Etude et conception d'un filtre numérique RIF à réponse impulsionnelle programmable

Placement-Routage avec SoC Encounter

1. Création d'un netlist verilog sous design vision

SoC Encounter est un outil de placement routage automatique. Il prend en entrée une netlist au format verilog. A l'aide de design vision, effectuer une synthèse du filtre seul, sans scan et sans portes uC pour des conditions de fonctionnement typiques et une fréquence d'horloge correcte. Sauvegarder la netlist sous le format Verilog.

2. Création du répertoire de travail

a. Création des fichiers pour SoC Encounter avec la techno AMS0.35 :

Se placer dans le répertoire <par> et lancer la commande suivante pour créer l'environnement de travail :

```
$> ams_encounter –tech c35b4
```

Attention : cette commande n'est à utiliser qu'une seule fois pour créer l'environnement (ne pas la relancer à chaque fois).

Créer un répertoire WORK qui servira de base de donnée pour le routeur. C'est depuis ce répertoire que Encounter sera lancé.

Copier les netlists générées par l'outil de synthèse dans le répertoire VERILOG. Le répertoire <par_files> copié dans le répertoire <Project> contient un fichier <TOP_io.v> qui contient la description verilog des plots d'entrée/sortie du circuit. Avant de l'importer, vérifier qu'il est compatible avec votre propre netlist.

b. Organisation du répertoire de travail :

Le répertoire <par> contient les différents fichiers utilisés pour le Placement-Routage :

- VERILOG : Répertoire contenant les Netlist à importer (exemple : TOP_io.v, NETLIST.v).
- Constraints : Répertoire contenant les fichiers de contraintes. Il contient notamment le fichier ctgen.ctstcsh qui spécifie les contraintes pour l'horloge.
- c35b4_std.conf : fichier de configuration par défaut pour l'importation du Design dans Encounter.
- corners.io : fichier contenant les cellules « coins » (à insérer).
- top.io : fichier contenant les informations pour ordonner la couronnes de plots.
- fillxx.tcl : scripts permettant d'insérer les fillers (standard cells et IOs).
- gds2.map : fichier permettant d'exporter le design au format GDSII.
- gemma.tcl : script générique de placement routage.

c. Bibliothèques AMS0.35 utilisées :

Pour pouvoir importer correctement le circuit dans SoC Encounter, l'outil utilise les fichiers suivants :

- ✓ Library Exchange Format (LEF) : ce sont les bibliothèques contenant les différentes cellules utilisées dans le circuit ainsi que le fichier technologique.
  /softslin/AMS_3.70_CDS/artist/HK_C35/LEF/c35b4/c35b4.lef
  /softslin/AMS_3.70_CDS/artist/HK_C35/LEF/c35b4/CORELIB.lef
  /softslin/AMS_3.70_CDS/artist/HK_C35/LEF/c35b4/IOLIB_4M.lef

- ✓ Timing Library (.lib ou .tlf) : ce sont les fichiers contenant les informations de timing pour chaque bibliothèque :
  /softslin/AMS_3.70_CDS/liberty/c35_3.3V/c35_CORELIB.lib
  /softslin/AMS_3.70_CDS/liberty/c35_3.3V/c35_IOLIB_4M.lib

- ✓ Netlists Verilog :
  netlist.v : netlist du circuit obtenue après synthèse
  top_io.v : netlist verilog qui instancie le top et les plots d'entrée/sortie

- ✓ Fichier Captable : c35b4.capTable (optionnel)

- ✓ Fichier LEF et TLF des blocs durs. (optionnel)

- ✓ Fichier de contrainte (SDC). (exemple : FILTRE.sdc) (optionnel)

3. Lancement de SoC Encounter :

Se placer dans le répertoire <WORK>, et taper la commande <encounter> pour lancer SoC Encounter.
La fenêtre dans laquelle SoC Encounter est lancée devient la console de l'outil.
Attention : ne pas lancer en arrière plan (<encounter &>), sinon la fenêtre console de l'outil est perdue et certaines commandes ne pourront être lancées.

Lorsque SoC Encounter est lancé, 2 fichiers sont créés dans le répertoire de travail :

- Encounter.logX: journal de SoC Encounter.
- Encounter.cmdX : historiques des commandes lancées durant la session.

NB : les bibliothèques utilisées par la GUI de SoC Encounter peuvent parfois être incompatibles avec certaines versions du système d'exploitation. Dans ce cas, il faut utiliser la commande suivante (éventuellement avec l'option « -f » si on veut le plein écran)

rdesktop -g 95% localhost

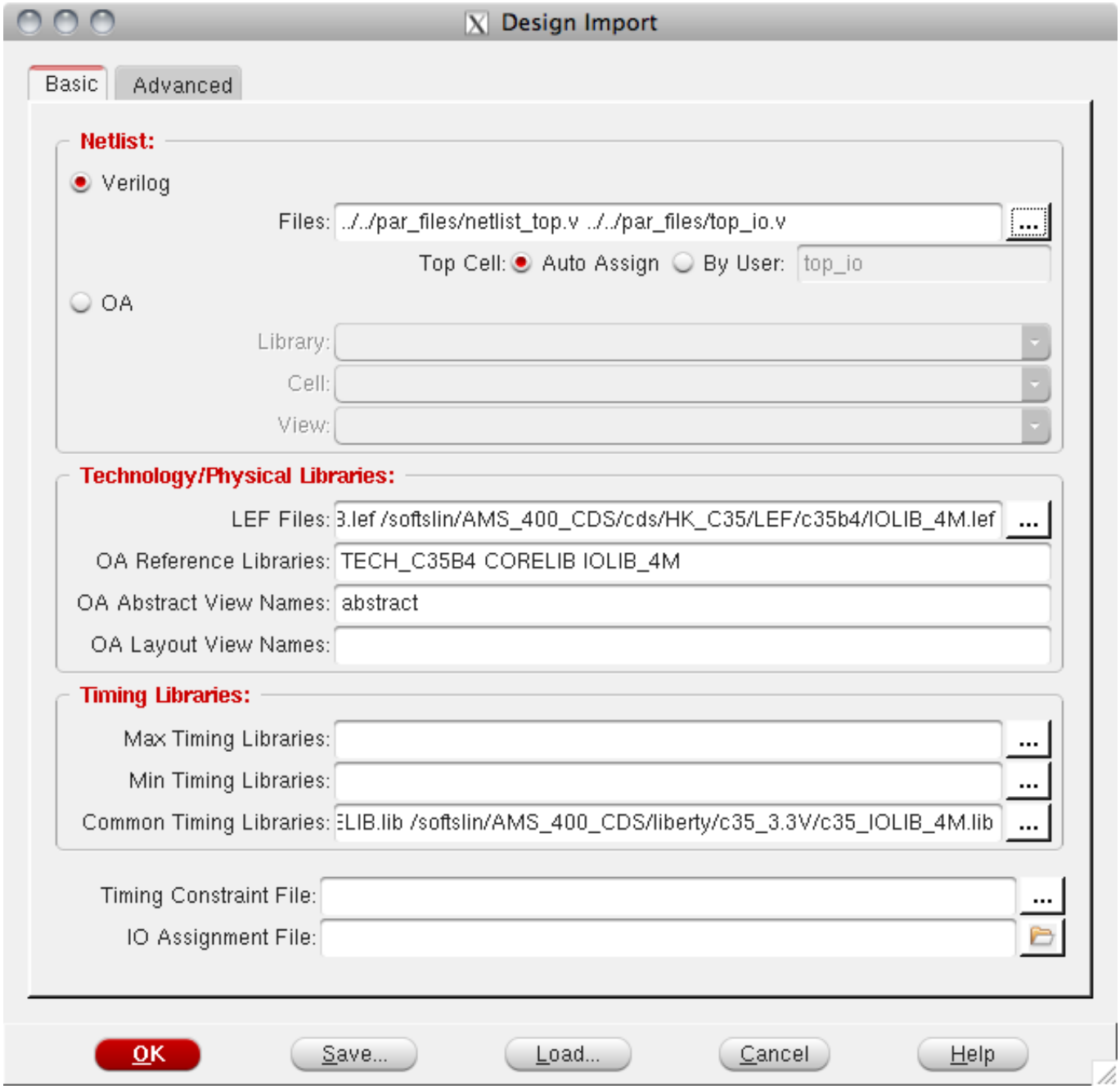
4. Importation du Design

Pour importer les bibliothèques nécessaires, leurs fichiers de timing respectifs ainsi que le Design, aller dans le menu:

Design > Design Import…

Cliquer sur Load… et selectionner le fichier <../c35b4_std.conf> qui contient les options par défauts. Ajouter les fichiers de votre (vos) netlist(s) (TOP_io.v et NETLSIT.v). Sélectionner « Auto Assign » pour la TOP Cell.

Menu Basic :



Dans l'onglet « Advanced » et le menu « Power », spécifiez les noms globaux des alimentations de votre circuit: vdd! et gnd!. Ces noms seront utilisés plus tard lors du routage des alims.

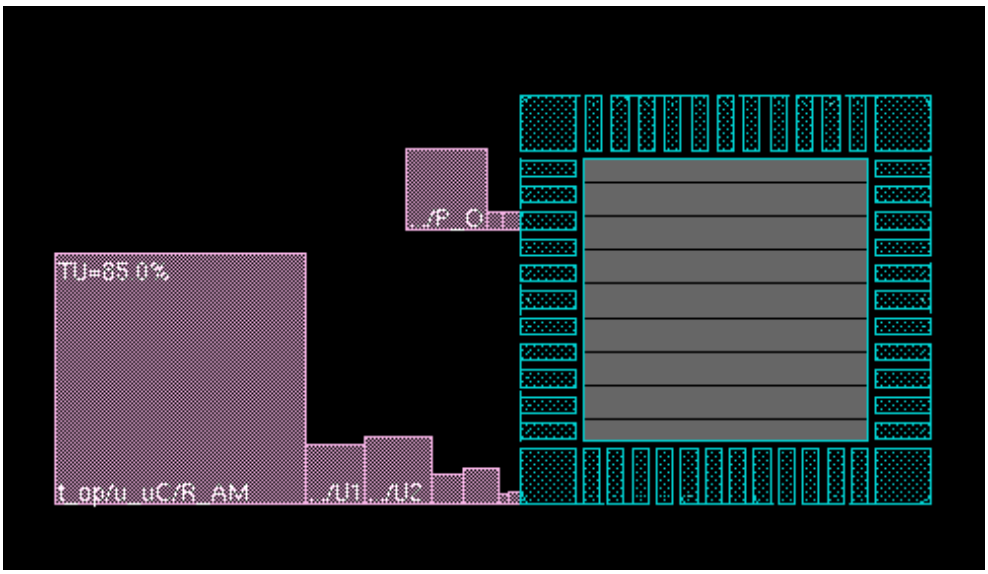
Il est possible de sauvegarder la configuration pour l'importation du Design (Save…). Pour importer le Design durant une autre session il suffit alors de faire :

- Design>Design Import.
- Cliquer sur Load.
- Choisir le fichier de configuration (exemple TOP.conf).
- Cliquer sur OK.

Quand tous les paramètres sont définis et que le fichier de configuration est sauvegardé, cliquer sur OK pour importer le Design dans SoC Encounter.

Une fois le design chargé, aller dans le menu :
Option > Set Preference > Display

Prendre 1 comme valeur pour le « min Floorplan Module Size ». On peut alors voir apparaître tous les blocs du circuit (si la netlist est sous forme hierarchique) en utilisant le bouton <Ungroup>.



5. Sauvegarde du Design

Penser à sauvegarder le Design régulièrement !

Il est possible (très fortement conseillé) de sauvegarder le Design complet aux différentes étapes du flot sous différents noms.

Sauvegarde du Design complet : **Design > Save Design…** (ex : TOP_s1.enc)

On pourra alors charger le Design lors d'une autre session : **Design > Restore Design…**

6. Spécification du Floorplan

a. Couronne de plots :

Lors de l'importation du Design, les plots sont placés aléatoirement, et les coins et les plots d'alimentation ne sont pas instanciés. Il est possible de créer et charger un fichier TOP.io pour déterminer la position des plots.

1) Editer le fichier corner.io et instancier les plots d'alimentation en ajoutant ces 3 lignes:

Pad: PWR1 N VDD3ALLP
Pad: GND1 S GND3ALLP

2) Aller dans le Menu **Design > Load > I/O File…** et choisir le fichier corner.io → OK
Les coins sont chargés.

3) Vous pouvez (éventuellement) utiliser un fichier de plots et le modifier, afin que les plots soient mieux répartis autour du cœur :

a- Editer et modifier le fichier TOP.io (voir exemple en annexe)
b- Le charger dans SoC Encounter : **Design > Load > I/O File…**

Remarque:

L'idéal est d'avoir le même nombre de plots sur chaque côté du circuit, et que les plots d'alimentation VDD et GND, ainsi que le plot d'horloge (CLK), soient placés à peu près au milieu d'un des côtés.

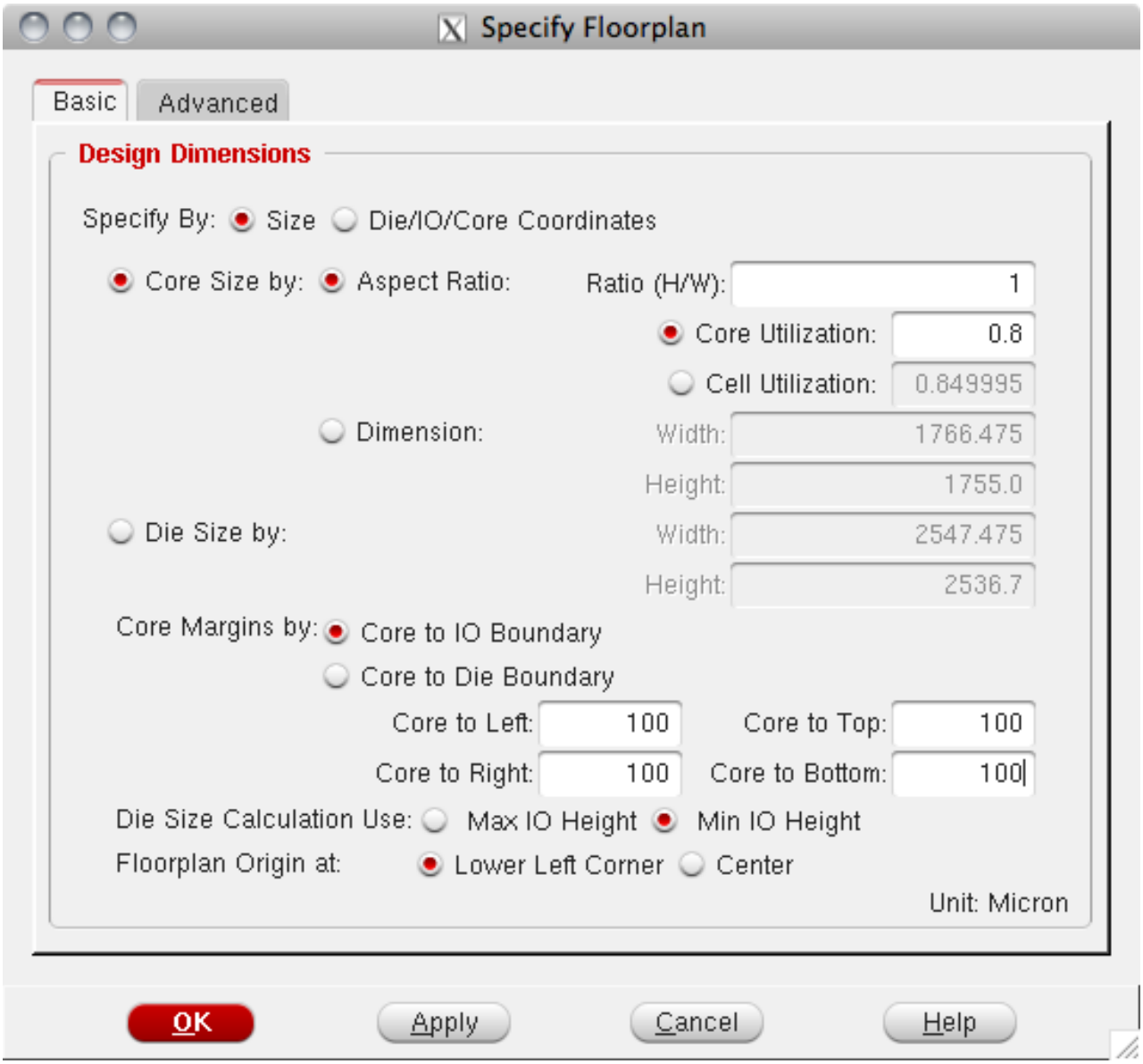
b. Préparation du floorplan :

Floorplan > Specify Floorplan…

Choisissez les valeurs suivantes :

- Ratio : **1.0**
- Core Utilisation : **0.80**
- Core Margins by : Core to IO Boundary : **100.0** (sur les quatre côtés)

* *il peut être nécessaire de laisser une « marge » autour du cœur pour pouvoir, par exemple, mettre les anneaux d'alimentation.*



c. Power planning

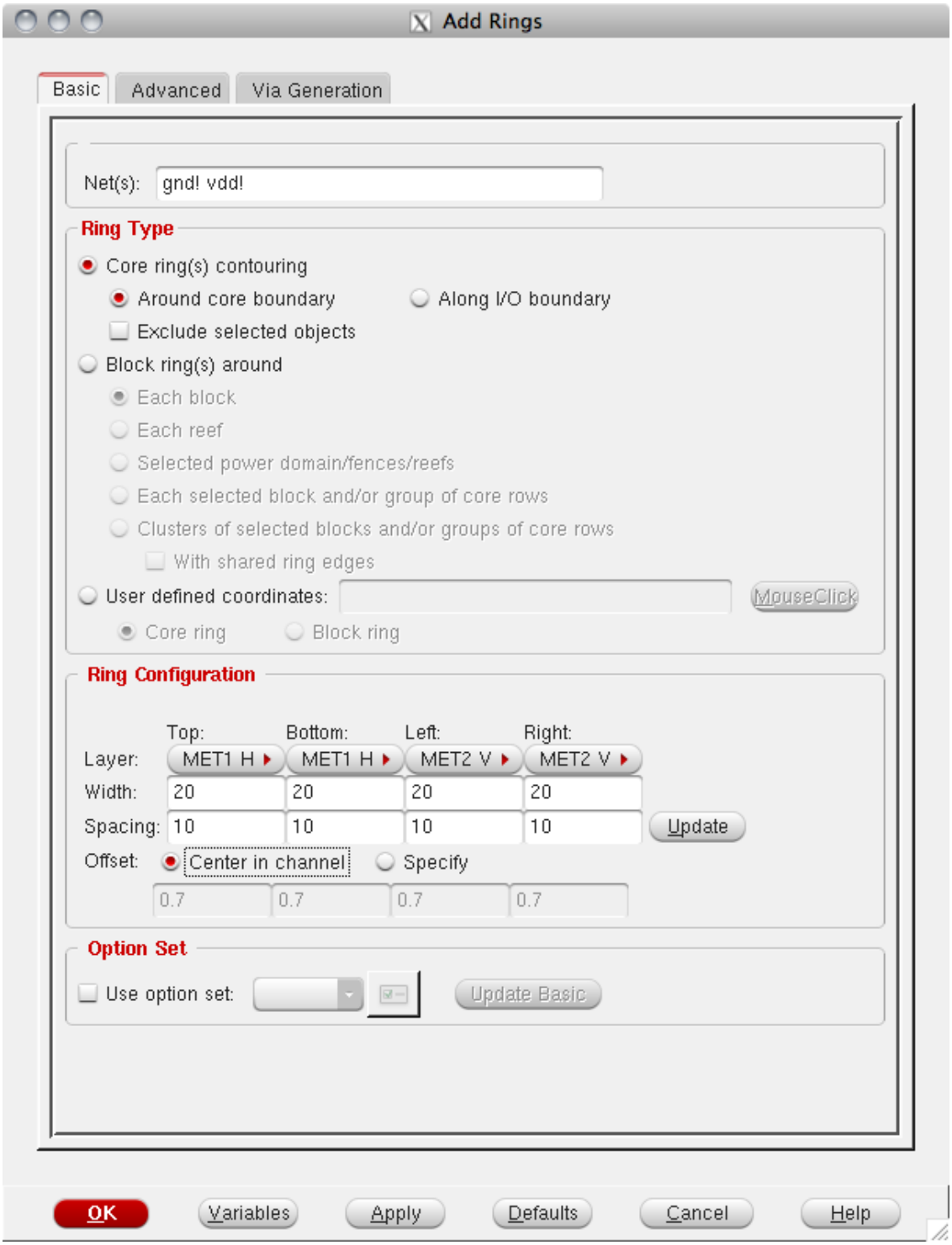
Cette étape consiste à ajouter les anneaux et les rails d'alimentation. La largeur des anneaux d'alimentation va dépendre de la consommation de votre circuit et des caractéristiques de la techno utilisée. Ici il faut compter environ 1µm pour 1mA.

- **Anneaux d'alimentation :**

Power > Power Planning > Add Rings…

Définir la largeur (**width = 20**) et l'espacement (**spacing = 10**) des rails d'alimentation pour chaque côté du cœur.

Attention : seuls **gnd!** et **vdd!** doivent être définis dans l'onglet « **Net(s**) ».



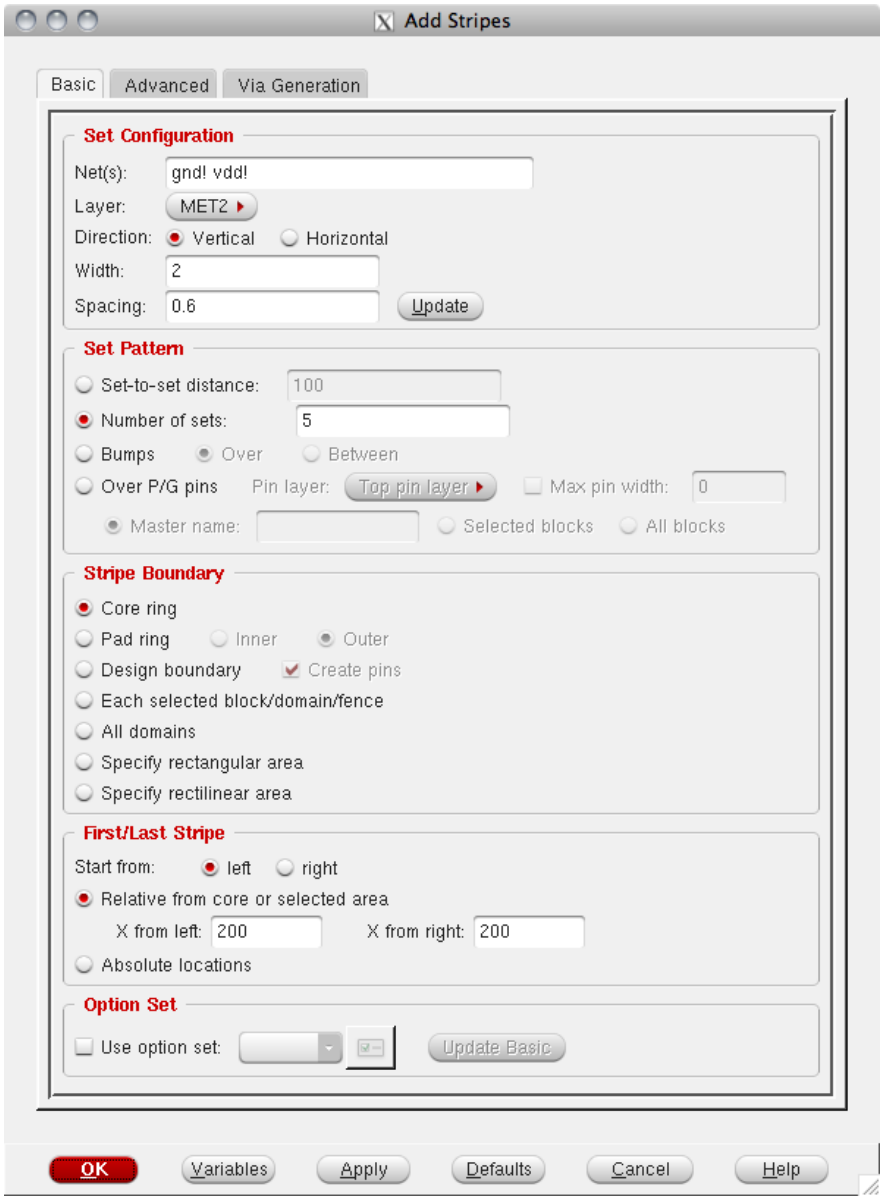
*Note 1 : Si le design contient des **blocs**, il faut commencer par définir leurs rails d'alimentations en sélectionnant **Block ring(s) around**, puis ceux du cœur en sélectionnant **Core ring(s)contouring**.*

*Note 2 : L'**offset** est l'espacement entre le cœur du circuit et le premier anneau d'alimentation. En sélectionnant **Center in channel**, les anneaux d'alimentation sont placés à égale distance des plots et du cœur.*

- **Stripes:**

Power > Power Planning > Add Stripes…

Définir l'espacement entre les 2 stripes ainsi que leur largeur. (Width = 2, Spacing = 0,6)



Pour positionner les stripes, 2 solutions : On donne soit la distance entre chaque paire de stripes, ou on spécifie le nombre de pairs (ici, on choisira 4) et la position de la première et de la dernière paire de stripe (200 et 200).

- **Routage des lignes vdd! et gnd! du coeur:**

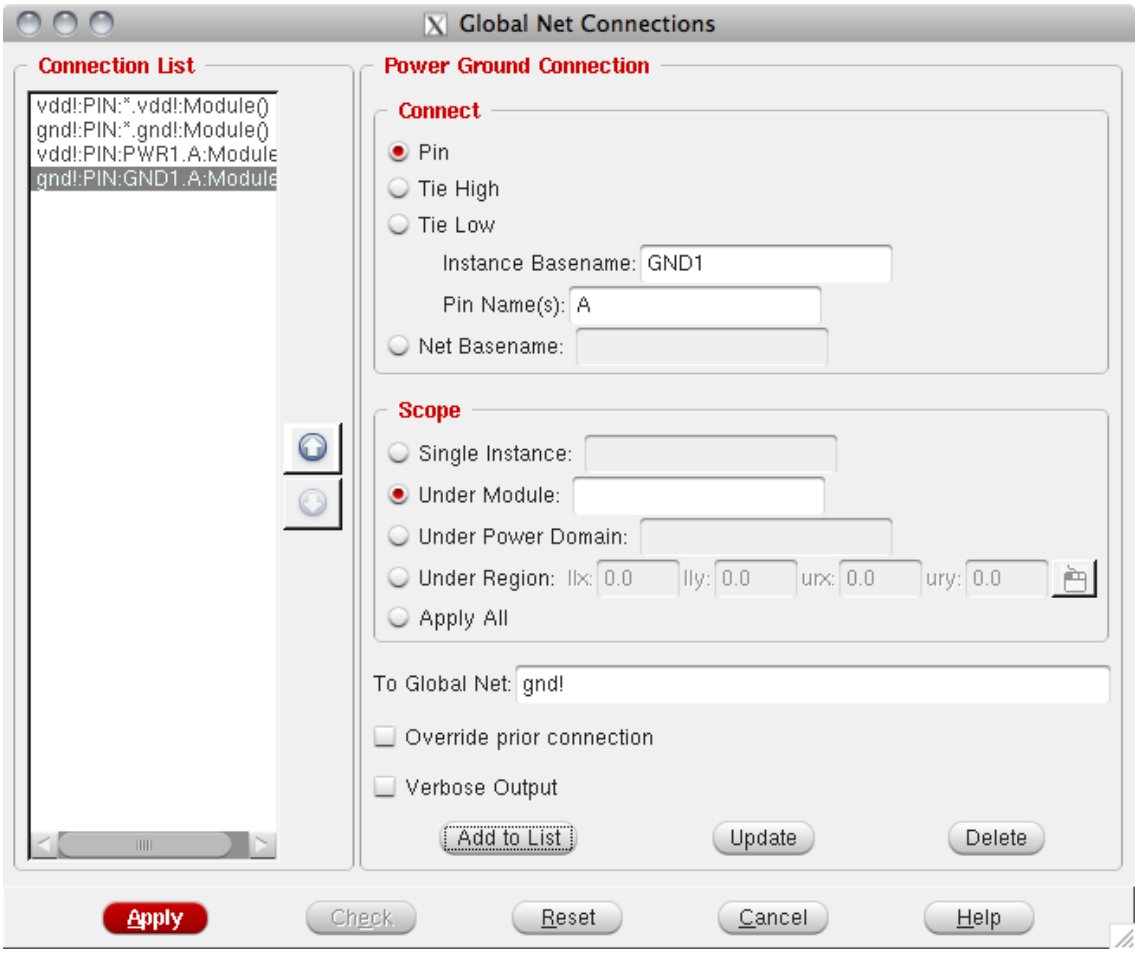
1) Dans le menu **Floorplan > Connect Global Net :**

a- Définir les pins de connexion des cellules :

- Taper **vdd !** dans les champs **« Pins »** et **«To Global Net »**, puis cliquer sur **« Add to List ».**
- Recommencer pour **gnd !**

b- Définir les pins de connexion des plots d'alimentation

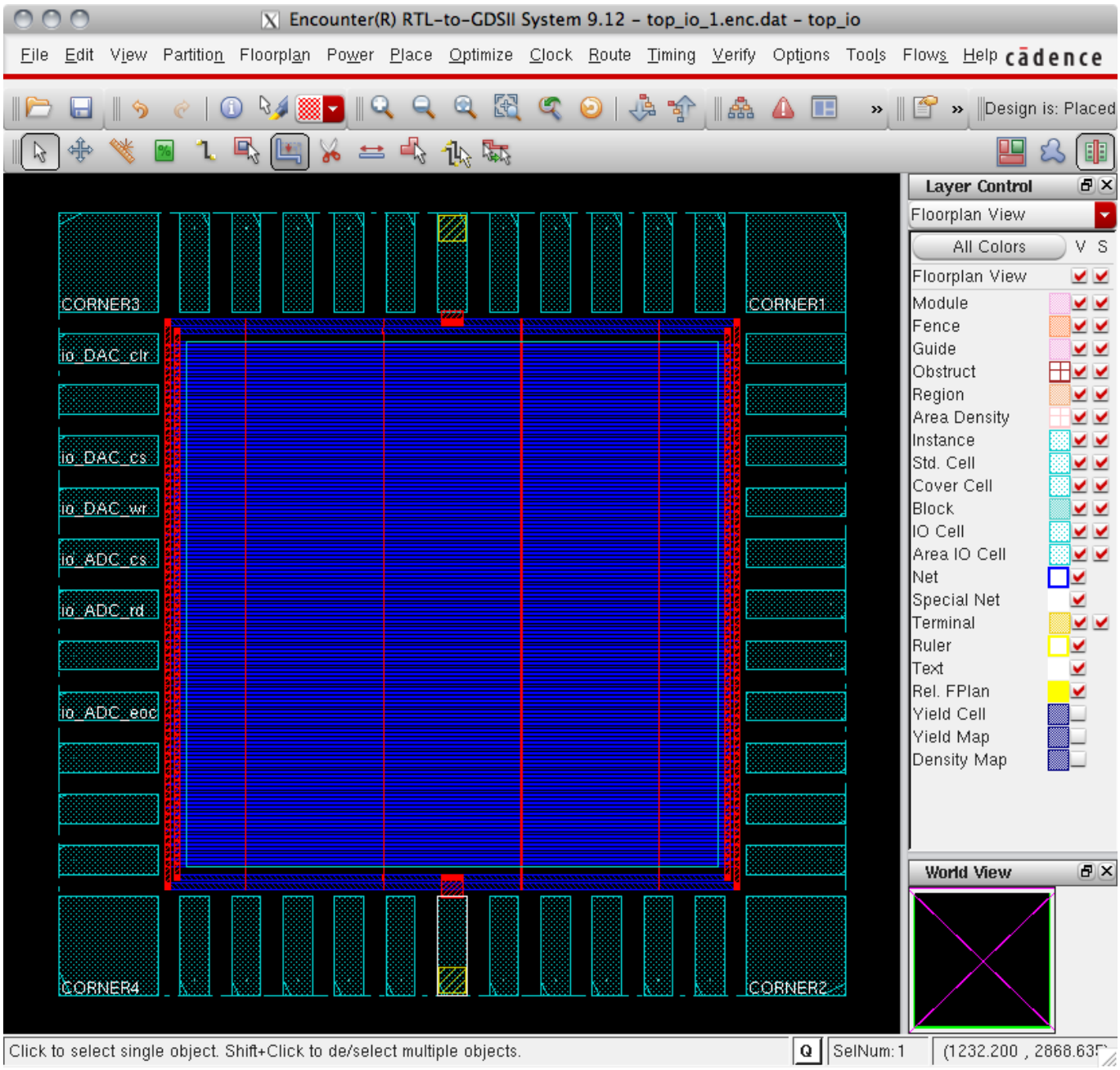
- pin : A
- Instance Basename : PWR1 (GND1)
- To Global Net : vdd ! (gnd!)
- cliquer sur **« Add to List »**



c- Quand toutes les connexions sont définies, cliquer sur « **Apply** » puis « **Close** ».

2) Dans le menu **Route > Special Route…** sélectionner tout et valider.

Au final, vous devez obtenir quelque chose comme ceci (pensez à faire une sauvegarde !)



7. Placement des cellules

Place > Place Standard Cells…

1. Sélectionner le mode (Full placement)
2. Sélectionner l'option (Include pre-place optimisation)

En modifiant le mode de vue, vous pouvez voir les différentes cellules placées.

Pour un placement (et un routage) optimal, il est conseiller de fixer une densité de placement < à 90%
Comme vu précédemment, le choix de la densité de placement dépend beaucoup de la constitution de chaque bloc).

Dans le cas de blocs pré-placés, la densité de placement de chaque bloc est indiquée en haut à gauche du bloc. Dans le cas d'un floorplan très contraint, le placeur peut avoir des difficultés à respecter les régions définies (en particulier pour les blocs définis en fence).

Pour s'en assurer, il faut faire ceci après le placement:

Place > Check Placement …
Un petit rapport de violations de placement est alors généré dans la console.

Exemple :
```
Begin checking placement ...
Region/Fence Violation:    633
*info: Placed = 22604
*info: Unplaced = 0
Placement Density:75.01
```

Si il y a des violations, faire la manipulation suivante :
Place > Refine Placement …
Puis refaire **Place > Check Placement …**

```
Begin checking placement ...
*info: Placed = 22604
*info: Unplaced = 0
Placement Density:75.01
```

Si il reste encore des violations de placement, c'est que le floorplan est trop contraint. Il faut revoir les dimensions des blocs.

8. Génération de l'arbre d'horloge

a. Spécifications :

Pour générer l'arbre d'horloge, il faut un fichier de spécifications. Editer le fichier **ctgen.ctstch** et remplir les champs suivant pour l'horloge (CLK) et éventuellement le Reset:

```
AutoCTSRootPin      io_CLK/Y
#noms du plot et port de l'horloge
MaxDelay            30ns
MinDelay            0ps
#delay entre le plot et la fin de l'arbre (FF)
MaxSkew             200ps
#différence max de delay entre les différentes bascules
NoGating            rising
Buffer              BUF2 BUF4 BUF6 BUF8 BUF12 BUF15
#type de buffers pour   équilibrer l'arbre d'horloge
End
```

b. Synthèse de l'arbre d'horloge:

Clock > Synthetise Clock Tree…
Choisir le fichier de spécifications et valider

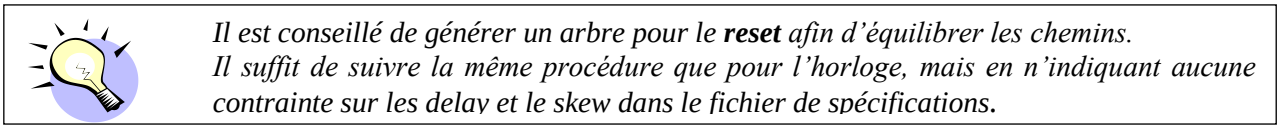
c. Visualisation:

Pour visualiser l'arbre, sélectionner :

Clock > Display > Display Clock Tree…

Sélectionner **Display Clock Tree / All Level** pour visualiser les "branches" de l'arbre.

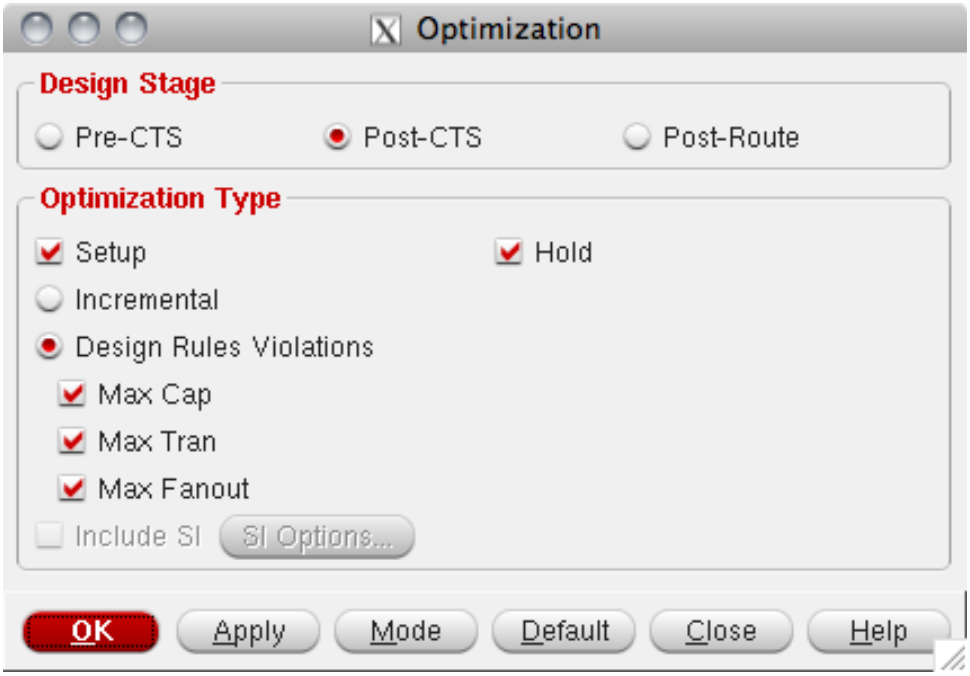
Sélectionner **Display Clock Phase Delay** pour visualiser les différents niveaux de l'arbre d'horloge.



9. In Place Optimization (IPO) post-CTS

Après le placement et la synthèse de l'arbre d'horloge, il peut être nécessaire de bufféríser certains chemins afin de régler les éventuels problèmes de timing, fanout, transitions ou capacités; cette action se fait automatiquement dans SoC Encounter, c'est l'IPO.

Dans le menu **Optimize > Omtimize Design :** Choisir Post-CTS, Setup and Hold Time, et Max Cap, Max Tran et Max Fanout.



10. Insertion des Fillers

Cette étape consiste à insérer des **cellules** ou des **plots vides** (sans fonction), dans tous les espaces libres du design. Ceci peut être réalisé grâce aux scripts fillperi.tcl et fillcore.tcl.

a. IO Fillers

```
$> source  ../fillperi.tcl
```

b. Core Fillers

```
$> source ../fillcore.tcl
```

11. Routage de l'horloge

Il est impératif de router les signaux spécifiques d'abord. Ceci sera fait à l'aide des commandes suivantes à la console (permettant de sélectionner les fils et de les router) :

```
selectNet –allDefClock
setNanoRouteMode -quiet –routeSelectedNet0nly true
globalDetailRoute
```

Aller voir l'arbre d'horloge, identifier les connexions de la CLK (et éventuellement le RESET !)

12. Routage complet du circuit

Route > Nanoroute > Route

Il est possible de choisir le type de routage désiré. On peut aussi modifier certains paramètres le menu Attribute de NanoRoute.

Désélectionner l'option Selected Nets Only et lancer le routage.

Le routage est réalisé en plusieurs passes afin d'obtenir un circuit sans violation des règles de dessin.

13. Verification :

Après le routage, il est nécessaire de lancer une vérification sur les règles de dessins, d'antennes ainsi que sur les différentes connections.

Verify > Verify Geometry…

Selectionner les verifications à effectuer et cliquer sur OK.

14. Exportation de fichiers

La dernière étape est l'exportation des fichiers nécessaires pour le DRC, le LVS et les simulations post-routage.

Créer un répertoire pour sauvegarder ces fichiers dans le répertoire <par>:

```
$> mkdir RESULTS
```

- **SDF :**

Ce fichier contient le calcul des délais des interconnections. Il est utilisé pour les simulations rétro-annotées.

Il faut d'abord refaire une extraction RC.
Timing > Extract RC …
Timing > Write SDF …

Se placer dans le répertoire RESULTS et entrer le nom du fichier. Cliquer sur OK pour sauvegarder le fichier.

- **DEF :**

Ce fichier permet d'importer le design du circuit dans cadence. On peut aussi utiliser le fichier gdsII.

File > Save > DEF…

Cliquer sur OK pour sauvegarder le fichier dans RESULTS.

- **GDSII :**

Ce fichier est une vue physique du circuit. Il sera importé dans cadence pour les vérifications DRC et LVS.

File > Save > GDSII…

Cliquer sur OK pour sauvegarder le fichier dans RESULTS.

- **Verilog :**

Design > Save > Netlist…

Cette étape est nécessaire car des cellules (buffers) ont été ajoutées lors de la génération de l'arbre d'horloge, la netlist a donc changé.

Cliquer sur OK pour sauvegarder le fichier dans RESULTS.

15. Caractéristiques

Les principales caractéristiques du circuit (surface globale, nombre de cellules, nombre d'interconnexions, …) peuvent être obtenues en générant le « summary report ». (Interface graphique).

16. Validation après placement-routage

La netlist obtenue après placement routage (en Verilog) peut être simulée, avec rétro-annotation des informations contenues dans le fichier SDF. Le testbench VHDL développé précédemment (version avec ou sans scan, selon la netlist routée) doit être légèrement adapté : le composant à tester est TOP_io …

Il faut aussi compiler la netlist du circuit et l'ensemble des fichiers nécessaires pour le bench dans la même bibliothèque (sous le répertoire bench). La netlist Verilog doit par ailleurs être compilée avec la commande "vlog" au lieu de "vcom". Il n'y a pas à écrire de configuration, l'ensemble des blocs se trouvant alors dans la même bibliothèque.

Avant de lancer la compilation, la netlist Verilog doit être éditée pour enlever les "corner" (non reconnus par le simulateur). Il peut aussi être nécessaire de corriger certaines erreurs signalées pendant la compilation (par exemple, un signal défini avec un indice, comme "test_se[7]" qui doit être seulement écrit "test_se").

La simulation est lancée comme précédemment par la commande "vsim". Toutefois, il faut sélectionner le menu Simulate et définir non seulement le bench à simuler, dans la nouvelle bibliothèque créée, mais aussi les bibliothèques à employer (CORELIB et IOLIB_4M, cf. chemins d'accès en section 1) dans l'onglet "Libraries" et le fichier SDF obtenu dans l'onglet "SDF" (cocher "Reduce SDF errors to warnings").

17. Ce qu'il faut faire

Effectuer le placement routage du circuit en commentant précisément les différentes étapes. Par exemple, lors de la génération de l'arbre d'horloge, de grandes zones du circuit sont dépourvues d'amplificateurs. En vous basant sur la vue "amoeba", expliquez ce phénomène.

Valider cette dernière étape du flot de conception en effectuant la simulation après placement-routage (verilog + sdf)

18. Annexe

Corner.io

Version: 2

Pad: CORNER4 SW CORNERP
Pad: CORNER3 NW CORNERP
Pad: CORNER2 SE CORNERP
Pad: CORNER1 NE CORNERP

Pad: PWR1 N VDD3ALLP
Pad: GND1 S GND3ALLP

Top.io :

Pad: CORNER1 NE
Pad: CORNER2 SE
Pad: CORNER3 NW
Pad: CORNER4 SW

Pad: io_CLK N
Pad: io_RESETN
Pad: io_uC_Config_0 N
Pad: io_uC_Config_1 N
Pad: io_uC_Config_2 N
Pad: io_uC_Config_3 N
Pad: GND2 N
Pad: io_uC_Config_4 N
Pad: io_uC_Config_5 N
Pad: io_uC_Config_6 N
Pad: io_uC_Config_7 N

Pad: io_uC_Config_8 E
Pad: io_uC_Config_9 E
Pad: io_uC_Config_10 E
Pad: io_uC_Config_11 E
Pad: io_uC_Config_12 E
Pad: io_uC_Config_13 E
Pad: io_uC_Config_14 E
Pad: io_uC_Config_15 E
Pad: io_Filter_In_7 E
Pad: io_Filter_In_6 E
Pad: io_Filter_In_5 E

Pad: io_Filter_In_4 S
Pad: io_Filter_In_3 S
Pad: io_Filter_In_2 S
Pad: io_Filter_In_1 S
Pad: io_Filter_In_0 S
Pad: GND1 S
Pad: PWR1 S
Pad: io_Filter_Out_7 S

Pad: io_Filter_Out_6 S
Pad: io_Filter_Out_5 S
Pad: io_Filter_Out_4 S
Pad: io_Filter_Out_3 S

Pad: io_Filter_Out_2 W
Pad: io_Filter_Out_1 W
Pad: io_Filter_Out_0 W
Pad: io_ADC_eoc W
Pad: io_ADC_convst W
Pad: io_ADC_rd W
Pad: io_ADC_cs W
Pad: io_DAC_wr W
Pad: io_DAC_cs W
Pad: io_DAC_ldac W
Pad: io_DAC_clr W