Synthèse de Circuits utilisant l'Arithmétique Virgule Fixe

Soutenane de Master SESI

Maminionja Ravoson

Roselyne Chotin-Avot / Thibault Hilaire

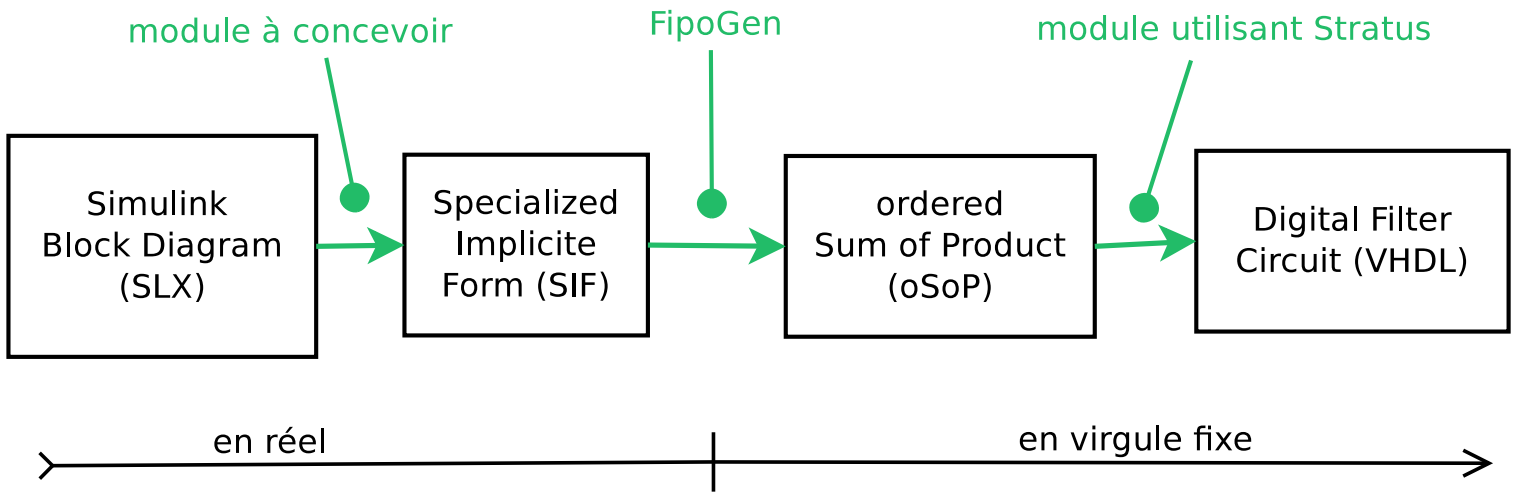
29 Septembre 2015

Contexte du stage

- Le projet **FxPSynthesis** - Action initiative du LIP6
  - Collaboration entre 2 équipes du LIP6 : CIAN et PEQUAN
  - Développement d'un **outil commun** pour la conception de circuits
  - **Un flot complet** - accélérer le développement d'applications
- Les participants:
  - **CIAN** (Circuits Intégrés Analogiques et Numériques)
    Dpt Système sur Puce
  - **PEQUAN** (Performance et Qualité des Algorithmes Numériques)
    Dpt Calcul Scientifique

Du filtre au circuit

- **Objectif**: avoir un outil de synthèse de haut-niveau pour les filtres linéaires, en utilisant l'arithmétique virgule fixe



FiPoGen : Fixed Point code Generator

Stratus : Générateurs paramétrables de composants matériels

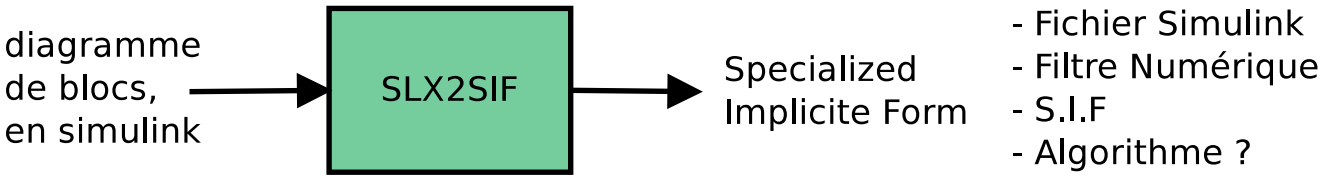
L'existant au LIP6

À notre disposition, on a:

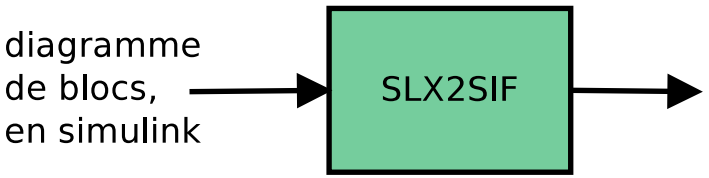
- **FiPoGen**: un outil qui permet de transformer un algorithme de haut niveau en du code virgule fixe déjà spécifié.
  - développé dans l'équipe PEQUAN
  - Entrée : la forme implicite spécialisée - S.I.F
  - Sortie : l'oSoP - graphe de calcul en virgule fixe

- **Stratus**: ensemble de générateurs paramétrables et méthodes qui permet de décrire des composants matériels de manière procédurale.
  - développé dans l'équipe CIAN
  - en Python
  - sortie : VHDL

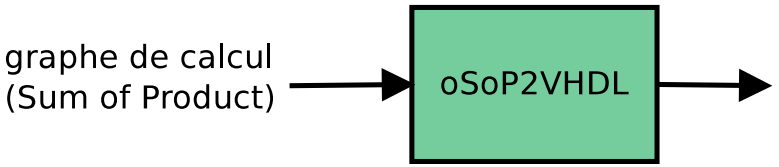
La problématique



La problématique

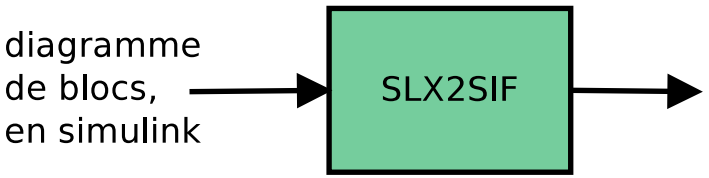


- Fichier Simulink
- Filtre Numérique
- S.I.F
- Algorithme ?

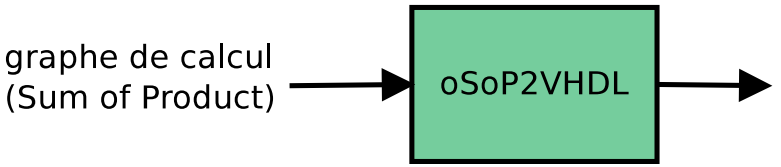


- Arithmétique FxP
- FiPoGen & oSoP
- Generateurs Stratus
- Validation du VHDL ?

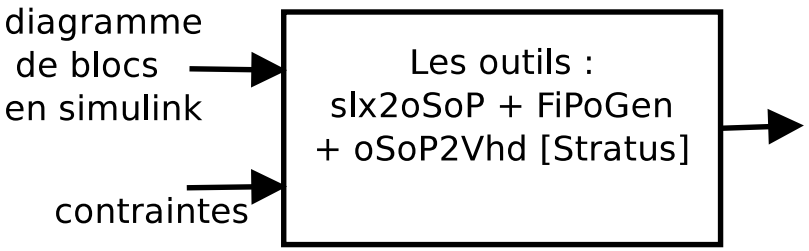
La problématique



- Fichier Simulink
- Filtre Numérique
- S.I.F
- Algorithme ?



- Arithmétique FxP
- FiPoGen & oSoP
- Generateurs Stratus
- Validation du VHDL ?



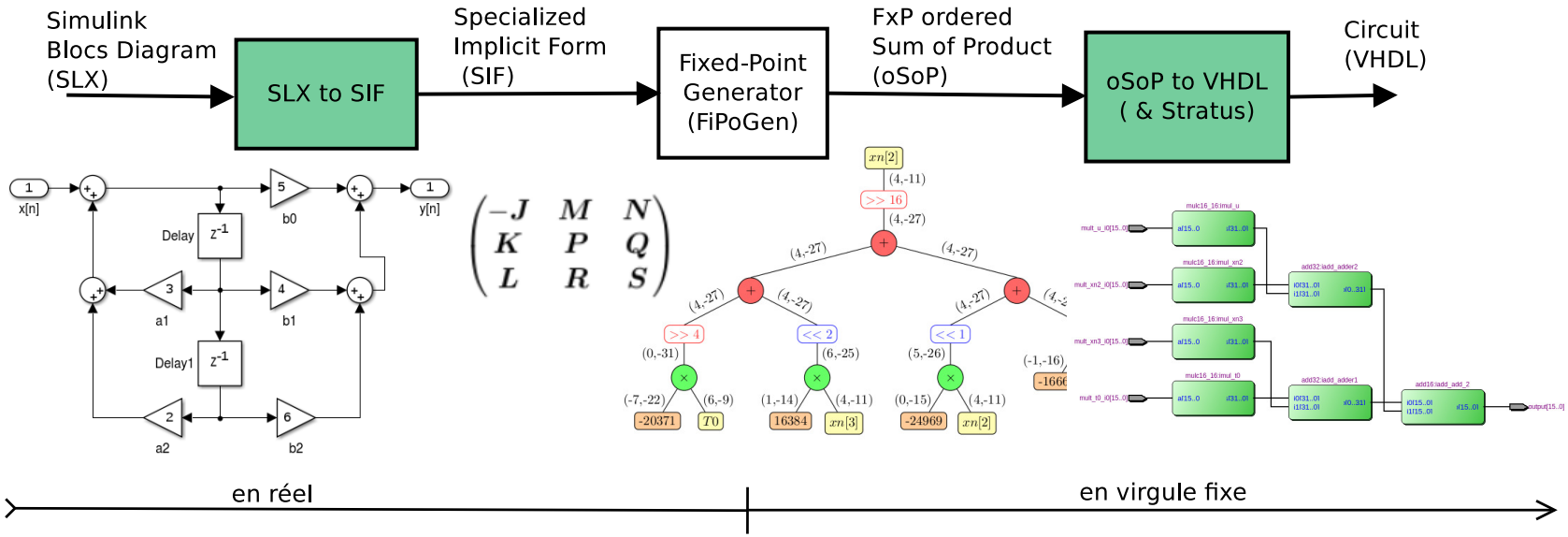
- Intégration
- Flot automatique
- code réutilisable

Environnement de dev et d'expérimentation

- Gnu/**Linux** 2.6
- **Python 2.7** avec zipfile, lxml.etree, math, scipy.io, numpy, pickle, shutil
- FiPoGen, AMPL/Bonmin
- Stratus (lib arith, modules stratus, packages stratus)
- **Git** pour la gestion de version
- Matlab/Simulink
- GHDL pour la simulation VHDL
- Quartus pour voir la NetList RTL

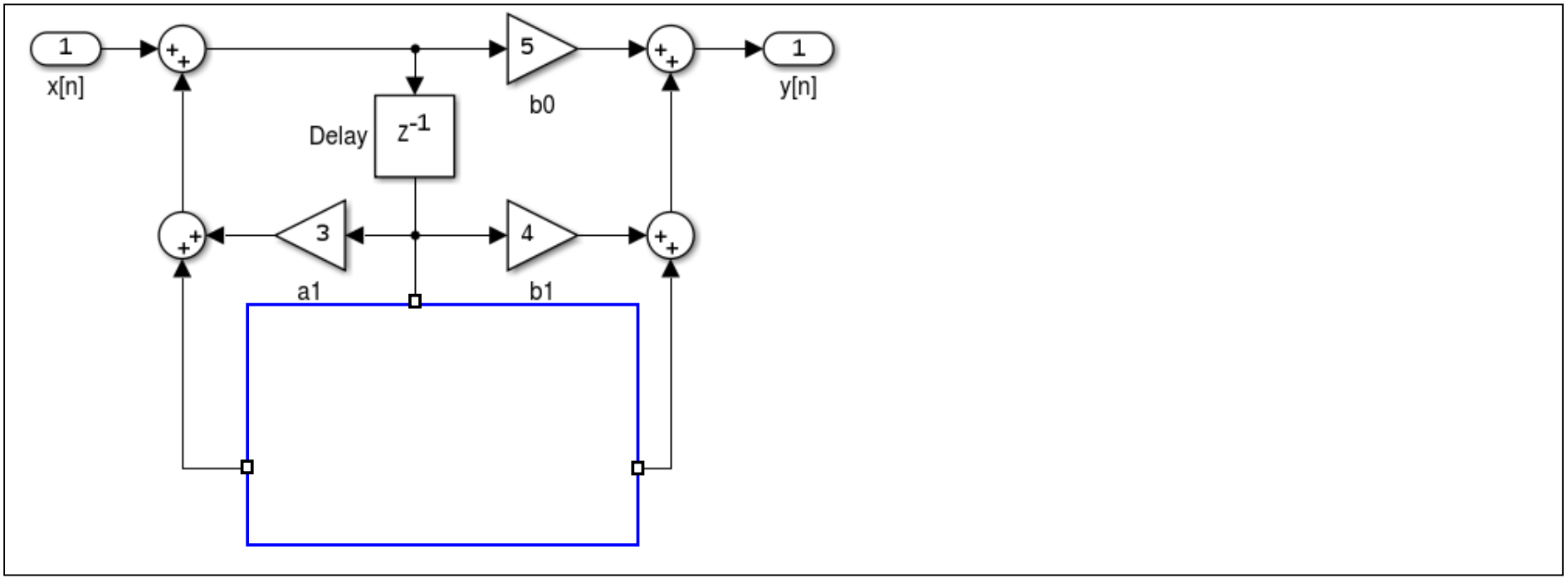
Le flot à développer

- Schéma du flot à développer

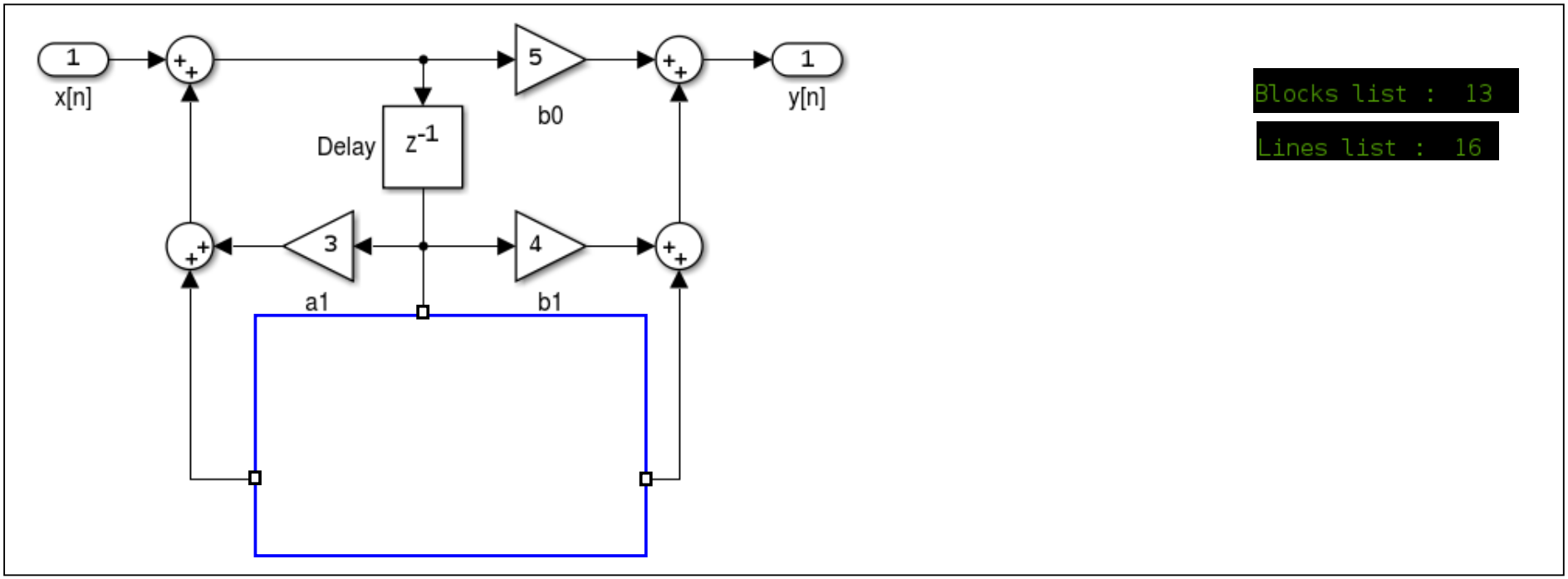


- **les tâches** spécifiques au stage:
  - Concevoir le **module "SLX to SIF"**
  - Développer le **module "oSoP to VHDL"** à l'aide de Stratus
  - **Intégrer** ces modules avec FiPoGen pour former un outil de conception commun entre les 2 équipes

Développement du module SLX2SIF

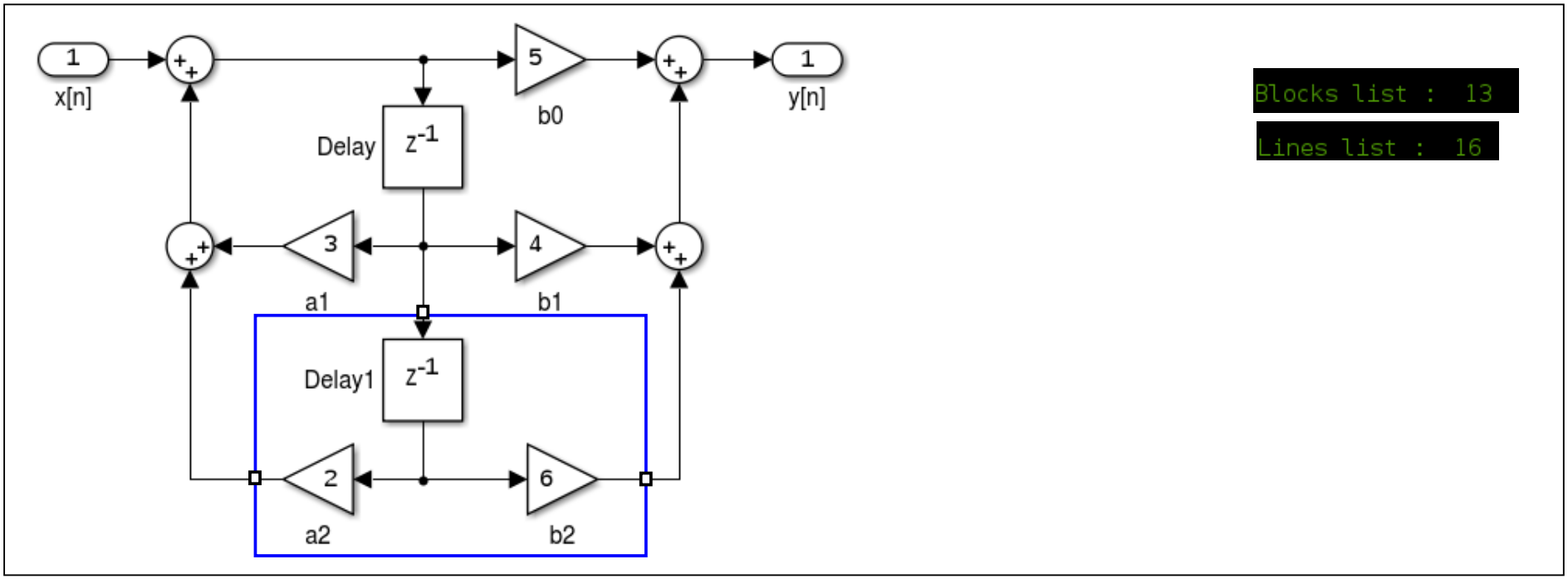


Développement du module SLX2SIF



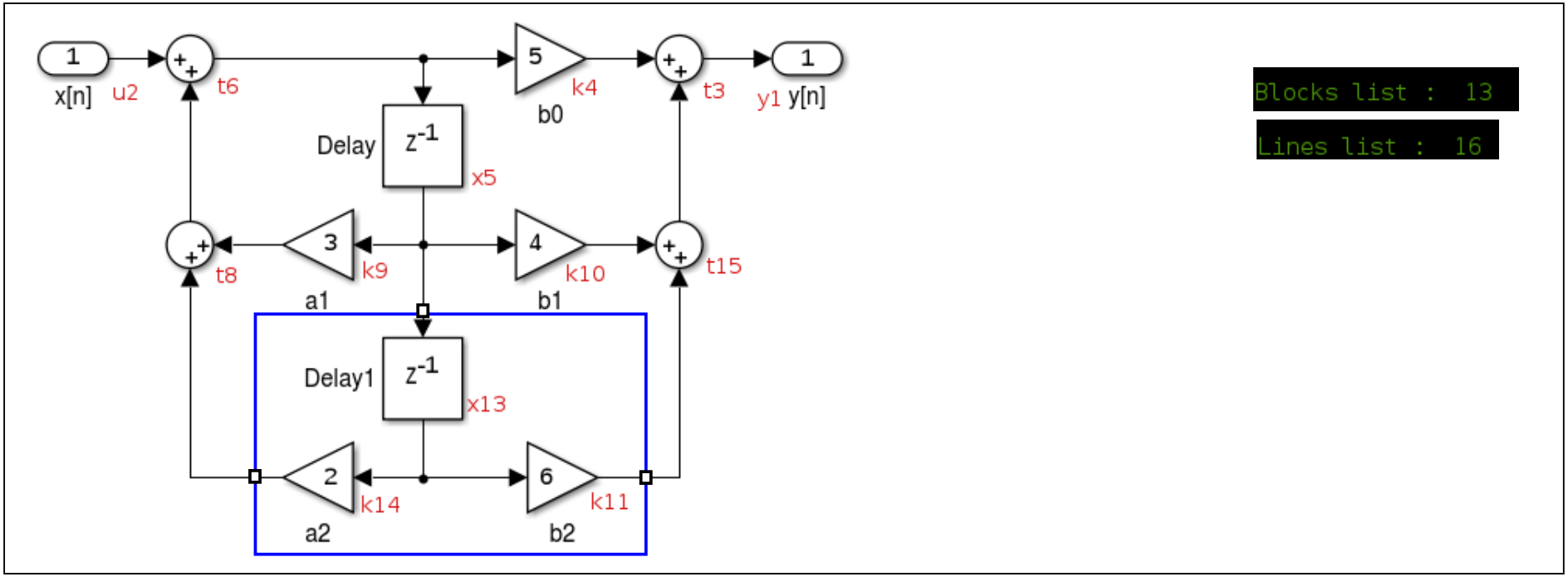
- Extraire la liste des blocs et la liste des lignes

Développement du module SLX2SIF



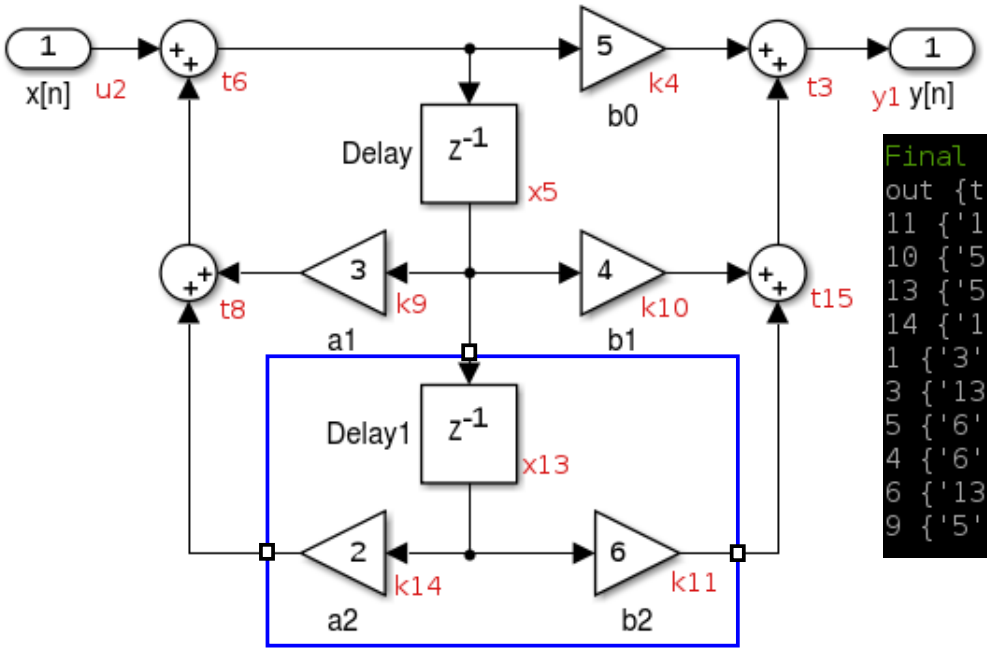
- Extraire la liste des blocs et la liste des lignes
- Aplatir le design s'il y a des sous-systèmes

Développement du module SLX2SIF



- Extraire la liste des blocs et la liste des lignes
- Aplatir le design s'il y a des sous-systèmes
- Labelliser chaque blocs

Développement du module SLX2SIF

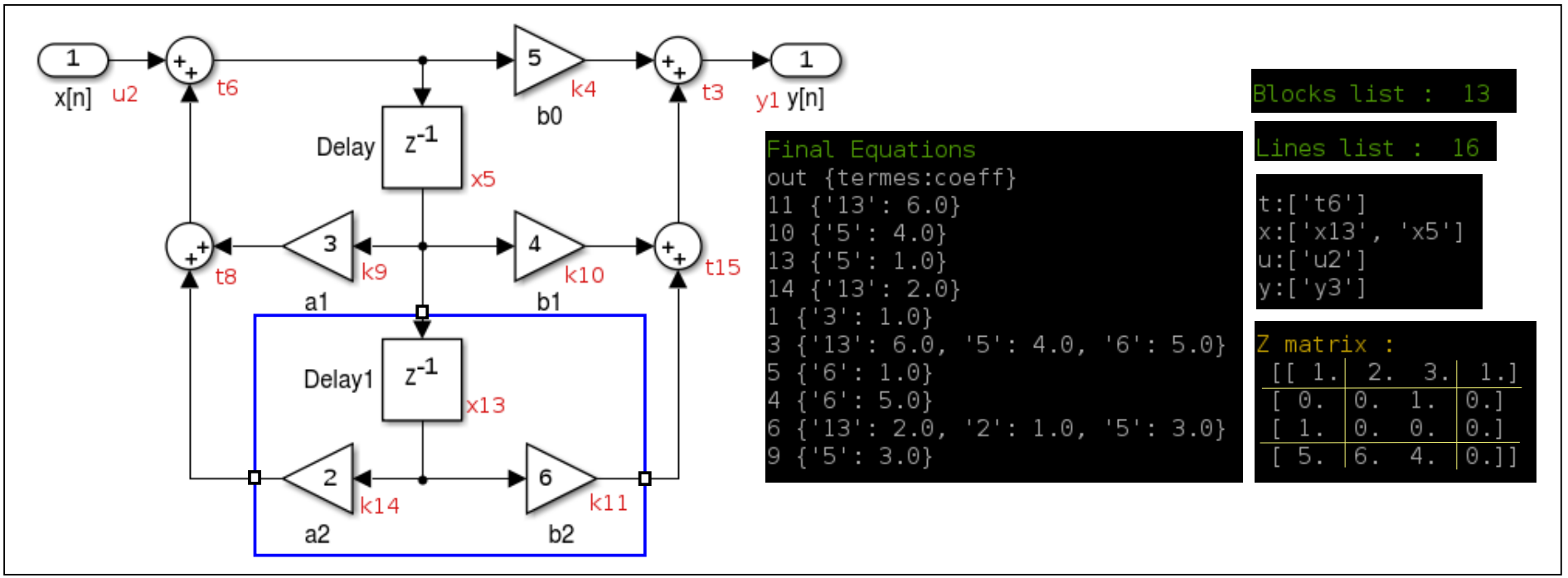



```
Blocks list :  13
Lines list :  16
```

```
Final Equations
out {termes:coeff}
11 {'13': 6.0}
10 {'5': 4.0}
13 {'5': 1.0}
14 {'13': 2.0}
1 {'3': 1.0}
3 {'13': 6.0, '5': 4.0, '6': 5.0}
5 {'6': 1.0}
4 {'6': 5.0}
6 {'13': 2.0, '2': 1.0, '5': 3.0}
9 {'5': 3.0}
```

- Extraire la liste des blocs et la liste des lignes
- Aplatir le design s'il y a des sous-systèmes
- Labelliser chaque blocs
- Établir l'équation au niveau de chaque bloc, regrouper les sommes

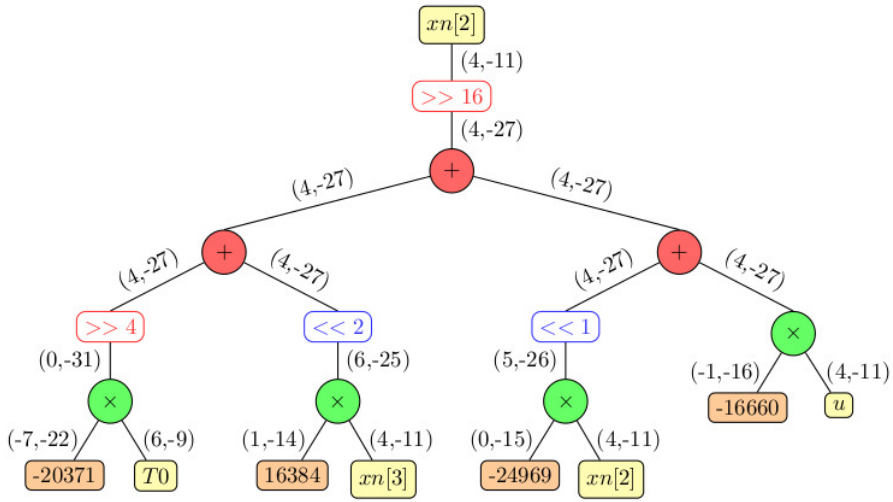
Développement du module SLX2SIF

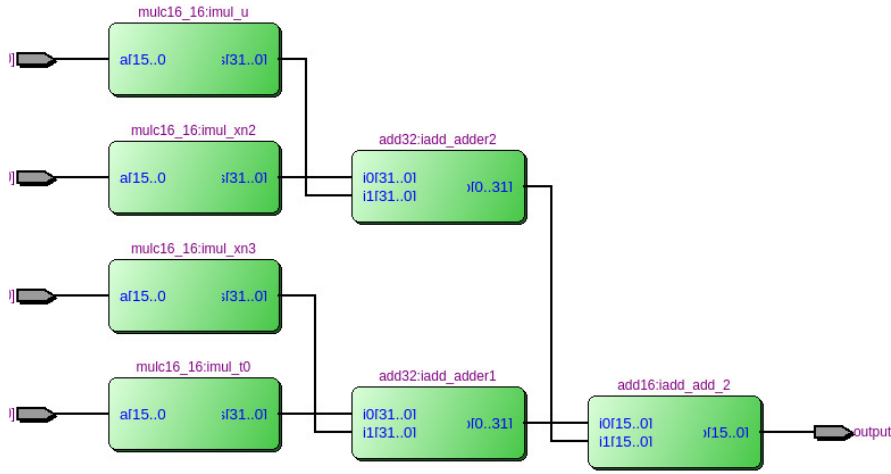


- Extraire la liste des blocs et la liste des lignes
- Aplatir le design s'il y a des sous-systèmes
- Labelliser chaque blocs
- Établir l'équation au niveau de chaque bloc, regrouper les sommes
- Déduire le SIF, rendre la matrice J triangulaire

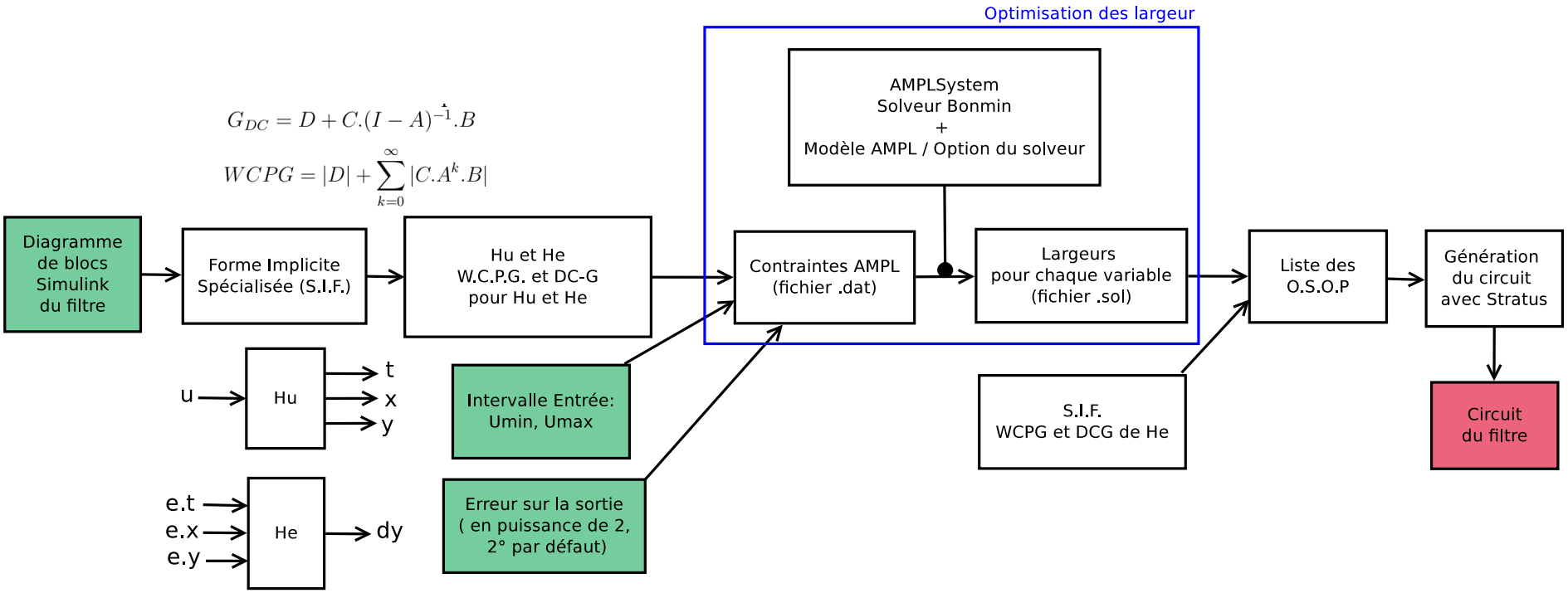
Développement du module oSoP2VHD

- **Principes :** Parcourir l'oSoP et instancier le générateur Stratus correspondant à chaque opérateur



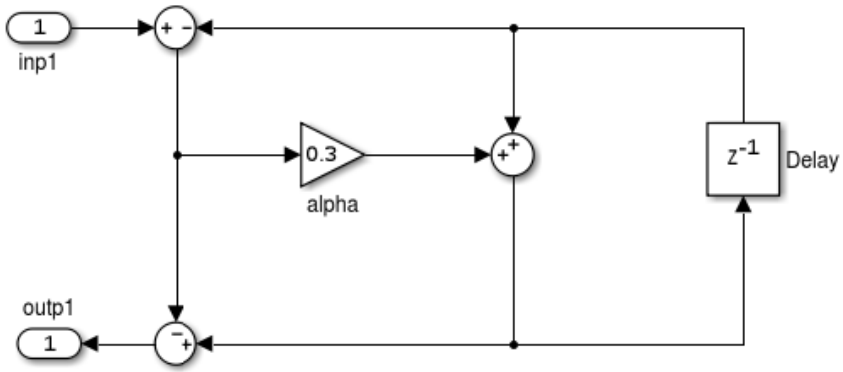


Intégration : le flot complet



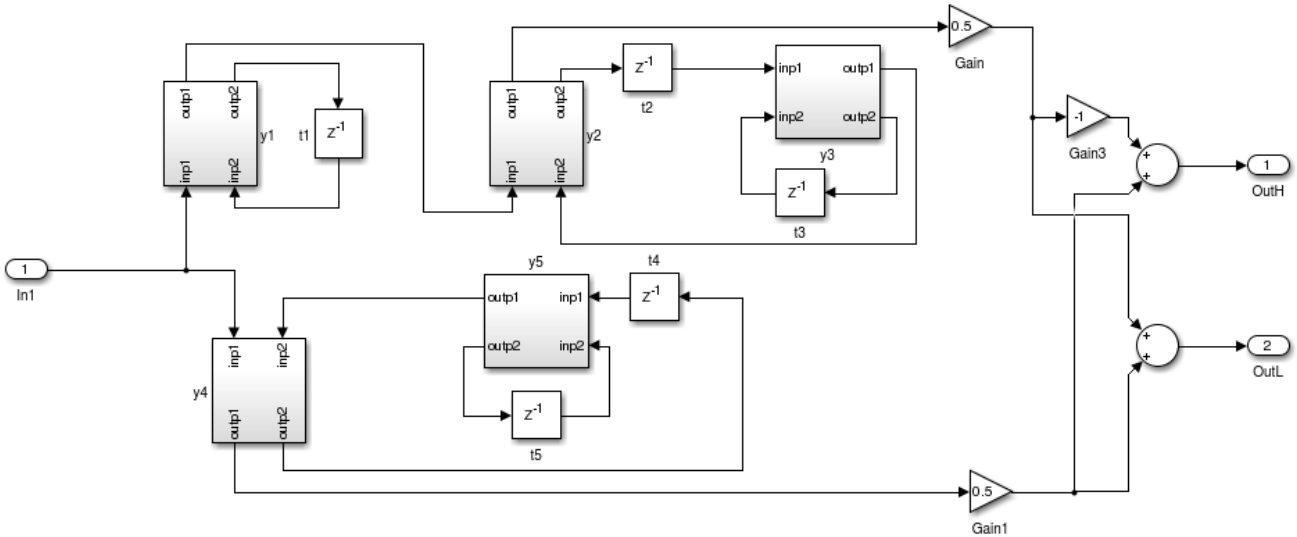
Validation du module SLX2SIF

- Validation sur les 2 types de structures de d'adaptateur pour LWDF




```
J:2x2          M:2x1    N:2x1
[1.0, 0.0]     [-1.0]   [1.0]
['-0.3', 1.0]  [1.0]    [0.0]

K:1x2          P:1x1    Q:1x1
[0.0, 1.0]     [0.0]    [0.0]

L:1x2          R:1x1    S:1x1
[-1.0, 1.0]    [0.0]    [0.0]
```

- Validation sur un exemple de LWDF de 5è ordre (EUSIPCO15)



LWDF : Lattice Wave Digital Filter

Validation du module SLX2SIF

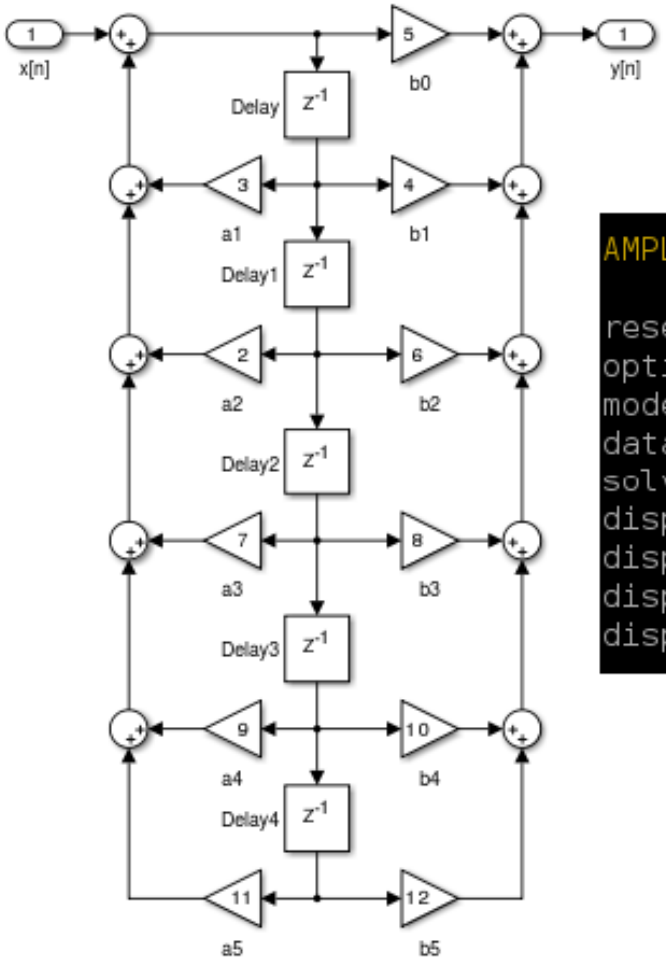
Les fonctionnalités internes vérifiés sont :

- Récupération des paramètres des blocs
  - Gain (Gain)
  - DelayLength (Delay)
  - Inputs (Sum)
- Traitement des sous-systèmes (SubSystem)
- Groupement des additions
- Établissement de l'équation pour chaque bloc

On a les mêmes matrices à la sortie

Validation de la génération de la liste oSoP

On a **automatisé** la génération de la liste d'oSoP



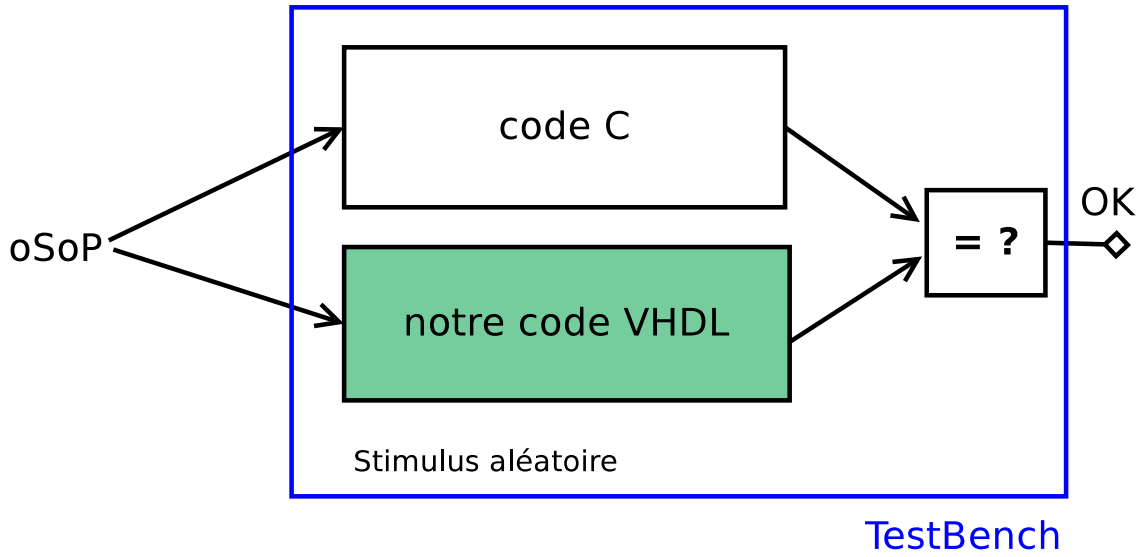

```
Blocks list :  28
Lines list :  37
```

```
AMPL commands :

reset;
option solver bonmin;
model expinlk.mod;
data direct_form2_5_constrains.dat;
solve;
display _ampl_time > width.sol;
display _total_solve_time > width.sol;
display Wx > width.sol;
display Wy > width.sol;
```

```
Z matrix :
 [[  1.  11.   9.   3.   2.   7.   1.]
 [  0.   0.   1.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   1.   0.]
 [  1.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   1.   0.   0.   0.]
 [  0.   0.   0.   0.   1.   0.   0.]
 [  5.  12.  10.   4.   6.   8.   0.]]
```

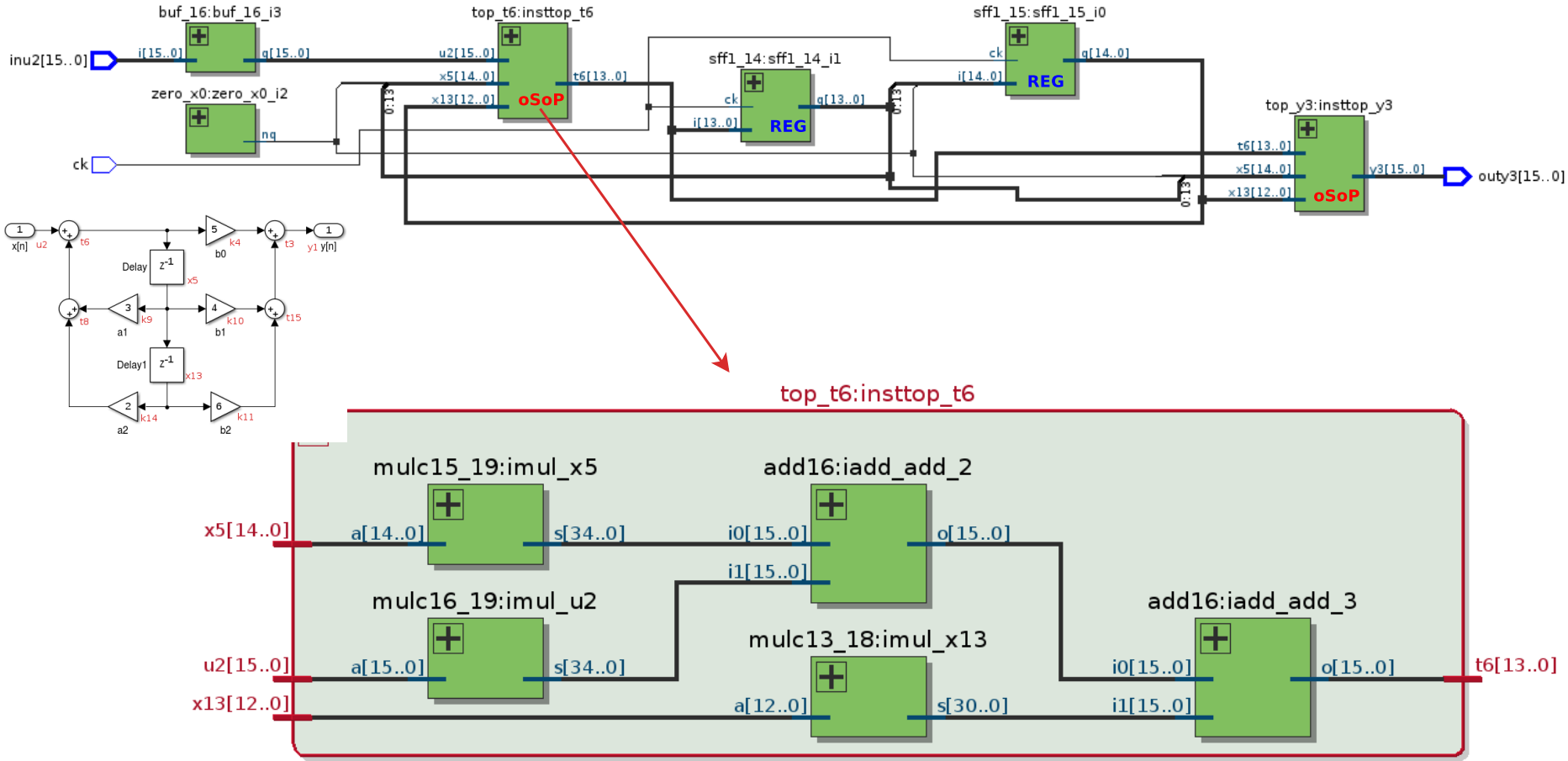
```
Do variables width optimisation
ampl_time = 0.002999
total_solve_time = 3.44347

Optimization saved to :  width.sol

Generate list of oSoP
Var_W =  [15, 11, 12, 14, 11, 12, 17]
```

Validation du module SIF2VHDL



Les 2 codes génèrent les même sorties

Exemple de génération de circuit



Validation visuelle du circuit

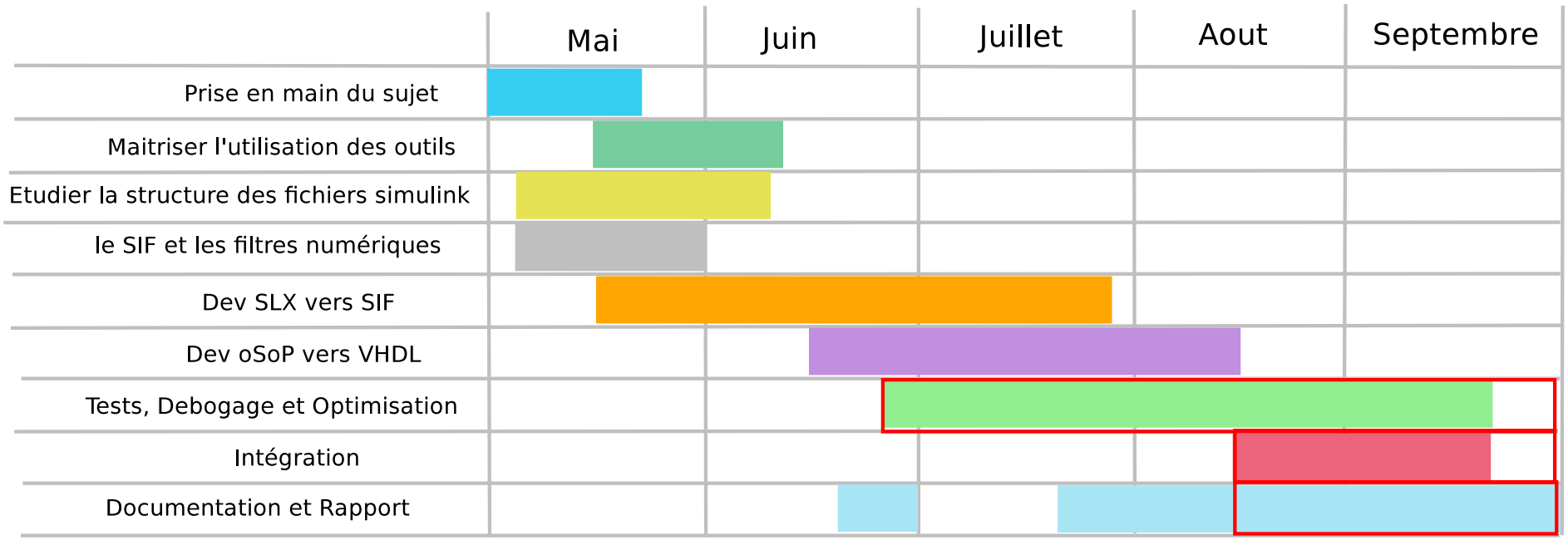
Analyse des Résultats

Fonctionnalités	**Validation**
Comparaison à un exemple fait manuellement	OK
Test sur différents exemples de petites structures	OK
Test sur un exemple avec plus de 80 blocs	OK
Test sur le LWDF	OK
SLX2SIF	OK
Support des systèmes MIMO	OK
Génération d'oSoP	OK
Génération automatique de la liste des oSoPs	OK
Le module oSoP2VHDL	OK
Génération automatique de tout le circuit	OK
Vérification visuel des netlists	OK
La simulation du circuit entier généré	NOK

Remarque :

- C'est l'optimisation des largeurs qui prends beaucoup plus de temps
- Des cas ou la largeur calculé vaut 0 poseront des problèmes !

Calendrier


Mai
Juin
Juillet
Aout
Septembre
Prise en main du sujet
Maitriser l'utilisation des outils
Etudier la structure des fichiers simulink
le SIF et les filtres numériques
Dev SLX vers SIF
Dev oSoP vers VHDL
Tests, Debogage et Optimisation
Intégration
Documentation et Rapport

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 : module **oSoP2VHD**
- Un **flot automatique** : 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**
- **Expérience** en Python, en Filtre numérique, en FxP

Perspectives

- Améliorer le temps d'**optimisation des largeurs**, Essayer d'autres solveurs
- Développer FiPoGen pour traiter complètement les **systèmes MIMO**
- Étendre le module d'import Simulink pour supporter **plus de type de blocs**
- Créer un module inverse **SIF vers Simulink**
- Validation du circuit entier au niveau VHDL
- Gestion plus fine des erreurs

Merci beaucoup

Merci de votre aimable attention !