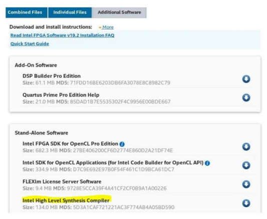


Combined Files | Individual Files | Additional Software

Download and install instructions: More

Read Intel FPGA Software v19.2 Installation FAQ

Quick Start Guide

Add-On Software

DSP Builder Pro Edition
Size: 61.1 MB **MD5:** 71FDD16BE6203DB6FA3078E8C8982C79

Quartus Prime Pro Edition Help
Size: 21.0 MB **MD5:** 85DAD1B7E5535302F4C9956E00BDE667

Stand-Alone Software

Intel FPGA SDK for OpenCL Pro Edition
Size: 682.3 MB **MD5:** 27BE406200CF6D2774E860D2A21DF74E

Intel SDK for OpenCL Applications (for Intel Code Builder for OpenCL API)
Size: 334.9 MB **MD5:** D7C9E692E97B0F54F461C1D9BCA61DC7

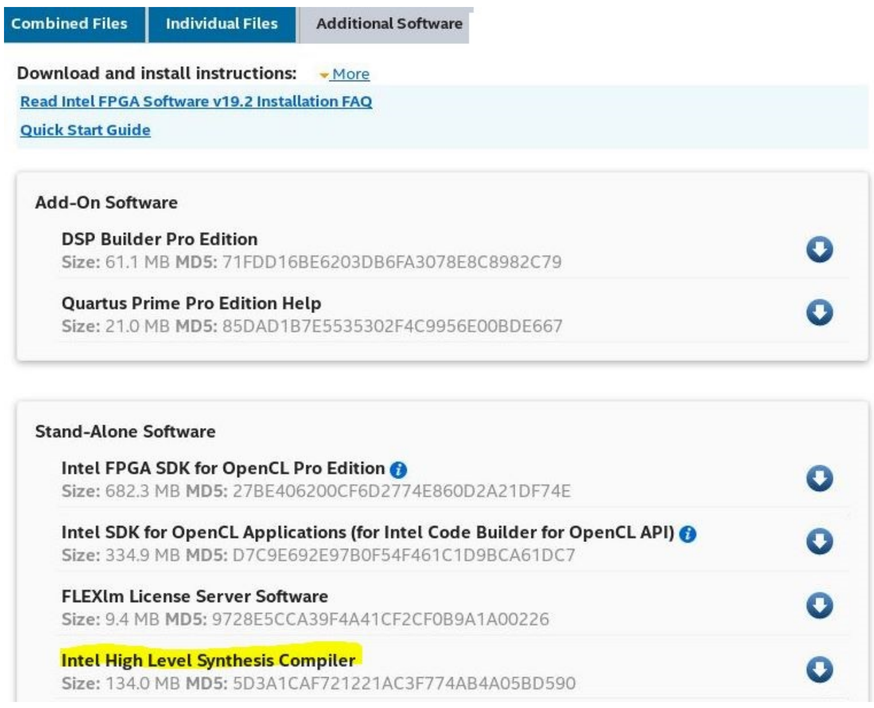
FLEXlm License Server Software
Size: 9.4 MB **MD5:** 9728E5CCA39F4A41CF2CF0B9A1A00226

Intel High Level Synthesis Compiler
Size: 134.0 MB **MD5:** 5D3A1CAF721221AC3F774AB4A05BD590

intel High Level Synthesis Compiler Pro Edition Instructions

Contents

intel High-Level Synthesis Compiler Pro Edition


Combined Files | Individual Files | Additional Software

Download and install instructions: More
Read Intel FPGA Software v19.2 Installation FAQ
Quick Start Guide

Add-On Software

DSP Builder Pro Edition
Size: 61.1 MB MD5: 71FDD16BE6203DB6FA3078E8C8982C79

Quartus Prime Pro Edition Help
Size: 21.0 MB MD5: 85DAD1B7E5535302F4C9956E00BDE667

Stand-Alone Software

Intel FPGA SDK for OpenCL Pro Edition
Size: 682.3 MB MD5: 27BE406200CF6D2774E860D2A21DF74E

Intel SDK for OpenCL Applications (for Intel Code Builder for OpenCL API)
Size: 334.9 MB MD5: D7C9E692E97B0F54F461C1D9BCA61DC7

FLEXlm License Server Software
Size: 9.4 MB MD5: 9728E5CCA39F4A41CF2CF0B9A1A00226

Intel High Level Synthesis Compiler
Size: 134.0 MB MD5: 5D3A1CAF721221AC3F774AB4A05BD590

Product Information

Product Name	Intel High-Level Synthesis Compiler Pro Edition
Version	22.4
Release Date	December 19, 2022
Deprecation Notice	The Intel High-Level Synthesis Compiler is planned to be deprecated after Version 23.4.

Product Usage Instructions

Getting Started Guide

1. Initialize your compiler environment.
2. Review the various design examples and tutorials provided with the Intel HLS Compiler.

User Guide

- The User Guide provides instructions on
- Synthesizing, verifying, and simulating intellectual property (IP) designed for Intel FPGA products.
- Going 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.

Best Practices Guide

The Best Practices Guide provides techniques and practices that you can apply to improve the FPGA area utilization and performance of your HLS component. Apply these best practices after verifying the functional correctness of your component.

Reference Manual

The 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.

Intel® High Level Synthesis Compiler Pro Edition Version 22.4 Release Notes

- The Intel® High Level Synthesis Compiler Pro Edition Release Notes provide late-breaking information about the Intel High Level Synthesis Compiler Pro Edition Version 22.4.

Pending Deprecation of the Intel HLS Compiler

- To keep access to the latest FPGA high-level design features, optimizations, and development utilities, migrate your existing designs to use the Intel oneAPI Base Toolkit.
- The Intel High Level Synthesis (HLS) Compiler is planned to be deprecated after Version 23.4.
- Visit the Intel oneAPI product page for migration advice, or go to the Intel High Level Design community forum for any questions or requests.

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	
Release Notes Provides 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 d esign for Intel FPGA products. Go through the entire development flow of your component from cre ating your component and testbench up to integrating your component IP into a larger system with the Intel Quartus Prime software.	Link
Best Practices Guide Provides techniques and practices that you can apply to improve the FPGA area utilization and per formance of your HLS component. Typically, you apply these best practices after you verify the functional correctness of your component.	Link
Reference Manual Provides reference information about the features supported by the Intel HLS Compiler. Find detail s on Intel HLS Compiler command options, header files, pragmas, attributes, macros, declarations, arguments, and template libraries.	Link

Pending Deprecation of the Intel HLS Compiler

- To keep access to the latest FPGA high-level design features, optimizations, and development utilities, migrate your existing designs to use the Intel oneAPI Base Toolkit.
- The Intel High Level Synthesis (HLS) Compiler is planned to be deprecated after Version 23.4.
- Visit the Intel oneAPI product page for migration advice, or go to the Intel High Level Design community forum for any questions or requests.

New Features and Enhancements

- The Intel High Level Synthesis Compiler Pro Edition Version 22.4 includes the following new features:
- Maintenance release.

    - No new features or enhancements for Intel HLS Compiler Pro Edition Version 22.4.

Changes in Software Behavior

- This section documents instances where Intel HLS Compiler Pro Edition Version 22.4 features have changed from earlier releases of the compiler.

Maintenance release.

- No changes in software behavior for Intel HLS Compiler Pro Edition Version 22.4.

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.

Siemens* EDA Questa® Simulation Software

- On Windows and RedHat Linux systems, you can install the Questa® simulation software from the Intel Quartus Prime software installer. The available options are as follows:
- Questa Intel FPGA Edition
- Questa Intel FPGA Starter Edition
- Both Questa Intel FPGA Edition and Questa Intel FPGA Starter Edition require licenses. The license for Questa Intel FPGA Starter Edition is free. For details, refer to Intel FPGA Software Installation and Licensing.
- Alternatively, you can use your own licensed version of Siemens* EDA ModelSim* SE or Siemens EDA Questa Advanced Simulator software.

- On Linux systems, Questa – Intel FPGA Edition and Questa – Intel FPGA Starter Edition require the Red Hat* development tools packages.
- For information about all the ModelSim and Questa 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

- Intel High Level Synthesis Compiler Getting Started Guide
- Supported Operating Systems
- Software Requirements in Intel FPGA Software Installation and Licensing
- EDA Interface Information (Intel Quartus Prime Pro Edition)

Known Issues and Workarounds

- This section provides information about known issues that affect the Intel HLS Compiler Pro Edition Version 22.4.

Description	**Workaround**
When you use the deprecated class mm_master, the compiler emits a warning message like the following: `'operator[]' has been explicitly marked deprecated here [[deprecated("Use mm_host instead.")]]` This message does not indicate which part of your code needs to change.	Avoid this warning message by using the class mm_host, which replaces the deprecated class mm_master.
(Windows only) Compiling a design in a directory with a long path name can result in compile failures. Check the debug.log file for “could not find file” errors. These errors can indicate that your path is too long.	Compile the design in a directory with a short path name.
(Windows only) A long path for your Intel Quartus Prime installation directory can prevent you from successfully compiling and running the Intel HLS Compiler tutorials and example designs. Check the debug.log file for “could not find file” errors. These errors can indicate that your path is too long.	Move the tutorials and examples to a short path name before trying to run them.

Description	**Workaround**

Libraries that target OpenCL* and are written in HLS cannot use streams or pipes as an interface between OpenCL code and the library written in HLS. However, the library in HLS can use streams or pipes if both endpoints are within the library (for example, a stream that connects two task functions).	N/A
Applying the ihc::maxburst parameter to Avalon® Memory-Mapped host interfaces can cause your design to hang in simulation.	N/A

In some uncommon cases, if you have two classes whose constructors each require instances of the other class as input, the compiler might crash.
 	
For example, compiling the following code snippet causes the compiler to crash:
 	
`struct foo;`
 	
`struct bar {`
`  int a, b, c;`
`  bar() : a(0), b(0), c(0) {};`
`  bar(const foo x);`
`};`
 	
`struct foo {`
`  int a, b, c;`
`  foo() : a(0), b(0), c(0) {};`
`  foo(const bar x) {};`
`};`
 	
`bar::bar(const foo x) {};`
	Avoid creating a circular definition. Instead, use a pointer or reference in your copy constructor.
	
	For example, transform the earlier code snippet into the following code and pass in the struct as a reference to the constructor:
	
	`struct bar {`
	`  int a, b, c;`
	`  bar() : a(0), b(0), c(0) {};`
	`  bar(const foo &x);`
	`};`
	
	`struct foo {`
	`  int a, b, c;`
	`  foo() : a(0), b(0), c(0) {};`
	`  foo(const bar &x) {};`
	`};`
	
	`bar::bar(const foo &x) {};`

Libraries that target OpenCL and are written in HLS might cause OpenCL kernels that include the library to have a more conservative incremental compilation.	N/A

When developing a library, if you have a #define defining a value that you use later in a #pragma, the fpga_crossgen command fails.
 	
For example, the following code cannot be compiled by the
 	
`#define unroll_factor 5`
 	
`int foo(int array_size) {`
`  int tmp[100];`
`  int sum =0;`
`//pragma unroll unroll_factor`
`#pragma ivdep array(tmp) safelen(unroll_factor)`
`  for (int i=0;i<array_size;i++) {`
`    sum+=tmp[i];`
`  }`
`  return sum;`
`}`
	Use pragma instead of #pragma.
	
	For example, the following compiles successfully with the
	
	`#define unroll_factor 5`
	
	`int foo(int array_size) {`
	`  int tmp[100];`
	`  int sum =0;`
	`//pragma unroll unroll_factor`
	`__pragma ivdep array(tmp) safelen(unroll_factor)`
	`  for (int i=0;i<array_size;i++) {`
	`    sum+=tmp[i];`
	`  }`
	`  return sum;`
	`}`

When you use the -c command option to have separate compilation and linking stages in your workflow, and if you do not specify the -march option in the linking stage (or specify a different -march option value), your linking stage might fail with or without error messages.	Ensure that you use the same -march option value for both the compilation with the -c command option stage and the linking stage.

Description	**Workaround**

Applying the hls_merge memory attribute to an array declared within an unrolled or partially unrolled loop causes copies of the array to be merged across the unrolled loop iterations.
 	
`#pragma unroll 2`
`for (int I = 0; I < 8; i++) {`
`   hls_merge(“WidthMerged”, “width”) int MyMem1[128];`
`   hls_merge(“WidthMerged”, “width”) int MyMem2[128];`
`   ...`
`   hls_merge(“DepthMerged”, “depth”) int MyMem3[128];`
`   hls_merge(“DepthMerged”, “depth”) int MyMem4[128];`
`   ...`
`}`
	Avoid using the hls_merge memory attribute in unrolled loops.
	
	If you need to merge memories in an unrolled loop, explicitly declare an array of struct type for width merging , or declare a deeper array for depth merging.
	
	`struct Type {int A; int B;};`
	`#pragma unroll 2`
	`for (int I = 0; I < 8; i++) {`
	`   Type WidthMerged[128];  // Manual width merging`
	`   ...`
	`   int DepthMerged[256];   // Manual depth merging`
	`   ...`
	`}`

In the Function Memory Viewer high-level design report, some function-scoped memories might appear as “optimized away”.	None. When a file contains functions that are components and functions that are not components, all function-scoped variables are listed in the Function Memory List pane, but only variables from components have information about them to show in the Function Memory View pane.
Some high-level design reports fail in Microsoft Internet Explorer*.	Use one of the following browsers to view the reports: • Google Chrome* • Microsoft Edge* • Mozilla* Firefox*
The Loop Viewer in the High-Level Design Reports has the following restrictions: • The behavior of stall-free clusters is not modeled in the Loop Viewer. The final latency shown in the Loop Viewer for a stall-free cluster is typically more pessimistic (that is, higher) than the actual latency of your design. For a description of clustering and stall-free clusters, refer to *Clustering the Datapath* in the *Intel High-Level Synthesis Compiler Pro Edition Best Practices Guide*. • Stalls from reads and writes from memory or print statements are not modeled. • High-iteration counts (>1000) cause slow performance of the Loop Viewer. • You cannot specify an iteration count of zero (0) in the Loop Viewer.	None.
Links in some reports in the High-Level Design Reports generated on Windows systems do not work.	Generate the High-Level Design Reports (that is, compile your code) on a Linux system.

Using a struct of a single ac_int data type in a steaming interface that uses packets (ihc::usesPackets<true>) does not work.
 	
For example, the following code snippet does not work:
 	
`// class definition`
`class DataType {`
`      ac_int<155, false> data;`
`...`
`}`
`// stream definition`
`typedef ihc::stream_in<DataType,`
`                        ihc::usesPackets<true>,`
`                        ihc::usesEmpty<true>`
`                       > DataStreamIn;`
	To use this combination in your design, obey the following restrictions:
	
	• The internal ac_int data size must be multiple of 8
	
	• The stream interface type declaration must specify
	
	ihc::bitsPerSymbol<8>
	
	For example, the following code snippet works:
	
	`// class definition`
	`class DataType {`
	`      ac_int<160, false> data;`
	`// data width must be multiple of 8`
	`...`
	`}`
	`// stream definition`
	`typedef ihc::stream_in<DataType,`
	`                        ihc::usesPackets<true>,`
	`                        ihc::usesEmpty<true>,`
	`                        ihc::bitsPerSymbol<8>`
	`                       > DataStreamIn;`
	`// added ihc::bitsPerSymbol<8>`

Description	**Workaround**
When running a high-throughput simulation of your component using enqueue function calls, if you do not use the ihc_hls_component_run_all function to run the enqueued component calls after all of the ihc_hls_enqueue calls for that component, the following behaviors occur: • In emulation, the enqueued component functions are run. • In simulation, the enqueued component functions are not run, with no error or warning messages provided.	Ensure that you use the ihc_hls_component_run_all function after all of the ihc_hls_enqueue calls for that component to run enqueued component function calls.
Launching a task function with ihc::launch_always_run	To avoid stripping away the optimization, add a while(1)
strips away optimization attributes applied to the task	loop to the affected function apply the corresponding control
function.	pragma to the while(1) loop instead of the function.

In the following code example, the attribute applied to the function is ignored. The High-Level Design Reports show an II of 1 for this task instead of the requested II of 4.

The following code example show how you can implement this change for the earlier code example:

```
hls_component_ii(4) void noop()
{
    bool sop, eop;
    int empty;
    auto const data = data_in.read(sop, eop, empty);

    data_out.write(data, sop, eop, empty);
}

component void main_component()
{
    ihc::launch<noop>();
}
```

```
void noop()
{
#pragma ii 4
    while (1)
    {
        bool sop, eop;
        int empty;
        auto const data = data_in.read(sop, eop, empty);

        data_out.write(data, sop, eop, empty);
    }
}

component void
main_component()
{
    ihc::launch_always_run<noop>();
}
```

For Cyclone® V projects that contain multiple HLS components, when you use the i++ command to compile your project to hardware (i++ -march=CycloneV), you might receive an error.

While the error text differs depending on your project, the error signature is an Intel Quartus Prime compilation failure due to bad Verilog syntax. A module tries to use a function that the Intel Quartus Prime compiler cannot find.

If you encounter this issue, put each HLS component in a separate project.

Compiling some designs that contain multiple components generates an error about stream reuse.

If you encounter this issue, compile each component in the design separately. You might need to add macros to your code to enable each component to compiled separately.

Consider the following example:

```
ihc::stream<...> s1;
ihc::stream<...> s2;

#if USE_COMP1
component
#endif
void comp1(...)
{
    // code s1.write();
}

#if USE_COMP2
component
#endif
void comp2(...)
{
   // code s2.read();
}
```


Intel High-Level Synthesis Compiler Pro Edition Release Notes Archives

For the latest and previous versions of this user guide, refer to Intel HLS Compiler Pro Edition Release Notes. If a software version is not listed, the release notes for the previous software version apply.

Document Revision History for Intel HLS Compiler Pro Edition Version 22.4 Release Notes

Document Version	Intel Quartus Prime Version	Changes
2022.12.19	22.4	• Initial release.

Documents / Resources

intel. Intel® High Level Synthesis Compiler Pro Edition. Version 22.4 Release Notes. Updated for Intel® Quartus® Prime Design Suite: 22.4. Online Version. Send Feedback. RN-1146. ID: 683682. Version: 2022.12.19	[intel High Level Synthesis Compiler Pro Edition]() [pdf] Instructions Version 22.4, Version 23.4, High Level Synthesis Compiler Pro Edition, High Level Synthesis C ompiler, Pro Edition

References

- Intel® High Level Design - Intel Communities
- intel oneAPI: A New Era of Heterogeneous Computing
- intel 1. Intel® HLS Compiler Pro Edition Best Practices Guide
- intel 3.3.2.2. Clustering the Datapath
- intel 1. Intel® HLS Compiler Pro Edition Best Practices Guide
- intel 1. Intel® HLS Compiler Pro Edition Reference Manual
- intel 1. Intel® High Level Synthesis Compiler Pro Edition User Guide
- intel 1. Answers to Top FAQs
- intel 2. Introduction to Intel® FPGA Software Installation and Licensing
- intel 3.3. Software Requirements
- intel 1. Intel® High Level Synthesis (HLS) Compiler Pro Edition Getting...
- intel 1. Intel® High Level Synthesis (HLS) Compiler Pro Edition Getting...
- intel 1. Intel® High Level Synthesis Compiler Pro Edition Version 23.1...
- intel 1. Intel® Quartus® Prime Pro Edition Version 23.1 Software and Device...
- intel Software Operating System (OS) Support List Information | Intel
- intel Intel ISO 9001:2015 Registrations