TP 3 – Fonctions logiques séquentielles

Objectifs :

- Instructions VHDL séquentielles
- Écriture de testbench
- Instanciation de composant

Règles de simulation de fonctions logiques séquentielles

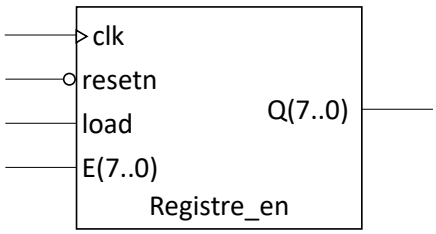
Le principe fondamental qui doit vous guider est de prouver le bon fonctionnement du composant de manière logique et cohérente. Il ne faut surtout pas faire bouger les entrées dans un ordre aléatoire, sans objectif.

C'est pour cela qu'une simulation devra d'abord vérifier que la fonctionnalité principale est correcte (par exemple pour un compteur, qu'il … compte toute la plage de valeurs prévues lorsqu'il est autorisé à compter, qu'il mémorise la valeur quand il n'est pas autorisé à compter). Vous devrez ensuite tester le bon fonctionnement des entrées annexes (typiquement raz synchrone, chargement en vérifiant que les priorités définies sont respectées). Dans le cas où une entrée est sur plusieurs bits, vous devez obligatoirement tester différentes valeurs de cette entrée.

Exercice 1 : Préparation en séance

1) Registre simple avec enable

On souhaite réaliser un registre sur 8 bits avec autorisation de chargement et réinitialisation asynchrone. La vue externe du composant sera la suivante :



1. D'après le schéma, quel est le niveau d'activité des différentes entrées ?
2. Rappeler la priorité entre les différentes entrées.
3. Comment simuler ce composant ?

2) Registre à décalage

On souhaite construire un registre à décalage, non circulaire, sur 8 bits, qui disposera, par ordre de priorité décroissante, de :

- une entrée de réinitialisation asynchrone active à l'état bas.
- une autorisation de chargement active à l'état haut
- une autorisation de décalage active à l'état haut
- une entrée de choix du sens de décalage (1 = droite ; 0 = gauche)

1. Rappelez le principe d'un registre à décalage.
2. Dessinez la vue externe du composant en choisissant des noms pertinents pour le composant et ses entrées/sorties.

3. Dressez la table de vérité de ce composant.
4. Comment simuler ce composant ?

3) Compteur

On souhaite réaliser un compteur sur 4 bits qui disposera, par ordre de priorité croissante, de :

- une entrée de réinitialisation asynchrone active à l'état bas,
- une autorisation de chargement active à l'état haut,
- Une autorisation de comptage active à l'état haut.

1. Dessinez la vue externe du composant
2. Comment simuler ce composant ?

Exercice 2 : Finir le TP 2 (optionnel)

Si vous n'avez pas terminé le TP 2, vous pouvez reprendre l'exercice 4 du TP et faire le test sur site.

Il n'est pas nécessaire de faire la partie simulation, sauf si vous aviez déjà bien avancé sur ce point et souhaitez également le valider.

Exercice 3 : Registre simple avec enable

Téléchargez le projet **TP_3** 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 **registre_en.vhd** dans le dossier **src** du projet.

Dans le fichier nouvellement créé, décrivez le code du registre.

2) Synthèse

Afin de vérifier la syntaxe de votre code, synthétisez le composant comme vu dans les TP précédents. Corrigez les éventuelles erreurs.

3) Écriture du testbench

Rappel : accédez au menu « Tools ➔ Options », puis dans la catégorie « EDA Tool Options », saisissez le chemin ci-dessous sur la ligne « ModelSim-Altera » :

- Si vous êtes dans la salle C01 ou la salle D02 :

```
C:\intelFPGA\18.1\modelsim_ase\win32aloem
```

- Si vous êtes dans la salle C04 :

```
C:\applis\altera\11.1\modelsim_ase\win32aloem
```

Lancez ModelSim.

Créez un fichier **testbench_registre_en.vhd** (File ➔ New ➔ Source ➔ VHDL) et sauvegardez le dans le dossier **src** du projet.

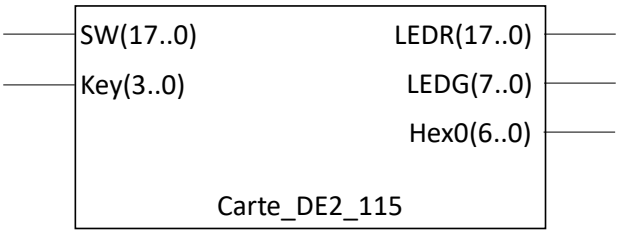
Compilez le fichier. À chaque fois que vous ferez des changements à ce fichier, vous devrez le recompiler (attention, il n'y a pas de sauvegarde automatique lors d'une recompilation). Une fois le fichier compilé une première fois, plutôt que d'utiliser le menu **Compile ➔ Compile**, vous pouvez utiliser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

4) Simulation

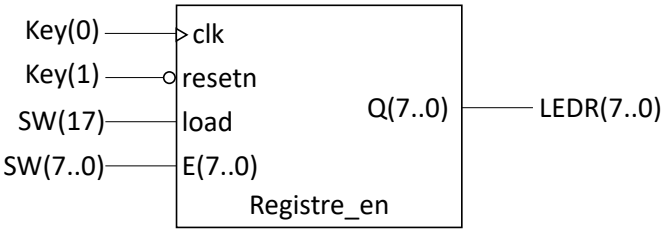
Simulez le composant avec votre testbench. Vérifiez que le comportement est bien correct. Faites valider.

5) Test sur site

Dans Quartus, créez un nouveau composant que vous nommerez **Carte_DE2_115**. La vue externe de ce composant devra être la suivante :



Dans ce composant, instanciez le registre avec la correspondance suivante sur les entrées/sorties :



Placez ce nouveau composant en top level, compilez et chargez sur la platine. Faites valider.

Exercice 4 : Registre à décalage

1) Code

Dans le même projet que l'exercice 1, créez un fichier VHDL qui portera le nom que vous avez choisi dans la préparation et sauvegardez-le dans le dossier **src**.

Codez le composant.

2) Synthèse

Synthétisez le composant créé afin d'en vérifier la syntaxe.

3) Écriture du testbench

Lancez ModelSim. Créez un fichier de testbench dont le nom respecte la nomenclature utilisée jusqu'à présent**.**

4) Simulation

Simulez le composant avec votre testbench. Vérifiez que le comportement est bien correct. Faites valider.

5) Test sur site

Dessinez la vue interne du composant **Carte_DE2_115** en ajoutant, en plus du registre de l'exercice précédent, le registre à décalage avec les connexions ci-dessous. Laissez de la place autour de votre dessin, on ajoutera encore de nouveaux composants par la suite !

Connexions du registre à décalage :

- L'horloge sur `Key(0)`,
- Le signal de réinitialisation sur `Key(1)`,
- L'autorisation de chargement sur `SW(16)`,
- L'autorisation de décalage sur `SW(15)`,
- Le choix du sens sur `SW(14)`,
- L'entrée de chargement sur `SW(7..0)`,
- La sortie Q sur `LEDR(15..8)`.

Reprenez le composant **Carte_DE2_115**, et modifiez-le pour ajouter le nouveau composant.

Testez sur site et faites valider. Vous devrez présenter l'ensemble des comportements possibles.

Exercice 5 – Compteur

1) Réalisation

Toujours dans le même projet, créez un fichier VHDL pour le compteur, et synthétisez-le.

2) Simulation

Lancez ModelSim. Créez un testbench, et testez l'ensemble des comportements possibles du compteur. On devra notamment voir un cycle complet ininterrompu, un cycle interrompu par une réinitialisation, mettre en évidence le mode mémoire, et présenter le chargement.

3) Test sur site

Reprenez le dessin du composant **Carte_DE2_115**, et ajoutez le compteur avec les connexions suivantes :

- L'horloge sur `Key(0)`,
- Le signal de réinitialisation sur `Key(1)`,
- L'autorisation de comptage sur `SW(13)`,
- L'autorisation de chargement sur `SW(12)`,
- L'entrée de chargement sur `SW(3..0)`,
- La sortie `Q` sur `LEDG(3..0)`,
- La sortie `Q` sur l'entrée d'un décodeur 7 segments (voir TP 1),
- La sortie du décodeur 7 segments sur `Hex0`.

Copiez-collez le fichier **deco7seg.vhd** que vous avez créé dans le premier TP dans le dossier **src** du projet actuel, et ajoutez-le au projet Quartus.

Réalisez les connexions du dessin et compilez. Utilisez l'outil **Tools ➔ Netlist Viewer ➔ RTL Viewer** de Quartus pour étudier le schéma synthétisé. Ouvrez chaque composant et identifiez les primitives utilisées dans chacun.

Chargez sur la platine et faites valider la liste des primitives et le fonctionnement sur site.

Exercice bonus – Compteur/décompteur

Modifiez le fichier de l'exercice précédent pour ajouter une entrée permettant de choisir entre comptage et décomptage.

Modifiez le testbench de l'exercice précédent pour ajouter à la fin le test du comportement en décomptage et faites valider.

Pour le test sur site, reliez l'entrée permettant de choisir entre comptage et décomptage à `SW(11)`, testez et faites valider.