heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

# Systèmes séquentiels
# Partie 2

Profs. Peña & Perez-Uribe & Mosqueron
Basé sur le cours du Prof. E. Sanchez

## 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

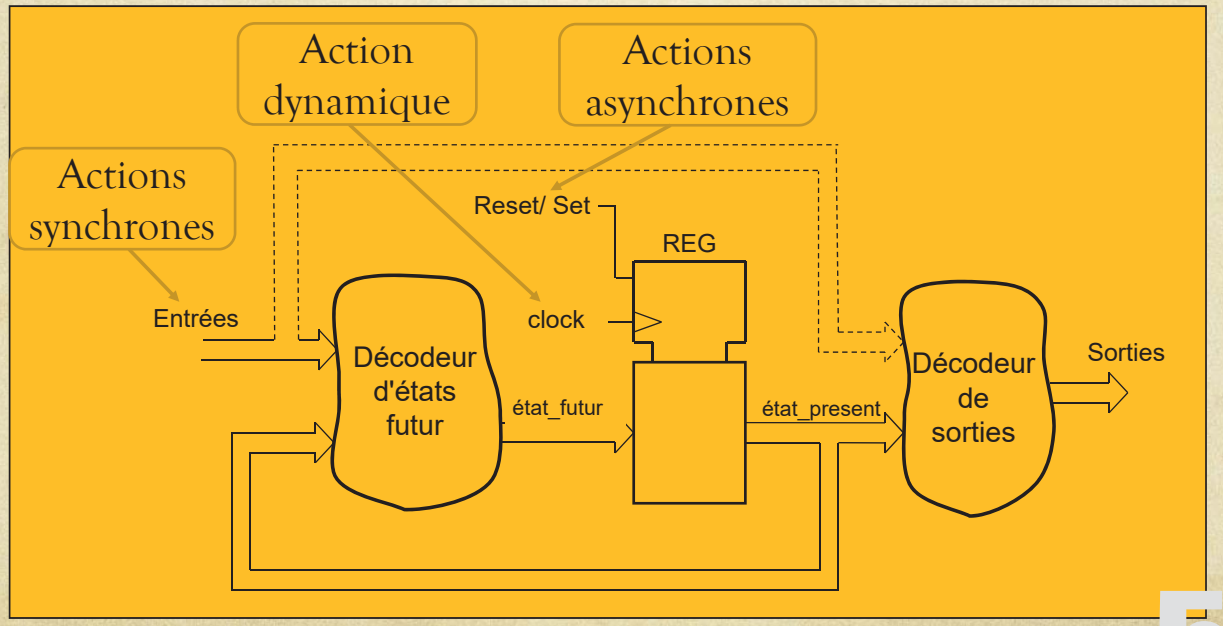
- Registres
  - parallèle - parallèle
  - série - parallèle
  - parallèle - série
  - mélange des fonctions série et parallèle
- Compteurs
  - Séquence : binaire, BCD, modulo N, ....
  - Mode : chargement parallèle, compte, décompte, ...



# Système séquentiel: types d'entrées

- 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

# Système séquentiel: schéma bloc






# 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
  - ....

# Le registre

- Un registre est un ensemble de bascules qui partagent le même signal d'horloge, ce qui permet le stockage simultané de plusieurs bits d'information
- L'existence d'un signal de contrôle LOAD (L) permet d'inhiber ou pas le chargement de l'information: à la montée du signal d'horloge, si L=1, les signaux d'entrée sont stockés dans le registre
- Sa table d'opérations est:

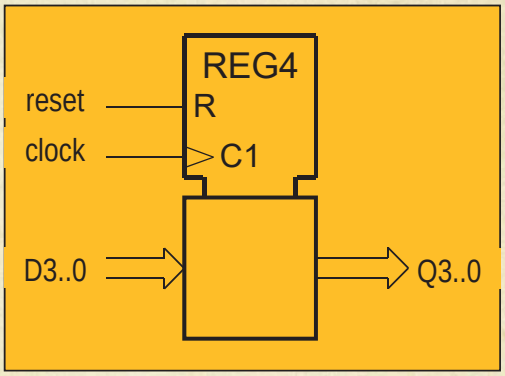
| Opération | description | L |
|---|---|---|
| HOLD | Q ← Q | 0 |
| LOAD | Q ← D | 1 |



# Registre simple

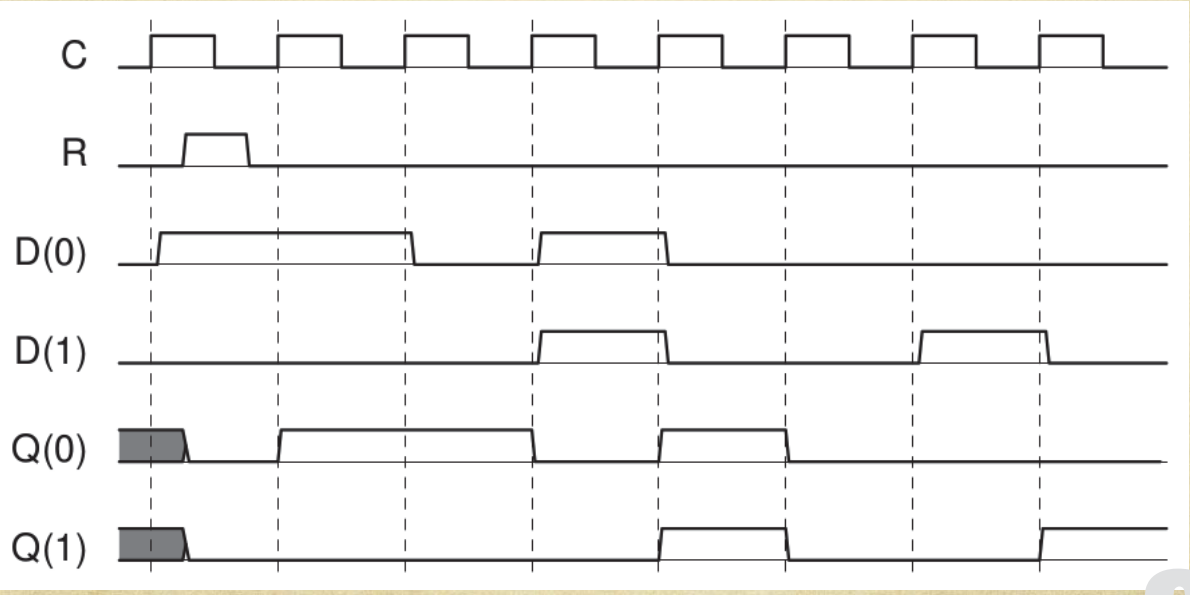
- Le registre simple est un ensemble de bascules en parallèle

Voici le symbole CEI:




- Il est à la base des systèmes séquentiels
- La conception des systèmes consistera principalement à établir quelle information doit être fournie sur l'entrée D de l'élément mémoire

# Registre simple: chronogramme

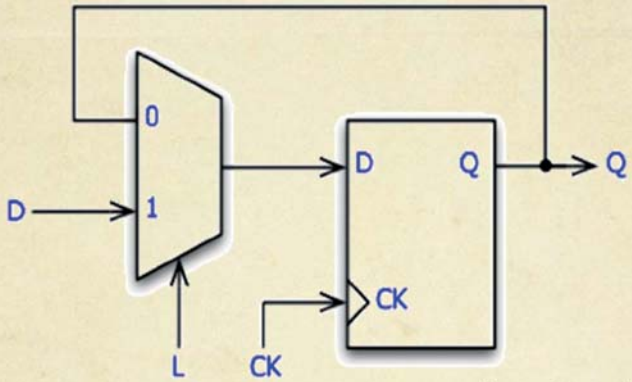



source: présentation reg/cpt Yann Thor

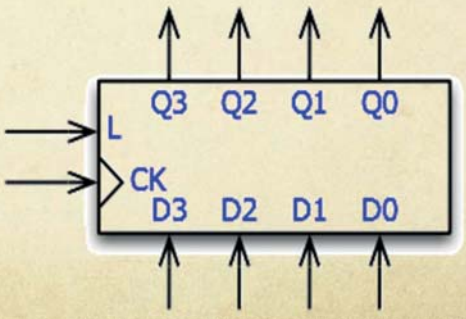


# Le registre

- Chaque bit d'un registre possède la structure suivante:




- Exemple: registre à 4 bits

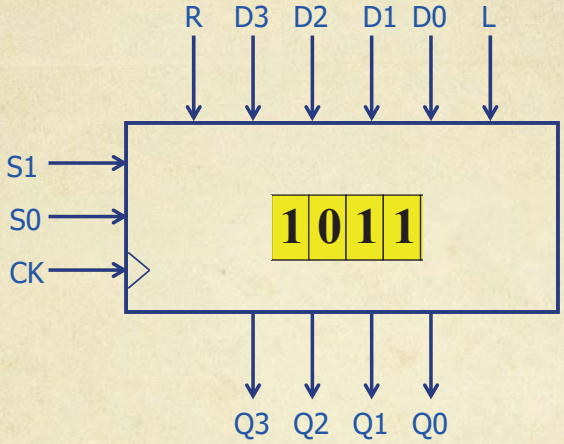


# Le registre à décalage (*shift register*)

- Un registre à décalage est un registre de taille fixe dans lequel les bits sont décalés à chaque coup d'horloge.
- Un registre à décalage est constitué d'un chaînage de bascules synchronisées sur l'horloge, la sortie d'une bascule étant reliée à l'entrée de la suivante.
- Les registres à décalage sont utilisés dans les liaisons série ; ils forment la base des UART et des modems. Imaginons que l'on veuille transmettre une information entre deux ordinateurs distants de quelques mètres. Transmettre l'information sous forme "parallèle" nécessiterait au moins 9 fils (8 pour les 8 bits, un pour la masse). Il est plus simple d'employer un registre à décalage pour envoyer les bits constituant chaque octet que l'on désire transmettre en une suite de 8 bits apparaissant l'un après l'autre sur une seule ligne.



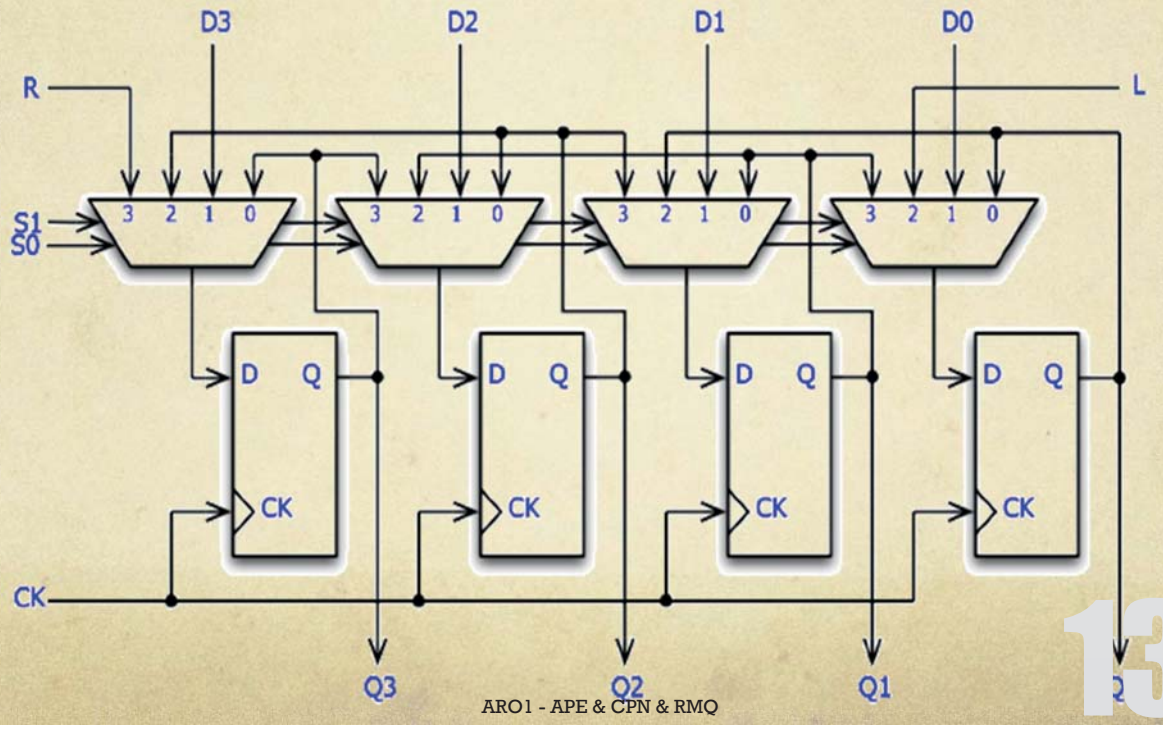
# Le registre à décalage (*shift register*)




| S1 | S0 | | |
|---|---|---|---|
| 0 | 0 | hold | $Q^+ = Q$ |
| 0 | 1 | load | $Q^+ = D$ |
| 1 | 0 | shift left | $Q^+ = <<Q$ |
| 1 | 1 | shift right | $Q^+ = >>Q$ |

# Le registre à décalage (*shift register*)

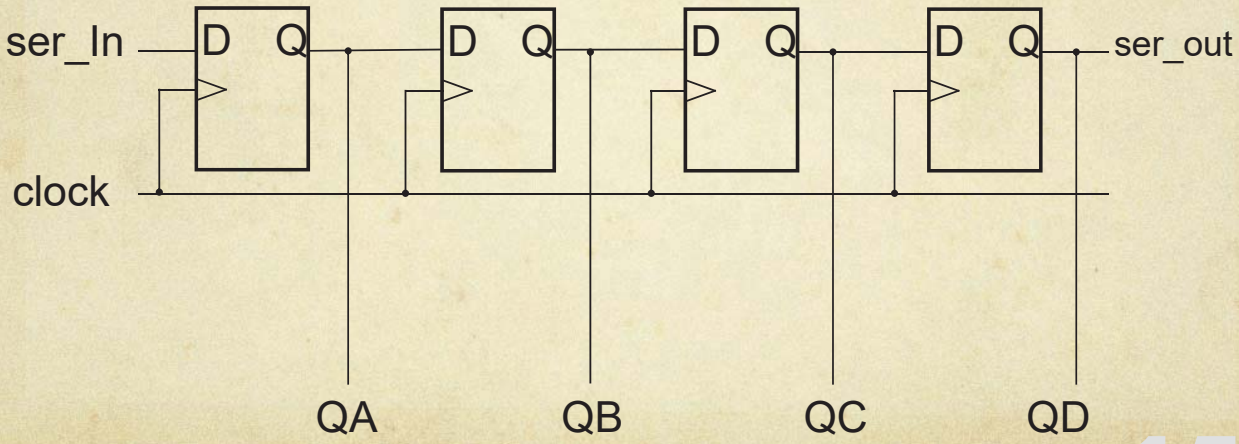
- Exemple d'un registre à décalage à 4 bits:



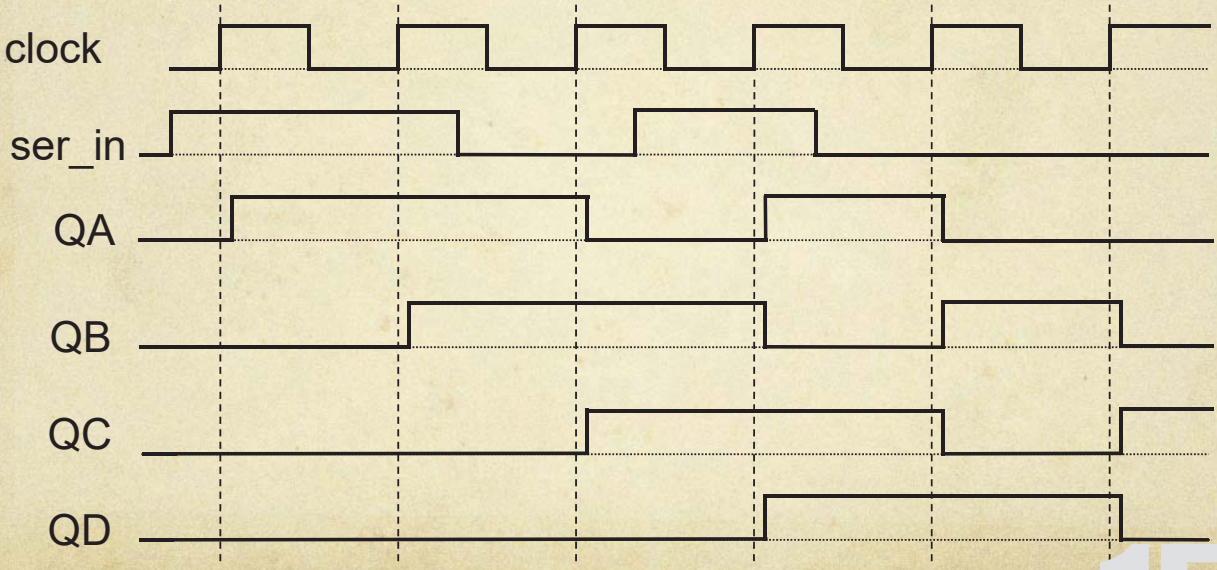



# Registre à décalage : principe

- 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 : contenu du registre décalé à droite
   - Si *en_shift* est inactif : état du registre maintenu

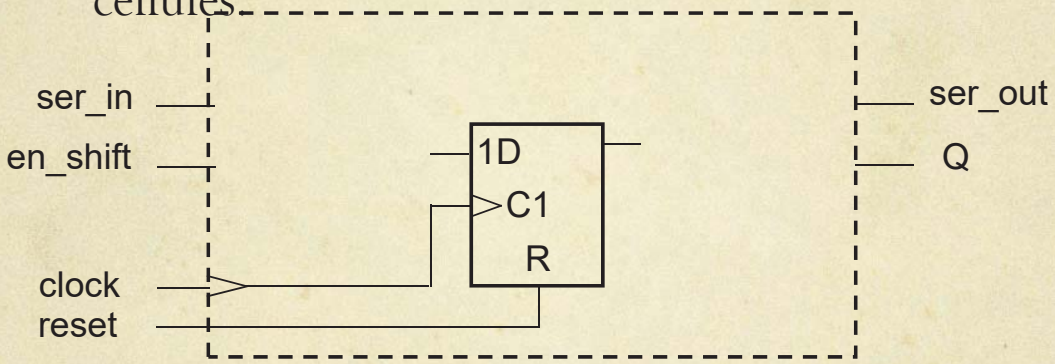
Commencez par concevoir une cellule (1 bit),

Ensuite réaliserez un registre 4 bits en cascadant 4 cellules

Finalement réalisez celui-ci avec un registre //-// de 4 bits et un bloc logique combinatoire (décodeur d'états futur)

# Exercice I.1

- Donner le schéma d'une cellule, puis chainez 4 cellules

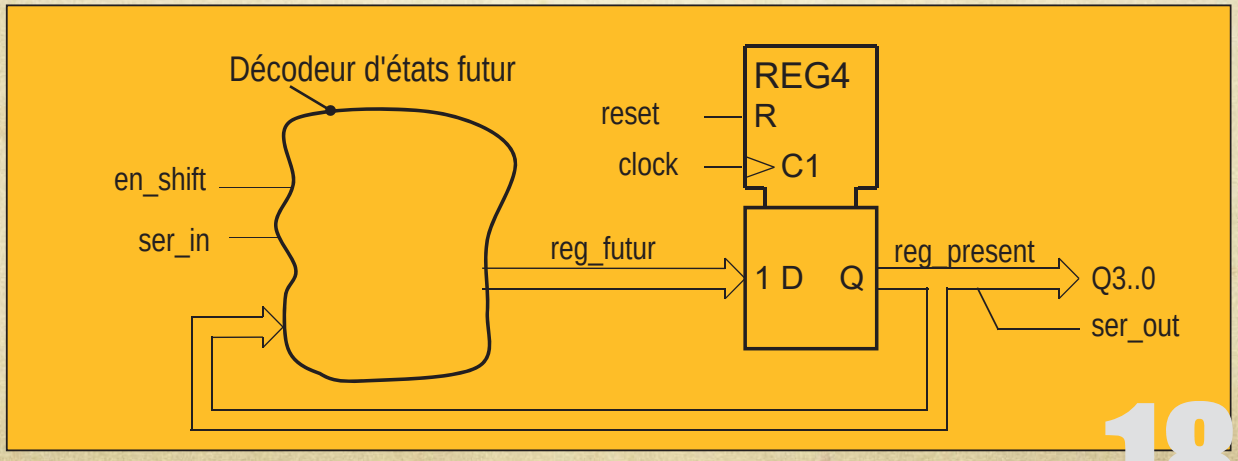





# Exercices série I

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

srg4_spl.vhd



# 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 |

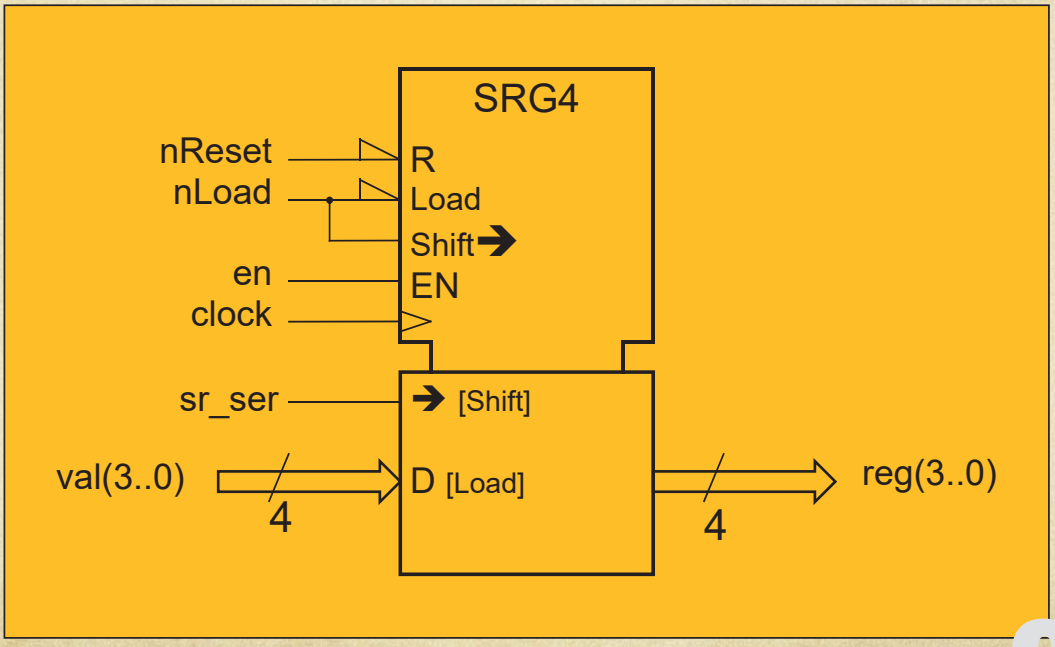
+ modes de fonctionnement combinés



# 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



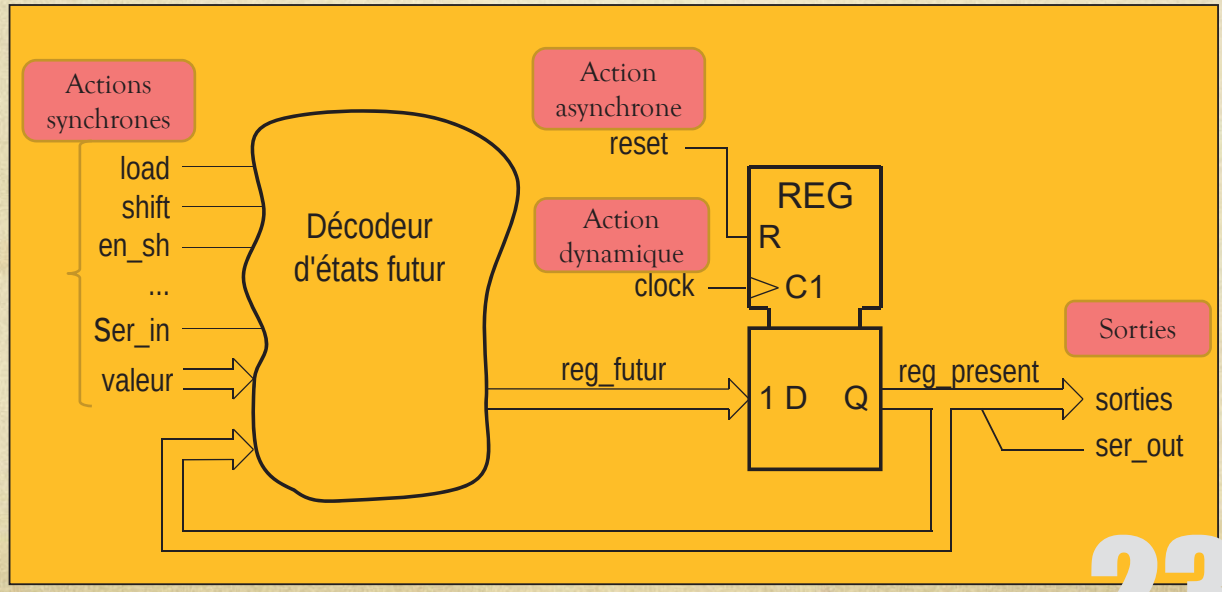



# Registre SRG4 : symbole

- 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 sr_ser
    sinon — maintien

# SRG4: table des fonctions

- Table de fonctions synchrones:

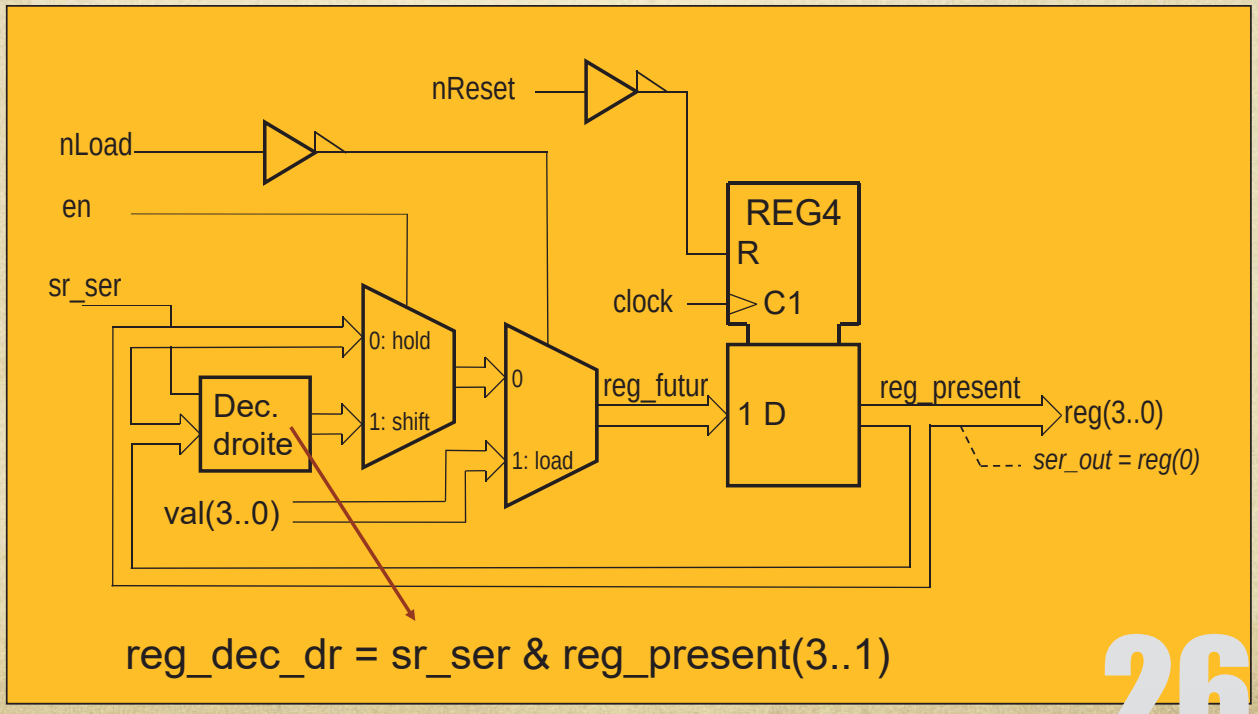
| load | en | sr_ser | Fonction |
|---|---|---|---|
| 1 | - | - | reg = val; chargement |
| 0 | 1 | - | reg = sr_ser & reg(3..1) décalage à droite |
| 0 | 0 | - | reg = reg; maintien |

nécessaire:

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



# SRG4: schéma bloc de la décomposition



# Conception de registres : Méthodologie

1. Identifier les fonctions du registre à décalalge
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

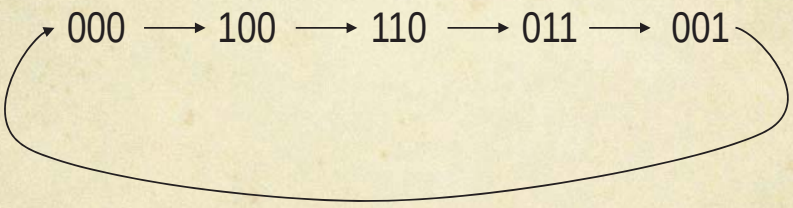
| Ctrl1 | Ctrl2 | ... | reg_pres | Fonction (reg_fut) |
|---|---|---|---|---|
| 1 | - | ... | - | Chargement |
| 0 | 1 | ... | - | Une certaine fonction |
| 0 | 1 | ... | = 10 | Une autre fonction |
| ... | ... | ... | ... | ... |
| autres cas | | | | Maintien |

6. Etablir le schéma du registre (penser aux Mux)
7. Réaliser la description VHDL du registre



# Exercices série II

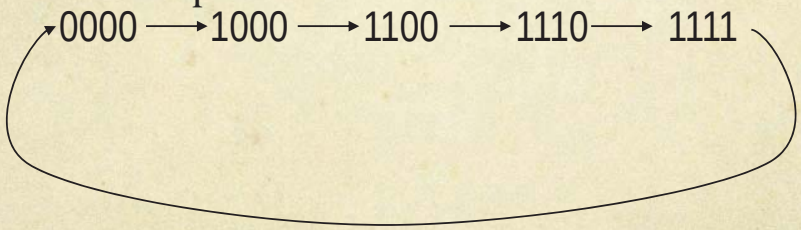
1. Concevoir un système séquentiel, en suivant la décom-position d'une machine séquentielle simple, afin de générer la séquence indiquée ci-dessous:



# Exercices série II

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

Voici la séquence à réaliser :





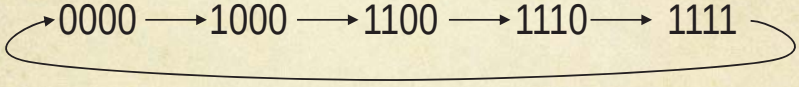

# Exercices série II

3. Concevoir un système séquentiel qui suit la décomposition d'une machine séquentielle simple qui répond au cahier des charges suivants :
   - Si *enable* est actif alors :

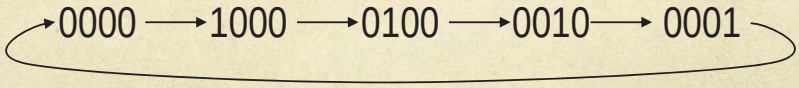
     Si *seq_plus* est actif alors

     srg_seq_en.vhd

     le système parcourt la séquence 1 suivante (plusieurs '1')




     sinon le système parcourt la séquence 2 suivante (un seul '1')

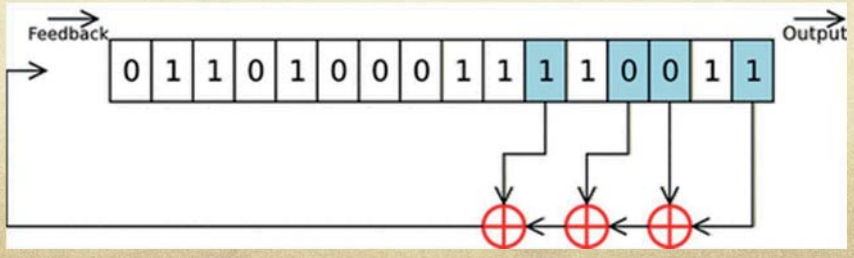



     sinon le système reste dans l'état actuel (maintien)

# Registre à décalage avec rétroaction linéaire

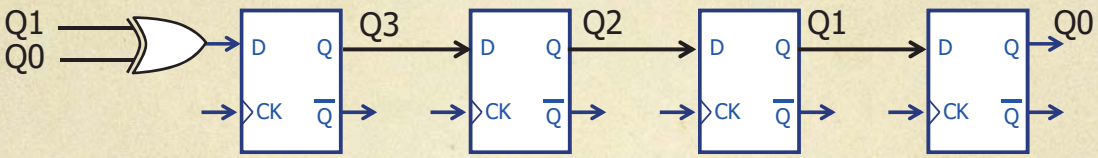
- Il s'agit d'une variante avec une unité logique ou arithmétique (*Linear Feedback Shift Register* ou *LFSR* en anglais). Le ou les bit(s) en sortie du registre subissent une série d'opérations pour être réinsérés dans le registre, en générant ainsi une séquence pseudo-aléatoire.
- Ce type de registre est utilisé en cryptographie pour les implantations matérielles de certains algorithmes de chiffrement de flot.
- Ce type de circuit est aussi utilisé lors de la phase de test des circuits intégrés en permettant la génération automatique d'entrées (vecteurs de tests).

⊕ ≡ XOR






# LFSR 4-bit




1111 -> 0111 -> 0011 -> 0001 -> 1000 ->0100 ->0010->

1001 ->1100 -> 0110 -> 1011 -> 0101 -> 1010 -> 1101->

1110 ->

En décimal: 15, 7, 3, 1, 8, 4, 2, 9, 12, 6, 11, 5, 10, 13, 14, …

# 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 pré-définie.
- L 'instant où a lieu le changement des sorties dépend d 'une entrée appelée horloge (ou clock)

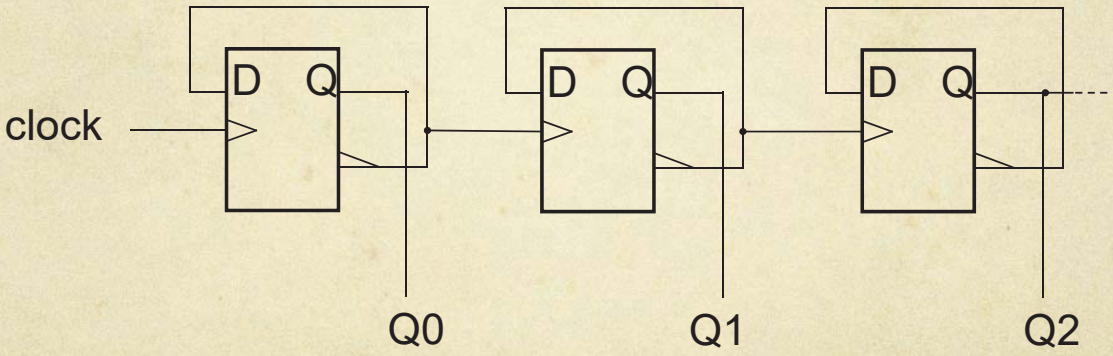


# Structure de compteur

- 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**

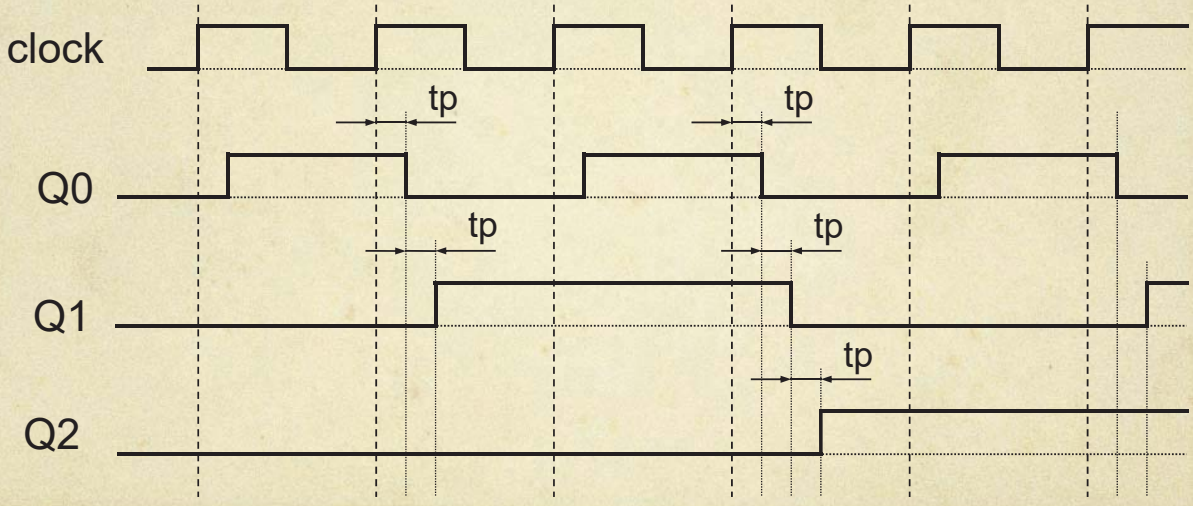
# Compteur asynchrone (ripple counter)

- 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 cellule N : Tn = N x tp
  - Très difficile à tester (vérification timing)
  - Possible sortie change après flanc suivant de l'horloge
- Très mauvaise intégration dans des PLDs
- Utilisation **NON RECOMMANDÉE** avec les circuits logiques programmables (CPLD et FPGA)

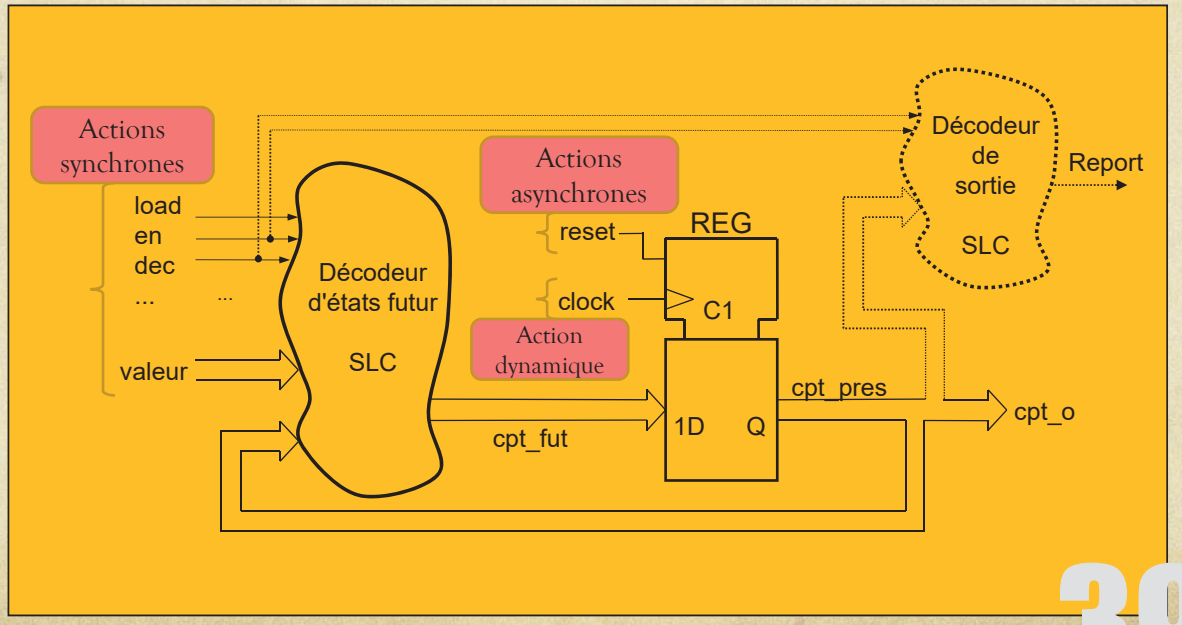


# Système séquentiel: types d'entrées

Rappel

- 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, ..)

# Compteurs standard 1/5

- Sont modulaires :

  ils disposent d'entrées
  - permission de compter / décompter

  et de sorties
  - report /emprunt

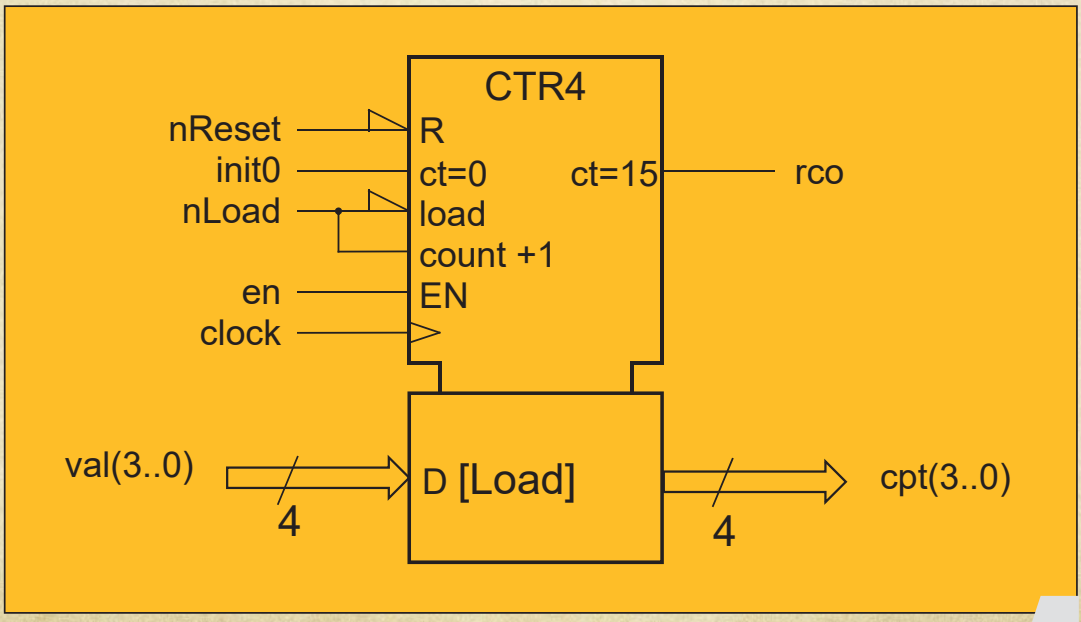
  prévues pour un montage en cascade



# Compteurs synchrones : symboles

- CTRm — compteur binaire m bits, $2^m$ états
- CTRDIVm — compteur modulo m (m états)
- mode fonctionnement — sélection du mode de (chargement, initialisation, comptage, maintien)
- enable (up ou down) — autorise les fonctions de comptage
- clock — entrée d'horloge (action dynamique)
- actions asynchr. compteur — commandes asynchrones sur le

# Compteur 4 bits, symbole 1/4






# Compteur 4 bits, fonctionnement

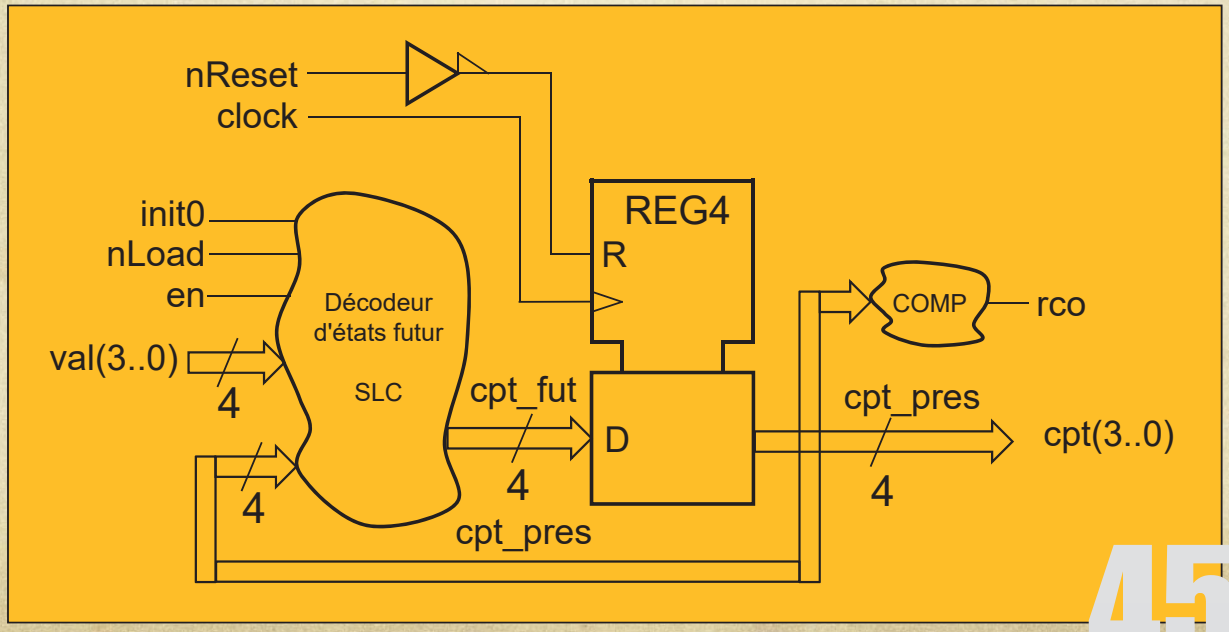
2/4

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:





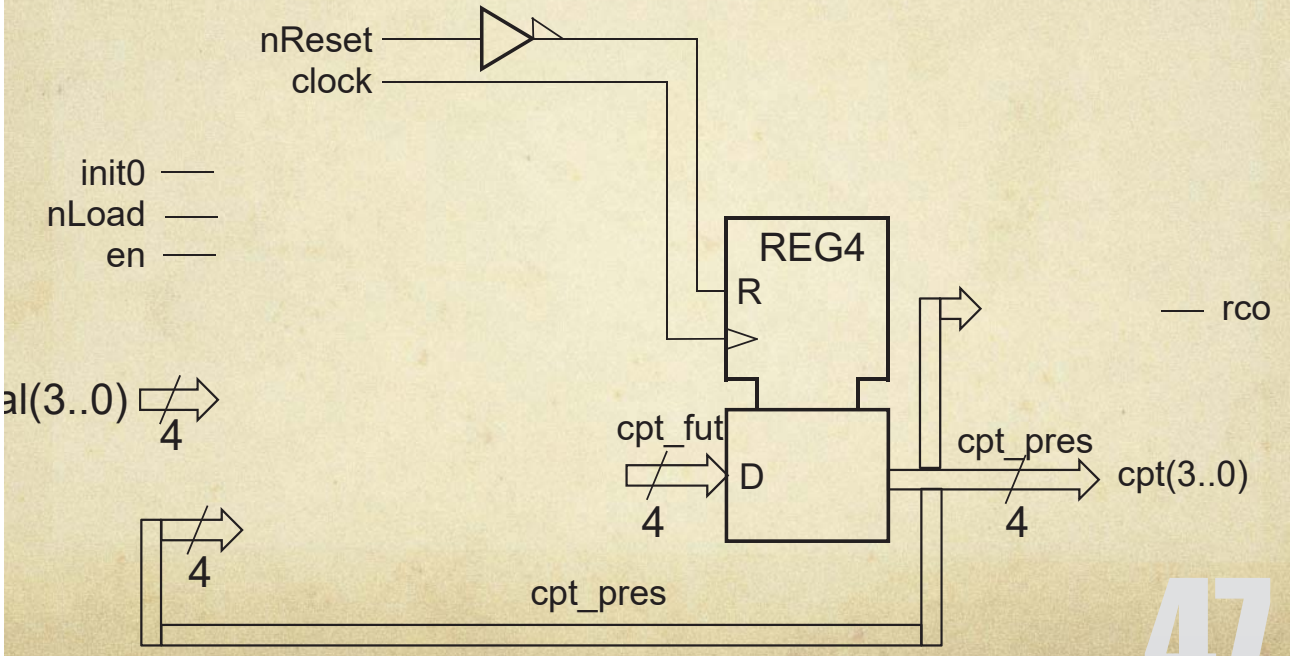

# Compteur 4 bits: table des fonctions synchr.

- 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 |

# Compteur 4 bits, schéma 4/4

- Compléter le schéma du compteur 4 bits:






# Conception de compteurs : Méthodologie

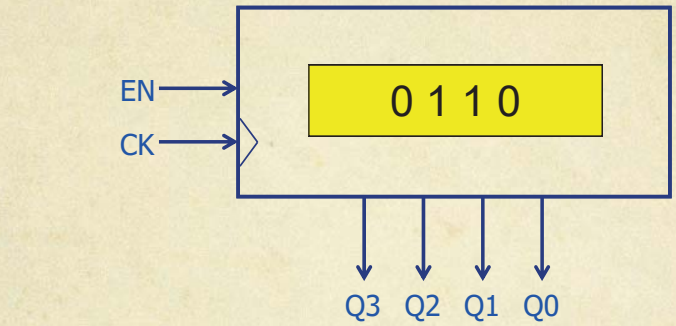
1. Identifier les fonctions du compteur
2. Lister ces fonctions (classer synchrones et asynchrones)
3. Définir les fonctions asynchrones (prioritaires)
4. Fixer les priorités des fonctions synchrones
5. Etablir la table des transitions synchrones

Exemple de table

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

6. Etablir le schéma du compteur (penser aux Mux)
7. Réaliser la description VHDL du compteur

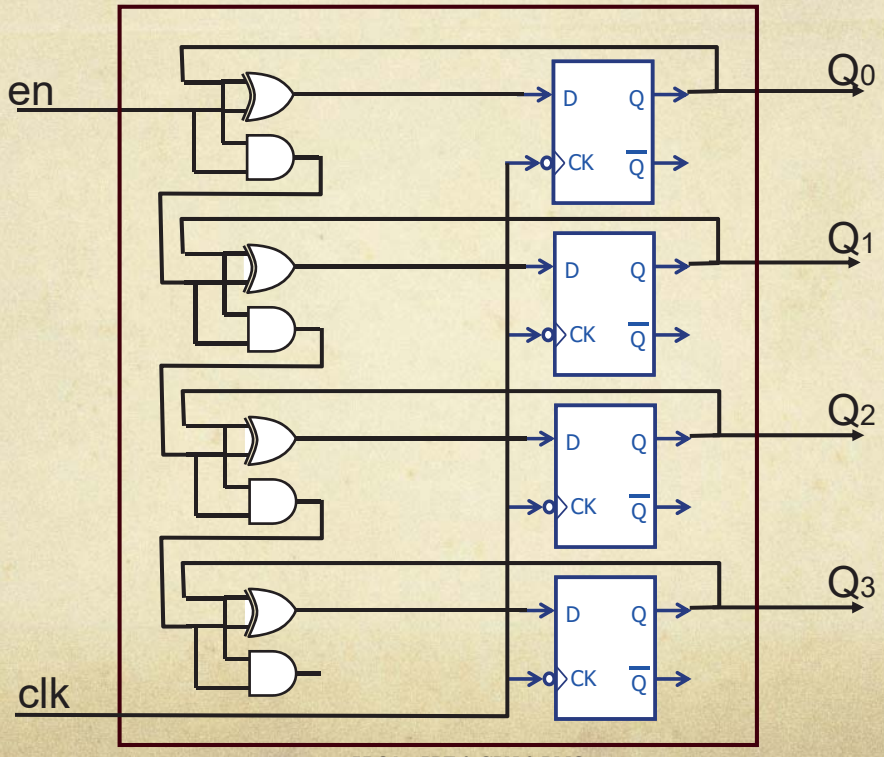
# Compteur



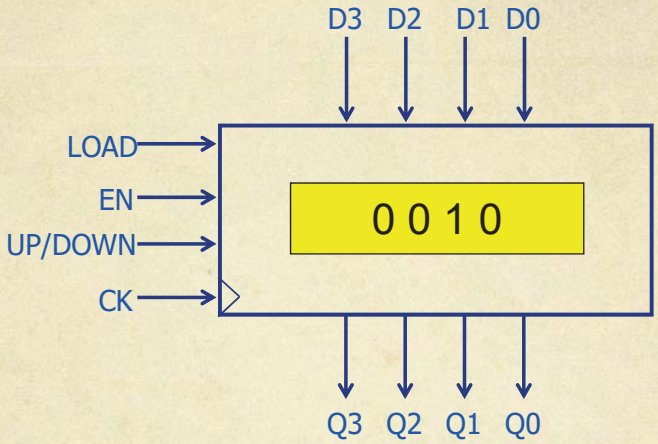

| EN | | |
|---|---|---|
| 0 | hold | $Q^+ = Q$ |
| 1 | count | $Q^+ = Q + 1$ |



# Compteur: structure



# Compteur




| LOAD | EN | Up/Down | | |
|---|---|---|---|---|
| 0 | 0 | - | hold | $Q^+ = Q$ |
| 0 | 1 | 0 | increase | $Q^+ = Q + 1$ |
| 0 | 1 | 1 | decrease | $Q^+ = Q - 1$ |
| 1 | - | - | load | $Q^+ = D$ |

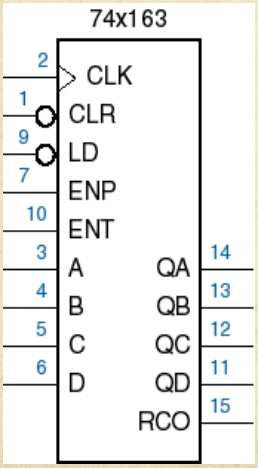


# Applications des compteurs

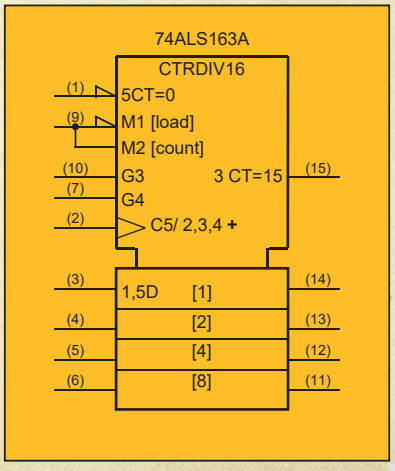
- Dans un processeur, le pointeur d'instruction (ou *program counter* en anglais) est un compteur qui contient l'adresse mémoire de la prochaine instruction à exécuter. Une fois l'instruction chargée, il est automatiquement incrémenté pour pointer l'instruction suivante.
- Un timer est un périphérique matériel permettant de mesurer des durées (généralement inclus dans les microcontrôleurs). Son rôle est de permettre la synchronisation des opérations que le microcontrôleur est chargé d'effectuer.

# Compteurs standard 1/4

- Exemple de compteur très populaire : 74x163




Symbole MIL (déconseillé)




Symbole CEI (compréhensible!



# Compteurs standard 2/4

Lecture du symbole CEI

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

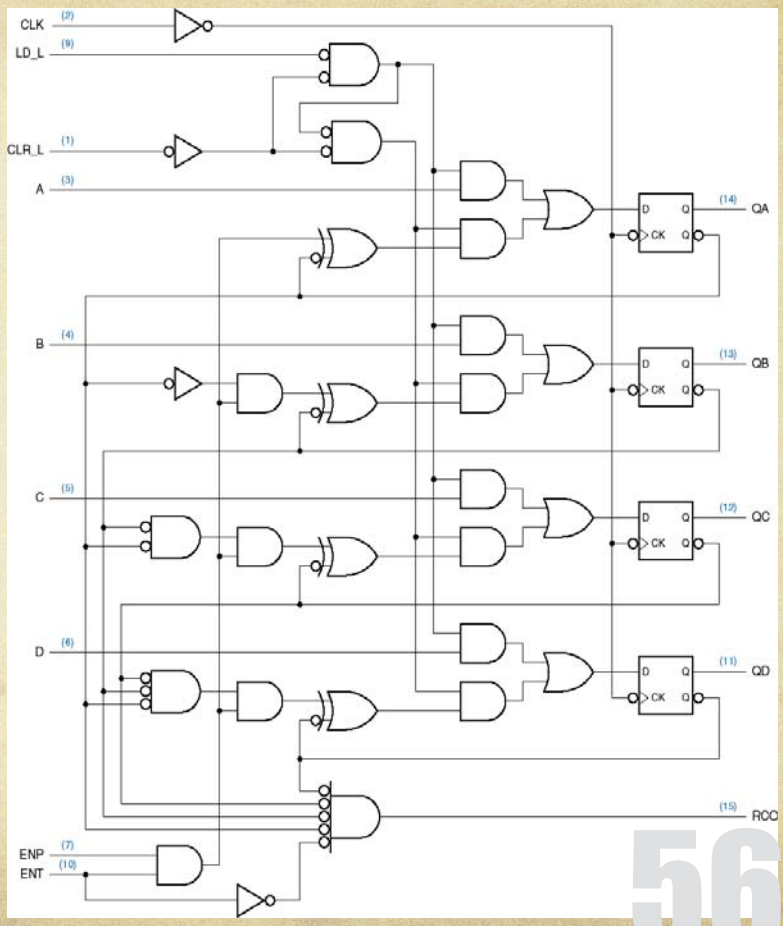
# Compteurs standard 3/4

- G4 ⇒ idem 4 : permet le comptage, est sans effet sur le report
- C5 ⇒ relation de commande, dynamique à cause du symbole > , entrée d'horloge des bascules
- / ⇒ cette entrée (ou sortie) a d'autres fonctions
- 2,3,4 + ⇒ comptage lorsque les relations 2, 3 et 4 sont satisfaites
- 1,5D ⇒ 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.

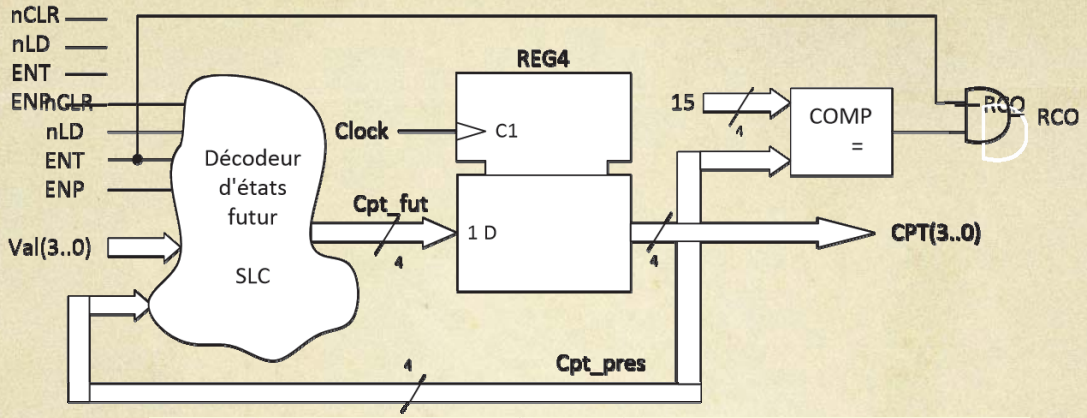


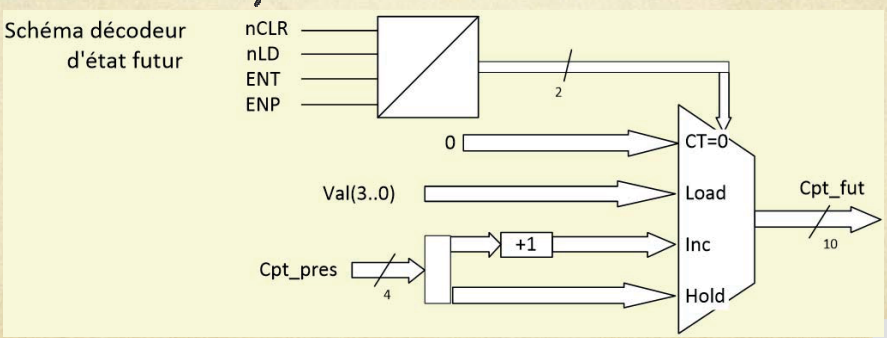
# Exercices série III

1) 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)



# Schéma compteur 74x163



# Exercices série III

2) Réalisez un compteur modulo 12 à l'aide d'un PLD selon la décomposition d'un système séquentiel. Le compteur dispose des fonctions suivantes:
   - load: chargement parallèle de val
   - en: active le comptage

Donner les spécifications du compteur

Liste des fonctions, table, ...

Donner le schéma bloc de la décomposition

*Décrire le circuit en VHDL synthétisable*



# Modification du modulo 1/2

- Par détection (décodage) de l'état où la rupture de séquence doit avoir lieu, et action **sur les entrées synchrones uniquement** :
  - insensibles aux transitoires
  - action au prochain coup d'horloge → ne provoquent pas la disparition immédiate de l'état décodé
  - n'ajoutent pas de décalage

# Exercices série III

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.

- Spécifiez la liste des fonctions et établir table des fonctions synchrones
- Donnez le schéma bloc du compteur selon la décomposition d'un système séquentiel