Intel® High Level Synthesis Compiler Pro Edition

Getting Started Guide

Updated for Intel® Quartus® Prime Design Suite: **21.1**

Contents

intel.

1. Intel® High Level Synthesis (HLS) Compiler Pro Edition Getting Started Guide

The Intel® High Level Synthesis (HLS) Compiler is a separately-installable component of Intel Quartus® Prime Pro Edition design software. The Intel HLS Compiler synthesizes a C++ function into an RTL implementation that is optimized for Intel FPGA products. The compiler is sometimes referred to as the i++ compiler, reflecting the name of the compiler command.

The *Intel High Level Synthesis Compiler Pro Edition Getting Started Guide* describes the procedures to set up the Intel HLS Compiler and to run an HLS design example.

In this publication, `<quartus_installdir>` refers to the location where you installed Intel Quartus Prime Design Suite.

The default Intel Quartus Prime Design Suite installation location depends on your operating system:

Windows	`C:\intelFPGA_pro\21.1`
Linux	`/home/<username>/intelFPGA_pro/21.1`

About the Intel HLS Compiler Pro Edition Documentation Library

Documentation for the Intel HLS Compiler Pro Edition is split across a few publications. Use the following table to find the publication that contains the Intel HLS Compiler Pro Edition information that you are looking for:

Table 1.   Intel High Level Synthesis Compiler Pro Edition Documentation Library

Title and Description	PRO
Release Notes Provide late-breaking information about the Intel HLS Compiler.	Link
Getting Started Guide Get up and running with the Intel HLS Compiler by learning how to initialize your compiler environment and reviewing the various design examples and tutorials provided with the Intel HLS Compiler.	Link
User Guide Provides instructions on synthesizing, verifying, and simulating intellectual property (IP) that you design for Intel FPGA products. Go through the entire development flow of your component from creating your component and testbench up to integrating your component IP into a larger system with the Intel Quartus Prime software.	Link
continued...	

ISO
9001:2015
Registered

Title and Description	PRO
Reference Manual Provides reference information about the features supported by the Intel HLS Compiler. Find details on Intel HLS Compiler command options, header files, pragmas, attributes, macros, declarations, arguments, and template libraries.	Link
Best Practices Guide Provides techniques and practices that you can apply to improve the FPGA area utilization and performance of your HLS component. Typically, you apply these best practices after you verify the functional correctness of your component.	Link
Quick Reference Provides a brief summary of Intel HLS Compiler declarations and attributes on a single two-sided page.	Link

1.1. Intel High Level Synthesis Compiler Pro Edition Prerequisites

The Intel HLS Compiler Pro Edition is part of the Intel Quartus Prime Pro Edition Design Suite. You can install the Intel HLS Compiler as part of your Intel Quartus Prime software installation or install it separately. It requires Intel Quartus Prime and additional software to use.

For detailed instructions about installing Intel Quartus Prime Pro Edition software, including system requirements, prerequisites, and licensing requirements, see Intel FPGA Software Installation and Licensing.

The Intel HLS Compiler requires the following software in addition to Intel Quartus Prime:

C++ Compiler

On Linux, Intel HLS Compiler requires GCC 9.3.0 including the GNU C++ library and binary utilities (binutils).

This version of GCC is provided as part of your Intel HLS Compiler installation. After installing the Intel HLS Compiler, GCC 9.3.0 is available in `<quartus_installdir>/gcc`.

Important: The Intel HLS Compiler uses the `<quartus_installdir>/gcc` directory as its toolchain directory. Use this installation of GCC for all your HLS-related design work.

For Windows, install one of the following versions of Microsoft* Visual Studio* Professional:

- Microsoft Visual Studio 2017 Professional
- Microsoft Visual Studio 2017 Community

For the most up-to-date C++17 support, ensure that you are using the latest version of Visual Studio 2017.

Important: The Intel HLS Compiler software does not support versions of Microsoft Visual Studio other than those specified for the edition of the software.

Mentor Graphics* ModelSim* Software

On Windows and RedHat Linux systems, you can install the ModelSim* software from the Intel Quartus Prime software installer. The available options are:

- ModelSim - Intel FPGA Edition
- ModelSim - Intel FPGA Starter Edition

Alternatively, you can use your own licensed version of Mentor Graphics* ModelSim or Mentor Graphics Questa* Advanced Simulator software.

On Linux systems, ModelSim software requires the Red Hat* development tools packages. Additionally, any 32-bit versions of ModelSim software (including those provided with Intel Quartus Prime) require additional 32-bit libraries. The commands to install these requirements are provided in Installing the Intel HLS Compiler on Linux Systems.

For information about all the ModelSim software versions that the Intel software supports, refer to the *EDA Interface Information* section in the Software and Device Support Release Notes for your edition of Intel Quartus Prime Pro Edition

Related Information

- Supported Operating Systems
- Software Requirements
    in *Intel FPGA Software Installation and Licensing*
- EDA Interface Information (Intel Quartus Prime Pro Edition)
- Mentor Graphics ModelSim Website

1.1.1. Intel HLS Compiler Pro Edition Backwards Compatibility

The Intel HLS Compiler Pro Edition Version 21.1 is compatible with Intel Quartus Prime Pro Edition Version 21.1, Version 20.4, Version 20.3, and Version 17.1.1. This backwards compatibility lets you to take advantage of improvements in the RTL generated by the Intel HLS Compiler without changing the other parts of your current FPGA development environment.

To install Intel HLS Compiler Pro Edition Version 21.1 into an older version of Intel Quartus Prime Pro Edition:

1. Backup the existing `<quartus_installdir>`[1]`/hls` directory. For example, rename `<quartus_installdir>/hls` to `<quartus_installdir>/hls_old`.

2. Download the **Intel High Level Synthesis Compiler** standalone installation package from the **Additional Software** tab of the Intel Quartus Prime Download Center for your edition of Intel Quartus Prime.

3. Run the Intel HLS Compiler standalone installer, and specify your `<quartus_installdir>` directory when prompted.

You can revert to your previous version of the Intel HLS Compiler by renaming your backup directory. Whatever version of the the Intel HLS Compiler that in a directory called `<quartus_installdir>/hls` is the active version of the Intel HLS Compiler.

---

[1] Where *<quartus_installdir>* is the directory where you installed the Intel Quartus Prime Pro Edition Design Suite. For example, `C:\intelFPGA_pro\21.1`.

1.2. Downloading the Intel HLS Compiler Pro Edition

Download the Intel HLS Compiler Pro Edition from the Intel Quartus Prime Download Center.

The Intel HLS Compiler Pro Edition is installed as part of your Intel Quartus Prime Pro Edition installation package or from a separate standalone installer.

Download the Intel HLS Compiler Pro Edition standalone installer when you want to add the Intel HLS Compiler to an existing Intel Quartus Prime Pro Edition installation. The Intel HLS Compiler Pro Edition Version 21.1 is compatible with Intel Quartus Prime Pro Edition Version 21.1, Version 20.4, Version 20.3, and Version 17.1.1.

- Download the Intel HLS Compiler from the Intel Quartus Prime Download Center :
  - http://fpgasoftware.intel.com/?edition=pro

  The Intel HLS Compiler Pro Edition standalone installation package is available on the **Additional Software** tab.

  To include the Intel HLS Compiler Pro Edition in your Intel Quartus Prime Pro Edition installation, download one of the following combinations of installation packages:

  - Download a complete installation package from the **Combined Files** tab.
  - Download a Quartus Prime installation package from the **Individual Files** tab and download the Intel HLS Compiler installation package from the **Additional Software** tab. Place both installation packages in the same directory.

  For detailed instructions about installing Intel Quartus Prime software, including system requirements, prerequisites, and licensing requirements, refer to Intel FPGA Software Installation and Licensing.

  The Intel HLS Compiler standalone installation package requires you to have an existing Intel Quartus Prime Pro Edition installation as a target for the Intel HLS Compiler installation.

1.3. Installing the Intel HLS Compiler Pro Edition for Cyclone® V Device Support

Cyclone® V support in the Intel HLS Compiler Pro Edition requires Intel Quartus Prime Standard Edition or Intel Quartus Prime Lite Edition.

The Intel HLS Compiler Pro Edition uses your default Intel Quartus Prime installation when compiling your component for an FPGA device family. You have the following options to install the Intel HLS Compiler Pro Edition for Cyclone V device support:

- Depending on your version of Intel Quartus Prime, upgrade or add the Intel HLS Compiler to your Intel Quartus Prime Standard Edition or Intel Quartus Prime Lite Edition installation:
  - For Intel Quartus Prime Standard or Lite Editions Version 19.1 or earlier, you have an installation of the Intel HLS Compiler Standard Edition that you can upgrade to Intel HLS Compiler Pro Edition.

Upgrading the compiler replaces your current Intel HLS Compiler Standard Edition installation with an Intel HLS Compiler Pro Edition installation in your Intel Quartus Prime Standard Edition or Intel Quartus Prime Lite Edition installation directory.

— For Intel Quartus Prime Standard or Lite Editions Version 20.1 or later, your initial installation of Intel Quartus Prime does not have the Intel HLS Compiler. You must add the Intel HLS Compiler Pro Edition to your Intel Quartus Prime installation.

To upgrade or add the Intel HLS Compiler installation in your Intel Quartus Prime installation:

a. Download the **Intel High Level Synthesis Compiler** installation package from the **Additional Software** tab of the Intel Quartus Prime Download Center (http://fpgasoftware.intel.com/?edition=pro).

b. Follow the instructions for upgrading the Intel HLS Compiler in an existing Intel Quartus Prime installation in one of following sections:

— Installing the Intel HLS Compiler Pro Edition on Linux Systems on page 7

— Installing the Intel HLS Compiler Pro Edition on Microsoft Windows* Systems on page 9

If you have multiple Intel Quartus Prime installations, upgrade or add the Intel HLS Compiler in each installation.

• Install the Intel HLS Compiler Pro Edition into its default installation directory and set the `QUARTUS_ROOTDIR_OVERRIDE` environment variable to point to the Intel Quartus Prime Standard Edition or Intel Quartus Prime Lite Edition installation that you want the Intel HLS Compiler to use for Cyclone V device support.

a. Download the **Intel High Level Synthesis Compiler** installation package from the **Additional Software** tab of the Intel Quartus Prime Download Center (http://fpgasoftware.intel.com/?edition=pro).

b. Run the Intel HLS Compiler standalone installer and accept the default installation location.

If you have multiple Intel Quartus Prime installations, confirm that the `QUARTUS_ROOTDIR_OVERRIDE` environment variable points to the Intel Quartus Prime installation that you want the Intel HLS Compiler Pro Edition to use. The environment variable must be set correctly before you initialize the Intel HLS Compiler Pro Edition environment.

1.4. Installing the Intel HLS Compiler Pro Edition on Linux Systems

You must have administrator privileges to install the Intel HLS Compiler Pro Edition prerequisites. However, the Intel HLS Compiler and Intel Quartus Prime do not require administrator privileges to install.

To install the Intel HLS Compiler on Linux Systems:

1. Confirm that your operating system version is supported by the Intel HLS Compiler:

- CentOS* 7

- Red Hat Enterprise Linux* 7

- SUSE* Linux Enterprise Server 12

2. Depending on your current system and what installation package or packages you have downloaded, you might have to complete some additional steps to prepare your installation:

Only Intel HLS Compiler Installation Package

If you downloaded only the the Intel HLS Compiler installation package for upgrading (or adding) the Intel HLS Compiler in an existing Intel Quartus Prime installation:

a. Ensure that you have Intel Quartus Prime already installed.

b. Take note of the path to your Intel Quartus Prime installation.

   You need this path information to complete the Intel HLS Compiler installation wizard.

c. If you are upgrading your Intel HLS Compiler installation, rename the existing HLS directory in your current Intel Quartus Prime version to keep that version as a backup.

   For example, if you are installing Intel HLS Compiler Version 21.1 into an Intel Quartus Prime Pro Edition Version 20.4 installation, rename `/home/<username>/intelFPGA_pro/20.4/hls` to `/home/<username>/intelFPGA_pro/20.4/hls_old`.

Separate Intel Quartus Prime and Intel HLS Compiler Installation Packages

If you are installing separately downloaded Intel Quartus Prime (from **Individual Files** tab of the Quartus Prime download page at the Download Center for FPGAs) and Intel HLS Compiler (from **Additional Software** tab of the Quartus Prime download page at the Download Center for FPGAs) installation packages:

Ensure that you have both installation packages in the same directory. The Intel Quartus Prime installer detects the Intel HLS Compiler installation package and installs both software packages for you.

Combined Installation Package

If you are installing the complete Intel Quartus Prime and Intel HLS Compiler installation package (from the **Combined Files** tab of the Quartus Prime download page at the Download Center for FPGAs), no additional steps are needed before running the installation package.

3. Install the package that you downloaded in .

   For the standalone Intel HLS Compiler installation package, you are asked to specify the path of the destination Intel Quartus Prime installation.

   For detailed instructions about installing Intel Quartus Prime software, including system requirements, prerequisites, and licensing requirements, refer to Intel FPGA Software Installation and Licensing.

4. Update your Linux repositories with the following command:

```
sudo yum update
```

5. If you want the Intel HLS Compiler to simulate your components with the 32-bit Mentor Graphics ModelSim software provided with Intel Quartus Prime (ModelSim - Intel FPGA Edition), install the required additional 32-bit libraries with the following command:

   - CentOS 7 and Red Hat Enterprise Linux 7

```
$ sudo yum install -y glibc.i686 glibc-devel.i686 libX11.i686 \
               libXext.i686 libXft.i686 libgcc.i686 libgcc.x86_64 \
               libstdc++.i686 libstdc++-devel.i686 ncurses-devel.i686 \
               qt.i686
```

   - SUSE Linux Enterprise Server 12

     Obtain the equivalent or substitute packages listed for CentOS and Red Hat Linux.

6. If you use the Mentor Graphics ModelSim software provided with Intel Quartus Prime, add the path to ModelSim to your `PATH` environment variable

   For example:

```
$ export PATH=$PATH:<quartus_installdir>/modelsim_ase/bin
```

7. Optional: If you plan to use Platform Designer to integrate your component with a system, add the path to Platform Designer to your `PATH` environment variable.

   For example:

```
$ export PATH=$PATH:<quartus_installdir>/qsys/bin
```

After completing these steps, the Intel HLS Compiler is installed on your system. Before you can compile your component with the Intel HLS Compiler `i++` command, you must initialize your Intel HLS Compiler environment for the `i++` command to run successfully. For details, see Initializing the Intel HLS Compiler Pro Edition Environment on page 11.

Important: The Intel HLS Compiler uses the `<quartus_installdir>/gcc` as its toolchain directory. Use this installation of GCC for all your HLS-related design work.

1.5. Installing the Intel HLS Compiler Pro Edition on Microsoft Windows* Systems

To install the Intel HLS Compiler on Microsoft Windows* Systems:

1. Confirm that your operating system version is supported by the Intel HLS Compiler:

   - Microsoft Windows 10
   - Microsoft Windows Server* 2016

2. Install one of the following software products:

   — Microsoft Visual Studio 2017 Professional
   — Microsoft Visual Studio 2017 Community

Important: The Intel HLS Compiler software does not support versions of Microsoft Visual Studio other than those specified for the edition of the software.

If you have multiple versions of Visual Studio, Microsoft recommends installing Visual Studio* versions in the order in which the versions were released. For example, install Visual Studio 2010 before installing Visual Studio 2015. For details, see Install Visual Studio Versions Side-by-Side in the MSDN Library.

3. Depending on your current system and what installation package or packages you have downloaded, you might have to complete some additional steps to prepare your installation:

Only Intel HLS Compiler Installation Package	If you are upgrading (or adding) the Intel HLS Compiler in an existing Intel Quartus Prime installation:

   a. Ensure that you have Intel Quartus Prime already installed.

   b. Take note of the path to your Intel Quartus Prime installation.

   You need this path information to complete the Intel HLS Compiler installation wizard.

   c. If you are upgrading your Intel HLS Compiler installation, rename the existing HLS directory in your current Intel Quartus Prime version to keep that version as a backup.

   For example, if you are installing Intel HLS Compiler Version 21.1 into an Intel Quartus Prime Pro Edition Version 20.4 installation, rename `C:\intelFPGA_pro\20.4\hls` to `C:\intelFPGA_pro\20.4\hls_old`.

Separate Intel Quartus Prime and Intel HLS Compiler Installation Packages	If you are installing separately downloaded Intel Quartus Prime (from **Individual Files** tab of the Quartus Prime download page at the Download Center for FPGAs) and Intel HLS Compiler (from **Additional Software** tab of the Quartus Prime download page at the Download Center for FPGAs) installation packages:

   Ensure that you have both installation packages in the same directory. The Intel Quartus Prime installer detects the Intel HLS Compiler installation package and installs both software packages for you.

Combined Installation Package	If you are installing the complete Intel Quartus Prime and Intel HLS Compiler installation package (from the **Combined Files** tab of the Quartus Prime download page at the Download Center for FPGAs), no additional steps are needed before running the installation package.

4. Install the package that you downloaded in Downloading the Intel HLS Compiler Pro Edition on page 6.

For the standalone Intel HLS Compiler installation package, you are asked to specify the path of the destination Intel Quartus Prime installation.

For detailed instructions about installing Intel Quartus Prime software, including system requirements, prerequisites, and licensing requirements, refer to the Intel FPGA Software Installation and Licensing.

5. If you use the Mentor Graphics ModelSim software provided with Intel Quartus Prime, add the path to ModelSim to your `PATH` environment variable.

   For example:

   ```
   set "PATH=%PATH%:<quartus_installdir>\modelsim_ase\win32aloem"
   ```

   To make these changes permanent, edit the system environment variables from the Windows Control Panel.

6. Optional: If you plan to use Platform Designer to integrate your component with a system, add the path to Platform Designer to your `PATH` environment variable.

   For example:

   ```
   set "PATH=%PATH%:<quartus_installdir>\qsys\bin"
   ```

   To make these changes permanent, edit the system environment variables from the Windows Control Panel.

After completing these steps, the Intel HLS Compiler is installed on your system. Before you can compile your component with the Intel HLS Compiler `i++` command, you must initialize your Intel HLS Compiler environment for the `i++` command to run successfully.

1.6. Initializing the Intel HLS Compiler Pro Edition Environment

Before you can compile your component with the Intel HLS Compiler `i++` command, a number of environment variables must be set for the `i++` command to run successfully.

The Intel HLS Compiler environment initialization script applies only to the environment variable settings in your *current* terminal or command prompt session. You must initialize the Intel HLS Compiler environment each time that you start a terminal or command prompt session to develop your design.

To initialize your current terminal or command prompt session so that you can run the Intel HLS Compiler:

- On Linux systems, initialize your environment as follows:

  a. (Optional) Set the `QUARTUS_ROOTDIR_OVERRIDE` environment variable to point at the Intel Quartus Prime installation that Intel HLS Compiler uses for Quartus compiles.

     Typically, you set the `QUARTUS_ROOTDIR_OVERRIDE` environment variable when you have one Intel HLS Compiler installation that you want to use with multiple Intel Quartus Prime installations.

     For more information about the `QUARTUS_ROOTDIR_OVERRIDE` environment variable and managing multiple installations of Intel Quartus Prime, refer to "Managing Multiple Versions and Copies of Intel FPGA Software" in *Intel FPGA Software Installation and Licensing*.

  b. Start a terminal session and run the following command from the command prompt:

     ```
     <quartus_installdir>/hls/init_hls.sh
     ```

     Where *<quartus_installdir>* is the path to your Intel Quartus Prime installation. For example, `/home/<username>/intelFPGA_pro/21.1`.

     This script creates a subshell that does not change the environment in your initial working shell.

     If you want modify your current working shell, use the `source` command to invoke the script:

     ```
     source <quartus_installdir>/hls/init_hls.sh
     ```

  The environment initialization script shows the environment variables that it set, and you can now run the `i++` command from this terminal session.

- On Windows systems, initialize your environment as follows:

  a. (Optional) Set the `QUARTUS_ROOTDIR_OVERRIDE` environment variable to point at the Intel Quartus Prime installation that Intel HLS Compiler uses for Quartus compiles.

     Typically, you set the `QUARTUS_ROOTDIR_OVERRIDE` environment variable when you have one Intel HLS Compiler installation that you want to use with multiple Intel Quartus Prime installations.

     For more information about the `QUARTUS_ROOTDIR_OVERRIDE` environment variable and managing multiple installations of Intel Quartus Prime, refer to "Managing Multiple Versions and Copies of Intel FPGA Software" in *Intel FPGA Software Installation and Licensing*.

  b. Start a Windows Command Prompt (`cmd.exe`) session and run the following command from the command prompt:

     ```
     <quartus_installdir>\hls\init_hls.bat
     ```

     Where *<quartus_installdir>* is the path to your Intel Quartus Prime installation. For example, `C:\intelFPGA_pro\21.1`.

  You can now run the `i++` command from this command prompt session.

Tip: To set the environment variables permanently, follow your operating system's standard procedure for making persistent changes to environment variable settings. Review the output of the environment initialization script to determine the environment variables to set permanently.

intel.

2. High Level Synthesis (HLS) Design Examples and Tutorials

The Intel High Level Synthesis (HLS) Compiler Pro Edition includes design examples and tutorials to provide you with example components and demonstrate ways to model or code your components to get the best results from the Intel HLS Compiler for your design.

High Level Synthesis Design Examples

The high level synthesis (HLS) design examples give you a quick way to see how various algorithms can be effectively implemented to get the best results from the Intel HLS Compiler.

You can find the HLS design examples in the following location:

```
<quartus_installdir>/hls/examples/<design_example_name>
```

Where *<quartus_installdir>* is the directory where you installed the Intel Quartus Prime Design Suite. For example, `/home/<username>/intelFPGA_pro/21.1` or `C:\intelFPGA_pro\21.1`.

For instructions on running the examples, see the following sections:

Table 2.   **HLS design examples**

Focus area	Name	Description
Linear algebra	`QRD`	Uses the Modified Gram-Schmidt algorithm for QR factorization of a matrix.
Signal processing	`interp_decim_filter`	Implements a simple and efficient interpolation/decimation filter.
Simple design	`counter`	Implements a simple and efficient 32-bit counter component.
Video processing	`YUV2RGB`	Implements a basic YUV422 to RGB888 color space conversion.
Video processing	`image_downsample`	Implements an image downsampling algorithm to scale an image to a smaller size using bilinear interpolation.

HLS Design Tutorials

The HLS design tutorials show you important HLS-specific programming concepts as well demonstrating good coding practices.

Each tutorial has a README file that gives you details about what the tutorial covers and instructions on how to run the tutorial.

Table 3. **Arbitrary precision datatypes design tutorials**

Name	Description
You can find these tutorials in the following location on your Intel Quartus Prime system:	
`<quartus_installdir>`/hls/examples/tutorials/ac_datatypes	
`ac_fixed_constructor`	Demonstrates the use of the `ac_fixed` constructor where you can get a better QoR by using minor variations in coding style.
`ac_fixed_math_library`	Demonstrates the use of the Intel HLS Compiler `ac_fixed_math` fixed point math library functions.
`ac_int_basic_ops`	Demonstrates the operators available for the `ac_int` class.
`ac_int_overflow`	Demonstrates the usage of the `DEBUG_AC_INT_WARNING` and `DEBUG_AC_INT_ERROR` keywords to help detect overflow during emulation runtime.
You can find these tutorials in the following location on your Intel Quartus Prime system:	
`<quartus_installdir>`/hls/examples/tutorials/hls_float	
`1_reduced_double`	Demonstrates how your application can benefit from `hls_float` by changing the underlining type from `double` to `hls_float<11, 44>` (reduced double).
`2_explicit_arithmetic`	Demonstrates how to use the explicit versions of `hls_float` binary operators to perform floating-point arithmetic operations based on your needs.
`3_conversions`	Demonstrates when conversions appear in designs with `hls_float` types and how to use different conversion modes to generate compile-type constants using various `hls_float` types.

Table 4. **Component memories design tutorials**

Name	Description
You can find these tutorials in the following location on your Intel Quartus Prime system:	
`<quartus_installdir>`/hls/examples/tutorials/component_memories	
`attributes_on_mm_slave_arg`	Demonstrates how to apply memory attributes to Avalon® Memory Mapped (MM) slave arguments.
`exceptions`	Demonstrates how to use memory attributes on constants and `struct` members.
`memory_bank_configuration`	Demonstrates how to control the number of load/store ports of each memory bank and optimize your component area usage, throughput, or both by using one or more of the following memory attributes: • `hls_max_replicates` • `hls_singlepump` • `hls_doublepump` • `hls_simple_dual_port_memory` • `non_power_of_two_memory` • `non_trivial_initialization`
`memory_geometry`	Demonstrates how to split your memory into banks and control the number of load/store ports of each memory bank by using one or more of the following memory attributes: • `hls_bankwidth` • `hls_numbanks` • `hls_bankbits`

continued...

Name	Description
memory_implementation	Demonstrates how to implement variables or arrays in registers, MLABs, or RAMs by using the following memory attributes: • hls_register • hls_memory • hls_memory_impl
memory_merging	Demonstrates how to improve resource utilization by implementing two logical memories as a single physical memory by merging them depth-wise or width-wise with the hls_merge memory attribute.
non_power_of_two_memory	Demonstrates how to use the force_pow2_depth memory attribute to control the padding of memories that are non-power-of-two deep, and how that impacts the FPGA memory resource usage.
non_trivial_initialization	Demonstrates how to use the C++ keyword constexpr to achieve efficient initialization of read-only variables.
static_var_init	Demonstrates how to control the initialization behavior of statics in a component using the hls_init_on_reset or hls_init_on_powerup memory attribute.

Table 5.   Interface design tutorials

Name	Description
\<_quartus_installdir_>/hls/examples/tutorials/interfaces	You can find these tutorials in the following location on your Intel Quartus Prime system:
overview	Demonstrates the effects on quality-of-results (QoR) of choosing different component interfaces even when the component algorithm remains the same.
explicit_streams_buffer	Demonstrates how to use explicit stream_in and stream_out interfaces in the component and testbench.
explicit_streams_packets_empty	Demonstrates how to use the usesPackets, usesEmpty, and firstSymbolInHighOrderBits stream template parameters.
explicit_streams_packets_ready_valid	Demonstrates how to use the usesPackets, usesValid, and usesReady stream template parameters.
mm_master_testbench_operators	Demonstrates how to invoke a component at different indicies of an Avalon Memory Mapped (MM) Master (mm_master class) interface.
mm_slaves	Demonstrates how to create Avalon-MM Slave interfaces (slave registers and slave memories).
mm_slaves_double_buffering	Demonstrates the effect of using the hls_readwrite_mode macro to control how memory masters access the slave memories
mm_slaves_csr_volatile	Demonstrates the effect of using volatile keyword to allow concurrent slave memory accesses while your component is running.
multiple_stream_call_sites	Demonstrates the benefits of using multiple stream call sites.
pointer_mm_master	Demonstrates how to create Avalon-MM Master interfaces and control their parameters.
stable_arguments	Demonstrates how to use the stable attribute for unchanging arguments to improve resource utilization.

Table 6.   Best practices design tutorials

Name	Description
You can find these tutorials in the following location on your Intel Quartus Prime system: `<quartus_installdir>/hls/examples/tutorials/best_practices`	
`ac_datatypes`	Demonstrates the effect of using `ac_int` datatype instead of `int` datatype.
`control_of_dsp_usage`	Demonstrates the effects of controlling whether some supported data types and math functions implemented by DSPs or soft logic with the `--dsp-mode` option of the `i++` command and the `ihc::math_dsp_control` function.
`const_global`	Demonstrates the performance and resource utilization improvements of using `const` qualified global variables.
`divergent_loops`	Demonstrates a source-level optimization for designs with divergent loops
`floating_point_contract`	Demonstrates how to use the `fp_contract` option to improve the performance of your design for double-precision floating-point operations.
`floating_point_ops`	Demonstrates the impact of `--fpc` and `--fp-relaxed` flags in i++ on floating point operations using a 32-tap finite impulse response (FIR) filter design that is optimized for throughput.
`fpga_reg`	Demonstrates how to use the `fpga_reg` macro to precisely tune pipelining in your design.
`hyper_optimized_handshaking`	Demonstrates how to use the `--hyper-optimized-handshaking` option of the Intel HLS Compiler i++ command.
`loop_coalesce`	Demonstrates the performance and resource utilization improvements of using `loop_coalesce` pragma on nested loops. While the `#pragma loop_coalesce` is provided with both Standard and Pro edition, the design tutorial is provided only with Pro edition.
`loop_fusion`	Demonstrates the latency and resource utilization improvements of loop fusion.
`loop_memory_dependency`	Demonstrates breaking loop carried dependencies using the `ivdep` pragma.
`lsu_control`	Demonstrates the effects of controlling the types of LSUs instantiated for variable-latency Avalon MM Master interfaces.
`parallelize_array_operation`	Demonstrates how to improve f_{MAX} by correcting a bottleneck that arises when performing operations on an array in a loop.
`optimize_ii_using_hls_register`	Demonstrates how to use the `hls_register` attribute to reduce loop II and how to use `hls_max_concurrency` to improve component throughput
`parameter_aliasing`	Demonstrates the use of the `__restrict` keyword on component arguments.
`random_number_generator`	Demonstrates how to use the random number generator library.
`reduce_exit_fifo_width`	Demonstrates how to improve f_{MAX} by reducing the width of the FIFO belonging to the exit node of a stall-free cluster
`relax_reduction_dependency`	Demonstrates a method to reduce the II of a loop that includes a floating point accumulator, or other reduction operation that cannot be computed at high speed in a single clock cycle.
`remove_loop_carried_dependency`	Demonstrates how you can improve loop performance by removing accesses to the same variable across nested loops.
`resource_sharing_filter`	Demonstrates an optimized-for-area variant of a 32-tap finite impulse response (FIR) filter design
`set_component_target_fmax_1`	Demonstrates how to the target f_{MAX} in various ways by leveraging the Loop Analysis report in the High-Level Design Reports.

continued...

Name	Description
`set_component_target_fmax_2`	Demonstrates how the compiler handles the tradeoff between f_{MAX} and II based on the presence or absence of the `hls_scheduler_target_fmax_mhz` component attribute and the `ii` loop pragma.
`shift_register`	Demonstrates the recommended coding style for implementing shift registers.
`sincos_func`	Demonstrates the effects of using `sinpi` or `cospi` functions in your component instead of `sin` or `cos` functions.
`single_vs_double_precision_math`	Demonstrates the effect of using single precision literals and functions instead of double precision literals and functions.
`stall_enable`	Demonstrates how to replace stall-free clusters with stall-enabled clusters to improve latency in some small designs.
`struct_interface`	Demonstrates how to use `ac_int` to implement interfaces with no padding bits.
`submnormal_and_rounding`	Demonstrates the effects of use the `--daz` and `--rounding` i++ command options.
`swap_vs_copy`	Demonstrates the impact of using deep copying with registers on the performance and resource utilization of a component design.
`triangular_loop`	Demonstrates a method for describing triangular loop patterns with dependencies.

Table 7.   Usability design tutorials

Name	Description
You can find these tutorials in the following location on your Intel Quartus Prime system: `<quartus_installdir>`/hls/examples/tutorials/usability	
`compiler_interoperability`	Demonstrates how to build your design using testbench code compiled with the Intel HLS Compiler, GCC, or Microsoft Visual Studio* and component code compiled separately with the Intel HLS Compiler).
`enqueue_call`	Demonstrates how to run components asynchronously and exercise their pipeline performance in the test bench using enqueue functionality.
`platform_designer_2xclock`	Demonstrates the recommended clock and reset generation for a component with a `clock2x` input.
`platform_designer_stitching`	Demonstrates how to combine multiple components to function as a single cohesive design.

Table 8.   System of tasks design tutorials

Name	Description
You can find these tutorials in the following location on your Intel Quartus Prime system: `<quartus_installdir>`/hls/examples/tutorials/system_of_tasks	
`balancing_loop_delay`	Demonstrates how to improve the throughput of a component that uses a system of tasks by buffering streams.
`balancing_pipeline_latency`	Demonstrates how to improve the throughput of a component that uses a system of tasks by buffering streams.
`interfaces_sot`	Demonstrates how to transfer information between, into, and out of tasks using Avalon streaming and Avalon memory-mapped master interfaces.
`internal_stream`	Demonstrates how to use "internal streams" in HLS tasks with the `ihc::stream` object.

continued...

Send Feedback

Name	Description
launch_and_collect_capacity	Demonstrates how to use the capacity template parameter of the ihc::launch and ihc::collect functions to improve throughput in components that have systems of tasks.
parallel_loop	Demonstrates how you can run sequential loops in a pipelined manner by using a system of HLS tasks in your component.
resource_sharing	Demonstrates how you can share expensive compute blocks in your component to save area usage.
task_reuse	Demonstrates how to invoke multiple copies of the same task function.

Table 9.   HLS Libraries design tutorials

Name	Description
You can find these tutorials in the following location on your Intel Quartus Prime system: *<quartus_installdir>*/hls/examples/tutorials/libraries	
basic_rtl_library_flow	Demonstrates the process of developing an RTL library and using it in an HLS component.
rtl_struct_mapping	Demonstrates how to obtain a mapping from C++ struct fields to bit-slices of RTL module interface signals.

Table 10.   HLS Loop Control tutorials

Name	Description
You can find these tutorials in the following location on your Intel Quartus Prime system: *<quartus_installdir>*/hls/examples/tutorials/loop_controls	
max_interleaving	Demonstrates a method to reduce the area utilization of a loop that meets the following conditions: • The loop has an II > 1 • The loop is contained in a pipelined loop • The loop execution is serialized across the invocations of the pipelined loop
small_speculated_iterations	Demonstrates how decreasing the number of speculated iterations improves latency when a loop body has low latency and is expected to be frequently invoked.
speculated_iterations	Demonstrates how increasing the number of speculated iterations improves II when the exit condition calculation is the bottleneck preventing a lower II.

2.1. Running an Intel HLS Compiler Design Example (Linux)

To run an Intel HLS Compiler design example on Linux systems:

1. Start a terminal session and initialize the Intel HLS Compiler environment.

   For instructions how to initialize the environment, see Initializing the Intel HLS Compiler Pro Edition Environment on page 11.

2. Navigate to the *<quartus_installdir>*/hls/examples/ *<design_example_name>* directory, where *<quartus_installdir>* is the directory where you installed Intel Quartus Prime software.

For example, /home/<*username*>/intelFPGA_pro/21.1.

3. Run the `make test-x86-64` command. This command compiles the C++ source code to an x86-64 binary executable. Then, run the generated executable on your CPU.
Expected outcome after you run the `make test-x86-64` command:

- The console displays the command used to generate the binary. For example, `i++ -march=x86-64 -o test-x86-64 <source_files>`.

- The HLS compiler creates an executable file (for example, `test-x86-64`) in the current working directory.

- The console displays the output of the executable to signify a successful execution.

```
$ make test-x86-64
i++ MGS.cpp QRD_Testbench.cpp TestbenchHelpers.cpp  -ffp-contract=fast -ffp-
reassoc -march=x86-64 -o test-x86-64
+-----------------------------------+
| Run ./test-x86-64 to execute the test. |
+-----------------------------------+
```

4. Run the `make test-fpga` command. The command compiles the C++ source code to a hardware executable and then runs a simulation of the generated HDL.
Expected outcome after you run the `make test-fpga` command:

- The console displays the command it uses to generate the testbench binary and the contents of the project directory. For example, `i++ -march="<FPGA_family_or_part_number>" <source_files> -o test-fpga`.

- The HLS compiler creates a `.prj` directory (for example, `test-fpga.prj`) in the current working directory.

- The console displays the output of the executable to signify a successful execution.

```
$ make test-fpga
i++ MGS.cpp QRD_Testbench.cpp TestbenchHelpers.cpp  -v -ffp-contract=fast -
ffp-reassoc -march=Arria10 -o test-fpga
Target FPGA part name:   10AX115U1F45I1SG
Target FPGA family name: Arria 10
Target FPGA speed grade: -2
Analyzing MGS.cpp for testbench generation
Creating x86-64 testbench
Analyzing MGS.cpp for hardware generation
Analyzing QRD_Testbench.cpp for testbench generation
Creating x86-64 testbench
Analyzing QRD_Testbench.cpp for hardware generation
Analyzing TestbenchHelpers.cpp for testbench generation
Creating x86-64 testbench
Analyzing TestbenchHelpers.cpp for hardware generation
Optimizing component(s) and generating Verilog files
Generating cosimulation support
Generating simulation files for components: qrd
HLS simulation directory: /data/username/HLS_Trainings/examples/QRD/test-
fpga.prj/verification.
Linking x86 objects
+-----------------------------------+
| Run ./test-fpga to execute the test. |
+-----------------------------------+
```

Intel High Level Synthesis Compiler Pro Edition: Getting Started Guide

Send Feedback

intel.

2.2. Running an Intel HLS Compiler Design Example (Windows)

To run an Intel HLS Compiler design example on Windows systems:

1. Start a terminal session and initialize the Intel HLS Compiler environment.

   For instructions how to initialize the environment, see Initializing the Intel HLS Compiler Pro Edition Environment on page 11.

2. Navigate to the `<quartus_installdir>\hls\examples \<design_example_name>` directory, where `<quartus_installdir>` is the directory where you installed Intel Quartus Prime software.

   For example, `C:\intelFPGA_pro\21.1`.

3. Run the `build.bat test-x86-64`. This command compiles the C++ source code to an x86-64 binary executable. Then, run the generated executable on your CPU.
   Expected outcome after you run the `build.bat test-x86-64` command:

   - The console displays the command it uses to generate the binary. For example, `i++ –march=x86-64 –o test-x86-64 <source_files>`.

   - The HLS compiler creates an executable file (for example, `test-x86-64.exe`) in the current working directory.

   - The console displays the output of the executable to signify a successful execution.

   ```
   C:\intelFPGA_pro\21.1\hls\examples\QRD>build.bat test-x86-64
   i++ –ffp-contract=fast –ffp-reassoc –march=x86-64 MGS.cpp QRD_Testbench.cpp
   TestbenchHelpers.cpp –o test-x86-64.exe
   Run test-x86-64.exe to execute the test.
   ```

4. Run the `build.bat test-fpga` command. The command compiles the C++ source code to a hardware executable and then runs a simulation of the generated HDL.
   Expected outcome after you run the `build.bat test-fpga` command:

   - The console displays the command it uses to generate the testbench binary and the contents of the project directory. For example,
     `i++ –march="<FPGA_family_or_part_number>" <source_files> –o test-fpga`.

   - The HLS compiler creates a `.prj` directory (for example, `test-fpga.prj`) in the current working directory.

   - The console displays the output of the executable to signify a successful execution.

   ```
   C:\intelFPGA_pro\21.1\hls\examples\QRD>build.bat test-fpga
   i++ –ffp-contract=fast –ffp-reassoc –march=Arria10 MGS.cpp
   QRD_Testbench.cpp TestbenchHelpers.cpp –o test-fpga.exe
   Run test-fpga.exe to execute the test.
   ```

intel.

3. Troubleshooting the Setup of the Intel HLS Compiler

This section provides information that can help you troubleshoot problems you might encounter when setting up the HLS compiler.

3.1. Intel HLS Compiler Licensing Issues

The Intel High Level Synthesis (HLS) Compiler is licensed as part of your Intel Quartus Prime license. However, the Intel HLS Compiler depends on ModelSim software. If you use a version of ModelSim software other than ModelSim - Intel FPGA Edition or ModelSim - Intel FPGA Starter Edition, ensure that your version of ModelSim software is licensed correctly.

In some cases, you might encounter problems with the licensing for ModelSim software.

3.1.1. ModelSim Licensing Error Messages

The HLS compiler issues error messages if it cannot locate the license for the installed version of ModelSim software.

If the HLS compiler fails to locate the ModelSim software license, it issues the following error message when you compile your design to the FPGA architecture:

```
$ i++ -march="<FPGA_family_or_part_number>" program.cpp
HLS Elaborate cosim testbench. FAILED.
See ./a.prj/a.log for details.
Error: Missing simulator license. Either:
1) Ensure you have a valid ModelSim license
2) Use the --simulator none flag to skip the verification flow
```

Common causes for these errors include:

- Missing, expired, or invalid licenses
- Incorrect license server name in the `license.dat` file
- Unspecified or incorrectly-specified license location

Note: The running speed of the HLS compiler might decrease if the compiler has to search the network for missing or corrupted licenses. If this problem occurs, correct the license file or license location accordingly.

3.1.2. LM_LICENSE_FILE Environment Variable

Intel and third-party software use the *LM_LICENSE_FILE* environment variable to specify the locations of license files or license servers. For example, both the Intel Quartus Prime software and the ModelSim software use the *LM_LICENSE_FILE* variable to specify the locations of their licenses.

**ISO
9001:2015
Registered**

Note: The time it takes for your development machine to communicate with the license server directly affects compilation time. If your *LM_LICENSE_FILE* environment variable setting includes paths to many license servers, or if the license server is hosted in a distant locale, you will notice a significant increase in compilation time.

On Linux or UNIX systems, insert a colon (:) after each license file or license server location that you append to the *LM_LICENSE_FILE* environment variable.

On Windows systems, insert a semicolon (;) after each license file or license server location that you append to the *LM_LICENSE FILE* environment variable.

Note: When modifying the *LM_LICENSE_FILE* setting to include the locations of your software licenses, do not remove any existing license locations appended to the variable.

3.1.2.1. ModelSim Software License-Specific Considerations

When setting up the ModelSim software license, you need to append the license location to the *LM_LICENSE_FILE* environment variable. However, you can also append the location of the ModelSim software license to the *MGLS_LICENSE_FILE* environment variable.

For Mentor Graphics applications, including the ModelSim software, you can specify the paths to license files and license servers in five different locations. If you specify paths to license files or license servers in multiple locations, the following search order is used to find the first valid path:

- *MGLS_LICENSE_FILE* environment variable you set in the user environment

- *MGLS_LICENSE_FILE* environment variable you set in the registry

- *LM_LICENSE_FILE* environment variable you set in the environment

- *LM_LICENSE_FILE* environment variable you set in the registry

- `<path to FLEXlm>\license.dat`, where *<path to FLEXlm>* is the default location of the FLEXlm license file.

When you install a Mentor Graphics product license on a computer, the *MGLS_LICENSE_FILE* environment variable settings take precedence over the *LM_LICENSE_FILE* environment variable settings. If you set both environment variables, set *LM_LICENSE_FILE* to point to the ModelSim license server and set *MGLS_LICENSE_FILE* to only point to the license server for other Mentor Graphics applications. If you only use the *MGLS_LICENSE_FILE* environment variable, ensure that the ModelSim license server and the license servers for other Mentor Graphics applications are on the same machine.

intel.

A. Intel HLS Compiler Pro Edition Getting Started Guide Archives

Intel HLS Compiler Version	Title
20.4	Intel High Level Synthesis Compiler Pro Edition Getting Started Guide
20.3	Intel High Level Synthesis Compiler Pro Edition Getting Started Guide
20.2	Intel High Level Synthesis Compiler Pro Edition Getting Started Guide
20.1	Intel High Level Synthesis Compiler Pro Edition Getting Started Guide
19.4	Intel High Level Synthesis Compiler Pro Edition Getting Started Guide
19.3	Intel High Level Synthesis Compiler Getting Started Guide
19.2	Intel High Level Synthesis Compiler Getting Started Guide
19.1	Intel High Level Synthesis Compiler Getting Started Guide
18.1.1	Intel High Level Synthesis Compiler Getting Started Guide
18.1	Intel High Level Synthesis Compiler Getting Started Guide
18.0	Intel High Level Synthesis Compiler Getting Started Guide
17.1.1	Intel High Level Synthesis Compiler Getting Started Guide
17.1	Intel High Level Synthesis Compiler Getting Started Guide

ISO 9001:2015 Registered

B. Document Revision History for Intel HLS Compiler Pro Edition Getting Started Guide

Document Version	Intel Quartus Prime Pro Edition Version	Changes
2021.03.29	21.1	• Revised the description of the `compiler_interoperability` tutorial in High Level Synthesis (HLS) Design Examples and Tutorials on page 14. • Added `launch_and_collect_capacity` tutorial to High Level Synthesis (HLS) Design Examples and Tutorials on page 14. • Removed the documented restrictions for SUSE Linux Enterprise Server 12 and ModelSim - Intel FPGA Edition. ModelSim - Intel FPGA Edition has been supported on SUSE Linux Enterprise Server 12 since Version 20.3.
2020.12.14	20.4	• Added Installing the Intel HLS Compiler Pro Edition for Cyclone® V Device Support on page 6, • Revised Initializing the Intel HLS Compiler Pro Edition Environment on page 11 to include optional instructions about setting the `QUARTUS_ROOTDIR_OVERRIDE` environment variable.

2020.09.28	20.3	• Added CentOS 7 to the list of support operating systems in Installing the Intel HLS Compiler Pro Edition on Linux Systems on page 7.
		• Added Microsoft Windows Server* 2016 to the list of supported operating systems in Installing the Intel HLS Compiler Pro Edition on Microsoft Windows* Systems on page 9.
		• Added the following tutorials to High Level Synthesis (HLS) Design Examples and Tutorials on page 14:
		  — `set_target_fmax_1`
		  — `set_target_fmax_2`
		  — `small_speculated_iterations`
		  — `speculated_iterations`
		• Removed Red Hat Enterprise Linux* 6 (or a community equivalent) from the list of supported operating systems in Installing the Intel HLS Compiler Pro Edition on Linux Systems on page 7.
		
		Support for Red Hat Enterprise Linux 6 is removed from the Intel HLS Compiler Pro Edition Version 20.3.
		• Removed Microsoft Windows 7 SP1 and Microsoft Windows 8.1 as supported operation systems from Installing the Intel HLS Compiler Pro Edition on Microsoft Windows* Systems on page 9.

continued...

Document Version	Intel Quartus Prime Pro Edition Version	Changes
2020.06.22	20.2	• Added the following tutorials to High Level Synthesis (HLS) Design Examples and Tutorials on page 14: — `remove_loop_carried_dependency` — `stall_enable`
2020.04.13	20.1	• Revised Linux prerequisites in Intel High Level Synthesis Compiler Pro Edition Prerequisites on page 4 and Installing the Intel HLS Compiler Pro Edition on Linux Systems on page 7. • Added the following tutorials to High Level Synthesis (HLS) Design Examples and Tutorials on page 14: — `parallelize_array_operation` — `optimize_ii_using_hls_register` — `reduce_exit_fifo_width` — `mm_slaves_csr_volatile` — `mm_slaves_double_buffering` — `non_power_of_two_memory` — `non_trivial_initialization` — `stall_enable` — `subnormal_and_rounding` • Removed *Installing GCC Offline*. This information is not required for V20.1 and later.
2019.12.16	19.4	• Removed information about Intel HLS Compiler Standard Edition. For installation and configuration information for the Intel HLS Compiler Standard Edition, see Intel HLS Compiler Standard Edition Getting Started Guide. • Added *Installing GCC Offline*.

Document Revision History for Intel HLS Compiler Getting Started Guide

Previous versions of the Intel HLS Compiler Getting Started Guide contained information for both Intel HLS Compiler Standard Edition and Intel HLS Compiler Pro Edition.

Send Feedback

Document Version	Intel Quartus Prime Version	Changes
2019.09.30	19.3	• **PRO** Removed the following tutorials: — `bank_bits` — `mm_slave` — `rom` — `struct_member_attributes` • Added the following tutorials: — `memory_bank_configuration` — `memory_geometry` — `memory_implementation` — `memory_merging` — `static_var_init` — `attributes_on_mm_slave_arg` — `exceptions` — `lsu_control` — `relax_reduction_dependency` • **PRO** Added Microsoft Visual Studio 2017 Professional and Microsoft Visual Studio 2017 Community as supported C++ compilers. • **PRO** Removed Microsoft Visual Studio 2015 Professional and Microsoft Visual Studio 2015 Community as supported C++ compilers.
2019.07.01	19.2	• Updated Installing the Intel HLS Compiler Pro Edition on Linux Systems on page 7 to clarify the command to run to install the 32-bit libraries required by ModelSim - Intel FPGA Edition.
2019.04.01	19.1	• The Intel HLS Compiler can now be installed separately from Intel Quartus Prime. The following sections were added or updated: — Downloading the Intel HLS Compiler Pro Edition on page 6 — Installing the Intel HLS Compiler Pro Edition on Linux Systems on page 7 — Installing the Intel HLS Compiler Pro Edition on Microsoft Windows* Systems on page 9 • The Intel HLS Compiler is now backwards-compatible with earlier versions of Intel Quartus Prime. See Intel HLS Compiler Pro Edition Backwards Compatibility on page 5 for details. • Updated Initializing the Intel HLS Compiler Pro Edition Environment on page 11 to include the requirement that you must initialize your Intel HLS Compiler environment on Windows operating systems from a Visual Studio x64 Native Tools command prompt. • Revised and updated the list of tutorials that are provided with the Intel HLS Compiler in High Level Synthesis (HLS) Design Examples and Tutorials on page 14.
2018.12.24	18.1	• **PRO** Added Microsoft Visual Studio 2015 Community to the list of supported C++ compilers on Microsoft Windows systems. • Added step to add path to Mentor Graphics ModelSim software provided with Intel Quartus Prime to operating system `PATH` environment variable in Installing the Intel HLS Compiler Pro Edition on Linux Systems on page 7 and Installing the Intel HLS Compiler Pro Edition on Microsoft Windows* Systems on page 9.

continued...

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1	• **PRO** The Intel HLS Compiler has a new front end. For a summary of the changes introduced by this new front end, see *Improved Intel HLS Compiler Front End* in the Intel High Level Synthesis Compiler Version 18.1 Release Notes. • **PRO** The Intel HLS Compiler provided with Intel Quartus Prime Pro Edition has new prerequisites. Review Intel High Level Synthesis Compiler Pro Edition Prerequisites on page 4 to learn more. • **PRO** The installation instructions for Linux systems have changed. See Installing the Intel HLS Compiler Pro Edition on Linux Systems on page 7 for details. • **PRO** The `best_practices/parameter_aliasing` tutorial description in High Level Synthesis (HLS) Design Examples and Tutorials on page 14 changed to cover the `__restrict` keyword. The `restrict` keyword is no longer supported in the Intel HLS Compiler Pro Edition. • **PRO** Removed the `best_practices/integer_promotion` tutorial. Integer promotion is now done by default when use the Intel HLS Compiler Pro Edition.
2018.05.07	18.0	• Starting with Intel Quartus Prime Version 18.0, the features and devices supported by the Intel HLS Compiler depend on what edition of Intel Quartus Prime you have. Intel HLS Compiler publications now use icons to indicate content and features that apply only to a specific edition as follows: **PRO** Indicates that a feature or content applies only to the Intel HLS Compiler provided with Intel Quartus Prime Pro Edition. **STD** Indicates that a feature or content applies only to the Intel HLS Compiler provided with Intel Quartus Prime Standard Edition. • **PRO** Added the following tutorials to the list of tutorials in High Level Synthesis (HLS) Design Examples and Tutorials on page 14: — `interfaces/explicit_streams_packets_empty` — `interfaces/explicit_streams_ready_latency` — `best_practices/ac_datatypes` — `best_practices/loop_coalesce` — `best_practices/random_number_generator` • **PRO** Renamed the following tutorials to reflect some Intel Quartus Prime component name changes: — `usability/qsys_2xclock` is now `usability/platform_designer_2xclock` — `usability/qsys_stitching` is now `usability/platform_designer_stitching`
2017.12.22	17.1.1	• Added the `interfaces/overview` tutorial to the list of tutorials in High Level Synthesis (HLS) Design Examples and Tutorials on page 14.
2017.12.08	17.0	• Updated the Mentor Graphics ModelSim software requirements to include the required Red Hat development tools packages.

continued...

Send Feedback

Document Version	Intel Quartus Prime Version	Changes
2017.11.06	17.0	• The Intel High Level Synthesis (HLS) Compiler is now part of Intel Quartus Prime Design Suite, resulting in the following changes: — Revised to document to reflect that you now get the Intel HLS Compiler by installing Intel Quartus Prime software. — Removed most licensing information. Licensing the Intel HLS Compiler is now covered by your Intel Quartus Prime Design Suite licensing. Some third-party software required by the HLS compiler might continue to require additional licensing. — Removed information about overriding compilers. — Revised prerequisites to reflect only additional prerequisites required by the HLS compiler. — Revised path information to reflect the new file system locations of the Intel HLS Compiler files. • Renamed the following tutorials: — The `explicit_streams` tutorial is now called `explicit_streams_buffer`. — The `explicit_streams_2` tutorial is now called `explicit_streams_packets_ready_valid`.
2017.06.23	—	• Minor changes and corrections.
2017.06.09	—	• Updated High Level Synthesis (HLS) Design Examples and Tutorials on page 14 with information about new examples. • Revised *Overriding the Default GCC Compiler for Intel HLS Compiler*.
2017.03.14	—	• Removed bit operations (3DES) from list of supplied design examples.
2017.03.01	—	• Added installation of required packages and libraries needed for Linux.
2017.02.03	—	• Changed success message in Quick Start sections to `PASSED`. • Added *HLS Design Examples and Tutorials* section. • Moved *Running an HLS Design Example on Linux* and *Running an HLS Design Example on Windows* to *HLS Design Examples and Tutorials*.
2016.11.30	—	• In *HLS Compiler Prerequisites*, updated software requirements to note that the HLS compiler supports all ModelSim software editions that the Intel Quartus Prime software supports. • In *HLS Compiler Quick Start*, added a note that you must run the `init_hls` script each time you start a shell or terminal to develop your design. • In HLS Compiler Quick Start, separated the Linux and Windows instructions. • In *Running an HLS Design Example*, separated the Linux and Windows instructions. For Linux, run the `make` command; for Windows, run the `build.bat` command. • Changed the `test_x86-64` command option to `test-x86-64`. • Changed the `test_fpga` command option to `test-fpga`. • Removed the instruction to run the `make test_qii` command for Linux and the `build.bat test_qii` command for Windows because it is no longer necessary. • In *HLS Licensing Error Messages*, updated the error message you will see if the HLS compiler fails to locate the ModelSim software license.
2016.09.12	—	• Initial release.