

Université M'hamed BOUDIAF de M'SILA

Faculté de Technologie

Département Electronique

TP n°3 Electronique Numérique Avancée

Développement d'un premier exemple de circuit : compteur
décimal.

1. Objectifs

Les compteurs sont des éléments très utiles en VHDL. Ils permettent de gérer tout ce qui est temporisation et évidemment le comptage.

Dans ce TP, nous trouverons du code VHDL pour 4 genres de compteurs. Ces compteurs sont utilisés pour des applications différentes mais sont basés sur la même structure. Inspirez-vous de ces concepts pour générer votre propre code VHDL.

2. Manipulations

a) Compteur simple

Dans ce premier exemple, nous allons créer un compteur allant de 0 à 9 et en même temps, nous allons utiliser ce signal pour générer une horloge qui est 10 fois plus lente que celle à l'entrée. Pour ce faire, il faut commencer par réaliser que, pour compter jusqu'à 9, il faut 4 bits). Par la suite, on devrait aussi savoir qu'un compteur est composé de flip flops et de logique. Il faut donc avoir un processus séquentiel (avec clock).

Un compteur devrait augmenter son compte à chaque front montant d'horloge SAUF quand il arrive à 9. Quand il arrive à 9, il doit recommencer à 0. La partie principale du programme devrait donc être :

```
PROCESS (clk)
  BEGIN
    IF clk'EVENT AND clk = '1' THEN
      IF s_compt_sortie >= 9 THEN
s_compt_sortie <= "0000";
      ELSE
s_compt_sortie <= s_compt_sortie + 1;
      END IF;
    END IF;
  END PROCESS;
```

Il est à remarquer que *s_compt_sortie* n'est pas le nom du signal de sortie. VHDL n'accepte pas qu'on LISE un signal de sortie. Au lieu de le lire, nous allons créer un fil interne qui sera connecté à cette sortie. Nous avons le droit de lire et d'écrire aux fils internes. Donc, le compteur se fera avec le fil interne. Par la suite, nous connecterons ce fil interne à la sortie.

Pour générer l'horloge de sortie, il faut qu'elle garde '1' la moitié du temps de comptage à dix et générer '0' durant l'autre moitié. Pour cette raison, nous avons :

```
horloge_sortie <= '0' WHEN s_compt_sortie < 5
  ELSE '1';
```

Le code final (fichier *Compte_simple.vhd*) ressemblera à ceci :

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
  ENTITY compteur IS
    PORT (
clk : IN STD_LOGIC;
compt_sortie : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
horloge_sortie : OUT STD_LOGIC
    );
  END compteur;
  ARCHITECTURE rtl OF compteur IS
```

```

    SIGNAL s_compt_sortie : STD_LOGIC_VECTOR(3 DOWNT0 0) := "0000";
BEGIN
    PROCESS (clk)
    BEGIN
        IF clk'EVENT AND clk = '1' THEN
            IF s_compt_sortie >= 9 THEN
s_compt_sortie <= "0000";
            ELSE
s_compt_sortie <= s_compt_sortie + 1;
            END IF;
        END IF;
    END PROCESS;
compt_sortie <= s_compt_sortie;
horloge_sortie <= '0' WHEN s_compt_sortie < 5
ELSE
'1';
END;

```

- Créer un projet dans ise8.2 et ajouter lui le fichier Compte_simple.vhd et test1_tb.vhd
- Simuler le comportement du compteur avec Isim du ise (se reporter au TP 1).

b) Compteur avec autorisation

Prenons maintenant un deuxième exemple d'un compteur. Imaginons cette fois-ci qu'on veuille contrôler le compteur avec un signal externe. Nous voulons que le compteur incrémente seulement quand un bouton est pesé.

Quand on relâche le bouton, on aimerait que le compte arrête.

Pour ce faire, il faut procéder de la même manière que l'exemple précédent. La différence, cependant, est qu'il y a une condition de plus pour que le compteur fonctionne : « A chaque front montant d'horloge, SI LE BOUTON EST PESE, tu peux agir comme un compteur. »

Le process deviendrait donc ceci :

```

PROCESS (clk)
BEGIN
    IF clk'EVENT AND clk = '1' THEN
        IF entree = '1' THEN
            IF s_compt_sortie >= 9 THEN
s_compt_sortie <= "0000";
            ELSE
s_compt_sortie <= s_compt_sortie + 1;
            END IF;
        END IF;
    END IF;
END PROCESS;

```

Le code complet (fichier Compte_autor.vhd) ressemble à ceci:

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY compteur IS
    PORT (
clk : IN STD_LOGIC;
entree : IN STD_LOGIC;
compt_sortie : OUT STD_LOGIC_VECTOR(3 DOWNT0 0)
);

```

```

END compteur;
ARCHITECTURE rtl OF compteur IS
    SIGNAL s_compt_sortie : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000" ;
    BEGIN
        PROCESS (clk)
            BEGIN
                IF clk'EVENT AND clk = '1' THEN
                    IF entree = '1' THEN
                        IF s_compt_sortie >= 9 THEN
s_compt_sortie <= "0000";
                        ELSE
s_compt_sortie <= s_compt_sortie + 1;
                        END IF;
                    END IF;
                END IF;
            END PROCESS;
compt_sortie <= s_compt_sortie;
END;

```

- Créer un projet dans ise8.2 et ajouter lui le fichier Compte_autor.vhd et test1_tb.vhd
- Simuler le comportement du compteur avec lsim du ise (se reporter au TP 1).

c) Compteur avec signal d'activation

Dans notre 3e exemple, on va vouloir que le compteur compte de 0 à 9 une seule fois et arrête immédiatement après. Ce processus sera déclenché quand on reçoit '1' à l'entrée pendant au moins 1 coup d'horloge.

Il faut premièrement comprendre que la situation sera un peu plus difficile. L'idée est que l'entrée ne pourra pas servir directement pour nous dire s'il faut compter ou pas. Il faut que l'entrée génère un AUTRE signal qui sera élevé sur toute la durée du compte.

C'est-à-dire que, si on détectait '1' à l'entrée, on devrait activer un signal. Ce signal devrait rester a '1' jusqu'à ce que le compte arrive à 9 et puis il tombera à 0. Ce nouveau signal servira à activer le compteur.

Le signal d'activation doit être mémorisé en quelque part et donc, il doit créer une flip flop. Son signal sera '1' quand l'entrée est à '1' et deviendra '0' quand le compte arrivera à 9.

```

PROCESS (clk)
    BEGIN
        IF clk'EVENT AND clk = '1' THEN
            IF entree = '1' THEN
s_jaipese <= '1';
                ELSIF s_compt_sortie >= 9 THEN
s_jaipese <= '0';
                END IF;
            END IF;
        END PROCESS;

```

Le signal *s_j'aipese* sera utilisé pour contrôler le compteur, de façon semblable à ce qu'il s'est fait dans le dernier exemple.

Le code complet (fichier Compte_active.vhd) ressemble à ceci:

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

```

```

ENTITY compteur IS
  PORT (
    clk : IN STD_LOGIC;
    entree : IN STD_LOGIC;
    compt_sortie : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
  );
END compteur;
ARCHITECTURE rtl OF compteur IS
  SIGNAL s_compt_sortie : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000";
  SIGNAL s_jaipese : STD_LOGIC;
BEGIN
  PROCESS (clk)
    BEGIN
      IF clk'EVENT AND clk = '1' THEN
        IF s_jaipese = '1' THEN
          IF s_compt_sortie >= 9 THEN
            s_compt_sortie <= "0000";
          ELSE
            s_compt_sortie <= s_compt_sortie + 1;
          END IF;
        END IF;
      END IF;
    END PROCESS;
  PROCESS (clk)
    BEGIN
      IF clk'EVENT AND clk = '1' THEN
        IF entree = '1' THEN
          s_jaipese <= '1';
        ELSIF s_compt_sortie >= 9 THEN
          s_jaipese <= '0';
        END IF;
      END IF;
    END PROCESS;
  compt_sortie <= s_compt_sortie;
END;

```

- Créer un projet dans ise8.2 et ajouter lui le fichier Compte_active.vhd et test3_tb.vhd
- Simuler le comportement du compteur avec lsim du ise (se reporter au TP 1).

d) Compteur avec signal de début et signal de fin

Finalement, notre dernier exemple c'est encore un compteur qui va de 0 à 9 et recommence. Cette fois-ci, quand il recevra le signal start, il comptera de 0 à 9 et recommencera continuellement. Cependant, lorsqu'il reçoit un signal stop, il arrêtera le compte. Alors cet exemple ressemble un peu à ce qui s'est fait dans l'exemple précédent. Cette fois-ci, cependant, il va y avoir un autre signal. L'idée est « si je presse sur start, le signal d'activation montera a '1'. Si je presse sur stop, le signal d'activation tombera a '0' ». Ce signal d'activation est ensuite utilisé pour contrôler le compteur.

```

PROCESS (clk)
  BEGIN
    IF clk'EVENT AND clk = '1' THEN
      IF start = '1' THEN

```

```

s_jecompte<= '1';
      ELSIF stop = '1' THEN
s_jecompte<= '0';
      END IF;
    END IF;
  END PROCESS;
Le code complet (fichier Compte_start_stop.vhd) ressemble à ceci:
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
  ENTITY compteur IS
    PORT (
clk : IN STD_LOGIC;
start : IN STD_LOGIC;
stop : IN STD_LOGIC;
compt_sortie : OUT STD_LOGIC_VECTOR(3DOWNTO 0)
    );
  END compteur;
  ARCHITECTURE rtl OF compteur IS
    SIGNAL s_compt_sortie : STD_LOGIC_VECTOR(4 DOWNTO 0) := "0000";
    SIGNAL s_jecompte : STD_LOGIC;
    BEGIN
      PROCESS (clk)
      BEGIN
        IF clk'EVENT AND clk = '1' THEN
          IF s_jecompte = '1' THEN
            IF s_compt_sortie>= 9 THEN
s_compt_sortie<= "0000";
            ELSE
s_compt_sortie<= s_compt_sortie + 1;
            END IF;
          END IF;
        END IF;
      END PROCESS;
      PROCESS (clk)
      BEGIN
        IF clk'EVENT AND clk = '1' THEN
          IF start = '1' THEN
s_jecompte<= '1';
            ELSIF stop = '1' THEN
s_jecompte<= '0';
            END IF;
          END IF;
        END PROCESS;
compt_sortie<= s_compt_sortie;
END;

```

- Créer un projet dans ise8.2 et ajouter lui le fichier Compte_start_stop.vhd et test4_tb.vhd
- Simuler le comportement du compteur avec lsimdu ise (se reporter au TP 1).
- ❖ Donner pour chaque cas le nombre de slices, de bascule D, de LUT et d'entrées sorties nécessaires à la réalisation du circuit.