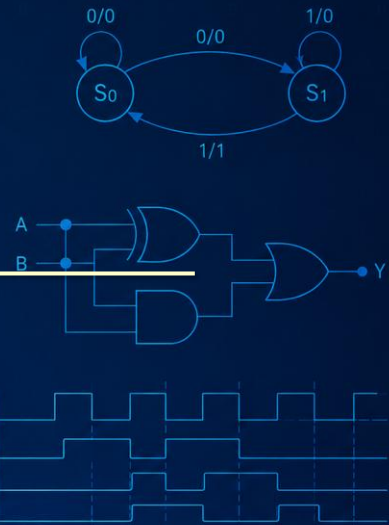


V. Logique séquentielle

Registres, compteurs, machines d'états


ENSGTI
ÉCOLE D'INGÉNIEURS

Arnaud Martin

1

1



V. Logique
séquentielle

Logique séquentielle – rappel logique combinatoire

Rappel : Une **fonction logique combinatoire** est une fonction qui prend un ou plusieurs bits d'entrée et produit un ou plusieurs bits de sortie. Les sorties de fonctions logiques combinatoires **ne dépendent que des valeurs actuelles des bits d'entrée**, et ne tiennent pas compte de l'historique des entrées ou des sorties, contrairement aux sorties des fonctions logiques séquentielles.

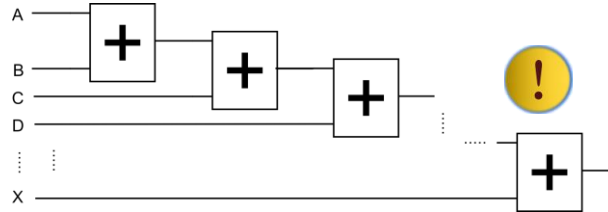
La logique combinatoire est utilisée par exemple pour les **transcodeurs** (passage d'un code à un autre), le **roulage** de données (multiplexeur, démultiplexeur) ou les **opérateurs arithmétiques** (additionneur, soustracteur, multiplieur). La synthèse d'un bloc logique combinatoire se fait à partir d'une table de vérité, exprimant les sorties de la fonction logique en fonction des entrées.

2

2

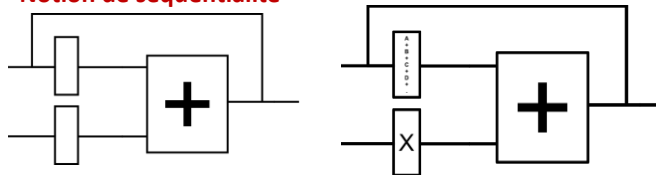
Généralités

$A+B+C+D+\dots$



Calcul les uns après les autres

Notion de séquentialité



3

3

Compteur : état $N \Rightarrow$ état $N+1$



Incrémenter d'un état



L'état suivant dépend de l'état présent.

Il faut connaître l'état présent.

Être capable de mémoriser, de stabiliser l'état présent

4

4

Les portes logiques combinatoires n'ont pas de « **mémoire** », la valeur de leurs sorties ne dépend que de l'état des entrées.

Ces portes adaptées au décodage ou aux opérations arithmétiques ne permettent pas, seules, de décrire un **système** dont l'**état** évolue.

Un état = une configuration possible du système.

Exemples :	lampe	allumée, éteinte, cassée
	feu tricolore	rouge, vert, orange clignotant, hors service
	ordinateur	allumé, en veille, en veille prolongé, éteint
	machine à laver	prélavage, lavage, rinçage, essorage, arrêt
	compteur	0, 1, 2, 3, ..., N
	smartphone	verrouillé, déverrouillé

Des **événements** font évoluer le système (passage d'un état à un autre).

Exemples : signaux logiques activés (capteurs, boutons), durée écoulée (Timer).

5

5

Fonction logique combinatoire :



Entrée : interrupteur / sortie : ampoule
La valeur de la sortie ne dépend que de la valeur de l'entrée.

Fonction logique séquentielle :



Entrée : interrupteur / sortie : feu piéton
La valeur de la sortie dépend de l'état du système et de l'entrée.

Idée : en logique séquentielle, le système se souvient que l'on a appuyé et sait dans quel état il est.

6

6

Besoin d'un autre type de structures logiques :

Portes logiques séquentielles → bascules

Points mémoires

1. Bascules

Réalisées à partir de portes logiques élémentaires (ensembles de transistors). Eléments fondamentaux permettant de **mémoriser** et de **modifier l'information**.

Utilisées pour le stockage temporaire de données et la réalisation d'automates pour le contrôle de l'évolution des systèmes.



Remarque : les compteurs sont un cas particulier d'automate.

7

7

1. Bascules

Opérateurs élémentaires de mémorisation.

Leur état dépend de celui de leurs entrées mais également de leur état précédent.

$$\text{état}(i+1) = f(\text{état}(i), \text{entrées})$$

(verrouillé + touche écran => déverrouillé)

On distingue les bascules **asynchrones** sensibles au niveau des entrées, et les bascules **synchrones** sensibles au front d'un signal spécial cadencant le système (typiquement un signal d'horloge).

Historiquement : bascules RS, bascules JK. **À l'heure actuelle, uniquement des bascules synchrones, les bascules D** à front constituées de 2 bascules *latch* en série.

8

8

Bascule latch

Entrée d'état D (état) et entrée de commande (LE *latch enable*).

Q : état présent

Q+ : état suivant

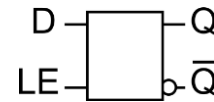
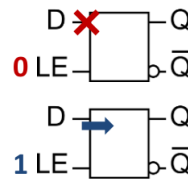


Table de vérité de la bascule latch

LE	D	Q+
0	0	Q
0	1	Q
1	0	0
1	1	1

mode verrouillé, mémorisation

mode transparent



Idee : soit on laisse entrer une nouvelle valeur, soit on verrouille l'entrée et on mémorise le contenu.

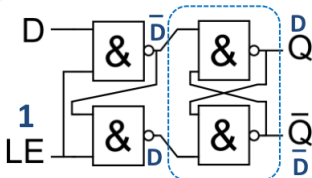
9

9

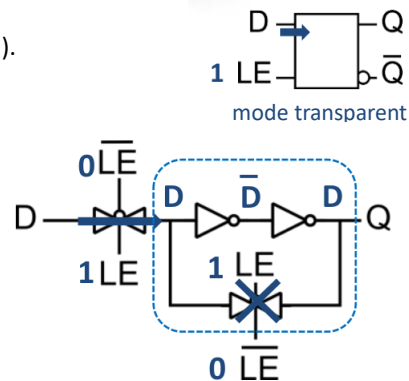
Bascule latch : structures internes

Entrée de donnée D (état) et entrée de commande (LE).

Q : état présent , Q+ : état suivant



Structure à portes NAND



Structure compacte à portes de transmission et inverseurs

Idee : c'est le rebouclage interne qui permet de mémoriser un état. La contre-réaction stabilise l'état.

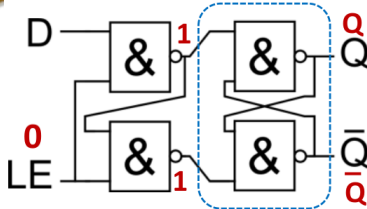
10

10

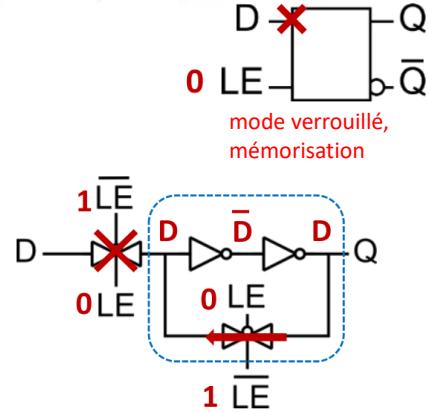
Bascule latch : structures internes

Entrée de donnée D (état) et entrée de commande (LE).

Q : état présent , Q+ : état suivant



Structure à portes NAND

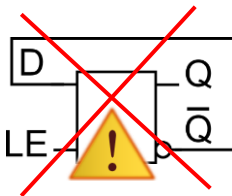


Structure compacte à portes de transmission et inverseurs

Idée : c'est le rebouclage interne qui permet de mémoriser un état.

11

11



Pas possible avec ce type de bascule

Compteur :

0	1	2	3	4	5
0	0	0	0	1	1
0	0	1	1	0	0
0	1	0	1	0	1

C'est un rebouclage utilisé dans beaucoup de systèmes séquentiels (compteurs par exemple).

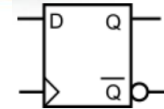
Besoin d'une structure pas entièrement transparente.

12

12

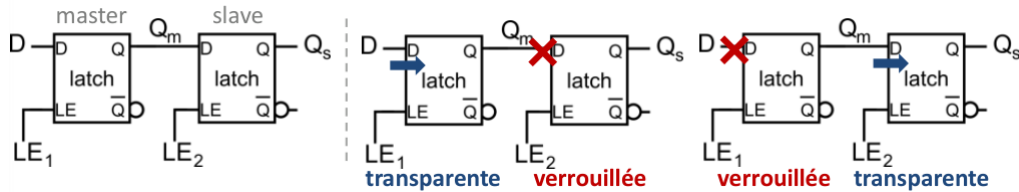
Basculer D à front

C'est LA porte logique
séquentielle dont on se
sert en pratique



à retenir !

La bascule D est **non transparente**. Pas de liaison directe entre l'entrée et la sortie. Elle résulte de l'association en série de deux bascules *latch*.



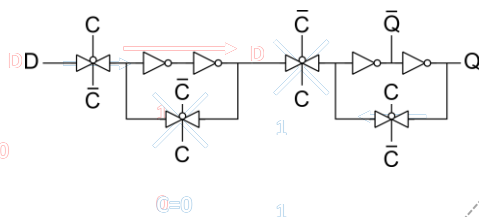
Une bascule maître et une bascule esclave. Lorsque l'une est transparente, l'autre est verrouillée.

Pour garantir un fonctionnement correct, il faut des signaux de commandes LE_1 et LE_2 sans recouvrement.

13

13

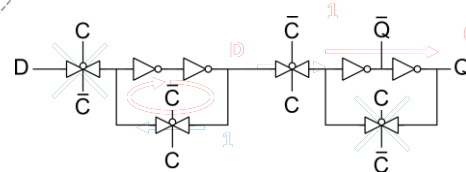
Structure interne possible de la bascule D à front.



comment
ça marche ?

Table de vérité de la
bascule D à front

D	C	Q+
0	1	0
1	1	1



C : 0 → 1

14

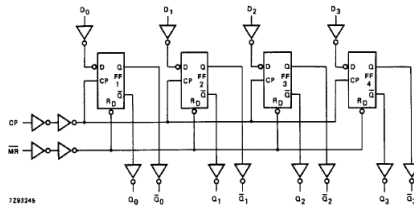
14



Remarques générales en pratique :

Les bascules comprennent des entrées supplémentaires.

Il faut respecter des temps de mise en place
et de maintien des niveaux logiques pour garantir
un bon fonctionnement des bascules.



datasheet 74HC175 Edge Triggered Flip-Flop

Fig.5 Logic diagram.

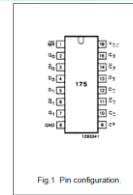


Fig.1 Pin configuration.

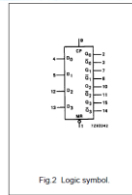
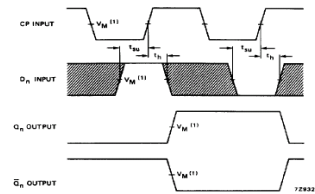


Fig.2 Logic symbol.



The shaded areas indicate when the input is permitted to change for predictable output performance.

(1) HC : $V_M = 50\%$; $V_I = GND$ to V_{CC} .
HCT : $V_M = 1.3 V$; $V_I = GND$ to $3 V$.

Fig.7 Waveforms showing the data set-up and hold times for the data input (D_n).

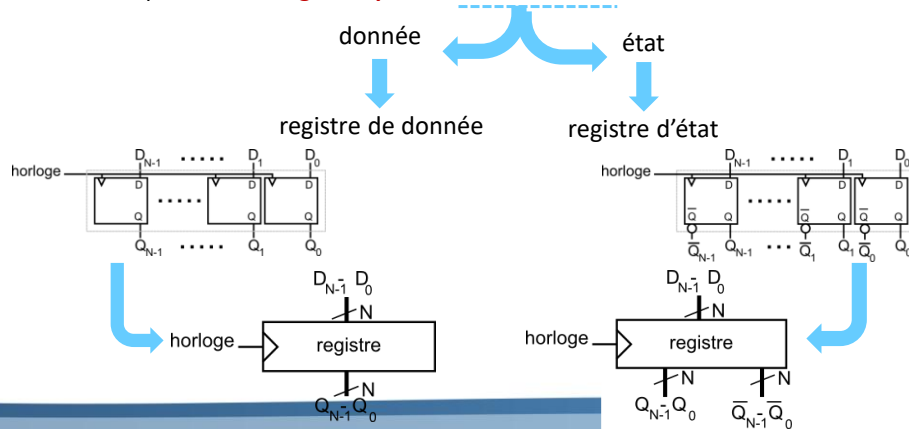
15

15

Registres

registre = ensemble de bascules

pour le **stockage temporaire d'information**

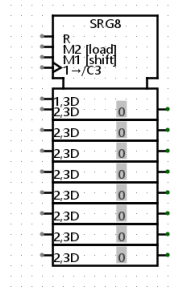


16

16

Autres types de registres

- Registre à chargement parallèle
- [Registre à décalage](#)
- Conversion série-parallèle
- Conversion parallèle-série
- Génération SBPA



Les données sont traitées sous forme parallèle par les opérateurs arithmétiques. Les données lors de communications entre équipements peuvent être sous forme série



Quelles communications série connaissez-vous ?

17

17

3. Machines à états finis (MAEF)

Finite State Machine (FSM) ou automate synchrones.

Une machine d'état est un **opérateur séquentiel** dont la sortie est fonction des entrées et de l'état présent de la machine d'état. Elle utilise des bascules pour mémoriser l'état présent, et des **blocs de logique combinatoire** pour générer les sorties et l'état futur.

Il existe 2 types de structure de machine d'état:

1. Machine de **Mealy** : les sorties dépendent de l'état et des entrées.
2. Machine de **Moore** : les sorties dépendent exclusivement de l'état.

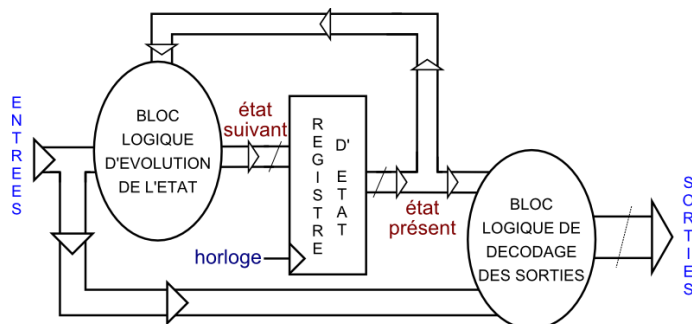


Remarque : les compteurs sont un cas particulier de machine à états finis.

18

18

3.1 Machines à états finis (MAEF) type Mealy



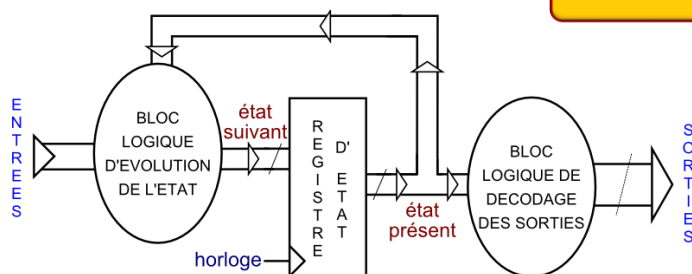
Les sorties peuvent changer de manière asynchrone.

À éviter si non indispensable.

19

19

3.2 Machines à états finis (MAEF) type Moore



C'est la structure dont on se sert en pratique.

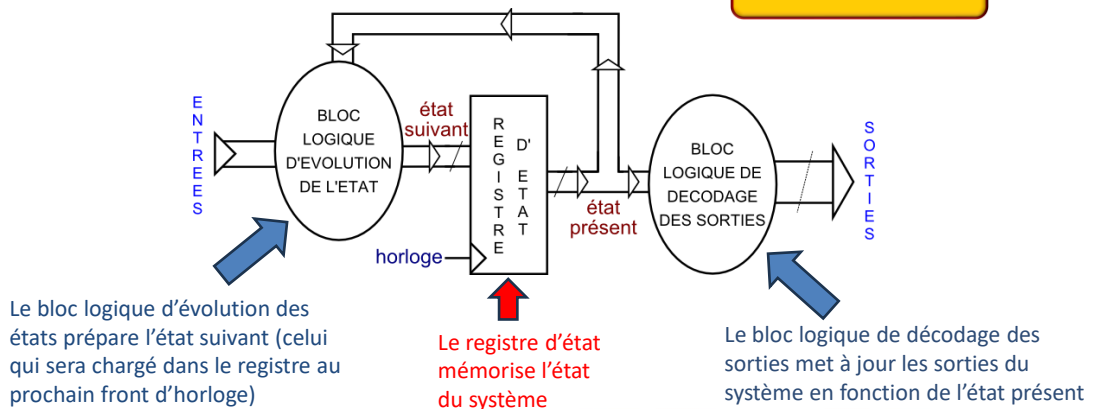
Le type Moore garantit des **signaux logiques synchrones**. Il est privilégié dans les applications intégrées et cœur de calculateur pour le séquenceur.

20

20

3.2 Machines à états finis (MAEF) type Moore

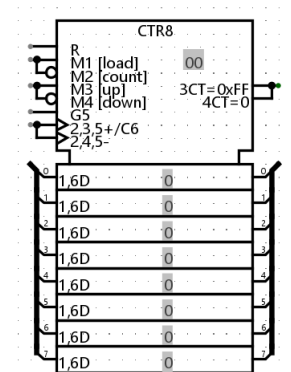
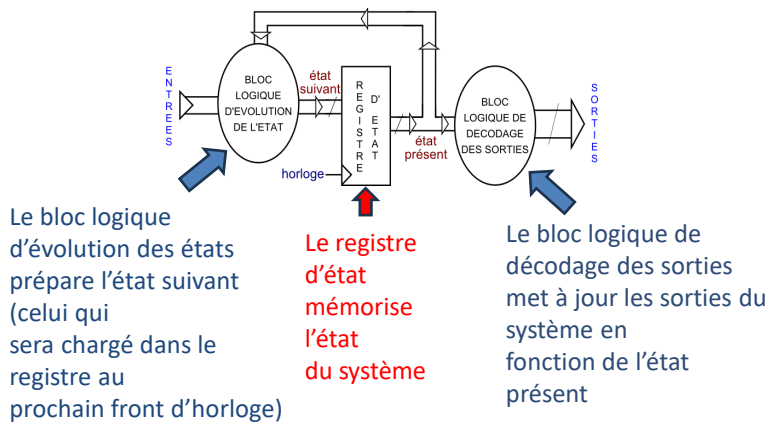
C'est la structure dont on se sert en pratique.



21

21

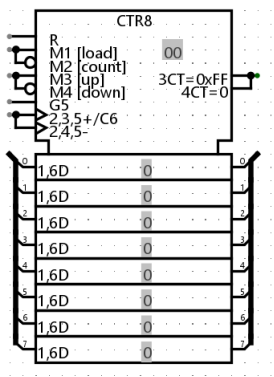
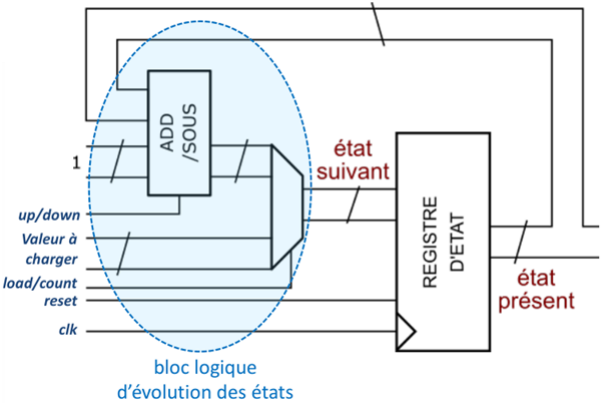
3.2 Machines à états finis (MAEF) type Moore : exemple d'un compteur [1/3]



22

22

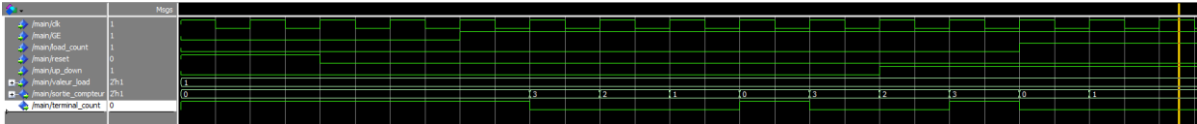
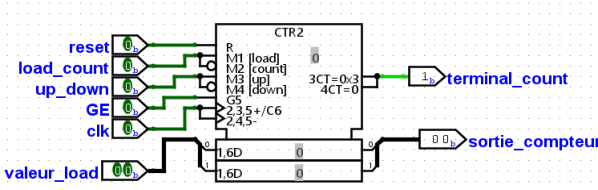
3.2 Machines à états finis (MAEF) type Moore : exemple d'un compteur [2/3]



23

23

3.2 Machines à états finis (MAEF) type Moore : exemple d'un compteur [3/3]



24

24

3.4 Machines à états finis (MAEF), synthèse



En **6 étapes** :

1. Déterminer le diagramme d'état en fonction du cahier des charges.

Identifier les entrées, les sorties et le nombre d'états.

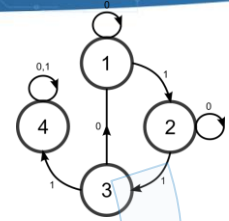
2. Choisir un codage pour la représentation des états (*par défaut binaire*).

3. Établir la table d'évolution des états.

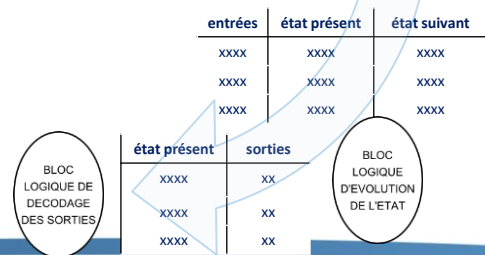
4. Synthétiser les blocs logiques d'évolution des états.

5. Établir la table de vérité des sorties.

6. Synthétiser le bloc logique de décodage des sorties



binaire, Gray, One hot, ... ?



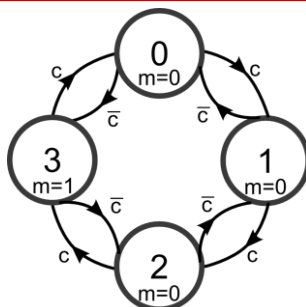
25

25

Exemple : compteur-décompteur, avec m actif pour la valeur maximum

1. Diagramme d'états

une entrée c, une sortie m, 4 états



c=1 : comptage
c=0 : décomptage

2. Choix d'un codage pour les états

ici exemple avec binaire, Gray, one hot

c	état présent	état suivant
comptage		
1	0	1
1	1	2
1	2	3
1	3	0
0	0	3
0	1	0
0	2	1
0	3	2
décomptage		
0	0	1
0	1	2
0	2	3
0	3	0

26

26

Exemple : compteur-décompteur

3. Établir la table d'évolution des étatsTable d'évolution avec codage binaire

c	état présent			état suivant		
		Q1	Q0		D1	D0
1	0	0	0	1	0	1
1	1	0	1	2	1	0
1	2	1	0	3	1	1
1	3	1	1	0	0	0
0	0	0	0	3	1	1
0	1	0	1	0	0	0
0	2	1	0	1	0	1
0	3	1	1	2	1	0

27

27

Exemple : compteur-décompteur

4. Synthétiser le bloc logique d'évolution des étatstable d'évolution avec codage binaire

c	état présent			état suivant		
		Q1	Q0		D1	D0
1	0	0	0	1	0	1
1	1	0	1	2	1	0
1	2	1	0	3	1	1
1	3	1	1	0	0	0
0	0	0	0	3	1	1
0	1	0	1	0	0	0
0	2	1	0	1	0	1
0	3	1	1	2	1	0

28

28

Exemple : compteur-décompteur

5. Établir la table de vérité des sorties

avec codage binaire

état présent	m	
	Q1	Q0
0	0	0
1	0	1
2	1	0
3	1	1

Ne dépend que de l'état présent (machine de type Moore)

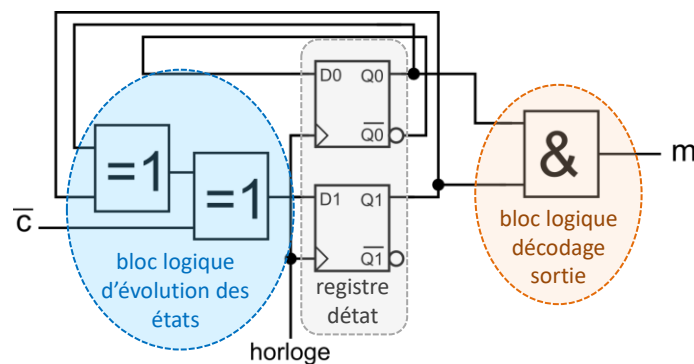
6. Synthétiser le bloc logique de décodage des sorties

29

29

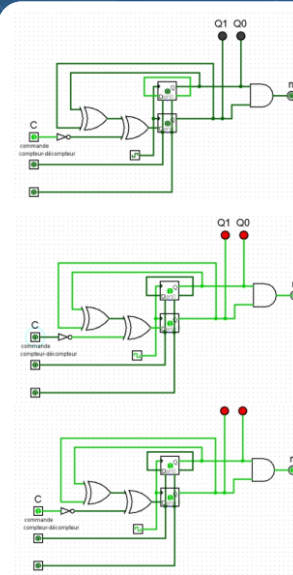
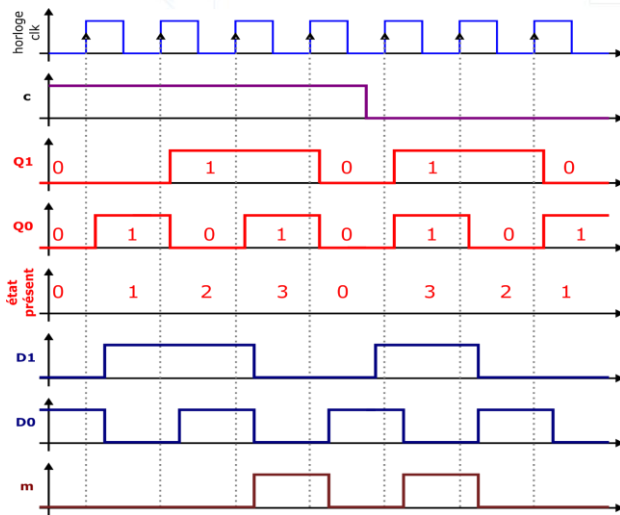
Exemple : compteur-décompteur 4

Avec codage binaire



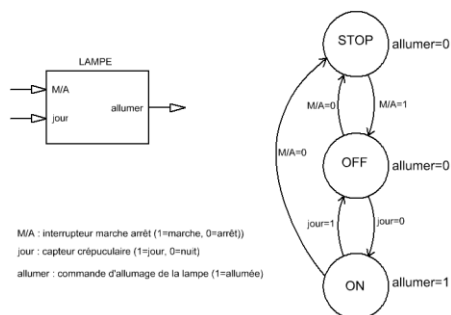
30

30



Lampe crépusculaire :

La machine d'états suivante représente le fonctionnement du circuit de contrôle d'une lampe qui s'allume automatiquement à la tombée de la nuit. :



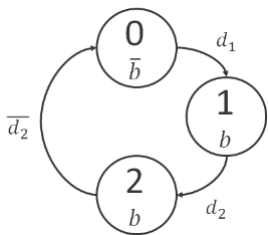


Diagramme d'état



3 états → 2 bascules
Machine d'état à 3 états, 2 entrées d1 et d2 et 1 sortie b.

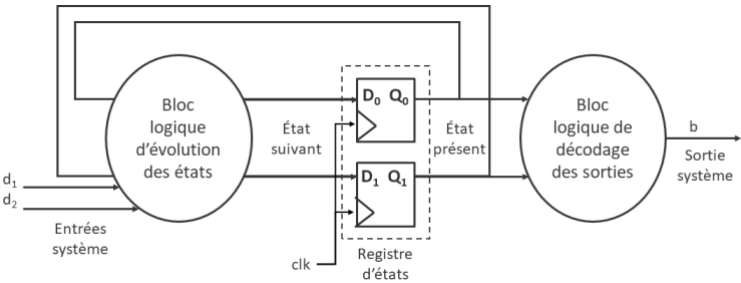


Table de vérité du bloc logique d'évolution des états

entrées				sorties			
Entrées systèmes		État présent			État suivant		
d ₁	d ₂	Q ₁	Q ₀	État prés.	État suiv.	D ₁	D ₀
0	X	0	0	0	0	0	0
1	X	0	0	0	1	0	1
X	0	0	1	1	1	0	1
X	1	0	1	1	2	1	0
X	1	1	0	2	2	1	0
X	0	1	0	2	0	0	0
X	X	1	1	3	0*	0	0

*Intérêt à renvoyer vers un cycle du système, sinon reste planté. Mais cela dépend des applications et cahiers des charges

Table de Karnaugh pour D1

Q ₁ Q ₀	00	01	11	10
d ₁ d ₂				
00	0	0	0	0
01	0	1	0	1
11	0	1	0	1
10	0	0	0	0

Table de Karnaugh pour D2

Q ₁ Q ₀	00	01	11	10
d ₁ d ₂				
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	1	1	0	0

35

35

Les tables de Karnaugh ne permettent pas de grande simplification dans ce cas.

$$D_1 = d_2 \cdot (Q_1 \oplus Q_0)$$

$$D_0 = d_1 \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{d_2} \cdot \overline{Q_1} \cdot Q_0$$

Table de vérité du bloc logique de décodage des sorties

État présent (entrée fonction)			sortie
Q ₁	Q ₀	état	b
0	0	0	0
0	1	1	1
1	0	2	1
1	1	3	X

→ $b = Q_1 + Q_0$

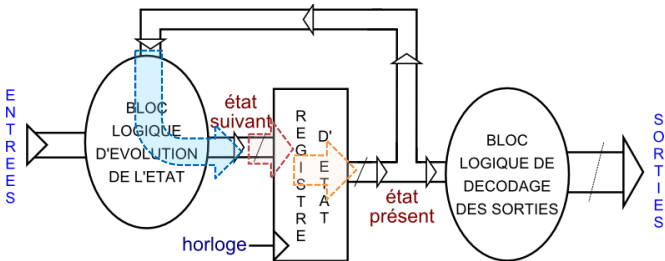
36

36

Remarque : fréquence de fonctionnement, cadencement de l'horloge



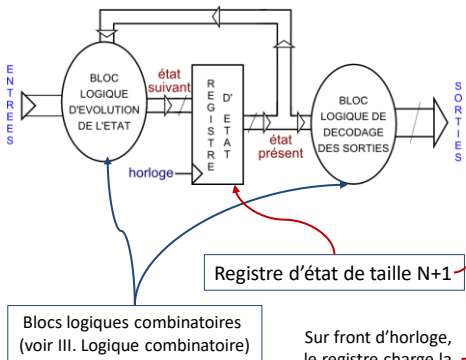
$$F_{\max} = \frac{1}{T_{\text{horloge}_{\min}}} = \frac{1}{t_{p_{\text{bascule}}} + t_{p_{\text{bloc logique}}} + t_{\text{set-up}}}$$



37

37

Description d'une machine à états finis de type Moore en VHDL



```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;

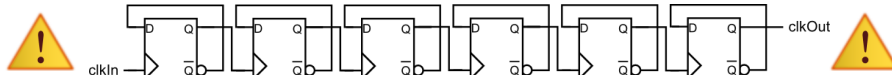
entity RegistreEtat is
Port ( D : in STD_LOGIC_VECTOR(N downto 0);
      clk : in STD_LOGIC;
      Q : out STD_LOGIC_VECTOR(N downto 0));
end RegistreEtat;

architecture Behavioral of RegistreEtat is
signal Q_sig : STD_LOGIC_VECTOR(N downto 0);
begin
process (clk)
begin
if rising_edge(clk) then
Q_sig<=D;
end if;
end process;
Q<=Q_sig;
end Behavioral;
```

38

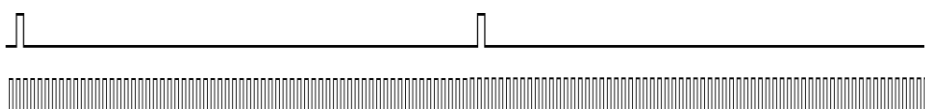
38

Structure à éviter



Les signaux de sortie des bascules de cette structure ne sont pas synchrones. On ne peut pas s'en servir pour générer des signaux à cause des aléas (états transitoires non souhaités).

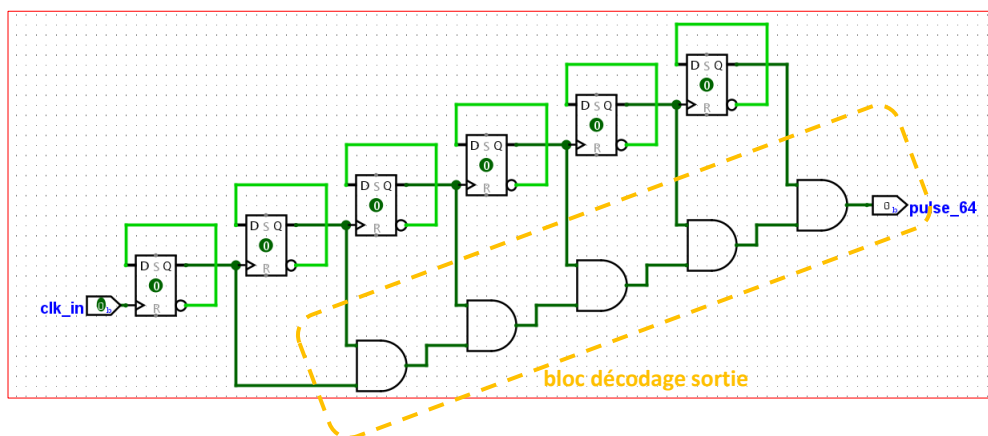
Exemple : on souhaite générer une impulsion toutes les 64 périodes (fin de comptage (cf Timer 0 du PIC32)).



39

39

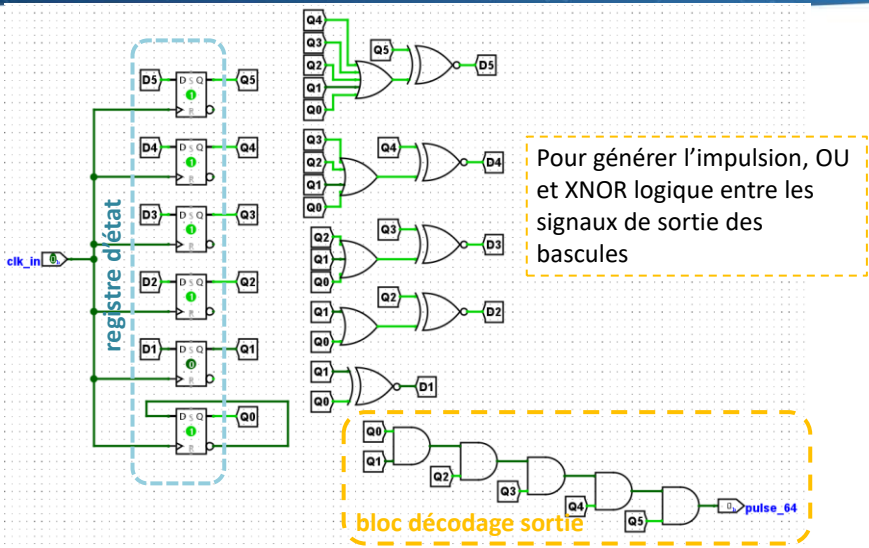
structure à éviter (asynchrone)



40

40

Structure
(synchrone)

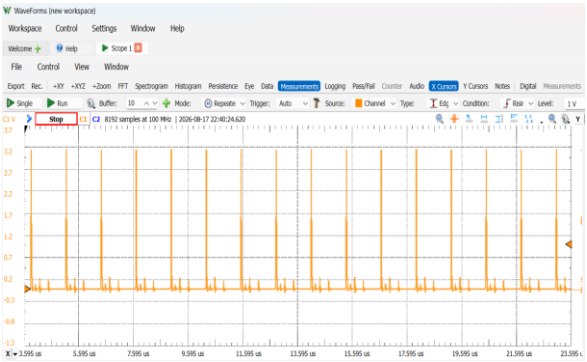


41

41



structure (synchrone)

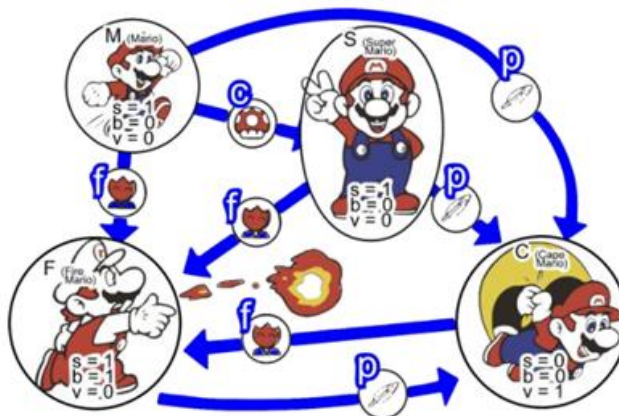


structure à éviter
(asynchrone)

42

42

Diagramme d'état : notion plus large que l'électronique numérique, adaptée aux systèmes séquentiels, à la programmation.



43

43

Dans les CPU/MCU:

Compteurs/décompteurs à préchargement

- Compteur programme
- Timer (signal de fin de comptage/décomptage, IT)

Séquenceur

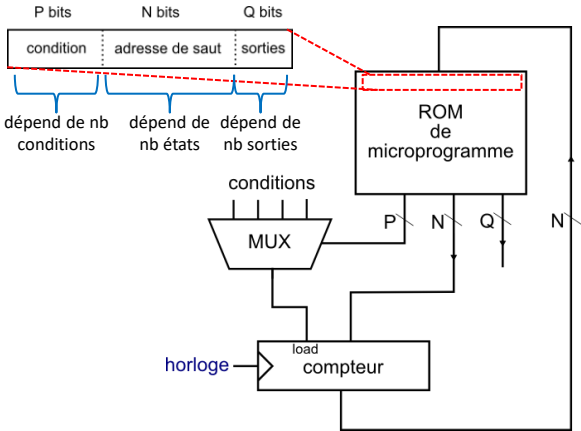
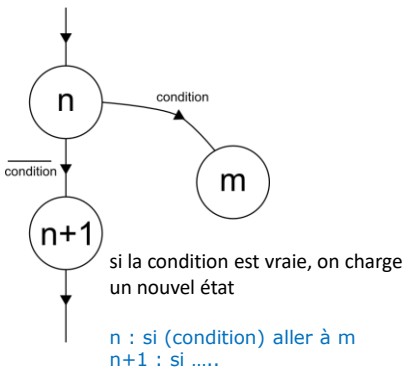
- Les machines à états finis sont une des structures pour réaliser des **séquenceurs (automates) dans les cœurs de calculateur**.
- L'autre structure est le séquenceur micro-programmé.

44

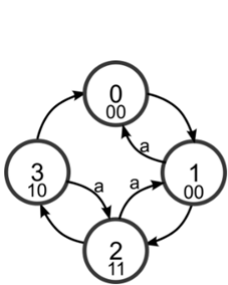
44

4. Séquenceur micro-programmé

Possibilités d'évolution restreintes, moins flexible, moins grande efficacité
Branchement conditionnel, incrémentation implicite



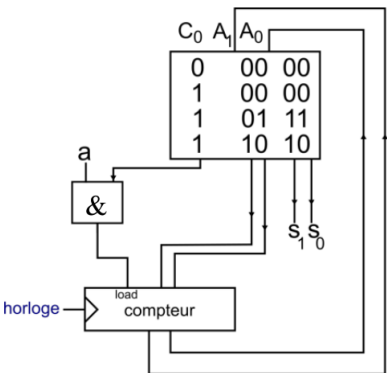
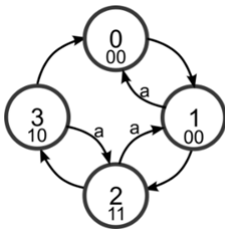
Exemple simple :



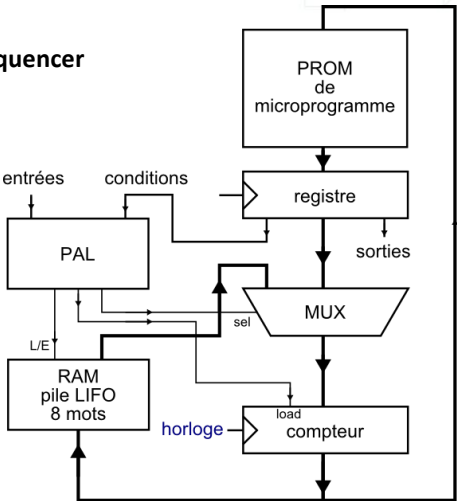
1 bit	2 bits	2 bits
cond.	adresse de saut	sorties

état	instruction	sorties	micro-instruction		
		S_1S_0	C_0	A_1A_0	S_1S_0
0	-----	00	0	XX	00
1	Si a aller en 0	00	1	00	00
2	Si a aller en 1	11	1	01	11
3	Si a aller en 2	10	1	11	10

Exemple : (cont.)



PROSE : PROgrammable SEquencer



Séquenceur

Le séquenceur coordonne les sous-opérations nécessaires pour l'exécution des instructions.

Solution câblée :

Elle est l'automate du séquenceur est une machine à états finis câblée.

privilégiée pour les processeurs RISC (*Reduced Instruction Set Core/Computer*); exemple PIC

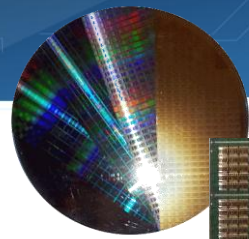
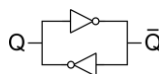
Solution micro-programmée :

Elle est l'automate est un séquenceur micro-programmé (microprogramme contenu dans une ROM) privilégiée pour les processeurs CISC (*Complex Instruction Set Computer*)

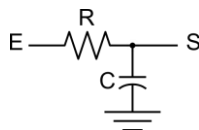
49

49

Point mémoire statique (bistable)

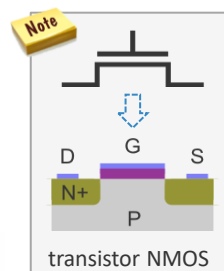
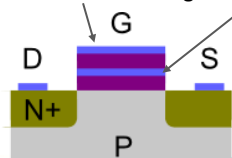


Point mémoire dynamique



Mémoire flash

grille de commande grille flottante

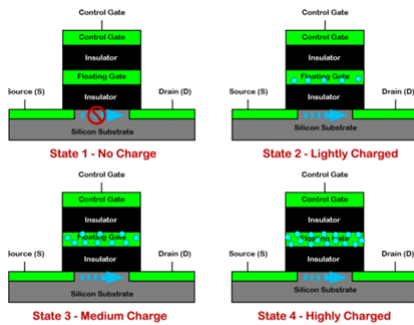


50

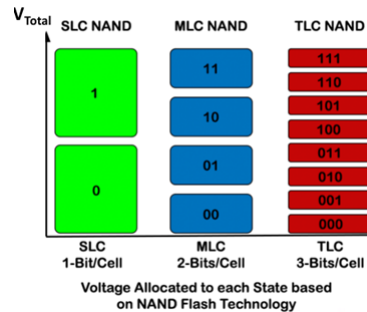
50

Inventée par Toshiba en 1980

Mémoire flash



État de la grille flottante	État logique
Chargée (programmée)	« 0 » - bas
Non chargée (effacée)	« 1 » - haut



MLC (Multi Level Cell) TLC (Tri Level Cell)

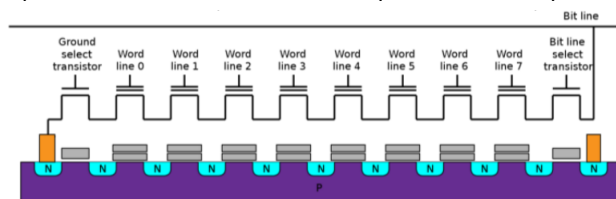
51

51

Mémoire flash

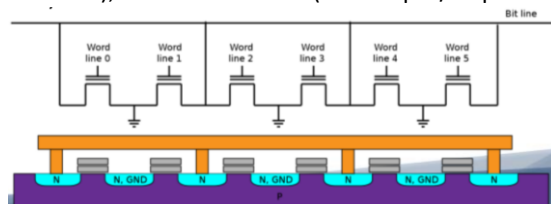
Flash NAND : écriture, lecture, effacement par blocs. Dans les clefs USB, disques durs, smartphones.

Transistors en série. Bit line à «0» quand toutes les Word lines à «1»



Flash NOR : écriture, lecture, effacement par mots. Stockage du code exécutable dans les microcontrôleurs, mémoire de démarrage (Boot ROM), firmware et BIOS (Basic Input/Output System).

«1» sur Word line programme un «0»

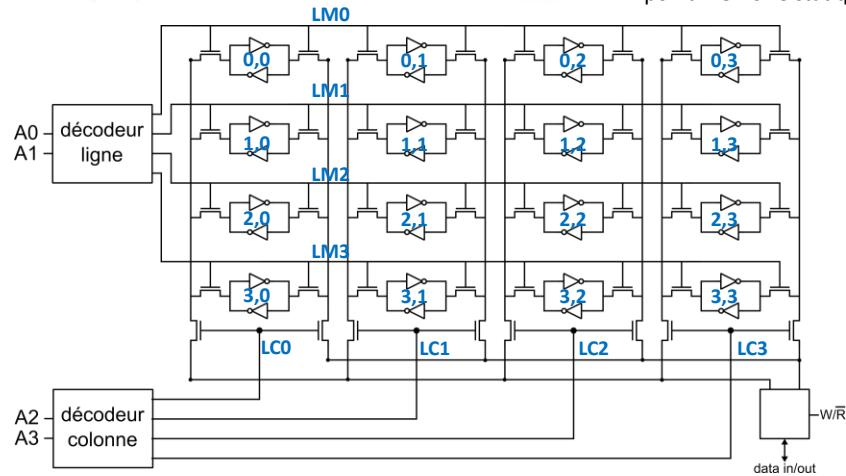


52

52



point mémoire statique



16 mots de 1 bit

53

53



Quelle différence essentielle existe-t-il entre un système asynchrone et un système synchrone ?

Quels sont les avantages/inconvénients respectifs des systèmes asynchrones et synchrones ?

De quel type sont la grande majorité des microprocesseurs et microcontrôleurs (synchrone ou asynchrone) ?

54

54



Quels sont les deux types de machine à états finis ?

Quelles sont les étapes de la synthèse d'une machine à états finis ?

Quels sont les avantages/inconvénients relatifs des solutions machine à états finis et séquenceur microprogrammé pour la synthèse de circuits logiques séquentiels ?