Chapitre 1

plan

- Introduction
- Objectif :
- Filtre numérique
- Arithmétique virgule fixe
- FiPoGen : diagramme de classe [all-util]
- Le SIF
- l'oSoP
- Génération de la liste d'oSoP
- Stratus et les générateurs utilisés : +, x, reg
- Méthode de création et connexion des composants pour un oSoP
- Connexion des oSoP pour créer le circuit
- simulation du circuit
- Synthèse logique
- Les tests : unitaire, Comparaison du résultat avec LWDF et d'autres filtres, C-VHDL avec GHDL, test du circuit en ghdl, (Simulation Matlab ?)

1.1 sujet et contexte

1.2 Objectifs

objectif – solution

Chapitre 2

Les filtres numériques

Le filtrage est un traitement qui permet de transformer un signal. Les filtres numériques agissent sur des signaux à temps discrets c.a.d une séquence de nombres noté $\{x(n)\}$ ou $x(n)$ est la valeur du signal réel au temps $t = n.T$, T étant la période d'échantillonnage.

FIGURE 2.1 – Un filtre numérique

Un filtre numérique est entièrement définie par une équation au différence liant l'entrée $x(n)$ et la sortie $y(n)$:

$$y(n) = \sum_{i=0}^{N} b_i.x(n-i) - \sum_{i=1}^{N} a_i.y(n-i) \tag{2.1}$$

A partir de l'équation au différence, on obtient, par transformation en Z, la fonction de transfert du filtre :

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{N} b_i.z^{-i}}{1 + \sum_{i=1}^{N} a_i.z^{-i}} \tag{2.2}$$

Un filtre peut être spécifié par sa réponse impulsionnelle, ou son gain en fréquence ou mieux par son gabarit. Notons que, dans le vocabulaire du traitement des signaux, la "synthèse" consiste à trouver la valeur des coefficients

a_n et b_n qui répond à la spécification du filtre.
Dans ce travail, on considère que ce calcul a déjà fait au préalable. Donc, on prend en entrée un filtre déjà définit avec les coefficients exprimés en précision "infinie". On essaiera d'obtenir une description matérielle du filtre.

2.1 Réalisation ou Structure d'un filtre

La manière de faire les calculs dans le filtre définit la structure du filtre. Il existe différente structure d'implémentation d'un filtre, à savoir :

- Forme Directe I
- Forme Directe II : économie de stockage
- Forme Transposée
- Représentation d'État : utilisée en automatique

Si les calculs dans le filtre sont faits de manière exacte, il y a aucune différence sur les structures. Mais en précision finie, le résultat dépend beaucoup de la structure utilisé.

- les coefficients ne sont pas exactes
- les calculs ne sont pas exactes

2.2 La Forme implicite - S.I.F

[Hilaire06] [Benoit - page 27] [FWRUserGuide - page 10] La représentation d'état est très utilisée pour étudier un filtre, mais elle n'est pas complète et a quelques limitations :

- difficile d'analyser l'effet d'arrondi sur un coefficient particulier
- Absence de variable intermédiaire utile pour l'expression de certaines réalisations

Avantages :

- analyse facile des effets de la précision finie
- description de haut niveau plus généralisé et plus précis

Chapitre 3

l'arithmétique virgule fixe

Chapitre 4

FiPoGen

Re-Modélisation d'une partie de FiPoGen

[Avantages :

- Cela organisera la compréhension du système et facilite l'intégration d'un nouveau développeur.
- Permet une exploration d'autres alternatives
- facilite le développement et la maintenance du système
- facilite la réutilisation]

4.1 Présentation de FiPoGen

FiPoGen est un outil développé dans le cadre de la thèse de Benoit Lopez et du projet ANR DEFIS (Design of fixed-point embedded systems, 2011--2014) permettant de transformer un algorithme en un graphe d'opérations virgule fixe, avec calcul des erreurs commises, optimisations des largeurs des opérateurs (sous contrainte d'erreur de sortie). Dans sa version actuelle, FiPoGen est principalement dédié à l'implantation des filtres linéaires récursifs.

4.2 Les problèmes Liés à la précision Finie

[FWRUserGuide.pdf][Thèse Benoit] Les filtres numériques sont toujours implémentés dans une précision finie car les calculateurs ont un nombre de bits limités pour représenter les nombres et faire les calculs.
Cela introduit 2 effets :

- le bruit d'arrondi sur les variables - round-off noise
- la dégradation de performance due à l'arrondi des coefficients - coefficients sensitivity

Il est donc nécessaire de s'assurer que ces effets n'entraine pas une dégradation considérable sur la performance du filtre.
Ces effets dépendent du :

- format d'arithmétique choisi (FIP ou FxP)
- la longueur de mots pour représenter les nombres
- la réalisation choisie (structure du filtre)

4.3 L'architecture de FiPoGen

Chapitre 5

Stratus

Stratus peut être vue comme une bibliothèque de Générateur de composants paramétrable, qui génère à la sortie du code VHDL synthétisable.

Notons que les filtres linéaires n'utilisent que 3 opérateurs, à savoir :

- l'addition
- la multiplication par une constante
- le délai

Ceux-là correspond respectivement dans le circuit à des additionneurs, multiplicateurs par une constante et des registres.

Documentation des générateurs de Stratus utilisés dans le travail de stage.

5.1 Les configurations de Stratus

Chapitre 6

Génération de la Forme Implicite Spécialisé à partir du diagramme de Blocs Simulink

6.1 L'idée

L'objectif ici est de déduire les 9 matrices (J, K, L, M, N, P, Q, R, S) du S.I.F à partir d'un schéma-blocs représentant un système linéaire, de manière automatique.

6.2 Le diagramme de bloc Matlab/Simulink

Un diagramme de bloc ou schéma-bloc est une représentation graphique simplifié d'un système plus ou moins complexe. Il est composé de blocs connectés par des lignes qui montre l'interaction entre-eux. Il permet d'exprimer la structure et le flux d'un système pour juste montrer le concept sans entrer dans les détails d'implémentation.

[exemple de schéma-bloc]

Dans Simulink, on a 2 formats pour stocker les schéma-blocs : MDL : ancienne format SLX : nouvelle format de fichier, à partir de la r2012a de Matlab

Av : C'est un standard dans l'industrie. Pb : C'est un nouveau format très peu documenté.

6.3 Méthode pour déduire le SIF à partir d'un Diagramme de Blocs

[faire la liste en diagramme] - Décompresser le fichier SLX pour extraire le fichier *blocdiagram.xml* - Parser le fichier pour extraire la liste des blocs et la liste des lignes (qui indiques les inter-connexions) - Aplatir le design s'il y a des sous-systèmes - Identifier les blocs en entrée de chaque bloc, cela donne l'équation au niveau de chaque bloc - regrouper si possible les blocs sommes - Identifier les équations avec celles du SIF et donc les coefficients formeront les matrices du SIF - Rendre triangulaire inférieure la matrice J et réorganiser les matrices concernées : M, N, K, L

Explications : Ce qu'on a :

– Chaque bloc simulink est identifié par un numéro unique appelé ***SID***

Ce que l'on met comme hypothèse :

– chaque bloc est labellisé par t, x, u, y suivit du SID [example]

6.4 Implémentation - le module SLX2SIF

Il suffit de lui donner un fichier Simulink au format SLX, contenant le schéma-bloc pour un système linéaire, et il donne en sortie les matrices du SIF avec les variables liées (t, x, u, y).

[schéma]

6.4.1 Les caractéristiques du module

Pour l'état actuel de notre module, on peut traiter 6 types de blocs Simulinnk :

– Gain
– Delay
– Sum
– SubSystem
– Inport
– Outport

[tableau : blocs - attributs]

Il peut traiter : - des systèmes MIMO - les sous-systèmes

Chapitre 7

Génération de la liste des oSoP à partir du SIF

La forme implicite spécialisée aka S.I.F [1] permet de représenter efficacement un filtre linéaire. Pour l'implémenter réellement, il faut choisir la nature de la cible : logicielle ou matérielle. [Thèse Benoit]

Dans le cas de l'implémentation logicielle, la taille des variables, constantes et opérateurs est fixée par la cible. Par contre, pour l'implémentation matérielle, les largeurs peuvent varier selon la plage des valeurs que peuvent contenir chaque variable ainsi que la précision voulue. C'est ce qu'on appelle le paradigme des largeurs multiples. (Voir Benoit page 146 : Optimisation des largeurs) [Voir aussi page 166 : implémentation HW]

7.1 La représentation en Espace d'État équivalent au SIF

7.1.1 la représentation d'état

Il y a plusieurs manière de décrire un système dynamique. La représentation d'état est une modélisation sous forme matricielle en utilisant des variables d'état... Pour notre cas, on a une représentation discrète et linéaire. [Equation + légende + petite explication]

1. Specialized Implicite Form

7.1.2 Déduction du State-Space équivalente

[Equation du SIF + nom des équations]

$$\begin{aligned} J.t(k+1) = & \quad M.x(k) + N.u(k) \\ x(k+1) = & \; K.t(k+1) + P.x(k) + Q.u(k) \\ y(k) = & \; L.t(k+1) + R.x(k) + S.u(k) \end{aligned}$$

Le principe consiste à intégrer les calculs intermédiaires dans l'équation d'état et de sortie.

$$\begin{aligned} [calculs] & \\ A &= K.J^{-1}.M + P \\ B &= K.J^{-1}.N + Q \\ C &= L.J^{-1}.M + R \\ D &= L.J^{-1}.N + S \end{aligned}$$

7.2 Calcul de Hu

La fonction de transfert Hu sert à évaluer quel effet a l'entrée sur les variables .

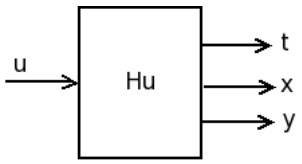

FIGURE 7.1 – La fonction de transfert Hu

Pour le calculer, on part toujours du S.I.F qui décrit notre système.
[Equations]

7.3 Calcul de He

He sert à voir l'effet des erreurs accumulées par les variables sur la sortie.

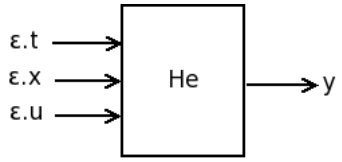

FIGURE 7.2 – La fonction de transfert He

7.4 Calcul de DC-Gain pour un State-Space

Le DC-G est le gain en mode statique ou la fréquence du signal est nulle. Le gain DC-G en temps discret est la valeur de la fonction de transfert lorsque $z = 1$
Pour les modèles en représentation d'états avec les matrices (A, B, C, D) :
$G_{DC} = D + C.(I - A)^{-1}.B$

7.5 Calcul du WCPG - Worst Case Peak Gain - pour un State-Space

Le W.C.P.G est la plus grande valeur de la sortie pour toute valeur possible de l'entrée. Pour un système modélisé dans la représentation d'états, on a le définit comme suit :

$$WCPG = |D| + \sum_{k=0}^{\infty} |C.A^k.B|$$

Chapitre 8

Connexion entre les oSoP

Un fois qu'on a le code VHDL correspondant à chaque somme de produits (oSoP) dans le filtre, on peut les connecter entre eux avec des registres à l'aides des équations d'états, pour avoir le circuit entier qui correspond au filtre.

8.1 Synthèse des codes VHDL

(eval /soc/coriolis2/etc/coriolis2/coriolisEnv.py –root= ravoson/coriolis/)

Chapitre 9

Résultats

Les résultats :

- temps d'exécution de l'outil (noos 2 modules, fipogen, stratus, ampl/bonmin) [benchmark python]
- Erreur su les résultats finaux

Méthodes :

- Supporter avec des chiffres (quantitatives et qualitatives)

9.1 Environnement de développement et Conditions d'expérimentation

- PC avec intel Core 2 Duo E7500 - 2,93Ghz / 2Go de mémoire tournant sous Gnu/Linux 2.6 32 bits
- Python 2.7 (les packages pythons)

Chapitre 10

Liste des choses à faire

- un cas d'utilisation : voir aussi exemple [Benoit page 158]
- les taches d'un développeur "manuel" ?
-
- Présentation de l'arithmétique virgule fixe (un petit peu sur l'arithmétique à virgule flottant)
- les limites de FiPoGen
- filtre LWDF
- les tests
- Contribution par rapport à l'existant au LIP6
- bench - (graphe) . résultat synthèse . temps optimisation ampl/bonmin
- simulation du circuit avec GHDL
- synthèse avec Synopsis
- les noms de variables dans le SIF ? !

Dans la présentation :

- parler un peu plus de la contribution

Chapitre 11

Conclusion Générale

Apport du travail de stage :

- Mise à l'épreuve de FiPoGen et correction de quelques bugs, surtout sur la manipulation des SIF et oSoP
- Ajout d'import Simulink dans FiPogen : le module slx2sif
- Développement d'un module de génération de circuit à partir d'un oSoP avec Stratus
- Création d'un preuve de concept pour un outil de synthèse de filtre numérique : de la modélisation en diagramme de blocs vers la description matérielle
- documentation de FiPoGen

Perspectives :

- (Améliorer le temps d'optimisation des largeurs)
- Développer FiPoGen pour traiter complètement les systèmes MIMO
- Etendre le module d'import Simulink pour supporter plus de type de blocs
- Créer un module inverse SIF vers Simulink