TP N°2 : Synthèse de système numériques, VHDL

Ce TP de SY41 consiste en la réalisation d'un programme permettant l'affichage et la somme de deux nombres 8 bits signés. L'utilisation du langage VHDL ainsi que du logiciel Quartus Prime sont nécessaires au bon déroulement de ce TP. Lors de ce TP, plusieurs problématiques ont été rencontrées, comment afficher un nombre de 8 bits sur un affichage limité ? comment garder en mémoire le nombre précédent afin d'additionner le nouveau ? Comment additionne-t-on deux nombres sur 8 bits ? Et surtout, quelles sont les options, les interfaces, à mettre en place afin de pouvoir interagir avec la carte ?

Heureusement, plusieurs solutions ont été envisagées au préalable et peu de questions nécessitaient d'effectuer des recherches :

L'affichage se fait à l'aide de compteur sept segments présents sur la carte, deux d'entre eux sont utilisés pour la partie numérique du nombre, et un troisième est utilisé afin d'afficher le signe « - » lorsque le nombre écrit est négatif, l'implémentation de l'afficheur sept segments se fait dans le fichier **segment.vhd** et permet ainsi de le réutiliser aisément pour chaque digit (8 bits en binaire donne 2 digits en hexadécimal). Le component prend en entrée un vecteur de 4 bits et ressort un vecteur de 8 bits correspondant à chaque segment d'un chiffre :

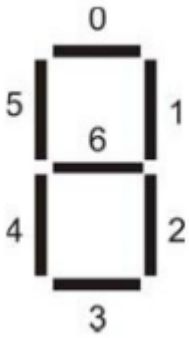

Figure 1 : Chaque segment à un bit de sortie attribué

Figure 2 : Affichage des chiffres hexadécimaux présents sur un afficheur 7 segments

Ensuite, dans la mesure où il est nécessaire de garder en mémoire la valeur précédemment donnée, et d'afficher cette valeur dès que l'horloge actualise sa valeur, pour cela, l'enregistrement de la valeur se fait grâce au component **reg.vdh** et permet de garder en mémoire un vecteur de 8 bits de long. Afin de faciliter l'interaction avec la carte, la valeur s'actualise dès que le bouton poussoir est activé.

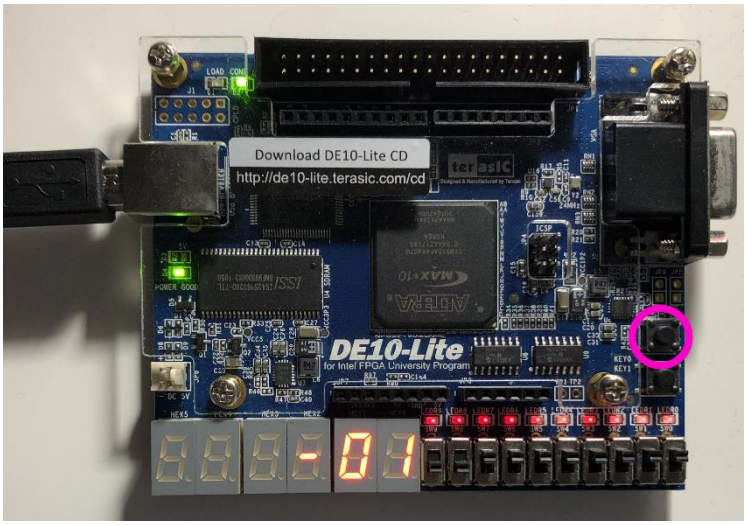

Figure 3 : Le bouton poussoir utilisé entouré en rose

Grâce à la précédente implémentation de la mémoire, on peut désormais intégrer notre additioneur au programme, pour cela on créer un component **somme.vhd** dans lequel on décrit l'additionneur auquel on passera en arguments la valeur enregistrée en mémoire ainsi que la valeur écrite sur les switches on additionne alors les deux valeurs en convertissant le binaire à l'aide des fonctions **to_integer()** et **to_unsigned()** qui permette le temps d'un passage en base 10 la somme des deux nombres, on enregistre la sortie dans la mémoire et on affiche le résultat.

Il faut cependant faire attention aux deux paramètres du component somme qui dicte comment elle doit interagir, si le switch « enter » est actif, alors la valeur présente sur les switches est mémorisée et affichée, sinon, si le switch « add » est actif, alors la valeur mémorisée et la valeur des switches s'additionnent et est mémorisée puis affiché.

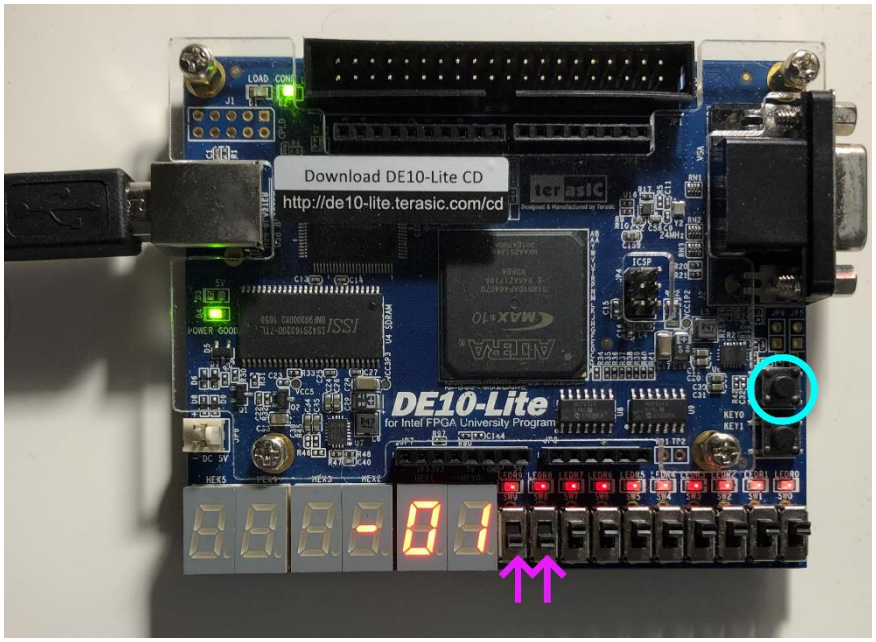

Figure 4 : En bleu, le bouton qui lance l'addition, en rose, respectivement « enter » et « add »

Pour finir, il est important de se délaisser de l'horloge manuelle et d'utiliser celle présente sur la carte afin d'éviter de devoir générer les mémorisations à la main qui amène plus de problèmes qu'elle n'en résout. Dans ce dernier component, **pulse_input.vhd**, on utilise un signal sur deux bits pour mémoriser l'état du système et synchroniser notre bouton poussoir sur l'horloge de la carte (on s'assure que la durée de pression du bouton soit constante).

Pour conclure, l'implémentation actuelle satisfait les demandes du TP mais reste assez restreinte au niveau de l'utilisateur final qui ne peut pas effacer la valeur précédente ou la modifier de façon directe, il est obligé de « réécrire » par-dessus. En revanche, l'utilisation de l'horloge de la carte permet d'augmenter sensiblement la fiabilité et la rapidité d'exécution du programme.