ISim チュートリアル

UG682 (v13.4) 2012 年 1 月 18 日

XILINX®

本資料は英語版 (v13.4) を翻訳したもので、内容に相違が生じる場合には原文を優先します。
資料によっては英語版の更新に対応していないものがあります。
日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

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

改訂履歴

次の表に、この資料の改訂履歴を示します。

日付	バージョン	改訂内容
2011 年 10 月 19 日	13.3	• 「このチュートリアルについて」の章を削除 • 13.3 リリースに合わせて更新 (書式および表現の改善のみ)
2011 年 1 月 18 日	13.4	• 「必要なデザイン ツール」、「チュートリアル デザイン ファイルのインストール」、および「デザインの概要」を第 1 章「ISim の概要」に移動 • コンパイル、シミュレーション、GUI、デバッグ フローに合わせてトピックの順序を変更し、スタンドアロン ISim を最終章に移動 • 27 ページの図 3-12 を修正 • 言語を標準化

目次

XILINX®

第 1 章

ISim の概要

ISim チュートリアルでは、ザイリンクス設計者向けに ISim シミュレーション ツールを詳細に説明します。

このチュートリアルは、Windows 環境で ISim ツール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するため変更が必要な場合があります。

ISim は、VHDL、Verilog、および混合言語デザインで論理 (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実行できるハードウェア記述言語 (HDL) シミュレータです。ISim 環境は、次の主要エレメントで構成されています。

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

注記：ISim の詳細は、『ISim ユーザー ガイド』を参照してください。この資料へのリンクは、付録 A 「その他のリソース」に含まれています。

チュートリアル フロー

このチュートリアルでは、ISE® Project Navigator から 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フローを紹介します。

このフローでは、ISE Project Navigator で実行可能なシミュレーション プロセスから ISim を起動します。ISE Project Navigator を使用してプロジェクトを作成し、デザインをザイリンクス FPGA にインプリメントします。

チュートリアル ファイルには HDL 以外のソースも含まれており、これらのソースを ISim でコンパイルできるように Project Navigator で HDL ソースフ ァイルに変換する方法も示します。

また、スタンドアロン ISim を使用して、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、HDL リンカーおよびシミュレーション実行ファイルを起動してデザインをシミュレーションする方法を説明する章もあります。

ISim コンポーネント

表 1-1 に、ISim のコンポーネントを示します。

表 1-1 : ISim コンポーネント

ISim コンポーネント	説明
パーサー : vhpcomp および vlogcomp	vhpcomp は VHDL ソース ファイル、vlogcomp は Verilog ソース ファイルを解析およびコンパイルします。解析されたダンプは fuse コマンドで使用され、オブジェクト コードが生成され、このコードがシミュレーション カーネル ライブラリとリンクされて、シミュレーション実行ファイルが生成されます。
fuse コマンド	fuse コマンドは、ISim ツールで使用される HDL エラボレーターおよびリンカーです。最上位デザイン ユニットを指定したデザインに対してスタティック エラボレーション (既存デザインのエラボレーション) を実行し、これらのデザイン ユニットをオブジェクト コードにコンパイルします。その後デザイン ユニットのオブジェクト ファイル間がリンクされて、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) に含まれている各 VHDL または Verilog ソース コードに対して vlogcomp または vhpcomp を自動的に実行し、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。デフォルト名は x.exe で、ISim ツールでデザインのシミュレーションを実行するときに使用できます。 • ISim ツールを ISE Project Navigator または PlanAhead ツールから実行する場合は、これらのツールで生成された x.exe が起動されます。 • ISim ツールをコマンド ラインから実行する場合は、生成された実行ファイルを明示的に指定して起動する必要があります。詳細は、第 4 章「スタンドアロン ISim の実行」を参照してください。 シミュレーション実行ファイルでは、デザインを駆動およびプローブする Tcl コマンドを使用して、イベント ドリブンのシミュレーションが開始されます。 **注記** : 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のフォーマットは x.exe です。
isimgui.exe	isimgui.exe (Linux では isimgui) は ISim GUI で、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。

必要なデザイン ツール

このチュートリアルを実行するには、次のいずれかのデザイン ツールをインストールする必要があります。

- ISE WebPACK™ 13.x
- ISE Design Suite 13.x Edition (Logic、DSP、Embedded、System)

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 : インストールおよびライセンス ガイド』(UG798) を参照してください。この資料へのリンクは、付録 A 「その他のリソース」に含まれています。

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、ザイリンクスのウェブサイトのチュートリアル ページから入手できます。

- `ug682.zip` デザイン 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。

この `.zip` ファイルには、表 1-2 に示すファイルが含まれています。

表 1-2 : 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、チュートリアルで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。
`readme.txt`	このチュートリアル デザインに関する readme ファイル

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。

出力クロック = 入力クロック * (逓倍率 / 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、逓倍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。

詳細は、『Virtex-5 FPGA ユーザー ガイド』(UG190) の「クロック リソース」の章を参照してください。この資料へのリンクは、付録 A 「その他のリソース」に含まれています。

論理ブロック

このチュートリアル デザインには、次の論理ブロックが含まれています。

drp_dcm (drp_dcm.vhd)

内部フィードバック、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む Virtex-5 DCM マクロです。

CLKFX_OUT 出力からは、次の式で定義されるクロックが供給されます。

```
CLKFX_OUT = CLKIN_IN * (逓倍率/分周率)
```

たとえば、100MHz 入力クロックを使用するとき、逓倍率を 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、逓倍率 (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、これらの逓倍率および分周率のパラメーターを 16 ビット幅の DI_IN ポートを使用して 16 進数フォーマットで DCM に供給する方法を示します。

```
DI_IN[15:8] = M - 1

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)

drp_stmach.vhd モジュールには、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 (DRP) 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、フル サイクルが完了します。

drp_done 信号は、DRP 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 デザインの最上位モジュールで、DCM マクロおよび DRP コントローラー モジュールが外部 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、次のサブセクション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するため、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)。セルフチェック テストベンチには、予測され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 クロック用に 100MHz 入力クロックを生成 (clk_in)
- デザインの出力周波数をダイナミックに変更する 4 つのテストを実行。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、出力クロックが異なる周波数に設定されます。

9 ページの表 1-3 に、各テストで予測される出力周波数および逓倍率/分周率パラメーターを示します。

表 1-3 : 出力周波数および逓倍率/分周率パラメーター

テスト	周波数(MHz)	周期 (ps)	逓倍率 (M)	分周率 (D)
1	75	13,332	3	4
2	120	8,332	6	5
3	250	4,000	5	2
4	400	2,500	4	1

- 各テストでは、テストベンチにより予測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、その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かを示すメッセージが表示されます。テストベンチで使用される 2 つのソース ファイルにはエラーが含まれているので、チュートリアル後半で ISim ツールのデバッグ機能を使用して修正します。
- 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。

注記 : 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 『効率的なテストベンチの作成』を参照してください。

XILINX®

第 2 章

ISim の実行

ザイリンクス ISE® Design Suite では、ISim を使用した統合フローが提供されており、ISE Project Navigator または PlanAhead™ ツール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、デザインをシミュレーションすると生成され、バックグランドで自動的に実行されます。

この章では、ISE Project Navigator でデザインをコンパイルし、シミュレーションを設定および実行する方法を説明します。

デザインのコンパイル

ISim 統合フローでは、ISE Project Navigator または PlanAhead ツール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実行できます。

このチュートリアル フローでは、まずチュートリアル デザイン用の ISE プロジェクトを作成し、その後ビヘイビアー 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 デザイン用の ISE プロジェクトを作成します。

注記：この手順の前に、7 ページの「チュートリアル デザイン ファイルのインストール」を参照して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

Project Navigator を起動して、ISE プロジェクトを作成します。

1. ISE Project Navigator を起動します。
2. [File] → [New Project] をクリックして New Project Wizard を起動します。
3. [Create New Project] ページで、プロジェクトの名前 (例 : ISim_Tutorial) および保存先を入力します。
4. [Next] をクリックします。
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。12 ページの図 2-1 と同様に設定し、[Next] をクリックします。

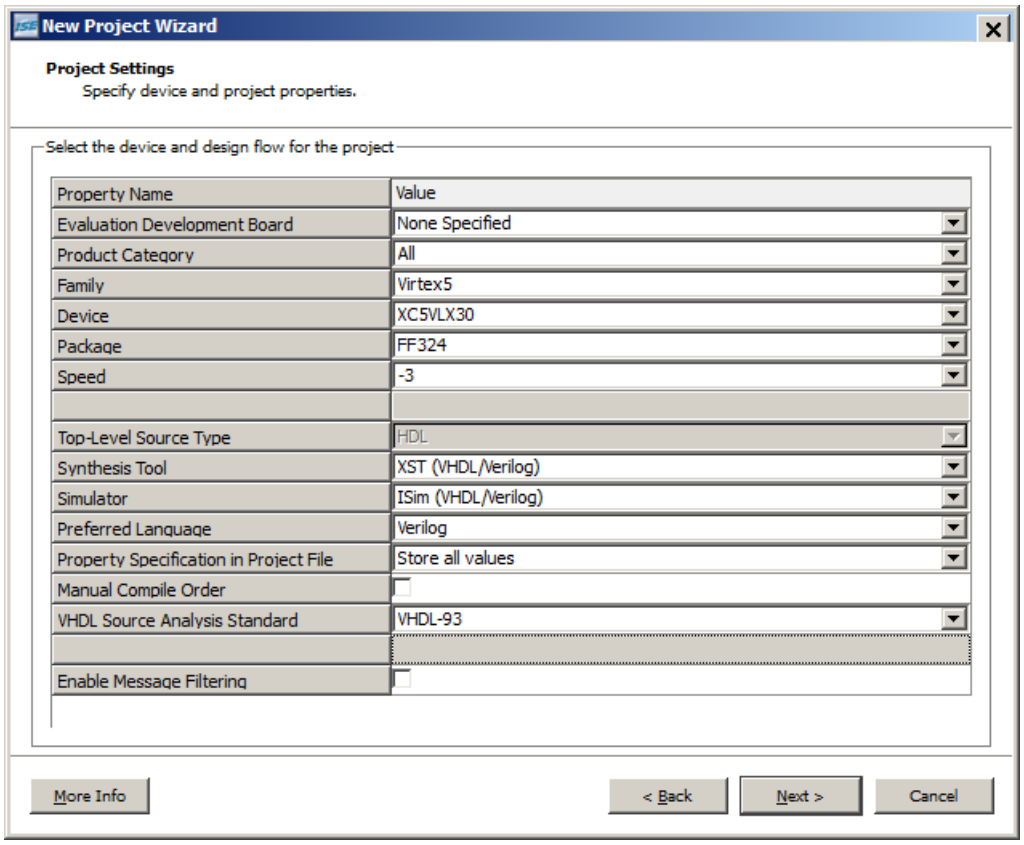


図 2-1 : New Project Wizard : [Project Settings] ページ

6. [Project Summary] ページを確認し、[Finish] をクリックします。

プロジェクトへのチュートリアル ソース ファイルの追加

1. [Project] → [Add Source] をクリックします。
2. 7 ページの「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。
3. /sources フォルダーに含まれるファイルをすべて選択し、[開く] をクリックします。
4. [Adding Source Files] ダイアログ ボックスで、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 2-2 と比較してください。

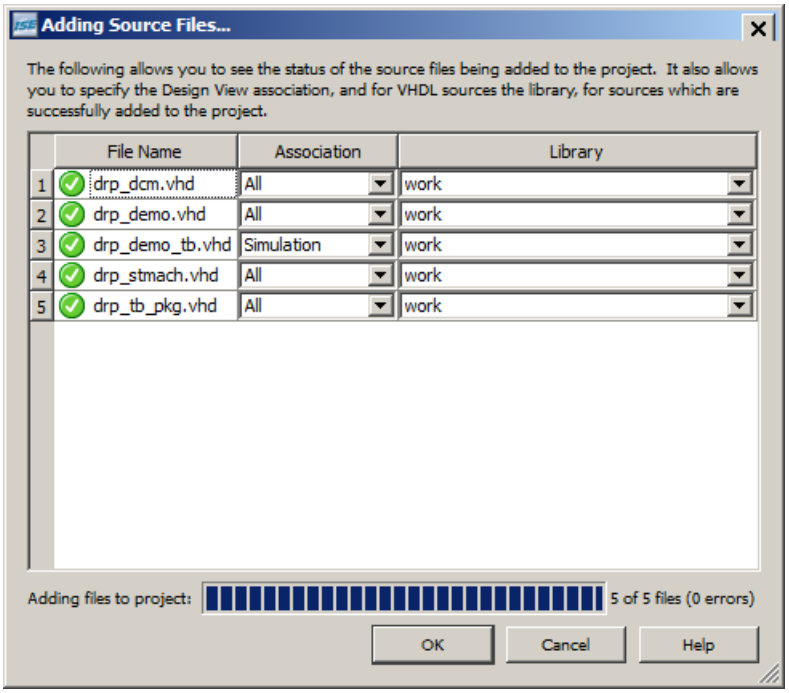


図 2-2 : [Adding Source Files] ダイアログ ボックス : ソース ファイルと関連付けのステータス

5. [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 のメイン ウィンドウは、14 ページの図 2-3 のように表示されます。

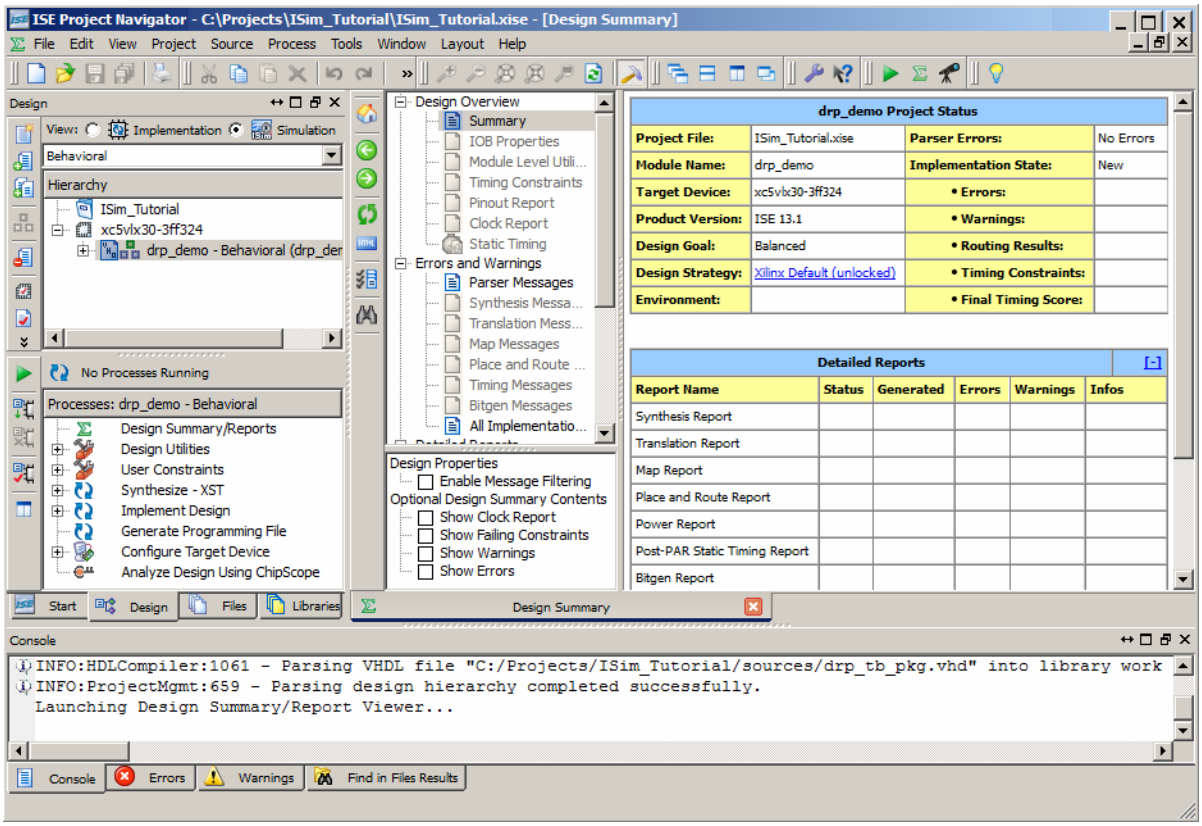


図 2-3 : ISE Project Navigator のデザイン サマリ

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`) 用のユーザー VHDL ライブラリを作成します。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 ライブラリを作成したら、VHDL パッケージ ファイルを `/work` 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。

VHDL ライブラリを作成するには、次の手順に従います。

1. Project Navigator で [Project] → [New Source] をクリックします。

   New Source Wizard が開きます。

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 (15 ページの図 2-4)。

   注記：[Add to project] はオンのままにします。

4. [Next] をクリックします。

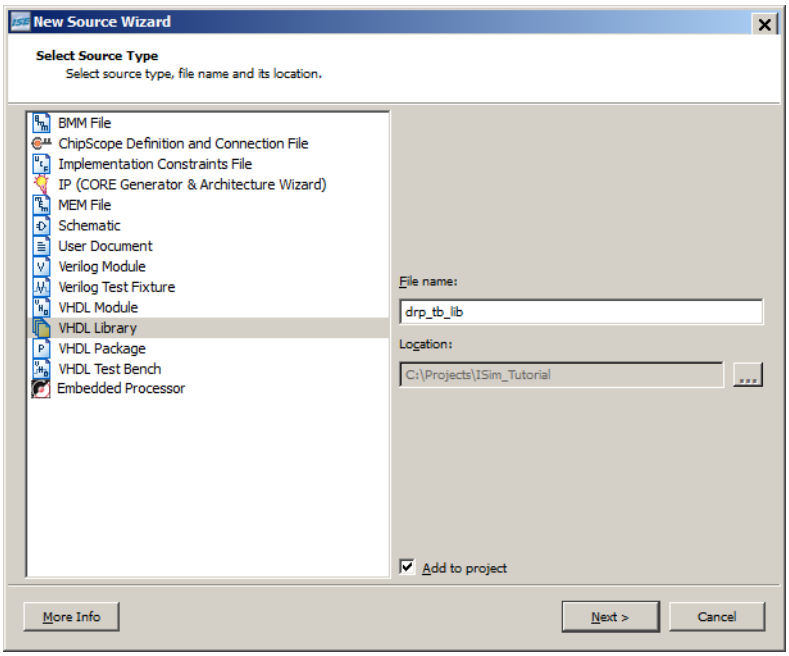


図 2-4 : ソース タイプの選択

5. [Summary] ページを確認してから [Finish] をクリックします。

VHDL ファイルのライブラリへの移動

VHDL パッケージ ファイルを drp_tb_lib ライブラリに移動します。

1. [Libraries] パネルをクリックします。
2. [work] ライブラリを展開表示します (図 2-5)。

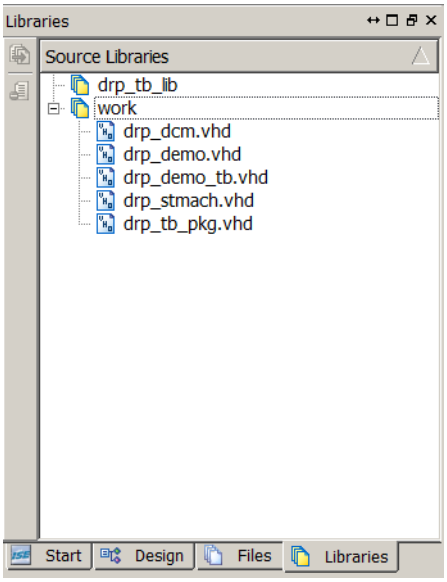


図 2-5 : [Libraries] パネル

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として `drp_tb_lib` を選択します。

   または、ファイルをこのライブラリにドラッグアンドドロップすることもできます。

   `fuse` コマンドでは、プロジェクト ファイル (`.prj`) に含まれている各 VHDL または Verilog ソース コードに対して `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。

5. [OK] をクリックします。

   [drp_tb_lib] ライブラリに VHDL パッケージ ファイル `drp_tb_pkg.vhd` が含まれていることを確認します (図 2-6)。

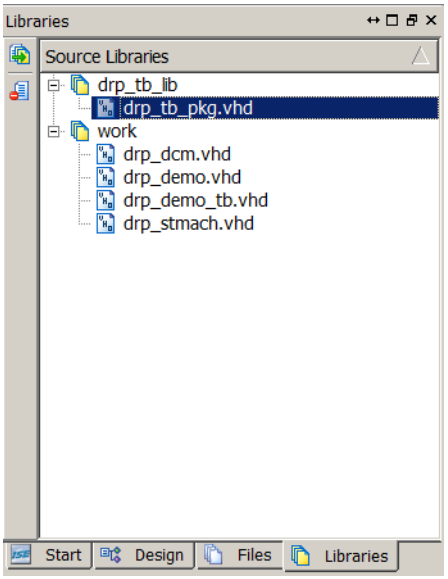


図 2-6 : ソース ライブラリ

ビヘイビアー シミュレーションの設定および起動

ISE プロジェクトを作成したので、次に ISim ビヘイビアー シミュレーションを設定および起動します。

ビヘイビアー シミュレーション プロパティの設定

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

1. [Design] パネルの上部にある [View] ペインで [Simulation] ボタンをオンにします。

   シミュレーション タイプを指定するドロップダウン リストが表示されます。

2. [Behavioral] を選択します。

3. `drp_demo_tb` テストベンチ ファイルを選択し、[Processes] ペインで [ISim Simulator] を展開します。

   [Processes] ペインにデザインで実行可能なシミュレーション プロセスが表示されます (17 ページの図 2-7)。

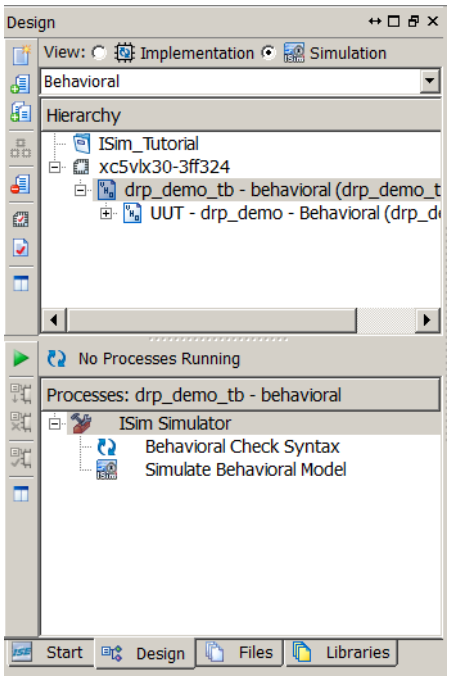


図 2-7 : ビヘイビアー シミュレーション プロセス

4. [ISim Simulator] の下の [Simulate Behavioral Model] を右クリックし、[Process Properties] をクリックします。

   [Process Properties - ISim Properties] ダイアログ ボックスが開きます (図 2-8)。

   このダイアログ ボックスでは、次のようなシミュレーション プロパティを設定できます。

   - シミュレーション実行時間
   - 波形データベースフ ァイルの保存先
   - ユーザー定義のシミュレーション コマンド ファイル

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

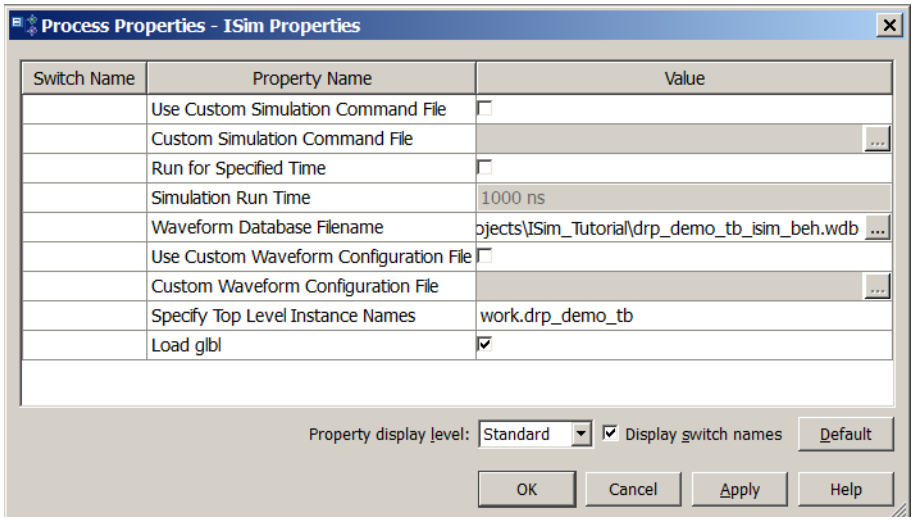


図 2-8 : [Process Properties - ISim Properties] ダイアログ ボックス

ビヘイビアー シミュレーションの起動

ISim を起動してチュートリアル 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。

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

結果

デザインが解析およびコンパイルされ、ISim GUI (図 2-9) が開きます。

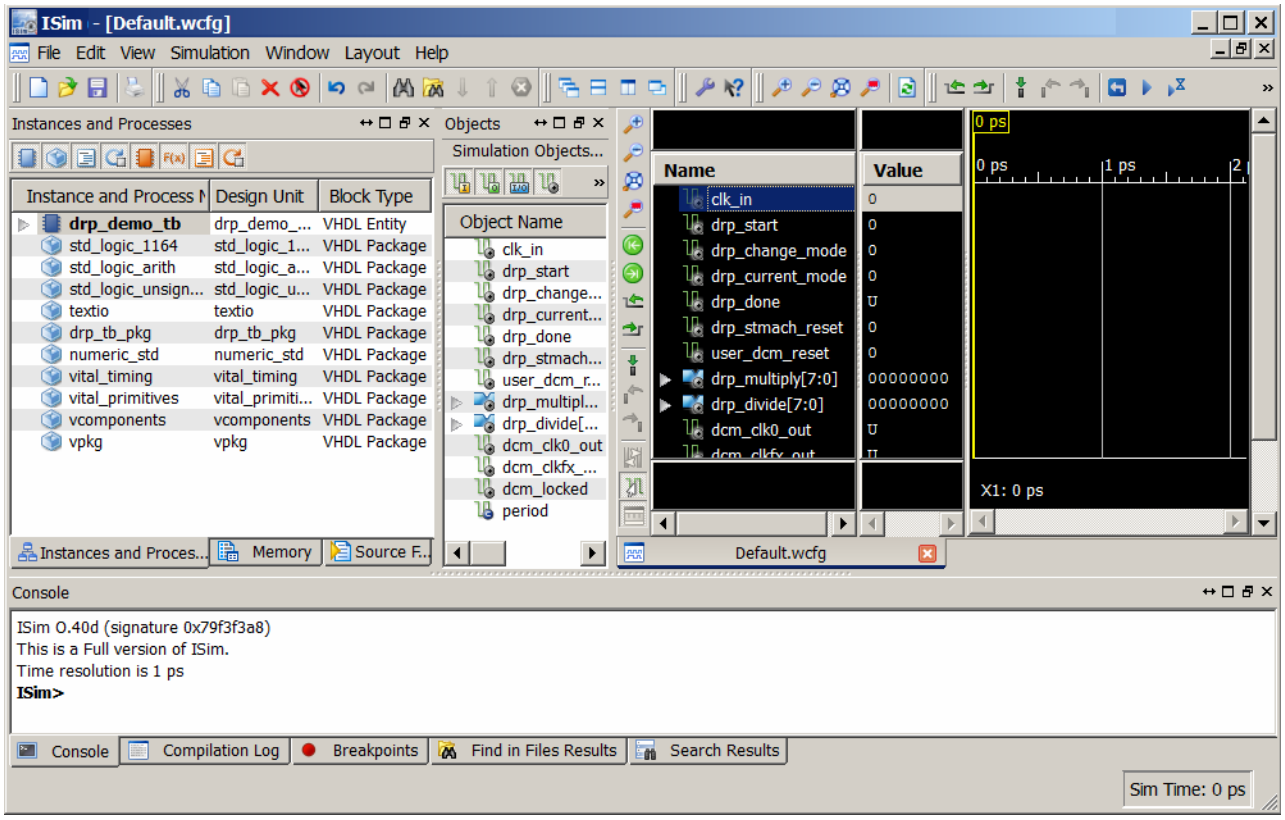


図 2-9 : シミュレーションを実行した後の ISim GUI

次の手順

第 3 章「ISim GUI の使用およびデザインのデバッグ」では、ISim GUI の機能と HDL デザインの解析およびデバッグに使用するツールについて説明し、第 4 章「スタンドアロン ISim の実行」では同じデザインを ISim コマンドを使用してスタンドアロンで実行します。

XILINX®

第 3 章

ISim GUI の使用およびデザインのデバッグ

ISim 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 3-1)。

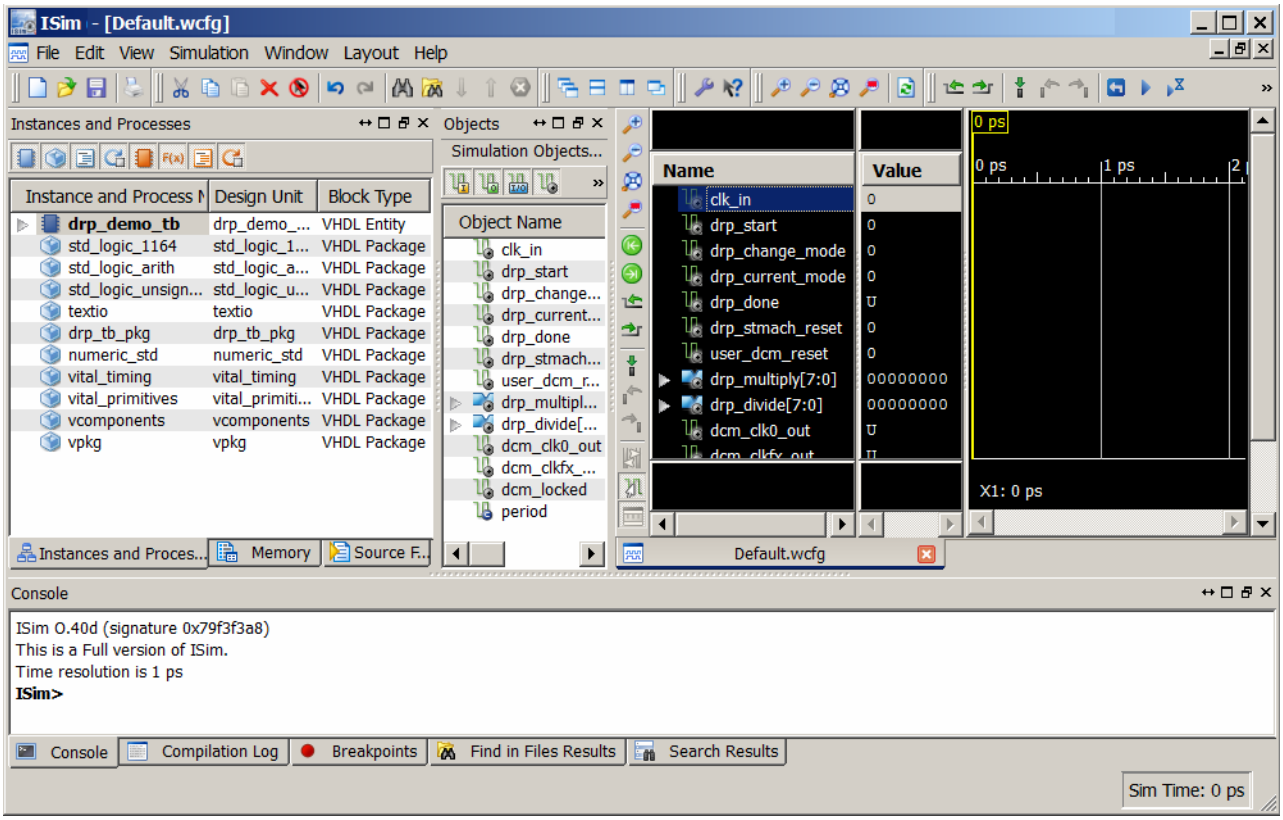


図 3-1 : ISim グラフィカル ユーザー インターフェイス

GUI コンポーネントの詳細は、『ISim ユーザー ガイド』(UG660) を参照してください。この資料へのリンクは、付録 A 「その他のリソース」に含まれています。

ISim GUI の説明

ツールバー

ISim のメイン ウィンドウで提供されているツールバーからは、さまざまな機能を実行できます。各ツールバーから、メニューでよく使用するコマンドにアクセスできます。GUI コンポーネントの詳細は、『ISim ユーザー ガイド』(UG660) を参照してください。この資料へのリンクは、付録 A 「その他のリソース」に含まれています。

メイン ウィンドウのツールバー アイコンは、ISim GUI 上部に配置されています。図 3-2 に、ツールバー アイコンを示します。ツールバーはカスタマイズ可能で、必要なツールバーのみを表示できます。

図 3-2 : 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。インスタンシエート、エラーボレートされたエンティティ (VHDL) およびモジュール (Verilog) が、ポート、信号、およびクロック コンポーネントと共にツリー構造で表示されます (図 3-3)。

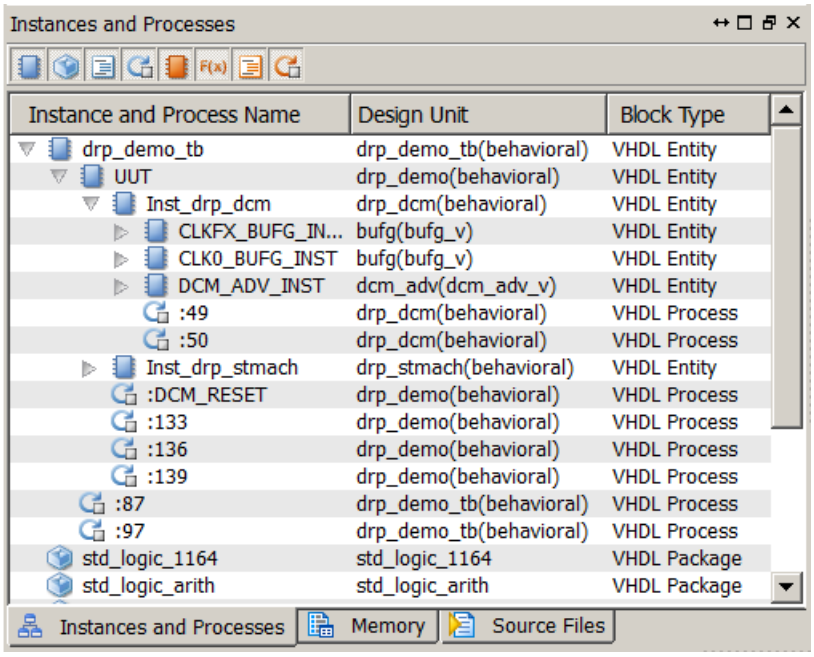


図 3-3 : [Instances and Processes] パネル

[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。`fuse` コマンドにより、デザインの解析およびエラボレーション ファイルのリストが供給されます。この処理は、GUI ではバックグラウンドで実行されます。HDL ソース ファイルのソース コードは、ISim 内で開くことができます。図 3-4 に [Source Files] パネルを示します。

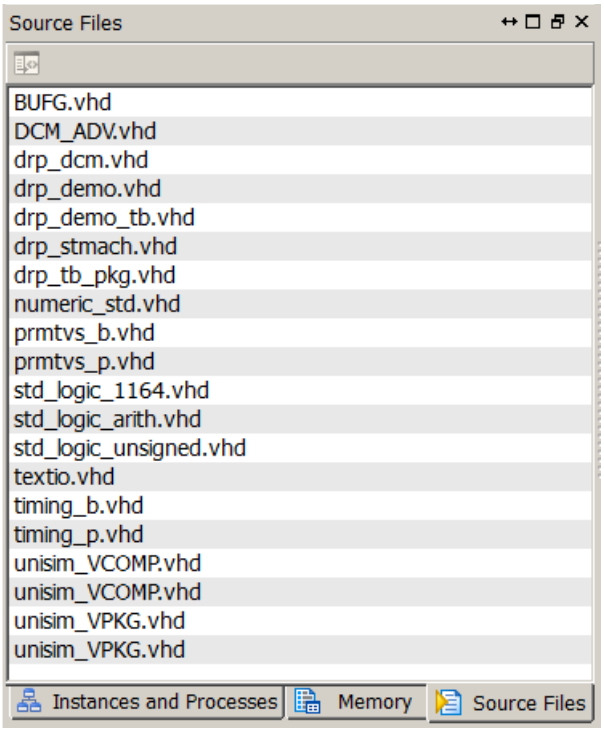


図 3-4 : [Source Files] パネル

[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表示されます。

パネルの上部には [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。

[Objects] パネルの表には、次の列があります。

- [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 がオンであるかオフであるかに基づき、シミュレーション時間またはメイン カーソルの位置の信号値を表示します。
- [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型を表示します。

22 ページの図 3-5 に [Objects] パネルを示します。

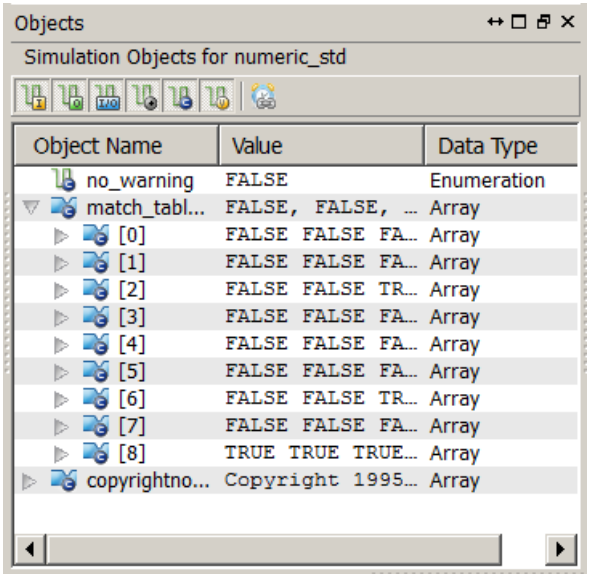


図 3-5 : [Objects] パネル

波形ウィンドウ

波形ウィンドウ (図 3-6) 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でシミュレーションを実行し、シミュレーション結果を検証します。

デザインおよびシミュレーション データはフラット ファイル 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、波形コンフィギュレーション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。

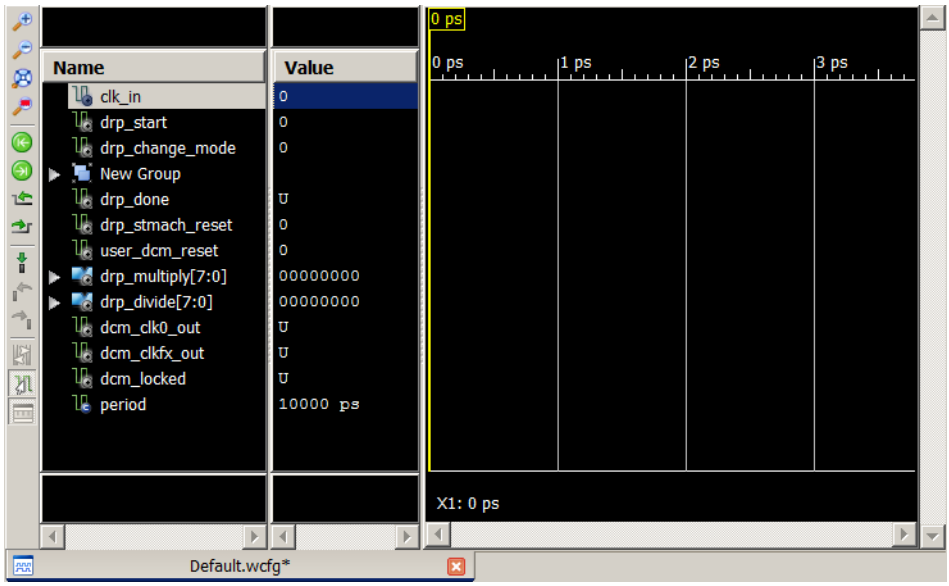


図 3-6 : 波形ウィンドウ

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 (図 3-7) では、シミュレーションで使用される HDL ソース ファイルを表示できます。テキスト エディターでは、次を実行できます。

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

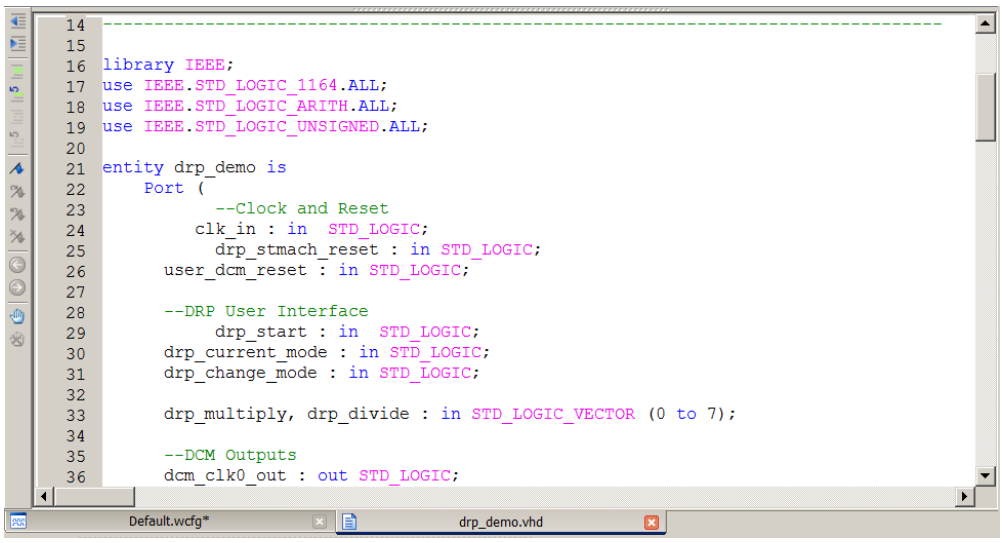


図 3-7 : テキスト エディター ウィンドウ

[Breakpoints] パネル

[Breakpoints] パネル (図 3-8) では、デザインに現在設定されているブレークポイントがリスト表示されます。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』(UG660) の第 8 章「デバッグ」を参照してください。この資料へのリンクは、付録 A 「その他のリソース」に含まれています。

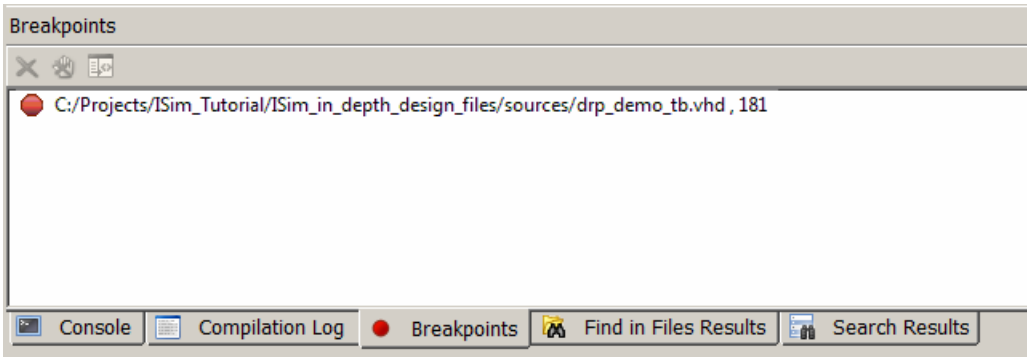


図 3-8 : [Breakpoints] パネル

[Console] パネル

[Console] パネル (図 3-9) では、ISim で生成されるメッセージを確認し、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。

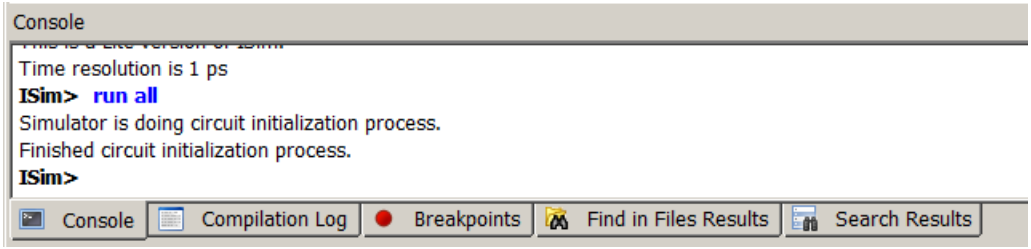


図 3-9 : [Console] パネル

デザインの確認およびデバッグ

このセクションでは、次の操作を実行してチュートリアル デザインの論理動作を確認します。

- 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ため、グループや仕切りを追加します。
- 信号および波形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

注記：第 2 章「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 オブジェクトはすべて波形ウィンドウに追加されています。

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。テストベンチでこれらの信号をアサートおよび監視し、DCM の DRP 機能を確認および制御します。
- DCM 出力信号 (`dcm_*`) : DCM の出力クロックです。

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 ユニットを右クリックします。
2. [Add to Wave Window] をクリックします。

   注記：インスタンスを選択して波形ウィンドウにドラッグすることも可能です。

   rp_demo_tb テストベンチのシミュレーション オブジェクトが波形ウィンドウに表示されます (図 3-10)。

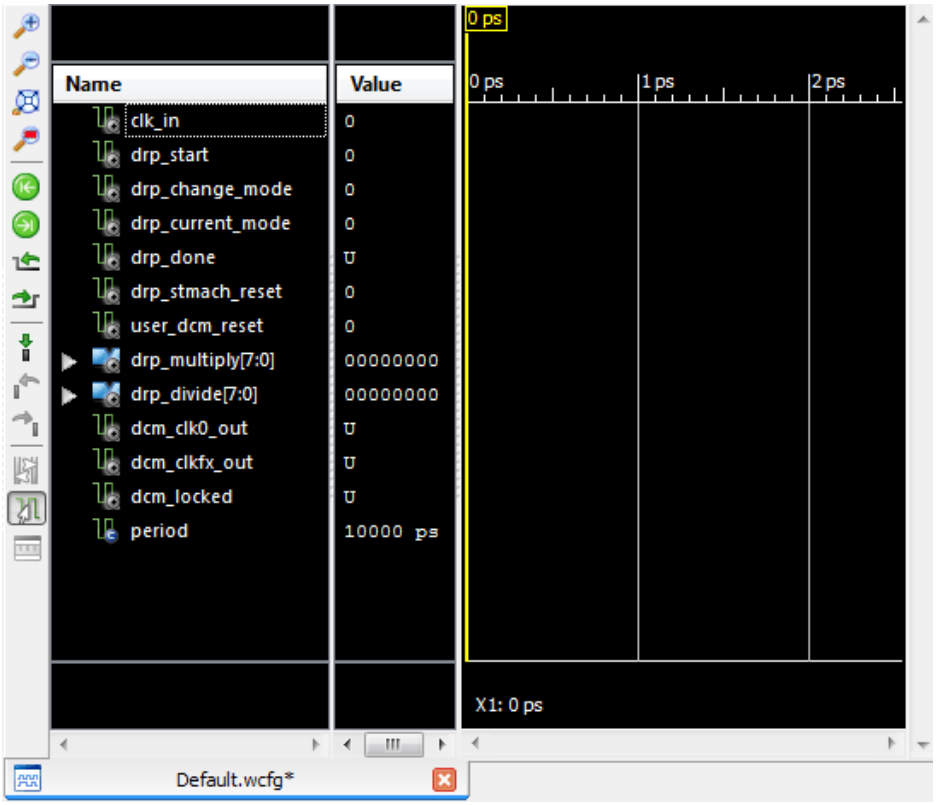


図 3-10 : 波形ウィンドウ

指定時間シミュレーションを実行

シミュレーションを指定した時間 (5 マイクロ秒 (µs)) だけ実行します。

1. ISim ツールバーにある [Run for the time specified on the toolbar] の横にあるボックスに「5µs」と入力し、[Run for the time specified on the toolbar] ボタン をクリックします。

   注記：Tcl プロンプトに「**run 5 µs**」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、シミュレーション時間 5µs までの信号のトレースが表示されます (図 3-11)。

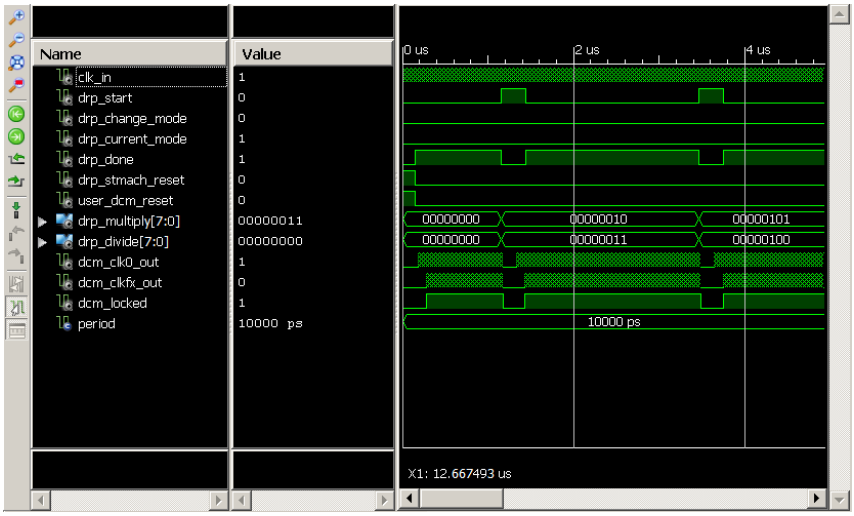


図 3-11 : 波形ウィンドウ

2. 波形ウィンドウに実行されたシミュレーションの波形全体を表示するには、[View] → [Zoom] → [To Full View] をクリックするか、またはツールバーの [Zoom To Full View] ボタンをクリックします。

   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。

   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。

3. [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 3-12)。

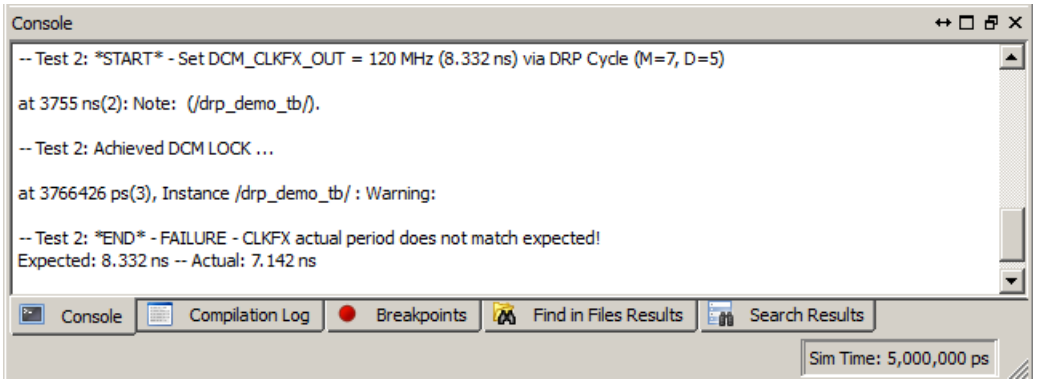


図 3-12 : [Console] パネル

Test 2 が予測どおりエラーになっていることを確認してください。この問題をチュートリアルの後半で修正します。

シミュレーションの再スタート

次に、[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ます。 

再スタートすると波形ウィンドウがクリアされ、シミュレーション時間が 0ps に設定されます。

注記：Tcl 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 3-13 のように表示されます。

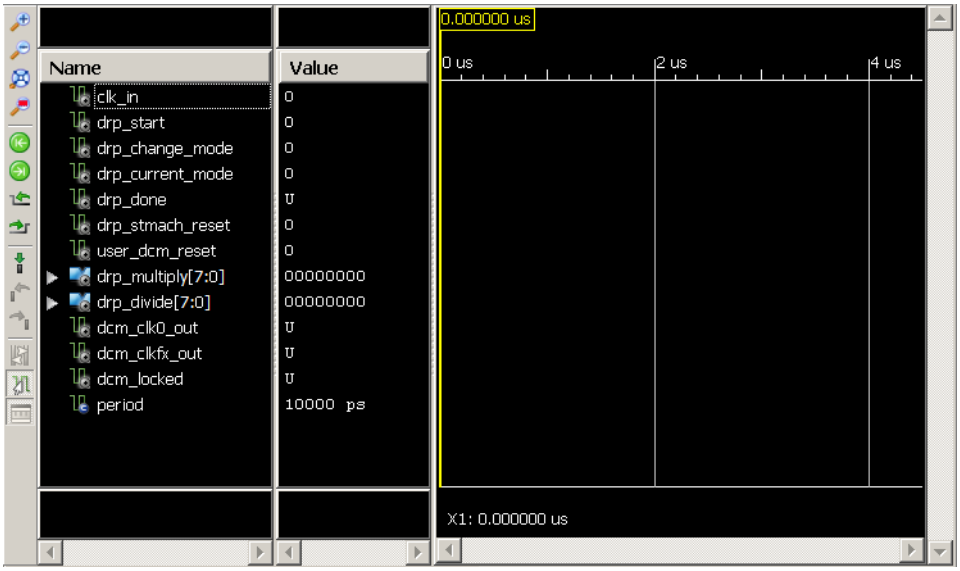


図 3-13：波形ウィンドウ

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

次に、このデザインの機能をより詳細に解析するため、別のデザイン ユニットの信号を追加します。

波形ウィンドウに信号を多く追加すると、すべての信号が波形ウィンドウのサイズ内に収まらなくなります。すべての信号を確認するのに波形ウィンドウの垂直方向スクロール バーを使用することになるため、確認作業が面倒になります。

信号をグループにまとめることで、表示環境を向上できます。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 キーを押しながら、`drp_demo_tb` デザイン ユニットに含まれている `drp_` で始まる信号をすべて選択します。
2. 選択した信号を右クリックし、[New Group] をクリックします。
3. 新しいグループの名前を入力します。ここでは、「DRP Test Signals」という名前を付けます。

   波形ウィンドウに階層が閉じた状態でグループが作成されます。

4. グループの階層を展開するには、グループ名の左側のプラス記号をクリックします。
5. デザイン ユニット `drp_demo_tb` に含まれる「`dcm_`」で始まるすべての信号を含むグループを作成します。
6. このグループに「DCM Test Signals」という名前を付けます。
7. 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 3-14 のように表示されます。

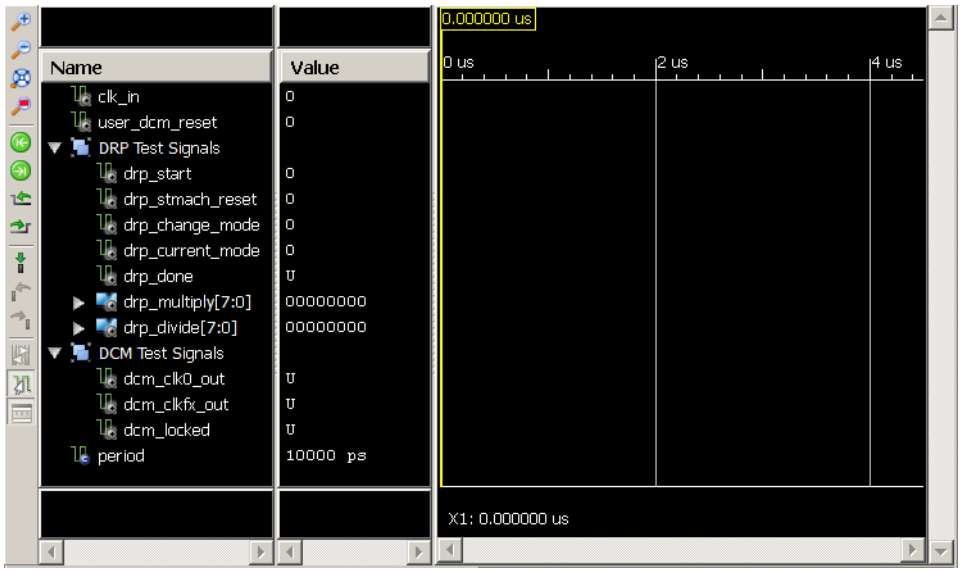


図 3-14 : グループの追加

信号グループが図 3-14 に示されるもの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てしまった場合は、切り取り/貼り付けを使用して信号をメイン リストに移動します。
- グループに含めるはずの信号がメイン リストに残っている場合は、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。
- [Edit] → [Undo] をクリックすると、グループ追加操作を取り消すことができます。
- グループを右クリックして [Ungroup] をクリックすると、グループを解除して作り直すことができます。

仕切りの追加

信号がどのデザイン ユニットに属するかをわかりやすく表示するため、デザイン ユニットごとに信号を分ける仕切りを追加します。

波形ウィンドウに仕切りを追加するには、波形ウィンドウの任意の場所を右クリックして [New Divider] をクリックし、仕切り名を入力します。

1. 次の 3 つの仕切りを追加します。
   - TEST BENCH
   - DCM
   - DRP CONTROLLER
2. [TEST BENCH] をクリックしてリストの一番上にドラッグします。
3. その他の仕切りは、リストの最後に移動します。

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

波形ウィンドウは、図 3-15 のように表示されます。

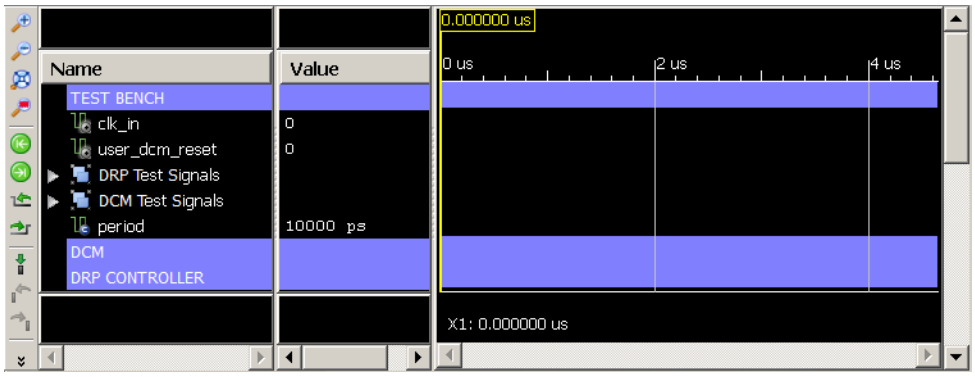


図 3-15：仕切りの追加

サブモジュールからの信号の追加

このセクションでは、インスタンシエートされている 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_stmach`) の信号を追加して、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ます。信号をグループに追加する最も簡単な方法は、信号をフィルターを使用して選択する方法です。

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 3-16 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。

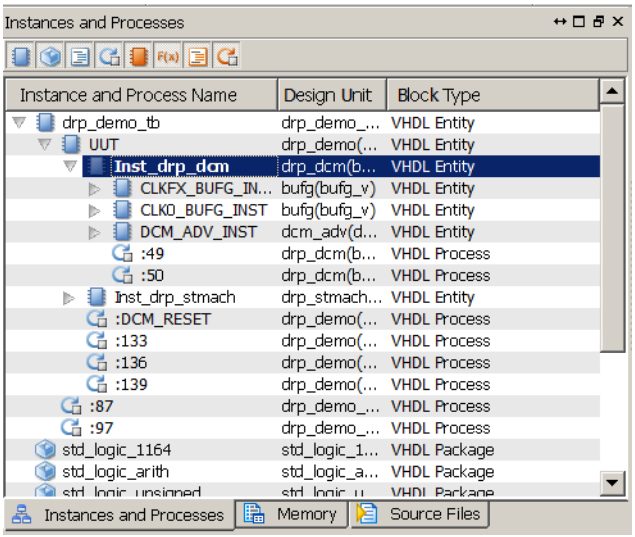


図 3-16 : [Instances and Processes] パネル

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 3-17)。

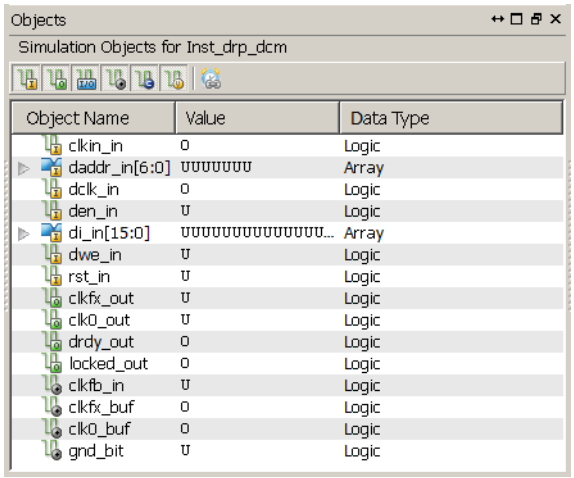


図 3-17 : [Objects] パネル

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、32 ページの図 3-18 のように示されます。

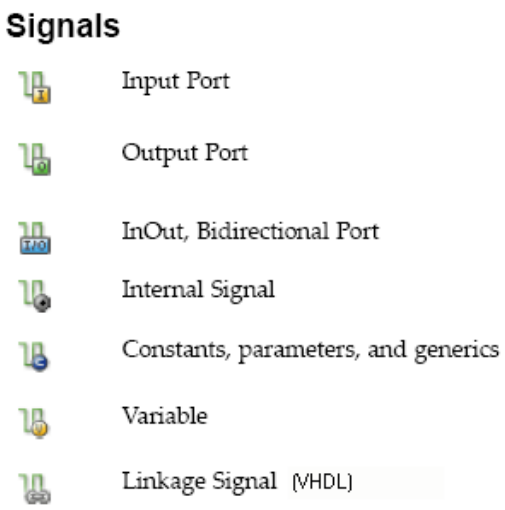


図 3-18 : 信号およびアイコン

このパネルでは表示するシミュレーション オブジェクトの種類をフィルターできます。このパネルのツールバーをクリックすると、入力、出力、双方向、内部、定数、および変数の表示/非表示をそれぞれ切り替えることができます。

3. オブジェクトの種類に対応するボタンをクリックしてオン/オフを切り替えます (図 3-19)。

図 3-19 : [Objects] パネルのツールバー ボタン

4. [Instances and Processes] パネルで `Inst_drp_dcm` デザイン ユニットを選択した状態で、[Objects] パネルのツール バーでボタンをクリックし、入力ポート、出力ポート、および内部信号が表示されるようにします。
5. [Objects] パネルに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。

   注記 : 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
   ```

6. 同様に、[DRP CONTROLLER] 仕切りの下にインスタンシエート済みデザイン ユニット `Inst_drp_stmach` の入力ポート、出力ポート、および内部信号を追加します。
7. 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 3-20 のように表示されます。

図 3-20 : 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、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

波形ウィンドウの信号名は、デフォルトでは階層を省いた短い名前で表示されます。信号によっては、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 Crtl 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。

信号の基数の変更

信号によっては、2 進数より 16 進数で表示したほうが認識しやすくなるものもあります。drp_multiply および drp_divide 信号はその例です。

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

波形ウィンドウに表示される信号の色を変更し、類似する信号を見分けやすくすることができます。

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 3-21)。

図 3-21 : 信号の表示色の変更

注記：[Divider Color] 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、波形ウィンドウに表示できる以上の信号が含められている場合があります。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場合、図 3-22 のように表示されます。

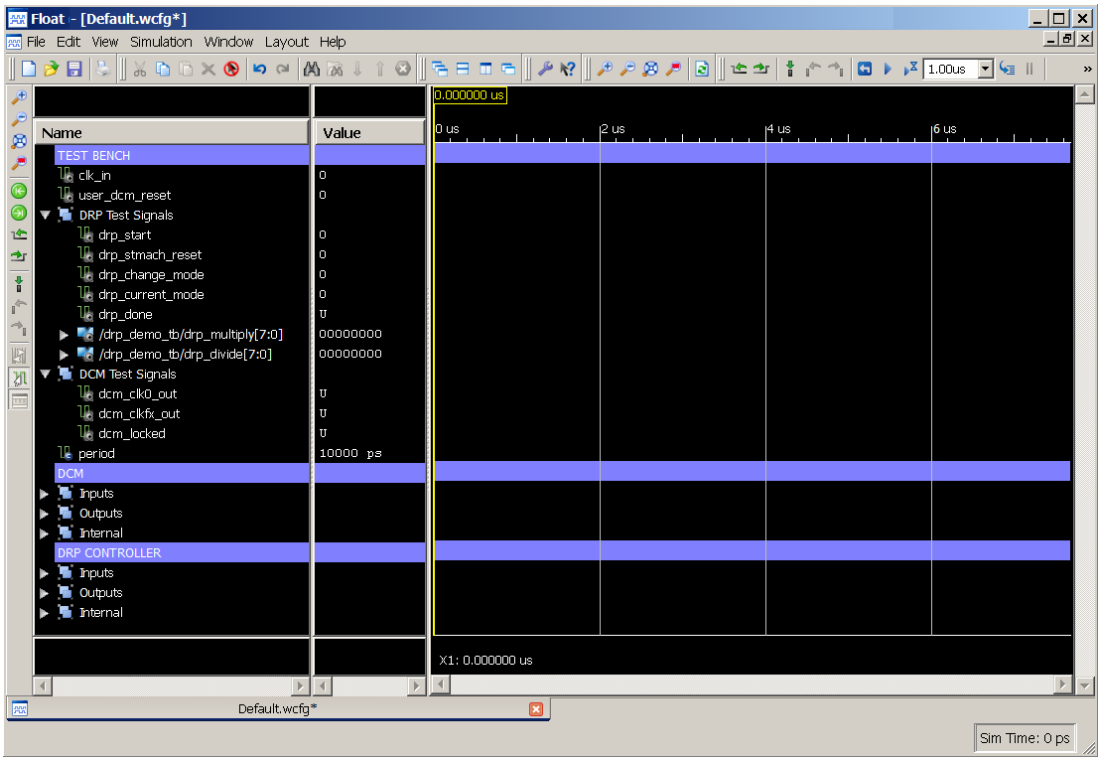


図 3-22 : 完全に設定されたフロート波形ウィンドウ

波形ウィンドウの設定の保存

今後の ISim シミュレーション セッションで使用できるよう、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。
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。

注記 : 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。

デザインの再シミュレーション

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。

注記 : 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約 13 マイクロ秒 (µs) 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 3-23 のように表示されます。

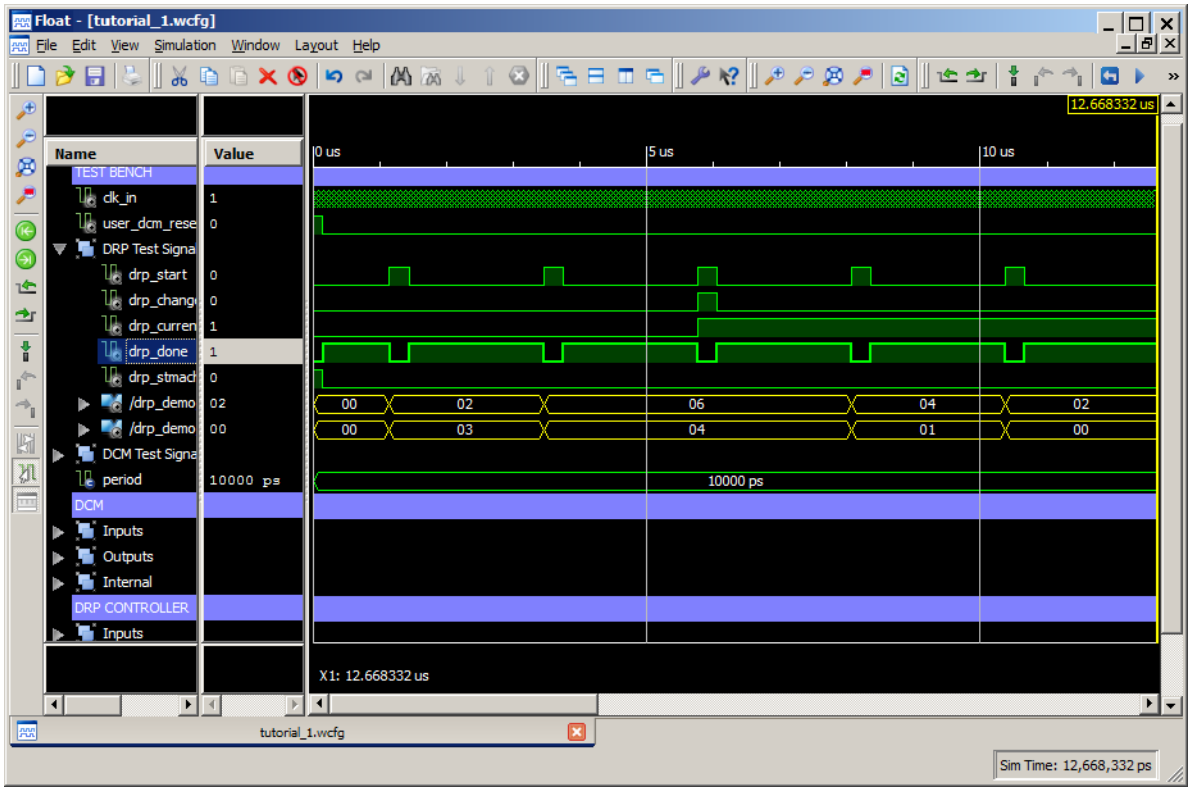


図 3-23 : 13µs シミュレーション時間後の波形ウィンドウ

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示す 4 つのテストが実行されます。

次の手順に従い、各テストの開始点にマーカーを追加します。

1. [Console] パネルで各テストの開始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 図 3-24 に示すように約 1,150ns で開始しています。

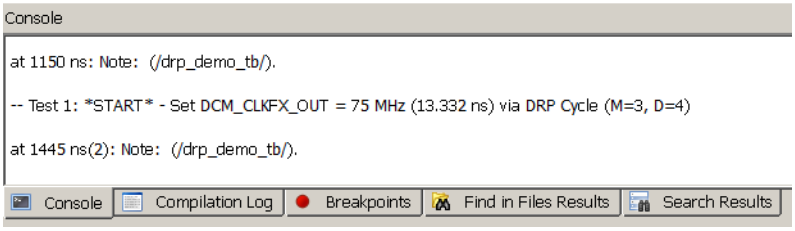


図 3-24 : [Console] パネル

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。

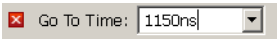


図 3-25 : [Go To Time] フィールド

3. ツールバーの [Add Marker] ボタンをクリックします。 

   注記 : 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、選択した計測単位合わせて変更されます。

4. [Conlsole] パネルでテストベンチで実行された 4 つのテストの開始時間を確認し、そこにマーカーを追加します。波形ウィンドウは、図 3-26 のように表示されます。

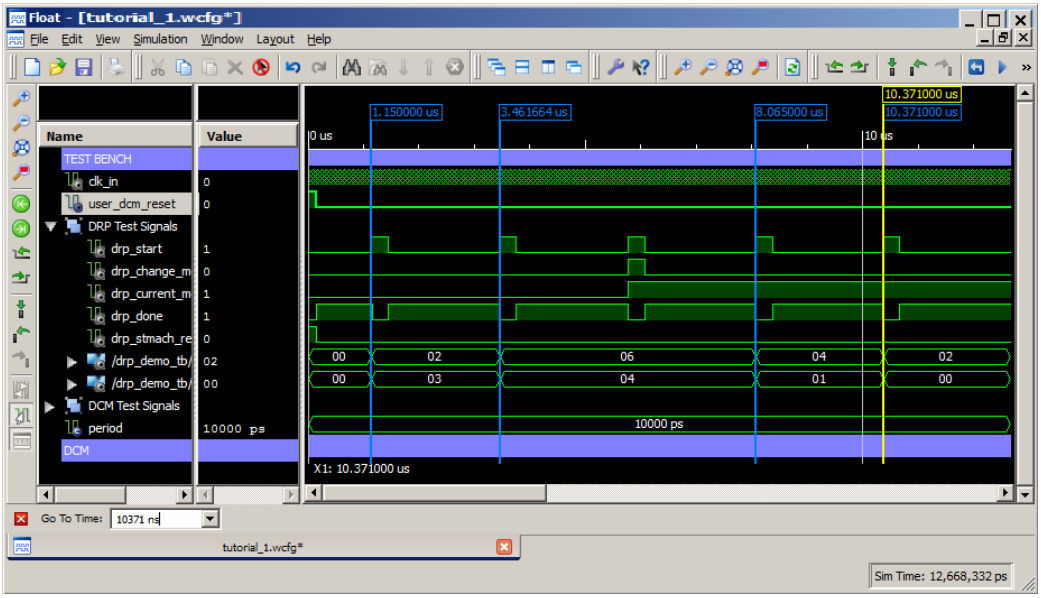


図 3-26 : テストの開始点をマーカーで識別

カーソルの使用

ISim の [Console] パネルには、テスト 2 とテスト 4 がエラーになったことがレポートされています (図 3-27)。

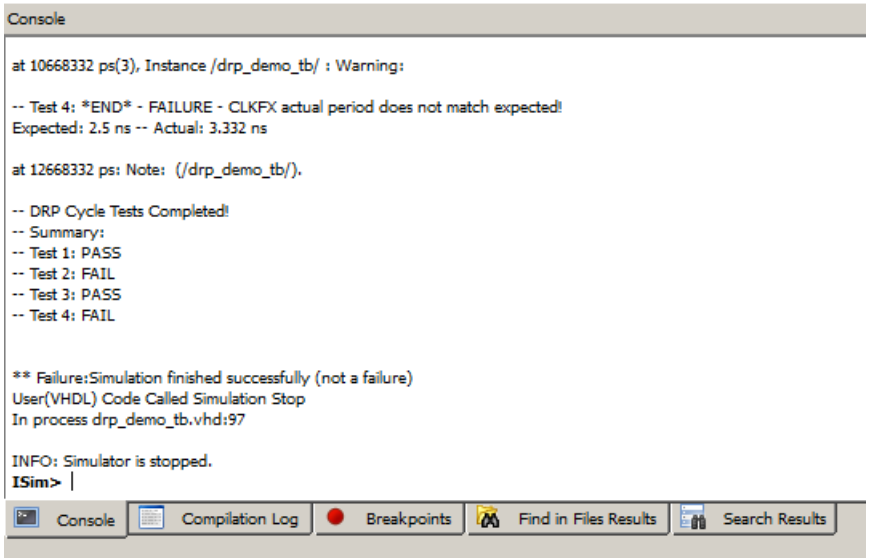


図 3-27 : [Console] パネル レポート : テスト 2 およびテスト 4 でエラーが発生

テスト 2 およびテスト 4 では、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、デジタル周波数合成の逓倍率および分周率が変更されてクロック出力 (CLKFX) の新しい周波数 (テスト 2 では 120MHz、テスト 4 では 400MHz) が設定されます。

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測される周期と一致しないことが判明しました。テスト 2 とテスト 4 は、周期の不一致が原因でエラーとなっています。テスト 2 のエラーを、図 3-28 に示します。

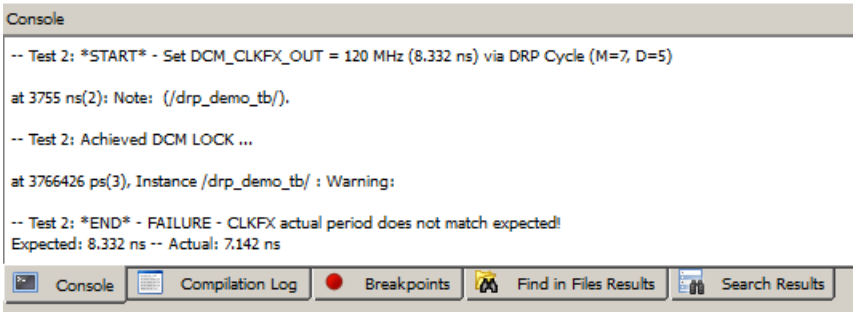


図 3-28 : テスト 2 では周期の不一致が原因でエラーが発生

次の手順では、ISim のメイン カーソル (黄色) を使用して、波形ウィンドウでエラーが発生した時点の波形を拡大表示します。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どうかを確認します。

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、テスト 2 の開始点を示すマーカー (3.461664ms のマーカー) の近くにドラッグします。カーソルがマーカーに揃えられます。

   注記 : 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の配置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ようになるまで、手順 2 を繰り返します (図 3-29)。

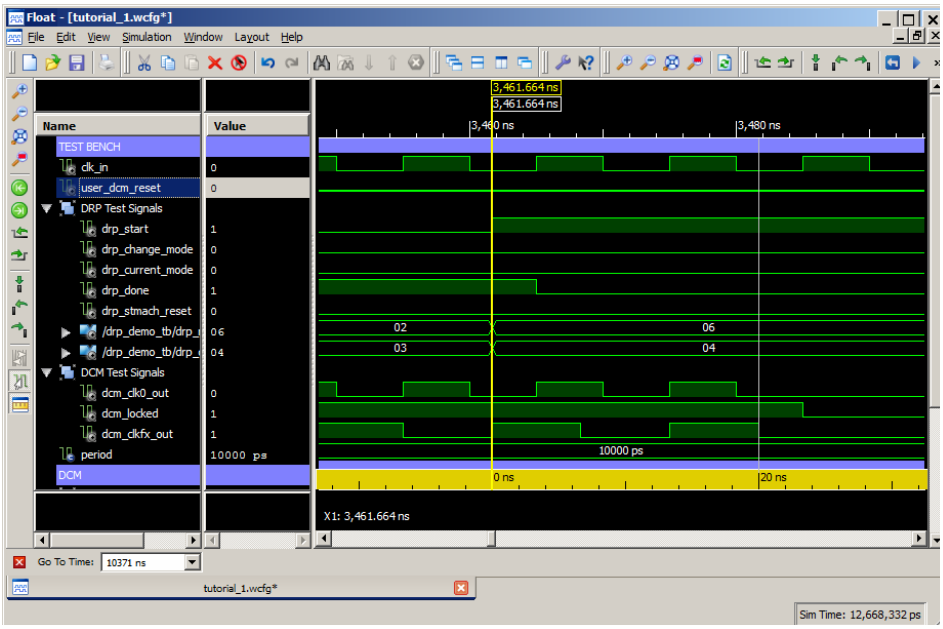


図 3-29 : dcm_clk0_out 信号および dcm_clkfx_out 信号の確認

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。この機能を使用して、DRP サイクルが完了した後 (drp_done 信号がアサート) の dcm_clkfx_out の周期を計測し、[Console] パネルにレポートされたテスト 2 で計測された値を確認します。

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 をオンにし、カーソルが遷移エッジに合わせられるようにします。
2. DRP サイクル完了後 (drp_done 信号のアサート後) のクロックの最初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. マウス ボタンを押したまま、マウスをクロックの次の立ち上がりエッジに移動します。

   2 つ目の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の下に時間デルタとして表示されます (図 3-30)。

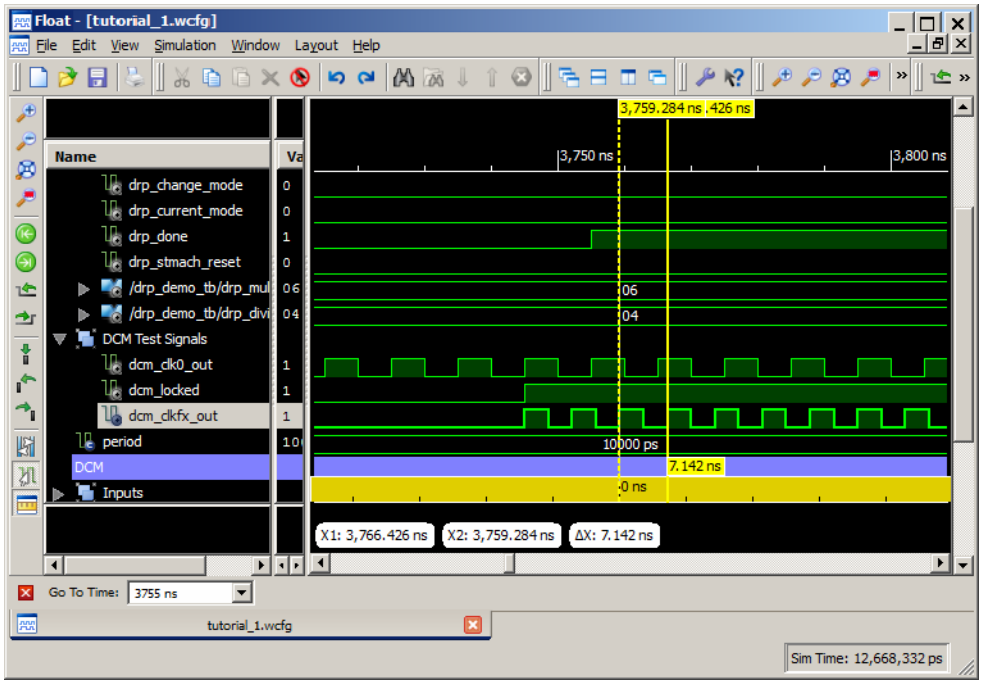


図 3-30 : 波形ウィンドウでの時間計測

   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 であることが示されます。つまり、140MHz のクロック信号です。テスト 2 で予測される周波数は 120MHz であるため、エラーになります。

4. 同様の手順でテスト 4 を解析します。テストベンチで予測される周波数は 400MHz であるのに対し、計測される周波数は 300MHz であることがわかります。

注記：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 ルーラーを表示できます。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。このルーラー上では、最初の時間エンドポイントがゼロ (0ps) になります。この機能は、最初のエンドポイントに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、それぞれに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。

   注記：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。

   インスタンスを選択して波形ウィンドウにドラッグすることも可能です。
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、[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 3-31 および 41 ページの図 3-32 に示す 2 つの波形ウィンドウが表示されます。

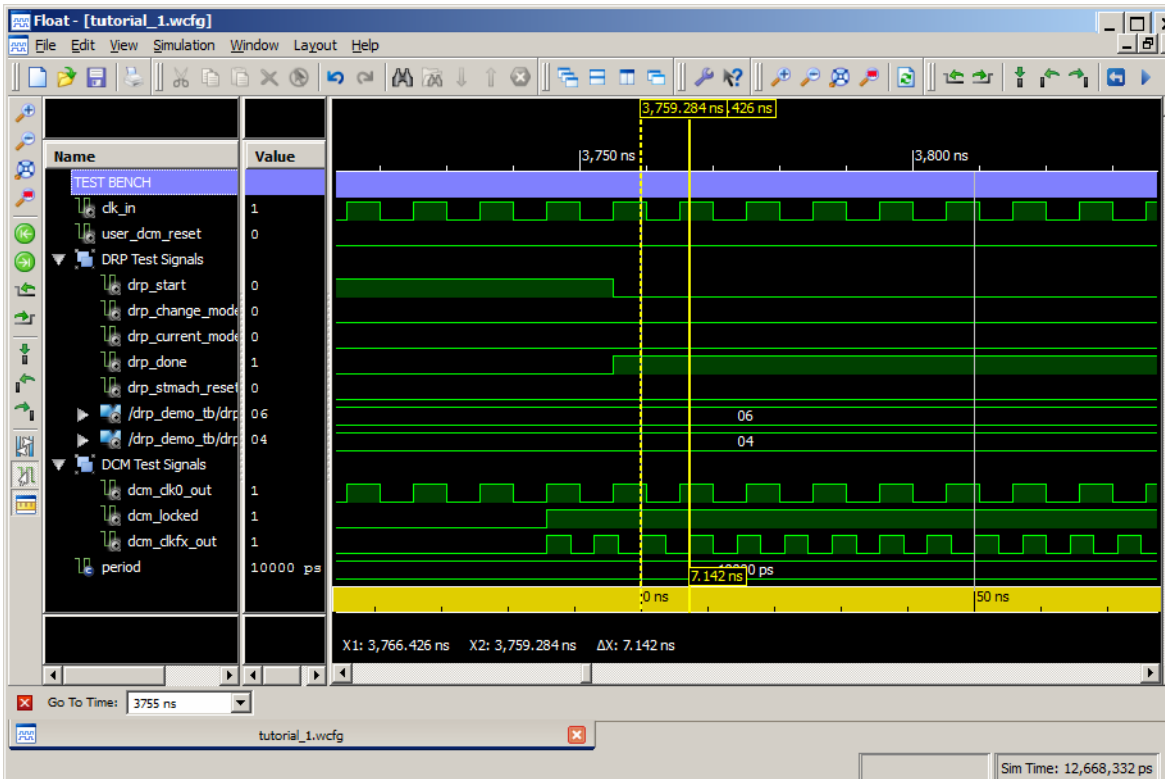


図 3-31 : tutorial_1.wcfg

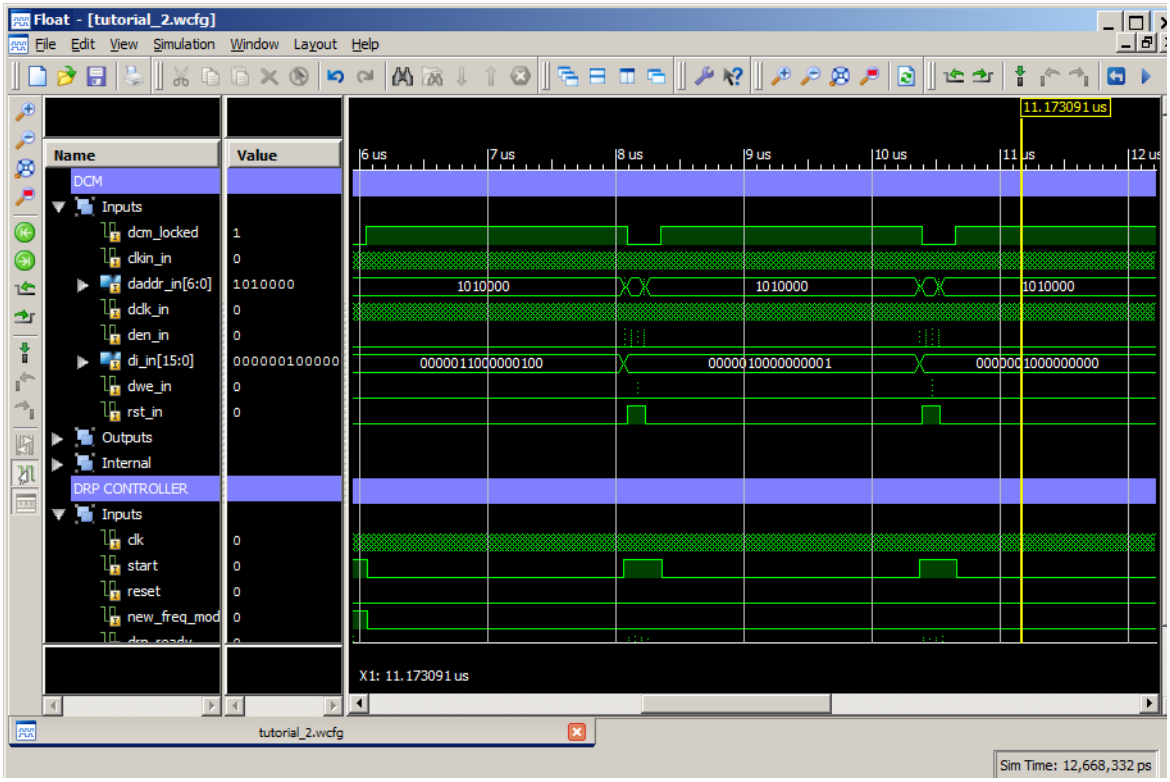


図 3-32 : tutorial_2.wcfg

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ました。

次に、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の ISim のデバッグ機能を使用してデザインをデバッグし、エラーが発生した 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を学びます。

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 エディターで開きます (42 ページの図 3-33)。

- [File] → [Open] をクリックし、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、[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 オブジェクトのいずれかを右クリックし、[Go to Source Code] をクリックします。
- [Source Files] パネルでソース ファイルをダブルクリックします。

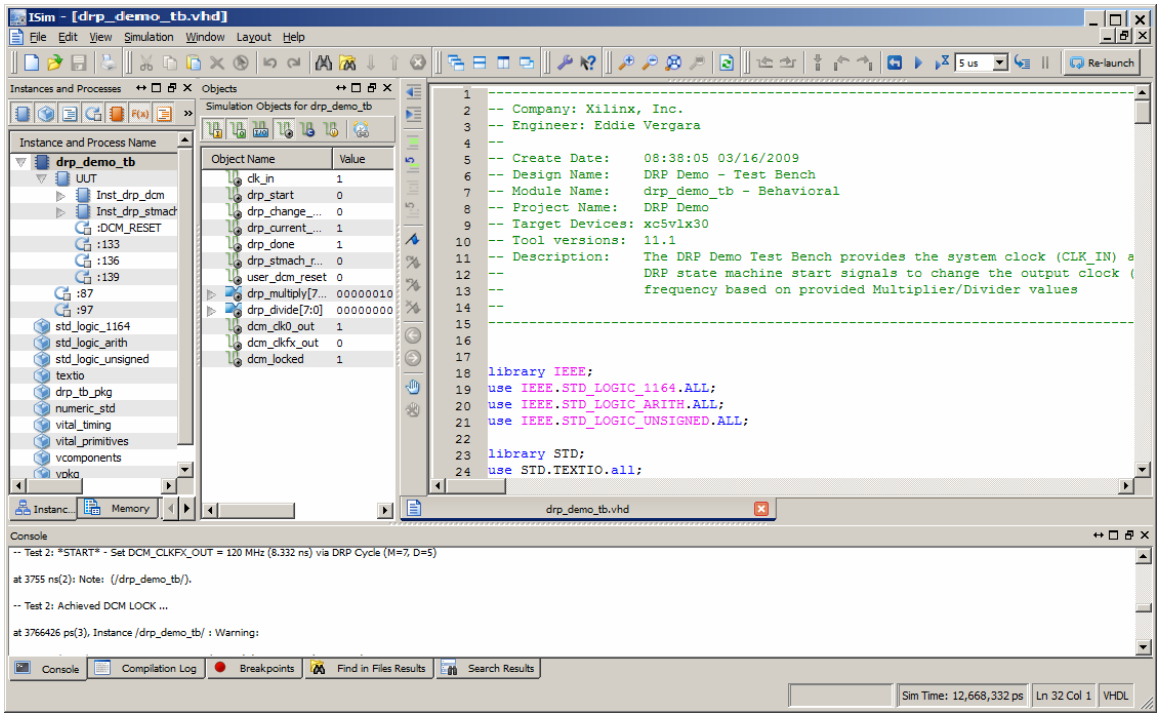


図 3-33 : 統合されたテキスト エディター

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、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ーが表示され、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

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。

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 (43 ページの図 3-35)。

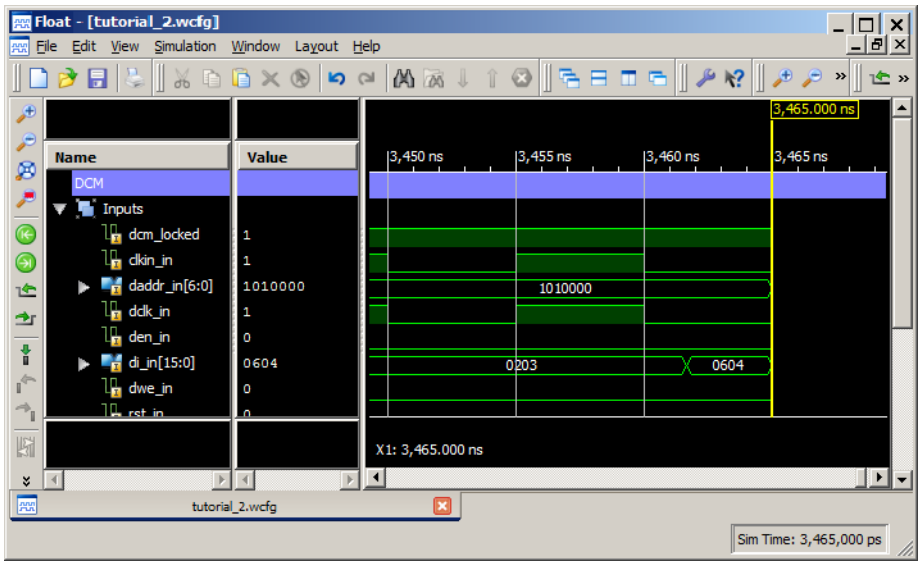


図 3-34 : 波形ウィンドウ : DCM DI_IN 入力バスの出力を確認

注記 : [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。

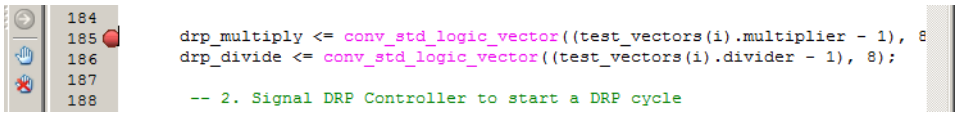


図 3-35 : drp_demo_tb.vhd の 185 行目にブレークポイントを設定

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が停止します。

[Console] パネルの横にある [Breakpoints] パネルをクリックすると、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 3-36)

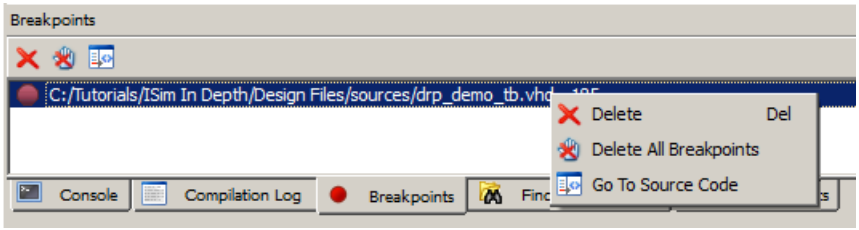


図 3-36 : [Breakpoints] パネルの右クリック メニュー

ブレー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、ブレー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。

ブレークポイント機能およびソース 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. [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ます。
2. [Run All] ボタンをクリックしてシミュレーションを実行します。

最初のテストの開始付近ま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 (図 3-37)。

```
183
184
185
186
187
        -- 1. Set Multiplier and Divider values from test vector

    drp_multiply <= conv_std_logic_vector((test_vectors(i).multiplier - 1), 8
    drp_divide <= conv_std_logic_vector((test_vectors(i).divider - 1), 8);
```

図 3-37 : 実行されたソース コードの最後の行

[Console] パネルには、シミュレータが停止したことを示すメッセージと、シミュレータ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。

テスト 1 が正しく終了することは既に確認されているので、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。

   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 3-38)。

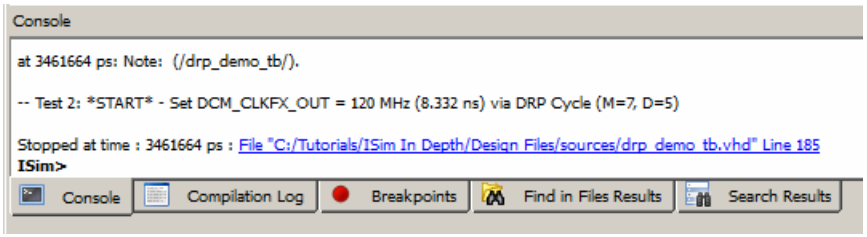


図 3-38 : シミュレータが停止したことを示す [Console] パネルのメッセージ

ソース コードの 1 行ずつの実行

テスト 2 で drp_multiply および drp_divide バス信号を介して逓倍率と分周率パラメーターが正しく設定されるかどうかを確認します。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。

次の手順に従い、シミュレーションを 1 行ずつ実行します。

1. [Step] ボタンをクリックします。

   注記：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。

2. 一行ずつソース コードを進めることで、次のイベントを確認します。
   - drp_multiply および drp_divide バス信号が定数 test_vectors から割り当てられている。
   - drp_start バス信号がアサートされ、DRP サイクルが開始する。
   - 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、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。
3. tutorial_2 波形ウィンドウで DCM 入力バスを展開表示します。
4. 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 に更新されるはずです (43 ページの図 3-34)。

   注記：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。

   このデザインの出力クロック周波数 (dcm_clkfx_out) は、逓倍率と分周率によって決まります。次に、テスト 2 で使用されるパラメーター値と予測される出力クロック周波数を示します。

   Table 3-1: パラメーター値と予測される出力クロック周波数

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

   M=6 および D=5 では、di_in[15:0] のバス値は 0504h になるはずですが、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 係数が不正であるからです。

5. 上記の手順をテスト 4 で繰り返して、エラーの原因を追求します。テスト 4 でも、テストベンチでの逓倍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逓倍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 およびテスト 4 で正しい逓倍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。

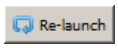
1. ISim メイン ウィンドウで [Source Files] パネルをクリックします。
2. drp_demo_tb.vhd ファイルを右クリックし、[Go To Source Code] をクリックします。
3. 4 つの DRP テストのテスト ベクターは、117 行目から 127 行目で定義されています。定数宣言を次のように変更します (変更は太文字で表記)。

```
-------------------------------------------------
-- ** 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。

修正の確認

テストベンチのソース コードを修正したので、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生成します。

1. [Breakpoints] パネルをクリックして、以前に設定したブレー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し直され、シミュレーションが読み込まれます。

   アップデート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。
3. [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。

   注記 : 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 3-39)。

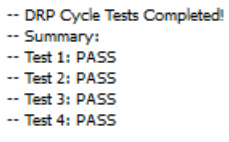


図 3-39 : すべてのテストが完了したことを示す [Console] パネル

まとめ

これで ISE® ISim チュートリアルが終了しました。このチュートリアルでは、次を実行しました。

- ISim を Project Navigator またはスタンドアロン モードで実行
  - Project Navigator では次を実行
    - プロジェクトの作成
    - プロジェクトへのソース ファイルの追加
    - VHDL ライブラリを作成してライブラリにファイルを移動
    - プロジェクト内でデザインをコンパイル
    - プロパティを設定してビヘイビアー シミュレーションを実行
    - ISim GUI で確認
    - デザインのデバッグ
    - エラー修正の確認
  - スタンドアロン モードの ISim で同じ操作をコマンド ライン モードで実行 (第 4 章「スタンドアロン ISim の実行」)

ISim の詳細は、付録 A 「その他のリソース」に含まれる資料へのリンクを参照してください。

XILINX®

第 4 章

スタンドアロン ISim の実行

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。

- ISim プロジェクト ファイルを手動で作成し、fuse コマンドを使用してシミュレーション実行ファイルを作成します。
- fuse コマンド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

デザイン ツール要件およびチュートリアル デザイン ファイルの入手先は、第 1 章「必要なデザイン ツール」および第 1 章「チュートリアル デザイン ファイルのインストール」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、ISim プロジェクト ファイルを手動で作成し、fuse コマンドでシミュレーション実行ファイルを作成します。この手順が完了すると、シミュレーション実行ファイルを実行して ISim GUI を起動できます。

ISim プロジェクト ファイルの作成

ISim プロジェクト ファイルの構文は、次のとおりです。

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

説明 :

- verilog|vhdl : ソース ファイルが Verilog であるか VHDL であるかを示します。Verilog または VHDL ソース ファイルを含めます。
- *<library_name>* : 指定行のソースをコンパイルするライブラリを指定します。デフォルトのライブラリは /work です。
- *<file_name>* : ライブラリに関連付けるソース ファイルを指定します。

注記 : 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 ファイルを作成します。

1. ダウンロードしたチュートリアル ファイルの /scripts フォルダーに移動します。
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. 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は、次のとおりです。

   - drp_dcm.vhd : VHDL ソース ファイル。work ライブラリで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 ライブラリでコンパイルする必要があります。

   注記 : 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 コマンド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 フォルダーに完成したプロジェクト ファイルが含まれており、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閉じます。

シミュレーション実行ファイルの生成

この手順では、前述のセクションで作成したプロジェクト ファイルを fuse コマンドで使用し、デザインのすべてのソースを解析、コンパイル、リンクします。この手順によりシミュレーション実行ファイルが生成され、ISim GUI でシミュレーションを実行できるようになります。

fuse コマンドの使用

次に、fuse の構文を示します。

```
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. チュートリアル ファイルの /scripts フォルダーに移動します。
2. テキスト エディターで fuse_batch.bat 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閉じます。
5. ISE のコマンド プロンプトで fuse_batch.bat ファイルに移動して実行し、fuse を起動します。

注記：ISE 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] → [ISE Design Suite コマンド プロンプト] をクリックします。

`fuse` コマンド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 コード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` フォルダーに含められます。

`/completed` フォルダーには完成した `fuse_batch.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してください。

手動でのデザインのシミュレーション

50 ページの「シミュレーション実行ファイルの生成」セクションで `fuse` コマンドを使用して、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

シミュレーション実行ファイルを起動するには、次の構文を使用します。

```
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。

シミュレーションの起動

シミュレーションを起動するには、次を実行します。

1. チュートリアル ファイルの `/scripts` フォルダーに移動します。
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

   注記：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、シミュレーション中に作成されます。
4. ファイルを保存して閉じます。
5. ISE のコマンド プロンプトで `simulate_isim.bat` ファイルに移動して実行し、シミュレータを起動します。

結果

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 フォルダーに完成した simulate_isim.bat バッチ ファイルが含まれており、作成したファイルと比較できます。

XILINX®

付録 A

その他のリソース

ザイリンクス リソース

この付録では、このチュートリアルで参照されているザイリンクス資料へのリンクを示します。

- 『ISE Design Suite : インストールおよびライセンス ガイド』(UG798)
  http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/iil.pdf
- 『ISE Design Suite : リリース ノート ガイド』(UG631) :
  http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/irn.pdf
- ザイリンクス用語集 :
  http://japan.xilinx.com/support/documentation/sw_manuals/glossary
- ザイリンクス製品サポートと資料 : http://japan.xilinx.com/support
- ISE® 資料 : http://japan.xilinx.com/support/documentation/dt_ise13-4.htm
- 『Virtex-5 FPGA ユーザー ガイド』(UG190) :
  http://japan.xilinx.com/support/documentation/virtex-5_user_guides.htm
- 『効率的なテストベンチの作成』(XAPP199) :
  http://www.xilinx.com/support/documentation/application_notes/xapp199.pdf
- 『ISim ユーザー ガイド』(UG660) :
  http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/plugin_ism.pdf

チュートリアル ページ :『ISim チュートリアル』のデザイン ファイル (ug682.zip)、その他の ISE® Design Suite チュートリアル、ISim ハードウェア協調シミュレーション 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_ise13-4_tutorials.htm