HOGESCHOOL VOOR WETENSCHAP & KUNST DE NAYER INSTITUUT SINT-KATELIJNE-WAVER

DE NAYER Instituut
J. De Nayerlaan 5
B-2860 Sint-Katelijne-Waver
Tel. (015) 31 69 44
Fax. (015) 31 74 53
e-mail: ppe@denayer.wenk.be
ddr@denayer.wenk.be
tti@denayer.wenk.be
website: emsys.denayer.wenk.be

LEON 2 1.0.15 Tutorial

Custom Xilinx platform Version 1.11

HOBU-Funds
Project IWT 020079

Title	:	Embedded systemdesign based upon Soft- and Hardcore FPGA's
Projectleader	:	Ing. Patrick Pelgrims
Projectassistants	:	Ing. Dries Driessens Ing. Tom Tierens

Introduction

This tutorial is created to help you design your first embedded system with a Leon SPARC softcore processor. Before you proceed you must have the following software and hardware:

Software:
- Xilinx ISE 5.x
- Cygwin, not Xygwin **(installed with devel. tools option)**
- WinRAR
- Synplify PRO 7.2 *[Optional]*

Hardware:
- windows-PC
- Linux-PC
- Xilinx FPGA development board

Chapter I is a general chapter where you can learn how to define your Leon SPARC processor system.

In chapter II, there are possible 2 tracks to follow:

TRACK 1: completing the design with Xilinx ISE:
Easiest way to set up a Leon SPARC microprocessor system on an FPGA. Synthesis, place & route is done with Xilinx XST.

TRACK 2: completing the design with a third party synthesis tool
An alternative way to set up the Leon SPARC microprocessor system on an FPGA. Synthesis is done in a third party tool (in this case SYNPLIFY PRO), place and route is done in Xilinx ISE

Chapter III describes how to download the embedded system into the FPGA and how to run it.

I Defining the embedded system

1) download and installation of leon 2 source files:

   a) download leon2-1.0.15.tar.gz from http://www.gaisler.com

   b) unzip the tar.gz archive to e.g. **c:\leon2** (From now on we will suppose that this is your working directory. If you choose another one, you'll have to replace "c:\leon2" in this tutorial with your own directory.)

   c) To get quick access to the leon configuration utility (which makes use of cygwin), create 2 files in the same directory:

leon.bat

```
@C:\Cygwin\bin\bash.exe --rcfile c:\leon2\leon.bashrc
```

leon.bashrc

```
PATH="/bin:/contrib/bin: /cygdrive/c/leon2/leon"
cd /cygdrive/c/leon2
```

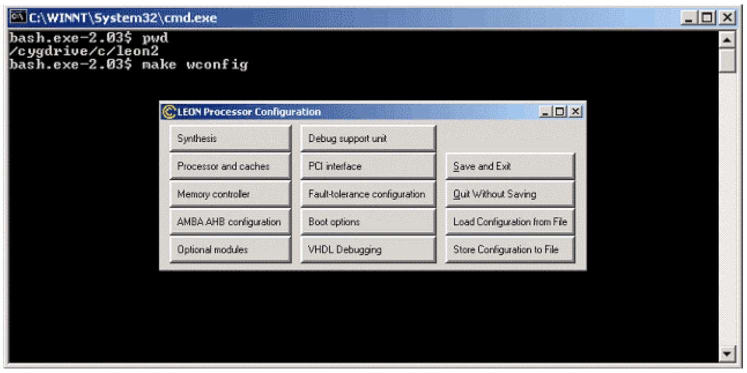
2) configure the processor

   a) start cygwin by doubleclicking on leon.bat

   b) normally you should be in the correct directory (you can check it with the "pwd"-command)

   c) start the leon configuration utility by typing "**make xconfig**" (figure 1)

Figure 1: start the leon configuration-utility

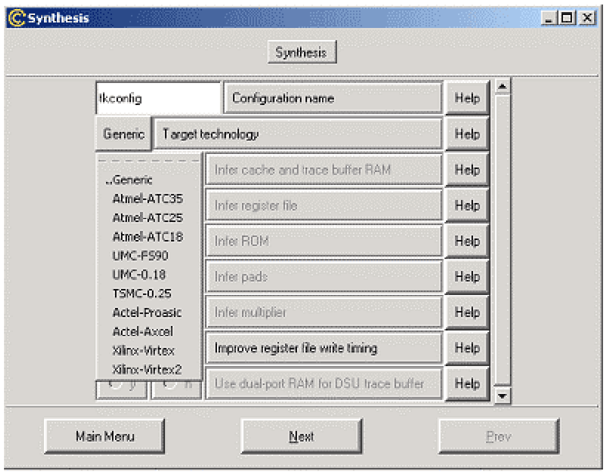


```
bash.exe-2.03$ pwd
/cygdrive/c/leon2
bash.exe-2.03$ make wconfig
```

d) In the leon configuration utility you can define your own processor system. Below you can find a short description of the necessary configuration you've got to do for building your own embedded system on an FPGA. Most of the settings are correct by default, to make sure however that everything is configured correctly, run trough the following steps.

   i) Click on the "synthesis"-option (figure 2), where you'll be able to define your target technology. In your case this will be "Xilinx Virtex (2)" for "Virtex (2)" targets, or "generic" for spartan targets. All other parameters shouldn't be changed unless you know what you're doing.

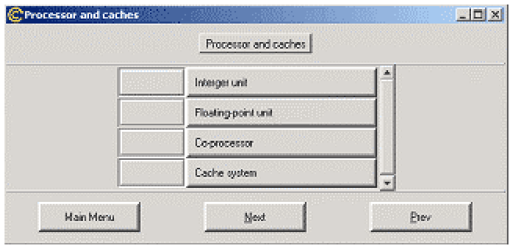
Figure 2: The synthesis options window



---

   ii) In the "processor and caches" window (figure 3) you can configure the integer unit and the cache system. For FPGA's this means the following (to avoid possible timing problems):

Figure 3: The processor and caches window



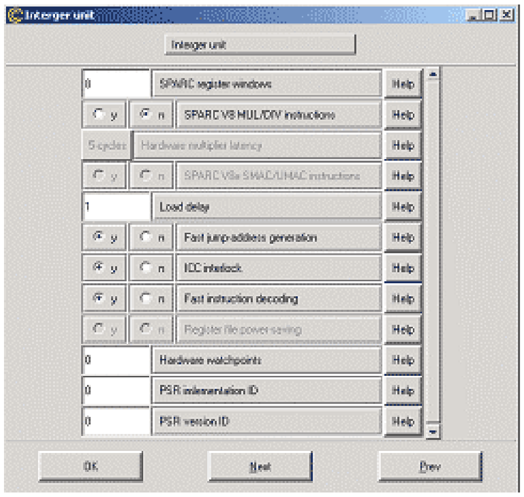
---

(1) Disable the floating point unit and coprocessor unit.

(2) Configure the integer unit (figure 4):

- Make sure "fast jump-address" is selected
- Make sure "ICC interlock" is selected
- Make sure "fast instruction decoding" is selected

Figure 4: The integer unit window



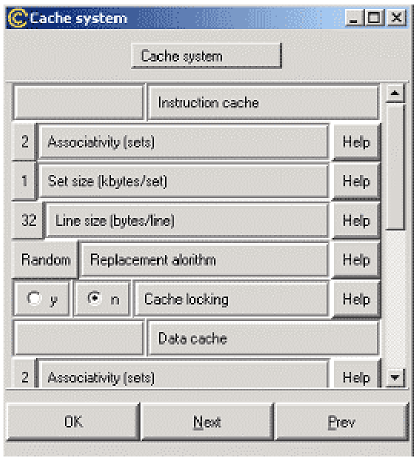
Integer unit

0	SPARC register windows
y [✓ n]	SPARC V8 MUL/DIV instructions
5 cycles	Hardware multiplier latency
y n	SPARC V8e SMAC/UMAC instructions
1	Load delay
[✓ y] n	Fast jump-address generation
[✓ y] n	ICC interlock
[✓ y] n	Fast instruction decoding
y n	Register file power saving
0	Hardware watchpoints
0	PSR implementation ID
0	PSR version ID

(3) Configure the data and instruction cache system (figure 5). Depending on the amount of block RAM available, you can use different values. The following are example settings for instruction and data cache:

- Associativity on 2 sets
- Set size on 1 kbyte/set
- Line size on 32 bytes/line
- Random replacement algorithm

Figure 5: The cache system window

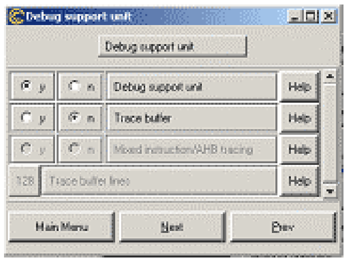


Cache system

	Instruction cache
2	Associativity (sets)
1	Set size (kbytes/set)
32	Line size (bytes/line)
Random	Replacement alorithm
y [✓ n]	Cache locking
	Data cache
2	Associativity (sets)

iii) Turn on the "debug support unit" in the "debug support unit" window (figure 6)

Figure 6: The debug support unit window



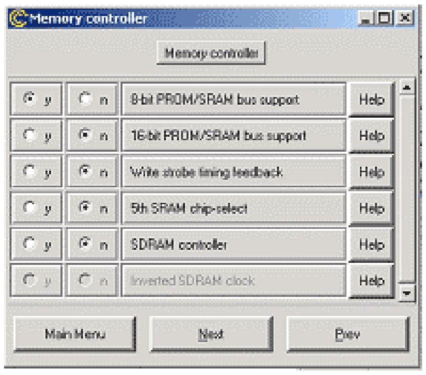
Debug support unit

[✓ y]	n	Debug support unit	Help
y	[✓ n]	Trace buffer	Help
y	n	Mixed instruction/AHB tracing	Help
128		Trace buffer lines	Help

iv) Now you've got to configure your "memory controller" (figure 7). Depending on your hardware, use the following settings:

- You've got SRAM:
    - o Select 8/16 bit prom/sram bus depending on the data bus width to your SRAM/PROM. For 32 bit access select none.
- You've got SDRAM
    - o Make sure "SDRAM controller" is selected
    - o Make sure you selected the "inverted SDRAM clock" option when using an FPGA. Otherwise you could create an unstable system caused by clock-skew
- You want to use on-chip RAM (look at step (v))

Figure 7: The memory controller window



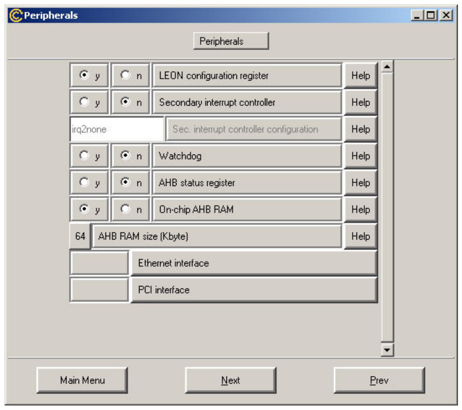
Memory controller

[✓ y]	n	8-bit PROM/SRAM bus support	Help
y	[✓ n]	16-bit PROM/SRAM bus support	Help
y	[✓ n]	Write strobe timing feedback	Help
y	[✓ n]	5th SRAM chip-select	Help
y	[✓ n]	SDRAM controller	Help
y	n	Inverted SDRAM clock	Help

v) In the "peripherals" window (figure 8), make sure that the Leon configuration register is selected. If you need on chip RAM (AHB-RAM), select the On-chip AHB RAM option, and define its size.

<u>Hint:</u> Take care though that you don't exaggerate it's size, as there is a very limited amount of RAM available in the FPGA, and the cache also makes use of it. Disable the Ethernet and PCI interface.

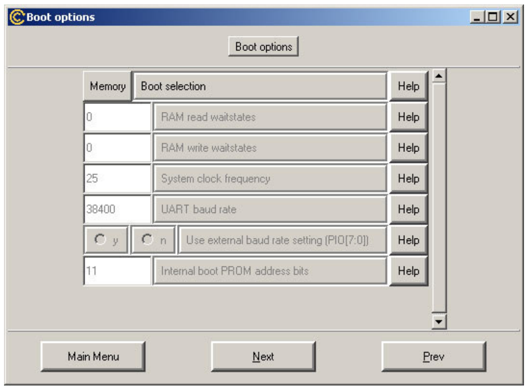
Figure 8: The peripherals window


Peripherals

[✓ y]	n	LEON configuration register	Help
y	[✓ n]	Secondary interrupt controller	Help
irq2none		Sec. interrupt controller configuration	Help
y	[✓ n]	Watchdog	Help
y	[✓ n]	AHB status register	Help
[✓ y]	n	On-chip AHB RAM	Help
64		AHB RAM size (Kbyte)	Help
		Ethernet interface	
		PCI interface	

vi) In the "boot options" window (figure 8) you can select where you want your device to boot from. To avoid synthesis problems, select "memory" as boot location. This way the Leon will try to boot from address location 0x0, probably there will no valid data on address 0x0, but that's no problem as you'll still be able to connect to the debugging unit and download your executable manually.

Figure 9: The boot options window


Boot options

Memory		Boot selection	Help
0		RAM read waitstates	Help
0		RAM write waitstates	Help
25		System clock frequency	Help
38400		UART baud rate	Help
y	n	Use external baud rate setting (PIO[7:0])	Help
11		Internal boot PROM address bits	Help

vii) Don't change anything in the "VHDL-debugging" window.

e) The configuration of the LEON processor system is now complete. In the main window, push on "save and exit". This will save your configuration, after this you'll get the message that you have to type the "make dep" command to generate the proper VHDL configuration files. At the prompt type "**make dep**", and hit the enter button. The VHDL configuration file will now be generated automatically.

3) Adjust the top file to your needs

a) Create a backup copy of the file "c:\leon2\leon\leon.vhd".

b) Open “c:\leon2\leon\leon.vhd” and adjust the in & out ports to correspond with your pinning file*:

i) Critical basic signals are:

- resetn: active low reset signal, if possible connect it with a button, otherwise write a little reset-state machine in the leon entity that drives the signal low for some clock cycles.
- clk: The clock signal, if needed divide/multiply the incoming clock with your own state machine, or use the Virtex DLL’s. Make sure however that your target component is able to handle the clock speed you selected. Table 1 gives you an idea of what clock speed is achievable on different Xilinx hardware devices
- dsutx: debugging unit transmit UART port
- dsurx: debugging unit receive UART port
- dsuen: debugging unit enable, this active high input signal enables the debugging unit, a logical ‘1’ has to be provided to activate the debugging unit
- bexcn: memory mapped I/O bus exception, this active low input indicates a memory problem, a logical ‘1’ has to be provided to avoid exceptions
- brdyn: memory mapped I/O bus ready, this active low input indicates that the access to a memory mapped I/O area can be terminated on the next rising clock edge, a logical ‘1’ has to be provided when no memory mapped I/O is used
- dsubre: debugging unit break enable, this active high input will generate break condition and put the processor in debug mode, in this case a logical ‘1’ has to be provided.

Table 1: Leon clock speed

Technology	Timing
Xilinx Virtex 1000-4	19 MHz
Xilinx Virtex 1000-6	24 MHz
Xilinx Virtex 2 PRO 7-5	59 MHz
Xilinx Virtex 2 PRO 7-7	69 MHz
Xilinx Spartan 2E 600-6	33 MHz
Xilinx Spartan 2E 600-7	40 MHz

ii) Optional signals are:

- errorn: active low output which indicates error condition, connect to a led
- dsuact: active low output that indicates whether the debugging unit is active, connect to a led
- pio[15]: Transmit signal of standard I/O in/out UART
- pio[14]: Receive signal of standard I/O in/out UART

iii) When you use SRAM, following signals have to be connected:

- data: data path
- address: address path
- ramsn: RAM chip select
- ramoen: RAM output enable
- rwen: RAM write enable

iv) When you use SDRAM [PC100/PC133 compatible], following signals have to be connected:

- <u>data:</u> data path
- <u>address:</u> address path ([14:2] is used as address, [16:15] are the bank select signals)
- <u>sdcke:</u> clock enable
- <u>sdcsn:</u> chip select
- <u>sdwen:</u> write enable
- <u>sdrasn:</u> row address strobe
- <u>sdcasn:</u> column address strobe
- <u>sddqm:</u> data I/O mask
- <u>sdclk:</u> SDRAM clock

c) Save all changes to the leon top file

<u>HINT:</u> * The leon topfile makes use of “pads” to route it’s signals to the outside world. Depending on the selected target technologies (Atmel, Xilinx,…) other types of pads are inferred. The 3 types of paths you’ll encounter are: [pad = output pin, d/q/en: leon in/output]

- outpad(d: in st_logic, pad : out std_logic)
- inpad(pad: in st_logic, q : out std_logic)
- iopad(d: in st_logic, en : in std_logic, q : out std_logic, pad : inout std_logic)

With this knowledge you can easily create extra pads (e.g. an extra address pad to connect your SRAM and SDRAM simultaneously).

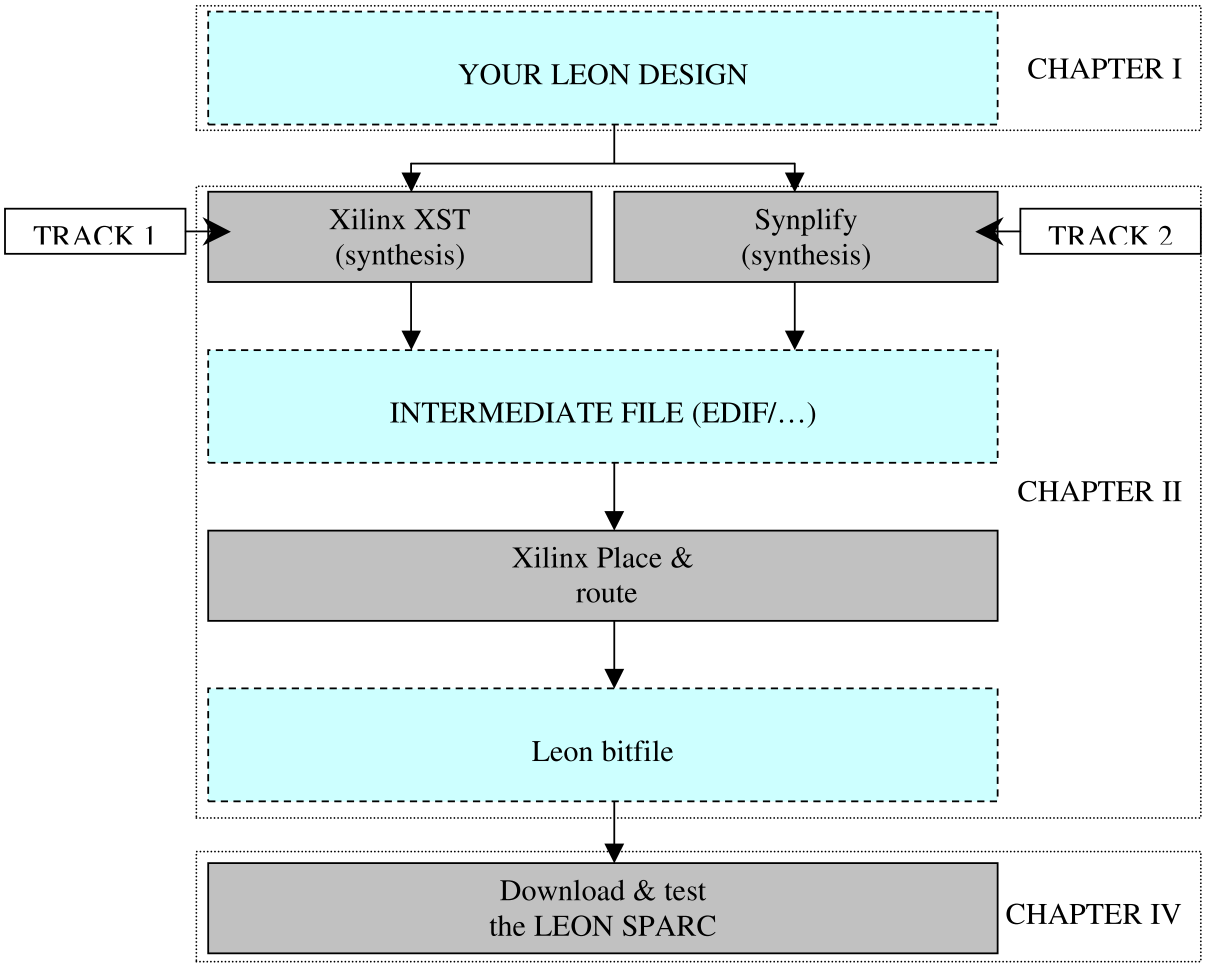
II Synthesis, place & route, generating the bitstream

As mentioned in the introduction, there are 2 possible tracks to complete your design (figure 10):

TRACK 1: completing the design with Xilinx ISE:
Easiest way to set up a Leon SPARC microprocessor system on an FPGA. Synthesis, place & route is done with Xilinx XST.

TRACK 2: completing the design with a third party synthesis tool
An alternative way to set up the Leon SPARC microprocessor system on an FPGA. Synthesis is done in a third party tool (in this case SYNPLIFY PRO), place and route is done in Xilinx ISE

Figure 10: The design flow



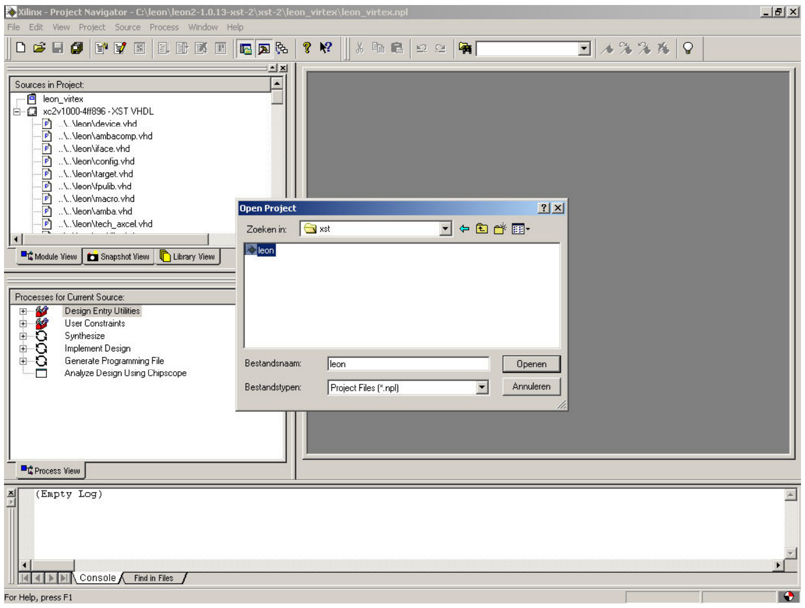
TRACK 1 Complete your design with Xilinx XST

1) Start Xilinx project navigator (*Start* → *programs* → *Xilinx ISE 5* → *Project navigator*)

2) Open the standard leon ISE project (File → open project)

3) Browse to “c:\leon2\syn\xst” (figure 11) and click on “Open”

Figure 11: Open the Leon SPARC ISE project



Xilinx - Project Navigator - C:\leon\leon2-1.0.13-xst-2\xst-2\leon_virtex\leon_virtex.npl

Sources in Project:

```
leon_virtex
xc2v1000-4ff896 - XST VHDL
    ..\..\leon\device.vhd
    ..\..\leon\ambacomp.vhd
    ..\..\leon\iface.vhd
    ..\..\leon\config.vhd
    ..\..\leon\target.vhd
    ..\..\leon\fpulib.vhd
    ..\..\leon\macro.vhd
    ..\..\leon\amba.vhd
    ..\..\leon\tech_axcel.vhd
```

Processes for Current Source:

```
Design Entry Utilities
User Constraints
Synthesize
Implement Design
Generate Programming File
Analyze Design Using Chipscope
```

Open Project

Zoeken in: xst

leon

Bestandsnaam: leon

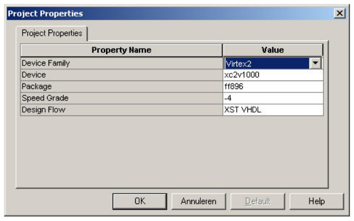
Bestandstypen: Project Files (*.npl)

```
(Empty Log)
```

For Help, press F1

4) Configure your target device by double-clicking on the currently selected hardware target in the “project workspace” (figure 12)

Figure 12: Configuring the target device



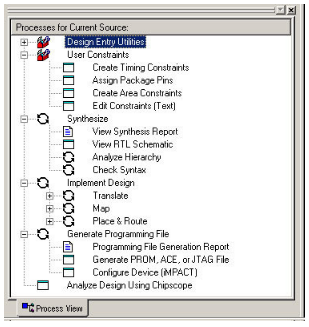
Project Properties

Property Name	Value
Device Family	Virtex2
Device	xc2v1000
Package	ff896
Speed Grade	-4
Design Flow	XST VHDL

5) Add your vhdl files that are needed to complete the design (e.g. clock divider,…) by right clicking on your selected hardware target and then clicking on “add source”. Also add your pinning file (*.UCF) and assign it to the leon top entity.

6) Select the “leon” top entity in the “project workspace”. This will show the available you the available processes for this file (figure 12)

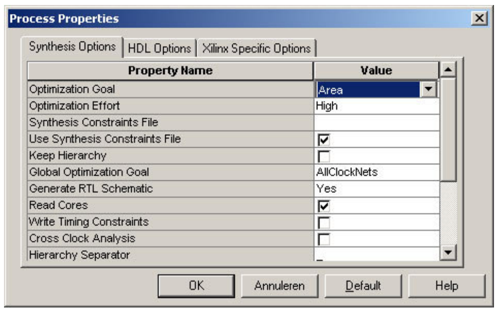
Figure 13: The process workspace



```
Processes for Current Source:
+ Design Entry Utilities
- User Constraints
    Create Timing Constraints
    Assign Package Pins
    Create Area Constraints
    Edit Constraints (Text)
- Synthesize
    View Synthesis Report
    View RTL Schematic
    Analyze Hierarchy
    Check Syntax
- Implement Design
  + Translate
  + Map
  + Place & Route
- Generate Programming File
    Programming File Generation Report
    Generate PROM, ACE, or JTAG File
    Configure Device (iMPACT)
  Analyze Design Using Chipscope
```

7) Right click on "synthesize", set optimisation goal on "Area" or "speed" depending on your own expectations, and set the optimisation effort to "high", push on "OK". (figure 14)

Figure 14: Process properties



Property Name	Value
Optimization Goal	Area
Optimization Effort	High
Synthesis Constraints File	
Use Synthesis Constraints File	✓
Keep Hierarchy	☐
Global Optimization Goal	AllClockNets
Generate RTL Schematic	Yes
Read Cores	✓
Write Timing Constraints	☐
Cross Clock Analysis	☐
Hierarchy Separator	_

8) Double-click on "synthesise" in the "process workspace" and your design will be synthesised. If there are any errors, you'll have to trace back to figure out what is going on.

9) Double-click on "Implement design" in the "process workspace" and your design will be placed & routed on your target component. If there are any errors, they will probably be generated because of errors in your pinning file (*.UCF).

   <u>HINT:</u> If you have unmatched LOC constraints in your pinning file, do not remove them as you might need them later. Instead use the following trick to avoid getting into trouble later:

   1. Open the preferences window (edit → preferences), select the "processes" tab, and select the "advanced" property display level.
   2. Right click on "Implement design", select "properties", and make sure the "allow unmatched LOC constraints" value is selected, after this, push on "OK".

10) Open the "Place & Route Report" by double-clicking on it in the "process workspace", and check whether your design fits into your component.

11) Open the “Text-based Post Place & Route Static Timing Report” by double-clicking on it in the “process workspace”, and check whether you meet your desired clock speed. If you don’t meet it, trace back and alter your design.

12) Double-click on “Generate Programming file” in “the process workspace” and the bitstream will be generated. Normally you will not experience any errors here

TRACK 2 Complete your design using Synplify.

1) Start synplify. (*start* → *programs* → *synplicity* → *synplify pro*)

2) Open the existing leon project (*File* → *open project* → *existing project*)

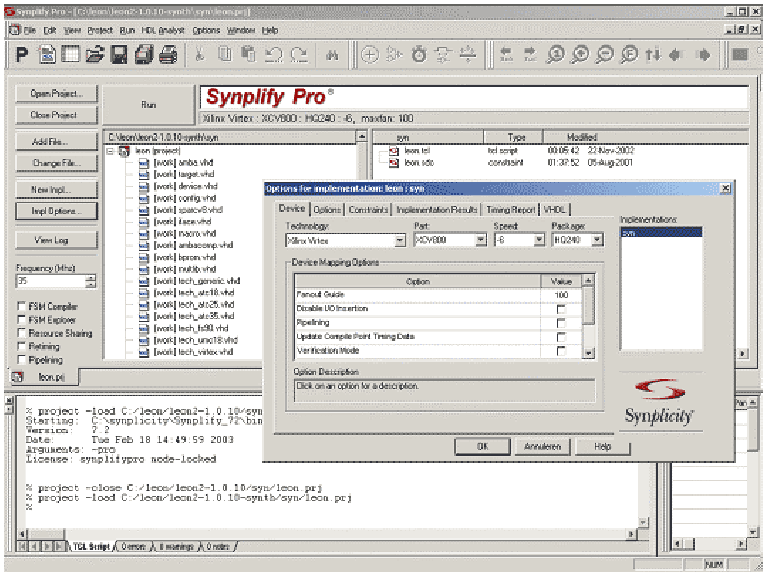
3) Browse to “c:\leon2\syn” and open the “leon” project.

4) Add your own vhdl files to the project by clicking on “add file”. Make sure your hdl files are synthesised before the leon top entity by optionally altering the synthesise order.

5) Configure the implementation options (*project* → *Implementation Options*) (figure 15)
   - *The device tab:*
     Select your target component (Technology/part/speed/package)
   - *The options tab:*
     Not a single option may be selected, as this could result into netlists with a non-working leon.
   - *The constraints tab:*
     Enter your desired clock speed
   - *Implementation results*
     Make sure the resulting format is “edif”
   - *Timing report:*
     These values should be correct
   - *Vhdl-tab:*
     Make sure the top level entity is “leon”

   Press on “OK”

Figure 15: Synplify implementation options



Xilinx Virtex : XCV800 : HQ240 : -6, maxfan: 100

C:\leon\leon2-1.0.10-synth\syn
leon [project]
[work] amba.vhd
[work] target.vhd
[work] device.vhd
[work] config.vhd
[work] sparcv8.vhd
[work] iface.vhd
[work] macro.vhd
[work] ambacomp.vhd
[work] bprom.vhd
[work] multlib.vhd
[work] tech_generic.vhd
[work] tech_atc18.vhd
[work] tech_atc25.vhd
[work] tech_atc35.vhd
[work] tech_fs90.vhd
[work] tech_umc18.vhd
[work] tech_virtex.vhd

syn	Type	Modified
leon.tcl	tcl script	00:05:42 22-Nov-2002
leon.sdc	constraint	01:37:52 05-Aug-2001

Frequency (Mhz): 35
☐ FSM Compiler
☐ FSM Explorer
☐ Resource Sharing
☐ Retiming
☐ Pipelining

leon.prj

Options for implementation: leon : syn

Device | Options | Constraints | Implementation Results | Timing Report | VHDL

Technology: Xilinx Virtex — Part: XCV800 — Speed: -6 — Package: HQ240

Implementations: syn

Device Mapping Options

Option	Value
Fanout Guide	100
Disable I/O Insertion	☐
Pipelining	☐
Update Compile Point Timing Data	☐
Verification Mode	☐

Option Description: Click on an option for a description.

```
% project -load C:/leon/leon2-1.0.10/syn
Starting:  C:\synplicity\Synplify_72\bin
Version:   7.2
Date:      Tue Feb 18 14:49:59 2003
Arguments: -pro
License: synplifypro node-locked

% project -close C:/leon/leon2-1.0.10/syn/leon.prj
% project -load C:/leon/leon2-1.0.10-synth/syn/leon.prj
%
```

6) Press on the famous “Run” button to start synthesis, wait until synthesis is complete

7) Press on the “view log” button, and check whether your desired clock speed has been met. If not, track back and change your clock divider to a lower value.

8) Now you have to place & route your design in Xilinx ISE.

    a. Start Xilinx ISE (*Start → programs → Xilinx ISE 5 → Project navigator*)

    b. Start a new project (*File → new project*)

    c. Select a project name, e.g. “leon” and a working directory (create a new one). Make sure the correct target device is selected, as design flow choose “EDIF”.

    d. Add the “leon.edf” file generated by synplify by right clicking on your target device in the “project workspace”, selecting “add source”, and adding the EDIF file to the project

    e. Add your pinning file (*.UCF) the same way

    f. Select the “leon” entity in the “project workspace”.

    g. Double-click on “Implement design” in the “process workspace” and your design will be placed & routed on your target component. If there are any errors, they will probably be generated because of errors in your pinning file (*.UCF).

       HINT: If you have unmatched LOC constraints in your pinning file, do not remove them as you might need them later. Instead use the following trick to avoid getting into trouble later:
       - Open the preferences window (edit → preferences), select the “processes” tab, and select the “advanced” property display level.
       - Right click on “Implement design”, select “properties”, and make sure the “allow unmatched LOC constraints” value is selected, after this, push on “OK”.

    h. Open the “Place & Route Report” by double-clicking on it in the “process workspace”, and check whether your design fits into your component.

    i. Open the “Text-based Post Place & Route Static Timing Report” by double-clicking on it in the “process workspace”, and check whether you meet your desired clock speed. If you don’t meet it, trace back and alter your design.

9) Double-click on “Generate Programming file” in the “process workspace” and the bitstream will be generated. Normally you will not experience any errors here.

III Download and test the Leon SPARC

1) Download your design with Xilinx iMPACT

   a. Make sure you connected your download cable correctly (e.g. parallel cable IV,…), make sure your board is powered on

   b. Start iMPACT *(Start → programs → Xilinx ISE 5 → Accessories → iMPACT)*

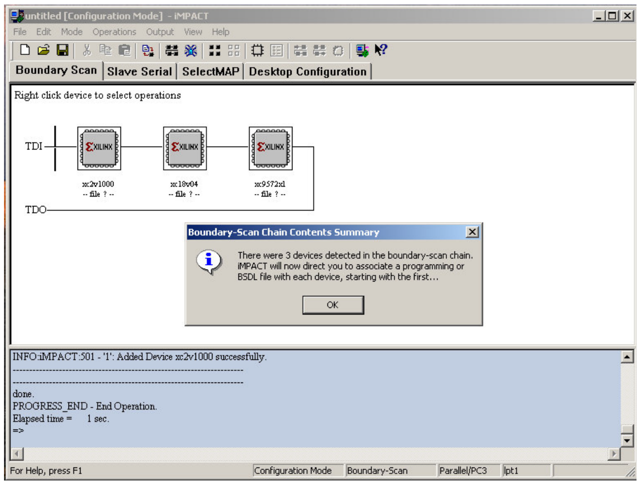
   c. As operation mode select “configure devices”, press on “next”

   d. Configure the device via “Boundary-Scan Mode”, press on “next”

   e. Select “automatically connect to cable and identify Boundary-Scan chain”, press on “next”

   f. Xilinx will report which hardware has been found on the JTAG chain (figure 16)

Figure 16: iMPACT



Right click device to select operations

There were 3 devices detected in the boundary-scan chain. iMPACT will now direct you to associate a programming or BSDL file with each device, starting with the first…

```
INFO:iMPACT:501 - '1': Added Device xc2v1000 successfully.
----------------------------------------------------------------------
----------------------------------------------------------------------
done.
PROGRESS_END - End Operation.
Elapsed time =      1 sec.
=>
```

   g. Now Xilinx asks for the configuration files for each device. Select “leon.”bit for the correct device and click op “open”, press “cancel” for the other devices.

   h. Right click on the target component and select “program…”, in the next screen press on “OK”, and your target device will be programmed.

2) Now you’ll have to try to connect to the leon and run “hello world”. (you need linux for that).

   a. Download LECCS from http://www.gaisler.org

   b. Install the LECCS tools with following commands into the /opt/rtems directory:
      - Cd /opt
      - gunzip –c leccs-linux.tar.gz | tar xf –

- unzip –c leccs-docs-tools.tar.gz | tar xf –
- unzip –c leccs-docs-rtems.tar.gz | tar xf –
- add /opt/rtems/bin to your search path

c. Now compile the test programs that are available within the installed package.
   - cd /opt/rtems/src/examples/samples
   - sparc-rtems-gcc –msoft-float –Ttext=XXXXXXXXXX hello.c –o hello
     (with -Ttext=0x40000000 for S(D)RAM or -Ttext=0x60000000 for AHB-RAM)

d. Connect the DSU-UART cable to you linux-system

e. Connect to the target board:
   - cd /opt/rtems/src/examples/samples
   - ./dsumon –i –u –uart /dev/ttyS0
     (or ttyS1 if you use COM1)

f. If all goes well, you get the following message (with your correct clock speed and other settings):

```
 LEON DSU Monitor, version 1.0.6
 Copyright © 2001, Gaisler Research - all rights reserved
 Comments or bug-reports to jiri@gaisler.com


 Clock frequency        :  20.28 MHz
 Register windows       :  8
 V8 hardware mul/div    :  no
 floating-point unit    :  not found
 instruction cache      :  2 * 1 kbytes, 32 bytes/line (2 kbytes  total)
 data cache             :  2 * 1 kbytes, 32 bytes/line (2 kbytes total)
 sram                   :  not found
 sdram                  :  not found
 stack pointer          :  0x43fffff0
dsu>
```

g. Connection with the LEON has been successful!

h. Use the following commands to load the "hello world" program into the leon
   - lo hello
   - run 0xXXXXXXXX (with XXXXXXXX = 40000000 for S(D)RAM, or 60000000 for AHB-RAM)

i. The message "Hello world" should appear on your screen!