Guide for getting started with SystemC development

By Senior Consultant Kim Bjerge (kim.bjerge@teknologisk.dk)

Contents

Preface

SystemC is a language for modelling systems that contains software and hardware components. Typically, today's systems contain application-specific hardware and software. Hardware and software are usually co-developed on a tight schedule, the systems have tight real-time performance constraints and thorough functional verification is required to avoid expensive and sometimes catastrophic failures. SystemC contains a co-design method whereby the system can be modelled together in an early phase of the development. It can be performed at a high level of abstraction and enable the design team to get a fundamental understanding at an early stage of the design process.

SystemC can be used by developers to elaborate an architectural model of an embedded system early in the development phase. It is possible to simulate the hardware on a higher level with a focus on functionality and communication compared to the hardware description languages as VHDL and Verilog. They are on the "Register Transfer Level" (RTL) focussing on registers and wires. The largest gain in system-level design can be achieved in the specification phase, by rapid trade-off analyses and functional experiments.

SystemC basis behaviour can be defined only once, however, more details can be added or behaviour can be refined during the design process all the way down to the RTL level. The embedded software design and programming can start in an earlier development phase based on the SystemC model of the hardware.

SystemC can be use to model a system on different abstraction levels. The SystemC model consists of communicating processes and in this respect it is the same as many other modelling languages, including the hardware description languages as VHDL and Verilog. Like the two languages, processes and interconnection between them may be mixed. SystemC allows models of computation that covers a mixture of software, hardware and digital electronics to be mixed in the same model. The model can be refined for the hardware down to the RTL level and at present it is possible to make a synthesis of the SystemC code for FPGA development.

SystemC requires basic knowledge of C++ and weeks of study in order to be productive. People with a background in embedded software, FPGA or System-on-Chip (SoC) development will be target users of the SystemC modelling language.

Getting started with SystemC development

SystemC is a C++ class library developed by the Open SystemC Initiative (OSCI) which is an independent, non-for-profit association dedicated to defining an advanced open industry standard for system-level modelling, design and verification. The source code for the SystemC library can be downloaded free of charge from the OSCI website. The code contains a number of implementation examples for areas such as: signal processing (FFT, FIR), data communication (Package switch), CPU simulation (RISC), hardware fifo's and bus topology.

This document covers how to get started with the tools required for SystemC development. It covers the setup and required software for developing on Windows XP or Linux and describes how to set up the environment for working with SystemC. Eclipse can be used as the editor and development tool on Windows and Linux and is an open source development platform. For Windows XP this document also describes how to work with the Microsoft Visual Studio .NET 2005.

The document also contains a description of a waveform viewer which can be used to display wave files in the "Value Change Dump" (VCD) format. It can be generated by SystemC code.

The source files for the example (ADCInput.zip) used in this document can be downloaded from the website "Teknologisk Institut". http://www.teknologisk.dk/gswsystemc

The last part of the document contains books recommended for getting started where you can find SystemC examples with reference to where you can download the example of source code.

Windows setup for SystemC

This part contains a description of the tools to be installed for developing on Windows. It contains a chapter for setup of Eclipse together with cygwin and SystemC. The next chapter contains the setup for Microsoft Visual Studio .NET applications.

Setup for Eclipse applications

1. Install the cygwin environment from http://www.cygwin.com/ and use the following instructions:

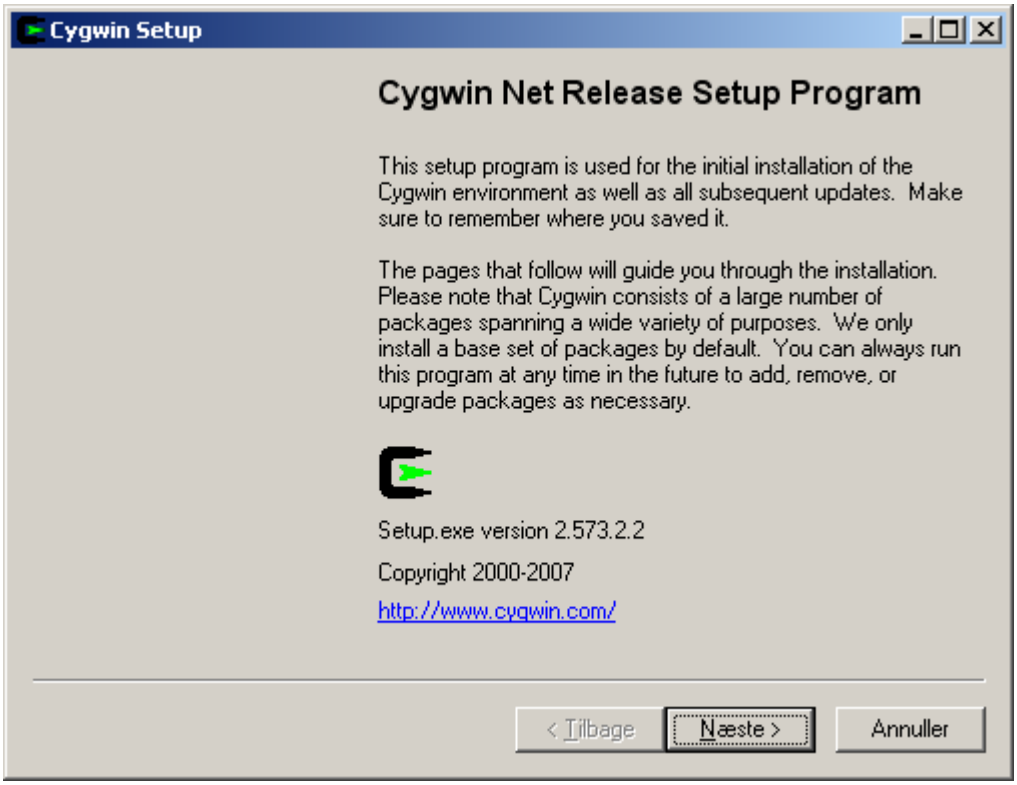

Figure 1 Cygwin setup

2. Choose Install from Internet or from Local Directory in case you already have a cygwin installation on DVD or a network drive.

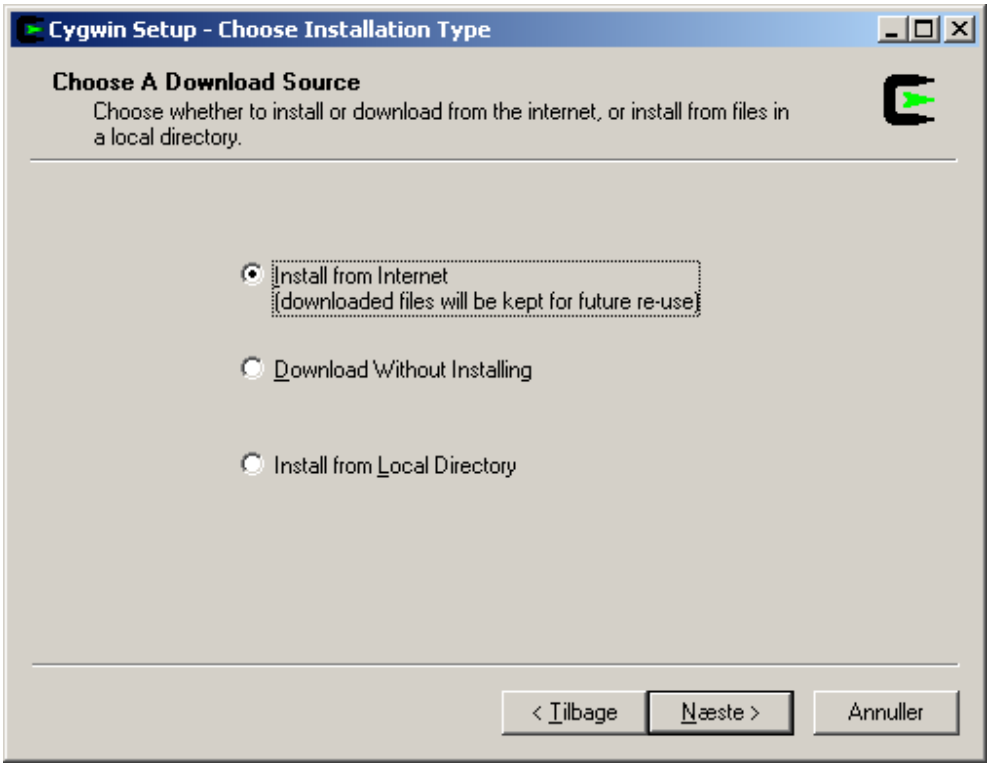

Figure 2 Cygwin installation type

3. In the next window it is important to select 'Install For' All Users and set the 'Default Text File Type' to '**DOS/text'**. The DOS/text setting is very important for integration with Xilinx EDK.

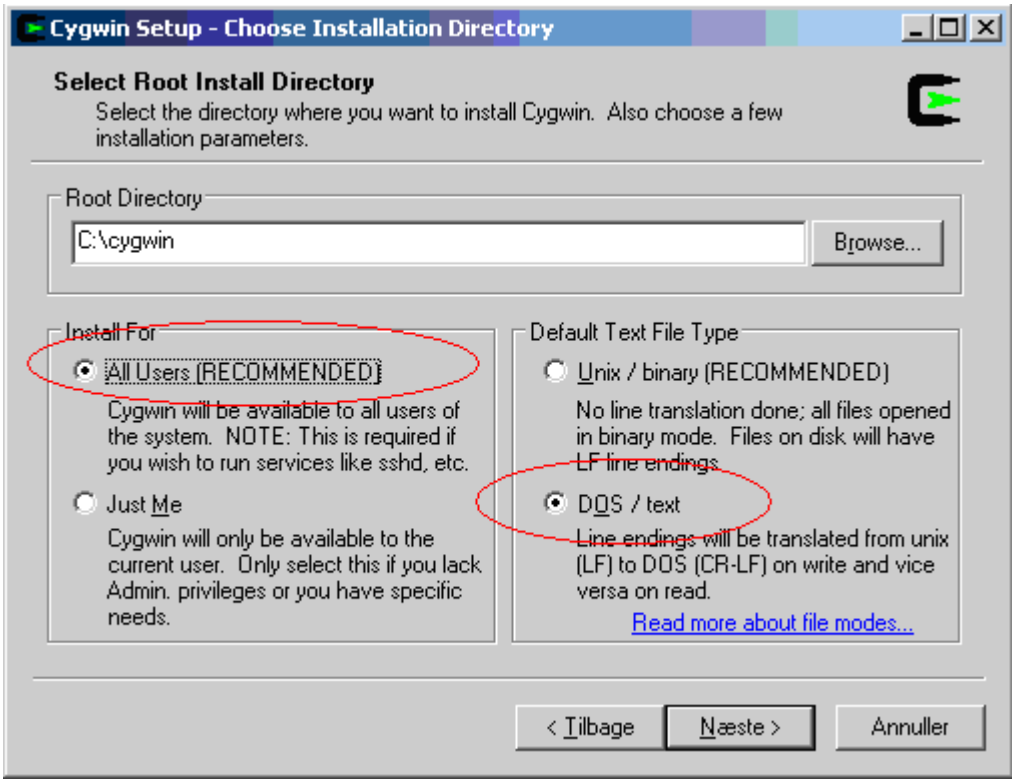

Figure 3 Cygwin installation directory

4. The next step is to specify where the installation should be stored when downloading from the Internet of if a local installation is used to find the installation.

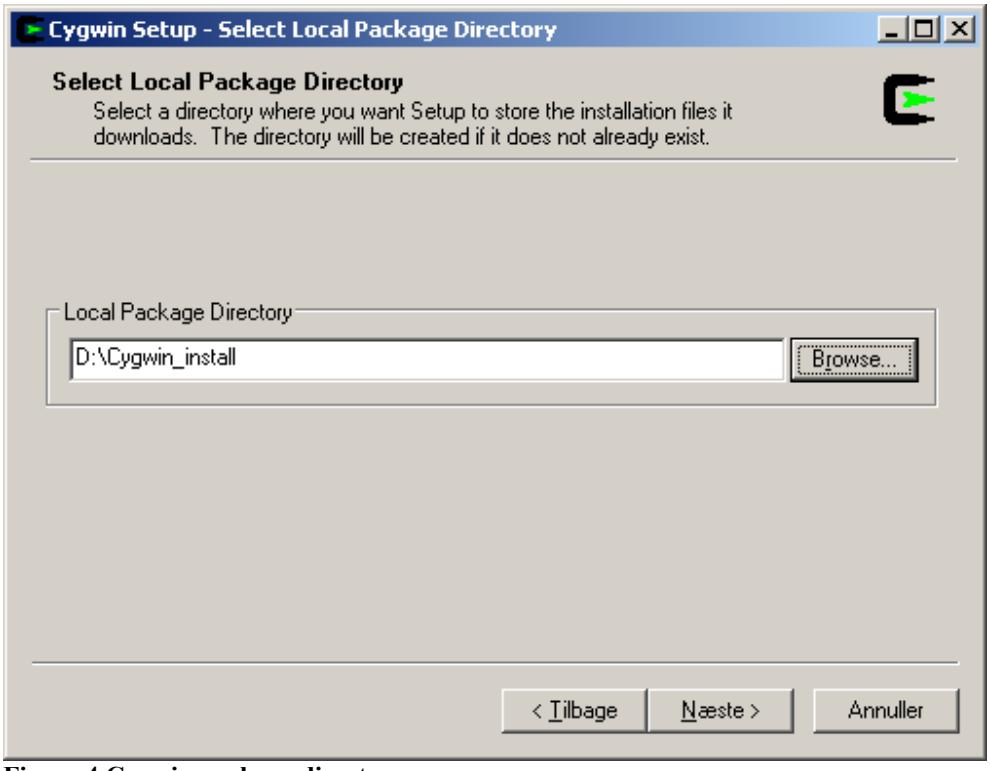

Figure 4 Cygwin package directory

5. If you choose Internet then select 'Direct connection' or 'Use IE5 Settings' and choose a mirror site. Possible mirror sites are to be found here: http://cygwin.com/mirrors.html.

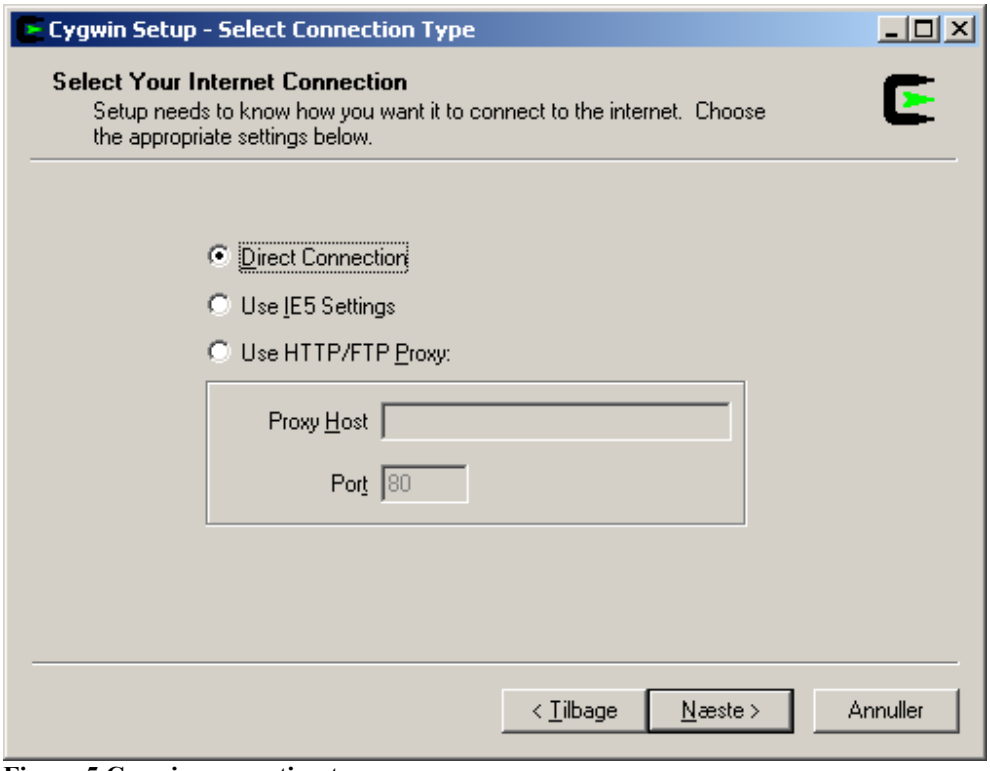

Figure 5 Cygwin connection type

6. The next page shows a list of categories, each containing a specific group of packages to install. Select **Base, Devel, Editors and Libs**[1]**,** install by clicking on the ↻ icon until the word Install appears to the right of the icon.

The installation will start when clicking 'next' and takes about 30 minutes.

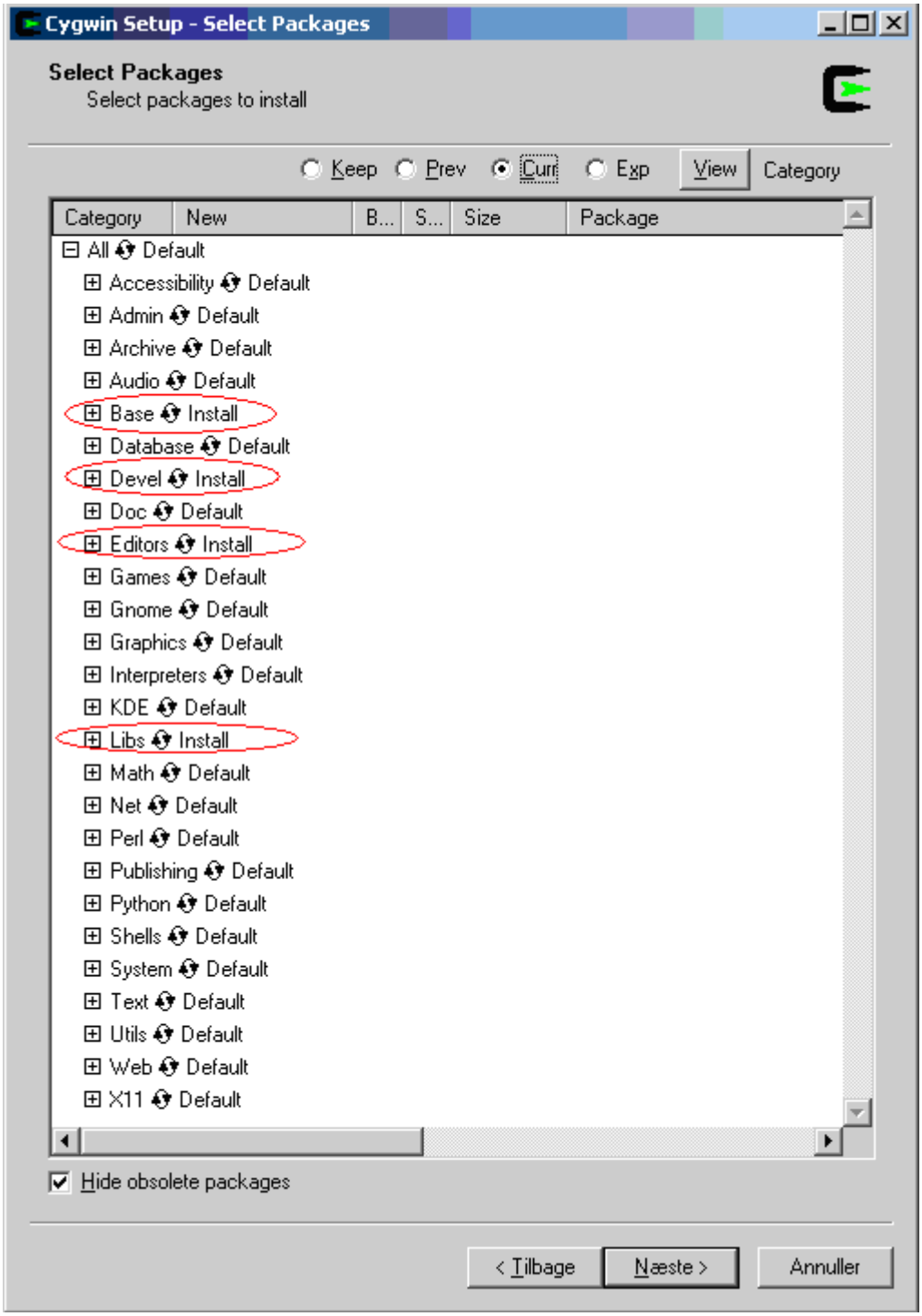

Figure 6 Cygwing package selection

[1] We have seen situations where the gtk2 lib has to be skipped in the installation to prevent the installation from getting stocked. Select "↻ Skip" for all the gtk2-x11 libs in the Devel and Libs packages.

7. When the installation is completed, press finish.

You can now add the 'C:\Cygwin\bin' to the path environment variable or copy the cygwin1.dll file (Found in 'C:\cygwin\bin') to the 'C: \Windows\System32' directory. This is required in order to run programmes compiled and linked by cygwin.

Installing SystemC with cygwin

This guide concerns installation of version 2.2.0 for SystemC.

SystemC can be downloaded from the website of (OSCI) Open SystemC Initiative, see link http://www.systemc.org/home.

The SystemC source is to be found on the website: **DOWNLOADS -> OSCI Standards -> SystemC 2.2**. Before being able to download this source code, you have to create an account.

Additional information is to be found in the INSTALL and README files located in the SystemC package. This guide should, however, be sufficient to get started.

If you type **'$ uname –a**' in your cygwin bash shell you can see the version of cygwin. In this installation we have used the following version:

CYGWIN_NT-5.1 5cvgb2j 1.5.24(0.156/4/2) 2007-01-31 10:57 i686 Cygwin

We have found that the default thread system (qt) used by SystemC when following the instructions in the INSTALL file will not work with the above version of Cygwin. You have to build the pthreads version as described below. Furthermore, you need to modify the systemc.h file as the standard libraries std::wctomb and std::wcstombs are not supported by Cygwin. We have found some older versions of Cygwin where these errors do not appear. If you already have an old version of Cygwin installed, modification of the systemc.h file may not be required.

Before installing the SystemC, you must have completed the cygwin installation described in the previous chapter:

1. Unzip the systemc-2[1].2.0.tgz to the folder:

C:\systemc-2.2.0

2. Modify the C:\systemc-2.2.0\lib-cygwin\src**systemc.h** file as described below:

Line 175 – 177 looks like:

```
using std::wctomb;
using std::mbstowcs;
using std::wcstombs;
```

Must be modified to:

```
#if !defined(__CYGWIN__)
    using std::wctomb;
    using std::wcstombs;
#endif
    using std::mbstowcs;
```

3. Start the Cygwin bash and create the objdir (mkdir) and change to this directory

/cygdrive/c/systemc-2.2.0/objdir

4. Configure the package for your system.

$ ../configure

5. Compile the package.

For a debug SystemC library, enter:

$ make pthreads_debug

Alternativly for an optimized SystemC library, enter: **$ make pthreads**

6. Install the package.

$ make install

The SystemC library will be stored in :

Cygwin bash shell:
/cygdrive/c/ systemc-2.2.0/lib-cygwin/libsystemc.a

Windows Explorer:
C:\systemc-2.2.0\lib-cygwin\libsystemc.a

7. At this point you may wish to verify the installation by testing the example suite.

$ make check

This will compile and run the SystemC examples in the subdirectory examples.

Creating SystemC Applications with Eclipse

You can download the Eclipse IDE for C/C++ Developers from http://www.eclipse.org/downloads/. The downloaded file must be unzipped and copied to the root directory like: 'C:\Eclipse'. This description is made for version 3.3.0, (Build id: I20070621-1340).

Start the eclipse.exe file and select a work space for your project like C:\Projects\TestSCEclipse and select 'Go to the workbench'.

To create a SystemC project you can create a simple C++ "Hello World" project to check if the cygwin compiler and debugger is installed correctly. Follow the steps described below.

1. Select File -> New -> **C++ Project** and enter a name for your project like TestSC and select in Project Types -> Executable -> **Hello World C++ Project**

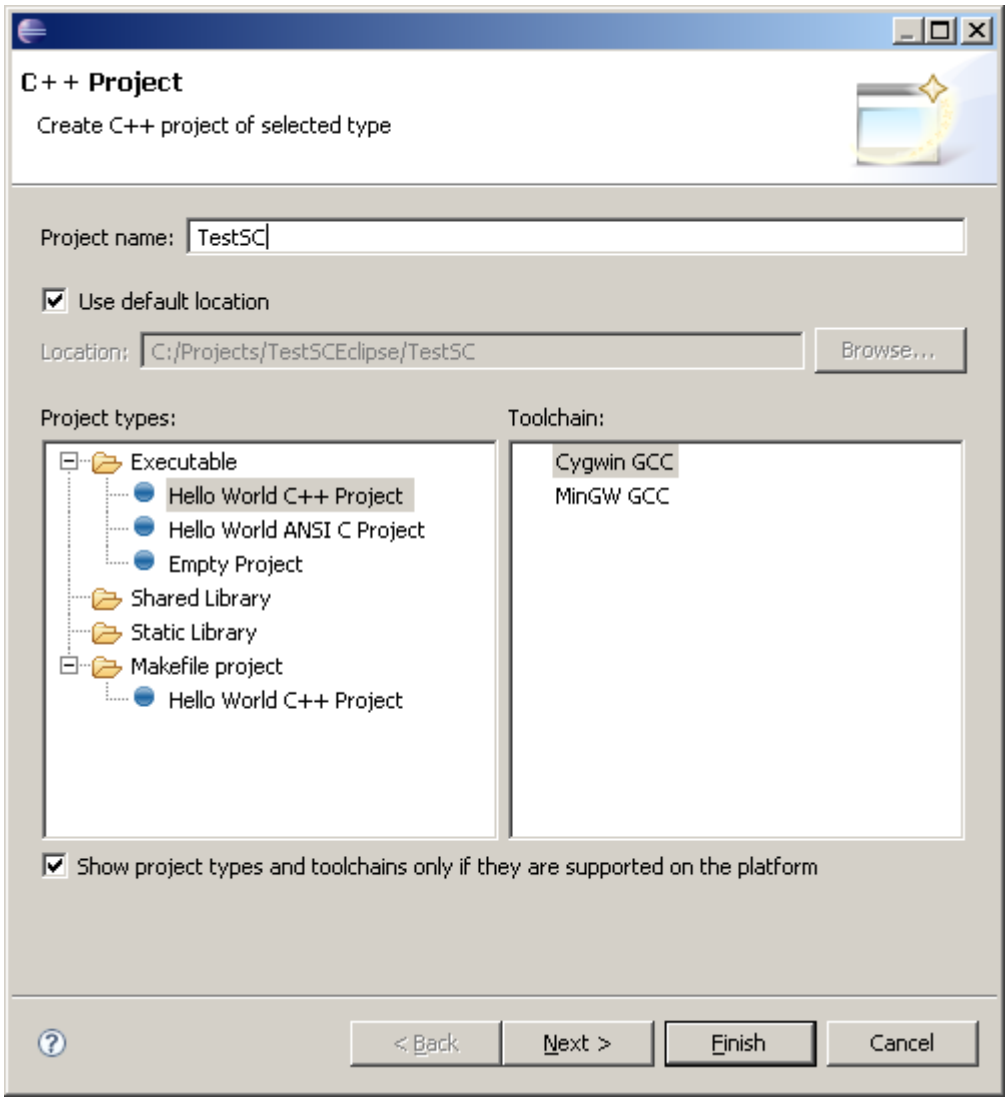

Figure 7 Eclipse create project

2. Press Next and enter your Author name

3. Press next and accept the Debug and Release configurations in the window that appears.

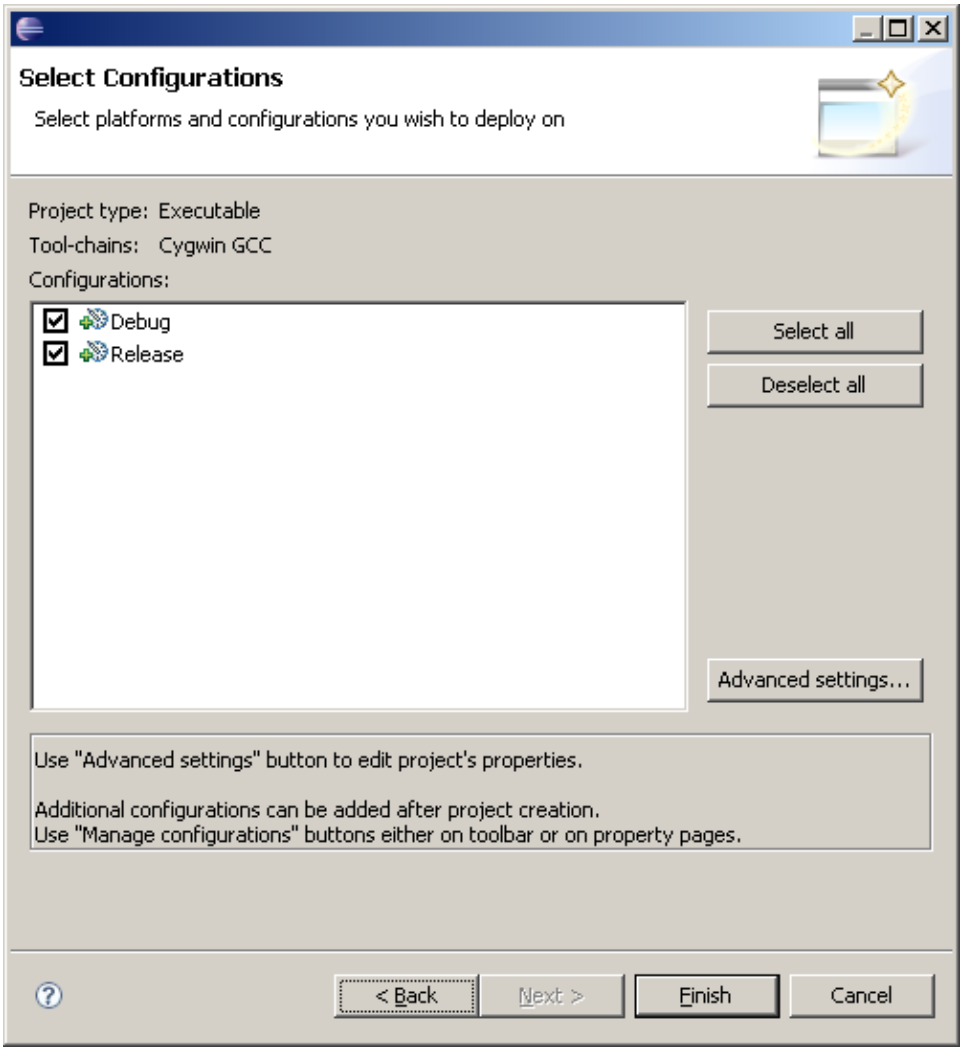

Figure 8 Eclipse configurations

4. For setting the include path for SystemC select the '**Advanced settings**' and add the include path '**C:\systemc-2.2.0\include**' in C/C++ Build -> Settings -> Tool settings -> Cygwin C++ Compiler -> Directories.

The "**–I** " directive indicates for the cygwin compiler where to look for additional include directories. (On Linux: **/home/<user>/systemc-2.2.0/include/)**

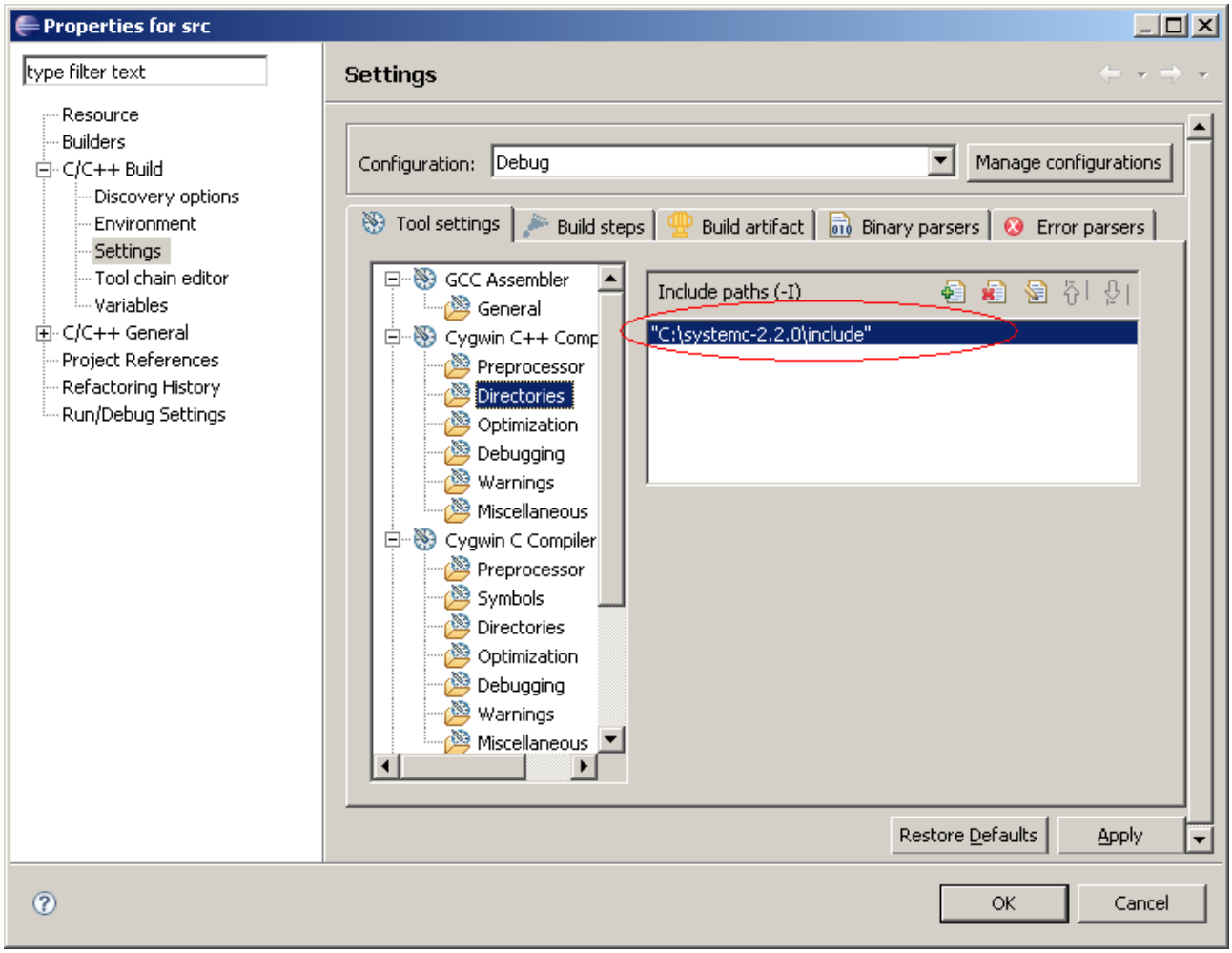

Figure 9 Eclipse SystemC include path

5. Select the systemc library and add the path to the library that was built in the previous chapter ‘Installing SystemC with cygwin’.

The “**-I**” directive indicates the additional libraries to be linked into the programme in this case “**systemc**”. The “**–L** “directive indicates for the cygwin linker where to look for additional library directories.

(On Linux Library search path (-L): **/home/<user>/systemc-2.2.0/lib-linux/)**

.

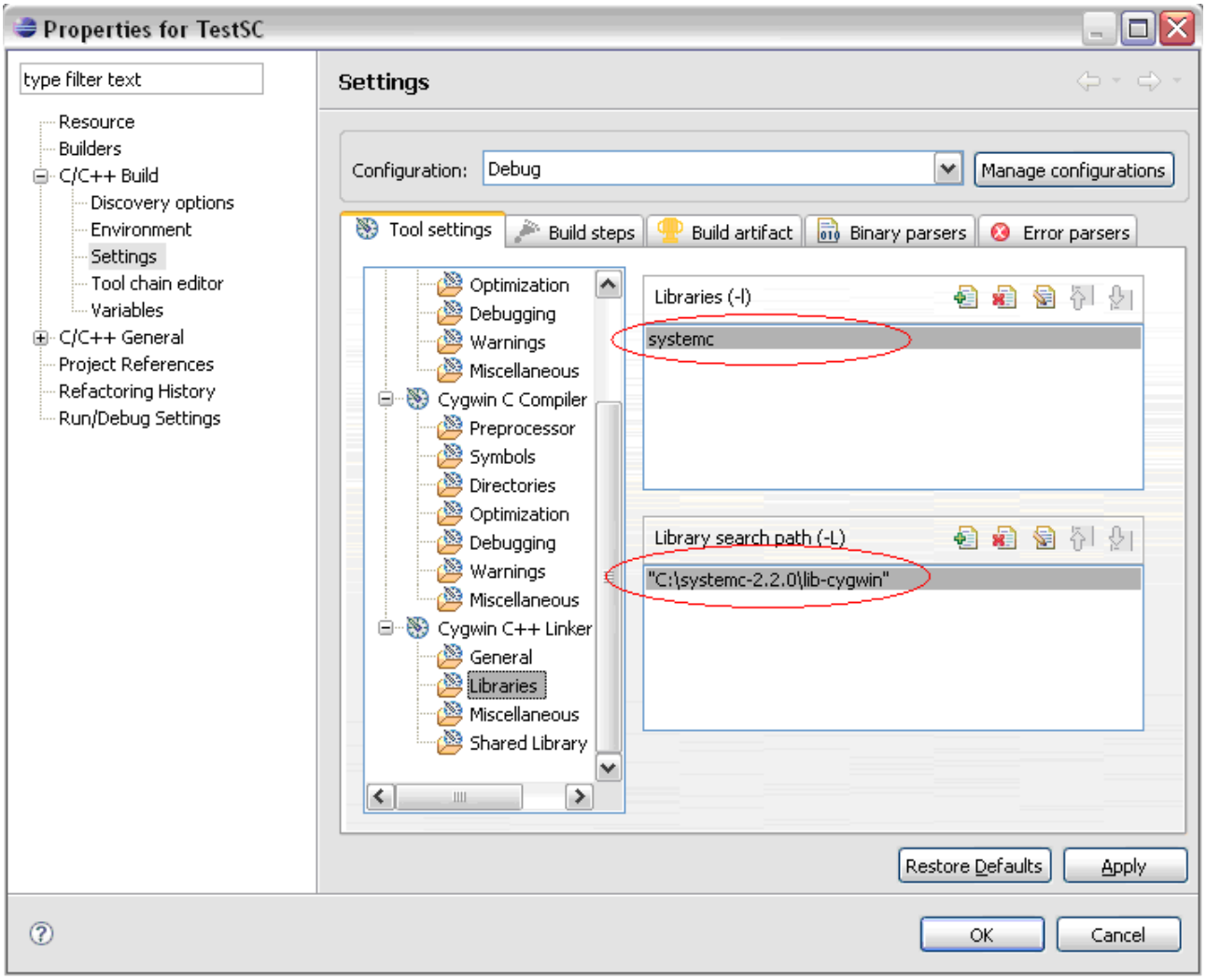

Figure 10 Eclipse SystemC library

6. Press finish and you will now have a small C++ programme that will print “Hello World”. You can rebuild this project by selecting Project -> Clean, and the programme will automatically perform a ‘make all’. See below.

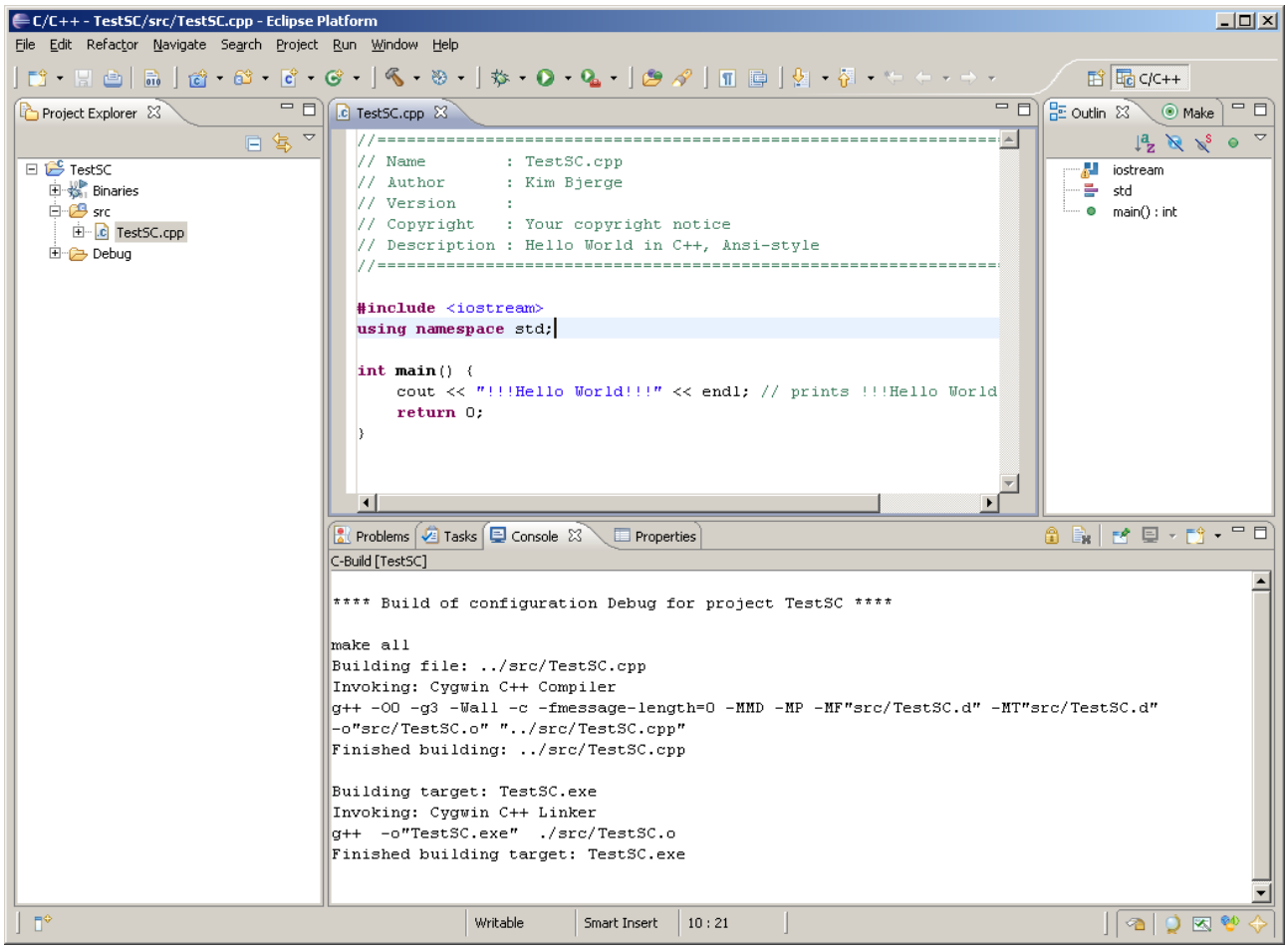

Figure 11 Eclipse build of ”Hello World”

7. The programme is now ready to be executed by selecting the ►Run button. You have to choose the 'Cygwin gdb Debugger' (On Linux just 'gdb Debugger')

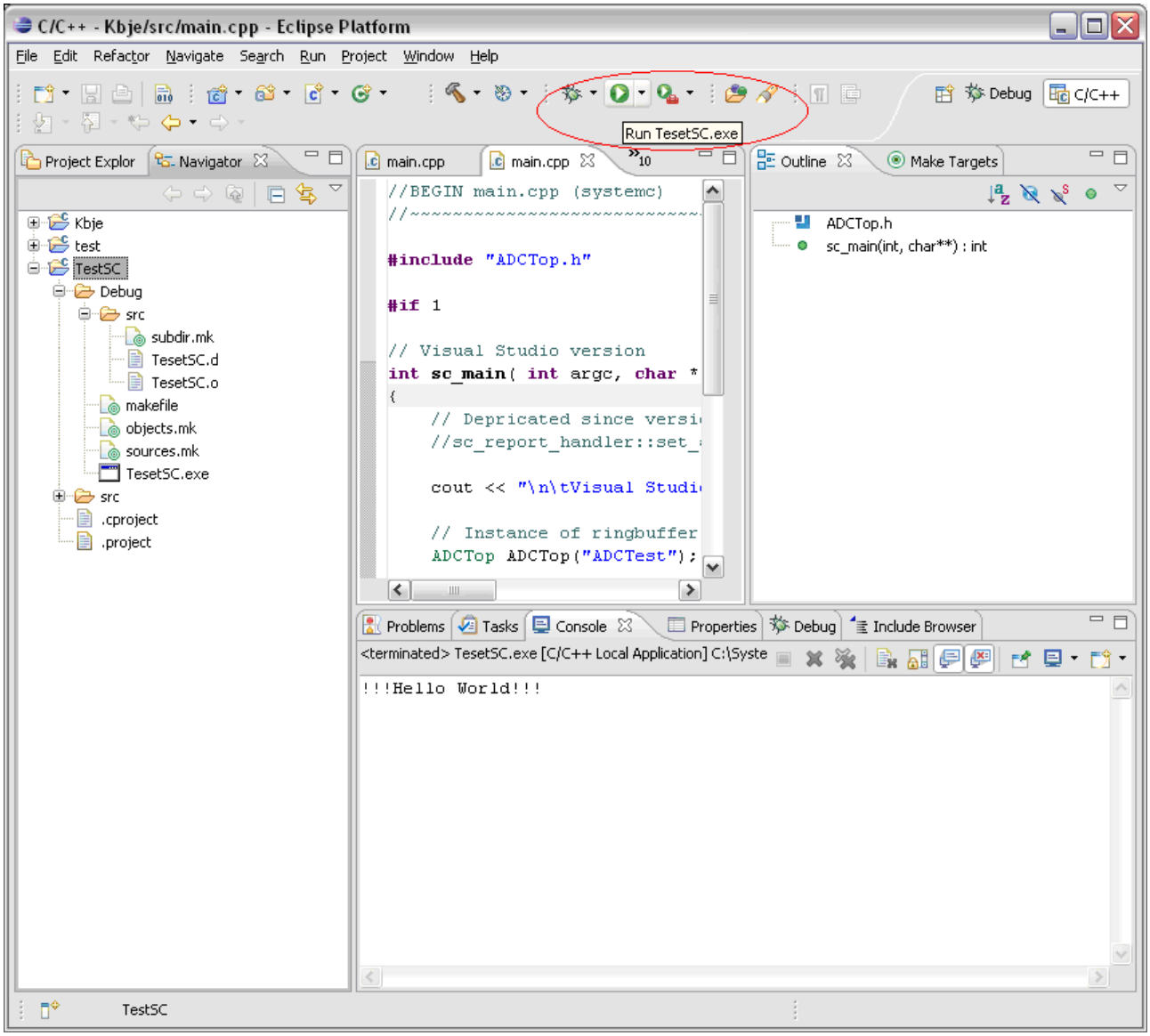

Figure 12 Eclipse run of "Hello World"

8. The next step is to copy the SystemC example files (ADCInput.zip file) to the 'TestSC\src' directory.

Right-click on the '**src**' folder and press **Refresh (F5).** The added example files will then appear in the source folder. Right-click on the TestSC.cpp file and delete this file.

You can now rebuild the project by right-clicking on the mouse for the TestSC project. Select 'Build project' and hereafter select the ►Run button. The result from the ADCInput project will appear in the Console window, see below. The sample programme has generated an ADCInputWave.vcd file which you can view with the GtkWave.exe file. See chapter '*Setup of the GtkWave viewer*'.

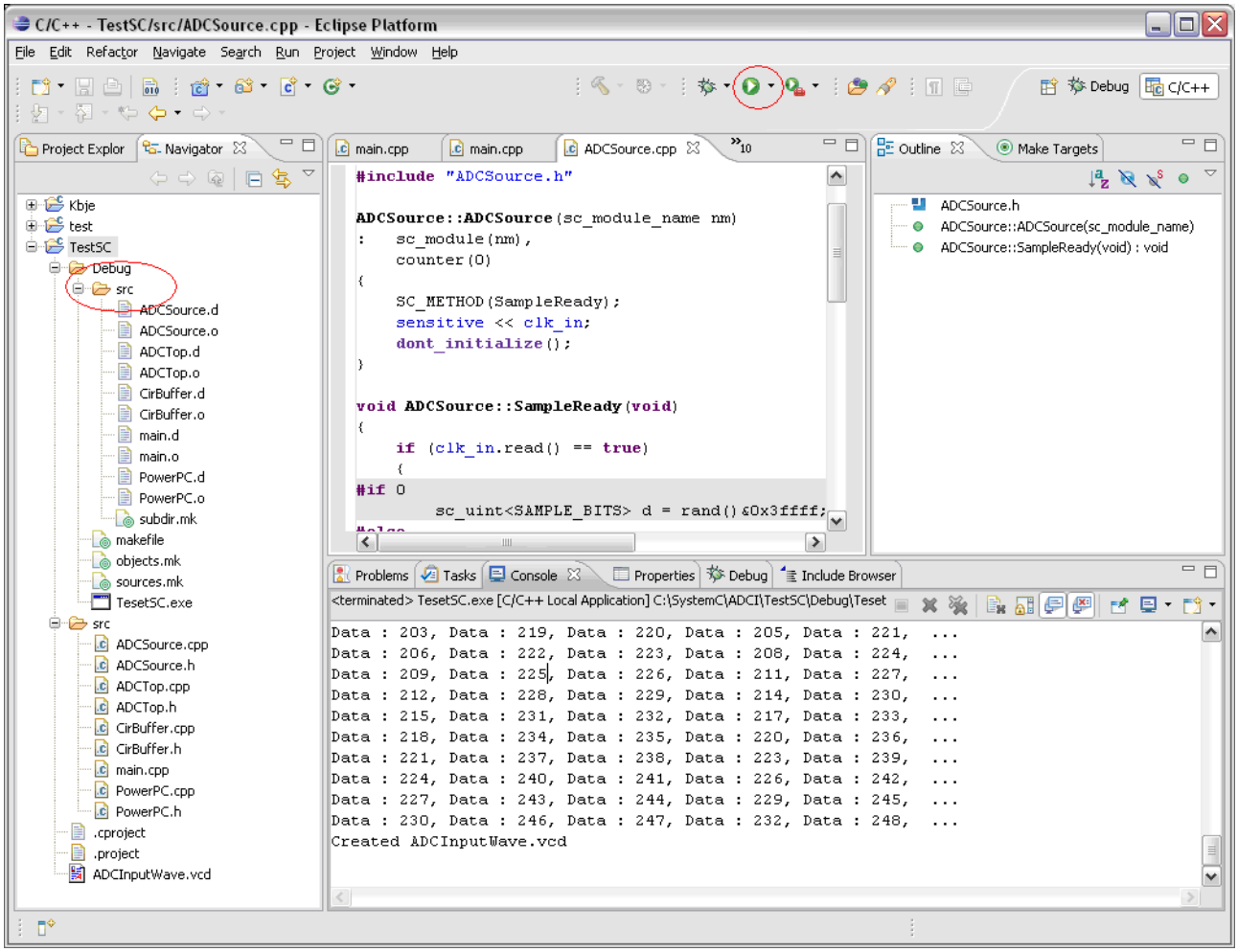

Figure 13 Eclipse build and run of SystemC test project

Setup for Microsoft Visual Studio .NET applications

This part concerns installation and setup for developing SystemC applications with Microsoft Visual Studio .NET 2005.

Installing SystemC with Visual Studio .NET 2005

It is assumed that Visual Studio .NET 2005 is already installed.

This guide concerns installation of version 2.2.0 for SystemC.

SystemC can be downloaded from the website of (OSCI) Open SystemC Initiative. See link http://www.systemc.org/home.

The SystemC source is to be found on the web page: **DOWNLOADS -> OSCI Standards -> SystemC 2.2**. Before being able to download this source code, you have to create an account.

Additional information is to be found in the INSTALL and README files located in the SystemC package. This guide should, however, be sufficient to get started.

1. Unzip the systemc-2[1].2.0.tgz to the folder: (NB: Skip this stage if already unzipped)

   C:\systemc-2.2.0

2. Open the Visual Studio solution for SystemC and accept to update it from version 7.1 to 8.

   The solution is to be found in: C:\systemc-2.2.0\msvc71\SystemC\SystemC.sln

3. The next step is to build the SystemC libraries for Visual Studio projects

   Select the Debug configuration and make a build solution (F7).
   Select the Release configuration and make a build solution (F7).

The compiled SystemC libraries are to be found here:

C:\ systemc-2.2.0\msvc71\SystemC\Release\SystemC.lib
C:\ systemc-2.2.0\msvc71\SystemC\Debug\SystemC.lib

4. Create the environment variable:

   SYSTEMC = C:\systemc-2.2.0\msvc71

5. Test the setup by starting the C:\systemc-2.2.0\examples\sysc\fir\fir.vcproj

It is found in the SystemC examples which are part of the SystemC package.
Exclude the main.cpp before build of this example.

If you try to build the release version you have to change the properties in '**C/C++ -> Code Generation -> Basic Runtime Checks**' from 'Both (/RTC1, equiv. to /RTCsu)' to '**Default**'.

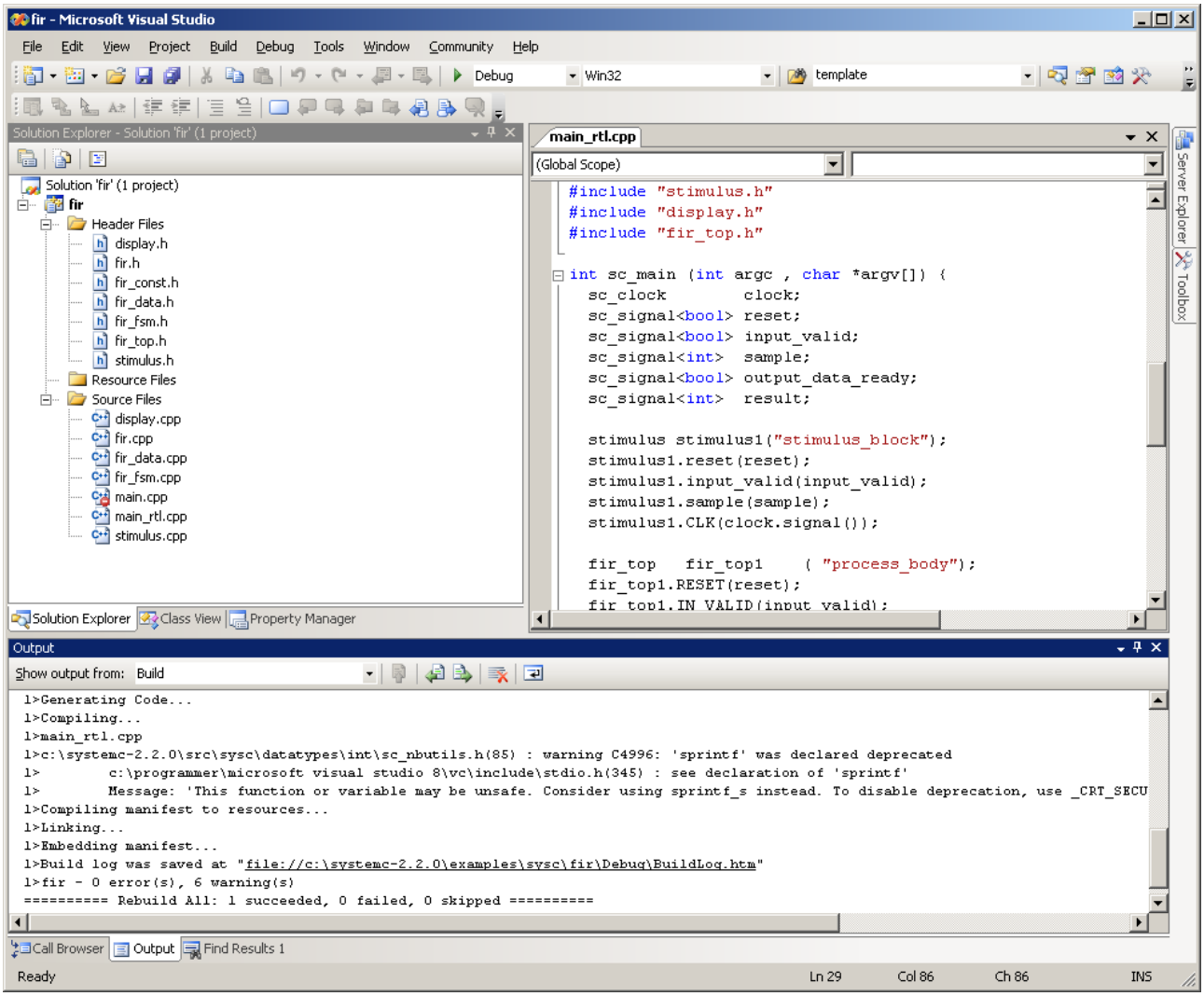

Figure 14 Visual Studio build of SystemC example project

Now run the fir.exe programme and the result should appear in the cmd window as shown below:

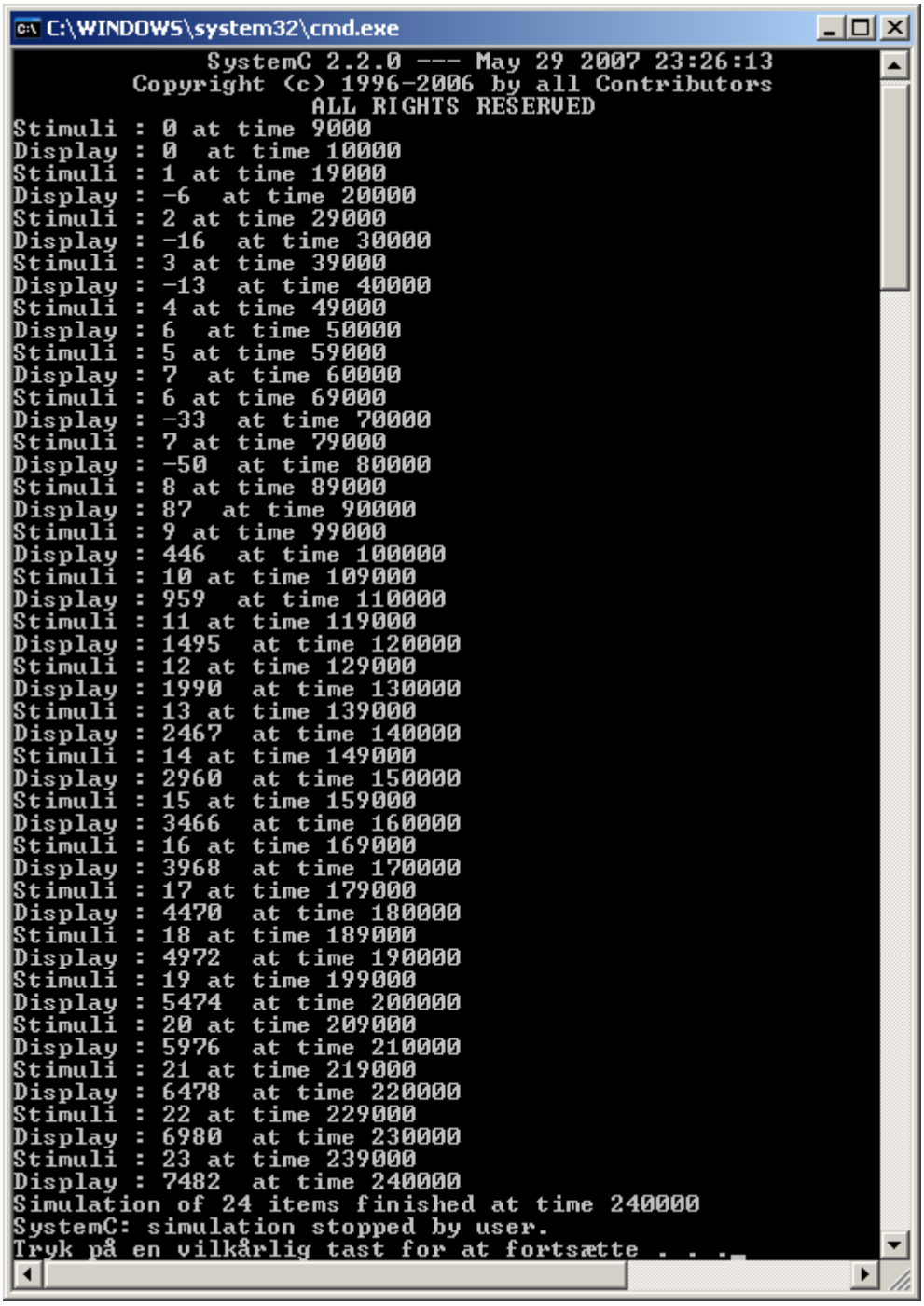

Figure 15 Visual Studio running SystemC example project

Creating SystemC Applications with Visual Studio .NET 2005

1. Start Visual Studio. From the Start Page select **New Project** and **Win32 Console Application**. Type project name and select a suitable location. Click OK.

2. Select the Application Settings page of the Win32 Application Wizard and make sure that the **'Empty project'** box is ticked. Click 'Finish' to complete the wizard.

3. Add **new/existing C++ files** to the project and edit code.

You could use the example files from the "**ADCInput.zip**" SystemC project made by the Danish Technological Institute.

4. Display the project Property Pages by selecting 'Properties...' from the Project menu.

5. From the **C/C++** tab, select the **General** properties and set **'Detect 64-bit Portability Issues' to No**

6. From the **C/C++** tab, select the **Preprocessor** properties and add the **_CRT_SECURE_NO_DEPRECATE** definition.

7. From the **C/C++ tab**, select the **Code Generation** properties and set
   For Debug build:
   'Runtime Library' to Multi-threaded Debug (/MTd)
   For Release build:
   'Runtime Library' to Multi-threaded (/MT)

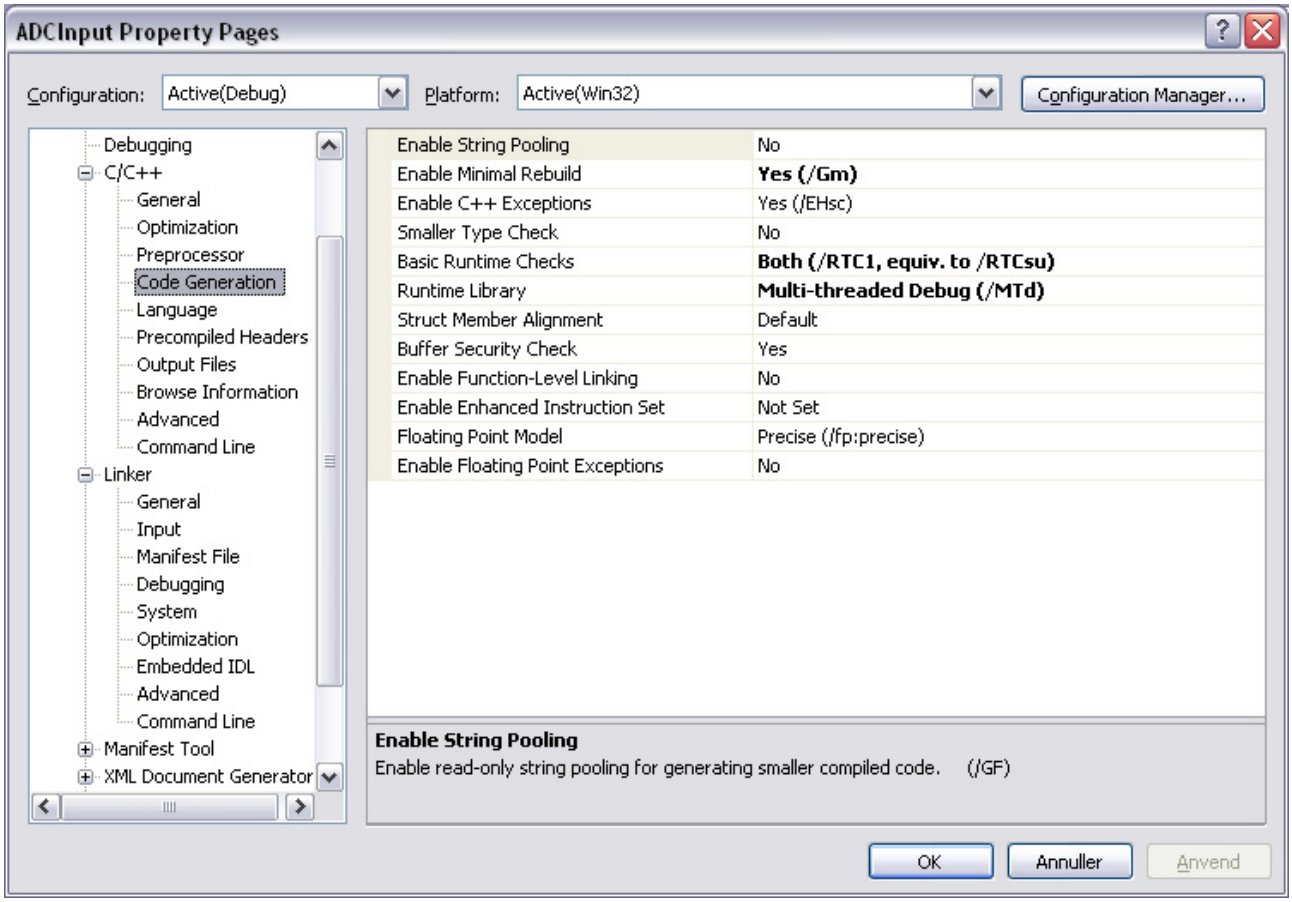

Figure 16 Visual Studio SystemC project properties

8. From the **C/C++** tab, select the **Language** properties and set **'Enable Run-Time Type Info' to Yes**

9. From the **C/C++** tab, select the **Command Line** properties and add **/vmg** to **the 'Additional Options:'** box.

10. From the Linker tab, select the **Input** properties and type **'systemc.lib'** in **the 'Additional Dependencies'** box.

11. Click OK

Also make sure that the compiler and linker can find the SystemC header and library files respectively. There are two ways to do this, both can refer to an environment variable pointing to the SystemC install path:

Update Include and Library directory for current project only

To add the include file and library directory search paths for the current project only:

1. Display the project Property Pages by selecting **'Properties...'** from the Project menu.

2. From the **C/C++** tab, select the **General** properties and type the path to the SystemC 'src' directory in the text entry field labeled **'Additional include directories'** (e.g. the examples use **'$(SYSTEMC)\..\src'**).

3. From the **Linker** tab, select **General** properties and type the path to the SystemC library in the text entry field labeled **'Additional Library Directories'**
(e.g. the examples use **'$(SYSTEMC)\SystemC\Debug'**).

4. From the **Linker** tab, select **Input** properties and enter **'systemc.lib'** in the **'Additional Dependencies'** text entry field.

5. Click OK

Update Include and Library directory for all projects

To update the include file and library directory search paths for all projects:

1. Select **Tools** -> **Options** . . . and the **Projects** -> **VC++ Directories tab**

2. Select show directories for: **Library** files

3. Select the **'New'** icon and enter**: $(SYSTEMC)\SystemC\Debug**

4. Select show directories for: **Include** files

5. Select the **'New'** icon and enter: **$(SYSTEMC)\..\src**

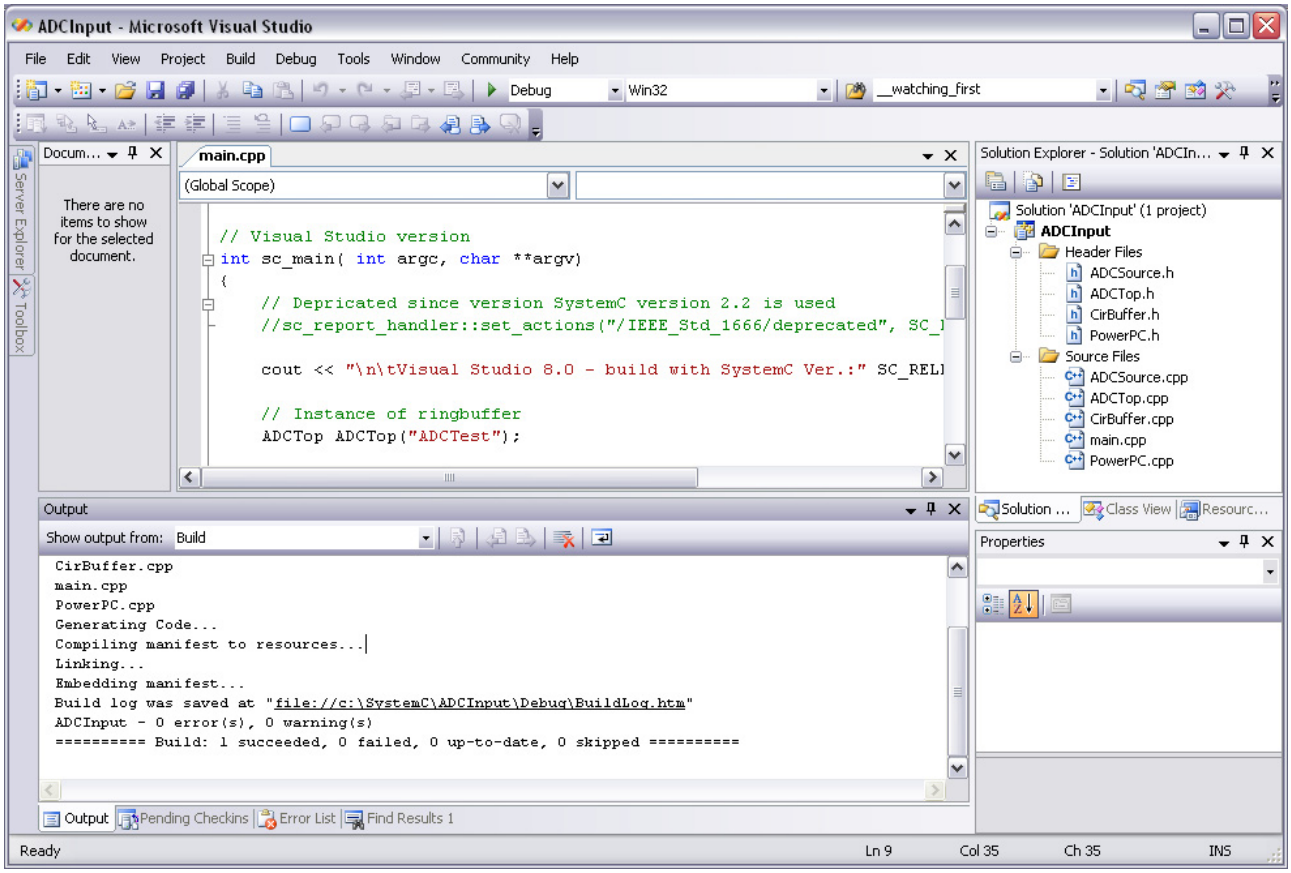

Figure 17 Visual Studio build of SystemC project

The project should now be ready to perform a build solution (F7). If you have used the ADCInput files you will be able to build and run the programme. You will get a result like this :

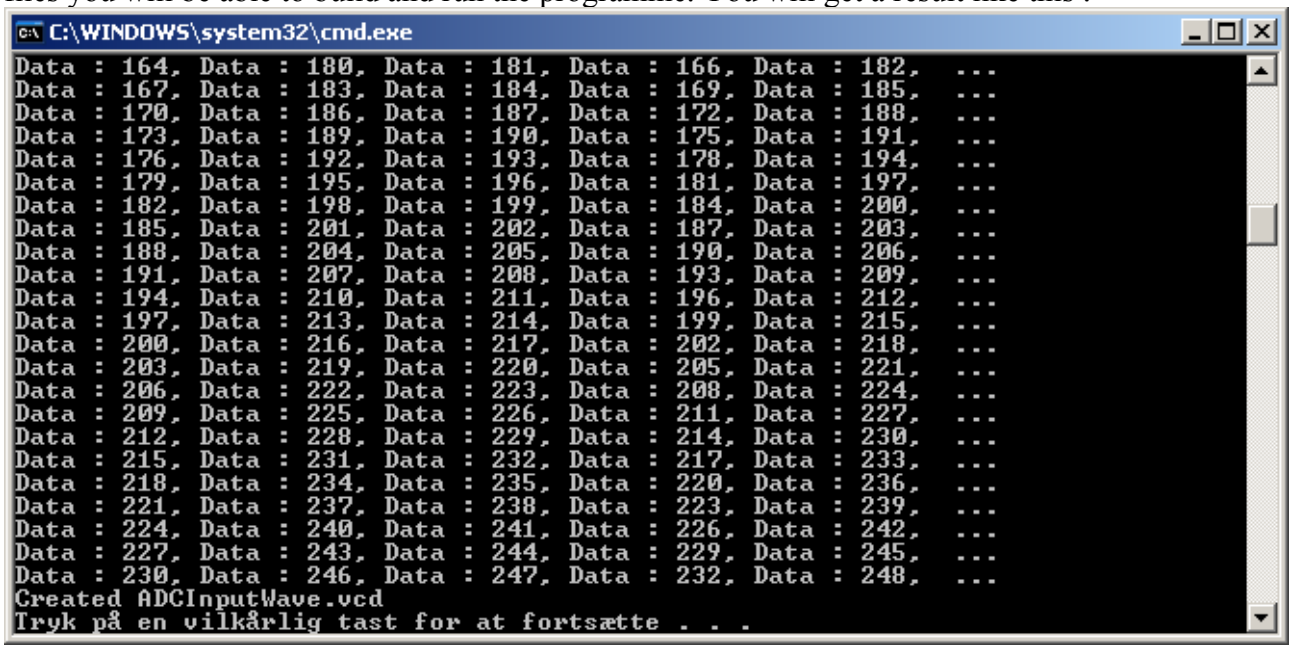

Figure 18 Visual Studio running SystemC programme

You have to make the same setup in the project Property Page for the Release build configuration when building the release solution. The sample programme has generated an ADCInputWave.vcd file which you can view with the GtkWave.exe file. See chapter "*Setup of the GtkWave viewer*".

Setup of the GtkWave viewer

The GtkWave programme can be downloaded via http://home.nc.rr.com/gtkwave/.

1. You have to unzip the gtkwave-1.3.24.tar.gz file and place it in the root like C:\gtkw. Then add the directory **C:\gtkw\bin** to the path in the **environment variable** settings for your Windows.

2. Next step is to copy the GtkWave.exe file to the C:\gtkw\bin directory.

3. Then set the GtkWave.exe programme to be the default programme in Explorer to open files with the extension *.VCD (See below Danish version of Windows XP)

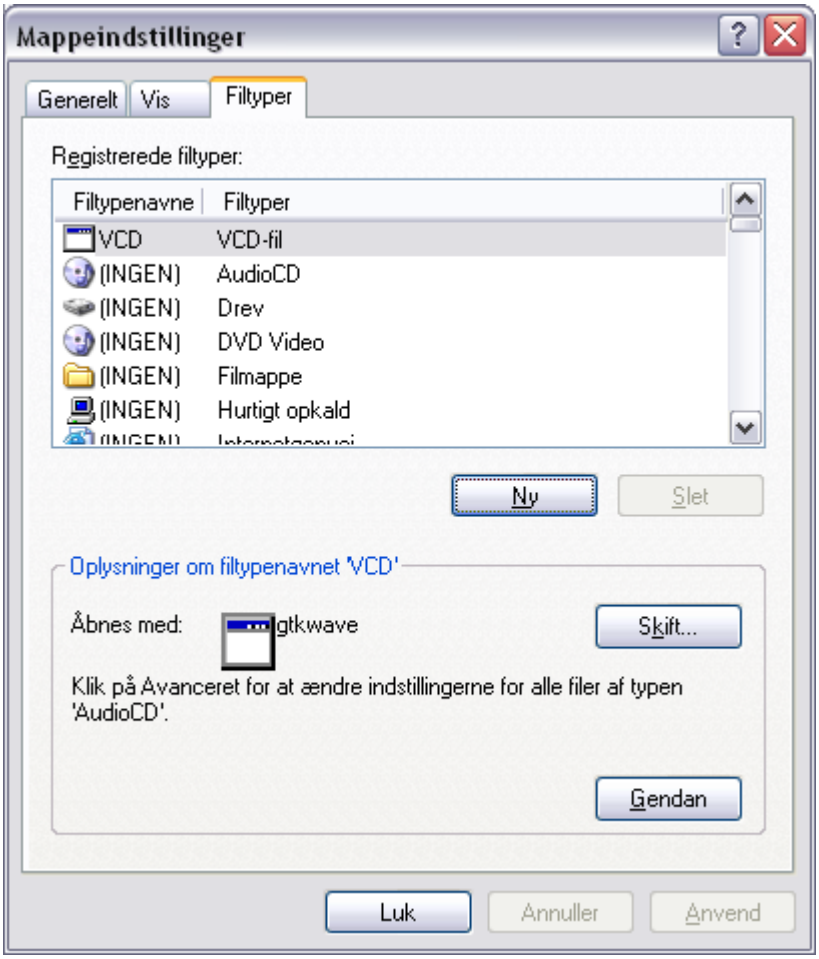

Figure 19 Explorer settings for opening VCD files with GtkWave

4. When you now try to open by clicking on the ADCInput.vcd file which was generated by the example programme in previous chapters, the wave file will be open automatically. You have to select SystemC in the SST window and add signals to the Waves window. Now zoom – out until you can see the signals shown in the figure below.

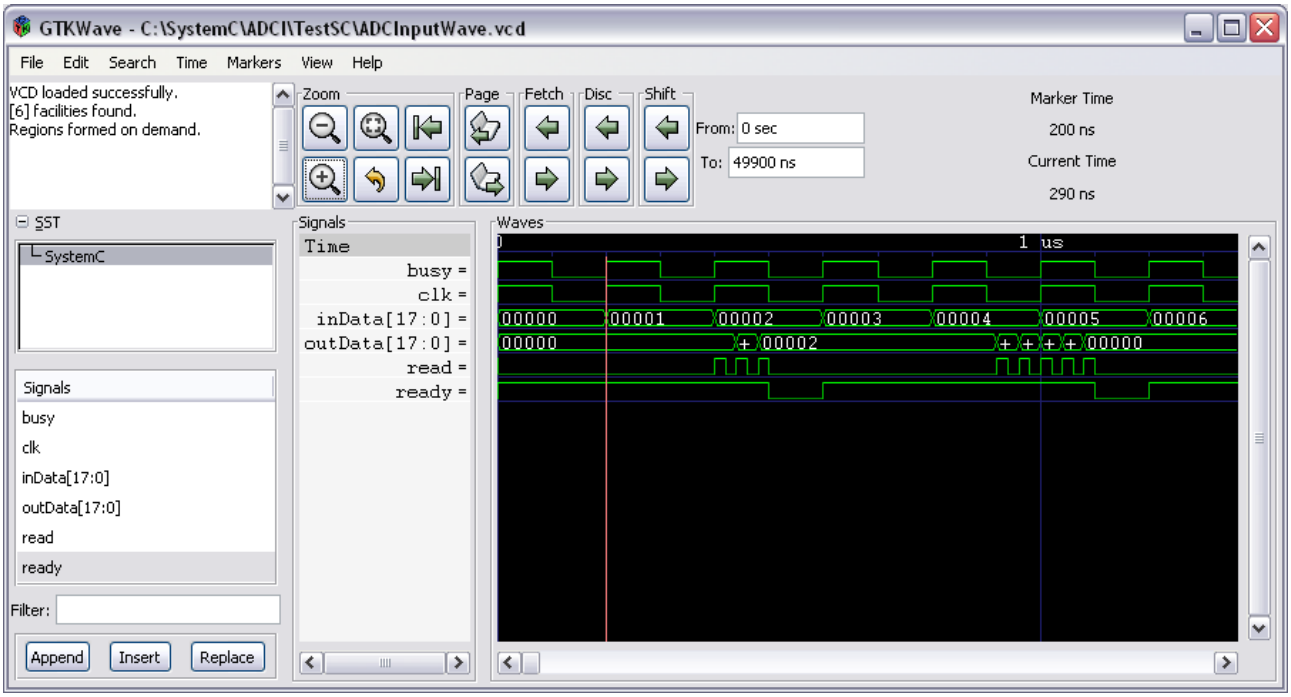

Figure 20 GTKWave viewer

Linux setup for SystemC

This part contains the description for using Eclipse and SystemC on Linux. For this installation we have used the Toplogilinux distribution that can be downloaded from http://www.topologilinux.com/.

Topologiliunx is a linux version to be run on top of or inside your existing windows system. Most importantly, it does not require any partitioning at all. (It uses a single file as the linux root system) The kernel version of linux is not modified and based on kernel version 2.6.17.13, the release of Topologilinux is 6.1.0.

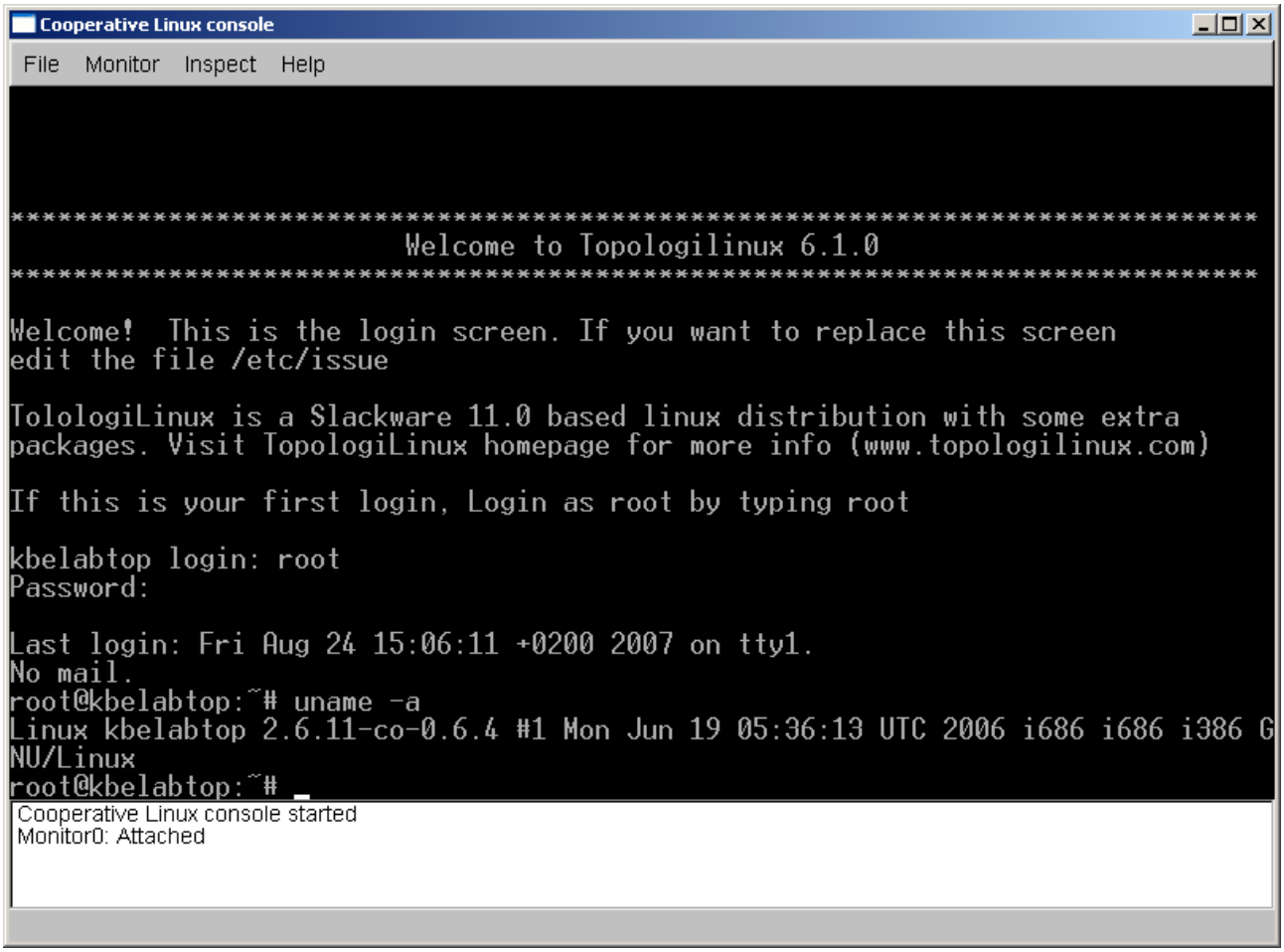

Figure 21 Linux version

Installing SystemC on Linux

This guide concerns installation of version 2.2.0 for SystemC.

SystemC can be downloaded from the website of (OSCI) Open SystemC Initiative, see link http://www.systemc.org/home.

The SystemC source is to be found on the website: **DOWNLOADS -> OSCI Standards -> SystemC 2.2**. Before being able to download this source code, you have to create an account.

Additional information is to be found in the INSTALL and README files that are located in the SystemC package. This guide should, however, be sufficient to get started.

Before you start installing the SystemC you must have a Linux platform to work on:

1. Unzip the systemc-2[1].2.0.tgz to the folder:

   /home/<user>/systemc-2.2.0

2. Create the objdir and change to this directory

   /home/<user>/systemc-2.2.0/objdir

3. Configure the package for your system.

   $ ../configure

4. Compile the package.

   For a debug SystemC library, enter:

   $ make debug

   Alternatively, for an optimized SystemC library, enter: **$ make opt**

5. Install the package

   $ make install

   The SystemC library will be stored in :

   /home/<user>/systemc-2.2.0/lib-linux/libsystemc.a

6. At this point you may wish to verify the installation by testing the example suite.

   $ make check

   This will compile and run the SystemC examples in the subdirectory example

Creating SystemC application on Linux with Eclipse

Since Eclipse is the same on Windows and Linux you can use the same guide as described in the Windows installations. See "Creating SystemC Applications with Eclipse". Just follow the same guide for creating a SystemC project.

When you create a new project, the Tool chain must be "Linux GCC" instead of "Cygwin GCC".

Include and library paths must be different on Linux see below:

Include path (-I): **/home/<user>/systemc-2.2.0/include/**
Library search path (-L): **/home/<user>/systemc-2.2.0/lib-linux/**

References

The following books will provide a good starting point for learning about SystemC.

"SystemC: From the Ground Up" by David C. Black and Jack Donovan

This is a fine book for learning SystemC from a system-level design point. It gives an introduction to Transaction-Level Modelling (TLM) and SystemC language, including the architecture and C++ classes' library. It covers the newer version of SystemC 2.1 and requires a good knowledge of C++. There are a lot of SystemC examples for every chapter which can be downloaded from http://www.eklectically.com/.

"A SystemC Primer, Seconf Edition" by J. Bhasker

This is a good book for learning SystemC from a register transfer level (RTL) design point. It focuses on making simulations in SystemC of digital electronics and is a good starting point if you are used to making implementations in Verilog or VHDL code. It only covers the TLM modelling features of the SystemC languages beyond RTL to some extent.

"Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded System"by Frank Ghenassia

This is a good book which provides an overview for designing System-on-Chip (SoC), including complex FPGA designs. It covers the theory about modelling and verification. It is not a book on how to programme SystemC but provides a good overview of areas like Transaction Level Modelling, Techniques, Embedded Software Development and Functional Verification in the area of modelling and simulation for developing SoC and advanced FPGA systems.