PlanAhead ソフトウェア チュートリアル

ChipScope を使用したデバッグ

UG 677 (v 12.2) 2010 年 7 月 23 日

XILINX®

PlanAhead™ ソース コードには、次のプログラムのソース コードが使用されています。

Centerpoint XML

- The initial developer of the original code is CenterPoint – Connective Software
- Software Engineering GmbH. portions created by CenterPoint – Connective Software
- Software Engineering GmbH. are Copyright© 1998-2000 CenterPoint - Connective Software Engineering GmbH. All Rights Reserved. Source code for CenterPoint is available at http://www.cpointc.com/XML/

NLView Schematic Engine

- Copyright© Concept Engineering.

Static Timing Engine by Parallax Software Inc.

- Copyright© Parallax Software Inc.

Java Two Standard Edition

- Includes portions of software from RSA Security, Inc. and some portions licensed from IBM are available at http://oss.software.ibm.com/icu4j/
- Powered By JIDE – http://www.jidesoft.com

The BSD License for the JGoodies Looks

Copyright© 2001-2010 JGoodies Karsten Lentzsch. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of JGoodies Karsten Lentzsch nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

Free IP Core License

This is the Entire License for all of our Free IP Cores.

Redistribution and use in source, netlist, binary and silicon forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of ASICS World Services, the Authors and/or the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

目次

ChipScope を使用したデバッグ 7
はじめに.....7
サンプル デザイン データ 7
ザイリンクス ISE および PlanAhead ソフトウェア 8
ハードウェア要件.....8
PlanAhead のマニュアルと情報 8
チュートリアルの説明 9
チュートリアルの目標 9
チュートリアルの手順 10
プロジェクトを開く 手順 1 11
Set Up ChipScope ウィザードの実行 手順 2 13
デバッグ ネットの追加 手順 3.....19
デバッグ コア属性の変更 手順 4 24
ChipScope デバッグ コアのインプリメント 手順 5.....26
デザインのインプリメンテーション 手順 6.....30
ビットストリームの生成 - ChipScope Analyzer の起動 手順 7.....31
まとめ 32

PlanAhead ソフトウェア チュートリアル

ChipScope を使用したデバッグ

はじめに

このチュートリアルでは、

- ChipScope™ デバッグ ツール使用して PlanAhead™ でデザインをデバッグする方法について説明します。
- PlanAhead を使用して ChipScope デバッグ コアをデザインに挿入するフローを簡単に説明します。
- デバッグ用ネットを選択し、CORE Generator™ を使用してデバッグ コアの生成、インスタンシエーション、接続、合成などを自動化するウィザード インターフェイスの使用方法を説明します。
- ChipScope コアを使用してデザインをインプリメントし、配置およびタイミング レポートをインポートし、BitGen を実行し、ChipScope Analyzer ツールを起動する方法について説明します。
- このチュートリアルは ISE® Design Suite 12.1 を使用して記述されています。このバージョンおよびそれ以降のバージョンがインストールされていることをご確認ください。
- このチュートリアルでは、ISE® Design Suite の PlanAhead™ ソフトウェア製品に含まれる機能の一部だけを使用しています。

PlanAhead の解析機能の詳細は、ほかのチュートリアルで紹介しています。すべてのコマンド オプションについて説明されているわけではりあせん。このチュートリアルでは、ISE Design Suite ソフトウェアの一部として含まれる PlanAhead ソフトウェアの機能を使用しています。チュートリアルに関する質問および問題は、ザイリンクス テクニカル サポート (ホットライン) までご連絡ください。

サンプル デザイン データ

このチュートリアルでは、PlanAhead ソフトウェアをインストールすると含まれるサンプル デザイン データを使用します。サンプル デザイン データは、次のディレクトリにあります。

<*ISE_install_Dir*>/PlanAhead/testcases/**PlanAhead_Tutorial.zip**

書き込み権のあるディレクトリに ZIP ファイルを保存し、抽出します。チュートリアルでは、解凍ファイルのディレクトリを <*Install_Dir*> と記述しています。

チュートリアルのサンプル データは、チュートリアルを実行中に変更されます。各チュートリアルを実行する前に、まず元の PlanAhead_Tutorial データのコピーを取っておいてください。サンプル デザインの詳細は、「チュートリアルの説明」セクションを参照してください。

ザイリンクス ISE および PlanAhead ソフトウェア

PlanAhead ソフトウェアは、ISE Design Suite 12.1 ソフトウェアをインストールするとインストールされます。チュートリアルを始める前に、PlanAhead が起動できるか、サンプル デザイン データがインストールされているかを確認してください。ソフトウェアのインストール方法および詳細は、次のザイリンクス サイトから『ISE Design Suite 12 : インストール、ライセンス、リリース ノート』を参照してください。

http://japan.xilinx.com/support/documentation/sw_manuals/xilinx12_2/irn.pdf

ハードウェア要件

ターゲット デバイスが大規模の場合、2GB 以上の RAM 容量が必要です。このチュートリアルでは、小型のデザインを使用し、1 度に開くことができるデザインの数を制限していますので、1GB で十分ですが、パフォーマンスに影響のでることもあります。

PlanAhead のマニュアルと情報

PlanAhead ソフトウェアの詳細については、次のマニュアルを参照してください。

- 『PlanAhead ユーザー ガイド』(UG632) – PlanAhead ソフトウェアに関する詳細情報
  http://japan.xilinx.com/support/documentation/sw_manuals/xilinx12_2/PlanAhead_UserGuide.pdf
- 『フロアプラン手法ガイド』(UG633) – フロアプランのヒント情報
  http://japan.xilinx.com/support/documentation/sw_manuals/xilinx12_2/Floorplanning_Methodology_Guide.pdf
- 『階層デザイン手法ガイド』(UG748) – PlanAhead の階層デザインの概要
  http://japan.xilinx.com/support/documentation/sw_manuals/xilinx12_2/Hierarchical_Design_Methodology_Guide.pdf
- ビデオ デモなど、PlanAhead のその他の情報については、http://www.xilinx.com/planahead を参照してください。

チュートリアルの説明

この演習では、Wishbone バス アービタを介して複数のペリフェラル コアに接続される RISC CPU コアを含む小型のサンプル デザインを使用します。Verilog および VHDL 形式のソース ファイルが含まれています。

このデザインは、XC6VLX75TFF784 デバイスをターゲットにしています。このチュートリアルでは、既に合成済みでインプリメンテーション可能な HDL を含むプロジェクト ファイルを使用します。

チュートリアルのデザイン データは、チュートリアルを実行すると変更されるため、各チュートリアルを実行する前に、まず元の PlanAhead_Tutorial データのコピーを取っておいてください。チュートリアル データの詳細は、本書の「サンプル デザイン データ」セクションを参照してください。

チュートリアルに関する質問および問題は、ザイリンクス テクニカル サポート (ホットライン) までご連絡ください。

チュートリアルの目標

After completing this tutorial, you will have:

- Set Up ChipScope ウィザードを使用した ChipScope デバッグ コアの挿入
- デバッグ ネットの追加と ChipScope デバッグ コアの再生成
- ChipScope デバッグ コアのデフォルト オプションの変更
- ChipScope デバッグ コアのインプリメンテーション
- PlanAhead での ChipScope デバッグ コアを含むデザインのインプリメンテーション

チュートリアルの手順

チュートリアルは各手順に分けられ、それぞれで大まかな手順が説明された後、細かい手順が説明されていますので、スキル レベルに合った方の手順を参照してください。

大まかな手順でわからない場合はその後の詳細な手順を参照してください。既に手順を理解している場合は、その部分は飛ばして次の手順に進んでください。

このチュートリアル次の 7 手順で構成されます。

手順 1 : プロジェクトを開く

手順 2 : Set Up ChipScope ウィザードの実行

手順 3 : デバッグ ネットの追加

手順 4 : デバッグ コア属性の変更

手順 5 : ChipScope デバッグ コアのインプリメント

手順 6 : デザインのインプリメンテーション

手順 7 : ビットストリームの生成 - ChipScope Analyzer の起動 (オプション)

プロジェクトを開く 手順 1

PlanAhead では、使用されるデザイン フローの段階によってさまざまなタイプのプロジェクトを作成できます。RTL ソースまたは合成済みネットリストは、開発、解析、またはインプリメンテーション ～ ビット ファイル生成までのプロジェクトを作成するために使用できます。このチュートリアルでは、まだインプリメントされていない合成済みネットリスト プロジェクトを使用します。

1-1. ソフトウェアを起動します。

1-1-1. Windows では、デスクトップ アイコンをダブルクリックするか、[スタート] → [プログラム] → [Xilinx ISE Design Suite 12.2] → [PlanAhead] → [PlanAhead] をクリックします。

1-1-2. Linux の場合は、<*Install_Dir*>/PlanAhead_Tutorial/Tutorial_Created_Data ディレクトリに移動し、**planAhead** と入力します。

PlanAhead の Getting Started ページが開きます。

1-2. project_cpu_netlist プロジェクトを開きます。

サンプル デザイン プロジェクトのネットリストは、チュートリアルを実行中に変更されます。元のチュートリアル デザイン プロジェクトを後で再利用できるように、プロジェクトを別名で保存します。

1-2-1. Getting Started ページで [Open Example Project] → [CPU (Synthesized)] をクリックします。

1-2-2. **[File]** → **[Save Project As]** をクリックし、プロジェクトを別名で保存します。

[Save Project As] ダイアログ ボックスが開きます。

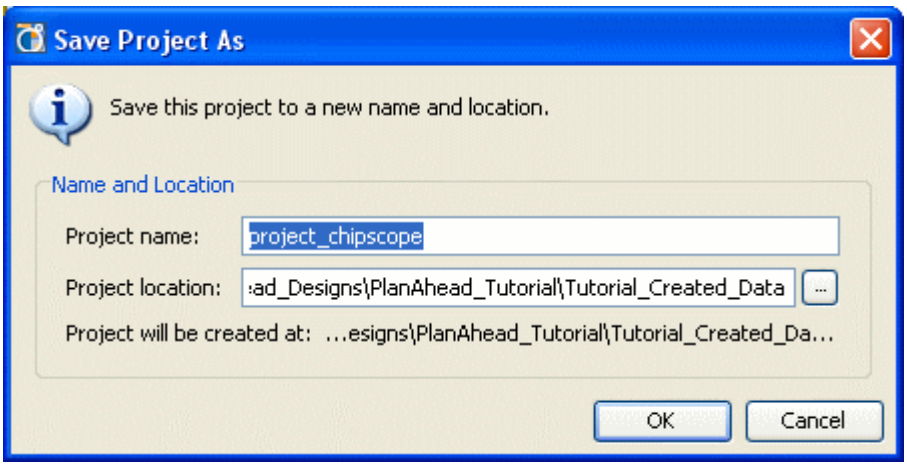

図 1 : プロジェクトの保存

1-2-3. [Project name] テキスト ボックスにプロジェクト名を入力します (例 : **project_chipscope**)。

1-2-4. プロジェクト ディレクトリに次を指定します。

<Install_Dir>/**PlanAhead_Tutorial/Tutorial_Created_Data/**

1-2-5. **[OK]** をクリックします。

Project Manager が開き、[Sources] ビューにデザイン ソースが表示されます。

1-2-6. PlanAhead 環境の左側の Flow Navigator で **[Netlist Design]** をクリックします。

[Netlist Design] が開き、デバッグ ロジックを挿入する準備ができました。次の図を参照してください。

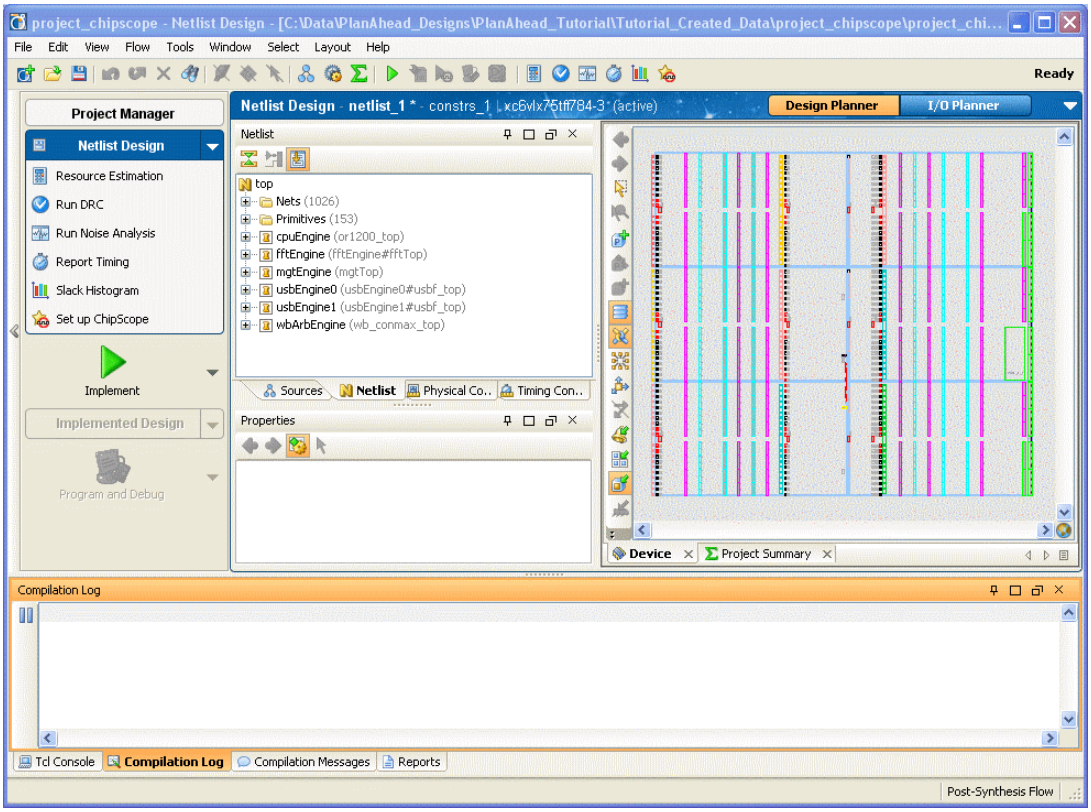

図 2 : [Netlist Design] 環境のプロジェクト

Set Up ChipScope ウィザードの実行 手順 2

ChipScope デバッグ コアの挿入フローを使用して、Integrated CONtroller (ICON) および Integrated Logic Analyze (ILA) コアをコンフィギュレーションし、合成済みネットリスト プロジェクトにインスタンシエートします。

Set Up ChipScope ウィザードでは、次が実行できます。

- デバッグ用ネットの選択
- 必要なコア数の指定
- これらのネットへのコア トリガ ポートのインスタンシエートと接続

2-1. wbArbEngine/m0/wb* ネットを選択し ChipScope コアに接続します。

2-1-1. Flow Navigator で **[Up ChipScope]** をクリックします。

Set Up ChipScope ウィザードが開きます。

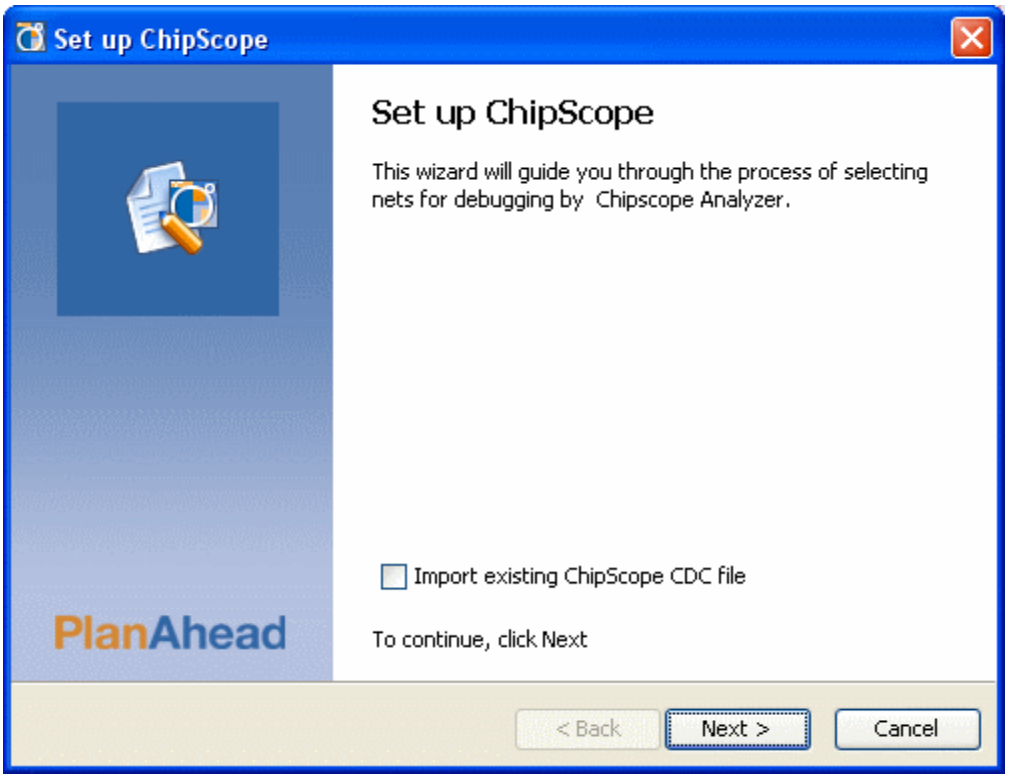

図 3 : Set Up ChipScope ウィザード

2-1-2. **[Next]** をクリックします。

[Specify Nets to Debug] ページが表示されます。

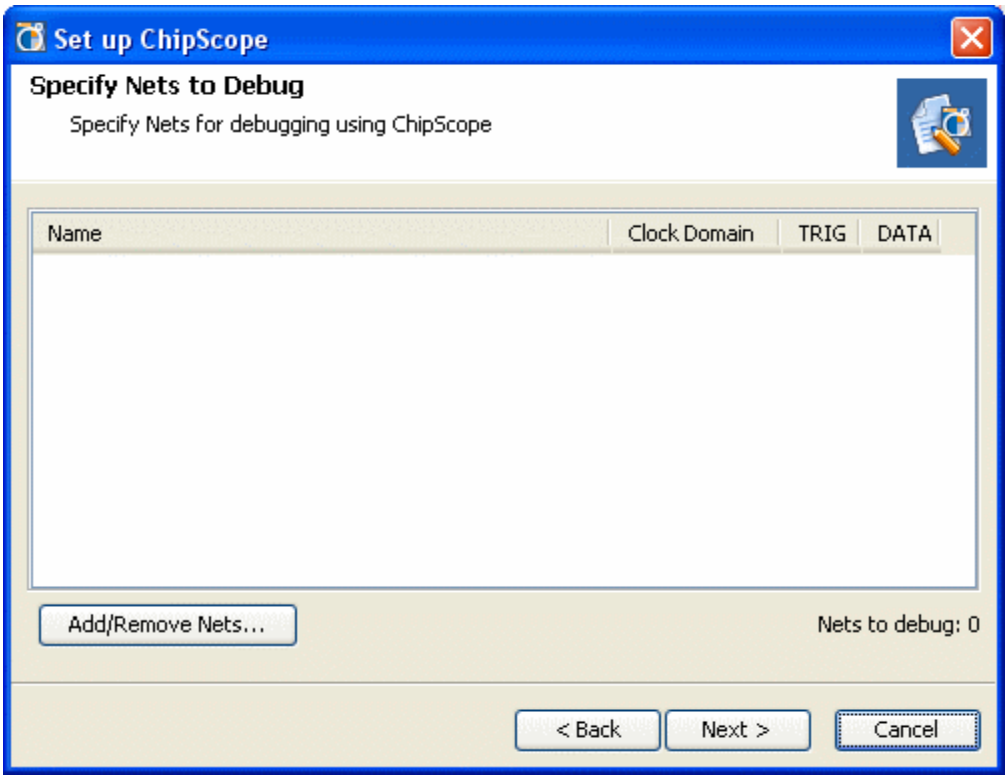

図 4 : [Specify Nets to Debug] ページ

2-1-3. [Add/Remove Nets] をクリックします。

[Add/Remove Nets] ダイアログ ボックスが表示されます (図 5)。

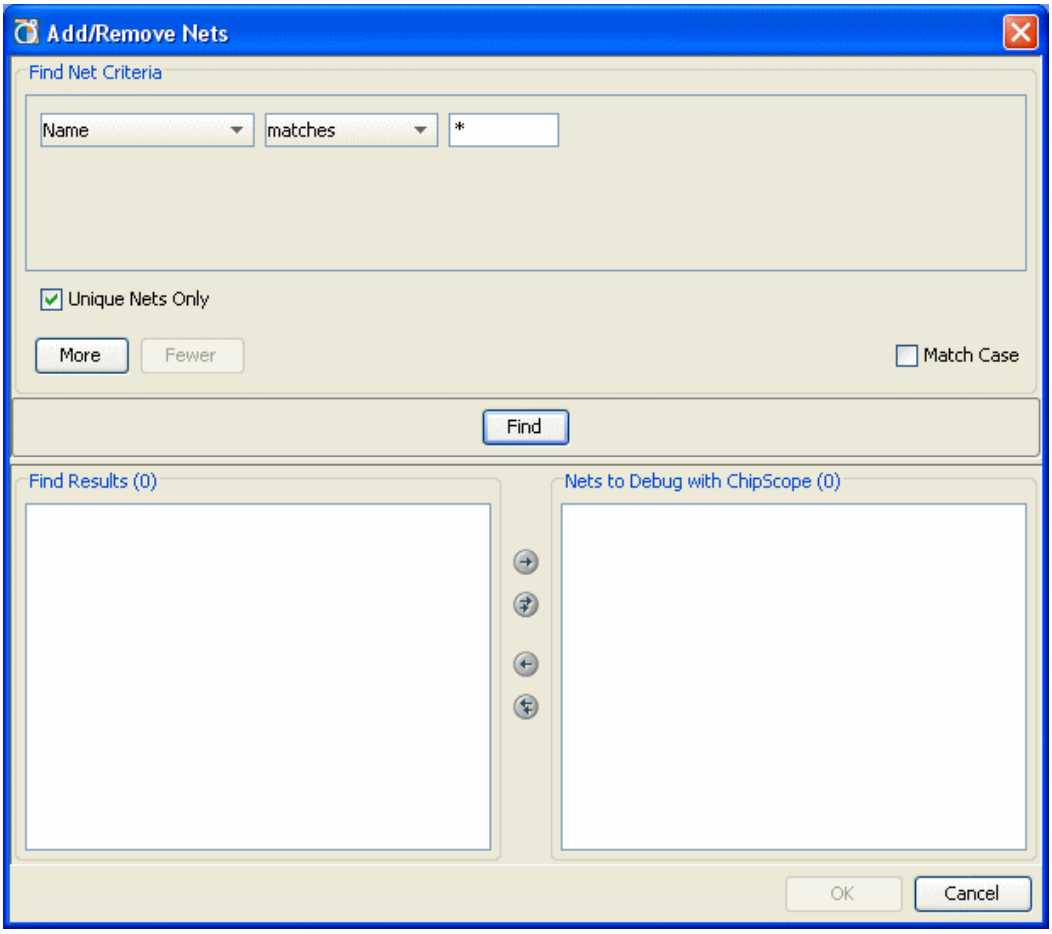

図 5 : [Add/Remove Nets] ダイアログ ボックス

2-1-4. [Find Net Criteria] のテキスト ボックスに **wbArbEngine/m0/wb_*** と入力します。

2-1-5. **[Find]** をクリックします。

[Add/Remove Nets] ダイアログ ボックスの [Find Results] ボックスに結果が表示されます (図 6)。

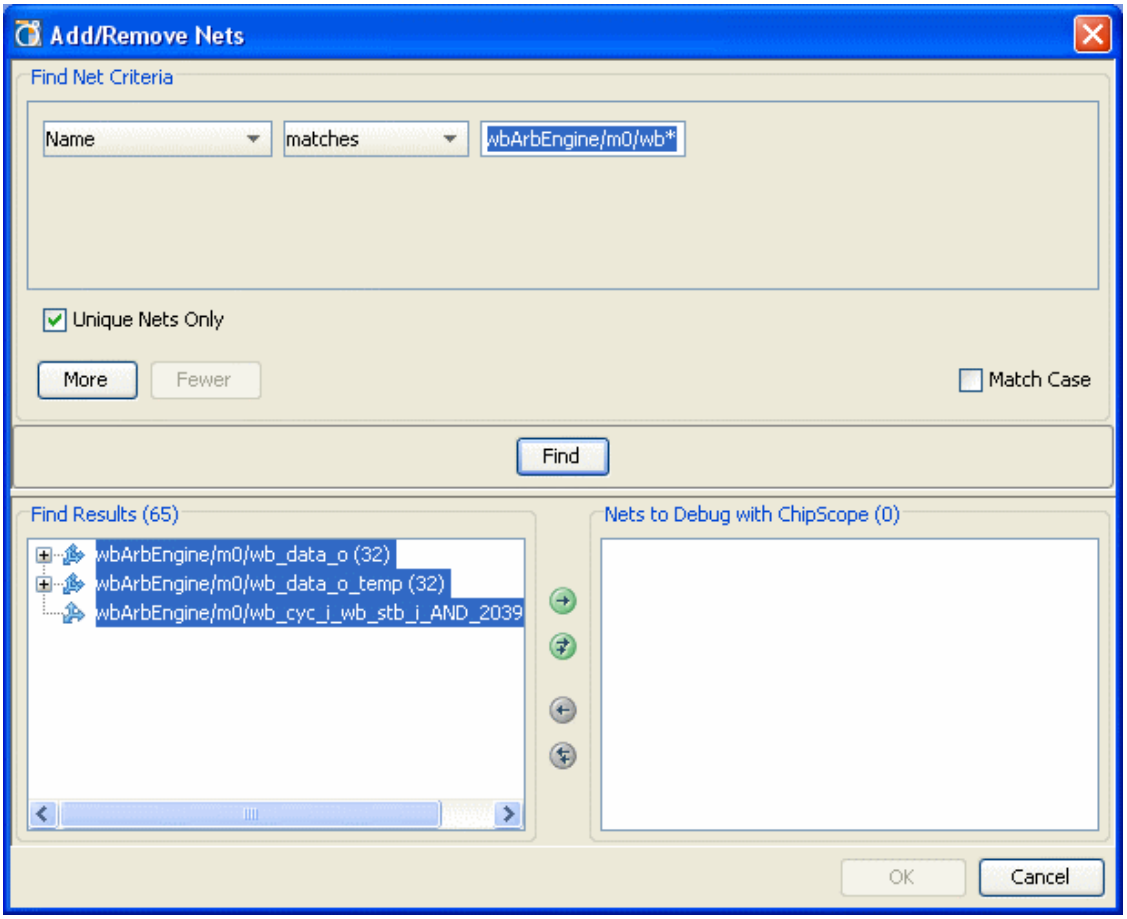

図 6 : [Add/Remove Nets] ダイアログ ボックスの検索結果表示

2-1-6. [Move all Nets to the Right] ボタン ( ) をクリックし、すべてのネットを [Nets to Debug with ChipScope] リストに移動します。

[Add/Remove Nets] ダイアログ ボックスの [Nets to Debug with ChipScope] に移動したネットが表示されます。

2-1-7. **[OK]** をクリックします。

Set Up ChipScope ウィザードの [Specify Nets to Debug] ページが表示されます (図 7)。

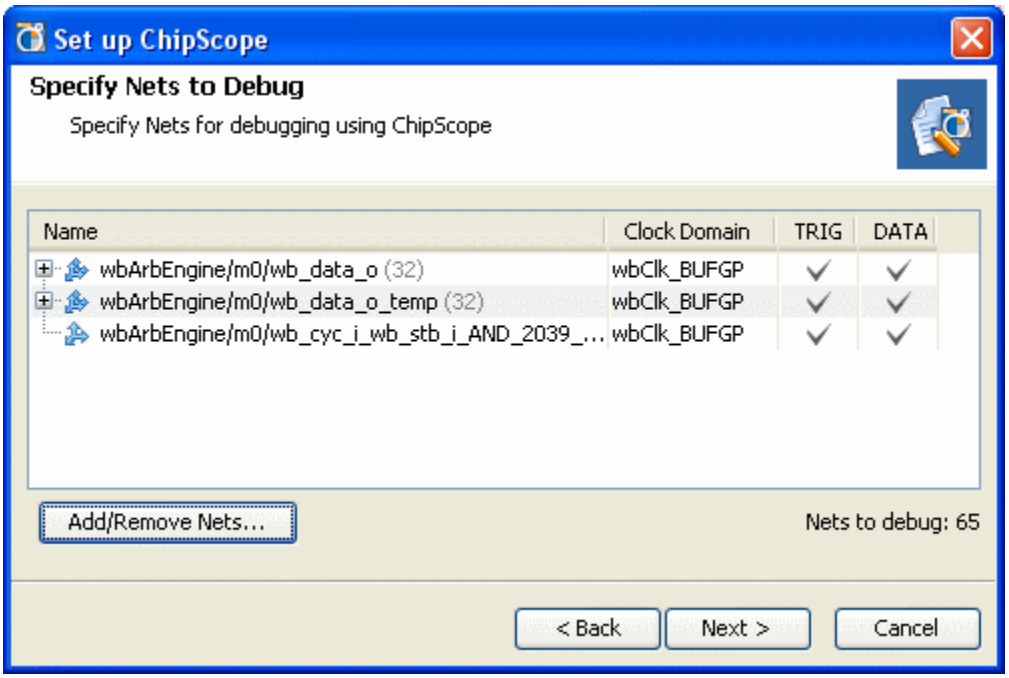

図 7 : [Specify Nets to Debug] ページ

ネットのクロックドメインはすべて同じなので、必要なのは ILA コア 1 つだけです。

2-1-8. **[Next]** をクリックします。

サマリ ページが表示されます。

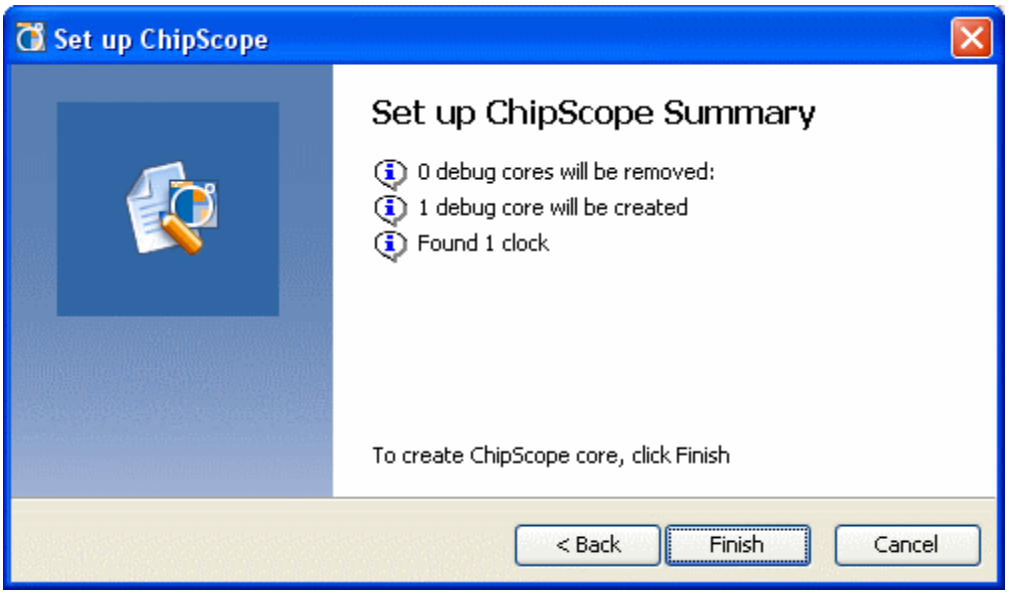

図 8 : [Set Up ChipScope Summary] ページ

2-1-9. **[Finish]** をクリックします。

[Netlist Design] ビューに戻りますが、今度は [ChipScope] ビューが表示されます。

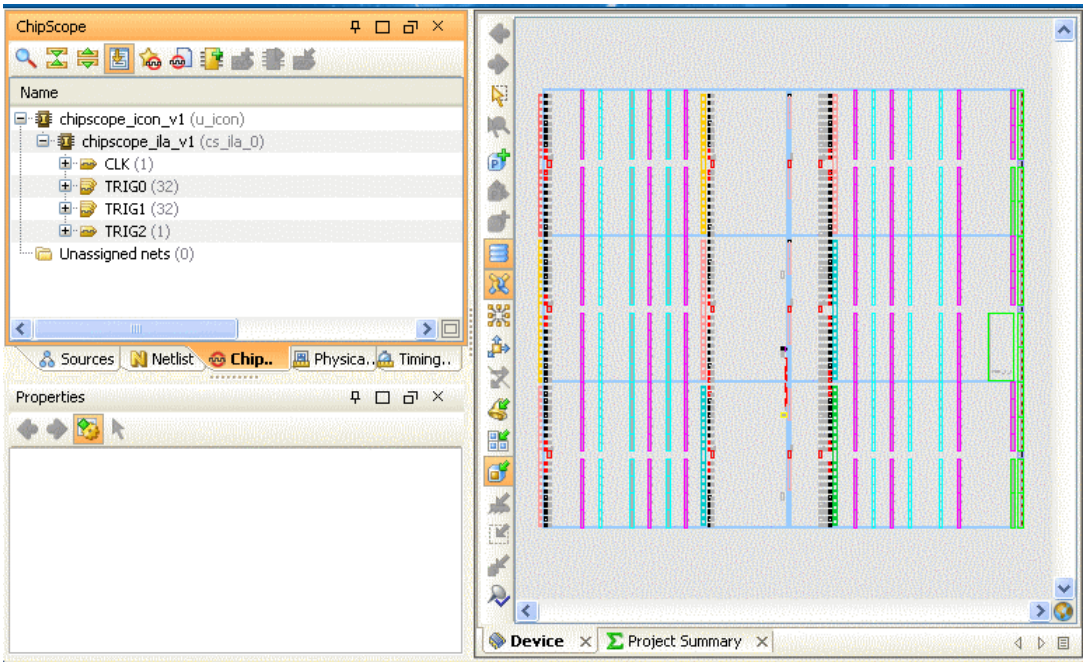

図 9 : ChipScope デバッグ コアの挿入済み PlanAhead 環境

[ChipScope] ビューには、生成されたコアおよびまだ割り当てられていないネットに関する情報が含まれます。コアのコンフィギュレーションと管理は、このビューで実行されます。このチュートリアルでは後でこのビューを使用します。

2-2. [Netlist] ビューで ILA および ICON コアを表示します。

2-2-1. **[Netlist]** タブをクリックします。

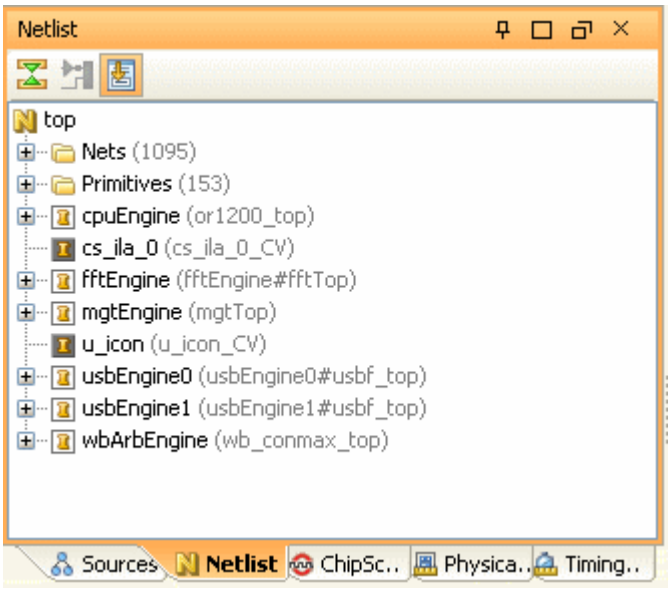

図 10 : ネットリスト デザインのデバッグ コアのブラック ボックス インスタンス

u_icon および cs_ila_0 は、ネットリストに挿入された新しいインスタンスです。どちらにも [Netlist] ビューでブラック ボックスのアイコン ( ) が付いています。

デバッグ ネットの追加 手順 3

これで、ChipScope デバッグに接続されたネットを追加し、デザインにインプリメントする準備ができました。ここでは、別の方法を使用して、コアをデバッグして再生成するためのネットをさらに追加する手順について説明します。既にデバッグされたネットのリストについてこれ以上の説明が必要ない場合は、この手順は飛ばしてください。

3-1. Set Up ChipScope ウィザードを使用してデバッグ コアをリコンフィギュレーションします。

3-1-1. [Netlist] ビューの cpuEngine 階層の横の アイコンをクリックし、CPU モジュールの下の階層を展開します。

3-1-2. cpuEngine モジュールの下の iwb_biu 階層の横の アイコンをクリックし、CPU の命令フェッチ モジュールの Wishbone バスの下の階層を展開します。

3-1-3. iwb_biu 階層の [Nets] フォルダの横の アイコンをクリックし、Wishbone インターフェイス内のネットのリストを展開します。

3-1-4. [Nets] フォルダの下の wb_adr_o バスをクリックし、32 ビットの Wishbone アドレス出力ベクタすべてを選択します。

PlanAhead の [Netlist] ビューが表示されます (図 11)。

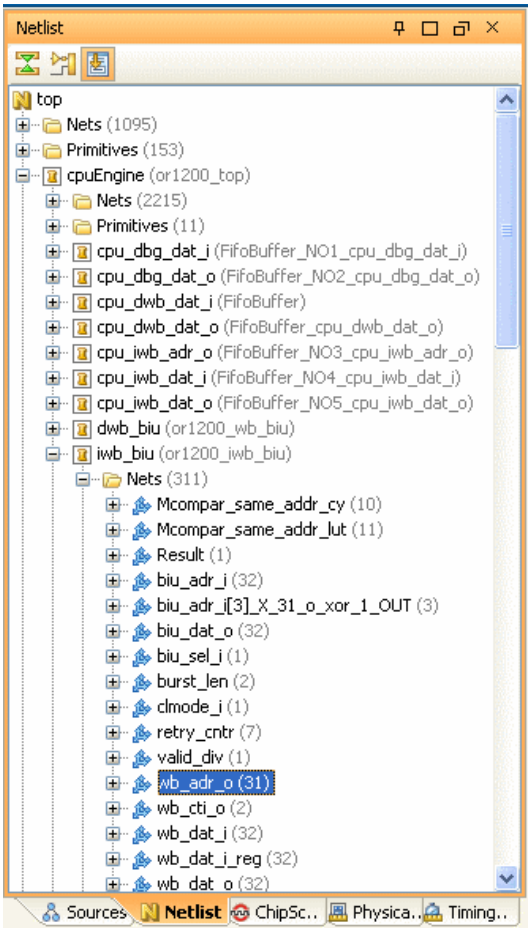

図 11 : wb_adr_o バスが選択された状態の [Netlist] ビュー

3-1-5. Flow Navigator で **[Set up ChipScope]** をクリックし、Set Up ChipScope ウィザードを開きます。

Set Up ChipScope ウィザードが開きます。

3-1-6. **[Next]** をクリックします。

[Existing Debug Nets] ページが表示されます。

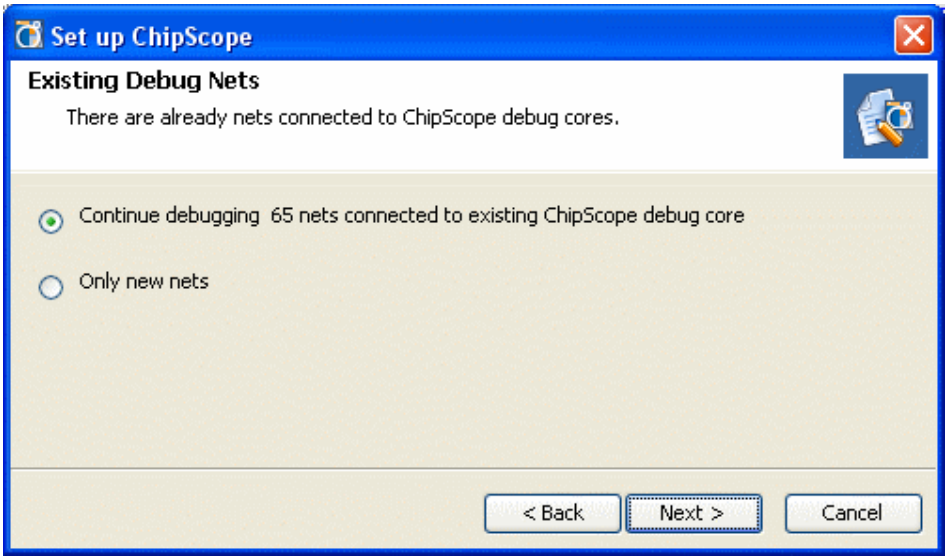

図 12 : Set Up ChipScope ウィザードの [Existing Debug Nets] ページ

3-1-7. **[Next]** をクリックし、前のデバッグ ネットに接続されたコアを新しいネットでリコンフィギュレーションします。

[Specify Nets to Debug] ページが表示されます。デバッグするネットが 32 個追加されています。

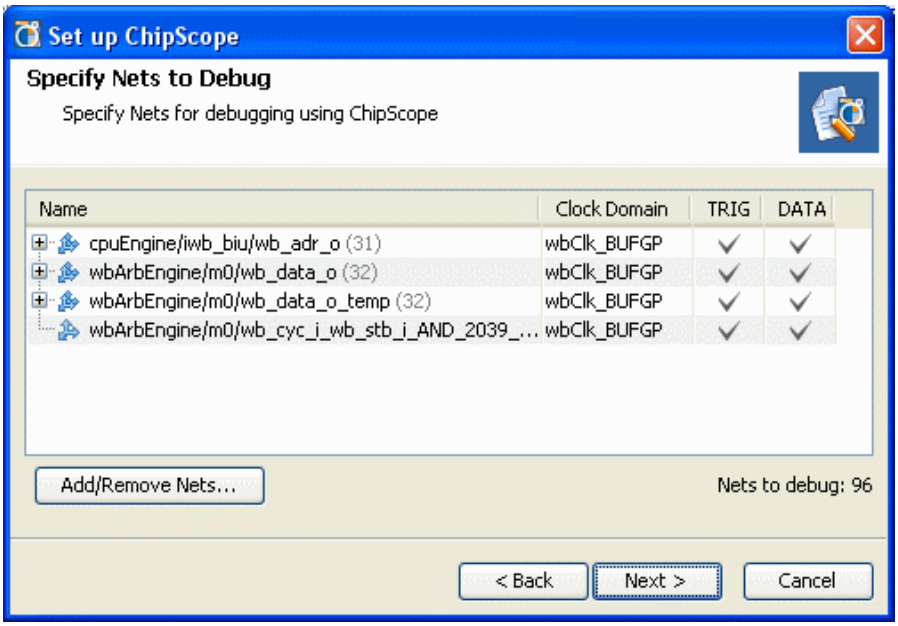

図 13 : Set Up ChipScope ウィザードの [Specify Nets to Debug] ページ

クロックドメイン信号はここでもすべて wbClk_BUFGP です。

3-1-8. **[Next]** をクリックします。

[Existing ChipScope Cores] ページは、前のデバッグ コアを削除して新しいネットを作成するか指定できます。

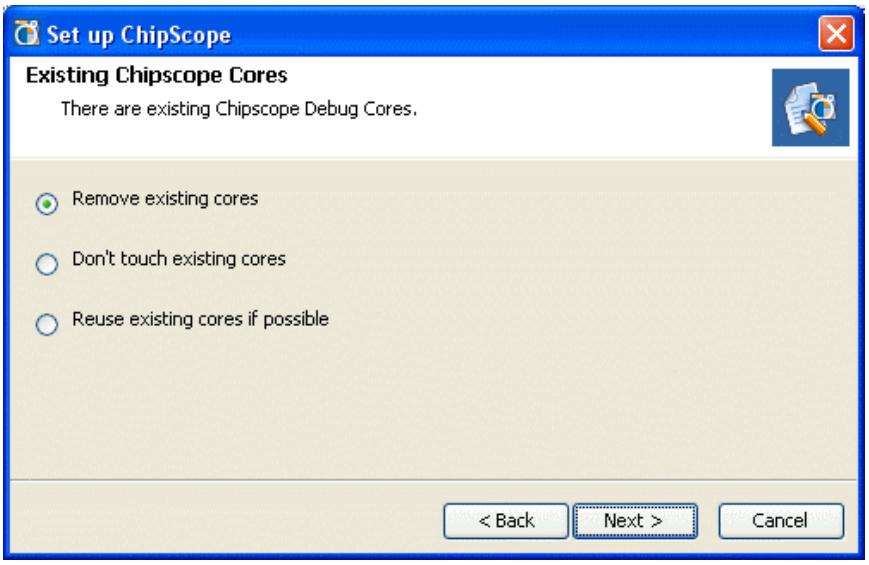

図 14 : Set Up ChipScope ウィザードの [Existing ChipScope Cores] ページ

3-1-9. **[Next]** をクリックし、既存のコアを削除して新しいネットとクロック ドメインのリストに基づいたセットを新しく作成します。

サマリ ページが表示されます。

図 15 : Set Up ChipScope ウィザードのサマリ ページ

3-1-10. **[Finish]** をクリックし、前に定義したコアを削除し、新しいコアを作成します。

デバッグ コア属性の変更 手順 4

ChipScope ILA コアのデフォルト属性を変更できます。

4-1. csdebugcore_1_0 のトリガ ポート TRIG0 の match_type プロパティを変更します。

4-1-1. [ChipScope] ビューのタブをクリックし、手前に表示します。

4-1-2. chipscope_ila_v1 インスタンスをクリックします。

4-1-3. chipscope_ila_v1 の TRIG0 デバッグ ポートをクリックします。

4-1-4. [Debug Port Properties] ビューで [Options] タブをクリックします。

4-1-5. [Debug Port Properties] ビューで match_type の右列をクリックしてドロップ ダウン リストを表示します。

4-1-6. **[extended_with_edges]** を選択して、トリガ ポートのマッチ タイプを変更します。

[Debug Port Properties] ビューは次のようになります。

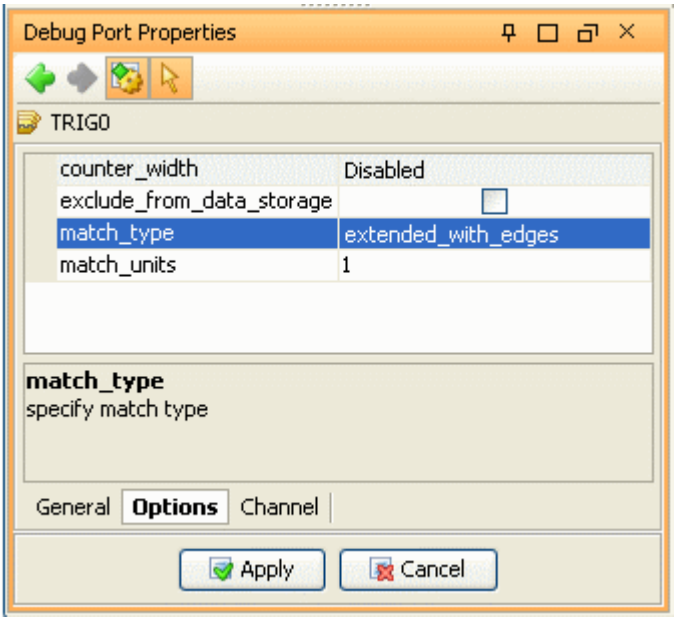

図 16 : match_type を設定した [Debug Port Properties] ビュー

4-1-7. **[Apply]** をクリックし、変更を適用します。

4-2. データ サンプリングの深さを変更します。

データ サンプリングの深さをデフォルトの 1024 から 2048 に変更して、ブロック RAM の容量を増加します。

4-2-1. [ChipScope] ビューで chipscope_ila_v1 をクリックします。

4-2-2. [Debug Core Properties] ビューで **sample_data_depth** をクリックします。

4-2-3. [sample_data_depth] の右の列でドロップ ダウン リストから **2048** を選択します。

[Debug Core Properties] ビューが開きます。

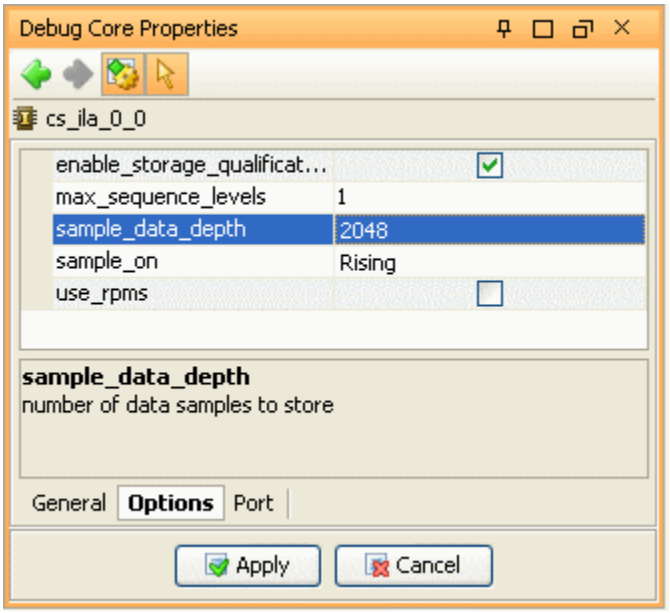

図 17 : sample_data_depth を 2048 に変更した状態の [Debug Core Properties] ビュー

4-2-4. **[Apply]** をクリックして、プロパティの変更を適用します。

ChipScope デバッグ コアのインプリメント 手順 5

この段階で、選択したデバッグ ネットに接続される ChipScope デバッグ コアのブラック ボックス モデルが挿入されています。PlanAhead の **[Run Implementation]** を使用して FPGA をインプリメントすると、NGDBuild、マップ、配置配線でインプリメントされる前に CORE Generator が呼び出され、これらのブラック ボックス デバッグ コアが合成済みのコアに自動的に変換されます。

ただし、デバッグ コアをデバッグするクリティカル ロジックと共にフロアプランする場合は、まず最初にインプリメントする必要があります。

5-1. ChipScope デバッグ コアをインプリメントします。

5-1-1. [ChipScope] ビューのタブをクリックし、**chipscope_ila_v1** コアが選択されていることを確認します。

5-1-2. [Implement ChipScope Debug Cores] ボタン () をクリックします。

5-1-3. **[OK]** をクリックしてプロジェクトを保存します。

これにより、CORE Generator が起動され、前の手順で作成した ChipScope デバッグ コアがコンフィギュレーションされます。この処理は数分かかります。

5-1-4. **[Netlist]** タブをクリックします。

5-1-5. [Collapse All] () をクリックします。

コアはインプリメントされたので、[Netlist] ビューの u_icon および chipscope_ila_v1 インスタンスがブラックボックスからコアに変更されます。

5-1-6. **cs_ila_0_0** インスタンスを展開して選択します。

5-1-7. [Instance Properties] ビューで **[Statistics]** タブをクリックします。

[Primitive Statistics] にコアのインプリメントに使用されたロジックが表示されます。

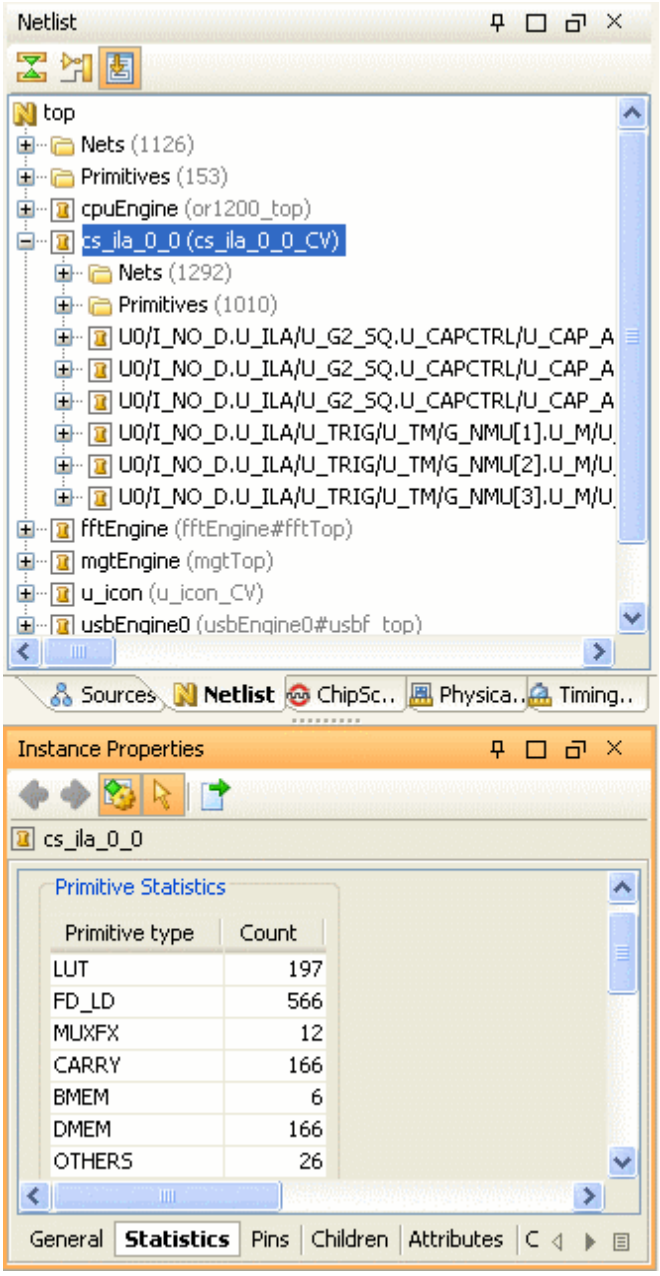

図 18 : デバッグ コアのリソース使用率の表示

5-1-8. [Netlist] ビューで cs_ila_0_0 インスタンスが選択されたままであることを確認します。

5-1-9. [Schematic] ( ) をクリックします。

ILA コアのインスタンシエーションである cs_ila_0_0 の回路図が表示されます。

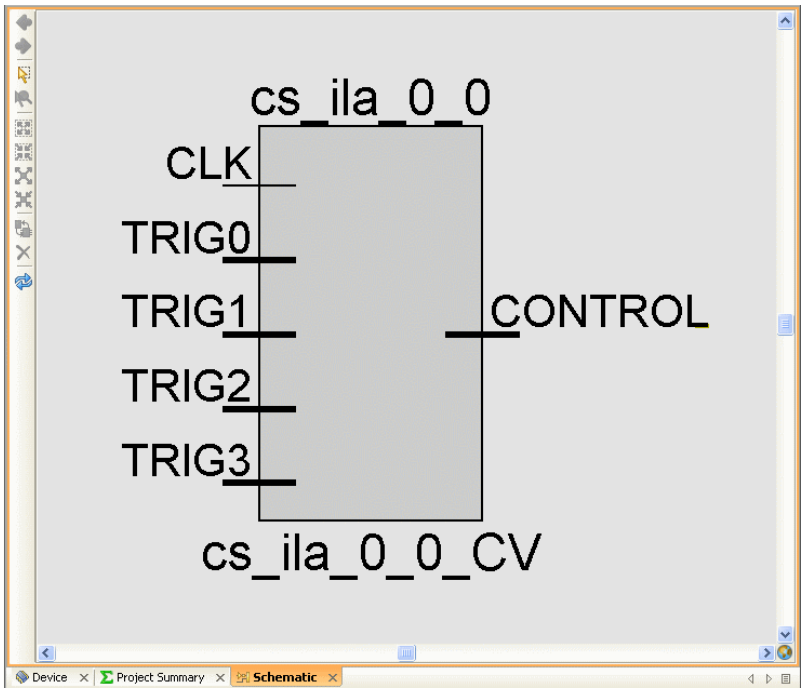

図 19 : cs_ila_0_0 デバッグ コアの回路図

5-1-10. **CLK** ピンと **TRIG0** ピンをダブルクリックし、回路図でこれらのネットを展開します。

5-1-11. [Schematic] ビューで [Regenerate Schematic] ボタン ( ) をクリックします。

5-1-12. 展開した cs_ila_0_0 回路図を拡大します (図 20)。拡大するには、回路図の左上をクリックし、右下に向かってドラッグします。

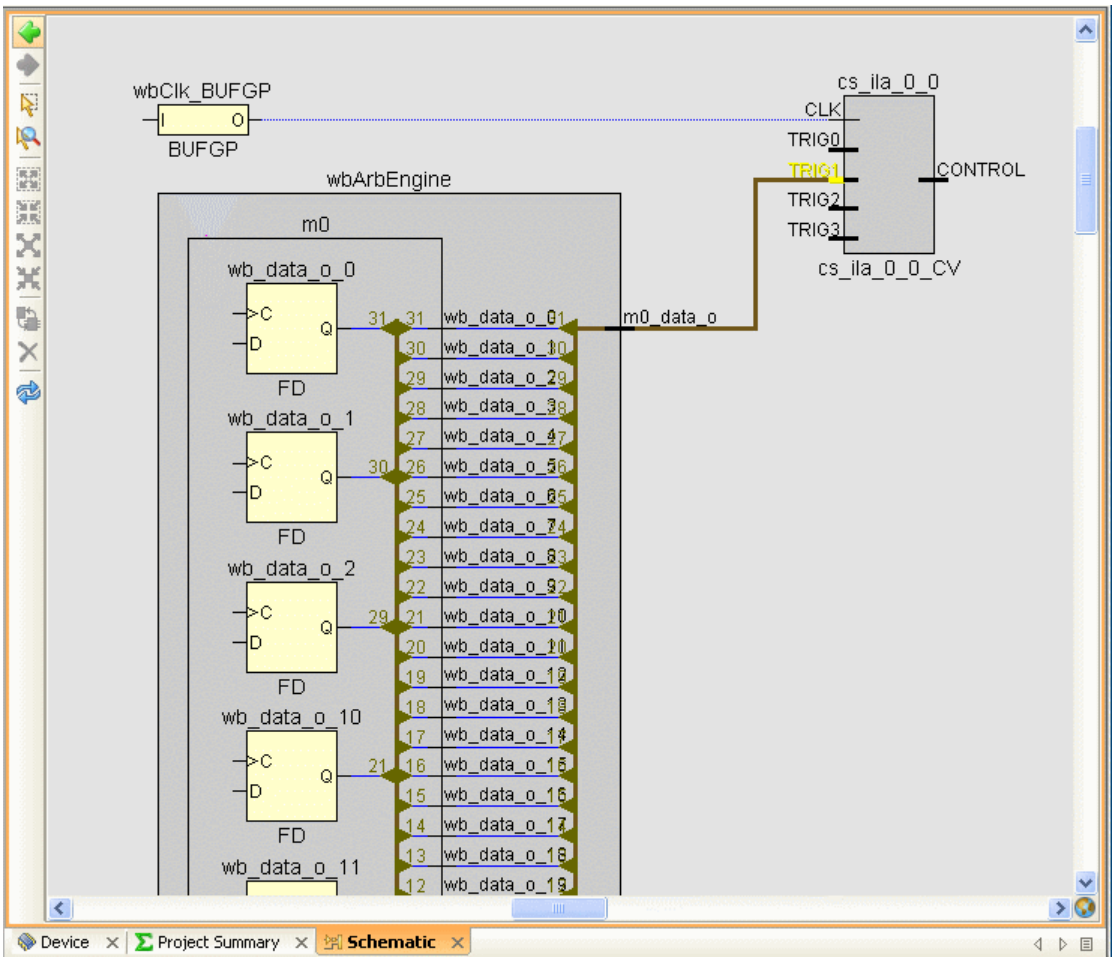

図 20 : PlanAhead で展開した TRIG1 ネットの回路図

これらのトリガ ポートは wbArbEngine から生成された Wishbone インターフェイス アドレス制御信号に接続され、クロック信号の wbClk が使用されます。

フロアプランについてはこのチュートリアルでは説明しませんが、クリティカル ロジックにデバッグ コアを挿入すると、結果に影響が出ますので、ネットはデバッグ用に選択したフリップフロップの出力ピンで直接駆動するのをお勧めします。また、AREA_GROUP 制約 (PlanAhead では Pblock) を使用して、デバッグ コアとデバッグするクリティカル ロジックが近くに配置されるようにまとめ、コアの挿入によるタイミングに影響がなるべく出ないようにする必要のあることもあります。

フロアプランおよびロジック階層を AREA_GROUP 制約に割り当てる方法については、その他の PlanAhead チュートリアルを参照してください。

デザインのインプリメンテーション 手順 6

この段階までで、次を実行しました。

- ChipScope デバッグ コアを作成
- ChipScope デバッグ コアをデバッグ ネットへ接続
- デバッグ コアのオプションをデフォルトから変更
- コアのネットリストを生成

これで、デバイスをインプリメンテーションできます。

PlanAhead を使用した FPGA のインプリメンテーションの詳細については、その他の PlanAhead チュートリアルで説明されています。

6-1. デザインをインプリメントします。

6-1-1. Flow Navigator で **[Implement]** をクリックします。

ISE Design Suite のインプリメンテーションがアクティブな制約セットと run ストラテジを使用して実行されます。

PlanAhead 環境では、[Compilation Log] ビューや [Compilation Messages] ビューにインプリメンテーションの進捗状況が表示されます。

このデザインはインプリメントするのに約 20 分かかります。インプリメンテーションが終了すると、それがステータス バーに表示されます。インプリメンテーションを実行せずに、チュートリアルの残りの手順を進めることもできます。

6-1-2. [Netlist] ダイアログ ボックスを閉じます。メッセージが表示されたら、**[Yes]** をクリックします。

6-1-3. インプリメンテーションが終了したら、[Implementation Complete] ダイアログ ボックスで **[Open Implemented Design]** を選択します。

インプリメントされたデザインが読み込まれます。

6-2. デバッグ コア ロジックの配置をハイライトします。

6-2-1. [Netlist] ビューで Ctrl キーを押しつつ cs_ila_0_0 と u_icon インスタンスの両方を選択します。

6-2-2. インスタンスを右クリックし、[Highlight Primitives] → [Cycle Colors] をクリックします。

デバッグ コアのロジックが別々の色でハイライトされます。

6-2-3. メイン ツールバーで [Unhighlight All] ボタン ( ) をクリックします。

ビットストリームの生成 - ChipScope Analyzer の起動 手順 7

ここまでで、デザインのインプリメンテーションが終了しました。このサンプル デザインにはタイミング エラーが含まれることもありますが、それでもビットストリームを生成してデバイスをプログラムできます。

7-1. インプリメントされた run のビットストリームを生成します。

7-1-1. Flow Navigator で [Program and Debug] → [Generate Bitstream] をクリックします。

[Generate Bitstream] ダイアログ ボックスを開きます。

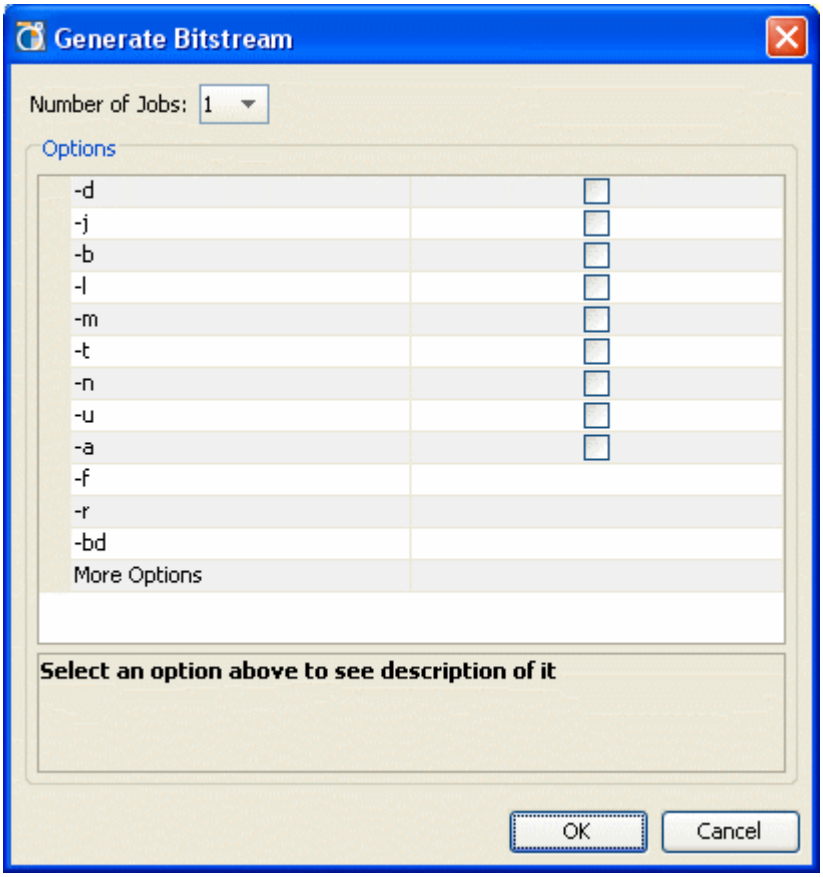

図 21 : [Generate Bitstream] ダイアログ ボックス

7-1-2. **[OK]** をクリックします。

7-1-3. [BitGen Completed] ダイアログ ボックスで **[OK]** をクリックします。

[Generate Bitstream] コマンドが終了すると、通常のプログラム方法を使用してその BIT ファイルを FPGA デバイスにダウンロードできます。

これで ChipScope Analyzer を実行できるようになりました。

7-2. impl_1 ビットストリームに対して ChipScope Analyzer を起動します。

7-2-1. Flow Navigator で [Program and Debug] → [**Launch ChipScope Analyzer**] をクリックします。

ChipScope Analyzer が起動されます。

この手順を実行するには、マシンに機能しているボードを接続しておく必要があります。これはデモ デザインなので、上記の手順は ChipScope Analyzer ツールで実際にサポートされる FPGA デバイスを実際に接続していない限り、実行できません。

7-2-2. **[Close]** をクリックします。

ChipScope Analyzer を閉じます。

まとめ

このチュートリアルでは、次を実行しました。

- サンプル デザインを使用して ChipScope デバッグ コアを設定およびコンフィギュレーション
- デバッグ コアのトリガ ポートに接続するネットを選択
- 選択したネットをデバッグ コアに追加
- デバッグ コアのインプリメンテーションのデフォルト設定を変更
- CORE Generator を呼び出してこれらのデバッグ コアをインプリメントし、コンフィギュレーションされたデバッグ コアを合成して、それらをブラック ボックスからインプリメント済みコアに変換
- デザインをインプリメントし、配置およびタイミング レポートをデザインにバックアノテート
- BitGen を実行
- 完了したデザインに対して ChipScope Analyzer ツールを起動