

ELE8307 - Laboratoire 1

Introduction à Quartus II et conception d'un contrôleur clavier PS/2

Introduction

Le logiciel Quartus II est un outil de conception de circuit pour FPGA d'Altera. Ce logiciel permet essentiellement d'effectuer la synthèse, la simulation, et la programmation de circuits sur FPGA. La carte de développement disponible au laboratoire est la DE2, équipée d'un FPGA Cyclone II.

Objectifs

Dans ce laboratoire, il sera question de s'initier à la conception de circuits numériques sur FPGA avec l'outil Quartus II d'Altera. Ce laboratoire vous permettra de développer les compétences suivantes :

- ✓ Synthèse et implémentation de circuits numériques avec Quartus II
- ✓ Simulation de circuits numériques avec ModelSim
- ✓ Utilisation de l'outil de déverminage SignalTap II
- ✓ Utilisation de module IP pour la conception de circuit numériques

Pour y parvenir, vous serez amené à implémenter un contrôleur clavier PS/2 qui pourra être utilisé dans les prochains laboratoires de la session.

Documentation

Toute l'information nécessaire pour les manipulations dans Quartus II est présente dans l'énoncé.

Vous trouverez toutefois sur Moodle des annexes qui pourraient vous intéresser :

- Un tutoriel sur l'utilisation de SignalTap II
- Guide d'utilisation de ModelSim avec Quartus II
- Modèle d'instanciation des modules LPM en VHDL
- Un lien vers la liste complète des codes des touches du clavier PS/2

Notez que les annexes de chacun des laboratoires resteront disponibles sur Moodle tout au long de la session.

Travail à réaliser

Le laboratoire se divise en 3 parties. Dans la première partie vous apprendrez à utiliser Quartus II en implémentant une première version d'un contrôleur clavier. Le contrôleur est défectueux et vous serez amené à observer les problèmes par simulation. Dans la deuxième partie, vous travaillerez avec une autre version du contrôleur dont vous allez observer le comportement à basse fréquence avec l'outil Signal Tap II. Ce contrôleur amélioré contient toujours une petite erreur et vous devrez la corriger. Dans la troisième et dernière partie, vous allez concevoir, en utilisant des modules IP, un circuit permettant d'afficher en base décimale sur les afficheurs 7-segments les octets reçus du clavier.

Commencez par récupérer les deux fichiers suivants (disponibles sur Moodle) :

- ✓ DE2 pin assignments (DE2_pin_assignments.csv)
- ✓ Sources VHDL (sources_vhdl_lab1.zip)

1. Implémentation de la première version du contrôleur clavier

Dans cette partie vous apprendrez à utiliser Quartus II afin de synthétiser un design et de le programmer dans la carte DE2, ainsi qu'à simuler un design avec ModelSim.

A. Création et préparation d'un projet Quartus II

Ouvrez le logiciel Quartus II en cliquant sur :



Tous les programmes → Altera → Quartus II 13.0.1 → Quartus II 13.0sp1 (64-bit)

Il faut d'abord créer un projet. Si une fenêtre de bienvenue s'affiche, cliquez sur « Create a New Project ». Sinon, une fois le logiciel ouvert, allez dans :

File → New project wizard...

Une page d'introduction s'affiche, cliquez sur **Next**, puis sur la page suivante :

- Indiquez l'emplacement voulu du projet ainsi que le nom du projet. Notez que lorsque vous voulez parcourir les dossiers de l'ordinateur, des erreurs « Restrictions » pourraient apparaître. Elles sont dues à l'interdiction d'aller dans le disque C; validez et continuez. Lisez le document d'informations sur Moodle pour choisir votre disque de travail ! conseil : créez un dossier « lab1 » dans le disque D et sélectionnez-le.
- Ignorez le champ concernant « top-level » (il devrait se remplir par lui-même)

Sur la page suivante vous proposant d'ajouter des fichiers au projet, n'ajoutez rien pour l'instant.

La page suivante vous permet de spécifier le FPGA qui sera utilisé pour le projet. Au laboratoire nous utilisons le **Cyclone II EP2C35F672C6**. Choisissez-le, puis passez à la page suivante.

Sur la page EDA Tools, il est possible de spécifier l'utilisation d'outils tiers pour la synthèse, la simulation, et l'analyse des contraintes temporelles. Sélectionnez pour la **simulation** l'outil **ModelSim-Altera** et le format **VHDL** (ne pas cocher la case qui s'affiche dans la dernière colonne), puis avancez à la dernière page.

La dernière page affiche un résumé du projet qui sera produit, appuyez sur **Finish**.

Vous arrivez alors dans l'interface principale de Quartus. Vous trouverez une barre d'outils pour lancer différentes étapes du flow de conception. Placez votre curseur sur un bouton pour que sa fonction apparaisse. L'environnement est composé des principales régions suivantes :

- « Project Navigator » permet de voir la hiérarchie (onglet « Hierarchy ») de votre design et les fichiers inclus dans votre projet (onglet « Files »).
- « Tasks » permet d'observer l'évolution de l'exécution des différentes tâches du flow de conception.
- « Messages » permet de lire les messages produits lors de l'exécution des tâches.

Maintenant, afin de compléter la création du projet, rendez-vous dans le menu

Assignments → Device... → Device and Pin options...

Dans la fenêtre qui s'ouvre, cliquez sur la catégorie **unused pins** et assurez-vous de sélectionner l'option **As input tri-stated**. Validez sur les deux fenêtres pour retourner sur la page principale.

Il ne reste maintenant qu'à importer l'assignation des broches qui fait correspondre les broches réelles du FPGA (ex: PIN_D26) avec des noms plus évocateurs (ex: PS2_CLK). Pour y parvenir allez dans :

Assignments → Import assignment...

Sélectionnez alors le fichier **DE2_pin_assignments.csv** disponible sur Moodle et validez. Notez qu'il est alors possible d'aller observer les noms de broches en allant dans Assignments → Pin Planner.

B. Intégration du contrôleur clavier dans le projet Quartus II

Avant de démarrer, copiez les fichiers suivants (fournis dans « fichiers_vhd.zip ») dans votre répertoire de projet :

- ✓ ctrl_clavier_tres_mauvais.vhd
- ✓ ctrl_clavier_mauvais.vhd
- ✓ lpm_fifo0.vhd
- ✓ ctrl_clavier_tb.vhd

Nous allons maintenant ajouter au projet les fichiers VHDL du contrôleur clavier en ouvrant la fenêtre :

Project → Add/Remove Files in Project...


Sélectionnez les fichiers **ctrl_clavier_tres_mauvais.vhd** et **lpm_fifo0.vhd** de votre dossier projet. N'oubliez pas de cliquer sur **Add** après les avoir sélectionnés s'ils ne sont pas ajoutés automatiquement à la liste ! Cliquez ensuite sur **OK**, vos fichiers apparaissent dans l'onglet « Files » du *Project Navigator*.

Nous allons ensuite débiter par l'ajout d'un fichier schématique au projet qui servira d'entité de haut-niveau (*top-level*). Pour y arriver, faites :

File → New... → Design Files → Block Diagram/Schematic File

Afin d'ajouter notre contrôleur clavier dans le diagramme, nous allons produire un symbole schématique à partir du fichier VHDL. Dans le *Project Navigator*, faites un clic-droit sur le fichier *ctrl_clavier_tres_mauvais.vhd* et cliquez sur **Create Symbol Files for Current File**.

Fermez alors l'onglet « Compilation Report » qui s'est affiché pour retourner sur le schéma.

La barre d'outil à gauche (ou en haut) permet d'éditer un fichier schématique, cliquez sur le bouton *Symbol Tool* (), vous apercevrez deux dossiers, un pour les composants du projet, l'autre pour la librairie d'Altera. Ouvrez le dossier du projet et ajoutez le symbole du contrôleur clavier créé plus haut au schéma (la touche ESC s'avère utile pour ne faire l'instanciation qu'une seule fois...).

Sauvegardez alors ce fichier dans votre répertoire projet, et nommez-le **partie1_top.bdf** (la sauvegarde n'était pas autorisée sur le fichier vide).

Pour indiquer quel fichier se situe au sommet de la hiérarchie de design, allez dans le *Project Navigator*, et puis dans l'onglet « Files », faites un clic droit sur le fichier *partie1_top.bdf*, et puis cliquez simplement sur **Set as Top-Level Entity**. Il apparaît alors dans l'onglet « Hierarchy ».

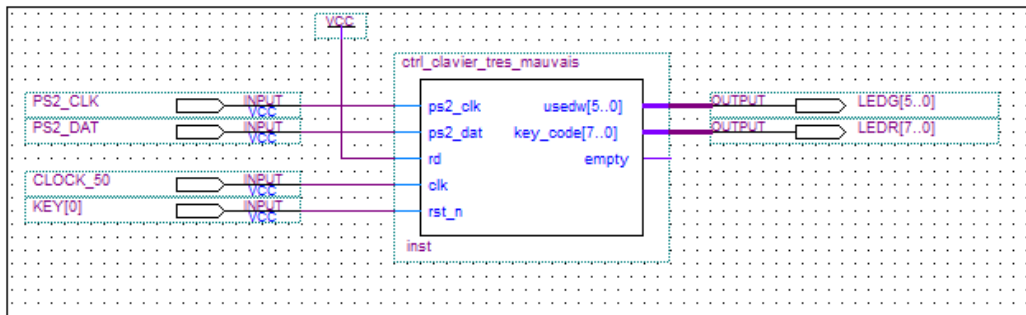
Nous allons maintenant connecter le contrôleur clavier aux broches du FPGA. En cliquant sur *Symbol Tool*, allez dans la *librairie d'Altera* → *primitives* → *pin*, et puis ajoutez des entrées (input) et des sorties (output), sauf pour le port *rd* (*read*) qui sera connecté à VCC (*primitives* → *others*) et le port *empty* qui sera flottant. Maintenant, nous allons nommer les broches (en double-cliquant dessus) et les relier aux ports comme suit :

Broche du FPGA	Port du composant clavier PS/2
PS2_CLK	ps2_clk
PS2_DAT	ps2_dat
CLOCK_50	clk
KEY[0]	rst_n
LEDG[5..0]	usedw[5..0]
LEDR[7..0]	key_code[7..0]

Notez que la présence des crochets [] indique qu'il s'agit d'un bus.


Sauvegardez le fichier.

Le schéma devrait ressembler à celui de la figure suivante :




C. Compilation et Programmation du contrôleur clavier dans le FPGA

La compilation dans Quartus réfère au processus de synthèse, placement, routage, et génération du fichier de programmation de votre circuit pour le FPGA. Pour la lancer :

Bouton «  » ou : Processing → Start Compilation.

Une fois la compilation terminée, un rapport de compilation devient disponible. On y trouve notamment l'information relative aux ressources utilisées dans le design.

La programmation du FPGA est faite en y envoyant le fichier de programmation produit à la fin de la compilation. Comme pour la compilation :

Bouton «  » ou : Tools → Programmer

Dans la fenêtre de programmation, assurez-vous que :

- ✓ le champ à côté du bouton « Hardware Setup » indique « USB-Blaster »
(Sinon, allumez la carte, cliquez sur **Hardware Setup** et sélectionnez **USB Blaster**)
- ✓ le mode est **JTAG**
- ✓ seule la case **Program/Configure** est cochée.

Cliquez sur **Start**.

Le circuit est maintenant en opération dans le FPGA, appuyez sur les touches du clavier que vous avez branché à la DE2 et observez les codes clavier PS/2 sur les LEDs rouges (voir Annexe 1 et Annexe 2 pour valider le fonctionnement).

D. Simulation fonctionnelle en utilisant ModelSim

Lorsque l'on utilise ModelSim via Quartus pour la première fois, il faut indiquer à Quartus l'emplacement de ModelSim. Pour ce faire allez dans **Tools → Options**, puis dans la catégorie **General**, cliquez sur **EDA Tools Options**. Dans la liste des outils, indiquez l'emplacement de l'exécutable de **ModelSim-Altera** en recopiant le chemin suivant : « C:\Altera\13.0sp1\modelsim_ase\win32aloem\ » puis faites **OK**.

Afin de réaliser une simulation fonctionnelle, il faut d'abord avoir un banc d'essai (testbench). Vous utiliserez celui qui est fourni (**ctrl_clavier_tb.vhd**). Ajoutez-le à votre projet. Avant d'aller plus loin, prenez un moment pour analyser le fichier source VHDL du contrôleur clavier PS/2 ainsi que du fichier du banc d'essai. Celui-ci génère une horloge « **CLOCK_50** » avec une période de 20ns, et une horloge « **PS2_CLK** » de période 66.6ns (bien qu'en réalité cette dernière soit bien plus grande). On y voit également un envoi caractérisant la pression d'une touche. Le bit transmis varie à intervalles fixes correspondant à l'horloge PS2.

Vous devez maintenant configurer l'utilisation de ce banc d'essai. Ouvrez la fenêtre de paramètres :

Assignments → Settings... → EDA Tool Settings

Assurez-vous d'avoir le bon outil de simulation. Dans la sous-section **Simulation**, cochez l'option **Compile test bench** et cliquez sur **TestBenches...** Vous allez maintenant rajouter votre banc d'essai. Faites **New** et nommez-le de la même façon que votre entité (**ctrl_clavier_tb**). Cochez l'option **Use test bench to perform VHDL simulation** et dans le champ maintenant disponible, inscrivez « U0_ctrl_clavier_tres_mauvais » représentant l'instance à simuler. Faites terminer la simulation après 2us et ajoutez finalement votre fichier de banc d'essai. Appuyez sur **OK** jusqu'au retour à la fenêtre principale du logiciel. Mettez le fichier à tester (ctrl_clavier_tres_mauvais.vhd) en **top-level** pour que le banc d'essai lui envoie directement ses stimuli. Compilez maintenant votre projet. Une fois la compilation complétée, Lancez la simulation :

Tools → Run EDA Simulation Tool → EDA RTL Simulation

Vous pouvez maintenant voir votre simulation (réglez le zoom pour mieux observer l'ensemble de la simulation). Faites quelques changements dans votre banc d'essai et assurez-vous d'en observer les résultats en simulation.

Note : Il arrive parfois que l'on désire voir les signaux internes du circuit testé. Cliquez sur le petit « + » de l'instance « ctrl_clavier_tb » dans la fenêtre « sim », puis cliquez sur l'instance de votre contrôleur (normalement sous le nom « u0_ctrl_clavier_tb »). Vous verrez alors apparaître, dans la fenêtre « Objects », les noms de tous les signaux de cette instance. Vous pouvez rajouter un signal en faisant un **clic droit → To Wave → Selected items**. Vous devez par la suite relancer la simulation à partir de Modelsim. Pour ce faire, cliquez sur l'icône **Restart** située à la gauche du temps de simulation (que vous devez mettre à 2us), dans la barre d'outils supérieure de Modelsim. Cliquez ensuite sur l'icône **Run** (à la droite du temps de simulation) pour relancer la simulation et obtenir le résultat voulu.

E. Simulation avec délais en utilisant ModelSim

Il est possible de faire une simulation avec délais en utilisant les mêmes outils. Vous devez d'abord produire la netlist :

Processing → Start → Start EDA Netlist Writer

Cette étape n'est pas toujours nécessaire puisqu'elle est effectuée lors de la compilation du projet. Vous devez vous assurer qu'à tout changement de votre VHDL, vous recompilez avant une nouvelle simulation. Vous pouvez maintenant lancer la simulation avec délais :

Tools → Run EDA Simulation Tool → EDA Gate Level Simulation

Utilisez le « Slow Model » pour augmenter les chances d'erreur. Observez les changements en simulation, et répondez aux questions suivantes dans votre rapport. Vous pouvez changer le banc d'essai pour mieux voir le problème en rajoutant des envois et en rallongeant la simulation. Le problème est déjà visible avec le fichier fourni, mais il serait intéressant pour vous d'essayer de le reproduire.



- Quel problème observez-vous avec le registre *fifo_req_in* ?
- Pouvez-vous l'expliquer ?
- Quel est le lien avec le registre *ps2_req_decal* ?

Note : il n'est pas nécessaire de corriger ce contrôleur !

Note : n'hésitez pas à passer à la suite et à revenir plus tard à ces questions. La partie suivante est indépendante.

2. Deuxième version du contrôleur PS/2

Dans cette partie nous allons utiliser l'outil SignalTap II, qui est en quelque sorte un analyseur logique interne, pour observer le fonctionnement du contrôleur clavier PS/2 lors de la réception complète d'un octet.

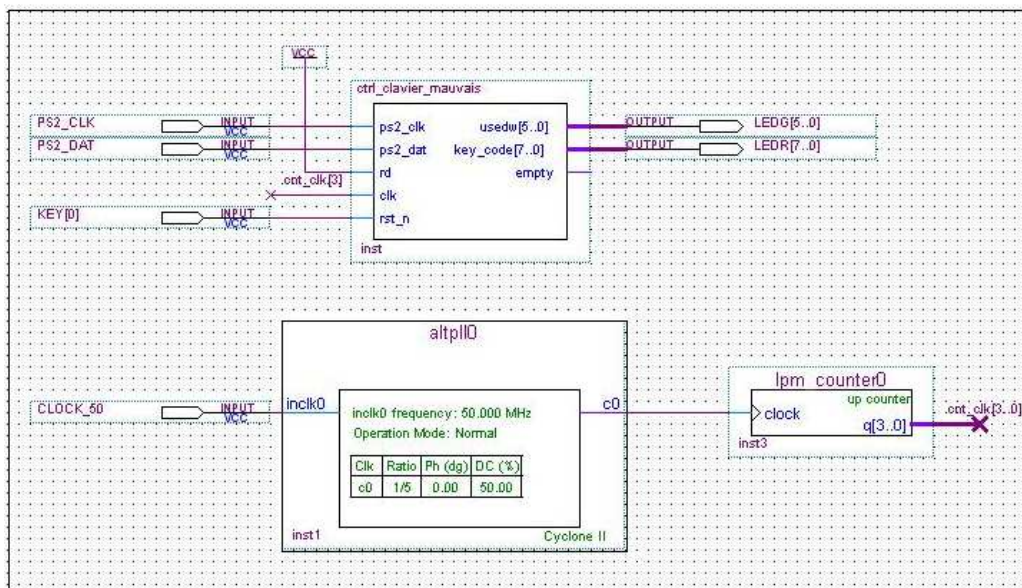
A. Intégration de la 2^e version du contrôleur clavier PS/2

Créez un nouveau fichier schématique **partie2_top.bdf** et intégrez-y cette fois le contrôleur clavier PS/2 provenant du fichier **ctrl_clavier_mauvais.vhd**. Utilisez les mêmes branchements que pour le contrôleur précédent, sauf pour l'horloge CLOCK_50 à 50MHz. Étant donné que SignalTap II doit conserver en mémoire les signaux échantillonnés, et que la différence entre les fréquences d'horloge système et PS/2 est trop importante (50MHz vs ~10kHz) nous allons utiliser une horloge système moins rapide.

Ajoutez une PLL (Phase Locked Loop) au design, en cherchant le module **altpll** dans la librairie d'Altera (**megafunctions/IO/altpll**). Gardez le format VHDL et cliquez Next. La PLL devra produire une sortie de 10MHz, enlevez toutes les entrées et sorties proposées par le Wizard, de sorte à n'avoir qu'une entrée et une sortie dans la PLL produite.

Pour ralentir la fréquence d'horloge encore plus, ajoutez un compteur binaire en cherchant le module **lpm_counter**, configurez-le sur 4bits et enlevez tout le superflu de sorte à n'avoir qu'une entrée d'horloge et une sortie sur 4bits dans le compteur produit. Nous allons utiliser le bit de poids fort comme horloge système, et l'horloge a 10MHz pour échantillonner les signaux d'intérêt.

Pour relier le bit de poids fort du bus sortant du compteur à l'horloge du contrôleur PS/2, connectez un bout de bus à la sortie du compteur, et nommez ce bus **cnt_clk[3..0]** (clic droit sur le bus, puis **properties**), ensuite connectez un bout de fil à l'entrée clk du contrôleur PS/2 et nommez ce fil **cnt_clk[3]**. La connexion implicite créée sera détectée par Quartus II. À ce stade, vous devriez avoir un schéma ressemblant à la figure suivante :



Compilez votre design en prenant soins de mettre **partie2_top.bdf** en tant que top-level entity.

B. Utilisation de Signal Tap II

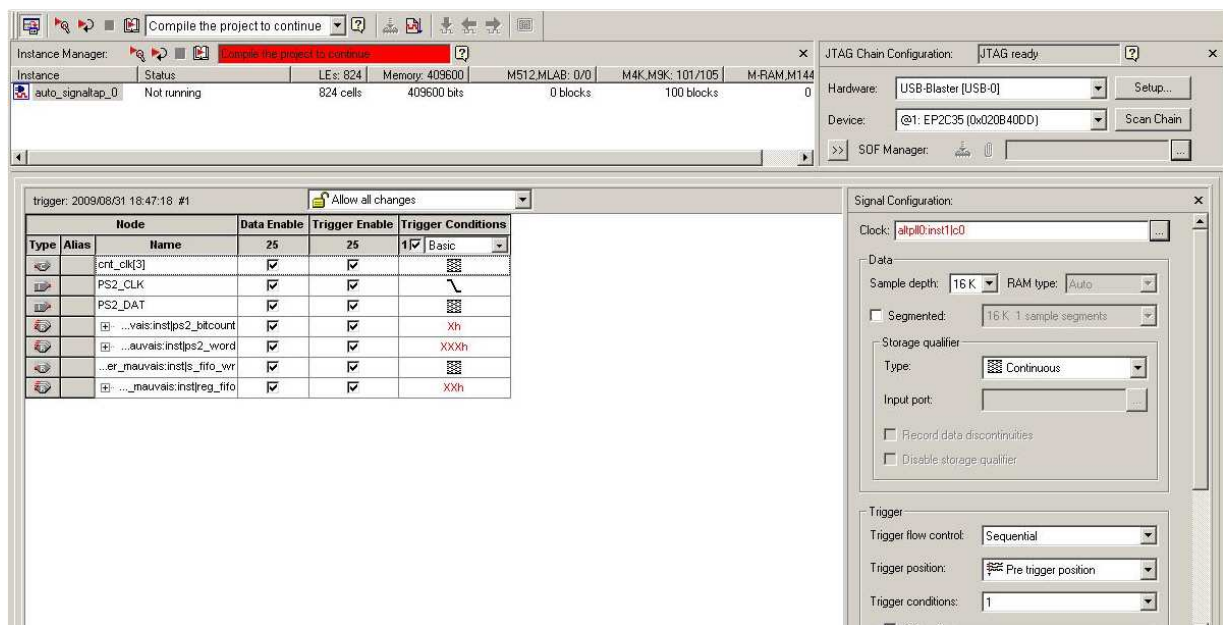
File → New... → Verification/Debugging Files → Signal Tap II Logic Analyzer File

Enregistrez le fichier sous le nom **partie2_stp1.stp**, une fenêtre apparaîtra vous demandant « Do you want to enable Signal Tap II File 'partie2_stp1.stp' for the current project? », répondez oui. Il est possible d'accéder à cette option dans Assignments → Settings... dans la catégorie Signal Tap II Logic Analyzer.

L'onglet Setup de la fenêtre Signal Tap II permet de sélectionner les signaux d'intérêt, l'horloge d'échantillonnage, les conditions de déclenchement de l'échantillonnage, la quantité de mémoire, et autres.

Ajoutez les signaux d'intérêt suivants : PS2_CLK, PS2_DAT, cnt_clk[3], ps2_word, ps2_bitcount, s_fifo_wr et reg_fifo. Vous pouvez choisir le filtre Signal Tap II (Pre-synthesis) et faire List pour avoir la liste des signaux avec les noms originaux. Afin de visualiser l'envoi PS/2, mettez une condition de type **falling edge** sur l'horloge PS/2. Également, vous devrez éliminer les bits 0 et 1 de ps2_word qui ne servent pas (clique droit → delete).

Dans la région Signal Configuration, sélectionnez la sortie **c0** de la PLL comme horloge d'échantillonnage (une fois dans le Node Finder appuyez sur le bouton '...' à droite du champ Look in et sélectionnez altpll pour accélérer la recherche), et ajustez le champ Sample Depth à **16K**. Dans la région Trigger, le champ position permet de décider si on veut observer ce qui se passe avant, après ou autour du moment où la condition de détente est vraie, choisissez **Pre trigger position**. À ce stade, vous devriez avoir une fenêtre comme sur la figure suivante :



Dans la région JTAG Chain Configuration, ajustez le champ Hardware à **USB-Blaster[USB-0]** et assurez-vous que le champ Device soit rempli. Compilez le projet Quartus II. **Une fois la carte programmée par la fenêtre Signal Tap II** (dans le coin haut à droite en utilisant le fichier .sof, comme sur la figure précédente), la mention Ready to Acquire devrait être visible, cliquez sur le bouton **Run Analysis** pour lancer l'acquisition.

Il devrait vous être possible d'observer un premier problème avec ce contrôleur clavier. Il semble en effet qu'un problème existe : on observe que les codes PS/2 sont systématiquement écrits plusieurs fois dans la FIFO. Corrigez ce petit problème ayant échappé à l'attention du concepteur.

C. Un problème subsiste...

Il existe toujours un problème avec la 2^e version du contrôleur. À basse fréquence il est plus difficile à observer, mais à haute fréquence il se produit fréquemment. Les concepteurs de ce contrôleur clavier ne comprennent pas bien la source du problème, il semble pourtant que la fréquence d'horloge utilisée respecte bien les contraintes liées aux délais du circuit. Voilà une belle occasion de briller : trouvez le problème et réglez-le !

Afin d'observer ce phénomène, créez un nouveau fichier schématique nommé **partie2_1_top.bdf**, et reprenez le design précédent mais en le cadencant directement avec l'horloge à 50MHz, n'oubliez pas de sélectionner le bon fichier comme **top-level**. Enregistrez votre nouveau design sous le nom **ctrl_clavier_bon.vhd** (qui devra également comprendre vos changements de la section précédente). Utilisez au besoin la simulation avec délais sous ModelSim pour vous aider à comprendre et résoudre le problème.

Note : le clavier sera réutilisé dans le projet final, dans lequel vous aurez besoin d'un contrôleur très stable. Arrangez-vous pour éliminer tous les bugs, même rares, le temps que vous y passerez sera du temps gagné pendant le projet !

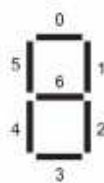
3. Utilisation de modules IP

Lorsque l'on fait de la conception de circuits, une pratique répandue consiste à utiliser des blocs d'IP (Intellectual Property) préconçus. De tels modules ont été rigoureusement vérifiés et ils sont souvent paramétrables. Une telle pratique offre l'avantage important de permettre au concepteur de concentrer son énergie sur la valeur ajoutée de son nouveau design et de diminuer considérablement l'effort de design et vérification.

La librairie d'Altera offre plusieurs blocs IP préconçus et paramétrables. Ainsi, on retrouve dans le dossier megafonctions de la librairie des modules LPM, tous paramétrables. Ces modules se séparent dans 3 classes dénotées arithmetic, gates, et storage. On y retrouve ainsi des additionneurs, diviseurs, constantes, multiplexeurs, FIFOs, et autres. Prenez un moment pour survoler les modules disponibles. Les modules LPM sont tous identifiés par le format « lpm_<module> ».

D. Conversion de format binaire à décimal

Dans cette partie, votre travail consistera à augmenter votre design (**partie3_top.bdf**) d'un circuit permettant d'afficher en format décimal sur les afficheurs 7-segments les valeurs des octets reçus du contrôleur clavier PS/2. Les octets pouvant avoir une valeur comprise en 0 et 255, nous aurons besoin de 3 afficheurs, HEX0, HEX1, et HEX2. Vous pouvez aller dans Assignments → Pin Planner pour identifier les noms des broches requis. Afin d'allumer un segment, il s'agit d'appliquer un '0' logique sur une des pattes de l'afficheur, un '1' éteint le segment. La carte de correspondance des segments est donnée ci-dessous :



En premier lieu, il faudra utiliser la sortie key_code du bloc schématique du contrôleur PS2 qui représente l'octet en question et trouver une façon d'aller chercher le chiffre des centaines, des dizaines et des unités. Par la suite, il faudra envoyer ces chiffres vers 3 modules « decodeur7segments » qui permettront d'afficher la valeur sur les 3 afficheurs hexadécimaux.

Lorsque vous sélectionnez un module LPM dans la librairie pour l'ajouter au design, un assistant de paramétrage apparaît et vous guide dans la configuration du module. Si vous désirez plus tard intégrer ce module dans un fichier VHDL plutôt qu'un fichier Schématique, dans la première fenêtre de configuration de l'assistant, indiquez le type de sortie en format VHDL plutôt que l'AHDL qui est mis par défaut. Il suffira ensuite d'aller lire la description du fichier pour pouvoir aller chercher la déclaration vous permettant de l'instancier dans un autre fichier VHDL (voir la documentation sur Moodle). Pour cette partie, nous n'aurons pas besoin de cette astuce.

Il serait tout indiqué de commencer par regarder le module **decodeur7segments.vhd** pour afficher un nombre de 0-9 sur l'afficheur hexadécimal. Celui-ci prend en entrée le nombre et fournit en sortie un vecteur de 7 bits permettant l'affichage 7-segments. Cependant, celui qu'on vous fournit n'est pas en logique inverse, à vous de le modifier...

Vérifiez que la valeur décimale affichée correspond bien au code binaire de l'octet du caractère. Bien entendu, vous ne pouvez pas tricher et utiliser un module d'Altera qui fait la conversion à lui seul, le but ici étant de vous familiariser avec les modules LPM et de réfléchir un peu!

Instructions pour la Remise

Voir :

- ✓ plan de cours
- ✓ page d'informations sur Moodle

Vous devez remettre votre rapport ainsi que votre dossier de projet (contenant tous les fichiers projets !) compressé. Envoyez le tout sur la page Moodle du laboratoire, sur laquelle vous pourrez répondre à deux questions très courtes (facultatif, cela nous aidera à améliorer les laboratoires pour les années suivantes).



**N'oubliez pas de sauvegarder votre travail
et de nettoyer le disque temporaire D !**

**Il est accessible à tout le monde
et est nettoyé toutes les nuits !**



Grille d'évaluation

Ce laboratoire compte pour 6% de la session :

Partie 1	1/6
Partie 2	3/6
Partie 3	2/6

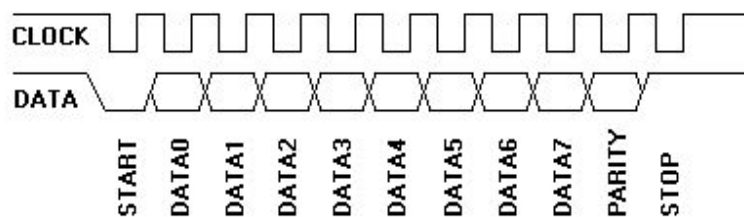
Version du 7 septembre 2014

Annexe 1

Le fonctionnement complet du clavier PS2 se retrouve sous format html dans le dossier de départ. Lors d'une communication, le clavier utilise un protocole sériel sur 11 bits :

- 1 bit de démarrage (toujours '0');
- 8 bits de données, en commençant par le bit le moins significatif;
- 1 bit de parité;
- 1 bit d'arrêt (toujours '1')

Le clavier écrit un bit de donnée lorsque l'horloge est à '1', et celui-ci est lu lorsque l'horloge est à '0'. La figure suivante montre ce fonctionnement :



Il est à noter que l'horloge (qui provient du clavier) reste à '1' en état *IDLE* (lorsqu'il n'y a pas de communication).

Annexe 2

Voici un tableau représentant quelques codes de presse (MAKE) et de relâche (BREAK) de quelques touches pour effectuer vos tests.

Vous trouverez le tableau complet sur <http://www.computer-engineering.org/ps2keyboard/scancodes2.html>.

KEY	MAKE	BREAK	KEY	MAKE	BREAK
A	1C	F0, 1C	N	31	F0, 31
B	32	F0, 32	O	44	F0, 44
C	21	F0, 21	P	4D	F0, 4D
D	23	F0, 23	Q	15	F0, 15
E	24	F0, 24	R	2D	F0, 2D
F	2B	F0, 2B	S	1B	F0, 1B
G	34	F0, 34	T	2C	F0, 2C
H	33	F0, 33	U	3C	F0, 3C
I	43	F0, 43	V	2A	F0, 2A
J	3B	F0, 3B	W	1D	F0, 1D
K	42	F0, 42	X	22	F0, 22
L	4B	F0, 4B	Y	35	F0, 35
M	3A	F0, 3A	Z	1A	F0, 1A