Tutorial

Universität Rostock

Traditio et Innovatio

University of Rostock

Institute of Applied Microelectronics
and Computer Engineering

Dipl.-Ing. Philipp Gorski

SETUP: VLSI DESIGN ENVIRONMENT

Preface

This document contains a tutorial for the setup and installation processes of a complete VLSI Design / EDA environment based on opensource hardware/software design and simulation tools. This environment generally addresses students at the Institute of Applied Microelectronics and Computer Engineering (IMD), other electronic design students and hobby designers who need a fully bundled tool suite for their electronic design projects or study works. The included tools support all major levels of modern electronic circuit design and most of them have already proven their productivity in real world projects. In details, the tool suite covers the following domains on hard- and software side:

- Analog/Mixed Signal Design
- Circuit & PCB Design
- Circuit & Digital Logic Simulations
- Digital IC Design
- Embedded Systems Design
- Verilog, VHDL & SystemC

Moreover, this setup tutorial for the design environment targets a portable and flexible EDA solution without affecting an existing tool suite or configurations. For that purpose the installation on a virtual machine host represents the best way to go.

About the NoC Research Group @ IMD

The NoC Research Group bundles all research investigations in the domain of on-chips networks at the IMD in Rostock. Collaborations and discussions are welcome and you will find us under the following links:

Institute of Applied Microelectronics and Computer Engineering:

- **http://www.imd.uni-rostock.de**

NoC Research Group @ IMD:

- **http://www.networks-on-chip.com**

There you will get all information about us, the NoC Research team, contacts and descriptions about our current research projects / topics.

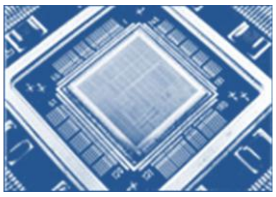


Table of contents

1 Introduction

This tutorial provides the setup of a complete VLSI Design / EDA environment as an OS-independent virtual machine (VM) on a host pc using the opensource virtualization software VirtualBox. Going that way offers the following advantages for the engineers or students:

- There will be one standard development system without the need to configure or install software on the host pc and independent of the utilized operating system on it (Linux, Unix, Windows with Cygwin/MinGW). Furthermore, no host specific changes or adaptation have to be regarded. It is just one simple way to go for all persons that wants to use it and offers cross-platform capabilities.

- The setup as virtual appliance offers a maximum portability and fits best for the usage in greater development teams (or student classes). Moreover, multiple of those systems in different versions can run in parallel without buggy interactions. When newer tool versions will be available you simply need to create a new virtual appliance to be up to date without the loss of the old development environment.

The EDA environment offers a great variety of tools that covers the complete workflow of electrical hard-/software engineers and is a buildup of the following main components:

- **Fedora 12 (Constantine):** Fedora is a Linux-based operating system that showcases the latest in free and open source software. This component represents the operating system of the EDA environment.

- **Fedora Electronic Lab (FEL):** Fedora's Electronic Laboratory is an opensource hardware design and simulation platform and is dedicated to support the innovation and development brought by opensource Electronic Design Automation (EDA) community.

- **SystemC Resources:** SystemC is a single, unified design and verification language that expresses architectural and other system-level attributes in the form of open-source C++ classes. It enables design and verification at the system level, independent of any detailed hardware and software implementation, as well as enabling co-verification with RTL design.

The usage of FEL in combination with the installation of SystemC on top of Fedora 12 offers tools and simulators for all levels of the hard-/software design process. In detail, this solution provides the following capabilities:

- **alliance** - VLSI EDA System
- **dinotrace** - Waveform viewer for electronics
- **dfu-programmer** - A Device Firmware Update based USB programmer for Atmel chips
- **electric** - Sophisticated Java based VLSI CAD System
- **emacs-dinotrace** - Elisp source files for dinotrace under GNU Emacs
- **emacs-verilog-mode** - Verilog mode for Emacs
- **emacs-vregs-mode** - Elisp source files for systemc-vregs under Emacs
- **eqntott** - Generates truth tables from Boolean equations

- **espresso-ab** - A boolean minimization tool
- **freehdl** - GPLed free VHDL
- **geda-docs** - Documentation for gEDA
- **geda-examples** - Circuit examples for gEDA
- **geda-gattrib** - Attribute editor for gEDA
- **geda-gnetlist** - Netlister for the gEDA project
- **geda-gschem** - Electronics schematics editor
- **geda-gsymcheck** - Symbol checker for electronics schematics editor
- **geda-symbols** - Electronic symbols for gEDA
- **geda-utils** - Several utilities for the gEDA project
- **gerbv** - Gerber file viewer from the gEDA toolkit
- **gnucap** - The Gnu Circuit Analysis Package
- **gplcver** - An interpreted Verilog HDL simulator
- **gtkwave** - Waveform Viewer
- **irsim** - Switch-level simulator used even for VLSI
- **iverilog** - Icarus Verilog is a verilog compiler and simulator
- **linsmith** - A Smith charting program
- **magic** - A very capable VLSI layout tool
- **mcu8051ide** - IDE for MCS-51 based microcontrollers
- **netgen** - LVS netlist comparison tool for VLSI
- **ngspice** - A mixed level/signal circuit simulator
- **octave-forge** - Contributed functions for octave
- **pcb** - An interactive printed circuit board editor
- **perl-Hardware-Verilog-Parser** - Grammar for parsing Verilog code using perl
- **perl-Hardware-Vhdl-Parser** - Grammar for parsing VHDL code using perl
- **perl-ModelSim-List** - Analyse the 'list' output of the ModelSim simulator
- **perl-Perlilog** - Verilog environment and IP core handling in Perl
- **perl-SystemC-Vregs** - Utility routines used by vregs
- **perl-Verilog** - Verilog parsing routines
- **perl-Verilog-CodeGen** - Verilog code generator
- **perl-Verilog-Readmem** - Parse Verilog $readmemh or $readmemb files
- **qemu** - QEMU is a FAST! processor emulator
- **qucs** - Circuit simulator
- **tkcvs** - TkCVS and TkDiff
- **toped** - VLSI IC Layout Editor
- **trac-peerreview-plugin** - Framework for realtime code review within Trac
- **tkgate** - An event driven digital circuit simulator
- **vhd2vl** - VHDL to Verilog translator
- **vym** - View your mind
- **xcircuit** - Electronic circuit schematic drawing program
- **systemC** – Electronic System Level design and verification with C/C++
- **Eclipse IDE** – Advanced and modular Integrated Development Environment
- **eclipse-veditor-plugin** – Helps digital IC designers/FPGA designers develop Verilog/VHDL code on Eclipse. Provides a realtime error and warnings notification of typos, missing signals, unnecessary signals etc.
- **eclipse-eclox-plugin** – If the vhdl code entails doxygen style comments, a pdf can be autogenerated and used either during internal meetings or sent to the client
- **eclipse-texlipse-plugin** – Since the pdf is generated from latex, the texlipse plugin will provide some additional page layout formatting and easy pdf creation. The pdf creation is now only Ctrl-S, rather than a manual click like one would do on kile. That said, kile was removed from the FEL livedvd

- **eclipse-cdt-plugin** – Provides Embedded C and C++ development tools
- **eclipse-dltk-tcl-plugin** – Tcl scripts can be maintained along side with the HDL code
- **eclipse-epic-plugin** – Perl scripts can be maintained along side with the HDL code
- **eclipse-subclipse-plugin** – Adds Subversion integration to the Eclipse IDE.
- **eclipse-egit-plugin** – Adds distributed version controlled GIT integration to the Eclipse IDE

These included tools can vary between different releases of FEL and it may be more or less of them in upcoming releases. For detailed information visit the website of the FEL-project and read the release notes.

The setup of the EDA environment is not restricted to the listed tools above, and there might be other interesting and useful solutions that can be additionally installed without any problems. It is recommended to use the supported installation of .rpm packages for new software. Moreover, the eclipse community offers a great variety of plugins to adapt the IDE to your needs.

2 Preliminaries

Before the installation process can begin a few requirements have to be met. To create and run the virtual machine you have to download the VirtualBox software from http://www.virtualbox.org/ and install it on the host pc, where the virtual machine shall run on. There you will also find the complete documentation and the binaries for different host platforms (Windows, Linux, Unix, OS X etc.).

To run VirtualBox and a virtual machine your host machine will need:

- A powerful x86 hardware from Intel or AMD. More than one core and built in virtualization support are recommended.
- At least 512 MB of free available RAM for the virtual machine. More than 1024 MB of free available RAM is recommended (optimal >= 2048 MB for the virtual machine).
- Several GB free hard disk space. 10 GB are recommended for this setup.

Instead of using VirtualBox as virtualization software you can also setup the virtual machine with other software like QEMU, KVM, VMWare, VirtualPC, Xen or OpenVZ.

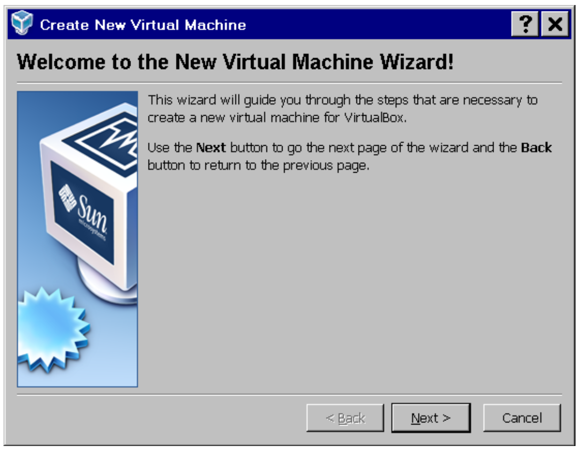
For the installation of the Fedora OS two different solutions exist.

- First, there is the possibility to download the current version of Fedora prepacked with the complete software offering of the Electronic Laboratory form the FEL site as LiveDVD.
- Second, download the blank Fedora .iso from the site of the Fedora projects.

This tutorial uses the second option and explains the stepwise installation procedure of the other components.

3 Setup for the Virtual Machine with VirtualBox

This virtual machine setup was created using VirtualBox version 3.1.6 release 59338.

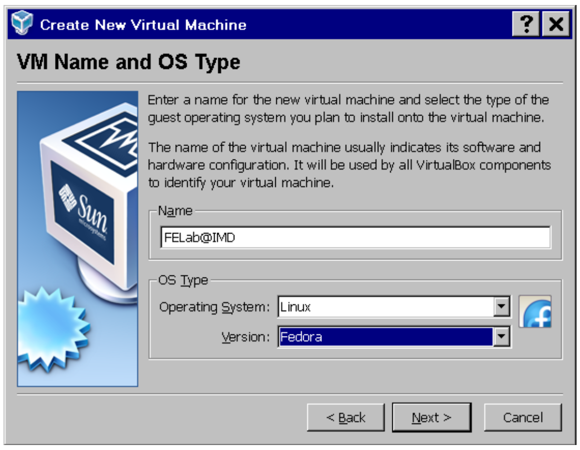

Create New Virtual Machine

Welcome to the New Virtual Machine Wizard!

This wizard will guide you through the steps that are necessary to create a new virtual machine for VirtualBox.

Use the **Next** button to go the next page of the wizard and the **Back** button to return to the previous page.

After the successful installation of VirtualBox a new virtual machine has to be created. First you have to start VirtualBox and then create a new VM.


Create New Virtual Machine

VM Name and OS Type

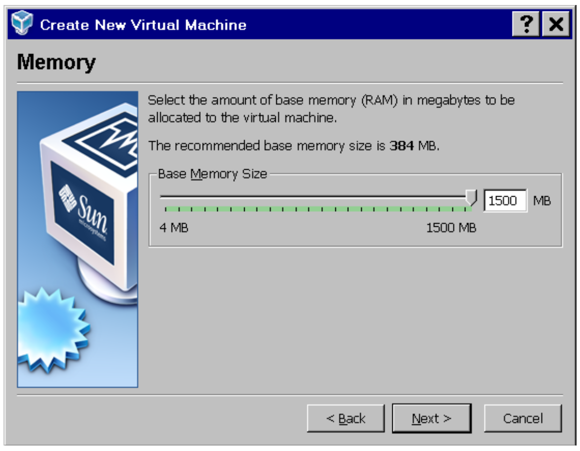
Enter a name for the new virtual machine and select the type of the guest operating system you plan to install onto the virtual machine.

The name of the virtual machine usually indicates its software and hardware configuration. It will be used by all VirtualBox components to identify your virtual machine.

Name: FELab@IMD

OS Type
Operating System: Linux
Version: Fedora

Type in the name of your VM and select the correct OS-type for it.


Create New Virtual Machine

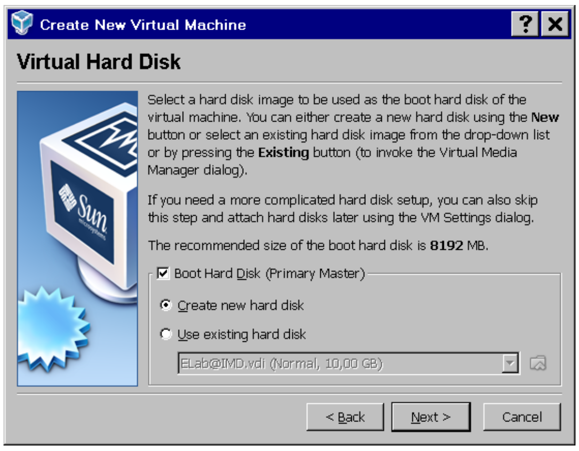
Memory

Select the amount of base memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended base memory size is **384** MB.

Base Memory Size: 1500 MB
4 MB 1500 MB

Then you have to define the size of the available RAM for your VM. More is better. You can change the size of the RAM everytime in VirtualBox when your VM is down.


Create New Virtual Machine

Virtual Hard Disk

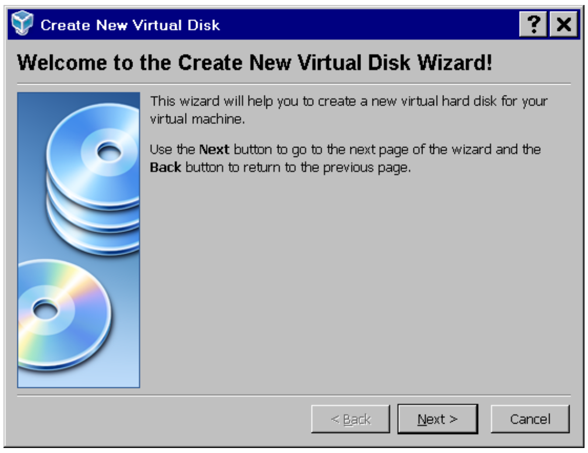
Select a hard disk image to be used as the boot hard disk of the virtual machine. You can either create a new hard disk using the **New** button or select an existing hard disk image from the drop-down list or by pressing the **Existing** button (to invoke the Virtual Media Manager dialog).

If you need a more complicated hard disk setup, you can also skip this step and attach hard disks later using the VM Settings dialog.

The recommended size of the boot hard disk is **8192** MB.

[✓] Boot Hard Disk (Primary Master)
[● Create new hard disk]
○ Use existing hard disk
ELab@IMD.vdi (Normal, 10,00 GB)

The next step contains the creation of a hard disk for your VM.

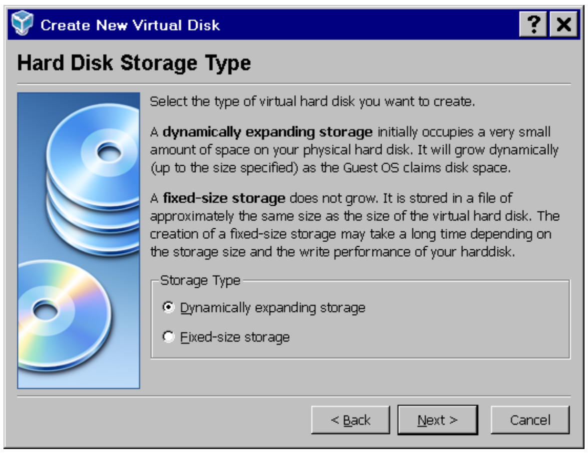


Welcome to the Create New Virtual Disk Wizard!

This wizard will help you to create a new virtual hard disk for your virtual machine.

Use the **Next** button to go to the next page of the wizard and the **Back** button to return to the previous page.

Select the creation of a new hard disk and go on with the installation process.



Hard Disk Storage Type

Select the type of virtual hard disk you want to create.

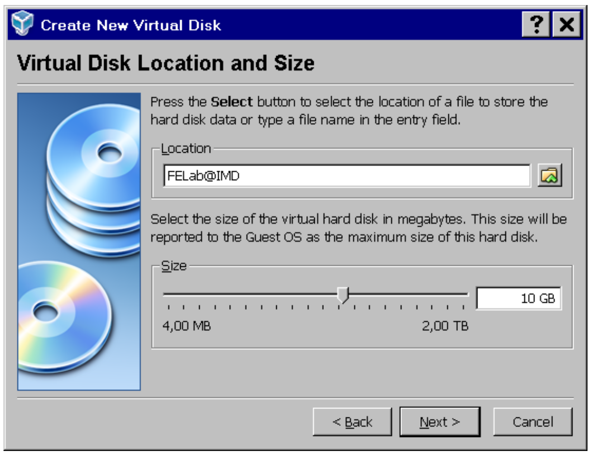
A **dynamically expanding storage** initially occupies a very small amount of space on your physical hard disk. It will grow dynamically (up to the size specified) as the Guest OS claims disk space.

A **fixed-size storage** does not grow. It is stored in a file of approximately the same size as the size of the virtual hard disk. The creation of a fixed-size storage may take a long time depending on the storage size and the write performance of your harddisk.

Storage Type

[✓ Dynamically expanding storage]
Fixed-size storage

Let VirtualBox create a dynamically expanding hard disk for your VM. This has the benefit that the hard disk does not have its maximum size from beginning. Each time you store data on it the hard disk will be expanded until the defined upper border is reached.



Virtual Disk Location and Size

Press the **Select** button to select the location of a file to store the hard disk data or type a file name in the entry field.

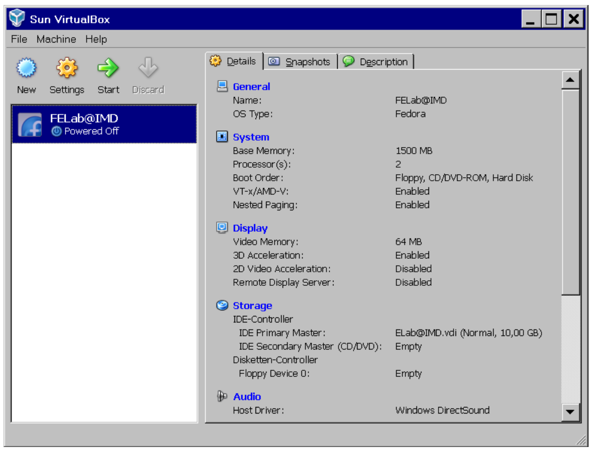
Location: FELab@IMD

Select the size of the virtual hard disk in megabytes. This size will be reported to the Guest OS as the maximum size of this hard disk.

Size: 10 GB

4,00 MB — 2,00 TB

Select a maximum size of 10 GB at least for the installation of the proposed EDA environment. This should be enough space for the installation and additional data or tools.



FELab@IMD
Powered Off

General

```
Name:                              FELab@IMD
OS Type:                           Fedora
```

System

```
Base Memory:                       1500 MB
Processor(s):                      2
Boot Order:                        Floppy, CD/DVD-ROM, Hard Disk
VT-x/AMD-V:                        Enabled
Nested Paging:                     Enabled
```

Display

```
Video Memory:                      64 MB
3D Acceleration:                   Enabled
2D Video Acceleration:             Disabled
Remote Display Server:             Disabled
```

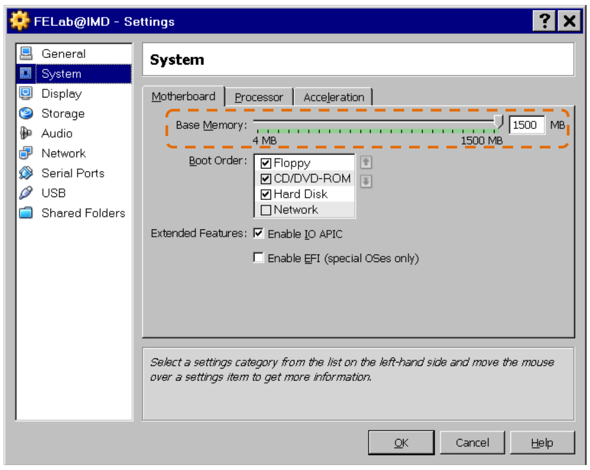
Storage

```
IDE-Controller
  IDE Primary Master:              ELab@IMD.vdi (Normal, 10,00 GB)
  IDE Secondary Master (CD/DVD):   Empty
Disketten-Controller
  Floppy Device 0:                 Empty
```

Audio

```
Host Driver:                       Windows DirectSound
```

When the hard disk is created successfully you have to define the settings of your VM. Here you can set different parameter regarding the performance and the hardware configuration of the VM.


FELab@IMD - Settings

General, System, Display, Storage, Audio, Network, Serial Ports, USB, Shared Folders

System

Motherboard | Processor | Acceleration

Base Memory: 1500 MB (4 MB – 1500 MB)

Boot Order: ☑ Floppy, ☑ CD/DVD-ROM, ☑ Hard Disk, ☐ Network

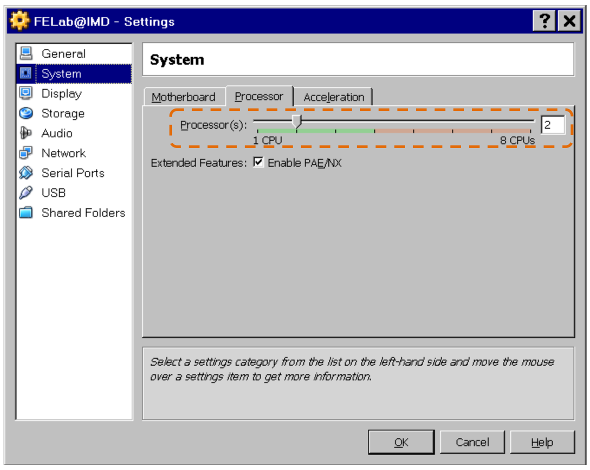
Extended Features: ☑ Enable IO APIC

☐ Enable EFI (special OSes only)

Select a settings category from the list on the left-hand side and move the mouse over a settings item to get more information.

OK | Cancel | Help

If your Processor has multiple cores and built in virtualization support you should activate the IO APIC option of VirtualBox.


FELab@IMD - Settings

General, System, Display, Storage, Audio, Network, Serial Ports, USB, Shared Folders

System

Motherboard | Processor | Acceleration

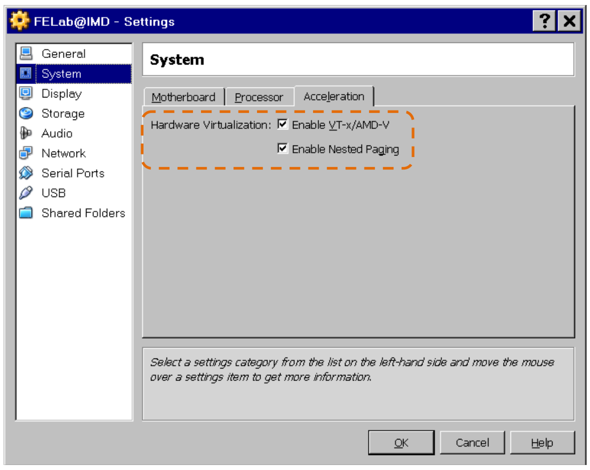
Processor(s): 2 (1 CPU – 8 CPUs)

Extended Features: ☑ Enable PAE/NX

Select a settings category from the list on the left-hand side and move the mouse over a settings item to get more information.

OK | Cancel | Help

You can assign more than one core of the CPU explicit to your VM. This feature will enable more performance for your VM.


FELab@IMD - Settings

General, System, Display, Storage, Audio, Network, Serial Ports, USB, Shared Folders

System

Motherboard | Processor | Acceleration

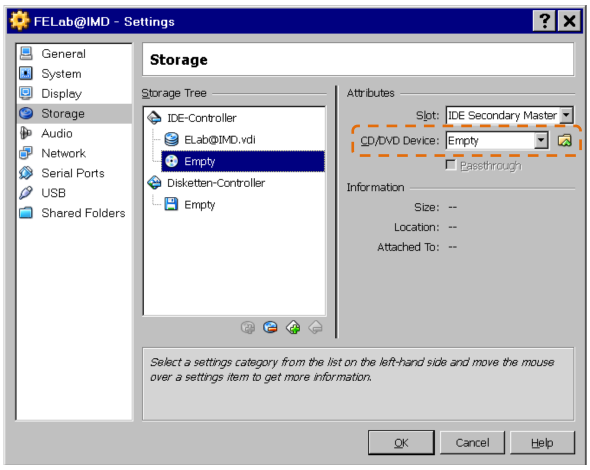
Hardware Virtualization: ☑ Enable VT-x/AMD-V

☑ Enable Nested Paging

Select a settings category from the list on the left-hand side and move the mouse over a settings item to get more information.

OK | Cancel | Help

If hardware virtualization is supported by your CPU then it should be activated.


FELab@IMD - Settings

General, System, Display, Storage, Audio, Network, Serial Ports, USB, Shared Folders

Storage

Storage Tree

- IDE-Controller
  - ELab@IMD.vdi
  - Empty
- Disketten-Controller
  - Empty

Attributes

Slot: IDE Secondary Master

CD/DVD Device: Empty

☐ Passthrough

Information

Size: --

Location: --

Attached To: --

Select a settings category from the list on the left-hand side and move the mouse over a settings item to get more information.

OK | Cancel | Help

The last step contains the assignment of the downloaded Fedora image as CD/DVD device to your VM. On startup of the VM this image will be mounted and the installation procedure of Fedora begins. Hence, you have to point the DVD/CD device of your VM to the Fedora image.

4 Installation of Fedora 12

After the completed setup for the VM you have to start this virtual machine and initiate the installation of Fedora 12 from the mounted ISO-image. This installation process is self-explaining and well guided so it is not part of this tutorial.

When the installation of Fedora is done and the VM is already restarted (don't forget to free the CD/DVD-drive from the Fedora ISO-image) a few software packages have to be installed for the correct setup of the EDA environment and the installation of the VBox-GuestAdditions. So open a terminal and login as root.

```
[ user@ELab ~]$  su --login
Password:
[ root@ELab ~]#
```

First, you have to install packages for the linux-kernel and kernel-header files, because the GuestAdditions of VirtualBox will recompile the kernel with their own integrated modules. The installation of the GuestAdditions will benefit in better performance and handling of your VM. So it is recommended to make it. Now install the missing packages with:

```
[ root@ELab ~]#  yum –y install kernel-devel kernel-headers dkms
```

Furthermore, you need to install the needed packages for software development under Fedora like compiler, libraries and make tools. Fedora does not come with it by default and you must type the following command into your terminal:

```
[ root@ELab ~]#  yum –y groupinstall ‘Development Tools’
```

This command installs a bundle of tools to setup a complete development environment including the GNU compiler toolchain, standard libraries, autotools and many more.

Before installing the VBox-GuestAdditions you have to mount the ISO-image that contains the needed files. This can be done in two different ways (don't forget to free the CD/DVD-drive of the Fedora image before!):

- Type CTRL-RIGHT + D and the GuestAdditions image will be mounted to the CD/DVD-drive
- Go to the window menu of the VM and click Devices->Install Guest Additions … and the GuestAdditions image will be mounted to the CD/DVD-drive

Now the ISO-image with the guest additions will be mounted and you can proceed. Go to the media and get a list of the contents on the GuestAdditions-image. Before you are able to install the guest additions you have to set the correct path for the kernel libs (KERN_DIR=…). Afterwards, start the correct install script with ./Script.run !

Change the directory to the ISO-image

```
[ root@ELab ~]#  cd /media/VBOX_ADDITIONS_3.1.6_59338
```

List the content of the ISO-image

```
[root@ELab VBOXADDITIONS_3.1.6_59338]# ls

32Bit   VBoxLinuxAdditions-amd64.run    VBoxWindowsAdditions.exe
64Bit   VBoxLinuxAdditions-x86.run      VBoxWindowsAdditions-x86.exe
AUTORUN.INF  VBoxSolarisAdditions.pkg autorun.sh   VBoxWindowsAdditions-
amd64.exe
```

Set the correct path to the kernel files of Fedora

```
[root@ELab VBOXADDITIONS_3.1.6_59338]# export
KERN_DIR=/usr/src/kernels/2.6.32.11-99.fc12.i686
```

Run the installation script of the guest additions

```
[root@ELab VBOXADDITIONS_3.1.6_59338]# ./VBoxLinuxAdditions-x86.run
```

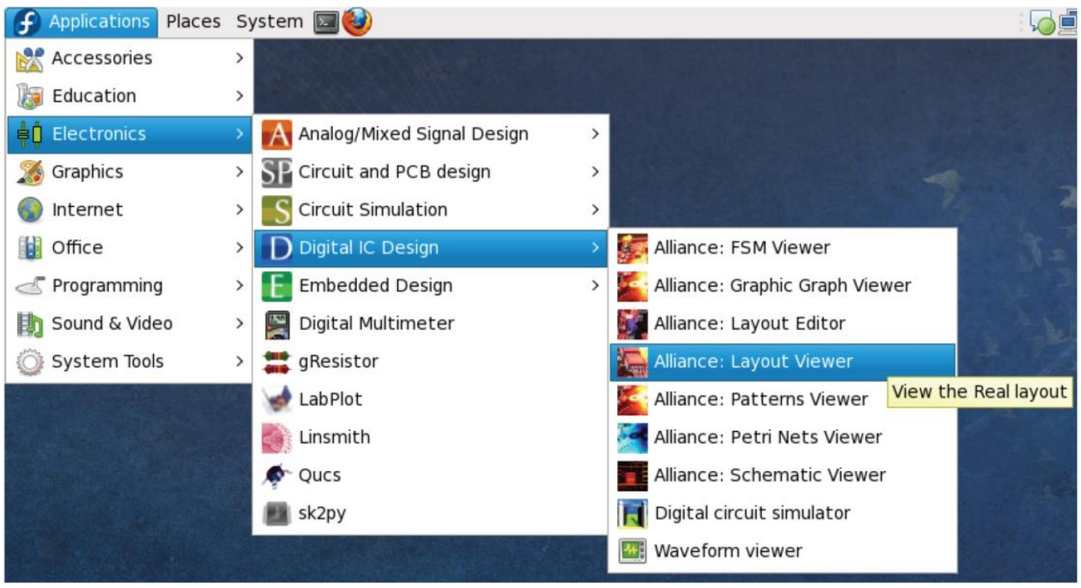
After the installation process of the guest additions has finished you will need to restart the virtual machine. Now the VM is ready for the installation of the VLSI tools and the SystemC resources.

5 Installation of VLSI and EDA tools

To install the “Electronic Laboratory” package you must simply type:

```
[ root@ELab ~]# yum –y groupinstall ‘Electronic Lab’
```

This installation process may take a while before it has finished and afterwards your applications menu should look like this screenshot below:



6 Installation of SystemC

Before the SystemC installation can begin you have to download the source files from the website of the Open SystemC Initiative (OSCI). Therefore, you have to sign up for a free member account. There you can get all packages of the current OSCI standards like:

- Core SystemC language libraries + test pakages
- SystemC AMS extensions for analog/mixed signal simulations
- TLM: The Transaction-Level Modeling library
- SCV: The SystemC Verification library

Please download the SystemC (here systemc-2.2.0) and the TLM (here TLM-2.0.1) packages to proceed with this tutorial. The next step is to unpack the .tgz files to a temporary folder (*like download/path/to/systemc e.g. /home/user/downloads/systemc-2.2.0*) and to edit a few sources for the successful installation of SystemC. For this tutorial the following tools and libraries were used (included in the groupinstall 'Development Tools'):

- g++ version 4.4.3 20100127
- gcc version 4.4.3 20100127
- GNU automake version 4.4.3 20100127
- GNU Make version 3.81

First, after unpacking the SystemC resources you have to navigate to the containing folder and must edit a few lines in */home/user/downloads/systemc-2.2.0/src/sysc/utils/sc_utils_ids.cpp.* Open this file with an editor and add the following lines:

Go to the position of:

line 61: #include "sysc/utils/sc_report.h"

Now include the lines below:

line 62: #include <cstdlib>
line 63: #include <string.h>
line 64: using namespace std;

These changes will prevent you to get a compilation error for gcc versions higher than 4.x.x and was proven for this setup.

Now open a terminal, login as root and navigate to the folder containing the unpacked SystemC resources. There you have to create the temporary objdir directory for the SystemC installation process, set the CXX and CC variables for the compiler search of the configuration script, create the folder structure in the SystemC destination path and start the configuration script. Afterwards, you can start the make process and test the generated installation of SystemC. Please see the code listing below.

Change directory to the unpacked SystemC resources and create the objdir. Afterwards, set the CXX and the CC path for your compiler.

```
[root@ELab ~]# cd /home/user/downloads/systemc-2.2.0
[root@ELab systemc-2.2.0]# mkdir objdir
[root@ELab systemc-2.2.0]# export CXX=g++
[root@ELab systemc-2.2.0]# export CC=gcc
```

The folder creation lines in the configuration script seem to be a little bit buggy so it is better to create them by yourself before. SystemC gets an own path to enable the installation of different versions in parallel.

```
[root@ELab systemc-2.2.0]# mkdir /usr/local/systemc-2.2.0
[root@ELab systemc-2.2.0]# mkdir /usr/local/systemc-2.2.0/include
[root@ELab systemc-2.2.0]# mkdir /usr/local/systemc-2.2.0/lib-linux
[root@ELab systemc-2.2.0]# mkdir /usr/local/systemc-2.2.0/doc
[root@ELab systemc-2.2.0]# mkdir /usr/local/systemc-2.2.0/examples
```

Change the current destination to the created objdir and run the configuration script. You have to call the configuration script with the argument of the path where SystemC shall be installed.

```
[root@ELab systemc-2.2.0]# cd objdir
[root@ELab objdir]# ../configure --prefix=/usr/local/systemc-2.2.0
```

After the configuration script has finished its work you have to call the make process.

```
[root@ELab objdir]# make
[root@ELab objdir]# make install
```

This may take a while and you can check the installed files/libraries with the following command:

```
[root@ELab objdir]# make check
```

You must change the rights for the SystemC resources.

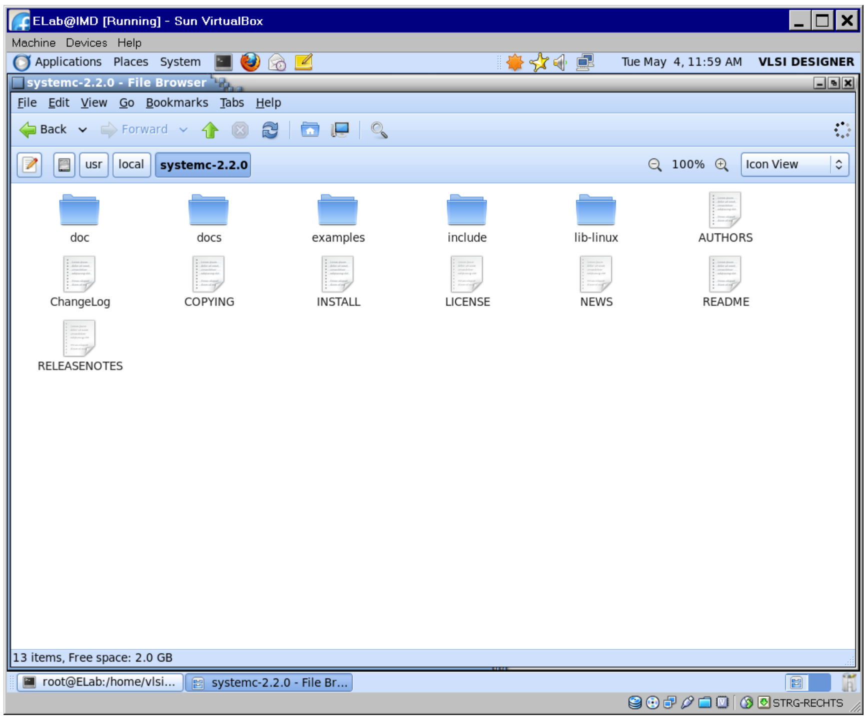
```
[root@ELab objdir]# chmod –R a+x /usr/local/systemc-2.2.0
```

Now SystemC is installed on your system and you need to clean up the generated objdir with:

```
[root@ELab objdir]# rm –rf *
```

Done !!!

The folder structure of your SystemC installation in */usr/local/systemc-2.2.0* should look the same like demonstrated in the screenshot below.


ELab@IMD [Running] - Sun VirtualBox
Machine Devices Help
Applications Places System
Tue May 4, 11:59 AM VLSI DESIGNER
systemc-2.2.0 - File Browser
File Edit View Go Bookmarks Tabs Help
Back
Forward
usr local systemc-2.2.0
100%
Icon View
doc
docs
examples
include
lib-linux
AUTHORS
ChangeLog
COPYING
INSTALL
LICENSE
NEWS
README
RELEASENOTES
13 items, Free space: 2.0 GB
root@ELab:/home/vlsi...
systemc-2.2.0 - File Br...
STRG-RECHTS

The installation process of the TLM libraries is quite easy, because you just need to copy the unpacked folder with the TLM resources to a destination of your choice. It is recommended to follow the same way like mentioned in the SystemC installation process.

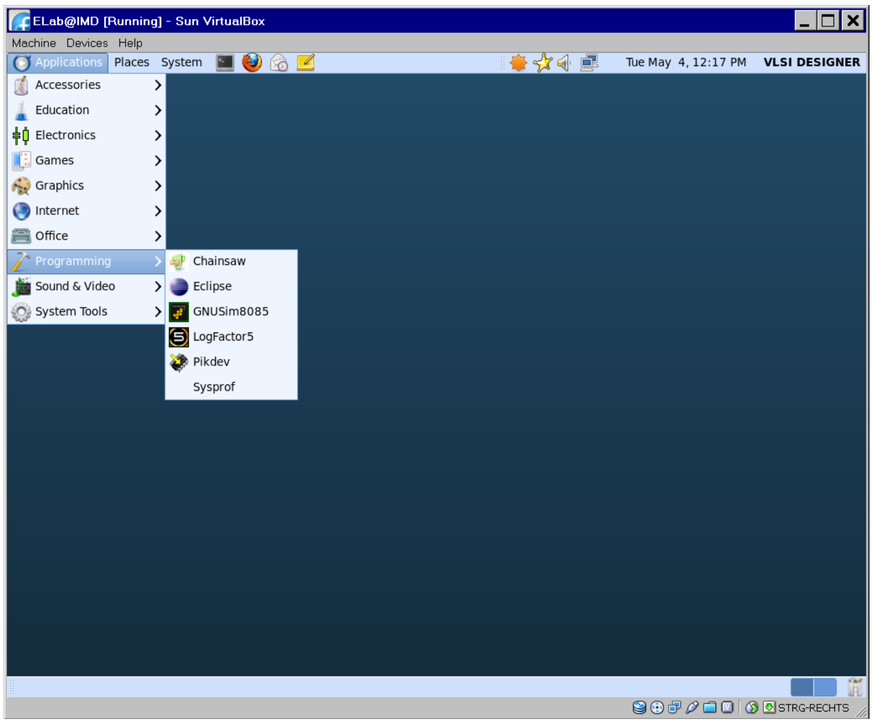
Simply copy the unpacked TLM folder to /usr/local and change the rights.

```
[root@ELab ~]# mv /home/user/downloads/TLM-2.0.1  /usr/local
[root@ELab ~]# cd /usr/local/
[root@ELab local]# ls

bin    include  OSCI_SystemC_AMS_extensions_1v0_Standard  src
etc    lib      sbin                                      systemc-2.2.0
games  libexec  share                                     TLM-2.0.1

[root@ELab local]# chmod -R a+x /TLM-2.0.1
```

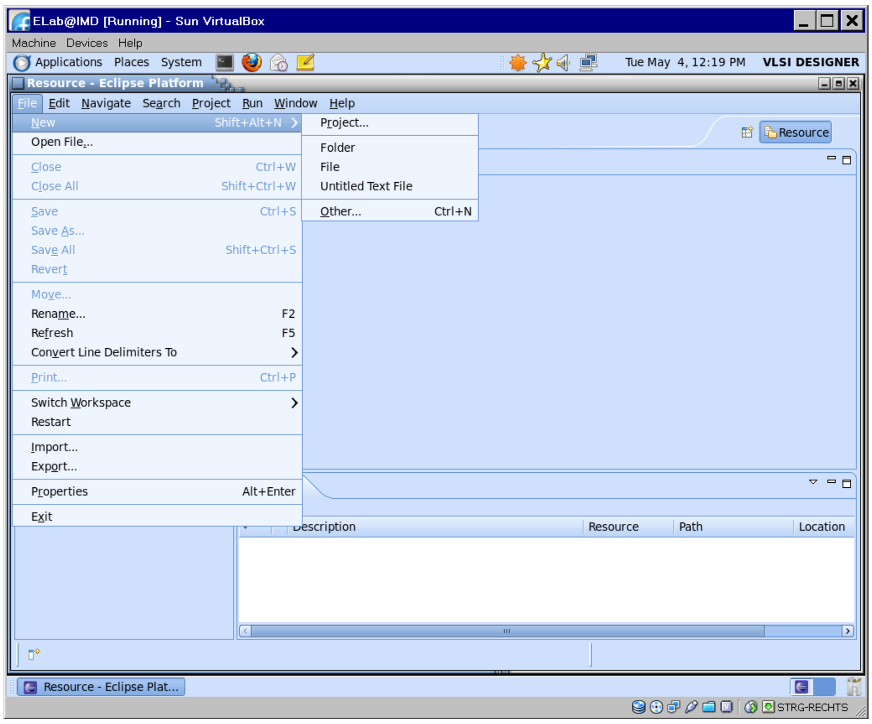
Done !!!

Finally, all installation processes for the SystemC sources are done and to use these sources in your projects you have to add the path information to the Eclipse IDE. The SystemC Verification library is not treated by this setup tutorial, because there were heavy problems during the installation and it failed every time we tried.

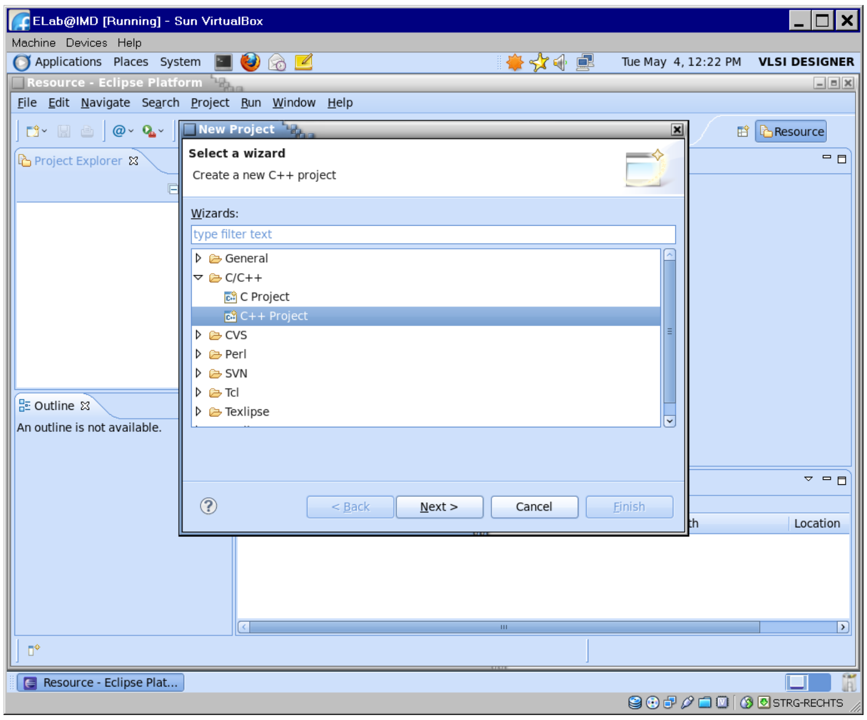
Now you have to start the Eclipse IDE from the applications panel like shown below.



After it has opened create a new project (File -> New -> Project…) inside the Eclipse IDE.



Select the project type as new “C++ Project” and proceed by clicking the next button.


ELab@IMD [Running] - Sun VirtualBox
Machine Devices Help
Applications Places System
Tue May 4, 12:22 PM VLSI DESIGNER
Resource - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Project Explorer
Outline
An outline is not available.
Resource
Location
< Back
Next >
Cancel
Finish
Resource - Eclipse Plat...
STRG-RECHTS

New Project

Select a wizard

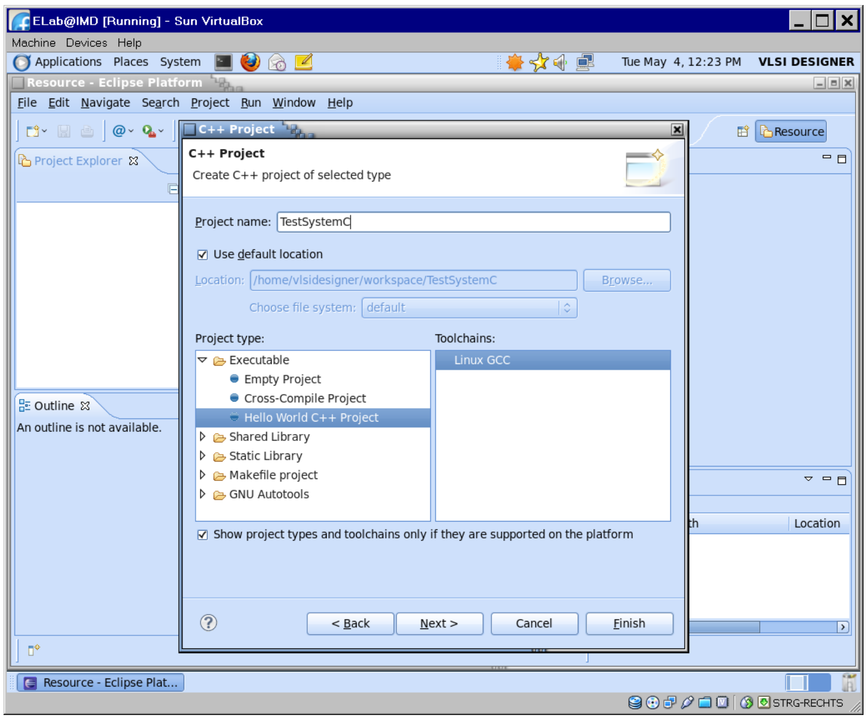
Create a new C++ project

Wizards:

type filter text

- General
- C/C++
  - C Project
  - C++ Project
- CVS
- Perl
- SVN
- Tcl
- Texlipse

Create a simple exemplary project by selecting the “Hello World C++ Project” and name it TestSystemC. Afterwards, finish this setup and the project will be generated by Eclipse.


ELab@IMD [Running] - Sun VirtualBox
Machine Devices Help
Applications Places System
Tue May 4, 12:23 PM VLSI DESIGNER
Resource - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Project Explorer
Outline
An outline is not available.
Resource
Location
< Back
Next >
Cancel
Finish
Resource - Eclipse Plat...
STRG-RECHTS

C++ Project

C++ Project

Create C++ project of selected type

Project name: TestSystemC

☑ Use default location

Location: /home/vlsidesigner/workspace/TestSystemC Browse...

Choose file system: default

Project type:

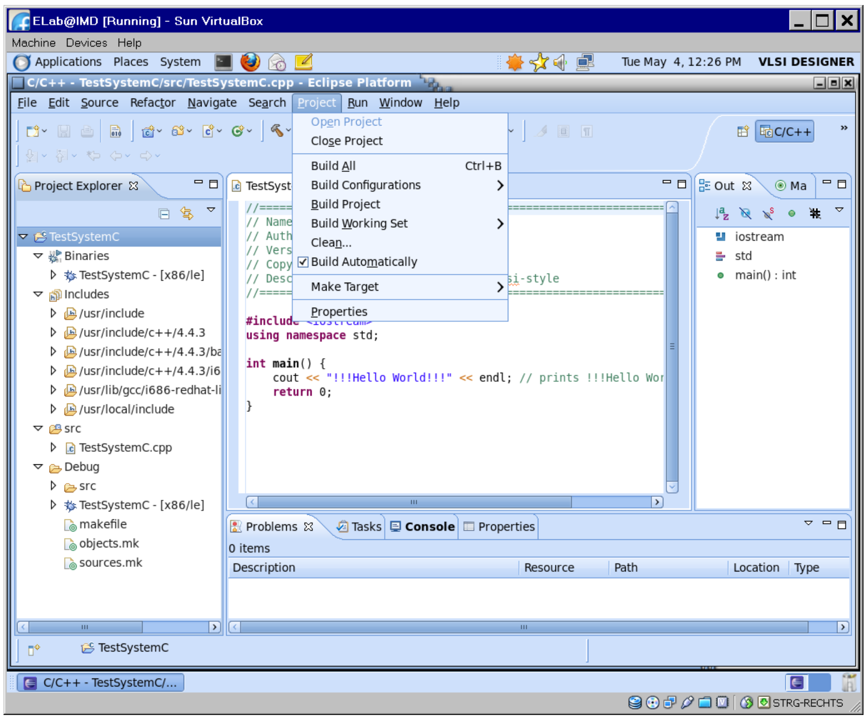
- Executable
  - Empty Project
  - Cross-Compile Project
  - Hello World C++ Project
- Shared Library
- Static Library
- Makefile project
- GNU Autotools

Toolchains:

Linux GCC

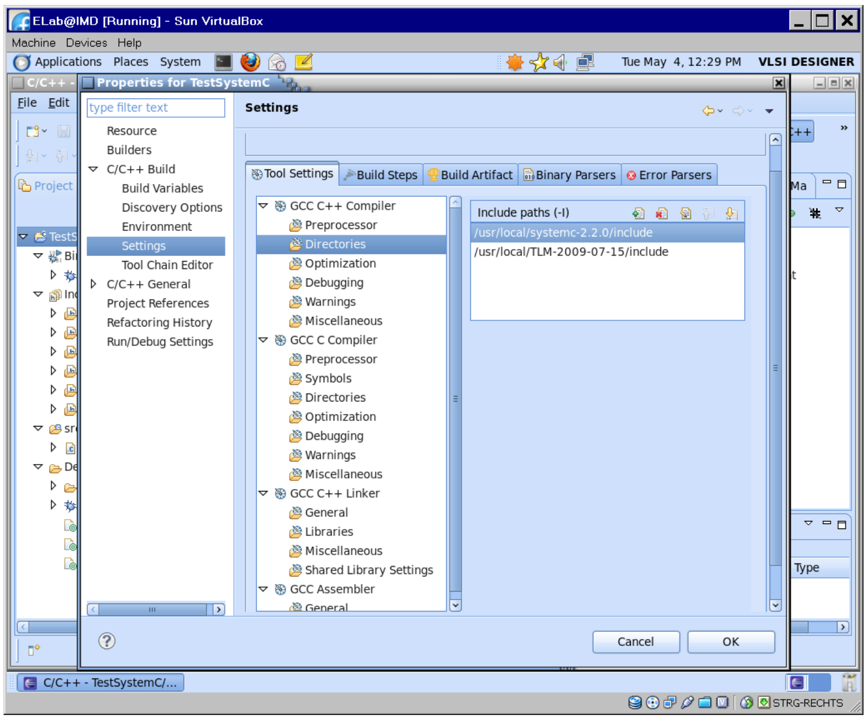
☑ Show project types and toolchains only if they are supported on the platform

Now you have to open the properties dialog for this project (menu: Project -> Properties).

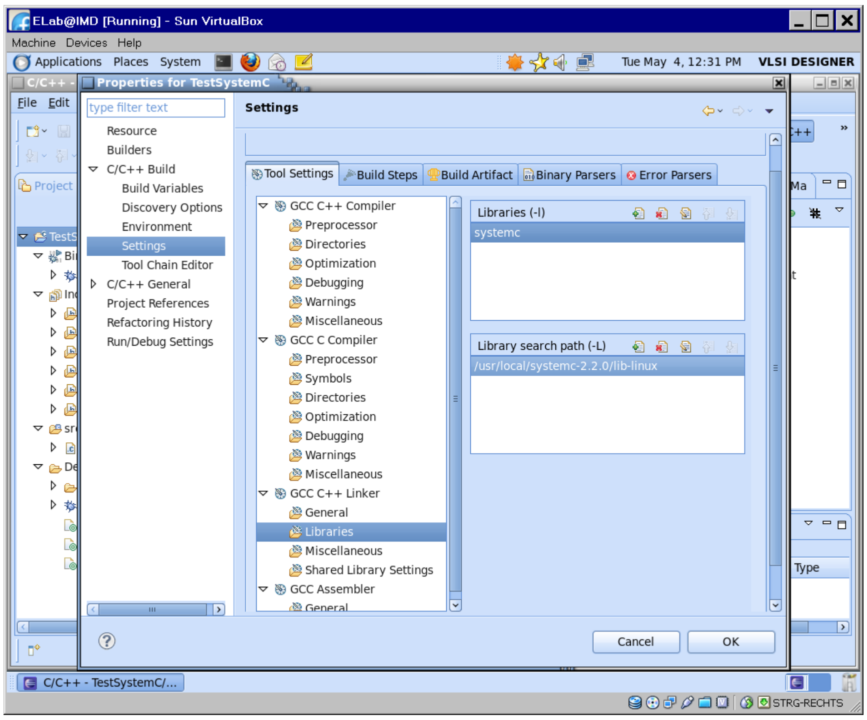


```
//=====
// Name
// Auth
// Vers
// Copy
// Desc
//=====

#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl; // prints !!!Hello Wor
    return 0;
}
```

Finally, edit the Directories and Libraries options for the C++ compiler and C++ linker by adding the path information of the SystemC and SystemC-TLM installations.



Include paths (-I)
/usr/local/systemc-2.2.0/include
/usr/local/TLM-2009-07-15/include



Libraries (-l)
systemc

Library search path (-L)
/usr/local/systemc-2.2.0/lib-linux

Now the Eclipse IDE and the created project will be ready for coding with SystemC. You can write, compile and debug some examples to test it. Have fun !

7 Links and additional Informations

A good alternative to this EDA environment might be SCLive CD if your focus relies on coding with VHDL, Verilog and SystemC without the need for a complete EDA workflow. The current SCLive release version 3.0 you can download from http://sclive.wordpress.com/ and there you will find a tutorial for its integration with QEMU.

The website http://www.eda.org/ provides a good overview and an extensive database for the major standards of the EDA industry. Furthermore, you will find links and information about free EDA tools.

The greatest database about opensource hardware and software tools in the EDA domain you can find on the website of the OpenCollector.org community.

If you are interested in opensource hardware the community of OpenCores.org offers an extensive database of IP Cores developed with SystemC, VHDL and Verilog.