6GEI228 – Systèmes Digitaux

Laboratoire #2

Décodeur BCD à 7-Segments

Hiver 2018

1. Objectifs

- Développer ses compétences en simplification logique
- Se familiariser avec la plaquette DE10-Lite et son environnement
- Comprendre ce que le titre du laboratoire veut dire

2. Méthodologie

Le but de ce laboratoire est de concevoir un système simple qui prend un nombre décimal de 4 bits en entrée et qui le montre sur un afficheur à 7 segments. La vue à haut niveau de ce système est présentée à la figure 1.

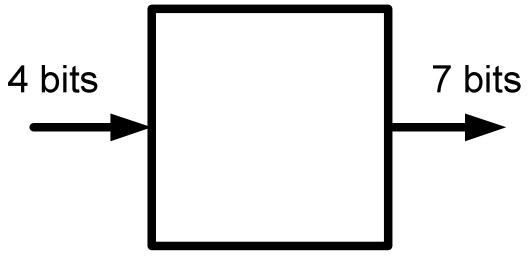

Figure 1. Entrées et sorties de notre système

Avec 4 bits, il est possible d'obtenir des valeurs allant de 0 à 15. Cependant, nous nous intéressons seulement aux valeurs décimales allant de 0 à 9. L'approche qui consiste à utiliser seulement les valeurs 0 à 9 et d'ignorer les valeurs 10 à 15 se nomme de la traduction mot à mot "Décimale codée binaire " ou "Binary-Coded Decimal" (BCD) en anglais.

Pour aborder ce problème, il sera question de décomposer ce système en blocs qui sont plus faciles à gérer. Chaque bloc aura une tâche bien définie et le tout sera assemblé pour obtenir le système final.

Un afficheur à 7 segments est un assemblage de 7 diodes électroluminescentes (DEL, ou en anglais LED pour Light-Emitting Diode) qui sont disposées de façon à afficher un chiffre. Un exemple de ce genre d'affichage est présenté à la Figure 2.

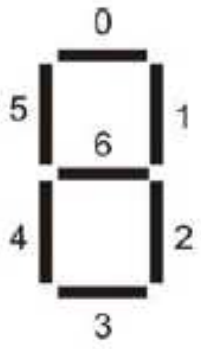

Figure 2. Afficheur a 7 segments

Comme discuté précédemment, l'entrée de 4 bits est un nombre décimal et nous voulons le montrer sur l'afficheur à 7 segments. Si l'entrée était 0111, par exemple, le système devrait afficher le chiffre 7. Pour ce faire, on doit allumer les segments 0, 1 et 2 de la Figure 2. Donc, pour faire un système complet, il faudrait que l'afficheur allume les bonnes DEL pour **chacun des 10 chiffres possibles**.

Pour pouvoir implémenter un circuit, nous devons commencer par décrire ce système sous forme de table de vérité. Pour ce faire, nous allons considérer chaque segment comme étant une fonction différente et nous allons voir quelles valeurs en entrées allument ce segment. Par exemple, pour le segment 6 de la figure 2, il est possible de voir que ce sont les chiffres 2, 3, 4, 5, 6, 8 et 9 qui l'allument. Donc, pour implémenter la fonction, il faudrait commencer par traduire cette dernière situation en table de vérité. Par la suite, il sera question de faire la simplification avec la table de Karnaugh et finalement, implémenter le circuit dans l'environnement Quartus Prime. Il faut faire le même processus avec tous les segments de la Figure 2 et une fois que tous les segments auront été conçus, il sera temps de programmer le circuit sur FPGA et faire les tests finaux.

Revenons à l'exemple avec le segment 6. Nous savons que la lumière sera allumée lorsque les entrées sont 2, 3, 4, 5, 6, 8 ou 9. Avec cette information, il est possible de créer une table de vérité (**attention : elle n'est pas bonne !**) :

	D3	D2	D1	D0	Segment6
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
A	1	0	1	0	X
B	1	0	1	1	X
C	1	1	0	0	X
D	1	1	0	1	X
E	1	1	1	0	X
F	1	1	1	1	X

En regardant la documentation, il est possible de voir que toutes les LED de notre afficheur à 7 segments sont connectées d'une façon particulière : l'anode est connectée à 3.3v tandis que la cathode est connectée aux plots du FPGA par l'intermédiaire d'une résistance (figure 3). Puisque l'anode est connectée à 3.3v et que notre système envoie un signal à la cathode, il faut changer notre stratégie de commande. Pour allumer un segment, il faut envoyer un 0 tandis que, pour l'éteindre, il faut envoyer un 1. Un circuit explicatif un peu plus clair est présenté à la figure 4.

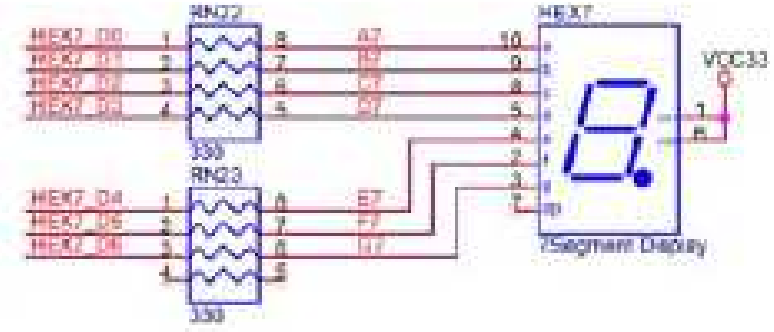

Figure 3. Circuit de l'afficheur a 7 segments

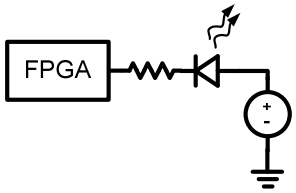

Figure 4. Circuit représentant les connexions de l'afficheur à 7 segments

Pour rectifier la situation, il suffit d'inverser le signal de sortie. La table de vérité devient ceci :

	D3	D2	D1	D0	Segment6
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
A	1	0	1	0	X
B	1	0	1	1	X
C	1	1	0	0	X
D	1	1	0	1	X
E	1	1	1	0	X
F	1	1	1	1	X

À partir de cette table de vérité, il est possible de générer un circuit simple en utilisant la table de Karnaugh et finalement, de l'implémenter sur FPGA.

3. Travail demandé

Pour chaque segment (il y en a 7 au total) :

1. Écrivez l'équation de la fonction qui contrôle chaque segment sous forme de sommation. Par exemple, la réponse pour le 6^{e} segment est :
   SEG6 = Σ (0, 1, 7).
2. Dessinez la table de Karnaugh pour chaque segment en encerclant les termes à simplifier pour clairement les identifier.
3. Écrivez l'équation logique simplifiée pour chaque segment.
4. Faites une simulation dans Quartus et montrez la sortie du **segment 3** pour toutes les valeurs possibles en entrée. La figure 5 montre le résultat attendu pour

la simulation du segment 3. Les premiers 50ns de simulation montrent que la sortie est 0 (segment 3 est allumé) lorsque l'entrée est 0000 (représentant le chiffre 0). Les portions encadrées montrent que, lorsque les entrées sont 1, 4 ou 7, le segment 3 n'est pas allumé. Il est à noter que les chiffres en entrée vont de 0 à 15 (pas dans cet ordre) et que, les sorties qui correspondent aux valeurs 10 à 15 devraient être ignorées.

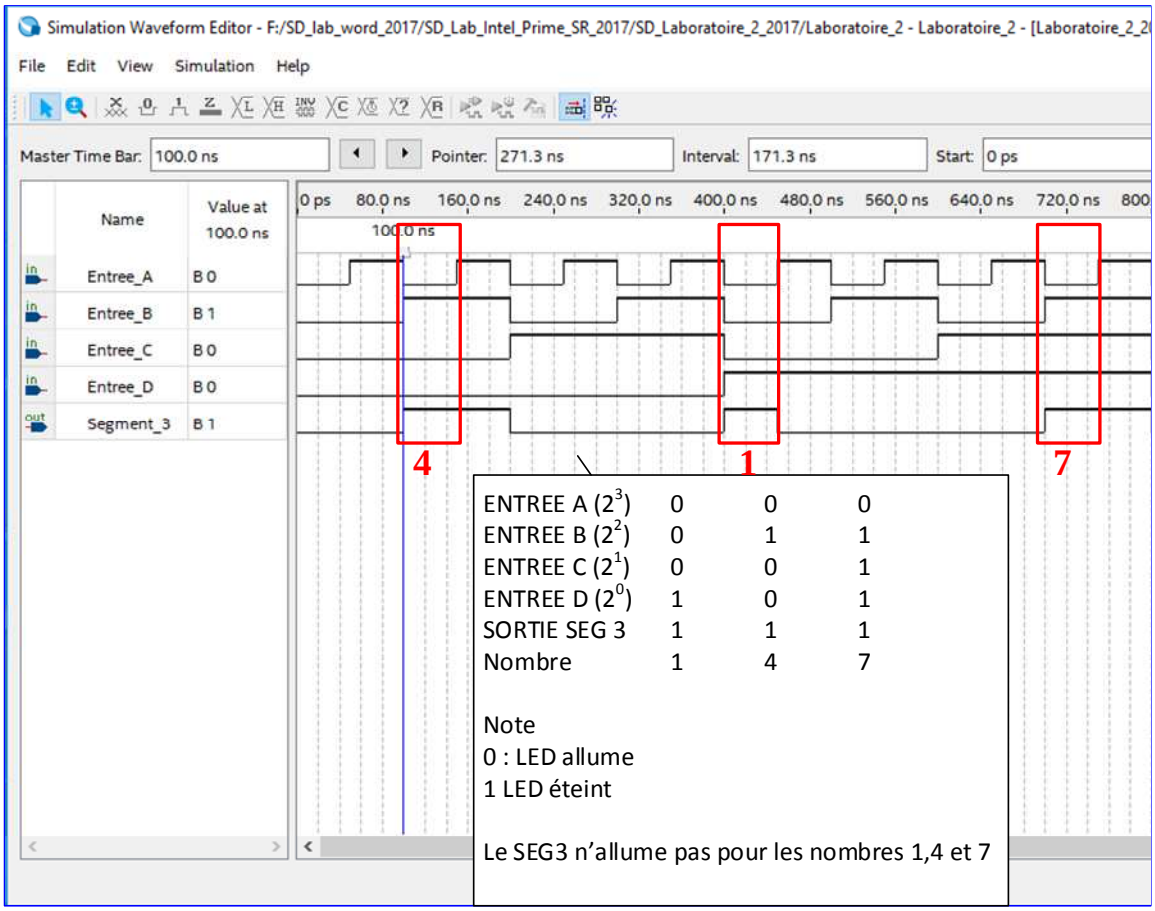

Figure 5 : Simulation pour le segment 3pour toutes les valeurs en entrée.

Implémentez le circuit complet sur FPGA. Assignez les bits d'entrée aux commutateurs en bas de la plaquette et assignez les sorties à un des afficheurs à 7 segments.

5. Montrez le circuit final au chargé de laboratoire.