LICENCE EEA

4 – Machines à Etats

Systemes Numeriques & Processeurs Embarques

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Julien DENOULET
julien.denoulet@sorbonne-universite.fr

SU – LIP6
L3 EEA – LU3EE100

Plan

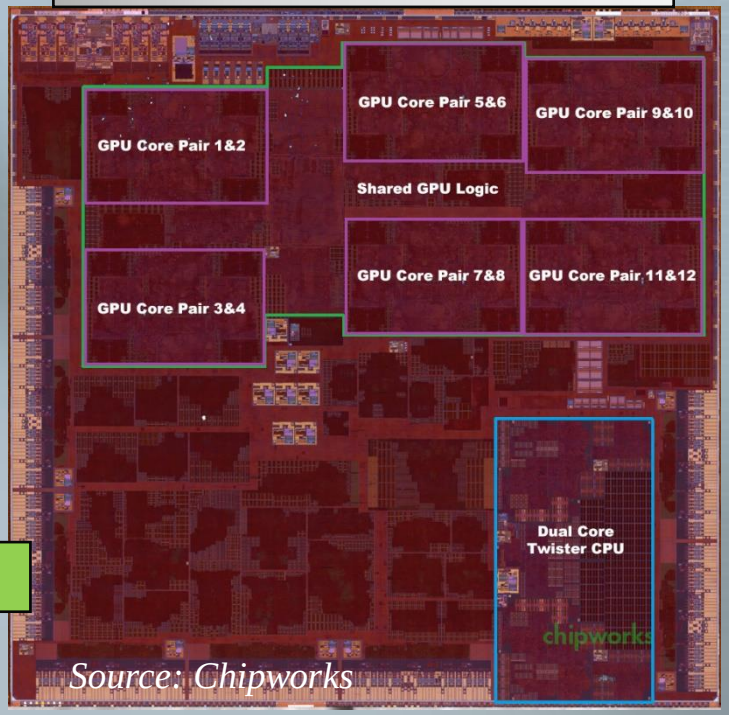
- Graphes d'état
- Transposition – Machine de Moore
- Codage des états
  - Aléatoire
  - Adjacent
  - One Hot
  - Total Synchrone
- Machine de Mealy
- Description VHDL

Previously on LU3EE100…

iPad

Source: Apple

- 2 processeurs ARM
- Mémoire
- Périphériques (video, E/S)



Source: Chipworks

Processeur A9X

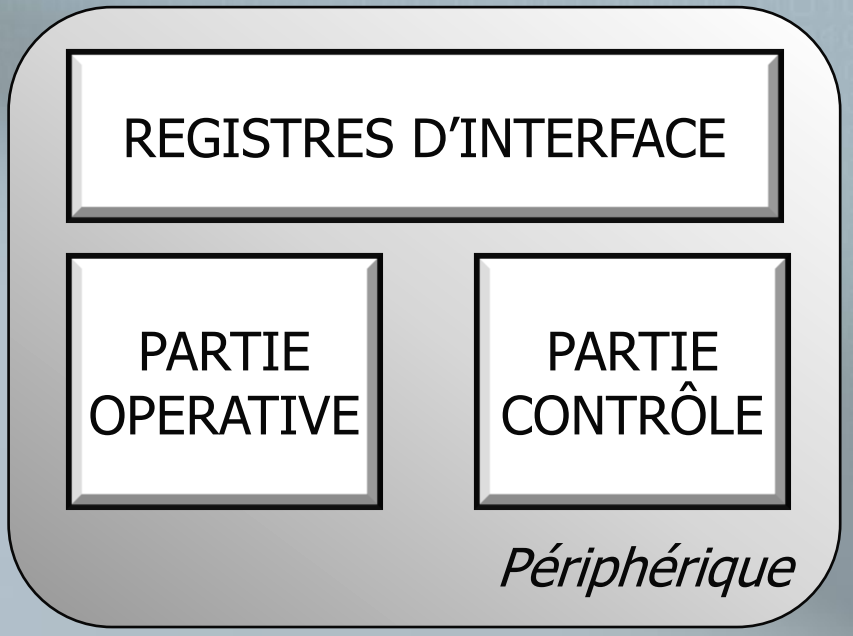
C4

3

Périphérique Matériel

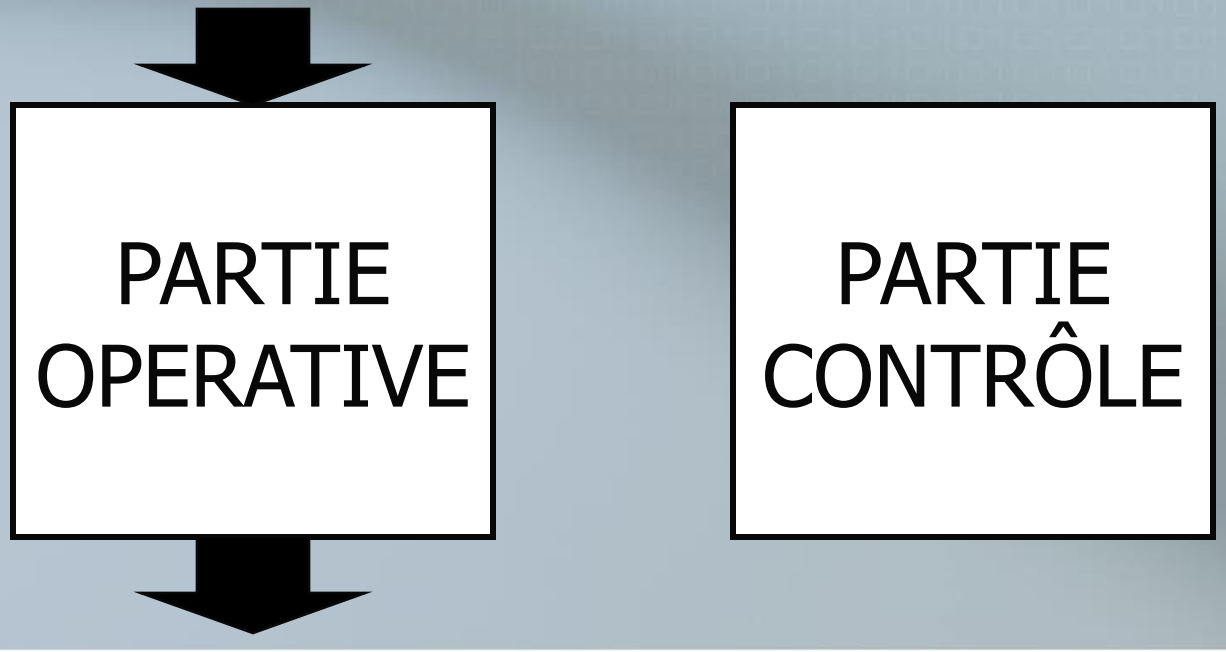
- Composant qui aide le processeur à réaliser une tâche spécifique

- Architecture d'un périphérique
  - Registres d'interface avec le processeur *(voir plus tard…)*
  - Partie opérative
  - Partie contrôle



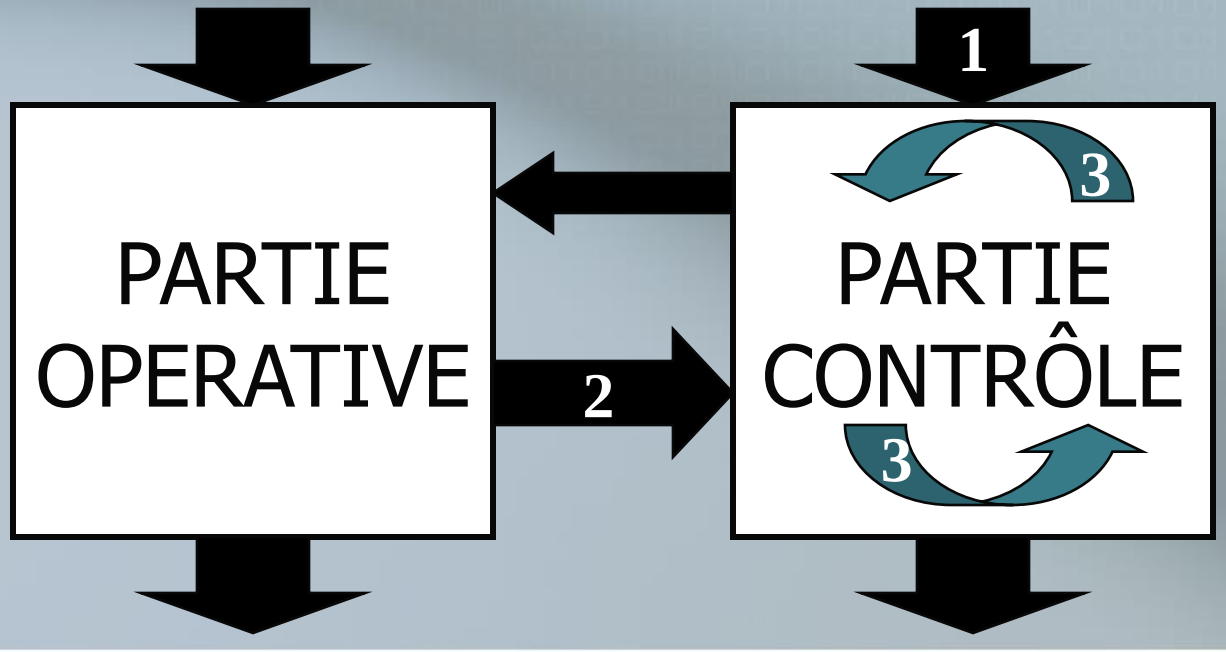
Partie Opérative / Contrôle

- Système Numérique Complexe
  - Partition des Ressources
    - Calcul et Mémorisation (PARTIE OPÉRATIVE)
    - Contrôle (PARTIE CONTRÔLE)



Partie Opérative / Contrôle

- P.Ctrl commande les actions de la P.Opé
  - En fonction
    1) Des entrées du système
    2) Des résultats engendrés par la partie opérative
    3) De la propre évolution de la partie contrôle



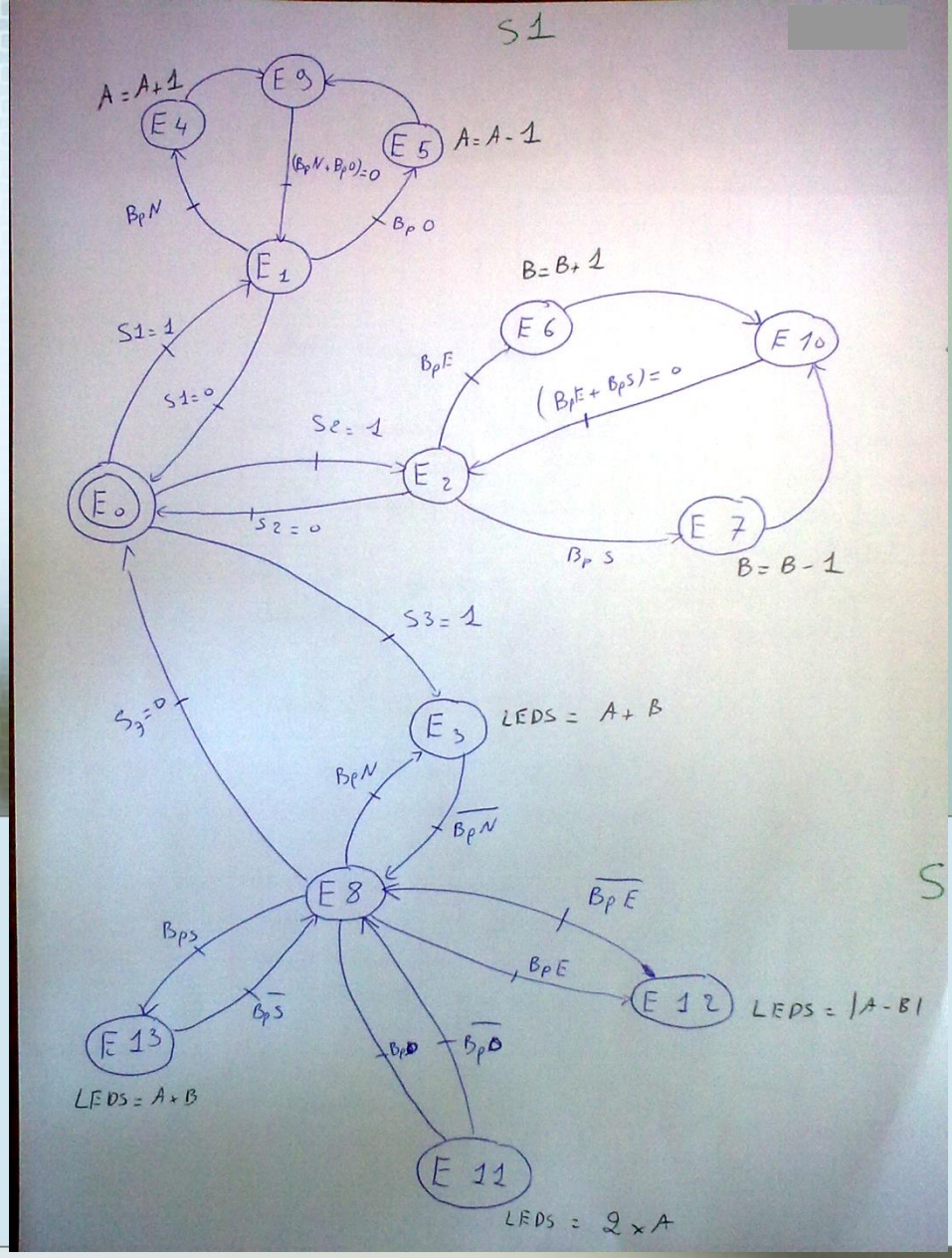
Partie Contrôle

- Déroulement des états successifs du système
  - État: donne la valeur des sorties à un instant donné
- Implémenté sous la forme d'une machine à états
  - Idem "automate« , "séquenceur« , "machine à états finis"
  - Traduction électronique d'une représentation graphique de l'évolution du système
    - GRAPHE D'ÉTATS

C4

7

Graphes d'état



Graphe d'états

- Description du comportement du système:
  - Son état

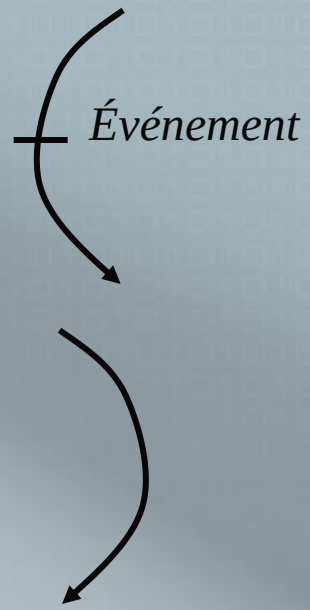
  - Pour chaque état va correspondre une série d'actions

Sortie 1 = ...
Sortie 2 = ...

  - Son évolution
    - En fonction (ou non) d'événements sur les entrées du système

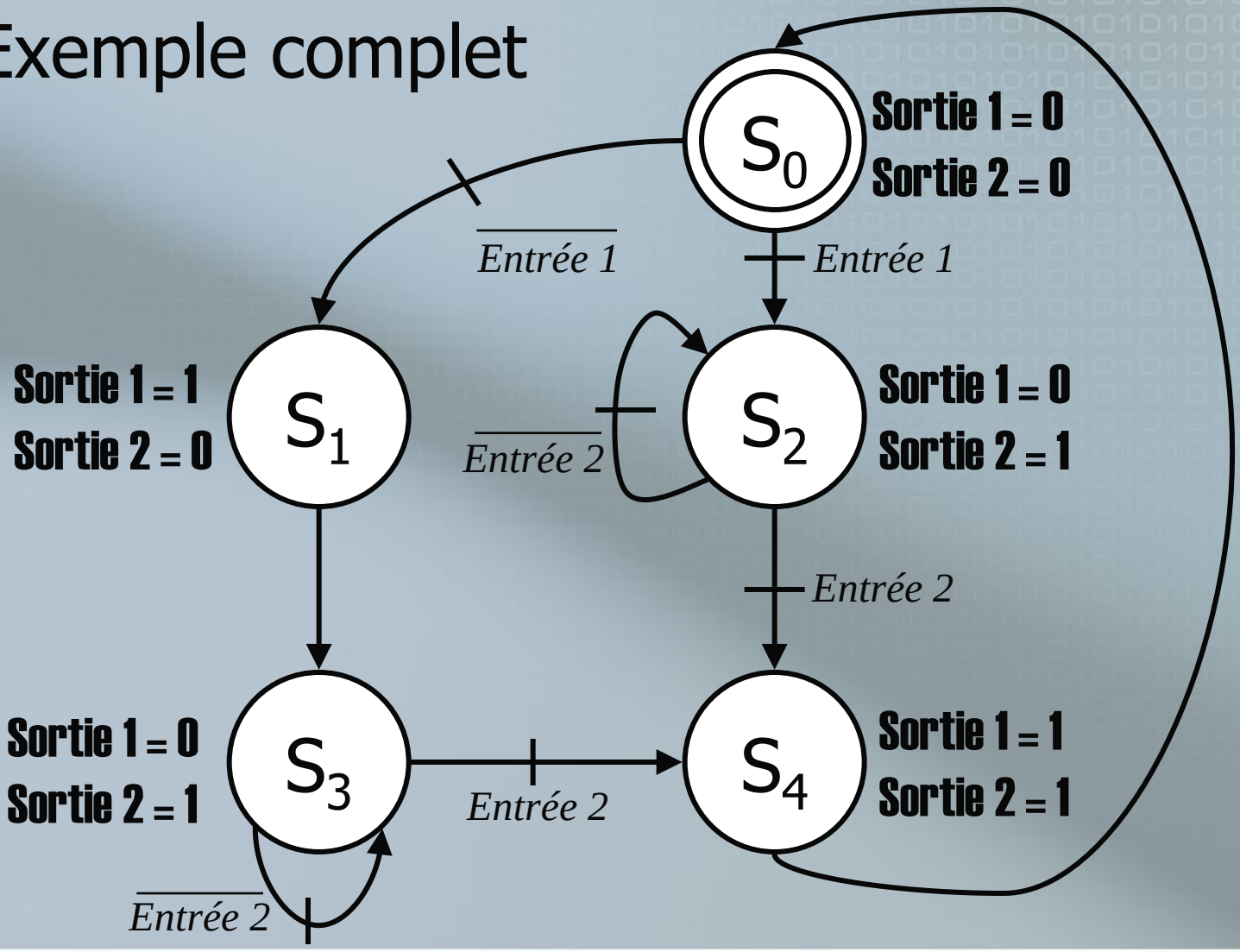
Évolution des États

- A chaque étape (cycle d'horloge), le système peut changer d'état
- Il existe deux types de transitions
  - Conditionnelle
    - La transition s'effectue si un événement survient sur l'une des entrées
  - Inconditionnelle
    - La transition s'effectue indépendamment de l'évolution des entrées



Graphe d'états

- Exemple complet

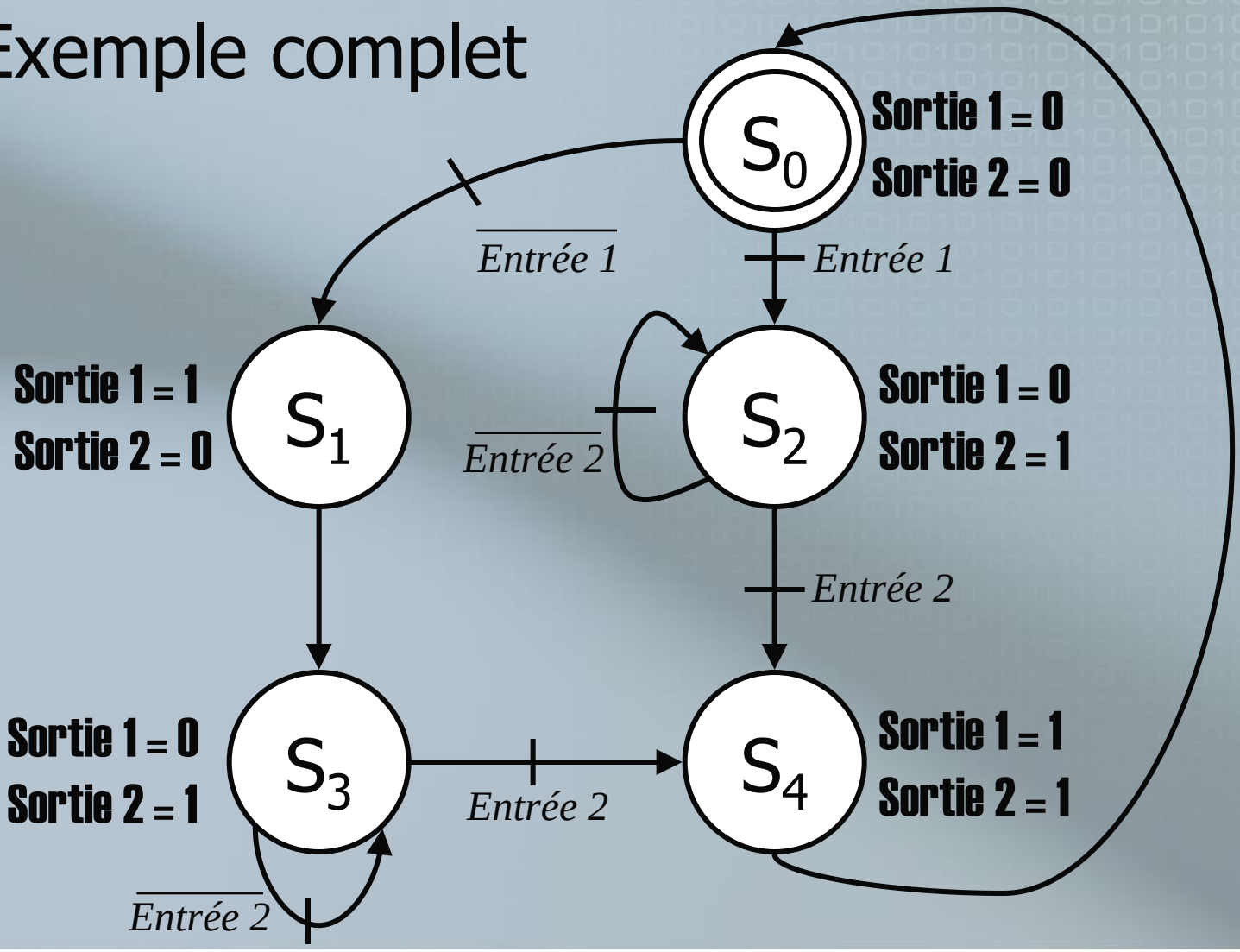


Critères de Déterminisme

- Pour être synthétisable, une machine à états doit être déterministe
- Dans le graphe d'états, à tout instant, le système doit être dans un et un seul état
- Contraintes de construction du graphe
  - Il ne peut y avoir qu'une transition partant d'un état si celle-ci est inconditionnelle
  - Si une transition conditionnée par X part d'un état, il existe une ou plusieurs autres transitions partant de ce même état et implémentant la condition $\overline{X}$
  - Les transitions conditionnelles partant d'un même état sont mutuellement exclusives entre elles

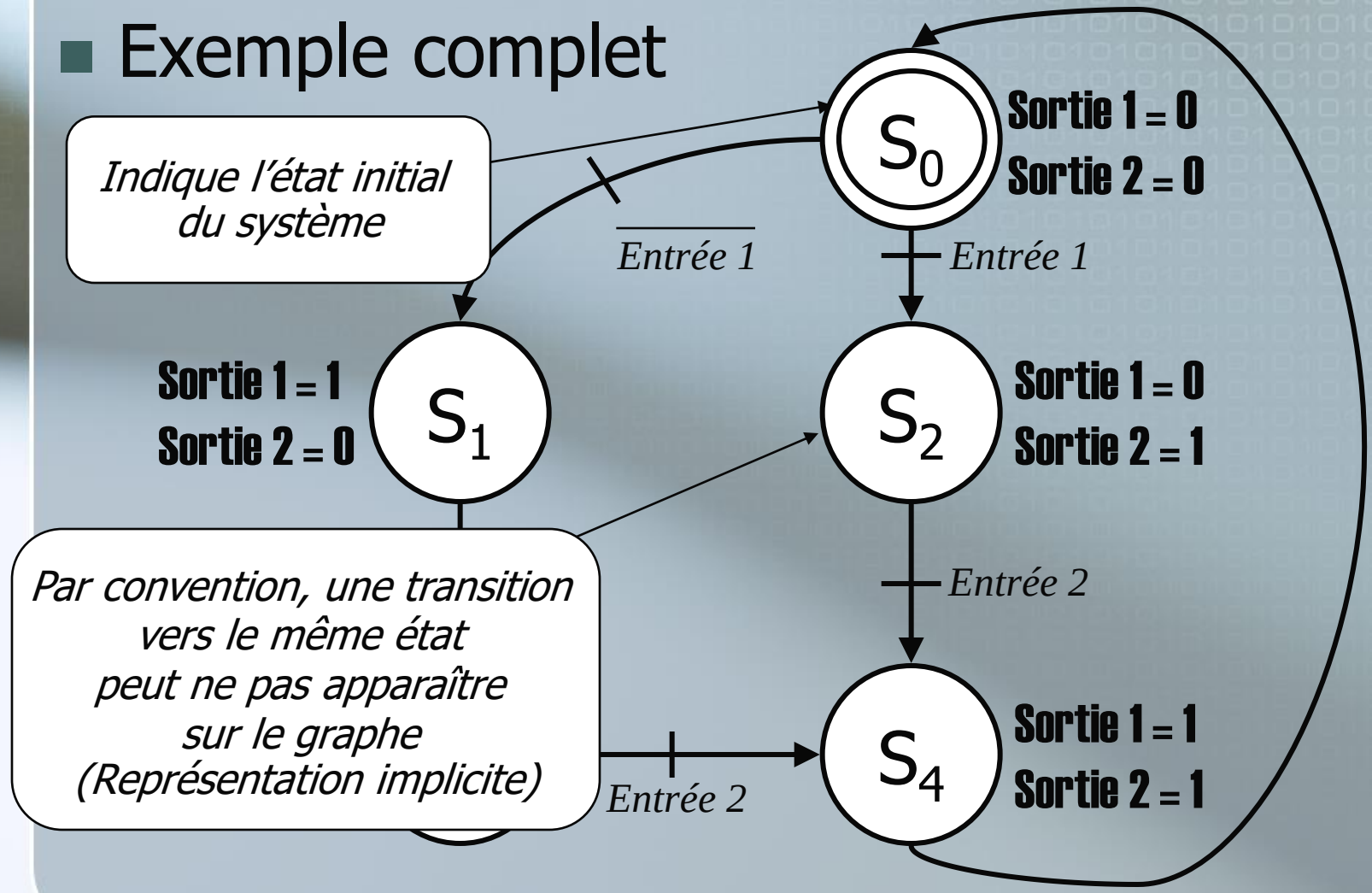
Graphe d'états

- Exemple complet



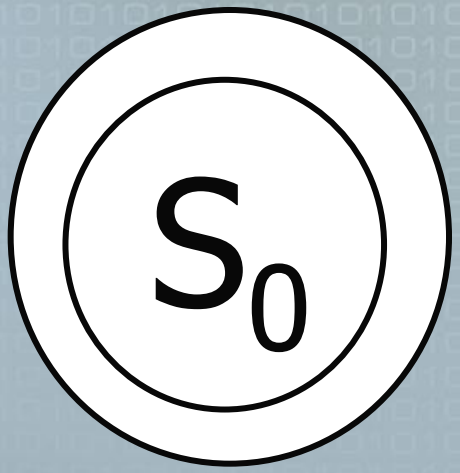
Graphe d'états

- Exemple complet



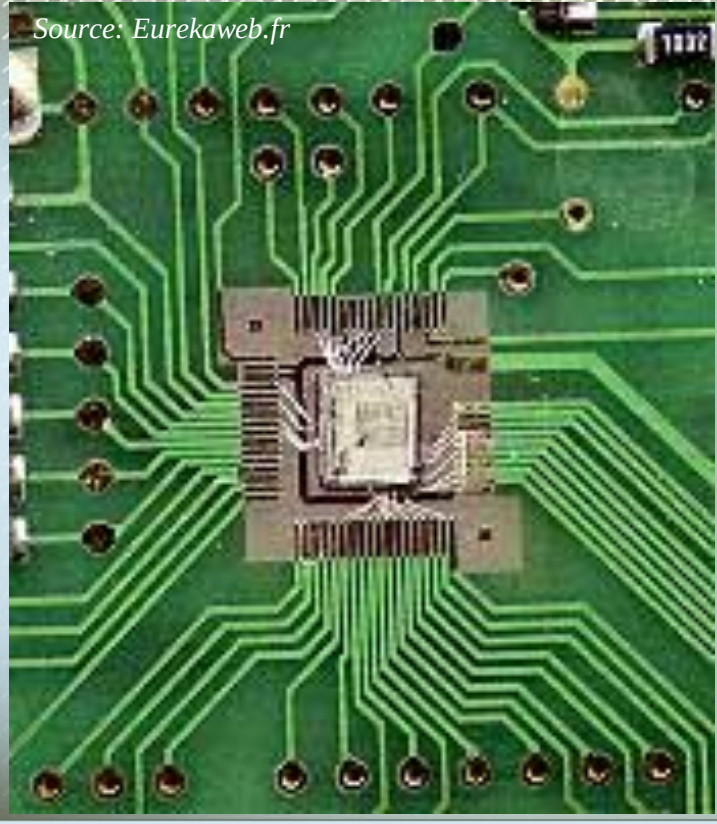
Etat d'initialisation

- Dans le graphe, l'état d'initialisation représente l'état dans lequel se retrouve le système après un RESET ASYNCHRONE



- Les transitions d'un état vers l'état initial par suite d'un RESET ASYNCHRONE ne sont pas explicitées dans le graphe

- SEULES les TRANSITIONS SYNCHRONES (sur un front d'horloge) sont représentées dans le graphe d'état



Transposition Matérielle

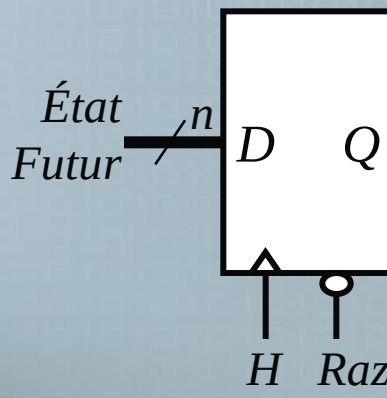
Du graphe aux portes logiques…

Réalisation Machine à États

- Transposition du graphe d'états en un système électronique
- Méthode de réalisation

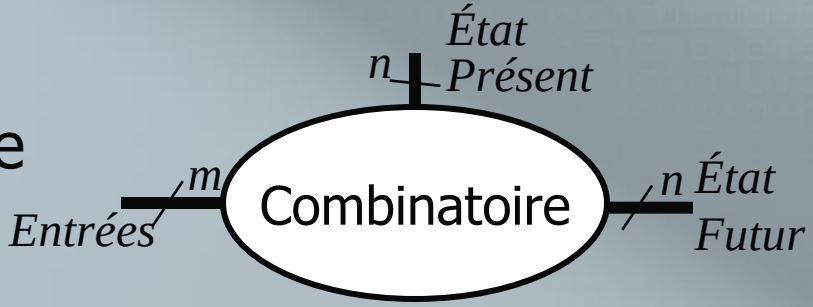
Transposition Électronique

- Éléments nécessaires
  - État du système
    - Registre d'états



  - Actions du système

  - Évolution du système

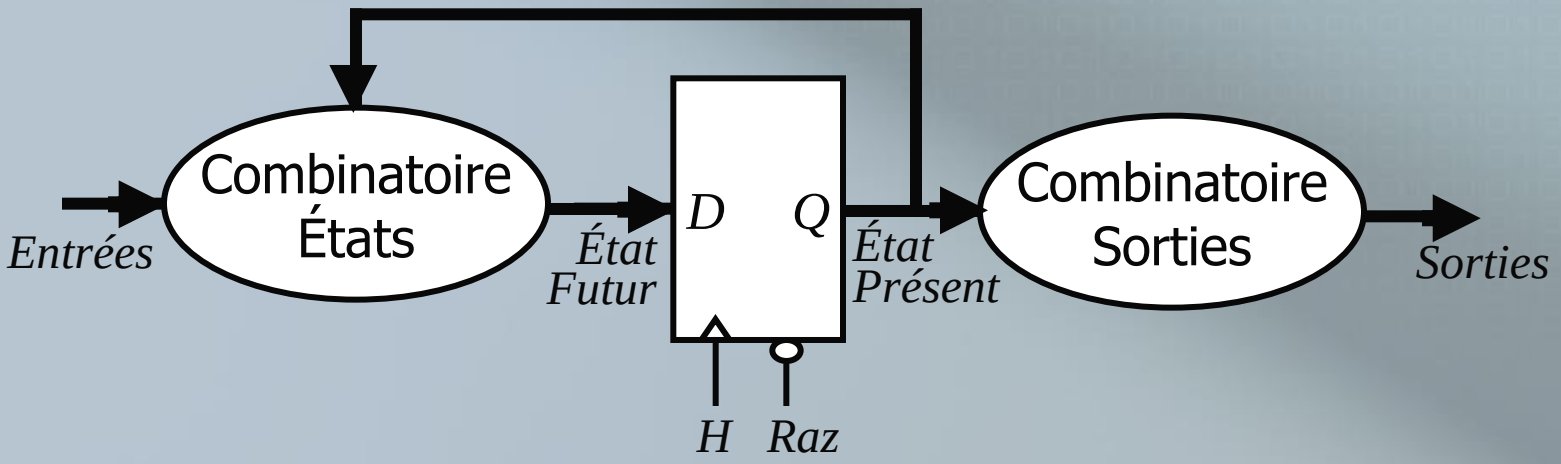


C4

18

Machine de Moore

- Horloge & RAZ sont des entrées implicites du système
  - Elles n'apparaissent pas sur le graphe d'états
- Un RAZ asynchrone ramène à l'état initial
- En dehors du RAZ, les changements d'états se font sur un front d'horloge



Réalisation Machine d'états

1) Spécifications du cahier des charges (*CRUCIAL*)
2) Détermination des états (nombre, fonction)
3) Identification des entrées/sorties de la MAE
4) Établissement du graphe d'états.

5) Détermination de l'encodage des états
6) Calcul des équations de l'état futur et des sorties
   (A l'aide éventuellement d'une table de transition des états)
7) Réalisation – Synthèse logique

Réalisation Machine d'états

1)
2)
3)
4)

Etapes 5 à 7
Semi-automatiques
(outils de synthèse)

5)
6)
7)

Etapes 1 à 4
à effectuer manuellement

C4

21

Source: CDVI

Codage des états

Table de Transition *(optionnel)*

Etat Pres.	E1	E2	Etat Fut.	Sortie 1	Sortie 2
S0	0	0	S1	0	0
S0	0	1	S1	0	0
S0	1	0	S2	0	0
S0	1	1	S2	0	0
S1	0	0	S3	1	0
S1	0	1	S3	1	0
S1	1	0	S3	1	0
S1	1	1	S3	1	0
S2	0	0	S2	0	1
S2	0	1	S4	0	1
S2	1	0	S2	0	1
S2	1	1	S4	0	1

Etat Pres.	E1	E2	Etat Fut.	Sortie 1	Sortie 2
S3	0	0	S3	0	1
S3	0	1	S4	0	1
S3	1	0	S3	0	1
S3	1	1	S4	0	1
S4	0	0	S0	1	1
S4	0	1	S0	1	1
S4	1	0	S0	1	1
S4	1	1	S0	1	1

Table de transition du graphe du slide 11

- Les transitions s'effectuent sur des fronts d'horloge
- Une machine d'état est un système SYNCHRONE

Table de Transition *(optionnel)*

Etat Pres.	E1	E2	Etat Fut.	Sortie 1	Sortie 2
S0	0	0	S1	0	0
S0	0	1	S1	0	0
S0	1	0	S2	0	0
S0	1	1	S2	0	0
S1	0	0	S3	1	0
S1	0	1	S3	1	0
S1	1	0	S3	1	0
S1	1	1	S3	1	0
S2	0	0	S2	0	1
S2	0	1	S4	0	1
S2	1	0	S2	0	1
S2	1	1	S4	0	1

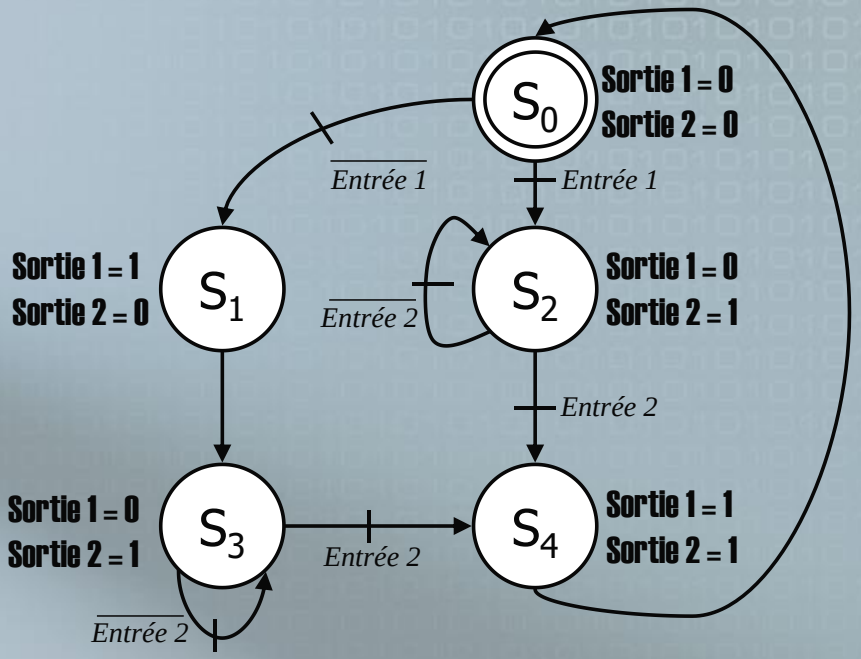


Table de transition du graphe du slide 11

- Les transitions s'effectuent sur des fronts d'horloge
- Une machine d'état est un système SYNCHRONE

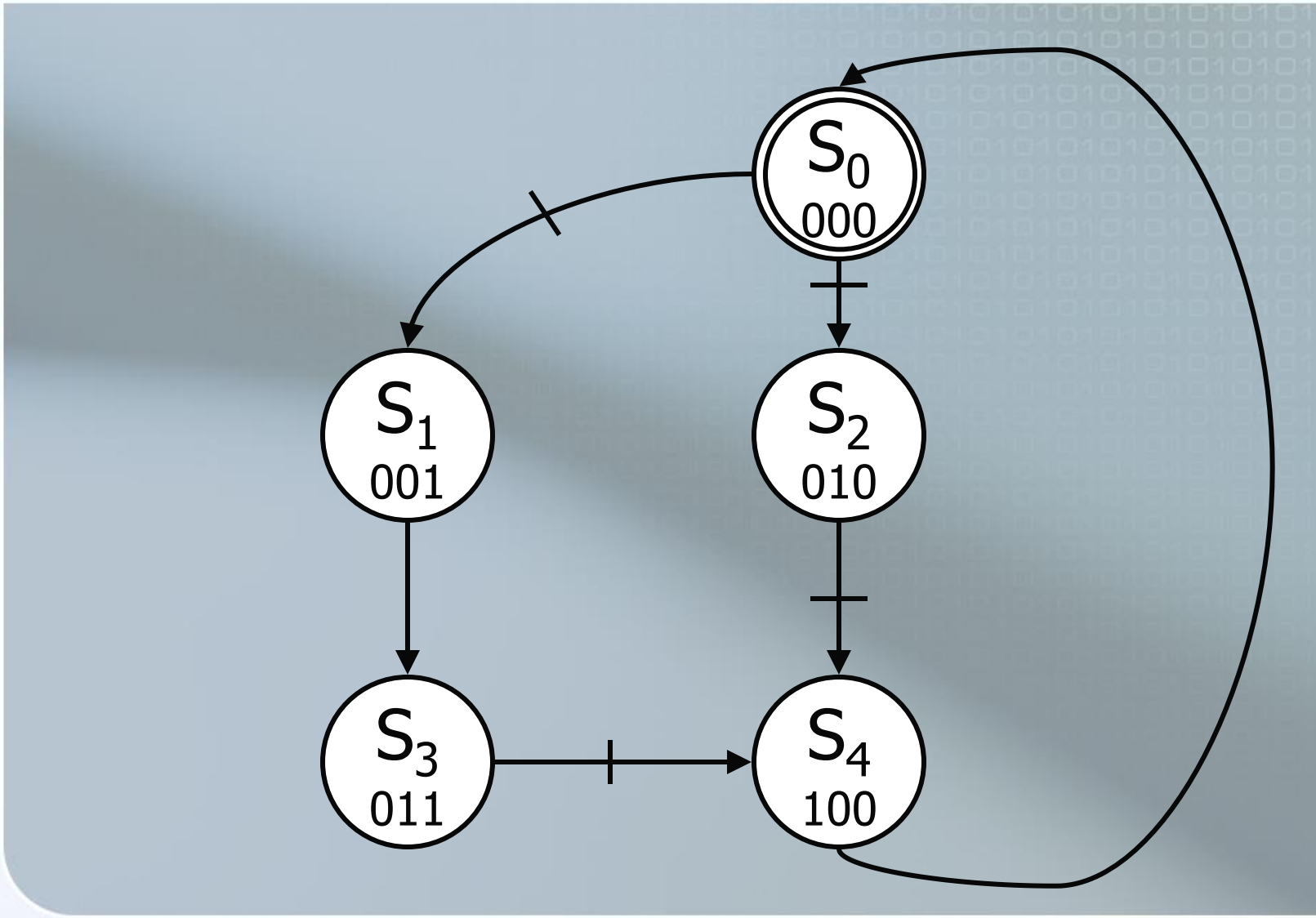
Codage des états

- L'état du système est représenté par la valeur stockée dans le registre d'états
  - Détermination d'un code pour chaque état
  - La longueur du code détermine la taille du registre d'états

Codage des états

- Nombre de bascules minimales nécessaires
  - Si 2^n états dans le graphe
  - n bascules dans le registre d'états
  - Choix d'un code binaire pour chaque état
- Codage aléatoire
  - Le code de chaque état est choisi arbitrairement par le concepteur
- Codage adjacent
  - Un seul bit peut être modifié pour chaque transition
  - Cet encodage n'est pas toujours possible selon les graphes

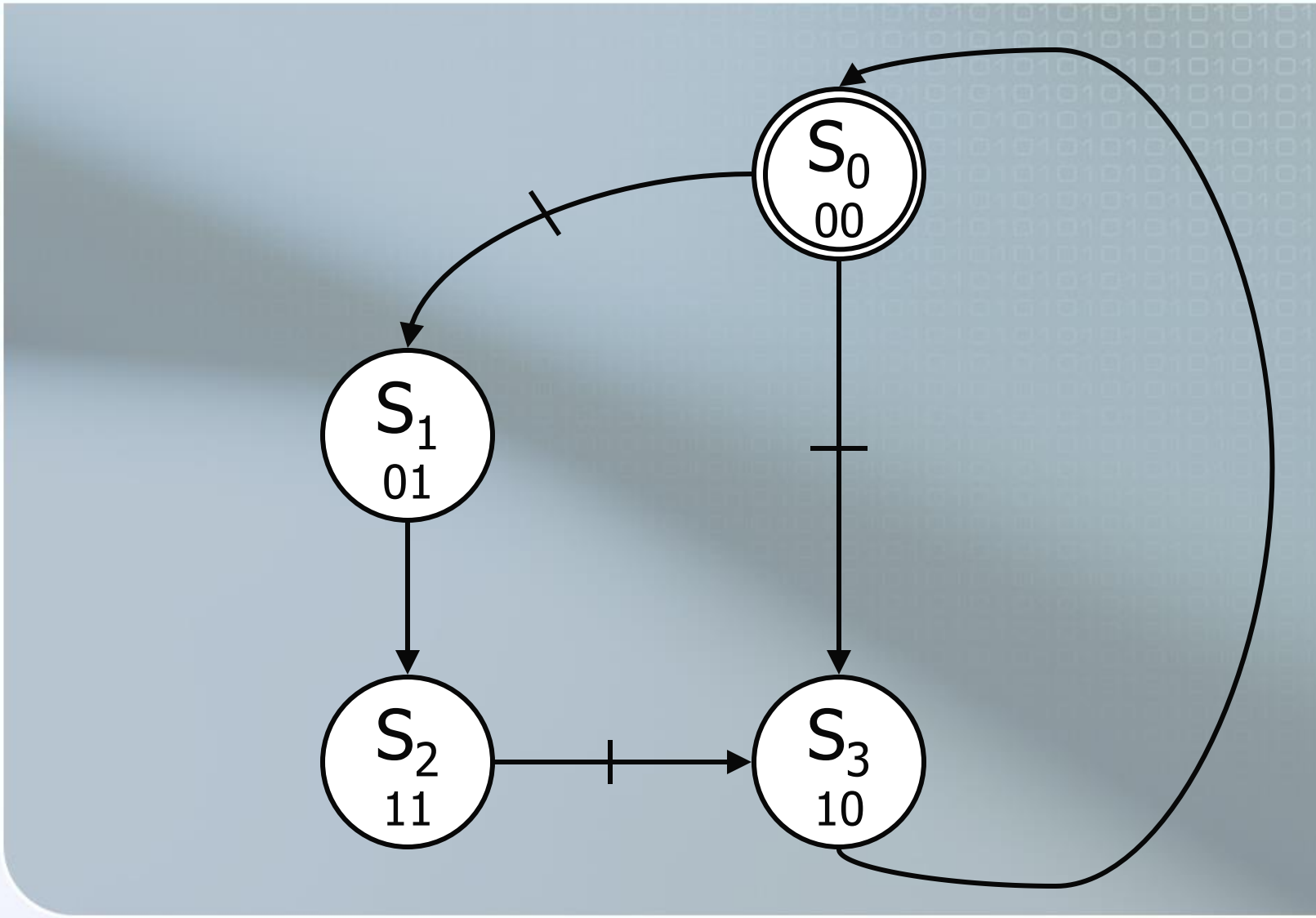
Exemple Codage Aléatoire



C4

27

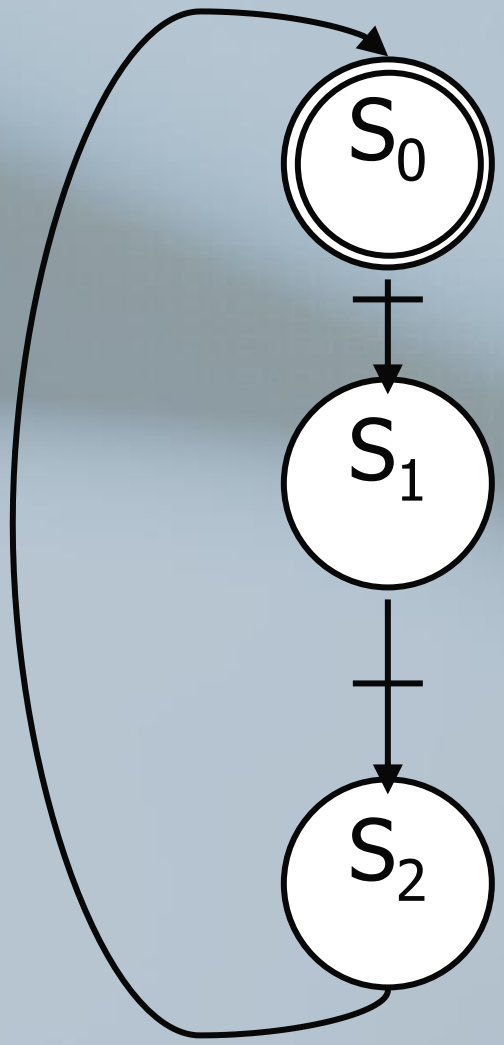
Exemple Codage Adjacent



C4

28

Exemple Codage Adjacent



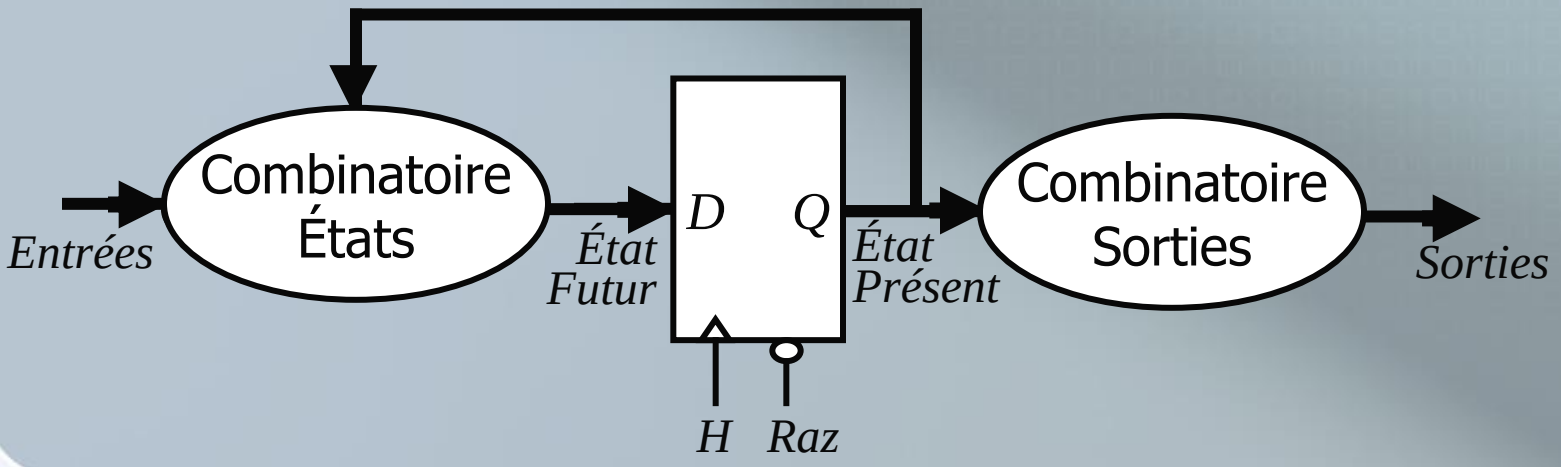
- Impossible sur ce graphe

C4

29

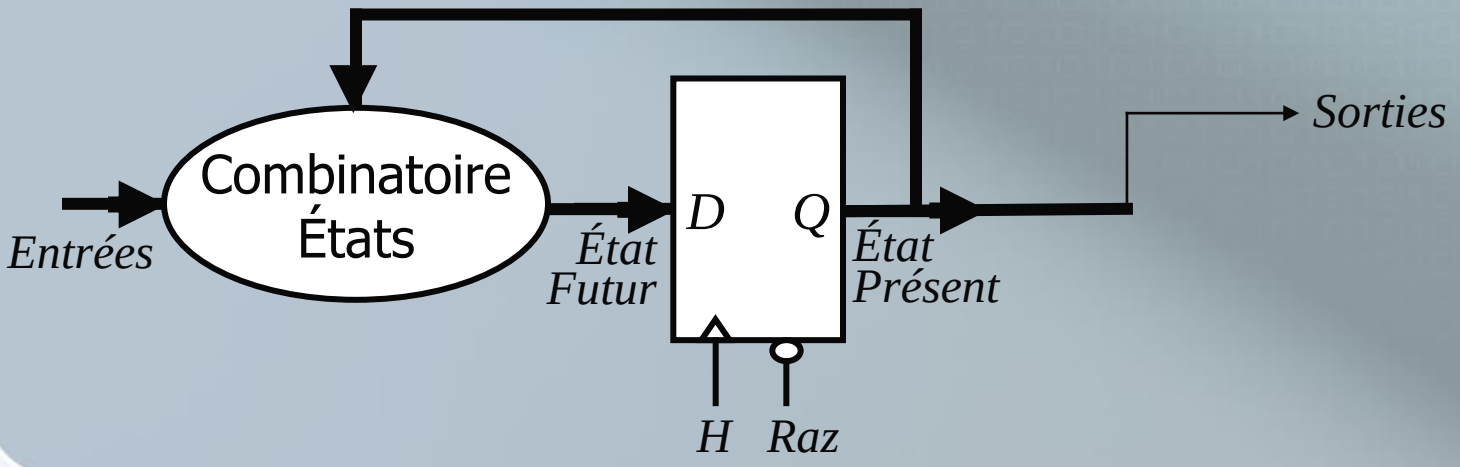
Autres codages

- Principe: La taille du code n'est plus forcément optimale
  - Augmentation de la taille du registre d'états
- Cela va permettre de simplifier les fonctions combinatoires



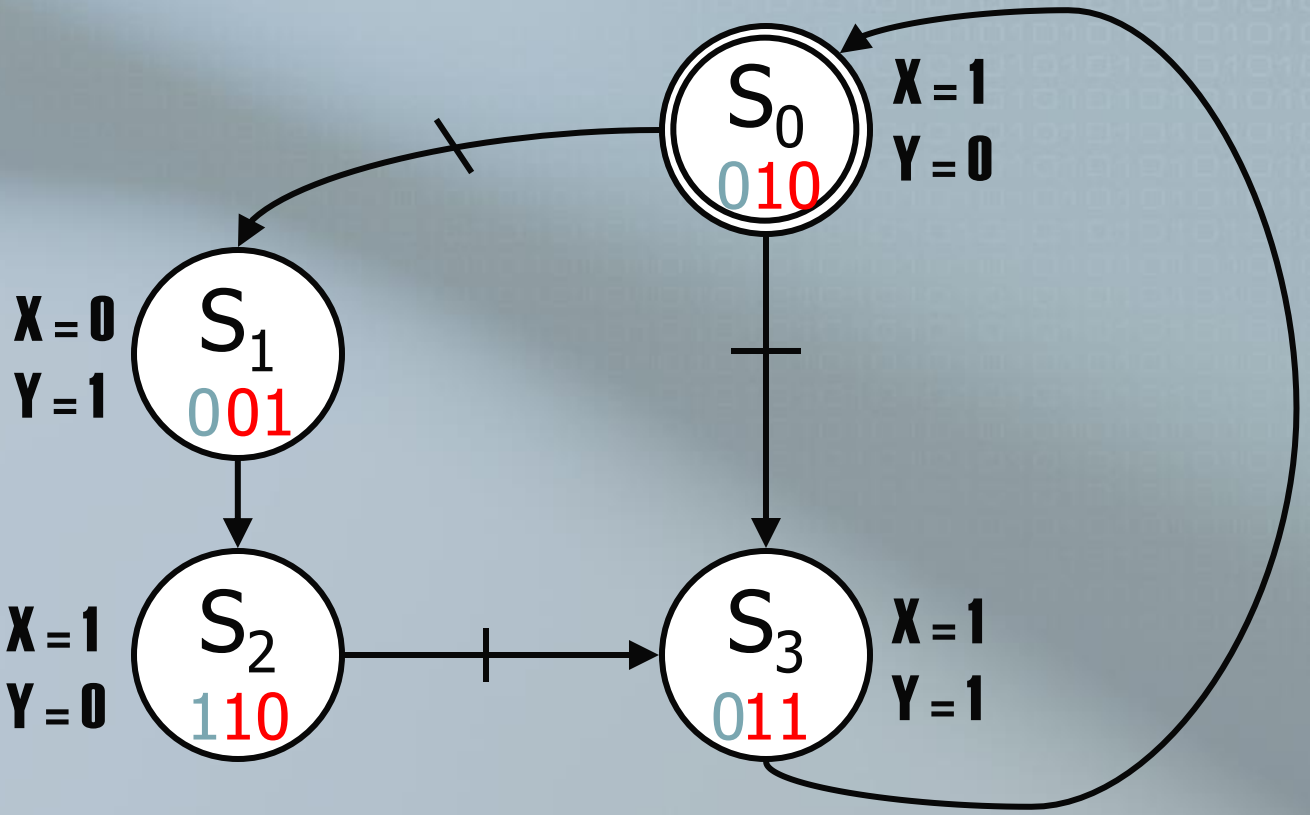
Autres codages

- Codage total synchrone
  - Principe: Coder les états de façon à éliminer le combinatoire de sortie
  - Peut nécessiter plus de bascules pour le registre



Codage Total Synchrone

- Méthode
  - Recopier l'état des sorties dans le codage des états
  - Compléter éventuellement avec d'autres bits



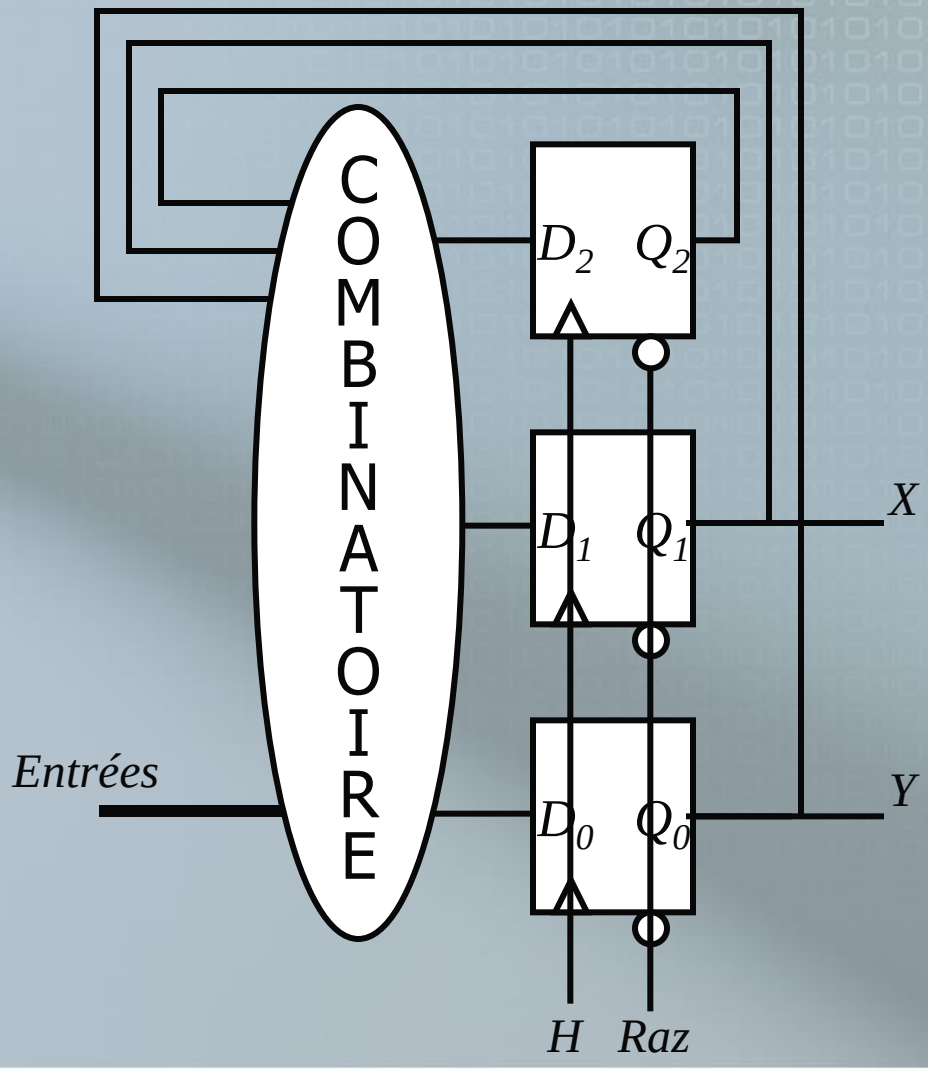
C4

32

Codage Total Synchrone

Etat Pres.	Q_2	Q_1	Q_0	X	Y
S0	0	1	0	1	0
S1	0	0	1	0	1
S2	1	1	0	1	0
S3	0	1	1	1	1

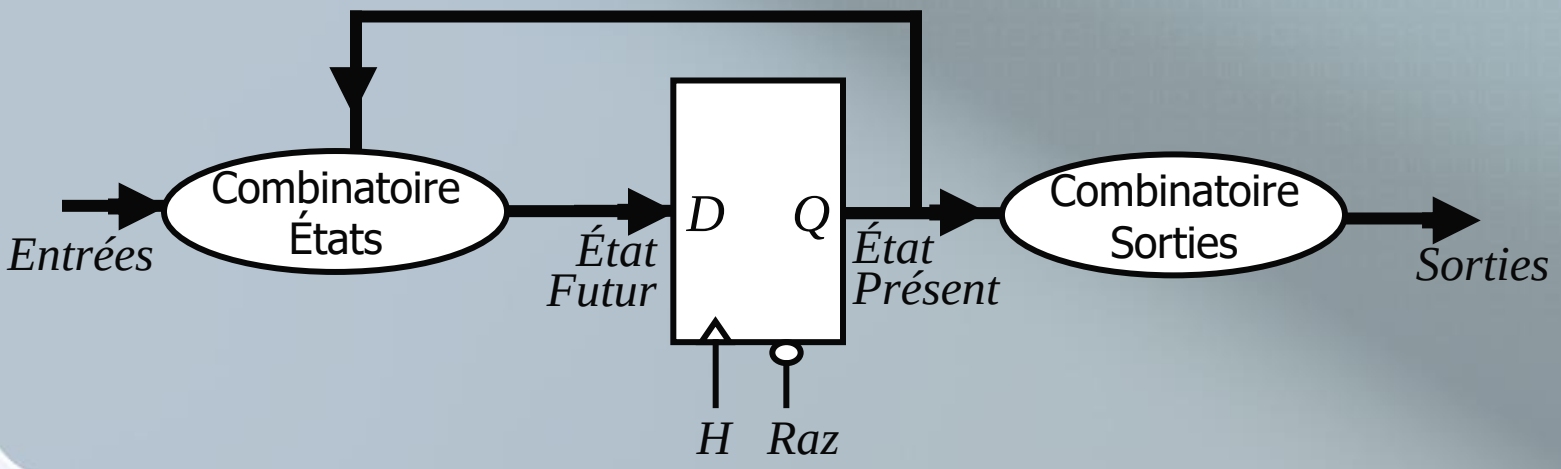
$$X = Q_1$$
$$Y = Q_0$$



Autres codages

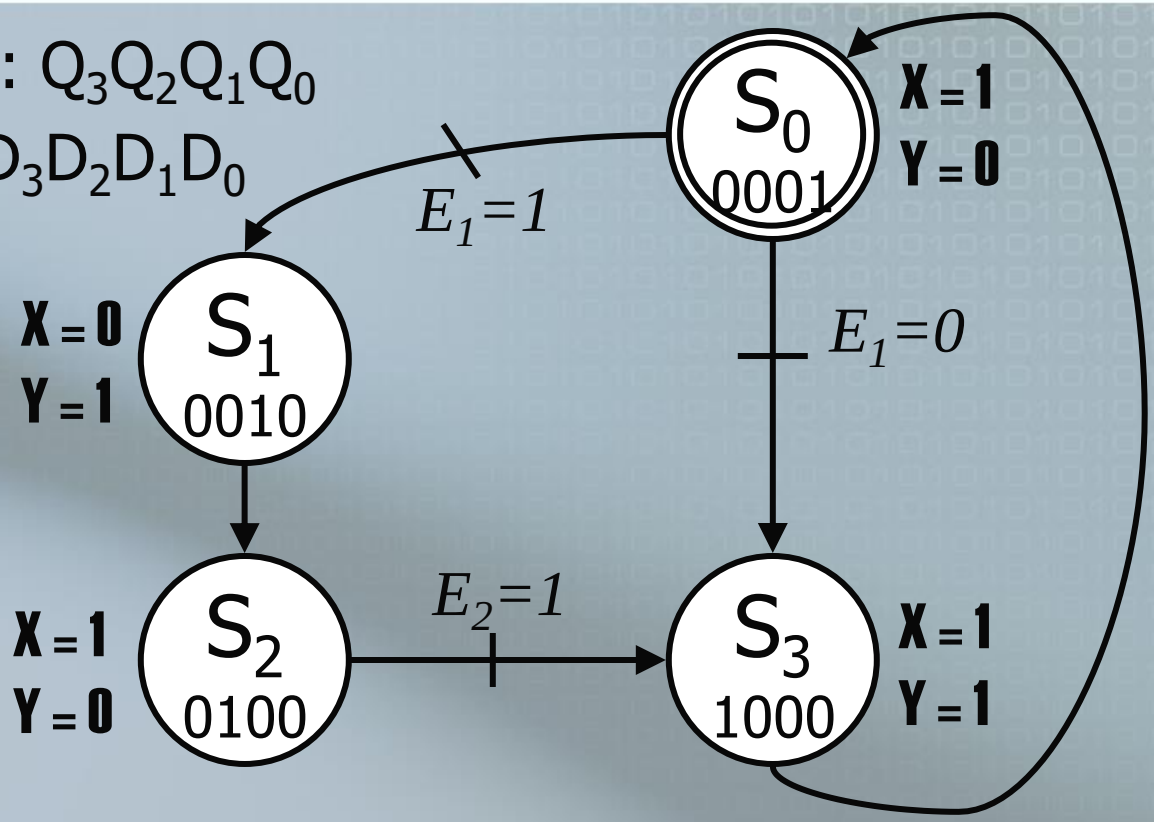
- Codage One-Hot
  - Principe: 1 bascule par état

    Pour chaque état, le code a 1 seul bit à 1
    Si Qi = 1 alors la machine est à l'état i
  - Permet de simplifier les combinatoires



Codage One-Hot

- Etat Présent: $Q_3Q_2Q_1Q_0$
- Etat Futur: $D_3D_2D_1D_0$



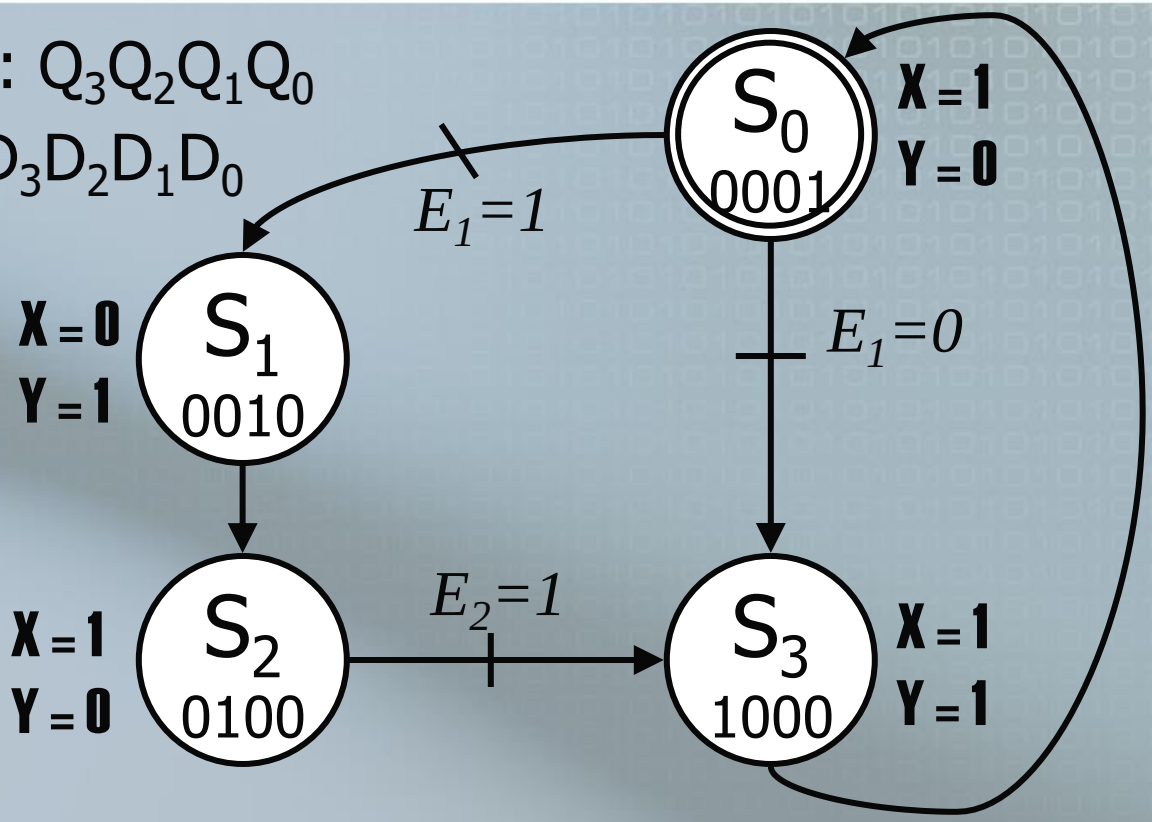
- Etat Prséent = S0 → Q0=1
- Etat Futur = S2 → D2 = 1

C4

35

Codage One-Hot

- Etat Présent: $Q_3Q_2Q_1Q_0$
- Etat Futur: $D_3D_2D_1D_0$



$$D_0 = Q_3 \qquad D_2 = Q_1 + Q_2.\overline{E_2} \qquad X = Q_0 + Q_2 + Q_3$$
$$D_1 = Q_0.E_1 \qquad D_3 = Q_0.\overline{E_1} + Q_2.E_2 \qquad Y = Q_1 + Q_3$$

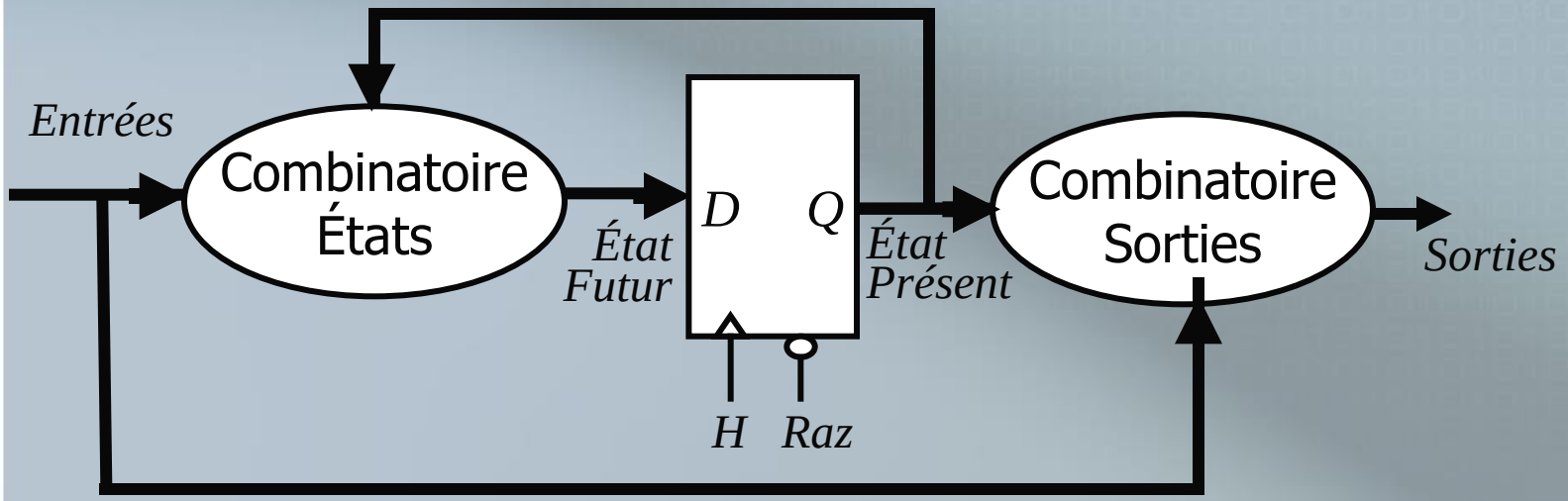
C4

36

Machines de Mealy

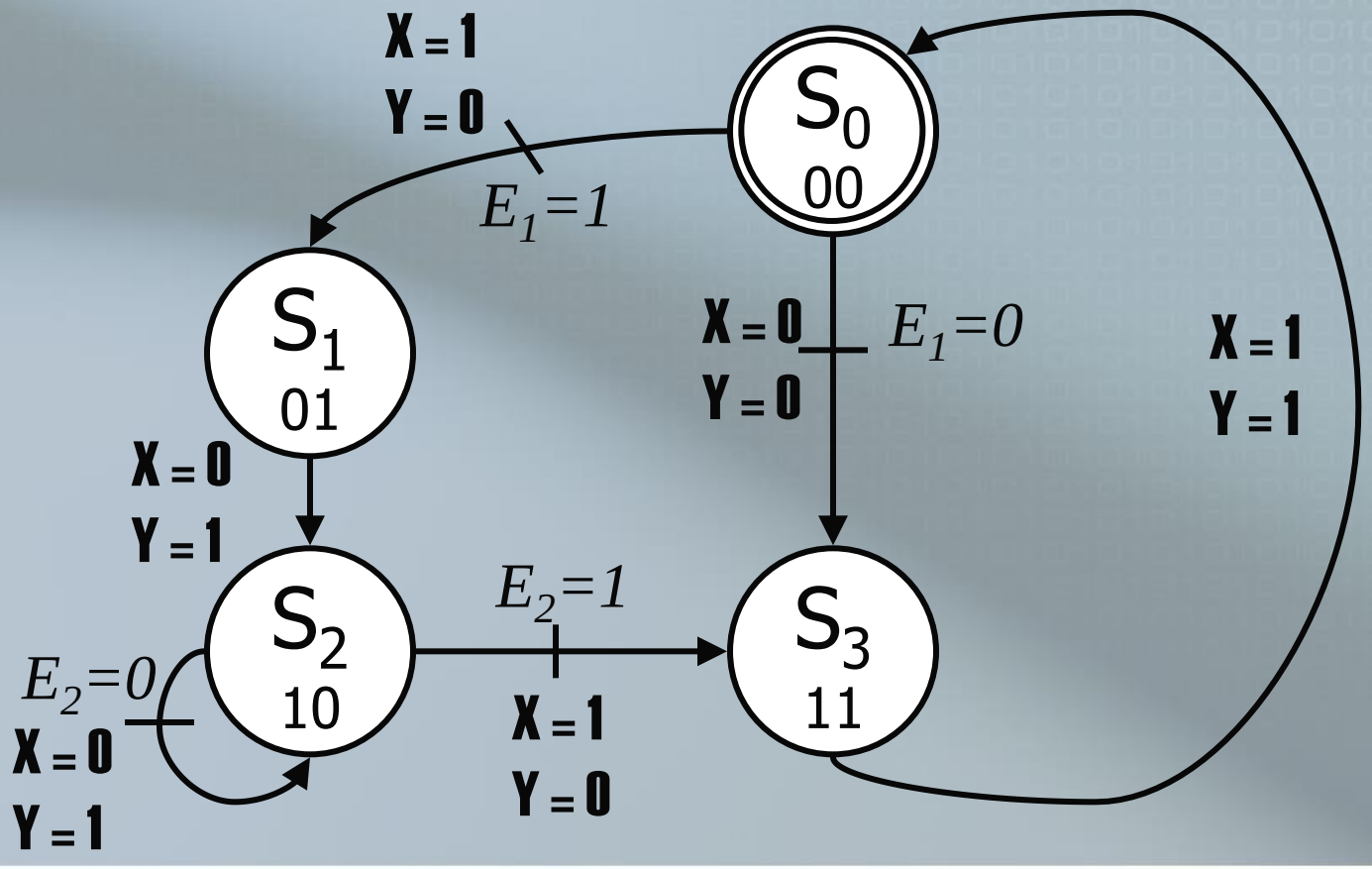
Machines de Mealy

- Structure similaire à Moore
- Différence: Le calcul des sorties dépend
  - De l'état présent
  - Des entrées



Machines de Mealy

- Représentation sur le graphe d'états
  - État des sorties sur les arcs



Machines de Mealy

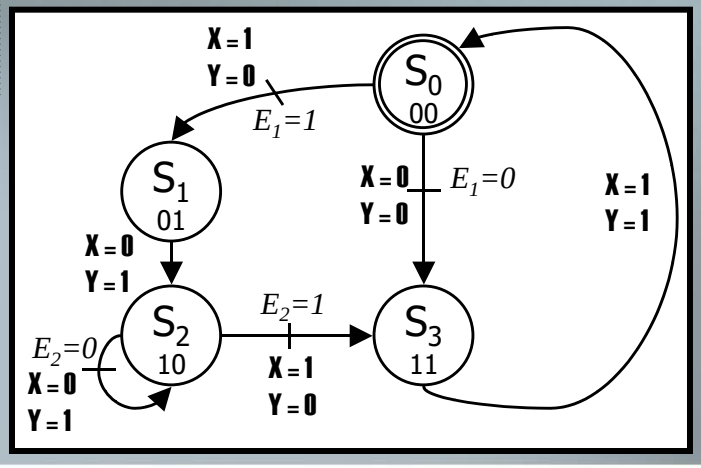
- Équations de l'état futur

$$D_0 = \overline{Q_0}.\overline{Q_1} + \overline{Q_0}.E_2 \qquad Q_1 = Q_1 \oplus Q_0 + \overline{Q_1}.\overline{E_1}$$

- Équations des sorties

$$X = E_2.Q_1 + Q_0.Q_1 + E_1.\overline{Q_0}.\overline{Q_1}$$

$$Y = Q_0 + Q_1.\overline{E_2}$$

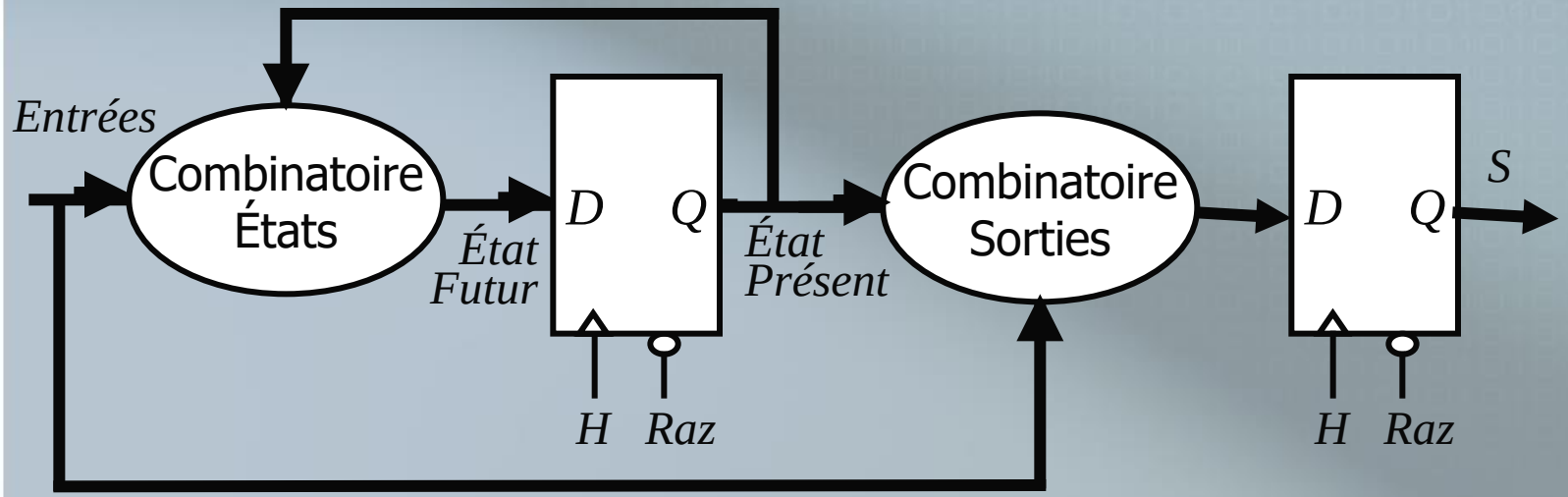


C4

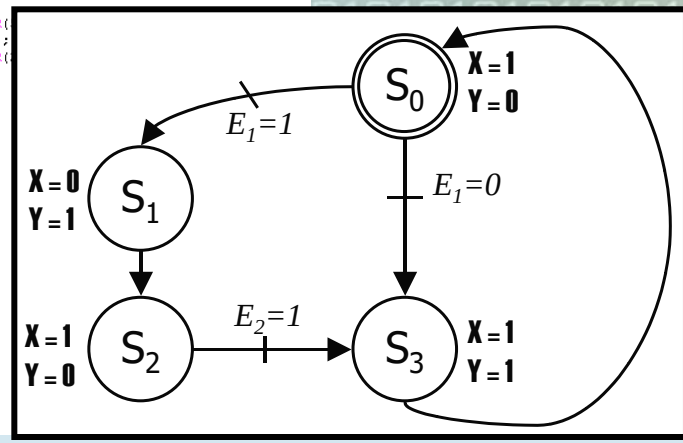
40

Machine de Mealy Synchrone

- Inconvénient Mealy:
  - Valeurs transitoires des sorties
  - Synchronisation des différents signaux de sortie
- Solution: Ajout d'un registre de sortie



```
1  ----------------------------------------------------------------------------------
2  -- Company: Young Embedded Systems LLC
3  -- Engineer: Gene Breniman
4  -- Module Name:    ARM_SEQ_RAM- - Behavioral
5  -- Revisions:
6  --    0.01 - 07/02/2007 File Created
7  -- Additional Comments:
8  --
9  ----------------------------------------------------------------------------------
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.STD_LOGIC_ARITH.ALL;
13 use IEEE.STD_LOGIC_UNSIGNED.ALL;
14
15 entity ClkDiv is
16     Port (  InByte : in STD_LOGIC_VECTOR(3 downto 0);       --<-- Seq_CPLD
17             RegSel : in STD_LOGIC_VECTOR(1 downto 0);       --<-- Seq_CPLD
18             RegStrb : in STD_LOGIC;                         --<-- Seq_CPLD
19             MClk : in  STD_LOGIC;                           --<-- OSC
20             SeqReset : in STD_LOGIC;                        --<-- Power Monitor
21             ADC_Clk : out STD_LOGIC);                       -->-- ADC
22 end ClkDiv;
23
24 architecture Behavioral of ClkDiv is
25    signal   ADC_div :      STD_LOGIC_VECTOR(
26    signal   ADCClk :       STD_LOGIC := '0';
27    signal   ClkSel :       STD_LOGIC_VECTOR(
28
29 begin
30
31    ClkDivP : process(Mclk,SeqReset)
32       begin
33       if SeqReset = '0' then
34          ADCClk <= '0';
35          ADC_div <= "001001";
36       elsif Mclk = '0' and Mclk'event then
37          if ADC_div = "000000" then
38             ADCClk <= not(ADCClk);
39             case ClkSel is
40                when "000" =>
41                when "001" =>
42                   ADC_div <= "000001";
43                when "010" =>
44                   ADC_div <= "000100";
45                when "011" =>
46                   ADC_div <= "001001";
47                when "100" =>
48                   ADC_div <= "001001";
49                when others =>
```

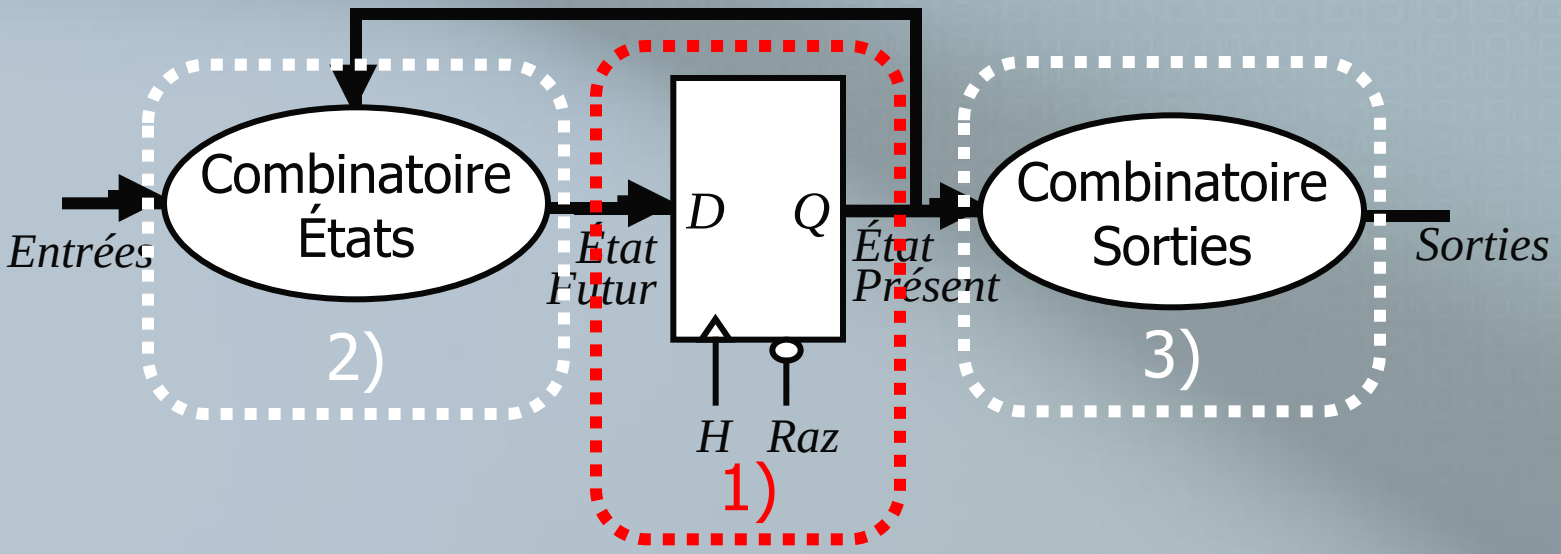


Codage VHDL

Machines à états - VHDL

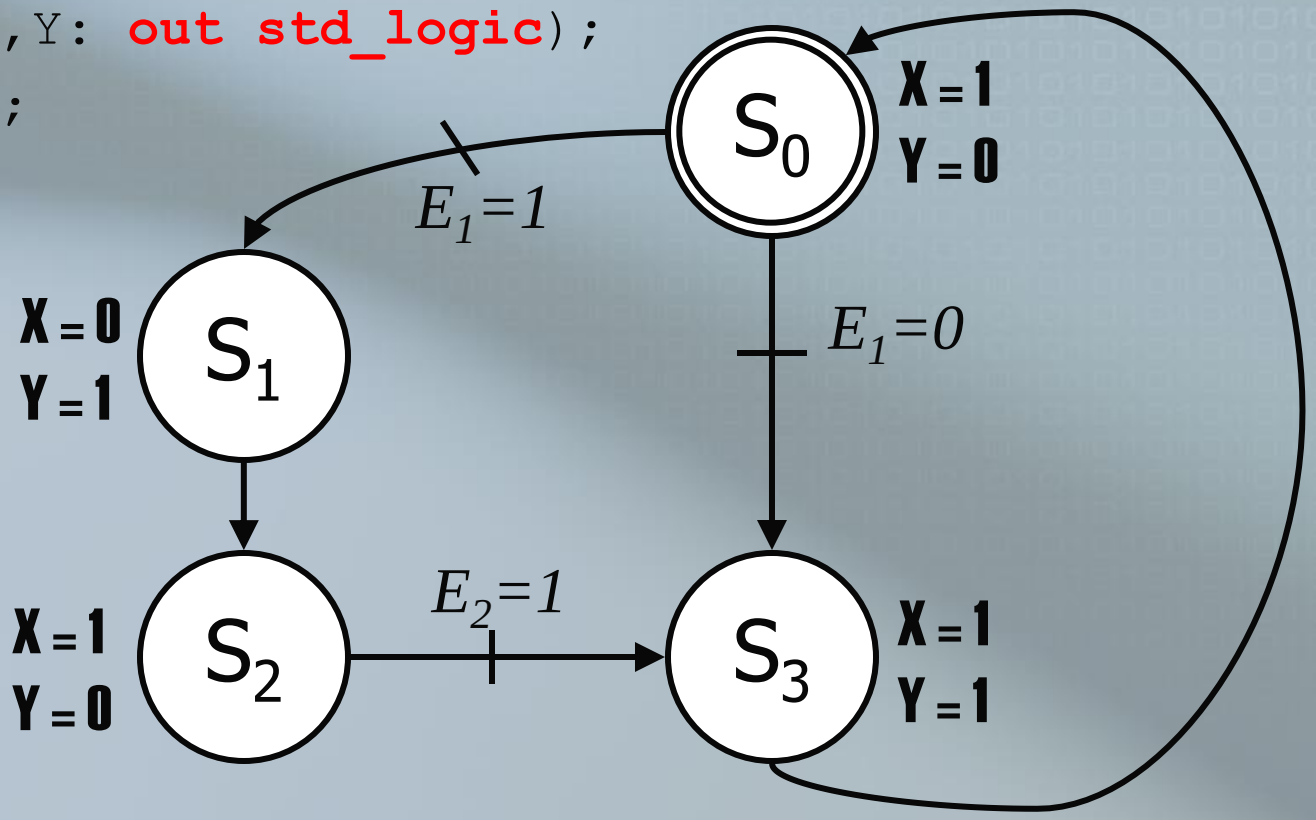
- Description à l'aide de process
  1) Process décrivant le séquentiel (registre d'états)
  2) Process décrivant le combinatoire des états futurs
  3) Process décrivant le combinatoire des sorties

2) et 3) peuvent être mis dans un même process



Machine de Moore - VHDL

```
entity MAE is
port( h,raz: in std_logic;
      e1,e2: in std_logic;
      X,Y: out std_logic);
end MAE;
```



C4

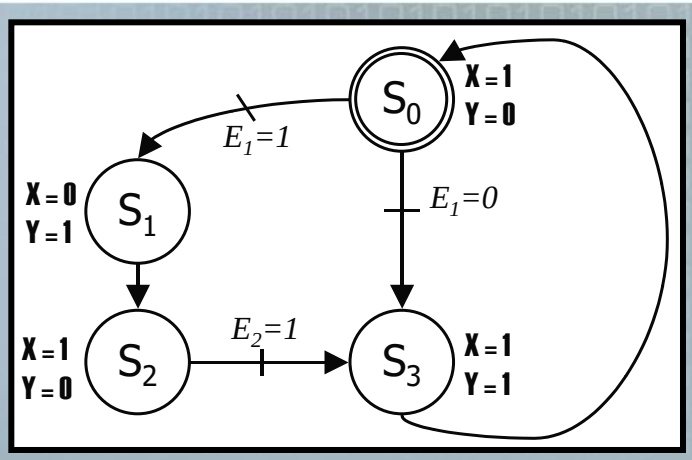
44

Machine de Moore - VHDL

```
architecture Moore3 of MAE is
-- Definition d'un type etat
type etat is(S0,S1,S2,S3);
signal EP, EF: etat;

begin

 -- Process du Registre d'etats
 process(h,raz)
 begin
   if raz='0' then EP <= S0;
   elsif rising_edge(h) then EP <= EF;
   end if;
 end process;
```

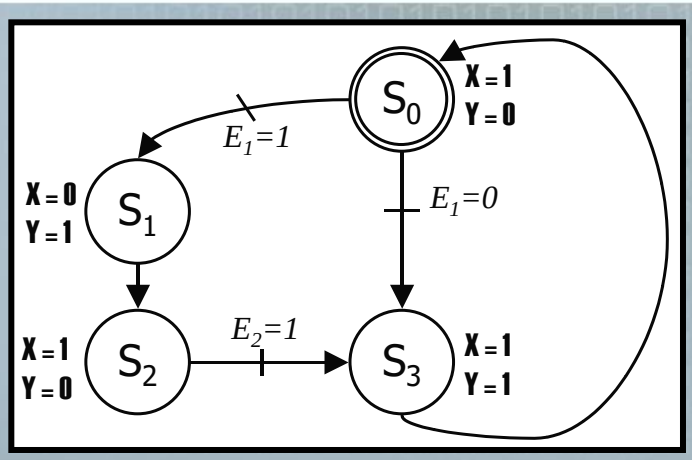


Codage des états non spécifié. Établi par un outil de synthèse en fonction de consignes

C4

45

Machine de Moore - VHDL

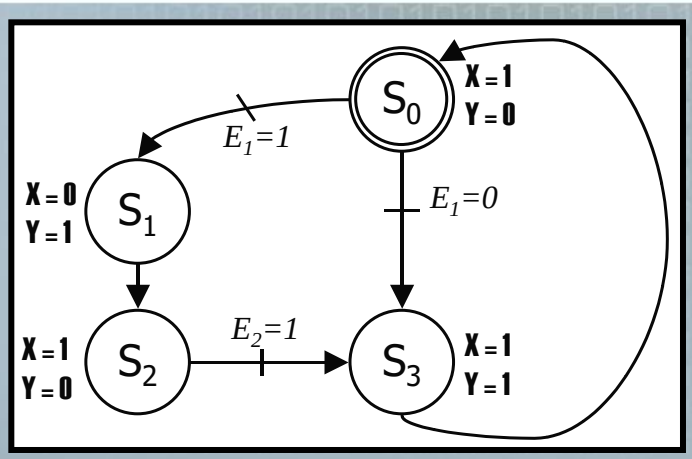



```
-- Combinatoire des etats
process(EP,e1,e2)
begin
  case (EP) is
    when S0 => EF<=S3; if e1='1' then EF<=S1; end if;
    when S1 => EF<=S2;
    when S2 => EF<=S2; if e2='1' then EF<=S3; end if;
    when S3 => EF<=s0;
  end case;
end process;
```

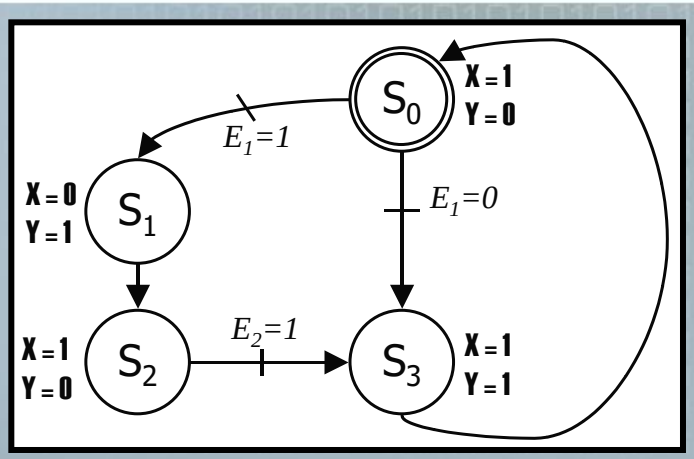
L'état futur doit être spécifié Pour TOUTES les combinaisons des entrées et de l'état présent

Machine de Moore - VHDL

```
-- Combinatoire des sorties
process(EP)
begin
  case (EP) is
    when S0 => X<='1'; Y<='0';
    when S1 => X<='0'; Y<='1';
    when S2 => X<='1'; Y<='0';
    when S3 => X<='1'; Y<='1';
  end case;
 end process;
end Moore3;
```



Machine de Moore - VHDL



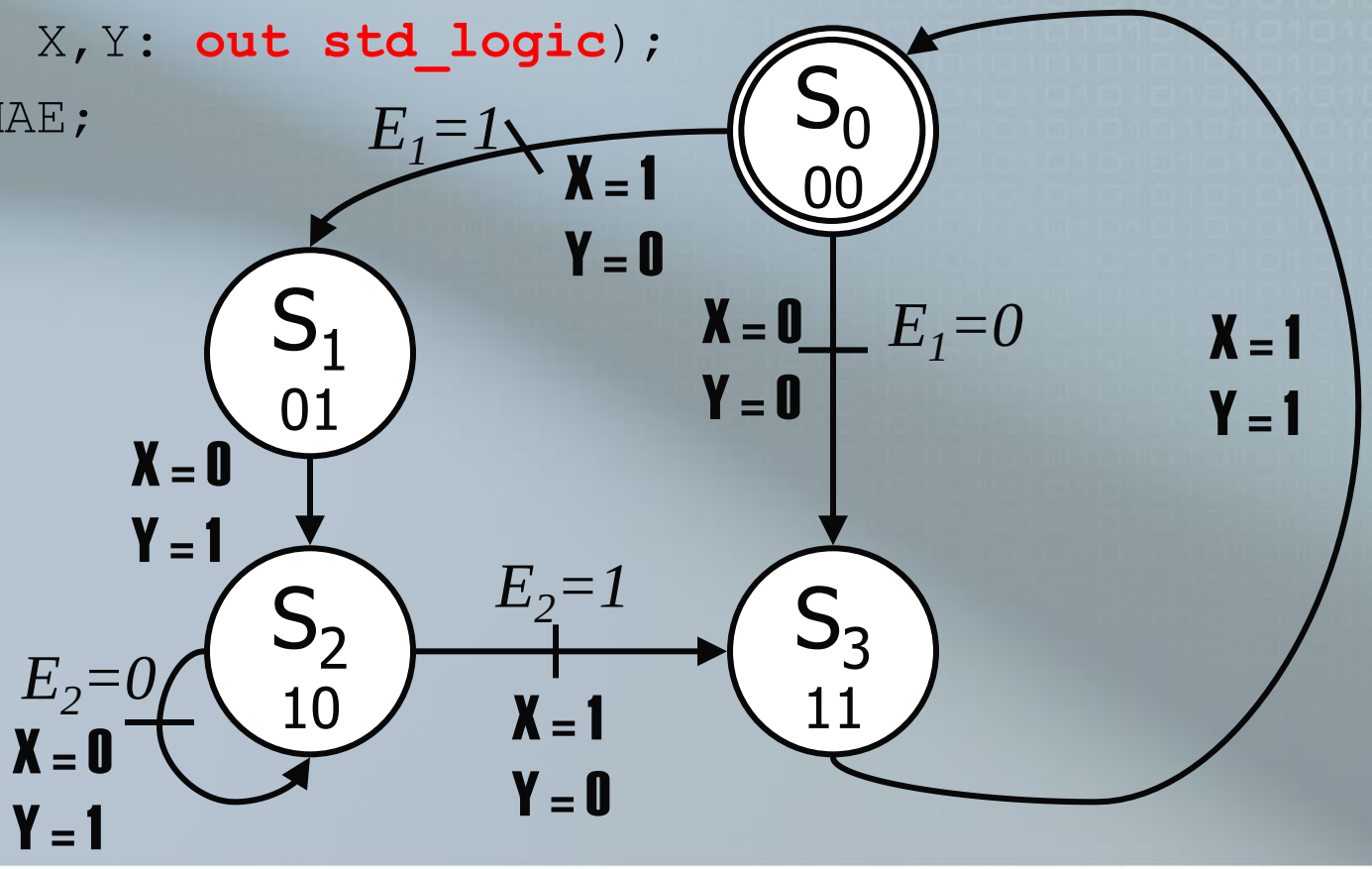

```
-- Tout le combinatoire en un process
process(EP,e1,e2)
begin
  X<='1'; Y<='1';
  case (EP) is
    when S0 => Y<='0';
               EF<=S3; if e1='1' then EF<=S1; end if;
    when S1 => X<='0';
               EF<=S2;
    when S2 => Y<='0';
               EF<=S2; if e2='1' then EF<=S3; end if;
    when S3 => EF<=s0;
  end case;
end process;
```

C4

48

Machine de Mealy - VHDL

```
entity MAE is
port( h,raz: in std_logic;
      e1,e2: in std_logic;
      X,Y: out std_logic);
end MAE;
```



C4

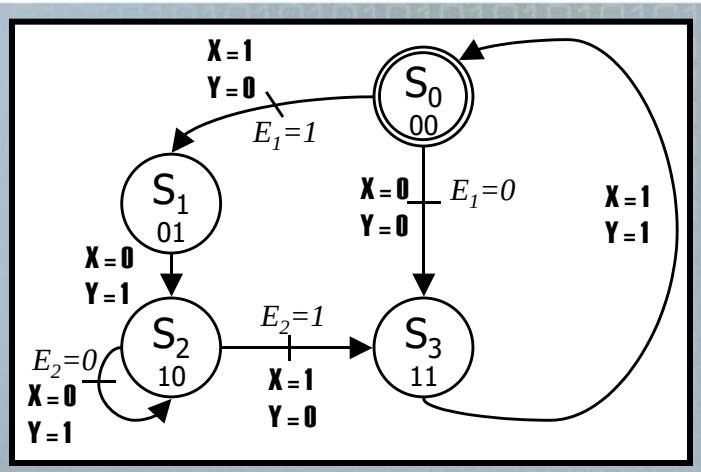
49

Machine de Mealy - VHDL

```
architecture Mealy of MAE is
-- Definition d'un type etat
type etat is(S0,S1,S2,S3);
signal EP, EF: etat;

begin

 -- Process du Registre d'etats
 process(h,raz)
 begin
   if raz='0' then EP <= S0;
   elsif rising_edge(h) then EP <= EF;
   end if;
 end process;
```



C4

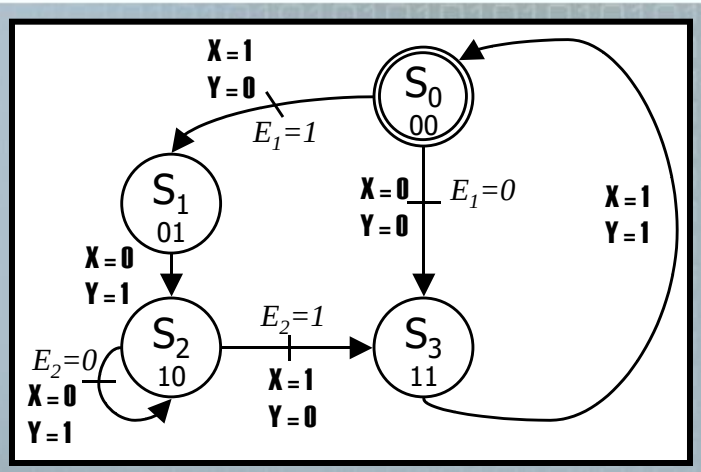
50

Machine de Mealy - VHDL

```
process(EP,e1,e2)
begin

  X<='1'; Y<='1';

  case (EP) is
    when S0 => if e1='1' then EF<=S1; Y<='0';
               else EF<=S3; X<='0'; Y<='0'; end if;
    when S1 => X<='0'; EF<=S2;
    when S2 => if e2='1' then EF<=S3; Y<='0';
               else EF<=S2; X<='0'; end if;
    when S3 => EF<=S0;
    when others => null;
  end case;
end process;
end Mealy;
```



C4

51

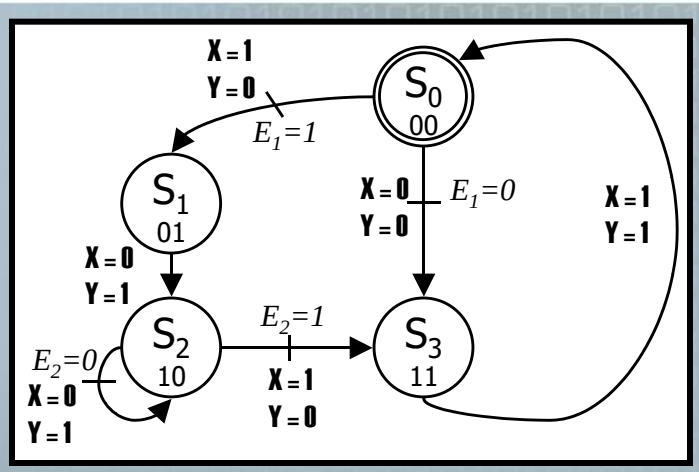
Mealy Synchrone - VHDL

```
architecture Mealy2 of MAE is
-- Definition d'un type etat
type etat is(S0,S1,S2,S3);
signal EP, EF: etat;

-- Sortie du Combinatoire de sortie
signal C1,C2: std_logic;

begin

 -- Process du Registre d'etats
 process(h,raz)
 begin
   if raz='0' then EP <= S0; X<='0'; Y<='0';
   elsif rising_edge(h) then EP <= EF; X<=C1; Y<=C2;
   end if;
 end process;
```



C4

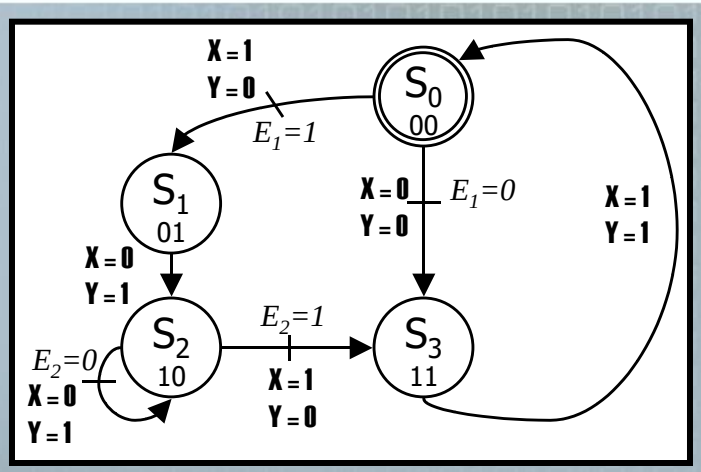
52

Mealy Synchrone - VHDL

```
process(EP,e1,e2)
begin

  C1<='1'; C2<='1';

  case (EP) is
    when S0 => if e1='1' then EF<=S1; C2<='0';
               else EF<=S1; C1<='0'; C2<='0'; end if;
    when S1 => C1<='0'; EF<=S2;
    when S2 => if e2='1' then EF<=S3; C2<='0';
               else EF<=S2; C1<='0'; end if;
    when S3 => EF<=S0;
    when others => null;
  end case;
end process;
end Mealy2;
```



C4

53