Labo d'introduction
Familiarisation avec les étapes d'un projet VHDL

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

Ce laboratoire d'introduction est réalisé individuellement.

Vous pourrez ainsi pratiquer les différentes étapes et utiliser les deux outils :

le **simulateur** Questasim

l'outil de **placement et routage** Quartus Prime (inclus un **synthétiseur**)

Il est important de comprendre la fonction de chaque outil. Nous vous recommandons de faire un résumé pour l'utilisation des outils.

Nous utiliserons pour cette manipulation un circuit combinatoire. Il s'agit d'un transcodeur binaire-linéaire. Il dispose en entrée de la valeur en binaire à convertir. En sortie, le circuit dispose d'une seule sortie qui correspond à la valeur d'entrée linéarisée.

Voici la table de vérité du transcodeur d'une valeur binaire en un affichage linéaire.

bin_i		*lin_o*			
1	***0***	***3***	***2***	***1***	***0***
0	*0*	*0*	*0*	*0*	*1*
0	*1*	*0*	*0*	*1*	*1*
1	*0*	*0*	*1*	*1*	*1*
1	*1*	*1*	*1*	*1*	*1*

La description VHDL du transcodeur, sous la forme d'une description de type "table de vérité" (tdv), vous est fournie dans le fichier nommé **bin_lin_2to4.vhd**.

Le fichier **console_sim.vhd** (dossier *src_tb*) 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 bin_lin_2to4_tb.vhd sert à la simulation automatique de la description bin_lin_2to4.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".

Projet fourni :

Une archive du projet vous est fournie sur Cyberlearn, page "20_HEIG-VD_CSN".

Vous devez travailler sur les machines de labo à l'emplacement suivant :

- il y a un raccourci sur le bureau : cours_REDS
- puis travailler dans le répertoire : cours_REDS/<unité>/<nom>/…

Documents à rendre :

Ce laboratoire est réalisé individuellement. Celui-ci sera validé et vous devez rendre les documents suivants dans un seul fichier PDF :

- Vues RTL et Technology de la synthèse du **composant Bin-Lin 2 à 4** de la description par flot de données.
- Description VHDL par flot de données du transcodeur Bin-Lin 3 à 8 en utilisant l'instruction when .. else (flot_don).
- Log de Questasim de la simulation automatique de la description par flot de données du transcodeur Bin-Lin 3 à 8.
- Vues RTL et Technology de la synthèse du transcodeur Bin-Lin 3 à 8 pour la description par flot de données.

Vous devrez déposer les documents ci-dessus dans **un seul fichier PDF** sur Cyberlearn (Moodle) page du cours CSN.

1ère partie

Travail demandé :

Documentation à utiliser : Introduction à QuestaSim
Résumé utilisation Quartus Prime

1) Compiler la description **bin_lin_2to4.vhd** dans le simulateur Questasim (vérification de la syntaxe du fichier source VHDL), soit:
   - ouvrir une fenêtre terminal dans le répertoire *../bin_lin/comp/*
     puis tapez la commande *vsim* pour lancer le simulateur avec ce répertoire comme répertoire de travail
   - 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 **bin_lin_2to4.vhd** (répertoire ../src/..)
2) Compiler le fichier pour la simulation manuelle **console_sim.vhd** avec Questasim. Ce fichier faisant référence au composant **bin_lin_2to4** celui-ci doit être compilé après le fichier **bin_lin_2to4.vhd**.
   - Compiler le fichier VHDL source **console_sim.vhd** (répertoire ../src_tb/..)
3) Simuler manuellement le composant **bin_lin_2to4** 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, **bin_lin_2to4**, 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 **bin_lin_2to4** à l'aide de la console manuelle (toujours fermer la console en utilisant le bouton "Quitter", la croix ne fonctionnant pas correctement)
4) Vérifier que la description VHDL soit synthétisable et la quantité de logique :

   Synthèse du composant **bin_lin_2to4** à l'aide de Quartus II. Vous choisirez la technologie Max-V et le circuit 5M570ZF256C5.

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

   Vous devez visualiser la vue RTL fournie 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 ..."

Intégration du composant Bin-Lin dans la carte Max-V.

5) 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 **bin_lin_2to4**. Le répertoire de travail est ../pr_cpld/..

   Le fichier **maxv_top.vhd** définit les interconnections entre le circuit **bin_lin_2to4** et les différentes I/O de la carte Max-V 25-80 pôles.

   Important : ne pas oublier d'effectuer 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 avec le fichier le **maxv_top.pof"** (fichier de programmation du CPLD)

6) Vérifier le fonctionnement du circuit **bin_lin_2to4**

   Vous devez lire le fichier **maxv_top.vhd** afin de déterminer où sont connectés les signaux du bin_lin_2to4.vhd

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

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

   Pour exécuter le script pour la simulation manuelle :

   Tapez dans la fenêtre log : *do ../run_comp_bin_lin_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.

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

   Pour exécuter le script pour la simulation automatique :

   Tapez dans la fenêtre log : *do ../run_comp_bin_lin_2to4_tb.tcl*

   Exécuter la simulation en tapant : *run -all*

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

Le fichier **bin_lin_2to4.vhd**

```
------------------------------------------------------------------
-- HEIG-VD, Haute Ecole Ingenierie et Gestion du canton de Vaud
-- Institut REDS, Reconfigurable & Embedded Digital Systems
--
-- Fichier      : bin_lin_2to4.vhd
-- Description  : decodeur 2 bits en lineaire
--                Description avec une table de verite (TDV)
--
-- Auteur       : Etienne Messerli
-- Date         : 15.02.2015
-- Version      : 0.0
--
--| Modifications |-------------------------------------------
-- Vers   Qui   Date        Description
--
------------------------------------------------------------------

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

entity bin_lin_2to4 is
  port (bin_i              : in  std_logic_vector(1 downto 0);
                             -- valeur binaire en entree
        lin_o              : out std_logic_vector(3 downto 0)
                             -- valeur lineaire en sortie
        );
end bin_lin_2to4;

architecture tdv of bin_lin_2to4 is
begin
   with bin_i select
     lin_o <= "0001" when "00",
              "0011" when "01",
              "0111" when "10",
              "1111" when "11",
              "XXXX" when others; -- simulation

end tdv;
```

Remarques sur le fichier :

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

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      : bin_lin_2to4.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 à la suite.

```
entity bin_lin_2to4 is
  port (bin_i              : in  std_logic_vector(1 downto 0);
                             -- valeur binaire en entree
        lin_o              : out std_logic_vector(3 downto 0)
                             -- valeur lineaire en sortie
        );
end bin_lin_2to4;
```

L'architecture contient le comportement du bloc (relation entre les entrées et sorties). La description est donnée ici sous la forme d'une table de vérité.

```
architecture tdv of bin_lin_2to4 is
begin
   with bin_i select
     lin_o <= "0001" when "00",
              "0011" when "01",
              "0111" when "10",
              "1111" when "11",
              "XXXX" when others; -- simulation

end tdv;
```

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 `bin_i` a donc plus que 4 états en simulation ! Les états autres que '0' et '1' sont utilisés uniquement durant la simulation

2ème partie

Travail demandé

1) Réaliser la description du transcodeur Bin-Lin 3 à 8 en utilisant l'instruction concurrente *when … else*.

   Un chablon vous est fourni: ../bin_lin/src/*bin_lin_3to8.vhd*

   L'objectif est de décrire le comportement **sans utiliser des équations logiques** !

bin_i			*lin_o*							
2	***1***	***0***	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*
0	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*
.	.	.	.	.	.	.	.	.	.	.
1	*1*	*0*	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
1	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*

2) Réaliser la simulation automatique avec le banc de test fourni *bin_lin_3to8_tb.vhd* et le script *run_comp_bin_lin_3to8_tb.tcl*

3) Réaliser l'intégration de la description par flot de données du transcodeur Bin-Lin avec le top spécifique pour la carte Max-V: *maxv_top_3to8.vhd*. Vous devez travailler dans le répertoire /pr. Le circuit disponible sur la carte Max-V est le 5M570ZF256C5. **Remarque** : ne pas oublier d'effectuer l'assignation des pins du CPLD.

4) Vous devez analyser et commenter le résultat de la synthèse, pour le composant *bin_lin_3to8*, en consultant les vues RTL et Technology. Vous devez donner la quantité de logique utilisée pour le composant *bin_lin_2to4*.

   Quantité de logique voir dans Quartus « Compilation Report » :
   Analsis&Synthetis > Resource Utilization by Entity

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

5) Programmer le circuit 5M570ZF256C5 d'une maquette MaxV 80p-25p avec le fichier **.pof* généré lors du point précédent.

6) Faire valider l'intégration de votre Bin-Lin 3 à 8 par le professeur ou l'assistant.