VHDL – TD3

1 Boucles

Quelques exemples de boucles posant problème.

1.1 Boucle non synthétisable

```
ENTITY cpt100 IS
      PORT( hor : IN BIT ;
         un, dix : OUT INTEGER RANGE 0 TO 9);
END cpt100 ;

ARCHITECTURE boucle OF cpt100 IS -- non synthétisable
BEGIN
      compteur : PROCESS
      BEGIN
            dizaines: FOR d IN 0 TO 9 LOOP
                  dix <= d;
                  unites: FOR u IN 0 TO 9 LOOP
                        un <= u;
                        WAIT UNTIL hor = '1';
                  END LOOP unites;
            END LOOP dizaines;
      END PROCESS compteur;
END boucle;
```

Ce programme est simulable mais n'est pas synthétisable. Pourquoi ?
Donner une forme synthétisable de ce programme.

1.2 Boucles inutiles

On veut créer des impulsions périodiques de période 40 ns. Le programme suivant est une programmation possible, mais utilise une boucle inutile. Pourquoi ?

```
ENTITY hor_simple IS
   PORT (hor: BUFFER BIT);
END hor_simple;

ARCHITECTURE naive OF hor_simple IS
BEGIN
   horloge: PROCESS
   BEGIN
      hor <= '0';
      LOOP
         WAIT FOR 20 ns;
         hor <= NOT hor;
      END LOOP;
   END PROCESS horloge;
END naive;
```

Proposer une version sans boucle inutile.

2 Sorties trois états

On veut modéliser une porte avec sortie 3 états. On pourrait penser créer le programme suivant :

```
PACKAGE troisetpkg IS
   TYPE z_0_1 IS ('0','1','Z') ;
   FUNCTION to_z_0_1(e, oe: BIT) RETURN z_0_1 ;
END PACKAGE troisetpkg ;

PACKAGE BODY troisetpkg IS
   FUNCTION to_z_0_1(e, oe: BIT) RETURN z_0_1 IS
   BEGIN
      IF oe = '0' THEN
         IF e = '0' THEN
            RETURN '0';
         ELSE
            RETURN '1';
         END IF;
      ELSE
         RETURN 'Z';
      END IF;
   END to_z_0_1 ;
END PACKAGE BODY troisetpkg;

USE work.troisetpkg.all;

ENTITY tri_buffer IS
   PORT(e, oe: IN BIT;
        s: OUT z_0_1);
END tri_buffer;

ARCHITECTURE test OF tri_buffer IS
BEGIN
   s <= to_z_0_1(e,oe);
END test;
```

Ce programme est synthétisable, mais il ne produit pas le résultat escompté. Pourquoi ? (penser au codage du type énuméré). Modifier ce programme pour qu'il rende la fonctionnalité attendue.

3 Prise en compte des retards en simulation

3.1 Un paquetage pour les durées

On propose le package suivant (J. Weber, M. Meaude. Le Langage VHDL. DUNOD) :

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-- ------------------------------
PACKAGE timing IS

   PROCEDURE propagate (
      SIGNAL sigOut: OUT STD_LOGIC;
      VARIABLE new_value: IN STD_LOGIC;
      VARIABLE old_value: INOUT STD_LOGIC;
      CONSTANT tphl, tplh: TIME);
```

```
    PROCEDURE check_rules (
       CONSTANT name: IN STRING;
       SIGNAL clk: IN STD_LOGIC;
       SIGNAL sigIn: IN STD_LOGIC;
       VARIABLE clk_edge, sigIn_chg: INOUT TIME;
       CONSTANT tsu, th: IN TIME;
       VARIABLE diag: OUT BOOLEAN);
END PACKAGE timing;

PACKAGE BODY timing IS
   PROCEDURE propagate (
      SIGNAL sigOut: OUT STD_LOGIC;
      VARIABLE new_value: IN STD_LOGIC;
      VARIABLE old_value: INOUT STD_LOGIC;
      CONSTANT tphl, tplh: TIME) IS

      VARIABLE values: STD_LOGIC_VECTOR(0 to 1); // CA's patch
   BEGIN
      -- CASE STD_LOGIC_VECTOR'(new_value,old_value) IS
      values := old_value & new_value; // CA's patch
      CASE values IS
         WHEN "00" | "11" => RETURN;
         WHEN "01" | "X1" =>
             sigOut <= new_value AFTER tplh;
             old_value := new_value;
         WHEN "10" | "X0" =>
             sigOut <= new_value AFTER tphl;
             old_value := new_value;
         WHEN OTHERS =>
             sigOut <= 'X';
             old_value := 'X';
      END CASE;
   END PROCEDURE propagate;

   PROCEDURE check_rules (
       CONSTANT name: IN STRING;
       SIGNAL clk: IN STD_LOGIC;
       SIGNAL sigIn: IN STD_LOGIC;
       VARIABLE clk_edge, sigIn_chg: INOUT TIME;
       CONSTANT tsu, th: IN TIME;
       VARIABLE diag: OUT BOOLEAN) IS
   BEGIN
       IF NOW = 0 ns THEN
           clk_edge := 0 ns;
           sigIn_chg := 0 ns;
           RETURN;
       END IF;
       diag := TRUE; -- default
       IF sigIn'EVENT  THEN
           IF RISING_EDGE(clk) THEN
               clk_edge := NOW;
           END IF;
           IF NOW - clk_edge < th THEN
               REPORT name & ": hold_value time violation";
               diag := FALSE;
           END IF;
           sigIn_chg := NOW;
        ELSIF RISING_EDGE(clk) THEN
            IF NOW - sigIn_chg < tsu THEN
                REPORT name & ": set up time violation";
                diag := FALSE;
```

```
            END IF;
            clk_edge := NOW;
         ELSIF
            clk'EVENT AND (clk = '1' OR clk'LAST_VALUE = '0') THEN
               REPORT name & ":illegal clock edge";
               diag := FALSE;
           END IF;
   END PROCEDURE check_rules;
END PACKAGE BODY timing;
```

Analyser le contenu de ce package.

3.2 Modélisation d'un DFF avec retards

```
-- timed_DFF.vhd
LIBRARY ieee;

USE work.timing.all;
USE ieee.std_logic_1164.all;

ENTITY t_dff IS
    GENERIC (
        tp : TIME := 3 ns ; -- propagation
        tsu : TIME := 8 ns ; -- set up > 0
        tho : TIME := 2 ns -- hold > 0
    );
    PORT (
       clk, d: IN STD_LOGIC;
       q: OUT STD_LOGIC
    );
END ENTITY t_dff;
```

Écrire une architecture pour l'entité `t_dff` (bascule D avec temps de propagation, de set up et de hold).

3.3 Programme de test

Écrire un programme de test détectant les éventuelles violations de temps de set up ou de hold.