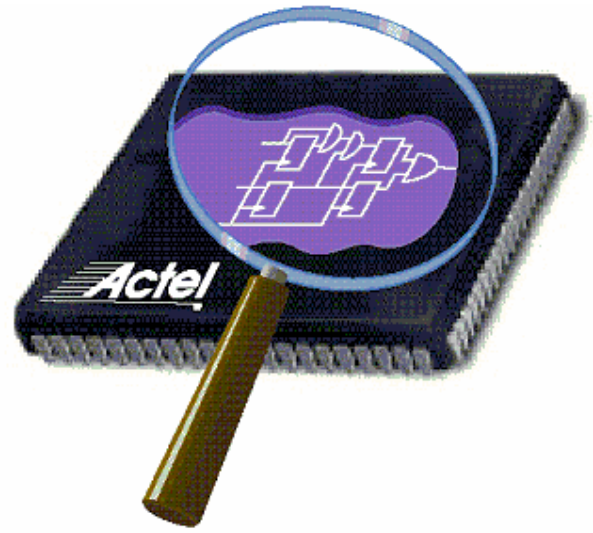
Didacticiel d'initiation à l'environnement de conception de FPGA



ELE3301 et ELE4301

ÉCOLE
POLYTECHNIQUE
MONTRÉAL

Stéphane Boyer
Tommy Désilets

Version 5.0 (Hiver 2005)

TABLE DES MATIÈRES

Introduction

Le troisième projet des cours systèmes logiques II (ELE4301) et systèmes numériques programmables (ELE3301) consiste en partie à réaliser un système commandé par des machines à états qui sera intégré dans un FPGA. Ce didacticiel a comme principal but de vous permettre de vous familiariser avec les différents outils utilisés sur le réseau de systèmes logiques (SYSLOG). L'emphase est mise sur la démarche à suivre et non sur la complexité des circuits qui seront conçus. Tout d'abord vous allez décrire et simuler une machine séquentielle algorithmique (MSA) simple. Par la suite, vous ajouterez un compteur BCD (*binary coded decimal*) et un convertisseur vers un affichage 7 segments. La méthodologie proposée utilise uniquement le langage de description matériel VHDL pour la saisie du design.

Logiciels utilisés dans ce didacticiel

- ModelSim SE 5.8c
- Synplify 7.6
- Actel Designer v6.0

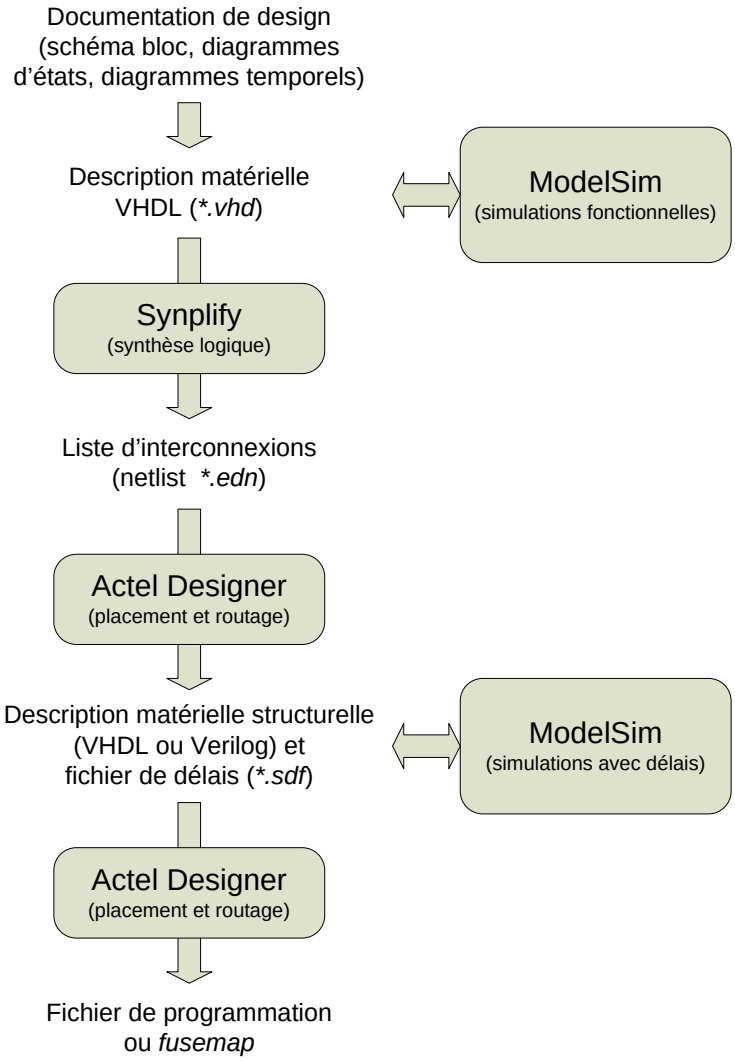
Notes

- Il est recommandé d'utiliser en tout temps des noms de fichiers ne dépassant pas 8 caractères (en plus des 3 caractères habituels d'extension). Plus particulièrement, il vous sera impossible de faire la génération du fichier de programmation si votre niveau hiérarchique supérieur est sauvegardé avec un nom de fichier dépassant 8 caractères.

- Pour la réalisation d'un projet différent de celui proposé, vous devez avoir compléter le design de votre système avant de poursuivre avec la rédaction de la description matériel en VHDL, la vérification, la synthèse et la génération du fichier de programmation. Vous devez avoir en main le schéma bloc détaillé avec tous les noms des signaux, la description de chacun des blocs (par exemple, le diagramme d'états pour une MSA) et les diagrammes temporels pour chacun des blocs de votre système et pour l'ensemble du système.

Description de la méthodologie

Les outils utilisés dans cette méthodologie de design sont parmi les plus répandus dans l'industrie. La figure ci-dessous illustre la méthodologie utilisée dans ce didacticiel. L'outil de simulation *ModelSim* permet de vérifier une description matérielle VHDL avant et après la synthèse. Le synthétiseur logique *Synplify* de Synplicity génère la liste des interconnexions ou « netlist » à partir de la description matérielle. Enfin, l'outil de placement et de routage *Actel Designer* génère le fichier de programmation ainsi que les fichiers nécessaires aux simulations avec délais.

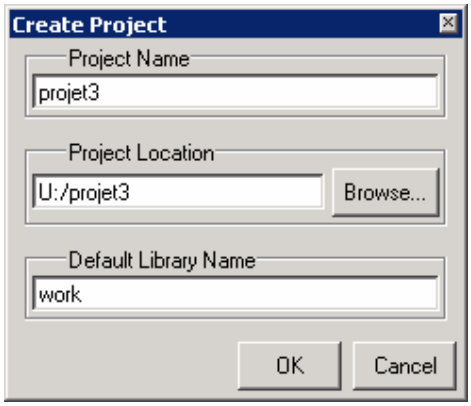


Création d'un nouveau projet dans ModelSim

- Démarrez *ModelSim* via le menu *Start* de Windows.

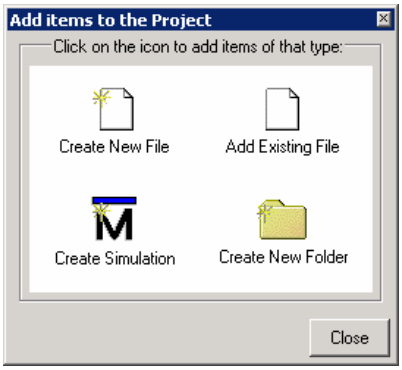
- Créez un nouveau projet : File → New → Project. La création d'un projet génère un fichier **.mpf* dans le répertoire de projet. Vous pouvez utiliser ce fichier ultérieurement pour ouvrir un projet existant dans ModelSim.

- Dans la fenêtre « Create Project », complétez les champs pour le nom du projet et le répertoire par défaut, puis cliquez sur *OK*.



- Si le répertoire n'existe pas déjà, on vous demandera de confirmer sa création.

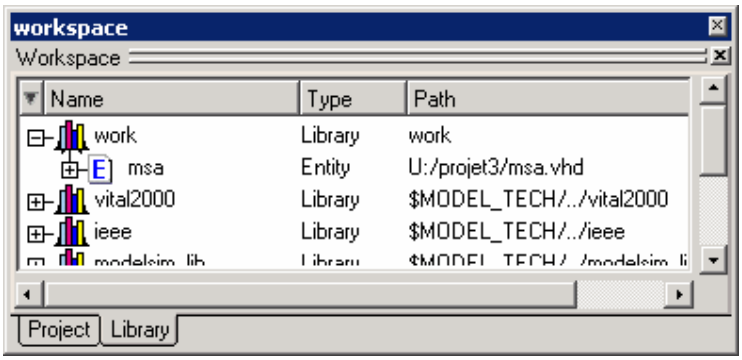
- Ajoutez votre fichier VHDL (*msa.vhd* disponible à l'annexe) au projet courant en utilisant la fenêtre « Add items to the Project » ou par le menu de ModelSim : File → Add to Project → Existing File.



- Compilez le fichier VHDL en accédant au menu : Compile → Compile All.

La fenêtre principale de ModelSim donne l'information suivante :
Compile of msa.vhd was successful.

Vous pouvez également vérifier dans l'espace de travail « Workspace » que la librarie *work* contient l'entité **msa**.



Simulation fonctionnelle avec ModelSim

La présente section montre comment utiliser ModelSim pour simuler un module à partir de la ligne de commande ou d'un fichier de commandes (*.*do*). Il est également possible de réaliser un banc d'essais comme niveau hiérarchique supérieur en VHDL et ainsi d'assigner une valeur donnée aux signaux d'entrée.

- Préparez l'environnement de simulation en accédant au menu ModelSim : Simulate → Simulate… Sous l'onglet design, développez la librairie *work*, choisissez ***msa*** et cliquez sur OK.

Dans l'espace de travail « workspace », l'onglet « sim » est maintenant disponible.

- Pour simuler le module ***msa***, il est nécessaire de forcer une valeur sur les entrées de ce module. Entrez les commandes suivantes dans la fenêtre principale de ModelSim :

```
VSIM 1>  restart –f
VSIM 2>  log –r *
VSIM 3>  add wave *
VSIM 4>  force rstN 0, 1 10 us
VSIM 5>  force clk 0, 1 50 us -repeat 100 us
VSIM 6>  force bt_up 0, 1 100 us, 0 200 us, 1 400 us, 0 600 us
VSIM 7>  force bt_down 0, 1 700 us, 0 800 us
VSIM 8>  run 1000 us
```

- Le détail des commandes peut être connu en tapant la commande **`help`**. Par exemple, pour connaître les détails de la commande **`force`** :

```
VSIM 9>  help force
```

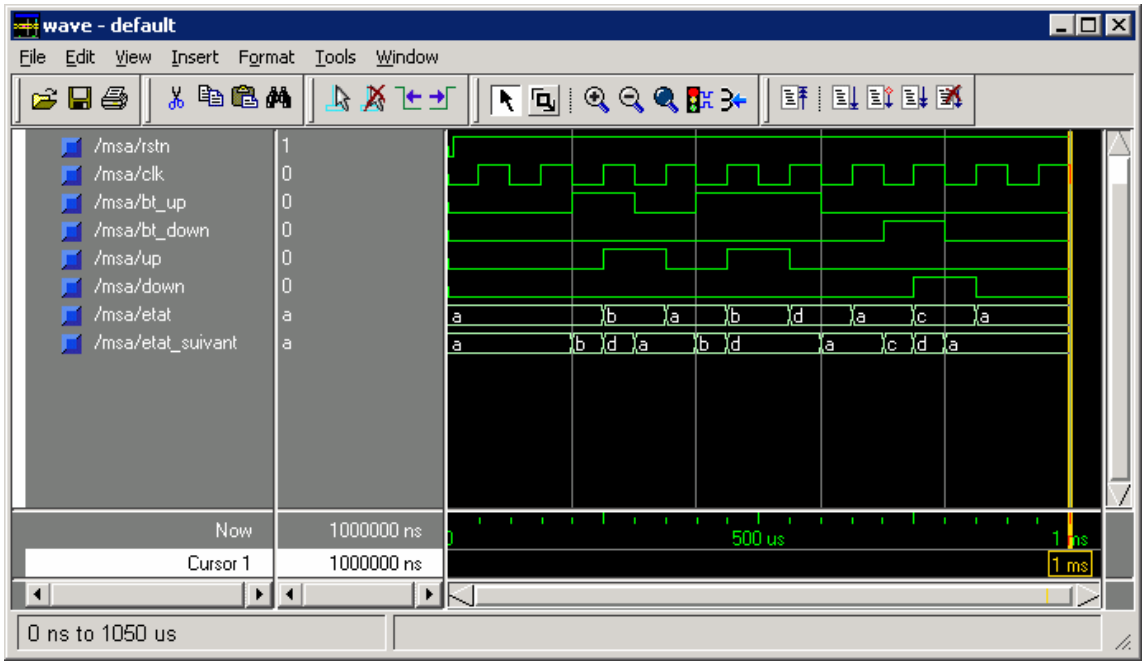
  Un fichier de référence (*modelsim_qk_guide.pdf*) est également disponible sur le disque réseau.

- Enfin, un fichier (*msa.do*) contenant les commandes peut être créé et exécuter avec la commande :

```
VSIM 10>  do msa.do
```

Notez que l'instruction « add wave * » ne devrait pas être comprise dans le fichier de commande puisqu'elle ajouterait l'ensemble des signaux du module choisi à chaque fois que le fichier serait exécuté.

Ci desssous, la fenêtre *wave* montre le fonctionnement adéquat de la MSA.

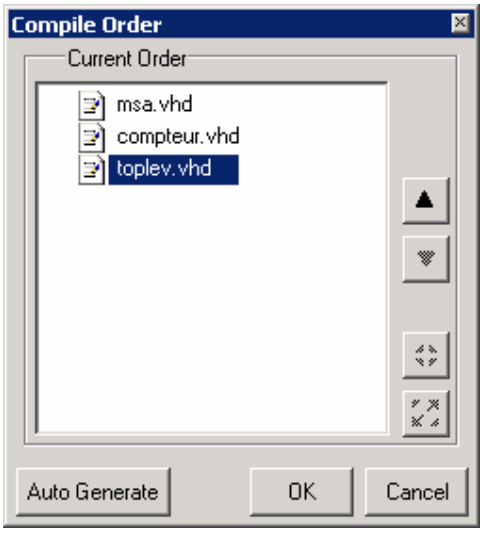


Ajout de fichiers au projet dans ModelSim

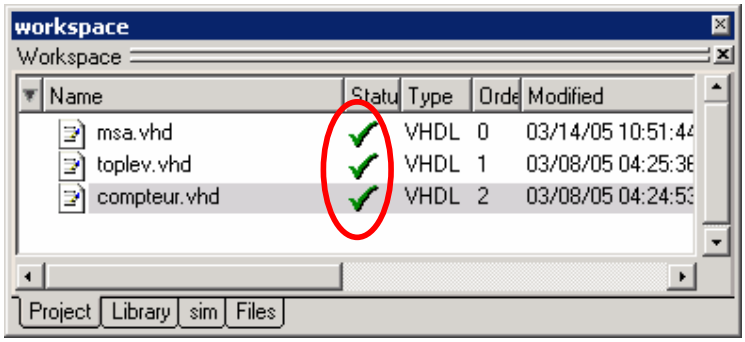
Un projet peut contenir plusieurs fichiers VHDL organisés hiérarchiquement. Cette section montre comment ajouter des fichiers VHDL au projet, le compiler dans l'ordre adéquat et amorcer une simulation du niveau hiérarchique supérieur à l'aide d'un fichier de commande.

- Ajoutez vos fichiers VHDL (*compteur.vhd* et *toplev.vhd* disponibles à l'annexe) au projet courant en accédant au menu de ModelSim : File → Add to Project → Existing File.

- Assurez-vous que l'ordre de compilation est adéquat, i.e. les fichiers de hiérarchie supérieure doivent être compilé en dernier : Compile → Compile Order :



- Compilez l'ensemble des fichiers VHDL et vérifier que la compilation se termine normalement :



- Préparez l'environnement de simulation en accédant au menu ModelSim : Simulate → Simulate… Sous l'onglet design, développez la librairie *work*, choisissez ***toplev*** et cliquez sur OK.

- Vous pouvez utiliser le fichier de commandes défini précédemment en tapant :

```
VSIM 11>  do msa.do
```

Ajoutez le signal *bcd*, interne au module XCOMPTEUR de la façon suivante :

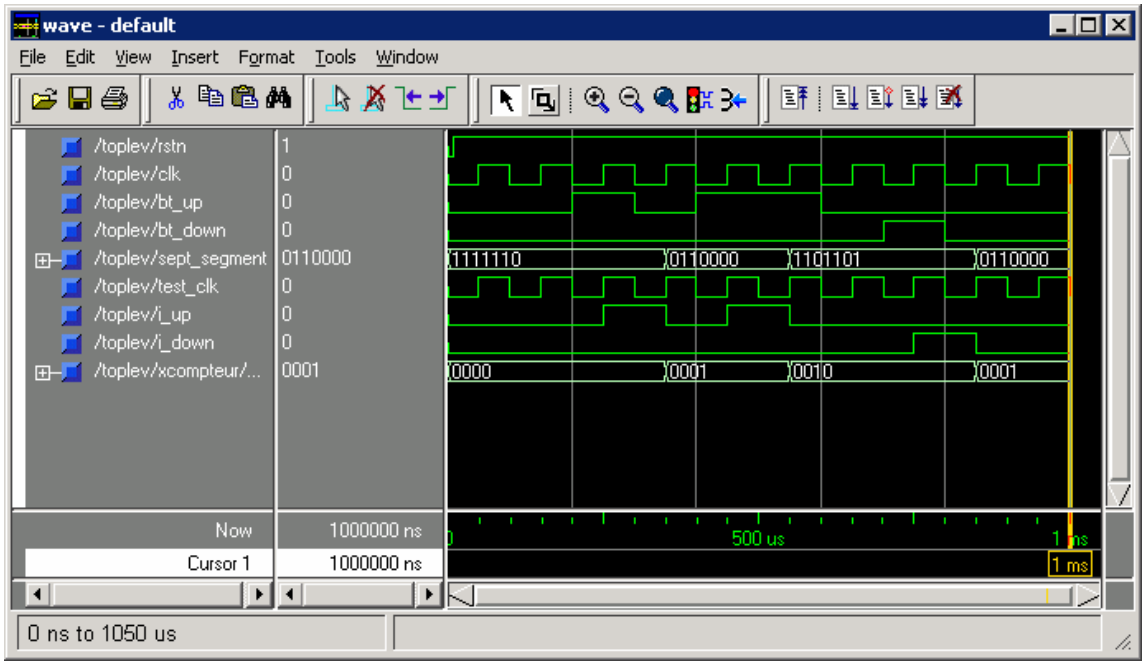
- Visualiser la structure hiérarchique des éléments en cours de simulation en accédant au menu ModelSim : View → Structure ou en accédant à l'espace de travail « workspace » et à l'onglet ***sim***.
- Sélectionner l'instance **xcompteur** après avoir développé le module toplev.
- Visualiser les signaux :

```
VSIM 12>  view signals
```

- Ajoutez le signal interne bcd en sélectionnant et glissant (click and drag) l'élément bcd de la fenêtre « signals » vers la fenêtre « wave » ou en tapant :

```
VSIM 13>  add wave bcd
```

Le résultat de simulation devrait être le suivant:



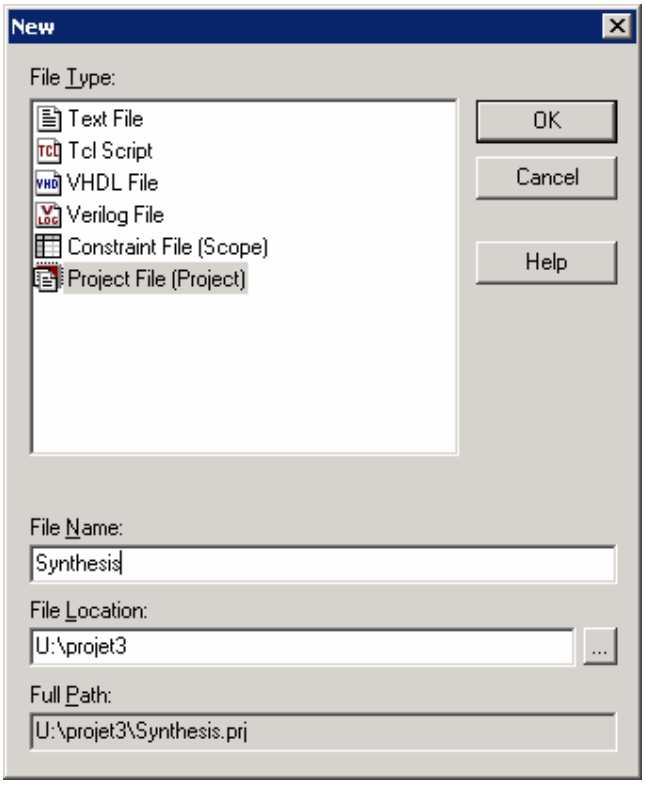
Synthèse de fichiers VHDL

Vous êtes maintenant rendu à l'étape de la synthèse des fichiers VHDL par Synplify 7.6 pour générer une liste des interconnexions (netlist).

- Démarrez *Synplify 7.6* à partir du menu *Start* de *Windows*. Notez que *Synplify* se trouve dans le groupe *Synplicity*.

- Créez un nouveau projet à partir du menu : File → New…

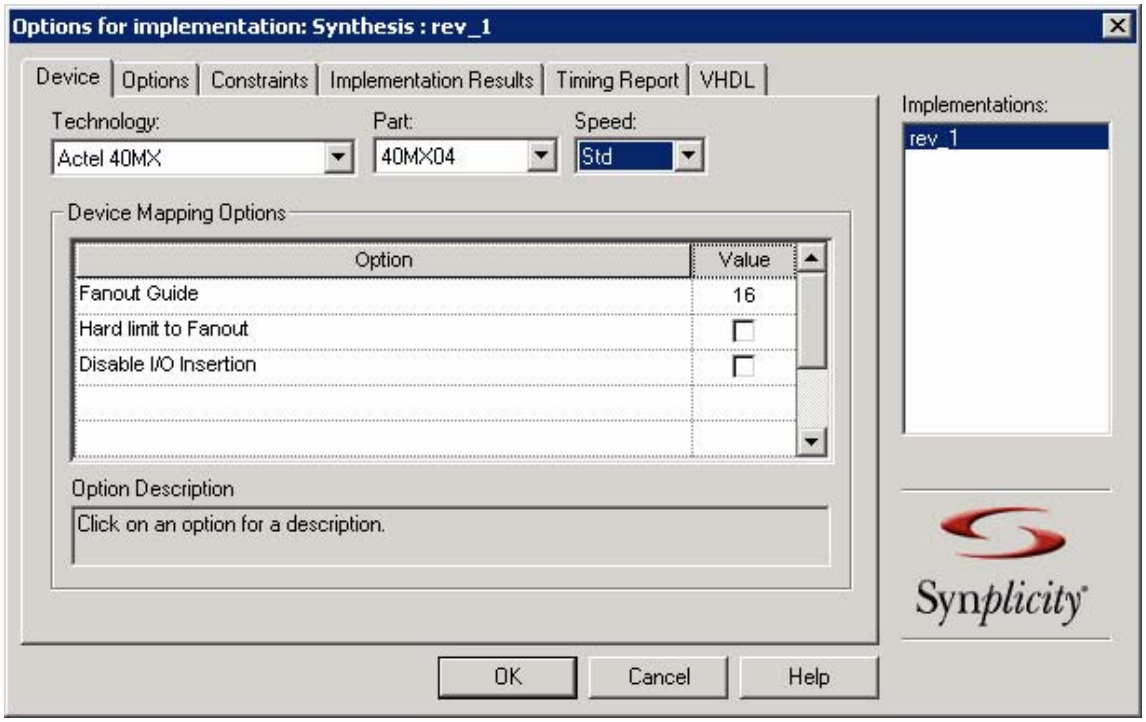
- Choisissez « Project File », entrez un nom de fichier et assurez-vous que vous travaillez bien dans votre répertoire (File Location). Faites *OK*.



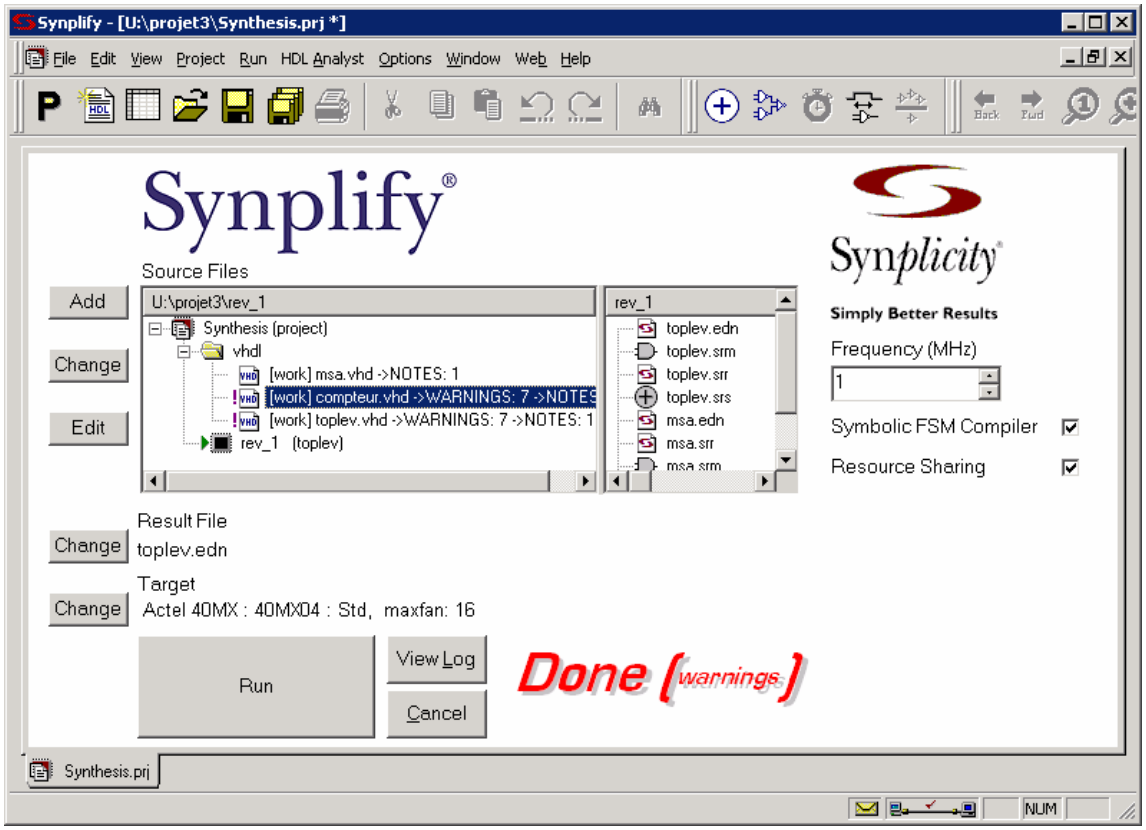
- Cliquez sur le bouton *Add* qui est situé près de la fenêtre « Source Files ». Sélectionnez les fichiers **compteur.vhd, msa.vhd** et **toplev.vhd** puis faites *OK*. Les fichiers peuvent être choisis en gardant la touche *CTRL* enfoncée.

- Ajustez l'ordre de synthèse dans la fenêtre « Source Files » en développant la section **vhdl** et en vous assurant que le fichier **toplev.vhd** est au bas de la liste.

- Vous devez maintenant spécifier la famille cible pour la synthèse. Cliquez sur le bouton *Change* qui est situé près de la ligne **Target**. Entrez les renseignements tels que décrits par la figure ci-dessous et faites *OK*.



- Appuyer sur *RUN* pour lancer la synthèse. Synplify doit afficher ***Done [warnings]*** pour indiquer que la synthèse est complétée.



- Vous pouvez visualiser les erreurs et les avertissements à partir de la zone *Source Files* en double-cliquant sur les fichiers VHDL. Pour passer au prochain message, il suffit de faire F5. Pour ce qui a trait aux messages concernant les ports ***sept_segment***, vous pouvez les ignorer et poursuivre le didacticiel. Ces avertissement (@W) sont dus aux ports de sortie de type **buffer** des modules *compteur* et *toplev*. Les ports ***sept_segment*** ne sont pas utilisés en lecture et pourraient être remplacés par des ports de type **out**.

- Vous pouvez aussi consulter les résultats de synthèse en appuyant sur le bouton *View Log*. La section « Resource Usage Report permet d'avoir une idée qu'en à la taille du circuit généré. En effet, on peut ainsi connaître l'espace utilisé en terme de composants logiques dans le FPGA. Dans notre exemple, le module *toplev* utilise uniquement 97 des 547 (18 %) cellules que peut contenir le FPGA 40MX04 de Actel. Notez que le nombre de registres (cellules séquentielles) est également donné et devrait correspondre au nombre de registres prévus lors du design.

- Vous pouvez aussi visualiser le circuit généré grâce aux boutons  et  qui sont situés sur la barre d'outils de *Synplify*. Dans le cas de la « Technology View », vous pouvez voir les ports d'entrée (inbuf) et de sortie (outbuf) ainsi que le port d'horloge (clkbuf) et le signal interne *test_clk_c* redirigé vers un port de sortie.

Génération du fichier de programmation

La génération du fichier de programmation nécessite la conversion de la liste d'interconnexions (netlist).

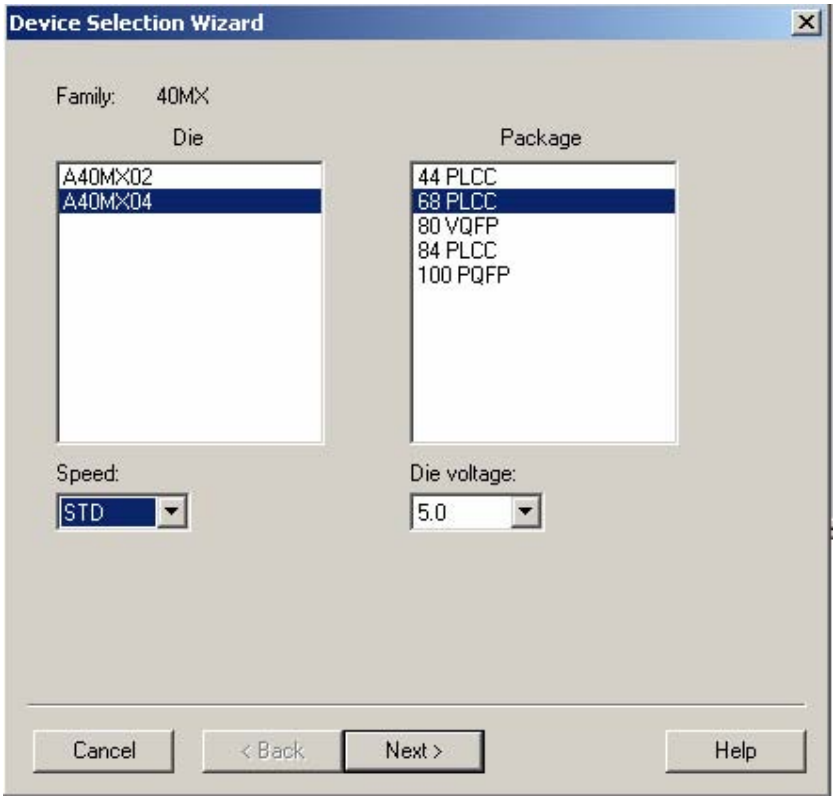
- Démarrez *Designer v6.0* à partir du menu *Start* de *Windows*. Notez que *Designer* se trouve dans le groupe *Actel Designer v6.0*.

- Cliquez sur *Start New Design*. Inscrivez "toplev" comme nom de votre design. Choisissez la famille **40MX** et votre répertoire primaire.

- Cliquez sur *Compile* et faites *Add* pour choisir votre fichier **toplev.edn** dans le répertoire **rev_1/**. Faites *Import* et *OK*.

- Une fenêtre s'ouvre et choisissez comme *EDIF flavor* **GENERIC.**

- Ensuite choisissez les options appropriées et faites *Next* 2 fois puis *Finish*.



- Vous devriez obtenir le rapport suivant:

```
Created a new design.
Imported the file:
   U:\projet3\rev_1\toplev.edn

The Import command succeeded ( 00:00:02 )
Inserted BUFF before test_clk_pad:D, because this macro pin cannot
be directly connected to the routed clock network.
Info: Fanout of 12 exceeds recommended limit of 10 . Net: rstN_c
Info: Fanout of 15 exceeds recommended limit of 10 . Net:
XCOMPTEUR/bcdZ0Z_2
Info: Fanout of 13 exceeds recommended limit of 10 . Net:
XCOMPTEUR/bcdZ0Z_1
Post-Combiner device utilization:
    LOGIC           Used:   75  Total:  547   (13.71%)
    IO              Used:   12  Total:   57
    CLOCK           Used:    1  Total:    1
There were 0 error(s) and 0 warning(s) in this design.

The Compile command succeeded ( 00:00:01 )
```

- Notez que le nombre de modules utilisés est inférieur au rapport généré par *Synplify*. Ceci est dû aux outils d'optimisation d'Actel qui réussissent à combiner les fonctions et ainsi utiliser un minimum de ressources pour un design donné.

Vous êtes maintenant en mesure d'attribuer vous-même les signaux aux broches du FPGA :

- Cliquez sur le bouton **PinEditor** qui est situé dans la zone « User Tools ». Vous avez à présent une fenêtre qui montre la représentation graphique du FPGA ainsi que des signaux qui sont à placer.

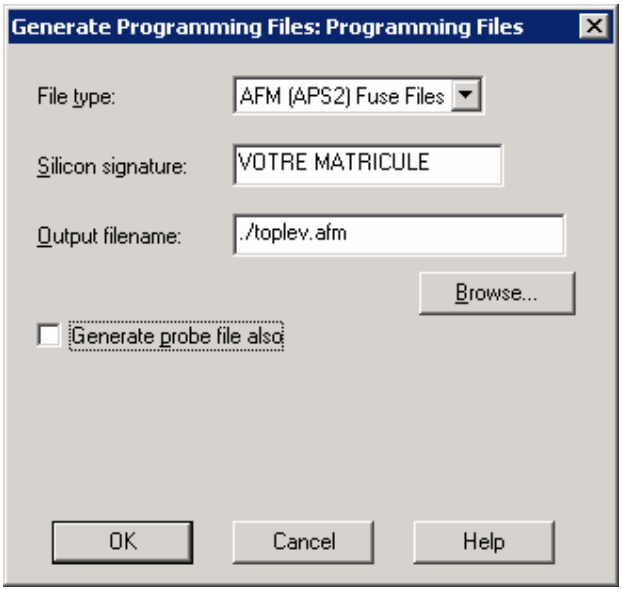
- Il vous faut vous tout simplement utiliser la technique de « click and drag » pour placer les signaux de la colonne **Unassigned**. Le signal CLK doit obligatoirement être placé à la broche 52 puisque c'est la seule qui est dotée d'un *buffer* d'entrée plus puissant (**clkbuf**). Aussi, il est préférable de ne rien assigner aux broches 56 à 59, qui sont réservées à des fonctions particulières. Une fois les signaux placés, vous les verrez apparaître à coté de la broche de FPGA correspondante et être transférés de la colonne **UNPLACED** à la colonne **PLACED**. Vous venez maintenant d'assigner une préférence pour l'assignation des broches.

- L'attribution des signaux aux broches est maintenant terminée. Vous pouvez fermer en faisant File-Commit puis File-Close.

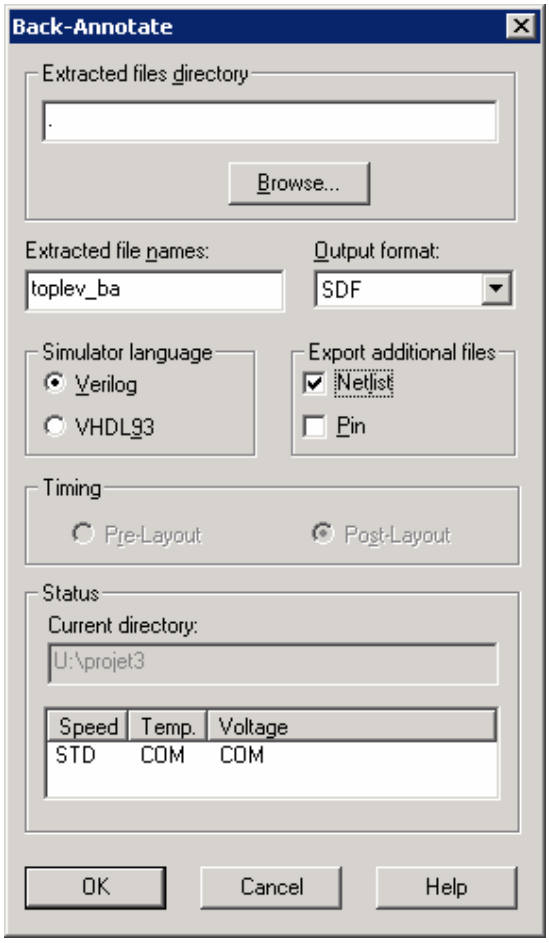
- Sauvegarder votre projet sous le nom de toplev.adb en faisant File-Save dans la fenêtre de Designer.

- Appuyez sur le bouton **Layout** pour effectuer le placement et le routage du FPGA. Vérifiez que l'option *Place incrementally* est bel et bien désactivée et faites *OK*.

- La dernière étape est la génération du *fusemap*, qui est le fichier de programmation. Cliquez sur le bouton **Fuse**. Assurez-vous d'avoir les informations de la figure suivante en inscrivant dans le champ de Silicon signature les cinq derniers chiffres de votre matricule. Faites *OK*.



- Enfin, cliquez sur le bouton **Back-Annotate** pour générer l'information sur les délais et le fichier VHDL décrivant votre design au niveau « portes ». Les options **Verilog** et **Netlist** doivent être choisies.



- Vous devriez obtenir le rapport suivant :

```
Generating SDF timing back-annotation file.
The timing file is `./toplev_ba.sdf'.
Back-annotated to the file(s):
   .\toplev_ba.sdf
   .\toplev_ba.v

The Back-Annotate command succeeded ( 00:00:05 )
```

Les fichiers générés seront utilisés pour les simulations avec délais.

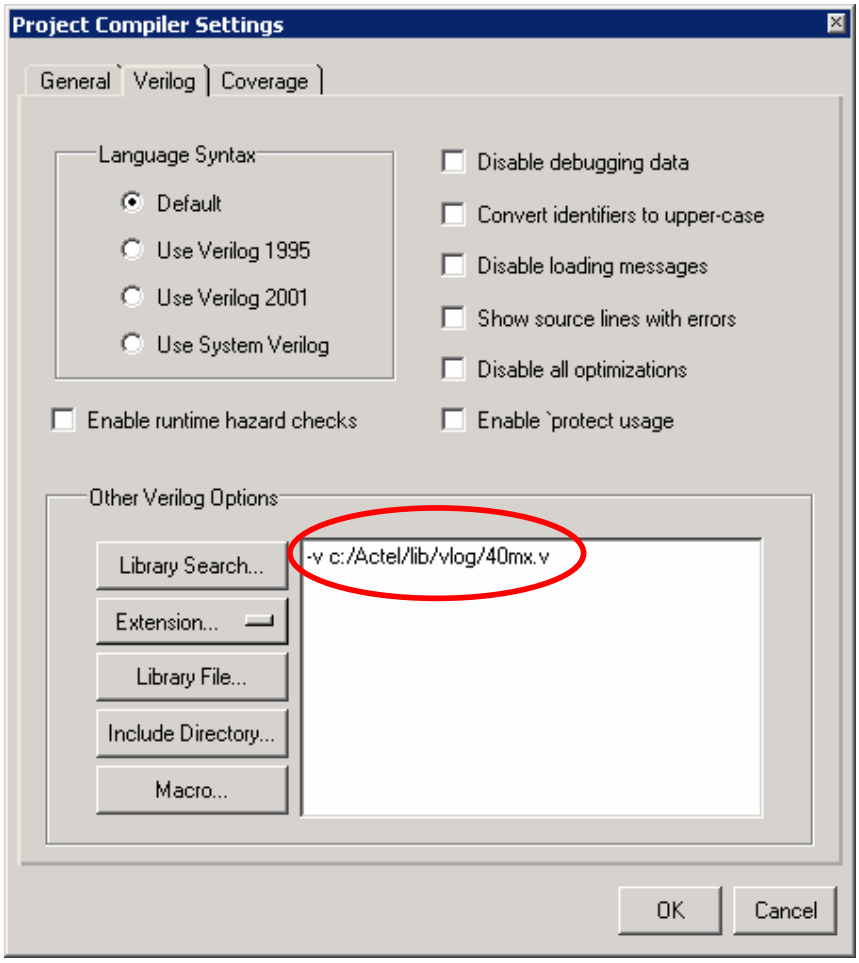
Simulation avec délais

La dernière étape consiste à simuler votre design avec les délais internes du FPGA. Ces délais sont estimés en fonction de la liste d'interconnexions, du placement et du routage des portes et des fils constituant votre design. Cette information est maintenant disponible puisque la façon dont le placement et le routage sont faits vient d'être déterminée lors de la génération du fichier de programmation, spécifiquement sur du *Back-Annotate*.

- Retournez dans ModelSim et ajoutez le fichier **toplev_ba.v** à l'espace de travail « workspace ».
  Le fichier **toplev_ba.v** est une description structurelle au niveau porte qui utilise la librairie *40mx* et ses composants (portes logiques, bascules, multiplexeurs, etc.)

- Sélectionnez le fichier Verilog **toplev_ba.v** et modifiez-en les propriétés : View → Properties. Cliquez sur l'onglet Verilog et ensuite, sur le bouton « Library File… ». Dans la fenêtre « Select Library File » entrez : **c:\Actel\lib\vlog\40mx.v** puis faites *OK*.

- Vous devez obtenir la fenêtre suivante :

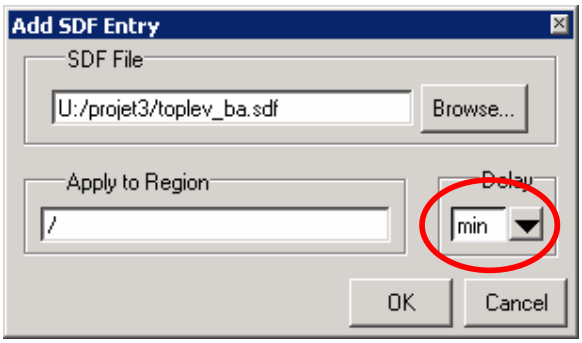


- Compilez le fichier Verilog : Compile → Compile Selected.

Compile of toplev_ba.v was successful.

Préparez l'environnement de simulation avec délais :

- Ouvrez la fenêtre pour amorcer une simulation : Simulate → Simulate…
- Développez la librairie *work* et choisissez le module **toplev** (U:/projet3/toplev_ba.v)
- Passez à l'onglet SDF et ajoutez le fichier de délais en appuyant sur le bouton **Add…**
- Sélectionnez le ficher de délais **toplev_ba.sdf** et les délais minimaux, puis faites *OK*.



- Faites *OK* pour charger l'environnement de simulation.

Loading U:/projet3/toplev_ba.sdf
** Note: (vsim-3587) SDF Backannotation Successfully Completed.

Le module toplev est maintenant prêt à être simulé.

- Ajoutez les signaux d'intérêt pour la simulation

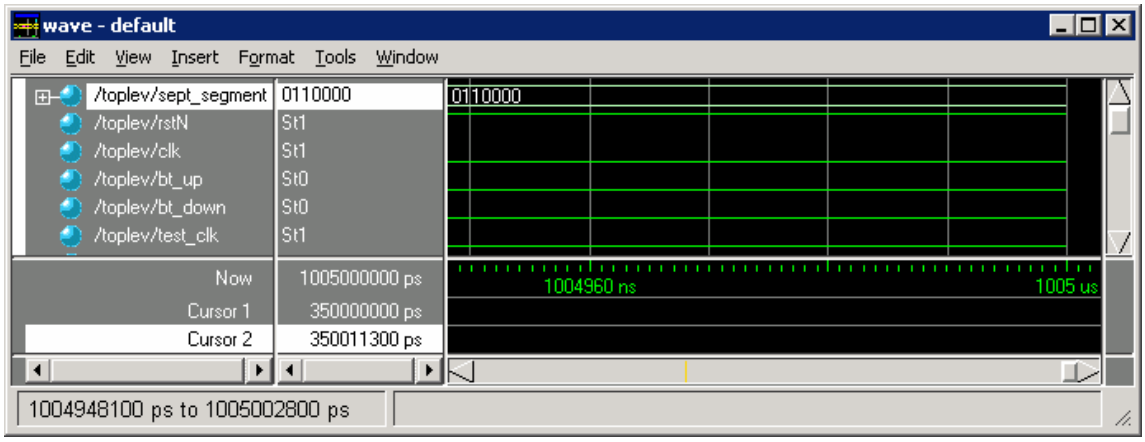
```
VSIM 1>  add wave *
```

Notez que les signaux ajoutés diffèrent de la simulation fonctionnelle et que pour une simulation sur un fichier Verilog, les états '0' et '1' sont St0 et St1 respectivement.

- Lancez la simulation à l'aide du fichier de commande défini lors de la simulation fonctionnelle :

```
VSIM 2>  do msa.do
```

Vous pouvez voir le délai des composants, de l'ordre de la nanoseconde, en faisant un zoom sur un front montant de l'horloge près d'une transition de *sept_segment*.



Après avoir vérifié que la simulation avec délais donne les mêmes résultats que la simulation fonctionnelle, vous pouvez compléter le projet en faisant programmer votre FPGA. Pour ce faire, contactez un technicien qui se chargera des étapes nécessaires. On vous fournira également un adaptateur approprié pour pour monter votre FPGA sur votre plaquette de montage. Consultez la fiche technique du 40MX04 avant de monter votre circuit. Enfin, l'impression de l'assignation des broches à l'aide de Designer vous sera fort utile.

Annexes

Fichier msa.vhd

```
---------------------------------------
-- Fichier: msa.vhd
---------------------------------------
library IEEE;
   use IEEE.std_logic_1164.all;

entity msa is
port(
   rstN, clk         : in     std_logic;
   bt_up, bt_down    : in     std_logic;
   up, down          : buffer std_logic
);
end entity msa;

architecture didac of msa is
   type tetat is (a, b, c, d);
   signal etat, etat_suivant : tetat;
begin

      XIFL: process(etat, bt_up, bt_down)
      begin
         case etat is
            when a =>
               if (bt_up = '1') then
                  etat_suivant <= b;
               elsif (bt_down = '1') then
                  etat_suivant <= c;
               else
                  etat_suivant <= a;
               end if;
            when b | c | d =>
               if (bt_up = '0' and bt_down = '0') then
                  etat_suivant <= a;
               else
                  etat_suivant <= d;
               end if;
            when others =>
                  etat_suivant <= a;
         end case;
      end process XIFL;

      -- Memoire
      XREG: process(rstN, clk)
      begin
         if (rstN = '0') then
            etat <= a;
         elsif (clk'event and clk = '1') then
            etat <= etat_suivant;
         end if;
      end process XREG;

      -- OFL
      up    <= '1' when etat = b else
               '0';
      down  <= '1' when etat = c else
               '0';
end architecture didac;
```

Fichier msa.do

```
restart -f
force rstN 0, 1 10 us
force clk 0, 1 50 us -repeat 100 us
force bt_up 0, 1 200 us, 0 300 us, 1 400 us, 0 600 us
force bt_down 0, 1 700 us, 0 800 us
run 1000 us
```

Fichier compteur.vhd

```
---------------------------------------
-- Fichier: compteur.vhd
---------------------------------------
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_unsigned.all;

entity compteur is
port(
   rstN, clk         : in     std_logic;
   up, down          : in     std_logic;
   sept_segment      : buffer std_logic_vector(6 downto 0)
);
end entity compteur;

architecture didac of compteur is
   signal bcd, bcd_suivant : std_logic_vector(3 downto 0);
begin
   -- Processus combinatoire
      XBCD: process(bcd, up, down)
      begin
      if (up = '1') then
         if (bcd = 9) then
            bcd_suivant <= X"0";
         else
            bcd_suivant <= bcd + 1;
         end if;
      elsif (down = '1') then
         if (bcd = 0) then
            bcd_suivant <= X"9";
         else
            bcd_suivant <= bcd - 1;
         end if;
      else
         bcd_suivant <= bcd;
      end if;
      end process XBCD;

      -- Registres
      XREG: process(rstN, clk)
      begin
         if (rstN = '0') then
            bcd <= (others => '0');
         elsif (clk'event and clk = '1') then
            bcd <= bcd_suivant;
         end if;
      end process XREG;
```

```
      -- Conversion BCD a 7 segments (combinatoire)
      -- Correspondance: 6 downto 0 => abcdefg
      --   _a_
      --  |   |
      -- f|_g_|b
      --  |   |
      -- e|_d_|c
      --
      sept_segment  <= "1111110"when bcd = X"0" else
                       "0110000"when bcd = X"1" else
                       "1101101"when bcd = X"2" else
                       "1111001"when bcd = X"3" else
                       "0110011"when bcd = X"4" else
                       "1011011"when bcd = X"5" else
                       "1011111"when bcd = X"6" else
                       "1110000"when bcd = X"7" else
                       "1111111"when bcd = X"8" else
                       "1111011"when bcd = X"9" else
                       "1110111"when bcd = X"a" else
                       "0011111"when bcd = X"b" else
                       "1001110"when bcd = X"c" else
                       "0111101"when bcd = X"d" else
                       "1001111"when bcd = X"e" else
                       "1000111"when bcd = X"f" else
                       "-------";

end architecture didac;
```

Fichier toplev.vhd

```
--------------------------------------
-- Fichier: toplev.vhd
--------------------------------------
library IEEE;
   use IEEE.std_logic_1164.all;

entity toplev is
port(
   rstN, clk         : in     std_logic;
   bt_up, bt_down    : in     std_logic;
   sept_segment      : buffer std_logic_vector(6 downto 0);
   test_clk          : buffer std_logic
);
end entity toplev;

architecture struct of toplev is
   component msa is
      port(
            rstN, clk         : in     std_logic;
            bt_up, bt_down    : in     std_logic;
            up, down          : buffer std_logic
      );
      end component msa;

      component compteur is
      port(
            rstN, clk         : in     std_logic;
            up, down          : in     std_logic;
            sept_segment      : buffer std_logic_vector(6 downto 0)
      );
      end component compteur;

   -- signaux internes
   signal i_up, i_down : std_logic;
begin
   XMSA: msa
   port map(
      rstN         => rstN,
      clk          => clk,
      bt_up        => bt_up,
      bt_down      => bt_down,
      up           => i_up,
      down         => i_down
   );

   XCOMPTEUR: compteur
   port map(
      rstN         => rstN,
      clk          => clk,
      up           => i_up,
      down         => i_down,
      sept_segment => sept_segment
   );


   XTEST_CLK: test_clk <= clk;

end architecture struct;
```