Familiarisation avec les étapes d'un projet VHDL

Design flow

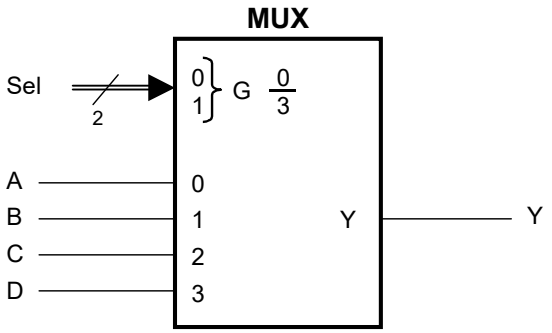
Cette manipulation a pour objectif de familiariser l'étudiant aux différentes étapes de **réalisation** d'un projet en VHDL. Le terme couramment utilisé en anglais est : *Design flow*.

L'étudiant pourra ainsi pratiquer les différentes étapes et utiliser les deux outils :

le **simulateur** Questasim
l'outil de **placement et routage** Quartus II (inclus un **synthétiseur**)

Il est important de comprendre la fonction de chaque outil.

Nous utiliserons pour cette manipulation un circuit combinatoire. Il s'agit d'un multiplexeur 4 à 1. Il dispose d'une entrée de sélection sur deux bits (Sel) et de quatre entrées à sélectionner. En sortie, le circuit dispose d'une seule sortie qui prend la valeur de l'entrée sélectionnée.



La description VHDL du multiplexeur, sous la forme d'une description de type "flot de donnée" (flot_don), vous est fournie dans le fichier nommé **mux4to1.vhd**.

Le fichier **console_sim.vhd** sert à la simulation manuelle de la description. Il sera utilisé avec la *Console_REDS*. Celle-ci comprend des *Switchs* et des *Leds* pour réaliser une simulation manuelle du circuit.

Le fichier **mux4to1_tb.vhd** sert à la simulation automatique de la description **mux4to1.vhd**. Le suffixe **_tb** est l'abréviation de "**test-bench**" qui signifie "banc de test". Tous les fichiers de simulation automatique porteront le nom du fichier à tester auquel on rajoute le suffixe "**_tb".**

Travail demandé :

Documentation à utiliser : Introduction à QuestaSim
Résumé utilisation Quartus Prime

1) Compiler la description **mux4to1.vhd** dans le simulateur Questasim (vérification de la syntaxe du fichier source VHDL)
   - ouvrir une fenêtre terminal dans le répertoire *../mux4a1/comp/* puis tapez la commande *vsim* pour lancer le simulateur avec ce répertoire comme répertoire de travail
   - *ou* lancer Questasim via le menu, et choisir le répertoire *../mux4a1/comp/*

   Créer la librairie de travail work, soit : *File* ⇒ *New* ⇒ *Library*
   laisser les valeurs des champs par défaut (work, work), puis *cliquer OK*

   Compiler le fichier VHDL source **mux4to1.vhd** (répertoire ../src/..)

2) Compiler le fichier pour la simulation manuelle **console_sim.vhd** avec Questasim. Ce fichier faisant référence au composant **mux4to1** celui-ci doit être compilé après le fichier **mux4to1.vhd**.

   Compiler le fichier VHDL source **console_sim.vhd** (répertoire ../src_tb/..)

3) Simuler manuellement le composant **mux4to1** en utilisant une console manuelle à l'aide de Questasim, soit :

   Charger la simulation, soit : *Simulate* ⇒ *Start Simulation…*
   ouvrir la librairie *work* (click +), sélectionner *console_sim*, puis *cliquer OK*

   Inclure les signaux du composant à simuler, **mux4to1**, dans la fenêtre *Wave*

   Lancer la console de simulation manuelle (*Tools* ⇒ *Tcl* ⇒ *Execute_macro*)
   emplacement: /opt/tools_reds/REDS_console.tcl

   Vérifier le fonctionnement du **mux4to1** à l'aide de la console manuelle

4) Réaliser la synthèse à l'aide de Quartus II du composant **mux4to1** pour vérifier que la description est synthétisable et voir la logique réalisée. Vous choisirez la technologie Max-V et le circuit 5M570ZF256C5.

   Analyser la quantité de logique utilisé par le circuit, voir le *report*

   Vous devez visualiser la vue RTL fourni par l'outil (*Tools* ⇒ *Netlist_Viewer* ⇒ *RTL_Viewer*). Vérifier le schéma affiché.

   Vue du détail d'une connexion dans la vue RTL :
   - Sélectionner une connexion,
   - puis clic droit, sélectionner "Connectivity Details ..."

Nous pouvons faciliter la compilation des fichiers pour le lancement de la simulation à l'aide de script Tcl pour Questasim.

5) Lancer Questasim via une fenêtre terminal dans le répertoire *../mux4a1/comp/* et tapez la commande *vsim*

   Pour exécuter le script pour la simulation manuelle :

   Tapez dans la fenêtre log : *do ../run_comp_mux4to1_sim.tcl*

   Dès lors la compilation des différents fichiers et le lancement de la console REDS_console est fait automatiquement.

Nous pouvons automatiser la simulation avec l'utilisation d'un banc de test automatique l'aide de script Tcl avec Questasim.

Important : **fermer** Questasim pour l'étape suivante.

6) Lancer Questasim via une fenêtre terminal dans le répertoire *../mux4a1/comp/* et tapez la commande *vsim*

   Pour exécuter le script pour la simulation automatique :

   Tapez dans la fenêtre log : *do ../run_comp_mux4to1_tb.tcl*

   Exécuter la simulation en tapant : *run -all*

7) La preuve de la simulation automatique est constituée par les messages affichés par le test-bench dans la fenêtre log.

Voici les étapes pour réaliser l'intégration du composant Mux4to1 dans la carte Max-V.

8) Réaliser l'intégration du système à l'aide du logiciel Quartus (synthèse et placement routage) en utilisant le composant **maxv_top.vhd** contenant le circuit **mux4to1**. Le répertoire de travail est ../pr_cpld/..

   Le fichier **maxv_top.vhd** définit les interconnections entre le circuit **mux4to1** et les différentes I/O de la carte Max-V 25-80 pôles.

   Important : ne pas oublier d'effectué l'assignation des pins du circuit Max-V. Voir le résumé pour l'utilisation de Quartus II

   maxv_top _pin_assignment.qsf : ce fichier contient les assignations de toutes les pins de la carte Max-V

   Programmer le circuit 5M570ZF256C5 d'une maquette MaxV 80p-25p 25p-25p avec le fichier le **maxv_top.pof"** (fichier de programmation du CPLD)

9) Vérifier le fonctionnement du circuit **mux4to1**

   Vous devez lire le fichier **maxv_top.vhd** afin de déterminer où sont connectés les signaux du mux4to1.vhd

Le fichier **Mux4_1.vhd**

```
------------------------------------------------------------------
-- HEIG-VD, Haute Ecole Ingenierie et Gestion du canton de Vaud
-- Institut REDS, Reconfigurable & Embedded Digital Systems
--
-- Fichier      : mux4to1.vhd
-- Description  : Multiplexeur 4 a 1, decription flot donnee
--
-- Auteur       : Etienne Messerli
-- Date         : 07.01.2004
--
--| Modifications |-------------------------------------------
-- Vers   Qui    Date        Description
-- 1.0    EMI    13.08.2015  Remise a jour entete, noms signaux
--
------------------------------------------------------------------

library IEEE;                   -- Librairie IEEE
use IEEE.Std_Logic_1164.all;    -- Definition du type standard logic

entity mux4to1 is
  port (sel_i              : in  Std_Logic_Vector(1 downto 0);
        a_i, b_i, c_i, d_i : in  Std_Logic;
        y_o                : out Std_Logic
        );
end Mux4_1;

architecture flot_don of mux4to1 is
begin
   with sel_i select
     Y <= a_i  when "00",
          b_i  when "01",
          c_i  when "10",
          d_i  when "11",
          'X'  when others; -- simulation
end flot_don;
```

Remarques sur le fichier :

-- indique le début d'un commentaire, la fin de ligne ferme le commentaire

L'en-tête donne des indications sur l'école, le ou les auteurs, le nom et le but du fichier, la date

```
------------------------------------------------------------------
-- HEIG-VD, Haute Ecole d'….
-- Institut REDS, … …
--
-- Fichier       : mux4to1.vhd
-- …
…
------------------------------------------------------------------
```

Déclaration des librairies :

la librairie IEEE.Std_Logic_1164.all sera toujours déclarée (indispensable)

```
library IEEE;
use IEEE.Std_Logic_1164.all;
```

L'entité constitue la définition du bloc avec ses entrées/sorties : Le type Std_logic sera utilisé pour un signal un bit et le type Std_Logic_Vector pour un signal de plusieurs bits (vecteur), les indices sont précisés en suivant.

```
entity mux4to1 is
  port (sel_i              : in  Std_Logic_Vector(1 downto 0);
        a_i, b_i, c_i, d_i : in  Std_Logic;
        y_o                : out Std_Logic
        );
end mux4to1;
```

L'architecture contient le comportement du bloc (relation entre les entrées et sorties). La description est donnée ici sous la forme d'un flot de donnée (l'instruction de sélection correspond à un multiplexeur).

```
architecture flot_don of mux4to1 is
begin
   with sel_i select
     Y <= a_i  when "00",
          b_i  when "01",
          c_i  when "10",
          d_i  when "11",
          'X'  when others; -- simulation
end flot_don;
```

La ligne 'X' **when others**; paraît inutile, cela n'est pas le cas, car le type Std_Logic ne recouvre pas uniquement les deux états logiques '0' ou '1', mais 9 états ('0', '1', '-', 'Z', ...). Le vecteur sel_i a donc plus que 4 états en simulation ! Les états autres que '0' et '1' sont utilisés uniquement durant la simulation