Ensimag 3A, filière SLE

2016-2017

Modélisation TLM en SystemC
TP n°3 : Demo de la plate-forme FPGA Zybo

Ce document décrit la plateforme matérielle sur FPGA modélisée dans le TP3, et la procédure pour la faire tourner en pratique. La plateforme est un système minimaliste classique (processeur, mémoire, contrôleur d'interruption, timer), qui contient aussi un contrôleur VGA qui permet un affichage à l'écran.

1 Lancement de Vivado

Dans un terminal, lancez :

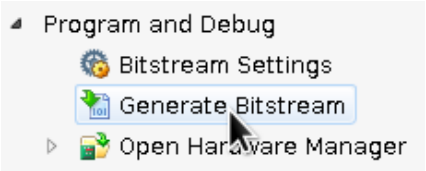
```
source /matieres/5MMMTSP/Xilinx/Vivado/2016.2/settings64.sh
vivado
```

Attention, la première ligne positionne des variables d'environnement de manière très intrusive. Il est probable que la plupart des commandes autres que `vivado` cessent de marcher, mais pas de panique : la modification ne s'applique qu'au terminal courant, il suffit d'en ouvrir un deuxième en cas de problème.

2 Ouverture du projet

Au lancement de Vivado, faites « open project », puis choisissez le fichier `TPs/squelette/tp3/zybo/project_1/project_1.xpr`. Le projet s'ouvre et Vivado affiche une fenêtre « Project Summary » peu intéressante pour l'instant. La barre latérale de gauche affiche les différentes étapes de la conception à l'implémentation. Cliquez sur « Open Block Design » dans la rubrique « IP Integrator ». Vous devriez obtenir le schéma de la figure 1.

La synthèse étant assez longue (environ 10 minutes), lancez-la dès maintenant en cliquant sur « generate bitstream » en bas à gauche de l'écran (les étapes se font de haut en bas, mais cliquer directement sur la dernière étape enchaîne toutes les autres automatiquement : synthèse, implémentation et génération du fichier .bit) :



En attendant que la synthèse soit terminée, quelques informations sur la plateforme dans la section suivante.

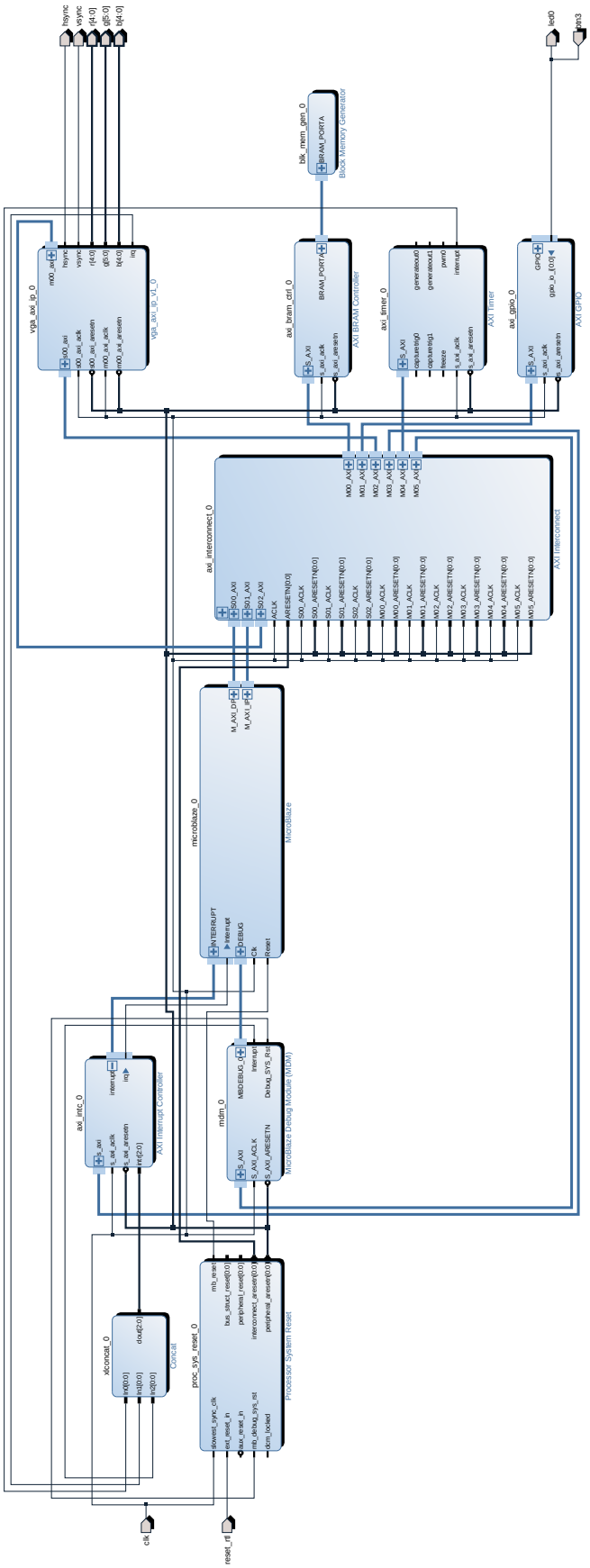


FIGURE 1 – Block Diagram de la plateforme

3 La plateforme

Le composant central de la plateforme est « AXI Interconnect » : c'est le bus de notre système. On voit beaucoup de détails absents d'une plateforme TLM typique : les signaux d'initialisation (`reset`), nécessaires pour initialiser la plateforme correctement, et horloges (`clk`) qui sont évidemment nécessaires physiquement, mais également pertinentes dans une représentation logique car il peut y en avoir plusieurs : il faut préciser quel composant est connecté à quelle horloge.

Les autres composants sont :

MicroBlaze, MicroBlaze Debug Module, Processor System Reset : Le processeur de la plateforme. Le processeur MicroBlaze a la particularité d'être conçu pour FPGA (on parle de « softcore »), et d'être très simple. Le « Debug Module » permet d'accéder au processeur via le JTAG pour pouvoir utiliser un debugger, et le « Processor System Reset » génère les signaux d'horloge et d'initialisation.

AXI Interrupt Controller est un contrôleur d'interruption. Il prend en entrée un signal 3 bits. On utilise donc le module « Concat » pour passer de 3 signaux 1 bit à un signal 3 bits (ce module est nécessaire du point de vue typage, mais ne correspond à rien physiquement).

vga_axi_ip_v1_0 Le contrôleur VGA (fait maison, merci à Frédéric Pétrot), qui lit une image (1 bit par pixel) en RAM et l'affiche à l'écran.

AXI BRAM Controller et Block Memory Generator correspondent à une mémoire. La BRAM est la mémoire embarquée dans le FPGA. Le premier module fait l'interface avec le bus AXI, et le second correspond à la mémoire à proprement parler.

AXI Timer est un timer. On le programmera pour envoyer des interruptions périodiquement pour rythmer l'exécution de la plateforme.

AXI GPIO (General Purpose Input Output) est un module d'entrée sortie. On l'utilisera pour gérer un bouton poussoir. Pour avoir un retour visuel, le bouton poussoir est également connecté à une LED.

Pour avoir les détails sur un composant, double-cliquez sur l'un d'eux. Il y a un bouton « documentation » en haut à gauche qui vous permet d'accéder au manuel et à la documentation en ligne pour chaque composant.

4 Branchement du FPGA

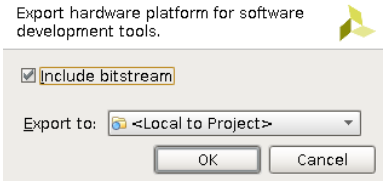
Même si la synthèse n'est pas terminée, vous pouvez brancher votre carte Zybo : le cordon USB sur le port USB du PC (utilisé pour programme le FPGA et pour le debug), et l'écran externe branché sur le port VGA du Zybo. Vérifiez que l'interrupteur SW4 à côté du branchement USB est bien sur « On ».

5 Lancement de l'environnement logiciel (SDK)

Pour cette étape, il est nécessaire que la synthèse soit terminée. Si ce n'est pas le cas, avancez sur la plateforme TLM en attendant. Quand la synthèse se termine, une fenêtre « Bitstream Generation Completed » s'ouvre : elle n'est pas importante pour nous, vous pouvez la fermer en cliquant sur « Cancel ».

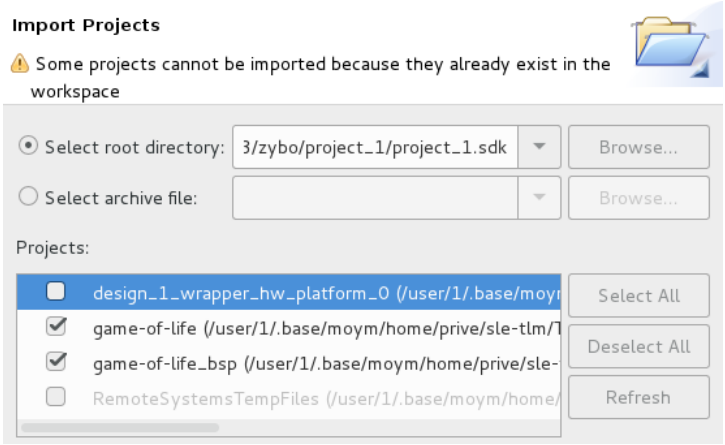
Le SDK Xilinx permet de développer du logiciel embarqué paramétré par le système matériel, et permet de programmer le FPGA. Il a donc besoin d'une description du matériel, que

nos allons lui fournir : depuis Vivado, faites « File » → « Export » → « Export hardware ». Cochez la case « include bitstream » et validez :



Lancez maintenant le SDK : menu « File » → « Launch SDK ». Gardez les paramètres par défaut et validez : une version spécialisée Xilinx d'Eclipse s'ouvre. L'espace de travail (*workspace*) contient pour l'instant un projet « `design_1_wrapper_hw_platform_0` » : ce projet correspond à la description du matériel que nous venons d'exporter. Le SDK l'utilise, mais nous n'y toucherons pas.

Ouvrez le projet « game-of-life » : menu « File » → « Import... ». Choisissez « General » → « Existing projects into workspace ». Choisissez « `project_1.sdk` » comme répertoire racine et importez « game-of-life » et « game-of-life_bsp » :



Le projet « game-of-life » est notre logiciel embarqué. Si vous ne connaissez pas le jeu de la vie, voir par exemple : `http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life`. Le projet « game-of-life_bsp » (BSP = Board Support Package) contient le logiciel de bas niveau spécifique à notre plateforme. Il a été généré automatiquement par Vivado. Nous n'utiliserons que très peu ce BSP : le logiciel tourne sur machine nue et accède directement aux registres des périphériques. L'intérêt pour nous est de maîtriser et comprendre totalement ce qu'il se passe, mais dans la vraie vie le BSP est bien utile !

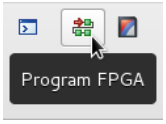
Dans le projet « game-of-life », cherchez le fichier `main.c`. C'est exactement le même que celui de la plateforme TLM.

On trouve aussi le fichier `hal.h`, qui implémente une partie des primitives vues en cours en utilisant le BSP Xilinx (`Xil_In32`, `Xil_Out32`, `xil_printf`).

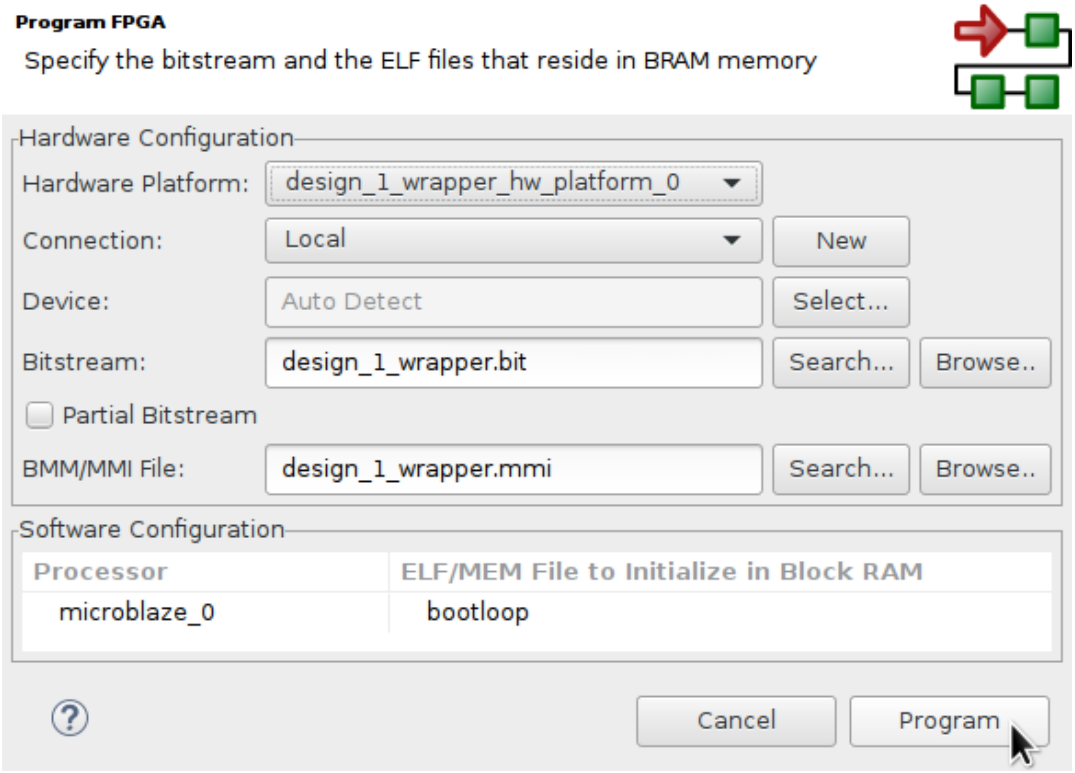
Les autres fichiers sont ceux définissant l'adressmap, identiques à ceux utilisés côté TLM.

6 Programmation du FPGA

Lancez la programmation du FPGA avec le bouton de la barre d'outils :



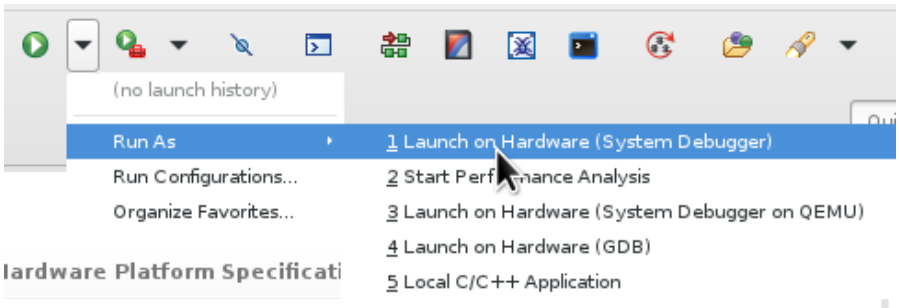
Gardez les paramètres par défaut et validez :



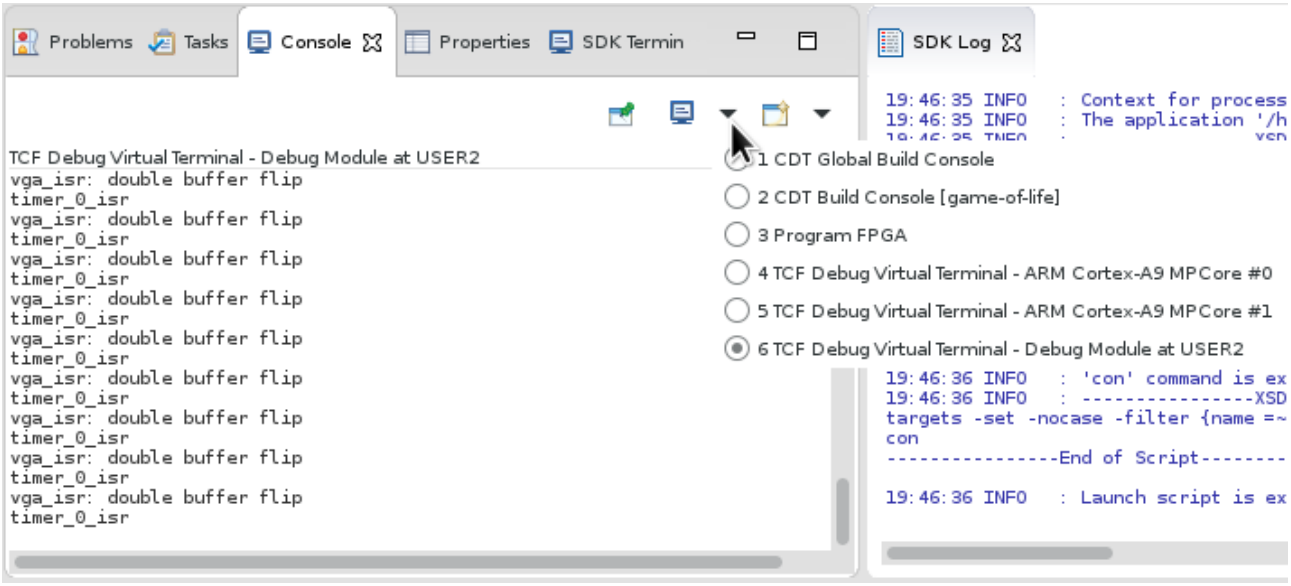
Pendant la programmation, vous aurez sans doute un avertissement « There is no PS in the design » (PS = Processing System), sans gravité : ignorez-le et validez avec « OK ».

Une fois le FPGA programmé, une mire (bandes noires et blanches verticales) apparaît à l'écran : c'est le motif par défaut de notre module VGA, qui n'est pas encore programmé, c'est normal !

Lancez maintenant l'application logicielle : dans « Project Explorer », sélectionnez « game-of-life », puis cliquez sur le bouton « Run As ... » et choisissez « Launch on hardware (System Debugger) » :



L'application s'exécute, le jeu de la vie doit apparaître à l'écran. On peut récupérer la sortie de printf directement depuis le SDK dans la fenêtre « console » (si besoin, choisissez la console « TCF Virtual Terminal, Debug Module ») :



Sur la carte FPGA, trouvez le bouton poussoir « btn3 (y16) » et appuyez dessus : une LED s'allume, et un glider supplémentaire doit apparaître à l'écran. Retrouvez dans le logiciel embarqué (`main.c`) la boucle d'ajout des gliders.

Vous pouvez essayer de modifier le logiciel embarqué, par exemple changer un message affiché par `printf`. Sauvegarder le fichier recompile l'application. Il n'est pas nécessaire de re-programmer le FPGA : cliquer sur « Run as » relance l'application.

7 Pour les curieux : les bugs du hardware

La partie qui suit n'est pas indispensable, mais si vous êtes curieux et que vous avez le goût du challenge, continuez à lire !

La plateforme matérielle est suffisante pour faire tourner notre logiciel, mais le contrôleur VGA comporte en réalité plusieurs bugs d'affichage.

Le premier est que la lecture en RAM n'est pas synchronisée explicitement avec l'affichage à l'écran. Par conséquents, il est possible que l'image soit décalée verticalement et/ou horizontalement. Ça n'arrive pas en pratique car tous les composants sont démarrées en même temps ... sauf si on ré-initialise la plateforme manuellement : appuyez sur le bouton BTN0(R18) pour le faire, et voyez l'image se décaler. Ce bug est bien identifié mais non-trivial à corriger.

Le second est que certains pixels sont affichés au mauvais endroit. On ne le voit pas sur le jeu de la vie où l'écran est essentiellement noir, mais essayez d'afficher une mire avec du noir et du blanc, et vous pourrez observer quelques portions de lignes mal affichées. Pour ce bug là, personne ne sait d'où il vient (sauf peut-être vous, chers lecteurs ?) !