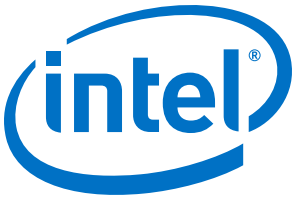

インテル®高位合成 (HLS) コンパイラー プロ・エディション

スタートガイド

インテル® Quartus® Prime 開発デザインスイートの更新情報: **20.2**

この翻訳版は参照用であり、翻訳版と英語版の内容に相違がある場合は、英語版が優先されるものとします。翻訳版は、資料によっては英語版の更新に対応していない場合があります。最新情報につきましては、必ず英語版の最新資料をご確認ください。

更新情報
フィードバック

UG-20036 | 2020.06.22
日本語版の最新資料: **PDF** | **HTML**

目次

UG-20036 | 2020.06.22

フィードバック

1. インテル®HLS (高位合成) コンパイラー・プロ・エディション スタートガイド

インテル®HLS (高位合成) コンパイラーは、個別インストール可能な インテル Quartus® Prime プロ・エディション・デザイン・ソフトウェアのコンポーネントです。インテル HLS コンパイラーでは、C++ 関数を インテル FPGA 製品向けに最適化された RTL 実装に合成します。このコンパイラーは、コンパイラー・コマンドの名前を反映して、i++ コンパイラーと呼ばれることもあります。

*インテル HLS コンパイラー・プロ・エディション スタートガイド*では、インテル HLS コンパイラーのセットアップおよび HLS デザイン例の実行の手順について説明します。

このドキュメントでは、 <*quartus_installdir*> が指すのは、インテル Quartus Prime デザインスイートをインストールした場所です 。

インテル Quartus Prime デザインスイートのデフォルトのインストール先は、使用しているオペレーティング・システムにより次のようになります。

Windows `C:\intelFPGA_pro\20.2`

Linux `/home/<username>/intelFPGA_pro/20.2`

インテル HLS コンパイラー・プロ・エディションのドキュメント・ライブラリーについて

インテル HLS コンパイラー・プロ・エディションのドキュメントは、いくつかの文書に分かれています。次の表を使用して、目的の インテル HLS コンパイラー・プロ・エディションの情報を含む資料を見つけてください。

表 1. インテル HLS コンパイラー・プロ・エディションのドキュメント・ライブラリー

タイトルと説明	PRO
Release Notes インテル HLS コンパイラーに関する最新情報を提供しています。	リンク
スタートガイド インテル HLS コンパイラーを動作させるために、コンパイラー環境を初期化する方法を身に付け、また、インテル HLS コンパイラーで用意されているさまざまなデザイン例とチュートリアルを確認します。	リンク
ユーザーガイド インテル FPGA 製品向けにデザインする知的財産 (IP) の合成、検証、およびシミュレーションに関する手順を説明しています。コンポーネントの開発フロー全体を確認できます。フローは、コンポーネントやテストベンチの作成から、コンポーネント IP をインテル Quartus Prime 開発ソフトウェアを使用してより大規模なシステムへ統合するまでをカバーしています。	リンク

continued...

タイトルと説明	PRO
リファレンス・マニュアル インテル HLS コンパイラーでサポートしている機能に関する参照情報を提供しています。インテル HLS コンパイラーのコマンド オプション、ヘッダーファイル、プラグマ、属性、マクロ、宣言、引数、およびテンプレート・ライブラリーの詳細を確認できます。	リンク
ベスト・プラクティス・ガイド ここで紹介している手法と実践方法を使用することによって、HLS コンポーネントの FPGA 領域の使用とパフォーマンスを向上させます。通常、このベスト・プラクティスは、コンポーネントの機能の正確性の確認後に適用します。	リンク
Quick Reference インテル HLS コンパイラーの宣言と属性の概要を 2 ページ (両面印刷で 1 枚) で提供しています。	リンク

1.1. インテル HLS コンパイラー・プロ・エディションの前提条件

インテル HLS コンパイラー・プロ・エディションは、インテル Quartus Prime プロ・エディション デザインスイートの一部です。インテル HLS コンパイラーは、インテル Quartus Prime 開発ソフトウェアの一部としてインストールするか、個別にインストールします。インテル HLS コンパイラーを使用するには、インテル Quartus Prime および追加のソフトウェアが必要です 。

システム要件、前提条件、ライセンス要件などの インテル Quartus Prime プロ・エディション開発ソフトウェアのインストールの詳しい手順については、Intel® FPGA Software Installation and Licensing を参照してください。

インテル HLS コンパイラーには、インテル Quartus Prime のほかにも次のソフトウェアの追加が必要です。

C/C++ コンパイラー

Linux では、インテル HLS コンパイラーには GCC 9.1.0 が必要です。これには、GNU C++ ライブラリーおよびバイナリー・ユーティリティー (binutils) が含まれます。

このバージョンの GCC は、インテル HLS コンパイラーのインストールの一部になっています。インテル HLS コンパイラーをインストールすると、GCC 9.1.0 が *<quartus_installdir>*/gcc で使用可能になります。

重要: インテル HLS コンパイラーでは、 *<quartus_installdir>*/gcc ディレクトリーをツールチェーン・ディレクトリーとして使用します。この GCC のインストールは、HLS 関連のすべてのデザイン作業に使用します。

Windows の場合、Microsoft Visual Studio Professional の次のバージョンのいずれかをインストールします。

- Microsoft Visual Studio 2017 Professional
- Microsoft Visual Studio 2017 コミュニティ

重要: インテル HLS コンパイラー・ソフトウェアでサポートしている Microsoft Visual Studio のバージョンは、HLS コンパイラーのエディションに対して指定されたバージョンのみです。

Mentor Graphics* ModelSim* ソフトウェア

Windows および RedHat Linux システムでは、ModelSim* ソフトウェアのインストールには、インテル Quartus Prime 開発ソフトウェアのインストーラーを使用します。使用可能なオプションは次のとおりです。

- ModelSim - インテル FPGA エディション
- ModelSim - インテル FPGA スタンダード・エディション

また、お持ちのライセンスバージョンの Mentor Graphics* ModelSim または Mentor Graphics Questa* Advanced Simulator ソフトウェアを使用することもできます。

RedHat Linux システムでは、ModelSim ソフトウェアには、Red Hat 開発ツールパッケージが必要です。さらに、32 ビットバージョンの ModelSim ソフトウェアには、インテル Quartus Prime 付属のソフトウェアを含み、32 ビット・ライブラリーが追加で必要です。このような要件をインストールするコマンドは、Linux システムでの インテル HLS コンパイラーのインストール で説明しています。

SUSE Linux システムでは、お持ちのライセンスのバージョンの Mentor Graphics ModelSim 開発ソフトウェアを使用してください。

インテル のソフトウェアでサポートしているすべての ModelSim ソフトウェア・バージョンの情報は、お使いのエディションの インテル Quartus Prime プロ・エディションの Software and Device Support Release Notes の *EDA Interface Information* の項を参照してください。

関連情報

- オペレーティング・システム・サポート
- Software Requirements
  Intel FPGA Software Installation and Licensing 内
- EDA Interface Information (インテル Quartus Prime プロ・エディション)
- Mentor Graphics ModelSim のウェブサイト

1.1.1. インテル HLS コンパイラー・プロ・エディションの下位互換性

インテル HLS コンパイラー・プロ・エディションのバージョン 20.2 は、インテル Quartus Prime プロ・エディションのバージョン 20.2、バージョン 20.1、バージョン 19.4、およびバージョン 17.1.1 と互換性があります。この下位互換性により、インテル HLS コンパイラーによって生成された RTL の改善の活用ができます。このとき、現在の FPGA 開発環境の他の部分を変更する必要はありません。

インテル HLS コンパイラー・プロ・エディションのバージョン 20.2 を旧バージョンの インテル Quartus Prime プロ・エディションにインストールするには、次の手順を実行します。:

1. 既存の <*quartus_installdir*> [(1)]/hls ディレクトリーをバックアップします。例えば、<*quartus_installdir*>/hls の名前を <*quartus_installdir*>/hls_old に変更します。
2. インテル HLS コンパイラー・プロ・エディションのスタンドアロンのインストール・パッケージをダウンロードします。インテル Quartus Prime Download Center の**個別ファイル**のタブからお使いの インテル Quartus Prime のエディションを選びます。
3. インテル HLS コンパイラーのスタンドアロンのインストーラーを実行します。プロンプトが出たら <*quartus_installdir*> ディレクトリーを指定します。

インテル HLS コンパイラーの以前のバージョンに戻すには、バックアップ・ディレクトリーの名前を変更します。どのバージョンの インテル HLS コンパイラーでも、<*quartus_installdir*>/hls ディレクトリーにあれば、アクティブなバージョンの インテル HLS コンパイラーです。

(1) <*quartus_installdir*> は、インテル Quartus Prime プロ・エディション デザインスイートをインストールした場所です。例 : C:\intelFPGA_pro\20.2

1.2. インテル HLS コンパイラー・プロ・エディションのダウンロード

インテル HLS コンパイラー・プロ・エディションのダウンロードは、インテル Quartus Prime ダウンロード・センターからできます。

インテル HLS コンパイラー・プロ・エディションのインストールは、インテル Quartus Prime プロ・エディションのインストール・パッケージの一部として行うか、または別のスタンドアロンのインストーラーから行います。

インテル HLS コンパイラー・プロ・エディションのスタンドアロンのインストーラーをダウンロードするのは、インテル HLS コンパイラーを既存の インテル Quartus Prime プロ・エディションのインストールに追加する場合です。インテル HLS コンパイラー・プロ・エディションのバージョン 20.2 と互換性があるのは、インテル Quartus Prime プロ・エディションのバージョン 20.2、バージョン 20.1、バージョン 19.4、およびバージョン 17.1.1 です 。

- インテル HLS コンパイラーのダウンロードは、インテル Quartus Prime ダウンロード・センターからできます。
  - http://fpgasoftware.intel.com/?edition=pro

  インテル HLS コンパイラー・プロ・エディションのスタンドアロンのインストール・パッケージは、**追加ソフトウェア**のタブからできます。

  インテル HLS コンパイラー・プロ・エディションを インテル Quartus Prime プロ・エディションのインストールに含めるには 、インストール・パッケージのうち次の組み合わせのいずれかをダウンロードします。
  - **一式ファイル**タブから完全なインストール・パッケージをダウンロードします。
  - Quartus Prime インストール・パッケージを **個別ファイル**タブからダウンロードします。インテル HLS コンパイラーのインストール・パッケージを**追加ソフトウェア**タブからダウンロードします。両方のインストール・パッケージを同じディレクトリーに配置します。

  システム要件、前提条件、およびライセンス要件を含む インテル Quartus Prime 開発ソフトウェアのインストールの詳しい手順については、Intel FPGA Software Installation and Licensing を参照してください。

  インテル HLS コンパイラーのスタンドアロンのインストール・パッケージには、インテル HLS コンパイラーのインストールのターゲットとして、既存の インテル Quartus Prime プロ・エディションのインストールが必要です。

1.3. Linux システムでの インテル HLS コンパイラー・プロ・エディションのインストール

インテル HLS コンパイラー・プロ・エディションの前提条件をインストールするには、管理者権限が必要です 。ただし、インテル HLS コンパイラーおよび インテル Quartus Prime をインストールするのに管理者権限は必要ありません。

Linux システムに インテル HLS コンパイラーをインストールするには、次の手順を実行します。

1. お使いのオペレーティング・システムのバージョンが、インテル HLS コンパイラーによってサポートされている次のいずれかであることを確認します。

- Red Hat Enterprise Linux 6.x、または相当コミュニティー版
- Red Hat Enterprise Linux 7.x、または相当コミュニティー版
- SUSE Linux Enterprise Server 12

2. 現在のシステムやダウンロード済みのインストール・パッケージによっては、インストールの準備のために次の追加手順が必要になる場合があります。
   - 完全な インテル Quartus Prime インストール・パッケージを (FPGAs ダウンロード・センターの Quartus Prime ダウンロードのページの**一式ファイル**タブから) インストールする場合 -

     インストール・パッケージ実行前の追加手順は不要です。
   - (FPGAs ダウンロード・センターの Quartus Prime ダウンロード・ページの**個別ファイル**タブから) 個別にダウンロードした インテル Quartus Prime と (FPGAs ダウンロードセンターの Quartus Prime ダウンロードのページの**追加ソフトウェア**タブから) 個別にダウンロードした インテル HLS コンパイラーのインストール・パッケージをインストールする場合 -

     両方のインストール・パッケージが同じディレクトリーにあることを確認してください。インテル Quartus Prime インストーラーでは、インテル HLS コンパイラーのインストール・パッケージを検出し、両方のソフトウェア・パッケージをインストールします。
   - インテル HLS コンパイラーの更新を既存の インテル Quartus Prime プロ・エディションのインストールで行う場合は、次の事項を実行します。
     a. インテル Quartus Prime プロ・エディションがインストール済みであることを確認してください。
     b. インテル Quartus Prime へのパスに注意してください。

        このパス情報は、インテル HLS コンパイラーのインストール・ウィザードを完了するために必要です。
     c. 現在の インテル Quartus Prime バージョンの HLS ディレクトリーの名前を変更して、そのバージョンをバックアップとして保持します。

        例えば、インテル HLS コンパイラーのバージョン 20.2 を インテル Quartus Prime プロ・エディションのバージョン 20.1 インストールにインストールする場合は、`/home/<username>/intelFPGA_pro/20.1/hls` の名前を `/home/<username>/intelFPGA_pro/20.1/hls_old` に変更します。
3. インテル HLS コンパイラー・プロ・エディションのダウンロード (6 ページ) でダウンロードしたパッケージをインストールします。

   システム要件、前提条件、ライセンス要件などの インテル Quartus Prime 開発ソフトウェアのインストールの詳しい手順については、Intel FPGA Software Installation and Licensing を参照してください。
4. 次のコマンドを使用して Linux リポジトリーを更新します。

   ```
   sudo yum update
   ```
5. (RedHat Linux のみ) インテル Quartus Prime (ModelSim - インテル FPGA エディション) に付属の 32 ビット Mentor Graphics ModelSim ソフトウェアを使用して、インテル HLS コンパイラーでコンポーネントのシミュレーションを行う場合は、次のコマンドを使用して、必要な 32 ビット・ライブラリーを追加でインストールします。

- Red Hat Enterprise Linux 6.x、または相当コミュニティー版

```
$ 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 qt-x11.i686
```

- Red Hat Enterprise Linux 7.x、または相当コミュニティー版

```
$ 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

  ModelSim - Intel FPGA Edition は、SUSE Linux ではサポートされていません。お持ちのライセンスバージョンの Mentor Graphics ModelSim ソフトウェアを使用してください。

6. (RedHat Linux のみ) インテル Quartus Prime 付属の Mentor Graphics ModelSim ソフトウェアを使用する場合は、ModelSim へのパスを PATH 環境変数に追加します。

   例：

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

7. オプション：Platform Designer を使用してコンポーネントをシステムに統合するには、Platform Designer へのパスを PATH 環境変数に追加します。

   例：

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

上記の手順を完了すると、インテル HLS コンパイラーがシステムにインストールされます。インテル HLS コンパイラー i++ コマンドを使用してコンポーネントをコンパイルする前に、インテル HLS コンパイラー環境を初期化してください。これにより、i++ コマンドが正常に実行されます。詳しくは、インテル HLS コンパイラー・プロ・エディション環境の初期化 (9 ページ) を参照してください。

重要: インテル HLS コンパイラーでは、 <*quartus_installdir*>/gcc をそのツールチェーン・ディレクトリーとして使用します。この GCC のインストールは、HLS 関連のすべてのデザイン作業に使用します。

1.4. Microsoft* Wndows* システムでの インテル HLS コンパイラー・プロ・エディションのインストール

インテル HLS コンパイラーを Microsoft* Windows* システムにインストールするには、次の手順を実行します。

1. お使いのオペレーティング・システムのバージョンが、インテル HLS コンパイラーによってサポートされている次のいずれかであることを確認します。(Microsoft* Windows* 7 SP1、8.1 または 10)
2. 次のソフトウェア製品のいずれかをインストールします。
   - Microsoft Visual Studio 2017 Professional
   - Microsoft Visual Studio 2017 コミュニティ

   重要: インテル HLS コンパイラー・ソフトウェアでサポートしている Microsoft Visual Studio のバージョンは、HLS コンパイラーのエディションに対して指定されたバージョンのみです。

複数のバージョンの Visual Studio をお持ちの場合、Microsoft では、Visual Studio のバージョンのリリース順に Visual Studio バージョンをインストールすることを推奨しています。例えば、Visual Studio 2010 のインストールは、Visual Studio 2015 をインストールする前に行います。詳しくは、Microsoft Docs の複数バージョンの Visual Studio をインストールするを参照してください。

3. 現在のシステムやダウンロード済みのインストール・パッケージによっては、インストールの準備のために次の追加手順が必要になる場合があります。
   - 完全な インテル Quartus Prime インストール・パッケージを (FPGAs ダウンロード・センターの Quartus Prime ダウンロード・ページの**一式ファイル**タブから)インストールする場合 - インストール・パッケージ実行前の追加手順は不要です。
   - の更新を既存ののインストールで行う場合は、次の事項を実行します。がインストール済みであることを確認してください。へのパスに注意してください。このパス情報は、のインストール・ウィザードを完了するために必要です。現在ののバージョンの HLS ディレクトリーの名前を変更して、そのバージョンをバックアップとして保持します。例えば、のバージョンをのバージョンインストールにインストールする場合は、C:\intelFPGA_pro\\hls の名前を C:\intelFPGA_pro\\hls_old に変更します。
   - (FPGAs ダウンロード・センターの Quartus Prime ダウンロード・ページの個別ファイルタブから) 個別にダウンロードしたと (FPGAs ダウンロードセンターの Quartus Prime ダウンロードのページの追加ソフトウェアタブから) 個別にダウンロードしたのインストール・パッケージをインストールする場合 - 両方のインストール・パッケージが同じディレクトリーにあることを確認してください。インストーラーでは、のインストール・パッケージを検出し、両方のソフトウェア・パッケージをインストールします。
4. インテル HLS コンパイラー・プロ・エディションのダウンロード (6 ページ) でダウンロードしたパッケージをインストールします。

   システム要件、前提条件、ライセンス要件などの インテル Quartus Prime ソフトウェアのインストールの詳しい手順については、Intel FPGA Software Installation and Licensing を参照してください。
5. インテル Quartus Prime 付属の Mentor Graphics ModelSim ソフトウェアを使用する場合は、ModelSim へのパスを PATH 環境変数に追加します。

   例：

   ```
   set "PATH=%PATH%:<quartus_installdir>\modelsim_ase\win32aloem"
   ```
6. オプション：Platform Designer を使用してコンポーネントをシステムに統合するには、Platform Designer へのパスを PATH 環境変数に追加します。

   例：

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

上記の手順を完了すると、インテル HLS コンパイラーがシステムにインストールされます。インテル HLS コンパイラー i++ コマンドを使用してコンポーネントをコンパイルする前に、インテル HLS コンパイラー環境を初期化してください。これにより、i++ コマンドが正常に実行されます。詳しくは、インテル HLS コンパイラー・プロ・エディション環境の初期化 (9 ページ) を参照してください。

1.5. インテル HLS コンパイラー・プロ・エディション環境の初期化

インテル HLS コンパイラーの i++ コマンドを使用してコンポーネントをコンパイルする前に、いくつかの環境変数を設定して、i++ コマンドが正常に実行されるようにしてください。

インテル HLS コンパイラー環境の初期化スクリプトが適用されるのは、*現在の*ターミナルまたはコマンド・プロンプト・セッションの環境変数に対してのみです。インテル HLS コンパイラーの環境の初期化は、デザイン開発のためにターミナルまたはコマンド・プロンプト・セッションを開始するたびに行ってください。

現在のターミナルまたはコマンド・プロンプト・セッションを初期化して、インテル HLS コンパイラーを実行するには、次の手順を実行します。

- Linux システムでは、環境の初期化を次のとおり行います。
  a. ターミナルセッションを開始し、コマンドプロンプトから次のコマンドを実行します。

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

     <*quartus_installdir*> の場所は、インテル Quartus Prime インストールへのパスです。
     例：`/home/<username>/intelFPGA_pro/20.2`

     このスクリプトで作成するサブシェルでは、最初の作業シェルの環境の変更はしません。

     現在の作業シェルを変更する場合は、`source` コマンドを使用して次のスクリプトを呼び出します。

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

  環境初期化スクリプトには設定した環境変数が表示され、このターミナルセッションから `i++` コマンドが実行できるようになります。
- Windows システムでは、環境の初期化を次のとおり行います。
  a. Windows コマンド・プロンプト・セッションを開始し、次のコマンドをコマンドプロンプトから実行します。

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

     <*quartus_installdir*> の場所は、インテル Quartus Prime インストールへのパスです。
     例：`C:\intelFPGA_pro\20.2`

  これで、このコマンド・プロンプト・セッションからの `i++` コマンドが実行できます。

ヒント: 環境変数を恒久的に設定するには、環境変数の設定を永続的に変更するためのオペレーティング・システムの標準的な手順に従ってください。環境初期化スクリプトの出力を確認して、恒久的に設定する環境変数を決定します。

2. 高位合成 (HLS) のデザイン例およびチュートリアル

インテル HLS (高位合成) コンパイラー・プロ・エディションに含まれているデザイン例およびチュートリアルでは、サンプル・コンポーネントが用意されており、コンポーネントをモデル化またはコード化することにより、インテル HLS コンパイラーからデザインにとって最良の結果を得る方法を示します。

高位合成 (HLS) デザイン例

高位合成 (HLS) デザイン例では、さまざまなアルゴリズムを効果的に実装する方法を素早く確認し、インテル HLS コンパイラーから最良の結果を得ることができます。

HLS デザイン例は次の場所にあります。

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

<quartus_installdir> のある場所は、インテル Quartus Prime デザインスイートがインストールされているディレクトリーです。例 : `/home/<username>/intelFPGA_pro/20.2` または `C:\intelFPGA_pro\20.2`

デザイン例の実行方法については、次の項を参照してください。

- インテル HLS コンパイラー のデザイン例の実行 (Linux) (16 ページ)
- インテル HLS コンパイラー のデザイン例の実行 (Windows) (17 ページ)

表 2. HLS デザイン例

重点分野	名前	説明
線形代数	QRD	修正グラムシュミット・アルゴリズムを使用して、行列を QR 分解します。
信号処理	interp_decim_filter	シンプルで効率的な補間/デシメーション・フィルターを実装します。
シンプルなデザイン	counter	シンプルで効率的な 32 ビット・カウンター・コンポーネントを実装します。
ビデオ処理	YUV2RGB	基本的な YUV422 から RGB888 への色空間変換を実装します。
ビデオ処理	image_downsample	画像ダウンサンプリング・アルゴリズムを実装し、バイリニア補間を使用して、画像サイズを縮小します。

HLS デザインのチュートリアル

HLS デザインのチュートリアルでは、HLS 固有の重要なプログラミング概念のほか、優れたコーディング方法を示します。

各チュートリアルの README ファイルには、チュートリアルでカバーしている内容やチュートリアルの詳しい実行方法に関する説明が記載されています。

表 3. 任意精度データ型デザインのチュートリアル

名前	説明
下欄のチュートリアルは、インテル Quartus Prime システムの次の場所にあります。 `<quartus_installdir>/hls/examples/tutorials/ac_datatypes`	
ac_fixed_constructor	ac_fixed コンストラクターの使用方法を示します。コーディング・スタイルの小数の変動を使用して、より良い QoR (結果の品質) を得ることができます。
ac_fixed_math_library	インテル HLS コンパイラーの ac_fixed_math 固定小数点の数学ライブラリー関数の使用方法を示します。
ac_int_basic_ops	ac_int クラスで使用可能な演算子を示します。
ac_int_overflow	DEBUG_AC_INT_WARNING および DEBUG_AC_INT_ERROR キーワードの使用方法を示します。エミュレーション実行時のオーバーフロー検出に役立ちます。
下欄のチュートリアルは、インテル Quartus Prime システムの次の場所にあります。 `<quartus_installdir>/hls/examples/tutorials/hls_float`	
1_reduced_double	アプリケーションが hls_float からメリットを得る方法を示します。下線の種類を double から hls_float <11,44> (reduced double) に変更します。
2_explicit_arithmetic	hls_float バイナリー演算子の明示的なバージョンを使用して、浮動小数点算術演算を必要に応じて実行する方法を示します。
3_conversions	his_float 型のデザインに変換が現れるタイミングと、異なる変換モードを使用してコンパイル型定数を生成する方法を示します。異なる hls_float 型を使用します。

表 4. コンポーネント・メモリー・デザインのチュートリアル

名前	説明
下欄のチュートリアルは、インテル Quartus Prime システムの次の場所にあります。 `<quartus_installdir>/hls/examples/tutorials/component_memories`	
attributes_on_mm_slave_arg	Avalon® Memory Mapped (MM) スレーブ引数をメモリー属性に適用する方法を示します。
exceptions	定数上のメモリー属性および struct メンバーを使用する方法を示します。
memory_bank_configuration	各メモリーバンクのロードポート/ストアポートの数を制御し、コンポーネント領域の使用率とスループットのいずれかまたはその両方を最適化する方法を示します。次のメモリー属性を 1 つ以上使用します。 • hls_max_replicates • hls_singlepump • hls_doublepump • hls_simple_dual_port_memory • non_power_of_two_memory • non_trivial_initialization
memory_geometry	メモリーをバンクに分割し、各メモリーバンクのロードポート/ストアポートの数を制御する方法を示します。次のメモリ属性を 1 つ以上使用します。 • hls_bankwidth • hls_numbanks • hls_bankbits
memory_implementation	レジスターの変数もしくは配列、MLAB、または RAM の実装方法を示します。次のメモリー属性を使用します。

continued...

名前	説明
	• hls_register • hls_memory • hls_memory_impl
memory_merging	リソース使用率を向上させる方法を示します。2 つのロジックメモリーを 1 つの物理メモリーとして実装します。hls_merge メモリー属性を使用して、深さ方向または幅方向にマージします。
non_power_of_two_memory	force_pow2_depth メモリー属性を使用して、2 のべき乗以外の深さのメモリーのパディングを制御する方法と、それによって FPGA メモリーリソース使用率がどのように影響されるかを示します。
non_trivial_initialization	C++ キーワード constexpr を使用して読み出し専用変数の効率的な初期化を実現する方法を示します。
static_var_init	コンポーネント内のスタティックの初期化動作を制御する方法を示します。 hls_init_on_reset または hls_init_on_powerup メモリー属性を使用します。

表 5. インターフェイス・デザインのチュートリアル

名前	説明
下欄のチュートリアルは、インテル Quartus Prime システムの次の場所にあります。 *<quartus_installdir>*/hls/examples/tutorials/interfaces	
overview	コンポーネント・アルゴリズムが同じ場合に、異なるコンポーネント・インターフェイスを選択することによる QoR (結果の品質) への影響を示します。
explicit_streams_buffer	明示的な stream_in インターフェイスおよび stream_out インターフェイスをコンポーネントおよびテストベンチで使用する方法を示します。
explicit_streams_packets_empty	usesPackets、usesEmpty および firstSymbolInHighOrderBits ストリーム・テンプレート・パラメーターを使用する方法を示します。
explicit_streams_packets_ready_valid	usesPackets、usesValid および usesReady ストリーム・テンプレート・パラメーターを使用する方法を示します。
mm_master_testbench_operators	Avalon Memory Mapped (MM) Master (mm_master クラス) インターフェイスの異なるインデックスでコンポーネントを呼び出す方法を示します。
mm_slaves	Avalon-MM Slave インターフェイス (スレーブレジスターおよびスレーブメモリー) を作成する方法を示します。
mm_slaves_double_buffering	hls_readwrite_mode マクロを使用してメモリーマスターからスレーブメモリーへアクセスする方法を制御することによる効果を示します。
mm_slaves_csr_volatile	volatile キーワードを使用してコンポーネントの実行中にスレーブメモリーへの同時アクセスを許可することによる効果を示します。
multiple_stream_call_sites	複数のストリーム呼び出しサイトを使用する利点を示します。
pointer_mm_master	Avalon-MM Master インターフェイスおよびそのパラメーターを作成する方法を示します。
stable_arguments	stable 属性を不変の引数に使用してリソース使用率を向上させる方法を示します。

表 6. ベスト・プラクティス・デザインのチュートリアル

名前	説明
下欄のチュートリアルは、インテル Quartus Prime システムの次の場所にあります。 *<quartus_installdir>*/hls/examples/tutorials/best_practices	
ac_datatypes	int データ型の代わりに ac_int データ型を使用することによる効果を示します。

continued...

名前	説明
const_global	const 修飾グローバル変数を使用したパフォーマンスおよびリソース使用率の向上を示します。
divergent_loops	分岐ループのあるデザインのソースレベルの最適化を示します。
floating_point_contract	fp_contract オプションを使用してデザインの倍精度浮動小数点演算のパフォーマンスを向上させる方法を示します 。
floating_point_ops	I++ 内の --fpc および --fp-relaxed フラグによる浮動小数点演算への影響を示します。スループット用に最適化された 32 タップ有限インパルス応答 (FIR) フィルターデザインを使用します。
fpga_reg	fpga_reg マクロを使用してデザインのパイプラインを正確に調整する方法を示します。
hyper_optimized_handshaking	インテル HLS コンパイラー i++ コマンドの --hyper-optimized-handshaking オプションを使用する方法を示します。
loop_coalesce	ネスト化ループで loop_coalesce プラグマを使用したパフォーマンスとリソース使用率の向上を示します。 #pragma loop_coalesce は、スタンダード・エディションとプロ・エディションの両方にありますが、デザイン・チュートリアルがあるのは、プロ・エディションのみです。
loop_fusion	ループ・フュージョンのレイテンシーおよびリソース使用率の向上を示します。
loop_memory_dependency	Ivdep プラグマを使用してループ運搬依存を解除する方法を示します。
lsu_control	可変レイテンシー Avalon MM Master インターフェイス向けにインスタンス化された LSU 型を制御する効果を示します。
parallelize_array_operation	f_{MAX} を向上させる方法を示します。ループ内の配列に対する動作の実行時に発生するボトルネックを修正します。
optimize_ii_using_hls_register	hls_register 属性を使用して、ループ II を削減する方法や、hls_max_concurrency を使用してコンポーネントのスループットを向上させる方法を示します。
parameter_aliasing	コンポーネント引数で restrict キーワードを使用する方法を示します。
random_number_generator	乱数ジェネレーター・ライブラリーを使用する方法を示します。
reduce_exit_fifo_width	f_{MAX} を向上させる方法を示します。ストールのないクラスターの出口ノードに属する FIFO の幅を小さくします。
relax_reduction_dependency	浮動小数点アキュムレーターを含むループの II を削減する方法、またはその他の単一のクロックサイクルでの高速計算ができない削減演算を示します。
remove_loop_carried_dependency	ループのパフォーマンスを向上させる方法を示します。ネストされたループ間での同じ変数へのアクセスを削除します。
resource_sharing_filter	32 タップ有限インパルス応答 (FIR) フィルターデザインのエリア最適化バリアントを示します
set_component_target_fmax	hls_scheduler_target_fmax_mhz コンポーネント属性を使用する方法と、それが ii ループプラグマとどのように相互作用するかを示します。
shift_register	シフトレジスターを実装するための推奨コーディング・スタイルを示します。
sincos_func	sin または cos 関数の代わりに sinpi または ospi 関数をコンポーネントで使用した場合の効果を示します。
single_vs_double_precision_math	倍精度リテラルおよび関数の代わりに単精度リテラルおよび関数を使用した場合の効果を示します。
stall_enable	ストール・フリー・クラスターをストール・イネーブル・クラスターに置き換え、一部の小規模なデザインのレイテンシーを改善する方法を示します。
struct_interface	ac_int を使用して、パディングビットなしでインターフェイスを実装する方法を示します。

continued...

名前	説明
submnormal_and_rounding	--daz および --rounding i++ コマンドオプション使用の効果を示します。
swap_vs_copy	レジスターを使ったディープコピーによる、コンポーネント・デザインのパフォーマンスおよびリソース使用率への影響を示します。
triangular_loop	依存関係を持つ三角ループパターンを記述する方法を示します。

表 7. ユーザビリティー・デザインのチュートリアル

名前	説明
下欄のチュートリアルは、インテル Quartus Prime システムの次の場所にあります。 <quartus_installdir>/hls/examples/tutorials/usability	
compiler_interoperability	(Linux のみ) GCC でコンパイルされたテストベンチ・コードを、i++ コマンドでコンパイルされたコードと共に使用する方法を示します。
enqueue_call	コンポーネントを非同期で実行し、テストベンチでパイプライン・パフォーマンスを実行する方法を示します。エンキュー機能を使用します。
platform_designer_2xclock	clock2x 入力を備えたコンポーネント向けに推奨されるクロックおよびリセットの生成を示します。
platform_designer_stitching	複数のコンポーネントを結合して 1 つのまとまったデザインとして機能させる方法を示します。

表 8. タスクのシステムのデザインのチュートリアル

名前	説明
下欄のチュートリアルは、インテル Quartus Prime システムの次の場所にあります。 <quartus_installdir>/hls/examples/tutorials/system_of_tasks	
balancing_loop_delay	タスクのシステムを使用しているコンポーネントのスループットを改善する方法を示します。ストリームをバッファリングします。
balancing_pipeline_latency	タスクのシステムを使用しているコンポーネントのスループットを改善する方法を示します。ストリームをバッファリングします。
interfaces_sot	タスク間で情報を転送する方法を示します。Avalon ストリーミングおよび Avalon メモリーマップド・マスター・インターフェイスを使用します。
internal_stream	HLS タスクで ihc::stream オブジェクトを使って「内部ストリーム」を使用する方法を示します。
parallel_loop	パイプライン方式で順次ループを実行する方法を示します。コンポーネントの HLS タスクのシステムを使用します。
resource_sharing	コンポーネント内の高価な計算ブロックを共有して、領域使用率を節約する方法を示します。
task_reuse	同じタスク関数の複数コピーを呼び出す方法を示します。

表 9. HLS ライブラリー・デザインのチュートリアル

名前	説明
下欄のチュートリアルは、インテル Quartus Prime システムの次の場所にあります。 <quartus_installdir>/hls/examples/tutorials/libraries	
basic_rtl_library_flow	RTL ライブラリーを開発し、それを HLS コンポーネントで使用するプロセスを示します。
rtl_struct_mapping	C++ 構造体フィールドから RTL モジュール・インターフェイス信号のビットスライスへのマッピングを取得する方法を示します。

表 10. HLS ループ制御のチュートリアル

名前	説明
下欄のチュートリアルは、インテル Quartus Prime システムの次の場所にあります。 `<quartus_installdir>/hls/examples/tutorials/loop_controls`	
`max_interleaving`	次の条件を満たすループの領域使用率を削減する方法を示します。 • ループに II > 1 があること • ループがパイプライン・ループに含まれていること • ループの実行が、パイプライン化されたループの呼び出し全体でシリアル化されること

2.1. インテル HLS コンパイラー のデザイン例の実行 (Linux)

Linux システムで インテル HLS コンパイラー のデザイン例を実行するには、次の手順に従います。

1. ターミナルセッションをスタートし、インテル HLS コンパイラー環境を初期化します。

   環境の初期化については、インテル HLS コンパイラー・プロ・エディション環境の初期化 (9 ページ) を参照してください。

2. `<quartus_installdir>/hls/examples/<design_example_name>` ディレクトリーに移動します。`<quartus_installdir>`は、インテル Quartus Prime をインストールしたディレクトリーです。

   例：`/home/<username>/intelFPGA_pro/20.2`

3. `make test-x86-64` コマンドを実行します。このコマンドでは、C++ ソースコードを x86-64 バイナリー実行可能ファイルにコンパイルします。次に、生成された実行可能ファイルを CPU で実行します。
   `make test-x86-64` コマンド実行後に予想される結果は次のとおりです。

   - バイナリーの生成に使用されたコマンドがコンソールに表示されます。例：
     `i++ -march=x86-64 -o test-x86-64 <source_files>`
   - HLS コンパイラーによって実行可能ファイル (例：`test-x86-64`) が現在の作業ディレクトリーに作成されます。
   - 実行可能ファイルの出力がコンソールに表示され、正常に実行されたことが分かります。

   ```
   $ make test-x86-64
   i++ MGS.cpp QRD_Testbench.cpp TestbenchHelpers.cpp  --fpc --fp-relaxed -
   march=x86-64 -o test-x86-64
   +----------------------------------------+
   | Run ./test-x86-64 to execute the test. |
   +----------------------------------------+
   ```

4. `make test-fpga` コマンドを実行します。このコマンドでは、C++ ソースコードをハードウェア実行可能ファイルにコンパイルし、その後、生成された HDL のシミュレーションを実行します。
   `make test-fpga` コマンド実行後に予想される結果は次のとおりです。

- テストベンチ・バイナリーの生成に使用されたコマンドがコンソールに表示されます。例：`i++ -march="<FPGA_family_or_part_number>" <source_files> -o test-fpga`
- HLS コンパイラーによって、`.prj` ディレクトリー (例：`test-fpga.prj`) が現在作業中のディレクトリーに作成されます。
- 実行可能ファイルの出力がコンソールに表示され、正常に実行されたことが分かります。

```
$ make test-fpga
i++ MGS.cpp QRD_Testbench.cpp TestbenchHelpers.cpp  -v --fpc --fp-relaxed -
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. |
+-------------------------------------+
```

2.2. インテル HLS コンパイラー のデザイン例の実行 (Windows)

Windows システムで インテル HLS コンパイラーのデザイン例を実行するには、次の手順に従います。

1. ターミナルセッションを開始し、インテル HLS コンパイラー環境を初期化します。

   環境の初期化については、インテル HLS コンパイラー・プロ・エディション環境の初期化 (9 ページ) を参照してください。

2. `<quartus_installdir>\hls\examples\<design_example_name>` ディレクトリーに移動します。ここで `<quartus_installdir>` は、インテル Quartus Prime 開発ソフトウェアをインストールしたディレクトリーです。

   例：`C:\intelFPGA_pro\20.2`

3. `build.bat test-x86-64` コマンドを実行します。このコマンドでは、C++ ソースコードを x86-64 バイナリ実行可能ファイルにコンパイルします。次に、生成された実行可能ファイルを CPU で実行します。
   `build.bat test-x86-64` コマンド実行後に予想される結果は次のとおりです。

- バイナリーの生成に使用されたコマンドがコンソールに表示されます。例：
  `i++ -march=x86-64 -o test-x86-64 <source_files>`
- コンパイラーによって、実行可能ファイル (例：`test-x86-64.exe`) が、現在の作業ディレクトリーに作成されます。
- 実行可能ファイルの出力がコンソールに表示され、正常に実行されたことが分かります。

```
C:\intelFPGA_pro\20.2\hls\examples\QRD>build.bat test-x86-64
i++ --fpc --fp-relaxed -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. `build.bat test-fpga` コマンドを実行します。このコマンドでは、C++ ソースコードをハードウェア実行可能ファイルにコンパイルし、その後、生成された HDL のシミュレーションを実行します。
   `build.bat test-fpga` コマンド実行後に予想される結果は次のとおりです。
   - テストベンチ・バイナリーの生成に使用されたコマンドがコンソールに表示されます。例：
     `i++ -march="<FPGA_family_or_part_number>" <source_files> -o test-fpga`
   - HLS コンパイラーによって、`.prj` ディレクトリー (例：`test-fpga.prj`) が現在作業中のディレクトリーに作成されます。
   - 実行可能ファイルの出力がコンソールに表示され、正常に実行されたことが分かります。

```
C:\intelFPGA_pro\20.2\hls\examples\QRD>build.bat test-fpga
i++ --fpc --fp-relaxed -march=Arria10 MGS.cpp QRD_Testbench.cpp
TestbenchHelpers.cpp -o test-fpga.exe
Run test-fpga.exe to execute the test.
```

3. インテル HLS コンパイラーのセットアップのトラブルシューティング

このセクションで提供する情報は、HLS コンパイラーのセットアップ時に発生する可能性がある問題のトラブルシューティングに役立ちます。

3.1. インテル HLS コンパイラーのライセンスの問題

インテル HLS (高位合成) コンパイラーのライセンスは、インテル Quartus Prime ライセンスの一部です。ただし、インテル HLS コンパイラーは、ModelSim ソフトウェアに依存しています。使用する ModelSim ソフトウェアのバージョンが、ModelSim - インテル FPGA エディション または ModelSim - インテル FPGA スタンダード・エディション以外の場合は、ご使用の ModelSim ソフトウェアのバージョンが正しくライセンスされていることを確認してください。

場合によっては、ModelSim ソフトウェアのライセンスの問題が発生することがあります 。

3.1.1. ModelSim ライセンスのエラーメッセージ

HLS コンパイラーでエラーメッセージを発行するのは、ModelSim ソフトウェアのインストール済みバージョンのライセンスが見つからない場合です。

ModelSim ソフトウェアのライセンスが見つからない場合、HLS コンパイラーでは、デザインを FPGA アーキテクチャーにコンパイルする際に次のエラーメッセージを発行します。

```
$ 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
```

エラーの一般的な原因は次のとおりです。

- ライセンスが見つからない、期限切れ、または無効
- `license.dat` ファイルのライセンスサーバー名が正しくない
- ライセンスの場所が指定されていない、または正しく指定されていない

注意: HLS コンパイラーの実行速度が低下する可能性があるのは、コンパイラーでネットワークを検索してもライセンスが見つからなかったり、ライセンスが破損している場合です。このような問題が発生した場合は、ライセンスファイルまたはライセンスの場所を修正してください。

3.1.2. LM_LICENSE_FILE 環境変数

インテル およびサードパーティーのソフトウェアでは、*LM_LICENSE_FILE* 環境変数を使用して、ライセンスファイルまたはライセンスサーバーの場所を指定します。例えば、インテル Quartus Prime 開発ソフトウェアと ModelSim ソフトウェアの両方で、*LM_LICENSE_FILE* 変数を使用してライセンスの場所を指定します。

注意: 開発マシンによるライセンスサーバーとの通信に要する時間は、コンパイル時間に直接影響します。*LM_LICENSE_FILE* 環境変数の設定に多数のライセンスサーバーへのパスが含まれる場合、またはライセンスサーバーが遠隔地でホストされている場合は、コンパイル時間が大幅に増加します。

Linux または UNIX システムでは、*LM_LICENSE_FILE* 環境変数に追加する各ライセンスファイルまたはライセンスサーバーの場所の後にコロン (:) を挿入します。

Windows システムでは、*LM_LICENSE FILE* 環境変数に追加する各ライセンスファイルまたはライセンスサーバーの場所の後にセミコロン (;) を挿入します。

注意: *LM_LICENSE_FILE* の設定を変更してソフトウェア・ライセンスの場所を含める場合は、変数に追加されている既存ライセンスの場所は削除しないでください。

3.1.2.1. ModelSim ソフトウェア・ライセンス 固有の考慮事項

ModelSim ソフトウェア・ライセンスをセットアップする場合、ライセンスの場所を *LM_LICENSE_FILE* 環境変数に追加する必要があります。ModelSim ソフトウェア・ライセンスの場所を *MGLS_LICENSE_FILE* 環境変数に追加することも可能です。

Mentor Graphics のアプリケーションでは、ModelSim ソフトウェアも含めて、ライセンスファイルおよびライセンスサーバーへのパスは、5 つの異なる場所に指定することができます。ライセンスファイルまたはライセンスサーバーへのパスを複数の場所に指定する場合、次の検索順で有効なパスを検索します。

- ユーザー環境で設定した *MGLS_LICENSE_FILE* 環境変数
- レジストリーで設定した *MGLS_LICENSE_FILE* 環境変数
- ユーザー環境で設定した *LM_LICENSE_FILE* 環境変数
- レジストリーで設定した *LM_LICENSE_FILE* 環境変数
- `<path to FLEXlm>\license.dat`、ここでは *<path to FLEXlm>* は、FLEXlm ライセンスファイルのデフォルトの場所です。

Mentor Graphics 製品のライセンスをコンピューターにインストールする場合、*MGLS_LICENSE_FILE* 環境変数の設定は、*LM_LICENSE_FILE* 環境変数の設定よりも優先されます。両方の環境変数を設定する場合、*LM_LICENSE_FILE* は ModelSim ライセンスサーバーを指すように設定し、*MGLS_LICENSE_FILE* は他の Mentor Graphics アプリケーション用のライセンスサーバーのみを指すように設定します。*MGLS_LICENSE_FILE* 環境変数のみを使用する場合は、ModelSim ライセンスサーバーと他の Mentor Graphics アプリケーション用のライセンスサーバーが、同じマシン上にあることを確認してください。

A. インテル HLS コンパイラー・プロ・エディション スタートガイドのアーカイブ

インテル HLS コンパイラーバージョン	タイトル
20.1	Intel HLS Compiler Pro Edition Getting Started Guide
19.4	Intel HLS Compiler Pro Edition Getting Started Guide
19.3	Intel HLS Compiler Getting Started Guide
19.2	Intel HLS Compiler Getting Started Guide
19.1	Intel HLS Compiler Getting Started Guide
18.1.1	Intel HLS Compiler Getting Started Guide
18.1	Intel HLS Compiler Getting Started Guide
18.0	Intel HLS Compiler Getting Started Guide
17.1.1	Intel HLS Compiler Getting Started Guide
17.1	Intel HLS Compiler Getting Started Guide

B. インテル HLS コンパイラー・プロ・エディション スタートガイドの改訂履歴

ドキュメント・バージョン	インテル Quartus Prime プロ・エディションバージョン	変更内容
2020.06.22	20.2	• 次のチュートリアルを追加しました。高位合成 (HLS) のデザイン例およびチュートリアル (11 ページ): — `remove_loop_carried_dependency` — `stall_enable`
2020.04.13	20.1	• インテル HLS コンパイラー・プロ・エディションの前提条件 (4 ページ) および Linux システムでの インテル HLS コンパイラー・プロ・エディションのインストール (6 ページ) の Linux の前提条件を改訂しました。 • 次のチュートリアルを高位合成 (HLS) のデザイン例およびチュートリアル (11 ページ) に追加しました。 — `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` • *GCC をオフラインでインストールする*を削除しました。この情報は V20.1 以降では不要です。
2019.12.16	19.4	• インテル HLS コンパイラー・スタンダード・エディションに関する情報を削除しました。インテル HLS コンパイラー・スタンダード・エディションのインストールおよびコンフィグレーションの情報については、Intel High Level Synthesis Compiler Standard Edition: Getting Started Guide を参照してください。 • *GCC をオフラインでインストールする*を追加しました。

インテル HLS コンパイラー・プロ・エディション スタートガイドの文書改訂履歴

旧バージョンの インテル HLS コンパイラー スタートガイドには、インテル HLS コンパイラー・スタンダード・エディションおよび インテル HLS コンパイラー・プロ・エディションの両方の情報が含まれています。

ドキュメント・バージョン	インテル Quartus Prime のバージョン	変更内容
2019.09.30	19.3	• PRO 次のチュートリアルを削除しました。 — bank_bits — mm_slave — ROM — struct_member_attributes • 次のチュートリアルを追加しました。 — memory_bank_configuration — memory_geometry — memory_implementation — memory_merging — static_var_init — attributes_on_mm_slave_arg — exceptions — lsu_control — relax_reducation_dependency • PRO サポートされている C++ コンパイラーに Microsoft Visual Studio 2017 Professional および Microsoft Visual Studio 2017 Community を追加しました。 • PRO サポートされている C++ コンパイラーから Microsoft Visual Studio 2015 Professional および Microsoft Visual Studio 2015 Community を削除しました。
2019.07.01	19.2	• Linux システムでの インテル HLS コンパイラー・プロ・エディションのインストール (6 ページ) を更新して、ModelSim - インテル FPGA エディションに必要な 32 ビット・ライブラリーをインストールするための実行コマンドを明確にしました。
2019.04.01	19.1	• インテル HLS コンパイラーは、インテル Quartus Prime とは個別にインストールできるようになりました。次のセクションを追加または更新しました。 — インテル HLS コンパイラー・プロ・エディションのダウンロード (6 ページ) — Linux システムでの インテル HLS コンパイラー・プロ・エディションのインストール (6 ページ) — Microsoft* Wndows* システムでの インテル HLS コンパイラー・プロ・エディションのインストール (8 ページ) • インテル HLS コンパイラーは、インテル Quartus Prime の旧バージョンと下位互換性を持つようになりました。詳しくは、インテル HLS コンパイラー・プロ・エディションの下位互換性 (5 ページ) を参照してください。 • インテル HLS コンパイラー・プロ・エディション環境の初期化 (9 ページ) を更新して、Visual Studio x64 Native Tools コマンドプロンプトから Windows オペレーティング・システムの インテル HLS コンパイラー環境を初期化するという要件を含めました。 • 高位合成 (HLS) のデザイン例およびチュートリアル (11 ページ) の インテル HLS コンパイラーで提供されるチュートリアルのリストの改訂および更新を行いました。
2018.12.24	18.1	• PRO Microsoft Visual Studio 2015 Community を Microsoft Windows システムでサポートされる C++ コンパイラーのリストに追加しました。 • インテル Quartus Prime 付属の Mentor Graphics ModelSim ソフトウェアへのパスをオペレーティング・システムの PTTH 環境変数に追加する手順を Linux システムでの インテル HLS コンパイラー・プロ・エディションのインストール (6 ページ) および Microsoft* Wndows* システムでの インテル HLS コンパイラー・プロ・エディションのインストール (8 ページ) に追加しました。

continued...

ドキュメント・バージョン	インテル Quartus Prime のバージョン	変更内容
2018.09.24	18.1	• PRO インテル HLS コンパイラーには新しいフロントエンドがあります。この新しいフロントエンドによって導入された変更の概要については、Intel HLS Compiler Version 18.1 Release Notes の *Improved Intel HLS Compiler Front End* を参照してください。 • PRO インテル Quartus Prime プロ・エディション付属の インテル HLS コンパイラー には、新しい前提条件があります。詳しくは インテル HLS コンパイラー・プロ・エディションの前提条件 (4 ページ) を参照してください。 • PRO Linux システムのインストール手順を変更しました。詳しくは Linux システムでの インテル HLS コンパイラー・プロ・エディションのインストール (6 ページ) を参照してください。 • PRO 高位合成 (HLS) のデザイン例およびチュートリアル (11 ページ) の `best_practices/parameter_aliasing` チュートリアルの説明を変更して、`__restrict` キーワードがカバーされるようにしました。`restrict` キーワードの インテル HLS コンパイラー・プロ・エディションでのサポートは終了しました。 • PRO `best_practices/integer_promotion` チュートリアルを削除しました。整数の昇格は、インテル HLS コンパイラー・プロ・エディションの使用時にデフォルトで行われるようになりました。
2018.05.07	18.0	• インテル Quartus Prime のバージョン 18.0 以降では、インテル HLS コンパイラーでサポートする機能とデバイスは、お使いの インテル Quartus Prime のエディションよって異なります。インテル HLS コンパイラーのドキュメントでは、アイコンを使用して、特定のエディションにのみ適用される内容や機能を次のように示しています。 PRO このアイコンでは、機能または内容はが インテル Quartus Prime プロ・エディションに付属の インテル HLS コンパイラーのみに適用されることを示します。 STD このアイコンでは、機能または内容が インテル Quartus Prime スタンダード・エディションに付属の インテル HLS コンパイラーのみに適用されることを示します。 • PRO 次のチュートリアルを高位合成 (HLS) のデザイン例およびチュートリアル (11 ページ) のチュートリアルのリストに追加しました。 — `interfaces/explicit_streams_packets_empty` — `interfaces/explicit_streams_ready_latency` — `best_practices/ac_datatypes` — `best_practices/loop_coalesce` — `best_practices/random_number_generator` • PRO 次のチュートリアルの名前を変更して、インテル Quartus Prime コンポーネント名を反映させました。 — `usability/qsys_2xclock` を `usability/platform_designer_2xclock` に変更しました。 — `usability/qsys_stitching` を `usability/platform_designer_stitching` に変更しました。
2017.12.22	17.1.1	• `interfaces/overview` チュートリアルを高位合成 (HLS) のデザイン例およびチュートリアル (11 ページ) のチュートリアルのリストに追加しました。
2017.12.08	17.0	• Mentor Graphics ModelSim ソフトウェア要件を更新して、必要な Red Hat 開発ツールパッケージを含めました。

continued...

ドキュメント・バージョン	インテル Quartus Prime のバージョン	変更内容
2017.11.06	17.0	• インテル HLS (高位合成) コンパイラーが インテル Quartus Prime デザインスイートの一部になったことに伴い、次の変更を行いました。 — ドキュメントを改訂して、インテル Quartus Prime 開発ソフトウェアのインストールによって インテル HLS コンパイラーが入手できることを反映しました。 — ほとんどのライセンス情報を削除しました。インテル HLS コンパイラーのライセンスは、現在 インテル Quartus Prime デザインスイートのライセンスによりカバーされています。HLS コンパイラーで必要とするサードパーティーのソフトウェアの一部では、追加ライセンスが引き続き必要な場合があります。 — コンパイラーの上書きに関する情報を追加しました。 — 前提条件を改訂して、HLS コンパイラーで必要とされる前提条件のみを反映するようにしました。 — パス情報を改訂して、インテル HLS コンパイラー・ファイルの新しいファイルシステムの場所を反映しました。 • 次のチュートリアル名を変更しました。 — `explicit_streams` チュートリアルを `explicit_streams_buffer` に変更しました。 — `explicit_streams_2` チュートリアルを `explicit_streams_packets_ready_valid` に変更しました。
2017.06.23	—	• 軽微な変更および訂正を行いました。
2017.06.09	—	• 高位合成 (HLS) のデザイン例およびチュートリアル (11 ページ) の新しい例に関する情報を更新しました。 • *インテル HLS コンパイラーのデフォルト GCC コンパイラーのオーバーライド*を改訂しました。
2017.03.14	—	• 提供デザイン例のリストからビット演算 (3DES) を削除しました。
2017.03.01	—	• Linux に必要なパッケージおよびライブラリーのインストールを追加しました。
2017.02.03	—	• クイックスタートの項の成功メッセージを PASSED に変更しました。 • *HLS デザイン例およびチュートリアル*の章を追加しました。 • *Linux での HLS デザイン例の実行*および *Windows での HLS デザイン例の実行*を *HLS デザイン例およびチュートリアル*に移動しました。
2016.11.30	—	• *HLS コンパイラーの前提条件*で、ソフトウェア要件を更新して、HLS コンパイラーでは、インテル Quartus Prime 開発ソフトウェアでサポートする ModelSim ソフトウェアのすべてのエディションをサポートしていることを追記しました。 • *HLS コンパイラー・クイックスタート*で、`init_hls` スクリプトの実行が、シェルまたはターミナルを起動してデザインを開発するたびに必要であることを追記しました。 • HLS コンパイラー・クイックスタートで、Linux と Windows の説明を分けました。 • *HLS デザイン例の実行*で、Linux と Windows の説明を分けて、Linux では `make` コマンドを実行、Windows では `build.bat` コマンドを実行するという内容に更新しました。 • `test_x86-64` コマンドオプションを `test-x86-64` に変更しました。 • `test_fpga` コマンドオプションを `test-fpga` に変更しました。 • Linux での `make test_qii` コマンド、および Windows での `build.bat test_qii` コマンドの実行の説明は、不要になったため削除しました。 • *HLS ライセンスのエラーメッセージ*で、HLS コンパイラーによる ModelSim ソフトウェア・ライセンスの検出が出来なかった場合に表示されるエラーメッセージを更新しました。
2016.09.12	—	• 初版