XAPP1165 (v1.0) 2013 年 8 月 5 日

Vivado Design Suite におけるバージョン管理システムの使用

著者 : Jim Wu

概要

このアプリケーション ノートでは、Vivado® Design Suite の非プロジェクト モードとプロジェクト モードの両方でバージョン管理システムを使用する場合の推奨事項について説明します。具体的には、Vivado Design Suite の Manage IP 機能を使用して IP を管理すること、デザイン ソースをリモート ファイルのまま扱うこと、Tcl (Tool Command Language) スクリプトを使用してデザイン ソース管理、デザイン実行、プロジェクト作成を行うことを推奨します。このアプリケーション ノートには、この推奨設計事項に従って RTL からビットストリーム生成までを実行できる完全なサンプル デザインが付属しています。また、非プロジェクト モードとプロジェクト モードの両方で Git を使用して初期デザイン リポジトリを作成する Tcl スクリプトも付属します。

はじめに

Vivado Design Suite は IP/システム中心型の画期的なデザイン環境で、各種デザイン ツールを共通の環境に統合することで、デザインの統合とインプリメンテーションを簡単にしています。チームで設計を行う場合、プロジェクトの管理とコラボレーションを円滑にするためにバージョン管理システムを用いますが、Vivado Design Suite のようなデザイン環境でのバージョン管理システムの運用は容易ではありません。たとえば複雑な FPGA プロジェクトの場合、1 つのプロジェクトに複数のデザイン ソースおよびコンフィギュレーション ファイルが含まれます。このとき、プロジェクトを再作成してインプリメンテーションを再実行するためにバージョン管理が必要となるのは、一部のファイルのみです。たとえば、Vivado Design Suite では次のファイルが使用されます。

- RTL およびネットリスト ファイル (Verilog、SystemVerilog、VHDL、EDIF を含む)
- System Generator モデル (.mdl、.slx)
- IP-XACT コア ファイル (.xci、.mem、.coe)
- Vivado IP インテグレーター ブロック図ファイル (.bd)
- エンベデッド サブシステム (.xmp) およびファイル (.elf、.bmm)
- ザイリンクス デザイン制約ファイル (.xdc)
- デザインの生成とコンパイルを実行するスクリプト (.tcl)
- コンフィギュレーション ファイル。Vivado シミュレータおよび Vivado ロジック解析のコンフィギュレーション ファイル (.wcfg) を含む

Vivado Design Suite は特定のバージョン管理システムとの組み合わせを想定していません。あらゆるバージョン管理システムと組み合わせて使用できるよう、次の機能を備えています。

- ファイルが変更された場合のみプロジェクト ファイルとタイムスタンプを更新する

  Vivado Design Suite プロジェクトを開いても、ファイルを変更しない限りバージョンが古くなることはありません。

- ASCII ベースのプロジェクト ファイルをサポート

  これには、プロジェクト、ソース管理、IP 設定、フロー制御、run 管理などに関するファイルが含まれます。ほとんどの場合、ファイルは XML 形式のため、容易にカスタム解析が可能です。バイナリ ファイルは、知的所有権で保護された情報を格納する場合など、最小限にとどめています。

- 広範な Tcl スクリプト機能をサポート

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

オブジェクト指向の Tcl 拡張により、複雑なもの、カスタマー独自のもの、自動化されたものを含め、あらゆるバージョン管理システムと組み合わせて使用できます。

Vivado がサポートするデザイン フロー モードには、非プロジェクト モードとプロジェクト モードの 2 つがあります。非プロジェクト モードは Tcl スクリプトを利用してコンパイルを実行するスタイルのフローで、ソースおよびデザイン プロセスはユーザー自身で管理します。プロジェクト モードはプロジェクトをベースにしたフローで、デザイン プロセスおよびデザイン データはプロジェクトおよびプロジェクト ステートを使用して自動的に管理されます。どちらのフローも、Tcl スクリプトによるバッチ モードでの実行および Vivado 統合設計環境 (IDE) と呼ばれる GUI 環境でのインタラクティブな実行の両方が可能です。これら 2 種類のデザイン フロー モードの詳細は、『Vivado Design Suite ユーザー ガイド : デザイン フローの概要』(UG892) [参照 1] を参照してください。

表 1 に、これら 2 つのモードでバージョン管理システムを使用する場合の推奨事項の概要をまとめます。

表 1 : バージョン管理システムを使用する場合の推奨事項の概要

モード	推奨事項
非プロジェクト モード	デザイン ソースの管理、ツール オプションの制御、デザインの実行、およびインプリメンテーション結果の生成は Tcl スクリプトを作成して実行する
プロジェクト モード	Vivado Design Suite の Tcl コマンド `write_project_tcl` で生成したプロジェクト作成スクリプトを用いて、リモート プロジェクト ソースを使用する
両モード共通	すべての IP 出力を専用のディレクトリに生成し、再帰的にチェックインできるよう、IP の作成、設定、生成にはスタンドアロンの Manage IP 機能を使用する

注記 : Tcl コマンドの詳細は、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) [参照 2] を参照してください。

次のセクションからは、プロジェクト モードと非プロジェクト モードの両方で実際にサンプル デザインを使用しながら、プロジェクト ファイル システムについて説明します。これらのセクションでは、Git を使用してリポジトリを作成する手順についても説明していきます。Git はオープンソースで分散型のバージョン管理/ソース コード管理システムです。分散モデルを採用し、速度と効率に優れた Git は、近年ユーザー数を増やしています。ザイリンクス オープンソース Wiki [参照 3] で提供しているプロジェクトにも、ソース管理に Git を使用しているものがあります。Git の詳細は、Git のウェブサイト [参照 4] を参照してください。

バージョン管理と IP

IP の管理

非プロジェクト モード、プロジェクト モードいずれの場合も、IP の作成、設定、生成には Vivado IDE (図 1) が備えるスタンドアロンの Manage IP 機能を使用し、IP および IP 出力をバージョン管理下にあるデザインの専用ディレクトリに生成することを推奨します。非プロジェクト モードおよびプロジェクト モードで IP を管理するには、次の手順を実行します。

1. Vivado IDE の Getting Started ページで [Manage IP] をクリックします。

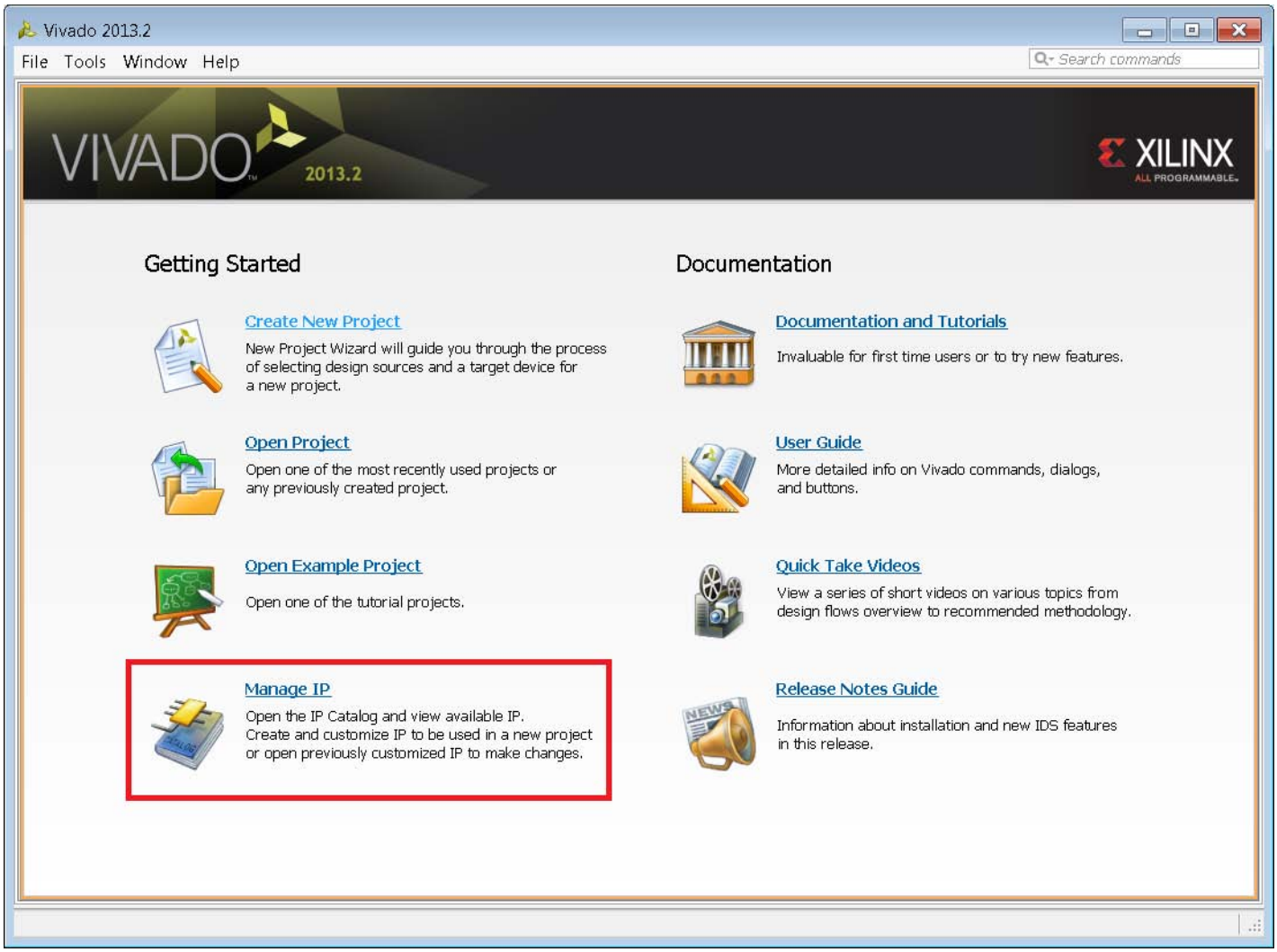

図 1 : Vivado IDE の [Manage IP]

2. Vivado IDE で、Vivado IP カタログを使用して新規 IP を作成します。

3. IP の設定が完了したら、IP リポジトリに含まれるファイルの数を最小限にするために [Generate Output Products] ダイアログ ボックスで [Skip] をクリックします (図 2)。

   これにより、この IP の IP カタログ ファイル (.xml)、IP コンフィギュレーション ファイル (.xci)、そして VHDL (.vho) または Verilog (.veo) の IP インスタンシエーション ファイルのみが生成されます。

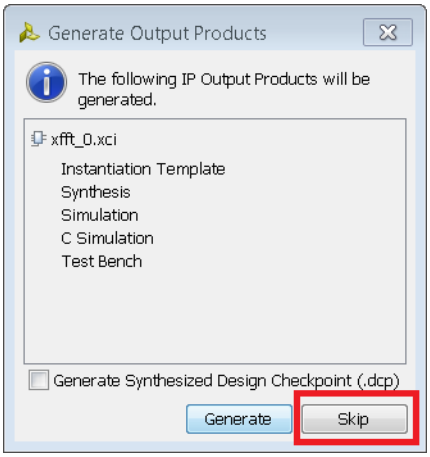

図 2 : [Generate Output Products] ダイアログ ボックス

4. ある IP に関連して、バージョン管理下に置く必要のあるファイルすべてをリストとして取得するには、`get_files` コマンドを実行します。次に例を示します。

```
get_files -all -of [get_files <IP_NAME>.xci]
```

   注記：Vivado Design Suite 2013.2 を使用する場合は、このアプリケーション ノートのデザイン ファイルに含まれる `git_manage_ip.tcl` スクリプトを実行する必要があります。この Tcl スクリプトは、管理対象の IP 格納場所にある IP に関連するすべてのファイルのリストを返し、バージョン管理が必要なすべての IP ファイルを取得して、これらのファイルを Git リポジトリにチェックインするバッチ ファイルを生成します。

5. 次の方法で、既存の IP を IP リポジトリにインポートします。

   a. IP ディレクトリを現在の IP 格納場所にコピーします。次に例を示します。

```
file copy {C:\xapp\working\vcs_project_flow\xfft_0}
{C:\xapp\working\vcs_ip_location}
```

   b. `add_files` Tcl コマンドを使用して、IP コンフィギュレーション ファイル (.xci) を追加します。次に例を示します。

```
add_files {C:\xapp\working\ip_location\xfft_0\xfft_0.xci}
```

   図 3 は、これらのコマンドを使用して既存の FFT コアを管理対象の IP 格納場所に追加した例です。

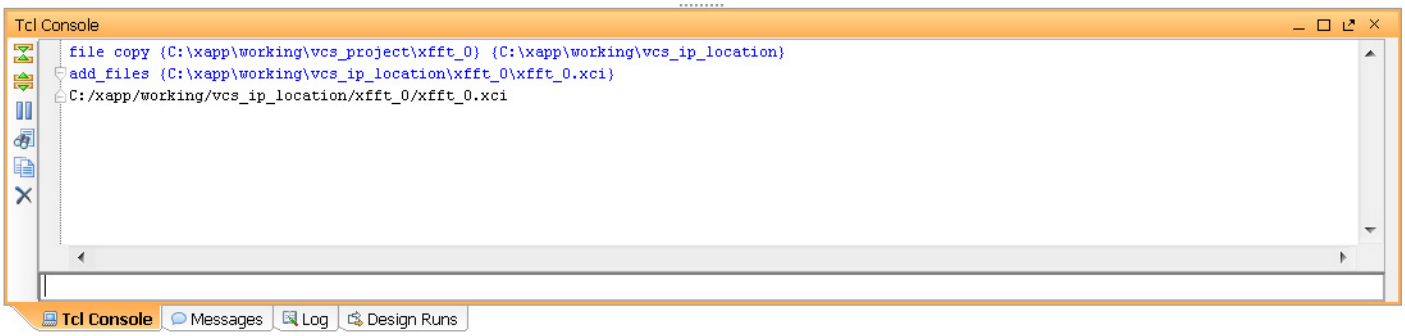

図 3 : add_files Tcl コマンド

6. IP を作成または追加したら、同じディレクトリ構造、IP コンフィギュレーション ファイル (.xci)、データ ファイル (.coe、.mem など) をデザイン リポジトリに再帰的にチェックインします。

管理対象の IP 格納場所にある IP を用いたシミュレーション

シミュレータまたはその他の EDA (Electronic Design Automation) ツールで使用する IP 出力ファイルの順序リストを生成する方法には、次の 2 つがあります。

report_compile_order Tcl コマンドを使用する方法

現在の Manage IP プロジェクトに含まれるすべての IP のコンパイル順を返すには、次のコマンドを実行します (図 4)。

```
report_compile_order -fileset {sources_1}
```

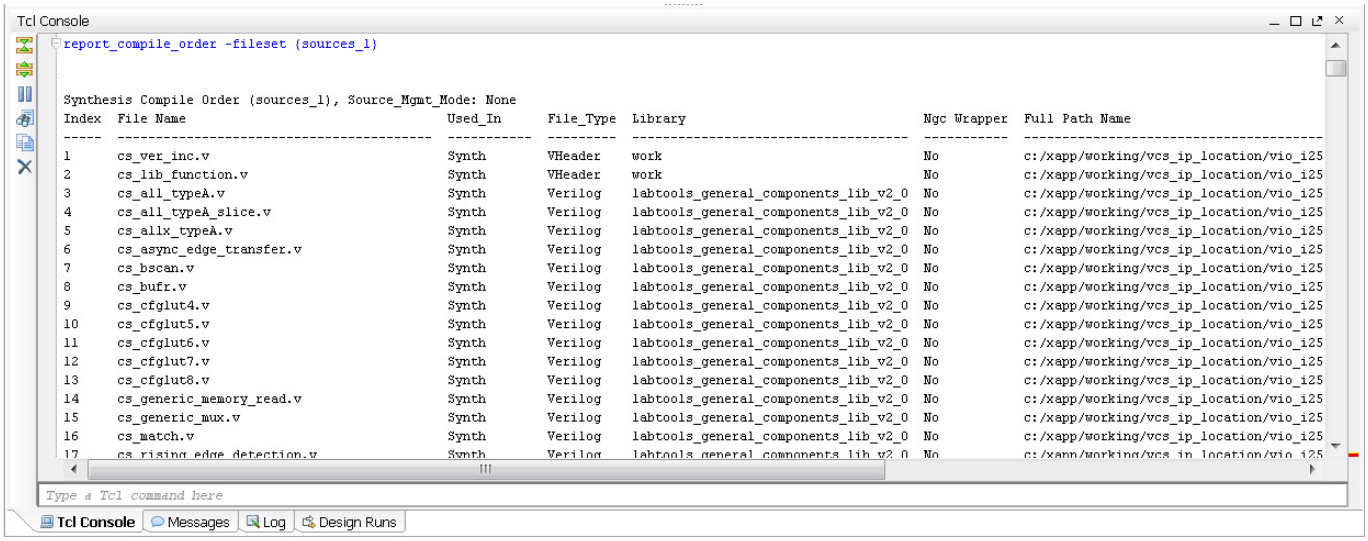

図 4 : report_compile_order Tcl コマンド

get_files Tcl コマンドを使用する方法

管理対象の IP 格納場所にある特定の IP のコンパイル順を返すには、次のコマンドを実行します (図 5)。

```
join [get_files -compile_order sources -used_in simulation -of [get_files
xfft_0.xci]] \n
```

注記 : `join` コマンドを使用すると、返されたリストが 1 ファイルずつ改行して表示されます。

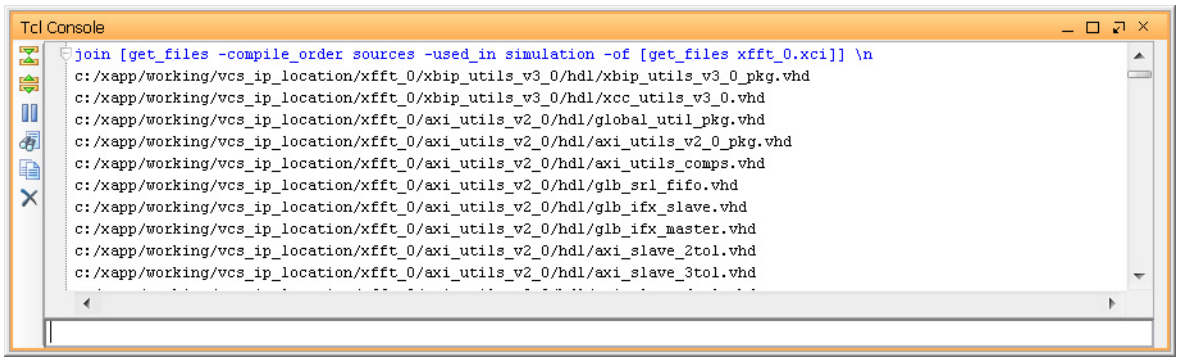

図 5 : get_files Tcl コマンド

管理対象の IP 格納場所にある IP を用いたボトムアップ合成フロー

デザイン環境によっては、IP の再コンパイル時間を短縮し、合成結果を維持するためにボトムアップ合成を使用することがあります。Manage IP プロジェクトではこれがサポートされ、自動化されています。IP 出力を生成する際に、[Generate Output Products] ダイアログ ボックスで [Generate Synthesized Design Checkpoint (.dcp)] をオンにします (図 6)。これで、自動的に OOC (Out-of-Context) モジュール run が生成され、合成されます。ボトムアップ合成フローでデザイン リポジトリにチェックインするには、この DCP ファイルが必要です。

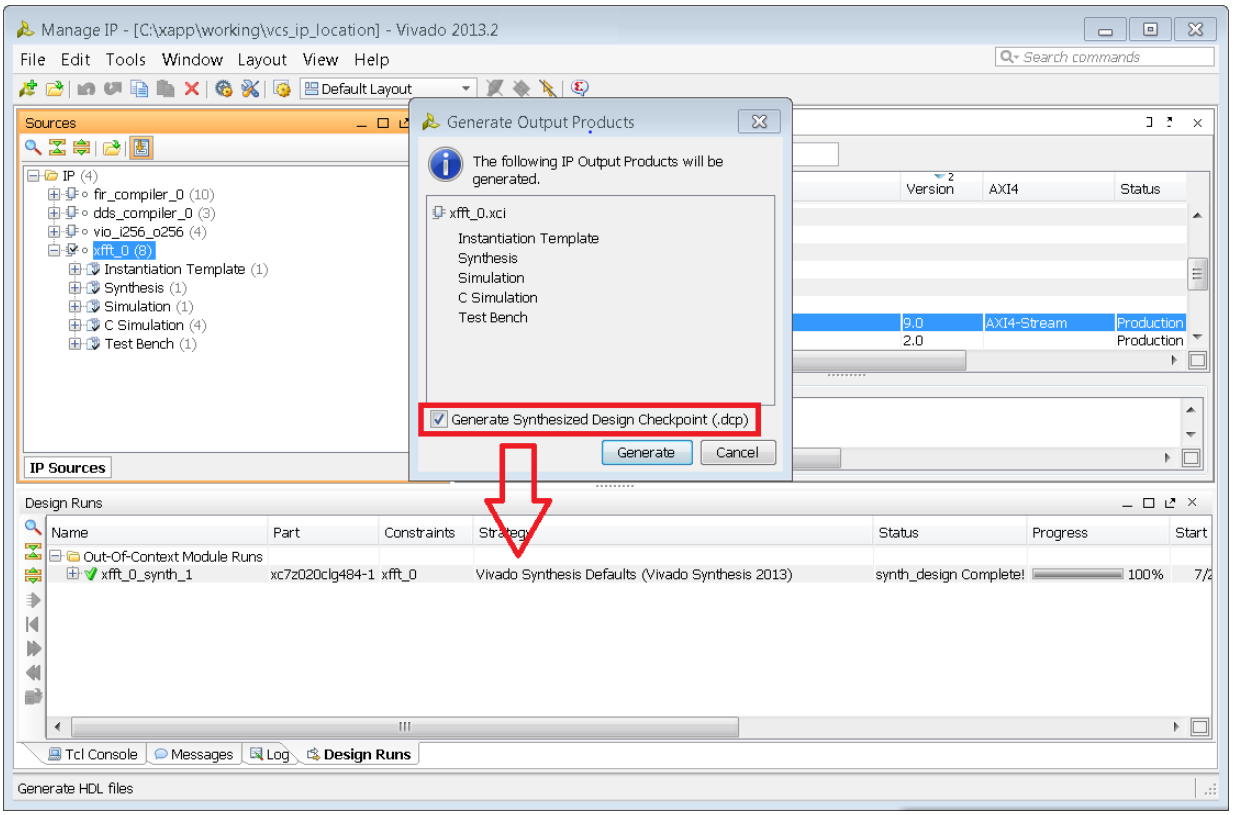

図 6 : 合成済みデザイン チェックポイントの生成

サンプル プロジェクトの IP 格納場所

図 7 に、非プロジェクト モードおよびプロジェクト モードでのサンプル デザインの IP 格納場所を示します。

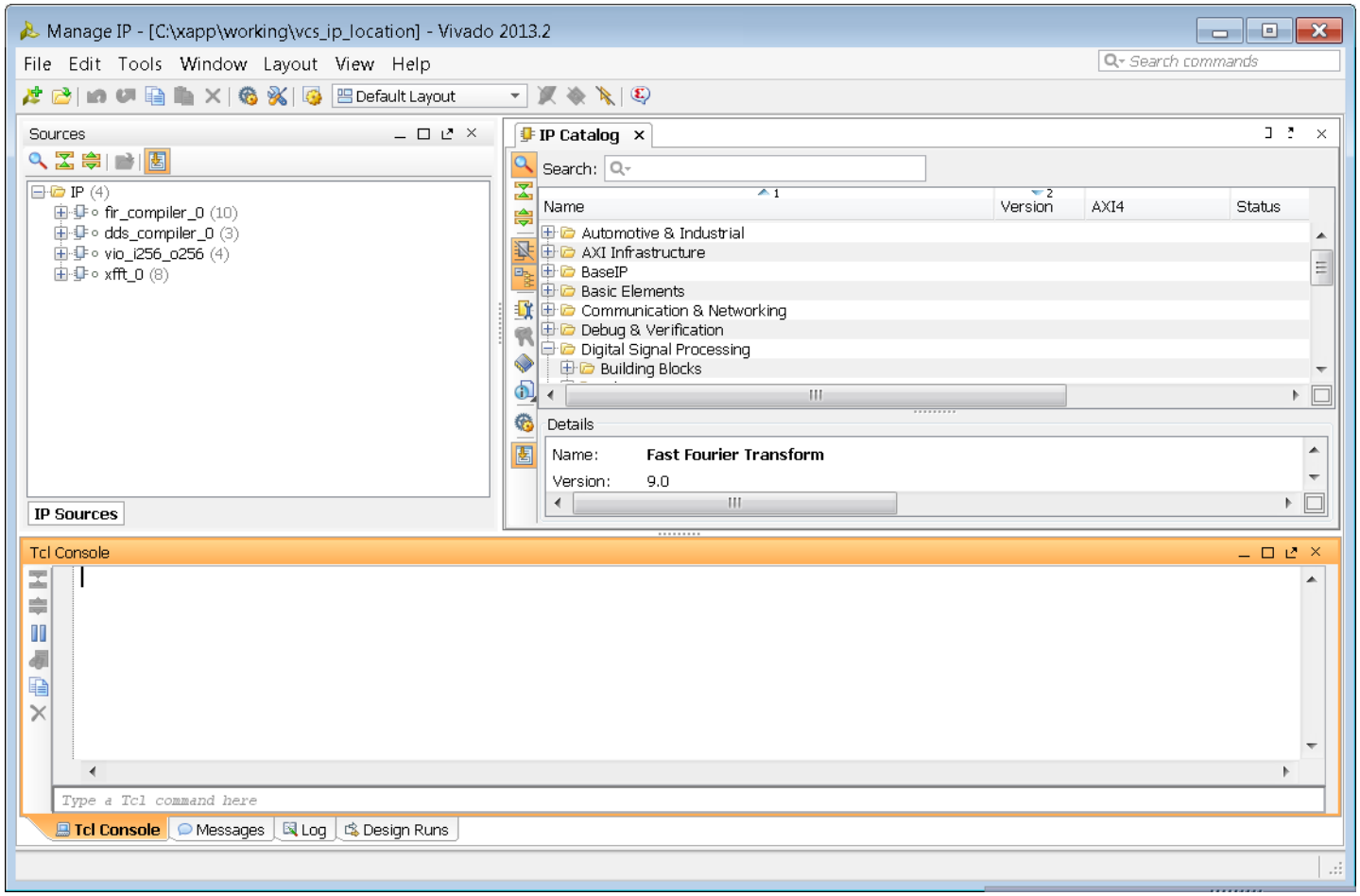

図 7 : サンプル デザインの IP 格納場所

サンプル デザインの IP 格納場所のディレクトリ構造は次のとおりです。

```
vcs_ip_location
|-- git_manage_ip.tcl
|-- coeffs
|   `-- fir_p2_1db_s3_50db.coe
|-- dds_compiler_0
|   |-- dds_compiler_0.vho
|   |-- dds_compiler_0.xci
|   `-- dds_compiler_0.xml
|-- fir_compiler_0
|   |-- fir_compiler_0.vho
|   |-- fir_compiler_0.xci
|   `-- fir_compiler_0.xml
|-- ila_0
|   |-- ila_0.xci
|   `-- ila_0.xml
|-- vio_i256_o256
|   |-- vio_i256_o256.xci
|   `-- vio_i256_o256.xml
|-- xfft_0
        |-- xfft_0.vho
        |-- xfft_0.xci
        `-- xfft_0.xml
```

非プロジェクト モード

非プロジェクト モードでは、Vivado Design Suite はデザイン ソースと制約ファイルを直接入力として取り込み、フロー全体をメモリ内で実行します。このモードではコンフィギュレーション ファイルはハード ディスクに保存されず、プロジェクト管理やフロー ナビゲーションの機能もありません。ただし、任意の段階でデザイン チェックポイントやレポートを作成し、これらを Vivado IDE で開いてデザインを解析できます。非プロジェクト モードでは、コマンド ラインを使用した制御が可能なほか、デザイン チェックポイントを使用した制約の割り当てやクロスプローブなど、Vivado IDE の強力な機能も利用できます。

バージョン管理システムでは一般に非プロジェクト モードが使用され、事実、その使用が強く推奨されています。非プロジェクト モードでは、ツールに渡される入力および使用するオプションはユーザーが管理します。コンパイルへのすべての入力、およびこれらの入力からのコンパイルを制御して希望の最終出力を生成する Tcl スクリプトをバージョン管理の対象に含めることを推奨します。このフローでリポジトリにチェックインする必要のあるファイルは、次のとおりです。

- コンパイルを実行し、ツールのオプションを制御する Tcl スクリプト
- `read_*` または `add_source` Tcl コマンドに渡される入力ファイル
- オプションのツール レポートおよび出力ファイル (ビットストリーム、デザイン チェックポイント、レポートなど)

注記：プロジェクト全体をバージョン管理の対象にしないでください。

非プロジェクト モードにおけるサンプル デザインのディレクトリ ツリー

非プロジェクト モードにおけるデザインのディレクトリ構造の例を次に示します。これは、インプリメンテーションの実行および合成/インプリメンテーション実行結果の再作成のために、リポジトリでバージョン管理が必要な最小限のファイルセットです。IP はすべて Manage IP 機能を使用して生成した IP 格納場所にあります。

```
|-- script
|   |-- file_list.txt
|   `-- vivado_non-project.tcl
|-- sim
|   `-- spectrum_top_tb.v
`-- src
    |-- constrs
    |   |-- spectrum_top_physical.xdc
    |   `-- spectrum_top_timing.xdc
    |-- verilog
    |   |-- cplx_mult.v
    |   `-- spectrum_top.v
    `-- vhdl
        `-- heartbeat_gen.vhd
```

プロジェクト モード

プロジェクト モードでは、Vivado IDE がプロジェクト ファイル システムを作成し、プロジェクトのデータとステータスを管理します。Vivado IDE でプロジェクトを管理するには、プロジェクト モードで Tcl コマンドをインタラクティブに使用する方法と、Tcl スクリプトを使用する方法があります。Vivado IDE は、デザイン サイクル全体にわたってインターフェイスを統一した堅牢なデザイン環境を提供します。プロジェクト モードでは、次の自動機能を利用できます。

- デザイン ソースおよびデザイン実行の設定と管理
- IP の設定、インプリメンテーション、パッケージング
- 包括的なクロスプローブ機能によるデザインの解析と検証
- I/O、クロック、タイミング、フロアプランに対する制約の作成と割り当て
- デバッグおよび解析のための Vivado ロジック解析コアの挿入
- ビットストリームの生成とプログラミング

図 8 は、Vivado IDE でサンプル デザインを開いたところです。

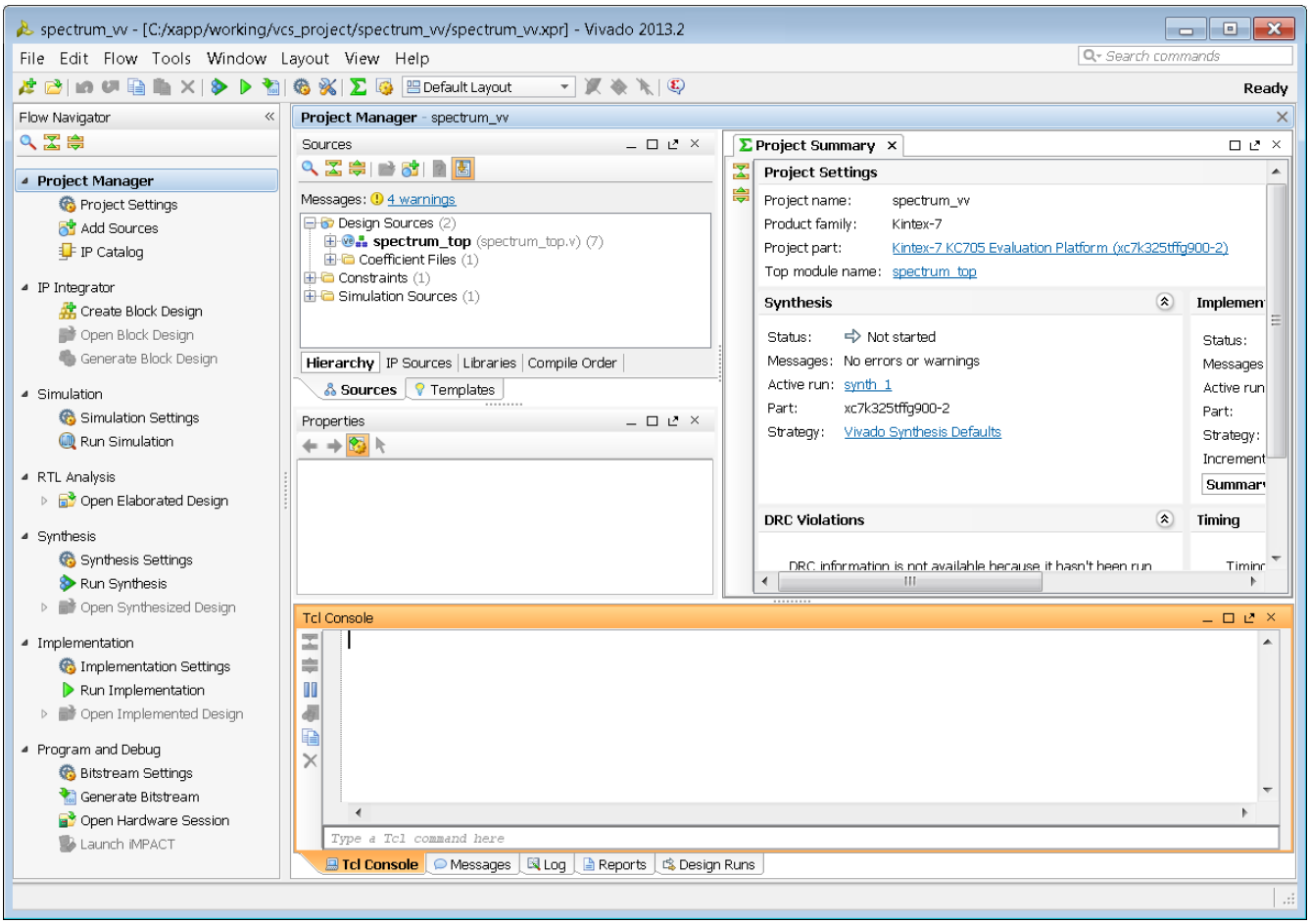

図 8 : サンプル デザイン

プロジェクト モードでバージョン管理を使用する場合の推奨事項

プロジェクト モードでバージョン管理システムを使用する場合の推奨事項は、次のとおりです。

- すべての入力ソース (特に IP ソース) をリモート ソース ディレクトリに置いたままにする。

  ソースはプロジェクトにインポートするのではなく、参照先ソースとして追加します。

- プロジェクト、ツール オプション、設定を作成する Tcl スクリプトを作成および維持する。

  Vivado Design Suite 2013.1 以降には、プロジェクト作成用の Tcl スクリプトをエクスポートする Tcl コマンド `write_project_tcl` が用意されています。このスクリプトをプロジェクト作成スクリプトのテンプレートとして使用し、フローまたは入力ソースの場所を変更してカスタマイズできます。

- プロジェクト設定の変更はなるべくプロジェクト作成 Tcl スクリプト内で行う。

  Vivado IDE でインタラクティブに行ったプロジェクトへの変更は、`write_project_tcl` コマンドで新しい Tcl スクリプトへ書き出す必要があります。バージョン管理システムのマージ機能を利用して、新しいスクリプトとリポジトリ内のスクリプトの差異をなくします。

- バージョン管理システムでサポートされている方法でファイルをチェックインおよびチェックアウトする。

  Vivado IDE の Tcl コンソールでコマンドラインのコマンドを使用してファイルをチェックインおよびチェックアウトすることもできます。

プロジェクト モードにおけるサンプル プロジェクトのディレクトリ ツリー

プロジェクト モードにおけるプロジェクトのディレクトリ構造の例を次に示します。`write_project_tcl` コマンドを実行すると、script ディレクトリに `create_specturm_prj.tcl` スクリプトが生成されます。このスクリプトは、プロジェクトおよびプロジェクト モードで使用するすべての設定を作成するものです。

```
vcs_project
|-- debug
|   `-- hw_ila_data_1.wcfg
|-- script
|   |-- file_list.txt
|   |-- git_project.tcl
|   `-- spectrum_vv_prj_flow.tcl
|-- sim
|   |-- spectrum_top_tb.v
|   `-- spectrum_top_tb.wcfg
|-- script
|   `-- spectrum_top_tb.wcfg
`-- src
    |-- constrs
    |   |-- spectrum_top_physical.xdc
    |   `-- spectrum_top_timing.xdc
    |-- verilog
    |   `-- cplx_mult.v
    |   `-- spectrum_top.v
    `-- vhdl
        `-- heartbeat_gen.vhd
```

write_project_tcl Tcl コマンド

Vivado Design Suite 2013.1 以降には `write_project_tcl` コマンドが用意されており、Vivado Design Suite プロジェクトを任意のバージョン管理システムと組み合わせて使用できます。このコマンドを実行すると、現在のプロジェクトを再作成する Tcl スクリプトがエクスポートされます。ここで、次のいずれかを実行します。

- Vivado IDE を使用して手動でプロジェクトを作成する。
- 既存のプロジェクトを使用し、入力ソースとプロジェクト設定を用いてプロジェクトを再作成する Tcl スクリプトをエクスポートする。

注記：プロジェクトを正しく再作成できるように、このスクリプトをバージョン管理の対象に含めることを推奨します。実際に管理されているプロジェクト自体をバージョン管理下に置くことは推奨していません。

必要であれば、ファイルをエクスポートする Tcl コマンドを実行する前に、現在の作業ディレクトリを適切なデスティネーション ディレクトリに変更しておきます。次に示すコマンドは、現在のプロジェクト ディレクトリに移動してから write_project_tcl コマンドを実行します (図 9)。

```
pwd
cd [get_property DIRECTORY [current_project]]
pwd
write_project_tcl -force spectrum_vv_prj_flow.tcl
```

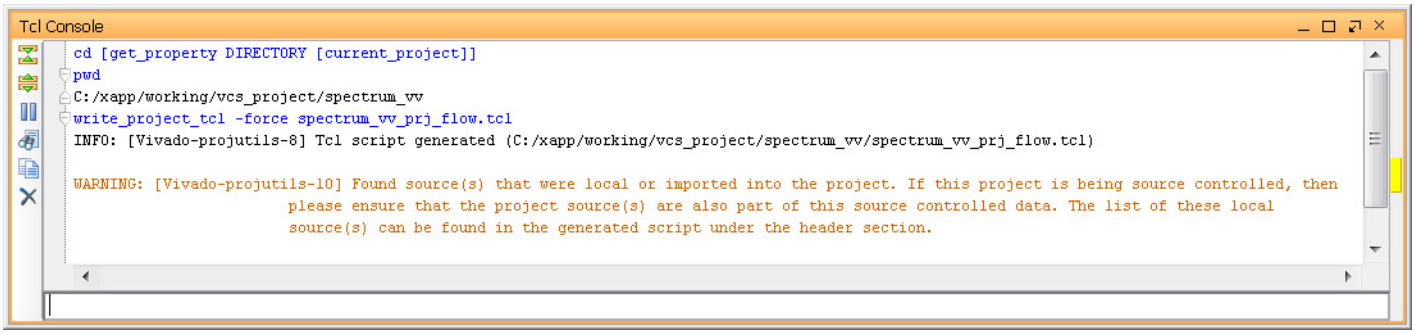

図 9 : write_project_tcl Tcl コマンド

これらのコマンドを実行すると、次のようになります。

- ローカルの入力ファイルまたはプロジェクトにインポートされた入力ファイルがある場合は、警告が表示される。

  入力ソースはすべてリモート ソース ディレクトリに置くことを推奨します。

- Tcl スクリプト spectrum_vv_prj_flow.tcl が現在の作業ディレクトリに保存される。

  このスクリプトに含まれる重要なコマンドの詳細を次に示します。これらを使用して、スクリプトをカスタマイズできます。

```
# Create project
create_project spectrum_vv ./spectrum_vv

# Set project properties (parts, boards, etc)
set obj [get_projects spectrum_vv]
set_property "board" "xilinx.com:kintex7:kc705:1.0" $obj

# Create 'sources_1' fileset
if {[string equal [get_filesets sources_1] ""]} {
  create_fileset -srcset sources_1
}

# Add files to 'sources_1' fileset.All files including IP configuration
files added with add_files command need to be checked in to the design
repository
set obj [get_filesets sources_1]
set files [list \
 "C:/dds_compiler_v5_0_0/dds_compiler_v5_0_0.xci"\

"C:/spectrum_vv/spectrum_vv.srcs/sources_1/ip/xfft_v8_0_0/xfft_v8_0_0.xci
"\
 "C:/xapp/working/vcs_project/src/coeff/fir_p2_1db_s3_50db.coe"\
 "C:/xapp/working/vcs_project/src/vhdl/heartbeat_gen.vhd"\
]
add_files -norecurse -fileset $obj $files

# Set the top module for the project
set obj [get_filesets sources_1]
```

```
    set_property "top" "spectrum_top" $obj

    # Create 'constrs_1' fileset (if not found)
    if {[string equal [get_filesets constrs_1] ""]} {
      create_fileset -constrset constrs_1
    }

    # Add files to 'constrs_1' fileset All files added with add_files command
    need to be checked in to the design repository
    set obj [get_filesets constrs_1]
    set files [list \
     "C:/xapp/working/vcs_project/src/constrs/spectrum_top_physical.xdc"\
     "C:/xapp/working/vcs_project/src/constrs/spectrum_top_timing.xdc"\
    ]
    add_files -norecurse -fileset $obj $files

    # Set 'constrs_1' fileset file properties for local files
    set file "constrs/spectrum_top_physical.xdc"
    set file_obj [get_files "*$file" -of_objects constrs_1]
    set_property "file_type" "XDC" $file_obj

    # Add files to 'sim_1' fileset All files added with add_files command need
    to be checked in to the design repository
    set obj [get_filesets sim_1]
    set files [list \
     "C:/xapp/working/vcs_project/sim/spectrum_top_tb.v"\
    ]
    add_files -norecurse -fileset $obj $files

    # Set top module for the testbench
    set obj [get_filesets sim_1]
    set_property "top" "spectrum_top_tb" $obj

    # Create 'synth_1' run (if not found)
    if {[string equal [get_runs synth_1] ""]} {
      create_run -name synth_1 -part xc7k325tffg900-2 -flow {Vivado Synthesis
    2012} -strategy "Vivado Synthesis Defaults" -constrset constrs_1
    }

    # Create 'impl_1' run (if not found)
    if {[string equal [get_runs impl_1] ""]} {
      create_run -name impl_1 -part xc7k325tffg900-2 -flow {Vivado
    Implementation 2012} -strategy "Vivado Implementation Defaults" -constrset
    constrs_1 -parent_run synth_1

}
```

Git によるバージョン管理

Git はオープンソースで分散型のバージョン管理/ソース コード管理システムです。Git の作業ディレクトリはそれぞれが完全なリポジトリであり、履歴およびバージョン追跡データを保存するための中央サーバーは必要ありません。Git リポジトリは、プロジェクトのルート ディレクトリにある `.git` ディレクトリ 1 つに格納されています。このため、作業ディレクトリを動的に追加または削除する EDA ツールとの相性に優れています。

Git のインストール

Git をインストールするには、次の手順を実行します。

1. Git のウェブサイト [参照 4] から最新バージョンの Git をダウンロードします。
2. Git をインストールします。[Adjusting your PATH environment] ページで [Run Git from the Windows Command Prompt] を選択する点を除き、セットアップ オプションはすべてデフォルトのままにします (図 10)。

   注記：このアプリケーション ノートでは、Windows 7 64 ビット環境で Git-1.8.3-preview20130601 を使用します。

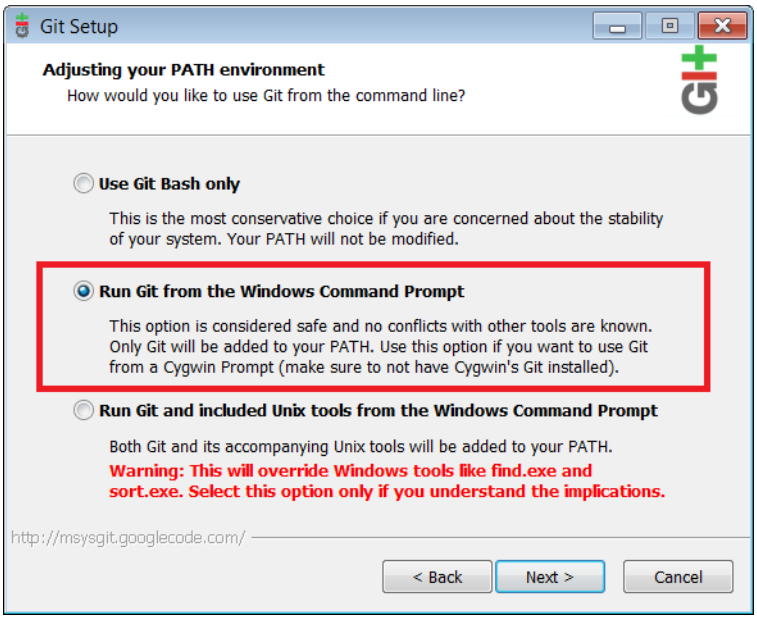

図 10 : Git のセットアップ オプション

Git リポジトリの作成

このセクションでは、非プロジェクト モードでサンプル デザインの初期デザイン リポジトリを作成する手順について説明します。

注記：プロジェクト モードでのリポジトリ作成手順も同様で、プロジェクト管理にプロジェクト作成 Tcl スクリプトを使用します。

非プロジェクト モードでの Git リポジトリの初期化

Git リポジトリを初期化するには、次の手順を実行します。

1. Windows エクスプローラーで、リポジトリ用のディレクトリを作成します。次に例を示します。
   `C:/xapp/repository/vcs_non_project`
2. 作成したディレクトリへ移動し、右クリックして [Git Bash] をクリックします (図 11)。

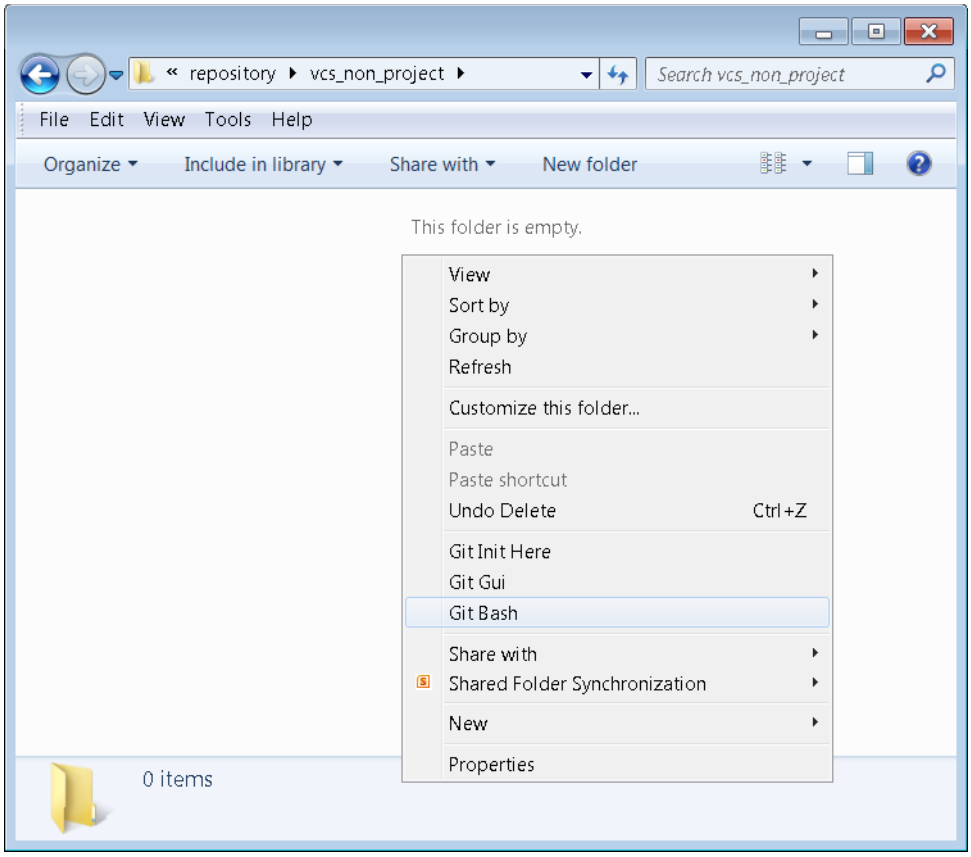

図 11 : vcs_non_project ディレクトリで [Git Bash] コマンドを実行

3. Git Bash ウィンドウで次のコマンドを実行し、ベア リポジトリを作成します (図 12)。

```
$ git init --bare
```

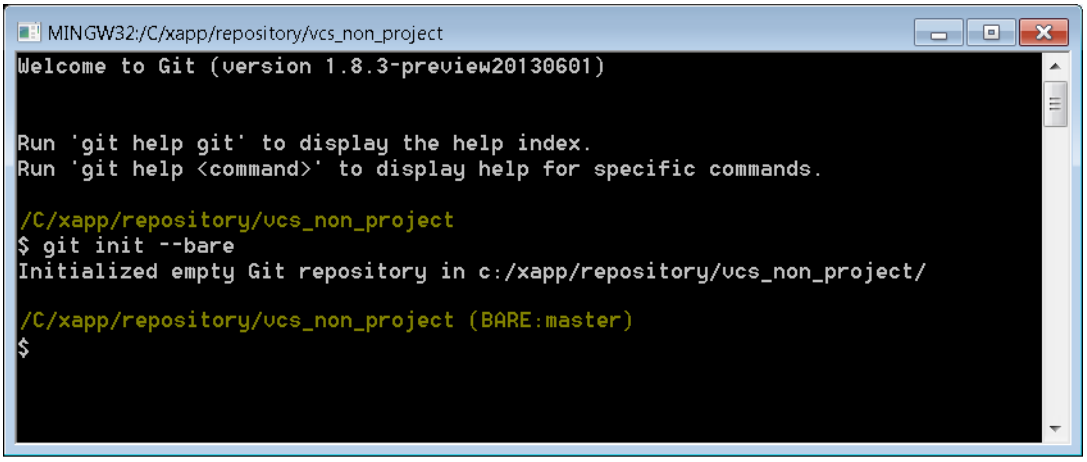

図 12 : ベア リポジトリの作成

Concurrent Versions System (CVS) または Apache Subversion (SVN) バージョン管理システムの中央リポジトリ同様、すべてのチーム メンバーがベア リポジトリとの間で変更を双方向にプッシュ / プルできます。図 13 に、ベア リポジトリのディレクトリおよびファイル構造を示します。

注記：ベア リポジトリに格納できるのはバージョン管理情報のみで、作業ファイルは保存されません。

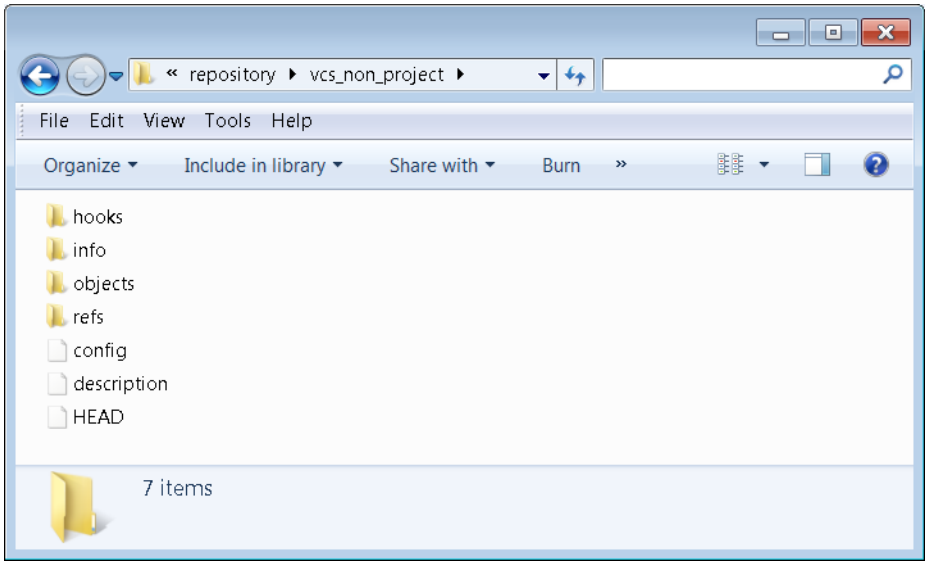

図 13 : ベア リポジトリ

非プロジェクト モードでの作業リポジトリの作成

Git は分散型のバージョン管理システムです。各チーム メンバーの作業エリアは、Git ベア リポジトリの完全なクローンです。ベア リポジトリのクローンを作成するには、次の手順を実行します。

1. Windows エクスプローラーで、作業エリア用のディレクトリを作成します。次に例を示します。
   `C:/xapp/working`
2. 作成したディレクトリへ移動し、右クリックして [Git Bash] をクリックします (図 14)。

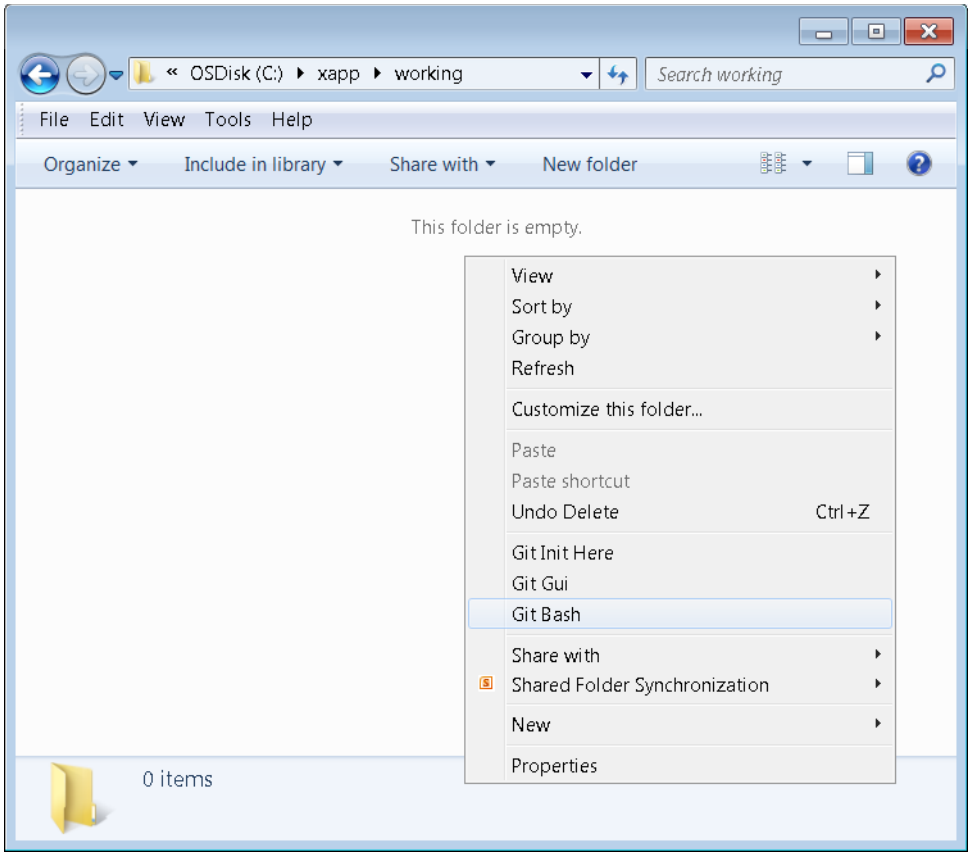

図 14 : 作業ディレクトリで [Git Bash] コマンドを実行

3. Git Bash ウィンドウで次のコマンドを実行し、ベア リポジトリのクローンを作成します (図 15)。

```
$ git clone c:/xapp/repository/vcs_non_project
```

このコマンドを実行すると、作業リポジトリとして `vcs_non_project` ディレクトリが作成されます。このディレクトリには、バージョン管理情報を格納する `.git` という名前の隠し属性のサブディレクトリが含まれます。

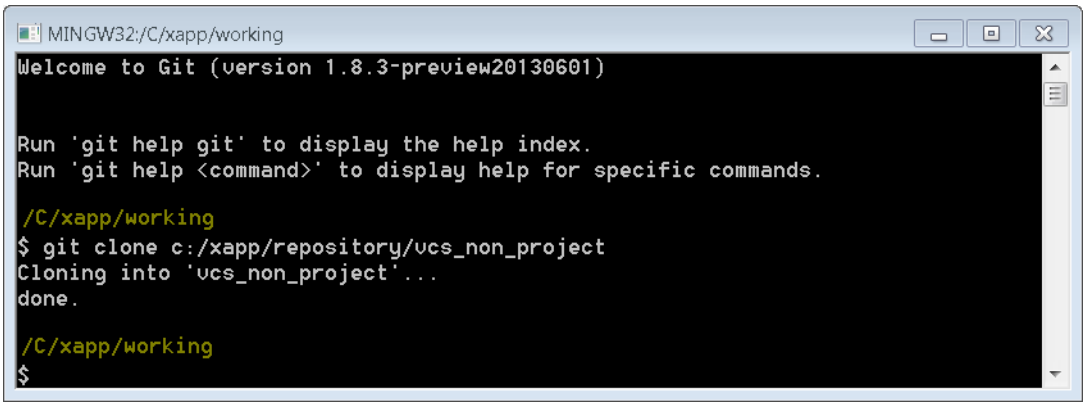

図 15 : ベア リポジトリのクローンを作業リポジトリに作成

4. サンプル デザインのアーカイブ ファイル `xapp1165.zip` を任意のディレクトリに展開します。

5. `vcs_non_project` ディレクトリ以下のすべてのサブディレクトリを作業リポジトリへコピーします。次に例を示します。`C:/xapp/working/vcs_non_project`

   図 16 に、作業リポジトリのディレクトリおよびファイル構造を示します。

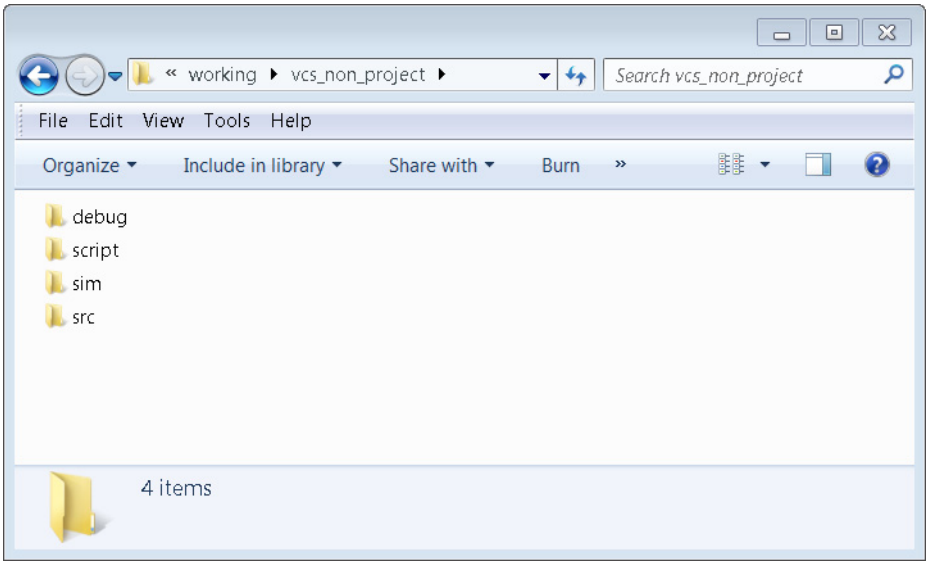

図 16 : 作業リポジトリ

6. Vivado Design Suite のコマンド プロンプト ウィンドウで `run` ディレクトリを作成します。次に例を示します。`C:/xapp/working/vcs_non_project/run`
7. `run` ディレクトリに移動し、次のコマンドを実行して `git_non_project.bat` ファイルを作成します。

```
vivado -mode batch -source ../script/vivado_non_project.tcl -notrace
-tclargs -cmd vcs -ifn ../script/file_list.txt -top spectrum_top -part
xc7k325tffg900-2 -out git_non_project.bat
```

8. Windows エクスプローラーで、`run` ディレクトリへ移動します。次に例を示します。`C:/xapp/working/vcs_non_project/run` です。
9. 右クリックし、[Git Bash] をクリックします。
10. Git Bash ウィンドウで次のコマンドを実行し、すべてのデザイン ソースをチェックインしてこれらを共有リポジトリへプッシュします。

```
git_non_project.bat
```

非プロジェクト モードでの共有リポジトリのテスト

共有リポジトリ内にある最小限のファイルセットの中に、非プロジェクト モードでデザインをインプリメントするのに必要なファイルがすべて含まれていることを確認しておく必要があります。これを確認するには、次の手順を実行します。

1. Git Bash ウィンドウで次のコマンドを実行し、共有リポジトリのクローンとテスト リポジトリを作成します (図 17)。

   ```
   $ cd /C/xapp/test
   $ git clone c:/xapp/repository/vcs_non_project
   ```

   注記：テスト リポジトリのルートは、`vcs_ip_location` と同じディレクトリに設定する必要があります。

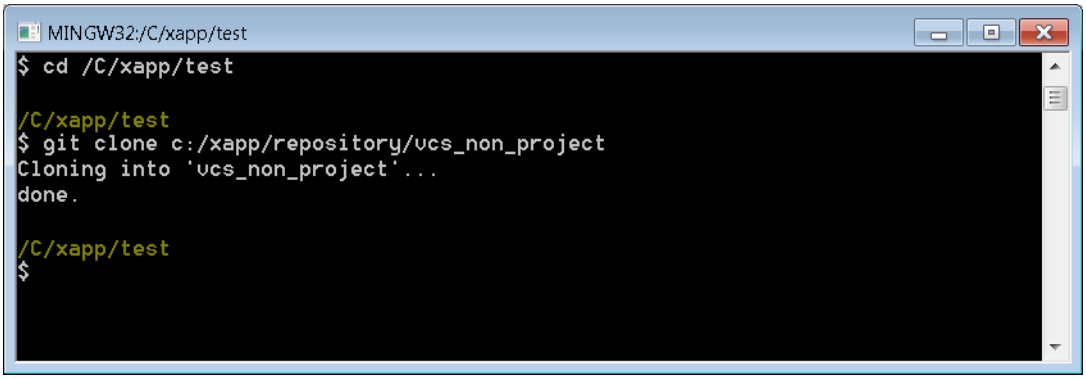

図 17 : 作業リポジトリのクローンをテスト リポジトリに作成

図 18 に、テスト リポジトリのディレクトリおよびファイル構造を示します。

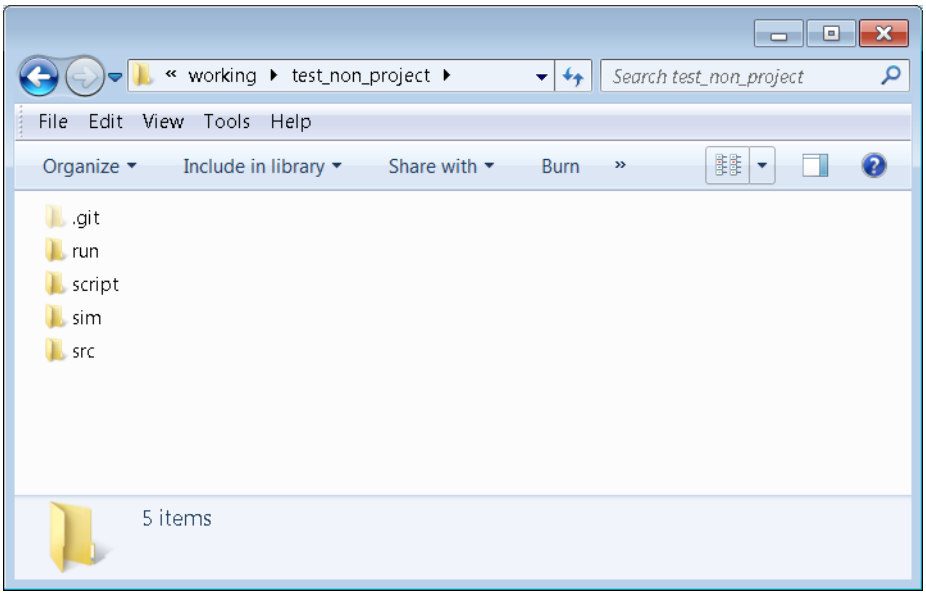

図 18 : テスト リポジトリ

2. Windows エクスプローラーで、テスト ディレクトリに `run` という名前のディレクトリを作成します。次に例を示します。`C:/xapp/test/test_non_project`

3. Windows コマンド プロンプトを開き、Vivado Design Suite 環境設定バッチ ファイルを実行して Vivado Design Suite のコマンド プロンプトを開きます (図 19)。次に例を示します。

```
C:\Xilinx\Vivado\2013.2\settings64.bat
```

図 19 : Vivado Design Suite 環境設定バッチ ファイル

4. run ディレクトリに移動し、次のコマンドを実行して非プロジェクト モードでデザインをインプリメントします。

```
vivado -mode batch -source ../script/non-project_run.tcl -notrace
-tclargs -cmd run -ifn ../script/file_list.txt -top spectrum_top -part
xc7k325tffg900-2
```

この run スクリプトはすべての IP を生成し、合成およびインプリメンテーション、そしてビットストリームの生成を実行します。

リファレンス デザイン

このアプリケーション ノートに含まれるサンプル プロジェクトは完全なスペクトラム解析デザインで、RTL からビットストリーム作成までを実行し、KC705 評価ボードをターゲットにして Vivado ロジック解析でハードウェア デバッグを行うことができます。このデザインは Direct Digital Synthesizer (DDS) Compiler コアを使用して正弦波信号を生成し、これを Finite Impulse Response (FIR) Compiler コアおよび Fast Fourier Transform (FFT) コアで処理します。デジタル信号処理 (DSP) の出力は、ビヘイビア RTL でインプリメントした複素乗算器に渡して振幅を計算します。このプロジェクトは、ユーザーが管理するディレクトリとツールが管理するディレクトリの両方にあるデザイン ソースを使用し、RTL、IP-XACT IP、シミュレーション テストベンチ、コンフィギュレーション ファイル、Vivado ロジック解析コア Inserter ファイルなど複数のソース タイプをサポートする標準的なデザイン環境です。このサンプル プロジェクトは、Vivado Design Suite 2013.2 で検証しています。

このアプリケーション ノートのリファレンス デザインは、次のサイトからダウンロードできます。
https://secure.xilinx.com/webreg/clickthrough.do?cid=344210

表 2 に、リファレンス デザインの詳細を示します。

表 2 : リファレンス デザインの詳細

パラメーター	説明
全般	
開発者	Jim Wu
ターゲット デバイス	Kintex-7 FPGA
ソース コードの提供	あり
ソース コードの形式	VHDL、Verilog
既存のザイリンクス アプリケーション ノート/リファレンス デザイン、CORE Generator™ ツール、サードパーティからデザインへのコード/IP の使用	なし
シミュレーション	
論理シミュレーションの実施	あり
タイミング シミュレーションの実施	N/A
論理シミュレーションおよびタイミング シミュレーションでのテストベンチの利用	あり
テストベンチの形式	Verilog
使用したシミュレータ/バージョン	Vivado simulator 2013.2
SPICE/IBIS シミュレーションの実施	N/A
インプリメンテーション	
使用した合成ツール/バージョン	Vivado synthesis 2013.2
使用したインプリメンテーション ツール/バージョン	Vivado implementation 2013.2
スタティック タイミング解析の実施	あり
ハードウェア検証	
ハードウェア検証の実施	あり
使用したハードウェア プラットフォーム	KC705 評価ボード

参考資料

この文書では、次の参考資料を使用しています。

1. 『Vivado Design Suite ユーザー ガイド : デザイン フローの概要』(UG892)
2. 『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835)
3. ザイリンクス オープンソース Wiki (http://wiki.xilinx.com/)
4. Git のウェブサイト (http://git-scm.com/)

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	内容
2013 年 8 月 5 日	1.0	初版

Notice of Disclaimer

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

本資料は英語版 (v1.0) 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。