ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Enseignes et afficheurs à LED

Circuits logiques séquentiels en VHDL

Dr. Mamadou Lamine NDIAYE

Circuits séquentiels en VHDL

Dr. Mamadou Lamine NDIAYE

Circuits séquentiels en VHDL

- Les Process
- Les instructions de contrôle
- Les bascules
- Les compteurs
- Les régistres

Instructions séquentielles en VHDL

Logique séquentielle : Mot clè PROCESS

- Modélisation des comportement procéduraux

Les Process

Les instructions séquentielles du VHDL sont très évoluées à l'image des langages de haut niveau :

- Utilisation d'un process ou un sous-programme
- Déroulement **séquentiel** (ordre d'écriture des instructions)
- Possibilité d'exécution de plusieurs Process en **parallèle**

Les Process

Les instructions séquentielles du VHDL sont très évoluées à l'image des langages de haut niveau :

- Utilisation d'un process ou un sous-programme
- Déroulement **séquentiel** (ordre d'écriture des instructions)
- Possibilité d'exécution de plusieurs Process en **parallèle**
- Un Process peut contenir des parties **combinatoires** et des parties **séquentielles**
- Un Process est activé lors d'un changement d'état d'un des signaux de la liste de sensibilité
- Les instructions utilisables dans un Process sont **spécifiques**
- Les signaux sont mis à jour uniquement à la fin du Process

Les Process

Les règles de fonctionnement :

- Boucle infinie
- Synchronisation par des points d'arrêt

SYNTHAXE

```
Process (S1, S2)
Begin

…….

end process;
```

```
Process
Begin
…….
Wait ……..;
…….
end process;
```

Les Process

L'instruction séquentielle **wait** peut prendre plusieurs formes :

- wait on signal
- wait for temps
- wait until conditions
- wait on signal until conditions for temps

SYNTHAXE

```
-- liste de sensibilité
wait on S1, S2;
```

```
-- délai
wait for 100ns;
```

```
-- conditions
wait until ck = '1';
```

```
-- conditions
wait on S1, S2 until ck = '1';
```

```
-- forme générale
wait on S1, S2 until ck = '1' for 100ns;
```

Instructions de contrôle

- Instruction IF

```
if      expression1  then    action1 ;
elsif   expression2  then    action2 ;
        ……...
else                         action3 ;
end if;
```

  - If … then …; [elsif …then] ; [else …] ; end if;
  - Toute instruction if doit se terminer par un end if;

Instructions de contrôle

- Instruction CASE…IS

```
CASE    selecteur  IS
WHEN    condition1     =>    instructions 1;
WHEN    condition2     =>    instructions 2;
          ……..
WHEN OTHERS            =>    instructions N
END CASE;
```

- Permet de sélectionner une séquence d'instructions en fonction de la valeur d'une expression.
- Tous les cas doivent être traités, on utilisera ainsi la directive when others pour lister les différents cas possibles.
- Souvent utilisée pour la description de machines d'état ou toute fonction de type table de vérité.

Instructions de contrôle

- Instructions de boucle
  - Mot clé **LOOP**.

```
-- boucle for
etiquette: FOR i IN 1 TO 100 LOOP -- la variable de boucle est i de 1 à 100
-- instruction répétitive
……..
END LOOP etiquette;
```

```
-- boucle while
etiquette: WHILE conditions  LOOP – boucle tant que
-- instruction répétitive
……..
END LOOP etiquette;
```

```
-- boocle générale
etiquette:  LOOP – boucle infinie
-- instruction répétitive
……..
END LOOP etiquette;
```

Instructions de contrôle

- Instructions de boucle
  - Mot clé LOOP.
  - Possibilité d'utiliser Next
  - Possibilité d'utiliser Exit

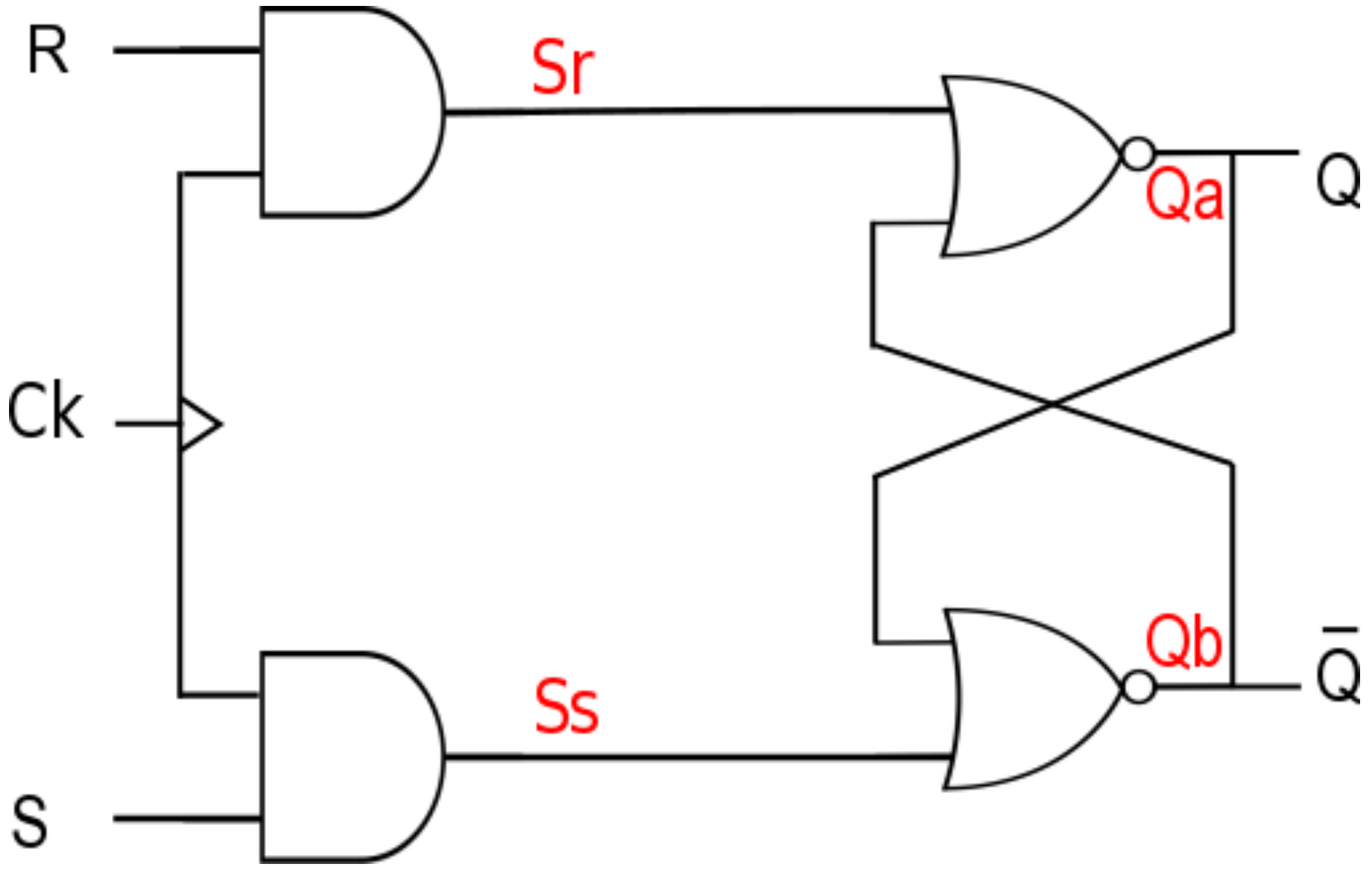
```
-- boucle for
etiquette: FOR i IN 1 TO 100 LOOP -- la variable de boucle est i de 1 à 100
-- instruction répétitive
……..
END LOOP etiquette;
```

```
-- boucle while
etiquette: WHILE conditions  LOOP – boucle tant que
-- instruction répétitive
……..
END LOOP etiquette;
```

```
-- boocle générale
etiquette:  LOOP – boucle infinie
-- instruction répétitive
……..
END LOOP etiquette;
```

Synthèse des bascules

- Bascule RS



Synthèse des bascules

- Bascule RS
  - Activé sur front montant de l'horloge (Ck)

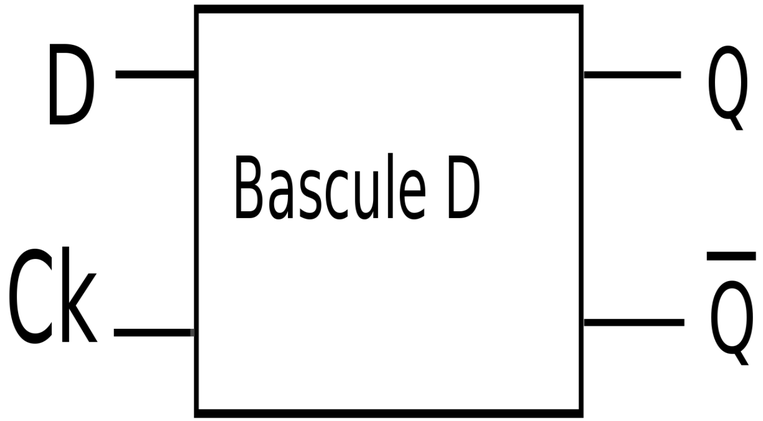
```
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY BasculeRS IS
PORT (Ck, R, S : IN STD_LOGIC;
                 Q : OUT  STD_LOGIC);
END BasculeRS;
```

```
ARCHITECTURE Archit OF BasculeRS IS
SIGNAL Sr, Ss, Qa, Qb: STD_LOGIC ;
BEGIN
        PROCESS(Ck, R, S)   -- liste de sensibilité
BEGIN
       IF Ck'event and Ck = '1' then  -- validation du front montant
        --  rising_edge(Ck)  front montant
        -- falling_edge(Ck)  descendant
        Sr <= R;
        Ss <= S;
        Qa <= NOT ( Sr OR Qb);
        Qb <= NOT ( Ss OR Qa);
        END IF;
        END PROCESS;
        Q <= Qa;
END Archit;
```

Synthèse des bascules

- Bascule D
  - La sortie Q prend l'état de l'entrée D sur front montant de l'horloge (Ck)



Synthèse des bascules

- Bascule D
  - La sortie Q prend l'état de l'entrée D sur front montant de l'horloge (Ck)

```
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY BasculeD IS
PORT (Ck, D : IN     STD_LOGIC;
              Q : OUT  STD_LOGIC);
END BasculeD;
```

```
ARCHITECTURE Structural OF BasculeD IS
SIGNAL Qa: STD_LOGIC ;
BEGIN
        PROCESS(Ck, D)   -- liste de sensibilité
BEGIN
        IF RISING_EDGE(Ck) then  -- validation du front montant
        Qa <= D;
        ELSE Qa <=Qa ;
        END IF;
        END PROCESS;
        Q <= Qa;
END Structural;
```

Synthèse des bascules

- Bascule D avec entrées de forçage SET et RESET

```
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY BasculeD IS
PORT (Ck, D, R, S : IN      STD_LOGIC;
      Q           : OUT     STD_LOGIC);
END BasculeD;
```

```
ARCHITECTURE Structural OF BasculeD IS
SIGNAL Qa: STD_LOGIC ;
BEGIN
        PROCESS(Ck, D, R, S)   -- liste de sensibilité
BEGIN
  IF R = '1'  THEN      Qa <= '0';
        ELSIF S = '1' THEN Qa <= '1';
        ELSIF Ck'EVENT AND Ck = '1' THEN  Qa <= D;
        -- validation du front montant
        ELSE Qa <=Qa ;
 END IF;
END PROCESS;
       Q <= Qa;
END Structural;
```

Synthèse des bascules

- **Bascule JK**

J	K	Q
0	0	Inchangée
0	1	0
1	0	1
1	1	Opposée

Synthèse des bascules

- **Bascule JK**

```
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY BasculeJK IS
Port ( Ck, J, K: in std_logic;
              Q : out std_logic);
End BasculeJK;
```

```
ARCHITECTURE Structural OF BasculeJK IS
SIGNAL  Qa : std_logic;
BEGIN
PROCESS (Ck, J, K)
BEGIN
        IF Ck'event and Ck='1'          then
                IF J='0' and K='0'      then   Qa <= Qa;
                ELSIF J='0' and K='1'   then   Qa <= '0' ;
                ELSIF J='1' and K='0'   then   Qa <= '1' ;
                ELSE                           Qa <= (NOT) Qa ;
                END IF;
        END IF;
END PROCESS;
        Q <= Qa;
END Strctural;
```

Synthèse des compteurs

- Compteur modulo 10

Synthèse des compteurs

- **Compteur modulo 10**

```
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY Compteur IS
PORT (Ck, R : IN    STD_LOGIC;
Q: OUT STD_LOGIC_VECTOR (3 downto 0));
END Compteur;
```

```
ARCHITECTURE Structural OF Compteur IS
SIGNAL Qa: STD_LOGIC_VECTOR (3 downto 0);
BEGIN
        PROCESS(Ck, R)   -- liste de sensibilité
BEGIN
 IF R = '0' then
        Qa <=  "0000";
 ELSIF Ck'event and Ck = '1' then   -- validation du front montant
        Qa <= STD_LOGIC_VECTOR (UNSIGNED(Qa)+ 1);
        IF   Qa="1001" THEN  Qa <= "0000";
         END IF;
 END IF;
END PROCESS;
        Q <= Qa;
END Structural;
```

Synthèse des compteurs

- Compteur modulo 10
- Fréquence du comptage

Synthèse des compteurs

- Compteur modulo 10
- Fréquence du comptage

```
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY Compteur IS
PORT (Ck, R : IN        STD_LOGIC;
Qh: OUT         STD_LOGIC_VECTOR (23 downto 0);
Q: OUT          STD_LOGIC_VECTOR (3 downto 0));
END Compteur;
```

```
ARCHITECTURE Structural OF Compteur IS
SIGNAL Qa: STD_LOGIC_VECTOR (3 downto 0) ;
SIGNAL Qha: STD_LOGIC_VECTOR (23 downto 0) ;
BEGIN
        PROCESS(Ck, R)   -- liste de sensibilité
BEGIN
 IF R = '1'  THEN   Qa <=  "0000";
ELSIF (Ck'event and Ck = '1') THEN
Qha <= STD_LOGIC_VECTOR (UNSIGNED(Qha)+ 1);
        IF (Qha ="100110001001011010000000") THEN
        IF   Qa="1010" THEN
        Qa <= "0000";
        ELSE
        Qa <= STD_LOGIC_VECTOR (UNSIGNED( Qa) + 1);
         END IF;
         ELSE     Qa <= Qa;
         END IF;
 END IF;
END PROCESS;
        Q <= Qa;
END Structural;
```

Synthèse des compteurs

- Compteur modulo N

Synthèse des compteurs

- **Compteur modulo N**

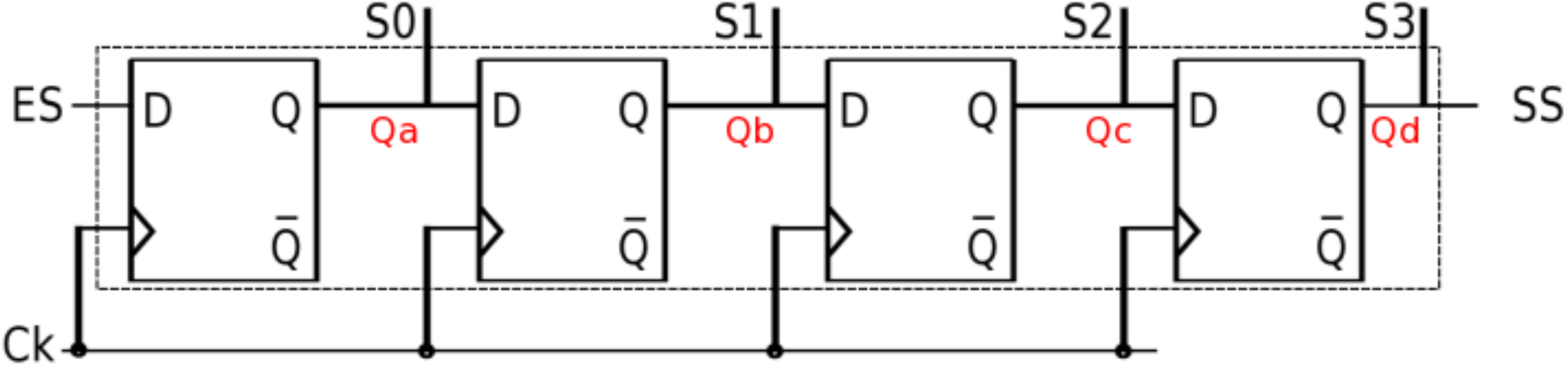
```
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY Compteur IS
PORT (Ck, R : IN        STD_LOGIC;
Q0: OUT                 STD_LOGIC_VECTOR (3 downto 0);
Q1: OUT                 STD_LOGIC_VECTOR (3 downto 0);
Q2: OUT                 STD_LOGIC_VECTOR (3 downto 0));
END Compteur;
```

```
ARCHITECTURE Structural OF Compteur IS
SIGNAL Qa0, Qa1, Qa2: STD_LOGIC_VECTOR (3 downto 0) ;
BEGIN
PROCESS(Ck, R)   -- liste de sensibilité
        VARIABLE Qh : INTEGER RANGE 0 TO 50000000;
        BEGIN
IF R = '1'  THEN
        Qa0 <=  "0000"; Qa1 <=  "0000"; Qa2 <=  "0000";
ELSIF (Ck'event and Ck = '1') THEN        Qh := Qh + 1;
  IF (Qh = 50000000) THEN      Qa0 <= Qa0 + 1;
          IF Qa0 ="1001" then    Qa1 <= Qa1 + 1;      Qa0 <= "0000";
          IF Qa1 ="1001" then    Qa2 <= Qa2 + 1;      Qa1 <= "0000";
           END IF;
          END IF;
    ELSE     Qa0 <= Qa0; Qa1 <= Qa1; Qa2 <= Qa2;
   END IF;
 END IF;
END PROCESS;
  Q0 <= Qa0; Q1 <= Qa1; Q2 <= Qa2;
END Structural;
```

Synthèse des regitres

- Registre à décalage série parallèle



Synthèse des regitres

- Registre à décalage série parallèle

```
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY BasculeD IS
PORT (Ck, D : IN     STD_LOGIC;
              Q : OUT  STD_LOGIC);
END BasculeD;
```

```
ARCHITECTURE Structural OF BasculeD IS
SIGNAL Qa: STD_LOGIC ;
BEGIN
        PROCESS(Ck, D)   -- liste de sensibilité
BEGIN
       IF Ck'event and Ck = '1' then  -- validation du front montant
         Qa <= D;
         ELSE Qa <=Qa ;
         END IF;
         END PROCESS;
         Q <= Qa;
END Structural;
```

Synthèse des regitres

- **Registre à décalage série parallèle**

```
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY RegistreSP IS
PORT (Ck, ES : IN   STD_LOGIC;
          SS : OUT     STD_LOGIC;
          SP : OUT     STD_LOGIC_VECTOR (0 TO 3));
END RegistreSP;
```

```
ARCHITECTURE Structural OF RegistreSP IS
COMPONENT BasculeD
PORT (Ck, D : IN     STD_LOGIC;
            Q : OUT  STD_LOGIC);
END COMPONENT ;

SIGNAL Qa, Qb, Qc, Qd: STD_LOGIC ;
BEGIN
 U1:   BasculeD PORT MAP (Ck, ES, Qa);
 U2:   BasculeD PORT MAP (Ck, Qa, Qb);
 U3:   BasculeD PORT MAP (Ck, Qb, Qc);
 U4:   BasculeD PORT MAP (Ck, Qc, Qd);
       SP(0) <= Qa; SP(1) <= Qb; SP(2) <= Qc; SP(3) <= Qd;
END Structural;
```

Logique séquentielle en VHDL

- Les Process
- Les instructions de contrôle
- Les bascules
- Les compteurs
- Les régistres