TP5 : Dessin de cellules précaractérisées

Introduction Le but de ces quatre séances de TP est de présenter quelques outils de la chaîne ALLIANCE ainsi que du flot back-end CORIOLIS, dont :

> Le langage de description procédural STRATUS L'éditeur de layout GRAAL Le vérificateur de règles de dessin DRUC L'extracteur de netlist COUGAR L'abstracteur fonctionnel YAGLE L'outil de preuve formelle PROOF L'outil de placement du flot CORIOLIS MISTRAL L'outil de routage de la chaîne ALLIANCE NERO
>
> La première séance portera sur le dessin sous GRAAL d'une cellule inverseuse et

d'un buffer instanciant cet inverseur. Les notions de cellules précaractérisées, de gabarit et de hiérarchie de cellules seront introduites. La deuxième séance portera sur le routage "overcell" d'une cellule compteur de bits. Les deux dernières séances vous permettront d'obtenir une vue physique de l'amd2901 que vous avez conçu lors du TP numéro 3 et 4. Munissez-vous de tous les documents né- cessaires et des fichiers déjà créés.

1-Dessin d'un inverseur et d'un buffer sous

GRAAL =

Introduction

Dans les TP précédents nous avns utilisé des cellules d'une bibliothèque. Cette biblioth èque 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. La première partie de cette séance a pour objectif de dessiner une cellule inverseuse inv_x1 sous la forme d'une cellule précaractérisée sxlib en respectant règles de dessin fournies. Cette cellule sera instanciée pour concevoir un buffer.

1.1 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
```

1.2 GRAAL

L'éditeur de layout GRAAL manipule six 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 /asim/alliance/etc/cmos.graal.

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

1.3 COUGAR

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

1.4 YAGLE

L'outil YAGLE est capable d'extraire la description VHDL comportementale d'un circuit 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

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 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. 2.1.5 PROOF Lorsqu'on veut prouver l'équivalence de deux descriptions comportementales de type dataflows d'un même circuit à n entrées, on peut simuler par asimut 2n 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 descriptions file1.vbe et file2.vbe par la commande :

proof file1 file2

2.2 Schéma d'un inverseur Le schéma théorique d'un inverseur est présenté à la gure suivante : Vdd Vss In Out FIG. 1 Schéma en transistors d'un inverseur C-MOS 2.3 Schéma d'un buffer Le schéma théorique d'un buffer et la hiérarchie utilisée sont présentés sur les - gures suivantes : Vdd Vss Vdd Vss In Out FIG. 2 Schéma en

transistors d'un buffer C-MOS inv_x1 inv_x1 buff_x1 inv_x1 inv_x1 buff_x1 FIG. 3 Hiérarchie d'un buffer C-MOS Master ACSI 5

2.4 Le gabarit sxlib Les cellules sxlib ont toutes une hauteur de 50 et une largeur multiple de 5 Les alimentations Vdd et Vss sont réalisées en Calu1 ; elles ont une largeur de 6 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 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 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 (gure 5) La largeur minimale de CAlu1 est de 2 , plus 1 pour l'extension (gure 6) Les caissons N et P doivent être polarisés. Il faut donc les relier respectivement à Vdd et à Vss Le schéma de la gure 4 présente un résumé de ces contraintes

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 sur la figure 1

Saisir sous GRAAL le dessin de la cellule en respectant le gabarit spéci é sur la

figure 4

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 fichier .vbe extrait par YAGLE et le fichier

.vbe de la spécification initiale Automatisez la vérification en écrivant un Makefile.

5.2 Réalisation d'un buffer

Le buffer est réalisé sous GRAAL à partir de l'instanciation de deux inverseurs. La hiérarchie ainsi créée est représentée sur la Le schéma en transistors est représenté sur la

Décrire le comportement de la cellule buffer dans un fichier .vbe Saisir sous GRAAL le dessin de la cellule en respectant le gabarit spécifé sur

la gure 4. Vous utiliserez pour cela la fonction d'instanciation de GRAAL. La cellule à instancier est bien sûr l'inverseur créé précédemment, que vous relierez (routerez) manuellement

Valider les règles de dessin symbolique en lançant DRUC sous GRAAL Extraire la netlist du buffer au format .al avec COUGAR Extraire le VHDL comportemental avec YAGLE Effectuer la preuve formelle entre le fichier .vbe extrait par YAGLE et le fichier

.vbe de la spécification initiale Automatisez la vérification en écrivant un Makefile. N'oubliez pas que les mans existent