

Flot de synthèse logique VHDL sur ASIC


avec *Cadence* et *Synopsys*

v5.32 – Novembre 2020

Table des matières

1	Introduction	3
2	Configuration du compte et lancement des logiciels	3
2.1	Configuration initiale	3
2.2	Lancement des logiciels	3
3	Synthèse élémentaire.	4
4	Simulation de la montre.	6
4.1	Simulation sous NcSim	7
5	Préparation de la synthèse de la montre	7
5.1	Fichier technologique	7
5.2	Chargement des module en vue de la synthèse	7
6	Application des contraintes de synthèse	8
6.1	Contraintes environnementales	8
6.1.1	Ports de sortie	8
6.1.2	Ports d'entrée	8
6.1.3	Modèle de charge pour les connexions	8
6.1.4	Conditions de fonctionnement	9
6.1.5	Définition de l'horloge	9
6.1.6	Délais sur les entrées et sorties	9
6.2	Contraintes d'optimisation	9
7	Synthèse	10
7.1	Mise en place de stratégies de compilation	10
7.2	Rapports de synthèse	11
7.3	Comparaison des résultats	11

8	Post-simulation	11
A	Cahier des charges	13
A.1	module séquentiel	13
A.2	module combinatoire	13
B	Codage BCD	13
C	Utilisation de NcSim	13

Vous avez la possibilité de sélectionner le texte écrit dans ce sujet (icône ) , notamment pour copier/coller les commandes demandées et ainsi éviter les erreurs de frappe. Cependant, suivant le lecteur PDF utilisé, le copier/coller peut ajouter des espaces là où il n'y en a pas et il faut donc plutôt recopier les commandes à la main...

1 Introduction

Le but de ces travaux pratiques est de réaliser un système numérique simple, sur circuit intégré, à partir d'une description de son fonctionnement à haut niveau. Le langage de description utilisé pour la description comportementale et la simulation initiales est VHDL.

L'outil de synthèse logique Synopsys est utilisé pour effectuer une conversion de la description comportementale vers une description structurelle, avec toutes les références nécessaires aux cellules pré-caractérisées de la bibliothèque de la technologie utilisée. À l'issue de cette étape, un premier retour peut-être effectué sur la schématique de départ pour prendre en compte une première série d'informations supplémentaires (impédances réelles des ports d'entrée/sortie des cellules utilisées et estimation statistique du routage (qui n'est pas encore fait à ce stade)). Une rétro-simulation doit alors vérifier que le cahier des charges est toujours respecté. Enfin, un placement/routage sur silicium doit être effectué : à partir de lui une extraction réaliste des éléments parasites liés au routage devient possible, à la suite duquel une deuxième rétro-simulation, prenant cette fois en compte tous les éléments parasites du dessin de masques, est nécessaire pour vérifier le bon fonctionnement avant fabrication.

2 Configuration du compte et lancement des logiciels

2.1 Configuration initiale (à ne faire qu'en début de 1^{re} séance)

On crée un répertoire de travail, en le copiant d'un modèle. Ouvrir une fenêtre Shell et copier/coller ou taper (y compris le point final!) :

Shell

```
cp -r /home/tournier/FlotSynthese .
```

Cette commande crée un répertoire **FlotSynthese** et des sous répertoires **Simulation**, **Synthese** et **PlacementRoutage** contenant déjà des fichiers de configuration (dont le nom commence par un point) pour chacun des logiciels utilisés. Chaque logiciel doit être lancé depuis le répertoire qui lui est destiné. On y placera également les fichiers de travail de ce TP.

2.2 Lancement des logiciels

À chaque étape, on ouvre une nouvelle fenêtre shell, on entre l'alias de configuration du logiciel ainsi que celui de la technologie, on descend dans le bon répertoire, puis on lance le logiciel par la commande appropriée, ainsi que le résume ce tableau :

Étape	Simulation	Synthèse logique
Alias de configuration du Shell	incisive152	synopsys13
Alias d'accès à la technologie AMS	ams411	ams411
Répertoire	FlotSynthese/ Simulation	FlotSynthese/ Synthese
Commande de lancement du logiciel	nclaunch	design_vision-xg

TABLE 1 – Configuration des différentes étapes

La technologie que nous allons utiliser est choisie parmi les technologies CMOS disponibles du fondeur AMS : il s'agit de la technologie CMOS 0,18 μ m 1,2 V.

3 Synthèse élémentaire.

À faire

Lancer le logiciel de synthèse

La fenêtre de la fig. 1 doit apparaître, vide pour l'instant. Cette interface graphique de Synopsys n'est pas indispensable à la synthèse. Elle n'est de plus pas conseillée lorsque l'on a affaire à des systèmes professionnels de tailles importantes : on travaille alors plutôt par scripts de commande. Dans notre cas, elle reste cependant le meilleur moyen de visualiser étape par étape l'ensemble du processus de synthèse, lorsqu'on n'est pas encore familier avec les scripts et les commandes Synopsys.

La partie « console » permettra, dans un premier temps, d'associer à chaque commande « graphique » son équivalent en syntaxe Synopsys. Avec l'habitude, on pourra directement rentrer les commandes sur la ligne de commande **design_vision** une par une, voire à travers un script de commandes (File puis **Execute Script...**).

Deux étapes sont nécessaires pour procéder à une synthèse de modules décrits au format VHDL dans un fichier texte : l'analyse et l'élaboration.

On choisira pour commencer d'illustrer deux planches de cours : page 20 avec l'utilisation de parenthèses, et page 22 avec le partage de ressources.

À faire

Écrire dans un premier fichier texte "**parenthese.vhd**" les lignes VHDL de la page 20 illustrant l'utilisation de parenthèses, puis dans un second fichier texte "**ressource.vhd**" les lignes de la page 22 illustrant l'effet du partage de ressource, en écrivant les codes de partage forcé et interdit. Bien sûr, dans chacun de ces deux fichiers, les lignes VHDL recopiées doivent être insérées dans une architecture complétée par l'entité correspondante. On pourra s'inspirer du fichier "**afficheur.vhd**" déjà présent dans le répertoire pour le formatage général du code VHDL et l'appel des librairies. Pour faciliter la comparaison, on écrira les codes à comparer dans la même architecture, en utilisant les mêmes entrées, mais bien sûr en déclarant deux sorties différentes pour chacun. Le **type** des signaux pourra par exemple être "**integer**".

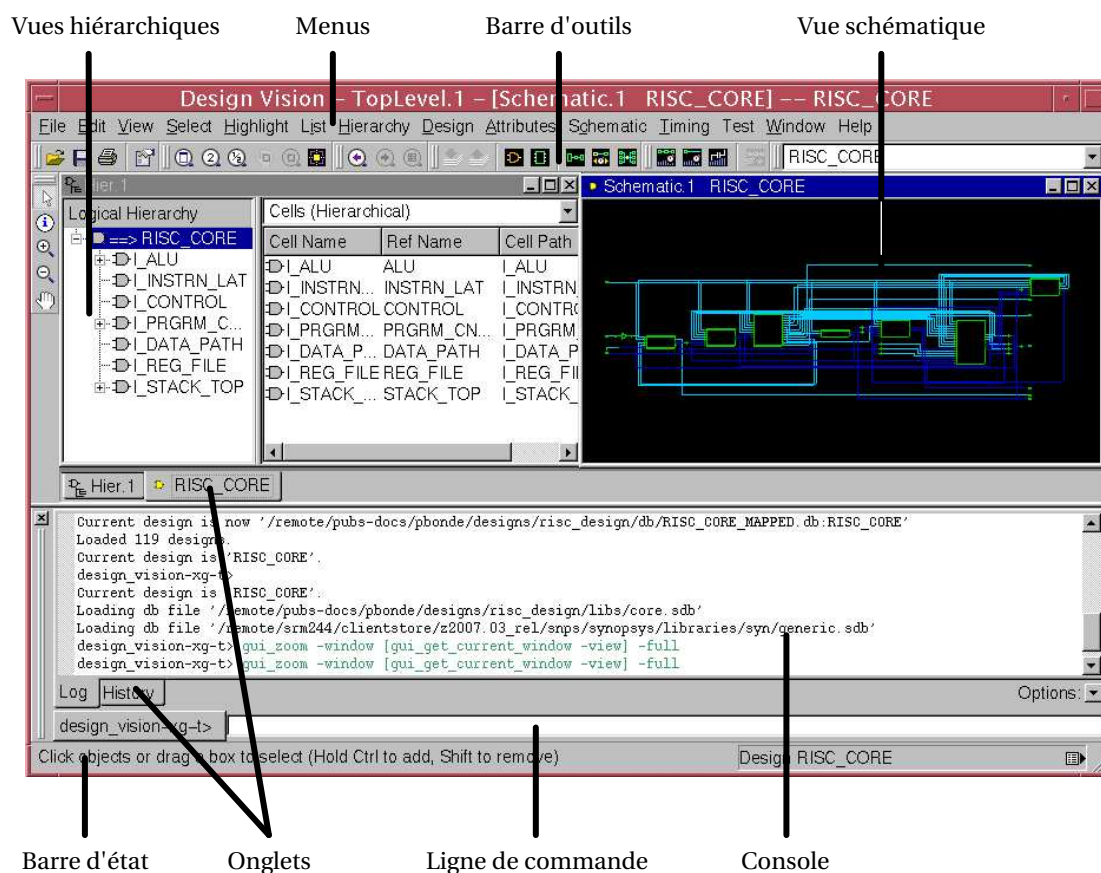


FIGURE 1 – Fenêtre graphique

1.

Menu Synopsys

Aller dans Files puis Analyze.... Sélectionner les fichiers textes créés.

La syntaxe VHDL est alors vérifiée, ainsi que la logique sur son aspect synthétisable. Tout message d'erreur généré à ce stade implique qu'il faille changer le code VHDL à cause d'erreurs de syntaxe ou parce qu'il n'est pas synthétisable. Si aucun message d'erreur n'apparaît, le module est placé dans la librairie de travail (DEFAULT ou WORK et se trouve physiquement dans le sous-répertoire **work_synopsys**) dans un format propre à Synopsys.

2.

Menu Synopsys

Aller dans Files puis Elaborate... Sélectionner le module analysé à ouvrir



Le module apparaît dans la fenêtre principale, ainsi que ses dépendances éventuelles (qu'il aura fallu analyser aussi avant).

Il est possible de visualiser la hiérarchie de la cellule élaborée dans la fenêtre **Logical Hierarchy...**

À faire

Cliquer (clic droit) sur un module afin d'en récupérer les informations.

Le menu contextuel d'une cellule permet d'en afficher la schématique sur la droite (Schematic View...).

Le dessin sur chaque icône en représente le contenu : comportemental (symbole avec une lettre G : ) ou structurel (symbole de porte ET : .

La schématique laisse apparaître des fonctions génériques indépendante de la technologie (c'est normal, la synthèse n'a pas encore eu lieu!). Ce sont des modules génériques *structurels* de fonctions plus ou moins complexes mais connues (porte logique, additionneur, comparateur, ...), sans référence technologique, qui font partie de la bibliothèque de connaissance de Synopsys. Suivant le degré de complexité ou de sophistication attendu, une librairie « standard » est utilisée ou bien des librairies qui contiennent des architectures avancées (et donc payantes...).

À faire

Vérifier à ce stade que l'utilisation des parenthèses et le forçage ou non du partage de ressources a bien eu une influence sur la schématique.

Lancer une synthèse élémentaire sans contrainte :

À faire

Aller dans Design puis Compile design... Laisser les options par défaut et valider.

Remarquer que tous les icônes sont maintenant de la forme structurelle (🔍).

À faire

Relever les valeurs de l'aire du circuit (Design, Report Area...)

Question

Quel gain de place a permis de réaliser le partage de l'additionneur ?

4 Simulation de la montre.

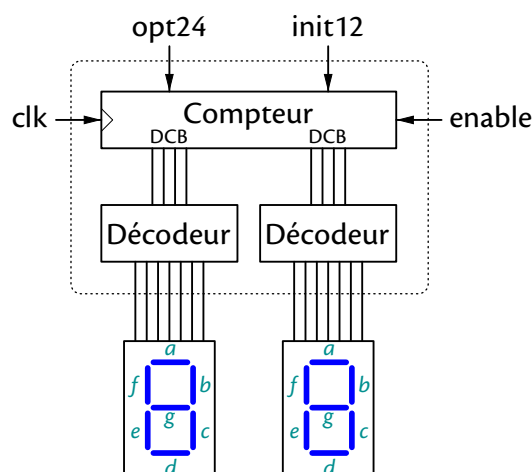


FIGURE 2 – schéma de principe

Le cahier des charges de la montre est rappelé en annexe A et sur la fig. 2.

4.1 Simulation sous NcSim

Avant de simuler et de chercher à valider les descriptions VHDL que vous avez normalement écrits avant d'arriver en TP, il est *indispensable* de vérifier d'abord que le circuit est synthétisable (sans le synthétiser pour autant, pour le moment). Il serait inutile d'optimiser un code VHDL qui à l'arrivée ne pourrait être synthétisé.

À faire

Après des modifications significatives de code, effectuer l'analyse des codes VHDL par Synopsys avant de les simuler avec NcSim (et seulement l'analyse !).

Si les codes génèrent une erreur d'analyse, il faut les modifier jusqu'à les rendre synthétisables.

À faire

Mettre au point les codes VHDL en simulant avec NcSim

On se référera à l'annexe C pour l'utilisation du simulateur NcSim. Pour vous aider, dans le répertoire *Simulation* se trouve un fichier `montre.vhd` qui contient la connexion du compteur, des décodeurs et des afficheurs, ainsi qu'une petite procédure de test pour la validation. On modifiera ce fichier pour y corriger les noms des entités/architectures afin de les faire correspondre aux vôtres. Les différents commentaires doivent vous aider à adapter ce fichier à vos modules.

Une fois que la simulation donne satisfaction, on revient à Synopsys pour effectuer la synthèse.

→ Pour le moment, allez à l'annexe C pour prendre en main le simulateur NcSim, simuler et valider vos codes VHDL.

5 Préparation de la synthèse de la montre

5.1 Fichier technologique

On peut prendre connaissance du principal fichier technologique du fondeur à destination de Synopsys avec la commande :

Shell

```
> t /soft/DK/ams411/liberty/c18_1.2V/c18_IOLIB_TYP.lib
```

dans lequel on consultera notamment quelques descriptions de cellules disponibles, et surtout les unités de base des différents paramètres communs à toutes les cellules (`timeunit`, `voltageunit`, `currentunit`, ...).

5.2 Chargement des module en vue de la synthèse

À faire

Reprendre le fichier décrivant le module de haut niveau ("`montre`"), le modifier pour l'adapter à la synthèse, et le sauvegarder dans un fichier "`top_synth.vhd`". Renommer également l'entité en "`top_synth`".

À faire

Analyser le compteur, le décodeur 4 bits-7 segments, et la montre.

À faire

Élaborer la montre.

Question

Pourquoi ne doit-on pas charger le module d'affichage ?

6 Application des contraintes de synthèse

Il s'agit dès maintenant de fixer les contraintes et attributs inhérents au circuit et à son environnement, ainsi que ceux souhaités pour la synthèse. Bien sûr, vous devez partir des modules originaux, qui n'ont pas encore subi de tentative de synthèse. Mieux vaut donc faire **Files**, **Remove All Designs**, puis à nouveau un **Elaborate...** de votre module de haut niveau, afin de repartir d'un état vierge.

6.1 Contraintes environnementales

6.1.1 Ports de sortie

Les ports de sortie seront connectés sur des nœuds avec capacité parasite. Il s'agit donc de sélectionner les ports de sortie et d'indiquer la valeur de cette charge capacitive. Lorsque le port est un bus, on peut différencier les valeurs pour chaque bit.

Sélectionner tous les ports de sortie, et indiquer la valeur de 1 (**Attributes, Operating Environment, Load...**). Dans la fenêtre de commande, on voit quelles sont les commandes scripts équivalentes à la boîte de dialogue qu'on vient de remplir.

6.1.2 Ports d'entrée

Les ports d'entrée du circuit seront commandés avec une certaine résistance de commande (**Drive Strength**). On directement la valeur de cette résistance de commande.

Sélectionner tous les ports d'entrée (avec la touche CTRL), et indiquer la valeur de 1 (**Attributes, Operating Environment, Drive strength...**).

En fait cette fonction apparaît bugué au niveau de l'interface graphique : on n'a pas la possibilité d'entrer une valeur numérique ! Il faut alors rentrer la ligne de script équivalente en s'aidant du code généré précédemment pour indiquer les charges de sorties, et en remplaçant `set_load` par `set_drive`.

6.1.3 Modèle de charge pour les connexions

Suivant la taille du circuit, on peut extraire de façon statistique une estimation pour la charge (R , C) de chaque ligne de connexion, en plus de la charge due à la connexion suivante. Ce modèle statistique permet de dégrader

un peu la qualité des signaux puisque c'est inévitablement ce qui se produira lors du placement/routage des cellules, et ce bien qu'il n'ait pas encore été effectué. Le but est de rester réaliste en évitant d'ignorer complètement à ce stade l'effet non négligeable du placement/routage.

On choisira 10k de la librairie "core" (voir la définition dans le fichier précédent, correspond à une taille de circuit finale de 10000 portes) (**Attributes, Operating Environment, Wire load...**). Si un message d'erreur apparaît, il s'agit encore d'un bug du logiciel. On le résout en quittant le logiciel, en le relançant, et en exécutant immédiatement cette commande (et celle qui suit) sans avoir fait afficher la schématique du circuit.

6.1.4 Conditions de fonctionnement

Diverses conditions de fonctionnement peuvent être imposées au circuit. Là également, les définitions correspondantes sont données dans le fichier précédent et regroupées sous des noms clefs. On choisira le mode **Typical** (dans **Attributes, Operating Environment, Operating conditions...**) de la librairie "core".

6.1.5 Définition de l'horloge

La première chose à faire avant de définir d'éventuelles contraintes temporelles est de définir l'élément de référence qui est bien sûr l'horloge (**Attributes, Specify clock...**).

On choisira une horloge de période 5 ns.

6.1.6 Délais sur les entrées et sorties

Le délai sur une entrée est le délai que subit un signal entre le front actif de la dernière bascule du module commandant le module à contraindre, et cette entrée. Il est donc défini à la droite du front actif et vient se cumuler au délai, dans le module à contraindre, depuis cette entrée jusqu'au front actif de la première bascule partageant la même horloge.

Le délai sur une sortie est le délai que subit le signal entre cette sortie et le front actif de la première bascule du circuit commandé. Il est défini à la gauche du front actif et vient se cumuler au délai, dans le module à contraindre, depuis la dernière bascule partageant la même horloge jusqu'à cette sortie.

Définir 0,2 ns pour les délais sur les entrées et les sorties.

6.2 Contraintes d'optimisation

Une fois que les contraintes environnementales ont été définies (c.-à-d. les contraintes non maîtrisées par le concepteur), il est possible de définir un jeu de contraintes supplémentaire, fixé cette fois par le concepteur, et qui devient alors un jeu de contraintes d'optimisation. Notamment, on peut donner comme contrainte d'optimisation une aire d'intégration minimale.

Aller dans **Attributes, Optimization Constraints..., Design Constraints...**

On indique **Max Area : 0.0** pour imposer une contrainte d'aire minimale. Puis **Apply**.

7 Synthèse

Avant de lancer une synthèse, on sauvegardera l'ensemble des contraintes appliquées dans un fichier `setup.dc` que l'on pourra réexécuter ensuite sous forme de script pour réappliquer toutes les contraintes d'un coup.

Menu Synopsys

Aller dans File > Save Info > Design Setup

Entre chaque style de synthèse, il faudra redémarrer d'un état vierge afin d'être sûr de repartir du même état comportemental de départ, pour pouvoir comparer correctement les types de synthèses. Pour repartir du même état vierge, il suffit de faire :

Menu Synopsys

File > Remove All Designs

pour tout effacer, puis faire à nouveau un «elaborate» du module de haut niveau, puis :

Menu Synopsys

File > Execute Script...

en choisissant `setup.dc`. Une nouvelle synthèse peut alors être effectuée.

7.1 Mise en place de stratégies de compilation

Les stratégies de synthèse seront les suivantes :

- descendant (uniquify automatique);
- mise à plat;
- uniquify manuel + ascendant.

Les différents critères de jugement seront :

- l'aire du circuit;
- le temps de propagation du chemin critique du compteur;
- le temps de propagation du décodeur.

Dans le cas de la synthèse descendante, on sélectionne le bloc de plus haut niveau, puis on lance la synthèse : Synopsys se charge de parcourir la hiérarchie et de synthétiser tout ce qu'il trouve (et qui n'est pas figé par l'option `setdonttouch`). Dans le cas de la synthèse ascendante, il faut :

- sélectionner chaque bloc de bas niveau et le synthétiser.
- sélectionner chaque bloc du niveau supérieur et le synthétiser.
- resynthétiser les modules de plus bas niveaux afin de prendre en compte les nouvelles contraintes issues de la synthèse du niveau supérieur.
- monter au niveau encore supérieur et ainsi de suite jusqu'à convergence : c'est plus lourd à manipuler mais offre plus de degrés de liberté dus aux compilations séparées des blocs.

À l'issue de chaque résultat de synthèse, on sauvegardera dans un répertoire à part, un fichier au format DDC (*.ddc) qui est le format propre de Synopsys. Pour la synthèse descendante, on sauvegardera en plus :

- au format VHDL (*.vhd) qui permettra une première rétro-simulation après synthèse;
- au format VERILOG (*.v) à destination du logiciel de placement/routage si nécessaire.

7.2 Rapports de synthèse

Il est possible de générer des informations concernant les résultats de synthèse. Aller dans les menus **Design** ou **Timing**, et choisir un des **Report XXX...** Relever notamment les valeurs de l'aire du circuit (**Design, Report Area...**), du chemin critique du compteur et du temps de propagation à travers le décodeur (**Timing, Report Timing Path...**).

On remarquera que la fenêtre de rapport contient du texte en bleu qui est cliquable et qui conduit directement à un emplacement précis de la schématique.

Pour obtenir les rapports de timing concernant un décodeur, le choisir d'abord dans la barre d'icône (ni sur le schéma, ni dans la liste hiérarchique!).

Pour l'exemple, on pourrait chercher à réduire le temps de propagation à travers le décodeur. Appliquer une contrainte sur le temps de propagation maximum (par exemple 0,2 ns) entre les entrées et les sorties d'un des deux décodeurs et en relancer la synthèse.

Question

À quelle valeur minimale peut-on aboutir ?

7.3 Comparaison des résultats

Normalement, VHDL est capable de lier différentes instances d'un même composant à plusieurs architectures d'une même entité via les configurations :

Shell

```
for <instance>: <component> use entity <library>.<entity>(<architecture>);
```

Cependant, Synopsys refuse ces dernières, et dans ce cas, pour lier un composant à une entité, il faut qu'ils portent le même nom, de même que leurs déclarations de ports doivent être identiques (faire un copier /coller des ports de l'entité vers ceux du « component » pour être sûr). L'architecture utilisée est alors la première compilée! Impossible dans ce cas, sans configuration, d'avoir plusieurs architectures possibles pour un même nom d'entité.

À l'issue d'une synthèse, n'importe laquelle, SYNOPSYS renomme toutes les architectures comportementales qu'il a synthétisé en leur rajoutant le préfixe « SYN_ », identique pour les trois. Les choses deviennent alors compliquées si l'on souhaite simuler en parallèle des architectures différentes pour une même entité, pour les comparer entre elles (SYNOPSYS n'écrit pas non plus de configurations).

Pour simplifier, on ne comparera que l'architecture issue de la synthèse descendante avec l'architecture comportementale de référence.

8 Post-simulation

Après la synthèse, il faut vérifier que les circuits synthétisés se comportent toujours comme souhaité. On n'effectuera la vérification que sur le résultat de la synthèse descendante. On reprendra l'environnement de simulation

mis au point avant synthèse, auquel on ajoutera une nouvelle instance de composant « top_synth » dont l'architecture sera bien sûr celle issue de la synthèse descendante. Ne pas oublier de connecter des afficheurs à chaque nouvelle sortie (figure 3).

Avant de pouvoir simuler, il faut avoir référencé tous les composants de la technologie utilisée, ce qui a été fait automatiquement par Synopsys lorsqu'il a généré ses codes VHDL.

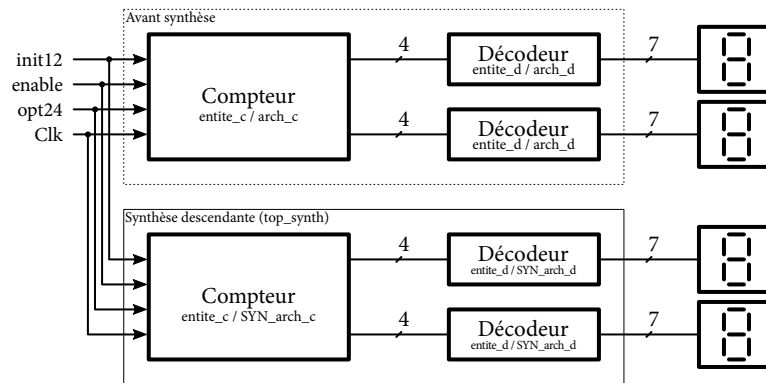


FIGURE 3 – Schéma de principe

À quelle fréquence décrochent les circuits ?

Question

Pourquoi ce décrochage ne pouvait-il être testé avant synthèse ?

Question

Pensez-vous qu'il soit nécessaire de sauvegarder les informations de **"timing"** à ce niveau ?

Question

Pensez-vous que nous pouvons d'ores et déjà conclure que nous avons déterminé la fréquence maximale d'utilisation ?

A Cahier des charges

On cherche à réaliser un module gérant la partie « heure » d'une montre digitale.

A.1 module séquentiel

Faire un compteur binaire BCD (« Binary Coded Decimal ») synchrone, avec un cycle de 12 ou 24 états selon l'état de l'entrée **opt24**.

- lorsque **opt24** = 0, le compteur compte de 1 à 12,
- lorsque **opt24** = 1, le compteur compte de 0 à 23.

Il sera muni d'une entrée **enable** et d'une entrée **init12**, qui initialisera le compteur à l'état 12 de manière synchrone.

L'entrée **init12** est active même en l'absence d'**enable**.

A.2 module combinatoire

Faire un décodeur 4 bits \rightarrow 7 segments.

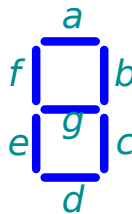


FIGURE 4 – Numérotation des segments

B Codage BCD

Le « Décimal Codé Binaire » (BCD ou « Binary Coded Decimal ») est un code de représentation des nombres dans les systèmes numériques. Dans ce code, chaque chiffre de la représentation décimale est codé sur un groupe de 4 bits. *Exemple* : 1664 se code $\underbrace{0001}_1 \underbrace{0110}_6 \underbrace{0110}_6 \underbrace{0100}_4$

Avantage : affichage décimal grandement facilité

Inconvénient : code redondant (toutes les combinaisons ne sont pas utilisées)

(*Exemple* : 1664 prend 16 bits en BCD, seulement 11 en binaire naturel)

C Utilisation de NcSim

1) Introduction

Il est possible de lancer l'environnement de simulation en tapant ***nclaunch &***.

Une fenêtre similaire à la figure 1 doit apparaître à l'écran. Cet environnement, ou Framework, permet :

- ➡ Edition de fichier texte .vhd
- ➡ Compilation dans la bibliothèque
- ➡ Lancement de l'outil de Simulation

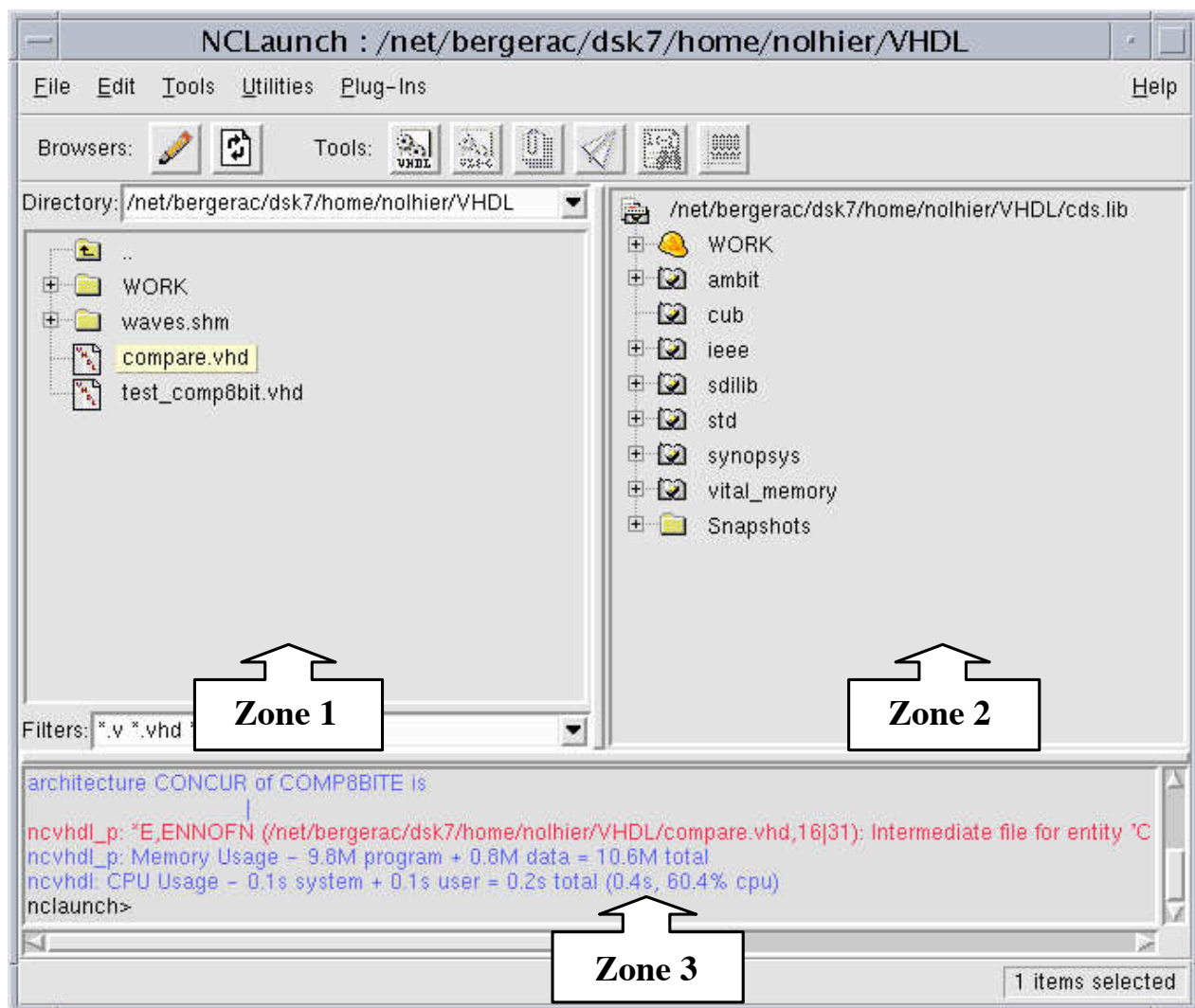


Figure 1 : Fenêtre de l'environnement de VHDL

La fenêtre comporte plusieurs zones :

- ➡ Zone 1 : affiche les fichiers texte des descriptions VHDL.
- ➡ Zone 2 : affiche la liste des descriptions VHDL déjà compilées dans la librairie de travail WORK ainsi que le contenu des autres librairies.
- ➡ Zone 3 : Une zone où sont affichés les erreurs, les warnings, et les résultats de compilation ou autre.

Configuration du compilateur

Pour configurer le compilateur sélectionner  **Tool** ➡ **VHDL Compiler**. Apparaît alors la fenêtre suivante (figure 2). Il faut **impérativement** cocher l'option ***Enable VHDL'93***.

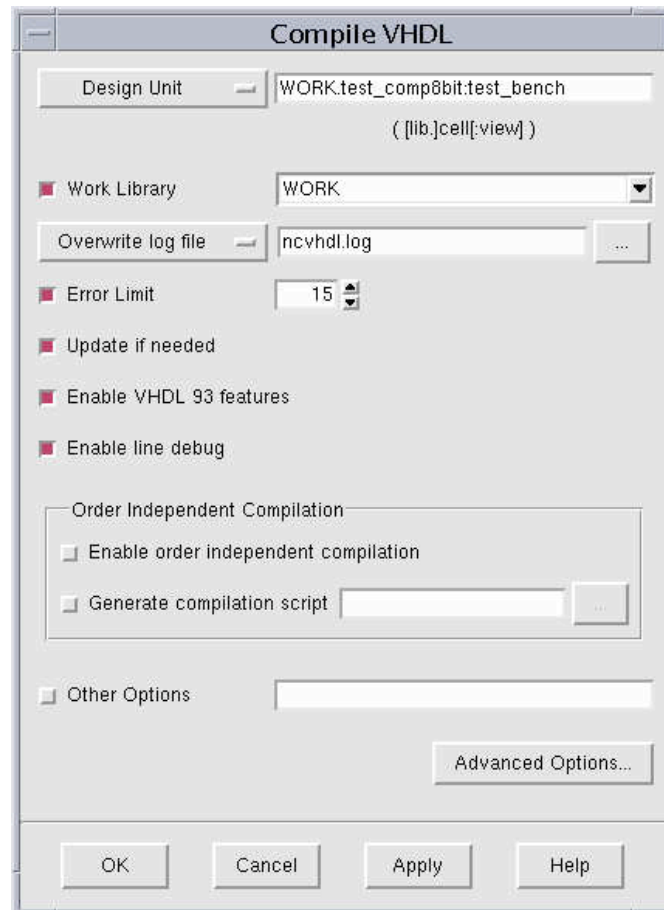



Figure 2 : Mode de compatibilité

2) Edition d'un fichier de description VHDL

a) cas d'un nouveau fichier

- dans le menu **File** sélectionner  **Edit New File**. La fenêtre **Edit a New File** doit apparaître (Figure 3).
- taper le nom du nouveau fichier avec une extension **.vhd** (ex: bascule.vhd)

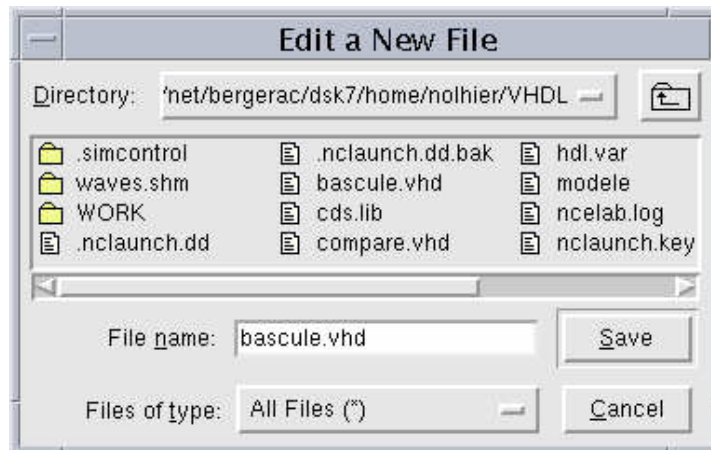


Figure 3 : Fenêtre de création de nouveau fichiers

- La fenêtre de l'éditeur est alors à l'écran, taper la description VHDL de votre modèle
- Pour enregistrer le fichier, sélectionner dans le menu **File** de nedit la commande **Save**.


b) cas d'un fichier déjà créé.

Le sélectionner dans la zone 1 et cliquer  sur l'icône du crayon .

3) Compilation d'une description VHDL






Le fichier d'entrée .vhd peut contenir, soit une entité, soit des architectures, soit le tout. Quand une description est compilée, et qu'il n'y a pas d'erreurs de compilation, la librairie reçoit deux types de composants : l'entité du modèle et une ou plusieurs architectures.

➡ pour compiler un modèle VHDL, sélectionner  le fichier **.vhd** dans la zone puis cliquer sur le bouton de commande **VHDL COMPILER**  ou alors double-cliquer  sur le fichier.

 Lors de la compilation les erreurs sont reportées dans la zone 3. Il faut alors rééditer le fichier source (.vhd), apporter les corrections nécessaires, et relancer une compilation.

4) Simulation

a) Lancement du simulateur

Le lancement du simulateur *NC VHDL* passe par la création d'un fichier de liaison, fichier "snapshot", qui s'insère dans la librairie. Pour générer ce fichier il faut double-cliquer  sur le nom du modèle dans le dossier **Work** de la zone 2, puis sélectionner alors  l'architecture que l'on veut simuler et cliquer sur le bouton de commande **Launch Elaborator**  (voir Figure 4). Il reste alors à sélectionner  le fichier **snapshot** correspondant dans le dossier **Snapshots** de la zone 2 (voir Figure 5) et cliquer sur le bouton de commande **Launch Simulator** .

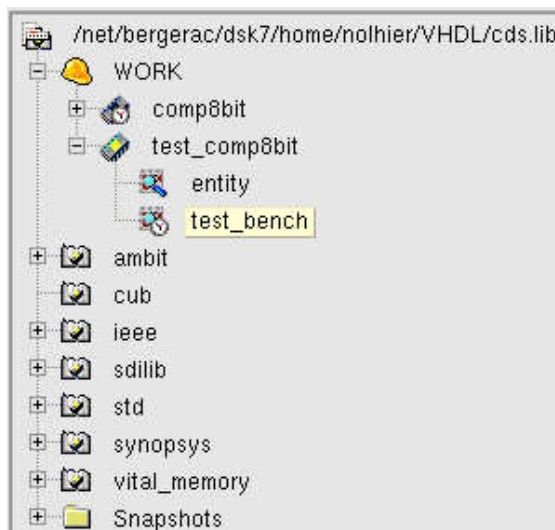


Figure 4 : Sélection de l'architecture

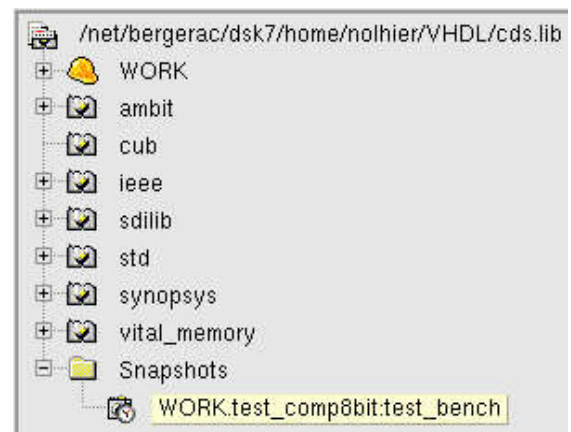


Figure 5 : Sélection du snapshot

b) l'environnement du simulateur

A la commande **Launch Simulator** lancée depuis *NCLaunch*, deux nouvelles fenêtres apparaissent avec l'environnement de contrôle *SIMVISION* de *NCSIM* (Figure 6). La première fenêtre **Design Browser** permet de voir les signaux ou variables de votre modèle. La seconde fenêtre **Console** donne l'état du simulateur.

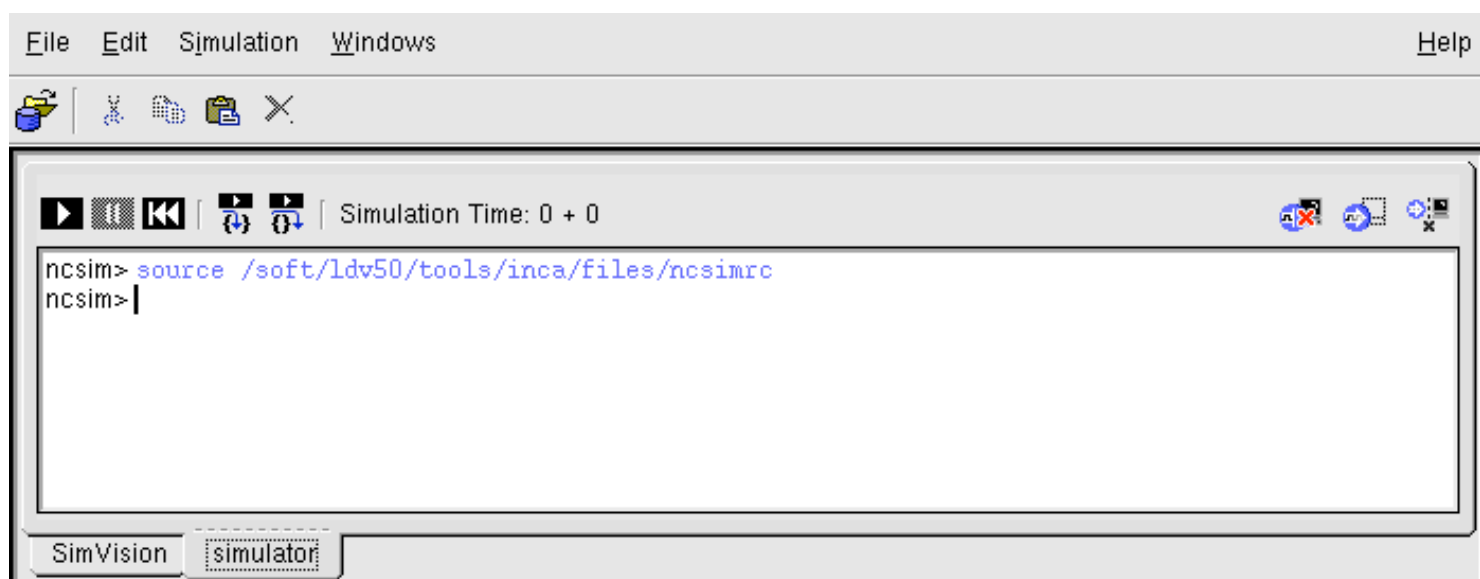
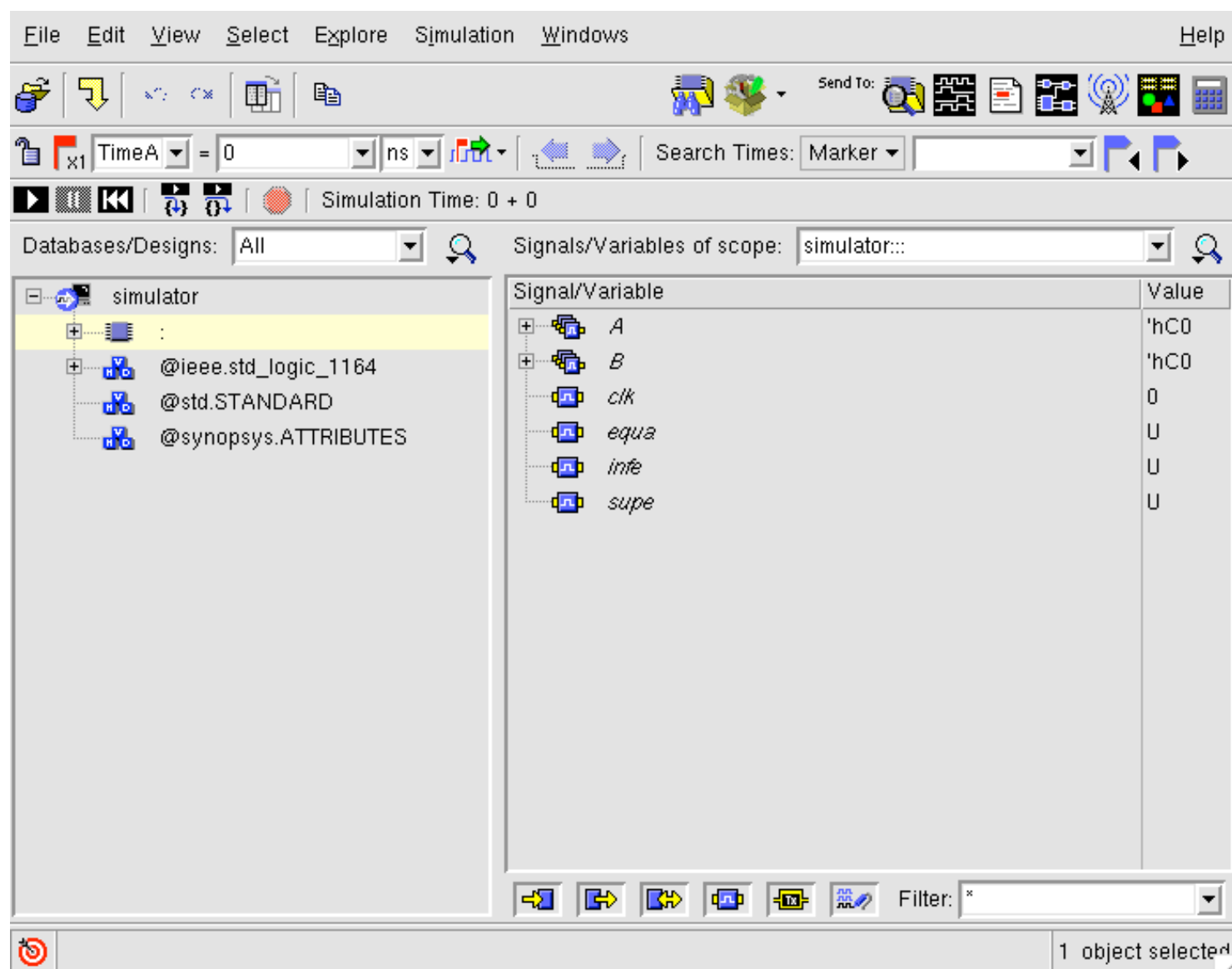


Figure 6 : Environnement de *NCSIM*

La description VHDL de votre modèle peut aussi apparaître dans une troisième fenêtre. Il suffit de sélectionner dans la fenêtre **Design Browser** le menu **Windows** ➔ **New** ➔ **Source Browse**.

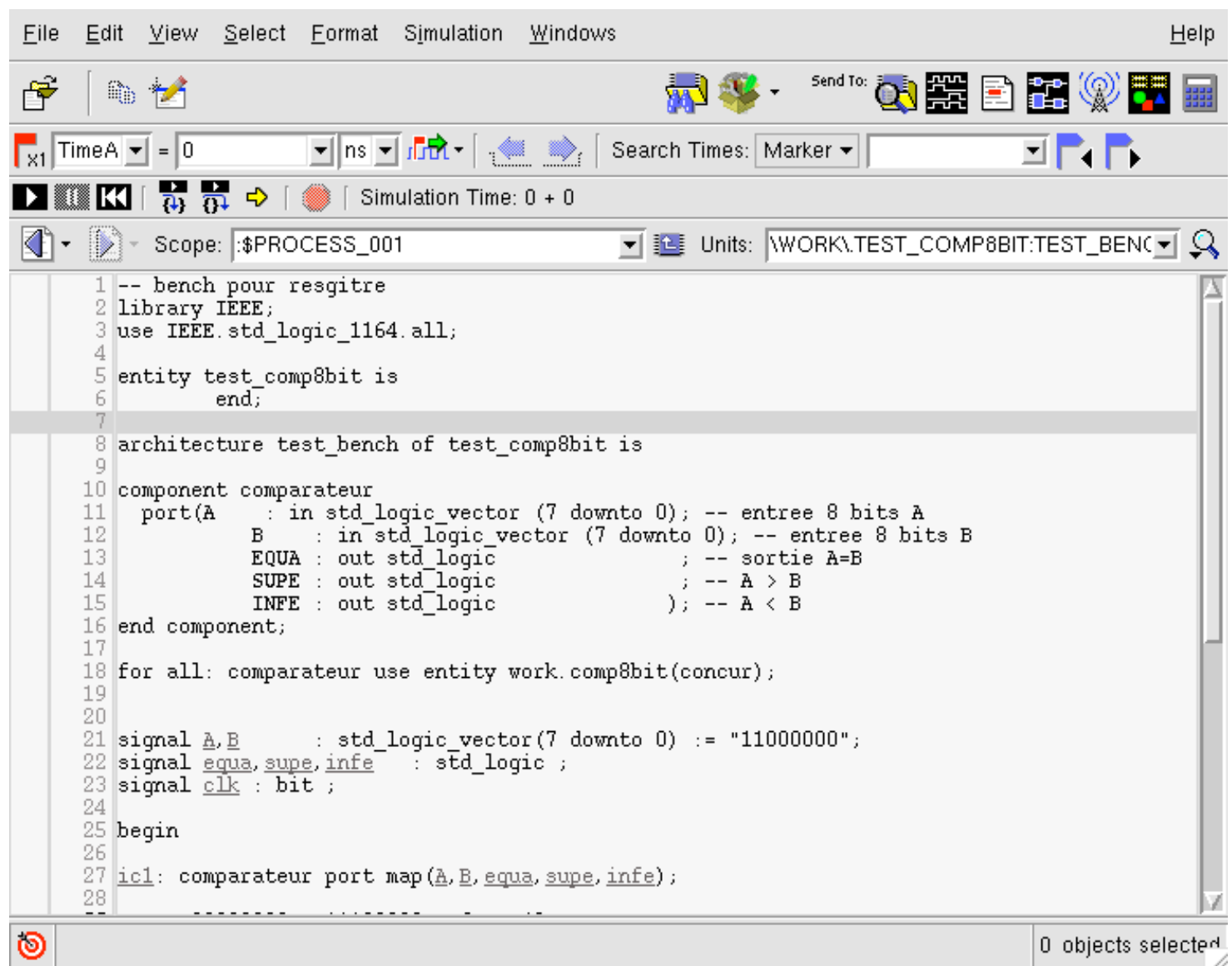




Figure 7 Visualisation du source VHDL dans le simulateur



NCSIM est un simulateur très complet qui permet d'effectuer la simulation pas à pas au niveau source, mettre des points d'arrêt, forcer la valeur des signaux

Plusieurs modes de simulation sont possibles. Nous allons décrire les plus utilisés :

➡ simulation pendant une durée prédéfini :

- Sélection dans le menu **Simulation** ➡ **Set Breakpoint** ➡ **Time**
- Entrer alors la durée (ex : 300 ns); cette valeur peut être relative
- Puis  cliquer sur le bouton **Run Simulation** 
- ✓ Ne pas oublier de paramétrer l'affichage des résultats avant !! (voir chap.5)

➡ simulation pas à pas

- En utilisant la commande **Simulation** ➡ **Step** ou  cliquer sur le bouton 
- Décrit chaque événement de la simulation en suivant la description du source

➡ simulation jusqu'à un point d'arrêt conditionnel

Ce peut être par exemple la valeur d'un signal qui termine la simulation

- Pour ce faire choisissez **Simulation** ➡ **Set Breakpoint** ➡ **Condition**

La figure 8 montre le choix d'un arrêt conditionnel quand le signal **A** est à 000000. Pour insérer un signal utiliser **Browser** puis le bouton **Add**

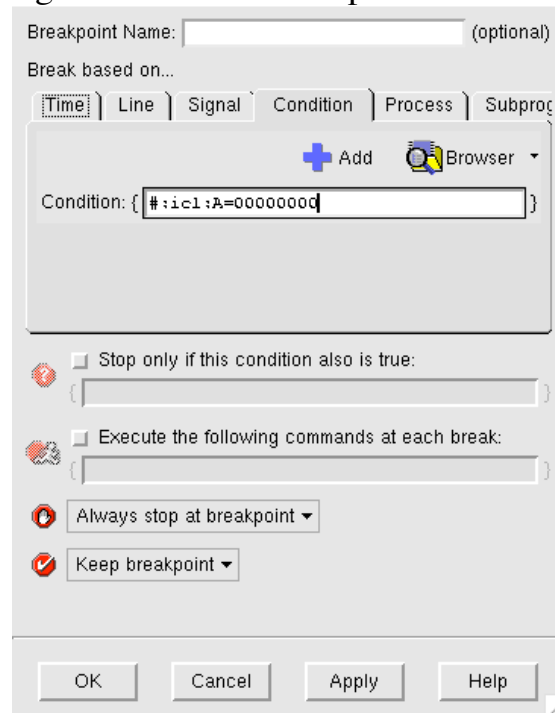


Figure 8 : Choix d'un point d'arrêt conditionnel

☞ La commande **Simulation** ➡ **Show** ➡ **Breakpoints** vous montre tous les points d'arrêt (temporels ou conditionnels) que vous avez choisis.



Vous pouvez les modifier, les rendre actifs ou inactifs ou les effacer.

☞ Les simulations s'effectuent suivant une horloge (Time) qui donne la référence courante du temps. Pour remettre cette horloge à zéro, si vous voulez relancer une simulation depuis le début, vous devez utiliser la commande dans le menu **Simulation** ➡ **Reset to Start**.

Dans ce cas vous gardez les points d'arrêt que vous avez placés préalablement.

5) Exploitation des résultats de simulation

Vous pouvez examiner l'état de vos signaux en cours ou en fin de simulation

- Avant de lancer la simulation (Run), sélectionner ☞ dans le menu **Select** ➡ **Signals**, puis sur le bouton de commande .
- La fenêtre de **WaveForm** apparaît ainsi que vos signaux (sans valeur).
- Lancer la simulation ☞ avec **Simulation** ➡ **Run**.
- A la fin de la simulation, cliquer sur le bouton  dans **Waveform** afin de voir vos valeurs

sur l'intégralité du temps de simulation (Figure 9).

Vous pouvez effectuer des zooms entre deux curseurs:

- placer le curseur **Baseline** , pour la référence gauche
- placer un second curseur **TimeA** où vous voulez comme référence droite
- cliquer alors sur le bouton droit de la souris sur Zoom C-B.

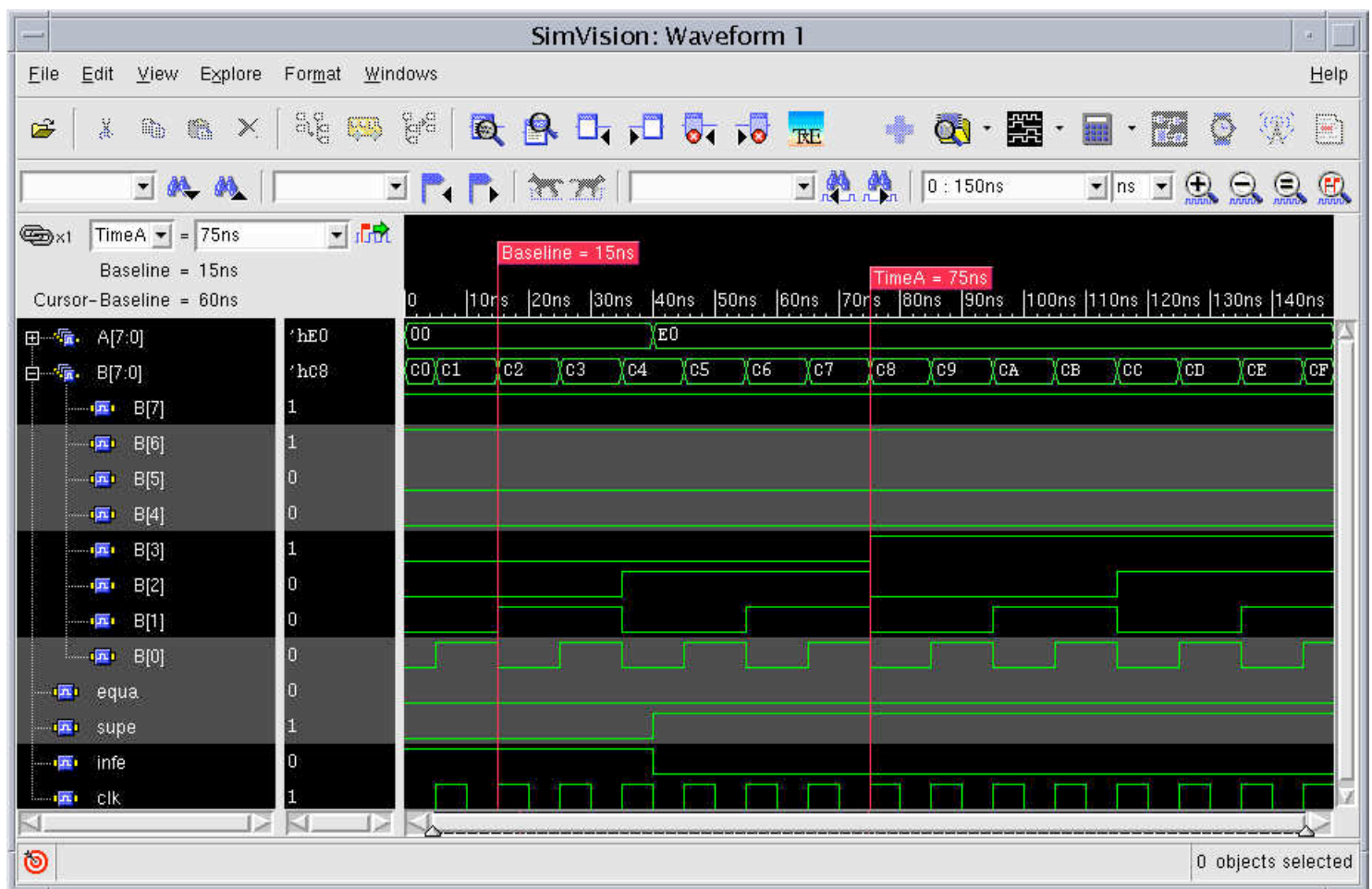


Figure 9 : exemple de résultat graphique avec WaveForm