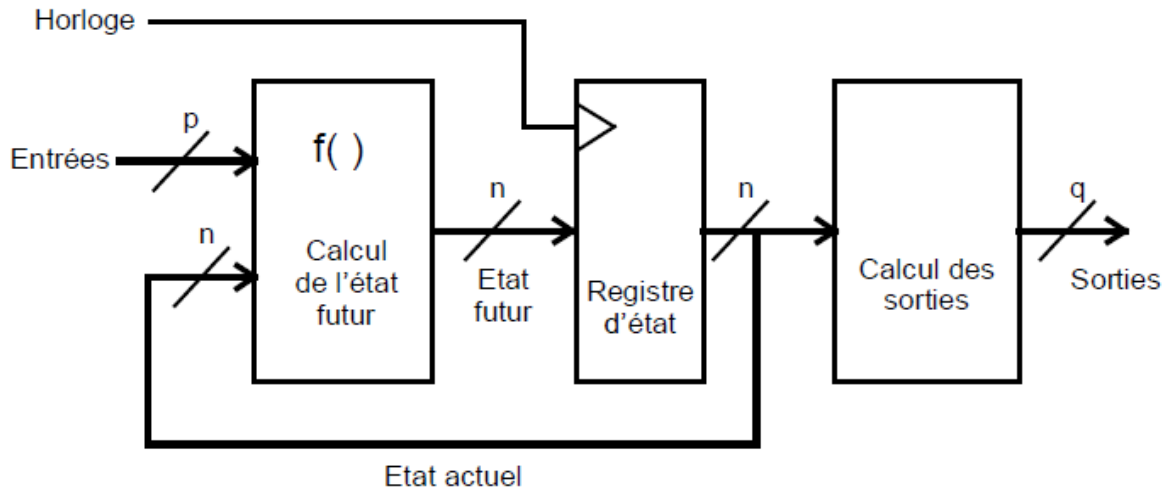
Chapitre 5 : Analyse, conception et description VHDL des machines à états

I. Introduction :

Les machines d'états sont largement utilisées dans les fonctions logiques de contrôle, qui forment le coeur de nombreux systèmes numériques. La quasi totalité des fonctions séquentielles standard, compteurs, par exemple, peuvent être analysées, ou synthétisées, en adoptant le point de vue « machine d'états ».
Une machine d'états est un système dynamique (i.e. évolutif) qui peut se trouver, à chaque instant, dans une position parmi un nombre fini de positions possibles. Elle parcourt des cycles, en changeant éventuellement d'état lors des transitions actives de l'horloge, dans un ordre qui dépend des entrées externes, de façon à fixer sur ses sorties des séquences déterminées par l'application à contrôler.
Un programmateur de machine à laver en est l'illustration typique : suite à la mise en marche, le programmateur contrôle que la porte est fermée, si oui il commande l'ouverture de la vanne d'arrivée de l'eau, quand la machine est pleine, etc.
L'architecture générale d'une machine d'états simple est celle de la figure ci-dessous :



Le registre d'état, piloté par une horloge, constitue le cœur d'une machine d'états. Le registre d'état est constitué de n bascules synchrones, nous admettrons dans ce qui suit, sauf précision contraire, que ce sont des bascules D.

L'entrée du registre d'état constitue l'état futur, celui qui sera chargé lors de la prochaine transition active de l'horloge. Le registre d'état constitue la mémoire de la machine, l'élément qui matérialise l'histoire de son évolution.

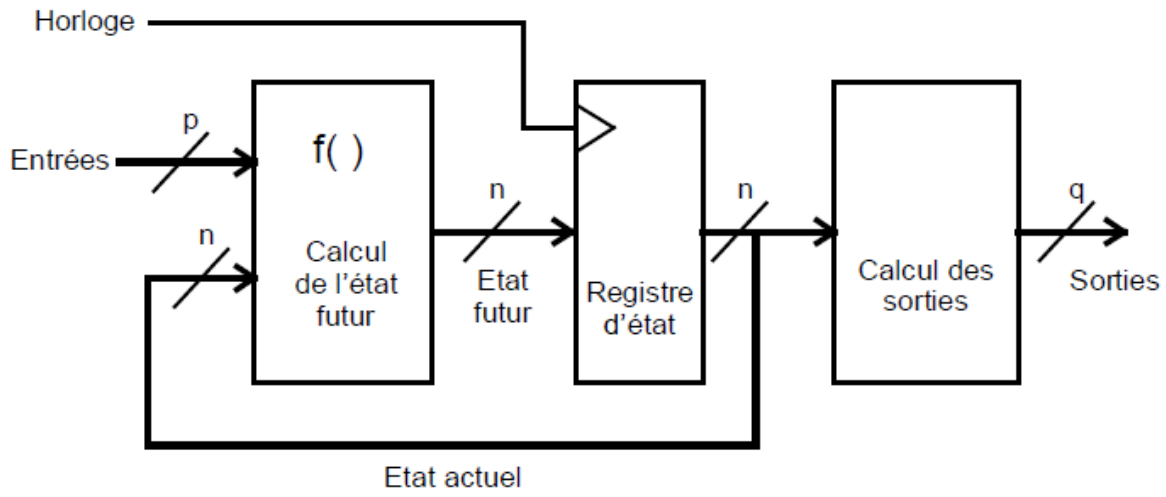
La fonction logique combinatoire, f( ), qui calcule l'état futur, fournit une valeur qui dépend de l'état présent et des entrées.

II- Architectures des machines d'état :

Suivant la façon dont les sorties dépendent des états et des commandes, on distingue deux types de machines d'états : les machines de Moore et les machines de Mealy.

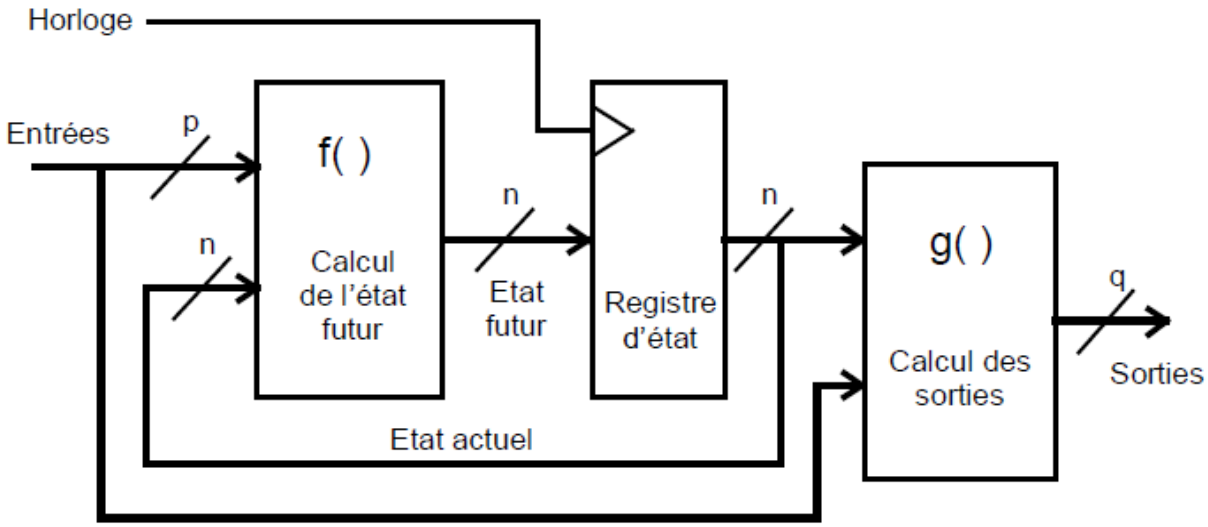
1- Machine de Moore :

Dans les machines de Moore, les sorties ne dépendent que de l'état actuel de la machine.
L'état futur est calculé à partir des entrées et de l'état présent.



2- Machines de Mealy :

Dans les machines de Mealy, les sorties dépendent de l'état de la machine et des entrées. Les sorties dépendent directement de l'entrée et ceci indépendamment de l'horloge (clk). Il s'agit d'une sortie asynchrone.
L'état futur est calculé à partir des entrées et de l'état présent.
Le nombre d'états plus réduit que pour une machine de Moore.



III- Diagramme ou Graphe d'états :

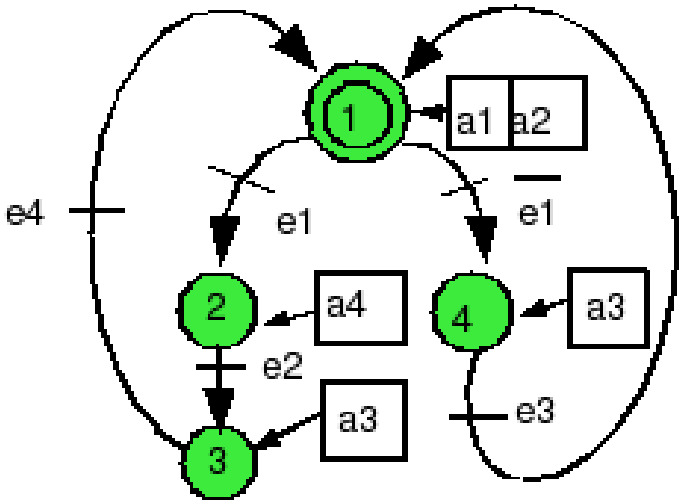
1- Définition :
- Un diagramme ou graphe d'états permet d'avoir une représentation graphique d'un système séquentiel.
- Il est constitué par l'énumération de tous les états du système.
- Un seul de ces états peut être actif à la fois.
- A chaque état est associé la valeur de la (ou des) grandeur(s) de sortie.

Dans un diagramme d'états, on associe à *chaque* valeur possible du registre d'état, une case. L'évolution du système est représentée par des flèches, les transitions, qui vont d'un état à un autre. Comme pour les bascules élémentaires, une transition est effectuée si trois conditions sont réunies :
1. Le système est dans l'état « source » de la transition considérée,
2. une éventuelle condition de réalisation sur les entrées doit être vraie,
3. un front actif d'horloge survient.

Si aucune transition n'est active, le système reste dans son état initial. S'il n'y a pas d'ambiguïté le signal d'horloge est généralement omis (horloge unique), mais il conditionne toutes les transitions.

2- Exemple :

On considère le système séquentiel modélisé par le graphe d'états suivants :



Les cercles numérotés de 1 à 4 représentent les 4 états du système.
Les entrées du système sont e1, e2, e3 et e4 et les sorties sont a1, a2, a3 et a4.
Le système passe de l'état initial 1 à l'état 2 si e1=1 et à l'état 4 si e1=0.
Le système passe de l'état 2 à l'état 3 si e2=1.
Le système passe de l'état 3 à l'état 1 si e4=1.
Le système passe de l'état 4 à l'état 1 si e3=1.

Les sorties a1 et a2 sont activés (a1=a2=1) si l'état 1 est actif.
La sortie a3=1 si l'état 1 est actif ou l'état 4 est actif.
La sortie a4=1 si l'état 2 est actif.

III- Analyse et description VHDL des machines d'états (utilisant le codage binaire naturel) :

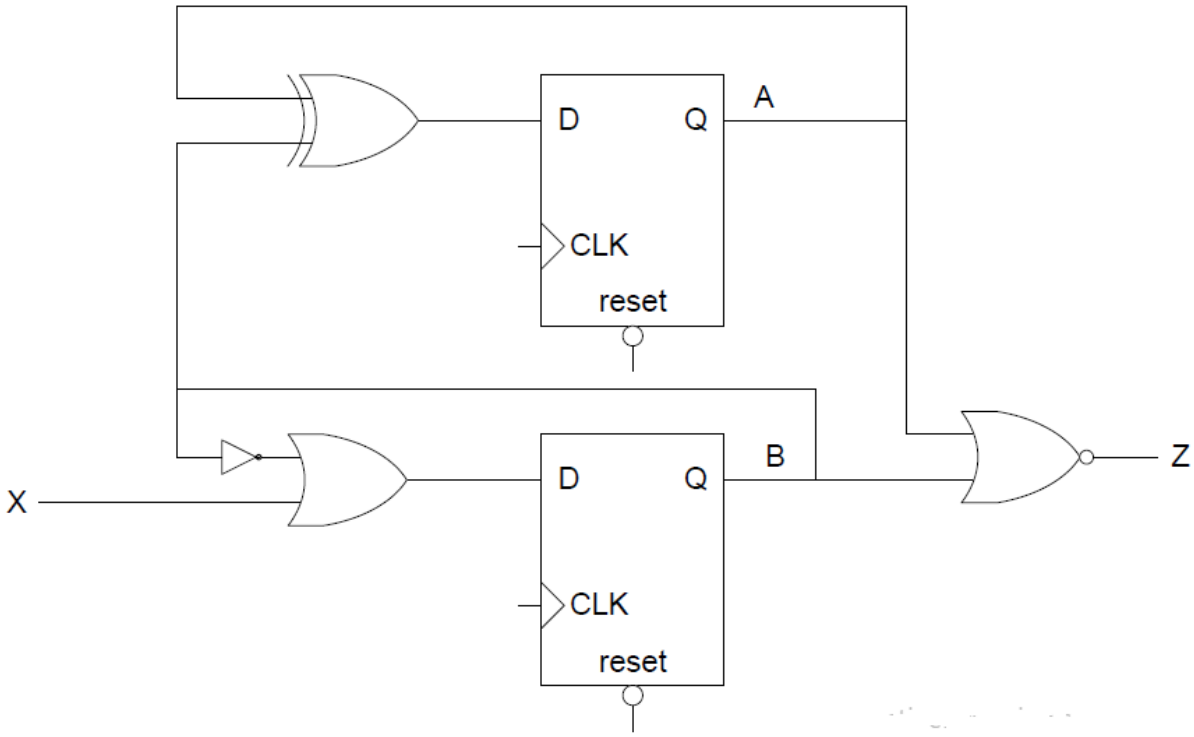
Dans les machines d'états utilisant le codage binaire naturel, si n est le nombre des bascules utilisés et N est le nombre des états du système. Le nombre des bascules n à utiliser est le plus petit entier telle que : $\boldsymbol{N \leq 2^n}$

Par exemple, si le nombre d'état est N=5, on a besoin de 3 bascules (n=3). Ainsi, on a $\boldsymbol{N = 5 \leq 2^n = 2^3 = 8.}$

L'analyse d'un circuit séquentiel consiste à déterminer les équations des états et des sorties, établir la table de vérité des transitions et de dresser son graphe d'état.

1- Exemple introductif:

On considère le circuit logique donné par le logigramme suivant :



Les équations du circuit sont données par :

A+ = A xor B;
B+ = not(B) or X;
Z = A nor B;

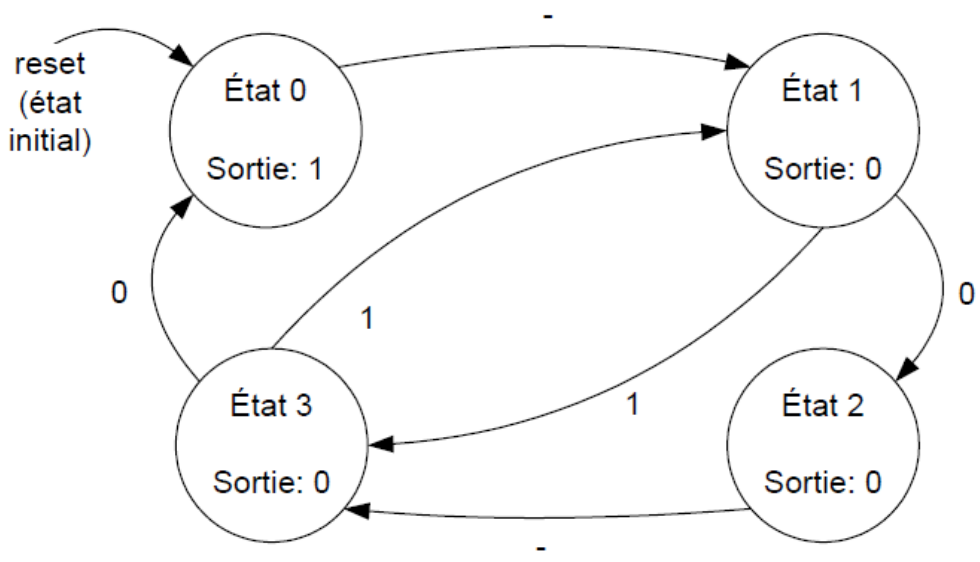
Table de transitions

Pour passer des équations de commandes des bascules au diagramme d'états (ou inversement), le concepteur peut toujours recourir à une table de vérité qui récapitule toutes les transitions possibles.
Si cette méthode est systématique, elle présente évidemment l'inconvénient d'être fort lourde. Le nombre de variables d'entrées de la table devient vite, même pour des problèmes simples, très élevé.

Le tableau d'état correspondant au circuit précédent est donc donné par :

Etat présent		Entrée	Etat futur		Sortie
A	B	X	A+	B+	Z
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	1	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	1	0

Le graphe d'état est donc donné par :



Avec

état 0: (AB) = « 00 »
état 1: (AB) = « 01 »
état 2: (AB) = « 10 »
état 3: (AB) = « 11 »

Il s'agit d'une machine de Moore : la sortie ne dépend que de l'état présent.

Description VHDL à partir du logigramme :

```
library IEEE;
use IEEE.std_logic_1164.all;
entity cctsequentielex1 is
port (
reset, CLK, X : in STD_LOGIC;
Z : out STD_LOGIC
);
end cctsequentielex1;
architecture arch1 of cctsequentielex1 is
signal A, B : STD_LOGIC;
begin
process(CLK, reset) is
begin
if (reset = '0') then
A <= '0';
B <= '0';
elsif (rising_edge(CLK)) then
```

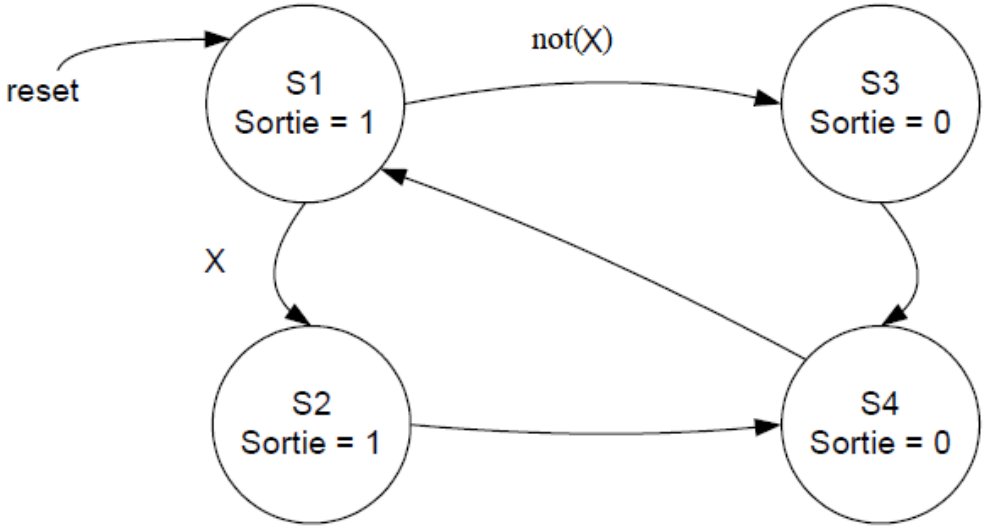
```
A <= A xor B;
B <= x or not(B);
end if;
end process;
z <= not(A or B);
end arch1;
```

Description VHDL à partir du diagramme d'états :

```
architecture arch3 of cctsequentielex1 is
type type_etat is (Etat0, Etat1, Etat2, Etat3);
signal etat : type_etat := Etat0;
begin
process(CLK, reset) is
begin
if (reset = '0') then
etat <= Etat0;
elsif (rising_edge(CLK)) then
case etat is
when Etat0 =>
etat <= Etat1;
when Etat1 =>
if x = '0' then etat <= Etat2;
else etat <= Etat3;
end if;
when Etat2 =>
etat <= Etat3;
when Etat3 =>
if x = '0' then etat <= Etat0;
else etat <= Etat1;
end if;
end case;
end if;
end process;
z <= '1' when etat = Etat0 else '0';
end arch3;
```

2- Styles de descriptions VHDL d'une machine d'états :

On considère un circuit séquentiel modélisé par le graphe d'état ci-dessous



Description VHDL avec un seul processus :

Dans ce style, un seul processus est utilisé pour le calcul des états et des sorties.

```
Architecture unprocessus of cctsequentielex2 is
type type_etat is (S1, S2, S3, S4);
signal etat : type_etat := S1;
begin
process(CLK, reset) is
begin
if (reset = '0') then
etat <= S1;
sortie <= '1';
elsif (rising_edge(CLK)) then
case etat is
when S1 =>
if x = '0' then
etat <= S3;
sortie <= '0';
else
etat <= S2;
sortie <= '1';
end if;
when S2 | S3 =>
etat <= S4;
sortie <= '0';
when S4 =>
etat <= S1;
sortie <= '1';
end case;
end if;
end process;
end unprocessus;
```

Description VHDL avec deux processus :

Dans ce style, deux processus sont utilisés : un processus séquentiel pour le calcul de l'état et un autre processus combinatoire pour les sorties (peut être remplacé par des énoncés concurrents).

```
architecture deuxprocessus of cctsequentielex2 is
type type_etat is (S1, S2, S3, S4);
signal etat : type_etat := S1;
begin
process(CLK, reset) is
begin
if (reset = '0') then
etat <= S1;
elsif (rising_edge(CLK)) then
case etat is
when S1 =>
if x = '0' then
etat <= S3;
else
etat <= S2;
end if;
when S2 | S3 =>
etat <= S4;
when S4 =>
etat <= S1;
end case;
end if;
end process;
process(etat)
begin
case etat is
when S1 | S2 => sortie <= '1';
when S3 | S4 => sortie <= '0';
end case;
end process;
end deuxprocessus;
```

V- Description VHDL des machines d'états (utilisant le type de codage one hot) :

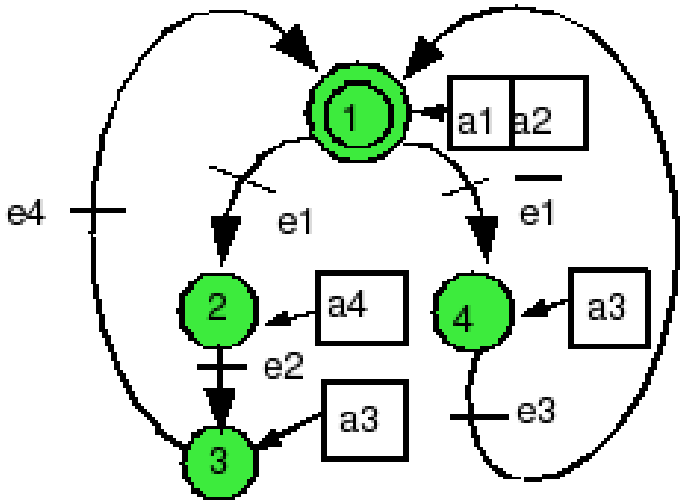
Dans les machines d'états utilisant le type de codage « one hot », chaque état est représenté par une seule bascule. Si un état est actif, la sortie de la bascule correspondante est active.

Cette méthode de codage est une méthode systématique et plus pratique. En effet, on n'a pas besoin de passer par la table de transitions et les tableaux de Karnaught comme dans les machines d'état utilisant le type de codage binaire naturel. En contrepartie, on passe directement à la synthèse des équations d'états en identifiant les conditions d'activation et désactivation de chaque état.

L'inconvénient des machines d'états utilisant le type de codage « one hot » est qu'elles consomment beaucoup plus des bascules par rapport aux machines d'état utilisant le type de codage binaire naturel.

Exemple :

On reprend l'exemple du système séquentiel décrit par le graphe d'états suivants :



On utilisera un bouton poussoir Init pour initialiser le système (activer l'état initial Etat 1)

On détermine les conditions d'activation et de désactivation de chaque état :

Etat	**Condition d'activation**	**Condition de désactivation**
Etat 1	$Q_3e_4 + Q_4e_3 + Init$	$e_1 + \overline{e_1} = 1$
Etat 2	Q_1e_1	e_2
Etat 3	Q_2e_2	e_4
Etat 4	$Q_1\overline{e_1}$	e_3

Remarque 1: Réalisation à l'aide des bascules RS

On peut réaliser ce système en utilisant 4 bascules RS. Dans ce cas, pour chaque état, on met la condition d'activation dans l'entrée S de la bascule et la condition de désactivation dans l'entrée RS de la bascule.

Remarque 2: Réalisation à l'aide des bascules D

Dans ce cas, on doit déterminer l'équation de l'entrée $D_i = Q_i^+$ (calcul de l'état futur) correspondant à chaque bascule Di représentant un état Etat i.

Dans notre exemple, $i \in \{1,2,3,4\}$

Mise en équations des états :

Etat 1 :

$$Q_1^+ = Q_3e_4 + Q_4e_3 + init + \overline{e_1 + \overline{e_1}}Q_1$$

Après simplification

$$Q_1{}^+ = Q_3 e_4 + Q_4 e_3 + init$$

Etat 2 :

$$Q_2{}^+ = Q_1 e_1 + \overline{e_2} Q_2$$

Etat 3 :

$$Q_3{}^+ = Q_2 e_2 + \overline{e_3} Q_3$$

Etat 4 :

$$Q_4{}^+ = Q_1 \overline{e_1} + \overline{e_3} Q_4$$

Mise en équations des sorties :

$$a_1 = Q_1$$

$$a_2 = Q_1$$

$$a_3 = Q_3 + Q_4$$

$$a_4 = Q_2$$

Description VHDL se basant sur les équations des états et des sorties :

```
-- programme VHDL correspondant au graphe d'états précédent
ENTITY graf1 IS
PORT (Init,e1,e2,e3,e4,clk : IN std_logic;
      a1,a2,a3,a4 : OUT std_logic);
END graf1;
ARCHITECTURE onehot OF graf1 IS
SIGNAL Q1,Q2,Q3,Q4 : std_logic;
BEGIN
  PROCESS(clk) BEGIN
    IF (clk'event AND clk='1') THEN
-- équations des états
      Q1 <= (Q3 AND e4) OR (Q4 AND e3) OR Init;
      Q2 <= (Q1 AND e1) OR (Q2 AND NOT e2);
      Q3 <= (Q2 AND e2) OR (Q3 AND NOT e4);
      Q4 <= (Q1 AND NOT e1) OR (Q4 AND NOT e3);
    END IF;
  END PROCESS;
--équations de sorties
  a1 <= Q1;
  a2 <= Q1;
  a3 <= Q3 OR Q4;
  a4 <= Q2;
END onehot;
```