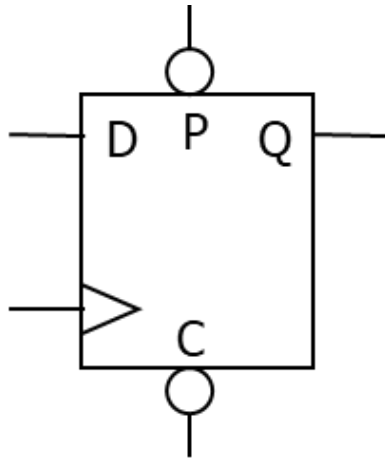


# Séance 3

# Circuits séquentiels en VHDL

## Bascule D :

**Circuit synchrone : la sortie n'évolue que sur un front d'horloge**

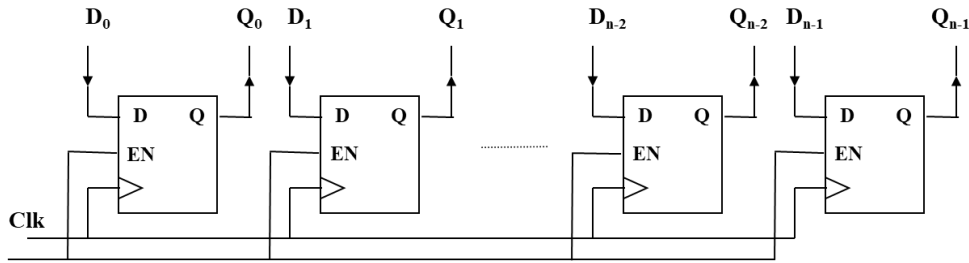


Avec entrée preset synchrone  
et clear asynchrone

```
entity dff is
port (CLK, P, C: in STD_LOGIC; -- commandes
      D:          in STD_LOGIC; -- entrée
      Q:          out STD_LOGIC); -- sortie
end dff;
```

```
architecture bhv of dff is
begin
  process (CLK, C)
  begin
    if C = '0' then
      Q <= '0';
    elsif CLK'EVENT AND CLK = '1' then
      if P = '0' then
        Q <= '1';
      else
        Q <= D;
      end if;
    end if;
  end process;
end bhv;
```

# Registres en VHDL



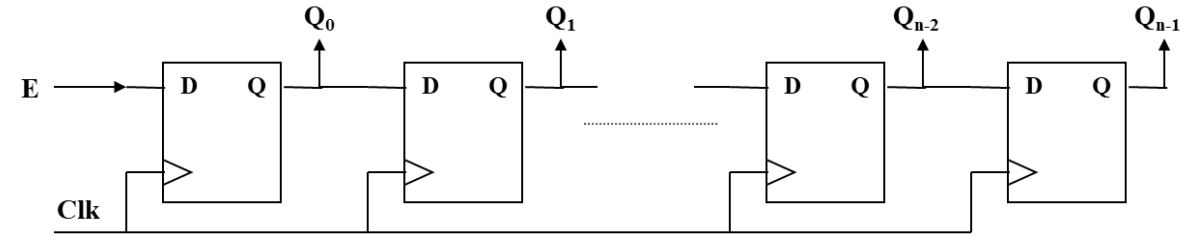
**Registre en VHDL :**

```
entity registre is
port (
    CLK, load : in std_logic;
    D : in std_logic_vector(15 downto 0);
    Q : out std_logic_vector(15 downto 0)
);
end registre;

architecture bhv of registre is
begin

    process (CLK)
    begin
        if CLK'EVENT AND CLK = '1' then
            if load = '1' then
                Q <= D;
            end if;
        end if;
    end process;

end bhv;
```



**Registre à décalage en VHDL :**

```
entity regdec is
port (
    CLK, E : in std_logic;
    Q : out std_logic_vector(15 downto 0)
);
end regdec;

architecture bhv of regdec is

    signal Qint : std_logic_vector(15 downto 0);

begin

    Q <= Qint;

    process (CLK)
    begin
        if CLK'EVENT AND CLK = '1' then
            Qint <= Qint(14 downto 0) & E;
        end if;
    end process;

end bhv;
```

# Compteur en VHDL

**Compteur  
modulo 7 en  
VHDL avec  
remise à zéro  
asynchrone :**

```
entity compteur_mod7 is
port (CLK, raz : in std_logic;
      cpt : out std_logic_vector(2 downto 0));
end compteur_mod7 ;

architecture bhv of compteur_mod7 is
  signal cpt_int : unsigned(2 downto 0);
begin

  cpt <= std_logic_vector(cpt_int);
  process (CLK,raz)
  begin
    if raz = '1' then
      cpt_int <= (others => '0'); -- met à 0 tous les bits
    elsif CLK'EVENT AND CLK = '1' then
      if cpt_int = to_unsigned(6,3) then
        cpt_int <= to_unsigned(0,3); -- transforme l'entier 0 en unsigned sur 3 bits
      else
        cpt_int <= cpt_int + 1 ;
      end if;
    end if;
  end process;

end bhv;
```

# Séance 4

# Machines à états finis en VHDL

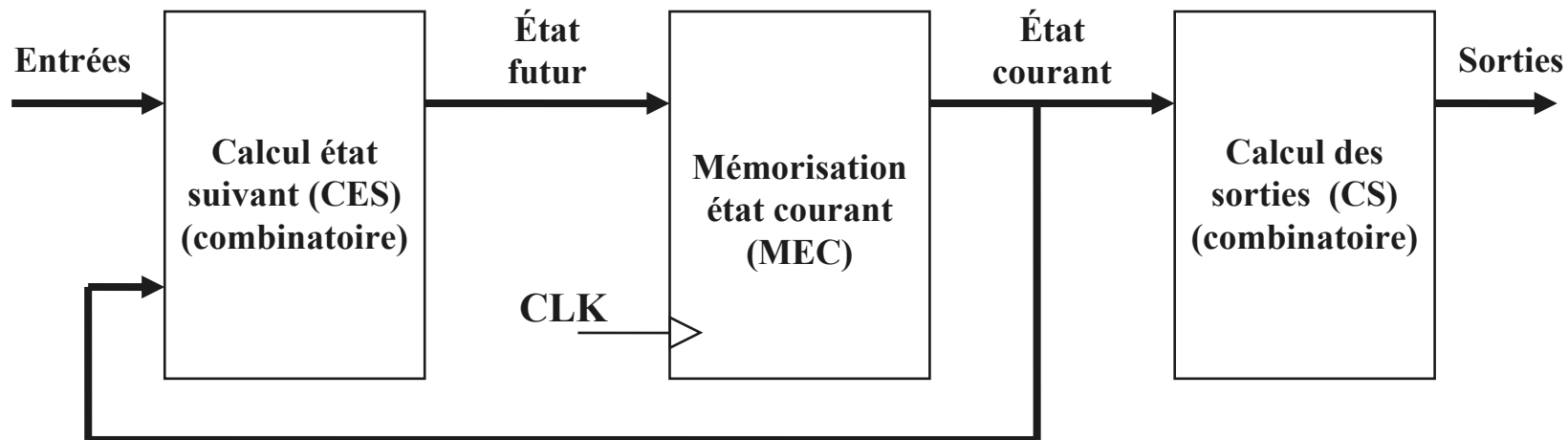
## Définition :

Système à base de bascules synchrones permettant de réaliser n'importe quel processus séquentiel (compteur, contrôle ...).

Les bascules permettent de mémoriser l'état courant du processus

Des parties combinatoires permettent d'en déduire l'état futur et la valeur des sorties

**Machine de Moore :** Les sorties ne dépendent que de l'état courant  
L'état futur dépend de l'état courant et des entrées



# Machines à états finis en VHDL

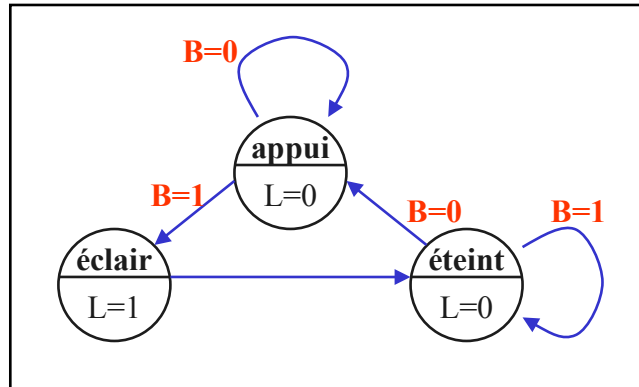
**Exemple :** fonction flash

Un appui sur B provoque un éclair pendant un coup d'horloge après relâchement de B

**Réalisation sur machine de Moore :**

- **Spécification :**
  - une entrée B qui vaut 0 quand on appuie, 1 au repos
  - une sortie L (=1 quand allumée)

- **Diagramme d'états :**



- **Codage des états :** détermine le nombre de bascules nécessaires  
éteint  $\Leftrightarrow$  "00" ; appui  $\Leftrightarrow$  "01" ; éclair  $\Leftrightarrow$  "10" 2 bascules ( $Q_1Q_0$ )

# Machines à états finis en VHDL

**Machines de Moore : fonction flash**  
**Représentation VHDL :**

```
entity flash is
port (CLK, RST, B: in std_logic;
      L: out std_logic);
end flash;

architecture fsm_moore of flash is
type état is (éteint,appui,éclair);
signal ecur, esuv: état;
begin
CES: process(ecur,B) -- combinatoire
begin
case ecur is
when éteint =>
if B = '0' then esuv <= appui; else
esuv <= éteint; end if;
when appui =>
if B = '1' then esuv <= éclair; else
esuv <= appui; end if;
when éclair => esuv <= éteint;
when others => esuv <= éteint;
end case;
end process CES;
```

**Utilisation de 3 process pour représenter  
les trois blocs (CES,MEC,CS)**

```
MEC: process(RST,CLK)
begin
if RST = '1' then
ecur <= éteint;
elsif CLK'event AND CLK='1' then
ecur <= esuv;
end if;
end process MEC;

CS: process (ecur) -- combinatoire
begin
case ecur is
when éteint => L <= '0';
when appui => L <= '0';
when éclair => L <= '1';
when others => L <= '0';
end case;
end process CS;

end fsm_moore;
```