

INF1500

Logique des systèmes numériques

Laboratoire 4

Soumis par:

Bourai Sami - 2041659

Djellab Souheib-Mounib - 1909732

Le 19 novembre 2019

***Introduction :***

Tout d’abord, ce laboratoire avait pour objectif de créer des composantes VHDL en utilisant exclusivement le code VHDL, pour ensuite les relier dans un circuit logique final. Le circuit complet prend en paramètre un nombre A binaire de 5 bits (c’est-à-dire entre 0 et 31), puis selon le bit de l’entré SEL, A sera multiplié par 2 ou on fera modulo 5 de A. La sortie dans les 2 cas sera en binaire et sera transformé en nombre décimal.

***Circuit complet :***

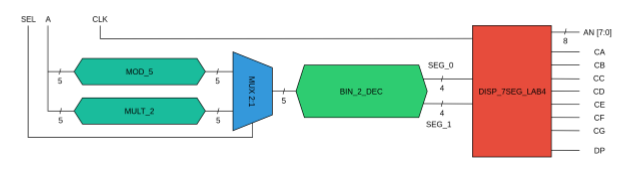


Schéma complet

***Composante MOD\_5 :***

Ce composant prend en paramètre A, un nombre binaire de 5 bits, puis donne en sorti (B) le reste de la division par 5, c’est-à-dire le modulo 5. La sortie B est aussi un nombre binaire de 5 bits. La logique de cette opération est la suivante :

while ( A>= 5)

B=A-5

Il y a aussi une autre méthode dans notre cas qui est de définir tous les nombres entre 0 et 31 dont le modulo 5 est 0, 1, 2, 3 ou 4. Dans ce cas, la logique est la suivante :

Switch A

Case 0, 5, 10, 15, 20,25, 30: B=0 break

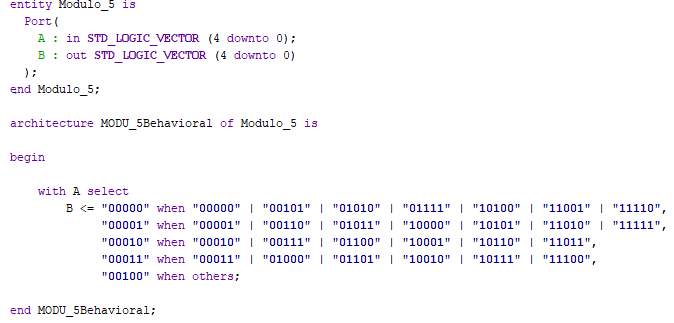
Case 1, 6, 11, 16, 21,26, 31: B=1 break

Case 2, 7, 12, 17, 22, 27, 32: B=2 break

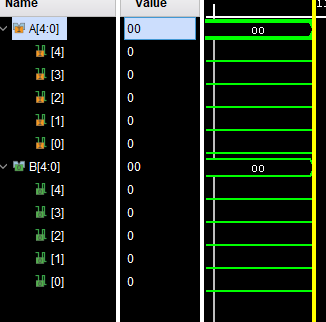
Case 3, 8, 13, 18, 23, 28: B=3 break

Case 4, 9, 14, 19, 24, 29: B=4 break

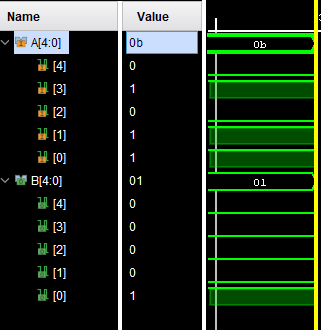
Ainsi voici le code de ce module :



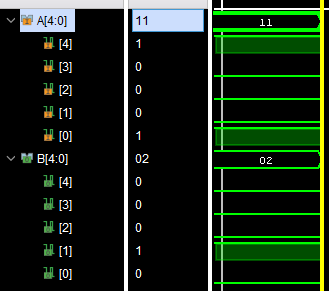
Finalement, il ne reste plus qu’à tester ce dernier :



Ici nous testons la possibilité qui donne pour reste 0. 0%5 donne 0 ce qui la bonne réponse.



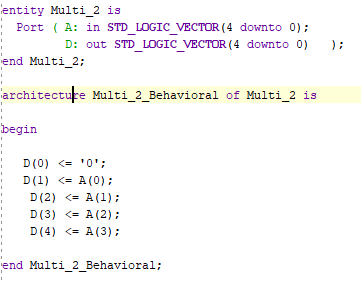
Dans cette autre possibilité, nous testons ce qui donne pour reste 1. Ainsi 11(b en hexadécimal) %5 =1.



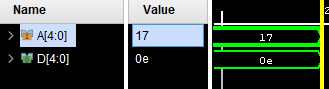
Ici, il faut comprendre ce qui se passe. Sinon, on pourrait le considérer comme une erreur alors que ce n’est pas le cas. On force 11, mais en hexadécimale. Ceci veut donc dire que c’est 16e1+16e0 = 17. Et 17%5 = 2 ce qui la bonne réponse.

***Composante MULTI\_2 :***

Cette composante a pour rôle de multiplier l’entrée A de 5 bits par deux 2 (00010). Et la sortie sera de 5 bits aussi. En faisant la table de vérité de ce module, il est possible, avec une certaine clairvoyance de savoir que cette opération n’est en effet qu’un décalage vers la gauche. Par exemple, en multipliant 2\*4=8, en binaire cela équivaut à 00010\*00100=01000. Donc, le résultat 8, est obtenu en enlevant le bit le plus à gauche de 4 (0) et en ajoutant un bit du côté droit (0). Concernant les entrées de A qui sont supérieurs à 15, c’est-à-dire 16 et plus, la multiplication par 2 donne comme résultats un chiffre à 6 bits. Exemple : 16\*2=32 ce qui est équivalent à 10000\*00010=100000. Or, comme on ne considère que les 5 bits les moins significatifs, on obtient 00000.

Ainsi, voici le code VHDL de ce module :

Suite à cela, le module a été testé séparément telle que dans les captures d’écrans ici-bas :

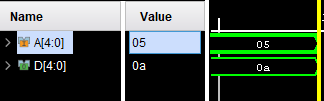


Ici la possibilité qu’un chiffre soit plus grand que 16 a été testée : On a testé 17 (10001) en hexadécimale. Donc (16e0\*7) + (16e1)=23.

23 en binaire est équivalent à 10111.

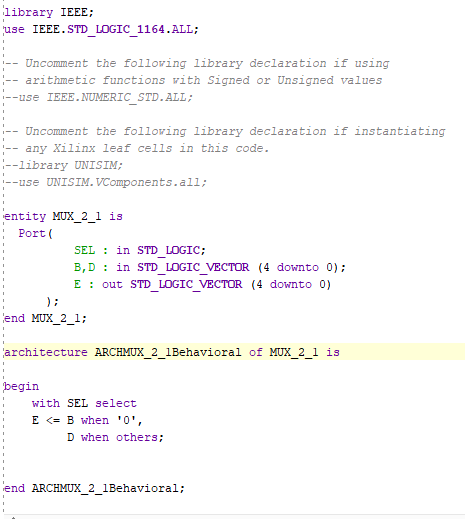
10111\*00010= 01110 ce qui est 14 d’où le fait que c’est écrit (0e) soit la réponse, mais en hexadécimale.

Normalement, 23\*2= 46 (101110). Or, comme on prend les 5 bits les moins significatifs, on obtient 01110 ce qui correspond à 14.

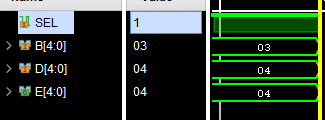


Dans cette deuxième capture, on a testé le fait qu’un nombre inférieur à 16 soit entré. On force 5 (00101) et on obtient 10 (01010) qui est bel et bien la réponse, mais en hexadécimale (0a).

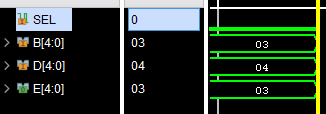
***Composante MUX:2:1 :***

Le multiplexeur est composé de 2 entrées à 5 bits chacune, soit la sortie B venant du MOD\_5 et la sortie D de la MULT\_2. Ce dernier a donc comme tâche de sélectionner la donnée MOD\_5 ou MULT\_2 dépendamment de la valeur forcée en SEL (0 pour choisir MOD\_5 et 1 pour choisir MULT\_2). Voici le code VHDL de ce module :

Par la suite on à réaliser le teste de ce dernier de façon séparé et voici les résultats :



Ici on force 4 en D (pour la multiplication) et 3 en B (pour le modulo 5). En forçant SEL à 1 on sélectionne les données de la multiplication, ce qui fait que nous obtenons 4 dans la sortie E.



Dans l’autre possibilité, la valeur SEL a été changée pour 0, pour ainsi sélectionner les données sortant du MOD\_5 ce qui donne 3 à la sortie E.

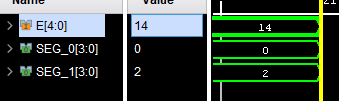
En testant les deux cas, on remarque que le multiplexeur fait bien le travail. Donc, il est valide.

Composantes BIN\_2\_DEC :

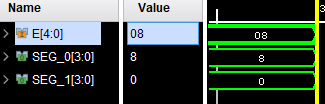
Ce module contient une entrée à 5 bits, soit la sortie du multiplexeur (E) et deux sorties à 4 bits chacune. La sortie SEG\_0 a pour fonction d’afficher le chiffre représentant les unités. Quant à la deuxième sortie SEG\_1, cette dernière a pour fonction d’afficher le chiffre se trouvant aux unités. La réalisation de ce module était la plus longue et la plus complexe de toutes puisqu’il fallait d’abord réaliser deux tables de vérité (une pour la sortie SEG\_0 et une autre pour SEG\_1) et par la suite dégager les cas qui donnaient pour unité 1, 2,3,4,5,6,7,8 ou 9 ainsi que les cas qui donnaient pour dizaines 1,2 ou 3.

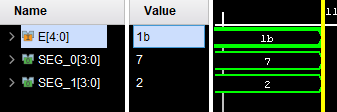
Ainsi voici la résultante du programme écrit : 

Suite à cela, voici les quelques captures d’écrans des tests réalisés :



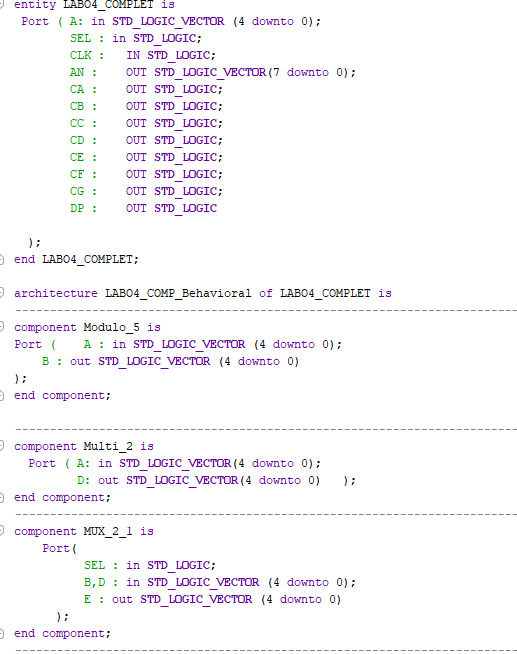
Dans cette première image nous testons 14 en hexadécimale qui est équivalent à 20 en décimale. Ainsi, cela donne 0 pour le SEG\_0, soit pour les unités et 2 pour SEG\_1, les dizaines.

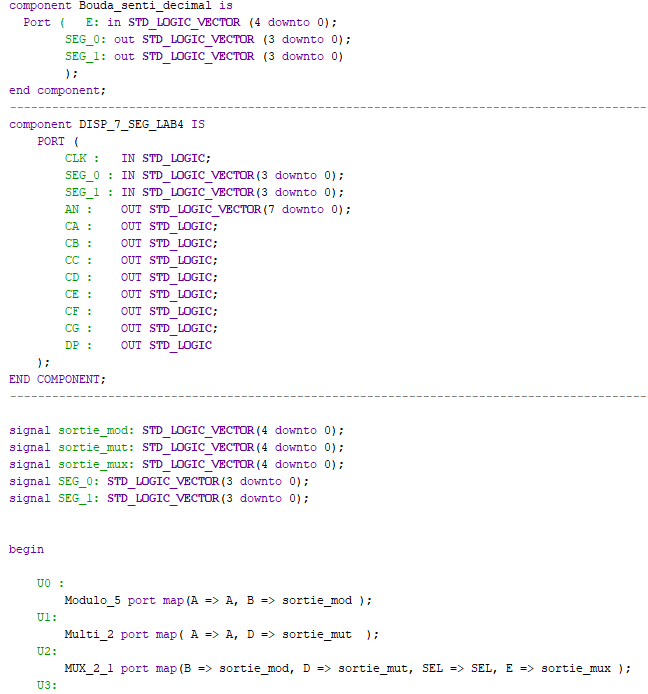


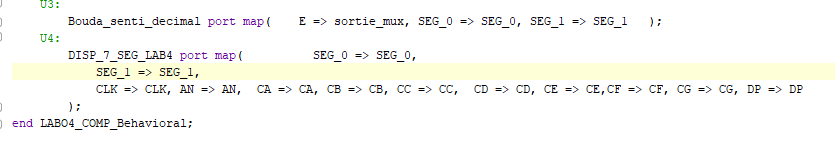
Cette deuxième capture concerne le cas où nous entrons un chiffre inférieur à 10. Donc pour SEG\_0 qui représente les unités la réponse 8 lorsque nous forçons 08. Comme ce chiffre est en quelque sorte « dépourvu » de dizaines alors il affiche 0.

Finalement, on force 1b en hexadécimale qui est 20 en décimale et on obtient 7 en SEG\_0 et 2 en SEG\_1. Donc, nos exemples font le tour et valident ce dernier module.

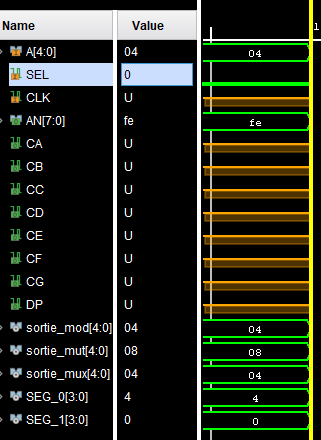
***Montage du circuit final :***

En suivant le guide fourni dans ce laboratoire, nous avons réalisé le montage complet et en voici le code VHDL :





Finalement, il ne reste plus qu’à tester le bon fonctionnement de ce dernier et voici une capture d’écran :



Dans cette image nous forçons 04 en A et nous ajustons SEL à 0. Ainsi, c’est le MOD\_5 qui sera sélectionné par le multiplexeur tel que l’indique l’output ***sorti\_mux***. Comme le chiffre entré en A est inférieur À 10, ce qui sera affiché c’est tout simplement SEG\_0 et SEG\_1.

***Conclusion*** :

Dans ce laboratoire nous avons réalisé un circuit d’une tout autre façon comparativement aux 3 derniers laboratoires. Dans celui-ci, on plutôt programmer le circuit à l’aide de VHDL. D’abord on a codé les différents compartiments (MOD\_5, MUTL\_2, MUX2 :1 et le BIN\_2\_DEC), ensuite on a assemblé le tout afin d’obtenir le circuit final. De plus, dans ce laboratoire il était encore question d’utiliser les notions basiques comme les tables de vérité. Finalement, cette introduction au langage VHDL était un succès puisque nous avons pu obtenir les résultats théoriques souhaités.