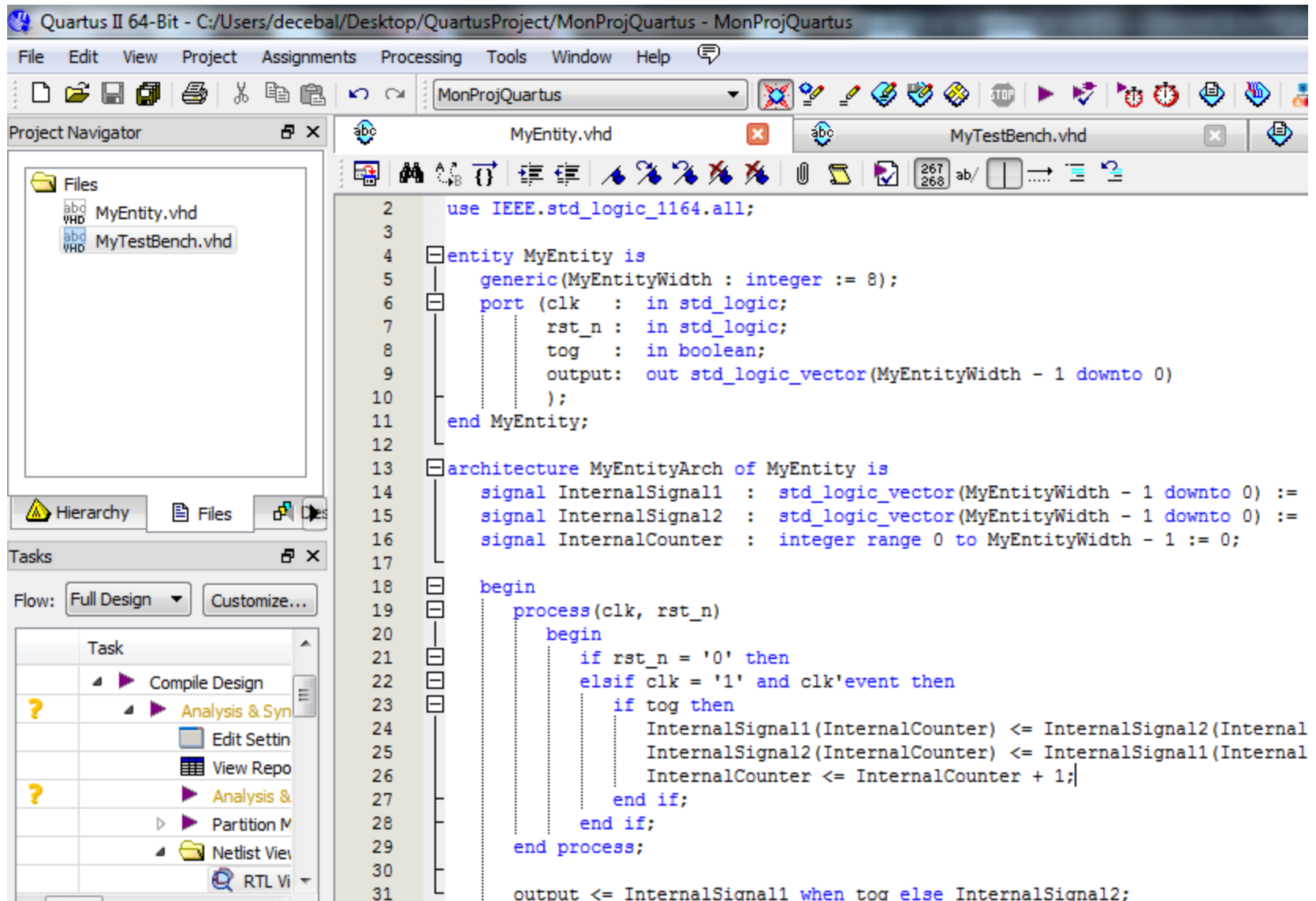


ModelSim et Testbench en VHDL

- Vous avez le code VHDL de votre entité TOP (celle qui renferme toutes les autres entités)



```
Quartus II 64-Bit - C:/Users/decebal/Desktop/QuartusProject/MonProjQuartus - MonProjQuartus
File Edit View Project Assignments Processing Tools Window Help
MonProjQuartus
Project Navigator
Files
  MyEntity.vhd
  MyTestBench.vhd
Hierarchy Files
Tasks
Flow: Full Design Customize...
Task
  Compile Design
  Analysis & Syn
  Edit Settin
  View Repo
  Analysis &
  Partition M
  Netlist View
  RTL Vi

MyEntity.vhd
2 use IEEE.std_logic_1164.all;
3
4 entity MyEntity is
5     generic(MyEntityWidth : integer := 8);
6     port (clk : in std_logic;
7           rst_n : in std_logic;
8           tog : in boolean;
9           output: out std_logic_vector(MyEntityWidth - 1 downto 0)
10          );
11 end MyEntity;
12
13 architecture MyEntityArch of MyEntity is
14     signal InternalSignal1 : std_logic_vector(MyEntityWidth - 1 downto 0) :=
15     signal InternalSignal2 : std_logic_vector(MyEntityWidth - 1 downto 0) :=
16     signal InternalCounter : integer range 0 to MyEntityWidth - 1 := 0;
17
18 begin
19     process(clk, rst_n)
20     begin
21         if rst_n = '0' then
22         elsif clk = '1' and clk'event then
23             if tog then
24                 InternalSignal1(InternalCounter) <= InternalSignal2(Internal
25                 InternalSignal2(InternalCounter) <= InternalSignal1(Internal
26                 InternalCounter <= InternalCounter + 1;
27             end if;
28         end if;
29     end process;
30
31     output <= InternalSignal1 when tog else InternalSignal2;
```

Testbench

- Créez un autre fichier VHDL, le banc d'essais (testbench) qui va vous aider à fournir des entrées à votre circuit pour que vous puissiez observer ce qui se passe à l'intérieur. Ce fichier se trouve dans le projet Quartus.
- L'entité est vide puisqu'il n'y a aucune entrée ni sortie dans un testbench.

```
library IEEE;  
use IEEE.std_logic_1164.all;  
  
entity MyTestBench is end entity;
```

- Dans l'architecture, avant le mot clé '*Begin*' vous déclarez votre composante, l'entité TOP. Copier-coller l'entité.

```
entity MyTestBench is end entity;  
  
architecture sim of MyTestBench is  
  
    component MyEntity is  
        generic(MyEntityWidth : integer := 8);  
        port (clk      : in std_logic;  
              rst_n    : in std_logic;  
              tog       : in boolean;  
              output:   out std_logic_vector(MyEntityWidth - 1 downto 0)  
              );  
    end component;
```

Testbench

- Déclarez les signaux nécessaires pour connecter votre entité avec le banc d'essais. Ici, il faut 4 signaux: une horloge, un signal reset, un signal toggle et un signal output.
- La constante sera utilisé pour le paramètre générique de l'entité.

```
    | | | );  
end component;  
  
constant TB_MY_ENTITY_WITDH : integer := 16;  
  
signal tb_clk      : std_logic := '0';  
signal tb_rst_n    : std_logic := '0';  
signal tb_tog      : boolean   := false;  
signal tb_output    : std_logic_vector(TB_MY_ENTITY_WITDH - 1 downto 0);
```

- Il est souvent utile d'utiliser une machine à états finis pour que vous sachiez quelles entrées vous donnez à votre circuit et à quel moment.
- Une FSM est déclarée à l'aide d'une énumération, avec le mot clé '*Type*' suivi du nom, du mot clé '*is*' et, entre parenthèses, des différents états dont vous avez besoin.

```
signal tb_output    : std_logic_vector(TB_MY_ENTITY_WITDH - 1 downto 0);  
  
type SimStateType is (State0, State1, FinalState);  
signal SimState     : SimStateType := State0;
```

Testbench

- Le reste de l'architecture, avec l'utilisation du '*case-is-when*' pour indiquer le changement d'états.

```

process(tb_clk, tb_rst_n)
begin
    if tb_rst_n = '0' then
    elsif tb_clk = '1' and tb_clk'event then
        case SimState is
            when State0 =>
                tb_tog <= true;
                SimState <= State1;

            when State1 =>
                tb_tog <= false;
                SimState <= FinalState;

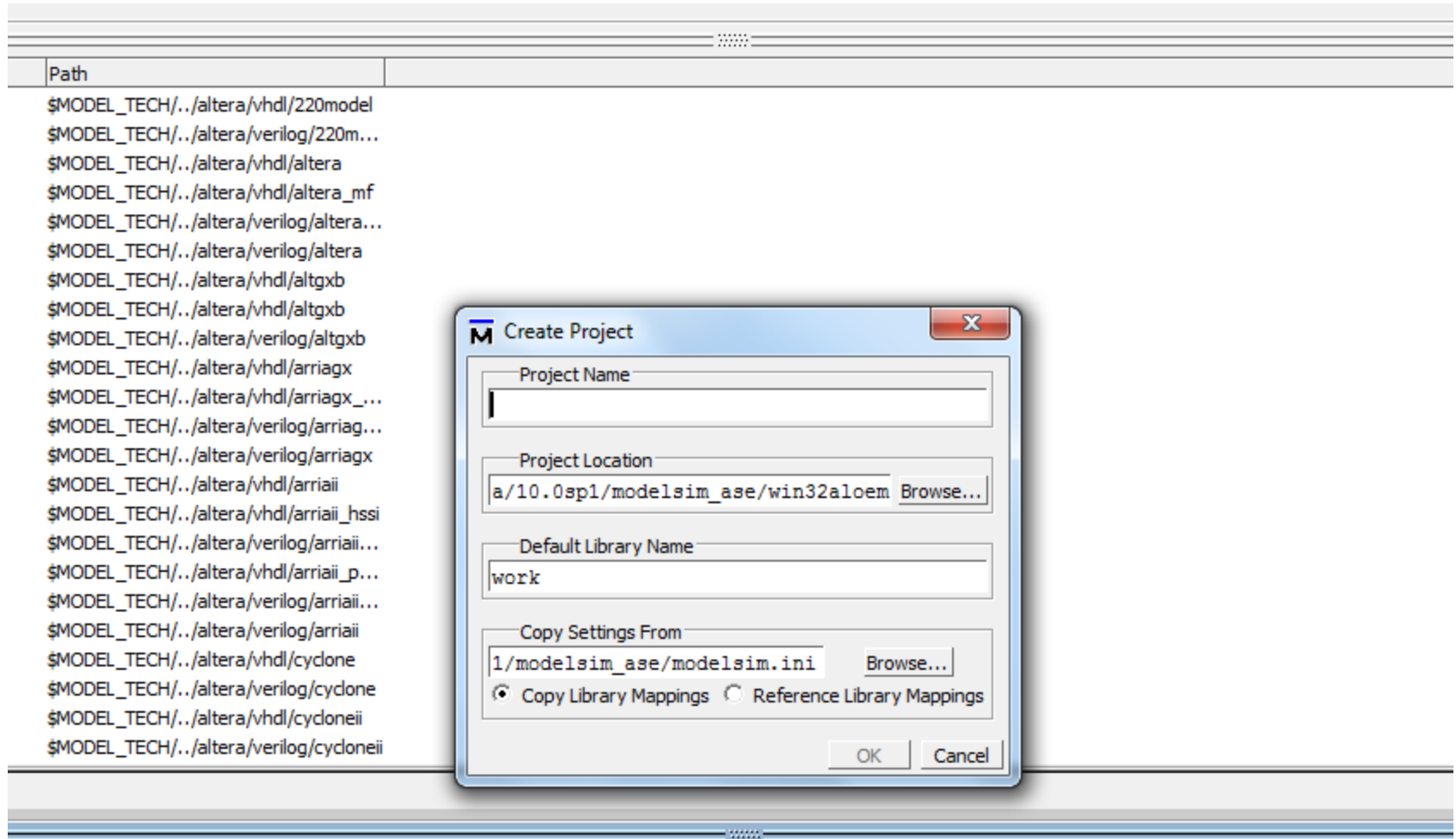
            when FinalState =>
                -- Reste ici pour toujours

            when others =>
                -- Important pour la simulation meme si on a specifie tous les cas
        end case;
    end if;
end process;
end architecture;

```

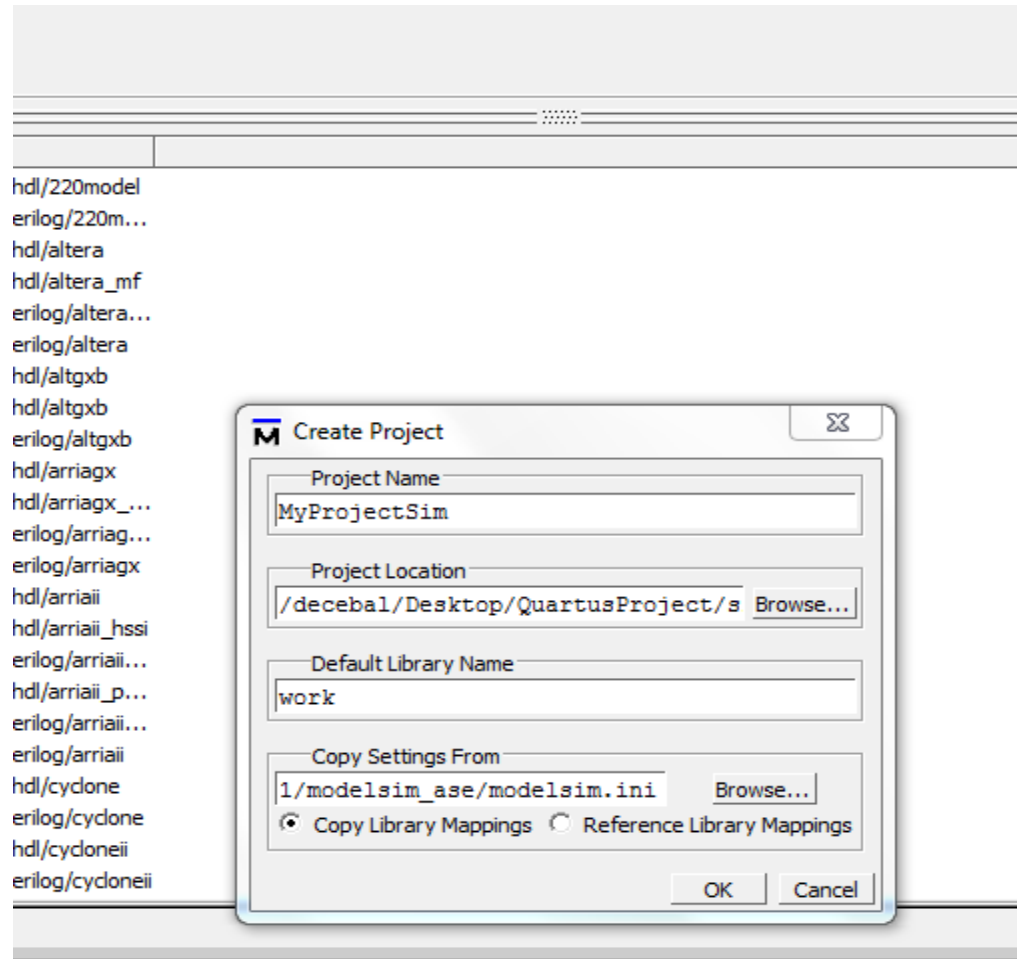
Testbench

- Ouvrez ModelSim. Dans le répertoire de votre projet Quartus, créez un autre répertoire et appelez-le sim (ou n'importe quel autre nom).
- Dans ModelSim, faites *File -> New -> Project...*



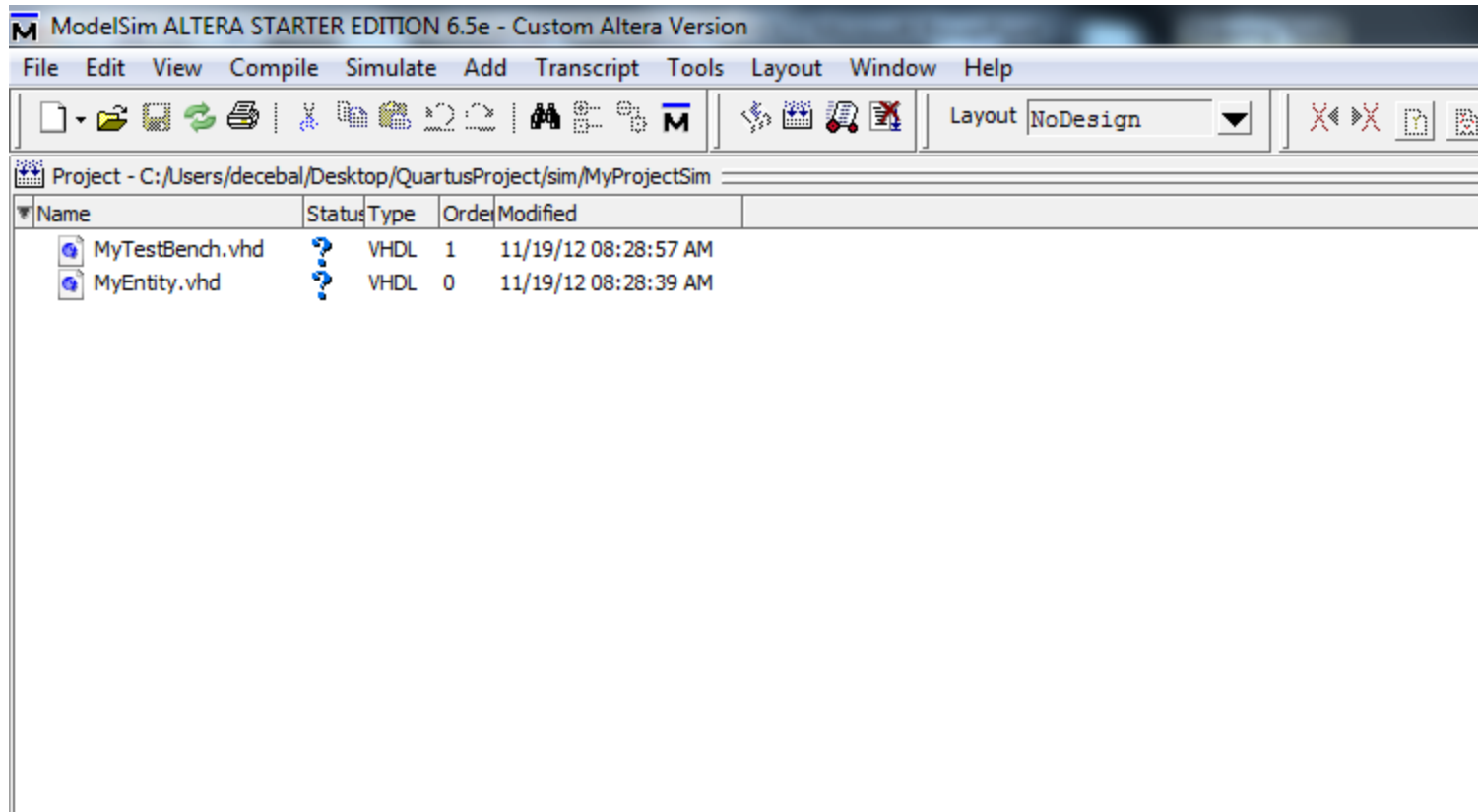
Testbench

- Cherchez le répertoire que vous avez créé. Donnez un nom à votre projet. Faites Ok.



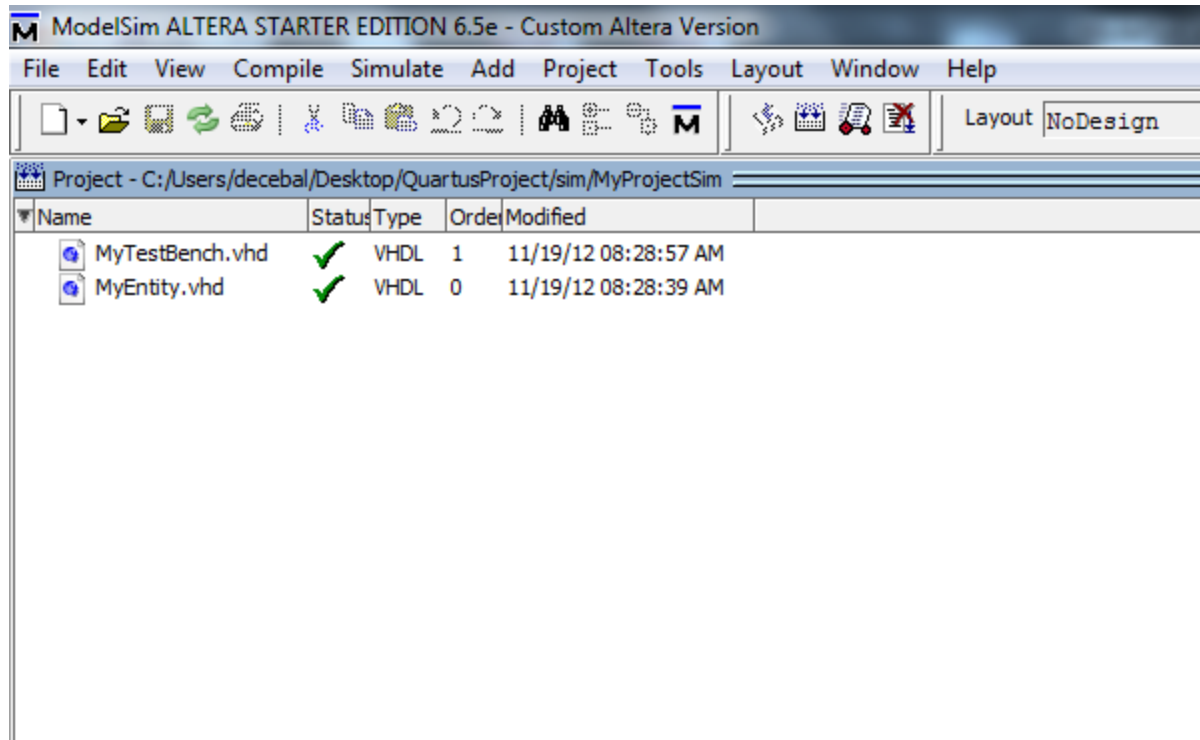
Testbench

- ModelSim va vous demander si vous voulez ajouter des fichiers à votre projet. Cliquez sur '*Add Existing File*'.
- Sélectionnez tous les fichiers que vous utilisez, y compris le testbench. Faites Ok et après Close. Vous allez voir vos fichiers dans ModelSim.



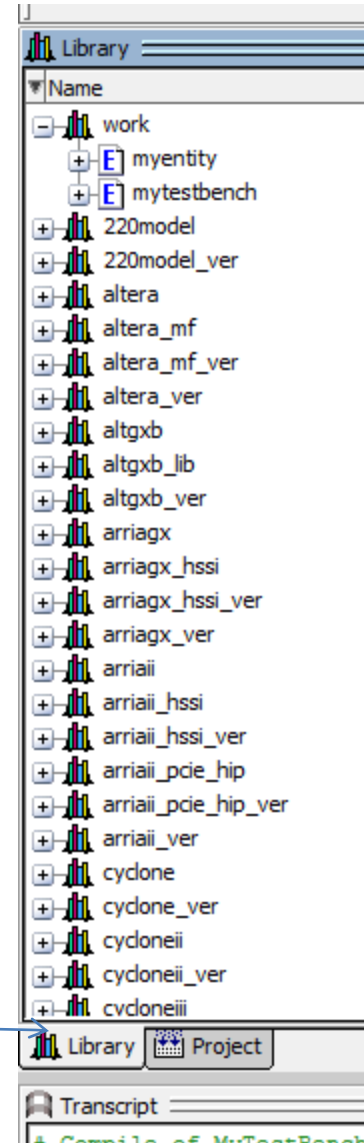
Testbench

- Avec un clic de souris droit, sélectionnez *'Compile' → 'Compile All'*



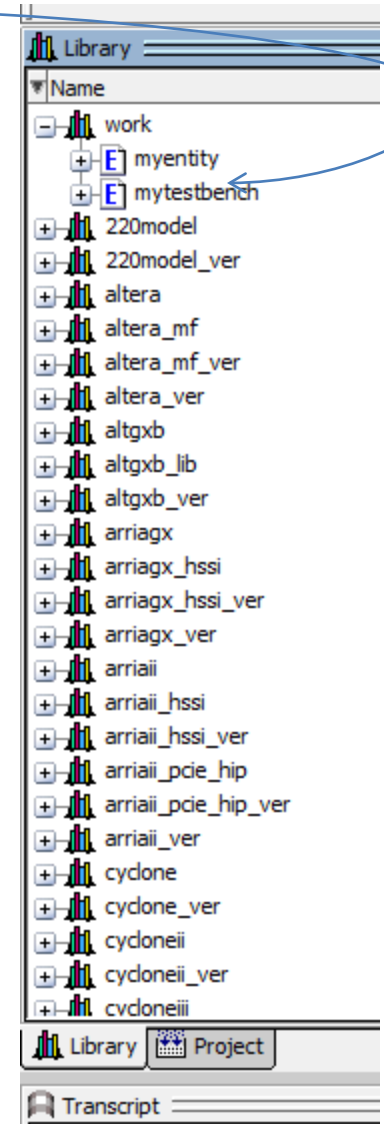
Testbench

- Cliquez sur '*Library*' et déroulez la librairie '*work*'.



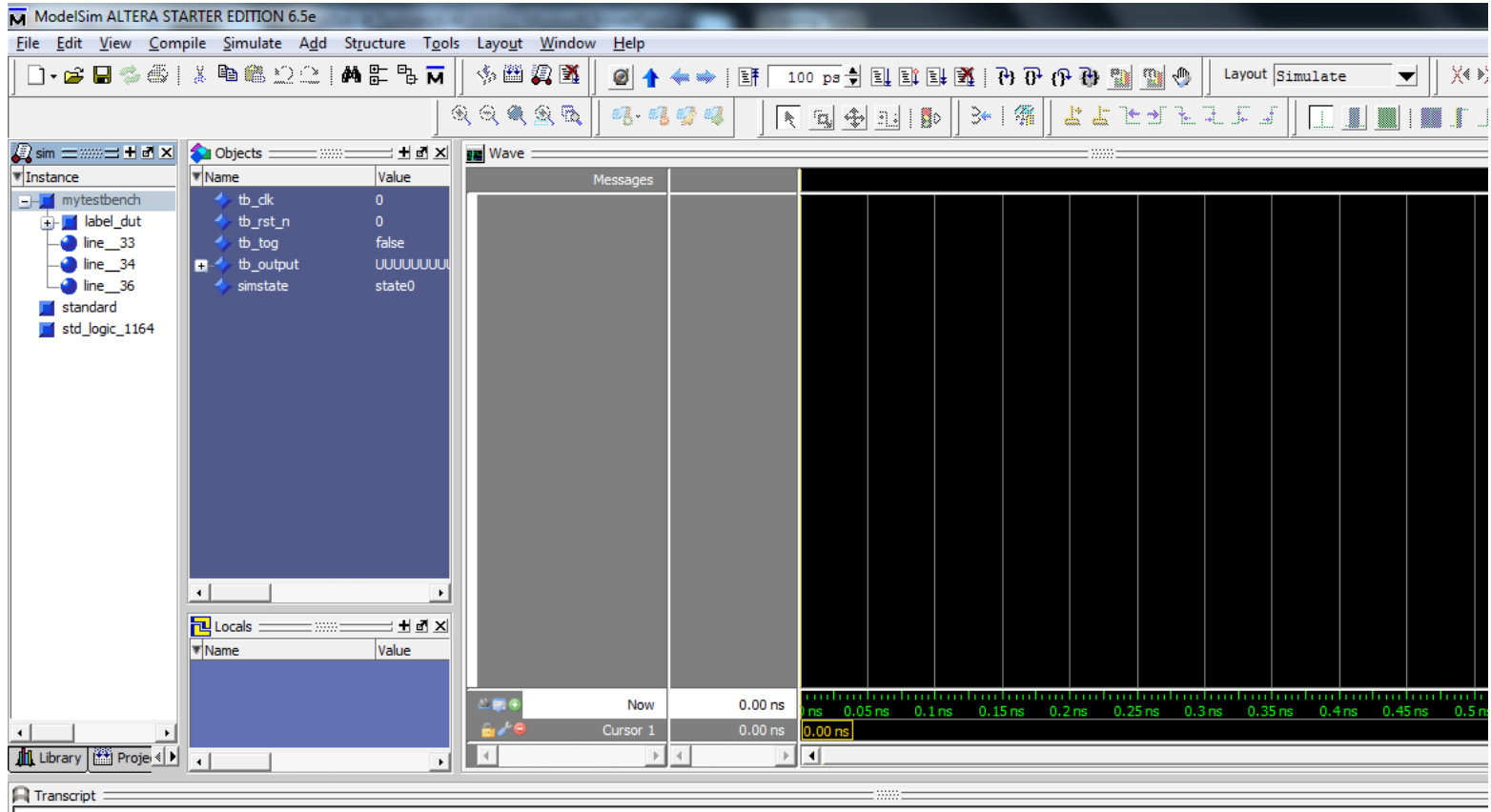
Testbench

- Cliquez deux fois sur le nom du fichier testbench. Ici, c'est '*mytestbench*'.
- L'environnement de simulation apparaît.



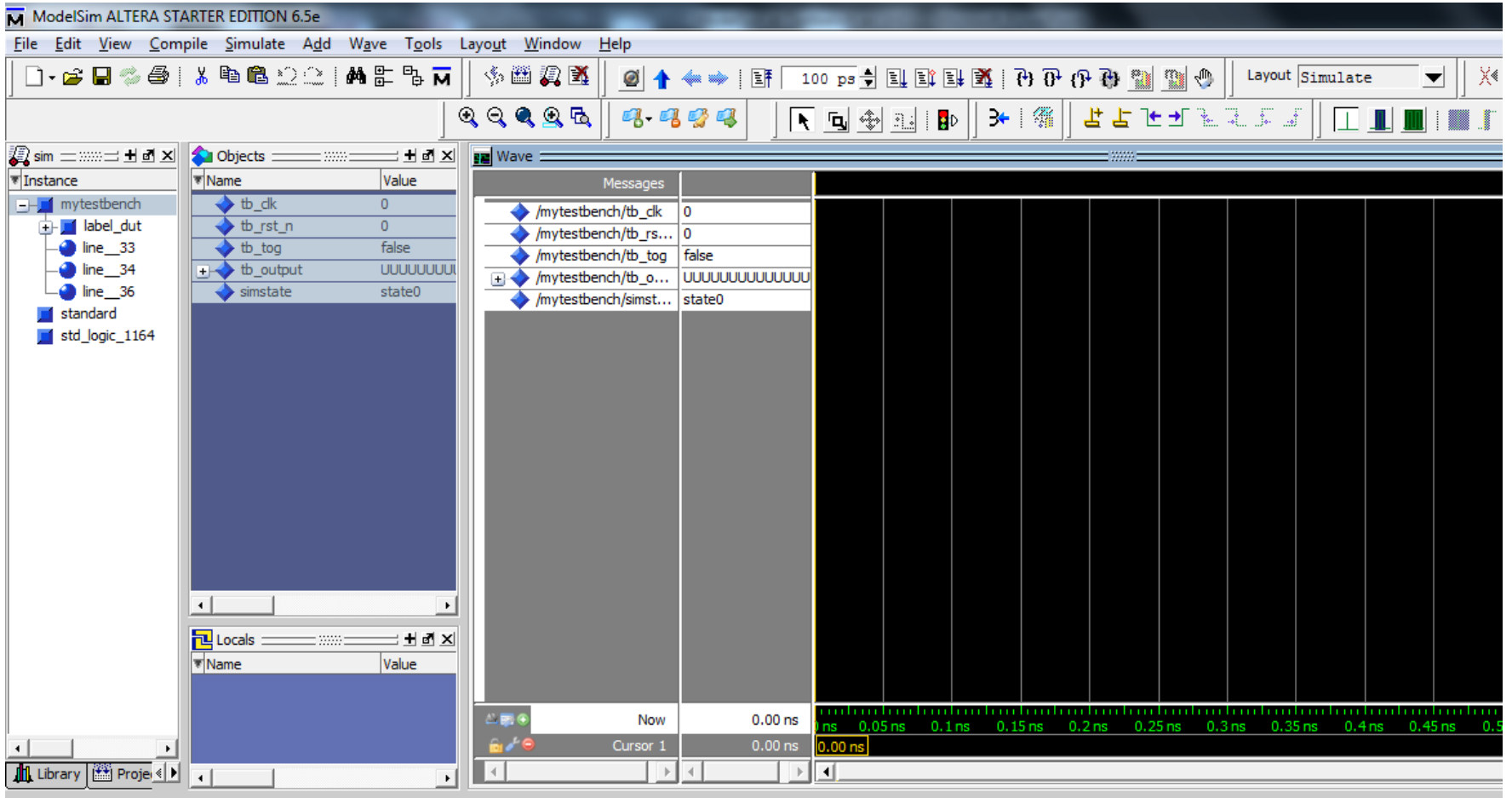
Testbench

- Pour voir le wave, faites '*View -> Wave*'.



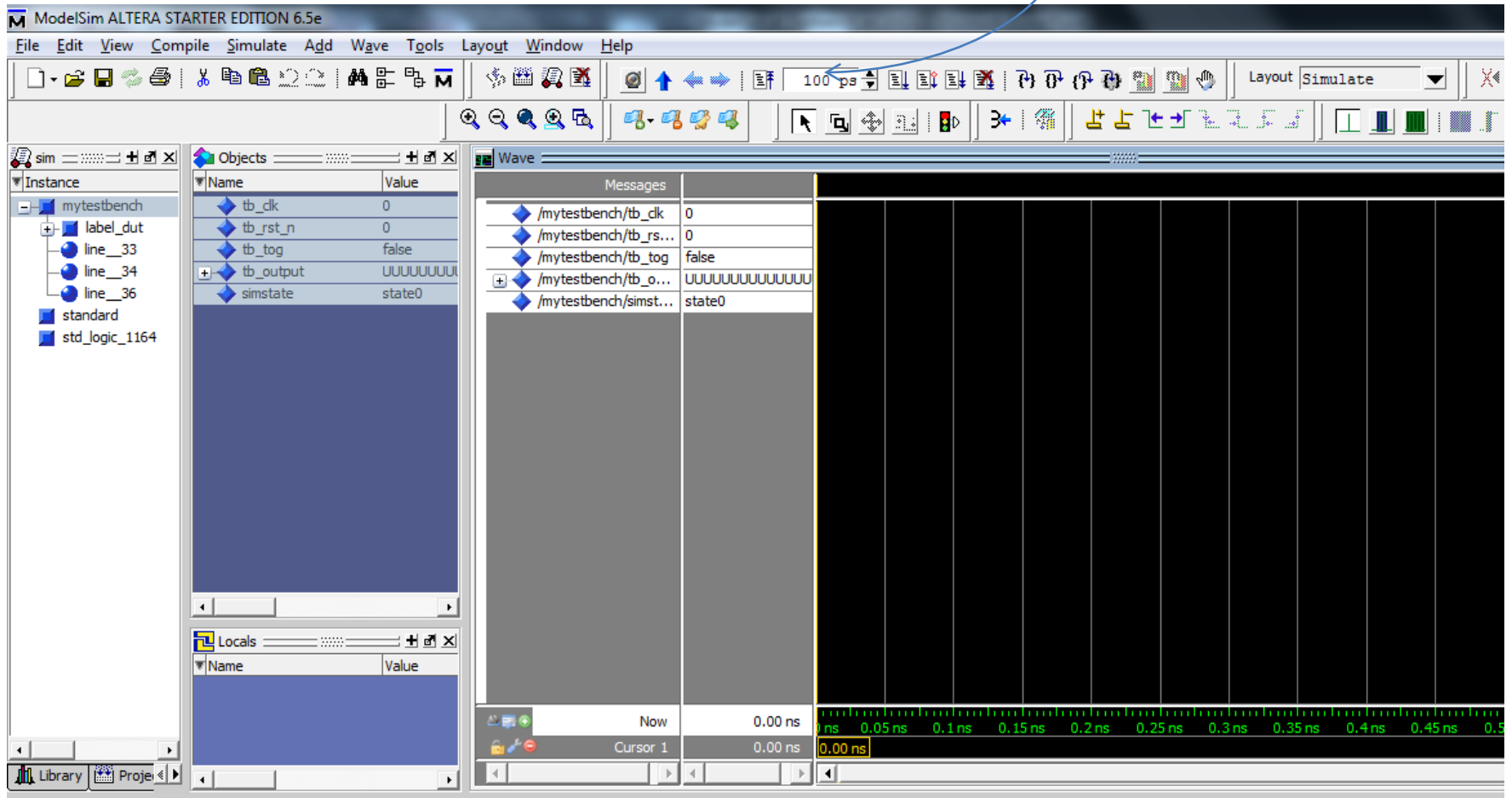
Testbench

- Sélectionnez tous les signaux que vous voyez dans 'Objects' et 'drag-and-drop'ez les dans le wave.



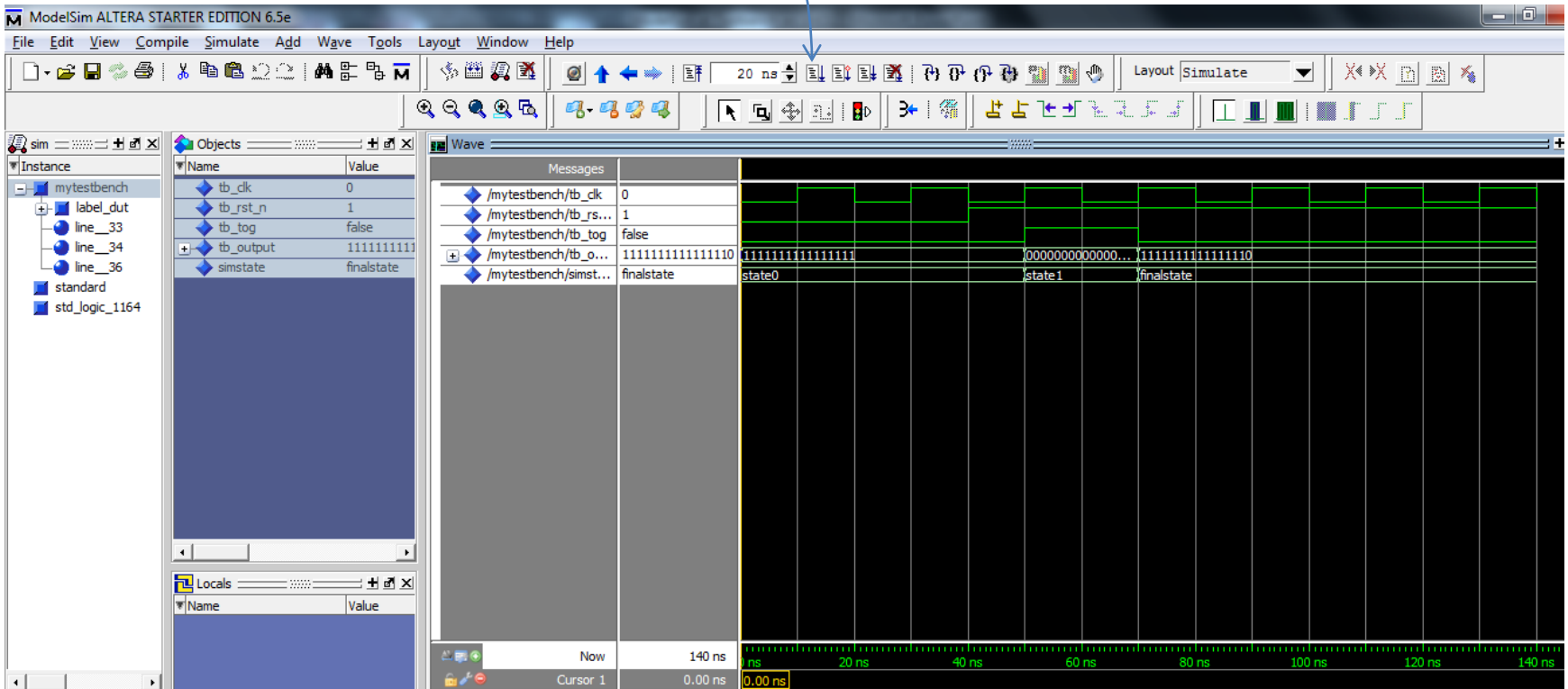
Testbench

- Dans cette case vous pouvez choisir le temps de chaque pas de la simulation. Dans le testbench on avait choisi une horloge de 20 ns par cycle.



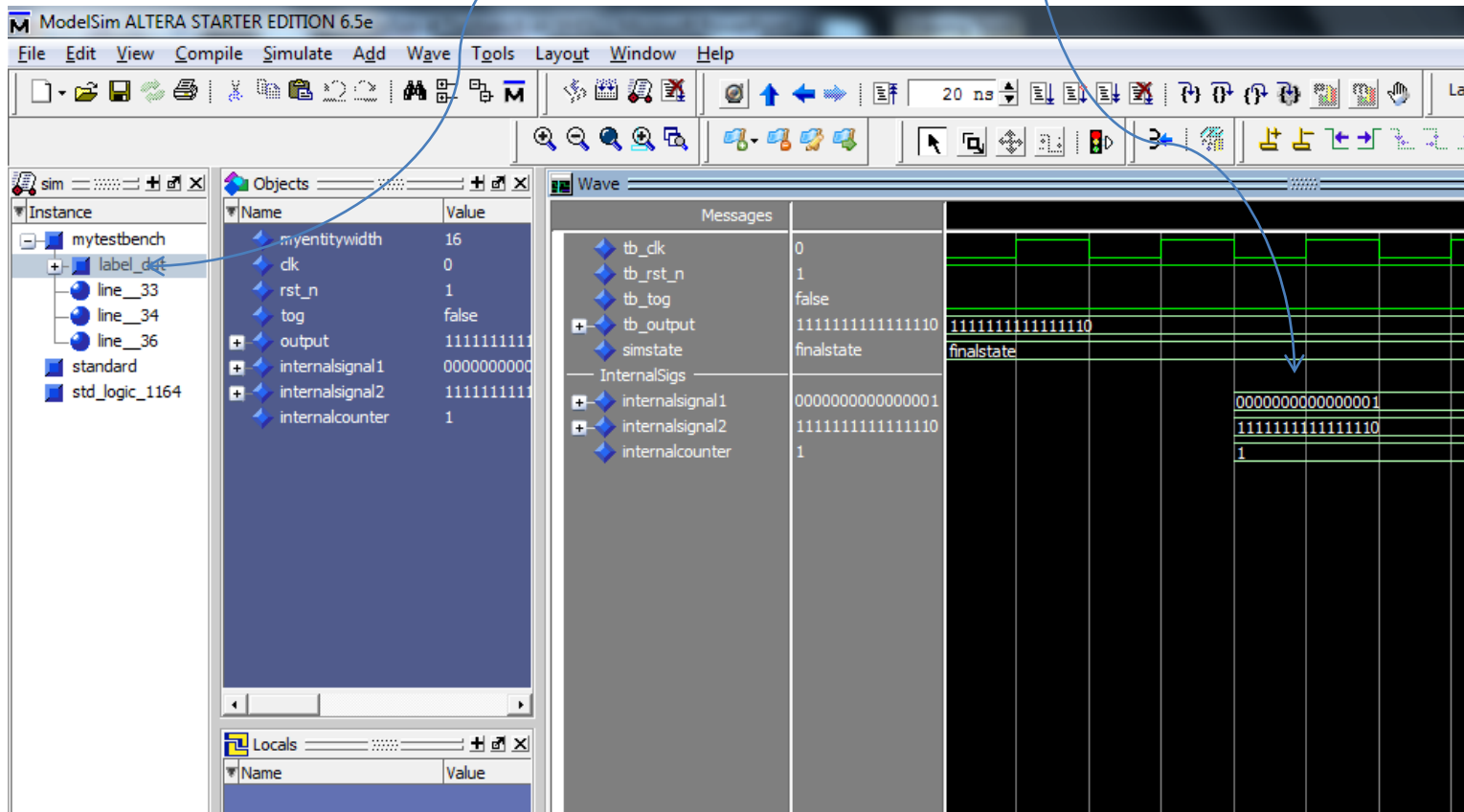
Testbench

- Cliquez sur ce bouton pour avancer dans la simulation.



Testbench

- Pour voir des signaux internes, qui n'apparaissent pas dans le testbench, cliquez ici. Vous pouvez les rajouter au wave pour observer ce qu'ils font.



Testbench

- Si vous changez votre code VHDL dans n'importe quel fichier de la simulation, il suffit de faire '*Compile -> Compile All*' suivi de Restart. **IMPORTANT DE GARDER L'ORDRE** sinon le simulateur ne prendra pas en compte les changements.

