Synthèse en VHDL des systèmes embarqués

Dr. Ing. Chiheb Ameur ABID

Contact: chiheb.abid@gmail.com

Mars 2023

Plan

Synthèse de mémoire

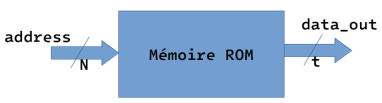
Présentation

- Une mémoire est vue comme un tableau permettant de stocker des données
  - Un mot mémoire désigne une donnée stockée dans une case mémoire
  - Toutes les cases (les mots mémoires) possèdent la même taille
  - Chaque case est identifiée par son emplacement (son adresse)
- Deux types de mémoires
  - Volatile : accès en écriture et en lecture
  - Non volatile : accès en lecture seulement

Synthèse de mémoire

Synthèse de mémoire non volatile

- Mémoire ROM (Read Only Memory)
- Le contenu d'une mémoire morte est défini lors de sa synthèse/fabrication
  - Une mémoire morte correspond à un circuit combinatoire
- Le contenu d'une mémoire ROM ne peut pas être modifié



Synthèse de mémoire

Exemple de synthèse d'une mémoire ROM

- Soit à décrire en VHDL la mémoire ROM ayant le contenu suivant :

Adresse	Donnée
0	01000000
1	01111001
2	00100100
3	00110000
4	00011001
5	00010010
6	00000010
7	01111000
8	00000000
9	00010000
10	00000000
11	00000000
12	00000000
13	00000000
14	00000000
15	00000000

Synthèse de mémoire

Exemple de synthèse d'une mémoire ROM

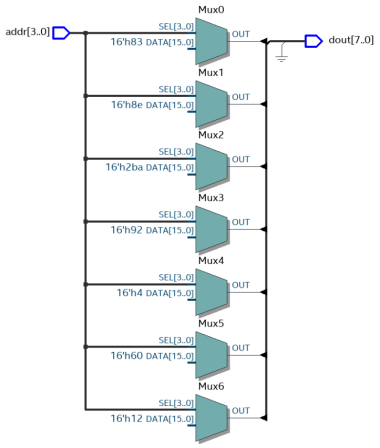
- Chaque donnée est codée sur 8 bits
  - La taille du bus données est 8
- La capacité de la mémoire est de 16 mots mémoire
  - La taille de bus adresse est 4

Synthèse de mémoire

Exemple de synthèse d'une mémoire ROM

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity rom is
6    port (address : in std_logic_vector(3 downto 0);
7    data_out : out std_logic_vector(7 downto 0));
8  end entity;
9  architecture arch_rom of rom is
10 TYPE mem_data IS ARRAY(0 TO 15) OF std_logic_vector(7 DOWNTO 0);
11 constant data : mem_data := (
12   ("01000000"),("01111001"),("00100100"),("00110000"),
13   ("00011001"),("00010010"),("00000010"),("01111000"),
14   ("00000000"),("00010000"),("00000000"),("00000000"),
15   ("00000000"),("00000000"),("00000000"),("00000000"));
16 begin
17   process(address)
18   begin
19     data_out <= data(to_integer(unsigned(address)));
20   end process;
21 end architecture;
```

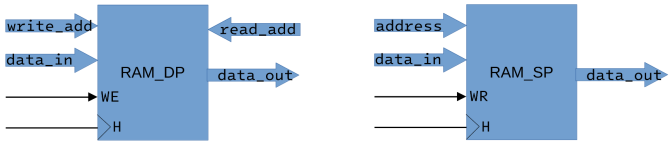

Synthèse de mémoire



Synthèse de mémoire

Synthèse de mémoire volatile

- Simple port / Double port
  - Simple port : un seul bus d'adresse pour la lecture et l'écriture
  - Deux bus d'adresse : un pour la lecture d'un emplacement et l'autre pour l'écriture dans un autre emplacement



Synthèse de mémoire

Synthèse de mémoire volatile

- Mémoire synchrone/asynchrone
  - Synchrone : les données de lecture sont mises à jour à un front de l'horloge
  - Asynchrone : les données de lecture sont mises à jour immédiatement

Synthèse de mémoire

Exemple de synthèse d'une mémoire RAM (1/2)

- Synthèse d'une mémoire RAM de capacité 16 mots où chaque mot est un octet

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.numeric_std.ALL;
4  entity ramDP is port (
5    in_clk,we,reset : in std_logic;
6    adrL, adrE : in std_logic_vector(3 downto 0);
7    dataE : in std_logic_vector(7 downto 0);
8    dataL : out std_logic_vector(7 downto 0)
9    );
10 end entity;
```


Synthèse de mémoire

Exemple de synthèse d'une mémoire RAM (2/2)

```
1  architecture arch_ram of ramDP is
2  subtype mot is std_logic_vector(7 downto 0);
3  type tab is array (0 to 4**2) of mot;
4  signal mem : tab ;
5  begin
6  dataL <= mem(to_integer(unsigned(adrL)));
7  ecriture : process(in_clk,reset)
8  begin
9        if reset='1' then
10         mem(0)<="11101100";
11         mem(1)<="00000100";
12         mem(2)<="11111100";
13       elsif in_clk'event and in_clk='1' then
14         if we='1' then
15           mem(to_integer(unsigned(adrE))) <=dataE;
16         end if;
17       end if;
18   end process;
19 end architecture;
```

Plan

Méthodologie

Décomposition

- Décomposer votre projet
  - Une seule fonction par module VHDL
  - Vos descriptions doivent être simples et lisibles
  - Les descriptions doivent être faciles à comprendre pour le synthétiseur
- Permet une meilleure optimisation lors de la synthèse

Méthodologie

Choix d'architecture

- Quel type d'architecture choisir ?
- Décomposer chaque fois que c'est pertinent :
  - Architecture structurelle
  - Si un composant pourrait être utiliser ailleurs
- Utiliser le comportemental ou data flow lorsque la décomposition n'est plus pertinente
- Data flow
  - Proche des portes logiques
  - Plus de contrôle sur la logique générée
- Comportemental
  - Risque de faire de l'algorithmique éloignée de la synthèse
  - Potentiellement plus clair que data flow
  - Les machines à états

Méthodologie

Processus combinatoire

- L'écriture d'un processus combinatoire doit suivre 2 règles :
  1. Tous les signaux présents à droite d'une affectation ou dans un test doivent être déclarés dans la liste de sensibilité du processus
  2. Tous les signaux affectés dans le processus doivent se voir assigner une valeur par défaut en début de processus
  3. Un signal ne peut être en entrée et sortie du processus (utilisé et affecté)

Bon exemple

```
process(a,b,c,state)
begin
  s <= (others=>'0');
  next_state <= state;
  ...
  if (c < 10) then
    s <= a + b;
    next_state <= S2;
  end if;
end process;
```

Mauvais exemple

```
process(a)
begin
  b <= a + 2;
  ...
  if (c < 10) then
    s <= a + b;
    next_state <= S2;
  end if;
end process;
```

Méthodologie

Processus séquentiel

- Un processus séquentiel doit suivre les règles suivantes :
- La liste de sensibilité ne contient que :
  L'horloge
  Le reset, si celui-ci est asynchrone
- Toutes les affectations sont faites dans le `if (rising_edge(clk_i)`
- Dans le `if (reset_i='1')` si le reset est asynchrone
- Aucune affectation permise en dehors de ces if
- Les valeurs affectées au reset doivent être fixes
  - Ne dépendent pas d'autres signaux que le reset

Méthodologie

Processus séquentiel : bon exemple

```
1  process(clk_i,reset_i) is
2  begin
3    if ( reset_i = '1' ) then
4      a <='0'; b <= (others =>'0');
5    elsif (rising_edge(clk_i)) then
6      a <= next_a;
7      if (affecte = '1') then
8        b <= c + d;
9      end if;
10   end if;
11 end process;
12
```


Méthodologie

Processus séquentiel : mauvais exemple

```
1  process(clk_i) is
2  begin
3    if ( reset_i = '1' ) then
4      a <= '0';
5      b <= (others => '0');
6    elsif (rising_edge(clk_i)) then
7      a <= next_a;
8      if (affecte = '1') then
9        b <= c + d;
10     end if;
11   else
12     b <= (others=>'0');
13   end if;
14 end process;
```


Méthodologie

Types de design

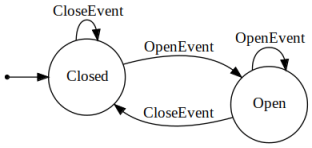
- Un design, ou module peut être :
  1. Purement combinatoire
     - Description via des processus implicites ou explicites
  2. Synchrone (ou séquentiel)
     - Combinaison de processus synchrones et combinatoires

Plan

Les machines à états

Présentation

- Un automate fini ou machine à états finis (finite state machine) est un modèle mathématique pour la modélisation de comportement des systèmes numériques
  - Son rôle, est de décrire le fonctionnement d'une machine (ou d'un objet) ayant un comportement **séquentiel**
  - Modèle graphique
  - Utilisé dans de nombreux domaines : conception de programmes informatiques, protocoles de communication, contrôle des processus, analyse linguistique, etc.



Les machines à états

Présentation

- Dans un système séquentiel, le comportement d'une machine (ou d'un objet) dépend non seulement des évènements qu'il reçoit mais aussi de ce qui s'est passé avant ces évènements.
  - Par exemple, on ne peut pas faire descendre un ascenseur s'il est déjà en bas
  - Les machines ayant un fonctionnement séquentiel passent par un nombre de situations (états) limitées et clairement identifiées. Nous disons alors que ce sont des machines à états **finis**.

Les machines à états

Les machines à états

- Une machine à états est construite de quatre éléments de base
  - Des états
  - Des évènements
  - Des transitions
  - Des actions

États

- Un état est une situation stable qui possède une certaine durée pendant laquelle un objet exécute une activité ou attend un évènement
- Un état représenté par un cercle dans lequel le nom de l'état est inscrit

Les machines à états

Transitions

- Une transition définit la réponse d'un objet à l'arrivée d'un évènement. Elle indique qu'un objet qui se trouve dans un état peut « transiter » vers un autre état en exécutant éventuellement certaines activités
- Une transition est représentée par une flèche allant de l'état de départ vers l'état d'arrivée

Évènements

- Un évènement est un fait qui déclenche le changement d'état, qui fait donc passer un objet d'un état à un autre état
- Un évènement se produit à un instant précis et est dépourvu de durée. Quand un événement est reçu, une transition peut être déclenchée et faire basculer l'objet dans un nouvel état.
- Un évènement est représenté par une expression inscrite sur une transition

Actions

- Une action consiste à envoyer un signal, à faire appel à une méthode, à affecter une valeur à un attribut, etc.

Les machines à états

Types de machines à états

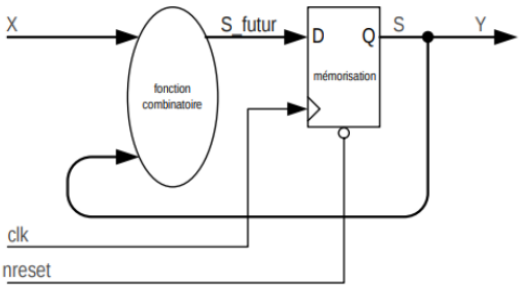
- On considère les notations suivantes :
  - X : le vecteur d'entrée
  - S : le vecteur d'état
  - Y : le vecteur de sortie
- Trois types de bases de machines à états
  1. Medvedev : le vecteur de sortie Y correspond au vecteur d'état S, i.e. $Y = S$
  2. Moore : le vecteur de sortie Y est fonction du vecteur d'état S, i.e. $Y = f(S)$
  3. Mealy : le vecteur de sortie Y est fonction du vecteur d'état S ET du vecteur d'entrée X, i.e. $Y = f(S, X)$

Plan

Machine de Medvedev

Schéma bloc

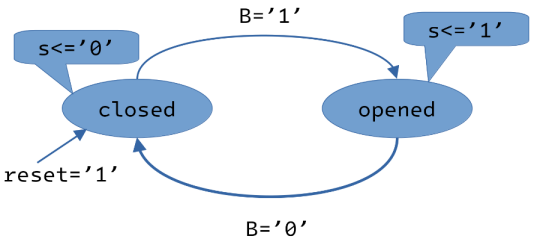
- Les sorties sont égales aux bits d'états
  - Sorties inconditionnelles



Machine de Medvedev

Exemple de traduction d'une machine à états de Medvedev

- Exemple de contrôle d'une porte avec un interrupteur
  - Lorsque l'interrupteur est à l'état '0', la porte se ferme
  - Lorsque l'interrupteur est à l'état '1', la porte s'ouvre



Machine de Medvedev

Exemple de traduction d'une machine à états de Medvedev

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  entity porte is port(
5    B,reset,clk: in std_logic;
6    s:out std_logic);
7  end entity;
```


Machine de Medvedev

Exemple de traduction d'une machine à états de Medvedev

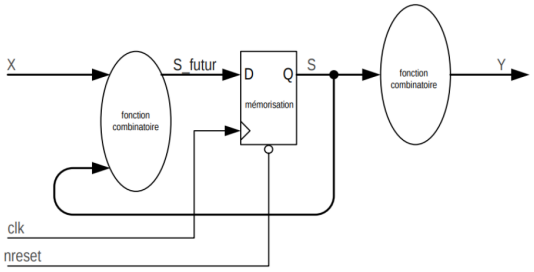
```
1  architecture sm_porte of porte is
2  type state is (closed,opened);
3  signal etat : state;
4  begin
5    s<=std_logic_vector(to_unsigned(etat'pos(etat),1))(0);
6    process(clk,reset)
7    begin
8      if (reset='1') then
9        etat<=closed;
10     elsif rising_edge(clk) then
11       case etat is
12         when closed =>
13           if B='1' then etat<=opened; end if;
14         when opened =>
15           if B='0' then etat<=closed;  end if;
16       end case;
17     end if;
18   end process;
19 end architecture;
```


Plan

Machine de Moore

Schéma bloc

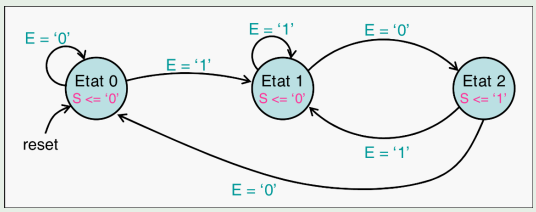
- Les sorties changent uniquement après la mise à jour de l'état interne (vecteur d'état)
  - Sorties inconditionnelles



Machine de Moore

Exemple de machine de Moore

- Une machine de Moore reconnaissant la séquence 10



Machine de Moore

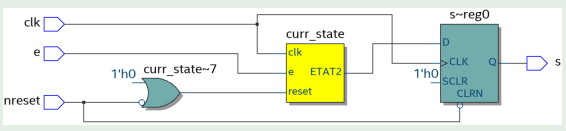
Exemple de machine de Moore : description avec 1 process

```
1  type state is (ETAT0,ETAT1,ETAT2);
2  signal curr_state : state;
3  begin
4    process(clk,reset)
5    begin
6      if (reset='1') then  curr_state<=ETAT0; s<='0';
7      elsif rising_edge(clk) then
8        case (curr_state) is
9          when ETAT0 => s<='0';
10           if (e='1') then curr_state<=ETAT1; end if;
11         when ETAT1 => s<='0';
12           if (e='0') then curr_state<=ETAT2; end if;
13         when ETAT2 => s<='1';
14           if (e='0') then curr_state<=ETAT0;
15           else curr_state<=ETAT1;
16           end if;
17       end case;
18     end if;
19   end process;
20 end architecture;
```


Machine de Moore

Exemple de machine de Moore : description avec 1 process

- Schéma RTL généré par l'outil de synthèse

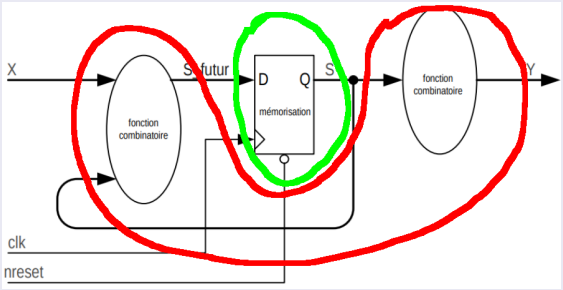


- Chronogramme des variations de signaux

Machine de Moore

Machine de Moore : description avec 2 process

- Deux process
  - Un pour la partie combinatoire
  - L'autre pour la partie séquentielle



Machine de Moore

Machine de Moore : description avec 2 process

```
1  sorties_etatfutur: process(etat,entrées)
2  begin
3          -- définition des sorties en fonction de l'état
4          ...
5          -- définition de l'état suivant en fonction de l'état
6          -- et des entrées
7          etat_futur <= ...;
8  end process;
9
10 etats: process(clk_i,rst_i)
11 begin
12         if rst_i='1' then
13                 etat <= valeurs initiales
14         elsif rising_edge(clk_i) then
15                 -- affectation des registres avec l'état futur
16                 etat <= etat_futur;
17         end if;
18 end process;
```


Machine de Moore

Exemple de machine de Moore : description avec 2 process (1/2)

```
1  type state is (ETAT0,ETAT1,ETAT2);
2  signal curr_state,next_state : state;
3  begin
4    process(clk,reset) -- Partie séquentielle
5    begin
6      if (reset='1') then
7        curr_state<=ETAT0;
8      elsif rising_edge(clk) then
9        curr_state<=next_state;
10     end if;
11   end process;
```


Machine de Moore

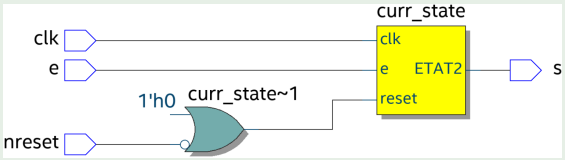
Exemple de machine de Moore : description avec 2 process (2/2)

```
1    process(e,curr_state)-- Partie combinatoire
2    begin
3      case (curr_state) is
4        when ETAT0 => s<='0';
5          if (e='1') then
6            next_state<=ETAT1;
7          else next_state<=ETAT0;
8          end if;
9        when ETAT1 => s<='0';
10         if (e='0') then
11           next_state<=ETAT2;
12         else next_state<=ETAT1;
13         end if;
14       when ETAT2 => s<='1';
15         if (e='0') then
16           next_state<=ETAT0;
17         else next_state<=ETAT1;
18         end if;
19     end case;
20   end process;
21 end architecture;
```

Machine de Moore

Exemple de machine de Moore : description avec 2 process

- Schéma RTL généré par l'outil de synthèse



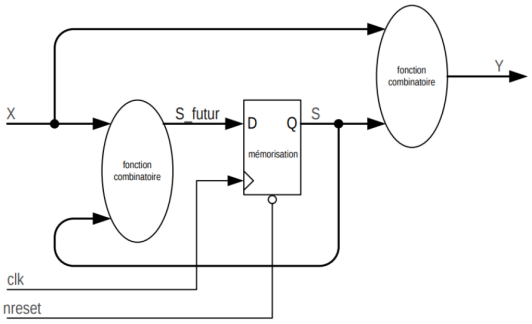
- Chronogramme des variations de signaux

Plan

Machine de Mealy

Schéma bloc

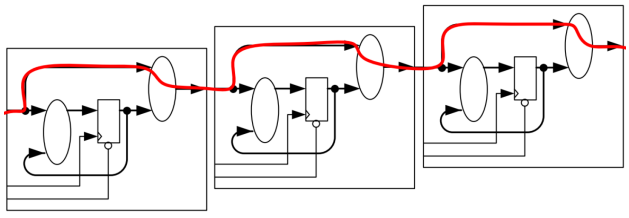
- Les sorties changent immédiatement avec les entrées et après la mise à jours de l'état interne (vecteur d'état)
  - Sorties conditionnelles



Machine de Mealy

Problème de chainage des machines de Mealy

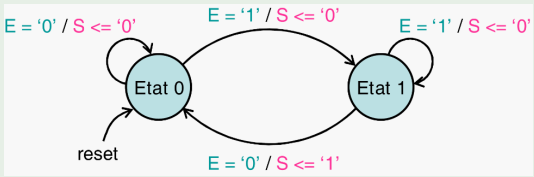
- En chaînant des machines de Mealy, il faut éviter de propager les signaux combinatoires ! ! !



Machine de Mealy

Exemple de machine de Mealy

- Une machine de Mealy reconnaissant la séquence 10



Machine de Mealy

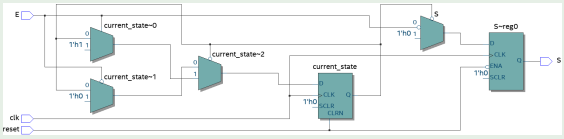
Machine de Mealy : description avec 1 process

```
1  architecture sm_mealy of ex_mealy is
2  type state is (Etat0,Etat1);
3  signal current_state : state;
4  begin
5    process(clk,reset)
6    begin
7      if reset='1' then current_state<=Etat0;
8      elsif rising_edge(clk) then
9        case current_state is
10         when Etat0 =>
11           if E='0' then S<='0';
12           else S<='0'; current_state<=Etat1;
13           end if;
14         when Etat1 =>
15           if E='0' then S<='1'; current_state<=Etat0;
16           else S<='0';
17           end if;
18       end case;
19     end if;
20   end process;
21 end architecture;
```


Machine de Mealy

Machine de Mealy : description avec 1 process

- Schéma RTL généré par l'outil de synthèse

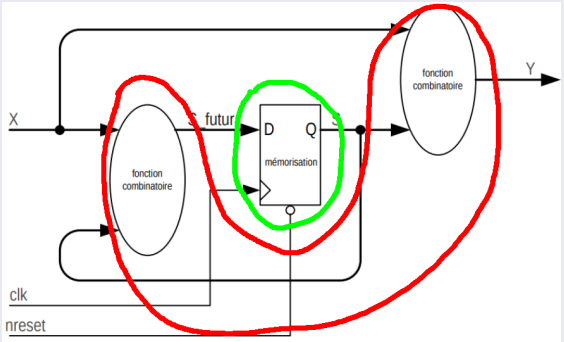


- Chronogramme des variations de signaux

Machine de Mealy

Machine de Mealy : description avec 2 process

- Deux process
  - Un pour la partie combinatoire
  - L'autre pour la partie séquentielle



Machine de Mealy

Machine de Mealy : description avec 2 process

```
1  sorties_n_etats: process(etat,entrees) -- Partie combinatoire
2  begin
3    -- définition des sorties en fonction de l'état
4    -- et des entrées
5    ...
6    -- définition de l'état suivant en fonction de l'état
7    -- et des entrées
8    etat_futur<= ...;
9  end process;
10
11 etats: process(clk,rst) -- Partie séquentielle
12 begin
13   if rst='1' then
14     etat <= valeurs initiales
15   elsif rising_edge(clk) then
16     etat <= etat_futur;
17   end if;
18 end process;
```


Machine de Mealy

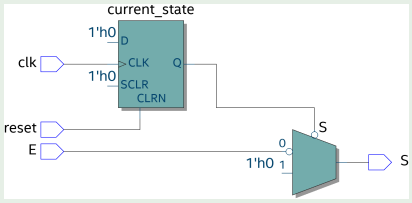
Exemple de description avec 2 process d'une machine de Mealy

```
1  type state is (Etat0,Etat1);
2  signal current_state,next_state : state;
3  begin
4    process(clk,reset)
5    begin
6      if reset='1' then current_state<=Etat0;
7      elsif rising_edge(clk) then current_state<=next_state;
8      end if;
9    end process;
10   state_machine:process(current_state,E)
11   begin
12     case current_state is
13       when Etat0 => if E='0' then S<='0';
14             else S<='0';
15           end if;
16       when Etat1 => if E='0' then S<='1';
17             else S<='0';
18           end if;
19     end case;
20   end process;
21 end architecture;
```


Machine de Mealy

Machine de Mealy : description avec 2 process

- Schéma RTL généré par l'outil de synthèse



- Chronogramme des variations de signaux

MERCI POUR VOTRE ATTENTION

Questions ?