



Your FPGA Design Partner  
swiss made

## Using Xilinx IP cores

**PROPRIETARY**

---

an.ipcore.rp.000.1 / version 0.1 / 07.06.2001

by gej

---

## AUTHORS

Laurent Gauch  
Amontec  
Survigne  
CH-1641 Vuippens  
[laurent.gauch@amontec.com](mailto:laurent.gauch@amontec.com)

Jean-Pierre Gehrig  
HEVs  
Rte du Rawyl 47  
1950 Sion  
[jpierre.gehrig@hevs.ch](mailto:jpierre.gehrig@hevs.ch)  
<http://www.amontec.com>

## TABLE OF CONTENTS

|     |  |    |
|-----|--|----|
| 1   | SUMMARY .....                          | 1  |
| 2   | GENERIC DESIGN FLOW .....              | 2  |
| 3   | UNDERSTANDING THE CORE GENERATOR ..... | 3  |
| 3.1 | Generating the core.....               | 4  |
| 4   | VHDL DESIGN ENTRY .....                | 5  |
| 4.1 | Writing the hdl import file.....       | 7  |
| 5   | SIMULATION .....                       | 11 |
| 6   | SYNTHESIS .....                        | 11 |
| 7   | PLACE AND ROUTE .....                  | 11 |

## 1 SUMMARY

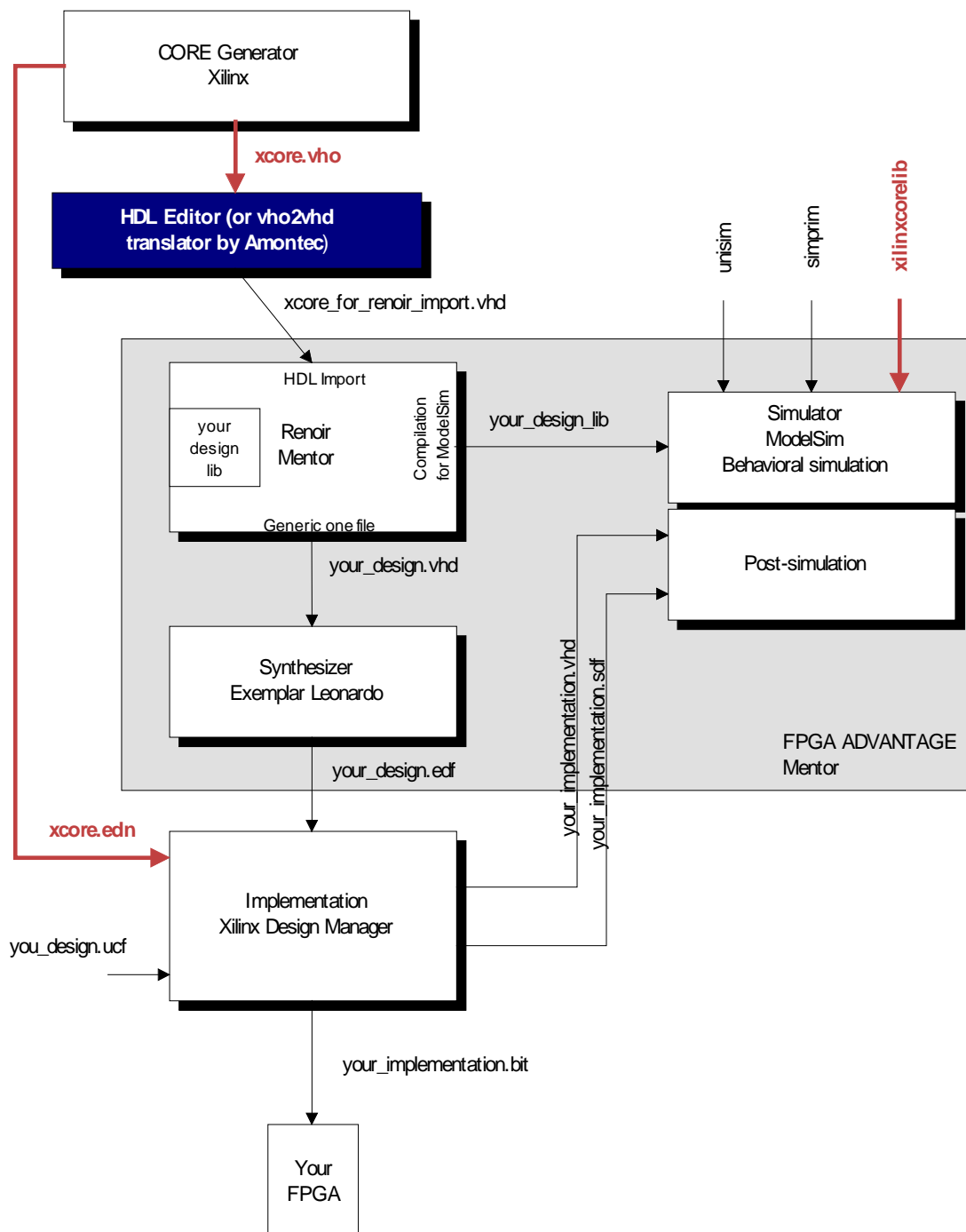
Xilinx Corporation distributes free and commercial Intellectual properties (IP) Cores for FPGAs through its Core Generator System. Various cores are already available and Xilinx regularly posts library updates on the Internet. The purpose of this document is to describe the integration of an IP Core from the Xilinx Core Generator System library into a standard VHDL project using the following EDA tools:

| Function      | Software          | Vendor    |
|---------------|-------------------|-----------|
| HDL Entry     | Renoir            | Mentor    |
| Simulation    | Modelsim          | Modeltech |
| Synthesis     | Leonardo Spectrum | Exemplar  |
| Place & Route | Design Manager    | Xilinx    |

*This document describes the integration of a dual port RAM core in a Spartan-II FPGA.*

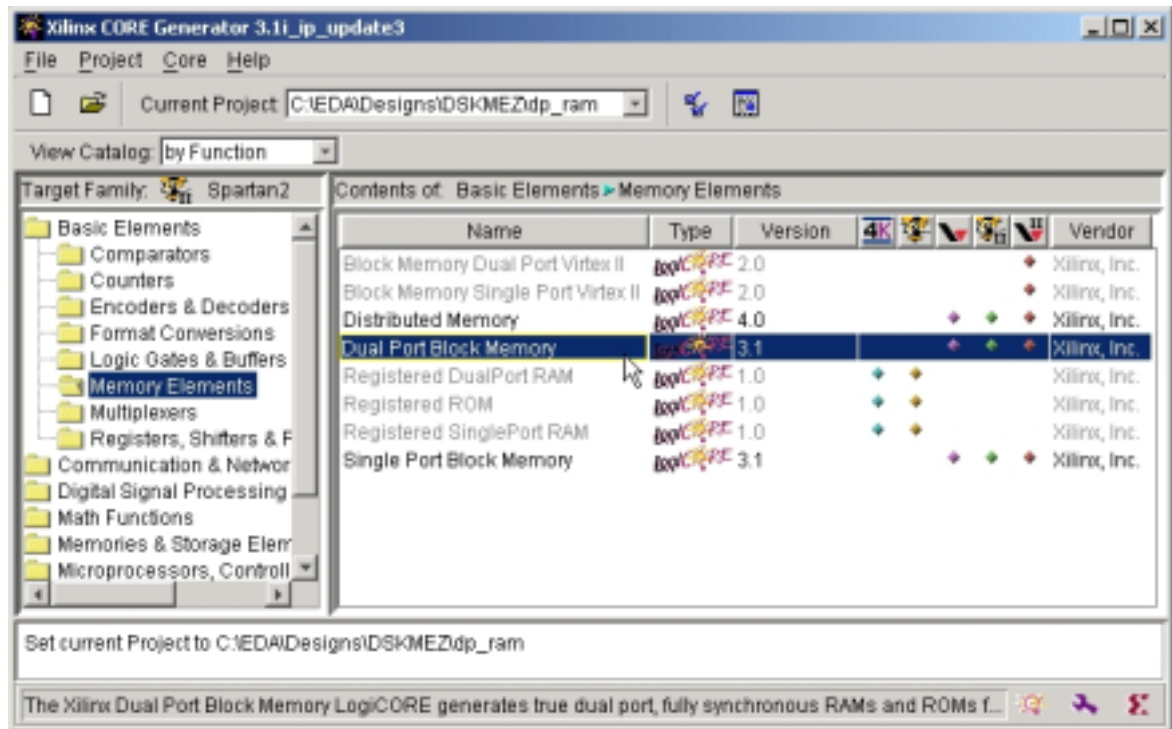
## 2 GENERIC DESIGN FLOW

The diagram below shows the typical design flow when integrating IP cores generated by the Xilinx Core Generator System library in a standard VHDL project.



### 3 UNDERSTANDING THE CORE GENERATOR

The Core Generator System is an IP library including FPGA cores from Xilinx and third-party companies. An IP core generally comes with a datasheet describing the core's interface and features.

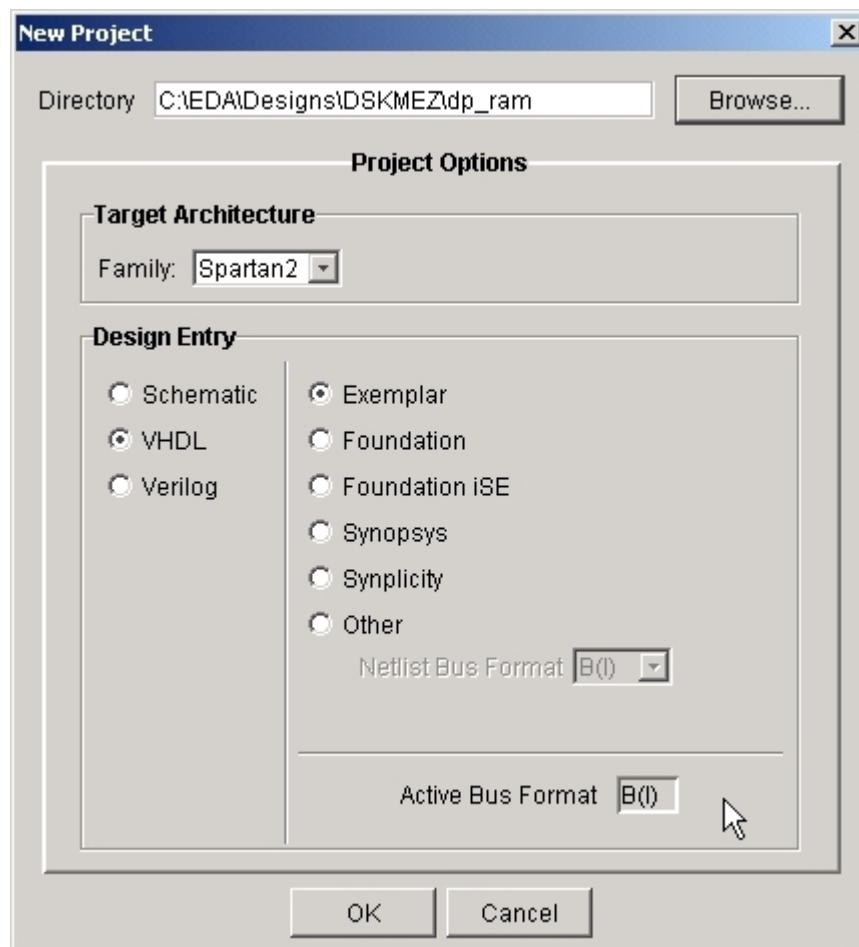


For more information about the Core Generator, please refer to the Xilinx documentation.



### 3.1 GENERATING THE CORE

When creating a new project, you are prompted to enter a directory path indicating where the generated core is to be saved. The Core Generator project directory should be set in the design directory next to other VHDL source files. The target architecture must be specified to allow the Core Generator to create an EDN file describing the core. Furthermore, the design entry method must be given in order to set the right Active Bus Format for the EDN file.



Now the desired IP core can be selected, configured and generated. A generation creates two files; the VHO file contains a VHDL instantiation template, which will be used to create a component during the HDL entry. The EDN file describes how the core is to be implemented by the Xilinx implementation tool.

## 4 VHDL DESIGN ENTRY

From the generated VHO file, a VHDL architecture can be written and imported by Renoir. The listing below shows the generated VHO file for a dual port RAM.

```
-----
-- This file was created by the Xilinx CORE Generator tool, and --
-- is (c) Xilinx, Inc. 1998, 1999. No part of this file may be --
-- transmitted to any third party (other than intended by Xilinx) --
-- or used without a Xilinx programmable or hardware device without --
-- Xilinx's prior written permission. --
-----

-- The following code must appear in the VHDL architecture header:

----- Begin Cut here for COMPONENT Declaration ----- COMP_TAG
component dp_ram16_32
  port (
    addrb: IN std_logic_VECTOR(10 downto 0);
    addrb: IN std_logic_VECTOR(9 downto 0);
    clka: IN std_logic;
    clkb: IN std_logic;
    dina: IN std_logic_VECTOR(15 downto 0);
    dinb: IN std_logic_VECTOR(31 downto 0);
    douta: OUT std_logic_VECTOR(15 downto 0);
    doutb: OUT std_logic_VECTOR(31 downto 0);
    wea: IN std_logic;
    web: IN std_logic);
end component;

-- COMP_TAG_END ----- End COMPONENT Declaration -----

-- The following code must appear in the VHDL architecture
-- body. Substitute your own instance name and net names.

----- Begin Cut here for INSTANTIATION Template ----- INST_TAG
your_instance_name : dp_ram16_32
  port map (
    addrb => addrb,
    addrb => addrb,
    clka => clka,
    clkb => clkb,
    dina => dina,
    dinb => dinb,
    douta => douta,
    doutb => doutb,
    wea => wea,
    web => web);
-- INST_TAG_END ----- End INSTANTIATION Template -----

-- The following code must appear above the VHDL configuration
-- declaration. An example is given at the end of this file.

----- Begin Cut here for LIBRARY Declaration ----- LIB_TAG

-- synopsys translate_off

Library XilinxCoreLib;

-- synopsys translate_on
```

```
-- LIB_TAG_END ----- End LIBRARY Declaration -----

-- The following code must appear within the VHDL top-level
-- configuration declaration. Ensure that the translate_off/on
-- compiler directives are correct for your synthesis tool(s).

----- Begin Cut here for CONFIGURATION snippet ----- CONF_TAG

-- synopsys translate_off

    for all : dp_ram16_32 use entity XilinxCoreLib.blkmemdp_v3_1(behavioral)
        generic map(
            c_depth_b => 1024,
            c_depth_a => 2048,
            c_has_rdyb => 0,
            c_has_rdyb => 0,
            c_has_web => 1,
            c_has_wea => 1,
            c_sinitb_value => "00000000",
            c_has_doutb => 1,
            c_has_douta => 1,
            c_has_limit_data_pitch => 0,
            c_sinita_value => "0000",
            c_limit_data_pitch => 18,
            c_width_b => 32,
            c_width_a => 16,
            c_write_modeb => 0,
            c_write_modea => 0,
            c_addra_width => 11,
            c_has_ndb => 0,
            c_has_nda => 0,
            c_has_dinb => 1,
            c_has_dina => 1,
            c_pipe_stages_b => 0,
            c_pipe_stages_a => 0,
            c_has_rfdb => 0,
            c_has_rfda => 0,
            c_has_enb => 0,
            c_has_ena => 0,
            c_reg_inputsb => 0,
            c_reg_inputsa => 0,
            c_default_data => "0000",
            c_has_default_data => 1,
            c_mem_init_file => "mif_file_16_1",
            c_has_sinitb => 0,
            c_has_sinita => 0,
            c_enable_rlocs => 0,
            c_addrb_width => 10);
        end for;

-- synopsys translate_on

-- CONF_TAG_END ----- End CONFIGURATION snippet -----

-----
-- Example of configuration declaration...
-----
--
-- <Insert LIBRARY Declaration here>
--
-- configuration <cfg_my_design> of <my_design> is
--     for <my_arch_name>
--         <Insert CONFIGURATION Declaration here>
--     end for;
-- end <cfg_my_design>;
--
-- If this is not the top-level design then in the next level up, the following text
```

```
-- should appear at the end of that file:
--
-- configuration <cfg> of <next_level> is
--   for <arch_name>
--     for all : <my_design> use configuration <cfg_my_design>;
--     end for;
--   end for;
-- end <cfg>;
--
```

## 4.1 WRITING THE HDL IMPORT FILE

In order to create a component for Renoir, a VHD file must be written by copying some parts of the generated VHO file. Basically, the VHD import file contains the following code:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all;

ENTITY your_entity_name IS
  PORT(...);
END your_entity_name ;

-- renoir translate_off
LIBRARY XilinxCoreLib;
-- renoir translate_on

ARCHITECTURE bhv OF your_entity_name IS
  COMPONENT your_core_name
    PORT (...);
  END COMPONENT;

-- renoir translate_off
-- exemplar translate_off

  FOR ALL : your_core_name USE ENTITY XilinxCoreLib.xxx -- (behavioral)
    GENERIC MAP(...);

-- exemplar translate_on
-- renoir translate_on

BEGIN

-- renoir translate_off

  your_instance_name : your_core_name
    PORT MAP (...);

-- renoir translate_on

END bhv;
```

The `translate_xx` pragmas hide specific code referring to the `XilinxCoreLib` and interpreted only during a behavioral simulation. Thus, Renoir and the synthesizer aren't aware of the existence of the `XilinxCoreLib`, the core is seen as a black box.

The listing below shows the import file for the dual port RAM:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all;

ENTITY DSKMEZ_DP_RAM16_32 IS
  PORT(
    addra : IN      std_logic_VECTOR (10 downto 0) ;
    addrb : IN      std_logic_VECTOR (9  downto 0) ;
    clka  : IN      std_logic      ;
    clkb  : IN      std_logic      ;
    dina  : IN      std_logic_VECTOR (15 downto 0) ;
    dinb  : IN      std_logic_VECTOR (31 downto 0) ;
    wea   : IN      std_logic      ;
    web   : IN      std_logic      ;
    douta : OUT     std_logic_VECTOR (15 downto 0) ;
    doutb : OUT     std_logic_VECTOR (31 downto 0)
  );
END DSKMEZ_DP_RAM16_32 ;

-- renoir translate_off
LIBRARY XilinxCoreLib;
-- renoir translate_on

ARCHITECTURE bhv OF DSKMEZ_DP_RAM16_32 IS

-- Description : This file contains the declarations required for behavioral
--               simulation and synthesis of the dual port ram core created by
--               the Xilinx core generator system.

COMPONENT dp_ram16_32
  PORT (
    addra: IN std_logic_VECTOR(10 DOWNTO 0);
    addrb: IN std_logic_VECTOR(9  DOWNTO 0);
    clka: IN std_logic;
    clkb: IN std_logic;
    dina: IN std_logic_VECTOR(15 DOWNTO 0);
    dinb: IN std_logic_VECTOR(31 DOWNTO 0);
    douta: OUT std_logic_VECTOR(15 DOWNTO 0);
    doutb: OUT std_logic_VECTOR(31 DOWNTO 0);
    wea: IN std_logic;
    web: IN std_logic);
END COMPONENT;

-- renoir translate_off
-- exemplar translate_off

FOR ALL : dp_ram16_32 USE ENTITY XilinxCoreLib.blkmemdp_v3_1 -- (behavioral)
  GENERIC MAP(
    c_depth_b => 1024,
    c_depth_a => 2048,
    c_has_rdyb => 0,
    c_has_rdya => 0,
    c_has_web  => 1,
    c_has_wea  => 1,
    c_sinitb_value => "00000000",
    c_has_doutb => 1,
    c_has_douta => 1,
    c_has_limit_data_pitch => 0,
    c_sinita_value => "0000",
    c_limit_data_pitch => 18,
    c_width_b => 32,
    c_width_a => 16,
    c_write_modeb => 0,
```

```
        c_write_modea => 0,
        c_addra_width => 11,
        c_has_ndb => 0,
        c_has_nda => 0,
        c_has_dinb => 1,
        c_has_dina => 1,
        c_pipe_stages_b => 0,
        c_pipe_stages_a => 0,
        c_has_rfdb => 0,
        c_has_rfda => 0,
        c_has_enb => 0,
        c_has_ena => 0,
        c_reg_inputsb => 0,
        c_reg_inputsa => 0,
        c_default_data => "0000",
        c_has_default_data => 1,
        c_mem_init_file => "mif_file_16_1",
        c_has_sinitb => 0,
        c_has_sinita => 0,
        c_enable_rlocs => 0,
        c_addrb_width => 10);

-- exemplar translate_on
-- renoir translate_on

BEGIN

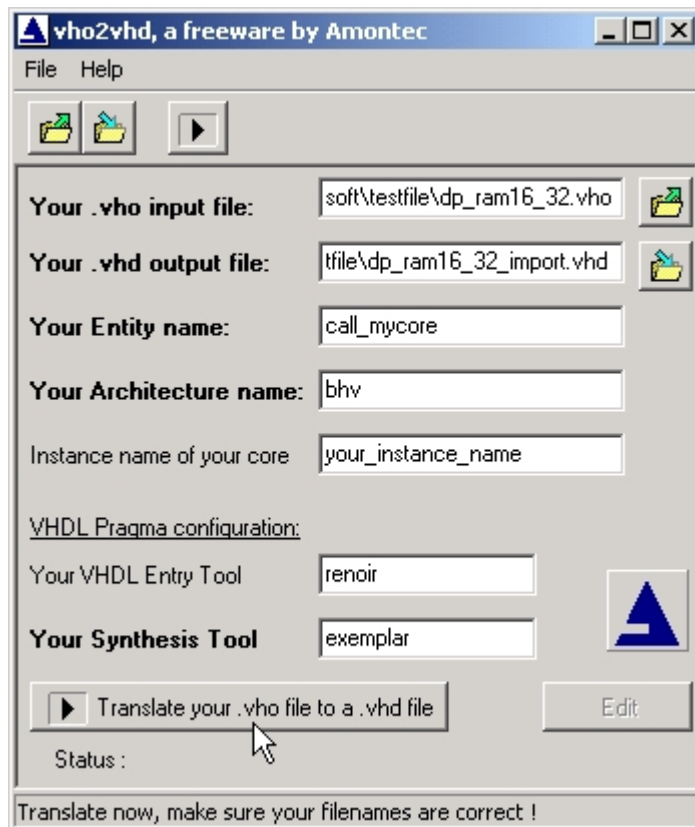
-- renoir translate_off

    my_ram : dp_ram16_32
        PORT MAP (
            addra => addra,
            addrb => addrb,
            clka => clka,
            clkb => clkb,
            dina => dina,
            dinb => dinb,
            douta => douta,
            doutb => doutb,
            wea => wea,
            web => web);

-- renoir translate_on

END bhv;
```

To do easy the generation of this .vhd file including the VHDL architecture, Amontec has written a freeware vho2vhd.exe tool. You can download it on [www.amontec.com](http://www.amontec.com). The following picture shows the screen window of the vho2vhd.exe:



When you have generated your .vhd file, use the HDL Import command of Renoir to automatically create a component. Renoir will also create a symbol, which you may want to modify according to the desired pinout.

## 5 SIMULATION

In order to simulate a design integrating an IP core, you need to create the `unisim`, `simprim` and `xilinxcorelib` libraries with Modelsim. Then, the sources located in:

```
<Xilinx Core Generator installation directory>\vhdl\src\unisims  
<Xilinx Core Generator installation directory>\vhdl\src\simprims  
<Xilinx Core Generator installation directory>\vhdl\src\XilinxCoreLib
```

must be compiled to the newly created libraries.



*The XilinxCoreLib source directory contains a `vhdl_analyze_order` file indicating a specific file compilation order. A nice way to compile the library is to create a compilation script using the Find/Replace command of your favorite text editor to obtain a list of Modelsim compilation commands like this:*

```
vcom -reportprogress 300 -work xilinxcorelib {<src_lib_dir>/ul_utils.vhd}  
vcom -reportprogress 300 -work xilinxcorelib {<src_lib_dir>/dafir_pack.vhd}  
...
```

Once the libraries are set up correctly, the design including the IP core can be generated and compiled with Renoir in order to be loaded into Modelsim for a behavioral simulation.

## 6 SYNTHESIS

During the synthesis an IP core is treated as a black box. Therefore, all references to the XilinxCoreLib must be hidden using the pragmas added to the VHDL import file described in chapter 4.1. Thus, the generic VHDL file created by Renoir will not instance any element from the XilinxCoreLib.

## 7 PLACE AND ROUTE

During the core generation, an EDN file was created for the IP core. This file contains a detailed description on how the core is to be implemented into a specific FPGA family (Spartan-II, Virtex, ...). The file must be copied to the same directory as the EDIF file created during synthesis.