

Vivado Design Suite ユーザー ガイド

システム レベル デザイン入力

UG895 (v2013.1) 2013 年 3 月 20 日



Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

© Copyright 2012-2013 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

本資料は英語版 (v2013.1) を翻訳したもので、内容に相違が生じる場合には原文を優先します。

資料によっては英語版の更新に対応していないものがあります。

日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com までお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	改訂内容
2013/03/20	2013.1	<ul style="list-style-type: none">第 1 章の「Vivado Design Suite の起動」を追加第 2 章「プロジェクトの操作」のさまざまな手順に関連する Tcl コマンドを追加「IP ソースの操作」をアップデート、「IP インテグレーターのソースの操作」を追加、Vivado™ IP インテグレーターに関する注意書きを第 3 章「ソース ファイルの操作」の「エンベデッド ソースの操作」に追加さまざまな手順に関連する Tcl コマンドを追加、RTL レベルで制約を作成および追加する方法を第 4 章「RTL デザインのエラボレーション」に追加図 2-16、図 3-1、図 3-2、図 3-5、図 3-7、図 3-13、図 3-14、図 3-15、図 3-16、図 3-17、図 3-21、図 3-25、図 3-26、図 3-30、図 4-2、および図 4-3 をアップデート

目次

改訂履歴	2
第 1 章：はじめに	
概要	4
Vivado Design Suite の起動	4
第 2 章：プロジェクトの操作	
概要	6
プロジェクト タイプ	6
プロジェクトの作成	7
プロジェクトの管理	18
プロジェクト サマリの使用	21
プロジェクト設定	23
Tcl スクリプトを使用したプロジェクトの作成	30
第 3 章：ソース ファイルの操作	
概要	32
プロジェクト モードでのソースの操作	32
非プロジェクト モードでのソースの操作	63
第 4 章：RTL デザインのエラボレーション	
概要	65
プロジェクト モードでのデザインのエラボレーション	65
非プロジェクト モードでのデザインのエラボレーション	70
第 5 章：デザインのデバッグ	
概要	72
RTL レベルのデザイン シミュレーション	72
インシステム デバッグ	72
付録 A：その他のリソース	
ザイリンクス リソース	73
ソリューション センター	73
リファレンス	73

はじめに

概要

Vivado™ Design Suite では、レジスタトランスファーレベル (RTL) を作成するとビットストリームまで生成できます。システムレベルデザイン入力には、プロジェクトの作成、ソースファイルの作成および追加、RTL デザインのエラボレート、デバッグ情報の挿入およびコンフィギュレーションなど、デザインの設定が含まれます。デザインはグラフィカルユーザーインターフェイス (GUI) である Vivado IDE (Integrated Design Environment) または Tcl コマンド/スクリプトを使用して入力できます。

注記 : 本書には、2013.1 リリースでは早期アクセス機能としてライセンス付与される新しい Vivado IP インテグレーターの環境に関する情報が含まれます。ライセンスの取得については、フィールドアプリケーションエンジニア (FAE) にご連絡ください。

Vivado Design Suite の起動

Vivado Design Suite は、さまざまな方法で起動できます。たとえば、Tcl スクリプトベースのコンパイル方法を使用すると、ソースやデザインプロセスをユーザーが管理できます。この方法は、「非プロジェクトモード」と呼ばれます。また、プロジェクトベースの方法を使用して、プロジェクトおよびプロジェクトの状態を使用して、デザインプロセスおよびデザインデータを自動的に管理させることもできます。この方法は、「プロジェクトモード」と呼ばれます。どちらの方法でも Tcl スクリプトのバッチモードで実行できるほか、Vivado IDE でインタラクティブに実行できます。さまざまなデザインフローモードの詳細は、『Vivado Design Suite User Guid ユーザーガイド : デザインフローの概要』(UG892) [参照 1] を参照してください。

Tcl プロジェクトの起動

Tcl を直接使用する場合は、Tcl コマンドで次のいずれかの方法を使用してデザインを処理します。

- Vivado IDE の外の Vivado Design Suite Tcl シェルに Tcl コマンドを入力します。
- Vivado IDE の一番下の Tcl コンソールに Tcl コマンドを入力します。
- Vivado Design Suite Tcl シェルから Tcl スクリプトを実行します。
- Vivado IDE から Tcl スクリプトを実行します。

Tcl および Tcl スクリプトに関する情報は、『Vivado Design Suite ユーザーガイド : Tcl スクリプト機能の使用』(UG894) [参照 2] を参照してください。Vivado ツールでの Tcl の使用方法の詳細については、『Vivado Design Suite チュートリアル : デザインフロー概要』(UG888) [参照 3] を参照してください。

Vivado Design Suite Tcl シェルの起動

Linux コマンド プロンプトまたは Windows のコマンド プロンプトで次のコマンドを使用して、Vivado Design Suite Tcl シェルを起動します。

```
vivado -mode tcl
```

注記: Windows の場合、[スタート] → [すべてのプログラム] → [Xilinx Design Tools] → [Vivado 2013.x] → [Vivado 2013.x Tcl Shell] をクリックしても起動できます。

バッチ Tcl スクリプトを使用した Vivado ツールの起動

Vivado ツールはを起動したときに Tcl スクリプトを提供しておく、バッチ モードで使用できます。Linux コマンド プロンプトまたは Windows のコマンド プロンプトで次のコマンドを使用します。

```
vivado -mode batch -source <your_Tcl_script>
```

注記: バッチ モードの場合、指定したスクリプトが実行されてから Vivado ツールが閉じます。

Vivado IDE の起動

GUI を使用する場合は、Windows または Linux から Vivado IDE を起動します。Vivado IDE の詳細は、『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』(UG893) [参照 4] を参照してください。



推奨: Vivado IDE は作業ディレクトリから起動してください。これにより、起動ディレクトリに書き込まれるプロジェクト ファイル、ログ ファイル、ジャーナル ファイルが見つけやすくなります。

Windows での Vivado IDE の起動

[スタート] → [すべてのプログラム] → [Xilinx Design Tools] → [Vivado 2013.x] → [Vivado 2013.x] をクリックします。

注記: または、デスクトップの Vivado IDE のショートカットをダブルクリックします。



図 1-1: Vivado IDE デスクトップ アイコン

Windows または Linux のコマンド ラインからの Vivado IDE の起動

コマンド プロンプトに次のコマンドを入力します。

```
vivado
```

注記: このコマンドを入力すると、自動的に `vivado -mode gui` が実行され、Vivado IDE が起動されます。ヘルプが必要な場合は、「`vivado -help`」と入力します。

Vivado Design Suite の Tcl シェルからの Vivado IDE の起動

コマンド プロンプトに次の Tcl コマンドを入力します。

```
start_gui
```

プロジェクトの操作

概要

プロジェクト モードで操作する場合は、さまざまなプロジェクト タイプを使用してデザインを入力できます。本章では、各プロジェクト タイプとそのプロジェクトの作成および管理方法について説明します。また、プロジェクト サマリ、プロジェクト設定のほか、Tcl スクリプトを使用したプロジェクトの作成方法についても説明します。

プロジェクト タイプ

Vivado™ IDE を使用すると、次のプロジェクト タイプを作成できます。各プロジェクト タイプの入力ソース タイプは異なります。

- レジスタトランスファー レベル (RTL) プロジェクト
- 合成後のプロジェクト
- I/O 配置プロジェクト
- インポート プロジェクト

注記 : プロジェクトは、作成後は別のプロジェクト タイプに変更できません。例外は I/O 配置プロジェクトのみで、このプロジェクトは RTL プロジェクトの基礎として使用することができます。

RTL プロジェクト

Vivado IDE を使用すると、RTL の作成からビットストリームの生成まで、デザイン フロー全体を管理できます。ユーザーは、RTL ソース ファイルのほか、デザイン ブロックには EDIF ネットリストを追加できるほか、IP を追加できます。IP には、Vivado ツールで生成された XCI ファイル、CORE Generator™ ツールで生成された XCO ファイル、およびコンパイル済みの NGC 形式の IP ネットリストを含めることができます。

RTL をエラボレートして解析し、構文が正しいことを確認したら、さまざまな合成やインプリメンテーション run を実行および管理し、デザインと実行結果を解析できます。また、さまざまな制約やインプリメンテーション ストラテジを試すこともできます。

合成後のプロジェクト

Xilinx® Synthesis Technology (XST) やサポートされているサードパーティの合成ツールを使用して Vivado IDE 環境外で合成されたデザインから、プロジェクトを作成することもできます。Vivado IDE には、EDIF、NGC、構造型 SystemVerilog、Verilog 形式のネットリストをインポートできます。ネットリストは、1 つのファイルにまとめられているか、複数のモジュールレベルのネットリストから構成される階層構造になっています。

ロジック ネットリストを解析し、さまざまなインプリメンテーション run を実行および管理し、デザインと実行結果を解析できます。また、さまざまな制約やインプリメンテーション ストラテジを試すこともできます。

注記 : エンベデッド タイミング制約を含む NGC または EDIF ファイルをインポートすると、その制約は使用されません。ザイリンクス® デザイン制約 (XDC) ファイルの詳細は、『Vivado Design Suite ユーザー ガイド : 制約の使用』(UG903) [参照 5] を参照してください。ユーザー制約ファイル (UCF) を XDC 制約に変換する方法については、Vivado Design Suite 移行手法ガイド (UG911) [参照 6] を参照してください。

I/O 配置プロジェクト

空の I/O 配置プロジェクトを作成すると、デザイン サイクルの初期段階で I/O 配置を実行できます。I/O ポートは Vivado IDE 内で作成できますが、CSV または XDC 入力ファイルのいずれかの形式でインポートすることもできます。I/O 配置プロジェクトを使用すると、別のデバイス アーキテクチャで使用可能なロジック リソースも確認できます。

I/O を割り当てた後、Vivado IDE で CSV、XDC、および RTL 出力ファイルを作成できます。このファイルは、RTL ソースまたはネットリストが使用可能になってから、デザイン フローの後の段階で使用します。この出力ファイルは、プリント回路基板 (PCB) デザイン プロセスで使用する回路図シンボルの作成にも使用できます。

注記 : I/O 配置プロジェクトは、RTL ベースのデザイン プロジェクトの基礎として使用できます。詳細は、『Vivado Design Suite ユーザー ガイド : I/O およびクロック配置』(UG899) [参照 7] を参照してください。


インポート プロジェクト

Synopsys Synplify、XST、または ISE® Design Suite プロジェクトからのデータは、Vivado ツールの RTL プロジェクトに移行できます。プロジェクトのソース ファイルおよびコンパイル順はインポートされますが、インプリメンテーション結果およびプロジェクト設定はインポートされません。

プロジェクトの作成

New Project ウィザードでは、プロジェクト名およびディレクトリの指定、プロジェクトへのソース ファイルと制約 ファイルの追加、ターゲット デバイスの選択をウィザードに従って実行できます。

1. Vivado IDE で [File] → [New Project] をクリックします。

注記 : または、[New Project] ツールバー ボタン  をクリックします。または、Getting Started ページで [Create a New Project] のリンクをクリックします。

2. New Project ウィザードで概要を確認し、[Next] をクリックします。
3. [Project Name] ページ (図 2-1) で次のオプションを設定し、[Next] をクリックします。

- [Project name] : プロジェクト名を指定します (例 : project_1)。
- [Project location] : 新しいプロジェクトのディレクトリを指定します。
- [Create Project Subdirectory] : プロジェクトと同じ名前の下位ディレクトリを指定したプロジェクト ディレクトリに追加します。

注記 : チェック ボックスはデフォルトではオンになっており、プロジェクト ファイル (.xpr) が <project_location>/<project_name> に作成されます。プロジェクトで作成されたすべてのフォルダーおよびデータ ファイルは <project_name> ディレクトリに保存されます。オフにすると、プロジェクト ファイル (.xpr) が <project_location> に作成され、そのプロジェクトで作成されたすべてのフォルダーおよびデータ ファイルがそのディレクトリに保存されます。

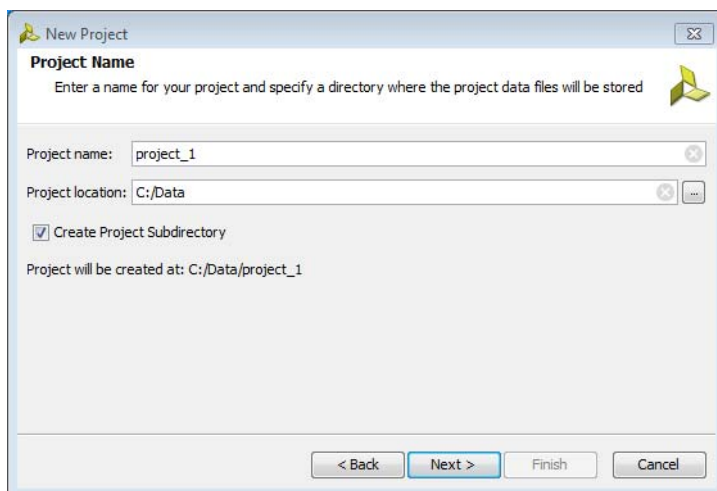


図 2-1 : New Project ウィザード : [Project Name] ページ

4. [Project Type] ページ (図 2-2) でプロジェクト タイプを指定し、プロジェクトに関連付けるソース ファイルのタイプを決めます。

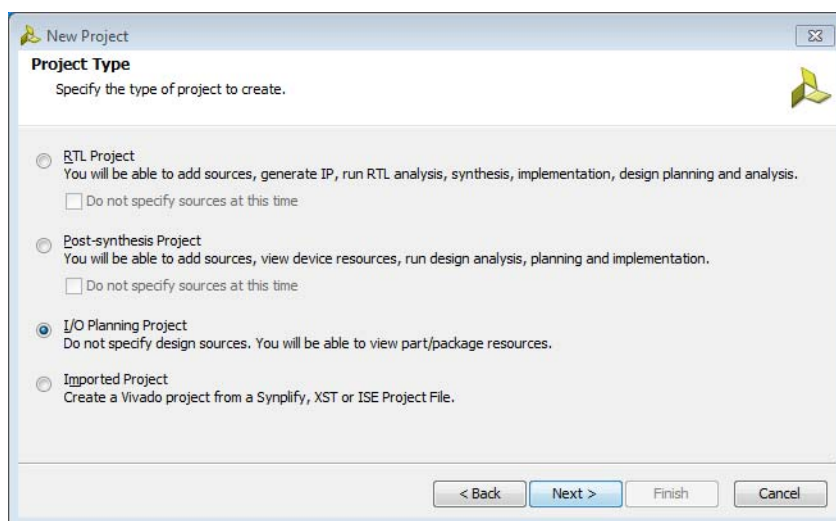


図 2-2 : New Project ウィザード : [Project Type] ページ

5. 作成するプロジェクト タイプに応じて、次のセクションのいずれかの説明を参照してください。ウィザードの残りのページで、プロジェクトに最適なソースを追加していきます。
- 。 [RTL プロジェクトの作成](#)
 - 。 [合成後プロジェクトの作成](#)
 - 。 [I/O ピン配置プロジェクトの作成](#)
 - 。 [外部プロジェクトのインポート](#)

RTL プロジェクトの作成

RTL ソース ファイルを指定してプロジェクトを作成すると、RTL ソース ファイルを合成およびインプリメンテーションだけでなく、RTL コード開発および解析でも使用できます。RTL の開発および解析の詳細は、[第 4 章「RTL デザインのエラボレーション」](#)を参照してください。



重要: デザインに XMP ファイルを追加する際は、パスにスペースを含めないようにしてください。XPS では現在のところパスにスペースを使用できません。

1. **プロジェクトの作成**の手順に従ってプロジェクトを作成します。
2. [Project Type] ページで [RTL Project] をオンにして、[Next] をクリックします。

注記: 必要であれば、[Do not specify sources at this time] をオンにします。これをオンにすると、デザイン ソースを追加する手順を飛ばして、ターゲット パーツを選択してプロジェクトを作成できます。
3. [Add Sources] ページ (図 2-3) で次のオプションを設定し、[Next] をクリックします。
 - [Add Files]: プロジェクトに追加するファイルを選択するためのファイル ブラウザーが表示されます。RTL プロジェクトには、HDL、EDIF、NGC、BMM、ELF およびその他のファイル タイプを追加できます。

注記: [Add Source Files] ダイアログ ボックスでは、各ファイルまたはディレクトリがそれとわかるようなアイコンで表示されます。小さい赤い四角は、読み出し専用であることを示します。
 - [Add Directories]: 選択したディレクトリに含まれるすべてのファイルを追加します。指定したディレクトリにある有効なソース ファイルがすべてプロジェクトに追加されます。
 - [Create File]: VHDL、Verilog、Verilog ヘッダー、または SystemVerilog ファイルを作成する [Create Source File] ダイアログ ボックスが開きます。[Create Source File] ダイアログ ボックスで次のようにオプションを設定します。
 - [File type]: Verilog ファイル (.v)、Verilog ヘッダー ファイル (.vh)、SystemVerilog ファイル (.sv)、VHDL ファイル (.vhd) などのファイル形式のいずれかを指定します。
 - [File name]: 新しい HDL ソース ファイルの名前を指定します。
 - [File location]: ファイルを作成するディレクトリを指定します。

注記: ファイルのプレースホルダーがソースのリストに追加されます。ファイルは [Finish] をクリックすると作成されます。
 - [Library]: ファイルまたはディレクトリの RTL ライブラリを指定します。ライブラリ名は選択するか、[Library] テキスト フィールドに新しいライブラリ名を入力して指定します。

注記: このオプションは、VHDL ファイルの場合のみ使用できます。デフォルトでは、HDL ソース ファイルは work ライブラリに追加されます。必要に応じて、ユーザー VHDL ライブラリを作成し、参照できます。Verilog および SystemVerilog ファイルの場合は、work のままにしておいてください。
 - [HDL Source for]: 読み込むソースが合成およびシミュレーション用の RTL ソース ファイルであるか、シミュレーションのみで使用する RTL テストベンチであるかを指定します。
 - [Delete]: 選択したソース ファイルを削除します。
 - [Move Selected File Up]: ファイルまたはディレクトリをリストの上方向に移動します。ファイル順は、合成やシミュレーションなどのダウンストリーム プロセスでのエラボレーションおよびコンパイルの順序に影響します。
 - [Move Selected File Down]: ファイルまたはディレクトリをリストの下方向に移動します。
 - [Scan and Add RTL Include Files into Project]: すべての RTL ファイルをスキャンし、参照された Verilog の 'include ファイルをローカル プロジェクト ディレクトリにインポートします。
 - [Copy Sources into Project]: 元のファイルを参照するのではなく、ローカル プロジェクト ディレクトリにファイルをコピーします。[Add Directories] ボタンをクリックしてソース ファイルのディレクトリを追加した場合は、ファイルがローカル プロジェクトにコピーされる際にディレクトリ構造もそのまま保持されます。詳細は、第 3 章「リモート ソースの参照またはプロジェクト ディレクトリへのソースのコピー」を参照してください。
 - [Add Sources from Subdirectories]: [Add Directories] で指定したディレクトリのサブディレクトリに含まれるソース ファイルをすべて追加します。
 - [Target Language]: Verilog または VHDL のいずれかにデザインのターゲット言語を指定します。新しい RTF ファイルはデフォルトで指定したターゲット言語になります。指定したターゲット言語でデザインの出力ファイルが生成されます。

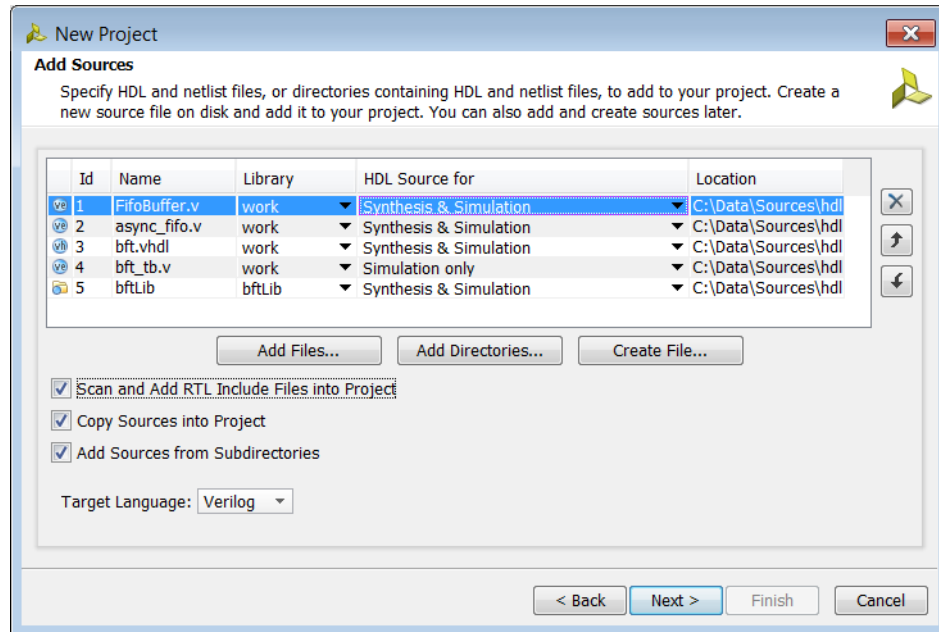


図 2-3 : New Project ウィザード : [Add Sources] ページ

4. オプション : [Add Existing IP] ページ (図 2-4) で次のオプションを設定し、[Next] をクリックします。

- [Add Files] : Vivado Design Suite の Xilinx Core Instance (XCI) ファイルまたは CORE Generator コア (XCO) ファイルのいずれかを選択するファイルブラウザーが開きます。

XCI ファイルは、IP の作成に使用されるプロジェクト オプション、カスタマイズ パラメーター、ポート パラメーターなどの値を記録する IP-XACT コンポーネント インスタンスの XML ファイルです。

注記 : Vivado IP カタログを使用して XCI IP を追加すると、Vivado IDE は自動的に HDL ソースなどの生成されたターゲットをすべてプロジェクトにインポートします。合成を実行すると、その IP と最上位デザインが一緒に合成されます。

- [Add Directories] : Vivado Design Suite の XCI ファイルまたは XCO ファイルのいずれかを含むディレクトリを選択するファイルブラウザーが開きます。ディレクトリを選択すると、そのディレクトリに含まれるすべての XCI および XCO ファイルが IP リストに追加されます。
- [Remove Selected Files and Directories] : X ボタンで表示され、選択したファイルおよびディレクトリを削除します。
- [Copy Sources into Project] : 元のファイルを参照するのではなく、ローカルプロジェクトディレクトリにファイルをコピーします。

注記 : サードパーティから合成済み NGC または EDIF ネットリストとして提供されている IP もあります。これらのファイルをデザインに読み込むには、[Add Sources] コマンドをクリックし、[Add or Create Design Sources] をオンにしてファイルを読み込みます。第 3 章「IP ソースの操作」に示すように、IP カタログを使用すると Vivado IDE 内のプロジェクトにパラメーター指定可能なコアを読み込むこともできます。

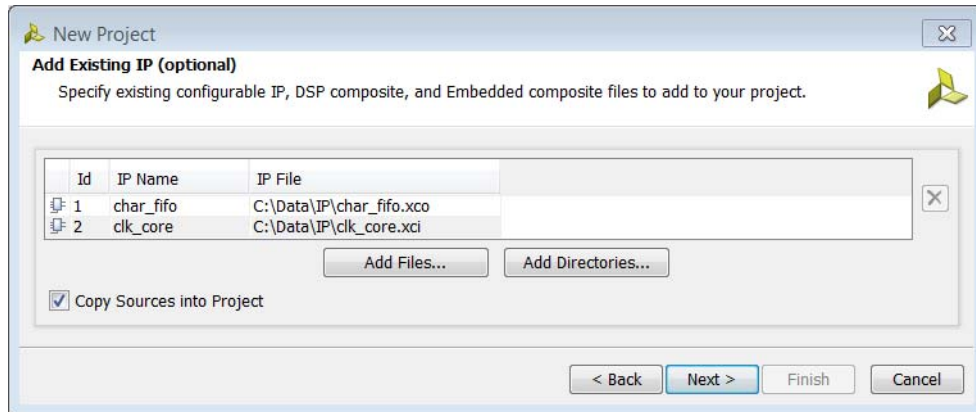


図 2-4 : New Project ウィザード : [Add Existing IP] ページ

5. オプション : [Add Constraints] ページ (図 2-5) で次のオプションを設定し、[Next] をクリックします。
- [Add Files] : プロジェクトに追加する Synopsys デザイン制約 (SDC) また XDC ファイルを指定するためのファイルブラウザーが開きます。
 - [Create File] : 新しい最上位の XDC ファイルが作成されます。
 - [Remove] : 制約リストから選択したファイルが削除されます。
 - [Up]/[Down] : 制約ファイルをリストの上下方向に移動します。コマンドはリストされる順序に依存し、制約の最後のコマンドがそれより前のコマンドの結果を上書きします。
 - [Copy Constraints into Project] : 元のファイルを参照するのではなく、ローカルプロジェクトディレクトリに制約ファイルをコピーします。

注記 : プロジェクトに関連付けられた RTL またはネットリスト ソースファイルと同じディレクトリの SDC または XDC ファイルは、プロジェクトに追加される制約ファイルとして自動的に表示されます。

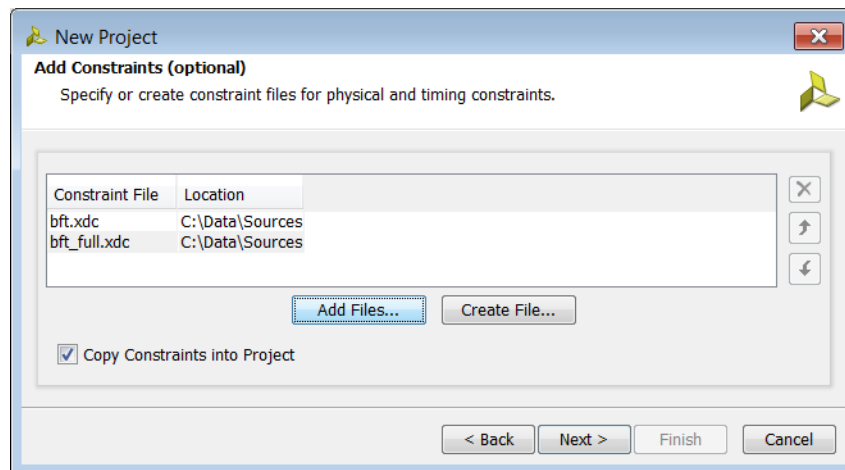


図 2-5 : New Project ウィザード : [Add Constraints] ページ

6. [Default Part] ページ (図 2-6) でザイリンクス パーツまたはターゲット デザイン プラットフォーム (TDP) ボードを選択し、[Next] をクリックします。
- [Parts] : 使用可能なデバイスがリストされます。デバイス リソースに関する情報が、表形式で表示されます。このリストでは、製品、ファミリー、サブファミリー、パッケージ、スピード グレード、および温度などのフィルターを使用して、デバイスを絞り込むことができます。

- [Boards] : 使用可能な TDP ボードと、そのボードで使用されるザイリンクス パーツがリストされます。I/O ピンのカウントやルックアップ テーブル (LUT) およびフリップフロップ (FF) の数、使用可能なブロック RAM などのデバイス リソースに関する情報が表形式で表示されます。リストは、ファミリー、パッケージ、スピード グレードでフィルターをかけて表示させることもできます。
- [Search] : 指定した検索条件に合うデバイスのみがリストできます。

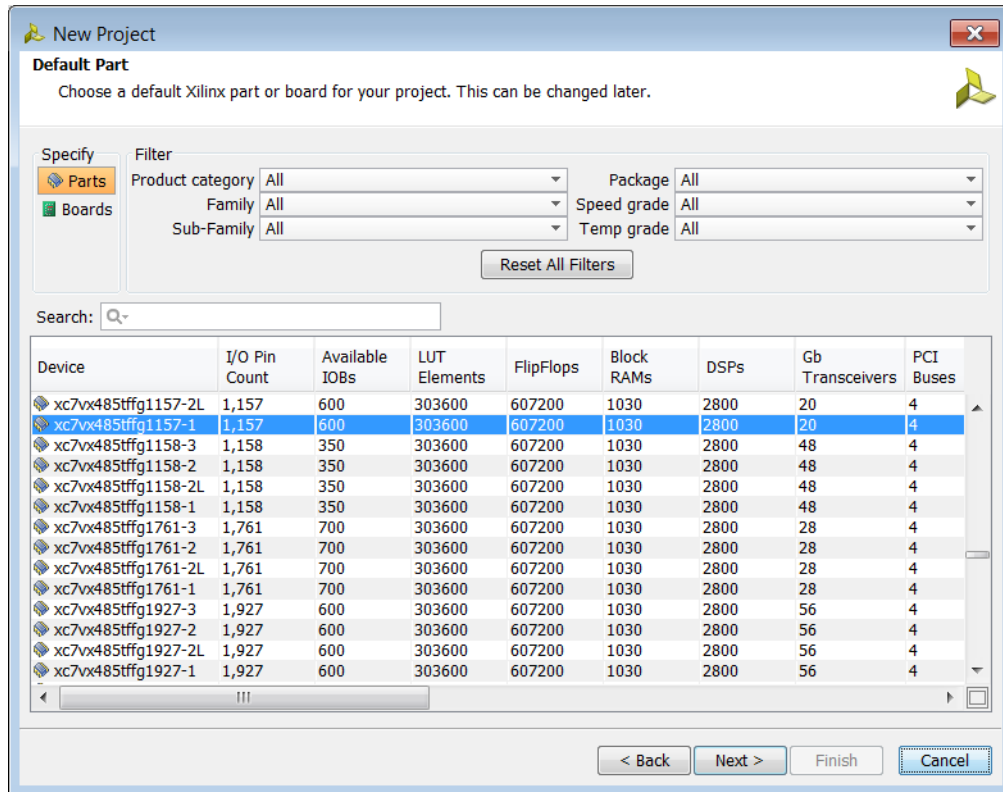


図 2-6 : New Project ウィザード : [Default Part] ページ

7. [New Project Summary] ページでプロジェクトに選択されたオプションを確認したら、[Finish] をクリックします。
8. オプション : **手順 3** で [Create File] オプションを使用すると、[Define Module] ダイアログ ボックス (図 2-7) が開きます。このダイアログ ボックスで次のオプションを使用して Verilog、Verilog ヘッダー、SystemVerilog、または VHDL でモジュールまたはアーキテクチャを定義したら、[OK] をクリックします。
 - [Entity name/Module name] : VHDL コードのエンティティまたは Verilog または SystemVerilog コードのモジュール名の名前を指定します。
注記 : エンティティまたはモジュール名はデフォルトでそのファイル名になりますが、別の名前を付けることもできます。
 - [Architecture name] : RTL ソース ファイルのアーキテクチャを指定します。デフォルトでは [Behavioral] です。
注記 : このオプションは、VHDL コードの場合にのみ表示され、Verilog または SystemVerilog モジュールを定義する場合には表示されません。
 - [I/O Port Definitions] : モジュール定義に追加するポートを定義します。
 - [Port Name] : RTL コードに記述されるポートの名前を定義します。
 - [Direction] : ポートを入力、出力、双方向のいずれかに指定します。
 - [Bus] : ポートがバス ポートかどうかを指定します。次の [MSB] および [LSB] オプションを使用してポートのバス幅を定義します。

- [MSB]: 最上位ビット (MSB) の数を定義します。[LSB] フィールドと組み合わせて、定義されるバスの幅を指定します。
- [LSB]: 最下位ビット (LSB) の数を定義します。

注記: ポートがバス ポートでない場合は、MSB および LSB は無視されます。

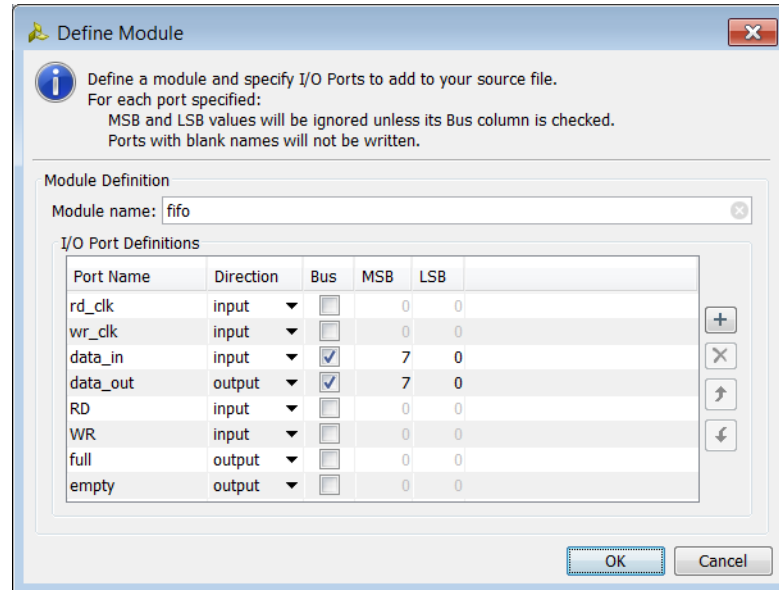


図 2-7: [Define Module] ダイアログ ボックス

RTL ソース ファイルが作成され、プロジェクトに追加されます。[Sources] ビューに新しく定義したモジュールがリストされます。Vivado IDE テキスト エディターで新しいソース ファイルを編集するには、ファイルをダブルクリックするか、[Open File] ポップアップ メニューをクリックします。新規作成したファイルの編集方法については、第 3 章「テキスト エディターの使用」を参照してください。

合成後プロジェクトの作成

合成後プロジェクトは、合成済みネットリストおよびその制約を使用して開始します。この後、デザインを解析、フロアプラン、インプリメントできます。

注記: 合成済みネットリストを作成するには、XST またはサードパーティの合成ツールを使用できます。

1. プロジェクトの作成の手順に従ってプロジェクトを作成します。
2. [Project Type] ページで [Post-Synthesis Project] をオンにして、[Next] をクリックします。

注記: 必要であれば、[Do not specify sources at this time] をオンにします。これをオンにすると、デザイン ソースを追加する手順を飛ばして、ターゲット パーツを選択してプロジェクトを作成できます。

3. [Add Netlist Sources] ページ (図 2-8) で次のオプションを使用して、読み込むネットリスト ファイルを指定し、最上位モジュールを含むファイルを識別し、下位レベル モジュールのネットリストを検索するためのディレクトリを定義したら、[Next] をクリックします。
 - 。 [Add Files]: プロジェクトに追加するネットリスト ファイル (Verilog、SystemVerilog、EDIF または NGC) を選択します。

注記: ファイルに最上位ネットリストが含まれる場合は [Top] をオンにします。

 - 。 [Add Directories]: ディレクトリ ブラウザーが起動され、モジュールを検索するディレクトリを選択できます。指定したディレクトリにある有効なソース ファイルがすべてプロジェクトに追加されます。

- [Remove Selected Files and Directories] : X ボタンで表示され、選択したファイルおよびディレクトリを削除します。
- [Move Selected Files and Directories Up] : 上向き矢印のアイコンは、ファイルまたはディレクトリをリストの上方向に移動します。
- [Move Selected Files and Directories Down] : 下向き矢印のアイコンは、ファイルまたはディレクトリをリストの下方向に移動します。
- [Copy Sources into Project] : 元のファイルを参照するのではなく、ローカルプロジェクトディレクトリにファイルをコピーします。[Add Directories] ボタンをクリックしてソース ファイルのディレクトリを追加した場合は、ファイルがローカルプロジェクトにコピーされる際にディレクトリ構造もそのまま保持されます。詳細は、第3章「リモート ソースの参照またはプロジェクト ディレクトリへのソースのコピー」を参照してください。
- [Add Sources from Subdirectories] : [Add Directories] で指定したディレクトリの下位ディレクトリに含まれるネットリスト ファイルを検索して追加します。

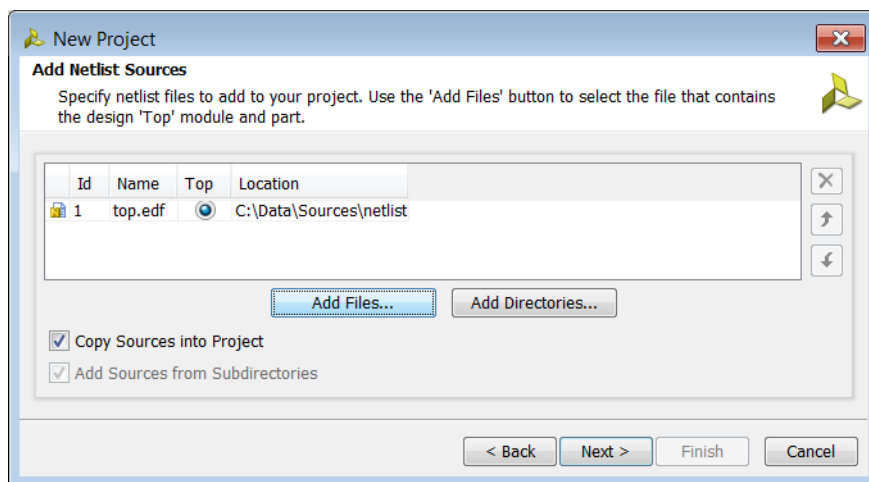


図 2-8 : New Project ウィザード : [Add Netlist Sources] ページ

4. オプション : [Add Constraints] ページ (図 2-5) で次のオプションを設定し、[Next] をクリックします。
 - [Add Files] : プロジェクトに追加する SDC または XDC ファイルを指定するためのファイル ブラウザーが開きます。
 - [Create File] : 新しい最上位の XDC ファイルが作成されます。
 - [Remove] : 制約リストから選択したファイルが削除されます。
 - [Up]/[Down] : 制約ファイルをリストの上下方向に移動します。コマンドはリストされる順序に依存し、制約の最後のコマンドがそれより前のコマンドの結果を上書きします。
 - [Copy Constraints into Project] : 元のファイルを参照するのではなく、ローカルプロジェクトディレクトリに制約ファイルをコピーします。

注記 : プロジェクトに関連付けられた RTL またはネットリスト ソース ファイルと同じディレクトリの SDC または XDC ファイルは、プロジェクトに追加される制約ファイルとして自動的に表示されます。

5. [Default Part] ページ (図 2-6) でザイリンクス パーツまたはターゲット デザイン プラットフォーム (TDP) ボードを選択し、[Next] をクリックします。
 - [Parts] : 使用可能なデバイスがリストされます。デバイス リソースに関する情報が、表形式で表示されます。このリストでは、製品、ファミリ、サブファミリ、パッケージ、スピード グレード、および温度などのフィルターを使用して、デバイスを絞り込むことができます。
 - [Boards] : 使用可能な TDP ボードと、そのボードで使用されるザイリンクス パーツがリストされます。I/O ピンのカウントや LUT およびフリップフロップの数、使用可能なブロック RAM などのデバイス リソース

に関する情報が表形式で表示されます。リストは、ファミリー、パッケージ、スピード グレードでフィルターをかけて表示させることもできます。

- 。 [Search] : 指定した検索条件に合うデバイスのみがリストできます。

6. [New Project Summary] ページでプロジェクトに選択されたオプションを確認したら、[Finish] をクリックします。

I/O ピン配置プロジェクトの作成

I/O ピン配置プロジェクトは、システム レベル デザインのデバイス ピン配置を指定するために使用します。このタイプのプロジェクトは、HDL または合成済みネットリストを完了する前に作成できます。たとえば、システム レベルまたは PCB 設計者とデザイン情報を共有する目的などに使用できます。I/O ピン配置の詳細は、『Vivado Design Suite ユーザー ガイド : I/O およびクロック配置』(UG899) [参照 7] を参照してください。

1. プロジェクトの作成の手順に従ってプロジェクトを作成します。
2. [Project Type] ページで [I/O Planning Project] をオンにして、[Next] をクリックします。
3. オプション : [Import Ports] ダイアログ ボックス (図 2-9) で次のオプションを指定し、I/O ポート定義および制約をインポートするためのファイルを選択したら、[Next] をクリックします。
 - 。 [Import CSV] : I/O 定義を含む CSV ファイルを選択します。CSV ファイルに関する詳細は、『Vivado Design Suite ユーザー ガイド : I/O およびクロックの配置』(UG899) [参照 7] を参照してください。
 - 。 [Import XDC] : I/O ポート関連の制約のみを含む XDC ファイルを選択します。
 - 。 [Do not import I/O ports at this time] : 空のプロジェクトを作成します。I/O は後ほど作成またはインポートできます。

注記 : RTL ヘッダーまたはソース ファイルを使用してデザインの I/O ピン配置を実行するには、RTL プロジェクトを使用します。

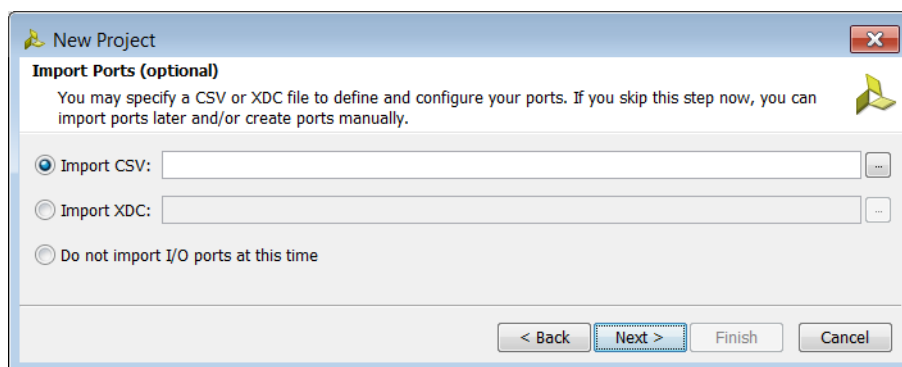


図 2-9 : New Project ウィザード : [Import Ports] ページ

4. [Default Part] ページ (図 2-6) でザイリンクス パーツまたはターゲット デザイン プラットフォーム (TDP) ボードを選択し、[Next] をクリックします。
 - 。 [Parts] : 使用可能なデバイスがリストされます。デバイス リソースに関する情報が、表形式で表示されます。このリストでは、製品、ファミリー、サブファミリー、パッケージ、スピード グレード、および温度などのフィルターを使用して、デバイスを絞り込むことができます。
 - 。 [Boards] : 使用可能な TDP ボードと、そのボードで使用されるザイリンクス パーツがリストされます。I/O ピンのカウントや LUT およびフリップフロップの数、使用可能なブロック RAM などのデバイス リソースに関する情報が表形式で表示されます。リストは、ファミリー、パッケージ、スピード グレードでフィルターをかけて表示させることもできます。
 - 。 [Search] : 指定した検索条件に合うデバイスのみがリストできます。
5. [New Project Summary] ページでプロジェクトを定義するために選択したオプションを確認したら、[Finish] をクリックします。

外部プロジェクトのインポート

Vivado IDE 以外 (例 : Synopsys の Synplify、XST、または ISE Design Suite の Project Navigator) で作成した既存の RTL レベルのプロジェクト ファイルをインポートできます。Vivado IDE では、指定したプロジェクトのソース ファイルが検出され、新規プロジェクトへ自動的に追加されます。最上位モジュール、ターゲット デバイス、VHDL ライブラリなどの設定も既存プロジェクトからインポートされます。

注記 : XST または ISE Design Suite プロジェクトのインポート方法については、Vivado Design Suite 移行手法ガイド (UG911) [参照 6] を参照してください。

1. [プロジェクトの作成](#)の手順に従ってプロジェクトを作成します。
2. [Project Type] ページで [Imported Project] をオンにして、[Next] をクリックします。
3. [Import Project] ページ (図 2-10) で次のオプションを使用して、インポートするプロジェクトファイルを指定したら、[Next] をクリックします。
 - [ISE] : 指定したザイリンクス ISE Design Suite プロジェクト ファイル (拡張子は.xise) をインポートします。
 - Synplify : 指定した Synplify プロジェクト ファイル (拡張子は .prj) をインポートします。
 - XST : 指定した XST プロジェクト ファイル (拡張子は .xst) をインポートします。
 - [Copy Sources into Project] : 元のファイルを参照するのではなく、ローカルプロジェクト ディレクトリにファイルをコピーします。

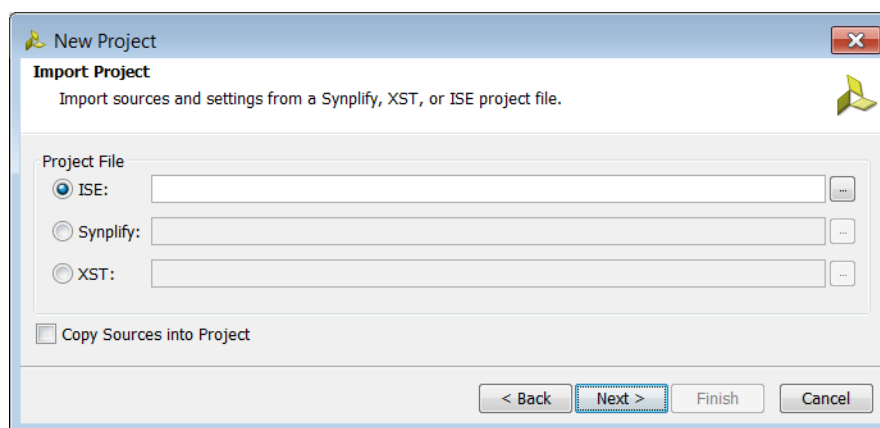


図 2-10 : New Project ウィザード : [Import Project] ページ

4. [New Project Summary] ページでプロジェクトを定義するオプションを確認したら、[Finish] をクリックします。

指定したプロジェクトから RTL ソース ファイル、制約ファイルがインポートされ、指定したディレクトリにプロジェクト ファイルが作成されます。インポート プロセスのサマリがインポート サマリ レポートのログ ファイルに記述され、新規プロジェクト ディレクトリに保存されます。このサマリ ファイルでは、プロジェクトを作成する際に使用された手順およびエラーや警告メッセージを確認できます。

プロジェクトを作成する Tcl コマンド

次の Tcl コマンドは、プロジェクトの作成に使用できます。スクリプト例については、[Tcl スクリプトを使用したプロジェクトの作成](#)を参照してください。

注記 : Tcl コマンドの詳細については、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) [参照 9] を参照するか、<command> -help を入力してください。

プロジェクトを作成する Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド:** `create_project` および `set_property`
- **Tcl コマンドの例 (RTL プロジェクト):** `create_project my_project C:/team/designs/my_project -part xc7k325tffg676-2`
- **Tcl コマンドの例 (合成後プロジェクト):**

```
create_project my_IO_project C:/team/designs/my_IO_project -part xc7k325tffg676-2
set_property design_mode GateLvl [current_fileset]
```
- **Tcl コマンドの例 (I/O 配置プロジェクト):**

```
create_file project_io C:/projects/project_io -part xc7vx485tffg1157-1
set_property design_mode PinPlanning [current_fileset]
```

プロジェクトをインポートする Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド:** `create_project` および `import_xise`
- **Tcl コマンドの例:**

```
create_project project_import_ise C:/projects/project_import_ise
import_xise C:/projects/old/wave_gen_vhd_s6/wave_gen_vhd_s6.xise -copy_sources
```

デザイン ソース、制約ファイル、シミュレーション ソースを追加する Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド:** `add_files` または `import_files`
- **Tcl コマンドの例:**

```
add_files top.v
import_files -fileset constrs_1 C:/projects/sources/timing.xdc
```

注記: `add_files` コマンドで、現在のディレクトリから追加するファイルを参照し、`import_files` コマンドでプロジェクトにファイルをコピーします。

既存の IP ソース ファイルを追加する Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド:** `add_files` または `import_ip`
- **Tcl コマンドの例:** `import_ip C:/projects/sources/char_fifo/char_fifo.xci`

注記: `add_files` コマンドで、現在のディレクトリから XCI ファイルおよび関連する出力ファイルを参照し、`import_ip` コマンドで、プロジェクトにその XCI ファイルおよび関連する出力ファイルをコピーします。

プロジェクト パーツを設定する Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド:** `create_project` または `set_property`
- **Tcl コマンドの例:**

```
create_project my_project C:/projects/my_project -part xc7k325tffg676-2  
  
set_property PART xc7k70tffg676-2 [current_project]
```


注記: パーツは、プロジェクトの作成時もしくは作成後に設定できます。

プロジェクトの管理

プロジェクトを開く

プロジェクトを開くと、前回プロジェクトを開じたときの状態が復元されます。プロジェクトの状態とは、ソース ファイル順、ソース ファイルのディスエーブル/イネーブル、アクティブおよびターゲット制約ファイル、合成、シミュレーション、インプリメンテーション `run` のステートなどです。

プロジェクトを開くには、次のいずれかを実行します。

- Getting Started ページの [Open Project] リンクをクリックします。
- [File] → [Open Project] をクリックします。
- [Open Project] ツールバー ボタン  をクリックします。
- Tcl コンソールに `open_project` コマンドを入力します。

[Open Project] ダイアログ ボックスで、プロジェクト ファイル (.xpr) を選択します。[Open Project] ダイアログ ボックスの [File Preview] に現在選択されているファイルに関する情報が表示されます。

注記: または、Windows エクスプローラーで Vivado IDE プロジェクト ファイル (.xpr) を直接ダブルクリックしてプロジェクトを開きます。

プロジェクトを開く Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド:** `open_project`
- **Tcl コマンドの例:** `open_project c:/projects/project1.xpr`

複数のプロジェクトを開く

1 つのセッションで複数のプロジェクトを開くには、プロジェクトが開いている状態で**プロジェクトを開く**のいずれかの方法を使用して別のプロジェクトを開きます。Vivado IDE で現在のプロジェクトを閉じるかどうか尋ねるメッセージが表示されます。[No] をクリックして開いているプロジェクトを閉じないようにすると、両方のプロジェクトが開きます。各プロジェクトに対してそれぞれメイン ウィンドウが開きます。

複数プロジェクトを同じ Vivado IDE プロセスから開くと、開いているすべてのプロジェクトで使用されたコマンドが Tcl コンソールへ書き込まれます。ただし、表示されるコマンドがどのプロジェクトで使用されたものかわかりにくいことがあります。また、すべてのプロジェクトに対して 1 つの `vivado.jou` と 1 つの `vivado.log` しか作成されません。

注記: 複数のプロジェクトを開いた場合、システム メモリの要件により、パフォーマンスが低下することがあります。

プロジェクトの保存

プロジェクトは自動的に保存されます。たとえば、ソース設定、ファイルのプロパティ、run オプションなど、プロジェクトに変更を加えると、プロジェクトはディスクに自動的に保存されます。

別のディレクトリにプロジェクトを保存するには、[File] → [Save Project As] をクリックしてください。これにより、プロジェクト ディレクトリ構造全体が新しく指定されたディレクトリにコピーされ、run のステータスも保持されます。

プロジェクトを保存する Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド:** `save_project_as`
- **Tcl コマンドの例:** `save_project_as new_project c:/projects/`

プロジェクトを閉じる

プロジェクトを閉じるには、[File] → [Close Project] をクリックします。保存されていない変更がある場合は、それを示すメッセージが表示されます。

プロジェクトを閉じる Tcl コマンド

次は、関連する Tcl コマンドです。 `close_project`

プロジェクトのアーカイブ

プロジェクト アーカイブを作成して、バックアップとして保存したり、リモート サイトに送信したりできます。プロジェクトをアーカイブする際、Vivado IDE では次が実行されます。

- デザイン階層を解析します。
- ライブラリ ディレクトリから必要なソース ファイル、インクルード ファイル、リモート ファイルをコピーします。
- 制約をコピーします。
- 合成、シミュレーション、およびインプリメンテーションの実行結果をコピーします (オプション)。
- プロジェクトの ZIP ファイルを作成します。

プロジェクトのアーカイブを作成するには、次の手順に従います。

1. [File] → [Archive Project] をクリックします。
2. [Archive Project] ダイアログ ボックス (図 2-11) で次のオプションを設定し、[Next] をクリックします。
 - [Archive name] : プロジェクト アーカイブ名を指定します。
 - [Archive location] : プロジェクト アーカイブ ファイルを保存するディレクトリを指定します。
 - [Include Run Results] : プロジェクトで実行した run の設定と結果を含めます。

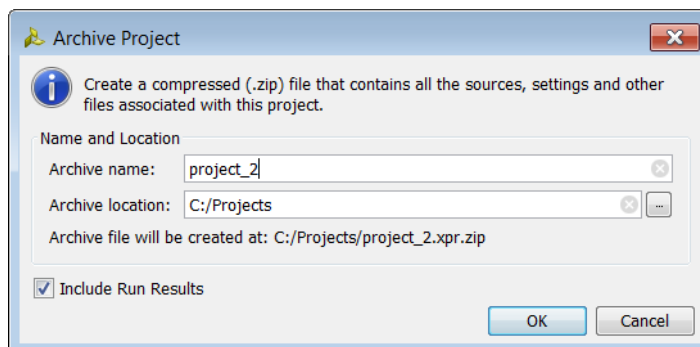


図 2-11 : [Archive Project] ダイアログ ボックス

Vivado IDE でプロジェクト アーカイブが ZIP ファイル形式で作成されます。この ZIP ファイルには、必要なソース ファイル、インクルード ファイル、run ファイル (指定した場合のみ)、およびアーカイブ プロセスを記述した archive.log ファイルが含まれます。archive.log ファイルでアーカイブの作成プロセスを確認できます。

プロジェクトを圧縮する Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド** : archive_project
- **Tcl コマンドの例** : archive_project -exclude_run_results proj3.zip

プロジェクト サマリの使用

Vivado IDE には、対話型のプロジェクト サマリ (図 2-12) が含まれ、デザイン コマンドが実行され、デザイン プロセスが実行されると随時アップデートされます。これには、プロジェクト パーツ、プロジェクト ステータス、合成およびインプリメンテーションの状態などのプロジェクトおよびデザイン情報が含まれます。また、メッセージ、ログ、レポートなどを示すビューや [Project Settings] ダイアログ ボックスへのリンクなど、詳細な情報へのリンクも含まれます。スクロール バーを使用したり、[Collapse All] および [Expand All] ボタンを使用してデータ カテゴリの表示/非表示を切り替えることができます。

プロジェクト サマリには、次のセクションが含まれます。


- [Project Settings] : プロジェクト名、プロジェクト パーツ、デフォルト デバイス、およびトップ モジュール名が表示されます。
- [Messages] :
 - [Summary] : デザイン プロセス中に発生したエラーおよび警告の数を示します。警告またはエラーのみを表示した [Messages] ビューを開くリンクもあります。
 - [Go To] : [Messages]、[Log]、[Reports] ビューへのリンクが提供されます。これらのビューの詳細については、『Vivado Design Suite ユーザー ガイド : Vivado IDE の使用』(UG893) [参照 4] を参照してください。
- [Synthesis] および [Implementation] : アクティブ run の合成およびインプリメンテーションの状態のサマリを表示します。ターゲット パーツ、run で使用されたストラテジ、使用されたツールフローおよび制約セットなどが表示されます。

パーツ、ストラテジ、フローのリンクをクリックすると、[Project Settings] ダイアログ ボックスが開きます。制約リンクをクリックすると、[Source File Properties] ビューに [Constraint Set Properties] タブが表示されます。

注記 : 詳細は、プロジェクト設定および第 3 章「制約の操作」を参照してください。

- [DRC Violations] : インプリメンテーション後にデザイン ルール チェック (DRC) に関する情報がまとめられます。
- [Timing] : インプリメンテーション後にタイミング結果がまとめられます。
- [Utilization] : 合成後にプロジェクトの使用量結果が表形式およびグラフ形式でまとめられます。
- [Power] : インプリメンテーション後に消費電力解析の結果がまとめられます。

プロジェクト サマリを開くには、次のいずれかを実行します。

- [Window] → [Project Summary] をクリックします。
- ツールバーの [Project Summary] ボタン  をクリックします。

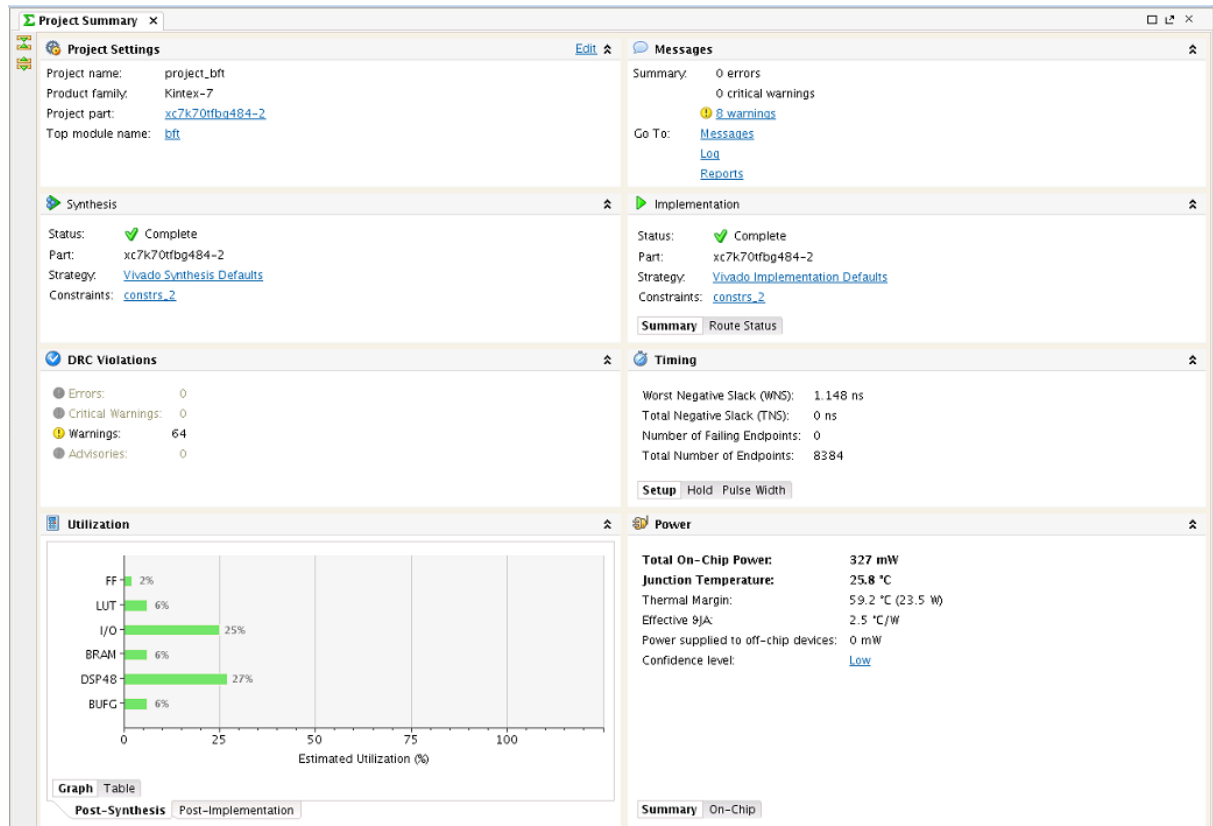



図 2-12 : [Project Summary] ビュー

プロジェクト設定

プロジェクト設定は、各プロジェクトの必要に合わせて指定できます。プロジェクト設定には、最上位モジュールの定義に関する一般的な設定、およびシミュレーション、合成、インプリメンテーション、IP などの設定が含まれます。

[Project Settings] ダイアログ ボックスを表示するには、次のいずれかを実行します。

- [Tools] → [Project Settings] をクリックします。
- ツールバーの [Project Settings] ボタン  をクリックします。
- Flow Navigator で [Project Settings] をクリックするか、[Simulation Settings]、[Synthesis Settings]、[Implementation Settings]、[Bitstream Settings] のいずれかをクリックします。
- プロジェクト サマリで [Project Settings] ヘッダーの横にある Edit リンクをクリックするか、[Synthesis] または [Implementation] セクションのいずれかでストラテジまたはフローをクリックします。

[Project Settings] ダイアログ ボックスの開き方によって、最適なカテゴリがデフォルトで表示されるようになっていきます。たとえば、Flow Navigator で [Simulation Settings] をクリックした場合、[Project Settings] ダイアログ ボックスには [Simulation] カテゴリが表示されます。次のセクションは、各カテゴリの詳細を示しています。

[General] ページ

[General] ページ (図 2-13) では、プロジェクト名、パーツ、ターゲット言語、ターゲット シミュレータ、最上位モジュール名、言語オプションを指定できます。

- [Name] : プロジェクト名を指定します。
- [Project Device] : 合成およびインプリメンテーション両方でデフォルトとして使用するターゲット デバイスを指定します。参照ボタンをクリックすると [Select Device] ダイアログ ボックスが表示され、デバイスを変更できます。

注記 : 合成 run またはインプリメンテーション run が複数ある場合は、[Run Properties] ビューから run 設定を変更して特定 run で使用されるデバイスを変更することもできます。[Runs Properties] ビューの使用方法の詳細については、『Vivado Design Suite ユーザー ガイド : Vivado IDE の使用』(UG893) [参照 4] を参照してください。

- [Target Language] : Verilog または VHDL のいずれかにデザインのターゲット出力言語を指定します。指定したターゲット言語でデザインの RTL 出力ファイルが生成されます。ターゲット言語で制御される出力の例は、合成、シミュレーション、最上位ラッパー ファイル、テストベンチ、および IP のインスタンス化テンプレートです。
- [Top Module Name] : デザインの最上位 RTL モジュール名を指定します。下位モジュール名を入力し、特定のモジュールに対して合成を実行することもできます。参照ボタンをクリックすると、最上位モジュールが自動的に検索され、可能性のある最上位モジュールのリストが表示されます。
- [Language Options] : 参照ボタンをクリックし、[Language Options] ダイアログ ボックスで次のオプションを設定します。
 - [Verilog Options] : 参照ボタンをクリックし、[Verilog Options] ダイアログ ボックスで次のオプションを設定します。
 - [Verilog Include Files Search Paths] : Verilog ソース ファイルの 'include' 文で参照されるファイルの検索パスを指定します。
 - [Defines] : プロジェクトの Verilog マクロ定義を指定します。
 - [Uppercase all identifiers] : すべての Verilog 識別子を大文字に設定します。
 - [Generics/Parameters] : VHDL ではジェネリックが、Verilog では定数値のパラメーター定義がサポートされます。どちらの方法でも、パラメーターを変更できるので、さまざまな状況で再利用可能です。参照ボタンをクリックすると、ジェネリックおよびパラメーターの値を定義して、ソース ファイルで定義されたデフォルトの値を上書きできます。

- [Top Library] : 最上位モジュールのライブラリ名を指定します。
- [Loop Count] : 最大ループ反復値を指定します。デフォルト値は 1000 です。

注記 : [Loop Count] オプションは、合成中ではなくエラボレーション中に使用されます。合成に対しては、[Synthesis] ページの [More Options] フィールドに 「-loop_iteration_limit」 と入力します。

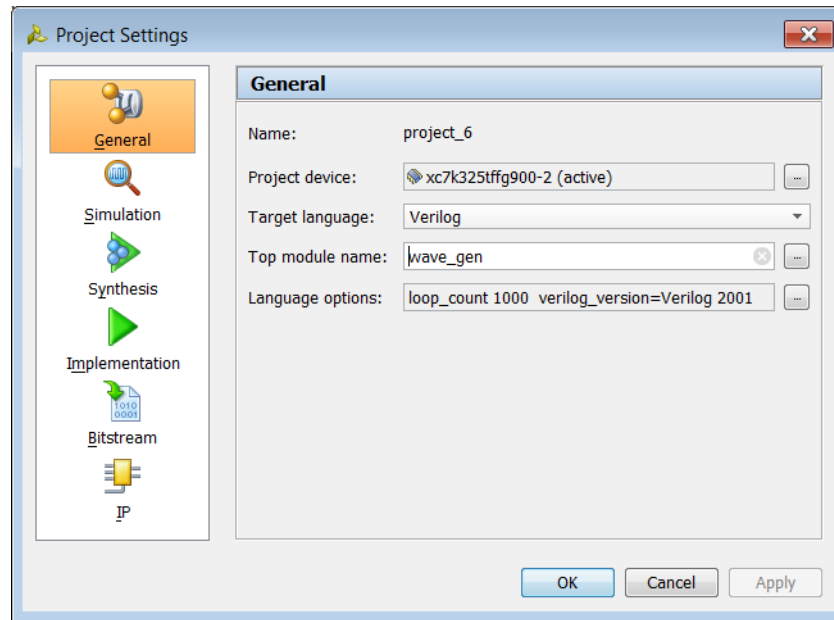


図 2-13 : [General] ページ

[Simulation] ページ

[Simulation] ページ (図 2-14) では、シミュレーション セット、シミュレーション最上位モジュール名、コンパイルおよびシミュレーション オプションを指定できます。オプションをクリックすると、ダイアログ ボックスの一番下にその説明が表示されます。[Simulation] ページの詳細は、『Vivado Design Suite ユーザー ガイド：ロジック シミュレーション』(UG900) [参照 8] を参照してください。

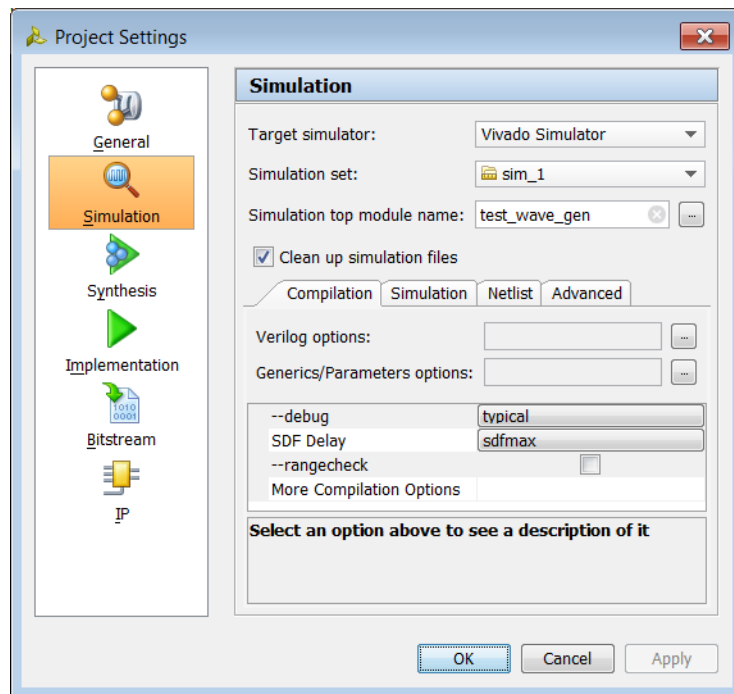


図 2-14 : [Simulation] ページ

[Project Settings] ダイアログ ボックスの [Synthesis] ページ

[Synthesis] ページ (図 2-15) では、制約セット、合成ストラテジおよび合成オプションを指定できます。オプションは選択した合成ストラテジで定義されますが、これらは変更できます。オプションをクリックすると、ダイアログ ボックスの一番下にその説明が表示されます。[Synthesis] ページの詳細は、『Vivado Design Suite ユーザー ガイド：合成』(UG901) [参照 10] を参照してください。



ヒント : tcl.pre および tcl.post ファイルを使用すると、Tcl スクリプトを追加して合成前後に読み出されるようになります。詳細は、Vivado Design Suite User Guide: Using Tcl Scripting (UG894) [参照 2] を参照してください。

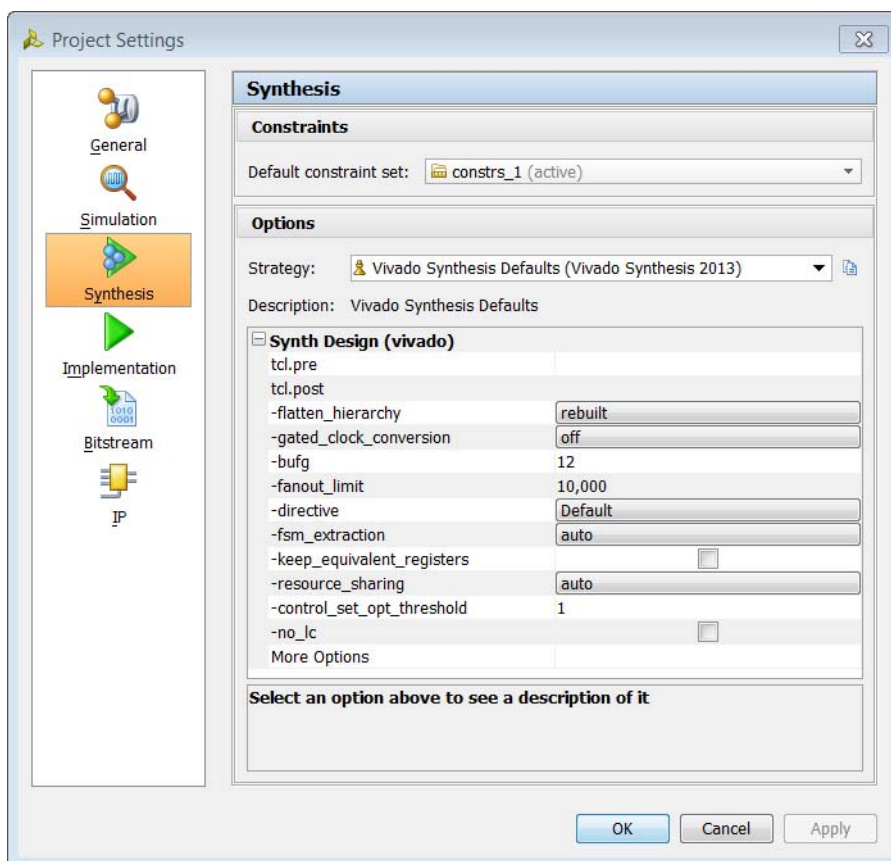


図 2-15 : [Synthesis] ページ

[Project Settings] ダイアログ ボックスの [Implementation] ページ

[Implementation] ページ (図 2-16) では、制約セット、インプリメンテーション ストラテジおよびインプリメンテーション オプションを指定できます。オプションは選択したインプリメンテーション ストラテジで定義されますが、これらは変更できます。たとえば、消費電力および物理合成などのオプションの段階を実行するオプションを使用可能です。オプションをクリックすると、ダイアログ ボックスの一番下にその説明が表示されます。[Implementation] ページの詳細は、『Vivado Design Suite ユーザー ガイド：インプリメンテーション』(UG904) [参照 11] を参照してください。



ヒント : Tcl スクリプトを追加すると、tcl.pre および tcl.post ファイルでインプリメンテーションのどの段階の前後にでも source コマンドで読み出すことができます。詳細は、Vivado Design Suite User Guide: Using Tcl Scripting (UG894) [参照 2] を参照してください。

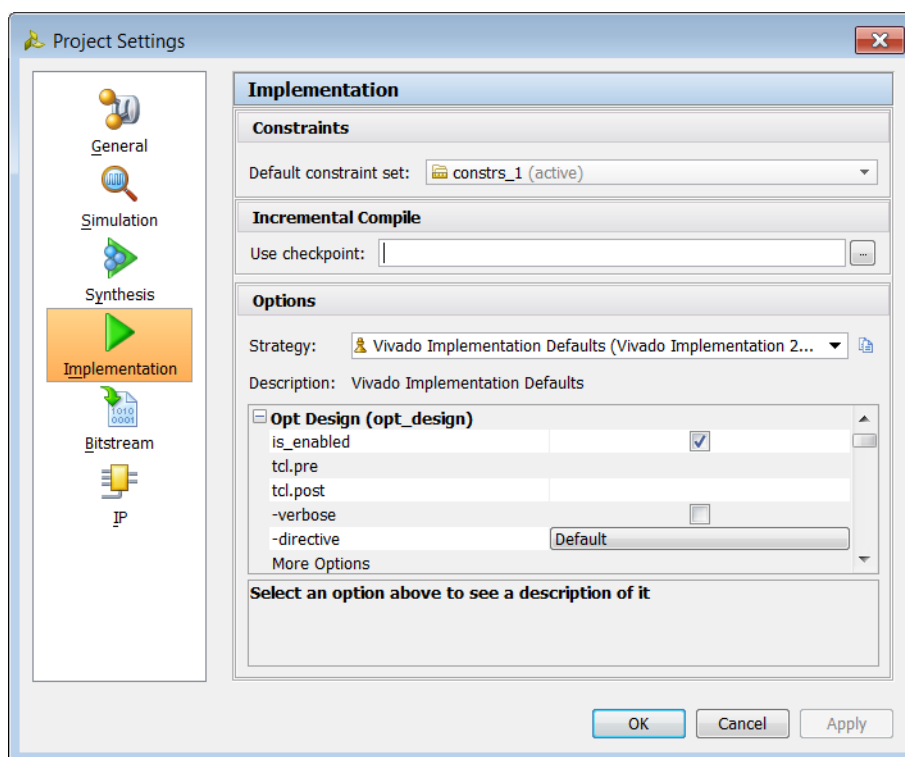


図 2-16 : [Implementation] ページ

[Bitstream] ページ

[Bitstream] ページ (図 2-17) では、ビットストリームを生成する前にオプションを定義できます。オプションをクリックすると、ダイアログ ボックスの一番下にその説明が表示されます。[Bitstream] ページの詳細は、『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』(UG908) [参照 12] を参照してください。

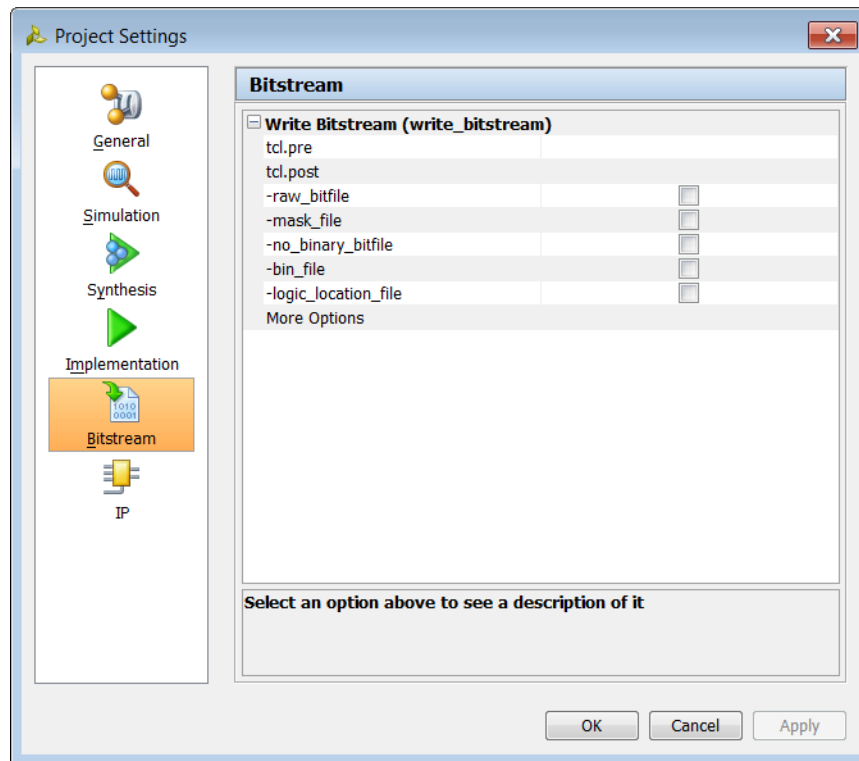


図 2-17 : [Bitstream] ページ

[IP] ページ

[IP] ページ (図 2-18) には、次のタブが含まれます。

- **[Repository Manager]** : IP レポジトリ リストに追加するディレクトリを指定します。IP は、ユーザーがパッケージできるほか、サードパーティ サプライヤーから入手することもできます。[Apply] をクリックすると、各レポジトリに IP が表示されるようになります。
- **[Generation]** : デフォルトで生成される IP 出力ファイルを設定します。
- **[Packager]** : ベンダー、ライブラリ、命名規則を含めて新しい IP をパッケージにする際のデフォルト値を設定します。このタブでは、IP パッケージャーを開いたときのデフォルト ビヘイビアを設定したり、自動的にフィルターされるファイル拡張子を指定できたりします。

注記 : 必要であれば、IP パッケージ プロセス中に IP をパッケージにする際のデフォルト値は変更できます。

[IP] ページの詳細は、『Vivado Design Suite ユーザー ガイド : IP を使用した設計』(UG896) [参照 13] を参照してください。

注記 : [IP] ページおよび Vivado IP カタログは、RTL プロジェクトまたは Getting Started ページから [Manage IP] リンクを使用した場合にのみ使用可能です。[Manage IP] を使用する場合、プロジェクトを作成しないと、[IP] ページのサブセットは表示されません。

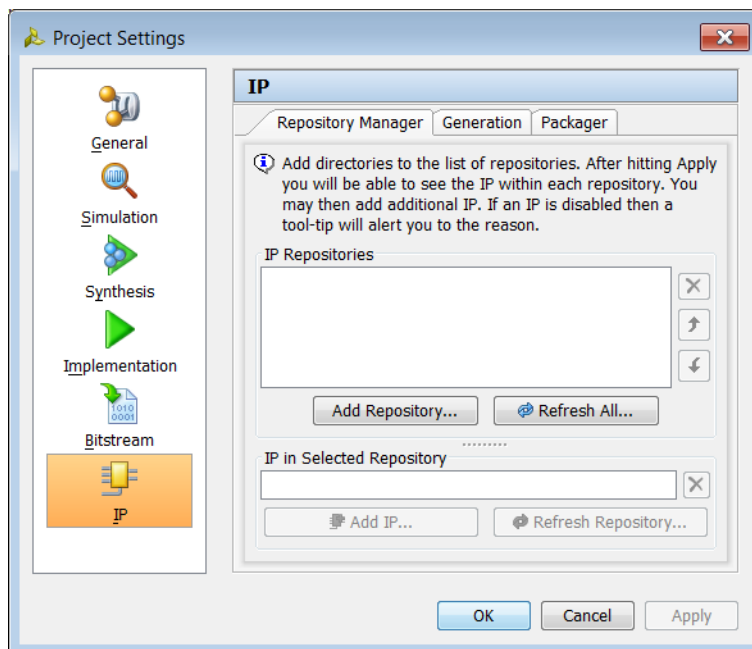


図 2-18 : [IP] ページ

プロジェクト設定をする Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド**: `set_property`
- **Tcl コマンドの例**: `set_property target_language Verilog [current_project]`



推奨: プロジェクト、合成またはインプリメンテーション `run` のプロパティを含む複数のプロパティを設定できます。プロパティ名およびターゲットの詳細については、Vivado IDE で操作を実行し、Tcl コンソールに表示される対応する Tcl コマンドを参照してください。

Tcl スクリプトを使用したプロジェクトの作成

Vivado IDE では、Tcl スクリプトを使用してプロジェクトを作成することもできます。Vivado IDE で実行したほとんどの動作が Tcl コマンドで実行されます。Vivado IDE の Tcl コンソールに表示される Tcl コマンドは、`vivado.jou` および `vivado.log` ファイルに保存されます。`vivado.jou` ファイルにはコマンドだけが、`vivado.log` ファイルにはコマンドと返されたメッセージがすべて含まれます。これらのファイルを使用して、スクリプトを開発し、プロジェクト モードでします。Tcl コマンドの詳細については、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) [参照 9] を参照してください。

次は、プロジェクトを作成し、さまざまなソースを追加し、設定をコンフィギュレーションし、合成およびインプリメンテーション `run` を開始し、ビットストリームを作成するスクリプト例です。

```
# Typical usage: vivado -mode tcl -source run_bft_project.tcl
# Create the project and directory structure
create_project -force project_bft_batch ./project_bft_batch -part xc7k70tfbg484-2
#
# Add various sources to the project
add_files {./Sources/hdl/FifoBuffer.v ./Sources/hdl/async_fifo.v \
./Sources/hdl/bft.vhdl}
add_files -fileset sim_1 ./Sources/hdl/bft_tb.v
add_files ./Sources/hdl/bftLib/
add_files -fileset constrs_1 ./Sources/bft_full.xdc
#
# Now import/copy the files into the project
import_files -force
#
# Set VHDL library property on some files
set_property library bftLib [get_files {*round*.vhdl core_transform.vhdl \
bft_package.vhdl}]
#
# Update to set top and file compile order
update_compile_order -fileset sources_1
update_compile_order -fileset sim_1
#
# Launch Synthesis
launch_runs synth_1
wait_on_run synth_1
open_run synth_1 -name netlist_1
#
# Generate a timing and power reports and write to disk
# Can create custom reports as required
report_timing_summary -delay_type max -report_unconstrained -check_timing_verbose \
-max_paths 10 -input_pins -file syn_timing.rpt
```

```
report_power -file syn_power.rpt
#
# Launch Implementation
launch_runs impl_1 -to_step write_bitstream
wait_on_run impl_1
#
# Generate a timing and power reports and write to disk
# comment out the open_run for batch mode
open_run impl_1
report_timing_summary -delay_type min_max -report_unconstrained \
-check_timing_verbose -max_paths 10 -input_pins -file imp_timing.rpt
report_power -file imp_power.rpt
#
# Can open the graphical environment if visualization desired
# comment out the for batch mode
#start_gui
```



ヒント :Tcl スクリプトで改行する場合は、バックスラッシュ文字 (\) を使用します。バックスラッシュの後の行は、前の行の一部として処理されます。

ソース ファイルの操作

概要

ソース ファイルにはプロジェクト ソース、デザイン ソース、制約ソース、シミュレーション ソース、IP ソース、デジタル信号処理 (DSP) ソース、エンベデッド ソース、IP サブシステムなどが含まれます。プロジェクト モードの場合は、Vivado™ IDE または Tcl コマンドカスクリプトを使用してこれらのソース ファイルを作成できます。Vivado IDE では、自動的にソース ファイルが管理されます。非プロジェクト モードの場合、これらのソース ファイルは Tcl コマンドカスクリプトを使用して作成できますが、ソース ファイルは手動で管理する必要があります。本章では、プロジェクト モードでのソースの作成と管理、非プロジェクト モードでのソースの作成について説明します。

プロジェクト モードでのソースの操作

Vivado IDE では、プロジェクトに対してローカルにあるファイルか、リモートにあってライブラリから参照しているソース ファイルを作成および管理できます。Verilog および VHDL ソース ファイルは、デザイン フローのどの段階でもプロジェクトに追加できます。また、制約ファイル、シミュレーション ソース、DSP ソース、エンベデッド ソースも作成したり、デザインに追加したりできるほか、既存 IP を追加したりできます。

注記：ソースの追加に関する Tcl コマンドの詳細は、第 2 章「デザイン ソース、制約ファイル、シミュレーション ソースを追加する Tcl コマンド」を参照してください。Tcl コマンドの詳細については、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) [参照 9] を参照するか、<command> -help を入力してください。

デザイン ソースの操作

Vivado IDE では、HDL またはネットリスト ファイルを含めてデザイン ソース ファイルを作成および管理できます。

デザイン ソースの作成

1. [File] → [Add Sources] をクリックします。

注記：または、Flow Navigator で [Add Sources] をクリックするか、[Sources] ビューのポップアップ メニューから [Add Sources] をクリックします。

2. Add Sources ウィザード (図 3-1) で [Add or Create Design Sources] をオンにし、[Next] をクリックします。

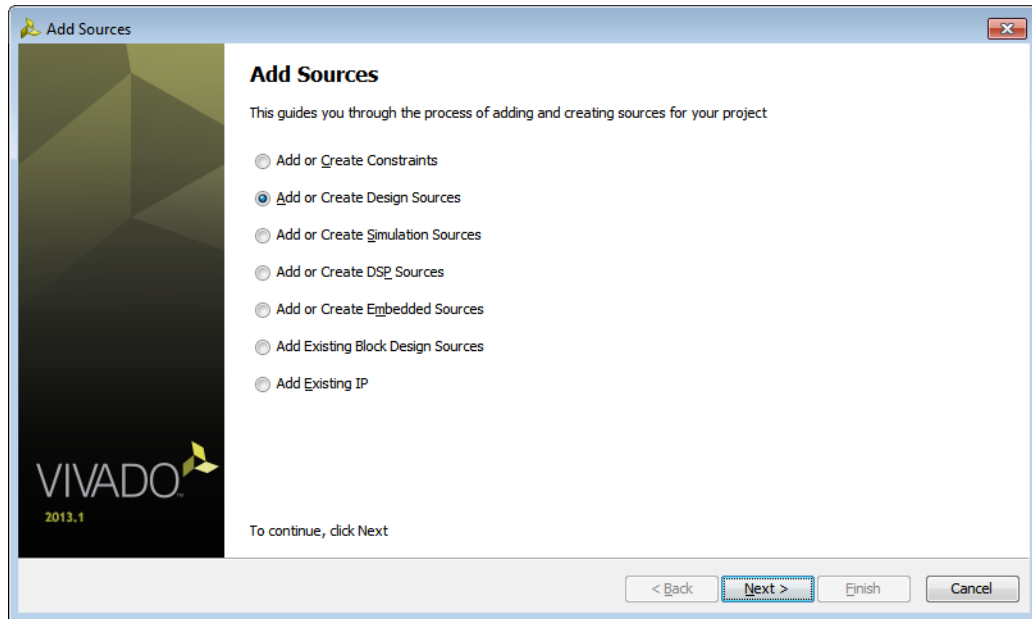


図 3-1 : Add Sources ウィザード

3. [Add or Create Design Sources] ページ (図 3-2) で [Create File] をクリックします。

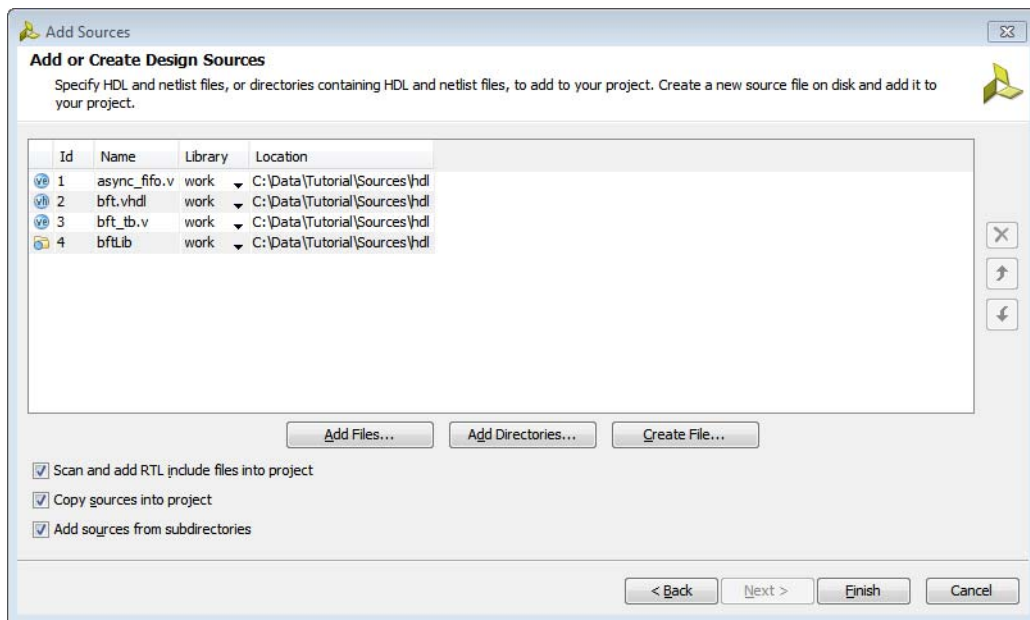


図 3-2 : Add Sources ウィザード : [Add or Create Design Sources] ページ

4. [Create Source File] ダイアログ ボックス (図 3-3) で次のオプションを設定し、[Next] をクリックします。
 - [File type] : Verilog ファイル (.v)、Verilog ヘッダー ファイル (.vh)、SystemVerilog ファイル (.sv)、VHDL ファイル (.vhd) などのファイル形式のいずれかを指定します。
 - [File name] : 新しい HDL ソース ファイルの名前を指定します。
 - [File location] : ファイルを作成するディレクトリを指定します。

注記 : ファイルのプレースホルダーがソースのリストに追加されます。ファイルは [Finish] をクリックすると作成されます。

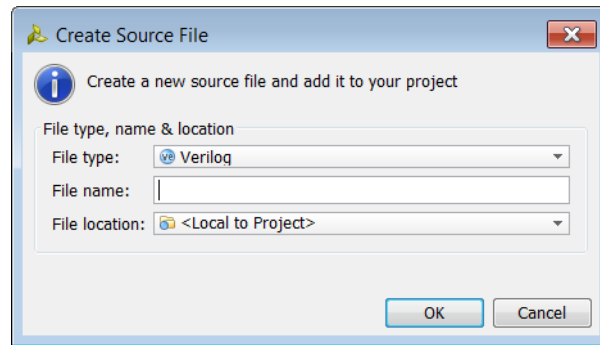


図 3-3 : [Create Source File] ダイアログ ボックス

5. [Create File] を何度かクリックして、プロジェクトに追加するモジュール数を定義します。
6. [Add Sources] ページでソース ファイルに最適なライブラリを指定します。

注記 : デフォルトでは、HDL ソース ファイルは work ライブラリに追加されます。必要に応じて、ユーザー VHDL ライブラリを作成し、参照できます。
7. [Finish] をクリックすると、指定したソースがプロジェクトに追加されます。
8. オプション : [Define Module] ダイアログ ボックス (図 3-4) で次のオプションを使用して Verilog、Verilog ヘッダー、SystemVerilog、または VHDL でモジュールまたはアーキテクチャを定義したら、[OK] をクリックします。
 - [New Source Files] : 複数ファイルを作成したら、定義するモジュールの名前をクリックします。

注記 : このフィールドは複数ファイルを作成した場合にのみ表示されます。
 - [Entity name/Module name] : VHDL コードのエンティティまたは Verilog または SystemVerilog コードのモジュール名の名前を指定します。

注記 : エンティティまたはモジュール名はデフォルトでそのファイル名になりますが、別の名前を付けることもできます。
 - [Architecture name] : RTL ソース ファイルのアーキテクチャを指定します。デフォルトでは [Behavioral] です。

注記 : このオプションは、VHDL コードの場合にのみ表示され、Verilog または SystemVerilog モジュールを定義する場合には表示されません。
 - [I/O Port Definitions] : モジュール定義に追加するポートを定義します。
 - [Port Name] : RTL コードに記述されるポートの名前を定義します。
 - [Direction] : ポートを入力、出力、双方向のいずれかに指定します。
 - [Bus] : ポートがバス ポートかどうかを指定します。次の [MSB] および [LSB] オプションを使用してポートのバス幅を定義します。
 - [MSB] : 最上位ビット (MSB) の数を定義します。[LSB] フィールドと組み合わせて、定義されるバスの幅を指定します。
 - [LSB] : 最下位ビット (LSB) の数を定義します。

注記 : ポートがバス ポートでない場合は、MSB および LSB は無視されます。

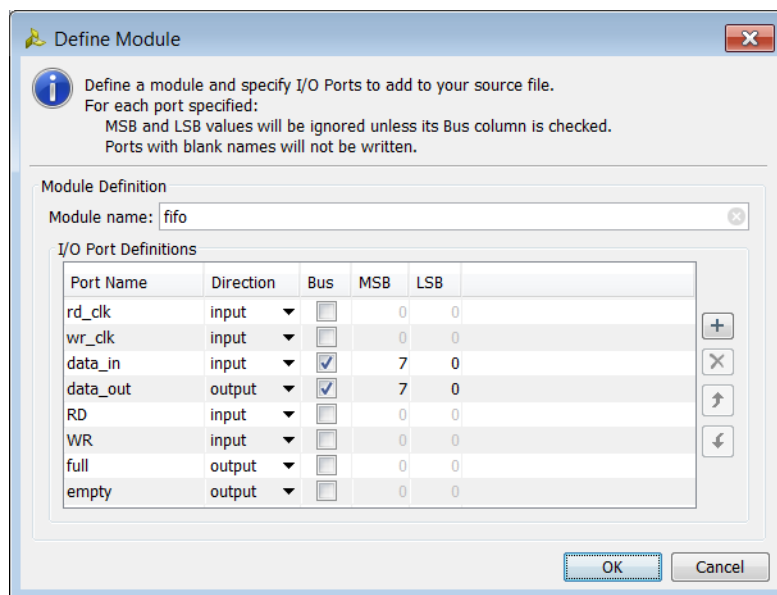


図 3-4 : [Define Module] ダイアログ ボックス

RTL ソース ファイルが作成され、プロジェクトに追加されます。[Sources] ビューに新しく定義したモジュールがリストされます。Vivado IDE テキスト エディターで新しいソース ファイルを編集するには、ファイルをダブルクリックするか、[Open File] ポップアップ メニューをクリックします。新規作成したファイルの編集方法については、第 3 章「テキスト エディターの使用」を参照してください。

デザイン ソースの追加

1. [File] → [Add Sources] をクリックします。

注記：または、Flow Navigator で [Add Sources] をクリックするか、[Sources] ビューのポップアップ メニューから [Add Sources] をクリックします。

2. Add Sources ウィザード (図 3-1) で [Add or Create Design Sources] をオンにし、[Next] をクリックします。

3. [Add or Create Design Sources] ページ (図 3-2) で次のオプションを設定し、[Finish] をクリックします。

- [Add Files] : プロジェクトに追加するファイルを選択するためのファイル ブラウザーが表示されます。RTL プロジェクトには、HDL、EDIF、NGC、BMM、ELF およびその他のファイル タイプを追加できます。

注記：[Add Source Files] ダイアログ ボックスでは、各ファイルまたはディレクトリがそれとわかるようなアイコンで表示されます。小さい赤い四角は、読み出し専用であることを示します。

- [Add Directories] : 選択したディレクトリに含まれるすべてのファイルを追加します。指定したディレクトリにある有効なソース ファイルがすべてプロジェクトに追加されます。
- [Create File] : VHDL、Verilog、Verilog ヘッダー、または SystemVerilog ファイルを作成する [Create Source File] ダイアログ ボックスが開きます。
- [Library] : ファイルまたはディレクトリの RTL ライブラリを指定します。定義済みのライブラリ名から選択するか、新規ライブラリ名を入力します。

注記：このオプションは、VHDL ファイルの場合のみ使用できます。デフォルトでは、HDL ソース ファイルは work ライブラリに追加されます。必要に応じて、ユーザー VHDL ライブラリを作成し、参照できます。Verilog および SystemVerilog ファイルの場合は、work のままにしておいてください。

- [Delete] : 選択したソース ファイルを削除します。
- [Move Selected File Up] : ファイルまたはディレクトリをリストの上方向に移動します。ファイル順は、合成やシミュレーションなどのダウンストリーム プロセスでのエラーレーションおよびコンパイルの順序に影響します。

- [Move Selected File Down]: ファイルまたはディレクトリをリストの下方向に移動します。
- [Scan and Add RTL Include Files into Project]: 追加した RTL ファイルをスキャンし、参照された Verilog の 'include ファイルをローカル プロジェクト ディレクトリにインポートします。
- [Copy Sources into Project]: 元のファイルを参照するのではなく、ローカル プロジェクト ディレクトリにファイルをコピーします。

注記: [Add Directories] ボタンをクリックしてソース ファイルのディレクトリを追加した場合は、ファイルがローカル プロジェクトにコピーされる際にディレクトリ構造もそのまま保持されます。詳細は、[リモート ソースの参照またはプロジェクト ディレクトリへのソースのコピー](#)を参照してください。

- [Add Sources from Subdirectories]: [Add Directories] で指定したディレクトリのサブディレクトリに含まれるソース ファイルをすべて追加します。

最上位モジュールの指定とソース ファイルの順序の変更

Vivado IDE では、デザイン階層の最上位およびプロジェクトに追加されるファイルのエラボレーション、合成、シミュレーションの順序が自動的に判断されます。デザイン階層は、[Sources] ビューの [Hierarchy] タブに表示されます。ファイルの順序は、[Sources] ビューの [Compile Order] タブに表示されている順序になります。

最上位モジュールの自動指定は、デザイン階層の最上位を手動で指定すると上書きできます。最上位モジュールを指定するには、[Sources] ビューの [Hierarchy] タブでファイルを右クリックして [Set as Top] をクリックします。

注記: 選択した最上位モジュールがデザイン ソース ファイルで見つからず、階層アップデート モードが automatic に設定されている場合は、選択した最上位モジュールは自動的に最適なモジュールにリセットされます。

最上位モジュールを変更した場合、Vivado IDE ではその新しい最上位モジュールの要件に基づいて自動的にファイル順が並び替えられます。[Sources] ビューのポップアップ メニューから [Refresh Hierarchy] を使用すると、ソース ファイルのアップデートに基づいてファイルが自動的に並び替えられます。

[Sources] ビューのポップアップ メニューから [Hierarchy Update] を使用すると、このコンパイル順序の自動指定を上書きできます。手動モードの場合は、ユーザーの要件に従って手動でファイル順を変更できます。ソース ファイルの順序を手動で指定するには、[Sources] ビューの [Compile Order] タブでファイルをドラッグして適切な位置に移動します。または、ファイルを右クリックして [Move Up]、[Move Down]、[Move to Top] または [Move to Bottom] をクリックして並び替えることもできます。

すべてのソースのコンパイルまたは評価順のリストを確認するには、Tcl コンソールで `report_compile_order` コマンドを使用します。このコマンドでは、合成、インプリメンテーションおよびシミュレーションでコンパイルまたは評価される順序でファイルがリストされます。RTL のコンパイル順は、合成およびシミュレーション用にリストされます。制約の評価順序は、合成およびシミュレーション用にリストされます。

注記: [Sources] ビューの詳細は、『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』(UG893) [\[参照 4\]](#) を参照してください。

ソース ファイルのイネーブル/ディスエーブル

ソース ファイルを追加または作成すると、[Sources] ビューでデフォルトでイネーブルになります。ソース ファイルは、エラボレーション、合成、またはシミュレーションで使用されないようディスエーブルにできます。

- ソース ファイルをディスエーブルにするには、[Sources] ビューでファイルを右クリックし、[Disable File] をクリックします。
- ソース ファイルをイネーブルにするには、[Sources] ビューでファイルを右クリックし、[Enable File] をクリックします。

リモート ソースの参照またはプロジェクト ディレクトリへのソースのコピー





ソース ファイルは、リモート ロケーションから参照するか、プロジェクト ディレクトリにコピーできます。リモート ファイルを追加した場合、最新のファイルが自動的に検出され、開いているデザインを更新するか ([Refresh your open Designs])、アップデートされたファイルを使用して合成を実行するか ([Synthesize with the latest updates]) を選択するダイアログ ボックスが表示されます。プロジェクトを移動またはアーカイブする可能性がある場合は、すべてのファイルがプロジェクト内に含まれるように、ファイルをプロジェクトにコピーすることをお勧めします。

注記：プロジェクトにファイルをコピーすると、プロジェクトを別のシステムに移行しやすくなりますが、元のファイルへの変更は Vivado IDE では自動的に検出されません。元のファイルを変更したときにコピーしたファイルもアップデートするには、ファイルを削除して追加し直すか、[Sources] ビューの [Replace File] コマンドを使用してファイルを置き換える必要があります。

プロジェクトにソースをコピーするには、次のいずれかを実行します。

- [Add Sources] コマンドでソース ファイルをプロジェクトに追加する際に [Copy Sources into Project] をオンにすると、ソース ファイルがプロジェクト ディレクトリにコピーされます。
- ソース ファイルを最初リモート ソースとして追加し、後でプロジェクト ディレクトリにコピーする場合は、[Sources] ビューでファイルを右クリックして [Copy File into Project] をクリックしてソース ファイルを個別にコピーするか、または [Copy All Files Into Project] をクリックしてすべてのリモート ソース ファイルをコピーします。

[Sources] ビューでは、ソースがローカルにあるかリモートにあるかが次のアイコンで示されます。

- [Local source] : ローカルプロジェクト ディレクトリにコピーされたファイル 
- [Remote source] : ローカルプロジェクト ディレクトリにコピーされなかったファイル 
- [Missing source] : ローカルにもリモートにも見つからないファイル 
- [Read-only source] : Vivado IDE での読み出し専用ファイル 

注記：ファイルはディスクから読み出し/書き込み自体はできますが、Vivado IDE には読み込み/書き込みされません。

ローカル ソース ファイルのアップデート

リモート ソースを参照すると、そのアップデートは Vivado IDE で自動的に検出されます。ソース ファイルがプロジェクトにコピーされている場合は、元のファイルへの変更は検出されません。必要に応じて、ローカル ソース ファイルを手動でアップデートする必要があります。

ローカル ソース ファイルをアップデートするには、次のいずれかの方法を使用してください。

- [Sources] ビューでファイルを選択し、ポップアップ メニューから [Replace File] をクリックします。

ファイル ブラウザーにコピー元のソース ファイルが表示されます。元のディレクトリが変更された場合は、ディレクトリを指定してファイルを選択する必要があります。[OK] をクリックして元のソース ファイルを読み込みすると、ソース ファイルへの変更を含めてプロジェクト ファイルがアップデートされます。

注記：別のファイルを指定すると、選択したファイルがその新しいファイルに置き換えられます。たとえば、元のファイルが File_1.v で、File_2.v を選択した場合、元の File_1.v がプロジェクトから削除され、File_2.v がプロジェクトにコピーされます。

- [Sources] ビューで右クリックして [Add Sources] をクリックし、アップデートされたソース ファイルをプロジェクトに追加します。

追加したファイルがプロジェクトにインポートされます。ただし、同じ名前のローカル ソース ファイルが既に存在するので、[図 3-5](#) に示す [Import Source Conflicts] ダイアログ ボックスが表示され、既存のファイルを上書きするか、新しく追加したファイルを読み込まないかを選択するよう求められます。これはウィザードで [Copy Sources into Project] がオンになっている場合にのみ表示されます。これ以外の場合、同じ名前の外部参照ファイルの方がプロジェクトに追加されます。

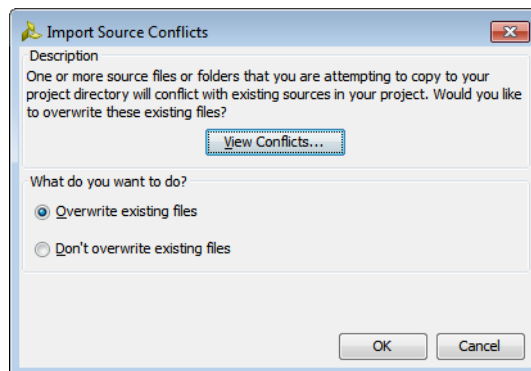


図 3-5 : [Import Source Conflicts] ダイアログ ボックス

制約の操作

Vivado IDE では、Xilinx® Design Constraint (XDC) および Synopsys Design Constraint (SDC) ファイル形式がサポートされます。SDC 形式ではタイミング制約、XDC 形式はタイミングおよび物理制約の両方が指定されます。制約には、配置、タイミング、および I/O に関する指示があります。制約は、RTL 解析、合成、インプリメンテーションを含めたデザインフローのさまざまな段階中に作成できます。

Vivado IDE を使用すると、プロジェクトで制約を柔軟に定義および作成できます。1 つの XDC ファイルを使用してデザインで使用するすべての制約を追加および管理するか、または制約を複数の XDC ファイルに分類して管理できます。複数の制約セットを作成して、さまざまな制約を試したり、複数のバージョンの制約を保存したりすることも可能です。各制約セットには、1 つまたはそれ以上の制約ファイルを含めることができます。

1 つの制約セットを複数のデザインで使用できます。ただし、この場合は変更の管理に注意を払う必要があります。複数のデザインに保存されていない変更がある場合、どの制約ファイルを保存するかを選択するダイアログ ボックスが表示されます。



注意：制約ファイルを保存する際は、保存していないデザインの保存していない制約定義を上書きしないように注意してください。

インプリメント済みデザインには、インプリメンテーション run の実行中に使用された制約セットのスナップショットが保存されます。この制約セットの名前は、開いているプロジェクトのアクティブ制約セットと同じ名前のこともあります。インプリメント済みデザインを開く際、インプリメンテーション run から読み込まれた制約セットがプロジェクト メモリに現在ある制約セットより古い場合があり、デザインを保存したときに新しく定義された制約が失われる可能性があります。通常は Vivado IDE でこれらのリビジョンの問題が管理され、必要に応じて適切な処置をとるようメッセージが表示されますが、メモリにある現在の制約セットとインプリメント済みデザインに関連付けられている制約との間に競合がある可能性があることを念頭に置いてください。

Vivado IDE では、次のビューを使用して制約を作成および変更できます。

- [Timing Constraints] ビュー：プロジェクトの XDC ファイル タイミング制約がすべて表で表示されます。既存の制約はインタラクティブに編集して、ソース ファイルに保存し戻すことができるほか、新規に制約を作成することもできます。
- [Device Constraints] ビュー：表示されたバンクでさまざまな SelectIO™ インターフェイスを設定できます。
- [Physical Constraints] ビュー：Pblock を作成および管理できます。

注記：詳細は、『Vivado Design Suite ユーザー ガイド：制約の使用』(UG903) [参照 14] を参照してください。

制約ファイルの追加と作成

1. [File] → [Add Sources] をクリックします。

注記 : または、Flow Navigator で [Add Sources] をクリックするか、[Sources] ビューのポップアップ メニューから [Add Sources] をクリックします。

2. Add Sources ウィザード (図 3-1) で [Add or Create Constraints] をオンにし、[Next] をクリックします。
3. [Add or Create Constraints] ページ (図 3-6) で次のオプションを設定し、[Finish] をクリックします。
 - [Specify Constraint Set] : 制約ファイルを追加する制約セットを選択します。デフォルトでは現在アクティブな制約セットが選択されていますが、別の制約セットを指定したり、ドロップダウン メニューを使用して新しい制約セットを作成することもできます。
 - [Add Files] : プロジェクトに追加する XDC、SDC、または Tcl ファイルを指定します。

注記 : Tcl スクリプトの詳細については、『Vivado Design Suite ユーザー ガイド : 制約の使用』(UG903) [参照 5] を参照してください。

- [Create File] : 新しい最上位の XDC が作成されます。
- [Remove] : 制約ファイルのリストから選択したファイルが削除されます。
- [Up]/[Down] : XDC、SDC、または Tcl ファイルをリストの上下方向に移動します。XDC、SDC、または Tcl ファイルには、タイミングおよび物理制約を設定し、記述順序に依存するコマンドが含まれます。制約セットに複数の制約ファイルが含まれる場合、[Sources] ビューに表示される順序でファイルが処理されます。最初にリストされているファイルが最初に処理されます。複数の制約ファイルに同じ制約が含まれている場合、後に処理されたファイルの制約定義が使用されます。
- [Copy Constraints into Project] : 元のファイルを参照するのではなく、ローカルプロジェクトディレクトリに制約ファイルをコピーします。

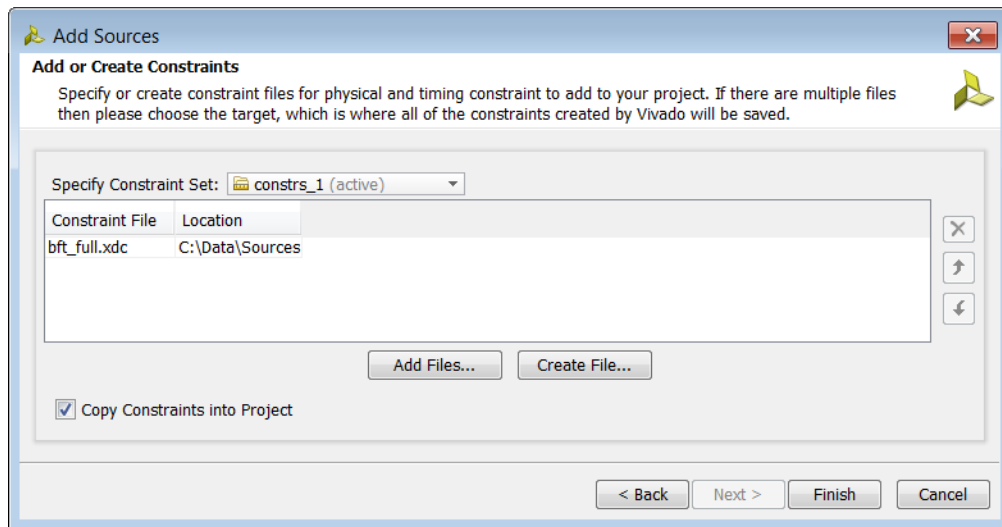



図 3-6 : Add Sources ウィザード : [Add or Create Constraints] ページ

ターゲット XDC ファイルの設定

Vivado IDE では、制約を保存すると、新しく作成された制約が XDC ファイルに書き込まれ、ターゲット XDC ファイルとして認識されます。デフォルトでは、新しい制約セットにはターゲット XDC ファイルはありません。新しく制約を作成する場合は、制約を保存する際にターゲット XDC ファイルを設定する必要があります。制約を保存する必要がある場合は、[Save Constraints] ツールバー ボタン  がオンになります。

[Save Constraints] ツールバー ボタンをクリックすると、[Save Constraints File] ダイアログ ボックス (図 3-7) が表示され、アクティブ制約セットから既存 XDC ファイルを選択するか、新規ファイルを作成してアクティブ制約セットに追加できます。

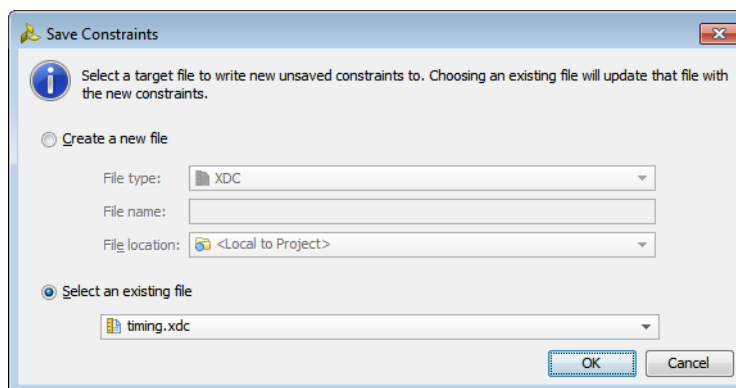


図 3-7 : [Save Target Constraints File] ダイアログ ボックス

XDC ファイルがターゲットとして設定されると、[Sources] ビューのファイル名の横に (target) と表示されます (図 3-8)。ターゲット XDC ファイルは、[Sources] ビューを右クリックして [Set as Target Constraint File] をクリックしていつでも変更できます。



図 3-8 : [Sources] ビューのターゲット XDC ファイル

注記 : [Timing Constraints] ビューで変更した既存の制約は、ターゲット XDC ではなく、その制約が記述されていた元の XDC ファイルに書き込まれます。

元の XDC ファイルの参照またはファイルのコピー

ほかのソース ファイルと同様、XDC ファイルもリモートにあるもの参照するか、ローカルプロジェクトにコピーできます。リモート ファイルを追加した場合、最新のファイル バージョンが自動的に検出され、最新のファイルを使用してデザインを更新するようメッセージが表示されます。

プロジェクトに制約をコピーするには、次のいずれかを実行します。

- [Add Sources] コマンドで制約ファイルをプロジェクトに追加する際に [Copy Constraints into Project] をオンにすると、制約ファイルがプロジェクト ディレクトリにコピーされます。
- 制約ファイルを最初リモート ソースとして追加し、後でプロジェクト ディレクトリにコピーする場合は、[Sources] ビューでファイルを右クリックして [Copy File into Project] をクリックしてソース ファイルを個別にコピーするか、または [Copy All Files Into Project] をクリックしてすべてのリモート ソース ファイルをコピーします。

注記 : 詳細は、[リモート ソースの参照またはプロジェクト ディレクトリへのソースのコピー](#)を参照してください。

制約セットの使用

制約セットとは、個別に管理されている 1 つまたは複数の制約ファイルで、解析およびインプリメンテーションでは 1 つの XDC ファイルに連結されます。制約セットでは、デザイン プロセスのある時点または特定の条件化で使われる制約ファイルが定義されます。複数の制約セットを定義することにより、フロアプランやタイミングの問題を解決するために異なる制約を試すことができます。

XDC ファイルは、合成とインプリメンテーションのいずれか、または両方で使用されます。デフォルトでは、すべての XDC ファイルが合成とインプリメンテーションの両方で使用されるように設定されています。XDC ファイルの [Used In] 設定を変更するには、[Sources] ビューでファイルを選択し、[Source File Properties] ビューの [Used In] フィー

ルドでオン/オフを切り替えます (図 3-9)。

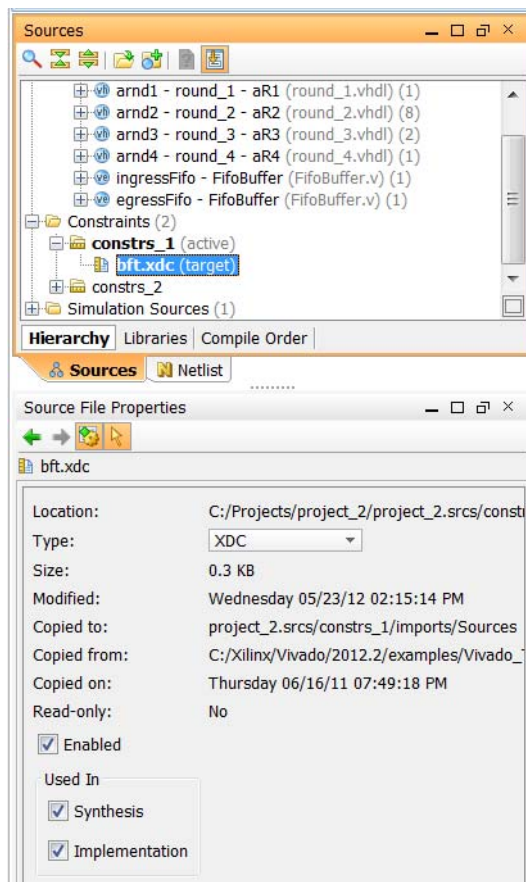


図 3-9 : XDC ファイルの [Used In] 設定

制約セットの作成と編集

1. [Sources] ビューでポップアップメニューから [Edit Constraint Sets] をクリックします。
2. [Edit Constraint Set] ダイアログ ボックスで次のいずれかを実行します。
 - 。 制約セットを編集する場合は、[Specify Constraint Set] フィールドの隣のドロップダウン メニューをクリックし、制約セットを選択します。
 - 。 制約セットを作成する場合は、[Specify Constraint Set] フィールドの隣のドロップダウン メニューをクリックし、[Create Constraint Set] を選択します。[Create Constraint Set Name] ダイアログ ボックス (図 3-10) で制約セットの名前を入力し、[OK] をクリックします。

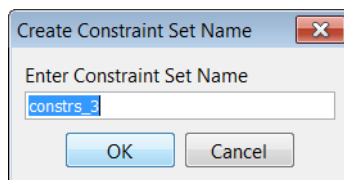


図 3-10 : [Create Constraint Set Name] ダイアログ ボックス

3. [Edit Constraint Set] ダイアログ ボックスで次のオプションを設定し、[OK] をクリックします。
 - 。 [Add Files] : 制約セットに追加する XDC または SDC ファイルを指定します。

- [Create File] : 制約セットに追加する新しい XDC ファイルの名前とディレクトリを指定します。
- [Remove] : 制約ファイルのリストから選択したファイルが削除されます。

注記 : [OK] ボタンを使用すると、まだ制約セットに追加されていないファイルのみを削除できます。既に制約セットに追加されたファイルを削除するには、[Sources] ビューでファイルを右クリックし、ポップアップメニューから [Remove File from Project] をクリックします。

- [Up]/[Down] : XDC および SDC ファイルをリストの上下方向に移動します。XDC および SDC ファイルには、タイミングおよび物理制約を設定し、記述順序に依存するコマンドが含まれます。制約セットに複数の制約ファイルが含まれる場合、[Sources] ビューに表示される順序でファイルが処理されます。最初にリストされているファイルが最初に処理されます。複数の制約ファイルに同じ制約が含まれている場合、後に処理されたファイルの制約定義が使用されます。
- [Copy Constraints into Project] : 元のファイルを参照するのではなく、ローカルプロジェクトディレクトリに制約ファイルをコピーします。

[Save Constraints As] コマンドを使用した制約セットの作成

設計および解析プロセスで制約に加えた変更を新しい制約セットを作成して保存できます。制約は複数の方法で変更できるので、変更を新しい制約セットとして保存すると便利です。[File] → [Save Constraints As] をクリックし、[Save Design As] ダイアログ ボックス (図 3-11) を開き、すべての制約を保存する新しい制約セット名を入力します。

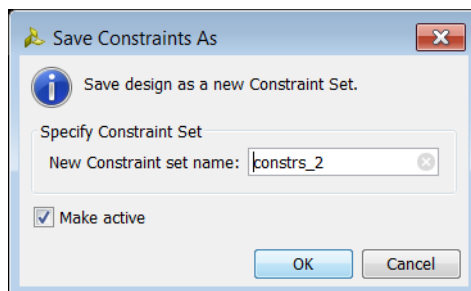


図 3-11 : [Save Constraints As] ダイアログ ボックス

[Save Constraints As] コマンドを使用すると、次の処理が実行されます。

- 新しい制約セットが作成されます。
- アクティブ制約ファイルがローカルプロジェクトディレクトリの新しい制約セットにコピーされます。
- 制約への変更はコピーされた制約ファイルに記述され、元の XDC ファイルは変更されません。
- [Save Design As] ダイアログ ボックスには、新しい制約セットをアクティブにするオプションがあります。

アクティブ制約セットの定義

複数の制約セットが存在する場合、アクティブ制約セットを指定する必要があります。デフォルトでは、合成またはインプリメンテーション run を開始したとき、またはエラーレポート済み、合成済み、またはインプリメンテーション済みデザインを開いたときに、アクティブな制約セットが使用されます。

制約セットをアクティブに設定するには、[Sources] ビューで制約セットを右クリックし、[Make active] をクリックします。[Sources] ビューのアクティブ制約セットの横に太字で (active) と表示されます (図 3-12)。

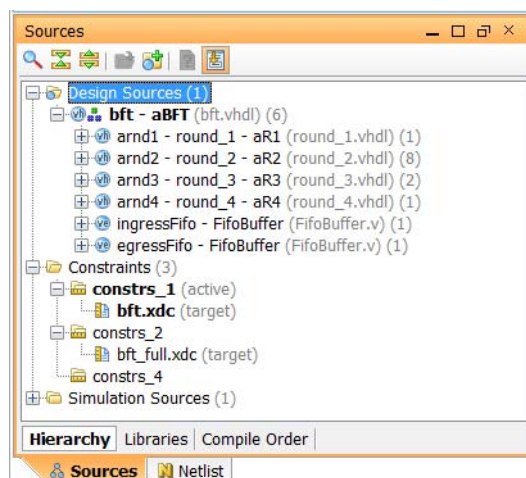


図 3-12 : アクティブ制約セット

制約のエクスポート

Vivado IDE で制約ファイルを作成し、コマンド ライン デザイン フローでのスクリプトに使用することもできます。コマンド ライン フロー用に制約をエクスポートするには、[File] → [Export] → [Export Constraints] をクリックします。

I/O ポートおよびバンクの I/O 規格制約 (ユーザー指定値および Vivado IDE で自動的に割り当てられたデフォルト値の両方) を XDC ファイルにエクスポートするには、[File] → [Export] → [Export I/O Ports] をクリックし、XDC ファイルを生成します。

制約ファイルのイネーブル/ディスエーブル

制約ファイルを追加または作成すると、[Sources] ビューでデフォルトでイネーブルになります。制約ファイルをディスエーブルにすると、エラボレーション、合成、またはインプリメンテーションで使用されないようになります。

- 制約ファイルをディスエーブルにするには、[Sources] ビューでファイルを右クリックし、[Disable File] をクリックします。
- ソース ファイルをイネーブルにするには、[Sources] ビューでファイルを右クリックし、[Enable File] をクリックします。

制約評価順の変更

関連する制約セット内ではユーザー制約の順序を変更できます。順序を変更するには、[Sources] ビューで XDC ファイルをドラッグ アンド ドロップします。

Vivado IDE で処理されるすべての XDC ファイルの順序を取得するには、Tcl コンソールに `report_compile_order -constraints` コマンドを入力します。これにより、ユーザー制約および IP を含むデザインのすべての制約がリストされます。

UCF 制約の変換

Vivado IDE でサポートされるのは、XDC および SDC ファイルのみで、ユーザー制約ファイル (UCF) はサポートされません。次のいずれかの方法を使用すると、UCF 制約を XDC 制約に変換できます。

- PlanAhead™ ツールでデザインを開きます。Tcl コンソールに「write_xdc <filename>.xdc」と入力します。

注記: write_xdc コマンドは、ファイル コンバーターではありません。デザインに XDC ファイルとして適用可能な制約が記述されます。この変換は単に XDC ベースの制約へ移行する際の開始点として使用することを目的としています。



推奨: この方法は、物理制約を変換する場合にのみ推奨されます。タイミング制約、特にタイミング例外制約を変換するのには推奨されません。

- UCF 制約を手動で XDC に変換します。



推奨: この方法は、特にタイミング制約およびタイミング例外制約などの UCF 制約を変換するのに推奨されます。UCF と XDC 制約の構造が異なるため、自動変換はあまりうまくいきません。たとえば、UCF は制約のネットをターゲットにするのに対し、XDC ではセル、ポート、またはピンをターゲットにします。

注記: 詳細は、Vivado Design Suite 移行手法ガイド (UG911) [参照 6] または『Vivado Design Suite ユーザー ガイド : 制約の使用』(UG903) [参照 5] を参照してください。

シミュレーション ソースの操作

Vivado IDE では、RTL プロジェクトのビヘイビアシミュレーション用に、シミュレーション ソースを追加できます。シミュレーション ソース ファイルには、シミュレーションのスティミュラスとして使用する HDL ベースのテストベンチ ファイルが含まれます。シミュレーション ソースは、Vivado シミュレータでのビヘイビアー シミュレーションに使用されます。

シミュレーション ソース ファイルはシミュレーション ファイル セットに格納され、[Sources] ビューにフォルダーとして表示されます。リモートのものを参照するか、ローカル プロジェクト ディレクトリに保存されているものを使用できます。シミュレーション セットを使用すると、シミュレーション コンフィギュレーションごとに異なるソースを定義できます。たとえば、1 つのシミュレーション ソースに 1 つのテストベンチを使用してビヘイビアシミュレーション用のスティミュラスを含め、別のシミュレーション ソースには別のテストベンチを使用することができます。シミュレーション ソースをプロジェクトに追加する際、ファイルを追加するシミュレーション ソース セットを指定できます。

注記: 詳細は、『Vivado Design Suite ユーザー ガイド : ロジック シミュレーション』(UG900) [参照 8] を参照してください。

シミュレーション ソース ファイルの追加と作成

- [File] → [Add Sources] をクリックします。

注記: または、ポップアップメニューまたは Flow Navigator から [Add Sources] をクリックします。

- Add Sources ウィザード (図 3-1) で [Add or Create Simulation Sources] をオンにし、[Next] をクリックします。

- [Add or Create Simulation Sources] ページ (図 3-13) で次のオプションを設定し、[Finish] をクリックします。

- [Specify Simulation Set]: テストベンチ ファイルを含めるシミュレーション セットの名前およびディレクトリを入力します。ドロップダウン リストから [Create Simulation Set] を選択すると、新規シミュレーション セットを定義できます。
- [Add Files]: プロジェクトに追加するシミュレーション ソース ファイルを選択するためのファイル ブラウザーが表示されます。
- [Add Directories]: 選択したディレクトリに含まれるすべてのシミュレーション ソース ファイルを追加します。指定したディレクトリにある有効なソース ファイルがすべてプロジェクトに追加されます。

- [Library] : ファイルまたはディレクトリのライブラリを指定します。定義済みのライブラリ名から選択するか、新規ライブラリ名を入力します。

注記 : このオプションは、VHDL ファイルの場合のみ使用できます。デフォルトでは、HDL ソース ファイルは work ライブラリに追加されます。必要に応じて、ユーザー VHDL ライブラリを作成し、参照できます。Verilog および SystemVerilog ファイルの場合は、work のままにしておいてください。

- [Create File] : シミュレーション ソース ファイルを作成する [Create Source File] ダイアログ ボックスが開きます。
- [Remove] : 選択したソース ファイルを削除します。
- [Move Selected File Up] : ファイルをリストの上方向へ移動します。
- [Move Selected File Down] : ファイルをリストの下方向へ移動します。
- [Scan and Add RTL Include Files into Project] : 追加した RTL ファイルをスキャンし、参照されるインクルード ファイルをすべて追加します。
- [Copy Sources into Project] : ソース ファイルをプロジェクト ディレクトリにコピーします。プロジェクトではローカルにコピーされたバージョンが使用されます。

注記 : [Add Directories] ボタンをクリックしてソース ファイルのディレクトリを追加した場合は、ファイルがローカルプロジェクトにコピーされる際にディレクトリ構造もそのまま保持されます。詳細は、[リモート ソースの参照またはプロジェクト ディレクトリへのソースのコピー](#)を参照してください。

- [Add Sources from Subdirectories] : [Add Directories] で指定したディレクトリのサブディレクトリに含まれるソース ファイルをすべて追加します。
- [Include all design sources for simulation] : sources_1 ファイルセットからのデザイン ソース ファイルをすべてシミュレーション ファイルセットにコピーします。

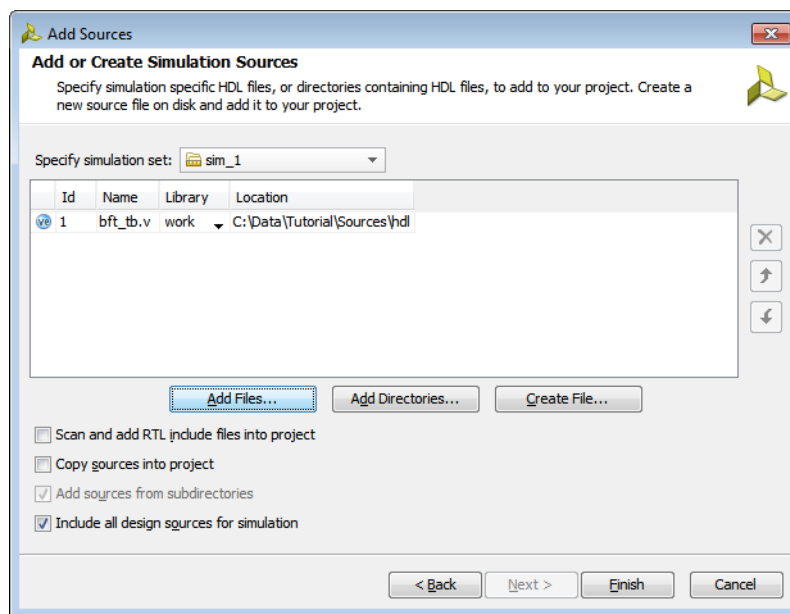


図 3-13 : Add Sources ウィザード : [Add or Create Simulation Sources] ページ

IP ソースの操作

Vivado IDE では、RTL プロジェクトで次のようなタイプの RTL プロジェクトの IP コアを追加および管理できます。

- Vivado Design Suite Xilinx Core Instance (XCI) ファイル
- Vivado IP パッケージャーを使用したユーザーのパックした IP (XCI)

- CORE Generator™ コア (XCO)

XCO ファイルを追加する際に IP コアの既存ステートを維持するには、それに対応する NGC ファイルがディレクトリにないと、インプリメンテーションが問題なく実行されません。ファイルがない場合は、IP のアップグレードがあれば、IP コアを右クリックし [Upgrade IP] をクリックします。

- サードパーティ IP

サードパーティから合成済み NGC または EDIF ネットリストとして提供されている IP もあります。[Add Sources] コマンドを使用すると、これらのファイルをデザインに読み込むことができます。詳細は、[デザイン ソースの操作](#)を参照してください。

- デザイン チェックポイント (DCP) ファイル

注記 : IP の追加、パッケージ、シミュレーション、アップグレードなど IP に関する詳細は、『Vivado Design Suite ユーザー ガイド : IP を使用した設計』(UG896) [\[参照 13\]](#) を参照してください。

既存 IP の追加

1. [File] → [Add Sources] をクリックします。

注記 : または、ポップアップ メニューまたは Flow Navigator から [Add Sources] をクリックします。

2. Add Sources ウィザード ([図 3-1](#)) で [Add Existing IP] をオンにし、[Next] をクリックします。

3. [Add Existing IP] ページ ([図 3-14](#)) で次のオプションを設定し、[Finish] をクリックします。

- [Add Files] : Vivado Design Suite 用の XCI ファイルまたは XCO ファイルを選択できるファイルブラウザーが開きます。
- [Add Directories] : 指定したディレクトリまたは下位ディレクトリから XCI または XCO ファイルを選択できるディレクトリブラウザーが開きます。
- [Remove] : 選択したソース ファイルを削除します。
- [Copy Sources into Project] : IP コア ファイルをプロジェクト ディレクトリにコピーします。プロジェクトではローカルにコピーされたバージョンが使用されます。

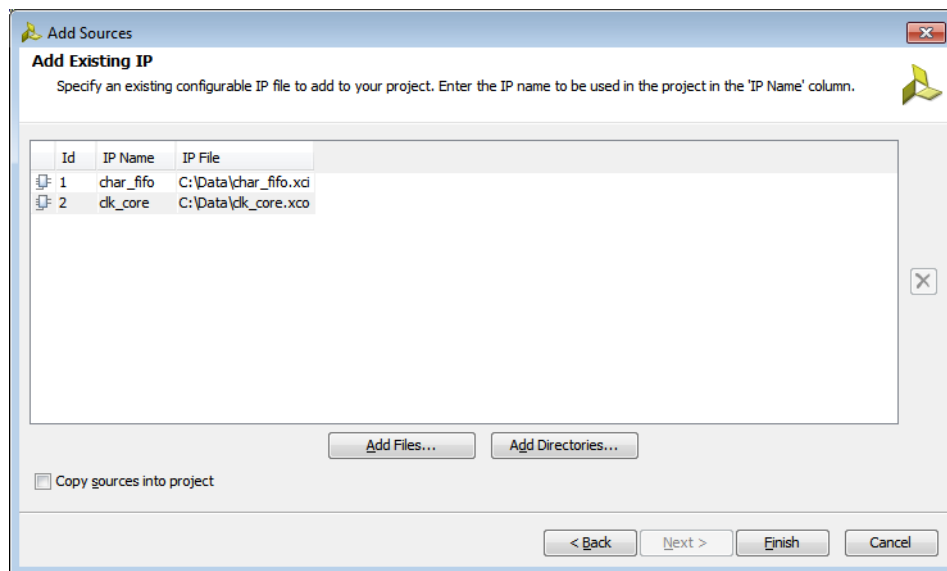


図 3-14 : Add Sources ウィザード : [Add Existing IP] ページ

追加された IP コアは、[Hierarchy]、[Libraries]、[Compile Order] タブのその他のソース ファイルと一緒に、[Sources] ビューの [IP Sources] タブにそれぞれ表示されます。[Sources] ビューにはコアを構成するファイルが表示され、コアを選択すると [Source File Properties] ビューにプロパティが表示されます。

注記 : IP コアの EDIF、Verilog、SystemVerilog ネットリストまたは NGC ファイルは、RTL または ネットリスト ベースのプロジェクトに追加することもできます。詳細は、[第 2 章「合成後プロジェクトの作成」](#)を参照してください。

IP インテグレーターのソースの操作

Vivado IDE では、RTL プロジェクトで IP サブシステム デザイン (拡張子は .bd) を追加および管理できます。Vivado IP インテグレーターを使用すると、IP サブシステム デザインを作成できます。IP インテグレーターでは、Vivado IP カタログからの複数の IP コアをインスタンス化および相互接続することで、複雑なシステム デザインを作成できます。デザインは、Vivado IDE 内の IP インテグレーターを使用してインタラクティブに、または Tcl コマンドのプログラムで作成できます。IP インテグレーターの詳細は、『Vivado Design Suite ユーザー ガイド : IP を使用した設計』(UG896) [\[参照 13\]](#) を参照してください。

注記 : Vivado IP インテグレーターには、Vivado IP カタログで使用可能な IP の一部が提供されています。Vivado IP インテグレーターは、2013.1 リリースでは早期アクセス用になっています。ライセンスの取得については、フィールドアプリケーション エンジニア (FAE) にご連絡ください。

ブロック デザイン ソースを作成または開く

プロジェクトに追加されたブロック デザイン ソースを作成または開くには、次の手順に従ってください。

1. Flow Navigator で [IP Integrator] をクリックします。
2. [Create Block Design] をクリックします。

ブロック デザイン ソースの追加

プロジェクト外に作成されたブロック デザイン ソースを追加するには、次の手順に従ってください。

1. [File] → [Add Sources] をクリックします。
注記 : または、ポップアップ メニューまたは Flow Navigator から [Add Sources] をクリックします。
2. Add Sources ウィザード ([図 3-1](#)) で [Add Existing Block Design Sources] をオンにし、[Next] をクリックします。
3. [Add Existing Block Design Sources] ページ ([図 3-15](#)) で次のオプションを設定し、[Finish] をクリックします。
 - [Add Files] : デザインに追加する IP インテグレーター ブロック デザイン (BD) ファイルを選択できるファイル ブラウザーが開きます。
 - [Remove] : 選択したソース ファイルを削除します。
 - [Move Selected File Up] : ファイルをリストの上方向へ移動します。
 - [Move Selected File Down] : ファイルをリストの下方向へ移動します。

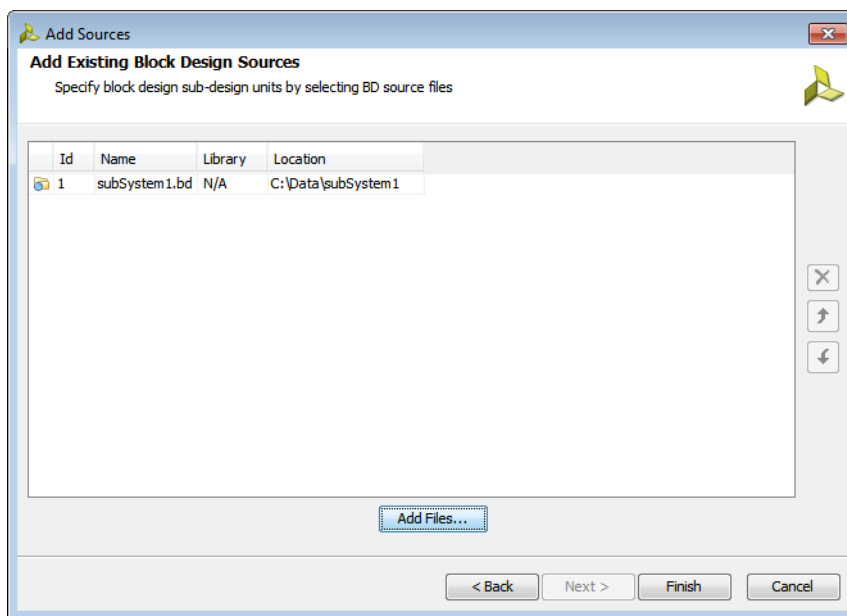


図 3-15 : Add Sources ウィザード : [Add Existing Block Designs] ページ

サブシステム デザインで使用するターゲット パーツが現在のプロジェクトのターゲット パーツと異なる場合は、[Mismatched Parts] ダイアログ ボックス (図 3-16) が開きます。次のオプションのいずれかを選択します。

- [Ignore the Difference] : 現在のプロジェクトまたは IP サブシステムのターゲット パーツを変更せずに、IP サブシステム デザインをインポートします。
- [Apply sub-design part to the project] : 現在のプロジェクトのターゲット パーツを変更し、サブシステム デザインからのターゲット パーツを使用します。
- [Apply project part to sub-designs] : サブシステム デザインのターゲット パーツを変更し、現在のプロジェクトのターゲット パーツを使用します。

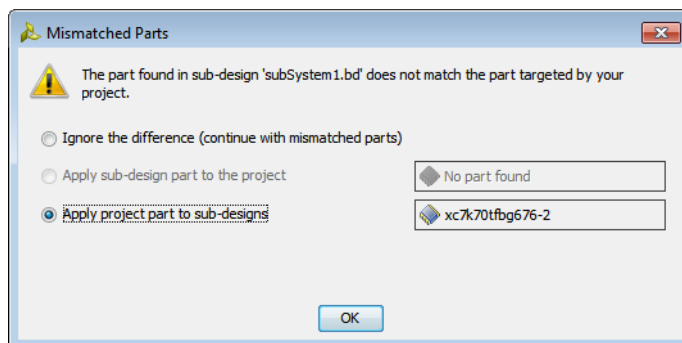


図 3-16 : [Mismatched Parts] ダイアログ ボックス

DSP ソースの操作

Vivado IDE ツールでは、既存のザイリンクス System Generator デザイン モデル ファイル (.mdl) を DSP モジュールとしてインポートできます。このモデルは下位モジュールとして階層レベルに追加、またはデザインの最上位にインポートできます。Vivado IDE ツール内から新規 DSP モジュールを定義して、ザイリンクス System Generator でデザインを完了させることもできます。

System Generator はザイリンクスの DSP デザイン ツールで、RTL ソース ファイル、Simulink® および MATLAB® ソフトウェア モデル、および DSP システムの C/C++ コンポーネントを 1 つのシミュレーションおよびインプリメンテーション環境にまとめることができます。

System Generator デザインは、よく大容量 HDL デザインに組み込まれます。System Generator ではスタンドアロンの FPGA デザインの作成およびインプリメンテーションがサポートされますが、Vivado IDE ツールでプロジェクトを開始して、System Generator を使用してプロジェクトの DSP モジュールを開発することをお勧めします。こうすることで、Vivado IDE ツールで FPGA デザインのプロジェクトを管理できます。まず、DSP モジュールを System Generator 内で開発および管理しておいてから、Vivado IDE で 1 つのソース ファイルとして処理します。

DSP モジュールの追加

1. [File] → [Add Sources] をクリックします。

注記 : または、ポップアップ メニューまたは Flow Navigator から [Add Sources] をクリックします。

2. Add Sources ウィザード (図 3-1) で [Add or Create DSP Sources] をオンにし、[Next] をクリックします。

3. [Add or Create DSP Sources] ページ (図 3-17) で次のオプションを設定し、[Finish] をクリックします。

- [Add Sub-Design] : プロジェクトに追加する既存の System Generator モデル ファイル (MDL) を指定するためのファイルブラウザが開きます。
- [Create Sub-Design] : System Generator が起動するので、プロジェクトに追加する新しい DSP モジュールを定義できます。
- [Add Files] : プロジェクトに追加する既存の System Generator モデル ファイル (MDL) を指定するためのファイルブラウザが開きます。
- [Create File] : System Generator が起動するので、プロジェクトに追加する新しい DSP モジュールを定義できます。
- [Remove] : 選択した DSP ソース ファイルを削除します。
- [Move Up] : 選択したソースをリストの上方向に移動します。
- [Move Down] : 選択したソースをリストの下方向に移動します。
- [Copy Sources into Project] : DSP モデル ファイルをプロジェクト ディレクトリにコピーします。プロジェクトではローカルにコピーされたバージョンが使用されます。

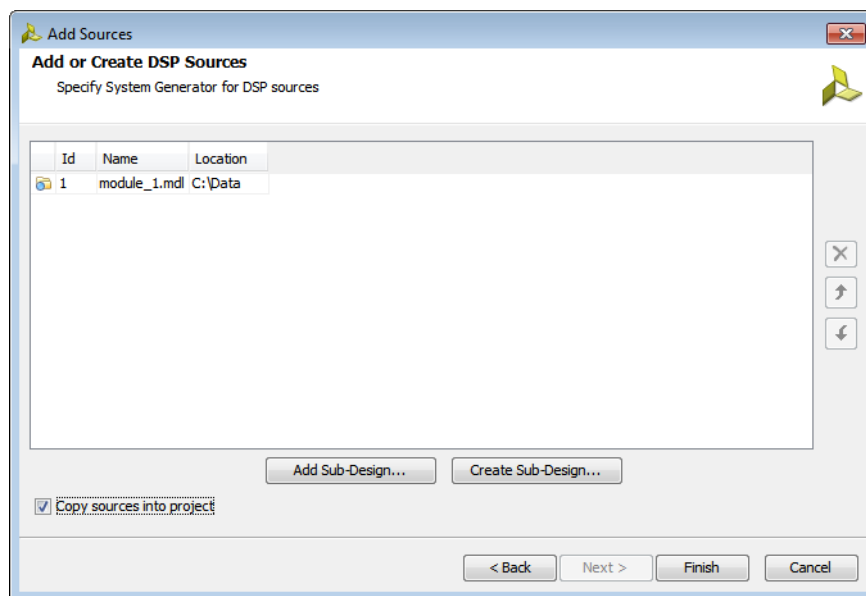


図 3-17 : Add Sources ウィザード : [Add or Create DSP Sources] ページ

4. [Create Sub-Design] を選択すると、System Generator および MATLAB で DSP ソースの作成および管理が開始されます (図 3-18)。

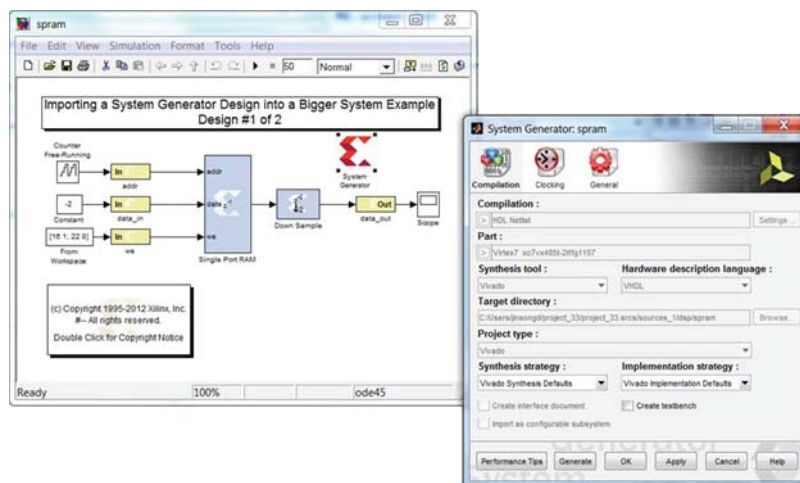


図 3-18 : System Generator

追加された DSP ソースが [Sources] ビューの [IP Sources] タブにそれぞれ表示されます。DSP ソースは、[Hierarchy]、[Libraries]、[Compile Order] タブのその他のソース ファイルと一緒に表示されます。[Sources] ビューで DSP モジュールを選択すると、それに関連するファイルが表示され、[Source File Properties] ビューにはそのプロパティが表示されます。

注記 : DSP ソースは Tcl コマンドの `create_sysgen` を使用しても追加できます。このコマンドは新しい DSP 下位モジュールを作成するためにも使用できます。Vivado IDE では、新規 MDL ファイルを作成し、それを下位モジュールとしてプロジェクトに追加します。

ターゲットの生成

System Generator デザインが終了したら、[Sources] ビューの DSP モジュールのポップアップ メニューを使用して FPGA ターゲット ファイルを生成できます。これらのコマンドは、DSP ソースを [Sources] ビューで選択すると使用できるようになります (図 3-19)。

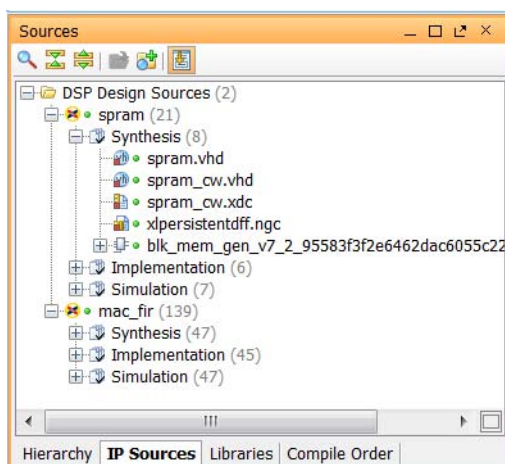


図 3-19 : [Sources] ビューの DSP ソース

ターゲットとは、現在のデザインの合成、シミュレーション、インプリメンテーションをサポートするのに必要な DSP モジュールのさまざまなデザイン エLEMENT のことです。これには、最上位モジュールの定義、インスタンス化テンプレート、合成済みネットリスト、その他の関連資料が含まれます。

[Sources] ビューで DSP ソースを選択すると、次のポップアップ メニューが表示されます。

- **[Create Top HDL]** : DSP モジュールの最上位ラッパー ファイルを作成し、プロジェクトに追加します。このコマンドは、System Generator デザインが現在のプロジェクトの最上位の場合に使用します。
- **[View Instantiation Template]** : DSP モジュールを RTL デザインにインスタンス化するために使用するインスタンス化テンプレートを作成します。インスタンス化テンプレートは別の RTL ファイルにコピーして貼り付けると、その階層で DSP モジュールのインスタンスを作成できます。
- **[Create Testbench]** : Simulink シミュレーションから抽出したテスト ベクター ファイルが System Generator で書き出され、シミュレーション用の HDL テストベンチとスクリプト ファイルが生成されます。テストベンチは、シミュレーション セットの [Sources] ビューに追加されます。
- **[Generate]** : System Generator モデルから合成、インプリメンテーション、およびシミュレーションのターゲット データが生成されます。これにより、System Generator および MATLAB が起動し、必要なデータが作成されます。
- **[Reset]** : 現在のプロジェクトおよびローカルプロジェクト レポジトリから指定したターゲット データが削除されます。ターゲット データは、必要に応じて再生成できます。

エンベデッド ソースの操作

エンベデッド開発キット (EDK) は、ユーザーのハードウェアおよびソフトウェア システム コンポーネントに統合させるために使用可能なツールおよび IP の総称です。EDK には、Xilinx Platform Studio (XPS) および Software Development Kit (SDK) という 2 つのツールが含まれます。

エンベデッド プロセッサ システムのハードウェア部分を設計するには、XPS を使用します。XPS では、マイクロプロセッサおよびペリフェラルの仕様、これらのコンポーネントの接続およびプロパティを設定します。効果的なエンベデッド システム デザインの詳細は、『EDK コンセプト、ツール、テクニック ガイド』(UG683)[[参照 14](#)]を参照してください。

EDK 環境ではデザインの作成およびインプリメンテーションがサポートされますが、Vivado IDE ツールでプロジェクトを開始して、XPS を使用してプロジェクトのエンベデッド プロセッサ ソースを開発することをお勧めします。こうすることで、Vivado IDE ツールで FPGA デザインのプロジェクトを管理できます。まず、エンベデッド プロセッサ デザインを XPS 内で開発および管理しておいてから、Vivado IDE で 1 つのソース ファイルとして処理します。[図 3-20](#) は、この統合エンベデッド デザイン フローを示しています。



重要 : Vivado IP インテグレーター は、Zynq™ デバイスおよび MicroBlaze™ プロセッサをターゲットにするデザインを含めたエンベデッド プロセッサ デザイン用の Xilinx Platform Studio (XPS) に代わるものです。XPS では MicroBlaze プロセッサをターゲットにするデザインはサポートされますが、Zynq デバイスはサポートされません。IP インテグレーターも XPS も Vivado IDE から使用できます。



ヒント : Zynq デバイスをターゲットにする既存の XPS プロジェクトの場合は、Vivado Design Suite 移行手法ガイド (UG911) [[参照 6](#)] に示す方法でデザインを Vivado IP インテグレーター に移行する必要があります。

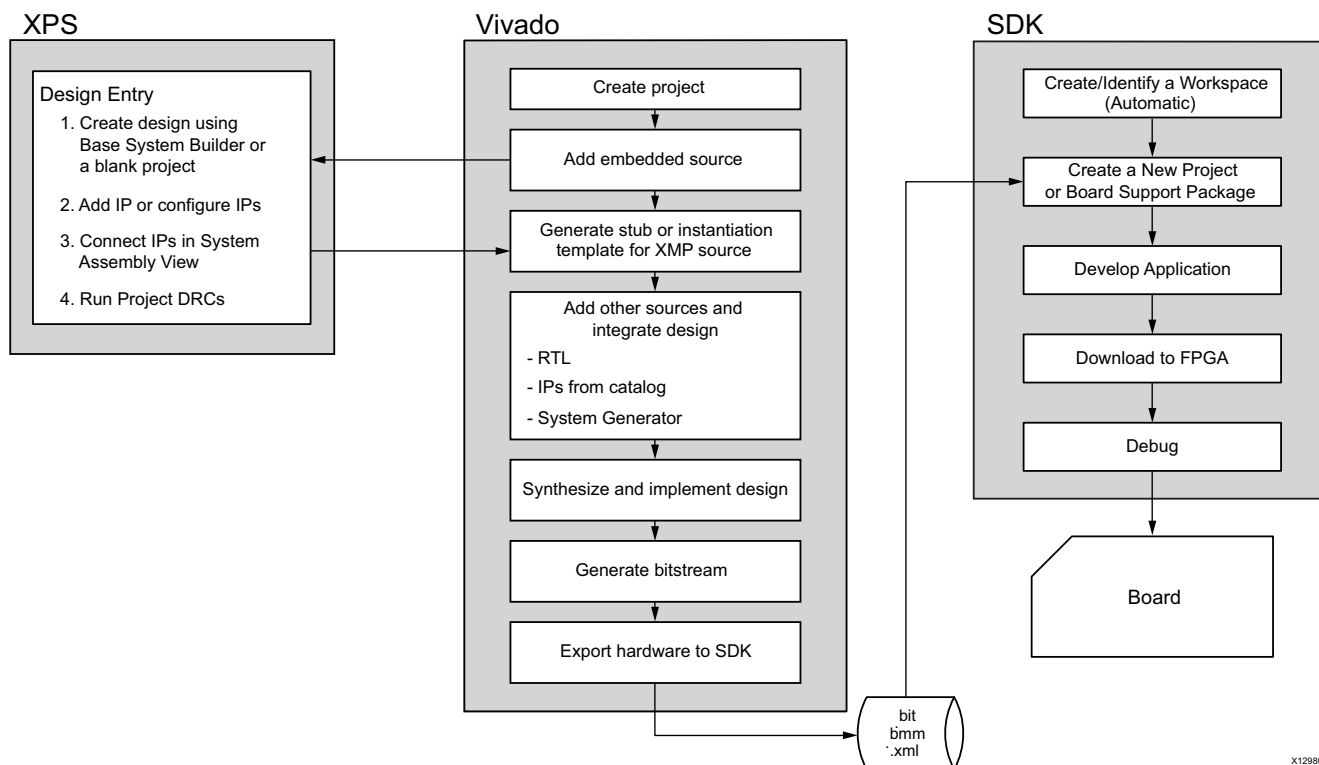


図 3-20 : エンベデッド デザイン フロー

エンベデッド プロセッサの追加

Vivado IDE では、既存の Xilinx Microprocessor Project (.xmp) ファイルを XPS からインポートしたり、Vivado IDE ツール内で新しいエンベデッド プロセッサ サブデザインを定義しておいてから、XPS を開いてプロジェクトを作成および管理したりできます。

注記：または、XPS および System Generator をスタンドアロンで使用して、結果のネットリストおよび制約を Vivado プロジェクトにソースとして追加することもできます。



重要：デザインに XMP ファイルを追加する際は、パスにスペースを含めないようにしてください。XPS では現在のところパスにスペースを使用できません。

1. [File] → [Add Sources] をクリックします。

注記：または、ポップアップ メニューまたは Flow Navigator から [Add Sources] をクリックします。

2. Add Sources ウィザード (図 3-1) で [Add or Create Embedded Sources] をオンにし、[Next] をクリックします。

3. [Add or Create Embedded Sources] ページ (図 3-21) で次のオプションを設定し、[Finish] をクリックします。
 - [Add Sub-Design] : Vivado IDE プロジェクトに追加する既存の Xilinx Microprocessor Project (XMP) ファイルを指定するためのファイル ブラウザーが開きます。
 - [Create Sub-Design] : Vivado IDE プロジェクトに追加する新しいサブデザインを定義するため、XPS が起動します。詳細は、[サブデザインの作成](#)を参照してください。
 - [Remove] : 選択したサブ デザインを削除します。
 - [Move Up] : 選択したサブ デザインをリストの上方向に移動します。
 - [Move Down] : 選択したサブ デザインをリストの下方向に移動します。
 - [Copy Sources into Project] : 元のエンベデッド プロセッサ デザイン (XMP) ファイルをプロジェクト ディレクトリにコピーします。プロジェクトではローカルにコピーされたバージョンが使用されます。

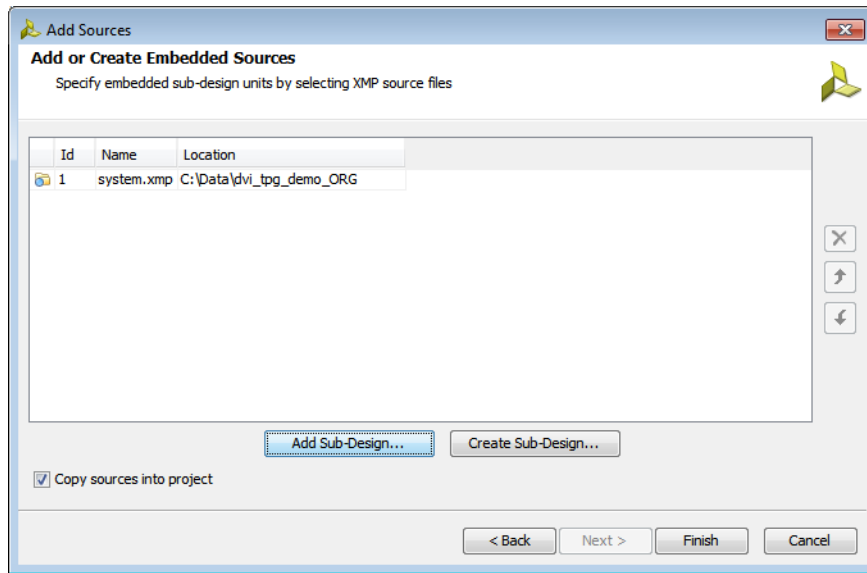


図 3-21 : Add Sources ウィザード : [Add or Create Embedded Sources] ページ

追加されたサブデザインは、[Sources] ビューの [IP Sources] タブにそれぞれ表示されます。[Hierarchy]、[Libraries]、[Compile Order] タブにもその他のソース ファイルが表示されます。[Sources] ビューにはサブデザインを構成するファイルが表示され、サブデザインを選択すると [Source File Properties] ビューにプロパティが表示されます。

サブデザインの作成

次は、エンベデッド デザインを定義するプロセスの簡単な概要です。詳細は、『EDK のコンセプト、ツール、テクニック』(UG683)[\[参照 14\]](#) および『エンベデッド システム ツール リファレンス マニュアル』(UG111) [\[参照 15\]](#) を参照してください。

[Add or Create Embedded Sources] ダイアログ ボックスで [Create Sub-Design] を選択すると、XPS が起動し、新しいエンベデッド サブデザインを定義できるようになります。ターゲット パーツや TDP などの Vivado IDE ツールのプロジェクトのプロパティは、XPS を開くと自動的に移行されます。XPS では、これが新しいサブデザインであることが認識され、ボード デザインのしやすい Base System Builder ウィザードを起動するかどうか尋ねるメッセージが表示されます。

1. [Yes] をクリックします。
2. Base System Builder (BSB) ウィザード (図 3-22) で次のオプションを設定し、[OK] をクリックします。

BSB ウィザードを使用すると、素早くシステムを構築できます。エンベデッド デザイン プロジェクトの中には、BSB ウィザードだけで完成できるものもあります。複雑なプロジェクトの場合は、BSB ウィザードで作成したものをベースに、エンベデッド デザインをカスタマイズしていきます。

注記 : フォームのほとんどが現在のプロジェクトからのデータで埋まっており、変更はできません。これは、Vivado IDE プロジェクトと XPS プロジェクトの統合を保護するためです。

- [Project File] : [Create Sub-Design] ダイアログ ボックスで指定したサブデザインの名前が表示されます。この名前は、Vivado IDE からインポートされたものです。
- [Select an Interconnect Type] : AXI System を指定します。レガシー プロセッサ ローカル バス (PLB) デザインは Vivado ツールではサポートされないため、これはハード コードで選択されます。
- [Select Existing .bsb Settings File] : オプションで、以前のセッションからの BSB 設定ファイルを指定して、同じ選択がこの BSB セッションでも自動的に使用されるようにします。
- [Set Project Peripheral Repository Search Path] : カスタム pcore、Board Support Packages (BSP)、およびソフトウェア サービスを含むユーザー レポジトリを指定します。レポジトリ検索パスを複数指定する場合は、パスをセミコロン (;) で区切ってください。

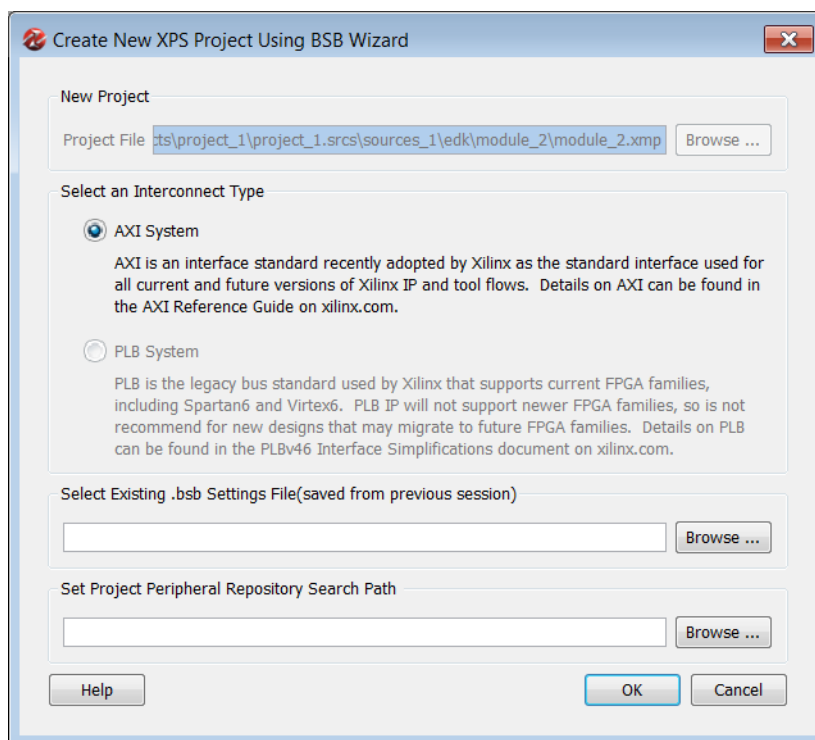


図 3-22 : Base System Builder ウィザード

3. [Board and System Selection] ページ (図 3-23) でエンベデッド デザインのターゲット デザイン プラットフォーム (TDP) またはプラットフォームを選択し、[Next] をクリックします。

提供される TDP の定義は、Vivado IDE ツールのプロジェクトで選択したターゲット パーツを含んだものに制限されます。BSB には、特定の FPGA デバイス、外部メモリ、I/O デバイス、クロック リソース、リセット極性など、どのデバイスがターゲット ボードにあるかを決定する機能があります。次のウィザード ページは、選択したボードからの情報を元にカスタマイズされており、必要な入力が最小限で済むようになっています。

注記 : 選択した FPGA がサポートされるボードで使用可能でない場合は、デバイスに対するボードが存在しないメッセージが表示されます。この場合は、[Create a System for a Custom Board] をオンにする必要があります。

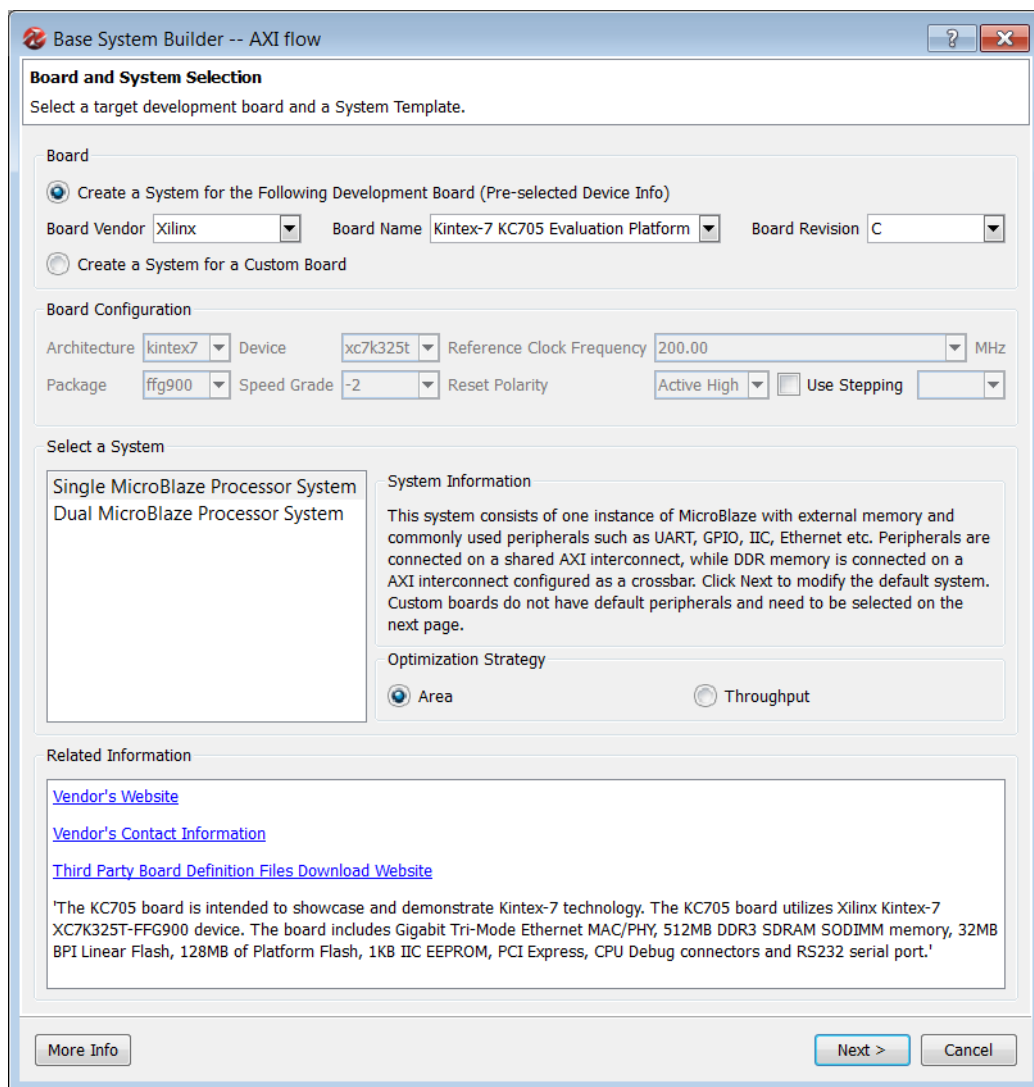


図 3-23 : Base System Builder ウィザード : [Board and System Selection] ページ

4. [Processor, Cache, and Peripheral Configuration] ダイアログ ボックス (図 3-24) で、指定した TDP で使用可能なエンベデッド デザインに含めるペリフェラルを指定し、[Finish] をクリックします。

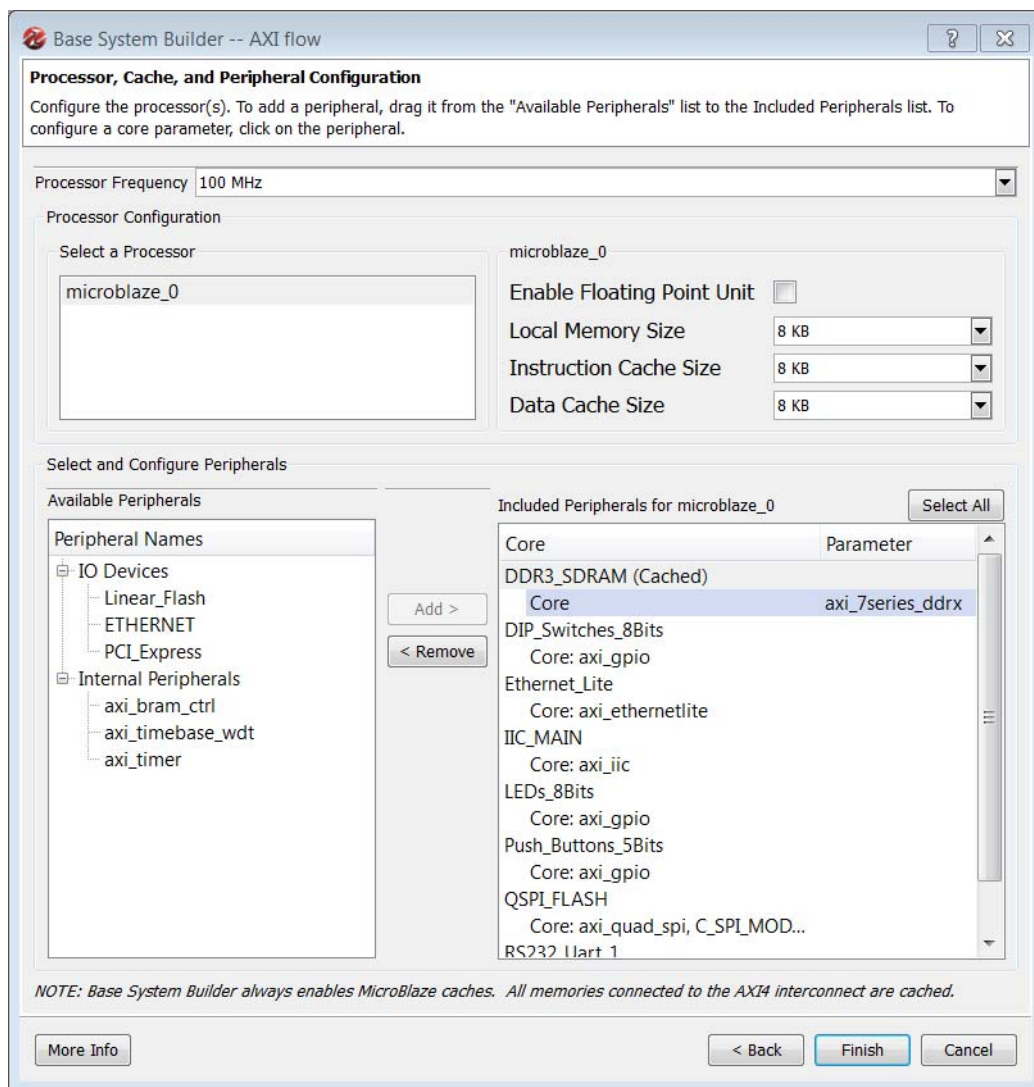


図 3-24 : Base System Builder ウィザード : [Processor, Cache, and Peripheral Configuration] ページ

エンベデッド デザインが作成され、プロジェクトが XPS で開きます。エンベデッド プロセッサ サブデザインは、XPS ツールで変更および管理できます。

注記 : XPS ソースは Tcl コマンドの `create_xps` を使用しても作成できます。このコマンドは新しいエンベデッド システムをプロジェクトに追加するためにも使用できます。Vivado IDE では、新規 XMP ファイルを作成し、それを下位モジュールとしてプロジェクトに追加します。

ターゲットの生成

XPS ツールを閉じると、最上位プロジェクト デザイン ファイル (.xmp) および Microprocessor Hardware Specification (.mhs) ファイルが Vivado IDE の [Sources] ビューに追加されます。[Sources] ビューでエンベデッド デザイン ソースを展開表示すると、そのサブデザインに関連するさまざまなターゲット ファイルが表示されます (図 3-25)。

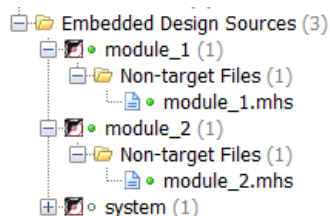


図 3-25: エンベデッド デザイン ソース

ターゲットとは、現在のプロジェクトのオブジェクトをサポートするのに必要な XPS サブデザインのさまざまなデザイン エレメントのことです。これには、最上位モジュール定義、インスタンス化テンプレート、合成済みネットリスト、ログ ファイルやデータシートなどの関連資料などが含まれます。エンベデッド デザインのプロジェクトファイル (.xmp) は、[Sources] ビューの [Hierarchy] タブに表示されます。

[Sources] ビューでエンベデッド ソースを選択すると、次のポップアップ メニューが表示されます。

- **[Create Top HDL]**: エンベデッド デザインの最上位ラッパー ファイルを作成し、プロジェクトに追加します。このコマンドは、エンベデッド デザインが現在のデザインの最上位の場合に使用します。
- **[View Instantiation Template]**: エンベデッド デザインを RTL デザインにインスタンス化するために使用するインスタンス化テンプレートを作成します。インスタンス化テンプレートは別の RTL ファイルにコピーして貼り付けると、その階層でサブデザインのインスタンスを作成できます。
注記: テンプレート ファイルはプロジェクトに追加されません。
- **[Create Testbench]**: エンベデッド デザインのテストベンチを作成します。テストベンチは、シミュレーション セットの [Sources] ビューに追加されます。
- **[Generate]**: 合成、インプリメンテーション、シミュレーション用に指定したターゲット データを作成します。ターゲット データには、そのサブデザインの Verilog または VHDL ファイル、ラッパー ファイル、BMM (Block Memory Map) モデル、最上位シミュレーション モデルが含まれます。
- **[Reset]**: 現在のプロジェクトから指定したターゲット データが削除されます。ローカルなプロジェクト レポジトリからも生成されたターゲット データが削除されます。ターゲット データは、必要に応じて再生成できます。

ターゲット データを生成すると、そのエンベデッド サブデザインに対して /synthesis および /implementation ディレクトリが作成されます。これは、XPS で [Hardware] → [Generate Netlist] または [Hardware] → [Generate Bitstream] をクリックしても作成されます。

既存の XPS エンベデッド デザイン ソースを追加すると、/synthesis および /implementation ディレクトリはエンベデッド デザインのサブディレクトリ (プロジェクトの外部) に作成されますが、[Create Sub-Design] コマンドでエンベデッド プロセッサ デザインを現在のプロジェクトに追加すると、/synthesis および /implementation 下位ディレクトリがローカルなプロジェクト ディレクトリ (<project>.srcs\sources_1\edk\<subdesign_name>) の下にできます。

ハードウェアのエクスポート

Vivado IDE は SDK とも統合されており、プロジェクト内のエンベデッド プロセッサ ソースのソフトウェア デザインがサポートされます。SDK を使用してエンベデッド プロセッサを含むプロジェクトのソフトウェアを開発するには、次を実行します。

1. [File] → [Export] → [Export Hardware] をクリックします。
2. [Export Hardware for SDK] ダイアログ ボックス (図 3-26) で次のオプションを設定し、[OK] をクリックします。

- [Source]: エクスポートするソース XPS プロジェクト ファイルを指定します。
- [Export to]: ハードウェアをエクスポートするディレクトリを指定します。デフォルトでは、ハードウェアファイルは次のローカルのプロジェクト ディレクトリに書き込まれます。
<project>.sdk/SDK/SDK_Export/hw
- [Workspace]: SDK で使用されるワークスペースのディレクトリを指定します。デフォルトでは、SDK ワークスペース ファイルは次のローカルのプロジェクト ディレクトリに書き込まれます。
<project>.sdk/SDK/SDK_Export
- [Include Bitstream]: [Export to] フィールドで指定したディレクトリにビットストリーム ファイルをコピーします。
- [Export Hardware]: エンベデッド プロセッサ デザインのソフトウェア開発をサポートするのに必要なファイルが生成されます。
- [Launch SDK]: ハードウェア ファイルの生成後に SDK ツールを起動します。

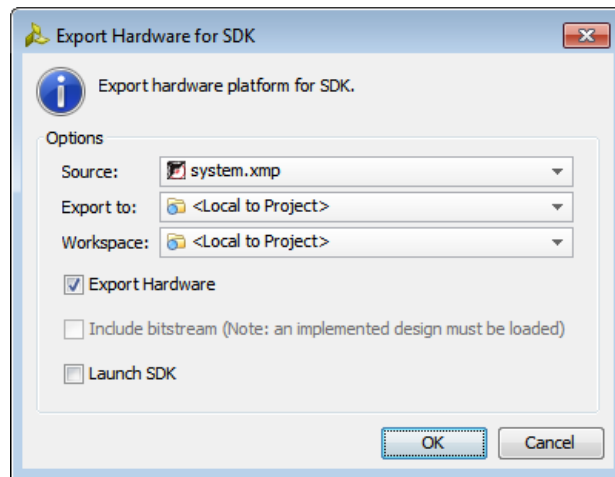


図 3-26 : [Export Hardware for SDK] ダイアログ ボックス

Vivado IDE で [Launch SDK] をオンにしておくと、デザインの Hardware Platform Specification (system.xml) ファイルがエクスポートされ、SDK が起動します。詳細は、SDK ヘルプ を参照してください。

ビットストリーム ファイルの作成

エンベデッド プロセッサ システムを起動するには、システムのハードウェアおよびソフトウェア コンポーネントの両方を FPGA にダウンロードして、メモリをそれぞれプログラムする必要があります。このためには、ブロック RAM をターゲットとしたソフトウェア アプリケーションを含むビットストリーム ファイルを作成する必要があります。Vivado IDE では、エンベデッド プロセッサに関連するソフトウェアの Executable and Linkable Format (ELF) ファイルで初期化されたブロック RAM を使用してハードウェア ビットストリームが作成されます。この後、Vivado IDE および iMPACT ツールで、ビットストリームを使用して FPGA をプログラムできます。詳細は、『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』(UG908) [参照 12] を参照してください。

使用可能なプロセッサ インスタンスに関連する ELF ファイルは、Vivado IDE で [Tools] → [Associate ELF Files] をクリックすると、追加またはアップデートできます。[Associate ELF Files] ダイアログ ボックス (図 3-27) が開きます。このダイアログ ボックスでは、ビットストリーム ファイルを生成するとき、またはデザインをシミュレーションするときに、すべての使用可能なプロセッサ インスタンスに対して、使用する ELF ファイルを指定できます。

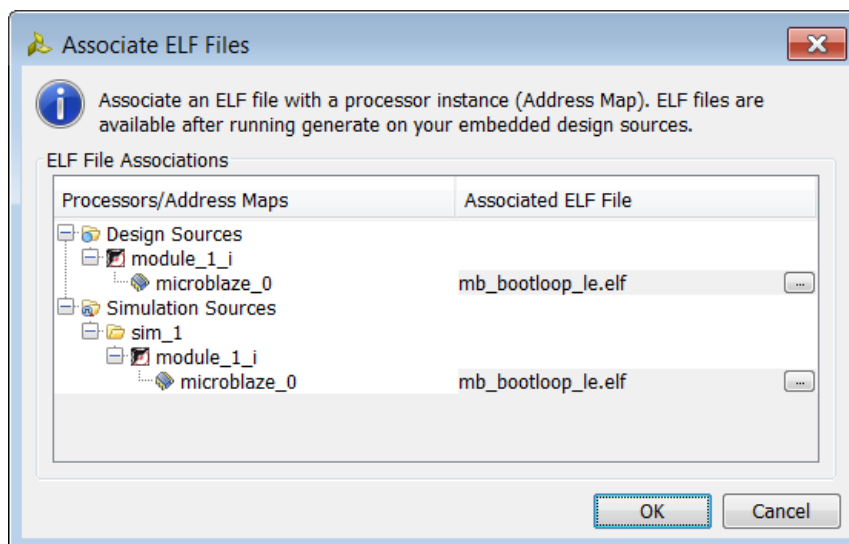


図 3-27 : [Associate ELF Files] ダイアログ ボックス

ELF ファイルは BMM ファイルで指定したブロック RAM を生成します。Vivado IDE で生成されたこの BIT ファイルには、選択した実行コードで初期化されたブロック RAM が含まれます。詳細は、『EDK コンセプト、ツール、テクニック ガイド』(UG683)[[参照 14](#)] を参照してください。

ソース ファイルの編集

Vivado IDE には、RTL、XDC、Tcl およびその他のテキスト ソースを作成または変更するためのテキスト エディターが含まれます。このテキスト エディターには、構文認識機能があるので、RTL、XDC、Tcl のキーワードが認識されて色分けされます。同時に複数のファイルを開くことができ、各ファイルのタブをクリックすると、開いているファイルを表示できます。

ファイルを変更して保存していない場合は、ビュー タブのファイル名の横にアスタリスク (*) が表示されます。ファイルを保存するには、次のいずれかを実行します。

- [File] → [Save File] をクリックします。
- Vivado IDE テキスト エディターでポップアップ メニューから [Save File] をクリックします。
- Vivado IDE テキスト エディターで [Save File] ツールバー ボタンをクリックします。

ファイルを閉じるときに保存していない変更がある場合は、変更を保存するかどうかを確認するダイアログ ボックスが表示されます。ソース ファイルは、[Save As] コマンドで新しいディレクトリに保存することもできます。

注記 : Vivado IDE テキスト エディターの詳細は、『Vivado Design Suite ユーザー ガイド : Vivado IDE の使用』(UG893) [[参照 4](#)] を参照してください。

テキスト エディターの使用

Vivado IDE のテキスト エディターは、[Schematic]、[Messages]、[RTL Netlist]、[Hierarchy] などのその他ビューとクロスプロープできます。

テキスト エディターのツールバーには、次のようなボタンが含まれます。

- [Save] : 開いたファイルに現在の変更を保存します。
- [Undo] : 開いているファイルで実行した最後の変更を取り消します。
- [Redo] : 開いているファイルで実行した最後の変更をやり直します。

- [Cut]: 選択したセクションを切り取って、クリップボードに貼り付けます。
- [Copy]: 選択したセクションをコピーして、クリップボードに貼り付けます。
- [Paste]: クリップボードの内容を指定した位置に貼り付けます。
- [Delete]: 選択したセクションを削除します。削除したセクションは、クリップボードにはコピーされません。
- [Toggle Line Comments]: 選択した行の開始に適切なコメント文字を入力します。
- [Toggle Column Selection]: テキスト エディターが列選択モードになり、列を切り取り、コピー、削除、貼り付けることができます。
- [Find]: テキスト エディターの一番下に検索バーを表示し、ファイルの検索ができるようになります。
- [Find in Files]: プロジェクト全体でファイルを検索するための [Find in Files] ビューを表示します。
- [Language Templates]: [Language Templates] タブを表示します。多くのよくあるデザイン/制約構造用の Verilog、VHDL、Tcl、XDC のテンプレートが使用できます。
- [Insert Template]: 選択したテンプレートをテキスト ファイルのカーソルの位置に挿入します。このコマンドは、テンプレートを選択している場合にのみ使用できます。
- [Move Caret to Document Start]: カーソルを編集する文書の開始箇所まで移動します。
- [Move Caret to Document End]: カーソルを編集する文書の終了箇所まで移動します。

テンプレートの使用

Vivado IDE には、多くの Verilog、VHDL、XDC 構造用のテンプレートが含まれています。テンプレートを表示するには、Vivado IDE テキスト エディターのツールバーから [Language Templates] をクリックします。[Templates] ビューが Verilog、VHDL、XDC のフォルダー別に表示されます。テンプレートを選択すると、それが [Preview] エリアに開きます (図 3-28)。

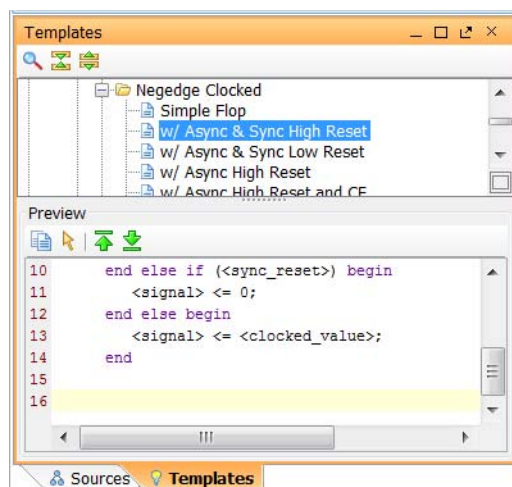


図 3-28: 言語テンプレートのプレビュー

テンプレートを選択すると、テキスト エディターのツールバーの [Insert Template] ボタンが使用できるようになります。これをクリックすると、選択したテンプレートが編集するファイルのカーソルの位置に挿入されます。

検索/置換コマンドの使用

[Find] および [Find in Files] コマンドを使用すると、開いているソース ファイルまたは選択したファイル内で文字列を検索できます。次の操作を実行できます。

- 検索条件として、ワイルドカード (*) を含む任意のテキスト文字列を入力できます。

- フィルター オプションを使用して、ソース ファイル、制約ファイル、レポート ファイルを検索できます。

詳細は、『Vivado Design Suite ユーザー ガイド : Vivado IDE の使用』(UG893) [参照 4] を参照してください。

ソース ファイルへのクロスプローブ

Vivado IDE では、次のビューから RTL ソース ファイルへクロスプローブできます。

- [Schematic] ビュー (エラボレート済み RTL、合成、またはインプリメンテーション)
- [Netlist] ビュー (合成またはインプリメンテーション後)
- [Device] ビュー (インプリメンテーション後)

クロスプローブするには、これらのビューからセルを右クリックし、ポップアップ メニューから [Go To Instantiation]、[Go To Definition] または [Go To Source] をクリックします。RTL ソースが開き、そのインスタンスの行がハイライトされます (図 3-29)。

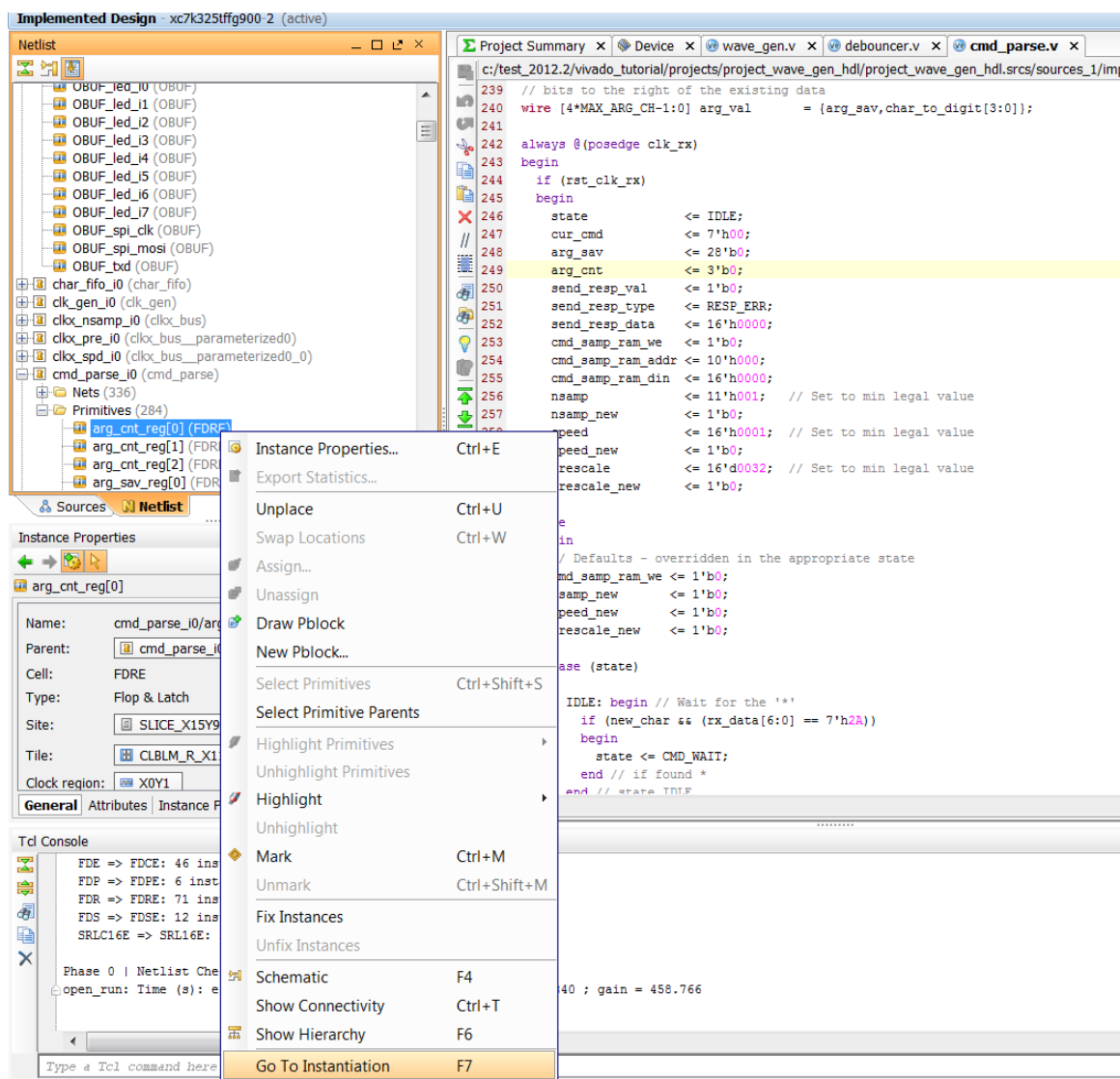


図 3-29 : RTL ソースへのクロスプローブ

その他のテキスト エディターの使用

Vivado IDE では、次を実行すると別のテキスト エディターを使用することもできます。

1. [Tools] → [Options] をクリックします。
2. [Vivado Options] ダイアログ ボックスの [General] ページ (図 3-30) の [Text Editor] セクションまでスクロールダウンし、ドロップダウン リストから別のテキスト エディターを選択します。

リストからテキスト エディターを選択すると、実行ファイル名が表示されます。この実行ファイルへのパスは、ユーザーのパスに含まれている必要があります。ユーザー環境にパスを追加する方法については、Windows または Linux の資料を参照してください。

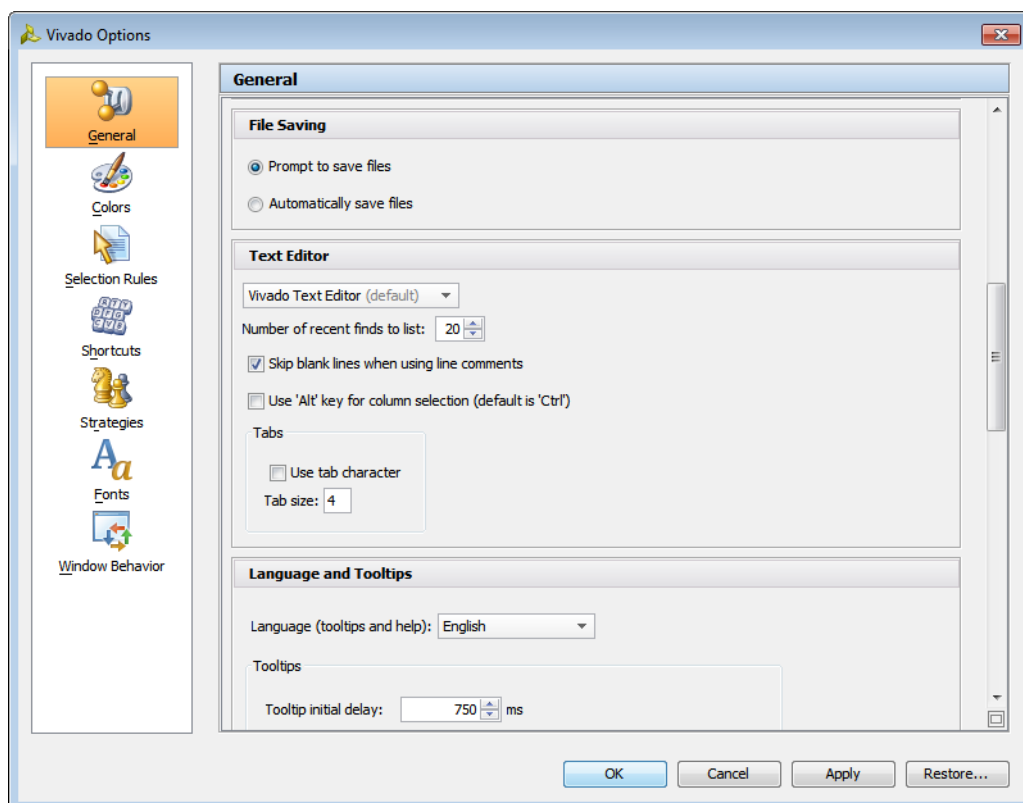


図 3-30 : [Vivado Options] ダイアログ ボックス : [General] ページ

ご希望のテキスト エディターがリストされていない場合は、[Custom Editor] を選択します。[Custom Editor Definition] ダイアログ ボックス (図 3-31) にそのテキスト エディターを実行する実行ファイルの名前またはディレクトリとコマンド ライン構文を入力します。

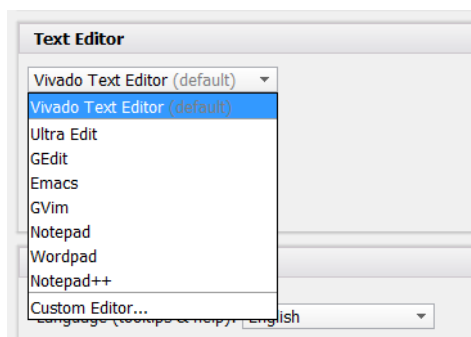


図 3-31: カスタム エディターの設定

注記: 別のテキスト エディターを使用する場合は、クロスプローブは同じようには動作しません。ファイルは外部エディターで開きますが、自動的にその行がハイライト表示されることはありません。

非プロジェクト モードでのソースの操作

非プロジェクト モードのソース ファイルは、ソース ファイルが自動的に管理されるプロジェクト モードと異なり、ユーザーが制御する必要があります。ファイルは、`read_xdc`、`read_verilog`、`read_vhdl`、`read_ip` などのさまざまな Tcl コマンドを使用して直接参照する必要があります。プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite User Guid ユーザー ガイド: デザイン フローの概要』(UG892) [参照 1] を参照してください。Tcl コマンドの詳細については、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) [参照 9] を参照してください。

次は、さまざまなソース ファイルを読み込む非プロジェクト モードのスクリプト例です。

```
# create_bft_batch.tcl
# bft sample design
# A Vivado script that demonstrates a very simple RTL-to-bitstream batch flow
#
# NOTE:typical usage would be "vivado -mode tcl -source create_bft_batch.tcl"
#
# STEP#0: define output directory area.
#
set outputDir ./Tutorial_Created_Data/bft_output
file mkdir $outputDir
#
# STEP#1: setup design sources and constraints
#
read_vhdl -library bftLib [ glob ./Sources/hdl/bftLib/*.vhdl ]
read_vhdl ./Sources/hdl/bft.vhdl
read_verilog [ glob ./Sources/hdl/*.v ]
read_xdc ./Sources/bft_full.xdc
#
# STEP#2: run synthesis, report utilization and timing estimates, write checkpoint
design
#
synth_design -top bft -part xc7k70tfbg484-2 -flatten rebuilt
write_checkpoint -force $outputDir/post_synth
report_timing_summary -file $outputDir/post_synth_timing_summary.rpt
report_power -file $outputDir/post_synth_power.rpt
#
# STEP#3: run placement and logic optimization, report utilization and timing
estimates, write checkpoint design
#
opt_design
power_opt_design
place_design
phys_opt_design
write_checkpoint -force $outputDir/post_place
report_timing_summary -file $outputDir/post_place_timing_summary.rpt
#
# STEP#4: run router, report actual utilization and timing, write checkpoint design,
run drc, write verilog and xdc out
#
route_design
write_checkpoint -force $outputDir/post_route
report_timing_summary -file $outputDir/post_route_timing_summary.rpt
report_timing -sort_by group -max_paths 100 -path_type summary -file
$outputDir/post_route_timing.rpt
report_clock_utilization -file $outputDir/clock_util.rpt
report_utilization -file $outputDir/post_route_util.rpt
report_power -file $outputDir/post_route_power.rpt
report_drc -file $outputDir/post_imp_drc.rpt
write_verilog -force $outputDir/bft_impl_netlist.v
write_xdc -no_fixed_only -force $outputDir/bft_impl.xdc
#
# STEP#5: generate a bitstream
#
write_bitstream -force $outputDir/bft.bit
```


RTL デザインのエラボレーション

概要

Vivado™ IDE には、RTL デザインの解析機能が多く含まれます。たとえば、次を実行できます。

- [Schematic] および [Hierarchy] ビューを使用したデザイン詳細の視覚化
- ビュー間のクロスプローブ
- デザイン ルール チェック (DRC) の実行
- メッセージのチェック
- [Find] コマンドを使用した生成された RTL ネットリストの検索
- RTL レベルでの制約の作成および適用

注記：この段階ではタイミング解析は実行できません。

プロジェクト モードでのデザインのエラボレーション

プロジェクトでイネーブルになっている RTL ソース ファイルは、合成中に自動的にエラボレートされます。ソース ファイルは、制約の開発および RTL ネットリスト エラボレーション用に手動でエラボレートすることもできます。エラボレーションおよびコンパイルに関するメッセージは、[Messages] ビューに表示されます。エラボレーションに使用される HDL 言語は、[Project Settings] ダイアログ ボックスの [General] ページで選択できます。詳細は、[第 2 章「\[General\] ページ」](#)を参照してください。

エラボレーション結果は、デザインと一緒に保存されません。エラボレート済みデザインを開くたびに、エラボレーションが再実行されます。エラボレート済みデザインを合成すると、合成済みデザインとして保存されます。

デザイン ソース ファイルをプロジェクトにインポートしたら、次のいずれかのコマンドを使用してデザインをエラボレートして開きます。

- [Flow] → [Open Elaborated Design] をクリックします。
- Flow Navigator の [RTL Analysis] セクションで [Open Elaborated Design] をクリックすると、エラボレート済みネットリスト、アクティブな制約セットおよびターゲット デバイスがメモリに読み込まれます。

エラボレートするデザイン名を指定するには、次のいずれかの方法を使用します。

- [Flow] → [New Elaborated Design] をクリックします。
- Flow Navigator の [RTL Analysis] ポップアップ メニューから [New Elaborated Design] をクリックします。

エラボレート済みデザインを開くと、RTL ソース ファイルがエラボレートされ、最上位回路図表示が生成され、デフォルトのビューレイアウトでデザインが表示されます。図 4-1 は、エラボレート済みデザインのデフォルトビューレイアウトの [RTL Schematic] ビューを示しています。

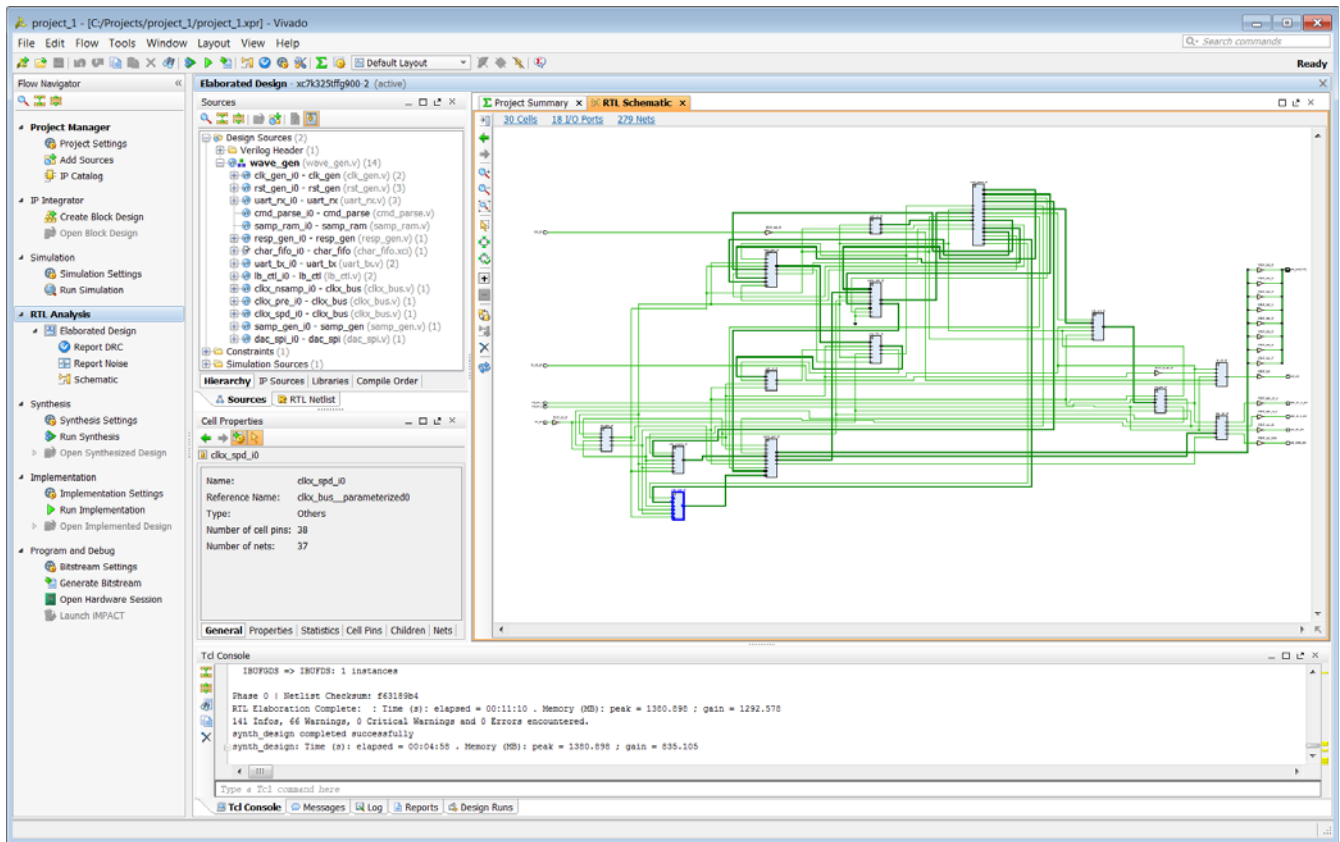



図 4-1 : [RTL Schematic] ビューのエラボレート済みデザイン

ほとんどの場合は、Vivado IDE で自動的に最上位モジュールが特定されます。候補が複数ある場合は、選択することを探るメッセージが表示されます。最上位モジュールは、[Sources] ビューのポップアップメニューから [Set as Top] コマンドを使用して手動で定義することもできます。

注記 : [Sources] ビューの [Hierarchy] タブでは、最上位モジュールが  アイコンで表示されます。

プロジェクト モードでのデザイン エラボレーションに使用する Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド:** synth_design
- **Tcl コマンドの例:** synth_design -rtl -name rtl_1

エラボレーション メッセージの表示

[Messages] ビューにコンパイル結果が表示され、RTL ソース ファイルに問題がある場合は [Elaborated Design] セクションの下に表示されます (図 4-2)。

RTL エラボレーションの結果からは、エラー、警告、情報メッセージの表示/非表示を制御できます。[Messages] ビューの上部にあるチェックボックスのオン/オフを切り替え、エラー、クリティカル警告、警告、情報メッセージを表示/非表示にします。

[Messages] ビューでエラーまたは警告メッセージを選択すると、該当する RTL ソース ファイルが Vivado IDE テキスト エディターに読み込まれ、問題のソース コードがハイライトされます。

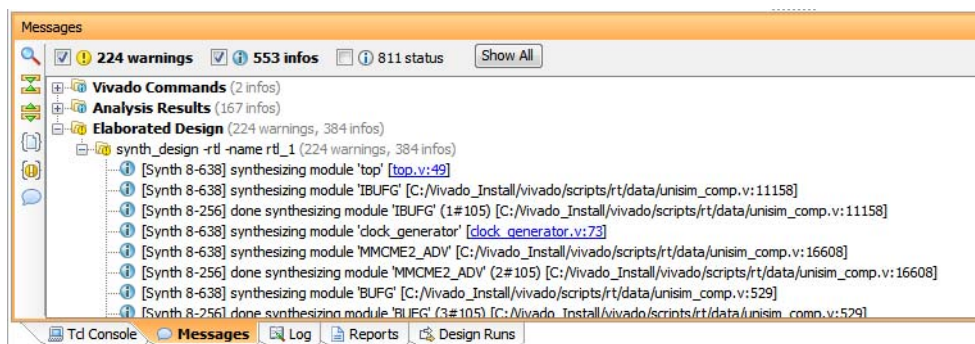


図 4-2 : エラボレートされたデザインのメッセージ

RTL ロジック階層の解析

Vivado IDE には、デザインのロジック階層を表示する複数のビューがあります。

- [RTL Netlist] ビュー : 展開可能なロジック ツリーを表示します。
- [RTL Hierarchy] ビュー : ロジック階層をグラフィカルに表示します。
- [RTL Schematic] ビュー : 回路図表示でロジックおよび階層を調べるのに使用します。

デフォルトでは Flow Navigator で [Elaborate Design] をクリックしてデザインをエラボレートすると、[RTL Schematic] ビューにデザイン全体が表示されます。1 つのビュー選択したオブジェクトはほかのビューでも選択され、ロジック デザインを解析しやすくなっています。詳細は、『Vivado Design Suite ユーザー ガイド : Vivado IDE の使用』(UG893) [参照 4] を参照してください。

エラボレート済みデザイン回路図の解析

[RTL Netlist] ビューで任意のロジック階層を選択し、[RTL Schematic] ビューで表示できます。選択したロジックの [RTL Schematic] ビューを開くには、次のいずれかを実行します。

- [Tools] → [Schematic] をクリックします。
- [RTL Netlist] ビューでポップアップ メニューから [Schematic] をクリックします。

[RTL Schematic] ビューでの操作の詳細は、『Vivado Design Suite ユーザー ガイド : Vivado IDE の使用』(UG893) [参照 4] を参照してください。

注記 : デザインがエラボレートされたら、[Find] コマンドを使用してロジック オブジェクトを検索できます。

[RTL Hierarchy] ビューの使用

Vivado IDE には、デザインの階層を表示するのに便利な [RTL Hierarchy] ビューが含まれます。選択したロジックの [RTL Hierarchy] ビューを開くには、次のいずれかを実行します。

- [Tools] → [Show Hierarchy] をクリックします。
- [RTL Netlist] または [Schematic] ビューでポップアップ メニューから [Show Hierarchy] をクリックします。

これらのビューでは、クロスプローブがサポートされています。[RTL Netlist] または [Schematic] ビューでロジックを選択すると、それが [RTL Hierarchy] ビューでハイライトされます。

RTL ソース ファイルの解析

[RTL Netlist] または [Schematic] ビューでロジック エレメントを選択し、インスタンス化されている RTL ソース ファイルでそのオブジェクトのインスタンス化を開くことができるほか、RTL ファイルのロジックの定義を開くことができます。

選択したロジックのインスタンス化または定義を RTL ソース ファイルで開くには、オブジェクトを右クリックして [Go To Instantiation] または [Go To Definition] をポップアップ メニューからクリックします。ソース ファイルが開き、該当するインスタンスがハイライトされます。

RTL DRC の実行

このセクションでは、Vivado IDE でデザイン ルール チェック (DRC) のルールを選択して DRC 違反を解析する方法について説明します。



推奨 : RTL DRC を実行すると、合成前のエラボレーション段階で早期にデザインの問題を発見できるので、設計全体の時間を節約することができます。

DRC ルールの選択

エラボレート済みデザインで DRC を実行できます。これらの DRC は、消費電力の削減およびパフォーマンスの向上に焦点を当てています。

1. [Tools] → [Run DRC] をクリックします。

注記 : または、Flow Navigator の [RTL Analysis] セクションで [Report DRC] をクリックするか、Tel コンソールに「report_drc」と入力します。

2. [Report DRC] ダイアログ ボックス (図 4-3) で実行するルールを選択し、[OK] をクリックします。

注記 : オプションで、ファイル名を入力して結果をファイルに保存することもできます。デフォルトとは違うパスを選択する場合は、参照ボタンを使用してください。

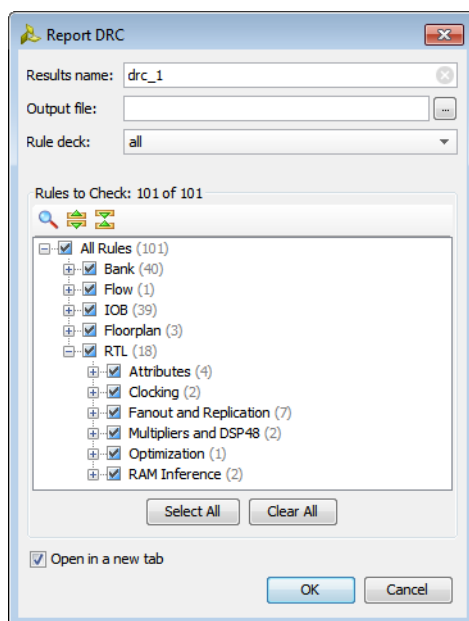


図 4-3 : [Report DRC] ダイアログ ボックス

DRC 違反の解析

DRC で違反が検出された場合、[図 4-4](#) に示す [DRC] ビューが表示されます。[DRC] ビューには、検出されたルール違反が [Run DRC] ダイアログ ボックスで定義された違反カテゴリ別に表示されます。

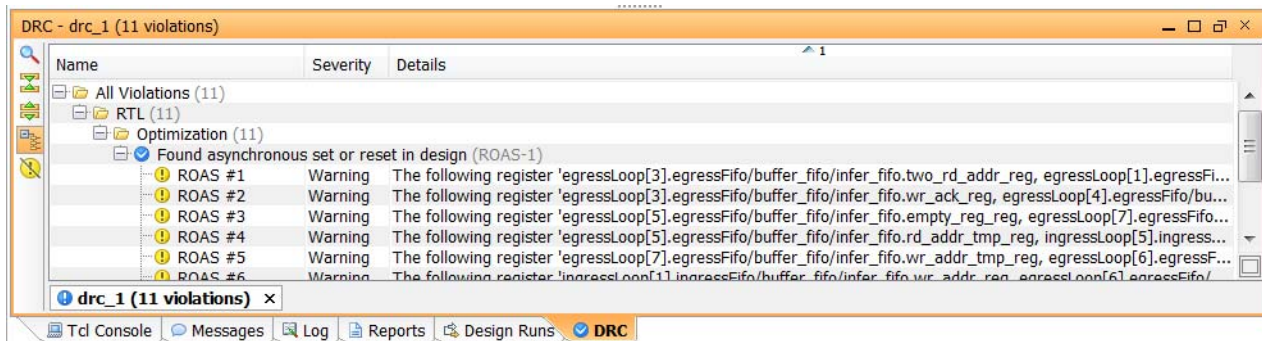






図 4-4: エラボレートされたデザインの DRC 違反を示す [DRC] ビュー

ルール違反はその重要度によっても分類されており、次のように色分け表示されます。

- 情報: 発生する可能性のある問題 
- 警告: 解決する必要がある可能性がある問題 
- エラー: インプリメンテーションの障害となる問題 



ヒント: 警告および情報メッセージをオフにしてレポートされたエラー メッセージのみを確認するには、[Hide Warning and Informational Messages] ツールバー ボタン  をクリックします。

[Severity] 列のヘッダーをクリックすると、違反の重要度で並べ替えることができます。

- 列ヘッダーを一度クリックすると重要度の低い方から並べられます。
- もう 1 回クリックすると降順で並べ替えられます。

注記: 詳細は、『Vivado Design Suite ユーザー ガイド : Vivado IDE の使用』(UG893) [\[参照 4\]](#) を参照してください。

[DRC] ビューで違反メッセージをクリックすると、デフォルトで違反プロパティが [Violation Properties] ビューに表示されます ([図 4-5](#))。[DRC] ビューでポップアップ メニューから [Violation Properties] をクリックしても、[Violation Properties] ビューを開くことができます。

このビューには、DRC ルール違反の概要 ([General] タブ)と、違反しているデザイン エレメントの詳細 ([Details] タブ)が表示されます。[Details] タブには、DRC に違反している特定のデザイン オブジェクトへのリンクが含まれます。リンクをクリックすると、そのデザイン オブジェクトが [RTL Netlist] ビュー、[Device] ビュー、[Schematic] ビュー、およびソース RTL ファイルで選択されます。

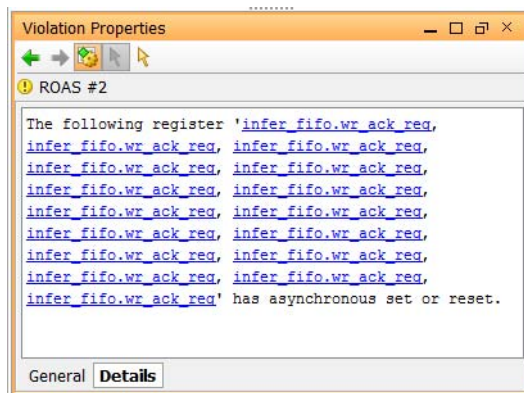


図 4-5 : [Violation Properties] ビュー

RTL DRC を実行する Tcl コマンド

次は、関連する Tcl コマンドです。

- **Tcl コマンド**: report_drc
- **Tcl コマンドの例**: report_drc -name drc_1

注記: デフォルトでは、テキスト ベースのレポートが出力されます。-name オプションを使用すると、そのレポート用にインタラクティブなタブを作成できます。

非プロジェクト モードでのデザインのエラボレーション

非プロジェクト モードでは、RTL のエラボレーションを実行できます。RTL をクロスプロンプし直して、DRC を実行することもできます。クロスプロンプには、Tcl コマンドの start_gui を使用して Vivado IDE を起動する必要があります。DRC は、Vivado IDE の起動の有無に関係なく実行できます。

次は、さまざまなファイルを読み込んで、Tcl コマンドの synth_design を -rtl オプションと共に使用して RTL をエラボレートするスクリプトです。このスクリプトでは、Vivado IDE も起動するので、回路図またはネットリストから RTL ソースにクロスプロンプできます。

注記: 非プロジェクト モードで Vivado IDE を起動する場合は、Flow Navigator はありませんので、[Tools] メニューから Tcl コンソールを使用してタスクを実行する必要があります。

```
# create_bft_batch.tcl
# bft sample design
# A Vivado script that demonstrates a very simple RTL-to-bitstream batch flow
#
# NOTE:typical usage would be "vivado -mode tcl -source create_bft_batch.tcl"
#
# STEP#0: define output directory area.
#
set outputDir ./Tutorial_Created_Data/bft_output
file mkdir $outputDir
#
# STEP#1: setup design sources and constraints
#
```

```
read_vhdl -library bftLib [ glob ./Sources/hdl/bftLib/*.vhdl ]
read_vhdl ./Sources/hdl/bft.vhdl
read_verilog [ glob ./Sources/hdl/*.v ]
read_xdc ./Sources/bft_full.xdc
#
# STEP #2 Elaborate the RTL and start the GUI for interaction
#
synth_design -top bft -part xc7k70tfbg484-2 -rtl
start_gui
# Use stop_gui to quit the GUI and return back to the Vivado IDE Tcl command line
```


デザインのデバッグ

概要

FPGA デザインのデバッグは複数の手順の繰り返しプロセスです。ほとんどの複雑な問題と同様、1 度にデザイン全体を処理するのではなく、FPGA デザインのデバッグ プロセスを小さく分け、小さいセクションごとに 1 つずつ処理していくのが最適な方法です。問題のなかったデザイン例とデバッグ方法をデザイン フローを通して繰り返し使用し、1 つずつモジュールを追加していった、デザイン全体でうまく動作するかどうかその都度確認します。このデザインおよびデバッグ手法は、次のデザイン フロー段階のどの組み合わせでも使用できます。

- RTL レベルのデザイン シミュレーション
- インシステム デバッグ

Set up Debug ウィザードまたは Tcl コマンドを使用すると、デバッグ コアを作成、接続し、合成済みデザイン ネットリストに挿入できます。デバッグの詳細については、『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』(UG908) [\[参照 12\]](#) を参照してください。

RTL レベルのデザイン シミュレーション

シミュレーション検証プロセス中は、デザインを機能的にデバッグできます。Vivado™ シミュレータ には、完全なデザイン シミュレーション機能が含まれます。Vivado シミュレータを使用すると、デザインの RTL シミュレーションを実行できます。RTL レベル シミュレーション環境でデザインをデバッグする利点には、デザイン全体を視覚化できる点やデザインおよびデバッグ サイクルを素早く繰り返すことができる点などがあります。シミュレーションの設定および起動方法については、『Vivado Design Suite ユーザー ガイド : ロジック シミュレーション』(UG900) [\[参照 8\]](#) を参照してください。

インシステム デバッグ

Vivado IDE には、インプリメンテーション後の FPGA デザインをインシステムでデバッグできるロジック解析機能もあります。インシステム デバッグの利点は、実際のシステム環境でシステム速度でインプリメント後のタイミングの正確なデザインをデバッグできる点にあります。インシステム デバッグでは、デザインのサイズおよび複雑さによって、シミュレーション モデルを使用した場合と比べてデバッグ信号の表示精度が落ちたり、潜在的にデザイン、インプリメンテーション、デバッグの繰り返しが長くなることがあります。

Vivado IDE には、デザインをデバッグする方法が複数あります。必要に応じて、これらの方法のいずれかを使用してデザインをデバッグできます。詳細は、『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』(UG908) [\[参照 12\]](#) を参照してください。

その他のリソース

ザイリンクス リソース

アンサー、資料、ダウンロード、フォーラムなどのサポート リソースは、次のザイリンクス サポート サイトを参照してください。

<http://japan.xilinx.com/support>

ザイリンクス資料で使用する用語集は、次を参照してください。

<http://japan.xilinx.com/company/terms.htm>

ソリューション センター

デバイス、ツール、IP のサポートについては、[ザイリンクス ソリューション センター](#)を参照してください。トピックには、デザイン アシスタント、アドバイザリ、トラブルシュート ヒントなどが含まれます。

リファレンス

1. 『Vivado Design Suite ユーザー ガイド : デザイン フローの概要』([UG892](#))
2. 『Vivado Design Suite ユーザー ガイド : Tcl スクリプト機能の使用』([UG894](#))
3. 『Vivado Design Suite チュートリアル : デザイン フローの概要』([UG892](#))
4. 『Vivado Design Suite ユーザー ガイド : Vivado IDE の使用』([UG893](#))
5. 『Vivado Design Suite ユーザー ガイド : 制約の使用』([UG903](#))
6. 『Vivado Design Suite 移行手法ガイド』([UG911](#))
7. 『Vivado Design Suite ユーザー ガイド : I/O およびクロックの配置』([UG899](#))
8. 『Vivado Design Suite ユーザー ガイド : ロジック シミュレーション』([UG900](#))
9. 『Vivado Design Suite Tcl コマンド リファレンス ガイド』([UG835](#))
10. 『Vivado Design Suite ユーザー ガイド : 合成』([UG901](#))
11. 『Vivado Design Suite ユーザー ガイド : インプリメンテーション』([UG904](#))
12. 『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』([UG908](#))
13. 『Vivado Design Suite ユーザー ガイド : IP を使用したデザイン』([UG896](#))
14. 『EDK コンセプト、ツール、テクニック』([UG683](#))

15. 『エンベデッド システム ツール リファレンス マニュアル』([UG111](#))
16. Vivado Design Suite ビデオ チュートリアル (<http://japan.xilinx.com/training/vivado/index.htm>)
17. Vivado Design Suite 資料 (http://japan.xilinx.com/support/documentation/dt_vivado2013-1.htm)