Architecture des systèmes numériques et informatiques

Cours 5 - Circuits séquentiels

Halim Djerroud

LISV
Laboratoire d'ingénierie des systèmes de Versailles

UVSQ
université PARIS-SACLAY

révision : 2.0

Rappel : Circuits combinatoires

Caractéristiques des circuits combinatoires

- Les sorties dépendent **uniquement** des entrées actuelles
- Pas de mémoire, pas d'état interne
- Fonction : $S(t) = f(E(t))$

Exemples

Additionneur, comparateur, multiplexeur, décodeur, encodeur

Limitation

Impossible de mémoriser une information ou de créer des compteurs !

Introduction aux circuits séquentiels

Définition

Un circuit séquentiel est un circuit logique dont les sorties dépendent :

- Des entrées actuelles
- De l'**état interne** (historique des entrées)

Fonction générale

$$S(t) = f(E(t), \text{État}(t-1))$$

$$\text{État}(t) = g(E(t), \text{État}(t-1))$$

Avantage fondamental

Permet de **mémoriser des informations** et de créer des systèmes avec états

Différences circuits combinatoires vs séquentiels

Circuits combinatoires

- Sorties = f(Entrées)
- Pas de mémoire
- Réponse instantanée
- Pas de notion de temps
- Pas d'horloge nécessaire

Exemples

Portes logiques, additionneur, MUX

Circuits séquentiels

- Sorties = f(Entrées, État)
- Avec mémoire
- Dépend de l'historique
- Notion de temps/séquence
- Horloge (souvent)

Exemples

Bascules, registres, compteurs

Applications des circuits séquentiels

Domaines d'utilisation

- **Mémoires** : RAM, registres, caches
- **Compteurs** : Division de fréquence, horloges
- **Machines à états** : Contrôleurs, protocoles
- **Registres à décalage** : Communication série
- **Séquenceurs** : Unités de contrôle des processeurs

Importance

Les circuits séquentiels sont au cœur de tous les systèmes numériques modernes !

Plan du cours

1. **Les bascules** : éléments de base de la mémoire
   - Bascule RS (Reset-Set)
   - Bascule JK
   - Bascule D (Data/Delay)
   - Bascule T (Toggle)
2. **Les registres** : groupes de bascules
   - Registres série
   - Registres parallèles
   - Registres à décalage
3. **Les compteurs** : circuits de comptage
   - Compteurs asynchrones
   - Compteurs synchrones
   - Compteurs modulo-N

Qu'est-ce qu'une bascule ?

Définition

Une bascule (flip-flop) est un circuit électronique bistable capable de mémoriser **un bit** d'information (0 ou 1)

Caractéristiques principales

- Possède **deux états stables** : 0 ou 1
- Conserve son état même si les entrées changent
- Deux sorties complémentaires : Q et $\overline{Q}$ (ou Q')
- Si $Q = 1$ alors $\overline{Q} = 0$ et vice-versa

Élément de base

La bascule est l'élément de base de toute mémoire numérique

Principe de la mémorisation

Comment mémoriser un bit ?

Utiliser un **circuit bouclé** (rétroaction/feedback) :

- La sortie est réinjectée à l'entrée
- Crée une boucle qui maintient l'état
- Stable dans deux configurations : 0 ou 1

Analyse du circuit à boucle

Circuit simple avec 2 portes NOR

Les différents types de bascules

Quatre types principaux :

1. **Bascule RS** (Reset-Set)
   - Entrées : R (Reset/Remise à 0) et S (Set/Mise à 1)
   - La plus simple, mais possède un état interdit
2. **Bascule JK**
   - Entrées : J et K
   - Amélioration de RS sans état interdit
3. **Bascule D** (Data ou Delay)
   - Entrée : D (Data)
   - La plus simple à utiliser, recopie l'entrée
4. **Bascule T** (Toggle)
   - Entrée : T
   - Change d'état quand T=1

Bascules asynchrones vs synchrones

Asynchrones

- Changent d'état **immédiatement** quand les entrées changent
- Pas d'horloge
- Plus rapides
- Difficiles à synchroniser

Problème

Désynchronisation dans les systèmes complexes

Synchrones

- Changent d'état **uniquement** sur un signal d'horloge
- Entrée horloge H (ou CLK)
- Tous les changements coordonnés
- Standard dans les systèmes modernes

Avantage

Synchronisation garantie

Notation

Dans les tables de vérité : Q = état présent, Q^+ = état suivant

Bascule RS : Principe

Signification des entrées

- **R** (Reset) : Remise à zéro ($Q = 0$)
- **S** (Set) : Mise à un ($Q = 1$)

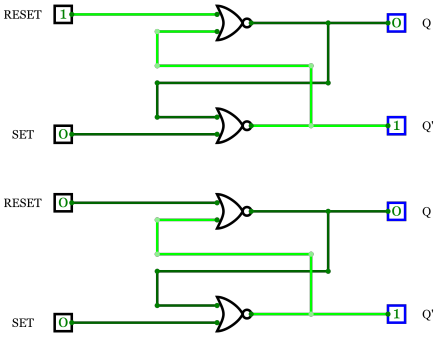
Comportements

- $S = 1, R = 0$: Force $Q = 1$ (Set)
- $S = 0, R = 1$: Force $Q = 0$ (Reset)
- $S = 0, R = 0$: Maintient l'état actuel (Mémoire)
- $S = 1, R = 1$: **État interdit** ! (conflit)

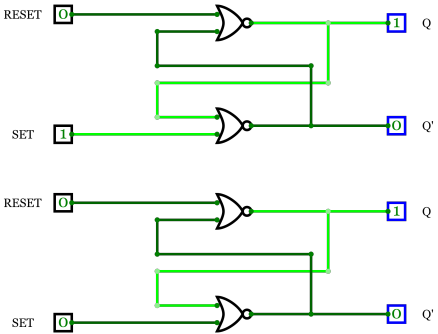
État interdit

Quand $S = R = 1$, les deux sorties tentent de passer à 0 simultanément, violant la complémentarité Q et $\overline{Q}$

Bascule RS : Reset (R=1, S=0)



Bascule RS : Set (R=0, S=1)



Bascule RS : Symbole bloc

Représentation symbolique

- Entrées : S (Set) et R (Reset)
- Sorties : Q et $\overline{Q}$
- Simplifie les schémas de circuits complexes

Table de vérité complète de la bascule RS

S	R	Q	Q^+	Observation
0	0	0	0	Mémoire
0	0	1	1	Mémoire
0	1	0	0	Reset
0	1	1	0	Reset
1	0	0	1	Set
1	0	1	1	Set
1	1	0	X	Interdit
1	1	1	X	Interdit

Analyse par cas :

- **S=0, R=0** : $Q^+ = Q$ (conservation)
- **S=0, R=1** : $Q^+ = 0$ (reset)
- **S=1, R=0** : $Q^+ = 1$ (set)
- **S=1, R=1** : Indéterminé (X)

Contrainte

Il faut toujours garantir $SR = 0$

Table de vérité simplifiée de la bascule RS

S	R	Q^+	Observation
0	0	Q	Point mémoire
0	1	0	Remise à Zéro
1	0	1	Mise à Un
1	1	X	État interdit

Équation caractéristique

$$Q^+ = S + R'Q \quad \text{avec } SR = 0$$

Interprétation

- Si $S = 1$: $Q^+ = 1$ (quel que soit Q)
- Si $S = 0$ et $R = 0$: $Q^+ = Q$ (mémoire)
- Si $S = 0$ et $R = 1$: $Q^+ = 0$ (reset)

Exemple de fonctionnement : Séquence temporelle

Évolution de la bascule RS

Instant	S	R	Q (avant)	Q (après)
t_0	0	0	0	0
t_1	1	0	0	1 (Set)
t_2	0	0	1	1 (Mémoire)
t_3	0	1	1	0 (Reset)
t_4	0	0	0	0 (Mémoire)
t_5	1	0	0	1 (Set)

Observations

- L'état change immédiatement quand S ou R est actif
- L'état est maintenu tant que S=R=0
- Bascule asynchrone : pas besoin d'horloge

Nécessité de la synchronisation

Problème des bascules asynchrones

- Changements d'état à n'importe quel moment
- Difficile de coordonner plusieurs bascules
- Risques de désynchronisation dans les systèmes complexes

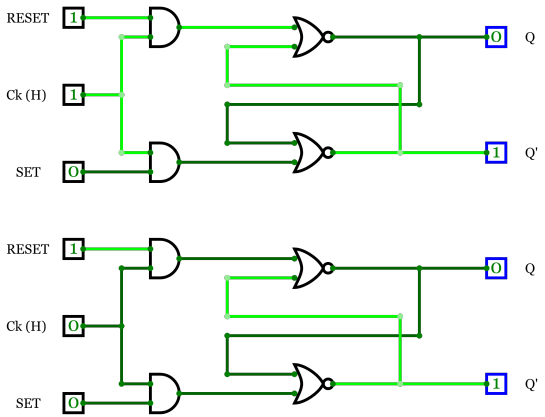
Solution : Bascule synchrone

- Ajout d'une entrée d'horloge **H** (ou CLK)
- La bascule ne change d'état que si $H = 1$
- Si $H = 0$: les entrées S et R sont ignorées
- Tous les changements synchronisés sur l'horloge

Avantage

Contrôle précis du moment où les changements d'état se produisent

Bascule RS synchrone : Circuit

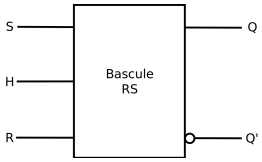


LISV | UVSQ

Principe

- Deux portes AND en entrée

Bascule RS synchrone : Symbole et table de vérité



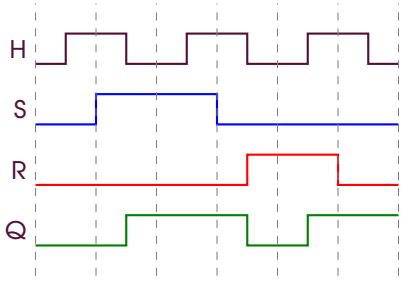
H	S	R	Q^+	Observation
0	X	X	Q	Mémoire
1	0	0	Q	Mémoire
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	X	Interdit

Fonctionnement

- Quand $H = 0$: bascule "gelée", conserve son état
- Quand $H = 1$: fonctionne comme une RS normale
- X = Don't Care (valeur indifférente)

Chronogramme : Bascule RS synchrone

Diagramme temporel



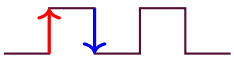
Observations

- Q change uniquement sur les fronts montants de H
- Entre t_1 et t_2 : $S = 1, H = 1 \Rightarrow Q = 1$ (Set)
- Entre t_3 et t_4 : $R = 1, H = 1 \Rightarrow Q = 0$ (Reset)

Types de déclenchement

Sensibilité au signal d'horloge

- **Niveau haut** (level-triggered) : Active tant que $H = 1$
- **Front montant** (positive edge-triggered) : Active sur $0 \rightarrow 1$
- **Front descendant** (negative edge-triggered) : Active sur $1 \rightarrow 0$



Standard moderne

Les bascules modernes sont généralement déclenchées sur front (edge-triggered) pour éviter les changements multiples pendant $H = 1$

Bascule JK : Motivation

Problème de la bascule RS

La combinaison $S = R = 1$ est interdite, ce qui :

- Complique l'utilisation
- Nécessite des circuits de contrôle supplémentaires
- Peut causer des erreurs

Solution : Bascule JK

- Fonctionne comme RS avec $J \equiv S$ et $K \equiv R$
- **Mais** : $J = K = 1$ n'est pas interdit !
- Comportement : **basculement** (toggle) quand $J = K = 1$
- Toujours synchrone (nécessite une horloge)

Avantage

Toutes les combinaisons d'entrées sont valides !

Bascule JK : Table de vérité

J	K	Q^+	Observation
0	0	Q	Mémoire
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}$	Toggle

Comportements :

- $J = 0, K = 0$: Conservation de l'état
- $J = 0, K = 1$: Force $Q = 0$ (comme R)
- $J = 1, K = 0$: Force $Q = 1$ (comme S)
- $J = 1, K = 1$: Inversion de l'état

Innovation

Le mode Toggle ($J = K = 1$) est unique à la JK !

Équation caractéristique

$$Q^+ = J\overline{Q} + \overline{K}Q$$

Vérification de l'équation $Q^+ = J\overline{Q} + \overline{K}Q$

Test de tous les cas :

- **Cas 1** : $J = 0, K = 0$

$$Q^+ = 0 \cdot \overline{Q} + 1 \cdot Q = Q$$

(Mémoire)

- **Cas 2** : $J = 0, K = 1$

$$Q^+ = 0 \cdot \overline{Q} + 0 \cdot Q = 0$$

(Reset)

- **Cas 3** : $J = 1, K = 0$

$$Q^+ = 1 \cdot \overline{Q} + 1 \cdot Q = \overline{Q} + Q = 1$$

(Set)

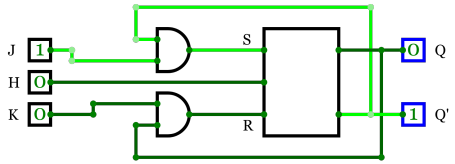
- **Cas 4** : $J = 1, K = 1$

$$Q^+ = 1 \cdot \overline{Q} + 0 \cdot Q = \overline{Q}$$

(Toggle)

L'équation est bien vérifiée pour tous les cas !

Bascule JK : Circuit

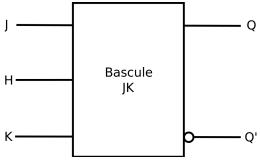


Structure

- Bascule RS de base avec rétroaction
- Les sorties Q et $\overline{Q}$ sont rebouclées vers les entrées
- J et $\overline{Q}$ alimentent une porte AND
- K et Q alimentent une autre porte AND
- L'horloge H synchronise le tout

Rétroaction intelligente : Les sorties influencent les entrées pour créer le comportement de Toggle.

Bascule JK : Symbole et table



J	K	Q^+	Observation
0	0	Q	Mémoire
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}$	Toggle

Entrées :

- J, K : Commandes
- H : Horloge (front montant)

Sorties :

- Q, $\overline{Q}$: État

Exemple : Séquence JK

Évolution de la bascule JK synchrone

Instant	H	J	K	Q (avant)	Q (après)
t_0	↑	0	0	0	0 (Mémoire)
t_1	↑	1	0	0	1 (Set)
t_2	↑	0	0	1	1 (Mémoire)
t_3	↑	1	1	1	0 (Toggle)
t_4	↑	1	1	0	1 (Toggle)
t_5	↑	0	1	1	0 (Reset)

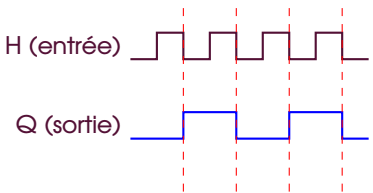
Observations

- Les changements ont lieu sur front montant (↑)
- Le mode Toggle ($J = K = 1$) inverse l'état à chaque coup d'horloge
- Très utile pour créer des diviseurs de fréquence

Application : Diviseur de fréquence par 2

Configuration

Bascule JK avec $J = K = 1$ (mode Toggle permanent)



Résultat : La sortie Q change à la moitié de la fréquence de H !

Utilisation

Créer des horloges de fréquences différentes (compteurs, séquenceurs)

Bascule D : Le concept le plus simple

Définition

La bascule D (Data ou Delay) possède une seule entrée de données D :

- Si $D = 0$: $Q^+ = 0$
- Si $D = 1$: $Q^+ = 1$
- La sortie **recopie** simplement l'entrée au rythme de l'horloge

Équation caractéristique

$$Q^+ = D$$

Simplicité

C'est la bascule la plus simple à utiliser : pas d'état interdit, pas de toggle, juste une copie !

Bascule D : Table de vérité

Table complète

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

Table simplifiée

D	Q^+
0	0
1	1

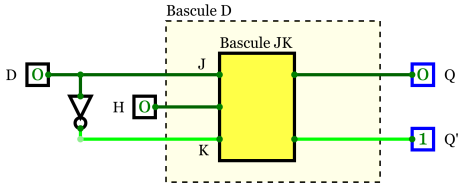
Interprétation :

- L'état présent Q n'a **aucune influence**
- L'état suivant Q^+ dépend uniquement de D
- C'est une "mémoire à retard" d'un cycle d'horloge

Nom "Delay"

La donnée D est "retardée" d'un cycle d'horloge avant d'apparaître en sortie

Bascule D : Circuit



Implémentation

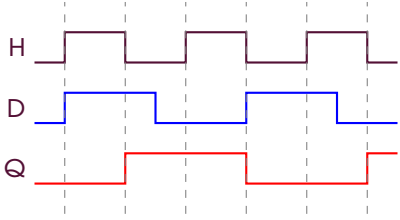
- Peut être construite à partir d'une bascule RS synchrone
- $S = D$ et $R = \overline{D}$
- Garantit que S et R sont toujours opposés : pas d'état interdit !
- Plus simple : circuit dédié avec moins de portes

Avantage majeur

Impossible d'avoir un état interdit : si $D = 1$ alors $R = 0$ et vice-versa

Chronogramme : Bascule D

Fonctionnement temporel



Observations

- Sur chaque front montant de H, Q prend la valeur de D
- Entre les fronts, Q conserve sa valeur (mémoire)
- Q est "une copie retardée" de D

Exemple : Séquence bascule D

Évolution pas à pas

Instant	H	D	Q (avant)	Q (après)
t_0	-	0	X	X (état initial)
t_1	↑	0	X	0 (D=0)
t_2	-	1	0	0 (pas de front)
t_3	↑	1	0	1 (D=1)
t_4	-	1	1	1 (pas de front)
t_5	↑	1	1	1 (D=1)
t_6	-	0	1	1 (pas de front)
t_7	↑	0	1	0 (D=0)

Règle importante

Q ne change que sur les fronts d'horloge, même si D change entre deux fronts !

Applications de la bascule D

Utilisations principales :

1. **Registres** : Stockage de données
   - Les registres de processeurs sont faits de bascules D
   - Chaque bit est une bascule D
2. **Mémoires tampons** : Synchronisation de données
   - Alignement de signaux sur l'horloge
   - Élimination de signaux transitoires
3. **Pipeline** : Registres entre étages
   - Mémorisation des résultats intermédiaires
   - Augmentation de la fréquence d'horloge
4. **Échantillonnage** : Capture de signaux
   - Conversion analogique-numérique
   - Acquisition de données

Comparaison des bascules

Type	Entrées	Équation	Particularité
RS	S, R	$Q^+ = S + R'Q$ (avec $SR = 0$)	État interdit ($S = R = 1$)
JK	J, K	$Q^+ = J\overline{Q} + \overline{K}Q$	Toggle ($J = K = 1$) Pas d'état interdit
D	D	$Q^+ = D$	La plus simple Copie directe
T	T	$Q^+ = T\overline{Q} + \overline{T}Q$ $Q^+ = Q \oplus T$	Toggle si $T = 1$ Mémoire si $T = 0$

Note

La bascule T est une simplification de la JK avec $J = K = T$

Conversions entre bascules

RS $\rightarrow$ JK

Ajouter la rétroaction : $S = J\overline{Q}$ et $R = KQ$

JK $\rightarrow$ D

Relier les entrées : $J = D$ et $K = \overline{D}$

JK $\rightarrow$ T

Relier les entrées : $J = K = T$

D $\rightarrow$ T

Ajouter un XOR : $D = Q \oplus T$

Interchangeabilité

On peut implémenter n'importe quel type de bascule à partir d'un autre type avec quelques portes logiques supplémentaires

Qu'est-ce qu'un registre ?

Définition

Un registre est un ensemble de bascules permettant de mémoriser un mot binaire de n bits

Caractéristiques

- Composé de n bascules (généralement de type D)
- Peut mémoriser un nombre de n bits
- Toutes les bascules partagent la même horloge
- Opérations : lecture et écriture

Exemple

Un registre 8 bits = 8 bascules D = peut stocker un octet (0 à 255)

Dans un processeur

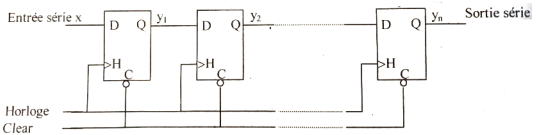
Les registres sont les mémoires les plus rapides et les plus proches du CPU

Classification des registres

Selon le mode d'entrée/sortie :

1. **Série-Série** :
   - Écriture série (bit par bit)
   - Lecture série (bit par bit)
2. **Parallèle-Parallèle** :
   - Écriture parallèle (tous les bits ensemble)
   - Lecture parallèle (tous les bits ensemble)
3. **Série-Parallèle** :
   - Écriture série
   - Lecture parallèle
4. **Parallèle-Série** :
   - Écriture parallèle
   - Lecture série

Registre à décalage série-série



Principe

- Les bascules D sont connectées en cascade
- La sortie Q_i est reliée à l'entrée D_{i+1}
- À chaque coup d'horloge : décalage d'une position vers la droite
- Entrée série : un bit à la fois sur D_0
- Sortie série : un bit à la fois sur Q_{n-1}

Fonctionnement du registre série-série

Exemple : Registre 4 bits, écriture de 1101

Cycle	Entrée	Q_0	Q_1	Q_2	Q_3
0	-	0	0	0	0
1	1	1	0	0	0
2	0	0	1	0	0
3	1	1	0	1	0
4	1	1	1	0	1

Observations

- Il faut 4 cycles pour écrire 4 bits
- Les données "glissent" de gauche à droite
- Après 4 cycles : $Q_3 Q_2 Q_1 Q_0 = 1101$

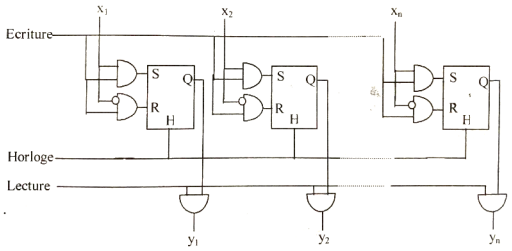
Lenteur : Le mode série est plus lent que le mode parallèle

Applications du registre série-série

Utilisations :

- **Communication série** : UART, SPI, I2C
  - Transmission de données sur un seul fil
  - Réduction du nombre de connexions
- **Conversion série-parallèle**
  - Réception de données série
  - Conversion en format parallèle
- **Ligne à retard**
  - Retarder un signal de n cycles
  - Synchronisation temporelle
- **Traitement du signal**
  - Filtres numériques
  - Corrélateurs

Registre parallèle-parallèle



Principe

- n bascules D indépendantes
- Toutes partagent la même horloge
- Entrées : $D_0, D_1, \ldots, D_{n-1}$ (parallèles)
- Sorties : $Q_0, Q_1, \ldots, Q_{n-1}$ (parallèles)
- Un seul cycle pour écrire/lire n bits !

Fonctionnement du registre parallèle-parallèle

Exemple : Registre 4 bits

Cycle	Entrées				Sorties			
	D_3	D_2	D_1	D_0	Q_3	Q_2	Q_1	Q_0
0	-	-	-	-	0	0	0	0
1	1	0	1	1	1	0	1	1
2	0	1	0	0	0	1	0	0

Observations

- Un seul cycle pour charger les 4 bits !
- Tous les bits sont écrits/lus simultanément
- Beaucoup plus rapide que le mode série

Avantage

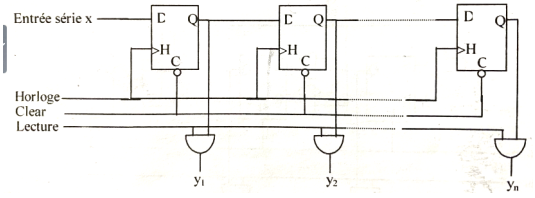
Vitesse : transfert de n bits en un seul cycle

Applications du registre parallèle-parallèle

Utilisations :

- **Registres de processeur**
  - Registres généraux (EAX, EBX, etc.)
  - Compteur de programme (PC)
  - Registre d'instruction
- **Mémoire tampon (buffer)**
  - Stockage temporaire de données
  - Interface entre systèmes à vitesses différentes
- **Pipeline des processeurs**
  - Registres inter-étages
  - Mémorisation des résultats intermédiaires
- **Ports d'entrée/sortie**
  - Communication parallèle rapide
  - Bus de données

Registre série-parallèle



Principe

- **Écriture série** : données entrées bit par bit
- **Lecture parallèle** : toutes les sorties accessibles simultanément
- Combine registre à décalage et sorties parallèles
- Nécessite n cycles pour écrire, 1 cycle pour lire

Fonctionnement série-parallèle

Exemple : Chargement de 1101

Cycle	Entrée	Q_3	Q_2	Q_1	Q_0
0	-	0	0	0	0
1	1	1	0	0	0
2	0	0	1	0	0
3	1	1	0	1	0
4	1	1	1	0	1

Après 4 cycles :

- Toutes les sorties $Q_3Q_2Q_1Q_0 = 1101$ sont disponibles
- Lecture instantanée de tous les bits

Application typique

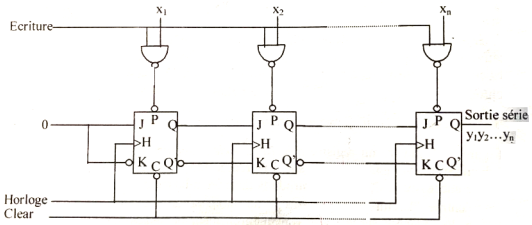
Convertisseur série-parallèle pour réception de données

Applications série-parallèle

Utilisations :

- **Récepteur série**
  - UART, RS-232
  - Réception de données série
  - Conversion pour traitement parallèle
- **Désérialisation**
  - Interface entre bus série et bus parallèle
  - Augmentation du débit de traitement
- **Décodage de protocoles**
  - Réception de trames série
  - Analyse parallèle du contenu

Registre parallèle-série



Principe

- **Écriture parallèle** : toutes les données chargées en un coup
- **Lecture série** : données sorties bit par bit
- Nécessite 1 cycle pour écrire, n cycles pour lire
- Utilise un multiplexeur ou un mode "shift"

Fonctionnement parallèle-série

Exemple : Transmission de 1101

Cycle	Mode	Q_3	Q_2	Q_1	Q_0	Sortie
0	Charge	1	1	0	1	-
1	Shift	0	1	1	0	1
2	Shift	0	0	1	1	0
3	Shift	0	0	0	1	1
4	Shift	0	0	0	0	1

Observations

- Cycle 0 : chargement parallèle instantané
- Cycles 1-4 : décalage et sortie série
- Ordre de sortie : Q_0, Q_1, Q_2, Q_3 (LSB first)

Applications parallèle-série

Utilisations :

- **Émetteur série**
  - UART, SPI
  - Transmission de données sur un seul fil
  - Réduction des broches nécessaires
- **Sérialisation**
  - Conversion parallèle $\rightarrow$ série
  - Communication longue distance
  - Protocoles série (USB, Ethernet)
- **Multiplexage temporel**
  - Partage d'un canal de communication
  - Transmission séquentielle de plusieurs données

Comparaison des types de registres

Type	Écriture	Lecture	Application
Série-Série	n cycles bit/cycle	n cycles bit/cycle	Communication série Ligne à retard
Parallèle- Parallèle	1 cycle n bits	1 cycle n bits	Registres CPU Mémoire rapide
Série- Parallèle	n cycles bit/cycle	1 cycle n bits	Récepteur série Désérialisation
Parallèle- Série	1 cycle n bits	n cycles bit/cycle	Émetteur série Sérialisation

Compromis

Vitesse vs Nombre de connexions

- Parallèle : rapide mais nécessite n fils
- Série : lent mais nécessite 1 seul fil

Qu'est-ce qu'un compteur ?

Définition

Un compteur est un circuit séquentiel qui génère une séquence binaire prédéfinie à chaque impulsion d'horloge

Types principaux

- **Compteur binaire** : 0, 1, 2, 3, ... (binaire naturel)
- **Compteur modulo-N** : compte de 0 à N-1 puis recommence
- **Compteur/décompteur** : peut compter dans les deux sens
- **Compteur BCD** : compte de 0 à 9 (décimal)

Applications

Horloges, timers, générateurs de fréquences, séquenceurs, diviseurs

Compteurs asynchrones vs synchrones

Asynchrone (ripple)

- Chaque bascule déclenche la suivante
- Horloge uniquement sur la 1ère bascule
- Propagation en cascade
- Plus simple
- Plus lent (délais cumulés)

Problème

Délai de propagation important pour n bits

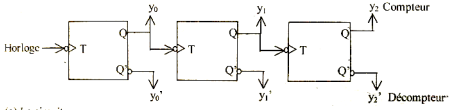
Synchrone

- Toutes les bascules sur même horloge
- Changement simultané
- Logique de contrôle plus complexe
- Plus rapide
- Fréquence max élevée

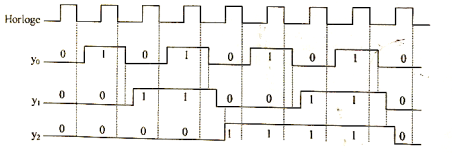
Avantage

Pas de délai de propagation entre bascules

Compteur asynchrone 4 bits



(a) Le circuit



Principe

- Bascules JK ou T en mode Toggle ($J = K = 1$ ou $T = 1$)
- La sortie Q_i commande l'horloge de la bascule suivante
- Compte de 0000 à 1111 (0 à 15)
- Aussi appelé "Ripple Counter" (propagation en cascade)

Fonctionnement du compteur asynchrone

Séquence de comptage

Impulsion	Q_3	Q_2	Q_1	Q_0	Décimal
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
...	...	...	...	...	...
15	1	1	1	1	15
16	0	0	0	0	0 (reset)

Modulo

Compteur 4 bits = Modulo-16 (compte de 0 à 15)

Analyse du compteur asynchrone

Détail des transitions

Passage de 0111 (7) à 1000 (8) :

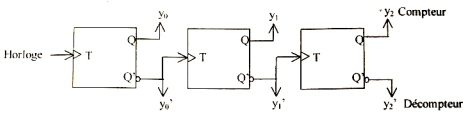
1. $Q_0 : 1 \rightarrow 0$ (front descendant déclenche Q_1)
2. $Q_1 : 1 \rightarrow 0$ (front descendant déclenche Q_2)
3. $Q_2 : 1 \rightarrow 0$ (front descendant déclenche Q_3)
4. $Q_3 : 0 \rightarrow 1$

Problème de délai

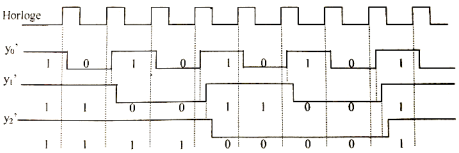
Les changements se propagent séquentiellement :

- Délai total = n $\times$ délai d'une bascule
- Pour 8 bits : 8 $\times$ t_d avant stabilisation
- Limite la fréquence maximale

Chronogramme du compteur asynchrone



(a) Le circuit



Observations

- Q_0 change à chaque front d'horloge : fréquence $\div$ 2
- Q_1 change à chaque front de Q_0 : fréquence $\div$ 4
- Q_2 change à chaque front de Q_1 : fréquence $\div$ 8
- Q_3 change à chaque front de Q_2 : fréquence $\div$ 16

Applications du compteur asynchrone

Utilisations

- **Diviseurs de fréquence**
  - Génération d'horloges lentes à partir d'une horloge rapide
  - Exemple : Horloge 1 MHz $\rightarrow$ 1 kHz, 1 Hz, etc.
- **Compteurs simples**
  - Applications basse vitesse
  - Où le délai n'est pas critique
- **Timers**
  - Mesure d'intervalles de temps
  - Génération de délais

Limitation

Non adapté aux applications haute fréquence à cause des délais de propagation

Compteur modulo-N : Principe

Définition

Un compteur modulo-N compte de 0 à N-1, puis revient à 0

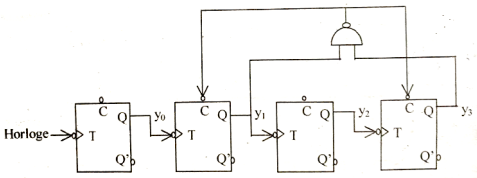
Méthode

1. Utiliser un compteur binaire normal
2. Détecter l'état N avec une porte logique
3. Réinitialiser le compteur quand N est atteint

Exemples

- Modulo-10 : compte de 0 à 9 (compteur décimal/BCD)
- Modulo-60 : compte de 0 à 59 (secondes/minutes)
- Modulo-24 : compte de 0 à 23 (heures)

Compteur modulo-10 (BCD)



Principe

- Compteur 4 bits normal (peut compter jusqu'à 15)
- Porte NAND détecte l'état 1010 (10 en décimal)
- Quand 10 est atteint : RESET des bascules
- Revient immédiatement à 0000

Fonctionnement du modulo-10

Séquence de comptage

#	Q_3	Q_2	Q_1	Q_0	Dec
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	0	0	0	0	RESET

Détection de l'état 10 :

- État 10 = 1010_2
- $Q_3 = 1$ et $Q_1 = 1$
- NAND(Q_3, Q_1) = 0
- Signal RESET actif (bas)
- Toutes les bascules remises à 0

Important

L'état 10 n'est présent que très brièvement (durée de propagation)

Généralisation : Compteur modulo-N quelconque

Méthode de conception

Pour créer un compteur modulo-N :

1. Déterminer le nombre de bits nécessaires : $n = \lceil \log_2(N) \rceil$
2. Créer un compteur n bits
3. Identifier la combinaison binaire de N
4. Concevoir la logique de détection (portes AND/NAND)
5. Connecter à l'entrée RESET

Exemple : Modulo-7

- $7 = 111_2 \rightarrow$ nécessite 3 bits
- Détecter $Q_2 \cdot Q_1 \cdot Q_0$ (toutes à 1)
- RESET quand cette condition est vraie

Applications des compteurs modulo-N

Utilisations courantes :

- **Horloges numériques**
  - Modulo-10 : chiffres 0-9
  - Modulo-6 : dizaines de secondes/minutes (0-5)
  - Modulo-24 : heures (0-23)
- **Diviseurs de fréquence programmables**
  - Division par N arbitraire
  - Génération de fréquences spécifiques
- **Séquenceurs**
  - Machines à états finis
  - Contrôle de processus cycliques
- **Générateurs d'adresses**
  - Balayage cyclique de mémoires
  - Adressage périodique

Compteurs synchrones

Principe du compteur synchrone :

- Toutes les bascules reçoivent la même horloge
- Logique combinatoire détermine quelles bascules doivent basculer
- Tous les changements se font simultanément
- Pas de délai de propagation entre étages

Équations pour compteur binaire synchrone

Pour chaque bit i :

- Q_i bascule si tous les bits de poids inférieur sont à 1
- $T_0 = 1$ (toujours)
- $T_1 = Q_0$
- $T_2 = Q_0 \cdot Q_1$
- $T_3 = Q_0 \cdot Q_1 \cdot Q_2$
- etc.

Avantages des compteurs synchrones

Avantages

- Pas de délai de propagation
- Fréquence max élevée
- Tous les bits changent ensemble
- Pas d'états transitoires
- Adapté aux hautes fréquences

Inconvénients

- Plus de portes logiques
- Circuit plus complexe
- Consommation plus élevée
- Coût supérieur

Performance

Peut fonctionner 10× plus vite qu'un asynchrone !

Compromis

Vitesse vs Complexité

Choix

- Asynchrone : applications basse vitesse, circuits simples
- Synchrone : systèmes haute performance, processeurs

Applications des circuits séquentiels

Dans les systèmes numériques modernes :

- **Processeurs**
  - Registres (bascules D)
  - Pipeline (registres inter-étages)
  - Compteur de programme (PC)
- **Mémoires**
  - RAM, caches (bascules D)
  - Registres à décalage
- **Communications**
  - UART, SPI (registres à décalage)
  - Protocoles série/parallèle
- **Temporisation**
  - Horloges, timers (compteurs)
  - Générateurs de signaux