XAPP1201 (v1.0) 2014 年 1 月 23 日

Virtex-7 (XT/HT) および UltraScale の Gen3 Integrated Block for PCI Express コアから AXI4-Lite へのブリッジ

著者 : Luis Bielich

概要

このアプリケーション ノートでは、Gen3 Integrated Block for PCI Express IP コアのコンプリータ ストリーミング インターフェイスから AXI4-Lite マスター インターフェイスへのブリッジの使用について説明します。リファレンス デザインは、Vivado® IP インテグレーターで Integrated Block for PCI Express IP コアへ接続するパッケージ化された IP コアを提供します。この IP が使用する LUT は 300 個未満です。AXI4-Lite マスター ポートは、AXI4 スレーブ インターフェイスを備えるペリフェラルへ接続します。

はじめに

Virtex®-7 (XT および HT) デバイス ファミリおよび UltraScale™ のアーキテクチャには、Gen3 PCI Express 用の統合ブロックがハードとして備えられています。このハード ブロックは高性能システム用に設計されていますが、エンドポイントがホストから受信する dword (DW) 要求は通常 1 つのみです。1 つの DW 要求で DMA エンジンをセットアップしたり、あるいはこれを使用して AXI4 ベースのシステムにあるペリフェラル レジスタの監視や変更が可能です。

Integrated Block for PCI Express IP コアはストリーミング インターフェイスを提供するため、AXI4-Lite インターコネクト上の制御側ペリフェラルへのアクセスには通常 AXI4 へのブリッジが使用されます。受信するすべての単一 DW 要求は、Integrated Block for PCI Express IP コアの CQ (Completer reQuest) と CC (Completer Completion) インターフェイスで動作します。ブリッジは、この CC と CQ インターフェイスのみを使用して AXI4-Lite インターフェイスへ接続します。高性能アプリケーションではエンドポイントがマスターとなり、複数の DW 要求をアップストリーム方向に伝送します。エンドポイントがマスターになる場合は、RQ (Requester reQuest) と RC (Requester Completion) インターフェイスが使用されます。ブリッジはいずれのリクエスター インターフェイスも使用しないため、これらの高性能ポートはバス マスタリング用にそのまま使用できます。図 1 に、コンプリータ インターフェイスはブロック RAM へのアクセスに使用され、リクエスター インターフェイスはバス マスタリング用にオープン (未接続) となっているシステムの全体図を示します。

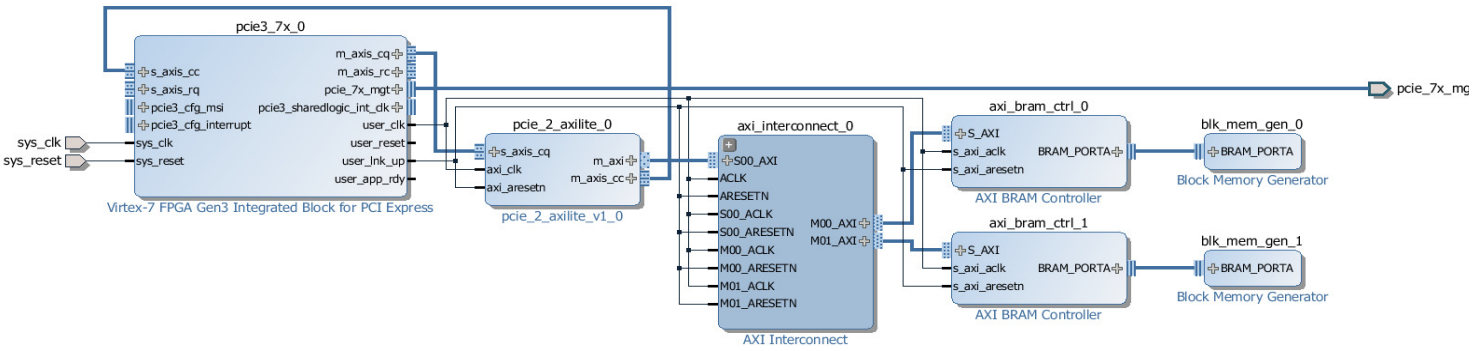

図 1 : IP インテグレーターのサブシステム

図 1 に示すシステムでは、ローカル ブロック RAM をメモリ マップされたストレージとして接続しています。Vivado IP カタログにはこの他にも接続可能な多数の AXI ペリフェラルがあります。図 1 のブロック RAM と同じように接続できるペリフェラルの例をいくつか挙げます。

- AXI Quad SPI

- AXI UART Lite
- AXI Timer
- AXI IIC Bus Interface
- AXI Ethernet Lite MAC
- AXI GPIO
- AXI EMC

このアプリケーション ノートでは、CQ/CC インターフェイスを AXI-Lite インターフェイスに変換するための RTL (Resistor–Transistor Logic) を提供しています。この RTL は IP としてパッケージ化されているため、IP インテグレーター内で接続、あるいは IP モジュールとしてインスタンシエートできます。RTL は暗号化されていないため、ブリッジは必要に応じてカスタマイズできます。

機能

ブリッジは Vivado IP コアとしてパッケージ化されており、次の機能をサポートします。

- 単一 DW のメモリ読み出しおよびメモリ書き込み要求
- 最大 6 個のベース アドレス レジスタ (BAR)
- 32 ビットおよび 64 ビットの BAR
- トランザクション層パケット (TLP) のヘッダー アドレスから AXI4 アドレッシングへのアドレス変換
- すべてのレーン幅とスピード (gen1、gen2、または gen3) で、コンプリータ インターフェイスの全データ幅のサポート
- Vivado IP インテグレーターのサポート
- ソースの提供 (Verilog のみ)
- データ アライン モードのみ (アドレス アライン モードはサポート外)

ハードウェアの説明

ブリッジは、Integrated Block for PCI Express IP コアの CC および CQ インターフェイスのみを使用します。このハード ブロックの `m_axis_cq` インターフェイスがブリッジの `s_axis_cq` インターフェイスに直接接続し、ハード ブロックの `s_axis_cc` がブリッジの `m_axis_cc` インターフェイスに接続します。この IP コアの `user_clk` 出力は、CC および CQ インターフェイスのクロック信号と同期し、ブリッジを駆動するクロックとして機能します。`axi_aresetn` は、ブリッジをリセットする非同期のアクティブ Low 信号であり、ブリッジがリセット状態を維持している間パケットはブリッジを通過できません。通常は統合ハードブロックの `user_lnk_up` 出力をブリッジのリセット信号として使用しますが、別の信号をリセットに使用することも可能です。図 2 に、IP インテグレーターでブリッジへ接続された CQ インターフェイスと CC インターフェイス、そして対応するクロック信号とリセット信号を示します。

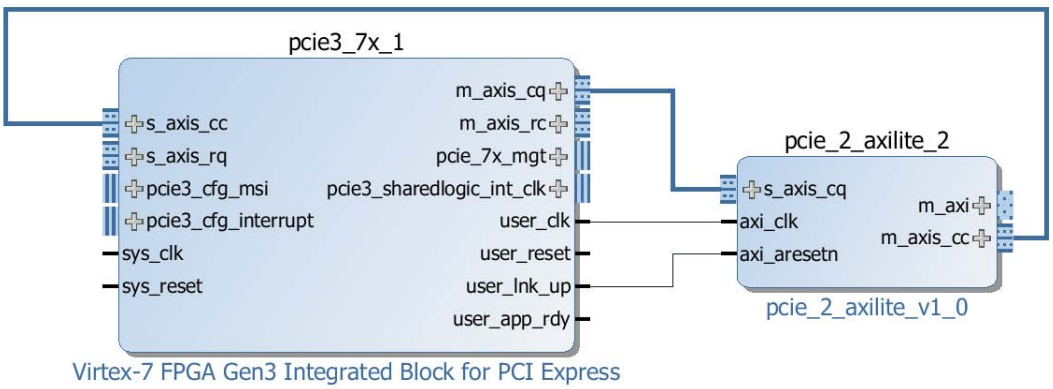

図 2 : コンプリータ インターフェイスの接続

CQ インターフェイス (m_axis_cq) は、ホストからのメモリ読み出し/メモリ書き込み要求を与えます。このインターフェイスからのメモリ読み出し/メモリ書き込み要求をブリッジがデコードし、AXI4-Lite システムのマスターへ変換して渡します。ホストからのメモリ書き込み要求は、AXI4 の書き込みアドレス チャネルと AXI4 の書き込みデータ チャネルのトランザクションに変換されます。AXI4 の書き込み応答チャネルが接続されていますが、ブリッジでは使用されません。また、AXI4 の書き込み応答チャネルの READY 信号がアサートされますが、このチャネルから送信されるデータは無視されます。

PCI Express では、ホストがメモリの読み出しを要求すると、データ TLP と共にコンプリーション信号が返されます。CQ インターフェイスを介してメモリの読み出しが要求されると、まずブリッジが読み出し要求を AXI4 の読み出しアドレス チャネルのトランザクションに変換します。次に AXI4 スレーブが AXI4 の読み出しデータ チャネルにデータを与えて応答します。ブリッジは AXI4 の読み出しデータ チャネルのデータを受信して、AXI4 の読み出しデータ チャネルからのペイロードと共に CC インターフェイス (m_axis_cc) にコンプリーション TLP を作成します。図 3 に、トランザクション フローの概要図を示します。

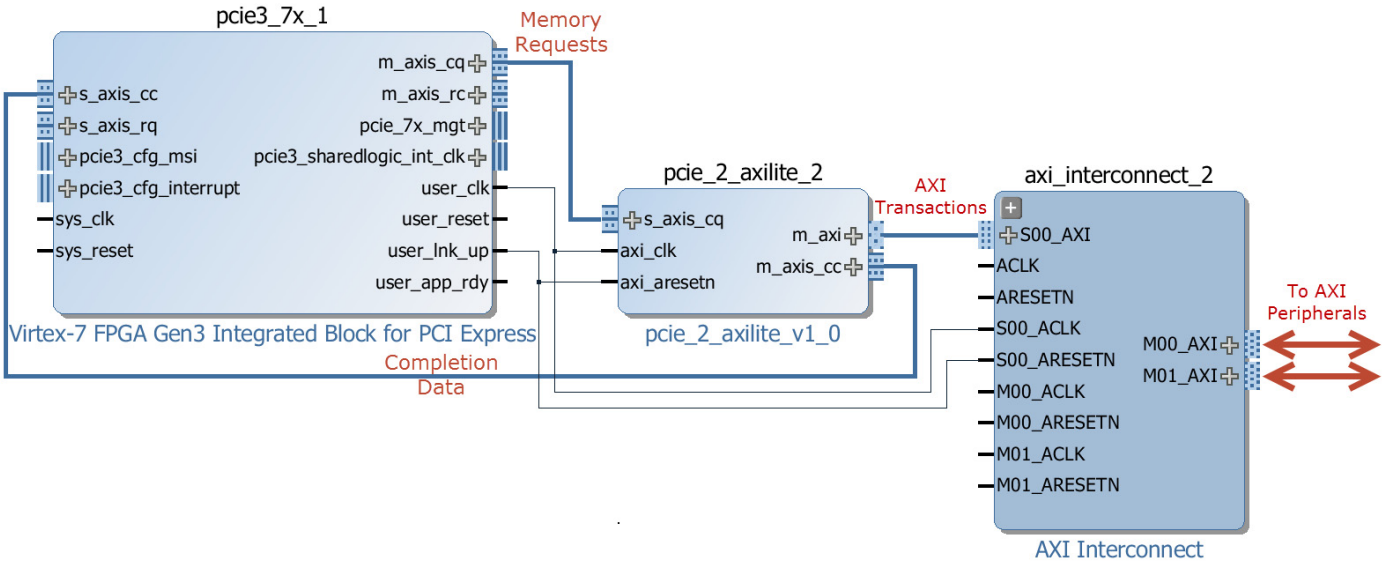

図 3 : インターフェイス接続および対応するトラフィック タイプ

アドレス変換

Integrated Block for PCI Express IP コアからの CQ 記述子には TLP からのアドレスが含まれます。図 4 に CQ 記述子を示します。DW+0 と DW+1 が TLP からのアドレスです。

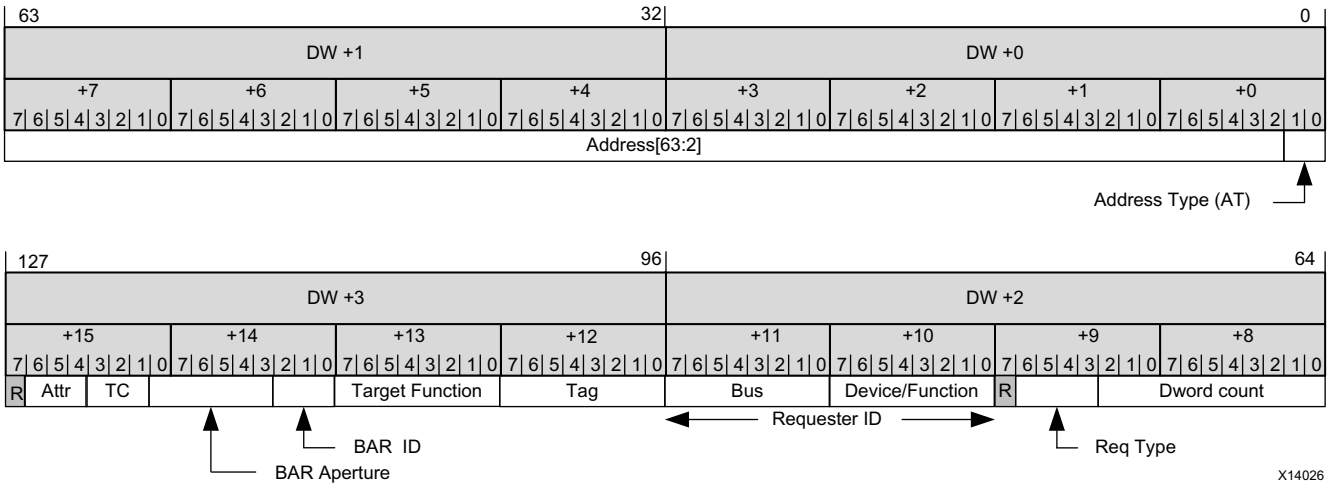

図 4 : CQ 記述子

記述子で提供されるアドレスは、PCI Express の TLP から直接きています。TLP アドレスは AXI4 システムに変換されます。アドレスはまた、BAR ヒット情報に応じて異なる AXI4 アドレスへ変換されます。IP コアは、[BAR Options] タブにある AXI4 空間への変換のオプションでパッケージ化されます。図 5 に有効なオプションを示します。

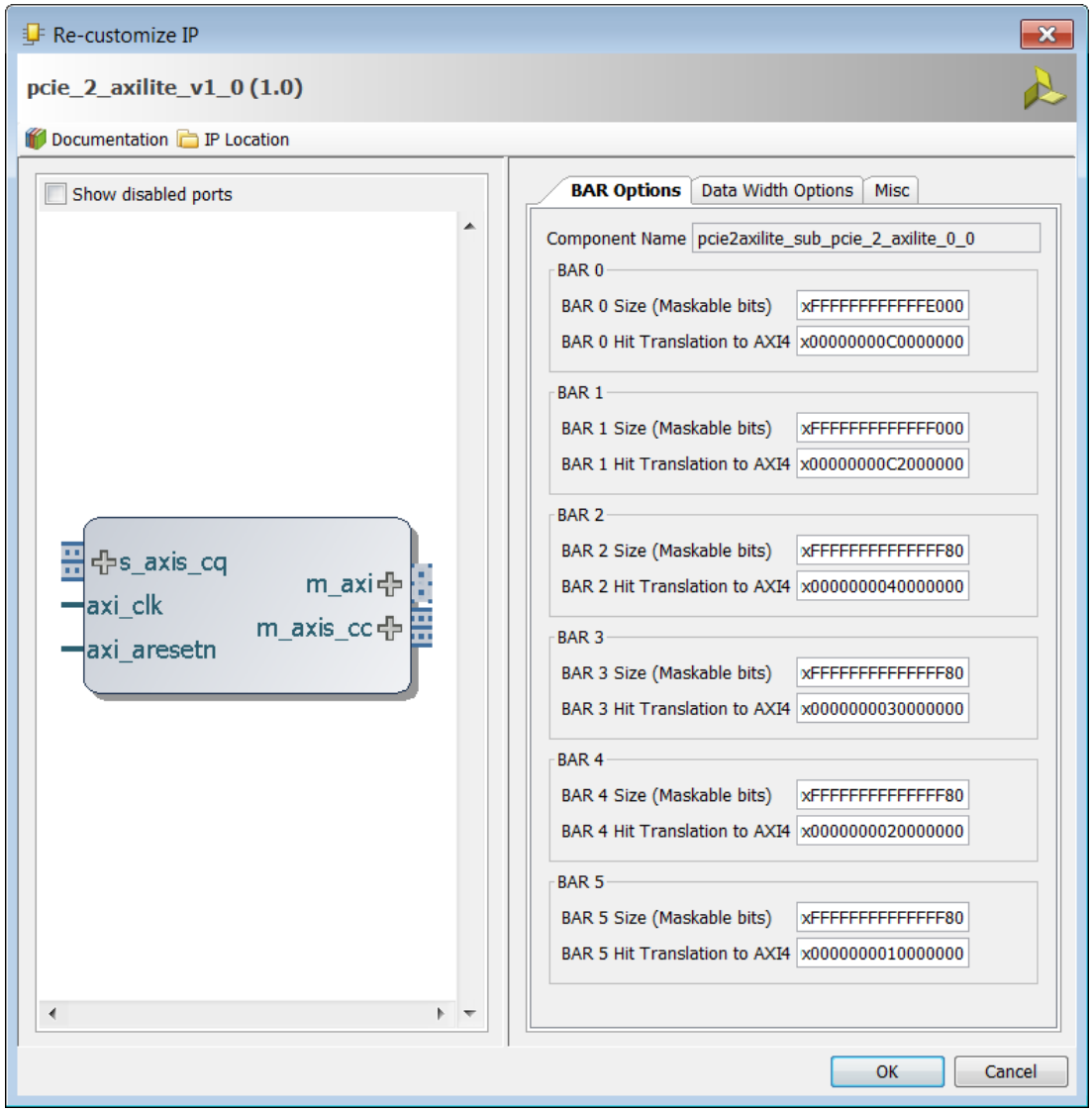

図 5 : BAR 変換オプション

BAR サイズは、割り当てられたマスク可能なビットの量によって変わります。IP の BAR サイズは、エンドポイントに設定される BAR サイズと同じになる必要があります。表 1 に、[BAR # Size] フィールドの有効な値と対応するアパーチャー サイズを示します。

表 1 : マスク可能なビットと BAR サイズの例

有効なマスク可能な値	対応する BAR のアパーチャー サイズ
0xFFFFFFFFFFFFFF80	128 バイト (最小)
0xFFFFFFFFFFFFFF00	256 バイト
0xFFFFFFFFFFFFFE00	512 バイト
0xFFFFFFFFFFFFFC00	1KB
…	…
0xFFFFFFF800000000	32GB
0xFFFFFFF000000000	64GB
0xFFFFFFE000000000	128GB
0xFFFFFFC000000000	256GB (最大)

[Translation to AXI4] 値は、BAR ヒットに応じて変換される AXI4 アドレス空間のベース アドレスを提供します。[Maskable bits] に示されるアサートされたビットは、PCI Express 空間の対応するベース アドレス オフセットを除外し、BAR ヒットからのオフセットのみを提供します。たとえば、1KB BAR が 0x00000000C0000000 にエニュメレートされる場合、マスクされたビットによって上位 54 ビット (マスクは 0xFFFFFFFFFFFFFC00 であるため) が除外され、下位 10 ビットのみで [Translation to AXI] オプションからのオフセットが判断されます。下位 10 ビットはオフセット用に使用されているため、[Translation to AXI] では下位 10 ビットをオフセット値として使用しないでください。これはアパーチャー サイズとアドレスのアライメントを維持しないためです。[Translation to AXI4] アドレスを決める際は、これを必ず考慮する必要があります。

表 2 に、BAR サイズと変換オプションの例を 2 つ示します。例 1 は、特定の BAR へ割り当てられた 1KB のアドレス指定可能な空間を持つブリッジのセットアップを示しています。この 1KB 範囲は、ブリッジの [Maskable bits] オプションからきています。この BAR は、ホストによってアドレス x000000000C000000 へエニュメレートされました。ホストが BAR レジスタのオフセット アドレス 4 から要求し、これは TLP アドレス x000000000C000004 となります。これが BAR レジスタのエニュメレートによる結果です。記述子アドレス x000000000C000004 からマスクによって上位ビットをマスクすると、アドレスはオフセット アドレス 4 になります。このオフセットを AXI ドメインへ変換するため、オフセットに [Translation to AXI4] オプションを追加します。その結果、AXI ドメインのアドレスは x0000000080000004 となります。これが、記述子から AXI ドメインへの変換プロセスです。

表 2 の例 2 では、記述子アドレスから変換されるもう 1 つの AXI アドレスの例とその変換プロセスを示します。

表 2 : アドレス変換の例

	例 1	例 2
BAR のエニュメレートされたアドレス (ホストで割り当てられる)	x000000000C000000	x0000000000008000
記述子のアドレス (ホストからの要求)	x000000000C000004	x00000000000080CC
[Maskable bits] (ブリッジのオプション)	xFFFFFFFFFFFFFC00	xFFFFFFFFFFFFF000
BAR サイズ	1KB	4KB
[Translation to AXI] (ブリッジのオプション)	x0000000080000000	x0000000040000000
結果の AXI アドレス	x0000000080000004	x00000000400000CC

図 6 には表 2 の例 1 のフロー図を示します。

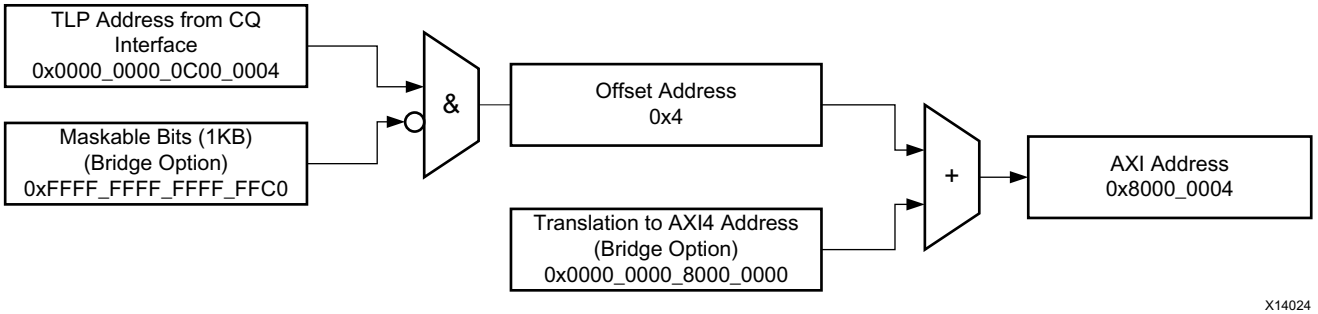

図 6 : 例1 のフロー図

図 7 に表 2 の例 2 のフロー図を示します。

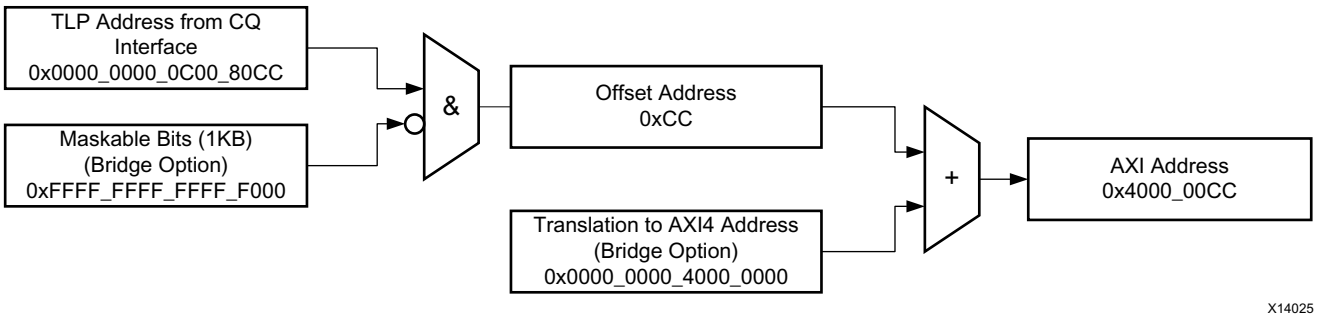

図 7 : 例2 のフロー図

図 8 に、Translation to AXI4 (AXI4 への変換) を実行する RTL コードを示します。[BAR#SIZE] の値は、[Maskable bits] フィールドに設定された、最も右にある値の低いビットによって決まります。たとえば、xFFFFFFFFFFFFC00 の場合は、11 番目のビットがこのビットに該当するため、[BAR#SIZE] は 10 となります。[BAR#SIZE] の値は [Translation to AXI4] フィールドから派生します。

```
always @(mem_req_bar_hit, mem_req_pcie_address)
  case (mem_req_bar_hit)
     3'b000: m_axi_addr_c <= { BAR0AXI[M_AXI_ADDR_WIDTH-1:BAR0SIZE], mem_req_pcie_address[BAR0SIZE-1:2],2'b00};
     3'b001: m_axi_addr_c <= { BAR1AXI[M_AXI_ADDR_WIDTH-1:BAR1SIZE], mem_req_pcie_address[BAR1SIZE-1:2],2'b00};
     3'b010: m_axi_addr_c <= { BAR2AXI[M_AXI_ADDR_WIDTH-1:BAR2SIZE], mem_req_pcie_address[BAR2SIZE-1:2],2'b00};
     3'b011: m_axi_addr_c <= { BAR3AXI[M_AXI_ADDR_WIDTH-1:BAR3SIZE], mem_req_pcie_address[BAR3SIZE-1:2],2'b00};
     3'b100: m_axi_addr_c <= { BAR4AXI[M_AXI_ADDR_WIDTH-1:BAR4SIZE], mem_req_pcie_address[BAR4SIZE-1:2],2'b00};
     3'b101: m_axi_addr_c <= { BAR5AXI[M_AXI_ADDR_WIDTH-1:BAR5SIZE], mem_req_pcie_address[BAR5SIZE-1:2],2'b00};
     3'b110: m_axi_addr_c <= 32'd0;
     3'b111: m_axi_addr_c <= 32'd0;
  endcase
```

図 8 : BAR 変換の RTL

データ幅およびアドレス幅

AXI マスターのデータ幅は 32 ビットで固定されています。一方、コンプリーター インターフェイス (CC および CQ) のデータ幅はブリッジの [AXI Streaming Data Width] で変更可能です。このオプションの値は、Integrated Block for PCI Express IP で設定されている幅と一致する必要があります。[AXI Master Address Width] は、マスター AXI インターフェイスがアクセスできるアドレス指定可能な空間を指定します。32 に設定した場合、ブリッジはマスターが 4GB のアドレス空間へアクセスできるように許可します。33 に設定した場合は、8GB の空間が可能になります。この値は BAR サイズとは無関係です。

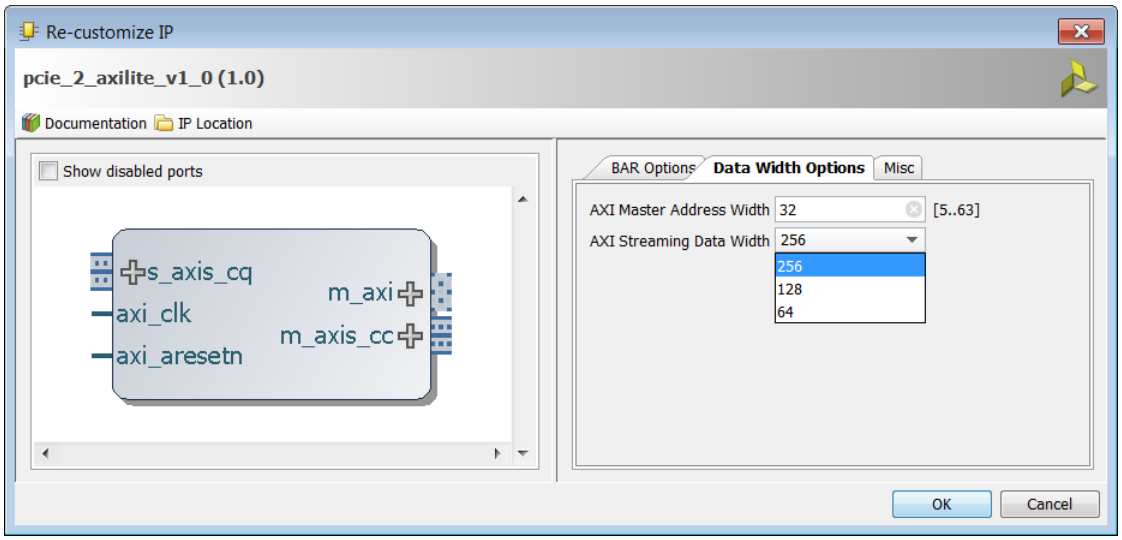

図 9 : データ幅およびアドレス幅のオプション

その他のオプション

ブリッジは、データが AXI インターフェイスから返される前に、PCI Express ブロックからの複数の読み出し要求を受けることが可能です。読み出し要求がブリッジで受け入れられ、ブリッジがまだデータ返していない場合、この状況を「outstanding read request (未処理の読み出し要求)」といいます。ブリッジ オプションでは、未処理の読み出し要求の数も指定できます。図 10 では、未処理の読み出し要求数は 32 (2^5) ～ 256 (2^8) の範囲で設定可能です。

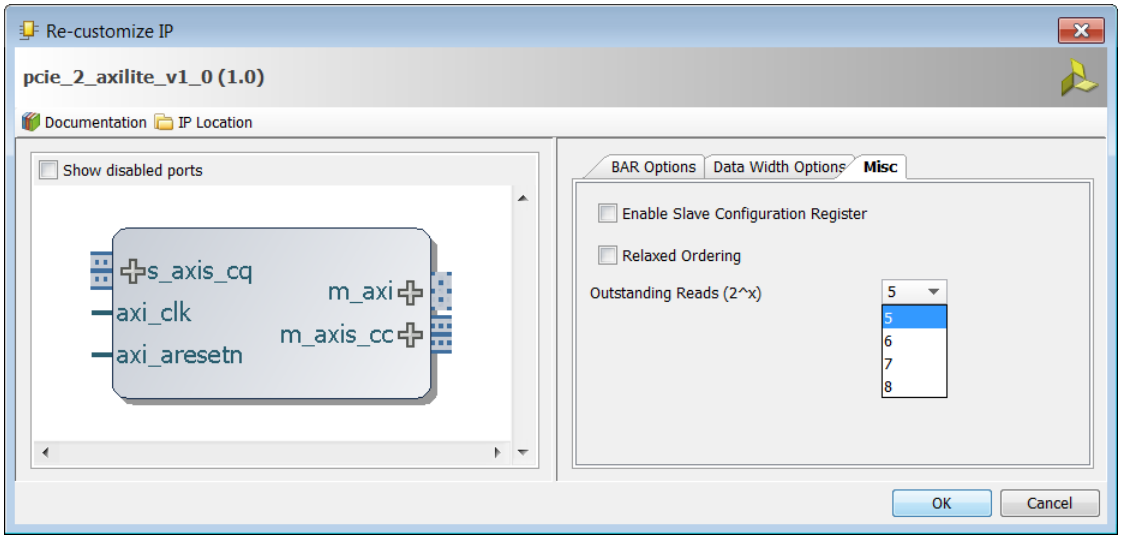

図 10 : その他のオプション

ほかにも 2 つのオプションで独自の設定が可能です。[Enable Slave Configuration Register] は、スレーブ インターフェイスに機能を追加できるようにします。このオプションを有効にした場合は、`pcie2axilite_bridge\rtl` ディレクトリにある RTL を変更して、カスタマイズしたオプションを追加する必要があります。スレーブ インターフェイスをカスタマイズする事例を次に示します。

- 動的な BAR の変換
- エラー条件
- デバッグ レジスタ

[Relaxed Ordering] は、TLS が相互に通信できるようにします。この場合、PCI Express の仕様に反する可能性があるため、慎重な解析を行った上で使用することを推奨します。

リファレンス デザインの実行

このアプリケーション ノートに関連するデザイン ファイルは、こちらからダウンロードしてください。

リファレンス デザインは、3 つの異なるユーザー インターフェイス幅 (64、128、256 ビット) に設定された 3 つのブリッジ例を示します。Vivado IDE で、`dw#/build` ディレクトリにある `build_design_#.tcl` を読み込みます (# はデータ幅)。

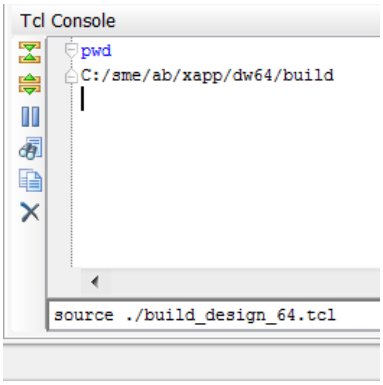

図 11 : Vivado IDE でサンプル デザインを実行

Tcl スクリプトを読み込むと Vivado プロジェクトが生成され、そのプロジェクトがビット ファイルを生成します。生成されたビット ファイルは、『PCI Express Endpoint コアの Programmed Input/Output サンプル デザインにおける Memory Endpoint Test (MET) ドライバーの使用』[参照 2] で提供されている MET ドライバーとアプリケーションを使用して機能します。MET ドライバーを 1 箇所で管理するため、このアプリケーション ノートではソフトウェア ファイルを提供していません。最新の MET ドライバーについては、『PCI Express Endpoint コアの Programmed Input/Output サンプル デザインにおける Memory Endpoint Test (MET) ドライバーの使用』[参照 2] を参照してください。

リファレンス デザインには、3 つの異なるインターフェイスに ILA (Integrated Logic Analyzer) コアが含まれ、TLP が送信される際のトラフィック フローを解析できます。ILA コアを備えた 3 つのインターフェイスは次のとおりです。

- CQ インターフェイス
- CC インターフェイス
- AXI4-Lite マスター

Valid 信号の立ち上がりエッジでのトリガーは、ブリッジを通過しているパケットを示します。

シミュレーション

TLP から AXI4 へのトランザクションへの変換を示すためにリファレンス シミュレーションを提供しています。サンプル シミュレーションのテスト ベンチでは、メモリ書き込み要求とメモリ読み出し要求に CQ インターフェイスを使用します。変換後の AXI トランザクションは、それらの要求に応答するように AXI4 BRAM モジュールへ指示します。

シミュレーションの実行には、`simulation` ディレクトリのバッチ ファイルを使用します。この `run_sim.bat` スクリプトは、Vivado Simulator を使用してシミュレーションをコンパイル、エラボレート、そして実行します。`pcie_2_axilite_tb.v` ファイルには、インターフェイス幅を 64 ビット、128 ビット、または 256 ビットに設定するためのパラメーター `C_DATA_WIDTH` が含まれます。トランザクションの変更には、`cq_axis_stimulus.v` モジュールに次の 4 つの呼び出しがあります。

- pcie_write (address, data, enable bits, data width)
- pcie_read (address, data width)
- write_seq (write count, address, data, data width)
- read_seq (read count, data width)

構築済みのイメージを使用

3 つすべてのインターフェイス幅用にあらかじめ構築済みのビットストリーム (.bit) とハードウェア アナライザー ファイル (.ltx) を提供しています。これらのビットストリームは、『PCI Express Endpoint コアの Programmed Input/Output サンプル デザインにおける Memory Endpoint Test (MET) ドライバーの使用』[参照 2] で提供されている MET ドライバーを使用して VC709 ボード上で機能します。構

築済みビットストリームを使用する場合は、次の手順に従ってください。

1. プロジェクトを開かずに Vivado IDE を開いて、Tcl コンソールに次を入力します。

   ```
   >> open_hw
   ```

2. JTAG 接続を確立し、VC709 ボードの Virtex-7 ファミリをターゲットにしたビットストリームと ltx ファイルを選択します。
3. FPGA をプログラムし、デザイン内のすべての ILA コアを確認します。
4. すべての AXI インターフェイスの任意の Valid 信号で立ち上がりエッジを検出するためにトリガーを設定します。
5. 『PCI Express Endpoint コアの Programmed Input/Output サンプル デザインにおける Memory Endpoint Test (MET) ドライバーの使用』[参照 2] で提供されている PIO アプリケーションを実行すると、トラフィックが ILA コアに現れます。

リソース使用状況

表 3 に、VC709 ボードにおける Virtex-7 のリソース使用状況を示します。。

表 3 : リソース使用状況

TDATA 幅	LUT	FF	RAM
x64	277	276	0
x128	289	297	0
x256	289	297	0

ファイルの説明

表 4 では、リファレンス デザインのディレクトリ構造について説明します。

表 4 : リファレンス デザイン ファイル

ディレクトリとファイル	説明
< data width: dw64, dw128, dw256 > <build> build_design_[data width].tcl	ビットストリームを構築するための Tcl ファイルです。
< data width: dw64, dw128, dw256 > <source> <constaints> top.xdc	ロケーション制約と Vivado ハードウェア デバッグ制約を含む制約ファイルです。
< data width: dw64, dw128, dw256 > <source> <ipi> ipi_design_[data width].tcl	IPI システムを構築するための Tcl ファイルです。
< data width: dw64, dw128, dw256 > <source> <rtl> pcie2axilite_bridge.v	IPI システム用の最上位ラッパー ファイルです。
< pcie2axilite_bridge > <rtl> Verilog ファイル	IP ソース ファイルです。

表 4 : リファレンス デザイン ファイル (続き)

ディレクトリとファイル	説明
< pcie2axilite_bridge > <xgui> pcie_2_axilite_v1_0.tcl	GUI 用の Vivado パッケージャー Tcl ファイルです。
< pcie2axilite_bridge > component.xml	Vivado IP パッケージング ファイルです。
< simulation > run_sim.bat/run_sim.sh run_time.tcl source.prj xsim_test.wcfg	Vivado Simulator でシミュレーションを実行するためのファイルです。
< simulation > < verilog > *.v	シミュレーション テスト ベンチ ファイルです。
注記 : 1. <> はディレクトリを表します。	

まとめ

PCI Express のエンドポイントがホストから受信する DW 要求は通常 1 つのみです。要求は、Integrated Block for PCI Express IP コアの CQ インターフェイスと CC インターフェイスで動作します。高性能アプリケーションではエンドポイントがマスターになり、複数の要求アップストリームを構成します。エンドポイントがマスターになる場合には、RQ と RC インターフェイスが使用されます。AXI4-Lite へのブリッジはリクエスター インターフェイスを使用しないため、これらの高性能ポートを継続して使用できます。通常、ホストからの DW 要求は 1 つであるため、コンプリータ インターフェイスを介して AXI4-Lite へのブリッジを使用することを推奨します。このアプリケーション ノートで提供するパッケージ化された IP コアを利用することによって、ホストからの要求に素早く反応し、それらを AXI4 トランザクションに変換できます。

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. 『Virtex-7 FPGA Gen3 Integrated Block for PCI Express』(PG023)
2. 『Using the Memory Endpoint Test Driver (MET) with the Programmed Input/Output Example Design for PCI Express Endpoint Cores』(XAPP1022)

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
2014 年 1 月 21 日	1.0	初版

Notice of Disclaimer

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.

Automotive Applications Disclaimer

XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO:(I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY.CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.

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。