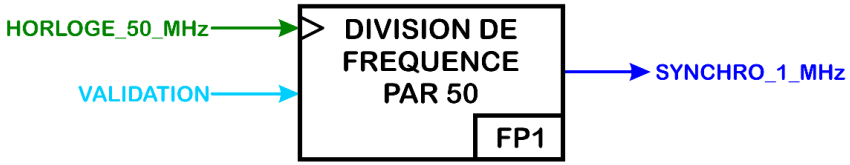
BTS SN	FORMATION ALTIUM CODEUR INCRÉMENTAL VHDL + FSM + SIMULATION	TP10 -3-

Altium Designer

I. PRÉSENTATION DE LA FONCTION FP1

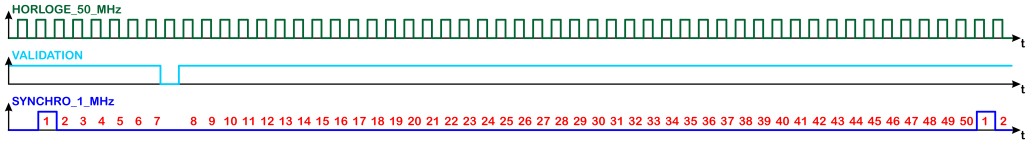
Le signal d'horloge disponible sur la carte FPGA utilisée est de 50 MHz, cette fréquence est trop élevée pour certaines des fonctions du projet à réaliser.

I.1. DESCRIPTION DU COMPOSANT DIVISION_FRÉQUENCE_50



A partir du signal HORLOGE_50_MHz nous devons produire un signal d'une période 50 fois plus grande avec un rapport cyclique de 2 %.
Une entrée de validation active à 1 permet d'autoriser la division de fréquence.
Il devient alors possible de cascader plusieurs fonctions FP1 afin de produire des signaux de fréquences 1 MHz, 20 KHz, 400 Hz suivant le besoin.
Il est bien sur facile de modifier le fichier VHDL pour effectuer une division par un nombre différent de 50, suivant le besoin.

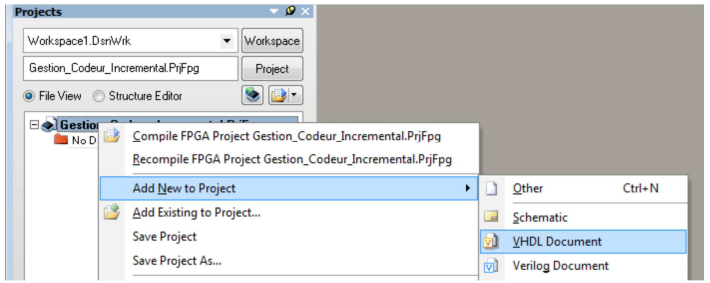
I.2. CHRONOGRAMMES ATTENDUS



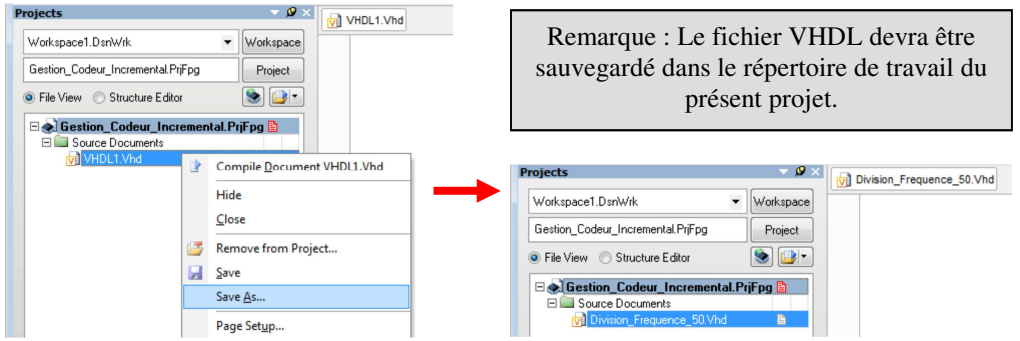
II. ÉDITION D'UN FICHIER VHDL

II.1. AJOUT AU PROJET D'UN FICHIER VHDL

Ajouter un nouveau fichier VHDL par un **clic droit** sur le nom du projet FPGA
⇒ **Add New to Project** ⇒ **VHDL Document.**



Puis sauvegarder ce fichier VHDL par un **clic droit** sur le nom du fichier : VHDL1.Vhd
⇒ **Save As…** ⇒ **Division_Frequence_50.Vhd**

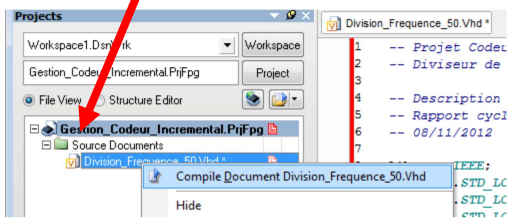


Remarque : Le fichier VHDL devra être sauvegardé dans le répertoire de travail du présent projet.

II.2. ÉDITION ET COMPILATION DU FICHIER VHDL

Editer le fichier VHDL et le sauvegarder.
(Voir listing en annexe page 5).

Compiler le fichier VHDL par un **clic droit** sur le nom du fichier **« Division_Frequence-_50.Vhd»** dans l'onglet **Projets** et choisir la commande **Compile Document :**



Remarque : Si la compilation n'aboutit pas corriger le fichier VHDL.
Le résultat de compilation apparaît dans la fenêtre Messages.

III. SIMULATION FONCTIONNELLE AVEC ALDEC OEM

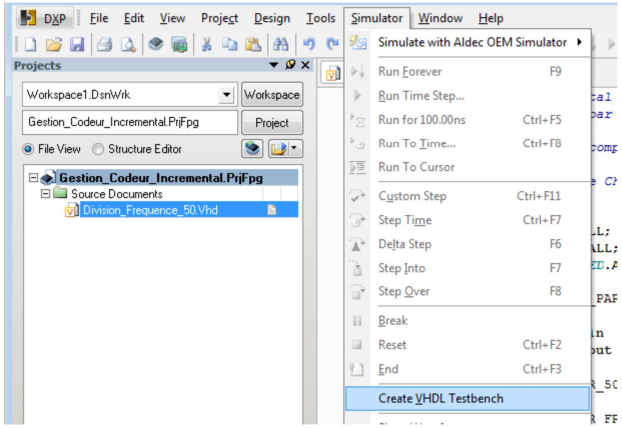
III.1. PRODUCTION DU FICHIER DE TEST

Altium designer est livré avec un simulateur fonctionnel VHDL de la société Aldec.

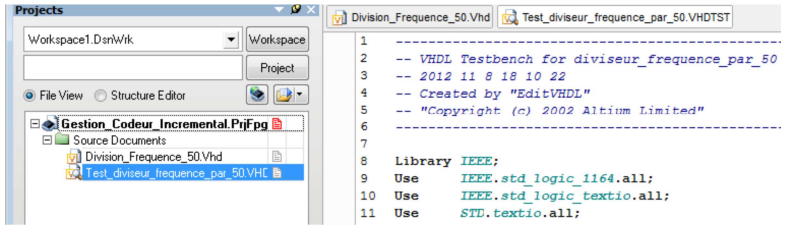
Vérifier que le simulateur ALDEC OEM est bien sélectionné par la commande :
⇒ **Project ⇒ Project Options …**
Puis dans l'onglet **Simulation** vérifier que ce soit bien **Adec OEM Simulator** qui soit sélectionné dans le menu déroulant **Tool**.

Le langage VHDL est aussi prévu pour faire des simulations numériques. Pour cela il faut produire un fichier de tests (testbenches) contenant des informations à destination du simulateur, en particulier les signaux d'entrées avec leurs excitations et la liste des signaux de sorties à observer.

Produire le fichier VHDL de test par la commande :
⇒ **Simulator ⇒ Create VHDL Testbench**



Un Nouveau fichier VHDL est ajouté au projet :



Ce fichier de test doit être complété afin de pouvoir être utilisé.

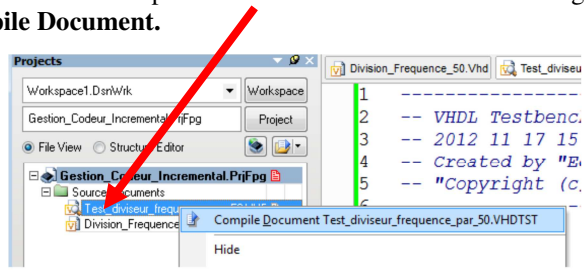
Repérer la ligne 54 puis ajouter en dessous un processus nommé GENERATION_HORLOGE comme ci-après afin de générer le signal HORLOGE_50MHz avec une période de 20 ns (2 * 10 ns) avec un rapport cyclique de 50 %.

```
54
55      GENERATION_HORLOGE:process
56      begin
57          HORLOGE_50MHz <= '0';
58          wait for 10 ns;
59          HORLOGE_50MHz <= '1';
60          wait for 10 ns;
61      end process;
62
63      STIMULUS0:process
64      begin
65          -- insert stimulus here
66          VALIDATION <= '0';
67          wait for 100 ns;
68          VALIDATION <= '1';
69          wait;
70      end process;
71
```

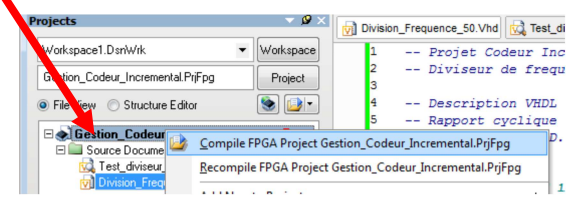
Ajouter les lignes dans le processus STIMULUS0 afin de décrire l'évolution du signal VALIDATION (Simulis). Celui-ci sera à 0 au début de la simulation et passera à 1 au bout de 100 ns.
(Voir listing en annexe page 6).

Enregistrer le fichier de test sans changer le nom.
⇒ **File ⇒ Save**

Compiler le fichier de test VHDL par un **clic droit** sur son nom dans l'onglet **Projets** et choisir la commande **Compile Document.**

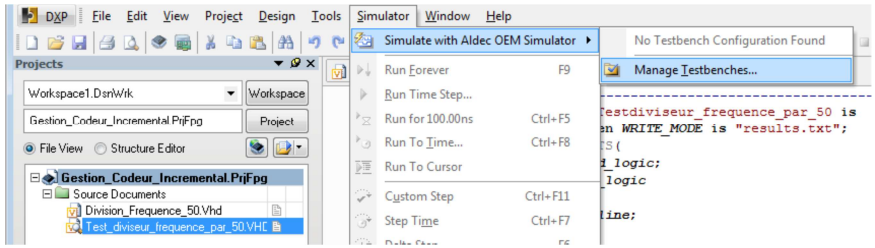


⇒ Cliquer droit sur le nom du projet.
⇒ Choisir la commande **Compile FPGA Projet** pour compiler l'ensemble des fichiers.

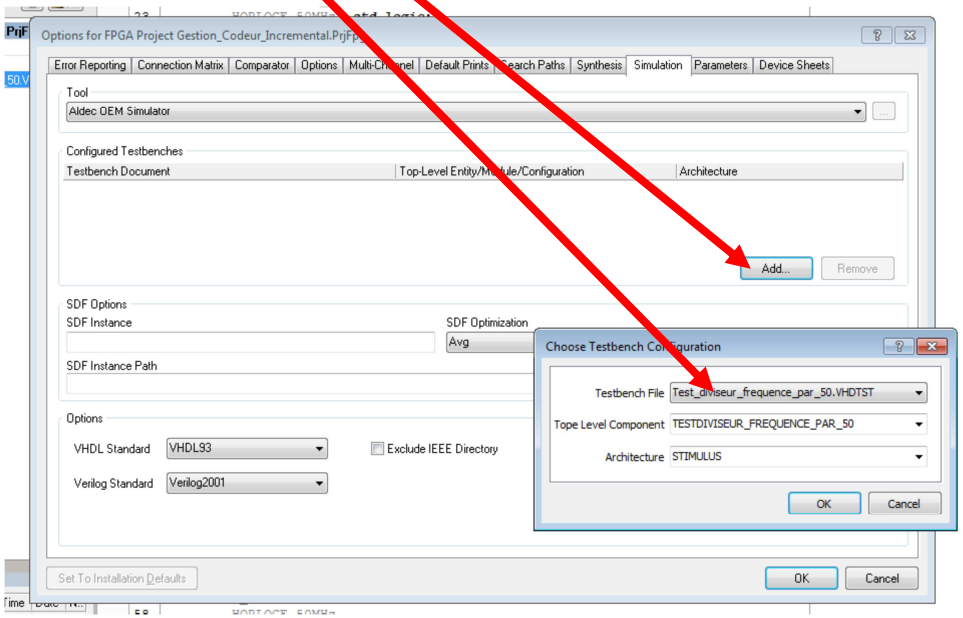


III.2. ASSOCIATION DES FICHIERS

Il faut ensuite associer le fichier VHDL à simuler à son fichier de test avec la commande :
⇒ **Simulator ⇒ Simulator with Aldec OEM Simulator ⇒ Manage Testbenches…**

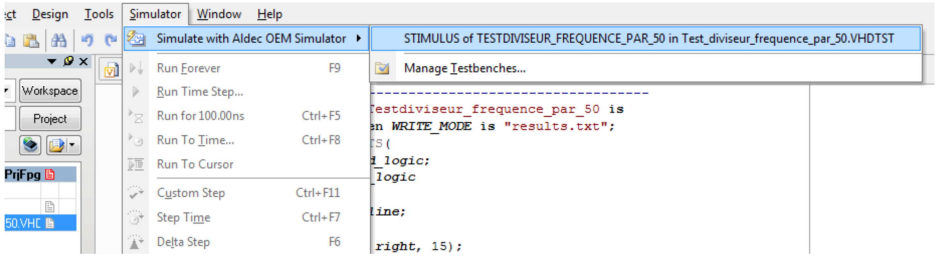


Cliquer sur le bouton **Add…** dans la fenêtre qui apparait alors et choisir « **Test_Diviseur_Frequence_par_50.VHDTST** » puis cliquer sur **OK**.



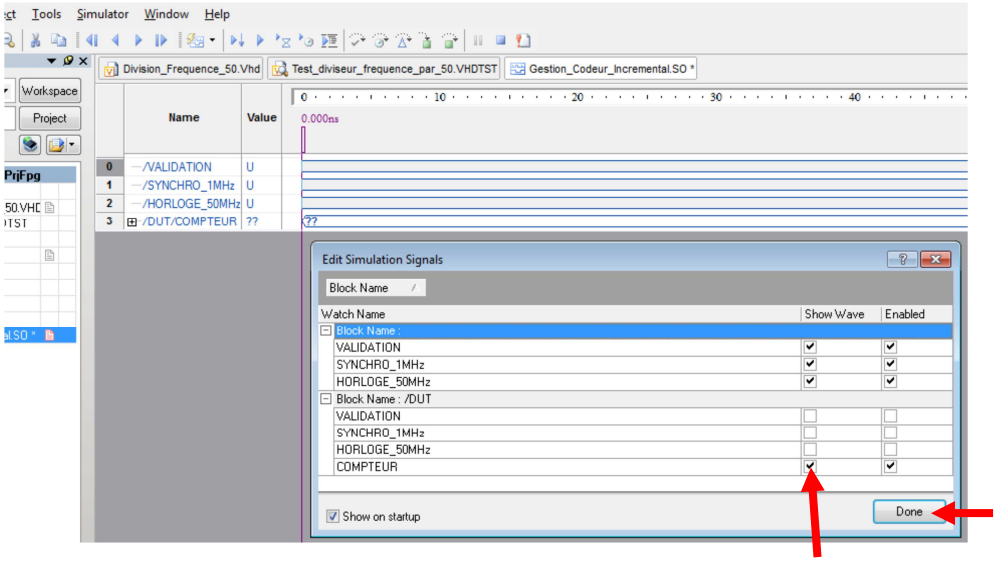
III.3. SIMULATION

Lancer maintenant la simulation avec la commande :
⇒ **Simulator ⇒ Stimulus of TESTDIVISEUR_FREQUENCE_PAR_50 in Test_Div…**



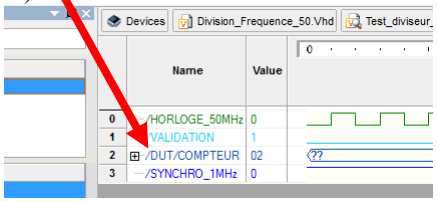
Il est aussi possible de lancer la simulation avec l'icône : 

Un nouvel onglet apparait alors dans Altium Designer.

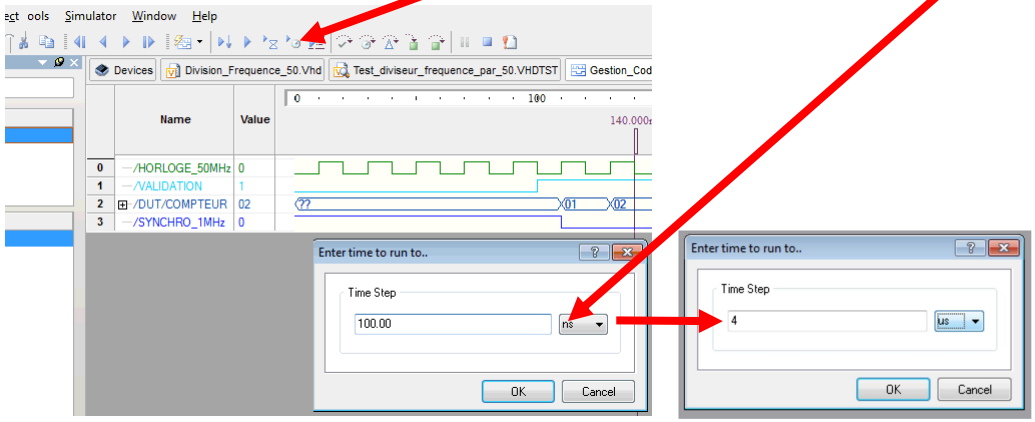


Les ports (signaux externes) sont cochés par défaut pour l'affichage de leurs chronogrammes. Cocher le signal interne COMPTEUR pour le faire apparaitre dans la fenêtre de simulation. Cliquer sur le bouton **Done**.

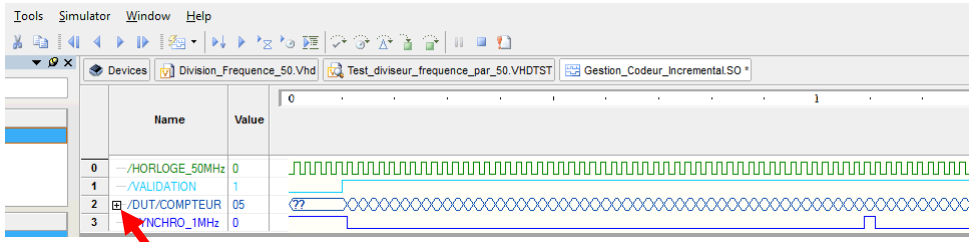
Déplacer les signaux en cliquant dessus afin de mettre les entrées en haut (avec les horloges tout en haut) puis les signaux internes et les sorties pour finir (afin d'aider à la lecture donc à une meilleure compréhension).



Lancer la simulation en cliquant sur le bouton indiqué et préciser le temps de simulation (4 us).

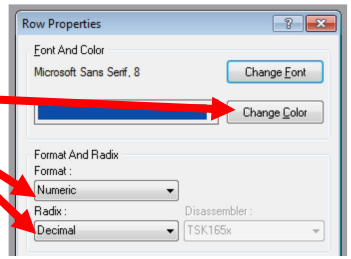


Cliquer sur OK. La simulation s'effectue sur une durée de 4 µs.
Faire CTRL PAGE-DOWN pour faire apparaitre toute la durée de simulation.



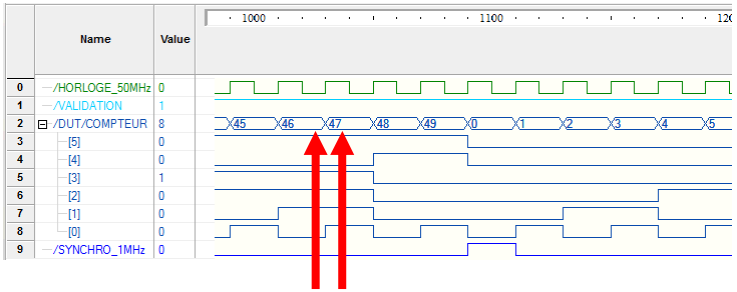
Cliquer sur le petit + devant /DUT/COMPTEUR pour en faire apparaitre tous les bits.

Effectuer un **clic droit** sur /DUT/COMPTEUR
Puis cliquer sur Propriétés.
Modifier la couleur si besoin.
Modifier les propriétés d'affichage du vecteur comme indiqué ci-contre.



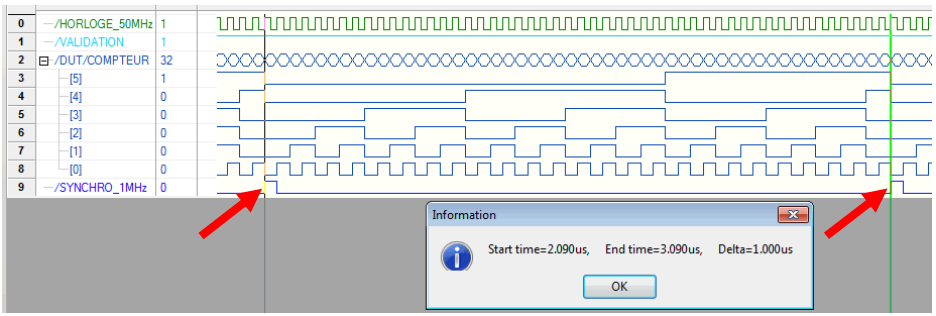
Zoomer sur une partie intéressante des chronogrammes.

L'affichage devient :



On peut remarquer que l'état haut de SYNCHRO_1MHz dure une période du signal d'horloge HORLOGE_50MHz et se répète toutes les 50 périodes (0 à 49).

Effectuer un **clic droit** dans la fenêtre et sélectionner **Measure Time** puis placer les 2 curseurs verts sur les fonts montants de SYNCHRO_1MHz. On trouve bien 1 µs (50 * [10µs + 10µs]).



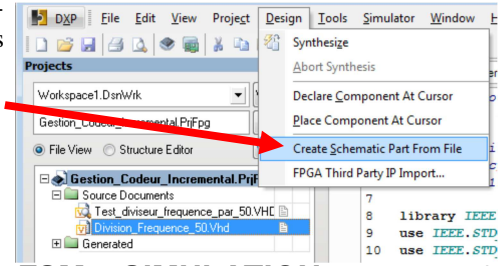
Ici la fonction réalisée est très simple, il n'est donc pas nécessaire de faire d'autres simulations. Le résultat de la simulation est bien conforme à ce qui était attendu : nous pouvons donc valider la fonction FP1.

Sauvegarder les chronogrammes par la commande :
⇒ **File ⇒ Save** Accepter le nom proposé.
Fermer la fenêtre de simulation par la commande :
⇒ **File ⇒ Close**

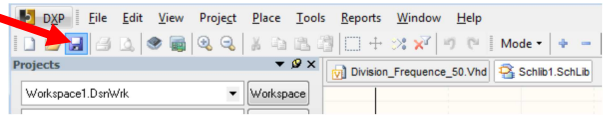
IV. CRÉATION D'UN SYMBOLE POUR LA FONCTION FP1

Vérifier que la fenêtre de travail est bien le fichier VHDL **Division_frequence_50.Vhd** puis créer un symbole avec la commande :
⇒ **Design ⇒ Create Shematic Part From File**

Faire **Ok**. Le symbole apparait alors à l'écran.



Sauvegarder sous le nom **Division_Frequence_50.SchLlib** puis fermez-le : il nous servira plus tard.



V. ANNEXES

```
1   -- Projet Gestion_Codeur_Incremental
2   -- Diviseur de frequence par 50
3
4   -- Description VHDL d'un composant diviseur de fréquence par 50
5   -- Rapport cyclique 1/50
6   -- 08/11/2012  C. D. Lycée Chevrollier Angers
7
8   library IEEE;                                          --Liste des librairies prédéfinies utilisées
9   use IEEE.STD_LOGIC_1164.ALL;
10  use IEEE.STD_LOGIC_ARITH.ALL;
11  use IEEE.STD_LOGIC_UNSIGNED.ALL;
12
13 entity DIVISEUR_FREQUENCE_PAR_50 is                     --Description de l'entité (broches du composant)
14    port (
15          HORLOGE_50MHz   : in  std_logic;               --Entrée de l'horloge à dividser (FM)
16          VALIDATION      : in  std_logic;               --Entrée de validation du fonctionnement
17          SYNCHRO_1MHz    : out std_logic                --Sortie de d'horloge divisiée de rapport cyclique 1/50
18         );
19  end DIVISEUR_FREQUENCE_PAR_50;                         --Fin de la description de l'entité
20
21 architecture ARCH_DIVISEUR_FREQUENCE_PAR_50 of
22              DIVISEUR_FREQUENCE_PAR_50 is               --Description du comportement interne (silicium)
23    signal COMPTEUR : std_logic_vector (5 downto 0) := "000000";  --Déclaration d'un compteur sur 6 bits
24    begin
25      process (HORLOGE_50MHz)                            --Partie synchrone sur le signal HORLOGE
26        begin                                            --Fonctionnement complètement synchrone
27          if (HORLOGE_50MHz'event and HORLOGE_50MHz = '1')  --Attente d'un font montant du signal HORLOGE_50
28            then
29              if (VALIDATION = '1')                      --Autorisation de fonctionnement
30                Then
31                  if (COMPTEUR >= "110001")              --Si le compteur est rendu à 49 ou plus
32                    then
33                      COMPTEUR <= "000000";              --Remetre le compteur à 0
34                    else                                 --sinon
35                      COMPTEUR <= COMPTEUR + 1;          --Incrémenter le compteur
36                  end if;
37             end if;
38          end if;
39      end process;                                       --Fin de la partie sychrone
40      SYNCHRO_1MHz <= '1' when (COMPTEUR = "000000")     --Actualisation de SYNCHRO_1MHz à 1 si compteur vaut 0
41              else '0';                                  --sinon remise de SYNCHRO_1MHz à 0
42  end ARCH_DIVISEUR_FREQUENCE_PAR_50;                    --Fin de la description interne
```

```vhdl
--------------------------------------------------------------------
-- VHDL Testbench for diviseur_frequence_par_50
-- 2012 11 17 15 33 32
-- Created by "EditVHDL"
-- "Copyright (c) 2002 Altium Limited"
--------------------------------------------------------------------

Library IEEE;
Use     IEEE.std_logic_1164.all;
Use     IEEE.std_logic_textio.all;
Use     STD.textio.all;
--------------------------------------------------------------------

--------------------------------------------------------------------
entity Testdiviseur_frequence_par_50 is
end Testdiviseur_frequence_par_50;
--------------------------------------------------------------------

--------------------------------------------------------------------
architecture stimulus of Testdiviseur_frequence_par_50 is
    file RESULTS: TEXT open WRITE_MODE is "results.txt";
    procedure WRITE_RESULTS(
        HORLOGE_50MHz: std_logic;
        SYNCHRO_1MHz: std_logic;
        VALIDATION: std_logic
    ) is
        variable l_out : line;
    begin
        write(l_out, now, right, 15);
        write(l_out, HORLOGE_50MHz, right, 2);
        write(l_out, SYNCHRO_1MHz, right, 2);
        write(l_out, VALIDATION, right, 2);
        writeline(RESULTS, l_out);
    end procedure;

    component diviseur_frequence_par_50
        port (
            HORLOGE_50MHz: in std_logic;
            SYNCHRO_1MHz: out std_logic;
            VALIDATION: in std_logic
        );
    end component;

    signal HORLOGE_50MHz: std_logic;
    signal SYNCHRO_1MHz: std_logic;
    signal VALIDATION: std_logic;

begin
    DUT:diviseur_frequence_par_50 port map (
        HORLOGE_50MHz => HORLOGE_50MHz,
        SYNCHRO_1MHz => SYNCHRO_1MHz,
        VALIDATION => VALIDATION
    );

    GENERATION_HORLOGE:process
    begin
        HORLOGE_50MHz <= '0';
        wait for 10 ns;
        HORLOGE_50MHz <= '1';
        wait for 10 ns;
    end process;

    STIMULUS0:process
    begin
        -- insert stimulus here
        VALIDATION <= '0';
        wait for 100 ns;
        VALIDATION <= '1';
        wait;
    end process;

    WRITE_RESULTS(
        HORLOGE_50MHz,
        SYNCHRO_1MHz,
        VALIDATION
    );

end architecture;
--------------------------------------------------------------------

--------------------------------------------------------------------
```