

VHDL – TD2

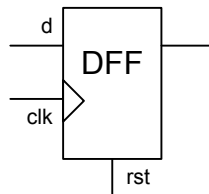
Exemples de circuits

Programmer en VHDL les circuits ci-dessous.

1 Bascules

1.1 Bascule D avec remise à zéro asynchrone.

(DFF with asynchronous reset)



Avec un **PROCESS** ayant une liste de sensibilité.

Avec un **PROCESS** ayant un **WAIT**.

Avec un **PROCESS** incluant un **CASE**.

1.2 Bascule D avec sorties q et qbar .

Expliquer pourquoi la solution suivante est incorrecte. Proposer une solution correcte.

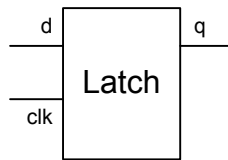
```

1 ---- Solution 1: not OK -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -----
5 ENTITY dff IS
6   PORT ( d, clk: IN STD_LOGIC;
7         q: BUFFER STD_LOGIC;
8         qbar: OUT STD_LOGIC);
9 END dff;
10 -----
11 ARCHITECTURE not_ok OF dff IS
12 BEGIN
13   PROCESS (clk)
14   BEGIN
15     IF (clk'EVENT AND clk='1') THEN
16       q <= d;
17       qbar <= NOT q;
18     END IF;
19   END PROCESS;
20 END not_ok;
21 -----

```

1.3 Latch

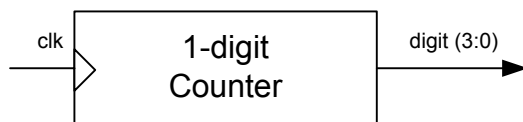
Le latch n'est pas une bascule, c'est une mémoire. Tant que clk est à '1', q prend les valeurs de d. Lorsque clk passe à '0', la dernière valeur de d est mémorisée. Programmer un Latch.



2 Compteurs

2.1 Compteur décimal 1 chiffre

Programmer un compteur de 0 à 9 qui est incrémenté (modulo 10) à chaque front montant d'horloge.



2.2 Compteur générique

On crée un paquetage "decimal" défini par:

```
PACKAGE decimal IS
  TYPE chiffre IS RANGE 0 TO 9;
  TYPE nombre IS ARRAY (INTEGER RANGE <>) OF chiffre;

  COMPONENT mod_cnt IS
    GENERIC(modulo: INTEGER:=16);
    PORT (clk, clear, cen: IN BIT;
          cout: OUT INTEGER RANGE 0 TO modulo-1;
          rco : OUT BIT);
  END COMPONENT mod_cnt;
END DECIMAL;
```

Le composant "mod_cnt" est un compteur modulo générique pour lequel on demande d'écrire la spécification en VHDL.

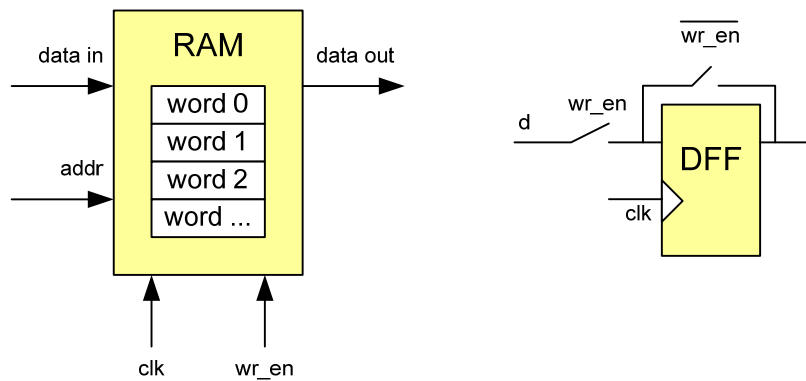
Ce compteur est compilé dans une bibliothèque genlib. Expliquer comment l'instancier pour obtenir un compteur modulo 10 (entité decade).

3 Additionneurs

3.1 Additionneur avec propagation de retenue

(Carry Ripple Adder) La solution doit être élaborée à partir d'additionneurs complets 1 bit (1-bit full-adder). On demande une solution générique utilisant des STD_LOGIC_VECTORS et une solution non-générique utilisant les entiers.

4 RAM



La mémoire a un bus d'entrée (data in), un bus d'adresse (addr), une horloge (clk) et une validation d'écriture (wr_en). Quand wr_en est activé, data in est rangé dans le mot d'adresse addr lors du front suivant de clk. Quant à la sortie data out elle doit en permanence afficher le mot sélectionné par addr. Le schéma de droite résume ce comportement.

Programmer une mémoire RAM générique de words mots de bits bits (défauts: bits=8 et words=16).