Synthèse logique d'un circuit avec Alliance

L'objectif de ce TME est de vous faire faire un peu de modélisation de circuit. Dans le TP suivant, vous allez pouvoir placer et router le circuit que vous aurez décrit et validé.

Dans ce TME, vous allez travailler sur deux modèles: un PGCD et un CORDIC. Le PGCD va servir d'échauffement pour la synthèse et également un peu de modèle pour la validation de CORDIC. Le code du PGCD présenté dans le cours est presque complet, vous devez le compléter, le tester et en faire la synthèse. Le code de CORDIC est complet, mais vous allez devoir le faire évoluer. Il y a plusieurs degrés d'évolutions possibles. Si vous n'y arrivez pas, vous pouvez quand même expliquer votre démarche. L'idée de ce TME, c'est comme si on vous donnait la version 1.0 d?un circuit et que vous devez fabriquer la version 2.0.

Pour le TP sur le placement-routage, vous routerez la version de base ou votre version. Donc pas d?inquiétude, si vous avez des problèmes :-)

Vous noterez que le automates sont codés en one-hot. Si vous avez déjà codé des automates, ce n?est probablement pas ce qu?on vous a appris. Ce codage est beaucoup plus performant lors de la synthèse, mais il y a un risque lors de l?écriture que votre description ne soit pas déterministe. En effet, la description est incomplète si vous n?avez aucun bit à 1 dans le registre d?état ou non orthogonal si vous avez plus d?un bit à 1 dans le registre d?état. On peut ajouter des `assert` qui vérifie que la somme de tous les bits du registre d?état vaut toujours `1`, mais ici, les `assert` ne sont pas acceptés par `vasy`. C?est donc à vous d?être vigilant. On peut aussi faire la somme des bits dans le VHDL, mais cela prend de la place.

PGCD

- Récupérez l'archive de PGCD.
- Ecrivez un fichier Readme.md donnant une explication succincte (sur une ligne) du rôle de chaque fichier.
- Complétez le fichier `pgcd_core.vhd`
- Validez-le avec le Makefile
- Que devriez-vous faire pour augmenter la précision des nombres ? (Vous pouvez le faire, mais ce n?est pas obligatoire)
- Faites la synthèse sur SXLIB avec BOOM, BOOG et LOON. Vous devez faire évoluer le Makefile et là vous avez deux possibilités:
    1. Vous ajoutez une règle `synthesis:` avec une cible sans dépendance dans laquelle vous mettez la séquence des outils à lancer (boom, boog, loon, asimut , etc.). Cette règle est donc comme un script shell a l?instar des règles déjà présentes dans le Makefile.
    2. Vous décrivez un Makefile dans lequel les règles ont la forme `fichier_produit : liste de fichiers sources`. Chaque règle ne fait qu?une étape de la conception ou de la vérification, une règle pour `boom`, une règle pour `boog`, etc.

       La seconde possibilité offre l?avantage d?être explicite du point de vue des fichiers produits et des fichiers dont ils dépendent et elle permet une reconstruction partielle. Cependant, elle est plus complexe à décrire et finalement pas très utile. En effet, l?intérêt principal du Makefile, c?est de décrire la séquence des outils et leur arguments, ce que fait mieux la première possibilité. En effet, c?est plus compact et donc plus simple à comprendre. J?ai donc une préférence pour la première possibilité.

- Vous devez aussi vérifier la synthèse par simulation, d?où le lancement d?asimut après la synthèse logique.

CORDIC

Fonction

Le circuit réalise la rotation d'un vecteur (x,y) par un angle a et produit le vecteur (nx,ny)

- Le circuit prend en entrée
  - les coordonnées x_p et y_p qui sont des nombres entiers signés de -127 à +127.
  - L'angle a_p est exprimé en radian et il est représenté par un nombre en virgule non signé 3-5. À l'intérieur du circuit, c'est un nombre en virgule fixe 1-8-7. C'est un choix pour réduire le nombre de broches. (Je ferai une remarque la dessus)
- le circuit reçoit aussi une horloge et un signal reset.
- Le circuit produit en sortie les coordonnées (nx_p, ny_p) du vecteur après rotation.
- Le protocole de communication en entrée et en sortie est FIFO.

Questions

- Récuperez l'archive de CORDIC
- Ecrivez un fichier Readme.md donnant une explication succincte (sur une ligne) du rôle de chaque fichier.
- Que fait `make plot`?
- Expliquez quel est le principe de la validation utilisé pour CORDIC et la différence avec celle utilisée pour PGCD.
- Pourquoi celle utilisée pour PGCD est préférable ? (Elle est aussi plus simple)
- faire la synthèse logique sur SXLIB et vérifier le résultat.
- Récrivez `cordic_data` pour qu?il soit semblable à `pgcd_data`.
- Vous allez ensuite faire évoluer le circuit. Vous n?êtes pas obligés de faire tout, cela dépend de votre intérêt et de votre niveau de départ. Notez que vous devez d?abord faire une étude et spécifier sur papier vos changement. Si vous ne faites que la spécification, c?est déjà bien. L?évolution 1 est la plus simple, les évolutions 2 et 3 sont plus complexes, vous pouvez tenter l?une des deux ou les deux.
  1. Réduction du nombre de ports:
     Vous allez faire entrer les arguments `x`, `y` et `a`, l?un après l?autre et faire sortir les résultats également l?un après l?autre. Ainsi vous économisez 24 broches, mais vous perdez en latences puisqu?il faut plusieurs cycles pour entrer les arguments et plusieurs cycles pour sortir les résultats.
  2. Augmentation du débit en créant un pipeline à deux ou trois étages.
     CORDIC est réalisé en plusieurs étapes dont la durée n?est pas identique et d?ailleurs pas fixe non plus pour certaines étapes : lecture des arguments et normalisation de l?angle, calcul, placement des résultats dans le bon quadrant, multiplication par KC et écriture des résultats. Je vous laisse libre de choisir ce que vous faite dans chaque étage.

Rendu

Vous devez écrire le compte-rendu contenant les réponses aux questions, la description de vos travaux et la spécification de vos évolutions. J?ai une préférence pour markdown, mais vous pouvez utiliser LaTeX ou Word.