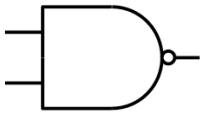
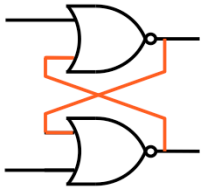
# Chapitre 13:
# Logique synchrone et logique séquentielle

## 13.1 - Introduction

## Introduction: ce qu'on a déjà vu...

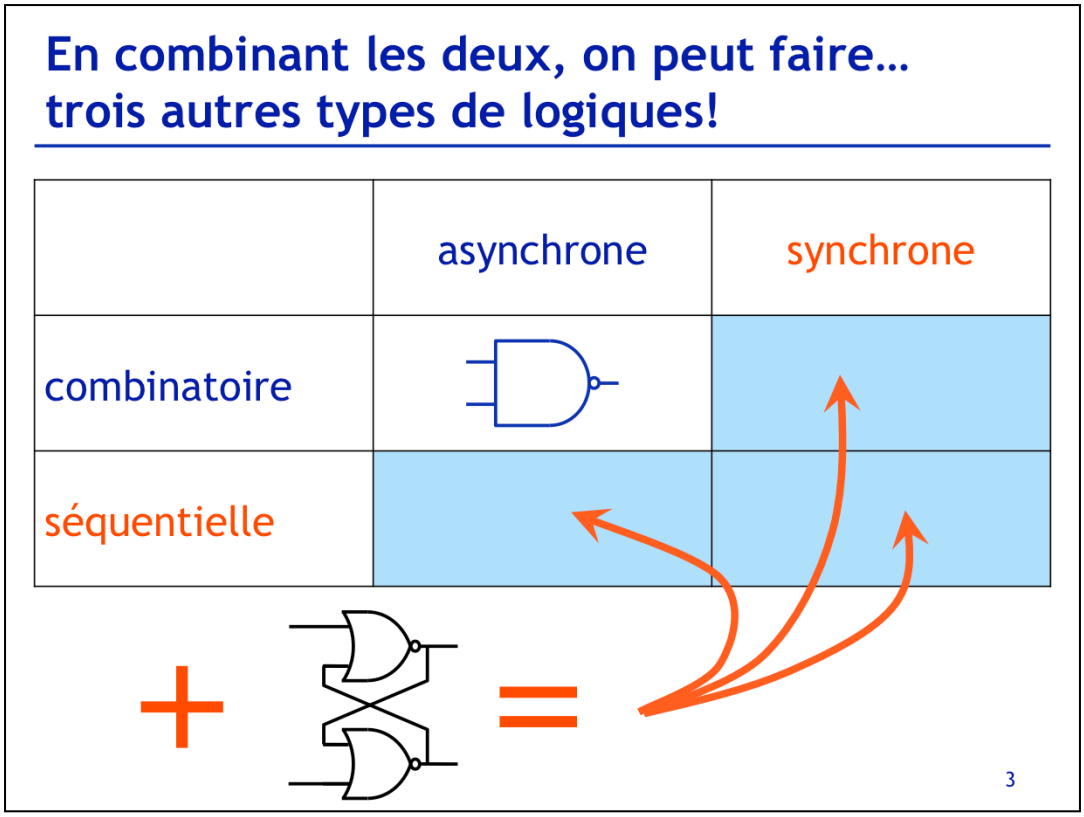
| chap. 11 | chap. 12 |
|---|---|
|  |  |
| **fonctions combinatoires** | **bistables** |
| **traitement** | **mémorisation "pure"** |
| pas de rétroaction | rétroaction |
| **combinatoire** | ... |
| **asynchrone** | ... |
| conditions de course (!) | entrée d'activation (latch ou flip-flop) |

2

Jusqu'ici, nous avons vu fondamentalement deux types de circuits:

-dans le chapitre 11, des portes et des fonctions logiques "combinatoires asynchrones"
-dans le chapitre 12, des mémoires et en particulier des mémoires de 1 bit appelées "bistables"

Le tableau ci-dessus récapitule et compare quelques points-clés de ces deux types de dispositifs:
-la logique combinatoire asynchrone a pour fonction de réaliser un *traitement* sur des données. Au contraire un bistable permet de *mémoriser* une donnée (d'un bit).
-les deux types de fonctions sont fondamentalement réalisées à l'aide de portes logiques, néanmoins la logique combinatoire asynchrone est constituée sur base d'un ensemble de portes logiques connectées sans rétroaction des sorties sur les entrées, tandis qu'un bistable comporte fondamentalement deux portes et une rétroaction positive. Nous verrons dans le présent chapitre comment qualifier la mémoire (car elle ne fait pas partie de la logique combinatoire asynchrone).
-Enfin nous avons vu qu'à la sortie des circuits logiques combinatoires asynchrones sont susceptibles de se produire des impulsions parasites qui ont pour origine les délai incontrôlés des différentes portes logiques (problème des "conditions de course"). Lors de l'étude des bistables, nous avons vu qu'une solution consiste à utiliser une "entrée d'activation" qui, pilotée par un signal adéquat (déclenchement sur flanc ou sur niveau), permet de masquer les signaux d'entrée pendant qu'ils subissent ces impulsions parasites. Ce principe va être approfondi dans le présent chapitre.



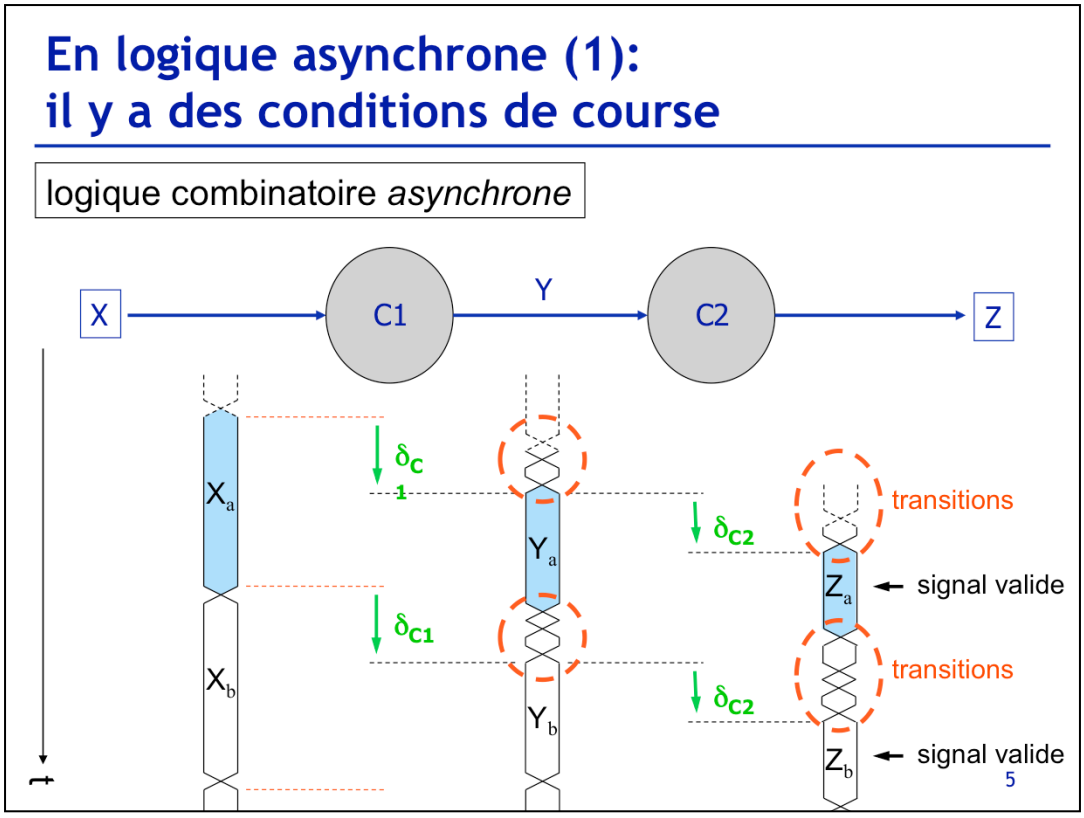

En combinant la logique combinatoire asynchrone (pour le traitement) et des bistables (mémorisation), on peut créer des logiques qui répondent à d'autres principes de fonctionnement que la logique combinatoire asynchrone.

On peut ainsi introduire:
-d'une part la logique "synchrone" (déjà abordée partiellement lors de l'explication des entrées d'activation), qui s'oppose à la logique asynchrone
-d'autre part la logique "séquentielle", qui s'oppose à la logique combinatoire

A l'analyse, il apparaît qu'on peut ainsi, en plus de la fonction de mémorisation "pure" définir quatre types de logiques numériques différentes. C'est ce que nous allons analyser dans ce chapitre.

# Chapitre 13:
# Logique synchrone et logique séquentielle

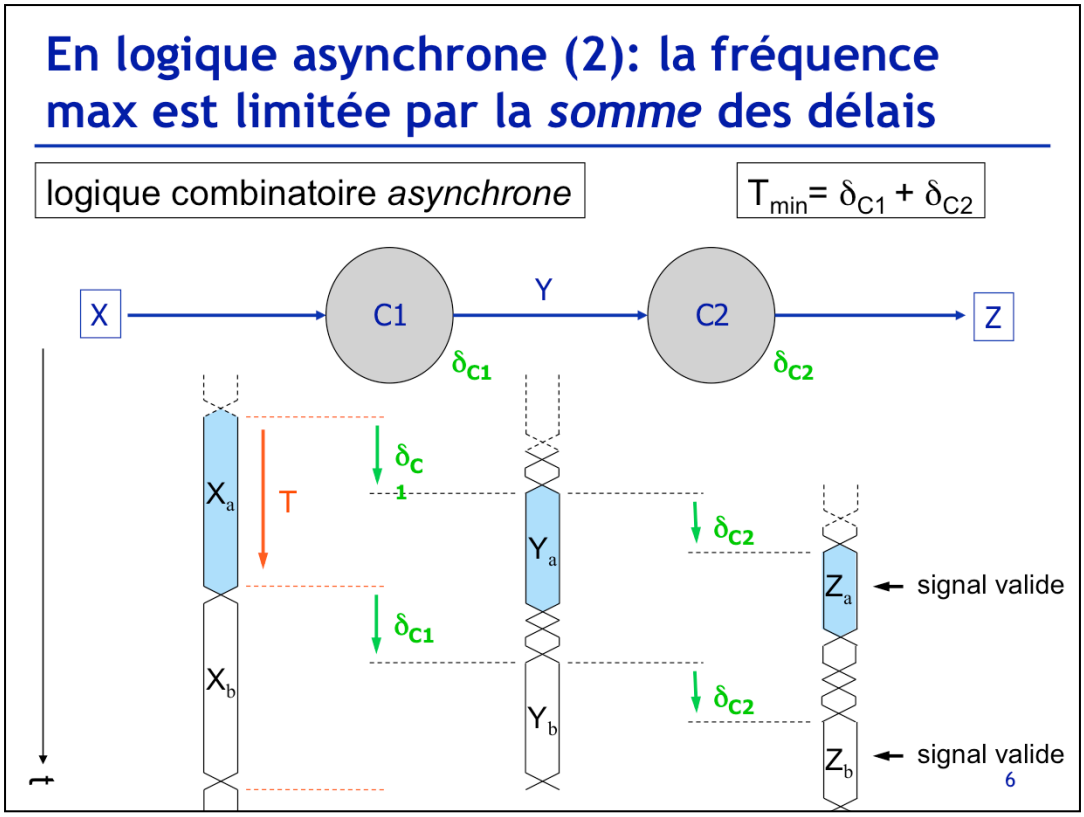
## 13.2 – Logique synchrone




On a representé ci-dessus un circuit combinatoire asynchrone constitué de deux blocs C1 (traitement transformant une valeur X en valeur Y) et C2 (traitement transformant une valeur Y en valeur Z). Ce circuit est sensible aux conditions de course.

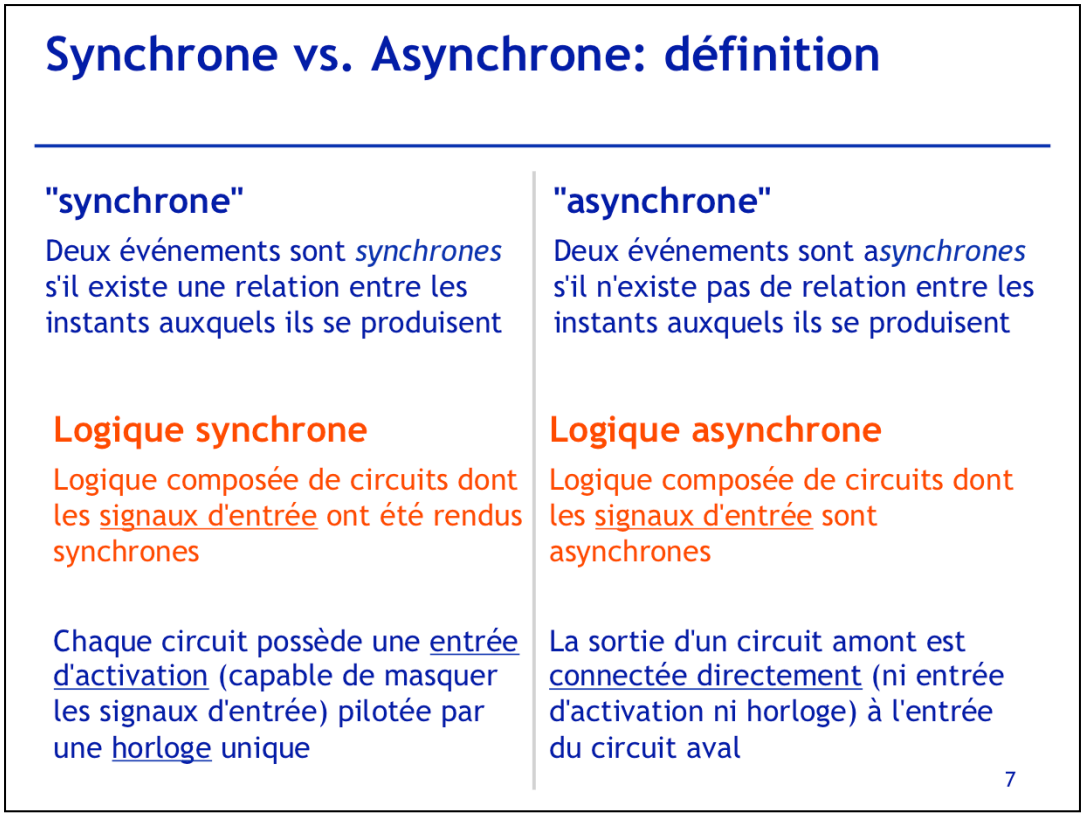
En particulier, si on impose à l'entrée du circuit (valeur X) des données qui changent à un certain rythme (voir chronogrammes selon l'axe vertical):
- pour un signal Xa imposé à un certain instant, la sortie du circuit C1 subit d'abord des transitions parasites dues aux conditions de course et ne se stabilise à une valeur Ya qu'après un délai $\delta_{C1}$
- de la même manière, lorsque Ya change, la sortie du circuit C2 (dont l'entrée reçoit la valeur Ya) subit d'abord des transitions parasites dues aux conditions de course et ne se stabilise à une valeur Za qu'après un délai supplémentaire $\delta_{C2}$

On peut se rendre compte que le laps de temps pendant lequel le signal Za est valide est beaucoup plus court que la durée pendant laquelle on a imposé la valeur Xa…




En particulier, si on envisage de faire changer régulièrement Xa (pour traiter des données successives, ce qui est le cas de la plupart des circuits), on voit que la période T du signal X doit être supérieure à la *somme* des délais des différents circuits combinatoires.

=> en logique combinatoire asynchrone, plus le circuit est complexe, plus le traitement est lent, chaque circuit (et même chaque porte) additionnant son délai aux délais précédents.

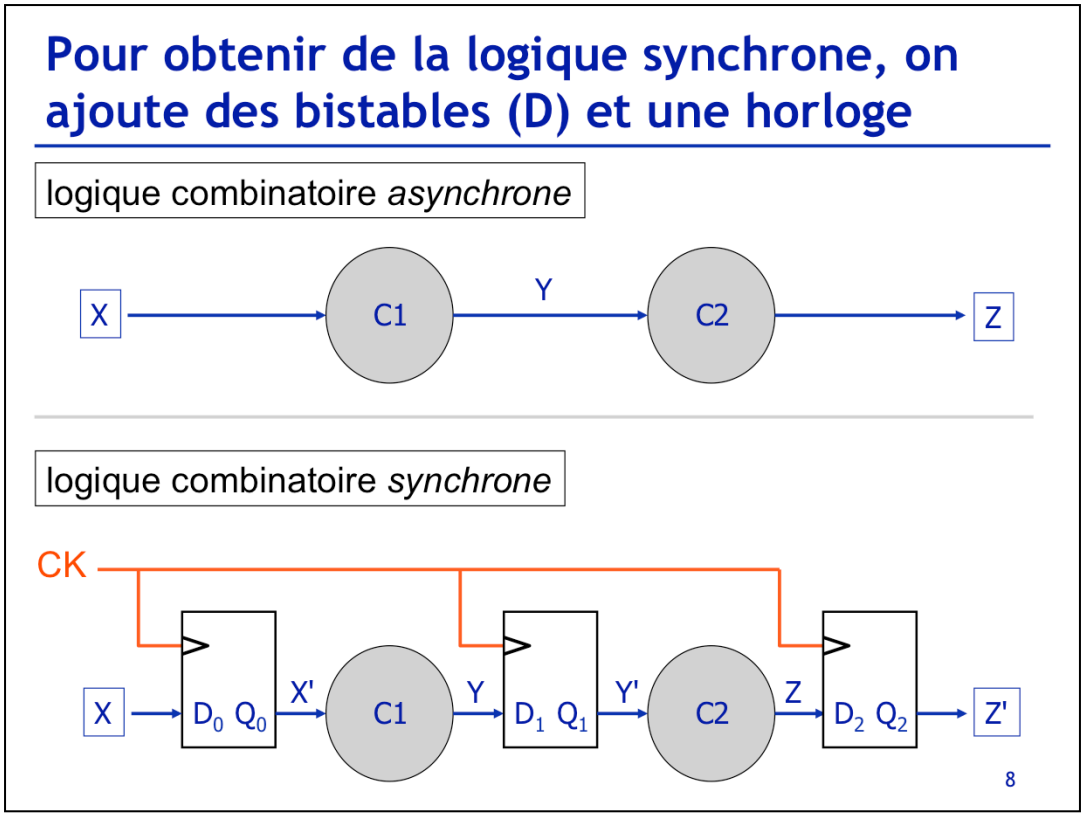



Avant de discuter en détail les propriétés de la logique synchrone et de la logique asynchrone, nous allons en donner une définition de référence.

Examinons d'abord pour cela la signification des termes synchrone et asynchrone en langage courant:
-en langage courant, deux événement sont "synchrones" (ou "synchronisés") s'ils ont lieu au même instant (ou de manière plus générale s'il existe une relation bien définie entre les instants où ces événements se produisent)
-au contraire, deux événements sont "asynchrones" si les instants auxquels ils se produisent ne sont pas liés

Sur base de ces définitions, on peut dire que:
-la logique synchrone est une logique composée de circuits dont les *signaux d'entrée* ont été rendus synchrones (c'est-à-dire synchronisés sur un même signal temporel appelé "horloge")
-la logique asynchrone est une logique composée de circuits dont les signaux d'entrée ne sont pas synchronisés
Comme on le voit, la différence entre logique synchrone et asynchrone se situe fondamentalement au niveau des *signaux d'entrée* des circuits.

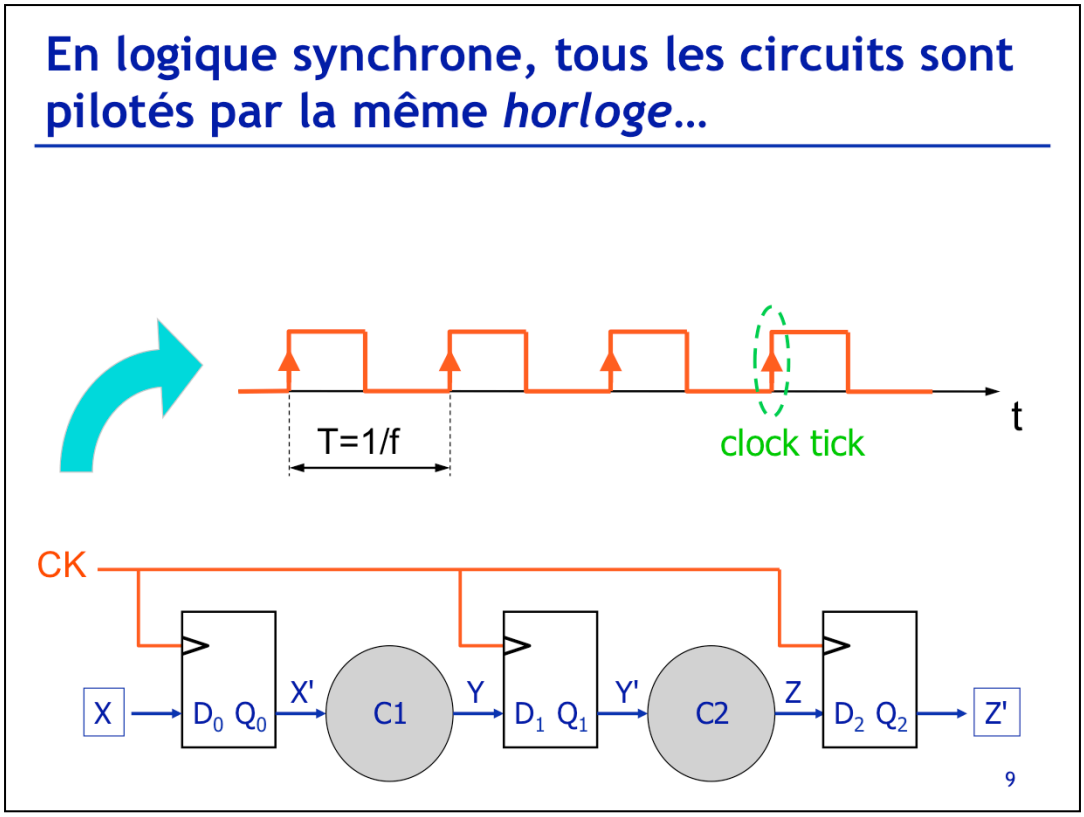
Les définitions précédentes étant assez abstraites, il est intéressant de les compléter de la manière suivante (précisant ainsi les *moyens* utilisés):
-en logique synchrone, chaque circuit possède une entrée d'activation (c'est-à-dire un dispositif capable de masquer sur commande les entrées de données du circuit). Toutes les entrées d'activation sont commandées par un signal unique appelé horloge.
-en logique asynchrone, les circuits ne possèdent ni entrée d'activation ni horloge: la sortie d'un circuit amont est connectée directement aux entrées de données du circuit aval.
La présence ou l'absence de ces éléments (entrée d'activation et horloge) permet facilement de déterminer si un schéma est synchrone ou asynchrone.




La version synchrone du circuit combinatoire présenté précédemment (C1/C2) est montrée dans la figure inférieure: on a inséré derrière chaque bloc combinatoire (ainsi qu'en entrée du montage) un bistable D.

Chaque bloc combinatoire, pris isolément, est encore sensible aux conditions de course. Les signaux Y et Z sont donc susceptibles de contenir des impulsions parasites.
Mais chacun de ces blocs est suivi (*) d'un bistable D qui "masque" ces impulsions parasites (et maintient constante la valeur "utile" du signal grâce à sa fonction de mémorisation) vis-à-vis du circuit suivant, de sorte que les signaux Y' et Z' sont débarrassés de ces impulsions parasites.

(*) Le premier bistable (X) n'est pas nécessaire si le signal X est lui-même synchronisé sur la même horloge. Un bistable d'entrée a été inclus en supposant que X était un signal asynchrone vis-à-vis de notre circuit.



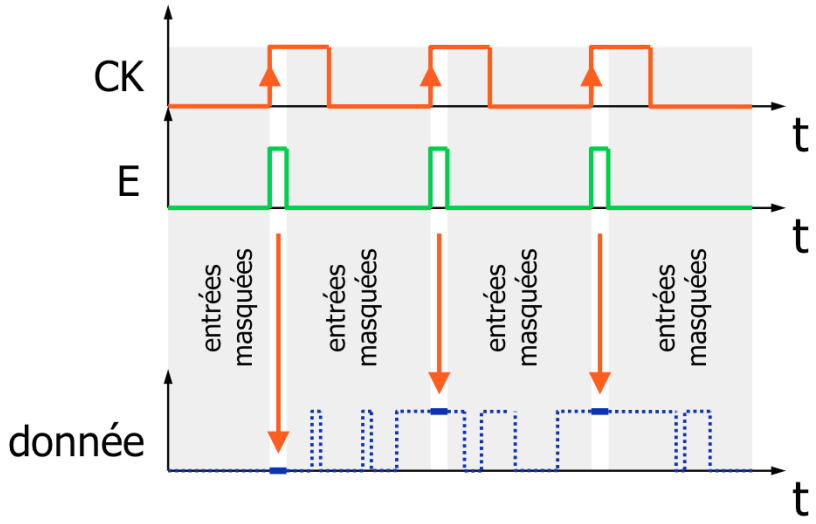

Outre les bistables, un élément caractéristique de la logique synchrone est donc le fait que tous les circuits sont pilotés par une "**horloge**" (ou "signal d'horloge").

Ce signal d'horloge est un signal binaire périodique (en d'autres termes: une onde carrée). Une de ses propriétés essentielles est bien entendu sa fréquence f (ou sa période T, qui en est l'inverse: T=1/f).

Ce signal d'horloge peut être vu comme une référence temporelle pour tous les circuits: il active périodiquement et en même temps toutes les entrées d'activation des autres circuits (et en particulier des bistables). C'est donc lui qui provoque la synchronisation de tous les signaux d'entrée comportant des données.

L'activation des différents circuit peut avoir lieu sur niveau ou, plus souvent, sur flanc. Chaque passage du signal d'horloge par le niveau ou le flanc qui active les autres circuits (et, par extension, chaque activation d'un autre circuit) est appelée un "**coup d'horloge**" ("clock tick" en anglais).

## ...qui active périodiquement, brièvement et simultanément les entrées d'activation




10

L'idée de la logique synchrone est d'activer simultanément, via ce signal d'horloge, toutes les entrées d'activation des différents circuits.

A ce moment (c'est-à-dire à chaque coup d'horloge), chaque circuit "voit" brièvement ses entrées, c'est-à-dire reçoit une nouvelle donnée qu'il va pouvoir traiter.

Comme illustré ci-dessus, les impulsions parasites existent toujours mais sont masquées car elles surviennent pendant que les entrées d'activation sont inactives (entre les coups d'horloge).

## Synchrone *vs*. asynchrone: analogie


11

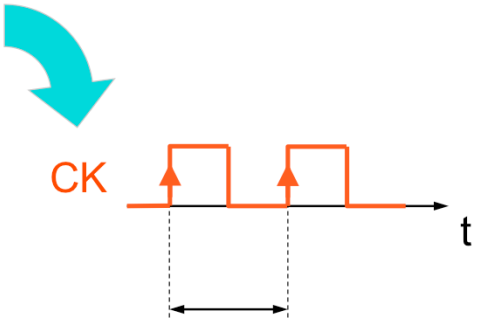
Analogie: une chaîne de porteurs d'eau se passant des seaux.

En bas (troisième dessin): fonctionnement asynchrone
Aucune mesure n'est prise pour synchroniser globalement le passage des seaux (pas de "métronome"): chaque individu passe son seau dès qu'il l'a reçu.

En haut: fonctionnement synchrone
Il existe un dispositif qui synchronise tous les individus (symbolisé par le métronome à gauche) => le fonctionnement est périodique
-pendant la première phase (dessin du haut): chaque individu dépose son seau à sa gauche
-pendant la seconde phase (dessin du milieu): chaque individu prend le seau qui se trouve à sa droite

Remarques:
1) dans le cas de la chaîne de seaux, il n'y a aucun traitement de "l'information" (le seau): elle passe simplement d'un individu à l'autre sans être modifiée
2) les avantages du mode de fonctionnement synchrones ne sont pas évidents dans le cas illustré (sinon que personne ne doit *attendre* de prendre ou de donner le seau à son voisin. Tout le monde doit par contre attendre le signal du métronome). En électronique, les principaux avantages sont d'éviter les conditions de course et d'augmenter le débit d'information traitée

## Exemple: un PC, c'est de la logique synchrone…

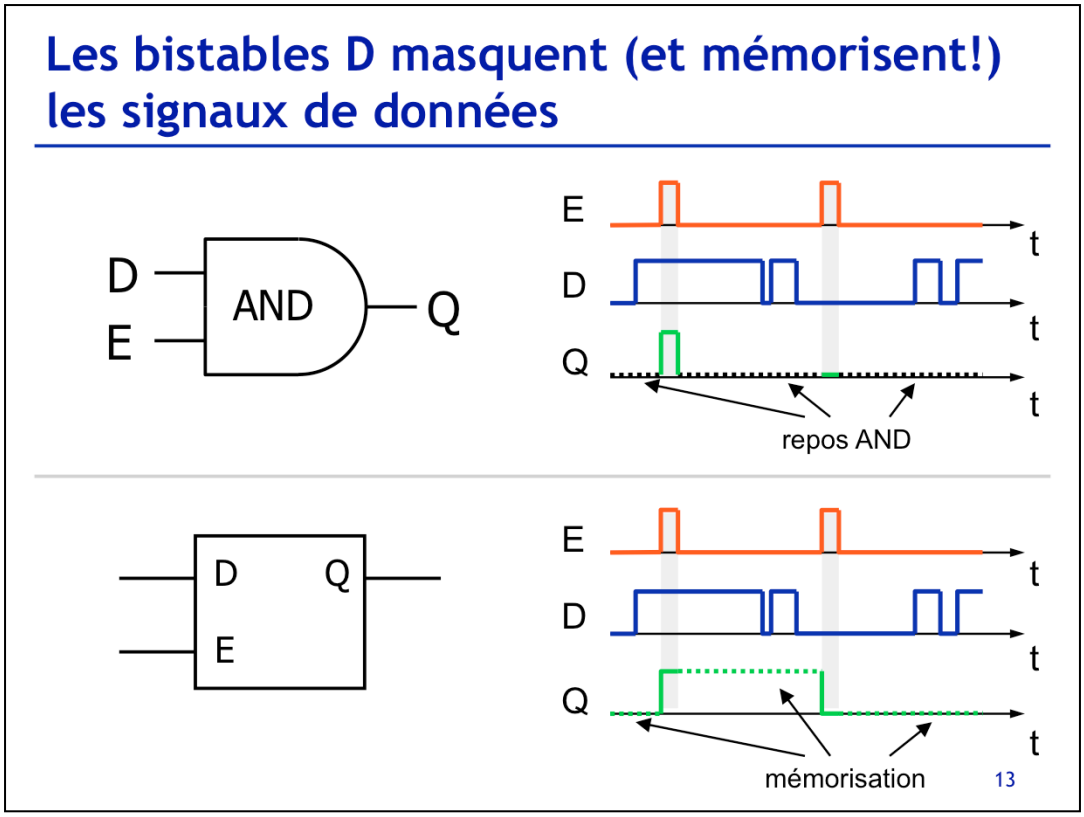




T = 0,33 ns !

12

L'horloge d'un PC (la fréquence qui figure dans toutes les publicités comme "mesure" -discutable- des performances du PC) est bien une horloge au sens de la logique synchrone. Pour une horloge de fréquence 3GHz, la période du signal d'horloge est donc de $T=1/3.10^9=0,33ns$.

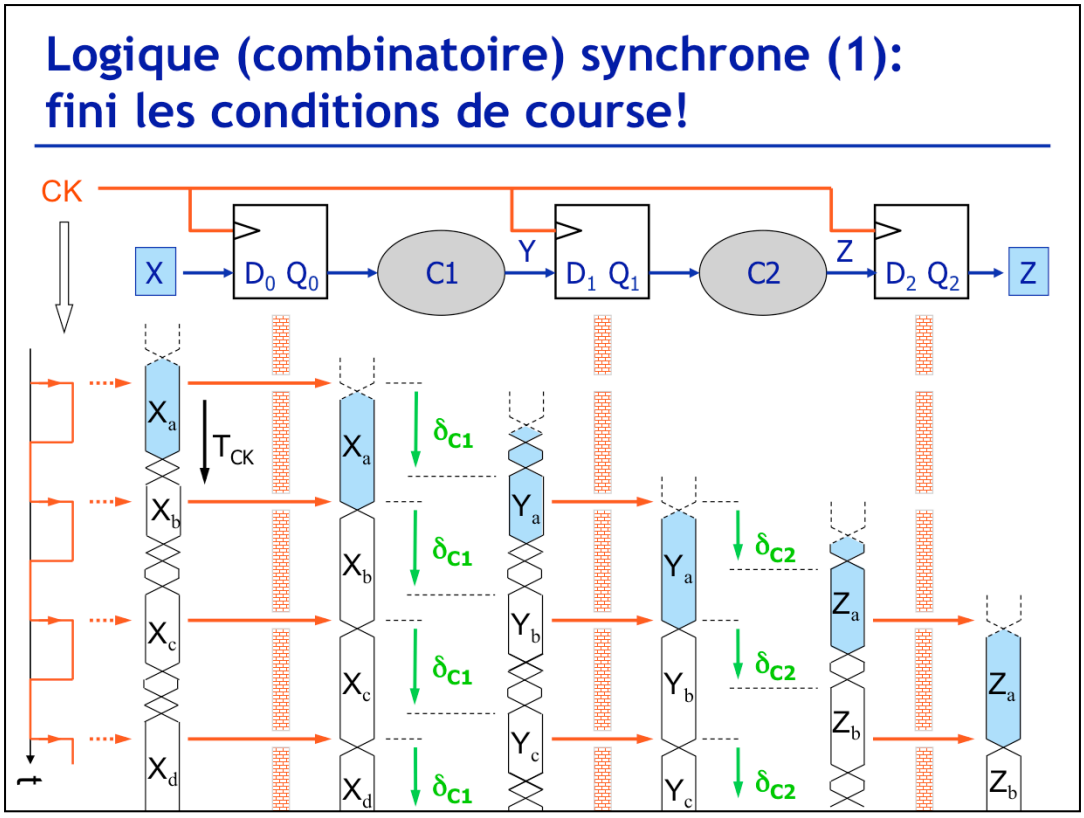
N.B.: attention, un PC, c'est de la logique synchrone, mais pas de la logique combinatoire! (voir plus loin)




Pourquoi utiliser un bistable D plutôt qu'une simple porte AND pour masquer (cfr chap. 11) périodiquement les signaux de données?

La porte AND, si elle permet effectivement de masquer le signal D, a l'inconvénient de délivrer un niveau logique "0" (puisque sa sortie est active à l'état haut) chaque fois que l'horloge E est inactive (vaut 0).

Une solution plus avantageuse consiste à remplacer l'étage de portes AND par… un bistable D complet (qui comprend donc un étage de AND ou NAND pour masquer D mais est en plus capable de mémoriser). On peut vérifier dans ce cas que la sortie du bistable reste constante sur une période (contrairement au cas des portes AND), ce qui est un avantage pour les circuits situés en aval.




Pour la bonne compréhension, voici la description de ce qui se passe dans un circuit synchrone sous forme de chronogrammes (pour rappel: chaque chronogramme représente une valeur logique ou numérique, un "X" dans le chronogramme représentant un changement de valeur du signal).

L'axe du temps est vertical et l'horloge CK est représentée à gauche du slide. Chaque flanc montant de CK représente un "coup d'horloge".
On présente à l'entrée du montage une variable X qui prend successivement les valeurs Xa, Xb, Xc, Xd, etc (cette variable X comporte des impulsions parasites pour bien montrer l'utilité du premier bistable).

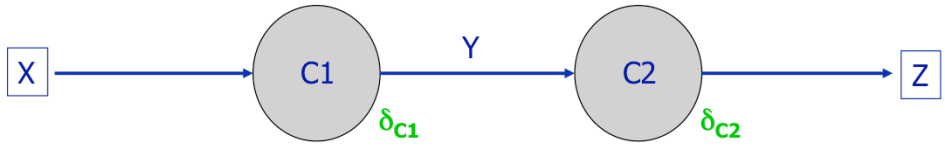
Que font les bistables?
A intervalle régulier (égal à la période d'horloge T), le premier bistable "regarde" et mémorise son entrée (le reste du temps, son entrée et sa sortie sont découplées, ce qui est symbolisé ci-dessus par petit "mur de briques"). On voit à ce stade que ce bistable produit un signal équivalent au signal X mais débarrassé des impulsions parasites. C'est ce signal "propre" qui est présenté à l'entrée du bloc C1.
Le processus est le même (élimination des impulsions parasites par mémorisation) pour les deux autres bistables connectés en sortie des blocs C1 et C2.

Que font les blocs combinatoires?
A chaque coup d'horloge, le bloc C1 reçoit une nouvelle donnée. Il la traite, c'est-à-dire qu'il produit une valeur Y correspondante (par exemple Ya sur base de Xa) après un certain délai de calcul $\delta_{C1}$. Pendant ce délai ont lieu des impulsions parasites.
Le bloc C2 fonctionne de la même manière, éventuellement avec un délai $\delta_{C2}$ différent, transformant la valeurs Y en valeurs Z.

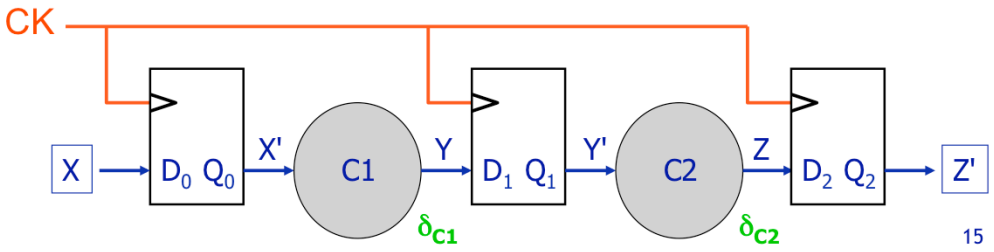
Pour que le système fonctionne, il faut que la période d'horloge soit plus grande que le maximum des délais des blocs combinatoires ($\delta_{C1}$ ,$\delta_{C2}$) .

## Logique (combinatoire) synchrone (2): et c'est plus "rapide" que l'asynchrone!

logique combinatoire *asynchrone*: $T_{min}=\delta_{C1}+\delta_{C2}$



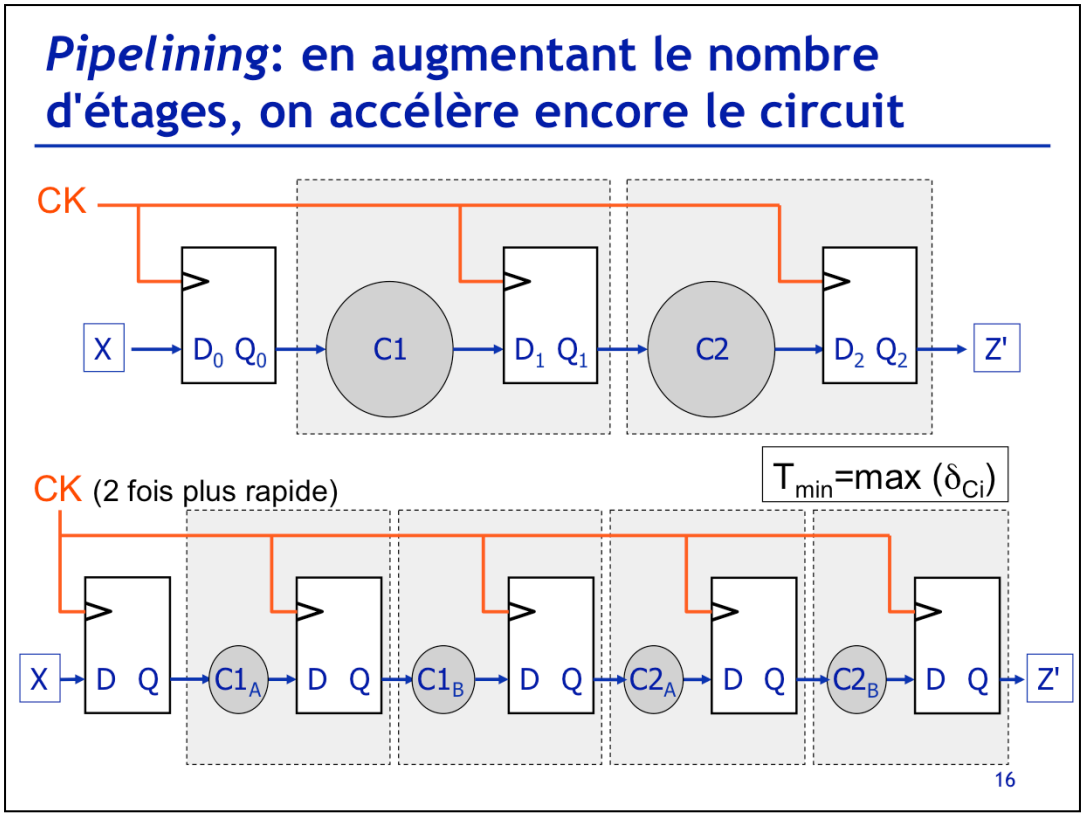

logique combinatoire *synchrone*: $T_{min}=\max(\delta_{C1};\delta_{C2})$




Outre l'élimination des conditions de course, la logique synchrone a un second avantage: elle augmente le débit d'information traitée par le système.

En effet:
-en logique asynchrone, la période doit être supérieure à la somme des délais des différents blocs combinatoires
-en logique synchrone, la période doit simplement être supérieure au délai du bloc combinatoire le plus lent (puisque les données sont "nettoyées" des impulsions parasites et mémorisées entre chaque bloc combinatoire).

Attention: pour produire la première information, un circuit synchrone est forcément plus lent qu'un circuit asynchrone (car il faut autant de périodes qu'il y a de circuits combinatoires et que cette période correspond au délai le plus long). Par contre, une fois que le circuit a produit sa première donnée, une nouvelle donnée apparaît à sa sortie à chaque coup d'horloge. Le débit d'information est donc bien plus élevé que pour le circuit asynchrone correspondant.
Un circuit synchrone est donc avantageux pour traiter un débit régulier d'information tandis qu'un circuit asynchrone peut être plus avantageux pour traiter des événements apériodiques.



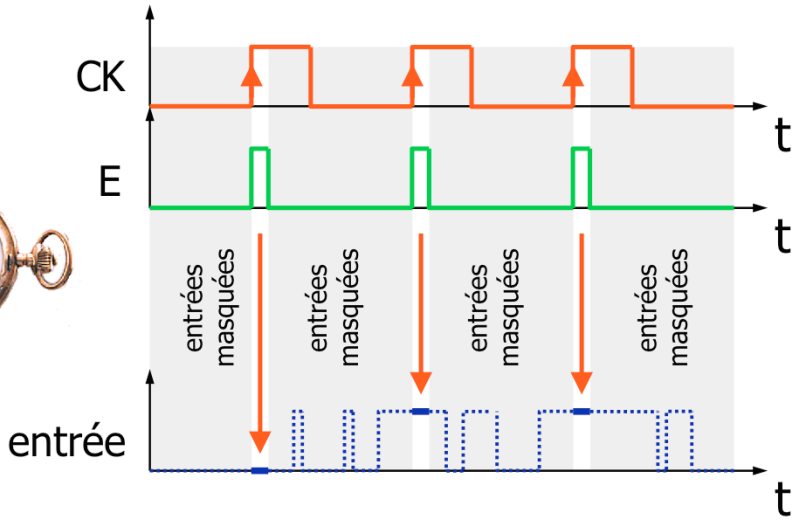

En logique synchrone (combinatoire ou séquentielle), le "pipelining" est une technique qui permet d'accélérer le fonctionnement du circuit. Elle découle des constatations déjà faites sur la fréquence maximum d'horloge dans un circuit synchrone.

En augmentant, pour un même traitement, le nombre d'étages (bloc combinatoire + bistable), on peut raisonnablement supposer qu'on réduit le temps de calcul des blocs combinatoires présents dans la chaîne. Comme la période d'horloge minimale est fixée par le bloc combinatoire le plus lent, des blocs combinatoires plus petits, donc plus rapides, permettent d'augmenter la cadence d'horloge globale du système.

En conséquence, et puisqu'une nouvelle valeur en sortie est délivrée à chaque coup d'horloge, le débit d'information traitée augmente d'autant (par exemple deux fois dans l'exemple représenté ci-dessus, en supposant que le délai de chaque bloc a été divisé par deux en divisant ce bloc, ce qui n'est pas forcément trivial!).
Le temps de "mise en route" du système (le délai nécessaire pour obtenir la première valeur en sortie), lui, n'a pas changé.
Le pipelining a tout son intérêt lorsqu'il fonctionne en régime. Toute perturbation d'un pipeline qui amènerait à redémarrer celui-ci dégrade significativement le gain en performance.

La technique du pipe-line est couramment utilisée (également en logique séquentielle), notamment dans les processeurs.

## N.B.: Le déclenchement sur flanc introduit une *discrétisation* du temps

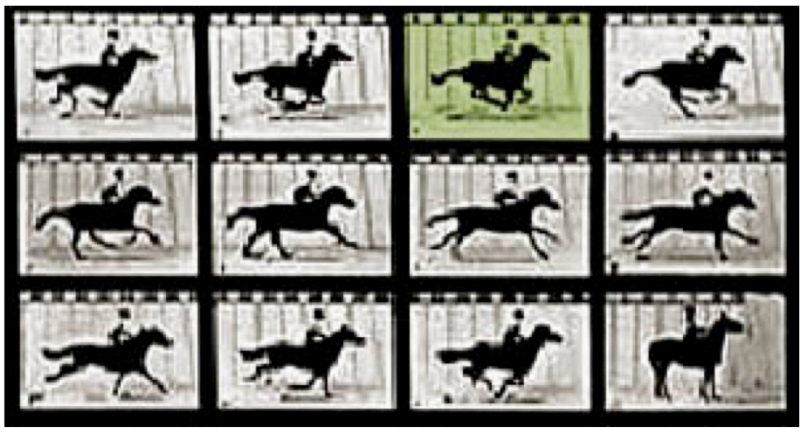



17

Une petite remarque en passant: le principe de la logique synchrone introduit la notion de temps discret (= quantifié). On peut en effet considérer que les signaux n'ont une signification (= ne portent un signal utile) que pendant le niveau/flanc actif de l'horloge. Chaque niveau/flanc actif représente un "instant" discret.

On voit donc apparaître deux notions de quantification (qu'il ne faut pas confondre!) liées à la logique numérique:
- en logique numérique, la *valeur* du signal lui-même est quantifiée puisque, par définition, l'information est codée sous forme de signal numérique (N états) ou logique (2 états)
- en logique synchrone le *temps* est également quantifié, faisant apparaître une suite d'instants successifs

## Analogie: notion d'*échantillonnage*

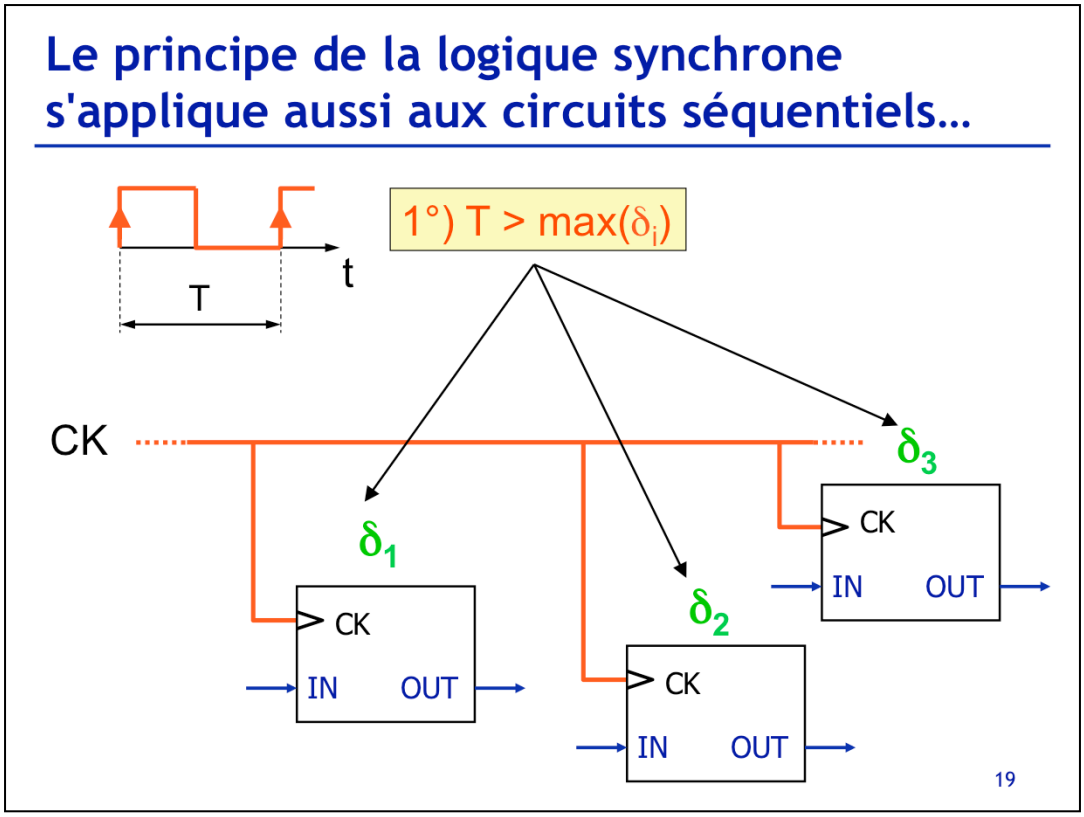



18

Le principe consistant à "regarder" les entrées d'un système périodiquement mais sur une durée très courte (et au contraire le reste du temps à masquer ces entrées qui peuvent alors évoluer sans perturber le système) revient à faire un *échantillonnage* de ce signal d'entrée.

Ce n'est en fait rien d'autre que ce que fait une caméra: un film est une succession d'images, chaque image étant une photo prise suffisamment rapidement pour supposer que la scène ne "bouge" pas (ou de manière négligeable).

Nous reviendrons sur la notion d'échantillonnage dans un chapitre ultérieur.




Le principe de la logique synchrone s'applique aussi aux circuits séquentiels: on peut généraliser les principes précédents à un ensemble de circuits séquentiels (pas forcément des bistables) possédant chacun une entrée d'activation et recevant une horloge commune.

## En bref

- Logique synchrone
  - entrées d'activation + horloge
  - moyen: bistables
  - temps discret (échantillonnage)
- avantages
  - conditions de course maîtrisées
  - augmente le débit d'information

21

# Chapitre 13:
# Logique synchrone et logique séquentielle

## 13.3 – Logique séquentielle

# Chapitre 13:
# Logique synchrone et logique séquentielle

## 13.3.1 – Séquentiel *vs.* Combinatoire

## Séquentiel vs. Combinatoire: définition

**"séquentiel"**

séquence = *suite* d'actions ou d'états déterminée

**"combinatoire"**

dénombrement d'un certain nombre de possibilités *concurrentes*

**Logique séquentielle**

Les sorties à l'instant t dépendent des entrées à cet instant ET des entrées aux instants précédents

- rétroaction dans la fonctionnalité du circuit
- requiert de la mémoire

**Logique combinatoire**

Les sorties à l'instant t dépendent des entrées à cet instant t uniquement

- pas de rétroaction dans la fonctionnalité du circuit
- ne requiert pas de mémoire

24

Nous allons maintenant donner une définition de référence de l'opposition "combinatoire/séquentiel".

Examinons d'abord pour cela la signification des termes combinatoire et séquentiel en langage courant:
-en langage courant, l'adjectif "combinatoire" désigne un ensemble de possibilités *simultanées* ou concurrentes (dont l'une peut se réaliser – exemples: les 6 faces d'un dé; le nombre de combinaisons au lotto, etc)
-à l'inverse, l'adjectif "séquentiel" évoque une *suite* d'actions ou d'états

Sur base de ces définitions, on peut dire que:
-en logique combinatoire, la notion de temps n'intervient pas dans le calcul: les sorties du circuit combinatoire à un instant t dépendent uniquement des entrées de ce circuit *au même instant*
-à l'inverse, en logique séquentielle, les sorties du circuit à un instant t dépendent des entrées de ce circuit *au même instant mais également aux instants précédents*. Il y a donc, dans la fonction même du circuit, une notion de temps et plus précisément de *passé*.

Quelques remarques:
-Les délais des portes logiques combinatoires semblent contradictoires avec la définition ci-dessus ("le temps ne joue pas pour les circuits combinatoires"). On peut néanmoins considérer que ces délais n'interviennent pas en tant que tel dans le *traitement* réalisé dans le circuit (comme c'est au contraire le cas en logique séquentielle)
-La logique séquentielle introduit forcément la notion de temps et en particulier d'états successifs (et d'instants successifs) du système. Un circuit séquentiel nécessite en particulier une forme de mémoire (sans quoi la sortie ne pourrait pas dépendre des valeurs prises par le circuit dans son passé). La logique séquentielle nécessite une rétroaction dans la fonctionnalité du circuit.
-La logique séquentielle, faisant appel à la notion d'instant successifs, est souvent synchrone (néanmoins elle peut être asynchrone comme nous en verrons des exemples).

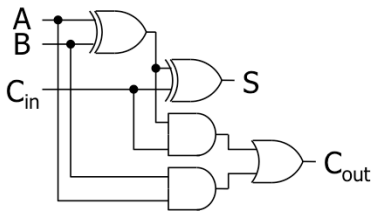
# La logique séquentielle est basée sur la notion d'états successifs

## Logique séquentielle

$D_{i+1}=D_i + 1$

| $t_i$ | $D_{base10}$ | $D_{base2}$ |
|---|---|---|
| $t_0$ | **0** | 0 0 0 |
| $t_1$ | **1** | 0 0 1 |
| $t_2$ | **2** | 0 1 0 |
| $t_3$ | **3** | 0 1 1 |
| $t_4$ | **4** | 1 0 0 |
| $t_5$ | **5** | 1 0 1 |
| ... | ... | ... |


## Logique combinatoire

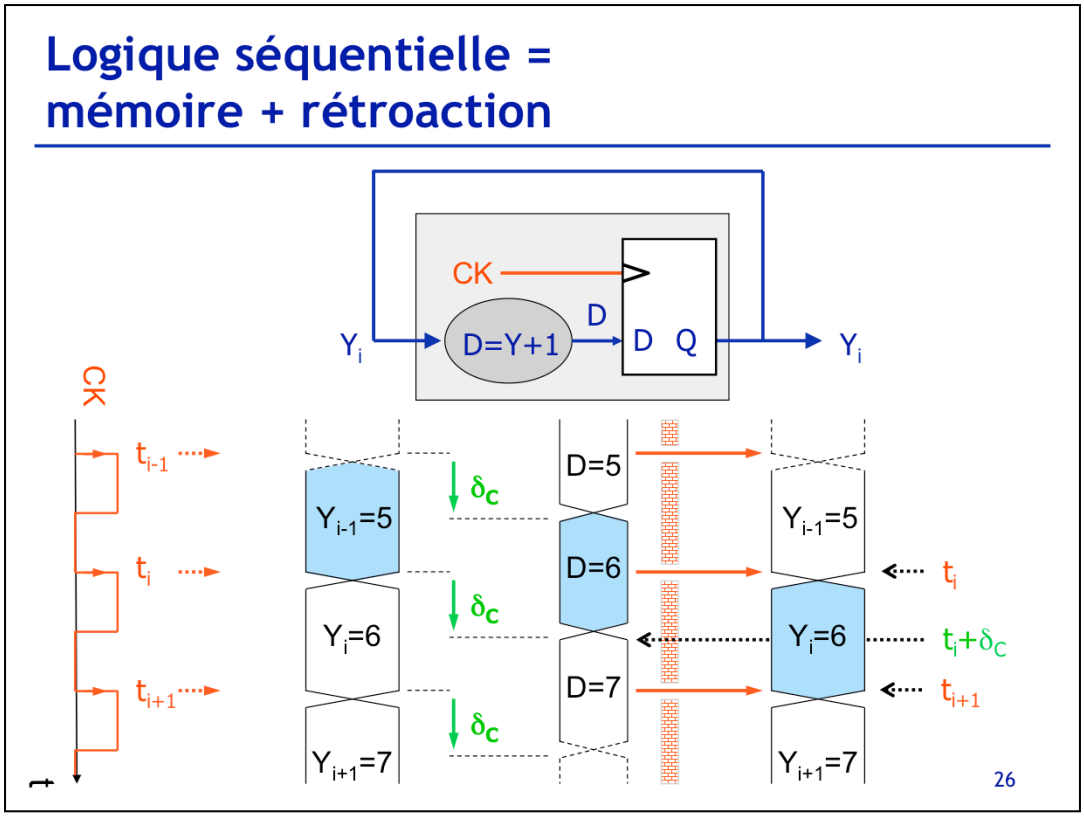



(full-adder)

25

A droite: un exemple de fonction combinatoire (l'addition de deux bits avec reports entrant et sortant: full-adder).

A gauche: un exemple de fonction séquentielle: compter. Le comptage peut être vu comme l'action d'ajouter, à chaque coup d'horloge, une unité à l'état passé du système. Cette fonction est typiquement traduite sous forme d'une équation itérative.



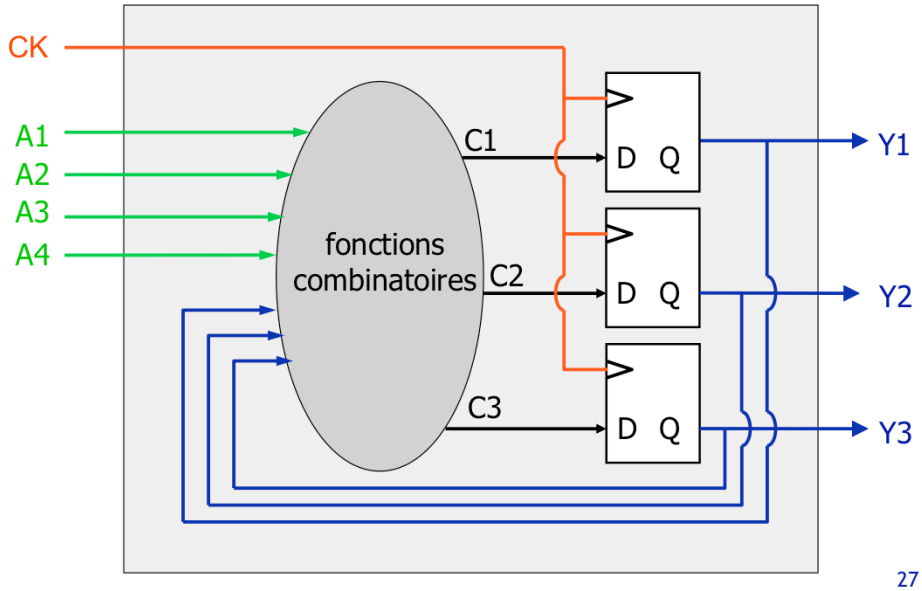

Voici un circuit –séquentiel- permettant de compter.

Il comporte:
-un bloc combinatoire (permettant d'ajouter une unité à la valeur d'entrée)
-un bistable D piloté par une horloge (c'est donc un circuit synchrone)
-une rétroaction de la sortie sur l'entrée, caractéristique des circuits séquentiels.

Voici comment fonctionne un tel montage:
-juste avant l'instant $t_i$, l'entrée du bloc combinatoire (qui est aussi la sortie du montage) vaut 5 et la sortie de ce bloc vaut 6 (les délais dus aux conditions de course sont passés et la sortie du bloc combinatoire est stabilisée). L'entrée du bistable vaut 6 et sa sortie 5.
-Au coup d'horloge $t_i$, le bistable recopie son entrée sur sa sortie. Sa sortie passe donc à 6, valeur qui est mémorisée par le bistable.
-Cette valeur est répercutée à l'entrée du circuit par la rétroaction. Le bloc combinatoire reçoit donc une nouvelle valeur d'entrée (6) sur base de laquelle il commence à calculer une nouvelle valeur de sortie (qui se stabilisera à 7 une fois les délais dus aux conditions de course passés). Le calcul du bloc combinatoire n'influence cependant pas la sortie Y du circuit car l'entrée d'activation du bistable n'est déjà plus active (elle n'est activée que pendant un temps beaucoup plus court que le délai de calcul du bloc combinatoire).
-juste avant l'instant $t_{i+1}$, l'entrée du bloc combinatoire (qui est aussi la sortie du montage) vaut 6 et la sortie de ce bloc vaut 7. L'entrée du bistable vaut 7 et sa sortie 6. Et on peut recommencer le raisonnement.

Comme on le voit:
-le bloc combinatoire a pour fonction de calculer le "prochain état du système"
-ce prochain état devient l'état présent au coup d'horloge, grâce au bistable

## Généralisation: circuit de base en logique séquentielle




Voici un schéma plus général de circuit séquentiel (on pourrait encore généraliser davantage en mettant plusieurs circuits de ce type en cascade).

Il comporte:
-des signaux d'entrée
-des fonctions combinatoires (calculant notamment les prochains états du circuit)
-des bistables pour mémoriser les états du circuit
-des rétroactions des sorties sur les entrées
-une horloge

# Chapitre 13:
# Logique synchrone et logique séquentielle

## 13.3.2 – Principales fonctions logiques séquentielles

## Principales fonctions de la logique séquentielle

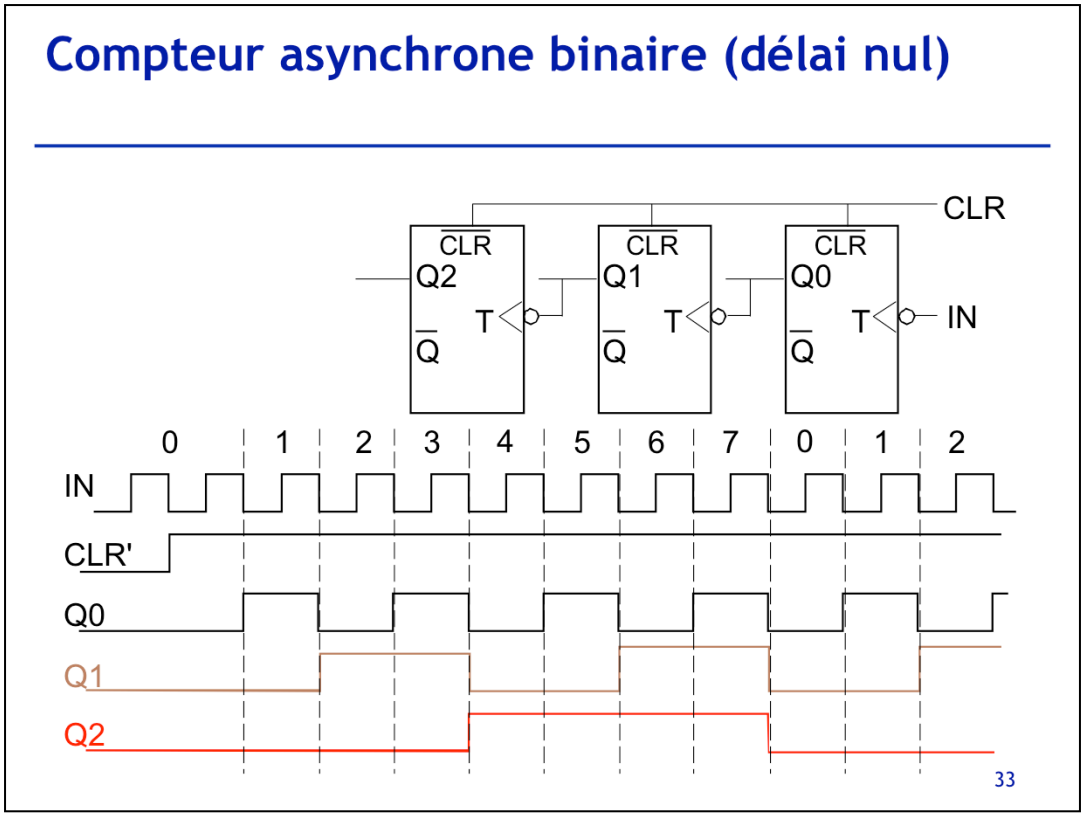
- (mémoire)
- tout traitement itératif
- compteurs
  - et division de fréquence
- (registres) et registres à décalage
- tous les systèmes séquentiels
  - machines d'états
- notamment: informatique
  - microprocesseurs et autres

32

La plupart des équipements qui nous entourent sont en fait séquentiels (très peu de fonctions complexes sont purement combinatoires).

Les principales fonctions "de base" en logique séquentielle sont:
-les mémoires "pures" (mises ci-dessus entre parenthèses), dont les registres
-les registres à décalage (qui, en plus de mémoriser, réalisent un traitement via le décalage)
-les compteurs de tout type
-tous les systèmes qui peuvent être décrits par une ou plusieurs équations itératives
-toutes les machines d'états

Les pages suivantes montrent la réalisation d'un compteur simple (circuit séquentiel) en logique asynchrone et en logique synchrone.

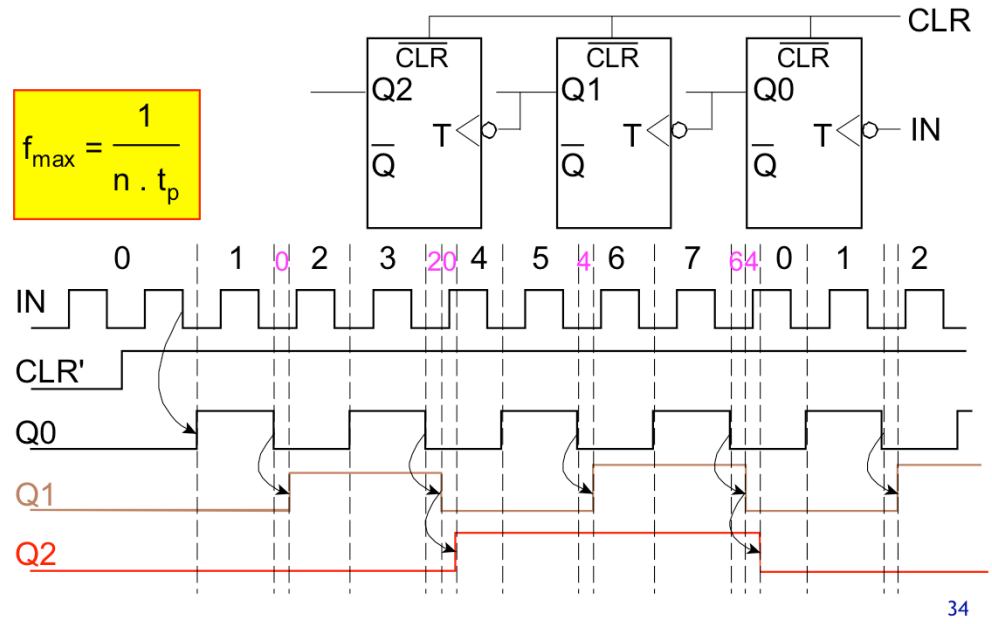



Le compteur le plus simple que l'on puisse réaliser est le compteur binaire asynchrone. Pour ce faire, on cascade des bistables T en connectant le signal d'entrée à la borne T du premier bistable, puis chaque sortie Q à l'entrée T du bistable suivant. On a choisi ici des bistables actifs sur le flanc descendant de l'horloge.

L'utilité de l'entrée CLR' apparaît : on l'active pour mettre toutes les sorties à 0 au départ.
Dès que l'on désactive CLR'
- le premier flanc descendant du signal d'entrée provoque le basculement de Q0 à HI
- le flanc suivant fait passer Q0 à LO, ce qui donne un coup d'horloge au bistable suivant et Q1 à HI
- et ainsi de suite...

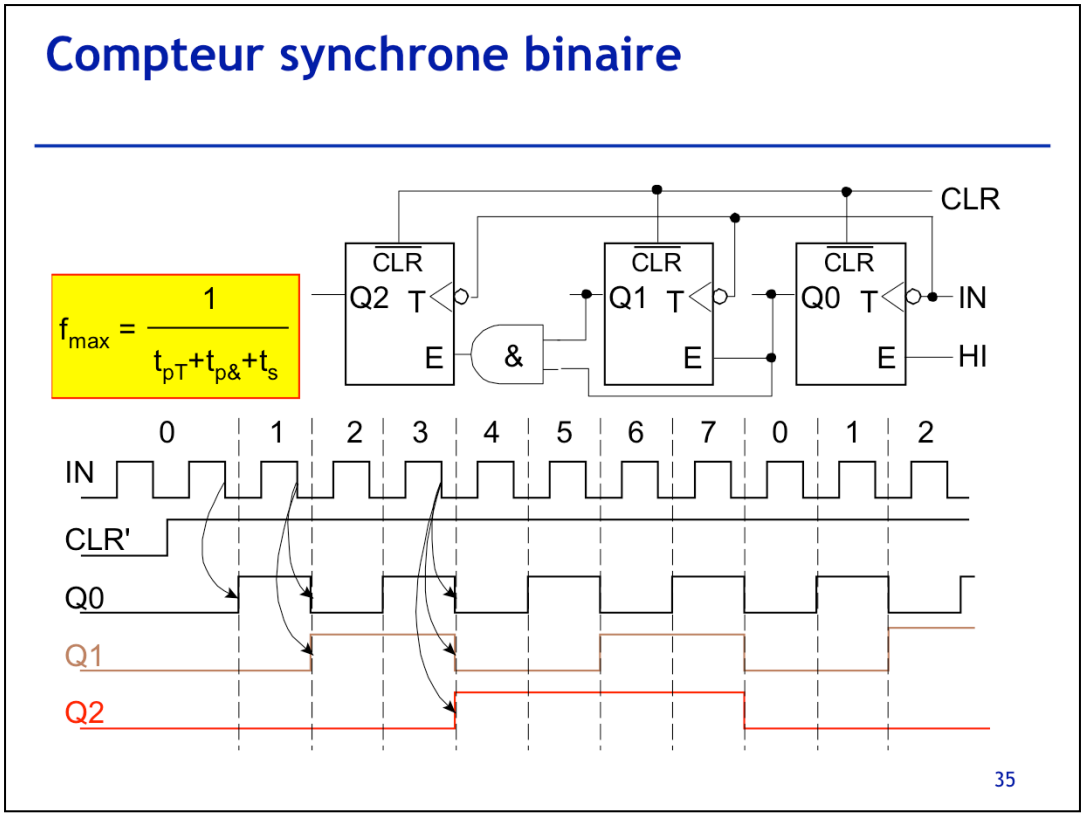
On parle de compteur binaire puisque chaque étage divise la fréquence du signal par deux. Le nombre binaire formé des 3 bits {Q2,Q1,Q0} est égal (en binaire) au nombre de flancs actifs du signal d'entrée depuis la désactivation du CLR.
Il s'agit d'un compteur MODULO $2^N$, où N est le nombre d'étages du compteur.

## Compteur asynchrone binaire




Il s'agit du même compteur qu'à la page précédente, mais on considère ici que les bistables ont un délai non nul: on voit apparaître dans le signal de sortie des états parasites temporaires.

L'existence de ces états parasites limite la fréquence maximale du signal d'horloge applicable au compteur.




Réalisons maintenant un compteur synchrone: il faut compliquer le montage et disposer de bistables JK (voir chapitre correspondant) ou de bistables T munis d'une entrée d'activation supplémentaire (l'entrée E doit être active à l'état haut pour que le bistable T change d'état.).

Si l'on examine la séquence binaire normale à 3 bits

000
001
010
011
100
101
110
111

on constate:
- que le bit de poids faible change d'état à chaque étape;
- qu'un bit de poids fort change lorsque tous les bits de poids plus faible sont à 1.

On en déduit le schéma présenté ci-dessus, dans lequel les portes supplémentaires aboutissant sur les entrées E réalisent la condition ci-dessus. Le synchronisme est assuré par la distribution de l'horloge à toutes les entrées T.

Au prix d'un câblage plus complexe, le compteur synchrone est plus avantageux que l'asynchrone:
- les états parasites ont disparu, puisque toutes les sorties changent d'état "simultanément" (à la dispersion près sur le temps de propagation)
- la fréquence maximum du compteur ne dépend plus du nombre d'étages; la contrainte est que le nouveau flanc d'horloge ne peut pas intervenir avant que les entrées E ne soient stabilisées, soit une période minimum

$T > t_{pT} + t_{p\&} + t_s$

où tpT, tp& et ts sont respectivement les temps de propagation du bistable T et de la porte AND et le "setup time" du bistable.

On voit bien sur le schéma ci-dessus la principale caractéristique qui permet de reconnaître un circuit synchrone: tous les bistables sont alimentés par la même horloge (ce qui n'était pas le cas dans le compteur asynchrone).

## Les circuits complexes (1): logique câblée

- ASIC = Application Specific Integrated Circuit
  - puce conçue spécifiquement pour une application donnée
  - analogique ou numérique
  - performances optimales
  - flexibilité inexistante
  - coût uniquement justifié pour de très grandes séries
- => on ne peut pas tout faire en ASICs!

41

Pour faire des circuits séquentiels complexes (ce qui est le cas dans la plupart des montages d'électronique numérique), on a fondamentalement trois possibilités:

1) utiliser de la logique "câblée"

Ceci consiste à faire fabriquer un circuit spécifique (ASIC) qui remplit la fonction voulue. En termes de performances, c'est la meilleure solution. C'est également le plus cher.

En pratique, cette solution n'est viable que pour des très grandes séries (centaines de milliers d'exemplaires au moins).

## Les circuits complexes (2): logique programmée

- microprocesseurs (µP)
  - principe: fonction définie par un programme
- propriétés
  - performances faibles (par rapport à ASIC)
  - flexibilité maximale
- => intéressant pour des applications très changeantes

42

2) une solution beaucoup plus courante consiste à utiliser un microprocesseur, qui relève de la logique "programmée".

L'utilisation d'un programme (voir chapitre correspondant) permet une souplesse d'utilisation beaucoup plus grande, mais les performances sont intrinsèquement nettement inférieures qu'en logique câblée.

## Les circuits complexes (3): logique programmable

- FPGA = Field Programmable Gate Array
  - "Gate Array" = matrice de portes logiques
  - "Field Programmable" = dont les connexions sont programmables par l'utilisateur
  - fonction sur mesure
  - compromis flexibilité/performances
- applications
  - prototypage avant ASIC
  - "petites" séries (moins cher qu'un ASIC)
  - "time-to-market" / produits à courte durée de vie
- variante: PLD = Programmable Logic Device

43

Enfin il existe une troisième possibilité, très couramment utilisée aujourd'hui: la logique *programmable* (à ne pas confondre avec la logique *programmée* !), principalement représentée par une gamme de circuits appelés FPGA.

Les FPGAs peuvent être configurés (et reconfigurés un certain nombre de fois) pour exécuter une fonction spécifique. Ils offrent un compromis intéressant entre flexibilité et performance.

En pratique, on trouve souvent des systèmes hybrides (mélange de deux ou trois des logiques précédentes), qui permettent d'exploiter au mieux les avantages de chaque technologie.

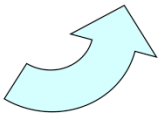
# Chapitre 13:
# Logique synchrone et logique séquentielle

## 13.4 – Synthèse

## Les types de logique: synthèse

| | asynchrone<br>temps continu<br>conditions de course | synchrone<br>temps discret<br>horloge/entrée activation |
|---|---|---|
| **combinatoire**<br>états concurrents<br>sans rétroaction | • portes logiques<br>• fonctions logiques | • logique synchrone<br>• pipelines |
| **séquentielle**<br>états successifs<br>rétroaction | • (bistable RS/D)<br>• (compteur asynchrone) | • calcul itératif<br>• machines d'état |

mémoires



45

N.B.: la majorité des systèmes *numériques* sont synchrones et séquentiels…

Et s'il faut mettre la mémoire "pure" (bistables) quelque part, c'est plutôt dans cette catégorie également.

# Applications réelles

- Un vrai système, c'est
  - de l'analogique
  - du numérique séquentiel (synchrone)
    - contient combinatoire asynchrone
    - mémoires
- et en plus: flexible
  - logique programmée ou programmable

46