TP 2 – Fonctions combinatoires

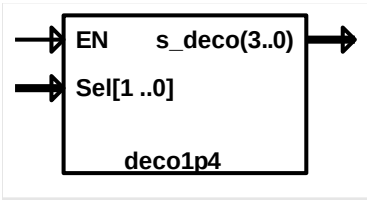
Objectifs :

- Instructions concurrentes d'affectation
- Écriture de testbench
- Instanciation de composant

Exercice 1 : Préparation en salle

1) Décodeur 1 parmi 4 avec enable

On souhaite réaliser la fonction de décodage 1 parmi 4 avec une entrée de validation enable.



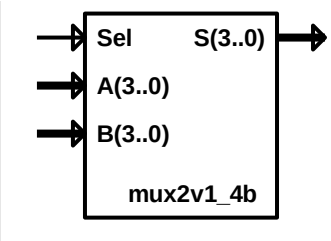
Entrées / sorties du composant :

- l'entrée `EN` est une autorisation de fonctionnement, active à '1'
- le vecteur `Sel[1..0]` représente les entrées de sélection
- la sortie est `s_deco[3..0]`, active à l'état haut

1. Quel est le rôle d'une autorisation de fonctionnement ?
2. Donner la table de vérité du composant, en s'inspirant du TD et en prenant en compte l'entrée EN.
3. Comment tester le composant de manière exhaustive ? Compléter les chronogrammes sur la feuille distribuée en séance.

2) Multiplexeur 2 voies de 4 bits

On souhaite réaliser la fonction de multiplexage de 2 voies de 4 bits.



Entrées / sorties du composant :

- l'entrée `Sel` de choix (`A` est recopiée si `Sel` vaut '0', `B` si `Sel` vaut '1')
- 2 entrées `A` et `B` de donnée sur 4 bits
- la sortie `S[3..0]`.

1. Rappeler la table de vérité du composant.
2. Comment tester le composant ? Compléter les chronogrammes sur la feuille distribuée en séance.

Exercice 2 : Décodeur 1p4

Téléchargez le projet TP_2_Ex_2 depuis Moodle, et décompressez le dans votre dossier personnel. Ouvrez-le à l'aide de Quartus 18.1.

1) Code

Créez un fichier en utilisant le menu « File ➔ New… ➔ VHDL File », et enregistrez ce fichier sous le nom **deco1p4.vhd** dans le dossier **src** du projet.

On souhaite décrire la fonction en codant la table de vérité rappelée à l'exercice 1.

Dans le fichier nouvellement créé, ajoutez :

- Les appels de bibliothèque adaptés,
- L'entité du composant en respectant les noms des entrées/sorties vus à l'exercice 1,
- L'architecture du composant à l'aide de l'instruction adaptée.

2) Synthèse

Placez le composant **deco1p4** en « top level » et synthétisez-le avec les commandes vues dans le TP1.

3) Simulation

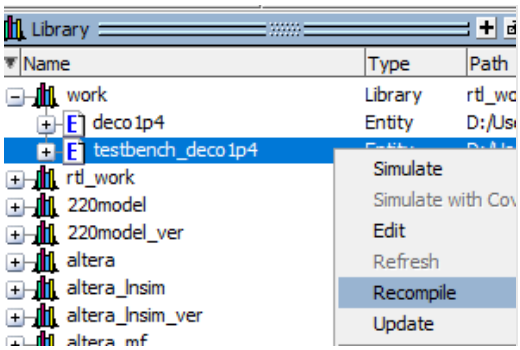
Rappel : accédez au menu « Tools ➔ Options », puis dans la catégorie « EDA Tool Options », coller le texte suivant sur la ligne « ModelSim-Altera » (**PAS sur la ligne « ModelSim » tout court**) :
`C:\intelFPGA\18.1\modelsim_ase\win32aloem`

Lancez ModelSim depuis le menu « Tools è Run Simulation Tool è RTL Simulation ».

Ouvrez le fichier **testbench_deco1p4.vhd** (File ==> Open). Complétez-le de manière à ce qu'il réalise le chronogramme vu à l'exercice 1.

1. Déclarez les signaux nécessaires pour générer les stimuli d'entrée
2. Instanciez le composant à tester (note : dans un testbench, il n'est pas nécessaire de relier les sorties du composant testé)
3. Générez les stimuli : commencez par la valeur initiale des signaux, puis alternez attentes et changement de valeur des signaux. Finissez le process par l'instruction `wait`.

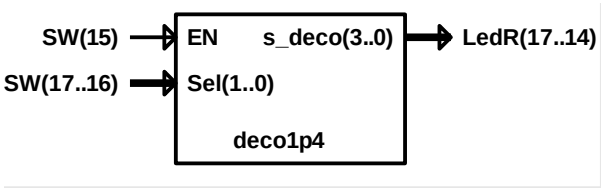
Compilez le fichier. À chaque fois que vous ferez des changements à ce fichier, vous devrez le recompiler (attention, il n'y a pas de sauvegarde automatique lors d'une recompilation). Une fois le fichier compilé une première fois, plutôt que d'utiliser le menu **Compile ==> Compile**, vous pouvez utiliser **Recompile** :



En reprenant au besoin les consignes données lors du TP 1, simulez le composant avec votre testbench. Vérifiez que le comportement est bien correct. Faites valider les réponses à la préparation et le résultat de la simulation.

4) Test sur site

Les connexions sont les suivantes :



En reprenant au besoin les consignes données lors du TP 1, vérifiez le bon fonctionnement du multiplexeur, et faites valider. Vous devrez être capable d'expliquer le comportement à l'enseignant.

Exercice 3 : multiplexeur 2 voies 4 bits

Téléchargez le projet TP_2_Ex_3 sur Moodle et ouvrez-le à l'aide de Quartus.

1) Code

Créez un fichier **mux2v1_4b.vhd**.

On souhaite décrire la fonction en codant la table de vérité rappelée à l'exercice 1.

Réalisez le code de l'entité pour qu'elle corresponde à la vue externe. Attention à bien respecter les noms des signaux.

Codez l'architecture du composant à l'aide de l'instruction adaptée.

2) Synthèse

Placez le composant **mux2v1_4b** en « top level » et synthétisez-le avec les commandes vues dans le TP 1.

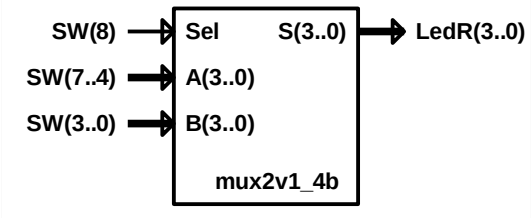
3) Simulation

Lancez ModelSim. Ouvrez le fichier **testbench_mux2v1_4b.vhd**. Complétez-le pour réaliser le chronogramme de l'exercice 1.

En reprenant au besoin les consignes données lors du TP 1, simulez le composant avec votre testbench. Vérifiez que le comportement est bien correct. Faites valider.

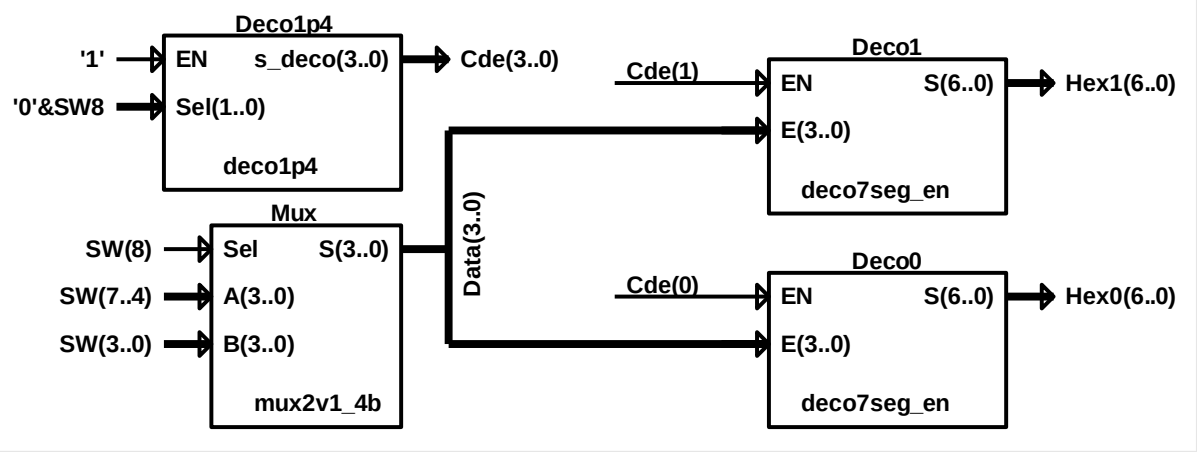
4) Test sur site

Les connexions sont les suivantes :



En reprenant au besoin les consignes données lors du TP 1, vérifiez le bon fonctionnement du multiplexeur, et faites valider. Vous devrez être capable d'expliquer le comportement à l'enseignant.

Exercice 4 – Affichage multiplexé



1) Réflexion préliminaire

- Comprendre le schéma
  - Le composant deco7seg_en fonctionne comme le deco7seg que vous avez réalisé quand en='1'. Lorsque en='0', toutes les sorties sont à '1' : quel est l'effet recherché ?
  - Que vaut Cde si SW8=0 ? Quel afficheur est allumé ? Qu'affiche-t-il ?
  - Même question si SW8=1.
  - Finalement, que verra l'utilisateur si SW8 alterne rapidement entre 0 et 1 ?
- Comment simuler ?
- Comment réaliser ce composant en VHDL ?

2) Réalisation

En utilisant le projet TP_2_Ex_4, créez le fichier **aff_mux.vhd** dans lequel vous réaliserez le schéma ci-dessus.

3) Synthèse

Dans la liste des fichiers du projet, vous devez ajouter tous les fichiers utilisés par le composant aff_mux : deco7seg_en, deco1p4 et mux2v1_4b. Ne dupliquez pas les fichiers, mais ajoutez les fichiers en allant les chercher dans les dossiers TP_2_Ex_2 et TP_2_Ex_3. Placez le composant **aff_mux** en « top level » et synthétisez-le avec les commandes vues dans le TP1.

L'outil **Tools ➔ Netlist Viewer ➔ RTL Viewer** de Quartus vous permettra de vérifier si vos connexions sont bien réalisées.

4) Simulation

Lancez Modelsim. Si des messages d'erreur apparaissent dès l'ouverture, c'est que ModelSim n'a pas réussi à compiler certains fichiers. Vous devez donc compiler (Compile / Compile…) les fichiers indiqués dans les messages, puis compiler aff_mux. Ouvrez le fichier **testbench_aff_mux.vhd**. Complétez-le pour réaliser les tests vus à la question 1.

5) Test sur site

Vérifiez le bon fonctionnement du multiplexeur, et faites valider. Vous devrez être capable d'expliquer le comportement à l'enseignant.

Exercice bonus –