

# ISim アドバンス チュートリアル

UG682 (v13.1) 2011 年 3 月 1 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.

© 2011 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 (v13.1) を翻訳したもので、内容に相違が生じる場合には原文を優先します。

資料によっては英語版の更新に対応していないものがあります。

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

# 目次

---

## このチュートリアルについて

ISim アドバンス チュートリアル .....	7
チュートリアルの内容 .....	7
チュートリアル フロー .....	7
ISE Project Navigator からの ISim の実行 .....	7
スタンドアロン ISim の実行 .....	8
その他のリソース .....	8

## 第 1 章 : ISim の概要

ISim の概要 .....	9
vhpcomp、vlogcomp .....	9
fuse .....	9
ISE シミュレーション実行ファイル .....	9
isimgui.exe .....	10

## 第 2 章 : ISE Project Navigator からの ISim の実行

ISim - Project Navigator 統合フローの概要 .....	11
はじめに .....	11
必要なソフトウェア .....	11
チュートリアル デザイン ファイルのインストール .....	11
デザインの概要 .....	12
論理ブロック .....	12
デザイン セルフチェック テストベンチ .....	13
デザインのシミュレーション .....	14
ISE Project Navigator でのプロジェクトの作成 .....	14
ビヘイビアー シミュレーションの起動 .....	20
次の操作 .....	22

## 第 3 章 : スタンドアロン ISim の実行

ISim スタンドアロン フローの概要 .....	23
はじめに .....	23
必要なソフトウェア .....	23
チュートリアル デザイン ファイルのインストール .....	23
デザインの概要 .....	24
論理ブロック .....	24
デザイン セルフチェック テストベンチ .....	25
シミュレーションの準備 .....	26
ISim プロジェクト ファイルの作成 .....	26
シミュレーション実行ファイルの構築 .....	26
デザインのシミュレーション .....	27
次の操作 .....	28

## 第 4 章 : ISim グラフィカル ユーザー インターフェイスの使用

ISim グラフィック ユーザー インターフェイスの概要 .....	29
ユーザー インターフェイスの使用 .....	30
デザインの検証 .....	34
信号の追加 .....	34
特定時間のシミュレーションの実行 .....	36
シミュレーションの再スタート .....	37

グループの追加 .....	38
仕切りの追加 .....	39
サブモジュールからの信号の追加 .....	40
信号および波形ウィンドウのプロパティの変更 .....	42
波形ウィンドウの設定の保存 .....	44
デザインのシミュレーション .....	44
マーカーの使用 .....	45
カーソルの使用 .....	46
複数の波形コンフィギュレーションの使用 .....	49
デザインのデバッグ .....	51
ソース コードの表示 .....	51
ブレークポイントの使用とソース コードの 1 行ずつの実行 .....	52
バグ修正の確認 .....	56
次の操作 .....	56

# このチュートリアルについて

---

## ISim アドバンス チュートリアル

ISim アドバンス チュートリアルでは、ザイリンクス PLD 設計者向けに ISim ソフトウェアを詳細に説明します。このチュートリアルを完成させると、ISim を使用した HDL シミュレーションによりデザインを解析、デバッグする方法を習得できます。

このチュートリアルは、Windows 環境で ISim ソフトウェアを実行するユーザー向けに設計されています。ほかのオペレーティング システムでは、一部の手順を正しく実行できるように変更する必要がある可能性があります。

## 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「ISim の概要」**では、ISim コンパイラ、リンカー、シミュレーション実行ファイル、およびグラフィカル ユーザー インターフェイス (GUI) などの ISim ソフトウェア環境を説明します。

**第 2 章「ISE Project Navigator からの ISim の実行」**では、ISE Project Navigator ソフトウェアから論理シミュレーションを実行する方法について説明します。

**第 3 章「スタンドアロン ISim の実行」**では、ISE Project Navigator 環境外にある ISim コンパイラ、リンカー、およびシミュレーション実行ファイルを使用して論理シミュレーションを実行する手順を紹介します。

**第 4 章「ISim グラフィカル ユーザー インターフェイスの使用」**では、ISim GUI で論理シミュレーションを確認、デバッグ、検証します。

## チュートリアル フロー

このチュートリアルでは、ISim を使用して論理 (ビヘイビアー) シミュレーションを実行する 2 つのフローを紹介します。

### ISE Project Navigator からの ISim の実行

このフローでは、ISE Project Navigator で使用可能なシミュレーション プロセスから ISim を起動します。このフローは、デザインをザイリンクスの FPGA または CPLD にインプリメントするために ISE Project Navigator プロジェクトを作成するときに最適です。このフローは、回路図やコアなどの HDL 以外のソースを含むデザイン向けです。これらのソースは Project Navigator により ISim でコンパイル可能な HDL ソースに変換されます。

このフローを使用する場合は、次の章に従ってください。

第 1 章「ISim の概要」

第 2 章「ISE Project Navigator からの ISim の実行」

第 4 章「ISim グラフィカル ユーザー インターフェイスの使用」

## スタンドアロン ISim の実行

このモードでは、コマンド ラインまたはバッチ ファイル モードで ISim プロジェクト ファイルを作成して、HDL リンカーおよびシミュレーション実行ファイルを実行することで、デザインをシミュレーションします。このフローは、HDL デザインの管理に Project Navigator を使用する必要がないユーザー向けです。

このフローを使用する場合は、次の章に従ってください。

第 1 章「ISim の概要」

第 3 章「スタンドアロン ISim の実行」

第 4 章「ISim グラフィカル ユーザー インターフェイスの使用」

## その他のリソース

このチュートリアルで説明される ISim の内容に関する詳細情報は、次の資料を参照してください。

ISim ユーザー ガイド：ザイリンクスのウェブサイトのソフトウェア マニュアル ページより参照できます。

- [http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx13\\_1/plugin\\_ism.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/plugin_ism.pdf)

メモ：ISim ヘルプは、ISim で F1 キーをクリックするか、[Help] メニューから参照できます。

ソフトウェア マニュアル：追加資料はこちらのザイリンクスのウェブサイトを参照してください。

- <http://japan.xilinx.com/support/documentation/index.htm>

チュートリアル ページ：『ISim アドバンス チュートリアル』のデザイン ファイルおよびその他の ISE Design Suite チュートリアルは、次のウェブサイトから入手してください。

- [http://japan.xilinx.com/support/documentation/dt\\_ise13-1\\_tutorials.htm](http://japan.xilinx.com/support/documentation/dt_ise13-1_tutorials.htm)

アンサー レコード：シリコン、ソフトウェア、IP に関する問題をアンサー データベースで検索したり、テクニカル サポートのウェブ ケースを開くには、次のザイリンクスのウェブサイトにアクセスしてください。

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

ザイリンクス フォーラム：ほかのザイリンクス ユーザーと特定のトピックについてディスカッションする場合は、次のサイトにあるザイリンクス ユーザー フォーラムにアクセスしてください。

- <http://forums.xilinx.com/xlnx/>

# ISim の概要

---

## ISim の概要

ザイリンクスの ISim は、VHDL、Verilog、および混合言語デザインで論理 (ビヘイビア) シミュレーションおよびタイミング シミュレーションを実行できるハードウェア記述言語 (HDL) シミュレーションです。

ISim 環境は、次の主要エレメントで構成されています。

- vhpcomp (VHDL パーサー)
- vlogcomp (Verilog パーサー)
- fuse (HDL エラボーレータおよびリンカー)
- ISE シミュレーション実行ファイル
- isimgui (ISim グラフィカル ユーザー インターフェイス)

メモ : ISim の詳細は、[『ISim ユーザー ガイド』](#)を参照してください。

## vhpcomp、vlogcomp

vhpcomp では VHDL ソース ファイル、vlogcomp では Verilog ソース ファイルが解析、コンパイルされます。パーサーで生成された解析出力は fuse で使用されて、シミュレーション カーネル ライブラリと共にオブジェクト コードおよびリンク オブジェクト コードが生成され、シミュレーション実行ファイルが作成されます。

## fuse

fuse コマンドは ISim で使用されるハードウェア記述言語 (HDL) エラボーレータおよびリンカーです。fuse コマンドでは、最上位デザイン ユニットにスタティック エラボーレーションを実行し、これらのユニットをオブジェクト にコンパイルします。次にデザイン ユニットのオブジェクト ファイルが共にリンクされて、シミュレーション実行ファイルが作成されます。

fuse コマンドでは、プロジェクト ファイル (.prj) に含まれている VHDL または Verilog ソース コードそれぞれに vlogcomp または vhpcomp を自動的に実行して、オンザフライでソースをコンパイルできます。

## ISE シミュレーション実行ファイル

シミュレーション実行ファイルは、fuse コマンドにより生成されます。ISim でデザインのシミュレーションを実行するには、このファイルを実行する必要があります。ISim を ISE Project Navigator のインターフェイスから実行した場合は、このファイルは ISE により実行されます。コマンド ラインからデザインのシミュレーションを実行するには、このファイルを指定する必要があります。

ります。シミュレーション実行ファイルでは、イベントドリブン型のシミュレーションが実行でき、Tcl コマンドを使用したシミュレーションの実行およびプローブが多様にサポートされています。

**メモ**：ISE シミュレーション実行ファイルの拡張子は Linux および Windows 共に .exe です。デフォルトの実行ファイル名フォーマットは、x.exe です。

## isimgui.exe

isimgui.exe (Linux では isimgui) は、ISim グラフィカル ユーザー インターフェイスです。このインターフェイスには、波形ウィンドウ、ツールバー、パネル、およびステータス バーが含まれています。メイン ウィンドウでは、デザインでシミュレーションが可能な箇所の表示、波形ウィンドウでの信号の追加および表示、ISim コマンドを使用したシミュレーションの実行、デザインの検証、およびデバッグを実行できます。



# ISE Project Navigator からの ISim の実行

---

## ISim - Project Navigator 統合フローの概要

ザイリンクス ISE Design Suite では、ISim との統合フローが提供され、Project Navigator (ISE) から直接シミュレーションを実行できます。ISim シミュレーションを実行するシミュレーション コマンドは、すべて ISE Project Navigator で生成され、このフローを使用してデザインをシミュレーションするときに自動的にバックグラウンドで実行されます。

## はじめに

### 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、次のいずれかのソフトウェアをインストールする必要があります。

- ISE WebPACK™ 13.1
- ISE Design Suite 13.1 Edition (Logic、DSP、Embedded、System) のいずれか

ザイリンクス ソフトウェアのインストールに関する詳細は、[『ISE Design Suite 13 : 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 デザイン ファイルのインストール

このチュートリアルのデザイン ファイルは、ザイリンクス Web サイトの[チュートリアル ページ](#)から入手できます。

- ウェブサイトにある『ISE Simulator アドバンス チュートリアル』に関連する ug682.zip ファイルをダウンロードします。
- デザイン ファイルを書き込み/読み取り権があるディレクトリに解凍します。

チュートリアル デザイン ファイルの内容は、次のとおりです。

表 2-1：チュートリアル デザイン ファイル

フォルダ	説明
sources	デザインの論理シミュレーションに必要な HDL ファイルが含まれています。
scripts	シミュレーションを実行するための未完成のスクリプト ファイルが含まれています。これらのスクリプト ファイルは、チュートリアルの進行に伴い完成させていきます。
completed	完成したスクリプト ファイル、シミュレーション ファイル、および波形コンフィギュレーション ファイルに加え、完成している ISE 13 プロジェクトのチュートリアル デザインが含まれており、進行中のデザイン ファイルと比較できます。

## デザインの概要

チュートリアル デザインは、Virtex®-5 デジタル クロック マネージャー (DCM) のダイナミック リコンフィギュレーション機能を簡単に示したものです。

Virtex-5 DCM を使用すると、デザインで次の式に基づいて出力クロックが生成されます。

$$\text{出力クロック} = \text{入力クロック} * (\text{倍周率} / \text{分周率})$$

DCM のダイナミック リコンフィギュレーション ポート (DRP) を使用すると、倍周率および分周率を定義し直してさまざまな出力周波数を生成できます。

## 論理ブロック

チュートリアル デザインは、次の論理ブロックから構成されています。

### drp\_dcm (drp\_dcm.vhd)

内部フィードバック付き、周波数制御出力、デューティ サイクル調整、およびダイナミック リコンフィギュレーション機能を含んだ Virtex-5 DCM マクロです。

CLKFX\_OUT 出力では、次の式で定義されたクロックが供給されます。

$$\text{CLKFX\_OUT} = \text{CLKIN\_IN} * (\text{倍周率} / \text{分周率})$$

たとえば、100MHz 入力クロックを使用するとき、倍周率を 6、分周率を 5 にすると 120MHz の CLKFX\_OUT 出力クロックが生成されます。

DCM の DRP ポートを使用すると、倍周率 (M) および分周率 (D) パラメーターをダイナミックに定義し直して異なる CLKFX\_OUT 周波数を生成できます。このチュートリアルでは、これらの倍周率および分周率のパラメーターが 16 ビット幅の DI\_IN ポートから DCM にどのように渡されるかを示します。

$$\text{DI\_IN}[15:8] = M - 1$$

$$\text{DI\_IN}[7:0] = D - 1$$

たとえば、M/D が 6/5 のとき、DI\_IN は 0504h になります。

## drp\_stmach (drp\_stmach.vhd)

このモジュールでは、ダイナミック リコンフィギュレーション コントローラーが記述されています。DRP コントローラーでは、ダイナミック リコンフィギュレーション サイクルを実行するために DCM の DRP 信号をアサート、監視します。

ダイナミック リコンフィギュレーション サイクルは、drp\_start 信号がアサートされると開始します。この手順に従うと、DRP コントローラーで該当する DCM の DRP ピンがアサートされ、ダイナミック リコンフィギュレーション サイクルが完了します。

drp\_done 信号は、ダイナミック リコンフィギュレーション サイクルが正しく完了したことを通知します。

## drp\_demo (drp\_demo.vhd)

チュートリアル デザインのトップ モジュールで、DCM マクロおよび DRP コントローラー モジュールが外部の I/O ポートに接続されています。

## drp\_demo\_tb (drp\_demo\_tb.vhd)

セルフチェック HDL テストベンチです。詳細は、「[デザイン セルフチェック テストベンチ](#)」を参照してください。

## デザイン セルフチェック テストベンチ

このデザインの機能性をテストできるように、セルフチェック テストベンチが提供されています ([sources] フォルダに含まれている drp\_demo\_tb.vhd を参照)。セルフチェック テストベンチには、予期する結果とシミュレーションでのサンプル値を比較する検証ルーチンまたは関数が含まれています。このデザインで提供されるセルフチェック テストベンチでは、次が実行されます。

- デザインのシステム クロック (clk\_in) 向けに 100MHz 入力クロックを生成
- デザインの出力周波数をダイナミックに変更するため、4 つの異なるテストを実行。各テストでは、drp\_start 信号を使用して DRP サイクルが開始され、出力クロックに個別の周波数が設定されます。次の表では、各テストでの理想的な出力周波数および倍周率/分周率パラメーターが表示されています。

表 2-1：理想的な出力周波数および倍周率/分周率パラメーター

テスト	周波数(MHz)	周期 (ps)	倍周率 (M)	分周率 (D)
1	75	13,332	3	4
2	120	8,332	6	5
3	250	4000	5	2
4	400	2,500	4	1

- 各テストでは、テストベンチで予期されるクロック周期とシミュレーション中に計測されたクロック周期が比較されます。比較結果に基づいて、テストが正常に完了したか、またはエラーが発生したを示すメッセージがシミュレーターで表示されます。
- シミュレーションの完了時に生成されるサマリ レポートには、正常に完了したテストとエラーが発生したテストの両方を含むリストが含まれています。

このデザインの機能の詳細は、デザインのソースに含まれるインライン コメントを参照してください。

ヒント：Project Navigator でテストベンチを作成するには、[Project] → [Create New Sources] をクリックし、[VHDL Testbench] または [Verilog Text Fixture] を選択します。

メモ：HDL テストベンチ設計方法の詳細は、アプリケーション ノート [XAPP199](#) 『Writing Effective Testbenches』を参照してください。

## デザインのシミュレーション

ISE - ISim 統合フローでは、ISE Project Navigator でデザインのビヘイビアー シミュレーションおよびタイミング シミュレーションを即座に実行できます。

このチュートリアルでは、最初に ISE Project Navigator でチュートリアル デザイン向けの ISE プロジェクトを作成します。次に、ビヘイビアー シミュレーションのプロパティを設定してから ISim シミュレーターを起動してデザインのビヘイビアー シミュレーションを実行します。

## ISE Project Navigator でのプロジェクトの作成

ISE Project Navigator の New Project Wizard を使用して、チュートリアル デザイン向けの ISE プロジェクトを作成します。

メモ：「[チュートリアル デザイン ファイルのインストール](#)」を読んで、このデザインに必要なファイルを入手してください。

### Project Navigator の起動と New Project Wizard の使用

次の手順に従い、Project Navigator を起動して、ISE プロジェクトを作成します。

1. [Xilinx ISE Design Suite] デスクトップ アイコンをダブルクリックして ISE Project Navigator を起動します。



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

2. [File] → [New Project] をクリックして New Project Wizard を起動します。
3. [図 2-2](#) に示すように [Create New Project] ページでプロジェクトの名前および保存場所を入力します。

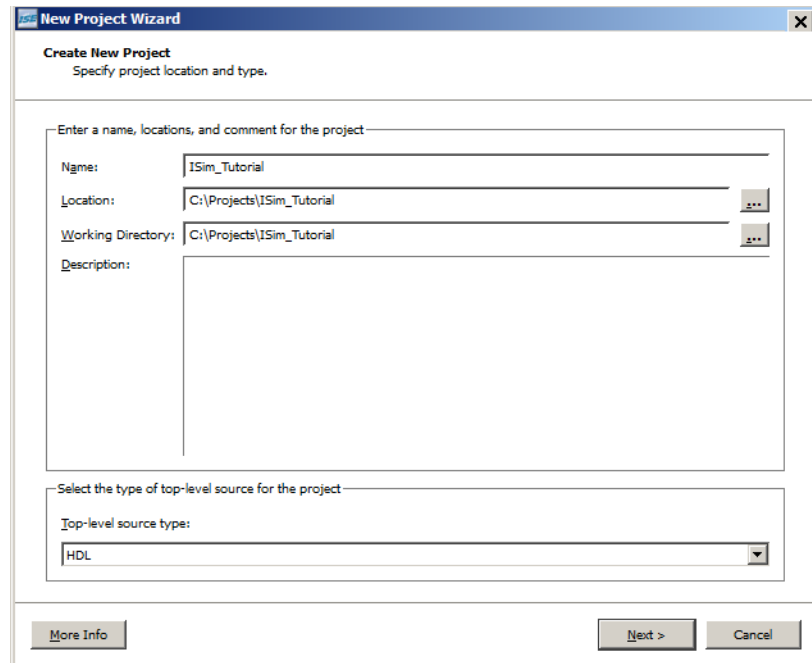


図 2-2 : New Project Wizard :[Create New Project] ページ

4. [Next] をクリックして次に進みます。
5. [Project Settings] ページで、デバイスおよびプロジェクトのプロパティを選択します。図 2-3 に示す設定に合わせて変更します。

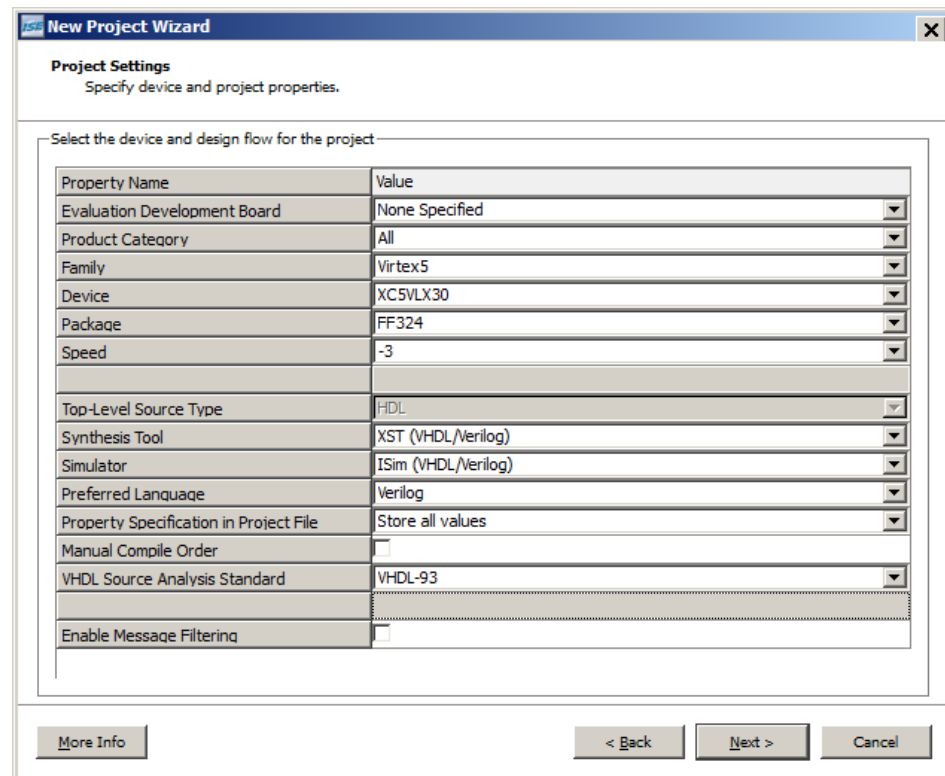


図 2-3 : New Project Wizard :[Project Settings] ページ

6. [Next] をクリックして次に進みます。
7. [Project Summary] ページの設定が図 2-4 に表示されている設定と同じであることを確認します。

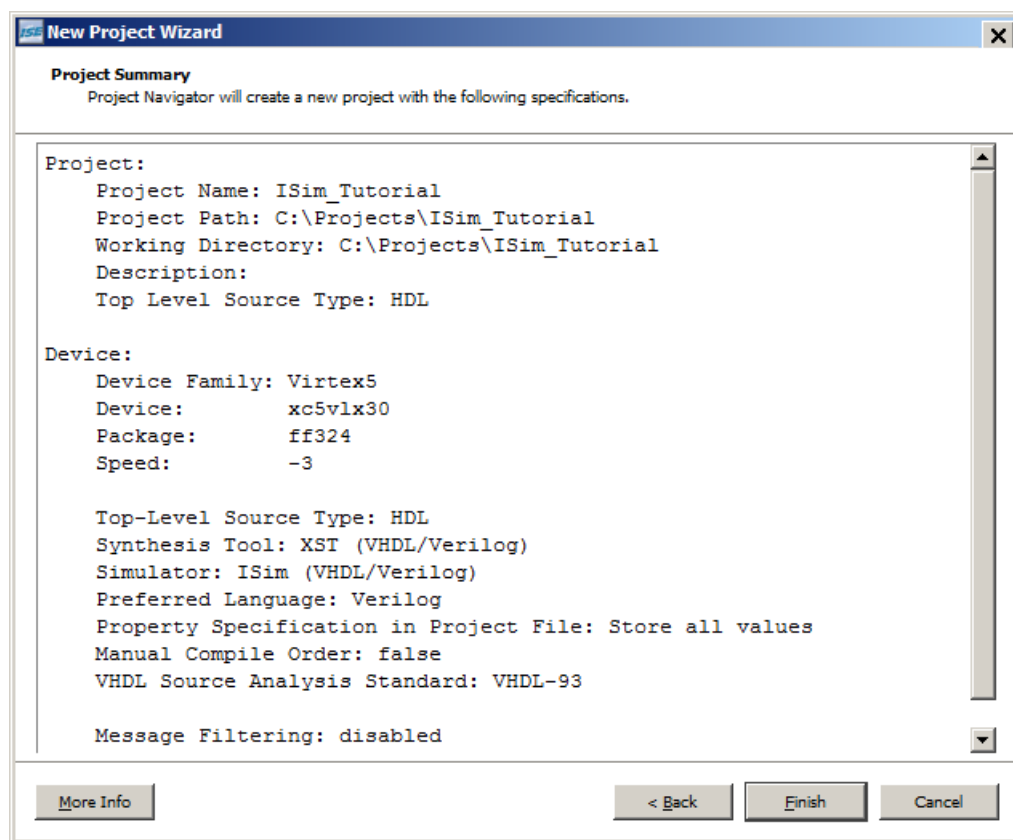


図 2-4 : [Project Summary] ページ

8. [Finish] をクリックして次に進みます。

## プロジェクトへのチュートリアル ファイルの追加

1. [Project] → [Add Source] をクリックしてソース ファイルを追加します。このチュートリアルでは、「チュートリアル デザイン ファイルのインストール」でソース ファイルを保存したディレクトリに移動します。
2. [sources] サブフォルダーで、すべてのファイルを選択して [開く] をクリックします。
3. 次のウィンドウで、これらのチュートリアル ソースに対して関連付けとライブラリが正しく設定されていることを確認します。図 2-5 の設定と比較してください。

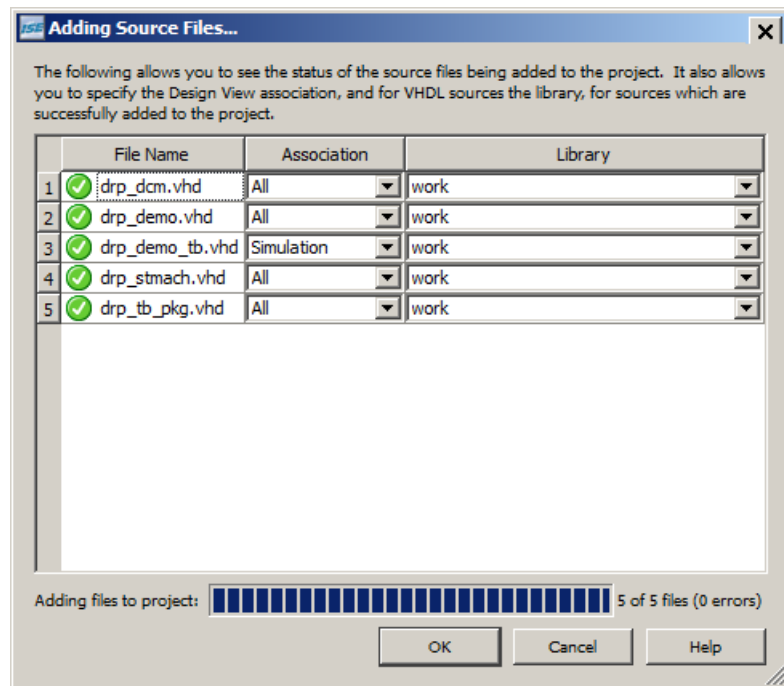


図 2-5 : ソース ファイルと関連付けのステータス

4. [OK] をクリックします。

これで、ソース ファイルがプロジェクトに追加されます。Project Navigator は、図 2-6 のように表示されます。

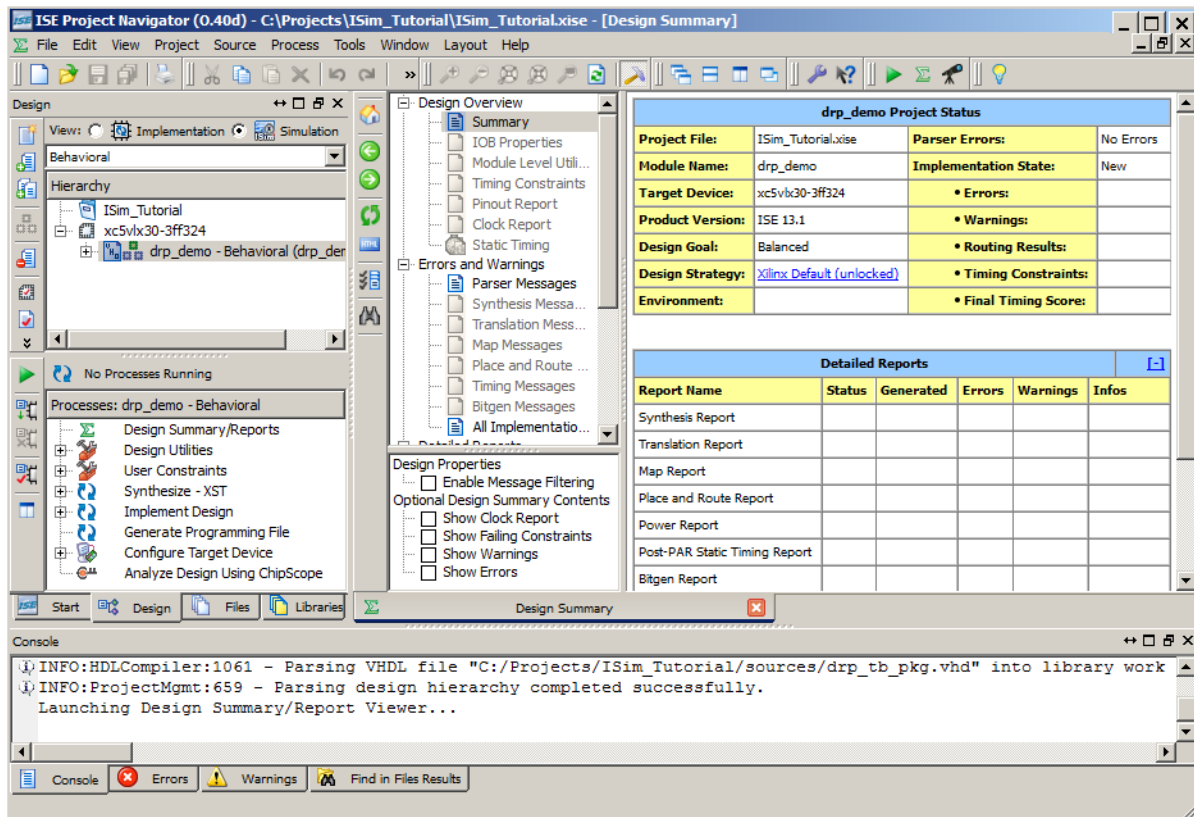


図 2-6：ISE Project Navigator のデザイン サマリ

## VHDL ライブラリの作成

次に、このデザインのテストベンチで使用される VHDL パッケージ (drp\_tb\_pkg.vhd) のユーザー VHDL ライブラリを作成する必要があります。VHDL パッケージには、テストベンチで検証ルーチンを実行するときに使用される VHDL 関数が含まれています。VHDL パッケージを作成したら、作業ライブラリから新しく作成した VHDL ライブラリに移動します。

次の手順に従い、VHDL ライブラリを作成します。

1. Project Navigator で [Project] → [New Source] をクリックし、New Source Wizard を開きます。
2. ソース タイプに [VHDL Library] を選択します。
3. VHDL のライブラリ名に「drp\_tb\_lib」と入力します (図 2-7)。

メモ：[Add to project] チェック ボックスをオンにしたままにします。

4. [Next] をクリックして次に進みます。



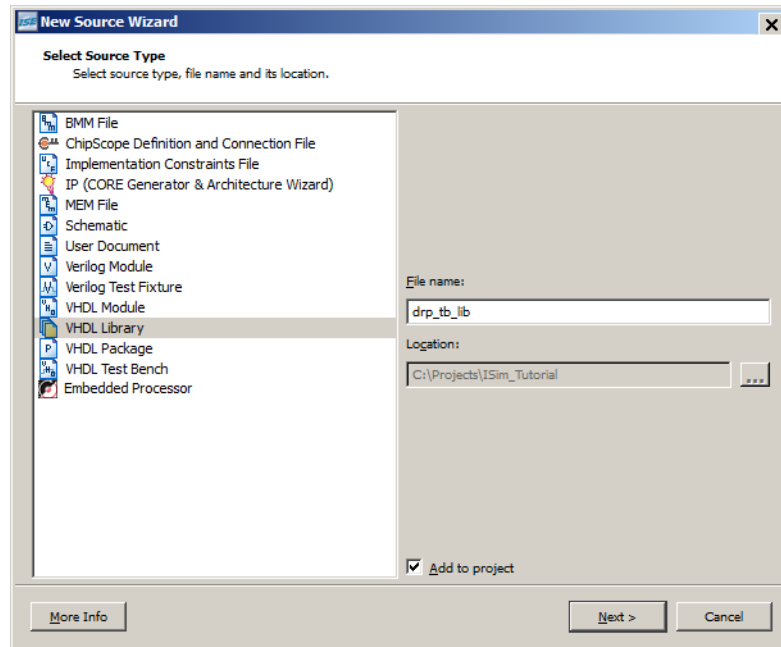


図 2-7 : ソース タイプの選択

5. [Summary] ページで [Finish] をクリックして New Source Wizard を終了します。

### ライブラリへの VHDL ファイルの移動

次の手順に従い、VHDL パッケージ ファイルを VHDL ライブラリ (drp\_tb\_lib) に移動します。

1. [Libraries] タブをクリックして [Libraries] パネルに表示を切り替えます。
2. [work] ライブラリを展開表示します (図 2-8)。

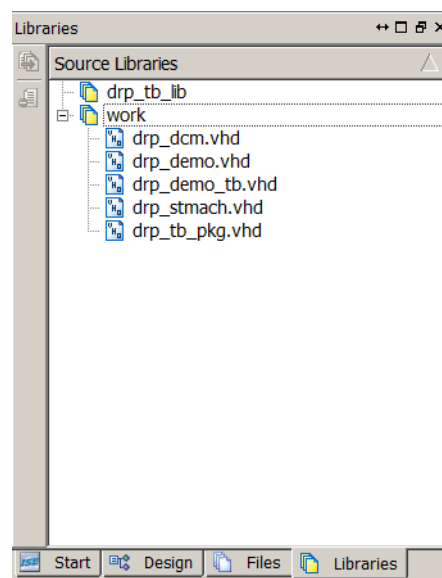


図 2-8 : [Libraries] パネル

3. drp\_tb\_pkg.vhd ファイルを右クリックして、[Move to Library] をクリックします。

4. [Move to Library] ダイアログ ボックスで、VHDL パッケージ ファイル `drp_tb_pkg.vhd` の移動先ライブラリに `drp_tb_lib` を選択します。
5. [OK] をクリックします

これで、新しい VHDL ライブラリ `drp_tb_lib` に VHDL パッケージ ファイル `drp_tb_pkg.vhd` が含まれていることが確認できます (図 2-9)。

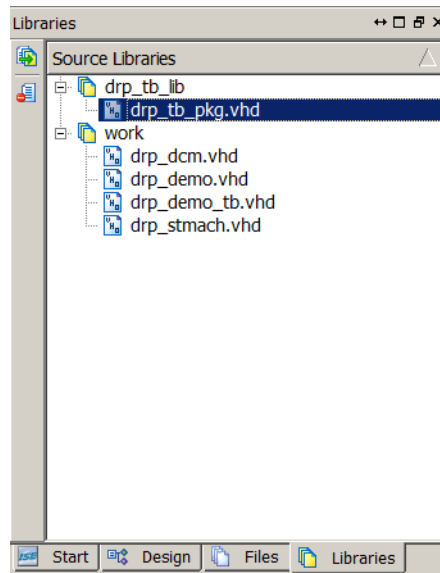


図 2-9：ソース ライブラリ

## ビヘイビア シミュレーションの起動

これでチュートリアル デザイン向けに ISE プロジェクトが作成されました。次に ISim を使用してビヘイビア シミュレーションを設定、起動します。

### ビヘイビア シミュレーション プロパティの設定

次の手順に従い、ISE でビヘイビア シミュレーションのプロパティを設定します。

1. [Design] パネルで [Simulation] ボタンをクリックします。  
シミュレーション タイプを示すドロップダウン リストが表示されます。
2. [Behavioral] を選択します。
3. チュートリアル デザインのベンチ ファイル `drp_demo_tb` をクリックします。  
[Processes] ペインにデザインで使用可能なシミュレーション プロセスが表示されます (図 2-10)。

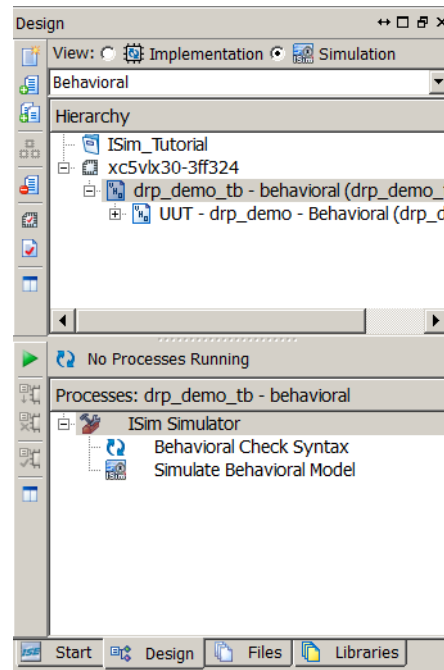


図 2-10 : ビヘイビア シミュレーション プロセス

4. [ISim Simulator] の下位に表示されている [Simulate Behavioral Model] を右クリックして [Process Properties] をクリックし、[Process Properties - ISim Properties] ダイアログ ボックスを表示します (図 2-11)。

このダイアログ ボックスでは、シミュレーション実行時間、波形データベース ファイルの位置、およびシミュレーションを実行するためのユーザー定義のシミュレーション コマンド ファイルなど、さまざまなシミュレーション プロパティを設定できます。このチュートリアルでは、特定時間のシミュレーションを実行する機能をディスエーブルにします。

5. [Run for Specified Time] をオフにして、[OK] をクリックします。

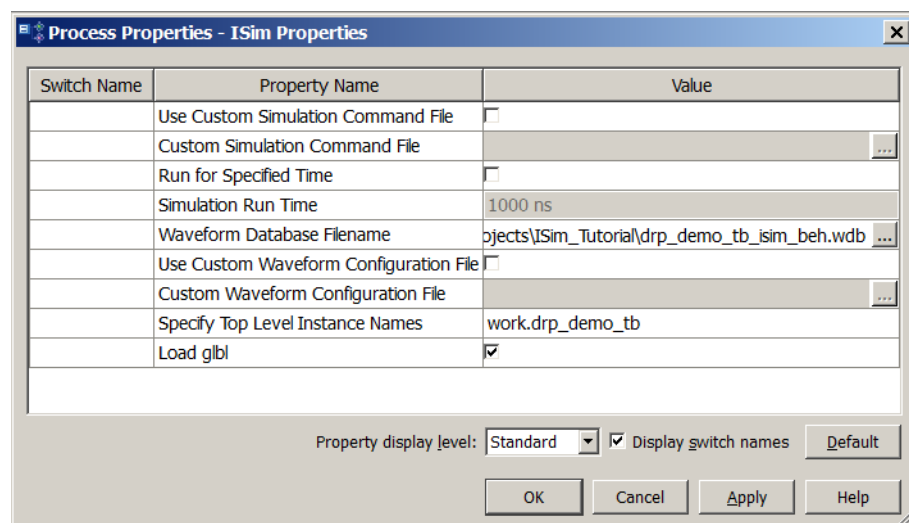


図 2-11 : [Process Properties - ISim Properties] ダイアログ ボックス

## ビヘイビア シミュレーションの起動

これで、ISim を起動してチュートリアル デザインのビヘイビア シミュレーションを実行する準備ができました。シミュレーションを起動するには、次を実行します。

[Processes] ペインで、[Simulate Behavioral Model] をダブルクリックします。

デザインが正しく解析、コンパイルされると、ISim インターフェイス (図 2-12) が表示されます。

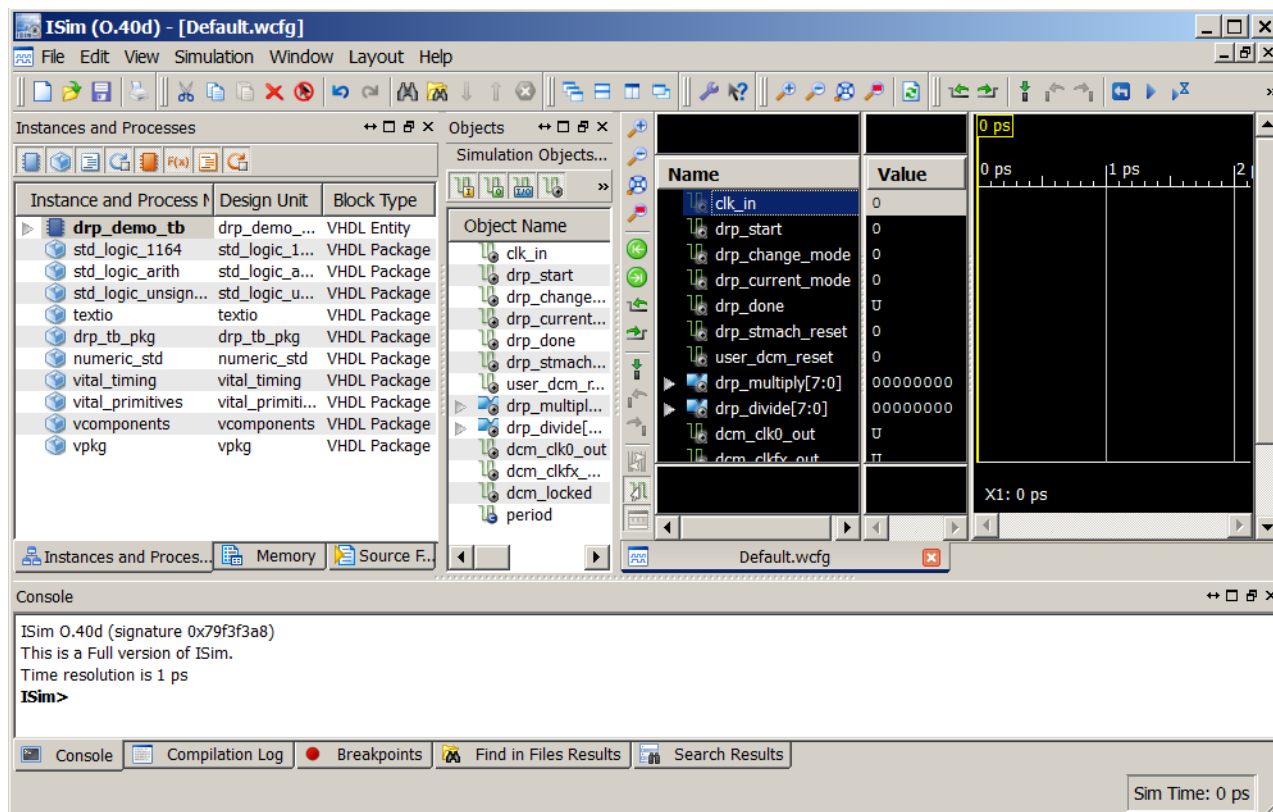


図 2-12：ISim グラフィカル ユーザー インターフェイス

## 次の操作

第 4 章「ISim グラフィカル ユーザー インターフェイスの使用」に進み、HDL デザインの解析およびデバッグに使用する ISim GUI 機能およびツールについて学びます。

# スタンドアロン ISim の実行

---

## ISim スタンドアロン フローの概要

ISim スタンドアロン フローでは、ISE® Project Navigator でプロジェクトを設定せずにデザインをシミュレーションできます。このフローでは、次を実行します。

- fuse を使用してシミュレーション実行ファイルを作成できるように、ISim プロジェクト ファイルを手動で作成します。
- fuse で生成したシミュレーション実行ファイルを実行し、ISim グラフィカル ユーザー インターフェイス (GUI) を起動します。

## はじめに

### 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、次のいずれかのソフトウェアをインストールする必要があります。

- ISE WebPACK™ 13.1
- ISE Design Suite 13.1 Edition (Logic、DSP、Embedded、System) のいずれか

ザイリンクス ソフトウェアのインストールに関する詳細は、[『ISE Design Suite 13 : 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 デザイン ファイルのインストール

このチュートリアルのデザイン ファイルは、ザイリンクス Web サイトの[チュートリアル ページ](#)から入手できます。

- Web サイトからチュートリアルのデザイン ZIP ファイルをダウンロードします。
- デザイン ファイルを書き込み/読み取り権があるディレクトリに解凍します。

チュートリアル デザイン ファイルの内容は、次のとおりです。

Table 3-1：チュートリアル デザイン ファイル

フォルダー	説明
sources	デザインの論理シミュレーションに必要な HDL ファイルが含まれています。
scripts	シミュレーションを実行するための未完成のスクリプト ファイルが含まれています。これらのスクリプト ファイルは、チュートリアルの進行に伴い完成させていきます。
completed	完成したスクリプト ファイル、シミュレーション ファイル、および波形コンフィギュレーション ファイルに加え、完成している ISE 13 プロジェクトのチュートリアル デザインが含まれており、進行中のデザイン ファイルと比較できます。

## デザインの概要

このチュートリアル デザインは、Virtex®-5 デジタル クロック マネージャー (DCM) のダイナミック リコンフィギュレーション機能を簡単に示したものです。

Virtex-5 DCM を使用すると、デザインで次の式に基づいて出力クロックが生成されます。

$$\text{出力クロック} = \text{入力クロック} * (\text{倍周率} / \text{分周率})$$

DCM のダイナミック リコンフィギュレーション ポート (DRP) を使用すると、倍周率および分周率を定義し直してさまざまな出力周波数を生成できます。

## 論理ブロック

チュートリアル デザインは、次の論理ブロックから構成されています。

### drp\_dcm (drp\_dcm.vhd)

内部フィードバック付き、周波数制御出力、デューティ サイクル調整、およびダイナミック リコンフィギュレーション機能を含んだ Virtex-5 DCM マクロです。

CLKFX\_OUT 出力では、次の式で定義されたクロックが供給されます。

$$\text{CLKFX\_OUT} = \text{CLKIN\_IN} * (\text{倍周率} / \text{分周率})$$

たとえば、100MHz 入力クロックを使用するとき、倍周率を 6、分周率を 5 にすると 120MHz の CLKFX\_OUT 出力クロックが生成されます。

DCM の DRP ポートを使用すると、倍周率 (M) および分周率 (D) パラメーターをダイナミックに定義し直して異なる CLKFX\_OUT 周波数を生成できます。このチュートリアルでは、これらの倍周率および分周率のパラメーターが 16 ビット幅の DI\_IN ポートから DCM にどのように渡されるかを示します。

$$\text{DI\_IN}[15:8] = M - 1$$

$$\text{DI\_IN}[7:0] = D - 1$$

たとえば、M/D が 6/5 のとき、DI\_IN は 0504h になります。

## drp\_stmach (drp\_stmach.vhd)

このモジュールでは、ダイナミック リコンフィギュレーション コントローラーについて説明します。DRP コントローラーでは、ダイナミック リコンフィギュレーション サイクルを実行するために DCM の DRP 信号をアサート、監視します。

ダイナミック リコンフィギュレーション サイクルは、drp\_start 信号がアサートされると開始します。この手順に従うと、DRP コントローラーで該当する DCM DRP ピンがアサートされ、ダイナミック リコンフィギュレーション サイクルが完了します。

drp\_done 信号は、ダイナミック リコンフィギュレーション サイクルが正しく完了したことを通知します。

## drp\_demo (drp\_demo.vhd)

チュートリアル デザインのトップ モジュールで、DCM マクロおよび DRP コントローラー モジュールが外部の I/O ポートに接続されています。

## drp\_demo\_tb (drp\_demo\_tb.vhd)

セルフチェック HDL テストベンチです。詳細は、「[デザイン セルフチェック テストベンチ](#)」を参照してください。

## デザイン セルフチェック テストベンチ

このデザインの機能性をテストできるように、セルフチェックテストベンチが提供されています ([sources] フォルダーに含まれている drp\_demo\_tb.vhd を参照)。セルフチェック テストベンチには、予期する結果とシミュレーションでのサンプル値を比較する検証ルーチンまたは関数が含まれています。このデザインで提供されるセルフチェック テストベンチでは、次が実行されます。

- デザインのシステム クロック (clk\_in) 向けに 100MHz 入力クロックを生成
- デザインの出力周波数をダイナミックに変更するため、4 つの異なるテストを実行。各テストでは、drp\_start 信号を使用して DRP サイクルが開始され、出力クロックに個別の周波数が設定されます。次の表では、各テストでの理想的な出力周波数および倍周率/分周率パラメーターが表示されています。

表 3-1：各テストで使用される理想的な出力周波数および倍周率/分周率パラメーター

テスト	周波数 (MHz)	周期 (ps)	倍周率 (M)	分周率 (D)
1	75	13,332	3	4
2	120	8,332	6	5
3	250	4000	5	2
4	400	2,500	4	1

- 各テストでは、テストベンチで予期されるクロック周期とシミュレーション中に計測されたクロック周期が比較されます。比較結果に基づいて、テストが正常に完了したかまたはエラーが発生したを示すメッセージがシミュレーターで表示されます。
- シミュレーションの完了時に生成されるサマリ レポートには、正常に完了したテストとエラーが発生したテストの両方を含むリストが含まれています。

このデザインの機能の詳細は、デザインのソースに含まれるインライン コメントを参照してください。

## シミュレーションの準備

ISim スタンドアロン フローでは、ISE Project Navigator でプロジェクトを設定せずにデザインをシミュレーションできます。このフローでは、fuse でシミュレーション実行ファイルを作成するのに使用する ISim プロジェクト ファイルを手動で作成します。この手順に従うと、シミュレーション実行ファイルを実行することで ISim グラフィカル ユーザー インターフェイス (GUI) を起動できます。

### ISim プロジェクト ファイルの作成

次に、通常の ISim ファイルの構文を示します。

```
verilog|vhdl <library_name> {<file_name_1>.v|.vhd}
```

説明：

- verilog|vhdl：ソース ファイルが Verilog または VHDL ファイルであることを示します。verilog または vhdl を含めます。
- <library\_name>：指定行のソースがコンパイルされるライブラリを指定します。デフォルト ライブラリは work です。
- <file\_name>：ライブラリに関連するソース ファイルを指定します。

**メモ：** Verilog ソース ファイルは 1 行に複数指定できますが、VHDL ソース ファイルは 1 つのみしか指定できません。

次の手順に従い、チュートリアル デザインの ISim プロジェクト ファイルを構築します。

1. ダウンロードしたファイルの [scripts] フォルダを参照します。
2. テキスト エディターで simulate\_isim.prj プロジェクト ファイルを開きます。  
この時点では、プロジェクト ファイルは未完成です。
3. 前述の構文ガイドラインに従い、含まれていないソースをリストします。

含まれていないソース：

- drp\_dcm.vhd：VHDL ソース ファイル。work ライブラリにコンパイルする必要があります。
- drp\_tb\_pkg.vhd：VHDL パッケージ ファイル。drp\_tb\_lib ライブラリにコンパイルする必要があります。

**メモ：** ソース ファイルは、依存順にリストする必要はありません。fuse では依存順が自動的に解決され、正しい順序でファイルが処理されます。

チュートリアル ファイルの [completed] フォルダには完成したプロジェクト ファイルが含まれているので、作成したプロジェクト ファイルと比較できます。

4. ファイルを保存してから閉じます。

### シミュレーション実行ファイルの構築

この手順では、fuse で作成したプロジェクト ファイルが使用され、デザインのすべてのソースが解析、コンパイル、リンクされます。これらの手順に従うと、ISim GUI でシミュレーションを実行できるシミュレーション実行ファイルが作成されます。



## fuse の使用

次に、通常の fuse 構文を示します。

```
fuse -incremental -prj <project file> -o <simulation executable>
<library.top_unit>
```

説明：

- **-incremental** : 最後にコンパイルされてから変更されたファイルのみを再コンパイルします。
- **-prj** : 入力として使用する ISim プロジェクト ファイルを指定します。
- **-o** : シミュレーション実行出力ファイルの名前を指定します。
- **<library.top\_unit>** : 最上位デザイン ユニットを指定します。

次の手順に従い、fuse を使用してチュートリアル デザインの解析、コンパイル、およびエラボレーションを実行します。

1. ダウンロードしたファイルの [scripts] フォルダを参照します。
2. テキスト エディターで fuse\_batch.bat バッチ ファイルを開きます。
3. この fuse コマンドは未完成の状態です。上記の構文情報を使用して、次のオプションを含むようにコマンド行を編集します。
  - a. インクリメンタル コンパイルを使用
  - b. simulate\_isim.prj をプロジェクト ファイルとして使用
  - c. simulate\_isim.exe をシミュレーション ファイルとして使用
  - d. work.drp\_demo\_tb をシミュレーションの最上位デザイン ユニットとして使用
4. バッチ ファイルを保存してから閉じます。
5. ISE Design Suite のコマンド プロンプトで fuse\_batch.bat ファイルまで移動して、fuse を実行します。

**メモ** : ISE Design Suite のコマンド プロンプトを開くには、[スタート] → [プログラム] → [Xilinx ISE Design Suite] → [アクセサリ] で [ISE Design Suite コマンド プロンプト] をクリックします。

fuse でソースのコンパイル、デザイン ユニットのエラボレーション、オブジェクトのリンクが完了すると、シミュレーション実行ファイル (simulate\_isim.exe) が [scripts] フォルダに含められます。

[completed] フォルダには完成した fuse バッチ ファイルが含まれているので、作成したファイルと比較できます。

## デザインのシミュレーション

「[シミュレーション実行ファイルの構築](#)」セクションで fuse ツールを使用して生成したシミュレーション実行ファイルを実行して ISim GUI を起動します。この手順を完了すると、ISim GUI で詳細にデザインを調べることができます。

### シミュレーション実行ファイルの実行

次に、シミュレーション実行ファイルを起動するときに使用する通常の構文を示します。

```
Simulation_executable -gui -view <wave_configuration_file> -wdb
<waveform_database_file>
```

説明：

- `-gui` : ISim を GUI モードで起動します。
- `-view` : ISim GUI で指定の波形ファイルを開きます。
- `-wdb` : シミュレーション データベース出力ファイルの名前を指定します。

次の手順に従い、シミュレーションを実行します。

1. ダウンロードしたファイルの [scripts] フォルダを参照します。
2. テキスト エディターで `simulate_isim.bat` バッチ ファイルを開きます。バッチ ファイルは意図的に空白になっています。
3. 上記の構文情報を使用して、次の設定を含むようにバッチ ファイルを編集します。
  - a. シミュレーション実行ファイル名 : `simulate_isim.exe`
  - b. GUI モードで実行
  - c. シミュレーション データベース出力名を `simulate_isim.wdb` に設定

**メモ** : 波形コンフィギュレーション ファイルは、このチュートリアル ファイルに含まれていません。このファイルは生成されます。
4. ファイルを保存してから閉じます。
5. ISE Design Suite のコマンド プロンプトで `simulate_isim.bat` ファイルまで移動して、シミュレーターを実行します。

ISim GUI が開き、デザインが読み込まれます。シミュレーション時間は、実行時間を指定するまで 0ns になっています。

[completed] フォルダには完成した `simulate_isim.bat` バッチ ファイルが含まれているので、作成したファイルと比較できます。

## 次の操作

第 4 章「[ISim グラフィカル ユーザー インターフェイスの使用](#)」に進み、HDL デザインの解析およびデバッグに使用する ISim GUI 機能およびツールについて学びます。

## ISim グラフィカル ユーザー インターフェイスの使用

### ISim グラフィック ユーザー インターフェイスの概要

ISim グラフィカル ユーザー インターフェイス (GUI) には、波形ウィンドウ、ツールバー、パネル、およびステータス バーが含まれています。メイン ウィンドウでは、デザインでシミュレーションが可能な箇所の表示、波形ウィンドウでの信号の追加および表示、ISim コマンドを使用したシミュレーションの実行、デザインの検証、およびデバッグを実行できます。

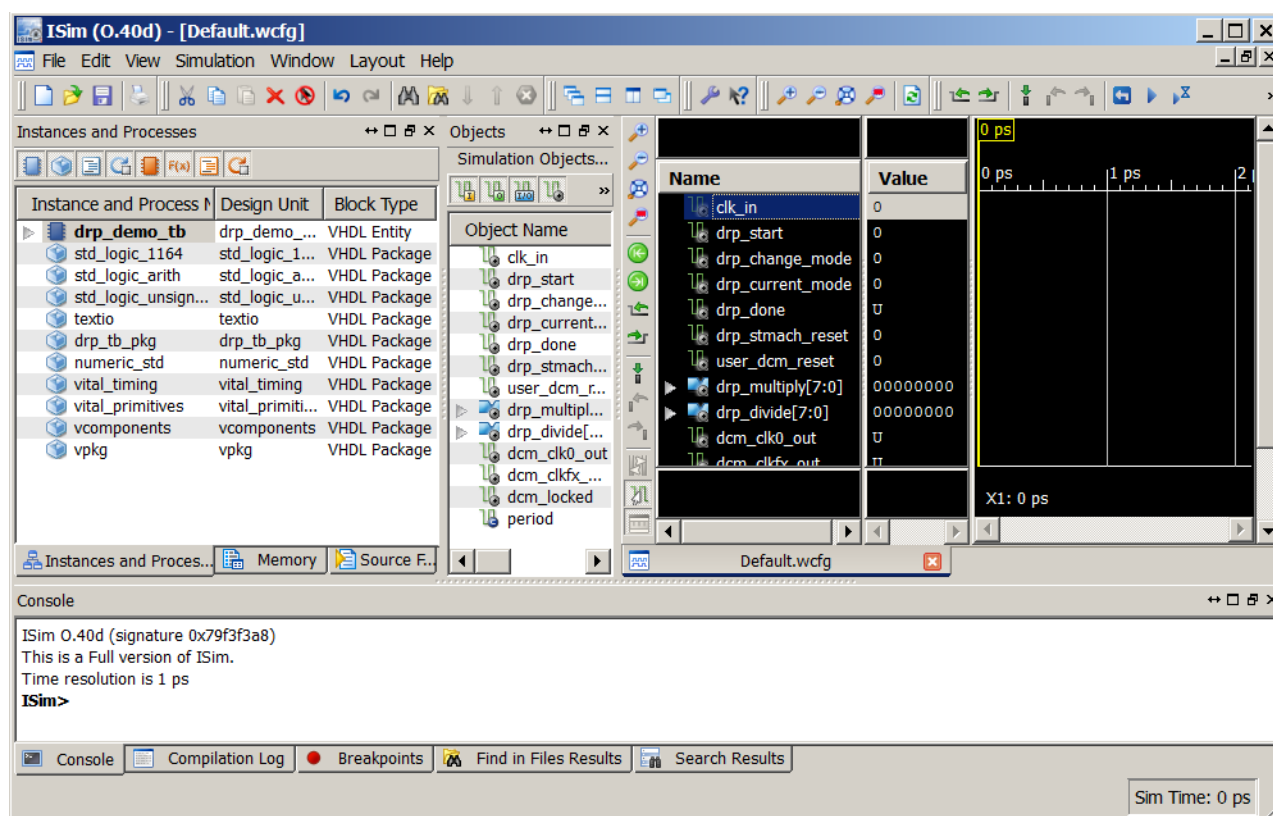


図 4-1 : ISim グラフィカル ユーザー インターフェイス

## ユーザー インターフェイスの使用

### 主要ツールバー

ISim のメイン ウィンドウで使用可能なツールバーは、さまざまな機能別ツールバーで構成されています。これらのツールバーからは、メニューからもアクセス可能なよく使用するコマンドにアクセスできます。

メイン ウィンドウのツールバー アイコンは、ユーザー インターフェイスの上部に配置されています。

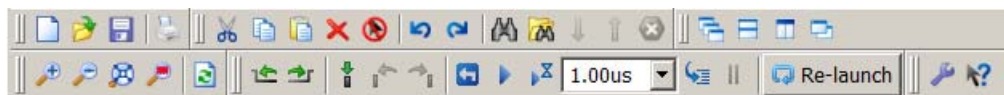


図 4-2：主要ツールバー

### [Instances and Processes] パネル

[Instances and Processes] パネルには、波形ウィンドウの波形コンフィギュレーションと関連するブロック (インスタンスおよびプロセス) の階層が表示されます。インスタンス化してエラレートされたエンティティ / モジュールは、ポート、信号、およびその他のエンティティ / モジュールなどのエンティティ コンポーネントと共にツリー構造で表示されます。

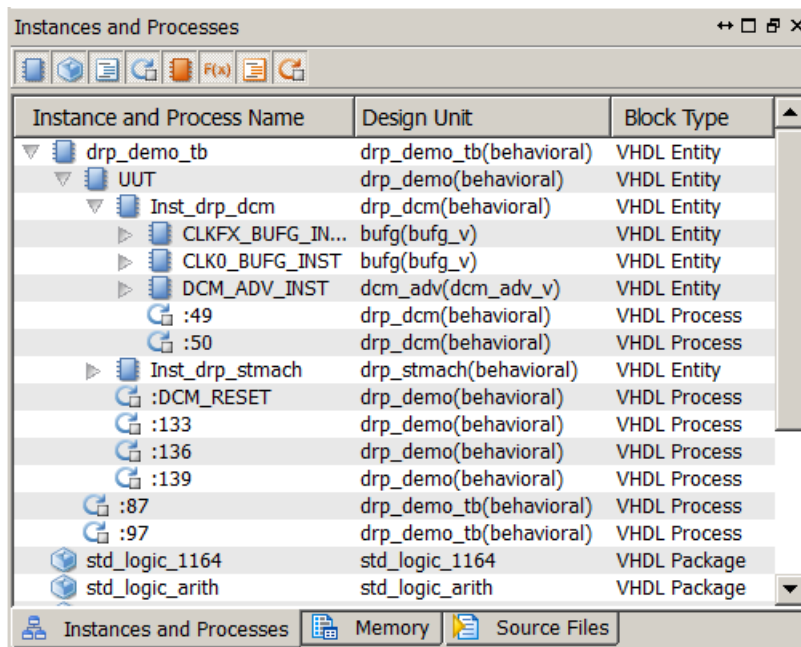


図 4-3：[Instances and Processes] パネル

## [Source Files] パネル

[Source Files] パネルには、デザインに関連するファイルのリストが表示されます。このリストは、GUI のバックグラウンドで実行されるデザインの解析およびエラボレーション中に **fuse** コマンドにより渡されます。HDL ソース ファイルは、ソース コードを読み取り専用で開くことができます。

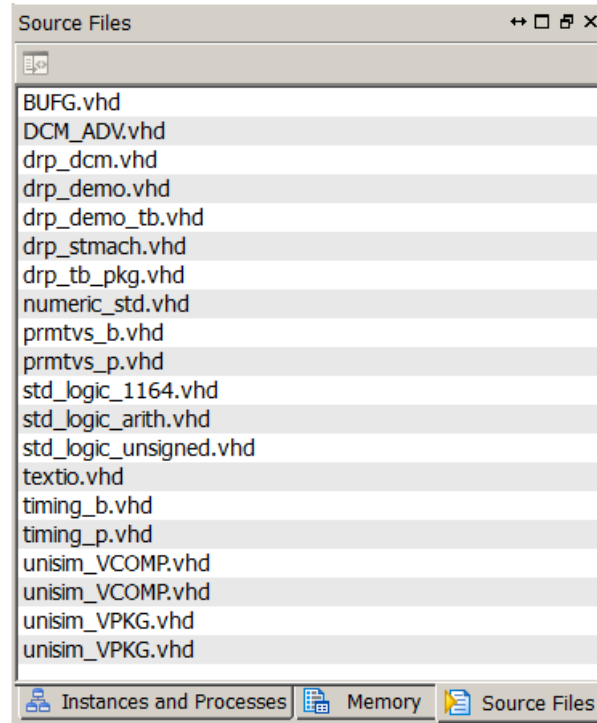


図 4-4 : [Source Files] パネル

## [Objects] パネル

[Objects] パネルでは、[Instances and Processes] パネルで選択したインスタンスおよびプロセスに関連するポートおよび信号がすべて表示されます。

このパネル上部の [Simulation Objects] には [Instances and Processes] パネルで選択されているインスタンス/プロセスが表示され、そのオブジェクトおよび値が [Objects] パネルに表示されます。

次に、[Objects] パネルの表の各列について説明します。

- [Object Name] : 信号名とそのタイプを示すシンボルが表示されます。
- [Value] : [Sync Time] ボタンに基づき現在のシミュレーション時間またはメイン カーソルの場所の信号の値を表示します。
- [Data Type] : シミュレーション オブジェクト、ロジック、またはアレイのデータ タイプを表示します。

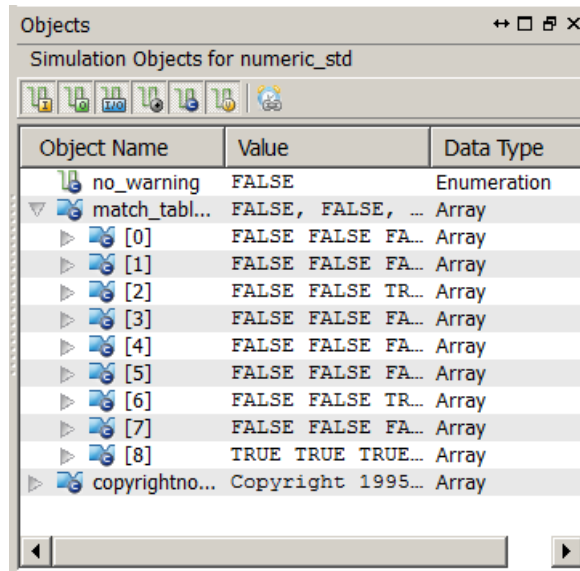


図 4-5 : [Objects] パネル

## 波形ウィンドウ

波形ウィンドウには、信号、バス、およびこれらの波形が表示されます。波形ウィンドウの各タブには、信号およびバスのリストとそのプロパティ、仕切り、カーソル、またはマーカーなどの波形オブジェクトから構成される波形コンフィギュレーションが表示されます。

GUI では波形コンフィギュレーションの信号およびバスがシミュレーション中にトレースされるため、波形コンフィギュレーションはシミュレーションを実行してシミュレーション結果を調べるときに使用されます。デザインおよびシミュレーション データはデータベースに含まれるので、波形コンフィギュレーションに信号を追加したり、またそこから信号を削除してもシミュレーション データは影響を受けません。

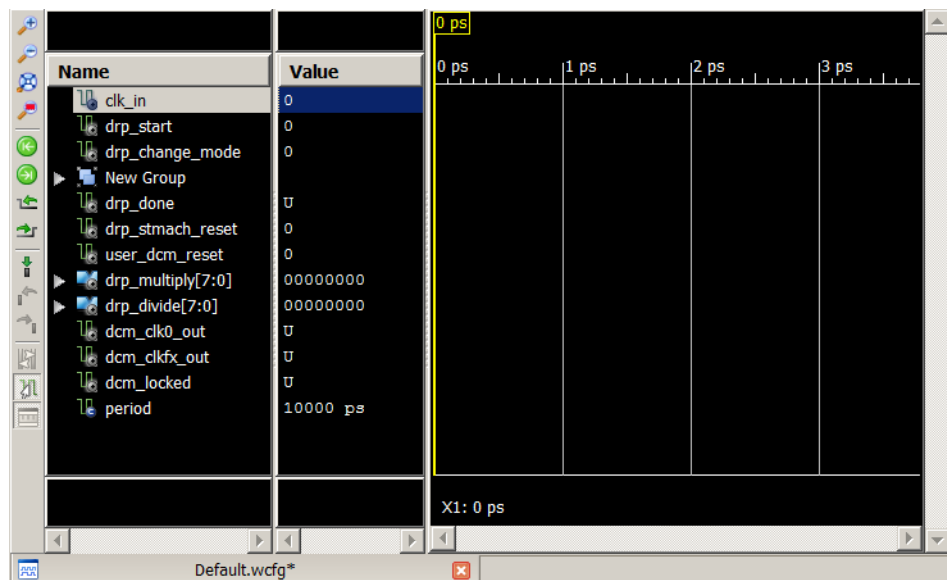


図 4-6 : 波形ウィンドウ

## テキスト エディター ウィンドウ

テキスト エディター ウィンドウでは、シミュレーションで使用する HDL ソース ファイルに簡単にアクセスできます。次の基本的な手順を実行できます。

- HDL ソース ファイルを表示して編集
- ソース ファイルにブレークポイントを設定してデバッグ
- ソース コードを 1 行ずつ進める

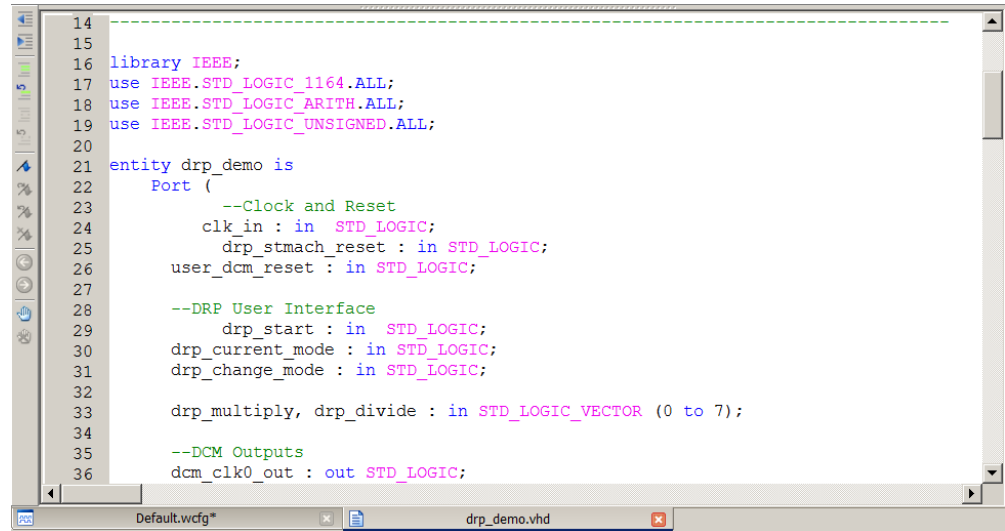


図 4-7 : テキスト エディター ウィンドウ

## [Breakpoints] パネル

[Breakpoints] パネルには、デザインに現在設定されているブレークポイントすべてがリスト表示されます。このリストでは、ソース ファイルに設定されている各ブレークポイントに対して、ファイルの保存場所、ファイル名、および行番号が表示されます。[Breakpoints] パネルのツールバーや文脈依存メニューを使用して、選択したブレークポイントまたはすべてのブレークポイントを削除したり、ソース コードに移動できます。

詳細は、[『ISim ユーザー ガイド』](#)の第 4 章「デザインのデバッグ」を参照してください。

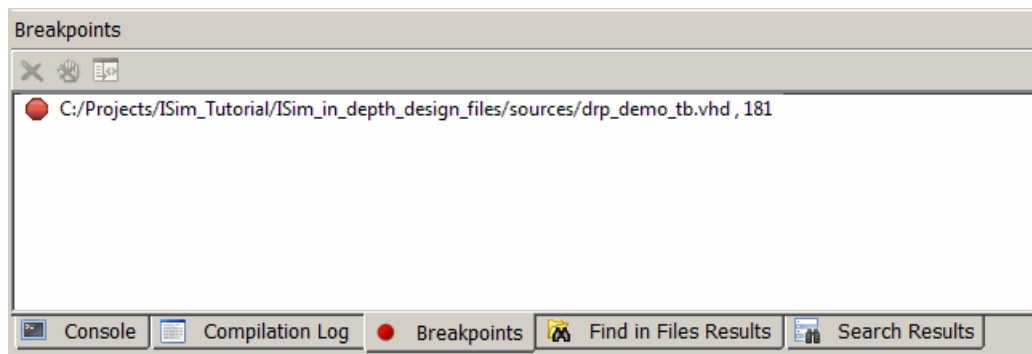


図 4-8 : [Breakpoints] パネル

## [Console] パネル

[Console] パネルでは、ISim で生成されるメッセージ ログを確認し、コマンド プロンプトで標準 Tcl コマンドおよび ISim 特有のコマンドを入力できます。

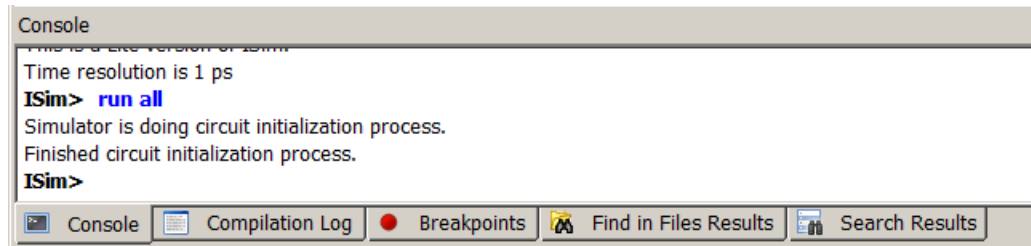


図 4-9 : [Console] パネル

## デザインの検証

このセクションでは、手順に従いチュートリアル デザインの論理動作をさらに解析します。次の作業が含まれます。

- 信号を波形ウィンドウで使用したり、[Console] パネルに表示されているテストベンチのメッセージを確認しながら、シミュレーションを実行、再実行してデザインの機能を確認します。
- テストベンチの信号およびその他のデザイン ユニットを波形ウィンドウに追加し、これらのステータスを監視します。
- 波形ウィンドウの信号を識別しやすくするよう、グループや仕切りを追加します。
- 信号および波形ウィンドウのプロパティを変更し、波形ウィンドウで信号を確認しやすくします。
- マーカーおよびカーソルを使用してシミュレーションでの主なイベントをハイライトしたり、ズーム機能や時間計測機能を使用します。
- 複数の波形ウィンドウ コンフィギュレーションを使用して、1 つのシミュレーション セッションで複数の信号を確認しやすくします。

## 信号の追加

**メモ :** 第 2 章「ISE Project Navigator からの ISim の実行」を完了している場合は、この手順を飛ばして進んでください。テストベンチのシミュレーション オブジェクトすべてが波形ウィンドウに追加されています。

特定時間シミュレーションを実行する前に、信号のステータスを観察できるよう波形ウィンドウに信号を追加する必要があります。

テストベンチのすべてのシミュレーション オブジェクトを波形ウィンドウに追加します。シミュレーション オブジェクトには、次が含まれます。

- 入力クロック (clk\_in) : テストベンチで生成される 100MHz クロックで、デジタル クロック マネージャー (DCM) への入力クロックです。
- ダイナミック リコンフィギュレーション ポート (DRP) (drp\_\*) : DCM の DRP 機能に関連する信号です。テストベンチではこれらの信号がアサート、監視され、DCM の DRP 機能が確認、制御されます。
- DCM 出力信号 (dcm\_\*) : DCM の出力クロックです。



これらの信号を波形ウィンドウに追加するには、次の手順に従います。

1. [Instances and Processes] パネルで `drp_demo_tb` インスタンス ユニットの右クリックします。
2. [Add to Wave Window] をクリックします

これで、`drp_demo_tb` テストベンチのシミュレーション オブジェクトすべてが波形ウィンドウに表示されます (図 4-10)。

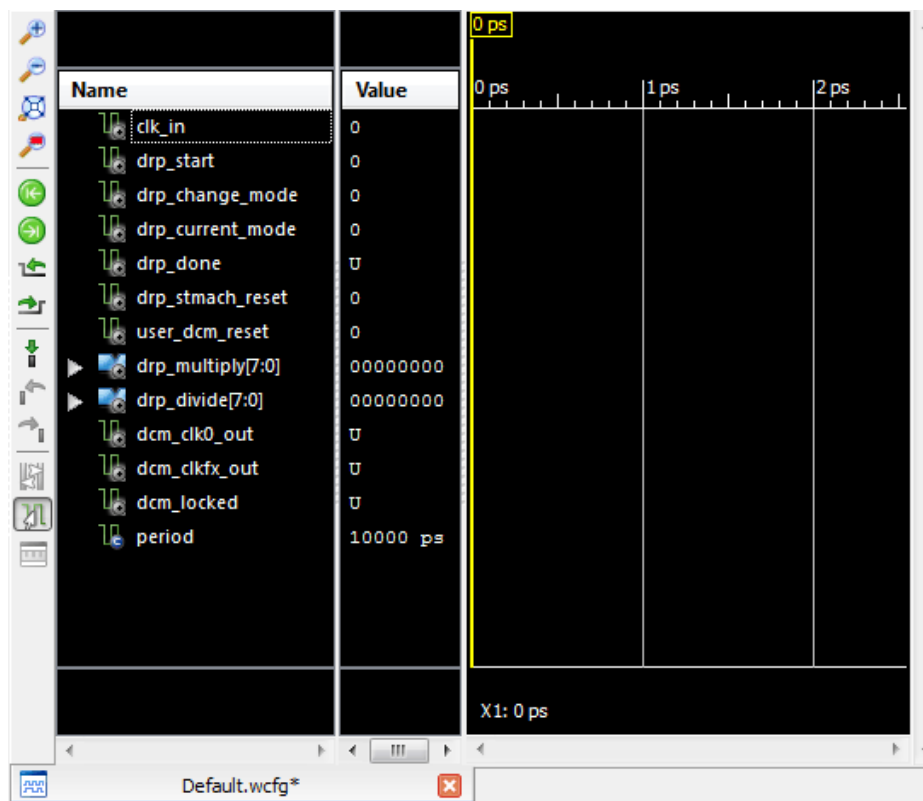



図 4-10 : 波形ウィンドウ

## 特定時間のシミュレーションの実行

特定時間だけシミュレーションを実行できます。シミュレーションを 5 マイクロ秒 (us) 間実行します。

1. ISim ツールバーにある [Run for the time specified on the toolbar] に「5us」と入力して [Run] ボタン  をクリックします。

**メモ：** また、Tcl プロンプトに「run 5 us」と入力し、Enter キーを押しても実行できます。

波形ウィンドウでは、5us シミュレーション時間までの信号のトレースが表示されます (図 4-11)。

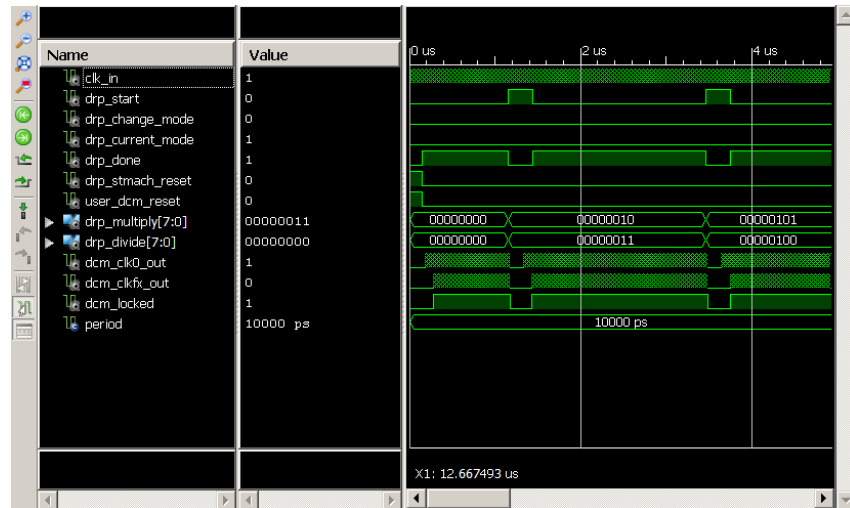



図 4-11：波形ウィンドウ

2. 波形ウィンドウでスペクトラム全体を表示する場合は、[View] → [Zoon] → [To Full View] をクリックするか、[Zoom Full View] ボタン  をクリックします。
3. 水平方向または垂直方向のスクロール バーを使用して、波形コンフィギュレーション全体を表示できます。
4. シミュレーション中にテストベンチからのアサートがあります。[Console] パネルでテストベンチから出力されたメッセージを確認します (図 4-12)。

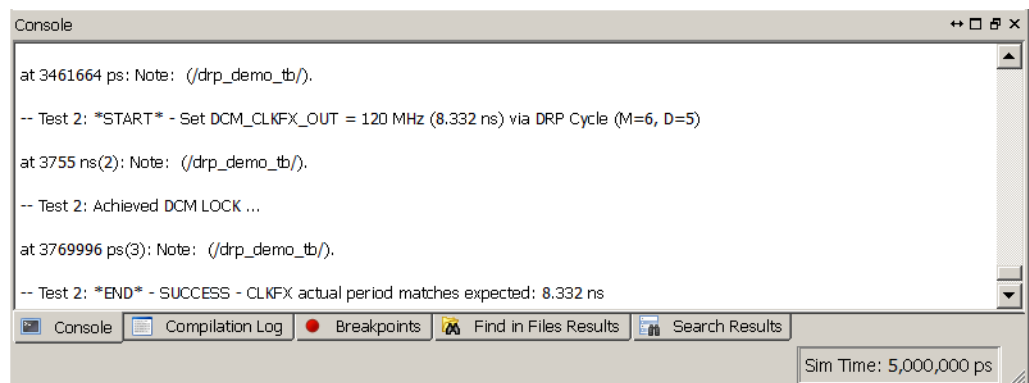



図 4-12：[Console] パネル

## シミュレーションの再スタート

1. 作業を進める前に、[Restart] ボタン  をクリックしてシミュレーションを再スタートして波形ウィンドウをクリアにし、シミュレーション時間を 0 ピコ秒 (9ps) に設定します。

メモ：または、Tel プロンプトに「restart」と入力してもシミュレーションを再スタートできます。

波形ウィンドウは、[図 4-13](#) のように表示されます。

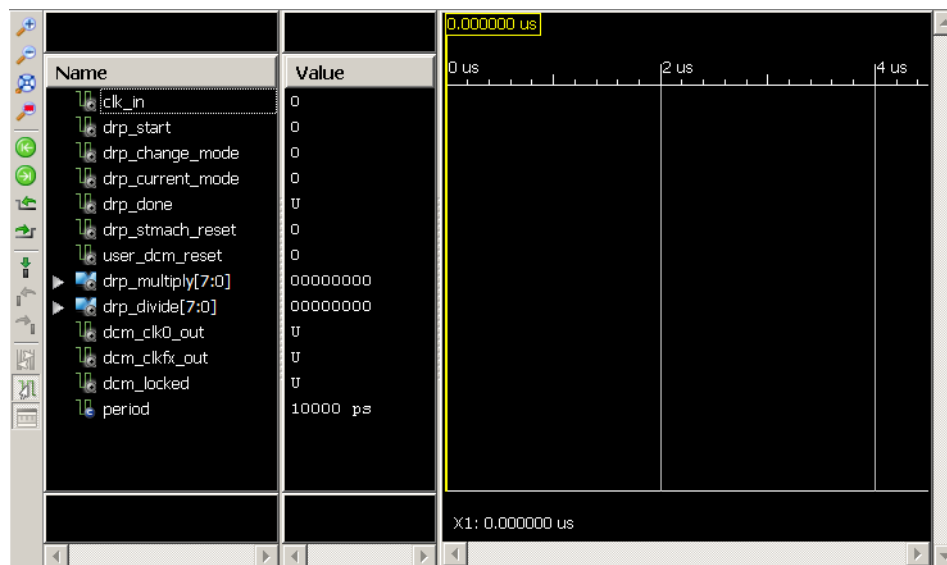


図 4-13：波形ウィンドウ

次のセクションでは、波形ウィンドウの仕切り、グループ、カーソル、およびマーカーなどの機能を使用して、チュートリアル デザインのシミュレーションを詳細に解析します。

## グループの追加

このデザインの機能をさらに解析できるよう、次の手順に従って別のデザイン ユニットの信号を追加します。ただし、波形ウィンドウに信号を追加すると、波形ウィンドウのサイズが同じビューですべての信号を表示するのに十分なサイズではなくなります。すべての信号を確認するには、波形ウィンドウの垂直方向スクロール バーを使用する必要があるため、確認作業が面倒になります。

このような状況は、信号をグループに含めることで解決できます。グループを使用すると、同じ目的の複数の信号をまとめて表示/非表示できます。波形コンフィギュレーションの信号をグループ化するには、次の手順に従います。

1. 波形ウィンドウで、**Ctrl** キーを押したままにして、**drp\_** で始まる **drp\_demo\_tb** デザイン ユニットの含まれている信号すべてを選択します。
2. 選択されている信号を右クリックして **[New Group]** をクリックします。
3. 新しいグループに名前を入力します。ここでは、「**DRP Test Signals**」という名前を付けます。
4. 階層が閉じた状態のグループが波形ウィンドウに作成されます。グループの階層を展開するには、グループ名左横のプラス記号をクリックします。
5. 手順 1 ～ 3 に従って、**dcm\_** で始まる **drp\_demo\_tb** デザイン ユニットの含まれているすべての信号を含めるグループを 1 つ作成します。「**DCM Test Signals**」という名前を付けます。
6. 作成したグループをすべて展開表示します。波形ウィンドウは、[図 4-14](#) のように表示されます。

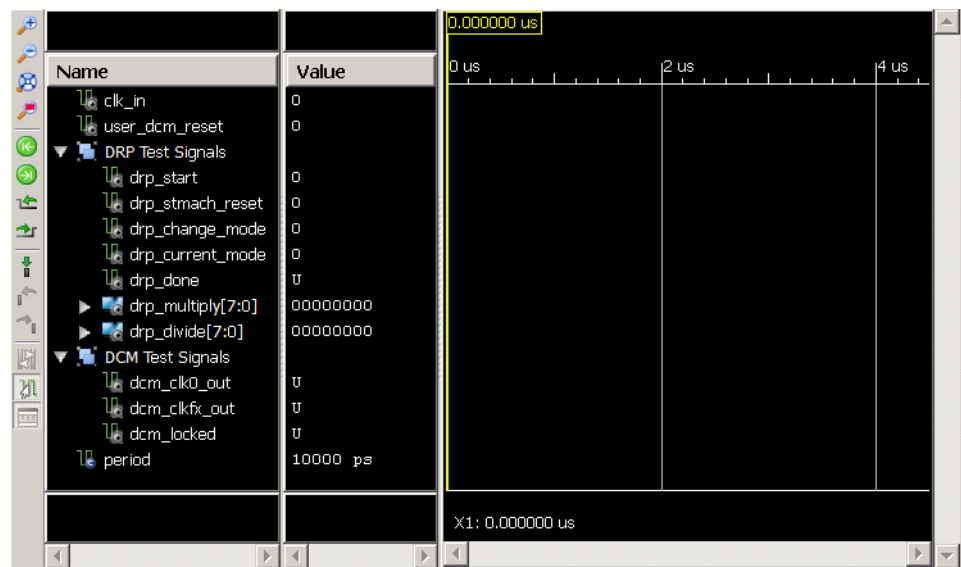


図 4-14：グループの追加

信号グループが [図 4-14](#) と一致しない場合は、次の方法で修正できます。

- 無関係の信号を含めた場合は、カットアンドペーストで信号をメイン リストに移動します。
- グループを作成したがメイン リストの信号を含め忘れた場合は、ドラッグアンドドロップで信号をグループに移動します。これで、信号がグループに含められます。
- **[Edit]** → **[Undo]** をクリックすると、グループ化を解除できます。
- グループを解除することでやり直すことができます。この場合は、グループを右クリックして **[Ungroup]** をクリックします。

## 仕切りの追加

このデザインの機能をさらに解析できるよう、次の手順に従って別のデザイン ユニットの信号を追加します。どのデザイン ユニットに属する信号かをわかりやすく表示できるよう、デザイン ユニットごとに信号を分ける仕切りを追加できます。

仕切りを波形ウィンドウに追加するには、次の手順に従います。

1. 波形ウィンドウの任意の場所で右クリックして、[New Divider] をクリックします。
2. 仕切り名を入力します。
3. 上記の手順に従い、次の 3 つ名前の仕切りを追加します。
  - TEST BENCH
  - DCM
  - DRP CONTROLLER
4. [TEST BENCH] をクリックしてリストの一番上までドラッグします。
5. その他の仕切りは、リストの最後に移動します。

メモ：仕切り名は、ダブルクリックするか、F2 キーを押すといつでも変更可能です。

波形ウィンドウは、図 4-15 のように表示されます。



図 4-15：仕切りの追加

## サブモジュールからの信号の追加

次に、サブモジュールとテストベンチのテスト信号間の通信を調べるために、インスタンス化されている DCM モジュール (Inst\_drp\_dcm) および DRP コントローラー モジュール (Inst\_drp\_statmach) の信号を追加します。

次の手順に従い、必要な信号を追加します。

1. [Instances and Processes] パネルで、図 4-16 に示すように drp\_demo\_tb の階層を展開表示します。
2. Inst\_drp\_dcm エンティティをクリックします。

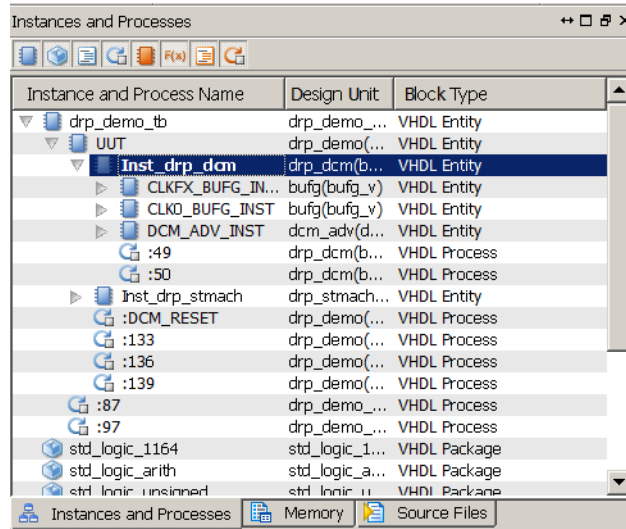


図 4-16 : [Instances and Processes] パネル

3. 現在ハイライトされているデザイン ユニットに関連するシミュレーション オブジェクトが [Objects] パネルに表示されます (図 4-17)。

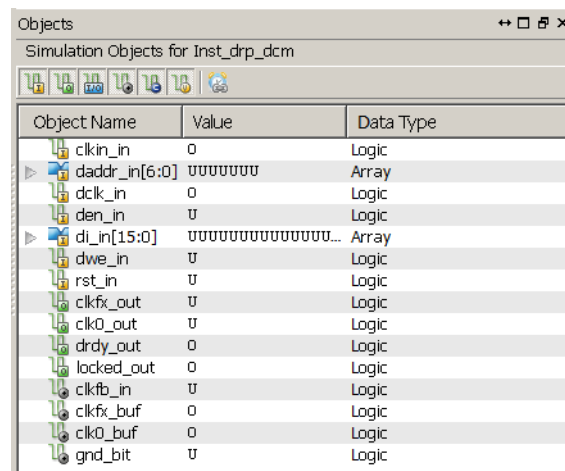


図 4-17 : [Objects] パネル

次に、Inst\_drp\_dcm デザイン ユニットのインスタンス化のすべての入力ポートおよび出力ポートと内部信号を波形ウィンドウに追加します。

デフォルトでは、変数、定数などのすべての種類のシミュレーション オブジェクトが [Objects] パネルに表示されます。オブジェクトのタイプは、図 4-18 のように示されます。

### 信号

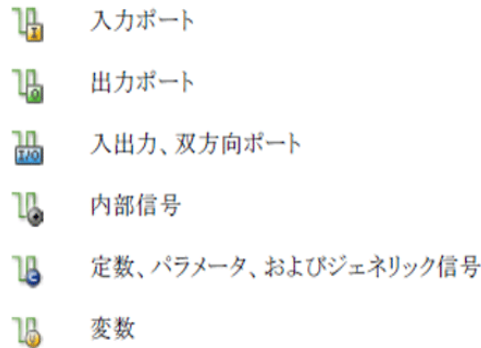


図 4-18 : 信号およびアイコン

このパネルでは、表示されるシミュレーション オブジェクトの種類にフィルターを設定できます。[Objects] パネルのツールバーで入力、出力、双方向、内部、定数、および変数ごとにフィルターして表示します。ボタンを切り替えるごとに表示されるオブジェクトの種類が切り替わります。



図 4-19 : [Objects] パネルのツールバー ボタン

1. [Instances and Processes] パネルで `Inst_drp_dcm` デザイン ユニットが選択されているときにこれらのツールバー ボタンを使用すると、入力および出力ポートと内部信号のみを表示できます。
2. [Objects] パネルで表示されているすべてのオブジェクトを選択し、波形ウィンドウの [DCM] 仕切りの下にドラッグアンドドロップします。

**メモ：**また、ISim Tcl プロンプトで `wave add Tcl` コマンドを使用しても、波形ウィンドウにこれらの信号を追加できます。次に例を示します。

```
wave add /drp_demo_tb/uut/inst_drp_dcm
```

3. 手順 1 ～ 3 を繰り返して [DRP CONTROLLER] 仕切りの下にあるインスタンス済みデザイン ユニット `Inst_drp_stmach` の入力ポートおよび出力ポートを追加します。
4. 追加した信号にグループを作成します。各信号セットを「Inputs」、「Outputs」、および「Internal」のグループに分けます。

波形ウィンドウは、図 4-20 のように表示されます (グループの階層は非展開の状態)。

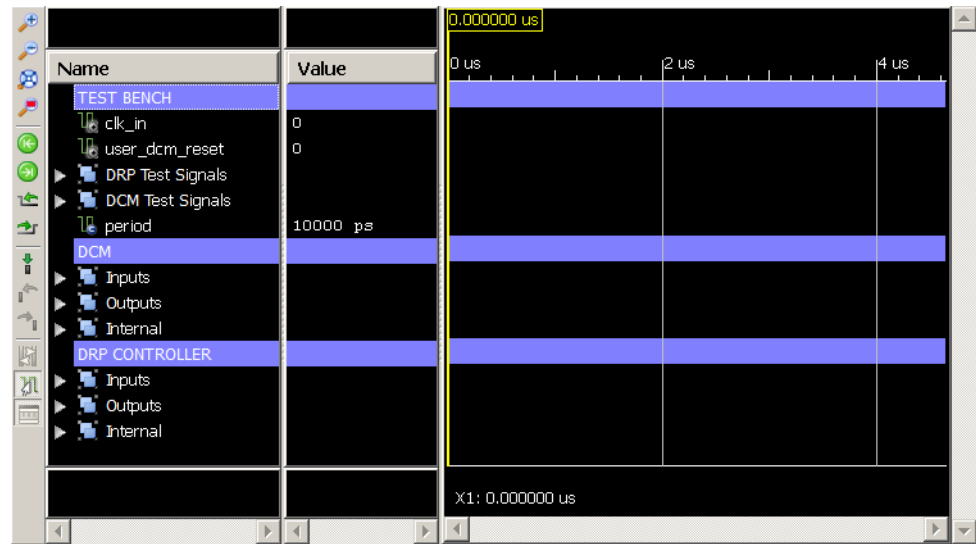


図 4-20：波形ウィンドウの設定

## 信号および波形ウィンドウのプロパティの変更

次に、波形ウィンドウに表示されている信号の一部のプロパティを変更し、ビヘイビア シミュレーションの表示をわかりやすくします。

### 信号名の表示形式の変更

ISim ではデフォルトで階層リファレンスを省いた短い名前で信号が波形に追加されます。一部の信号では、属しているモジュールを知ることが重要です。

次の手順に従って、`drp_multiply` および `drp_divide` バス信号の信号名の表示形式を変更します。

1. 波形ウィンドウで **DRP Test Signals** グループに含まれている `drp_multiply` および `drp_divide` 信号を **Ctrl** キーを押しながら選択します。
2. 右クリックして **[Name] → [Long]** をクリックします。

これで、信号名の表示形式が変更されます。

### 信号名の基数変更

一部の信号は 2 進数より 16 進数で表示されたほうが認識しやすくなるものがあります。たとえば、信号 `drp_multiply` および `drp_divide` がその例です。

次の手順に従って、これらの信号の基数オプションを変更します。

1. 波形ウィンドウで `drp_demo_tb/drp_multiply` および `drp_demo_tb/drp_divide` 信号を選択します。
2. 右クリックして **[Radix] → [Hexadecimal]** をクリックします。

### 信号の表示色の変更

ISim では、波形ウィンドウに表示される信号の色を変更し、類似する信号をすぐに見分けることができます。



次の手順に従って、drp\_multiply および drp\_divide 信号の表示色を変更します。

1. 波形ウィンドウで [Name] 列にリストされている信号名を右クリックします。
2. [Signal Color] をクリックしてカラー パレットから色を選択するか、(...) ボタンをクリックしてカスタム カラーを選択します。

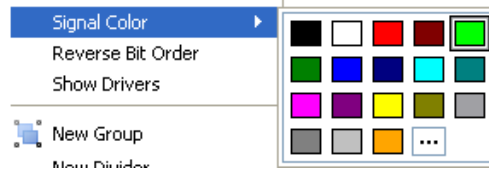



図 4-21：信号の表示色の変更

メモ：また、[Divider Color] オプションを使用すると、波形ウィンドウで作成した仕切りの色を変更できます。

### 波形ウィンドウのフロート表示

画面の解像度設定によっては、波形ウィンドウにインターフェイスで一度で表示できる以上の信号が含まれている可能性があります。波形ウィンドウをフロートすると、表示エリアを拡張できます。フロートさせると、波形の内容のみを含む新しいウィンドウが開きます。

ウィンドウをフロートするには、[Float Window] ボタン  をクリックします。

これで、波形ウィンドウでの変更が終了しました。波形ウィンドウは、テストベンチ グループが展開表示されているとき、[図 4-22](#) のように表示されます。

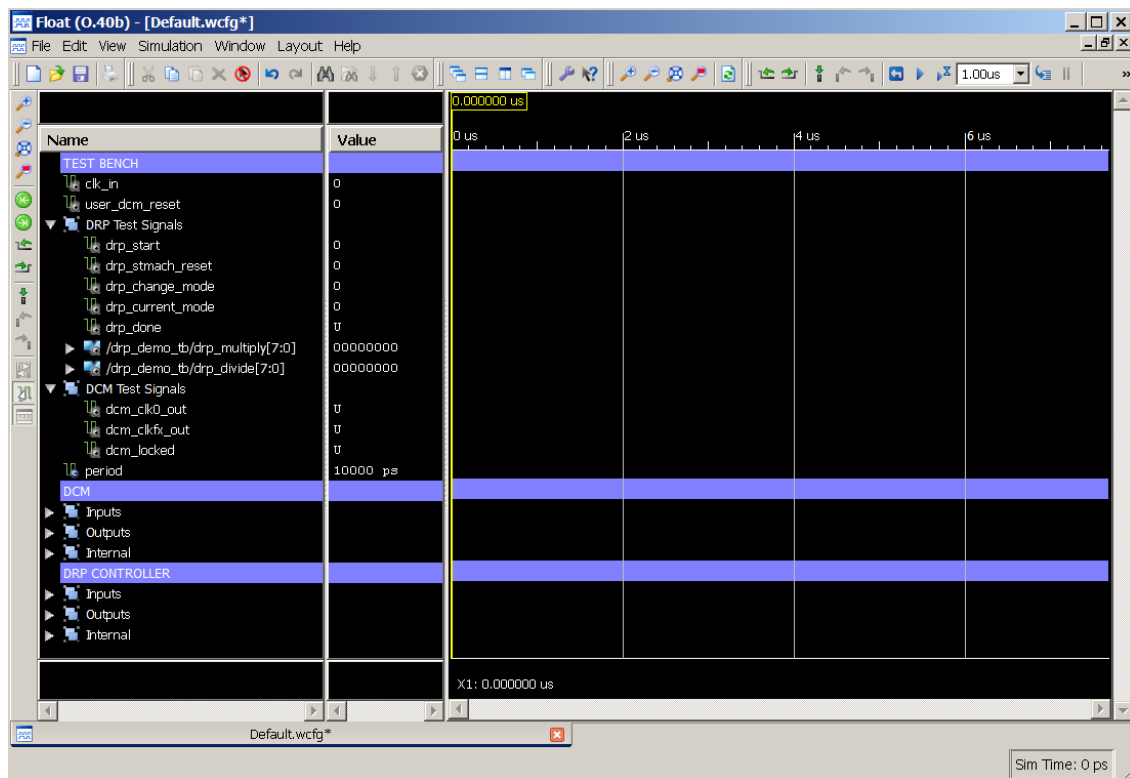


図 4-22：完全に設定されたフロート波形ウィンドウ

## 波形ウィンドウの設定の保存

現在の波形ウィンドウ (波形コンフィギュレーション) を保存して、今後の ISim シミュレーションセッションで使用できます。


波形コンフィギュレーションを保存するには、次の手順に従います。

1. 現在の波形コンフィギュレーションに名前を付けるには、[File] → [Save As] をクリックします (図 4-23)。
2. 作業中の波形コンフィギュレーションを「tutorial\_1.wcfg」という名前で保存します。

これで、波形コンフィギュレーションが保存されました。


**メモ：**保存した波形ウィンドウ設定は、[File] → [Open] をクリックすると読み込むことができます。

## デザインのシミュレーション

アップデートした波形コンフィギュレーションでデザインをもう一度シミュレーションします。  
[Run All] ボタン  をクリックして、シミュレーションを再実行します。

**メモ：**または、Tcl プロンプトに「run all」と入力してもシミュレーションを再実行できます。

シミュレーションが 13 マイクロ秒 (us) 間実行されます。

シミュレーションが完了したら、[Zoom to Full View] ボタン  をクリックして波形全体を表示します。

波形コンフィギュレーションは、図 4-23 のように表示されます。

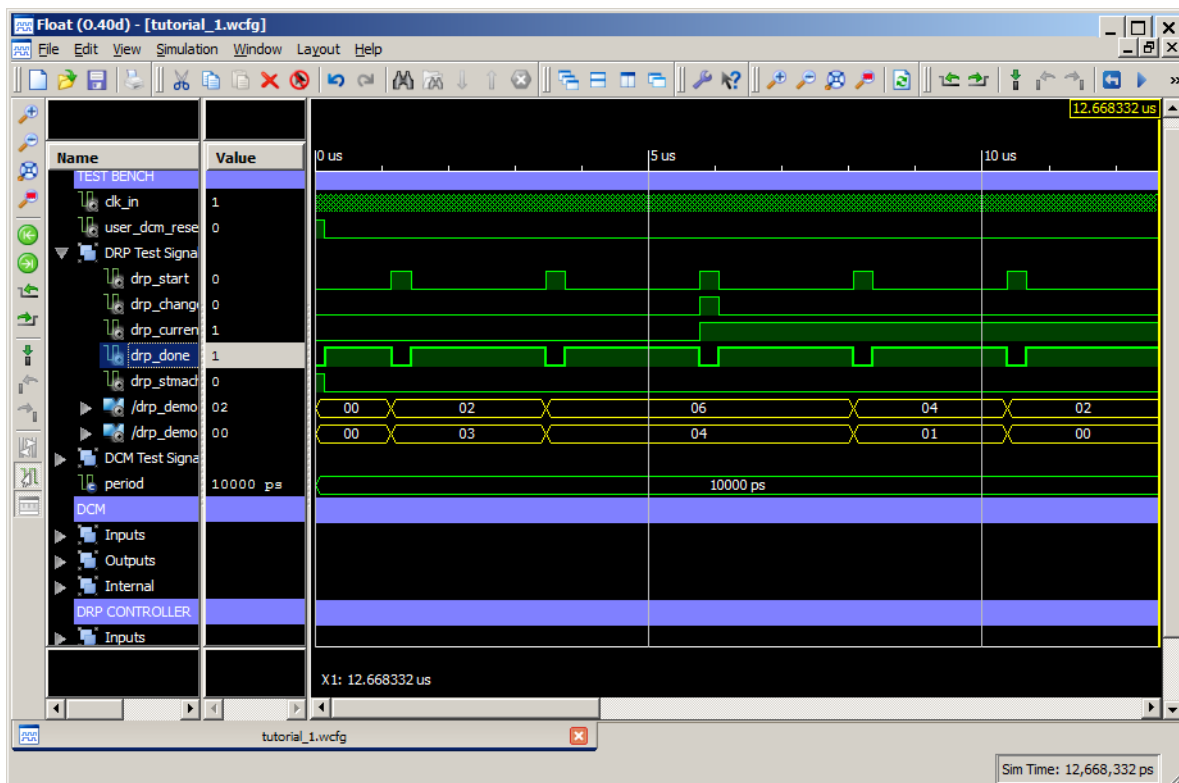


図 4-23 : 13us シミュレーション時間後の波形ウィンドウ

## マーカーの使用

このデザインで使用されるセルフチェック テストベンチでは、DCM のダイナミック リコンフィギュレーション機能を紹介するために 4 つのテストが実行されます。次の手順に従い、新しいテストの開始ごとにマーカーを使用します。

1. [Console] パネルで各テストの開始時にシミュレーション時間を確認します。たとえば、テスト 1 は次の [Console] パネルに表示されているように、1,150 ナノ秒あたりで開始します。

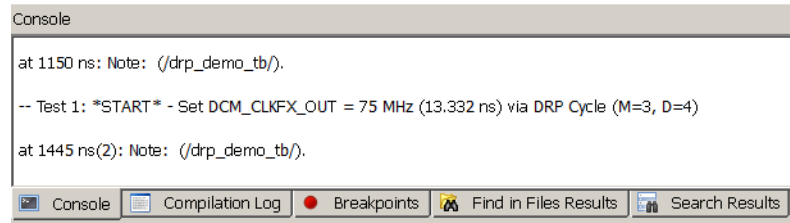


図 4-24 : [Console] パネル

2. [Edit] → [Go To] をクリックし、[Go To Time] に「1150 ns」と入力して、メイン カーソル (黄色) を最初のテストベンチ テストに移動させます。

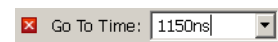



図 4-25 : [Go To Time] フィールド

3. [Add Marker] ボタン  をクリックしてここにマーカーを追加します。

メモ：入力した時間の単位はナノ秒ですが、波形ウィンドウにはマイクロ秒で表示されます。ピコ秒で入力することも可能です。波形ウィンドウは、選択した計測単位に合わせて変更されます。

4. 手順 2 ～ 3 をテストベンチで実行される 4 つのテストすべてで繰り返します。波形ウィンドウは、図 4-26 のように表示されます。

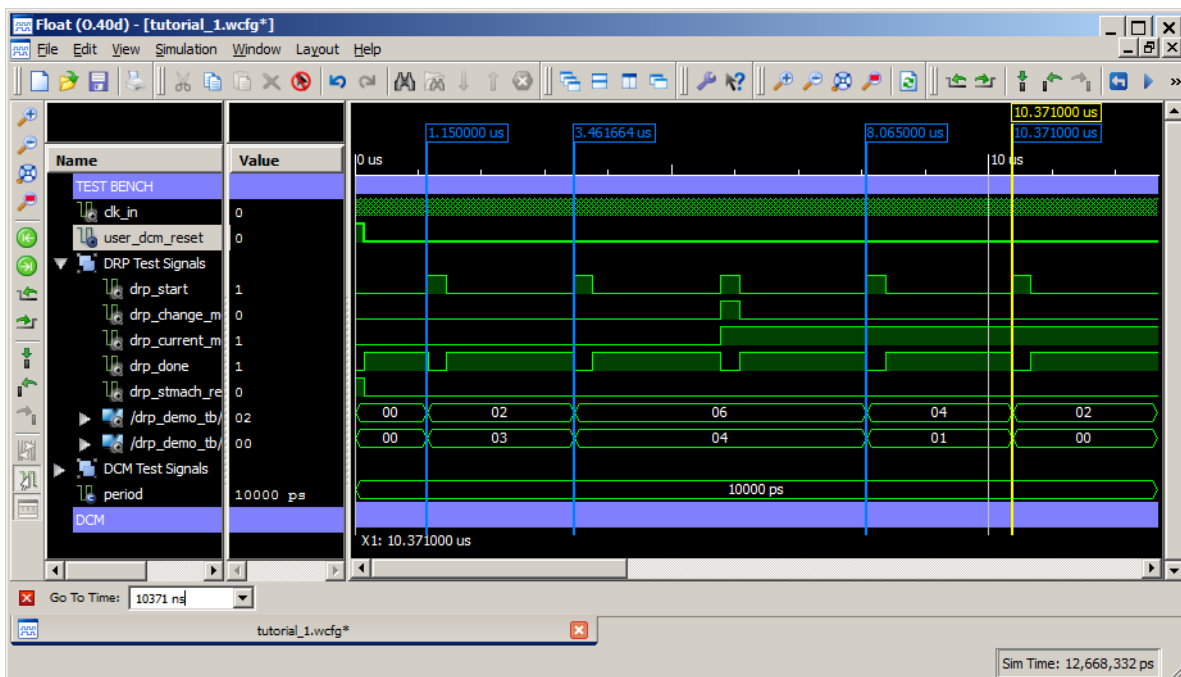


図 4-26 : テスト開始地点をマーカーで識別

## カーソルの使用

ISim の [Console] パネルでテスト 2 と 4 がエラーになったことがレポートされています (図 4-27)。

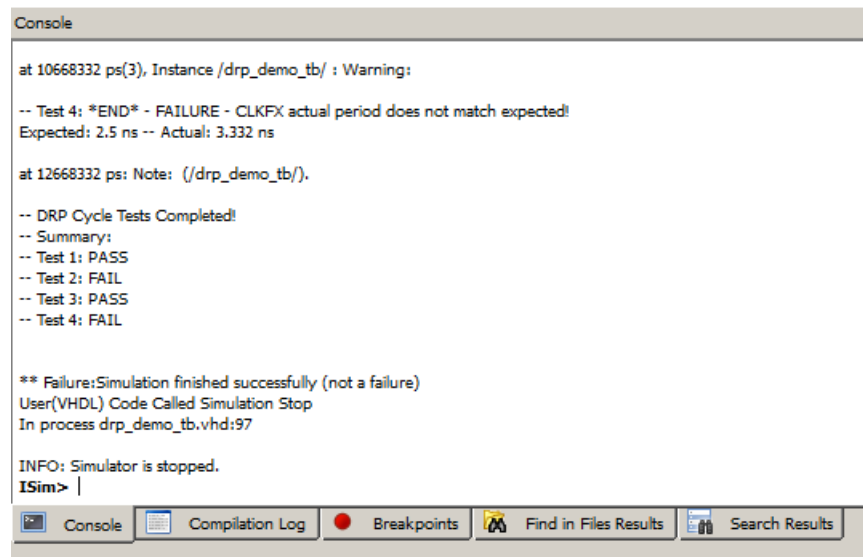


図 4-27：コンソール レポート：テスト 2 およびテスト 4 でエラーが発生

テスト 2 および 4 では、デジタル周波数合成での倍周率および分周率を変更して新しいクロック出力 (CLKFX) の周波数 (テスト 2 では 120MHz、テスト 4 では 400MHz) が設定されるようダイナミック リコンフィギュレーションポート (DRP) の書き込みサイクルが実行されます。ただし、DRP サイクルの最後でテストベンチにより計測された周期が予期する周期と一致しないことが判明しました。テスト 2 と 4 は、周期の不一致が原因でエラーになっています。テスト 2 のエラーを、図 4-28 に示します。

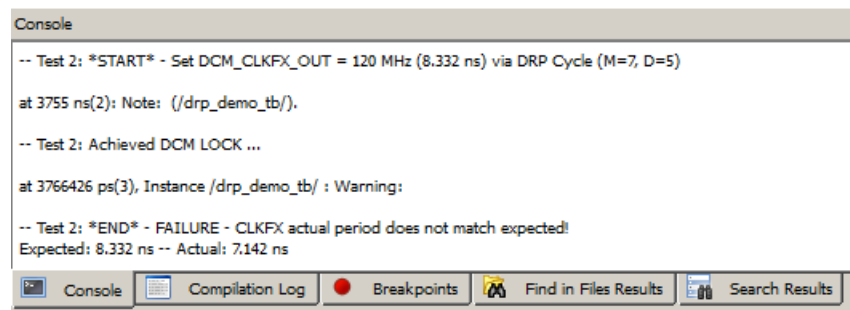


図 4-28：テスト 2 は周期の不一致が原因でエラーが発生



次の手順では、エラーが発生したときに ISim のメイン カーソル (黄色) を使用して波形ウィンドウで拡大表示します。また、dcm\_clkfx\_out 信号の周期をカーソルを使用して計測し、テストベンチの計測が正しいか確認します。


## 拡大表示

まず、波形ウィンドウでテスト 2 の開始地点を拡大表示して、出力クロック `dcm_clkfx_out` のステータスを確認します。

カーソルを使用して特定エリアを拡大表示するには、次の手順に従います。

1. メイン (黄色) カーソルをクリックしてテスト 2 の開始地点を示すマーカーの近くにドラッグします (マーカーの 3.461664  $\mu$ s 地点)。これで、カーソルがマーカーに揃います。

**メモ:** または、ツールバーの [Previous Marker] ボタン  または [Next Marker] ボタン  をクリックして、メイン カーソルをマーカー間で移動することも可能です。

2. [Zoom In] ボタン  をクリックして拡大表示します。

波形ウィンドウでカーソルで特定されたエリアが拡大表示されます。

3. DCM テスト信号 `dcm_clk0_out` および `dcm_clkfx_out` の立ち上がりと立ち下がりが確認できるまで、手順 2 を繰り返します。

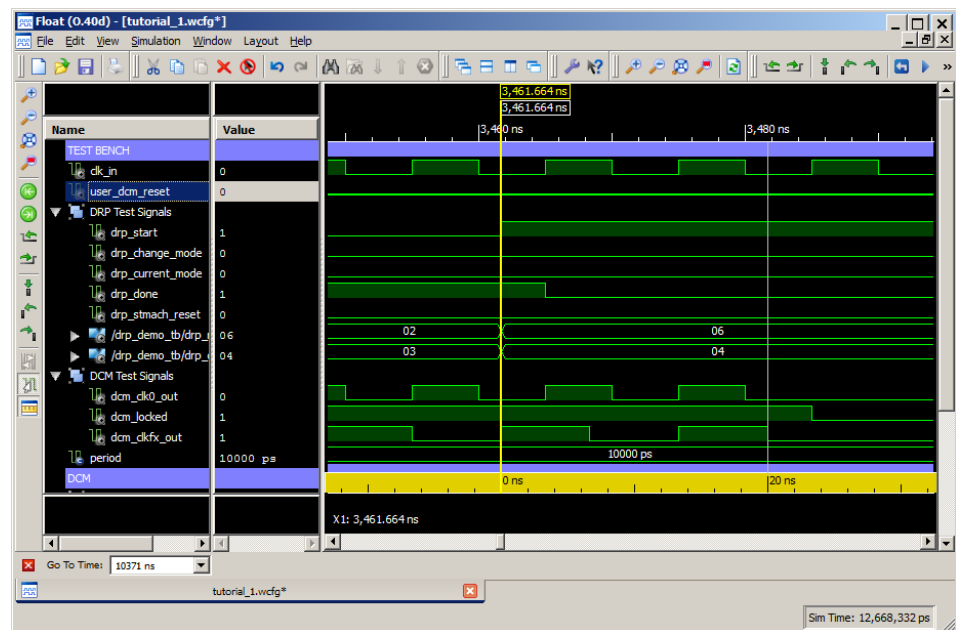



図 4-29 : `dcm_clk0_out` 信号および `dcm_clkfx_out` 信号の確認

## 時間の計測

カーソルを使用すると、2 つのエンドポイント間の時間を計測できます。この機能を使用すると、DRP サイクルが完了した後 (drp\_done 信号がアサートされた後) の dcm\_clkfx\_out の時間を計測することで、[Console] パネルにレポートされている、テスト 2 実行中にテストベンチで計測された値を確認できます。

カーソルを使用して時間を計測するには、次の手順に従います。

1. [Snap to Transition] ボタン  をクリックすると、カーソルを遷移エッジに合わせることができます。
2. DRP サイクル完了後 (drp\_done 信号のアサート後) の最初のクロックの立ち上がりエッジ付近でクリックしたままにします。メイン カーソルは dcm\_clkfx\_out の立ち上がりエッジに合わせられます。
3. ボタンを押したままにすると、マウスが次のクロックの立ち上がりエッジに移動します。セカンダリ マーカーが表示されます。

定義された 2 つのエンドポイント間の時間が波形ウィンドウ下に時間デルタとして表示されます (図 4-30)。

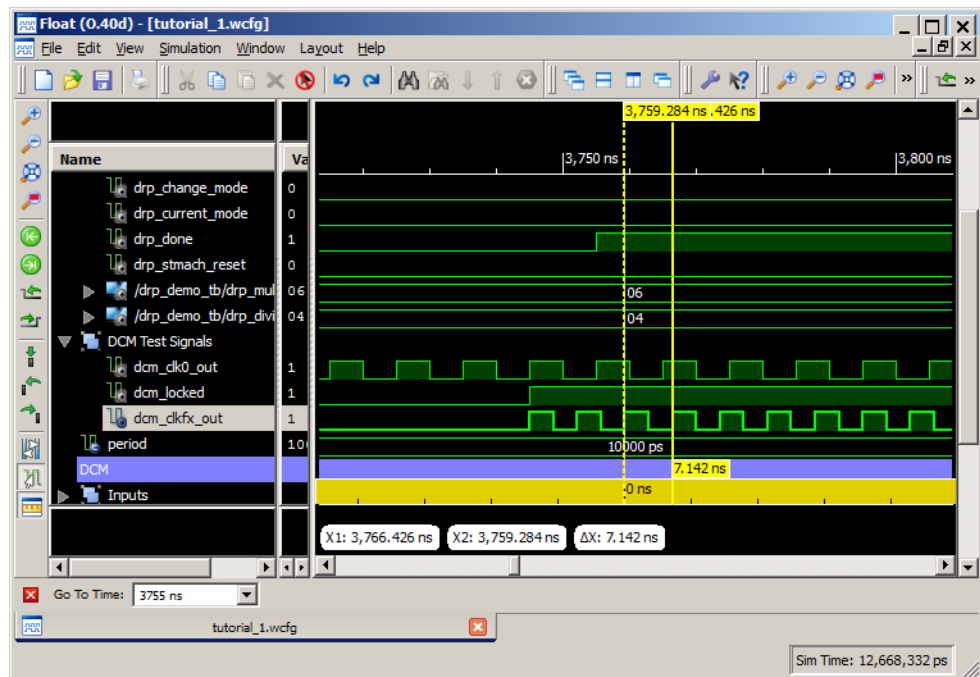



図 4-30：波形ウィンドウでの時間計測

カーソルを使用すると、dcm\_clkfx\_out 出力クロックの 2 つの立ち上がりエッジ間の時間は 7,142ps です。この値は、140MHz クロック信号と同等です。テスト 2 は 120MHz が予期される値のため、不一致が原因でエラーになります。

4. 同じ手順をテスト 4 で繰り返して解析します。テストベンチで 400Mhz が予期されるのに実際の周波数が 300Mhz と計測されることが確認できるはずです。

**メモ：**波形ウィンドウのツールバーにある [Floating Ruler] ボタン  をクリックして、波形コンフィギュレーション上にフロート ルーラーを表示します。この機能は、2 つのエンドポイント間をカーソルを使用して計測するときに使用できます。最初の時間エンドポイントには、ゼロ (0px) が

ルーラー上に表示されます。この機能は、最初のエンドポイントに相対して複数の時間計測を実行するときに便利です。

## 複数の波形コンフィギュレーションの使用

画面の解像度によっては、1つの波形ウィンドウで一度にすべての信号が表示されない場合があります。複数の波形ウィンドウを開くと、それぞれで特定の信号セットおよび信号のプロパティを表示できます。

次の手順に従い、新しい波形ウィンドウを開きます。

1. ISim で [File] → [New] をクリックします。
2. [New] ダイアログ ボックスで [Wave Configuration] を選択して [OK] をクリックします。

空の波形コンフィギュレーションが開きます。

仕切り、グループ、およびシミュレーション オブジェクトを新しい波形コンフィギュレーションに移動できます。次の手順に従い、DCM および DRP コントローラー ユニットに関連するすべてのシミュレーション オブジェクトを新しい波形ウィンドウに移動します。

1. Ctrl キーを押しながら新しい波形ウィンドウに移動するオブジェクト (仕切り、グループなど) をハイライトします。
2. 選択されている信号のいずれかを右クリックして [Cut] をクリックします。  
**メモ:** このチュートリアルでは、信号を別の波形コンフィギュレーションに移動しますが、[Copy] コマンドを使用すると信号をコピーでき、両方のウィンドウに信号を保持できます。
3. 新しい波形コンフィギュレーションのタブをクリックします。
4. 波形コンフィギュレーションの [Name] 列を右クリックして、[Paste] をクリックします。
5. [File] → [Save As] をクリックして、この波形コンフィギュレーションを「tutorial\_2.wcfg」という名前で保存します。

これで、図 4-31 および 図 4-32 のような 2 つの波形ウィンドウが表示されます。

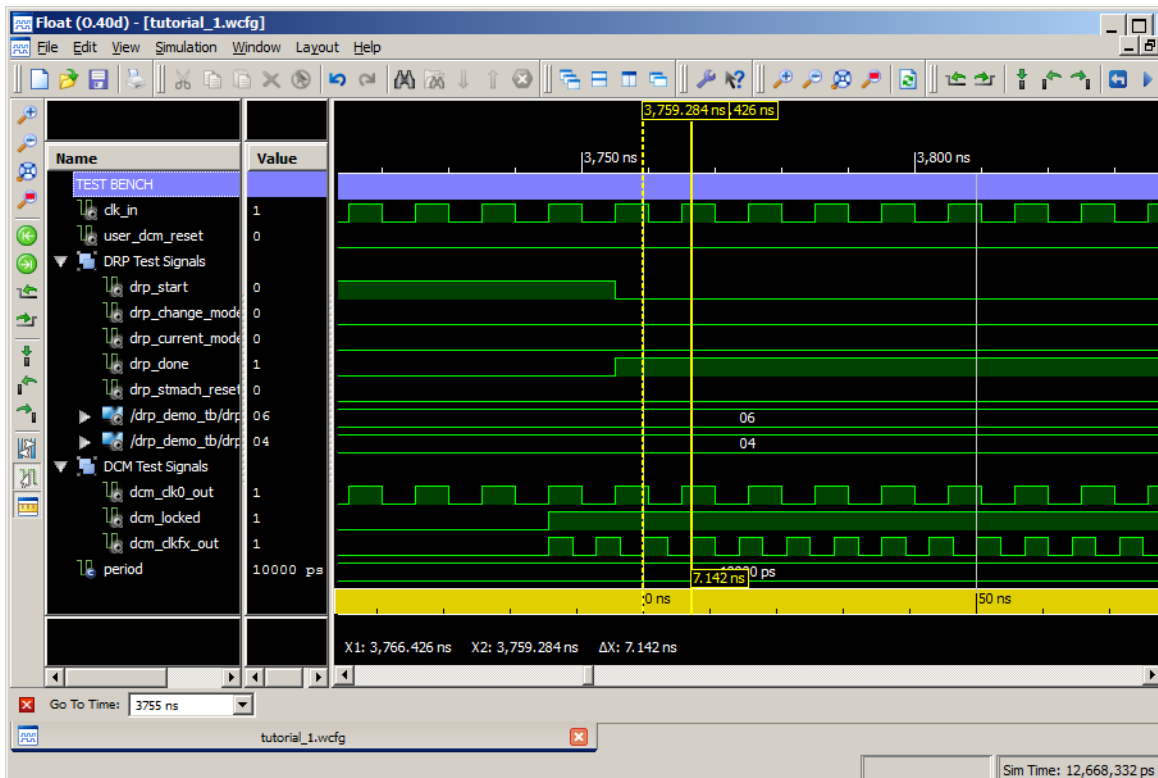


図 4-31 : tutorial\_1.wcfg

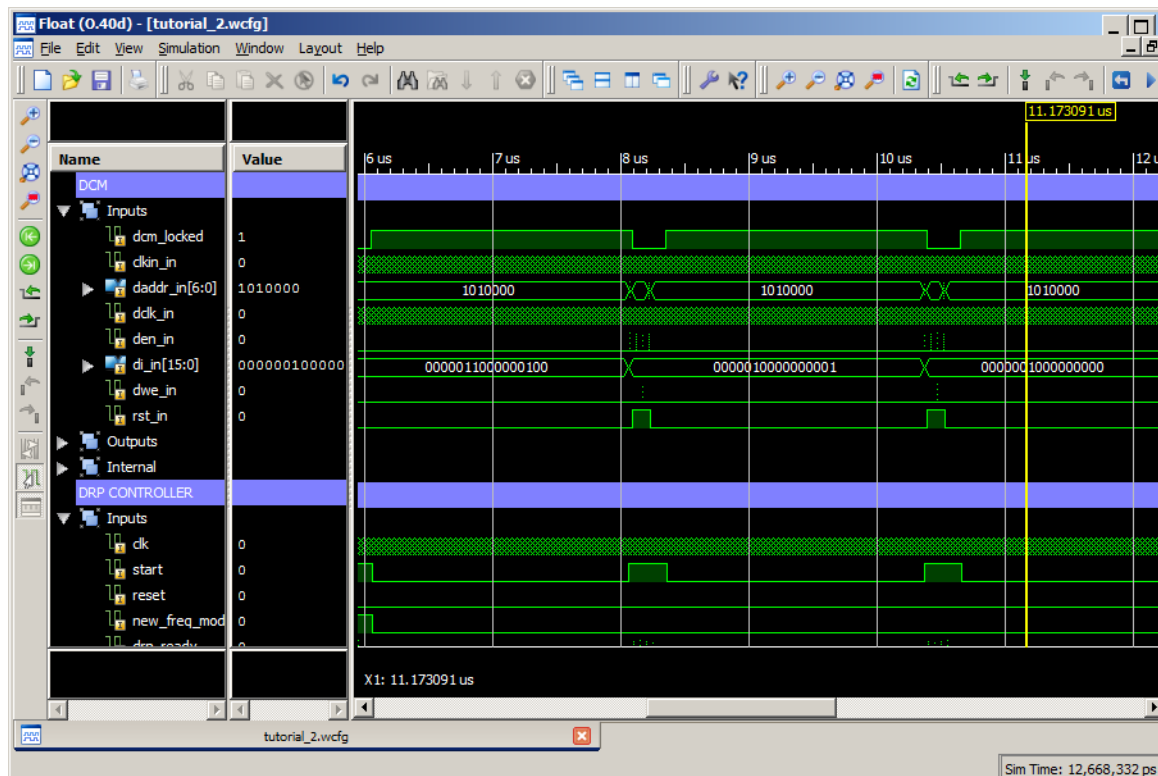


図 4-32 : tutorial\_2.wcfg



## デザインのデバッグ

マーカー、カーソル、および複数の波形コンフィギュレーションを使用してデザインを確認したので、次はブレークポイントを設定したりソースコードを1行ずつ実行するなど、ISim デバッグ機能を使用してデザインをデバッグし、エラーになった2つのDRPテストの問題を修正します。

### ソースコードの表示

まず、チュートリアルデザインのテストベンチを確認して、各テストがどのように実行されるか学びます。

次のいずれかの手順を実行して、チュートリアルデザインのテストベンチ (drp\_demo\_tb.vhd) のソースコードを開きます。ソースコードは統合されているテキストエディターで読み取り専用モードで開きます (図 4-33)。

- [File] → [Open] でファイルを選択します。
- [Instances and Processes] パネルでデザインユニットを右クリックして、[Go to Source Code] をクリックします。
- [Objects] パネルでソースファイルで宣言されているシミュレーションオブジェクトを右クリックして、[Go to Source Code] をクリックします。
- [Source Files] パネル ([Source Files] タブをクリックして表示) でソースファイルをダブルクリックします。

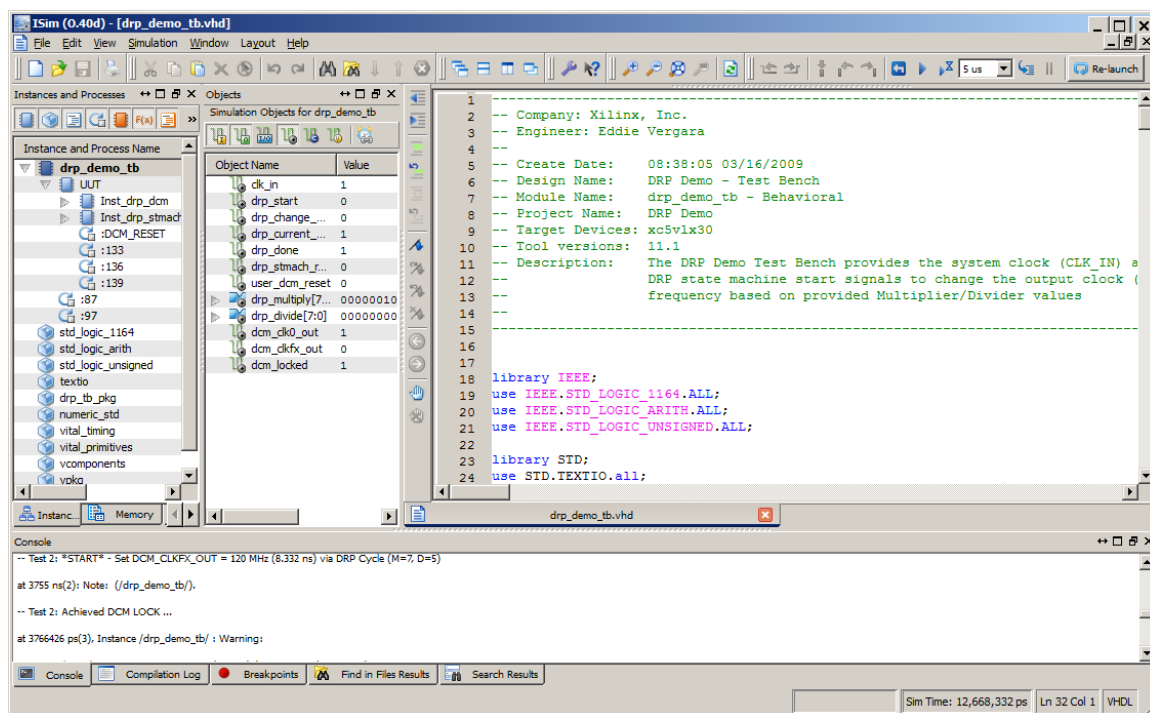


図 4-33 : 統合されたテキスト エディター

## ブレークポイントの使用とソース コードの 1 行ずつの実行

ブレークポイントは、ソース コードに含まれるユーザー定義の停止ポイントで、ISim でデザインをデバッグするときに使用します。ブレークポイントが設定されているデザインをシミュレーションすると、デザインのシミュレーションが各ブレークポイントで停止し、デザインの動作を確認できます。シミュレーションが停止すると、テキスト エディターでブレークポイントが設定されているソース コードの横にインジケータが表示され、ソース コードの特定のイベントと波形ウィンドウの結果を比較できます。

また、ISim デバッグ ツールでは、ソース コードを 1 行ずつ実行できます。1 行ずつソース コードを進めることでシミュレーション ユニットを 1 つずつ実行できます。この機能は、ソース コードがシミュレーション結果にどう影響するかを確認するときに便利です。

これら 2 つのデバッグ機能を使用すると、テスト 2 で DRP サイクルがどのように実行されるかを確認して、エラーが発生したテストをデバッグできます。

### ブレークポイントの設定

最初に、各 DRP サイクル テスト中に実行される最初の信号割り当て付近にブレークポイントを設定します。

次の手順に従い、drp\_demo\_tb.vhd の 185 行目にブレークポイントを設定します (図 4-34)。ブレークポイントを設定すると、信号 drp\_multiply に値が割り当てられるたびにシミュレーターが停止します。

1. ブレークポイントを追加するソース コードを開きます。
2. ソース コードに含まれている実行行に移動します。
3. 実行行を右クリックして [Toggle Breakpoint] をクリックし、ブレークポイントを追加します。

**メモ：**または、[Toggle Breakpoint] ボタン  をクリックしてもブレークポイントを追加できます。

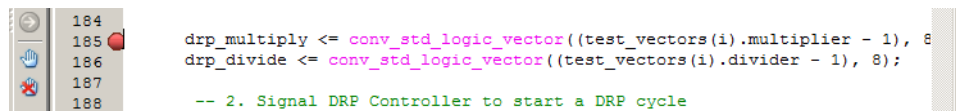


図 4-34 : drp\_demo\_tb.vhd の 185 行目にブレークポイントを設定

**メモ：**[Breakpoints] タブ ([Console] タブの右横) をクリックすると、ブレークポイントを管理できます。リストには、すべてのブレークポイントが表示されます。このリストで次を実行できます。

- 選択したブレークポイントまたはすべてのブレークポイントの削除
- 選択したブレークポイントが設定されているソース コードの行に移動

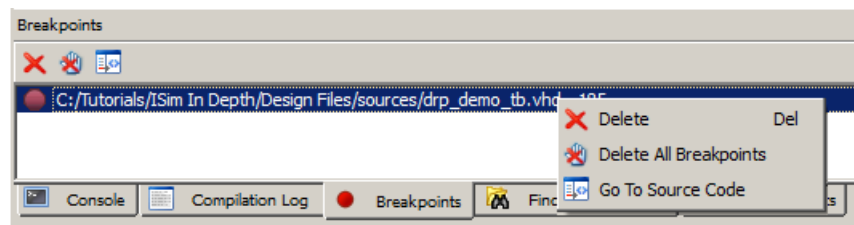




図 4-35 : [Breakpoints] タブの右クリック メニュー


## ブレイクポイントをイネーブルにした状態でのシミュレーションの再実行

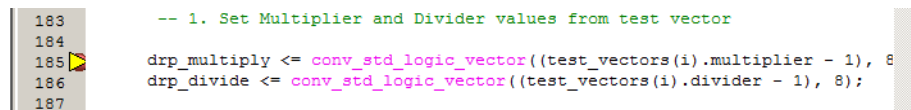
次に、ブレイクポイントをイネーブルにした状態でシミュレーションを再実行します。

ブレイクポイントとソースコードの 1 行ずつの実行機能を使用してデバッグするときは、[Console] パネルと波形ウィンドウが同時に確認できる環境が最適です。ISim の各パネルのフロート機能を使用するか、またはシミュレーターの各ウィンドウ サイズを調整して、これらが同時に確認できるようにしてください。

1. [Run All] ボタン  をクリックして、シミュレーションを再実行します。
2. シミュレーションを実行するには、[Run All] ボタン  をクリックします。

最初のテストの開始付近でシミュレーションが実行されます。

テキスト エディターが表示されシミュレーターで実行されたソースコードの最後の行の横に黄色のインジケーター  が表示されます。



```

183      -- 1. Set Multiplier and Divider values from test vector
184
185      drp_multiply <= conv_std_logic_vector((test_vectors(i).multiplier - 1), 8)
186      drp_divide <= conv_std_logic_vector((test_vectors(i).divider - 1), 8);
187

```

図 4-36 : 実行されたソースコードの最後の行

また、[Console] パネルには、シミュレーターが停止したことを示すメッセージと共にその際シミュレーターにより最後に実行されたソースコード行が表示されます。

デザインを確認した際にテスト 1 が正しく終了することは、すでに確認されていますので、このテストのデバッグは飛ばすことができます。

3. テスト 2 に進むには、[Run All] ボタン  をクリックします。

これで、シミュレーションがテスト 2 の開始付近で停止します。

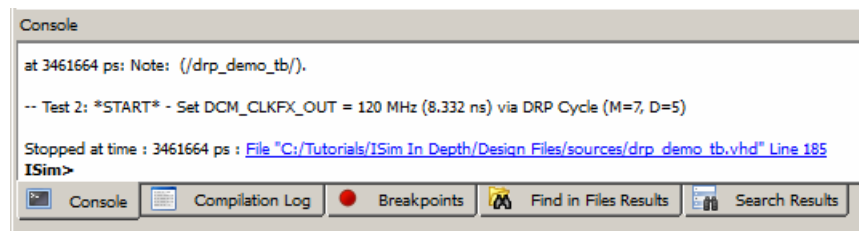



図 4-37 : シミュレーターが停止したことを示す [Console] パネルのメッセージ

## ソース コードの 1 行ずつの実行

まず、テスト 2 で `drp_multiply` および `drp_divide` バス信号を介して倍周率と分周率パラメーターが正しく設定されることを確認します。ソース コードを一行ずつ進めることで、`drp_multiply` および `drp_divide` バス信号がどのように DCM の DRP ポートに割り当てられるかを確認します。

ソース コードを一行ずつ進めるには、[Step] ボタン  をクリックします。

**メモ：**または、Tcl プロンプトに「`step`」と入力してもソース コードを一行ずつ進めることができます。

1. このように一行ずつソース コードを進めることで、次のイベントそれぞれを確認します。
  - `drp_multiply` および `drp_divide` バス信号が定数の `test_vectors` から割り当てられている。
  - DRP サイクルが開始するよう `drp_start` バス信号がアサートされる。
  - `drp_multiply` バス信号が `DI_IN` バス信号の上位 8 ビットに割り当てられており、`drp_divide` バス信号が同じバスの下位 8 ビットに割り当てられている。
  - DRP コントローラー (`drp_stmach.vhd`) が `idle` モードから次の DRP サイクルに移行し、DCM ステータス レジスタがクリアされる。
2. `tutorial_2` 波形ウィンドウで DCM 入力バスを展開表示します。
3. `di_in` バス信号が新しい値で更新されるまでシミュレーションを進めます。この変化を確認するために、波形を各拡大表示する必要がある場合があります。バスは、3,465ns 付近で 0203h から 0604h に更新されているはずです。

**メモ：**バス信号 `di_in` の基数を [Hexadecimal] に変更し、値の変化を確認します。

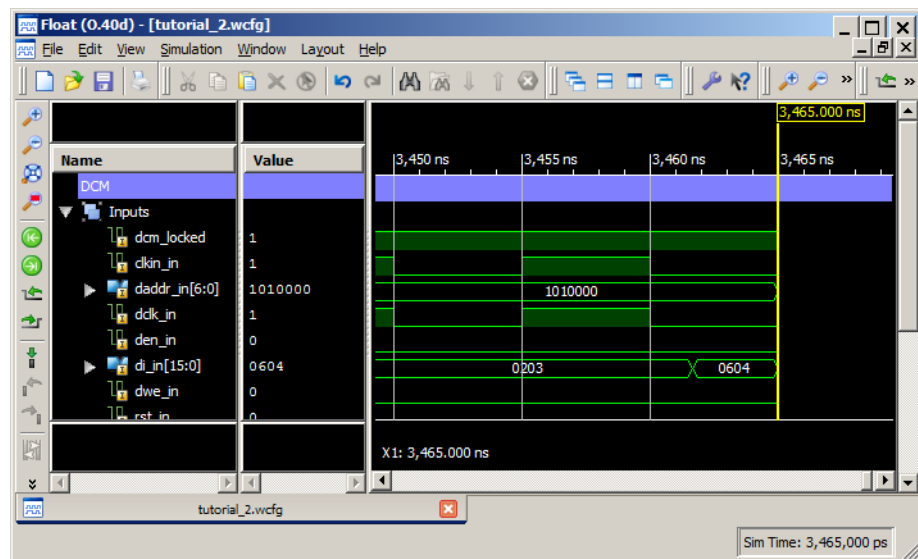


図 4-38：波形ウィンドウで DCM DI\_IN 入力バスの出力を確認

このデザイン (`dcm_clkfx_out`) の出力クロック周波数は、倍周率と分周率に基づいています。テスト 2 では、次のパラメーターおよび予期される出力クロック周波数を使用します。

表 4-1：パラメーターおよび予期される出力クロック周波数

テスト	周波数(MHz)	周期 (ps)	倍周率 (M)	分周率 (D)
2	120	8,332	6	5

M=6 および D=5 では、di\_in[15:0] のバス値が 0504h になっているはずです。テスト 2 では di\_in のステータスが 0604h になっています。テスト 2 がエラーになったのは、テストベンチの drp\_multiply 信号および drp\_divide 信号で供給される M/D 係数が不正のためです。

上記の手順をテスト 4 で繰り返すと、エラーの原因を追求できます。テスト 4 でもテストベンチでの倍周率および分周率の不正な割り当てが原因であることがわかります。

## デザインのバグの修正

ブレークポイントと 1 行ずつのソースコードの実行機能を使用することで、テストベンチの drp\_multiply 信号および drp\_divide 信号に割り当てられている倍周率および分周率の値が不正であることがわかりました。

次の手順に従い、テスト 2 および 4 で正しい倍周率と分周率が使用されるよう、テストベンチのテストベクターを変更します。

1. ISim メイン ウィンドウで [Source Files] タブをクリックします。
2. drp\_demo\_tb.vhd ファイルを右クリックして、[Go To Source Code] をクリックします。
3. 4 つの DRP テストのテストベクターは、117 行目から 127 行目で定義されています。定数宣言を次のように変更します (変更は太文字で表記)。

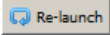
```
-----  
-- ** TEST VECTORS **  
-- (Test, Frequency, Period, Multiplier, Divider)  
-----  
  
constant test_vectors : vector_array := (  
  
    ( 1, 75, 13332 ps, 3, 4),  
    ( 2, 120, 8332 ps, 6, 5),  
    ( 3, 250, 4000 ps, 5, 2),  
    ( 4, 400, 2500 ps, 4, 1));
```

4. ファイルを保存します。


## バグ修正の確認

テストベンチのソース コードが修正されたので、ソース コードをコンパイルし直して、新しくシミュレーション実行ファイルを構築します。

1. [Breakpoints] タブをクリックして、以前に設定したブレイクポイントを削除します。

2. [Re-launch] ボタン  をクリックして、ISim を再起動します。

ISim でソース ファイルがコンパイルされ直されて、シミュレーションが読み込まれます。

3. 更新したテストベンチでデザインをもう一度シミュレーションします。[Run All] ボタン  をクリックして、シミュレーションを再実行します。

**メモ：**または、Tcl プロンプトに「run all」と入力してもシミュレーションを再実行できます。

テストベンチのテスト ベクターが正しく変更されている場合は、シミュレーションが正しく完了し、すべてのテストが完了したことが通知されます (図 4-39)。

```
-- DRP Cycle Tests Completed!  
-- Summary:  
-- Test 1: PASS  
-- Test 2: PASS  
-- Test 3: PASS  
-- Test 4: PASS
```

図 4-39：すべてのテストが完了したことを示す [Console] パネル

## 次の操作

これで ISim アドバンス チュートリアルが終了しました。ISim の詳細は、「このチュートリアルについて」の「[その他のリソース](#)」セクションを参照してください。