

# TP7 : Placement et routage du circuit AMD2901

1. 1 Outils utilisés
2. 2 Environnement technologique
3. 3 Précautions quant au nommage des fichiers
4. 4 Travail sur le chemin de données : Préplacement des opérateurs
5. 5 Travail sur le coeur : Préplacement des structures régulières
6. 6 Travail sur le circuit complet
  1. 6.1 Placement de la couronne de plots et du coeur
  2. 6.2 Routage des alimentations
  3. 6.3 Placement de la logique irrégulière
  4. 6.4 Routage des signaux d'horloge
  5. 6.5 Routage des signaux logiques
  6. 6.6 Validation du chip
7. Conclusion

## 1 Outils utilisés

Vous allez utiliser les outils de placement flot Coriolis et le routeur d'Alliance, ainsi que tous les outils de vérification vus dans le TP précédent. Vous utiliserez aussi **lvx**, le comparateur de netlists. Lorsque le système est trop complexe il est difficile d'utiliser **proof**, le comparateur formel (calculs trop long). On utilise alors une comparaison de netlists. Essayez les deux méthodes (proof et lvx).

## 2 Environnement technologique

Outre l'environnement technologique de la première partie du TP, vous devez positionner :

```
> export VH_MAXERR=10
> export MBK_WORK_LIB=.
> export MBK_CATA_LIB=$ALLIANCE_TOP/cells/sxlib
> export MBK_CATA_LIB=$MBK_CATA_LIB :$ALLIANCE_TOP/cells/dp_sxlib
> export MBK_CATA_LIB=$MBK_CATA_LIB :$ALLIANCE_TOP/cells/pxlib
> export MBK_CATA_LIB=$MBK_CATA_LIB :.
> export MBK_CATAL_NAME=CATAL
> export MBK_IN_IO=vst
> export MBK_OUT_IO=vst
> export MBK_IN_PH=ap
> export MBK_OUT_PH=ap
> export CRL_OUT_IO=vst
> export CRL_OUT_PH=ap
> export PYTHONPATH=/asim/coriolis/lib/python2.3/site-packages/stratus
> export PYTHONPATH=/asim/coriolis/lib/python2.3/site-packages/isobar :$PYTHONPATH
> export PYTHONPATH=/asim/coriolis/lib/python2.3/site-packages :$PYTHONPATH
```

NB : Ces variables d'environnement sont positionnées par défaut, mais il peut être utile de les vérifier.

## 3 Précautions quant au nommage des fichiers

D'une manière générale, les fichiers décrivant une netlist logique doivent porter le même nom que le fichier correspondant décrivant la vue physique. C'est à dire que le fichier amd2901\_dpt.vst (vue logique) doit correspondre au fichier amd2901\_dpt.ap (vue physique). Il en va de même pour le fichier amd2901\_core.

## 4 Travail sur le chemin de données : Préplacement des opérateurs

Le TP précédent vous a permis de décrire la netlist hiérarchique du chemin de données. Vous allez maintenant placer les colonnes du chemin de données de manière à profiter de la régularité du chemin de données. Pour réaliser votre placement des opérateurs, vous disposez des fonctions de STRATUS suivantes :

- Place()
- PlaceRight(), PlaceTop(), PlaceLeft(), PlaceBottom()
- SetRefIns()
- DefAb(), ResizeAb()

Toutes ces fonctions doivent être utilisées dans la méthode Layout. Reprenons l'exemple du TP précédent, on donne le code suivant pour le fichier circuit.py :

```
#!/usr/bin/env python
from stratus import *
# definition de la cellule
class circuit ( Model ):
...
def Layout ( self ):
    Place ( self.instance_nand2_4bits, NOSYM, XY ( 0, 0 ) )
    PlaceRight ( self.instance_or2_4bits, NOSYM )
    PlaceRight ( self.instance_add2_4bits, NOSYM )
```

Ensuite pour le fichier test\_circuit.py, il faut rajouter l'appel à la méthode Layout :

```
#!/usr/bin/env python
from stratus import *
from circuit import circuit
my_circuit = circuit ( "mon_circuit" ) # creation du circuit
my_circuit.Interface() # creation de l'interface
my_circuit.Netlist() # creation de la netlist
my_circuit.Layout() # creation du layout
my_circuit.View() # pour afficher le layout
my_circuit.Save ( PHYSICAL ) # sauver les fichiers mon_circuit.vst et mon_circuit.ap
```

Reprenez le fichier amd2901\_dpt.py. Pour l'instant, ce fichier ne comporte qu'une description de la netlist. Cela vous a permis de générer une description structurelle sous la forme d'un fichier .vst. Il s'agit maintenant de placer explicitement les colonnes. **Le placement des colonnes du chemin de données ne doit pas être fait au hasard. La faisabilité et la qualité du routage en dépendent**'''

Aidez-vous du manuel de STRATUS :

*~file :/asim/coriolis/share/doc/en/html/stratus/index.html*

Utilisez STRATUS pour générer le tout :

```
> ./execute_amd2901_dpt.py
```



## 5 Travail sur le coeur : Préplacement des structures régulières

à partir du fichier de description structurelle coeur du circuit effectuez les étapes suivantes dans la méthode Layout :

- Placer le chemin de données : fonction Place ()
- Agrandir la boîte d'aboutement du coeur : fonction ResizeAb ()
- Placer les rails de rappels d'alimentation dans le coeur : fonctions AlimVerticalRail () et AlimHorizontalRail ()
- Placer les connecteurs du coeur : fonction AlimConnectors ()

**ATTENTION** : La logique "irrégulière" constituant la partie contrôle n'a pas besoin

d'être placée explicitement. Cela sera fait automatiquement par la suite !

### Vérifiez le résultat

```
> ./execute_amd2901_core.py
```

## 6 Travail sur le circuit complet

Prenez le fichier coeur du circuit et complétez la méthode Layout.

### 6.1 Placement de la couronne de plots et du coeur

Dans le fichier amd2901\_chip.py fourni, les plots sont instanciés dans la méthode

Netlist :

```
def Netlist ( self ) :  
    ...  
    Inst ( "pck_px", "p_ck"  
        , map = { ?pad? : self.ck  
                , ?ck? : cki  
                , ?vddi? : self.vdd  
                , ?vssi? : self.vss  
                , ?vdde? : self.vdde  
                , ?vsse? : self.vsse  
                }  
        )
```

Il vous faut donc, dans la méthode Layout :

- Définir la taille de la boîte d'aboutement globale du circuit de façon à ce que les plots puissent être placés à la périphérie : fonction DefAb () (Commencer par définir une boîte d'aboutement de 4000 par 4000 et vous essaieriez ensuite de la diminuer)
- Placer le coeur du circuit au centre de la boîte d'aboutement du chip : fonction PlaceCentric ()
- Définir sur quelle face et dans quel ordre vous souhaitez placer les plots. Cela se fait à l'aide des 4 fonctions : PadNorth (), PadSouth (), PadEast () et PadWest ().

- Vérifiez le résultat :

```
> ./execute_amd2901_chip.py
```

## 6.2 Routage des alimentations

Vous devez utiliser la fonction `PowerRing ()` pour créer la grille d'alimentation. Vérifiez le résultat :

```
> ./execute_amd2901_chip.py
```

## 6.3 Placement de la logique irrégulière

C'est le placeur Mistral qui se charge de placer les cellules de la partie de contrôle. Il détecte quelles sont les cellules qui n'ont pas été placées et complète le placement en utilisant les zones "vides". Pour appeler le placeur Mistral, vous devez faire appel à la fonction `PlaceGlue ()`

🔒

Attention : Pour pouvoir placer automatiquement la logique "irrégulière", il faut que les plots soient placés. L'outil de placement du flot CORIOLIS place les cellules en se basant sur les attirances de celles-ci vers les plots ainsi que vers les cellules déjà placées. 🔒 Vérifiez le résultat :

```
> ./execute_amd2901_chip.py
```

🔒

Le placement automatique se termine par l'appel à la fonction `FillCell ()` qui effectue le placement automatique de cellules de bourrage. 🔒

Vérifiez le résultat :

```
> ./execute_amd2901_chip.py
```

## 6.4 Routage des signaux d'horloge

Vous devez utiliser la fonction `RouteCk ()` qui route le signal d'horloge. Vérifiez le résultat :

```
> ./execute_amd2901_chip.py
```

🔒

## 6.5 Routage des signaux logiques

Routez automatiquement tous les signaux autres que le signal d'horloge et les signaux d'alimentation en utilisant NERO de la manière suivante :

```
> nero -V -p amd2901_chip amd2901_chip amd2901_chip_r
```

L'option `-p` indique que vous transmettez un placement, à savoir celui du chip. Le troisième argument est la netlist du chip, le quatrième est le fichier résultat.

NOTA BENE : La variable `MBK_CATA_LIB` ne doit contenir qu'une seule fois les chemins d'accès aux bibliothèques.

## 6.6 Validation du chip

- On validera le travail de NERO avec les outils druc, cougar et lvx.

```
> druc amd2901_chip_r
> export MBK_OUT_LO=a1
> cougar -f amd2901_chip_r
> lvx vst a1 amd2901_chip amd2901_chip_r -f
```

- Simulez à nouveau la netlist extraite avec ASIMUT. Précisez le format de la netlist

dans la variable d'entrée MBK\_IN\_LO avant la simulation.

```
> export MBK_IN_LO=a1
```

### Faites attention au fichier CATAL''

- Pour connaître le nombre de transistors, on effectue une extraction du circuit au niveau

transistor :

```
> cougar -v -t amd2901_chip_r amd2901_chip_r_t
```

## Conclusion

Ce TP vous a permis de passer par la plupart des étapes nécessaires à la conception "back-end" et la validation d'un circuit réalisé en cellules précaractérisées avec préplacement des parties régulières.

Ces mêmes outils seront utilisés pour la réalisation du processeur MIPS R3000. Le compte-rendu du TP doit comporter :

Vos logins, vos noms et prénoms, et vos répertoires de travail pour ce TP (laissez libre accès à vos répertoires en lecture !). Une description exacte de la méthodologie employée, incluant les éventuels problèmes rencontrés.

Pour l'amd2901, décrivez le flot de conception. Quels choix avez-vous retenus pour le placement des colonnes du chemin de données, votre circuit est-il limité par les plots ou par la taille du coeur (pad limited ou core limited)... Quels sont les résultats donnés par lvx... Les schémas sont appréciés.

Les Makefiles du flot total. ( Les Makefiles seront testés à la fin de ce TP) NE PAS JOINDRE DE LISTINGS DE FICHIERS (SAUF LES MAKEFILES). Merci et bon courage !