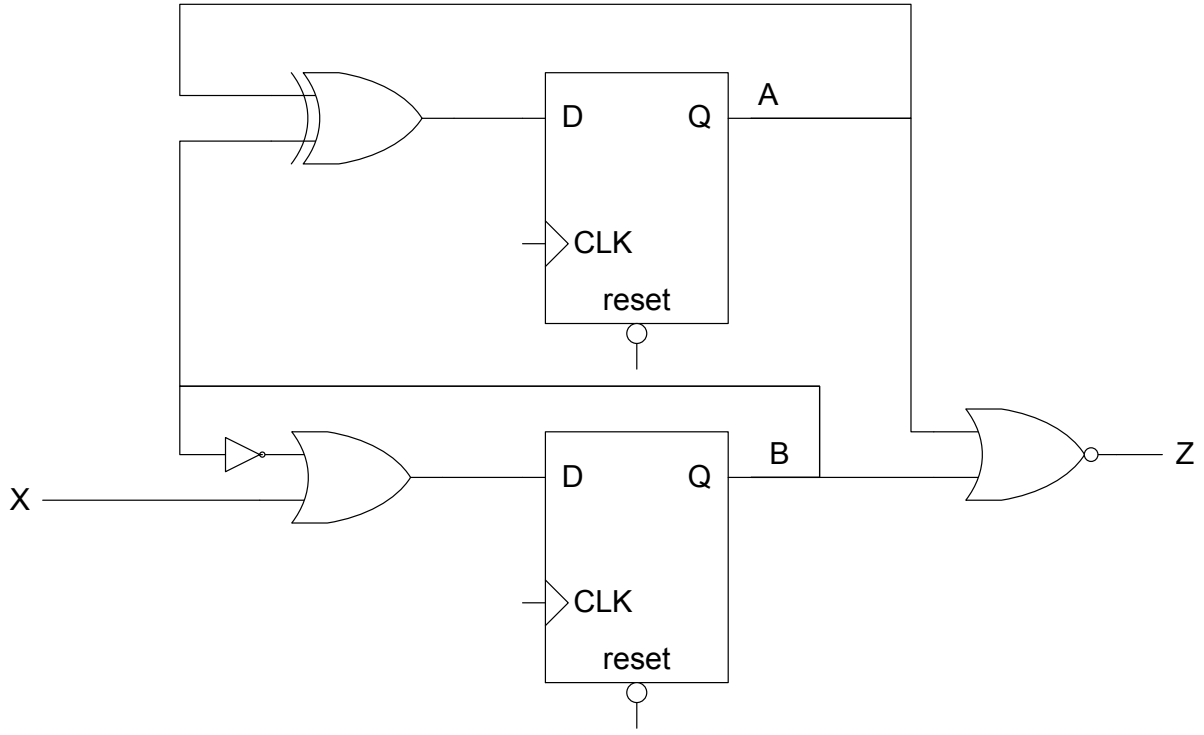
Analyse de machines à états et leur description en VHDL

Sujets de ce thème

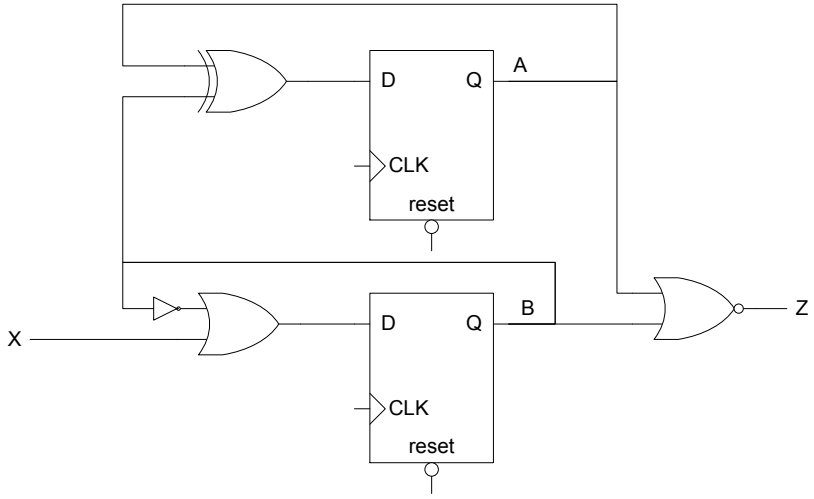
- Analyser un circuit séquentiel synchrone à partir de son schéma.
- État et sorties en fonction du temps.
- Diagrammes d’états et code VHDL.

Analyse d’un circuit séquentiel synchrone (1)

- On analyse un circuit pour en comprendre le fonctionnement.
- Analyser un circuit séquentiel synchrone en quatre étapes:
  1. identifier les variables d’états: les sorties des éléments à mémoire;
  2. écrire les équations d’états et les équations de sortie;
  3. dresser le tableau d’états; et,
  4. dessiner le diagramme d’états.



Analyse d'un circuit séquentiel synchrone (2)



1.{A, B}

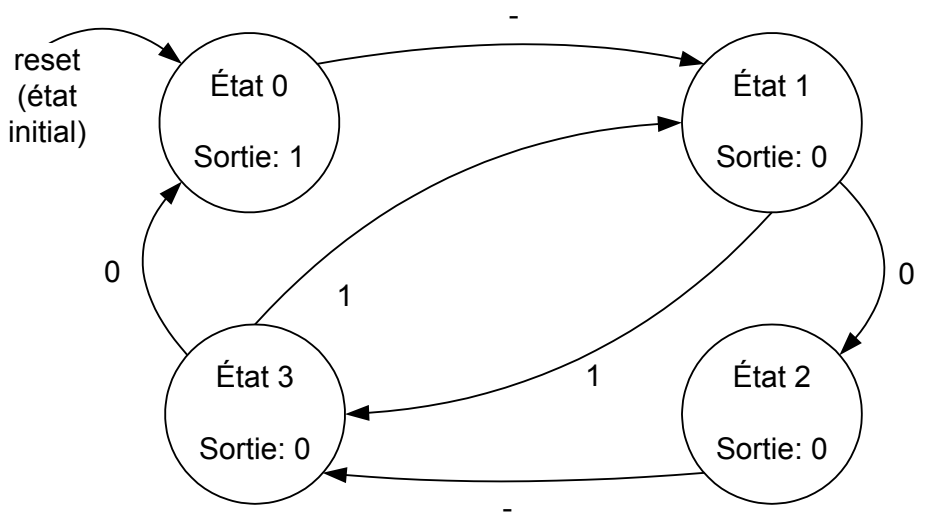
2.
A^+ = A xor B;
B^+ = B' or X;
Z = A nor B;

3. Tableau d'états:

état présent		entrée	état prochain		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

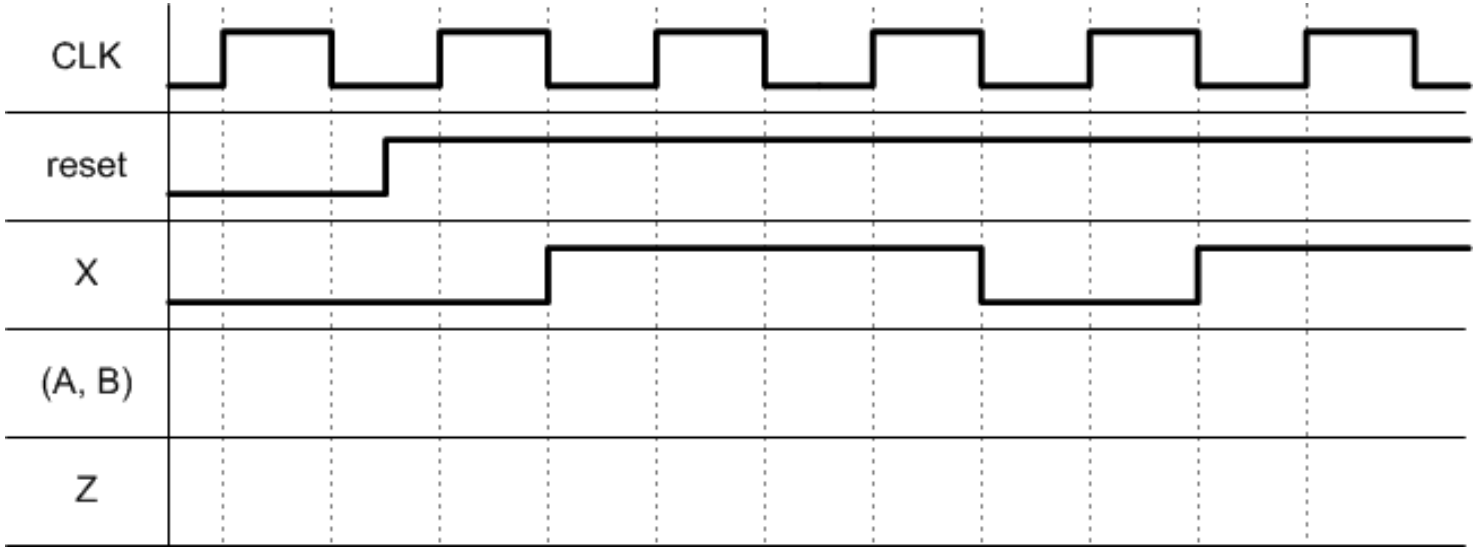
4. Diagramme d'états:

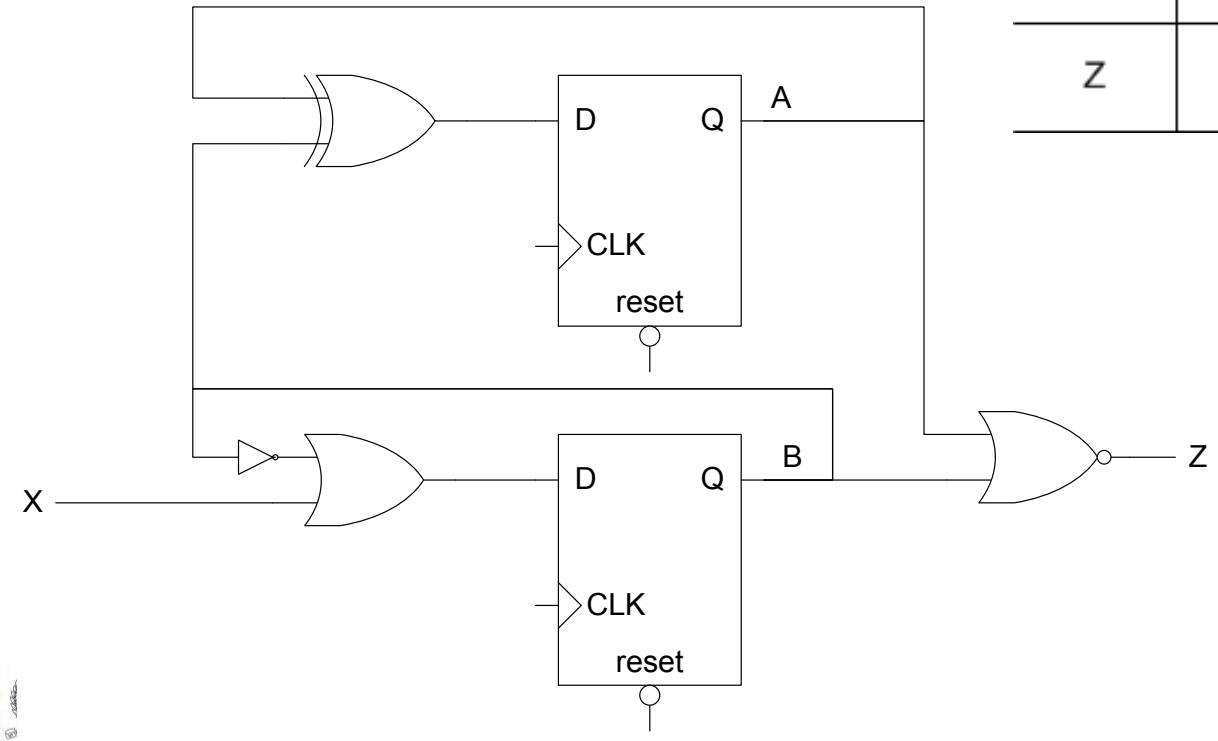
état 0: AB = « 00 »
état 1: AB = « 01 »
état 2: AB = « 10 »
état 3: AB = « 11 »

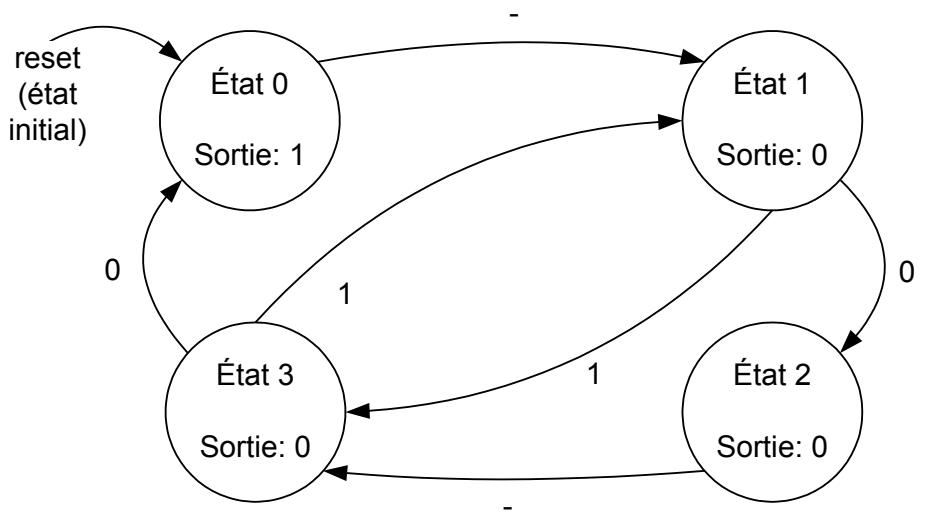


C'est une machine de Moore,
la sortie ne dépend que de l'état présent.

Évolution de l'état et des sorties en fonction du temps

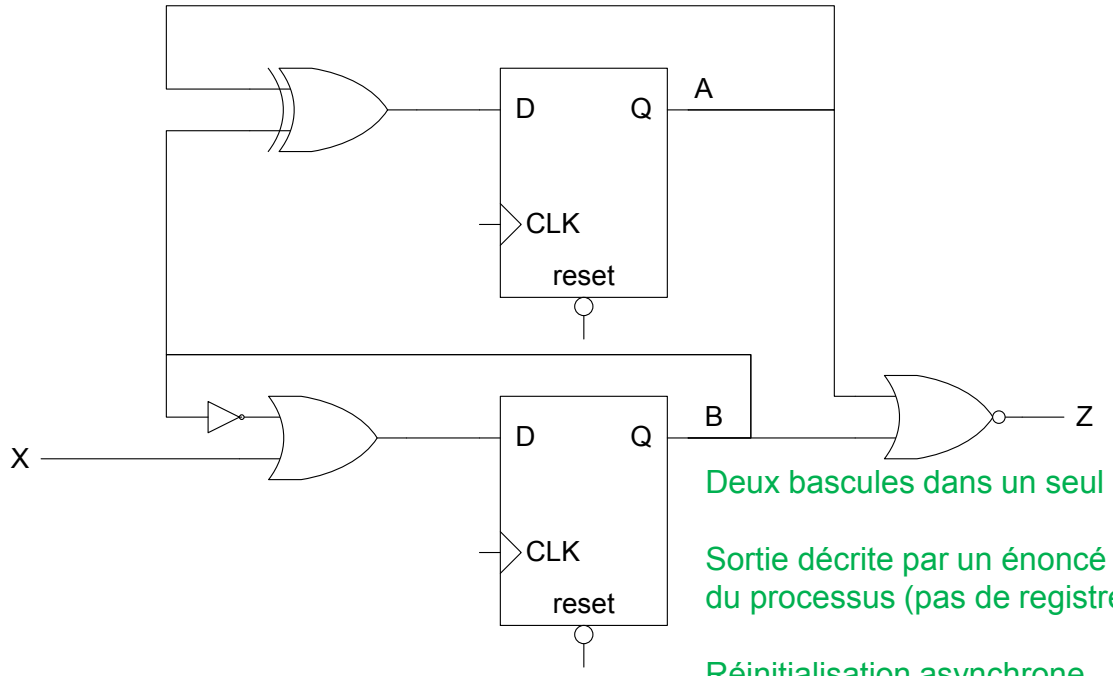






Description d'une machine à états en VHDL: 1. à partir d'un schéma

- Approche adéquate:
  - quand on désire modéliser un circuit pour lequel on a le schéma
  - quand on a les équations d'états et de sortie



Deux bascules dans un seul processus

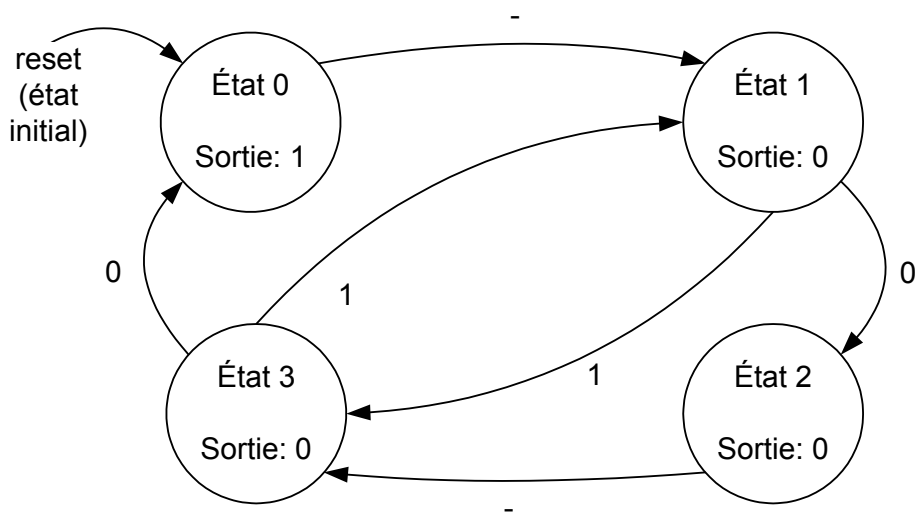
Sortie décrite par un énoncé concurrent à l'extérieur du processus (pas de registre).

Réinitialisation asynchrone.

```
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 d'une machine à états en VHDL
2. à partir d'un diagramme d'états

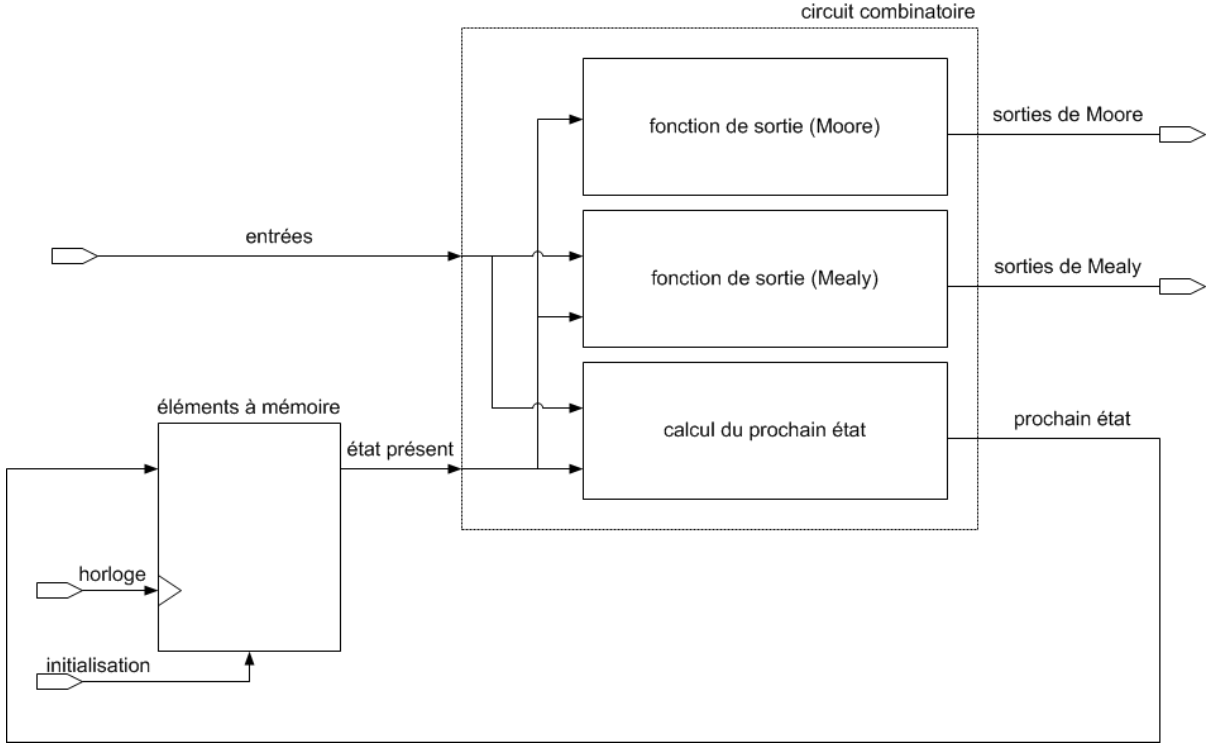
- Approche beaucoup plus puissante:
  - identifier les états, les conditions de transition et les sorties pour chaque état
  - pas besoin d'équations d'états
  - plus lisible, robuste, facile à maintenir




```
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;
```

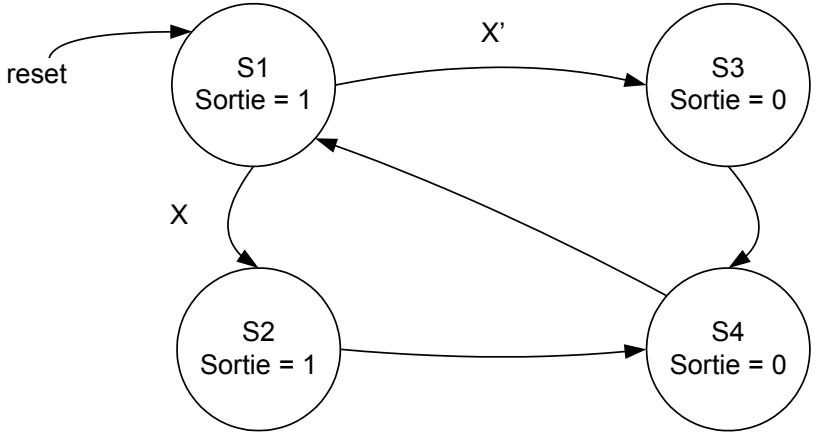
Trois styles de description d'une machine à états en VHDL

- Les trois parties d'une machine à états sont :
  - les éléments à mémoire qui conservent l'état présent de la machine;
  - un circuit combinatoire qui calcule le prochain état; et,
  - un circuit combinatoire qui calcule les sorties de Moore et de Mealy.
- Il y a trois styles principaux de description selon la répartition des trois parties de la machine sur un ou plusieurs processus.



Trois styles de description d'une machine à états en VHDL
1. Avec un seul processus

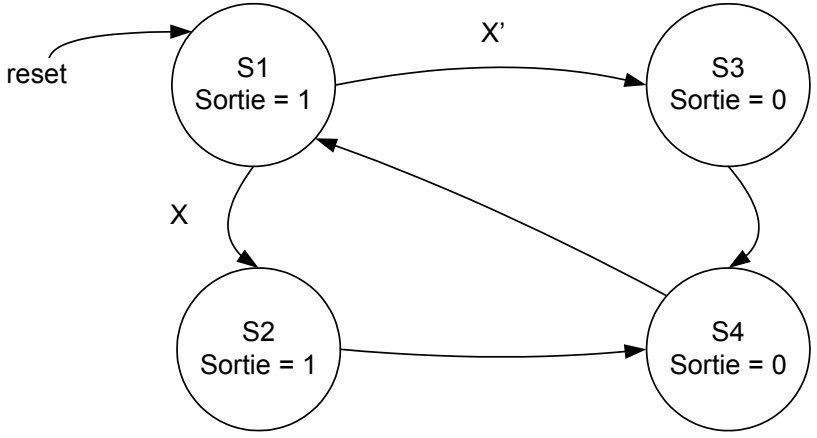
- Attention aux sorties:
  - inférence de registres pour les sorties
  - spécifier la sortie du prochain état étant donnés un état et une entrée présentes.
  - si plusieurs conditions résultent en un état donné, il faut spécifier la sortie de Moore de cet état à chaque fois.




```
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;
```

Trois styles de description d'une machine à états en VHDL
2. Avec deux processus

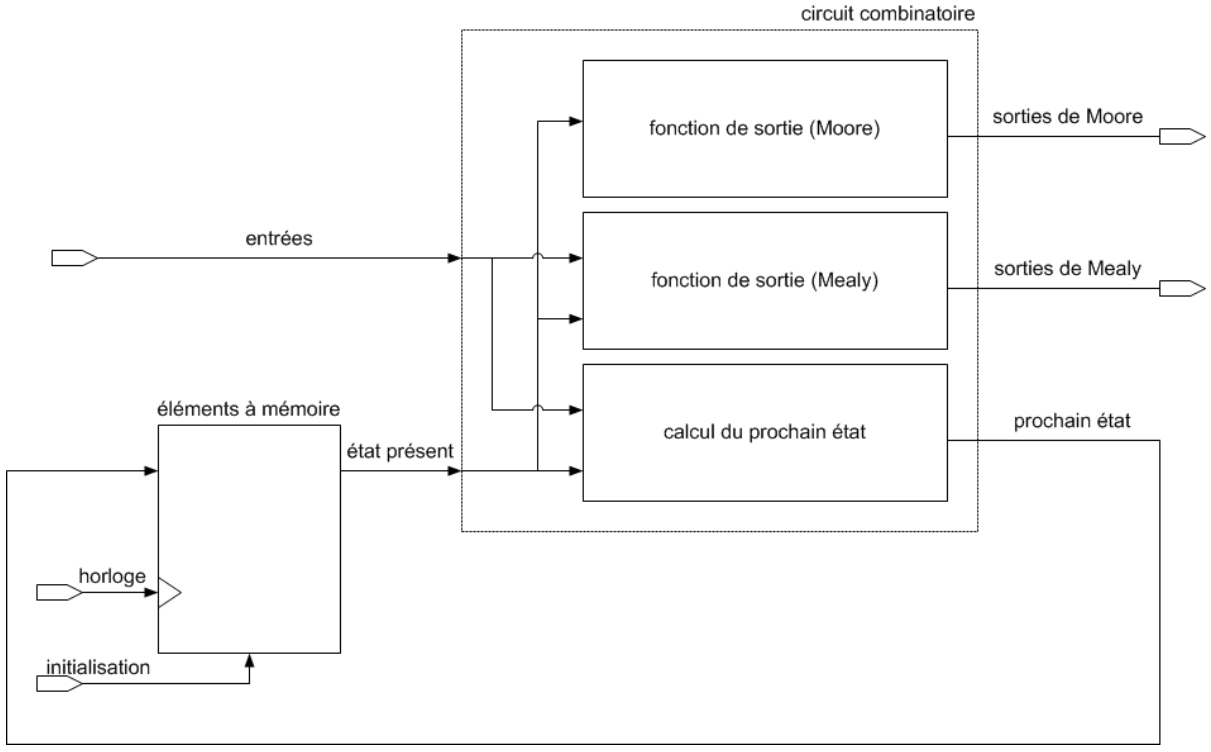
- Bon compromis entre la flexibilité et la lisibilité du code.
- Deux processus:
  - un pour le calcul et l'entreposage de l'état
  - un pour les sorties (peut être remplacé par des énoncés concurrents)




```
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;
```

Trois styles de description d'une machine à états en VHDL
3. Avec trois processus

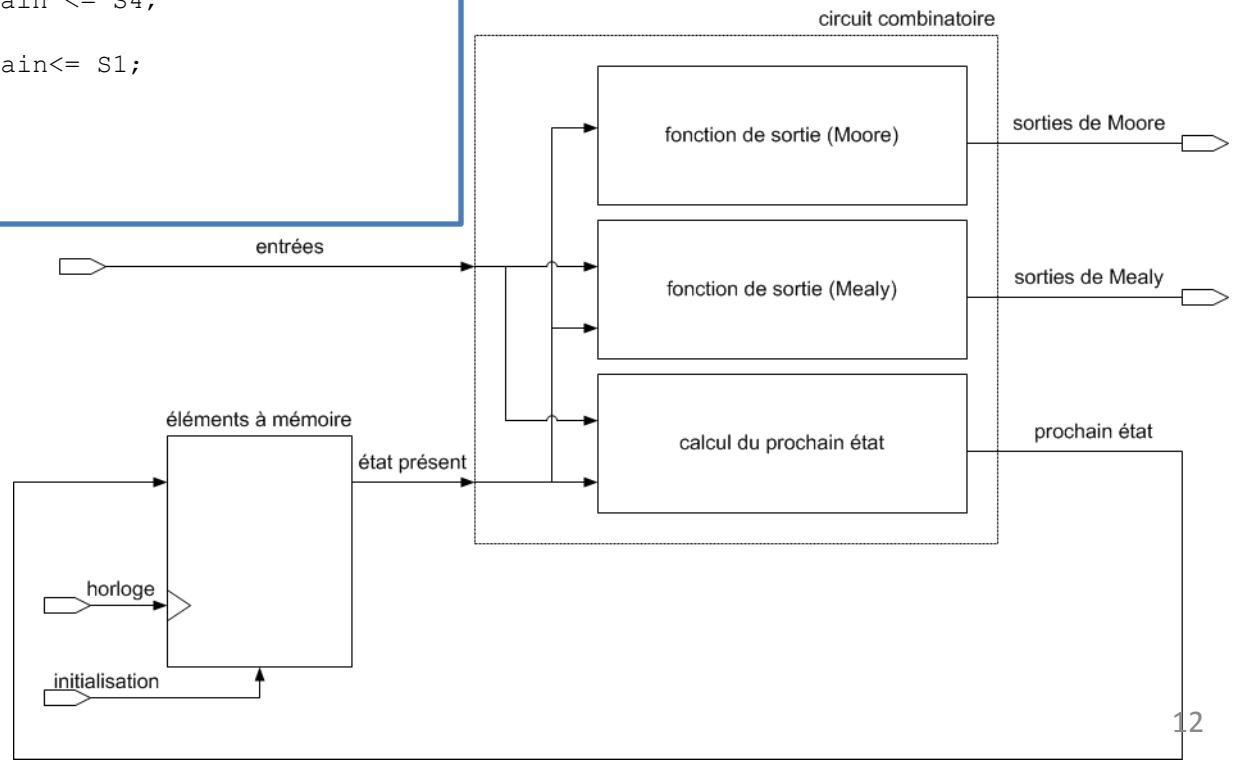
- Style qui correspondre exactement au modèle.
- Code est très lisible mais moins compact que la version à deux processus.
- La liste de sensibilité du processus qui calcule le prochain état inclut le signal qui entrepose l'état courant ainsi que toutes les entrées.
- Le même principe s'applique au processus qui calcule les sorties (pour une machine de Mealy).



Trois styles de description d'une machine à états en VHDL
3. Avec trois processus

```
architecture troisprocessus of cctsequentielex2 is

type type_etat is (S1, S2, S3, S4);
signal etat : type_etat := S1;
signal etat_prochain : type_etat := S1;

begin

   -- processus pour garder l'état actuel en mémoire
   process(CLK, reset) is
   begin
       if (reset = '0') then
         etat <= S1;
      elsif (rising_edge(CLK)) then
         etat <= etat_prochain;
      end if;
   end process;

   -- processus pour les sorties
   process(etat)
   begin
      case etat is
         when S1 | S2 => sortie <= '1';
         when S3 | S4 => sortie <= '0';
      end case;
   end process;
```

```
   -- processus pour le calcul du prochain état
   process(etat, x) is
   begin
      case etat is
         when S1 =>
            if x = '0' then
               etat_prochain <= S3;
            else
               etat_prochain <= S2;
            end if;
         when S2 | S3 =>
            etat_prochain <= S4;
         when S4 =>
            etat_prochain<= S1;
      end case;
   end process;

end troisprocessus;
```



Trois styles de description d'une machine à états en VHDL: Conclusion

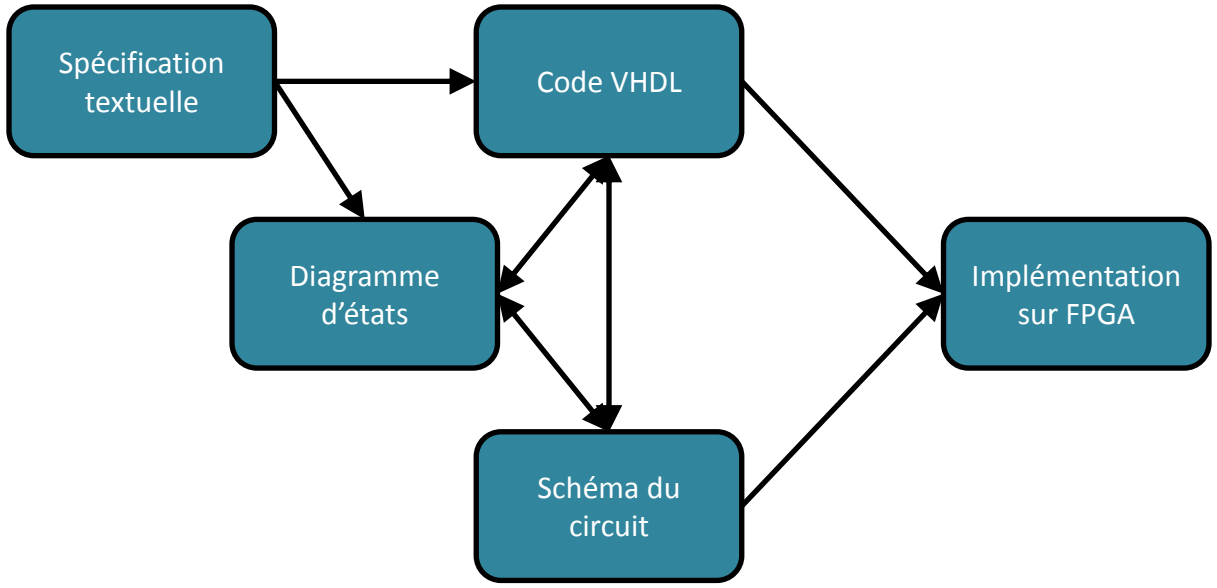
- Les trois styles ont chacun leurs mérites et inconvénients.
- Cette variété d'options illustre à nouveau la très grande richesse de VHDL.
- Cette richesse cause cependant des difficultés parce qu'il n'existe pas une norme unique pour la description de machines à états.
- Il est donc plus difficile de concevoir un synthétiseur qui puisse reconnaître de façon satisfaisante les intentions du concepteur.
- Il est nécessaire de consulter le manuel d'utilisation du synthétiseur utilisé afin de connaître les styles d'encodage de machines à états reconnus.

Conception de machine à états: procédure traditionnelle et avec un HDL

Étape	Procédure traditionnelle	Procédure avec un HDL
Bâtir un diagramme d'états à partir des données du problème.	oui	oui
Bâtir le tableau d'états à partir du diagramme d'états, en identifiant les états par des symboles.	oui	non
Réduire le nombre d'états nécessaires en éliminant les états équivalents.	oui	pas obligatoire peut simplifier la clarté du code
Assigner un code binaire à chaque état, et ajouter cette information au tableau d'état.	oui	par l'outil de synthèse
À partir du tableau d'état complet, obtenir les équations booléennes d'entrée des bascules du type choisi ainsi que les équations booléennes des sorties du système, en simplifiant si possible.	oui	par l'outil de synthèse
Donner le diagramme et/ou construire le circuit.	oui	par l'outil de synthèse
Vérifier, vérifier, vérifier.	oui	oui

Vous devriez maintenant être capable de …

- Analyser un circuit séquentiel synchrone à partir de son schéma. Donner le diagramme d'états qui lui correspond. Donner son état et la valeur de ses sorties en fonction du temps. (B4)
- Décrire une machine à états en VHDL à partir d'un diagramme d'états en choisissant un style approprié et donner le diagramme d'états correspondant à un code VHDL. (B3)



Code	Niveau (http://fr.wikipedia.org/wiki/Taxonomie_de_Bloom)
B1	Connaissance - mémoriser de l'information.
B2	Compréhension – interpréter l'information.
B3	Application – confronter les connaissances à des cas pratiques simples.
B4	Analyse – décomposer un problème, cas pratiques plus complexes.
B5	Synthèse – expression personnelle, cas pratiques plus complexes.