Electronique numérique
Initiation à VHDL (3/3)
Synthèse sur FPGA

Ce BE d'Electronique vise à concevoir un circuit concret et le synthétiser sur FPGA.

Remarque 1 : Il n'est pas possible de vous mettre à disposition un tel FPGA par personne, ni même par binôme (plusieurs groupes en parallèle). Vous pourrez accéder aux cartes à tour de rôle. Ces objets sont fragiles et coûteux. **Merci d'en prendre le plus grand soin** : notamment le cable USB doit être manipulé avec précaution.

Remarque 2 : Lancer la commande suivante afin de bénéficier des logiciels d'Electronique.

```
$ module load electronique
```

1 Synthèse sur FPGA : serrure numérique

Spécifications Nous cherchons à réaliser le système numérique suivant, dont les spécifications peuvent se résumer ainsi : une porte est sécurisée par un code d'accès. Si le code tapé est correct, une diode clignote pendant 5 secondes, la porte s'ouvre et se referme en 5 secondes, et l'accès est alors à nouveau protégé.

Voici quelques éléments techniques :

Carte et FPGA Le clavier est connecté à une carte Nexys4DDR de la société Digilent (fig 1).

Cette carte embarque une FPGA Artix7 de la société Xilinx [1]. Les ports d'entrées/sorties (ou "pattes") du FPGA ont été soudés par Digilent : ils sont connectés 1 par 1 vers des connecteurs d'extensions comme les PMOD que nous allons utiliser, mais également des composants discrets présents

1. Inventeur des FPGA, et leader mondial avec Intel

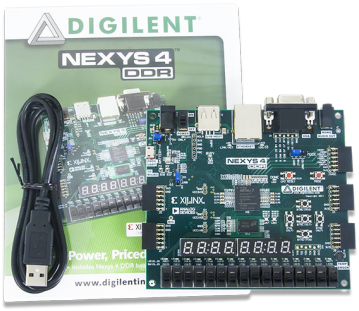

FIGURE 1 – Carte Digilent Nexys4DDR avec Xilinx Artix-7

sur cette même carte : boutons, LEDs, interrupteurs, VGA, micro, mémoire externe, accéléromètre, micro, capteur de température, carte SD, ethernet ! Il n'est évidemment pas possible de modifier (ou "reconfigurer") ces connexions. Votre code VHDL sera synthétisé au sein du FPGA, à l'inverse intégralement **reconfigurable**.

Synthèse et chargement sur FPGA Nous procéderons par *script de synthèse*, que l'on peut lancer directement en ligne de commande [2]. Ce script permet donc de passer du VHDL à un circuit (netlist) et génère au final un fichier appelé *bitstream*, qui permet de configurer le FPGA avec vos portes logiques. Il faut par ailleurs indiquer au préalable une *correspondance* (ou "mapping") entre votre entité VHDL (vos entrées/sorties nommées) et les pattes physiques du FPGA. Cette mise en correspondance se fait dans un *fichier de contraintes* dont l'extension est *.xdc*. Enfin, le bitstream sera chargé sur FPGA à l'aide d'une autre commande, donnée par Digilent.

Afin de simplifier la manipulation, ces différents éléments sont donnés sous Moodle.

- Entité VHDL du circuit à réaliser, ainsi qu'une architecture partiellement concue (vous devrez y instancier quelques composants, discutés par la suite).
- Script de synthèse (script.tcl).
- Fichier de contraintes : Nexys4DDR.xdc

Le fait de vous donner cette entité vous affranchi d'écrire vous-mêmes le fichier xdc, ainsi que le script de synthèse.

Commandes Linux La procédure de mise au point se fera de la manière suivante :

- Analyse syntaxique avec GHDL :

2. Pour information, ce script est écrit dans le langage TCL, langage de script très utilisé pour commander des logiciels, y compris en dehors de l'Electronique.

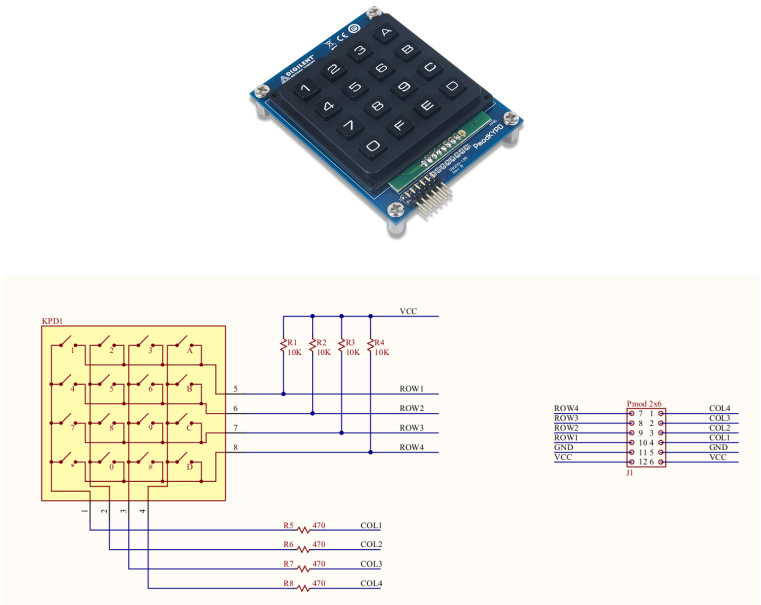

FIGURE 2 – Clavier 16 touches

```
$ ghdl -a nom.vhd
```

— Synthèse FPGA avec la commande :

```
$ vivado -mode tcl -source script.tcl
```

— Chargement sur FPGA du bitstream "top.bit" généré par la commande précédente :

```
$ djtgcfg prog -d Nexys4DDR -i 0 -f SYNTH_OUTPUTS/top.bit
```

2 Principe de fonctionnement du clavier

Le clavier 16 boutons est un matériel très frustre, comme on peut l'observer sur la figure 2 : outre les boutons, seules des resistances de pullup sont nécessaires à sa constitution [3].

Le principe de fonctionnement est le suivant : sans appui sur un bouton, les résistances de pull-up conduisent les sorties "row" à 3.3 v (le 1 logique sur ce FPGA). Lorsqu'on appuie sur un bouton, les tensions des sorties row dépendent des tensions appliquées sur les entrées "column". Pour détecter le bouton sur lequel on a appuyé, il suffit alors de piloter savamment les entrées "column". On procèdera comme suit :

3. Tout nous porte à penser que les touches sont aidées d'un mécanisme d'*anti-rebond* `http://hades.mech.northwestern.edu/index.php/Switch_Debouncing`.

— On envoit une valeur 0 sur chacune des colonnes, successivement, les autres restant à 1. Le rythme de ces excitations, retenu ici, est de 1kHz (1 colonne "testée" par milliseconde).
— On échantillonne la réponse des 4 signaux row : si l'un des row présente un signal à 0, c'est sur ce row qu'il y a eu un appui. On réalise cet échantillonnage juste avant de modifier à nouveau la colonne, de manière à laisser un maximum de temps à la resistance de pullup (et des capacités parasites) de se stabiliser à son éventuelle nouvelle valeur.
— On connait alors la coordonnées de la touche, en row et col.

Le clavier 16 boutons est connecté au port PMOD JA de la carte, à l'aide de 12 broches. Outre l'alimentation et la masse, on trouve 4 broches "row" et 4 broches "col".

Travail à réaliser On doit se doter de plusieurs circuits VHDL :

— Un *timer* qui déclenche un événement toutes les millisecondes. Ce timer sera basé ici sur un simple compteur, qui compte de 0 à $2^n - 1$. Sachant que la carte délivre une horloge cadencée à 100 Mhz, quelle est la bonne valeur de n ?
— Au passage, on peut créer un second timer, qui délivre un événement toute les secondes. On ajoutera une entrée "start", qui permet de déclencher ce timer.
— Créer l'automate qui pilote les signaux d'excitation des colonnes du clavier, et échantillone les réponses des rangées du clavier.
— Instancier les composants précédents dans cette même architecture. Faire les connexions nécessaires : création des signaux, port map etc.
— Compléter le code VHDL "top.vhd" donné sous Moodle de manière à déterminer la touche appuyée.
— Compléter le code VHDL de manière à afficher le numéro de touche sur un afficheur 7 segments. Un circuit déjà instancié dans le code gère cet affichage.

3 Automate de vérification du code d'accès

On cherche maintenant à compléter notre circuit avec un automate qui contrôle un code à 4 digits (4 touches). Le code secret est ici : 1971 [4]. Si le code est correct, on peut compléter avec l'énoncé : clignotement pendant 5 secondes, puis déclenchement du mécanisme d'ouverture-fermeture. Cette ouverture-fermeture est déclenchée par le signal "door_open_close" à 1.

4. Année de commercialisation du premier microprocesseur par Intel : le 4004.

4 Synthèse

Appliquez les commandes Linux comme indiqué précédemment. **Faites preuves de bon sens** à la lecture des erreurs probables rapportées par les différents outils. Lorsque vous obtenez un bitstream correct, appelez votre enseignant, de manière à le charger sur FPGA et vérifier si votre mécanisme fonctionne.