(機械で日本語に翻訳)

Getting started with the FPGA demo bundle for Intel FPGA

Xillybus Ltd.

www.xillybus.com

Version 3.1

この文書はコンピューターによって英語から自動的に翻訳されているため、言語が不明瞭になる可能性があります。このドキュメントは、元のドキュメントに比べて少し古くなっている可能性もあります。

可能であれば、英語のドキュメントを参照してください。

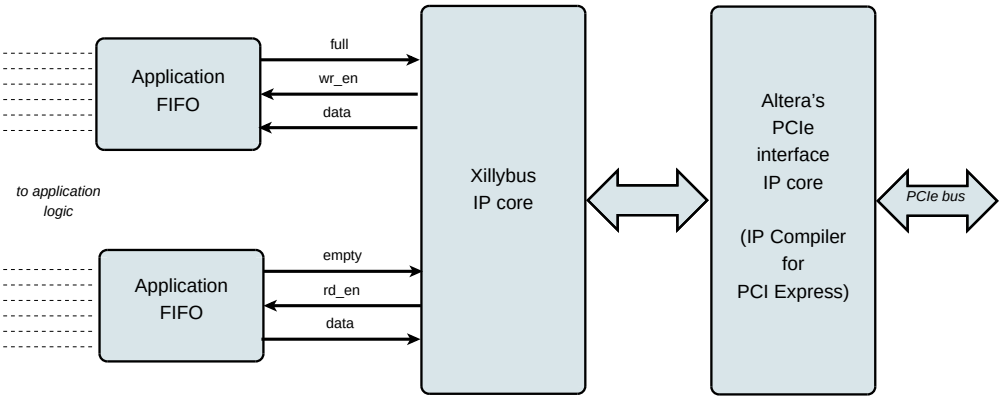
This document has been automatically translated from English by a computer, which may result in unclear language. This document may also be slightly outdated in relation to the original.

If possible, please refer to the document in English.

1

序章

Xillybus は、FPGA と、Linux または Microsoft Windows を実行する host との間のデータ転送用の DMA ベースのエンドツーエンド ソリューションです。 FPGA logic の設計者だけでなく、ソフトウェアのプログラマーにもシンプルで直感的なインターフェイスを提供します。



上記のように、FPGA 上の application logic は、標準の FIFOs と対話するだけで済みます。

たとえば、図の下部のFIFOにデータを書き込むと、Xillybus IP coreは、FIFOのもう一方の端でデータを送信できることを認識します。間もなく、IP coreはFIFOからデータを読み取り、それをhostに送信して、userspace softwareで読み取り可能にします。データ転送メカニズムは、FIFOと相互作用するだけのFPGAのapplication logicに対して透過的です。

一方、Xillybus IP core は、PCI Express の Transport Layer level を利用してデータフローを実装し、TLP パケットを生成および受信します。下位層については、開発ツールの一部である Intel の公式 PCIe core に依存しており、追加のライセンスは必

要ありません (Quartus の Web/Lite Edition を使用する場合でも)。

コンピューター上のアプリケーションは、named pipes のように動作する device files と対話します。 Xillybus IP core と driver は、FPGAs の FIFOs と host の関連する device files の間でデータを効率的かつ直感的に転送します。

XillyUSB では、MGT transceiver を使用して USB 3.0 インターフェイスを実装します。USB 3.0 インターフェイスは、上記の PCIe インターフェイスの代わりにデータ転送に使用されます。

IP core は、オンライン Web アプリケーションを使用して、顧客の仕様に従って即座に構築されます。 streams の数、方向、およびその他の属性は、design の帯域幅パフォーマンス、同期、およびシンプルさの間で最適なバランスを実現するために、お客様が定義します。このガイドで説明されているように、demo bundle で準備手順を実行した後、http://xillybus.com/custom-ip-factory でカスタム IP core をビルドしてダウンロードすることをお勧めします。

このガイドでは、FPGA を Xillybus IP core で迅速にセットアップする方法について説明します。Xillybus IP core は、ユーザー提供のデータ ソースとデータ コンシューマーに接続して、実際のアプリケーション シナリオ テストを行うことができます。 IP core は demo bundle に含まれており、Web サイトからダウンロードできます。

その名前にもかかわらず、demo bundle はデモンストレーション キットではなく、有用なタスクをそのまま実行できる完全に機能する starter design です。

興味のある方は、Xillybus host application programming guide for Linux または Xillybus host application programming guide for Windowsの付録 A に Xillybus の実装方法に関する簡単な説明があります。

2

前提条件

2.1 ハードウェア

Xillybus は、PCI Express を Intel の hardware IP block に依存しているため、このコンポーネントを備えたすべての Intel FPGA デバイスで使用できます。サポートされている FPGA ファミリの中で:

- Arria II GX/GZ
- Cyclone IV GX
- HardCopy IV GX
- Stratix IV GX
- Arria V GX/GT/SX/ST
- Cyclone V GX/GT/SX/ST
- Stratix V GS/GX/GT
- Arria 10 GX/GT/SX
- Cyclone 10 GX
- Stratix 10 と H-tile または L-tile

XillyUSB は Cyclone 10 GX でのみサポートされます (LP ファミリはサポートされていません)。

Xillybus FPGA demo bundle は、ダウンロード ページにリストされているように、いくつかのボードとデバイスで動作するようにパッケージ化されています (以下のセクション 2.2 を参照してください)。

他のボードの所有者は、pin placements で必要な変更を行い、MGT の reference clock が適切に処理されることを確認した後、自分のハードウェアで demo bundle を実行できます。これは、かなり経験のある FPGA エンジニアにとっては簡単なことです。これについてはセクション 4.4 で詳しく説明します。

2.2 FPGA プロジェクト

Xillybus demo bundle は、Xillybus サイトのダウンロード ページからダウンロードできます。 PCIe ベースの cores の場合:

http://xillybus.com/pcie-download

XillyUSB の場合:

http://xillybus.com/usb-download

demo bundle には、単純なテストを目的とした Xillybus IP core の特定の構成が含まれています。そのため、特定のアプリケーションではパフォーマンスが比較的低くなります。

カスタムIP coresは、IP Core Factory Webアプリケーションを使用して構成、自動構築、およびダウンロードできます。このツールの使用については、http://xillybus.com/custom-ip-factoryにアクセスしてください。

Xillybus IP core を含むダウンロードされたバンドルは、この使用が “evaluation” という用語に合理的に一致する限り、無料で使用できます。これには、core をエンドユーザー designs に組み込み、実際のデータとフィールド テストを実行することが含まれます。 core の使用方法に制限はありません。この使用の唯一の目的が、特定のアプリケーションに対するその機能と適合性を評価することである限りです。

2.3 開発ソフトウェア

Xillybus の demo bundle (および Xillybus を含む他の designs) の implementation に推奨されるツールは、FPGA のファミリーに応じて以下にリストされています。

PCIe の Xillybus :

- series-IV および Arria II の FPGAs の場合: Quartus 12.0 以降。
- Arria 10 および Cyclone 10 の場合: Quartus Prime 17.1 以降。 Standard と Pro の両方のエディションがサポートされています。
- Stratix 10 の場合: Quartus Pro 19.2 以降。

- 他のすべての FPGA ファミリ: Quartus II、バージョン 15.0 以降。

XillyUSB :

- Cyclone 10 の場合: Quartus Pro 17.1 以降を使用する必要があります。

このソフトウェアは、Intel の Web サイト (https://www.intel.com) から直接ダウンロードできます。

Quartus の Web/ Lite Edition は無料で入手でき、いくつかの FPGA ファミリ、特に Cyclone デバイスをサポートしていることに注意してください。

Xillybus の implementation は、Quartus によって提供される一部の IP cores に依存しています。 Quartus のすべてのエディションは、これらの IP cores に対応しており、追加のライセンスは必要ありません。

2.4 FPGA designの使用経験

design が demo bundles のリストに表示されるボードを対象としている場合、demo bundle を FPGA で動作させるために FPGA design の以前の経験は必要ありません。他のボードを使用する場合、Intel FPGA のツールの使用、特に pin placements と clocks の定義に関する知識が必要です。

demo bundle を最大限に活用するには、logic design の手法を十分に理解し、HDL 言語 (Verilog または VHDL) を習得する必要があります。それにもかかわらず、Xillybus demo bundle は、実験するための単純な starter design を提供するため、これらを学習するための良い出発点です。

3

demo bundleのimplementation

3.1 概要

Xillybus の demo bundle の implementation、および bit stream ファイル (SOF) の取得には、次の 3 つの方法があります。

- バンドル内のプロジェクト ファイルをそのまま使用します。これは最も簡単な方法で、demo bundles のリストに表示されるボードで作業する場合に適しています。
- 別の FPGA に一致するようにファイルを変更します。これは、他のボードや他の FPGAs で作業する場合に適しています。これについての詳細は、段落 4.4 を参照してください。
- Quartus プロジェクトをゼロからセットアップします。 demo bundle を既存の application logic と統合する場合に必要になる可能性があります。段落 4.3 の詳細。

このセクションの残りの部分では、最初の作業手順について詳しく説明します。これは、最も単純で最も一般的に選択される手順です。他の 2 つの作業手順は、最初の作業手順に基づいていますが、上記の段落で詳しく説明されている違いがあります。

重要:

評価バンドルは、パフォーマンスよりも単純化のために構成されています。特に高帯域幅の場合、持続的かつ継続的なデータ フローを必要とするアプリケーションでは、大幅に優れた結果が得られます。これらのシナリオでは、カスタム *IP core* を簡単に構築し、*Web* アプリケーションでダウンロードできます。

3.2 ファイル概要

バンドルは、次の 5 つのディレクトリで構成されます。

- core– Xillybus IP core はここに格納されます
- instantiation templates – core 用の instantiation templates が含まれています (Verilog および VHDL 内)。
- verilog – demo bundle のプロジェクト ファイルと Verilog のソースが含まれています ('src' サブディレクトリ内)。
- vhdl – demo bundle のプロジェクト ファイルと VHDL のソースが含まれています ('src' サブディレクトリ内)。
- pcie_core (PCIe バンドルのみ) – Intel の PCIe IP core は、バンドルの残りの部分をビルドする前にここでビルドされます。
- quartus-essentials (XillyUSB バンドルのみ) – Verilog および VHDL プロジェクトに共通のさまざまなファイル。

demo bundle がダウンロードされたサイトの Web ページにリストされているように、各 demo bundle は特定のボード用であることに注意してください。別のボードが使用されている場合、または特定の構成抵抗がボードに追加または削除されている場合は、それに応じて constraints ファイルを編集する必要があります。

また、vhdl ディレクトリには Verilog ファイルが含まれていますが、これらのファイルを大幅に変更する必要はないことに注意してください。

Xillybus の IP core と application logic の間のインターフェースは、xillydemo.v ファイルまたは xillydemo.vhd ファイル (それぞれの 'src' サブディレクトリ内) で行われます。これは、Xillybus を自分のデータで試すために編集するファイルです。

3.3 demo bundle の構築

3.3.1 プロジェクトを開く

好みに応じて、'verilog' または 'vhdl' サブディレクトリにある 'xillydemo.qpf' ファイルをダブルクリックします。 Quartus が起動し、正しい設定でプロジェクトが開きます。

series-V (例: Cyclone V) または series-10 (例: Arria 10) FPGA が使用されている場合は、段落 3.3.3 に進みます。

それ以外の場合は、次に説明するように PCIe ブロックをビルドします。

3.3.2 PCIe ブロックのビルド (Arria II および series-IV FPGAs)

重要:

この段落は *series-V FPGAs* 以降には関係ないことに注意してください。

12.0 より後の Quartus バージョンを使用する場合、 megafunction variation ファイルを手動で編集する必要があります。そうしないと、MegaWizard Plug-In Manager はこのファイルの受け入れを拒否します。

編集が必要な場合は、pcie_core/ ディレクトリ (例: pcie_core/ pcie_s4_4x.v) のファイルをテキスト エディタで開きます。古い Quartus バージョンの出現箇所をすべて、使用されているものに置き換えます。バージョン番号を変更するだけです (たとえば、12.0 を 15.0 に置き換えます)。

通常、次のような行は変更が必要です:

```
// megafunction wizard: %IP Compiler for PCI Express v12.0%
```

と

```
// Retrieval info: <MEGACORE title="IP Compiler for PCI Express"  version="12.0"
```

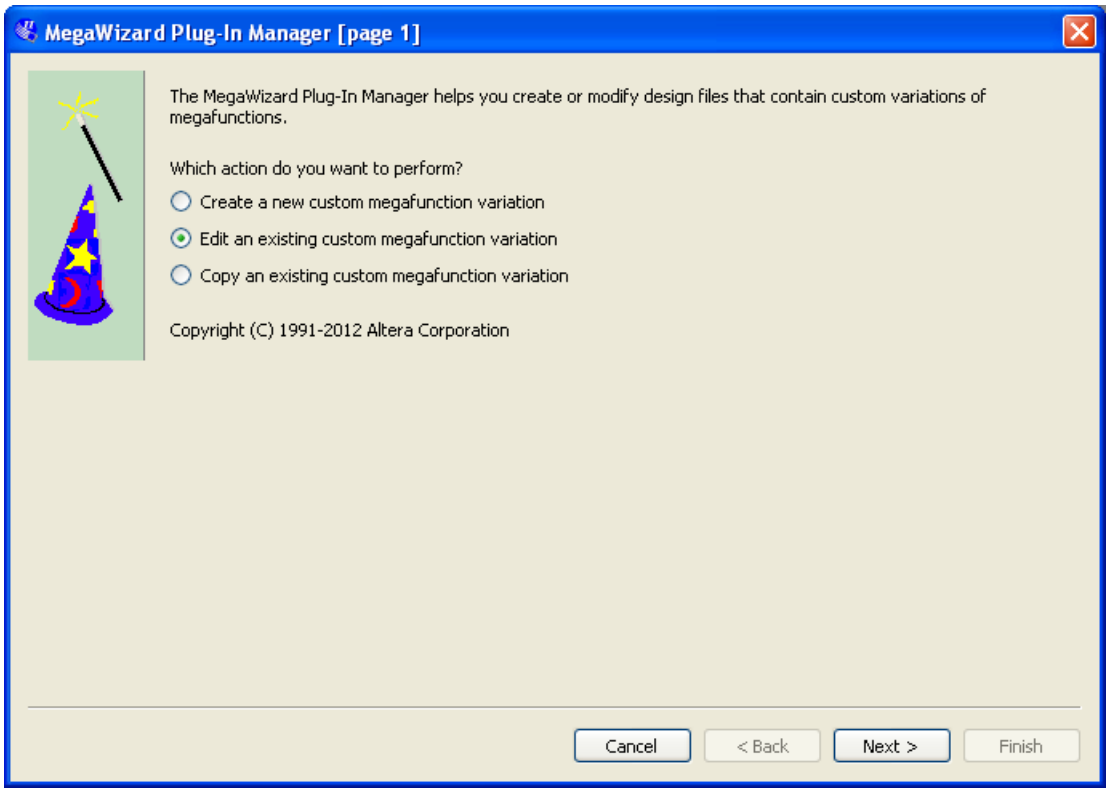
および場合によっては

```
// Generated using ACDS version 12.0 178
```

Quartus 内から MegaWizard Plug-In Manager を起動します。

- Quartus 14 以降: Command Prompt ウィンドウ (または Linux の terminal) を開きます。 Quartus のユーティリティが execution path (通常は /some/path/quartus/bin/) にあることを確認してください。 MegaWizard Plug-In Manager を起動するには、“qmegawiz” と入力します。
- Quartus 13 以前: [ツール] メニューで、[MegaWizard Plug-In Manager] を選択します。

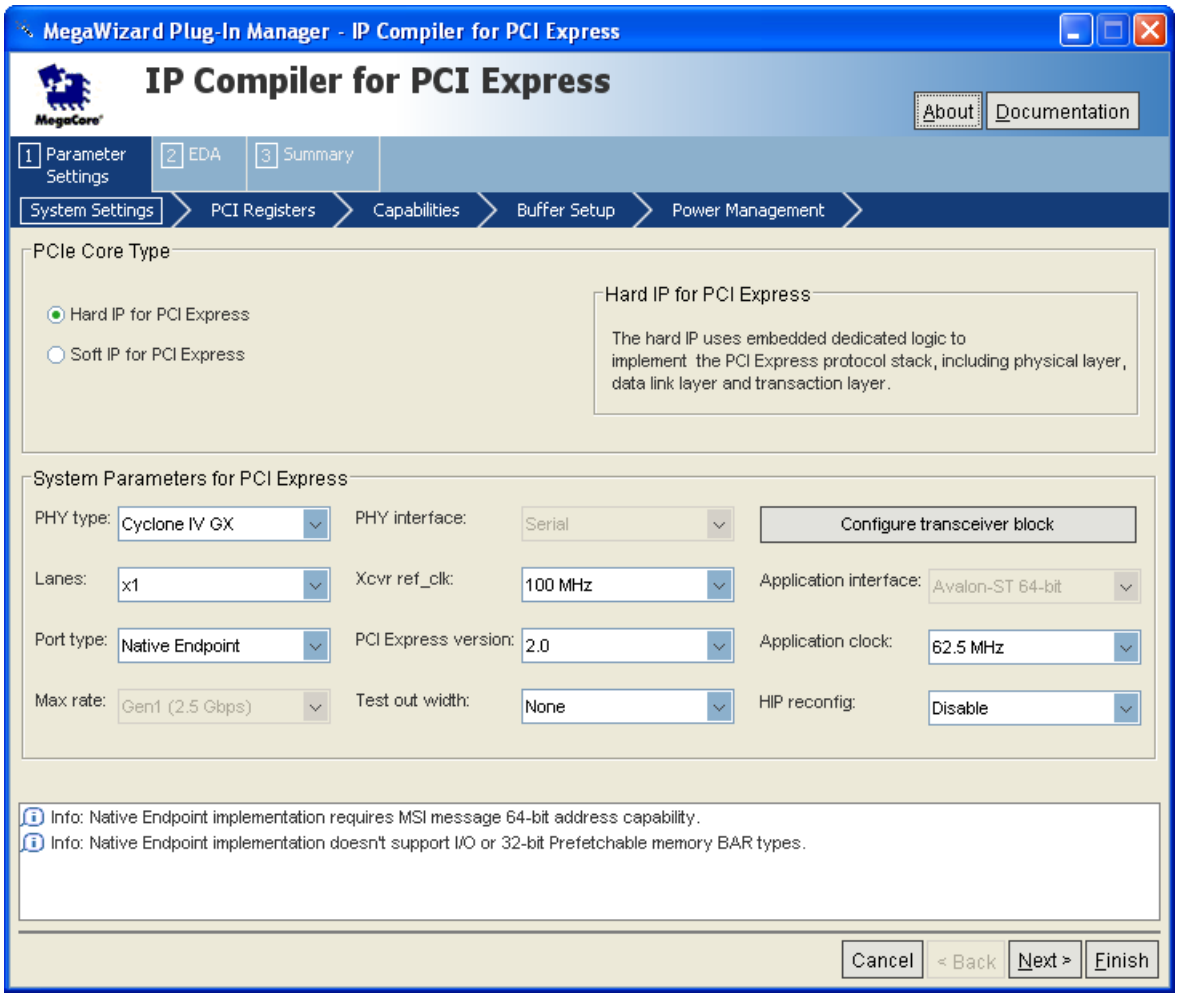
次の (または同様の) ウィンドウが表示されます。



“Edit an existing custom megafunction variation” を選択し、[次へ] をクリックします。

次に、ウィンドウ (ここには表示されていません) が、編集するカスタム megafunction variation ファイルを要求します。 pcie_core ディレクトリ内の pcie_c4_1x.v (または同様の) ファイルを選択します (適切なファイルが 1 つだけ存在します)。通常、開始点から 1 つのディレクトリを上に移動すると、入力するディレクトリが表示されます。

“Loading MegaWizard...” という通知の後、次のようなウィンドウが表示されます。

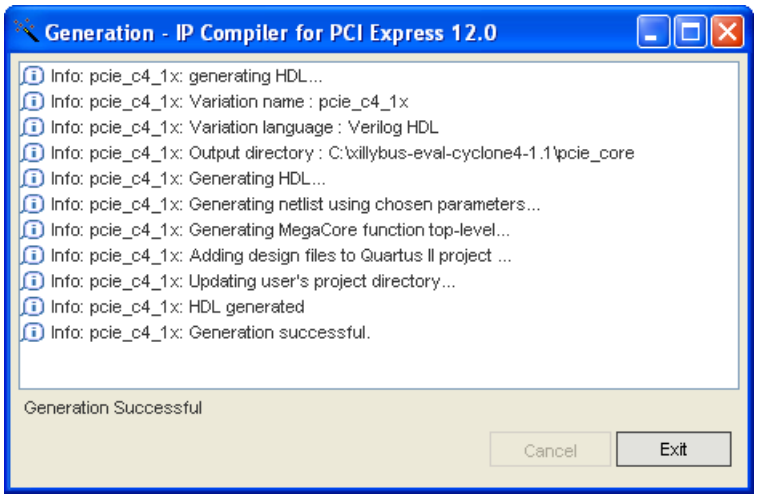

MegaWizard Plug-In Manager - IP Compiler for PCI Express
IP Compiler for PCI Express
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
System Settings
PCI Registers
Capabilities
Buffer Setup
Power Management
PCIe Core Type
Hard IP for PCI Express
Soft IP for PCI Express
Hard IP for PCI Express
The hard IP uses embedded dedicated logic to implement the PCI Express protocol stack, including physical layer, data link layer and transaction layer.
System Parameters for PCI Express
PHY type: Cyclone IV GX
PHY interface: Serial
Configure transceiver block
Lanes: x1
Xcvr ref_clk: 100 MHz
Application interface: Avalon-ST 64-bit
Port type: Native Endpoint
PCI Express version: 2.0
Application clock: 62.5 MHz
Max rate: Gen1 (2.5 Gbps)
Test out width: None
HIP reconfig: Disable
Info: Native Endpoint implementation requires MSI message 64-bit address capability.
Info: Native Endpoint implementation doesn't support I/O or 32-bit Prefetchable memory BAR types.
Cancel
< Back
Next >
Finish

Megawizard が “Specify a valid MegaWizard-generated variation file” と言ってファイルを開くことを拒否する場合、この問題にはいくつかの可能性があります。

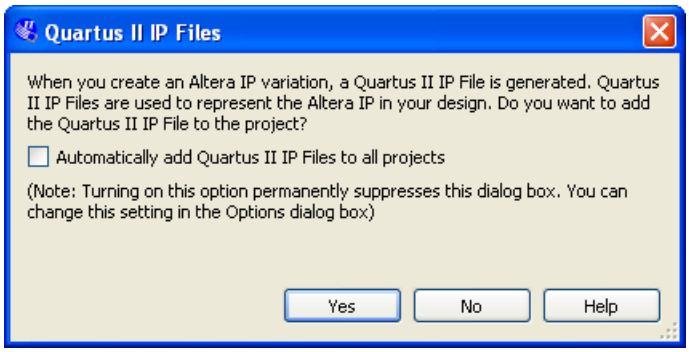
- 最近の FPGA (例: Cyclone V) を使用しています。この場合、PCIe ブロックをビルドする必要はまったくありません。
- Quartus のリビジョン番号を現在のものに変更するために、バリエーションファイルが適切に編集されていませんでした。このセクションの冒頭を参照してください。
- Megawizard がファイルを無効であるとして拒否した場合、ファイルを編集して再度ロードしようとしても役に立ちません。 Megawizard プログラムを終了し、コマンド prompt から再度起動する必要があります。そうしないと、ファイルが適切に編集されていても、引き続きファイルが拒否されます。

表示されるウィンドウと表示されるパラメーターは、FPGA ファミリごとに異なります (特に、“IP Compiler for PCI Express” とは異なるタイトルで問題ありません)。

変更は行わないでください。 “Finish”をクリックするだけです。次のようなウィンドウに進行状況が表示されます (最終的な状態がここに表示されます)。



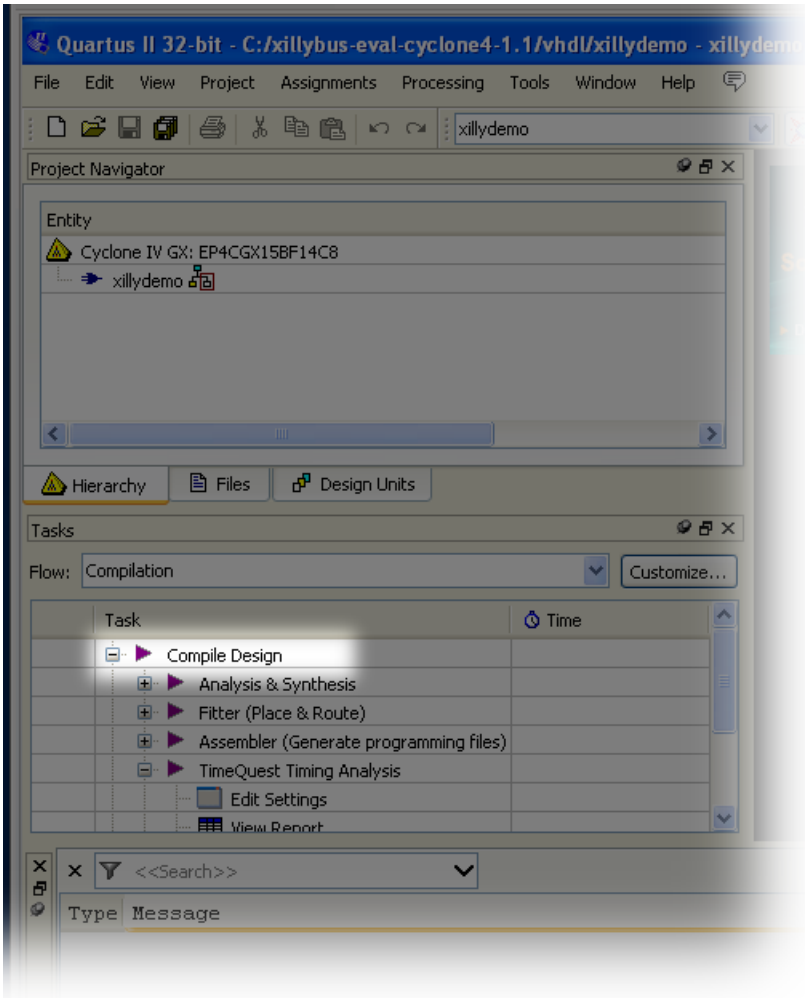
完了したら、[終了] をクリックします。次のウィンドウが直後に表示され、Quartus IP (QIP) ファイルをプロジェクトに追加するよう提案されます。



QIP ファイルをプロジェクトに追加すると、不要な SDC ファイルが間接的に追加され、constraints を含むすべて無視されるため、推奨される答えは “No” です。この QIP ファイルがプロジェクトに含まれていても問題はありませんが、compilation 中に大量の warnings が発生します。通常、SDC ファイルで無視された constraint ごとに 2 つの warnings が発生します。

3.3.3 design ファイルの Compilation

Quartus のメイン ウィンドウで、“Compile Design” をクリックして FPGA bitstream ファイルを作成します。一部の Quartus リビジョンでは、compilation のデフォルトの手順が “Rapid Recompile” に設定されている可能性があることに注意してください。これにより、タスク ペインで “Rapid Recompile” のみが許可されます。この場合は、“Compilation” に変更して続行します。



このプロセスでは、20 50 個の warnings が生成されますが (QIP ファイルがプロジェクトに含まれているかどうかによって異なります)、“Full Compilation was successful” というダイアログ ボックスが表示されて終了するはずです。

エラーも Critical Warnings も生成されていないことを確認することは必須です (メインの Quartus ウィンドウの下部にあるタブは、各タイプのメッセージの数を示します)。

プロセスの最後に、プログラミング ファイルは xillydemo.sof として見つかります。

3.4 FPGA のプログラミング

開発の初期段階では、JTAG 経由で FPGA をロードすることをお勧めします。これは通常、USB Blaster / Intel FPGA Download Cable (オンボードまたは外部) で行われます。 JTAG 経由で FPGA をロードする方法については、ボードの説明書を参照してください。

XillyUSB プロジェクトの場合、USB インターフェイスが動作中のコンピューターに接続されている場合でも、FPGA はいつでもロードおよび再ロードできます。

PCIe プロジェクトでは、コンピューターの電源を入れる前に、FPGA に bitfile をロードする必要があります。コンピューターは、電源を入れたときに PCIe 周辺機器が適切な状態にあることを期待しており、その後の予期せぬ事態に耐えられない場合があります。

したがって、host が動作している限り、FPGA をリロードし**ない**でください。PCIe 仕様では hotplugging のサポートが必要ですが、マザーボードは通常、PCIe カードが消えてから再び現れることを想定していません。したがって、一部のマザーボードは正しく応答しない場合があります。それでも、一部のマザーボードでは、オペレーティング システムの実行中に FPGA をリロードできます。

Xillybus の driver は hotplugging に正常に応答するように設計されていますが、コンピュータの一般的な安定性を保証するものは何もありません。これについては、次のページで説明しています。

http://xillybus.com/doc/hot-reconfiguration

FPGA の電源を入れ、コンピューターの電源を入れると同時に flash memory からロードする場合、BIOS が bus をスキャンするときに PCIe デバイスが存在するように、FPGA が十分に速くロードされることを確認することが不可欠です。

4

変更

4.1 カスタムlogicとの統合

Xillybus demo bundle は、application logic と簡単に統合できるように構築されています。データを接続する場所は、xillydemo.v または xillydemo.vhd ファイルです (優先言語によって異なります)。バンドル内の他のすべての HDL ファイルは、Xillybus IP core を使用して host (Linux または Windows) と FPGA の間でデータを転送する目的で無視できます。

カスタム logic designs を含む追加の HDL ファイルを、段落 3.3 の説明に従って準備されたプロジェクトに追加し、“Compile Design” をクリックして再構築することができます。初期展開の他の手順を繰り返す必要がないため、logic の開発サイクルは非常に迅速かつ単純です。

Xillybus IP core をカスタム application logic に接続する場合、少なくとも最初の段階では、FIFOs を介してのみ Xillybus IP core と対話し、logic でそれらの動作を模倣しようとしないことを強くお勧めします。

これに対する例外は、メモリまたは register arrays を Xillybus に接続する場合です。この場合、xillydemo モジュールに示されている例に従う必要があります。

xillydemo モジュールでは、FIFOs を使用して data loopback を実行します。つまり、host から到着したデータは host に送り返されます。 FIFO の両側が Xillybus IP core に接続されているため、core はデータのソースであり、データの消費者でもあります。

実際の使用シナリオでは、FIFO の片側だけが Xillybus IP core に接続されています。 FIFO のもう一方の側は、データを供給または消費する application logic に接続されています。

xillydemo モジュールで使用される FIFOs は、両側が Xillybus のメイン clock によっ

て駆動されるため、両側 (scfifo) に 1 つの共通 clock のみで動作します。実際のアプリケーションでは、読み取り用と書き込み用に別々の clocks を持つ FIFOs に置き換えることが望ましい場合があります。これにより、bus_clk 以外の clock を使用してデータ ソースとデータ コンシューマーを駆動できます。これにより、FIFOs はメディエーターとしてだけでなく、適切な clock domain クロッシングにも役立ちます。

Xillybus IP core は、FPGA から host への streams に対して (“show-ahead” とは対照的に)プレーンなFIFO インターフェースを想定していることに注意してください。

次のドキュメントは、カスタムlogicの統合に関連しています。

- logic design用のAPI：Xillybus FPGA designer’s guide
- Getting started with Xillybus on a Linux host
- Getting started with Xillybus on a Windows host
- Xillybus host application programming guide for Linux
- Xillybus host application programming guide for Windows
- The guide to defining a custom Xillybus IP core

4.2 Intel の PCIe IP / Megafunction に変更を加える

どうしても必要な場合を除き、Intel IP compiler for PCI Express Megafunction の構成を変更しないでください。

特に、receive buffers の割り当てには高い感度があります。 Xillybus IP core は、demo bundle の IP Compiler / MegaWizard によって指定された数値に従って、これらの buffers のサイズに依存します。 PCIe core が Xillybus IP core で予想されるよりも少ない buffer スペースを割り当てる場合、host から FPGA への通信で散発的なデータ エラーが発生する可能性があります。これにより、この方向の通信が完全に停止することもあります。このような問題は、これらの buffers の overflow を引き起こす FPGA に到着するパケットの結果です。

IP / Megafunction に何らかの変更が必要な場合は、Xillybus の Web サイトで公開されている電子メール アドレスから支援を求めてください。

4.3 カスタム プロジェクトへの組み込み

必要に応じて、Xillybus IP core を既存の Quartus プロジェクトに含めるか、新し

いプロジェクトを最初から作成することができます。このセクションは、PCIe の Xillybus に関連していますが、同様の手順が XillyUSB にも適用されます。

プロジェクトがまだ存在しない場合は、新しいプロジェクトを開始し、好みの HDL 言語と目的の FPGA に基づいて設定します。

Xillybus IP core をプロジェクトに含めるには:

- pcie_c4_1x.v ファイル (または同様のもの) を pcie_core/ サブディレクトリからプロジェクトに関連するディレクトリにコピーします。
- Cyclone V、Arria V、Stratix V、および series-10 FPGAs の場合、pcie_core/pcie_reconfig.qsys (または pcie_core/pcie_ip.ip) も同じディレクトリにコピーします。
- Intel の PCIe ブロックを構築するために、選択した FPGA で必要な場合は、3.3.2 で概説されている手順に従ってください。
- これらのファイルは、Cyclone IV 用に追加する必要があります (おそらくそれらのコピー)。
  - pcie_c4_1x.v
  - pcie_c4_1x_core.v
  - pcie_c4_1x_serdes.v
  - pcie_c4_1x_examples/chaining_dma/pcie_c4_1x_rs_hip.v
  - ip_compiler_for_pci_express-library/altpcie_hip_pipen1b.v
  - ip_compiler_for_pci_express-library/altpcie_reconfig_3cgx.v
  - ip_compiler_for_pci_express-library/altpcie_rs_serdes.v

  Stratix IV および Arria II の場合、次のファイルを追加する必要があります。

  - pcie_s4_4x.v
  - pcie_s4_4x_core.v
  - pcie_s4_4x_serdes.v
  - pcie_s4_4x_examples/chaining_dma/pcie_s4_4x_rs_hip.v
  - ip_compiler_for_pci_express-library/altpcie_hip_pipen1b.v
  - ip_compiler_for_pci_express-library/altpcie_reconfig_4sgx.v
  - ip_compiler_for_pci_express-library/altpcie_rs_serdes.v

  series-V FPGAs の場合、同様のファイルは Qsys によって生成され、含まれているため、コピーする必要はありません。

- 2 つの HDL ファイル、xillybus.v および xillydemo.v(hd) を 2 つの src/ サブディレクトリのいずれかにコピーし (言語設定に応じて)、プロジェクトに追加します。
- src/ ディレクトリのいずれかにある SDC ファイルで constraints を採用します (プロジェクト内の top level 信号の名前と一致するように、わずかな変更が必要になる場合があります)。
- core/ ディレクトリから xillybus_core.qxp または xillybus_core.vqm をコピーし、このファイルもプロジェクトに追加します。
- xillydemo モジュールがプロジェクトの top level module でない場合は、そのポートを top level に接続します。
- Xillybus IP core をカスタム application logic に接続するには、xillydemo モジュールを編集して、既存の application logic を目的のものに置き換えます。

4.4 他のボードを使用する

4.4.1 全般的

demo bundles のリストにないボードを使用する場合は、バンドルに若干の変更が必要です。

4.4.2 デバイスファミリー

demo bundle には、特定のデバイス ファミリ用の synthesized ファイルである QXP または VQM ファイルが含まれています。このファミリ内のどのデバイスでも使用できます。

4.4.3 Pin placements

購入したほとんどのボードには、そのボードで PCIe インターフェイスがどのように使用されているかを示す FPGA design の独自の例があります。多くの場合、目的のボードの QSF ファイルで関連するピン割り当てを見つけ、ピンの名前を Xillybus の QSF ファイルで使用されている名前に変更するのが最も簡単です。これを行うと、Xillybus のプロジェクトで使用される QSF ファイル内の関連する行を置き換えることができます。

機能上の重要性を持たない 4 本の user_led ワイヤもあります。ただし、ボード上に空いている LEDs がある場合は、それらを接続することをお勧めします。これ

は、通信ステータスに関するいくつかの指標を提供するためです。 user_led 信号は active low logic を想定しています (logic '0' は LED がオンであることを意味します)。

4.4.4 PCIe のみ: Clocking

xillydemo モジュールは、これら 3 つの clocks の一部またはすべてを入力として必要とします。必要な clocks は、xillydemo モジュールへの入力です。

- pcie_refclk– host のマザーボードによって提供される reference clock に直接接続されます。この clock の周波数は 100 MHz である必要があります。この clock は、host の電源がオフになっている場合や、PCIe 仕様で許可されているその他の状況ではアクティブにならない場合があります。マザーボードの clock と FPGA に接続されている FPGA の必要に応じて、clock クリーニング回路が存在する場合があります。

  異なる clock 周波数がこの入力ポートに供給される場合 (clock クリーナーによって供給される 125 MHz など)、それに応じて PCI Express 設定の IP Compiler の Xcvr_ref_clk パラメータを変更する必要があります。これは、段落 3.3 に示すように、関連する IP Compiler / MegaWizard plug-in manager を呼び出すときに行われます。

- clk_50 – gigabit transceiver の reconfig_clk を駆動し、常にアクティブにする必要があります。したがって、マザーボードの reference clock に依存してはなりません。この clock の許容周波数範囲は、FPGA ファミリーによって異なります (たとえば、Cyclone IV 上の 37.5 MHz から 50 MHz まで)。この clock の仕様は、関連する FPGA の device handbook で "reconfig_clk" を検索すると見つかります。

- clk_125 – transceiver に必要な常時アクティブ clock を駆動します。この clock の周波数は 125 MHz でなければならず、マザーボードの reference clock に依存してはなりません。

重要:

pcie_refclk は、ボード上の一部の発振器だけで駆動してはなりません。 *host* の *clock* とのわずかな周波数の違いにより、通信の信頼性が低下したり、*FPGA* を *PCIe* デバイスとして検出できなかったりします。

Cyclone IV GX Transceiver Starter Board 用の demo bundle では、clk_50 および clk_125 入力は、ボード上の 50 MHz および 125 MHz の clock ソースにそれぞれ接

続されている I/O ピンによって直接駆動されます。ボードから直接適切な clocks がない場合、Intel によって公開されている IP Compiler for PCI Express User Guide のセクション 7 で説明されているように、PLL を使用して両方を生成できます。

PLL を使用する場合、xillybus.v を編集して、reconfig_clk_locked 信号を PLL のロック インジケータに接続する必要があります。 DE4 ボードは 125 MHz 周波数で常にアクティブな clock を提供しないため、これは Stratix IV 用の demo bundle で既に行われています。

PLL の使用例は、PCI Express ファイル用の PCI Compiler が生成された後、pcie_core サブディレクトリ内のファイルの中にあります。これは、 pcie_c4_1x_examples/chaining_dma ディレクトリ (または Cyclone IV 以外のファミリで作業する場合は同様のファイル) の pcie_c4_1x_example_chaining_pipen1b.v として見つけることができます。

4.4.5 XillyUSB の操作

XillyUSB は、SFP+ インターフェイスを持つ他のボードで使用できます。この場合、SFP+ コネクタに配線された MGT を使用するように design の constraints を設定するだけです。

ボードは、MGT 用に低 jitter を備えた 125 MHz reference clock も提供する必要があります。 USB 仕様の要件にもかかわらず、Spread Spectrum Clocking (SSC) を有効にしないでください (そのようなオプションが存在する場合)。SSC reference clock が使用されている場合、MGT は受信信号を適切にロックしません。

カスタム ボードの場合、SFP+ コネクタのピンが FPGA の MGT に直接接続されているため、sfp2usb モジュールの回路図を参照することをお勧めします。 sfp2usb モジュールで行ったように、SSRX ワイヤを交換することはオプションです。これは、PCB design を簡素化する場合にのみ推奨されます。

必要に応じて、SSTX ワイヤを交換することもできます。これには、*_frontend.v ファイルを編集して、送信ビットの極性を反転し、ワイヤ ペア スワップを補正する必要があります。 USB の仕様では、極性を入れ替えても link partners が正しく動作することが求められているため、この編集がなくても USB 接続が正しく動作する可能性が高いことに注意してください。ただし、これに依存しないことをお勧めします。

5

トラブルシューティング

5.1 implementation中のエラー

Intel のツールのリリース間で若干の違いがある場合があります。これにより、SOF ファイルを作成するための implementation の実行に失敗する可能性があります。

問題がすぐに解決しない場合は、Xillybus の Web サイトにある電子メール アドレスから支援を求めてください。失敗したプロセスの出力ログを添付してください。特に、ツールによって報告された最初のエラーの前後に注意してください。また、design でカスタム変更 (つまり、demo bundle からの流用) が行われた場合は、これらの変更について詳しく説明してください。また、使用された Quartus ツールのバージョンもお知らせください。

5.2 PCIe ハードウェアの問題

通常、PCIe カードは host の BIOS および/またはオペレーティング システムによって適切に検出され、host の driver は正常に起動します。

ほとんどの PC コンピューターでは、BIOS は boot プロセスの早い段階で、検出された周辺機器のリストを短時間表示します。 Xillybus インターフェイスが正常に検出されると、vendor ID 1172 および device ID EBEB を含む周辺機器がリストに表示されます。

オペレーティング システムによるカードの検出については、次の 2 つのドキュメントのいずれか該当する方を参照してください。

- Getting started with Xillybus on a Linux host
- Getting started with Xillybus on a Windows host

カードの検出の失敗 (またはコンピューターの boot プロセスの失敗) は、bus とのインターフェイスを Intel の PCIe IP core に依存している Xillybus IP core とは関係ありません。

最初に、次のことを確認することをお勧めします。

- bitstream は、コンピューターの電源を入れたとき (または PCI-SIG 仕様に関しては、電源を入れた直後) に既に FPGA にロードされていました。
- reference clock を含む PCIe ワイヤの pinouts は正しいです (これは fitter のレポートで確認できます)。
- ボードは、正しい reference clock を FPGA に提供します。

問題がすぐに見つからない場合は、ボードに付属の PCIe のサンプル プロジェクトを試すことをお勧めします。これにより、不適切なジャンパ設定や、ハードウェアの欠陥が明らかになる場合があります。

このサンプルではカードが検出され、Xillybus では検出されない場合は、2 つの designs の pinouts を比較すると役立つ場合があります。それらが等しい場合、次のステップは、関連する GUI ツールをそれぞれで呼び出して、Intel の PCIe cores の属性を比較することです。

次の構成要素は、調整が必要な場合があります。

- reference clock の周波数。
- base class および sub class (可能性は低いですが、class が不明であると見なされた場合、一部の比較的古い PC コンピューターは boot プロセスに失敗しました)。
- ベース アドレスの register 設定、vendor ID、device ID、および割り込み設定を除き、変更しないでください。

問題が解決しない場合は、Xillybus の Web サイトに記載されている電子メール アドレスから支援を求めてください。