PlanAhead ソフトウェア チュートリアル

プロセッサ ペリフェラルの
パーシャル リコンフィギュレーション

UG 744 (v 12.1) 2010 年 5 月 3 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

PlanAhead ソフトウェア チュートリアル

プロセッサ ペリフェラルのパーシャル リコンフィギュレーション

はじめに

このチュートリアルでは、Xilinx® Platform Studio (XPS) と PlanAhead™ ソフトウェアを使用してパーシャル リコンフィギュレーション デザインを開発する方法について説明します。XPS は、リコンフィギャブル パーティション 1 つのリコンフィギャブル モジュール 2 つを定義して下位レベル モジュールを含むプロセッサ システムを作成するために使用します。この 2 つのリコンフィギャブル モジュールは、加算と乗算を実行します。

XPS は、エンベデッド デザイン キット (EDK) の一部で、ISE Design Suite の Embedded および System Edition に含まれています。

PlanAhead は、次のために使用します。

1. リコンフィギャブル領域のリコンフィギャブル パーティションの定義などを含めたデザインのフロアプラン
2. 複数コンフィギュレーションの作成と、パーシャル リコンフィギュレーション インプリメンテーション フローを実行したフルおよびパーシャル ビットストリームの生成

   このチュートリアルでは、ML-605 評価ボードを使用してハードウェアでデザインを検証します。 これには、最初に FPGA デバイスをコンフィギュレーションするのにコンパクト フラッシュ (CF) メモリ カードが、その CF に格納されたパーシャル ビットストリーム ファイルを読み込んでデバイスを部分的にリコンフィギュレーションするのに XPS HWICAP ペリフェラルが使用されています。

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。それ以外の機能については、別の PlanAhead チュートリアルで説明します。

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 UG744_design_files.zip リファレンス デザインを使用します。 このファイルは、マシンのいずれかのディレクトリに解凍しておいてください。その際、選択したディレクトリ パス名にはスペースが含まれないようにしてください。

リファレンス デザインのコピーは、http://japan.xilinx.com/tools/partial-reconfiguration からダウンロードできます。

パーシャル リコンフィギュレーションの機能を使用するには、パーシャル リコンフィギュレーションの FlexLM ライセンスを取得する必要があります。ライセンスを取得するには、ザイリンクスのフィールド アプリケーション エンジニア (FAE) にご連絡いただくか、ザイリンクス Web サイト http://japan.xilinx.com/getproduct にアクセスしてください。

ハードウェアのテスト用に、オプションで ML605 ボードおよび USB ダウンロード ケーブルを使用できます。

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。このチュートリアルを実行するには、ISE Design Suite の Embedded Edition か System Edition をインストールする必要があります。チュートリアルを始める前に、PlanAhead が起動できるか、リファレンス デザインが解凍されているかどう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_1/irn.pdf

ハードウェア要件

このデザインを問題なく使用するには、2GB 以上の RAM が推奨されます。

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_1/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_1/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_1/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。

パーシャル リコンフィギュレーションの詳細は、『パーシャル リコンフィギュレーション ユーザー ガイド』(UG702) を含めたパーシャル リコンフィギュレーション関連のマニュアルを参照してください。
http://japan.xilinx.com/tools/partial-reconfiguration

チュートリアルの目標

- XPS を使用してプロセッサ システムを生成
- PlanAhead のパーシャル リコンフィギュレーション デザイン フローの機能を使用してフルおよびパーシャル ビットストリームを生成して、XPS HWICAP ペリフェラルを使用して FPGA デザインをダイナミックにリコンフィギュレーション

チュートリアルの手順

このチュートリアルでは、XPS HWICAP ペリフェラルを使用してダイナミックにリコンフィギュレーション可能なデザインをインプリメントする方法について説明します。

次の図は、プロセッサ システムを示しています。デザインには、加算と乗算の 2 つの演算関数を実行するペリフェラル ケーブルが含まれます。

この機能は、ユーザー アプリケーションからハイパーターミナルで確認できます。ダイナミック モジュールは、XPS HWICAP ペリフェラルを使用してリコンフィギュレーションされます。

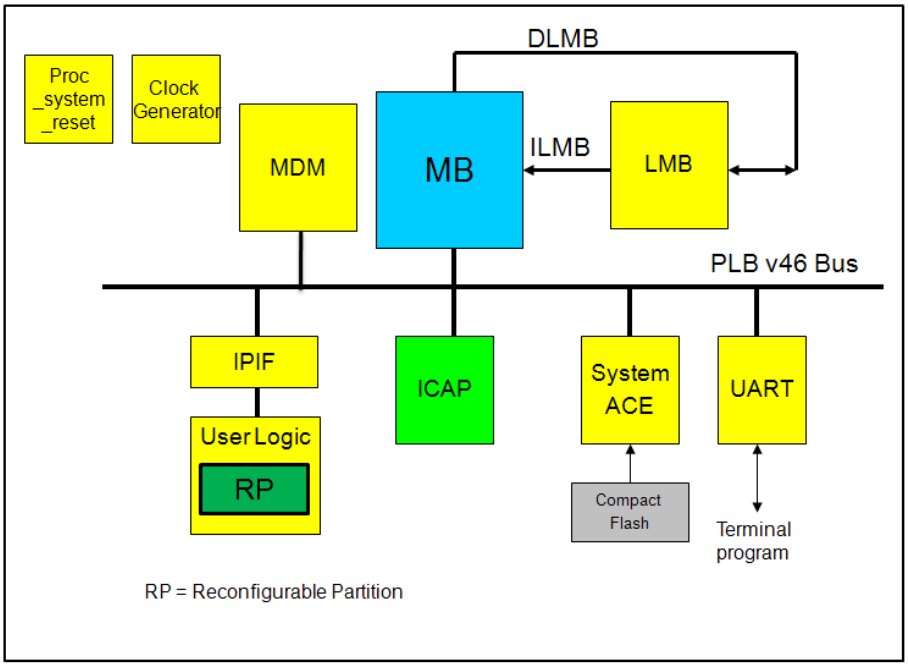

図 1 : 最上位デザイン

ディレクトリ構造

ディレクトリ構造は次のとおりです。

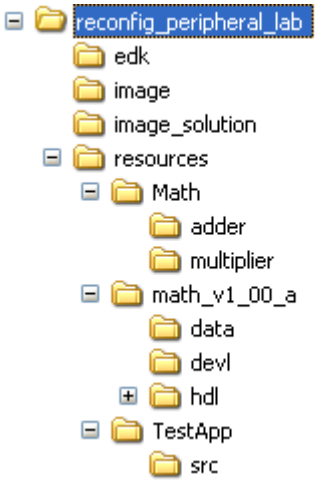

図 2 : プロジェクト ディレクトリ

- \edk ディレクトリはプロセッサ システムの作成に使用されます。
- \resources ディレクトリには、次が含まれます。

- Math には加算および乗算用のコンパイル済みネットリストとそれに関連する下位ディレクトリが含まれます。
- 関数を実行するソフトウェア アプリケーション
- 次を実行する演算プロセッサ コア (pcore) が含まれます。
    - 必要なプロセッサ バス接続を提供
    - 必要なペリフェラル サービスを提供 (この場合、スレーブ レジスタ 1 つとソフトウェア リセット)
    - 演算モジュールのプレースホルダ
- \image ディレクトリには System ACE™ 形式のフルのコンフィギュレーション ビットストリーム ファイルとパーシャル ビットストリーム フィルが保持されます。
- \image_solution ディレクトリには、素早くテストできるように最終的な system.ace とパーシャル ビット ファイルが含まれます。

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。

手順番号	手順名
1	プロセッサ ハードウェア システムの作成
2	ソフトウェア プロジェクトの作成
3	PlanAhead プロジェクトの作成
4	リコンフィギャブル パーティションの定義
5	リコンフィギャブル モジュールの追加
6	リコンフィギャブル パーティション領域の定義
7	デザイン ルール チェックの実行
8	最初のコンフィギュレーションの作成とインプリメント
9	その他のコンフィギュレーションの作成とインプリメント
10	パーシャル リコンフィギュレーション ユーティリティの実行
11	BIT ファイルの生成
12	イメージの作成とテスト

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。

プロセッサ ハードウェア システムの作成　　手順 1

1-1. XPS の Base System Builder (BSB) ウィザードを使用してプロセッサ システムを作成します。

1-1-1. [スタート] → [プログラム] → [Xilinx Design Suite 12.1] → [EDK] → [Xilinx Platform Studio] をクリックして、XPS を起動します。

1-1-2. ダイアログ ボックスが開いたら、Base System Builder ウィザードを使用して新規プロジェクトを作成するオプションをオンにし、[OK] をクリックします。

1-1-3. reconfig_peripheral_lab¥edk ディレクトリを指定します。

1-1-4. [保存] をクリックします。

1-1-5. [OK] をクリックします。

1-1-6. [I would like to create a new design] をオンにし、[Next] をクリックします。

1-1-7. Virtex®-6 ML605 評価プラットフォームのシステムを作成します。

1-1-8. [Board Vendor] で [Xilinx] を選択します。

1-1-9. [Board Name] で [Virtex 6 ML605 Evaluation Platform] を選択します。

1-1-10. [Board Revision] で [D] を選択します。

1-1-11. [Next] をクリックします。

1-1-12. [Single-Processor System] をオンにします。

1-1-13. [Next] をクリックします。

1-1-14. [System Clock Frequency] を 100.00 MHz に設定します。

1-1-15. [Local Memory] ドロップダウン メニューから [64 KB] を選択します。

1-1-16. [Next] をクリックします。

1-1-17. 選択したペリフェラルが右側に表示されるので、次を除くすべてのデバイスを [Remove] ボタンで削除します。

- RS232_Uart_1
- SysACE_CompactFlash
- dlmb_cntlr
- ilmb_cntlr

1-1-18. [RS232_Uart_1] をクリックし、[Baud Rate] を 115200 に変更します。

1-1-19. [Next] をクリックします。

1-1-20. 続く 2 つの画面でそれぞれ [Next] をクリックします。

1-1-21. サマリを確認したら、[Finish] をクリックします。

1-1-22. [The Next Step] ダイアログ ボックスで [OK] をクリックし、Platform Studio を起動すると、次の図のような [System Assembly View] タブが表示されます。

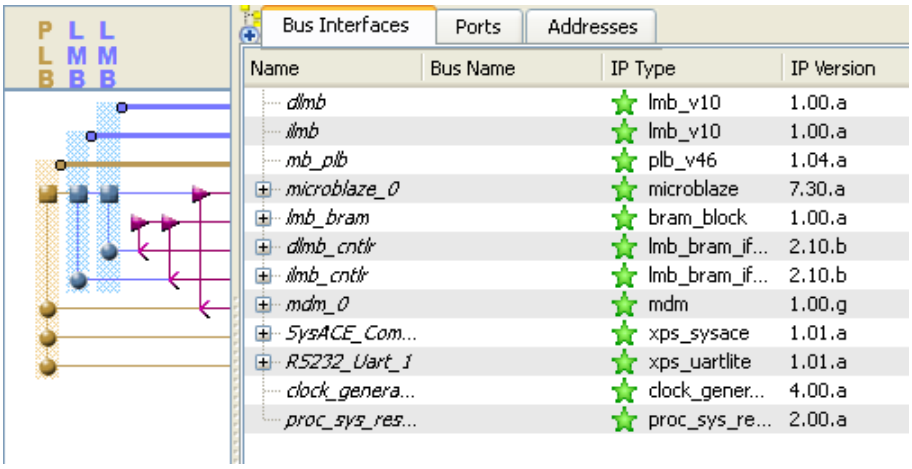

図 3 : [System Assembly View] タブ

1-2. プロセッサ システムへ必要な IP を追加します。

1-2-1. reconfig_peripheral_lab¥resources¥math_v1_00_a フォルダを reconfig_peripheral_lab¥edk¥pcores フォルダにコピーします。

パーシャル リコンフィギュレーションのデザインの詳細

reconfig_peripheral_lab\resources\math_v1_00_a\hdl\vhdl\user_logic.vhd ファイルを XPS で開きます。このファイルの 131 行目では、リコンフィギャブル パーティションで使用されるコンポーネントが宣言されています。 156 行目も同じです。 このコンポーネントへのデータ入力には 189 行目から開始する行でクロックが供給されています。 このコンポーネントへのリセット入力は、ハードウェア バス リセットとソフトウェア リセットの組み合わさったものです。ソフトウェア リセットは 同じディレクトリの math.vhd ファイルの 395 行目の soft_reset ブロックで生成されています。ソフトウェア リセットはパーティションのリコンフィギュレーション後にリコンフィギュレーションされたロジックをリセットするのに必要です。

メモ : 行番号が XPS で表示されていない場合は、次の手順で行番号を表示させてください。

1. **[Edit]** → **[Preferences]** → **[ISE Text Editor]** をクリックします。
2. **[Show line numbers]** をオンにします。
3. **[Apply]** をクリックしてから **[OK]** をクリックします。

1-2-2. [Project] → [Rescan User Repositories] をクリックし、XPS でユーザー レポジトリをスキャンし直します。

[IP Catalog] タブで、[Project Local pcores] フォルダの下の [USER] フォルダの下に MATH が表示されます。

1-2-3. [USER] フォルダを展開表示します。

1-2-4. [MATH] をクリックします。

1-2-5. [MATH] を [System Assembly View] タブにドラッグしてドロップします。

1-2-6. [IP Catalog] タブで、[FPGA Reconfiguration] フォルダの下の [FPGA Internal Configuration Access Port] (v4.00.a) という IP を右クリックし、[Add IP] をクリックします。これでこの IP のインスタンスが [System Assembly View] に追加されます。

1-3. バスを接続し、アドレスを生成します。

1-3-1. [System Assembly View] タブで math_0 および xps_hwicap_0 インスタンスを展開表示します。

1-3-2. 図 4 のように、さまざまなバスを接続します。

ペリフェラルをバスに接続するには、バス ウィンドウで対応する空洞の丸または四角のアイコンをクリックします。丸または四角のアイコンが塗りつぶされていると、ペリフェラルが接続されていることを示しています。

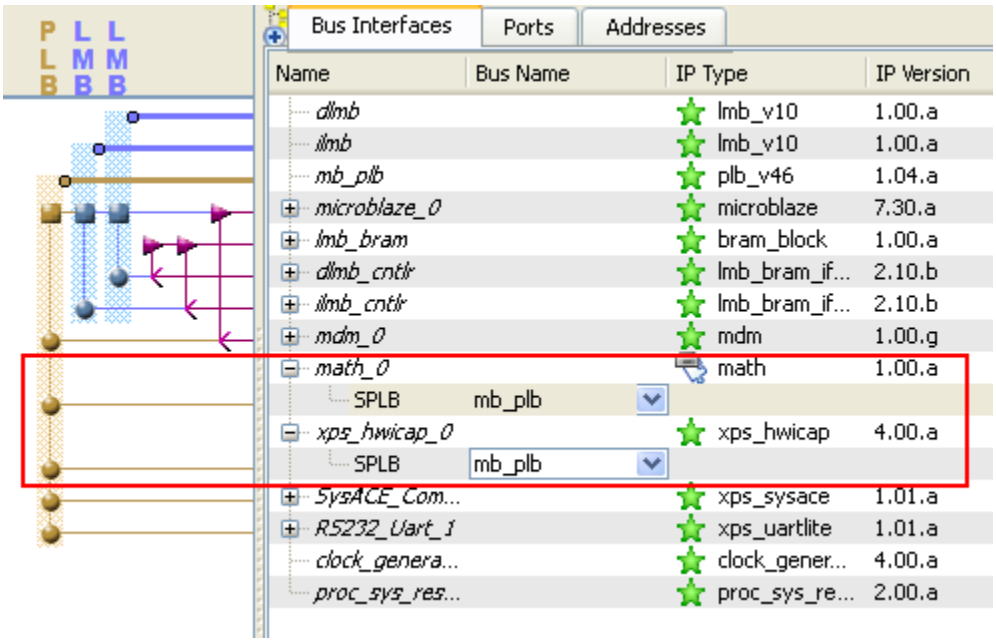

図 4 : [System Assembly View] タブでのバス接続

1-3-3. [Addresses] タブで右上に表示される [Generate Addresses] ボタンをクリックして math_0 および xps_hwicap_0 インスタンスへアドレスを割り当てます。

アドレス マップは次の図のように表示されるはずです。

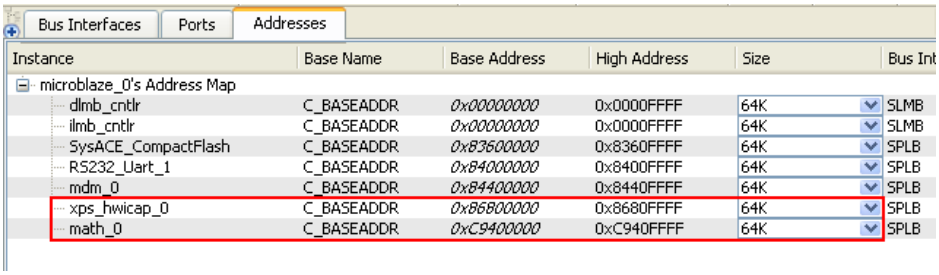

図 5 : システム ペリフェラルのアドレス マップ

1-4. ポートを接続します。

1-4-1. [System Assembly View] タブで [Ports] タブをクリックします。

1-4-2. xps_hwicap_0 インスタンスを展開表示します。

1-4-3. [ICAP_Clk] のドロップダウン ボタンをクリックします。

1-4-4. [clk_100_0000MHz] を選択します。

接続は図 6 のように表示されます。

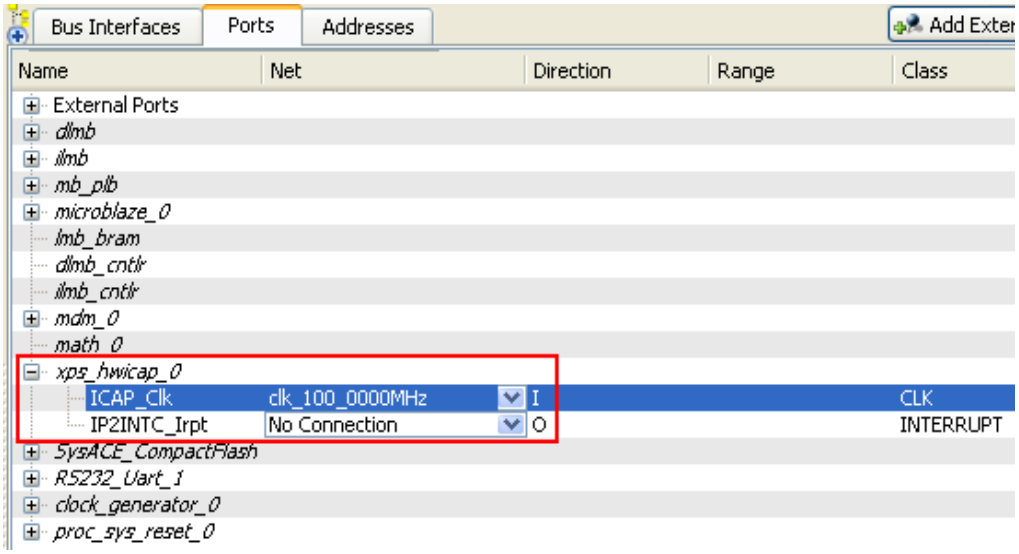

図 6 : クロック ソースの ICAP への接続

パーシャル リコンフィギュレーションのデザインの詳細

xps_hwicap pcore では hwicap 用に別のクロック ドメインを使用できるので、システムが違う速度で実行される場合は 100MHz で実行できます。このチュートリアルでは、システム クロックも 100MHz なので、デザイン全体を 1 つのクロックドメインで実行します。

1-5. ネットリストを生成します。

1-5-1. [Hardware] → [Generate Netlist] をクリックし、Platform Generator を実行します。

これにより、ペリフェラル、システム ネットリスト、および system.bmm ファイルが生成されます。 これらはすべて PlanAhead ソフトウェアのインプリメンテーションで使用されます。

ソフトウェア プロジェクトの作成　　手順 2

ハードウェア ネットリストが生成されたら、EDK に含まれるソフトウェア開発キット (SDK) を使用して、次を実行します。

- ソフトウェア プロジェクトの作成
- 提供されたソース ファイルのインポート
- 提供されたソース ファイルのコンパイル
- 実行ファイルの生成

2-1. ハードウェア デザインを SDK にエクスポートし、ボード サポート パッケージを作成します。

xilfatfs ライブラリ サポートを追加します。

2-1-1. XPS で、[Project] → [Export Hardware Design to SDK] をクリックし、SDK を起動します。

2-1-2. [Include bitstream and BMM file] をオフにします。

2-1-3. [Export & Launch SDK] をクリックします。

2-1-4. SDK で [File] → [New] → [Xilinx Board Support Package] をクリックします。

2-1-5. デフォルトのプロジェクト名は Standalone_bsp_0、OS は Standalone です。

2-1-6. デフォルト設定のまま [Finish] をクリックします。

[Board Support Package Settings] ダイアログ ボックスが開きます。

2-1-7. コンパクト フラッシュ カードでサポートされる FAT ファイル システムである [xilfatfs] をオンにします。

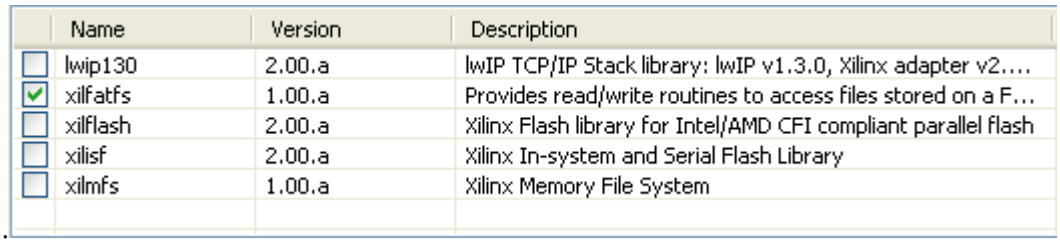

図 7 : ファイル システム サポートの選択

2-1-8. デフォルト設定のまま [OK] をクリックし、このフォームを閉じます。

2-2. Make C アプリケーション プロジェクトを作成します。

2-2-1. [Project Explorer] ビューで standalone_bsp_0 を選択します。

2-2-2. 右クリックし、[New] → [Project] をクリックします。

2-2-3. [Xilinx C Project] を選択します。

2-2-4. [Next] をクリックします。

2-2-5. [Project Name] に TestApp と入力します。

2-2-6. [Project Application Template] フィールドで [Empty Application] を選択します。

2-2-7. [Next] をクリックします。

2-2-8. [Target an existing Board Support Package] をオンにします。

2-2-9. [Finish] をクリックします。

2-3. テスト アプリケーションを生成します。

2-3-1. [Project Explorer] ビューで TestApp を選択します。

2-3-2. 右クリックし、[Import] を選択します。

2-3-3. [General] をダブルクリックします。

2-3-4. [File System] をダブルクリックします。

2-3-5. reconfig_peripheral_lab¥resources¥TestApp¥src フォルダを指定します。

2-3-6. [OK] をクリックします。

2-3-7. [main.c] と [xhwicap_parse.h] をオンにします。

2-3-8. [Finish] をクリックします。

これでソース ファイルがコンパイルされ、reconfig_peripheral_lab\edk\SDK\SDK_Workspace_35\TestApp\Debug フォルダに TestApp.elf ファイルが生成されます。

パーシャル リコンフィギュレーションのデザインの詳細

reconfig_peripheral_lab¥resources¥TestApp¥src¥main.c ファイルを確認します。

このコードの 164 行目からは、コンパクト フラッシュからパーシャル ビットを読み込んで ICAP に書き込む関数が記述されています。

この関数の呼び出しは 433 行目から開始され、特定のパーシャル ビット ファイルを読み込んでソフトウェア リセットをアサートするようプログラムに命令しています。

空白のビットストリームが読み込まれる場合は、リコンフィギャブル領域に実在するロジックがないので、ソフトウェア リセットは必要ありません。

2-4. リンカ スクリプトを生成します。

ヒープおよびスタックのサイズを 2048 (0x800) に設定します。

2-4-1. SDK で [Xilinx Tools] → [Generate linker script] をクリックします。

2-4-2. 参照ボタンで次のフォルダを指定します。

reconfig_peripheral_lab\edk\SDK\SDK_Workspace_35\TestApp\src

2-4-3. lscript.ld を選択します。

2-4-4. [保存] をクリックします。

2-4-5. [Heap Size] に 0x800 と入力し、[Stack Size] のボックスをクリックします。スタック サイズは次の図のように 2048 に変更されます。

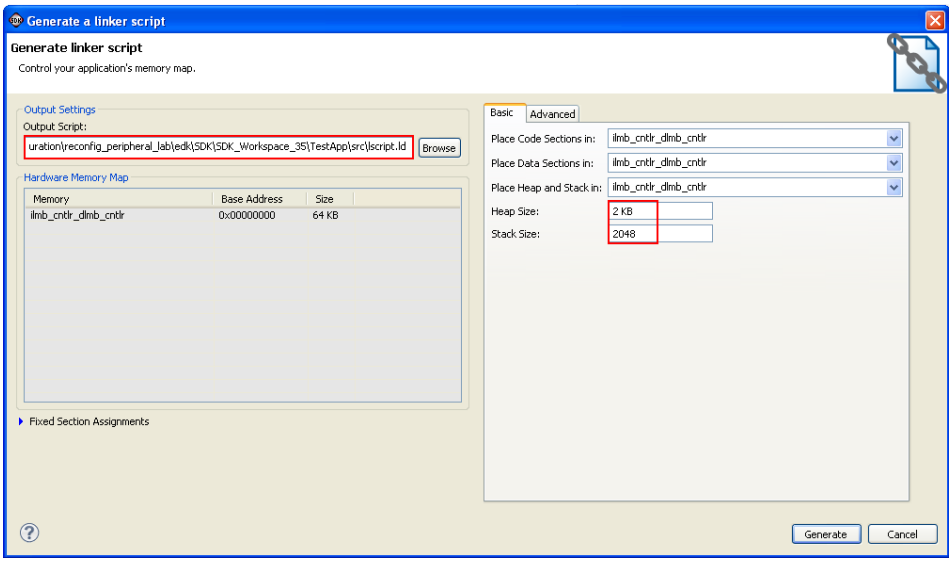

図 8 : リンカ スクリプトの生成

2-4-6. [Generate] をクリックします。

2-4-7. 既存ファイルを上書きしてアプリケーションをリコンパイルするかどうか尋ねるメッセージが表示されるので、[Yes] をクリックします。

2-4-8. [File] → [Exit] をクリックし、SDK を閉じます。

PlanAhead プロジェクトの作成　手順 3

デザインに必要なネットリストは生成したので、次は PlanAhead を使用して次を実行します。

- デザインのフロアプラン
- リコンフィギャブル パーティションの定義
- リコンフィギャブル モジュールの追加
- インプリメンテーション ツールの実行
- フルおよびパーシャル ビットストリームの生成

次の手順では、プロジェクトを新規作成します。

3-1. PlanAhead プロジェクトを作成し、生成したネットリスト ファイルをインポートします。

3-1-1. PlanAhead を起動するには、PlanAhead のデスクトップ 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 12.1] → [PlanAhead] → [PlanAhead] をクリックします。

3-1-2. [Create a New Project] をクリックします。

3-1-3. [Next] をクリックします。

3-1-4. プロジェクト ディレクトリに reconfig_peripheral_lab を参照して指定します。

3-1-5. [Select] をクリックします。

3-1-6. [Project name] に PlanAhead と入力します。

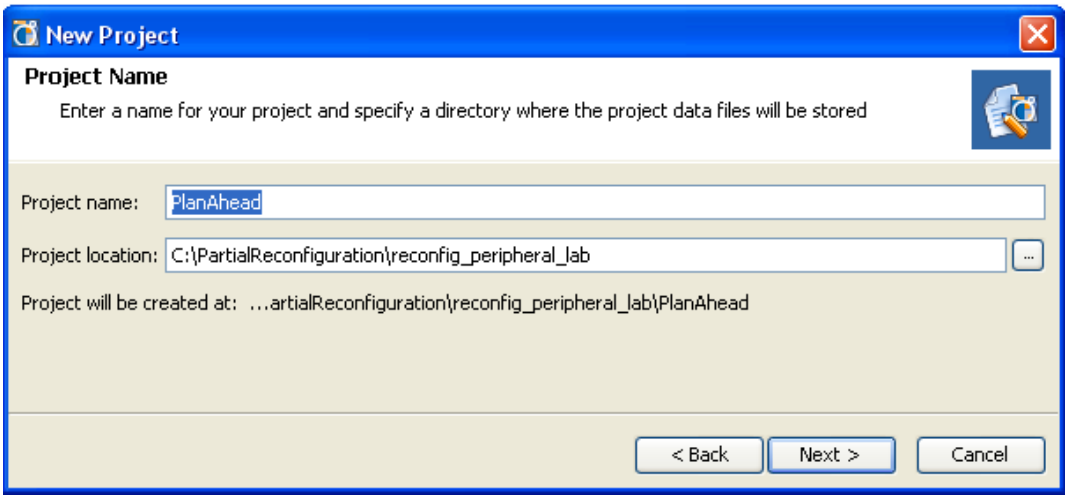

図 9 : [New Project] ダイアログ ボックスの [Project Name] ページ

3-1-7. [Next] をクリックします。

3-1-8. [Design Source] ページで [Specify synthesized (EDIF or NGC) netlist] をオンにします。

3-1-9. [Set PR Project] をオンにします。

3-1-10. [Next] をクリックします。

メモ : [Set PR Project] オプションは、パーシャル リコンフィギュレーションのライセンスがある場合にのみ使用可能になります。

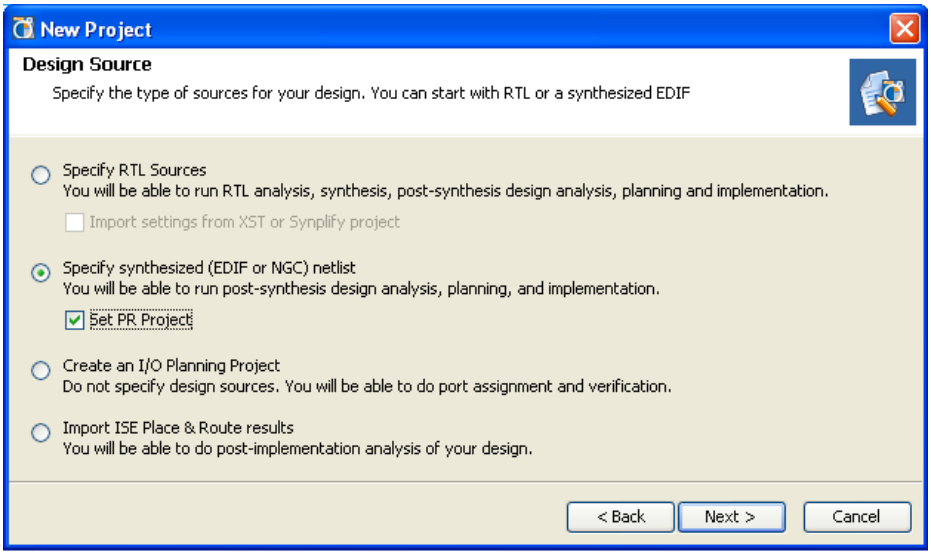

図 10 : 合成済みネットリストのインポート

3-1-11. reconfig_peripheral_lab¥edk¥implementation フォルダを指定します。

3-1-12. system.ngc ファイルを選択します。

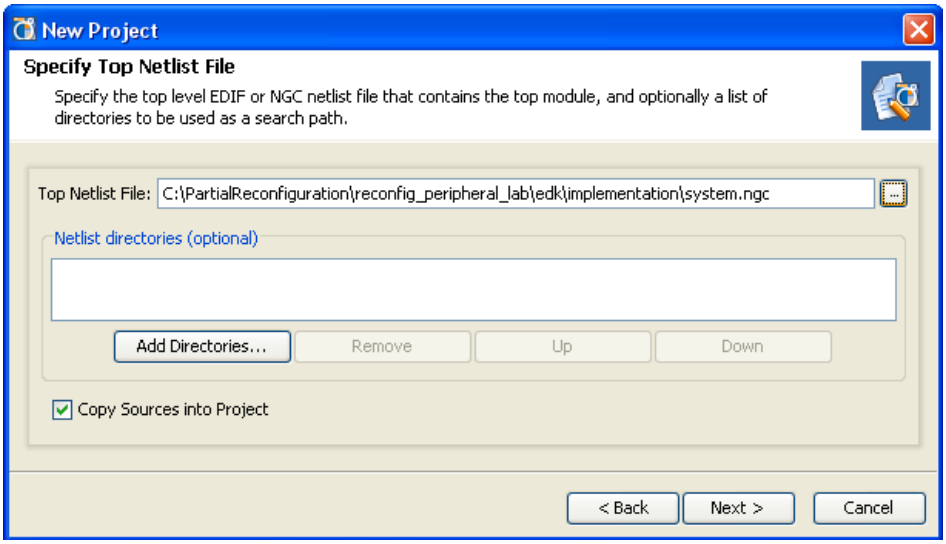

図 11：最上位ネットリスト ファイルの指定

3-1-13. [Next] をクリックします。

[Constraints Files] ページ (オプション) が表示されます。

3-1-14. [Add Files] をクリックします。

3-1-15. ¥reconfig_peripheral_lab¥edk¥data フォルダを指定します。

3-1-16. system.ucf ファイルを選択します。

3-1-17. [保存] をクリックします。

3-1-18. [Next] をクリックします。[Product Family and Default Part] ページが表示されます。

3-1-19. xc6vlx240tff1156-1 が選択されていることを確認します。選択されていない場合は、次の図のようにフィルタを使用して xc6vlx240tff1156-1 パーツを選択します。

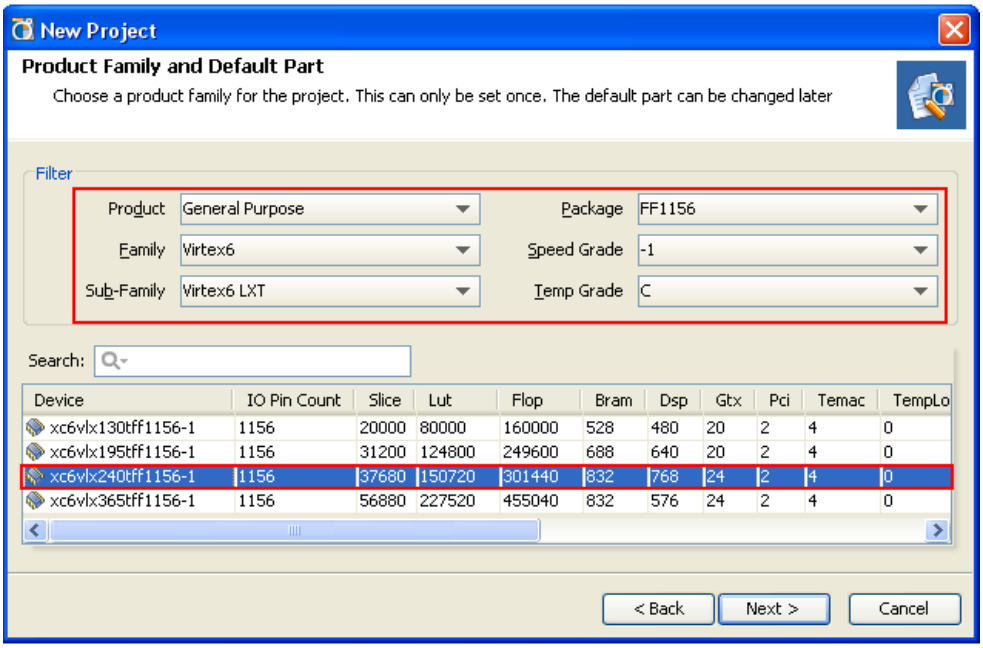

図 12 : ターゲット デバイスの選択

3-1-20. [Next] をクリックします。

3-1-21. [Finish] をクリックします。

プロジェクトが作成されます。Project Manager にデザインに含まれるモジュールが表示されるようになります。

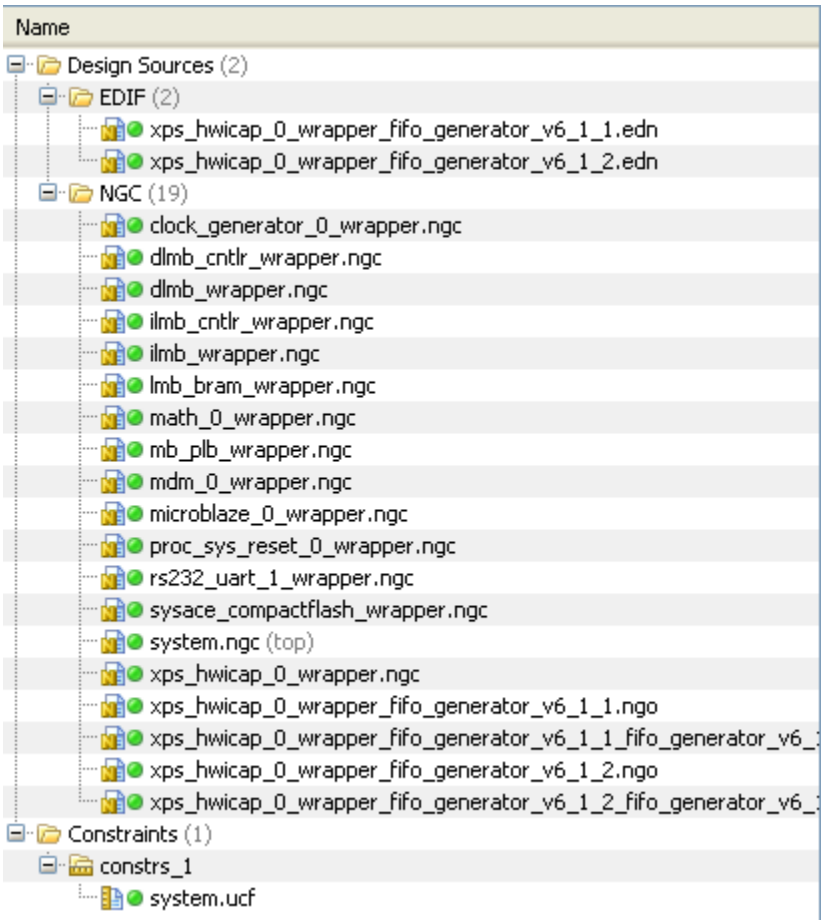

図 13 : PlanAhead のデザイン階層

リコンフィギャブル パーティションの定義　手順 4

デザインには、明確に定義する必要のあるリコンフィギャブル パーティションが 1 つ含まれています。

4-1. ブラック ボックス リコンフィギャブル モジュールを使用して、リコンフィギャブル パーティションを定義します。

4-1-1. [Netlist Design] をクリックし、ネットリスト ファイルを解析します。

この手順は、リコンフィギャブル パーティションを定義するのに、下位レベル モジュールへアクセスする必要があるので必ず実行します。

1 つのインスタンスがネットリスト ファイルがないため、ブラック ボックスに変換されることを示す警告メッセージが表示されます。このモジュールにはまだネットリストを関連付けていないため、これは予測通りのメッセージです。

図 14 のように、[Netlist] タブにはシステムの階層が表示されます。

4-1-2. [OK] をクリックします。

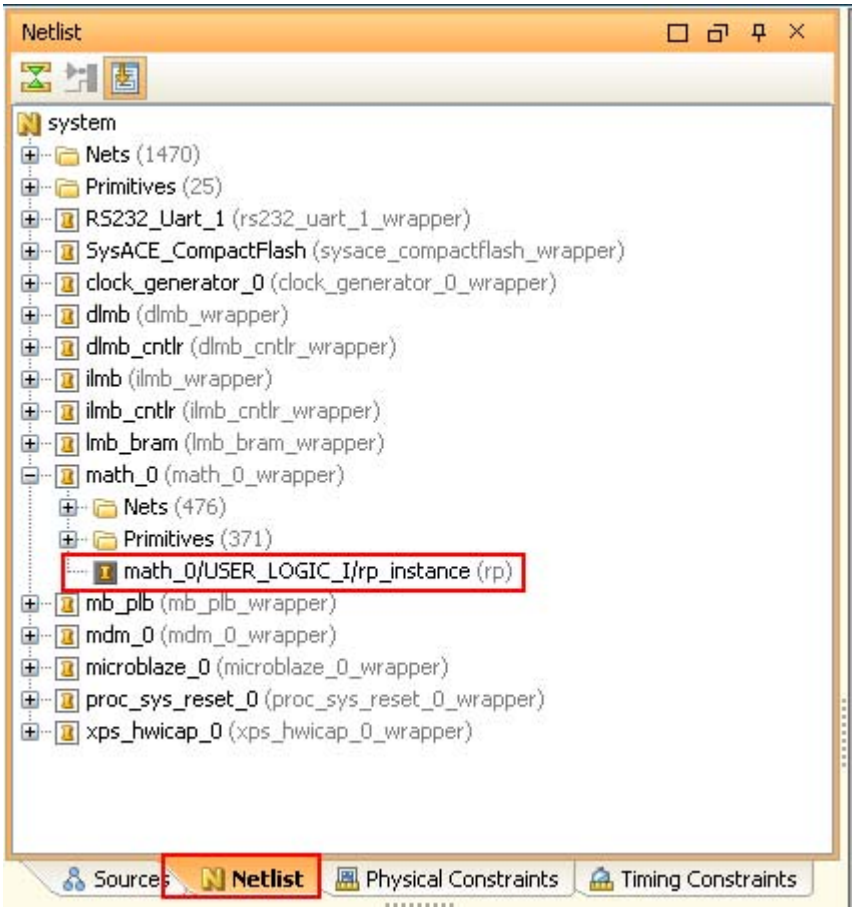

図 14 : ネットリストの階層表示

4-1-3. [Netlist] タブで math_0/USER_LOGIC_I/rp_instance を右クリックします。

4-1-4. [Set Partition] をクリックします。

4-1-5. [Next] を 2 回クリックします。

4-1-6. 図 15 のように、[Set Partition] ダイアログ ボックスが表示されます。

4-1-7. [Add this Reconfigurable module as a black box without a netlist] を選択します。

4-1-8. パーティションにはまだ定義されたネットリストが含まれていないので、リコンフィギャブル モジュール名に math_BB と入力します。

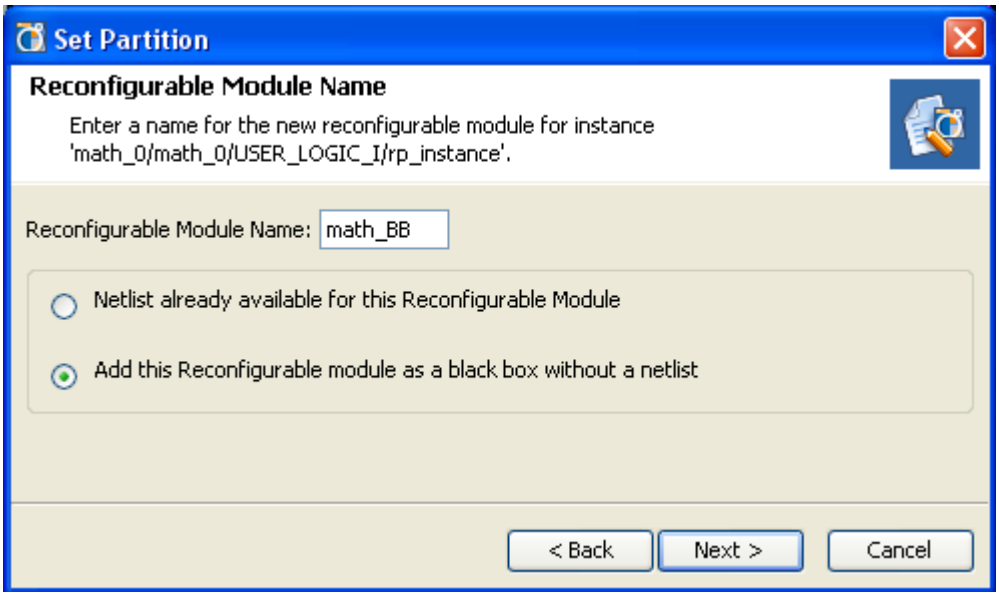

図 15 : パーティションの設定

4-1-9. [Next] をクリックします。

4-1-10. [Finish] をクリックします。

メモ : ブラック ボックス アイコンがひし形のアイコンに変わります。

リコンフィギャブル モジュールの追加　　手順 5

このデザインには、リコンフィギャブル パーティションに対して 2 つのリコンフィギャブル モジュールが含まれます。この手順では、この 2 つのモジュールを追加します。

5-1. adder と multiplier の 2 つのリコンフィギャブル モジュールを追加します。

5-1-1. [Netlist] タブで **math_0/USER_LOGIC_I/rp_instance** を右クリックします。

5-1-2. [**Add Reconfigurable Module**] をクリックします。

5-1-3. [**Next**] をクリックします。

図 16 のように、[Add Reconfigurable Module] ダイアログ ボックスが表示されます。

5-1-4. [Reconfigurable Module Name] に **adder** と入力します。

5-1-5. [**Netlist already available for this Reconfigurable Module**] がオンになっていることを確認します。

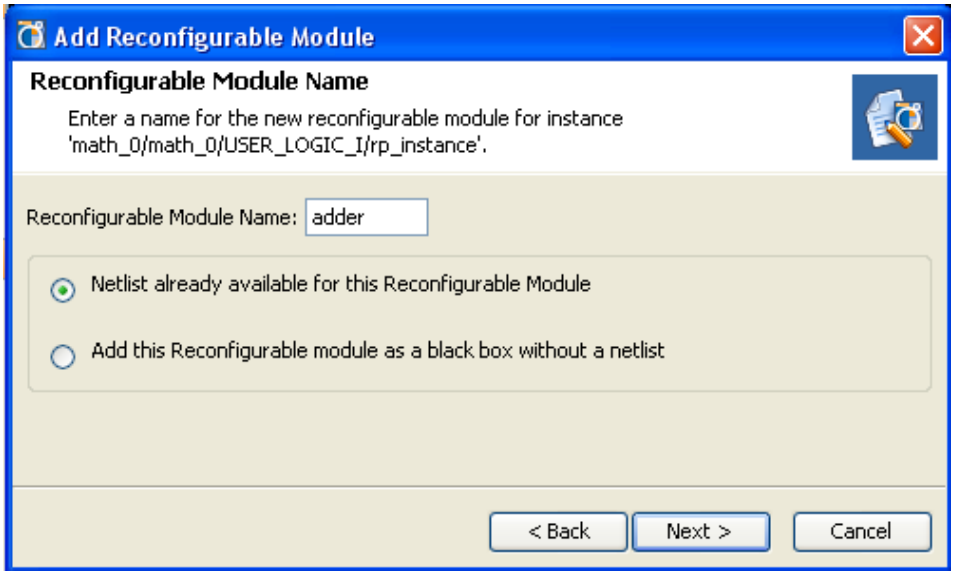

図 16 : リコンフィギャブル モジュールの追加

5-1-6. [**Next**] をクリックします。

5-1-7. reconfig_peripheral_lab¥resources¥Math¥adder ディレクトリを指定し、**rp.ngc** ファイルを選択します。

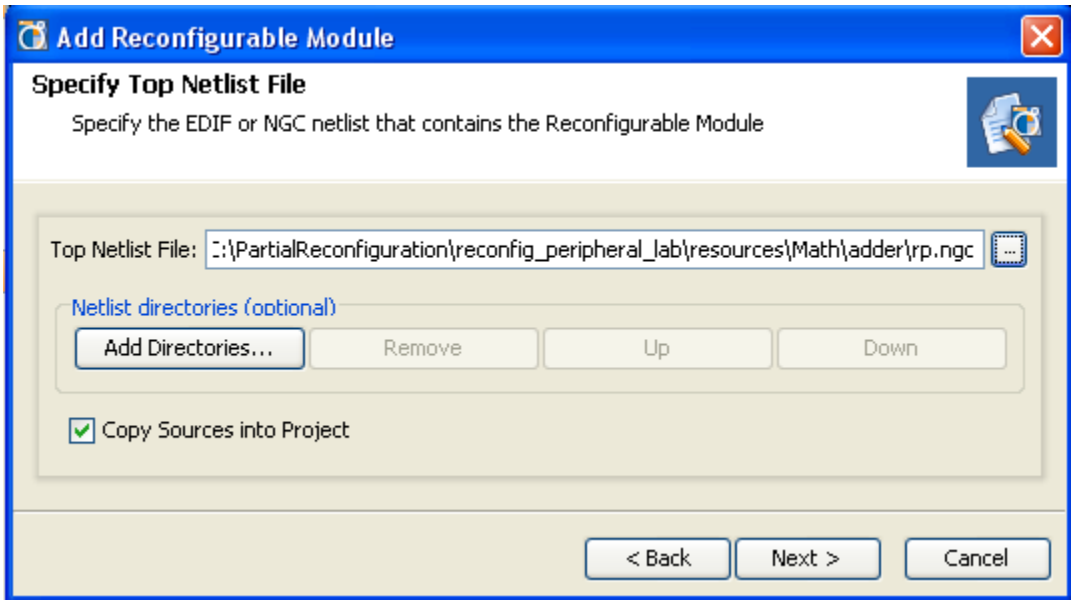

図 17 : 最上位のネットリスト ファイルの指定

5-1-8. [Next] を 2 回クリックします。

5-1-9. [Finish] をクリックします。

5-1-10. [Netlist] タブで math_0/USER_LOGIC_I/rp_instance の下のリコンフィギャブル モジュールを展開表示し、加算器 (adder) を確認します。

5-1-11. 手順 5 に従い、reconfig_peripheral_lab¥resource¥Math¥multiplier¥rp.ngc ディレクトリから乗算器のリコンフィギャブル モジュールを追加します。リコンフィギャブル モジュールの名前には mult と指定します。

[Netlist] タブの math リコンフィギャブル パーティションには、ブラック ボックスを含め、3 つのリコンフィギャブル モジュールが表示されます。

プロジェクトに最後に追加したネットリストである乗算器モジュールがアクティブ (ひし形にチェック マークの付いたアイコン) になっています。

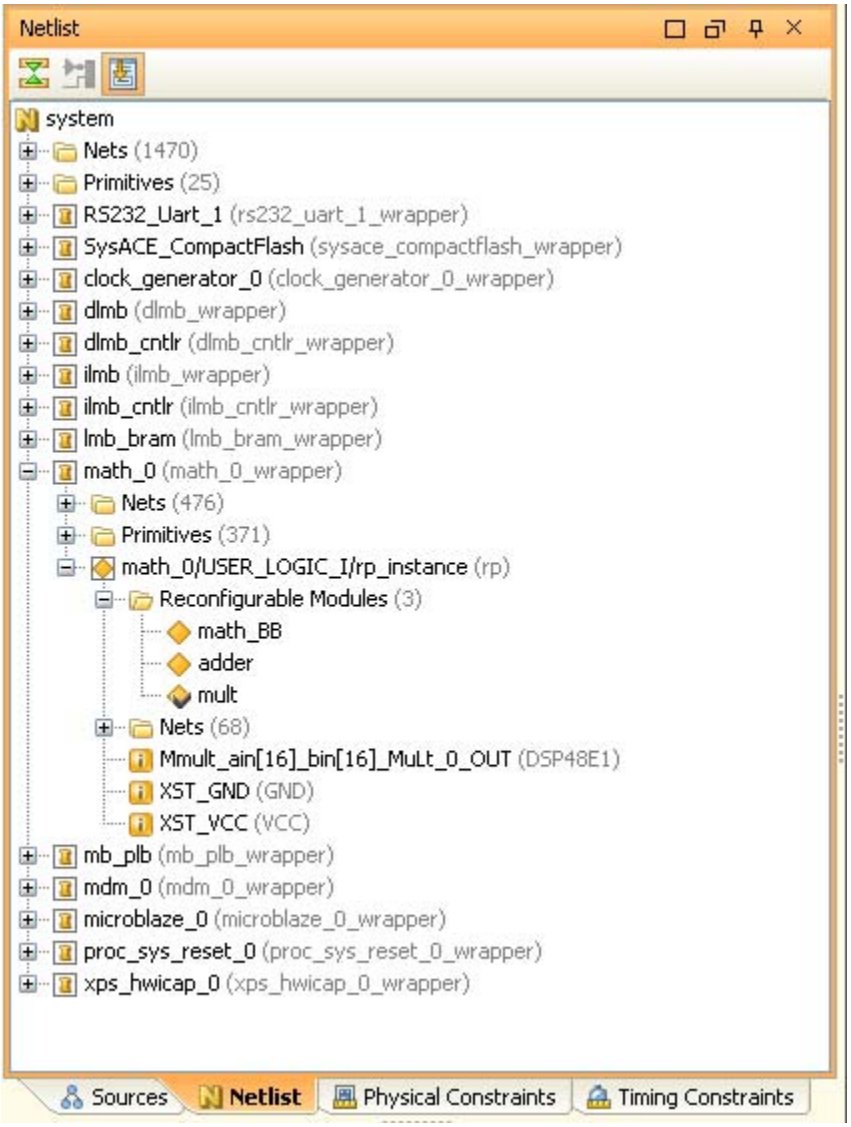

図 18 : adder および mult リコンフィギャブル モジュールを追加したプロジェクト

リコンフィギャブル パーティション領域の定義　　手順 6

次は、リコンフィギャブル パーティション領域をフロアプランする必要があります。この領域は、各リコンフィギャブル モジュールで使用されるリソースの種類や量によって定義する必要があります。

6-1. リコンフィギャブル パーティション領域を設定します。

6-1-1. [Physical Constraints] タブで pblock_math_0/USER_LOGIC_I/rp_instance を選択します。

6-1-2. 右クリックし、[Set Pblock Size] をクリックします。

図 19 :[Physical Constraints] タブ

6-1-3. FPGA の左上の区画を拡大します。

6-1-4. [Device] ビューでカーソルを移動します。

6-1-5. カーソルをクリックしてドラッグし、次の図のように SLICE_X10Y230:SLICE_X15Y239 を囲むようなボックスを描きます。

乗算器のリコンフィギャブル モジュールには DSP48E が 1 つ、加算器のリコンフィギャブル モジュールには 32 ビットの長細いキャリー チェーンが必要なので、この領域は必ずボックスで囲む必要があります。

カーソルの座標は、PlanAhead メイン ウィンドウ一番下のステータス バーに表示されます。

領域をボックスで囲むと、[Set Pblock] ダイアログ ボックスが表示されます。

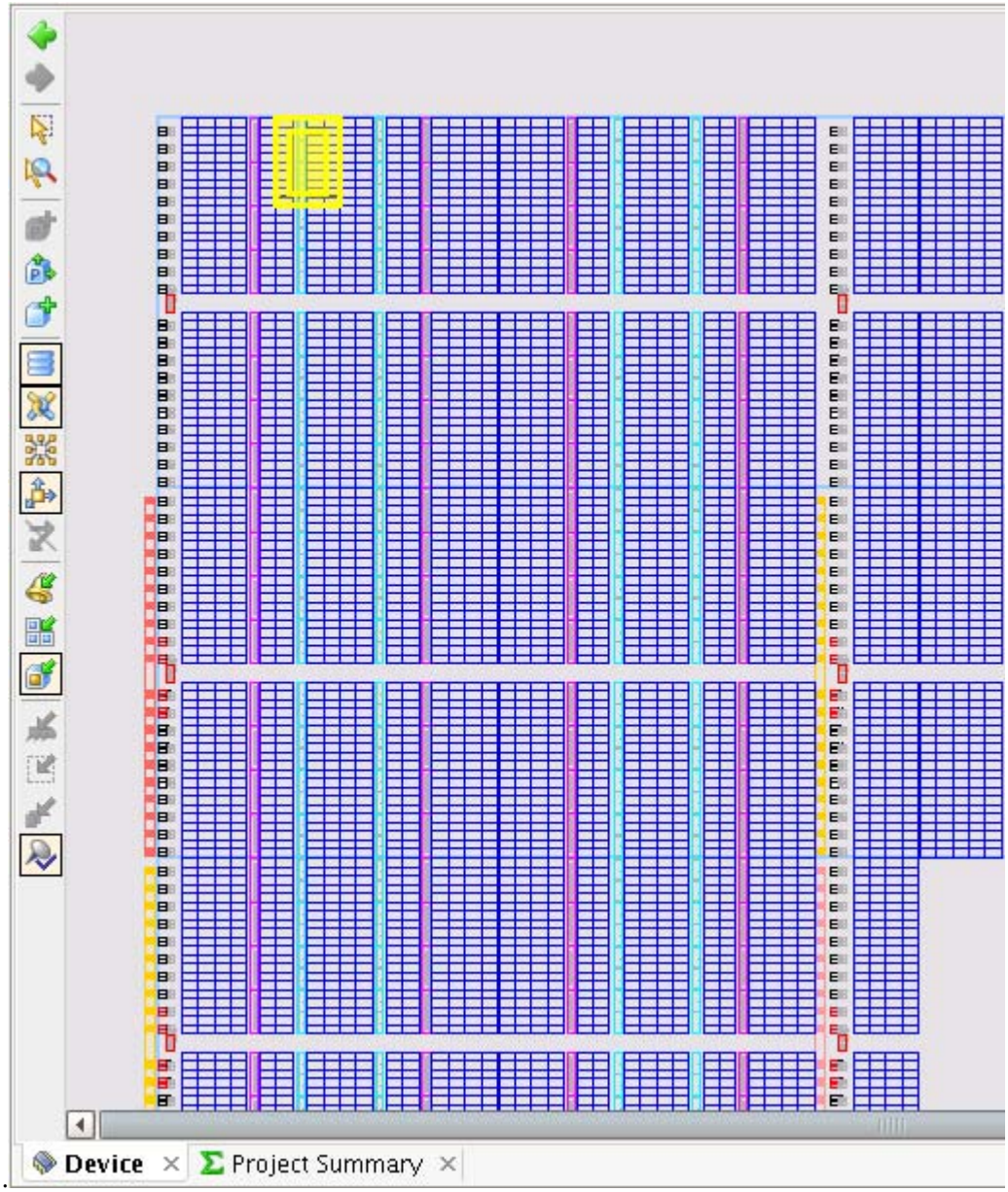

図 20 : Pblock エリアの拡大表示

6-1-6. 次の図のように、[Set Pblock] ダイアログ ボックスでリコンフィギュレーションされるリソース、[SLICE] と [DSP48] をオンにします。

6-1-7. [OK] をクリックします。

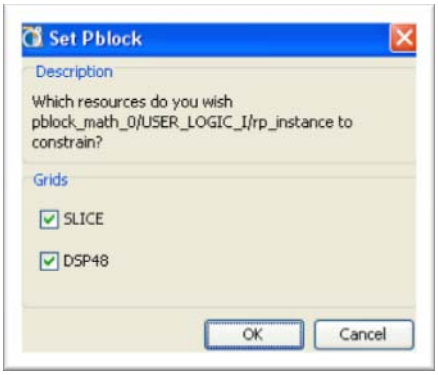

図 21 : [SLICE] と [DSP48] をオンにした [Set Pblock] ダイアログ ボックス

デザイン ルール チェックの実行　　手順 7

ザイリンクスでは、デザイン ルール チェック (DRC) を実行して、できるだけ早い段階でエラーを検出することをお勧めしています。

7-1. パーシャル リコンフィギュレーション用の DRC を選択して実行します。

7-1-1. [Tools] → [Run DRC] をクリックします。

7-1-2. [All Rules] をオフにします。

7-1-3. [Partial Reconfig] をオンにします。

7-1-4. [OK] をクリックすると、パーシャル リコンフィギュレーションのみのデザイン ルール チェックが実行されます。

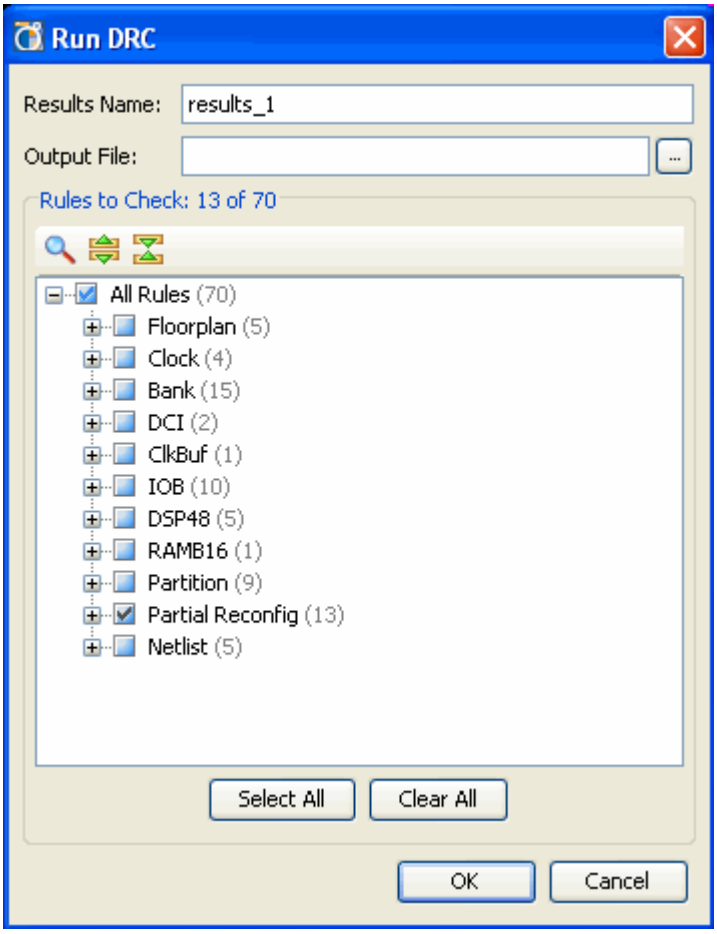

図 22 : デザイン ルール チェックの実行

リコンフィギャブル モジュールがインプリメントされていないことを示す警告メッセージが表示されます。

最初のコンフィギュレーションの作成とインプリメントとプロモート　　手順 8

これで最初のコンフィギュレーションを作成し、インプリメントできるようになりました。

8-1. 新しくストラテジを作成します。

8-1-1. 新しいストラテジには、-bm オプションで system.bmm ファイルを指定します。

8-1-2. [Tools] → [Options] をクリックします。

8-1-3. 左側で [Strategies] をクリックします。

8-1-4. [Flow] ドロップダウン ボックスから [ISE 12] を選択します。

8-1-5. [PlanAhead Strategies] の下の [ISE Defaults] を選択します。

8-1-6. + マークをクリックし、新しいストラテジを作成します。

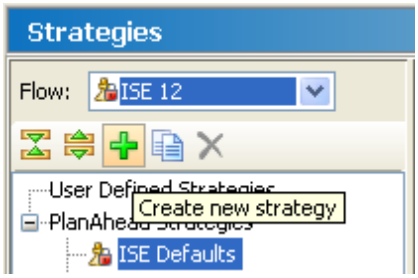

図 23 : 新規ストラテジの作成

8-1-7. 新しいストラテジに ISE12_BM という名前を付けます。

8-1-8. [OK] をクリックします。

8-1-9. [Translate (ngdbuild)] の下で -bm オプションのドロップダウン ボタンをクリックします。

8-1-10. reconfig_peripheral_lab¥edk¥implementation¥system.bmm ファイルを参照して指定します。

8-1-11. [開く] をクリックします。

8-1-12. [OK] をクリックします。

8-2. mult を使用してインプリメンテーションを実行します。

8-2-1. [Create Multiple Runs] をクリックします。

8-2-2. PlanAhead の一番下の [Design Runs] タブをクリックします。

8-2-3. config_1 run を選択します。

8-2-4. [Implementation Run Properties] ビューで [General] タブをクリックします。

8-2-5. [Name] に run 名として **mult** を入力します。

8-2-6. **[Apply]** をクリックして、run 名を config_1 から **mult** に変更します。

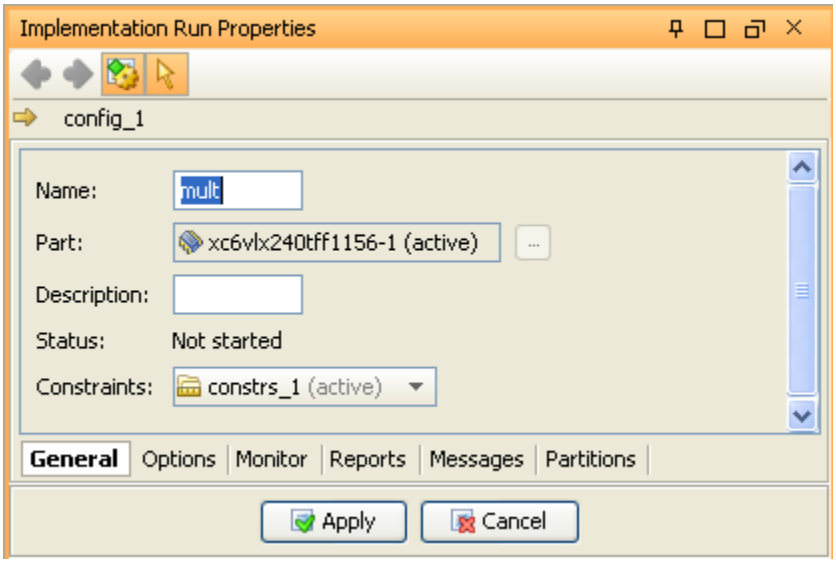

図 24 : [Implementation Run Properties] ビュー

8-2-7. [Options] タブで、ストラテジを [ISE12_BM] に変更します。

8-2-8. **[Apply]** をクリックします。

8-2-9. 図 25 のように、[Partitions] タブの [Module Variant] 列のドロップダウン ボタンをクリックし、**mult** を選択します。

8-2-10. **[Apply]** をクリックします。

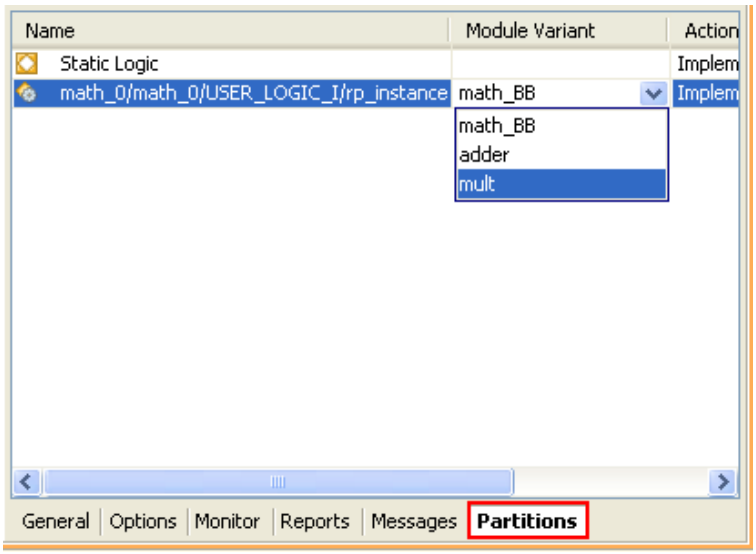

図 25 : [Module Variant] の選択

8-2-11. [Design Run] ビューで **mult** を右クリックし、**[Launch Selected Runs]** をクリックしてインプリメンテーションを実行します。

8-2-12. **[Launch Runs on Local Host]** を選択します。

8-2-13. **[OK]** をクリックします。

8-2-14. **[Save]** をクリックしてプロジェクトを保存してから、インプリメンテーションを実行します。

インプリメンテーションが実行されます。

インプリメンテーションが終了したら、インプリメントされた結果を読み込んだり、インプリメントされたパーティションをプロモートしたり、ほかのオプションを実行できるダイアログ ボックスが表示されます。

8-2-15. [Promote Partitions] をオンにし、[OK] をクリックします。

8-2-16. [Promote Partitions] ダイアログ ボックスで **[OK]** をクリックして現在のコンフィギュレーションをプロモートします。これにより、インプリメンテーション結果がほかのコンフィギュレーションで使用できるようになります。

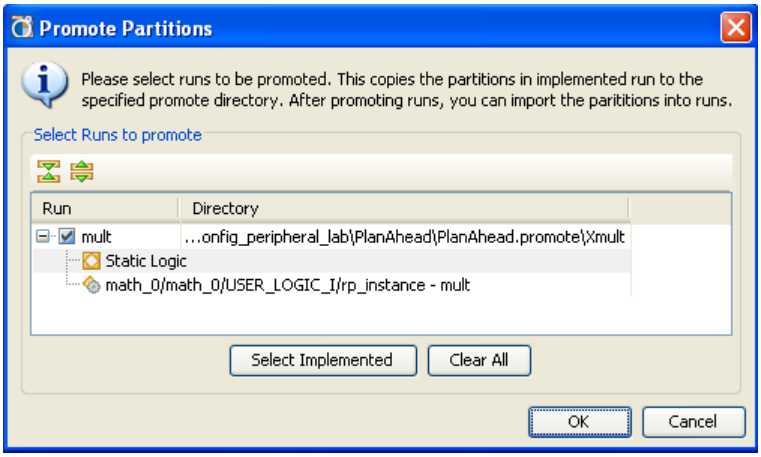

図 26 : パーティションのプロモート

その他のコンフィギュレーションの作成とインプリメント　　手順 9

最初のコンフィギュレーションを作成したら、そのスタティック ロジックのインプリメンテーションを残りのコンフィギュレーションに再利用します。この後、必要な数だけのコンフィギュレーションを追加して、それらをインプリメントします。

9-1. 複数の run を作成します。

9-1-1. [Tools] → [Create Multiple Runs] をクリックします。

Create Multiple Runs ウィザードが開きます。

9-1-2. [Next] を 2 回クリックします。

9-1-3. [Choose Reconfigurable Modules and Implementation Strategies] ページでコンフィギュレーション名を config_1 から **adder** に変更します。

9-1-4. [More] をクリックします。

9-1-5. 名前を config_1 から **black_box** に変更します。

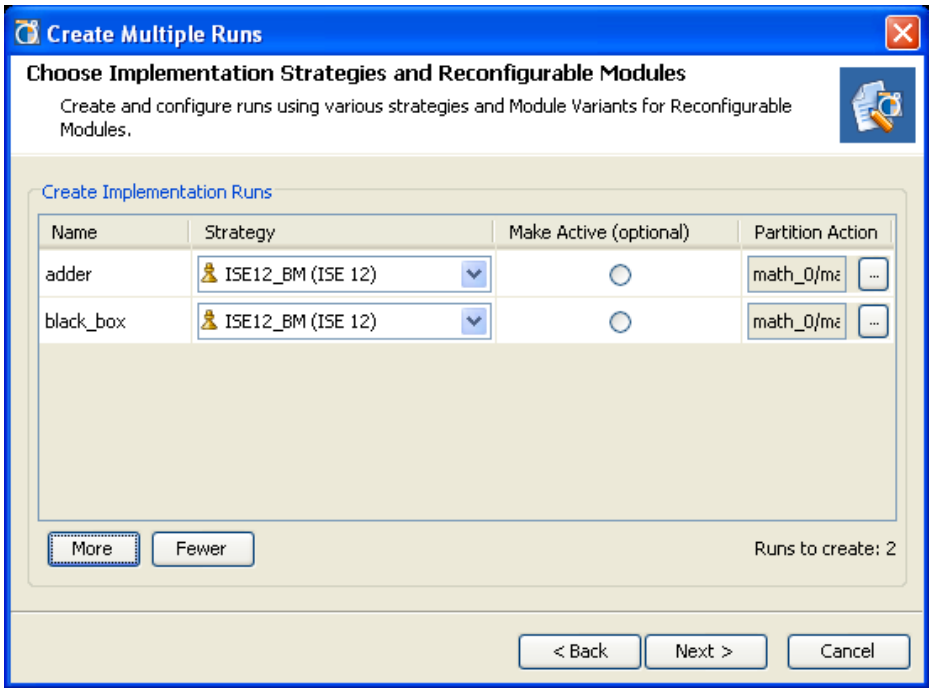

図 27 : Create Multiple Runs ウィザード

9-1-6. adder の [Partition Action] 列の参照ボタンをクリックします。

9-1-7. rp_instance の [Module Variant] 列のドロップダウン ボタンをクリックし、**[adder]** を選択します (図 28)。

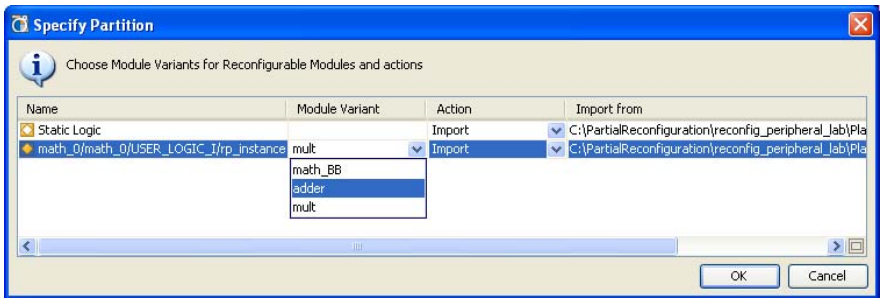

図 28 : adder の [Module Variant] の選択

9-1-8. **[OK]** をクリックします。

9-1-9. 同じように、black_box の [Module Variant] に **math_BB** を選択します。

9-1-10. **[Next]** をクリックします。

9-1-11. [Launch Runs on Local Host] をオンにして [Number of Jobs] を 2 に設定します。

9-1-12. [Next] をクリックします。

9-1-13. [Design Run] ビューでこの 2 つの run をクリックし、[Launch Selected Runs] ボタンをクリックします。

9-1-14. [Finish] をクリックし、両方のコンフィギュレーションのインプリメンテーションを実行します。

9-2.

パーシャル リコンフィギュレーションの検証ユーティリティの実行　　手順 10

リコンフィギャブル領域へのインターフェイスを含め、スタティック インプリメンテーションがすべてのコンフィギュレーション間で矛盾しないようにする必要があります。これには、パーシャル リコンフィギュレーション検証ユーティリティを実行します。

10-1. パーシャル リコンフィギュレーション検証ユーティリティを実行します。

パーシャル リコンフィギュレーション検証ユーティリティは、エラーがないようにするために実行します。

10-1-1. [Configurations] ビューでいずれかのコンフィギュレーションを選択します。

10-1-2. 右クリックし、[Verify Configuration] をクリックします。

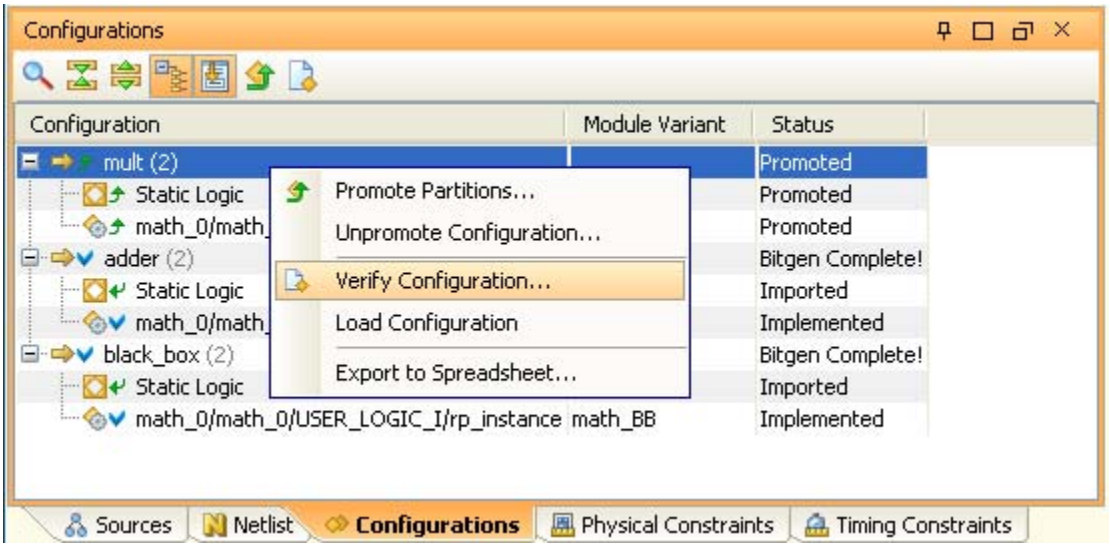

図 29 : コンフィギュレーションの検証

10-1-3. Shift キーを押して、すべてのコンフィギュレーションを選択します。

10-1-4. [OK] をクリックします。

10-1-5. パーシャル リコンフィギュレーション検証ユーティリティが実行され、エラーがなかったことがレポートされます。

BIT ファイルの生成　　手順 11

コンフィギュレーションをパーシャル リコンフィギュレーション検証ユーティリティでチェックしたら、プロジェクト全体のフルおよびパーシャル BIT ファイルを生成できます。

11-1. フルおよびパーシャル ビットストリームを生成します。

11-1-1. [Design Runs] ビューで Shift キーを押しながら次の 3 つの run を選択します。

- **mult**
- **adder**
- **black_box**

11-1-2. 右クリックで [Generate Bitstream] をクリックします。

ビットストリーム生成プロセスが実行され、フルおよびパーシャル ビットストリームが生成されます。

BIT ファイルは、reconfig_peripheral_lab\PlanAhead\PlanAhead.runs\ ディレクトリの mult、adder、および black box フォルダにそれぞれ保存されます。

11-1-3. プロジェクトを保存します。

11-1-4. PlanAhead を閉じます。

イメージの作成とテスト　　手順 12

この手順では、EDK シェルを開いて、download.bit および system.ace ファイルを ¥image ディレクトリに作成します。生成されたパーシャル BIT ファイルを ディレクトリにコピーし、それぞれ adder.bit、mult.bit、blank.bit という名前を付けます。

12-1. パーシャル ビットストリーム ファイルと生成された system.ace ファイルの名前を変更します。

12-1-1. EDK Bash シェルを次のいずれかの方法で起動します。

12-1-2. XPS で [Project] → [Launch Xilinx Bash Shell] をクリックします。

12-1-3. Windows 環境からは [スタート] → [プログラム] → [Xilinx ISE Design Suite 12.1] → [EDK] → [Tools] → [Xilinx] → [Bash Shell] をクリックします。

12-1-4. Bash シェルで ¥reconfig_peripheral_lab¥image ディレクトリに移動します。

12-1-5. 次のコマンドを実行して adder.bit (ハードウェア コンポーネント) から download.bit (ソフトウェア コンポーネント) ファイルを生成します。

data2mem -bm ../edk/implementation/system_bd -
bt ../PlanAhead/PlanAhead.runs/adder/adder.bit -
bd ../edk/SDK/SDK_Workspace_35/TestApp/Debug/TestApp.elf tag microblaze_0 -o b download.bit

ヒント：上記のコマンド テキストをコピーし、タイトルバーを右クリックして **[Edit]** → **[Paste]** をクリックして Bash シェルに貼り付けます。

これで \image ディレクトリに download.bit が生成されます。

12-1-6. Bash シェルで次のコマンドを実行し、¥image ディレクトリの system.ace ファイルを生成します。

xmd -tcl genace.tcl -jprog -target mdm -hw download.bit -board ml605 -ace system.ace

12-1-7. Windows エクスプローラを使用し、次の表 1 のファイルをコピーして名前を変更します。

表 1：パーシャル BIT ファイルの名前変更

ファイル名	コピー先ディレクトリ	変更後の名前
reconfig_peripheral_lab\PlanAhead\PlanAhead.runs\adder\ adder_math_0_math_0_user_logic_i_rp_instance_adder_partial.bit	\image	adder.bit
reconfig_peripheral_lab\PlanAhead\PlanAhead.runs\mult\ mult_math_0_math_0_user_logic_i_rp_instance_mult_partial.bit	\image	mult.bit
reconfig_peripheral_lab\PlanAhead\PlanAhead.runs\black_box\ black_box_math_0_math_0_user_logic_i_rp_instance_math_bb_partial.bit	\image	blank.bit

12-2. system.ace とコンパクト フラッシュ メモリ カードの 3 つのパーシャル BIT ファイルをコピーします。

12-2-1. 空白のコンパクト フラッシュ メモリ カードをコンパクト フラッシュ ライターに配置します。

12-2-2. Windows エクスプローラを使用し、3 つのパーシャル BIT ファイルと system.ace ファイルをコンパクト フラッシュ カードの reconfig_peripheral_lab¥image フォルダからコピーします。

12-2-3. ML605 ボードにコンパクト フラッシュ カードを配置します。

12-2-4. SACE モード ピン (S1) を 0111 (dn-up-up-up) に設定し、コンパクト フラッシュから FPGA デバイスをコンフィギュレーションします。

12-2-5. 提供されている USB ケーブルを使用し、ML605 ボードと PC を接続します。

12-2-6. 必要であれば、ドライバをインストールします。詳細は、『ML605 ハードウェア ユーザー ガイド』を参照してください。
http://japan.xilinx.com/support/documentation/boards_and_kits/ug534.pdf.

12-2-7. ハイパーターミナル ウィンドウを開き、115200 ボー レートの COMx を使用して接続し、ML605 ボードの電源をオンにします。

12-2-8. [CPU Reset] を押します。

12-2-9. メニューに従い、さまざまなリコンフィギュレーションをテストします。

まとめ

このチュートリアルでは、XPS を使用してプロセッサ システムを作成し、リコンフィギャブル パーティション用のプレースホルダを含むユーザー ペリフェラルを追加し、ネットリスト ファイルを生成しました。また、SDK を使用してアプリケーションを作成しました。PlanAhead ソフトウェアを使用してフルおよびパーシャル ビットストリームを生成しました。この後、コンパクト フラッシュ メモリ カードの ACE ファイルを生成し、ML605 評価ボードを使用してその機能を検証しました。