

Travaux pratiques d'électronique:

Électronique digitale

S. Orsi, E. Corrin
Lundi 9 novembre 2009

Ce labo a pour but de comprendre comment connecter de simples dispositifs d'entrée/sortie avec une puce FPGA et bâtir un circuit qui utilise ces éléments. Vous utiliserez les interrupteurs SW17-0 sur la carte DE2 comme entrées à votre circuit. Vous utiliserez des LEDs et les afficheurs à 7 segments comme sorties.

Il est recommandé de lire attentivement l'introduction à ce labo, disponible sur le site web.

Rappel: Lors de la première utilisation de Quartus, il est possible que vous receviez un message d'erreur concernant une license manquante. Il suffira alors de choisir l'option permettant de choisir un fichier de license, ensuite de spécifier le fichier "{longue série de chiffres et de lettres}.dat" situé dans le répertoire "quartus".

1 Logique simple

La carte DE2 est équipée de 18 interrupteurs à bascule, appelées SW17-0, qui peuvent être utilisées comme entrées dans un circuit, et 18 LEDs rouges, appelées LEDR17-0, qui peuvent être utilisées pour afficher les valeurs de sortie. La figure 1 montre un exemple de code VHDL simple qui utilise ces interrupteurs et affiche leur état sur les LEDs. La carte DE2 possède des connections imprimées entre la puce FPGA et les interrupteurs et les LEDs. La procédure pour réaliser un brochage, tel que décrit par l'assistant au début du cours, est également indiquée dans le tutoriel "Quartus II Introduction using VHDL Design".

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Simple module that connects the SW switches to the LEDR lights
ENTITY part1 IS
    PORT ( SW : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
          LEDR : OUT STD_LOGIC_VECTOR(17 DOWNTO 0)); -- red LEDs
END part1;

ARCHITECTURE Behavior OF part1 IS
BEGIN
    LEDR <= SW;
END Behavior
```

Figure 1: Un exemple de code VHDL utilisant les interrupteurs et les LEDs de la carte DE2

1. Créez un nouveau projet avec Quartus II en suivant les instructions.
2. Dessinez avec Quartus un schéma pour la logique suivante: $(A \text{ AND } B) \text{ OR } [(NOT A) \text{ AND } (NOT B)]$.

3. Obtenez le code VHDL pour le design que vous venez de réaliser et modifiez-le pour obtenir la logique suivante: $(A \text{ AND } B) \text{ OR } [A \text{ OR } (\text{NOT } B)]$.
4. Écrivez la table de vérité pour votre design.
5. Simulez votre design et vérifiez la table de vérité précédente. Expliquez la différence entre les simulations “timing” et “functional”.
6. Programmez le FPGA Cyclone II sur la carte DE2 et vérifiez vos résultats en jouant avec les interrupteurs.

2 Multiplexeurs

En électronique, un multiplexeur (ou “MUX”) est un dispositif qui choisit un des signaux d’entrée (analogiques ou digitaux) et le dirige dans une seule ligne de sortie. Un multiplexeur permet à plusieurs signaux de partager un seul dispositif coûteux ou une autre ressource (par exemple un convertisseur analogique-digital ou une ligne de communication) plutôt que d’utiliser un dispositif par signal d’entrée. Vous allez réaliser un multiplexeur “2 vers 1”.

1. Créez un nouveau projet Quartus II
2. Réalisez un schéma (avec des portes ET, OU et NON) avec 3 entrées: 2 signaux x,y (SW0, SW1) et une ligne de sélection s (SW17). La sortie m (LEDG0) correspondra à x si $s=1$ et à y si $s=0$.
3. Écrivez la table de vérité pour votre design.
4. Simulez votre design et vérifiez la table de vérité précédente.
5. Programmez le FPGA Cyclone II sur la carte DE2 et vérifiez vos résultats en jouant avec les interrupteurs.

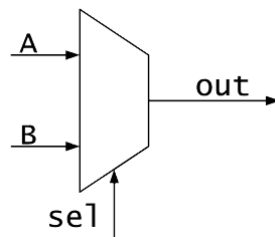


Figure 2: Symbole du multiplexeur

En connectant plusieurs multiplexeurs “2 vers 1”, on peut réaliser des circuits plus complexes. La figure 3 montre un multiplexeur “2 vers 1” de 8 bits, la figure 4 montre un multiplexeur “5 vers 1” et la figure 5, un multiplexeur “5 vers 1” de 3 bits.

6. Ajoutez à votre projet existant un multiplexeur “2 vers 1” de 4 bits en utilisant les interrupteurs comme entrées et affichez la sortie sur LEDR13-16.

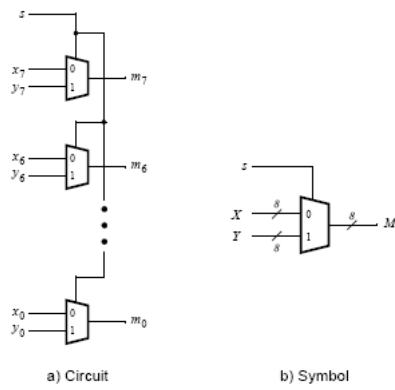


Figure 3: Multiplexeur "2 vers 1" de 8 bits

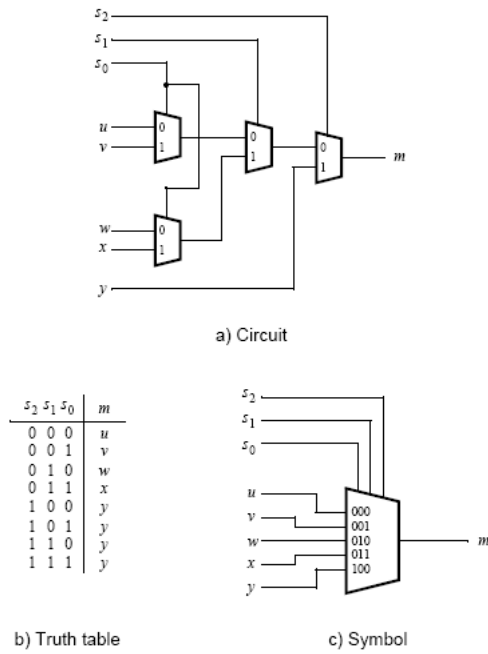


Figure 4: Multiplexeur "5 vers 1"

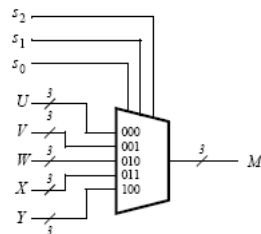


Figure 5: Multiplexeur "5 vers 1" de 3 bits

3 Additionneur

Note: Afin d'utiliser SW17-0 et LEDR17-0, il est nécessaire d'inclure dans votre projet Quartus II le brochage exact ("pin assignment"), qui vous sera donné dans le classeur d'instructions de labo. Par exemple, le manuel spécifie que SW0 est connecté à la broche N25 du FPGA et que LEDR0 est connecté à la broche AE23. [Importez le fichier "DE2_pin_assignments.csv" dans le logiciel Quartus II. Le brochage dans ce fichier est utile seulement si les noms des broches listés dans le fichier sont exactement les mêmes que les noms de ports utilisés dans votre entité VHDL.]

1. Réalisez avec Quartus II un "additionneur complet" qui prend 3 entrées de 1 bit (2 nombres, SW0 et SW8, et 1 "retenue", SW1) et affichez les résultats sur LEDG0 et LEDG1 en utilisant seulement des portes MUX et OU-exclusif (XOR). [Voir l'intro par E. Cortina, pages 10-11]
2. Simulez votre design et comparez vos résultats avec la table de vérité attendue.
3. Programmez le FPGA Cyclone II sur la carte DE2 et vérifiez vos résultats.
4. Réalisez un "additionneur parallèle" qui prend 2 entrées de 8 bits (SW0-SW7 et SW8-SW15) et affichez les résultats sur LEDR0-LEDR8 en utilisant seulement des portes MUX et OU-exclusif (XOR).
5. Simulez votre design et comparez vos résultats avec la table de vérité attendue.
6. Programmez le FPGA Cyclone II sur la carte DE2 et vérifiez vos résultats en jouant avec les interrupteurs.
7. Créez un objet á partir du programme que vous venez d'ecrire: créez un fichier .bsf (export - ...); incluez le fichier dans la bibliothèque [library] (assignments - settings - ...). Pour utiliser l'objet après, choisissez: 'insert object' et après 'choose from file'.

4 Affichage digital

La figure 6 montre un décodeur de 7 segments avec 7 sorties qui sont utilisées pour afficher un caractère sur un affichage à 7 segments. Les 7 segments de l'affichage sont identifiés par les chiffres 0-6 comme indiqué sur la figure. Chaque segment est illuminé lorsque la valeur logique 0 lui est assignée.

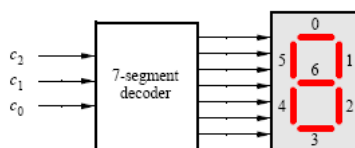


Figure 6: Décodeur de 7 segments

1. Ouvrez le projet “affich_digit”.
2. Si une entrée de 3 bits (c_2, c_1, c_0) est utilisée, comme dans la figure, combien de caractères différents peuvent être affichés sur chaque affichage?
3. Afin de garder le design le plus simple possible, vous utiliserez 2 entrées (c_1, c_0) et vous afficherez des nombres en base 4. Rappelez-vous qu’en base 4, les relations suivantes sont respectées: $1+2=3$, $2+2=10$, $10+30=100$ et $(33)_4=(15)_{10}=(1111)_2$, où le chiffre après la parenthèse indique la base utilisée. Faites une table avec les colonnes suivantes: les 2 entrées c_1, c_0 , le nombre à afficher, et les 7 segments d’affichage d_0-d_6 . Par exemple, la paire $c_1=0, c_0=0$ correspondra au nombre 0, qui sera affiché comme suit pour d_6-d_0 : 0000001 (0: on; 1: off).
4. Évaluez les 3 rangées restantes de la table.
5. Évaluez les expressions logiques pour les segments d_0-d_6 (exemple: $d_0 = \bar{c}_1 c_0$).
6. Ajoutez à votre projet existant un décodeur de 7 segments (d_0-d_6) avec 2 entrées (c_0, c_1), de façon à représenter la table de vérité que vous venez de réaliser. Utilisez seulement les fonctions logiques simples (ET, OU, OU-exclusif, NON). Prenez soin d’éteindre les autres affichages.
7. Tester le décodeur avec une entrée de 2 bits.
8. Améliorez votre design de décodeur de 7 segments en utilisant une entrée de 4 bits.
9. Affichez le résultat de l’exercice précédent (“additionneur parallèle” à 8 bits) sur la carte. Vous n’avez pas besoin de simuler votre design. Copiez-collez les logigrammes requis pour ce dernier design, ou utilisez l’objet créé avant.
10. (Facultatif) Faites défiler les chiffres sur l’affichage de la droite vers la gauche.

5 Bascules et compteur

Les bascules (flip-flops) sont des dispositifs très utiles entre autres dans la construction des mémoires d’ordinateurs. Ils ont deux états stables et ils oscillent entre ces derniers selon des critères différents, pour différents types de bascules.

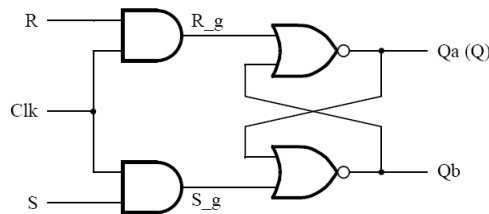


Figure 7: Bascule SR

1. Ouvrez un nouveau projet et réalisez le circuit de la figure 7.
2. Écrivez la table logique pour ce circuit (Note: il faut prendre en compte de délai). Simulez votre design à l'aide du logiciel Quartus. Expliquez pourquoi on l'appelle Set-Reset (SR).

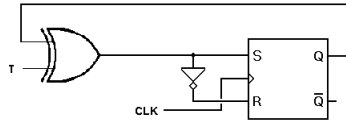


Figure 8: Bascule T

3. On peut réaliser une bascule de type T en modifiant le circuit précédent comme montré dans la figure 8. Écrivez la table logique (réutilisez la table du circuit précédent) et expliquez pourquoi on l'appelle Toggle (T).

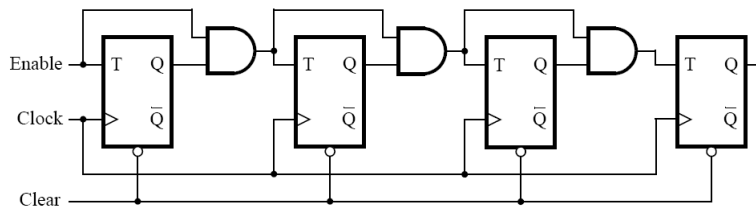


Figure 9: Compteur de 4 bits

4. À l'aide de la bascule précédente, réalisez le circuit de la figure 9 en le modifiant légèrement pour obtenir un compteur à 2 bits. Utilisez les interrupteurs SW0 et SW1 pour les entrées *Enable* et *Clear* et le bouton KEY0 pour l'entrée *Clock*.
5. Affichez les 2 bits de sorties sur un affichage digital en utilisant votre décodeur de 7 segments de l'exercice précédent. Jouez avec les boutons et les interrupteurs désignés pour montrer le bon fonctionnement de votre design.