TP5 : Dessin de cellules précaractérisées

Introduction :

Le but de ce TP est le dessin sous **GRAAL** d'une cellule inverseuse, d'une porte Nand à 2 entrées et d'une porte And à 2 entrées.

Les notions de cellules précaractérisées, de gabarit et de hiérarchie de cellules seront introduites.

Introduction

Dans les TP précédents nous avons utilisé des cellules d'une bibliothèque. Cette bibliothèque peut être enrichie de nouvelles cellules grâce à l'éditeur **GRAAL**.

GRAAL est un éditeur de layout symbolique intégrant le vérificateur de règles de dessin **DRUC**.

Cette séance a pour objectif de dessiner des cellule en tenant compte des règles de dessin fournies.

Il s'agit tout d'abord de la cellule inverseuse **inv_x1** sous la forme d'une cellule précaractérisée de la bibliothèque **sxlib** en respectant les règles de dessin fournies, puis le Nand2, et enfin le And2.

2 Environnement technologique

Certains outils utilisent un environnement technologique particulier. Il est désigné la variable d'environnement **RDS_TECHNO_NAME** qui doit être positionnée à *opt/alliance/etc/cmos.rds* :

```
> export RDS_TECHNO_NAME=/opt/alliance/etc/cmos.rds
```

2.1 GRAAL

L'éditeur de layout **GRAAL** manipule plusieurs types d'objets différents que l'on peut créer avec le menu **CREATE** :

- les instances (importation de cellules physiques),
- les boîtes d'aboutement qui définissent les limites de la cellule,
- les segments : DiffN, DiffP, Poly, Alu1, Alu2 ...
- le CAluX est utilisé pour désigner une portion possible pour les connecteurs,

- les VIAs ou contacts :ContDiffN, ContDiffP, ContPoly et Via Metal1/Metal2,
- les Big VIAs,
- les transistors : NMOS ou PMOS.

GRAAL utilise la variable d'environnement **GRAAL_TECHNO_NAME**. Elle doit être positionnée à */opt/alliance/etc/cmos.graal* :

```
> export GRAAL_TECHNO_NAME=/opt/alliance/etc/cmos.graal
```

2.2 COUGAR

L'outil **COUGAR** est capable d'extraire la netlist d'un circuit aux formats **.vst** ou **.al** à partir d'une description au format **.ap**.

Pour extraire au niveau transistor, la commande à utiliser est :

```
> cougar -t file1 file2
```

COUGAR utilise les variables d'environnement **MBK_IN_PH** et **MBK_OUT_LO** suivant les formats d'entrée et de sortie. Par exemple pour générer une netlist au format **.al** à partir d'une description **.ap** il faut écrire :

```
> export MBK_IN_PH=ap
> export MBK_OUT_LO=al
> cougar -t file1 file2
```

2.3 YAGLE

l'outil **YAGLE** est capable d'extraire la description VHDL comportementale d'uncircuit au format **.vhd** à partir d'une *netlist* au format **.al** *si celle-ci est au niveau transistor*. L'outil **VASY** permet de convertir une description VHDL comportementale du format **.vhd** au format **.vbe**. La commande à utiliser est :

```
> export MBK_IN_LO=al
> export YAGLE_BEH_FORMAT=vbe
> yagle -s file1 file2
> vasy -a -I vhd file1 file2
```

Avant tout vous devez utiliser la commande :

```
> source avt_env.sh
```

Cette commande permet de mettre en place l'environnement nécessaire à l'utilisation de **YAGLE** (le fichier avt_env.sh étant fourni) Les documentations pour cet outil se trouvent en :**/users/soft/AvtTools2003/doc** .

2.4 PROOF

Lorsqu'on veut prouver l'équivalence de deux descriptions comportementales de type *dataflow* d'un même circuit à n entrées, on peut simuler par **asimut** des vecteurs pour les deux descriptions et les comparer. Cette solution devient vite coûteuse en temps CPU et il vaut mieux faire appel à un outil de preuve formelle qui effectue la comparaison *mathématique* des deux réseaux booléens. **PROOF** réalise cette opération entre les description file1.vbe et file2.vbe par la commande :

```
> proof file1 file2
```

3 Le gabarit sxlib

- Les cellules de la bibliothèque **sxlib** ont toutes une hauteur de 50 lambdas et une largeur multiple de 5 lambda.
- Les alimentations Vdd et Vss sont réalisées en Calu1 ; elles ont une largeur de 6 lambdas et sont placées horizontalement en haut et en bas de la cellule.
- Les transistors P sont placés près du rail Vdd tandis que les transistors N sont placés près du rail Vss.
- Le caisson N doit avoir une hauteur de 24 lambdas :
- Les segments spéciaux CAluX (CAlu1, Calu2, CAlu3...) forment l'interface de la cellule et jouent le rôle de connecteurs "étalés". Ils doivent obligatoirement être placés sur une grille de 5x5 lambdas et peuvent se trouver n'importe où à l'intérieur de la cellule.
- Les segments spéciaux TAlux (TAlu1, TAlu2, ...) servent à désigner les obstacles au routeur Lorsque vous voulez protéger des segments AluX, il faut les recouvrir ou les entourer de TAlux correspondant (même couche). Les TAluX sont placés sur une grille au pas de 5 lambdas.
- La largeur minimale de CAlu1 est de 2 lambdas, plus 1 lambda pour l'extension.
- Les caissons N et P doivent être polarisés. Il faut donc les relier respectivement à Vdd et à Vss.

Le schéma de la ?gure suivante présente un résumé de ces contraintes :

4 Schémas

Les schéma théoriques d'un inverseur et d'un Nand2 sont présentés dans les figures suivantes :

5 Travail à effectuer

5.1 Réalisation d'un inverseur

- Décrire le comportement de la cellule inverseuse dans un fichier **.vbe**.
- Dessiner le "stick-diagram" de l'inverseur inv_x1 dont le schéma en transistors est représenté.
- Saisir sous **GRAAL** le dessin de la cellule en respectant le gabarit spécifié.
- Valider les règles de dessin symbolique en lançant **DRUC** sous **GRAAL**.
- Extraire la netlist de l'inverseur au format **.al** avec **COUGAR**.

- Extraire le VHDL comportemental avec **YAGLE**.
- Effectuer la preuve formelle entre le ?chier **.vbe** extrait par **YAGLE** et le ?chier **.vbe** de la spéci?cation initiale.
- Automatisez la véri?cation en écrivant un Make?le.

5.2 Réalisation d'un Nand2

- Décrire le comportement de la cellule Nand à 2 entrées dans un fichier **.vbe**.
- Dessiner le "stick-diagram" du Nand2 dont le schéma en transistors est représenté.
- Saisir sous **GRAAL** le dessin de la cellule en respectant le gabarit spécifié.
- Valider les règles de dessin symbolique en lançant **DRUC** sous **GRAAL**.
- Extraire la netlist de la porte nand2 au format **.al** avec **COUGAR**.
- Extraire le VHDL comportemental avec **YAGLE**.
- Effectuer la preuve formelle entre le ?chier **.vbe** extrait par **YAGLE** et le ?chier **.vbe** de la spéci?cation initiale.

- Automatisez la véri?cation en écrivant un Make?le.

5.3 Réalisation d'un And2

- Décrire le comportement de la cellule And à 2 entrées dans un fichier **.vbe**.
- Dessiner le "stick-diagram" du And2.
- Définir la hiérarchie à utiliser pour créer cette cellule.
- Saisir sous **GRAAL** le dessin de la cellule en respectant le gabarit spécifié.
- Valider les règles de dessin symbolique en lançant **DRUC** sous **GRAAL**.
- Extraire la netlist de la porte nand2 au format **.al** avec **COUGAR**.
- Extraire le VHDL comportemental avec **YAGLE**.
- Effectuer la preuve formelle entre le ?chier **.vbe** extrait par **YAGLE** et le ?chier **.vbe** de la spéci?cation initiale.
- Automatisez la véri?cation en écrivant un Make?le.

6 Rapport

Vous rédigerez un compte-rendu d'une page maximum pour ce TP dans lequel vous expliquerez les choix effectués pour la création des cellules, ainsi que la démarche de validation. Vous joindrez les différents fichiers.