

ISE アドバンス チュートリアル

UG695 (v12.3) 2010 年 9 月 21 日



Xilinx is disclosing this user guide, manual, release note, and/or specification (the "Documentation") to you solely for use in the development of designs to operate with Xilinx hardware devices. You may not reproduce, distribute, republish, download, display, post, or transmit the Documentation in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Xilinx expressly disclaims any liability arising out of your use of the Documentation. Xilinx reserves the right, at its sole discretion, to change the Documentation without notice at any time. Xilinx assumes no obligation to correct any errors contained in the Documentation, or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information.

THE DOCUMENTATION IS DISCLOSED TO YOU "AS-IS" WITH NO WARRANTY OF ANY KIND. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DOCUMENTATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS. IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOSS OF DATA OR LOST PROFITS, ARISING FROM YOUR USE OF THE DOCUMENTATION.

© 2010 Xilinx, Inc. XILINX, the Xilinx logo, Virtex, Spartan, ISE, 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 (v12.3) 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。

目次

このチュートリアルについて

アドバンス チュートリアル概要	9
チュートリアルの内容	9
チュートリアル フロー	10
HDL デザイン フロー	10
回路図デザイン フロー	10
インプリメンテーションのみのフロー	10
その他のリソース	11

第 1 章：ISE ソフトウェアの概要

ソフトウェアの概要	13
Project Navigator のインターフェイス	13
[Design] パネル	14
[View] ペイン	14
[Hierarchy] ペイン	14
[Processes] ペイン	15
[Files] パネル	16
[Libraries] パネル	16
[Console] パネル	16
[Errors] パネル	16
[Warnings] パネル	16
ソース ファイルへのエラー ナビゲーション	16
アンサー データベースへのエラー ナビゲーション	16
ワークスペース	16
デザイン サマリ/レポート ビューア	17
プロジェクトのリビジョン管理機能の使用	17
ISE プロジェクト ファイルの理解	17
プロジェクトのコピーの作成	18
Project Browser の使用	18
プロジェクト アーカイブの使用	18
アーカイブの作成	18
アーカイブの復元	18

第 2 章：HDL ベースのデザイン

HDL ベースのデザインの概要	19
入門	20
必要なソフトウェア	20
その他のソフトウェア (オプション)	20
VHDL または Verilog の使用	20
チュートリアル プロジェクト ファイルのダウンロード	20
ISE ソフトウェアの起動	21
新規プロジェクトの作成	22
チュートリアルの中止	23
デザインの概要	24
入力	24
出力	24
論理ブロック	25
デザイン入力	25
ソース ファイルの追加	25
HDL エラーの修正	26
HDL ベースのモジュールの作成	27

New Source Wizard および ISE Text Editor の使用	27
言語テンプレートの使用	29
言語テンプレートのファイルへの追加	30
CORE Generator ソフトウェア モジュールの作成	31
timer_preset CORE Generator ソフトウェア モジュールの作成	31
HDL コードへの CORE Generator ソフトウェア モジュールの インスタンス化	35
DCM モジュールの作成	36
Clocking Wizard の使用	36
dcm1 マクロのインスタンス化 (VHDL デザイン)	38
dcm1 マクロのインスタンス化 (Verilog)	39
デザインの合成	40
XST を使用した合成	41
合成オプションの入力	41
デザインの合成	41
RTL Viewer/Technology Viewer の使用	42
Synplify/Synplify Pro ソフトウェアを使用したデザインの合成	44
合成オプションの入力とデザインの合成	44
合成結果の確認	44
Precision Synthesis を使用した合成	45
合成オプションの入力とデザインの合成	46
RTL Viewer/Technology Viewer の使用	46

第 3 章：回路図ベースのデザイン

回路図ベースのデザインの概要	47
入門	48
必要なソフトウェア	48
チュートリアル プロジェクト ファイルのダウンロード	48
ISE ソフトウェアの起動	48
新規プロジェクトの作成	49
チュートリアルの中止	49
デザインの概要	49
入力	50
出力	51
論理ブロック	51
デザイン入力	52
ソース ファイルの追加	52
ザイリンクス Schematic Editor で回路図ファイルを開く	53
ウィンドウの表示の変更	53
回路図ベースのマクロの作成	54
time_cnt 回路図の定義	55
I/O マーカーの追加	55
回路図コンポーネントの追加	55
間違った配置の修正	58
ワイヤの描画	58
バスの追加	58
バス タップの追加	59
ネット名の追加	60
回路図のチェック	62
回路図の保存	62
time_cnt のシンボルの作成および配置	62
time_cnt シンボルの作成	62
time_cnt シンボルの配置	63
CORE Generator ソフトウェア モジュールの作成	63
timer_preset CORE Generator ソフトウェア モジュールの作成	63
DCM モジュールの作成	66
Clocking Wizard の使用法	66
dcm1 シンボルの作成	68

HDL ベースのモジュールの作成	68
New Source Wizard および ISE Text Editor の使用	68
言語テンプレートの使用	70
言語テンプレートのファイルへの追加	71
HDL モジュールの回路図シンボルの作成	72
statmach、timer_preset、dcm1、debounce シンボルの配置	73
インスタンス名の変更	75
上位および下位階層への移動	75
デバイスの入力および出力の指定	76
入力ピンの追加	76
I/O マーカーおよびネット名の追加	76
ピン ロケーションの割り当て	77
回路図の完成	79
 第 4 章：ビヘイビア シミュレーション	
ビヘイビア シミュレーション フローの概要	81
ModelSim の設定	81
ISim の設定	82
入門	82
必要なファイル	82
デザイン ファイル (VHDL、Verilog、または回路図)	82
テストベンチ ファイル	82
シミュレーション ライブラリ	82
ザイリンクス シミュレーション ライブラリ	82
ザイリンクス シミュレーション ライブラリのアップデート	83
Modelsim.ini ファイルでのシミュレーション ライブラリのマップ	83
HDL テストベンチの追加	84
チュートリアル of テストベンチ ファイルの追加	84
VHDL シミュレーション	84
Verilog シミュレーション	85
ModelSim を使用したビヘイビア シミュレーション	85
ISE のシミュレーション プロセス	86
シミュレーション プロパティの指定	86
シミュレーションの実行	87
信号の追加	87
仕切りの追加	89
シミュレーションの再実行	90
信号の解析	90
シミュレーションの保存	91
ISim を使用したビヘイビア シミュレーション	91
ISE のシミュレーション プロセス	92
シミュレーション プロパティの指定	92
シミュレーションの実行	93
信号の追加	94
シミュレーションの再実行	95
信号の解析	95
 第 5 章：デザイン インプリメンテーション	
インプリメンテーションの概要	97
入門	98
デザイン入力からの継続	98
デザイン インプリメンテーションからの開始	98
プロパティの設定	99
タイミング制約の作成	100
デザインの変換	101
Constraints Editor の使用	101

PlanAhead ソフトウェアを使用した I/O ロケーションの割り当て	108
デザインのマップ	110
タイミング解析を使用したマップ後のブロック遅延の評価	112
タイミング要件の評価	112
マップ後のスタティック タイミング レポートの表示	113
デザインの配置配線	114
FPGA Editor を使用した配置配線の検証	115
配置配線後のタイミングの評価	117
配置配線後のスタティック タイミング レポートの表示	117
PlanAhead ソフトウェアを使用したデザインの解析	118
コンフィギュレーション データの作成	119
iMPACT を使用した PROM ファイルの生成	120
コマンド ラインを使用したインプリメンテーション	122

第 6 章：タイミング シミュレーション

タイミング シミュレーション フローの概要	123
入門	123
必要なソフトウェア	123
必要なファイル	124
シミュレータの指定	124
ModelSim を使用したタイミング シミュレーション	125
シミュレーション プロパティの指定	125
シミュレーションの実行	127
信号の追加	127
仕切りの追加	129
シミュレーションの再実行	130
信号の解析	130
シミュレーションの保存	131
ザイリンクス ISim を使用したタイミング シミュレーション	132
シミュレーション プロパティの指定	132
シミュレーションの実行	133
信号の追加	133
信号名の表示の切り替え	134
シミュレーションの再実行	135
信号の解析	135

第 7 章：iMPACT チュートリアル

デバイス サポート	137
サポートされるダウンロード ケーブル	137
パラレル ケーブル IV	137
プラットフォーム ケーブル USB	137
プラットフォーム ケーブル USB-II	138
サポートされるコンフィギュレーション モード	138
入門	138
コンフィギュレーション ファイルの生成	138
ケーブルの接続	138
ソフトウェアの起動	139
Project Navigator からの iMPACT の起動	139
スタンドアロンでの iMPACT の起動	139
バウンダリ スキャン コンフィギュレーション モードの使用	139
バウンダリ スキャン コンフィギュレーション モードの指定	139
コンフィギュレーション ファイルの指定	141
プロジェクト ファイルの保存	143
プリファレンスの編集	143
バウンダリ スキャン操作の実行	144

バウンダリ スキャン コンフィギュレーションのトラブルシューティング.....	148
ケーブル接続の確認.....	148
チェーンの設定の検証.....	148
SVF ファイルの作成.....	150
バウンダリ スキャン チェーンの設定.....	150
SVF ファイル生成用の JTAG チェーンの設定.....	150
SVF ファイル生成での JTAG チェーンの手動設定.....	150
SVF ファイルへの書き込み.....	151
SVF ファイルへの書き込み停止.....	153
SVF または XSVF ファイルの再生.....	153

このチュートリアルについて

アドバンス チュートリアルの概要

このチュートリアルでは、ザイリンクス ISE® Design Suite 12 の機能および追加された機能について紹介します。デザイン入力ツール、ザイリンクスおよびサードパーティ ツール、およびデザインインプリメンテーション ツールの関連性に重点を置いて説明していきます。

このチュートリアルは、ISE ソフトウェアの機能に精通していない方や使用方法を復習したい方を対象としています。

このチュートリアルには、複数のデザイン フローが含まれています。各チュートリアルのフローについては、「[チュートリアル フロー](#)」を参照してください。

チュートリアルの内容

このチュートリアルには、次の内容が含まれています。

- 第 1 章「[ISE ソフトウェアの概要](#)」: ISE ソフトウェアのユーザー インターフェイスである Project Navigator および合成ツールを紹介します。
- 第 2 章「[HDL ベースのデザイン](#)」: stopwatch デザインを使用して、HDL に基づく典型的な FPGA デザイン フローを説明します。また、CORE Generator™ ソフトウェアや ISE Text Editor などの ISE ソフトウェアのツールの使用方法も説明します。
- 第 3 章「[回路図ベースのデザイン](#)」: stopwatch デザインを使用して回路図に基づくデザイン フローを説明します。また、CORE Generator ソフトウェアや ISE Text Editor などの ISE ソフトウェアのツールの使用方法も説明します。
- 第 4 章「[ビヘイビア シミュレーション](#)」: インプリメンテーションの前にデザインをシミュレーションし、作成したロジックが正しいかどうかを検証する方法を説明します。
- 第 5 章「[デザイン インプリメンテーション](#)」: 変換、マップ、配置、配線、デザインのビットストリーム ファイルの生成方法を説明します。
- 第 6 章「[タイミング シミュレーション](#)」: 配線済みのデザインからのブロック遅延および配線遅延を使用してタイミング シミュレーションを実行し、ワーストケースでの回路の動作を予測する方法を説明します。
- 第 7 章「[iMPACT チュートリアル](#)」: iMPACT コンフィギュレーション ツールを使用して作成したデザインでデバイスをプログラムする方法を説明します。

チュートリアル フロー

このチュートリアルには、次の3つのチュートリアルフローが含まれています。このセクションでは、各チュートリアルフローの概要を示します。使用するフローによって参照する章は異なります。次がチュートリアルフローです。

- HDL デザイン フロー
- 回路図デザイン フロー
- インプリメンテーションのみのフロー

HDL デザイン フロー

このフローには、次が含まれます。

1. 第2章「[HDL ベースのデザイン](#)」
2. 第4章「[ビヘイビア シミュレーション](#)」

メモ: ビヘイビア シミュレーションはオプションですが、このチュートリアルでは実行することを推奨します。

3. 第5章「[デザイン インプリメンテーション](#)」
4. 第6章「[タイミング シミュレーション](#)」

メモ: タイミング シミュレーションはオプションですが、このチュートリアルでは実行することを推奨します。

5. 第7章「[iMPACT チュートリアル](#)」

回路図デザイン フロー

このフローには、次が含まれます。

1. 第3章「[回路図ベースのデザイン](#)」
2. 第4章「[ビヘイビア シミュレーション](#)」

メモ: ビヘイビア シミュレーションはオプションですが、このチュートリアルでは実行することを推奨します。

3. 第5章「[デザイン インプリメンテーション](#)」
4. 第6章「[タイミング シミュレーション](#)」

メモ: タイミング シミュレーションはオプションですが、このチュートリアルでは実行することを推奨します。

5. 第7章「[iMPACT チュートリアル](#)」

インプリメンテーションのみのフロー

このフローには、次が含まれます。

1. 第5章「[デザイン インプリメンテーション](#)」
2. 第6章「[タイミング シミュレーション](#)」

メモ: タイミング シミュレーションはオプションですが、このチュートリアルでは実行することを推奨します。

3. 第7章「[iMPACT チュートリアル](#)」

その他のリソース

その他の資料は、ザイリンクス Web サイトの[資料](#)ページから参照できます。シリコン、ソフトウェア、IP に関するアンサー データベースを検索したり、テクニカル サポートのウェブケースを開く場合は、ザイリンクス Web サイトの[サポート](#) ページにアクセスしてください。

ISE ソフトウェアの概要

この章は、次のセクションから構成されています。

- 「ソフトウェアの概要」
- 「プロジェクトのリビジョン管理機能の使用」

ソフトウェアの概要

ISE® ソフトウェアは、デザイン フローを最初から最後まで管理する総合デザイン開発環境です。Project Navigator のインターフェイスからは、すべてのデザイン入力およびデザイン インプリメンテーション ツールにアクセスできます。また、プロジェクトに関連するファイルおよびドキュメントにもアクセスできます。

Project Navigator のインターフェイス

デフォルト では Project Navigator のインターフェイスは、図 1-1 に示すように 4 つのパネル サブウィンドウから構成されています。上部左には [Start]、[Design]、[Files]、[Libraries] パネルがあり、プロジェクト内のソース ファイルを表示し、それらのファイルにアクセスできます。また、ソースを選択してプロセスを実行できます。[Start] パネルでは、プロジェクトをすばやく開き、頻繁に使用するリファレンス資料、文書、チュートリアルなどに簡単にアクセスできます。Project Navigator の下部には、[Console]、[Errors]、[Warnings] パネルがあり、ステータス メッセージ、エラー、警告が表示されます。右側にはワークスペースと呼ばれるマルチドキュメント インターフェイス (MDI) ウィンドウがあり、デザイン レポート、テキスト ファイル、回路図、およびシミュレーション 波形などを表示できます。どのウィンドウも、サイズの変更、Project Navigator からの切り離し、Project Navigator のメイン ウィンドウ内で別の場所へ移動、タイル表示、重ねて表示、閉じることが可能です。パネルを開いたり閉じたりするには、[View] → [Panels] メニューコマンドを使用します。デフォルト のウィンドウ レイアウトを復元するには、[Layout] → [Load Default Layout] をクリックします。これらのウィンドウの詳細は、次のセクションで説明します。

次の図に、Project Navigator のインターフェイスを示します。

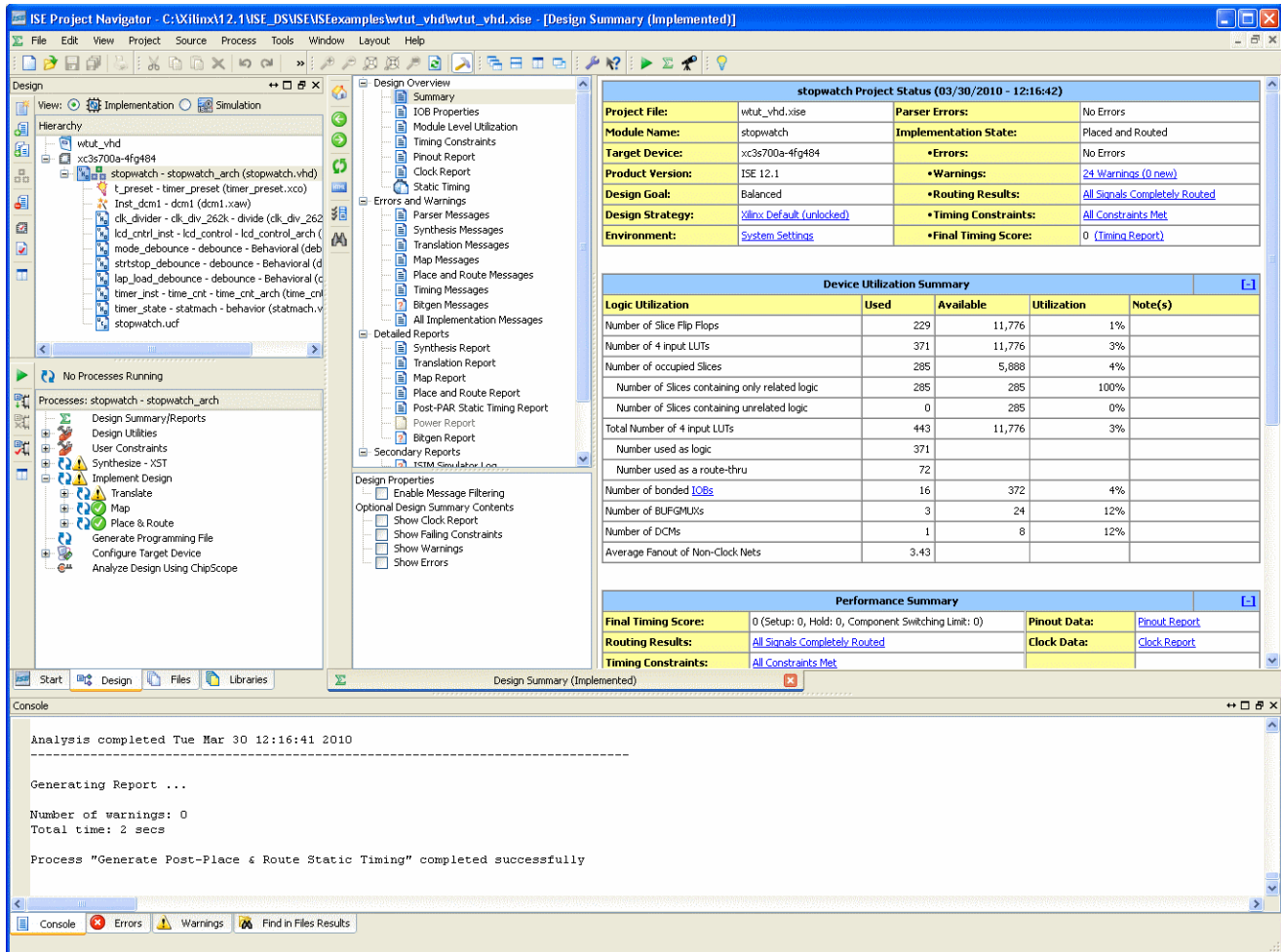


図 1-1 : Project Navigator

[Design] パネル

[Design] パネルには、[View]、[Hierarchy]、[Processes] ペインがあります。

[View] ペイン

ラジオ ボタンにより、インプリメンテーションまたはシミュレーション デザイン ビューに関連付けられているソース モジュールを [Hierarchy] ペインに表示します。[Simulation] をオンにした場合には、ドロップダウン リストからシミュレーション フェーズを選択する必要があります。

[Hierarchy] ペイン

プロジェクト名、ターゲット デバイス、ユーザー ドキュメント、および選択したデザイン ビューに関連付けられているデザイン ソース ファイルを表示します。[Design] パネルの上部にある [View] ペインで [Implementation] または [Simulation] をオンにすると、そのデザイン ビューに関連するソース ファイルのみを表示できます。

[Hierarchy] ペインに表示される各ファイルの横にはアイコンが表示され、ファイルのタイプ (HDL ファイル、回路図、コア、テキスト ファイルなど) が示されます。ソース タイプおよびアイコンの一覧は、ISE ヘルプの「ソース ファイルのタイプ」トピックを参照してください。ISE ヘルプを表示するには、Project Navigator で [Help] → [Help Topics] をクリックします。

ファイルに下位階層が含まれる場合は、ファイル名の左側にプラス記号 (+) が表示されます。このプラス記号をクリックすると、下位階層を表示できます。ファイル名をダブルクリックすると、ファイルを開いて編集できます。

[Processes] ペイン

[Hierarchy] ペインで選択したソース タイプおよびプロジェクトの最上位ソースに関連するプロセスを表示します。このペインからプロセスを実行して、デザインを定義、実行、解析できます。次のプロセスを実行できます。

- [Design Summary/Reports]

デザイン レポート、メッセージ、および結果データのサマリを表示します。メッセージのフィルタ処理も実行できます。

- [Design Utilities]

シンボルの生成、インスタンスエーション テンプレートの表示、コマンド ライン ログ ファイルの表示、およびシミュレーション ライブラリのコンパイルを実行します。

- [User Constraints]

ロケーション制約およびタイミング制約を編集します。

- [Synthesis]

構文チェック、合成、RTL またはテクノロジー回路図の表示、および合成レポートの表示を実行できます。実行できるプロセスは、使用する合成ツールによって異なります。

- [Implement Design]

インプリメンテーション ツールおよびインプリメンテーション後の解析ツールを実行します。

- [Generate Programming File]

ビットストリームを生成します。

- [Configure Target Device]

プログラム ファイルを生成し、デバイスをプログラムするためコンフィギュレーション ツールを実行します。

[Processes] ペインではプロセスの依存関係が認識され、どのプロセスが実行されたか、どのプロセスを実行する必要があるかが管理されます。プロセスの横にはアイコンにより、フローの状態が示されます。フローに含まれるプロセスを選択すると、そのプロセスに到達するために必要なプロセスが自動的に実行されます。たとえば、[Implement Design] プロセスを実行すると、インプリメンテーションは最新の合成結果に依存するので、合成プロセスも実行されます。

プロジェクトに対して実行されたコマンド ライン引数を表示するには、[Design Utilities] → [View Command Line Log File] をダブルクリックします。詳細は、第 5 章の「コマンド ラインを使用したインプリメンテーション」を参照してください。

[Files] パネル

プロジェクト内のソース ファイルを、階層ではなく、並べ替え可能なフラットなリストとして表示します。ファイルは、どの列でも基準にして並べ替えることができます。ファイルを右クリックして [Source Properties] をクリックすると、そのファイルのプロパティを表示できます。

[Libraries] パネル

HDL ライブラリおよびそれに関連付けられている HDL ソース ファイルを管理します。ライブラリとその関連付けられたソースを作成、表示、編集できます。

[Console] パネル

Project Navigator で実行したプロセスの標準出力を表示します。エラー、警告、および情報メッセージを表示します。エラーには赤の X マーク、警告には黄色の感嘆符 (!) がメッセージの横に表示されます。

[Errors] パネル

エラー メッセージのみを表示します。

[Warnings] パネル

警告メッセージのみを表示します。

ソース ファイルへのエラー ナビゲーション

[Console]、[Errors]、または [Warnings] パネルの合成エラーまたは警告メッセージから、ソース HDL ファイル内の問題の箇所にナビゲートできます。エラーまたは警告メッセージを右クリックして [Go to Source] をクリックすると、HDL ソース ファイルが開き、問題のある箇所にカーソルが移動します。

アンサー データベースへのエラー ナビゲーション

[Console]、[Errors]、または [Warnings] パネルのエラーまたは警告メッセージから、ザイリンクスの Web サイトの[サポート ページ](#)にある関連のアンサー データベースにアクセスできます。エラーまたは警告メッセージを右クリックして [Go to Answer Record] をクリックすると、このメッセージに関連するアンサー データベースのリストがデフォルトの Web ブラウザに表示されます。

ワークスペース

ワークスペースでは、デザイン エディタ、ビューア、および解析ツールが開きます。ワークスペースで開くツールには、ISE Text Editor、Schematic Editor、Constraint Editor、デザイン サマリ/レポート ビューア、RTL Viewer、Technology Viewer、Timing Analyzer があります。

I/O 配置およびフロアプランに使用する PlanAhead™ ソフトウェア、ISim、サードパーティ テキスト エディタ、XPower Analyzer、iMPACT などのツールは、Project Navigator のメイン ウィンドウ外に別ウィンドウで開きます。

デザイン サマリ/レポート ビューア

デザイン サマリでは、主要なデザイン データのサマリに加え、合成ツールおよびインプリメンテーション ツールからのメッセージや詳細レポートを表示できます。プロジェクトの情報、デバイス使用率、配置配線 (PAR) レポートのパフォーマンス データ、制約に関する情報、およびすべてのレポートからのサマリ情報 (各レポートへのリンク付き) が表示されます。システム設定レポートへのリンクをクリックすると、デザイン インプリメンテーションで使用された環境変数およびツール設定に関する情報が表示されます。メッセージ フィルタ、タグ付け、インクリメンタル メッセージなどのメッセージ機能も、このビューから利用できます。

プロジェクトのリビジョン管理機能の使用

Project Navigator ではプロジェクトを次のように管理できます。

ISE プロジェクト ファイルの理解

ISE プロジェクト ファイル (拡張子 `.xise`) は、プロジェクトのソースに関連するデータすべてを含む XML ファイルで、次の情報が含まれます。

- ISE ソフトウェア バージョン情報
- プロジェクトに含まれるソース ファイルのリスト
- デザイン プロパティおよびプロセス プロパティを含むソース設定

ISE プロジェクト ファイルに次のものは含まれません。

- プロセス ステータス情報
- コマンド履歴
- 制約データ

メモ： プロセス ステータスなどの生成されたデータを含む **GISE** ファイルも存在しますが、ユーザーがこのファイルを直接使用する必要はありません。

ISE プロジェクト ファイルは、ソース制御環境と互換する次のような特徴があります。

- プロジェクトに必要なソース設定および入力データをすべて含みます。
- Project Navigator 内で読み込み専用ファイルとして開くことができます。
- ソース レベルでの変更がプロジェクトに加えられた場合にのみ、アップデートまたは変更されます。
- 生成される出力ディレクトリ (作業ディレクトリ) とは別のディレクトリに保管できます。

メモ： ソース レベルの変更とは、プロパティの変更、ソース ファイルの追加や削除を指します。ソース ファイルの内容の変更、インプリメンテーション実行 の状態の変更などは、ソースレベルの変更とはみなされず、プロジェクト ファイルは変更されません。

プロジェクトのコピーの作成

[File] → [Copy Project] をクリックしてプロジェクトのコピーを作成すると、異なるソース オプションやインプリメンテーションを試すことができます。必要に応じて、コピーしたプロジェクトのデザイン ソース ファイルとその格納場所は、次のようにできます。

- デザイン ソース ファイルは既存の場所にそのまま残し、コピーしたプロジェクトでそれらのファイルを指定する。
- デザイン ソース ファイルと生成されたファイルをコピーし、指定ディレクトリに保存する。
- デザイン ソース ファイルのみをコピーし、指定ディレクトリにコピーし保存する。

Project Browser の使用

Project Browser は、[File] → [Project Browser] をクリックすると開き、次のようにプロジェクトを比較、表示、開くことができます。

- 複数のプロジェクトの主要な特性を比較します。
- プロジェクトを開く前に、選択したプロジェクトのデザイン サマリやレポートを表示します。
- 選択した 2 つのプロジェクトの詳細情報を比較します。
- 現在の Project Navigator セッションで選択したプロジェクトを開きます。
- 新規の Project Navigator セッションで選択したプロジェクトを開きます。

プロジェクト アーカイブの使用

プロジェクト全体を 1 つのファイルに圧縮し、アーカイブを作成できます。アーカイブを作成すると、プロジェクトを電子メールで送信しやすくなり、限られたスペースに多数のプロジェクトを保存できるようになります。

アーカイブの作成

アーカイブを作成するには、次の手順に従います。

1. [Project] → [Archive] をクリックします。
2. [Project Archive] ダイアログ ボックスで、アーカイブ名と保存場所を入力します。
3. [保存] をクリックします。

メモ：アーカイブには、プロジェクト ディレクトリに含まれるすべてのファイルおよびプロジェクト設定が含まれています。リモート ソースは、アーカイブ内の `remote_sources` というフォルダにそのコピーが保存されます。詳細は、ISE ヘルプを参照してください。

アーカイブの復元

アーカイブしたファイルは、直接 Project Navigator に復元することはできません。圧縮されたファイルを任意の圧縮/解凍ユーティリティで解凍すると、Project Navigator で開くことができます。

HDL ベースのデザイン

この章は、次のセクションから構成されています。

- 「[HDL ベースのデザインの概要](#)」
- 「[入門](#)」
- 「[デザインの概要](#)」
- 「[デザイン入力](#)」
- 「[デザインの合成](#)」

HDL ベースのデザインの概要

この章では、stopwatch デザインを使用して、HDL に基づく典型的な FPGA デザインの手順を説明します。このチュートリアルで使用されるデザイン例では、今後のデザイン作成に応用可能なデバイスのさまざまな機能、ソフトウェアの機能、およびデザイン フローが示されています。このデザインには Spartan™-3A デバイスが使用されていますが、ここで説明する原則およびフローは、注記がない限りすべてのザイリンクス デバイス ファミリーに応用できます。

このデザインは、HDL エlement および 2 つのコアで構成されており、XST (Xilinx Synthesis Technology)、Synplify/Synplify Pro、または Precision Synthesis ソフトウェアを使用して合成できます。

この章は、「[HDL デザイン フロー](#)」の第 1 ステップです。この章で正しくデザインを定義した後に、[ビヘイビア シミュレーション \(第 4 章「ビヘイビア シミュレーション」\)](#)、[ザイリンクス インプリメンテーション ツールを使用したインプリメンテーション \(第 5 章「デザイン インプリメンテーション」\)](#)、[タイミング シミュレーション \(第 6 章「タイミング シミュレーション」\)](#)、および Spartan-3A デバイス (XC3S700A) デモ ボードへのコンフィギュレーションおよびダウンロード ([第 7 章「iMPACT チュートリアル」](#)) を実行します。

入門

次に、このチュートリアルを実行するのに必要な条件を示します。

必要なソフトウェア

このチュートリアルを実行するには、ザイリンクス ISE® Design Suite 12 がインストールされている必要があります。

このチュートリアルでは、ソフトウェアがデフォルトのディレクトリ `c:\xilinx\12.1\ISE_DS\ISE` にインストールされていることを前提としています。ソフトウェアを別のディレクトリにインストールしている場合は、そのインストールパスと置き換えてこのチュートリアルを実行してください。

メモ： ソフトウェアのインストール方法に関する詳細は、ザイリンクス Web サイトから [『ISE Design Suite 12：インストール、ライセンス、リリースノート』](#) を参照してください。

その他のソフトウェア (オプション)

次のサードパーティの合成ツールを XST の代わりに使用できます。

- Synopsys 社 Synplify/Synplify PRO D-2010.03 以降
- Mentor Graphics 社 Precision Synthesis 2010a 以降

次のサードパーティのシミュレーション ツールを、ISim の代わりに使用できます。

- ModelSim SE/PE/DE 6.5c 以降

VHDL または Verilog の使用

VHDL および Verilog デザインの両方がサポートされており、いずれも同等に使用できます。ただし、このチュートリアルで使用する HDL 言語を選択して、その言語に必要なファイルをダウンロードする必要があります。XST では混合言語のデザインも合成できますが、このチュートリアルでは使用しません。

チュートリアル プロジェクト ファイルのダウンロード

チュートリアル プロジェクト ファイルは、ザイリンクスの Web サイトの ISE Design Suite 12 [チュートリアル ページ](#) からダウンロードできます。VHDL または Verilog のいずれかのプロジェクト ファイルをダウンロードします。

チュートリアル プロジェクト ファイルをダウンロードしたら、そのファイルを `c:\xilinx\12.1\ISE_DS\ISE\ISEexamples` ディレクトリに解凍します。既存のファイルがある場合は、すべて置き換えます。

プロジェクト ファイルを `c:\xilinx\12.1\ISE_DS\ISE\ISEexamples` に解凍すると、`wtut_vhd` (VHDL デザイン フロー) または `wtut_ver` (Verilog デザイン フロー) が `c:\xilinx\12.1\ISE\ISEexamples` ディレクトリ内に作成され、チュートリアル ファイルがこのディレクトリにコピーされます。

次の表に、チュートリアルプロジェクトのディレクトリを示します。

表 2-1 : チュートリアルのディレクトリ

ディレクトリ	説明
wtut_vhd	未完成の VHDL ソース ファイル
wtut_ver	未完成の Verilog ソース ファイル
wtut_vhd\wtut_vhd_completed	完成した VHDL ソース ファイル
wtut_ver\wtut_ver_completed	完成した Verilog ソース ファイル

メモ : completed ディレクトリには、完成した HDL ソース ファイルが含まれます。completed ディレクトリに含まれるファイルは上書きしないでください。

このチュートリアルでは、c:\xilinx\12.1\ISE_DS\ISE\ISEexamples でファイルが解凍されており、読み出し/書き込みの権限があることを前提としています。別のディレクトリにファイルを解凍した場合は、そのプロジェクト パスと置き換えてチュートリアルを実行してください。

ISE ソフトウェアの起動

ISE ソフトウェアを起動するには、デスクトップの [Xilinx ISE 12.1] アイコンをダブルクリックするか、[スタート] → [すべてのプログラム] → [Xilinx ISE Design Suite 12.1] → [ISE デザイン ツール] → [Project Navigator] をクリックします。



図 2-1 : デスクトップの [Xilinx ISE 12.1] アイコン

新規プロジェクトの作成

New Project Wizard を使用して新規プロジェクトを作成するには、次の手順に従います。

1. Project Navigator で、[File] → [New Project] をクリックします。
New Project Wizard の [Create New Project] ページが表示されます。

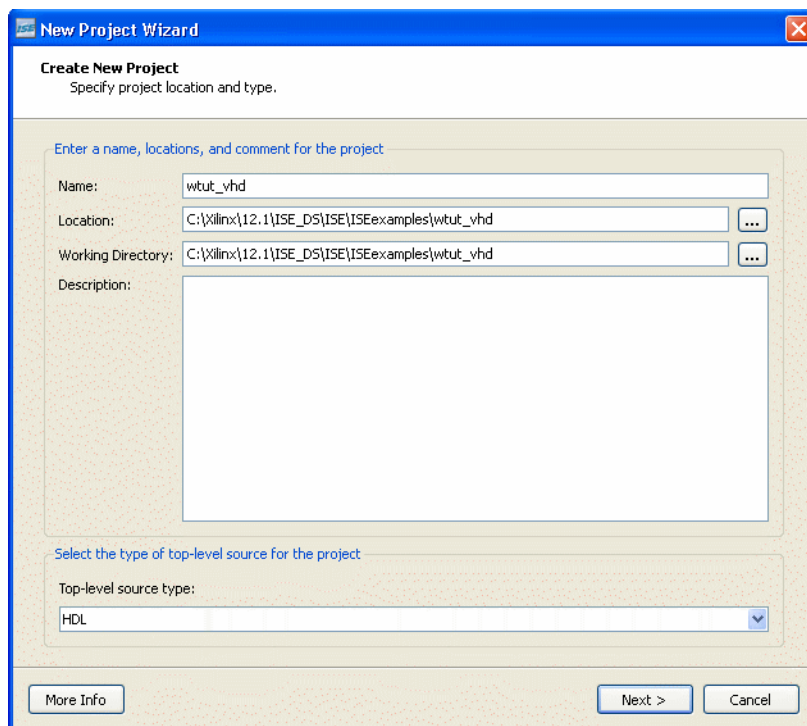


図 2-2 : New Project Wizard の [Create New Project] ページ

2. [Location] で、c:\xilinx\12.1\ISE\ISEexamples またはプロジェクトを配置したディレクトリを選択します。
3. [Name] に「wtut_vhd」または「wtut_ver」と入力します。
4. [Top-level source type] が [HDL] であることを確認し、[Next] をクリックします。

[Project Settings] ページが表示されます。

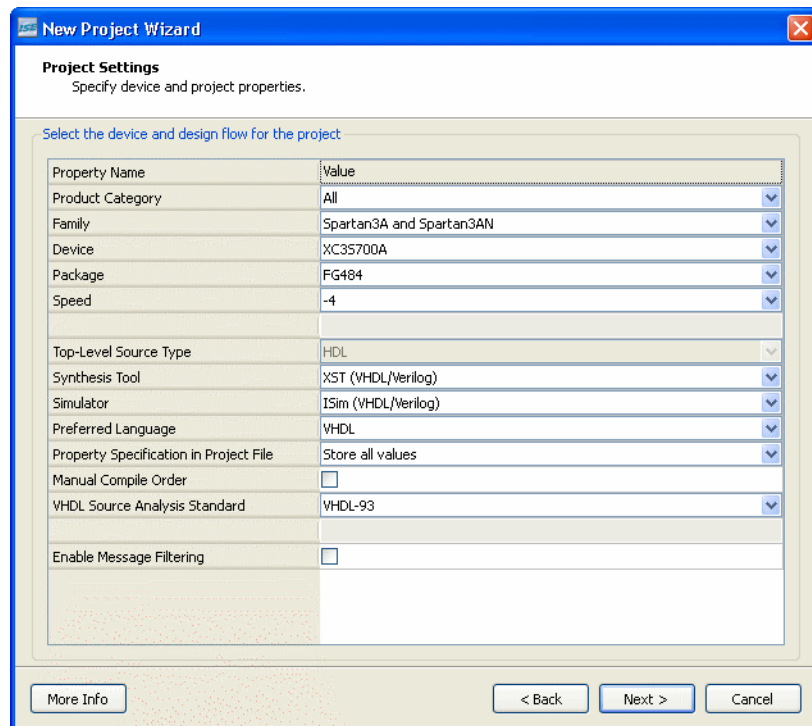


図 2-3 : New Project Wizard の [Project Settings] ページ

5. [Project Settings] ページで、次のように値を設定します。

- ◆ [Product Category] : [All]
- ◆ [Family] : [Spartan3A and Spartan3AN]
- ◆ [Device] : [XC3S700A]
- ◆ [Package] : [FG484]
- ◆ [Speed] : [-4]
- ◆ [Synthesis Tool] : [XST (VHDL/Verilog)]
- ◆ [Simulator] : [ISim (VHDL/Verilog)]
- ◆ [Preferred Language] : [VHDL] または [Verilog]
ここで選択した言語が、HDL ファイルを生成するすべてのプロセスのデフォルト言語になります。

その他のプロパティはデフォルト値のままにします。

6. [Next] をクリックして [Project Summary] ページで [Finish] をクリックし、プロジェクトの作成を完了します。

チュートリアルの中止

チュートリアルを中止する場合は、[File] → [Save All] をクリックして作業中のファイルを保存してください。

デザインの概要

このチュートリアルに使用されるデザインは、**HDL** に基づく階層デザインで、最上位デザイン ファイルから複数の下位マクロが参照されます。下位のマクロは、**HDL** モジュールまたは **IP** モジュールです。

stopwatch デザインは、未完成のデザインです。このチュートリアルでは、モジュールを作成したり、既存のファイルを編集したりして、デザインを完成させます。デザインを完成させたら、シミュレーションを実行してその機能を検証します。

このデザインには、5 つの外部入力および 4 つの外部出力バスが含まれています。システム クロックは、外部で生成される信号です。次に、デザインの入力信号および出力信号のサマリを示します。

入力

次は、**stopwatch** デザインの入力信号です。

- **strtstop**
ストップウォッチを開始、停止します。アクティブ **Low** 信号で、ストップウォッチの開始/停止ボタンと同じ動作をします。
- **reset**
ストップウォッチをクロック モードに切り替え、時間を **0:00:00** にリセットします。
- **clk**
外部で生成されるシステム クロックです。
- **mode**
クロック モードとタイマー モードを切り替えます。クロックまたはタイマーが動作していないときにのみ有効になります。
- **lap_load**
2 つの機能を持つ信号で、クロック モードでは **Lap** ディスプレイに現在のクロック値を表示し、タイマー モードではタイマーが動作していないときに **ROM** からプリセット値をタイマー ディスプレイに読み込みます。

出力

次は、デザインの出力信号です。

- **lcd_e**、**lcd_rs**、**lcd_rw**
ストップウォッチの時間を表示するために使用される **Spartan-3A** デモ ボードの **LCD** ディスプレイを制御する信号です。
- **sf_d[7:0]**
LCD ディスプレイにデータ値を供給します。

論理ブロック

完成デザインは、次の論理ブロックから構成されています。

- **clk_div_262k**
クロック周波数を 262,144 で分周するマクロです。26.2144MHz のクロックをデューティ サイクルが 50% で 100Hz のクロックに変換します。
- **dcm1**
内部フィードバック、周波数を調整した出力、およびデューティ サイクル調整を含む **Clocking Wizard** マクロです。CLKFX_OUT 出力は、Spartan-3A デモ ボードの 50MHz を 26.2144MHz に変換します。
- **debounce**
strtstop、mode、lap_load 入力信号のデバウンス回路をインプリメントする回路図モジュールです。
- **lcd_control**
LCD ディスプレイの初期化と出力を制御するモジュールです。
- **statmach**
ストップウォッチの状態を制御するステート マシン HDL モジュールです。
- **timer_preset**
CORE Generator™ ソフトウェアの 64X20 ROM です。このマクロには、タイマーに読み込む 0:00:00 ~ 9:59:99 の 64 個のプリセット時間が含まれています。
- **time_cnt**
0:00:00 ~ 9:59:99 の 10 進値をカウントするアップ/ダウン カウンタ モジュールで、ストップウォッチの時間の各桁を表す 4 ビットの出力が 5 つあります。

デザイン入力

この階層デザインでは、HDL ファイルの確認、構文エラーの修正、HDL マクロの作成、CORE Generator ソフトウェア コアおよびクロッキング モジュールの追加を実行します。各タイプのデザイン マクロを作成および使用します。このチュートリアルで使用する手順は、今後作成するデザインに応用できます。

ソース ファイルの追加

デザインを合成する前に、HDL ファイルをプロジェクトに追加する必要があります。プロジェクトに 5 つのソース ファイルを次のように追加します。

1. [Project] → [Add Source] をクリックします。
2. プロジェクト ディレクトリから次のファイル (VHDL デザインの場合は拡張子は .vhd、Verilog デザインの場合は拡張子は .v) を選択し、[開く] をクリックします。
 - ◆ clk_div_262k
 - ◆ lcd_control
 - ◆ statmach
 - ◆ stopwatch
 - ◆ time_cnt

3. [Adding Source Files] ダイアログ ボックスで、ファイルの [Association] が [All]、[Library] が [work] に指定されていることを確認し、[OK] をクリックします。

[Design] パネルの [Hierarchy] ペインに、プロジェクトに現在追加されているすべてのソース ファイルが、エンティティ名またはモジュール名と共に表示されます。各ソース デザイン ユニットは、[Hierarchy] ペインに次のような形式で表示されます。

インスタンス名 - エンティティ名 - アーキテクチャ名 - (ファイル名)

インスタンス化されたコンポーネントにエンティティまたはモジュール宣言がない場合は、クエスチョン マーク (?) がコンポーネントの横に表示されます。

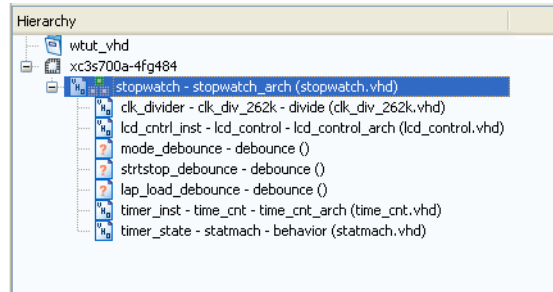


図 2-4 : stopwatch デザインを表示した [Hierarchy] ペイン

HDL エラーの修正

ファイルの構文が正しいかどうかは、ファイルをプロジェクトに追加したとき、およびファイルを保存したときに確認されます。各ファイルの構文が解析され、その結果が [Console] パネルおよび [Design Summary] の [Parser Messages] に表示されます。

`time_cnt` モジュールには、修正を必要とする構文エラーが含まれています。[Console] パネルに「ERROR」メッセージが表示され、その概要と構文に問題がある行番号が示されます。

ソース ファイルのエラーを表示するには、次の手順に従います。

1. [Console] または [Errors] パネルでエラー メッセージのファイル名をクリックします。

ワークスペースにソース コードが表示され、エラーを含む行の横に黄色の矢印アイコンが表示されます。

2. HDL ソースのエラーを修正します。エラー行の上のコメント行に、このエラーの修正方法が示されています。

3. [File] → [Save] をクリックしてファイルを保存します。

[Console] パネルの構文解析メッセージに、解析でエラーが検出されなかったことが示されます。

HDL ベースのモジュールの作成

次に、HDL コードからモジュールを作成します。ISE ソフトウェアでは、ISE Text Editor を使用して HDL コードから簡単にモジュールを作成できます。この HDL コードをインスタンス化して最上位の HDL デザインと連結させ、残りのデザインと共にコンパイルします。

このセクションでは、新規 HDL モジュールを作成します。このマクロは、`strtstop`、`mode`、および `lap_load` 入力をデバウンスするために使用します。

New Source Wizard および ISE Text Editor の使用

New Source Wizard を使用してコンポーネントの名前およびポートを指定し、新規ソース ファイルを作成した後、作成された HDL ファイルを ISE Text Editor で編集します。

ソース ファイルを作成するには、次の手順に従います。

1. [Project] → [New Source] をクリックします。
New Source Wizard の [Select Source Type] ページが表示されます。
2. [Select Source Type] ページで、[VHDL Module] または [Verilog Module] を選択します。
3. [File name] に「debounce」と入力します。

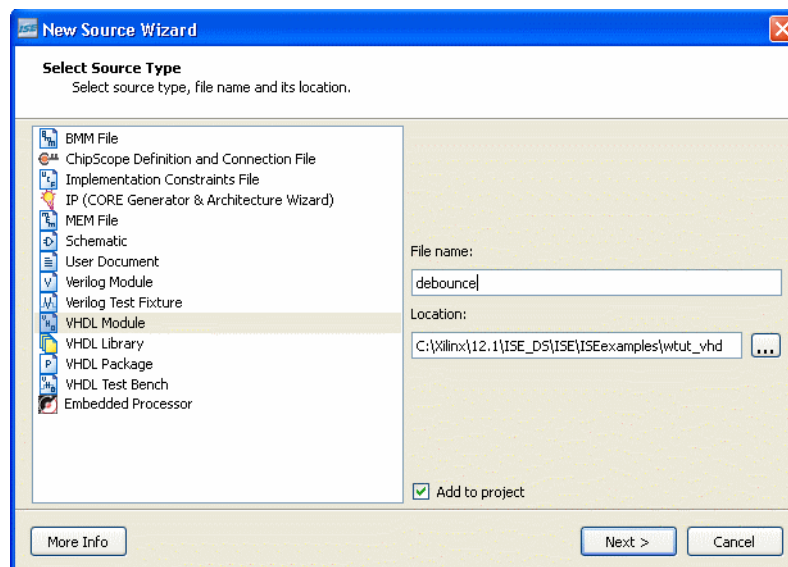


図 2-5 : New Source Wizard の [Select Source Type] ページ

4. [Next] をクリックします。
5. [Define Module] ページで、次の手順に従って `debounce` コンポーネントの 2 つの入力ポート `sig_in` および `clk` と、1 つの出力ポート `sig_out` を入力します。
 - a. [Port Name] 列の最初の 3 つに「`sig_in`」、「`clk`」、および「`sig_out`」と入力します。
 - b. [Direction] 列で `sig_in` および `clk` には [in] または [input]、`sig_out` には [out] または [output] を選択します。

- c. [Bus] 列のチェック ボックスはオフのままにします。

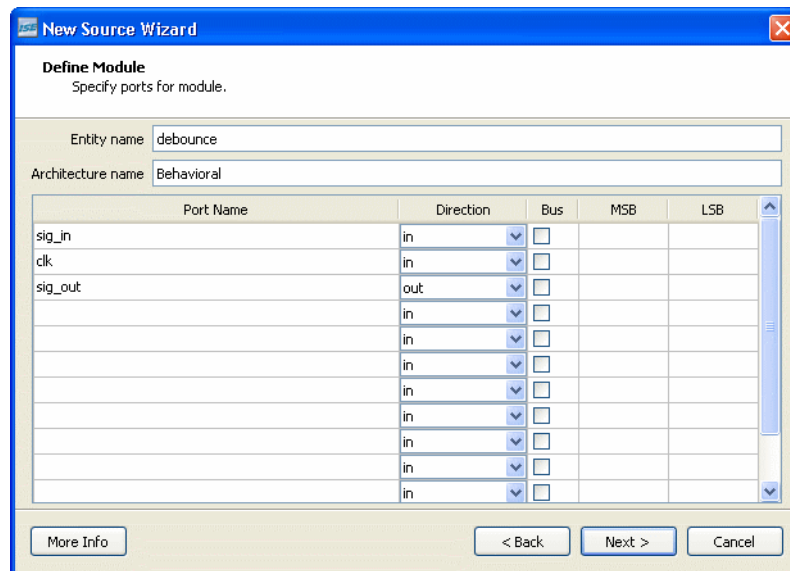


図 2-6 : New Source Wizard の [Define Module] ページ

6. [Next] をクリックします。[Summary] ページにモジュールの説明が表示されます。
7. [Finish] をクリックします。ISE Text Editor で空の HDL ファイルが開きます。

次の図に、VHDL ファイルを示します。

```

7  -- Module Name:      debounce - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity debounce is
31     Port ( sig_in : in  STD_LOGIC;
32           clk : in  STD_LOGIC;
33           sig_out : out STD_LOGIC);
34 end debounce;
35
36 architecture Behavioral of debounce is
37
38 begin
39
40
41 end Behavioral;
42

```

図 2-7 : ISE Text Editor に表示された VHDL ファイル

次の図に、Verilog ファイルを示します。

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    14:12:53 03/15/2007
7  // Design Name:
8  // Module Name:    debounce
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21 module debounce(sig_in, clk, sig_out);
22     input sig_in;
23     input clk;
24     output sig_out;
25
26
27 endmodule
28
```

図 2-8 : ISE Text Editor に表示された Verilog ファイル

ISE Text Editor に表示される HDL ファイルでは、既にポートが宣言されており、基本ファイル構造の一部が記述されています。キーワードは青色、データタイプは赤色、コメントは緑色、値は黒色で表示されています。このようにコードが色分けして表示されるので、コードが読みやすくなり、誤字エラーを見つけやすくなります。

言語テンプレートの使用

ISE の言語テンプレートには、カウンタ、D フリップフロップ、マルチプレクサ、プリミティブなど、よく使用されるロジック コンポーネントを記述した HDL 構文および合成テンプレートが含まれています。このセクションでは、デバウンス回路のテンプレートを使用します。

メモ：よく使用するコンポーネントまたは構文を言語テンプレートに追加することもできます。

言語テンプレートのウィンドウを表示して、このチュートリアルで使用するテンプレートを選択するには、次の手順に従います。

1. [Edit] → [Language Templates] をクリックします。

VHDL および Verilog の言語テンプレートは、それぞれ [Common Constructs]、[Device Macro Instantiation]、[Device Primitive Instantiation]、[Simulation Constructs]、[Synthesis Constructs]、[User Templates] のカテゴリに分類されています。これらのカテゴリを展開するには、プラス記号 (+) をクリックします。テンプレートをクリックすると、右側にテンプレートが表示されます。

2. [VHDL] または [Verilog] の階層から [Synthesis Constructs] → [Coding Examples] → [Misc] を展開し、[Debounce circuit] (VHDL) または [One Shot, Debounce Circuit] (Verilog) テンプレートを選択します。テンプレートが使用している言語のものであることを確認してください。

テンプレートを選択すると、右側にデバウンス回路の HDL コードが表示されます。

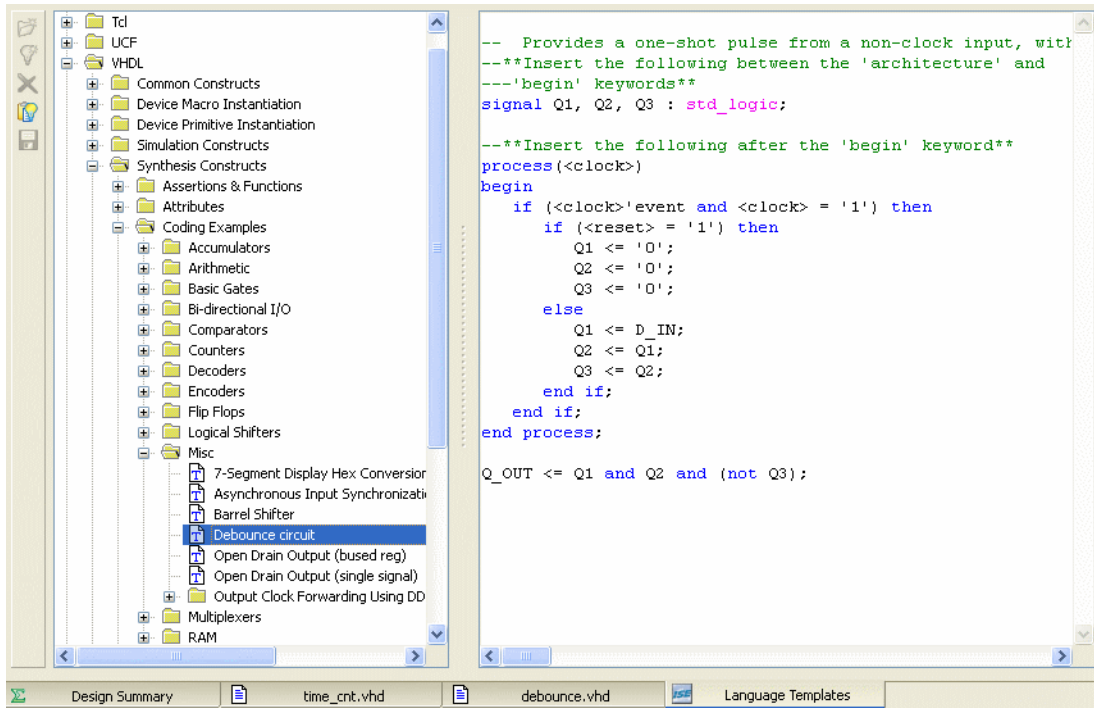


図 2-9：言語テンプレート

言語テンプレートのファイルへの追加

次に、[Use in File] コマンドを使用して、テンプレートを HDL ファイルに追加します。ドラッグアンドドロップで追加する方法など、言語テンプレートに関する詳細は、ISE ヘルプの「言語テンプレートの使用」を参照してください。

テンプレートを HDL ファイルに追加するには、次の手順に従います。

1. debounce.v または debounce.vhd ソース ファイルをアクティブにし、カーソルを VHDL ファイルでは architecture の begin 文の下、Verilog ファイルでは module およびピン宣言の下に置きます。
2. [Language Templates] ウィンドウに戻り、テンプレート インデックスの [Debounce circuit] テンプレートを右クリックし、[Use in File] をクリックします。

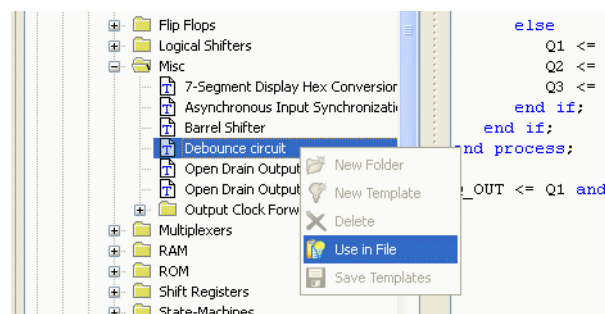


図 2-10：テンプレートを [Use in File] を使用して HDL ファイルに追加

3. [Language Templates] ウィンドウを閉じます。

4. debounce.v または debounce.vhd ソース ファイルを開き、言語テンプレート が正しく挿入されていることを確認します。
5. Verilog のみ : 次の変更を加えて Verilog モジュールを完成させます。
 - a. 「if」から「else」までの 3 行を削除して、リセット ロジックを削除します。リセット ロジックは、このデザインでは使用されません。
 - b. <reg_name> を 6箇所すべてで q に変更します。
 - c. <clock> を clk に、<input> を sig_in に、<output> を sig_out に変更します。

メモ : [Edit] → [Find & Replace] をクリックすると、これを簡単に実行できます。ISE Text Editor の下部に検索フィールドが表示されます。
6. VHDL のみ : 次の変更を加えて、VHDL モジュールを完成させます。
 - a. 「signal」で始まる行を architecture 行と begin 行の間に移動します。
 - b. 「if (<reset>...)」から「else」までの 5 行を削除して、リセット ロジックを削除します。2つある「end if;」行の 1 つも削除します。
 - c. <clock> を clk に、D_IN を sig_in に、Q_OUT を sig_out に変更します。

メモ : [Edit] → [Find & Replace] をクリックすると、これを簡単に実行できます。ISE Text Editor の下部に検索フィールドが表示されます。
7. [File] → [Save] をクリックしてファイルを保存します。
8. [Hierarchy] ペインの debounce インスタンスの 1 つを選択します。
9. [Processes] ペインで [Check Syntax] をダブルクリックし、構文にエラーがないかどうかを確認します。必要に応じてエラーを修正します。
10. ISE Text Editor を閉じます。

CORE Generator ソフトウェア モジュールの作成

CORE Generator ソフトウェアは、メモリ エレメント、演算ファンクション、通信、I/O インターフェイス コアなどの高機能モジュールを作成する、対話型のグラフィカル デザイン ツールです。このツールを使用して、モジュールをカスタマイズおよび最適化することにより、高速キャリー ロジック、SRL16、分散 RAM、ブロック RAM などのザイリンクス FPGA デバイスのアーキテクチャ機能を最大限に活用できます。

このセクションでは、timer_preset という CORE Generator ソフトウェア モジュールを作成します。このモジュールは、タイマーに読み込む 64 個のプリセット値を保存するのに使用します。

timer_preset CORE Generator ソフトウェア モジュールの作成

CORE Generator ソフトウェア モジュールを作成するには、次の手順に従います。

1. [Project] → [New Source] をクリックします。
2. [IP (CORE Generator & Architecture Wizard)] を選択します。
3. [File name] に「timer_preset」と入力します。
4. [Next] をクリックします。
5. [Memories & Storage Elements] → [RAMs & ROMs] を展開します。

6. [Distributed Memory Generator] を選択して [Next] をクリックし、次のページで [Finish] をクリックします。CORE Generator で Distributed Memory Generator のカスタマイズ ウィンドウが開きます。このウィンドウで、メモリをデザインの仕様に合わせてカスタマイズします。

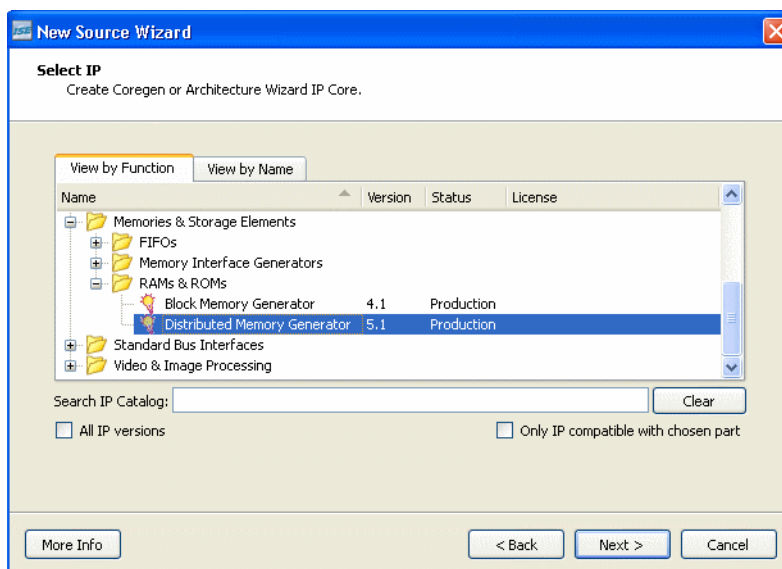


図 2-11 : New Source Wizard の [Select IP] ページ

7. Distributed Memory Generator のカスタマイズ ウィンドウで次の設定を入力します。
- ◆ [Component Name] : timer_preset (モジュール名を指定)
 - ◆ [Depth] : 64 (保存する値の数を指定)
 - ◆ [Data Width] : 20 (出力バスの幅を指定)
 - ◆ [Memory Type] : [ROM]

8. [Next] をクリックします。

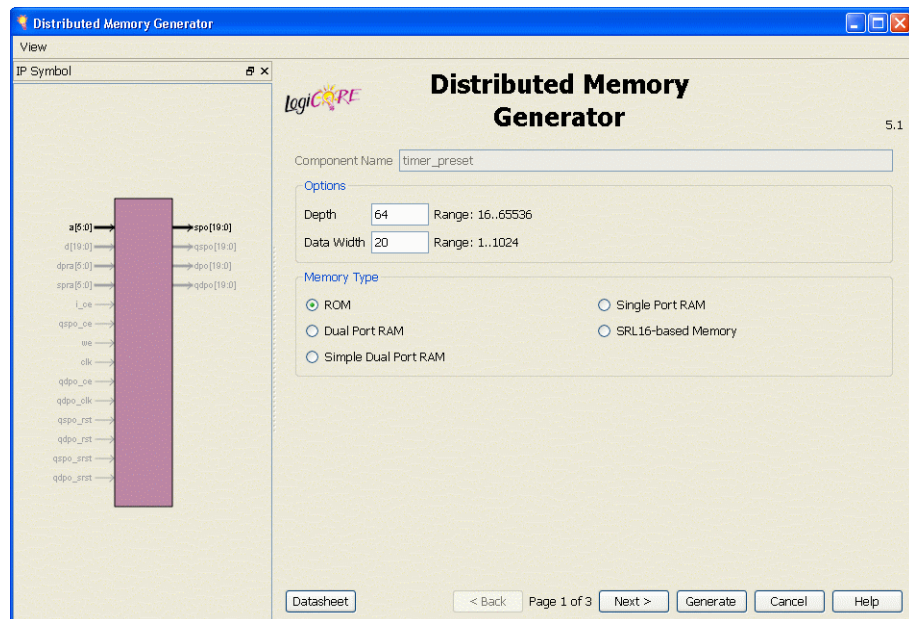


図 2-12 : CORE Generator ソフトウェアの Distributed Memory Generator のカスタマイズ ウィンドウ (1 ページ目)

9. [Input Options] および [Output Options] を [Non Registered] のままにし、[Next] をクリックします。

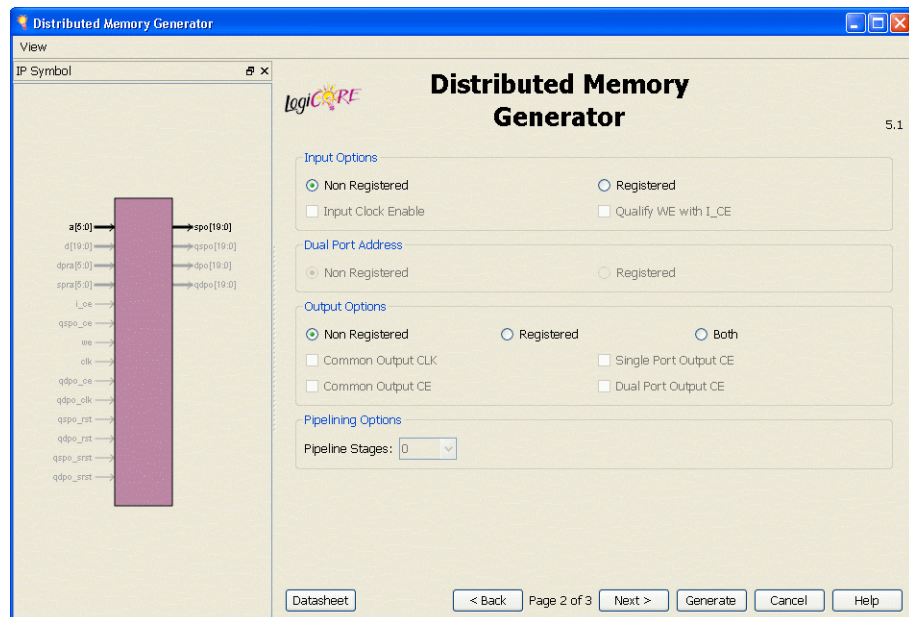


図 2-13 : CORE Generator ソフトウェアの Distributed Memory Generator のカスタマイズ ウィンドウ (2 ページ目)

10. [Coefficients File] で [Browse] ボタンをクリックし、プロジェクト ディレクトリにある definition1_times.coe を選択します。

11. カスタマイズ ウィンドウの左側のシンボルで、次のピンのみがハイライトされていることを確認します。
 - ◆ a[5:0]
 - ◆ spo[19:0]
12. [Generate] をクリックします。

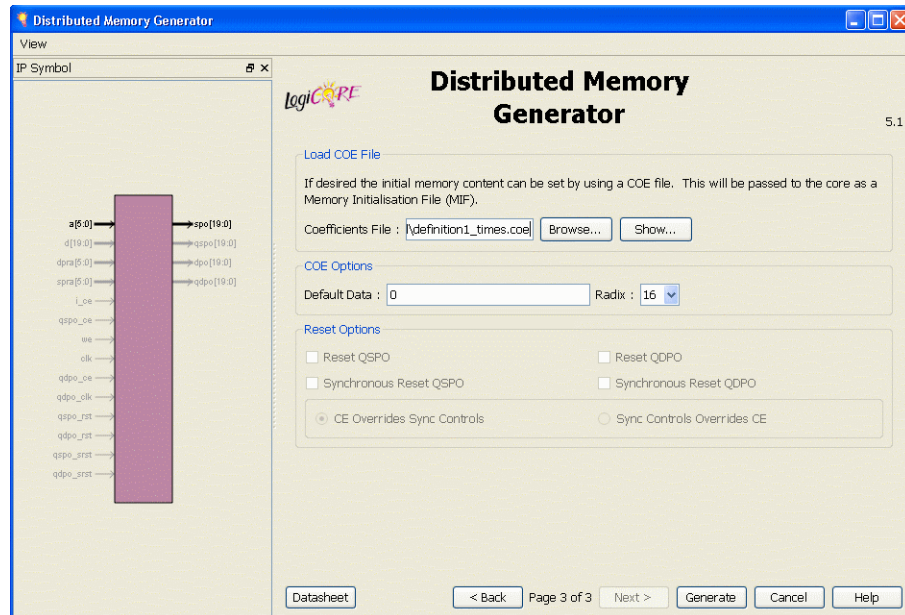


図 2-14 : CORE Generator ソフトウェアの Distributed Memory Generator のカスタマイズ ウィンドウ (3 ページ目)

作成されたマクロは、自動的にプロジェクトのライブラリに追加されます。

プロジェクト ディレクトリの `ipcore_dir` サブディレクトリに多数のファイルが追加されます。この中には、次のファイルが含まれます。

- ◆ `timer_preset.vho` または `timer_preset.v eo`
CORE Generator ソフトウェア モジュールをソース HDL に組み込むのに使用するインスタンスエーション テンプレートです。
- ◆ `timer_preset.vhd` または `timer_preset.v`
シミュレーションのみで使用するコアの HDL ラッパ ファイルです。
- ◆ `timer_preset.ngc`
インプリメンテーションの変換プロセスで使用するネットリストです。
- ◆ `timer_preset.xco`
`timer_preset` モジュールのコンフィギュレーション情報が格納されており、ISE プロジェクトでプロジェクト ソースとして使用されます。
- ◆ `timer_preset.mif`
シミュレーション用の ROM の初期値を供給します。

HDL コードへの CORE Generator ソフトウェア モジュールの インスタンス化

次に、VHDL フローまたは Verilog フローを使用して、CORE Generator ソフトウェア モジュールを HDL コードにインスタンス化します。

VHDL フロー

VHDL フローで CORE Generator ソフトウェア モジュールをインスタンス化するには、次の手順に従います。

1. [Hierarchy] ペインで stopwatch.vhd をダブルクリックし、ISE Text Editor で開きます。
2. カーソルを次の行の下に置きます。

```
-- Insert CORE Generator ROM component declaration here
```

3. [Edit] → [Insert File] をクリックし、[Choose a file to insert] ダイアログ ボックスで ipcore_dir/timer_preset.vho ファイルを選択して [開く] をクリックします。

次の図のように CORE Generator ソフトウェアのインスタンス化用の VHDL テンプレートが挿入されます。

```
121 ----- Begin Cut here for COMPONENT Declaration ----- COMP_TAG
122 component timer_preset
123     port (
124         a: IN std_logic_VECTOR(5 downto 0);
125         spo: OUT std_logic_VECTOR(19 downto 0));
126 end component;
127
128 -- Synplicity black box declaration
129 attribute syn_black_box : boolean;
130 attribute syn_black_box of timer_preset: component is true;
131
132 -- COMP_TAG_END ----- End COMPONENT Declaration -----
```

図 2-15 : CORE Generator ソフトウェア モジュールの VHDL コンポーネント宣言

4. コードの次の行から指定の行までを選択します。

```
-- Begin Cut here for INSTANTIATION Template ----
```

最後の行

```
-- INST_TAG_END ----- END INSTANTIATION Template -----
```

5. [Edit] → [Cut] をクリックします。
6. カーソルを次の行の下に置きます。
7. [Edit] → [Paste] をクリックしてコア インスタンス化を挿入します。
8. インスタンス名を「your_instance_name」から「t_preset」に変更します。
9. インスタンス化したコードを編集して、次に示すように stopwatch デザインの信号と CORE Generator ソフトウェア モジュールのポートを接続します。

```
169 ----- Insert CORE Generator ROM instantiation here
170 ----- Begin Cut here for INSTANTIATION Template ----- INST_TAG
171 t_preset : timer_preset
172     port map (
173         a => address,
174         spo => preset_time);
175 -- INST_TAG_END ----- End INSTANTIATION Template -----
```

図 2-16 : CORE Generator ソフトウェア モジュールの VHDL コンポーネントの
インスタンス化

10. 挿入した `timer_preset.vho` のコードには、説明および法的記述を含むコメント行が含まれています。これらのコメント行は、削除してもかまいません。
11. [File] → [Save] をクリックしてデザインを保存し、ISE Text Editor を閉じます。

Verilog フロー

Verilog フローで CORE Generator ソフトウェア モジュールをインスタンスエートするには、次の手順に従います。

1. [Hierarchy] ペインで `stopwatch.v` をダブルクリックし、ISE Text Editor で開きます。
2. カーソルを次の行の後に置きます。

```
// Place the Coregen module instantiation for timer_preset here
```
3. [Edit] → [Insert File] をクリックし、[Choose a file to insert] ダイアログ ボックスで `ipcore_dir/timer_preset.veo` を選択して [開く] をクリックします。
4. 挿入した `timer_preset.veo` のコードには、説明および法的記述を含むコメント行が含まれています。これらのコメント行は、削除してもかまいません。
5. インスタンス名を「YourInstanceName」から「t_preset」に変更します。
6. インスタンスエートしたコードを編集して、次に示すように `stopwatch` デザインの信号と CORE Generator ソフトウェア モジュールのポートを接続します。

```

36 // Place the Coregen module instantiation for timer_preset here
37 //----- Begin Cut here for INSTANTIATION Template ---// INST_TAG
38 timer_preset t_preset (
39     .a(address), // Bus [5 : 0]
40     .spo(preset_time)); // Bus [19 : 0]
41
42 // INST_TAG_END ----- End INSTANTIATION Template -----

```

図 2-17：CORE Generator ソフトウェア モジュールの Verilog コンポーネントのインスタンスエーション

7. [File] → [Save] をクリックしてデザインを保存し、ISE Text Editor を閉じます。
- コア モジュールが [Hierarchy] ペインの `stopwatch` モジュールの下に表示されます。

DCM モジュールの作成

Architecture Wizard の一部である Clocking Wizard を使用すると、DCM (デジタル クロック マネージャ) モジュールをグラフィカルに作成できます。このセクションでは、CLK0 フィードバックおよびデューティ サイクル調整がある基本的な DCM モジュールを作成します。

Clocking Wizard の使用

dcm1 モジュールを作成するには、次の手順に従います。

1. [Project] → [New Source] をクリックします。
2. New Source Wizard の [Select Source Type] ページで、ソース タイプとして [IP (CORE Generator & Architecture Wizard)] を選択し、[File name] に「dcm1」と入力します。
3. [Next] をクリックします。

4. [Select IP] ページで、[FPGA Features and Design] → [Clocking] → [Spartan-3E, Spartan-3A] → [Single DCM_SP] をクリックします。

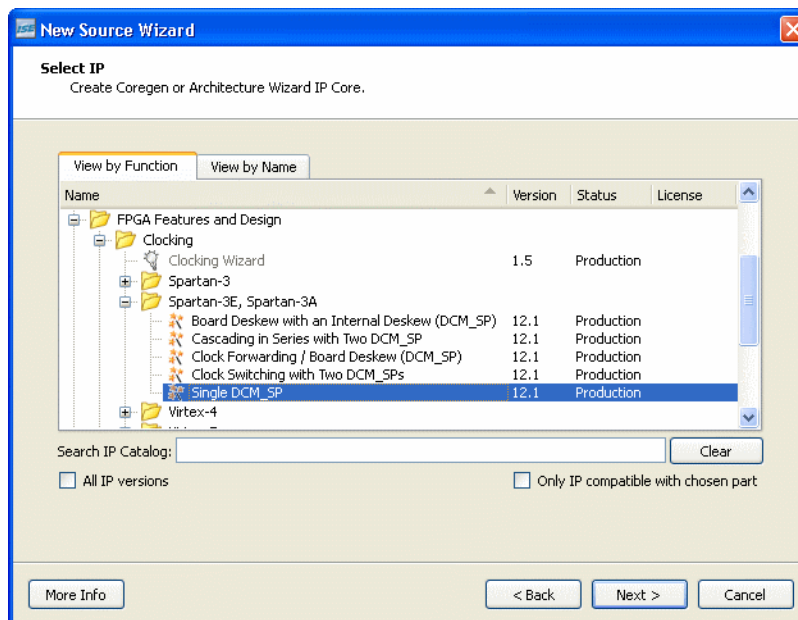


図 2-18 : Single DCM_SP コアを選択

5. [Next] をクリックし、次のページで [Finish] をクリックします。Clocking Wizard が開きます。
6. Architecture Wizard の [Setup] ページで [OK] をクリックします。
7. [General Setup] ページで [RST]、[CLK0]、および [LOCKED] がオンになっていることを確認します。
8. [CLKFX] ポートをオンにします。
9. [Input Clock Frequency] に「50」と入力し、[MHz] をオンにします。
10. 次の設定を確認します。
- ◆ [Phase Shift] : [NONE]
 - ◆ [CLKIN Source] : [External]、[Single]
 - ◆ [Feedback Source] : [Internal]
 - ◆ [Feedback Value] : [1X]
 - ◆ [Use Duty Cycle Correction] : オン
11. [Advanced] をクリックします。
12. [Wait for DCM lock before DONE signal goes high] をオンにします。
13. [OK] をクリックします。
14. [Next] をクリックし、次のページで [Next] をクリックします。
15. [Use output frequency] をオンにし、下のボックスに「26.2144」と入力して [MHz] をオンにします。

$$(26.2144\text{MHz})/2^{18} = 100\text{Hz}$$

16. [Next] をクリックし、次のページで [Finish] をクリックします。

dcm1.xaw ファイルが作成され、[Design] パネルの [Hierarchy] ペインのプロジェクト ソース ファイル リストに追加されます。

dcm1 マクロのインスタンス化 (VHDL デザイン)

次に、VHDL または Verilog デザインに dcm1 マクロをインスタンス化します。VHDL デザインに dcm1 マクロをインスタンス化するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで dcm1.xaw を選択します。
2. [Processes] ペインで [View HDL Instantiation Template] を右クリックし、[Process Properties] をクリックします。
3. [HDL Instantiation Template Target Language] を [VHDL] に設定し、[OK] をクリックします。
4. [Processes] ペインで [View HDL Instantiation Template] をダブルクリックします。
5. 表示された HDL インスタンス化テンプレート (dcm1.vhi) で、次に示すコンポーネント宣言のテンプレートを選択します。

```

4  -- Notes:
5  -- 1) This instantiation template has been auto
6  -- std_logic and std_logic_vector for the ports
7  -- 2) To use this template to instantiate this
8
9  COMPONENT dcm1
10 PORT(
11     CLKIN_IN : IN std_logic;
12     RST_IN : IN std_logic;
13     CLKFX_OUT : OUT std_logic;
14     CLKIN_IBUFG_OUT : OUT std_logic;
15     CLKD_OUT : OUT std_logic;
16     LOCKED_OUT : OUT std_logic;
17 );
18 END COMPONENT;
```

図 2-19 : DCM のコンポーネント宣言 (VHDL)

6. [Edit] → [Copy] をクリックします。
7. カーソルを stopwatch.vhd ファイルの次のセクションに置きます。
-- Insert dcm1 component declaration here.
8. [Edit] → [Paste] をクリックして、コンポーネント宣言を貼り付けます。
9. HDL インスタンス化テンプレートで、次に示すインスタンス化テンプレートを選択します。

```

19
20 Inst_dcm1: dcm1 PORT MAP (
21     CLKIN_IN => ,
22     RST_IN => ,
23     CLKFX_OUT => ,
24     CLKIN_IBUFG_OUT => ,
25     CLKD_OUT => ,
26     LOCKED_OUT =>
27 );
28
```

図 2-20 : DCM コンポーネントのインスタンス化 (VHDL)

10. [Edit] → [Copy] をクリックします。
11. カーソルを stopwatch.vhd ファイルの次の行の下に置きます。
-- Insert dcm1 instantiation here.

12. [Edit] → [Paste] をクリックして、インスタンス化テンプレートを貼り付けます。

13. 次の図と同じになるようにコードを変更します。

```

141 ----- Insert dcm1 instantiation here
142     Inst_dcm1: dcm1 PORT MAP (
143         CLKIN_IN => clk,
144         RST_IN => reset,
145         CLKFX_OUT => clk_26214k,
146         CLKIN_IBUFG_OUT => open,
147         CLKO_OUT => open,
148         LOCKED_OUT => locked
149     );

```

図 2-21 : dcm1 の VHDL インスタンス化

14. [File] → [Save] をクリックして、stopwatch.vhd ファイルを保存します。

dcm1 モジュールがデザイン階層のstopwatch モジュールの下に表示されます。

dcm1 マクロのインスタンス化 (Verilog)

Verilog デザインに dcm1 マクロをインスタンス化するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで dcm1.xaw を選択します。
2. [Processes] ペインで [View HDL Instantiation Template] をダブルクリックします。
3. 表示された HDL インスタンス化テンプレート (dcm1.tfi) で、次のインスタンス化テンプレートをコピーします。

```

4  // Instantiate the module
5  dcm1 instance_name (
6      .CLKIN_IN(CLKIN_IN),
7      .RST_IN(RST_IN),
8      .CLKFX_OUT(CLKFX_OUT),
9      .CLKIN_IBUFG_OUT(CLKIN_IBUFG_OUT),
10     .CLKO_OUT(CLKO_OUT),
11     .LOCKED_OUT(LOCKED_OUT)
12 );

```

図 2-22 : dcm1 マクロとインスタンス化テンプレート

4. コピーしたインスタンス化テンプレートを、stopwatch.v ファイルの次の行の下に貼り付けます。

```
//Insert dcm1 instantiation here.
```

5. 次の図と同じになるようにコードを変更します。

```

82
83 //Insert dcm1 instantiation here
84
85 dcm1 inst_dcm1 (
86     .CLKIN_IN(clk),
87     .RST_IN(reset),
88     .CLKFX_OUT(clk_26214k),
89     .CLKIN_IBUFG_OUT(),
90     .CLKO_OUT(),
91     .LOCKED_OUT(locked)
92 );

```

図 2-23 : dcm1 の Verilog インスタンス化

6. [File] → [Save] をクリックして、stopwatch.v ファイルを保存します。

dcm1 モジュールがデザイン階層のstopwatch モジュールの下に表示されます。

デザインの合成

ここまでは、Xilinx Synthesis Technology (XST) を構文チェックに使用しました。次に、XST、Synplify/Synplify Pro または Precision Synthesis ソフトウェアを使用してデザインを合成します。これらの合成ツールでは、デザインの HDL コードを使用してザイリンクス インプリメンテーション ツールでサポートされるネットリスト (EDIF または NGC) が生成されます。ネットリストの生成では、通常次のプロセスが実行されます。これらのプロセスは、どの合成ツールでもさらに細かいプロセスに分かれています。

- 構文の解析/構文チェック
ソース コードの構文をチェックします。
- コンパイル
HDL コードを合成ツールで識別可能な一連のコンポーネントに変換し、最適化します。
- マップ
コンパイル段階のコンポーネントをターゲット テクノロジーのプリミティブ コンポーネントに変換します。

合成ツールは、デザイン フローのどの段階でも変更できます。合成ツールを変更するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、デバイス名を選択します。
2. 右クリックして [Design Properties] をクリックします。
3. [Design Properties] ダイアログ ボックスで、[Synthesis Tool] フィールドのドロップダウン リストから使用する合成ツールを選択して、[OK] をクリックします。

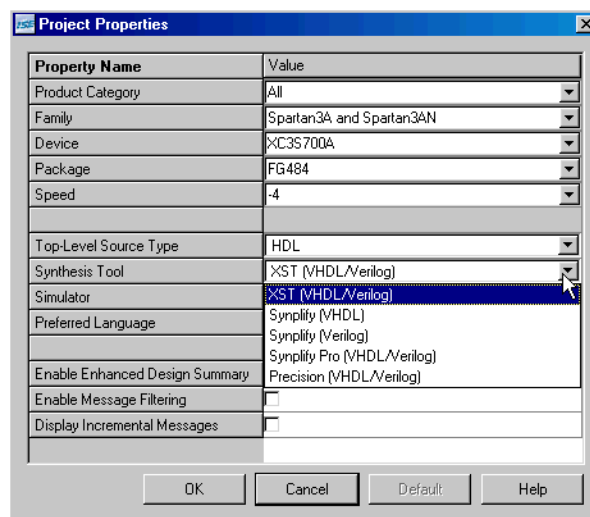


図 2-24：合成ツールの指定

メモ： 合成ツールがドロップリストに表示されない場合は、そのツールがインストールされていないか、または ISE ソフトウェアに統合されていない可能性があります。インストールされている合成ツールのパスを指定するには、[Edit] → [Preferences] をクリックし、[Preferences] ダイアログ ボックスで [ISE General] を展開して [Integrated Tools] をクリックします。

デザイン フローを変更すると、インプリメンテーション データが削除されます。このチュートリアルでは、インプリメンテーション データは作成されていません。インプリメンテーション データを含むプロジェクトでは、合成ツールを変更する前に [File] → [Copy Project] をクリックしてプロジェクトのコピーを作成することをお勧めします。

XST を使用した合成

この時点までで、デザインの作成および解析が完了しています。次に、デザインを合成します。合成では、HDL ファイルがゲートに変換されて、ターゲット アーキテクチャに対して最適化されます。

次に、XST を使用した合成で利用できるプロセスを示します。

- [View RTL Schematic]
RTL ネットリストの回路図を生成します。
- [View Technology Schematic]
テクノロジー ネットリストの回路図を生成します。
- [Check Syntax]
HDL コードが正しく記述されているかを確認します。
- [Generate Post-Synthesis Simulation Model]
合成ネットリストに基づいて HDL シミュレーション モデルを作成します。

合成オプションの入力

合成オプションを使用すると、合成ツールでの最適化の実行をデザイン要件に応じて制御できます。よく使用されるオプションは、エリアまたはスピードに基づいて最適化を実行するオプションです。そのほかに、フリップフロップ出力の最大ファンアウトの制御やデザインでの周波数の設定などのオプションがあります。

合成オプションを入力するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、stopwatch.vhd または stopwatch.v をクリックします。
2. [Processes] ペインで [Synthesize] を右クリックし、[Process Properties] をクリックします。
3. [Process Properties] ダイアログ ボックスの [Synthesis Options] ページで、[Netlist Hierarchy] プロパティを [Rebuilt] に設定します。
4. [OK] をクリックします。

デザインの合成

デザインを合成する準備が完了しました。HDL コードから互換性のあるネットリストを生成するには、次を実行します。

1. [Hierarchy] ペインで、stopwatch.vhd または stopwatch.v をクリックします。
2. [Processes] ペインで [Synthesize] をダブルクリックします。

RTL Viewer/Technology Viewer の使用

入力した HDL コードの回路図表現を生成できます。コードを回路図で表示すると、XST で推論されたさまざまなコンポーネント間の接続がグラフィカルに表示されるので、デザインを解析しやすくなります。回路図には、次の 2 つの形式があります。

- RTL 表示

HDL コードの最適化前の回路図表示

- テクノロジ表示

ターゲット テクノロジにマップされた HDL デザインの合成後の回路図表示

HDL コードの回路図を表示するには、次の手順に従います。

1. [Processes] ペインで [Synthesize] を展開し、[View RTL Schematic] または [View Technology Schematic] をダブルクリックします。
2. [Set RTL/Tech Viewer Startup Mode] ダイアログが表示されたら、[Start with the Explorer Wizard] をクリックします。
3. [Create RTL Schematic] ページで、[Available Elements] リストから clk_divider および lap_load_debounce コンポーネントを選択し、[Add] ボタンをクリックして選択した項目を [Selected Elements] に移動します。
4. [Create Schematic] をクリックします。

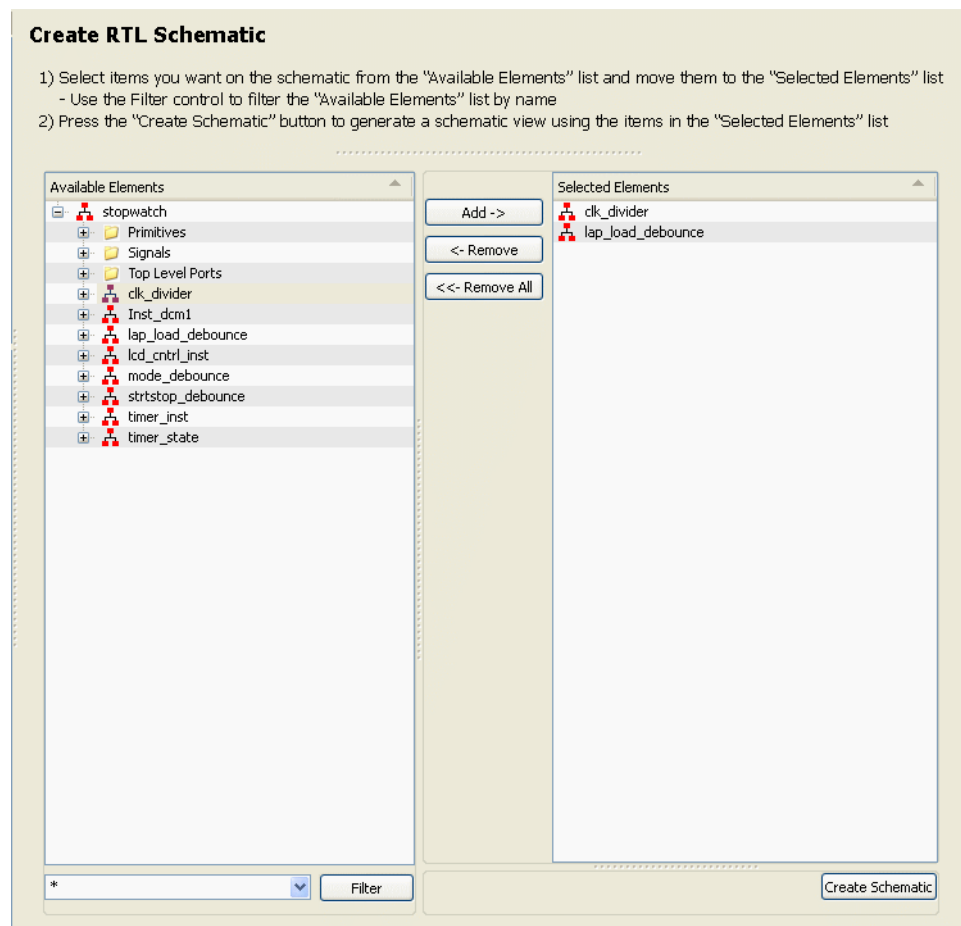


図 2-25 : [Create RTL Schematic] ページ

RTL Viewer では、デザインの一部のみを回路図として表示できます。表示されている回路図でシンボルをダブルクリックすると、そのシンボルの下位階層が表示され、デザインエレメントおよびその接続を確認できます。回路図を右クリックすると、回路図で実行可能な操作コマンドが表示されます。

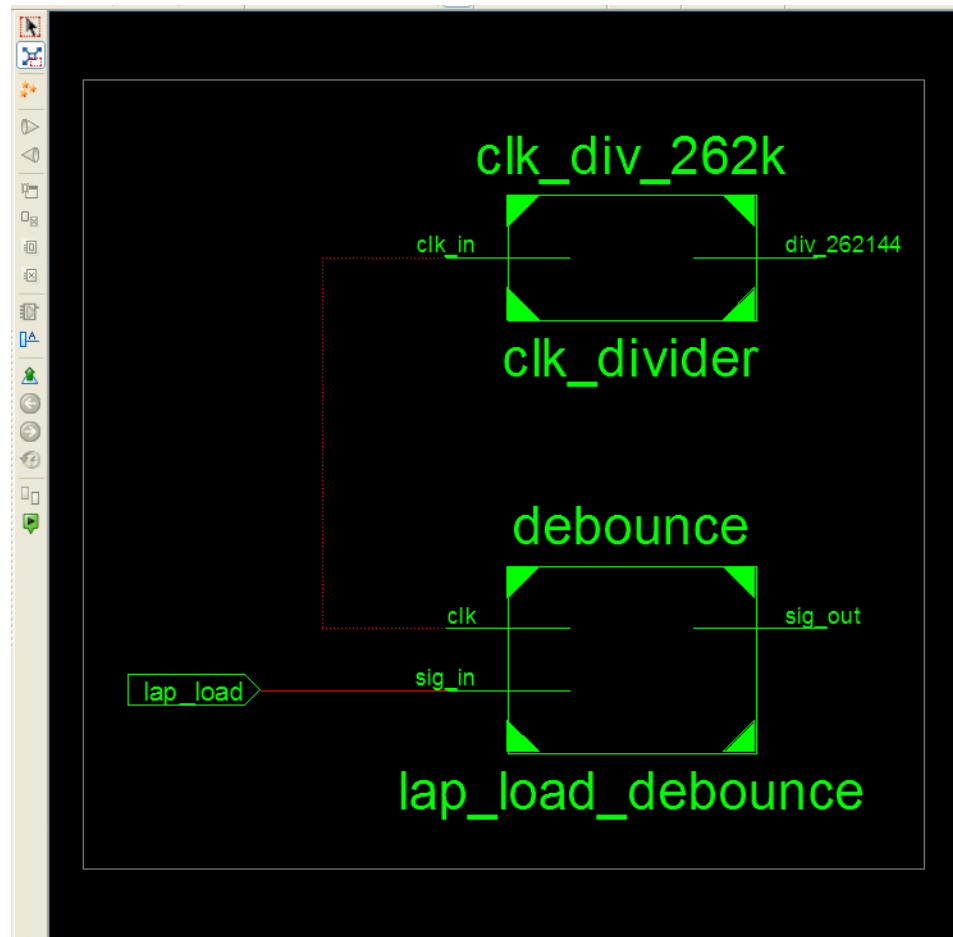


図 2-26 : RTL 回路図

XST での合成が完了し、stopwatch デザインの NGC ファイルが生成されました。

HDL フローを継続するには、次のいずれかの操作を実行します。

- 第 4 章「ビヘイビア シミュレーション」を参照して合成前のシミュレーションを実行します。
- 第 5 章「デザイン インプリメンテーション」を参照して配置配線を実行します。

メモ : XST の制約、オプション、レポート、およびコマンド ラインの使用に関する詳細は、『XST ユーザー ガイド』を参照してください。このガイドは ISE ソフトウェア マニュアル コレクションに含まれています。ソフトウェア マニュアル コレクションを開くには、[Help] → [Software Manuals] をクリックするか、ザイリンクスの Web サイトの [ソフトウェア マニュアル コレクション](#) にアクセスします。

Synplify/Synplify Pro ソフトウェアを使用したデザインの合成

この時点までで、デザインの作成および解析が完了しています。次に、デザインを合成します。合成では、HDL ファイルがゲートに変換されて、ターゲット アーキテクチャに対して最適化されます。Synplify ソフトウェアの RTL Viwer および制約エディタにアクセスするには、Synplify ソフトウェアをスタンドアロンで実行する必要があります。

Synplify および Synplify Pro ソフトウェアを使用した合成で利用できるプロセスは、次のとおりです。

- [View Synthesis Report]
最適化のレポートおよびマップ/タイミング サマリを表示します。
- [View RTL Schematic]
[Launch Tools] の下にあり、HDL コードを回路図で表示します。
- [View Technology Schematic]
[Launch Tools] の下にあり、ターゲット テクノロジーに関連するプリミティブにマップされている HDL コードを回路図で表示します。

合成オプションの入力とデザインの合成

グローバル合成オプションを設定してデザインを合成するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、stopwatch.vhd または stopwatch.v を選択します。
2. [Processes] ペインで [Synthesize] を右クリックし、[Process Properties] をクリックします。
3. [Process Properties] ダイアログ ボックスの [Synthesis Options] ページで、[Write Vendor Constraint File] をオンにします。
4. [OK] をクリックします。
5. [Processes] タブで [Synthesize] をダブルクリックして合成を実行します。

メモ：ファイルの指定後に [Processes] ペインで [Synthesize] をクリックしてから [Process] → [Run] をクリックするか、または [Synthesize] を右クリックして [Run] をクリックしてもこのプロセスを実行できます。

合成結果の確認

全般的な合成結果を表示するには、[Synthesize] の下にある [View Synthesis Report] をダブルクリックします。このレポートには、次のセクションが含まれています。

- 「コンパイラ レポート」
- 「マップ レポート」
- 「タイミング レポート」
- 「リソース使用率」

コンパイラ レポート

コンパイルされたすべての HDL ファイル名のリスト、最上位ファイルの名前、および各ファイルの構文チェックがレポートされます。また、FSM の抽出、推論されたメモリ、ラッチでの警告、未使用のポート、および不要なロジックの削除もレポートされます。

メモ：ブラック ボックス (デザイン環境に読み込まれていないモジュール) は、Synplify では常に「unbound」と示されます。ブラック ボックスを表すネットリスト (.ngo、.ngc、または .edn) がプロジェクト ディレクトリに存在していれば、インプリメンテーション ツールでこのネットリストが変換プロセス中にデザインに統合されます。

マップ レポート

使用した制約ファイル、ターゲット テクノロジ、設定されている属性がレポートされます。フラット化されたインスタンス、抽出されたカウンタ、最適化されたフリップフロップ、作成されたクロックおよびバッファ付きネットのマップ結果、および FSM のエンコード方式が記述されます。

タイミング レポート

入力した制約および制約がないデザイン部分の遅延についての詳細情報がレポートされます。遅延値は、ワイヤロード モデルに基づいた概算値です。より正確な遅延情報は、[第 5 章「デザイン インプリメンテーション」](#)に示されている配置配線後のタイミング レポートを参照してください。

```
Performance Summary
*****

Worst slack in design: -1.581
```

Starting Clock	Requested Frequency	Estimated Frequency	Requested Period	Estimated Period	Slack
stopwatch clk_divider.clk_100_inferred_clock	305.3 MHz	259.5 MHz	3.276	3.854	-0.578
stopwatch dcml_inst.CLKFX_BUF_derived_clock	111.6 MHz	94.9 MHz	8.959	10.540	-1.581

図 2-27： タイミングの予測データ (Synplify)

リソース使用率

ターゲット テクノロジで使用されるリソースすべてが示されます。

これで Synplify での合成が完了し、stopwatch デザインの EDN ファイルが生成されました。

HDL フローを継続するには、次のいずれかの操作を実行します。

- [第 4 章「ビヘイビア シミュレーション」](#)を参照して合成前のシミュレーションを実行します。
- [第 5 章「デザイン インプリメンテーション」](#)を参照して配置配線を実行します。

Precision Synthesis を使用した合成

この時点までで、デザインの作成および解析が完了しています。次に、デザインを合成します。合成では、HDL ファイルがゲートに変換されて、ターゲット アーキテクチャに対して最適化されます。

次に、Precision ソフトウェアを使用した合成で使用するプロセスを示します。

- [Check Syntax]
HDL コードの構文をチェックします。
- [View Log File]
最適化のレポートおよびマップ/タイミング サマリを表示します。
- [View RTL Schematic]
[Launch Tools] の下にあり、HDL コードを回路図で表示します。

- [View Technology Schematic]

[Launch Tools] の下にあり、ターゲット テクノロジに関連するプリミティブにマップされている HDL コードを回路図で表示します。

- [View Critical Path Schematic]

[Launch Tools] の下にあり、ターゲット テクノロジに関連するプリミティブにマップされている HDL コードのクリティカル パスが回路図で表示されます。

合成オプションの入力とデザインの合成

合成オプションを使用すると、合成ツールでの最適化の実行をデザイン要件に応じて制御できます。このチュートリアルでは、デフォルトのプロパティ設定を使用します。

デザインを合成するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、stopwatch.vhd または stopwatch.v をクリックします。
2. [Processes] ペインで [Synthesize] プロセスをダブルクリックします。

RTL Viewer/Technology Viewer の使用

入力した HDL コードの回路図表現を生成できます。コードを回路図で表示すると、Precision で推論されたさまざまなコンポーネント間の接続がグラフィカルに表示されるので、デザインを解析しやすくなります。RTL Viewer でデザインを表示するには、[View RTL Schematic] プロセスをダブルクリックします。次の図に、RTL Viewer に表示されるデザインを示します。

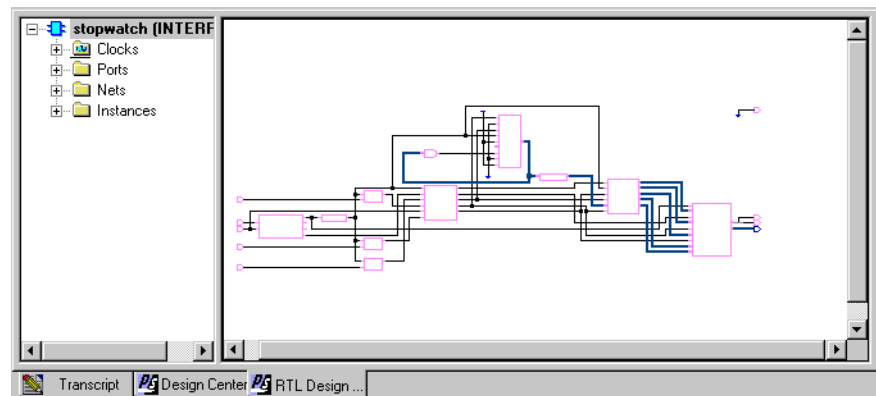


図 2-28 : Precision Synthesis の RTL Viewer に表示される stopwatch デザイン

これで合成が完了し、stopwatch デザインの EDN ファイルが生成されました。

HDL フローを継続するには、次のいずれかの操作を実行します。

- 第 4 章「ビヘイビア シミュレーション」を参照して合成前のシミュレーションを実行します。
- 第 5 章「デザイン インプリメンテーション」を参照して配置配線を実行します。

回路図ベースのデザイン

この章は、次のセクションから構成されています。

- 「回路図ベースのデザインの概要」
- 「入門」
- 「デザインの概要」
- 「デザイン入力」

回路図ベースのデザインの概要

この章では、stopwatch デザインを使用して、回路図に基づく典型的な FPGA デザインの手順を説明します。このチュートリアルで使用するデザイン例では、今後のデザイン作成に応用可能なデバイスのさまざまな機能、ソフトウェアの機能、およびデザイン フローが示されています。このデザインには Spartan™-3A デバイスが使用されていますが、ここで説明する原則およびフローは、注記がない限りすべてのザイリンクス デバイス ファミリに応用できます。

この章は、「回路図デザイン フロー」の第 1 ステップです。このチュートリアルの最初の部分では、ISE® のデザイン入力 ツールを使用してデザインを完成させます。このデザインは、回路図エレメント、CORE Generator™ ソフトウェア コンポーネント、および HDL マクロで構成されています。この章で正しくデザインを定義した後に、ビヘイビア シミュレーション (第 4 章「ビヘイビア シミュレーション」)、ザイリンクス インプリメンテーション ツールを使用したインプリメンテーション (第 5 章「デザイン インプリメンテーション」)、タイミング シミュレーション (第 6 章「タイミング シミュレーション」)、および Spartan-3A (XC3S700A) デモ ボードへのコンフィギュレーションおよびダウンロード (第 7 章「iMPACT チュートリアル」) を実行します。

入門

次に、このチュートリアルを実行するのに必要な条件を示します。

必要なソフトウェア

このチュートリアルを実行するには、ザイリンクス ISE Design Suite 12 がインストールされている必要があります。このデザインでは、Spartan-3A デバイス ライブラリおよびデバイス ファイルをインストールする必要があります。

このチュートリアルでは、ソフトウェアがデフォルトのディレクトリ `c:\xilinx\12.1\ISE_DS\ISE` にインストールされていることを前提としています。ソフトウェアを別のディレクトリにインストールしている場合は、そのインストール パスと置き換えてこのチュートリアルを実行してください。

メモ：ソフトウェアのインストール方法に関する詳細は、ザイリンクス Web サイトの『[ISE Design Suite 12：インストール、ライセンス、リリース ノート](#)』を参照してください。

チュートリアル プロジェクト ファイルのダウンロード

チュートリアル プロジェクト ファイルは、ザイリンクス Web サイトの ISE Design Suite 12 [チュートリアル ページ](#) からダウンロードできます。回路図デザイン ファイル (wtut_sc.zip) をダウンロードしてください。このファイルには、次の 2 つのディレクトリが含まれています。

- wtut_sc

回路図チュートリアルのソース ファイルが含まれます。チュートリアル プロジェクトはこのディレクトリに作成します。

- wtut_sc\wtut_sc_completed

回路図、HDL、ステート マシン ファイルを含む完成したチュートリアルのデザイン ファイルが含まれます。

メモ：このディレクトリに含まれるファイルは上書きしないでください。

ダウンロードしたファイルを解凍すると、回路図チュートリアル ファイルがそのディレクトリにコピーされます。このチュートリアルでは、`c:\xilinx\12.1\ISE_DS\ISE\ISEexamples` でファイルが解凍されていることを前提としていますが、読み出し/書き込み権限があればどのディレクトリに解凍してもかまいません。別のディレクトリにファイルを解凍した場合は、そのプロジェクトパスと置き換えてチュートリアルを実行してください。

ISE ソフトウェアの起動

ISE ソフトウェアを起動するには、デスクトップの [Xilinx ISE 12.1] アイコンをダブルクリックするか、または [スタート] → [すべてのプログラム] → [Xilinx ISE Design Suite 12.1] → [ISE デザイン ツール] → [Project Navigator] をクリックします。



図 3-1：デスクトップの [Xilinx ISE 12.1] アイコン

新規プロジェクトの作成

New Project Wizard を使用して新規プロジェクトを作成するには、次の手順に従います。

1. Project Navigator で、[File] → [New Project] をクリックします。
2. [Location] で、`c:\xilinx\12.1\ISE_DS\ISE\ISEexamples` またはプロジェクトを配置したディレクトリを選択します。
3. [Name] に「wtut_sc」と入力します。
4. [Top-level source type] を [Schematic] に設定し、[Next] をクリックします。
5. [Project Settings] ページで、次のように値を設定します。
 - ◆ [Product Category] : [All]
 - ◆ [Family] : [Spartan3A and Spartan3AN]
 - ◆ [Device] : [XC3S700A]
 - ◆ [Package] : [FG484]
 - ◆ [Speed] : [-4]
 - ◆ [Synthesis Tool] : [XST (VHDL/Verilog)]
 - ◆ [Simulator] : [ISim (VHDL/Verilog)]
 - ◆ [Preferred Language] : [VHDL] または [Verilog]
ここで選択した言語が、HDL ファイルを生成するすべてのプロセスのデフォルト言語になります。

その他のプロパティはデフォルト値のままにします。

6. [Next] をクリックして [Project Summary] ページで [Finish] をクリックし、プロジェクトの作成を完了します。

チュートリアルの中止

チュートリアルを中止する場合は、[File] → [Save All] をクリックして作業中のファイルを保存してください。

デザインの概要

このチュートリアルに使用されるデザインは、回路図に基づく階層デザインで、最上位デザインファイルである回路図シートから複数の下位マクロが参照されます。下位のマクロは、回路図モジュール、CORE Generator ソフトウェア モジュール、Architecture Wizard モジュール、および HDL モジュールなど、さまざまなモジュールから構成されています。

stopwatch デザインは、未完成のデザインです。このチュートリアルでは、モジュールを作成したり既存のファイルを編集したりして、デザインを完成させます。この章では、これらのモジュールを作成し、インスタンスシートして、接続します。完成したデザインの回路図を次の図に示します。

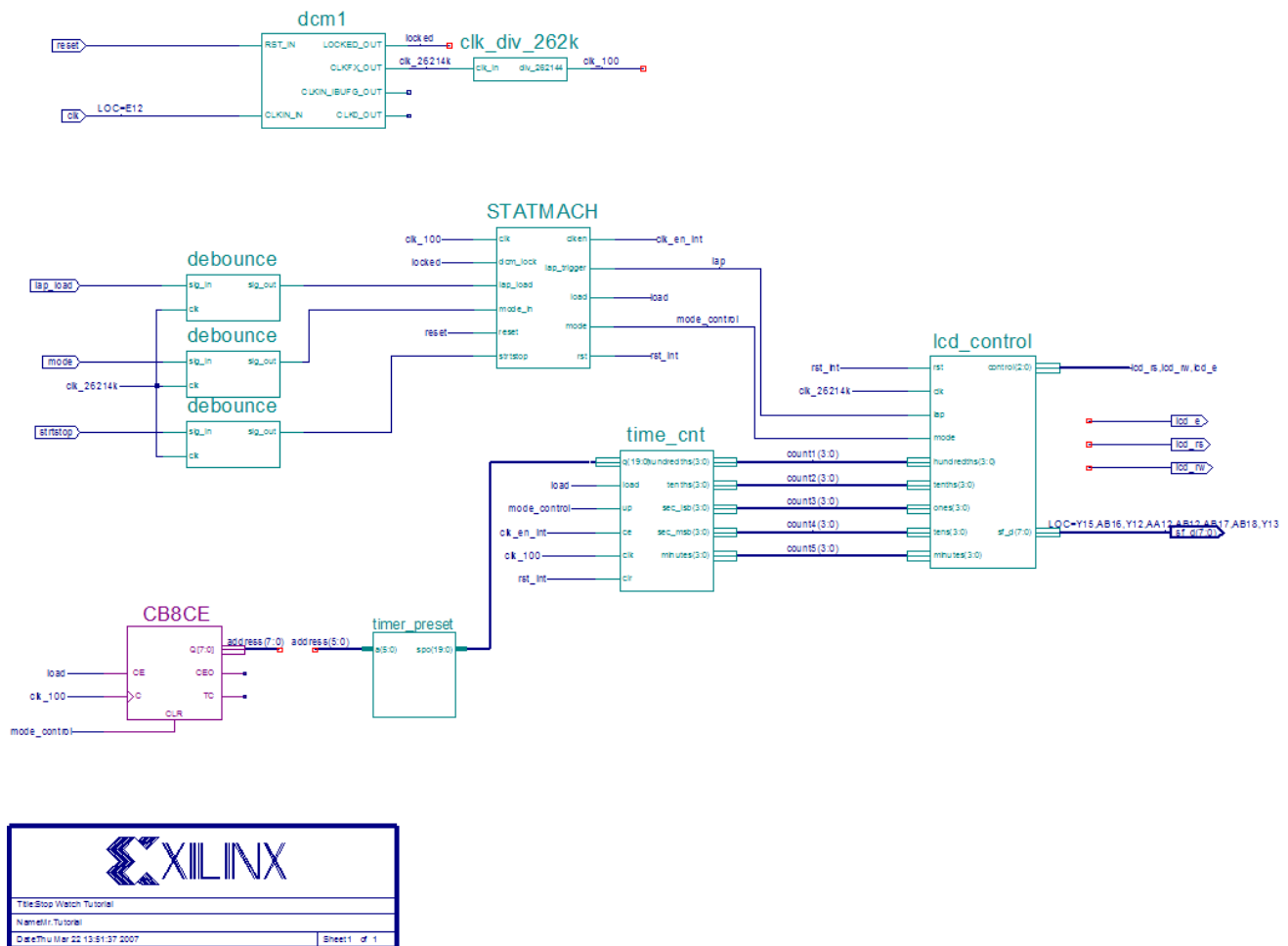


図 3-2：完成した stopwatch デザインの回路図

デザインが完成したら、シミュレーションを実行してその機能を検証します。シミュレーションの詳細は、第 4 章「[ビヘイビア シミュレーション](#)」を参照してください。

このデザインには、5 個の外部入力と 4 個の外部出力があります。次に、入力信号および出力信号のサマリを示します。

入力

次は、stopwatch デザインの入力信号です。

- **startstop**

ストップウォッチを開始、停止します。アクティブ **Low** 信号で、ストップウォッチの開始/停止ボタンと同じ動作をします。

- **reset**

ストップウォッチをクロック モードにして、時間を 0:00:00 にリセットします。

- **clk**

外部で生成されるシステム クロックです。

- **mode**
クロック モードとタイマー モードを切り替えます。クロックまたはタイマーが動作していないときにのみ有効になります。
- **lap_load**
2 つの機能を持つ信号で、クロック モードでは **Lap** ディスプレイに現在のクロック値を表示し、タイマー モードではタイマーが動作していないときに **ROM** からプリセット値をタイマー ディスプレイに読み込みます。

出力

次は、デザインの出力信号です。

- **lcd_e, lcd_rs, lcd_rw**
ストップウォッチの時間を表示するために使用される **Spartan-3A** デモ ボードの **LCD** ディスプレイを制御する信号です。
- **sf_d[7:0]**
LCD ディスプレイ用にデータ値を供給します。

論理ブロック

完成デザインは、次の論理ブロックから構成されています。これらのブロックのほとんどは、このチュートリアルで作成して回路図に追加するまで、プロジェクトの回路図シートには表示されません。

- **clk_div_262k**
クロック周波数を 262,144 で分周するマクロです。26.2144MHz のクロックをデューティ サイクル 50% で 100 Hz のクロックに変換します。
- **dcm1**
内部フィードバック、周波数を調整した出力、およびデューティ サイクル調整を含む **Clocking Wizard** マクロです。CLKFX_OUT 出力は、Spartan-3A デモ ボードの 50MHz を 26.2144MHz に変換します。
- **debounce**
strtstop、mode、lap_load 入力信号のデバウンス回路をインプリメントするモジュールです。
- **lcd_control**
LCD ディスプレイの初期化と出力を制御するモジュールです。
- **statmach**
ストップウォッチの状態を制御するステート マシン モジュールです。
- **timer_preset**
CORE Generator ソフトウェアの **64X20 ROM** です。このマクロには、タイマーに読み込む 0:00:00 ~ 9:59:99 の 64 個のプリセット時間が含まれています。
- **time_cnt**
0:00:00 ~ 9:59:99 の 10 進値をカウントするアップ/ダウン カウンタ モジュールで、ストップウォッチの時間の各桁を表す 4 ビットの出力が 5 つあります。

デザイン入力

この階層デザインでは、回路図ベースのマクロ、HDL ベースのマクロ、および CORE Generator ソフトウェア マクロなど、さまざまなマクロを作成します。各マクロを作成後に、これらを接続して stopwatch デザインを完成させます。このチュートリアルで使用する手順は、今後作成するデザインに応用できます。

ソース ファイルの追加

デザインを編集、合成、インプリメントするには、その前にソース ファイルがプロジェクトに追加されている必要があります。次に示すように、6 つのソース ファイルをプロジェクトに追加します。

1. [Project] → [Add Source] をクリックします。
 2. 次のファイルをプロジェクト ディレクトリから選択し、[Open] をクリックします。
 - ◆ cd4rled.sch
 - ◆ ch4rled.sch
 - ◆ clk_div_262k.vhd
 - ◆ lcd_control.vhd
 - ◆ stopwatch.sch
 - ◆ statmach.vhd
 3. [Adding Source Files] ダイアログ ボックスで、ファイルが [All] に関連付けられており、関連付けられたライブラリが [work] であることを確認してから、[OK] をクリックします。
- [Design] パネルの [Hierarchy] ペインには、現在プロジェクトに加えられているすべてのソース ファイルが、関連付けられているエンティティ名またはモジュールの名と共に表示されます。

ザイリンクス Schematic Editor で回路図ファイルを開く

stopwatch の回路図は、未完成の wtut_sc プロジェクトに含まれています。このチュートリアルでは、Schematic Editor で回路図を更新します。ISE ソフトウェアでプロジェクトを作成し、ソース ファイルを加えると、stopwatch.sch ファイルを開いて編集できます。回路図ファイルを開くには、[Design] パネルの [Hierarchy] ペインで stopwatch.sch をダブルクリックします。

ワークスペースに stopwatch の回路図が表示されます。次の図に示すように、未完成のデザインがエレメントと共に右下に表示されます。

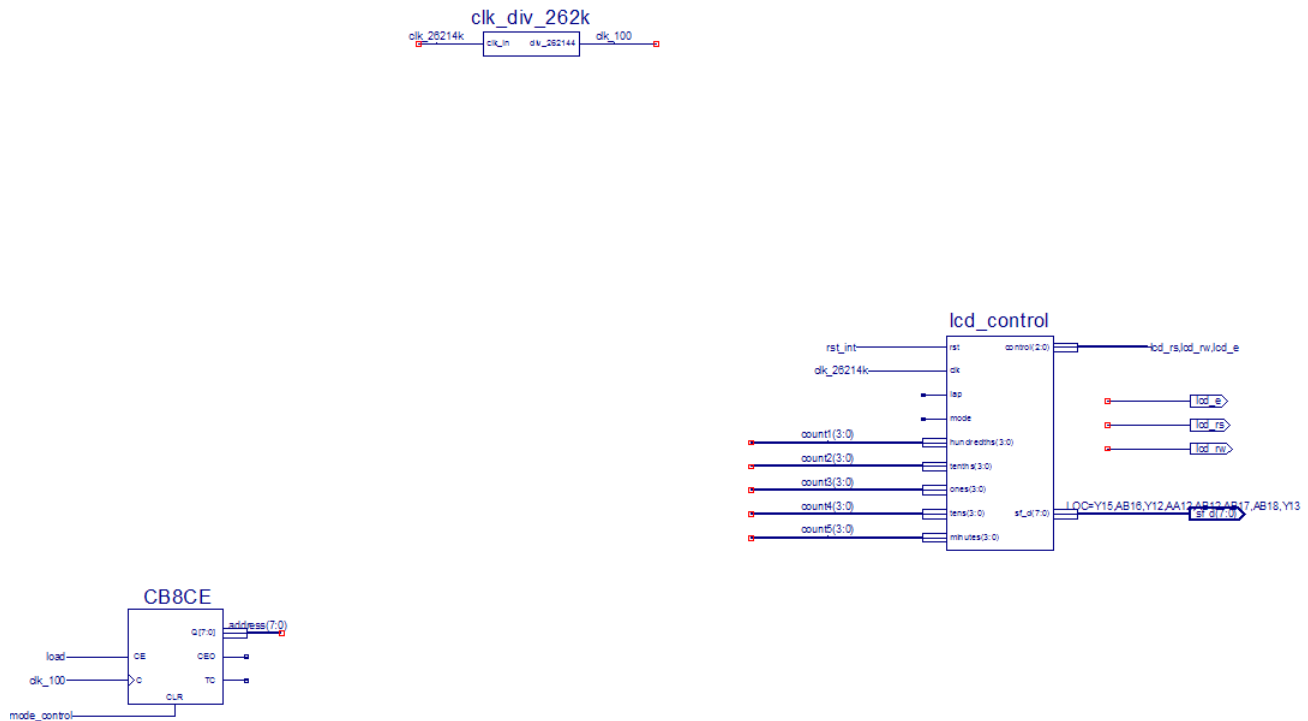


図 3-3 : 未完成の stopwatch 回路図

ウィンドウの表示の変更

[View] メニュー コマンドを使用すると、回路図の表示を変更できます。[View] → [Zoom] → [In] をクリックすると、回路図を拡大表示できます。

[Window] → [Float] をクリックすると、Project Navigator から回路図ウィンドウを切り離して別のウィンドウとして表示できます。

切り離れた回路図ウィンドウは、[Window] → [Dock] をクリックすると元に戻すことができます。

回路図ベースのマクロの作成

回路図ベースのマクロは、シンボルとそれを表す回路図から構成されます。いずれかを作成すると、もう一方のファイルを自動的に作成できます。

次に、[New Source] コマンドを使用して回路図ベースのマクロを作成します。空の回路図ファイルが作成され、このファイルで適切なロジックを定義できます。作成されたマクロは、自動的にプロジェクトのライブラリに追加されます。

作成するマクロ名は、`time_cnt` です。このマクロは、ストップウォッチの各桁を表す 4 ビット出力が 5 つあるバイナリ カウンタです。

回路図ベースのマクロを作成するには、次の手順に従います。

1. [Project] → [New Source] をクリックします。

New Source Wizard が開き、使用可能なソース タイプの一覧が表示されます。

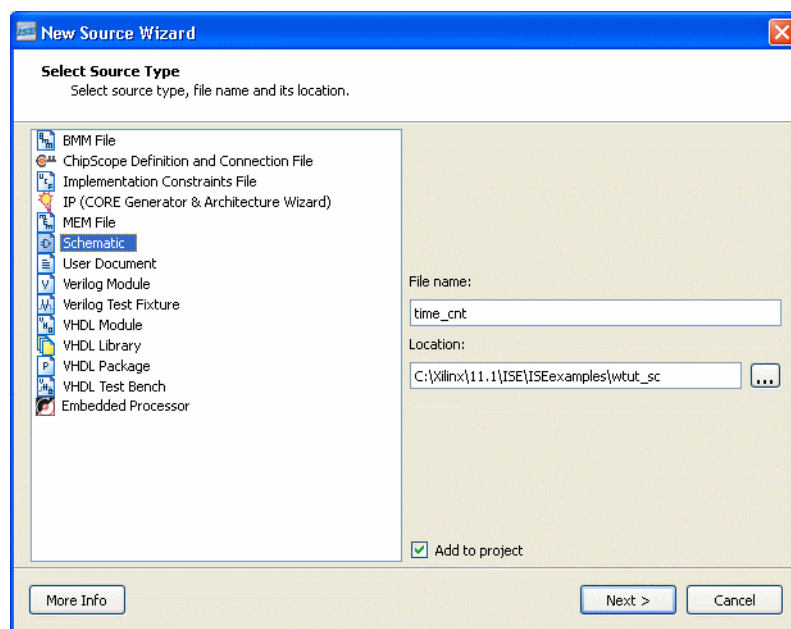


図 3-4 : [New Source Wizard] の [Select Source Type] ページ

2. ソース タイプとして [Schematic] を選択します。
3. [File name] に「`time_cnt`」と入力します。
4. [Next] をクリックし、次のダイアログ ボックスで [Finish] をクリックします。

回路図 `time_cnt.sch` が作成されてプロジェクトに追加され、ワークスペースに開きます。

5. 次の手順に従って、回路図シートのサイズを変更します。
 - ◆ 回路図を右クリックし、[Object Properties] をクリックします。
 - ◆ シート サイズ値の右側にある矢印をクリックし、[D = 34 x 22 in] を選択します。
 - ◆ [OK] をクリックします。シート サイズの変更は [Edit] → [Undo] オプションで元に戻すことができないことを示す警告メッセージが表示されたら、[Yes] をクリックします。

time_cnt 回路図の定義

time_cnt の空の回路図を作成したら、time_cnt マクロを構成するコンポーネントを追加します。このマクロ シンボルを回路図シートに配置すると、このマクロが参照されます。

I/O マーカーの追加

I/O マーカーを使用すると、マクロまたは最上位にあるポートを定義できます。シンボルの回路図には、シンボルの各ピンに対応するコネクタが含まれる必要があります。I/O マーカーを time_cnt に追加して、マクロのポートを定義します。

I/O マーカーを追加するには、次の手順に従います。

1. [Tools] → [Create I/O Markers] をクリックします。
[Create I/O Markers] ダイアログ ボックスが表示されます。
2. [Inputs] ボックスに「q(19:0),load,up,ce,clk,clr」と入力します。
3. [Outputs] ボックスに「hundredths(3:0),tenths(3:0),sec_lsb(3:0),sec_msb(3:0),minutes(3:0)」と入力します。

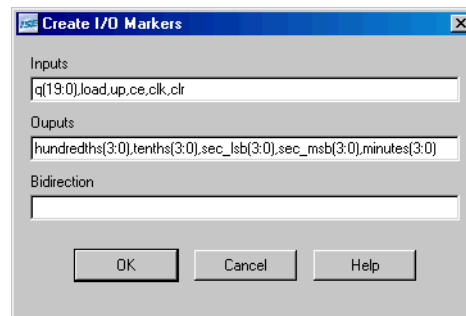


図 3-5 : [Create I/O Markers] ダイアログ ボックス

4. [OK] をクリックします。

11 個の I/O マーカーが回路図に追加されます。

メモ : [Create I/O Markers] コマンドは、空の回路図シートでのみ使用できます。[Add] → [I/O Marker] をクリックしてネットを選択すると、いつでもネットに I/O マーカーを追加できます。

回路図コンポーネントの追加

デバイスおよびプロジェクトのライブラリに含まれるコンポーネントは、そのシンボルを [Symbols] パネルを使用して回路図に配置できます。[Symbols] パネルに表示される使用可能なコンポーネントの一覧は、各ライブラリでアルファベット順に表示されます。

回路図コンポーネントを追加するには、次の手順に従います。

1. [Add] → [Symbol] をクリックするか、またはツールバーの [Add Symbol] ボタンをクリックします。



図 3-6 : ツールバーの [Add Symbol] ボタン

回路図の左側の [Options] パネルが表示されているエリアに、[Symbols] パネルが開き、ライブラリおよび対応するコンポーネントが表示されます。

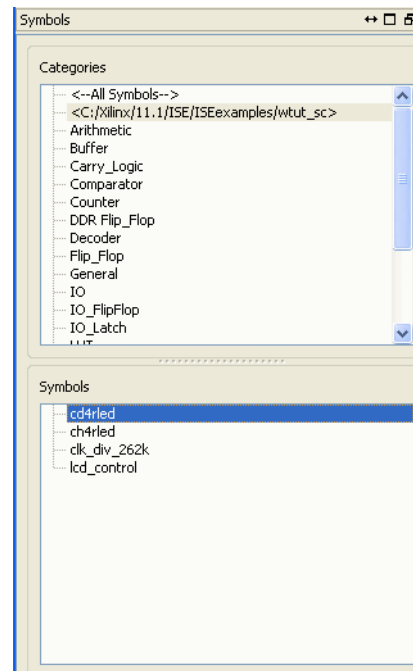


図 3-7 : [Symbols] パネル

メモ： [Options] パネルは、回路図で実行する動作によって変化します。

2. cd4rled (クロック イネーブルおよび同期クリア付きロード可能、双方向 4 ビット BCD カウンタ) を配置します。次のいずれかの方法を使用して、cd4rled コンポーネントを選択します。
 - ◆ [Symbols] パネルの [Categories] でプロジェクト ディレクトリを選択し、[Symbols] で [cd4rled] をクリックします。
 - ◆ [Categories] で [All Symbols] 選択し、[Symbol Name Filter] に「cd4rled」と入力します。
3. マウスを回路図ウィンドウに移動します。
カーソルが cd4rled のシンボルに変化します。
4. シンボルの輪郭をシートの上部中央に移動してクリックし、オブジェクトを配置します。

メモ： Ctrl + R キーを押すことにより、回路図に追加するコンポーネントを回転できます。既に配置されているコンポーネントも、コンポーネントを選択した後に Ctrl + R キーを押すと回転できます。

5. 同様に、さらに3つの cd4rled シンボルを回路図に配置します。次の図を参照してください。

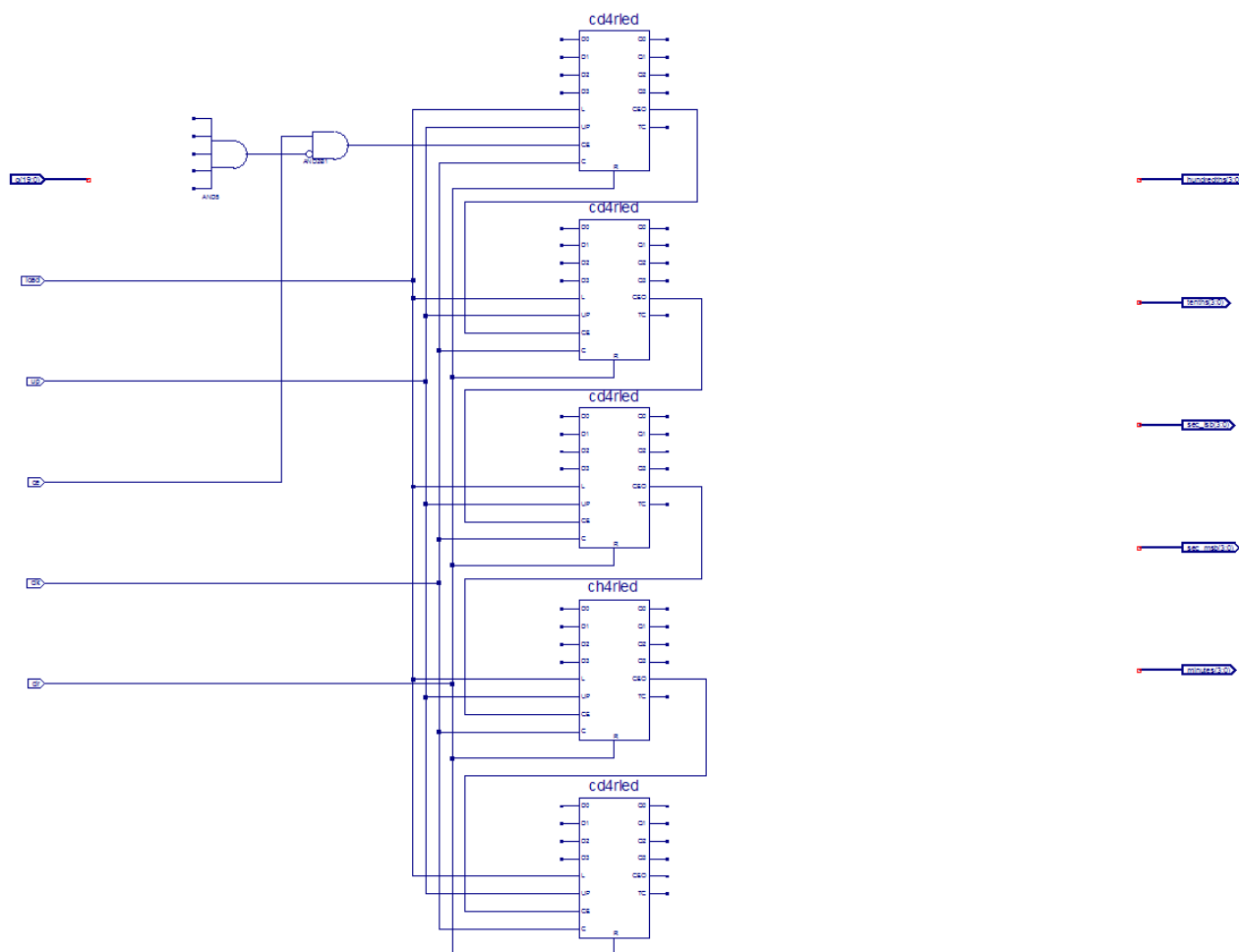


図 3-8：一部が完成した time_cnt の回路図

6. 手順 1 ～ 5 を繰り返して、次のコンポーネントを回路図シートに配置します。

- ◆ and2b1
- ◆ ch4rled
- ◆ and5

配置する位置は、図 3-8 を参照してください。

7. シンボルの入力を終了するには、Esc キーを押します。

ザイリンクス ライブラリ コンポーネントの機能の詳細を表示するには、コンポーネントを右クリックして [Object Properties] をクリックし、[Object Properties] ダイアログ ボックスで [Symbol Info] をクリックします。シンボルの情報は、ライブラリ ガイドでも参照できます。このガイドは、ISE ソフトウェア マニュアル コレクションに含まれています。ソフトウェア マニュアル コレクションを開くには、[Help] → [Software Manuals] をクリックするか、ザイリンクス Web サイトの [ソフトウェア マニュアル コレクション](#) にアクセスします。

間違った配置の修正

コンポーネントを間違えて配置した場合、次のように簡単に移動または削除できます。

- コンポーネントを移動するには、コンポーネントをクリックしてウィンドウ内でドラッグします。
- 配置済みのコンポーネントを削除するには、次のいずれかの方法を使用します。
 - ◆ コンポーネントをクリックし、Delete キーを押します。
 - ◆ コンポーネントを右クリックし、[Delete] をクリックします。

ワイヤの描画

ワイヤ (ネット) を描画して、回路図内に配置されたコンポーネントを接続できます。次の手順に従い、time_cnt 回路図上の AND2B1 コンポーネントと最上部 cd4rled コンポーネントの間にワイヤを描画します。

1. [Add] → [Wire] をクリックするか、またはツールバーの [Add Wire] ボタンをクリックします。



図 3-9 : ツールバーの [Add Wire] ボタン

2. AND2B1 の出力ピンをクリックし、cd4rled コンポーネントの CE ピンをクリックして、この 2 つのピン間にネットを描画します。
3. AND5 コンポーネントの出力と AND2B1 コンポーネントの反転入力を接続するネットを描画します。AND2B1 のもう 1 つの入力は、ce 入力 I/O マーカーに接続します。
4. 図 3-8 に図示されているように、load、up、clk、clr 入力 I/O マーカーを 5 つのカウンタ ブロックの L、UP、C、R ピンにそれぞれ接続し、最初の 4 つのカウンタの CEO ピンを次のカウンタの CE ピンに接続します。

ネットの形を指定するには、次の手順に従います。

1. ネットを描画する方向にマウスを移動します。
2. 任意の位置でクリックするとワイヤを 90 度曲げることができます。

メモ： 既存のネットとピンの間にネットを描画するには、コンポーネントのピンをクリックした後、既存のネットをクリックします。既存のネットに接続を示す点が表示されます。

バスの追加

Schematic Editor では、バスは多ビット名を持つワイヤとして示されます。バスを追加するには、まずワイヤを追加し、その後多ビット名を追加します。バスを作成したら、オプションで [Bus Tap] コマンドを使用し、バスの信号の 1 ビットを取り出すこともできます。

次に、time_cnt 回路図の 5 つの出力それぞれにバスを作成します。これらのバスは、完成した回路図に含まれています。

回路図に hundredths(3:0)、tenths(3:0)、sec_lsb(3:0)、sec_msb(3:0)、minutes(3:0) バスを追加するには、次の手順に従います。

1. 出力 I/O マーカーすべてを囲むボックスを描いてそれらを選択し、minutes(3:0) が最下位のカウンタ ブロックの Q3 出力よりも下になるように、ドラッグして移動します。
2. [Add] → [Wire] をクリックするか、またはツールバーの [Add Wire] ボタンをクリックします。

3. 1 番上の cd4rled の右上部の空白部分をクリックし、I/O マーカー hundredths(3:0) のピンをクリックします。I/O マーカーと同じ名前のバスを表す太い線が描画されます。

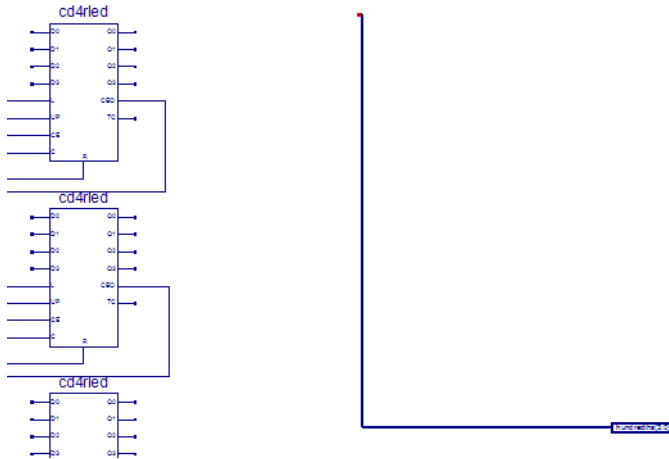


図 3-10 : バスの追加

4. 残りの 4 つのバスに対し、手順 2 ～ 3 を繰り返します。
5. バスを 5 つ追加したら、Esc キーを押すか、またはバスの終了点を右クリックして [Add Wire] モードを終了します。

バス タップの追加

次に、cd4rled および ch4rled カウンタのピンをバスに接続するネットを追加します。バス タップを使用してバスの 1 ビットを取り出し、別のコンポーネントに接続します。

メモ：回路図を拡大表示すると、ネットをより正確に描画できます。

各バスの 1 ビットを取り出すには、次の手順に従います。

1. [Add] → [Bus Tap] をクリックするか、またはツールバーの [Add Bus Tap] ボタンをクリックします。



図 3-11 : ツールバーの [Add Bus Tap] ボタン

カーソルがバス タップの形に変化します。

2. [Options] パネルの [Add Bus Tap Options] で、バス タップの方向を [--< Right] に設定します。
3. カーソルの中央のバーで hundredths(3:0) バスをクリックします。

[Options] パネルの [Selected Bus Name] と [Net Name] に値が自動的に挿入されます。

メモ：ネット名のインデックス値は、[Net Name] ボックスの横にある矢印ボタンをクリックして、増やしたり減らしたりできます。

4. [Net Name] の値が hundredths(3) の状態で、タップの先が 1 番上の cd4rled コンポーネントの Q3 ピンに触れるようにカーソルを移動します。

メモ：カーソルを正しい位置に移動すると、ピンの周りに 4 つの正方形が表示されます。

5. カーソルを正しい位置に移動したら、1 クリックします。
タップが `hundredths(3:0)` バスに接続され、`hundreths(3)` という名前のワイヤがタップと Q3 ピンの間に描画されます。

Q2、Q1、Q0 の各ピンを順にクリックして、`hundredths(3:0)` の残りのビットにタップを作成します。
6. 残りの 4 つのバスに対して手順 3 ～ 5 を繰り返します。
メモ：バスとワイヤの間は、ワイヤ名を付けることによって電氣的に接続されます (例：`sec_msb(2)` と `sec(3:0)` の 3 番目のビット)。バス タップは、表示目的のみで使用されます。次のセクションで、ネット名を追加してその他のコンポーネントを電氣的に接続します。
7. Esc キーを押して、[Add Bus Tap] コマンドを終了します。
8. 作成した `time_cnt` 回路図と図 3-13 を比較して、すべてが正しく接続されていることを確認します。

ネット名の追加

まず、AND5 コンポーネントの 5 つの入力と、各カウンタ ブロックの TC ピンに、未接続のワイヤを追加します。

次に、ワイヤにネット名を追加します。ネット名を追加するには、次の手順に従います。

1. [Add] → [Net Name] をクリックするか、またはツールバーの [Add Net Name] ボタンをクリックします。



図 3-12：ツールバーの [Add Net Name] ボタン

2. [Options] パネルの [Add Net Name Options] で、次の手順を実行します。
 - a. [Name] に「`tc_out0`」と入力します。
 - b. [Increase the name] をオンにします。入力したネット名 `tc_out0` がカーソルに表示されます。
3. AND5 コンポーネントの 1 番上の入力に接続されているネットをクリックします。
これでネット名がネットに付けられます。ネット名は、ネットのエンドポイント以外の位置に配置すると、ネットの上部に表示されます。
4. AND5 の残りの入力ネットを順にクリックして、`tc_out1`、`tc_out2`、`tc_out3`、`tc_out4` を追加します。

ネットに名前を追加するたびに、名前のインデックス値が増加します。
メモ：最初のネット名を `tc_out4` として [Decrease the name] をオンにすると、インデックス値は減少します。
5. 手順 2 を繰り返して、TC 出力に接続されている各ネットを順にクリックして、`tc_out0`、`tc_out1`、`tc_out2`、`tc_out3`、`tc_out4` を追加します。
メモ：同一名のワイヤ同士は、電氣的に接続されます。この場合、ネットを論理的に接続するために、回路図で物理的に接続する必要はありません。

最後に、次の手順に従ってカウンタの入力ピンをネット名を付けることにより接続します。

1. [Add] → [Wire] をクリックするか、またはツールバーの [Add Wire] ボタンをクリックして、各カウンタの 4 つのデータ ピンに未接続のワイヤを追加します。
2. [Add] → [Net Name] をクリックするか、またはツールバーの [Add Net Name] ボタンをクリックします。
3. [Options] パネルの [Add Net Name Options] で、[Name] に「q(0)」と入力します。
4. [Increase the Name] をオンにします。

入力したネット名「q(0)」がカーソルに表示されます。

5. データ入力に接続されているネットを上から順にクリックし、1 番上のカウンタの D0 ピンに q(0) という名前が付けられ、1 番下のカウンタの D3 ピンに q(19) という名前が付けられるようにします。次の図を参照してください。

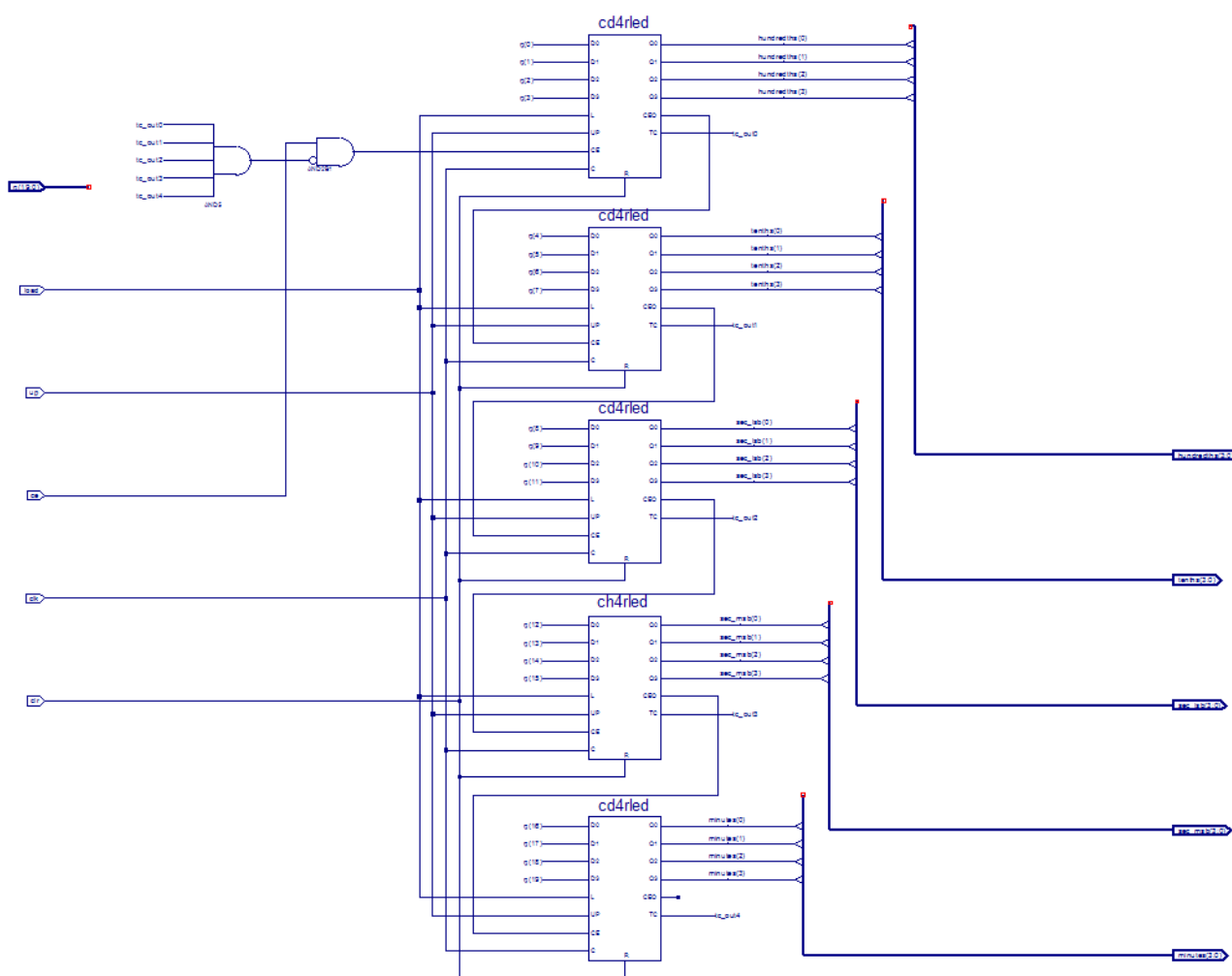


図 3-13 : 完成した time_cnt の回路図

メモ：ネットが接続されていないように見える場合、[View] → [Refresh] をクリックして画面を更新します。

回路図のチェック

これで、time_cnt 回路図が完成しました。デザインルールチェック (DRC) を実行して、回路図に論理エラーがないかどうかを確認します。これには、[Tools] → [Check Schematic] をクリックします。[Console] パネルに、エラーおよび警告がないことが示されるはずです。エラーまたは警告が示された場合は、次に進む前に修正してください。

回路図の保存

次の手順で回路図を保存します。

1. [File] → [Save] をクリックするか、またはツールバーの [Save] ボタンをクリックします。



図 3-14：ツールバーの [Save] ボタン

2. time_cnt 回路図を閉じます。

time_cnt のシンボルの作成および配置

time_cnt マクロを表すシンボルを作成します。シンボルは、マクロのインスタンスエーションです。time_cnt のシンボルを作成して、stopwatch デザインの最上位回路図に追加します作成した time_cnt マクロのシンボルは、最上位の回路図でこの後のセクションで作成するコンポーネントと接続します。

time_cnt シンボルの作成

シンボルの作成には、[Processes] ペインを使用するか、または [Tools] メニューを使用します。

[Processes] ペインを使用して time_cnt 回路図のシンボルを作成するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで time_cnt.sch を選択します。
2. [Processes] ペインで [Design Utilities] を展開し、[Create Schematic Symbol] をダブルクリックします。

[Tools] メニューを使用して、time_cnt 回路図のシンボルを作成するには、次の手順に従います。

1. time_cnt 回路図を開いた状態で、[Tools] → [Symbol Wizard] をクリックします。
2. Symbol Wizard の [Source Page] ページで、[Using Schematic] をオンにし、その横のボックスで「time_cnt」を選択します。
3. [Next]、[Next]、[Next]、[Finish] を順番にクリックして、デフォルトの設定でウィザードを完了します。
4. 作成された time_cnt シンボルが表示されます。確認して閉じます。

time_cnt シンボルの配置

次に、マクロのシンボルを最上位回路図 (stopwatch.sch) に、次の手順で配置します。

1. [Design] パネルの [Hierarchy] ペインで、stopwatch.sch をダブルクリックして回路図を開きます。
2. [Add] → [Symbol] をクリックするか、またはツールバーの [Add Symbol] ボタンをクリックします。



図 3-15 : ツールバーの [Add Symbol] ボタン

3. [Symbols] パネルの [Categories] でローカル シンボル ライブラリ (c:\xilinx\12.1\ISE_DS\ISE\ISEexamples\wtut_sc) をクリックし、[Symbols] で新規作成した [time_cnt] をクリックします。
4. time_cnt シンボルを、出力ピンが lcd_control コンポーネントの入力を駆動する 5 つのバスと並ぶように配置します。これは座標位置 [1612,1728] の近くになるはずです。座標位置は Project Navigator ウィンドウの右下に表示されており、回路図上でカーソルを動かすと値が変化します。
メモ：ここでは、time_cnt シンボルの入力ピンとネットは接続しません。stopwatch 回路図にほかのコンポーネントを追加した後で接続します。
5. 変更を保存し、stopwatch.sch を閉じます。

CORE Generator ソフトウェア モジュールの作成

Core Generator ソフトウェアは、メモリ エLEMENT、演算ファンクション、通信、I/O インターフェイス コアなどの高機能モジュールを作成する対話型のグラフィカル デザイン ツールです。このツールを使用して、モジュールをカスタマイズおよび最適化することにより、高速キャリー ロジック、SRL16、分散 RAM、ブロック RAM などのザイリンクス FPGA デバイスのアーキテクチャ機能を最大限に活用できます。

このセクションでは、timer_preset という CORE Generator ソフトウェア モジュールを作成します。このモジュールは、タイマーに読み込む 64 個のプリセット値を格納するために使用されます。

timer_preset CORE Generator ソフトウェア モジュールの作成

CORE Generator ソフトウェア モジュールを作成するには、次の手順に従います。

1. [Project] → [New Source] をクリックします。
2. [IP (CORE Generator & Architecture Wizard)] を選択します。
3. [File name] に「timer_preset」と入力します。
4. [Next] をクリックします。
5. [Memories & Storage Elements] → [RAMs & ROMs] を展開します。

6. [Distributed Memory Generator] を選択して [Next] をクリックし、次のページで [Finish] をクリックします。CORE Generator で Distributed Memory Generator のカスタマイズ ウィンドウが開きます。このウィンドウで、メモリをデザインの仕様に合わせてカスタマイズします。

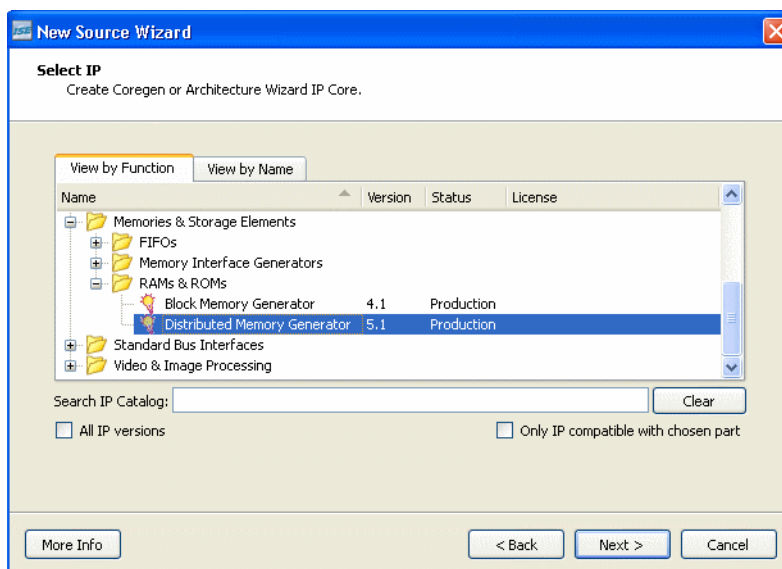


図 3-16 : New Source Wizard の [Select IP] ページ

7. Distributed Memory Generator のカスタマイズ ウィンドウで次の設定を入力します。
- ◆ [Component Name] : timer_preset (モジュール名を指定)
 - ◆ [Depth] : 64 (格納する値の数を指定)
 - ◆ [Data Width] : 20 (出力バスの幅を指定)
 - ◆ [Memory Type] : [ROM]
8. [Next] をクリックします。

9. [Input Options] および [Output Options] を [Non Registered] のままにし、[Next] をクリックします。

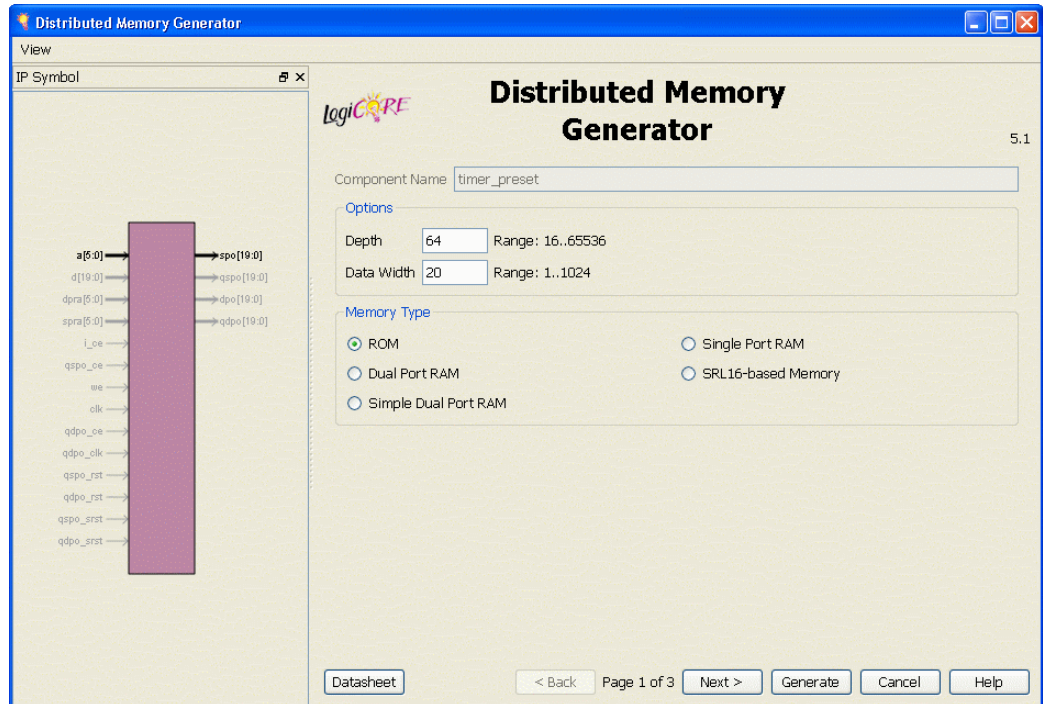


図 3-17 : Core Generator ソフトウェアの Distributed Memory Generator のカスタマイズ ウィンドウ

10. [Coefficients File] で [Browse] ボタンをクリックし、プロジェクト ディレクトリにある `definition1_times.coe` を選択します。
11. カスタマイズ ウィンドウの左側のシンボルで、次のピンのみがハイライトされていることを確認します。
- ◆ `a[5:0]`
 - ◆ `spo[19:0]`
12. [Generate] をクリックします。

作成されたマクロは、自動的にプロジェクトのライブラリに追加されます。

プロジェクト ディレクトリの `ipcore_dir` サブディレクトリに多数のファイルが追加されます。この中には、次のファイルが含まれます。

- `timer_preset.sym`
回路図のシンボル ファイルです。
- `timer_preset.vhd` または `timer_preset.v`
シミュレーションのみで使用されるコアの HDL ラップ ファイルです。
- `timer_preset.ngc`
インプリメンテーションの変換プロセスで使用されるネットリストです。

- timer_preset.xco

timer_preset モジュールのコンフィギュレーション情報が格納されており、プロジェクト ソースとして使用されます。

- timer_preset.mif

このファイルは、シミュレーション用の ROM の初期値を提供します。

DCM モジュールの作成

Architecture Wizard の一部である Clocking Wizard を使用すると、DCM (デジタル クロック マネージャ) モジュールをグラフィカルに作成できます。このセクションでは、CLK0 フィードバックおよびデューティ サイクル調整がある基本的な DCM モジュールを作成します。

Clocking Wizard の使用法

dcm1 モジュールを作成するには、次の手順に従います。

1. [Project] → [New Source] をクリックします。
2. New Source Wizard で [IP (CORE Generator & Architecture Wizard)] を選択し、[File name] に「dcm1」と入力します。
3. [Next] をクリックします。
4. [Select IP] ページで、[FPGA Features and Design] → [Clocking] → [Spartan-3E, Spartan-3A] → [Single DCM_SP] をクリックします。

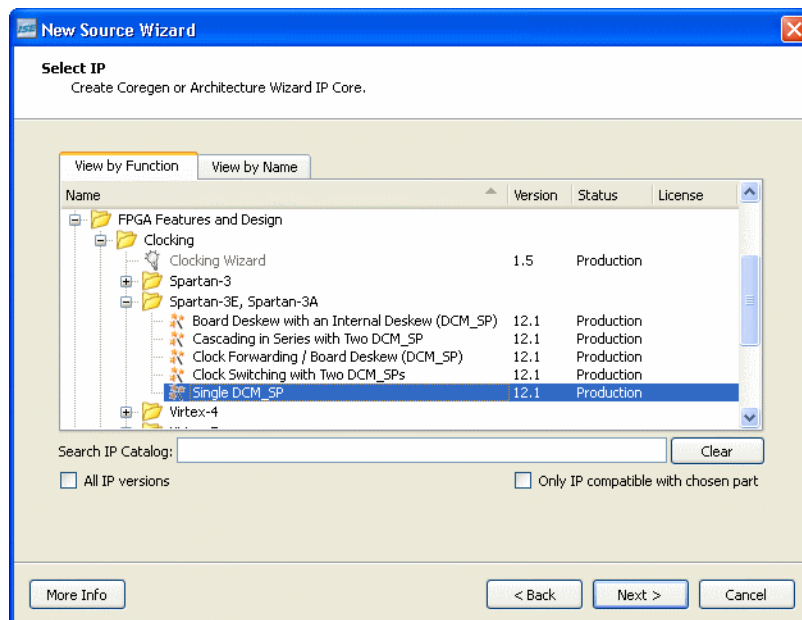


図 3-18 : Single DCM_SP コアを選択

5. [Next] をクリックし、次のページで [Finish] をクリックします。Clocking Wizard が開きます。
6. Architecture Wizard の [Setup] ページで [OK] をクリックします。
7. [General Setup] ページで [RST]、[CLK0]、および [LOCKED] がオンになっていることを確認します。

8. [CLKFX] ポートをオンにします。
9. [Input Clock Frequency] に「50」と入力し、[MHz] をオンにします。
10. 次の設定を確認します。
 - ◆ [Phase Shift] : [NONE]
 - ◆ [CLKIN Source] : [External]、[Single]
 - ◆ [Feedback Source] : [Internal]
 - ◆ [Feedback Value] : [1X]
 - ◆ [Use Duty Cycle Correction] : オン

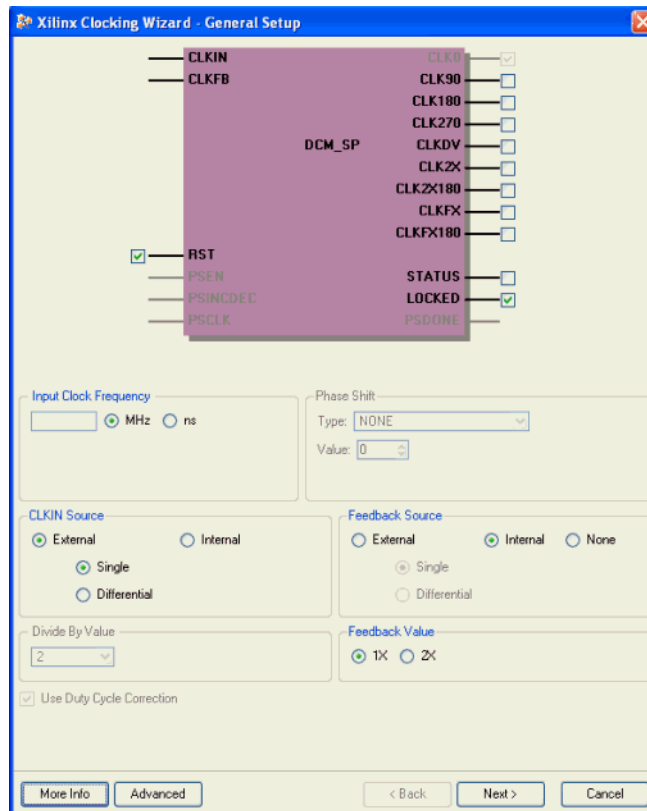


図 3-19 : Xilinx Clocking Wizard の [General Setup] ページ

11. [Advanced] をクリックします。
12. [Wait for DCM lock before DONE signal goes high] をオンにします。
13. [OK] をクリックします。
14. [Next] をクリックし、次のページで [Next] をクリックします。
15. [Use output frequency] をオンにし、下のボックスに「26.2144」と入力して [MHz] をオンにします。

$$(26.2144\text{MHz})/2^{18} = 100\text{Hz}$$

16. [Next] をクリックし、次のページで [Finish] をクリックします。

dcm1.xaw ファイルが作成され、[Design] パネルの [Hierarchy] ペインの プロジェクト ソース ファイル リストに追加されます。

dcm1 シンボルの作成

dcm1 マクロを表すシンボルを作成します。このシンボルは、後で最上位回路 (stopwatch.sch) に追加します。

1. [Design] パネルの [Hierarchy] ペインで dcm1.xaw を選択します。
2. [Processes] ペインで [Create Schematic Symbol] をダブルクリックします。

HDL ベースのモジュールの作成

ISE ソフトウェアでは、HDL コードから簡単にモジュールを作成できます。HDL コードは、インスタンスエートすることで最上位の回路図デザインに組み込み、残りのデザインと共にコンパイルします。

このセクションでは、新規 HDL モジュールを作成します。このマクロは、strtstop、mode、および lap_load 入力をデバウンスするために使用します。

New Source Wizard および ISE Text Editor の使用

New Source Wizard を使用してコンポーネントの名前およびポートを指定し、新規ソース ファイルを作成した後、作成された HDL ファイルを ISE Text Editor で編集します。

ソース ファイルを作成するには、次の手順に従います。

1. [Project] → [New Source] をクリックします。
2. [VHDL Module] または [Verilog Module] を選択します。
3. [File Name] に「debounce」と入力します。
4. [Next] をクリックします。
5. [Define Module] ページで、次の手順に従って debounce コンポーネントの 2 つの入力ポート sig_in および clk と、1 つの出力ポート sig_out を入力します。
 - a. [Port Name] 列の最初の 3 つに「sig_in」、「clk」、「sig_out」と入力します。
 - b. [Direction] 列で sig_in と clk には [in] または [input]、sig_out には [out] または [output] を選択します。

- c. [Bus] 列のチェック ボックスはオフのままにします。

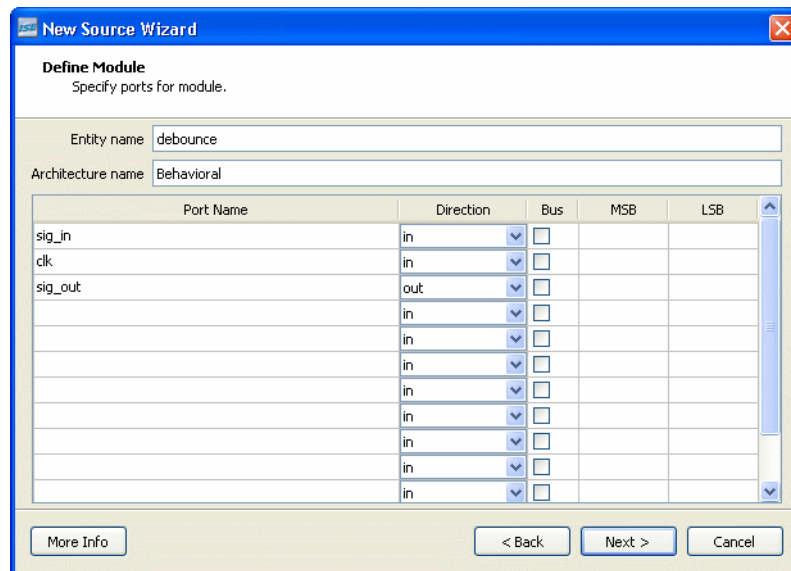


図 3-20 : New Source Wizard の [Define Module] ページ

6. [Next] をクリックします。[Summary] ページにモジュールの説明が表示されます。

7. [Finish] をクリックします。ISE Text Editor で空の HDL ファイルが開きます。

次の図に VHDL ファイルを示します。

```

7  -- Module Name:      debounce - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity debounce is
31     Port ( sig_in : in  STD_LOGIC;
32           clk : in  STD_LOGIC;
33           sig_out : out STD_LOGIC);
34 end debounce;
35
36 architecture Behavioral of debounce is
37
38 begin
39
40
41 end Behavioral;
42

```

図 3-21 : ISE Text Editor に表示された VHDL ファイル

次の図に VHDL ファイルを示します。

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    14:12:53 03/15/2007
7  // Design Name:
8  // Module Name:    debounce
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21 module debounce(sig_in, clk, sig_out);
22     input sig_in;
23     input clk;
24     output sig_out;
25
26
27 endmodule
28
```

図 3-22 : ISE Text Editor に表示された Verilog ファイル

ISE Text Editor に表示される HDL ファイルでは、既にポートが宣言されており、基本ファイル構造の一部が記述されています。キーワードは青色、データタイプは赤色、コメントは緑色、値は黒色で表示されています。このようにコードが色分けして表示されるので、コードが読みやすくなり、誤字エラーを見つけやすくなります。

言語テンプレートの使用

ISE の言語テンプレートには、カウンタ、D フリップフロップ、マルチプレクサ、プリミティブなど、よく使用されるロジックコンポーネントを記述した HDL 構文および合成テンプレートが含まれています。このセクションでは、デバウンス回路のテンプレートを使用します。

メモ：よく使用するコンポーネントまたは構文を言語テンプレートに追加することもできます。

言語テンプレートのウィンドウを表示して、このチュートリアルテンプレートを選択するには、次の手順に従います。

1. [Edit] → [Language Templates] をクリックします。

VHDL および Verilog の言語テンプレートは、それぞれ [Common Constructs]、[Device Macro Instantiation]、[Device Primitive Instantiation]、[Simulation Constructs]、[Synthesis Constructs]、[User Templates] のカテゴリに分類されています。これらのカテゴリを展開するには、プラス記号 (+) をクリックします。テンプレートをクリックすると、右側にテンプレートが表示されます。

2. [VHDL] または [Verilog] の階層から [Synthesis Constructs] → [Coding Examples] → [Misc] を展開し、[Debounce circuit] (VHDL) または [One Shot, Debounce Circuit] (Verilog) テンプレートを選択します。テンプレートが使用している言語のものであることを確認してください。

テンプレートを選択すると、右側にデバウンス回路の HDL コードが表示されます。

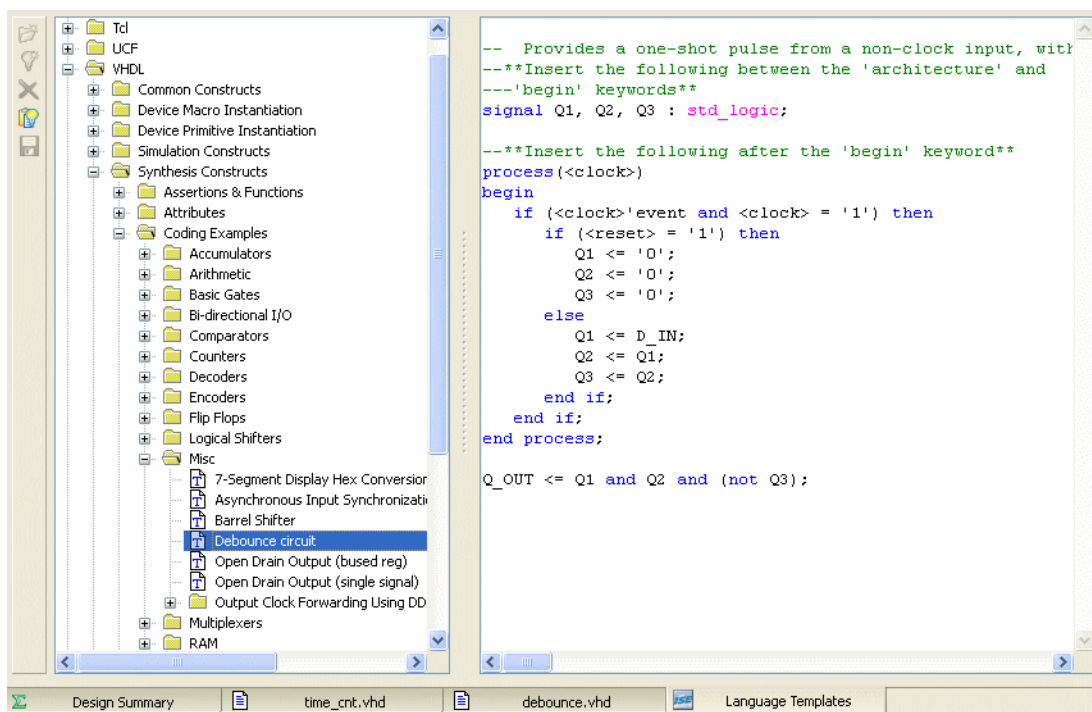


図 3-23 : 言語テンプレート

言語テンプレートのファイルへの追加

次に、[Use in File] コマンドを使用して、テンプレートを HDL ファイルに追加します。ドラッグアンドドロップで追加する方法など、言語テンプレートに関する詳細は、ISE ヘルプの「言語テンプレートの使用」を参照してください。

テンプレートを HDL ファイルに追加するには、次の手順に従います。

1. debounce.v または debounce.vhd ソース ファイルをアクティブにし、カーソルを VHDL ファイルでは architecture の begin 文の下、Verilog ファイルでは module およびピン宣言の下に置きます。
2. [Language Templates] ウィンドウに戻り、テンプレート インデックスで [Debounce circuit] テンプレートを右クリックし、[Use in File] をクリックします。

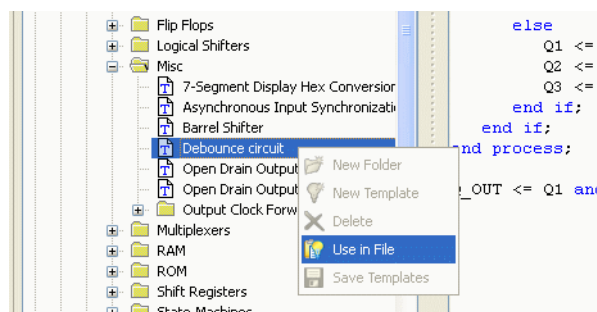


図 3-24 : テンプレートを [Use in File] を使用して HDL ファイルに追加

3. [Language Templates] ウィンドウを閉じます。

4. `debounce.v` または `debounce.vhd` ソース ファイルを開き、言語テンプレートが正しく挿入されていることを確認します。
5. Verilog のみ：次の変更を加えて Verilog モジュールを完成させます。
 - a. 「if」から「else」までの 3 行を削除して、リセット ロジックを削除します。リセット ロジックは、このデザインでは使用されません。
 - b. `<reg_name>` を 6 箇所すべてで `q` に変更します。
 - c. `<clock>` を `clk` に、`<input>` を `sig_in` に、`<output>` を `sig_out` に変更します。

メモ：[Edit] → [Find & Replace] をクリックすると、これを簡単に実行できます。ISE Text Editor の下部に検索フィールドが表示されます。
6. VHDL のみ：次の変更を加えて、VHDL モジュールを完成させます。
 - a. 「signal」で始まる行を `architecture` 行と `begin` 行の間に移動します。
 - b. 「if (`<reset>`...)」から「else」までの 5 行を削除して、リセット ロジックを削除します。2 つある「end if;」行の 1 つも削除します。
 - c. `<clock>` を `clk` に、`D_IN` を `sig_in` に、`Q_OUT` を `sig_out` に変更します。

メモ：[Edit] → [Find & Replace] をクリックすると、これを簡単に実行できます。ISE Text Editor の下部に検索フィールドが表示されます。
7. [File] → [Save] をクリックしてファイルを保存します。
8. [Hierarchy] ペインの `debounce` インスタンスの 1 つを選択します。
9. [Processes] ペインで [Check Syntax] をダブルクリックし、構文にエラーがないかどうかを確認します。必要に応じエラーを修正します。
10. ISE Text Editor を閉じます。

HDL モジュールの回路図シンボルの作成

次の手順に従って、`debounce` および `statmach` HDL ファイルの回路図シンボルを作成します。

1. [Design] パネルの [Hierarchy] ペインで、`debounce.vhd` または `debounce.v` を選択します。
2. [Processes] ペインで [Design Utilities] を展開し、[Create Schematic Symbol] をダブルクリックします。
3. `statmach.vhd` ファイルに対して、同じ操作を実行します。

これで、`stopwatch` 回路図に追加するシンボルが作成されました。

statmach、timer_preset、dcm1、debounce シンボルの配置

statmach、timer_preset、dcm1、および debounce シンボルを stopwatch 回路図 (stopwatch.sch) に配置します。

シンボルを配置するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、[stopwatch.sch] をダブルクリックします。
回路図ファイルがワークスペースに開きます。
2. [Add] → [Symbol] をクリックするか、またはツールバーの [Add Symbol] ボタンをクリックします。



図 3-25 : ツールバーの [Add Symbol] ボタン

回路図の左側の [Options] パネルが表示されているエリアに、[Symbols] パネルが開き、ライブラリおよび対応するコンポーネントが表示されます。

3. [Categories] でプロジェクト ディレクトリを選択します。
メモ : timer_preset のシンボルはプロジェクト ディレクトリ の ipcore_dir に含まれています。
4. [Symbols] でシンボルを選択して、図 3-26 に示すように回路図の適切な位置に配置します。
メモ : ここでは、シンボルを接続するワイヤは描画しません。コンポーネントは、後で接続します。
5. 回路図を保存します。

次の図に、シンボルが配置された stopwatch 回路図を示します。

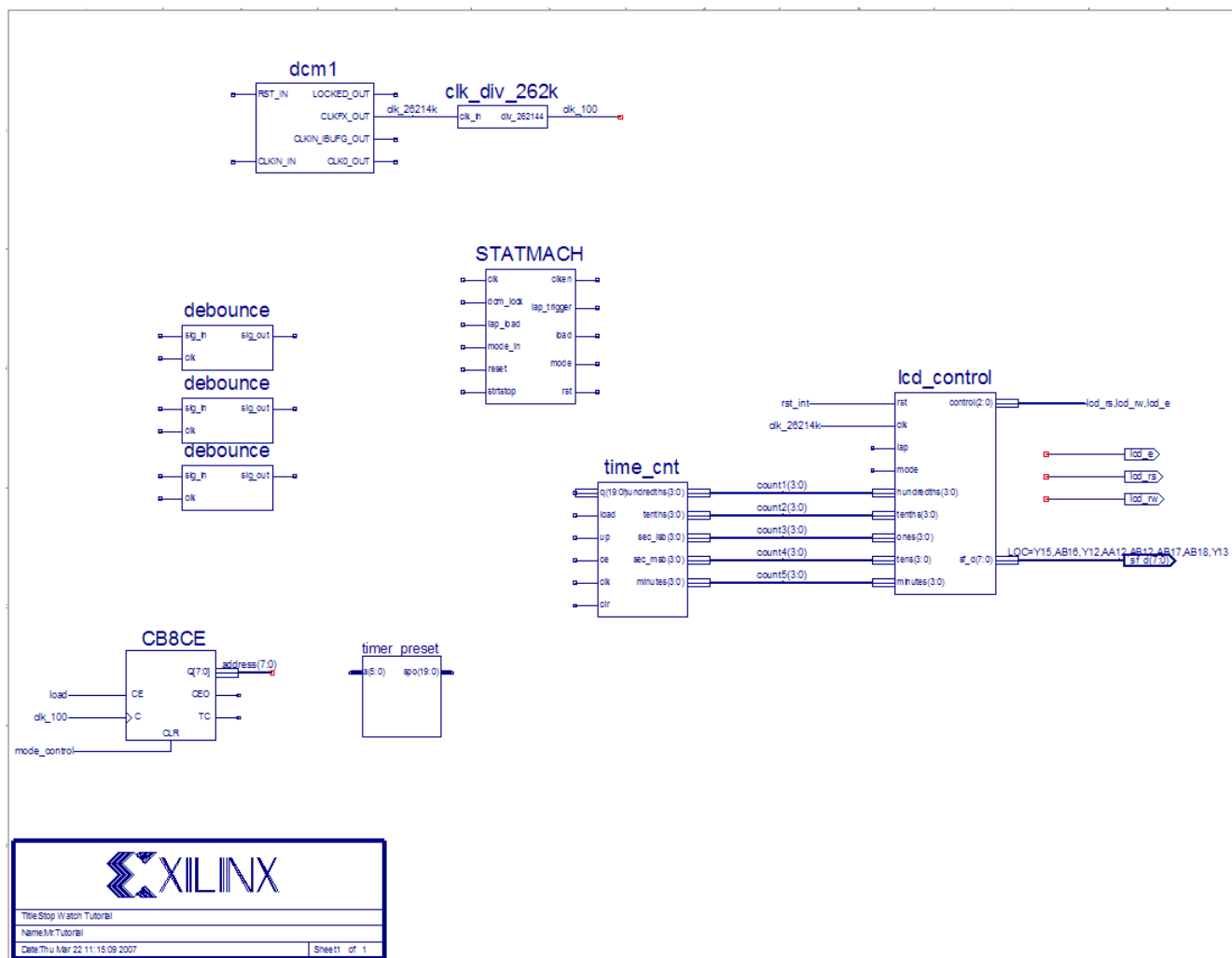


図 3-26：デザイン マクロの配置

インスタンス名の変更

回路図シートにシンボルを配置すると、接頭辞 `XLXI_` で始まる固有のインスタンス名が付けられます。Project Navigator の [Hierarchy] ペインに表示される階層をわかりやすくするため、次の手順に従ってシンボル インスタンスの名前を変更します。

1. `dcm1` シンボル インスタンスを右クリックし、[Object Properties] をクリックします。
2. [Object Properties] ダイアログ ボックスで、[InstName] の [Value] を「`dcm_inst`」に変更し、[OK] をクリックします。
3. 手順 1 ～ 2 を繰り返し、次のようにシンボル インスタンス名を変更します。
 - ◆ `statmach` インスタンス名 → `timer_state`
 - ◆ 1 番上の `debounce` インスタンス名 → `lap_load_debounce`
 - ◆ 2 番目の `debounce` インスタンス名 → `mode_debounce`
 - ◆ 1 番下の `debounce` インスタンス名 → `strtstop_debounce`
 - ◆ `timer_preset` インスタンス名 → `t_preset`
 - ◆ `time_cnt` インスタンス名 → `timer_cnt`

上位および下位階層への移動

階層で「プッシュ ダウン」を実行し、階層の下位のファイルを表示します。このチュートリアルで作成した回路図ベースのマクロ `time_cnt` を表示し、このファイルに含まれるコンポーネントを確認します。

最上位の `stopwatch` 回路図から `time_cnt` を表示するには、次の手順に従います。

1. 回路図で `time_cnt` シンボルをクリックし、ツールバーの [Push] ボタンをクリックするか、またはシンボルを右クリックして [Symbol] → [Push into Symbol] をクリックします。



図 3-27: ツールバーの [Push] ボタン

`time_cnt` 回路図には、5 つのカウンタ ブロックがあります。いずれかを選択して [Push] ボタンをクリックし、カウンタ マクロを表示します。回路図ページにザイリンクスプリミティブ コンポーネントのみが表示されるまで、このプロセスを繰り返すことができます。基になる下位ファイルが HDL または IP コア ファイルであるシンボルで [Push] ボタンをクリックすると、適切なテキスト エディタまたはカスタマイズ ウィンドウが開き、そこでファイルを編集できます。

2. マクロを確認したら、[View] → [Pop to Calling Schematic] をクリックするか、または回路図で何も選択していない状態でツールバーの [Pop] ボタンをクリックします。回路図の空白部分を右クリックして [Pop to Calling Schematic] をクリックしても、同じ操作を実行できます。



図 3-28: ツールバーの [Pop] ボタン

デバイスの入力および出力の指定

I/O マーカーを使用して、回路図のデバイス I/O を指定します。Schematic Editor の回路図は、すべて VHDL または Verilog のネットリストとして出力され、合成ツールで合成されます。合成ツールで最上位の回路図 HDL が合成されると、I/O マーカーは適切なパッドおよびバッファで置き換えられます。

入力ピンの追加

次に、stopwatch 回路図に入力ピン `reset`、`clk`、`lap_load`、`mode`、および `strtstop` を追加します。

これらのコンポーネントを追加するには、`dcm1` の 2 つの入力と、各 `debounce` シンボルの `sig_in` ピンに未接続のワイヤを描画します。

メモ：ワイヤの描画方法は、「[ワイヤの描画](#)」を参照してください。

I/O マーカーおよびネット名の追加

ネットおよびバスに名前を付けることは、次のような理由から重要です。

- デバッグおよびシミュレーションで、簡単にデザイン内のネットをトレースできる。
例えば、インプリメンテーション プロセスでは、名前が付けられていないネットに対して、ユーザーには意味をなさない任意の名前が付けられます。
- デザインを解読しやすくなり、デザインの記録に役立つ。

描画した 5 つの入力ネットに名前を付けます。[図 3-31](#) の完成した回路図を参照してください。`reset` ネットに名前を付けるには、次の手順に従います。

- [Add] → [Net Name] をクリックします。
- [Options] パネルの [Add Net Name Options] で、[Name] に「reset」と入力します。
入力したネット名がカーソルに表示されます。
- [図 3-29](#) に示すように、ネットの左端をクリックして名前を配置します。
- `clk`、`lap_load`、`mode`、および `strtstop` ピンに対して同じ手順を繰り返します。
すべてのネットに名前を付けたら、I/O マーカーを追加します。
- [Add] → [I/O Marker] をクリックします。

6. 次の図に示すように追加した 5 つのネット名の周りにボックスを描くようドラッグし、各ネットに I/O マーカーを追加します。

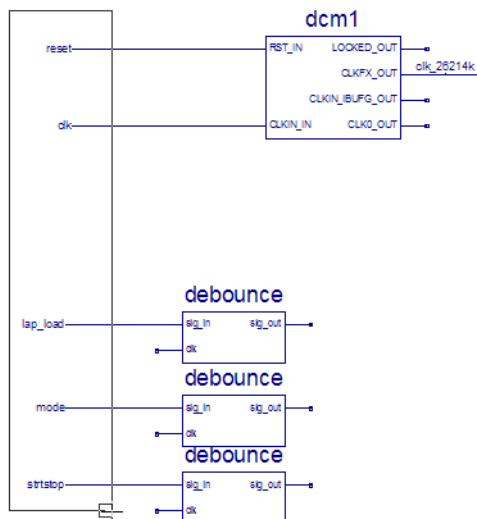


図 3-29 : 名前の付けられたネットへの I/O マーカーの追加

ピン ロケーションの割り当て

ザイリンクスでは、PAR (自動配置配線) プログラムを使用してデザインのピン配置を定義することを推奨します。ピンにロケーションを事前に割り当てると、PAR のパフォーマンスが低下する可能性があります。ただし、PCB (プリント回路基板) にデザインのピン配置を統合できるように、ある時点で配置を固定する必要がある場合もあります。

このチュートリアルでは、デザインを Spartan-3A デモ ボードに配置、ダウンロードするために入力および出力を特定のピンに固定します。このチュートリアルのデザインは単純でタイミングも重要ではないため、ピンを割り当てても PAR によるデザインの配置配線に悪影響はありません。

stopwatch 回路図の出力ネットに LOC パラメータを割り当てするには、次の手順に従います。

1. clk のネットを右クリックし、[Object Properties] をクリックします。
2. [Object Properties] ダイアログ ボックスで [New] をクリックします。
3. [New Attribute] ダイアログ ボックスで、[Attribute Name] に「LOC」、[Attribute Value] に「E12」と入力します。

4. [OK] をクリックして [Object Properties] ダイアログ ボックスに戻ります。

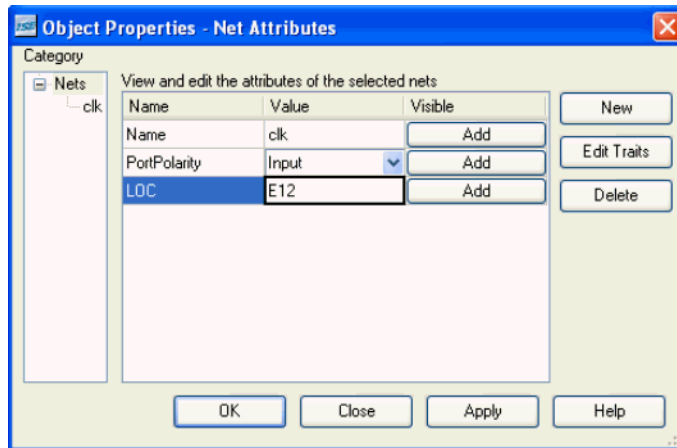


図 3-30：ピン ロケーションの割り当て

5. LOC 属性を表示するには、[LOC] 属性の横にある [Add] をクリックします。
6. [Net Attribute Visibility] ダイアログ ボックスで、表示されているネットの中心付近をクリックし、次に [OK] をクリックします。

これで、LOC 属性が回路図の clk ネットの上に表示されます。

7. [OK] をクリックして [Object Properties] ダイアログ ボックスを閉じます。

上記の手順により、clk がピン E12 に制約されます。sf_d(7:0) バスにも LOC プロパティが追加されていることを確認してください。残りのピン ロケーション制約は、第 5 章「デザイン インプリメンテーション」の「Constraints Editor の使用」および「PlanAhead ソフトウェアを使用した I/O ロケーションの割り当て」で追加します。

メモ： ロケーション制約を削除せずにオフにするには、[Object Properties] ダイアログ ボックスで LOC 属性を選択して [Edit Traits] をクリックします。[Attribute Traits] ダイアログ ボックスの [Category] で [VHDL] または [Verilog] をクリックし、[Ignore this attribute] をオンにします。

回路図の完成

作成して配置したコンポーネントをワイヤで接続し、必要に応じてロジックを追加し、ネットに名前を付けて回路図を完成させます。これには、次の手順に従います。次に示されている完成した回路図を参照してください。このセクションの手順に含まれる各操作の詳細は、ここまでのセクションを参照してください。

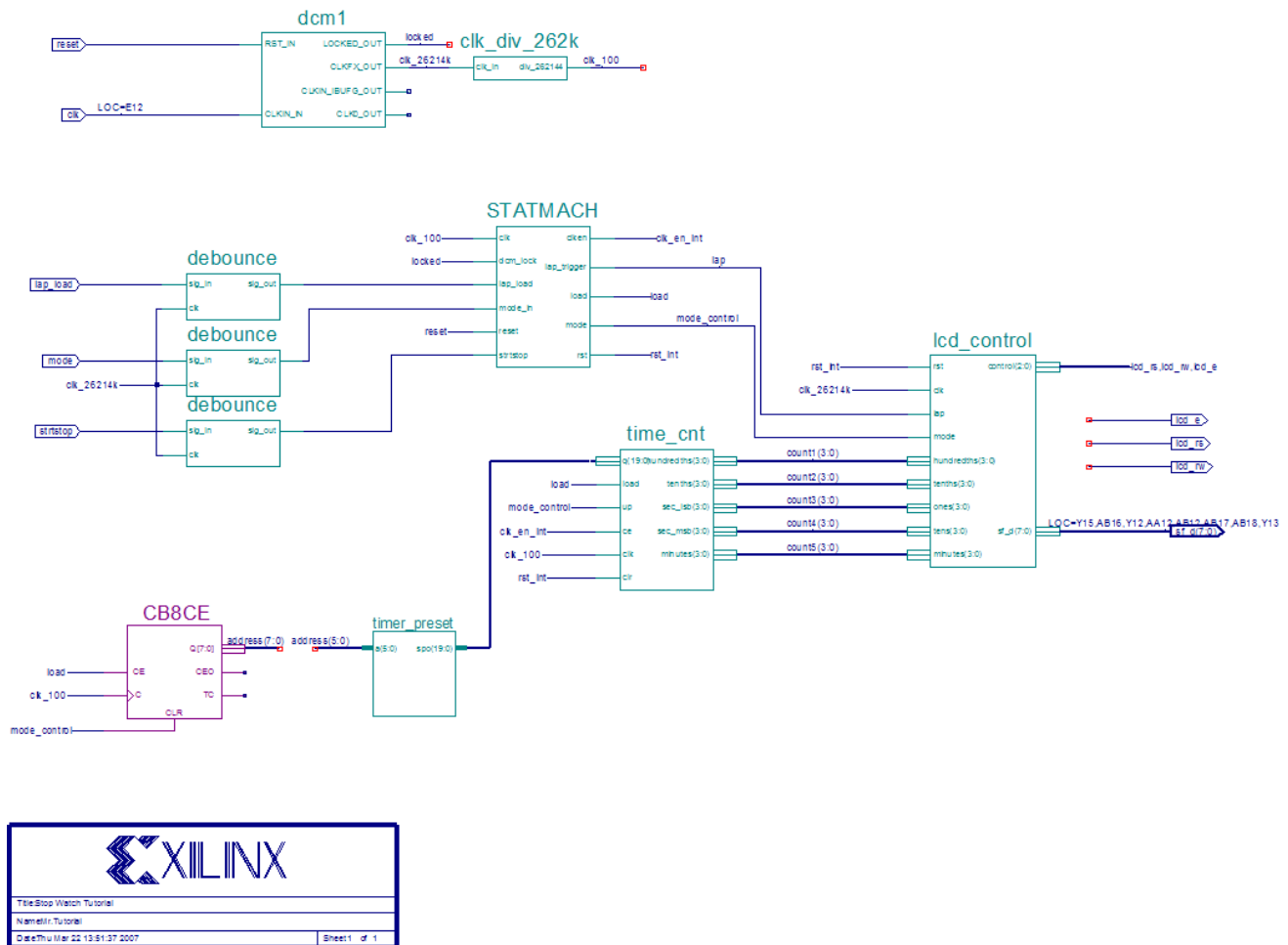


図 3-31 : 完成した stopwatch デザインの回路図

回路図を完了させるには、次の手順に従います。

1. dcm1 の LOCKED_OUT ピンに未接続のワイヤを描画し、「locked」という名前を付けます (「[ワイヤの描画](#)」および「[ネット名の追加](#)」を参照)。
2. time_cnt および statmach マクロの clk 入力に未接続ワイヤを描画します (「[ワイヤの描画](#)」を参照)。
3. このワイヤの両方に「clk_100」という名前を付けます (「[ネット名の追加](#)」を参照)。
メモ： ネットは名前が同一の場合、回路図で物理的に接続されていなくても、論理的に接続されることに注意してください。この手法を使用して、clk_100 とその他の信号を論理的に接続します。
4. 3 つの debounce マクロの clk 入力にワイヤを描画し、「clk_26214k」という名前を付けます。
5. 各 debounce コンポーネントの sig_out ピンと statmach マクロの lap_load、mode_in、および strtstop ピンの間にそれぞれワイヤを描画し、これらのネットに「ll_debounced」、「mode_debounced」、「strtstop_debounced」という名前を付けます (「[ワイヤの描画](#)」および「[ネット名の追加](#)」を参照)。
6. statmach マクロの dcm_lock ピンと reset ピンにそれぞれ未接続ワイヤを描画し、「locked」および「reset」という名前を付けます。
7. statmach コンポーネントの clken 出力と time_cnt コンポーネントの ce ピンに未接続ワイヤを描画し、両方に「clk_en_int」という名前を付けます。
8. statmach マクロの rst 出力ピンおよび time_cnt マクロの clr ピンに未接続ワイヤを描画し (「[ワイヤの描画](#)」を参照)、両方に「rst_int」と名前を付けます。
9. timer_preset マクロのバス出力から time_cnt マクロの q(19:0) 入力にワイヤを描画します (「[ワイヤの描画](#)」を参照)。ワイヤは自動的にバスに変換されます。
10. timer_preset マクロの入力に未接続バスを描画し、「address(5:0)」と名前を付けます。
11. statmach マクロの lap_trigger および mode 出力から lcd_control マクロの lap および mode 入力にワイヤを描画し (「[ワイヤの描画](#)」を参照)、それぞれ「lap」および「mode_control」という名前を付けます。
12. statmach マクロの load 出力および time_cnt マクロの load 入力に未接続ワイヤを描画し、(「[ワイヤの描画](#)」を参照)、両方に「load」という名前を付けます。
13. time_cnt マクロの up 入力に未接続のワイヤを描画し (「[ワイヤの描画](#)」を参照)、「mode_control」という名前を付けます。
14. [File] → [Save] をクリックしてデザインを保存します。

これで回路図デザインが完了しました。

回路図のフローを継続するには、次のいずれかの操作を実行します。

- [第 4 章「ビヘイビア シミュレーション」](#) を参照して合成前のシミュレーションを実行します。
- [第 5 章「デザイン インプリメンテーション」](#) を参照して配置配線を実行します。

ビヘイビア シミュレーション

この章は、次のセクションから構成されています。

- 「ビヘイビア シミュレーション フローの概要」
- 「ModelSim の設定」
- 「ISim の設定」
- 「入門」
- 「HDL テストベンチの追加」
- 「ModelSim を使用したビヘイビア シミュレーション」
- 「ISim を使用したビヘイビア シミュレーション」

ビヘイビア シミュレーション フローの概要

ザイリンクス ISE® Design Suite には、Mentor Graphics 社の ModelSim シミュレータおよびザイリンクスの ISim シミュレータを使用するフローが統合されており、Project Navigator からシミュレーションを実行できます。このチュートリアルでは、統合フローを使用する例を示します。このチュートリアルでは、ModelSim シミュレータまたは ISim シミュレータのどちらを使用しても、同じシミュレーション結果を達成できます。

シミュレーションの詳細およびサポートされるほかのシミュレータについては、『合成/シミュレーション デザイン ガイド』を参照してください。このガイドは ISE ソフトウェア マニュアル コレクションに含まれています。ソフトウェア マニュアル コレクションを開くには、[Help] → [Software Manuals] をクリックするか、ザイリンクス Web サイトの [ソフトウェア マニュアル コレクション](#) にアクセスします。

このチュートリアルでは、Project Navigator 内でシミュレーション フローを実行する方法、ModelSim および ISim シミュレータの主な機能を説明します。これらシミュレータの使用方法的詳細は、[ModelSim の Web サイト](#) から ModelSim の資料、またはザイリンクス Web サイトの ISE Design Suite 12 [チュートリアル ページ](#) から ISim チュートリアルを参照してください。

ModelSim の設定

このチュートリアルを実行するには、ModelSim をインストールする必要があります。ModelSim PE、ModelSim SE、および ModelSim DE は ModelSim の正規版であり、Mentor Graphics 社から直接購入できます。ISE Design Suite 12 のライブラリを使用してシミュレーションするには、ModelSim 6.5c 以降を使用してください。これより前のバージョンでもシミュレーションを実行できる可能性はありますが、サポートされていません。ModelSim PE、SE、および DE の詳細は、Mentor Graphics 社にお問い合わせください。

ISim の設定

ISim は、ISE Design Suite 12 インストーラによりサポートされている OS 上に自動的にインストールおよび設定されます。ISim でサポートされる OS の一覧は、ザイリンクスの Web サイトから『[ISE Design Suite 12: インストール、ライセンス、リリース ノート](#)』を参照してください。

入門

次に、このチュートリアルでビヘイビア シミュレーションを実行するための要件を示します。

必要なファイル

ビヘイビア シミュレーションには、デザイン ファイル、テストベンチ ファイル、ザイリンクス シミュレーション ライブラリが必要です。

デザイン ファイル (VHDL、Verilog、または回路図)

この章では、第 2 章「HDL ベースのデザイン」または第 3 章「回路図ベースのデザイン」に従ってデザイン入力チュートリアルを完了していることを前提としています。これらの章のチュートリアルに従うと、必要なデザイン ファイルを作成できます。

テストベンチ ファイル

デザインをシミュレーションするには、デザインにスティミュラスを供給するテストベンチが必要です。チュートリアル ファイルには、VHDL および Verilog のテストベンチ ファイルが含まれています。また、独自のテストベンチ ファイルを作成することも可能です。

シミュレーション ライブラリ

デザインにザイリンクス プリミティブまたは IP コアがインスタンシエートされている場合は、ザイリンクス シミュレーション ライブラリが必要です。このチュートリアルのデザインには、デジタル クロック マネージャ (DCM) および CORE Generator™ ソフトウェア コンポーネントがインスタンシエートされているので、シミュレーション ライブラリを使用する必要があります。シミュレーション ライブラリに関する情報とコンパイル方法は、次の「[ザイリンクス シミュレーション ライブラリ](#)」セクションで説明します。

ザイリンクス シミュレーション ライブラリ

ザイリンクス プリミティブ、CORE Generator ソフトウェア コンポーネント、およびその他のザイリンクス IP コアがインスタンシエートされているデザインをシミュレーションするには、ザイリンクス シミュレーション ライブラリを使用する必要があります。これらのライブラリには、各コンポーネントのモデルが含まれています。モデルは各コンポーネントの機能を表したもので、シミュレーションに必要な情報をシミュレータに提供します。

各ライブラリの詳細は、『合成/シミュレーション デザイン ガイド』を参照してください。このガイドは ISE ソフトウェア マニュアル コレクションに含まれています。ソフトウェア マニュアル コレクションを開くには、[Help] → [Software Manuals] をクリックするか、ザイリンクス Web サイトの[ソフトウェア マニュアル コレクション](#)にアクセスします。

ザイリンクス シミュレーション ライブラリのアップデート

ザイリンクス シミュレーション ライブラリに含まれるモデルは、定期的にアップデートされます。

- XilinxCoreLib モデルは、IP アップデートをインストールするとアップデートされます。
- その他のモデルは、ソフトウェア アップデートをインストールするとアップデートされます。

モデルがアップデートされたら、ライブラリを再コンパイルする必要があります。シミュレーション ライブラリをコンパイルすると、シミュレーションでモデルを使用できるようになります。

ModelSim PE、SE、または DE

ModelSim PE、SE、または DE を使用している場合は、シミュレーション ライブラリをアップデートされたモデルで再コンパイルする必要があります。詳細は、『合成/シミュレーション デザイン ガイド』を参照してください。このガイドは ISE ソフトウェア マニュアル コレクションに含まれています。ソフトウェア マニュアル コレクションを開くには、[Help] → [Software Manuals] をクリックするか、ザイリンクス Web サイトの [ソフトウェア マニュアル コレクション](#) にアクセスします。

ザイリンクス ISim

ISim 用のアップデートされたシミュレーション ライブラリはあらかじめコンパイルされており、ISE ソフトウェアと共にインストールされます。

Modelsim.ini ファイルでのシミュレーション ライブラリのマップ

ModelSim では、modelsim.ini ファイルによりコンパイル済みライブラリの場所を判断します。たとえば、UNISIM ライブラリを C:\lib\UNISIM にコンパイルした場合、modelsim.ini ファイルに次のように記述されます。

```
UNISIM = c:\lib\UNISIM
```

メモ：modelsim.ini は、ISim には適用されません。

ModelSim は、modelsim.ini ファイルが見つかるまで、次の場所をリストした順に検索します。

- MODELSIM 環境変数で指定された modelsim.ini ファイル
- 現在の作業ディレクトリ
- ModelSim がインストールされたディレクトリ

MODELSIM 環境変数が設定されておらず、作業ディレクトリに modelsim.ini ファイルをコピーしていない場合は、ModelSim インストール ディレクトリにある modelsim.ini ファイルが使用されます。

ModelSim PE、SE、または DE

ModelSim PE、SE、または DE を使用している場合は、『コマンド ライン ツール ユーザー ガイド』を参照し、COMPXLIB を使用してライブラリをコンパイルする必要があります。このガイドは ISE ソフトウェア マニュアル コレクションに含まれています。ソフトウェア マニュアル コレクションを開くには、[Help] → [Software Manuals] をクリックするか、ザイリンクス Web サイトの [ソフトウェア マニュアル コレクション](#) にアクセスします。

COMPXLIB を実行すると、modelsim.ini ファイルも正しいライブラリ マップ情報にアップデートされます。modelsim.ini ファイルを開いて、ライブラリのマップが正しいかどうかを確認してください。

modelsim.ini ファイルを作業ディレクトリにコピーしてプロジェクト 用に変更したり、MODELSIM 環境変数で modelsim.ini ファイルを設定することもできます。

ISim

modelsim.ini ファイルは、ISim には適用されません。

HDL テストベンチの追加

デザイン プロジェクトに HDL テストベンチを追加するには、このチュートリアルに含まれるテストベンチ ファイルを追加するか、または独自のテストベンチ ファイルを作成してプロジェクトに追加します。

チュートリアル のテストベンチ ファイルの追加

このセクションでは、既存のテストベンチをプロジェクトに追加する方法を示します。このチュートリアルには、VHDL テストベンチおよび Verilog テストベンチが含まれています。

メモ：ISE ソフトウェアで独自のテストベンチ ファイルを作成するには、[Project] → [New Source] をクリックし、New Source Wizard でソース タイプとして [VHDL Test Bench] または [Verilog Text Fixture] を選択します。この操作で空のスティミュラス ファイルがプロジェクトに追加されるので、テキスト エディタでテストベンチを定義する必要があります。

VHDL シミュレーション

チュートリアル VHDL テストベンチをプロジェクトに追加するには、次の手順に従います。

1. Project Navigator で、[Project] → [Add Source] をクリックします。
2. テストベンチ ファイル stopwatch_tb.vhd を選択します。
3. [開く] をクリックします。
4. [Adding Source Files] ダイアログボックスで、stopwatch_tb.vhd ファイルが [Simulation] に関連付けられていることを確認します。
5. [OK] をクリックします。

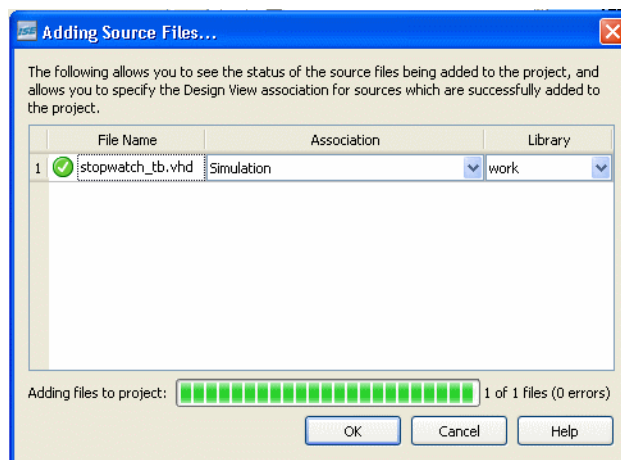


図 4-1 : VHDL テストベンチの追加

Verilog シミュレーション

Verilog テストベンチをプロジェクトに追加するには、次の手順に従います。

1. Project Navigator で、[Project] → [Add Source] をクリックします。
2. テストベンチ ファイル stopwatch_tb.v を選択します。
3. [開く] をクリックします。
4. [Adding Source Files] ダイアログボックスで、stopwatch_tb.vhd ファイルが [Simulation] に関連付けられていることを確認します。
5. [OK] をクリックします。

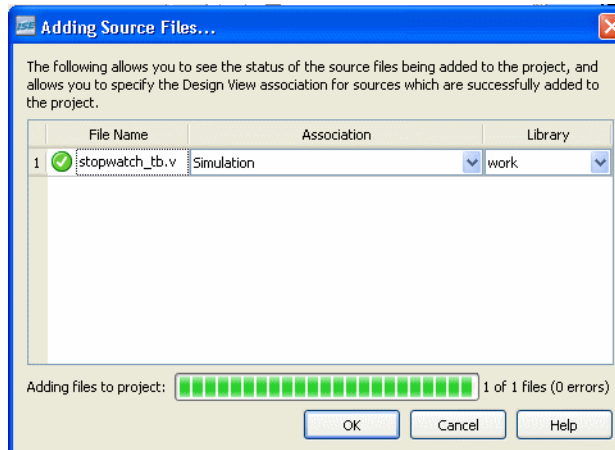


図 4-2 : Verilog テストベンチの追加

ModelSim を使用したビヘイビア シミュレーション

プロジェクトにテストベンチを追加したら、ModelSim シミュレータを使用してデザインのビヘイビア シミュレーションを実行します。ModelSim シミュレータは、ISE ソフトウェアと完全に統合されています。ISE ソフトウェアから ModelSim による作業ディレクトリの作成、ソースファイルのコンパイル、デザインの読み込み、シミュレーションプロパティを使用したシミュレーションの実行が可能です。

ISim でシミュレーションを実行する場合は、「ISim を使用したビヘイビア シミュレーション」に進んでください。このチュートリアルでは、ModelSim シミュレータまたは ISim シミュレータのどちらを使用しても、同じ結果を達成できます。

ModelSim をプロジェクトのシミュレータとして選択するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、デバイス名 (xc3s700A-4fg484) を右クリックして [Design Properties] をクリックします。
2. [Design Properties] ダイアログボックスの [Simulator] で ModelSim のタイプと言語の組み合わせを選択します。

ISE のシミュレーション プロセス

ISE ソフトウェアのシミュレーション プロセスを使用すると、ModelSim を使用したデザインのシミュレーションを実行できます。ModelSim シミュレータ用のプロセスを見つけるには、次の手順に従います。

1. [Design] パネルの [View] ペインで [Simulation] をオンにし、ドロップダウン リストから [Behavioral] を選択します。
2. [Hierarchy] ペインで、テストベンチ ファイル (stopwatch_tb) を選択します。
3. [Processes] ペインで [ModelSim Simulator] を展開してプロセスを表示します。

ここにある [Simulate Behavioral Model] プロセスで、デザインのシミュレーションを実行できます。

ModelSim がインストールされているのにプロセスが表示されない場合は、Project Navigator のプリファレンスが正しく設定されていない可能性があります。ModelSim の場所を指定するには、次の手順に従います。

1. [Edit] → [Preferences] をクリックします。
2. [Preferences] ダイアログ ボックスの [Category] で [ISE General] → [Integrated Tools] をクリックします。
3. [Model Tech Simulator] で ModelSim の実行ファイルを指定します。たとえば、`C:\modeltech_xe\win32xoem\modelsim.exe` のように指定します。

シミュレーション プロパティの指定

stopwatch デザインのビヘイビア シミュレーションを実行する前に、シミュレーションのプロセス プロパティを設定します。

ISE ソフトウェアでは、ModelSim シミュレータのプロパティおよびシミュレーション ネットリストのプロパティを設定できます。このチュートリアル のビヘイビア シミュレーションのプロパティを表示/変更するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、テストベンチ ファイル (stopwatch_tb) を選択します。
2. [Processes] ペインで [ModelSim Simulator] を展開し、[Simulate Behavioral Model] を右クリックして [Process Properties] をクリックします。
3. [Process Properties] ダイアログ ボックス (図 4-3) で、[Property display level] が [Advanced] に設定されていることを確認します。

[Advanced] に設定すると、すべてのプロパティが表示されます。

4. [Simulation Run Time] を「2000 ns」に変更します。

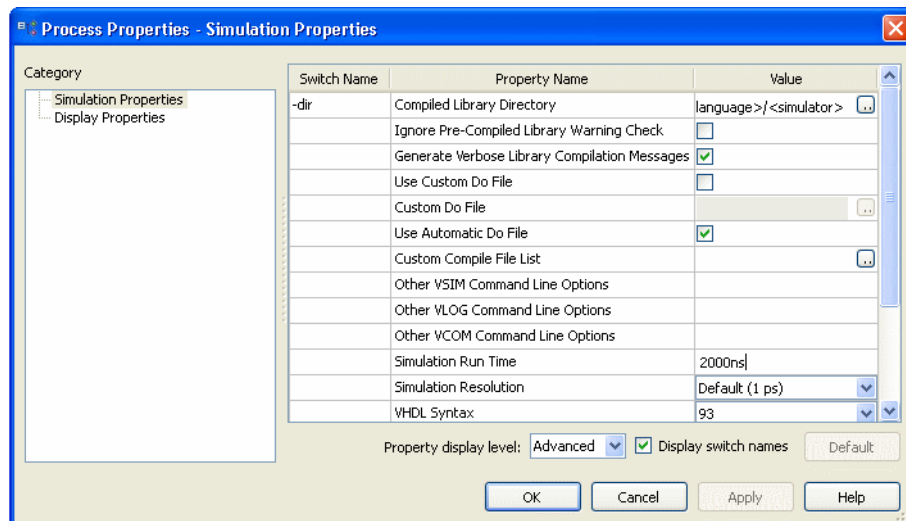


図 4-3 : ビヘイビア シミュレーションのプロセス プロパティ

5. [OK] をクリックします。

メモ : 各プロパティの詳細な説明は、[Process Properties] ダイアログ ボックスで [Help] ボタンをクリックしてください。

シミュレーションの実行

プロセス プロパティを設定したら、ModelSim を実行します。ビヘイビア シミュレーションを実行するには、[Simulate Behavioral Model] をダブルクリックします。ModelSim により、作業ディレクトリの作成、ソース ファイルのコンパイル、デザインを読み込みが実行され、指定した時間だけシミュレーションが実行されます。

このデザインの大部分は 100Hz で動作するので、シミュレーションに時間がかかります。RESET がディアサートされてから約 33ms 後に SF_D および LCD_E 制御信号が遷移し始めるので、短時間のシミュレーションではカウンタが動作していないように見えます。このチュートリアルでは、DCM の信号のみを調べて、正常に動作しているかどうかを検証します。

信号の追加

シミュレーション中の内部信号を表示するには、[Wave] ウィンドウに追加する必要があります。最上位のポートは、あらかじめ追加されています。その他の信号は、[Structure] ウィンドウの [Sim] タブでの選択に応じて [Objects] ウィンドウに表示されます。

[Wave] ウィンドウに信号を追加するには、次の 2 つの方法があります。

- [Objects] ウィンドウからドラッグ アンド ドロップする。
- [Objects] ウィンドウで信号をクリックし、[Add] → [Wave] → [Selected Signals] をクリックする。

次に、デザイン階層の信号を追加する方法を示します。このチュートリアルでは、波形に DCM 信号を追加します。

ModelSim のバージョン 6.0 以降を使用している場合は、すべてのウィンドウはデフォルトでメイン ウィンドウ内に表示されます。[Dock/Undock pane] ボタンをクリックすると、ウィンドウを切り離すことができます。



図 4-4 : [Dock/Undock pane] ボタン

デザイン階層の信号を追加するには、次の手順に従います。

1. [Sim] タブで、[uut] の横にあるプラス記号 (+) をクリックして階層を展開します。

次の図に、VHDL フローの [Sim] タブを示します。回路図または VHDL フローの [Sim] タブのレイアウトは、Verilog のものと異なる場合があります。

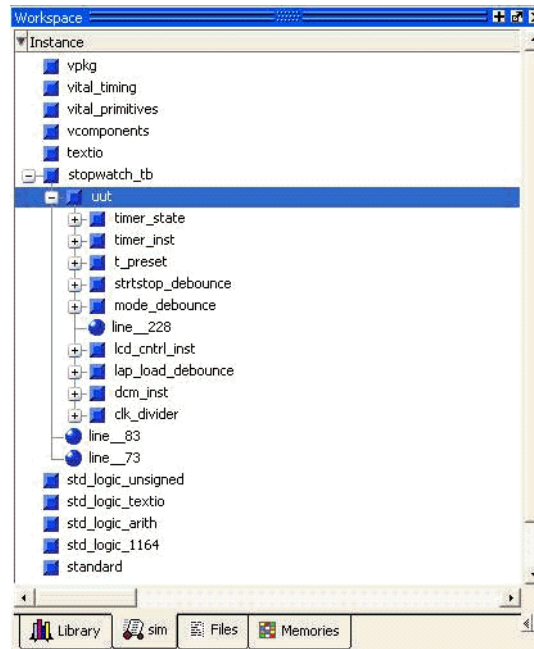


図 4-5 : [Sim] タブ (VHDL フロー)

2. [Sim] タブで、[dcm_inst] を選択します。[Objects] ウィンドウにリストされている信号がアップデートされます。
3. [CLKIN_IN] を、[Objects] ウィンドウから [Wave] ウィンドウにドラッグ アンド ドロップします。
4. [Object] ウィンドウで、次の信号を選択します。
 - ◆ RST_IN
 - ◆ CLKFX_OUT
 - ◆ CLK0_OUT
 - ◆ LOCKED_OUT

メモ：複数の信号を選択するには、Ctrl キーを押しながら信号をクリックします。

5. [Objects] ウィンドウを右クリックします。
6. [Add to Wave] → [Selected Signals] をクリックします。

仕切りの追加

ModelSim の [Wave] ウィンドウに仕切りを追加して、信号を区別しやすくします。「DCM Signals」という仕切りを追加するには、次の手順に従います。

1. 必要であれば、ウィンドウを切り離して最大化し、波形が大きく表示されるようにします。
2. [Wave] ウィンドウを右クリックし、[Insert Divider] をクリックします。
3. [Wave Divider Properties] ダイアログ ボックスの [Divider Name] に「DCM Signals」と入力します。
4. [OK] をクリックします。
5. 作成した仕切りを **CLKIN_IN** 信号の上にドラッグします。

仕切りを追加すると、次のような表示になります。

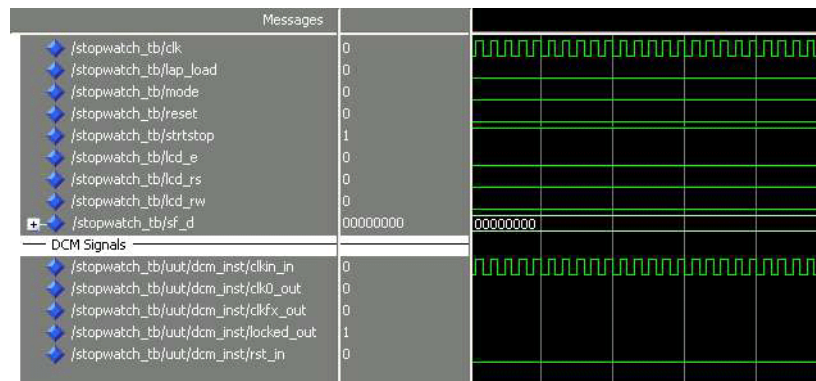


図 4-6 : 「DCM Signals」という仕切りを追加した後の [Wave] ウィンドウ

新しく追加した信号に対しては、波形は表示されていません。これは、ModelSim にこれらの信号のデータが記録されていないためです。デフォルトでは、シミュレーションを実行したときに [Wave] ウィンドウに追加されていた信号のデータのみが記録されるので、[Wave] ウィンドウに信号を追加した後、シミュレーションを再実行する必要があります。

シミュレーションの再実行

ModelSim でシミュレーションを再実行するには、次の手順に従います。

1. ツールバーの [Restart] ボタンをクリックします。



図 4-7 : [Restart] ボタン

2. [Restart] ダイアログ ボックスで [Restart] をクリックします。

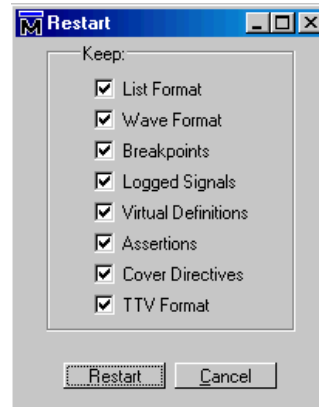


図 4-8 : [Restart] ダイアログ ボックス

3. ModelSim のコマンド プロンプトに「run 2000 ns」と入力します。
4. Enter キーを押します。

```
VSIM 5> run 2000 ns
```

図 4-9 : run コマンドの入力

シミュレーションが 2000ns 間実行され、DCM の波形が [Wave] ウィンドウに表示されます。

信号の解析

DCM 信号が予測どおりに動作しているかを検証します。CLK0_OUT は 50MHz、CLKFX_OUT は約 26MHz で動作する必要があります。DCM 出力は、LOCKED 信号が High になるまでは無効なので、LOCKED_OUT 信号が High になった後のものを解析する必要があります。

ModelSim では、波形にカーソルを追加して、遷移間の時間を確認できます。CLK0_OUT の周期を確認するには、次の手順に従います。

1. [Add] → [To Wave] → [Cursor] を 2 回クリックして 2 本のカーソルを追加します。
2. 1 つ目のカーソルを、LOCKED_OUT 信号が High になった後の、CLK0_OUT 信号の最初の立ち上がりエッジにドラッグします。
3. 2 つ目のカーソルを 1 つ目のカーソルのすぐ右にドラッグします。

4. [Find Next Transition] ボタンを 2 回クリックして、カーソルを CLK0_OUT 信号の次の立ち上がりエッジに移動します。



図 4-10 : [Find Next Transition] ボタン

5. 2 つのカーソル間の距離が、波形の下に表示されます。
結果は 20000ps となり、周波数に変換すると 50MHz になります。この値は、テストベンチからの入力周波数に等しく、DCM の CLK0 出力周波数は正しい値であると判断できます。
6. 同じ手順を使用して CLKFX_OUT を調べます。結果は 38462ps となり、周波数に変換すると約 26MHz になります。

シミュレーションの保存

信号またはステイミュラスを追加した後、またはシミュレーションを再実行した後に、[Wave] ウィンドウにリストされている信号を保存できます。保存した信号のリストは、シミュレーションを開始するときに簡単に開くことができます。

信号のリストを保存するには、次の手順に従います。

1. [Wave] ウィンドウを選択した状態で、[File] → [Save As] をクリックします。
2. [Save Format] ダイアログ ボックスで、ファイル名をデフォルトの wave.do から dcm_signal.do に変更します。
3. [OK] をクリックします。

シミュレーションを再実行するときに、[Wave] ウィンドウで [File] → [Load] をクリックしてこのファイルを読み込むことができます。

これで、ビヘイビア シミュレーションは完了しました。この後、[第 5 章「デザイン インプリメンテーション」](#)の手順に従って、デザインをインプリメントします。

ISim を使用したビヘイビア シミュレーション

ISim を使用してシミュレーションを実行する場合は、このセクションの手順に従ってください。

プロジェクトにテストベンチを追加したら、ISim を使用してデザインのビヘイビア シミュレーションを実行します。ISim は、ISE ソフトウェアと完全に統合されています。ISE ソフトウェアから ISim による作業ディレクトリの作成、ソース ファイルのコンパイル、デザインの読み込み、シミュレーション プロパティを使用したシミュレーションの実行が可能です。

ISim をプロジェクトのシミュレータとして選択するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、デバイス名 (xc3s700A-4fg484) を右クリックして [Design Properties] をクリックします。
2. [Design Properties] ダイアログ ボックスで [Simulator] を [ISim (VHDL/Verilog)] に設定します。

ISE のシミュレーション プロセス

ISE ソフトウェアのシミュレーション プロセスを使用すると、ISim を使用したデザインのシミュレーションを実行できます。ISim のプロセスを見つけるには、次の手順に従います。

1. [Design] パネルの [View] ペインで [Simulation] をオンにし、ドロップダウン リストから [Behavioral] を選択します。
2. [Hierarchy] ペインで、テストベンチ ファイル (stopwatch_tb) を選択します。
3. [Processes] ペインで [ISim Simulator] を展開してプロセスを表示します。

次のシミュレーション プロセスがあります。

- ◆ [Check Syntax]

テストベンチに構文エラーがないかどうかを確認します。

- ◆ [Simulate Behavioral Model]

デザインのビヘイビア シミュレーションを実行します。

シミュレーション プロパティの指定

stopwatch デザインのビヘイビア シミュレーションを実行する前に、シミュレーションのプロセス プロパティを設定します。

ISE ソフトウェアでは、ISim のプロパティおよびシミュレーション ネットリストのプロパティを設定できます。このチュートリアル of ビヘイビア シミュレーションのプロパティを表示/変更するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、テストベンチ ファイル (stopwatch_tb) を選択します。
2. [Processes] ペインで [ISim Simulator] を展開し、[Simulate Behavioral Model] を右クリックして [Process Properties] をクリックします。
3. [Property display level] を [Advanced] に設定します。

[Advanced] に設定すると、すべてのプロパティが表示されます。

メモ：各プロパティの詳細な説明は、[Process Properties] ダイアログ ボックスで [Help] ボタンをクリックしてください。

4. [Simulation Run Time] を「2000 ns」に変更します。
5. [OK] をクリックします。

次の図に、ビヘイビア シミュレーションのプロパティを示します。

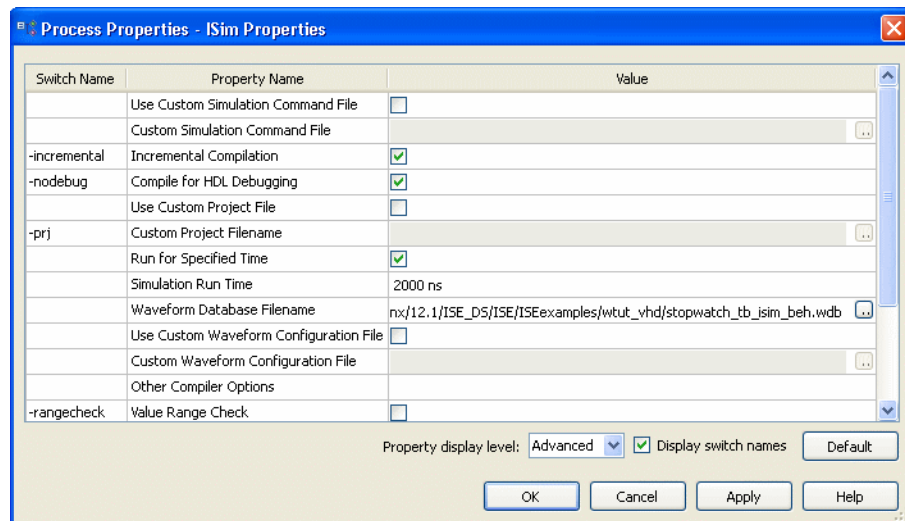


図 4-11 : ビヘイビア シミュレーションのプロセス プロパティ

シミュレーションの実行

プロセス プロパティを設定したら、ISim を実行してデザインをシミュレーションします。ビヘイビア シミュレーションを実行するには、[Simulate Behavioral Model] をダブルクリックします。ISim により、作業ディレクトリの作成、ソース ファイルのコンパイル、デザインの読み込みが実行され、指定した時間だけシミュレーションが実行されます。

このデザインの大部分は 100Hz で動作するので、シミュレーションに時間がかかります。RESET がディアサートされてから約 33ms 後に SF_D および LCD_E が遷移し始めるので、短時間のシミュレーションではカウンタが動作していないように見えます。このチュートリアルでは、DCM の信号のみを調べて、正常に動作しているかどうかを検証します。

信号の追加

シミュレーション中の信号を表示するには、波形ウィンドウに追加する必要があります。最上位のポートは、あらかじめ追加されています。その他の信号は、[Instances and Processes] パネルに表示されます。次に、デザイン階層の信号を追加する方法を示します。このチュートリアルでは、波形に DCM 信号を追加します。

デザイン階層の信号を追加するには、次の手順に従います。

1. [Instances and Processes] パネルで [stopwatch_tb] → [UUT] を展開します。

次の図に VHDL フローの [Instances and Processes] パネルを示します。回路図または Verilog フローの [Instances and Processes] パネルのレイアウトは、VHDL のものと異なる場合があります。

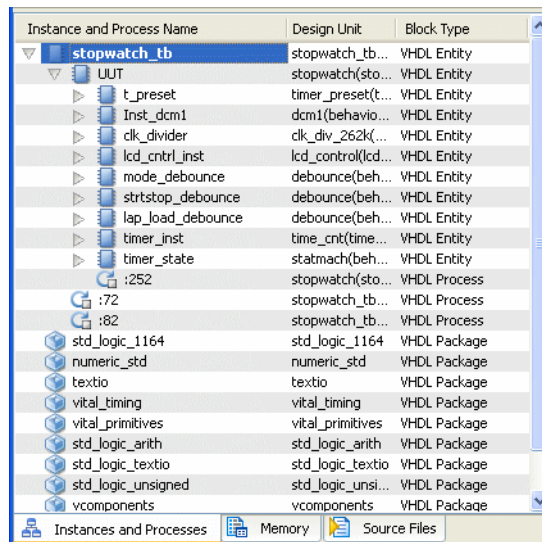


図 4-12 : [Instances and Processes] パネル (VHDL フロー)

2. [Instances and Processes] パネルで [Inst_dcm1] をクリックします。
3. clk_in_in を [Objects] パネルから波形ウィンドウにドラッグ アンド ドロップします。
4. 次の信号を選択します。

- ◆ RST_IN
- ◆ CLKFX_OUT
- ◆ CLK0_OUT
- ◆ LOCKED_OUT

メモ：複数の信号を選択するには、Ctrl キーを押しながら信号をクリックします。

5. 選択した信号を波形ウィンドウにドラッグします。

メモ：選択した信号を右クリックして [Add to Wave Window] をクリックしても、同じ操作を実行できます。

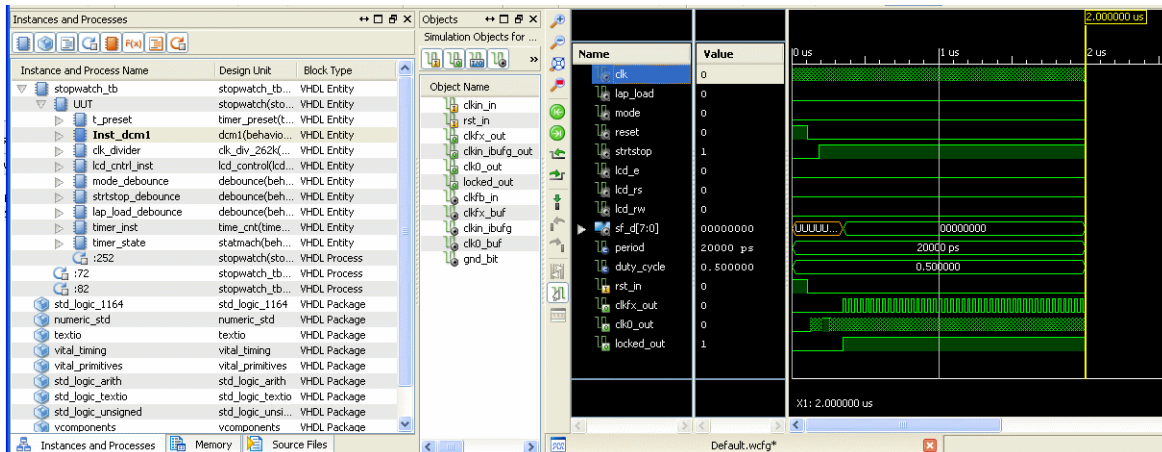


図 4-13 : シミュレーション波形に信号を追加

新しく追加した信号に対しては、波形は表示されていません。これは、ISim にこれらの信号のデータが記録されていないためです。デフォルトでは、シミュレーションを実行したときに波形ウィンドウに追加されていた信号のデータのみが記録されるので、波形ウィンドウに信号を追加した後、シミュレーションを再実行する必要があります。

シミュレーションの再実行

ISim でシミュレーションを再実行するには、次の手順に従います。

1. ツールバーの [Restart] ボタンをクリックします。



図 4-14 : ISim の [Restart] ボタン

2. ISim のコマンド プロンプトに「run 2000 ns」と入力し、Enter キーを押します。

シミュレーションが 2000ns 間実行され、DCM の波形が波形ウィンドウに表示されます。

信号の解析

DCM 信号が予測どおりに動作しているかを検証します。CLK0_OUT は 50MHz、CLKFX_OUT は約 26MHz で動作する必要があります。DCM 信号は、LOCKED 信号が High になるまでは無効なので、LOCKED 信号が High になった後のものを解析する必要があります。

ISim では、波形にマーカーを追加して、遷移間の時間を確認できます。CLK0_OUT の周期を確認するには、次の手順に従います。

1. 必要に応じて、ツールバーのズーム ボタンを使用して波形を拡大表示します。
2. 波形ウィンドウのツールバーで [Snap to Transition] ボタンをクリックします。



図 4-15 : ツールバーの [Snap to Transition] ボタン

3. LOCKED_OUT 信号が High になった後の、CLK0_OUT 信号の最初の立ち上がりエッジをクリックし、カーソルを CLK0_OUT 信号の次の立ち上がりエッジにドラッグします。
4. 波形ウィンドウの下部に開始点の時間、終了点の時間、および時間差が表示されます。この例では時間差は 20.0ns であり、周波数に変換すると 50MHz になります。この値はテストベンチからの入力周波数に等しく、DCM の CLK0 出力の周波数は正しいと判断できます。

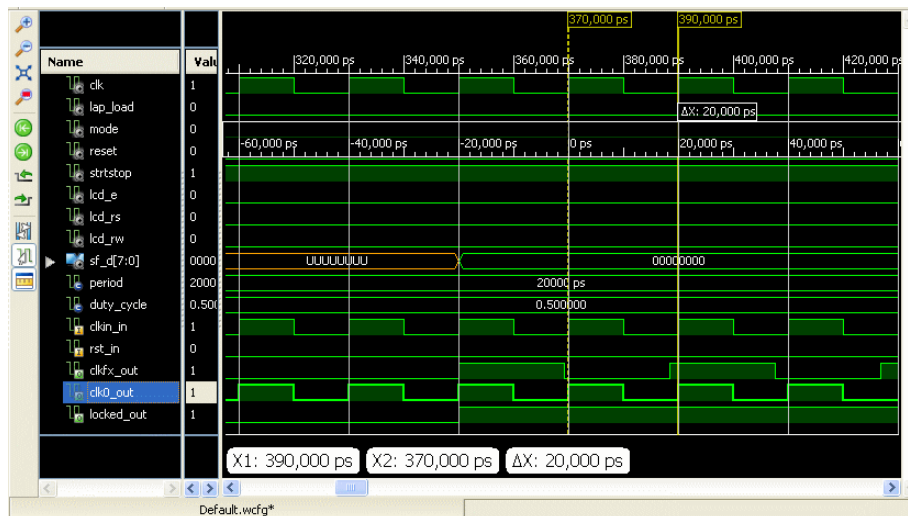


図 4-16：波形ウィンドウに表示される遷移間の時間

5. 同じ手順を使用して CLKFX_OUT を調べます。結果は 38.5 ns となり、周波数に変換すると約 26MHz になります。

これで、ビヘイビア シミュレーションは完了しました。この後、第 5 章「デザイン インプリメンテーション」の手順に従って、デザインをインプリメントします。

デザイン インプリメンテーション

この章は、次のセクションから構成されています。

- 「インプリメンテーションの概要」
- 「入門」
- 「プロパティの設定」
- 「タイミング制約の作成」
- 「デザインの変換」
- 「Constraints Editor の使用」
- 「PlanAhead ソフトウェアを使用した I/O ロケーションの割り当て」
- 「デザインのマップ」
- 「タイミング解析を使用したマップ後のブロック遅延の評価」
- 「デザインの配置配線」
- 「FPGA Editor を使用した配置配線の検証」
- 「配置配線後のタイミングの評価」
- 「コンフィギュレーション データの作成」
- 「コマンド ラインを使用したインプリメンテーション」

インプリメンテーションの概要

インプリメンテーションは、デザインを変換、マップ、配置配線し、ビットストリーム ファイルを生成するプロセスです。ザイリンクス ISE® ソフトウェアにはデザイン インプリメンテーション ツールが組み込まれており、Project Navigator から簡単に実行できます。

この章は、「インプリメンテーションのみのフロー」の最初の章であり、「HDL デザイン フロー」および「回路図デザイン フロー」の続きの章です。

この章では、ISE ソフトウェアのインプリメンテーション フローを示します。デザインは、既に EDA インターフェイス ツールでコンパイルされています。デザインのコンパイルについては、[第 2 章「HDL ベースのデザイン」](#)または[第 3 章「回路図ベースのデザイン」](#)を参照してください。この章では、フロントエンド ツールで生成された合成済みネットリスト (EDN、NGC) をインプリメンテーション ツールで処理し、ユーザー制約ファイル (UCF) を使用して配置制約を適用します。また、タイミング制約を追加し、さらに配置制約を追加します。

入門

チュートリアルへのデザインは、実時間とラップタイムを表示するストップウォッチの機能を表しています。このシステムには、CLK、RESET、LAP_LOAD、MODE、および SRTSTP の 5 つの入力があり、ラップタイムと通常のタイマーを LCD ディスプレイに表示します。

デザイン入力からの継続

HDL デザイン フローまたは回路図デザイン フローのチュートリアルに従いプロジェクトを作成した場合は、ソース ファイルは完成しており、合成済みデザインも生成されています。

プロジェクトに stopwatch.ucf 制約ファイルが含まれていない場合は、次の手順に従って作成します。

1. [Design] パネルの [Hierarchy] ペインで、最上位ソース ファイル stopwatch を選択します。
2. [Project] → [New Source] をクリックします。
3. [Implementation Constraints File] を選択します。
4. [File name] に「stopwatch.ucf」と入力します。
5. [Next] をクリックします。
6. [Finish] をクリックします。

プロジェクトに UCF を追加したら、この章のチュートリアルを開始できます。[「プロパティの設定」](#)に進んでください。

デザイン インプリメンテーションからの開始

チュートリアルをこの章から開始する場合は、ザイリンクスの Web サイトから合成済みのデザインをダウンロードし、ISE ソフトウェアでプロジェクトを作成して、ダウンロードしたファイルを追加します。

1. Watch という名前の作業ディレクトリを作成します。
2. 次の表に示される合成済みのチュートリアル ファイルをダウンロードして、作成した作業ディレクトリに追加します。ファイルは、ザイリンクスの Web サイトの ISE Design Suite 12 [チュートリアル ページ](#)からダウンロードできます。

表 5-1：必要なチュートリアル ファイル

ファイル名	説明
stopwatch.edn、stopwatch.edf、または stopwatch.ngc	入力ネットリスト ファイル (EDIF)
timer_preset.ngc	タイマー ネットリスト ファイル (NGC)
stopwatch.ucf	ユーザー制約ファイル

3. 次のいずれかの方法で、ISE ソフトウェアを開きます。
 - a. ワークステーション：「ise」と入力します。
 - b. PC：[スタート] → [すべてのプログラム] → [Xilinx ISE Design Suite 12.1] → [ISE デザイン ツール] → [Project Navigator] をクリックします。

4. 次の手順に従って新規プロジェクトを作成し、EDIF ネットリストを追加します。
 - a. [File] → [New Project] をクリックします。
 - b. [Name] に「wtut_edif」と入力します。
 - c. [Top-level source type] を [EDIF] に設定し、[Next] をクリックします。
 - d. [Input design] で stopwatch.edf または stopwatch.edn を選択します。
 - e. [Constraints file] で stopwatch.ucf を選択し、[Next] をクリックします。
 - f. 次のように設定します。
 - [Family] : [Spartan3A and Spartan3AN]
 - [Device] : [XC3S700A]
 - [Package] : [FG484]
 - [Speed] : [-4]
 - g. [Next] をクリックし、次のページで [Finish] をクリックして、プロジェクト作成を完了します。
- メモ : timer_preset.ngc ファイルがプロジェクト ディレクトリにない場合、解凍した ZIP ファイルからコピーします。

プロパティの設定

このセクションでは、デザイン インプリメンテーションのプロセス プロパティを設定する方法を示します。インプリメンテーション プロパティは、デザインのマップ、配置、配線、および最適化を制御します。

このチュートリアル のインプリメンテーション プロパティを設定するには、次の手順に従います。

1. [Design] パネルの [View] ペインで [Implementation] をオンにします。
2. [Hierarchy] ペインで最上位ソース ファイル stopwatch を選択します。
3. [Processes] ペインで [Implement Design] を右クリックし、[Process Properties] をクリックします。

[Process Properties] ダイアログ ボックスが開き、変換、マップ、配置配線、タイミング レポートのプロパティを設定できるようになります。ダイアログ ボックスの左ペインでカテゴリをクリックし、各デザイン インプリメンテーション フェーズのプロパティを設定します。
4. [Property display level] が [Advanced] に設定されていることを確認します。

[Advanced] に設定すると、すべてのプロパティが表示されます。
5. [Category] で [Place & Route Properties] をクリックします。
6. [Place & Route Effort Level (Overall)] を [High] に変更します。

このオプションは、配置配線の全般的なエフォート レベルを指定します。

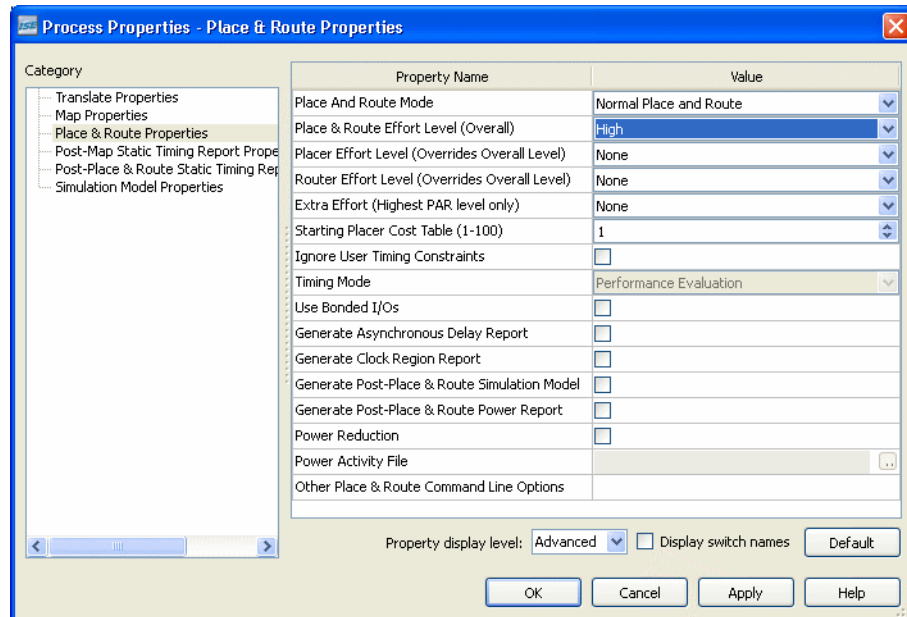


図 5-1 : [Process Properties] ダイアログ ボックスの [Place & Route Properties] ページ

7. [OK] をクリックして [Process Properties] ダイアログ ボックスを閉じます。

タイミング制約の作成

ユーザー制約ファイル (UCF) はテキスト ファイルであり、テキスト エディタで直接編集できます。このファイルを編集しやすくするため、制約を作成および編集するグラフィカル ツールが提供されています。Constraints Editor および PlanAhead™ ソフトウェアは、タイミング、I/O、および配置制約を入力するためのグラフィカル ツールです。

Constraints Editor を起動するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、stopwatch モジュールを選択します。
2. [Processes] ペインで [User Constraints] を展開し、[Create Timing Constraints] をダブルクリックします。

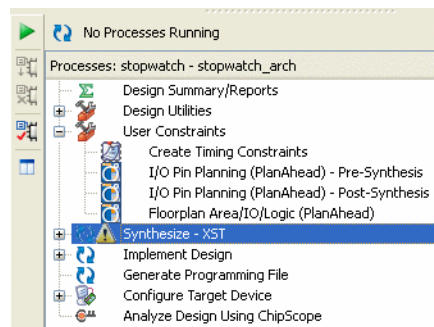


図 5-2 : [Create Timing Constraints] プロセス

変換プロセスが実行されます。変換プロセスについては、次のセクションで説明します。Constraints Editor が起動します。

デザインの変換

インプリメンテーションで作成されたファイルは、ISE ソフトウェアにより管理されます。ISE デザインのプロセスは、[Process Properties] ダイアログ ボックスで指定された設定を使用して実行されるので、デザインの処理方法を完全に制御することが可能です。通常は、まずプロパティを設定し、[Implement Design] プロセスを実行してインプリメンテーション全体を 1 つのステップで実行します。[Implement Design] プロセスには、[Translate]、[Map]、[Place & Route] の 3 つのサブプロセスがあります。[Implement Design] プロセスを実行して 3 つのサブプロセスをすべて実行するか、サブプロセスを個別に実行できます。このチュートリアルでは、各手順を理解しやすくするため、サブプロセスを個別に実行します。

変換プロセスでは、NGDBuild により次の処理が実行されます。

- 入力デザイン ネットリストが変換され、統合された 1 つの NGD ネットリストが生成されます。この NGD ネットリストでは、デザインの論理とロケーション制約およびタイミング制約が記述されます。
- タイミング仕様のチェックと論理デザイン ルール チェックが実行されます。
- 統合されたネットリストにユーザー制約ファイル (UCF) の制約情報が追加されます。

Constraints Editor の使用

[Create Timing Constraints] プロセスを実行すると、変換が自動的に実行され、Constraints Editor が起動します。

Constraints Editor では、次の操作を実行できます。

- UCF ファイルで定義されている制約の編集
- デザインへの制約の追加

Constraints Editor への入力ファイルは、次のとおりです。

- NGD (Native Generic Database) ファイル
マップ ツールへの入力ファイルで、このファイルから物理的デザイン データベースである NCD (Native Circuit Description) ファイルが生成されます。
- ユーザー制約ファイル (UCF)
ISE プロジェクトに含まれるすべての UCF ファイルが Constraints Editor に渡されます。

ISE プロジェクトでは複数の UCF ファイルがサポートされています。プロジェクトに含まれるすべての制約ファイルが Constraints Editor に読み込まれ、制約を編集するとその制約が記述されていた制約ファイルで更新されます。新しい制約は、Constraints Editor で指定した UCF ファイルに記述されます。

変換プロセス (NGDBuild) では、ソース ネットリストと共に UCF ファイルが使用され、NGD ファイルがアップデートされます。その次の段階であるマップ プロセスでは、この NGD ファイルが読み込まれます。このチュートリアルの場合、stopwatch.ngd ファイルと stopwatch.ucf ファイルが自動的に Constraints Editor に読み込まれます。

このセクションの手順に従うと、PERIOD、グローバル OFFSET IN、グローバル OFFSET OUT、および TIMEGRP OFFSET IN 制約が UCF に記述され、インプリメンテーションで使用されます。[Timing Constraints] パネルで [Timing Constraints] → [Clock Domains] を選択すると、デザインのクロック ネットがすべて表示され、PERIOD、pad-to-setup、および clock-to-pad の値を設定できます。内部名の多くは、デザイン フローおよび合成ツールによって異なることに注意してください。

次の図に Constraints Editor を示します。

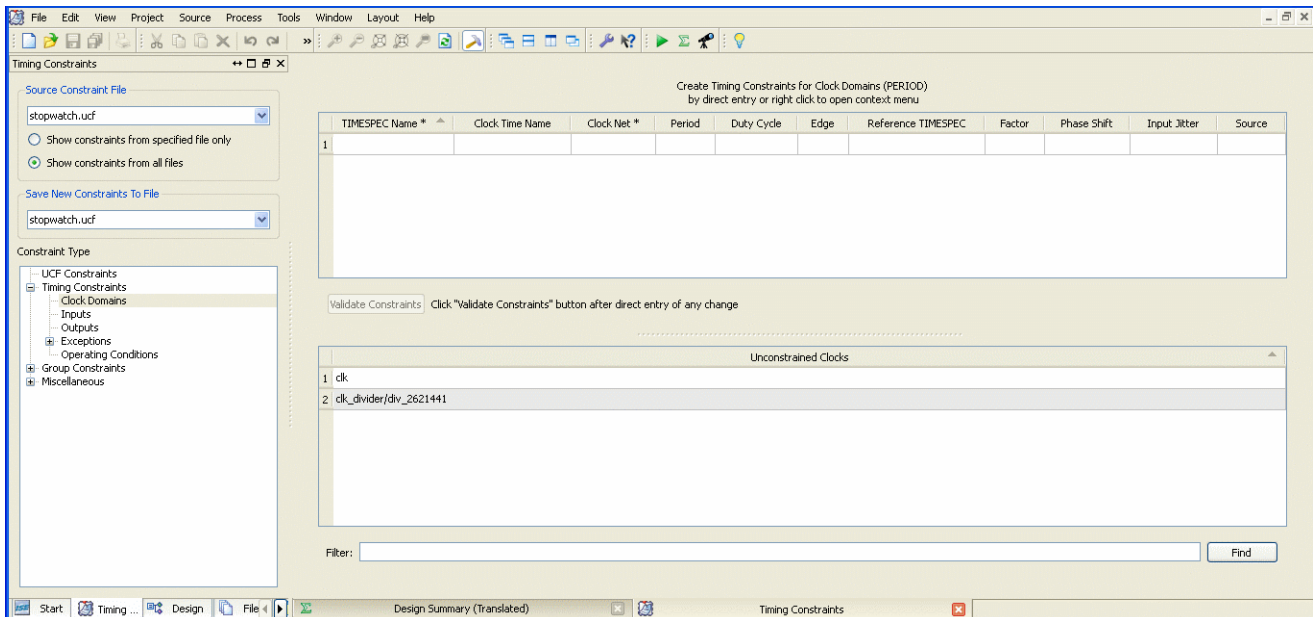


図 5-3 : Project Navigator の Constraints Editor ([Clock Domains] を選択)

Constraints Editor で、次の手順に従って制約を編集します。

1. [Unconstrained Clocks] 表で clk 信号の行をダブルクリックします。[Clock Period] ダイアログボックスが開きます。
2. [Clock signal definition] で、[Specify time] がオンになっていることを確認します。
このオプションをオンにすると、クロック周期の値を入力できます。
3. [Time] に「7.0」と入力します。
4. [Units] を [ns] に設定します。

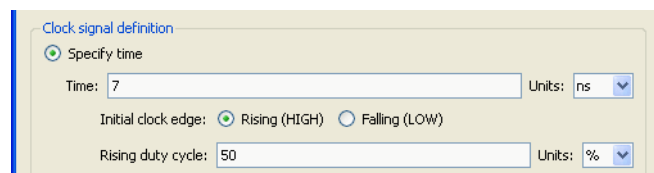


図 5-4 : PERIOD 制約の設定

5. [Input jitter] に「60」と入力します。
6. [Units] を [ps] に設定します。

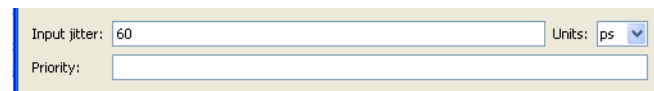


図 5-5 : INPUT JITTER 制約の設定

7. [OK] をクリックします。
ウィンドウ上部の制約の表に **PERIOD** 制約が表示されます。ダイアログ ボックスで定義した設定が示されています (デューティ サイクルはデフォルトの 50%)。
8. [Timing Constraints] パネルで [Timing Constraints] → [Inputs] をクリックします。
9. 右下のグローバル **OFFSET IN** 制約の表で clk 信号をダブルクリックし、[Create Setup Time (OFFSET IN)] ダイアログ ボックスを開きます。
10. 最初のページでデフォルト値をそのまま使用し、[Next] をクリックします。

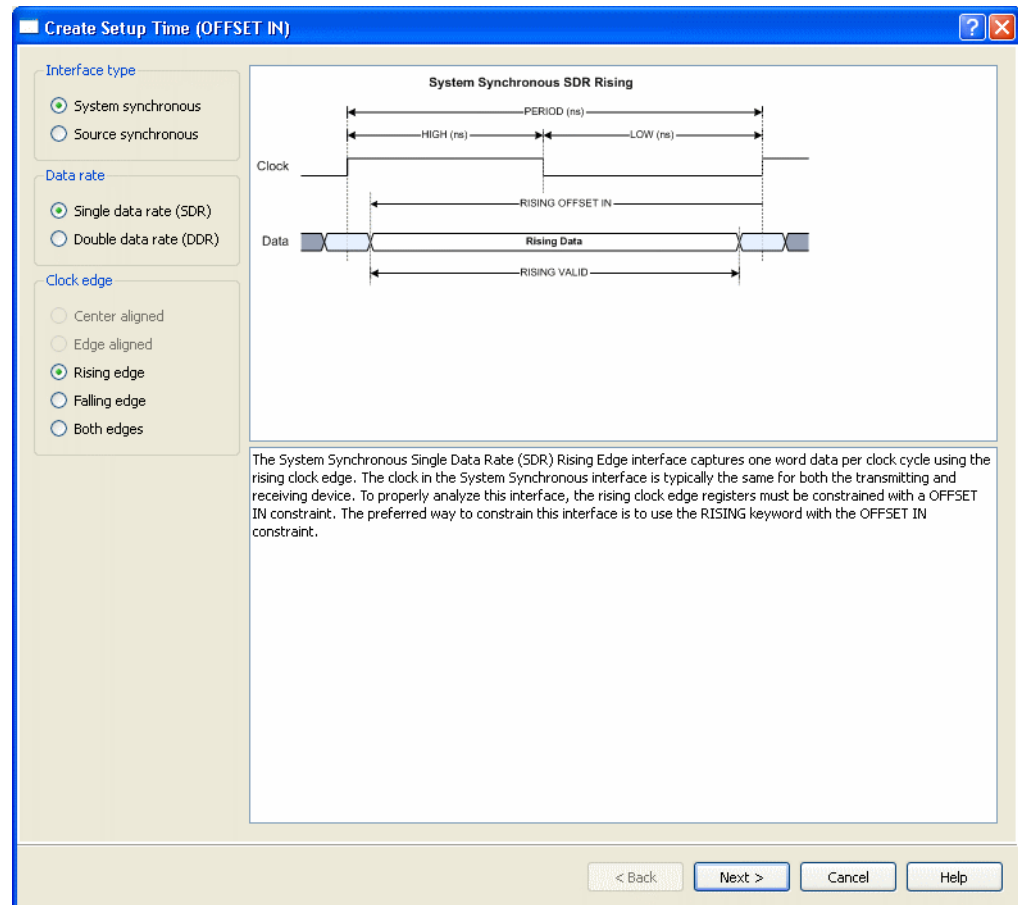


図 5-6 : [Create Setup Time (OFFSET IN)] ダイアログ ボックスの 1 ページ目

11. [External setup time (offset in)] に「6」と入力し、[Unit] を [ns] に設定します。
12. [Data valid duration] に「6」と入力し、[Unit] を [ns] に設定します。
CLK 信号のグローバル **OFFSET IN** 制約が作成されます。

13. [Finish] をクリックします。

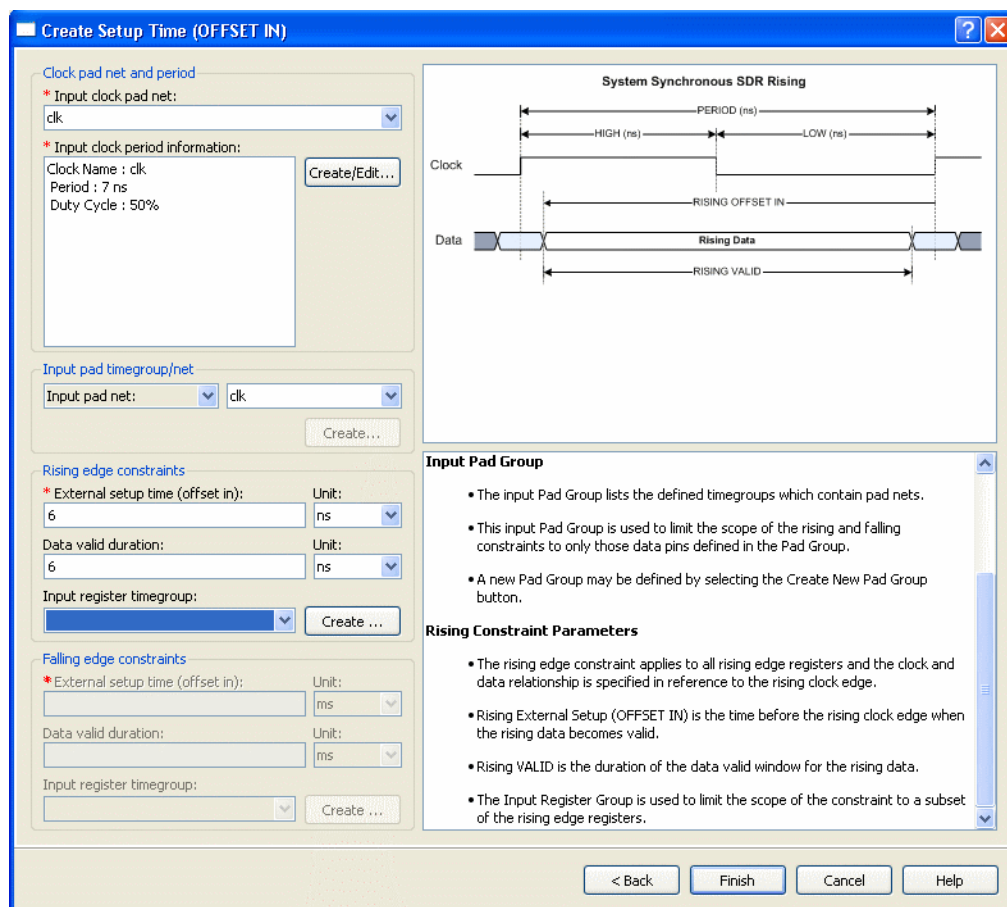


図 5-7 : [Create Setup Time (OFFSET IN)] ダイアログ ボックスの 2 ページ目

14. [Timing Constraints] パネルで [Timing Constraints] → [Outputs] をクリックします。

15. 右下のグローバル OFFSET OUT 制約の表で clk 信号をダブルクリックします。

16. [External clock to pad (offset out)] に「38」と入力し、[Units] を [ns] に設定します。

CLK 信号のグローバル OFFSET OUT 制約が作成されます。

17. [OK] をクリックします。

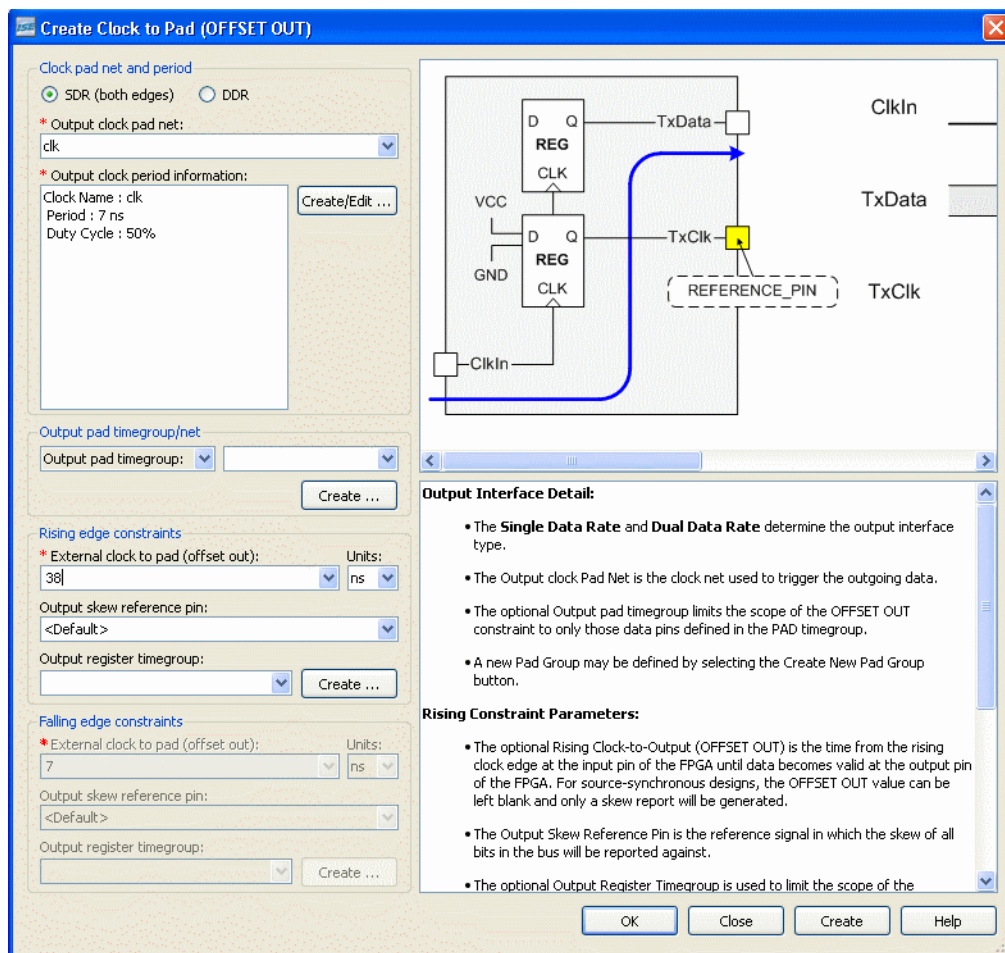


図 5-8 : [Create Clock to Pad (OFFSET OUT)] ダイアログ ボックス

18. [Unconstrained Output Ports] の表で、Shift キーを押しながら sf_d[0] から sf_d[7] までを選択します。

19. 右クリックし、[Create Time Group] をクリックします。

20. [Create Time Group] ダイアログ ボックスで、[Time group name] に「display_grp」と入力し、[OK] をクリックします。

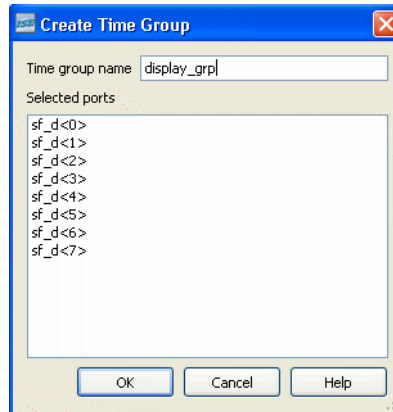


図 5-9：タイム グループの作成

21. オフセット制約を作成するかどうかを尋ねるダイアログ ボックスが表示されたら、[OK] をクリックします。
22. [External clock to pad (offset out)] に「32」と入力し、[Units] を [ns] に設定します。

23. [OK] をクリックします。

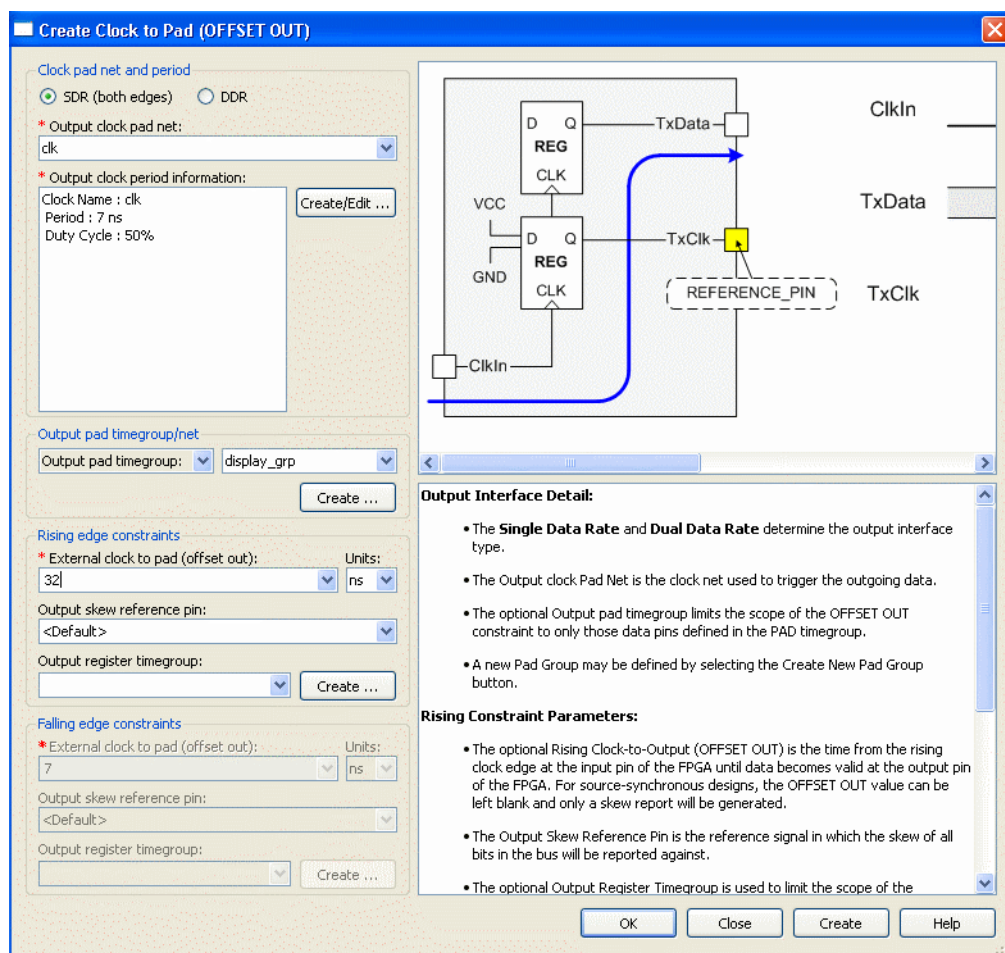


図 5-10 : [Create Clock to Pad (OFFSET OUT)] ダイアログ ボックス

24. Constraints Editor で [File] → [Save] をクリックします。

加えた変更が stopwatch.ucf ファイルに保存されます。

25. [File] → [Close] をクリックして Constraints Editor を終了します。

PlanAhead ソフトウェアを使用した I/O ロケーションの割り当て

NGD ファイルで定義されているピン ロケーションおよびエリア グループ制約を追加または編集するには、PlanAhead ソフトウェアを使用します。PlanAhead ソフトウェアでは、制約はプロジェクトの UCF ファイルに記述されます。プロジェクトに複数の UCF ファイルがある場合は、新しい制約を記述する制約ファイルを指定する必要があります。既存の制約を変更すると、その制約が記述されていた制約ファイルで更新されます。PlanAhead ソフトウェアには、ピン配置を支援するため、デバイス特定のデザイン ルール チェックも含まれています。

変換プロセスでは、ソース ネットリストと共にデザイン UCF ファイルが使用され、NGD ファイルがアップデートされます。NGD には、デザインおよび UCF ファイルに加えた変更が反映されます。

このセクションでは、いくつかの信号に対してピンを割り当てます。

1. [Design] パネルの [Hierarchy] ペインで、stopwatch モジュールを選択します。
2. [Processes] ペインで [User Constraints] を展開し、[I/O Pin Planning (PlanAhead) - Post-Synthesis] をダブルクリックします。

I/O ピン配置は、合成前または合成後のどちらでも実行できますが、できる限り合成後に実行することをお勧めします。合成後のデザインには、PlanAhead ソフトウェアで実行される I/O およびクロックに関連したデザイン ルール チェックに必要な情報が含まれています。

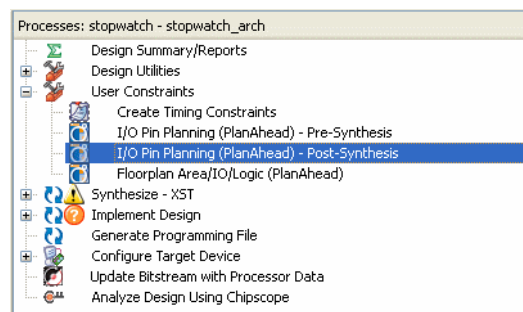


図 5-11 : [I/O Pin Planning (PlanAhead) - Post-Synthesis] プロセス

このプロセスにより、PlanAhead ソフトウェアが起動します。デザインが合成されていない場合、Project Navigator で合成が自動的に実行されます。

表示される [Welcome] ダイアログ ボックスに、PlanAhead ソフトウェアのマニュアル、チュートリアル、およびその他のトレーニング資料へのリンクがあります。このチュートリアルでは、PlanAhead ソフトウェアの使用方法与機能の概要を説明します。詳細情報、すべての機能を理解するには、その他のリソースを参照してください。

3. [Welcome] ダイアログ ボックスで [Close] をクリックします。

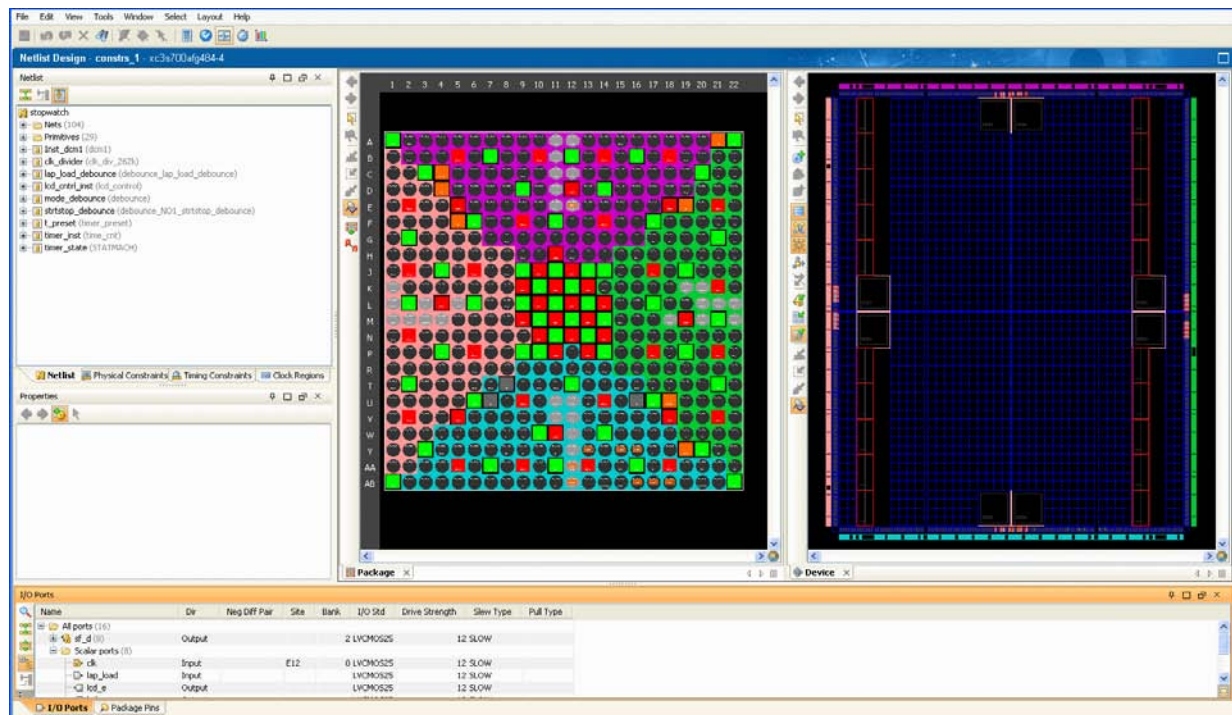


図 5-12 : PlanAhead ソフトウェア

4. [I/O Ports] ビューで [All ports] → [Scalar ports] を展開します。lcd_e、lcd_rs、および lcd_rw I/O 信号のピン割り当てを作成します。
5. lcd_e 出力信号を選択し、[Package] ビューの AB4 ピン ロケーションにドラッグ アンド ドロップします。



図 5-13 : [Package] ビューへのドラッグによる I/O ピンの割り当て

6. 手順 5 を繰り返して、次の出力ピンを配置します。
 - ◆ LCD_RS : Y14
 - ◆ LCD_RW : W13

または、I/O 信号を選択して [I/O Port Properties] ビューの [Site] にロケーションを入力して割り当てることもできます。

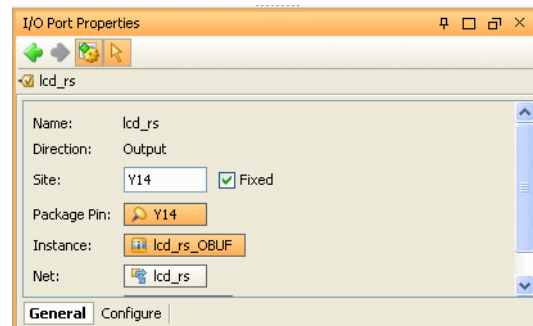


図 5-14 : [I/O Port Properties] を使用する I/O ピンの割り当て

7. ドラッグ アンド ドロップまたは [I/O Port Properties] ビューを使用した方法のいずれかを使用して、次の入力信号を適切な I/O ピン ロケーションに配置します。
 - ◆ LAP_LOAD : T16
 - ◆ RESET : U15
 - ◆ MODE : T14
 - ◆ STRTSTOP : T15
8. ピン ロケーションを設定したら、[File] → [Save Project] をクリックします。変更が stopwatch.ucf ファイルに保存されます。
9. [File] → [Exit] をクリックして PlanAhead ソフトウェアを終了します。

デザインのマップ

インプリメンテーション プロパティと制約が定義されたので、次のようにデザインのインプリメンテーションを継続します。

1. [Design] パネルの [Hierarchy] ペインで、stopwatch モジュールを選択します。
2. [Processes] ペインで [Implement Design] を展開し、[Map] をダブルクリックします。

変換プロセスが最新でない場合、変換プロセスも自動的に実行されます。

デザインは、CLB および IOB にマップされます。マップ プロセスでは、次の処理が実行されます。

- デザインに含まれるすべての基本ロジックに CLB および IOB リソースが割り当てられます。
- ロケーション制約およびタイミング制約の処理、ターゲット デバイスに対する最適化の実行、マップされたネットリストに対するデザイン ルール チェックが実行されます。

インプリメンテーションの各プロセスに対し、次のようにレポートが生成されます。

表 5-2 : マップ プロセスまでに生成されるインプリメンテーションのレポート

レポート	説明
変換レポート	変換プロセスの警告メッセージおよびエラーメッセージが記述されます。
マップ レポート	ターゲット デバイスのリソースがどのように割り当てられたか、最適化により削除されたロジックに関する情報、デバイスの使用率が示されます。
NGDBuild および MAP のすべてのレポート	マップ レポートの詳細は、『コマンド ライン ツール ユーザー ガイド』を参照してください。このガイドは ISE ソフトウェア マニュアル コレクションに含まれています。ソフトウェア マニュアル コレクションを開くには、[Help] → [Software Manuals] をクリックするか、ザイリンクス Web サイトの ソフトウェア マニュアル コレクション にアクセスします。

レポートを表示するには、次の手順に従います。

1. [Design] パネルの [Processes] ペインで [Design Summary/Reports] をダブルクリックします。

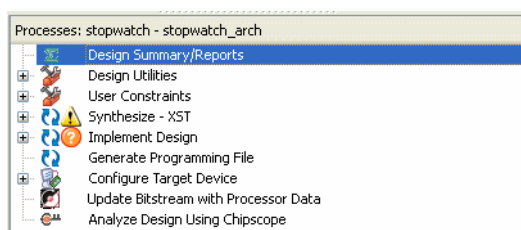


図 5-15 : [Design Summary/Reports] プロセス

次の図に、デザイン サマリ/レポート ビューアを示します。

stopwatch Project Status (03/30/2010 - 10:46:42)

Project File:	wtut_vhd.xise	Parser Errors:	No Errors
Module Name:	stopwatch	Implementation State:	Mapped
Target Device:	xc3s700a-4fg484	Errors:	No Errors
Product Version:	ISE 12.1	Warnings:	6 Warnings (6 new)
Design Goal:	Balanced	Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	Timing Constraints:	
Environment:	System Settings	Final Timing Score:	

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	229	11,776	1%	
Number of 4 input LUTs	371	11,776	3%	
Number of occupied Slices	285	5,888	4%	
Number of Slices containing only related logic	285	285	100%	
Number of Slices containing unrelated logic	0	285	0%	
Total Number of 4 input LUTs	443	11,776	3%	
Number used as logic	371			
Number used as a route-thru	72			
Number of bonded IOBs	16	372	4%	
Number of BUFMGUXs	3	24	12%	
Number of DCMs	1	8	12%	
Average Fanout of Non-Clock Nets	3.43			

Detailed Reports

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon Mar 29 22:46:06 2010	0	5 Warnings (5 new)	4 Infos (4 new)
Translation Report	Current	Tue Mar 30 10:46:29 2010	0	1 Warning (1 new)	1 Info (1 new)
Map Report	Current	Tue Mar 30 10:46:41 2010	0	0	5 Infos (5 new)
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					
Bitgen Report					

図 5-16：デザイン サマリ/レポート ビューア

2. デザイン サマリの左上の左ペインで、[Detailed Reports] セクションの [Translation Report] または [Map Report] を選択します。
3. レポートを確認します。

デザイン サマリには、デザイン結果の概要およびインプリメンテーション実行で生成されたすべてのメッセージ (エラー、警告、情報) も表示されます。

タイミング解析を使用したマップ後のブロック遅延の評価

デザインをマップしたら、ロジック レベルのタイミング レポートを使用してデザインの論理パスを評価します。マップ済みデザインの評価では、タイミング仕様に対してブロック遅延が適当であるかどうか判断されます。デザインは配置配線されていないので、実際の配線遅延はわかっていません。この段階のタイミング レポートでは、論理ブロック遅延と予測される配線遅延が示されます。ネット遅延は、ブロック間の最適な距離 (「未配置のフロア」とも呼ばれる) に基づいています。

タイミング要件の評価

タイミング要件が現実的であるかを評価するには、マップ後のデザインを確認します。ブロック遅延はパスの総遅延の約 50% を占めるという、おおよそのガイドラインがあります (「50/50 ルール」とも呼ばれる)。たとえば、パスのブロック遅延が 10ns である場合、配置配線後にタイミング制約 20ns を満たすと考えられます。

デザインの集積度が高い場合、マップ後のスタティック タイミング レポートで、ブロック遅延と予測される配線遅延に基づいて、タイミング制約を解析できます。この解析により、タイミング制約が満たされるかどうかを予測できます。このレポートは、マップの後、配置配線の前に生成します。

マップ後のスタティック タイミング レポートの表示

マップ後のスタティック タイミング レポートを使用すると、配置配線を実行する前に、タイミング違反が発生する可能性があるかを調べることができます。このチュートリアル の **stopwatch** デザインではタイミング制約を定義しているので、タイミング レポートには各タイミング制約が適用されたパスが表示されます。

マップ後のスタティック タイミング レポートを表示して入力した **PERIOD** 制約を評価するには、次の手順に従います。

1. [Processes] ペインで [Map] を展開し、[Generate Post-Map Static Timing] をダブルクリックします。
2. [Analyze Post-Map Static Timing] をダブルクリックして、マップ後のスタティック タイミング レポートを開きます。

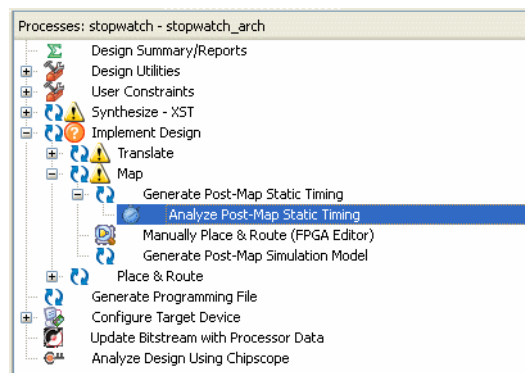


図 5-17 : [Analyze Post-Map Static Timing] プロセス

Timing Analyzer が起動し、レポートが表示されます。

3. [Report Navigation] で [TS_inst_dcm1_CLKFX_BUF] タイミング制約をクリックします。

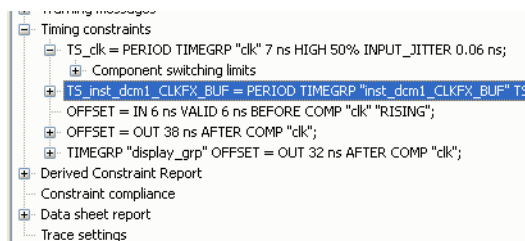


図 5-18 : マップ後のスタティック タイミング制約の選択

選択した制約のレポートがワークスペースに表示されます。このレポートには、選択した **PERIOD** 制約とマップ後の最小周期が表示されます。デフォルトでは、1 つのタイミング制約に対して 3 つのパスのみが表示されます。この 3 つのパスのいずれかをクリックすると、そのパスに関する詳細情報 (コンポーネント遅延と配線遅延) を確認できます。

各パスの最後に、ロジック遅延と配線遅延の割合がパーセントで示されます（「88.0% logic, 12.0% route」など）。横に「e」と記されているネット遅延は予測値であり、ブロックの最適な配置に基づいています。

4. レポートを確認したら、[File] → [Close] をクリックして **Timing Analyzer** を閉じます。

メモ： タイミング レポートを生成しなくても、配置配線プロセス (PAR) ではデザインのブロック遅延、フロア、およびタイミング仕様の関係に基づいてデザインが処理されます。たとえば、あるパスに 8ns の PERIOD 制約が設定されており、そのパスのブロック遅延が 7ns で未配置フロアのネット遅延が 3ns である場合、PAR は停止しエラー メッセージが表示されます。これは、総遅延 (10ns) がタイミング制約 (8ns) より大きいからです。マップ後のスタティック タイミング レポートには、PAR で発生する可能性があるタイミング違反が表示されます。

デザインの配置配線

マップ済みのデザインを評価したら、デザインを配置配線します。配置配線 (PAR) プロセスでは、次のいずれかの配置配線アルゴリズムが実行されます。

- タイミング ドリブン PAR

入力ネットリスト、制約ファイル、またはその両方で指定されたタイミング制約を使用して、配置配線が実行されます。

- 非タイミング ドリブン PAR

タイミング制約を無視して、配置配線が実行されます。

このチュートリアルではタイミング制約を定義しているので、タイミング ドリブンの配置配線が実行されます。

配線配置を実行するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、stopwatch モジュールを選択します。
2. [Processes] ペインで [Implement Design] を展開し [Place & Route] をダブルクリックします。

[Place & Route] プロセスでは、次の表に示すレポートが生成されます。

表 5-3：配置配線で生成されるレポート

レポート	説明
配置配線レポート	デバイスの使用率と遅延のサマリを示します。このレポートで、デザインの配置配線が完了しているか、タイミング制約が満たされているかを確認します。
非同期遅延レポート	デザインのすべてのネットおよびネット上のロードの遅延を示します。
PAR のすべてのレポート	PAR レポートの詳細は、『コマンド ライン ツール ユーザー ガイド』を参照してください。このガイドは ISE ソフトウェア マニュアル コレクションに含まれています。ソフトウェア マニュアル コレクションを開くには、[Help] → [Software Manuals] をクリックするか、ザイリンクス Web サイトの ソフトウェア マニュアル コレクション にアクセスします。

メモ：上記以外にオプションで生成可能なレポートもあります。これらのレポートの生成は、配置配線プロセス プロパティで有効にします。これらレポートを生成すると、デザイン サマリの [Secondary Reports] セクションに表示されます。

配置配線で生成されたレポートを表示するには、次の手順に従います。

1. [Design] パネルの [Processes] ペインで、[Design Summary/Reports] をダブルクリックします。
2. デザイン サマリの左上のペインで [Detailed Reports] → [Place and Route Report] をクリックします。

次の図に、デザイン サマリに表示された配置配線レポートを示します。

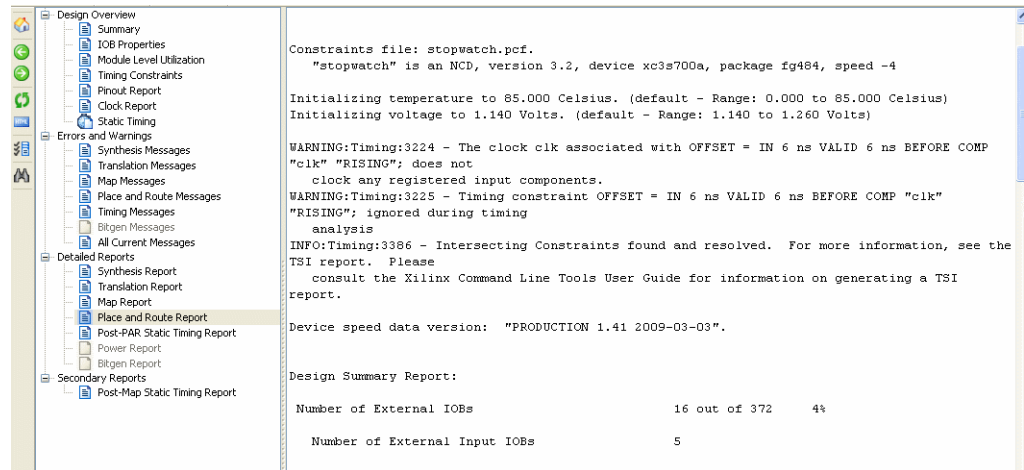


図 5-19：配置配線レポート

FPGA Editor を使用した配置配線の検証

FPGA Editor を使用して、FPGA を表示、コンフィギュレーションします。

FPGA Editor は、NCD (Native Circuit Description) ファイル、マクロ ファイル (NMC)、および物理制約ファイル (PCF) を読み込み、記述します。

FPGA Editor では、次の操作を実行できます。

- 自動配置配線を実行する前に、重要なコンポーネントを配置配線します。
- 配線プログラムでデザインの配線が完了しなかった場合に、配置配線を完了させます。
- デザインにプローブを追加して、ターゲット デバイスの信号のステートを調べます。プローブは、デバイスのデバッグで内部ネットの値を IOB (入力/出力ブロック) に割り当てるために使用します。
- BitGen を実行し、生成されたビットストリームをターゲット デバイスにダウンロードします。
- ILA (Integrated Logic Analyzer) コアのキャプチャ ユニットに接続されたネットを表示/変更します。

FPGA のデザインのレイアウトを表示するには、次の手順に従います。

1. [Design] パネルの [Processes] ペインで、[Place & Route] を展開して [View/Edit Routed Design (FPGA Editor)] をダブルクリックします。

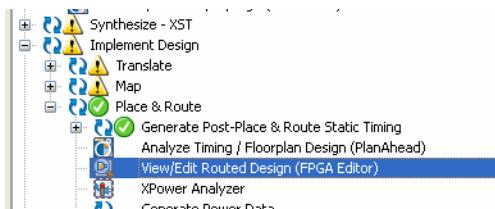


図 5-20 : [View/Edit Routed Design (FPGA Editor)] プロセス

2. FPGA Editor で、List ウィンドウの上部にあるドロップダウン リストから [All Nets] を選択します。

デザインのすべてのネットが、下のリストに表示されます。

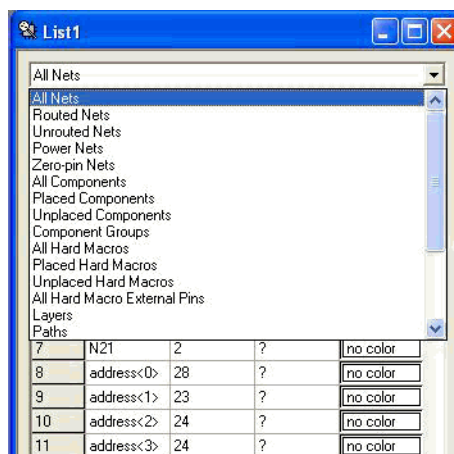


図 5-21 : FPGA Editor の List ウィンドウ

3. [clk_26214k] (クロック) ネットを選択し、クロック ネットのファンアウトを確認します。

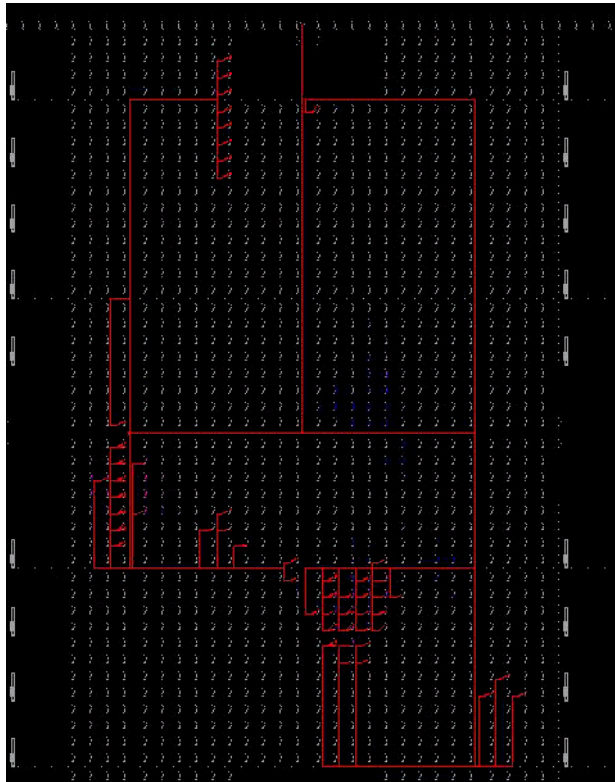


図 5-22 : クロック ネット

4. [File] → [Exit] をクリックして FPGA Editor を終了します。

配置配線後のタイミングの評価

デザインの配置配線後、指定のタイミング要件に対するデザインのパフォーマンスを検証するため、配置配線後のタイミング結果を解析できます。

タイミングを解析するには、次の方法があります。

- 配置配線後のスタティック タイミング レポートを表示する。
- PlanAhead ソフトウェアを使用して配置配線後のタイミング解析を実行する。
- デザイン サマリのハイパーリンクを使用して、個々のタイミング制約を解析する。

配置配線後のスタティック タイミング レポートの表示

配置配線後のスタティック タイミング レポートには、論理ブロック遅延と配線遅延が示されます。ネット遅延は、実際の配線遅延として示されます。レポートを表示するには、次の手順に従います。

1. デザイン サマリの左上のペインで、[Design Overview] → [Static Timing] をクリックします。
メモ : [Processes] ペインで [Implement Design] → [Place & Route] → [Generate Post-Place & Route Static Timing] を展開して [Analyze Post-Place & Route Static Timing] プロセスをダブルクリックしても、同じ操作を実行できます。
2. タイミング レポートが Timing Analyzer で開きます。

配置配線後のスタティック タイミング レポートから、次のことがわかります。

- ◆ 最小周期の値が、実際の配置遅延により増加しています。
マップ後のタイミング レポートでは、ロジック遅延が最小周期の 80% ~ 90% を占めていることが示されていましたが、配置配線後のレポートでは、ロジック遅延値が 30% ~ 40% を占めると示されています。未配置フロアの予測も変化しています。
- ◆ 配置配線のタイミングは、必ずしも 50/50 ルールに従いません。ワースト ケースのパスは、ほとんどがコンポーネント遅延です。
- ◆ 満たすことが困難なタイミング制約では、ワースト ケースのパスは大部分がロジック遅延です。パス遅延の合計に対して配線遅延の合計の占める割合は小さく、2 ~ 3 個のネットに分散されているので、これらのパスのタイミングをこれ以上減らすことは難しくなります。通常、ロジックのレベル数を減らすと、ブロック遅延が減少し、デザイン パフォーマンスが向上します。

PlanAhead ソフトウェアを使用したデザインの解析

PlanAhead ソフトウェアを使用して、配置配線後のデザインを解析できます。デザイン結果の解析およびデザイン クロージャのために、グラフィカル レイアウト解析、タイミング パス表示、およびフロアプランを実行できます。

1. [Design] パネルの [Hierarchy] ペインで、stopwatch モジュールを選択します。
2. [Processes] ペインで [Implement Design] → [Place & Route] を展開し、[Analyze Timing/Floorplan Design (PlanAhead)] をダブルクリックします。

次の図に、[Analyze Timing/Floorplan Design (PlanAhead)] を示します。

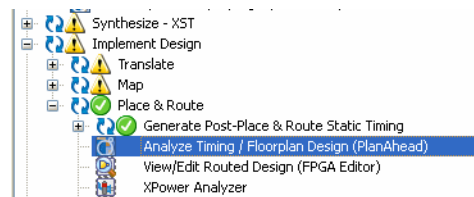


図 5-23 : [Analyze Timing/Floorplan Design (PlanAhead)] プロセス

3. PlanAhead ソフトウェアが開いたら、[Timing Results] ビューでタイミング パスの 1 つをクリックします。パスが [Device] ビューにグラフィカルに表示され、[Path Properties] タブにパスとその遅延の詳細が表示されます。

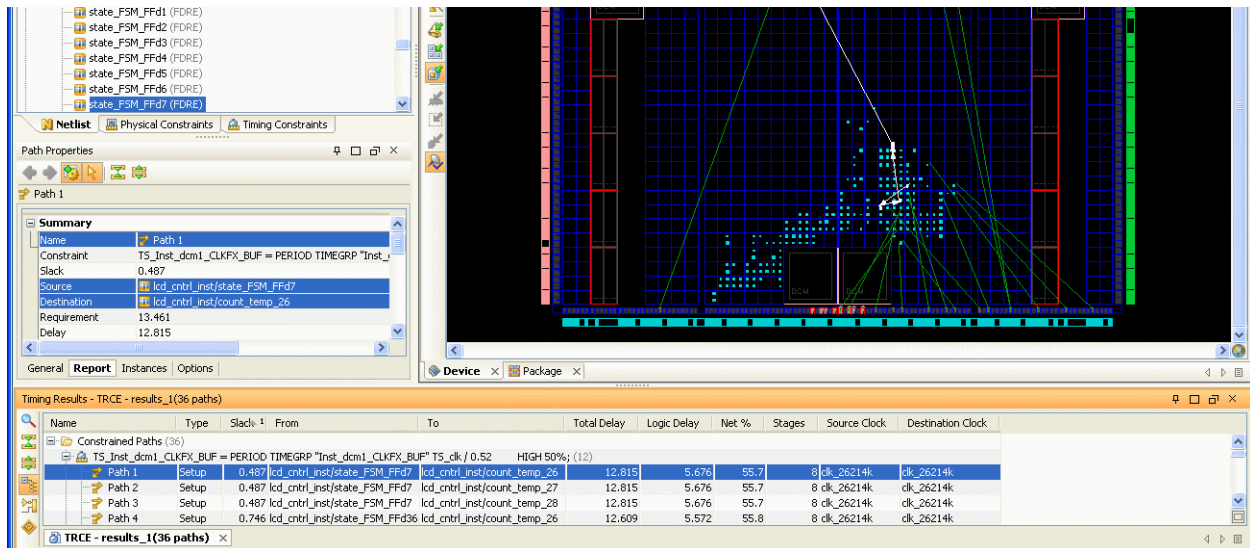


図 5-24 : PlanAhead ソフトウェアでタイミング パスを表示

4. [Device] ビューで、マウスをクリックおよびドラッグしてパス周辺を囲むボックスを描き、拡大表示します。

タイミング解析およびデザイン クロージャに関する PlanAhead ソフトウェアの全機能を説明する詳細なチュートリアルは、[Help] → [PlanAhead Tutorials] をクリックし、『PlanAhead ソフトウェア チュートリアル：デザイン解析およびフロアプラン』を参照してください。

5. [File] → [Exit] をクリックして PlanAhead ソフトウェアを終了します。

コンフィギュレーション データの作成

デザインを解析したら、コンフィギュレーション データを生成します。生成したコンフィギュレーション ビットストリームは、ターゲット デバイスにダウンロード するか、または PROM プログラム ファイルに変換します。

このチュートリアルでは、ザイリンクス シリアル PROM 用のコンフィギュレーション データを生成します。ターゲット デバイス用のビットストリームを生成するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペインで、stopwatch モジュールを選択します。
2. [Processes] ペインで [Generate Programming File] を右クリックし、[Process Properties] をクリックします。
3. [Process Properties] ダイアログ ボックスの [Category] で、[Startup Options] をクリックします。

4. [FPGA Start-Up Clock] プロパティを [JTAG Clock] に設定します。

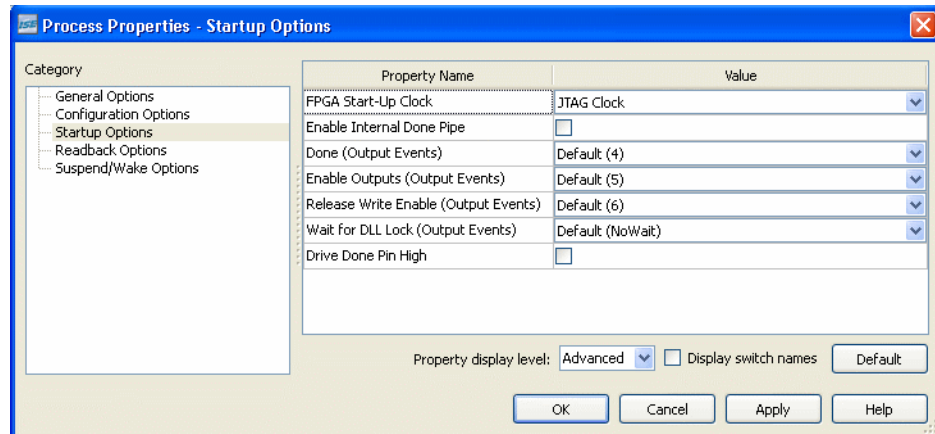


図 5-25 : [Process Properties] ダイアログ ボックスの [Startup Options] ページ

メモ : Select MAP またはシリアル スレーブをコンフィギュレーションする場合は、[CCLK] に設定します。

5. [OK] をクリックします。
6. [Processes] ペインで [Generate Programming File] をダブルクリックします。
BitGen により、コンフィギュレーション データを含む *design_name.bit* という名前のビットストリーム ファイル (このチュートリアルでは *stopwatch.bit*) が生成されます。
7. デザイン サマリで [Detailed Reports] → [Bitgen Report] をクリックし、プログラム ファイル生成レポートを表示します。コンフィギュレーション データを作成する際に指定したオプションが使用されていることを確認してください。

iMPACT を使用した PROM ファイルの生成

iMPACT を使用して 1 つのデバイスをプログラムするのに必要なファイルは、ビットストリーム ファイルのみです。デジタイズ チェーン コンフィギュレーションの複数のデバイスをプログラムする場合、または PROM を使用してデバイスをプログラムする場合は、iMPACT を使用して PROM ファイルを作成する必要があります。iMPACT では、任意の数のビットストリームを入力して、1 つのデバイスまたはデジタイズ チェーン コンフィギュレーションのデバイス用に 1 つまたは複数の PROM ファイルを生成できます。

iMPACT では、ウィザードの指示に従って次の操作を実行できます。

- PROM ファイルを作成します。
- デジタイズ チェーンにビットストリームを追加します。
- 追加のデジタイズ チェーンを作成します。
- 現在のビットストリームを削除して最初から始めるか、または現在の PROM ファイル コンフィギュレーションを保存します。

このチュートリアルでは、次の手順に従って PROM ファイルを生成します。

1. [Processes] ペインで [Configure Target Device] → [Generate Target PROM/ACE File] をダブルクリックします。

2. iMPACT の [iMPACT Flows] パネルで [Create PROM File (PROM File Formatter)] をダブルクリックします。

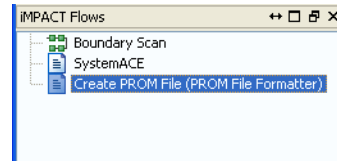


図 5-26 : iMPACT の [Create PROM File (PROM File Formatter)]

3. PROM File Formatter の [Select Storage Target] セクションで [Xilinx Flash/PROM] をクリックします。
4. 緑の矢印をクリックして次のセクションをアクティブにします。
5. [Add Storage Device(s)] セクションで、[Auto Select PROM] をオンにします。
6. 緑の矢印をクリックして次のセクションをアクティブにします。
7. [Enter Data] セクションの [Output File Name] に「stopwatch1」と入力します。
8. [Checksum Fill Value] が「FF」、[File Format] が [MCS] に設定されていることを確認します。

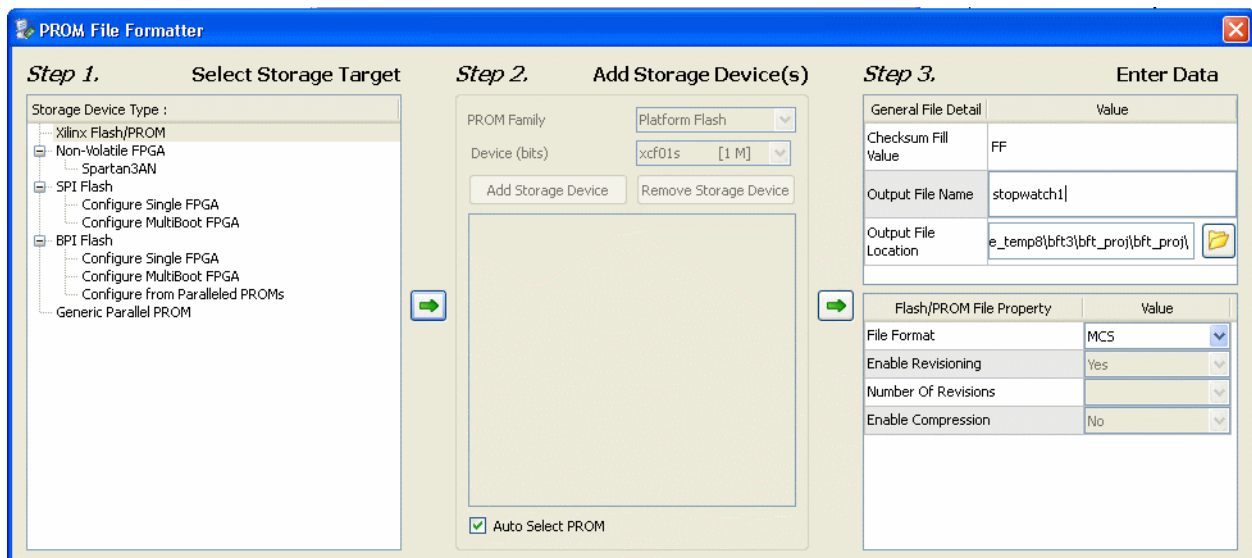


図 5-27 : PROM File Formatter

9. [OK] をクリックして PROM File Formatter を閉じます。
10. [Add Device] ダイアログ ボックスで [OK] をクリックし、stopwatch.bit ファイルを選択して [開く] をクリックします。
11. 別のデザイン ファイルをデータストリームに追加するかどうかを確認するダイアログ ボックスで、[No] をクリックします。
12. [OK] をクリックしてプロセスを終了します。
13. ワークスペースでデバイス グラフィックを選択し、[iMPACT Processes] パネルで [Generate File] をダブルクリックします。

iMPACT に PROM と関連付けられたビットストリーム ファイルが表示されます。

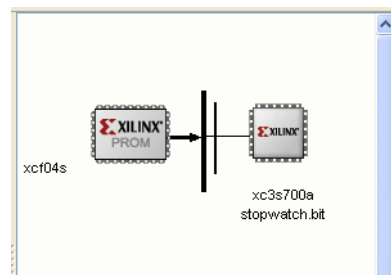


図 5-28 : PROM File

14. [File] → [Exit] をクリックして iMPACT を終了します。

15. プロジェクト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たら、[Yes] をクリックし、プロジェクト ファイルに「stopwatch_impact.ipf」という名前を付けます。

生成された stopwatch.bit、stopwatch1.mcs、および BIT ファイルと共に生成された MSK ファイルを使用すると、iMPACT を使用してデバイスをプログラムできます。デバイスのプログラムに関する詳細は、iMPACT ヘルプを参照してください。このヘルプは、iMPACT で [Help] → [Help Topics] をクリックすると表示されます。

これで、チュートリアル of デザイン インプリメンテーションが完了しました。デザイン フローの詳細およびインプリメンテーションのストラテジは、ISE ヘルプを参照してください。このヘルプは、ISE ソフトウェアで [Help] → [Help Topics] をクリックすると表示されます。

コマンド ラインを使用したインプリメンテーション

ISE ソフトウェアでは、インプリメンテーションの各プロセスで使用されたコマンド ライン コマンドとそのオプションおよび引数を、簡単に表示および抽出できます。この機能を使用すると、使用されたオプションを確認したり、コマンド バッチ ファイルを作成したりできます。

デザイン フローのどの段階でも、[Processes] ペインで [Design Utilities] の下にある [View Command Line Log File] をダブルクリックすると、使用されたコマンド ライン コマンドとそのオプションおよび引数を表示できます。このプロセスを実行すると、<source_name>.cmd_log というファイルが読み込み専用で開きます。編集可能なバッチ ファイルを作成するには、[File] → [Save As] をクリックし、ファイル名を指定してファイルを保存します。

また、このファイルの一部をコピーして、テキスト ファイルに貼り付けまたはドラッグ アンド ドロップすることも可能です。

ザイリンクスのコマンド プログラムのコマンド ライン オプションについては、『コマンド ライン ツール ユーザー ガイド』を参照してください。コマンド ライン オプションは、インプリメンテーション ツールごとに説明されています。このガイドは ISE ソフトウェア マニュアル コレクションに含まれています。ソフトウェア マニュアル コレクションを開くには、[Help] → [Software Manuals] をクリックするか、ザイリンクス Web サイトの [ソフトウェア マニュアル コレクション](#) にアクセスします。コマンド ライン オプションは、コマンド プロンプトでコマンド名を -h オプションを使用して実行しても表示されます。

タイミング シミュレーション

この章は、次のセクションから構成されています。

- 「タイミング シミュレーション フローの概要」
- 「入門」
- 「ModelSim を使用したタイミング シミュレーション」
- 「ザイリンクス ISim を使用したタイミング シミュレーション」

タイミング シミュレーション フローの概要

タイミング シミュレーションでは、配線済みデザインのブロックおよび配線遅延情報を使用するので、ワーストケースの回路の動作をより正確に評価できます。タイミング シミュレーションは、デザインの配置配線後に実行されます。

配置配線後のタイミング シミュレーションは、ザイリンクス デバイスの HDL デザイン フローで推奨されるステップです。タイミング シミュレーションでは、配置配線後に得られるタイミングおよびデザイン配置の詳細な情報が使用されます。これにより、実際のデバイス動作にほぼ一致するデザインのシミュレーションを実行できます。スタティック タイミング解析に加えてタイミング シミュレーションを実行すると、スタティック タイミング解析のみでは見つけることができない問題を検出できます。デザインを完全に検証するには、デザインをスタティックおよびダイナミックに解析する必要があります。

この章では、ModelSim シミュレータまたはザイリンクス ISim シミュレータを使用してタイミング シミュレーションを実行します。

入門

次に、このチュートリアルでタイミング シミュレーションを実行するための要件を示します。

必要なソフトウェア

ModelSim でシミュレーションを実行するには、ザイリンクス ISE® Design Suite 12 と ModelSim シミュレータがインストールされている必要があります。ModelSim のインストールおよび設定についての詳細は、第 4 章「ビヘイビア シミュレーション」を参照してください。ISim でシミュレーションを実行するには、ISE Design Suite 12 ソフトウェアがインストールされている必要があります。

必要なファイル

タイミング シミュレーション フローでは、次のファイルが必要です。

- デザイン ファイル (VHDL または Verilog)

第 5 章「[デザイン インプリメンテーション](#)」を完了し、デザインで配置配線が完了していることを前提とします。この章では、**NetGen** を使用して配置配線したデザインからシミュレーション ネットリストを生成します。このネットリストは、タイミング シミュレーションでデザインを表現するために使用されます。

- テストベンチ ファイル (VHDL または Verilog)

デザインをシミュレーションするには、デザインにスティミュラスを供給するテストベンチが必要です。ビヘイビア シミュレーションで使用したテストベンチを使用する必要があります。プロジェクトにテストベンチがない場合は、第 4 章の「[HDL テストベンチの追加](#)」を参照してください。

- ザイリンクス シミュレーション ライブラリ

タイミング シミュレーションでは、デザインをシミュレーションするのに **SIMPRIM** ライブラリが必要です。

ザイリンクス デザインでタイミング シミュレーションを実行するには、どの HDL シミュレータを使用する場合でも、**SIMPRIM** ライブラリが正しく設定されている必要があります。ザイリンクスで生成されたタイミング シミュレーションのネットリストには、インスタンス化されたプリミティブすべてが含まれており、これらのプリミティブのモデルが **SIMPRIM** ライブラリで提供されます。

第 4 章「[ビヘイビア シミュレーション](#)」を完了している場合は、**SIMPRIM** ライブラリは既にコンパイルされています。ザイリンクス シミュレーション ライブラリのコンパイルおよび設定についての詳細は、第 4 章の「[ザイリンクス シミュレーション ライブラリ](#)」を参照してください。

シミュレータの指定

stopwatch デザインのシミュレーションに使用するシミュレータを指定するには、次の手順に従います。

1. [Design] パネルの [Hierarchy] ペーンで、デバイス名 (xc3s700A-4fg484) を右クリックして [Design Properties] をクリックします。
2. [Design Properties] ダイアログ ボックスの [Simulator] で [ISim (VHDL/Verilog)] を選択するか、ModelSim のタイプと言語の組み合わせを選択します。

メモ：Project Navigator に統合されているシミュレータは、ModelSim とザイリンクス ISim のみです。NC-Sim、VCS などのほかのシミュレータを選択すると、NetGen によりそのシミュレータ用のシミュレーション ネットリストを作成するオプションは正しく設定されますが、Project Navigator からシミュレータを直接開くことはできません。シミュレーションの詳細およびサポートされるその他のシミュレータについては、『[合成/シミュレーション デザイン ガイド](#)』を参照してください。このガイドは ISE ソフトウェア マニュアル コレクションに含まれています。ソフトウェア マニュアル コレクションを開くには、[Help] → [Software Manuals] をクリックするか、ザイリンクスの Web サイトの [ソフトウェア マニュアル コレクション](#) にアクセスします。

ModelSim を使用したタイミング シミュレーション

ザイリンクス ISE ソフトウェアでは、Mentor Graphics 社の ModelSim シミュレータをフローに統合できます。ISE ソフトウェアから ModelSim による作業ディレクトリの作成、ソース ファイルのコンパイル、シミュレーションの初期化、およびシミュレータのプロパティの制御が可能です。

メモ：ISim でシミュレーションを実行する場合は、「[ザイリンクス ISim を使用したタイミング シミュレーション](#)」に進んでください。このチュートリアルでは、ModelSim シミュレータまたは ISim のどちらを使用しても、同じ結果を達成できます。

シミュレーション プロパティの指定

シミュレーション プロセス プロパティを設定するには、次の手順に従います。

1. [Design] パネルの [View] ペインで [Simulation] をオンにし、ドロップダウン リストから [Post-Route] を選択します。
2. [Hierarchy] ペインで、テストベンチ ファイル (stopwatch_tb) を選択します。
3. [Processes] ペインで [ModelSim Simulator] を展開し、[Simulate Post-Place & Route Model] を右クリックして [Process Properties] をクリックします。

メモ：[ModelSim Simulator] プロセスが表示されない場合は、「[シミュレータの指定](#)」を参照して [Design Properties] ダイアログ ボックスで ModelSim が選択されていることを確認します。正しく設定されているのに [ModelSim Simulator] プロセスが表示されない場合は、Project Navigator で modelsim.exe が認識されているかを確認します。このファイルの場所を設定するには、[Edit] → [Preferences] をクリックし、[Preferences] ダイアログ ボックスの左ペインで [ISE General] → [Integrated Tools] をクリックして、[Model Tech Simulator] で modelsim.exe ファイルを指定します。たとえば、c:\modeltech_xe\win32xoem\modelsim.exe のように指定します。

4. [Property display level] が [Advanced] に設定されていることを確認します。
[Advanced] に設定すると、すべてのプロパティが表示されます。
5. [Category] で [Simulation Model Properties] をクリックします。これらのプロパティを使用して、NetGen でシミュレーション ネットリストを生成する際のオプションを設定します。各プロパティの説明を表示するには、[Help] をクリックしてください。

次の図に示すような [Process Properties] ダイアログ ボックスが表示されます。このチュートリアルでは、デフォルトのプロパティ設定を使用します。

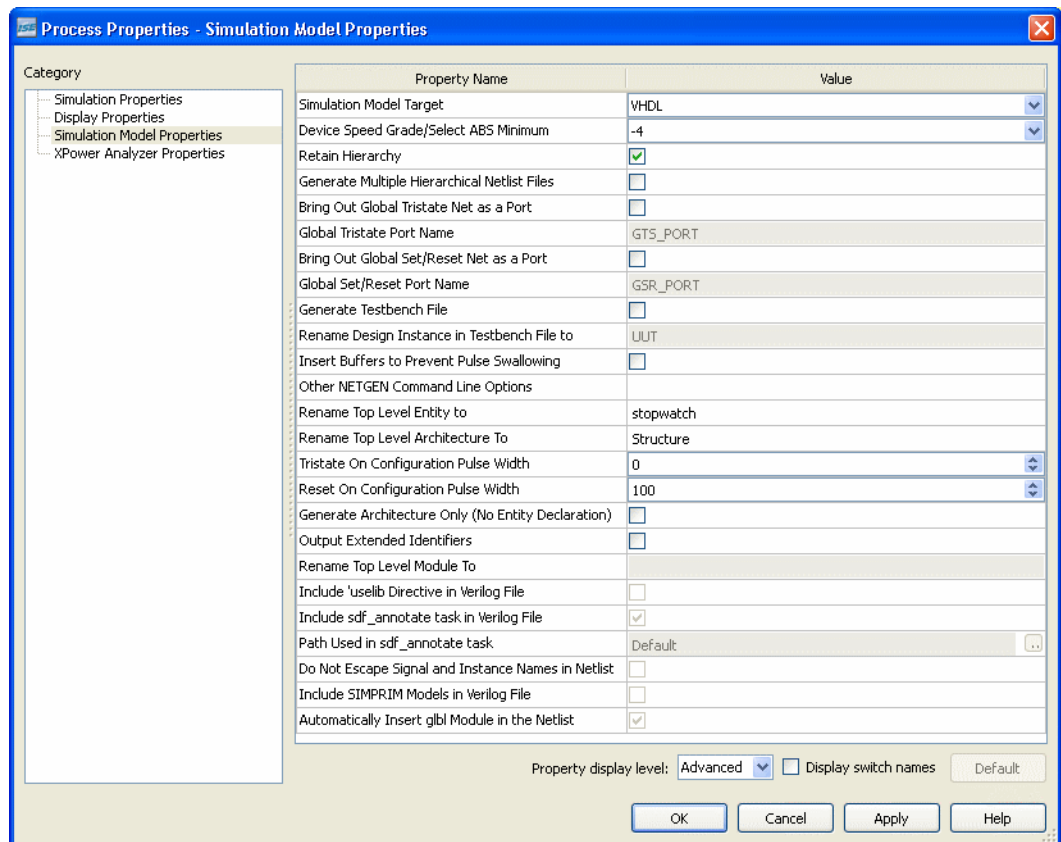


図 6-1：シミュレーション モデルのプロパティ

- [Category] で [Display Properties] をクリックします。このページのプロパティを使用して、ModelSim のウィンドウ表示を制御します。デフォルトでは、[Structure window]、[Signal window]、[Wave window] がオンになっており、ISE ソフトウェアからタイミング シミュレーションを起動したときに、[Structure]、[Objects]、[Wave] の 3 つのウィンドウが表示されます。ModelSim シミュレータのウィンドウ表示に関する詳細は、ModelSim のユーザー ガイドを参照してください。
- [Category] で [Simulation Properties] をクリックします。これらのプロパティを使用して、ModelSim でタイミング シミュレーションを実行するときのオプションを設定します。各プロパティの説明を表示するには、[Help] をクリックしてください。

次の図に示すような [Process Properties] ダイアログ ボックスが表示されます。[Simulation Run Time] プロパティを「2000 ns」に設定します。

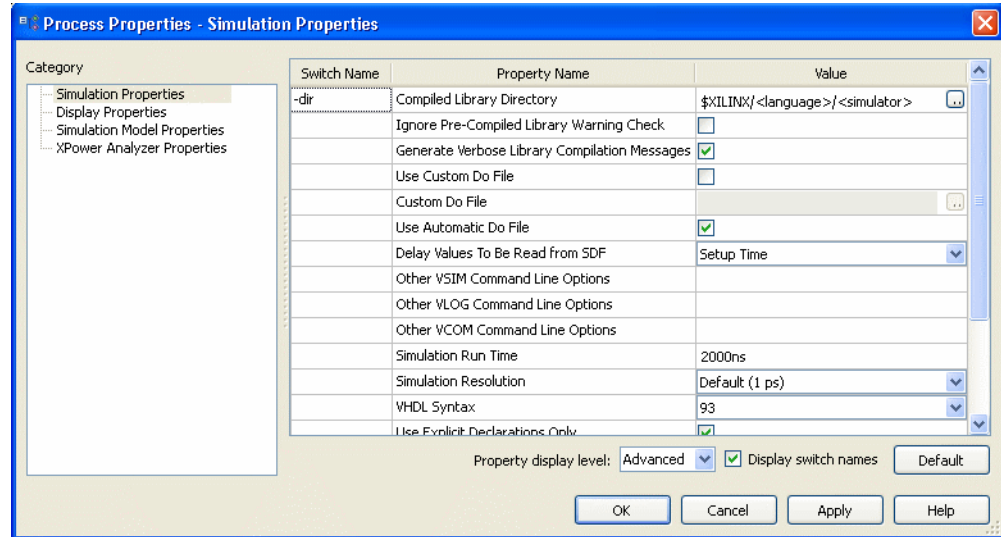


図 6-2 : [Simulation Properties] ページ

8. [OK] をクリックして [Process Properties] ダイアログ ボックスを閉じます。

シミュレーションの実行

タイミング シミュレーションを実行するには、[Processes] ペインで [Simulate Post-Place & Route Model] をダブルクリックします。

まず NetGen が実行されてタイミング シミュレータ モデルが生成されます。次に、ModelSim により作業ディレクトリの作成、ソース ファイルのコンパイル、デザインの読み込みが行われ、指定した時間だけシミュレーションが実行されます。

メモ：このデザインの大部分は 100Hz で動作するので、シミュレーションに時間がかかります。このため、短時間のシミュレーションではカウンタが動作していないように見えます。このチュートリアルでは、DCM の信号のみを調べて、正常に動作しているかどうかを検証します。

信号の追加

シミュレーション中の信号を表示するには、[Wave] ウィンドウに追加する必要があります。最上位のポートは、あらかじめ追加されています。その他の信号は、[Structure] ウィンドウの [Sim] タブでの選択に応じて [Objects] ウィンドウに表示されます。

[Wave] ウィンドウに信号を追加するには、次の 2 つの方法があります。

- [Objects] ウィンドウからドラッグ アンド ドロップする。
- [Objects] ウィンドウで信号をクリックし、[Add] → [Wave] → [Selected Signals] をクリックする。

次に、デザイン階層の信号を追加する方法を示します。このチュートリアルでは、波形に DCM 信号を追加します。

メモ： ModelSim のバージョン 6.0 以降を使用している場合は、すべてのウィンドウはデフォルトでメイン ウィンドウ内に表示されます。[Dock/Undock pane] ボタンをクリックすると、ウィンドウを切り離すことができます。



図 6-3 : [Dock/Undock pane] ボタン

1. [Structure] ウィンドウの [Sim] タブで、[uut] 階層を展開します。

次の図に、回路図フローの [Sim] タブを示します。Verilog または VHDL フローの [Sim] タブのレイアウトは、回路図のものとは異なる場合があります。

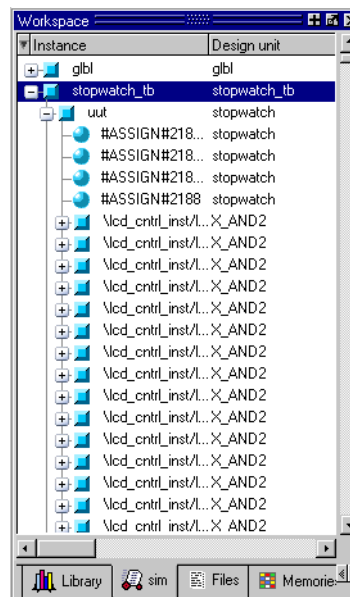


図 6-4 : [Sim] タブ (回路図フロー)

2. [Edit] → [Find] をクリックします。
3. [Find in sim] ダイアログ ボックスの [Find] に「X_DCM」と入力し、[Field] で [Entity/Module] をオンにして [Find Next] をクリックします。
4. X_DCM が表示されたら、X_DCM_SP を選択します。[Objects] ウィンドウに DCM のすべての信号が表示されます。
5. [Objects] ウィンドウをクリックし、[Edit] → [Find] をクリックします。
6. [Find in Objects] ダイアログ ボックスの [Find] に「CLKIN」と入力し、[Exact] をオンにして [Find Next] をクリックします。
7. [clkln] を、[Objects] ウィンドウから [Wave] ウィンドウにドラッグ アンド ドロップします。

8. 次の信号を、[Objects] ウィンドウから [Wave] ウィンドウにドラッグ アンド ドロップします。

- ◆ RST
- ◆ CLKFX
- ◆ CLK0
- ◆ LOCKED

メモ：複数の信号を選択するには、Ctrl キーを押しながら信号をクリックします。[Add] → [Wave] → [Selected Signals] をクリックしても、信号を追加します。

仕切りの追加

ModelSim の [Wave] ウィンドウに仕切りを追加して、信号を区別しやすくします。「DCM Signals」という仕切りを追加するには、次の手順に従います。

1. [Wave] ウィンドウの信号セクションを右クリックします。必要であれば、ウィンドウを切り離して最大化し、波形が大きく表示されるようにします。
2. [Insert Divider] をクリックします。
3. [Wave Divider Properties] ダイアログ ボックスの [Divider Name] に「DCM Signals」と入力します。
4. [OK] をクリックします。
5. 作成した仕切りを CLKIN 信号の上にドラッグします。

メモ：信号名を完全に表示するには、波形の最初の列を右側にドラッグします。信号名に含まれる階層は非表示にすることもできます。非表示するには、[Tools] → [Options] → [Wave Preferences] をクリックし、[Display Signal Path] ボックスに「2」と入力して [OK] をクリックします。

仕切りを追加すると、次の図に示すような表示になります。

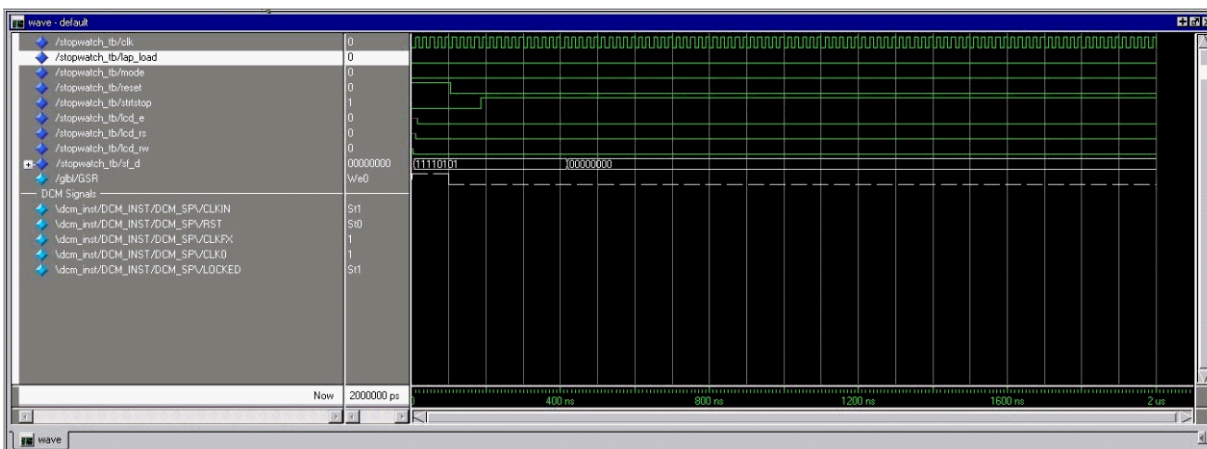


図 6-5：表示される波形

新しく追加した信号に対しては、波形は表示されていません。これは、ModelSim にこれらの信号のデータが記録されていないためです。デフォルトでは、シミュレーションを実行したときに [Wave] ウィンドウに追加されていた信号のデータのみが記録されるので、[Wave] ウィンドウに信号を追加した後、シミュレーションを再実行する必要があります。

シミュレーションの再実行

シミュレーションを再実行するには、次の手順に従います。

1. ツールバーの [Restart] ボタンをクリックします。



図 6-6 : [Restart] ボタン

[Restart] ダイアログ ボックスが開きます。

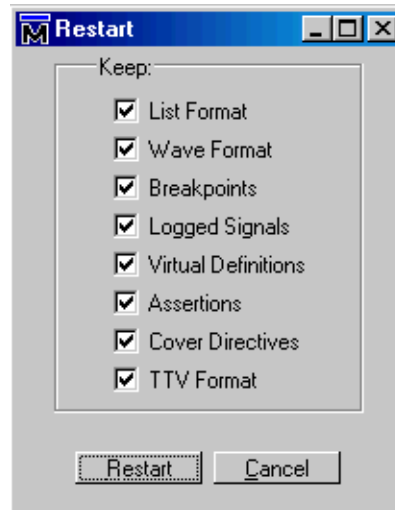


図 6-7 : [Restart] ダイアログ ボックス

2. [Restart] をクリックします。
3. ModelSim のコマンド プロンプトに「run 2000 ns」と入力し、[Enter] キーを押します。

A screenshot of the ModelSim command prompt showing the text 'VSIM 5> run 2000 ns'.

図 6-8 : run コマンドの入力

シミュレーションが 2000ns 間実行され、DCM の波形が [Wave] ウィンドウに表示されます。

信号の解析

DCM 信号が予測どおりに動作しているかを検証します。CLK0 は 50MHz、CLKFX は約 26MHz で動作する必要があります。DCM 信号は、LOCKED 信号が High になるまでは無効なので、LOCKED 信号が High になった後のものを解析する必要があります。

ModelSim では、波形にカーソルを追加して、遷移間の時間を確認できます。CLK0 の周期を確認するには、次の手順に従います。

1. [Add] → [To Wave] → [Cursor] を 2 回クリックして 2 本のカーソルを追加します。
2. 1 つ目のカーソルを、LOCKED_OUT 信号が High になった後の、CLK0 信号の最初の立ち上がりエッジにドラッグします。
3. 2 つ目のカーソルを 1 つ目のカーソルのすぐ右にドラッグします。

4. [Find Next Transition] を 2 回クリックして、カーソルを CLK0 信号の次の立ち上がりエッジに移動します。



図 6-9 : [Find Next Transition] ボタン

2 つのカーソル間の距離が、波形の下に表示されます。結果は 20000ps となり、周波数に変換すると 50MHz になります。この値は、テストベンチからの入力周波数に等しく、DCM の CLK0 出力周波数は正しい値であると判断できます。

同じ手順を使用して CLKFX を調べます。結果は 38462ps となり、周波数に変換すると約 26MHz になります。

シミュレーションの保存

信号またはステイミュラスを追加した後、またはシミュレーションを再実行した後に、[Wave] ウィンドウにリストされている信号を保存できます。保存した信号のリストは、シミュレーションを開始するときに簡単に開くことができます。信号のリストを保存するには、次の手順に従います。

1. [Wave] ウィンドウを選択した状態で、[File] → [Save As] をクリックします。
2. [Save Format] ダイアログ ボックスで、ファイル名をデフォルトの wave.do から dcm_signal_tim.do に変更します。

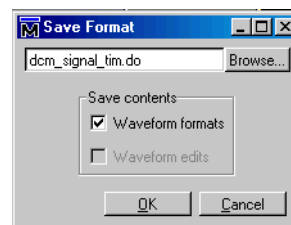


図 6-10 : [Save Format] ダイアログ ボックス

3. [OK] をクリックします。

シミュレーションを再実行するときに、[Wave] ウィンドウで [File] → [Load] をクリックしてこのファイルを読み込むことができます。

これで、タイミング シミュレーションは完了しました。この後、第 7 章「iMPACT チュートリアル」の手順に従って、デバイスをプログラムします。

ザイリンクス ISim を使用したタイミング シミュレーション

ISim を使用してシミュレーションを実行する場合は、このセクションの手順に従ってください。

シミュレーション プロパティの指定

シミュレーション プロセス プロパティを設定するには、次の手順に従います。

1. [Design] パネルの [View] ペインで [Simulation] をオンにし、ドロップダウン リストから [Post-Route] を選択します。
2. [Hierarchy] ペインで、テストベンチ ファイル (stopwatch_tb) を選択します。
3. [Processes] ペインで [ISim Simulator] を展開し、[Simulate Post-Place & Route Model] を右クリックして [Process Properties] をクリックします。
4. [Property display level] が [Advanced] に設定されていることを確認します。
[Advanced] に設定すると、すべてのプロパティが表示されます。
5. [Category] で [Simulation Model Properties] をクリックします。これらのプロパティを使用して、NetGen でシミュレーション ネットリストを生成する際のオプションを設定します。各プロパティの説明を表示するには、[Help] をクリックしてください。

このチュートリアルでは、デフォルトのプロパティ設定を使用します。

6. [Category] で [ISim Properties] をクリックします。これらのプロパティを使用して、ISim でタイミング シミュレーションを実行する際のオプションを設定します。各プロパティの説明を表示するには、[Help] をクリックしてください。

次の図に示すような [Process Properties] ダイアログ ボックスが表示されます。[Simulation Run Time] プロパティを「2000 ns」に設定します。

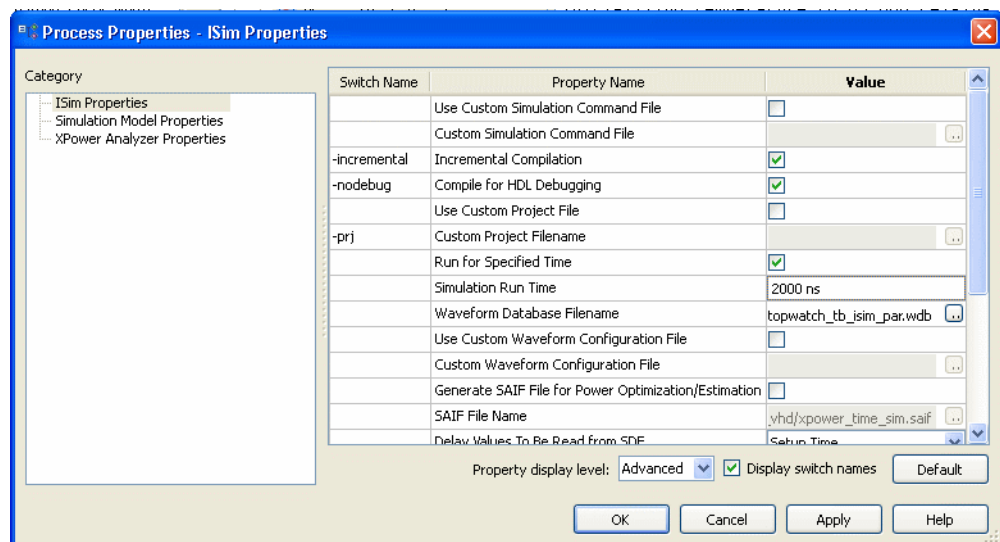


図 6-11 : [ISim Properties] ページ

7. [OK] をクリックして [Process Properties] ダイアログ ボックスを閉じます。

シミュレーションの実行

タイミング シミュレーションを実行するには、[Processes] ペインで [Simulate Post-Place & Route Model] をダブルクリックします。

シミュレーション プロセスを実行すると、NetGen により配置配線デザインからタイミング シミュレーション モデルが生成されます。その後、ISim によりソース ファイルがコンパイルされてデザインが読み込まれ、指定した時間だけシミュレーションが実行されます。

メモ：このデザインの大部分は 100Hz で動作するので、シミュレーションに時間がかかります。このため、短時間のシミュレーションではカウンタが動作していないように見えます。このチュートリアルでは、DCM の信号のみを調べて、正常に動作しているかどうかを検証します。

信号の追加

シミュレーション中の信号を表示するには、波形ウィンドウに追加する必要があります。最上位のポートは、あらかじめ追加されています。すべての外部信号（最上位ポート）および内部信号は、[Instances and Processes] パネルに表示されます。

次に、デザイン階層の信号を追加する方法を示します。このチュートリアルでは、波形に DCM 信号を追加します。

1. [Instances and Processes] パネルで [stopwatch_tb] 階層を展開します。
2. [uut] 階層を展開します。
3. [Inst_dcm1_DCM_SP_INST] をクリックします。
4. [Objects] パネルで [locked] 信号を右クリックし、[Add to Wave Window] をクリックします。

次の図に、VHDL フローの [Instances and Processes] および [Objects] パネルを示します。回路図または Verilog フローの [Instances and Processes] パネルのレイアウトは、VHDL のものと異なる場合があります。

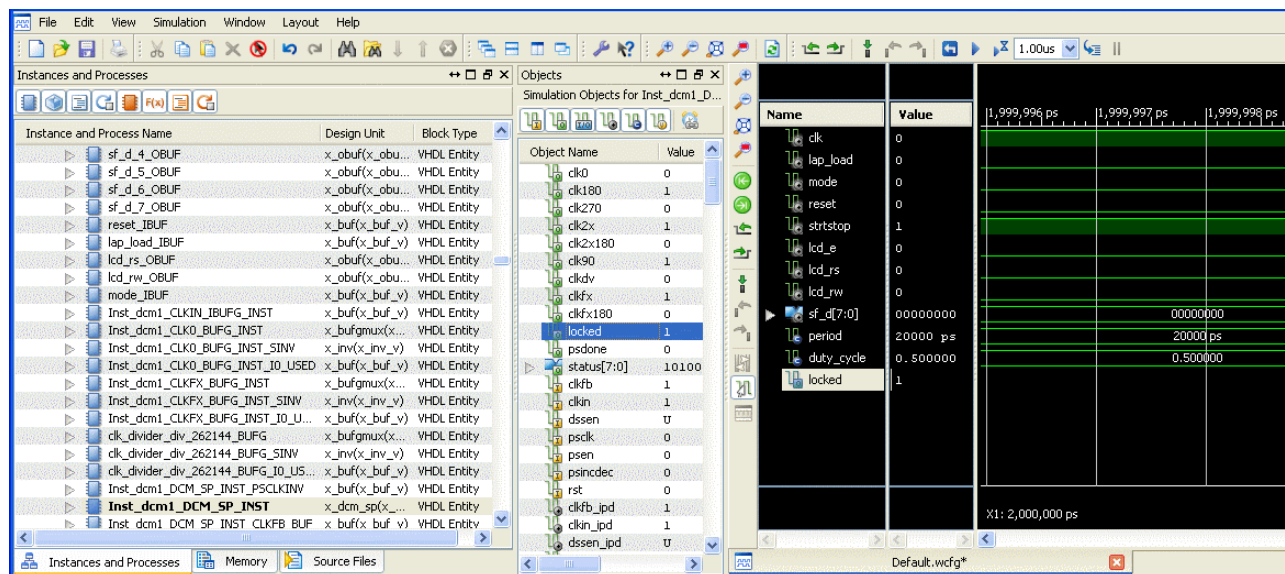


図 6-12 : [Instances and Processes] および [Objects] パネル (VHDL フロー)

5. X_DCM_SP の次の信号を、[Objects] パネルから波形ウィンドウにドラッグ アンド ドロップ します。

- ◆ RST
- ◆ CLKFX
- ◆ CLK0
- ◆ CLKIN

メモ：複数の信号を選択するには、Ctrl キーを押しながら信号をクリックします。

信号名の表示の切り替え

信号名は、階層名を含む完全な名前を表示するか、階層名を表示せずに信号名だけを表示するかを選択できます。信号名表示を変更するには、次の手順に従います。

1. 波形ウィンドウで信号名を右クリックします。
2. [Name] → [Long] または [Name] → [Short] をクリックします。

信号名を完全に表示するには、波形の最初の列の幅を広げます。

次の図に示すような波形が表示されます。

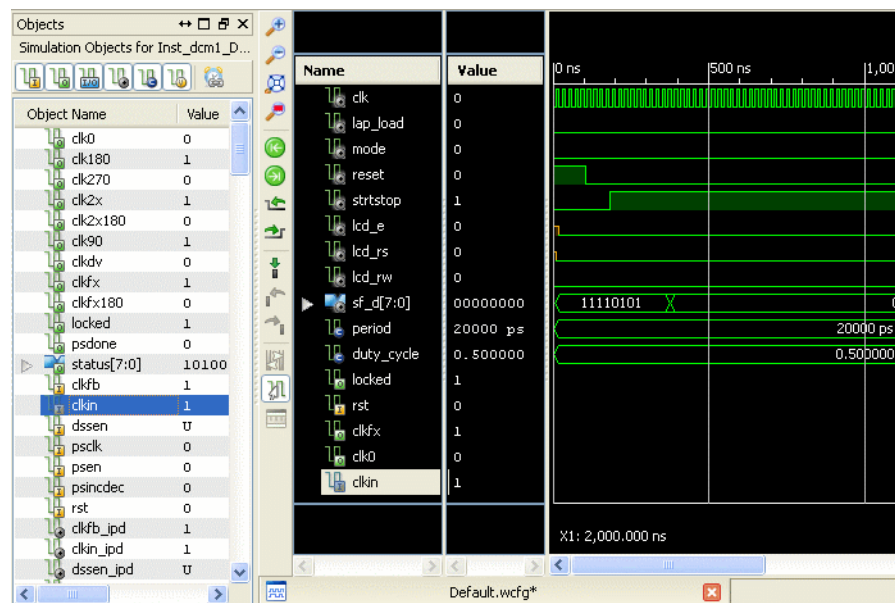


図 6-13：表示される波形

新しく追加した信号に対しては、波形は表示されていません。これは、ISim にこれらの信号のデータが記録されていないためです。デフォルトでは、シミュレーションを実行したときに波形ウィンドウに追加されていた信号のデータのみが記録されるので、波形ウィンドウに信号を追加した後、シミュレーションを再実行する必要があります。

シミュレーションの再実行

シミュレーションを再実行するには、次の手順に従います。

1. ツールバーの [Restart] ボタンをクリックします。



図 6-14 : [Restart] ボタン

2. ISim のコマンド プロンプトに「run 2000 ns」と入力し、Enter キーを押します。

```
* |run 2000 ns
```

図 6-15 : run コマンドの入力

シミュレーションが 2000ns 間実行され、DCM の波形が波形ウィンドウに表示されます。

信号の解析

DCM 信号が予測どおりに動作しているかを検証します。CLK0 は 50MHz、CLKFX は約 26MHz で動作する必要があります。DCM 信号は、LOCKED 信号が High になるまでは無効なので、LOCKED 信号が High になった後のものを解析する必要があります。

ISim では、波形にマーカを追加して、遷移間の時間を確認できます。CLK0 の周期を確認するには、次の手順に従います。

1. 必要に応じて、ツールバーのズーム ボタンを使用して波形を拡大表示します。
2. 波形ウィンドウのツールバーで [Snap to Transition] ボタンをクリックします。



図 6-16 : ツールバーの [Snap to Transition] ボタン

3. LOCKED 信号が High になった後の、CLK0 信号の最初の立ち上がりエッジをクリックして、カーソルを CLK0 信号の次の立ち上がりエッジにドラッグします。

波形ウィンドウの下部に開始点の時間、終了点の時間、および時間差が表示されます。この例では時間差は 20.0ns であり、周波数に変換すると 50MHz になります。この値はテストベンチからの入力周波数に等しく、DCM の CLK0 出力周波数は正しいと判断できます。

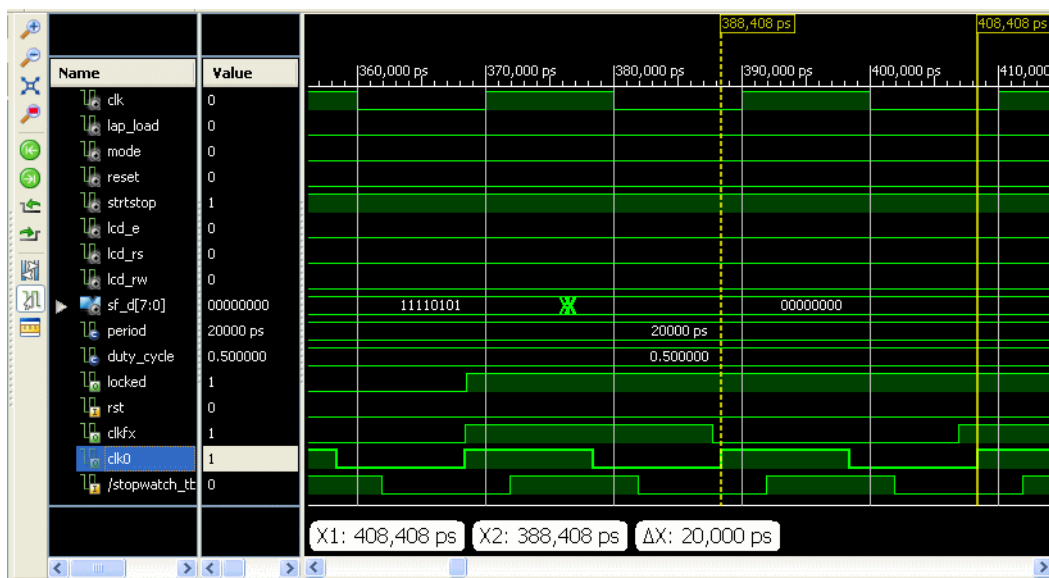


図 6-17：波形ウィンドウに表示される遷移間の時間

4. 同じ手順を使用して CLKFX を調べます。結果は 38.5 ns となり、周波数に変換すると約 26MHz になります。

これで、タイミングシミュレーションは完了しました。この後、第 7 章「iMPACT チュートリアル」の手順に従って、デバイスをプログラムします。

iMPACT チュートリアル

この章では、iMPACT を使用したファイルの生成方法およびデバイスのプログラム方法を説明します。iMPACT では、プラットフォーム ケーブル USB などの数種類の平行ケーブルを使用してプログラムできます。また、ビットストリーム ファイル、System ACE™ ソリューション ファイル、PROM ファイル、および SVF/XSVF ファイルを生成できます。SVF/XSVF ファイルは、チェーンを作成し直さずに再生できます。

このチュートリアルは、次のセクションから構成されています。

- 「デバイス サポート」
- 「サポートされるダウンロード ケーブル」
- 「サポートされるコンフィギュレーション モード」
- 「入門」
- 「バウンダリ スキャン コンフィギュレーション モードの使用」
- 「バウンダリ スキャン コンフィギュレーションのトラブルシューティング」
- 「SVF ファイルの作成」

デバイス サポート

サポートされているデバイスの詳細は、ザイリンクス Web サイトから [『ISE Design Suite 12: インストール、ライセンス、リリース ノート』](#) を参照してください。

サポートされるダウンロード ケーブル

次のケーブルがサポートされています。

平行ケーブル IV

コンピュータの平行ポートに接続し、バウンダリ スキャン モードでのコンフィギュレーションに使用します。詳細は、ザイリンクス Web サイトから [ザイリンクス 平行ケーブル IV のデータシート](#) を参照してください。

プラットフォーム ケーブル USB

コンピュータの USB ポートに接続し、バウンダリ スキャン モードでのコンフィギュレーションに使用します。詳細は、ザイリンクス Web サイトから [プラットフォーム ケーブル USB のデータシート](#) を参照してください。

プラットフォーム ケーブル USB-II

コンピュータの USB ポートに接続し、バウンダリ スキャン モードでのコンフィギュレーションに使用します。詳細は、ザイリックス Web サイトから [プラットフォーム ケーブル USB-II のデータシート](#) を参照してください。

サポートされるコンフィギュレーション モード

iMPACT では、FPGA、CPLD、PROM (XCFxxS および XCFxxP)、サードパーティの SPI/BPI フラッシュ デバイスのバウンダリ スキャン コンフィギュレーション モード がサポートされています。

入門

次に、このチュートリアルのこの章を実行するための要件を示します。

コンフィギュレーション ファイルの生成

この章では、stopwatch デザインの次のファイルが必要です。

- BIT ファイル
ヘッダ情報およびコンフィギュレーション データを含むバイナリ形式のファイル
- MCS ファイル
PROM コンフィギュレーション情報を含む ASCII 形式のファイル
- MSK ファイル
BIT ファイルと同じコンフィギュレーション コマンドを含むが、コンフィギュレーション データの代わりにマスク データを含むバイナリ ファイル。マスク データは、デバイスのコンフィギュレーションには使用されず、検証に使用されます。マスク ビットが 0 の場合、ビットはビット ストリーム データと比較され、1 の場合は比較されません。このファイルは、BIT ファイルと共に生成されます。

これらのファイルは、[第 5 章「デザイン インプリメンテーション」](#) で生成されています。

チュートリアル プロジェクト ファイルは、ザイリックスの Web サイトの [ISE Design Suite 12 チュートリアル](#) からダウンロードできます。VHDL、Verilog、または回路図デザイン フローのプロジェクト ファイルをダウンロードしてください。

ケーブルの接続

iMPACT を起動する前に、ケーブルの平行ポート コネクタをコンピュータの平行ポート に接続し、ケーブルを Spartan™-3 スタータ キットのデモ ボードに接続します。ボードの電源が入っていることを確認してください。

ソフトウェアの起動

このセクションでは、iMPACT ソフトウェアを ISE® ソフトウェアから起動する方法とスタンドアロンで起動する方法を説明します。

Project Navigator からの iMPACT の起動

Project Navigator から iMPACT を起動するには、次の図に示すように、[Design] パネルの [Processes] ペインで [Manage Configuration Project (iMPACT)] をダブルクリックします。

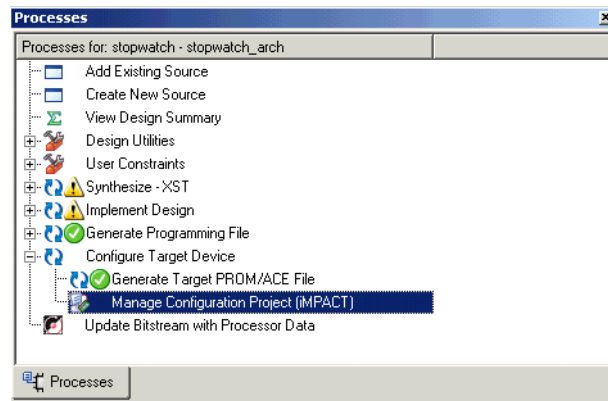


図 7-1 : Project Navigator からの iMPACT の起動

スタンドアロンでの iMPACT の起動

ISE プロジェクトを介さずにスタンドアロンで iMPACT を起動するには、次のいずれかの方法を使用します。

- PC のみ : [スタート] → [すべてのプログラム] → [Xilinx ISE Design Suite 12.1] → [ISE デザイン ツール] → [ツール] → [iMPACT] をクリックします。
- PC または Linux : コマンド プロンプトで「impact」と入力します。

バウンダリ スキャン コンフィギュレーション モードの使用

このチュートリアルでは、バウンダリ スキャン コンフィギュレーション モードを使用します。このモードでは、JTAG 規格に準拠したデバイスで構成されるチェーンに対してバウンダリ スキャン操作を実行できます。チェーンには、ザイリンクス デバイスおよびザイリンクス以外のデバイスの両方を含めることができますが、ザイリンクス以外のデバイスでは操作が制限されます。操作を実行するには、ケーブルがコンピュータに接続されており、JTAG ピン (TDI、TCK、TMS、TDO) とボードがケーブルで接続されている必要があります。

バウンダリ スキャン コンフィギュレーション モードの指定

iMPACT で新規プロジェクトを作成するには、コンフィギュレーション モードおよびプログラムするデバイスを指定します。バウンダリ スキャン モードを選択するには、次の手順に従います。

1. [File] → [New Project] をクリックします。
2. [Automatically create and save a project] ダイアログ ボックスで、[Yes] をクリックします。
3. [Welcome to iMPACT] ダイアログ ボックスで [Configure devices using Boundary-Scan (JTAG)] をオンにします。

4. [Automatically connect to cable and identify Boundary-Scan chain] が選択されていることを確認します。

メモ：このドロップダウン リストで [Enter a Boundary-Scan chain manually] を選択すると、手動でデバイスを追加してチェーンを作成できます。このオプションを使用すると、SVF/XSVF プログラム ファイルを生成できます。ただし、できる限りチェーンが自動的に検出および識別されるように設定してください。

5. [OK] をクリックします。

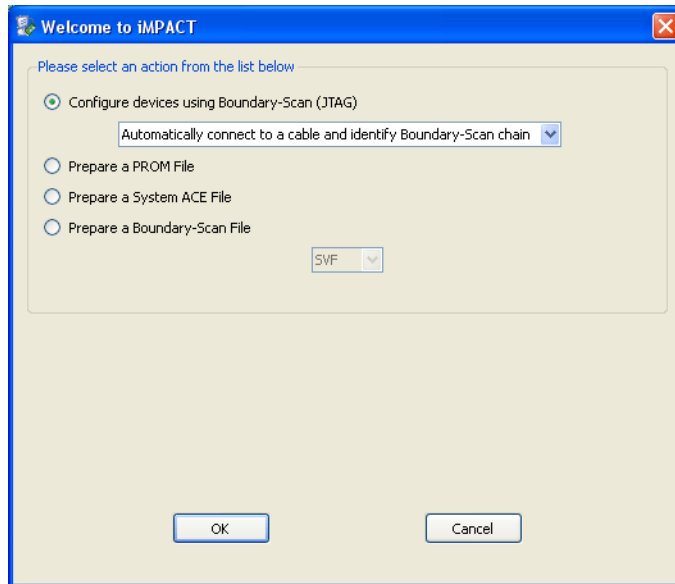


図 7-2：iMPACT ウィザードを使用して自動バウンダリ スキャンを選択

デバイスにデータが渡され、バウンダリ スキャン チェーンのサイズおよび構成が自動的に識別されます。サポートされているザイリンクス デバイスの場合は、識別されて名前が表示されます。その他のデバイスは、「UNKNOWN」と表示されます。次に、チェーン内の各デバイスがハイライトされ、コンフィギュレーション ファイルまたは BSDL ファイルを指定するダイアログ ボックスが開きます。

メモ：コンフィギュレーション モードまたは自動バウンダリ スキャン モードを選択するダイアログ ボックスが表示されなかった場合は、iMPACT ウィンドウ内で右クリックして [Initialize Chain] をクリックします。ボードに正しく接続されている場合は、チェーンが識別されます。識別されない場合は、「バウンダリ スキャン コンフィギュレーションのトラブルシューティング」を参照してください。

コンフィギュレーション ファイルの指定

チェーンを初期化すると、[Assign New Configuration File] ダイアログ ボックスが表示されます (図 7-3)。コンフィギュレーション ファイルは、デバイスをプログラムするのに使用され、次のような種類があります。

- ビット ストリーム ファイル (*.bit、*.rbit、*.isc) : FPGA のコンフィギュレーションに使用します。
- JEDEC ファイル (*.jed、*.isc) : CPLD のコンフィギュレーションに使用します。
- PROM ファイル (*.mcs、*.hex) : PROM のコンフィギュレーションに使用します。

最初のデバイス (XC3S700A) に対してコンフィギュレーション ファイルを指定するダイアログ ボックスが表示されたら、次の手順に従います。

1. プロジェクト ディレクトリから BIT ファイルを選択します。
2. [開く] をクリックします。
スタートアップ クロックが JtagClk に変更されたことを示す警告メッセージが表示されます。
3. [OK] をクリックします。

次の図に、[Assign New Configuration File] ダイアログ ボックスを示します。

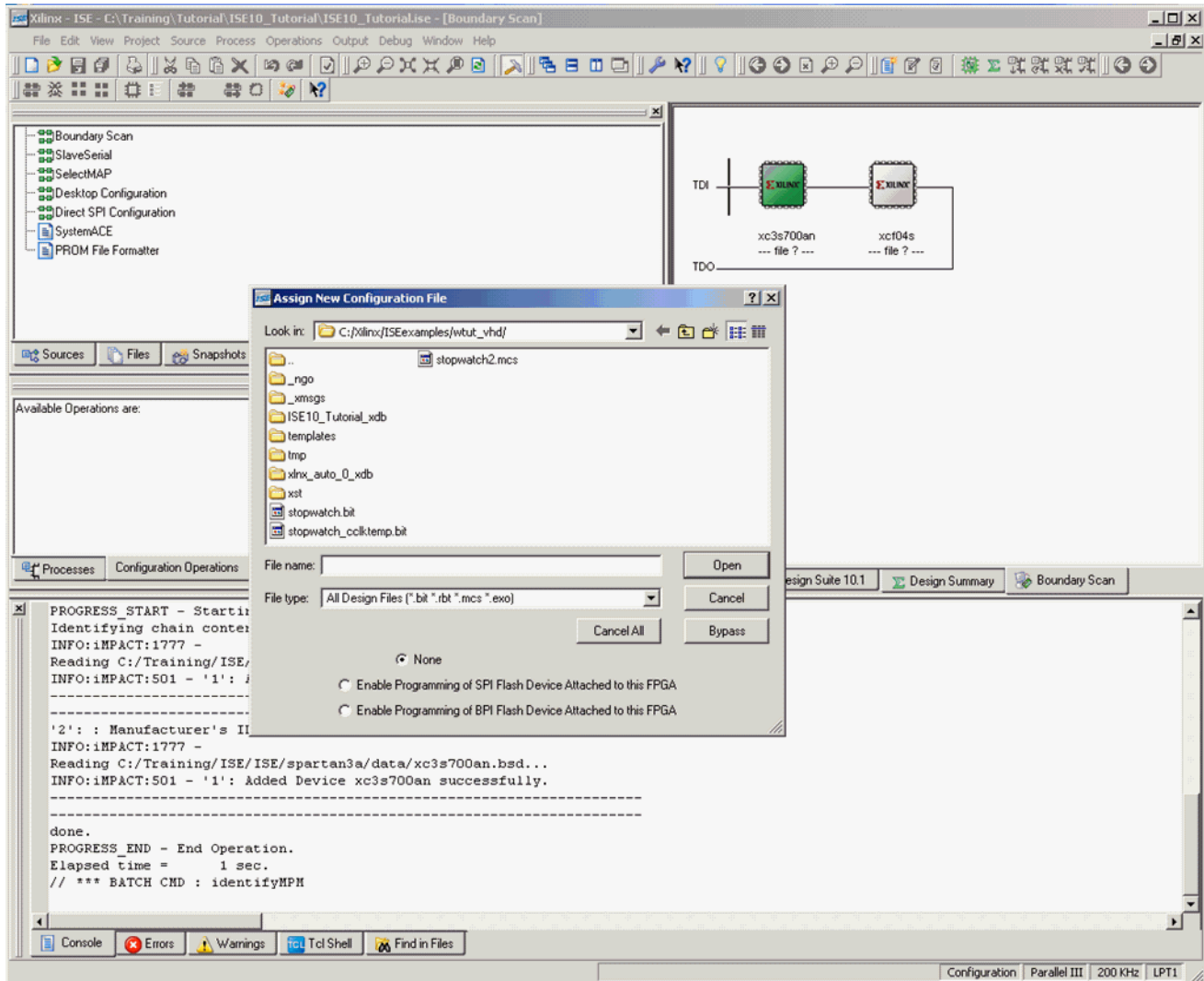


図 7-3 : [Assign New Configuration File] ダイアログ ボックス

メモ： コンフィギュレーション ファイルがない場合は、バウンダリ スキャン記述ファイル (BSDL または BSD) を代用できます。BSDL ファイルには、デバイスでバウンダリ スキャン操作のサブセットを実行可能にする情報が含まれています。ザイリンクスおよびザイリンクス以外のデバイスに対して BSDL ファイルが自動的に選択されるようにするには、[Assign New Configuration File] ダイアログ ボックスで [Bypass] をクリックします。

4. 2 番目のデバイス (XCF02S) に対しコンフィギュレーション ファイルを指定するダイアログ ボックスが表示されたら、プロジェクト ディレクトリから MCS ファイルを選択します。
5. [開く] をクリックします。

プロジェクト ファイルの保存

チェーンの記述を完了し、コンフィギュレーション ファイルを指定したら、次の手順に従いプロジェクト ファイル (IPF) を保存する必要があります。プロジェクトを保存するには、[File] → [Save Project As] をクリックします。[Save Project File] ダイアログ ボックスで、ディレクトリを指定してプロジェクト ファイルを保存します。iMPACT の再起動時にチェーンを復元するには、[File] → [Open Project] をクリックし、プロジェクト ファイルを指定します。

メモ：以前のバージョンの ISE ソフトウェアで使った CDF ファイル (コンフィギュレーション データ ファイル) も、iMPACT で開いて使用できます。また、IPF ファイルを CDF に抽出することも可能です。

プリファレンスの編集

バウンダリ スキャン コンフィギュレーションのプリファレンスを編集するには、[Edit] → [Preferences] をクリックします。次の図に示すダイアログ ボックスが表示されます。プリファレンスの詳細は、[Help] をクリックして iMPACT ヘルプを参照してください。このチュートリアルでは、デフォルトの値を使用するので、[OK] をクリックします。

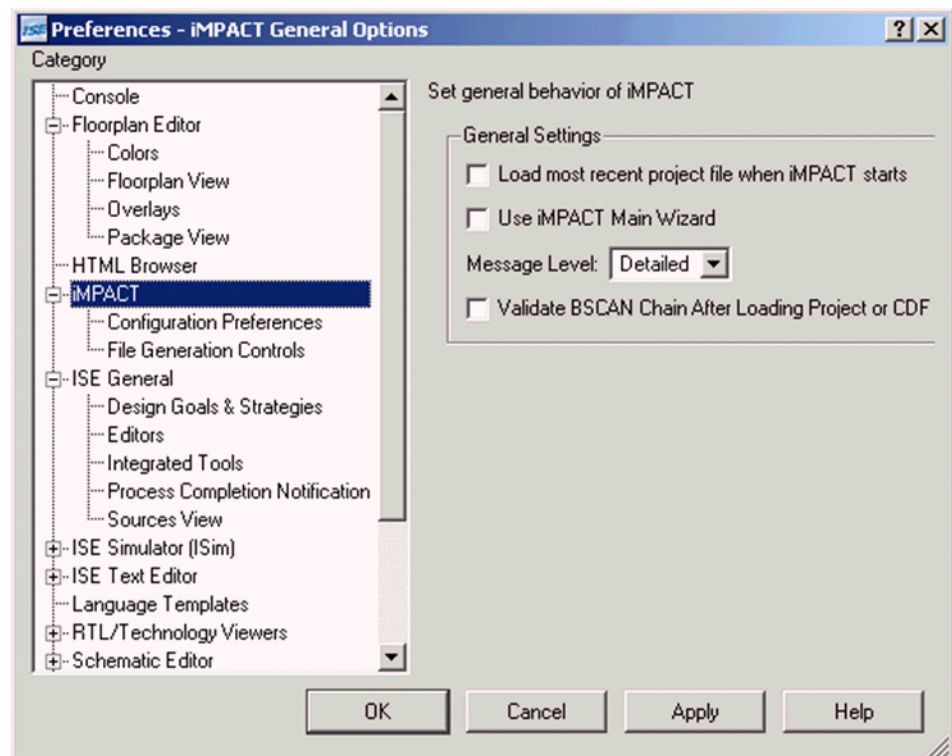


図 7-4 : [Preferences] ダイアログ ボックス

バウンダリ スキャン操作の実行

バウンダリ スキャン操作は、一度に 1 つのデバイスに対して実行できます。実行可能なバウンダリ スキャン操作は、デバイスおよびデバイスに指定されたコンフィギュレーション ファイルによって異なります。実行可能な操作コマンドのリストは、チェーン内のデバイスを右クリックすると表示されます。

デバイスを選択して操作を実行する場合、プリファレンスの設定に従ってチェーン内のほかのデバイスがすべて自動的に **BYPASS** または **HIGHZ** モードになります (プリファレンスの詳細は「[プリファレンスの編集](#)」を参照)。

操作を実行するには、デバイスを右クリックして操作コマンドのいずれかをクリックします。このセクションでは次に示すように、まず最初のデバイスのデバイス ID を取得し、次に検証オプションをオンにしてプログラムします。

1. XC3S700A デバイスで右クリックして、[Get Device ID] をクリックします。

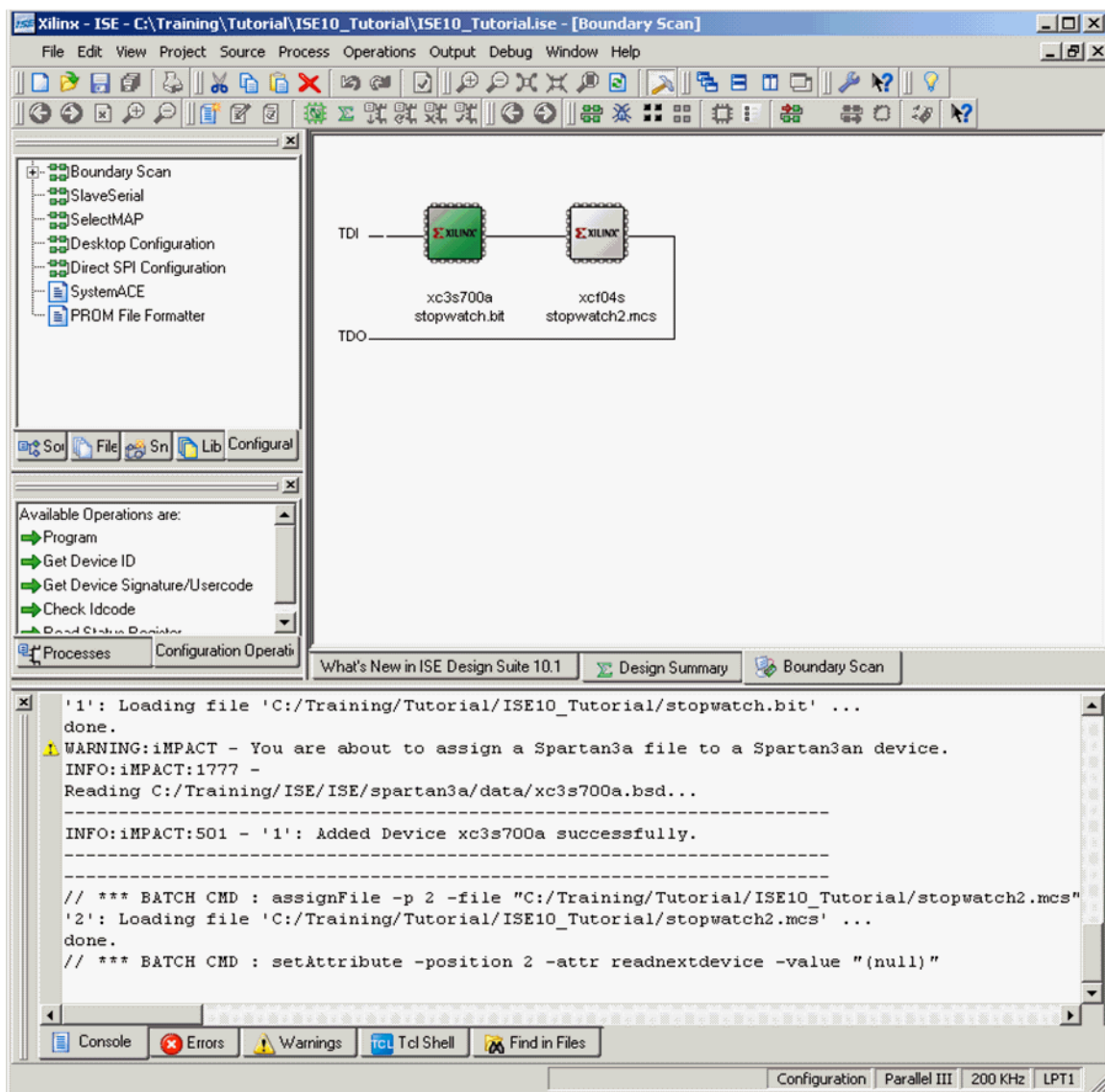


図 7-5：XC3S700A デバイスで実行可能なバウンダリ スキャン操作

Spartan-3A デバイスの IDCODE が取得され、結果が次の図に示すようにログ ウィンドウに表示されます。

```
// *** BATCH CMD : ReadIdcode -p 1
Maximum TCK operating frequency for this device chain: 0.
Validating chain...
Boundary-scan chain validated successfully.
'1': IDCODE is '00000010011000101000000010010011'
'1': IDCODE is '02628093' (in hex).
'1': : Manufacturer's ID =Xilinx xc3s700a , Version : 0
```

図 7-6 : [Get Device ID] コマンド実行後のログ ウィンドウ

- XC3S700A デバイスを右クリックして、[Set Programming Properties] をクリックします。
[Device Programming Properties] ダイアログ ボックスが表示されます。
- [Verify] をオンにします。
このオプションをオンにすると、デバイスがリードバックされ、先ほど生成した MSK ファイルと BIT ファイルが比較されます。
- [OK] をクリックしてプログラムを開始します。

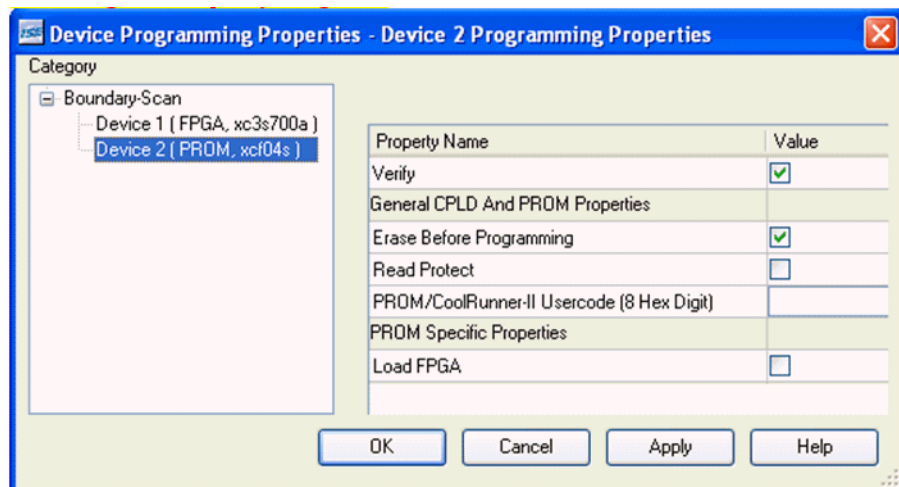


図 7-7 : XC3S700A デバイスのプログラム オプション

メモ : このダイアログ ボックスで設定可能なオプションは、選択したデバイスによって異なります。

5. XC3S700A デバイスを再度右クリックし、[Program] をクリックします。

プログラムが開始し、進捗状況を示すダイアログ ボックスが表示されます。同時に、ログ ウィンドウに実行されたすべての操作がレポートされます。

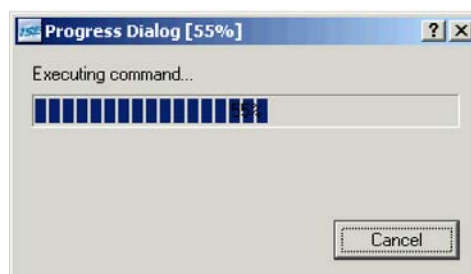


図 7-8 : [Progress Dialog] ダイアログ ボックス

プログラムが完了すると、次の図に示すように、プログラムが正しく終了したことを示す青色のメッセージが表示されます。このメッセージは、数秒間表示されます。

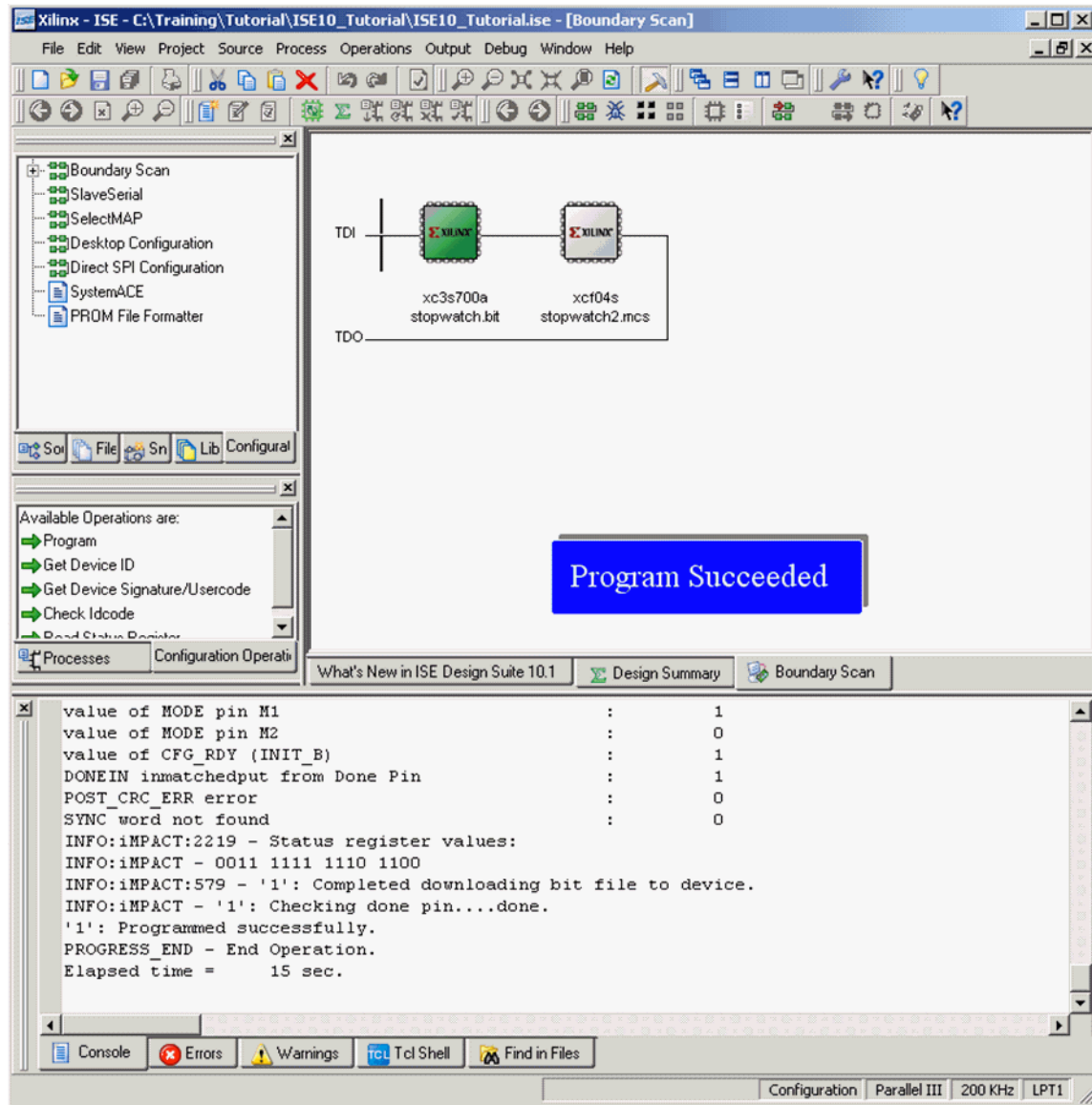


図 7-9 : プログラムが正しく完了したことを示すメッセージ

これで、デザインのプログラムおよび検証が完了しました。この段階で、ボードは正しく動作しており、ストップウォッチを開始、停止、およびリセットすることができるはずです。

バウンダリ スキャン コンフィギュレーションのトラブルシューティング

次のセクションでは、バウンダリ スキャン操作時にエラーが発生した場合のトラブルシューティングについて説明します。

ケーブル接続の確認

バウンダリ スキャン操作の実行中にエラーが発生した場合、まずケーブル接続が確立しているかを確認し、次にケーブルの自動識別機能が正しく機能しているかを確認してください。ボードとコンピュータをケーブルで接続しても接続が確立されない場合は、ウィンドウの空白部分を右クリックし、[Cable Auto Connect] または [Cable Setup] をクリックします。[Cable Auto Connect] をクリックすると、すべてのポートで接続が検索されます。[Cable Setup] をクリックすると、ケーブルおよびケーブルを接続するポートを選択できます。

接続が検出されると、次の図に示すように、メインウィンドウ下部に接続されているケーブルの種類、ケーブルが接続されているポート、およびケーブルの通信速度が表示されます。



図 7-10：ケーブル接続が確立されている場合のメイン ウィンドウの下部

メモ：ケーブルがシステムに接続されていてケーブルの自動識別でエラーが発生する場合は、[ザイリンクス アンサー #15742](#) を参照してください。

チェーンの設定の検証

バウンダリ スキャン操作の実行中にエラーが発生する場合は、チェーンが正しく設定されていて、iMPACT がデバイスと通信可能であることを確認してください。これは、チェーンを初期化すると簡単に確認できます。ウィンドウの空白部分で右クリックし、[Initialize Chain] をクリックします。ボードに正しく接続されている場合は、チェーンが識別されます。

チェーンが初期化できない場合は、ハードウェアが正しく設定されていないか、またはケーブルが正しく接続されていない可能性があります。チェーンが初期化できる場合は、単純な操作を実行してみてください。たとえば、チェーン内の各デバイスのデバイス ID を取得します。正しく実行できる場合は、ハードウェアは正しく設定されており、ケーブルも正しく接続されています。

チェーンのデバッグ機能を使用すると、次の図に示すように、手動で JTAG コマンドを入力することもできます。この機能は、コマンドをテストし、チェーンが正しく設定されていることを確認するときに使用できます。この機能を使用するには、[Debug] → [Enable/Disable Debug Chain] をクリックします。

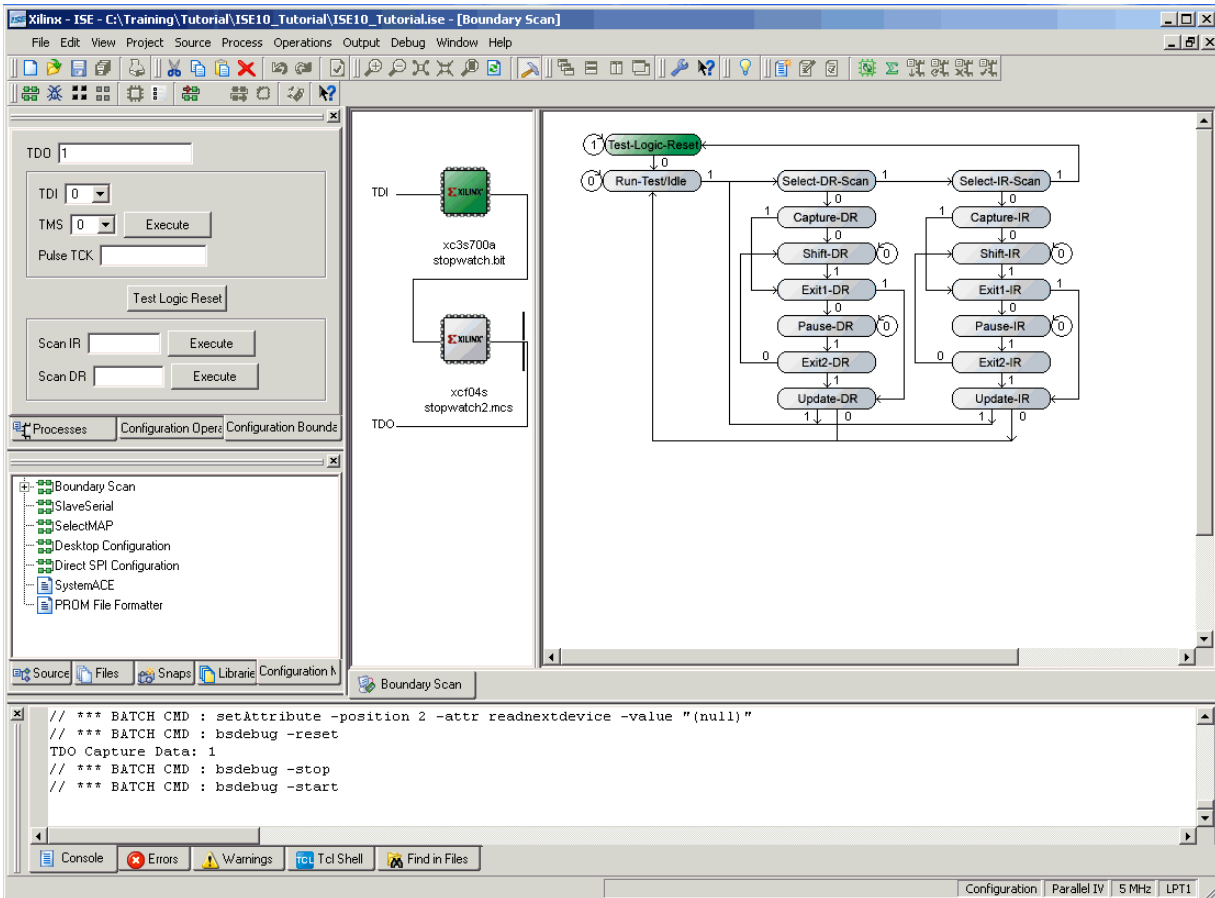


図 7-11：チェーンのデバッグ

iMPACT のバウンダリ スキャン デバッグに関して不明な点がある場合は、[Help] → [Help Topics] をクリックして iMPACT ヘルプを参照してください。トラブルシューティングに関して不明な点がある場合は、ザイリンクス Web サイトで [ウェブケース](#)を開いてください。

SVF ファイルの作成

このセクションはオプションです。「[バウンダリ スキャン コンフィギュレーション モードの使用](#)」セクションに従って正しくプログラムが完了していることを前提としています。このセクションでは、コンフィギュレーション情報のすべてが SVF ファイルに書き込まれます。

iMPACT では、SVF、XSVF、および STAPL の 3 種類のフォーマットのプログラム ファイルを生成できます。サードパーティのプログラム ソリューションを使用している場合、バウンダリ スキャン チェーンを手動で設定し、プログラム ファイルを生成する必要がある場合があります。これらのプログラム ファイルには、プログラム命令およびコンフィギュレーション データの両方が含まれており、自動テスト装置 (ATE) マシンおよびエンベデッド コントローラでバウンダリ スキャン操作を実行するときに使用されます。デバイスで操作は実行されないため、通常ケーブルを接続しておく必要はありません。

バウンダリ スキャン チェーンの設定

このセクションは、この章のこれまでのセクションに従いチェーンを検出していることを前提としています。検出していない場合は、「[SVF ファイル生成での JTAG チェーンの手動設定](#)」に従いチェーンを手動で定義してください。

SVF ファイル生成用の JTAG チェーンの設定

JTAG チェーンを設定するには、次の手順に従います。

1. [Output] → [SVF File] → [Create SVF File] をクリックします。
2. [Create a New SVF File] ダイアログ ボックスで、[ファイル名] に「getid」と入力し、[保存] をクリックします。
3. すべてのデバイス操作が SVF ファイルに記述されることを示すメッセージが表示されます。[OK] をクリックします。

SVF ファイル生成での JTAG チェーンの手動設定

「[バウンダリ スキャン コンフィギュレーション モードの使用](#)」の手順を完了している場合は、このセクションを実行する必要はありません。

バウンダリ スキャン チェーンは、手動で作成または変更することも可能です。手動で作成または変更するには、次の手順に従います。

1. [Boundary-Scan] タブをクリックして、バウンダリ スキャン モードであることを確認します。
これで、デバイスを 1 つずつ追加できます。
2. ウィンドウの空白部分を右クリックし、[Add Xilinx Device] または [Add Non-Xilinx Device] をクリックします。

コンフィギュレーション ファイルを選択するダイアログ ボックスが表示されます。

3. stopwatch.bit ファイルを選択し、[開く] をクリックします。

大きなカーソルが点滅している位置にデバイスが追加されます。既存のデバイスの間にデバイスを追加するには、デバイス間を接続する線を右クリックしてデバイスを追加します。

手順 2 と 3 を繰り返して、チェーンに stopwatch.mcs ファイルを追加します。

メモ：手動で作成したバウンダリ スキャン チェーンは、デバイスの一部分のみをプログラムする場合でもボード上のチェーンと一致させる必要があります。デバイスは、すべて iMPACT のウィンドウに表示されている必要があります。

SVF ファイルへの書き込み

SVF ファイルへの書き込み操作は、ケーブルを使用したバウンダリ スキャン操作と同じで、デバイスを右クリックして操作を選択するだけで実行できます。操作はいくつでも SVF ファイルに書き込むことができます。

このセクションでは、最初のデバイスのデバイス ID を取得し、2 番目のデバイスでプログラムを実行します。

デバイス ID を書き込むには、次の手順に従います。

1. 最初のデバイス (XC3S700A) をデバイスで右クリックし、[Get Device ID] を選択します。

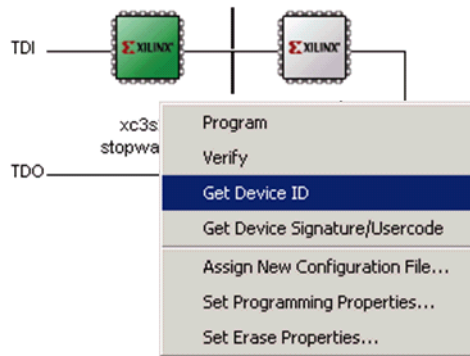


図 7-12 : バウンダリ スキャン操作の選択

[Get Device ID] を実行するのに必要な命令がファイルに書き込まれます。

2. 結果を確認するには、[View] → [View SVF-STAPL File] をクリックします。次の図に、[Get Device ID] コマンドを実行した後の SVF ファイルを示します。

```
// Created using Xilinx iMPACT Software [ISE - 10.1]
// Date: Mon May 19 21:44:00 2008

TRST OFF;
ENDIR IDLE;
ENDDR IDLE;
STATE RESET;
STATE IDLE;
FREQUENCY 1E6 HZ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
TIR 0 ;
HIR 14 TDI (3fff) SMASK (3fff) ;
HDR 2 TDI (00) SMASK (03) ;
TDR 0 ;
//Loading device with 'idcode' instruction.
SIR 6 TDI (09) SMASK (3f) ;
SDR 32 TDI (00000000) SMASK (fffffff) TDO (f2628093) MASK (0fbffff) ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
TIR 0 ;
HIR 14 TDI (3fff) SMASK (3fff) ;
HDR 2 TDI (00) SMASK (03) ;
TDR 0 ;
//Loading device with 'idcode' instruction.
SIR 6 TDI (09) ;
SDR 32 TDI (00000000) TDO (f2628093) ;
//Loading device with 'idcode' instruction.
SIR 6 TDI (09) ;
SDR 32 TDI (00000000) TDO (f2628093) ;
TIR 0 ;
HIR 14 TDI (3fff) ;
HDR 2 TDI (00) ;
TDR 0 ;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
SIR 20 TDI (0ffff) SMASK (0ffff) ;
SDR 3 TDI (00) SMASK (07) ;
```

図 7-13：チェーンの最初のデバイスのデバイス ID を取得した後の SVF ファイル

2 番目のデバイスのプログラムを SVF ファイルに書き込む場合は、次の手順に従います。

1. 2 番目の XCF02S デバイスを右クリックし、[Program] をクリックします。

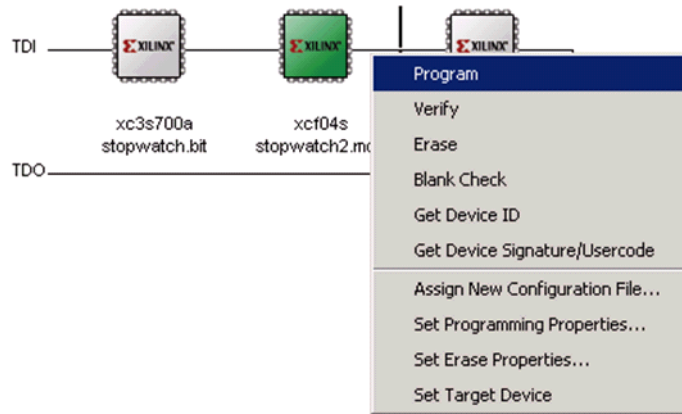


図 7-14 : XCF02S デバイスで実行可能なバウンダリ スキャン操作

2. [Programming Properties] ダイアログ ボックスで [OK] をクリックします。
2 番目のデバイスをプログラムするのに必要な命令およびコンフィギュレーション データが SVF ファイルに書き込まれます。

SVF ファイルへの書き込み停止

すべての操作を実行したら、これ以上命令が書き込まれないようにするために、ファイルを閉じる必要があります。プログラム ファイルへの書き込みを停止するには、[Output] → [SVF File] → [Stop Writing to SVF File] をクリックします。

後で別の操作を追加する場合は、[Output] → [SVF File] → [Append to SVF File] をクリックし、書き込む SVF ファイルを指定して [開く] をクリックします。

SVF または XSVF ファイルの再生

生成した SVF ファイルを再生して命令を検証するには、次の手順に従います。

1. 新規チェーンを手動で作成します。
2. ウィンドウの空白部分を右クリックして [Add Xilinx Device] をクリックし、SVF ファイルを選択して [開く] をクリックします。
3. バウンダリ スキャン チェーンの SVF ファイルを右クリックし、[Execute XSVF/SVF] をクリックします。

これでデバイスのプログラムは完了し『 ISE のアドバンス チュートリアル』コースは修了しました。

