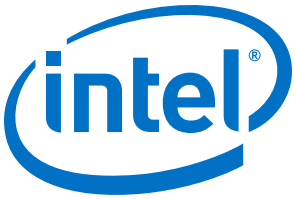

Intel® High Level Synthesis Compiler

Version 18.1 Release Notes

Updated for Intel® Quartus® Prime Design Suite: **18.1**

Subscribe
Send Feedback

RN-1146 | 2019.02.04
Latest document on the web: PDF | HTML

Contents

1. Intel® High Level Synthesis Compiler Version 18.1 Release Notes

The *Intel® High Level Synthesis Compiler Release Notes* provide late-breaking information about the Intel High Level Synthesis Compiler included with Intel Quartus® Prime Design Suite Version 18.1.

1.1. New Features and Enhancements

The Intel High Level Synthesis Compiler included with Intel Quartus Prime Design Software Version 18.1 includes the following new features:

- PRO Improved the Intel HLS Compiler compiler front end.

  For details, see Improved Intel HLS Compiler Front End on page 3.
- STD The Intel HLS Compiler included with Intel Quartus Prime Design Software Standard Edition Version 18.1 is a maintenance release. No new features or enhancements are added.

PRO Improved Intel HLS Compiler Front End

The Intel HLS Compiler in Intel Quartus Prime Design Software Pro Edition Version 18.1 has an improved front end that provides the following benefits:

- Improved error messages and warnings to help you debug and improve the code in your components.
- Adherence to the C++14 specification.
- Support for newer compilers including GCC compiler and C++ Libraries version 5.4.0 and Microsoft Visual Studio 2015 Professional.

You might need to adjust your code to comply with the C++14 standards and other changes introduced by the new compiler front end. The updates to the Intel HLS Compiler affect your component and testbench code in the following ways:

- Your component code must now include the `#include "HLS/hls.h"` statement.

  Previously, you could use the `component` keyword without including the `hls.h` header file in your code.
- The `--component` option of the `i++` command no longer supports component-level attributes (such as `hls_avalon_slave_component`) and parameter arguments (such as `hls_avalon_slave_memory_argument`).

  If your component uses component-level attributes, parameter arguments, or both, ensure that your component uses the `component` keyword.
- The heuristics to determine when to automatically unroll loops have changed.

  If the new loop unrolling decisions affect your component negatively, use the `#prama unroll <N>` statement to specify a loop unroll factor.
- Loop pragmas (for example, `#pragma unroll`) must immediately precede the loop that the pragma applies to.

  Previously, loop pragmas could be placed before elements such as labels on loops.
- The Intel HLS Compiler now adheres strictly to C++ integer promotion rules.

  For datatypes smaller than `int` (such as `short`, `char`, and `bool`), this integer promotion might increase the FPGA area that these datatypes in your component use. You might need to convert these datatypes in your component to use the Intel HLS Compiler `ac_int` datatype to generate more efficient hardware for these datatypes in your component.
- Any `ac_int` variables larger than 64 bits must be initialized with the `ac::init_array` constructor
- Pragmas must appear in the function bodies in your code.

  Previously, you could place a pragma statement between a function declaration and the function body.
- The Intel HLS Compiler enforces the C restriction that an `enum` cannnot be incremented with the ++ operator.
- The `restrict` type qualifier keyword is replaced with the `__restrict` keyword.
- If you apply Intel HLS Compiler register and memory attributes to member variables of a struct or class, you now receive error messages.

  Memory attributes applied to register and memory attributes are not supported and were previously ignored.
- Memory attributes applied to static variables declared outside of function calls are now accepted.
- Variable length arrays are not supported by the C++14 standard and cannot be used in your testbench or component code.
- The bank selection bits specified in the `hls_bankbits` memory attribute must be specified in ascending or descending order. If the bank selection bits are not ordered, you receive an error message.
- On Windows systems, object files and libraries that depend on Microsoft Visual Studio runtime libraries must be built with the `/MD` Microsoft compiler option.

1.2. Intel High Level Synthesis Compiler Prerequisites

The Intel HLS Compiler is installed as part of the Intel Quartus Prime software installation, but it requires additional software to use.

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

The Intel HLS Compiler requires the following additional software:

C++ Compiler

For Linux, install one of the following versions of the GCC compiler and C++ libraries, depending on your edition of Intel Quartus Prime software:

- PRO GCC compiler and C++ Libraries version 5.4.0

  You must install these libraries manually. See Installing the Intel HLS Compiler on Linux Systems for instructions.
- STD GCC compiler and C++ Libraries version 4.4.7

  These libraries are included in the version of Linux supported by the Intel HLS Compiler.

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

For Windows, install one of the following versions of the Microsoft Visual Studio Professional, depending on your edition of Intel Quartus Prime software:

- PRO Microsoft Visual Studio 2015 Professional
- PRO Microsoft Visual Studio 2015 Community
- STD Microsoft Visual Studio 2010 Professional

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

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 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

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 Standard Edition)
- EDA Interface Information (Intel Quartus Prime Pro Edition)
- Mentor Graphics Website

1.3. Known Issues and Workarounds

This section provides information about known issues that affect the Intel High Level Synthesis Compiler Version 18.1.

Description	Workaround
The Intel HLS Compiler is not included in the Windows* release of Intel Quartus Prime Design Suite Version 18.1. The Intel Quartus Prime Design Suite Version 18.1 HLS Compiler is available on all supported Linux distributions.	For details about the availability of the Intel HLS Compiler on Windows*, see Where is the Intel HLS Compiler version 18.1 for Windows? in the Intel FPGA Knowledge Base.
(Windows only) Compiling a design in a directory with a long path name can result in compile failures.	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.	Move the tutorials and examples to a short path name before trying to run them.

The Intel HLS Compiler does not merge identical read accesses to an array. This can result in a memory system with additional load ports, which consumes extra RAM resources on the FPGA.
For example, in the following code snippet, the Intel HLS Compiler infers two distinct read accesses to `array_a` rather than just one.
`component int duplicate_loads (int index )`
`  int array_a[256];`
`  int array_b[256];`
`  /* other code */`
`  if (index % 16 == 0){`
`  array_b[index]=array_a[index]; // read from array_a[index]`
`  }`
`  if (index % 16 == 1){`
`  array_b[(index+5) % 256]=array_a[index]; // identical read`
`  }`
`  /* other code */`
`}`
	Manually refactor your code to explicitly merge identical array accesses.
	For example, the code snippet could be refactored into the following code:
	`component int duplicate_loads (int index )`
	`  int array_a[256];`
	`  int array_b[256];`
	`  /* other code */`
	`  if ( index % 16 ==0 || index % 16 ==1){`
	`  int tmp=array_a[index]; // single read from array_a[index]`
	`  if (index % 16 == 0){`
	`  array_b[index]=tmp;`
	`  } else {`
	`  array_b[(index+5) % 256]=tmp;`
	`  }`
	`  }`
	`  /* other code */`
	`}`
	The refactored code has only a single access to `array_a[index]`, so it does not rely on the compiler to merge identical read accesses and achieve an efficient implementation.

continued...

Description	Workaround
If your component name or component parameters have the same names as Intel HLS Compiler keywords, you might receive the following error message when you run simulation: `Cannot find component name in map` You also receive this error during simulation if your component name has one or more leading underscore characters, one ore more trailing underscore, or more than one underscore in the middle of the component name.	To prevent this simulation error message, rename your component or component name to avoid Intel HLS Compiler keywords or invalid underscore use in names.
Component memory might not correctly respect the `hls_max_concurrency` memory attribute, which causes functionally incorrect behavior.	Avoid setting the `hls_max_concurrency` memory attribute to a value other than `1` in components with memory.
Sometimes an invalid warning message about unused function parameters might be generated. The warning message appears as follows: `Compiler Warning: Argument '<argument_name>' on component '<component_name>'is never used by the component. Note that the compiler may optimize it away.`	No workaround needed. This warning has no impact on the RTL generation or the simulation.
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. Ignore any compiler warnings during the linking stage that say that the `-march` option has no effect during the linking stage. Include the `-march` option in your linking despite any compiler warning messages such as the following warning: `Warning:-march has no effect. Using settings from -c compile`
For HLS components with a streaming interface where the readylatency attribute is set to zero, your system might have functional issues if the valid signal and data for an external component change when the HLS component is holding a stall signal. This behavior deviates from the Avalon Interface Specifications. For details about the specification, see Data Transfers Using readyLatency and readyAllowance (section 5.9.1) of the *Avalon Interface Specification*. This issue also applies for component invocation, where start is equivalent to valid, and busy is equivalent to not ready.	For any HLS component with a streaming interface where the readylatency attributes is set to zero, ensure that the valid signal and its associated data for an external component is kept unchanged when the HLS component is holding a stall signal.

1.4. Software Issues Resolved

The following issues were corrected or otherwise resolved in the Intel HLS Compiler Version 18.1.

Table 1. Issues Resolved in the Intel HLS Compiler Version

Customer Service Request Numbers						
11358219	11370677	11373525	11379002	11410991	11412171	11412627

1.5. Intel High Level Synthesis Compiler Release Notes Archives

Intel HLS Compiler Version	User Guide
18.0	Intel High Level Synthesis Compiler Version 18.0 Release Notes
17.1	Intel High Level Synthesis Compiler Version 17.1 Release Notes

1.6. Document Revision History for Intel HLS Compiler Version 18.1 Release Notes

Document Version	Intel Quartus Prime Version	Changes
2019.02.04	18.1	• Updated Known Issues and Workarounds on page 6 to restore an issue about how long paths in a Windows environment can prevent you from successfully compiling your component.
2018.12.24	18.1	• Updated Known Issues and Workarounds on page 6 with a missing issue affecting HLS components with a streaming interface where the readylatency attribute is set to zero. • Added Intel High Level Synthesis Compiler Release Notes Archives on page 8 topic to provide links to past versions of this document.
2018.09.24	18.1	• Initial release.