ISim ハードウェア協調シミュレーション チュートリアル : 浮動小数点高速フーリエ変換のシミュレーションの高速化

UG817 (v 13.2) 2011 年 7 月 28 日

XILINX®

本資料は英語版 (v.13.2) 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。

目次

第 1 章

概要

このチュートリアルでは、ISim ハードウェア協調シミュレーションを使用して、浮動小数点高速フーリエ変換 (FFT) のシミュレーションを高速化し、ザイリンクス ML605 ボード上で FFT インプリメンテーションを検証する方法を説明します。

DSP (Digital Signal Processing) デザインは、そのデータおよび計算量が多いため、ソフトウェアでシミュレーションすると非常に時間がかかります。DSP ファンクションのシミュレーションを高速化するため、高速ビット精度モデルがよく使用されますが、サイクル精度は提供されず、その他の RTL (Register Transfer Level) モジュールと統合するのも簡単ではありません。ビヘイビアー RTL モデルではビットおよびサイクル精度が提供されますが、シミュレーションが低速になります。構造 RTL またはゲート レベル モデルを使用すると、シミュレーション速度はさらに低下します。IP によっては高速ビット精度モデルは提供されておらず、ビヘイビアー RTL モデルがない場合もあり、構造/ゲート レベルのシミュレーションが唯一の方法となります。ISim ハードウェア協調シミュレーションでは、多量の計算を FPGA で実行させることにより、ソフトウェアの負担を軽減して DSP ファンクションのシミュレーションを実行できます。合成可能な HDL コード、CORE Generator™ で生成された IP コアなどの合成済みまたは保護されたネットリストを、協調シミュレーション用に FPGA に読み込むことができます。これにより、ビットおよびサイクル精度シミュレーション モデルを使用する必要はなく、シミュレーション パフォーマンスを向上できます。複雑な DSP デザインの多くでは、デザインのシミュレーションが高速化されるだけでなく、実際のハードウェア上でのデザインのインプリメンテーションを検証できます。ISim ハードウェア協調シミュレーションは、RTL シミュレーション、合成後のシミュレーション、インプリメンテーション後のシミュレーションを補足するものです。

要件

- ISE® Design Suite バージョン 13.2
- Virtex®-6 FPGA ML605 評価キット
- デザイン ファイル : rdf0125_fft_sim_tutorial.zip

チュートリアル ファイル

ファイル	説明
fp_fft_top.v	浮動小数点 FFT コアをインスタンシエートするラッパー
fp_fft_tb.v	FFT を実行するためのテスト ベクターを生成する最上位テスト ベンチ
FloatingPointFFT.xise	このチュートリアル用の ISE® プロジェクト
sim.tcl	ISim のシミュレーション時間を計測するカスタム シミュレーション コマンド ファイル
fp_fft_tb.wcfg	カスタム波形コンフィギュレーション ファイル
fp_fft_tb.prj	コマンド ライン フロー用の ISim プロジェクト ファイル
full_compile.bat	Fuse コマンド ラインを使用してハードウェア協調シミュレーション用にデザインを完全にコンパイルする Windows バッチ ファイル
full_compile.sh	Fuse コマンド ラインを使用してハードウェア協調シミュレーション用にデザインを完全にコンパイルする Linux シェル スクリプト
incr_compile.bat	Fuse コマンド ラインを使用してハードウェア協調シミュレーション用にテストベンチをインクリメンタルにコンパイルする Windows バッチ ファイル
incr_compile.sh	Fuse コマンド ラインを使用してハードウェア協調シミュレーション用にテストベンチをインクリメンタルにコンパイルする Linux シェル スクリプト
run_isim.bat	ISim シミュレーションを起動する Windows バッチ ファイル
run_isim.sh	ISim シミュレーションを起動する Linux シェル スクリプト

メモ : このチュートリアルを実行する際は、すべてのデータ ファイルを作業ディレクトリにコピーしてください。

XILINX®

第 2 章

チュートリアル

このチュートリアルでは、ISim ハードウェア協調シミュレーションを使用して、浮動小数点高速フーリエ変換 (FFT) のシミュレーションを高速化し、ザイリンクス ML605 ボード上で FFT インプリメンテーションを検証する方法を説明します。

4 つのセクションに分かれており、ISim ハードウェア協調シミュレーションを使用して FFT デザインを実行するのに必要な手順を示します。手順は順番に実行してください。このチュートリアルは、次のセクションから構成されています。

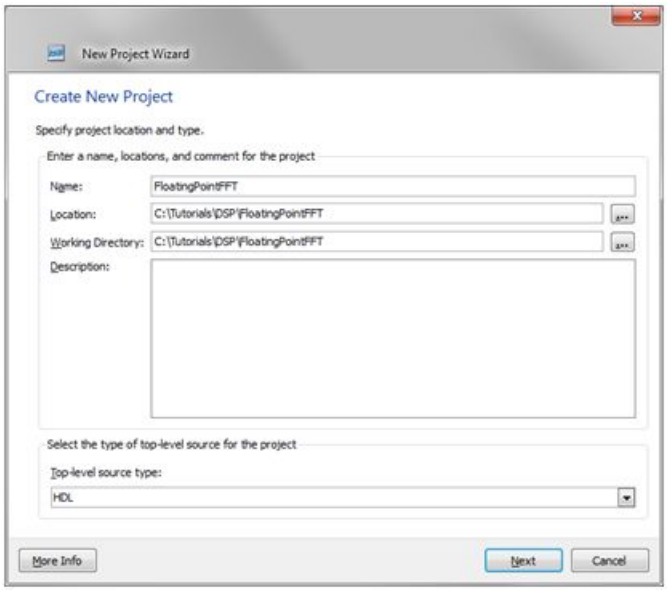
1. CORE Generator™ での FFT コアの生成
2. テストベンチの作成
3. ハードウェア協調シミュレーション用のデザインのコンパイル
4. ISim ハードウェア協調シミュレーションの実行

手順 1 : CORE Generator での FFT コアの生成

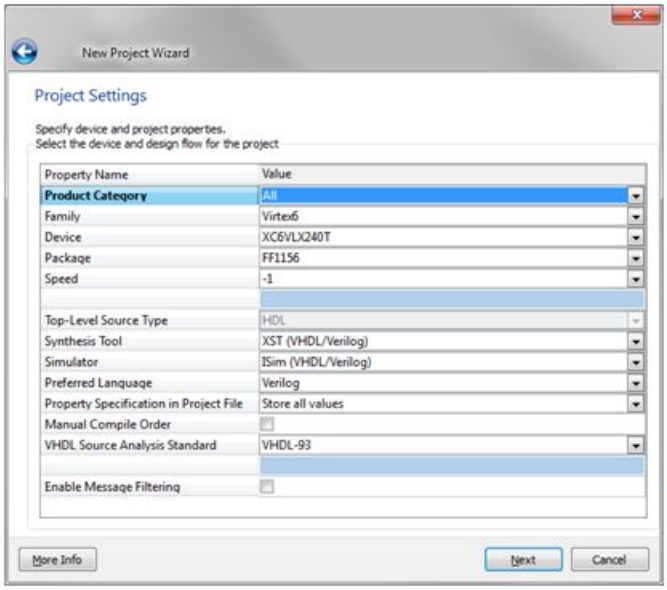
このチュートリアルでは、CORE Generator ツールの Fast Fourier Transform IP コアを使用し、Virtex®-6 FPGA ML605 評価キットで動作する ISim ハードウェア協調シミュレーション テストベンチを作成します。

メモ : このチュートリアルのスクリーン ショットは、Fast Fourier Transform v7.1 のものです。これ以外のバージョンでは、CORE Generator GUI が異なる場合があります。

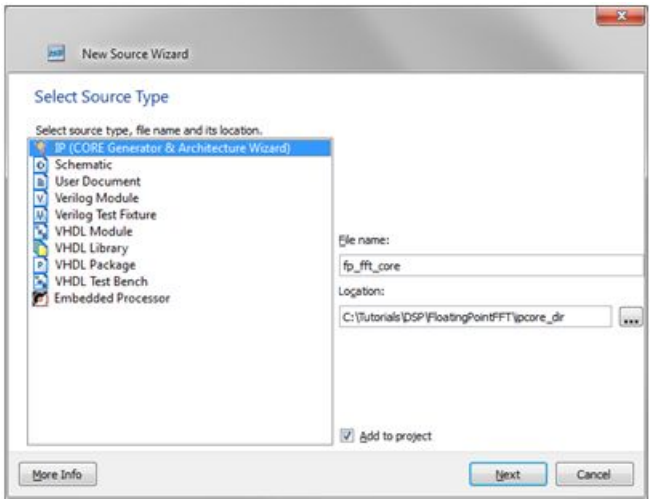
1. ISE® Project Navigator を起動します。
2. [File] → [New Project] をクリックし、New Project Wizard を開きます。プロジェクト名 (FloatingPointFFT) と保存ディレクトリを入力します。[Next] をクリックします。



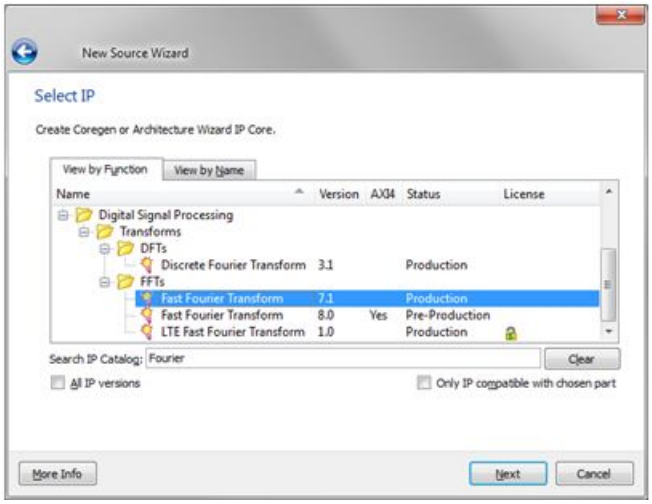
3. [Project Settings] ページで、[Family] に [Virtex6]、[Device] に [XC6VLX240T] (ML605 ボードの Virtex®-6 デバイス)、[Package] に [FF1156]、[Speed] に [-1] を選択します。[Simulator] に [ISim]、[Preferred Language] に [Verilog] を選択します。[Next] をクリックして [Project Summary] ページで [Finish] をクリックし、プロジェクトの作成を完了します。



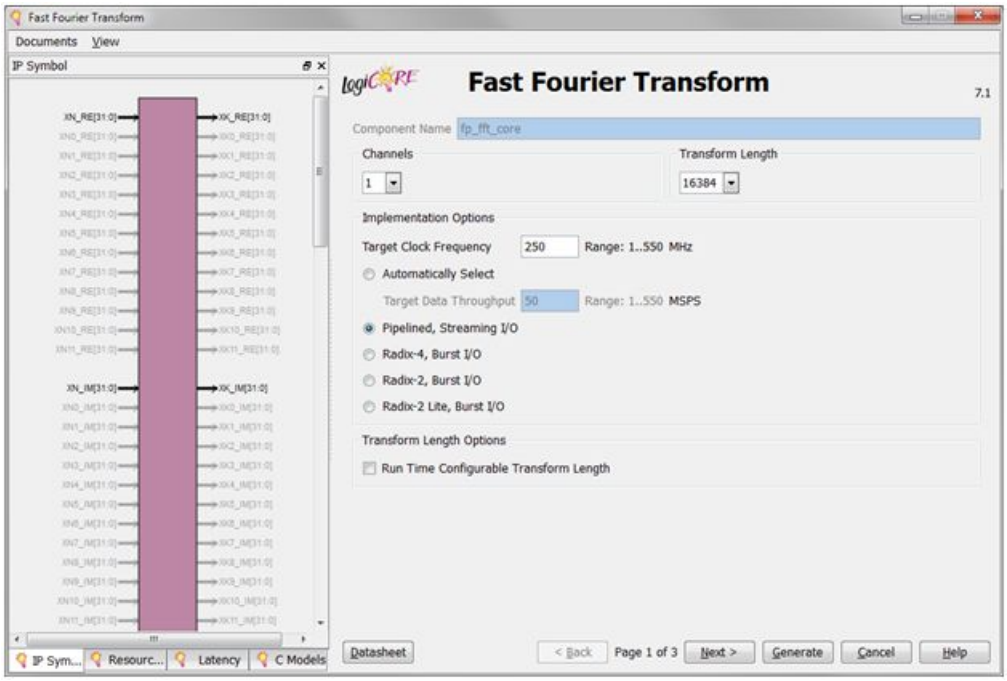
4. [Project] → [New Source] をクリックし、New Source Wizard を開きます。[IP (CORE Generator & Architecture Wizard)] を選択し、[File name] に「fp_fft_core」と入力します。[Next] をクリックします。



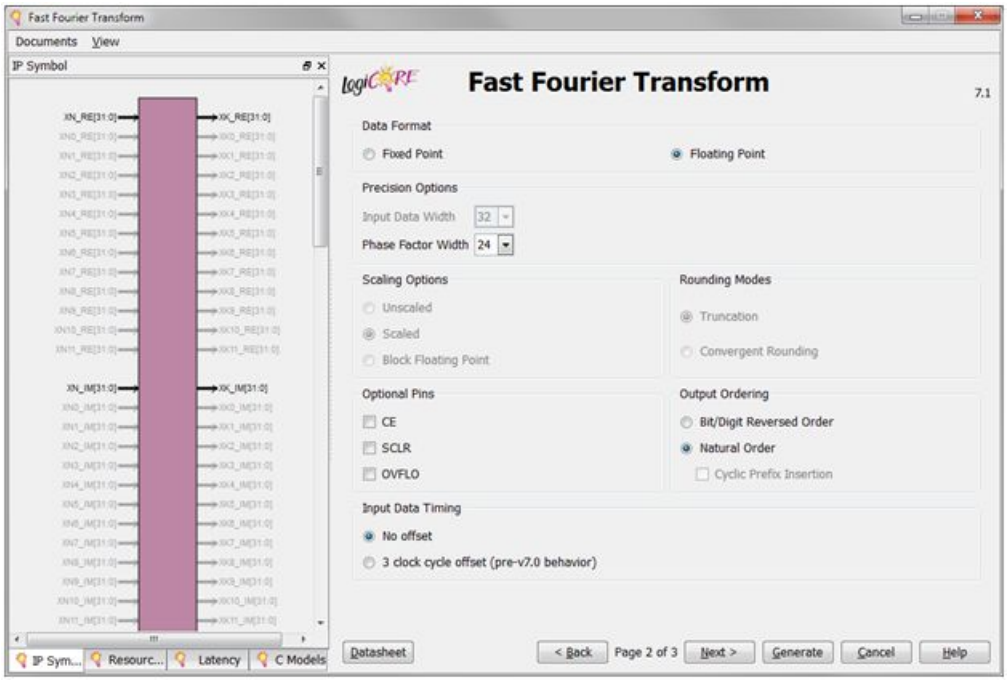
5. IP リストから [Fast Fourier Transform] のバージョン 7.1 を選択します。[Next] をクリックし、次のダイアログ ボックスで [Finish] をクリックします。



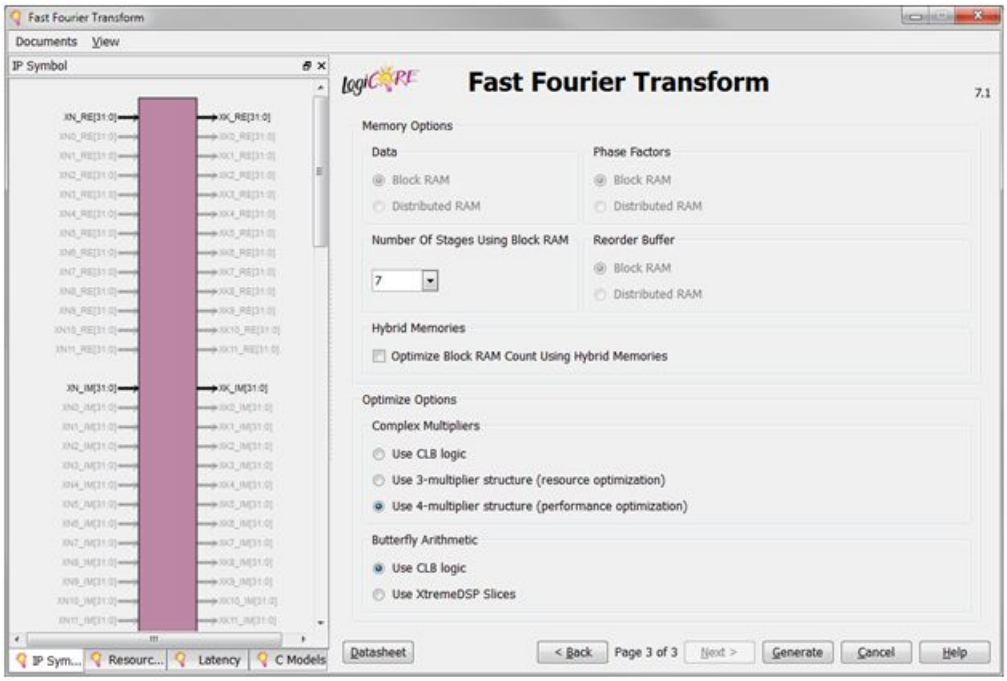
6. Fast Fourier Transform コアの GUI が開いたら、[Transform Length] を [16384] に設定します。[Implementation Options] で [Pipelined, Streaming I/O] をオンにします。[Next] をクリックします。



7. [Data Format] で [Floating Point] をオンにし、[Output Ordering] で [Natural Order] をオンにします。[Next] をクリックします。



8. [Complex Multipliers] で [Use 4-multiplier structure (performance optimization)] をオンにします。[Generate] をクリックしてコアを生成します。



9. 生成した fp_fft_core IP コアをインスタンシエートする最上位モジュール **fp_fft_top** を追加します。ISim ハードウェア協調シミュレーションでは、現在のところ HDL 最上位モジュールのみがサポートされています。このチュートリアルに含まれる完成した `fp_fft_top.v` を使用できます。[Project] → [Add Source] をクリックします。`fp_fft_top.v` を追加します。

手順 2 : テストベンチの作成

1. fp_fft_top インスタンスを実行するテスト ベクターを生成する Verilog テストベンチ モジュール `fp_fft_tb.v` を追加します。このチュートリアルで提供されている完成した `fp_fft_tb.v` を使用できます。[Project] → [Add Source] をクリックします。`fp_fft_tb.v` を追加します。

このテストベンチには、FFT 入力ベクター xn および出力ベクター xk の実数部分と虚数部分を保存する 4 つの 32 ビット × 16384 アレイ (fft_xn_re_data、fft_xn_im_data、fft_xk_re_data、fft_xk_im_data) が含まれています。

シミュレーションが開始すると、テストベンチによりデータ ファイル `fft_xn_data.txt` から FFT 入力ベクターが fft_xn_re_data および fft_xn_im_data アレイに読み込まれます。FFT での計算が終了すると、その値が結果ファイル `fft_xk_data.txt` の fft_xk_re_data および fft_xk_im_data アレイに保存されます。`fft_xn_data.txt` および `fft_xn_data.txt` では、行ごとにデータ ポイントが 1 つ保存されます。各データ ポイントは 32 ビットの 16 進数値 2 つ (実数部分と虚数部分) で構成されており、スペースで区切られています。この 16 進数は、IEEE 754 規格を使用した浮動小数点値を 2 進数で表現したものです。

メモ : TXT データ ファイルは、シミュレーションを実行するディレクトリに配置する必要があります。

テストベンチでは、次の Verilog タスクが定義されています。

- **clear_fft_xk_data**

  `fft_xk_re_data` および `fft_xk_im_data` アレイに 0 を挿入します。

- **load_fft_xn_data**

  データ ファイル `fft_xn_data.txt` からのデータ サンプルを `fft_xn_re_data` および `fft_xn_im_data` アレイに読み込みます。

- **save_fft_xk_data**

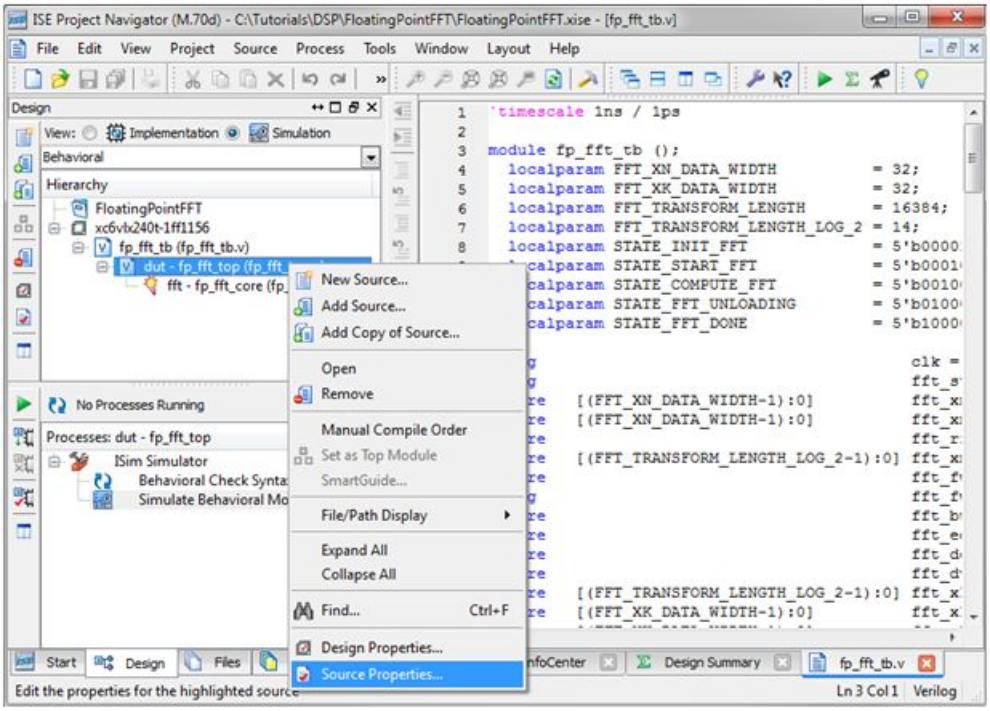
  `fft_xn_re_data` および `fft_xn_im_data` アレイのデータ サンプルを結果ファイル `fft_xk_data.txt` に書き込みます。

手順 3 : ハードウェア協調シミュレーション用のデザインのコンパイル

テストベンチとカスタム制約ファイルを作成したら、ISim コンパイラを使用して、デザインをハードウェア協調シミュレーション用にコンパイルします。これは、Project Navigator でデザインの選択したインスタンスでハードウェア協調シミュレーションをイネーブルにすると実行できます。選択したインスタンスとそれに含まれるサブモジュールは、ISim シミュレーション時にハード

ウェアで協調シミュレーションされます。その他のモジュールは、ソフトウェアでシミュレーションされます。

1. Project Navigator の [View] ペインで [Simulation] をオンにします。[Hierarchy] ペインで [dut - fp_fft_top] インスタンスを右クリックし、[Source Properties] をクリックします。



2. [Category] で [Hardware Co-Simulation] を選択します。[Enable Hardware Co-Simulation] をオンにします。[Clock Port] を [clk] に設定します。[Target Board for Hardware Co-Simulation] を [ML605 (JTAG)] に設定します。[Enable Incremental Implementation] はオフのままにします。

   メモ： ハードウェア協調シミュレーションをイネーブルにしたインスタンスには、特殊なアイコンが付きます。

   デザインをハードウェア協調シミュレーション用に一度コンパイルしたら、[Enable Incremental Implementation] を使用できます。ハードウェア協調シミュレーション用に選択されたインスタンスがその後の実行で変更されない場合、このオプションをオンにすると、ハードウェア協調シミュレーション用の合成、インプリメンテーション、ビットストリーム生成がスキップされます。このオプションを使用すると、ソフトウェアでシミュレーションする部分をすばやく変更し、再シミュレーションできます。

3. [Hierarchy] ペインで [fp_fft_tb] インスタンスを選択します。[Processes] ペインで [Simulate Behavioral Model] を右クリックし、[Process Properties] をクリックします。

4. [Process Properties] ダイアログ ボックスで [Property display level] を [Advanced] に変更します。[Simulate Behavioral Model] プロセスの次のプロパティを設定します。

5. fp_fft_tb インスタンスの [Simulate Behavioral Model] プロセスをダブルクリックしてシミュレーションを実行します。

コマンド ラインでのデザインのコンパイル

ISim コンパイラは、Fuse コマンド ライン ツールを使用して起動できます。Fuse を実行するときには、プロジェクト ファイル、デザインの最上位モジュール、およびリンクするライブラリやライブラリ検索パスなどの引数を指定する必要があります。ハードウェア協調シミュレーション用にデザインをコンパイルするには、次に示す引数を指定する必要があります。

```
fuse -prj [project file] [top level modules]
     -hwcosim_instance [instance]
     -hwcosim_clock [clock]
     -hwcosim_board [board]
     -hwcosim_constraints [constraints file]
     -hwcosim_incremental [0|1]
```

- -hwcosim_instance : ハードウェアで協調シミュレーションするインスタンスの完全階層パスを指定します。
- -hwcosim_clock : インスタンスのクロック入力のポート名を指定します。
  - これはロックステップ部分のクロックで、テストベンチで制御されます。
  - 複数のクロックを使用するデザインでは、このオプションで最高速のクロックを指定し、ISim でシミュレーションが最適化されるようにします。その他のクロック ポートは、通常のデータ ポートとして処理されます。
- -hwcosim_board : 協調シミュレーションに使用するハードウェア ボードを指定します。
- -hwcosim_constraints (オプション) : ハードウェア協調シミュレーション用にインスタンスをインプリメントするための追加制約を含むカスタム制約ファイルを指定します。この制約ファイルでは、インスタンスのどのポートを外部 I/O またはクロックにマップするかも指定します。
- -hwcosim_incremental (オプション) : Fuse で前回生成されたハードウェア協調シミュレーション ビットストリームを再利用し、インプリメンテーション フローをスキップするよう指定します。

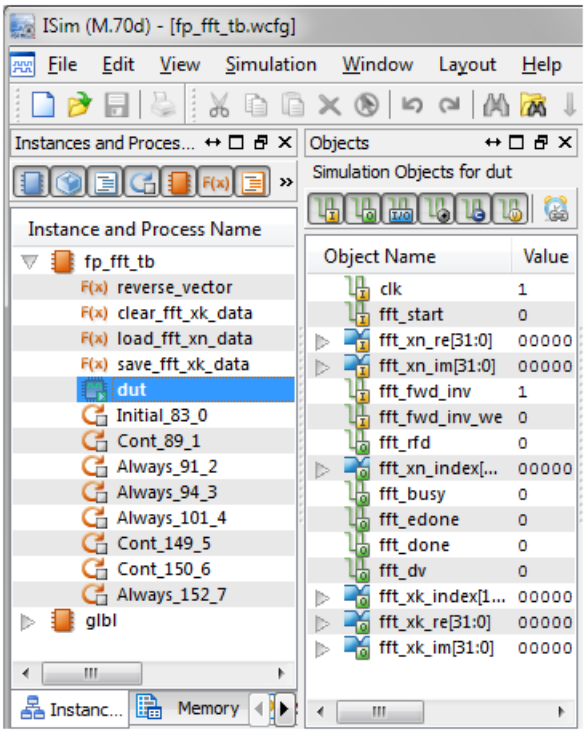
たとえば、このチュートリアルの FFT デザインをコンパイルするには、次のようにコマンド ラインに入力して Fuse を実行できます。

```
fuse -prj fp_fft_tb.prj fp_fft_tb
     –o fp_fft_tb.exe
     -hwcosim_instance /fp_fft_tb/dut
     -hwcosim_clock clk
     -hwcosim_board ml605-jtag
```

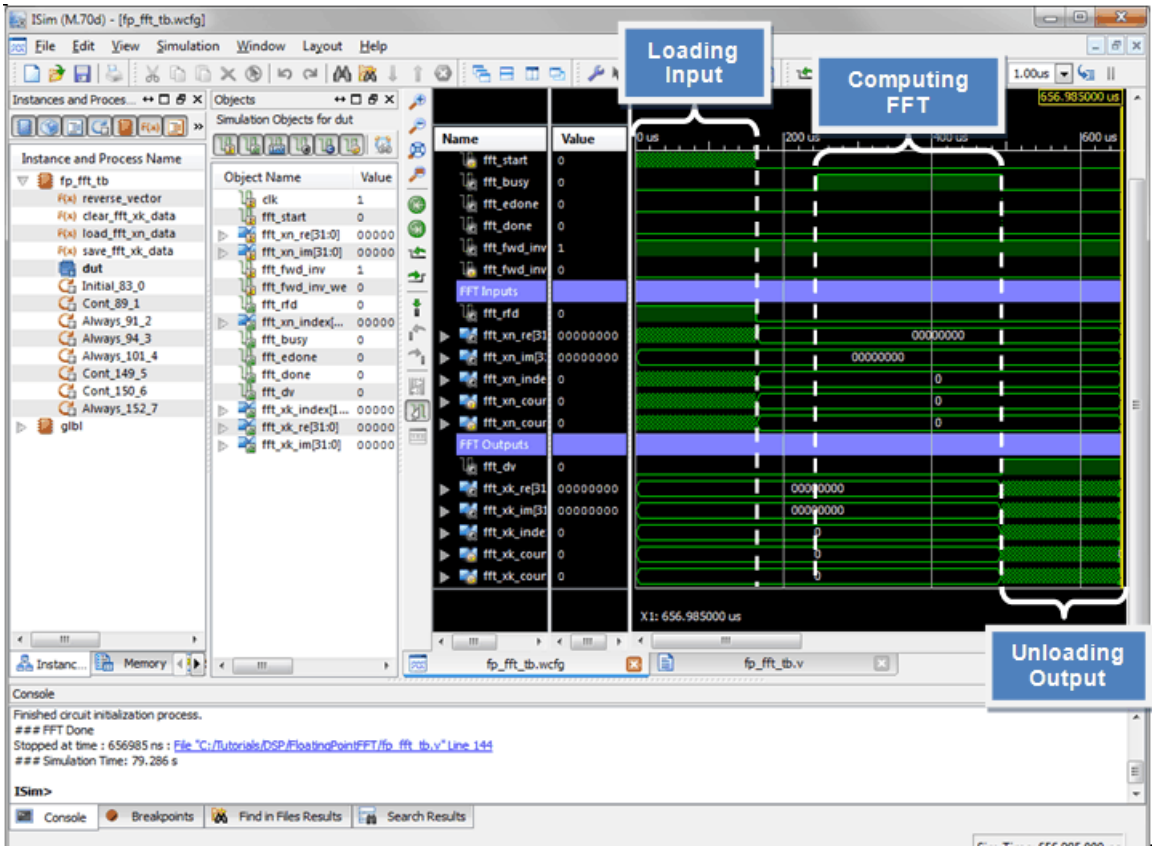
手順 4 : ISim ハードウェア協調シミュレーションの実行

コンパイラで生成されるシミュレーション実行ファイルは、ソフトウェア シミュレーションおよびハードウェア協調シミュレーション フローの両方で同様に使用できます。コンパイルが終了すると、Project Navigator によりシミュレーション実行ファイルが GUI モードで実行されます。

ハードウェア協調シミュレーションに選択されたインスタンスには、[Instances and Processes] パネルで アイコンが表示されます。ハードウェアでインスタンスを実行すると、その内部信号およびサブモジュールをモニターすることはできません。



シミュレーションを開始する前に、ハードウェア協調シミュレーション用に生成されたビットストリームで FPGA がプログラムされます。ISim の [Console] パネルに、ビットストリームをダウンロード中であることを示すメッセージ「Downloading bitstream, please wait till status is READY」が表示されます。FPGA がコンフィギュレーションされると、ビットストリームのダウンロードが完了し、シミュレーションの準備ができたことを示すメッセージ「Bitstream download is complete. READY for simulation」が表示されます。この時点で、ソフトウェア シミュレーション フローと同様に、ISim GUI でシミュレーションを実行できます。



XILINX®

第 3 章

ベンチマーク

次の表では、ISim シミュレーションと ISim ハードウェア協調シミュレーションのパフォーマンスを比較しています。[1]

- FFT デザインを終了するには、シミュレーション クロックの約 65700 サイクルかかります。
- コンパイル時間は、ISim コンパイラ (Fuse) を使用して FFT デザインをシミュレーション実行ファイルにコンパイルするのにかかる時間です。
- ISim ハードウェア協調シミュレーション用のコンパイルは、デザインの一部に対して合成、インプリメンテーション、ビットストリームの生成を実行する必要があるので、コンパイルにかかる時間が長くなります。シミュレーション サイクル数が増加するにつれ、ISim ハードウェア協調シミュレーションを使用したシミュレーションのパフォーマンスの向上が顕著になり、コンパイル時間が増加した分が相殺されます。

まずソフトウェアでデザインを数サイクル分実行していくつかのテスト ケースを検証した後、ISim ハードウェア協調シミュレーションに切り替え、シミュレーション時間を増加してより包括的なテスト ケースの検証を実行することをお勧めします。

[Enable Incremental Implementation] オプションをオンにすると、協調シミュレーションを実行するデザイン部分が変更されていない場合に、前回生成されたビットストリームを再利用することにより、コンパイル時間を短縮できます。この場合、デザインの一部のみが再コンパイルされるので、コンパイル時間が通常の ISim コンパイル時間より短くなる場合もあります。このようにすると、テストベンチやデザインのソフトウェアでシミュレーションする部分を変更しながら、シミュレーションの反復回数を増やすことができます。

ハードウェア協調シミュレーション インターフェイスによって、シミュレーションのパフォーマンスは異なります。JTAG 協調シミュレーション インターフェイスは、JTAG ポートのある FPGA ボードで使用できます。このインターフェイスでは、ハードウェア協調シミュレーションインターフェイスのインプリメントに使用されるリソースが少量に抑えられますが、イーサネットなどその他のインターフェイスに比べるとレイテンシおよびスループットは劣ります。

ポイントツーポイント イーサネット協調シミュレーションインターフェイスでは、10/100/1000Gbps のイーサネット接続がサポートされ、JTAG よりも高いスループットを達成できます。ただし、ハードウェア協調シミュレーションインターフェイスのインプリメントに使用されるリソースが多くなり、サポートされるボードも限られます。

1. ベンチマークは、64 ビット Windows OS で動作する Core i7 2.8GHz CPU および 8GB RAM を搭載したマシンで ISE® 13.2 を実行した結果です。

シミュレーション パフォーマンスの比較

メモ : ISim シミュレーションと比較した場合の速度の増加率をかっこに示します。

	コンパイル時間	シミュレーション時間	合計時間
ISim シミュレーション	49s	1383s	1432s
JTAG を使用した ML605 上での ISim ハードウェア協調シミュレーション[2]	685s (0.07x)	90s (15x)	775s (1.8x)
JTAG を使用した ML605 上での ISim ハードウェア協調シミュレーション[2] ([Enable Incremental Implementation] をオン)	5s (10x)	90s (15x)	95s (15x)
ポイント ツー ポイント イーサネットを使用した ML605 上での ISim ハードウェア協調シミュレーション[3]	887s (0.05x)	5s (277x)	892s (1.6x)
ポイント ツー ポイント イーサネット を使用した ML605 上での ハードウェア協調シミュレーション[3] ([Enable Incremental Implementation] をオン)	5s (10x)	5s (277x)	10s (143x)

2. JTAG 協調シミュレーション インターフェイスには、スピードを 12MHz に設定したプラットフォーム ケーブル USB を使用します。
3. ポイント ツー ポイント イーサネット協調シミュレーション インターフェイスには、PC と ML605 ボードを直接接続するギガビット イーサネット接続を使用します。

その他のリソース

- **ザイリンクス用語集 ：** http://japan.xilinx.com/support/documentation/sw_manuals/glossary.pdf
- **ザイリンクス マニュアル：** http://japan.xilinx.com/support/documentation
- **ザイリンクス サポート：** http://japan.xilinx.com/support