Intel® Parallel Composer 2011 Installation Guide and Release Notes

Document number: 321418-002US
13 August 2010

Table of Contents

1 Introduction

This document provides a summary of new and changed product features, and notes about features and problems not described in the product documentation.

First-time users are encouraged to read *Getting Started with the Intel® Parallel Composer* which can be found in the Intel® Parallel Composer Documentation folder.

1.1 Change History

This section highlights important changes in product updates.

This is the initial product release.

1.2 Product Contents

Intel Parallel Composer includes the following components:

- Intel® C++ Compiler 12.0 for building applications that run on IA-32 or Intel® 64 architecture systems running the Windows* operating system
- Intel® Integrated Performance Primitives 7.0
- Intel® Threading Building Blocks 3.0 Update 2
- Intel® Parallel Debugger Extension
- Integration into Microsoft* development environments
- Sample programs
- On-disk documentation

1.3 System Requirements

For an explanation of architecture names, see http://software.intel.com/en-us/articles/intel-architecture-platform-terminology/

- A PC based on an IA-32 or Intel® 64 architecture processor supporting the Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions (Intel® Pentium 4 processor or later, or compatible non-Intel processor)
    - For the best experience, a multi-core or multi-processor system is recommended
- 1GB RAM
- 3GB free disk space for all product features and all architectures
- Microsoft Windows XP*, Microsoft Windows Vista*, Microsoft Windows 7*, Microsoft Windows Server 2003* or Microsoft Windows Server 2008* (embedded editions not supported)
- When installed on Microsoft Windows Server 2008, one of:
    - Microsoft Visual Studio 2010* with C++ and “x64 Compiler and Tools” components installed [1]
    - Microsoft Visual Studio 2008* Standard Edition (or higher edition) SP1 with C++ and “x64 Compiler and Tools” components installed [1]
- When installed on Microsoft Windows 7, Windows XP, Windows Vista or Windows Server 2003, one of:
    - Microsoft Visual Studio 2010* with C++ and “x64 Compiler and Tools” components installed [1]
    - Microsoft Visual Studio 2008* Standard Edition (or higher edition) with C++ and “x64 Compiler and Tools” components installed [1]
    - Microsoft Visual Studio 2005* Standard Edition (or higher edition) with C++ and “x64 Compiler and Tools” components installed [1]
- To read the Release Notes, Adobe Reader* 7.0 or later

Notes:

1. Microsoft Visual Studio 2005 and 2008 Standard Edition installs the “x64 Compiler and Tools” component by default – the Professional and higher editions require a “Custom” install to select this. Microsoft Visual Studio 2010 provides the “x64” support by default in all editions.
2. The default for the Intel® compilers is to build IA-32 architecture applications that require a processor supporting the Intel® SSE2 instructions - for example, the Intel® Pentium®

4 processor. A compiler option is available to generate code that will run on any IA-32 architecture processor. However, if your application uses Intel® Integrated Performance Primitives or Intel® Threading Building Blocks, executing the application will require a processor supporting the Intel® SSE2 instructions.

3. Applications can be run on the same Windows versions as specified above for development. Applications may also run on non-embedded 32-bit versions of Microsoft Windows earlier than Windows XP, though Intel does not test these for compatibility. Your application may depend on a Win32 API routine not present in older versions of Windows. You are responsible for testing application compatibility. You may need to copy certain run-time DLLs onto the target system to run your application.

1.4 Documentation

Optimization Notice

Intel® Parallel Composer 2011 includes compiler options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel® Parallel Composer 2011 are reserved for Intel microprocessors. For a detailed description of these compiler options, including the instruction sets they implicate, please refer to "Intel® Parallel Composer 2011 Documentation > Intel® C++ Compiler 12.0 User and Reference Guides > Compiler Options." Many library routines that are part of Intel® Parallel Composer 2011 are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® Parallel Composer 2011 offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

While the paragraph above describes the basic optimization approach for Intel® Parallel Composer 2011, with respect to Intel's compilers and associated libraries as a whole, Intel® Parallel Composer 2011 may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

Intel recommends that you evaluate other compilers to determine which best meet your requirements.

Product documentation is primarily accessed through Help > Intel Parallel Studio 2011 in Microsoft Visual Studio. You can also access help outside Visual Studio through Start > All Programs > Intel Parallel Studio 2011 > Parallel Studio Documentation.

If you are new to Intel® Parallel Composer, please begin with *Parallel Studio Getting Started Tutorial* at Start > All Programs > Intel Parallel Studio 2011 > Getting Started.

1.5 Samples

Sample applications can be found in the `Samples` folder as shown under Installation Folders.

1.6 Technical Support

If you did not register your compiler during installation, please do so at the Intel® Software Development Products Registration Center. Registration entitles you to free technical support, product updates and upgrades for the duration of the support term.

For information about how to find Technical Support, Product Updates, User Forums, FAQs, tips and tricks, and other support information, please visit
http://www.intel.com/software/products/support

Note: If your distributor provides technical support for this product, please contact them for support rather than Intel.

2 Installation

2.1 Pre-Installation Steps

2.1.1 Configure Visual Studio for 64-bit Applications

If you are using Microsoft Visual Studio 2005 or 2008 and will be developing 64-bit applications you may need to change the configuration of Visual Studio to add 64-bit support.

If you are using Visual Studio 2005/2008 Standard Edition, or Visual Studio 2010, no configuration is needed to build 64-bit applications. Otherwise:

1. From Control Panel > Add or Remove Programs (or the equivalent on Windows 7 or Windows Vista), select “Microsoft Visual Studio 2005 (or 2008) > Change/Remove. The Visual Studio Maintenance Mode window will appear. Click Next.
2. Click Add or Remove Features
3. Under “Select features to install”, expand Language Tools > Visual C++
4. If the box “X64 Compiler and Tools” is not checked, check it, then click Update. If the box is already checked, click Cancel.

2.1.2 Installation on Microsoft Windows Vista* or Windows 7*

On Microsoft Windows Vista or Windows 7, Microsoft Visual Studio 2005 users should install Visual Studio 2005 Service Pack 1 (VS 2005 SP1) as well as the Visual Studio 2005 Service

Pack 1 Update for Windows Vista, which is linked to from the VS 2005 SP1 page. After installing these updates, you must ensure that Visual Studio runs with Administrator permissions, otherwise you will be unable to use the Intel compiler. For more information, please see Microsoft's Visual Studio on Windows Vista page (http://msdn2.microsoft.com/en-us/vstudio/aa948853.aspx) and related documents.

2.2 Installation

If you are installing the product for the first time, please be sure to have the product serial number available as you will be asked for it during installation. A valid license is required for installation and use.

To begin installation, insert the first product DVD in your computer's DVD-ROM drive; the installation should start automatically. If it does not, open the top-level folder of the DVD-ROM drive in Windows Explorer and double-click on `setup.exe`.

If you received your product as a downloadable file, double-click on the executable file (`.EXE`) to begin installation. Note that there are several different downloadable files available, each providing different combinations of components. Please read the download web page carefully to determine which file is appropriate for you.

You do not need to uninstall previous versions or updates before installing a newer version – the new version will replace the older version. If you have Intel C++ Compiler Professional Edition installed, installing Intel Parallel Composer will replace Intel C++ Compiler's integration into Microsoft Visual Studio, but the other compiler version will remain available for use from the command line.

2.3 Installation Folders

The product installs into a folder arrangement as shown below. Not all folders will be present in a given installation. If other Intel® Parallel Studio tools are installed, they will share the top-level installation folder.

- C:\Program Files\Intel\Parallel Studio 2011\
  - Composer
    - bin
      - ia32
      - ia32_intel64
    - compiler
      - include
        - cilk
        - ia32
      - lib
        - ia32
        - intel64
      - perf_headers
        - c++
    - debugger
    - Documentation

- help
- ipp
  - bin
    - ia32
    - intel64
  - demo
  - include
  - lib
    - ia32
    - intel64
  - tools
    - ia32
    - intel64
- redist
- Samples
- setup_c
- tbb
  - bin
  - examples
  - include
  - lib
- VS Integration

Architecture-specific subfolder names are:

- `ia32`: Files used in building applications that run on IA-32 architecture systems
- `intel64 and ia32_intel64`: Files used in building applications that run on Intel® 64 architecture systems

If you are installing on a system with a non-English language version of Windows, the name of the `Program Files` folder may be different. On Intel® 64 architecture systems, the folder name is `Program Files (X86)` or the equivalent.

2.4 Installation Known Issues

2.4.1 Installation Path Too Long or Filename Too Long

During installation, if the length of the full installation path of any installed file including the filename exceeds 256 characters, the installation will stop with an error message. One possible error message is:

```
Error 1304. Error writing to file: d:\Program Files\Development
Tools\Intel\Parallel Studio
2011\Composer\Documentation\en_US\ipp\ipp_manual\IPPI\ippi_ch16\functn
_YCrCb411ToYCbCr422_EdgeDV_YCrCb411ToYCbCr422_ZoomOut2_EdgeDV_YCrCb411
ToYCbCr422_ZoomOut4_EdgeDV_YCrCb411ToYCbCr422_ZoomOut8_EdgeDV.htm
```

This can occur because the user has specified a long custom installation root directory. Try shortening this path if you run into this error. Note that this may require reinstallation of other Parallel Studio products.

2.4.2 Additional Steps to Install Documentation for Microsoft Visual Studio 2010

When installing Intel Parallel Studio on a system with Microsoft Visual Studio 2010 for the first time, you will be asked to initialize the “Local Store” for documentation for Visual Studio 2010 if it was not done before. The "Help Library Manager" will register the Intel Parallel Studio help documentation within Visual Studio 2010. Please follow the instructions of the "Help Library Manager" installation wizard to install the Intel Parallel Studio help documentation for Visual Studio 2010.

This step is only needed once. When you install Intel Parallel Studio updates in the future, you will not be required to re-register the documentation through the “Help Library Manager”.

For the more information, see http://msdn.microsoft.com/en-us/library/dd264831.aspx or search for “Help Library Manager”.

2.4.3 Error Message "HelpLibAgent.exe has stopped working" When Uninstalling Intel Parallel Studio 2011

When installing or uninstalling Intel Parallel Studio 2011 on a system with Visual Studio 2010, you may see the error message “HelpLibAgent.exe has stopped working”.

This error does not prevent the installation or uninstallation of Intel Parallel Studio. It is an issue from a 3rd party tool. When there is a fix, the Release Notes will be updated. Please visit http://software.intel.com/en-us/articles/installation-error-helplibagentexe-has-stopped-working-when-uninstalling-intel-parallel-studio-2011/ for the latest update on this issue.

2.4.4 Unicode Characters in License Path

During installation, Intel® software cannot handle Unicode characters in license paths and the names of the licenses. Intel® software tries to find licenses in the standard location (`%CommonProgramFiles%\Intel\Licenses`, most commonly `C:\Program Files\Common Files\Intel\Licenses` on 32-bit systems and `C:\Program Files (x86)\Common Files\Intel\Licenses` on 64-bit systems). Do not place licenses in folders or paths containing localized characters. For example: C:¥インテル¥ライセンス. Do not rename licenses obtained from Intel using localized characters. For example マイライセンス.lic. Do not set the INTEL_LICENSE_FILE environment variable to contain directory paths and license names containing localized characters. Keep licenses either in the standard location (see above), or use ASCII characters in directory names and license names. For example: `C:\Intel\Licenses and License.lic`.

2.4.5 Documentation Issue with Multiple Visual Studio Versions

If you have both Microsoft Visual Studio* 2005 and 2008 installed on your system and integrate Intel® Parallel Studio 2011 into both versions, removing the integration from one of the versions will remove the integrated Intel® Parallel Studio documentation from both.

To re-install the documentation:

For Intel® Parallel Composer 2011:

1. Use the Control Panel to select the product.
    - For Windows XP* users: Select **Control Panel > Add/Remove Programs.**
    - For Windows 7* users: Select **Control Panel > Programs and Features**.
    - For Windows Vista* users: Select **Control Panel > Programs**.
2. With the product selected, click the **Change/Remove** button and choose Modify mode.
3. In the **Select Components** dialog box, unselect "Integrated Documentation;" this will remove the documentation.
4. Repeat steps 1 and 2.
5. In the **Select Components** dialog box, select "Integrated Documentation" to install documentation again

3 Intel® C++ Compiler

This section summarizes changes, new features and late-breaking news about the Intel® C++ Compiler as part of Intel® Parallel Composer 2011.

3.1 New Features

The following features are new or significantly enhanced in Intel® Parallel Composer 2011. For more information on these features, please refer to the documentation viewable from within Visual Studio.

- Intel® Cilk™ Plus language extensions for the Intel® C++ Compiler make it easy to add parallelism to both new and existing software.
- Guided Auto-Parallelism
- Features from C++0x
    - `rvalue` references
    - Standard atomics
    - Support of C99 hexadecimal floating point constants when in "Windows C++" mode
    - Right angle brackets
    - Extended friend declarations
    - Mixed string literal concatenations
    - Support for `long long`
    - Variadic macros
    - Static assertions
    - Auto-typed variables
    - Extern templates
    - `__func__` predefined identifier
    - Declared type of an expression (decltype)
    - Universal character name literals
    - Strongly-typed enums

    - Lambdas
- An option to use math library functions that are faster but return results with less precision or accuracy
- An option to use math library functions that return consistent results across different models and manufacturers of processors

3.2 New Command Line Options

For more information on these options, please refer to the documentation viewable from within Visual Studio.

- /Qansi-alias-checker
- /Qcilk-serialize
- /Qfp-trap
- /Qfp-trap-all
- /Qguide
- /Qguide-data-trans
- /Qguide-file
- /Qguide-file-append
- /Qguide-opts
- /Qguide-par
- /Qguide-vec
- /Qimf-absolute-error
- /Qimf-accuracy-bits
- /Qimf-arch-consistency
- /Qimf-max-error
- /Qimf-precision
- /Qintel-extensions
- /Qopt-args-in-regs
- /Qopt-matmul
- /Qpatchable-addresses
- /Qpar-runtime-control[n]
- /Qpar-runtime-control-
- /Qregcall

Detail information about some new options:

- /Qpar-runtime-control[n]: Generates code to perform run-time checks for loops that have symbolic loop bounds. “n” is a value denoting what kind of runtime checking to perform. Possible values are:

    0 Performs no runtime check based on auto-parallelization.

    1 Generates runtime check code under conservative mode. This is the default if you do not specify n.

    2 Generates runtime check code under heuristic mode.

    3 Generates runtime check code under aggressive mode.

- /Qpar-runtime-control-: The compiler uses default heuristics when checking loops.

3.3 Compatibility with Previous Versions

This section summarizes changes in the C++ compiler that may present compatibility issues when mixing code compiled with previous versions of Intel Parallel Composer or Intel C++ Compiler with code compiled with this version of Intel Parallel Composer.

3.3.1 New Optimization May Cause Multiple Definition Link Errors for Math Library Functions

The C++ Compiler now replaces some calls to double precision math library functions with calls to the corresponding single precision counterpart where only a single precision result is stored. For example:

```
float = (float)double_precision_intrinsic((double)(float_expression);
```

is optimized into:

```
float = single_precision_intrinsic(float_expression);
```

If the application has a user-supplied function of the same name as the single-precision intrinsic, this can lead to multiple-definition linking errors. This optimization is done at optimization level 2 (`/O2`) or above. It is not done if `/Qfreestanding` is specified nor if any of the `/RTC*` options are used. To tell the linker to allow multiple definitions, add the linker option `/FORCE:MULTIPLE`.

3.4 Known Issues

3.4.1 Visual Studio Known Issues

3.4.1.1 Visual Studio 2010 sets default of `/fp:precise`

A project created in or converted to Visual Studio 2010 will have the command line option `/fp:precise` set by default. This option sets the “floating point model” to improve consistency for floating point operations by disabling certain optimizations, reducing performance. To set the option back to the Intel default of `/fp:fast`, change the project property C++ > Optimization > Floating Point Model to Fast.

*3.4.1.2 Custom Build Rules in Microsoft Visual Studio 2005**

The Intel® C++ integration into Microsoft Visual Studio 2005* includes support for the Visual C++ Custom Build Rule functionality with some limitations. Custom Build Rules allow you to add custom tool invocations to your build process. See the Visual C++ documentation for a detailed explanation of Custom Build Rules.

You must create your custom build rules before converting your Visual C++ project to the Intel project system. If you need to modify your custom build rules after converting your project to the Intel project system, you must convert your project to the Visual C++ project system, make your custom build rule changes, and then convert the project back to the Intel project system.

The Macro lists that are available when you change the string properties defined for a Custom Build Rule don't contain Intel-specific macros (such as $(icInstallDir); $(icIDEInstallDir); $(icProjectExt); $(icProjectFileName)). However, you can use the Intel-specific macros in Custom Build Rule property values and they will be expanded correctly.

The Intel® C++ integration into Visual Studio 2005 does not currently support the Visual C++ Tool Build Order functionality which is available from the Visual C++ Tool Build Order dialog box. That is, you cannot change the order in which the build steps are run.

3.4.1.3 Language packs of Visual Studio 2010

If you install a new language pack of Visual Studio 2010 after installing the Intel Parallel Composer, you may not see the Intel C++ Compiler specific options in the Project Property dialog. Please try the following to fix the issue:

1) if directory "`<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel Parallel Composer 2011\1033`" exists, copy all files to "`<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel Parallel Composer 2011\<locale-ID>`".

2) if directory "`<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\1033\`" exists, copy all files to "`<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\<locale-ID>`".

* The `<locale-ID>` is the language pack.

Another method is to uninstall the Intel Parallel Composer 2011 and reinstall the Intel Parallel Composer 2011.

3.4.2 Intel® Cilk™ Plus Known Issues

- Microsoft C++ Structured Exception Handling (SEH) will fail if an SEH exception is thrown after a steal occurs and before the corresponding `_Cilk_sync`.
- A `cilk_spawn` under the `if` in an `if/else` conditional statement may result in a compilation error.
  The Intel C++ Compiler will complain about the following code:

```
if (expr)
    cilk_spawn a();
else
    b();

test.cpp
test.cpp(12): error: expected a statement
        else
        ^
```

The work-around is to add {} around "cilk_spawn" like below:

```
if (expr) {
    cilk_spawn a();
} else
    b();
```

3.4.3 Sample Programs Known Issues

- "libcmt.lib(invarg.obj) : error LNK2005: __invoke_watson already defined in MSVCRT.lib(MSVCR100.dll)" when building par-OpenMP* sample within Visual Studio 2010

  This issue happens when you turn on the /Qdebug:parallel from the par-OpenMP project properties dialog within Visual Studio 2010.

  The work-arounds are:

  - Change "/MD[d]" to "/MT[d]" from the project property C/C++ > Code Generation > Runtime Library field.
  - Or turn off /Qdebug:parallel from the project property C/C++ > Debug [Intel C++] > Enable Parallel Debug Checks

3.4.4 Guided Auto-Parallel Known Issues

Guided Auto Parallel (GAP) analysis for single file, function name or specific range of source code does not work when Whole Program Interprocedural Optimization (/Qipo) is enabled. /Qipo is the default for Intel Parallel Composer in Microsoft Visual Studio projects in Release configurations. The workaround is to disable /Qipo – in Visual Studio, this is Project > [projectname] Properties > C++ > Optimization > Interprocedural Optimization > No.

3.4.5 Using a Source Control System

If your project is managed under a source control system, for example, Microsoft Visual Source Safe* or Microsoft Visual Studio Team Foundation Server*, there are additional steps you must follow in order to use the Intel C++ project system with your project. A detailed article on this topic is available at http://software.intel.com/en-us/articles/tips-on-using-the-intel-c-compiler-with-source-code-control-software/

3.4.6 Option to Clean Project after Conversion to use Intel C++

When the option is selected to use Intel C++ for a project, a dialog box appears offering to "clean" the project (remove results of previous builds). This step is recommended and is selected by default; you may choose to disable the "clean" operation so that you may perform the "clean" manually.

However, if you are using a non-English version of Microsoft Visual Studio, this dialog does not appear and you must "clean" the project manually.

4 Intel® Integrated Performance Primitives

This section summarizes changes, new features and late-breaking news about the Intel Integrated Performance Primitives as part of Intel Parallel Composer 2011.

4.1 Intel® Integrated Performance Primitives 7.0

4.1.1 New and Changed Features

- Additional optimizations for the 256-bit AVX SIMD instruction set (available on Intel® processors code named "Sandy Bridge") have been incorporated.
- Further AES-NI optimizations have been applied to the cryptography domain (separate download, see below) and data compression (CRC32 for ipp_bzip2), substantially improving performance on those processors that support the AES-NI instructions.
- Microsoft* Visual Studio 2010 is now supported by the Windows edition of the IPP library; meaning, IPP help files and project files are compatible with the Visual Studio 2010 IDE. The Visual Studio 2005 and 2008 IDEs are also supported.
- Support for the JPEG-XR (HD Photo) forward and inverse transforms for 16s, 32s and 32f data types and variable length code (VLC) encode and decode functions for 32s data types has been added.
- A JPEG-XR (HD Photo) codec is now included in the IPP UIC sample framework for grayscale, RGB and RGBA images with 8, 16, and 32-bit integer and 16 and 32-bit floating point pixel depths.
- The DMIP sample now includes a Microsoft DSL (Domain Specific Language) add-on for use with Visual Studio 2008.
- A new *interfaces* directory has been added that contains high-level application code, in the form of source and pre-built binaries. Several popular data compression libraries (e.g., bzip2, zlib and gzip) have been modified for use with the IPP library and can be found in the *interfaces* directory for immediate use.
- There is a new ipp_lzopack (data compression) library, located in the *interfaces* directory mentioned above, as part of this release.
- Multi-threading is now part of the ipp_zlib library (by use of the OpenMP multi-threading library).
- A new directory hierarchy has been established to simplify integration of the Intel IPP library with the Intel Compiler products. This change may require that you update your build scripts, makefiles and/or Visual Studio project files.
- Directories formerly designated as "em64t" are now designated by the "intel64" tag. This change may require that you update your build scripts, makefiles and/or Visual Studio project files.
- Library filenames have been normalized to be consistent between 32-bit and 64-bit architectures (i.e., the "em64t" tag has been removed from all 64-bit library file names). This change may require that you update your build scripts, makefiles and/or Visual Studio project files.
- The domain-specific "emerged" and "merged" static library files have been combined for simpler reference (e.g., ippsemerged.lib + ippsmerged_t.lib ⇒ ipps_t.lib) and the single-threaded static libraries are now designated by a "_l" suffix (multi-threaded static

libraries continue to be designated with a "_t" suffix). This change may require that you update your build scripts, makefiles and/or Visual Studio project files.

- The speech recognition functions (ippSR domain) are not part of this release; this domain will continue to be supported in the IPP 6.1 product.
- Intel® Itanium® architecture (IA-64) support is not included in this release. Intel® IPP 6.1 is the latest release for the IA-64 architecture.
- The SSE2 optimization, and some obsolete SSE3 optimizations, (t7/m7 and w7 libraries) have been removed from this version of the IPP library; these library optimizations will continue to be supported in the IPP 6.1 product. Additionally, the base 32-bit optimization of the library (px) has been compiled to require a processor with a minimum SSE2 architecture; which is consistent with the 64-bit base optimization (mx).
- Redundant CPU static library optimizations have been combined (v8+s8 and u8+n8) into a single static library. A separate Intel® Atom processor-specific static library will no longer be provided. This change does not impact the dynamic libraries, which will continue to be distributed with separate v8/u8 and s8/n8 optimizations.
- The SPIRAL generated functions (ippGEN domain) are now distributed as a separate download. See instructions below for more information.

4.1.2 Directory Structure Changed

A new directory hierarchy has been established to simplify integration with the Intel® compiler products. This may require action on your part in order to be able to continue building applications that use Intel® Integrated Performance Primitives.

- If you used the “Select Build Components” dialog for the Intel® performance libraries with earlier versions of Intel® Parallel Composer or Intel® C++ Compiler, right click on your C++ project in Microsoft Visual Studio, select Intel Parallel Composer > Select Build Components, then click OK to update the paths.
- If you use the Tools > Options > Intel Parallel Composer > Compilers dialog to change the version of compiler you are using, you must also update the Build Components paths as above.

4.2 Intel® IPP Cryptography Libraries are Available as a Separate Download

The Intel® IPP cryptography libraries are available as a separate download. For download and installation instructions, please read
http://software.intel.com/en-us/articles/download-ipp-cryptography-libraries/

4.3 SPIRAL Generated Functions are Available as a Separate Download

The *SPIRAL generated functions*, a special subset of the Signal Processing domain of the Intel IPP library that was added in version 6.0 of the library, have been moved to a separately installed add-on library. This change was made to reduce the overall download size of the IPP library. The functions contained within the ippGEN domain all begin with the ippg prefix and are listed below using an abbreviated naming format.

- `ippgenGetLibVersion`

- ippgDCT4_[32f|64f]
- ippgDCT4Init_[32f|64f]
- ippgDCT4InitAlloc_[32f|64f]
- ippgDCT4Free_[32f|64f]
- ippgDCT4GetSize_[32f|64f]

- ippgDFTFwd_CToC_[32fc|64fc]
- ippgDFTFwd_CToC_*_[32fc|64fc]
- ippgDFTInv_CToC_[32fc|64fc]
- ippgDFTInv_CToC_*_[32fc|64fc]

- ippgHartley_[32f|64f]
- ippgHartley_*_[32f|64f]

You will find the download for the SPIRAL generated functions alongside the compiler download at the Intel® Software Development Products Registration Center.

4.4 Intel® IPP Code Samples

The Intel® IPP code samples are organized into downloadable packages at http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-code-samples/

The samples include source code for audio/video codecs, image processing and media player applications, and for calling functions from C++, C# and Java*. Instructions on how to build the sample are described in a readme file that comes with the installation package for each sample.

5 Intel® Threading Building Blocks

For information on changes to Intel® Threading Building Blocks, please read the file CHANGES in the TBB documentation directory.

5.1.1 Directory Structure Changed

A new directory hierarchy has been established to simplify integration with the Intel® compiler products. This may require action on your part in order to be able to continue building applications that use Intel® Threading Building Blocks.

- If you used the “Select Build Components” dialog for the Intel® performance libraries with earlier versions of Intel® Parallel Composer or Intel® C++ Compiler, right click on your C++ project in Microsoft Visual Studio, select Intel Parallel Composer > Select Build Components, then click OK to update the paths.
- If you use the Tools > Options > Intel Parallel Composer > Compilers dialog to change the version of compiler you are using, you must also update the Build Components paths as above.

- If you create a project using one version of Microsoft Visual Studio and then open the project in a different version, again you must update the Build Components paths as above.

6 Intel® Parallel Debugger Extension

This section summarizes changes, new features and late-breaking news about the Intel Parallel Debugger Extension as part of Intel Parallel Composer.

6.1 New Features

- Intel® Cilk™ Plus Support
  - Serialized execution of an Intel® Cilk Plus program at debug-time, without recompilation.
  - Intel® Cilk Plus worker threads are clearly marked in the Visual Studio Debugger.
  - Display of stealing points and current Cilk Plus workers on Cilk Plus Thread stack window
- Threads Window
  - Improved Data Sharing Detection
  - Support for OpenMP 3.0
  - Support for Windows OS synchronization functions
  - Improved data sharing detection analysis performance

6.2 Known Issues

- If you are using Microsoft Visual Studio 2005, there are six Intel-specific exceptions that must be enabled manually. Select `Debug` > `Exceptions`, expand the `Win32 Exceptions` tree, and enable items:

  ```
  a1a01db0 Intel Parallel Debugger Extension Exception 0
  a1a01db1 Intel Parallel Debugger Extension Exception 1
  a1a01db2 Intel Parallel Debugger Extension Exception 2
  a1a01db3 Intel Parallel Debugger Extension Exception 3
  a1a01db4 Intel Parallel Debugger Extension Exception 4
  a1a01db5 Intel Parallel Debugger Extension Exception 5
  ```

  This needs to be done once per project.

- Use of the Intel Parallel Debugger Extension requires that the OpenMP library be linked dynamically, which is the default. If you wish to use the Parallel Debugger Extension, do not use `/Qopenmp-link:static` to specify static linking of the OpenMP Library.

- Disabling the Intel Debugging exceptions during a debug session may cause Visual Studio (up to Visual Studio 2008, SP1) to hang.

- Be sure to enable the parallel debug instrumentation before you start parallel debugging: `Project > [project name] Properties` > `Configuration Properties` > `C/C++` > `Debug` > `Enable Parallel Debug Checks`. Otherwise, the debugger will

not detect data sharing events nor break on re-entrant calls.

- Microsoft Visual Studio 2010 users: If you have specified that the project be built with Microsoft Visual C++, and then change the project to use the Intel C++ compiler, you must close and re-open the solution for the Parallel Debug Environment property setting “Auto” to be recognized.

- Microsoft Visual Studio 2010 users: you will see link errors when building par-OpenMP sample program like:
  "libcmt.lib(invarg.obj) : error LNK2005: __invoke_watson already defined in MSVCRT.lib(MSVCR100.dll)" when building par-OpenMP sample within Visual Studio 2010

  This known issue happens when you turn on the /Qdebug:parallel from the par-OpenMP project properties dialog within Visual Studio 2010.
  The work-arounds are:
  . Change "/MD[d]" to "/MT[d]" from the project property C/C++ > Code Generation > Runtime Library field.
  . Or turn off /Qdebug:parallel from the project property C/C++ > Debug [Intel C++] > Enable Parallel Debug Checks
- If you are using Microsoft Visual Studio 2008 and debugging 64-bit applications, you must have Visual Studio 2008 Service Pack 1 installed.

- You can debug 64-bit applications under Visual Studio 2005 and 2008 without Service Packs only if they are linked to the low memory area. If not linked to the low memory area, you will not see any events until the debuggee terminates. After termination, all events are displayed in the event window. In order to debug 64-bit applications properly, set the base address to `0x10000` in `Project` > `[project name] Properties` > `Linker` > `Advanced`.

- Function local or heap variables are displayed as “???” in the data sharing event window.

- The SSE Registers window does not work for 64-bit applications - the window shows "`???`"

- Filters on static local variables are not set correctly via context menu.

- Reentrancy detection stops in Disassembly view.

- The debugger extension windows remain empty when their placement is changed from "docked" to "floating". The workaround is to either keep them docked or to restart the debug session after the placement was changed

- The debugger extension requires the application to be started from Visual Studio. It does not work when attaching to an existing process.

- Windows settings are restored to default (Hexadecimal) when the window is hidden or closed and reopened again.

- Reentrant call detection may display the wrong OpenMP Task ID. Consider using the Thread ID instead.

6.3 Documentation

Intel Parallel Debugger Extension Documentation can be accessed via the Help menu of Microsoft Visual Studio or by clicking the Help button of specific dialog boxes. Context Sensitive Help is also available by clicking the function key F1 after activation of a Debugger Extension window.

7 Disclaimer and Legal Information