

# Travaux pratiques de conception de circuit VLSI numérique

*Conception d'un filtre numérique à réponse  
impulsionnelle finie*



10 Septembre 2015

# Sommaire

<b>I) Démarrage.....</b>	<b><u>1</u></b>
I.1) Connexion.....	<u>1</u>
I.2) Structure des répertoires.....	<u>1</u>
I.3) Avant de commencer à travailler.....	<u>2</u>
<b>II) Présentation et spécification du filtre numérique.....</b>	<b><u>2</u></b>
II.1) Étude et spécification du filtre numérique.....	<u>2</u>
II.2) Travail demandé lors de la partie conception du filtre RIF (2 séances).....	<u>5</u>
II.3) Travail demandé lors de la partie validation RTL du filtre RIF (2 séances).....	<u>5</u>
<b>III) Réalisation du filtre sur FPGA.....</b>	<b><u>6</u></b>
III.1) Synthèse sur FPGA de la description VHDL avec l'outil Precision de Mentor Graphics (1 séance).....	<u>6</u>
III.1.1) Lancer l'outil Précision.....	<u>6</u>
III.1.2) Choix de la technologie cible.....	<u>6</u>
III.1.3) Définir les modèles à synthétiser.....	<u>7</u>
III.1.4) Options.....	<u>7</u>
III.1.5) Spécifier le format de sortie du résultat de synthèse.....	<u>7</u>
III.1.6) Réalisation de la synthèse.....	<u>7</u>
III.1.7) Visualisation du résultat de la synthèse.....	<u>7</u>
III.1.8) Enregistrement de la synthèse.....	<u>7</u>
III.1.9) Simulation après synthèse.....	<u>8</u>
III.2) Placement et routage cible FPGA avec l'outil ISE de Xilinx(1 séance).....	<u>8</u>
III.2.1) Le fichier des contraintes de l'utilisateur.....	<u>8</u>
III.2.2) Lancement de l'outil ISE et paramétrage du projet.....	<u>8</u>
III.2.3) Placement et routage.....	<u>10</u>
III.2.4) Génération du fichier de configuration du FPGA.....	<u>10</u>
III.2.5) Simulation après le placement et le routage.....	<u>10</u>
<b>IV) Travail demandé lors des étapes de synthèse, placement routage et test du filtre sur cible     FPGA.....</b>	<b><u>11</u></b>
IV.1) Programmation de la carte FPGA Xilinx Spartan3 et test en environnement réel (1 séance).....	<u>11</u>
IV.1.1) Configuration du FPGA.....	<u>11</u>
IV.1.2) Test du circuit implanté sur FPGA.....	<u>12</u>
IV.2) Travaux à réaliser et questions à traiter.....	<u>13</u>
<b>V) Réalisation du filtre numérique sur Asic (1 séance).....</b>	<b><u>1</u></b>
<b>VI) Synthèse logique sur cible ASIC, validation et analyse des résultats (1 Séance).....</b>	<b><u>2</u></b>
VI.1) Synthèse logique avec l'outil Design Compiler de Synopsys.....	<u>2</u>
VI.1.1) Paramétrage et lancement de l'outil de synthèse.....	<u>2</u>
VI.1.2) Menus de l'interface Design Vision.....	<u>3</u>
VI.2) Validation après la synthèse.....	<u>5</u>
VI.3) Travail demandé : analyse des résultats de synthèse.....	<u>5</u>
<b>VII) Étude d'une bibliothèque d'un fondeur (2 séances).....</b>	<b><u>6</u></b>
VII.1) Introduction aux bibliothèques.....	<u>6</u>
VII.2) Le logiciel Virtuoso de Cadence.....	<u>6</u>
VII.3) Aide à la lecture du dessin des masques (layout).....	<u>8</u>
VII.3.1) Le transistor NMOS.....	<u>8</u>
VII.3.2) Le transistor PMOS.....	<u>9</u>
VII.3.3) Autres informations.....	<u>9</u>
<b>VIII) Placement et routage ciblés ASIC avec l'outil SoC Encounter de Cadence (2 séances).....</b>	<b><u>10</u></b>
VIII.1) Configuration de SoC Encounter.....	<u>11</u>

VIII.2) Démarrage de SoC Encounter.....	<a href="#"><u>12</u></a>
VIII.3) Chargement des scripts du fondeur.....	<a href="#"><u>12</u></a>
VIII.4) Importation du Design.....	<a href="#"><u>12</u></a>
VIII.5) Définition des surfaces de placement-routage (Floorplan).....	<a href="#"><u>16</u></a>
VIII.5.1) Couronne de plots.....	<a href="#"><u>16</u></a>
VIII.5.2) Alimentation du circuit.....	<a href="#"><u>16</u></a>
VIII.6) Placement des cellules composant le cœur.....	<a href="#"><u>23</u></a>
VIII.7) Génération de l'arbre d'horloge.....	<a href="#"><u>24</u></a>
VIII.7.1) Synthèse de l'arbre d'horloge.....	<a href="#"><u>24</u></a>
VIII.7.2) Visualisation de l'arbre d'horloge.....	<a href="#"><u>24</u></a>
VIII.8) Remplissage des vides.....	<a href="#"><u>25</u></a>
VIII.9) Routage effectif.....	<a href="#"><u>25</u></a>
VIII.10) Vérification.....	<a href="#"><u>26</u></a>
VIII.11) Caractéristiques du circuit après P&R cible ASIC.....	<a href="#"><u>26</u></a>
VIII.12) Exportations des données de placement et routage.....	<a href="#"><u>27</u></a>
VIII.13) Simulation après placement et routage.....	<a href="#"><u>28</u></a>
VIII.14) Travail demandé pour la partie placement et routage cible ASIC.....	<a href="#"><u>28</u></a>

## I) Démarrage

---

### I.1) Connexion

#### Première connexion

Utilisez le mot de passe par défaut pour vous connecter au compte xph2appxxx où xxx désigne votre numéro de compte.

Ouvrez un terminal puis utilisez la commande yppasswd pour changer votre mot de passe.

#### Connexion ultérieures

Connectez vous au compte xph2appxxx avec votre mot de passe personnalisé

### I.2) Structure des répertoires

La structure des répertoires de ce TP est donnée dans le tableau ci-dessous. Vous aurez à vous reporter à ce tableau tout au long du TP pour savoir où trouver les données ou pour vous situer au bon endroit lors de l'exécution de certaines commandes.

Tableau 1 : Structure des répertoires pour le TP filtre numérique

Nom du répertoire	Description
asic	Dédié à la cible ASIC
asic/par	Dédié au placement et routage sur ASIC (Encounter™)
asic/synth	Dédié à la synthèse VHDL pour cible ASIC (Design Vision™)
bench	Contient les environnements de simulations et les programmes de tests
cadence	Dédié à l'étude et à la création de la cellule standard
config	Contient les scripts de configuration et les fichiers d'initialisation
doc	Contient la documentation relative au TP
fpga	Dédié à la cible FPGA
fpga/userconstraints	Contient le fichier des contraintes de placement et d'affectation des broches du FPGA
fpga/par	Dédié au placement et routage sur FPGA (Xilinx ISE™)
fpga/synth	Dédié à la synthèse VHDL
libs	Contiendra les bibliothèques VHDL compilées par vous même lib_VHDL : bibliothèque comportementale (avant synthèse) lib_BENCH : l'environnement de test lib_FPGA_SYNT : bibliothèque de compilation du fichier de synthèse sur FPGA lib_FPGA_PAR : bibliothèque de compilation pour le placement et routage pour le FPGA Note : les répertoires correspondant apparaitront lors de la création de ces bibliothèques
SpyGlass	Dédié à la vérification du code VHDL avec l'outil Lint
vhd	Contient les sources VHDL

### I.3) Avant de commencer à travailler

La première chose à faire avant de commencer à travailler est de sourcer<sup>1</sup> le script de configuration correspondant à l'étape du TP dans laquelle vous vous trouvez :

- config\_RTL : configuration pour la simulation comportementale au niveau RTL
- config\_FPGA : configuration une implantation sur FPGA

Tous ces fichiers de configuration ont été placés dans le sous-répertoire "config". Ils définissent tous une variable d'environnement appelée TP\_PATH qui transporte le chemin absolu de la base de l'arborescence du TP. Vous pouvez utiliser cette variable d'environnement pour exécuter vos commandes ou écrire vos scripts.

## II) Présentation et spécification du filtre numérique

Ce TP propose d'étudier et de réaliser un filtre numérique à réponse impulsionnelle finie à 32 coefficients. Ce filtre sera conçu et implanté sur cible physique FPGA et ASIC. Ceci permettra la prise en main de toutes les étapes de la conception et la validation de circuits numériques.

### II.1) Étude et spécification du filtre numérique

Une étude sous Matlab a permis de spécifier la réponse fréquentielle voulue. Les courbes obtenues sont reproduites aux figures 1 et 2. Les trente-deux coefficients correspondant sont donnés au tableau 2.

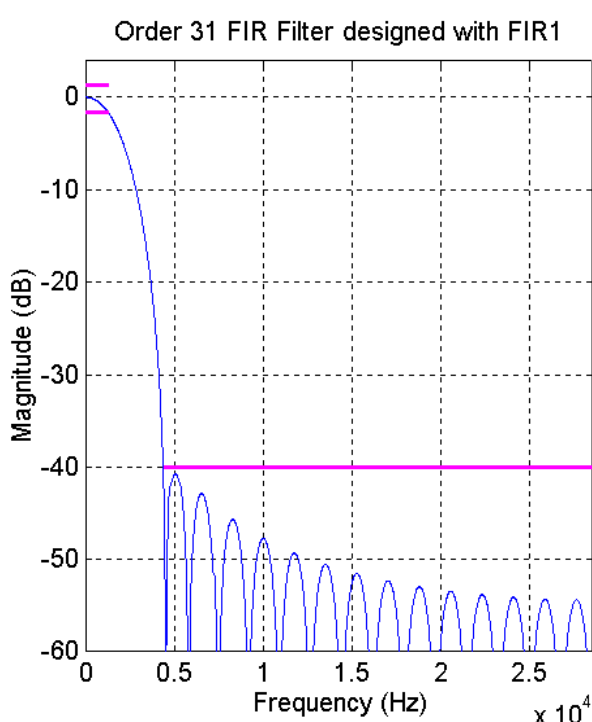


Fig 1 : Réponse fréquentielle du filtre numérique à 32 coefficients

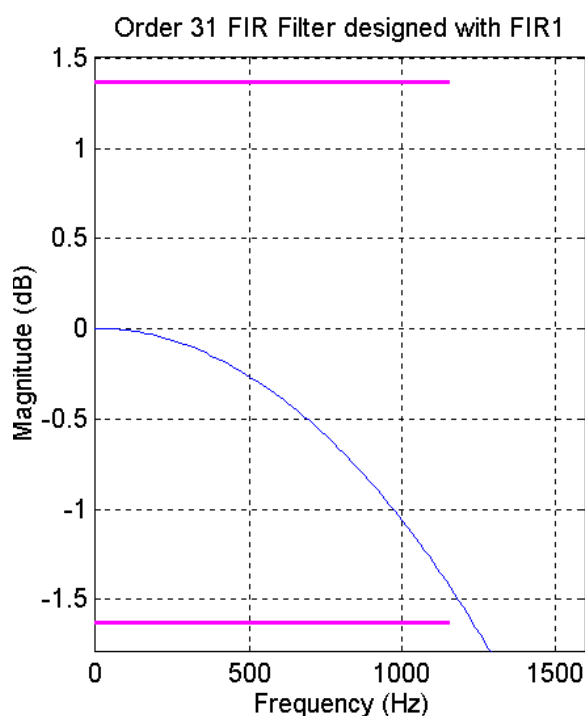


Fig 2 : Réponse fréquentielle du filtre numérique à 32 coefficients (zoom)

<sup>1</sup> Dans le contexte d'une utilisation des outils de CAO dans un environnement Linux, l'expression « sourcer un script » désigne (par contamination du nom de la commande) le lancement du script à l'aide de la commande Linux `source` plutôt que par une exécution simple sur la ligne de commande.

Tableau 2 : Liste des 32 coefficients du filtre numérique

Binaire	Hexadécimale	Décimale arrondie	Décimale	Erreur
0 0 0 0 1 1 0 1	0D	0.0508	0.0512	0.0004
0 0 0 1 0 1 0 1	15	0.0820	0.0832	0.0012
0 0 0 1 1 1 1 1	1F	0.1211	0.1245	0.0034
0 0 1 0 1 1 0 0	2C	0.1719	0.1754	0.0035
0 0 1 1 1 1 0 0	3C	0.2344	0.2355	0.0011
0 1 0 0 1 1 0 1	4D	0.3008	0.3039	0.0031
0 1 1 0 0 0 0 1	61	0.3789	0.3791	0.0002
0 1 1 1 0 1 0 1	75	0.4570	0.4591	0.0020
1 0 0 0 1 0 1 0	8A	0.5391	0.5412	0.0021
1 0 0 1 1 1 1 1	9F	0.6211	0.6226	0.0015
1 0 1 1 0 0 1 1	B3	0.6992	0.7001	0.0009
1 1 0 0 0 1 0 1	C5	0.7695	0.7707	0.0012
1 1 0 1 0 1 0 0	D4	0.8281	0.8314	0.0033
1 1 1 0 0 0 0 1	E1	0.8789	0.8795	0.0006
1 1 1 0 1 0 0 1	E9	0.9102	0.9128	0.0026
1 1 1 0 1 1 1 0	EE	0.9297	0.9298	0.0001
1 1 1 0 1 1 1 0	EE	0.9297	0.9298	0.0001
1 1 1 0 1 0 0 1	E9	0.9102	0.9128	0.0026
1 1 1 0 0 0 0 1	E1	0.8789	0.8795	0.0006
1 1 0 1 0 1 0 0	D4	0.8281	0.8314	0.0033
1 1 0 0 0 1 0 1	C5	0.7695	0.7707	0.0012
1 0 1 1 0 0 1 1	B3	0.6992	0.7001	0.0009
1 0 0 1 1 1 1 1	9F	0.6211	0.6226	0.0015
1 0 0 0 1 0 1 0	8A	0.5391	0.5412	0.0021
0 1 1 1 0 1 0 1	75	0.4570	0.4591	0.0020
0 1 1 0 0 0 0 1	61	0.3789	0.3791	0.0002
0 1 0 0 1 1 0 1	4D	0.3008	0.3039	0.0031
0 0 1 1 1 1 0 0	3C	0.2344	0.2355	0.0011
0 0 1 0 1 1 0 0	2C	0.1719	0.1754	0.0035
0 0 0 1 1 1 1 1	1F	0.1211	0.1245	0.0034
0 0 0 1 0 1 0 1	15	0.0820	0.0832	0.0012
0 0 0 0 1 1 0 1	0D	0.0508	0.0512	0.0004

L'expression analytique du filtrage, en notant  $a(i)$  le  $i^{\text{ème}}$  coefficient et  $x(n)$  le  $n^{\text{ème}}$  échantillon du signal d'entrée :

$$Out(n) = \sum_{i=0}^{N-1} a(i) \cdot x(n-i)$$

L'architecture canonique est donnée à la figure 3, on observe que celle-ci est trop coûteuse car le nombre d'opérateurs arithmétiques est trop important. C'est pourquoi, l'architecture optimisée de la figure 4 lui est préférée.

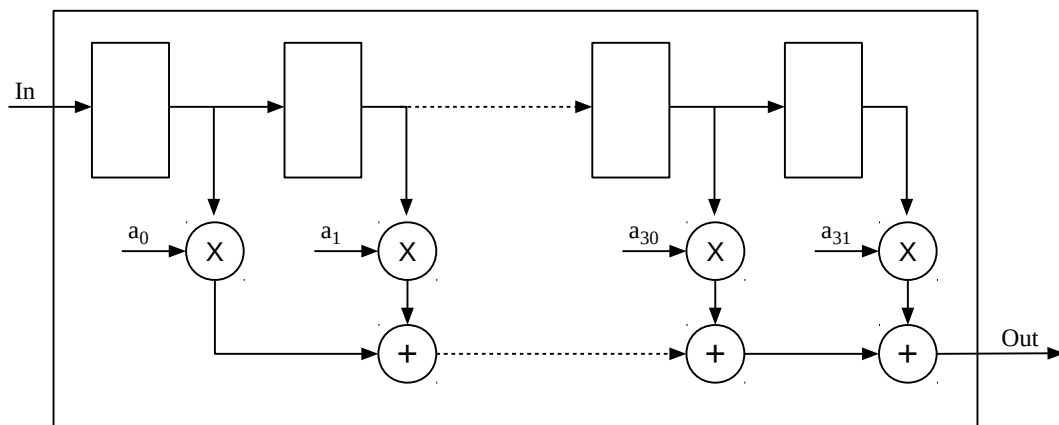
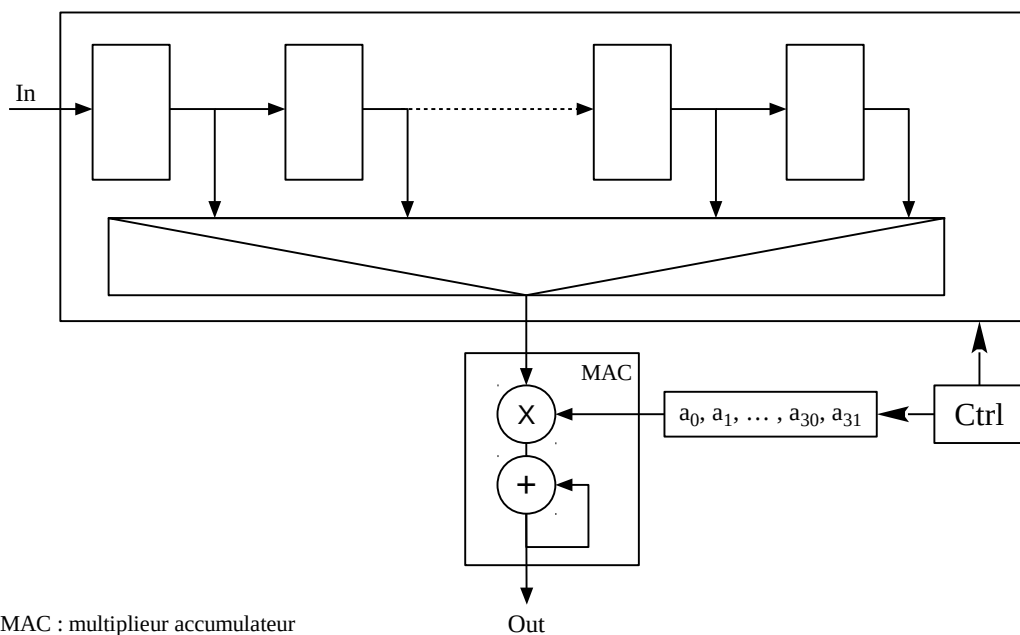


Fig 3 : Architecture canonique d'un filtre numérique RIF



MAC : multiplieur accumulateur

Fig 4 : Architecture optimisée en taille et en nombre d'opérations pour un filtre numérique RIF

Cette architecture est détaillée dans la figure 5. Sa description en VHDL est consignée par l'ensemble des fichiers du répertoire vhd.

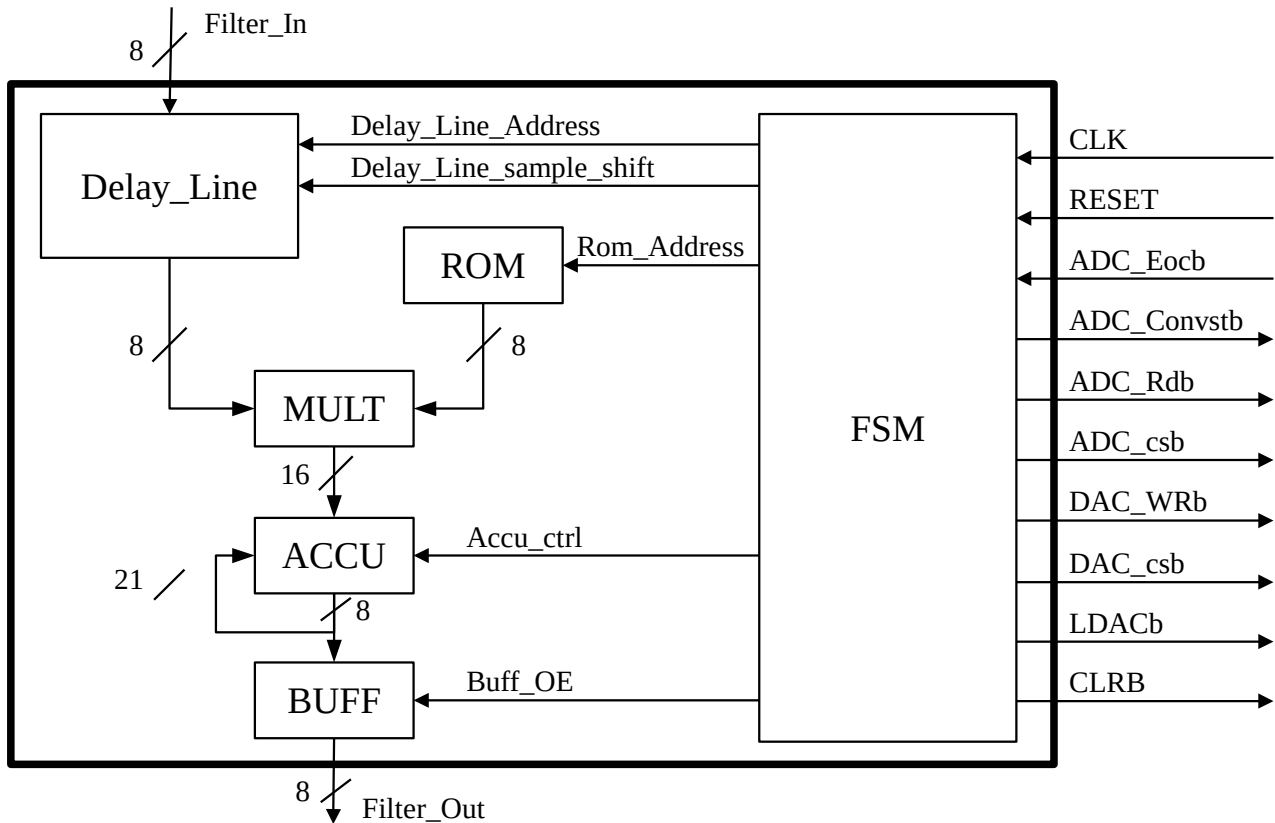


Fig 5 : Architecture détaillée de filtre RIF

## II.2) Travail demandé lors de la partie conception du filtre RIF (2 séances)

- Comprendre et commenter l'ensemble du code source donné dans les répertoires "vhd" et "bench".
- Faire un chronogramme à la main spécifiant le fonctionnement du système.
- Extraire la machine à états finis correspondant au chronogramme. La dessiner sous forme de graphe de transitions d'états.
- Compléter le fichier "vhd/fsm.vhd". Il s'agit d'écrire en VHDL l'architecture associée à l'entité fsm du contrôleur du filtre. Le contrôleur doit être tel que le filtre puisse s'interfacer avec le système de conversion analogique/numérique et numérique/analogique (cf. annexe).

## II.3) Travail demandé lors de la partie validation RTL du filtre RIF (2 séances)

- A l'aide de l'outil d'Atrenta SpyGlass™ vérifier le vhd du fichier fsm et de l'ensemble des fichiers VHDL en utilisant le module «Lint» (voir le Tutorial Spyglass dans le répertoire doc)
- Modifier et compléter le programme de test donné.

Le fichier de test "bench/bench.vhd" simule le fonctionnement des convertisseurs. Il fournit des signaux en entrée du filtre et observe ceux en sortie en tenant compte des caractéristiques fonctionnelles et temporelles des signaux allant ou provenant des convertisseurs (cf. documentation technique en annexe).

Le programme de test que vous avez à écrire, devra vérifier que le filtre est fonctionnellement correct. Vous devez appliquer les stimulus appropriés de façon à examiner la réponse impulsionnelle, la réponse indicielle ou la réponse à une somme de sinusoides. Dans un premier temps, il faudra modéliser les convertisseurs et vérifier leur bon comportement temporel à l'aide de clauses assert. Puis, il faudra fournir les stimulus appropriés pour tester (toujours à l'aide de clauses assert) les différentes réponses. Expliquez votre démarche.



- c) Simuler le filtre complet dans l'environnement de test développé pour vérifier son bon fonctionnement. Corriger toutes les erreurs de conception en plus des erreurs de syntaxe. Argumenter avec précision. Pourquoi le fonctionnement du filtre est correct ?

Pour effectuer des simulations comportementales avec le logiciel ModelSim, vous devez au préalable sourcer le script de configuration `config_RTL`. Le mini guide fourni sur ModelSim donne les principales commandes à connaître pour utiliser ce logiciel. N'hésitez pas à consulter aussi l'aide du logiciel (le mini guide indique où trouver cette aide).

Un compte rendu synthétique (compte rendu nr.1), décrivant l'ensemble de vos travaux, est à rendre en fin de cette étape de conception / validation RTL. Fournir le fichier `fsm.vhdl` et le programme de test (`bench`).

### III) Réalisation du filtre sur FPGA

---

L'objectif de cette partie de TP est de se familiariser avec le flot de synthèse et validation d'un circuit numérique sur cible FPGA. Pour cette partie vous devez sourcer le fichier de configuration `config_FPGA`.

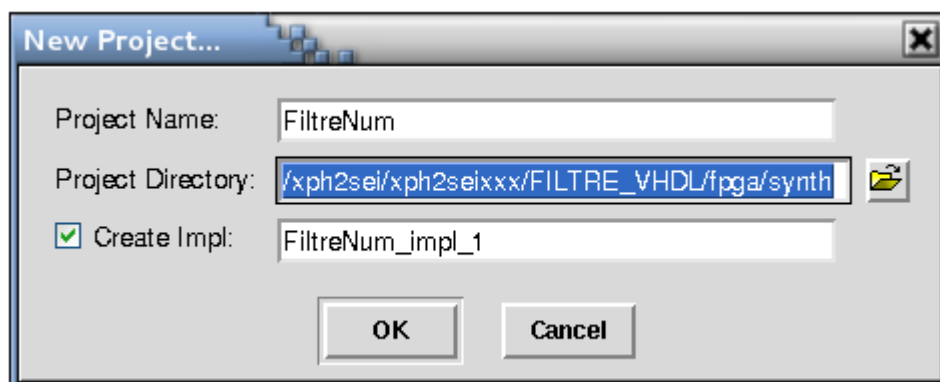
#### III.1) Synthèse sur FPGA de la description VHDL avec l'outil Precision de Mentor Graphics (1 seance)

##### III.1.1) Lancer l'outil Précision

- Sourcer le script de configuration `config_FPGA` et assurer vous que vous êtes dans le répertoire `${TP_PATH}/fpga/synth`
- Démarrer l'outil en exécutant la commande `precision &`
- Créer un nouveau projet via le menu « Project | New Project »
  - Donner le nom de votre choix à votre projet, par exemple « FiltreNum », en renseignant la zone de saisie « Project Name ».
  - Assurer vous que le champ « Project Directory » désigne bien le répertoire `${TP_PATH}/fpga/synth`.
  - La zone de saisie « Create Impl » donne le nom du répertoire qui contiendra la description du circuit synthétisé. Le nom par défaut est le nom du projet (champ « Project Name ») suivie de « `impl_1` » (adapter ce nom si besoin est). La case à cocher doit être cochée pour que le répertoire soit créé.

- Clicker OK.

Le sous répertoire destiné à contenir la description du circuit synthétisé est alors créé.



##### III.1.2) Choix de la technologie cible

A l'aide de l'icône « Setup Design », ouvrir la fenêtre « Project Settings ». Il peut être nécessaire de cliquer l'onglet « Design » dans la colonne de gauche pour voir cette icône. Choisir :

- Technology : Xilinx Spartan3
- Device : 3s200ft256
- Speed Grade : -5

Cocher Set Frequency et donner la fréquence de cadencement de la carte cible (50 MHz). Laisser les autres options par défaut.

Valider (OK) et sauvegarder le projet.

### III.1.3) Définir les modèles à synthétiser

L'icône « Add Input Files » (colonne de gauche onglet « Design »), permet de spécifier les fichiers qui contiennent les entités et les architectures à synthétiser.

Se placer dans `${TP_PATH}/vhd` et choisir tous les fichiers source du filtre (fichiers \*.vhd).

**ATTENTION** : Dans Precision, le fichier contenant le module englobant (*top module*) du projet est mis en gras et en bas de la liste « Input Files » dans la zone « Project Files ». Dans ce projet, le module englobant est contenu dans le fichier `filter.vhd`. Il faut donc le désigner comme tel. Pour cela, sélectionner le fichier `filter.vhd` cliquer droit et sélectionner « Move File to Bottom ».

### III.1.4) Options

Il est possible d'optimiser la synthèse, soit en minimisant la surface, soit en améliorant les performances temporelles. Pour accéder aux options d'optimisation sélectionner le menu « Tools | Options | Optimization Settings ».

### III.1.5) Spécifier le format de sortie du résultat de synthèse

Le menu « Tools | Options | Output Options » permet de définir le format de sortie de la liste de porte (*netlist*) obtenue après synthèse ainsi que le nom du fichier dans lequel la synthèse sera enregistrée.

Spécifier le nom du fichier de sauvegarde dans le champ « Output File Base Name » : `filtre_synth`.

Sélectionner les formats EDIF et VHDL.

Décocher l'option « Generate Vendor Constraint File ».

Avec ces choix la synthèse produira deux fichiers :

- `filtre_synth.edf` nécessaire à l'étape de placement et routage qui suit
- `filtre_synth.vhd` destiné à la simulation après synthèse

Ces fichiers seront enregistrés dans le sous répertoire `FiltreNum_impl_1` selon notre exemple.

### III.1.6) Réalisation de la synthèse

La synthèse peut être lancée à partir des icônes « Compile », pour la première étape, et « Synthesize » pour la synthèse finale. Il peut être nécessaire de cliquer l'onglet « Design » dans la colonne de gauche pour voir ces icônes.

Le compte-rendu de ces opérations apparaît dans l'onglet « Transcript ».

### III.1.7) Visualisation du résultat de la synthèse

Les résultats de la synthèse sont visualisable par le biais des icônes de l'onglet « Schematics » dans la colonne de gauche :

- L'icône « View RTL Schematic » permet de visualiser la vue RTL de la modélisation VHDL.
- L'icône « View Tech Schematic » permet de visualiser le schéma de portes obtenue avec la bibliothèque de portes logiques de la technologie ciblée.
- L'icône « View Critical Path » permet de visualiser le chemin critique dans le circuit synthétisé.

Pour connaître les statistiques de performances du circuit synthétisé, cliquer sur l'icône « Area report » pour le taux d'occupation du FPGA et sur l'icône « Report Timing » pour les timings. Ces icônes deviennent visibles en cliquant sur l'onglet « Design Analysis » dans la colonne de gauche

### III.1.8) Enregistrement de la synthèse

Le résultat de la synthèse est conservé dans le fichier `filtre_synth.edf`.

Vérifiez que les fichiers `filtre_synth.edf` et `filtre_synth.vhd` sont bien présents dans le répertoire créé pour la synthèse ( `FiltreNum_impl_1` dans notre exemple).

Procéder ensuite à l'enregistrement du projet en sélectionnant « Project | Save Project » avant de quitter l'outil Precision.

### III.1.9) Simulation après synthèse

En vous inspirant du script de compilation « `compil_VHDL` », créez un nouveau script pour la compilation du fichier VHDL du circuit synthétisé. Pensez à mettre à jour vos bibliothèques dans le bench. Valider le bon fonctionnement du filtre synthétisé.

## III.2) Placement et routage cible FPGA avec l'outil ISE de Xilinx(1 séance)

L'objectif de cette étape du flot de conception cible FPGA est d'implanter physiquement le filtre sur la Spartan 3 de Xilinx.

### III.2.1) Le fichier des contraintes de l'utilisateur

Même si un FPGA est une technologie très flexible, il n'est pas possible d'affecter n'importe quel signal à n'importe quelle broche. En effet, les broches sont regroupées en ports qui peuvent être associés à des fonctions spéciales comme un port sériel, un port JTAG ou un décodeur pour un afficheur. Même dans le cas le plus courant d'entrées/sorties tout ou rien à usage général, deux ports différents n'acceptent pas forcément la même plage de tension (par exemple un domaine en 0-5V et un autre en 0-3V), la sortance peut être différente et certaines terminaisons sont soit en entrée soit en sortie alors que d'autres sont bidirectionnelles. En outre, certaines broches sont dédiés aux alimentations et au moins une est réservée pour le signal d'horloge. Lorsque le FPGA est déjà câblé sur une carte électronique prête à l'emploi, comme celle utilisée dans ce TP, les contraintes sur l'affectation des ports et des broches sont encore plus importantes. Par exemple, une entrée tout ou rien initialement bidirectionnelle sur le FPGA vide, peut, une fois le FPGA câblé, être désormais en entrée uniquement (cas d'un bouton poussoir) ou en sortie seulement (cas d'une LED).

Il faut donc renseigner l'outil de placement et routage sur la correspondance qu'il doit respecter entre les ports physiques du FPGA et les ports formels définis par l'entité du module englobant dans la description matérielle. Il faudra également spécifier la direction pour les ports bidirectionnels, la plage de tension, la sortance, etc. Toutes ces informations sont consignées dans le fichier des contraintes de l'utilisateur (*User Constraint File* – UCF).

Dans le cas de ce TP, le fichier des contraintes de l'utilisateur a été préparé pour vous et placé dans le répertoire `${TP_PATH}/fpga/userconstraints`. Regarder et comprendre ces contraintes sans modifier ce fichier !

### III.2.2) Lancement de l'outil ISE et paramétrage du projet

**Rappel :** vous devez exécuter le script de configuration `config_FPGA` avant de pouvoir utiliser l'outil ISE.

- Se placer dans le répertoire `${TP_PATH}/fpga/par` et démarrer l'outil en exécutant la commande **ise &**
- Sélectionner le menu « Project | New Project »
- Dans la fenêtre « New Project Wizard – Create New Project », renseigner
  - le nom du projet pour le placement et le routage, par exemple « FiltreNum »
  - l'emplacement du répertoire pour le placement et le routage le « Working Directory ». Indiquer le chemin `${TP_PATH}/fpga/par`
  - sélectionner « EDIF » dans la liste déroulante « Top-level source type »
  - cliquer « Next »
- Dans la fenêtre « New Project Wizard – Import EDIF/NGC Project », indiquer
  - le fichier `edf` créer lors de la synthèse (cf. § 7), dans la zone de saisie « Input Design »
  - le fichier `${TP_PATH}/fpga/userconstraints/filtre.ucf` dans la zone de saisie « Constraint file »

- Décocher la case « Copy the constraints file to the project directory » de façon à ce que le fichier des contraintes de l'utilisateur \*.ucf ne soit pas dupliqué.
- cliquer « Next »
- Dans la fenêtre « New Project Wizard – Project Settings », les caractéristiques du FPGA (Family, Device, Package, Speed) doivent être automatiquement remplies.  
S'assurer que le champ « Preferred language » est positionné sur VHDL.
- Cliquer « Next » puis « Finish »

### III.2.3) Placement et routage

Les commandes de placement et routage sont regroupées dans le panneau de gauche « Processes » au sein d'une arborescence appelée « Implement Design » (voir Fig 6).

Les commandes de placement et routage donnent lieu à des rapports qui sont accessibles en dépliant l'arborescence.

Le rapport général (Design Overview ==> Summary) qui peut être visualisé par l'onglet « Design Summary » dans le panneau de droite, existe aussi sous forme d'un fichier HTML (FILTER\_summary.html) dans le répertoire du projet de placement routage. Il est ainsi possible de consulter ce rapport sans devoir relancer l'outil.

#### Génération des modèles de simulation temporelle

Double-cliquer sur la ligne « Generate Post-place & Route Simulation Model » dans le panneau « Processes ». Dans le Process Properties modifiez le model de simulation en VHDL, par défaut il est en Verilog.

#### Emplacement des fichiers produits pour la simulation temporelle

En supposant que l'entité du module englobant (*top module*) ait été appelée « FILTER » et que le nom du projet pour le placement et le routage soit « FiltreNum », les fichiers de timing nécessaires à la simulation après placement et routage sont automatiquement baptisés FILTER\_timesim.sdf et FILTER\_timesim.vhd. Ils sont générés dans le sous répertoire :

```
${TP_PATH}/fpga/par/FiltreNum/netgen/par/
```

### III.2.4) Génération du fichier de configuration du FPGA

Cette opération va créer un fichier qui sera chargé dans le FPGA pour changer sa configuration de façon à refléter le circuit que vous venez de placer et router. Ce type de fichier est appelé *bitstream* en anglais.

Double-cliquer sur la ligne « Generate Programming File » dans le panneau « Processes ».

### III.2.5) Simulation après le placement et le routage

Créez un nouveau script pour la compilation du circuit placé routé avec Modelsim. Le fichier VHDL à compiler est celui créé à l'étape 9 (le bench de la simulation après synthèse).

Le paquetage VITAL pour le FPGA utilisé dans ce projet a été préalablement précompilé. Il est introduit dans le projet via la bibliothèque logique *simprim* qui est définie dans le fichier *modelsim.ini*.

Pour que la simulation après placement et routage prenne tout son sens, elle doit être rétroannotée avec les informations temporelles contenues dans le fichier « sdf » généré lors du placement et routage (cf. § III.2.3). Pour cela il faut charger le fichier de rétroannotation dans le simulateur Modelsim. Ceci peut être à partir de l'interface graphique ou bien au démarrage sur la ligne de commande.

#### i) Chargement du fichier de rétroannotation sdf par l'interface graphique de Modelsim

- Dans ModelSim, sélectionnez le menu « Simulate | Start Simulation... ».

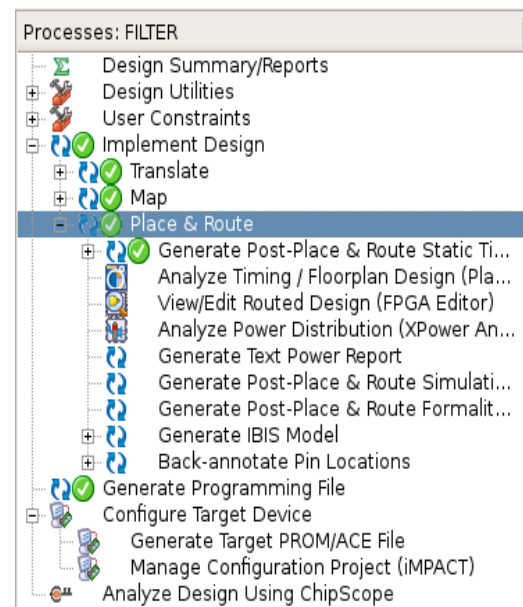


Fig 6 : Génération des modèles de simulation temporelle

- Dans l'onglet « Design » de la fenêtre « Start Simulation » qui vient d'apparaître, sélectionnez votre bibliothèque de test, votre entité de test et l'architecture de test que vous voulez utiliser. Vous noterez que la zone « Design Unit » se remplit automatiquement avec les éléments sélectionnés dans la liste. Vous pouvez directement écrire dans cette zone à condition de respecter la syntaxe  
`Bibliothèque.Entité(architecture)`
- La liste « Resolution » sert à sélectionner une résolution temporelle raisonnable avec la période des signaux à analyser. Inutile d'exiger la picoseconde avec une période d'horloge de 20 ns !
- Dans la zone « Optimization », cocher « Enable Optimization » et cliquer le bouton « Optimization Options... »
- Dans l'onglet « Visibility » de la fenêtre « Optimization Options », sélectionner « Add full visibility to all modules » et cliquer OK.
- Dans l'onglet « SDF » de la fenêtre « Start Simulation », zone « SDF Files », cliquez le bouton « Add »
- Dans la fenêtre « Add SDF Entry » qui est apparue, renseignez le nom complet du fichier SDF à ajouter. C'est le fichier généré lors du placement et routage (cf. § III.2.3).
- De retour dans l'onglet « SDF » de la fenêtre « Start Simulation », cochez les cases « Disable SDF Warnings » et « Reduce SDF Errors to Warnings » de la zone « SDF Options »

#### ii) Chargement du fichier de rétroannotation par la ligne de commande

La ligne de commande à saisir doit être de la forme

`vsim [options] Bibliothèque.Entité(architecture)`

Les options à utiliser ici sont les suivantes :

- sdf ttp : pour indiquer quelle instance de quelle entité doit être simulée avec quel fichier SDF, suivant le format  
`/Entité/Instance={fichier SDF}`  
ex : pour simuler une instance U1 d'une entité MonBench avec le fichier FILTER\_timesim.sdf du répertoire `${TP_PATH}/{répertoire des SDF}/`, la commande sera  
`vsim -sdf ttp /MonBench/U1=${TP_PATH}/{répertoire des SDF}/FILTER_timesim.sdf`
- sdfnoerror : pour désactiver les erreurs SDF
- sdfnowarn : pour désactiver les mises en garde SDF
- voptargs=+acc : pour donner une visibilité complète à tous les signaux de tous les modules

#### iii) Simulation

La simulation s'effectue de la même façon que pour les simulations avant et après synthèse.

Dès que la simulation du filtre placé routé donne satisfaction, vous pouvez passer à l'implantation sur la carte FPGA.

## IV) Travail demandé lors des étapes de synthèse, placement routage et test du filtre sur cible FPGA

---

### IV.1) Programmation de la carte FPGA Xilinx Spartan3 et test en environnement réel (1 séance)

Se reporter aux annexes (voir répertoire Doc) pour connaître l'emplacement et le branchement des connecteurs sur le banc de test.

#### IV.1.1) Configuration du FPGA

L'outil de programmation des FPGA Xilinx s'appelle iMPACT.

Pour charger un fichier de configuration « bitstream » dans un FPGA Xilinx, procéder comme suit :

- Mettre sous tension la carte *Spartan 3 Starter Board*.
- Dans une fenêtre terminal, taper la commande '**impact**'.
- Valider par « OK », le message d'information disant que le répertoire de travail a été changé et a été positionné sur le répertoire courant.
- Dans la fenêtre « iMPACT Project », sélectionner « create a new project » et désigner le répertoire où créer le projet ( $\{\text{TP\_PATH}\}/\text{fpga}/\text{par}$ ) grâce au bouton « Browse ». Cliquer « OK ».
- Dans la fenêtre « iMPACT – Welcome to iMPACT », s'assurer que « Configure devices using Boundary-Scan (JTAG) » est sélectionné et que la liste déroulante est positionnée sur « Automatically connect to a cable and identify Boundary -Scan chain » puis cliquer « Finish ».  
Si le message « Warning:iMPACT:923 – cable not found » apparaît, c'est que le port JTAG n'est pas connecté au port USB du PC (attention au sens du câble) ou que la carte FPGA n'est pas sous tension ou que le pilote USB n'est pas chargé (cf. ci-dessus les scripts à sourcer).
- Une fenêtre « Assign New Configuration File » apparaît pour indiquer la configuration du composant xc3s200 : c'est le FPGA. Parcourir l'arborescence pour retrouver le fichier de configuration généré au §III.2.4 (*bitstream* d'extension \*.bit ). Sélectionner ce fichier puis cliquer « Open ».
- La fenêtre « Assign New Configuration File » réapparaît pour la configuration d'un composant xcf02s : c'est une EEPROM présente sur la carte FPGA mais qui n'est pas utilisée dans ce TP. Continuer en cliquant « Bypass » puis valider par OK le message d'information suivant.
- Dans l'onglet « Boundary Scan » de la fenêtre d'iMPACT (cf. Fig 7), faire un clic droit sur l'icône du FPGA xc3s200, puis sélectionner « Program »
- Le FPGA est reconfiguré en une dizaine de secondes.

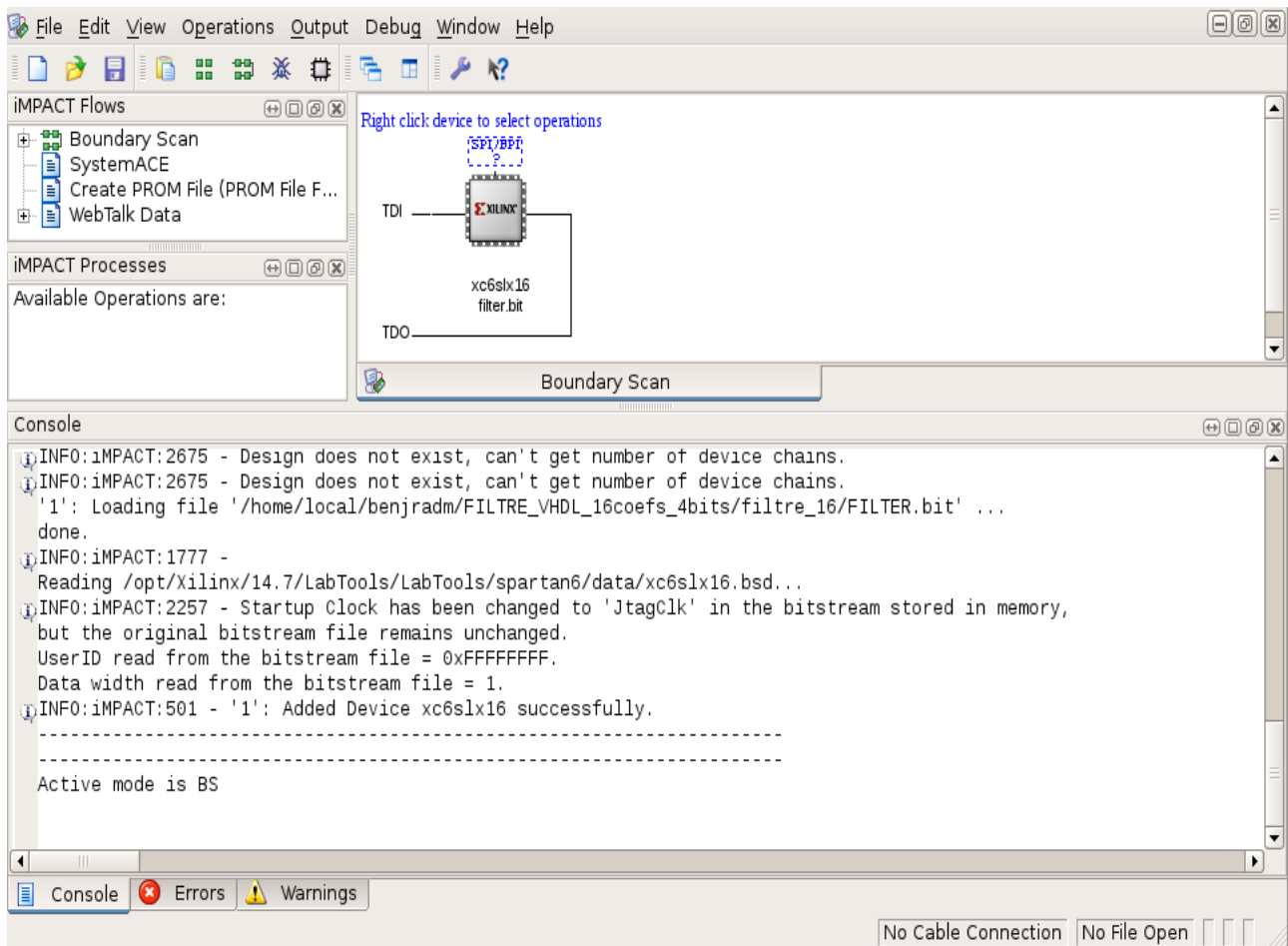


Fig 7: La fenêtre de l'outil iMPACT

#### IV.1.2) Test du circuit implanté sur FPGA

En vous aidant du schéma d'implantation simplifié et de la photo du banc de test matériel (cf. § IV des annexes), assurez vous que le générateur basse fréquence (Géné BF) est bien connecté à l'entrée AnaIn de la chaîne A et renvoyée sur une des entrées de l'oscilloscope. Vérifiez que la sortie AnaOut de la chaîne A est reliée à l'oscilloscope. Visualisez les signaux de contrôles du CAN et du CNA de la chaîne A.

A l'aide du Géné BF et de l'oscilloscope, contrôlez le fonctionnement du filtre numérique.

Afin de lever les doutes sur la chaîne de numérisation, en cas de non fonctionnement de votre circuit, les fichiers de configuration de référence suivants sont mis à votre disposition dans le répertoire /softslin/configCA0/Xilinx :

- le fichier test\_sans\_filtre\_spartan3.bit valide le fonctionnement de la chaîne de numérisation sans filtrage
- le fichiers valid\_cna\_spartan3.bit valide le fonctionnement du convertisseur numérique vers analogique

**ATTENTION** : s'il n'est pas possible de faire fonctionner le banc de test avec les fichiers de configuration de référence indiqués ci-dessus, réinitialiser la carte FPGA en débranchant le connecteur d'alimentation, le port JTAG et le Géné BF.

#### IV.2) Travaux à réaliser et questions à traiter

- Réaliser la synthèse du filtre à partir de l'ensemble du code source du répertoire vhd sur la technologie Xilinx Spartan3.
- A quoi correspond l'opération « compile » dans l'outil Precision ?
- Qu'effectue l'outil Precision lors de la phase « synthesize » ? Et lors de la phase « optimize » ?

- d) Relever la surface (nombre de primitives, FF, etc) et la fréquence de fonctionnement du circuit
- e) Réaliser le placement et routage sur le FPGA Spartan3 de Xilinx.
- f) Regarder le routage obtenu dans Xilinx FPGA Editor.
- g) Que dire du nombre de LUTs utilisées après placement-routage ? Qu'en est-il du nombre de registres ?
- h) A quoi sert le fichier ucf ? Que contient-il ?
- i) Simuler le filtre complet obtenu après placement et routage, toujours dans l'environnement de test développé, mais en tenant compte des délais de propagation introduits dans les portes et les interconnexions. Vérifier que le fonctionnement du filtre reste correct.
- j) Qu'est-ce que le packaging VITAL ? A quoi sert-il ?
- k) Effectuer la validation du filtre sur la carte de test en chargeant le fichier de configuration (fichier \*.bit) dans le circuit Xilinx à programmer.
- l) Produire un deuxième compte-rendu avec les réponses au question IV.1 en fournissant en plus :
  - le programme de test (*bench*) écrit, en détaillant la modélisation de l'environnement et en argumentant avec précision pourquoi ce test permet d'assurer que le fonctionnement du filtre est correct,
  - rapport sur la surface du filtre (#CLB, #FF, #IO) après la synthèse et après le placement et routage,
  - le délai du chemin critique après le placement et routage,
  - le schéma niveau portes logiques après la synthèse.
  - à la fréquence de fonctionnement choisie de 50 MHz, quelle est la fréquence maximale d'échantillonnage du système ? Répondre à cette question de deux façons différentes : calcul théorique et mesure,
  - préciser la fréquence de coupure du filtre à -3db (résultat de mesure).



## V) Réalisation du filtre numérique sur Asic (1 séance)

### Introduction générale

L'objectif de cette partie de TP est de comprendre et utiliser les méthodes et les outils pour concevoir des circuits intégrés visant une cible ASIC. L'ensemble des étapes essentielles à la réalisation d'un circuit est présenté et une approche pragmatique de l'emploi des outils industriels de CAO est proposée afin de comprendre les tenants et les aboutissants de la conception des circuits intégrés.

Le flot de conception d'un circuit complexe intégré sur cible FPGA, présenté figure 8, est organisé comme suit :

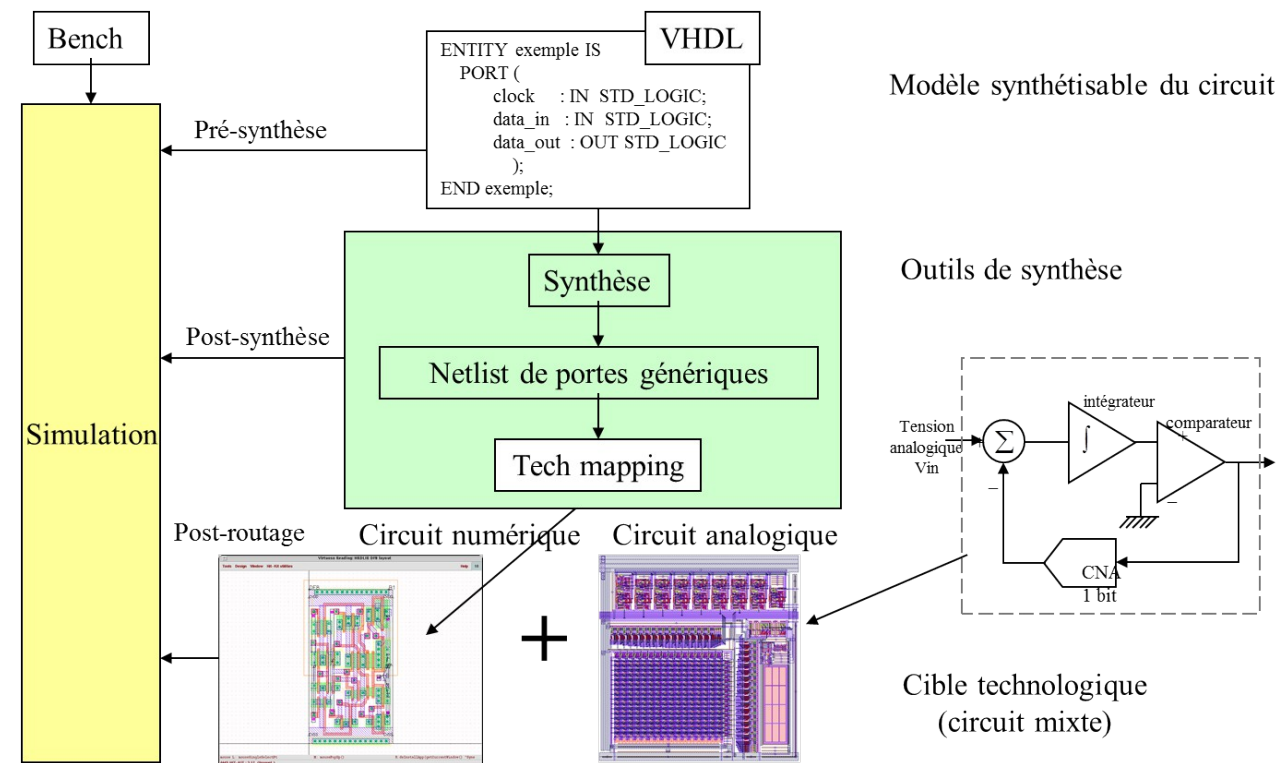


Fig 8: Flot de conception d'un circuit intégré complexe

- 1) La première tâche des concepteurs est d'analyser le cahier des charges et d'écrire une spécification simulable (en VHDL ou en Verilog par exemple). Cette étape a déjà été réalisée.
- 2) Afin de valider cette première description, il est nécessaire de concevoir un environnement de simulation (le « *testbench* ») qui garanti que la spécification est correcte. Cette étape a déjà été réalisée.
- 3) Ensuite, il est nécessaire de rendre synthétisable cette description afin de la traduire sous la forme d'une liste de portes (*gate netlist*).

L'étape de synthèse est réalisée en deux étapes :

- m) une étape de synthèse générique, qui ne fait pas appel à une cible spécifique à un fondeur,
- n) une étape de projection technologique ou « *technology mapping* » qui est spécifique à la technologie cible. A ce stade, il est possible d'effectuer à nouveau une simulation avec l'environnement défini auparavant.

- 4) L'étape suivante a pour but de faire les plans de masques du circuit qui seront utilisés pour sa fabrication. Cette phase est appelée le placement – routage, et il permet de placer les cellules (de la netlist), de définir les rails (fils) d'alimentation internes, de positionner les plots, de créer un arbre d'horloge et de router toutes les interconnexions entre les portes. C'est aussi à ce moment que l'on intègre les blocs analogiques et les blocs déjà conçus.
- 5) A l'issue de cette dernière étape, il est possible de faire à nouveau une simulation qui tient compte des temps de propagation dans les portes, mais aussi dans les interconnexions car celles-ci sont désormais connues (longueur, largeur, forme, épaisseur). Il est également envisageable de faire une simulation électrique du circuit.

## VI) Synthèse logique sur cible ASIC, validation et analyse des résultats (1 Séance)

---

Pour cette partie vous devez sourcer le fichier de configuration **config\_ASIC**.

Les outils sont à exécuter en s'étant placé dans le répertoire `${TP_PATH}/asic/synth`.

### VI.1) Synthèse logique avec l'outil Design Compiler de Synopsys

Dans cette section nous allons utiliser l'outil de synthèse de la suite Synopsys™. Cet outil s'appelle Design Compiler™. Là dessus, plusieurs outils y sont greffés, dont l'outil de synthèse simple, l'outil de testabilité (DFT Compiler™), l'outil d'évaluation du timing (PrimeTime™), l'outil d'évaluation de la puissance consommée (PrimePower™) et plein d'autres.

L'interface graphique de cet outil s'appelle Design Vision et nous allons l'utiliser autant que possible. Cependant toutes les commandes ne sont pas accessibles via l'interface graphique. Cela nous oblige à travailler de façon mixte : commandes de script – press bouton !

La configuration de l'outil Design Vision en vue de la synthèse est réalisée par l'intermédiaire d'un fichier qui contient un ensemble de commandes prédéfinies. Ce fichier est par défaut dénommé « .synopsys\_dc.setup » et il est utilisé lors de l'exécution de chaque commande liée à la lecture d'une description, d'une compilation ou d'une optimisation. Les liens entre bibliothèques logiques et physiques sont spécifiés dans ce fichier.

Pour ce TP nous utilisons la bibliothèque AMS 0.35u.

#### VI.1.1) Paramétrage et lancement de l'outil de synthèse

##### Paramétrage de l'outil de synthèse

Le paramétrage de l'outil Design Compiler en vue de la synthèse est réalisé par l'intermédiaire d'un fichier qui contient un ensemble de commandes. L'outil Design Compiler, tout comme sa version graphique Design Vision, utilise ce fichier lors de l'exécution de chaque commande liée à la lecture d'une description, à la compilation ou à l'optimisation. Les liens entre bibliothèques logiques et physiques sont spécifiés dans ce fichier.

Le nom de ce fichier de paramétrage pour Design Compiler et Design Vision est dénommé par défaut « .synopsys\_dc.setup ». Il doit être placé dans le répertoire du projet soit :

`${TP_PATH}/asic/synth`

##### Lancement de l'outil de synthèse

Placez- vous dans le répertoire `${TP_PATH}/asic/synth` et exécutez la commande :

**design\_vision &**

Le démarrage peut prendre du temps, soyez patient.

Dans la suite du document, on vous donne un extrait du UserManuel. C'est à vous de jouer avec les paramètres afin de répondre aux questions du VI.3.

## VI.1.2) Menus de l'interface Design Vision

Ce sous-chapitre présente quelques-uns des menus accessibles à travers l'interface graphique. Reportez-vous à la documentation de Synopsys pour plus de détail :

`${SYNOPSYS}/doc/syn/html/dvoh/enhanced/index.html`

la variable d'environnement SYNOPSYS étant définie lorsque le fichier de configuration est sourcé.

Vous obtiendrez la documentation complète sur les différents menus en tapant « menus » dans la zone de recherche de l'onglet « Index » dans la colonne de gauche de l'aide.

### i) Menu « File »

#### Analyze

Pour lire la source HDL, vérifier la syntaxe HDL et créer les formats objet de bibliothèque HDL. Utiliser la bibliothèque WORK. La liste des fichiers doit être ordonnée en fonction de la hiérarchie du circuit (niveaux inférieurs ou "feuilles" avant les niveaux plus élevés).

#### Elaborate

Pour créer un design en cellules génériques indépendantes de la technologie à partir d'un format HDL intermédiaire (objets générés par la commande « Analyze »). Utiliser la bibliothèque WORK.

#### Save

Pour sauvegarder en format interne sous forme de base de données (extension par défaut \*.db).

#### Save As

Pour sauvegarder dans format quelconque (choix multiple : VHDL, Verilog, Edif, ...)

Une fois le design compilé et élaboré, deux icônes deviennent visibles dans la fenêtre de travail. Elles ont la forme d'une porte logique (correspondant à la structure interne du circuit) respectivement d'un circuit à entrées/sorties multiple (correspondant à la vue externe). Dans la fenêtre « Hier », sélectionner un composant et cliquer ensuite sur ces icônes afin de parcourir la hiérarchie du composant sélectionné en observant les portes logiques, les pins d'entrée/sortie, les interconnexions.

#### Read

permet de lire le fichier source dans un format VHDL, Verilog, ...

L'option Read est équivalente aux commandes « Analyze + Elaborate » MAIS elle sera utilisée **seulement pour les fichiers** de bases de données \*.db.

Une fois l'élaboration faite plusieurs autres menus deviennent visibles.

### ii) Menu « Attributes »

Le menu « Attributes » est utilisé pour imposer des valeurs variables aux objets sélectionnés. Vous devez vous placer sur le module englobant (*top module*) pour appliquer une contrainte à l'ensemble du circuit.

#### Specify Clock

Pour préciser, pour le signal choisi comme signal d'horloge, la période et les instants des fronts (montant et descendant) à l'intérieur de la période. Avant d'utiliser ce menu, sélectionner le port CLK sur l'icône ou le schéma du composant englobant : le nom du port doit apparaître en grisé dans la fenêtre « Specify clock » qui s'ouvre.

#### Operating Environments

##### Operating Conditions

Pour préciser les conditions de fonctionnement (température, tension, et variation du processus de fabrication : PVT). Si aucun choix n'est fait, l'outil prend en compte les conditions TYPICAL de fonctionnement.

##### Wire load

Pour préciser un modèle RC pour les fils d'interconnexions. Si aucun choix n'est fait, l'outil fait une estimation de ces paramètres selon la surface du circuit.

### Optimization Constraints

Pour préciser des contraintes de fonctionnement autres que les « Operating Conditions », « Wire load » ou « Clock ».

Par exemple le sous-menu :

#### Design Constraints

Pour la surface, pour la consommation dynamique ou statique, ou la sortance (*fanout*).

On se contentera ici de n'imposer que des contraintes de surface. En entrant la valeur 0 comme contrainte de surface (ce qui est irréalisable, bien sûr), on indique au logiciel de faire au mieux. Ceci permet d'avoir une idée de la surface du circuit lors d'une première synthèse et ensuite l'optimiser en mettant une valeur non-nulle.

#### Timing Constraints

Pour les temps de propagation entre n'importe quelle entrée et n'importe quelle sortie. On ne l'utilise pas dans ce TP, mais s'avère indispensable dans le cas d'une conception plus sophistiquée.

### Optimization Directives

Pour imposer des contraintes globales sur tout le circuit ou sur une cellule particulière sélectionnée, comme par exemple mettre à plat la description.

#### iii) Menu « Design »

Le menu « Design » permet d'effectuer des optimisations, des analyses du design et de produire des rapports, par le lancement d'outils intégrés de Synopsys comme les outils spécialisés d'estimation de la puissance ou d'estimation des timings.

### Compile Design

Pour vérifier la cohérence du design, réaliser une projection sur la bibliothèque technologique (*mapping*), avec éventuellement les options de compilations suivantes (les plus importantes)

#### Top level

Optimisation du plus haut niveau hiérarchique sans toucher aux blocs internes.

#### Ungroup

Mise à plat du design. Le design sera mis à plat pour pouvoir passer au Placement Routage ASIC.

#### Incremental mapping

Permet de repartir de la version courante en cas d'optimisations successives.

**Attention : Pour une première synthèse vérifier qu'aucune option n'est cochée !**

### Check Design

Permet de vérifier les erreurs de structure et les violations de timing

### Report Design

Permet de générer des rapports sur toutes les attributs affectés au design

### Report Constraints

Permet de générer des rapports sur toutes les contraintes imposées au design ainsi que les contraintes de timing

### Report Ports, Cells, Nets, Clocks, Area

Permet de obtenir des informations sur le résultat de la synthèse.

### Report Power

Permet d'obtenir des résultats sur la puissance consommée (types et valeurs).

Attention : considérer « NETS and CELLS ». Faire « show nets histogram ». Cocher aussi « traverse hierarchy at all levels »

#### iv) Menu « Timing »

##### Check Timing

Permet de vérifier seulement le timing du circuit, rapport en cas de violation.

##### Report Timing Paths

Produit un rapport sur les chemins de propagation du plus long au plus court.  
Laisser toutes les options par défaut.

##### Path Slack

Permet de créer les histogrammes des *slacks* (marge d'un chemin de propagation par rapport au chemin critique). Cliquer sur un histogramme dans la fenêtre à droite pour voir les chemins et les valeurs des *slacks*.  
Le bouton droit de la souris vous permet de profiler les chemins et avoir plus d'information sur le timing.

##### Net capacitance

Permet de créer les histogrammes des capacités de nœuds. Cliquer sur un histogramme dans la fenêtre à droite pour voir les valeurs des capacités.

##### Path Profile View

Permet de visualiser le chemin critique avec le temps de propagation des portes se trouvant sur le chemin. Cliquer sur Flat dans la fenêtre de visualisation.  
Pour voir effectivement le chemin critique sur le design aller dans le menu « View | Highlight | Critical Path ». Pour le désactiver il suffit de faire Clear Selected.

## VI.2) Validation après la synthèse

Le résultat de la synthèse doit être sauvegardé sous la forme d'une *netlist* à plat (VHDL et Verilog sont conseillés à cette étape du projet).

Afin de pouvoir compiler et simuler la *netlist* VHDL, il est nécessaire de définir la bibliothèque de cellules. Pour cela, ajouter les deux lignes suivantes avant chaque entité dans le fichier VHDL généré par la synthèse.

```
Library C35_CORELIB;  
Use C35_CORELIB.Vcomponents.all;
```

Si vous avez choisi une synthèse à plat alors il n'y a qu'une entité et ces deux lignes ne sont donc à ajouter qu'une seule fois.

Grâce à la modification indiquée ci-dessus, le programme de test (*bench*) que vous avez écrit et utilisé dans la phase précédente du TP peut être réutilisé à ce stade. Toutefois de la même façon que lors de la synthèse sur FPGA, vous devez créer un nouveau script pour une compilation avec Modelsim du fichier VHDL du circuit synthétisé. Pensez à mettre à jour vos bibliothèques.

## VI.3) Travail demandé : analyse des résultats de synthèse

Voici quelques suggestions sur les analyses pouvant être réalisées. Toute autre analyse est possible ... et laissée à votre imagination !

- Faire une première synthèse, et observez les indications fournies par l'outil.
- Faire plusieurs synthèses différentes, en indiquant des contraintes diverses, notamment pour la fréquence d'horloge (prendre 20ns, 10ns, 5ns). Déterminer la fréquence maximum de votre circuit par des synthèses et optimisations successives.
- Commentez les conséquences sur la surface, la répartition des *slacks*, le chemin critique et les consommations statique et dynamique. Quel bloc consomme le plus ? Quel bloc impose le chemin critique ?
- Refaire une synthèse propre avec des contraintes de surface 0, conditions d'opérations TYPICAL et un signal d'horloge de 20ns de période. Vérifiez la surface du circuit et le temps de propagation. Regardez les *slacks* et la puissance totale dissipée.

- Sauvegarder la netlist à plat qui correspond à ces paramètres à la fois en VHDL pour la simulation après la synthèse et en Verilog pour une utilisation ultérieure dans l'étape de placement et routage.
- Faire une simulation après la synthèse avec le même programme de test (*bench*).
- Pourquoi la vérification statique du timing est-elle devenue nécessaire ? Comment peut-on utiliser ces informations pour augmenter les performances d'un circuit en vitesse ou en consommation ?
- Commenter les différences entre la technologie FPGA et la technologie ASIC. Est-ce que le résultat est conforme à votre intuition ? commenter.
- Fournir un cbref compte-rendu avec les réponses à ces questions.

## VII) Étude d'une bibliothèque d'un fondeur (2 séances)

---

Le fondeur est celui qui concrètement fabrique les puces électroniques à partir du dessin des masques (*layout*) remis par le concepteur du circuit. Le fondeur possède des salles blanches et maîtrise les procédés de fabrication. Chaque fondeur garde farouchement secret ses procédés de fabrication et ceci d'autant plus vigoureusement que la technologie est avancée car plus difficile à maîtriser. Il en résulte qu'un même circuit électronique, pour un même nœud technologique, présentera des caractéristiques particulières différentes selon le fondeur qui l'aura produit.

Pour concevoir correctement un ASIC, il est donc indispensable de posséder ce que l'on appelle le fichier technologique du fondeur (*design kit*) ainsi que la bibliothèque de composants précaractérisés sans lesquels il serait impossible de mener à bien des simulations. Le fondeur fournit une bibliothèque de composants pour la conception entièrement personnalisable (*full custom*) ou des cellules standard comme des portes logiques (pour les circuits numériques) ou des amplificateurs (pour les circuits analogiques).

Lorsque l'on désire réaliser un ASIC, la première chose à faire est de choisir un fondeur. Dans notre cas, nous travaillerons avec les composants d'AMS (Austria MicroSystems).

### VII.1) Introduction aux bibliothèques

Lorsqu'on souhaite réaliser un circuit intégré numérique, il est possible de concevoir le circuit en utilisant des cellules standards (*standard cells*) qui sont fournies par le fondeur. Ces cellules correspondent aux fonctions logiques élémentaires telles que AND, NAND, OR, NOR, XOR, bascules et verrous. Elles ont été conçues par le fondeur. Il en existe plusieurs modèles dans les outils de CAO : électrique, vue résumée (*abstract*), dessin des masques (*layout*), VITAL (modèle comportemental en VHDL), etc. Le fondeur met à disposition ses cellules sous forme de bibliothèques.

Afin de bien comprendre l'ensemble du flot de conception d'un ASIC, il est nécessaire d'appréhender ce que sont ces cellules standards et comment elles sont construites. Pour ce faire, nous utiliserons le logiciel Virtuoso de Cadence pour les observer.

### VII.2) Le logiciel Virtuoso de Cadence

Pour cette partie vous devez sourcer le fichier de configuration `config_ASIC`.

Le logiciel Virtuoso va produire un nombre conséquent de fichiers au cours de son exécution. Afin d'éviter le mélange et la confusion avec les fichiers produits par les autres outils, Virtuoso devra être exécuté dans le répertoire `${TP_PATH}/cadence`.

Pour lancer Virtuoso, tapez la commande :  
`amsc35b4`

L'outil peut être un peu long à démarrer la première fois, soyez patient.

### L'interpréteur de commande

Parmi les fenêtres qui sont apparues vous avez la fenêtre de l'interpréteur de commandes (*Command Interpreter Window – CIW*) montrée à la figure 9.

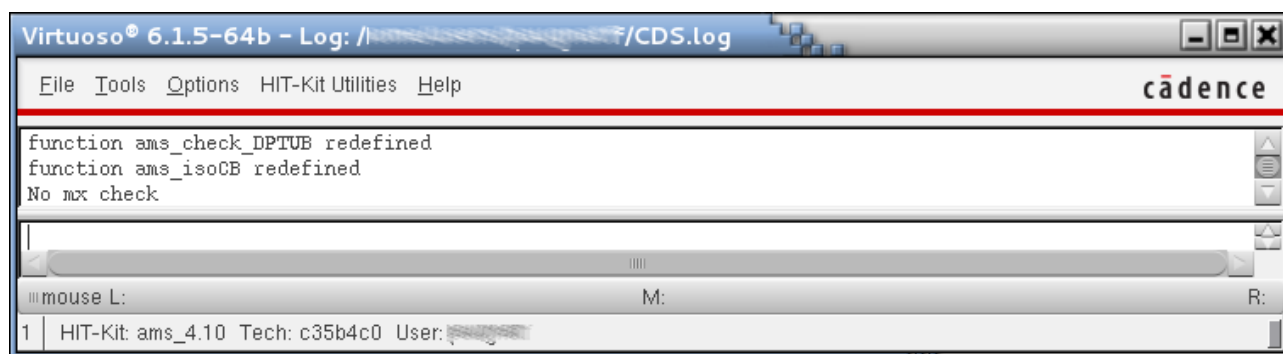


Fig 9: L'interpréteur de commandes de Virtuoso

Cette fenêtre est la fenêtre principale de la suite Cadence. Elle affiche tous les commentaires ainsi que les éventuels messages d'erreurs et donne des informations en cas de problème. Elle comporte aussi un certain nombre de menus qui sont utiles pour le fonctionnement général de tous les outils intégrés de la suite Cadence. Toutes les informations qui s'affichent dans cette fenêtre sont reportées dans un fichier dont le nom et le chemin sont inscrits dans la barre de titre.

### Le gestionnaire de bibliothèques

Le gestionnaire de bibliothèques (*Library Manager*), dont la fenêtre est présentée figure 10, est un outil central qui gère les bibliothèques de composants et donne l'accès aux composants qu'elles contiennent. Si cette fenêtre ne s'est pas affichée automatiquement, ou si vous l'avez perdue sous les autres fenêtres, vous pouvez la faire apparaître par le menu « Tools | Library Manager » de la fenêtre de l'interpréteur de commandes.

La colonne de gauche « Library » du gestionnaire de bibliothèques, vous permet de sélectionner la bibliothèque avec laquelle vous souhaitez travailler. Par exemple, CORELIB est la bibliothèque de cellules logiques standards d'AMS.

Quand une bibliothèque est sélectionnée dans la colonne « Library », les cellules qu'elle contient remplissent la colonne centrale « Cell ».

Après qu'une cellule ait été choisie dans la colonne « Cell », les vues enregistrées pour cette cellule sont listées dans la colonne de droite « View ».

Double cliquez sur une vue pour l'afficher. Si un message vous demandant de changer de licence apparaît, répondre « yes », au besoin plusieurs fois. Le logiciel donnera l'accès aux cellules lorsque la licence « Virtuoso\_Layout\_Suite\_GXL » aura été validée. Accepter (yes) le message vous demandant si vous voulez ouvrir la cellule en lecture seule.

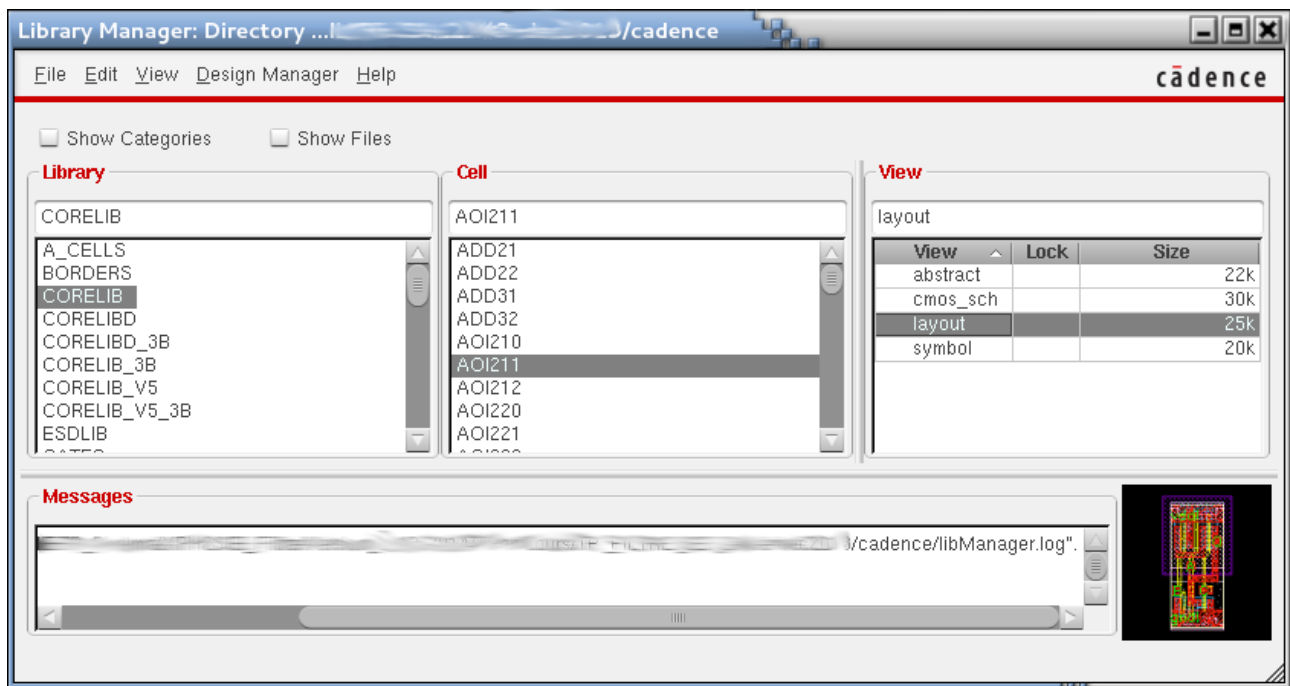


Fig 10: Le gestionnaire de bibliothèques de Virtuoso

## VII.3) Aide à la lecture du dessin des masques (*layout*)

### VII.3.1) Le transistor NMOS

Un transistor NMOS est réalisé à partir d'un substrat de type P. On réalise deux implantations de type N+ pour les contacts de drain et de source. En effet, le contact aura un comportement ohmique si la jonction métal-semiconducteur est réalisée avec un semiconducteur fortement dopé. La grille du transistor est réalisée par le dépôt d'une fine couche d'oxyde de silicium ( $\text{SiO}_2$ ) qui est un isolant. Ce dernier est recouvert de polysilicium, le contact de grille étant déporté afin de préserver la couche d'oxyde lors des étapes de fabrication. La vue *layout* d'un transistor ne montre jamais ce contact de grille. Voir figure 11.

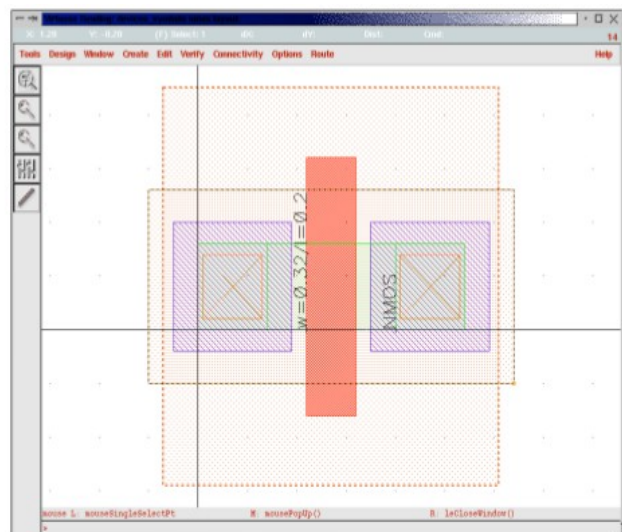
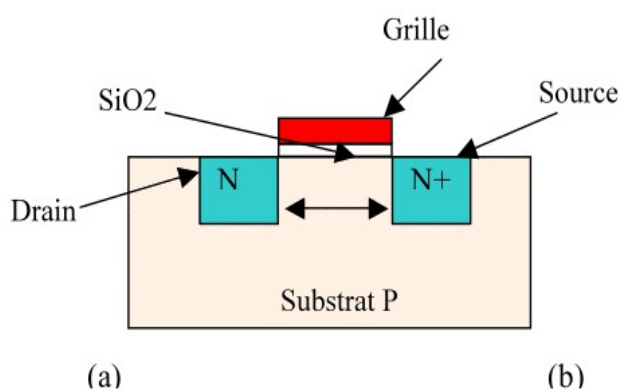


Fig 11: Coupe (a) et vue layout (b) d'un transistor NMOS



### VII.3.2) Le transistor PMOS

Comme on part toujours d'un substrat de type P, un caisson de type N est réalisé afin d'y loger le transistor. Il y a ensuite deux implantations de type P+ pour les contacts de drain et de source. La grille est réalisée comme pour le transistor NMOS (cf. figure 12).

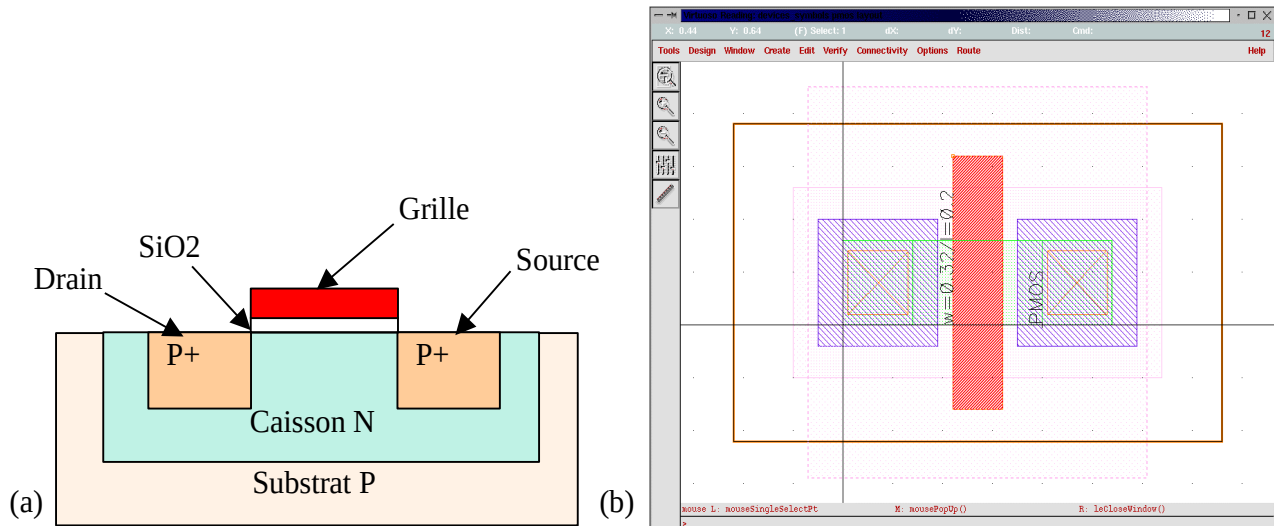


Fig 12: Coupe (a) et vue layout (b) d'un transistor PMOS

### VII.3.3) Autres informations

Les contacts sont représentés par des carrés contenant une croix, la zone de diffusion du canal est toujours représentée en vert, le polysilicium est en rouge et le premier niveau de métal en bleu.

## VIII) Placement et routage ciblés ASIC avec l'outil SoC Encounter de Cadence (2 séances)

Encounter® Digital Implementation System<sup>2</sup> (EDI System), plus communément dénommé SoC Encounter, est un outil de placement routage automatique des circuits numériques qui appartient à la suite Encounter de Cadence. Il s'agit d'un outil puissant très flexible et paramétrable mais également gourmand en ressource machine. Son utilisation est assez complexe et demande une certaine expertise pour tirer partie de toutes les possibilités. Le format d'entrée de SoC Encounter est une liste de portes (netlist) au format Verilog.

Un circuit complet est constitué d'un cœur (*core*), d'amplificateurs (*buffers*) pour les connexions vers l'extérieur et de plots de connexion (*pads*). Les amplificateurs de connexion servent à adapter les niveaux de tension et de puissance entre le cœur et la carte sur laquelle sera soudé le circuit. L'ensemble cœur + amplificateurs de connexion + plots de connexion constitue la puce (*chip = core + buffers + pads*)

A l'aide de l'outil de synthèse Design Vision (cf. VI), vous avez effectué une synthèse du cœur du circuit de filtre pour des conditions de fonctionnement typiques et une fréquence d'horloge correcte et vous avez produit une liste de portes au format Verilog. C'est cette liste que vous importerez dans l'outil SoC Encounter afin de procéder au placement et routage de votre circuit. Afin de simplifier la rédaction de ce document, nous supposons que vous l'avez nommé `filtre_synth_plat.v`, à vous de traduire ce nom par le nom que vous avez effectivement employé.

Pour des raisons de temps imparti, la description des plots d'entrée et de sortie et de leurs connexions avec le cœur a déjà été faite pour vous, car cette étape peut s'avérer assez chronophage. Ces fichiers seront étudiés dans la suite du document.

Un assez grand nombre de commandes est nécessaire pour placer et router totalement un circuit numérique complexe.

Le graphe de la figure 13 donne une idée de la succession des opérations à effectuer. L'outil SOC Encounter gère toutes ces étapes, offrant trois possibilités pour placer et router un circuit :

- de façon interactive par les menus de l'outil.
- en entrant directement des commandes « SOC Encounter » dans la console de l'outil.
- en enchainant ces commandes dans un script.

Dans ce TP, vous utiliserez un mélange de ces possibilités, afin de rendre le TP faisable dans le temps imparti tout en conservant les aspects pédagogiques :

Pour cette partie du TP vous devez sourcer le fichier de configuration `config_ASIC`. Ce script de configuration définit notamment les variables d'environnement `ENCOUNTER` et `AMS_DIR`.

### Trouver de l'aide sur SoC Encounter

- Manuel de l'interface graphique avec détails des menus  
`${ENCOUNTER}/doc/encounter/encounterTOC.html`
- Manuel de l'utilisateur  
`${ENCOUNTER}/doc/soceUG/soceUGTOC.html`
- Pour obtenir toute l'aide fournie par Cadence, démarrez le système d'aide soit depuis l'interface graphique soit en exécutant la commande :  
`${ENCOUNTER}/tools/bin/cdnshelp`

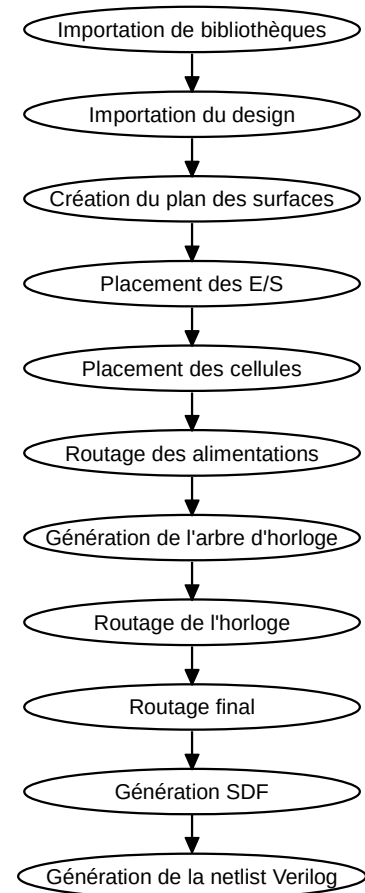


Fig 13 : Étapes de placement et routage

<sup>2</sup> nouvelle dénomination donnée par Cadence

## VIII.1) Configuration de SoC Encounter

**ATTENTION** : La procédure de configuration décrite dans ce paragraphe n'est à faire qu'une seule fois dans la vie du projet, avant le premier démarrage de SoC Encounter

L'outil de placement et routage doit être réglé de façon à utiliser le kit technologique du fondeur (design kit) et prendre en compte les spécificités du projet. Ceci est fait au moyen d'un fichier de configuration qui est en fait un script. Ce script garantit que la configuration initiale de l'outil est toujours la même à chaque fois qu'un travail est effectué sur le même circuit. En effet, le nombre de commandes à exécuter avant de commencer à travailler peut être conséquent ! D'où l'intérêt du script, même si un bon nombre de ces commandes auraient pu être lancées depuis l'interface graphique.

Pour l'essentiel, les commandes lancées depuis le script servent à :

- renseigner des valeurs par défaut, par exemple pour des délais ou pour les dimensions par défaut du cœur
- spécifier les bibliothèques requise pour le placement et routage dans la technologie visée.
- spécifier certains fichiers décrivant la puce à router (liste de portes pour le cœur + spécification des amplificateurs de connexion de certains nœuds tel que la clock, reset ...)

Nous utilisons ici la technologie de CMOS 0,35  $\mu\text{m}$  d'AMS avec 4 couches de métal (C35 thick4M). Pour créer le fichier de configuration pour SoC Encounter, le fondeur AMS mets à disposition de ses clients un programme prévu à cet effet baptisé *HitKit EDI Place & Route Setup*. La documentation de cet outil de configuration est disponible en ligne à l'URL <http://asic.ams.com/hitkit/hk410/edi/setup.html>

- Placez-vous dans le répertoire de travail  
`${TP_PATH}/asic/par`
- Lancez le générateur de scripts de configuration d'AMS en exécutant la commande :  
`${AMS_DIR}/programs/bin/ams_edi`
- Renseignez les champs (voir figure 14)
  - zone « Verilog Netlist Name », pour indiquer le nom du fichier (sans chemin) décrivant votre circuit synthétisé (cf. VI)
  - zone « Verilog TopCell Name », pour indiquer le nom du module englobant (*top cell*) de la puce complète (cœur avec amplificateurs et plots de connexion). C'est le nom du module défini dans le fichier  
`${TP_PATH}/asic/VERILOG/filter_io_etudiants.v`
  - prendre les bibliothèques typiques pour le cœur (CORELIB) et pour les entrées/sorties (IOLIB)
  - choisir une tension de 3,3 V pour le cœur et pour les entrées/sorties
- Cliquez le bouton « Create Setup Files », l'ensemble des fichiers et répertoires nécessaires pour le fonctionnement de SoC Encounter est alors créé dans le répertoire courant.  
Le script de configuration a un nom de la forme `{Technology}_{TopCell Name}.conf`

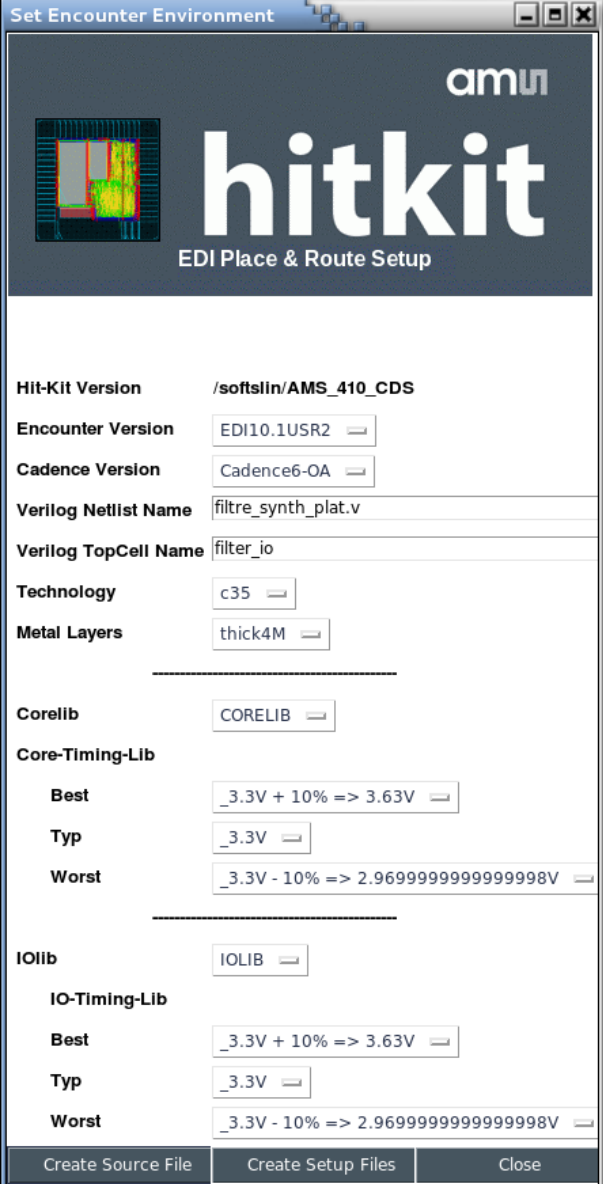


Fig 14 : Le générateur de scripts de configuration pour SoC Encounter

- Assurez-vous que la mention  
« `--# EDI Setup Script written and executed: amsEdiSetup` »  
soit apparue dans la console (sinon patientez, l'exécution du script n'est pas terminée) puis cliquez sur le bouton « Close » pour fermer le générateur de scripts de configuration
- Copiez le fichier Verilog décrivant le cœur du circuit synthétisé (fichier `filtre_synth_plat.v` – cf. VI)  
dans le répertoire  
`${TP_PATH}/asic/par/VERILOG`  
qui vient d'être créé par le générateur de scripts de configuration

**ATTENTION** : il a été supposé lors de la création du fichier  
`${TP_PATH}/asic/VERILOG/filter_io_etudiants.v`  
que le nom de l'entité du module englobant dans la description du cœur était `FILTER` (dans le fichier `filtre_synth_plat.v`). Si vous aviez choisi un autre nom vous devez modifier le fichier `filter_io_etudiants.v` pour assurer la concordance des noms.

**ATTENTION** : Visualiser et comprendre le document `fitre_io_etudiants.v` sans le modifier !

## VIII.2) Démarrage de SoC Encounter

La configuration décrite au VIII.1 ayant été effectuée, placez-vous dans le répertoire de travail  
`${TP_PATH}/asic/par`  
et tapez la commande (**sans '&'**)<sup>3</sup>  
**encounter**

## VIII.3) Chargement des scripts du fondeur

Le fondeur AMS mets gracieusement à disposition des scripts qui permettent de réaliser des opérations complexes dans SoC Encounter. Certains de ces scripts seront utilisés au cours de ce TP. Ceux-ci doivent d'abord être chargés dans SoC Encounter, ce qui peut se faire en exécutant la commande  
`source amsSetup.tcl`  
dans la console depuis laquelle SoC Encounter (celle dont le prompt a été remplacé par `encounter {n°}>`).

Cette opération n'est nécessaire que lors de l'importation d'un design (= création du projet de placement routage, voir §VIII.4). Lors de la restauration d'un projet commencé, ces scripts sont rechargés automatiquement s'ils avaient été chargés avant la sauvegarde.

Pour connaître la liste des scripts qui viennent d'être chargés, il suffit de taper dans cette même console, la commande `amsHelp`

L'aide sur ces scripts peut être trouvée à l'URL suivante :  
<http://asic.ams.com/hitkit/hk410/edi/flow.html>

## VIII.4) Importation du Design

Sélectionner la fenêtre graphique de SoC Encounter et allez dans le menu

**File | Import Design ...**

Dans la fenêtre « Design Import », cliquer le bouton « Load... » en bas de la fenêtre.

Sélectionnez le fichier `*.conf` créé par le générateur de scripts de configuration (cf. VIII.1) puis cliquez le bouton « Open » sur la droite.

Dans l'onglet « Basic » (voir figure 15), complétez la liste de fichiers Verilog à utiliser en ajoutant  
`${TP_PATH}/asic/par/VERILOG/filter_io_etudiants.v`  
et indiquez dans la zone de saisie « IO Assignment File » le nom du fichier décrivant l'agencement de la couronne de plots de connexion  
`${TP_PATH}/asic/par/CONSTRAINTS/top_FILTER_etudiants.io`

---

<sup>3</sup> Le nom du logiciel et de ses commandes ont changées plusieurs fois au cours de l'évolution de ce logiciel. À la place de `encounter`, certaines documentations peuvent indiquer la commande `velocity` qui est l'ancien nom.

Dans l'onglet « Advanced », menu « Power » (cf. figure 16), ne conserver que **vdd!** dans le champ « Power Nets » et **gnd!** dans le champ « Ground Net »

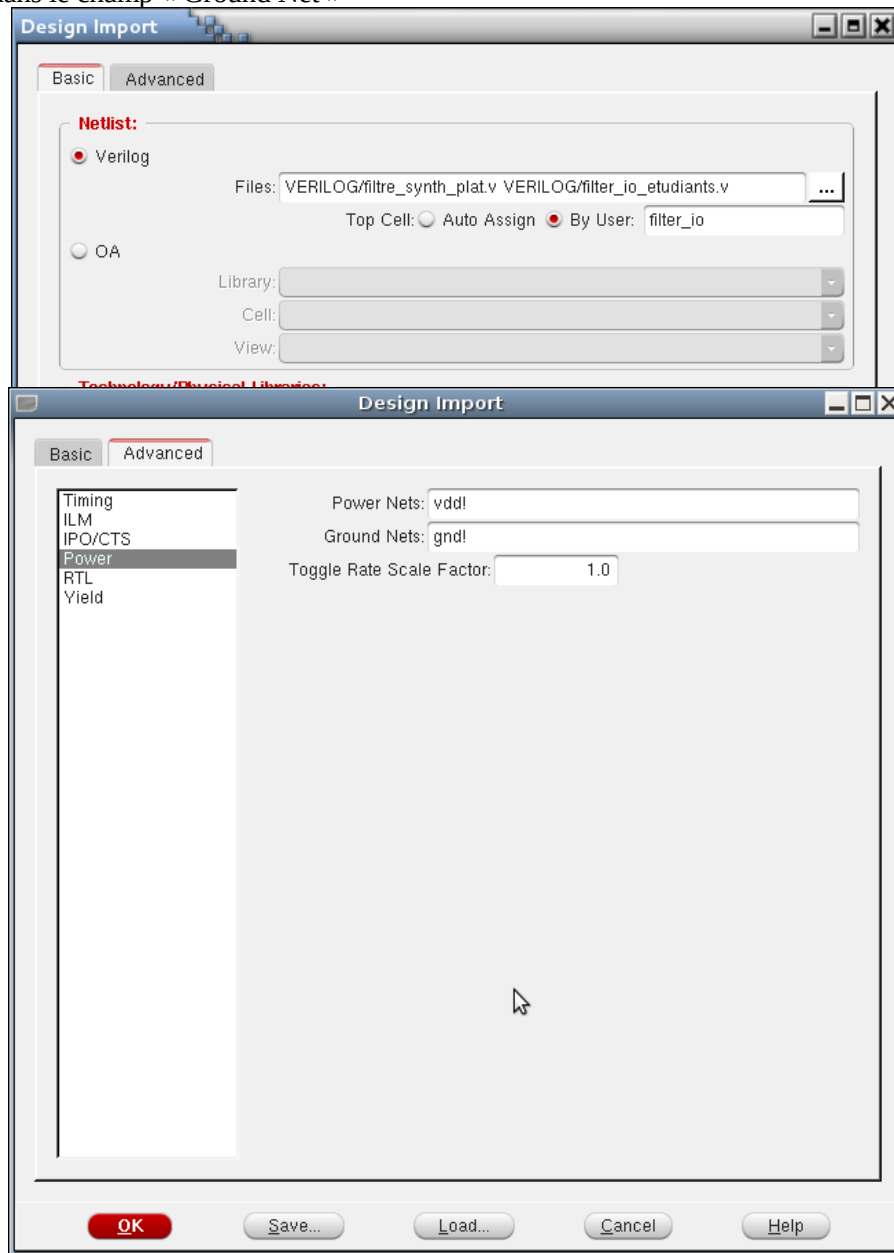


Fig 15 : Fenêtre « Design Import », onglet « Basic »

Fig 16 : Fenêtre « Design Import », onglet « Advanced »

Sauvegardez ces modifications en cliquant le bouton « Save... ». Le fichiers de configuration sera réécrit (vous pouvez écrasez celui créé par le générateur de scripts de configuration au §VIII.1).

Cliquez sur « OK » et vous obtenez une vue du circuit avec la couronne de plots placée et une zone centrale réservée pour le cœur comme cela est montré sur la figure 17.

### Remarques

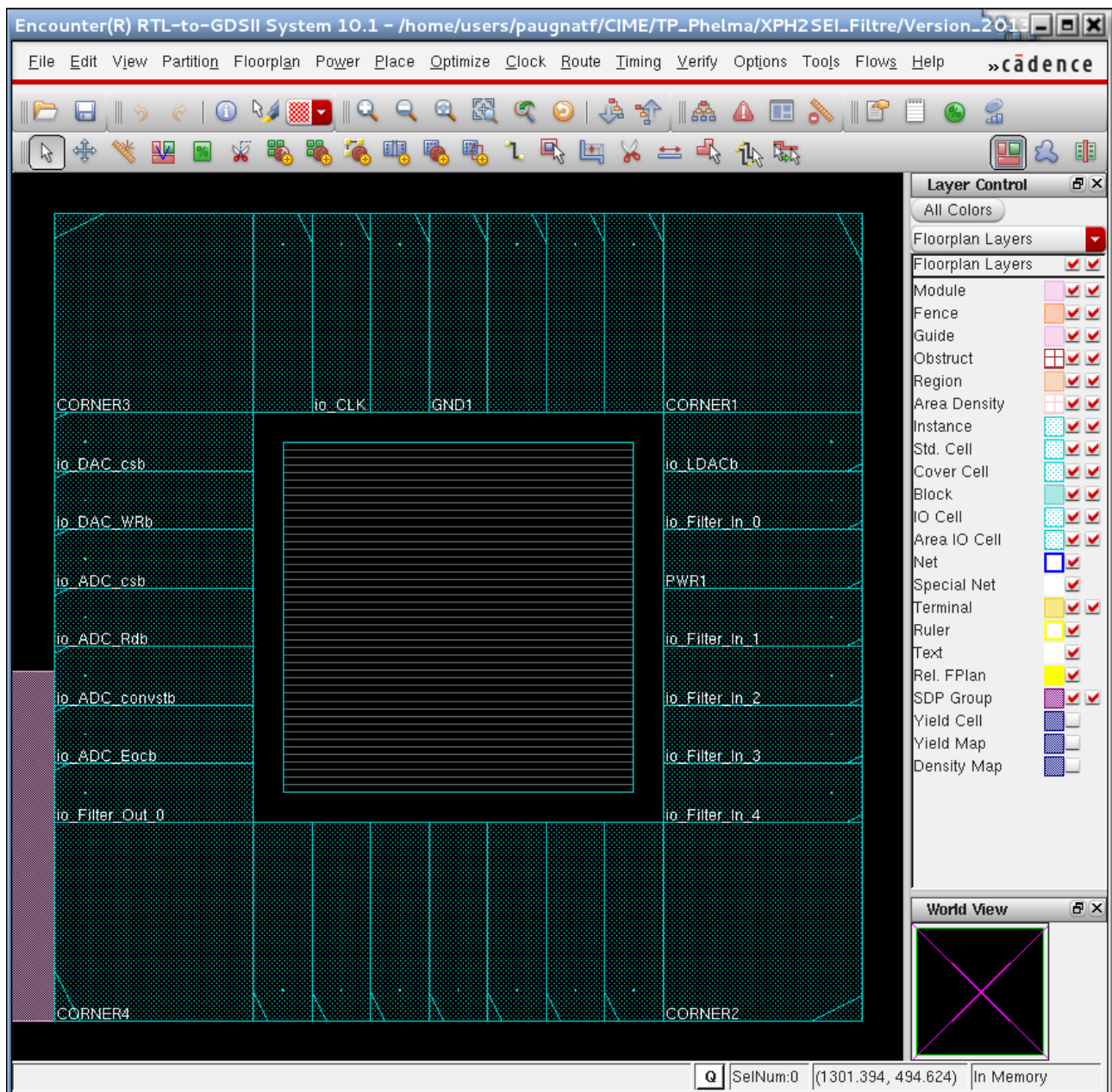



Fig 17 : Vue du circuit avec la couronne de plots avant routage

Le rafraîchissement de la zone de vue ne se fait pas toujours automatiquement. Pour forcer le programme à redessiner la zone de vue, tapez **Ctrl+R** ou cliquez sur l'icône .

Pour zoomer et dézoomer, tournez la molette de la souris.

Un clic sur le bouton central de la souris place la zone pointée par la flèche de la souris au centre de la zone de vue.

Le composition des touches **Ctrl** et **Maj (Shift)** avec la rotation de la molette permet de faire défiler la vue latéralement ou verticalement.

Dans la zone « World View » (en bas à droite par défaut), le carré noir représente le circuit tandis que le carré rose représente la vue. Cliquer et déplacer le carré rose dans cette zone fait se déplacer la vue dans la zone de vue.

Pour replacer la vue au centre du circuit et la réajuster de façon à voir le circuit dans son ensemble, appuyer sur la touche 'f'.

**Sauvegardez régulièrement votre travail**

Pour sauvegarder un design, menu

**File | Save Design ...** ou touche F2

Les fichiers de sauvegarde sont d'extension \*.enc. Il est possible de donner un nom différent à chaque sauvegarde.

Pour charger une sauvegarde et revenir à un état antérieur, menu

**File | Restore Design ...**

**ATTENTION** : certaines actions sont irréversibles ! Donc, sauvegardez votre travail à chaque étape et utilisez des noms de sauvegarde différents.

## VIII.5) Définition des surfaces de placement-routage (Floorplan)

### VIII.5.1) Couronne de plots

Le positionnement des plots est défini dans le fichier `${TP_PATH}/asic/par/CONSTRAINTS/top_FILTER_etudiants.io` qui a été préparé pour vous dans ce TP. Vous pouvez toutefois le visualiser dans un éditeur de texte à titre d'information.

La spécification d'un nouveau plot commence par le mot clé `Pad` : (premier champ).

Le deuxième champ est le nom de l'instance du plot. Ce nom s'affiche sur le dessin du plot (zoomer si le nom ne s'affiche pas) dans la zone de vue de la fenêtre principale de SoC Encounter. Pour les signaux d'entrée ou de sortie du cœur qui doivent être connecté avec l'extérieur, ce nom d'instance renvoie au nom de l'instance de l'amplificateur de connexion associé. Les amplificateurs de connexion sont instanciés dans le fichier décrivant la puce complète `VERILOG/filter_io_etudiants.v`.

Les coins (*corner*) sont des plots spéciaux qui n'interviennent pas dans l'interconnexion avec le cœur. Il n'apparaissent donc pas dans le fichier `VERILOG/filter_io_etudiants.v` et leur nom d'instance est conventionnel.

Le troisième champ est la zone de placement du plot par rapport au cœur, selon une désignation dans le style des points cardinaux. Le nord (N) est en haut de la zone de vue, l'est (E) à droite, etc. L'ordre d'écriture des lignes détermine l'ordre de placement dans une zone donnée, le n° 1 est à gauche ou en bas.

Le quatrième champ n'est renseigné que sur les plots spéciaux : les coins (CORNERP) et les plots d'alimentation (VDD3ALLP et GND3ALLP).

Des champs supplémentaires, non-renseignés ici, pourraient être ajoutés pour spécifier des contraintes de dimension du plot.

### VIII.5.2) Alimentation du circuit

Pendant la conception du cœur du circuit, vous ne vous êtes pas préoccupés de l'alimentation du circuit : aucun de vos modules ne comporte explicitement les entrées de puissance (Vdd ou Gnd). Pourtant les circuits électroniques fonctionnent à l'électricité ! Il faut donc bien apporter l'énergie jusqu'aux cellules du cœur ! En fait, les bornes de puissance sont sous-entendues dans la description des portes logiques. Dans le cas général, il peut y avoir plusieurs alimentations (par exemple +5V, +3V et -5V) et même plusieurs masses (par exemple un GND numérique et un GND analogique). Dans ce TP, nous restons dans le cas simple d'une seule alimentation qui sera désignée par `vdd!` et d'une seule masse appelée `gnd!` (le point d'exclamation fait partie du nom).

La mise en place de l'alimentation du cœur se fait en plusieurs étapes :

- aménagement d'un espace entre le cœur et la couronne de plots
- placement d'anneaux d'alimentation (respectivement `vdd!` et `gnd!`) dans l'espace aménagé entre le cœur et la couronne de plots
- placement de bandes de conduction destinées à réduire l'effet de l'impédance des rails d'alimentation du cœur
- connexion des rails d'alimentation du cœur aux bandes de conceptions

#### i) Aménagement d'un espace entre le cœur et la couronne de plots

Menu

## Floorplan | Specify Floorplan...

Choisissez les valeurs suivantes :

- Ratio : 1.0 (impose une forme carrée au cœur)
- Core Utilization : 0.70
- Core Margins by : Core to IO Boundary : 80 (sur les quatre côtés)

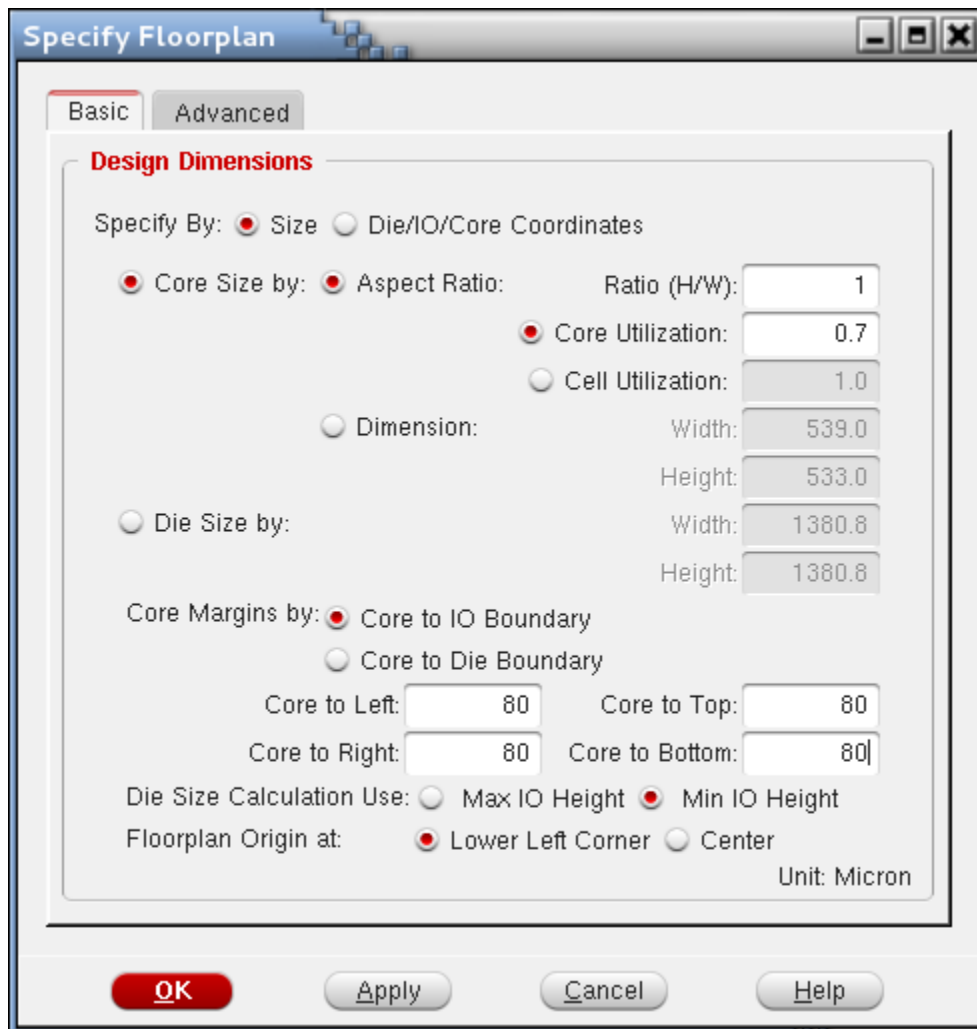


Fig 18 : Paramètres pour aménager l'espace entre le cœur et la couronne de plots

### ii) Alignement des objets sur la grille du fabricant

Afin d'éviter ultérieurement des erreurs de placement, les éléments (cellules, plots, ampli, etc.) doivent être alignés sur la grille définie par le fondeur. Pour cela, exécuter la commande suivante dans la console depuis laquelle SoC Encounter a été lancé :

```
amsUserGrid
```

### iii) Placement des anneaux d'alimentation

La largeur des anneaux d'alimentation dépend de la consommation de votre circuit et des caractéristiques de la technologie utilisée. Il faut compter une largeur de piste d'environ 1  $\mu\text{m}$  pour 1 mA.

Ici, des anneaux constitués de pistes de 20  $\mu\text{m}$  suffiront.

Menu

## Power | Power Planning | Add Ring...



Dans l'onglet « Basic », remplir la zone de saisie « Net(s) » avec le nom des alimentations, soit ici vdd! et gnd!

Définir la largeur et l'espacement des anneaux d'alimentation pour chaque côté du cœur :

width = 20 et spacing = 10

Le paramètre offset indique 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. Voir la figure 19.

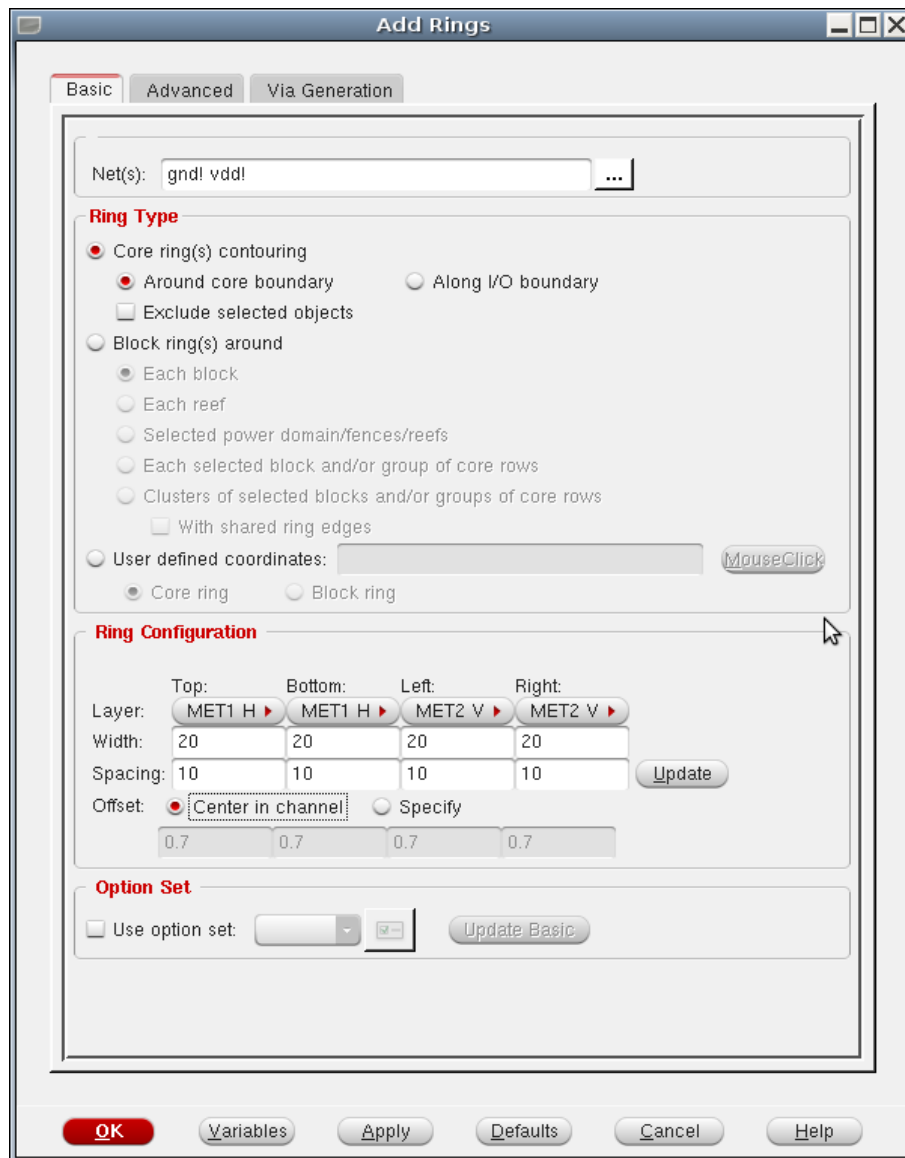


Fig 19 : Spécification des anneaux d'alimentation

#### iv) Placement de bandes de conduction

Quatre paires de bandes de conduction sont définies et régulièrement espacées sur le cœur. Destinée à répartir l'approvisionnement en courant, leur largeur est donc logiquement un quart de celle des anneaux d'alimentation.

Menu

**Power | Power Planning | Add Stripe...**

Dans la zone « Set Configuration » de l'onglet « Basic », remplir la zone de saisie « Net(s) » avec le nom des alimentations, soit ici vdd! et gnd!. Voir figure 20.

Définir la largeur et l'espacement des bandes de conduction pour chaque côté du cœur :  
width = 5 et spacing = 0.5

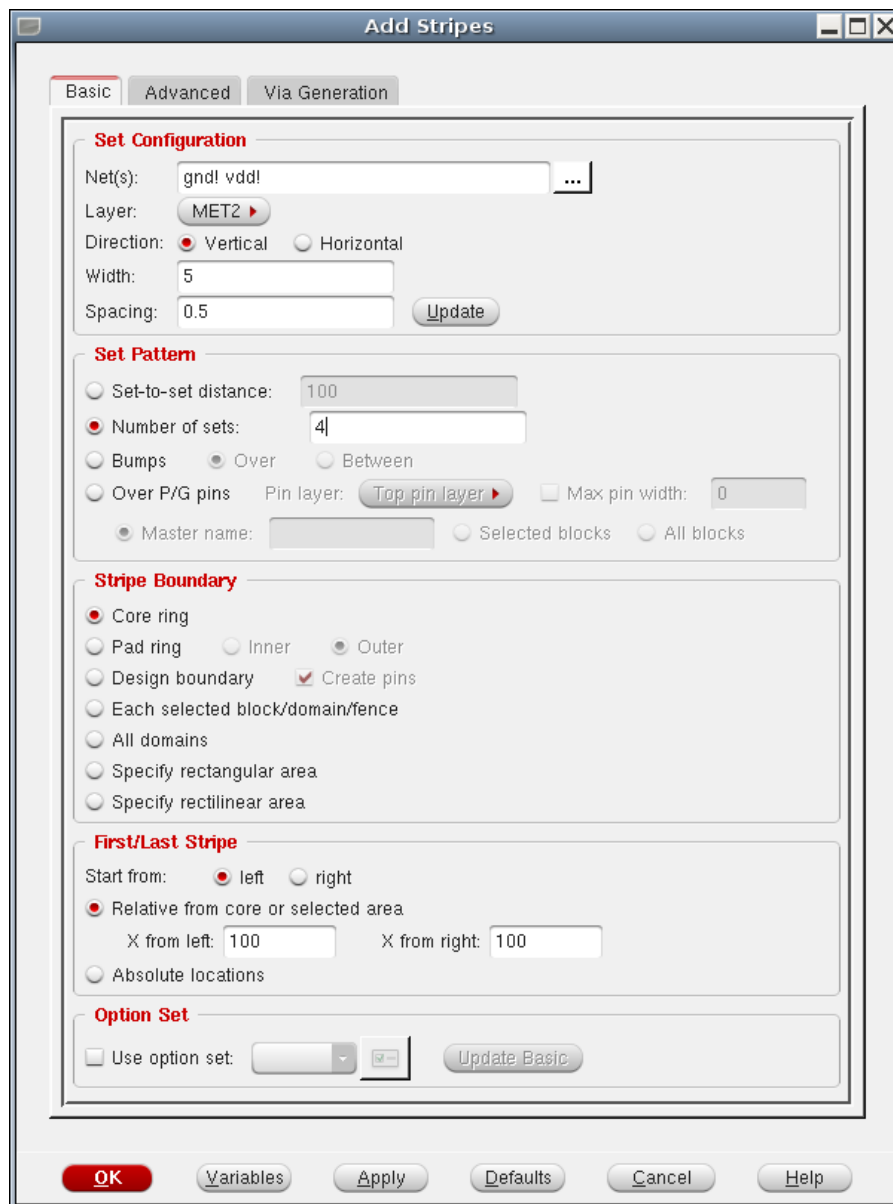


Fig 20 : Spécification des bandes de conduction

Dans la zone « Set Pattern », sélectionner « Number of sets » et spécifier 4.

Dans la zone « Stripe Boundary », sélectionner « Core Ring ».

Dans la zone « First/Last Stripe », sélectionner « left » et « Relative from core or selected area » puis indiquer 100 dans les champs « from left » et « from right ».

#### v) Connexion des rails d'alimentation du cœur

Cette étape va définir un lien entre les bornes de puissance de chaque cellule et les plots d'alimentation du circuit. Le lien est obtenu par l'intermédiaire de deux nœuds globaux auxquels sont respectivement connectés d'une part les bornes de puissance des cellules composant le cœur et d'autre part les amplis de connexion des plots d'alimentation du circuit. D'où une procédure de configuration en deux étapes. Elle est suivie d'une troisième étape au cours de laquelle les rails d'alimentation seront effectivement routés.

Le nom par défaut pour désigner la borne d'alimentation positive des cellules du cœur est vdd! Ce nom est réemployé ici pour désigner le nœud global (*Global Net*) d'alimentation positive. Le nom par défaut pour désigner la borne d'alimentation négative des cellules du cœur est gnd! Ce nom est réemployé ici pour désigner le nœud global d'alimentation négative.

**Attention** : le terme « *pin* » est employé dans les panneaux de configuration de SoC Encounter pour désigner les bornes de puissance des cellules composant le cœur et non-pas une broche du boîtier !

Pour l'ampli de connexion du plot d'alimentation positive (PWR1), le nom de la borne côté cœur est 'A'. Il en est de même pour l'ampli de connexion du plot de masse (GND1).

Menu

### **Power | Connect Global Nets...**

Sélectionner « Pin » dans la zone « Connect ».

#### **Étape n°1)**

Connexion des bornes de puissance des cellules composant le cœur aux nœuds globaux d'alimentation :

- Taper '\*' dans « Instance Basename »
- Taper vdd! dans les champs « Pin Name(s) » et « To Global Net » puis cliquer sur « Add to List »
- Taper gnd! dans les champs « Pin Name(s) » et « To Global Net » puis cliquer sur « Add to List »

#### **Étape n°2)**

Connexion des amplis de connexion des plots d'alimentation aux nœuds globaux d'alimentation :

- Taper PWR1 dans « Instance Basename »
- Taper 'A' dans le champ « Pin Name(s) »
- Taper vdd! dans le champ « To Global Net » puis cliquer sur « Add to List »
- Taper GND1 dans « Instance Basename »
- Laisser 'A' dans le champ « Pin Name(s) »
- Taper gnd! dans le champ « To Global Net » puis cliquer sur « Add to List »

Appliquer les opérations indiquées dans la colonne de gauche « Connection List » en cliquant le bouton « Apply » puis fermer la fenêtre en cliquant « Cancel » (voir figure 21). Il n'y a pour le moment aucun changement dans la zone de vue.

#### **Étape n°3)**

Routage des rails d'alimentation du cœur

Menu

### **Route | Special Route ...**

Dans l'onglet « Basic », la zone de saisie « Net(s) » sert à indiquer les nœuds qui seront à connecter aux bandes placées par dessus le cœur (cf. VIII.5.2)iv). Il s'agit ici des alimentations gnd! et vdd!.

La zone « SRoute » sert à indiquer quel routage est à effectuer avec les anneaux d'alimentation et les bandes de conduction.

- « Block Pins » pour permettre de relier plusieurs bloc sur les mêmes anneaux d'alimentation les entourant. Ceci n'a que peu de sens ici puisque le design n'a qu'un seul bloc.
- « Pad Pin » pour indiquer de connecter les plots d'alimentation à leur anneau d'alimentation respectif.
- « Pad Rings » (couronnes<sup>4</sup> de plots) pour indiquer de créer une couronne de plots (ou plusieurs concentriques) si celle-ci n'est pas encore définie.

---

4 il est possible d'avoir plusieurs couronnes de plots concentriques autour d'un même cœur

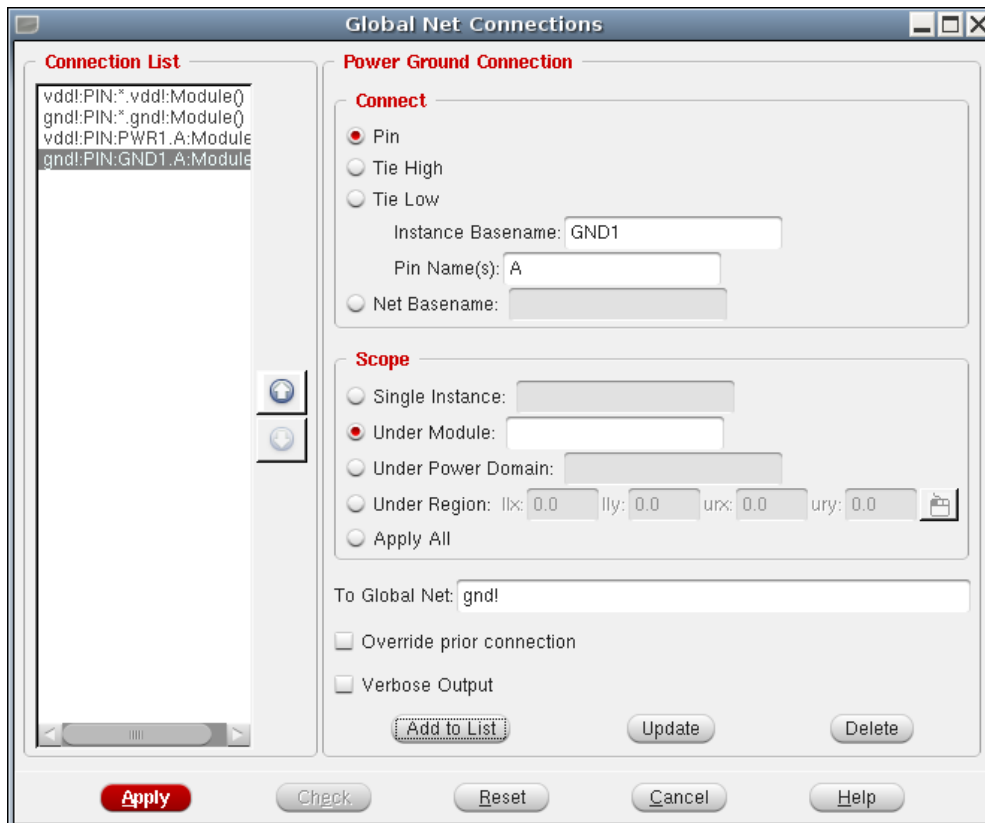


Fig 21 : Définition des connexions entre bornes de puissance des cellules du cœur et amplitudes de connexion des plots d'alimentation du circuit

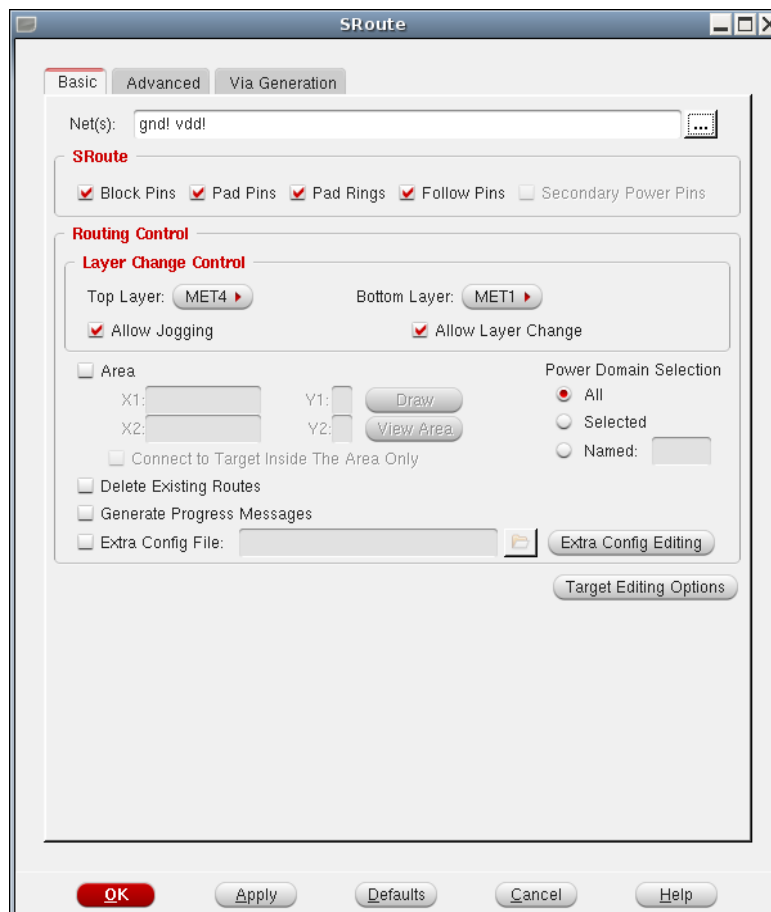


Fig 22 : Fenêtre « Special Route »

- « Follow Pin » pour indiquer de relier les bornes d'alimentation des cellules standards aux rails d'alimentation qui vont être disposés en lignes à travers le cœur.

La zone « Routing control » sert à spécifier des contraintes de routage. La zone « Layer Change Control » permet d'indiquer les couches de métal autorisées pour le routage des alimentations et si le changement de couche est permis.

Si la case à cocher « Delete Existing Routes » est cochée, le routage est recommencé à zéro sinon le routage est incrémental.

Parmi les autres options, « Area » sert à spécifier de restreindre le routage à seulement une partie de la puce. « Power Domain Selection » sert restreindre le routage que sur un domaine d'alimentation (un nœud à une même tension d'alimentation). Ces options prennent tout leur sens pour les gros circuits.

Inscrire « gnd! vdd! » dans la zone de saisie « Net(s) » et laissez les valeurs par défaut pour les autres réglages (voir figure 22) ; au besoin cliquez le bouton « Defaults ».

Cliquez « OK », vous obtenez une vue comparable à celle de la figure 23.

***Pensez à faire une sauvegarde !!***

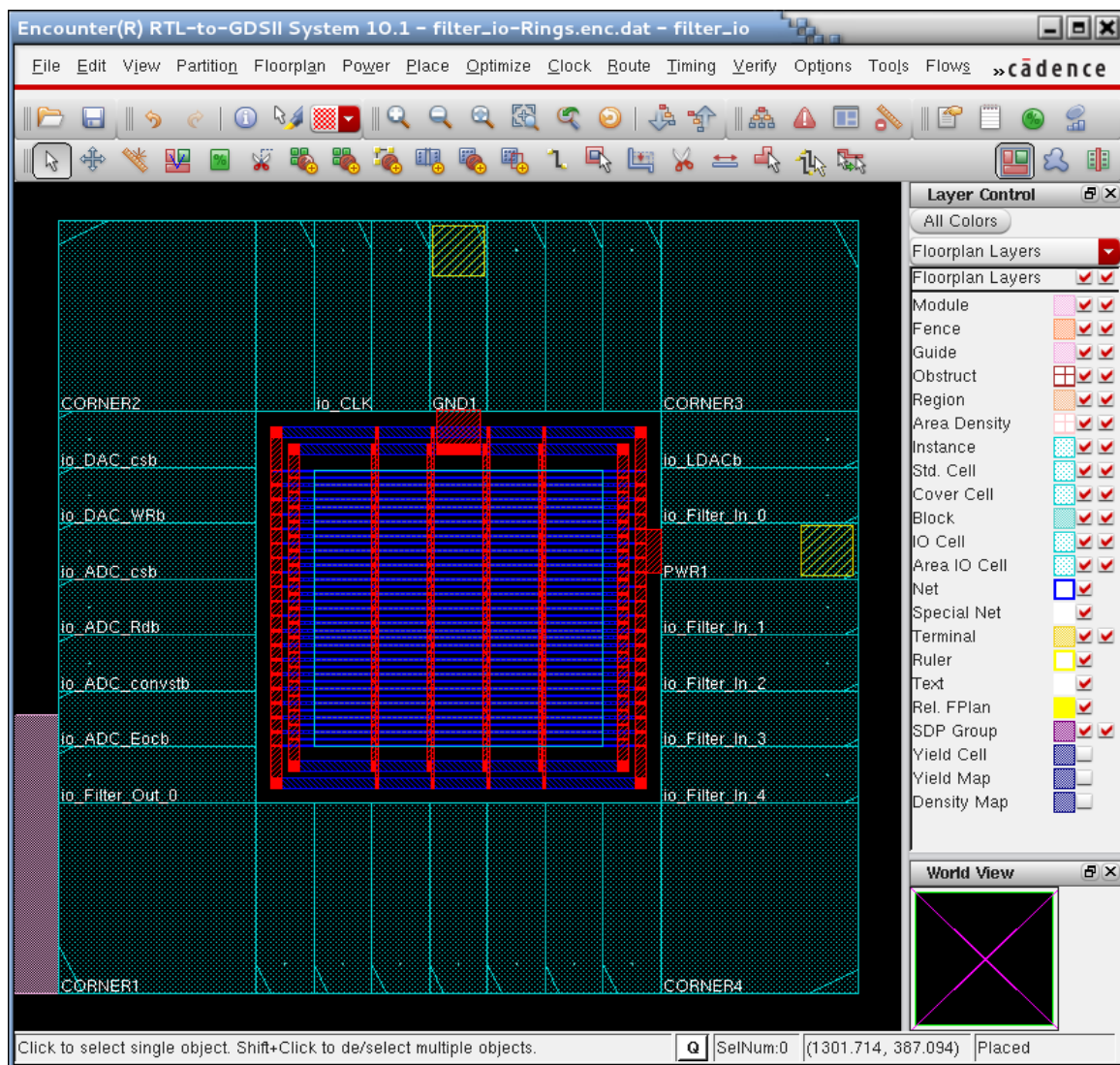


Fig 23 : Vue du circuit avec les alimentations routées

## VIII.6) Placement des cellules composant le cœur

Il faut d'abord informer SoC Encounter du nœud technologique à utiliser. Par défaut SoC Encounter utilise un nœud technologique de 90 nm. Il ne distingue cependant pas les valeurs au-delà de 130 nm. Or, la bibliothèque technologique AMS utilisée ici est en 350 nm (0.35  $\mu\text{m}$ ). Il faut donc indiquer d'utiliser un nœud technologique supérieur ou égal à 130 nm par la commande suivante, à taper dans la console dans laquelle l'outil a été lancé :

```
setDesignMode -process 130
```

Pour un bon fonctionnement du circuit, des condensateurs de découplage doivent être placés entre les rails d'alimentation vdd! et gnd!. Ils sont placés par le script suivant (toujours à exécuter dans la console de SoC Encounter) :

```
amsAddEndCaps
```

Ensuite sélectionnez le menu

### **Place | Place Standard Cells...**

- Sélectionnez « Run Full Placement »
- Dans la zone « Optimization options », sélectionnez « Include Pre-Place Optimization »
- Cliquer « OK »

Pour faire apparaître les cellules placées, sélectionnez la vue physique en cliquant l'icône *physical view* à droite de la barre d'outils



Menu

### **Place | Check Placement...**

Vérifiez que « Check Placement report to » est coché (un nom de fichier d'extension \*.checkPlace est proposé par défaut) puis cliquez « OK ».

Un rapport de placement a été généré dans la console. Si des violations apparaissent comme dans l'extrait de rapport de la figure 24, un placement plus raffiné est nécessaire.

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

*Fig 24 : Exemple d'un rapport de placement avec violations*

Un placement raffiné est lancé par le menu

### **Place | Refine Placement...**

puis à nouveau, menu

### **Place | Check Placement...**

Si il reste encore des violations de placement, c'est que les contraintes sur le plan d'utilisation des surfaces sont trop importantes. Il faut revoir les dimensions des blocs.

## VIII.7) Génération de l'arbre d'horloge

La génération de l'arbre d'horloge nécessite un fichier de contraintes qui spécifie les caractéristiques du signal d'horloge. C'est dans notre cas le fichier : `${TP_PATH}/asic/par/CONSTRAINTS/ctgen.ctstch` dont le contenu est reproduit figure 25.

### VIII.7.1) Synthèse de l'arbre d'horloge

Menu

**Clock | Synthesize Clock Tree...**

Dans « Clock specification file », saisir `CONSTRAINTS/ctgen.ctstch` puis cliquez « OK ».

Le compte rendu sur l'arbre d'horloge est enregistré dans le fichier `${TP_PATH}/asic/par/clock_report/clock.report`

```
# FirstEncounter(TM) Clock Synthesis Technology
# File Format
#
#-----
# Clock Root : CLK
# Clock Name : CLK
# Clock Period : 20ns
#-----
-
AutoCTSRootPin io_CLK/Y
Period 20ns
MaxDelay 20ns # default value
MinDelay 0ns # default value
MaxSkew 200ps # default value
SinkMaxTran 400ps # default value
BufMaxTran 400ps # default value
Buffer BUF2 BUF4 BUF6 BUF8 BUF12 BUF15
NoGating NO
DetailReport YES
END
```

Fig 25 : Configuration pour l'arbre d'horloge

### VIII.7.2) Visualisation de l'arbre d'horloge

Menu

**Clock | Display | Display Clock Tree...**

- Dans la zone « Clock Selection », sélectionnez « Selected Clock » et renseignez le nom de l'horloge : `io_CLK/Y`. Voir figure 26.
- Dans la zone « Route Selection », sélectionnez « Pre-Route ». Ce choix permet d'afficher des évaluations des temps de propagation et de profondeur de l'arbre d'horloge.
  - Dans la zone « Display Selection », sélectionnez « Display Clock Phase Delay » pour visualiser les différents niveaux de l'arbre d'horloge.
  - Cliquez « Apply ». Dans la fenêtre principale, un ensemble de pastilles colorée indiquent symboliquement (les valeurs numériques sont écrites dans le rapport) le temps de propagation allant du bleu pour le chemin le plus court au rouge pour le chemin plus long.
  - De nouveau dans la fenêtre « Display Clock Tree », sélectionnez la zone « Display Selection » les boutons radio « Display Clock Tree » et « Selected Level »

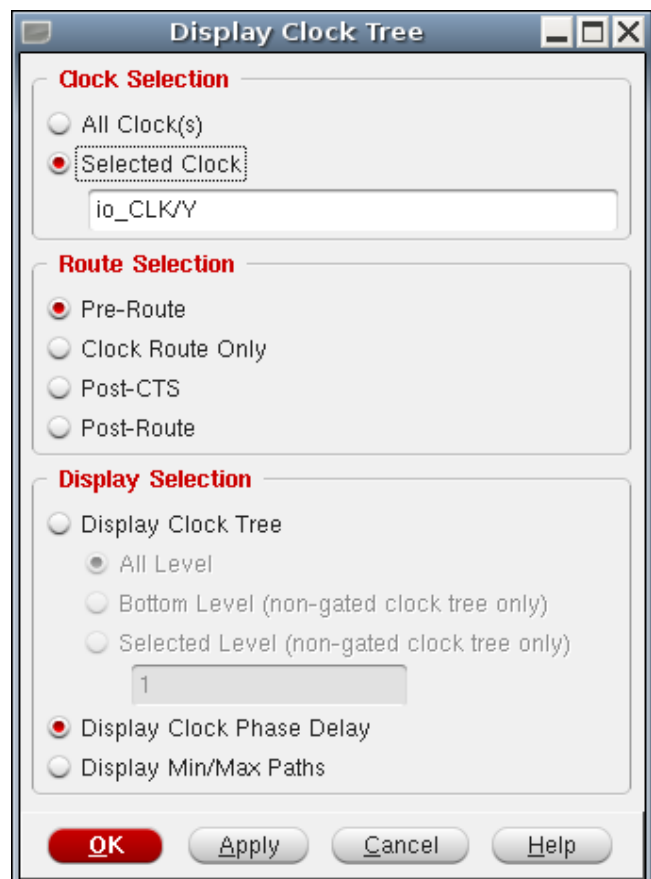


Fig 26 : Visualiser l'arbre d'horloge

- Augmentez progressivement le nombre de niveaux, c'est à dire ici le nombre de fourches franchies, en cliquant « Apply » à chaque fois. Un système de pastilles colorées du bleu au rouge montre les endroits du circuit atteint après le franchissement d'une, deux, trois, etc. fourches sur l'arbre d'horloge.
- Dans la zone « Route Selection », sélectionnez « Post-CTS » et dans la zone « Display Selection », sélectionnez les boutons radio « Display Clock Tree » et « All Level ». Ces choix permettent d'afficher le routage que l'outil propose pour l'arbre d'horloge
- Cliquez « OK » pour fermer la fenêtre « Display Clock Tree ».
- L'arbre d'horloge d'horloge (et ses pastilles colorées) peut être masqué par le menu

**Clock | Display | Clear Clock Tree Display**

**Remarque :** À ce stade l'arbre d'horloge n'est pas encore routé à proprement parlé. Cette étape a juste déterminé la profondeur des arbres (nombre de fourches) en fonction de la disposition des cellules constituant le cœur. Il peut s'avérer nécessaire d'ajouter des étages d'amplification (*buffers*) pour garantir des temps de montées ou remettre en forme le signal d'horloge. C'est l'objet de l'étape d'optimisation cellules en place (*In-Place Optimization Post Clock-Tree Synthesis – IPO post-CTS*).

## VIII.8) Remplissage des vides

Au cours de la fabrication, les dépôts et les enlèvements de matière se succèdent. Des procédés différents sont utilisés selon la quantité de matière à déposer ou retirer. Pour qu'un procédé soit le plus efficace possible, les surface à traiter doivent présenter une densité de motif la plus régulière possible. En effet, de petits motifs isolés seront gommés si un procédé destiné à réaliser de grands motifs est employé tandis que de larges motifs ne seront pas correctement reproduits si des procédés prévus pour garantir la finesse d'un petit motif est employé. C'est pourquoi, il est nécessaire de combler les vides qui peuvent exister entre les cellules placées dans le cœur ou ceux qu'il peut y avoir entre les plots dans la couronne.

Les vides sont comblés par des remplisseurs (*fillers*) qui sont des cellules vides (et inactives) pour le cœur, et des plots vides (inutilisés) pour la couronne de plots.

Ces opérations de remplissage vont être réalisées par l'exécution de scripts lancés depuis la console de SoC Encounter :

- pour le remplissage du cœur, la commande  
`amsFillcore`
- pour le remplissage de la couronne de plots, la commande  
`amsFillperi`

## VIII.9) Routage effectif

Le circuit est désormais prêt pour le routage cependant il est impératif de commencer par les signaux spéciaux, aux contraintes les plus fortes. Le routage se fait donc en deux temps, d'abord l'arbre d'horloge puis le reste du circuit.

Le routage est réalisé en deux étapes : le routage global (*Global Routing*) et le routage détaillé (*Detailed Routing*). Le routage global est destiné à évaluer les zones de congestion potentielles et par suite à les éviter. Pour cela l'outil découpe le design en blocs rectangulaires appelés cellules de routage globales (*global routing cells – gcells*), auxquelles il leur associe des nœuds correspondant aux signaux traversant ces blocs. Ensuite, il estime la possibilité de réussir le routage de ces cellules globales mais ne trace aucune piste. Une carte de congestion peut être générées. Le routage détaillé utilise les informations du routage global pour effectivement tracer les pistes d'interconnexion entre les différentes cellules qui composent le circuit.

### Routage de l'arbre d'horloge

Pour router l'arbre d'horloge, lancer, successivement depuis la console, les commandes suivantes. La première sert à sélectionner les nœuds définis comme des horloges (dans notre cas il n'y en a qu'une). La deuxième spécifie de router uniquement les nœuds sélectionnés (donc ici l'horloge). La troisième indique d'effectuer les deux étapes de routage en une seule opération (la carte de congestion ne peut donc pas être visualisée).

```
selectNet -allDefClock
```



```
setNanoRouteMode -quiet -routeSelectedNetOnly true  
globalDetailRoute
```

Visualisez l'arbre d'horloge et identifiez les connexions du signal CLK.

### Routage complet du circuit

Lancer le routage en exécutant la commande suivante :  
amsRoute

## VIII.10) Vérification

Après le routage, il est nécessaire de lancer une vérification sur les règles de dessins ainsi que sur les différentes connexions.

Menu

### Verify | Verify Geometry...

Puisque le placement a été fait en s'alignant sur la grille du fondeur, il faut s'assurer que (cf. figure 27) :

- « Off Routing Grid » est décoché
- « Off Manufacturing Grid » est coché

Menu

### Verify | Verify Connectivity...

Pour vérifier qu'il ne reste pas d'éléments non connectés (cf. figure 28).

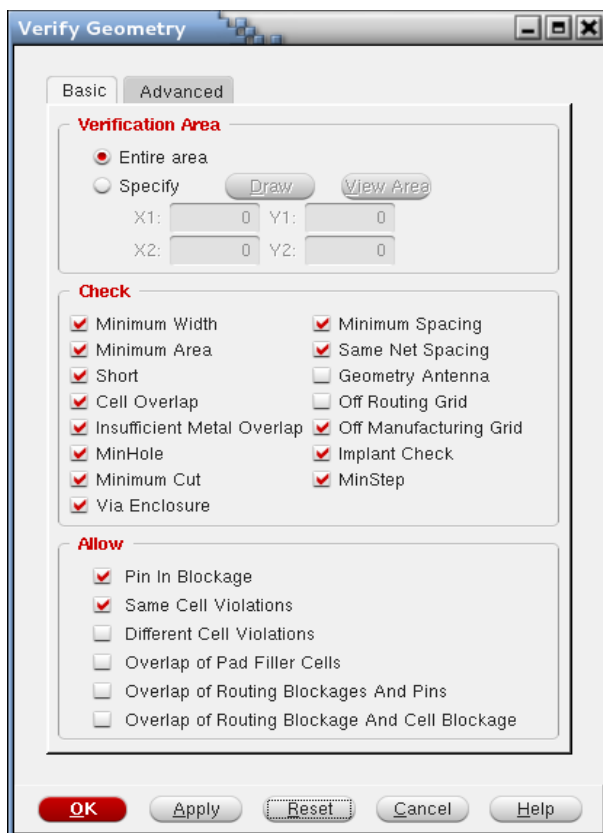


Fig 27 : Vérifications géométriques

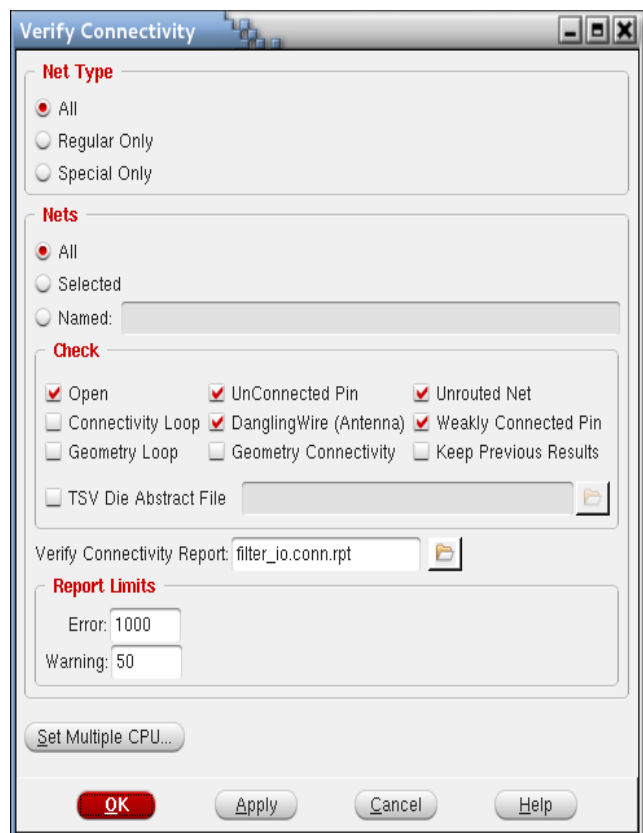


Fig 28 : Vérifications des connexions

## VIII.11) Caractéristiques du circuit après P&R cible ASIC

Les principales caractéristiques du circuit (surface globale, nombre de cellules, nombre d'interconnexions, etc.) peuvent être rassemblées dans un récapitulatif (*Summary Report*).

Menu

**File | Report | Summary...**

## VIII.12) Exportations des données de placement et routage

Après le placement et le routage, la réalisation du circuit se poursuivra par une simulation après placement et routage puis, si elle est satisfaisante, par le tracé des plans des masques (*layout*). Pour mener à bien ces opérations qui suivent ont besoin des informations sur les délais des interconnexions qui sont fortement influencés par la présence de RC parasites des interconnexions.

Il faut donc d'abord procéder à l'extraction des RC parasites (*parasitic extraction*) puis enregistrer les informations les délais des interconnexions. Le format de fichier utilisé est le *Standard Delay Format* – SDF .

Menus

**Timing | Extract RC...**

**Timing | Write SDF...**

Il est pertinent d'utiliser des extensions qui rappellent le type d'extraction.

Les fichiers permettant l'échange de données avec les outils de simulations et de tracé des masques doivent ensuite être générés :

- une description rétroannotée du circuit routé, dans un langage de description matériel, pour la simulation post placement et routage. SoC Encounter produit des fichiers en Verilog.
- une vue physique du circuit (description électrique) pour les vérifications DRC (*Design Rule Check*) et LVS (*Layout Versus Schematics*). Le format standard le plus classique est le GDSII (\*.gds). Le format OASIS est une version plus moderne qui devrait remplacer le format GDSII dans les prochaines années.

Le fondeur AMS fourni un script qui génère ces fichiers. Pour l'exécuter, taper la commande suivante dans la console :

```
amsWrite <postfix>
```

L'argument **postfix** est un suffixe ajouté automatiquement aux noms des fichiers générés. Il est recommandé d'utiliser le suffixe **par** pour distinguer les fichiers après placement et routage.

L'extrait ci-dessous de la documentation AMS indique quel fichier est généré et où il est placé :

- SDF/<topcell-name>\_<postfix>.spef: Encounter Parasitic Extraction File
- SDF/<topcell-name>\_<postfix>\_qrc.spef: Fire&Ice Parasitic Extraction File
- <topcell-name>\_<postfix>\_fe.gds: GDSII File
- <topcell-name>\_<postfix>.v: Verilog Netlist
- <topcell-name>\_<postfix>\_fillcap.v: Verilog Netlist including core filler cells with decoupling caps
- DB/<topcell-name>\_<postfix>.enc: Encounter Database

Déplacer les fichiers Verilog qui viennent d'être générés directement dans le répertoire {TP\_PATH}/par vers le répertoire \${TP\_PATH}/par/VERILOG.

Le produit phare de la société Cadence est Virtuoso, un outil modulaire pour le tracé des plans de masque (*layout*) et pour la simulation électrique (*spice-like simulation*). Cadence utilise ses propres formats de données pour l'échange entre ses outils : *Design Exchange Format* (DEF) qui est appelé à être prochainement remplacé par le format *Open Access* (OA). L'exportation du circuits placé et routé dans ces formats est possible par la biais des menus « File | Save Design » et « File | Save | DEF ».

Quitter SoC Encounter avant de continuer le TP.

### VIII.13) Simulation après placement et routage

La netlist obtenue après placement routage (en Verilog) peut être simulée, avec rétroannotation des informations contenues dans le fichier SDF. Ce test peut être effectué avec le même fichier de test que lors de la synthèse avant placement et routage. Toutefois, avant le placement et routage, le fichier de test se connectait sur la description du cœur (module FILTER). Il faut désormais le connecter sur la puce complète (module filter\_io). Le fichier de test doit donc être modifié de façon à remplacer le composant FILTER par le composant filter\_io.

La simulation après placement et routage sur Asic est très similaire à celle sur FPGA. Vous avez à créer un nouveau script de compilation avec Modelsim pour le circuit routé. Toutefois, la description post placement et routage produite par SoC Encounter étant Verilog, le compilateur de ModelSim à utiliser est vlog à la place de vcom, les options sont les mêmes pour ces deux compilateurs (voir le miniguide sur ModelSim). L'ensemble des blocs se trouvent alors dans la même bibliothèque.

Le simulateur est toujours démarré par vsim.

Lors du lancement de la simulation (menu « Simulate | Start Simulation... »), dans la fenêtre « Start Simulation » :

- onglet « SDF », sélectionner le fichier SDF produit au §VIII.12
- onglet « Libraries » puis bouton « Add », choisir la bibliothèque technologique c35\_CORELIB dans la liste déroulante de la fenêtre « Select Library ». Recommencer en ajoutant la bibliothèque technologique c35\_IOLIB\_4M.

### VIII.14) Travail demandé pour la partie placement et routage cible ASIC

- Effectuer le placement routage du circuit sans erreurs et violations en consignait et commentant précisément les différentes étapes dans un compte rendu. 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 le placement-routage (Verilog + SDF). Comprendre l'intérêt d'un fichier de type \*.sdf
- Dans votre compte rendu final traitez soigneusement cette dernière partie en expliquant en détails toutes ces étapes et faire des commentaires de ce que vous avez vu et compris.