Sourcery VSIPL++

Getting Started

Version 2.2-9

Sourcery VSIPL++: Getting Started: Version 2.2-9

CodeSourcery, Inc.

Table of Contents

Preface

Abstract

This preface introduces *Getting Started With Sourcery VSIPL++*. It explains the structure of this book and lists other sources of information that relate to Sourcery VSIPL++.

1. Intended Audience

This book is written for users and system administrators who will install Sourcery VSIPL++. Parts of this document assume that you have some familiarity with the UNIX Borne Shell and with compiling C++ programs from the command line.

2. Organization

This document is organized into the following chapters and appendices:

Chapter 1, "Licensing"	This chapter describes the terms under which Sourcery VSIPL++ is licensed. Read this chapter to find out what rules apply to you as a user of Sourcery VSIPL++.
Chapter 2, "Installation"	This chapter explains how to install Sourcery VSIPL++ binary packages.
Chapter 3, "Installation from Source"	This chapter explains how to configure, install, and build Sourcery VSIPL++ from a source package.
Chapter 4, "Building Applications"	This chapter explains how to build and run applications with Sourcery VSIPL++. Read this chapter to find out how to build a simple application. You can use the simple application as a template for building more complex applications.
Appendix B, "GNU General Public License"	This appendix contains the full text of the GNU General Public License. If you have not purchased a Sourcery VSIPL++ subscription from CodeSourcery, then this license applies to Sourcery VSIPL++.

3. Typographical Conventions

The following typographical conventions are used in this book:

`> command arg ...`	A command, typed by the user, and its output. The ">" character is the command prompt.
command	The name of a program, when used in a sentence, rather than in literal input or output.
`literal`	Text provided to or received from a computer program.
`placeholder`	Text that should be replaced with an appropriate value when typing a command.

4. Further Reading

Sourcery VSIPL++ Website	http://www.codesourcery.com/vsiplplusplus/
Sourcery VSIPL++ User's Guide	http://www.codesourcery.com/vsiplplusplus/users_guide.pdf
Sourcery VSIPL++ Reference Manual	http://www.codesourcery.com/vsiplplusplus/reference_manual.pdf

Sourcery VSIPL++ Frequently Asked Questions	http://www.codesourcery.com/vsiplplusplus/faq.html
VSIPL++ API Specification	http://www.codesourcery.com/public/vsiplplusplus/specification-1.0.pdf

Chapter 1
Licensing

Abstract

Sourcery VSIPL++ is available under three licenses. CodeSourcery customers may use the entire library under terms suitable for use in proprietary software, including software designed for use in classified systems. Other users may either use the entire library under the GNU General Public License, which requires that source code for Sourcery VSIPL++ applications be provided to their users, or they may use the core parts of the library that make up the reference implementation under the BSD license.

1.1. Proprietary Software

If you have a Sourcery VSIPL++ license from CodeSourcery, you may use Sourcery VSIPL++ under the terms of that agreement. The full text of the license is included in Appendix A, “Sourcery VSIPL++ Software License Agreement”.

If you are planning to use Sourcery VSIPL++ to develop proprietary applications, please contact `<sales@codesourcery.com>` or visit the Sourcery VSIPL++ web site[1] to find out more about obtaining a Sourcery VSIPL++ software license and support agreement.

1.2. Free and Open-Source Software

If you are planning to use Sourcery VSIPL++ to develop a free or open-source software application and are willing to share the source code for your application, you may use Sourcery VSIPL++ at no charge under the terms of the GNU General Public License 2.0. The full text of the license is included in Appendix B, “GNU General Public License”. The GPL requires that you share the source code for your application with everyone to whom you distribute your application. You have certain other rights and obligations under the GPL, as described in the appendix.

1.3. Reference Implementation

If you are planning to use the VSIPL++ reference implementation, you may use the core library parts of Sourcery VSIPL++ under the terms of the BSD license. The full text of the license is included in Appendix C, “ BSD License”. The exact files available under the BSD license are documented in the file `LICENSE` located in the top level directory of the source package.

[1] http://www.codesourcery.com/vsiplplusplus/

Chapter 2
Installation

Abstract

This chapter explains how to install Sourcery VSIPL++ from a precompiled binary package. Installing Sourcery VSIPL++ requires three steps:

1. Ensure that your system can run Sourcery VSIPL++.
2. Download a Sourcery VSIPL++ binary package for your system.
3. Install Sourcery VSIPL++.

Alternatively, if an appropriate pre-built binary package is not available, Sourcery VSIPL++ may be compiled and installed from a source package. Instructions for this option may be found in Chapter 3, “Installation from Source”.

2.1. System Requirements

This section explains what requirements your system must meet in order to run Sourcery VSIPL++.

2.1.1. Operating System

Sourcery VSIPL++ binary packages are available for the following distribution / architectures / compiler sets (The C++ ABI is not standardized between compilers, and thus a C++ library compiled with one compiler may not reliably work with a different compiler):

- x86 / RHEL 4 (32 and 64-bit) / GCC 3.4.6
- Cell/B.E. Fedora 9 (32 and 64-bit) / GCC 4.1.1
- PowerPC / Fedora 9 (32 and 64-bit) / GCC 4.3

2.1.2. General Required Software

This section describes software that you must have installed in order to install and use a Sourcery VSIPL++ binary package on any system.

2.1.2.1. RPM

For GNU/Linux systems, Sourcery VSIPL++ binary packages are available as Redhat Package Manager (RPM) files and as compressed Tape Archive (Tar) files. When using Sourcery VSIPL++ RPM files, it is necessary to have RPM installed. RPM is installed by default on RHEL and Fedora.

2.1.2.2. GNU Tar

When using Sourcery VSIPL++ Tar files, CodeSourcery strongly recommends that you use GNU Tar to unpack the binary package due to incompatibilities between various versions of Tar. You can download GNU Tar as source code from http://ftp.gnu.org/pub/gnu/tar. Pre-compiled GNU Tar binaries are available for all popular operating systems.

2.1.3. Specific Software for x86 / RHEL 4

This section describes required and optional software that you may install on your system in order to use the Sourcery VSIPL++ binary package for x86 / RHEL 4 systems. In order to use Sourcery VSIPL++, you must have software labeled as "Required" installed on your system. You can use Sourcery VSIPL++ without installing software labeled as "Optional." However, if you install these packages, Sourcery VSIPL++ may provide extra functionality or enhanced performance.

Both 32-bit and 64-bit applications can be built with Sourcery VSIPL++. To build 32-bit applications, 32-bit versions of the RPMs listed below must be installed. Likewise, to build 64-bit applications, 64-bit versions of the RPMs listed below must be installed.

2.1.3.1. x86 GNU Toolchain (Required)

In order to compile application programs using Sourcery VSIPL++ it is necessary to have the x86 GNU 3.4.6 toolchain installed.

The x86 GNU toolchain is part of the RHEL v4 distribution. The following RPM packages are required:

- gcc

- gcc-c++

2.1.3.2. Intel IPP and MKL (Optional)

The Intel Performance Primitives (IPP) and Intel Math Kernel Library (MKL) can be used by Sourcery VSIPL++ to accelerate some functions, including FFTs. Sourcery VSIPL++ for RHEL v4 requires that both libraries be used together. While IPP/MKL are optional, if they are not used, it is necessary to install ATLAS instead.

Sourcery VSIPL++ expects IPP to be installed in `/opt/intel/ipp` and MKL to be installed in `/opt/intel/mkl`.

To find out more about IPP and MKL visit

- http://software.intel.com/intel-ipp/
- http://software.intel.com/intel-mkl/

2.1.3.3. Automatically Tuned Linear Algebra Software (ATLAS) (Optional)

ATLAS may be used to accelerate some matrix-vector and linear-algebra functions in Sourcery VSIPL++. ATLAS is optional, but if it is not installed, Sourcery VSIPL++ requires Intel IPP and MKL to be installed instead.

ATLAS is available in the RHEL v4 distribution. The following RPM packages are required:

- atlas
- atlas-devel

2.1.3.4. LAM/MPI (Optional)

LAM/MPI is an optional library that provides MPI bindings for message passing. Sourcery VSIPL++ can use multiple cores, processors, or compute nodes simultaneously when LAM/MPI is installed.

LAM/MPI is available as an RPM package in the RHEL v4 distribution. The following RPM packages are required:

- lam-7.1.2
- lam-libs-7.1.2
- lam-devel-7.1.2

You can find out more information on LAM/MPI at http://www.lam-mpi.org/.

2.1.4. Specific Software for Cell/B.E. / Fedora 9

This section describes required and optional software that you may install on your system in order to use the Sourcery VSIPL++ binary package for Fedora 9 / Cell/B.E. systems. In order to use Sourcery VSIPL++, you must have software labeled as "Required" installed on your system. You can use Sourcery VSIPL++ without installing software labeled as "Optional." However, if you install these packages, Sourcery VSIPL++ may provide extra functionality or enhanced performance.

Both 32-bit and 64-bit applications can be built with Sourcery VSIPL++. To build-32-bit applications, 32-bit versions of the RPMs listed below must be installed. Likewise, to build 64-bit applications, 64-bit versions of the RPMs listed below must be installed.

2.1.4.1. Cell/B.E. GNU Toolchain (Required)

In order to compile application programs using Sourcery VSIPL++ it is necessary to have the Cell/B.E. GNU 4.1.1 toolchain installed. This toolchain is freely available and may be downloaded from the Barcelona Supercomputing website at http://www.bsc.es/plantillaH.php?cat_id=579.

The following RPM packages are required:

- ppu-gcc-4.1.1
- ppu-gcc-c++-4.1.1
- ppu-binutils

The following RPM packages are required only if you plan to write user-defined kernels:

- spu-gcc-4.1.1
- spu-gcc-c++-4.1.1
- spu-binutils
- spu-newlib

2.1.4.2. Cell/B.E. SPE Runtime Management Library 2 (Required)

In order for Sourcery VSIPL++ application programs to use the SPEs, it is necessary to have the Cell/B.E. SPE Runtime Management Library 2 (libspe2) installed.

This library is freely available and may be downloaded from the Barcelona Supercomputing Website at http://www.bsc.es/plantillaH.php?cat_id=581.

The following RPM packages are required:

- libspe2
- libspe2-devel

2.1.4.3. Automatically Tuned Linear Algebra Software (ATLAS) (Required)

Automatically Tuned Linear Algebra Software is used to accelerate some matrix-vector and linear-algebra functions in Sourcery VSIPL++.

ATLAS is available as an RPM package in the Fedora 9 distribution. For 32-bit, the following RPM packages are required:

- atlas-altivec
- atlas-altivec-devel

For 64-bit, the following packages are required:

- atlas
- atlas-devel

CodeSourcery recommends using the non-AltiVec version of ATLAS for 64-bit because Fedora's 64-bit AltiVec ATLAS contains bugs resulting in illegal-instruction segfaults. The 32-bit AltiVec ATLAS does not exhibit these bugs.

Visit http://math-atlas.sourceforge.net for general information about ATLAS.

2.1.4.4. NUMACTL (Optional)

NUMACTL provides mechanisms for Sourcery VSIPL++ to control the PPEs and SPEs used by a process and its threads. By controlling locality, Sourcery VSIPL++ can improve application performance in some cases.

NUMACTL is available in RPM packages provided by the Fedora 9 distribution. The following RPM packages are required:

- numactl
- numactl-devel

2.1.4.5. OpenMPI (Optional)

OpenMPI is an optional library that provides MPI bindings for message passing. Sourcery VSIPL++ can use multiple compute nodes simultaneously when OpenMPI is installed. Note that OpenMPI is not necessary for Sourcery VSIPL++ to use multiple SPEs on a single Cell/B.E., or to use multiple PPEs and SPEs with a coherent interconnect such as on an IBM BladeCenter(R) QS20, QS21, or QS22 blade. It is only necessary if Sourcery VSIPL++ will be used on multiple blades connected via ethernet or similar network fabric.

OpenMPI is available in RPM packages provided by the Fedora 9 distribution. The following RPM packages are required:

- openmpi
- openmpi-devel
- openmpi-libs

2.2. Installing a Binary Package

This section explains how to install and run Sourcery VSIPL++ from a pre-built package. Pre-built Sourcery VSIPL++ packages are available from CodeSourcery's customer support portal. Visit your account http://www.codesourcery.com/VSIPLXX to download packages.

2.2.1. Unpacking the Distribution RPM

Sourcery VSIPL++ binary packages are available as RPM packages. The following commands use RPM to unpack the binary distribution into the location `/opt/sourceryvsipl++-2.2-9`:

```
> rpm -i sourceryvsipl++-2.2-9-noarch.rpm
```

2.2.2. Unpacking the Distribution Tarball

Sourcery VSIPL++ binary packages are distributed as compressed Tape Archive (Tar) files that are intended to be installed in the `/opt` directory. The following commands use GNU Tar to unpack the binary distribution into the location `/opt/sourceryvsipl++-2.2-9`:

```
> cd /opt
> tar -xjf path/to/sourceryvsipl++-2.2-9.tar.bz2
```

Replace *path/to/sourceryvsipl++-2.2-9.tar.bz2* with the location and name of the particular Sourcery VSIPL++ package you are installing.

If you will be using pkg-config to determine compile and link time options, you should include the directory `/opt/sourceryvsipl++-2.2-9/lib/pkgconfig` in your **PKG_CONFIG_PATH** environment variable:

```
> export PKG_CONFIG_PATH=/opt/sourceryvsipl++-2.2-9/lib/pkgconfig
```

The directory layout of the installed package is as follows:

```
`- opt
   `- sourceryvsipl++-2.2-9
      |- bin                 // General executables and scripts
      |  |- [arch]           // Arch-specific binary files
      |  |  `- [variant]     // Variant-specific binary files
      |  |     `- benchmarks // Benchmark executables
      |- include
      |  |- vsip             // Sourcery VSIPL++ Headers
      |  |- vsip_csl         // CodeSourcery extensions
      |  `- ...             // Other non-SV++ headers as necessary
      |- lib
      |  |- [arch]           // Arch-specific library files
      |  |  `- [variant]     // Variant-specific library files
      |  |     `- pkgconfig  // Variant-specific pkg-config
      |  `- pkgconfig        // Pkg-config links for all variants
      |- sbin                // Installation scripts
      `- share               // Documentation and user files
         |- doc
         |  `- sourceryvsipl++ // PDF and HTML documentation
         `- sourceryvsipl++    // Example programs
```

Sourcery VSIPL++ binary packages contain a single set of shared library headers, and contain multiple library archives that are specialized by processor architecture they support and external libraries they use. These libraries are organized into lib/[arch]/[variant] directories, where [arch] is the processor architecture, and [variant] is the variant. For example, the GNU/Linux x86 binary package supports the ia32 and em64t architectures.

The [variant] subdirectory indicates which external libraries the VSIPL++ library is configured to use, and what level of optimization the library has been built with. For example, the GNU/Linux x86 binary package provides the following 4 variants built with high level of optimization (for release software): par-builtin, which uses LAM/MPI and the builtin ATLAS and FFTW3 libraries, par-intel, which uses LAM/MPI and the Intel IPP and MKL libraries, ser-builtin, which uses just the builtin ATLAS and FFTW3 libraries with no MPI, and ser-intel, which uses just the Intel IPP and MKL libraries with no MPI. In addition, 4 corresponding variants with debug optons: par-builtin-debug, par-intel-debug, ser-builtin-debug, and ser-intel-debug.

2.2.3. Other Install Locations

If you install Sourcery VSIPL++ into a directory other than `opt/sourceryvsipl++-2.2-9`, you must run the **set-prefix.sh** script to update the installation prefixes stored in the library's pkg-config `.pc` files.

The **set-prefix.sh** is located in the `sbin` subdirectory of the tarball.

For example, to install a binary package in `$HOME/sourceryvsipl++-2.2-9`:

Example 2.1. Installing a binary package in $HOME

```
> cd $HOME
> tar xfj sourceryvsipl++-2.2-9.tar.bz2
> $HOME/sourceryvsipl++-2.2-9/sbin/set-prefix.sh
> export PKG_CONFIG_PATH=$HOME/sourceryvsipl++-2.2-9/lib/pkgconfig
```

2.2.4. Paths for External Libraries

Sourcery VSIPL++ binary packages that use the following external libraries have the library installation paths hard-coded in their pkg-config files (install path in parenthesis):

- Intel IPP (`/opt/intel/ipp`).
- Intel MKL (`/opt/intel/mkl`).

If these libraries are not installed in these locations, it is necessary to do one of the following:

- Update the pkg-config file paths using **set-prefix.sh**.
- Create a symbolic link from the default install location to the actual install location.
- Manually specify the paths to the libraries on each invocation of pkg-config.

Each of the options is described in more detail below.

The **set-prefix.sh** script in the `sourceryvsipl++-2.2-9/sbin` will update the pkg-config files with the correct installation prefixes for external libraries. **set-prefix.sh** takes arguments of the form `ipp:/prefix/to/ipp`, `mkl:/prefix/to/mkl`, and `mpi:/prefix/to/mpi`, to specify prefixes for IPP, MKL, and MPICH respectively.

For example, if the library has been installed into `/opt/sourceryvsipl++-2.2-9` and IPP is installed in `/opt/intel/ipp41`:

Example 2.2. Using set-prefix.sh to use IPP from different prefix

```
> /opt/sourceryvsipl++-2.2-9/sbin/set-prefix.sh ipp:/opt/intel/ipp41
```

If multiple prefixes need to be changed, **set-prefix.sh** can either be called once with multiple prefixes:

```
> /opt/sourceryvsipl++-2.2-9/sbin/set-prefix.sh \
        ipp:/opt/intel/ipp41 mkl:/opt/intel/mkl821
```

Or multiple times, once for each prefix:

```
> /opt/sourceryvsipl++-2.2-9/sbin/set-prefix.sh ipp:/opt/intel/ipp41
> /opt/sourceryvsipl++-2.2-9/sbin/set-prefix.sh mkl:/opt/intel/mkl821
```

Using symbolic links, it is possible to direct Sourcery VSIPL++'s expected directory to the actual installation libraries.

For example, if IPP is installed in `/opt/intel/ipp41`:

Example 2.3. Using a symbolic link to use IPP from different prefix

```
> ln -s /opt/intel/ipp41 /opt/intel/ipp
```

Finally, it is possible to manually pass the prefixes for external libraries to pkg-config program on each invocation.

For example, if IPP is installed in `/opt/intel/ipp41` and that MKL is installed in `/opt/intel/mkl821`, to query `--libs` from **pkg-config**:

Example 2.4. Overriding library prefixes from the command line

```
LIBS = `pkg-config \
        --define-variable=ipp_prefix=/usr/local/ipp41  \
        --define-variable=mkl_prefix=/usr/local/mkl821 \
        --libs vsipl++`
```

Chapter 3
Installation from Source

Abstract

This chapter explains how to configure and install Sourcery VSIPL++ from source for use on your system. Installing Sourcery VSIPL++ from source requires three steps:

1. Ensure that your system can run Sourcery VSIPL++.

2. Download the Sourcery VSIPL++ source package.

3. Configure, build, and install Sourcery VSIPL++.

If an appropriate pre-built binary package is available for your architecture, operating system, and compiler, using that is preferrable to building from source. Instructions for this installing pre-built binary packages can be found in Chapter 2, “Installation”.

3.1. System Requirements

This section explains what requirements your system must meet in order to run Sourcery VSIPL++.

3.1.1. Operating System

Sourcery VSIPL++ can be built and installed on any UNIX-like system that has a satisfactory C++ compiler. CodeSourcery's reference GNU/Linux platform is Red Hat Enterprise Linux 4.0. CodeSourcery's reference MCOE platform is 6.3.0.

The following compilers have been tested by CodeSourcery to work with Sourcery VSIPL++ for the noted OS and architecture: OS, architecture, and compiler combinations have been tested by CodeSourcery to work with Sourcery VSIPL++:

- GCC 3.4 (IA32 GNU/Linux)
- GCC 3.4 (EM64t GNU/Linux)
- GCC 4.1 (Cell/B.E. GNU/Linux)
- GCC 4.2 (IA32 GNU/Linux)
- GCC 4.2 (EM64t GNU/Linux)
- Intel C++ 9.1 (IA32 GNU/Linux)
- Intel C++ 9.1 (EM64t GNU/Linux)
- Intel C++ 9.1 (IA32 Windows)
- GreenHills C++ 4.0.6 (PowerPC MCOE 6.3.0)

The following compilers are known not to work at this time with Sourcery VSIPL++:

- GCC 3.3
- Intel C++ 8.1
- Intel C++ 9.0

3.1.2. Required Software

This section describes software that you must have installed in order to build and install VSIPL++. Although the instructions below refer to obtaining this software in source form, you will be able to find pre-compiled binary distributions for most popular operating systems. Consult your operating system manuals for information about obtaining and installing pre-compiled versions of these packages.

3.1.2.1. GNU Make

You must use the GNU version of **make** to build Sourcery VSIPL++. You can download GNU Make as source code from http://ftp.gnu.org/pub/gnu/make. Pre-compiled GNU Make binaries are available for all popular operating systems.

3.1.2.2. GNU Tar

The Sourcery VSIPL++ source code is distributed as a compressed Tape Archive (Tar) file. Due to incompatibilities between various versions of Tar, CodeSourcery strongly recommends that you use GNU Tar to unpack the source code. You can download GNU Tar as source code from http://ftp.gnu.org/pub/gnu/tar. Pre-compiled GNU Tar binaries are available for all popular operating systems.

3.1.3. Optional Software

You can build and use Sourcery VSIPL++ without installing any other software packages. However, if you install the additional packages described in this section, Sourcery VSIPL++ will provide additional functionality and better performance. This section explains what software you might wish to install and how to obtain it. Although some the instructions below refer to obtaining this optional software in source form, you will be able to find pre-compiled binary distributions for most popular operating systems. Consult your operating system manuals for information about obtaining and installing pre-compiled versions of these packages.

3.1.3.1. Numerical Libraries

Sourcery VSIPL++ can take advantage of high-performance numerical libraries to improve performance. This section describes the supported libraries. In general, Sourcery VSIPL++ will automatically make use of these libraries are they are installed on your system.

3.1.3.1.1. ATLAS

Automatically Tuned Linear Algebra Software can be used to accelerate some linear-algebra functions in Sourcery VSIPL++. Sourcery VSIPL++ source packages come with the ATLAS sources. Alternatively, if ATLAS is already installed on your system, Sourcery VSIPL++ can be configured to use it.

Visit http://math-atlas.sourceforge.net for more information about ATLAS.

3.1.3.1.2. FFTW3

The Fastest Fourier Transform in the West can be used to accelerate Sourcery VSIPL++ FFT performance. Sourcery VSIPL++ source packages include the FFTW3 sources. Alternatively, if FFTW3 is already installed on your system, Sourcery VSIPL++ can be configured to use it.

Visit http://www.fftw.org for more information about FFTW.

3.1.3.1.3. Intel IPP and MKL

The Intel Performance Primitives and Intel Math Kernel Library can be used by Sourcery VSIPL++ to accelerate some functions, including FFTs. IPP and MKL are proprietary libraries, so you cannot distribute a Sourcery VSIPL++ application using IPP or MKL under the terms of the GPL.

To find out more about IPP and MKL visit

- http://www.intel.com/cd/software/products/asmo-na/eng/perflib/ipp/index.htm
- http://www.intel.com/cd/software/products/asmo-na/eng/perflib/mkl/index.htm

3.1.3.1.4. AMD Core Math Library (ACML)

The AMD Core Math Library (ACML) can be used by Sourcery VSIPL++ to accelerate some linear-algebra functions.

Visit http://developer.amd.com/acml.aspx for more information about ACML.

3.1.3.1.5. Mercury SAL

The Mercury Scientific Algorithm Library (SAL) can be used by Sourcery VSIPL++ to accelerate many functions, including elementwise view operations, linear algebra, solvers, and signal processing objects (including FFT). SAL is a proprietary library, so you cannot distribute a Sourcery VSIPL++ application using SAL under the terms of the GPL.

Visit http://www.mc.com/products/software.aspx for more information about SAL.

3.1.3.1.6. VSIPL Back End

An implementation of the C VSIPL API can be used by Sourcery VSIPL++ to implement many functions, including linear algebra, solvers, and signal processing objects (such as FFT).

Visit the http://www.vsipl.org/ for more information about the VSIPL API and a list of implementations.

3.1.3.1.7. CML Back End

The Cell Math Library (CML) is used by Sourcery VSIPL++ to accelerate performance on Cell processors by offloading many computations to the SPUs. *CML is required to build any version of Sourcery VSIPL++ from source.* CML is available from CodeSourcery and included in the binary installation packages for the full version of Sourcery VSIPL++.

Contact `<sales@codesourcery.com>` for more information about CML.

3.1.3.2. Communications Libraries

If you install a communication library such as Mercury PAS (Parallel Acceleration System) or MPI (Message Passing Interface), you can run Sourcery VSIPL++ programs on multiple compute nodes simultaneously. On GNU/Linux platforms, Sourcery VSIPL++ works with both the LAM and MPICH implementations of MPI, and will likely work with other MPI implementations as well. On MCOE platforms, Sourcery VSIPL++ works with both Mercury PAS and Verari MPI/Pro.

3.1.3.2.1. Mercury PAS

Mercury Parallel Acceleration System (PAS) is a library for high-performance communication on Mercury PowerPC embedded computer and Cell blade systems. PAS is a propreitary library, so you cannot distribute a Sourcery VSIPL++ application using PAS under the terms of the GPL.

For more information on PAS, visit http://www.mc.com/products/software.aspx.

The following releases of Mercury PAS have been tested by CodeSourcery to work with Sourcery VSIPL++:

- MCOE PAS 4.3.0
- PAS for Linux Clusters

3.1.3.2.2. LAM/MPI

You can download LAM/MPI as source code from http://www.lam-mpi.org/.

Recommended configuration options when building LAM/MPI for use with Sourcery VSIPL++:

`--without-mpi2cpp` Do not build MPI/C++ interface. Although Sourcery VSIPL++ does not use this interface, building it may create link-time warnings when using different compiler versions.

The following releases of LAM/MPI have been tested by CodeSourcery to work with Sourcery VSIPL++:

- LAM/MPI 7.0.6
- LAM/MPI 7.1.1

> **RHEL Users**
>
> The RHEL LAM 6.5.9-1 package available for RHEL 3 is not suitable for use with Sourcery VSIPL++. This package does not support compilation with a C++ compiler.

3.1.3.2.3. MPICH

You can download MPICH as source code from http://www-unix.mcs.anl.gov/mpi/mpich/, but pre-built binaries for most popular operating systems are available from the system distributors. If MPICH is not already installed on your system, see the documentation for your operating system for information about obtaining MPICH.

3.1.3.2.4. OpenMPI

You can download OpenMPI as source code from http://www.open-mpi.org/, but pre-built binaries for most popular operating systems are available from the system distributors. If OpenMPI is not already installed on your system, see the documentation for your operating system for information about obtaining OpenMPI.

3.1.3.2.5. Verari MPI/Pro

The following release of Verari MPI/Pro has been tested by CodeSourcery to work with Sourcery VSIPL++:

- Verari MPI/Pro 2.1.0

3.2. Obtaining the Source Code

The Sourcery VSIPL++ Source Code is available from CodeSourcery's web site. Visit http://www.codesourcery.com/vsiplplusplus/download.html for instructions on downloading VSIPL++.

Sourcery VSIPL++ source packages are distributed as compressed Tape Archive (Tar) files. Use GNU Tar to unpack the source code with the following command:

```
> tar xjf sourceryvsipl++-2.2-9.tar.bz2
```

This command will create a subdirectory of the current directory called `sourceryvsipl++-2.2-9`.

3.3. Configuration

Before building Sourcery VSIPL++, you must run a configuration script to tell Sourcery VSIPL++ what C++ compiler you are using and what optional software you wish to use. After running the configuration script, you will build and install the Sourcery VSIPL++ library.

These instructions assume that your shell's current directory is the `sourceryvsipl++-2.2-9` directory created when you unpacked the VSIPL++ distribution. If you want to allow Sourcery VSIPL++ to automatically configure itself, run:

```
> ./configure
```

You will see output explaining the configuration decisions that Sourcery VSIPL++ is making.

There are several options that you can use to tell Sourcery VSIPL++ about your particular environment.

`CXX=`*`path`*
: Use *`path`* as the C++ compiler. If you do not provide this option, Sourcery VSIPL++ will search for a C++ compiler in your `PATH`.

`CXXFLAGS=`*`flags`*
: Use *`flags`* as flags to pass to the C++ compiler. The default value depends on your compiler. If you are using multiple flags (like `-O2 -ffast-math`), you must enclose the *`flags`* in quotes so that the shell will consider all of the flags as a single argument.

`--prefix=`*`directory`*
: Install the library in *`directory`*. Header files will be placed in a subdirectory of *`directory`* named `include`; the library itself will be placed in `lib`. You will need to have sufficient permissions to write to the installation directory. The default installation directory is `/usr/local`, which is usually not writable by non-administrators; therefore, you may want to use your home directory as an installation directory.

`--host=`*`architecture`*
: Specify the host-architecture that Sourcery VSIPL++ will be built for. The default is to build Sourcery VSIPL++ to run native on build machine. This option is useful when cross-compiling Sourcery VSIPL++.

`--disable-parallel`
: Do not use a parallel communications library, even if an appropriate MPI library is detected. This option is useful if you want to build a uniprocessor version of Sourcery VSIPL++. By default, MPI support will be included if it is available.

`--enable-parallel`
: Search for and use a communications library for support of multi-processor systems for parallel computation.

`--enable-parallel=`*`lib`*
: Search for and use the parallel communications library indicated by *`lib`*. Available options are `lam`, `mpich2`, `intelmpi`, `openmpi`, `mpipro`, and `pas`.

  `lam` selects the LAM/MPI library.

  `mpich2` selects the MPICH2 library.

	`intelmpi` selects the Intel MPI Library. `openmpi` selects then Open MPI library. `mpipro` selects Verari's MPI/Pro. This option is necessary when using MPI/Pro on the Mercury platform. `pas` enables the use of Mercury Parallel Acceleration System (PAS) for parallel services if found. This option is necessary to use PAS on the Mercury platform, and when using PAS for Linux clusters.
`--with-mpi-prefix=`*`directory`*	Search for MPI installation in *`directory`* first. MPI headers should be in *`directory/include`*, MPI libraries in *`directory/lib`*, and MPI compilation commands (either `mpicxx` or `mpiCC`) should be in *`directory/bin`*. This option is useful if MPI is installed in a non-standard location, or if multiple MPI versions are installed.
`--with-mpi-cxxflags=`*`flags`* `--with-mpi-libs=`*`flags`*	In some cases, Sourcery VSIPL++ is unable to automatically detect the required compiler and linker options to enable MPI. In these cases, the required C++ compiler flags can be specified using the `--with-mpi-cxxflags` option, and the required linker (library) flags can be specified using the `--with-mpi-libs`. These options must be used together, and when they are used, the specific type of the MPI library in use must be specified with the `--enable-parallel=`*`type`* option.
`--disable-exceptions`	Do not use C++ exceptions. Errors that would previously have generated an exception now cause an abort(). This option is useful if you want to build Sourcery VSIPL++ with a compiler that does not implement exceptions. By default, exceptions are used.
`--with-ipp`	Enable the use of the Intel Performance Primitives (IPP) if found. Enabling IPP will accelerate the performance of signal processing and view element-wise operations.
`--with-ipp=win`	Enable the use of the Intel Performance Primitives (IPP) for Windows if found. This option is useful when configuring Sourcery VSIPL++ on a Windows system.
`--with-ipp-prefix=`*`directory`*	Search for IPP installation in *`directory`* first. IPP headers should be in the `include` subdirectory of *`directory`* and IPP libraries should be in the `lib` subdirectory. This option has the effect of enabling IPP (i.e. `--with-ipp`). This option is useful if IPP is installed in a non-standard location, or if multiple IPP versions are installed.
`--with-ipp-suffix=`*`suffix`*	Use a processor specific version of the IPP libraries, as indicated by *`suffix`*. For example, the suffix em64t will select IPP libraries specific to em64t processors. By default, non-suffix IPP libraries are used, which determine the architecture at run-time and dynamically load the appropriate processor-

	specific libraries. This option is useful if the automatic dispatcher is not able to determine the correct architecture.
`--with-sal`	Enable the use of the Mercury Scientific Algorithm Library (SAL) if found. Enabling SAL will accelerate the performance of view element-wise operations, linear algebra, solvers, and signal processing operations.
`--with-sal-include=`*`directory`*	Search for SAL header files in *`directory`* first. This option has the effect of enabling SAL (i.e. `--with-sal`). This option is useful if SAL headers is installed in a non-standard location, such as when using the CSAL library. However, it should not be necessary when building native on Mercury system.
`--with-sal-lib=`*`directory`*	Search for SAL library files in *`directory`* first. This option has the effect of enabling SAL (i.e. `--with-sal`). This option is useful if SAL libraries is installed in a non-standard location, such as when using the CSAL library. However, it should not be necessary when building native on Mercury system.
`--with-cuda`	Enable the use of NVidia's Compute Unified Device Architecture (CUDA). This enables the use of certain graphics processing units (GPUs) as computational accelerators (see NVidia's website for a list of compatible cards). For FFT support, use `--enable-fft=cuda` in addition to this option.
`--enable-fft=`*`lib`*	Search for and use the FFT library indicated by *`lib`* to perform FFTs. Valid choices for *`lib`* include `fftw3`, `cuda`, `ipp`, `sal`, and `cvsip` which select FFTW3, CUDA, IPP, SAL, and C VSIPL libraries respectively. A fourth option, `builtin`, selects the FFTW3 library that comes with Sourcery VSIPL++ (default). This option should be used if an existing FFTW3 library is not available. If no FFT library is to be used (disabling Sourcery VSIPL++'s FFT functionality), `no_fft` should be chosen for *`lib`*. Multiple libraries may be given as a comma separated list. When performing an FFT, VSIPL++ will use the first library in the list that can support the FFT parameters. For example, on Mercury systems `--enable-fft=sal,builtin` would use SAL's FFT when possible, falling back to VSIPL++'s builtin FFTW3 otherwise.
`--with-fftw3-prefix=`*`directory`*	Search for FFTW3 installation in *`directory`* first. FFTW3 headers should be in the `include` subdirectory of *`directory`* and FFTW3 libraries should be in the `lib` subdirectory. This option has the effect of enabling FFTW3 for FFTs (i.e. `--with-fft=fftw3`). This option is useful if FFTW3 is installed in a non-standard location, or if multiple FFTW3 versions are installed.
`--disable-fftw3-simd`	Disable builtin FFTW3 from using SIMD ISA extensions (such as AltiVec or SSE2). By default, FFTW3 uses SIMD

	ISA extensions because they improve performance. However, this option is useful when building for a platform that does not support the ISA extensions.
`--with-lapack`	Enable Sourcery VSIPL++ to search for an appropriate LAPACK implementation on the platform. If found, it will be used to perform linear algebra (matrix-vector products and solvers).
`--with-lapack=`*`lib`*	Search for and use the LAPACK library indicated by *`lib`* to perform linear algebra (matrix-vector products and solvers). Valid choices for *`lib`* include `mkl`, `acml`, `atlas`, `generic`, `builtin`, and `no`. `mkl` selects the Intel Math Kernel Library (MKL) to perform linear algebra if found. `mkl_win` selects the Intel Math Kernel Library (MKL) on Windows systems to perform linear algebra if found. `acml` selects the AMD Core Math Library (ACML) to perform linear algebra if found. `atlas` selects the ATLAS library to perform linear algebra if found. `generic` selects a generic LAPACK library (-llapack) to perform linear algebra if found. `builtin` selects a version of LAPACK that doesn't require ATLAS. `no` is used to disable searching for a LAPACK library.
`--with-acml-prefix=`*`directory`*	Search for ACML installation in *`directory`* first. ACML headers should be in the `include` subdirectory of the install directory, whose path depends on the exact version of the library you have. Similarly, ACML libraries should be in the `lib` subdirectory. This option has the effect of enabling ACML for lapack (i.e. `--with-lapack=acml`). This option is useful if the ACML is installed in a non-standard location, or if multiple ACML versions are installed.
`--with-atlas-prefix=`*`directory`*	Search for ATLAS installation in *`directory`* first. ATLAS headers should be in the `include` subdirectory of *`directory`* and ATLAS libraries should be in the `lib` subdirectory, unless otherwise specified by `--with-atlas-include` and `--with-atlas-libdir`, respectively. This option has the effect of enabling ATLAS for lapack (i.e. `--with-lapack=atlas`). This option is useful if ATLAS is installed in a non-standard location, or if multiple ATLAS versions are installed.
`--with-atlas-include=`*`directory`*	Search for ATLAS include headers in *`directory`* first. This option has the effect of enabling ATLAS for lapack (i.e.

	`--with-lapack=atlas`). This option is useful if ATLAS is installed in a location that does not fit the pattern assumed by `--with-atlas-prefix`.
`--with-atlas-libdir=`*`directory`*	Search for ATLAS library files in *`directory`* first. This option has the effect of enabling ATLAS for lapack (i.e. `--with-lapack=atlas`). This option is useful if ATLAS is installed in a location that does not fit the pattern assumed by `--with-atlas-prefix`.
`--with-mkl-prefix=`*`directory`*	Search for MKL installation in *`directory`* first. MKL headers should be in the `include` subdirectory of *`directory`* and MKL libraries should be in the `lib/(arch)` subdirectory. This option has the effect of enabling MKL for lapack (i.e. `--with-lapack=mkl`). This option is useful if MKL is installed in a non-standard location, or if multiple MKL versions are installed.
`--with-mkl-arch=`*`architecture`*	Used in conjunction with `--with-mkl-prefix` to specify which library subdirectory of MKL to use. If `--with-mkl-prefix=`*`directory`* is used to specify the MKL prefix, libraries are searched for in `directory/architecture`. By default *`architecture`* is deduced based on the platform. This option is useful if this deduction is incorrect.
`--without-cblas`	Disables the use of the C BLAS API, forcing the use of the Fortran BLAS API. This option is useful if building on a platform that does not provide the C BLAS API.
`--with-cbe-sdk`	Enable the use of the IBM Cell/B.E. Software Development Kit (SDK) version 3.0 or 3.1 if found. Enabling the Cell/B.E. SDK will accelerate the performance of FFTs, vector-multiplication, vector-matrix multiplication, and fast convolution.
`--with-cbe-sdk-sysroot=`*`directory`*	Search for Cell/B.E. SDK libraries and headers in a sysroot at *`directory`*, rather than in the system root directory (or the default sysroot location, in the case of SDK version 2.1). This option has the effect of enabling use of the Cell/B.E. SDK (i.e. `--with-cbe-sdk`). This option is used for cross-compilation.
`--with-numa`	Enable the use of libnuma. This is useful on Cell/B.E. systems to insure that SPE resources allocated for accelertion are local to the PPE running VSIPL++.
`--with-cvsip`	Enable Sourcery VSIPL++ to search for an appropriate C VSIPL implementation on the platform. If found, it will be used to perform linear algebra (matrix-vector products and solvers) and some signal processing (convolution, correlation, and FIR). If the `--enable-fft=cvsip` option is also given, the VSIPL implementation will be used to perform FFTs.

`--with-cvsip-prefix=`*`directory`*
: Search for a C VSIPL installation in *`directory`* first. Headers should be in the `include` subdirectory of *`directory`* and libraries should be in the `lib` subdirectory. This option has the effect of enabling the use of a VSIPL back end as if the option `--with-cvsip` had been given. This option is useful if VSIPL is installed in a non-standard location, or if multiple VSIPL versions are installed.

`--enable-only-ref-impl`
: Configure Sourcery VSIPL++ to be used as the VSIPL++ reference implementation. When the BSD licensed files are configured with this option, the result is the VSIPL++ reference implementation. This option implies the `--enable-fft=cvsip` and `--with-cvsip` options. Refer to Section 3.3.4, “Configuration Notes for the Reference Implementation” for more information on configuring the reference implementation.

`--with-png`
: Enables PNG I/O support, using libpng. By default, PNG support is enabled if libpng is found during configuration.

`--enable-simd-loop-fusion`
: Enable VSIPL++ to generate SIMD instructions for loop-fusion expressions (containing data that is SIMD aligned). This option is useful for increasing performance of many VSIPL++ expressions on platforms with SIMD instruction set extensions (such as Intel SSE, or Power VMX/AltiVec). The default is not to generate SIMD instructions.

`--enable-simd-unaligned-loop-fusion`
: Enable VSIPL++ to generate SIMD instructions for loop-fusion expressions, possibly containing data that is SIMD unaligned. This option is useful for increasing performance of VSIPL++ expressions that work with unaligned data on platforms with SIMD instruction set extensions (such as Intel SSE, or Power VMX/AltiVec). The default is to follow the setting of `--enable-simd-loop-fusion`.

`--with-complex=`*`format`*
: Specify the *`format`* for storing complex numbers. Valid choices for *`format`* are `inter` and `split`, which select interleaved and split storage respectively. This option is useful when a platform has better performance using a particular complex storage format. The default complex storage format is `inter`.

`--enable-timer=`*`timer`*
: Use *`timer`* type of timer for profiling. Valid choices for *`timer`* include `none`, `posix`, `realtime`, `pentiumtsc`, and `x86_64_tsc`, and `power_tb`. By default no timer is used (*`timer`*`=none` This option is necessary when you intent to use the libary's profiling or performance API features.

  `none` disables profile timing.

  `posix` selects the POSIX timer if present on the system.

  `realtime` selects the POSIX realtime timer if present on the system.

	`pentiumtsc` selects the Pentium time-stamp counter (TSC) timer if present on the system.
	`x86_64_tsc` selects the x86-64 (or em64t) time-stamp counter (TSC) timer if present on the system.
	`power_tb` selects the Power architecture timebase counter timer if present on the system.
`--enable-cpu-mhz=`*`speed`*	Use *`speed`* MHz as the counter frequency for the Pentium and x86-64 timestamp counters. By default, the counter frequency is queried from the operating system at runtime. This option is useful if the correct counter frequency cannot be determined.
`--with-obj-ext=`*`EXT`*	Specify *`EXT`* as the file extension to be used for object files. Object files will be named `file.`*`EXT`*. Default value is determined heuristically by configure.
`--with-lib-ext=`*`EXT`*	Specify *`EXT`* as the file extension to be used for library archive files. Library archive files will be named `file.`*`EXT`*. Default value is determined heuristically by configure.
`--with-exe-ext=`*`EXT`*	Specify *`EXT`* as the file extension to be used for executable files. Executable files will be named `file`*`EXT`*. Unlike `--with-obj-ext` and `--with-lib-ext`, no "." is implied. Default value is determined heuristically by configure.
`--enable-shared-acconfig`	Generate an acconfig.hpp that can be shared by different configurations by putting macros on the compiler command line. This is useful when building binary packages. Normally an acconfig.hpp file is generated that can only be used by one configuration.
`--enable-shared-libs`	Build shared libraries as well as static libraries. This requires that position-independent code be generated, which may reduce performance.

Example 3.1, “Configuring Sourcery VSIPL++” shows how to use the configure script to use particular optimization options for the C++ compiler on a system where MPI support is not required. The exact output will vary from system to system, but the output shown here is representative.

Example 3.1. Configuring Sourcery VSIPL++

```
> ./configure CXXFLAGS="-O2 -ffast-math" --disable-mpi
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking for g++... g++
checking for C++ compiler default output file name... a.out
checking whether the C++ compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking for bugs in g++ and its runtime... no bugs found
checking for openjade... openjade
checking for pdfjadetex... pdfjadetex
checking for a BSD-compatible install... /usr/bin/install -c
configure: creating ./config.status
config.status: creating src/vsip/impl/acconfig.hpp
config.status: creating src/vsip/GNUmakefile.inc
config.status: creating tests/context
config.status: creating tests/QMTest/configuration
config.status: creating doc/GNUmakefile.inc
config.status: creating GNUmakefile
config.status: creating src/vsip/impl/acconfig.hpp
```

3.3.1. Configuration Notes for Mercury Systems

When configuring Sourcery VSIPL++ for a Mercury PowerPC system, the following environment variables and configuration flags are recommended:

- `CXX=ccmc++`

  This selects the `ccmc++` cross compiler as the C++ compiler.

- `CC=ccmc`

  This selects the `ccmc` cross compiler as the C compiler.

- `AR=armc`

  This selects the `armc` archiver.

- `AR_FLAGS=cr`

  This selects the `c` (create archive if it does not exist) and `r` (replace files in archive) flags for the `armc` archiver. `armc` does not support the `u` flag (only replace files if they are an update).

- `CXXFLAGS="--no_implicit_include -Onotailrecursion -t` *`architecture`* `--no_exceptions -Ospeed --max_inlining -DNDEBUG --diag_suppress 177,550"`

  These are the recommended flags for compiling Sourcery VSIPL++ with the GreenHills C++ compiler on the Mercury platform. These flags fall into two categories: those necessary for a correct build, and those optional for good performance. The following are necessary to correctly build the library:

- `--no_implicit_include`

  GreenHills enables implicit inclusion by default. This permits the compiler to assume that if it needs to instantiate a template entity defined in a .hpp file it can implicitly include the corresponding .cpp file to get the source code for the definition.

  Sourcery VSIPL++ does not use this capability. Leaving this feature enabled will result in multiple symbol definition errors at link-time.

  Note: it is only necessary to disable implicit includes when building the library. After the library has been installed, applications using it may enable implicit includes.

- `-Onotailrecursion`

  This disables optimization of tail-recursive functions. This optimization has a defect which is triggered by some of Sourcery VSIPL++'s algorithms.

The following flags will improve the performance of the library and applications. These should be used for production.

- `-t` *`architecture`*

  This flag directs the compiler to generate code optimized for processor variant and endian-ness specified by *`architecture`*. Valid choices are listed in the `ccmc++` documentation and include `ppc7400`, `ppc7400_le`, `ppc7445`, and `ppc7445_le`.

- `--no_exceptions`

  Disable exception handling, which can have a large performance overhead with the GreenHills compiler. This should be used in conjunction with the configure flag `--disable-exceptions`.

- `-Ospeed`

  This option instructs the compiler to enable all optimizations which improve speed.

- `--max_inlining`

  By default, GreenHills will only consider functions composed entirely of straightline code (no control flow) for inlining. `--max_inlining` instructs the compiler to consider all functions (whether containing control flow statements or not) for inlining, subject to the usual restraints in the case of excessively large or complicated functions.

- `-DNDEBUG`

  Disable assertions. This option should be used when configuring the library for performance.

- `--diag_suppress 177,550`

  This option suppresses compiler diagnostics warning about unused variables. When compiling with `-DNDEBUG` assertions are removed that may be the only reference to a variable.

When compiling a development or debug version of the library, replace `-Ospeed -DNDEBUG` with `-g`.

- `--host=powerpc`

  Cross compile for the PowerPC processor.

- `--with-sal`

  Enable the SAL library.

- `--enable-fft=sal,builtin`

  Use SAL and Sourcery VSIPL++ builtin FFTW3 to perform FFT operations. SAL FFT will be used for FFTs with power-of-two sizes, FFTW3 will be used otherwise.

- `--with-fftw3-cflags="-O2"`

  Compile Sourcery VSIPL++'s builtin FFTW3 library with optimization level `-O2`. (Compiling FFTW3 with optimization level `-O3` produces link-errors with GreenHills C related to the handling of static functions. CodeSourcery is currently developing a work-around for this.)

- `--with-complex=split`

  Store complex data in split format by default.

- `--disable-exceptions`

  Disable the use of exceptions from within the library.

- `--enable-parallel=mpipro`

  Enable the use of Verari MPI/Pro for communications.

- `--enable-timer=realtime`

  Use the POSIX-realtime timer for profiling.

The file `examples/mercury/mcoe-setup.sh` is an example of how to configure Sourcery VSIPL++ for the Mercury with these options.

3.3.2. Configuration Notes for Windows Systems

Before configuring Sourcery VSIPL++ for a Microsoft Windows systems, the follow prerequisites are recommended:

- The Cygwin environment for Windows, including the GNU make and sed packages. Sourcery VSIPL++ uses this as development environment for configuring and building the Sourcery VSIPL++ library. Cygwin is not necessary to build and run Sourcery VSIPL++ applications. For more information on the Cygwin environment, visit http://www.cygwin.com/

- Intel C++ for Windows, version 9.1 or later. This may require installation of a Microsoft C++ compiler and Microsoft SDK for windows. For more information on Intel C++ and its requirements: http://www.intel.com/cd/software/products/asmo-na/eng/compilers/279578.htm

- Intel IPP and MKL for Windows.

When configuring Sourcery VSIPL++ for a Microsoft Windows system, the following environment variables and configuration flags are recommended:

- `CXX=icl`

  This selects the Intel C/C++ compiler **icl** as the C++ compiler.

- `CC=icl`

  This selects the Intel C/C++ compiler `icl` at the C compiler.

- `CXXFLAGS="/Qcxx-features /Qvc8"`

  These are the recommended flags for compiling Sourcery VSIPL++ with the Intel C++ compiler on Microsoft Windows platforms. The following are necessary to correctly build the library:

  - `/Qcxx-features`

    This enables standard C++ features for exception handling and RTTI.

  - `/Qvc8`

    This enables Microsoft Visual Studio 2005 compatibility. If using another version of Visual Studio, please consult the Intel C++ documentation for the correct option.

- `--build=i686-cygwin`

  Configure to build library in the cygwin environment.

- `--host=i686-mingw32`

  Target the resulting library to run on Microsoft Windows systems with the Win32 API.

- `--with-ipp=win`

  Enable the IPP library for Windows. This requires that the IPP header, library, and DLL directories be present in your `INCLUDE`, `LIB`, and `PATH` directories, respectively. Manually passing these paths to **configure** in Windows is not recommended.

- `--enable-fft=ipp`

  Use the IPP FFT functions to perform FFT operations.

- `--with-lapack=mkl_win`

  Use the MKL library for Windows to implement linear-algebra operations. This requires that the MKL header and library directories be present in your `INCLUDE` and `LIB`, directories, respectively. Manually passing these paths to **configure** in Windows is not recommended.

- `--disable-parallel`

  Disable parallel service. Sourcery VSIPL++ does not support MPI on Windows at this time.

3.3.3. Configuration Notes for Cell/B.E. Systems

When configuring Sourcery VSIPL++ for a Cell/B.E. host system, the following environment variables and configuration flags are recommended:

- `--with-cbe-sdk`

Enable use of the Cell/B.E. SDK and the Cell Math Library (CML). This is necessary to use the Cell/B.E.'s SPE processors to accelerate VSIPL++ functionality. If the SDK is not installed in the standard location, the `--with-cbe-sdk-prefix` should be used to specify the location.

- `--with-cml-prefix=PATH`

  Specify the installation path of CML. Headers are installed in a subdirectory named include; libraries in one named lib.

  To install headers and libraries in other places, use instead the options `--with-cml-include` and `--with-cml-libdir`.

- `--with-cml-include=PATH`

  Specify the directory containing CML header files. Use this option in conjunction with `--with-cml-libdir`. Do not use with `--with-cml-prefix`.

- `--with-cml-libdir=PATH`

  Specify the directory containing CML libraries. Use this option in conjunction with `--with-cml-include`. Do not use with `--with-cml-prefix`.

- `--with-numa`

  Enable use of libnuma for SPE/PPE affinity control. This may improve program performance by allocating SPEs close to the PPEs running VSIPL++.

- `--enable-timer=power_tb`

  Enable the Power Timebase high-resolution timer. This option is useful when using profiling or running library benchmarks.

Two additional options must be specified when using a non-Cell/B.E. build system to cross-compile Sourcery VSIPL++ for a Cell/B.E. host system.

- `--host=powerpc-cell-linux-gnu`

  Define the host system type.

- `--with-cbe-sdk-sysroot=`*`directory`*

  Specify the Cell/B.E. sysroot location. Typically, this will be `/opt/cell/sysroot` on a standard SDK 3.0 cross-compiler installation.

3.3.4. Configuration Notes for the Reference Implementation

If you wish to use the BSD-licensed reference-implementation subset of Sourcery VSIPL++, you must configure with the following option:

- `--enable-only-ref-impl`

  Build only the reference-implementation subset of Sourcery VSIPL++. If you do not use this option, the complete, optimized implementation of Sourcery VSIPL++ will be built.

3.4. Compilation and Installation

After you have configured Sourcery VSIPL++, build the library by running the following command:

```
> make
```

The command shown above assumes that GNU make is in your `PATH` and has been installed as **make**. On some systems, GNU make may be installed as **gmake**; if so, you will have to adjust the command shown above appropriately. After building the library, use the following command to install Sourcery VSIPL++:

```
> make install
```

3.5. Building PDF Documentation

The PDF documentation for VSIPL++ is provided in a pre-built form within the source package, and normally the build process simply copies this pre-built documentation into the build tree.

If for some reason you wish to rebuild the PDF documentation, you need the following additional software:

- **libxslt**

  Sourcery VSIPL++ uses the **xsltproc** program from libxslt to format the DocBook source files for the documentation. You can download libxslt from http://xmlsoft.org/XSLT/index.html if your operating system does not already provide a copy.

- **XEP**

  Sourcery VSIPL++ uses the RenderX XEP Engine to process the formatted DocBook files into PDF files. XEP is available from http://www.renderx.com/tools/xep.html.

In addition, the following configuration options are applicable to building the PDF documentation:

`--enable-maintainer-mode`
: This option allows Sourcery VSIPL++ to rebuild various generated files that are normally included in pre-built form in the source distributions. You must pass this option to the configure script in order to rebuild the PDF documentation.

`XEP=`*`path`*
: Specify *`path`* as the XEP render engine to use. If you do not provide this option, Sourcery VSIPL++ searches for `xep` in your `PATH`.

After you have configured Sourcery VSIPL++ appropriately, the PDF documentation is built automatically when you **make** the rest of the library, or you can rebuild just the PDF documenation by running the command

```
> make pdf
```

Chapter 4
Building Applications

Abstract

Sourcery VSIPL++ comes with example programs, installed in the `share/sourceryvsipl++` subdirectory. This chapter explains how to compile, link, and run these programs. You can modify these programs to develop your own Sourcery VSIPL++ applications.

This chapter assumes that you have installed Sourcery VSIPL++ in `/opt/sourceryvsipl++-2.2-9`. If you have used a different path, you will have to adjust the filenames below accordingly. It is also assumed that the current directory is writable by you. For example, you can use your home directory or `/tmp` as the current directory. Finally, the examples in this chapter assume that you are using the GNU C++ compiler. If you are using another C++ compiler, you may have to make minor changes to the commands shown.

4.1. Building Manually

The file `/opt/sourceryvsipl++-2.2-9/share/sourceryvsipl++/example1.cpp` contains a very simple VSIPL++ program. You can use this file as a template for developing much more complex programs.

When building Sourcery VSIPL++ applications, you must ensure that your compiler can find the necessary header and library files. Since Sourcery VSIPL++ may depend on other libraries, the easiest way to determine the necessary compiler directives is with the **pkg-config** command.

Before **pkg-config** can find information about Sourcery VSIPL++, it is necessary to make sure that Sourcery VSIPL++'s `lib/pkgconfig` subdirectory is in **pkg-config**'s search path. You can check the search path by examining the `PKG_CONFIG_PATH` environment variable. To set the path:

```
> export PKG_CONFIG_PATH=/opt/sourceryvsipl++-2.2-9/lib/pkgconfig
```

First, determine what compiler is recommended:

```
> CXX=`pkg-config vsipl++ --variable=cxx`
```

Second, to compile the program, use the following command:

```
> $CXX -c `pkg-config vsipl++ --cflags` \
                        `pkg-config vsipl++ --variable=cxxflags` \
     /opt/sourceryvsipl++-2.2-9/share/sourceryvsipl++/example1.cpp
```

Finally, to link the program, use the following command:

```
> $CXX -o example1 example1.o `pkg-config --libs vsipl++`
```

Now that you have built the example program, you can run it like any other program, with:

```
> ./example1
```

4.1.1. Using pkg-config

When building applications, it is important to use the same C++ compiler that was used to build the Sourcery VSIPL++ library. Different C++ compilers, even different versions of the same compiler, may have incompatible linking conventions or different standard library implementations. However, it is possible to determine the compiler used to build Sourcery VSIPL++ via pkg-config:

```
> pkg-config --variable=cxx vsipl++
```

Using this, the previous commands to compile and link the example program become:

```
> `pkg-config --variable=cxx vsipl++` \
          -c `pkg-config --cflags vsipl++`
   /opt/sourceryvsipl++-2.2-9/share/sourceryvsipl++/example1.cpp
```

```
> `pkg-config --variable=cxx vsipl++` \
          -o example1 example1.o `pkg-config --libs vsipl++`
```

If pkg-config is not available on your system, you can specify the search paths manually. With most compilers, the `-I` switch can be used to specify directories containing header files. Use the following command to compile the program:

```
> g++ -c -I /opt/vsip/include \
     /opt/sourceryvsipl++-2.2-9/share/sourceryvsipl++/example1.cpp
```

To link the program manually, you must tell the compiler where to find the libraries when linking. For most compilers, the `-L` switch is used to specify directories to search for libraries, while the `-l` switch is used to specify the names of libraries to use. Use the following command to link the program:

```
> g++ -o example1 -L /opt/sourceryvsipl++-2.2-9/lib example1.o -l \
vsip
```

If Sourcery VSIPL++ was configured to use other libraries, such as MPI, it will be necessary to manually specify `-L` and `-l` options accordingly. These necessary options can be determined by looking in the `/opt/sourceryvsipl++-2.2-9/lib/pkgconfig/vsipl++.pc` file. It contains a line prefixed with "Libs:" which indicates the libraries necessary to link a Sourcery VSIPL++ program.

4.2. Building with GNU Make

A makefile for compiling the example programs is provided in the same directory as the example source files (`/opt/vsip/share/sourceryvsipl++`). This example uses the pkg-config command to extract the information needed to correctly compile an application with the Sourcery VSIPL++ library. Create the executable by entering the **make** command from the examples directory (or directory where you have copied the makefile and source files). For example:

```
> cd /opt/sourceryvsipl++-2.2-9/share/sourceryvsipl++/
> make
g++ -I/opt/sourceryvsipl++-2.2-9/include -o example1 example1.cpp \
-L/opt/sourceryvsipl++-2.2-9/lib
  -lsvpp
```

Note that your command line will appear differently depending on install location and configuration options. As described above, the correct compiler, include paths and library information are obtained automatically.

4.3. Building with Microsoft Visual Studio

An example solution is provided in the `vcproj` sub-directory of the directory containing the example source files (`/opt/vsip/share/sourceryvsipl++`). It uses the Intel C++ compiler.

The Windows binary package uses Intel's IPP and MKL libraries, so you need to make sure their headers and libraries can be found by Microsoft Visual Studio, either by adding the appropriate search paths to the project files, or by adding them as global options to your Microsoft Visual Studio configuration (`Tools` → `Options` → `Projects and Solutions` → `VC++ Directories`).

4.4. Running Serial Applications

Serial VSIPL++ applications are run like other serial programs on your system. No special command-line arguments are required.

On Cell/B.E. it is necessary to set the ALF_LIBRARY_PATH environment variable before running Sourcery VSIPL++ aplications. This variable should contain the name of the library directory containing the files 'cml_kernels.so' and 'svpp_kernels.so.' This can be queried from pkg-config:

```
> export ALF_LIBRARY_PATH='pkg-config vsipl++ --variable libdir'
```

On most GNU/Linux systems, serial applications are run by typing their name at the command prompt:

```
> ./example1
```

On most Mercury MCOE systems, serial applications are run using the **sysmc** and **runmc** commands:

```
> sysmc -ce CE -bcs=0 init
> runmc -ce CE ./example1.ppc
> sysmc -ce CE reset
```

For more details, refer to your system documentation.

4.5. Running Parallel Applications

Parallel VSIPL++ applications are run depends on the particular communication library used.

For most MPI libraries, parallel applications are run with the **mpirun** command. For example, to run a VSIPL++ application on two processors:

```
> mpirun -np 2 ./example1
```

This may require that your MPI library has been first initialized on the system by starting a daemon process. Please consult your MPI library documentation for more details.

For PAS, it is necessary to pass extra command line options to VSIPL++. `-pas_size` *`nproc`*`>` indicates the total number of processors in the PAS set. `-pas_rank` *`rank`* indicates the rank of the particular process in the set.

For example, to run a parallel VSIPL++ application with MCOE PAS with four CEs, 2, 3, 4 and 5:

```
> sysmc -ce 2 -bcs=0 init 3-5
> runmc -ce 3 ./example1.ppc -pas_size 4 -pas_rank 1 &
> runmc -ce 4 ./example1.ppc -pas_size 4 -pas_rank 2 &
> runmc -ce 5 ./example1.ppc -pas_size 4 -pas_rank 3 &
> runmc -ce 2 ./example1.ppc -pas_size 4 -pas_rank 0
> sysmc -ce 2 reset
```

For example, to run a parallel VSIPL++ application with PAS for Linux Clusters on compute nodes c1, c2, c3, c4:

```
> rsh -n c1 example1 -pas_size 4 -pas_rank 0 &
> rsh -n c2 example1 -pas_size 4 -pas_rank 1 &
> rsh -n c3 example1 -pas_size 4 -pas_rank 2 &
```

```
> rsh -n c4 example1 -pas_size 4 -pas_rank 3 &
> wait
```

Please consult your PAS library documentation for more details on starting PAS and running PAS programs.

4.6. Building Applications with the VSIPL API

Building applications with Sourcery VSIPL works very much as described in Section 4.1, “Building Manually”. Start by setting the `PKG_CONFIG_PATH` variable, then use pkg-config to determine the relevant build parameters. The notable difference is that the `packagename` here is *vsip*, not *vsip++*:

```

> export PKG_CONFIG_PATH=/opt/sourceryvsipl++-2.2-9/lib/pkgconfig
> `pkg-config --variable=cc vsipl` -c `pkg-config --cflags vsipl` \
     /opt/sourceryvsipl++-2.2-9/share/sourceryvsipl++/cvsip/fft.cpp
> `pkg-config --variable=cxx vsipl` -o fft fft.o `pkg-config --libs \
vsipl`
    \
```

Please note that, while the compilation itself can be carried out with a C compiler, the link step has to be done with a C++ compiler to make sure all the C++ system libraries are properly linked in.

And, similar to Section 4.2, “Building with GNU Make”, a Makefile is provided with a demo application, which you may use as a starting point for writing VSIPL application:

```
> cd /opt/sourceryvsipl++-2.2-9/share/sourceryvsipl++/cvsip
> make
g++ -I/opt/sourceryvsipl++-2.2-9/include -o fft fft.c \
-L/opt/sourceryvsipl++-2.2-9/lib
  -lsvsip -lsvpp
```

Appendix A
Sourcery VSIPL++ Software License Agreement

December 2008

Preamble

CODESOURCERY, INC. (“ CODESOURCERY”) IS WILLING TO LICENSE THE SOFTWARE TO YOU ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS AGREEMENT. PLEASE READ THE TERMS CAREFULLY. BY DOWNLOADING THIS SOFTWARE, YOU WILL INDICATE YOUR AGREEMENT WITH THEM. IF YOU ARE ENTERING INTO THIS AGREEMENT ON BEHALF OF A COMPANY OR OTHER LEGAL ENTITY, YOUR ACCEPTANCE REPRESENTS THAT YOU HAVE THE AUTHORITY TO BIND SUCH ENTITY TO THESE TERMS, IN WHICH CASE “YOU” OR “YOUR” SHALL REFER TO YOUR ENTITY. **YOU MUST** ACCEPT THESE TERMS AND HAVE THE AUTHORITY TO BIND YOUR ENTITY IN ORDER TO**INSTALL OR USE SOURCERY VSIPL++.**

Sourcery VSIPL++ Software License Agreement

The parties to this Agreement are you, the licensee (“Licensee” or “You“) and CodeSourcery, Inc. (“CodeSourcery”).

1. **Definitions.**

   1. **“Authorized Users.”** The developers for whom license fees are fully paid by Licensee and that are authorized to use the Licensed Software. The number of Authorized Users is set forth in the Quotation.

   2. **“Effective Date.”** The date on which Licensee accepts this Agreement.

   3. **“End User.”** A party unaffiliated with Licensee who acquires Licensee Products incorporating the Licensed Software from Licensee for internal use and not for redistribution.

   4. **“Licensed Platforms.”** The set of host and target platforms for which CodeSourcery will provide support under this Agreement as set forth in the Quotation.

   5. **“Licensed Software.”** The libraries, in source code and/or binary form, and accompanying documentation, including any Updates, specified in the Quotation.

   6. **“Licensee Product.”** The Licensee software and/or hardware products incorporating the Licensed Software as set forth in the Quotation.

   7. **“License Type.”** The License Type specified in the Quotation: Evaluation, Internal Use, OEM (Application Development), or OEM (Device Manufacturer).

   8. **“Proprietary Rights.”** All rights in and to copyrights, rights to register copyrights, trade secrets, inventions, patents, patent rights, trademarks, trademark rights, confidential and proprietary information protected under contract or otherwise under law, and other similar rights or interests in intellectual or industrial property.

2. **License Grant.**

   2.1. **Evaluation License.** If the License Type is an Evaluation License, subject to the terms and conditions hereof, and only for the term hereof, CodeSourcery hereby grants to Licensee, and Licensee hereby accepts, a limited, non-exclusive license under the Proprietary Rights of CodeSourcery to install and use the Licensed Software and to create derivative worksbased the Licensed Software for internal use and solely for the purpose of evaluating the Licensed Software.

   2.2. **Internal Use License.** If the License Type is an Internal Use License, subject to the terms and conditions hereof, and only for the term hereof, CodeSourcery hereby grants to Licensee a limited, non-exclusive license under the Proprietary Rights of CodeSourcery for the Authorized Users to install and use the Licensed Software and to create derivative works based the Licensed Software for internal use; provided that said derivative works must not be a signal- or image-processing toolkit or library and must provide substantially different functionality than the Licensed Software.

2.3. **OEM License.** If the License Type is an OEM (Application Development) or OEM (Device Manufacturer) License, subject to the terms and conditions hereof, and only for the term hereof, CodeSourcery hereby grants to Licensee the right and license to install and use the Licensed Software, to create derivative works based the Licensed Software, and to reproduce and distribute Licensee Product(s), incorporating the Licensed Software to End Users in object code form only; provided that Licensee Product must not be a signal- or image-processing toolkit or library and must provide substantially different functionality than the Licensed Software.

3. **Updates.** During the term of this Agreement, Licensee may download, free of charge, any new version(s), update(s), or upgrade(s) ("Updates") to the Licensed Software that CodeSourcery makes available through CodeSourcery's electronic support system.

4. **Fees and Payment Terms.** The licenses and rights granted in this Agreement are subject to Licensee's payment of all fees owed to CodeSourcery as set forth in the Quotation.All fees are due and payable within thirty (30) days of receipt of invoice. All fees are non-refundable and are exclusive of sales or use taxes and any levy imposed on the transportation or use of the Licensed Software. Licensee shall pay all such charges either as levied by taxing authorities or as invoiced by CodeSourcery. Late payments will accrue interest at the rate of eighteen percent (18%) per annum, or at such lower rate required by applicable law.

5. **Royalty Payments, Reports and Audit Rights.** If royalties are due to CodeSourcery under this Agreement, royalty payments and reports shall be made to CodeSourcery on a quarterly basis within thirty (30) days of the end of each calendar quarter. Licensee shall keep full and accurate records in accordance with generally accepted accounting principles and in sufficient detail to permit the determination of such royalties. Licensee's written report shall contain the following information:

   a. the quantity sold or otherwise transferred during the accounting period, and the sum of the licensing fees for such quantity; and

   b. the amount of royalties due and how they were calculated.

   In the event no royalties are due, Licensee's report shall so state.

   Upon written notice for an audit, Licensee shall permit auditors designated by CodeSourcery, together with such legal and technical support as CodeSourcery deems necessary, to examine, during ordinary business hours, records, materials, and/or manufacturing processes of Licensee for the purpose of verifying compliance with this Agreement.

   Each party shall pay the costs that it incurs in the course of the audit. However, in the event that the audit establishes underpayment greater than five percent (5%) of the royalties due, Licensee shall reimburse CodeSourcery for the cost of the audit; provided, however, such reimbursement shall not exceed the amount of the underpayment.

6. **Purchase of Additional Licenses.** If Licensee purchases license rights for additional Authorized Users, such additional licenses shall be governed by the terms and conditions hereof. Pricing for additional licenses shall be in accordance with CodeSourcery's then-current price list, which may be updated by CodeSourcery from time to time, pro-rated as appropriate for the remainder of the

current subscription term. Licensee agrees that, absent CodeSourcery's express written acceptance thereof, the terms and conditions contained in any purchase order or other document issued by Licensee to CodeSourcery for the purchase of additional licenses, shall not be binding on CodeSourcery to the extent that such terms and conditions are additional to or inconsistent with those contained in this Agreement.

7. **Term and Termination.**

7.1. **Evaluation License.** This Agreement shall have a term of thirty (30) days. The evaluation term may only be extended with CodeSourcery's express written agreement. In addition, **CodeSourcery may terminate this Agreement upon seven (7) days written notice of a material breach of this Agreement if such breach is not cured; provided that the unauthorized use, copying, or distribution of the Licensed Software will be deemed a material breach that cannot be cured.** Upon the expiration or termination hereof, Licensee shall cease using the Licensed Software and any derivative works thereof.

7.2. **Internal Use and OEM Licenses.**

7.2.1. **Term.** This Agreement shall have a term of one (1) year. The initial subscription term of this Agreement shall commence as of the Effective Date hereof and shall continue for a period of one (1) year. The initial subscription term shall automatically renew for successive one (1) year terms provided that Licensee pays the Maintenance Fee in each subsequent year as invoiced by CodeSourcery.

7.2.2. **Grounds for Termination. CodeSourcery may terminate this Agreement upon thirty (30) days written notice of a material breach of this Agreement if such breach is not cured; provided that the distribution of the Licensed Software in source code form will be deemed a material breach that cannot be cured.**

7.2.3. **Effects of Expiration or Termination.** Upon the expiration or termination hereof, Licensee shall cease using the Licensed Software and any derivative works thereof and shall cease distributing Licensee Product(s) incorporating the Licensed Software. Notwithstanding anything to the contrary contained herein, sublicenses for all Licensee Product(s), including any derivative works based on the Licensed Software, granted by Licensee to an End User prior to the expiration date or the effective date of termination of this Agreement shall remain in full force and effect following such dates.

7.2.4. **Continuing Obligations.** The following obligations shall survive the expiration or termination hereof: (i) any and all warranty disclaimers or limitations of liability herein, (ii) any covenant granted herein for the purpose of determining ownership of, or protecting, the Proprietary Rights, or any remedy for breach thereof, and (iii) the payment of taxes, duties, or any fees to CodeSourcery hereunder.

8. **Technical Support.**

8.1. **Scope of Support.** CodeSourcery shall assist Licensee in installing and using the Licensed Software. CodeSourcery shall correct defects in the Licensed Software reported by Customer, subject to the limitations set forth below. CodeSourcery shall impose no limit on the number of support requests made by Licensee.

8.2. **Electronic Support System.** Licensee shall make all support requests via CodeSourcery's electronic support system. Licensee shall appoint up to two Technical Contacts, who will be provided access to CodeSourcery's electronic support system. Licensee may replace either or both of the Technical Contacts from time to time by written notification to CodeSourcery. CodeSourcery will not accept support requests by telephone or other means.

8.3. **Response Time.** CodeSourcery's electronic support system will provide Licensee with an immediate acknowledgement of the support request (including a unique tracking number) by electronic mail. If the License Type is an Internal Use or OEM License, CodeSourcery shall respond to all support requests within one business day, except in extraordinary circumstances, and CodeSourcery shall attempt to resolve all support requests within three business days.

8.4. **No Guarantee of Resolution.** CodeSourcery does not guarantee that it will be able to resolve all support requests. Without limitation, CodeSourcery may, in its sole discretion, determine that a defect in the Licensed Software is too difficult to correct, or than any correction would likely risk the introduction of additional defects, or that the defect is not likely to be encountered often enough to be worthy of correction, or that the defect is insufficiently severe to be worthy of correction.

9. **Confidentiality of Licensed Software.** Licensee acknowledges CodeSourcery's claim that the Licensed Software embodies valuable trade secrets consisting of algorithms, logic, design, and coding methodology proprietary to CodeSourcery. Licensee shall safeguard the confidentiality of the Licensed Software, using the same standard of care which Licensee uses for its similar confidential materials, but in no event less than reasonable care. Licensee shall not: (i) distribute, transfer, loan, rent, or provide access to the Licensed Software, except as provided herein; (ii) remove or add any Proprietary Rights notice associated with the Licensed Software without the express written permission of CodeSourcery; or (iii) publish any benchmark or performance data related to the Licensed Software without the prior written approval of CodeSourcery.

10. **Assignment and Transfers.** Licensee may not transfer any rights under this Agreement without the prior written consent of CodeSourcery, *except in the case of the sale of all or substantially all of the assets of the business unit associated with the Licensee Product(s) incorporating the Licensed Software*. A condition to any transfer or assignment shall be that the recipient agrees to the terms of this Agreement. Any attempted transfer or assignment in violation of this provision shall be null and void.

11. **Ownership.** CodeSourcery owns and/or has licensed the Licensed Software and all Proprietary Rights embodied therein, including copyrights and valuable trade secrets embodied in its design and coding methodology. The Licensed Software is protected by United States copyright laws and international treaty provisions. CodeSourcery also owns all rights, title and interest in and with respect to its trade names, domain names, trade dress, logos, trademarks, service marks, and other similar rights or interests in intellectual property. This Agreement provides Licensee only a limited use license, and no ownership of any intellectual property.

12. **Proprietary Rights Warranty and Indemnification.** CodeSourcery represents and warrants that CodeSourcery has the authority to license the rights to the Licensed Software which are granted herein. If the License Type is an Internal Use or OEM License, CodeSourcery shall defend, indemnify, and hold Licensee harmless from claim or damage arising out of (i) the lack of right or authority to license the Licensed Software, or (ii) infringement of any copyright, trade secret, or patent as a result of the use of a the Licensed Software; provided, however, that CodeSourcery is promptly notified in writing of any such suit or claim, and further provided that Licensee permits CodeSourcery to defend, compromise, or settle same, and provides all available information and reasonable assistance to enable CodeSourcery to do so. The foregoing is exclusive and states the entire liability of CodeSourcery with respect to infringements or misappropriation of any Proprietary Rights by the Licensed Software. IF THE LICENSE TYPE IS AN EVALUATION LICENSE, CODESOURCERY DISCLAIMS ALL WARRANTIES, BOTH EXPRESS AND IMPLIED, INCLUDING ANY IMPLIED WARRANTY OF NON-INFRINGEMENT.

13. **Warranty Disclaimers.** EXCEPT AS SET FORTH ABOVE, TECHNICAL SUPPORT IS PROVIDED IN LIEU OF ANY WARRANTY FOR THE LICENSED SOFTWARE. TO THE EXTENT ALLOWED BY LAW, CODESOURCERY AND ITS LICENSORS HEREBY DIS-

CLAIM ALL WARRANTIES, BOTH EXPRESS AND IMPLIED, INCLUDING IMPLIED WARRANTIES RESPECTING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THERE IS NO WARRANTY OR GUARANTEE THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT THE SOFTWARE WILL MEET ANY PARTICULAR CRITERIA OF PERFORMANCE, QUALITY, ACCURACY, PURPOSE, OR NEED. LICENSEE ACKNOWLEDGES THAT NO REPRESENTATIONS OTHER THAN THOSE CONTAINED IN THIS AGREEMENT HAVE BEEN MADE RESPECTING THE LICENSED SOFTWARE OR SERVICES TO BE PROVIDED HEREUNDER, AND THAT LICENSEE HAS NOT RELIED ON ANY REPRESENTATION NOT EXPRESSLY SET OUT IN THIS AGREEMENT.IF IMPLIED WARRANTIES MAY NOT BE DISCLAIMED UNDER APPLICABLE LAW, THEN ANY IMPLIED WARRANTIES ARE LIMITED IN DURATION TO THE PERIOD REQUIRED BY APPLICABLE LAW.

14. **Disclaimer of Incidental and Consequential Damages; Limitation of Liability.** INDEPENDENT OF THE FORGOING PROVISIONS, IN NO EVENT AND UNDER NO LEGAL THEORY, INCLUDING WITHOUT LIMITATION, TORT, CONTRACT, OR STRICT PRODUCTS LIABILITY, SHALL EITHER PARTY BE LIABLE TO THE OTHER OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND, INCLUDING WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER MALFUNCTION, OR ANY OTHER KIND OF COMMERCIAL DAMAGE, EVEN IF THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY TO THE EXTENT PROHIBITED BY APPLICABLE LAW. IN NO EVENT SHALL CODESOURCERY'S LIABILITY FOR ACTUAL DAMAGES FOR ANY CAUSE WHATSOEVER, AND REGARDLESS OF THE FORM OF ACTION, EXCEED THE AMOUNT PAID BY LICENSEE UNDER THIS AGREEMENT DURING THE APPLICABLE SUBSCRIPTION TERM.

15. **Restrictions on License to CML.** CML includes software owned by and licensed from *International Business Machines Corporation ("IBM")*. If the Licensed Software includes CML, then Licensee agrees to the following additional terms and conditions:

    a. Licensee may not copy or transfer CML except as otherwise provided in this Agreement; and

    b. Licensee may not reverse assemble, reverse compile, or otherwise translate CML; and

    c. CML is copyrighted and licensed (not sold) and title to CML is not transferred; and

    d. IBM DISCLAIMS ALL WARRANTIES WITH RESPECT TO THE USE OF CML INCLUDING (WITHOUT LIMITATION) ANY WARRANTY OF NON-INFRINGEMENT AND THE WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE; and

    e. IBM's liability for damages of all types is limited to the amount paid by the Licensee for CML; and

    f. Licensee may only use CML in conjunction with Cell Broadband Engine Processor-based hardware.

    If the License Type is an OEM License, then Licensee agrees to include terms and condition in any sublicense to an End User of a Licensee Product incorporating CML sufficient to:

    a. prohibit the further copying or transferring of CML; and

    b. prohibit reverse assembly, reverse compilation, or other translation of CML; and

c. notify the End User that CML is copyrighted and licensed (not sold) and that title to CML is not transferred; and

d. notify the End User that the licensor and the owner of CML, " DISCLAIMS ALL WARRANTIES WITH RESPECT TO THE USE OF THE LICENSED CODE INCLUDING (WITHOUT LIMITATION) ANY WARRANTY OF NON-INFRINGEMENT AND THE WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE;" and

e. notify the End User that the licensor and owner of CML's liability for damages of all types is limited to the amount paid by the End User for CML; and

f. prohibit the use of CML, except when used in conjunction with Cell Broadband Engine Processor-based hardware.

16. **Reservation of Rights.** All rights not expressly granted to Licensee herein are expressly reserved by CodeSourcery.

17. **Export Controls.** Licensee agrees to comply with all export laws and restrictions and regulations of the United States or foreign agencies or authorities, and not to export or re-export the Licensed Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. As applicable, each party shall obtain and bear all expenses relating to any necessary licenses and/or exemptions with respect to its own export of the Licensed Software from the U.S. Neither the Licensed Software nor the underlying information or technology may be electronically transmitted or otherwise exported or re-exported (i) into Cuba, Iran, Iraq, Libya, North Korea, Sudan, Syria or any other country subject to U.S. trade sanctions covering the Licensed Software, to individuals or entities controlled by such countries, or to nationals or residents of such countries other than nationals who are lawfully admitted permanent residents of countries not subject to such sanctions; or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals and Blocked Persons or the U.S. Commerce Department's Table of Denial Orders. Licensee is responsible for complying with any local laws in Licensee's jurisdiction which might impact Licensee's right to import, export or use the Licensed Software, and Licensee represents that Licensee has complied with any regulations or registration procedures required by applicable law to make this license enforceable.

18. **Severability.** If any provision of this Agreement is declared invalid or unenforceable, such provision shall be deemed modified to the extent necessary and possible to render it valid and enforceable. In any event, the unenforceability or invalidity of any provision shall not affect any other provision of this Agreement, and this Agreement shall continue in full force and effect, and be construed and enforced, as if such provision had not been included, or had been modified as above provided, as the case may be.

19. **Arbitration.** Except for actions to protect intellectual property rights and to enforce an arbitrator's decision hereunder, all disputes, controversies, or claims arising out of or relating to this Agreement or a breach thereof shall be submitted to and finally resolved by arbitration under the rules of the American Arbitration Association ("AAA") then in effect. There shall be one arbitrator, and such arbitrator shall be chosen by mutual agreement of the parties in accordance with AAA rules. The arbitration shall take place in Granite Bay, California, and may be conducted by telephone or online. The arbitrator shall apply the laws of the State of California, USA to all issues in dispute. The controversy or claim shall be arbitrated on an individual basis, and shall not be consolidated in any arbitration with any claim or controversy of any other party. The findings of the arbitrator shall be final and binding on the parties, and may be entered in any court of competent jurisdiction for enforcement. Enforcements of any award or judgment shall be governed by the United Nations Convention on the Recognition and Enforcement of Foreign Arbitral Awards. Should either party

file an action contrary to this provision, the other party may recover attorney's fees and costs up to $1000.00.

20. **U.S. Government End-Users.** The Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire the Licensed Software with only those rights set forth herein.

21. **Jurisdiction And Venue.** The courts of Placer County in the State of California, USA and the nearest U.S. District Court shall be the exclusive jurisdiction and venue for all legal proceedings relating to this Agreement.

22. **Independent Contractors.** The relationship of the parties is that of independent contractor, and nothing herein shall be construed to create a partnership, joint venture, franchise, employment, or agency relationship between the parties. Licensee shall have no authority to enter into agreements of any kind on behalf of CodeSourcery and shall not have the power or authority to bind or obligate CodeSourcery in any manner to any third party.

23. **Force Majeure.** Neither CodeSourcery nor Licensee shall be liable for damages for any delay or failure of delivery arising out of causes beyond their reasonable control and without their fault or negligence, including, but not limited to, Acts of God, acts of civil or military authority, fires, riots, wars, embargoes, or communications failures.

24. **Publicity.** CodeSourcery may disclose the fact that Licensee is a CodeSourcery customer and to display Client’s logo on its web site, together with the logos of other partners and customers.

25. **Miscellaneous.** This Agreement constitutes the entire understanding of the parties with respect to the subject matter of this Agreement and merges all prior communications, representations, and agreements. This Agreement may be modified only by a written agreement signed by the parties. If any provision of this Agreement is held to be unenforceable for any reason, such provision shall be reformed only to the extent necessary to make it enforceable. This Agreement shall be construed under the laws of the State of California, USA, excluding rules regarding conflicts of law. The application the United Nations Convention of Contracts for the International Sale of Goods is expressly excluded. This license is written in English, and English is its controlling language.

Appendix B
GNU General Public License

Version 2, June 1991

Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a “work based on the Program ” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification ”.) Each licensee is addressed as “you”.

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the

Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License.

      Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

11. **NO WARRANTY.** BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THxE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

    END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix C
BSD License