Intel® High Level Synthesis Compiler Pro Edition

Version 20.2 Release Notes

Updated for Intel® Quartus® Prime Design Suite: 20.2
Contents

   1.1. New Features and Enhancements ............................................................................. 4
   1.2. Changes in Software Behavior ............................................................................... 5
   1.3. Intel High Level Synthesis Compiler Pro Edition Prerequisites ............................... 5
   1.4. Known Issues and Workarounds ........................................................................... 6
   1.5. Software Issues Resolved ..................................................................................... 8
       Notes ................................................................................................................ 9
1. Intel® High Level Synthesis Compiler Pro Edition
Version 20.2 Release Notes


For the most recent Standard Edition release notes, see the Intel High Level Synthesis Compiler Standard Edition Release Notes.

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

<table>
<thead>
<tr>
<th>Title and Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release Notes</td>
<td>Link</td>
</tr>
<tr>
<td>Provide late-breaking information about the Intel HLS Compiler.</td>
<td></td>
</tr>
<tr>
<td>Getting Started Guide</td>
<td>Link</td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>User Guide</td>
<td>Link</td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Reference Manual</td>
<td>Link</td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Best Practices Guide</td>
<td>Link</td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Quick Reference</td>
<td>Link</td>
</tr>
<tr>
<td>Provides a brief summary of Intel HLS Compiler declarations and attributes on a single two-sided page.</td>
<td></td>
</tr>
</tbody>
</table>
1.1. New Features and Enhancements

The Intel High Level Synthesis Compiler Pro Edition Version 20.2 includes the following new features:

- Added support for Intel Agilex™ devices.
- The `hls_float.h` header file now includes `bfloat16` and `bfloat19` aliases for `hls_float<8,7>` and `hls_float<8,10>` respectively.

On Intel Agilex devices, dot products that use these data types (both the `bfloat` and `hls_float` variants) take advantage of the Intel Agilex FP16/19 DSPs to provide FPGA area savings.
1.2. Changes in Software Behavior

The section documents instances where Intel HLS Compiler Pro Edition features have changed from earlier releases of the compiler.

- On Windows operating systems, pipes are now supported.
- When compiling RTL source files with names that match terms for established RTL concepts, the compiler now issues a warning message.

Previously, you received only an error message in debug.log.

1.3. 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 9.1.0 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 9.1.0 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 the Microsoft Visual Studio Professional:
- 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.

**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 RedHat 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.

On SUSE Linux systems, you must use your own licensed version of Mentor Graphics ModelSim software.

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**

- 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)
- Mentor Graphics ModelSim Website

### 1.4. Known Issues and Workarounds

This section provides information about known issues that affect the Intel HLS Compiler Pro Edition Version 20.2.

<table>
<thead>
<tr>
<th>Description</th>
<th>Workaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Windows only) Compiling a design in a directory with a long path name can</td>
<td>Compile the design in a directory with a short</td>
</tr>
<tr>
<td>result in compile failures.</td>
<td>path name.</td>
</tr>
<tr>
<td>(Windows only) A long path for your Intel Quartus Prime installation</td>
<td>Move the tutorials and examples to a short path</td>
</tr>
<tr>
<td>directory can prevent you from successfully compiling and running the Intel</td>
<td>name before trying to run them.</td>
</tr>
<tr>
<td>HLS Compiler tutorials and example designs.</td>
<td></td>
</tr>
<tr>
<td>(Windows only) Library functions that use HLS systems of tasks the</td>
<td>N/A</td>
</tr>
<tr>
<td>cannot be emulated.</td>
<td></td>
</tr>
<tr>
<td>Libraries that target OpenCL* and are written in HLS cannot use streams or</td>
<td>N/A</td>
</tr>
<tr>
<td>pipes as an interface between OpenCL code and the library written in HLS.</td>
<td></td>
</tr>
<tr>
<td>However, the library in HLS can use streams or pipes if both endpoints are</td>
<td></td>
</tr>
<tr>
<td>within the library (for example, a stream that connects two task functions).</td>
<td></td>
</tr>
<tr>
<td>Libraries that target OpenCL and are written in HLS might cause OpenCL</td>
<td>N/A</td>
</tr>
<tr>
<td>kernels that include the library to have a more conservative incremental</td>
<td></td>
</tr>
<tr>
<td>compilation.</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Workaround</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>When developing a library, if you have a <code>#define</code> defining a value that you use later in a <code>#pragma</code>, the <code>fpga_crossgen</code> command fails. For example, the following code cannot be compiled by the <code>fpga_crossgen</code> command:</td>
<td>Use <code>__pragma</code> instead of <code>#pragma</code>. For example, the following compiles successfully with the <code>fpga_crossgen</code> command: <code>#define unroll_factor 5</code></td>
</tr>
</tbody>
</table>
| `#define unroll_factor 5`  
int foo(int array_size) {  
  int tmp[100];  
  int sum =0;  
  __pragma ivdep array(tmp) safelen(unroll_factor)  
  for (int i=0;i<array_size;i++) {  
    sum+=tmp[i];  
  }  
  return sum;  
} | `int foo(int array_size) {  
  int tmp[100];  
  int sum =0;  
  __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. |
| Applying the `hls_merge` memory attribute to an array declared within an unrolled or partially unrolled loop creates an unexpectedly wide memory. | 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. |
| When you apply the `hls_avalon_slave_memory_argument` attribute on an array function argument, the compiler crashes. For example, the following code causes the compiler to crash: | Use a pointer argument instead. The two styles are equivalent. For example: `component void foo (hls_avalon_slave_memory_argument
(128*sizeof(int)) int *A)` |
| `component void foo (hls_avalon_slave_memory_argument
(128*sizeof(int)) int A[128])` | 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. |
| In the Function Memory Viewer high-level design report, some function-scoped memories might appear as "optimized away". | 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* |
| Some high-level design reports fail in Microsoft* Internet Explorer*. | continued... |
### Description

Using a `struct` of a single `ac_int` data type in streaming interface that uses packets (ihc::usesPackets<true>) does not work. For example, the following code snippet does not work:

```cpp
// class definition
class DataType {
    ac_int<155, false> data;
    ...
} // stream definition
typedef ihc::stream_in<DataType,
    ihc::usesPackets<true>,
    ihc::usesEmpty<true>
> DataStreamIn;
```

### Workaround

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:

```cpp
// class definition
class DataType {
    ac_int<160, false> data;
    ...
} // stream definition
typedef ihc::stream_in<DataType,
    ihc::usesPackets<true>,
    ihc::usesEmpty<true>,
    ihc::bitsPerSymbol<8>
> DataStreamIn;
```

### 1.6. Intel High Level Synthesis Compiler Pro Edition Release Notes

#### Archives

<table>
<thead>
<tr>
<th>Intel HLS Compiler Version</th>
<th>Title</th>
</tr>
</thead>
</table>

---

Continue...

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020.06.22</td>
<td>20.2</td>
<td>• Initial release.</td>
</tr>
</tbody>
</table>