5 Génération du système 3 : multiplieur de matrices Hardware

5.1 Répondez aux questions suivantes

a) En utilisant la documentation Xilinx et Internet, décrivez les principales caractéristiques du bus AXI-lite

b) En une page, décrivez l'interface matériel/logiciel de cette IP. Décrivez l'interface logicielle : les entrées/sorties de la méthode mat_mult.c. Décrivez l'interface matérielle : les entrées sorties du mat_mult.vhd. Ce document doit décrire de manière non-ambiguë l'interface entre le matériel et le logiciel. En particulier, il doit définir les types des données échangées (sous quelle forme et dans quel ordre le logiciel envoie au/reçoit du matériel chaque élément de matrice)

5.2 Mise en place de l'environnement de développement « IP »

Afin de pouvoir s'interfacer avec le processeur ARM et entrer dans le flow de design Xilinx, l'IP que vous allez créer devra respecter un certain formalisme et certaines interfaces. VIVADO vous permettra de générer les fichiers qui vous serviront de base au développement.

5.2.1 Effectuez les opérations suivantes

1. Depuis votre projet VIVADO, allez dans « Tools »> « Create and Package New IP ».
2. Cliquez Next, et choisissez « Create a new AXI4 peripheral »
3. Choisissez les paramètres de nommage suivants
   Name : mat_mult
   Version 1.0
   IP location : […]Codesign_projectfiles/ip_repo (que vous aurez pris le soin de créer).
4. Choisissez les paramètres suivantes pour le bus AXI :
   Interface Type : Lite
   Mode : Slave
   Data Width : 32 bits
   Nomber of Registers : 128
5. A la fin de l'assistant choisissez « Edit IP ». L'IP est à présent crée, vous pouvez à maintenant l'éditer
6. Pour ajouter les fichiers .vhd sources qui vous sont fournis, allez dans « Add Sources » en faisant un clic droit sur les « Design sources » apparaissant à gauche de l'écran. Suivez l'assistant, et veillez à cocher « Copy sources into IP Directory » lorsque ce sera proposé

5.2.2 Intégration à mat_mult

7. Analysez le fichier mat_mult_v1_0_S00_AXI.vhd. C'est dans ce fichier que vous devrez instantier le matrix_multiplier

Nota : il est recommandé de commencer par le codage du matrix_multiplier (5.3) avant de réaliser son intégration au sein de mat-mult.

Le fichier proposé instantie des registres accessibles par le logiciel. Tel qu'implémenté lorsque le template est généré :

- slv_regx sont accessibles en lecture par votre IP lorsqu'ils sont écrits par le logiciel. Vous pouvez donc connecter ces signaux à votre logique sous la ligne *« -- Add user logic here »*
- Pour mettre à disposition du logiciel des signaux provenant de votre IP, il s'agit de modifier le process situé sous : « -- Implement memory mapped register select and read logic generation ». Par exemple, la ligne de code suivante dans le process permet de rendre accessible le signal my_signal par une lecture du logiciel à l'adresse « 0 »

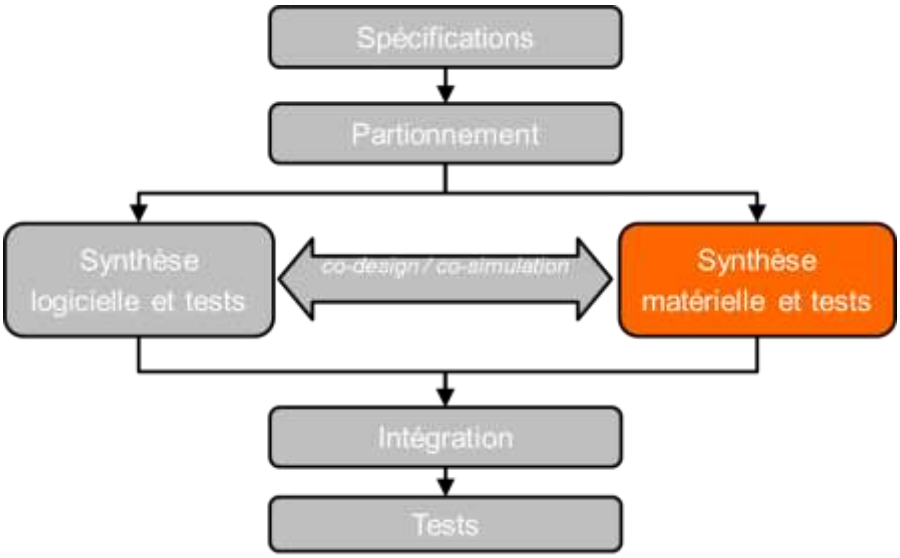
```
when b"0000000" =>
reg_data_out <=my_signal;
```

Vous pourrez ainsi instancier l'IP matrix_multiplier et la connecter au bus AXI. Lorsque vous aurez terminé la phase de codage et réussi la synthèse, réalisez les étapes suivantes pour passer aux tests sur cible.

8. Dans l'onglet « Package IP », rendez-vous a la ligne « File groups », puis « Merge Changes from File Groups Wizard »
9. A la ligne « Review and Package », cliquez sur « Re-Package IP » pour terminez le packaging de l'IP

5.3 Développement du module matrix_multiplier

Vous allez à présent coder, synthétiser et tester le matrix_multiplier



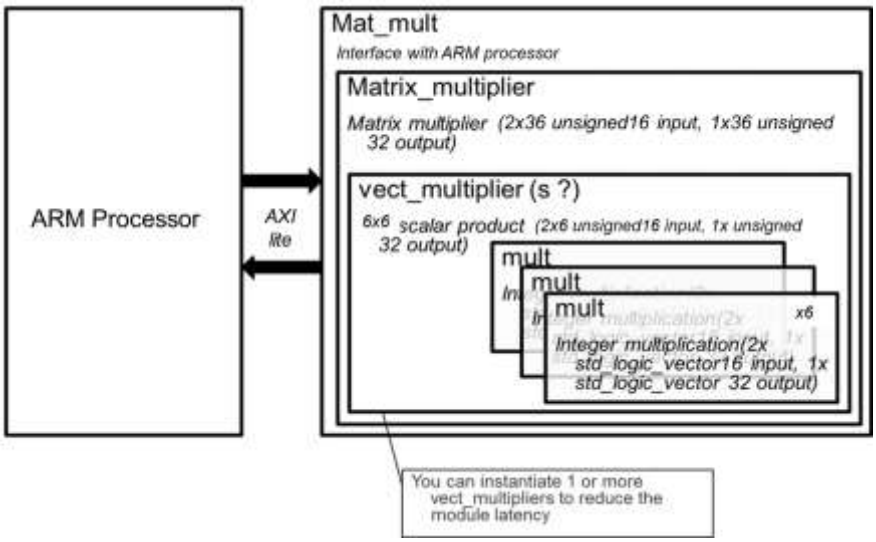
5.3.1 Architecture

matrix_multiplier

Le module « matrix_multiplier.vhd » est le module central de votre IP. Un template de ce fichier vous est fourni dans le répertoire design sources. Vous devez l'utiliser pour votre design et il est interdit d'en modifier l'interface.

Ce module réalise la multiplication de 2 matrices. Le résultat est valide après un certain nombre de coups d'horloge qui dépendra de votre design. Nous appellerons ce paramètre latency* et il devra être pris en compte dans le bloc de niveau supérieur mat_mult.

** : attention il ne s'agit pas à proprement parler de la latence telle que définie pour un circuit pipeliné.*



La multiplication de matrices peut être décomposée en autant de produits scalaires que le nombre d'éléments de la matrice résultat. En effet, chaque élément de la matrice résultat est le produit de deux vecteurs. Cette opération sera réalisée par un module « vect_multiplier.vhd ». Afin de réaliser la multiplication de matrices, vous pourrez utiliser un seul vect_multiplier que vous utiliserez de manière séquentielle pour calculer chacun des éléments de la matrice résultat, ou vous pourrez implémenter plusieurs vect_multiplier afin de réaliser les opérations en parallèle.

vect_multiplier

vect_multiplier réalise un produit scalaire, il a besoin pour cela de réaliser autant de multiplications que la taille de ses vecteurs d'entrée et il doit aussi réaliser des additions. Le module vect_multiplier vous est fourni.

5.3.2 Objectifs

Les objectifs suivants garantissent le bon fonctionnement de votre design et sa réutilisation dans différentes applications. Ils sont pris en compte dans l'évaluation de votre travail.

- Votre IP devra réaliser l'opération multiplication de matrices le plus rapidement possible
- Votre IP + le processeur ARM ne devront pas utiliser plus de ressources que celles disponibles sur la cible
- La fréquence d'horloge est 100 MHz. Votre IP devra supporter cette fréquence d'horloge une fois synthétisée sur la cible
- Commentez votre code afin qu'il soit compréhensible et puisse être maintenu dans le futur par un autre développeur
- Respectez les règles de nommage définies en entête du fichier « mat_mult.vhd ».
- Respectez les règles de codage vues en cours
- Utilisez au moins une fois la syntaxe for generate vue en cours
- Il faudra que le module matrix_multiplier soit évolutif par rapport à la taille des matrices qu'il devra traiter. Vous utiliserez les constantes définies dans le package definitions_mult dans votre code : utilisez vector_size au lieu de 5 ou matrix_size au lieu de 25 dans votre code afin que, dans le futur, il n'y ait qu'à modifier la valeur de ces constantes pour traiter des matrices de tailles différentes

5.3.3 Méthodologie

- Pour concevoir le système, représentez, sur le papier et de manière schématique, l'implémentation de votre IP. Evaluez les latences et le nombre de ressources critiques (les multiplieurs) de votre design
- Validez le fonctionnement de votre module matrix_multiplier grâce au testbench « matrix_multiplier_bench.vhd » qui vous est fourni dans les sources
- Assurez-vous que votre design ne nécessite pas plus de ressources que le FPGA peut en offrir et que les timings sont OK avec une clock à 100 MHz

5.3.4 Complétez le compte rendu

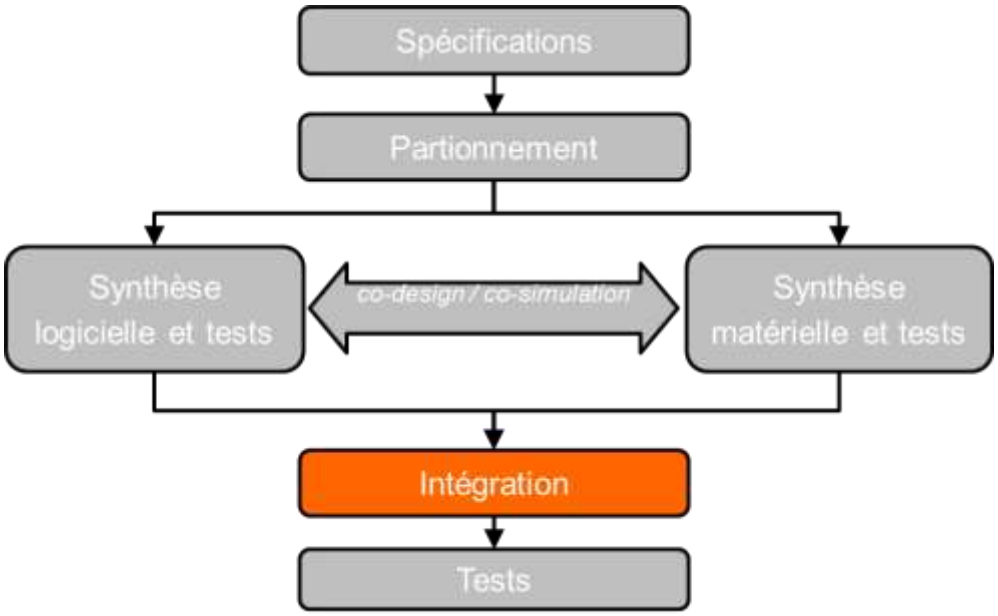
c) Mettez dans votre compte rendu une représentation schématique de votre module matrix_multiplier
d) Joignez au compte rendu la sortie texte de l'exécution du testbench matrix_multiplier_bench et un screenshot de la fenêtre wave du simulateur centrée sur un essai
e) Combien de multiplieurs votre design utilise-t-il ?
f) D'après le testbench, quelle est la latence de votre module ?
g) Collez dans votre compte rendu un exemple de l'utilisation de l'instruction for generate
h) Collez dans votre compte rendu des exemples issus du code que vous avez écrit, illustrant le respect d'au moins 7 des règles ou recoms de codage vues en cours. Identifiez explicitement chaque règle ou recom exemple (Num 6.1, Utilisation d'Assert, recom) »

5.3.5 Codage

Codez le module mat-mult conformément aux instructions données en 5.2.2 Intégration à mat_mult

6 Génération du système 2 : Intégration de l'IP sur le bus AXI lite

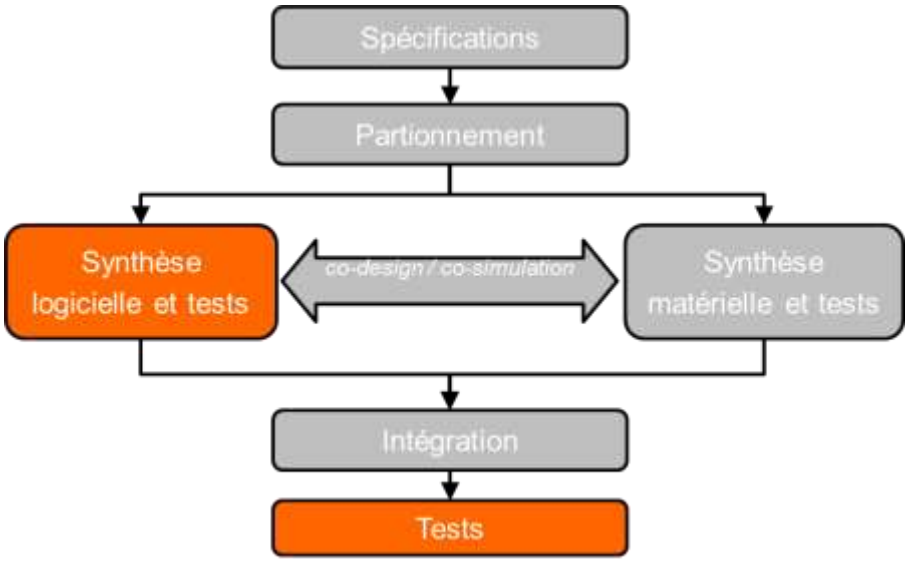
6.1 Intégration sur cible



Le moment est venu d'intégrer votre IP au système ARM.

1. Rendez-vous dans le diagramme schématique de votre système initial.
2. Rendez-vous sur « Add IP » pour intégrer votre IP au système, puis lancez la « run connection automation » proposée » pour faites les connections nécessaires. Nota : choisissez la même clock pour l'interface master et slave.
3. Lancez ensuite la synthèse, la génération du bitstream et faites l'import vers le SDK tel que vu précédemment.

6.1 Intégration et tests sur cible



1. Après l'export du nouveau système, rendez-vous dans le SDK. Depuis le project explorer, naviguez jusqu'au répertoire p_name_bsp/microblaze_0/libsrx/mat_mult_v1_0/src. Les fichiers présents correspondent au driver de votre IP : il convient de les éditer afin qu'ils implémentent l'interface logicielle vers votre IP.
2. Créez une fonction *void mat_mult (unsigned int A[][V_SIZE], unsigned int B[][V_SIZE], unsigned int Res[][V_SIZE])* qui envoie les éléments de A[][] et B[][] à votre IP, et récupère le résultat de la multiplication de matrices dans *Res[][]*. Pour ce faire vous pourrez vous inspirer de mat_mult_selftest.c qui utilise :

    - MAT_MULT_mWriteReg (XPAR_MAT_MULT_0_S00_AXI_BASEADDR, …) pour envoyer des éléments vers l'IP
    - MAT_MULT_mReadReg(XPAR_MAT_MULT_0_S00_AXI_BASEADDR,…) pour récupérer des éléments provenant de l'IP.

6.1.1 Complétez le compte rendu

i) Joindre à votre compte rendu la définition de la fonction mat_mult().

j) Copiez la sortie écran montrant des résultats de multiplication identiques pour test 4 et test 5 et montrant le résultat du temps d'exécution dans votre compte rendu. Reportez les résultats dans le fichier Excel, et n'oubliez pas de le joindre au compte rendu.

k) Commentez les résultats contenus dans le fichier Excel pour une multiplication de matrice 6x6.

l) De même que précédemment, nous allons modéliser le temps de réalisation de la multiplication de 2 matrices (n,n) par le système 2 sous la forme $T_{système2}(n)=K_2.n^m$ où m est entier. Au vu de l'implémentation de la multiplication de matrices dans le système 2, proposez une valeur pour m et justifiez là.

m) En utilisant le m fixé à la réponse précédente, déterminez K_2 en utilisant la valeur mesurée pour n=6.

n) Dessinez un graphe présentant $T_{système1}$ et $T_{système2}$ pour n < 100.

o) Sachant sur l'application du client, 80% du temps était dédié aux multiplications de matrices 6x6 sur le système 0, dire quel pourcentage du temps est dédié aux multiplications de matrices 6x6 sur le système 2.
p) Les objectifs de réduction du temps d'exécution cités en introduction ont-ils été atteints ?
q) Commentez la faisabilité du système_2 pour des matrices 7x7 sur votre cible.
r) Décrivez une architecture pour le calcul des matrices 36x36
s) Décrivez une architecture pour le calcul des matrices 100x100
t) Une alternative à l'interface AXI-lite utilisée est l'interface AXI4 (full). Quel impact aurait-elle pu avoir sur les performances ?
u) Une alternative à l'interface AXI-lite utilisée est l'interface AXI4-Stream. Quel impact aurait-elle pu avoir sur les performances ?