isolé
(sans entrée ni sortie)



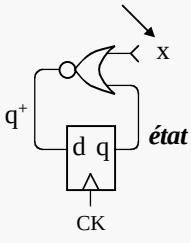
$q^+ = q'$

diagramme d'états

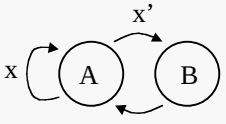


état *transition*

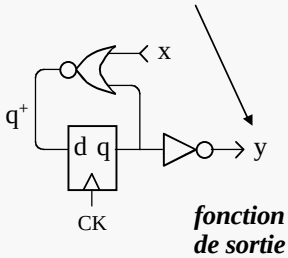
automate
(avec *entrée*)



loi de transition $q^+ = q'x'$

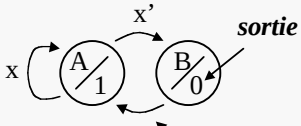


automate
avec *sortie*



fonction de sortie $y = q'$

$q^+ = q'x'$

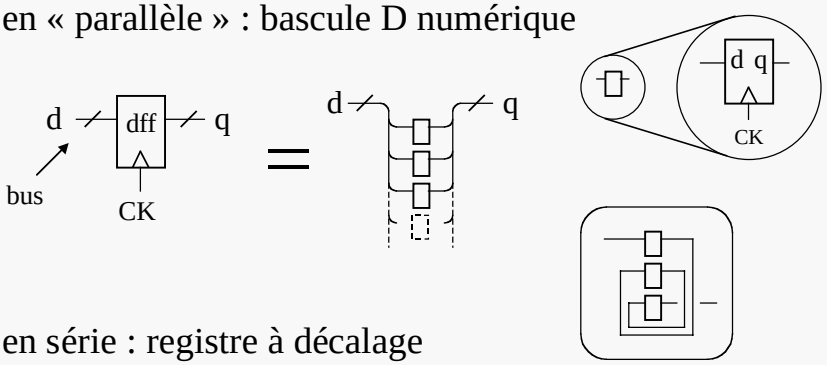


transition inconditionnelle

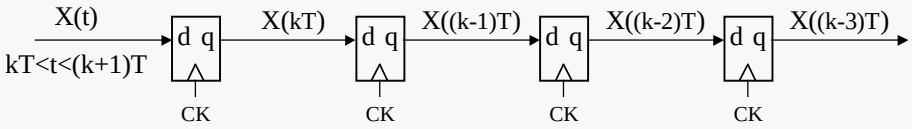
2 états possibles pour la bascule D :
a) q=0 → état A b) q=1 → état B

ASSOCIATIONS DE BASCULES D

- en « parallèle » : bascule D numérique



- en série : registre à décalage



X(t) : signal binaire

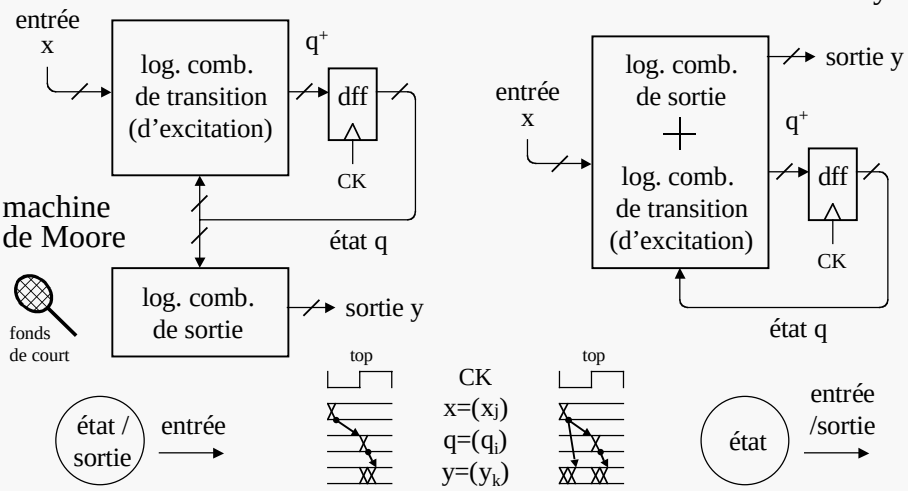
Un registre à décalage remonte le temps …

AUTOMATES FINIS AVEC SORTIE (FINITE STATE MACHINES)

log. comb. = logique combinatoire

q^+ : successeur de q après le prochain top d'horloge

machine de Mealy

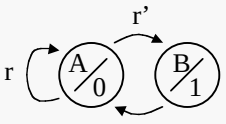


AUTOMATES : FORMALISATION

- Automate avec sortie = quintuplet (X,Q,f,Y,g) tel que :
  - X est l'ensemble (alphabet) des entrées, Y est celui des sorties
  - Q est l'ensemble des états
  - f : $Q \times X \rightarrow Q$, est appelée fonction d'excitation (ou de transition)
    $\rightarrow$ $q^+ = f(q,x)$ est le successeur de q soumis à l'entrée x
  - g : $Q \rightarrow Y$ (Moore) , est appelée fonction de sortie
    $Q \times X \rightarrow Y$ (Mealy)
- Fonctions généralisées de transition f* et de sortie g* :
  - X* = ensemble des séquences finies d'éléments de X (monoïde)
  - $\forall q \in Q, \forall \xi \in X^*, \forall x \in X$, $f^*(q,\xi x) = f(f^*(q,\xi),x)$
    $g^*(q,\xi x) = g(f^*(q,\xi x))$ ou $g(f^*(q,\xi),x)$
    Concaténation de ξ et de x
    Moore Mealy
- Automates finis $\Leftrightarrow$ X et Q sont des ensembles finis

PLUS CONCRÈTEMENT …

- La formalisation des automates (transparent précédent) a donné un sens encore nouveau aux notions d'entrées et sorties :
  une entrée (resp. une sortie) est une configuration possible des signaux d'entrée (resp. de sortie)
- Considérons un exemple antérieur :
- Il y a un seul signal d'entrée, r,
  mais il y a 2 entrées : r=0 et r=1, que l'on abrège en r' et r
- Avec n signaux d'entrée binaires, il y aurait 2^n entrées
- Ici : X = { 0, 1} Y = { 0, 1 } Q = { A, B }
  et f(A,1)=A ; f(A,0)=B ; f(B,-)=A ; g(A)=0 ; g(B)=1 ;
- Mais, pour l'utilisateur, l'important c'est que :
  g*(-,1)=0 ; g*(-,10)=1 ; g*(-,1(00)*)=0 ; g*(-,1(00)*0)=1 ;
  C'est le comportement !



c'est une machine de Moore

\- signifie quelconque

notation unixienne : sous-séquence 00 répétée 0, 1 ou plusieurs fois

SYNTHESE DES AUTOMATES

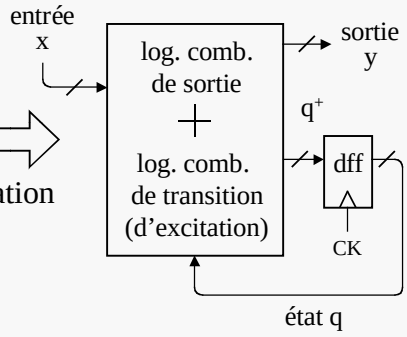
Spécifications comportementales :
entrées X - sorties Y - état initial « naturel » q_0
s : X* $\rightarrow$Y, où s(ξ) est la sortie de l'automate parti de q_0 et ayant subi la série ξ d'entrées

recherche du coût matériel minimal

Etablissement d'un diagramme d'états

Q ? f ? g ? tels que
$\forall \xi \in X^*, g^*(q_0,\xi) = s(\xi)$

Implantation

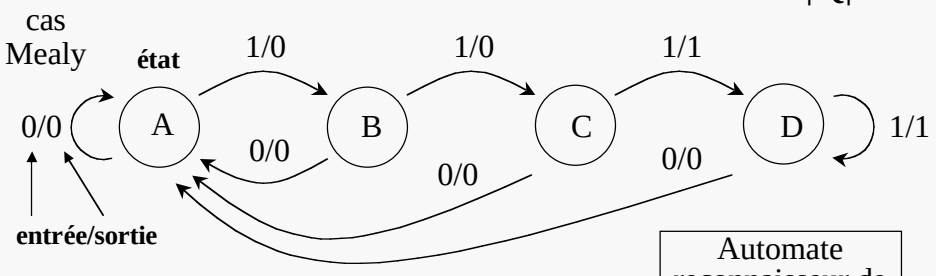


0) Q deviné *(|Q|≥|X|)*
1) Minimisation |Q|
2) Affectation des états → f,g

1) MINIMISATION DU NOMBRE D'ETATS

- Ayant trouvé un jeu Q, f et g fournissant le comportement voulu, il est intéressant de minimiser le cardinal de Q (|Q|)
- Solution : regrouper les états en classes d'équivalence comportementale (notée ≡)
  - $q_1 \equiv q_2$ ssi $\forall \xi \in X^*, g^*(q_1,\xi) = g^*(q_2,\xi)$
  - que l'on met sous une forme récursive plus utile (cas Mealy) : $q_1 \equiv q_2$ ssi $\forall x \in X, g(q_1,x) = g(q_2,x)$ et $f(q_1,x) \equiv f(q_2,x)$
- Algorithme pour établir les classes d'équivalence :
  - 1. Partitionner Q par groupes ayant les mêmes sorties $\forall x \in X$
  - 2. Vérifier si, $\forall x \in X$, chaque groupe a ses successeurs dans un même groupe, sinon séparer les états hétérogènes.
  - Itérer 2 jusqu'à stabilité

EXEMPLE DE MINIMISATION DE |Q|



Automate reconnaisseur de séquences 111

1. 0/0 & 1/0 : {A,B} - 0/0 & 1/1 : {C,D}

2. Mais A et B ne sont pas équivalents
car leurs successeurs f(A,1)=B et f(B,1)=C ne le sont pas,
d'où une partition plus fine : {A }, {B}, {C,D}

2'. Avec la partition précédente, chaque groupe a ses successeurs dans le même groupe. La stabilité est atteinte.

Finalement, C et D peuvent être regroupés en un seul état

2) AFFECTATION DES ETATS

- Pour pouvoir établir les f et g correspondant au nouveau Q, il faut choisir des codes numériques pour chaque état
- Si $|Q| \leq 2^k$, on peut choisir des codes de longueur k bits, ce qui correspond à utiliser k bascules D dans le montage.
- Stratégies d'affectation usuelles :
  - minimisation de la somme cumulée des distances de Hamming entre codes d'états successeurs $000 \rightarrow 100 \rightarrow 101 \rightarrow 111$
  - « one-hot » : une bascule par état 001, 010, 100
  - minimisation exacte du coût impossible sauf par énumération exhaustive, d'où usage d'heuristiques et d'optimisation stochastique

3) ACHEVEMENT DE LA SYNTHESE

- Le choix de Q et des codes correspondant à chaque état déterminent complètement les fonctions f et g.
- Il reste à les implanter sous forme de logique combinatoire en se connectant en entrée et en sortie des bascules D.
- Exemple générique :

  états, entrées et sorties chacun sur 2 bits

  $q = (q_1 q_0)$, $x = (x_1 x_0)$, $y = (y_1 y_0)$

  alors voici les fonctions booléennes à exprimer et implanter

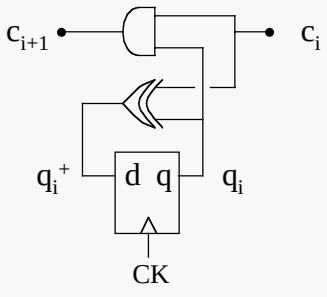
  $q_1^+ = f_1(q_1, q_0, x_1, x_0)$? $q_0^+ = f_0(q_1, q_0, x_1, x_0)$?

  $y_1 = g_1(q_1, q_0)$? $y_0 = g_0(q_1, q_0)$? cas Moore

  $y_1 = g_1(q_1, q_0, x_1, x_0)$? $y_0 = g_0(q_1, q_0, x_1, x_0)$? cas Mealy

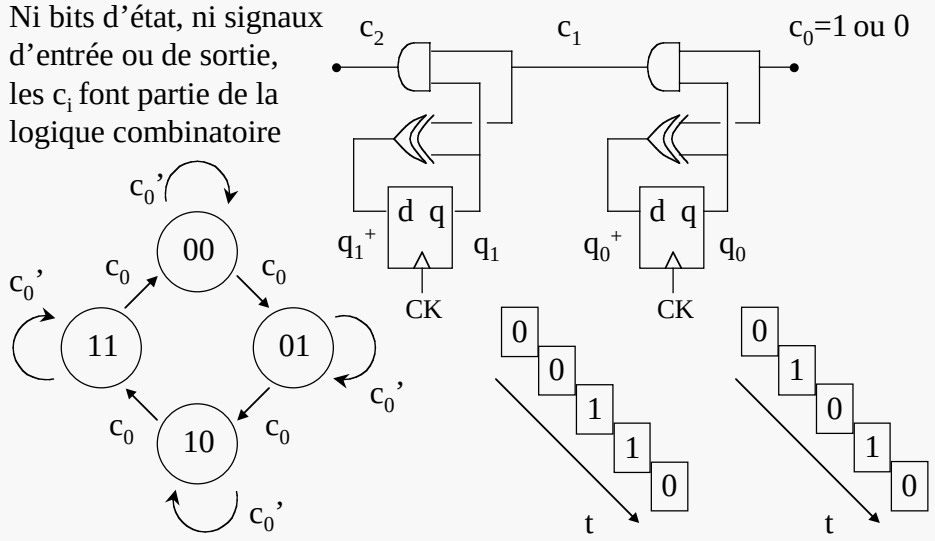
UN AUTOMATE QUI COMPTE …

- Un compteur simple est un automate dont l'état Q indique le temps, qui s'incrémente de 1 à chaque top d'horloge
- Affectation des états = codage binaire naturel de Q
- Le successeur de $Q = (q_i)_{0\le i<n}$ est $Q^+ = (q_i^+)_{0\le i<n} = Q+1$
- Incrémenter un nombre ⇔ lui ajouter 0 avec $c_0=1$
  d'où $c_{i+1} = q_i \cdot c_i$ et $q_i^+ = q_i \oplus c_i$
- Les équations d'évolution ci-dessus se traduisent par la « tranche 1-bit » ci-contre :
- Pour obtenir le compteur complet, il suffit de juxtaposer n tranches, avec $c_0=1$



COMPTEUR CYCLIQUE DE 0 À 3

Ni bits d'état, ni signaux d'entrée ou de sortie, les c_i font partie de la logique combinatoire



Mais c_0 est utilisable en tant qu'entrée : c_0=0 bloque le compteur

LA SEMAINE PROCHAINE : CM7

- Mémoires et bascules
- Registres
- Chemin de données
- Unité de contrôle
- Logique programmée