

Chapitre 3

Systèmes séquentiels

Les circuits logiques sont classés en deux catégories : combinatoires et séquentiels. Un circuit combinatoire se caractérise par le fait que l'état de ses sorties ne dépend que de l'état de ses entrées. C'est le cas du transcodeur hexadécimal vers 7 segments étudié précédemment.

Par contre, l'état futur des sorties d'un système séquentiel dépendra de l'état des entrées mais aussi de l'état actuel des sorties. C'est le cas des feux tricolores : si le feu est actuellement vert, il passera tout à l'heure à l'orange ; s'il est rouge, il passera au vert, ... Il passe de façon séquentielle, du vert à l'orange, de l'orange au rouge et du rouge au vert.

Pour réaliser un circuit séquentiel, il faut mémoriser l'état présent, et s'en servir pour le calcul de l'état futur avec une fonction combinatoire. Ce qui donne le schéma fonctionnel suivant, connu sous le nom de machine de Moore.

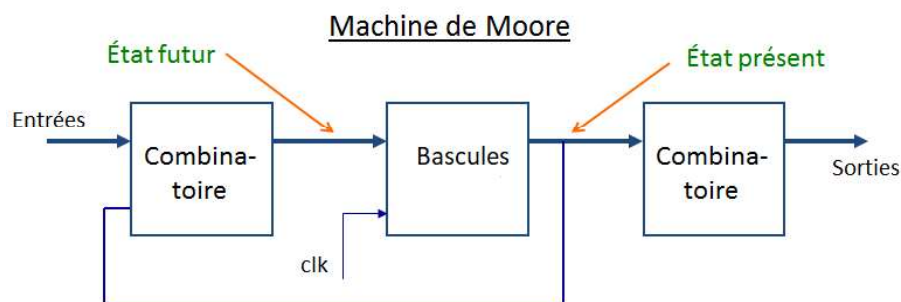


FIGURE 3.1 – Machine de Moore

3.1 Le compteur décompteur entre 1 et 6

On souhaite réaliser un circuit séquentiel qui compte et décompte de 1 à 6. Le système est compteur si $up/dn = 1$ et décompteur si $up/dn = 0$. Le schéma de principe du compteur-décompteur est représenté sur la figure suivante :

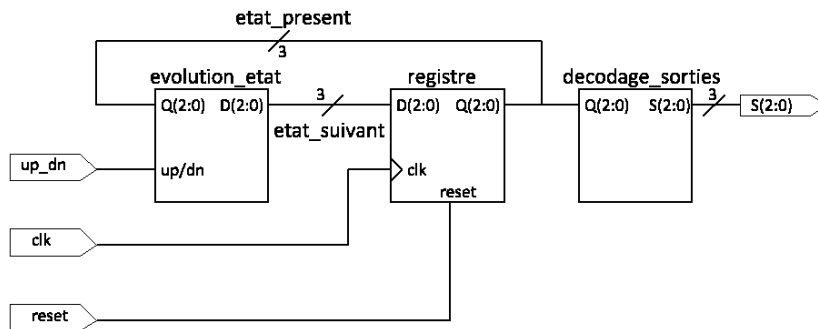


FIGURE 3.2 – Schéma de principe du compteur décompteur

- **Créer** un nouveau projet, nommé *compteur_decompteur*.

3.1.1 Circuit *evolution_etat*

- **Créer** un nouveau circuit nommé *evolution_etat*. Reprendre la table de vérité des états du compteur de 1 à 6 établie en Cours, la **compléter** en ajoutant une entrée nommée *up_dn* au système. **Réaliser** le circuit *evolution_etat* à partir de sa table de vérité.
- **Simuler** son fonctionnement.

3.1.2 Circuit *registre*

- **Créer** un nouveau circuit nommé *registre*.
- En utilisant les bascules D disponibles dans Logisim-Evolution, **câbler** l'architecture de l'entité *registre*.

3.1.3 Simulation du circuit *registre*

On souhaite pour ce circuit séquentiel visualiser l'évolution des signaux et afficher les chronogrammes.

- Il faut d'abord extraire le fichier vhd correspondant au circuit *registre*.
 - **Sélectionner** *FPGA >> Synthétiser et Télécharger ...*
 - **Choisir** *Générer le HDL uniquement* dans la rubrique *Action*.

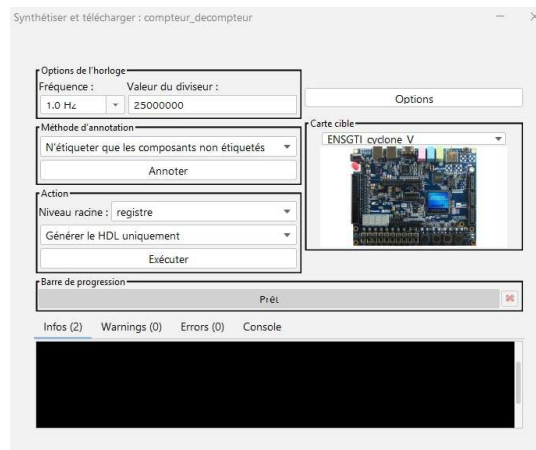


FIGURE 3.3 – Génération des fichiers vhd

- **Lancer** le logiciel *Questa Altera Starter FPGA Edition*
- **Créer** un nouveau projet : File >> New >> Project...

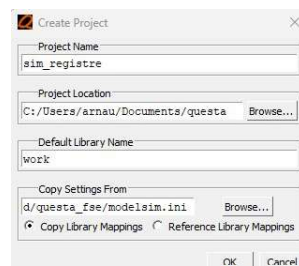


FIGURE 3.4 – Création du projet sous Questa

- **Ajouter** les 4 fichiers vhd précédemment créés dans les dossiers *circuit* et *memory* : Add Existing File

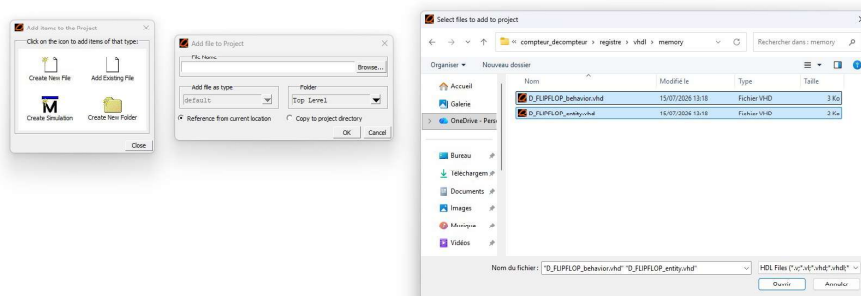


FIGURE 3.5 – Ajouter des fichiers vhd au projet sous Questa

- **Analyser** le contenu des 4 fichiers vhd.
- **Compiler** ces 4 fichiers. Il faudra **exécuter** 2 fois la compilation pour ne pas avoir d'erreur liée à l'ordre de la compilation.

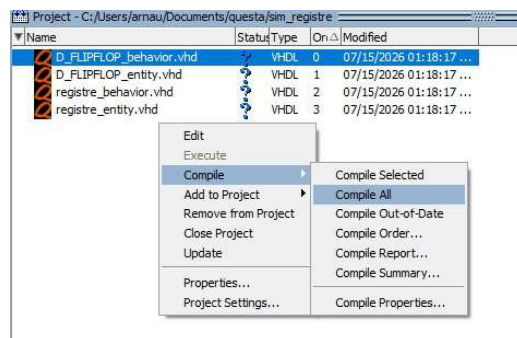


FIGURE 3.6 – Compilation des fichiers

- Lancer la simulation : *Simulate >> Start Simulation*
- Choisir l'entité *registre* dans la library *work*

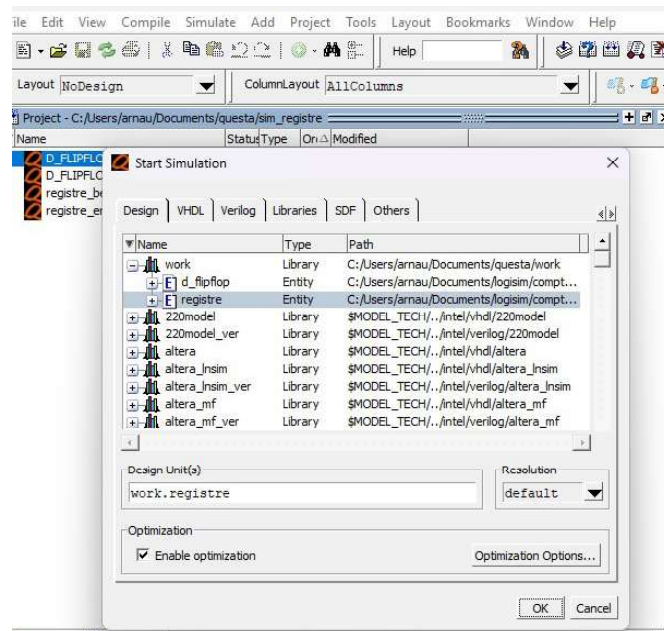


FIGURE 3.7 – Choix de l'entité à simuler

- Choisir les signaux à ajouter aux chronogrammes : *Add Wave (Ctrl + W)*

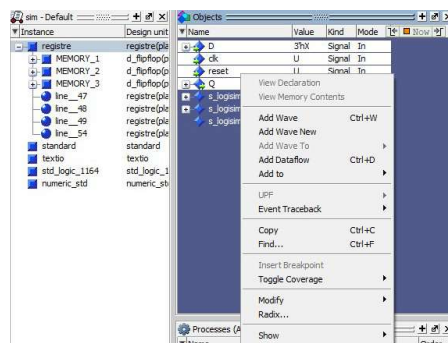


FIGURE 3.8 – Choix des signaux à analyser

- **Forcer** une valeur sur l'entrée *D* et pour le signal *reset* avec un clic-droit sur le nom du signal puis *Force...*

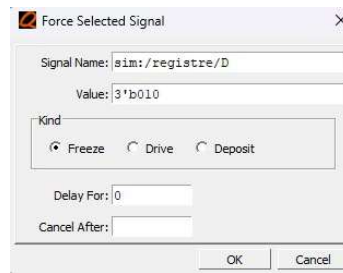


FIGURE 3.9 – Fixer une valeur à un signal d'entrée

- **Forcer** une horloge sur le signal *clk* avec un clic-droit sur le nom du signal puis *Clock...*

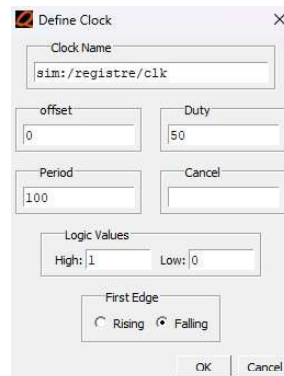


FIGURE 3.10 – Choix du signal d'horloge

- **Lancer** la simulation avec le bouton *run* à droite de la valeur du temps à simuler.

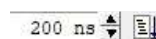


FIGURE 3.11 – Lancement de la simulation

- **Visualiser** et **valider** la simulation sur le chronogramme.

3.1.4 Circuit decodage_sorties

- **Créer** un nouveau circuit nommé *decodage_sorties*. **Réaliser** le circuit *decodage_sorties* à partir de sa table de vérité.
- **Simuler** son fonctionnement.

3.1.5 Compteur décompteur complet

- **Créer** un nouveau circuit nommé *compteur_decompteur*.

- **Réaliser** le circuit *compteur_decompteur* à partir des entités *evolution_etat*, *registre* et *decodage_sorties* selon le schéma de la figure 3.2.
- **Simuler** le circuit *compteur_decompteur* avec Questa.
- **Implémenter** le composant cible et **tester** le fonctionnement sur les LEDs de la maquette.
- Que constatez-vous ? Comment remédier à ce problème ?

3.1.6 Diviseur d'horloge

- Un compteur binaire à N bits peut être utilisé comme diviseur de fréquence.

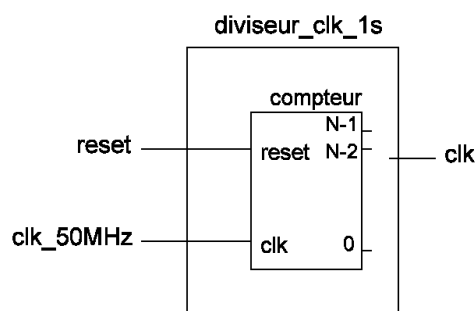


FIGURE 3.12 – Schéma de principe du diviseur

- **Créer** un circuit *diviseur_clk_1s*.
- Pour réaliser ce diviseur, **utiliser** le composant *Compteur* dans la librairie *Memoire/Sequentiel* de Logisim.
- **Configurer** et **câbler** le composant pour réaliser une division d'horloge par 8 en utilisant l'aide de Logisim : *Aide >> Guide Utilisateur >> Référence de la bibliothèque >> Memory Library >> Compteurs*
- **Simuler** et **valider** le fonctionnement grâce à Questa.
- La fréquence de la maquette Cyclone V GX Starter Kit est 50MHz. **Configurer** le compteur pour avoir une fréquence de sortie de 1Hz. **Utiliser** dans ce cas la possibilité de fixer la valeur maximum du comptage.

3.1.7 Compteur décompteur complet avec horloge à 1Hz

- **Compléter** le circuit *compteur_decompteur* avec le circuit *diviseur_clk_1s*.

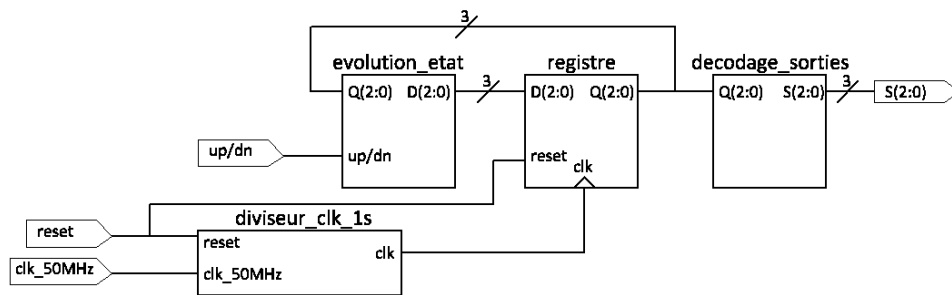


FIGURE 3.13 – Schéma de principe du compteur décompteur avec le diviseur d'horloge

- **Implémenter** le composant cible et **tester** le fonctionnement sur les LEDs de la maquette.

3.2 Compteur - décompteur et Affichage

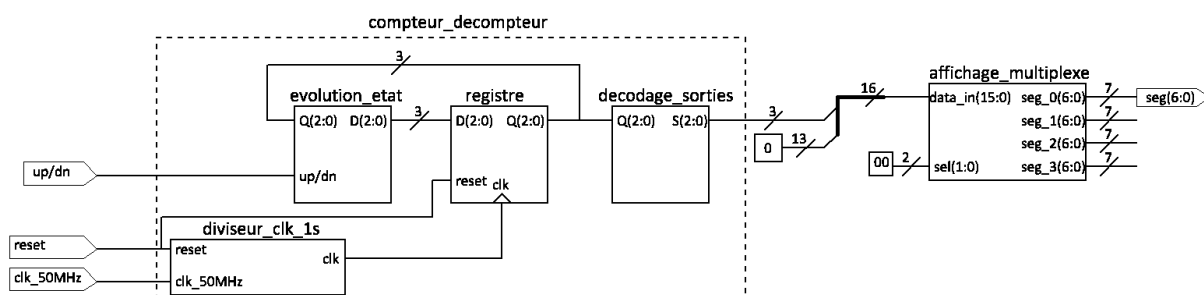


FIGURE 3.14 – Schéma de principe du compteur décompteur avec affichage

- **Créer** un nouveau projet *compteur_decompteur_afficheur*.
- **Ajouter** la librairie du TP précédent : *affichage_multiplexe.circ*, dans laquelle se trouve le circuit *affichage_multiplexe*.
- **Ajouter** la librairie du TP précédent : *compteur_decompteur.circ*, dans laquelle se trouve le circuit *compteur_decompteur*.
- **Réaliser** le circuit *compteur_decodeur_afficheur* à partir des entités *compteur_decompteur* et *affichage_multiplexe*, selon le schéma précédent.
- **Implémenter** le composant cible et **tester** le fonctionnement sur la maquette.

3.3 Retour sur le projet transcodeur

On souhaite reprendre et améliorer le transcodeur hexadécimal - 7 segments réalisé au TP précédent. Pour cela, on voudrait que l'entrée *sel* (qui permet de sélectionner l'afficheur) évolue régulièrement, sans avoir à la commander par un bouton poussoir.

3.3.1 Génération des signaux de sélection : sel(1 :0)

L'évolution de l'entrée *sel* permet de choisir l'afficheur et les données qui le concernent. Il faut faire évoluer la valeur *sel* de façon suffisamment rapide pour balayer les 4 afficheurs, sans que l'œil humain ne s'aperçoive du clignotement. On choisit une période de rafraîchissement de l'ordre de 2.6 ms . Un diviseur de fréquence permet d'obtenir cette fréquence à partir de l'horloge de la maquette $f_{clk-50MHz} = 50MHz$.

- Sachant que la période de *clk* vaut $T_{clk} = 20ns$, une division de fréquence 2^{17} permet de fournir un signal de période $T_{sel} = 2^{17} \cdot 20ns = 2,6ms$.

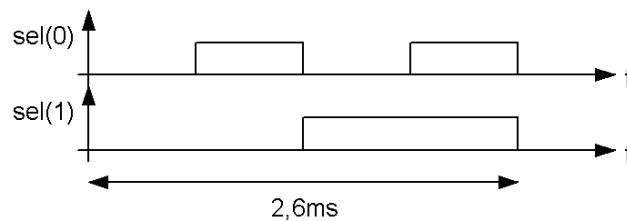


FIGURE 3.15 – Chronogramme des signaux *sel*

- Un compteur binaire à 17 bits (16 : 0) est utilisé comme diviseur de fréquence. La sortie sel(1) est connectée au bit 16 du compteur, la sortie sel(0) au bit 15.

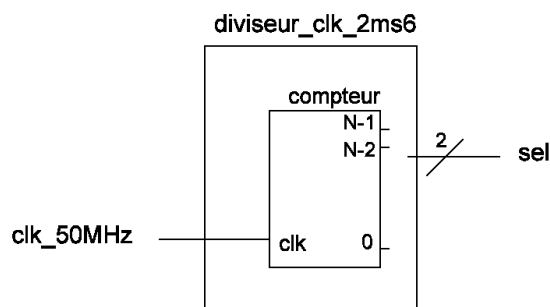


FIGURE 3.16 – Schéma de principe du diviseur

- **Copier** le projet *affichage_multiplexe* en *affichage_sequentiel_multiplexe*.
- **Créer** un nouveau circuit nommé *diviseur_clk_2ms6*.

- En utilisant un compteur disponible dans Logisim-Evolution, **câbler** l'architecture de l'entité *diviseur_clk_2ms6*.

3.3.2 Projet décodeur complet

- **Créer** un nouveau circuit *affichage_sequentiel_multiplexe* dont l'entité et l'architecture sont représentées par la figure suivante :

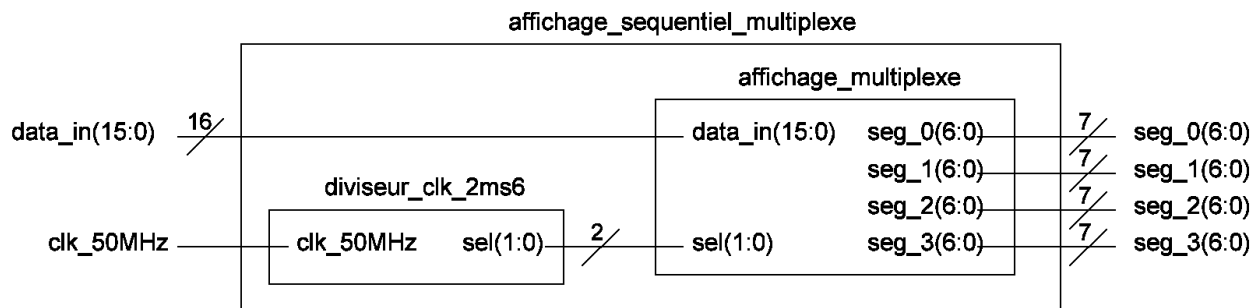


FIGURE 3.17 – Schéma de principe du décodeur complet

- **Simuler** le projet avec Questa.
- **Implémenter** le composant cible et **tester** le fonctionnement sur la maquette.