# Réalisation des simulations :

Nous avons dû progressivement réaliser plusieurs simulations, sur les différents composants de notre microprocesseur, afin de tester leur bon fonctionnement au fur et à mesure que nous réalisions les autres composants.  
Cependant, en lieu et place de réaliser un fichier de simulation pour chaque composant, nous simulions au fur et à mesure avec les différents composants en pipeline : ainsi, par exemple, nous avions testés le décodeur, pour ensuite ajouter l’ALU à sa suite, la RAM, puis l’unité de contrôle.

Nous avons choisis cette approche afin de simplifier la réalisation du top module, qui consista alors au final à une simple reproduction du test bench, agrémenté de quelques lignes de code que nous expliquerons par la suite.

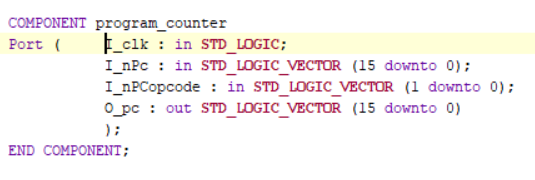
La réalisation des simulations fût aisée en termes de syntaxe VHDL : en effet, nous avions uniquement besoin de réaliser un PORT MAP sur les composants que nous souhaitions tester, créer les différents signaux nécessaires à leur fonctionnement, et à initialiser les composants avec ces signaux.  
Nous avons également dû mettre en place une horloge, simulée par un process à part.

Cependant, la mise en commun des différents composants fût plus compliquée à mettre en place car nécessitant des recherches approfondies sur le flux des instructions dans les différents composants. Nous avons ainsi commencés par tester le bon fonctionnement de la RAM ainsi que le décodeur car ces deux composants composent les portes d’entrée des instructions. Nous avons ensuite dû définir les codes d’ALU dans un fichier de constantes afin de simplifier leur utilisation dans les tests, pour ensuite réaliser l’ALU en lui-même : cette première partie de la réalisation du bench test final fût la plus facile à la réalisation car l’ALU, à l’exception des opérations nécessitants un branching, devait simplement se contenter d’afficher les valeurs de sommes, de soustractions, où le résultat d’un shift.

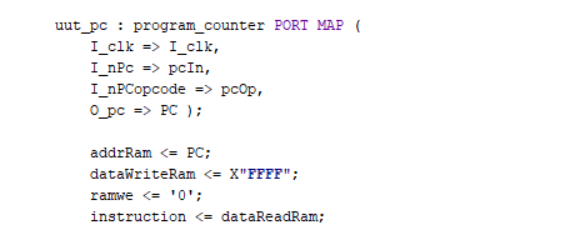
Nous avons ensuite ajoutés l’unité de contrôle, qui permet de synchroniser les exécutions des instructions dans les différents composants : en effet, sans unité de contrôle, nous avions remarqués que lors de l’exécution de plusieurs instructions, nécessitants plusieurs cycles d’horloge, nous retrouvions des résultats erronés, dû à l’exécution de certaines instructions avant celles supposés être exécutés.

L’ajout de l’unité de contrôle au test bench à permis de retrouver en simulation des résultats cohérents, cependant, nous ne simulions pas réellement le travail d’un microprocesseur car nous avions hard-codé toutes les étapes de fonctionnement. Il convenait alors d’implémenter la mémoire de programme qui permettait de stocker les instructions à exécuter sans devoir les programmer dans le test bench.   
Nous avons par la suite implémenté dans ce même test bench le program counter, qui permet de situer l’instruction à exécuter.

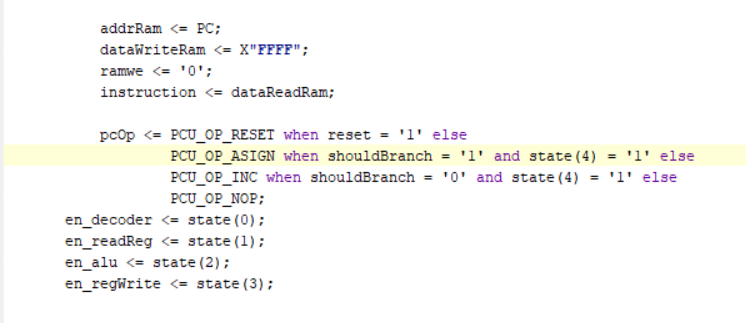
Un exemple de composant mappé, le compteur de programme, est présenté ci-dessous :



Ainsi, l’assignation des signaux est réalisé de la sorte :



Le bench test en lui-même tient dans les lignes suivantes :

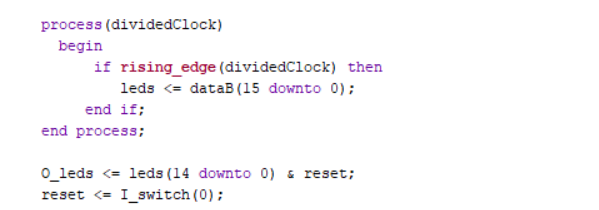


Nous assignons tout d’abord le program counter à l’adresse ram ou est contenu l’instruction à réaliser. Le signal pcOp stocke un bus de donnée conditionnel, afin de se conformer à l’instruction en question. Enfin, la variable state est assignée à l’unité de contrôle et permet d’assurer la synchronisation d’exécution des instructions.

# Réalisation du top module :

Le top module est une simple copie du bench test, en raison de la démarche de réalisation choisie pour ce dernier.   
Cependant, certaines différences sont à noter : nous n’utilisons plus d’horloge simulée, mais l’horloge de la plaque, passée au diviseur de fréquence, afin de pouvoir observer les LEDS.  
Nous choisissons d’utiliser les 16 leds disponibles sur la plaque afin d’observer la valeur finale retournée par le microprocesseur.   
En effet, les 15 premières Leds affichent les 15 premiers bits de sortie, tandis que la dernière Led est associée au reset, qui est activé par le switch.

Le code correspondant est contenu dans les quelques lignes suivantes, en plus du code utilisé dans le testbench :



DividedClock correspond ici à l’horloge en sortie du diviseur de fréquence.