|  |
| --- |
| École Polytechnique De Montréal |
| INF3610 Laboratoire 4 Partie 1 |
| Conception logicielle/matérielle à partir d’un modèle du Zynq |

|  |
| --- |
| Arnaud Desaulty  06/11/2015 |

Table des matières

[Introduction 2](#_Toc434524369)

[Objectifs du laboratoire 2](#_Toc434524370)

[I. Familiarisation avec SpaceStudio et présentation de la calculatrice 3](#_Toc434524371)

[1. La librairie SpaceLib 3](#_Toc434524372)

[a. Modules 3](#_Toc434524373)

[b. API de communication 5](#_Toc434524374)

[2. L’interface utilisateur : SpaceStudio 6](#_Toc434524375)

[a. Fenêtre principale 6](#_Toc434524376)

[b. Configurations 7](#_Toc434524377)

[c. Technologies de simulation 9](#_Toc434524378)

[d. Génération du projet 11](#_Toc434524379)

[e. Affichage des résultats du monitoring 11](#_Toc434524380)

[II. Travail à effectuer 12](#_Toc434524381)

[1. Tutoriel du JPEG 12](#_Toc434524382)

[2. Test des fonctionnalités sur la calculatrice 12](#_Toc434524383)

[3. Implémentation de la multiplication matricielle 12](#_Toc434524384)

# Introduction

La conception de système sur puce ou sur FPGA peut être un processus long et coûteux. Dans un contexte où les temps de mise sur le marché sont de plus en plus court et la complexité des systèmes grandit, les méthodes traditionnelles de conception ont beaucoup de mal à suivre le rythme. Il est donc nécessaire de penser à de nouvelles méthodes de conception pour de tels systèmes. La nécessité d’effectuer des simulations à un haut niveau d’abstraction apparait donc comme une solution intéressante à ce problème. En effet, avec des solutions telles que SystemC, il est possible de valider un design à différents niveaux d’abstraction, en commençant par une solution fonctionnelle, puis en affinant la solution jusqu’à obtenir une solution synthétisable et donc pouvant être implémentée.

A ce principe d’abstraction à haut niveau vient s’ajouter un second problème, celui de l’exploration architecturale. En effet, le design défini plus haut doit être appliqué à une architecture matérielle définie. Cette plateforme peut être un simple FPGA, une carte de développement avec processeurs et FPGA ou bien encore un ASIC. Il convient alors de fragmenter le design haut niveau et de l’appliquer aux ressources matérielles disponibles. Ce mappage est un travail très complexe qui peut nécessiter des quantités importantes de temps et d’expertise, surtout quand des tests doivent être effectués sur un prototype après synthèse du code.

Cette double nécessité de vérification à haut niveau de la fonctionnalité et de l’architecture a conduit à la naissance des outils de co-design logiciel/matériel. Ces outils permettent un flot de conception itératif, de la validation du code à son implémentation logiciel ou matérielle, à l’intérieur d’un seul outil. La simulation à haut niveau du mapping à l’aide des simulateurs de jeux d’instructions et de langages de description matérielle permettent d’explorer plus efficacement l’espace architectural.

# Objectifs du laboratoire

Au cours de ce laboratoire, nous allons vous présenter un logiciel de co-design: SpaceStudioTM. Après un cours tutorial sur l’utilisation de cet outil, il vous sera demandé d’implémenter une nouvelle fonctionnalité en suivant le flot de conception de SpaceStudio. Puis vous devrez explorer l’espace de solution afin de rendre votre nouvelle fonctionnalité la plus rapide possible.

L’objectif de ce laboratoire consiste à:

* introduire l’étudiant à la conception de SoC à haut niveau d’abstraction,
* initier l’étudiant au fonctionnement d’un logiciel de co-design et à son flot de conception,
* valider un système avec la co-simulation logicielle/matérielle,
* exposer l’étudiant à une brève exploration architecturale visant à améliorer l’efficacité de votre solution[[1]](#footnote-1).

# Familiarisation avec SpaceStudio et présentation de la calculatrice

Pour cette introduction aux fonctionnalités de SpaceStudio, nous allons nous intéresser au code qui vous est fourni : le projet Calculator :

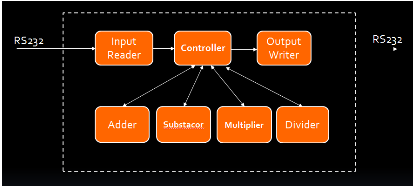
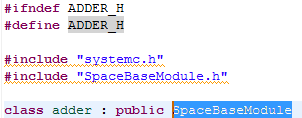


Figure : Modèle d'une calculatrice

## La librairie SpaceLib

### Modules

Les encadrés oranges dans la figure 1 sont des **modules.** Ces modules représentent des parties de la fonctionnalité complète du système. Un module SpaceStudio est une classe C++ qui hérite de la classe *SpaceBaseModule*



Code : Déclaration d'un module (adder.h)

Comme vous pouvez le voir dans le fichier *adder.cpp*, le constructeur du module *adder* appelle le constructeur de *SpaceBaseModule*. Vous pouvez aussi remarquer que nous incluons dans le header *systemC.h*. Cela est dû au fait que la librairie de SpaceStudio (appelé SpaceLib) est basée sur SystemC. Cela est encore plus visible avec la *SC\_THREAD(thread)* dans le constructeur ainsi que *SC\_HAS\_PROCESS(adder).*

La méthode thread doit décrire l’intégralité du fonctionnement du module. Elle est souvent défini sous la forme : Récupération de données, travail sur ces données, envoi des données.

Ainsi un module SpaceStudio est une classe C++:

* basée sur la classe *SpaceBaseModule*, elle-même basée sur un module SystemC
* possédant un *SC\_THREAD (uniquement !)*
* Qui récupère des données, travaille dessus et renvoie un résultat

Pour plus d’informations sur les modules, vous pouvez consulter le fichier d’aide de SpaceStudio

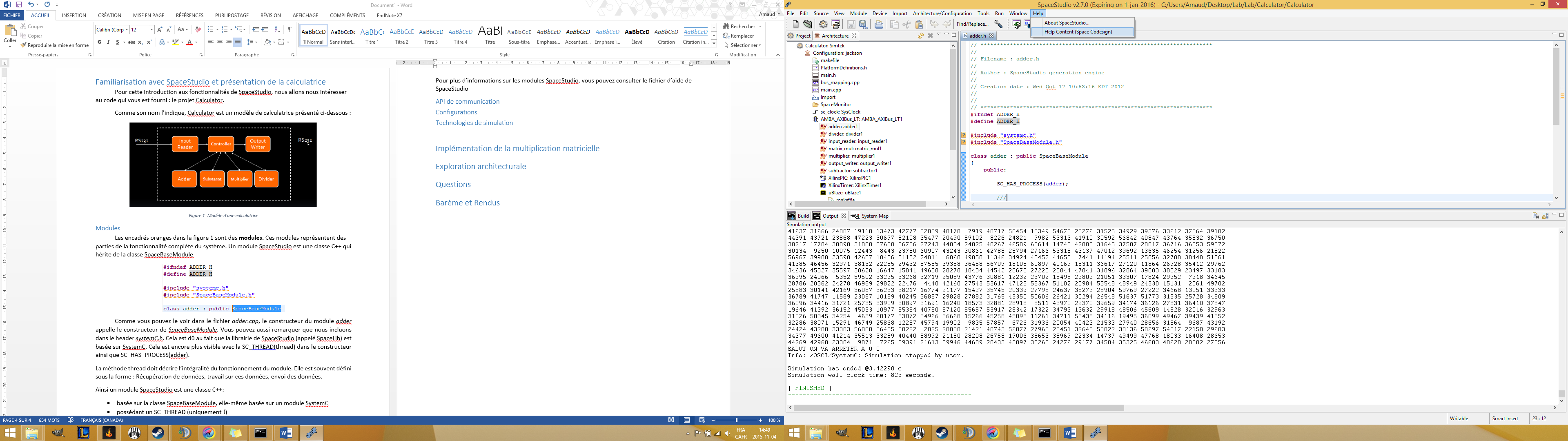
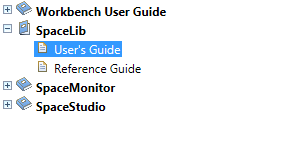
 

Figure : Accès à la documentation de SpaceLib

### API de communication

Afin d’envoyer et de recevoir des données entre les modules, SpaceLib définit une API de communication. Les fonctions les plus basiques de cette API sont moduleRead et moduleWrite. Ces fonctions permettent la communcation entre deux modules par le biais d’une fifo. La communication doit donc être effectuée des deux côtés, c’est à dire qu’un write doit toujours aller de pair avec un read. Lorsque le read et le write sont bloquants, la rencontre des deux se nomme *rendez-vous* et permet donc l’échange de données et la synchronisation (semblable à la sémaphore) des modules. A noter que chaque module possède sa propre ID dans SpaceStudio.

L’API de SpaceStudio permet d’autres types de communications que vous pouvez retrouver dans le manuel utilisateur (cf. Figure 2) et que nous verrons dans la deuxième partie du lab 4.

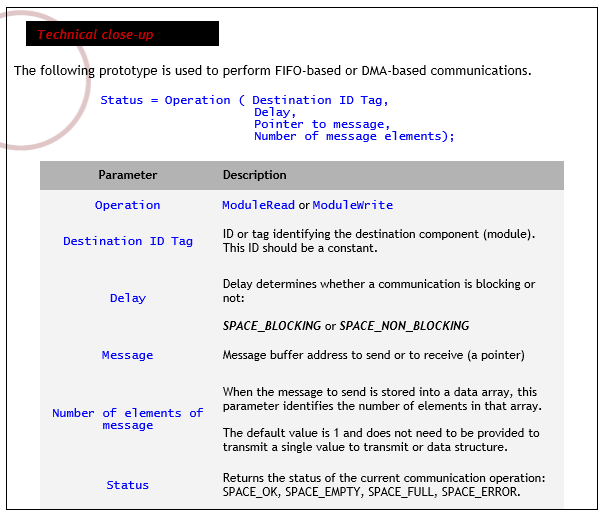


Tableau API de communication FIFO (tirée du manuel utilisateur [1])

## L’interface utilisateur : SpaceStudio

### Fenêtre principale

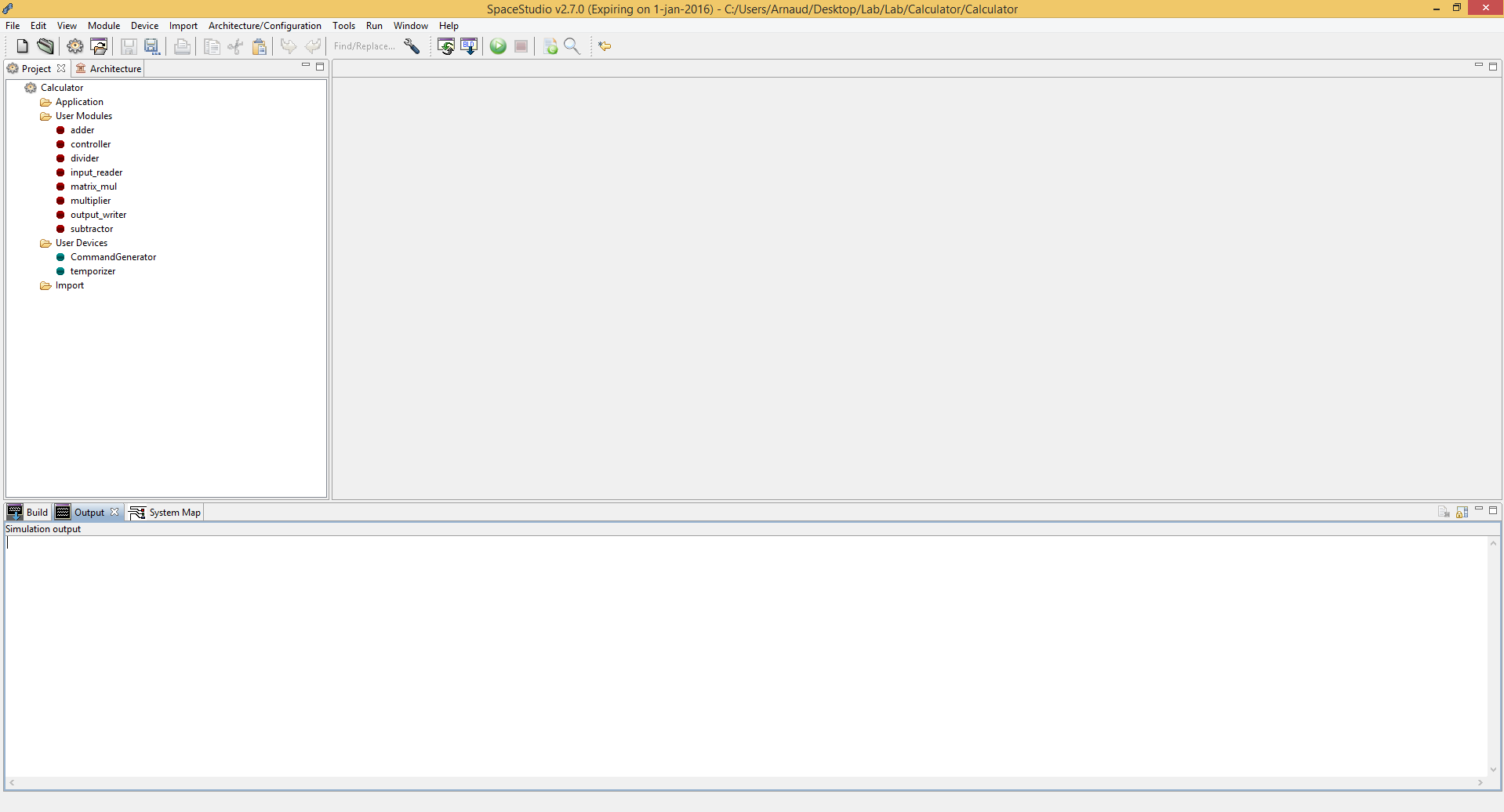


Figure Fenêtre principale de SpaceStudio

La fenêtre principale de SpaceStudio se décompose en plusieurs éléments importants :

1. **La fenêtre projet/architecture** : la perspective projet vous permet d’afficher les sources que vous utilisez dans votre code (que celles-ci soient des modules, des devices ou du code C++). La perspective architecture devient disponible lorsque vous chargez une configuration et montre le partitionnement de la configuration (c’est-à-dire le mappage module/ressources)
2. **La barre d’outils** : vous permet d’effectuer les actions principales d’un IDE classique. Notez l’icône qui vous permet d’ouvrir le gestionnaire de configuration ainsi que l’icône qui vous permet de lancer la simulation pour la configuration en cours (une fois que celle-ci a été générée et compilée).
3. **La barre de menus** : C’est depuis cette barre que vous avez accès au reste des fonctionnalités de SpaceStudio, comme l’ajout de module, de devices ou l’importation de sources. Vous pouvez aussi créer et sélectionnez la configuration active. C’est aussi depuis ce menu que vous aurez accès aux outils de génération du code (dans Tools.), aux outils de profilage.

### Configurations

Pour commencer à travailler dans SpaceStudio, vous devez séléctionner une configuration. Une configuration contient les informations sur la technologie de simulation que vous voulez utiliser, la plateforme matérielle sur laquelle vous voulez tester votre solution ainsi que votre mappage des modules à vos ressources.

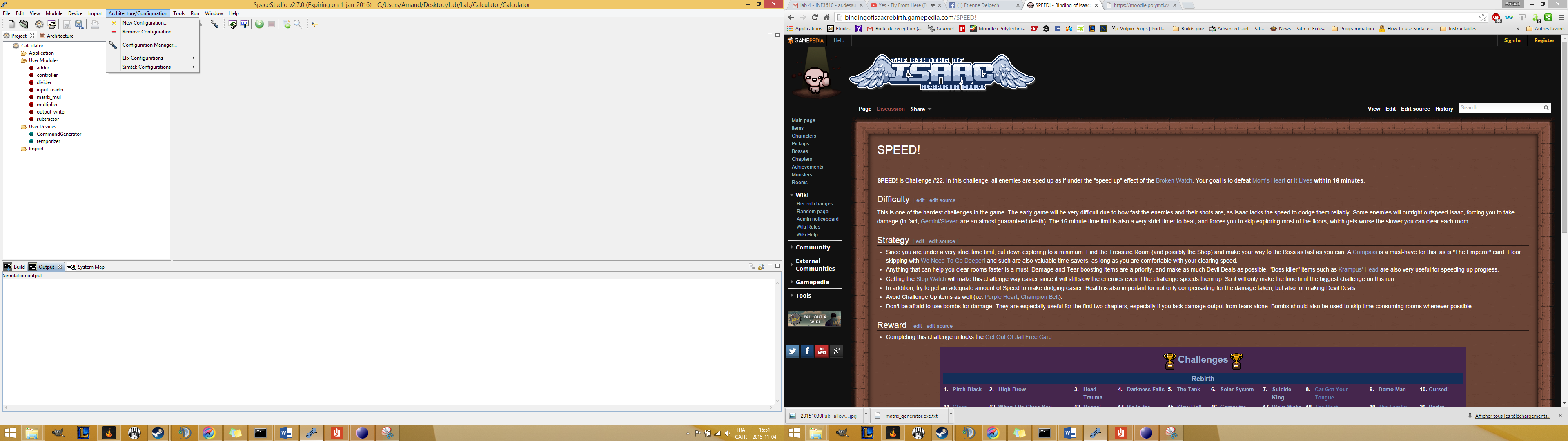


Figure Menu de gestion des configurations

Pour choisir une configuration active, il suffit de dérouler la liste des configurations Elix ou Simtek et de sélectionner une des configurations déjà créées.

Pour créer une nouvelle configuration cliquez sur New configuration, ce qui va permettre d’ouvrir la fenêtre de configuration :

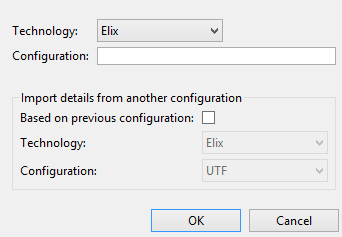


Figure Nom et choix de la technologie

Dans cette fenêtre vous devez renseigner le nom et la technologie de simulation que vous voulez assigner à votre module. Vous avez de plus le choix de calquer une autre configuration du même type. Une fois ces actions effectuées, votre nouvelle configuration est sélectionnée comme active. Pour commencer à la modifier, il faut entrer dans le configuration manager :

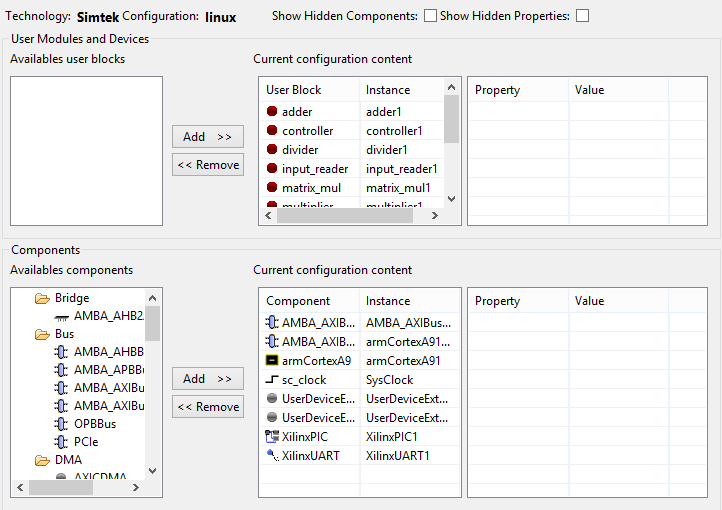


Figure Première fenêtre du Configuration Manager

Le *configuration manager* est séparé en deux parties. *User modules* and *devices* est la première partie et vous permet de choisir quels modules de votre projet vous voulez introduire dans votre configuration. Seuls ces modules seront actifs pour la simulation.

La deuxième partie *Components* vous permet d’instancier les composants qui composent votre plateforme matérielle. Selon la technologie de simulation utilisée, les composants disponibles ne seront pas les mêmes. Plusieurs technologies sont supportées dont celle du Zynq de Xilinx que vous avez utilisée au laboratoire no 2.

Une fois votre architecture composée, vous pouvez cliquer sur suivant pour accéder au mapping *modules/Architecture* :

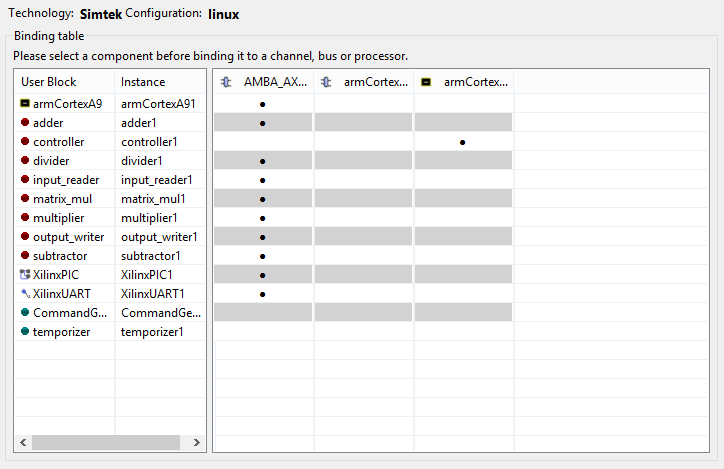


Figure Mappage des modules

Dans cette table, vous pouvez choisir a quelle composante de votre architecture vous voulez connecter vos différents modules. Ainsi le module contrôleur est assigné au processeur Arm Cortex et tournera donc en logiciel pour la simulation. Le reste des modules est assigné au bus de communication et sont donc par défaut considérés comme des modules matériels.

### Technologies de simulation

Comme cité dans l’introduction, SpaceStudio possède plusieurs technologies de simulation afin de créer une approche incrémentale lors du design d’un SoC. Ainsi, les deux technologies auxquelles vous aurez à faire sont Elix et SimTek.

Elix permet la vérification fonctionnelle de vos modules et ne requiert pas de plateforme matérielle. Il suffit juste d’un bus de communication qui relie tous les modules. Faire tourner votre code sous Elix vous permet de valider votre design avant de passer à l’exploration architecturale.

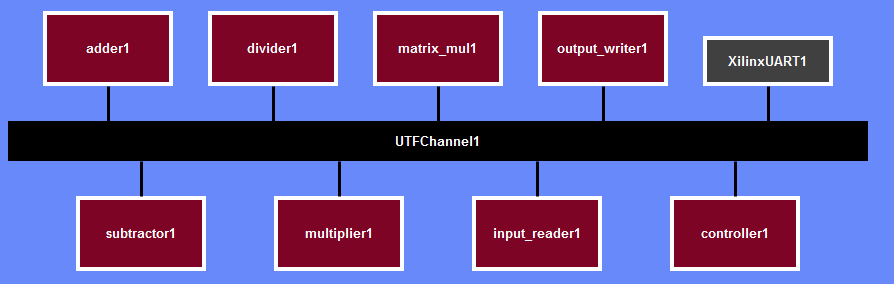


Figure Schéma du système sous Elix

SimTek vous permet, quant à lui, de simuler votre code en rapport avec une architecture matérielle donnée (p.e. une abstraction de la technologie du Zynq). Ainsi, la façon dont vous mapperiez vos modules à votre architecture influencera les performances du programme.

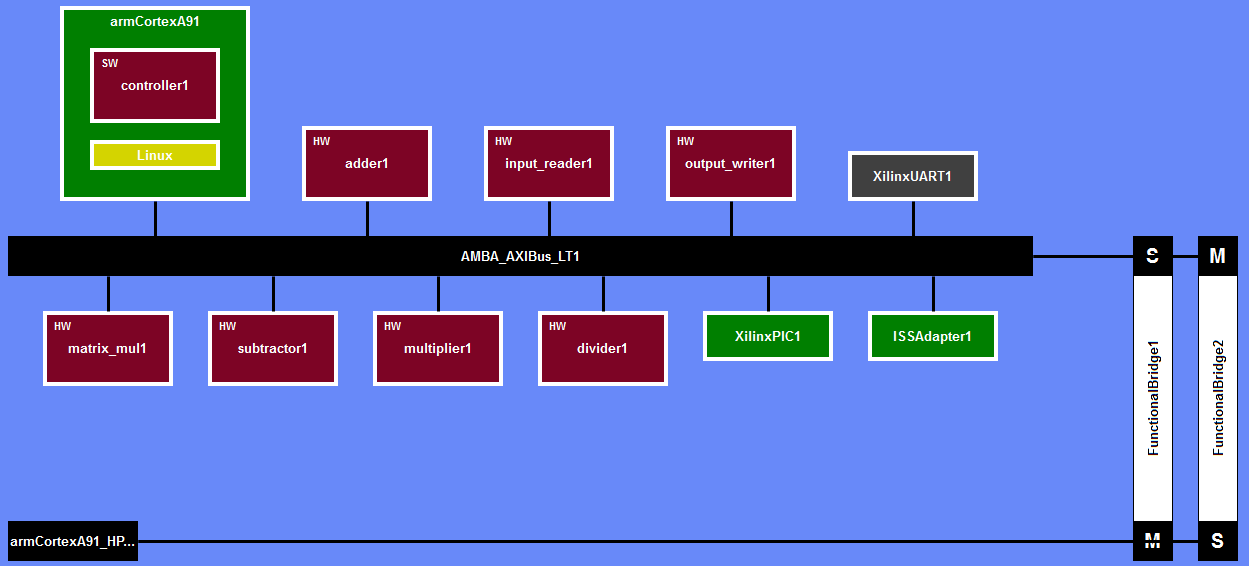


Figure Schéma du système sous SimTek

### Génération du projet

Pour pouvoir lancer votre configuration, vous avez plusieurs étapes à suivre.

Vous devez tout d’abord générer les fichiers nécessaires à la compilation. Pour ce faire, lancez *generate* depuis le menu Tools. Lors de la génération vous avez plusieurs options :

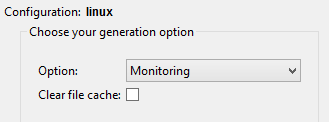


Figure Menu generate

Les options qui nous intéressent sont les options *Debug* et *Monitoring*. *Debug* permet l’utilisation du debugger tandis que *monitoring* va rajouter de l’instrumentation au code afin de pouvoir analyser des métriques comme l’utilisation du processeur. Bien entendu, le monitoring va augmenter le temps nécessaire à la simulation.

Une fois le projet généré avec l’option de votre choix, il faut maintenant compiler. Pour ce faire, il faut cliquer sur Tools -> Build -> All.

Enfin une fois le projet compilé, vous pouvez cliquer sur  pour lancer la simulation.

### Affichage des résultats du monitoring

Une fois la simulation terminée, vous pouvez récupérer les résultats du monitoring en cliquant sur Tools -> SpaceMonitor.

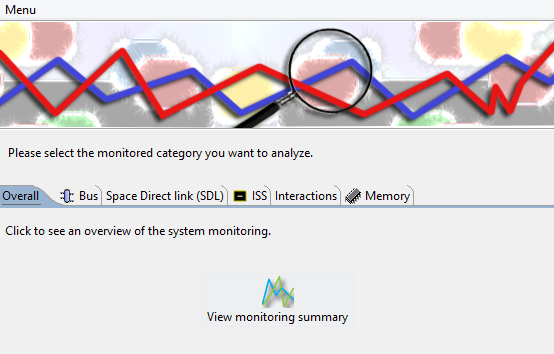


Figure Menu de SpaceMonitor

Vous pouvez alors cliquer sur *view monitoring summary* pour avoir les informations principales de votre simulation.

# Travail à effectuer

## Tutoriel du JPEG

Afin de vous familiariser avec les commandes décrites ci-haut, nous vous recommandons fortement de faire le tutoriel sur le JPEG. Une fois que vous aurez complété le tutoriel, vous pourrez passer à l’étape suivante.

## Analyse des fonctionnalités de la calculatrice

Vous devez prendre connaissance du projet *calculator* qui vous est fourni. Ce projet est fonctionnel et contient deux configurations : une configuration Elix appelée UTF et une configuration SimTek appelée linux. Intéressez-vous tout particulièrement au fonctionnement du module *controller*.

Les deux configurations possèdent un UART, qui à terme, est supposé permettre le test du projet synthétisé sur carte à l’aide d’un système de commande envoyée à l’aide d’un terminal comme *putty*. Toutefois, pour la simulation, l’UART reçoit directement les commandes de calcul depuis le device *command\_generator*. Familiarisez-vous bien avec ce code.

## Implémentation de la multiplication matricielle

Pour ce labo, il vous est demandé d’implémenter une nouvelle fonctionnalité sur la calculatrice, et il s’agit de la possibilité d’effectuer des multiplications matricielles. Pour ce faire, voici quelques indications :

* Vous devez modifier le module *controller* ainsi que le module *command\_generator* afin de permettre l’envoi de matrices.
* *command generator* doit définir un nouveau symbole d’opération (prenez « z ») et envoyer les index des matrices sur lesquelles travailler. Ainsi l’operation « 1 z 2 » demandera en fait de multiplier les matrices *matrix\_data[1] et matrix\_data[2].*
* Vous devrez créer un nouveau module appelé *matrix\_mul* directement dans SpaceStudio. Vous pouvez vous inspirer des autres modules de calculs pour créer ce module.
* Un exécutable vous est fourni pour créer des matrices. Pour générer des matrices, il suffit de l’appeler via un invite de commandes avec trois options : nombre de rangées, nombre de colonnes et nombre de matrices. Une fois le code exécuté, vous récupérez quatre fichiers que vous devrez importer dans votre projet SpaceStudio (il faut renommer le fichier .c en .cpp).
* Vous pouvez valider le fonctionnement de votre code sur la configuration UTF

Pour rappel, le produit matriciel d’une matrice de taille (m,n) et d’une matrice de taille (n,p) noté est tel que :



Formule Produit matriciel

1. En ce sens, il s’agit d’une petite introduction au cours INF8500 conception et vérification des systèmes embarqués. [↑](#footnote-ref-1)