

Les systèmes séquentiels : registres et compteurs

Qu'appelle-t-on fonctions standards

- Modules logiques renfermant une fonctionnalité simple
- Ces fonctionnalités correspondent à des éléments logiques fréquemment utilisés
- Ces modules se retrouvent fréquemment sous une forme simple ou combinée dans les circuits

Fonctions standards séquentielles

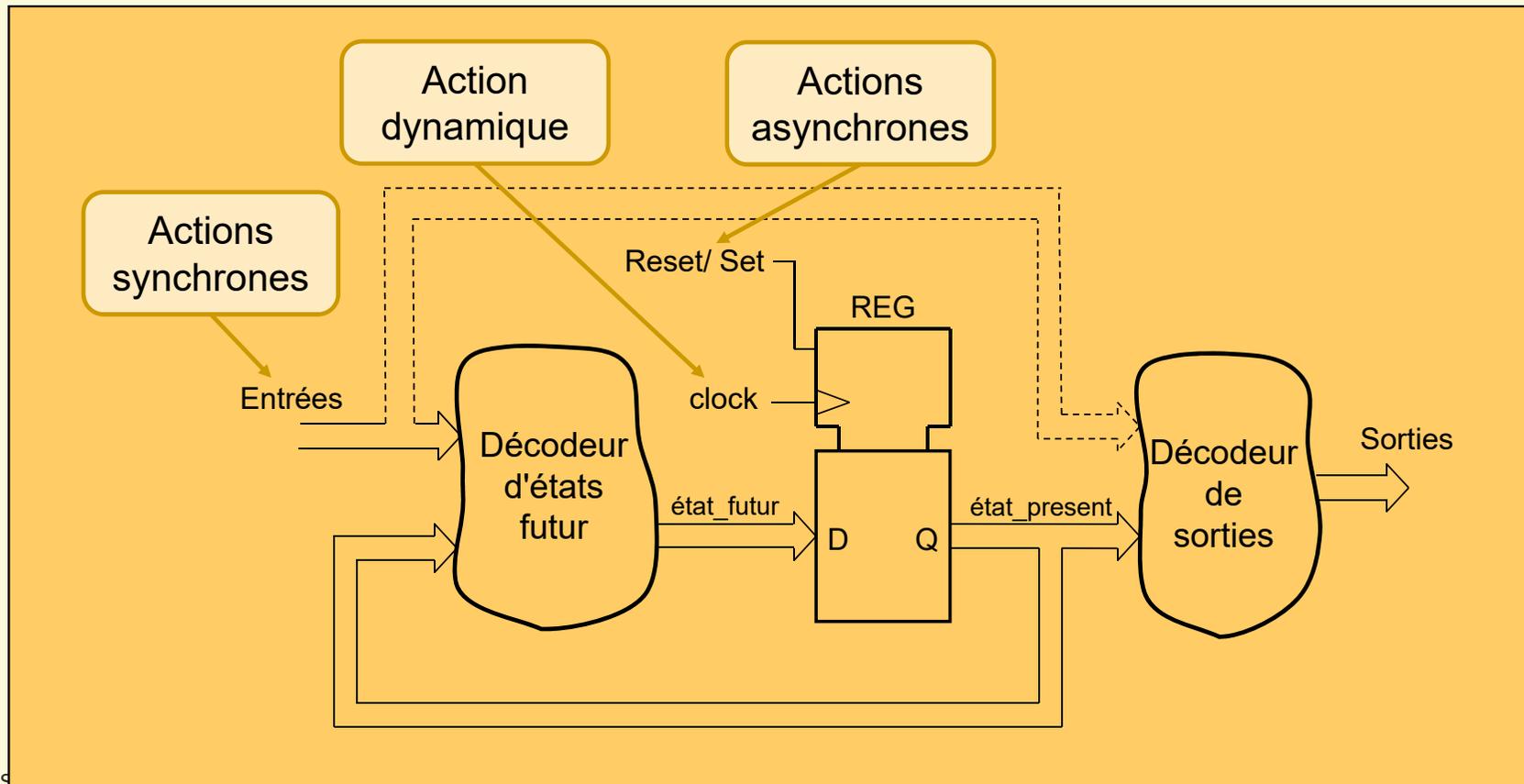
- Registres
 - parallèle - parallèle
 - série - parallèle
 - parallèle - série
 - mélange des fonctions série et parallèle
- Compteurs
 - Séquence : binaire, BCD, modulo N,
 - Mode : chargement parallèle, compte, décompte, ...

Systeme séquentiel: types d'entrées

- Un système séquentiel comprend **3 types** d'entrées :
 - Les entrées à action **asynchrone** :
 - action immédiate sur le registre (reset, set)
⇒ action immédiate sur les sorties
 - signaux **directement** connectés sur le registre
 - Une entrée à action **dynamique** :
 - entrée nommée clock (horloge)
 - définit l'instant où l'état interne change ⇒ **actions synchrones**
 - entrée connectée sur l'entrée d'horloge (sensible au flanc) du registre
 - Les entrées à **action synchrone** :
 - action réalisée au flanc d'horloge sur les sorties
 - ces signaux sont connectés sur le décodeur d'actions futurs

Systeme séquentiel: schéma bloc

- Un système séquentiel comprend **3 types** d'entrées :

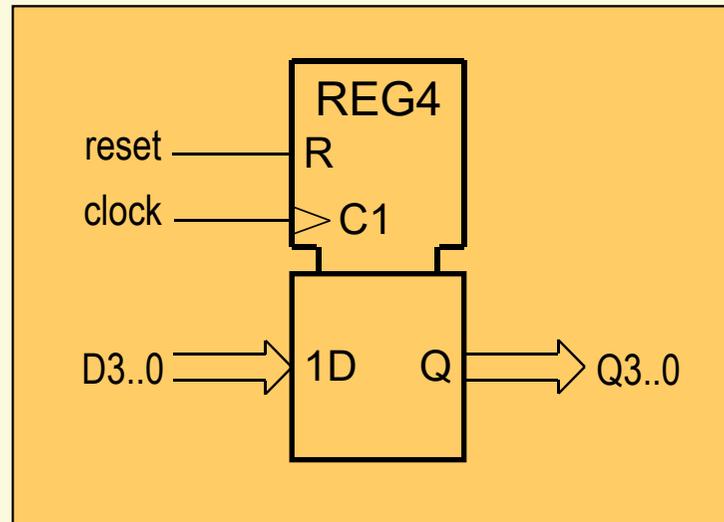


Registres synchrones

- Fonctions :
 - Mémorisation d'une information
 - Décalage d'un flip-flop à l'autre
- Utilisé comme :
 - Mémorisation
 - Conversion série-parallèle
 - Générateur de séquence
 - Diviseur de fréquence
 - Compteur
 -

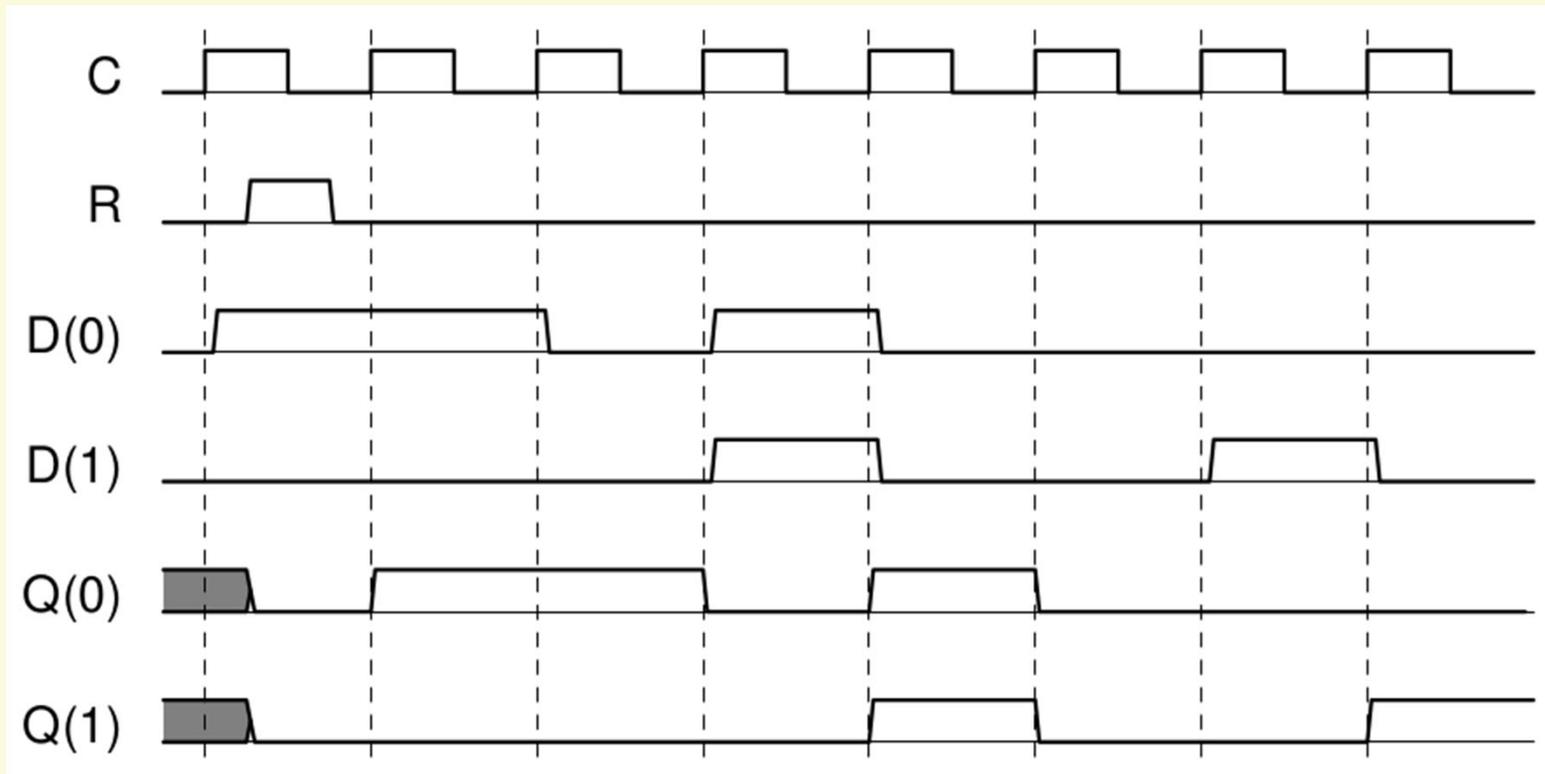
Registre simple

- Le registre simple est un ensemble de bascules en parallèle
Voici le symbole CEI:



- Il est le composant de base des systèmes séquentiels (état interne)
- La conception des systèmes séquentiels consistera principalement à établir l'évolution de la valeur à fournir sur l'entrée D du registre.

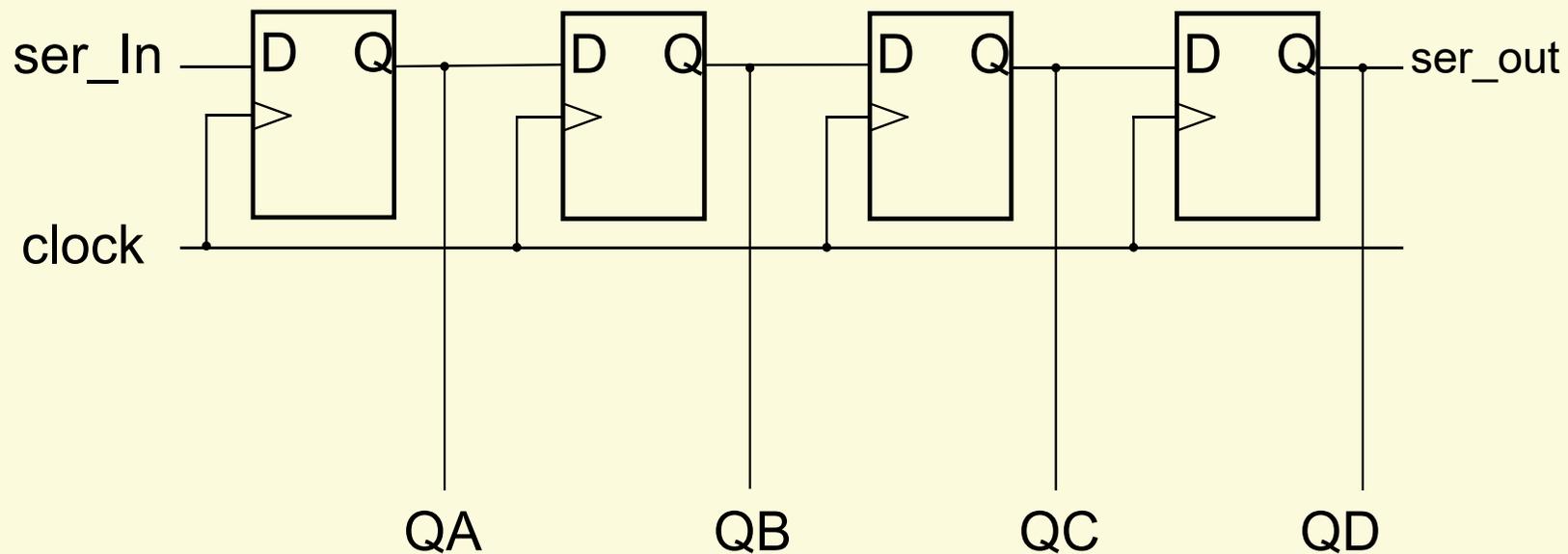
Registre simple: chronogramme



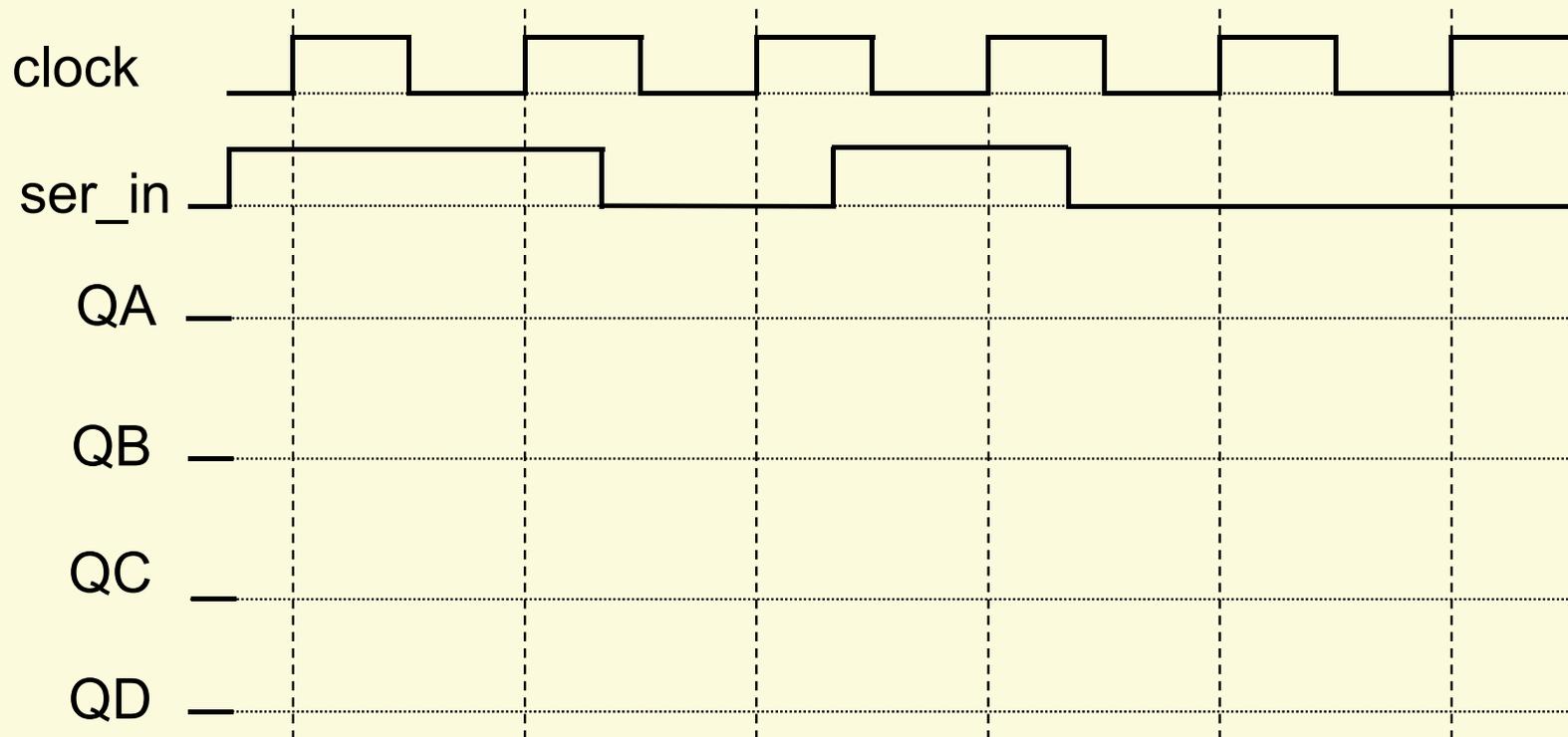
source: présentation reg/cpt Yann Thoma

Registre à décalage : principe

- Le registre à décalage décale son contenu à chaque cycle d'horloge



Chronogramme registre à décalage



Exercices série I

1. Concevoir un registre à décalage de 4 bits ayant les fonctionnalités suivantes :

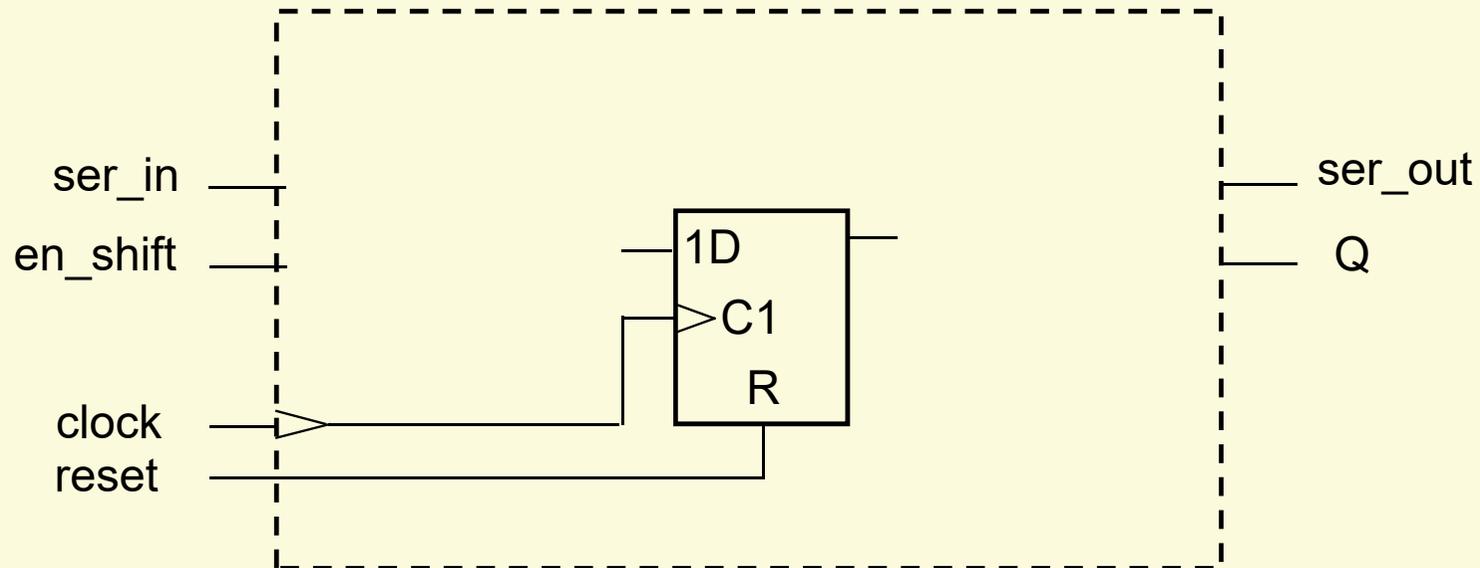
- Si *en_shift* est actif : le contenu du registre est décalé à droite
- Si *en_shift* est inactif : l'état du registre est maintenu

a) concevoir une cellule (1 bit), puis

b) réaliserez un registre 4 bits en cascasant 4 cellules

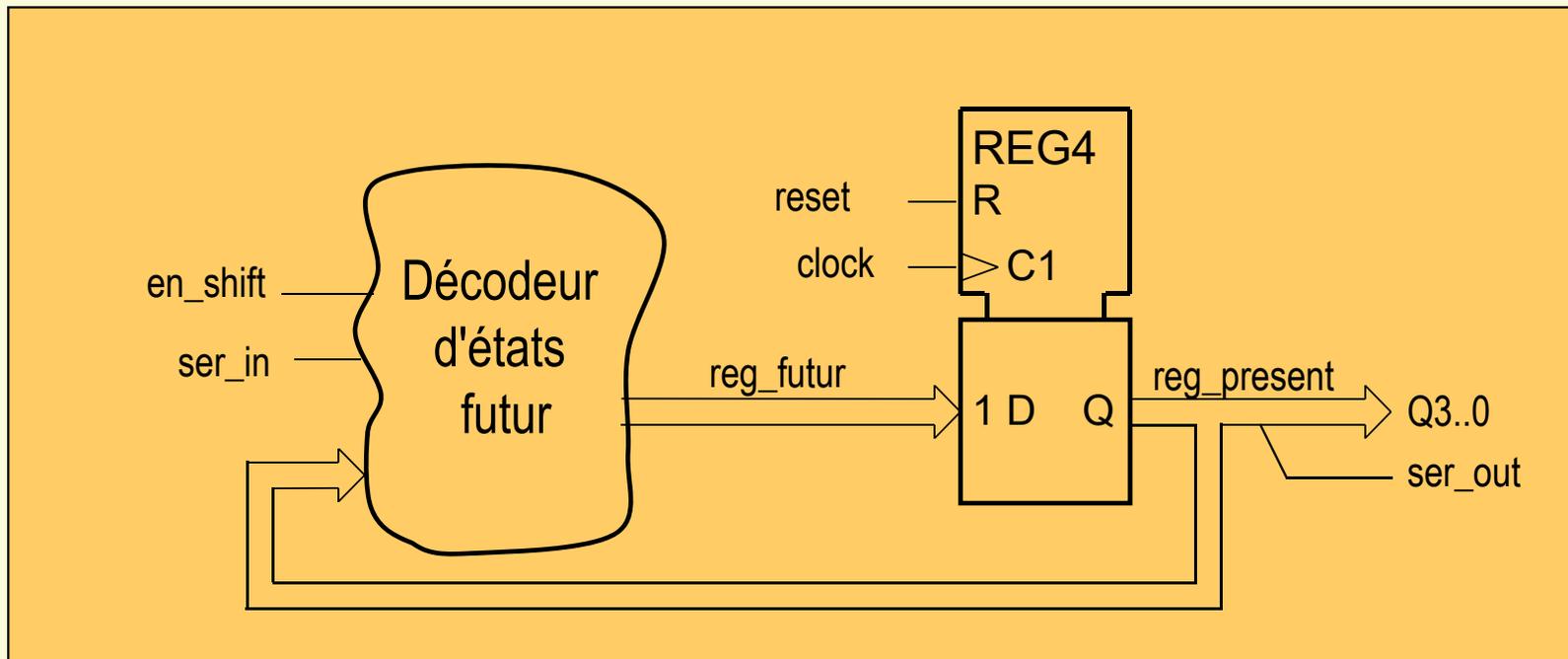
... suite exercice I

1. Donner le schéma d'une cellule, puis chaîner 4 cellules.



Exercices série I

2. Réaliser le registre à décalage de 4 bits selon la décomposition d'un décodeur d'état futur et d'un registre // de 4 bits
- Donner le schéma du décodeur d'état futur



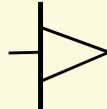
Registres synchrones : caractéristiques

- Nombre de bits
- Modes de fonctionnement

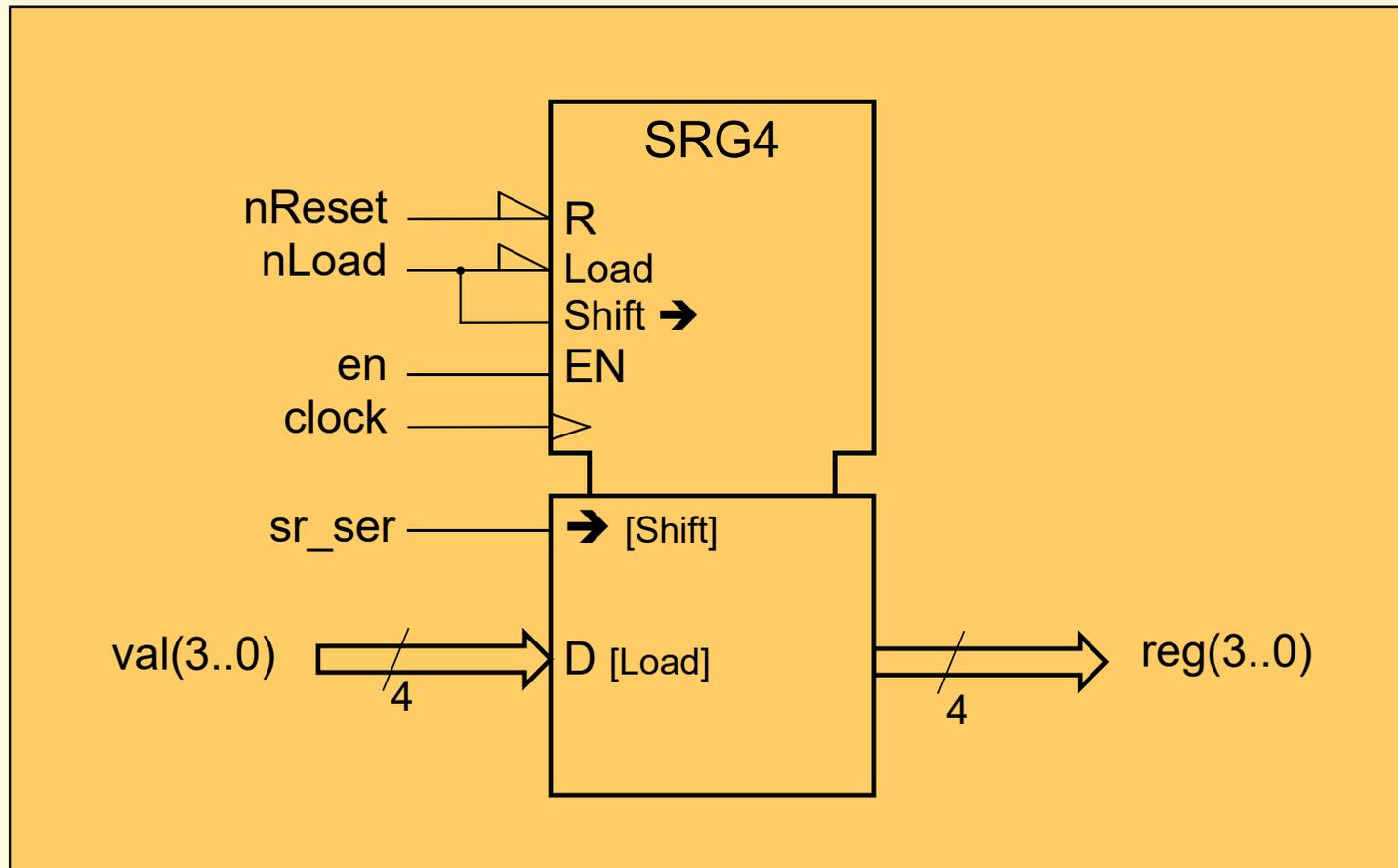
| entrée | sortie | Fonctionnement |
|-----------|-----------|--------------------------|
| parallèle | parallèle | mémorisation |
| parallèle | série | convers. parallèle-série |
| série | parallèle | convers. série-parallèle |
| série | série | peu d'utilité, sauf FIFO |

+ modes de fonctionnement combinés

Registres synchrones : symboles

- SRGm registre à décalage m bits (shift register)
- entrées de mode de fonctionnement:
 - chargement parallèle **load**
 - décalage droite → poids fort vers poids faible
 - décalage gauche ← poids faible vers poids fort
- entrée d'autorisation enable: **EN**
- entrée de remise à zéro asynchrone reset: **R**
- entrée d'horloge (action dynamique): 

Registre SRG4 : symboles ...



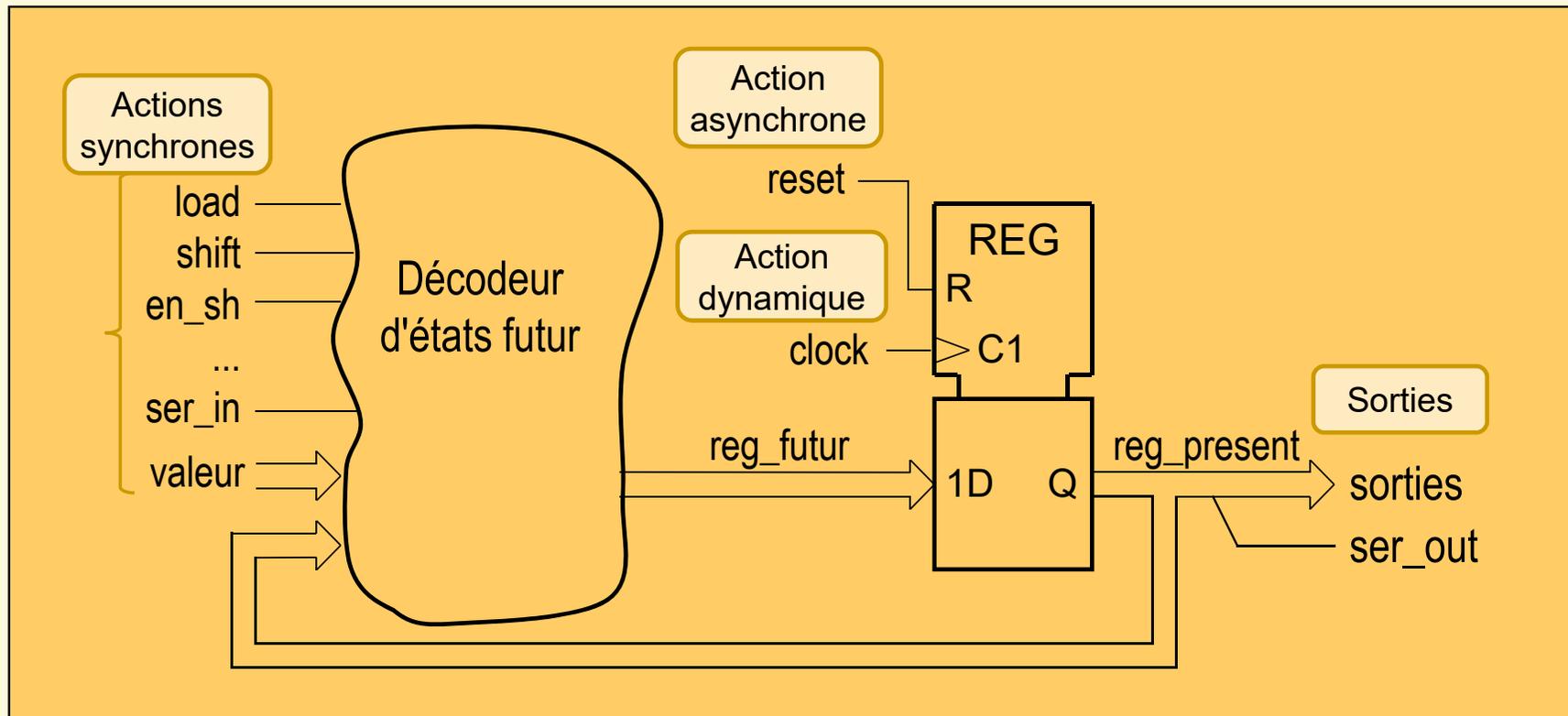
... registre SRG4 : symbole

- SRG4 ⇒ registre à décalage de 4 bits
- R ⇒ remise à 0 asynchrone,
- Load ⇒ chargement synchrone (indépendant enable)
- Shift ⇒ mode de décalage, mutuellement exclusif avec le chargement, actif seulement si enable
- EN ⇒ autorise la fonction décalage à droite
- →[Shift] ⇒ entrée série, valeur inséré dans MSB lors décalage
- D [Load] ⇒ entrée de chargement parallèle

- Clock  ⇒ entrée d'horloge, action dynamique

Registre synchrone: schéma bloc :

- Schéma bloc selon la décomposition d'un système séquentiel



SRG4: analyse du fonctionnement

- Fonctions asynchrones:
 - nReset: remise à zéro asynchrone (R)
- Fonctions synchrones:
 - si load actif chargement // de val
 sinon si en actif décalage à droite avec ser_in
 sinon maintien

SRG4: table des fonctions

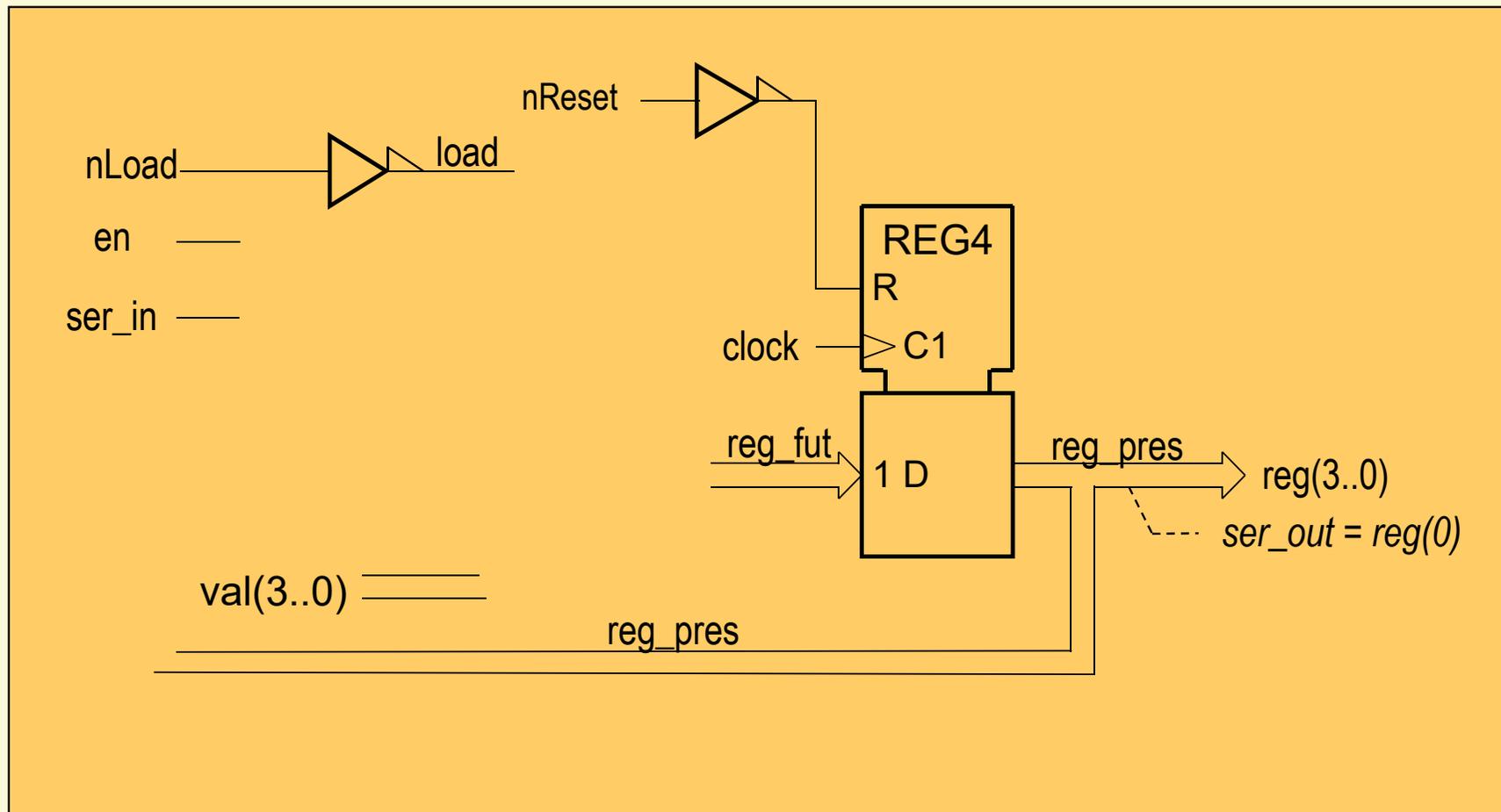
- Table de fonctions synchrones:

| load | en | sr_ser | Fonction |
|------|----|--------|--|
| 1 | - | - | reg = val; chargement |
| 0 | 1 | - | reg = ser_in & reg(3..1) décalage à droite |
| 0 | 0 | - | reg = reg; maintien |

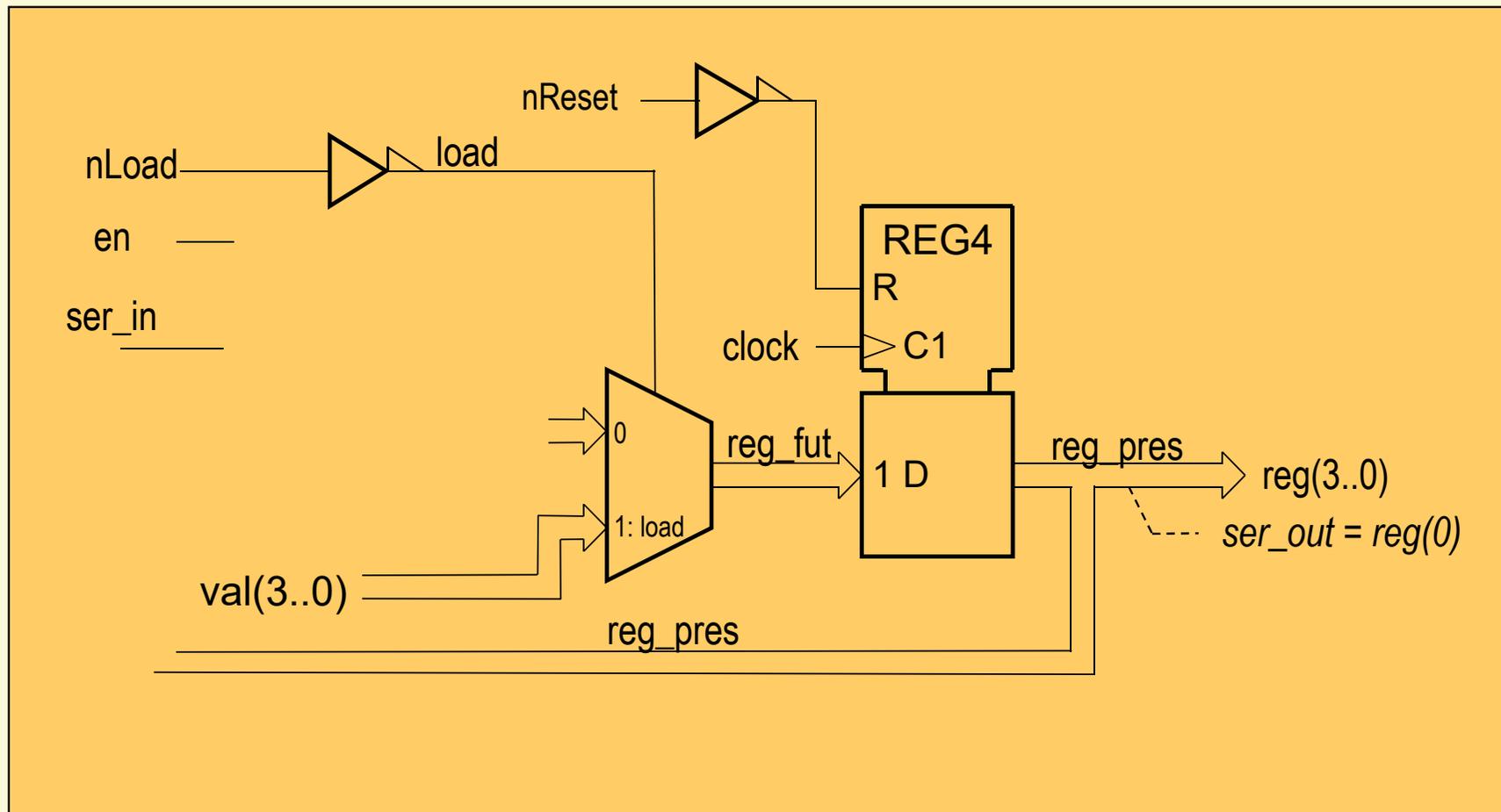
en explicitant reg_fut et reg_pres (recommandé) :

| load | en | reg_fut | Fonction |
|------|----|---------------------------|-------------------|
| 1 | - | = val | chargement |
| 0 | 1 | = ser_in & reg_pres(3..1) | décalage à droite |
| 0 | 0 | = reg_pres | maintien |

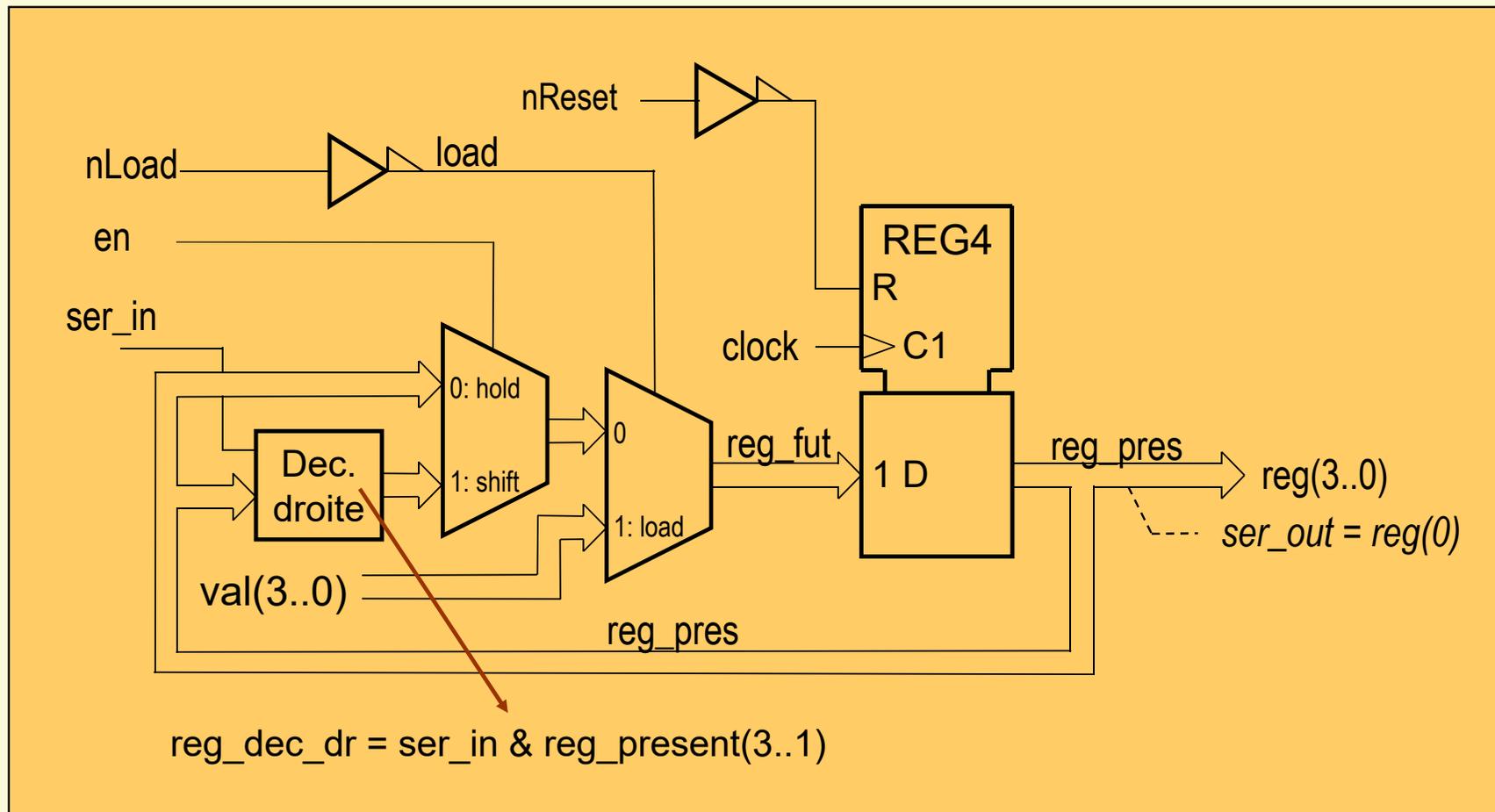
SRG4: schéma bloc de la décomposition



SRG4: schéma bloc de la décomposition



SRG4: schéma bloc de la décomposition



Conception de registres : Méthodologie

1. Identifier les fonctions du registre à décalage
2. Lister ces fonctions (classer synchrones et asynchrones)
3. Définir les fonctions asynchrones (prioritaires)
4. Fixer les priorités des fonctions synchrones
5. Etablir la table des fonctions synchrones

Exemple de table

| Ctrl1 | Ctrl2 | ... | reg_pres | reg_fut | Fonction |
|------------|-------|-----|----------|------------|-----------------------|
| 1 | - | ... | - | = val | Chargement |
| 0 | 1 | ... | - | = ... | Une certaine fonction |
| 0 | 1 | ... | = 10 | = ... | Une autre fonction |
| ... | ... | ... | ... | | ... |
| autres cas | | | | = reg_pres | Maintien |

6. Etablir le schéma du registre (conseil: utiliser cascades de mux2a1)
7. Réaliser la description VHDL du registre

Exercices série II

- Pour chaque exercice de la série II, établir les points suivants:
 - Analyser le système séquentiel à réaliser
 - Etablir la liste des fonctions (asynchrones et synchrones)
 - Etablir la table des fonctions synchrones dans l'ordre de priorité.
 - Etablir le schéma logique selon la décomposition d'un système séquentiel simple

Exercices série II

1. Concevoir le système séquentiel, en suivant la décomposition d'un système séquentiel simple, permettant de générer la séquence ci-dessous:



Exercice série II

2. Concevoir un système séquentiel, en suivant la décomposition d'un système séquentiel simple, permettant de générer la séquence donnée ci-dessous:

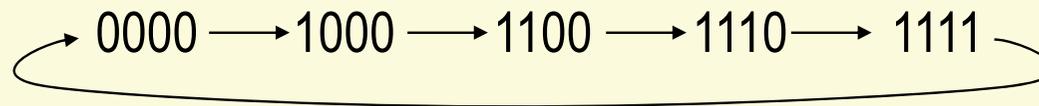


Exercice série II

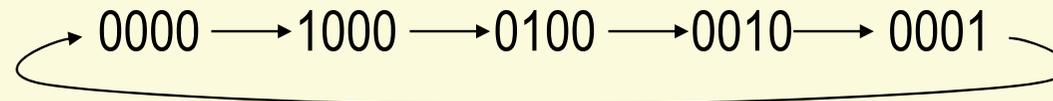
3. Concevoir un système séquentiel répondant au cahier des charges ci-dessous, en suivant la décomposition d'un système séquentiel simple :

- Si *enable* est actif alors :

Si *seq_plus* est actif alors le système parcourt la séquence A suivante :



sinon le système parcourt la séquence B suivante :



sinon le système reste dans l'état actuel (maintien)

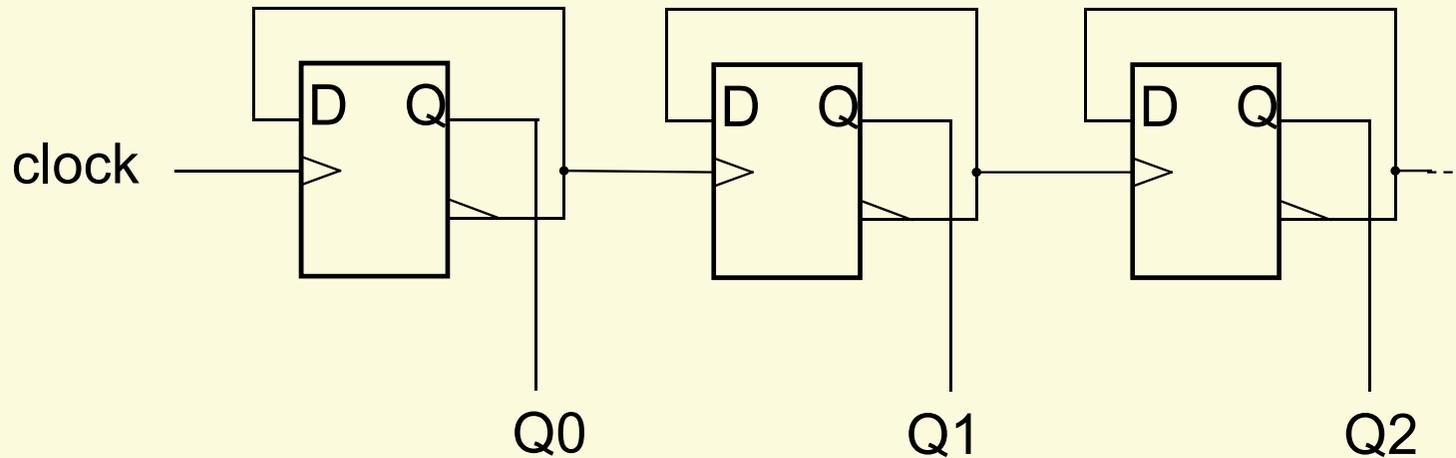
Qu'appelle-t-on compteur

- On appelle compteur un module logique séquentiel capable de faire évoluer ses sorties en fonction des entrées et d'une séquence prédéfinie.
- L'instant où a lieu le changement des sorties dépend d'une entrée appelée horloge (ou clock)

Structure de compteur

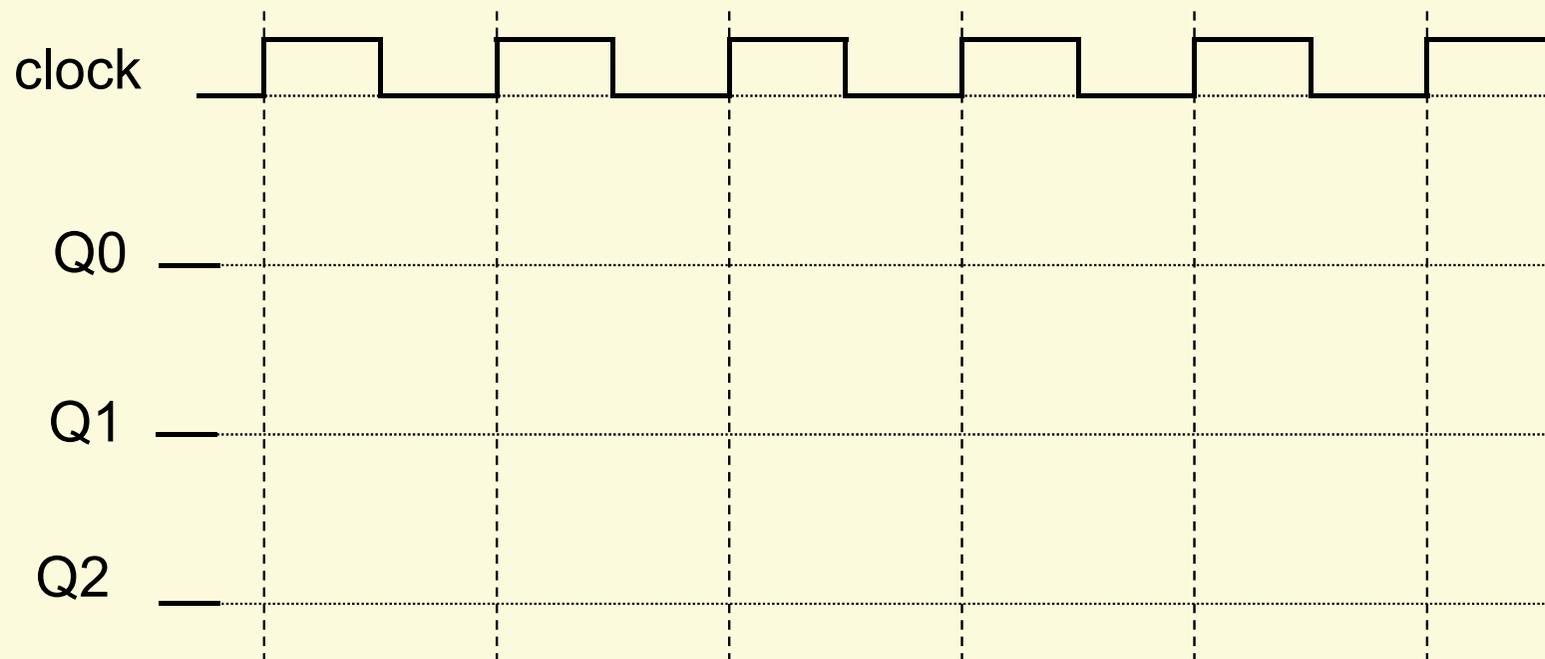
- Asynchrone (ou pseudo-synchrone)
 - Horloge d'un flip-flop dépend du ou des flip-flops précédents
- Full synchrone
 - Le même signal d'horloge est connecté sur tous les flip-flops => tous synchronisés avec même horloge :
d'où le terme de système **full synchrone**

Compteur asynchrone (ripple counter)



- Compteur à structure asynchrone
- Nommé parfois "Compteur pseudo-synchrone"

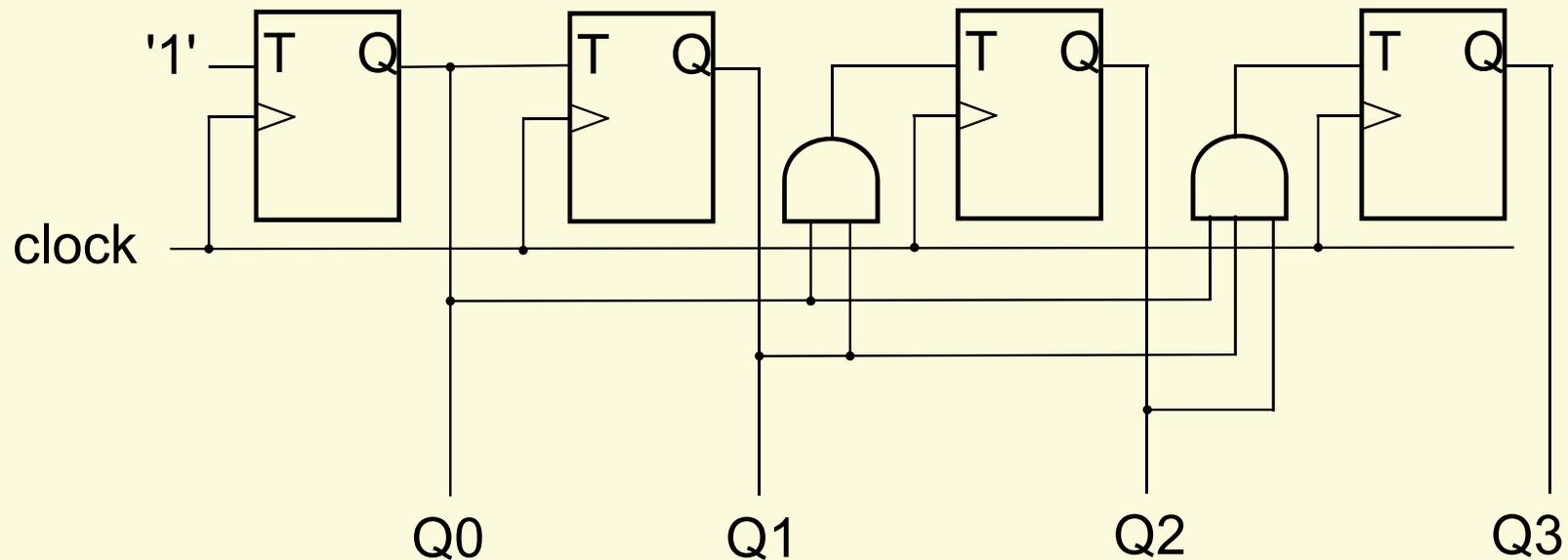
Chronogramme compteur asynchrone



Analyse du compteur asynchrone

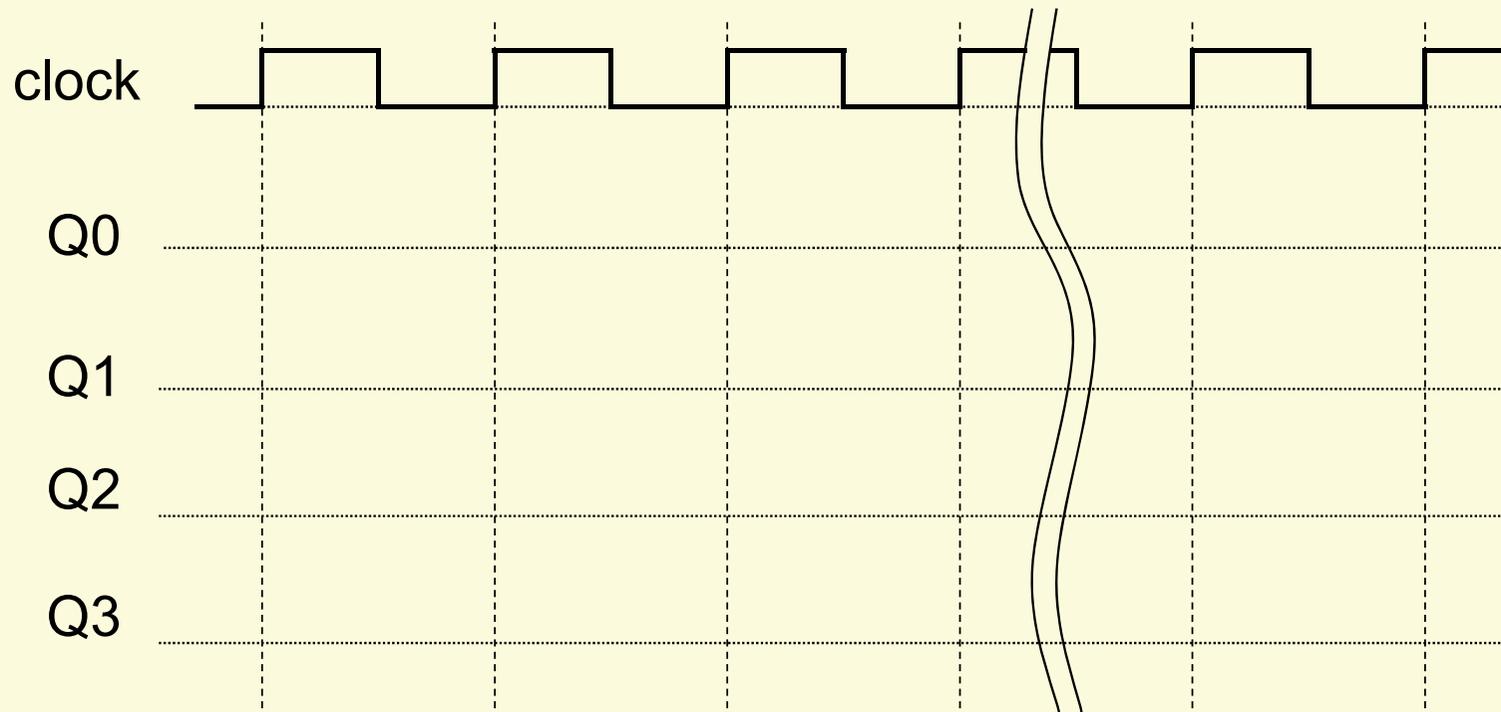
- Construction très simple
- Pas synchrone
 - Retard variable pour chaque sortie
 - Retard de la cellule N : $T_n = N \times t_p$
 - Très difficile à tester (vérification timing)
 - Possible que des sorties changent après flanc suivant de l'horloge
- Très mauvaise intégration dans des PLDs
- Utilisation **NON RECOMMANDÉE** avec les circuits logiques programmables (CPLD et FPGA)

Compteur "full synchrone"



- Compteur à structure "full synchrone":
tous les flip-flops sont contrôlés par la **même** horloge

Chronogramme compteur "full synchrone"



Analyse du compteur full synchrone

- Construction plus complexe
- Synchrone
 - Retard identique pour chaque sortie indépendant de la taille du compteur
 - Vérification fréquence maximum simple :
$$F_{\max} = 1 / (t_{p_TFF_{\max}} + t_{p_COMB_{\max}} + t_{\text{set-up}}_{TFF_{\max}})$$
 - Fonctionnement **garanti** tant que $F < F_{\max}$
- Très bonne intégration dans des PLDs

Full synchrone & PLD ?

- Dans PLD il y a un arbre d'horloge pré-cablé
- L'arbre garanti que **TOUS** les flip-flops voient l'horloge au même instant
 - skew (décalage) doit être être le plus faible possible
 - skew inférieur aux tp internes
 - caractéristiques pour Cyclone II de Altera

Cyclone II, Altera

Table 5-35. Clock Network Specifications

| Name | Description | Max | Unit |
|---|--|------|------|
| Clock skew adder EP2C5, EP2C8(1) | Inter-clock network, same bank | ±88 | ps |
| | Inter-clock network, same side and entire chip | ±88 | ps |
| Clock skew adder EP2C15, EP2C20, EP2C35, EP2C50, EP2C70 (1) | Inter-clock network, same bank | ±118 | ps |
| | Inter-clock network, same side and entire chip | ±138 | ps |

Note to Table 5-35:

(1) This is in addition to intra-clock network skew, which is modeled in the Quartus II software.

Table 5-16. LE_FF Internal Timing Microparameters (Part 1 of 2)

| Parameter | -6 Speed Grade (1) | | -7 Speed Grade (2) | | -8 Speed Grade (2) | | Unit |
|-----------|--------------------|-----|--------------------|-----|--------------------|-----|------|
| | Min | Max | Min | Max | Min | Max | |
| TCO | 141 | 250 | 141 | 277 | 135 | 304 | ps |
| | | | | | 141 | | ps |

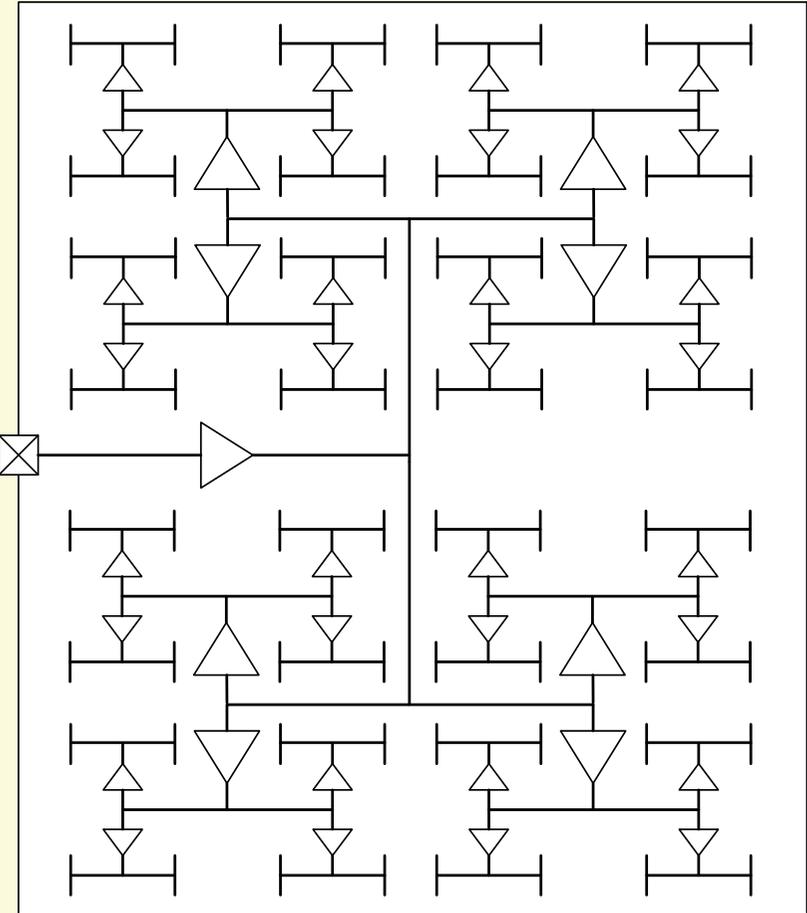
Cyclone II, Altera
TCO: clock-to-out

Full synchrone & PLD ?

Structure d'un arbre d'horloge

- Chemin équilibré afin de garantir un temps de propagation constant pour tous les flip-flops

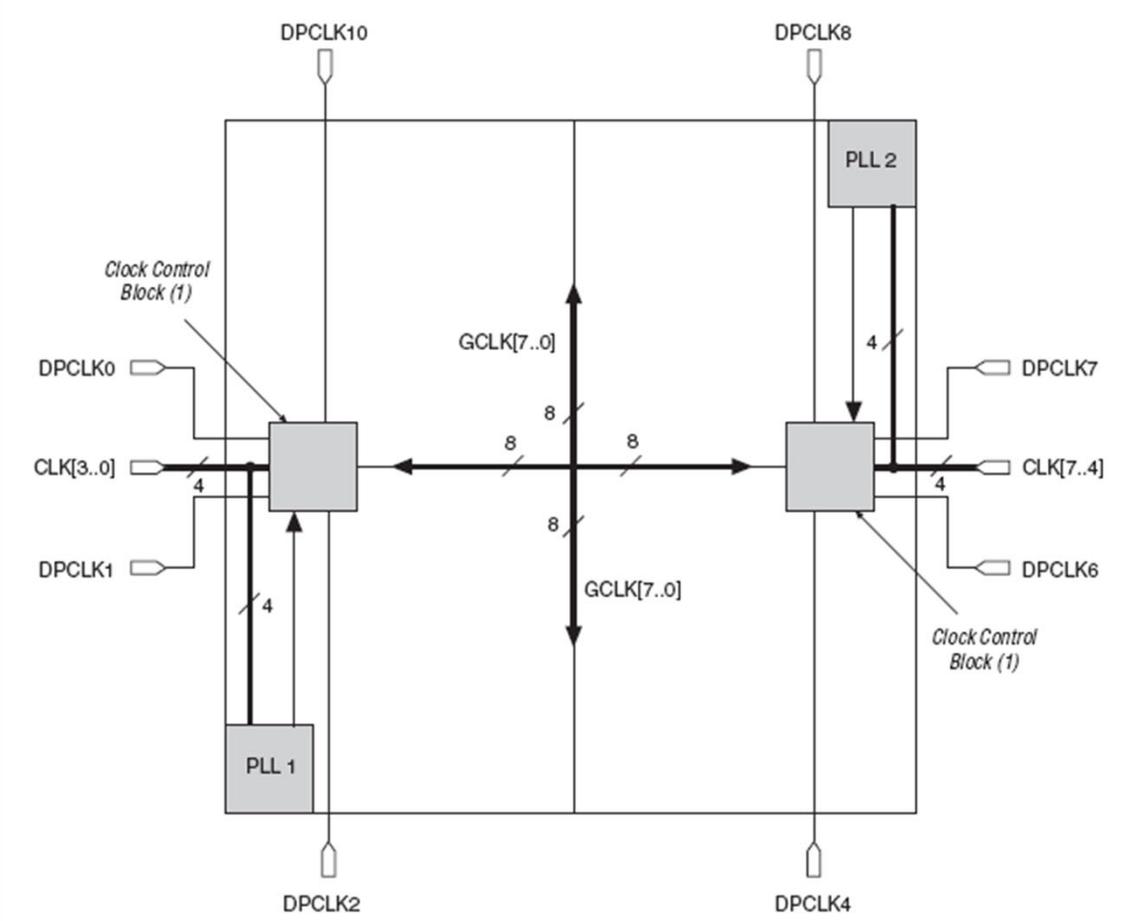
Entrée d'horloge



Full synchrone & PLD ?

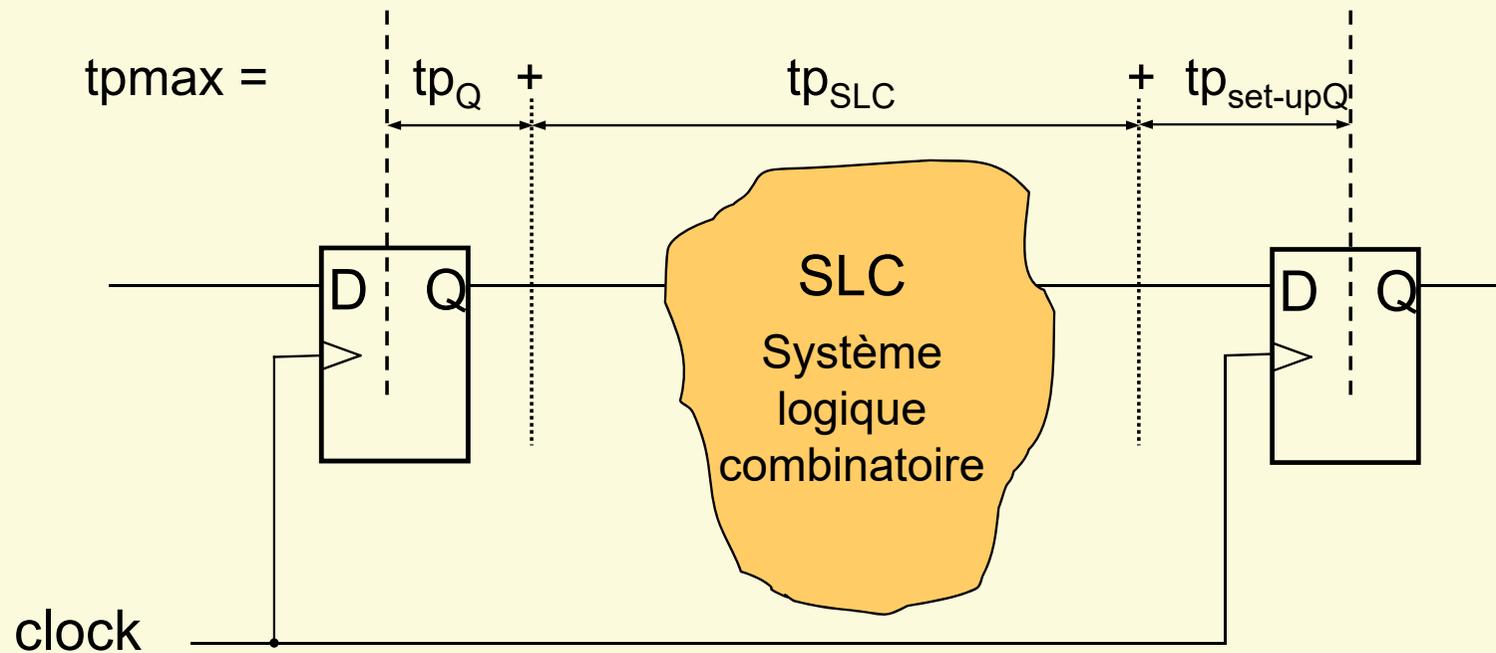
Exemple d'arbre d'horloge Cyclone II

Figure 2-11. EP2C5 & EP2C8 PLL, CLK[], DPCLK[] & Clock Control Block Locations



Analyse statique des temps

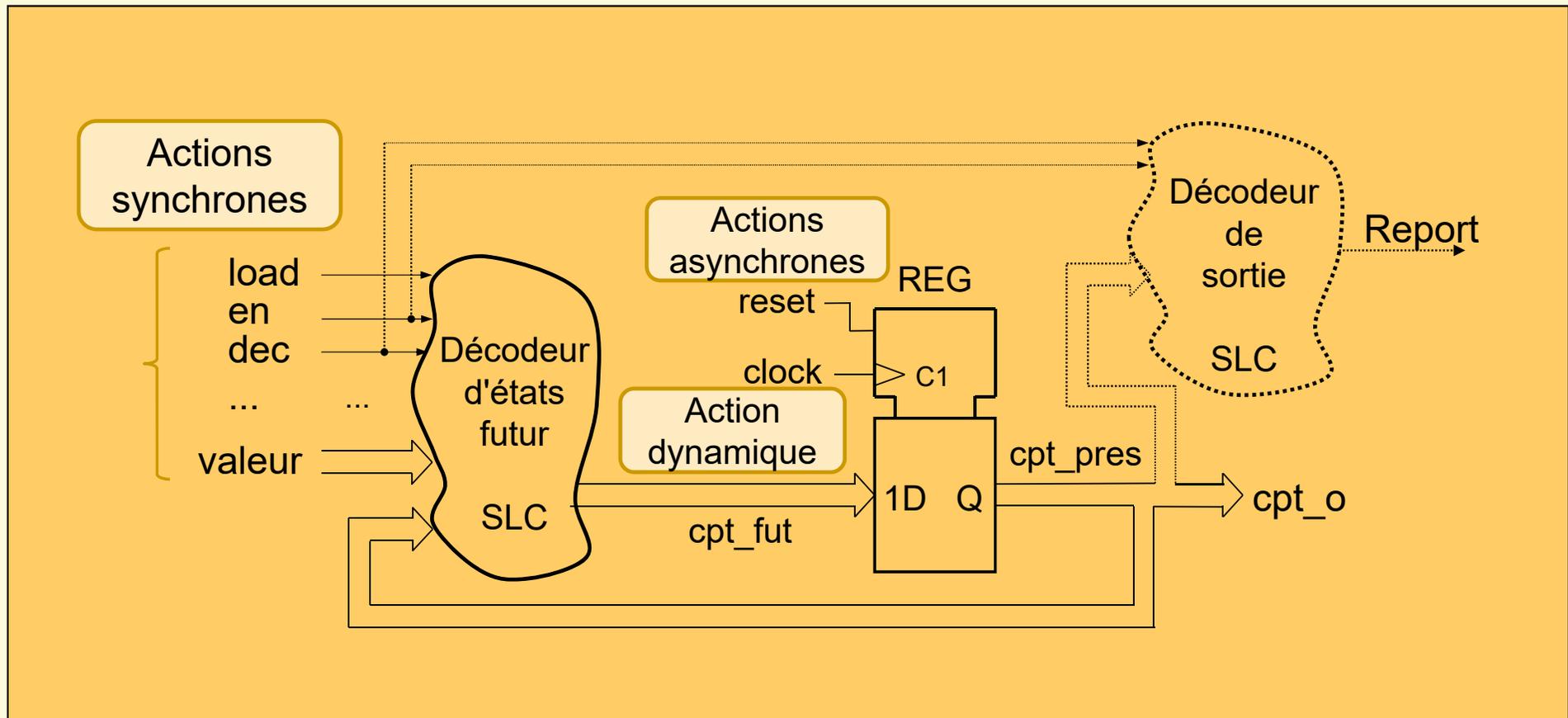
Calcul de F_{max} :



Si $F_{clock} \leq F_{max}$ alors le fonctionnement est garanti !

- Un système séquentiel comprend **3 types** d'entrées :
 - Les entrées à **action synchrone** :
 - action réalisée au flanc d'horloge sur les sorties
 - ces signaux sont connectés sur le décodeur d'actions futurs
 - Les entrées à action **asynchrone** :
 - action immédiate sur le registre (reset, set)
⇒ action immédiate sur les sorties
 - signaux **directement** connectés sur le registre
 - Une entrée à action **dynamique** :
 - entrée nommée clock (horloge)
 - définit l'instant où l'état interne change ⇒ **actions synchrones**
 - entrée connectée sur l'entrée d'horloge (sensible au flanc) du registre

Schéma bloc : compteur synchrone



Compteurs synchrones : caractéristiques

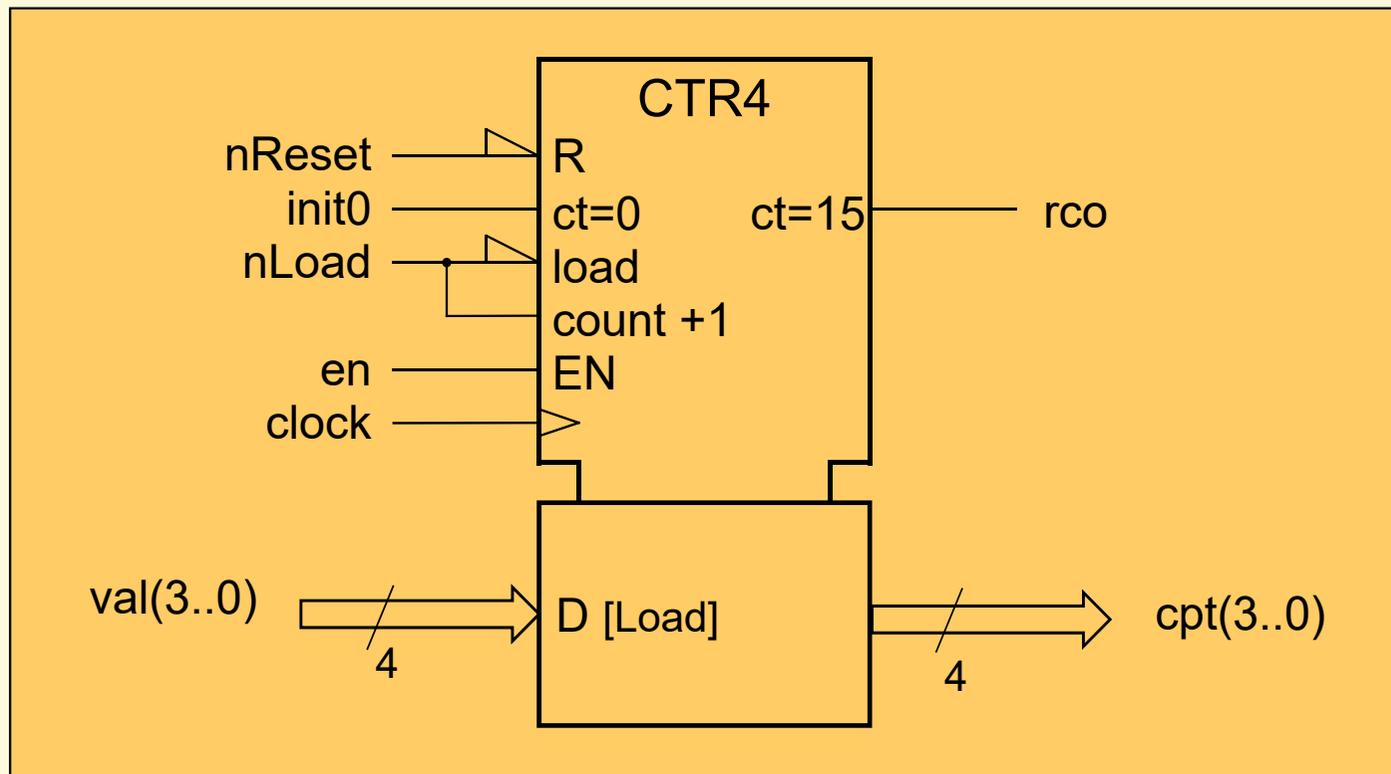
- Nombre de bits
- Nombre d'états : "modulo"
- Séquence des codes
- Action asynchrone: reset (evtl set)
- Modes de fonctionnement synchrone
 - init à une valeur fixe (constante: 0, max, autres)
 - load (chargement //)
 - up, down avec pas de 1 ou autres (2, 3, ..)

- Sont modulaires si :
 - ils disposent d'entrées
 - permission de compter / décompter
 - et de sorties
 - report /emprunt
- prévues pour un montage en cascade

Compteurs synchrones : symboles

- CTR_m compteur binaire m bits, 2^m états
- CTRDIV_m compteur modulo m (m états)
- mode sélection du mode de fonctionnement
(chargement, initialisation, comptage, maintien)
- enable autorise les fonctions de comptage (up ou down)
- clock entrée d'horloge (action dynamique)
- actions asynchr. commandes asynchrones sur le compteur
typiquement: reset (evtl set)

Compteur 4 bits, symbole ^{1/4}

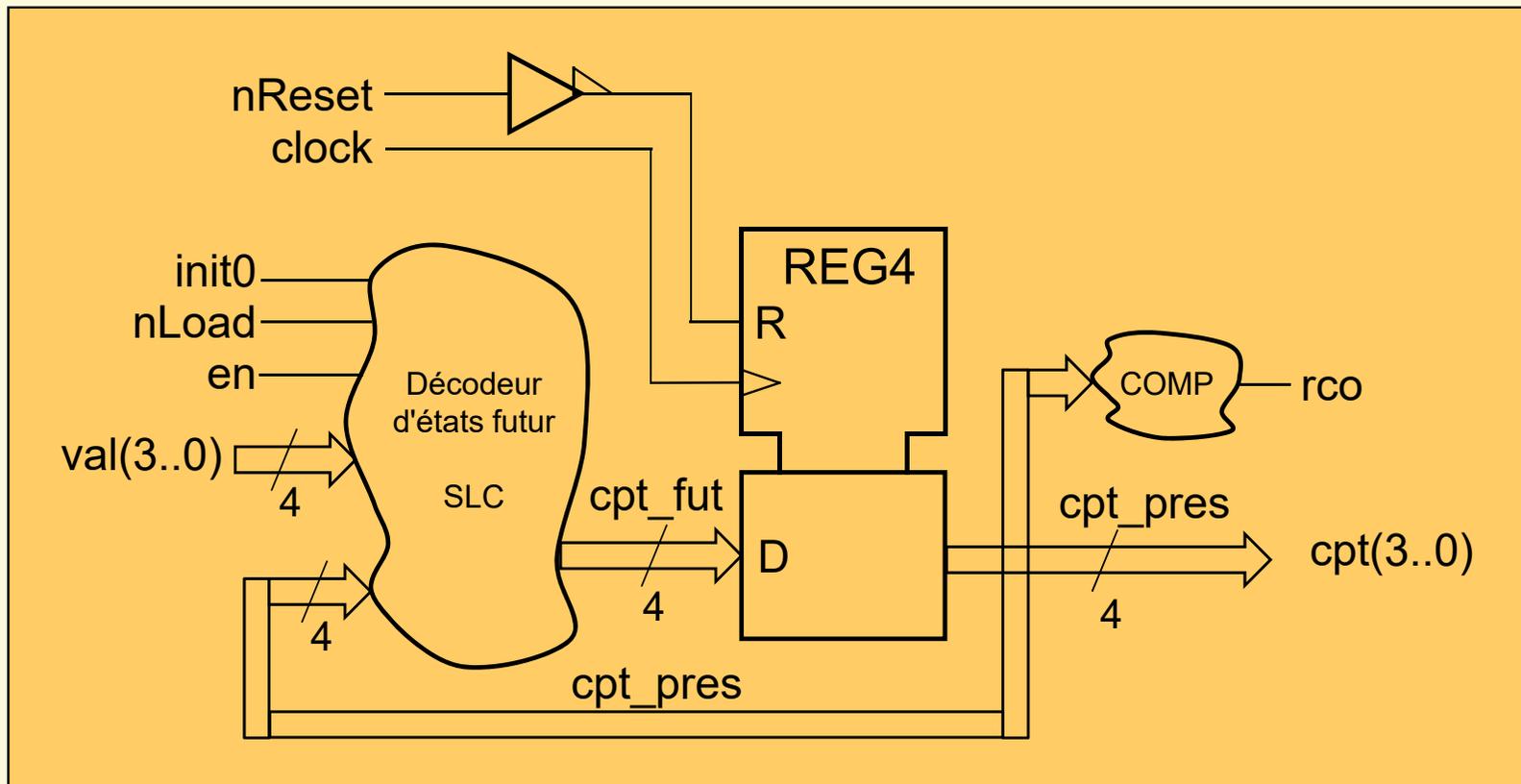


Analyse du fonctionnement:

- Action asynchrone:
 - Reset asynchrone (nReset)
- Actions synchrones
 - Initialisation à zéro indiqué par "ct=0" (init0)
 - Chargement parallèle (//) d'une valeur (nLoad)
 - Comptage par incrément de +1 (en avec nLoad inactif)
 - Maintien lorsqu'aucune commande active !
- important de définir l'ordre de priorité (pas visible sur symbole):
 - initialisation à zéro, chargement //, comptage, maintien

Compteur 4 bits, schéma 3/4

- Schéma bloc du compteur 4 bits (décomposition) :



Compteur 4 bits: table des fonctions synchr.

- Table fonctions synchrones:
 - Ordre priorité:
initialisation à zéro, chargement //, comptage, maintien

| init0 | load | en | cpt_fut | Fonction |
|-------|------|----|----------------|--------------------|
| 1 | - | - | = 0 | Initialisation à 0 |
| 0 | 1 | - | = val | Chargement // |
| 0 | 0 | 1 | = cpt_pres + 1 | Comptage |
| 0 | 0 | 0 | = cpt_pres | Maintien |

Conception de compteurs : Méthodologie

1. Identifier les fonctions du compteur
2. Lister ces fonctions (classer synchrones et asynchrones)
3. Définir les fonctions asynchrones (prioritaires)
4. Fixer les priorités des fonctions synchrones
5. Etablir la table des transitions synchrones

Exemple de table

| ctrl1 | ctrl2 | ... | cpt_pres | cpt_fut | Fonctions |
|------------|-------|-----|----------|----------------|--------------------|
| 1 | - | ... | - | = val | Chargement |
| 0 | 1 | ... | - | = cpt_pres + 1 | Comptage |
| 0 | 1 | ... | = 10 | = ... | Une autre fonction |
| ... | ... | ... | ... | | ... |
| autres cas | | | | = reg_pres | Maintien |

6. Etablir le schéma du compteur (conseil: utiliser cascades de mux2a1)
7. Réaliser la description VHDL du compteur

Exercices série III

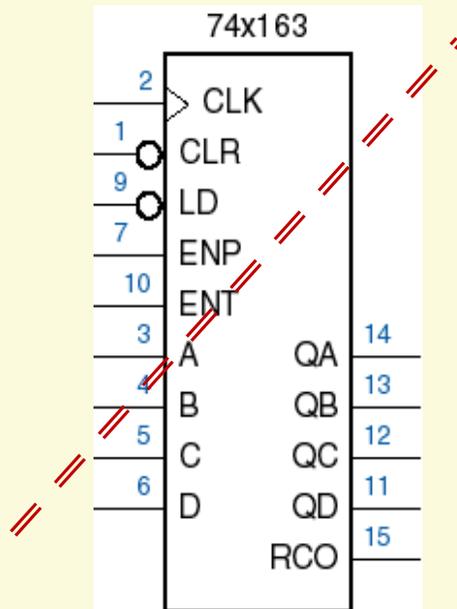
- 1) Réalisez un compteur modulo 12 à l'aide d'un PLD selon la décomposition d'un système séquentiel. Le compteur dispose des fonctions suivantes:
- chargement parallèle de val lorsque l'entrée *load* est active
 - comptage (+1) lorsque l'entrée *en* est active
 - sinon maintien

Concevoir celui-ci selon la décomposition d'un système séquentiel :

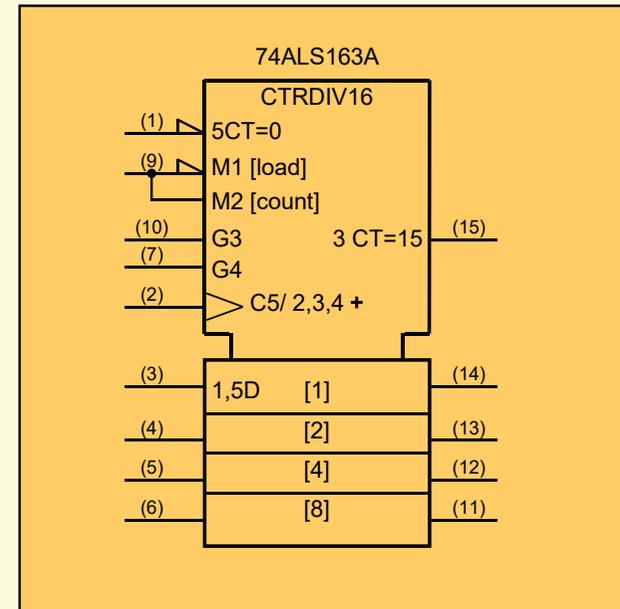
- Etablir la table des fonctions synchrones
- Donner le schéma bloc de la décomposition
- Décrire le circuit en VHDL synthétisable (CSN – SysLog2)

Compteurs standard 1/4

- Exemple de compteur très populaire : 74x163



Symbole MIL (**déconseillé**)



Symbole CEI (compréhensible!)

Lecture du symbole CEI

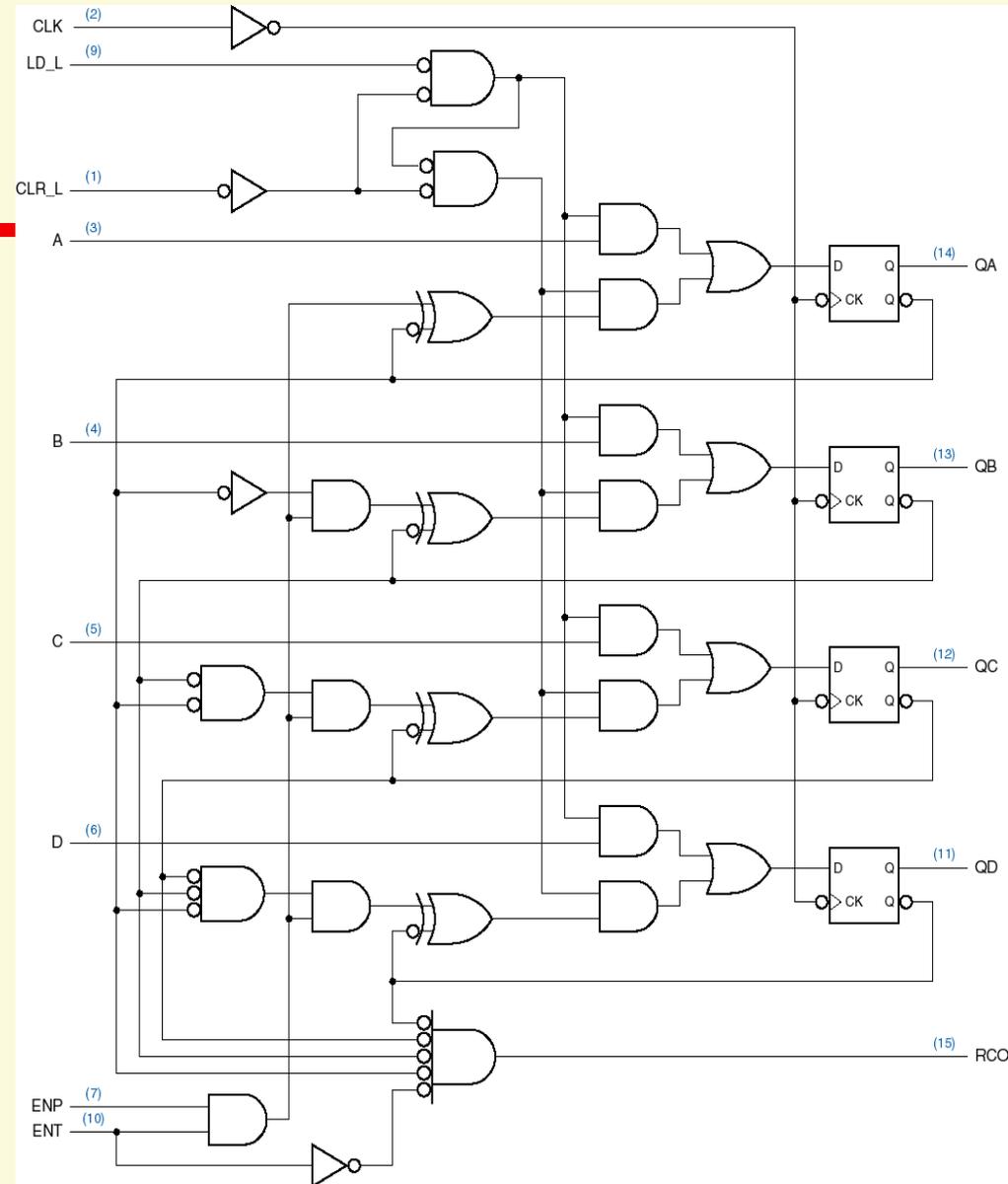
- CTRDIV16 \Rightarrow compteur modulo 16
- 5CT=0 \Rightarrow remise à 0 synchrone, car dépendant de la relation 5, qui est une relation C dynamique
- M1 \Rightarrow mode de fonctionnement 1, charge synchrone
- M2 \Rightarrow mode de comptage, mutuellement exclusif avec le chargement (une seule entrée physique)
- G3 \Rightarrow permet ce qui dépend de la relation 3 : le comptage et l'activation de la sortie de report

Compteurs standard ^{3/4}

- G4 \Rightarrow idem 4 : permet le comptage, est sans effet sur le report
- C5 \Rightarrow relation de commande, dynamique à cause du symbole $>$, entrée d'horloge des bascules
- / \Rightarrow cette entrée (ou sortie) a d'autres fonctions
- 2,3,4 + \Rightarrow comptage lorsque les relations 2, 3 et 4 sont satisfaites
- 1,5D \Rightarrow en mode 1, entrée D (synchrone car dépendant de la relation 5) de la bascule du module de poids 1

Compteurs standard 4/4

Schéma de principe d'un 74x163.



Exercices série III

2) D'après le symbole et le schéma du 74x163:

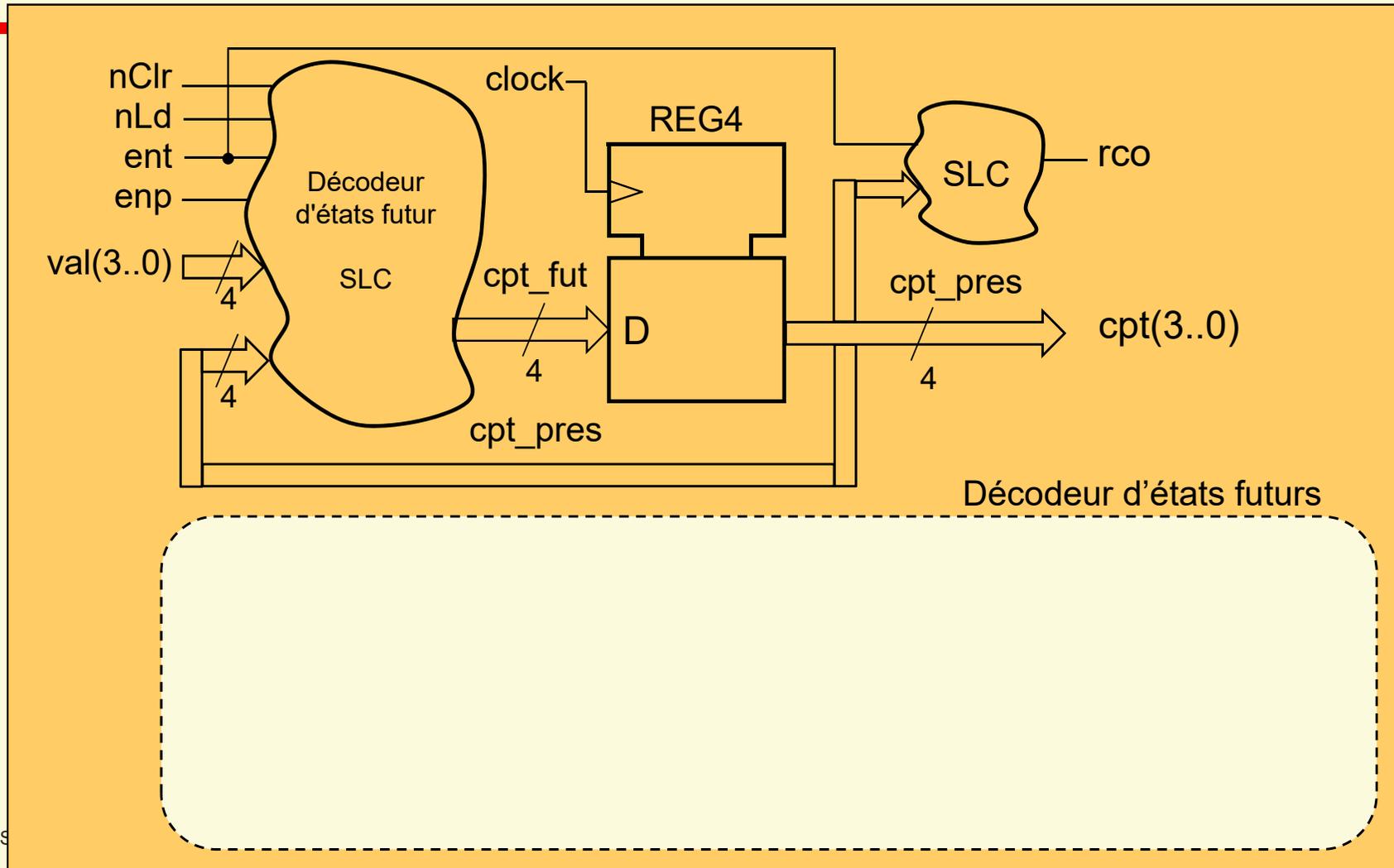
- listez les modes de fonctionnement de ce compteur
- quel mode de fonctionnement est le plus prioritaire?
- le décodeur d'état futur est-il décomposé en cascade?
- pourquoi avoir 2 entrées de permission de compter?
- faites une décomposition fonctionnelle du schéma
- décrivez textuellement le fonctionnement d'un module (ne pas confondre avec un bloc fonctionnel)

Conception du compteur 74x163

- Compteur 4 bits, modulo 16
- Aucune action asynchrone
- Table des fonctions synchrones :
 - Les fonctions sont données par ordre de priorité

| Clr | Ld | enp | ent | Cpt_fut | Fonction |
|-----|----|-----|-----|---------|----------|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

Schéma compteur 74x163



Exercices série III

3) Réalisez un compteur modulo 16 ayant les fonctions suivantes:

- Chargement parallèle (*load*)
 - Mode de comptage et décomptage (*Up_nDn*)
 - 2 entrées enable: *enp* et *ent*
 - actions comptage/décomptage exécutée si les 2 enable sont actifs
 - Sortie *rco* indiquant que le compteur est à max (CT=15)/min (CT=0) pour respectivement le mode comptage/décomptage (entrée *Up_nDn*). La sortie *rco* est activée si le enable *ent* est actif
- Concevoir celui-ci selon la décomposition d'un système séquentiel.
- Etablir la table des fonctions synchrones
- Donner le schéma bloc de la décomposition

Exercices série III

cpt_seq_stop.vhd

4) Réalisez un compteur qui réalise la séquence 2,3,...,7 lorsque l'entrée *long* est inactive, et la séquence 0,1,...,10 lorsque l'entrée *long* est active. De plus, ce compteur s'arrête à l'état 4 tant que l'entrée *stop_4* est active.

- Concevoir celui-ci selon la décomposition d'un système séquentiel
- Etablir la table des fonctions synchrones
- Donner le schéma bloc du compteur selon la décomposition d'un système séquentiel

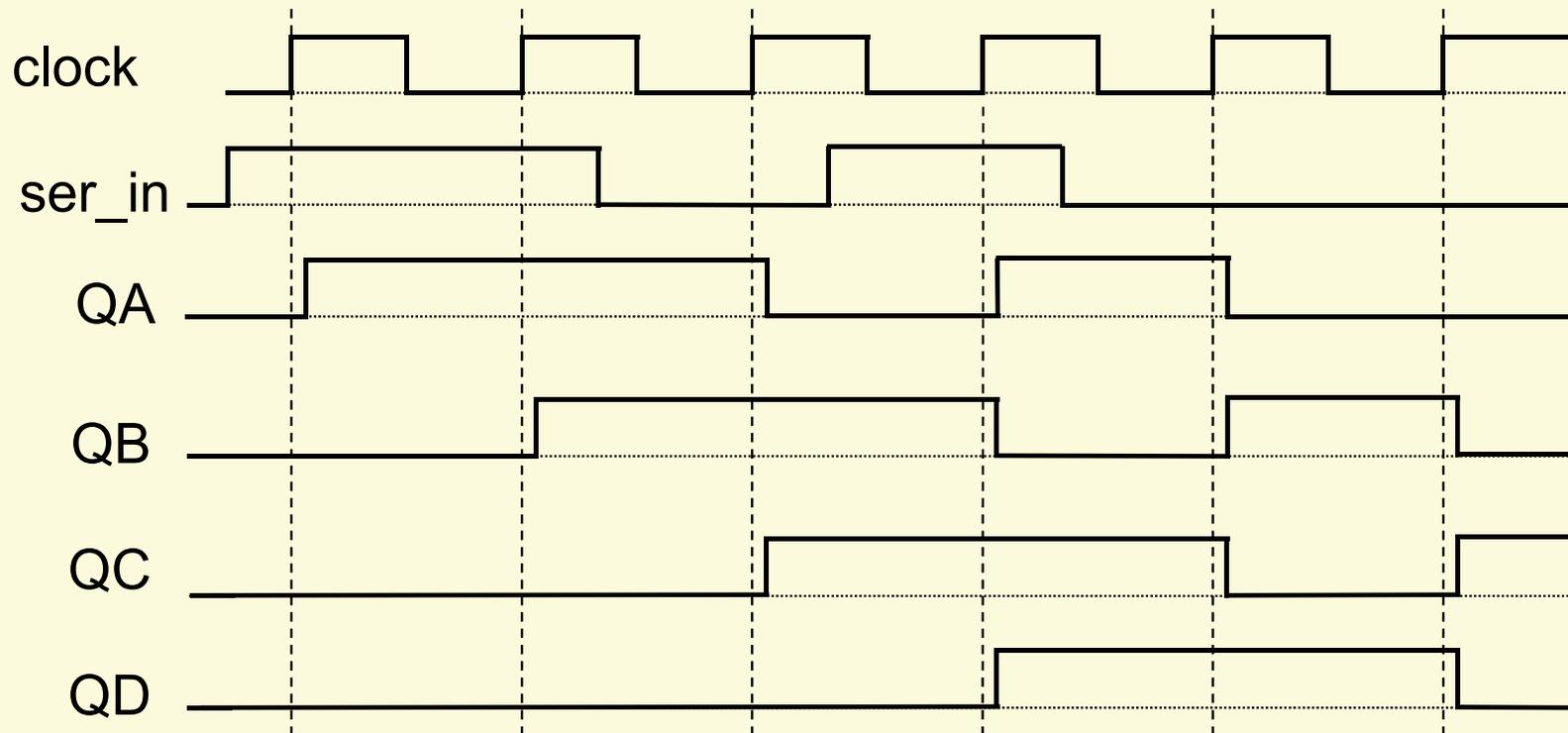
Fin de la présentation



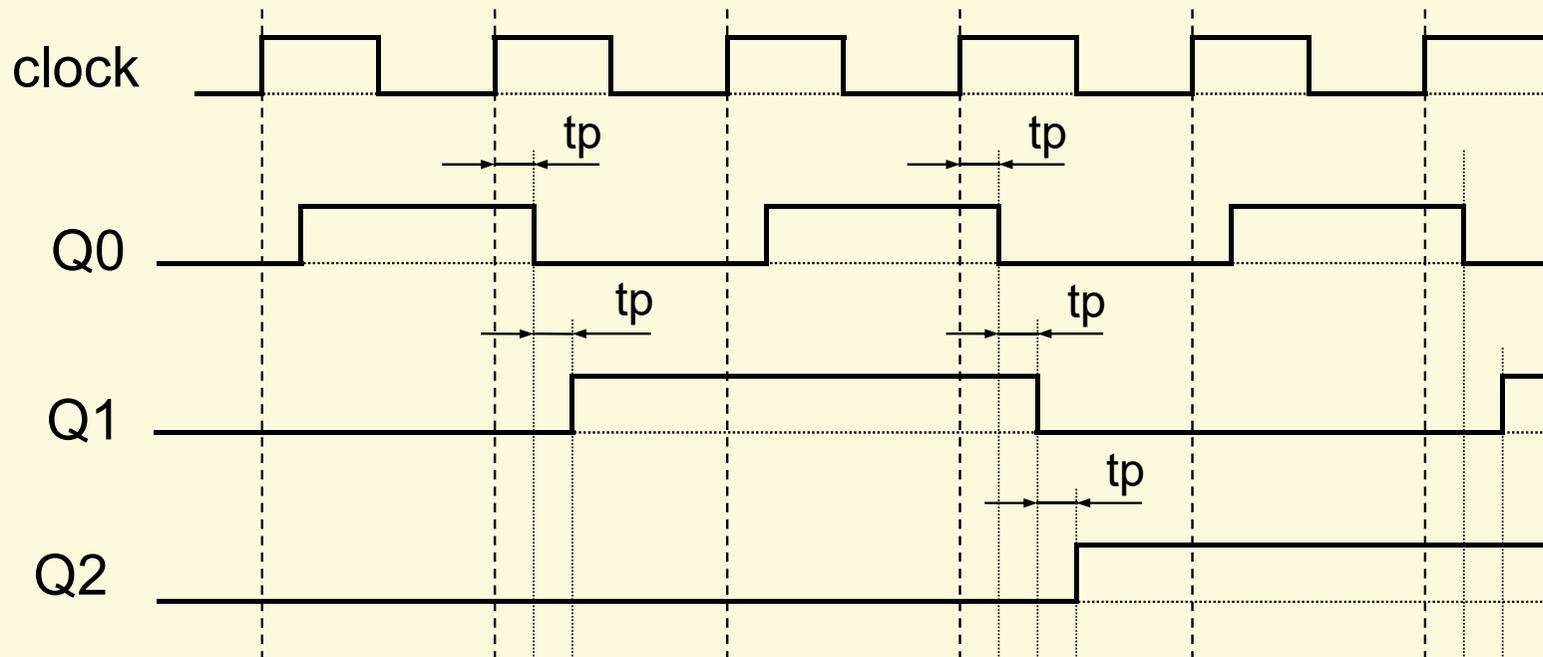
Chronogrammes et exemples complets

- Ci-après vous avez les dia avec le chronogramme et les exemples complétés tel que présentés.

Chronogramme registre à décalage



Chronogramme compteur asynchrone



Chronogramme compteur "full synchrone"

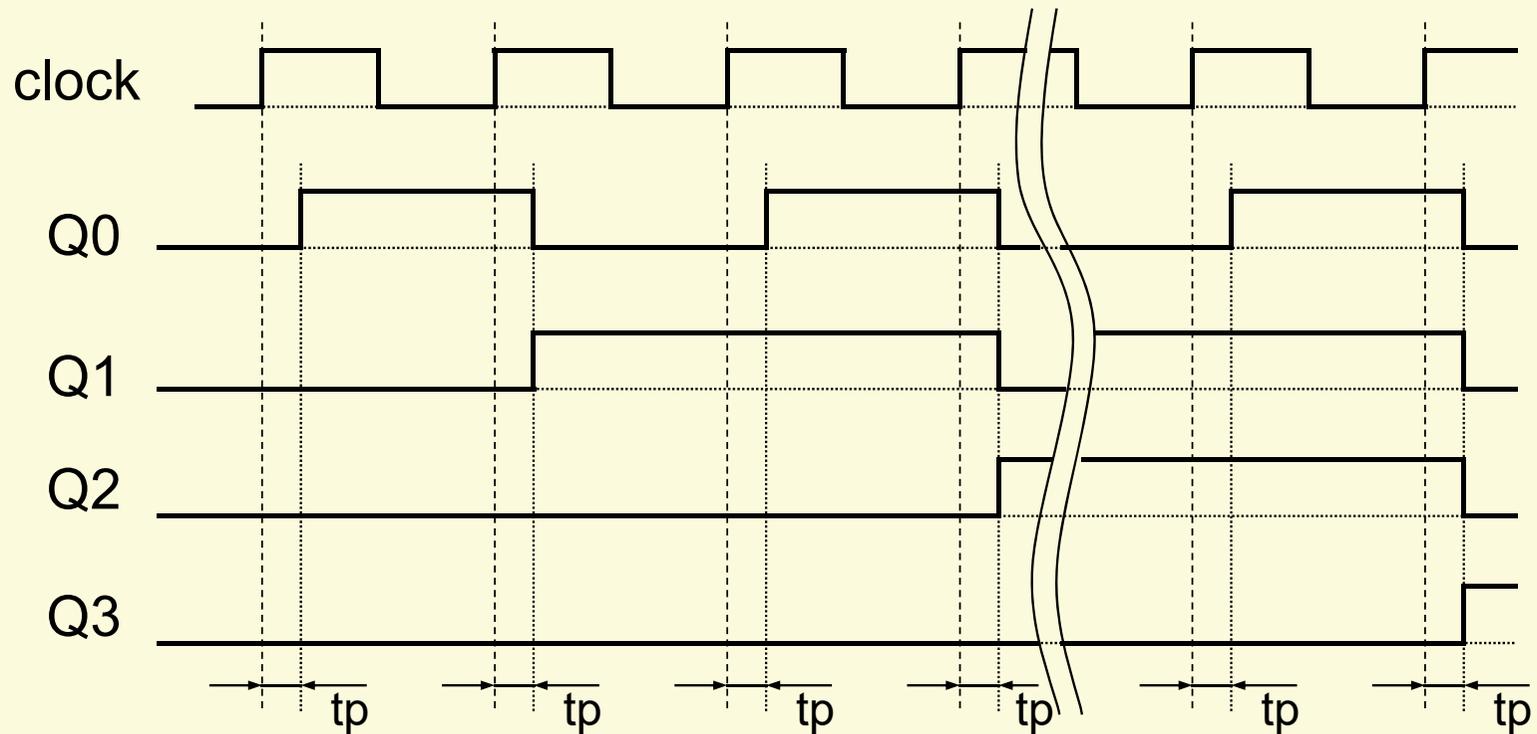


Schéma compteur 74x163

