

# ARCHITECTURE LOGIQUE PROGRAMMABLE

## Exposé 05

Cne. Brinsi Riadh

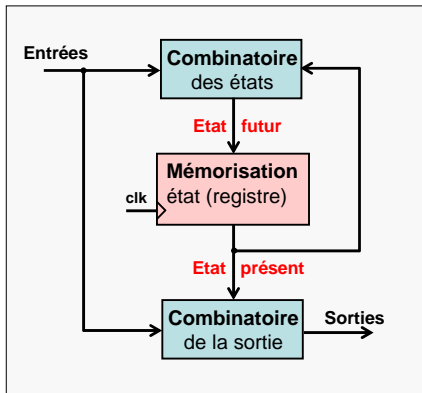
Académie militaire F.J

*br.riadh@gmail.com*

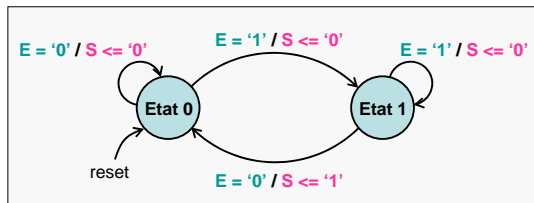
December 15, 2015

## Machine de Mealy.

- L'état futur est calculé à partir des entrées et de l'état présent.
  - Les sorties d'une machine de Mealy dépendent de l'état présent et des entrées.
  - Mémorisation synchrone des états (càd sur un front d'horloge).
  - La sortie dépend directement de l'entrée et ceci indépendamment de l'horloge (clk).
- ⇒ Sortie asynchrone.
- Nombre d'états plus réduit que pour une machine de Moore.
  - Il est possible de resynchroniser la sortie au besoin en ajoutant des bascules D.

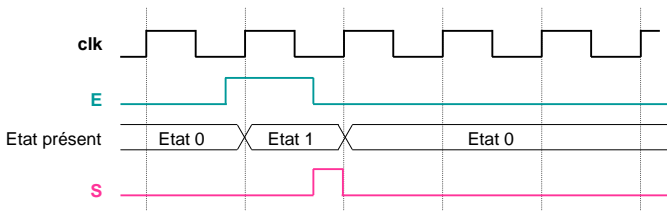


Exemple : Machine de Mealy reconnaissant la séquence 10



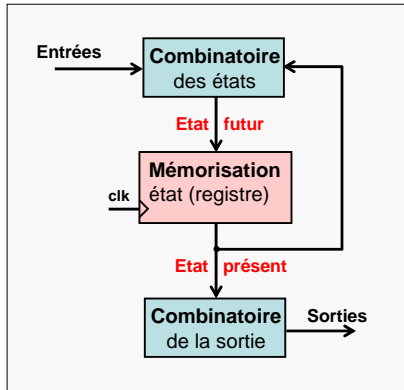
$E = '1' / S \leq '0'$

Condition de validation de la transition      Affection de la valeur '0' à la sortie

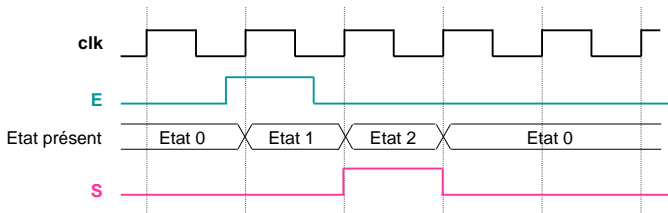
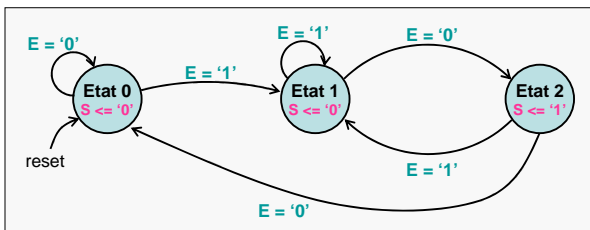


## Machine de Moore.

- Les sorties d'une machine de Moore dépendent de l'état présent (synchrones, elles changent sur un front d'horloge).
- L'état futur est calculé à partir des entrées et de l'état présent.

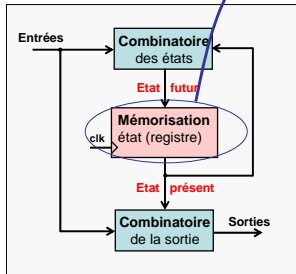


Exemple : Machine de Moore reconnaissant la séquence 10



# Ecriture VHDL - Machine de Mealy

## Description avec 3 process



- Un process séquentiel de mise à jour de l'état présent par l'état futur sur les fronts montant d'horloge (reset asynchrone inclus) :

```
type Etat is (Etat0, Etat1);  
Signal Etat_present, Etat_futur : Etat := Etat0;
```

```
Sequentiel_maj_etat : process (clk, reset)  
begin
```

```
    if reset = '0' then
```

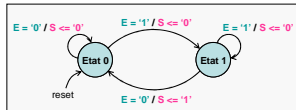
```
        Etat_present <= Etat0;
```

```
    elsif clk'event and clk = '1' then
```

```
        Etat_present <= Etat_futur;
```

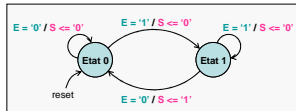
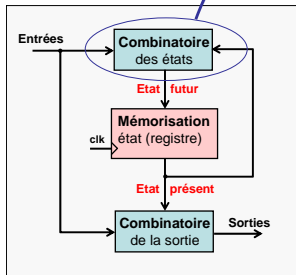
```
    end if;
```

```
end process Sequentiel_maj_etat;
```



# Ecriture VHDL - Machine de Mealy

## Description avec 3 process



- Un process combinatoire de calcul de l'état futur à partir des entrées et de l'état présent :

```
Combinatoire_etats : process (E, Etat_present)
begin
```

```
  case Etat_present is
```

```
    when Etat0 => if E = '1' then
                     Etat_futur <= Etat1;
                   else
                     Etat_futur <= Etat0;
                   end if;
```

```
    when Etat1 => if E = '1' then
                     Etat_futur <= Etat1;
                   else
                     Etat_futur <= Etat0;
                   end if;
```

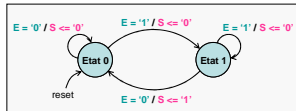
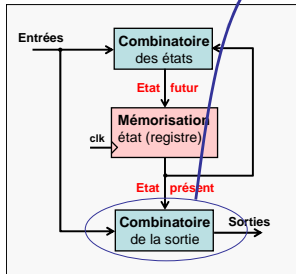
```
  end case;
```

```
end process Combinatoire_etats;
```

# Ecriture VHDL - Machine de Mealy

## Description avec 3 process

- Un process combinatoire de calcul des sorties à partir des entrées et de l'état présent :



```
Combinatoire_sorties : process (E, Etat_present)
begin
```

```
  case Etat_present is
```

```
    when Etat0 => if E = '1' then
                     S <= '0';
                   else
                     S <= '0';
                   end if;
```

```
    when Etat1 => if E = '0' then
                     S <= '1';
                   else
                     S <= '0';
                   end if;
```

```
  end case;
```

```
end process Combinatoire_sorties;
```



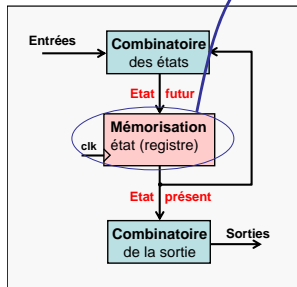
## Description avec 2 process

- Les 2 process combinatoires possèdent la même liste de sensibilité, ils peuvent donc être regroupés en un seul process afin d'abrégier l'écriture.
- ⇒ 2 process = 1 process séquentiel + 1 process combinatoire

## Description avec 1 process

- Description la plus compacte en utilisant une variable pour l'état (en lieu et place des 2 signaux).
  - A pour effet de resynchroniser la sortie de façon implicite.
  - Cependant perte de lisibilité lors de l'écriture. Alors que cette description n'apporte rien en terme de résultat de synthèse par rapport à une description 2 process.
- ⇒ À éviter (pt de vue personnel cependant ...).

## Description avec 3 process



- Un process séquentiel de mise à jour de l'état présent par l'état futur sur les fronts montant d'horloge (reset asynchrone inclus) :

```
type Etat is (Etat0, Etat1, Etat2);  
Signal Etat_present, Etat_futur : Etat := Etat0;
```

```
Sequentiel_maj_etat : process (clk, reset)  
begin
```

```
    if reset = '0' then
```

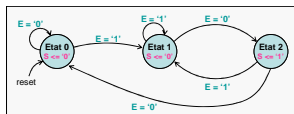
```
        Etat_present <= Etat0;
```

```
    elsif clk'event and clk = '1' then
```

```
        Etat_present <= Etat_futur;
```

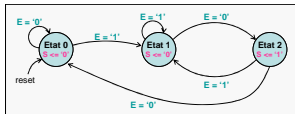
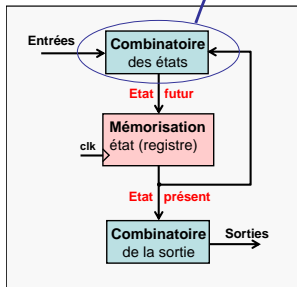
```
    end if;
```

```
end process Sequentiel_maj_etat;
```



## Description avec 3 process

- Un process combinatoire de calcul de l'état futur à partir des entrées et de l'état présent :



```
Combinatoire_etats : process (E, Etat_present)
begin
```

```
  case Etat_present is
```

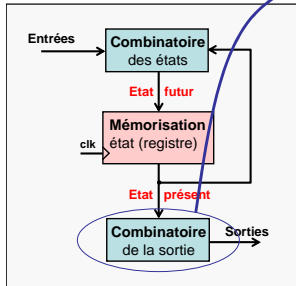
```
    when Etat0 => if E = '1' then
                     Etat_futur <= Etat1;
                   else
                     Etat_futur <= Etat0;
                   end if;
```

```
    when Etat1 => if E = '0' then
                     Etat_futur <= Etat2;
                   else
                     Etat_futur <= Etat1;
                   end if;
```

```
    when Etat2 => if E = '1' then
                     Etat_futur <= Etat1;
                   else
                     Etat_futur <= Etat0;
                   end if;
```

```
  end case;
end process Combinatoire_etats;
```

## Description avec 3 process

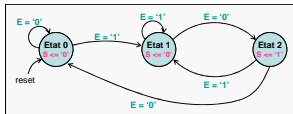


- Un process combinatoire de calcul des sorties à partir de l'état présent :

```
Combinatoire_sorties : process (Etat_present)  
begin
```

```
  case Etat_present is  
    when Etat0 => S <= '0';  
  
    when Etat1 => S <= '0';  
  
    when Etat2 => S <= '1';  
  end case;
```

```
end process Combinatoire_sorties;
```



## Description avec 2 process

- Les 2 process combinatoires possèdent des listes de sensibilité « compatibles », ils peuvent donc être regroupés en un seul process afin d'abrégier l'écriture.
- ⇒ 2 process = 1 process séquentiel + 1 process combinatoire

## Description avec 1 process

- Description la plus compacte en utilisant une variable pour l'état (en lieu et place des 2 signaux).
  - Cependant perte de lisibilité lors de l'écriture. Alors que cette description n'apporte rien en terme de résultat de synthèse par rapport à une description 2 process.
- ⇒ À éviter (pt de vue personnel cependant ...).