

QDMA Subsystem for PCI Express v3.0

製品ガイド

Vivado Design Suite

PG302 (v3.0) 2018 年 12 月 5 日

この資料は表記のバージョンの英語版を翻訳したもので、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。



目次

第 1 章: IP の概要.....	4
機能.....	4
IP の概要.....	5
第 2 章: 概要.....	6
用語解説.....	7
QDMA のアーキテクチャ.....	8
アプリケーション.....	22
ライセンスおよび注文情報.....	23
第 3 章: 製品仕様.....	24
規格.....	24
最小デバイス要件.....	24
QDMA 操作.....	25
ポートの説明.....	75
レジスタ空間.....	97
第 4 章: サブシステムを使用するデザイン.....	189
一般的なデザイン ガイドライン.....	189
クロッキング.....	190
第 5 章: デザイン フローの手順.....	191
サブシステムのカスタマイズおよび生成.....	191
サブシステムへの制約.....	203
シミュレーション.....	205
合成およびインプリメンテーション.....	208
第 6 章: サンプル デザイン.....	209
完了のある AXI4 メモリ マップドおよび AXI4-Stream のデフォルト サンプル デザイン.....	209
AXI メモリ マップド サンプル デザイン.....	210
完了のある AXI メモリ マップドのサンプル デザイン.....	211
完了のある AXI ストリームのサンプル デザイン.....	212
ディスクリプター バイパス入力/出力ループバックのあるサンプル デザイン.....	213
サンプル デザインのレジスタ.....	213
付録 A: アップグレード.....	223
v2.0 から v3.0 への変更点.....	223
DMA/Bridge Subsystem for PCI Express との比較.....	223

付録 B: デバッグ.....	224
ザイリンクス ウェブサイト.....	224
デバッグ ツール.....	225
ハードウェア デバッグ.....	225
付録 C: アプリケーション ソフトウェア開発.....	227
デバイス ドライバー.....	227
Linux DMA ソフトウェア アーキテクチャ (PF/VF).....	228
ドライバーの使用.....	229
リファレンス ソフトウェア ドライバーのフロー.....	230
付録 D: その他のリソースおよび法的通知.....	236
ザイリンクス リソース.....	236
Documentation Navigator およびデザイン ハブ.....	236
参考資料.....	236
改訂履歴.....	237
お読みください: 重要な法的通知.....	238

IP の概要

ザイリンクス QDMA Subsystem for PCI Express® (PCIe®) は、複数のザイリンクス C2H および H2C チャンネルを使用する DMA/Bridge Subsystem for PCI Express とは異なるマルチ キューのコンセプトを利用した、PCI Express® 3.x 統合ブロックと共に使用する高パフォーマンスの DMA をインプリメントします。

機能

- PCIe 統合ブロックは UltraScale+™ デバイス (高帯域幅メモリ (HBM) の Virtex® UltraScale+™ デバイスを含む) でサポートされています。
- 64、128、256、および 512 ビットのデータパスをサポート。
- X1、x2、x4、x8、x16 のリンク幅をサポート。
- Gen1、Gen2、Gen3 のリンク スピードをサポート。PCI4C ブロックの場合は Gen4。
- キューごとに AXI4 メモリ マップドおよび AXI4-Stream インターフェイスの両方をサポート。
- 2048 個のキュー セット
 - 2048 個の H2C ディスクリプター リング。
 - 2048 個の C2H ディスクリプター リング。
 - 2048 個の C2H。
- ポーリング モード (ステータス ディスクリプター ライトバック) および割り込みモードをサポート。
- 割り込み
 - 2048 個の MSI-X ベクター。
 - 各ファンクションに最高 8 個の MSI-X。より多くのベクターを 1 つのファンクションに割り当てることが可能。サポートが必要な場合は、ザイリンクスまでお問い合わせください。
 - 割り込みアグリゲーション。
- C2H ストリーム割り込みモデレーション。
- C2H ストリーム完了キュー エントリの結合。
- ユーザー ロジックを介したディスクリプターおよび DMA のカスタマイズ
 - カスタム ディスクリプターのフォーマットを使用可能。
 - トラフィック管理。
- 最高 4 個の物理ファンクション (PF) および最高 252 個の仮想ファンクション (VF) の SR-IOV をサポート。
 - 薄いハイパーバイザー モデル。

- QID 仮想化。
 - 権限が制限されているファンクションまたは物理ファンクションのみでコンテキストおよびレジスタをプログラム可能。
 - ファンクション レベル リセット (FLR) のサポート。
 - メールボックス。
- AMI-MM か AXI-ST かの選択肢など、キューごとにプログラム可能。

IP の概要

この LogiCORE IP について	
サブシステム コアの概要	
サポートされるデバイス ファミリ ¹	UltraScale+™
サポートされるユーザー インターフェイス	AXI4 メモリ マップ、AXI4-Stream、AXI4-Lite
リソース	Performance and Resource Use (ウェブ ページ)。
サブシステム	
デザイン ファイル	暗号化 System Verilog
サンプル デザイン	Verilog
テストベンチ	Verilog
制約ファイル	ザイリンクス ユーザー制約ファイル (UCF)
シミュレーション モデル	Verilog
サポートされるソフトウェア ドライバー	Linux、DPDK、および Windows ドライバー ²
テスト済みデザイン フロー ³	
デザイン入力	Vivado Design Suite
シミュレーション	サポートされるシミュレータは、『Vivado Design Suite ユーザーガイド: リリース ノート、インストールおよびライセンス』(UG973) を参照。
合成	Vivado 合成
サポート	
ザイリンクス サポート ウェブ ページで提供	

注記:

1. サポートされているデバイスの一覧は、Vivado IP カタログを参照してください。
2. Linux および DPDK ドライバーの詳細は、[ザイリンクス DMA IP ドライバー](#)を参照してください。Windows ドライバーの詳細は、[QDMA Windows ドライバー ラウンジ](#)を参照してください。
3. サポートされているツールのバージョンは、『Vivado Design Suite ユーザーガイド: リリース ノート、インストールおよびライセンス』(UG973) を参照してください。

概要

Queue Direct Memory Access (QDMA) サブシステムは、PCI Express (PCIe) ベースの DMA エンジンで、ハンド幅およびパケット カウントの高いデータ転送用に最適化されています。この QDMA は、UltraScale+ Integrated Block for PCI Express IP、拡張 DMA、および究極のパフォーマンスと柔軟性を可能にするブリッジ インフラストラクチャから構成されています。

QDMA Subsystem for PCIe には、幅広い設定や使用オプションがあり、メモリ マップド DMA やストリーム DMA、割り込みモードやポーリングなど、キューごとに選択可能なオプションが多数用意されています。また、このサブシステムには、複雑なトラフィック管理機能を提供する目的で、ユーザー ロジックを使用したディスクリプターおよび DMA のカスタマイズ オプションも数多く提供されています。

QDMA を使用したデータ転送は、ホストの OS から供給された命令 (ディスクリプター) に基づいて QDMA エンジンが操作することにより実行されます。このディスクリプターを使用して、QDMA はホストからカードへ (H2C)、またはカードからホストへ (C2H) の双方向にデータを移動させることができます。DMA トラフィックが AXI4 メモリ マップ インターフェイスに向かうのか、それとも AXI4-Stream インターフェイスに向かうのか、方向はキューごとに選択可能です。さらに、QDMA には、AXI4 マスター ポートおよび AXI4 スレーブ ポートの両方をインプリメントするオプションがあり、PCIe トラフィックが DMA エンジン完全にバイパスできるようになっています。使用可能なインターフェイスのリストは、[ポートの説明](#) を参照してください。

QDMA とほかの DMA の機能の主な違いは、キューの概念です。「キュー」は、ハイ パフォーマンス コンピューティング (HPC) インターコネクトのリモート直接メモリ アクセス (RDMA) の「キュー セット」から派生した概念です。これらのキューは、インターフェイス タイプ別に個別に設定可能で、さまざまなモードで機能します。1 つのキューに複数の DMA ディスクリプターをロードできるので、セットアップや継続的なアップデートを実行するときの各キューのオーバーヘッドは非常に低くなっています。キューをリソースとして複数の PCIe の物理ファンクション (PF) および仮想ファンクション (VF) に割り当てると、1 つの QDMA コアおよび PCI Express インターフェイスを、さまざまなマルチファンクションおよび仮想化されたアプリケーション空間で使用できるようになります。

QDMA Subsystem for PCIe は、ザイリンクス提供の QDMA リファレンス ドライバーと併用して実行可能なので、さまざまなタイプのアプリケーション空間に合わせて構築できます。

用語解説

次の表に、この資料でよく使用される略語とその意味を説明します。

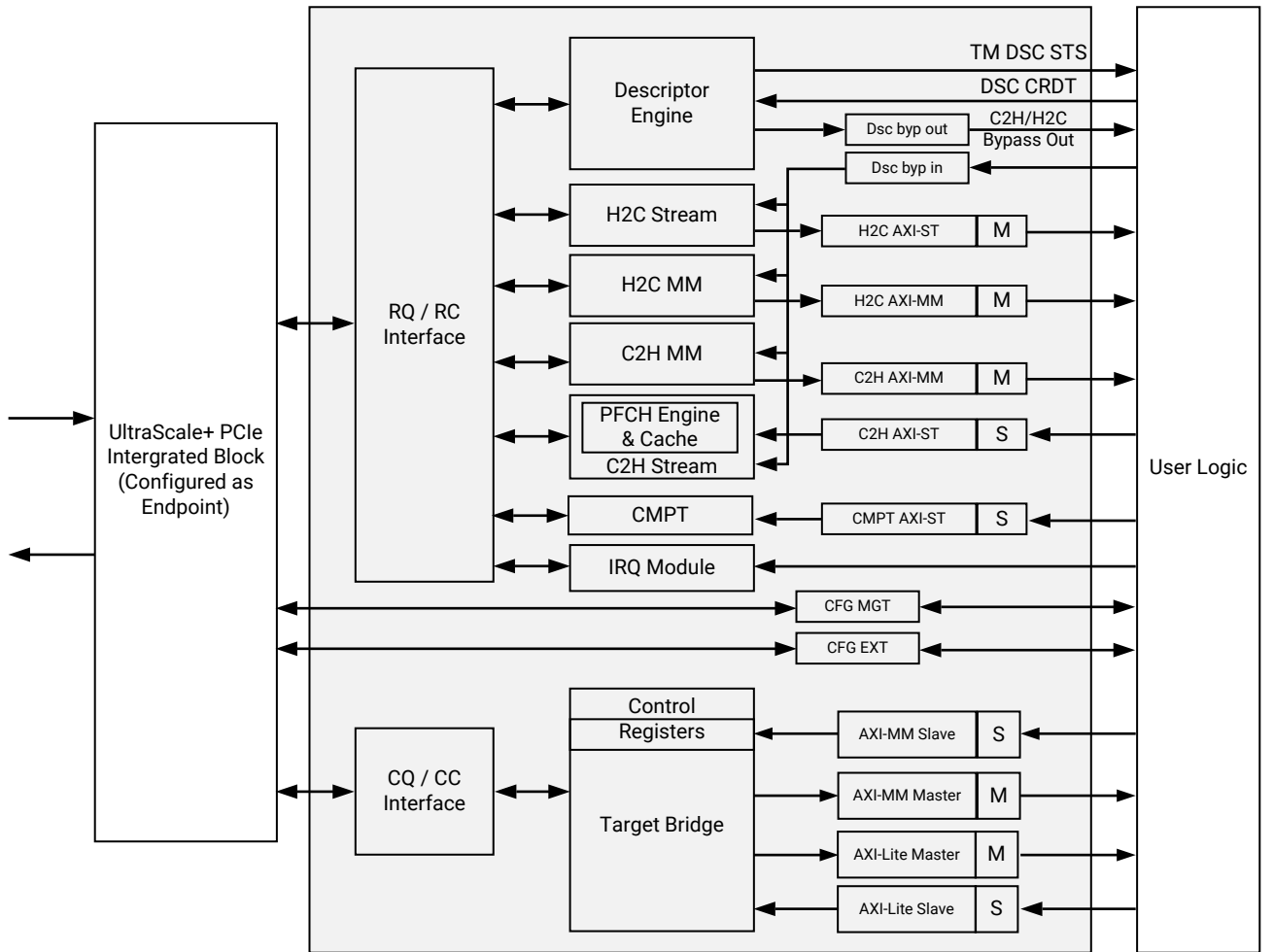
表 1: 用語集

略語	非略語
AXI-ST	AXI4-Stream
AXI-MM	AXI4 メモリ マップド
BADDR	ベース アドレス
C2H	カードからホスト
CC	コンプリーター完了
CIDX	コンシューマー インデックス ポインター
CMPT	完了
CQ	コンプリーター リクエスト
CRDT	クレジット
CSR	制御/ステータス レジスタ
CTXT	コンテキスト
DSC	ディスクリプター
FLR	ファンクション レベル リセット
H2C	ホストからカード
HW	ハードウェア
MM	メモリ マップド
OS	オペレーティング システム
PF	物理ファンクション
PFCH	プリフェッチ キャッシュ ブロック
PIDX	プロデューサー インデックス ポインター
QDMA	キュー直接メモリ アクセス
QID	キュー ID
RC	リクエスター完了
RDMA	リモート直接メモリ アクセス
RQ	リクエスター リクエスト
RXQ	受信キュー
SRIOV	シングル ルート入力/出力の仮想化
ST	ストリーム
SW	ソフトウェア
TM	トラフィック マネージャー
TXQ	送信キュー
VM	仮想マシン
VF/VFG	仮想ファンクション/仮想ファンクション グループ

QDMA のアーキテクチャ

次に、QDMA Subsystem for PCIe のブロック図を示します。

図 1: QDMA のアーキテクチャ



X20894-120318

DMA エンジン

ディスクリプター エンジン

H2C および C2H ディスクリプターは、内部モードまたはバイパス モードのいずれかでディスクリプター エンジンにフェッチされます。このエンジンではキューごとにコンテキストが管理され、このコンテキストによりソフトウェアの PIDX、CIDX、キューのベース アドレス (BADDR)、および各キューのキュー設定が管理されます。ディスクリプターのフェッチにはラウンドロビン アルゴリズムが使用されます。ディスクリプター エンジンには、H2C/C2H 用にそれぞれ別のバッファがあり、許容量を超えたディスクリプターはフェッチされないようになっています。また、未処理の DMA 読み出しは、一度に 1 キューにつき 1 つしか認められないため、MRRS に収まる数のディスクリプターしか処理できません。順不同の完了はディスクリプター エンジンにより並べ替えられるので、キューのディスクリプターは常に正しい順序になります。

ディスクリプター バイパスはキューごとにイネーブルにでき、フェッチされたディスクリプターは、直接 H2C や C2H エンジンに送られるのではなく、バッファリングされた後に、バイパス出力インターフェイスに送られます。内部モードでは、コンテキスト設定に基づいて、ディスクリプターが H2C MM、C2H MM、H2C ストリーム、または C2H ストリームのエンジンにそれぞれ送られます。

ディスクリプター エンジンでは、DMA 操作完了のステータス ディスクリプターも生成されます。C2H ストリーム モード以外のモードでは、このステータス ディスクリプターを使用して各 DMA 操作の完了を通知し、ディスクリプターをソフトウェアで再び使用できるようにし、関連バッファも解放します。これは、ステータス ディスクリプターの CIDX フィールドに示されます。



推奨: キューが割り込みアグリゲーションに関連付けられている場合、ザイリンクスでは、ステータス ディスクリプターをオフにし、割り込みアグリゲーション リングから DMA ステータスを受信するよう推奨します。割り込みアグリゲーション リングの詳細は、[割り込みアグリゲーション リング](#) を参照してください。

フェッチされるディスクリプターの数を制限する場合は (たとえば、ディスクリプターの格納に必要なバッファリング容量を制限する場合など)、キューごとにクレジットをイネーブルにしてスロットルさせることが可能です。ディスクリプターはクレジット上限までフェッチされるので、キューごとにフェッチされるディスクリプターの合計数はこのクレジット数に制限されます。クレジットは、ユーザー ロジックにより `dsc_crdt` インターフェイスを介して戻されます。また、クレジットの精度はディスクリプターのサイズで決まります。

トラフィック マネージャーでジョブの優先順位を付けやすくするため、PIDX アップデートの中のフェッチ可能なディスクリプター (インクリメンタルな PIDX 値) が、`tm_dsc_sts` インターフェイスを介してユーザー ロジックに送られます。このインターフェイスを使用すると、ディスクリプターのストレージの優先順位付けと最適化の両方を実行できるデザインをインプリメントできます。

H2C メモリ マップド エンジン

H2C メモリ マップド エンジンは、H2C AXI-MM インターフェイスを介してホスト メモリからカード メモリにデータを移動させます。このエンジンは、PCIe の最大読み出しリクエスト サイズ (MRRS)、および PCIe 読み出しが 4 KB の境界をまたいではいけないという要件に従って、ディスクリプターを複数の読み出しリクエストに分割し、PCIe の読み出しを生成します。読み出しリクエストの完了データを受信すると、H2C の AXI-MM インターフェイスで AXI 書き込みが生成されます。ソース アドレスとデスティネーション アドレスが揃っていない場合は、データが 4 KB の境界をまたがないようにするため、ハードウェアによりデータがシフトされ、AXI-MM で書き込みが分割されます。完了したディスクリプターはそれぞれ、ライトバックが必要なのか、割り込みが必要なのかを判断するためにチェックされます。

内部モードの場合は、ディスクリプター エンジンにより、メモリ マップド ディスクリプターが直接 H2C MM エンジンに送られます。また、ホストからカード メモリにデータを移動させるために、ユーザー ロジックからも H2C バイパス インターフェイスにディスクリプターを挿入できます。これで、同じキューに制御と DMA コマンドを混ぜたりすることが可能になります。制御情報が制御プロセッサに送信されると、DMA 操作が完了したことになります。

C2H メモリ マップド エンジン

C2H メモリ マップド エンジンは、C2H AXI-MM インターフェイスを介してカード メモリからホスト メモリにデータを移動させます。このエンジンは、ディスクリプターを 4 KB 境界に基づいて複数のリクエストに分割し、C2H AXI-MM バスでの AXI 読み出しを生成します。読み出しリクエストの完了データが AXI4 インターフェイスで受信されると、AXI 読み出しデータを書き込みのコンテンツとして使用して、PCIe 書き込みが生成されます。ソース アドレスと デスティネーション アドレスが揃っていない場合は、データが 4 KB 境界をまたがないようにし、また最大ペイロード サイズ (MPS) に従うため、ハードウェアによりデータがシフトされて PCIe での書き込みが分割されます。完了したディスクリプターはそれぞれ、ライトバックが必要なのか、割り込みが必要なのかを判断するためにチェックされます。

内部モードの場合は、ディスクリプター エンジンにより、メモリ マップド ディスクリプターが直接 C2H MM エンジンに送られます。また、H2C MM エンジンの場合と同様に、カードからホスト メモリにデータを移動させるために、ユーザー ロジックからも C2H バイパス インターフェイスにディスクリプターを挿入できます。

マルチファンクション コンフィギュレーションをサポートするには、ユーザー ロジックでカード メモリを仮想化しやすくするため、AXI-MM インターフェイス バスの `aruser` ビットに PCIe ファンクション番号が供給されます。パリティ バスも、エンド トゥ エンドのパリティをサポートするため、データおよびユーザー バスとは別に提供されています。

H2C ストリーム エンジン

H2C ストリーム エンジンはホストから H2C ストリーム インターフェイスにデータを移動させます。内部モードの場合はディスクリプターは H2C ストリーム エンジンに直接送られ、バイパス モードのキューの場合はディスクリプターは再フォーマットされてバイパス入力インターフェイスに送られます。エンジンは、DMA 読み出しを MRRS サイズに分割して完了用のスペースを確保し、H2C ストリーム データが順番にユーザー ロジックに送信されるようにするため、完了の順序を並べ替えます。

このエンジンには、最高 256 個までの DMA 読み出し、および最高 32 KB までのデータを格納するスペースがあります。DMA でフェッチされたデータは、AXI4 インターフェイス側で転送する最初のバイトに揃えられるので、どのディスクリプターのオフセットおよび長さもランダムになります。すべてのディスクリプターの長さ合計 64 KB 未満である必要があります。

キューが内部モードの場合、各ディスクリプターで、H2C AXI-ST インターフェイスに転送される 1 つの AXI4-Stream パケットが定義されます。複数のディスクリプターがストラドルしているパケットは、各キューのストレージが足りないため使用できません。ただし、複数のディスクリプターを含むパケットは、ディスクリプター バイパス モードを使用してインプリメントできます。このモードでは、ユーザー ロジックにパケットを形成するのに十分なディスクリプターがあれば、H2C DMA エンジンを開始できます。複数のディスクリプターがストラドルしているパケットを、バイパス インターフェイスを介してほかの H2C ST パケット ディスクリプターと共に送信し、これらのディスクリプターがインターリーブされていないかどうかを確認できると、DMA エンジンが開始します。また、バイパス インターフェイスでは、ユーザー ロジックでステータス ディスクリプターの生成を制御できます。

C2H ストリーム エンジン

C2H ストリーム エンジンは、ユーザー ロジックからデータを受信し、キューの C2H ディスクリプターから提供されたホスト メモリ アドレスに書き込みます。

このエンジンには、C2H ストリーミング DMA、ディスクリプター プリフェッチ キャッシュ (PFCH) および C2H-ST DMA 書き込みエンジンを実行するための主要ブロックが 2 つあります。PFCH には、そのファンクションのパフォーマンスと、それをプログラムするソフトウェアのパフォーマンスを向上するため、キューごとにコンテキストがあります。

PFCH には、キューごとに、単純バイパス モード、内部キャッシュ モード、キャッシュド バイパス モードという 3 つの主なモードがあります。

- 単純バイパス モードでは、C2H ストリーム エンジンでキューの情報は確認されず、ユーザー ロジックでディスクリプターの受信方法を定義できます。単純バイパス インターフェイスでのパケットおよび関連ディスクリプターの送信は、ユーザー ロジックにより実行されます。バイパス インターフェイスおよび C2H ストリーム インターフェイスで、キューによりフェッチされたディスクリプターの順序は、バイパス モードのすべてのキューで維持する必要があります。
- 内部キャッシュ モードおよびキャッシュド バイパス モードでは、PFCH モジュールに最高 512 個までのディスクリプターを格納するストレージがあり、これらのディスクリプターは最高 64 個までの異なるキューで使用可能です。この 2 つのモードでは、C2H ストリーム エンジンにより、パイプラインにどの受信パッケージがあるかによって、オンディマンドで C2H ディスクリプター キュー クレジットが管理され、フェッチされるディスクリプターが制御されます。また、プリフェッチ モードは、キューごとにイネーブルにでき、パッケージ データの前にディスクリプターが使用可能になるように、機会を見てディスクリプターがプリフェッチされます。ステータスはプリフェッチ コンテキストで確認できます。関連ディスクリプターがフェッチされるまで待機せず、パケット データを PCIe 統合ブロックにほぼ即時に転送できるようにすることで、レイテンシを大幅に削減できます。バッファのサイズはキュー (PFCH コンテキスト) で固定されており、C2H ストリーム エンジンでパケットを最大 7 個までのディスクリプターにスキップできます。また、キャッシュド バイパス モードでは、ディスクリプターはアドレス変換などの処理用にユーザー ロジックにバイパスされ、バイパス入力インターフェイス経由で送り返されます。このモードでは、ディスクリプターが特定の順序に並んでいることや C2H ストリーム パケット インターフェイスを使用することが想定されていないので、プリフェッチはパケットおよびディスクリプターと一致します。

完了エンジン

完了エンジンは、完了キューに書き込むために使用されます。このエンジンは、AXI-MM およびストリーム DMA エンジンと併用可能ですが、C2H ストリーム DMA エンジンは、完了エンジン用に設計されています。また、完了エンジンは、完了リングに即値データを渡す目的でも使用されます。完了リングには最高 64B までの完了を書き込むことができます。DMA エンジンと併用する場合、ドライバーは完了エンジンを使用して各パケットで何バイトのデータが転送されたのかを判断します。これにより、ドライバーはディスクリプターを再要求できます。

CMPT コンテキストは完了エンジンで管理されます。このコンテキストはドライバーによってプログラムされ、キューごとに管理されます。完了コンテキストには、完了リングのベース アドレス、PIDX、CIDX など、完了エンジンのさまざまな情報が格納されます。また、このエンジンは、完了コンテキストのフィールドを設定して制御できます。

このエンジンは、ソフトウェアのニーズに合わせて、割り込みまたは完了ステータス アップデート、またはその両方を生成するため、キューごとに設定することも可能です。複数のキューの割り込みが割り込みアグリゲーション リングにアグリゲートされる場合は、ステータス ディスクリプターは割り込みアグリゲーション リングでも使用できません。

CMPT エンジンには、PCIe の効率を上げる目的で、最高 64 エントリまでのキャッシュがあり、複数の小さな CMPT 書き込みを 64 KB の書き込みに結合させることができます。完了はいつでも最高 64 キューまで同時に結合させることが可能ですが、これを超えると、追加のキューに CMPT エントリを書き込む必要があると、キャッシュから 1 番古いキューが削除されます。この目的で使用されるキャッシュの深さは、8、16、32、または 64 に設定できます。

ブリッジ インターフェイス

ブリッジ マスター AXI メモリ マップド インターフェイス

ブリッジ マスター AXI メモリ マップド インターフェイスは、ホストから AXI メモリ マップド空間への高帯域幅アクセスに使用され、未処理の AXI 読み出しおよび書き込みが最高 32 個までサポートされています。このブリッジ マスター AXI メモリ マップド インターフェイスには、物理ファンクション (PF) または仮想ファンクション (VF) の 1 つまたは複数の PCIe BAR をマップできます。このオプションは、IP を設定するときに選択する必要があります。ファンクション ID、BAR ID、VF グループおよび VF グループ オフセットは、AXI メモリ マップド インターフェイスの `aruser` および `awuser` の一部として使用可能で、各メモリ アクセスのソースをユーザー ロジックで特定できるようになっています。`m_axib_awuser/m_axib_aruser` ユーザー ビットは次のようにマップされます。

- `m_axib_awuser/m_axib_aruser[29:0]` は 30 ビットあります。
- その 30 ビットの内訳は、次のとおりです。
 - `m_axib_awuser/m_axib_aruser[7:0]` = ファンクション番号
 - `m_axib_awuser/m_axib_aruser[15:8]` = 予約済み
 - `m_axib_awuser/m_axib_aruser[18:16]` = BAR ID
 - `m_axib_awuser/m_axib_aruser[26:19]` = VF グループ (VFG) オフセット
 - `m_axib_awuser/m_axib_aruser[28:27]` = VF グループ (VFG) ID

仮想ファンクショングループ (VFG) は VF グループ番号で、対応する VF に関連付けられている PF 番号と同じです。VFG_OFFSET は特定の PF に関連付けられている VF 番号で、各 PF の FIRST_VF_OFFSET ではありません。

たとえば、PF0 と PF1 の両方に 8 個の VF があり、PF0 と PF1 の FIRST_VF_OFFSET がそれぞれ 4 と 11 の場合、VFG および VFG_OFFSET のマップは次のようになります。

表 2: AXI-MM インターフェイス VFG

ファンクション番号	PF 番号	VFG	VFG_OFFSET
0	0	0	0
1	1	0	0
4	0	0	0 (PF0 の FIRST_VF_OFFSET は 4 なので、PF0 の最初の VF は FN_NUM = 4 で開始し、VFG_OFFSET = 0 はこれが PF0 の最初の VF となることを示す)
5	0	0	1 (VFG_OFFSET = 1 はこれが PF0 の 2 番目の VF となることを示す)
...
12	1	1	0 (VFG = 1 はこの VF が PF1 に関連付けられていることを示す)
13	1	1	1

アクセスを開始した各ホストは、PCIe を AXI BAR に変換して 64 ビットの AXI アドレス空間にマップできます。

すべてのファンクションで同じ AXI マスター アドレス空間を共有しているので、異なるファンクションからのリクエストは AXI マスター側の個別のアドレス空間にマップする必要があります。PCIe から AXI への変換ベクターの使用方法は次のようになります。ただし、同じ PF に属しているすべての VF が同じ PCIe から AXI への変換ベクターを共有している点に注意してください。したがって、各 VF の AXI アドレス空間は連結されています。特定の VF の AXI の実際の開始アドレスを計算するには、VFG_OFFSET を使用します。

まとめると、`m_axib_awaddr` は次のように計算されます。

- PF の場合: $m_axib_awaddr = pcie2axi_vec + axib_offset$
- VF の場合: $m_axib_awaddr = pcie2axi_vec + (VFG_OFFSET + 1) * vf_bar_size + axib_offset$

`pcie2axi_vec` は PCIe から AXI BAR への変換 (IP 設定中に指定可能)。

`axib_offset` は、リクエストされたターゲット空間のアドレス オフセットです。

ブリッジ マスター AXI4-Lite インターフェイス

物理ファンクション (PF) または仮想ファンクション (VF) の 1 つ以上の PCIe BAR を AXI4-Lite マスター インターフェイスにマップできます。このオプションは、IP を設定するときに選択する必要があります。ファンクション ID、BAR ID (BAR ヒット)、VF グループおよび VF グループ オフセットは、AXI4-Lite インターフェイスの `aruser` および `awuser` の一部として使用可能で、メモリ アクセスのソースをユーザー ロジックで特定できます。

`m_axil_awuser/m_axil_aruser` ユーザー ビットは次のようにマップされます。

- `m_axil_awuser/m_axil_aruser[29:0]` は 30 ビットあります。
- その 30 ビットの内訳は、次のとおりです。
 - `m_axil_awuser/m_axil_aruser[7:0]` = ファンクション番号
 - `m_axil_awuser/m_axil_aruser[15:8]` = 予約済み
 - `m_axil_awuser/m_axil_aruser[18:16]` = BAR ID
 - `m_axil_awuser/m_axil_aruser[26:19]` = VF グループ (VFG) オフセット
 - `m_axil_awuser/m_axil_aruser[28:27]` = VF グループ (VFG) ID

仮想ファンクショングループ (VFG) は VF グループ番号で、対応する VF に関連付けられている PF 番号と同じです。VFG_OFFSET は特定の PF に関連付けられている VF 番号で、各 PF の FIRST_VF_OFFSET ではありません。

たとえば、PF0 と PF1 の両方に 8 個の VF があり、PF0 と PF1 の FIRST_VF_OFFSET がそれぞれ 4 と 11 の場合、VFG および VFG_OFFSET のマップは次のようになります。

表 3: AXI4-Lite インターフェイス VFG

ファンクション番号	PF 番号	VFG	VFG_OFFSET
0	0	0	0
1	1	0	0
4	0	0	0 (PF0 の FIRST_VF_OFFSET は 4 なので、PF0 の最初の VF は FN_NUM = 4 で開始し、VFG_OFFSET = 0 はこれが PF0 の最初の VF となることを示す)
5	0	0	1 (VFG_OFFSET = 1 はこれが PF0 の 2 番目の VF となることを示す)
...
12	1	1	0 (VFG = 1 はこの VF が PF1 に関連付けられていることを示す)
13	1	1	1

アクセスを開始した各ホストは、PCIe を AXI BAR に変換して 64 ビットの AXI アドレス空間にマップできます。

すべてのファンクションで同じ AXI4 マスター アドレス空間を共有しているので、異なるファンクションからのリクエストは AXI マスター側の個別のアドレス空間にマップする必要があります。PCIe から AXI への変換ベクターの使用方法は次のようになります。ただし、同じ PF に属しているすべての VF が同じ PCIe から AXI への変換ベクターを共有している点に注意してください。つまり、各 VF の AXI アドレス空間は連結されています。特定の VF の AXI の実際の開始アドレスを計算するには、VFG_OFFSET を使用します。

まとめると、`m_axil_awaddr` は次のように計算されます。

- PF の場合: `m_axil_awaddr = pcie2axi_vec + axil_offset`

- VF の場合: $m_axil_awaddr = pcie2axi_vec + (VFG_OFFSET + 1) * vf_bar_size + axil_offset$

`pcie2axi_vec` は PCIe から AXI BAR への変換 (IP 設定中に指定可能)。

`axil_offset` は、リクエストされたターゲット空間のアドレス オフセットです。

アクセスを開始した各ホストは、64 ビットの AXI アドレス空間にマップできます。このインターフェイスでは、未処理の読み出しが 1 つ、未処理の書き込みが 1 つサポートされています。

拡張 ROM BAR も IP を設定するときに AXI4-Lite インターフェイスにマップ可能です。

PCIe から AXI への BAR

各物理ファンクションに対し、PCIe コンフィギュレーション空間は、6 つの 32 ビット メモリ BAR および 1 つの 32 ビット EXPROM BAR で構成されています。SR-IOV がイネーブルの場合、各仮想ファンクションに対し、追加でさらに 6 つの 32 ビット BAR がイネーブルになります。これらの BAR を使用して、AXI4 メモリ マップド空間の機能、インターフェイス配線、および AXI4 リクエスト属性設定用に、アドレスを変換します。BAR のペアはどれも 1 つの 64 ビット BAR として設定できます。各 BAR は、QDMA レジスタ空間、ブリッジ AXI4-Lite マスター インターフェイス、またはブリッジ AXI4 メモリ マップド マスター インターフェイスへリクエストを転送するように設定可能です。プログラミング例は、『AXI Bridge for PCI Express Gen3 Subsystem 製品ガイド』(PG194) のアドレス変換のセクション (例 3) を参照してください。

メモリ タイプのリクエスト

メモリ タイプは、属性 `attr_dma_pciebar2axibar_*_cache_pf*` を使用して各 PCIe BAR に設定できます。

- AxCache[0] は、変更可能な場合は 1 に、変更不可能な場合は 0 に設定します。
- AxCache[1] は、キャッシュ可能な場合は 1 に、キャッシュ不可能な場合は 0 に設定します。

ブリッジ スレーブ AXI メモリ マップド インターフェイス

ブリッジ スレーブ AXI メモリ マップド インターフェイスは、ユーザー ロジックおよびホスト間の高帯域幅のメモリ転送に使用され、AXI から PCIe への変換は、AXI から PCIe への BAR を使用してサポートされています。このインターフェイスは、PCIe MPS および 4 KB の境界に関する要件に従うために、必要に応じてリクエストを分割します。未処理の読み出しおよび書き込みリクエストは、32 個までサポートされています。

ブリッジ スレーブ AXI4-Lite インターフェイス

AXI4-Lite スレーブは、AXI ブリッジおよび QDMA 内部レジスタにアクセスするために使用されます。上位 4 つのアドレスビットは、アクセス先が QDMA レジスタなのか、ブリッジ レジスタなのかを示しています。

- `s_axil_awaddr[28] = 1'b1` の場合、書き込みアクセス先は QDMA レジスタです。
- `s_axil_awaddr[28] = 1'b0` の場合、書き込みアクセス先はブリッジ レジスタです。ブリッジ レジスタにアクセスする場合、アドレス `0x000` から `0xDEF` までのアクセスは PCIe コア コンフィギュレーション空間アクセスに変更され、アドレス `0xE00` 以降がブリッジ レジスタ アクセスになります。
- `s_axil_araddr[28] = 1'b1` の場合、読み出しアクセス先は QDMA レジスタです。
- `s_axil_araddr[28] = 1'b0` の場合、読み出しアクセス先はブリッジ レジスタです。ブリッジ レジスタにアクセスする場合、アドレス `0x000` から `0xDEF` までのアクセスは PCIe コア コンフィギュレーション空間アクセスに変更され、`0xE00` 以降のアドレスがブリッジ レジスタ アクセスになります。

QDMA レジスタは VF および PF 用に仮想化されます。たとえば、VF と PF は、アドレス空間の別の箇所にアクセスでき、それぞれのキューにアクセスします。ファンクション別のアクセスを可能にするには、書き込みアクセスの場合は `s_axil_awuser[7:0]` で、読み出しアクセスの場合は `s_axil_aruser[7:0]` で、ファンクション ID がユーザー ロジックから提供されるので、QDMA が正しく内部アクセスにできるようになります。未処理リクエストは、読み出しで 1 つ、書き込みで 1 つが AXI4-Lite スレーブ インターフェイスでサポートされています。

AXI4-Lite スレーブ インターフェイスは、ブリッジ レジスタを使用して、ベンダー定義のメッセージを生成するにも使用されます。ベンダー定義メッセージについては、[VDM](#) を参照してください。

AXI から PCIe への BAR

ブリッジ スレーブ インターフェイスには、32 ビットまたは 64 ビットに設定できる 6 つの BAR があります。これらの BAR は、アドレスを AXI アドレス空間から PCIe アドレス空間に変換します。アドレス変換は、[Aperture Base Address]、[Aperture High Address]、および [AXI to PCIe Translation] の Vivado IP カスタマイズ設定を使用して、各 AXI BAR に対して設定できます。

プログラミングの例は、『AXI Bridge for PCI Express Gen3 Subsystem 製品ガイド』 ([PG194](#)) のアドレス変換のセクション (例 4) を参照してください。

割り込みモジュール

IRQ モジュールは、さまざまなソースから割り込みを PCIe® 統合ブロック コアのインターフェイスにアグリゲートします。割り込みのソースは、キューベースの割り込み、ユーザー割り込み、エラー割り込みです。

キューベースの割り込みおよびユーザー割り込みは、PF および VF で使用可能ですが、エラー割り込みは PF のみで使用可能です。SRIOV がイネーブルになっていない場合は、MSI-X、MSI 割り込みのいずれか、または両方を使用する選択肢があります。SRIOV がイネーブルになっている場合は、MSI-X 割り込みのみがすべてのファンクションでサポートされます。

MSI-X または MSI 割り込みのサポートは属性を指定して追加できます。ホストシステム (ルート コンプレックス) により、ハードウェアでサポートされている割り込みタイプの 1 つまたはすべてがイネーブルになります。MSI-X がイネーブルになっている場合は、MSI より優先されます。

UltraScale+™ デバイスの PCIe 統合ブロック コアでは、最高 8 個までの割り込みが提供されます。1 つの PCIe ファンクションで多くのキューを使用し、それぞれのキューに割り込みを持たせるため、QDMA Subsystem for PCIe では複数のキューからの割り込みを 1 つの割り込みベクターにまとめる新しい方法が使用されます。この方法では、理論的には 2048 個のキューすべてを 1 つの割り込みベクターにマップできます。QDMA では 256 個の割り込みアグリゲーション リングが提供されており、256 個のファンクションに柔軟に割り当てることができます。

PCIe ブロック インターフェイス

PCIe CQ/CC

PCIe CQ/CC モジュールはリモートの PCIe エージェントから TLP リクエストを受信し、処理します。UltraScale+™ Integrated Block for PCIe IP へのこのインターフェイスは、アドレス アライメント モードで動作します。TLP リクエストを転送すべきかどうかの判断には、Integrated Block for PCIe IP からの BAR 情報がモジュールで使用されます。これらのリクエストの転送先には次のものが考えられます。

- 内部コンフィギュレーション モジュール
- AXI4 メモリ マップド ブリッジ マスター インターフェイス
- AXI4-Lite ブリッジ マスター インターフェイス

ノンポストッド リクエストは、デスティネーションから完了を受信するようになっていて、この完了はリモートの PCIe エージェントに転送されます。詳細は、『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』(PG213: [英語版](#)、[日本語版](#))を参照してください。

PCIe RQ/RC

PCIe RQ/RC インターフェイスの役割は、RQ バスで PCIe TLP を生成し、RC バスから PCIe 完了 TLP を処理することです。UltraScale+™ Integrated Block for PCIe® コアへのこのインターフェイスは、DWord アライメント モードで操作します。ストラドルも、512 ビット インターフェイスを使用してイネーブルにする必要があります。ストラドルはサポートされていますが、『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』(PG213: [英語版](#)、[日本語版](#))で定義されている RQ ストラドルされたトランザクションの組み合わせがすべてインプリメントされるとは限りません。

PCIe コンフィギュレーション

未処理のノンポストッド トランザクションがスロットルされる要因はいくつかあります。未処理ノンポストッド トランザクションは、ポストッド リクエストの HOL (Head of Line) ブロッキングを防ぐため、PCIe® 統合ブロックからのフロー制御情報に基づいてスロットルされます。Vivado® 統合設計環境で IP をカスタマイズするときに、PCIe® 有限完了クレジットをイネーブルにできます。このオプションは、デフォルトではイネーブルになっていません。イネーブルになっていない場合は、PCIe 受信 FIFO 空間に基づいて、DMA がノンポストッド トランザクションを計測します。

キューの一般的なデザイン

QDMA Subsystem for PCIe のマルチキュー DMA エンジンでは、ユーザー ロジックでの RNIC を可能にするため、RDMA モデル キューのペアが使用されます。各キュー セットは、ホストからカード (H2C)、カードからホスト (C2H)、および C2H ストリーム完了 (CMPT) から構成されています。各キューの要素はディスクリプターです。

H2C および C2H は常にドライバー/ソフトウェアによって書き込まれ、またハードウェアは常にこれらのキューから読み出します。H2C は、ホストから DMA 読み出し操作のディスクリプターを出力し、C2H は、ホストへ DMA 書き込み操作のディスクリプターを出力します。

内部モードでは、ギャザー ディスクリプターと呼ばれる H2C ディスクリプターによってアドレスおよび長さ情報が伝えられます。これらのディスクリプターでは 32 ビットのメタ データがサポートされており、各ディスクリプターと共にソフトウェアからハードウェアに渡されます。ディスクリプターは、コンテキスト設定に基づいて、メモリ マップド (ホスト アドレス、カード アドレス、および DMA 転送の長さを送信) か、またはストリーミング (ホスト アドレスおよび DMA 転送の長さのみを送信) になります。ディスクリプター フォーマットは、ディスクリプター バイパスを介して定義でき、ソフトウェアによって即値データまたは追加メタ データがパケットと共に転送されます。

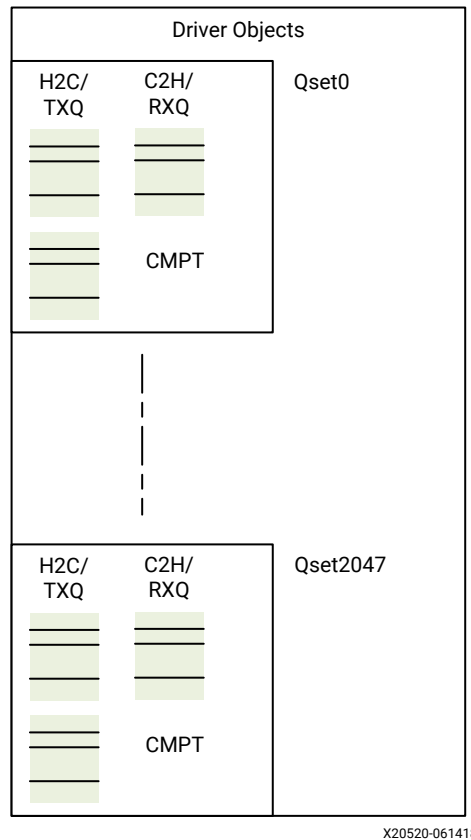
C2H キュー メモリ マップド ディスクリプターには、カード アドレス、ホスト アドレス、および長さが含まれています。ストリーミング内部キャッシュド モードの場合は、ホスト アドレスのみが送信されます。ディスクリプターのバッファ サイズは、ドライバーによってプログラムされ、キュー全体で固定されます。各ディスクリプターに関連付けられている実際に転送されるデータをバッファ サイズのフル サイズにする必要はありません。

ソフトウェアによってプロデューサー インデックスがハードウェアに書き込まれると、H2C および C2H キューの有効なディスクリプターがあることが知らされます。ステータス ディスクリプターは、C2H ストリーム リングの場合を除き、ディスクリプター リングの最終エントリです。ディスクリプターを再度使用し、ホストのバッファを解放するタイミングがドライバーにわかるように、ステータス ディスクリプターにはハードウェアのコンシューマー インデックス (CIDX) が含まれます。

C2H ストリーム モードの場合、CMPT キュー エントリを確認すると C2H ディスクリプターを再度使用できるかどうかわかります。通常、これは C2H パケットごとに 1 エントリあり、1 つ以上の C2H ディスクリプターが消費されたことを示します。CMPT キュー エントリには、消費されたディスクリプターをすべてソフトウェアで使用するのに十分な情報が含まれています。外部ロジックを使用して、ホストへのほかの完了や情報も含めることができます。

ハードウェアによってリングに書き込まれた CMPT エントリは、ディスクリプターのカラー ビットまたは CMPT リングの最後にあるステータス ディスクリプターを使用して、ドライバーで検出できます。各 CMPT エントリは、C2H ストリーム パケットのメタ データを送信するだけでなく、ユーザー アプリケーションのカスタム完了や即時通知を送信することもできます。

図 2: キュー リング アーキテクチャ



ソフトウェアで 16 個の異なるリング サイズをプログラムできます。各キューのリング サイズは、コンテキスト プログラミングから選択できます。最終キュー エントリはディスクリプター ステータスなので、使用可能なエントリの数はキュー サイズから 1 を引いた数になります。

たとえば、キュー サイズが 8 で、0 から 7 のエントリ インデックスが含まれている場合、最終エントリ (インデックス 7) はステータス用に予約されています。このインデックスは、PIDX アップデートには使用されません。PIDX アップデートが CIDX と等しくなることはありません。このケースだと、CIDX が 0 の場合、最大 PIDX アップデートは 6 になります。

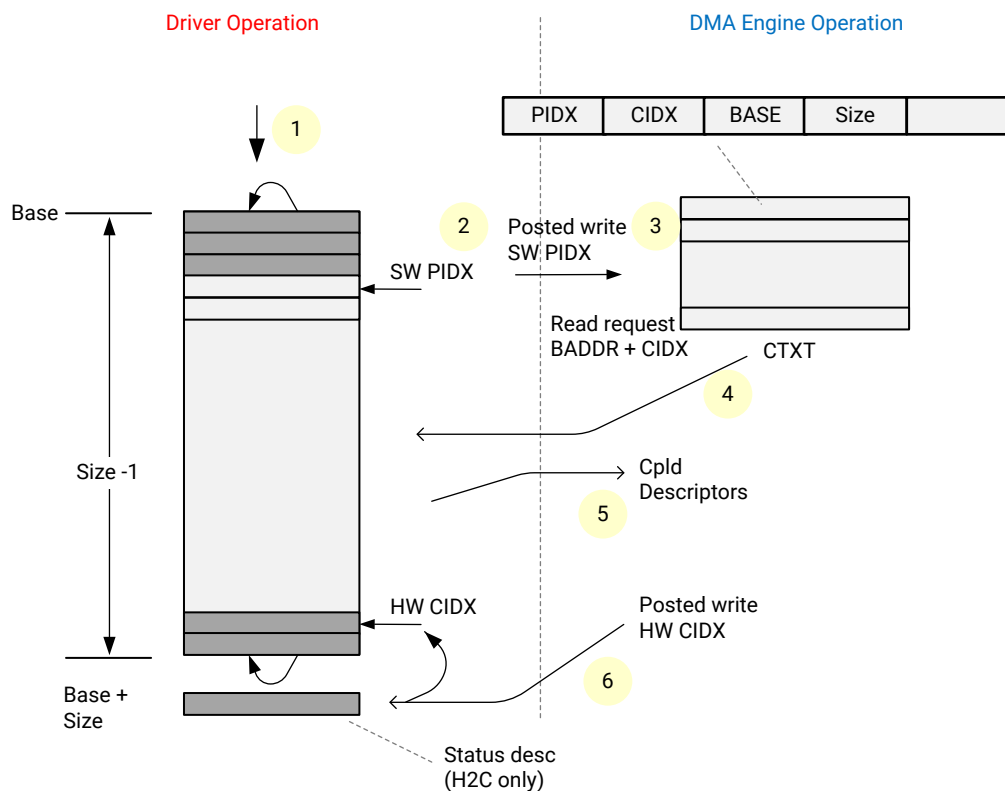
上記の例では、トラフィックが既に開始していて、CIDX が 4 の場合、最大 PIDX アップデートは 3 になります。

H2C および C2H のキュー

H2C/C2H のキューは、ホスト メモリにある環状リングです。どちらのタイプのキューも、プロデューサーはソフトウェア、コンシューマーはディスクリプター エンジンです。ソフトウェアでは、読み込まれていないディスクリプターを上書きするのを避けるため、プロデューサー インデックス (PIDX) およびハードウェア コンシューマー インデックスのコピー (HW CIDX) が管理されます。ディスクリプター エンジンでは、書き込まれていないディスクリプターが上書きされないようにするため、コンシューマー インデックス (CIDX) および SW PIDX のコピーが管理されます。キューの最終エントリはステータス ディスクリプター専用で、エンジンにより HW CIDX およびほかのステータスが書き込まれます。

エンジンでは、ローカル メモリにある H2C および C2H コンテキストがそれぞれ 2048 個まで管理されます。コンテキストには、ベース アドレス (BADDR)、SW PIDX、CIDX、長さといったキューのプロパティが格納されます。

図 3: 単純な H2C および C2H のキュー



X20895-061418

上の図は、H2C および C2H のフェッチ操作を示しています。

1. H2C の場合、ドライバーにより、ホスト バッファにペイロードが書き込まれ、そのペイロード バッファ情報を持った H2C ディスクリプターが形成され、PIDX ロケーションの H2C キューへ配置されます。C2H の場合は、ハードウェアがパケットを DMA に書き込めるよう、ドライバーで空きのあるバッファを使用してディスクリプターが形成されます。
2. ソフトウェアにより、現在の PIDX 値に関連付けられているキュー ID (QID) のディスクリプター エンジンにある PIDX レジスタにポストド ライトが送信されます。
3. PIDX アップデートが受信されると、エンジンにより、アドレス オフセットおよびファンクション ID に基づいたポインター アップデートの絶対 QID が計算されます。この QID を使用して、エンジンにより、QDMA Subsystem for PCIe に関連付けられているメモリから絶対 QID のコンテキストがフェッチされます。

4. このコンテキストに基づいてフェッチ可能なディスクリプターの数が決まります。ベース アドレス (BADDR)、CIDX、ディスクリプター サイズを使用して、ディスクリプター アドレスが計算され、DMA 読み出しリクエストが出力されます。
5. ディスクリプター エンジンでホストメモリから読み出し完了が受信されると、ディスクリプターが内部モードの H2C エンジンまたは C2H エンジンに送られます。バイパスする場合は、ディスクリプターが関連付けられているディスクリプター バイパス出力インターフェイスに送られます。
6. メモリ マップドまたは H2C ストリームのキューが内部モードにプログラムされている場合は、フェッチされたディスクリプターが完全に処理されてから、CIDX 値がステータス ディスクリプターに書き込まれます。キューがバイパス モードにプログラムされている場合は、ユーザー ロジックにより、バイパス入力インターフェイスを介してライト バックが制御されます。ステータス ディスクリプターはコンテキスト設定に基づいて制御されます。C2H ストリーム キューでは常に完了に CMPT リングが使用されます。

C2H の場合、フェッチ操作は CMPT リングを介して、内部的に実行されます。

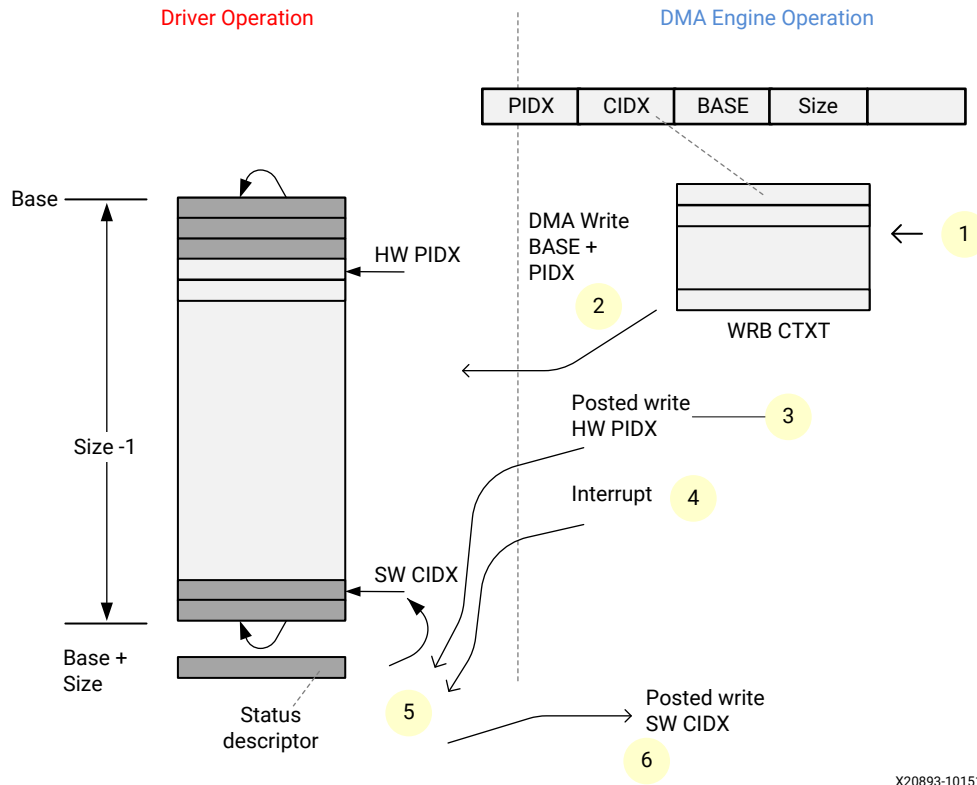
注記: C2H はプル モードで操作し、H2C はプルまたはプッシュ モードで操作可能です。

完了キュー

完了 (CMPT) キューは、ホスト メモリにある環状リングです。コンシューマーはソフトウェアで、プロデューサーは CMPT エンジンです。ソフトウェアでは、書き込まれていない完了を読み出すのを避けるため、プロデューサー インデックス (PIDX) およびハードウェア コンシューマー インデックスのコピー (HW CIDX) が管理されます。CMPT エンジンでは、読み出されていない完了を上書きしないようにするため、PIDX およびソフトウェア コンシューマー インデックスのコピー (SW CIDX) が管理されます。キューの最終エントリはステータス ディスクリプター専用で、ここにエンジンにより HW PIDX およびほかのステータスが書き込まれます。

エンジンでは、ローカル メモリにある合計 2048 個のコンテキストが管理されます。コンテキストには、ベース アドレス、SW CIDX、PIDX、深さなど、キューのプロパティが格納されます。

図 4: 単純な CMPT キューのフロー



X20893-101518

C2H ストリームは、ホストへの完了用に CMPT キューを使用するはずですが、ほかのタイプの完了用、またはホストドライバーへのメッセージ送信目的で使用される可能性もあります。CMPT を介したメッセージは、対応する C2H ストリーム パケット DMA をバイパスしません。

DMA CMPT キュー操作の単純フローは、上の図の番号の順番で進みます。

1. CMPT エンジンで完了メッセージが CMPT インターフェイスを介して受信されますが、完了メッセージの QID は C2H ストリーム インターフェイスから受信されます。エンジンにより、CMPT コンテキスト RAM の QID インデックスが読み出されます。
2. DMA により、アドレス $BASE + PIDX$ へ CMPT エントリが書き込まれます。
3. すべての完了が満たされると、オプションで、PIDX が CMPT キューのステータス ディスクリプターにカラービットと共に書き込まれます。
4. 割り込みモードがイネーブルになっている場合、割り込みモジュールの割り込みイベント メッセージが生成されます。
5. ソフトウェアはポーリング モードまたは割り込みモードになっています。いずれにしても、カラービットを一致させるか、ステータス ディスクリプターの PIDX 値を現在の SW CIDX 値と比較して、ソフトウェアで新しい CMPT エントリが特定されます。
6. ソフトウェアでキューの CIDX がアップデートされます。これにより、ハードウェアでディスクリプターを再利用できるようになります。ソフトウェアで CMPT の処理が終了したら、ポーリングを停止する前または割り込みハンドラーを離れる前に、ソフトウェアで関連キューの CIDX アップデート レジスタへの書き込みが実行されます。

SR-IOV のサポート

QDMA Subsystem for PCIe は Single Root I/O Virtualization (SR-IOV) をサポートするためのオプション機能を提供します。PCI-SIG® Single Root I/O Virtualization および Sharing (SR-IOV) の仕様 (PCI-SIG 仕様 (www.pcisig.com/specifications) を参照) は、データパス トランザクションに VMM を関与させない (バイパスさせる) 方法を標準化し、1 つの PCI Express® エンドポイントが複数の PCI Express エンドポイントに分かれているように見えるようにします。SR-IOV ではファンクションが次のように分類されています。

- 物理ファンクション (PF): SR-IOV 機能を含むフル PCIe® ファンクション。
- 仮想ファンクション (VF): ベース アドレス レジスタ (BAR) のあるコンフィギュレーション空間などの PCIe ファンクション。ただし、フル コンフィギュレーション リソースはなく、PF コンフィギュレーションにより制御されます。VF の主な役割はデータ転送です。

PCIe 定義のコンフィギュレーション空間以外に、QDMA Subsystem for PCI Express は、キューのポインター アップデートや割り込みなど、データパス操作を仮想化します。それ以外の管理機能およびコンフィギュレーション機能 (またはスローパス) は、物理ファンクションのドライバーに任されています。ドライバーに適切な権限がない場合は、QDMA Subsystem for PCI Express の一部として提供されているメールボックス インターフェイスを経由して権限のあるドライバーと通信する必要があります。

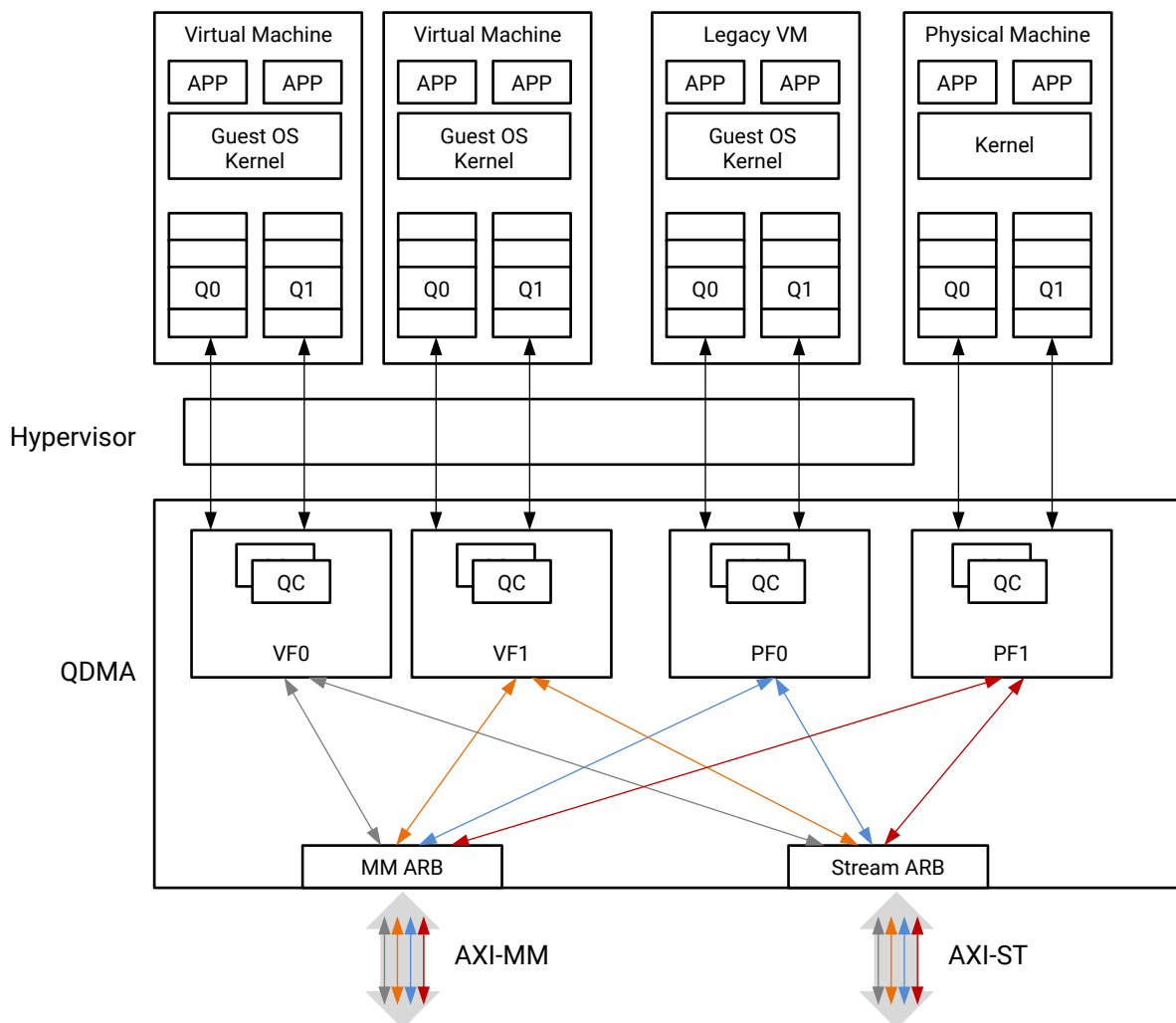
セキュリティは仮想化の重要な部分です。QDMA Subsystem for PCI Express では、次のセキュリティ機能が提供されています。

- QDMA では、権限のある PF のみがキューごとのコンテキストおよびレジスタを設定できます。
- ドライバーは、ドライバーに割り当てられているキューに対してのみ、ポインター アップデートを実行できます。
- システム IOMMU は、DMA が PF および VF でリクエストされていることをチェックする目的で使用できます。ARID は、権限のあるファンクションによってプログラムされるキュー コンテキストから来ます。

PF または VF はどちらもメールボックスを介して PF と通信できます (ただし同じ PF 内の通信はできない)。各ファンクションで、128B の受信ボックスおよび 128B の送信インボックスを 1 つインプリメントできます。これらのメールボックスは、そのファンクションの DMA BAR (通常は BAR0) のドライバーには見えています。どのファンクションも、いつの時点でも、両方のメールボックスに 1 つずつ未処理のメッセージを持つことができます。

次の図に、さまざまなファンクションおよび OS を持つ標準的なシステムでどのように QDMA が使用されるかを示します。さまざまなキューを異なるファンクションに割り当てることができるので、各ファンクションでの DMA パケット転送方法はそれぞれ独立しています。

図 5: システム内の QDMA



X21108-062218

アプリケーション

QDMA Subsystem for PCIe は、ネットワーキング、コンピューティング、およびデータ ストレージ アプリケーションで幅広く使用されています。

QDMA Subsystem for PCIe の一般的な使用例は、データセンターのおよび通信アプリケーションのインプリメンテーションで、ユーザー ロジックで使用される演算アクセラレーション、スマート NIC、NVMe、RDMA NIC (RNIC)、サーバー仮想化、NFV などがある代表例です。QDMA を共有するように複数アプリケーションをインプリメントすることもでき、その場合は、異なるキュー セットおよび PCIe 機能を各アプリケーションに割り当てます。割り当てられたキューは、レート制限、トラフィック優先順位、カスタムワークのキュー エントリ (WQE) をインプリメントするため、ユーザー ロジックで制御されます。

ライセンスおよび注文情報

このサイリンクス LogiCORE™ IP モジュールは、[サイリンクス エンドユーザー ライセンス規約](#)のもと、サイリンクス Vivado® Design Suite を使用して追加コストなしで提供されています。

このサブシステムの詳細は、[QDMA Subsystem for PCIe 製品ページ](#)を参照してください。

製品仕様

規格

このサブシステムは次の規格に準拠しています。

- AMBA AXI4-Stream Protocol Specification ([ARM IHI 0051A](#))
- PCI Express 基本仕様 v3.1
- PCI ローカル バス仕様
- PCI-SIG® シングル ルート I/O 仮想化および共有 (SR-IOV) 仕様

詳細は、PCI-SIG Specifications (<http://www.pcisig.com/specifications>) を参照してください。

最小デバイス要件

Gen3x16 機能には、最低 -2 のスピード グレードが必要です。

表 4: 最小デバイス要件

リンク スピード	リンク幅	サポートされるスピード グレード
UltraScale+™ ファミリ		
Gen1/Gen2	x1、x2、x4、x8、x16	-1、-1L、-1LV、-2、-2L、-2LV、-3
Gen3	x1、x2、x4	-1、-1L、-1LV、-2、-2L、-2LV、-3
	x8	-1、-2、-2L、-3
	x16	-2、-2L、-3
Virtex® UltraScale+ (HBM 付き)		
Gen1/Gen2	x1、x2、x4、x8、x16	-1、-2、-2L、-2LV、-3
Gen3	x1、x2、x4	-1、-2、-2L、-2LV、-3
	x8	-1、-2、-2L、-3
	x16	-2、-2L、-3

注記: この IP では、XCZU4EV、XCZU4CG、XCZU4EG、XAZU4EV、XCZU5CG、XCZU5EG、XAZU5EV、および XQZU5EV デバイスを除く、PCIe ブロックのある UltraScale+™ デバイスがすべてサポートされています。

QDMA 操作

ディスクリプター エンジン

ディスクリプター エンジンは、各キューの H2C および C2H ディスクリプター リング バッファのコンシューマー サイドを管理します。ディスクリプター エンジンの各キューの処理方法は、各キューのコンテキストにより決定されます。ディスクリプター が使用可能で、その他の条件も満たされている場合、ディスクリプター をフェッチするため、ディスクリプター エンジンによって読み出しリクエストが PCIe に発行されます。受信されたディスクリプター は、ディスクリプター バイパス出力インターフェイス (バイパス モード) に渡されるか、または直接 DMA エンジンに送られます (内部モード)。H2C ストリームまたはメモリ マップドの DMA エンジンがディスクリプター を完了すると、ステータス ディスクリプター がステータスにライトバックされ、ソフトウェアおよびユーザー ロジックに現在の DMA の進捗状況を知らせるため、割り込みまたはマーカー応答が生成されます。ディスクリプター エンジンは、ユーザー ロジックに各キューの一部ステータスを知らせるトラフィック マネージャー インターフェイスも提供します。これにより、DMA 動作のカスタマイズおよび最適化が必要な場合に、ユーザー ロジックで情報に基づいた判断ができるようになります。

ディスクリプター コンテキスト

ディスクリプター コンテキストには、ブロック RAM または URAM に格納可能なキューごとの設定、ステータス、制御情報がディスクリプター エンジンによって格納されます。コンテキストは、H2C または C2H の QID でインデックス化されます。キューをイネーブルにする前に、まずハードウェアおよびクレジット コンテキストをクリアする必要があります。それが終わると、キューをイネーブルにするため、ソフトウェア コンテキストがプログラムされ、gen ビットがセットされます。キューがイネーブルになったら、プロデューサー インデックスおよび割り込みアーム ビットをアップデートするために、直接マップされたアドレス空間を介してのみソフトウェア コンテキストをアップデートする必要があります (キューをディスエーブル中の場合を除く)。詳細は、[QDMA_DMAP_SEL_H2C_DSC_PIDX\[2048\] \(0x18004\)](#) および [QDMA_DMAP_SEL_C2H_DSC_PIDX\[2048\] \(0x18008\)](#) を参照してください。ハードウェア コンテキストおよびクレジット コンテキストにはステータスのみが含まれています。この 2 つのコンテキストを処理する必要があるのは、キュー初期化の一部としてすべて 0 にクリアするときだけです。キューがイネーブルになったら、コンテキストはハードウェアによりダイナミックにアップデートされます。キューがイネーブルのときに、間接的なバスを介してコンテキストを変更すると、予期しない動作が発生する可能性があります。また、キューがイネーブルのときにコンテキストを読み出すとパフォーマンスが低下するので推奨しません。

ソフトウェア ディスクリプター コンテキスト構造 (0x0 C2H および 0x1 H2C)

ディスクリプター コンテキストはディスクリプター エンジンによって使用されます。

表 5: ソフトウェア ディスクリプター コンテキスト構造の定義

ビット	ビット幅	フィールド名	説明
[139]	1	int_aggr	セットされている場合、割り込みは割り込みリングでまとめられます。
[138:128]	[10:0]	vec	直接割り込みの割り込み、またはアグリゲートされた割り込みの割り込みアグリゲーション エントリに使用される MSI-X ベクター。
[127:64]	64	dsc_base	ディスクリプター リングのベース アドレス。

表 5: ソフトウェア ディスクリプター コンテキスト構造の定義 (続き)

ビット	ビット幅	フィールド名	説明
[63]	1	is_mm	キューがメモリ マップドであるかどうかを判断します。 このフィールドがセットされている場合、ディスクリプターが関連付けられている H2C または C2H メモリ マップド エンジンに送信されます。 1: メモリ マップド 0: ストリーム
[62]	1	mrkr_dis	セットされている場合、マーカ-応答は内部モードです。 C2H ST は対象外です。
[61]	1	irq_req	エラーが原因の割り込みで、送信待機中です (irq_arm を待つ)。キュー コンテキストが初期化されると、このビットはクリアになるはずですが。 C2H ST は対象外です。
[60]	1	err_wb_sent	エラーがあると、ライトバック/割り込みが送信されます。 このビットがセットされると、ライトバックまたは割り込みはキューには送信されません。キュー コンテキストが初期化されると、このビットはクリアになるはずですが。 C2H ST は対象外です。
[59:58]	2	err	エラー ステータス。 ビット [1] dma - DMA 操作中にエラーが発生。エンジン ステータス レジスタをチェックします。 ビット [0] dsc - ディスクリプター フェッチまたはアップデート中にエラーが発生。ディスクリプター エンジン ステータス レジスタをチェックします。キュー コンテキストが初期化されると、このフィールドは 0 にセットされるはずですが。
[57]	1	irq_no_last	割り込みは送信されず、PIDX/CIDX が内部モードでアイドル状態です。irq_arm ビットがセットされると、割り込みが送信されます。割り込みが送信されると、またはキューの PIDX がアップデートされると、このビットは自動的にクリアになります。 キュー コンテキストが初期化されると、このビットは 0 に初期化されるはずですが。 C2H ST は対象外です。
[56:54]	3	port_id	ポート ID このキューに関連付けられているイベントに対し、ユーザー インターフェイスで送信されるポート ID です。
[53]	1	irq_en	割り込みイネーブル。 ホスト ステータス アップデートがあると、ホストへの割り込みが送信されます。 C2H ST の場合は 0 にセットされます。
[52]	1	wbk_en	ライトバック イネーブル。 ホスト ステータス アップデートがあると、ステータス ディスクリプターへのメモリ書き込みが送信されます。
[51]	1	mm_chn	0 にセットされます。
[50]	1	bypass	セットされていると、キューはバイパス モードで動作しますが、セットされていない場合は、内部モードで動作します。

表 5: ソフトウェア ディスクリプター コンテキスト構造の定義 (続き)

ビット	ビット幅	フィールド名	説明
[49:48]	2	dsc_sz	ディスクリプター フェッチのサイズ。0: 8B、1: 16B、2: 32B、3: 64B。 バイパス モードがイネーブルになっていない場合、メモリ マップド DAM には 32B、H2C ストリーム DMA には 16B、C2H ストリーム DMA には 8B が必要です。 キューがバイパス モードの場合、任意のディスクリプター サイズを選択できます。ディスクリプターはバイパス出力 インターフェイスで送信されます。ディスクリプターがディスクリプター バイパス入力に戻される前に処理されるかどうかは、ユーザー ロジックにより決まります。
[47:44]	4	rng_sz	リング サイズ レジスタへのディスクリプター リング サイズ インデックス。
[43:40]	4		予約
[39:37]	3	fetch_max	このキューに格納できるディスクリプター フェッチの最大数。fetch_max + 1 が最大数です。
[36]	1	at	ベース アドレスのアドレス タイプ。 0: 未変換 1: 変換済み ディスクリプター フェッチおよびステータス ディスクリプター ライトバックのときに PCIe で使用されるアドレス タイプ (AT) です。
[35]	1	wbi_intvl_en	ライトバック/割り込みの間隔。 処理されるディスクリプターの数に基づいた定期的なステータス アップデートをイネーブル。 内部モードが対象です。 C2H ST は対象外です。ライトバックの間隔は QDMA_GLBL_DSC_CFG.wb_acc_int で決定されます。
[34]	1	wbi_chk	チェック保留後のライトバック/割り込み。 キューが使用可能なディスクリプターをすべて完了させたときのステータス アップデートをイネーブルにします。 内部モードが対象です。
[33]	1	fcrd_en	フェッチ クレジットをイネーブル。 フェッチされたディスクリプターの数がこのキューに送信されたクレジット数で満たされます。 C2H ST の場合は 1 にセットされます。
[32]	1	qen	キューがイネーブルになっていることを示します。
[31:25]	7		予約
[24:17]	8	fnc_id	ファンクション ID
[16]	1	irq_arm	割り込み Arm。このビットがセットされていると、キューで割り込みを生成できます。
[15:0]	16	pidx	プロデューサー インデックス。

ハードウェア ディスクリプター コンテキスト構造 (0x2 C2H および 0x3 H2C)

表 6: ハードウェア ディスクリプター構造の定義

ビット	ビット幅	フィールド名	説明
[47]	1		予約

表 6: ハードウェア ディスクリプター 構造の定義 (続き)

ビット	ビット幅	フィールド名	説明
[46:43]	4	fetch_pnd	ディスクリプターのフェッチ保留
[42]	1	evt_pnd	イベント遅延
[41]	1	idl_stp_b	無効なキュー、および保留のディスクリプターなし。 キューがイネーブルになるとこのビットがセットされます。キューがディスエーブルになるとこのビットはクリアされ (ソフトウェア コンテキストの QEN ビット)、保留のディスクリプターはなくなります。
[40]	1	dsc_pnd	ディスクリプターの保留。完了した最終の CIDX が現在の PIDX に一致しない場合、ディスクリプターは保留に定義されます。
[39:32]	8		予約
[31:16]	16	crd_use	消費されたクレジット。フェッチ クレジットがソフトウェア コンテキストでイネーブルになっている場合に適用可能です。
[15:0]	16	cidx	最後にフェッチされたディスクリプターのコンシューマー インデックス。

クレジット ディスクリプター コンテキスト構造

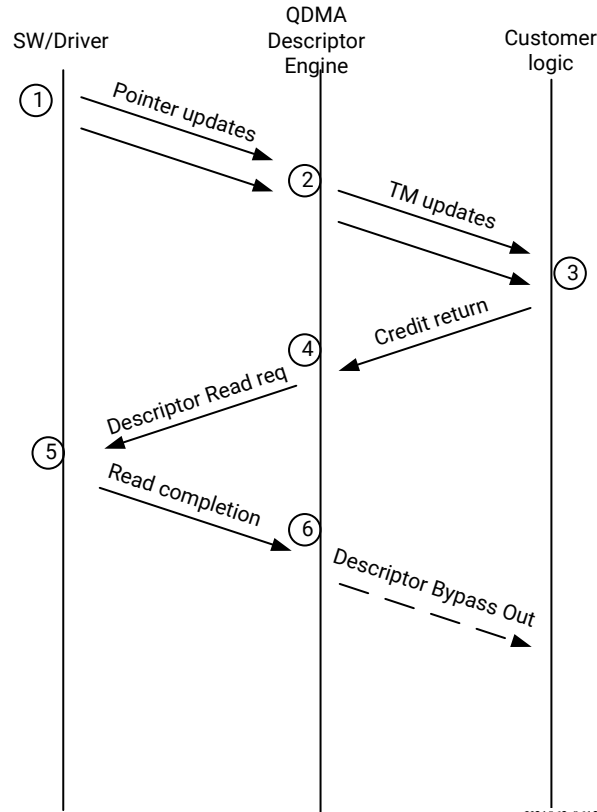
表 7: クレジット ディスクリプター コンテキスト構造の定義

ビット	ビット幅	フィールド名	説明
[31:16]	16		予約
[15:0]	16	cred	受信されたフェッチ クレジット。 フェッチ クレジットがソフトウェア コンテキストでイネーブルになっている場合に適用可能です。

ディスクリプター フェッチ

図 6: ディスクリプター フェッチのフロー

Descriptor Fetch Flow



X21062-061218

1. ディスクリプター エンジン、キューのディスクリプター PIDX コンテキストへのアップデートがあると、ディスクリプターの利用状況を確認します。コンテキストのこの部分は、QDMA_DMAP_SEL_H2C_DSC_PIDX および QDMA_DMAP_SEL_C2H_DSC_PIDX のアドレス空間に直接マップされています。
2. PIDX アップデートがあると、前回フェッチされたコンシューマー インデックス (CIDX) に基づいて使用可能なディスクリプターの数をディスクリプター エンジンが評価します。新しく使用可能なディスクリプターの数は、トラフィック マネージャー ステータス インターフェイスを介して、ユーザー ロジックに通知されます。
3. フェッチ クレジットがイネーブルの場合、ユーザー ロジックは、フェッチされるべきディスクリプターそれぞれに対しクレジットを出力する必要があります。
4. ディスクリプターが使用可能で、フェッチ クレジットがディスエーブルになっているか 0 以外の場合、ディスクリプター エンジン PCIe へディスクリプター フェッチを生成します。ディスクリプター フェッチの数は、必要に応じて、PCIe の最大読み出しリクエスト サイズ (MRRS) およびディスクリプター フェッチ クレジットによってさらに絞り込まれます。また、完了空間が足りない場合は、ディスクリプター フェッチはストールする可能性もあります。C2H および H2C には、それぞれにディスクリプター フェッチ完了用に 256 個のエントリが割り当てられています。各エントリがデータパスの幅になります。空間が十分あれば、フェッチは続行します。PCIe 上で 1 つのキューに保持できる保留状態のディスクリプター フェッチは、常に 1 つのみです。
5. ホストが読み出しリクエストを受信し、ディスクリプター読み出し完了をディスクリプター エンジンに出力します。

- ディスクリプターはオフロードできる状態になるまでバッファーに格納されます。キューがバイパス モードの場合、ディスクリプターはディスクリプター バイパス出力ポートに送信されます。バイパス モードでなければ、DMA エンジンに直接送信されます。いったん送信されると、ディスクリプター フェッチ完了バッファー空間の割り当ては解除されます。

注記: ソフトウェアで PIDX をリング サイズから 2 を引いた値を超える数にアップデートすることはできません。使用可能なディスクリプターの数に常に、リング サイズから 2 を引いた数になります。

内部モード

ソフトウェア コンテキスト バイパス フィールドを設定し、ディスクリプター バイパス モードまたは内部モードで操作するようにキューを設定できます。キューが内部モードの場合は、ディスクリプターの処理に外部ユーザー ロジックは必要ありません。ディスクリプター エンジンによってフェッチされたディスクリプターは、直接該当する DMA エンジンに送られ、処理されます。内部モードの場合は、ディスクリプターのフェッチ動作をランタイムでカスタマイズするため、フェッチ クレジット、およびユーザー ロジックへのステータス アップデートを可能にします。

内部モードのライトバックおよび割り込み (AXI MM および H2C ST)

ステータス ライトバックおよび割り込みは、キュー コンテキストに基づいてハードウェアで自動的に生成されます。wbi_intvl_en がセットされていると、レジスタ QDMA_GLBL_DSC_CFG.wb_intvl で選択されている間隔に基づいてライトバック/書き込みが送信されます。割り込みは遅いため、間隔モードでは、割り込みが遅れるか、または間隔がスキップされる可能性があります。wbi_chk コンテキスト ビットがセットされている場合、ディスクリプター エンジンが現在の PIDX での最終ディスクリプターが完了したことを検出すると、ライトバック/割り込みが送信されます。間隔モードがイネーブルの場合も含め、内部モードのすべての操作に対し、wbi_chk ビットをセットすることをお勧めします。irq_arm ビットがソフトウェアによってセットされるまで、割り込みは生成されません。割り込みが送信されると、irq_arm ビットはハードウェアによってクリアになります。irq_arm ビットがセットされていないときに割り込みが必要になった場合は、irq_arm ビットがセットされるまで、割り込みは保留状態になります。

ディスクリプター完了は、ディスクリプターのデータ転送が完了し、AXI で書き込みデータが受信されたことを示す肯定応答が送信されたか (AXI MM の H2C bresp、ST の有効/レディ)、または書き込みデータが転送用の PCIe コントローラーのトランザクション層 (C2H MM) に受領されたときに定義されています。

バイパス モード

バイパス モードでも、ユーザー ロジックへのクレジットおよびステータス アップデートがサポートされています。ディスクリプターおよびステータス アップデートの処理をユーザー ロジックでカスタマイズすることも可能です。ディスクリプター エンジンによってフェッチされたディスクリプターは、ディスクリプター バイパス出力インターフェイスを介してユーザー ロジックに送信されます。これにより、必要に応じて、ユーザー ロジックでディスクリプターを前処理したり、格納したりできるようになります。バイパス出力インターフェイスでは、ディスクリプターはカスタム フォーマットにできます (ディスクリプターのサイズは変えない)。DMA 操作を実行するには、ユーザー ロジックがディスクリプター (QDMA フォーマットにする必要がある) をディスクリプター バイパス入力インターフェイスに駆動します。

ユーザー ロジックに既にディスクリプターがある場合は、QDMA フォーマットである必要があり、そのフォーマットであれば、ディスクリプター バイパス ポートを介して DMA に直接出力できます。ディスクリプターが既にユーザー ロジックにあれば、ホストからディスクリプターをフェッチする必要はありません。

バイパス モードのライトバック/割り込み

バイパス モードでは、ホストへのステータス アップデート、およびユーザー ロジックに返されたマーカール応答をユーザー ロジックで明示的に制御できます。メモリ マップド エンジンまたは H2C ストリーム DMA エンジンのディスクリプター バイパス入力ポートに送信されている各ディスクリプターには、CIDX および wbi フィールドがあります。CIDX は、ディスクリプター完了時に生成されるステータス アップデート (ホスト ライトバック、マーカール応答、連結割り込み) で、どのディスクリプターが完了あるかを識別するために使用されます。ディスクリプターの wbi フィールドに入力がある場合、コンテキストの wbk_en ビットがセットされていればホストへのライトバックが生成されます。wbi ビットがセットされていて、コンテキストの irq_en および irq_arm のビットもセットされている場合は、割り込みも送信されます。

割り込みがイネーブルの場合、ユーザー ロジックで irq_arm のトラフィック マネージャー出力を監視する必要があります。キューの irq_arm ビットが検出された後のみ、wbi ビットがセットされているディスクリプターが DMA に送信されます。wbi ビットを持つディスクリプターが送信されると、別の irq_arm がアサートされていなければ、wbi ビットを持つ別のディスクリプターは送信できません。この irq_arm ビットが正しく検出されていないときに wbi ビットがセットされると、割り込みが送信される場合とされない場合があり、ソフトウェアが割り込みを恒久的に待つ状態になる可能性があります。割り込みがイネーブルでない場合は、wbi ビットのセットに関しては制限はありません。ただし、ライトバック イベントが多すぎると、ディスクリプター エンジンのパフォーマンスが著しく低下し、ホストへの書き込み帯域幅を使い切ってしまう可能性があります。

ディスクリプター完了は、ディスクリプターのデータ転送が完了し、AXI で書き込みデータが受信されたことを示す肯定応答が送信されたか (AXI MM の H2C bresp、ST の有効/レディ)、または書き込みデータが転送用の PCIe コントローラーのトランザクション層 (C2H MM) に受領されたときと定義されています。

バイパス モードのマーカール応答

mrkr_req ビットをセットすると、どのディスクリプターにもマーカール応答を生成できます。マーカール応答が生成されるのはディスクリプター完了後です。ホストのライトバックと同様、マーカール応答リクエストが多すぎると、ディスクリプター エンジンのパフォーマンスが低下します。また、ユーザー ロジックへのマーカール応答は、コンテキストで設定されていれば、wbi ビットと共に送信できます。送信されたマーカール応答は、ディスクリプターに関連付けられている CIDX、キュー ID、DMA の方向で識別できます。

ディスクリプター完了は、ディスクリプターのデータ転送が完了し、AXI で書き込みデータが受信されたことを示す肯定応答が送信されたか (AXI MM の H2C bresp、ST の有効/レディ)、または書き込みデータが転送用の PCIe コントローラーのトランザクション層 (C2H MM) に受領されたときと定義されています。

トラフィック マネージャー出力インターフェイス

トラフィック マネージャー インターフェイスは、ユーザー ロジックにキュー ステータスの詳細を供給します。ユーザー ロジックは、その情報に基づいてディスクリプターのフェッチおよび実行を管理します。標準操作では、イネーブルになっているキューに対して、irq_arm ビットがアサートされるたび、またはキューの PIDX がアップデートされるたびに、ディスクリプター エンジンが tm_dsc_sts_valid をアサートします。tm_dsc_sts_avl 信号は、前回のアップデート以降、新たに使用可能になったディスクリプターの数を示します。このメカニズムを利用して、ユーザー ロジックは各キューの作業量を確認します。これは、ディスクリプター エンジンのフェッチクレジットメカニズムやその他のユーザー最適化を利用したフェッチの優先順位付けに使用できます。有効サイクルで、tm_dsc_sts_irq_arm は irq_arm ビットが 0 でセットされたことを示します。バイパス モードでは実質的にこれがこのキューの割り込みのクレジットになります。詳細は、前述のバイパス モードの割り込みに関するセクションを参照してください。キューがソフトウェアにより無効にされるか、エラーが原因で無効になると、tm_dsc_sts_qinv ビットがセットされます。このビットが検出された場合は、ディスクリプター エンジンがそのキューの新しいディスクリプター フェッチを停止します。この場合、tm_dsc_sts_avl の内容は、ディスクリプタ

エンジンにより保持されているフェッチ クレジットの使用可能数を示します。この情報は、ユーザー ロジックでディスクリプター エンジンに送信されるクレジット数およびディスクリプター エンジンが受信するはずのディスクリプター の数を判断するために使用できます。tm_dsc_sts_qin がアサートされた後でも、既にフェッチ パイプラインにある有効なディスクリプターは、DMA エンジン (内部モード) またはディスクリプター バイパス出力ポート (バイパス モード) に引き続き送信されます。

tm_dsc_sts インターフェイスのその他のフィールドは、キュー ID、DMA 方向 (H2C または C2H)、内部モード/バイパス モード、ストリーム モード/メモリ マップド モード、キュー イネーブル ステータス、キュー エラー ステータス、およびポート ID を識別します。

tm_dsc_sts インターフェイスは有効/準備完了状態のインターフェイスですが、パフォーマンスを最適化する目的でこのインターフェイスをバック プレッシュャーすべきではありません。複数のイベントが tm_dsc_sts サイクルをトリガーするので、内部バッファがフルになると、新しいイベントが生成されないようにするため、ディスクリプターのフェッチが停止します。

ポートの詳細は、[QDMA トラフィック マネージャー クレジット出力ポート](#) を参照してください。

ディスクリプター クレジット入力インターフェイス

クレジット インターフェイスは、キューの fcrd_en コンテキスト ビットがセットされているときに使用されます。このインターフェイスでは、各キューでフェッチされるディスクリプターの優先順位がユーザー ロジックによって決められ、監視されます。DMA の方向、QID、クレジット値はユーザーが指定できます。一般的なユース ケースでは、ディスクリプターをフェッチするクレジット入力ディスクリプター エンジンにより使用されます。内部的には、受信および消費されたクレジットは、キューごとに管理されます。キューがイネーブルでないときにクレジットが追加された場合、クレジットは tm_dsc_sts_qinv がアサートされてトラフィック マネージャー出力インターフェイスを介して返され、返されたクレジットは tm_dsc_sts_avl に含まれます。

詳細は、[QDMA ディスクリプター クレジット入力ポート](#) を参照してください。

エラー

エラーは、ディスクリプター フェッチ中およびディスクリプター実行中の両方で発生する可能性があります。どちらの場合も、キューに対してエラーが検出されると、そのキューが無効になり、コンテキストにエラー ビットが記録されて、エラーの発生したキューでのディスクリプターのフェッチが停止します。ステータス レジスタでエラーを記録することも可能です。ライトバック、割り込み、またはマーカール応用にイネーブルになっていれば、DMA でそれぞれのインターフェイスにステータス アップデートが生成されます。この生成が完了すると、キュー コンテキストがクリアになるまで、キューに対しては追加のライトバック、割り込み、マーカール応答 (内部モード) は送信されません。エラーが発生してキューが無効になると、それを示すため、トラフィック マネージャー出力サイクルも生成されません。

新たなディスクリプター フェッチは実行されなくなりますが、既にパイプラインにあるフェッチの処理は続行され、通常どおり、ディスクリプターが DMA エンジンまたはディスクリプター バイパス出力インターフェイスに送信されます。ディスクリプター フェッチ自体にエラーが発生した場合は、そのディスクリプターがエラー ビットでマークされます。エラー ビットがセットされている場合、ディスクリプターの内容が無効だとみなされます。同じキューの後続ディスクリプター フェッチにはエラーがなく、エラー ビットがセットされていない可能性があります。

メモリ マップド DMA

メモリ マップド DMA 操作では、DMA のソースおよびデスティネーションの両方がメモリ マップド空間です。H2C 転送では、ソース アドレスは PCIe アドレス空間に、デスティネーション アドレスは AXI MM アドレス空間に属しています。C2H 転送では、ソース アドレスは AXI MM アドレス空間に、デスティネーション アドレスは PCIe アドレス空間に属しています。PCIe-to-PCIe および AXI MM-to-AXI MM の DMA はサポートされていません。DMA の方向以外は、H2C および C2H の DMA の転送は同様に動作をし、同じディスクリプター フォーマットを共有しています。

操作

メモリ マップド DMA エンジン (H2C および C2H) は、メモリ マップド エンジン制御レジスタの `run` ビットをセットするとイネーブルになります。`run` ビットがディアサートされると、ディスクリプターが破棄される可能性があります。ソース バッファ フェッチを既に開始しているディスクリプターは引き続き処理されます。`run` ビットが再アサートされると、内部エンジン ステートはリセットされますが、エンジンが静止状態の場合にのみリセットされるはずで、ディスクリプターは、ディスクリプター エンジンから直接受信されるか、またはディスクリプター バイパス入力インターフェイスを介して受信されます。キューが内部モードの場合は、ディスクリプター バイパス入力インターフェイスを介してディスクリプターを送信しないでください。稼動していないメモリ マップド エンジンに送信されたディスクリプターはすべて破棄されます。内部モードのキューおよびバイパス モードのキューが混合している場合は、順序を確立するため、ラウンドロビンのアービトレーションが実行されます。

DMA メモリ マップド エンジンで、まずソース インターフェイスへ読み出しリクエストが生成され、そのインターフェイスのアライメント境界でディスクリプターが分割されます。PCIe および AXI 読み出しインターフェイスはそれぞれ異なるアライメントで分割するように設定できます。読み出しが発行されると、読み出しデータの完了空間が先行して割り当てられます。書き込みリクエストの場合も同様に、DMA エンジンで適切なアライメントでディスクリプターが分割されます。AXI インターフェイスでは、各エンジンで 1 つの AXI ID が使用されます。DMA エンジンで、読み出しが出力された順序で、読み出し完了/書き込みデータの順序が入れ替えられます。十分な読み出し完了データが受信されると、読み出しデータがリクエストされたのと同じ順序で、デスティネーション インターフェイスに書き込みリクエストが出力されます。そのリクエストが完了する前に、デスティネーション インターフェイスはすべての書き込みデータを受領し、完了応答を送信する必要があります。PCIe では、書き込みリクエストがトランザクション層で受領されたときに書き込み完了が発行され、次のリンクで送信されます。AXI メモリマップドインターフェイスの場合は、`bresponse` が完了の条件になります。完了条件が満たされれば、ホスト ライトバック、割り込み、またはマーカール応答が該当するディスクリプター用に生成されます。詳細は、ディスクリプター エンジンの内部モード ライトバックおよび割り込み、バイパス モードのライトバックおよび割り込みを参照してください。

DMA メモリ マップド エンジンでは、ディスクリプター バイパス入力の `no_dma` フィールド、および長さ 0 の DMA もサポートされており、どちらのケースもまったく同じように処理されます。ほかのすべてのディスクリプターと同じように、このディスクリプターも DMA エンジンを通じて送信されるので、キュー内のディスクリプターの順序は維持されます。ただし、DMA 読み出しまたは書き込みリクエストは生成されません。ディスクリプターのステータス アップデート チェックが完了すると、長さが 0、または `no_dma` のディスクリプターのステータス アップデートが処理されます。

エラー

DMA メモリ マップド エンジンには主に 2 種類のエラーがあります。1 つは、入力ディスクリプターを使用して設定されるエラー ビットです。この場合、ディスクリプターの DMA 操作は実行されませんが、ディスクリプターはエンジンを介し、エラーが発生していることが示されている状態でステータス アップデート段階に進みます。コンテキストおよび設定によって、ライトバック、割り込み、またはマーカール応答が発生します。また、キューも無効になります。詳細は、「ディスクリプター エンジン エラー」を参照してください。もう 1 つは、DMA そのものを実行しているときに発生するエラーです。これには、PCIe 読み出し完了エラー、AXI Bresponse エラー (H2C)、または AXI Rresponse エラー、バス マスター イネーブルまたはファンクション レベル リセット (FLR) が原因の PCIe 書き込みエラー、および RAM ECC エラーが含まれます。最初にイネーブルになったエラーが DMA エンジンに記録されます。詳細は、メモリ マップド エンジンのエラー ログを参照してください。読み出しでエラーが発生した場合は、可能で

あれば、DMA 書き込みが中止されます。RAM からの書き込みデータ収集中にエラーが検出された場合は、リクエストを中止できません。代わりに、この問題がデスティネーションで認識されるようにするため、無効データ パリティが生成されます。エラーが発生したディスクリプターが DMA エンジンを通ると、エラーを示すステータス アップデートが生成されます。ディスクリプター エラーと同様に、キューが無効になります。詳細は、「ディスクリプター エンジン エラー」を参照してください。

AXI4 メモリ マップド ディスクリプター (H2C および C2H 用) (32B)

表 8: AXI4 メモリ マップド ディスクリプター 構造 (H2C および C2H 用)

ビット	ビット幅	フィールド名	説明
[255:192]	64		予約
[191:128]	64	dst_addr	デスティネーション アドレス
[127:92]	36		予約
[91:64]	28	lengthInByte	読み出しの長さ (バイト)
[63:0]	64	src_addr	ソース アドレス

内部モードのメモリ マップド DMA は、ディスクリプター キューを 32B に設定し、上記のディスクリプター フォーマットに従う必要があります。バイパス モードでは、ディスクリプター フォーマットはユーザー ロジックにより定義されます。このユーザー ロジックは、H2C または C2H のメモリ マップド バイパス入力ポートを駆動する必要があります。

AXI4 メモリ マップド ライトバック ステータス構造 (H2C および C2H 用)

メモリ マップド ライトバック ステータス レジスタは、(H2C および C2H 用) ディスクリプターの最後のエントリの後にあります。

表 9: AXI4 メモリ マップド ライトバック ステータス構造 (H2C および C2H 用)

ビット	ビット幅	フィールド名	説明
[63:48]	16		予約
[47:32]	16	pidx	ライトバック時のプロデューサー インデックス
[31:16]	16	cidx	コンシューマー インデックス
[15:2]	14		予約
[1:0]	2	err	エラー ビット 1: ディスクリプターのフェッチ エラー ビット 0: DMA エラー

ストリーム モードの DMA

H2C ストリーム エンジン

H2C ストリーム エンジンは、ホストからのストリーミング データをユーザー ロジックに送信するのが役割で、H2C ストリーム ディスクリプター上で動作します。ユーザー ロジックに送信される開始アドレスおよびデータの長さは、各ディスクリプターにより指定されます。H2C ストリーム エンジンにより、ディスクリプターが解析され、PCIe を介してホストに読み出しリクエストが発行されると、MRRS 境界でそのリクエストが分割されます。ホストの読み出しレイテンシを隠すため、H2C ストリーム エンジンには未処理のリクエストを最高 256 個まで格納できます。また、TLP が戻ってきたときの再順序付け用に、32 KB の再順序付けバッファが H2C ストリーム エンジンによりインプリメントされます。データは、リクエストが PCIe に送信された順番でユーザー ロジックに出力されます。

関連付けられている H2C コンテキストでステータス ディスクリプターがイネーブルになっている場合、ユーザー ロジックにデータを出力し終わると、このエンジンにより追加でステータス ライトバックが送信される可能性があります。

内部モードおよびバイパス モード

QDMA Subsystem for PCIe の各キューは、内部モードおよびバイパス モードという 2 つの H2C ストリーム モードのいずれかでプログラムできます。キュー コンテキストでいずれかのモードを指定します。ディスクリプターの処理が内部モードのキュー用なのか、バイパス モードのキュー用なのかは、H2C ストリーム エンジンによって認識されます。

次の図に、内部モードおよびバイパス モードのフローをそれぞれ示します。

図 7: H2C 内部モードのフロー

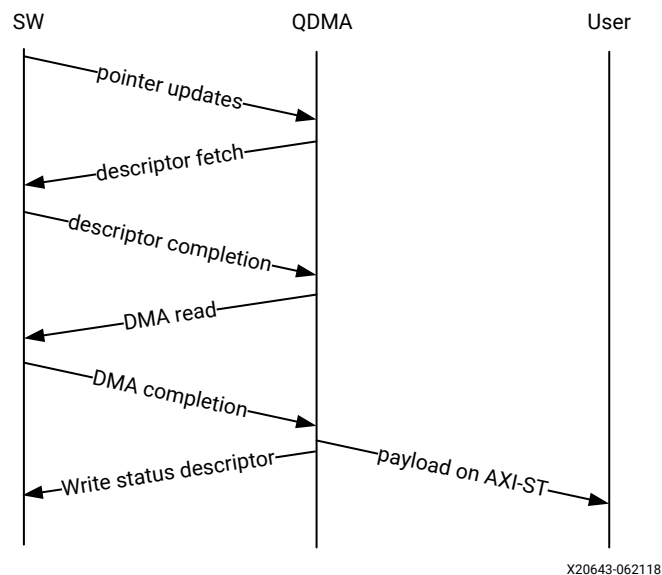
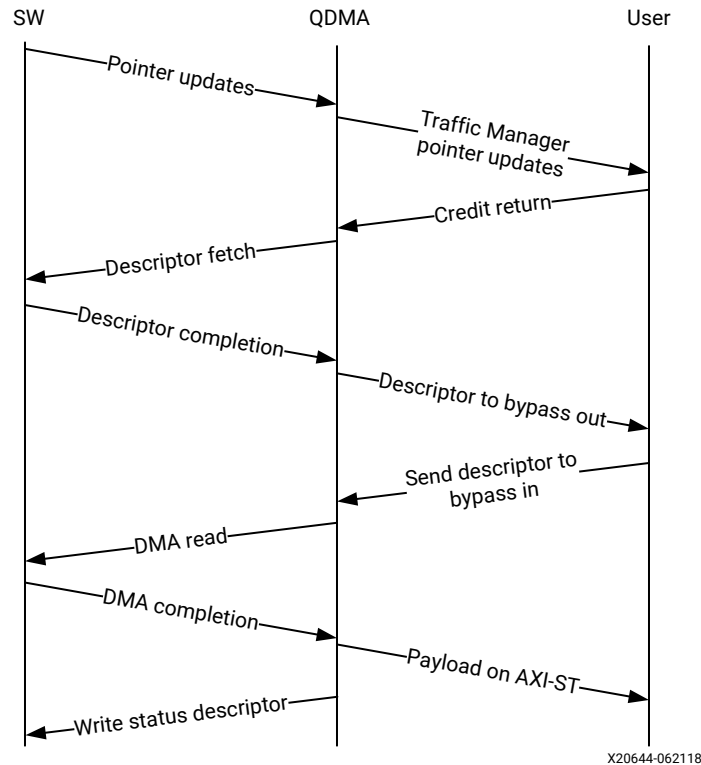


図 8: H2C バイパス モードのフロー



キューが内部モードの場合、ディスクリプターがホストからフェッチされた後、H2C ストリームエンジンに送られて処理されます。データの packets が複数のディスクリプターにまたがることはありません。したがって、キューが内部モードの場合、各ディスクリプターで QDMA H2C AXI ストリーム出力に AXI4-Stream パケットが 1 つだけ生成されます。このパケットが非連続空間のホスト メモリにある場合は、複数のディスクリプターで定義する必要がありますので、キューをバイパス モードでプログラムする必要があります。

バイパス モードでは、ディスクリプターがホストからフェッチされた後、QDMA バイパス出力ポートを介してユーザー ロジックに直接送信されます。QDMA ではこれらのディスクリプターは解析されません。ユーザー ロジックはこれらのディスクリプターを格納できるので、QDMA H2C ストリーム ディスクリプターのバイパス入力インターフェイスを使用して、ディスクリプターからの必要情報を QDMA に戻します。この情報を使用して、QDMA でディスクリプターが構築され、処理用に H2C ストリーム エンジンに送信されます。バイパス モードを使用する場合、次の利点があります。

- ユーザー ロジックにカスタムのディスクリプター フォーマットを使用できます。キューがバイパス モードの場合、QDMA Subsystem for PCIe ではディスクリプターは解析されないため、カスタム フォーマットの使用が可能です。ユーザー ロジックでこれらのディスクリプターが解析され、H2C ストリームのバイパス入力インターフェイスの QDMA に必要な情報が供給されます。
- DMA 操作なしに、即値データをソフトウェアからユーザー ロジックに渡すことができます。
- データをすべてシンクさせる準備が完了すると、QDMA にディスクリプターが送信され、ユーザー ロジックでトラフィックが管理されます。ディスクリプターはローカルの RAM にキャッシュ可能です。
- アドレス変換を実行します。

バイパス モードを使用する場合は、ユーザー ロジックにいくつかの要件があります。バイパス モードでは、パケットが複数のディスクリプターにまたがることのできるため、どのディスクリプターがパケットの開始 (SOP) およびパケットの終了 (EOP) なのかをユーザー ロジックから QDMA に通知する必要があります。QDMA H2C ストリームのバイパス入力インターフェイスには、アドレス、長さ、SOP、EOP などの情報もユーザー ロジックから供給する必要があります。ユーザー ロジックから QDMA に SOP ディスクリプター情報を供給したら、EOP ディスクリプター情報も供給する必要があります。複数のディスクリプターを含むパケットのディスクリプターは、順次送信する必要があります。パケットに属さないほかのディスクリプターを、複数のディスクリプターを含むパケット内に挿入することはできません。ユーザー ロジックで EOP ディスクリプターまでのディスクリプターを収集してから、それらのディスクリプターをまとめて QDMA に戻す必要があります。そうしないとハングする可能性があります。QDMA では、EOP ディスクリプターの最終ビットが送信された後、QDMA H2C AXI ストリームデータ出力で TLAST が生成されます。ユーザー ロジックでパケットのディスクリプターを順次送信するようになっているので、この操作は QDMA で必ず実行されます。

H2C ストリーム インターフェイスはすべてのキューで共有されているので、ユーザー ロジックでパケットをシンクする空間が予約されない場合は、HOL ブロッキング問題が発生する可能性があります。パケットサイズが大きいと、サービスの質が大幅に低下します。ストリーム エンジンは、128B 以上のパケット サイズで PCIe が飽和するように設計されているので、パケット サイズをホスト ページ サイズ、またはユーザー アプリケーションに必要な最大転送単位の制限するようにしてください。

PCIe 側である量のデータが未処理になっていることが H2C ストリーム エンジンで検出された場合、H2C ストリーム エンジンのパフォーマンス制御機能により PCIe RQ/RC に発行されるリクエストがストールされます。この制御機能を使用するには、ソフトウェアで H2C_REQ_THROT (0xE24) レジスタのしきい値をプログラムする必要があります。H2C ストリーム エンジンにある、ユーザー ロジックに送信予定の未処理のデータ量がこのしきい値を超えると、PCIe の RQ/RC に読み出しリクエストは送信されなくなります。この機能はデフォルトでオフになっていますが、H2C_REQ_THROT (0xE24) レジスタでイネーブルにできます。H2C ストリーム エンジンは C2H ストリーム エンジンよりも高速にリクエストを生成できるので、この機能は C2H ストリームのパフォーマンスの改善に役立ちます。ただし、H2C トラフィック用に PCIe 側のリソースを使い切ってしまう可能性があり、そうすると C2H トラフィックに悪影響が出ます。H2C_REQ_THROT (0xE24) レジスタを使用すると、H2C ストリーム エンジンで未処理になる可能性のある読み出しリクエストの最大数のしきい値を別にソフトウェアでイネーブルにし、プログラムすることもできます。つまり、このレジスタでは、H2C ストリーム エンジンの未処理リクエストおよびデータのしきい値を個別にイネーブルにし、プログラムできます。

H2C ストリーム ディスクリプター (16B)

表 10: H2C ディスクリプター構造

ビット	ビット幅	フィールド名	説明
[127:96]	32	addr_h	高位アドレス。ホストのソース アドレスの上位 32 ビット
[95:64]	32	addr_l	下位アドレス。ホストのソース アドレスの下位 32 ビット
[63:48]	16		予約
[47:32]	16	len	パケットの長さ。ディスクリプター用にフェッチされるべきデータの長さ。 内部モードでは、パケットが複数のディスクリプターにまたがることのできないため、これはパケットの長さでもありません。 パケットの最長は、64 K-1 バイトです。
[31:0]	32	metadata	メタデータ。QDMA により、毎ビットでデータと共に H2C-ST TUSER のこのフィールドが渡されます。内部モードのキューの場合、ソフトウェアからユーザー ロジックへデータと共にメッセージを渡すのに使用できます。

この H2C ディスクリプター フォーマットは内部モードの場合のみ使用可能です。バイパス モードの場合は、ユーザー アプリケーションでの必要に応じて、ユーザー ロジックでそのフォーマットを定義できます。

ディスクリプターのメタデータ

バイパス モードと同様、内部モードにもソフトウェアからユーザー ロジックに情報を直接渡すメカニズムがあります。アドレスおよび長さに加え、H2C ストリームのディスクリプターには 32b のメタデータ フィールドもあります。DMA 操作の場合、このフィールドは QDMA Subsystem for PCIe では使用されませんが、代わりに、パケットの各ビートで、H2C AXI4-Stream の `tuser` のユーザー ロジックに渡されます。バイパス モードのキューでは、`tuser` のメタデータを渡すことはサポートされていないので、QDMA H2C ストリームのバイパス入力インターフェイスのメタデータを提供する入力はありません。

長さが 0 のディスクリプター

ディスクリプターの長さフィールドは 0 にできます。0 になっている場合は、H2C ストリーム エンジンにより PCIe で 0 バイトの読み出しリクエストが発行されます。QDMA がこのリクエストの完了を受信すると、H2C ストリーム エンジンにより、QDMA H2C AXI Stream インターフェイスで `tlast` のあるデータが 1 ビート送信されます。インターフェイスで 0 バイトのパケットを示すには、`tuser` の `zero_b_dma` ビットをセットします。ディスクリプターが 0 バイトの場合、ユーザー ロジックで、SOP および EOP の両方を設定する必要があります。設定されていないと、H2C ストリーム エンジンでエラーがフラグされます。

H2C ストリーム ステータス ディスクリプター ライトバック

バイパス入力インターフェイスでディスクリプター情報を送信する場合は、ユーザー ロジックが、ホストからのデータをフェッチした後にホストにステータス ライトバックを送信するよう QDMA Subsystem for PCIe にリクエストできます。ユーザー ロジックは、DMA が完了したときにステータスを出力するようにリクエストすることもできます。これらの動作は、バイパス入力インターフェイスの `sdi` および `mrkr_req` 入力を使用して制御できます。詳細は、[QDMA ディスクリプター バイパス入力ポート](#) を参照してください。

AXI4-Stream H2C ライトバック ステータス構造

H2C ライトバック ステータス レジスタは、H2C ディスクリプター リストの最終エントリの後にあります。

表 11: AXI4-Stream H2C ライトバック ディスクリプター構造

ビット	ビット幅	フィールド名	説明
[63:32]	32		予約
[31:16]	16	<code>cidx</code>	コンシューマー インデックス
[15:0]	16		予約 (プロデューサー インデックス)

H2C ストリーム データ アライナー

H2C エンジンには、データをユーザー ロジックに出力する前に 0 バイト (0B) の境界に揃えるデータ アライナーがあります。これにより、ディスクリプターの開始アドレスを任意に揃えながら、H2C AXI ストリームのデータ バスで、データの開始部分に欠落なくデータを受信できます。ユーザー ロジックは、各ディスクリプターのアドレスおよび長さを任意に揃えてディスクリプターのグループを SOP から EOP に送信できます。アライナーは、異なるディスクリプターからのデータを揃えてまとめ、H2C AXI4-Stream のデータ バスで連続したデータ ストリームを送信します。EOP ディスクリプターの最終ビートが発行されると、そのインターフェイスの `tlast` がアサートされます。

エラーのあるディスクリプターの処理

ディスクリプターをフェッチしているときにエラーが発生した場合、QDMA ディスクリプター エンジンにより、エラーのあるディスクリプターがフラグされます。キューが内部モードの場合、H2C ストリーム エンジンは、PCIe または DMA アクティビティを実行せずに、エラー ディスクリプターを処理します。エラー ディスクリプターがパイプラインを通過するまで待機し、通過後にライトバックを実行します。キューがバイパス モードの場合は、エラー ディスクリプターを含むディスクリプターのバッチを発行しないようにするのはユーザー ロジックの役割です。エラー入力がアサートされたディスクリプターを 1 つだけ H2C ストリームのバイパス入力インターフェイスに送信し、SOP、EOP、no_dma 信号、および sdi または mrkr-req 信号を設定して、H2C ストリーム エンジンがホストにライトバックを実行するようにします。

PCIe からのデータ エラーの処理

H2C ストリーム エンジンで PCIe からのデータ エラーが発生すると、パケット全体にエラー フラグが適用されます。エラーは、H2C ストリーム データ出力の err ビットで示されます。PCIe データ エラーを示すパケットの最終ビットが H2C ストリームから送信されると、エラーを通知するため、ソフトウェアにライトバックも送信されます。

C2H ストリーム エンジン

C2H ストリーム エンジン DMA は、C2H ディスクリプター キューを介してホスト ドライバーにより供給されるディスクリプターにストリーム パケットを書き込みます。

パケットを書き込んでいる DMA に必要なディスクリプターの数を計算するのはプリフェッチ エンジンの役割です。バッファ サイズはキューごとに固定されています。内部モードおよびキャッシュド バイパス モードの場合、プリフェッチ モジュールは、最大で 64 個の異なるキューに対し、512 個までのディスクリプターをフェッチできます。

プリフェッチ エンジンには、低レイテンシ設定 pfch_en = 1 もあります。この場合、エンジンがパケットを受信したときに qdma_c2h_pfch_cfg.num_pfch 個までディスクリプターをプリフェッチでき、後続パケットでの PCIe レイテンシを回避できます。

C2H ストリーム ディスクリプター (8B)

表 12: AXI4-Stream C2H ディスクリプター構造

ビット	ビット幅	フィールド名	説明
[63:0]	64	addr	デスティネーション アドレス

C2H プリフェッチ エンジン

プリフェッチ エンジンは、ディスクリプターとそのペイロードをペアにするため、ディスクリプター フェッチ エンジンと C2H DMA 書き込みエンジンとの間で機能します。

表 13: C2H プリフェッチ コンテキスト構造

ビット	ビット幅	フィールド名	説明
[45]	1	valid	コンテキストが有効です。
[44:29]	16	sw_crdt	ソフトウェア クレジット このフィールドは、内部で使用するため、ハードウェアにより書き込まれます。これは、ソフトウェアにより 0 に初期化されてから、読み出し専用として処理されます。

表 13: C2H プリフェッチ コンテキスト構造 (続き)

ビット	ビット幅	フィールド名	説明
[28]	1	pfch	キューがプリフェッチ状態です。 このフィールドは、内部で使用するため、ハードウェアにより書き込まれます。これは、ソフトウェアにより 0 に初期化されてから、読み出し専用として処理されます。
[27]	1	pfch_en	プリフェッチをイネーブルにします。
[26]	1	err	このキューでエラーが検出されています。
[25:8]	18		予約
[7:5]	3	port_id	ポート ID
[4:1]	4	buf_size_idx	バッファー サイズ インデックス
[0]	1	bypass	C2H がバイパス モードです。

C2H ストリーム モード

C2H ディスクリプターのソースは、ディスクリプター フェッチ エンジンまたは C2H バイパス入力インターフェイスです。ディスクリプター フェッチ エンジンからのディスクリプターは常にキャッシュ モードです。ディスクリプターの順序は、ユーザーからの C2H データ パケットとペアにするため、プリフェッチ エンジンにより維持されます。C2H バイパス入力インターフェイスからのディスクリプターには、単純モード用にインターフェイスが 1 つ、キャッシュ モード用にもう 1 つインターフェイスがあります。単純モードの場合、ディスクリプターの順序は、C2H データ パケットとペアにするため、ユーザー アプリケーションにより維持されます。キャッシュ モードの場合、ディスクリプターの順序は、ユーザーからの C2H データ パケットとペアにするため、プリフェッチ エンジンにより維持されます。

プリフェッチ コンテキストにはバイパス ビットがあります。このビットが 1'b1 の場合は、ディスクリプターのクレジットはユーザー アプリケーションにより送信されます。このビットが 1'b0 の場合は、ディスクリプターのクレジットはプリフェッチ エンジンにより処理されます。

ディスクリプター コンテキストにはバイパス ビットがあります。このビットが 1'b1 の場合、ディスクリプター フェッチ エンジンにより C2H バイパス出力インターフェイスのディスクリプターが送信され、ユーザー アプリケーションによりそれが変換されて、C2H バイパス入力インターフェイスの QDMA Subsystem for PCIe にループバックされます。このビットが 1'b0 の場合は、ディスクリプター フェッチ エンジンにより ディスクリプターがプリフェッチ エンジンに直接送信されます。

キューごとに、3 つのケースがサポートされています。

表 14: C2H ストリーム モード

	c2h_byp_in	desc_ctxt.desc_byp	pfch_ctxt.bypass
単純バイパス モード	simple byp in	1	1
キャッシュ バイパス モード	cache byp in	1	0
キャッシュ内部モード	なし	0	0

単純バイパス モードの場合、ディスクリプター フェッチ エンジンにより、C2H バイパス出力インターフェイスのディスクリプターが送信され、ユーザー アプリケーションでそのディスクリプターが変換され、単純モードの C2H バイパス入力インターフェイスの QDMA にループバックされます。ユーザー アプリケーションにより、ディスクリプターのクレジットが送信され、ディスクリプターの順序が維持されます。

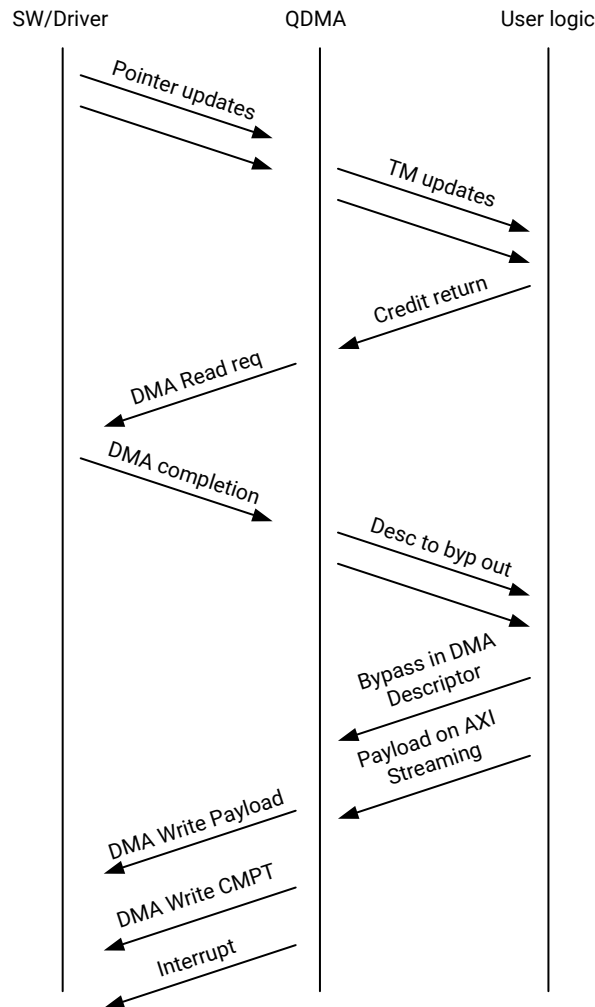
キャッシュバイパス モードの場合、ディスクリプター フェッチ エンジンにより、C2H バイパス出力インターフェイスのディスクリプターが送信され、ユーザー アプリケーションでディスクリプターが変換されて、キャッシュ モード C2H バイパス入力インターフェイスの QDMA にループバックされます。プリフェッチ エンジンにより、ディスクリプターのクレジットが送信され、ディスクリプターの順序が維持されます。

キャッシュ内部モードの場合、ディスクリプター フェッチ エンジンでディスクリプターがプリフェッチ エンジンに送信されます。プリフェッチ エンジンにより、ディスクリプターのクレジットが送信され、ディスクリプターの順序が維持されます。この場合、ディスクリプターは C2H バイパス出力から送信されず、C2H バイパス入力インターフェイスに戻ってきません。

C2H ディスクリプター バイパス フローは、次のとおりです。

図 9: C2H ディスクリプター バイパス フロー

C2H Flow



X20604-120718

ポートの詳細は、[QDMA ディスクリプター バイパス入力ポート](#) および [QDMA ディスクリプター バイパス出力ポート](#) を参照してください。

C2H ストリーム パケット タイプ

次に、C2H ストリーム パケットのタイプを示します。

一般的なパケット

一般的な C2H パケットには、データ パケットと完了パケット (CMPT) の両方があり、この 2 つは対になっています。

また、一般的な C2H データ パケットは複数のビットにまたがる可能性があります。

- `s_axis_c2h_ctrl_qid`: C2H ディスクリプターのキュー ID。
- `s_axis_c2h_ctrl_len`: パケットの長さ。
- `s_axis_c2h_mty`: ビートの空バイト。
- `s_axis_c2h_ctrl_has_cmpt = 1'b1`。このデータ パケットには対応する完了パケットがあります。

一般的な C2H CMPT パケットは 1 ビートです。

- `s_axis_c2h_cmpt_ctrl_qid`: パケットの完了キュー ID。これは C2H ディスクリプター QID とは異なります。
- `s_axis_c2h_cmpt_ctrl_cmpt_type = HAS_PLD`。この完了パケットには対応するデータ パケットがありません。
- `s_axis_c2h_cmpt_ctrl_wait_pld_pkt_id`: この完了パケットは、CMPT パケットの送信前に、この ID のあるデータ パケットの送信を待つ必要があります。

ユーザー アプリケーションがデータ パケットを送信する場合は、各パケットのパケット ID をカウントする必要があります。最初のデータ パケットのパケット ID は 1 で、データ パケットごとにインクリメントしていきます。

一般的な C2H パケットの場合は、データ パケットと完了パケットが 1 対 1 で一致しています。つまり、`s_axis_c2h_ctrl_has_cmpt` が `1'b1` のデータ パケットの数は、`s_axis_c2h_cmpt_ctrl_cmpt_type` が `HAS_PLD` の CMPT パケットの数と等しくなる必要があります。

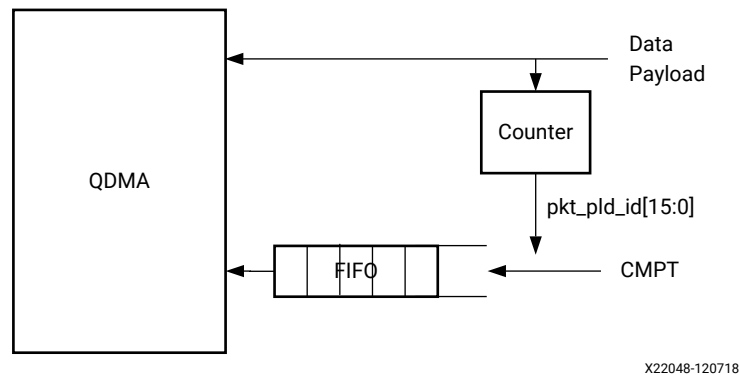
QDMA Subsystem for PCIe には、深さが 2 の浅い完了入力 FIFO があります。パフォーマンスを改善するには、次の図に示すように、完了入力用に FIFO を追加します。この FIFO の深さおよび幅はユース ケースによって異なります。幅はアプリケーションの最大 CMPT サイズに、また深さはパフォーマンス ニーズにそれぞれ依存しています。64 バイトの CMPT でベスト パフォーマンスを得るには、深さを 512 に設定することを推奨します。

ユーザー アプリケーションがデータ ペイロードを送信するときは、どのパケットもカウントされます。最初のパケットの `pkt_pld_id` は 1、次のパケットの `pkt_pld_id` は 2 のようになります。これは折り返し型の 16 ビットのカウンターです。

CMPT タイプはユーザー アプリケーションで定義されます。

- `s_axis_c2h_cmpt_ctrl_cmpt_type` が `HAS_PLD` の場合、CMPT にはそれに対応するデータ ペイロードがあるので、ユーザー アプリケーションは、そのパケットの `pkt_pld_id` を `s_axis_c2h_cmpt_ctrl_wait_pld_pkt_id` フィールドに配置する必要があります。DMA は、対応するデータ ペイロード パケットを送信した後でないと、この CMPT を送信しません。
- `s_axis_c2h_cmpt_ctrl_cmpt_type` が `NO_PLD_NO_WAIT` の場合、CMPT にはデータ ペイロードがまったくなく、ペイロードを待つ必要もないので、この CMPT を送信します。
- `s_axis_c2h_cmpt_ctrl_cmpt_type` が `NO_PLD_BUT_WAIT` の場合、CMPT にはそれに対応するデータ ペイロード パケットがないので、CMPT は送信されるまで、特定のデータ ペイロード パケットを待つ必要があります。そのため、ユーザー アプリケーションはそのデータ ペイロードの `pld_pkt_id` を `s_axis_c2h_cmpt_ctrl_wait_pld_pkt_id` フィールドに配置する必要があります。DMA は、その `pld_pkt_id` のあるデータ ペイロードが送信されるまで、CMPT を送信しません。

図 10: CMPT 入力 FIFO



即値データ パケット

ユーザー アプリケーションには、対応するデータ パケットをホストに転送せずに、完了リングに書き込むだけのパケットがあります。このタイプのパケットは即値データ パケットと呼ばれます。即値データ パケットの場合、QDMA はデータ ペイロードを送信しませんが、CMPT キューに書き込みをします。即値パケットではディスクリプターは使用されません。

即値データ パケットの場合、ユーザー アプリケーションは CMPT パケットを DMA に送信するだけで、データ パケットは送信しません。

次に示すのは、即値完了パケットの設定方法です。対応するデータ パケットはありません。

一部のアプリケーションでは、即値完了パケットはデータ パケットを待つ必要がありませんが、それを待つ必要のあるアプリケーションもある可能性があります。完了タイプが `NO_PLD_NO_WAIT` であれば、完了パケットは、データ パケットを待たずに送信できますが、完了タイプが `NO_PLD_BUT_WAIT` だと、完了パケットが待つ必要のあるデータ パケットの ID を指定する必要があります。

- `s_axis_c2h_cmpt_user_cmpt_type = NO_PLD_NO_WAIT` または `NO_PLD_BUT_WAIT`。
- `s_axis_c2h_cmpt_ctrl_wait_pld_pkt_id`: パケット カウントをインクリメントしない。

マーカー パケット

QDMA の C2H ストリーム エンジンには、ユーザー アプリケーションが C2H パケットと共にマーカーを QDMA に挿入できるようにします。マーカーが挿入されると、C2H エンジンのパイプラインを通過して、C2H ストリーム ディスクリプター バイパス出力インターフェイスに出力されます。マーカーは、C2H ストリーム パケットでマーカー ビットをセットすると挿入されます。マーカー応答は、QDMA により C2H ストリーム ディスクリプター バイパス出力インターフェイスで `mrkr_rsp` ビットをセットすることにより、ユーザー アプリケーションに送られます。マーカー パケットの場合、QDMA はペイロード パケットを送信しませんが、完了リングへの書き込みは実行します。マーカー リクエストがあったからといって、必ずしもマーカー応答が生成されると限りません。QDMA は例外的なイベントに遭遇すると、マーカー応答を生成することがあります。QDMA がいつ内部的にマーカー応答を生成するかについては、次のセクションを参照してください。

ユーザー アプリケーションから QDMA にマーカーを送信できるようにしているのは、マーカーに先行するトラフィックがすべてフラッシュされたタイミングを判断するのが主な目的です。これはユーザー アプリケーションのシャットダウン シーケンスで使用できます。また、必須ではありませんが、QDMA にマーカーを送信するときに、`user_trig` ビットをセットされたユーザー アプリケーションからマーカーを送信することも可能です。これにより QDMA が割り込みを生成できるようになり、マーカーに先行するすべてのトラフィックを確実にフラッシュできます。QDMA 完了エンジンは、ユーザー アプリケーションからマーカーを受信すると、次を実行します。

- マーカーと共に受信した完了を C2H ストリーム完了リングに送信します。

- イネーブルになっている (マーカーが挿入されたときに `user_trig` がセットされている) 場合は、ステータス ディスクリプターを生成します。
- イネーブルになっていて、未処理でない場合は、割り込みを生成します。
- マーカー応答を送信します。イネーブルになっているのに未処理だったためマーカー応答が送信されなかった場合は、マーカー応答の `retry_mrkr` ビットをセットして、このマーカー リクエストに対して割り込みを送信できなかったことを通知します。これらのフィールドの詳細は、C2H ストリーム ディスクリプター バイパス出力インターフェイスの説明を参照してください。

マーカー パケットにはデータ パケットと CMPT パケットの両方があります。この 2 つは対になっています。

マーカーのあるデータ パケットの設定は次のようになります。

- 1 ビートのデータ
- `s_axis_c2h_ctrl_marker = 1'b1`
- `s_axis_c2h_ctrl_len`: データ幅 (例: データ幅が 512 ビットの場合は 64)
- `s_axis_c2h_mty = 0`
- `s_axis_c2h_ctrl_has_cmpt = 1'b1`

マーカーのある CMPT パケットの設定は次のようになります。

- 1 ビートの CMPT パケット
- `s_axis_c2h_cmpt_ctrl_marker = 1'b1`
- `s_axis_c2h_cmpt_ctrl_cmpt_type = HAS_PLD`
- `s_axis_c2h_cmpt_ctrl_wait_pld_pkt_id`: CMPT パケット送信前に、この ID のデータ ペイロード パケットを待つ必要があります。

即値データ パケットおよびマーカー パケットは、ディスクリプターを使用せず、代わりに C2H 完了リングに書き込みます。未処理の即値パケットおよびマーカー パケットに合わせて、ソフトウェアは C2H 完了リングのサイズを設定する必要があります。

長さ 0 のパケット

データ パケットの長さは 0 にできます。入力で、1 ビートのデータを送信する必要があります。長さが 0 のパケットはディスクリプターを使用します。QDMA が 1 DW のペイロード データを送信します。

長さ 0 のパケットの設定方法は次のようになります。

- 1 ビートのデータ
- `s_axis_c2h_ctrl_len = 0`
- `s_axis_c2h_mty = 0`

完了パケットのディスエーブル

ユーザー アプリケーションは、特定のパケットの完了をディスエーブルにできます。QDMA はペイロードに DMA (直接メモリアクセス) を提供しますが、C2H 完了リングへは書き込みをしません。ユーザー アプリケーションは、DMA にデータ パケットのみを送信し、CMPT パケットは送信しません。

ディスエーブル完了パケットの設定は次のようになります。

- `s_axis_c2h_ctrl_has_cmpt = 1'b0`

完了エンジン

完了エンジンは、C2H AXI4-Stream 完了 (CMPT) を CMPT キューに書き込みます。ユーザー アプリケーションは、CMPT パケットだけでなく、CMPT QID、CMPT_TYPE などの情報を QDMA Subsystem for PCIe に送信します。QDMA は、この情報を使用して CMPT パケットを処理します。CMPT キューで変更されていない CMPT パケットを書き込むように指示できます。または、CMPT キューに書き込む前に、CMPT パケットにエラーやカラーなどのフィールドを挿入するようにユーザー アプリケーションから指示できます。さらに、CMPT インターフェイス信号を使用して、C2H データ入力のトラフィックに相対して、CMPT パケットを特定の順序で書き込むように指示することもできます。必須ではありませんが、CMPT は通常 C2H キューと共に使用されます。その場合、一定数の C2H ディスクリプターが C2H データの DMA によって使用されたことをソフトウェアに通知するため、CMPT が使用されます。これにより、ソフトウェアでディスクリプターが再び使用可能になります。CMPT は対応する C2H DMA 操作なしに使用することもでき、この場合は即値データとなります。

CMPT パケットのユーザー定義部分には、通常、転送されたデータ パケットの長さを指定する必要があり、また、データ パケット転送の結果、ディスクリプターが消費されたかどうかを指定する必要があります。パケットが即値およびマーカ タイプの場合は、どのディスクリプターも消費されません。ユーザー定義のデータの正確な内容は、ユーザーがどのように定義しているかで異なります。

完了コンテキスト構造

完了コンテキストは完了エンジンによって使用されます。

表 15: 完了コンテキスト構造の定義

ビット	ビット幅	フィールド名	説明
[159:143]	17		予約。0 に初期化されます。
[143]	1	int_aggr	割り込みアグリゲーション QID を割り込みアグリゲーション モードに設定します。
[142:132]	11	vec	割り込みベクター
131	1	at	アドレス変換 キュー アドレスが変換されているか、未変換かを判断するために使用されます。この情報は、CMPT の PCIe およびステータス書き込みに送られます。 0: アドレスが変換されていない 1: アドレスが変換されている
130	1	ovf_chk_dis	完了リング オーバーフロー チェック ディスエーブル 完了リングの完了エントリの書き込みによりリングがオーバーフローしたかどうかを CMPT エンジンでチェックしません。QDMA は、完了リングがオーバーフローするかどうかをチェックせずに完了を送信し、オーバーフローが発生しても何も対処しません。完了リングでオーバーフローが発生しないようにするのは、ソフトウェアおよびユーザー ロジックの役割になります。

表 15: 完了コンテキスト構造の定義 (続き)

ビット	ビット幅	フィールド名	説明
[129]	1	full_upd	フル アップデート セットされていない場合、完了/CIDX/アップデートの CIDX 以外のすべてのフィールドが無視されます。CIDX フィールドのみがアップデートからコンテキストにコピーされます。 セットされている場合、完了 CIDX アップデートがこのコンテキストで次のフィールドをアップデートします。 timer_ix counter_ix trig_mode en_int en_stat_desc
[128]	1	timer_running	このキューでタイマーが実行されていることを示します。このタイマーは CMTP 割り込みを調整する目的で使用されます。理想的には、キューを閉じる前にこの QID のキューでタイマーが実行されていないことをソフトウェアで確認する必要があります。これは内部的にハードウェアによって使用されるフィールドで、ソフトウェアにより 0 に初期化され、読み出し専用として処理されます。
[127]	1	user_trig_pend	ユーザー ロジックが開始した割り込みの生成が保留中であることを示します。ユーザー ロジックは s_axis_c2h_cmpt_ctrl_user_trig 信号を介して割り込みをリクエストできます。このビットは、ほかの割り込みが既にこの QID で保留になっているときに、ユーザー ロジックが割り込みをリクエストするとセットされます。QDMA で次の完了 CIDX アップデートが受信されると、完了リングのエントリで読み出されるのを待機しているエントリがあるかどうかによって、この保留中のビットが割り込みを生成する可能性があります。これは内部的にハードウェアによって使用されるフィールドで、ソフトウェアにより 0 に初期化され、読み出し専用として処理されます。
[126:125]	2	err	完了コンテキストでエラーが発生していることを示します。これはハードウェアによって書き込まれるフィールドで、ソフトウェアにより 0 に初期化され、読み出し専用として処理されます。次のエラーが示されます。 0: エラーなし。 1: ソフトウェアからの無効な CIDX アップデートを検出。 2: ディスクリプター エラーを検出。 3: 完了リングがフルだったためユーザー ロジックにより完了パケットが送信された。
[124]	1	valid	コンテキストが有効です。
[123:108]	16	cidx	完了リング コンシューマー インデックスのハードウェアコピーの現在値。
[107:92]	16	pidx	完了リング プロデューサー インデックス。これはハードウェアによって書き込まれるフィールドで、ソフトウェアにより 0 に初期化され、読み出し専用として処理されます。
[91:90]	2	desc_size	完了エントリ サイズ: 0: 8B 1: 16B 2: 32B 3: 64B
[89:38]	52	baddr	完了リングの 4K に揃えられたベース アドレス - ビット [63:12]。
[37:32]	6		予約。0 に初期化されます。

表 15: 完了コンテキスト構造の定義 (続き)

ビット	ビット幅	フィールド名	説明
[31:28]	4	qsize_idx	リング サイズ レジスタへの完了リング サイズ インデックス。
[27]	1	color	完了で使用されるカラー ビット。
[26:25]	2	int_st	割り込みステート: 0: ISR 1: TRIG これは内部的にハードウェアによって使用されるフィールドで、ソフトウェアにより 0 に初期化され、読み出し専用として処理されます。 リセットが解除されると、ハードウェアが ISR ステートに初期化し、トリガー イベントには反応しません。ソフトウェアに割り込みまたはステータス書き込みが必要な場合は、ソフトウェアが初期完了 CIDX アップデートを送信する必要があります。そうすると、ハードウェアが TRIG ステートに遷移するので、トリガー条件に反応するようになります。
[24:21]	4	timer_idx	タイマー ベースのトリガー モードのタイマー レジスタのインデックス。
[20:17]	4	counter_idx	カウント ベースのトリガー モードのカウンター レジスタのインデックス。
[16:13]	4		予約。0 に初期化されます。
[12:5]	8	fnc_id	ファンクション ID
[4:2]	3	trig_mode	割り込みおよび完了ステータス書き込みのトリガー モード: 0x0: 無効 0x1: すべて 0x2: ユーザー カウント 0x3: ユーザー 0x4: ユーザー タイマー 0x5: ユーザー タイマー カウント
[1]	1	en_int	完了割り込みをイネーブルにします。
[0]	1	en_stat_desc	完了ステータス書き込みをイネーブルにします。

完了ステータス構造

完了ステータスは、完了リングの最終位置、すなわち完了リング ベース アドレス + (完了の長さ (8、16、32) * (完了リング サイズ - 1)) にあります。

QDMA Subsystem for PCIe で完了ステータスが完了リングに書き込まれるようにするには、完了コンテキストで完了ステータスをイネーブルにする必要があります。割り込みに影響するだけでなく、完了コンテキストで定義されているトリガー モードは、完了ステータスの書き込みも制御します。割り込み/ステータスの制御によって、完了ステータスは、次のいずれかの状況が発生したときに書き込まれます。

1. CMPT パケットが完了リングに書き込まれた。
2. ソフトウェアから CMPT-CIDX アップデートが受信され、さらに読み出し待機状態の完了エントリがあることが示されている。
3. 該当する CMPT QID に関連付けられているタイマーが時間切れになり、タイマー ベースのトリガー モードでプログラムされている。

表 16: AXI4-Stream 完了ステータス構造

ビット	ビット幅	フィールド名	説明
[63:35]	29		予約
[34:33]	2	int_state	割り込みステート。 0: ISR 1: TRIG
[32]	1	color	カラー ステータス ビット
[31:16]	16	cidx	コンシューマー インデックス (RO)
[15:0]	16	pidx	プロデューサー インデックス

完了エン트리構造

完了 (CMPT) リング エントリのサイズは 512 ビットです。ユーザー定義のデータ、オプションのエラー ビット、オプションのカラー ビットがこれに含まれます。ユーザー定義のデータには、8B、16B、32B、および 64B の 4 つのオプションがあります。CMPT エントリのオプションのエラー ビットおよびカラー ビットのビット位置は、個別に設定可能です。これらのフィールドの位置は、QDMA Subsystem for PCIe のコンパイル中に、Vivado® IDE の IP カスタマイズ オプションを使用して指定できます。カラー ビットの位置オプションは 7 つ、エラー ビットの位置オプションは 8 つあります。位置は、完了エントリの LSB ビットからのオフセットとして指定されます。

ユーザー アプリケーションが完了パケットを QDMA Subsystem for PCIe に駆動すると、インターフェイスでの `s_axis_cmpt_ctrl_col_idx[2:0]` の値および `s_axis_cmpt_ctrl_err_idx[2:0]` の値が供給されます。これらのインデックスは、カラー ビットおよびエラー ビットの正しい位置を使用するため、QDMA Subsystem for PCIe で使用されます。たとえば、`s_axis_cmpt_ctrl_col_idx[2:0] = 0` および `s_axis_cmpt_ctrl_err_idx[2:0] = 1` の場合、QDMA Subsystem for PCIe では、カラー位置には [C2H Stream Completion Color bits] の位置オプション 0 が、エラー位置には [C2H Stream Completion Error bits] の位置オプション 1 が使用されます。カラー信号またはエラー信号の 7 つのインデックスは、完了エントリがアップデートされたとき、対応するカラー ビットまたはエラー ビットが DMA でアップデートされないことを示しています (これらのフィールドは無視される)。Vivado® IDE の C2H ストリーム完了ビットのオプションについては、[\[PCIe DMA\] タブ](#) を参照してください。

コンパイル時に使用されるエラー ビットおよびカラー ビットの位置の値は、MMIO レジスタからソフトウェアによって読み出されます。このために `QDMA_C2H_CMPT_FORMAT_0 (0xBC4)` から `QDMA_GLBL_ERR_MASK (0X24C)` までの 7 つのレジスタがあります。これらのレジスタはそれぞれ、カラー ビット位置を 1 つ、エラー ビット位置を 1 つ保持します。

- C2H ストリーム完了ビットのカラー ビット位置用のオプション 0 およびエラー ビット位置用のオプション 0 は、`QDMA_C2H_CMPT_FORMAT_0` レジスタにあります。
- カラー ビット位置用のオプション 1 およびエラー ビット位置用のオプション 1 は、`QDMA_C2H_CMPT_FORMAT_1` レジスタにある、というようになっています。
-

表 17: 完了エン트리構造

名前	サイズ (ビット)	インデックス
ユーザー定義ビット (64 バイト設定の場合)	510 ~ 512	カラー ビットおよびエラー ビットがあるかどうかによります。
ユーザー定義ビット (32 バイト設定の場合)	254 ~ 256	カラー ビットおよびエラー ビットがあるかどうかによります。
ユーザー定義ビット (16 バイト設定の場合)	126 ~ 128	カラー ビットおよびエラー ビットがあるかどうかによります。

表 17: 完了エン트리構造 (続き)

名前	サイズ (ビット)	インデックス
ユーザー定義ビット (8 バイト設定の場合)	62 ~ 64	カラー ビットおよびエラー ビットがあるかどうかによります。
Err	0 ~ 1	ビット位置はレジスタにより定義されます。エラー ビット (インデックス値 7) を含めないように選択できます。その場合、ユーザー定義データがその空間を占めます。
Color	0 ~ 1	ビット位置はレジスタにより定義されます。カラー ビット (インデックス値 7) を含めないように選択できます。その場合、ユーザー定義データがその空間を占めます。

完了入力パケット

CMPT パケットはユーザー アプリケーションによって QMDA に送信されます。

このパケットとデータ パケットが 1 対 1 で一致している必要はありません。たとえば、即値データ パケットには CMPT パケットだけでデータ パケットはなく、ディスエーブル完了パケットにはデータ パケットだけで CMPT パケットはありません。

各 CMPT パケットには CMPT ID が 1 つあります。これは関連付けられている CMPT キューの ID です。各 CMPT キューには CMPT コンテキストが 1 つあります。C2H ディスクリプター キューの CMPT キューへのマップは、ドライバーによって設定されます。同じ CMPT キューに複数の C2H ディスクリプターがマップされることもあります。その場合、同じ CMPT ID が複数のディスクリプター間で共有されます。また、C2H キューには関連付けられていない CMPT キューがある場合もあります。

ユーザー アプリケーションからの CMPT パケットは次のようになります。

表 18: CMPT 入力パケット

名前	サイズ	インデックス
データ	512 ビット	[511:0]

この CMPT パケットには、8B、16B、32B、64B の 4 タイプがあります。データ サイズは 512 ビットのみです。

完了ステータス/割り込み調整

QDMA Subsystem for PCIe には、キューごとに完了割り込みおよび完了ステータスを調整するメカニズムがあり、ソフトウェアで各キューに 5 つのモードから 1 つを選択できるようになっています。選択されたモードは、そのキューの完了リング コンテキストの QDMA Subsystem for PCIe に保存されます。キューにモードを選択した後も、QDMA に完了リング CIDX アップデートを送信するときに、ドライバーで常に別のモードを選択できます。

完了割り込みの調整は完了エンジンで実行されます。このエンジンには、すべてのキューの完了リング コンテキストが格納されます。割り込みおよび完了ステータスの送信は、キューごとに個別にオン/オフを切り替えることができ、この情報は完了リング コンテキストに保存されています。また、ここで説明されているモードでは、割り込みだけでなく完了ステータス書き込みも調整できます。さらに、割り込みも完了ステータス書き込みもキューごとに個別にオン/オフを切り替えることができるので、これらのモードは、そのキューの完了コンテキストで割り込み/完了ステータスがイネーブルになっている場合にのみ機能します。

QDMA Subsystem for PCIe は、未処理の割り込みをキューごとに 1 つだけ保持します。選択されているモードに対して、割り込み送信のほかのすべての条件が満たされていても、QDMA はこの制限が適用されます。QDMA Subsystem for PCIe は、ドライバーからキューの CIDX アップデートを受信すると、割り込みが実行されたと認識します。

割り込み調整モードにどのモードを選択していても、キューに未処理の割り込みがなければ、QDMA Subsystem for PCIe はそのモードで満たされるべきトリガー条件を監視し続けます。これらの条件が満たされると、割り込みが送信されます。QDMA Subsystem は、割り込み実行を待機している間、割り込み条件が満たされているかどうかを継続的にチェックし、その条件を記憶します。CIDX アップデートを受信すると、QDMA Subsystem はその条件がまだ満たされているかどうかを評価し、条件が満たされていれば、別の割り込みが送信されます。満たされていなければ、割り込みは送信されず、QDMA は再び満たされるべき条件の監視を再開します。

QDMA Subsystem の割り込み調整モードは、必ずしも正確ではありません。ユーザー アプリケーションから割り込みの送信を示す 2 つの CMPT パケットが送信されても、2 つの割り込みが生成されるとは限りません。この主な理由は、ドライバーに完了リングを読み出すよう割り込みが送信されても、ドライバーには割り込みが生成された完了までを読み出す義務はないからです。そのため、ドライバーが割り込みの完了まで読み出さないか、読み出し可能な有効なディスクリプターがあれば、割り込みの完了ディスクリプターを超えて読み出す可能性もあります。この場合、QDMA Subsystem for PCIe はドライバーから CIDX アップデートを受信するたびに、トリガー条件を再評価する必要があります。

各モードの詳細は、次のとおりです。

- TRIGGER_EVERY: 割り込みの頻度が最も高いモードです。完了エンジンが完了リングを読み出されていない完了ディスクリプターがあると判断すると、割り込みが送信されます。
- TRIGGER_USER: QDMA Subsystem for PCIe では、サブシステムがホストへのパケット送信を終了したときに割り込みを送信するように示して、サブシステムに CMPT パケットを送信する方法があります。TRIGGER_USER モードでは、ユーザー アプリケーションが割り込み調整を実行します。
- TRIGGER_USER_COUNT: このモードでは、QDMA Subsystem for PCIe は 2 つのトリガーのいずれかに応答します。2 つのトリガーの 1 つは、ユーザー アプリケーションによって CMPT パケットと共に送信されます。もう 1 つのトリガーは、完了リングの読み出されていない完了エントリの数が、プログラムされているしきい値を超えていることをハードウェアが認識した場合です。このしきい値はキューごとにドライバーでプログラムできます。いずれかのトリガーが検出されると、QDMA は割り込みを送信するかどうかを評価します。前のセクションで説明したように、割り込み送信のトリガーに加え、その他の条件が満たされている必要があります。
- TRIGGER_USER_TIMER: このモードでは、QDMA Subsystem for PCIe は 2 つのトリガーのいずれかに応答します。2 つのトリガーの 1 つは、ユーザー アプリケーションによって CMPT パケットと共に送信されます。もう 1 つのトリガーは、CMPT キューに関連付けられているタイマーの時間切れです。タイマーの時間はキューごとにドライバーでプログラムできます。いずれかのトリガーが検出されると、QDMA は割り込みを送信するかどうかを評価します。前のセクションで説明したように、割り込み送信のトリガーに加え、その他の条件が満たされている必要があります。詳細は、[完了タイマー](#) を参照してください。
- TRIGGER_USER_TIMER_COUNT: このモードでは、QDMA Subsystem for PCIe は 3 つのトリガーすべてに応答します。最初のトリガーは、ユーザー アプリケーションによって CMPT パケットと共に送信されます。2 番目のトリガーは、CMPT キューに関連付けられているタイマーの時間切れです。タイマーの時間はキューごとにドライバーでプログラムできます。3 番目のトリガーは、完了リングの読み出されていない完了エントリの数が、プログラムされているしきい値を超えていることをハードウェアが認識した場合です。このしきい値はキューごとにドライバーでプログラムできます。これらのトリガーのいずれかが検出されると、QDMA は割り込みを送信するかどうかを評価します。前のセクションで説明したように、割り込み送信のトリガーに加え、その他の条件が満たされている必要があります。
- TRIGGER_DIS: このモードでは、キューに対して完了割り込みがイネーブルになっていても、QDMA Subsystem for PCIe は完了割り込みを送信しません。この場合、ドライバーが完了リングを読み出すことができるのは、リングを定期的にポーリングするときだけです。このモードに設定されていると、完了リングへの完了ステータスディスクリプターの送信もディスエーブルになるので、完了リングで提供されているカラービットの機能をドライバーで利用する必要があります。

TRIGGER_USER_TIMER_COUNT モードでキューがプログラムされている場合、ソフトウェアは、割り込み (または完了ステータス書き込み) で示されているように、完了リングにある完了エントリを全部読み出さない可能性があります。その場合は、ソフトウェアが部分的な読み出しを実行して完了 CIDX アップデートを送信します。この CIDX アップデートを受信し、タイマーが時間切れになると、QDMA はタイマーを再開し、別の割り込みが生成されます。このプロセスは、完了エントリがすべて読み出されるまで、繰り返し実行されます。

TRIGGER_EVERY、TRIGGER_USER、および TRIGGER_USER_COUNT モードでは、ユーザー ロジックから QDMA が完了パケットを受信した結果、割り込みが送信されます。ユーザー ロジックが割り込み送信をリクエストするたびに、QDMA は割り込みを 1 つだけ送信します。したがって、ソフトウェアが読み出しできる完了エントリを全部読み出さず、ユーザー ロジックも割り込みをリクエストする完了をそれ以上送らない場合は、QDMA もそれ以上割り込みを生成しません。結果的に、完了リングに完了が無期限に残ることになります。これを避けるには、TRIGGER_EVERY、TRIGGER_USER、および TRIGGER_USER_COUNT モードのときに、割り込み (または完了ステータス書き込み) で示されているように、完了リングの完了エントリをすべて読み出す必要があります。

次に、各モードのフローチャートを示します。これらのフローチャートは、完了エンジンの視点から描かれています。完了パケットは、ユーザー ロジックから送信され、完了リングに書き込まれます。ソフトウェア アップデートは、ソフトウェアからハードウェアに送信された完了リング CIDX アップデートを参照します。

図 11: EVERY モードのフローチャート

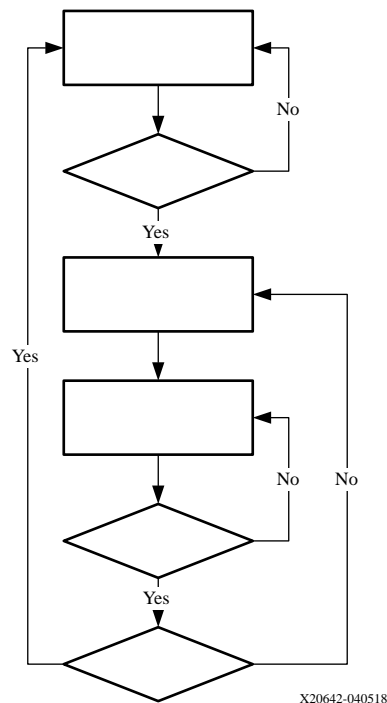
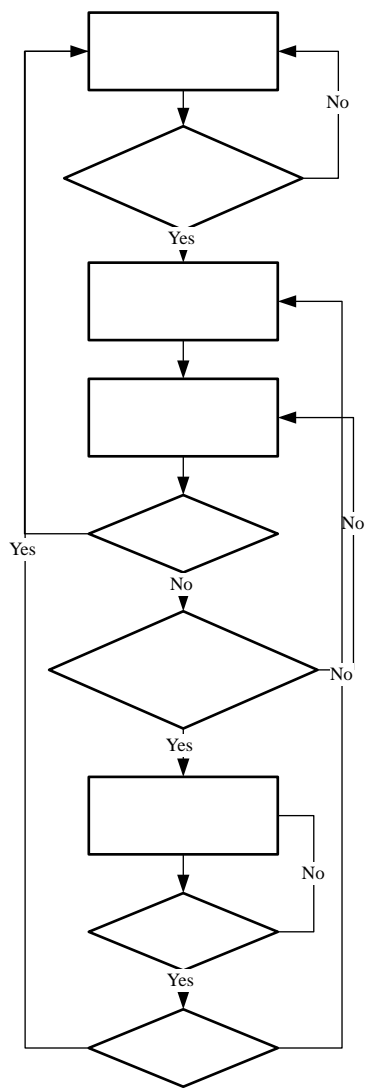
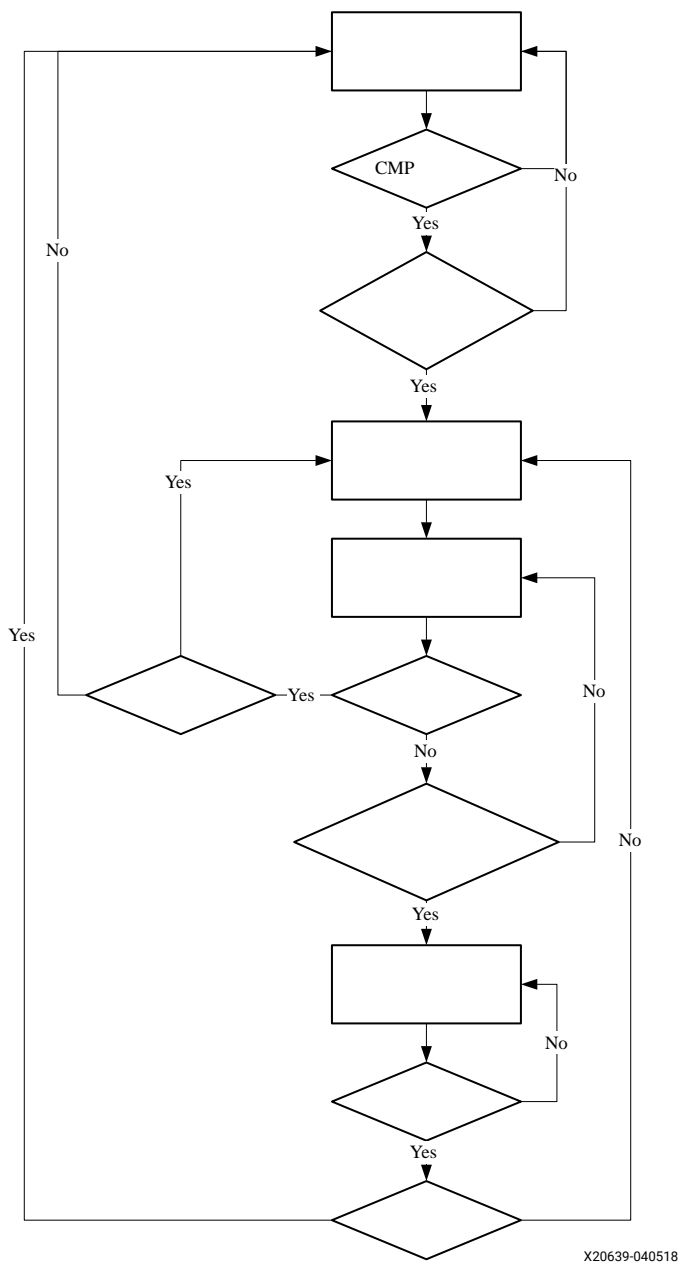


図 12: USER モードのフローチャート



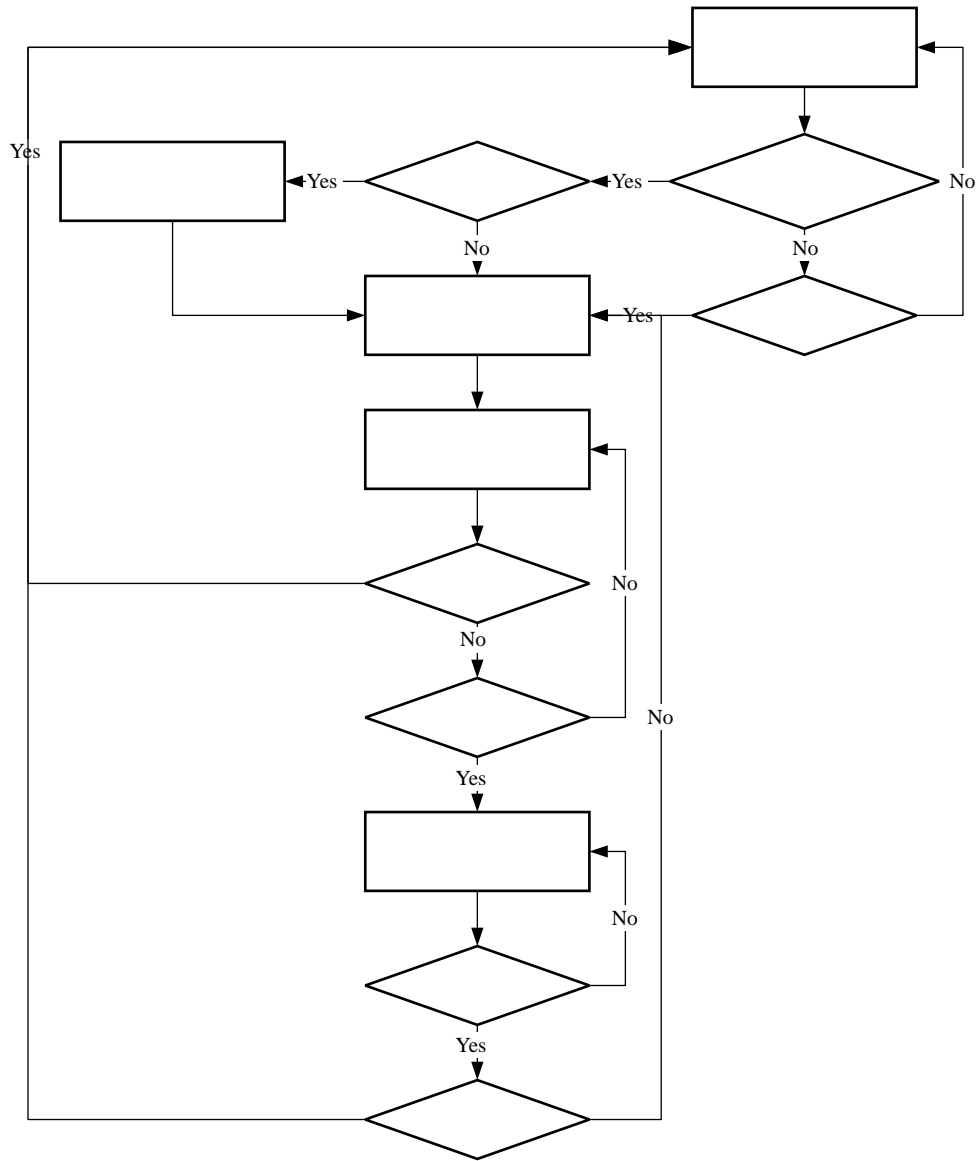
X20641-040518

図 13: USER_COUNT モードのフローチャート



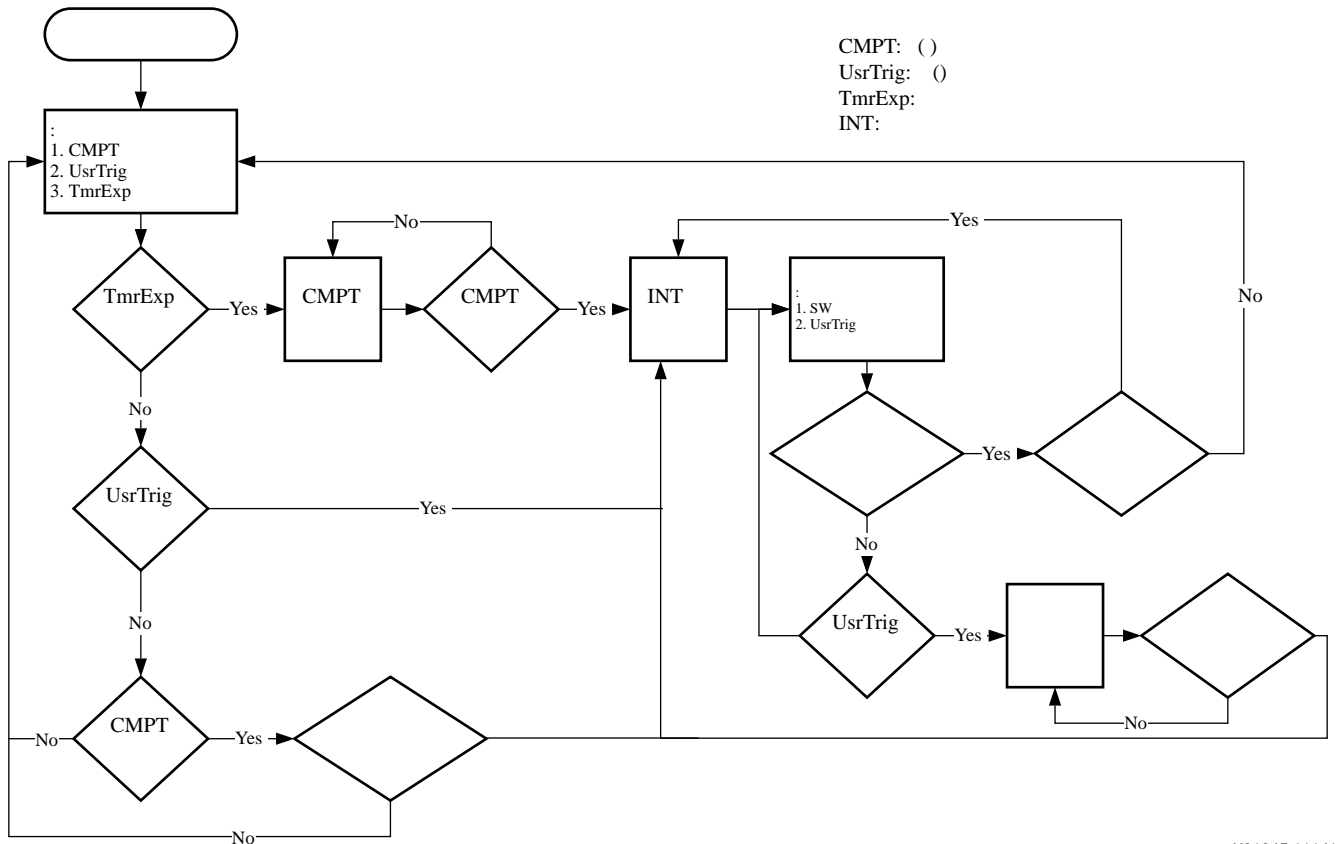
X20639-040518

図 14: USER_TIMER モードのフローチャート



X20637-040518

図 15: USER_TIMER_COUNT モードのフローチャート



X21845-111418

完了タイマー

完了タイマー エンジンでは、完了コンテキストでタイマー トリガー モードがサポートされます。2048 個のキューがサポートされ、各キューにタイマーがあります。タイマーが期限切れになると、タイマー期限切れ信号が完了モジュールに送信されます。同時に複数のタイマーが切れると、ラウンドロビン方式で信号が送信されます。

基準タイマー

基準タイマーはタイマーの刻みに基づいています。レジスタ QDMA_C2H_INT (0xB0C) は、タイマーの刻みを定義します。QDMA_C2H_TIMER_CNT (0xA00-0xA3c) の 16 個のレジスタには、このタイマーの刻みに基づいたタイマーカウント値が出力されます。完了コンテキストの timer_idx は、16 個の QDMA_C2H_TIMER_CNT レジスタへのインデックスです。timer_idx は各キューで選択できます。

例外イベントの処理

無効キューでの完了

CMPT コンテキストの有効ビットで示されているように、コンテキストが無効なキューで QDMA Subsystem for PCIe が完了を受信すると、その完了は何の通知もなく破棄されます。

フルリングでの完了

完了リングの完了エントリの最大数は、完了リングのエントリ合計から 2 を差し引いた数になります。完了コンテキストには PIDX および CIDX があるので、完了リングの完了数を QDMA で計算できます。フル状態のキューで QDMA が完了を受信すると、QDMA は次の操作を実行します。

- そのキューの完了コンテキストを無効にします。
- 完了コンテキストをエラーとマークします。
- 完了を破棄します。
- 完了ステータスがイネーブルの場合は、完了ステータスをエラーをマークして送信します。
- 割り込みがイネーブルで未処理でない場合は、割り込みを送信します。
- エラーのマーカー応答を送信します。
- C2H エラー ステータス レジスタにエラーを記録します。

ディスクリプター エラーのある完了

QDMA C2H エンジンでディスクリプター エラーが発生すると、完了エンジンのコンテキストで次の操作が実行されます。

- そのキューの完了コンテキストを無効にします。
- 完了コンテキストをエラーとマークします。
- 完了出力を完了リングに送信します。完了出力はエラーとマークされています。
- 完了ステータスがイネーブルの場合は、完了ステータスをエラーをマークして送信します。
- 割り込みがイネーブルで未処理でない場合は、割り込みを送信します。
- エラーのマーカー応答を送信します。

無効 CIDX のある完了

完了エンジンには、CIDX アップデートの CIDX 値が完了リングの空の位置にポイントしていることを検出するロジックがあります。このようなエラーが検出されると、完了エンジンは次の操作を実行します。

- 完了コンテキストを無効にします。
- 完了コンテキストをエラーとマークします。
- C2H エラー ステータス レジスタにエラーを記録します。

ブリッジ

ブリッジ コアは、AXI4 および PCI Express 統合ブロックの間のインターフェイスで、メモリ マップド AXI4 から AXI4-Stream へのブリッジ、PCIe の AXI4-Stream 拡張インターフェイス ブロックが含まれます。メモリ マップド AXI4 から AXI4-Stream へのブリッジには、レジスタ ブロックが 1 つと、スレーブ ブリッジおよびマスター ブリッジと呼ばれるハーフ ブリッジが 2 つあります。

- スレーブ ブリッジは、出力された AXI4 マスター読み出しまたは書き込みのリクエストを処理するため、スレーブ デバイスとして AXI4 インターコネクに接続します。
- マスター ブリッジは、PCIe 生成された読み出しまたは書き込みの TLP を処理するため、マスターとして AXI4 インターコネクに接続します。

- レジスタ ブロックには、ブリッジ コアで使用されているレジスタが含まれ、PCIe® 範囲のアドレスに AXIBAR パラメーターを使用している場合は、AXI4 メモリ マップド (MM) アドレス範囲をダイナミックにマップします。

コアでは、エラー条件を検出しフラグするための割り込みセットが使用されます。

スレーブ ブリッジは、AXI マスター デバイス (プロセッサなど) からのメモリ マップド AXI4 トランザクションを終了させ、また、AXI4 メモリ マップド アドレス ドメイン内でマップされているアドレスを PCIe のドメイン アドレスに変換するメカニズムを提供します。最大ペイロードサイズの設定によりますが、スレーブ ブリッジへの書き込みトランザクションは、1 つ以上の MemWr TLP に変換され、PCI Express の統合ブロックに渡されます。リモートの AXI マスターがスレーブ ブリッジへの読み出しトランザクションを開始すると、読み出しアドレスおよび修飾子が取り込まれ、MemRd リクエスト TLP がコアに渡され、完了タイムアウトのタイマーが開始します。コアから受信した完了は、保留の読み出しリクエストに関連付けられ、読み出しデータが AXI マスターに返されます。スレーブ ブリッジでは、最大 32 個までの AXI 書き込みリクエスト、32 個までの AXI 読み出しリクエストがサポートされます。

マスター ブリッジは、PCI Express の統合ブロックから受信した PCIe の MemWr および MemRd リクエスト TLP の両方を処理し、PCIe ドメインのアドレス内でマップされているアドレスをメモリ マップド AXI4 アドレス ドメインに変換するメカニズムを提供します。メモリ マップド AXI4 バスのアドレスおよび修飾子を作成するため、各 PCIe MemWr リクエスト TLP ヘッダーが使用され、関連付けられている書き込みデータがアドレス指定されたメモリ マップド AXI4 スレーブに渡されます。マスター ブリッジでは、最大 32 個までのアクティブ PCIe MemWr リクエスト TLP がサポートされます。PCIe の MemWr リクエスト TLP は次のようにサポートされます。

- 64 ビットの AXI データ幅の場合は 4 個
- 128 ビットの AXI データ幅の場合は 8 個
- 256 ビットの AXI データ幅の場合は 16 個
- 512 ビットの AXI データ幅の場合は 32 個

メモリ マップド AXI4 バスのアドレスおよび修飾子を作成するため、各 PCIe MemRd リクエスト TLP ヘッダーが使用され、アドレス指定されたメモリ マップド AXI4 スレーブから読み出しデータが収集され、完了 TLP の生成に使用されます。その後、この完了 TLP は PCI Express の統合ブロックへ渡されます。AXI ブリッジ モードのマスター ブリッジでは、AXI4 パイプライン処理パフォーマンスを改善するため、保留完了のあるアクティブ PCIe MemRd リクエスト TLP が最大 32 個までサポートされます。

割り込み

QDMA Subsystem for PCIe では、最大 2K までの MSI-X ベクターがサポートされています。1 つの MSI-X ベクターを使用して複数のキューをサポートできます。

QDMA では割り込みアグリゲーションがサポートされています。各ベクターには、割り込みアグリゲーション リングが 1 つ関連付けられています。サービスが必要な QID およびステータスは、割り込みアグリゲーション リングに書き込まれます。PCIe® MSI-X 割り込みがホストで受信されると、どのキューにサービスが必要かを判断するため、ソフトウェアで割り込みアグリゲーション リングが読み出されます。キューからベクターへのマップはプログラム可能です。物理ファンクション (PF) ごとに独立したテーブル プログラムがあります。SRIOV でない場合は MSI/MSI-X 割り込みモード、SRIOV の場合は MSI-X 割り込みモードがサポートされます。

非同期割り込みおよびキュー ベース割り込み

QDMA では、非同期割り込みおよびキュー ベースの割り込みの両方がサポートされています。

非同期割り込みは、エラー、ステータス、デバッグ条件など、DMA 操作に同期していないイベントの取り込みに使用されます。PF ごとに 1 つの非同期割り込みがあります。各非同期割り込みは、どの PF に対しても設定可能です。

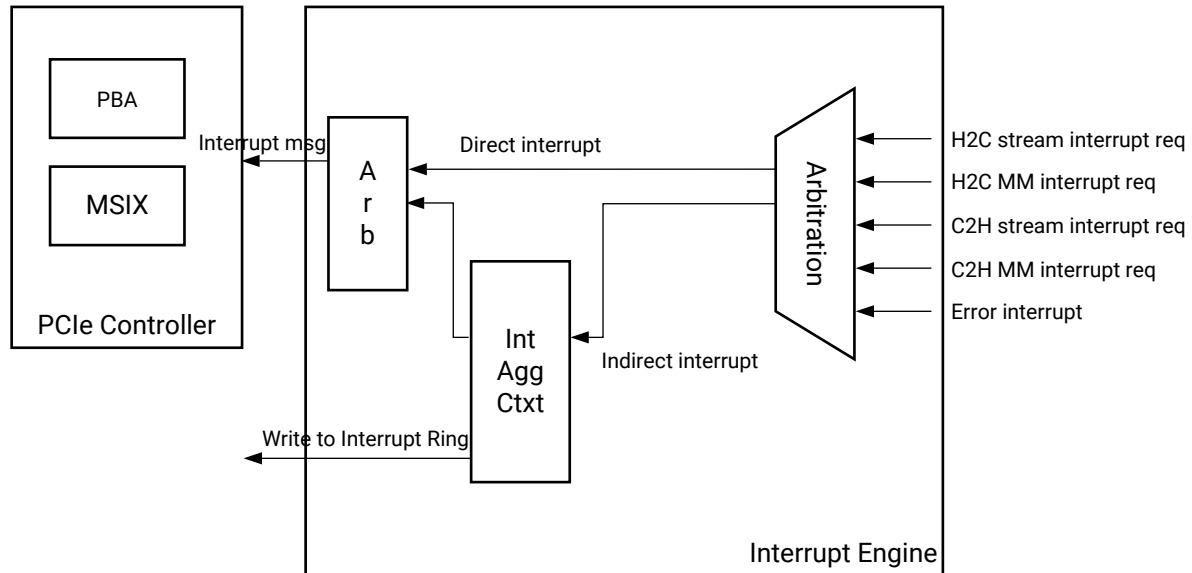
割り込みはすべての PF に通知され、キュー ベースの方式で各 PF のステータスが維持されます。キュー ベースの割り込みには、H2C MM、H2C ストリーム、C2H MM、C2H ストリームからの割り込みが含まれます。

割り込みエンジン

割り込エンジンでは、キューベースの割り込みおよびエラー割り込みが処理されます。

次の図に、割り込みエンジンのブロック図を示します。

図 16: 割り込みエンジンのブロック図



X20891-100818

割り込みエンジンは、H2C MM、H2C ストリーム、C2H MM、C2H ストリーム、またはエラー割り込みから割り込みを得ます。

このエンジンでは、直接割り込み/間接割り込みの 2 つの方法で割り込みが処理されます。割り込みが直接か間接かは、割り込みソースで確認できます。割り込みソースには、ベクターに関する情報もあります。直接割り込みの場合、ベクターは割り込みベクターで、PCIe MSI-X メッセージ (MSI-X テーブルの割り込みベクター `indix`) を生成するために使用されます。間接割り込みの場合、ベクターは割り込みアグリゲーション リングのリング インデックスです。割り込みソースには、割り込みタイプ、ディスクリプター ソフトウェア コンテキストからのベクター、完了コンテキスト、またはエラー割り込みレジスタの情報があります。

直接割り込み

直接割り込みの場合、割り込みはまずソースから割り込みベクターを取得して、それから PCIe MSI-X 割り込みを直接出力します。

割り込みアグリゲーション リング

間接割り込みの場合、割り込みアグリゲーションが実行されます。割り込みアグリゲーションには主に次のような制約があります。

- 各割り込みアグリゲーション リングには 1 ファンクションしか関連付けることができません。ただし、同じファンクションで複数のリングを関連付けることができます。
- 割り込みソースごとにエントリでサポートされるメッセージ数は 3 つまでです。

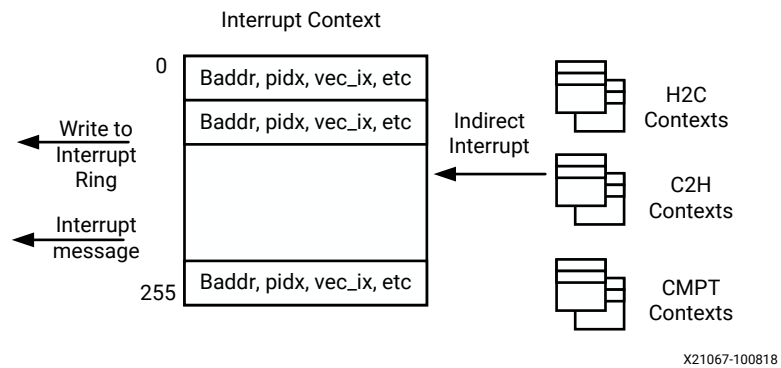
割り込みエンジンは次の手順で間接割り込みを処理します。

- 割り込みソースからアグリゲーション リング インデックスを取得します。

- 割り込みコンテキストを確認します。
- 割り込みアグリゲーション リングに書き込みます。
- PCIe MSI-X メッセージを送信します。

次の図に、間接割り込みのブロック図を示します。

図 17: 間接割り込み



割り込みコンテキストには割り込みアグリゲーション リングの情報が含まれていて、256 個のエントリがあるので、最大 256 個の割り込みアグリゲーション リングをサポートできます。

次は、割り込みコンテキスト構造 (0x8) の詳細です。

表 19: 割り込みコンテキスト構造 (0x8)

信号	ビット	所有者	説明
at	[82]	ドライバー	1'b0: 未変換アドレス 1'b1: 変換されたアドレス
pidx	[81:70]	DMA	プロデューサー インデックス
page_size	[69:67]	ドライバー	割り込みアグリゲーション リング サイズ: 0: 4 KB 1: 8 KB 2: 12 KB 3: 16 KB 4: 20 KB 5: 24 KB 6: 28 KB 7: 32 KB
baddr_4k	[66:15]	ドライバー	割り込みアグリゲーション リングのベース アドレス - ビット [63:12]
color	[14]	DMA	カラー ビット
int_st	[13]	DMA	割り込みステート: 0: WAIT_TRIGGER 1: ISR_RUNNING
Rsvd	[12]	NA	予約
vec	[11:1]	ドライバー	MSI-X テーブルの割り込みベクター インデックス
valid	[0]	ドライバー	有効

ソフトウェアで割り込みアグリゲーション リングのサイズを設定する必要があります。各ソースはリングに最大 3 つまでメッセージを送信できるので、リングのサイズは次の式を満たす必要があります。

エントリの数 $\geq 3 \times$ キューの数

割り込みコンテキストはコンテキスト アクセスによってプログラムされます。QDMA_IND_CTXT_CMD.Qid には、割り込みソースからのリング インデックスがあります。割り込みコンテキストのビットは、MDMA_CTXT_CMD_CLR の操作ですべてクリアできます。MDMA_CTXT_CMD_INV は、有効ビットをクリアできます。

- QDMA_TRQ_SEL_IND を介したコンテキスト アクセス:
 - QDMA_IND_CTXT_CMD.Qid = リング インデックス
 - QDMA_IND_CTXT_CMD.Sel = MDMA_CTXT_SEL_INT_COAL (0x8)
 - QDMA_IND_CTXT_CMD.cmd.Op =
 - MDMA_CTXT_CMD_WR
 - MDMA_CTXT_CMD_RD
 - MDMA_CTXT_CMD_CLR
 - MDMA_CTXT_CMD_INV

割り込みコンテキストを確認後、割り込みアグリゲーション リングに書き込みが実行されます。また、新しい PIDX、カラー、割り込み状態で割り込みコンテキストもアップデートされます。

割り込みアグリゲーション リング エントリ構造は、次のようになります。8B のデータがあります。

表 20: 割り込みアグリゲーション リング エントリ構造

信号	ビット	所有者	説明
Coal_color	[63:63]	DMA	割り込みアグリゲーション リングのカラー ビット。割り込みアグリゲーション リングで PIDX が折り返すたびに反転します。
Qid	[62:39]	DMA	割り込みソースからのキュー ID です。
Int_type	[38:38]	DMA	0: H2C 1: C2H
Rsvd	[37:37]	DMA	予約
Stat_desc	[36:0]	DMA	割り込みソースのステータス ディスクリプターです。

stat_desc の情報は、次のとおりです。

表 21: stat_desc の情報

信号	ビット	所有者	説明
Error	[36:35]	DMA	割り込みソース c2h_err[1:0] または h2c_err[1:0] から供給されます。
Int_st	[34:33]	DMA	割り込みソースから供給される割り込み状態。 0: WRB_INT_ISR 1: WRB_INT_TRIG 2: WRB_INT_ARMED
Color	[32:32]	DMA	割り込みソースから供給されます。PIDX が折り返すたびに反転し、ディスクリプターのカラー フィールドにコピーされます。

表 21: stat_desc の情報 (続き)

信号	ビット	所有者	説明
Cidx	[31:16]	DMA	割り込みソースから供給される累積消費済みポインタ。
Pidx	[15:0]	DMA	割り込みソースから供給されます。書き込まれた割り込みアグリゲーション リング エントリの合計の累積ポインタ。

ソフトウェアが割り込みアグリゲーション リング用にメモリ空間を割り当てると、`coal_color` が 1'b0 で開始するので、ソフトウェアで割り込みコンテキストのカラー ビットを 1'b1 に初期化する必要があります。ハードウェアが割り込みアグリゲーション リングに書き込むと、割り込みコンテキストからカラー ビットを読み出し、それをエントリに書き込みます。リングが折り返すと、ハードウェアは割り込みコンテキストのカラー ビットを反転します。これにより、ソフトウェアが割り込みアグリゲーション リングから読み出すときに、カラー ビットを確認すればどのエントリがハードウェアによって書き込まれたかがわかります。

ソフトウェアは、`Qid` および `int_type` (H2C または C2H) を取得するため、割り込みアグリゲーション リングを読み出します。`Qid` を確認すれば、キューがストリームなのかメモリ マップなのかを特定できます。

割り込みアグリゲーション リングの `stat_desc` は割り込みソースからのステータス ディスクリプターです。ステータス ディスクリプターがディスエーブルの場合、ソフトウェアは割り込みアグリゲーション リングからステータス ディスクリプター情報を取得できます。

その場合、次の 2 つのケースがあります。

- 割り込みソースは C2H ストリーム: C2H 完了リングのステータス ディスクリプターから情報を取得します。ソフトウェアで C2H 完了リングの `pidx` を読み出すことができます。
- 割り込みソースが C2H ストリーム以外 (H2C ストリーム、H2C MM、C2H MM): そのソースのステータス ディスクリプターから情報を取得します。ソフトウェアで `cidx` を読み出すことができます。

最後に、割り込みエンジンは、割り込みコンテキストからの割り込みベクターを使用して、PCIe MSI-X メッセージを送信します。

PCIe MSI-X 割り込みがホストで受信されると、どのキューにサービスが必要かを判断するため、ソフトウェアは割り込みアグリゲーション リングを読み出します。リングを読み出した後、ソフトウェアの読み出し先になる累積ポインタを示すため、ソフトウェア CIDX のダイナミック ポインタ アップデートを実行します。このダイナミック ポインタ アップデートは、`QDMA_DMAP_SEL_INT_CIDX[2048]` (0x18000) レジスタを使用して実行されます。ソフトウェア CIDX が PIDX に等しい場合、そのキューの割り込み状態で割り込みコンテキストに書き込みが実行されます。これは、割り込みアグリゲーション リングにあるエントリがすべてソフトウェアで読み出されたことを QDMA に通知するためです。ソフトウェア CIDX が PIDX に等しくない場合は、別の PCIe MSI-X メッセージが送信されます。そのため、ソフトウェアが再び割り込みアグリゲーション リングを読み出すことができ、その後、割り込みソース リングのポインタ アップデートを実行できます。たとえば C2H ストリーム割り込みの場合、ソフトウェアは、割り込みソース リング、つまり C2H 完了リングのポインタをアップデートします。

ソフトウェアの手順は、次のとおりです。

1. ソフトウェアが PCIe MSI-X メッセージを受信すると、割り込みアグリゲーション リング エントリを読み出します。
2. 書き込まれたエントリを特定するのに `coal_color` ビットを使用します。各エントリには `Qid` および `Int_type` (H2C または C2H) があり、ソフトウェアはこの `Qid` および `Int_type` からストリームなのかメモリ マップなのかをチェックできます。これは対応するソース リングをポイントしています。たとえば C2H ストリームの場合は、ソース リングは C2H 完了リングになります。この後、ソフトウェアはソース リングを読み出して情報を取得し、ソース リングのダイナミック ポインタをアップデートします。

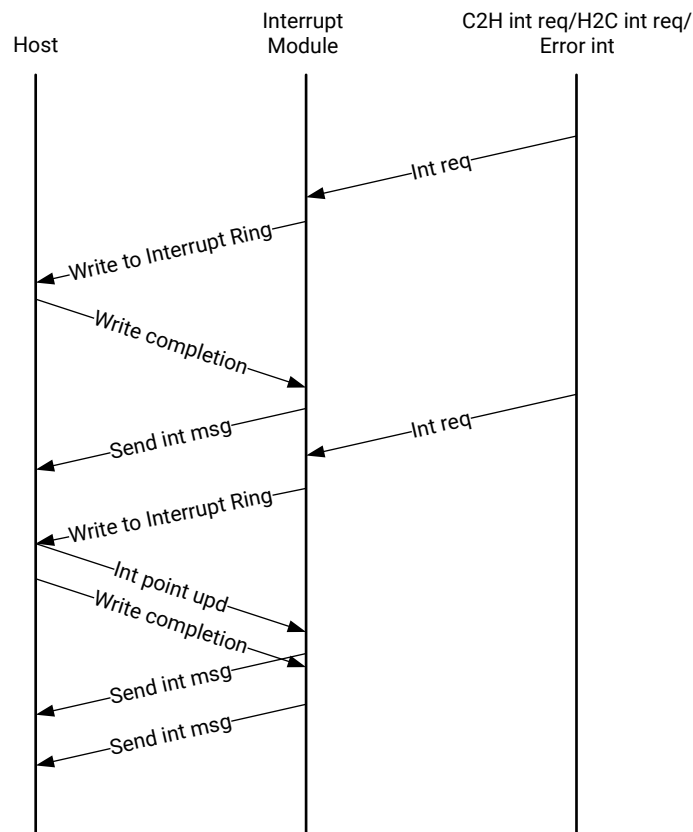
- ソフトウェアは、割り込みアグリゲーション リングの書き込まれたエントリをすべて読み終わると、QDMA_DMAP_SEL_INT_CIDX[2048] (0x18000) レジスタを使用して、ソフトウェア `cidx` のダイナミック ポインターをアップデートします。レジスタの `Qid` は、最終に書き込まれたエントリの `Qid` です。この QID により、ソフトウェアで使用される割り込みアグリゲーション リング ポインターがハードウェアに通知されます。

ソフトウェア `cidx` が `pidx` に等しくない場合、ソフトウェアが割り込みアグリゲーション リングを再度読み出せるようにするため、ハードウェアが別の PCIe MSI-X メッセージを送信します。

ソフトウェアが QDMA_DMAP_SEL_INT_CIDX[2048] (0x18000) レジスタを使用して割り込みアグリゲーション リングのダイナミック ポインターをアップデートしたときに、割り込みアグリゲーション リングのリング インデックスが送信されます。

次の図に、間接割り込みフローを示します。割り込みモジュールが割り込みリクエストを受信すると、まず割り込みアグリゲーション リングに書き込み、書き込み完了を待ちます。その後、PCIe MSI-X メッセージを送信します。割り込みリクエストが継続的に受信されれば、割り込みモジュールは処理し続けます。その間、ソフトウェアは割り込みアグリゲーション リングを読み出して、ダイナミック ポインター アップデートを実行します。ソフトウェア CIDX が PIDX に等しくない場合は、別の PCIe MSI-X メッセージが送信されます。

図 18: 割り込みフロー



X20890-052418

エラー割り込み

最下位エラー アグリゲーターはさまざまな場所にあり、エラーを記録して、中央のグローバル エラー アグリゲーターにそのエラーを伝搬します。各最下位エラー アグリゲーターにはエラー ステータス レジスタおよびエラー マスク レジスタがあります。このエラー マスクはイネーブル マスクです。エラー マスクがイネーブルになっている場合のみ、最下位エラー アグリゲーターはグローバル エラー アグリゲーターにエラーを伝搬します。

グローバル エラー アグリゲーターは、最下位エラー アグリゲーターから送られてきたすべてのエラーをまとめます。エラーが発生すると、エラー割り込みレジスタ QDMA_GLBL_ERR_INT (0B04) で `err_int_arm` ビットがセットされている場合は、グローバル エラー アグリゲーターによりエラー割り込みが生成されます。エラー割り込みが割り込みエンジンで処理されると、`err_int_arm` ビットはソフトウェアでセットされ、ハードウェアでクリアされます。このエラー割り込みは、H2C エラーおよび C2H エラーを含む、すべてのエラー用です。割り込みを再度生成するには、ソフトウェアでこの `err_int_arm` ビットをセットする必要があります。

エラー割り込みでは、直接割り込みのみがサポートされています。エラー割り込みレジスタ QDMA_GLBL_ERR_INT で `en_coal` ビットを 0 のままにしておきます。QDMA がハングしてしまうパリティやダブル ビットの ECC エラーなどの致命的エラーが発生していて、ソフトウェアでそれが認識されていないと、割り込みアグリゲーション エントリの書き込みはブロックされます。

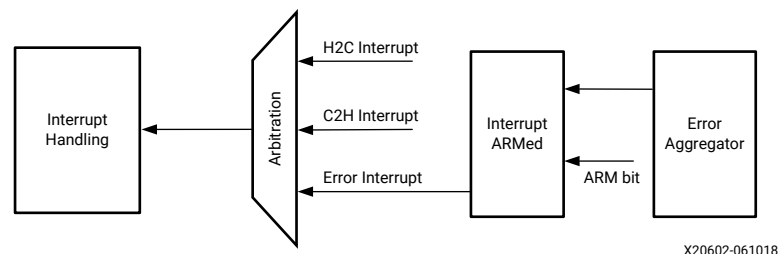
エラー割り込みは、エラー割り込みレジスタ QDMA_GLBL_ERR_INT からベクターを取得します。直接エラー割り込みの場合、ベクターは MSI-X テーブルの割り込みベクター インデックスです。

エラー割り込みの処理は次のようになります。

1. エラー割り込みレジスタ QDMA_C2H_GLBL_INT (0B04) を読み出します。
2. PCIe MSI-X メッセージを送信します。

次の図は、エラー割り込みレジスタのブロック図です。

図 19: エラー割り込み処理



レガシ割り込み

QDMA Subsystem for PCIe では物理ファンクションのレガシ割り込みがサポートされ、1 つのキューが割り込みに関連付けられます。

レガシ割り込みをイネーブルにするには、ソフトウェアで、QDMA_GLBL_INTERRUPT_CFG レジスタ (0x288) の `en_lgcy_intr` ビットを設定する必要があります。`en_lgcy_intr` が設定されていると、MSI または MSI-X 割り込みは QDMA によって送信されなくなります。

レガシ割り込みにより INTA、INTB、INTC、または INTD がアサートされると、QDMA のハードウェアにより QDMA_GLBL_INTERRUPT_CFG レジスタ (0x288) の `lgcy_intr_pending` ビットが設定されます。ソフトウェアがレガシ割り込みを受信したら、`lgcy_intr_pending` ビットをクリアにする必要があります。ハードウェアは、ソフトウェアが `lgcy_intr_pending` ビットをクリアにするまで、レガシ割り込みをアサートした状態に保ちます。

キュー管理

ファンクション マップ テーブル

ファンクション マップ テーブルは、各ファンクションにキューを割り当てるために使用されます。RAM へのインデックスはファンクション番号です。各エントリには、物理的な QID のベース番号と、ファンクションに割り当てられているキューの数が含まれています。物理的なキューへの論理的なアクセス (QDMA_TRQ_SEL_QUEUE_PF および QDMA_TRQ_SEL_QUEUE_VF 空間) を変換およびチェックすることで、ファンクション ベースのキュー アクセス保護メカニズムを提供します。テーブルのファンクションに割り当てられている範囲を超えてキュー空間に直接レジスタ アクセスがあると、そのアクセスはキャンセルされ、エラーが記録されます。

0 から 255 までで QID が 2048 未満のファンクションに対し、QDMA_TRQ_SEL_FMAP アドレス空間を介してテーブルをプログラムできます。すべてのファンクションは間接コンテキスト レジスタ空間 (QMD_IND_CTXT* レジスタ、QDMA_IND_CTXT_CMD.sel = 0xD) を介してアクセス可能です。間接コンテキスト レジスタ空間からアクセスがあると、コンテキスト構造はファンクション マップ コンテキスト構造テーブルにより定義されます。これらの空間は PF アドレス マップにしか存在しないので、物理ファンクションのみがこのテーブルを変更できます。

表 22: ファンクション マップ コンテキスト構造 (0xD)

ビット	ビット幅	フィールド名	説明
[255:44]			予約
[43:32]	12	Qid_max	ファンクションに割り当てることができる最大キュー数。
[31:11]			予約
[10:0]	11	Qid_base	ファンクションのベース キュー ID。

コンテキスト プログラミング

- 次のマスク レジスタ 8 個すべてを 1 にプログラムします。
 - [QDMA_IND_CTXT_MASK_0 \(0x824\)](#) から
 - [QDMA_IND_CTXT_MASK_7 \(0x840\)](#)
- 次のレジスタにコンテキスト値をプログラムします。
 - [QDMA_IND_CTXT_DATA_0 \(0x804\)](#) から
 - [QDMA_IND_CTXT_DATA_7 \(0x820\)](#)
 - コンテキスト データ レジスタをプログラムするには、「ソフトウェア ディスクリプター コンテキスト構造」および「C2H プリフェッチ コンテキスト構造」を参照してください。
- 対応するキューにコンテキストをプログラムするため、コンテキスト コマンド レジスタ [QDMA_IND_CTXT_CMD \(0x844\)](#) をプログラムします。

注記:

- QID はビット [17:7] で指定します。
- Opcode ビット [6:5] では、実行する操作を選択します。
- アクセスされるコンテキストはビット [4:1] で指定します。
- コンテキスト プログラミング書き込み/読み出しは、ビット [0] がセットされている場合は実行されません。

キューの設定

- ディスクリプター ソフトウェア コンテキストを消去します。
- ディスクリプター ハードウェア コンテキストを消去します。
- ディスクリプター クレジット コンテキストを消去します。
- ディスクリプター ソフトウェア コンテキストを設定します。
- PasID コンテキストを設定します (H2C/C2H キューの間で同じ ID を使用する必要あり)。
- プリフェッチ コンテキストを消去します。
- 完了コンテキストを消去します。
- 完了コンテキストを設定します。
 - 割り込み/ステータスの書き込みが必要な場合 (完了コンテキストでイネーブル)、初期完了 CIDX をアップデートして、ハードウェアをトリガー条件に応答するステートにする必要があります。リセット状態が終了したときに、ハードウェアが準備の整っていないステートに初期化されるので、この初期 CIDX アップデートが必要です。
- プリフェッチ コンテキストを設定します。

キューの除外

キューの除外 (C2H ストリーム):

- パイプラインを空にするため、マーカー パケットを送信します。
- 「マーカー」の完了を待ちます。
- ディスクリプター ソフトウェア コンテキストを無効化/消去します。
- プリフェッチ コンテキストを無効化/消去します。
- 完了コンテキストを無効化/消去します。
- タイマー コンテキストを無効化します (clear cmd はサポートされない)。

キューの除外 (H2C ストリームおよび MM):

- ディスクリプター ソフトウェア コンテキストを無効化/消去します。

仮想化

QDMA では SR-IOV パススルー仮想化がインプリメントされており、アダプターにより仮想マシン (VM) で使用する個別の仮想ファンクション (VF) にアクセスできます。オプションで、QDMA レジスタおよびリソースへのフル アクセス権を物理ファンクション (PF) に付与できますが、キューごとのポインター アップデート レジスタおよび割り込みのインプリメンテーションは VF によってのみ実行されます。VF ドライバーは、設定、リソース割り当て、および例外処理のため、メールボックスを介して PF に接続されているドライバーと通信する必要があります。プラットフォームのほかの部分に影響を与えずにデバイスをリセットするため、VM 上でオペレーティング システムをイネーブルにするファンクション レベル リセット (FLR) が QDMA によってインプリメントされます。

表 23: 権限別のアクセス

種類	注記
キュー コンテキスト/その他の制御レジスタ	PF (4 つの PF すべて) により制御されるコンテキスト アクセスのみのレジスタ。

表 23: 権限別のアクセス (続き)

種類	注記
ステータスおよび統計レジスタ	主に PF だけのレジスタ。エラー処理は、VF と PF ドライバーを連携させる必要があります。VF は、メールボックスを介して PF に接続されているドライバーと通信する必要があります。
データパス レジスタ	PF および VF の両方が、ハイパーバイザーを介さずに、データパスに関連付けられているレジスタに書き込むことができる必要があります。キューが独自の BAR 空間を使用してファンクションに関連付けられている場合、H2C/C2H ディスクリプター フェッチのポインター アップデートは VF または PF が直接実行できます。ファンクションに属さないキューへのポインター アップデートは、エラーが記録されて破棄されます。
その他の保護に関する推奨事項	VM からの無効なメモリ アクセスを防ぐため、IOMMU をオンにします。
PF ドライバーおよび VF ドライバーの通信	VF ドライバーは、全体的に影響する操作をリクエストするため、PF ドライバーと通信する必要があります。この通信チャンネルには、メッセージを送信し、割り込みを生成するためこの機能が 필요합니다。この通信チャンネルでは、各 VF にハードウェア メールボックスのセットが使用されません。

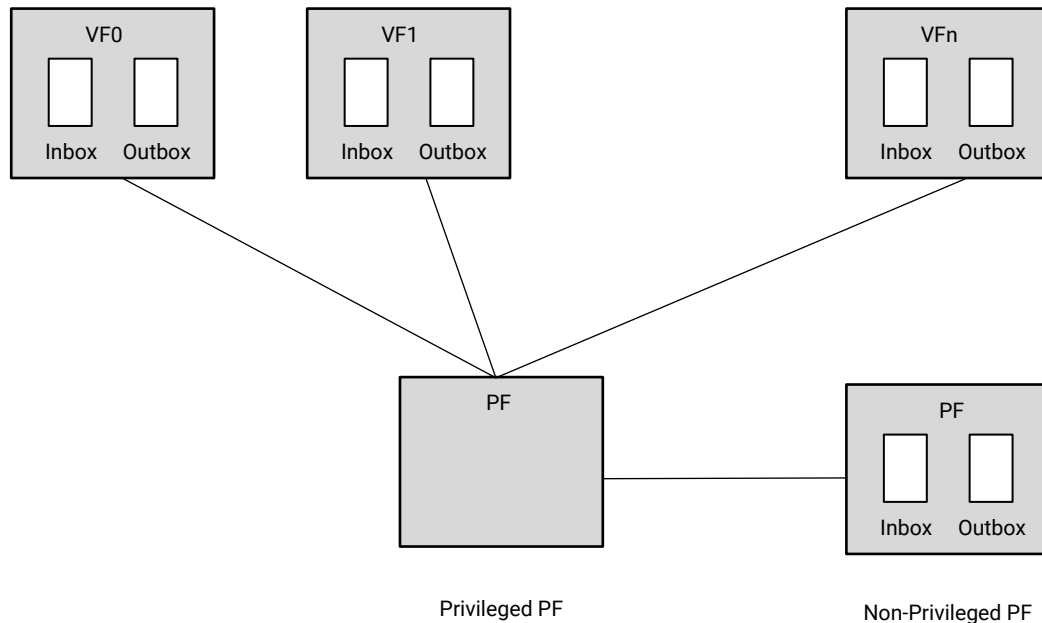
メールボックス

仮想化された環境では、PF に接続されているドライバーには、QDMA レジスタをプログラムしてアクセスするのに十分な権限があります。十分な権限のない一部の PF およびすべての VF は、メールボックス メカニズムを利用して権限のあるドライバーと通信する必要があり、通信 API をドライバーで定義する必要があります。QDMA IP では定義されません。

各ファンクション (PF および VF の両方) にはそれぞれ受信ボックスと送信ボックスが 1 つずつあり、最大 128 B のメッセージを格納できます。VF も PF もそれぞれのメールボックスにアクセスし、PF に関連付けられているすべてのファンクション (PF または VF) にアクセスできます。QDMA メールボックスでは、次のアクセスが許可されています。

- VF から関連付けられている PF へのアクセス。
- PF から、独自の仮想ファンクショングループ (VFG) に属している VF へのアクセス。
- PF (通常は QDMA レジスタへのアクセス権がないドライバー) から別の PF へのアクセス。

図 20: メールボックス



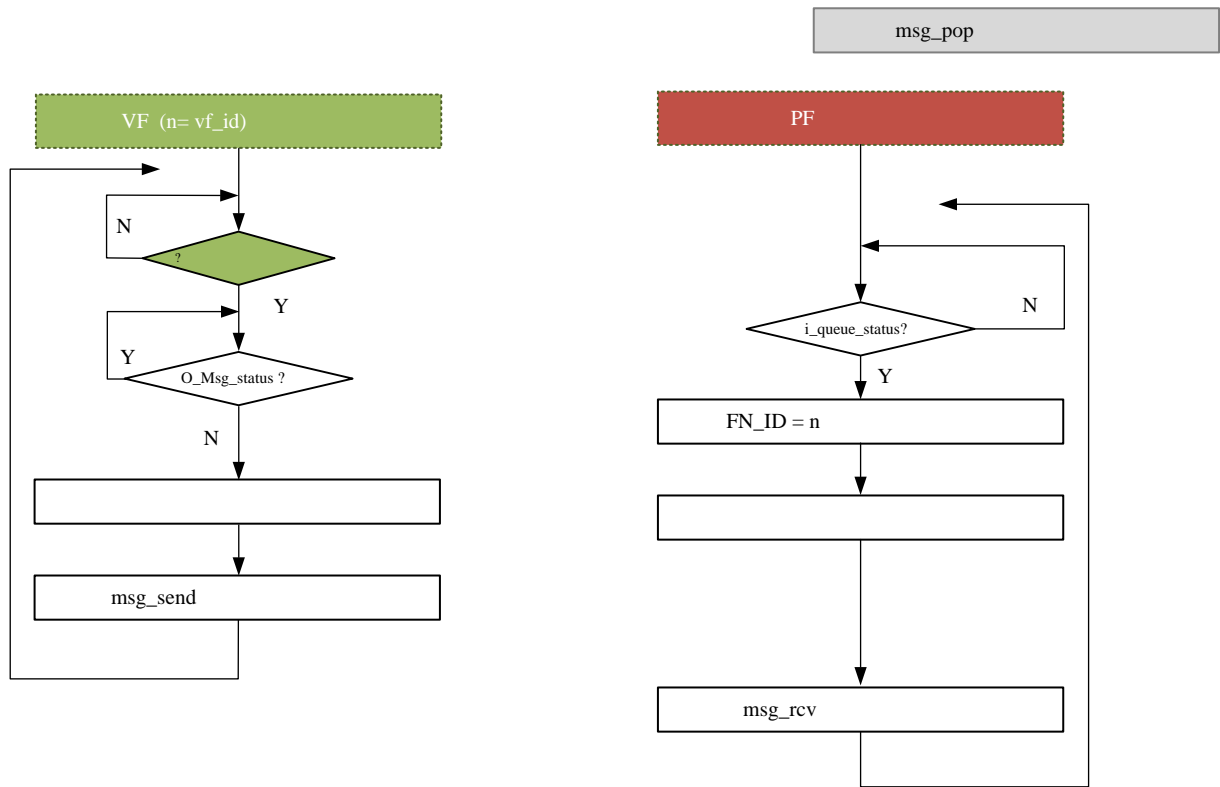
X21107-062118

VF から PF へのメッセージング

VF は、ターゲット ファンクション (PF) が受領するまで、ターゲットの PF メールボックスにメッセージを 1 つ投稿できます。メッセージが投稿される前に、ソースのファンクションの `o_msg_status` がクリアになっていることを確認する必要があります。そうすると VF が送信メッセージレジスタにメッセージを書き込むことができます。メッセージを書き込み終わると、VF ドライバーが、CSR アドレス `0x1004` に `0x1` を書き込んで `msg_send` コマンドを送信します。次に、メールボックスハードウェアが、`i_msg_status` フィールドをアサートして PF ドライバーに通知します。

入力メッセージがあるかどうかをチェックするには、ファンクションドライバーが定期的に `i_msg_status` をポーリングする必要があります。PF 側で `i_msg_status = 0x1` になっている場合、PF ドライバーが確認すべきメッセージが 1 つ以上あることを示します。メールボックスステータスレジスタの `cur_src_fn` は、最初のメッセージのファンクション ID を示します。PF ドライバーは、メールボックスターゲットファンクションレジスタを最初のメッセージのソースファンクション ID に設定する必要があります。PF の受信メッセージレジスタへのアクセスは間接的なので、メールボックスハードウェアは常に、ターゲットファンクションによって送信された対応するメッセージのバイト数を返します。PF ドライバーはメッセージの読み出しを終えると、CSR アドレス `0x1004` に `0x2` を書き込んで、`msg_rcv` コマンドも送信する必要があります。このハードウェアは、ソースファンクション側で `o_msg_status` をデアサートします。次の図に、ソースおよびデスティネーションの両方での VF から PF へのメッセージングフローを表します。

図 21: VF から PF へのメッセージング フロー



VF (#n) PF

X21105-062118

PF から VF へのメッセージング

PF から VF に属している VF へのメッセージング フローは、次の理由から、VF から PF へのフローとは若干異なります。

PF は複数のデスティネーション ファンクションにメッセージを送信できるので、ステータスをチェックするときに複数の肯定応答を受信する可能性があります。次の図にあるように、PF ドライバーは、受信メッセージ ステータスのチェック、メッセージの書き込み、コマンドの送信など、メッセージ操作を実行する前に、メールボックス ターゲット ファンクション レジスタをデスティネーション ファンクション ID にセットする必要があります。VF 側 (受信側) では、VF ドライバーが `i_msg_status = 0x1` になるたびに、メッセージを確認するため受信メッセージ レジスタを読み出す必要があります。アプリケーションによりませんが、メッセージを読み出した直後、または対応するメッセージが処理された後に、VF ドライバーは `msg_rcv` を送信できます。

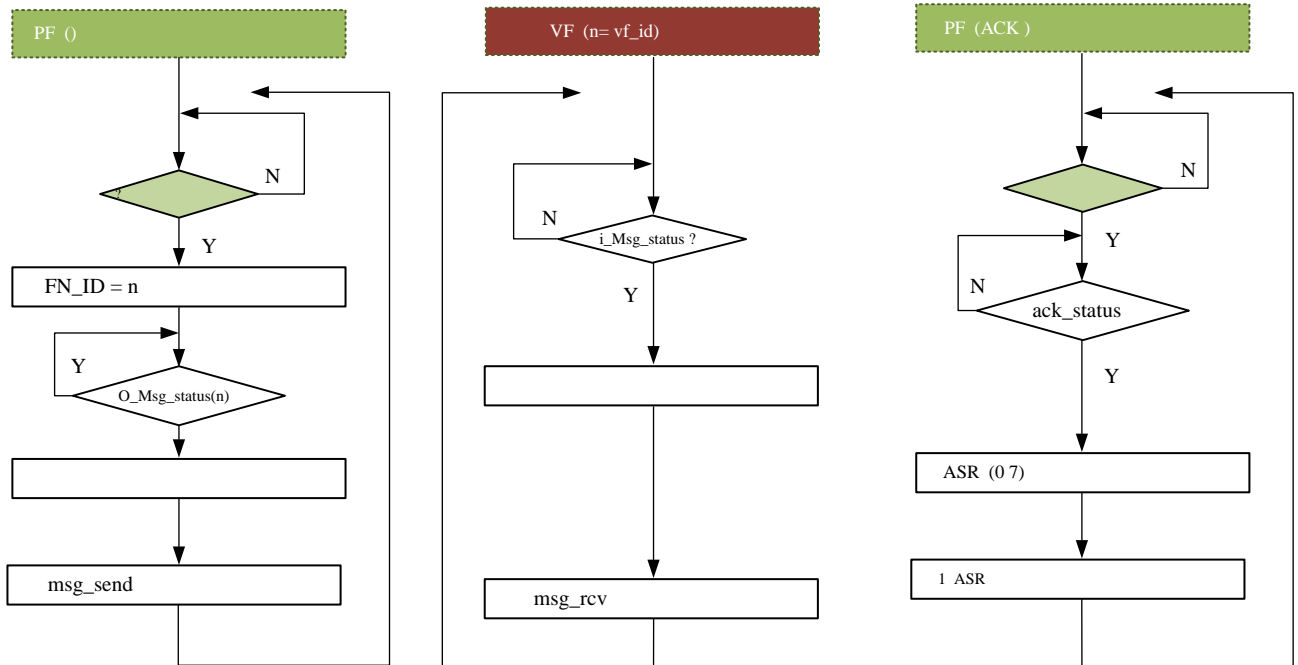
送信メッセージのステータスを 1 つずつポーリングするのを避けるため、メールボックス ハードウェアに、各 PF に対して肯定応答ステータス レジスタ (ASR) のセットがあります。VF からメールボックスが `msg_rcv` コマンドを受信すると、ソース PF の `o_msg_status` フィールドがディアサートされ、肯定応答ステータス レジスタの対応ビットもセットされます。ファンクション ID が `<N>` の VF の場合、肯定応答ステータスは次のようになります。

- 肯定応答ステータス レジスタのアドレス: `<N> / 32 + 0x2420` レジスタ アドレス
- 肯定応答ステータス ビット位置: `<N> % 32`

注記: 0x2420 レジスタ アドレスの詳細は、[PF 肯定応答レジスタ \(0x2420-0x243C\)](#) を参照してください。

肯定応答ステータスレジスタ (ASR) にアサートされているビットがあると、メールボックスハードウェアがステータスレジスタ (0x2400) の `ack_status` フィールドをアサートします。PF ドライバーは、肯定応答ステータスレジスタを実際に読み出す前に、`ack_status` をポーリングできます。1 回のレジスタアクセスで複数の完了が検出されることがあります。処理が終わると、PF ドライバーは、同じレジスタアドレスに値を書き込んでステータスをクリアにする必要があります。

図 22: PF から VF へのメッセージングフロー



X21106-062118

メールボックス割り込み

メールボックスモジュールでは、イベント通知メカニズムとして割り込みがサポートされています。各メールボックスには割り込み制御レジスタがあります (PF の場合はオフセット 0x2410 で、VF の場合はオフセット 0x1010)。割り込みをイネーブルにするには、このレジスタを 1 にセットします。割り込みがイネーブルになると、メールボックスで処理すべき保留イベント (保留の受信メッセージや送信メッセージの肯定応答など) がある場合は、メールボックスは割り込みを QDMA に送信します。ドライバーの設定に従って、ファンクション割り込みベクターレジスタ (PF の場合は 0x2408、VF の場合は 0x1008) を介して割り込みベクターを設定します。

割り込みをイネーブルにしてもイベントログメカニズムは変わりません。つまり、ユーザーがファンクションステータスレジスタを読み出して保留のイベントをチェックする必要があります。割り込みリクエストに回答するための最初の手順は、割り込みをディスエーブルにすることです。実際の保留イベントの数が、メールボックスが割り込みを送信した時点のイベント数よりも多い可能性があります。



推奨: ザイリンクスでは、ユーザーアプリケーションの割り込みハンドラーが、ステータスレジスタにある保留イベントをすべて処理することを推奨しています。割り込み応答が終了したら、ユーザーアプリケーションが割り込みを再度イネーブルにします。

割り込み制御がディスエーブルからイネーブルに変更された時点で、メールボックスがイベントステータスをチェックします。割り込みステータスを読み出してから割り込みを再度イネーブルにするまでの間にメールボックスに新しいイベントが送信された場合は、メールボックスが新しい割り込みリクエストをすぐに生成します。

ファンクション レベル リセット (FLR)

ファンクション レベル リセット (FLR) を詳細に設定し、ソフトウェアでエンドポイント ハードウェアを静止させたりリセットしたりできます。VF がリセットされると、その VF に関連付けられているリソースのみがリセットされず、PF がリセットされると、関連付けられている VF のリソースも含め、PF のリソースすべてがリセットされます。FLR は権限がないと実行できない操作なので、管理システムで実行されている PF ドライバーによって実行される必要があります。

使用モード

1. ファンクションのオン/オフ (電源のオン/オフなど) が切り替えられるときに、ハイパーバイザーが FLR をリクエストします。
2. FLR は次のようにリクエストできます。

```
echo 1 > /sys/bus/pci/devices/$BDF/reset
```

\$BDF は、ターゲット ファンクションのバス デバイス ファンクションの数です。

FLR プロセス

FLR プロセスには主に 3 つの段階があります。

1. プリ FLR: ターゲット ファンクションの QDMA コンテキスト構造、メールボックス、ユーザー ロジックをすべてリセットします。
 - 各ファンクションには、ファンクションのプリ FLR ステータスを監視する MDMA_PRE_FLR_STATUS というレジスタがあります。そのオフセットは次のように計算されます。
 - $MDMA_PRE_FLR_STATUS_OFFSET = MB_base + 0x100$
これは、ファンクションのメールボックス メモリ空間のオフセット 0x100 にあります。PF および VF にはそれぞれ異なる Mb_base があります。MDMA_PRE_FLR_STATUS の定義を、下の表に示します。
 - プリ FLR を開始するため、ソフトウェアによりターゲット ファンクションの MDMA_PRE_FLR_STATUS[0] (ビット 0) に 1 が書き込まれます。プリ FLR が完了すると、ハードウェアにより MDMA_PRE_FLR_STATUS[0] がクリアされます。ソフトウェアにより継続して MDMA_PRE_FLR_STATUS[0] のステータス情報が収集され、0 が返されると次の段階に進みます。

表 24: MDMA_PRE_FLR_STATUS レジスタ

オフセット	フィールド	R/W タイプ	幅	デフォルト	説明
0x100		RW	32	0	
		RW	32:1	0	
	pre_flr_st	RW	0	0	1: プリ FLR の開始 0: プリ FLR の完了 ドライバによりセットされ、ハードウェアによってクリアされず。

2. 静止: すべての保留トランザクションが完了していることをソフトウェアでチェックする必要があります。デバイス ステータス レジスタ (PCIe コンフィギュレーション空間) のトランザクション保留ビットがクリアになるまで、または一定期間後タイムアウトになるまで、このビットをポーリングすることでチェックできます。
3. PCIe-FLR: PCIe コントローラーのターゲット ファンクションのすべてのリソースがリセットされます。
 - ターゲット ファンクションの FLR ビット (PCIe デバイス制御レジスタのビット 15) を開始し、PCIe で FLR プロセスをトリガーするため 1 にセットする必要があります。

OS サポート

PF ドライバーが読み込まれて実行している場合 (使用モード 1 など) は、前述の 3 つのステップはこのドライバーによって実行されます。ただし、UltraScale+ で、PF ドライバーを読み込む前に FLR を実行する必要がある場合は (使用モード 2 など)、`///.../source/drivers/pci/quick.c` で定義されているファンクションを介して正しい FLR シーケンスを実行できるように、OS カーネルパッチが提供されます。

システム管理

リセット

QDMA Subsystem for PCIe では、リンクダウン、リセット、ホットリセット、ファンクションレベルリセット (FLR) (静止モードのみをサポート) などの PCIe 定義されたリセットがすべてサポートされています。

ソフトリセット

QDMA ロジックを `soft_reset_n` ポートを介してリセットします。このポートは、最低 100 クロックサイクル間 (`axi_aclk` サイクル) リセット状態にしておく必要があります。

これによって、PCIe ハードブロックはリセットされません。ロジックの DMA 部分のみがリセットされます。

VDM

ベンダー定義メッセージ (VDM) は、PCI Express の既存メッセージ機能の拡張機能です。PCI Express の仕様には、VDM の追加要件、ヘッダーフォーマット、配信情報が定義されています。詳細は、PCI-SIG Specifications (<http://www.pcisig.com/specifications>) を参照してください。

QDMA では VDM の送受信が可能です。この機能をイネーブルにするには、Vivado の [Customize IP] ダイアログボックスで [Enable Bridge Slave Mode] を選択します。

RX ベンダー定義メッセージ:

1. QDMA が VDM を受信すると、メッセージが `st_rx_msg` ポートで受信されます。
2. この入力データストリームは `st_rx_msg_data` ポート (DW ごと) で取り込まれます。
3. ユーザーアプリケーションでこの VDM を受領できるかどうかを通知するには、ユーザーアプリケーションで `st_rx_msg_rdy` を駆動する必要があります。
4. `st_rx_msg_rdy` が High になると、VDM がユーザーアプリケーションに転送されます。
5. ユーザーアプリケーションは、この VDM を格納し、受信したパケット数を監視する必要があります。

ポートの詳細は、[VDM ポート](#) を参照してください。

TX ベンダー定義メッセージ:

1. QDMA から VDM を送信できるようにするには、AXI-Lite スレーブインターフェイスを介して、ブリッジの TX メッセージレジスタをプログラムします。
2. ブリッジには、PCIe TX メッセージデータ FIFO レジスタ (TX_MSG_DFIFO) の表で示すように、TX メッセージ制御、ヘッダー L (バイト 8 から 11)、ヘッダー H (バイト 12 から 15)、および TX メッセージデータのレジスタがあります。
3. TX メッセージヘッダー L レジスタの場合は、AXI4-Lite スレーブインターフェイスを介して、オフセット 0xE64 に書き込みを実行します。

4. 必須の VDM TX ヘッダー H レジスタの場合は、オフセット 0xE68 をプログラムします。
5. DW0 から DW15 までの VDM メッセージの場合は、オフセット 0xE6C に書き込みを 1 つずつ実行して、16 個の DW ペイロードをプログラムします。
6. VDM TX パケットを送信するには、オフセット 0xE60 にある TX_MSG_CTRL レジスタの `msg_routing`、`msg_code`、データ長、リクエスター ファンクション フィールド、および `msg_execute` フィールドをプログラムします。
7. TX メッセージ制御レジスタも、ビット 23 でメッセージの完了ステータスを示します。VDM パケットの送信が問題なくできたことを確認するには、このビットを読み出す必要があります。
8. レジスタのすべてのフィールドは RW ですが、例外は、TX 制御レジスタのビット 23 (`msg_fail`) で、これは 1 を書き込むとクリアになります。
9. VDM TX パケットは、AXI-ST RQ 送信インターフェイスで送信されます。

これらレジスタについては、次を参照してください。

- [PCIe TX メッセージ制御レジスタ \(0xE60\) \(TX_MSG_CTRL\)](#)
- [PCIe TX メッセージ ヘッダー L レジスタ \(0xE64\) \(TX_MSG_HDR_L\)](#)
- [PCIe TX メッセージ ヘッダー H レジスタ \(0xE68\) \(TX_MSG_HDR_H\)](#)
- [PCIe TX メッセージ データ FIFO レジスタ \(0xE6C\) \(TX_MSG_DFIFO\)](#)

コンフィギュレーション拡張

PCIe 拡張インターフェイスは、コンフィギュレーション空間がさらに必要な場合に選択できます。コンフィギュレーション拡張インターフェイスが選択されている場合は、インターフェイスが正しく機能するように、それを拡張するためのロジックを追加してください。

拡張 ROM

このオプションが選択されている場合、拡張 ROM がアクティベートされ、2 KB から 4 GB までの値を指定できます。『PCI 3.0 Local Bus Specification』(PCI-SIG Specifications (<http://www.pcisig.com/specifications>)) によると、拡張 ROM BAR のサイズは 16 MB 以下に抑える必要があります。16 MB よりも大きなアドレス空間を選択すると、コアが規格に準拠しなくなる可能性があります。

エラー

リンク ダウン エラー

DMA 操作中に PCIe リンクがダウンすると、トランザクションが失われる可能性があり、DMA も完了できなくなる可能性があります。そういう場合でも、AXI4 インターフェイスは操作し続けます。C2H ブリッジ AXI4 MM インターフェイスでの未処理読み出しリクエストは、正しい完了またはスレーブ エラー応答のある完了を受信します。DMA は、ステータス レジスタにリンク ダウン エラーを記録します。タイムアウトの設定およびリンク ダウン状況からのリカバリ処理はドライバーが実行します。

パリティ エラー

パススルーパリティは、プライマリのデータパスでサポートされています。パリティエラーはC2Hストリーミング、H2Cストリーミング、メモリマップド、ブリッジマスター、およびブリッジスレーブインターフェイスで発生する可能性があります。書き込みペイロードのパリティエラーは、C2Hストリーミング、メモリマップド、およびブリッジスレーブで発生する可能性があります。ブリッジスレーブインターフェイスの書き込みペイロードのおよび読み出し完了のダブルビットエラーは、パリティエラーを引き起こします。PCIeへのリクエストのパリティエラーはコアにより破棄され、PCIeにより致命的エラーが記録されます。パリティエラーは回復不可能なので、予期しない動作になる可能性があります、パリティエラーが起きている最中および起きた後のDMAは無効だと考える必要があります。

DMA エラー

エラー アグリゲーター

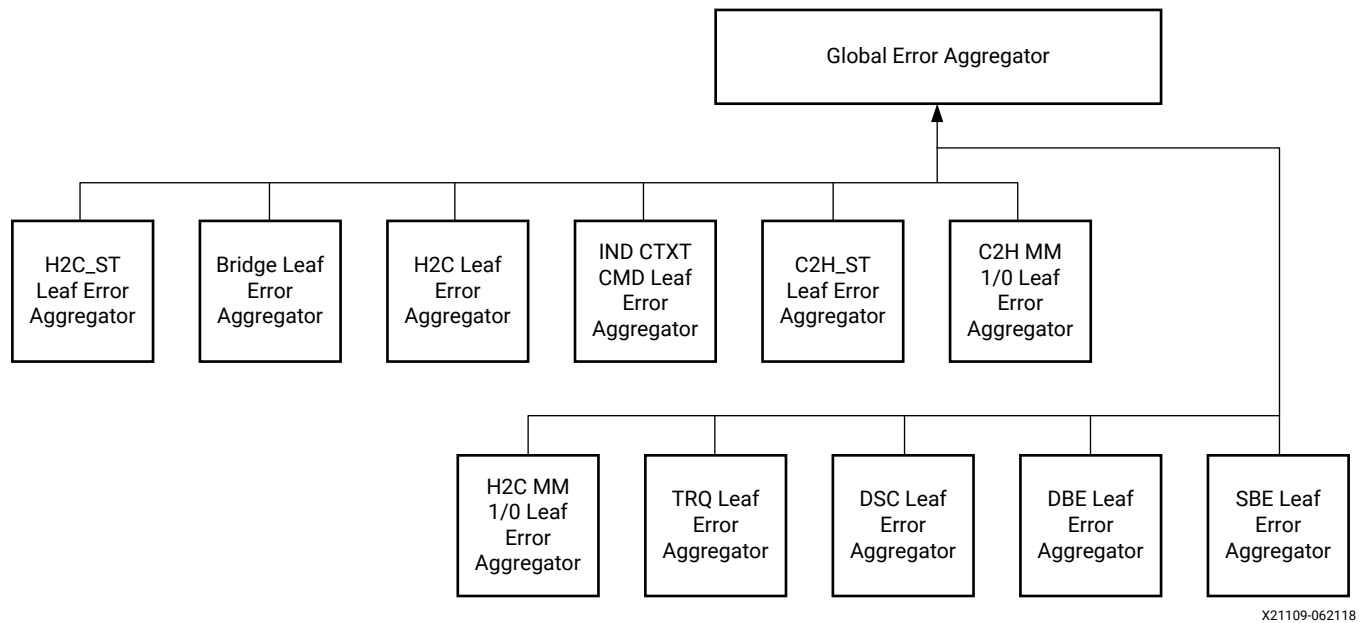
最下位エラーアグリゲーターはさまざまな場所にあり、エラーを記録して、中央のグローバルエラーアグリゲーターにそのエラーを伝搬します。グローバルエラーアグリゲーターは、最下位エラーアグリゲーターから送信されたすべてのエラーをまとめます。

QDMA_GLBL_ERR_STATレジスタは、グローバルエラーアグリゲーターのエラーステータスレジスタです。ビットフィールドは、最下位エラーアグリゲーターの位置を示しているため、エラーを正確に把握するため、個々の最下位エラーアグリゲーターのエラーステータスレジスタを確認できるようになっています。詳細は、[QDMA_GLBL_ERR_STAT \(0X248\)](#)を参照してください。

QDMA_GLBL_ERR_MASKは、グローバルエラーアグリゲーターのエラーマスクレジスタで、対応するエラーのマスクビットを含みます。マスクビットがセットされていると、割り込みを生成するため、対応するエラーが次のレベルに伝搬されます。エラー生成割り込みについては、割り込みのセクションで説明されています。詳細は、[QDMA_GLBL_ERR_MASK \(0X24C\)](#)を参照してください。エラー割り込みは[QDMA_GLBL_ERR_INT \(0xB04\)](#)によって制御されます。

各最下位エラーアグリゲーターには、エラーステータスレジスタおよびエラーマスクレジスタが含まれます。エラーステータスレジスタはエラーを記録します。エラーが発生するとハードウェアがビットをセットし、ソフトウェアが必要に応じて1'b1を書き込んでそのビットをクリアします。エラーマスクレジスタには、対応するエラーのマスクビットが含まれます。マスクビットがセットされていると、対応するエラーがグローバルエラーアグリゲーターに伝搬されるようになります。エラーマスクレジスタは、エラーステータスレジスタへのエラーの記録には影響しません。

図 23: エラー アグリゲーター



X21109-062118

最下位エラー アグリゲーターのエラー ステータス レジスタおよびエラー マスク レジスタへのリンクは、次のとおりです。

C2H ストリーミング エラー

[QDMA_C2H_ERR_STAT \(0xAF0\)](#): C2H ストリーミング エラーのエラー ステータス レジスタです。

[QDMA_C2H_ERR_MASK \(0xAF4\)](#): エラー マスク レジスタです。対応する C2H ストリーミング エラーがエラー アグリゲーターに伝搬されるように、ソフトウェアによりこのビットがセットされます。

[QDMA_C2H_FIRST_ERR_QID \(0xB30\)](#): 最初の C2H ストリーミング エラーの QID です。

C2H MM エラー

[QDMA_C2H MM ステータス \(0x1040\)](#)

[C2H MM エラー コード マスク \(0x1054\)](#)

[C2H MM エラー コード \(0x1058\)](#)

[C2H MM エラー情報 \(0x105C\)](#)

QDMA H2C0 MM エラー

[H2C0 MM ステータス \(0x1240\)](#)

[H2C MM エラー コード イネーブル マスク \(0x1254\)](#)

[H2C MM エラー コード \(0x1258\)](#)

[H2C MM エラー情報 \(0x125C\)](#)

TRQ エラー

[QDMA_GLBL_TRQ_ERR_STS \(0x260\)](#): TRQ エラーのエラー ステータス レジスタです。

[QDMA_GLBL_TRQ_ERR_MSK \(0x264\)](#): エラー マスク レジスタです。

[QDMA_GLBL_TRQ_ERR_LOG_A \(0x268\)](#): エラー記録レジスタです。エラーが発生したときのアクセスのセレクト、ファンクション、アドレスが記録されます。

ディスクリプター エラー

[QDMA_GLBL_DSC_ERR_STS \(0x254\)](#)

[QDMA_GLBL_DSC_ERR_MSK \(0x258\)](#)

これはエラー記録レジスタです。QID、DMA の方向、エラーのコンシューマー インデックスが記録されます。

[QDMA_GLBL_DSC_ERR_LOG0 \(0x25C\)](#)

[QDMA_GLBL_TRQ_ERR_STS \(0x260\)](#): TRQ エラーのエラー ステータス レジスタです。

RAM ダブル ビット エラー

[QDMA_RAM_DBE_STS_A \(0xFC\)](#)

[QDMA_RAM_DBE_MSK_A \(0xF8\)](#)

RAM シングル ビット エラー

[QDMA_RAM_SBE_STS_A \(0xF4\)](#)

[QDMA_RAM_SBE_MSK_A \(0xF0\)](#)

C2H ストリーミング致命的エラー処理

[QDMA_C2H_FATAL_ERR_STAT \(0xAF8\)](#): C2H ストリーミング致命的エラーのエラー ステータス レジスタ。

[QDMA_C2H_FATAL_ERR_MASK \(0xAFC\)](#): エラー マスク レジスタ。対応する C2H 致命的エラーが C2H 致命的処理ロジックに送信されるように、ソフトウェアによりこのビットがセットされます。

[QDMA_C2H_FATAL_ERR_ENABLE \(0xB00\)](#): このレジスタは、2 つの C2H ストリーミング致命的エラー処理プロセスをイネーブルにします:

- C2H DMA 書き込みエンジンからの WRQ をディスエーブルにすることでデータ転送が停止します。
- データ転送で WPL パリティを反転させます。

ポートの説明

QDMA Subsystem for PCIe は PCIe 統合ブロックに直接接続されます。PCIe Integrated Block IP へのデータパスインターフェイスは 64、128、256、または 512 ビット幅で、IP の設定により、最高 250 MHz までの速度で動作します。このデータパス幅はすべてのデータ インターフェイスを対象にしています。次に、このコアに関連付けられているポートの説明を示します。

サブシステム インターフェイスは [QDMA のアーキテクチャ](#) に示されています。

表 25: パラメーター

パラメーター名	説明
PL_LINK_CAP_MAX_LINK_WIDTH	物理的レーン幅
C_M_AXI_ADDR_WIDTH	AXI マスター インターフェイスのアドレス幅
C_M_AXI_ID_WIDTH	AXI マスター インターフェイスの ID 幅
C_M_AXI_DATA_WIDTH	AXI マスター インターフェイスのデータ幅 64、128、256、または 512 ビット
C_S_AXI_ID_WIDTH	AXI スレーブ インターフェイスの ID 幅
C_S_AXI_ADDR_WIDTH	AXI スレーブ インターフェイスのアドレス幅
C_S_AXI_DATA_WIDTH	AXI スレーブ インターフェイスのデータ幅 64、128、256、または 512 ビット
C_S_AXI_ID_WIDTH	AXI スレーブ インターフェイスの ID 幅
AXI_DATA_WIDTH	AXI DMA 転送データ幅。 64、128、256、または 512 ビット。

QDMA グローバル ポート

表 26: QDMA グローバル ポートの説明

ポート名	I/O	説明
sys_clk	I	基準クロック IBUFDS_GTE4 の ODIV2 ポートにより駆動される必要があります。『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』 (PG213: 英語版 、 日本語版) を参照してください。
sys_clk_gt	I	PCIe 基準クロック。基準クロック IBUFDS_GTE4 のポートから駆動される必要があります。『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』 (PG213: 英語版 、 日本語版) を参照してください。
sys_rst_n	I	PCIe エッジ コネクタ リセット信号からのリセット。
pci_exp_txp [PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	O	PCIe TX シリアル インターフェイス。
pci_exp_txn [PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	O	PCIe TX シリアル インターフェイス。
pci_exp_rxp [PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	I	PCIe RX シリアル インターフェイス。
pci_exp_rxn [PL_LINK_CAP_MAX_LINK_WIDTH-1:0]	I	PCIe RX シリアル インターフェイス。
user_lnk_up	O	PCI Express コアがホスト デバイスとリンク アップしたことを示す出力 (アクティブ High)。
axi_aclk	O	ユーザー クロック出力。QDMA のインターフェイス信号すべて (入出力両方) の PCIe 派生クロック出力。QDMA への入力を駆動し、QDMA からの出力をゲートするには、このクロックを使用します。
axi_aresetn	O	ユーザー リセット出力。axi_aclk 出力に供給されるクロックと同期する AXI リセット信号。このリセットは、対応する AXI インターコネクタ aresetn 信号すべてを駆動する必要があります。
user_reset_n	I	ユーザー リセット (アクティブ Low)。何らかの理由で DMA ロジックをリセットする必要がある場合は、このポートを使用して DMA ロジックのリセットをアサートします。

表 26: QDMA グローバル ポートの説明 (続き)

ポート名	I/O	説明
phy_ready	O	PHY レディ 出力ステータス。

すべての AXI インターフェイスは axi_aclk 信号によってクロック供給されます。DMA にすべての信号を駆動するには、axi_aclk を使用してください。

AXI ブリッジ マスター ポート

表 27: AXI4 メモリ マップド マスター ブリッジ読み出しアドレス インターフェイス ポートの説明

信号名	I/O	説明
m_axib_araddr [C_M_AXI_ADDR_WIDTH-1:0]	O	この信号は、ホストからユーザー ロジックへのメモリ マップド読み出し用のアドレスです。
m_axib_arid [C_M_AXI_ID_WIDTH-1:0]	O	マスター読み出しアドレス ID。
m_axib_arlen[7:0]	O	マスター読み出しアドレスの長さ。
m_axib_arsize[2:0]	O	マスター読み出しアドレスのサイズ。
m_axib_arprot[2:0]	O	マスター読み出し保護タイプ。
m_axib_arvalid	O	この信号がアサートされていると、m_axib_araddr のアドレスに対する有効な読み出しリクエストがあることを意味します。
m_axib_arready	I	マスター読み出しアドレス準備完了。
m_axib_arlock	O	マスター読み出しロック タイプ。
m_axib_arsize[3:0]	O	マスター読み出しメモリ タイプ。
m_axib_arburst[1:0]	O	マスター読み出しアドレスのバースト タイプ。
m_axib_aruser[28:0]	O	マスター読み出しユーザー ビット。 m_axib_aruser[7:0] = ファンクション番号 m_axib_aruser[15:8] = 予約 m_axib_aruser[18:16] = BAR ID m_axib_aruser[26:19] = VF オフセット m_axib_aruser[28:27] = VF ID

表 28: AXI4 メモリ マップド マスター ブリッジ読み出しインターフェイス ポートの説明

信号名	I/O	説明
m_axib_rdata [C_M_AXI_DATA_WIDTH-1:0]	I	マスター読み出しデータ。
m_axib_ruser [C_M_AXI_DATA_WIDTH/8-1:0]	I	m_axib_ruser[C_M_DATA_WIDTH/8-1:0] = 読み出しデータの奇数パリティ (バイトごと)。
m_axib_rid [C_M_AXI_ID_WIDTH-1:0]	I	マスター読み出し ID。
m_axib_rresp[1:0]	I	マスター読み出し応答。
m_axib_rlast	I	マスター読み出し最終。
m_axib_rvalid	I	マスター読み出し有効。
m_axib_rready	O	マスター読み出し準備完了。

表 29: AXI4 メモリ マップド マスター ブリッジ書き込みアドレス インターフェイス ポートの説明

信号名	I/O	説明
m_axib_awaddr [C_M_AXI_ADDR_WIDTH-1:0]	O	この信号は、ホストからユーザー ロジックへのメモリ マップド書き込み用のアドレスです。
m_axib_awid [C_M_AXI_ID_WIDTH-1:0]	O	マスター書き込みアドレス ID。
m_axib_awlen[7:0]	O	マスター書き込みアドレスの長さ。
m_axib_awsz[2:0]	O	マスター書き込みアドレスのサイズ。
m_axib_awburst[1:0]	O	マスター書き込みアドレスのバースト タイプ。
m_axib_awprot[2:0]	O	マスター書き込み保護タイプ。
m_axib_awvalid	O	この信号がアサートされていると、m_axib_araddr のアドレスに対する有効な書き込みリクエストがあることを意味します。
m_axib_awready	I	マスター書き込みアドレス準備完了。
m_axib_awlock	O	マスター書き込みロック タイプ。
m_axib_awcache[3:0]	O	マスター書き込みメモリ タイプ。
m_axib_awuser[28:0]	O	マスター書き込みユーザー ビット。 m_axib_awuser[7:0] = ファンクション番号 m_axib_awuser[15:8] = 予約 m_axib_awuser[18:16] = BAR ID m_axib_awuser[26:19] = VF オフセット m_axib_awuser[28:27] = VF ID

表 30: AXI4 メモリ マップド マスター ブリッジ書き込みインターフェイス ポートの説明

信号名	I/O	説明
m_axib_wdata [C_M_AXI_DATA_WIDTH-1:0]	O	マスター書き込みデータ。
m_axib_wuser [C_M_AXI_DATA_WIDTH/8-1:0]	O	m_axib_wuser [C_M_AXI_DATA_WIDTH/8-1:0] = 書き込みデータの奇数パリティ (バイトごと)。
m_axib_wlast	O	マスター書き込み最終。
m_axib_wstrb [C_M_AXI_DATA_WIDTH/8-1:0]	O	マスター書き込みストロブ。
m_axib_wvalid	O	マスター書き込み有効。
m_axib_wready	I	マスター書き込み準備完了。

表 31: AXI4 メモリ マップド マスター ブリッジ書き込み応答インターフェイス ポートの説明

信号名	I/O	説明
m_axib_bvalid	I	マスター書き込み応答有効。
m_axib_bresp[1:0]	I	マスター書き込み応答。
m_axib_bid [C_M_AXI_ID_WIDTH-1:0]	I	マスター書き込み応答 ID。
m_axib_bready	O	マスター応答準備完了。

AXI ブリッジ スレーブ ポート

表 32: AXI4 メモリ マップド スレーブ ブリッジ書き込みアドレス インターフェイス ポートの説明

ポート名	I/O	説明
s_axib_awid [C_S_AXI_ID_WIDTH-1:0]	I	スレーブ書き込みアドレスの ID。
s_axib_awaddr [C_S_AXI_ADDR_WIDTH-1:0]	I	スレーブ書き込みアドレス。
s_axib_awuser[7:0]	I	s_axib_awuser[7:0] はファンクション番号を示します。
s_axib_awregion[3:0]	I	スレーブ書き込み領域デコード。
s_axib_awlen[7:0]	I	スレーブ書き込みバーストの長さ。
s_axib_awsz[2:0]	I	スレーブ書き込みバーストのサイズ。
s_axib_awburst[1:0]	I	スレーブ書き込みバーストのタイプ。
s_axib_awvalid	I	スレーブ アドレス書き込み有効。
s_axib_awready	O	スレーブ アドレス書き込み準備完了。

表 33: AXI4 メモリ マップド スレーブ ブリッジ書き込みインターフェイス ポートの説明

ポート名	I/O	説明
s_axib_wdata [C_S_AXI_DATA_WIDTH-1:0]	I	スレーブ書き込みデータ。
s_axib_wstrb [C_S_AXI_DATA_WIDTH/8-1:0]	I	スレーブ書き込みストロブ。
s_axib_wlast	I	スレーブ書き込み最終。
s_axib_wvalid	I	スレーブ書き込み有効。
s_axib_wready	O	スレーブ書き込み準備完了。
s_axib_wuser [C_S_AXI_DATA_WIDTH/8-1:0]	I	s_axib_wuser [C_S_AXI_DATA_WIDTH/8-1:0] = 書き込みデータの奇数パリティ (バイトごと)。

表 34: AXI4 メモリ マップド スレーブ ブリッジ書き込み応答インターフェイス ポートの説明

ポート名	I/O	説明
s_axib_bid [C_S_AXI_ID_WIDTH-1:0]	O	スレーブ応答 ID。
s_axib_bresp[1:0]	O	スレーブ書き込み応答。
s_axib_bvalid	O	スレーブ書き込み応答有効。
s_axib_bready	I	スレーブ応答準備完了。

表 35: AXI4 メモリ マップド スレーブ ブリッジ読み出しアドレス インターフェイス ポートの説明

ポート名	I/O	説明
s_axib_arid [C_S_AXI_ID_WIDTH-1:0]	I	スレーブ読み出しアドレスの ID。
s_axib_araddr [C_S_AXI_ADDR_WIDTH-1:0]	I	スレーブ読み出しアドレス。

表 35: AXI4 メモリ マップド スレーブ ブリッジ読み出しアドレス インターフェイス ポートの説明 (続き)

ポート名	I/O	説明
s_axib_arregion[3:0]	I	スレーブ読み出し領域デコード。
s_axib_arlen[7:0]	I	スレーブ読み出しバーストの長さ。
s_axib_arsize[2:0]	I	スレーブ読み出しバーストのサイズ。
s_axib_arburst[1:0]	I	スレーブ読み出しバーストのタイプ。
s_axib_arvalid	I	スレーブ読み出しアドレス有効。
s_axib_arready	O	スレーブ読み出しアドレスの準備完了。

表 36: AXI4 メモリ マップド スレーブ ブリッジ読み出しインターフェイス ポートの説明

ポート名	I/O	説明
s_axib_rid [C_S_AXI_ID_WIDTH-1:0]	O	スレーブ読み出しの ID タグ。
s_axib_rdata [C_S_AXI_ID_WIDTH-1:0]	O	スレーブ読み出しデータ。
s_axib_ruser [C_S_AXI_DATA_WIDTH/8-1:0]	O	s_axib_aruser[C_S_AXI_ID_WIDTH/8-1:0] = 読み出しデータの奇数パリティ (バイトごと)。
s_axib_rresp[1:0]	O	スレーブ読み出し応答。
s_axib_rlast	O	スレーブ読み出し最終。
s_axib_rvalid	O	スレーブ読み出し有効。
s_axib_rready	I	スレーブ読み出し準備完了。

AXI4-Lite マスター ポート

表 37: コンフィギュレーション AXI4-Lite メモリ マップド書き込みマスター インターフェイスポートの説明

信号名	I/O	説明
m_axil_awaddr[31:0]	O	この信号は、ホストからユーザー ロジックへのメモリ マップド書き込み用のアドレスです。
m_axil_awprot[2:0]	O	プロテクション タイプです。
m_axil_awvalid	O	この信号がアサートされていると、m_axil_awaddr のアドレスに対する有効な書き込みリクエストがあることを意味します。
m_axil_awready	I	マスター書き込みアドレス準備完了。
m_axil_awuser [29:0]		m_axil_awuser[7:0] = ファンクション番号 M_axil_awuser[15:8] = 予約 m_axil_awuser[18:16] = BAR ID m_axil_awuser[26:19] = vfg オフセット m_axil_awuser[28:27] = vfg ID
m_axil_wdata[31:0]	O	マスター書き込みデータ。
m_axil_wstrb[3:0]	O	マスター書き込みストロブ。
m_axil_wvalid	O	マスター書き込み有効。
m_axil_wready	I	マスター書き込み準備完了。
m_axil_bvalid	I	マスター応答有効。

表 37: コンフィギュレーション AXI4-Lite メモリ マップド書き込みマスター インターフェイスポートの説明 (続き)

信号名	I/O	説明
m_axil_bresp[1:0]	I	
m_axil_bready	O	マスター応答有効。

表 38: コンフィギュレーション AXI4-Lite メモリ マップド読み出しマスター インターフェイスポートの説明

信号名	I/O	説明
m_axil_araddr[31:0]	O	この信号は、ホストからユーザー ロジックへのメモリ マップド読み出し用のアドレスです。
m_axil_aruser[28:0]		m_axil_aruser[7:0] = ファンクション番号 m_axil_aruser[15:8] = 予約 m_axil_aruser[18:16] = BAR ID m_axil_aruser[26:19] = vfg オフセット m_axil_aruser[28:27] = vfg ID
m_axil_arprot[2:0]	O	プロテクション タイプです。
m_axil_arvalid	O	この信号がアサートされていると、m_axil_araddr のアドレスに対する有効な読み出しリクエストがあることを意味します。
m_axil_arready	I	マスター読み出しアドレス準備完了。
m_axil_rdata[31:0]	I	マスター読み出しデータ。
m_axil_rresp[1:0]	I	マスター読み出し応答。
m_axil_rvalid	I	マスター読み出し有効。
m_axil_rready	O	マスター読み出し準備完了。

AXI4-Lite スレーブ ポート

表 39: コンフィギュレーション AXI4-Lite メモリ マップド書き込みスレーブ インターフェイス信号

信号名	I/O	説明
s_axil_awaddr[31:0]	I	この信号は、ユーザー ロジックから DMA へのメモリ マップド書き込み用のアドレスです。 s_axil_awaddr[31:28]: 4'b0011 - QDMA レジスタ 4'b0000 - ブリッジ レジスタ
s_axil_awvalid	I	この信号がアサートされていると、s_axil_awaddr のアドレスに対する有効な書き込みリクエストがあることを意味します。
s_axil_awuser	I	[7:0]: ファンクション番号
s_axil_awprot[2:0]	I	保護タイプ (未使用)。
s_axil_awready	O	スレーブ書き込みアドレスの準備完了。
s_axil_wdata[31:0]	I	スレーブ書き込みデータ。
s_axil_wstrb[3:0]	I	スレーブ書き込みストロブ。
s_axil_wvalid	I	スレーブ書き込み有効。
s_axil_wready	O	スレーブ書き込み準備完了。
s_axil_bvalid	O	スレーブ書き込み応答有効。

表 39: コンフィギュレーション AXI4-Lite メモリ マップド書き込みスレーブ インターフェイス信号 (続き)

信号名	I/O	説明
s_axil_bresp[1:0]	O	スレーブ書き込み応答。
s_axil_bready	I	スレーブ応答準備完了。

表 40: コンフィギュレーション AXI4-Lite メモリ マップド読み出しスレーブ インターフェイス信号

信号名	I/O	説明
s_axil_araddr[31:0]	I	この信号は、ユーザー ロジックから DMA へのメモリ マップド読み出し用のアドレスです。 s_axil_araddr[31:28]: 4'b0011 - QDMA レジスタ 4'b0000 - ブリッジ レジスタ
s_axil_arprot[2:0]	I	保護タイプ (未使用)。
s_axil_arvalid	I	この信号がアサートされていると、s_axil_araddr のアドレスに対する有効な読み出しリクエストがあることを意味します。
s_axil_aruser	I	[7:0]: ファンクション番号
s_axil_arready	O	スレーブ読み出しアドレスの準備完了。
s_axil_rdata[31:0]	O	スレーブ読み出しデータ。
s_axil_rresp[1:0]	O	スレーブ読み出し応答。
s_axil_rvalid	O	スレーブ読み出し有効。
s_axil_rready	I	スレーブ読み出し準備完了。

AXI4 メモリ マップド DMA ポート

表 41: AXI4 メモリ マップド DMA 読み出しアドレス インターフェイス信号

信号名	方向	説明
m_axi_araddr [C_M_AXI_ADDR_WIDTH-1:0]	O	この信号は、DMA からユーザー ロジックへのメモリ マップド読み出し用のアドレスです。
m_axi_arid [3:0]	O	標準 AXI4 は、AXI4 プロトコル仕様、AMBA AXI4-Stream Protocol Specification (ARM IHI 0051A) で規定されています。
m_axi_aruser[7:0]	O	[7:0]: ファンクション番号
m_axi_arlen[7:0]	O	マスター読み出しバーストの長さ。
m_axi_arsize[2:0]	O	マスター読み出しバーストのサイズ。
m_axi_arprot[2:0]	O	プロテクション タイプ。
m_axi_arvalid	O	この信号がアサートされていると、m_axi_araddr のアドレスに対する有効な読み出しリクエストがあることを意味します。
m_axi_arready	I	マスター読み出しアドレス準備完了。
m_axi_arlock	O	ロック タイプ。
m_axi_arcache[3:0]	O	メモリ タイプ。
m_axi_arburst[1:0]	O	マスター読み出しバーストのタイプ。

表 42: AXI4 メモリ マップド DMA 読み出しインターフェイス信号

信号名	方向	説明
m_axi_rdata [C_M_AXI_DATA_WIDTH-1:0]	I	マスター読み出しデータ。
m_axi_rid [3:0]	I	マスター読み出し ID。
m_axi_rresp[1:0]	I	マスター読み出し応答。
m_axi_rlast	I	マスター読み出し最終。
m_axi_rvalid	I	マスター読み出し有効。
m_axi_rready	O	マスター読み出し準備完了。
m_axi_ruser [C_M_AXI_DATA_WIDTH/8-1:0]	I	マスター読み出し奇数データパリティ (バイトごと)。このポートは、パリティ伝搬モードでのみイネーブルになります。

表 43: AXI4 メモリ マップド DMA 書き込みアドレス インターフェイス信号

信号名	方向	説明
m_axi_awaddr [C_M_AXI_ADDR_WIDTH-1:0]	O	この信号は、DMA からユーザー ロジックへのメモリ マップド書き込み用のアドレスです。
m_axi_awid[3:0]	O	マスター書き込みアドレス ID。
m_axi_aruser[7:0]	O	[7:0]: ファンクション番号
m_axi_awlen[7:0]	O	マスター書き込みアドレスの長さ。
m_axi_awsz[2:0]	O	マスター書き込みアドレスのサイズ。
m_axi_awburst[1:0]	O	マスター書き込みアドレスのバースト タイプ。
m_axi_awprot[2:0]	O	プロテクション タイプ。
m_axi_awvalid	O	この信号がアサートされていると、m_axi_araddr のアドレスに対する有効な書き込みリクエストがあることを意味します。
m_axi_awready	I	マスター書き込みアドレス準備完了。
m_axi_awlock	O	ロック タイプです。
m_axi_awcache[3:0]	O	メモリ タイプ。

表 44: AXI4 メモリ マップド DMA 書き込みインターフェイス信号

信号名	方向	説明
m_axi_wdata [C_M_AXI_DATA_WIDTH-1:0]	O	マスター書き込みデータ。
m_axi_wlast	O	マスター書き込み最終。
m_axi_wstrb[31:0]	O	マスター書き込みストロブ。
m_axi_wvalid	O	マスター書き込み有効。
m_axi_wready	I	マスター書き込み準備完了。
m_axi_wuser [C_M_AXI_DATA_WIDTH/8-1:0]	O	マスター書き込みユーザー。 m_axi_wuser[C_M_AXI_DATA_WIDTH/8-1:0] = 書き込みデータの奇数パリティ (バイトごと)。このポートは、パリティ伝搬モードでのみイネーブルになります。

表 45: AXI4 メモリ マップド DMA 書き込み応答インターフェイス信号

信号名	方向	説明
m_axi_bvalid	I	マスター書き込み応答有効。
m_axi_bresp[1:0]	I	マスター書き込み応答。
m_axi_bid[3:0]	I	マスター応答 ID。
m_axi_bready	O	マスター応答準備完了。

AXI4-Stream H2C ポート

表 46: AXI4-Stream H2C ポートの説明

ポート名	I/O	説明
m_axis_h2c_tdata [AXI_DATA_WIDTH-1:0]	O	H2C AXI4-Stream のデータ出力です。
m_axis_h2c_dpar [AXI_DATA_WIDTH/8-1:0]	O	m_axis_h2c_tdata でバイトごとのビットで計算された奇数パリティです。 m_axis_h2c_dpar[0] は m_axis_h2c_tdata[7:0] で計算されたパリティ、 m_axis_h2c_dpar[1] は m_axis_h2c_tdata[15:8] で計算されたパリティなど。
m_axis_h2c_tuser_qid[10:0]	O	キュー ID。
m_axis_h2c_tuser_port_id[2:0]	O	ポート ID
m_axis_h2c_tuser_err	O	セットされている場合、パケットにエラーがあることを示します。エラーは PCIe からのものであるか、QDMA でダブルビットエラーが発生しています。
m_axis_h2c_tuser_mdata[31:0]	O	メタデータ。 内部モードの場合、このフィールドの H2C AXI4-Stream ディスクリプターの低位 32 ビットが QDMA により渡されます。
m_axis_h2c_tuser_mty[5:0]	O	トランザクションの最終ビットで無効なバイト数。このフィールドは 64B 転送の場合 0 です。
m_axis_h2c_tuser_zero_byte	O	セットされている場合、現在のビットが空 (0 バイトが転送されている) であることを示します。
m_axis_h2c_tvalid	O	有効信号
m_axis_h2c_tlast	O	パケット転送の最終サイクルであることを示します。
m_axis_h2c_tready	I	レディ信号

AXI4-Stream C2H ポート

表 47: AXI4-Stream C2H ポートの説明

ポート名	I/O	説明
s_axis_c2h_tdata [AXI_DATA_WIDTH-1:0]	I	64、128、256、および 512 ビットの 4 つのデータ幅がサポートされています。どの C2H データ パケットにも対応する C2H 完了パケットがあります。
s_axis_c2h_dpar [AXI_DATA_WIDTH/8-1:0]	I	バイトごとのビットとして計算される奇数パリティです。
s_axis_c2h_ctrl_len [15:0]	I	パケットの長さです。0 バイトの書き込みの場合、長さは 0 です。
s_axis_c2h_ctrl_qid [10:0]	I	キュー ID。

表 47: AXI4-Stream C2H ポートの説明 (続き)

ポート名	I/O	説明
s_axis_c2h_ctrl_has_cmpt	I	1'b1: データ パケットには完了があります。 1'b0: データ パケットには完了がありません。
s_axis_c2h_ctrl_marker	I	パイプラインが完全にフラッシュされていることを確認するため使用されるマーカー メッセージです。この後、キューを安全に無効化できません。このビットがセットされている場合は、imm_data ビットもセットする必要があります。
s_axis_c2h_ctrl_port_id [2:0]	I	ポート ID。
s_axis_c2h_rmt [5:0]	I	最終データ パケットの空バイト。
s_axis_c2h_tvalid	I	有効。
s_axis_c2h_tlast	I	最終パケットであることを示します。
s_axis_c2h_tready	O	レディ 信号。

AXI4-Stream C2H 完了ポート

表 48: AXI4-Stream C2H 完了ポートの説明

ポート名	I/O	説明
s_axis_c2h_cmpt_tdata[511:0]	I	ユーザー アプリケーションからの完了データです。ホストの完了リングに書き込まれる情報が含まれています。
s_axis_c2h_cmpt_size [1:0]	I	00: 8B 完了。 01: 16B 完了。 10: 32B 完了。 11: 64B 完了。
s_axis_c2h_cmpt_dpar [15:0]	I	32 ビットごとにビットとして計算される奇数パリティです。 s_axis_c2h_cmpt_dpar[0] は s_axis_c2h_cmpt_tdata[31:0] で計算されたパリティ、 s_axis_c2h_cmpt_dpar[1] は s_axis_c2h_cmpt_tdata[63:31] で計算されたパリティなど。
s_axis_c2h_cmpt_ctrl_qid[10:0]	I	完了キュー ID。
s_axis_c2h_cmpt_ctrl_marker	I	パイプラインが完全にフラッシュされていることを確認するため使用されるマーカー メッセージです。この後、キューを安全に無効化できません。
s_axis_c2h_cmpt_ctrl_user_trig	I	イネーブルになっている場合は、割り込みおよびステータス ディスクリプター書き込みをトリガーできます。
s_axis_c2h_cmpt_ctrl_cmpt_type[1:0]	I	2'b00: NO_PLD_NO_WAIT。CMPT パケットには対応するペイロード パケットがなく、待機する必要はありません。 2'b01: NO_PLD_BUT_WAIT。CMPT パケットには対応するペイロード パケットがありませんが、それでも CMPT パケット送信前にペイロード パケットが送信されるのを待つ必要があります。 2'b10: RSVD。 2'b11: HAS_PLD。CMPT パケットには対応するペイロード パケットがなく、CMPT パケット送信前にペイロード パケットが送信されるのを待つ必要があります。
s_axis_c2h_cmpt_ctrl_wait_pld_pkt_id[15:0]	I	送信できるようになるまで CMPT パケットは待機する必要があることを示すデータ ペイロード パケット ID。
s_axis_c2h_cmpt_ctrl_port_id[2:0]	I	ポート ID。

表 48: AXI4-Stream C2H 完了ポートの説明 (続き)

ポート名	I/O	説明
s_axis_c2h_cmpt_ctrl_col_idx[2:0]	I	CMPT パケットにカラー ビットをセットする必要があるかどうかを定義し、また、カラー ビットがセットされている場合のビット位置を定義するカラー インデックス。
s_axis_c2h_cmpt_ctrl_err_idx[2:0]	I	CMPT パケットにエラー ビットをセットする必要があるかどうかを定義し、また、エラー ビットがセットされている場合のビット位置を定義するエラー インデックス。
s_axis_c2h_cmpt_tvalid	I	有効。
s_axis_c2h_cmpt_tready	O	レディ信号。

AXI4-Stream ステータス ポート

表 49: AXI-ST C2H ステータス ポートの説明

ポート名	I/O	説明
axis_c2h_status_valid	O	ディスクリプターごとに有効。
axis_c2h_status_qid [10:0]	O	パケットの QID。
axis_c2h_status_drop	O	QDMA Subsystem for PCIe に C2H パケットを格納するのに十分なデータバッファがない場合、またはフル パケットをホストに転送するのに十分なディスクリプターがない場合、パケットが破棄されます。このビットは、パケットが破棄されたかどうかを示します。破棄されていないパケットは受領されたとみなされます。 0: パケットは破棄されていません。 1: パケットは破棄されています。
axis_c2h_status_last	O	最終ディスクリプター。
axis_c2h_status_cmp	O	0: 破棄されたパケット、または has_cmpt が 1'b0 の C2H パケット。 1: 完了を含む C2H パケット。

AXI4-Stream C2H 書き込み完了ポート

表 50: AXI-ST C2H 書き込み完了ポートの説明

ポート名	I/O	説明
axis_c2h_dmawr_cmp	O	パケットの最終データ ペイロードの WRQ が WCP を完了受信すると、この信号がアサートされます。パケットごとに 1 パルスです。

VDM ポート

表 51: VDM ポートの説明

ポート名	I/O	説明
st_rx_msg_valid	O	有効信号

表 51: VDM ポートの説明 (続き)

ポート名	I/O	説明
st_rx_msg_data	O	ビート 1: {REQ_ID[15:0], VDM_MSG_CODE[7:0], VDM_MSG_ROUTING[2:0], VDM_DW_LENGTH[4:0]} ビート 2: VDM ヘッダー [31:0] または {(Payload_length=0), VDM Higher Header [31:0]} ビート 3 からビート <n>: VDM ペイロード
st_rx_msg_last	O	最終ビートを示します。
st_rx_msg_rdy	I	レディ信号。このインターフェイスを使用しない場合は、レディを 1 に接続する必要があります。 注記: このインターフェイスを使用しない場合は、レディを 1 に接続する必要があります。

コンフィギュレーション拡張インターフェイス ポート

外部にコンフィギュレーションレジスタがインプリメントされている場合、コンフィギュレーション拡張インターフェイスを使用すると、コンフィギュレーション情報をコアからユーザーアプリケーションに送信できます。

表 52: コンフィギュレーション拡張インターフェイス ポートの説明

ポート名	I/O	幅	説明
cfg_ext_read_received	O	1	コンフィギュレーション拡張インターフェイスの読み出しリクエストの受信 コアは、リンクからコンフィギュレーション読み出しリクエストを受信すると、この出力をアサートします。Vivado® IDE の [User Defined Configuration Capabilities] タブで [PCI Express Extended Configuration Space Enable] がオンになっている場合にセットされます。 <ul style="list-style-type: none"> 受信されたコンフィギュレーション読み出しで <code>cfg_ext_register_number</code> が 0xb0 から 0xbf にあるものはすべて、PCIe レガシ拡張コンフィギュレーション空間だとみなされます。 受信されたコンフィギュレーション読み出しで <code>cfg_ext_register_number</code> が 0x120 から 13F にあるものはすべて、PCIe 拡張コンフィギュレーション空間だとみなされます。 <code>cfg_ext_read_received</code> が 1 サイクル間アサートされると、アドレスに関係なく、コンフィギュレーション読み出しがすべて受信されたことを示します。有効データは <code>cfg_ext_register_number</code> および <code>cfg_ext_function_number</code> で駆動されます。 受信されたコンフィギュレーション読み出しで、上記の 2 つの範囲内にある読み出しのみ、IP 外のユーザーアプリケーションからの応答が必要になります。

表 52: コンフィギュレーション拡張インターフェイス ポートの説明 (続き)

ポート名	I/O	幅	説明
cfg_ext_write_received	O	1	<p>コンフィギュレーション拡張インターフェイスの書き込みリクエスト受信</p> <p>コアは、リンクからコンフィギュレーション書き込みリクエストを受信すると、この出力をアサートします。Vivado IDE の [Capabilities] タブで [PCI Express Extended Configuration Space Enable] がオンになっている場合にセットされます。</p> <ul style="list-style-type: none"> cfg_ext_register_number が 0xb0 から 0xbf の範囲にある受信されたコンフィギュレーション書き込みに対応するデータは、cfg_ext_register_number、cfg_ext_function_number、cfg_ext_write_data、および cfg_ext_write_byte_enable に出力されます。 受信されたコンフィギュレーション書き込みで、cfg_ext_register_number が 0x120 から 13F の範囲にあるものはすべて、cfg_ext_register_number、cfg_ext_function_number、cfg_ext_write_data、および cfg_ext_write_byte_enable に出力されます。
cfg_ext_register_number	O	10	<p>コンフィギュレーション拡張インターフェイスのレジスタ番号読み出しまたは書き込みが行われるコンフィギュレーション レジスタの 10 ビット アドレスです。cfg_ext_read_received または cfg_ext_write_received が High のとき、データは有効です。</p>
cfg_ext_function_number	O	8	<p>コンフィギュレーション拡張インターフェイスのファンクション番号を示します。</p> <p>コンフィギュレーション読み出しまたは書き込みリクエストに対応する 8 ビットのファンクション番号です。</p> <p>cfg_ext_read_received または cfg_ext_write_received が High のとき、データは有効です。</p>
cfg_ext_write_data	O	32	<p>コンフィギュレーション拡張インターフェイスの書き込みデータ</p> <p>コンフィギュレーション レジスタに書き込まれるデータです。この出力は、cfg_ext_write_received が High の場合に有効です。</p>
cfg_ext_write_byte_enable	O	4	<p>コンフィギュレーション拡張インターフェイスの書き込みバイトイネーブル</p> <p>コンフィギュレーション書き込みトランザクション用のバイトイネーブル信号です。</p>
cfg_ext_read_data	I	32	<p>コンフィギュレーション拡張インターフェイスの読み出しデータ</p> <p>このバスを使用して、外部にインプリメントされたコンフィギュレーション レジスタからコアヘデータを供給できます。</p> <p>cfg_ext_read_data_valid がセットされている場合は、cfg_ext_read_received が High になった後に、クロックの次の立ち上がりエッジでこのデータがコアによりサンプルされます。</p>
cfg_ext_read_data_valid	I	1	<p>コンフィギュレーション拡張インターフェイスの読み出しデータ有効</p> <p>ユーザー アプリケーションは、コアにこの入力のアサートして、外部にインプリメントされたコンフィギュレーション レジスタからデータを供給します。cfg_ext_read_received が High になった後、クロックの次の立ち上がりエッジで、この入力データがコアによりサンプルされます。cfg_ext_read_received 信号で読み出しリクエストを受信した後、ユーザー クロックの 262144 ('h4_0000) クロック サイクル内で、この信号がアサートされるはずですが、この時点までで応答がなかった場合は、'h0 のペイロードでコアから自動応答が送信されるので、ユーザー アプリケーションで応答を破棄して、そのリクエストを即時終了させる必要があります。</p>

FLR ポート

表 53: FLR ポートの説明

ポート名	I/O	説明
usr_flr_fnc [7:0]	O	機能 FLR ステータス変更のファンクション番号です。
usr_flr_set	O	セット 1 サイクル間アサートされた場合、usr_flr_fnc[7:0] 上のファンクションの FLR ステータスがアクティブであることを示します。
usr_flr_clr	O	クリア 1 サイクル間アサートされると、usr_flr_fnc[7:0] 上のファンクションの FLR ステータスが完了したことを示します。
usr_flr_done_fnc [7:0]	I	完了ファンクション どの FLR のファンクションがユーザー ロジックによって完了したのかを示します。
usr_flr_done_vld	I	完了有効 1 サイクル間アサートされると、usr_flr_done_fnc[7:0] 上のファンクションの FLR が完了したことを示します。

QDMA ディスクリプター バイパス入力ポート

表 54: QDMA H2C ストリーミング バイパス入力ポートの説明

ポート名	I/O	説明
h2c_byp_in_st_addr [63:0]	I	DMA 転送の 64 ビットの開始アドレス。
h2c_byp_in_st_len [15:0]	I	転送するバイト数。
h2c_byp_in_st_at [1:0]		アドレス タイプ。 2'b00: リクエスト内のアドレスは変換されません。 2'b01: 予約。 2'b10: リクエスト内のアドレスは変換されます。 2'b11: 予約。
h2c_byp_in_st_sop	I	パケットの開始を示します。最初のディスクリプターにセットします。それ以外のディスクリプターにはリセットします。
h2c_byp_in_st_eop	I	パケットの終了を示します。最終のディスクリプターにセットします。それ以外のディスクリプターにはリセットします。
h2c_byp_in_st_sdi	I	H2C バイパス入力ステータス ディスクリプター/割り込み セットされている場合、ユーザー アプリケーションから QDMA に対して、ステータス ディスクリプターをホストに送信し、QDMA がこのディスクリプターに関連付けられているデータの最終バイトをフェッチしたときにホストに割り込みを生成するよう指示されます。この QID の H2C ソフトウェア コンテキストで割り込みがイネーブルになっていて、ドライバーで保護されている場合にのみ、QDMA が割り込み生成のリクエストを受け入れます。これは EOP ディスクリプターに対してのみセット可能です。
h2c_byp_in_st_mrkr_req	I	H2C バイパス入力マーカールクエスト セットされている場合、ディスクリプターは H2C エンジンパイプラインを通過し、通過し終わると、H2C ストリーミング バイパス出力インターフェイスでマーカール応答を生成します。これは EOP ディスクリプターに対してのみセット可能です。

表 54: QDMA H2C ストリーミング バイパス入力ポートの説明 (続き)

ポート名	I/O	説明
h2c_byp_in_st_no_dma	I	H2C バイパス入力 No DMA この信号をアサートした状態でこのインターフェイスを介してディスクリプターを送信すると、このディスクリプターの PCIe リクエストを送信しないように、この信号が QDMA に知らせます。PCIe リクエストが送信されないため、対応する DMA データも H2C ストリーミング出力インターフェイスで出力されません。 これは通常、ユーザー ロジックに実際のディスクリプターがないにもかかわらず h2c_byp_in_st_sdi 信号を駆動する必要があるとき、ステータス ディスクリプター/割り込みが送信されるように、h2c_byp_in_st_sdi と共に使用されます。 No DMA ディスクリプターが送信されるときに h2c_byp_in_st_mrkr_req および h2c_byp_in_st_sdi がリセットされていると、そのディスクリプターは NOP として処理され、インターフェイス アクティビティなしに QDMA 内で完全に消費されます。 h2c_byp_in_st_no_dma がセットされると、h2c_byp_in_st_sop および h2c_byp_in_st_eop の両方をセットする必要があります。 h2c_byp_in_st_no_dma がセットされると、QDMA はこのインターフェイスのアドレスおよび長さのフィールドを無視します。
h2c_byp_in_st_qid [10:0]	I	H2C ディスクリプター リングに関連付けられている QID。
h2c_byp_in_st_error	I	このビットは、キューにエラーがあることを示すためにセットされます。ディスクリプターは処理されません。キューにエラーがあることを反映させるため、コンテキストがアップデートされます。
h2c_byp_in_st_func [7:0]	I	PCIE ファンクション ID
h2c_byp_in_st_cidx [15:0]	I	ステータス ディスクリプター アップデート/割り込み用に使用される CIDX (アグリゲーション モード)。一般的には、ディスクリプター バイパス出力インターフェイスから受信された時点から、CIDX は変更せずにそのままにしておきます。
h2c_byp_in_st_port_id [2:0]	I	QDMA ポート ID
h2c_byp_in_st_vld	I	有効。High の場合は、ディスクリプターが有効で、1 ディスクリプター 1 パルスであることを示します。
h2c_byp_in_st_rdy	O	ディスクリプター受信準備完了

表 55: QDMA H2C メモリ マップド ディスクリプター バイパス入力ポートの説明

ポート名	I/O	説明
h2c_byp_in_mm_radr[63:0]	I	DMA データの読み出しアドレス。
h2c_byp_in_mm_wadr[63:0]	I	DMA データの書き込みアドレス。
h2c_byp_in_mm_at[1:0]	I	アドレス タイプ。 2'b00: リクエスト内のアドレスは変換されません。 2'b01: 予約。 2'b10: リクエスト内のアドレスは変換されます。 2'b11: 予約。

表 55: QDMA H2C メモリ マップド ディスクリプター バイパス入力ポートの説明 (続き)

ポート名	I/O	説明
h2c_byp_in_mm_no_dma	I	<p>H2C バイパス入力 No DMA</p> <p>この信号をアサートした状態でこのインターフェイスを介してディスクリプターを送信すると、このディスクリプターの PCIe リクエストを送信しないように、この信号が QDMA に知らせます。PCIe リクエストが送信されないため、対応する DMA データも H2C メモリ マップド出力インターフェイスで出力されません。</p> <p>これは通常、ユーザー ロジックに実際のディスクリプターがないにもかかわらず h2c_byp_in_mm_sdi 信号を駆動する必要があるとき、ステータス ディスクリプター/割り込みが送信されるように、h2c_byp_in_mm_sdi と共に使用されます。</p> <p>No DMA ディスクリプターが送信されるときに h2c_byp_in_mm_mrkr_req および h2c_byp_in_mm_sdi がリセットされていると、そのディスクリプターは NOP (No Operation) として処理され、インターフェイス アクティビティなしに QDMA 内で完全に消費されます。</p> <p>h2c_byp_in_mm_no_dma がセットされると、QDMA はアドレスを無視します。フィールドは 0 に設定してください。</p>
h2c_byp_in_mm_len[27:0]	I	<p>DMA データの長さです。</p> <p>上位 12 ビットは 0 に固定しておく必要があります。したがって、長さ指定には下位 16 ビットのみが使用されます。</p>
h2c_byp_in_mm_sdi	I	<p>H2C メモリ マップド バイパス入力ステータス ディスクリプター/割り込み</p> <p>セットされている場合、ユーザーから QDMA に対して、ステータス ディスクリプターをホストに送信し、QDMA がこのディスクリプターに関連付けられているデータの最終バイトをフェッチしたときにホストに割り込み生成するよう指示されます。この QID の H2C リング コンテキストで割り込みがイネーブルになっていて、ドライバで保護されている場合にのみ、QDMA が割り込み生成のリクエストを受け入れます。</p>
h2c_byp_in_mm_mrkr_req	I	<p>H2C メモリ マップド バイパス入力完了リクエスト</p> <p>QDMA がこのディスクリプターのデータ転送を完了させると、ユーザーに完了ステータスを送信する必要があることをユーザーに知らせます。</p>
h2c_byp_in_mm_qid [10:0]	I	H2C ディスクリプター リングに関連付けられている QID。
h2c_byp_in_mm_error	I	<p>このビットは、キューにエラーがあることを示すためにセットされます。ディスクリプターは処理されません。キューにエラーがあることを反映させるため、コンテキストがアップデートされます。</p>
h2c_byp_in_mm_func [7:0]	I	PCIe ファンクション ID
h2c_byp_in_mm_cidx [15:0]	I	<p>ステータス ディスクリプター アップデート/割り込み用に使用される CIDX (アグリゲーション モード)。一般的には、ディスクリプター バイパス出力インターフェイスから受信された時点から、CIDX は変更せずにそのままにしておきます。</p>
h2c_byp_in_mm_port_id [2:0]	I	QDMA ポート ID
h2c_byp_in_mm_vld	I	有効。High の場合は、ディスクリプターが有効で、1 ディスクリプター 1 パルスであることを示します。
h2c_byp_in_mm_rdy	O	ディスクリプター受信準備完了

表 56: QDMA C2H ストリーミング単純バイパス入力ポートの説明

ポート名	I/O	説明
c2h_byp_in_st_sim_addr [63:0]	I	QDMA が DMA データを書き込む 64 ビット アドレス。
c2h_byp_in_st_sim_qid [10:0]	I	C2H ディスクリプター リングに関連付けられている QID。

表 56: QDMA C2H ストリーミング単純バイパス入力ポートの説明 (続き)

ポート名	I/O	説明
c2h_byp_in_st_sim_at [1:0]		アドレス タイプ。 2'b00: リクエスト内のアドレスは変換されません。 2'b01: 予約。 2'b10: リクエスト内のアドレスは変換されます。 2'b11: 予約。
c2h_byp_in_st_sim_error	I	このビットは、キューにエラーがあることを示すためにセットされます。ディスクリプターは処理されません。キューにエラーがあることを反映させるため、コンテキストがアップデートされます。
c2h_byp_in_st_sim_func [7:0]	I	PCIe ファンクション ID
c2h_byp_in_st_sim_port_id[2:0]	I	QDMA ポート ID
c2h_byp_in_st_sim_vld	I	有効。High の場合は、ディスクリプターが有効で、1 ディスクリプター 1 パルスであることを示します。
c2h_byp_in_st_sim_rdy	O	ディスクリプター受信準備完了

表 57: QDMA C2H ストリーミング キャッシュ バイパス入力ポートの説明

ポート名	I/O	説明
c2h_byp_in_st_csh_addr [63:0]	I	QDMA が DMA データを書き込む 64 ビット アドレス。
c2h_byp_in_st_csh_qid [10:0]	I	C2H ディスクリプター リングに関連付けられている QID。
c2h_byp_in_st_csh_at [1:0]		アドレス タイプ。 2'b00: リクエスト内のアドレスは変換されません。 2'b01: 予約。 2'b10: リクエスト内のアドレスは変換されます。 2'b11: 予約。
c2h_byp_in_st_csh_error	I	このビットは、キューにエラーがあることを示すためにセットされます。ディスクリプターは処理されません。キューにエラーがあることを反映させるため、コンテキストがアップデートされます。
c2h_byp_in_st_csh_func [7:0]	I	PCIe ファンクション ID
c2h_byp_in_st_csh_port_id[2:0]	I	QDMA ポート ID
c2h_byp_in_st_csh_vld	I	有効。High の場合は、ディスクリプターが有効で、1 ディスクリプター 1 パルスであることを示します。
c2h_byp_in_st_csh_rdy	O	ディスクリプター受信準備完了

表 58: QDMA C2H メモリ マップド ディスクリプター バイパス入力ポートの説明

ポート名	I/O	説明
c2h_byp_in_mm_raddr [63:0]	I	DMA データの読み出しアドレス。
c2h_byp_in_mm_waddr[63:0]	I	DMA データの書き込みアドレス。
c2h_byp_in_mm_at [1:0]	I	アドレス タイプ 2'b00: リクエスト内のアドレスは変換されません。 2'b01: 予約。 2'b10: リクエスト内のアドレスは変換されます。 2'b11: 予約。

表 58: QDMA C2H メモリ マップド ディスクリプター バイパス入力ポートの説明 (続き)

ポート名	I/O	説明
c2h_byp_in_mm_no_dma	I	C2H バイパス入力 No DMA この信号をアサートした状態でこのインターフェイスを介してディスクリプターを送信すると、このディスクリプターの PCIe リクエストを送信しないように、この信号が QDMA に知らせます。PCIe リクエストが送信されないため、対応する DMA データも C2H メモリ マップド インターフェイスから読み出されません。 これは通常、ユーザー ロジックに実際のディスクリプターがないにもかかわらず c2h_byp_in_mm_sdi 信号を駆動する必要があるとき、ステータス ディスクリプター/割り込みが送信されるように、c2h_byp_in_mm_sdi と共に使用されます。 No DMA ディスクリプターが送信されるときに c2h_byp_in_mm_mrkr_req および c2h_byp_in_mm_sdi がリセットされていると、そのディスクリプターは NOP として処理され、インターフェイス アクティビティなしに QDMA 内で完全に消費されます。 c2h_byp_in_mm_no_dma がセットされると、QDMA はアドレスを無視します。フィールドは 0 に設定してください。
c2h_byp_in_mm_len[27:0]	I	DMA データの長さ
c2h_byp_in_mm_sdi	I	C2H バイパス入力ステータス ディスクリプター/割り込み セットされている場合、ユーザーから QDMA に対して、ステータス ディスクリプターをホストに送信し、QDMA がこのディスクリプターに関連付けられているデータの最終バイトをフェッチしたときにホストに割り込み生成するよう指示されます。この QID の C2H リング コンテキストで割り込みがイネーブルになっていて、ドライバーで保護されている場合にのみ、QDMA が割り込み生成のリクエストを受け入れます。
c2h_byp_in_mm_mrkr_req	I	C2H バイパス入力マーカー リクエスト QDMA がこのディスクリプターのデータ転送を完了させると、ユーザーに完了ステータスを送信する必要があることをユーザーに知らせます。
c2h_byp_in_mm_qid [10:0]	I	C2H ディスクリプター リングに関連付けられている QID。
c2h_byp_in_mm_error	I	このビットは、キューにエラーがあることを示すためにセットされます。ディスクリプターは処理されません。キューにエラーがあることを反映させるため、コンテキストがアップデートされます。
c2h_byp_in_mm_func [7:0]	I	PCIe ファンクション ID
c2h_byp_in_mm_cidx [15:0]	I	バイパス出力インターフェイスで受信したディスクリプターから CIDX をユーザーはエコーする必要があります。
c2h_byp_in_mm_port_id[2:0]	I	QDMA ポート ID
c2h_byp_in_mm_vld	I	有効。High の場合は、ディスクリプターが有効で、1 ディスクリプター 1 パルスであることを示します。
c2h_byp_in_mm_rdy	O	ディスクリプター受信準備完了

QDMA ディスクリプター バイパス出力ポート

表 59: QDMA H2C ディスクリプター バイパス出力ポートの説明

ポート名	I/O	説明
h2c_byp_out_dsc [255:0]	O	ホストからフェッチされた H2C ディスクリプター。ストリーミング ディスクリプターの場合は、このフィールドの下位 64b をアドレスとして使用します。残りのビットは無視できます。
h2c_byp_out_mrkr_rsp	O	h2c_byp_in_st_mrkr_req (ストリーム) または h2c_byp_in_mm_mrkr_req (メモリ マップド) への応答での完了ステータスを示します。

表 59: QDMA H2C ディスクリプター バイパス出力ポートの説明 (続き)

ポート名	I/O	説明
h2c_byp_out_st_mm	O	ストリーミング データ ディスクリプターなのか、メモリ マップド ディスクリプターなのかを示します。 0: ストリーミング 1: メモリ マップド
h2c_byp_out_dsc_sz [1:0]	O	ディスクリプター サイズ。h2c_byp_out_dsc 上の有効なディスクリプター情報量を示します。 0: 8B 1: 16B 2: 32B 3: 64B - 64B のディスクリプターは有効/準備完了 2 サイクルで転送されます。最初のサイクルで最下位から 32 バイト、2 番目のサイクルで最上位から 32 バイトが転送されます。どちらのサイクルでも、CIDX などのキュー情報は同じです。
h2c_byp_out_qid [10:0]	O	H2C ディスクリプター リングに関連付けられている QID。
h2c_byp_out_error	O	ディスクリプター フェッチまたは先行ディスクリプターの実行でエラーが発生したことを示します。
h2c_byp_out_func [7:0]	O	PCIE ファンクション ID
h2c_byp_out_cidx [15:0]	O	H2C バイパス出力コンシューマー インデックス フェッチされたディスクリプターのリング インデックス。バイパス入力インターフェイスでディスクリプターを出力するときは、このフィールドを QDMA にエコー バックする必要があります。
h2c_byp_out_port_id [2:0]	O	QDMA ポート ID
h2c_byp_out_vld	O	有効。High の場合は、ディスクリプターが有効で、1 ディスクリプター 1 パルスであることを示します。
h2c_byp_out_rdy	I	レディ 信号。このインターフェイスを使用しない場合は、レディ 信号を 1 に接続する必要があります。

表 60: QDMA C2H ディスクリプター バイパス出力ポートの説明

ポート名	I/O	説明
c2h_byp_out_dsc [255:0]	O	ホストからフェッチされた C2H ディスクリプター。ストリーミング ディスクリプターの場合は、このフィールドの下位 64b をアドレスとして使用します。残りのビットは無視できます。 c2h_byp_out_mrkr_rsp がアサートされている場合の応答については、次の表を参照してください。
c2h_byp_out_mrkr_rsp	O	s_axis_c2h_ctrl_marker (ストリーム) または c2h_byp_in_mm_mrkr_req (メモリ マップド) への応答での完了ステータスを示します。s_axis_c2h_ctrl_marker (ストリーム) の完了ステータスの場合は、次の表を参照してください。
c2h_byp_out_st_mm	O	ストリーミング データ ディスクリプターなのか、メモリ マップド ディスクリプターなのかを示します。 0: ストリーミング 1: メモリ マップド

表 60: QDMA C2H ディスクリプター バイパス出力ポートの説明 (続き)

ポート名	I/O	説明
c2h_byp_out_dsc_sz [1:0]	O	ディスクリプター サイズ。h2c_byp_out_dsc 上の有効なディスクリプター情報量を示します。 0: 8B 1: 16B 2: 32B 3: 64B - 64B のディスクリプターは有効/準備完了 2 サイクルで転送されます。最初のサイクルで最下位から 32 バイト、2 番目のサイクルで最上位から 32 バイトが転送されます。どちらのサイクルでも、CIDX などのキュー情報は同じです。
c2h_byp_out_qid [10:0]	O	H2C ディスクリプター リングに関連付けられている QID。
c2h_byp_out_error	O	ディスクリプター フェッチまたは先行ディスクリプターの実行でエラーが発生したことを示します。
c2h_byp_out_func [7:0]	O	PCIe ファンクション ID。
c2h_byp_out_cidx [15:0]	O	C2H バイパス出力コンシューマー インデックス フェッチされたディスクリプターのリング インデックス。バイパス入力インターフェイスでディスクリプターを出力するときは、このフィールドを QDMA にエコー バックする必要があります。
c2h_byp_out_port_id [2:0]	O	QDMA ポート ID
c2h_byp_out_vld	O	有効。High の場合は、ディスクリプターが有効で、1 ディスクリプター 1 パルスであることを示します。
c2h_byp_out_rdy	I	レディ信号。このインターフェイスを使用しない場合は、レディ信号を 1 に接続する必要があります。

表 61: QDMA C2H ディスクリプター バイパス出力マーカール応答の説明

フィールド名	位置	説明
err[1:0]	[1:0]	C2H エンジンによってレポートされるエラー コード。 0: エラーなし 1: ソフトウェアから無効な完了 CIDX アップデートを受信 2: C2H パケット処理中に受信されたディスクリプター エラー 3: 完了リングがフルだったために C2H エンジンによって破棄された完了
retry_marker_req	[2]	イネーブルになっていたのに割り込みが生成できなかったため、マーカール応答が完了できなかったことを示します。これは、マーカールクエストが受信されたとき、割り込みが既にキュー上で未処理だったときに発生します。ユーザー ロジックはマーカールクエストを再試行できるまで待機する必要があります。
	[255:3]	予約

QDMA ディスクリプター クレジット入力ポート

表 62: QDMA ディスクリプター クレジット入力ポートの説明

ポート名	I/O	説明
dsc_crdt_in_vld	I	有効。アサートされている場合、バスに有効データを出力し、同じサイクルで有効信号およびレディ信号がアサートされるまでそのバス値を維持する必要があります。

表 62: QDMA ディスクリプター クレジット入力ポートの説明 (続き)

ポート名	I/O	説明
dsc_crdt_in_rdy	O	レディ信号。この信号がアサートされると、DMA がこのバスからデータを受領する準備が整ったことを意味します。
dsc_crdt_in_dir	I	クレジットが H2C ディスクリプター リング用なのか C2H ディスクリプター リング用なのかを示します。 0: H2C 1: C2H
dsc_crdt_in_fence	I	フェンス ビットがセットされている場合、クレジットは連結されず、キューは、後続のクレジット アップデートが処理される前にディスクリプター フェッチを必ず生成します。イネーブルになっていて、ディスクリプターおよびクレジットの両方が使用できるようになっているキューに対してのみ、フェンス ビットをセットする必要があります。そうでないと、ハングする可能性があります。
dsc_crdt_in_qid [10:0]	I	クレジットが追加されている場合のディスクリプター リングに関連付けられている QID。
dsc_crdt_in_crd [15:0]	I	ホストからディスクリプターをフェッチするため、ユーザー アプリケーションが QDMA Subsystem for PCIe に出力しているディスクリプタークレジットの数。

QDMA トラフィック マネージャー クレジット出力ポート

表 63: QDMA トラフィック マネージャー (TM) クレジット出力ポートの説明

ポート名	I/O	説明
tm_dsc_sts_vld	O	有効。出力バスのデータが有効であることを示します。tm_dsc_sts_rdy がユーザーによってアサートされるまで、バスの有効データは保持されます。
tm_dsc_sts_rdy	I	レディ信号。アサートされている場合、ユーザー ロジックでこのバスのデータを受領する準備が整っていることを示します。 注記: このインターフェイスを使用しない場合は、レディ信号を 1 に接続する必要があります。
tm_dsc_sts_byp	O	ソフトウェア ディスクリプター コンテキストのバイパス ビットを示します。
tm_dsc_sts_dir	O	ステータス アップデートが H2C ディスクリプター リング用なのか C2H ディスクリプター リング用なのかを示します。 0: H2C 1: C2H
tm_dsc_sts_mm	O	ステータス アップデートがストリーミング キュー用なのかメモリ マップド キュー用なのかを示します。 0: ストリーミング 1: メモリ マップド
tm_dsc_sts_qid [10:0]	O	リングの QID
tm_dsc_sts_avl [15:0]	O	tm_dsc_sts_qinv がセットされている場合は、ディスクリプター エンジンで使用可能なクレジット数です。tm_dsc_sts_qinv がセットされていない場合は、前回このアップデートが送信されて以降、新しくリングにポストされたディスクリプターの数です。
tm_dsc_sts_qinv	O	セットされている場合、キューが無効化されていることを示します。ユーザー アプリケーションと QDMA の間のクレジット カウントを仲裁するため、ユーザー アプリケーションによって使用されます。
tm_dsc_sts_qen	O	現在のキューのイネーブル ステータス。

表 63: QDMA トラフィック マネージャー (TM) クレジット出力ポートの説明 (続き)

ポート名	I/O	説明
tm_dsc_sts_irq_arm	O	セットされている場合、ドライバーで割り込みを受領する準備が整ったことをユーザーに通知します。
tm_dsc_sts_error	O	PIDX アップデートが関連付けられているキューの現在の CIDX を超えている場合は、1 にセットされます。
tm_dsc_sts_port_id [2:0]	O	キュー コンテキストからのキューに関連付けられているポート ID。

ユーザー割り込み

表 64: ユーザー割り込みポートの説明

ポート名	I/O	説明
usr_irq_in_vld	I	有効 アサートされた場合、バスのベクター、ファンクション、保留フィールドに関連付けられている割り込みが PCIe に生成される必要があることを示します。アサートされたら、usr_irq_out_ack が DMA によりアサートされるまで、usr_irq_in_vld を High に維持する必要があります。
usr_irq_in_vec [10:0]	I	ベクター MSIX ベクターを送信する必要があります。
usr_irq_in_fnc [7:0]	I	ファンクション 送信するベクターのファンクションです。
usr_irq_out_ack[4:0]	O	割り込み肯定応答 肯定応答ビットがアサートされると、割り込みがリンク上で送信されたことを示すため、ユーザー ロジックはこのパルスを待ってから、ほかの割り込み条件を出力する必要があります。
usr_irq_out_fail	O	割り込みフェール フェールがアサートされると、リンク上で送信される前に、割り込みリクエストが停止されたことを示します。

レジスタ空間

表 65: コンフィギュレーション レジスタ属性の定義

レジスタ属性	説明
NA	予約
RO	読み出し専用 - レジスタ ビットは読み出し専用で、ソフトウェアで変更できません。
RW	読み出し/書き込み - レジスタ ビットは読み出し/書き込みで、ソフトウェアでセットまたはクリアにして、目的のステータスにできます。
RW1C	1 を書き込んでステータスをクリア - レジスタ ビットは読み出されたときのステータスを示します。セット ビットは、1b を書き込むことでクリアになるステータス イベントを示します。RW1C ビットに 0b を書き込んでも何も起きません。
W1C	読み出し不可能な 1 を書き込んでステータスをクリア - 読み出されたとき、レジスタは 0 を返します。1b を書き込むと、そのビット インデックスのステータスがクリアになります。W1C ビットに 0b を書き込んでも何も起きません。

表 65: コンフィギュレーション レジスタ属性の定義 (続き)

レジスタ属性	説明
W1S	読み出し不可能な 1 を書き込んでセット - 読み出されたとき、レジスタは 0 を返します。1b を書き込むと、そのビット インデックスの制御セットがセットされます。W1S ビットに 0b を書き込んででも何も起きません。

QDMA PF アドレス レジスタ空間

表 66: QDMA PF アドレス レジスタ空間

ターゲット名	ベース (16 進数)	バイト サイズ (10 進数)	注記
QDMA_TRQ_SEL_GLBL1 (0x00000)	00000000	256	QDMA コンフィギュレーション CSR 空間
QDMA_TRQ_SEL_GLBL2 (0x00100)	00000100	256	ドライバーに可視化されている属性空間
QDMA_TRQ_SEL_GLBL (0x00200)	00000200	512	QDMA CSR 空間
QDMA_TRQ_SEL_FMAP (0x00400)	00000400	1024	ファンクションからキューへのマップ レジスタ空間
QDMA_TRQ_SEL_IND (0x00800)	00000800	512	間接コンテキスト レジスタ空間
QDMA_TRQ_SEL_C2H (0x00A00)	00000A00	512	カードからホストへのストリーミング レジスタ空間
QDMA_TRQ_SEL_H2C (0x00E00)	00000E00	512	ホストからカードへのストリーミング レジスタ空間
QDMA_TRQ_SEL_C2H_MM (0x1000)	00001000	256	カードからホストへの AXI-MM レジスタ空間
QDMA_TRQ_SEL_H2C_MM (0x1200)	00001200	256	ホストからカードへの AXI-MM レジスタ空間
QDMA_TRQ_EXT_0 (0x1400)	00001400	4096	予約
QDMA_PF_MAILBOX (0x2400)	00002400	16384	メールボックス/FLR レジスタ空間
QDMA_TRQ_EXT_1 (0x6400)	00006400	39936	予約
QDMA_TRQ_MSIX (0x10000)	00010000	32768	32 個の MSIX ベクターおよび PBA
QDMA_TRQ_SEL_QUEUE_PF (0x18000)	00018000	32768	PF ダイレクト QCSR (キューごとに 16B、ファンクションごとに最高 2048 個のキュー)

QDMA_TRQ_SEL_GLBL1 (0x00000)

表 67: QDMA_TRQ_SEL_GLBL1 (0x00000) レジスタ空間

レジスタ名	アドレス (16 進数)	説明
コンフィギュレーション ブロック識別子 (0x00)	0x00	コンフィギュレーション ブロック ID レジスタ
コンフィギュレーション ブロック BusDev (0x04)	0x04	バス デバイス ファンクション レジスタ
コンフィギュレーション ブロック PCIE 最大ペイロード サイズ (0x08)	0x08	最大ペイロード サイズ
コンフィギュレーション ブロック PCIE 最大読み出しリクエスト サイズ (0x0C)	0x0C	最大読み出しリクエスト サイズ
コンフィギュレーション ブロック システム ID (0x10)	0x10	システム ID レジスタ

表 67: QDMA_TRQ_SEL_GLBL1 (0x00000) レジスタ空間 (続き)

レジスタ名	アドレス (16 進数)	説明
コンフィギュレーション ブロック DRU イネーブル (0x14)	0x14	割り込みコンフィギュレーション レジスタ
コンフィギュレーション ブロック PCIE データ幅 (0x18)	0x18	PCIE データ幅レジスタ
コンフィギュレーション PCIE 制御 (0x1C)	0x1C	PCIE 制御レジスタ
コンフィギュレーション AXI ユーザー最大ペイロード サイズ (0x40)	0x40	AXI 最大ペイロード サイズ レジスタ
コンフィギュレーション AXI ユーザー最大読み出しリクエスト サイズ (0x44)	0x44	AXI 最大読み出しリクエスト レジスタ
コンフィギュレーション ブロック Misc 制御 (0x4C)	0x4C	その他の制御
コンフィギュレーション ブロック スクラッチ 7-0 (0x80-0x9C)	0x80-0x9C	汎用スクラッチ レジスタ
QDMA_RAM_SBE_MSK_A (0xF0)	0xF0	ECC マスク レジスタ (シングル ビット エラー)
QDMA_RAM_SBE_STS_A (0xF4)	0xF4	ECC シングル ビット エラー ステータス
QDMA_RAM_DBE_MSK_A (0xF8)	0xF8	ECC マスク レジスタ (ダブル ビット エラー)
QDMA_RAM_DBE_STS_A (0xFC)	0xFC	ECC ダブル ビット エラー ステータス

コンフィギュレーション ブロック識別子 (0x00)

表 68: コンフィギュレーション ブロック識別子 (0x00)

ビット	デフォルト	アクセス タイプ	フィールド	説明
31:20	12'h1fd	RO	Identifier	PCIE 識別子の DMA サブシステム
19:16	4'h3	RO	Config_block_identifier	コンフィギュレーション識別子
15:8	8'h0	RO		予約
7:0	8'h00	RO	Version	バージョン

コンフィギュレーション ブロック BusDev (0x04)

表 69: コンフィギュレーション ブロック BusDev (0x04)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[15:0]	PCIE IP	RO	BDF	bus_dev バス、デバイス、およびファンクション

コンフィギュレーション ブロック PCIE 最大ペイロード サイズ (0x08)

表 70: コンフィギュレーション ブロック PCIE 最大ペイロード サイズ (0x08)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[2:0]	PCIe IP	RO	pcie_max_payload	pcie_max_payload 最大書き込みペイロード サイズ。これは PCIe IP MPS か DMA Subsystem for PCIe のパラメーターの値の少 ない方になります。 3'b000: 128 バイト 3'b001: 256 バイト 3'b010: 512 バイト 3'b011: 1024 バイト 3'b100: 2048 バイト 3'b101: 4096 バイト

コンフィギュレーション ブロック PCIE 最大読み出しリクエスト サイズ (0x0C)

表 71: コンフィギュレーション ブロック PCIE 最大読み出しリクエスト サイズ (0x0C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[2:0]	PCIe IP	RO	pcie_max_read	pcie_max_read 最大読み出しリクエスト サイズ。これは PCIe IP MRRS か DMA Subsystem for PCIe のパラメーターの値の少 ない方になります。 3'b000: 128 バイト 3'b001: 256 バイト 3'b010: 512 バイト 3'b011: 1024 バイト 3'b100: 2048 バイト 3'b101: 4096 バイト

コンフィギュレーション ブロック システム ID (0x10)

表 72: コンフィギュレーション ブロック システム ID (0x10)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[15:0]	16'hff01	RO	system_id	system_id DMA Subsystem for PCIe のシステム ID

コンフィギュレーション ブロック DRU イネーブル (0x14)

表 73: コンフィギュレーション ブロック DRU イネーブル (0x14)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[17]	PCIe IP	RO	MSI_enable3	PF3 用の MSI イネーブル ステータス

表 73: コンフィギュレーション ブロック DRU イネーブル (0x14) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[16]	PCIe IP	RO	MSIX_enable3	PF3 用の MSIX イネーブル ステータス
[13]	PCIe IP	RO	MSI_enable2	PF2 用の MSI イネーブル ステータス
[12]	PCIe IP	RO	MSIX_enable2	PF2 用の MSIX イネーブル ステータス
[9]	PCIe IP	RO	MSI_enable1	PF1 用の MSI イネーブル ステータス
[8]	PCIe IP	RO	MSIX_enable1	PF1 用の MSIX イネーブル ステータス
[1]	PCIe IP	RO	MSI_enable0	PF0 用の MSI イネーブル ステータス
[0]	PCIe IP	RO	MSIX_enable0	PF0 用の MSIX イネーブル ステータス

コンフィギュレーション ブロック PCIE データ幅 (0x18)

表 74: コンフィギュレーション ブロック PCIE データ幅 (0x18)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[2:0]	C_DAT_WID TH	RO	データパス幅	データパス幅 0: 64 ビット 1: 128 ビット 2: 256 ビット 3: 512 ビット

コンフィギュレーション PCIE 制御 (0x1C)

表 75: コンフィギュレーション PCIE 制御 (0x1C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[1]	1'b0	RW	rrq_disable	PCIe への読み出しリクエストをディスエーブルにします。 ブリッジスレーブ、および PCIe への DMA 読み出しはキャンセルされ、完了エラーを戻します。
[0]	1'b1	RW	Relaxed_ordering	緩和型順序付け。 PCIe 読み出しリクエスト TLP が、緩和型順序付けビットセットを使用して生成されます。

コンフィギュレーション AXI ユーザー最大ペイロード サイズ (0x40)

表 76: コンフィギュレーション AXI ユーザー最大ペイロード サイズ (0x40)

ビット	デフォルト	アクセス タイプ	フィールド	説明
6:4	3'h5	RO	user_max_payload_issued	user_eff_payload ユーザー アプリケーションに出力される実際の最大ペイロード サイズ。IP の設定またはデータバス幅により、この値が user_prg_payload 未満になることがあります。 3'b000: 128 バイト 3'b001: 256 バイト 3'b010: 512 バイト 3'b011: 1024 バイト 3'b100: 2048 バイト 3'b101: 4096 バイト
2:0	3'h5	RW	user_max_payload_prog	user_prg_payload DMA のユーザー アプリケーションに出力されるプログラムされた最大ペイロード サイズ。このレジスタは、DMA がアイドル状態ときのみ変更する必要があります。 3'b000: 128 バイト 3'b001: 256 バイト 3'b010: 512 バイト 3'b011: 1024 バイト 3'b100: 2048 バイト 3'b101: 4096 バイト

コンフィギュレーション AXI ユーザー最大読み出しリクエスト サイズ (0x44)

表 77: コンフィギュレーション AXI ユーザー最大読み出しリクエスト サイズ (0x44)

ビット	デフォルト	アクセス タイプ	フィールド	説明
6:4	3'h5	RO	usr_max_read_request_issued	user_eff_read ユーザー アプリケーションに出力される最大読み出しリクエスト サイズ。PCIe のコンフィギュレーションまたはデータバス幅が原因で、この値が user_max_read 未満になる可能性があります。 3'b000: 128 バイト 3'b001: 256 バイト 3'b010: 512 バイト 3'b011: 1024 バイト 3'b100: 2048 バイト 3'b101: 4096 バイト

表 77: コンフィギュレーション AXI ユーザー最大読み出しリクエスト サイズ (0x44) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
2:0	3'h5	RW	usr_max_read_request_prg	user_prg_read DMA のユーザー アプリケーションに出力される最大読み出しリクエスト サイズ。このレジスタは、DMA がアイドル状態ときにのみ変更する必要があります。 3'b000: 128 バイト 3'b001: 256 バイト 3'b010: 512 バイト 3'b011: 1024 バイト 3'b100: 2048 バイト 3'b101: 4096 バイト

コンフィギュレーション ブロック Misc 制御 (0x4C)

表 78: コンフィギュレーション ブロック Misc 制御 (0x4C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[19:8]	NUM_TAGS	RW	num_tag	使用するタグの数を制限します。プログラムされた値が IP で設定されたタグの数以下になるように、ハードウェアでチェックされます。このレジスタは、ブリッジスレーブおよび DMA がアイドル状態ときにのみ変更する必要があります。
[4:0]	データパス幅: 64: 5'h2 128: 5'h3 256: 5'h6 512: 6'h9	RW	rq_metering_multiplier	PCIe コントローラーの完了バッファのオーバーフローを防ぐため、最大未処理読み出しデータを制限します。これは、設定された PCIe コントローラーの完了バッファサイズに合わせて、プログラムする必要があります。このレジスタは、ブリッジスレーブおよび DMA がアイドル状態ときにのみ変更する必要があります。 制限値 = (値 + 1) * 32 * 64 バイト

コンフィギュレーション ブロック スクラッチ 7-0 (0x80-0x9C)

表 79: コンフィギュレーション ブロック スクラッチ (0x80-9C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	scratch	汎用スクラッチ ステータス レジスタ。これらのフィールドは QDMA ハードウェア ファンクションには影響しません。

QDMA_RAM_SBE_MSK_A (0xF0)

表 80: QDMA_RAM_SBE_MSK_A (0xF0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]			mask	エラー ログをイネーブルにするマスクです。定義は、QMD_RAM_SBE_STS を参照してください。

QDMA_RAM_SBE_STS_A (0xF4)

表 81: QDMA_RAM_SBE_STS_A (0xF4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]			h2c_pend_fifo	H2C ST 保留 FIFO RAM シングル ビット ECC エラー。
[30]			pfch_ll_ram	C2H ST プリフェッチ リスト RAM シングル ビット ECC エラー。
[29]			wrb_ctxt_ram	C2H ST 完了コンテキスト RAM シングル ビット ECC エラー。
[28]			pfch_ctxt_ram	C2H ST プリフェッチ RAM シングル ビット ECC エラー。
[27]			desc_req_fifo_ram	C2H ST ディスクリプター リクエスト RAM シングル ビット ECC エラー。
[26]			int_ctxt_ram	割り込みコンテキスト RAM シングル ビット ECC エラー。
[25]			int_qid2vec_ram	割り込み QID2VEC RAM シングル ビット ECC エラー。
[24]			wrb_coal_data_ram	完了連結 RAM シングル ビット ECC エラー。
[23]			tuser_fifo_ram	C2H ST TUSER RAM シングル ビット ECC エラー。
[22]			qid_fifo_ram	C2H STST QID FIFO RAM シングル ビット ECC エラー。
[21]			payload_fifo_ram	C2H ST ペイロード RAM シングル ビット ECC エラー。
[20]			timer_fifo_ram	タイマー FIFO RAM シングル ビット ECC エラー。
[19]			pasid_ctxt_ram	PASID コンフィギュレーション RAM シングル ビット ECC エラー。
[18]			dsc_cpId	ディスクリプター エンジン フェッチ完了データ RAM シングル ビット ECC エラー。
[17]			dsc_cpIi	ディスクリプター エンジン フェッチ完了情報 RAM シングル ビット ECC エラー。
[16]			dsc_sw_ctxt	ディスクリプター エンジン ソフトウェア コンテキスト RAM シングル ビット ECC エラー。
[15]			dsc_crd_rcv	ディスクリプター エンジン受信クレジット コンテキスト RAM シングル ビット ECC エラー。
[14]			dsc_hw_ctxt	ディスクリプター エンジン ハードウェア コンテキスト RAM シングル ビット ECC エラー。
[13]			func_map	ファンクション マップ RAM シングル ビット ECC エラー。
[12]			c2h_wr_brg_dat	ブリッジ スレーブ書き込みデータ バッファ シングル ビット ECC エラー。
[11]			c2h_rd_brg_dat	ブリッジ スレーブ読み出しデータ バッファ シングル ビット ECC エラー。
[10]			h2c_wr_brg_dat	ブリッジ マスター書き込みシングル ビット ECC エラー。
[9]			h2c_rd_brg_dat	ブリッジ マスター読み出しシングル ビット ECC エラー。
[8:5]				予約
[4]			mi_c2h0_dat	C2H MM データ バッファ シングル ビット ECC エラー。
[3:1]				予約

表 81: QDMA_RAM_SBE_STS_A (0xF4) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[0]			mi_h2c0_dat	H2C MM データ バッファ シングル ビット ECC エラー。

QDMA_RAM_DBE_MSK_A (0xF8)

表 82: QDMA_RAM_DBE_MSK_A (0xF8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]			mask	エラー ログをイネーブルにするマスクです。定義は、QMD_RAM_DBE_STS を参照してください。

QDMA_RAM_DBE_STS_A (0xFC)

表 83: QDMA_RAM_DBE_STS_A (0xFC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]			h2c_pend_fifo	H2C 保留 FIFO RAM ダブル ビット ECC エラー
[30]			pfch_ll_ram	C2H ST プリフェッチ リスト RAM ダブル ビット ECC エラー。
[29]			wrb_ctxt_ram	C2H ST 完了コンテキスト RAM ダブル ビット ECC エラー。
[28]			pfch_ctxt_ram	C2H ST プリフェッチ RAM ダブル ビット ECC エラー。
[27]			desc_req_fifo_ram	C2H ST ディスクリプター リクエスト RAM ダブル ビット ECC エラー。
[26]			int_ctxt_ram	割り込みコンテキスト RAM ダブル ビット ECC エラー。
[25]			int_qid2vec_ram	割り込み QID2VEC RAM ダブル ビット ECC エラー。
[24]			wrb_coal_data_ram	完了連結 RAM ダブル ビット ECC エラー。
[23]			tuser_fifo_ram	C2H ST TUSER RAM ダブル ビット ECC エラー。
[22]			qid_fifo_ram	C2H STST QID FIFO RAM ダブル ビット ECC エラー。
[21]			payload_fifo_ram	C2H ST ペイロード RAM ダブル ビット ECC エラー。
[20]			timer_fifo_ram	タイマー FIFO RAM ダブル ビット ECC エラー。
[19]			pasid_ctxt_ram	PASID コンフィギュレーション RAM ダブル ビット ECC エラー。
[18]			dsc_cpld	ディスクリプター エンジン フェッチ完了データ RAM ダブル ビット ECC エラー。
[17]			dsc_cplic	ディスクリプター エンジン フェッチ完了情報 RAM ダブル ビット ECC エラー。
[16]			dsc_sw_ctxt	ディスクリプター エンジンソフトウェア コンテキスト RAM ダブル ビット ECC エラー。
[15]			dsc_crd_rcv	ディスクリプター エンジン受信クレジット コンテキスト RAM ダブル ビット ECC エラー。
[14]			dsc_hw_ctxt	ディスクリプター エンジンハードウェア コンテキスト RAM ダブル ビット ECC エラー。

表 83: QDMA_RAM_DBE_STS_A (0xFC) (続き)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[13]			func_map	ファンクションマップ RAM ダブルビット ECC エラー。
[12]			c2h_wr_brg_dat	ブリッジスレーブ書き込みデータバッファードブルビット ECC エラー。
[11]			c2h_rd_brg_dat	ブリッジスレーブ読み出しデータバッファードブルビット ECC エラー。
[10]			h2c_wr_brg_dat	ブリッジマスター書き込みダブルビット ECC エラー。
[9]			h2c_rd_brg_dat	ブリッジマスター読み出しダブルビット ECC エラー。
[8:5]			予約	
[4]			mi_c2h0_dat	C2H MM データバッファードブルビット ECC エラー。
[3:1]			予約	
[0]			mi_h2c0_dat	H2C MM データバッファードブルビット ECC エラー。

QDMA_TRQ_SEL_GLBL2 (0x00100)

表 84: QDMA_TRQ_SEL_GLBL2 (0x00100) レジスタ空間

レジスタ	アドレス	説明
QDMA_GLBL2_IDENTIFER (0x100)	0x100	識別子 0x1FD3xxxx。
QDMA_GLBL2_PF_BARLITE_INT (0x104)	0x104	内部 DMA レジスタの PF BAR 情報。
QDMA_GLBL2_PF_VF_BARLITE_INT (0x108)	0x108	内部 DMA レジスタの VF BAR 情報。
QDMA_GLBL2_PF_BARLITE_EXT (0x10C)	0x10C	外部 AXI-Lite マスター レジスタの PF BAR 情報。
QDMA_GLBL2_PF_VF_BARLITE_EXT (0x110)	0x110	外部 AXI-Lite マスター レジスタの VF BAR 情報。
QDMA_GLBL2_CHANNEL_INST (0x114)	0x114	DMA チャンネルのインスタンスエーション。
QDMA_GLBL2_CHANNEL_MDMA (0x118)	0x118	DMA チャンネルの QDMA モード。
QDMA_GLBL2_CHANNEL_STRM (0x11C)	0x11C	DMA チャンネルのストリーム モード。
QDMA_GLBL2_CHANNEL_QDMA_CAP (0x120)	0x120	QDMA コンフィギュレーション設定。
QDMA_GLBL2_CHANNEL_PASID_CAP (0x128)	0x128	Pasid 機能。
QDMA_GLBL2_CHANNEL_FUNC_RET (0x12C)	0x12C	ファンクションの戻し。
QDMA_GLBL2_SYSTEM_ID (0x130)	0x130	システム ID。
QDMA_GLBL2_MISC_CAP (0x134)	0x134	その他の機能。
QDMA_GLBL2_DBG_PCIE_RQ0 (0x1B8)	0x1B8	RQ インターフェイス デバッグ情報。
QDMA_GLBL2_DBG_PCIE_RQ1 (0x1BC)	0x1BC	RQ インターフェイス デバッグ情報。
QDMA_GLBL2_DBG_AXIMM_WR0 (0x1C0)	0x1C0	DMA AXIMM インターフェイス デバッグ情報。
QDMA_GLBL2_DBG_AXIMM_WR1 (0x1C4)	0x1C4	DMA AXIMM インターフェイス デバッグ情報。
QDMA_GLBL2_DBG_AXIMM_RD0 (0x1C8)	0x1C8	DMA AXIMM インターフェイス デバッグ情報。
QDMA_GLBL2_DBG_AXIMM_RD1 (0x1CC)	0x1CC	DMA AXIMM インターフェイス デバッグ情報。

QDMA_GLBL2_IDENTIFER (0x100)

表 85: QDMA_GLBL2_IDENTIFIER (0x100)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:8]	24'h1fd700	RO	identifier	識別子
[7:0]	8'h0	RO	version	バージョン

QDMA_GLBL2_PF_BARLITE_INT (0x104)

表 86: QDMA_GLBL2_PF_BARLITE_INT (0x104)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[23:18]		RO	pf3_bar_map[5:0]	Pf3_bar_map は 6 ビットで構成され、各バーに 1 ビット あります。バーのビット位置に 1 がある場合は、このバ ーをヒットするリクエストが DMA レジスタに送信され ることを示します。 対応するビットは、このレジスタおよび QDMA_GLBL2_PF_BARLITE_EXT レジスタの両方で設定 しないでください。どちらのレジスタもリクエストを再 送信しない場合、リクエストはブリッジ AXI-MM マスタ ーインターフェイスに送信されます。
[17:12]		RO	pf2_bar_map[5:0]	pf3_bar_map の説明を参照してください。
[11:6]		RO	pf1_bar_map[5:0]	pf3_bar_map の説明を参照してください。
[5:0]		RO	pf0_bar_map[5:0]	pf3_bar_map の説明を参照してください。

QDMA_GLBL2_PF_VF_BARLITE_INT (0x108)

表 87: QDMA_GLBL2_PF_VF_BARLITE_INT (0x108)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[23:18]		RO	pf3_vf_bar_map[5:0]	pf3_vf_bar_map は 6 ビットで構成され、PF3 の各 VF バ ーに 1 ビットあります。バーのビット位置に 1 がある 場合は、このバーをヒットするリクエストが DMA レジ スタに送信されることを示します。 対応するビットは、このレジスタおよび QDMA_GLBL2_PF_BARLITE_EXT レジスタの両方で設定 しないでください。どちらのレジスタもリクエストを再 送信しない場合、リクエストはブリッジ AXI-MM マスタ ーインターフェイスに送信されます。
[17:12]		RO	pf2_vf_bar_map[5:0]	pf3_bar_map の説明を参照してください。
[11:6]		RO	pf1_vf_bar_map[5:0]	pf3_bar_map の説明を参照してください。
[5:0]		RO	pf0_vf_bar_map[5:0]	pf3_bar_map の説明を参照してください。

QDMA_GLBL2_PF_BARLITE_EXT (0x10C)

表 88: QDMA_GLBL2_PF_BARLITE_EXT (0x10C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[23:18]		RO	pf3_bar_map[5:0]	pf3_bar_map は 6 ビットで構成され、各バーに 1 ビットあります。バーのビット位置に 1 がある場合は、このバーをヒットするリクエストがブリッジ AXI-Lite マスター インターフェイスに送信されることを示します。 対応するビットは、このレジスタおよび QDMA_GLBL2_PF_BARLITE_INT レジスタの両方で設定しないでください。どちらのレジスタもリクエストを再送信しない場合、リクエストはブリッジ AXI-MM マスター インターフェイスに送信されます。
[17:12]		RO	pf2_bar_map[5:0]	pf3_bar_map の説明を参照してください。
[11:6]		RO	pf1_bar_map[5:0]	pf3_bar_map の説明を参照してください。
[5:0]		RO	pf0_bar_map[5:0]	pf3_bar_map の説明を参照してください。

QDMA_GLBL2_PF_VF_BARLITE_EXT (0x110)

表 89: QDMA_GLBL2_PF_VF_BARLITE_EXT (0x110)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[23:18]		RO	pf3_vf_bar_map[5:0]	pf3_vf_bar_map は 6 ビットで構成され、PF3 の各 VF バーに 1 ビットあります。バーのビット位置に 1 がある場合は、このバーをヒットするリクエストがブリッジ AXI-Lite マスター インターフェイスに送信されることを示します。 対応するビットは、このレジスタおよび QDMA_GLBL2_PF_BARLITE_INT レジスタの両方で設定しないでください。どちらのレジスタもリクエストを再送信しない場合、リクエストはブリッジ AXI-MM マスター インターフェイスに送信されます。
[17:12]		RO	pf2_vf_bar_map[5:0]	pf3_vf_bar_map の説明を参照してください。
[11:6]		RO	pf1_vf_bar_map[5:0]	pf3_vf_bar_map の説明を参照してください。
[5:0]		RO	pf0_vf_bar_map[5:0]	pf3_vf_bar_map の説明を参照してください。

QDMA_GLBL2_CHANNEL_INST (0x114)

表 90: QDMA_GLBL2_CHANNEL_INST (0x114)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]				予約
[17]		RO	c2h_st	1 の場合は、C2H ST エンジンがインスタンス化されたことを示します。
[16]		RO	h2c_st	1 の場合は、H2C ST エンジンがインスタンス化されたことを示します。
[15:9]				予約
[8]		RO	c2h_eng[0]	1 の場合は、C2H MM エンジンがインスタンス化されたことを示します。

表 90: QDMA_GLBL2_CHANNEL_INST (0x114) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[7:1]				予約
[0]		RO	h2c_eng[0]	1 の場合は、H2C MM エンジンがインスタンス化されたことを示します。

QDMA_GLBL2_CHANNEL_MDMA (0x118)

表 91: QDMA_GLBL2_CHANNEL_MDMA (0x118)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]				予約
[17]		RO	c2h_st	1 の場合は、C2H ST エンジンが QDMA であることを示します。
[16]		RO	h2c_st	1 の場合は、H2C ST エンジンが QDMA であることを示します。
[15:9]				予約
[8]		RO	c2h_eng[0]	1 の場合は、C2H ST エンジンが QDMA であることを示します。
[7:1]				予約
[0]		RO	h2c_eng[0]	1 の場合は、H2C ST エンジンが QDMA であることを示します。

QDMA_GLBL2_CHANNEL_STRM (0x11C)

表 92: QDMA_GLBL2_CHANNEL_STRM (0x11C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]				予約
[17]		RO	c2h_st	1 の場合は、ストリーム モードの DMA であることを示します。
[16]		RO	h2c_st	1 の場合は、ストリーム モードの DMA であることを示します。
[15:9]				予約
[8]		RO	c2h_eng[0]	1 の場合は、メモリ マップド モードの DMA であることを示します。
[7:1]				予約
[0]		RO	h2c_eng[0]	1 の場合は、メモリ マップド モードの DMA であることを示します。

QDMA_GLBL2_CHANNEL_QDMA_CAP (0x120)

表 93: QDMA_GLBL2_CHANNEL_QDMA_CAP (0x120)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:12]				予約
[11:0]		RO	multq_max	サポートされるキューの数から 1 を引いた値です。

QDMA_GLBL2_CHANNEL_PASID_CAP (0x128)

表 94: QDMA_GLBL2_CHANNEL_PASID_CAP (0x128)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]				予約
[15:4]		RO	bridge_pasid_offset[11:0]	ブリッジ スレーブ リクエストの PASID テーブル オフセット。使用される PASID テーブル エントリは、PASID テーブル オフセットヘリクエストのファンクション数を追加すると決定されます。
[3:2]				予約
[1]		RO	bridge_pasid_en	1 の場合は、ブリッジ スレーブ リクエストが PASID 対応であることを示します。
[0]		RO	dma_pasid_en	1 の場合は、DMA リクエストが PASID 対応であることを示します。

QDMA_GLBL2_CHANNEL_FUNC_RET (0x12C)

表 95: QDMA_GLBL2_CHANNEL_FUNC_RET (0x12C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:8]				予約
[7:0]		RO	function[7:0]	レジスタ読み出しのコンプリーター ファンクションの数を返します。

QDMA_GLBL2_SYSTEM_ID (0x130)

表 96: QDMA_GLBL2_SYSTEM_ID (0x130)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]				予約
[15:0]		RO	system_id[15:0]	System_id 属性/パラメーターを返します。

QDMA_GLBL2_MISC_CAP (0x134)

表 97: QDMA_GLBL2_MISC_CAP (0x134)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]		RO		予約 属性で伝えられるその他の機能。

QDMA_GLBL2_DBG_PCIE_RQ0 (0x1B8)

表 98: QDMA_GLBL2_DBG_PCIE_RQ0 (0x1B8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:20]		RO	nph_avl[11:0]	使用可能な NPH クレジット
[19:10]		RO	rcb_avl[9:0]	使用可能な RCB クレジット (32B 精度)。
[9:4]		RO	slv_rd_credits[5:0]	ブリッジ スレーブ読み出し順序付けクレジット
[3:2]		RO	tag_ep[1:0]	タグ プール エンプティ ステータス
[1:0]		RO	tag_fl[1:0]	タグ プール フル ステータス

QDMA_GLBL2_DBG_PCIE_RQ1 (0x1BC)

表 99: QDMA_GLBL2_DBG_PCIE_RQ1 (0x1BC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:17]				予約
[16]		RO	wtlp_req	Wtlp リクエスト
[15]		RO	wtlp_header_fifo_fl	Wtlp ヘッダー FIFO フル
[14]		RO	wtlp_header_fifo_ep	Wtlp ヘッダー FIFO 空
[13]		RO	rq_fifo_ep	RQ FIFO 空
[12]		RO	rq_fifo_fl	RQ FIFO フル
[11:9]		RO	tlpsm[2:0]	tlp ステート
[8:6]		RO	tlpsm512[2:0]	tlp512 ステート
[5]		RO	rreq0_rcb_ok	読み出しリクエスト スロット 0 に十分な RCB があります。
[4]		RO	rreq0_slv	読み出しリクエスト スロット 0 はスレーブ リクエストです。
[3]		RO	rreq0_vld	読み出しリクエスト スロット 0 は保留です。
[2]		RO	rreq1_rcb_ok	読み出しリクエスト スロット 1 に十分な RCB があります。
[1]		RO	rreq1_slv	読み出しリクエスト スロット 1 はスレーブ リクエストです。
[0]		RO	rreq1_vld	読み出しリクエスト スロット 1 は保留です。

QDMA_GLBL2_DBG_AXIMM_WR0 (0x1C0)

表 100: QDMA_GLBL2_DBG_AXIMM_WR0 (0x1C0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:27]				予約
[26]		RO	wr_req	wr_req
[25:23]		RO	wr_chn[2:0]	wr_chn
[22]		RO	wtlp_dat_fifo_ep	wtlp_dat_fifo_ep
[21]		RO	wpl_fifo_ep	wpl_fifo_ep
[20:18]		RO	brsp_claim_chnl[2:0]	brsp_claim_chnl
[17:12]		RO	wrreq_cnt[5:0]	wrreq_cnt
[11:9]		RO	bid[2:0]	bid
[8]		RO	bvalid	bvalid
[7]		RO	bready	bready
[6]		RO	wvalid	wvalid
[5]		RO	wready	wready
[4:2]		RO	awid[2:0]	awid
[1]		RO	awvalid	awvalid
[0]		RO	awready	awready

QDMA_GLBL2_DBG_AXIMM_WR1 (0x1C4)

表 101: QDMA_GLBL2_DBG_AXIMM_WR1 (0x1C4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:30]				予約
[29:24]		RO	brsp_cnt4[5:0]	brspcnt4
[23:18]		RO	brsp_cnt3[5:0]	brspcnt3
[17:12]		RO	brsp_cnt2[5:0]	brspcnt2
[11:6]		RO	brsp_cnt1[5:0]	brspcnt1
[5:0]		RO	brsp_cnt0[5:0]	brspcnt0

QDMA_GLBL2_DBG_AXIMM_RD0 (0x1C8)

表 102: QDMA_GLBL2_DBG_AXIMM_RD0 (0x1C8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:23]				予約
[22:17]		RO	pnd_cnt[5:0]	pnd_cnt
[16:14]		RO	rd_chnl[2:0]	rd_chnl
[13]		RO	rd_req	rd_req

表 102: QDMA_GLBL2_DBG_AXIMM_RD0 (0x1C8) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[12:10]		RO	rrsp_claim_chnl[2:0]	rrsp_claim_chnl
[9:7]		RO	rid[2:0]	rid
[6]		RO	rvalid	rvalid
[5]		RO	rready	rready
[4:2]		RO	arid[2:0]	arid
[1]		RO	arvalid	arvalid
[0]		RO	arready	arready

QDMA_GLBL2_DBG_AXIMM_RD1 (0x1CC)

表 103: QDMA_GLBL2_DBG_AXIMM_RD1 (0x1CC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:30]				予約
[29:24]		RO	rrsp_cnt4[5:0]	rrspcnt4
[23:18]		RO	rrsp_cnt3[5:0]	rrspcnt3
[17:12]		RO	rrsp_cnt2[5:0]	rrspcnt2
[11:6]		RO	rrsp_cnt1[5:0]	rrspcnt1
[5:0]		RO	rrsp_cnt0[5:0]	rrspcnt0

QDMA_TRQ_SEL_GLBL (0x00200)

表 104: QDMA_TRQ_SEL_GLBL (0x00200) レジスタ空間

レジスタ (アドレス)	アドレス	説明
QDMA_GLBL_RNG_SZ (0x204-0x240)	0x204-0x240	グローバル リング サイズ レジスタ。 16 通りのリング サイズを設定できます。
QDMA_GLBL_ERR_STAT (0x248)	0x248	グローバル エラー ステータス
QDMA_GLBL_ERR_MASK (0x24C)	0x24C	グローバル エラー マスク イネーブル
QDMA_GLBL_DSC_CFG (0x250)	0x250	ディスクリプター設定および C2H 完了累積
QDMA_GLBL_DSC_ERR_STS (0x254)	0x254	ディスクリプター エラー ステータス ビット
QDMA_GLBL_DSC_ERR_MSK (0x258)	0x258	ディスクリプター エラー マスク イネーブル
QDMA_GLBL_DSC_ERR_LOG0 (0x25C)	0x25C	ディスクリプター エラー情報
QDMA_GLBL_DSC_ERR_LOG1 (0x260)	0x260	エラーのディスクリプター タイプ
QDMA_GLBL_TRQ_ERR_STS (0x264)	0x264	アドレス ターゲット エラー ステータス
QDMA_GLBL_TRQ_ERR_MSK (0x268)	0x268	アドレス ターゲット エラー マスク イネーブル
QDMA_GLBL_TRQ_ERR_LOG (0x26C)	0x26C	アドレス ターゲット エラー情報
QDMA_GLBL_DSC_DBG_DAT0 (0x270)	0x270	ディスクリプター エンジン デバッグ情報
QDMA_GLBL_DSC_DBG_DAT1 (0x274)	0x274	ディスクリプター エンジン デバッグ情報

表 104: QDMA_TRQ_SEL_GLBL (0x00200) レジスタ空間 (続き)

レジスタ (アドレス)	アドレス	説明
QDMA_GLBL_DSC_ERR_LOG2 (0x27C)	0x27C	ディスクリプター エラー情報
QDMA_GLBL_INTERRUPT_CFG (0x2C4)	0x2C4	割り込み設定

QDMA_GLBL_RNG_SZ (0x204-0x240)

表 105: QDMA_GLBL_RNG_SZ (0x204-0x240)

ビット	デフォルト	アクセス タイプ	フィールド	説明
31:16	16'h0	NA		予約
15:00	NA	RW	Ring_size	リング サイズ (ライト バック ステータスの位置を含む)

グローバル リング サイズは 16 個のレジスタから成る 1 つのグループで、ディスクリプターおよび完了コンテキストでは、グローバル リング サイズ フィールドを使用してサイズが選択されます。

アドレス = $0x200 + ((index + 1) * 4)$

インデックス = 0 の場合、リング サイズ レジスタ 0 はアドレス 0x204 にあります。

インデックス = 1 の場合、リング サイズ レジスタ 1 はアドレス 0x208 にあります。

16 個のリング サイズ レジスタはすべて、キューに使用される前に、書き込みで明示的にアップデートしておく必要があります。これらのレジスタにはリセット値はありません。

QDMA_GLBL_SCRATCH (0x244)

表 106: QDMA_GLBL_SCRATCH (0x244)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	scratch[31:0]	任意ファンクションのスクラッチ空間。各ファンクション (PF) にはそれぞれにスクラッチ空間があります。

QDMA_GLBL_ERR_STAT (0x248)

表 107: QDMA_GLBL_ERR_STAT (0x248)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:12]	0	NA		予約
11	0	RW1C	err_h2c_st	H2C-ST でエラーが発生したことを示します。
10	0	RW1C	err_bdg	ブリッジでエラーが発生したことを示します。
9	0	RW1C	ind_ctxt_cmd_err	
8	0	RW1C	err_c2h_st	C2H-ST でエラーが発生したことを示します。
7	0	RW1C	err_c2h_mm_1	C2H-MM チャネル 1 でエラーが発生したことを示します。
6	0	RW1C	err_c2h_mm_0	C2H-MM チャネル 0 でエラーが発生したことを示します。

表 107: QDMA_GLBL_ERR_STAT (0X248) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
5	0	RW1C	err_h2c_mm_1	H2C-MM チャンネル 1 でエラーが発生したことを示します。
4	0	RW1C	err_h2c_mm_0	H2C-MM チャンネル 0 でエラーが発生したことを示します。
3	0	RW1C	err_trq	
2	0	RW1C	err_dsc	
1	0	RW1C	err_ram_dbe	
0	0	RW1C	err_ram_sbe	

QDMA_GLBL_ERR_MASK (0X24C)

表 108: QDMA_GLBL_ERR_MASK (0X24C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:9]	0			予約
[8:0]	0	RW	mask	出力エラー イネーブル マスク。定義は、QDMA_GLBL_ERR_STAT_A を参照してください。

QDMA_GLBL_DSC_CFG (0x250)

表 109: QDMA_GLBL_DSC_CFG (0x250)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:10]	0	NA		予約
[9]	0	RW	unc_ovr_cor	訂正できないログを訂正できるように上書きします。
[8]	0	RW	ctxt_fer_dis	コンテキストでディスクリプターおよび DMA エラー ビットの両方を記録します (どちらが一方だけではなく)。
[7:6]	0	NA		予約
[5:3]	6	RW	max_dsc_fetch	1 つのリクエストでフェッチできるディスクリプターの最大数。8 * 2 ^{val} //最大値は 6 です。
[2:0]	0	RW	wb_int	非バイパス モードで実行している MM または H2C ストリーム キューに対し、完了が生成される間隔です。 3'h0: 4 3'h1: 8 3'h2: 16 3'h3: 32 3'h4: 64 3'h5: 128 3'h6: 256 3'h7: 512

完了の間隔はキュー コンテキスト設定を使用してディスエーブルにできます。ディスエーブルになっている場合、最近の PIDX を持つディスクリプターが完了すると、完了が生成されます。

QDMA_GLBL_DSC_ERR_STS (0x254)

表 110: QDMA_GLBL_DSC_ERR_STS (0x254)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:25]	0			予約
[24]	0	RW1C	sbe	COR_ERR_RAM_SBE
[23]	0	RW1C	dbe	UNC_ERR_RAM_DBE
[22]	0	RW1C	rq_cancel	レジスタ ステータスをディスエーブルにするため、DMA でディスクリプターのフェッチがキャンセルされます。
[21]	0	RW1C	dsc	無効な PIDX アップデート。
[20]	0	RW2C	dma	UNC_ERR_DMA。DMA エンジンにエラーが発生しています。
[19]	0	RW1C	flr_cancel	FLR が原因で、DMA でディスクリプターのフェッチがキャンセルされます。
[18:17]	0			予約
[16]	0	RW1C	dat_poison	ディスクリプターのフェッチ完了にポイズン データが含まれています。
[9]	0	RW1C	タイムアウト	ディスクリプターのフェッチ完了がタイムアウトしています。
[5]	0	RW1C	flr	ディスクリプターのフェッチ完了に FLR エラーが起きています。
[4]	0	RW1C	tag	ディスクリプターのフェッチ完了に予期しないタグが含まれています。
[3]	0	RW1C	addr	ディスクリプターのフェッチ完了にアドレス不一致があります。
[2]	0	RW1C	param	ディスクリプターのフェッチ完了にパラメーター不一致があります。
[1]	0	RW1C	ur_ca	ディスクリプターのフェッチ完了にサポートされないリクエスト、またはコンプリーター中止ステータスが含まれています。
[0]	0	RW1C	poison	ディスクリプターのフェッチ完了にヘッダー ポイズン データが含まれています。

QDMA_GLBL_DSC_ERR_MSK (0x258)

表 111: QDMA_GLBL_DSC_ERR_MSK (0x258)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	mask	エラー ログをイネーブルにするマスクです。 QDMA_GLBL_DSC_ERR_STS_A を参照してください。

QDMA_GLBL_DSC_ERR_LOG0 (0x25C)

表 112: QDMA_GLBL_DSC_ERR_LOG0 (0x25C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]	0	RW	valid	エラー ログは有効です。
[30]	0		sel	エラーの DMA 方向 0: H2C 1: C2H
[29:11]	0	NA		予約
[10:0]	0	RW	qid	エラーのキュー ID

QDMA_GLBL_DSC_ERR_LOG1 (0x260)

表 113: QDMA_GLBL_DSC_ERR_LOG1 (0x260)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:28]	0	RW		予約
[27:12]	0	RW	cidx	エラーのコンシューマー インデックス
[11:9]	0	NA		予約
[8:5]	0	RW	sub_type	エラーのサブタイプ。update_err 専用。 0: update_err 以外 2: PIDX アップデート オーバーフロー。リング サイズ と比べてポストされているディスクリプターの数が多い ります。
[4:0]	0	RW	err_type	エラー タイプ。QDMA_GLBL_DSC_ERR_LOG0 有効がセ ットされている場合は、どのマスクされていないエラー が最初に発生したか、また、ログに記録されているステ ータス レジスタのエラー タイプを示します。

QDMA_GLBL_TRQ_ERR_STS (0x264)

表 114: QDMA_GLBL_TRQ_ERR_STS (0x264)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:4]	0	NA		予約
[3]	0	RW1C	tcp_timeout	DMA 内部レジスタへのリクエストに対するタイムアウ ト。
[2]	0	RW1C	vf_access_err	VF がグローバル レジスタ空間またはファンクション マ ップにアクセスを試行。
[1]	0	RW1C	qid_range	ファンクションが、ファンクション マップの RAM で割 り当てられているキューを超えて、QID にアクセスを試 行。
[0]	0	RW1C	unmapped	マップされていないレジスタ空間へのアクセス。

QDMA_GLBL_TRQ_ERR_MSK (0x268)

表 115: QDMA_GLBL_TRQ_ERR_MSK (0x268)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	NA	mask	エラー ログをイネーブルにするマスク。定義は、QDMA_GLBL_TRQ_ERR_STS を参照してください。

QDMA_GLBL_TRQ_ERR_LOG (0x26C)

表 116: QDMA_GLBL_TRQ_ERR_LOG (0x26C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:28]	0	RW	target	レジスタ アクセスのターゲット。 0: 予約 1: QDMA_TRQ_SEL_GLBL1 2: QDMA_TRQ_SEL_GLBL2 3: QDMA_TRQ_SEL_GLBL 4: QDMA_TRQ_SEL_FMAP 5: QDMA_TRQ_SEL_IND 6: QDMA_TRQ_SEL_C2H 7: 予約 8: 予約 9: QDMA_TRQ_SEL_C2H_MM0 10: 予約 11: QDMA_TRQ_SEL_H2C_MM0 12: 予約 13: QDMA_TRQ_SEL_QUEUE_PF
[27:24]		NA		予約
[23:16]	0	RW	function	レジスタ アクセス空間ファンクション
[15:0]	0	RW	address	レジスタ アクセス空間アドレス

QDMA_GLBL_DSC_DBG_DAT0 (0x270)

表 117: QDMA_GLBL_DSC_DBG_DAT0 (0x270)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:30]	0	NA		予約
[29]	0	RO	ctxt_arb_dir	ディスクリプター キュー コンテキストのアービトレーション リクエストの DMA 方向。どのアービター ソースを読み出すかを選択するのに QDMA_DSC_DBG_CTL を使用します。
[28:17]	0	RO	ctxt_arb_qid[11:0]	ディスクリプター キュー コンテキストのアービトレーション リクエストの QID。どのアービター ソースを読み出すかを選択するのに QDMA_DSC_DBG_CTL を使用します。

表 117: QDMA_GLBL_DSC_DBG_DAT0 (0x270) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[16:12]	0	RO	ctxt_arb_req[4:0]	ctxt アービトレーション リクエストのベクター。ビット位置マップ: EVT_SRC=0、TRQ_SRC=1、WBC_SRC=2、CRD_SRC=3、IND_SRC=4
[11]	0	RO	irq_fifo_fl	直近の LRQ FIFO がフル
[10]	0	RO	tm_dsc_stall	Tm_dsc_sts 出力がバックプレッシャーになっています。
[9:8]	0	RO	rrq_stall[1:0]	ビット 1: C2H 読み出しリクエスト ストール ビット 0: H2C 読み出しリクエスト ストール
[7:6]	0	RO	rcp_fifo_spc_stall[1:0]	ビット 1: C2H 読み出し完了空間ストール ビット 0: H2C 読み出し完了空間ストール
[5:4]	0	RO	rrq_fifo_spc_stall[1:0]	ビット 1: C2H 読み出しリクエスト FIFO 空間ストール ビット 0: H2C 読み出しリクエスト FIFO 空間ストール
[3:2]	0	RO	fab_mrkr_rsp_stall[1:0]	ビット 1: C2H mrkr_rsp ストール ビット 0: H2C mrkr_rsp ストール
[1:0]	0	RO	dsc_out_stall[1:0]	ビット 1: C2H ディスクリプター バイパス出力ストール (vld && ~rdy) ビット 0: H2C ディスクリプター バイパス出力ストール (vld && ~rdy)

QDMA_GLBL_DSC_DBG_DAT1 (0x274)

表 118: QDMA_GLBL_DSC_DBG_DAT1 (0x274)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:28]	0	NA		予約
[27:22]	0	RO	evt_spc_c2h[5:0]	C2H のイベント空間。
[21:16]	0	RO	evt_spc_h2c[5:0]	H2C のイベント空間。
[15:8]	0x80	RO	dsc_spc_c2h[7:0]	C2H のディスクリプター フェッチ完了の RAM 空間。
[7:0]	x80	RO	dsc_spc_h2c[7:0]	H2C のディスクリプター フェッチ完了の RAM 空間。

QDMA_GLBL_DSC_ERR_LOG2 (0x27C)

表 119: QDMA_GLBL_DSC_ERR_LOG2 (0x27C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	RO	pidx_old[1:0]	エラー条件が発生する前に格納される古い PIDX。
[15:0]	0	RO	pidx_new[15:0]	新しいアップデートされた PIDX。

QDMA_GLBL_INTERRUPT_CFG (0x2C4)

表 120: QDMA_GLBL_INTERRUPT_CFG (0x2C4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:2]	0	RW		予約
[1:1]	0	RW1C	lgcy_intr_pending	レガシ割り込みの保留。このビットは、保留レガシ割り込み出力のハードウェアによって設定され、ソフトウェアがその割り込みを受信すると、クリアになります。このビットに 1 を書き込むと、ソフトウェアがクリアになります。
[0:0]	0	RW	en_lgcy_intr	レガシ割り込みをイネーブルにします。

QDMA_TRQ_SEL_FMAP (0x00400)

表 121: QDMA_TRQ_SEL_FMAP (0x00400) レジスタ空間

レジスタ (アドレス)	アドレス	説明
QDMA_TRQ_SEL_FMAP (0x400-0x7FC)	0x400 - 0x7FC	ファンクション マップ

QDMA_TRQ_SEL_FMAP (0x400-0x7FC)

ファンクション マップは、キューが連続しているブロックをファンクションにマップするために使用されます。これは、どの PF からでも可能です。

表 122: QDMA_TRQ_SEL_FMAP (0x400-0x7FC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:23]	0	NA		予約
[22:11]	0	RW	qid_max	このファンクションが持つことのできる最大キュー数です。
[10:0]	0	RW	qid_base	ファンクションの基本的な物理的キューの ID です。

各ファンクションのレジスタ アドレスは 0x400+ (ファンクション番号 *4) と計算されます。

- ファンクション番号 0 がアドレス 0x400 に書き込まれます。
- ファンクション番号 1 がアドレス 0x404 に書き込まれます。
- 最終ファンクション番号がアドレス 0x7FC に書き込まれます。

QDMA_TRQ_SEL_IND (0x00800)

表 123: QDMA_TRQ_SEL_IND (0x00800) レジスタ空間

レジスタ (アドレス)	アドレス	説明
QDMA_IND_CTXT_DATA_0 (0x804)	0x804	コンテキスト データ (個々のコンテキスト構造を参照)

表 123: QDMA_TRQ_SEL_IND (0x00800) レジスタ空間 (続き)

レジスタ (アドレス)	アドレス	説明
QDMA_IND_CTXT_DATA_1 (0x808)	0x808	コンテキスト データ (個々のコンテキスト構造を参照)
QDMA_IND_CTXT_DATA_1 (0x808)	0x80C	コンテキスト データ (個々のコンテキスト構造を参照)
QDMA_IND_CTXT_DATA_3 (0x810)	0x810	コンテキスト データ (個々のコンテキスト構造を参照)
QDMA_IND_CTXT_DATA_4 (0x814)	0x814	コンテキスト データ (個々のコンテキスト構造を参照)
QDMA_IND_CTXT_DATA_5 (0x818)	0x818	コンテキスト データ (個々のコンテキスト構造を参照)
QDMA_IND_CTXT_DATA_6 (0x81C)	0x81C	コンテキスト データ (個々のコンテキスト構造を参照)
QDMA_IND_CTXT_DATA_7 (0x820)	0x820	コンテキスト データ (個々のコンテキスト構造を参照)
QDMA_IND_CTXT_MASK_0 (0x824)	0x824	ライト イネーブル マスク
QDMA_IND_CTXT_MASK_1 (0x828)	0x828	ライト イネーブル マスク
QDMA_IND_CTXT_MASK_2 (0x82C)	0x82C	ライト イネーブル マスク
QDMA_IND_CTXT_MASK_3 (0x830)	0x830	ライト イネーブル マスク
QDMA_IND_CTXT_MASK_4 (0x834)	0x834	ライト イネーブル マスク
QDMA_IND_CTXT_MASK_5 (0x838)	0x838	ライト イネーブル マスク
QDMA_IND_CTXT_MASK_6 (0x83C)	0x83C	ライト イネーブル マスク
QDMA_IND_CTXT_MASK_7 (0x840)	0x840	ライト イネーブル マスク
QDMA_IND_CTXT_CMD (0x844)	0x844	コンテキスト コマンド

QDMA_IND_CTXT_DATA_0 (0x804)

表 124: QDMA_IND_CTXT_DATA_0 (0x804)

ビット	デフォルト	アクセスタ イプ	フィールド	説明
[31:0]	0	RW	data	コンテキスト データ [31:0]

8 個のレジスタすべて (0x804、0x808、0x80C、0x810、0x814、0x818、0x81C、および 0x820) が、キューのコンテキスト データから構成されています。

QDMA_IND_CTXT_DATA_1 (0x808)

表 125: QDMA_IND_CTXT_DATA_1 (0x808)

ビット	デフォルト	アクセスタ イプ	フィールド	説明
[31:0]	0	RW	data	コンテキスト データ [63:32]

QDMA_IND_CTXT_DATA_2 (0x80C)

表 126: QDMA_IND_CTXT_DATA_2 (0x80C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	data	コンテキスト データ [95:64]

QDMA_IND_CTXT_DATA_3 (0x810)

表 127: QDMA_IND_CTXT_DATA_3 (0x810)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	data	コンテキスト データ [127:96]

QDMA_IND_CTXT_DATA_4 (0x814)

表 128: QDMA_IND_CTXT_DATA_4 (0x814)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	data	コンテキスト データ [159:128]

QDMA_IND_CTXT_DATA_5 (0x818)

表 129: QDMA_IND_CTXT_DATA_5 (0x818)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	data	コンテキスト データ [191:160]

QDMA_IND_CTXT_DATA_6 (0x81C)

表 130: QDMA_IND_CTXT_DATA_6 (0x81C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	data	コンテキスト データ [223:192]

QDMA_IND_CTXT_DATA_7 (0x820)

表 131: QDMA_IND_CTXT_DATA_7 (0x820)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	data	コンテキスト データ [255:224]

QDMA_IND_CTXT_MASK_0 (0x824)

このマスクを設定して、対応するデータビットを書き込みます。データ マスキングはソフトウェア ディスクリプターのコンテキストでのみサポートされます。

表 132: QDMA_IND_CTXT_MASK_0 (0x824)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:0]	0	RW	mask	コンテキスト マスク [31:0]

8 個のレジスタすべて (0x824、0x828、0x82C、0x830、0x834、0x838、0x83C、0x840) が、キューのコンテキスト マスクから構成されています。

QDMA_IND_CTXT_MASK_1 (0x828)

このマスクを設定して、対応するデータビットを書き込みます。データ マスキングはソフトウェア ディスクリプターのコンテキストでのみサポートされます。

表 133: QDMA_IND_CTXT_MASK_1 (0x828)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:0]	0	RW	mask	コンテキスト マスク [63:32]

QDMA_IND_CTXT_MASK_2 (0x82C)

このマスクを設定して、対応するデータビットを書き込みます。データ マスキングはソフトウェア ディスクリプターのコンテキストでのみサポートされます。

表 134: QDMA_IND_CTXT_MASK_2 (0x82C)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:0]	0	RW	mask	コンテキスト マスク [95:64]

QDMA_IND_CTXT_MASK_3 (0x830)

このマスクを設定して、対応するデータビットを書き込みます。データ マスキングはソフトウェア ディスクリプターのコンテキストでのみサポートされます。

表 135: QDMA_IND_CTXT_MASK_3 (0x830)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:0]	0	RW	mask	コンテキスト マスク [127:96]

QDMA_IND_CTXT_MASK_4 (0x834)

表 136: QDMA_IND_CTXT_MASK_4 (0x834)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	mask	コンテキスト マスク [159:128]

QDMA_IND_CTXT_MASK_5 (0x838)

表 137: QDMA_IND_CTXT_MASK_5 (0x838)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	mask	コンテキスト マスク [191:160]

QDMA_IND_CTXT_MASK_6 (0x83C)

表 138: QDMA_IND_CTXT_MASK_6 (0x83C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	mask	コンテキスト マスク [223:192]

QDMA_IND_CTXT_MASK_7 (0x840)

表 139: QDMA_IND_CTXT_MASK_7 (0x840)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	mask	コンテキスト マスク [255:224]

QDMA_IND_CTXT_CMD (0x844)

表 140: QDMA_IND_CTXT_CMD (0x844)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:7]	0	RW	qid	コンテキストのキュー ID QDMA_CTXT_SELCT_FMAP の場合、このフィールドはアクセスするファンクションを示します。
[6:5]	0	RW	op	Opcode 2'h0 = QDMA_CTXT_CMD_CLR 2'h1 = QDMA_CTXT_CMD_WR 2'h2 = QDMA_CTXT_CMD_RD 2'h3 = QDMA_CTXT_CMD_INV

表 140: QDMA_IND_CTXT_CMD (0x844) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[4:1]	0	RW	sel	4'h0 = QDMA_CTXT_SEL_DEC_SW_C2H 4'h1 = QDMA_CTXT_SEL_DEC_SW_H2C 4'h2 = QDMA_CTXT_SEL_DEC_HW_C2H 4'h3 = QDMA_CTXT_SEL_DEC_HW_H2C 4'h4 = QDMA_CTXT_SEL_DEC_CR_C2H 4'h5 = QDMA_CTXT_SEL_DEC_CR_H2C 4'h6 = QDMA_CTXT_SEL_WRB 4'h7 = QDMA_CTXT_SEL_PFTCH 4'h8 = QDMA_CTXT_SEL_INT_COAL 4'h9 = 予約 4'hA = 予約 4'hB = QDMA_CTXT_SEL_TIMER 4'hC = QDMA_CTXT_SEL_FMAP
[0]	0	RO	busy	busy = 1 のとき、書き込みは破棄されます。 busy = 1 のとき、読み出しデータは無効です。

QDMA_TRQ_SEL_C2H (0x00A00)

表 141: QDMA_TRQ_SEL_C2H (0x00A00) レジスタ空間

レジスタ (アドレス)	アドレス	説明
QDMA_C2H_TIMER_CNT[16] (0xA00-0xA3C)	0xA00-0xA3C	CMPT タイマーしきい値のテーブル (間接)。
QDMA_C2H_CNT_TH[16] (0xA40-0xA7C)	0xA40-0xA7C	CMPT カウンターしきい値のテーブル (間接)。
QDMA_C2H_STAT_S_AXIS_C2H_ACCEPTED (0xA88)	0xA88	デバッグ ステータス レジスタ。受領された C2H パケットの数。
QDMA_C2H_STAT_S_AXIS_CMPT_ACCEPTED (0xA8C)	0xA8C	デバッグ ステータス レジスタ。受領された C2H CMPT パケットの数。
QDMA_C2H_STAT_DESC_RSP_PKT_ACCEPTED (0xA90)	0xA90	デバッグ ステータス レジスタ。プリフェッチから受領された desc_rsp パケットの数。
QDMA_C2H_STAT_AXIS_PKG_CMP (0xA94)	0xA94	デバッグ ステータス レジスタ。C2H DMA 書き込みエンジンからの完了した axis パケットの数。
QDMA_C2H_STAT_DESC_RSP_ACCEPTED (0xA98)	0xA98	デバッグ ステータス レジスタ。プリフェッチから受領された破棄およびエラーを含む desc_rsp の数。
QDMA_C2H_STAT_DESC_RSP_CMP (0xA9C)	0xA9C	デバッグ ステータス レジスタ。完了した desc_rsp の数で C2H DMA 書き込みエンジンの破棄およびエラーを含みます。
QDMA_C2H_STAT_WRQ_OUT (0xAA0)	0xAA0	デバッグ ステータス レジスタ。C2H DMA 書き込みエンジンから駆動される WRQ の数。
QDMA_C2H_STAT_WPL_REN_ACCEPTED (0xAA4)	0xAA4	デバッグ ステータス レジスタ。C2H DMA 書き込みエンジンで受領された WPL_REN の数。
QDMA_C2H_STAT_TOTAL_WRQ_LEN (0xAA8)	0xAA8	デバッグ ステータス レジスタ。C2H DMA 書き込みエンジンからの WRQ の長さ合計 (空のケットを含む)。
QDMA_C2H_STAT_TOTAL_WPL_LEN (0xAAC)	0xAAC	デバッグ ステータス レジスタ。C2H DMA 書き込みエンジンからの WPL の長さ合計 (空のケットを含む)。

表 141: QDMA_TRQ_SEL_C2H (00x00A00) レジスタ空間 (続き)

レジスタ (アドレス)	アドレス	説明
QDMA_C2H_BUF_SZ[16] (0xAB0-0xAEC)	0xAB0-0xAEC	バッファ サイズの選択肢。
QDMA_C2H_ERR_STAT (0xAF0)	0xAF0	C2H エラー ステータス。
QDMA_C2H_ERR_MASK (0xAF4)	0xAF4	C2H エラー イネーブル マスク。
QDMA_C2H_FATAL_ERR_STAT (0xAF8)	0xAF8	C2H 致命的エラー ステータス。
QDMA_C2H_FATAL_ERR_MASK (0xAFC)	0xAFC	C2H 致命的エラー イネーブル マスク。
QDMA_C2H_FATAL_ERR_ENABLE (0xB00)	0xB00	C2H 致命的エラー処理プロセスをイネーブルにします。
QDMA_GLBL_ERR_INT (0xB04)	0xB04	C2H エラー生成割り込み。
QDMA_C2H_PFCH_CFG (0xB08)	0xB08	プリフェッチ設定。
QDMA_C2H_INT_TIMER_TICK (0xB0C)	0xB0C	C2H 割り込みタイマーの刻み。
QDMA_C2H_STAT_DESC_RSP_DROP_ACCEPTED (0xB10)	0xB10	デバッグ ステータス レジスタ。受領された破棄を含むディスクリプター応答の数。
QDMA_C2H_STAT_DESC_RSP_ERR_ACCEPTED (0xB14)	0xB14	デバッグ ステータス レジスタ。受領されたエラーを含むディスクリプター応答の数。
QDMA_C2H_STAT_DESC_REQ (0xB18)	0xB18	デバッグ ステータス レジスタ。C2H DMA 書き込みエンジンから送信されるディスクリプター リクエストの数。
QDMA_C2H_STAT_DEBUG_DMA_ENG_0 (0xB1C)	0xB1C	デバッグ レジスタ 0。
QDMA_C2H_STAT_DEBUG_DMA_ENG_1 (0xB20)	0xB20	デバッグ レジスタ 1。
QDMA_C2H_STAT_DEBUG_DMA_ENG_2 (0xB24)	0xB24	デバッグ レジスタ 2。
QDMA_C2H_STAT_DEBUG_DMA_ENG_3 (0xB28)	0xB28	デバッグ レジスタ 3。
QDMA_C2H_DBG_PFCH_ERR_CTXT (0xB2C)	0xB2C	デバッグ ステータス レジスタ。
QDMA_C2H_FIRST_ERR_QID (0xB30)	0xB30	最初の C2H エラーの QID。
QDMA_STAT_NUM_WRB_IN (0xB34)	0xB34	デバッグ ステータス レジスタ。DmaWrEngine から WRB ブロックへ渡された WRB の数。
QDMA_STAT_NUM_WRB_OUT (0xB38)	0xB38	デバッグ ステータス レジスタ。WRB から WrbCoal ブロックへと渡された WRB の数 (STAT_DESC を除く)。
QDMA_STAT_NUM_WRB_DRP (0xB3C)	0xB3C	デバッグ ステータス レジスタ。WRB ブロック内で WRB が破棄された数。
QDMA_STAT_NUM_STAT_DESC_OUT (0xB40)	0xB40	デバッグ ステータス レジスタ。WRB から WrbCoal ブロックへと出力された STAT_DESC の数。
QDMA_STAT_NUM_DSC_CRDT_SENT (0xB44)	0xB44	デバッグ ステータス レジスタ。送信されたディスクリプター クレジット数と受信されたディスクリプター クレジット数 (キュー無効化の結果)。
QDMA_STAT_NUM_FCH_DSC_RCVD (0xB48)	0xB48	デバッグ ステータス レジスタ。フェッチ エンジンから受信したディスクリプターの数。
QDMA_STAT_NUM_BYP_DSC_RCVD (0xB4C)	0xB4C	デバッグ ステータス レジスタ。バイパスパスから受信したディスクリプターの数。
QDMA_C2H_WRB_COAL_CFG (0xB50)	0xB50	C2H 完了連結設定。
QDMA_C2H_INTR_H2C_REQ (0xB54)	0xB54	デバッグ ステータス レジスタ。H2C 割り込みリクエストの数。
QDMA_C2H_INTR_C2H_MM_REQ (0xB58)	0xB58	デバッグ ステータス レジスタ。C2H MM の割り込みリクエストの数。

表 141: QDMA_TRQ_SEL_C2H (00x00A00) レジスタ空間 (続き)

レジスタ (アドレス)	アドレス	説明
QDMA_C2H_INTR_ERR_INT_REQ (0xB5C)	0xB5C	デバッグ ステータス レジスタ。エラー生成の割り込みリクエストの数。
QDMA_C2H_INTR_C2H_ST_REQ (0xB60)	0xB60	デバッグ ステータス レジスタ。C2H ストリームの割り込みリクエストの数。
QDMA_C2H_INTR_H2C_ERR_C2H_MM_MSIX_ACK (0xB64)	0xB64	デバッグ ステータス レジスタ。H2C、C2H MM、およびエラー生成割り込みの MSIX ACK の数。
QDMA_C2H_INTR_H2C_ERR_C2H_MM_MSIX_FAIL (0xB68)	0xB68	デバッグ ステータス レジスタ。H2C、C2H MM、およびエラー生成割り込みの MSIX フェール数。
QDMA_C2H_INTR_H2C_ERR_C2H_MM_MSIX_NO_MSIX (0xB6C)	0xB6C	デバッグ ステータス レジスタ。H2C、C2H MM、およびエラー生成割り込みの NO MSIX の数。
QDMA_C2H_INTR_H2C_ERR_C2H_MM_CTXT_INVALID (0xB70)	0xB70	デバッグ ステータス レジスタ。H2C、C2H MM、およびエラー生成割り込みの無効割り込みリング ケース数。
QDMA_C2H_INTR_C2H_ST_MSIX_ACK (0xB74)	0xB74	デバッグ ステータス レジスタ。C2H ストリーム割り込みの MSIX ACK の数。
QDMA_C2H_INTR_C2H_ST_MSIX_FAIL (0xB78)	0xB78	デバッグ ステータス レジスタ。C2H ストリーム割り込みの MSIX フェールの数。
QDMA_C2H_INTR_C2H_ST_NO_MSIX (0xB7C)	0xB7C	デバッグ ステータス レジスタ。C2H ストリーム割り込みの NO MSIX の数。
QDMA_C2H_INTR_C2H_ST_CTXT_INVALID (0xB80)	0xB80	デバッグ ステータス レジスタ。C2H ストリーム割り込みの無効割り込みリング ケース数。
QDMA_C2H_STAT_WR_CMP (0xB84)	0xB84	デバッグ ステータス レジスタ。DMA 書き込みエンジンからのペイロード書き込み完了の数。
QDMA_C2H_STAT_DEBUG_DMA_ENG_4 (0xB88)	0xB88	DMA 書き込みエンジンのデバッグ レジスタ。
QDMA_C2H_DBG_PFCH_QID (0xB90)	0xB90	プリフェッチ モジュールのデバッグ レジスタ。
QDMA_C2H_DBG_PFCH (0xB94)	0xB94	プリフェッチ モジュールのデバッグ レジスタ。
QDMA_C2H_INT_DEBUG (0xB98)	0xB98	割り込みモジュールのデバッグ レジスタ。
QDMA_C2H_STAT_IMM_ACCEPTED (0xB9C)	0xB9C	デバッグ ステータス レジスタ。受領された即値データ パケットの数。
QDMA_C2H_STAT_MARKER_ACCEPTED (0xBA0)	0xBA0	デバッグ ステータス レジスタ。受領されたマーカー パケットの数。
QDMA_C2H_STAT_DISABLE_CMP_ACCEPTED (0xBA4)	0xBA4	デバッグ ステータス レジスタ。受領されたディスエーブル完了パケットの数。
QDMA_C2H_PAYLOAD_FIFO_CRDT_CNT (0xBA8)	0xBA8	デバッグ ステータス レジスタ。DMA 書き込みエンジンでのペイロード FIFO クレジット カウント数。
QDMA_C2H_INTR_DYN_REQ (0xBAC)	0xBAC	デバッグ ステータス レジスタ。割り込みエンジンに入ってくる割り込みアグリゲーション リング ダイナミック ポインタ アップデートの数。
QDMA_C2H_INTR_DYN_MSIX (0xBB0)	0xBB0	デバッグ ステータス レジスタ。PCIe-MSIX メッセージを送信させる割り込みアグリゲーション リング ダイナミック ポインタ アップデートの数。
QDMA_C2H_DROP_LEN_MISMATCH (0xBB4)	0xBB4	デバッグ ステータス レジスタ。破棄が発生したときの desc_rsp_eng.len が qid_fifo_out_data.len に等しくないケースの数。
QDMA_C2H_DROP_DESC_RSP_LEN (0xBB8)	0xBB8	デバッグ ステータス レジスタ。破棄が発生したときの desc_rsp_eng.len。
QDMA_C2H_DROP_QID_FIFO_LEN (0xBBC)	0xBBC	デバッグ ステータス レジスタ。破棄が発生したときの qid_fifo_out_data.len。

表 141: QDMA_TRQ_SEL_C2H (00x00A00) レジスタ空間 (続き)

レジスタ (アドレス)	アドレス	説明
QDMA_C2H_DROP_PAYLOAD_CNT (0xBC0)	0xBC0	デバッグ ステータス レジスタ。破棄が発生したときのペイロード FIFO クレジットの数。
QDMA_C2H_CMPT_FORMAT_0 (0xBC4)	0xBC4	完了エントリ フォーマット。
QDMA_C2H_CMPT_FORMAT_1 (0xBC8)	0xBC8	完了エントリ フォーマット。
QDMA_C2H_CMPT_FORMAT_2 (0xBCC)	0xBCC	完了エントリ フォーマット。
QDMA_C2H_CMPT_FORMAT_3 (0xBD0)	0xBD0	完了エントリ フォーマット。
QDMA_C2H_CMPT_FORMAT_4 (0xBD4)	0xBD4	完了エントリ フォーマット。
QDMA_C2H_CMPT_FORMAT_5 (0xBD8)	0xBD8	完了エントリ フォーマット。
QDMA_C2H_CMPT_FORMAT_6 (0xBDC)	0xBDC	完了エントリ フォーマット。
QDMA_C2H_PFCH_CACHE_DEPTH (0xBE0)	0xBE0	プリフェッチ キャッシュ サイズ。
QDMA_C2H_CMPT_COAL_BUF_DEPTH (0xBE4)	0xBE4	CMPT 連結バッファの深さ。
QDMA_C2H_PFCH_CRDT (0xBE8)	0xBE8	デバッグ レジスタ。プリフェッチ モジュールからのクレジット。
QDMA_C2H_STAT_HAS_CMPT_ACCEPTED (0xBEC)	0xBEC	デバッグ レジスタ。完了を含むデータ パケットの数。
QDMA_C2H_STAT_HAS_PLD_ACCEPTED (0xBF0)	0xBF0	デバッグ レジスタ。データ ペイロードを含む完了パケットの数。
MDMA_C2H_PLD_PKT_ID (0xBF4)	0xBF4	デバッグ レジスタ。データ ペイロード パケット ID。

QDMA_C2H_TIMER_CNT[16] (0xA00-0xA3C)

表 142: QDMA_C2H_TIMER_CNT[16] (0xA00-0xA3C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:8]	0	NA		予約
[7:0]	0	RW	timer_count	タイマーしきい値

タイマーしきい値は 16 個のレジスタから成る 1 つのグループで、C2H 完了コンテキストでは、タイマー カウント インデックス フィールドを使用してタイマー値が選択されます。

QDMA_C2H_CNT_TH[16] (0xA40-0xA7C)

表 143: QDMA_C2H_CNT_TH[16] (0xA40-0xA7C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:8]	0	NA		予約
[7:0]	0	RW	threshold_count	カウントしきい値

カウントしきい値は 16 個のレジスタから成る 1 つのグループで、C2H 完了コンテキストでは、カウントしきい値 インデックス フィールドを使用してこの値が選択されます。

QDMA_C2H_STAT_S_AXIS_C2H_ACCEPTED (0xA88)

表 144: QDMA_C2H_STAT_S_AXIS_C2H_ACCEPTED (0xA88)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	c2h_accepted	ユーザー アプリケーションから受領された C2H パケットの数。

QDMA_C2H_STAT_S_AXIS_CMPT_ACCEPTED (0xA8C)

表 145: QDMA_C2H_STAT_S_AXIS_CMPT_ACCEPTED (0xA8C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	cmpt_accepted	ユーザー アプリケーションから受領された C2H 完了パケットの数。

QDMA_C2H_STAT_DESC_RSP_PKT_ACCEPTED (0xA90)

表 146: QDMA_C2H_STAT_DESC_RSP_PKT_ACCEPTED (0xA90)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	dsc_rsp_pkt_accepted	受領された desc_rsp パケットの数

QDMA_C2H_STAT_AXIS_PKG_CMP (0xA94)

表 147: QDMA_C2H_STAT_AXIS_PKG_CMP (0xA94)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	pkg_cmp	C2H DMA 書き込みエンジンからの完了した C2H パケットの数。

QDMA_C2H_STAT_DESC_RSP_ACCEPTED (0xA98)

表 148: QDMA_C2H_STAT_DESC_RSP_ACCEPTED (0xA98)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	dsc_rsp_accepted	受領された破棄およびエラーを含む desc_rsp の数。

QDMA_C2H_STAT_DESC_RSP_CMP (0xA9C)

表 149: QDMA_C2H_STAT_DESC_RSP_CMP (0xA9C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	dsc_rsp_cmp	完了した desc_rsp の数で、C2H DMA 書き込みエンジンの破棄およびエラーを含みます。

QDMA_C2H_STAT_WRQ_OUT (0xAA0)

表 150: QDMA_C2H_STAT_WRQ_OUT (0xAA0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	wrq_out	C2H DMA 書き込みエンジンから駆動される WRQ の数。

QDMA_C2H_STAT_WPL_REN_ACCEPTED (0xAA4)

表 151: QDMA_C2H_STAT_WPL_REN_ACCEPTED (0xAA4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	wpl_ren_accepted	C2H DMA 書き込みエンジンで受領された WPL_REN の数。

QDMA_C2H_STAT_TOTAL_WRQ_LEN (0xAA8)

表 152: QDMA_C2H_STAT_TOTAL_WRQ_LEN (0xAA8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	total_wrq_len	C2H DMA 書き込みエンジンからの WRQ の長さ合計 (空の пакетを含む)。

QDMA_C2H_STAT_TOTAL_WPL_LEN (0xAAC)

表 153: QDMA_C2H_STAT_TOTAL_WPL_LEN (0xAAC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	total_wpl_len	C2H DMA 書き込みエンジンからの WPL の長さ合計 (空の пакетを含む)。

QDMA_C2H_BUF_SZ[16] (0xAB0-0xAEC)

表 154: QDMA_C2H_BUF_SZ[16] (0xAB0-0xAEC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	NA		予約
[15:0]	0	RW	size	キューにある各ディスクリプターの C2H バッファ サイズ (最大は 64 K-1)。

16 個のレジスタがあって、それぞれに異なる C2H バッファ サイズを指定できます。バッファはコンテキスト プログラミングで選択できます。

QDMA_C2H_ERR_STAT (0xAF0)

表 155: QDMA_C2H_ERR_STAT (0xAF0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	NA		予約
[15]	0	RW	wrb_prty_err	C2H 完了にパリティ エラーが検出されたことを示します。
[14]	0	RW	wrb_cidx_err	ソフトウェアにより無効な CIDX アップデートが C2H-ST 完了エンジンに送信されたことを示します。
[13]	0	RW	wrb_qfull_err	完了キューがフルであることを示します。
[12]	0	RW	wrb_inv_q_err	ソフトウェアにより CMPT CIDX アップデートが無効なキューに送られると、このエラーがフラグされます。
[11]	0	RW	port_id_byp_in_mismatch	C2H パケットからのポート ID とバイパス入力からのポート ID が一致しません。
[10]	0	RW	port_id_ctxt_mismatch	C2H パケットからのポート ID とプリフェッチ コンテキストのポート ID が一致しません。
[9]	0	RW	err_desc_cnt	パケット内のディスクリプターの数が 7 よりも大きい場合、エラーがフラグされます。
[8]	0	RW		予約
[7]	0	RW	msi_int_fail	MSI-X 割り込みメッセージがフェール応答を受信したことを示します。
[6]	0	RW	eng_wpl_data_par_err	データ パリティ エラー
[5]	0	RW		予約
[4]	0	RW	desc_rsp_err	ディスクリプターのエラー ビットがセットされています。
[3]	0	RW	qid_mismatch	s_axis_c2h_ctrl.qid からの QID が s_axis_wrb_data の QID と一致しない場合、エラーがフラグされます。
[2]	0	RW	Rsvd3	予約
[1]	0	RW	len_mismatch	パケットの合計の長さが s_axis_c2h_ctrl.len からの信号に一致しない場合は、エラーがフラグされます。
[0]	0	RW	mty_mismatch	最終パケットでない場合、Mty は 0 である必要があります。そうでない場合は、エラーがフラグされます。

これは C2H エラーのエラー記録レジスタです。エラーが発生するとハードウェアによりレジスタに書き込まれます。必要に応じて、ソフトウェアで 1'b1 を書き込んでエラーをクリアできます。QDMA_C2H_ERR_MASK レジスタはエラーの記録には影響しません。

QDMA_C2H_ERR_MASK (0xAF4)

表 156: QDMA_C2H_ERR_MASK (0xAF4)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:0]	0	RW	c2h_err_en_mask	C2H エラー イネーブル マスク

対応する C2H エラーがエラー アグリゲーターに伝搬されるよう、ソフトウェアによりこのビットがセットされます。

QDMA_C2H_FATAL_ERR_STAT (0xAF8)

表 157: QDMA_C2H_FATAL_ERR_STAT (0xAF8)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:19]	0	RO		予約
[18]	0	RO	wpl_data_par_err	RAM ダブルビットエラー
[17]	0	RO	payload_fifo_ram_rdbe	RAM ダブルビットエラー
[16]	0	RO	qid_fifo_ram_rdbe	RAM ダブルビットエラー
[15]	0	RO	tuser_fifo_ram_rdbe	RAM ダブルビットエラー
[14]	0	RO	wrb_coal_data_ram_rdbe	RAM ダブルビットエラー
[13]	0	RO		予約
[12]	0	RO	int_ctxt_ram_rdbe	RAM ダブルビットエラー
[11]	0	RO	desc_req_fifo_ram_rdbe	RAM ダブルビットエラー
[10]	0	RO	pfch_ctxt_ram_rdbe	RAM ダブルビットエラー
[9]	0	RO	wrb_ctxt_ram_rdbe	RAM ダブルビットエラー
[8]	0	RO	pfch_ll_ram_rdbe	RAM ダブルビットエラー
[7:4]	0	RO	timer_fifo_ram_rdbe	RAM ダブルビットエラー
[3]	0	RO	qid_mismatch	s_axis_c2h_ctrl.qid からの QID が s_axis_wrb_data の QID と一致しない場合、エラーがフラグされます。
[2]	0	RO		予約
[1]	0	RO	len_mismatch	パケットの合計の長さが s_axis_c2h_ctrl.len からの信号に一致しない場合は、エラーがフラグされます。
[0]	0	RO	mty_mismatch	最終パケットでない場合、Mty は 0 である必要があります。そうでない場合は、エラーがフラグされます。

これは、C2H 致命的エラーのエラー記録レジスタです。エラーが発生するとハードウェアによりレジスタに書き込まれます。必要に応じて、ソフトウェアで 1'b1 を書き込んでエラーをクリアできます。QDMA_C2H_FATAL_ERR_MASK レジスタはエラーの記録には影響しません。

QDMA_C2H_FATAL_ERR_MASK (0xAFC)

表 158: QDMA_C2H_FATAL_ERR_MASK (0xAFC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	c2h_fatal_err_en_mask	C2H 致命的エラー イネーブル マスク

対応する C2H 致命的エラーが C2H 致命的処理ロジックに送信されるように、ソフトウェアによりこのビットがセットされます。

QDMA_C2H_FATAL_ERR_ENABLE (0xB00)

表 159: QDMA_C2H_FATAL_ERR_ENABLE (0xB00)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:2]	0	RW		予約
[1]	0	RW	enable_wpl_par_inv	致命的エラーが発生すると、C2H Wpl パリティ反転がイネーブルになります。
[0]	0	RW	enable_wrq_dis	致命的エラーが発生すると、C2H Wrq ディスエーブルがイネーブルになります。

このレジスタは、C2H 致命的エラー処理プロセスをイネーブルにします。

- Wrq をディスエーブルにすることでデータ転送が停止します。
- データ転送で WPL パリティを反転させます。

QDMA_GLBL_ERR_INT (0xB04)

このレジスタはエラー生成割り込み用です。

表 160: QDMA_GLBL_ERR_INT (0xB04)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:25]	0	NA		予約
[24]	0	RW	err_int_arm	エラー割り込みを準備するため、ソフトウェアによりビットがセットされます。割り込みが割り込みモジュールで使用されると、ハードウェアでビットがクリアになります。エラー割り込みを生成するため、ソフトウェアでこのビットを再準備する必要があります。
[23]	0	RW		予約
[22:12]	0	RW	vec	直接エラー割り込みの場合、これは割り込みベクターです。間接エラー割り込みの場合、これは割り込みアグリゲーションのコンテキスト RAM インデックスです。
[11:8]	0	NA		予約
[7:0]	0	RW	func	ファンクション

QDMA_C2H_PFCH_CFG (0xB08)

表 161: QDMA_C2H_PFCH_CFG (0B08)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:25]	56	RW	evt_qcnt_th	プリフェッチ キュー カウントが evt_qcnt_th 以上になると、ハードウェアがエヴィクションを開始します。evt_qcnt_th は (pfch_qcnt - 2) 以下である必要があります。推奨値は 14 です。
[24:18]	60	RW	pfch_qcnt	プリフェッチ キュー カウントの最大許容数。最大値は 60 です。推奨値は 16 です。
[17:9]	16	RW	num_pfch	キューごとにキャッシュでプリフェッチされたエントリの数を制御します。推奨値は 8 です。
[8:0]	16	RW	pfch_fl_th	使用可能なディスクリプター空間が pfch_fl_th 以下のときに (最小値は 16)、プリフェッチを停止します。推奨値は 256 です。

QDMA_C2H_INT_TIMER_TICK (0xB0C)

表 162: QDMA_C2H_INT_TIMER_TICK (0xB0C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	timer_tick	ユーザー クロックでの C2H タイマー サイクルの値。

QDMA_C2H_STAT_DESC_RSP_DROP_ACCEPTED (0xB10)

表 163: QDMA_C2H_STAT_DESC_RSP_DROP_ACCEPTED (0xB10)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	dsc_rsp_drop_accepted	受領された破棄を含むディスクリプター 応答の数

QDMA_C2H_STAT_DESC_RSP_ERR_ACCEPTED (0xB14)

表 164: QDMA_C2H_STAT_DESC_RSP_ERR_ACCEPTED (0xB14)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	dsc_rsp_err_accepted	受領されたエラーを含むディスクリプター 応答の数

QDMA_C2H_STAT_DESC_REQ (0xB18)

表 165: QDMA_C2H_STAT_DESC_REQ (0xB18)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	desc_req	C2H DMA 書き込みエンジンから送信されるディスクリプター リクエストの数。

QDMA_C2H_STAT_DEBUG_DMA_ENG_0 (0xB1C)

表 166: QDMA_C2H_STAT_DEBUG_DMA_ENG_0 (0xB1C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]	0	RO	s_axis_c2h_tready	s_axis_c2h_tready
[30:28]	0	RO	wrb_fifo_out_cnt	WRB FIFO のカウント。
[27:18]	0	RO	qid_fifo_out_cnt	QID FIFO のカウント。
[17:8]	0	RO	payload_fifo_out_cnt	ペイロード FIFO のカウント。
[7:5]	0	RO	wrq_fifo_out_cnt	WRQ FIFO のカウント。
[4]	0	RO	wrb_sm_cs	CMPT ステート マシン。
[3:0]	0	RO	main_sm_cs	メイン ステート マシン。

これは C2H DMA 書き込みエンジンのデバッグ レジスタです。

QDMA_C2H_STAT_DEBUG_DMA_ENG_1 (0xB20)

表 167: QDMA_C2H_STAT_DEBUG_DMA_ENG_1 (0xB20)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]	1	NA	tuser_comb_in_rdy	tuser_comb_in_rdy
[30]	0	RO	desc_rsp_last	desc_rsp_last
[29:20]	0	RO	payload_fifo_in_cnt	ペイロード FIFO への入力エントリの数。
[19:10]	0	RO	payload_fifo_output_cnt	ペイロード FIFO からの出力エントリの数
[9:0]	0	RO	qid_fifo_in_cnt	QID FIFO への入力エントリの数。

これは C2H DMA 書き込みエンジンのデバッグ レジスタです。

QDMA_C2H_STAT_DEBUG_DMA_ENG_2 (0xB24)

表 168: QDMA_C2H_STAT_DEBUG_DMA_ENG_2 (0xB24)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]	0	RO	s_axis_wrb_tready	s_axis_wrb_tready
[30]	0	RO	wrb_fifo_in_rdy	wrb fifo 入力レディ
[29:20]	0	RO	wrb_fifo_in_cnt	WRB FIFO への入力エントリの数。
[19:10]	0	RO	wrb_fifo_output_cnt	WRB FIFO からの出力エントリの数。
[9:0]	0	RO	qid_fifo_output_cnt	QID FIFO からの出力エントリの数。

これは C2H DMA 書き込みエンジンのデバッグ レジスタです。

QDMA_C2H_STAT_DEBUG_DMA_ENG_3 (0xB28)

表 169: QDMA_C2H_STAT_DEBUG_DMA_ENG_3 (0xB28)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:25]	0	NA		予約
[24:21]	0	RO	pld_st_fifo_out_cnt	Pld_st fifo のエントリの数。
[20:20]	0	RO	pld_pkt_id_larger	ペイロード パケット ID は、CMPT パケットが予期する値よりも大きくなります。
[19:10]	0	RO	wrq_fifo_in_cnt	WRQ FIFO への入力エントリの数。
[9:0]	0	RO	wrq_fifo_output_cnt	WRQ FIFO からの出力エントリの数。

QDMA_C2H_DBG_PFCH_ERR_CTXT (0xB2C)

表 170: QDMA_C2H_DBG_PFCH_ERR_CTXT (0xB2C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:14]	0	RW		予約
[13]	0	RW	err_stat	エラー ステータス 読み出しコマンドの場合 キューが有効な場合は、err_stat = 0 キューが無効な場合は、err_stat = 1
[12]	0	RW	cmd_wr	書き込みまたは読み出しのコマンド。 1: 書き込み 0: 読み出し
[11:1]	0	RW	qid	キュー ID。
[0]	0	RW	done	完了。操作終了。

QDMA_C2H_FIRST_ERR_QID (0xB30)

表 171: QDMA_C2H_FIRST_ERR_QID (0xB30)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:20]		NA		予約

表 171: QDMA_C2H_FIRST_ERR_QID (0xB30) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[19:16]	0	RO	err_type	4'b1111: NA 4'b1110: wrb_cidx_err 4'b1101: wrb_qfull_err 4'b1100: wrb_inv_q_err 4'b1011: port_id_ctxt_mismatch 4'b1010: port_id_byp_in_mismatch 4'b1001: err_desc_cnt 4'b0111: msi_int_fail 4'b0110: eng_wpl_data_par_err 4'b0100: desc_rsp_error 4'b0011: qid_mismatch 4'b0001: len_mismatch 4'b0000: mty_mismatch
[15:11]	0	NA		予約
[10:0]	0	RO	qid	最初の C2H エラーの QID

このレジスタでは、最初の C2H エラー タイプおよび QID が記録されます。err_type をクリアにして 4'b1111 にするように、ソフトウェアによりこのレジスタに書き込みが実行されます。

QDMA_STAT_NUM_WRB_IN (0xB34)

表 172: QDMA_STAT_NUM_WRB_IN (0xB34)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	NA		予約
[15:0]	0	RO	wrb_cnt	DmaWrEnginre から WRB ブロックへ渡された WRB の数。

QDMA_STAT_NUM_WRB_OUT (0xB38)

表 173: QDMA_STAT_NUM_WRB_OUT (0xB38)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	NA		予約
[15:0]	0	RO	wrb_cnt	Wrb から WrbCoal ブロックへと渡された WRB の数 (STAT_DESC を除く)。

QDMA_STAT_NUM_WRB_DRP (0xB3C)

表 174: QDMA_STAT_NUM_WRB_DRP (0xB3C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	NA		予約

表 174: QDMA_STAT_NUM_WRB_DRP (0xB3C) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[15:0]	0	RO	wrb_cnt	WRB ブロック内で WRB が破棄された数

QDMA_STAT_NUM_STAT_DESC_OUT (0xB40)

表 175: QDMA_STAT_NUM_STAT_DESC_OUT (0xB40)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	NA		予約
[15:0]	0	RO	stat_desc_cnt	Wrb から WrbCoal ブロックへと出力された STAT_DESC の数。

QDMA_STAT_NUM_DSC_CRDT_SENT (0xB44)

表 176: QDMA_STAT_NUM_DSC_CRDT_SENT (0xB44)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	NA		予約
[15:0]	0	RO	crdt_cnt	送信されたディスクリプター クレジット数と受信されたディスクリプター クレジット数 (キュー無効化の結果)。

QDMA_STAT_NUM_FCH_DSC_RCVD (0xB48)

表 177: QDMA_STAT_NUM_FCH_DSC_RCVD (0xB48)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	NA		予約
[15:0]	0	RO	dsc_cnt	フェッチ エンジンから受信したディスクリプターの数。

QDMA_STAT_NUM_BYP_DSC_RCVD (0XB4C)

表 178: QDMA_STAT_NUM_BYP_DSC_RCVD (0XB4C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:11]	0	NA		予約
[10:0]	0	RO	dsc_cnt	バイパス パスから受信したディスクリプターの数。

QDMA_C2H_WRB_COAL_CFG (0xB50)

表 179: QDMA_C2H_WRB_COAL_CFG (0xB50)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:26]	32	RW	max_buf_sz	結合バッファのサイズを設定します。有効範囲をカバーするために使用します。 QDMA_C2H_CMPT_COAL_BUF_DEPTH レジスタで説明されているように、CMPT 結合バッファのサイズ未満でこのフィールドをプログラムしないように注意する必要があります。
[25:14]	20	RW	tick_val	結合バッファのタイマー値
[13:2]	4	RW	tick_cnt	結合バッファのタイマー カウント値
[1]	0	RW	set_glb_flush	完了を受信するとすぐに結合バッファがエントリを空にします。
[0]	0	RO	done_glb_flush	結合バッファがエントリを空にすると、このビットがセットされます。

QDMA_C2H_INTR_H2C_REQ (0xB54)

表 180: QDMA_C2H_INTR_H2C_REQ (0xB54)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。H2C 割り込みリクエストの数。

QDMA_C2H_INTR_C2H_MM_REQ (0xB58)

表 181: QDMA_C2H_INTR_C2H_MM_REQ (0xB58)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。C2H MM の割り込みリクエストの数。

QDMA_C2H_INTR_ERR_INT_REQ (0xB5C)

表 182: QDMA_C2H_INTR_ERR_INT_REQ (0xB5C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。エラー生成の割り込みリクエストの数。

QDMA_C2H_INTR_C2H_ST_REQ (0xB60)

表 183: QDMA_C2H_INTR_C2H_ST_REQ (0xB60)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。C2H ストリームの割り込みリクエストの数。

QDMA_C2H_INTR_H2C_ERR_C2H_MM_MSIX_ACK (0xB64)

表 184: QDMA_C2H_INTR_H2C_ERR_C2H_MM_MSIX_ACK (0xB64)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。H2C、C2H MM、およびエラー生成割り込みの MSIX ACK の数。

QDMA_C2H_INTR_H2C_ERR_C2H_MM_MSIX_FAIL (0xB68)

表 185: QDMA_C2H_INTR_H2C_ERR_C2H_MM_MSIX_FAIL (0xB68)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。H2C、C2H MM、およびエラー生成割り込みの MSIX フェール数。

QDMA_C2H_INTR_H2C_ERR_C2H_MM_MSIX_NO_MSIX (0xB6C)

表 186: QDMA_C2H_INTR_H2C_ERR_C2H_MM_MSIX_NO_MSIX (0xB6C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。H2C、C2H MM、およびエラー生成割り込みの NO MSIX の数。

QDMA_C2H_INTR_H2C_ERR_C2H_MM_CTXT_INVALID (0xB70)

表 187: QDMA_C2H_INTR_H2C_ERR_C2H_MM_CTXT_INVALID (0xB70)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。H2C、C2H MM、およびエラー生成割り込みの割り込みコンテキスト無効ケースの数。

QDMA_C2H_INTR_C2H_ST_MSIX_ACK (0xB74)

表 188: QDMA_C2H_INTR_C2H_ST_MSIX_ACK (0xB74)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。C2H ストリーム割り込みの MSIX ACK の数。

QDMA_C2H_INTR_C2H_ST_MSIX_FAIL (0xB78)

表 189: QDMA_C2H_INTR_C2H_ST_MSIX_FAIL (0xB78)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。C2H ストリーム割り込みの MSIX フェールの数。

QDMA_C2H_INTR_C2H_ST_NO_MSIX (0xB7C)

表 190: QDMA_C2H_INTR_C2H_ST_NO_MSIX (0xB7C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。C2H ストリーム割り込みの NO MSIX の数。

QDMA_C2H_INTR_C2H_ST_CTXT_INVALID (0xB80)

表 191: QDMA_C2H_INTR_C2H_ST_CTXT_INVALID (0xB80)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。C2H 割り込みの割り込みコンテキストの無効ケースの数。

QDMA_C2H_STAT_WR_CMP (0xB84)

表 192: QDMA_C2H_STAT_WR_CMP (0xB84)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	デバッグ ステータス レジスタ。DMA 書き込みエンジンからのペイロード書き込み完了の数。

QDMA_C2H_STAT_DEBUG_DMA_ENG_4 (0xB88)

表 193: QDMA_C2H_STAT_DEBUG_DMA_ENG_4 (0xB88)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]	0	RO	tuser_fifo_out_vld	tuser fifo 出力が有効。
[30]	1	RO	tuser_fifo_in_rdy	tuser fifo 入力レディ 信号。
[29:20]	0	RO	tuser_fifo_in_cnt	tuser fifo への入力エントリの数。
[19:10]	0	RO	tuser_fifo_output_cnt	tuser fifo からの出力エントリの数。
[9:0]	0	RO	tuser_fifo_out_cnt	tuser fifo のエントリの数。

QDMA_C2H_DBG_PFCH_QID (0xB90)

表 194: QDMA_C2H_DBG_PFCH_QID (0xB90)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:15]	0	RW		予約
[14]	0	RW	err_ctxt	エラー コンテキスト RAM へ書き込まれたデータ
[13:11]	0	RW	target	3'h0: Key_cam 3'h1: Tag_st 3'h2: Tag_used_cnt 3'h3: Tag_desc_cnt 3'h4: エラー コンテキスト RAM
[10:0]	0	RW	qid_or_tag	QID またはタグ

QDMA_C2H_DBG_PFCH (0xB94)

表 195: QDMA_C2H_DBG_PFCH (0xB94)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	data	データ

上記の 2 つはプリフェッチ モジュールのデバッグ レジスタです。まず、ターゲットおよび QID (またはタグ) を設定するため、QDMA_C2H_DBG_PFCH_QID レジスタに書き込みます。それから、次の作業をするために、QDMA_C2H_DBG_PFCH レジスタへの読み出しまたは書き込みを実行します。

- 各タグのキー CAM を読み出します。
- 各タグの tag_st を読み出します。
- 各タグの tag_used_cnt を読み出します。
- 各タグの tag_desc_cnt を読み出します。
- 各キューのエラー コンテキスト RAM の読み出しまたは書き込みを実行します。

QDMA_C2H_INT_DEBUG (0xB98)

表 196: QDMA_C2H_INT_DEBUG (0xB98)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:8]	0	NA		予約
[7:4]	0	RO	int_coal_sm	割り込みアグリゲーション モジュールのステート マシン。
[3:0]	0	RO	int_sm	割り込みモジュールのステート マシン。

QDMA_C2H_STAT_IMM_ACCEPTED (0xB9C)

表 197: QDMA_C2H_STAT_IMM_ACCEPTED (0xB9C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	受領された即値データ パケットの数。

QDMA_C2H_STAT_MARKER_ACCEPTED (0xBA0)

表 198: QDMA_C2H_STAT_MARKER_ACCEPTED (0xBA0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	受領されたマーカー パケットの数。

QDMA_C2H_STAT_DISABLE_CMP_ACCEPTED (0xBA4)

表 199: QDMA_C2H_STAT_DISABLE_CMP_ACCEPTED (0xBA4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	cnt	受領されたディスエーブル完了パケットの数。

QDMA_C2H_PAYLOAD_FIFO_CRDT_CNT (0xBA8)

表 200: QDMA_C2H_PAYLOAD_FIFO_CRDT_CNT (0xBA8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[18:0]	0	RO	cnt	DMA 書き込みエンジンでのペイロード FIFO クレジット カウント。

QDMA_C2H_INTR_DYN_REQ (0xBAC)

表 201: QDMA_C2H_INTR_DYN_REQ (0xBAC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	num	デバッグ ステータス レジスタ。割り込みアグリゲーション リング ポインター アップデートの数。

QDMA_C2H_INTR_DYN_MSIX (0xBB0)

表 202: QDMA_C2H_INTR_DYN_MSIX (0xBB0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	num	デバッグ ステータス レジスタ。PCIe MSI-X メッセージを生成させる割り込みアグリゲーション リング ポインター アップデートの数。

QDMA_C2H_DROP_LEN_MISMATCH (0xBB4)

表 203: QDMA_C2H_DROP_LEN_MISMATCH (0xBB4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	num	デバッグ ステータス レジスタ。破棄が発生したとき、desc_rsp_eng.len が qid_fifo_out_data.len に等しくないケースの数。

QDMA_C2H_DROP_DESC_RSP_LEN (0xBB8)

表 204: QDMA_C2H_DROP_DESC_RSP_LEN (0xBB8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	num	デバッグ ステータス レジスタ。破棄が発生したときの desc_rsp_eng.len。

QDMA_C2H_DROP_QID_FIFO_LEN (0xBBC)

表 205: QDMA_C2H_DROP_QID_FIFO_LEN (0xBBC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	num	デバッグ ステータス レジスタ。破棄が発生したときの qid_fifo_out_data.len。

QDMA_C2H_DROP_PAYLOAD_CNT (0xBC0)

表 206: QDMA_C2H_DROP_PAYLOAD_CNT (0xBC0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	num	デバッグ ステータス レジスタ。破棄が発生したときのペイロード FIFO クレジットの数。

QDMA_C2H_CMPT_FORMAT_0 (0xBC4)

表 207: QDMA_C2H_CMPT_FORMAT_0 (0xBC4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	-	RO	desc_err_loc	LSB から計測された CMPT エントリでの desc_err ビットの 9 ビット オフセットの Vivado IDE オプション 0。
[15:0]	-	RO	color_loc	LSB から計測された CMPT エントリでのカラー ビットの 9 ビット オフセットの Vivado IDE オプション 0。

このレジスタのフィールドのデフォルト値は、IP 生成時に決まります。

QDMA_C2H_CMPT_FORMAT_1 (0xBC8)

表 208: QDMA_C2H_CMPT_FORMAT_1 (0xBC8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	RO	desc_err_loc	LSB から計測された CMPT エントリでの desc_err ビットの 9 ビット オフセットの Vivado IDE オプション 1。
[15:0]	0	RO	color_loc	LSB から計測された CMPT エントリでのカラー ビットの 9 ビット オフセットの Vivado IDE オプション 1。

このレジスタのフィールドのデフォルト値は、IP 生成時に決まります。

QDMA_C2H_CMPT_FORMAT_2 (0xBCC)

表 209: QDMA_C2H_CMPT_FORMAT_2 (0xBCC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	RO	desc_err_loc	LSB から計測された CMPT エントリでの desc_err ビットの 9 ビット オフセットの Vivado IDE オプション 2。
[15:0]	0	RO	color_loc	LSB から計測された CMPT エントリでのカラー ビットの 9 ビット オフセットの Vivado IDE オプション 2。

このレジスタのフィールドのデフォルト値は、IP 生成時に決まります。

QDMA_C2H_CMPT_FORMAT_3 (0xBD0)

表 210: QDMA_C2H_CMPT_FORMAT_3 (0xBD0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	RO	desc_err_loc	LSB から計測された CMPT エントリでの desc_err ビットの 9 ビット オフセットの Vivado IDE オプション 3。
[15:0]	0	RO	color_loc	LSB から計測された CMPT エントリでのカラー ビットの 9 ビット オフセットの Vivado IDE オプション 3。

このレジスタのフィールドのデフォルト値は、IP 生成時に決まります。

QDMA_C2H_CMPT_FORMAT_4 (0xBD4)

表 211: QDMA_C2H_CMPT_FORMAT_4 (0xBD4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	RO	desc_err_loc	LSB から計測された CMPT エントリでの desc_err ビットの 9 ビット オフセットの Vivado IDE オプション 4。
[15:0]	0	RO	color_loc	LSB から計測された CMPT エントリでのカラー ビットの 9 ビット オフセットの Vivado IDE オプション 4。

このレジスタのフィールドのデフォルト値は、IP 生成時に決まります。

QDMA_C2H_CMPT_FORMAT_5 (0xBD8)

表 212: QDMA_C2H_CMPT_FORMAT_5 (0xBD8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	RO	desc_err_loc	LSB から計測された CMPT エントリでの desc_err ビットの 9 ビット オフセットの GUI オプション 5。
[15:0]	0	RO	color_loc	LSB から計測された CMPT エントリでのカラー ビットの 9 ビット オフセットの GUI オプション 5。

このレジスタのフィールドのデフォルト値は、IP 生成時に決まります。

QDMA_C2H_CMPT_FORMAT_6 (0xBDC)

表 213: QDMA_C2H_CMPT_FORMAT_6 (0xBDC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	RO	desc_err_loc	LSB から計測された CMPT エントリでの desc_err ビットの 9 ビット オフセットの GUI オプション 6。
[15:0]	0	RO	color_loc	LSB から計測された CMPT エントリでのカラー ビットの 9 ビット オフセットの GUI オプション 6。

このレジスタのフィールドのデフォルト値は、IP 生成時に決まります。

QDMA_C2H_PFCH_CACHE_DEPTH (0xBE0)

表 214: QDMA_C2H_PFCH_CACHE_DEPTH (0xBE0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:7]	0	NA		予約
[6:0]	0	RO	size	プリフェッチ キャッシュ サイズ: 8、16、32、または 64。

プリフェッチ キャッシュでは、最大 64 個のキューがサポートされます。8、16、32、または 64 を選択します。プリフェッチ キャッシュでは、常に選択した値までのアクティブ キューをサポートできます。アクティブ キューの 1 つがフェッチを完了し、そのキューのポケットのディスクリプターをすべて送信すると、ほかのアクティブ キューで使用できるようにキャッシュ エントリが解放されます。キャッシュ サイズが大きいと、より多くのキューをサポートできますが、そうするとエリアも増大します。

QDMA_C2H_CMPT_COAL_BUF_DEPTH (0xBE4)

表 215: QDMA_C2H_CMPT_COAL_BUF_DEPTH (0xBE4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:7]	0	RO		予約
[6:0]	-	RO	buffer depth	IP を構築するとき、完了結合バッファの深さを決めるときに、ソフトウェアでこのレジスタが使用されます。有効な値は、8、16、32、および 64 です。

CMPT 結合バッファの各エントリは、帯域幅の使用率を改善するため、ホストに書き込む前に複数の完了 (64B まで) を結合して 1 つのキューにします。CMPT 結合バッファがより深いと、複数のキューの中で完了を結合できませんが、一方でエリアが増えてしまうのがマイナスです。

QDMA_C2H_PFCH_CRDT (0xBE8)

表 216: QDMA_C2H_PFCH_CRDT (0xBE8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:1]	0	RO		予約
[0:0]	0	RO	fence	プリフェッチ モジュールからのクレジット。フェンスビットブロックは、フェッチがアップデートを完了するまで、クレジットのアップデートを継続します。

QDMA_C2H_STAT_HAS_CMPT_ACCEPTED (0xBEC)

表 217: QDMA_C2H_STAT_HAS_CMPT_ACCEPTED (0xBEC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	data	完了を含む C2H データ パケットの数。

QDMA_C2H_STAT_HAS_PLD_ACCEPTED (0xBF0)

表 218: QDMA_C2H_STAT_HAS_PLD_ACCEPTED (0xBF0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	data	データ ペイロードを含む C2H 完了パケットの数。

MDMA_C2H_PLD_PKT_ID (0xBF4)

表 219: QDMA_C2H_PLD_PKT_ID (0xBF4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:18]	0	NA		予約
[17:0]	0	RO	data	C2H DMA 書き込みエンジンが処理したデータ ペイロード パケット ID。

QDMA_TRQ_SEL_H2C (0x00E00)

表 220: QDMA_TRQ_SEL_H2C (0x00E00) レジスタ空間

レジスタ名	アドレス (16 進数)	説明
QDMA_H2C_ERR_STAT (0xE00)	0xE00	H2C エラー ステータス
QDMA_H2C_ERR_MASK (0xE04)	0xE04	H2C エラー マスク
QDMA_H2C_FIRST_ERR_QID (0xE08)	0xE08	H2C-ST で最初に発生したエラーの QID およびタイプ
QDMA_H2C_DBG_REG0 (0xE0C)	0xE0C	H2C-ST デバッグ レジスタ 0
QDMA_H2C_DBG_REG1 (0xE10)	0xE10	H2C-ST デバッグ レジスタ 1
QDMA_H2C_DBG_REG2 (0xE14)	0xE14	H2C-ST デバッグ レジスタ 2
QDMA_H2C_DBG_REG3 (0xE18)	0xE18	H2C-ST デバッグ レジスタ 3
QDMA_H2C_DBG_REG4 (0xE1C)	0xE1C	H2C-ST デバッグ レジスタ 4
QDMA_H2C_FATAL_ERR_EN (0xE20)	0xE20	H2C-ST 致命的エラー イネーブル
QDMA_H2C_REQ_THROT (0xE24)	0xE24	
QDMA_H2C_ALN_DBG_REG0 (0xE28)	0xE28	

QDMA_H2C_ERR_STAT (0xE00)

表 221: QDMA_H2C_ERR_STAT (0xE00)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:5]	0	NA		予約
[4]	0	RW	dbe	H2C-ST データで訂正されたダブルビット エラー。
[3]	0	RW	sbe	H2C-ST データで訂正されたシングルビット エラー。

表 221: QDMA_H2C_ERR_STAT (0xE00) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[2]	0	RW	no_dma_dsc_err	no_dma ディスクリプターは、SOP または EOP がリセットされると受信されます。
[1]	0	RW	sdi_mrkr_req_mop_err	非 EOP ディスクリプターは、SOP または EOP がセットされると受信されます。
[0]	0	RW	zero_len_dsc_err	長さゼロのディスクリプターは、SOP または EOP がリセットされると受信されます。

これは、H2C エラーのエラー記録レジスタです。エラーが発生するとハードウェアによりレジスタに書き込まれます。必要に応じて、ソフトウェアで 1'b1 を書き込んでエラーをクリアできます。QDMA_H2C_ERR_MASK レジスタはエラーの記録には影響しません。

QDMA_H2C_ERR_MASK (0xE04)

表 222: QDMA_H2C_ERR_MASK (0xE04)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW	h2c_err_en_mask	H2C エラー イネーブル マスク

エラー アグリゲーターへ伝搬された H2C エラーがイネーブルになるよう、ソフトウェアによりこのビットがセットされます。

QDMA_H2C_FIRST_ERR_QID (0xE08)

表 223: QDMA_H2C_FIRST_ERR_QID (0xE08)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:20]		NA		予約
[19:16]	0	RO	err_type	4'b1111: NA 4'b0000: zero_len_dsc_err 4'b0001: wbi_mop_err 4'b0010: no_dma_dsc_err 4'b0011: sbe 4'b0100: dbे
[15:11]	0	NA		予約
[10:0]	0	RO	qid	最初の H2C エラーの QID

QDMA_H2C_DBG_REG0 (0xE0C)

表 224: QDMA_H2C_DBG_REG0 (0xE0C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	RO	num_dsc_rcvd	H2C-ST エンジンによって受信されたディスクリプターの数。

表 224: QDMA_H2C_DBG_REG0 (0xE0C) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[15:0]	0	RO	num_wrb_sent	ステータス書き込みをホストに送信するようディスクリプター エンジンにリクエストするために、H2C-ST エンジンから H2C ステータス書き込みエンジンに送信されたステータス書き込みパケットの数。

QDMA_H2C_DBG_REG1 (0xE10)

表 225: QDMA_H2C_DBG_REG1 (0xE10)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	RO	num_req_sent	H2C-ST エンジンから送信された PCIe リクエストの数。
[15:0]	0	RO	num_cmp_rcvd	H2C-ST エンジンによって受信された PCIe 応答の数。

QDMA_H2C_DBG_REG2 (0xE14)

表 226: QDMA_H2C_DBG_REG2 (0xE14)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	RO		予約
[15:0]	0	RO	num_err_dsc_rcvd	H2C-ST エンジンからエラーと共に受信したディスクリプターの数。

QDMA_H2C_DBG_REG3 (0xE18)

表 227: QDMA_H2C_DBG_REG3 (0xE18)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]	0	RO	debug	予約
[30]	1	RO	dsco_fifo_empty	H2C-ST ステータスの書き込み FIFO 空。
[29]	0	RO	dsco_fifo_full	H2C-ST ステータスの書き込み FIFO フル。
[28:26]	1	RO	cur_rc_state	H2C-ST データ FSM ステート。
[25:16]	0	RO	rdreq_lines	H2C-ST のリクエスト FSM で処理されているディスクリプターがフェッチする行数。
[15:6]	512	RO	rdata_lines_avail	H2C-ST データ バッファで可能な行数。
[5]	1	RO	pend_fifo_empty	H2C-ST 保留リクエスト FIFO 空。
[4]	0	RO	pend_fifo_full	H2C-ST 保留リクエスト FIFO フル。
[3:2]	01	RO	cur_rq_state	H2C-ST リクエスト FSM ステート。
[1]	0	RO	dsci_fifo_full	H2C-ST ディスクリプター入力 FIFO フル。
[0]	1	RO	dsci_fifo_empty	H2C-ST ディスクリプター入力 FIFO 空。

QDMA_H2C_DBG_REG4 (0xE1C)

表 228: QDMA_H2C_DBG_REG4 (0xE1C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	rdreq_addr	H2C-ST のリクエスト FSM で処理されているディスクリプターのアドレス。

QDMA_H2C_FATAL_ERR_EN (0xE20)

表 229: QDMA_H2C_FATAL_ERR_EN (0xE20)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:1]	0	RO		予約
[0]	0	RW	h2c_fatal_err_en	セットされると、H2C-ST データのダブル ビット エラーがユーザーに渡されます。リセットされると、そのエラーは無視されます。

QDMA_H2C_REQ_THROT (0xE24)

表 230: QDMA_H2C_REQ_THROT (0xE24)

ビット	デフォルト 値	アクセス タイプ	フィールド	説明
[31]	0	RW	req_throt_en_req	リクエスト基準のリクエスト スロットル イネーブル H2C ストリーム エンジンからの読み出しリクエストの未処理リクエスト基準のスロットルをイネーブルにします。
[30:26]	0	RO		予約
[25:17]	0x100	RW	req_thresh	リクエストしきい値 読み出しリクエスト スロットルを開始するため、H2C ストリーム エンジンで未処理にしておく必要のある読み出しリクエスト数。
[16]	0	RW	req_throt_en_data	データ基準のリクエスト スロットル イネーブル H2C ストリーム エンジンからの読み出しリクエストの未処理データ基準のスロットルをイネーブルにします。
[15:0]	0x800	RW	data_thresh	データしきい値 読み出しリクエスト スロットルを開始するため、H2C ストリーム エンジンで未処理にしておく必要のあるデータ量。

QDMA_H2C_ALN_DBG_REG0 (0xE28)

表 231: QDMA_H2C_ALN_DBG_REG0 (0xE28)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:16]	0	-		予約
[15:0]	0	RO	num_pkt_sent	H2C-ST データ アライナーにより送信されたパケットの数。

QDMA_TRQ_SEL_C2H_MM (0x1000)

表 232: QDMA_TRQ_SEL_C2H_MM (0x1000) レジスタ空間

レジスタ	アドレス	説明
C2H MM 制御	0x1004	チャンネル制御ビット。
	0x1008	チャンネル制御ビット W1S。
	0x100C	チャンネル制御ビット W1C。
C2H MM ステータス	0x1040	ステータス ビット。
	0x1044	ステータス クリア。
C2H 完了したディスクリプター カウント	0x1048	完了したディスクリプター カウント。
C2H MM エラー コード マスク (0x1054)	0x1054	エラー マスキング。
C2H MM エラー コード (0x1058)	0x1058	エラー コード。
C2H MM エラー情報 (0x105C)	0x105C	エラー情報。
C2H MM パフォーマンス モニター制御 (0x10C0)	0x10C0	パフォーマンス モニター制御。
C2H MM パフォーマンス モニター サイクル カウント 0 (0x10C4)	0x10C4	パフォーマンス モニター サイクル カウント [31:0]。
C2H MM パフォーマンス モニター データ カウント 1 (0x10C8)	0x10C8	パフォーマンス モニター サイクル カウント [41:32]。
C2H MM パフォーマンス モニター データ カウント 0 (0x10CC)	0x10CC	パフォーマンス モニター データ カウント [31:0]。
C2H MM パフォーマンス モニター データ カウント 1 (0x10D0)	0x10D0	パフォーマンス モニター データ カウント [41:32]。
C2H MM デバッグ (0x10E8)	0x10E8	デバッグ情報。

C2H MM 制御

表 233: C2H チャンネル制御 (0x1004)

ビット	デフォルト	アクセス タイプ	フィールド	説明
31:1				予約
0	1'b0	RW	run	run 1 にセットすると、SGDMA エンジンが開始します。転送を停止するには 0 にリセットします。エンジンがビジーの場合は、現在のディスクリプターを完了させます。

表 234: C2H チャンネル制御 (0x1008)

ビット	デフォルト	アクセス タイプ	フィールド	説明
		W1S		制御 ビットの説明は C2H チャンネル制御 (0x04) と同じです。

表 235: C2H チャンネル制御 (0x100C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
		W1C		制御 ビットの説明は C2H Channel Control (0x04) と同じです。

C2H MM ステータス

表 236: QDMA_C2H MM ステータス (0x1040)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:1]				予約
[0]			sts_bsy	ビジー セットされている場合は、エンジンが実行しています。

C2H 完了したディスクリプター カウント

表 237: C2H チャンネル完了したディスクリプター カウント (0x1048)

ビット	デフォルト	アクセス タイプ	フィールド	説明
31:0	32'h0	RO	c2h_compl_desc_count	c2h_compl_desc_count リストの各ディスクリプターを完了させた後、エンジンによりアップデートされた完了ディスクリプターの数。 制御レジスタの実行ビットの立ち上がりエッジで 0 にリセット (C2H Channel Control (0x1004) を参照)。

C2H MM エラー コード マスク (0x1054)

表 238: C2H MM エラー コード マスク (0x1054)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]	0	RW		予約
[30]	0	RW	wr_uc_ram	セットされている場合は、書き込みエラー、RAM 訂正不可能、エラー コードのログがイネーブルになります。
[29]	0	RW	wr_ur	セットされている場合は、書き込みエラー、サポートされないリクエスト、エラー コードのログがイネーブルになります。
[28]	0	RW	wr_flr	セットされている場合は、書き込みエラー、FLR リセット、エラー コードのログがイネーブルになります。
[27:2]	0	RW		予約
[1]	0	RW	rd_slv_err	セットされている場合は、読み出しスレーブ エラー コードのログがイネーブルになります。
[0]	0	RW	wr_slv_err	セットされている場合は、読み出しデコード エラー コードのログがイネーブルになります。

C2H MM エラー コード (0x1058)

表 239: C2H MM エラー コード (0x1058)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:28]	0	RW		予約
[27:12]	0	RW	cidx	ディスクリプターのコンシューマー インデックス。
[11:6]	0	NA	予約	予約
[5]	0	RW	rdwr	読み出しまたは書き込みエラー 0: 読み出しエラー 1: 書き込みエラー
[4:0]	0	RW	error_code	書き込みエラーの場合: 2: RAM 訂正不可能エラー 1: サポートされないリクエスト 0: ファンクション レベル リセット ほかのビットは予約されています。 読み出しエラーの場合: 1: スレーブ エラー 0: デコード エラー

C2H MM エラー情報 (0x105C)

表 240: C2H MM エラー情報 (0x105C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]	0	RW	valid	エラー情報およびエラー コードが有効です。
[30:11]	0	NA		予約
[10:0]	0	RW	qid	ディスクリプターのキュー ID。

C2H MM パフォーマンス モニター制御 (0x10C0)

表 241: C2H MM パフォーマンス モニター制御 (0x10C0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:4]				予約
[3]	0	RW	imm_start	カウンターをすぐに開始します。
[2]	0	RW	run_start	1 にセットするとカウンターの準備が整います。実行ビットがアサートされると、カウンターが開始します。
[1]	0	WO	imm_clear	カウンターをすぐに消去します。
[0]	0	RW	run_clear	実行ビットがアサートされると、カウンターが消去されます。

C2H MM パフォーマンス モニター サイクル カウント 0 (0x10C4)

表 242: C2H MM パフォーマンス モニター サイクル カウント 0 (0x10C4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	cyc_cnt[31:0]	サイクル カウント。 実行中、各クロックに対し 1 カウントずつ増加します。

C2H MM パフォーマンス モニター データ カウント 1 (0x10C8)

表 243: C2H MM パフォーマンス モニター データ カウント 1 (0x10C8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:10]				予約。
[9:0]	0	RO	cyc_cnt[41:32]	サイクル カウント。

C2H MM パフォーマンス モニター データ カウント 0 (0x10CC)

表 244: C2H MM パフォーマンス モニター データ カウント 0 (0x10CC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	dcnt[31:0]	データ カウント。 データ ビートが受信されるたびに 1 カウントずつ増加 します。

C2H MM パフォーマンス モニター データ カウント 1 (0x10D0)

表 245: C2H MM パフォーマンス モニター データ カウント 1(0x10D0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:10]				予約。
[9:0]	0	RO	dcnt[41:32]	データ カウント。

C2H MM デバッグ (0x10E8)

表 246: C2H MM デバッグ (0x10E8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:24]				予約。
[23:17]	0	RO	rrq_entries[6:0]	未処理リクエスト。
[16:7]	512	RO	dat_fifo_spc[9:0]	データ FIFO 空間。
[6]	0	RO	rd_stall	読み出しストール (停止)。
[5]	0	RO	rrq_fifo_fl	読み出し FIFO がフル。
[4]	0	RO	wr_stall	書き込みストール (停止)。

表 246: C2H MM デバッグ (0x10E8) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[3]	0	RO	wrq_fifo_fl	書き込み FIFO がフル。
[2]	0	RO	wbk_stall	ライトバック ストール (停止)。
[1]	1	RO	dsc_fifo_ep	ディスクリプター FIFO が空。
[0]	0	RO	dsc_fifo_fl	ディスクリプター FIFO がフル。

QDMA_TRQ_SEL_H2C_MM (0x1200)

表 247: QDMA_TRQ_SEL_H2C_MM (0x1200) レジスタ空間

レジスタ	アドレス	説明
H2C MM 制御	0x1204	チャンネル制御ビット。
	0x1208	チャンネル制御ビット W1S。
	0x120C	チャンネル制御ビット W1C。
H2C MM ステータス	0x1240	ステータス ビット。
H2C 完了したディスクリプター カウント	0x1248	完了したディスクリプター カウント。
H2C MM エラー コード イネーブル マスク (0x1254)	0x1254	エラー マスキング。
H2C MM エラー コード (0x1258)	0x1258	エラー コード。
H2C MM エラー情報 (0x125C)	0x125C	エラー情報。
H2C MM パフォーマンス モニター制御 (0x12C0)	0x12C0	パフォーマンス モニター制御。
H2C MM パフォーマンス モニター サイクル カウント 0 (0x12C4)	0x12C4	パフォーマンス モニター サイクル カウント [31:0]。
H2C MM パフォーマンス モニター サイクル カウント 1 (0x12C8)	0x12C8	パフォーマンス モニター サイクル カウント [41:32]。
H2C MM パフォーマンス モニター データ カウント 0 (0x12CC)	0x12C8	パフォーマンス モニター データ カウント [31:0]。
H2C MM Performance Monitor Data Count 1 (0x12D0)	0x12D0	パフォーマンス モニター データ カウント [41:32]。
H2C MM デバッグ (0x12E8)	0x12E8	デバッグ情報。
QDMA_H2C_MM_REQ_THROT (0x12EC)	0x12EC	

H2C MM 制御

表 248: H2C チャンネル制御 (0x04)

ビット	デフォルト	アクセス タイプ	フィールド	説明
31:1				予約
0	1'b0	RW	run	実行 1 にセットすると、SGDMA エンジンが開始します。転送を停止するには 0 にリセットします。エンジンがビジーの場合は、現在のディスクリプターを完了させます。

ie_* レジスタ ビットは割り込みイネーブルです。この条件が満たされていて、正しい割り込みマスクが設定されている場合、割り込みが生成されます。

表 249: H2C チャンネル制御 (0x1208)

ビット	デフォルト	アクセスタイプ	フィールド	説明
0		W1S		制御 ビットの説明は H2C チャンネル制御 (0x04) と同じです。

表 250: H2C チャンネル制御 (0x120C)

ビット	デフォルト	アクセスタイプ	フィールド	説明
0		W1C		制御 ビットの説明は H2C チャンネル制御 (0x04) と同じです。

H2C MM ステータス

表 251: H2C チャンネル ステータス (0x1240)

ビット	デフォルト	アクセスタイプ	フィールド	説明
311				予約
0	1'b0	RO	busy	ビジー SGDMA エンジンがビジーの場合にセットします。アイドルのときはゼロになります。

H2C 完了したディスクリプター カウント

表 252: H2C チャンネル完了したディスクリプター カウント (0x1248)

ビット	デフォルト	アクセスタイプ	フィールド	説明
31:0	32'h0	RO	h2c_compl_desc_count	リストの各ディスクリプターを完了させた後、エンジンによりアップデートされた完了ディスクリプターの数。 制御レジスタの実行ビットの立ち上がりエッジで 0 にリセット。 H2C Channel Control (0x1204) を参照してください。

H2C MM エラー コード イネーブル マスク (0x1254)

表 253: H2C MM エラー コード イネーブル マスク (0x1254)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:30]		RW		予約
[29]	0	RW	wr_slv_error	セットされている場合は、書き込みスレーブ エラー コードのログがイネーブルになります。
[28]	0	RW	wr_dec_err	セットされている場合は、書き込みデコード エラー コードのログがイネーブルになります。

表 253: H2C MM エラー コード イネーブル マスク (0x1254) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[27:23]	0	RW		予約
[22]	0	RW	rd_rq_dis_err	セットされている場合は、読み出しリクエスト ディスエーブル エラー コードのログがイネーブルになります。
[21:17]	0	RW		予約
[16]	0	RW	rd_dat_poison_err	セットされている場合は、読み出しデータ ポイズン エラー コードのログがイネーブルになります。
[15:9]	0	RW		予約
[8]	0	RW	rd_flr_err	セットされている場合は、読み出し FLR エラー コードのログがイネーブルになります。
[7:6]	0	RW		予約
[5]	0	RW	rd_hdr_adr_err	セットされている場合は、読み出し完了ヘッダー アドレス不一致エラー コードのログがイネーブルになります。
[4]	0	RW	rd_hdr_param_err	セットされている場合は、読み出し完了ヘッダー param 不一致エラー コードのログがイネーブルになります。
[3]	0	RW	rd_hdr_byte_err	セットされている場合は、読み出し完了ヘッダー バイトカウント不一致エラー コードのログがイネーブルになります。
[2]	0	RW	rd_ur_ca	セットされている場合は、読み出し完了のサポートされていないリクエスト、またはコンプリーター中止のエラー コードのログがイネーブルになります。
[1]	0	RW	rd_hrd_poison_err	セットされている場合は、読み出し完了ヘッダー ポイズンエラー コードのログがイネーブルになります。
[0]	0	RW		予約

H2C MM エラー コード (0x1258)

表 254: H2C MM エラー コード (0x1258)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:28]	0	RW		予約
[27:12]	0	RW	cidx	ディスクリプターのコンシューマー インデックス。
[11:6]	0	NA		予約
[5]	0	RW	rdwr	読み出しまたは書き込みエラー。 0: 読み出しエラー 1: 書き込みエラー

表 254: H2C MM エラー コード (0x1258) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[4:0]	0	RW	error_code	読み出しエラーの場合: 1: ヘッダー ポイズンド 2: サポートされていないリクエストまたはコンプリーター中止 3: ヘッダー バイト カウントの不一致 4: ヘッダー param の不一致 5: ヘッダー アドレスの不一致 8: ファンクション レベル リセット 16: データ ポイズンド 22: PCIe 読み出しがディスエーブル。 ほかのビットは予約されています。 書き込みエラーの場合: 1: スレーブ エラー 0: デコード エラー

H2C MM エラー情報 (0x125C)

表 255: H2C MM エラー情報 (0x125C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31]	0	RW	valid	エラー情報およびエラー コード ログが有効です。
[30:11]	0	NA		予約
[10:0]	0	RW	qid	ディスクリプターのキュー ID。

H2C MM パフォーマンス モニター制御 (0x12C0)

表 256: H2C MM パフォーマンス モニター制御 (0x12C0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:4]				予約。
[3]	0	RW	imm_start	カウンターをすぐに開始します。
[2]	0	RW	run_start	1 にセットするとカウンターの準備が整います。実行ビットがアサートされると、カウンターが開始します。
[1]	0	WO	imm_clear	カウンターをすぐに消去します。
[0]	0	RW	run_clear	実行ビットがアサートされると、カウンターが消去されます。

H2C MM パフォーマンス モニター サイクル カウント 0 (0x12C4)

表 257: H2C MM パフォーマンス モニター サイクル カウント 0 (0x12C4)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	cyc_cnt[31:0]	サイクル カウント。 実行中、各クロックに対し 1 カウントずつ増加します。

H2C MM パフォーマンス モニター サイクル カウント 1 (0x12C8)

表 258: H2C MM パフォーマンス モニター サイクル カウント 1 (0x12C8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:10]				予約。
[9:0]	0	RO	cyc_cnt[41:32]	サイクル カウント。

H2C MM パフォーマンス モニター データ カウント 0 (0x12CC)

表 259: H2C MM パフォーマンス モニター データ カウント 0 (0x12CC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RO	dcnt[31:0]	データ カウント。 データ ビートが受信されるたびに 1 カウントずつ増加 します。

H2C MM Performance Monitor Data Count 1(0x12D0)

表 260: H2C MM Performance Monitor Data Count 1(0x12D0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:10]				予約。
[9:0]	0	RO	dcnt[41:32]	データ カウント。

H2C MM デバッグ (0x12E8)

表 261: H2C MM デバッグ (0x12E8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:24]				予約。
[23:17]	0	RO	rrq_entries[6:0]	未処理リクエスト。
[16:7]	512	RO	dat_fifo_spc[9:0]	データ FIFO 空間。
[6]	0	RO	rd_stall	読み出しストール (停止)。
[5]	0	RO	rrq_fifo_fl	読み出し FIFO がフル。
[4]	0	RO	wr_stall	書き込みストール (停止)。

表 261: H2C MM デバッグ (0x12E8) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[3]	0	RO	wrq_fifo_fl	書き込み FIFO がフル。
[2]	0	RO	wbk_stall	ライトバック ストール (停止)。
[1]	1	RO	dsc_fifo_ep	ディスクリプター FIFO が空。
[0]	0	RO	dsc_fifo_fl	ディスクリプター FIFO がフル。

QDMA_H2C_MM_REQ_THROT (0x12EC)

表 262: QDMA_H2C_MM_REQ_THROT (0x12EC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:17]	0	RO		予約
[16]	0	RW	req_throt_en	データ基準のリクエスト スロットル イネーブル H2C MM エンジンからの読み出しリクエストの未処理 データ基準のスロットルをイネーブルにします。
[15:0]	0x8000	RW	data_thresh	データしきい値 読み出しリクエスト スロットルを開始するため、H2C MM エンジンで未処理にしておく必要のあるデータ量。

QDMA_PF_MAILBOX (0x2400)

表 263: QDMA_PF_MAILBOX (0x2400) レジスタ空間

レジスタ	アドレス	説明
ファンクション ステータス レジスタ (0x2400)	0x2400	ステータス ビット
ファンクション コマンド レジスタ (0x2404)	0x2404	コマンド レジスタ ビット
ファンクション 割り込みベクター レジスタ (0x2408)	0x2408	割り込みベクター レジスタ
ターゲット ファンクション レジスタ (0x240C)	0x240C	ターゲット ファンクション レジスタ
ファンクション 割り込みベクター レジスタ (0x2410)	0x2410	割り込み制御レジスタ
PF 肯定応答レジスタ (0x2420-0x243C)	0x2420-0x243C	PF 肯定応答
FLR 制御/ステータス レジスタ (0x2500)	0x2500	FLR 制御およびステータス
受信メッセージ メモリ (0x2C00-0x2C7C)	0x2C00-0x2C7C	入力メッセージ (128 バイト)
送信メッセージ メモリ (0x3000-0x307C)	0x3000-0x307C	出力メッセージ (128 バイト)

メールボックス アドレス指定

PF アドレス指定:

```
Addr = PF_Bar_offset + CSR_addr
```

VF アドレス指定:

$$\text{Addr} = \text{VF_Bar_offset} + \text{VF_Start_offset} + \text{VF_offset} + \text{CSR_addr}$$

ファンクション ステータス レジスタ (0x2400)

表 264: ファンクション ステータス レジスタ (0x2400)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:12]	0	NA		予約
11-4	0	RO	cur_src_fn	このフィールドは PF 専用です。 入力リクエスト キューの上のメッセージのソース ファンクション番号。
2	0	RO	ack_status	このフィールドは PF 専用です。 肯定応答ステータス レジスタの任意のビットがアサートされると、ステータス ビットがセットされます。
1	0	RO	o_msg_status	VF の場合: VF ドライバーがコマンド レジスタに msg_send を書き込むと、ステータス ビットがセットされます。関連付けられている PF ドライバーがこの VF に肯定応答を送信すると、ハードウェアがこのフィールドをクリアにします。o_msg_status がアサートされている間は、VF ドライバーは送信メールボックス メモリ (OMM) の内容をアップデートできません。OMM への無効な書き込みは破棄されます。オプションで、AXI4-Lite 応答チャンネルでエラーを通知できます。 PF の場合: ターゲット FN レジスタで指定されるターゲット FN のメッセージ ステータスを示します。PF ドライバーが msg_send コマンドを送信すると、ステータス ビットがセットされます。対応するファンクション ドライバーが msg_rcv を介して肯定応答を送信すると、ハードウェアがこのフィールドをクリアにします。 o_msg_status (target_fn_id) がアサートされている間は、PF ドライバーは送信メールボックス メモリ (OMM) の内容をアップデートできません。OMM への無効な書き込みは破棄されます。オプションで、AXI4-Lite 応答チャンネルでエラーを通知できます。
0	0	RO	i_msg_status	VF の場合: アサートされていると、VF の受信メールボックス メモリのメッセージが保留になります。VF ドライバーがコマンド レジスタに msg_rcv を書き込むと、このフィールドはクリアになります。 PF の場合: アサートされていると、受信メールボックス メモリのメッセージが保留になります。イベント キューが空の場合にのみこのフィールドがクリアになります。

ファンクション コマンド レジスタ (0x2404)

表 265: ファンクション コマンド レジスタ (0x2404)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:3]	0	NA		予約
2	0	RO		予約

表 265: ファンクション コマンド レジスタ (0x2404) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
1	0	RW	msg_rcv	VF の場合: 受信メールボックス メモリのメッセージを受信済みにします。関連付けられている PF の肯定応答ビットをハードウェアがアサートします。 PF の場合: target_fn により送信されたメッセージを受信済みにします。ハードウェアが PF の i_msg_status をリフレッシュし、target_fn の o_msg_status をクリアします。
0	0	RW	msg_send	VF の場合: 送信メールボックスの現在のメッセージを有効にします。 PF の場合: <ul style="list-style-type: none"> 現在の target_fn_id は VF にあります。target_fn_id のある VF の受信メールボックス メモリに PF がメッセージを書き込み終えます。ハードウェアがターゲット FN のステータス レジスタの i_msg_status フィールドを設定します。 現在の target_fn_id は PF にあります。送信メールボックス メモリに PF がメッセージを書き込み終えます。target_fn_id のある PF のイベント キューにハードウェアがメッセージを押し出します。

ファンクション割り込みベクター レジスタ (0x2408)

表 266: ファンクション割り込みベクター レジスタ (0x2408)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:5]	0	NA		予約
[4:0]	0	RW	int_vect	ドライバによって割り当てられた 5 ビット割り込みベクター。

ターゲット ファンクション レジスタ (0x240C)

表 267: ターゲット ファンクション レジスタ (0x0C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:8]	0	NA		予約
[7:0]	0	RW	target_fn_id	このフィールドは PF 専用です。 FN 番号は現在の操作がターゲットにしているファンクションです。

ファンクション割り込みベクター レジスタ (0x2410)

表 268: ファンクション割り込みベクター レジスタ (0x2410)

ビット	デフォルト	アクセス タイプ	フィールド	説明
31:1	0	NA		予約
0	0	RW	int_en	割り込みイネーブル。

PF 肯定応答レジスタ (0x2420-0x243C)

表 269: PF 肯定応答レジスタ (0x2420-0x243C)

レジスタ	アドレス	デフォルト	アクセス タイプ	フィールド	幅	説明
Ack0	0x2420	0	RW		32	FN 31 から 0 までの肯定応答
Ack1	0x2424	0	RW		32	FN 63 から 32 までの肯定応答
Ack2	0x2428	0	RW		32	FN 95 から 64 までの肯定応答
Ack3	0x242C	0	RW		32	FN 127 から 96 までの肯定応答
Ack4	0x2430	0	RW		32	FN 159 から 128 までの肯定応答
Ack5	0x2434	0	RW		32	FN 191 から 160 までの肯定応答
Ack6	0x2438	0	RW		32	FN 223 から 192 までの肯定応答
Ack7	0x243C	0	RW		32	FN 255 から 224 までの肯定応答

FLR 制御/ステータス レジスタ (0x2500)

表 270: FLR 制御/ステータス レジスタ (0x2500)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:1]	0	NA		予約
0	0	RW	Flr_status	関連付けられているファンクションのファンクションレベルリセット (FLR) を開始するため、ソフトウェアにより 1 が書き込まれます。このフィールドは、FLR プロセス中アサートされたままになります。FLR が完了すると、ハードウェアはこのフィールドをデアサートします。

受信メッセージ メモリ (0x2C00-0x2C7C)

表 271: 受信メッセージ メモリ (0x2C00-0x2C7C)

レジスタ	アドレス	デフォルト	アクセス タイプ	フィールド	幅	説明
i_msg_i	0x2C00 + i*4	0	RW		32	受信メッセージの第 i 番目のワード (0 ≤ i < 128)。

送信メッセージ メモリ (0x3000-0x307C)

表 272: 送信メッセージ メモリ (0x3000-0x307C)

レジスタ	アドレス	デフォルト	アクセス タイプ	フィールド	幅	説明
o_msg_i	0x3000 + i * 4	0	RW		32	送信メッセージの第 i 番目のワード (0 ≤ i < 128)。

QDMA_TRQ_MSIX (0x10000)

表 273: QDMA_TRQ_MSIX (0x10000)

バイト オフセット	ビット	デフォルト	アクセス タイプ	フィールド	説明
0x10000	[31:12]	0	NA		MSIX_Vector0_Address[31:0]MSIX_Vector0_Address[63:32] MSI-X ベクター 0 メッセージの低位アドレス。
0x10004	11-4	0	RO		MSI-X ベクター 0 メッセージの上位アドレス。
0x10008	2	0	RO	ack_status	MSIX_Vector0_Address[63:32]MSIX_Vector0_Data[31:0] MSI-X ベクター 0 メッセージのデータ。
0x1000C	1	0	RO	o_msg_status	MSIX_Vector0_Control[31:0] MSI-X ベクター 0 制御。 ビット位置: 31:1: 予約。 0: マスク。1 の場合、この MSI-X ベクターはメッセージの生成には使用されません。0 の場合、この MSI-X ベクターはメッセージの生成に使用されます。

注記: 上記の表は、1 つの MSI-X テーブル エントリを表しています。QDMA の場合、MSI-X テーブル エントリは 2000 個あります。

QDMA_TRQ_SEL_QUEUE_PF (0x18000)

表 274: QDMA_TRQ_SEL_QUEUE_PF (0x18000) レジスタ空間

レジスタ	アドレス	説明
QDMA_DMAP_SEL_INT_CIDX[2048] (0x18000)	0x18000-0x1CFF0	割り込みリング コンシューマー インデックス (CIDX)
QDMA_DMAP_SEL_H2C_DSC_PIDX[2048] (0x18004)	0x18004-0x1CFF4	H2C ディスクリプター プロデューサー インデックス (PIDX)
QDMA_DMAP_SEL_C2H_DSC_PIDX[2048] (0x18008)	0x18008-0x1CFF8	C2H ディスクリプター プロデューサー インデックス (PIDX)
QDMA_DMAP_SEL_CMPT_CIDX[2048] (0x1800C)	0x1800C-0x1CFFC	C2H 完了コンシューマー インデックス (CIDX)

2048 個のキューがあり、各キューには 4 つ以上のレジスタがあります。これらのレジスタは任意時に動的にアップデートできます。レジスタのこのセットは、キュー番号に従ってアクセスできます。

キュー番号は絶対値で Qnumber [0 から 2047 まで] になります。

割り込み CIDX アドレス = $0x18000 + Qnumber * 16$
 H2C PIDX アドレス = $0x18004 + Qnumber * 16$
 C2H PIDX アドレス = $0x18008 + Qnumber * 16$
 ライトバック CIDX アドレス = $0x1800C + Qnumber * 16$

キューが 0 の場合:

0x18000 は QDMA_DMAP_SEL_INT_CIDX に対応します。
 0c18004 は QDMA_DMAP_SEL_H2C_DSC_PIDX に対応します。
 0x18008 は QDMA_DMAP_SEL_C2H_DSC_PIDX に対応します。
 0x1800C は QDMA_DMAP_SEL_WRB_CIDX に対応します。

キューが 1 の場合:

0x18010 は QDMA_DMAP_SEL_INT_CIDX に対応します。
 0c18014 は QDMA_DMAP_SEL_H2C_DSC_PIDX に対応します。
 0x18018 は QDMA_DMAP_SEL_C2H_DSC_PIDX に対応します。
 0x1801C は QDMA_DMAP_SEL_WRB_CIDX に対応します。

キューが 2 の場合:

0x18020 は QDMA_DMAP_SEL_INT_CIDX に対応します。
 0c18024 は QDMA_DMAP_SEL_H2C_DSC_PIDX に対応します。
 0x18028 は QDMA_DMAP_SEL_C2H_DSC_PIDX に対応します。
 0x1802C は QDMA_DMAP_SEL_WRB_CIDX に対応します。

QDMA_DMAP_SEL_INT_CIDX[2048] (0x18000)

表 275: QDMA_DMAP_SEL_INT_CIDX[2048] (0x18000)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:24]	0	NA		予約
[23:16]	0	RW	ring_idx	割り込みアグリゲーション リングのリング インデックス
[15:0]	0	RW	sw_cdix	ソフトウェア コンシューマー インデックス (CIDX)

QDMA_DMAP_SEL_H2C_DSC_PIDX[2048] (0x18004)

表 276: QDMA_DMAP_SEL_H2C_DSC_PIDX[2048] (0x18004)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:17]	0	NA		予約
[16]	0	RW	irq_en	割り込みアーム、割り込みイネーブル
[15:0]	0	RW	h2c_pidx	H2C プロデューサー インデックス

QDMA_DMAP_SEL_C2H_DSC_PIDX[2048] (0x18008)

表 277: QDMA_DMAP_SEL_C2H_DSC_PIDX[2048] (0x18008)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:17]	0	NA		予約
[16]	0	RW	irq_en	割り込みアーム、割り込みイネーブル
[15:0]	0	RW	c2h_pidx	C2H プロデューサー インデックス

QDMA_DMAP_SEL_CMPT_CIDX[2048] (0x1800C)

表 278: QDMA_DMAP_SEL_CMPT_CIDX[2048] (0x1800C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:29]	0	NA		予約
[28]	0	RW	irq_en_wrb	CMPT の割り込みをイネーブル
[27]	0	RW	en_sts_desc_wrb	CMPT のステータス ディスクリプターをイネーブル
[26:24]	0	RW	trigger_mode	割り込みおよびステータス ディスクリプターのトリガーモード: 0x0: 無効 0x1: すべて 0x2: ユーザー カウント 0x3: ユーザー 0x4: ユーザー タイマー 0x5: ユーザー タイマー カウント
[23:20]	0	RW	c2h_timer_cnt_index	QDMA_C2H_TIMER_CNT へのインデックス
[19:16]	0	RW	c2h_count_threshhold	QDMA_C2H_CNT_TH へのインデックス
[15:0]	0	RW	wrb_cidx	CMPT コンシューマー インデックス (CIDX)

QDMA VF アドレス レジスタ空間

表 279: QDMA VF アドレス レジスタ空間

ターゲット名	ベース (16 進数)	バイト サイズ (10 進数)	注記
QDMA_TRQ_MSIX_VF (0x0000)	00000000	4096	32 個の MSIX ベクターおよび PBA
QDMA_VF_MAILBOX (0x1000)	00001000	8192	メールボックス アドレス空間
QDMA_TRQ_SEL_QUEUE_VF (0x3000)	00003000	32768	VF ダイレクト QCSR (キューごとに 16B、ファンクションごとに最高 2048 個のキュー)

QDMA_TRQ_MSIX_VF (0x0000)

VF ファンクションは、そのファンクションのオフセット (0x0000) の MSIX テーブルにアクセスできます。このレジスタの説明は、[QDMA_TRQ_MSIX \(0x10000\)](#)と同じです。

QDMA_VF_MAILBOX (0x1000)

表 280: QDMA_TRQ_SEL_IND (0x00800) レジスタ空間

レジスタ (アドレス)	アドレス	説明
ファンクション ステータス レジスタ (0x1000)	0x1000	ステータス レジスタ ビット
ファンクション コマンド レジスタ (0x1004)	0x1004	コマンド レジスタ ビット
ファンクション 割り込みベクター レジスタ (0x1008)	0x1008	割り込みベクター レジスタ
ターゲット ファンクション レジスタ (0x100C)	0x100C	ターゲット ファンクション レジスタ
ファンクション 割り込み制御レジスタ (0x1010)	0x1010	割り込み制御レジスタ
受信メッセージ メモリ (0x1800-0x187C)	0x1800-0x187C	入力メッセージ (128 バイト)
送信メッセージ メモリ (0x1C00-0x1C7C)	0x1C00-0x1C7C	出力メッセージ (128 バイト)

ファンクション ステータス レジスタ (0x1000)

表 281: ファンクション ステータス レジスタ (0x1000)

ビットインデックス	デフォルト	アクセスタイプ	フィールド	説明
[31:12]	0	NA		予約
11-4	0	RO	cur_src_fn	このフィールドは PF 専用です。 入力リクエスト キューの上のメッセージのソース ファンクション番号。
2	0	RO	ack_status	このフィールドは PF 専用です。 肯定応答ステータス レジスタの任意のビットがアサートされると、ステータス ビットがセットされます。
1	0	RO	o_msg_status	VF の場合: VF ドライバーがコマンド レジスタに msg_send を書き込むと、ステータス ビットがセットされます。関連付けられている PF ドライバーがこの VF に肯定応答を送信すると、ハードウェアがこのフィールドをクリアにします。o_msg_status がアサートされている間は、VF ドライバーは送信メールボックス メモリ (OMM) の内容をアップデートできません。OMM への無効な書き込みは破棄されます。オプションで、AXI4-Lite 応答チャンネルでエラーを通知できます。 PF の場合: ターゲット FN レジスタで指定されるターゲット FN のメッセージ ステータスを示します。PF ドライバーが msg_send コマンドを送信すると、ステータス ビットがセットされます。対応するファンクション ドライバーが msg_rcv を介して肯定応答を送信すると、ハードウェアがこのフィールドをクリアにします。 o_msg_status (target_fn_id) がアサートされている間は、PF ドライバーは送信メールボックス メモリ (OMM) の内容をアップデートできません。OMM への無効な書き込みは破棄されます。オプションで、AXI4-Lite 応答チャンネルでエラーを通知できます。

表 281: ファンクション ステータス レジスタ (0x1000) (続き)

ビットインデックス	デフォルト	アクセスタイプ	フィールド	説明
0	0	RO	i_msg_status	VF の場合: アサートされていると、VF の受信メールボックス メモリのメッセージが保留になります。VF ドライバーがコマンド レジスタに msg_rcv を書き込むと、このフィールドはクリアになります。 PF の場合: アサートされていると、受信メールボックス メモリのメッセージが保留になります。イベント キューが空の場合にのみこのフィールドがクリアになります。

ファンクション コマンド レジスタ (0x1004)

表 282: ファンクション コマンド レジスタ (0x1004)

ビットインデックス	デフォルト	アクセスタイプ	フィールド	説明
[31:3]	0	NA		予約
2	0	RO		予約
1	0	RW	msg_rcv	VF の場合: 受信メールボックス メモリのメッセージを受信済みにします。関連付けられている PF の肯定応答ビットをハードウェアがアサートします。 PF の場合: target_fn により送信されたメッセージを受信済みにします。ハードウェアが PF の i_msg_status をリフレッシュし、target_fn の o_msg_status をクリアします。
0	0	RW	msg_send	VF の場合: 送信メールボックスの現在のメッセージを有効にします。 PF の場合: 現在の target_fn_id は VF にあります。target_fn_id のある VF の受信メールボックス メモリに PF がメッセージを書き込み終わります。ハードウェアがターゲット FN のステータス レジスタの i_msg_status フィールドを設定します。 現在の target_fn_id は PF にあります。送信メールボックス メモリに PF がメッセージを書き込み終わります。target_fn_id のある PF のイベント キューにハードウェアがメッセージを押し出します。

ファンクション 割り込みベクター レジスタ (0x1008)

表 283: ファンクション 割り込みベクター レジスタ (0x1008)

ビットインデックス	デフォルト	アクセスタイプ	フィールド	説明
[31:5]	0	NA		予約
[4:0]	0	RW	int_vect	ドライバー ソフトウェアによって割り当てられた 5 ビット割り込みベクター。

ターゲット ファンクション レジスタ (0x100C)

表 284: ターゲット ファンクション レジスタ (0x100C)

ビットインデックス	デフォルト	アクセスタイプ	フィールド	説明
[31:8]	0	NA		予約
[7:0]	0	RW	target_fn_id	このフィールドは PF 専用です。 FN 番号は現在の操作がターゲットにしているファンクションです。

ファンクション割り込み制御レジスタ (0x1010)

表 285: ファンクション割り込み制御レジスタ (0x1010)

ビットインデックス	デフォルト	アクセスタイプ	フィールド	説明
[31:1]	0	NA		予約
0	0	RW	int_en	割り込みイネーブル。

受信メッセージ メモリ (0x1800-0x187C)

表 286: 受信メッセージ メモリ (0x1800-0x187C)

レジスタ	アドレス	デフォルト	アクセスタイプ	フィールド	幅	説明
i_msg_i	0x1800 + i*4	0	RW		32	受信メッセージの第 i 番目のワード (i < 128)。

送信メッセージ メモリ (0x1C00-0x1C7C)

表 287: 送信メッセージ メモリ (0x1C00-0x1C7C)

レジスタ	アドレス	デフォルト	アクセスタイプ	フィールド	幅	説明
o_msg_i	0x1C00 + i*4	0	RW		32	送信メッセージの第 i 番目のワード (i < 128)。

QDMA_TRQ_SEL_QUEUE_VF (0x3000)

VF ファンクションは、オフセット (0x3000) でキューごとにダイレクト アップデート レジスタにアクセスできます。このレジスタの説明は、[QDMA_TRQ_SEL_QUEUE_PF \(0x18000\)](#)と同じです。

このセットのレジスタはキュー番号に基づいてアクセス可能です。キュー番号は絶対番号 (Qnumber) で 0 から 2047 までです。

割り込み CIDX アドレス = 0x3000 + Qnumber*16

H2C PIDX アドレス = 0x3004 + Qnumber*16

C2H PIDX アドレス = 0x3008 + Qnumber*16

完了 CIDX アドレス = 0x300C + Qnumber*16

キュー 0 の場合:

0x3000 は QDMA_DMAP_SEL_INT_CIDX に対応
 0x3004 は QDMA_DMAP_SEL_H2C_DSC_PIDX に対応
 0x3008 は QDMA_DMAP_SEL_C2H_DSC_PIDX に対応
 0x300C は QDMA_DMAP_SEL_WRB_CIDX に対応

キュー 1 の場合:

0x3010 は QDMA_DMAP_SEL_INT_CIDX に対応
 0x3014 は QDMA_DMAP_SEL_H2C_DSC_PIDX に対応
 0x3018 は QDMA_DMAP_SEL_C2H_DSC_PIDX に対応
 0x301C は QDMA_DMAP_SEL_WRB_CIDX に対応

AXI4-Lite スレーブ レジスタ空間

AXI4-Lite スレーブ インターフェイスを介して、ユーザーはブリッジ レジスタおよび DMA レジスタ空間にアクセスできます。

- ブリッジ レジスタ: AXI4-Lite スレーブ アドレス ビット [28] が 0 にセットされている場合は、ブリッジおよび PCIe コンフィギュレーション レジスタにアクセスできます。ブリッジおよび PCIe コンフィギュレーション レジスタには、AXI4-Lite スレーブ インターフェイスがイネーブルのときにのみアクセスできます。
- DMA レジスタ: AXI4-Lite スレーブ アドレス ビット [28] が 1 にセットされている場合は、DMA レジスタにアクセスできます。DMA レジスタの詳細は、次を参照してください。
 - [QDMA PF アドレス レジスタ空間](#)
 - [QDMA VF アドレス レジスタ空間](#)

ブリッジ レジスタ空間

ブリッジ レジスタ アドレスは 0xE00 で開始します。0x00 ~ 0xE00 のアドレスは、PCIe コアのコンフィギュレーション レジスタ空間に割り当てられます。

ブリッジ レジスタ メモリ レジスタ

表 288: ブリッジ レジスタ メモリ レジスタ

アクセス	オフセット	内容	位置
RO	0xE00	VSEC 機能	AXI ブリッジ定義のメモリ マップド レジスタ空間
RO	0xE04	VSEC ヘッダー	
RO	0xE08	ブリッジ情報	
R/W	0xE0C	ブリッジ ステータス/制御	
R/W	0xE10	割り込みデコード	
R/W	0xE14	割り込みマスク	
RO	0xE18	バス位置	
RO	0xE1C	物理サイド インターフェイス (PHY) のステータス/制御	
RO	0xE20 - 0xE34	予約された空間	
R/W	0xE38	割り込みデコード 2	
R/W	0xE3C	割り込みマスク 2	
RO	0xE40	コンフィギュレーション制御	
RO	0xE44	スレーブ エラー AID	
RO	0xE48 - 0xE54	予約された空間	
R/W	0xE58	ユーザー IRQ リクエスト	
R/W	0xE5C	ユーザー IRQ 肯定応答	
R/W	0xE60	PCIe TX メッセージ制御	
R/W	0xE64	PCIe TX メッセージ ヘッダー下位ビット	
R/W	0xE68	PCIe TX メッセージ ヘッダー高位ビット	
R/W	0xE6C	PCIe TX メッセージ データ FIFO	
R/W	0xE70	PCIe RX メッセージ制御およびステータス	
R/W	0xE74	PCIe RX メッセージ FIFO	
RO	0xE78	マスター保留カウンタ	
R/W	0xE7C	PCIe TX MSI/MSI-X 制御およびステータス	
RO	0xE80 - 0xED0	予約された空間	
RO	0xED8	VSEC 機能 2	
RO	0xEDC	VSEC ヘッダー 2	
R/W	0xEE0-0xF0C	AXI ベース アドレス変換コンフィギュレーション レジスタ	

VSEC 機能レジスタ (0xE00)

コアのメモリ空間が基本統合ブロック PCIe コンフィギュレーション空間の一部として表示されるようにするレジスタです。VSEC は、最後の改善された機能構造のすぐ後に挿入されます。

表 289: VSEC 機能レジスタ (0xE00)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[15:0]	0xB	RO	cap_id	この改善機能をベンダー別機能として特定する PCIe 定義の ID です。
[19:16]	0x1	RO	cap_ver	この機能構造のバージョンを示します。
[31:20]	0xED8	RO	nxt_cap_offset	次の機能のオフセットを示します。

VSEC ヘッダー レジスタ (0xE04)

VSEC 構造のレイアウトおよび内容、リビジョンおよび長さの ID (ベンダー内) を提供するレジスタです。VSEC ヘッダー レジスタは、VSEC ヘッダー レジスタ (オフセット 0xE08) の直後に開始するメインのブリッジ レジスタを含む AXI ブリッジの一部です。

表 290: VSEC ヘッダー レジスタ (0xE04)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[15:0]	0x0	RO	vsec_id	VSEC 構造の性質およびフォーマットを識別する ID です。
[19:16]	0x3	RO	vsec_rev	この機能構造のバージョンを示します。
[31:20]	0x80	RO	vsec_length	VSEC 機能構造全体の長さ (VSEC 機能レジスタを含む) をバイトで示します。

ブリッジ情報レジスタ (0xE08)

PCIe AXI ブリッジに関して一般的なコンフィギュレーション情報を提供するレジスタです。このレジスタの情報はスタティックで、操作中に変化しません。

表 291: ブリッジ情報レジスタ (0xE08)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[0]	0	RO	gen2_cap	セットされている場合、基本統合ブロックで PCIe Gen2 スピードがサポートされます。
[1]	0	RO	rootport_present	このビットがセットされていると、基本統合ブロックがルートポートであることを示します。セットされている場合、ルートポートレジスタがこのインターフェイスにあります。
[2]	0	RO	upconfig_cap	このビットがセットされていると、基本統合ブロックがアップコンフィギュレーション対応であることを示します。
[3]	0	RO	gen3_cap	セットされている場合、基本統合ブロックで PCIe Gen3 スピードがサポートされます。
[4]	0	RO	gen4_cap	セットされている場合、基本統合ブロックで PCIe Gen4 スピードがサポートされます。
[31:5]	0	RO	予約	予約

ブリッジ制御およびステータス レジスタ (0xE0C)

PCIe AXI ブリッジの現在のステータスに関する情報を提供するレジスタです。

表 292: ブリッジ制御およびステータス レジスタ (0xE0C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[7:0]	0	RO		予約
[8]	0	RW	g1b_int_dis	セットされている場合、割り込みラインはアサートされません。割り込みデコード レジスタのビットがセットされるのは回避されません。
[9]	0	RW	cfg_space_en	セットされている場合は、コンフィギュレーション リクエストに応じて一般の完了 (CRS ではない) が PCIe で生成されます。セットされていない場合は、PCIe により CRS が返されます。この制御ビットは、cfg_space_delay_en 属性が 1 に設定されている場合にのみ有効です。 (エンドポイントのみ対象)
[31:10]	0	RO		予約

割り込みデコード レジスタ (0xE10)

割り込みライン [0] がアサートされている原因を判断し、割り込みをクリアにする方法を決定するレジスタです。割り込みをクリアするには、エラー FIFO をまず空にしなければならない訂正可能、非致命的、致命的ビット以外の任意ビットに 1'b1 を書き込みます。

表 293: 割り込みデコード レジスタ (0xE10)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[0]	0	RW1C	link_down	PCI Express リンクでリンク アップが失われたことを示します。以前にリンク アップがなかった場合はアサートされません。
[1]	0	RW1C	sw_ctrl_int	DMA レジスタでソフトウェア割り込み (ホストなどから) がセットされたことを示します。
[2]	0	RW1C	flr_is_hit	スレーブ トランザクションが FLR をヒットしていることを示します。
[3]	0	RW1C	hot_reset	ホットリセットが検出されたことを示します。
[17:4]	0	RO		予約
[18]	0	RW1C	vdm_rcvd	VDM メッセージが受信されたことを示します。このメッセージは RX_MFIFO_READ レジスタから読み出される必要があります。
[19]	0	RW1C	pme_turn_off_rcvd	pme_turn_off メッセージが受信されたことを示します。 (エンドポイントのみ対象)
[20]	0	RW1C	slv_ur	完了 TLP が 3'b001 (サポートされていないリクエスト) のステータスで受信されたことを示します。
[21]	0	RW1C	slv_tz_violation	ブリッジ スレーブ ポートで TrustZone 違反が検出されたことを示します。スレーブ エラー AID レジスタで違反のある AXI リクエスト ID が記録されます。
[22]	0	RW1C	slv_cpl_timeout	C_COMP_TIMEOUT パラメータで選択されている時間内に、PCIe の読み出しリクエストの完了 TLP が返されなかったことを示します。
[23]	0	RW1C	slv_err_poison	エラー ポイズン (EP) ビットが完了 TLP でセットされたことを示します。
[24]	0	RW1C	slv_ca	完了 TLP が 3'b100 (コンプリーター中止) のステータスで受信されたことを示します。

表 293: 割り込みデコード レジスタ (0xE10) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[25]	0	RW1C	slv_illegal_burst	INCR 以外のバースト タイプが AXI マスターによってリクエストされたことを示します。
[26]	0	RW1C	mst_decerr	デコード エラー (DECERR) 応答が受信されたことを示します。
[27]	0	RW1C	mst_slvrr	スレーブ エラー (SLVERR) 応答が受信されたことを示します。
[28]	0	RW1C	slv_pcie_timeout	PCIe タイムアウト完了が受信されたことを示します。
[29]	0	RW1C	ecc_parity_err_rcvd	RAM ECC エラーまたはパリティ エラーが受信されたことを示します。ソースは割り込みデコード 2 レジスタのビット [9:0] から読み出される必要があります。
[30]	0	RW1C	pcie_local_err_rcvd	PCIe ローカル エラーが受信されたことを示します。エラー コードは、割り込みデコード 2 レジスタのビット [24:20] から読み出される必要があります。
[31]	0	RW1C	dma_int_rcvd	DMA 割り込みが受信されたことを示します。ユーザーアプリケーションは、第 2 レベルの DMA レジスタをチェックする必要があります。 (DMA がイネーブルの場合のみチェックが必要です。) XDMA の場合: IRQ ブロック ユーザー割り込みリクエスト レジスタ (0x2040)。 IRQ ブロック エンジン割り込みリクエスト レジスタ (0x2044)。

割り込みマスク レジスタ (0xE14)

各割り込みソースが割り込みライン [0] をアサートさせるかどうかを制御するためのレジスタです。任意位置で 1 になると、割り込みソースで割り込みラインがアサートされます。このレジスタはすべて 0 に初期化します。このため、デフォルトでは、どのイベントに対しても割り込みは生成されません。

表 294: 割り込みマスク レジスタ (0xE14)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[0]	0	RW	link_down	ビットがセットされると、リンク ダウン イベントの割り込みがイネーブルになります。
[1]	0	RW	sw_ctrl_int	ビットがセットされると、ソフトウェア割り込み (ホストなどからの) がイネーブルになります。
[2]	0	RW	flr_is_hit	ビットがセットされると、FLR イベントをヒットしているスレーブ トランザクションの割り込みがイネーブルになります。
[3]	0	RW	hot_reset	ビットがセットされると、ホット リセット イベントの割り込みがイネーブルになります。 (EP 設定に対してのみ書き込み可能、それ以外の場合は 0)
[17:4]	0	RO		予約
[18]	0	RW	vdm_rcvd	ビットがセットされると、VDM イベントの割り込みがイネーブルになります。

表 294: 割り込みマスク レジスタ (0xE14) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[19]	0	RW	pme_turn_off_rcvd	ビットがセットされると、PME_Turn_Off イベントの割り込みがイネーブルになります。 (EP 設定に対してのみ書き込み可能、それ以外の場合は 0)
[20]	0	RW	slv_ur	ビットがセットされると、スレーブのサポートされないリクエスト割り込みがイネーブルになります。
[21]	0	RW	slv_tz_violation	ビットがセットされると、スレーブ TrustZone 違反割り込みがイネーブルになります。
[22]	0	RW	slv_cpl_timeout	ビットがセットされると、スレーブ完了タイムアウト割り込みがイネーブルになります。
[23]	0	RW	slv_err_poison	ビットがセットされると、スレーブ エラー ポイズン割り込みがイネーブルになります。
[24]	0	RW	slv_ca	ビットがセットされると、スレーブ コンプリーター中止割り込みがイネーブルになります。
[25]	0	RW	slv_illegal_burst	ビットがセットされると、スレーブ無効バースト割り込みがイネーブルになります。
[26]	0	RW	mst_decerr	ビットがセットされると、マスター DECERR 割り込みがイネーブルになります。
[27]	0	RW	mst_slvrr	ビットがセットされると、マスター SLVERR 割り込みがイネーブルになります。
[28]	0	RW	slv_pcie_timeout	ビットがセットされると、スレーブ PCIe タイムアウト割り込みがイネーブルになります。
[29]	0	RW	ecc_parity_err	ビットがセットされると、RAM ECC/パリティ エラー割り込みがイネーブルになります。
[30]	0	RW	pcie_local_err	ビットがセットされると、PCIe ローカル エラー割り込みがイネーブルになります。
[31]	0	RW	dma_int	ビットがセットされると、DMA 割り込みがイネーブルになります。

バス ロケーション レジスタ (0xE18)

バス、デバイス、ファンクション番号、PCIe ポートのポート番号をレポートするレジスタです。

表 295: バス ロケーション レジスタ (0xE18)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[2:0]	0	RO	func_num	PCIe のポートのファンクション番号。0 に固定されています。
[7:3]	0	RO	dev_num	PCIe のポートのデバイス番号。エンドポイントの場合、このレジスタは RO で、ルート ポートによって設定されます。
[15:8]	0	RO	bus_num	PCIe のポートのバス番号。エンドポイントの場合、このレジスタは RO で、外部ルート ポートによって設定されます。
[23:16]	0	RW	port_num	リンク機能レジスタのポート番号フィールドを設定します。 EP: 常に 0 を読み出し、書き込み不可能です。 RP: 書き込み可能です。

表 295: バス ロケーション レジスタ (0xE18) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:24]	0	RO		予約

PHY 制御およびステータス レジスタ (0xE1C)

現在の PHY ステートのステータスを提供し、および PCIe コア の速度およびレート切り替えを制御するレジスタです。

表 296: PHY 制御およびステータス レジスタ (0xE1C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[0]	0	RO	link_is_gen2	現在のリンク レートが 5.0 Gt/s かどうかをレポートします。
[2:1]	0	RO	link_width	現在のリンク幅をレポートします。00b= x1、01b= x2、10b= x4、11b= x8。
[8:3]	0	RO	ltssm_state	現在のリンク トレーニングおよびステータス ステートマシン (LTSSM) ステートをレポートします。エンコードは基本になっている統合ブロックによって決まります。
[10:9]	0	RO		予約
[11]	0	RO	link_up	現在の PHY リンク アップ ステートをレポートします。 1b: リンク アップ 0b: リンク ダウン リンク アップはコアがリンク アップ ステータスに達したことを示します。つまり、LTSSM が L0 ステートにあり、コアはデータパケットを送受信できるという意味です。
[12]	0	RO	link_is_gen3	現在のリンク レートが 8.0 Gt/s かどうかをレポートします。
[13]	0	RO	link_width_is_x16	現在のリンク幅をレポートします。0b = ビット [2:1] を参照してください。1b = x16
[14]	0	RO	link_is_gen4	現在のリンク レートが 16.0 Gt/s かどうかをレポートします。
[31:15]	0	RO		予約

割り込みデコード 2 レジスタ (0xE38)

割り込みライン [0] がアサートされている原因を判断し、割り込みをクリアにする方法を決定するレジスタです。割り込みをクリアするには、任意ビットに 1'b1 を書き込みます。

表 297: 割り込みデコード 2 レジスタ (0xE38)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[0]	0	RW1C	slv_axis_par_err	AXIST RC インターフェイスでパリティ エラーが検出されたことを示します。
[1]	0	RW1C	slv_r_ecc_err	スレーブ読み出し RAM で ECC 訂正不可能エラーが検出されたことを示します。

表 297: 割り込みデコード 2 レジスタ (0xE38) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[2]	0	RW1C	slv_w_ecc_err	スレーブ書き込み RAM で ECC 訂正不可能エラーが検出されたことを示します。
[3]	0	RW1C	mst_axis_par_err	AXIST CQ インターフェイスでパリティ エラーが検出されたことを示します。
[4]	0	RW1C	mst_r_ecc_err	マスター読み出し RAM で ECC 訂正不可能エラーが検出されたことを示します。
[5]	0	RW1C	mst_w_ecc_err	マスター書き込み RAM で ECC 訂正不可能エラーが検出されたことを示します。
[6]	0	RW1C	slv_r_ecc_cerr	スレーブ読み出し RAM で ECC 訂正可能エラーが検出されたことを示します。
[7]	0	RW1C	slv_w_ecc_cerr	スレーブ書き込み RAM で ECC 訂正可能エラーが検出されたことを示します。
[8]	0	RW1C	mst_r_ecc_cerr	マスター読み出し RAM で ECC 訂正可能エラーが検出されたことを示します。
[9]	0	RW1C	mst_w_ecc_cerr	マスター書き込み RAM で ECC 訂正可能エラーが検出されたことを示します。
[10]	0	RW1C	slv_lite_par_err	スレーブ LITE 書き込みインターフェイスでパリティ エラーが検出されたことを示します。
[19:11]	0	RO		予約

表 297: 割り込みデコード 2 レジスタ (0xE38) (続き)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[24:20]	0	RW1C	pcie_local_err_code	PCIe ローカル エラーが受信されたことを示します。最初のエラー コードは、クリアになるまで保持されます。 00000b - 予約 00001b - 物理層エラー検出 00010b - リンク再生タイムアウト 00011b - リンク再生繰り越し 00100b - リンク無効 TLP の受信 00101b - リンク無効 DLLP の受信 00110b - リンク プロトコル エラー 00111b - 再生バッファ RAM 訂正可能 ECC エラー 01000b - 再生バッファ RAM 訂正不可能 ECC エラー 01001b - 受信ポストッド リクエスト RAM 訂正可能 ECC エラー 01010b - 受信ポストッド リクエスト RAM 訂正不可能 ECC エラー 01011b - 受信完了 RAM 訂正可能 ECC エラー 01100b - 受信完了 RAM 訂正不可能 ECC エラー 01101b - 受信ポストッド バッファ オーバーフロー エラー 01110b - 受信ノンポストッド バッファ オーバーフロー エラー 01111b - 受信完了バッファ オーバーフロー エラー 10000b - フロー制御プロトコル エラー 10001b - 送信パリティ エラーの検出 10010b - 予期せぬ完了の受信 10011b - 完了タイムアウトの検出 10100b - AXI4ST RQ INTFC パケットの破棄 10101b - AXI4ST CC INTFC パケットの破棄 10110b - AXI4ST CQ ポイズン破棄 10111b - ユーザー検出の内部訂正可能エラー 11000b - ユーザー検出の内部訂正不可能エラー 11001b - TPH RAM の内部訂正可能エラー 11010b - TPH RAM の内部:訂正不可能エラー 11011b - MSIX RAM の内部訂正可能エラー 11100b - MSIX RAM の内部訂正可能エラー 11101b - DVSEC RAM の内部訂正可能エラー 11110b - DVSEC RAM の内部:訂正不可能エラー 11111b - 予約
[31:25]	0	RO		予約

割り込みマスク 2 レジスタ (0xE3C)

各割り込みソースが割り込みライン [0] をアサートさせるかどうかを制御するためのレジスタです。任意位置で 1 になると、割り込みソースで割り込みラインがアサートされます。このレジスタはすべて 0 に初期化します。このため、デフォルトでは、どのイベントに対しても割り込みは生成されません。

表 298: 割り込みマスク 2 レジスタ (0xE3C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[0]	0	RW	slv_axis_par_err	ビットがセットされると、AXIST RC パリティ エラー イベントに対し、割り込みがイネーブルになります。
[1]	0	RW	slv_r_ecc_err	ビットがセットされると、スレーブ読み出し RAM ECC 訂正不可能エラー イベントに対し、割り込みがイネーブルになります。
[2]	0	RW	slv_w_ecc_err	ビットがセットされると、スレーブ書き込み RAM ECC 訂正不可能エラー イベントに対し、割り込みがイネーブルになります。
[3]	0	RW	mst_axis_par_err	ビットがセットされると、AXIST CQ パリティ エラー イベントに対し、割り込みがイネーブルになります。
[4]	0	RW	mst_r_ecc_err	ビットがセットされると、マスター読み出し RAM ECC 訂正不可能エラー イベントに対し、割り込みがイネーブルになります。
[5]	0	RW	mst_w_ecc_err	ビットがセットされると、マスター書き込み RAM ECC 訂正不可能エラー イベントに対し、割り込みがイネーブルになります。
[6]	0	RW	slv_r_ecc_cerr	ビットがセットされると、スレーブ読み出し RAM ECC 訂正可能エラー イベントに対し、割り込みがイネーブルになります。
[7]	0	RW	slv_w_ecc_cerr	ビットがセットされると、スレーブ書き込み RAM ECC 訂正可能エラー イベントに対し、割り込みがイネーブルになります。
[8]	0	RW	mst_r_ecc_cerr	ビットがセットされると、マスター読み出し RAM ECC 訂正不可能エラー イベントに対し、割り込みがイネーブルになります。
[9]	0	RW	mst_w_ecc_cerr	ビットがセットされると、マスター書き込み RAM ECC 訂正不可能エラー イベントに対し、割り込みがイネーブルになります。
[10]	0	RW	slv_lite_par_err	ビットがセットされると、スレーブ LITE 書き込みパリティ エラー イベントに対し、割り込みがイネーブルになります。
[19:11]	0	RO		予約
[31:20]	0	RO		予約

コンフィギュレーション制御レジスタ (0xE40)

ユーザー アプリケーションが、訂正可能または訂正不可能なエラーが発生したかどうかを示し、該当する AER エラー ステータス レジスタでレポートできるようにするレジスタです。

表 299: コンフィギュレーション制御レジスタ (0xE40)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[0]	0	RW	uc_err	修正不可能なエラーを検出したことを示します。ユーザーロジック内で、内部エラーとして PCIe Express アドバンスエラーレポートメカニズムを使用してレポートする必要のある訂正不可能なエラーが検出されたことを示すため、ユーザーアプリケーションにより 1 が書き込まれます。これに対応して、コアはすべての有効なファクションの AER 訂正不可能エラーステータスレジスタの未訂正内部エラーステータスビットをセットし、また可能な場合にはエラーメッセージを送信します。このエラーは、ファクション特定のものではありません。このビットは 1 クロックサイクル間のみアサートされ、次のクロックサイクルでは 0 に自動的にリセットされます。
[1]	0	RW	c_err	修正可能なエラーを検出したことを示します。ユーザーロジック内で、内部エラーとして PCIe Express アドバンスエラーレポートメカニズムを使用してレポートする必要のある訂正可能なエラーが検出されたことを示すため、ユーザーアプリケーションにより 1 が書き込まれます。これに対応して、コアはすべての有効なファクションの AER 修正可能エラーステータスレジスタの Corrected Internal Error Status ビットをセットし、また可能な場合にはエラーメッセージを送信します。このエラーは、ファクション特定のものではありません。このビットは 1 クロックサイクル間のみアサートされ、次のクロックサイクルでは 0 に自動的にリセットされます。
[31:2]	0	RO		予約

スレーブ エラー AID レジスタ (0xE44)

どの ID でスレーブ チェッカー違反が発生したかを判断するためのレジスタです。割り込みデコードレジスタにある対応する割り込みビットがクリアになった後、最初に違反が発生した ID のみが記録されます。

表 300: スレーブ エラー AID レジスタ (0xE44)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[7:0]	0	RO	tz_violation_aid	TrustZone 違反の割り込みビットがクリアになった後、最初にスレーブ TrustZone 違反が発生した ID をレポートします。
[15:8]	0	RO	tz_violation_func	TrustZone 違反の割り込みビットがクリアになった後、最初にスレーブ TrustZone 違反が発生したファクション番号 (SMID) をレポートします。
[23:16]	0	RO	cpl_err_aid	スレーブ UR、スレーブ CA、またはスレーブ エラーポイズンの割り込みビットがクリアになった後、最初にスレーブ完了エラーが発生した ID をレポートします。
[31:24]	0	RO	cpl_err_func	スレーブ UR、スレーブ CA、またはスレーブ エラーポイズンの割り込みビットがクリアになった後、最初にスレーブ完了エラーが発生したファクション番号 (SMID) をレポートします。

ユーザー IRQ リクエスト レジスタ (0xE58)

ユーザーアプリケーションから `usr_irq_req` インターフェイスにアクセスできるようにするためのレジスタです。

表 301: ユーザー IRQ リクエスト レジスタ (0xE58)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[15:0]	0	RW	usr_irq_req_set	ビット [n] に 1 を書き込んで、usr_irq_req[n] をセットします。読み出しでは、usr_irq_req の現在の値が戻されます。usr_irq_req の定義は、DMA IRQ ブロックを参照してください。 MSI または MSI-X の場合、ビット [n] は、usr_irq_ack[n] が受信されると、自動的にクリアになります。
[19:16]	0	wo	usr_irq_req_clr	INTx の場合、ビット [n] に 1 を書き込んで、usr_irq_req[n] をクリアにします。usr_irq_req の定義は、DMA IRQ ブロックを参照してください。
[31:20]	0	RO		予約

ユーザー IRQ 肯定応答レジスタ (0xE5C)

ユーザー アプリケーションから usr_irq_ack インターフェイスにアクセスできるようにするためのレジスタです。

表 302: ユーザー IRQ 肯定応答レジスタ (0xE5C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[15:0]	0	RO	usr_irq_ack	usr_irq_ack の値を示します。usr_irq_ack の定義は、DMA IRQ ブロックを参照してください。 ビット [n] は、INTx/MSI/MSI-X に対して usr_irq_req[i] がセットされているときか、INTx に対して usr_irq_req[i] がクリアになっているときに、自動的にクリアになります。
[31:16]	0	RO	usr_irq_fail	usr_irq_fail の値を示します。usr_irq_fail の定義は、DMA IRQ ブロックを参照してください。ビット [n] は、usr_irq_ack[n] がセットされているときにのみ有効です。

PCIe TX メッセージ制御レジスタ (0xE60)

PCIe MSI-X メッセージを生成し、リモート コンポーネントに送信するためのレジスタです。メッセージを合成するため、TX_MSG_HDR_L、TX_MSG_HDR_H、および TX_MSG_DFIFO と共にこのレジスタを使用します。

表 303: PCIe TX メッセージ制御レジスタ (0xE60)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[0]	0	RW	msg_execute	TX_MSG_CSR、TX_MSG_HDR_L、TX_MSG_HDR_H、および TX_MSG_DFIFO (前もってプログラムしておく必要がある) で定義される PCIe TX メッセージを送信するには、1 を書き込みます。読み出しで、メッセージの送信ステータスが返されます。 0b: PCIe に送信されます。完了ステータスの msg_fail をチェックします。 1b: 進行中です。

表 303: PCIe TX メッセージ制御レジスタ (0xE60) (続き)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[3:1]	0	RW	msg_routing	PCIe TX メッセージのメッセージ送信フィールドをプログラムします。 000b: ルート コンプレックスに送信 010b: ID 別に送信 011b: ルート コンプレックスから出力 100b: ローカル - レシーバーで終端 101b: 収集して、ルート コンプレックスに送信 その他: 予約 各メッセージの有効な設定は、PCIe 仕様を参照してください。
[7:4]	0	RW	msg_data_ptr_sel	TX_MSG_DFIFO ポインタを上書きします。これはデバッグ用で、普通の状況では 0 にセットする必要があります。
[15:8]	0	RW	msg_code	PCIe TX メッセージのメッセージ コード フィールドをプログラムします。 0001_0100b: PM_Active_State_Nak 0001_1000b: PM_PME 0001_1001b: PME_Turn_Off 0001_1011b: PME_TO_Ack 0111_1110b: Vendor_Defined Typo0 0111_1111b: Vendor_Defined Typo1 その他: 予約
[20:16]	0	RW	msg_data_length	PCIe TX メッセージの D ワードの長さをプログラムします。 0: ペイロードなし (Msg) 1: 1 個の D ワード (MsgD) 16: 16 個の D ワード (MsgD) その他: 予約 各メッセージの有効な設定は、PCI-SIG Specifications (http://www.pcisig.com/specifications) を参照してください。ベンダー定義のメッセージでは最大 16 個の D ワード (64 バイト) がサポートできます。
[22:21]	0	RO		予約
[23]	0	RW1C	msg_fail	メッセージの完了ステータスを示します。メッセージが送信されると有効になります。1 を書き込むと、このビットはクリアになります。msg_execute に 1 を書き込んでも、このビットはクリアになります。 0b: 完了済み 1b: 失敗
[31:24]	0	RW	msg_function	PCIe TX メッセージのリクエスト ファンクション番号 フィールドをプログラムします。

PCIe TX メッセージ ヘッダー L レジスタ (0xE64)

PCIe TX メッセージのヘッダー バイト 8 から 11 までをプログラムするためのレジスタです。

表 304: PCIe TX メッセージ ヘッダー L レジスタ (0xE64)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[7:0]	0	RW	msg_tlp_hdr8	PCIe TX メッセージのヘッダー バイト 8 をプログラム します。 各メッセージの有効な設定は、PCIe 仕様を参照してくだ さい。
[15:8]	0	RW	msg_tlp_hdr9	PCIe TX メッセージのヘッダー バイト 9 をプログラム します。 各メッセージの有効な設定は、PCIe 仕様を参照してくだ さい。
[23:16]	0	RW	msg_tlp_hdr10	PCIe TX メッセージのヘッダー バイト 10 をプログラム します。 各メッセージの有効な設定は、PCIe 仕様を参照してくだ さい。
[31:24]	0	RW	msg_tlp_hdr11	PCIe TX メッセージのヘッダー バイト 11 をプログラム します。 各メッセージの有効な設定は、PCIe 仕様を参照してくだ さい。

PCIe TX メッセージ ヘッダー H レジスタ (0xE68)

PCIe TX メッセージのヘッダー バイト 12 から 15 までをプログラムするためのレジスタです。

表 305: PCIe TX メッセージ ヘッダー H レジスタ (0xE68)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[7:0]	0	RW	msg_tlp_hdr12	PCIe TX メッセージのヘッダー バイト 12 をプログラム します。 各メッセージの有効な設定は、PCIe 仕様を参照してくだ さい。
[15:8]	0	RW	msg_tlp_hdr13	PCIe TX メッセージのヘッダー バイト 13 をプログラム します。 各メッセージの有効な設定は、PCIe 仕様を参照してくだ さい。
[23:16]	0	RW	msg_tlp_hdr14	PCIe TX メッセージのヘッダー バイト 14 をプログラム します。 各メッセージの有効な設定は、PCIe 仕様を参照してくだ さい。
[31:24]	0	RW	msg_tlp_hdr15	PCIe TX メッセージのヘッダー バイト 15 をプログラム します。 各メッセージの有効な設定は、PCIe 仕様を参照してくだ さい。

PCIe TX メッセージ データ FIFO レジスタ (0xE6C)

ペイロードを PCIe TX メッセージ (MsgD) と共に送信されるようにプログラムするためのレジスタです。

表 306: PCIe TX メッセージ データ FIFO レジスタ (0xE6C)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:0]	0	RW	msg_tlp_data	最初の D ワードからメッセージの長さまで、1 つずつ PCIe TX メッセージを出力します。書き込むたびに書き込みポインタが 1 つずつ増え、最大 15 まで増えます。書き込みポインタは、前の TX メッセージがリセットまたは確約されると 0 を返します。 デバッグ目的で、書き込みポインタおよび読み出しポインタは、msg_data_ptr_sel をプログラミングして上書きできます。書き込みは D ワードの値 (0 から 15 まで) をプログラムします。読み出しは D ワードの値 (0 から 15 まで) を返します。

PCIe RX メッセージ制御およびステータス レジスタ (0xE70)

PCIe RX メッセージのステータスおよび制御にアクセスを提供するレジスタです。

表 307: PCIe RX メッセージ制御およびステータス レジスタ (0xE70)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[0]	0	RO	mfifo_not_empty	メッセージ FIFO に読み出す必要のある VDM メッセージがあることを示します。
[1]	0	RW1C	mfifo_overflow	メッセージ FIFO がオーバーフローしていて、VDM メッセージが破棄されたことを示します。1 を書き込むとオーバーフロー ステータスがクリアになります。
[3:2]	0	RO		予約
[7:4]	0	RO	msg_count	メッセージ FIFO に格納されている VDM メッセージの数を示します。読み出す必要のあるメッセージの数はユーザー アプリケーションで確認できます。
[12:8]	0	RO	mfifo_read_ptr	RX MFIFO READ の現在の読み出しポインタを示します。これはデバッグ用です。
[15:13]	0	RO		予約
[31:16]	0	RO	overflow_rid	リセットまたは mfifo_overflow がクリアされた後、最初に破棄された VDM メッセージのリクエスター ID を示します。

PCIe RX メッセージ FIFO レジスタ (0xE74)

この位置からの読み出しにより、VDM メッセージが返されます。読み出しは問題を起すものではありません。FIFO からこのメッセージを削除するには書き込みが必要です。その書き込み値は無視されます。

表 308: PCIe RX メッセージ FIFO レジスタ (0xE74)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW1C	msg_tlp_dw	VDM メッセージからの D ワードが、ヘッダー D ワードからペイロード D ワードへ連続して送られることを示します。各読み出しの後、ユーザー アプリケーションが、D ワードを削除するため、このレジスタに書き込みを実行する必要があります。 最初のヘッダー D ワードのフィールド: [31:16] リクエスター ID [15:8] メッセージ コード [7:5] メッセージ ルーティング [4:0] ペイロード DW の長さ。0 の場合、ペイロードがなく、ユーザー アプリケーションは、残りのヘッダーを取得するため、さらに 2 つの D ワードを読み込む必要があります。 2 番目のヘッダー D ワードのフィールド: [31:0] ヘッダー バイト 11 - 8 3 番目のヘッダー D ワードのフィールド: [31:0] ヘッダー バイト 15 - 12 ペイロード D ワードのフィールド: [31:0] ペイロード バイト (N+3) - N

マスター保留カウンター レジスタ (0xE78)

デバッグおよびパフォーマンス モニター用のブリッジ マスター ポートでの保留リクエストをカウントするレジスタです。

表 309: マスター保留カウンター レジスタ (0xE78)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[7:0]	0	RO	wr_pend_cnt	ブリッジ マスター ポートでの保留の書き込みカウント。
[15:8]	0	RO	rd_pend_cnt	ブリッジ マスター ポートでの保留の読み出しカウント。
[31:16]	0	RO		予約

PCIe TX MSI/MSI-X 制御およびステータス レジスタ (0xE7C)

PCIe TX MSI (SR-IOV に対しては無効) または MSI-X 割り込みを生成するためのレジスタです。

表 310: PCIe TX MSI/MSI-X 制御およびステータス レジスタ (0xE7C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[4:0]	0	RW	int_vector	MSI または MSI-X のベクター番号。このフィールドは int_set をセットしてプログラムする必要があります。
[7:5]	0	RO		予約
[15:8]	0	RW	int_function	MSI または MSI-X のファンクション番号。このフィールドは int_set をセットしてプログラムする必要があります。MSI の場合は、物理ファンクションのみが有効です。

表 310: PCIe TX MSI/MSI-X 制御およびステータス レジスタ (0xE7C) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[16]	0	RW	int_set	PCIe TX MSI/MSI-X 割り込みを送信するため 1 を書き込みます。読み出しで、割り込みの送信ステータスが返されます。 0b: PCIe に送信されます。完了ステータスの int_fail をチェックします。 1b: 進行中です。
[17]	0	RW	int_is_msix	MSI-X または MSI です。このフィールドは int_set をセットしてプログラムする必要があります。 0b: MSI 1b: MSI-X
[19:18]	0	RO		予約
[20]	0	RW1C	int_fail	割り込みの完了ステータスを示します。割り込みが送信されると有効になります。1 を書き込むと、このビットはクリアになります。int_set に 1 を書き込んでも、このビットはクリアになります。 0b: 完了済み 1b: 失敗
[31:21]	0	RO		予約

VSEC 機能 2 レジスタ (0xED8)

コアのメモリ空間が基本統合ブロック PCIe コンフィギュレーション空間の一部として表示されるようにするレジスタです。VSEC は、最後の改善された機能構造のすぐ後に挿入されます。

表 311: VSEC 機能 2 レジスタ (0xED8)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[15:0]	0xB	RO	cap_id	この改善機能をベンダー別機能として特定する PCIe 定義の ID です。
[19:16]	0x1	RO	cap_ver	この機能構造のバージョンを示します。
[31:20]	0	RO	nxt_cap_offset	次の機能のオフセットを示します。

VSEC ヘッダー 2 レジスタ (0xEDC)

VSEC 構造のレイアウトおよび内容、リビジョンおよび長さの ID (ベンダー内) を提供するレジスタです。VSEC ヘッダー 2 レジスタは、VSEC ヘッダー 2 レジスタ (オフセット 0xEE0) の直後に開始する AXI ベース アドレス変換コンフィギュレーション レジスタを含む AXI ブリッジの一部です。

表 312: VSEC ヘッダー 2 レジスタ (0xEDC)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[15:0]	0x2	RO	vsec_id	VSEC 構造の性質およびフォーマットを識別する ID です。
[19:16]	0	RO	vsec_rev	この機能構造のバージョンを示します。
[31:20]	0x38	RO	vsec_length	VSEC 機能構造全体の長さ (VSEC 機能レジスタを含む) をバイトで示します。

AXI ベース アドレス変換コンフィギュレーション レジスタ (オフセット - 0xEE0 - 0xF0C)

AXI ベース アドレス変換コンフィギュレーション レジスタおよびそのオフセットは、最初の表に示されており、レジスタビットも説明されています。レジスタのこのセットは、PCIe BAR のアドレス幅に基づいて 2 つのコンフィギュレーションで使用できます。PCIe BAR が 32 ビットのアドレス空間に設定されていれば、変換ベクターを AXIBAR2PCIEBAR_nL レジスタに配置する必要があります (n には PCIe BAR の番号が入ります)。また、PCIe BAR が 64 ビットのアドレス空間に設定されていれば、変換ベクターを AXIBAR2PCIEBAR_nU レジスタに配置する必要があります、最下位から 32 ビットが AXIBAR2PCIEBAR_nL に書き込まれます。この場合は、アドレス変換レジスタに無効な値が書き込まれないように注意してください。

表 313: AXI ベース アドレス変換コンフィギュレーション レジスタ (オフセット 0xEE0 - 0xF0C)

オフセット	ビット	レジスタ ニーモニック
0xEE0	[31:0]	AXIBAR2PCIEBAR_0U
0xEE4	[31:0]	AXIBAR2PCIEBAR_0L
0xEE8	[31:0]	AXIBAR2PCIEBAR_1U
0xEEC	[31:0]	AXIBAR2PCIEBAR_1L
0xEF0	[31:0]	AXIBAR2PCIEBAR_2U
0xEF4	[31:0]	AXIBAR2PCIEBAR_2L
0xEF8	[31:0]	AXIBAR2PCIEBAR_3U
0xEFC	[31:0]	AXIBAR2PCIEBAR_3L
0xF00	[31:0]	AXIBAR2PCIEBAR_4U
0xF04	[31:0]	AXIBAR2PCIEBAR_4L
0xF08	[31:0]	AXIBAR2PCIEBAR_5U
0xF0C	[31:0]	AXIBAR2PCIEBAR_5L

サブシステムを使用するデザイン

一般的なデザイン ガイドライン

サンプル デザインの使用

Vivado® Design Suite によって作成される QDMA Subsystem for PCIe 各インスタンスには、デバイスにインプリメントしてからシミュレーションできるサンプル デザインが提供されています。サンプル デザインは、カスタム デザインを構築するための開始点として使用したり、問題発生時にユーザー アプリケーションをチェックしたりするのに使用できます。サブシステムのサンプル デザインの使用方法やカスタマイズ方法は、サンプル デザインについての資料を参照してください。

信号にレジスタを付ける

プログラマブル デバイスのデザインのタイミングを簡潔にし、パフォーマンスを向上させるには、ユーザー アプリケーションとサブシステムの間ですべての入力と出力にレジスタを付けます。つまり、ユーザー アプリケーションからのすべての入力と出力はフリップフロップを介すことになります。信号にレジスタを付けることが不可能なパスがある可能性もありますが、信号にレジスタを付けるとタイミング解析が容易になり、またサイリンクス ツールでのデザインの配置配線も簡単になります。

タイミング クリティカルな信号の特定

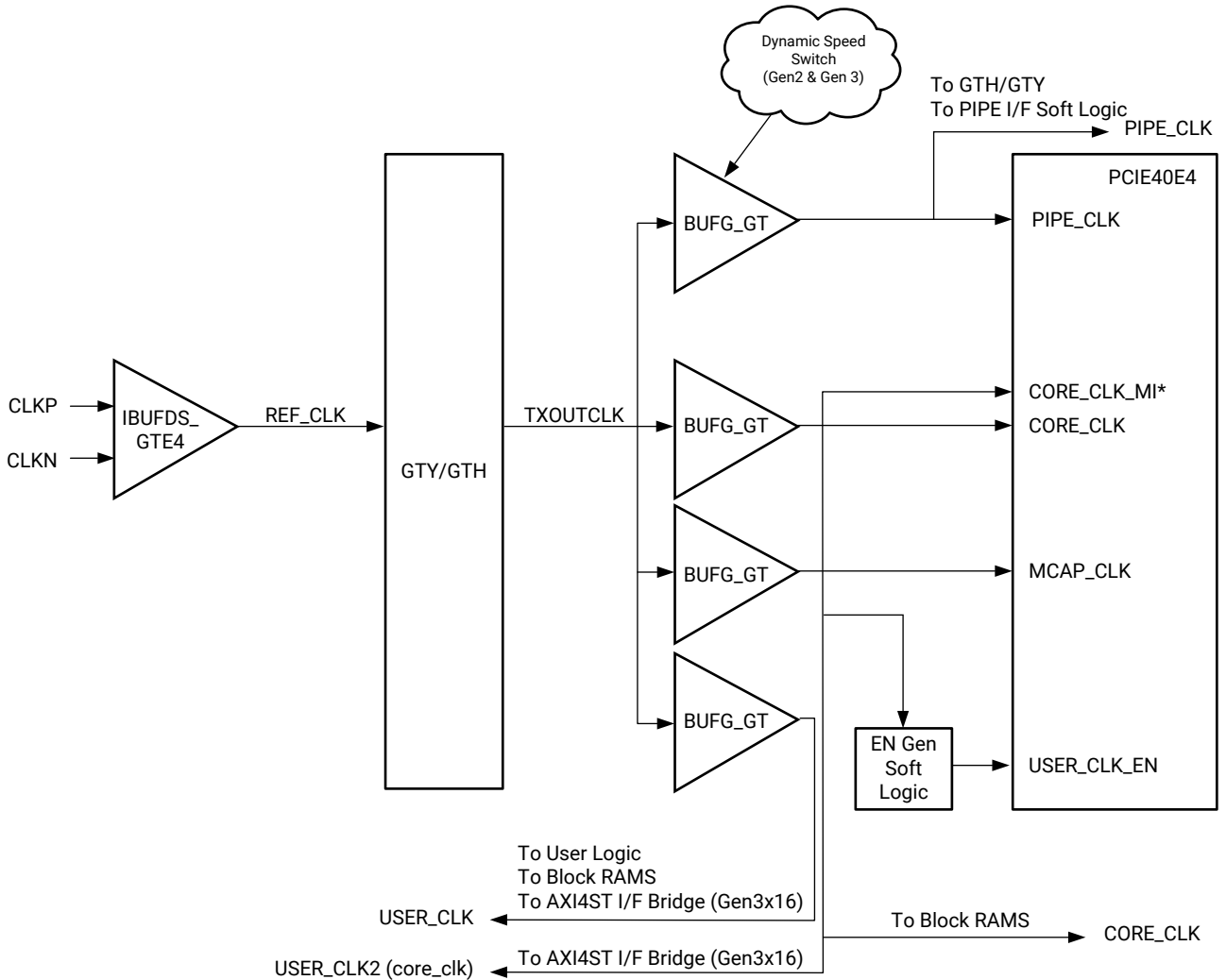
サンプル デザインで提供される制約は、クリティカルな信号を識別し、適用すべきタイミング制約を特定するのに役立ちます。

許可された変更のみ実行

サブシステムは変更しないでください。変更を加えると、システムのタイミングやプロトコル適合性に悪影響を与える可能性があります。サポートされるサブシステムのユーザー コンフィギュレーションは、サブシステムを生成するときにカスタマイズ IP ダイアログ ボックスでオプションを選択したもののみです。

クロッキング

図 24: クロッキング



X20597-040218

PCIe クロック (pipe_clk、core_clk、user_clk、および mcap_clk) はすべて、txoutclk ピンから供給される bufg_gt で駆動されます。これらのクロックは、CPLL を経由した gtrefclk0 からの派生クロックです。QPLL を使用するアプリケーションでは、QPLL は GT PCS/PMA ブロックにのみ供給され、txoutclk は引き続き CPLL から生成されます。IP のユーザー インターフェイス信号はすべて同じクロック (user_clk) と同期し、設定されたリンクスピードと幅に応じて 62.5、125、または 250 MHz の周波数になります。QDMA Subsystem for PCIe およびユーザーロジックは主に user_clk で動作します。

デザイン フローの手順

このセクションでは、サブシステムのカスタマイズと生成、サブシステムの制約、およびこの IP サブシステム特定のシミュレーション、合成、インプリメンテーション手順を説明します。標準 Vivado® デザイン フローおよび IP インテグレーターの詳細は、次の Vivado Design Suite ユーザー ガイドを参照してください。

- 『Vivado Design Suite ユーザー ガイド: IP インテグレーターを使用した IP サブシステムの設計』 (UG994: [英語版](#)、[日本語版](#))
- 『Vivado Design Suite ユーザー ガイド: IP を使用した設計』 (UG896: [英語版](#)、[日本語版](#))
- 『Vivado Design Suite ユーザー ガイド: 入門』 (UG910: [英語版](#)、[日本語版](#))
- 『Vivado Design Suite ユーザー ガイド: ロジック シミュレーション』 (UG900: [英語版](#)、[日本語版](#))

サブシステムのカスタマイズおよび生成

ここでは、ザイリンクス ツールを使用し、Vivado® Design Suite でサブシステムをカスタマイズおよび生成する方法について説明します。

Vivado IP インテグレーターでサブシステムをカスタマイズおよび生成する場合は、『Vivado Design Suite ユーザー ガイド: IP インテグレーターを使用した IP サブシステムの設計』 (UG994: [英語版](#)、[日本語版](#)) を参照してください。IP インテグレーターでは、デザインの検証または生成時に一部のコンフィギュレーション値が自動的に計算される場合があります。値が変更されるかどうかは、この章のパラメーターの説明で確認ください。パラメーター値を確認するには、Tcl コンソールで `validate_bd_design` コマンドを実行します。

IP はユーザー デザインに合わせてカスタマイズできます。それには、IP サブシステムに関連する各種パラメーターの値を次の手順に従って指定します。

1. IP カタログから IP を選択します。
2. 選択した IP をダブルクリックするか、ツールバーまたは右クリック メニューから [Customize IP] をクリックします。

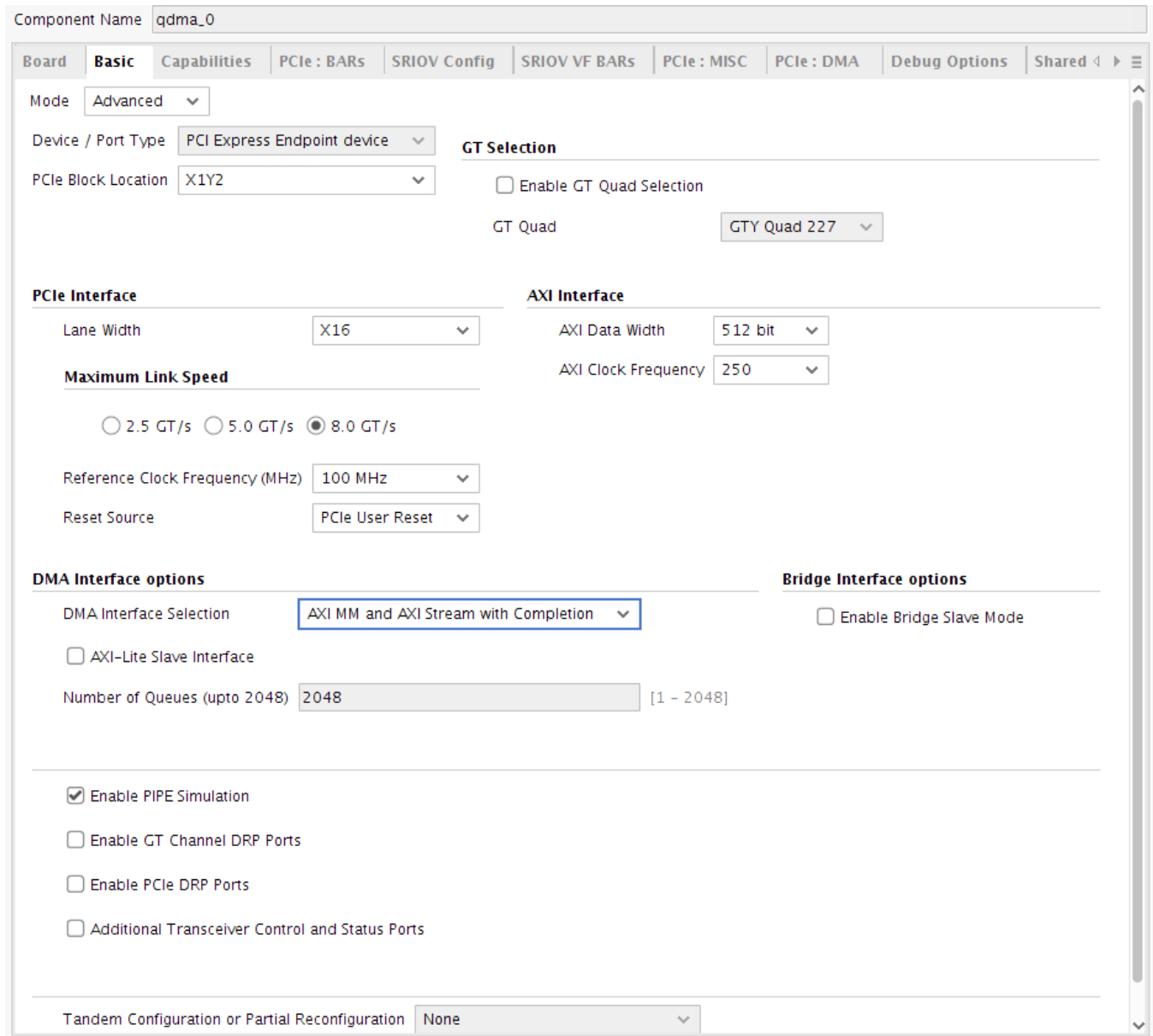
詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』 (UG896: [英語版](#)、[日本語版](#)) および『Vivado Design Suite ユーザー ガイド: 入門』 (UG910: [英語版](#)、[日本語版](#)) を参照してください。

この章の図には Vivado IDE のスクリーンショットが使用されていますが、現在のバージョンとはレイアウトが異なる場合があります。

[Basic] タブ

次の図に [Basic] タブを示します。

図 25: [Basic] タブ



Component Name: qdma_0

Board: **Basic** | Capabilities | PCIe : BARs | SRIOV Config | SRIOV VF BARs | PCIe : MISC | PCIe : DMA | Debug Options | Shared

Mode: **Advanced**

Device / Port Type: PCI Express Endpoint device

PCIe Block Location: X1Y2

GT Selection

Enable GT Quad Selection

GT Quad: GTY Quad 227

PCIe Interface

Lane Width: X16

AXI Interface

AXI Data Width: 512 bit

AXI Clock Frequency: 250

Maximum Link Speed

2.5 GT/s 5.0 GT/s 8.0 GT/s

Reference Clock Frequency (MHz): 100 MHz

Reset Source: PCIe User Reset

DMA Interface options

DMA Interface Selection: AXI MM and AXI Stream with Completion

AXI-Lite Slave Interface

Number of Queues (upto 2048): 2048 [1 - 2048]

Bridge Interface options

Enable Bridge Slave Mode

Enable PIPE Simulation

Enable GT Channel DRP Ports

Enable PCIe DRP Ports

Additional Transceiver Control and Status Ports

Tandem Configuration or Partial Reconfiguration: None

- [Mode]: コアのコンフィギュレーション モードとして [Basic] または [Advanced] のいずれかを選択します。
- [Device/Port Type]: PCI Express® エンドポイント デバイス モードのみがサポートされています。
- [GT Selection] / [Enable GT Quad Selection]: レーン 0 のあるクワッドを選択します。
- [PCIe Block Location]: ロケーション特定の制約ファイルおよびピン配置を生成する統合ブロックを使用可能なリストから選択します。この選択は、デフォルトのサンプル デザイン スクリプトで使用されます。ザイリンクス開発ボードが選択されている場合は、このオプションは設定できません。

- [Lane Width]: コアには初期レーン幅を選択する必要があります。選択可能なレーン幅とそれに対応するコアは、『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』(PG213: [英語版](#)、[日本語版](#)) で定義されています。レーン幅の広いコアがそれよりも狭いレーン幅のデバイスに接続されている場合は、狭いほうのレーン幅にトレーニングダウンできます。4、8、16 レーンのいずれかを選択できます。
- [Maximum Link Speed]: デバイスでサポートされている最大リンク スピードを選択できます。デバイスでサポートされているレーン幅およびリンク スピードは、『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』(PG213: [英語版](#)、[日本語版](#)) で定義されています。高速なリンク スピードのコアがそれより低いリンク スピードのデバイスに接続されている場合、低いほうのリンク スピードにトレーニングできます。デフォルトは Gen3 です。
- [Reference Clock Frequency]: デフォルト値は 100 MHz です。
- [Reset Source]: 次のいずれかを選択できます。
 - [PCIe User Reset]: リンクが確立された後に、PCIe コアから供給されるユーザー リセットです。PCIe リンクがダウンすると、ユーザー リセットがアサートされ、コアがリセット モードになります。リンクがまたアップすると、ユーザー リセットはデアサートされます。
 - [Phy Ready]: このオプションが選択されていると、コアは PCIe リンク ステータスの影響を受けなくなります。
- [AXI Data Width]: 128、256、または 512 ビットを選択します (UltraScale+ の場合のみ)。『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』(PG213: [英語版](#)、[日本語版](#)) で定義されているように、コアでインターフェイス幅を選択できます。[Customize IP] ダイアログ ボックスに設定されているデフォルトのインターフェイス幅は、最も狭いインターフェイス幅となります。
- [AXI Clock Frequency]: レーンの幅/スピードによりませんが、250 MHz です。
- [DMA Interface Option]: 次の中から 1 つオプションを選択できます。
 - [AXI Memory Mapped and AXI Stream with Completion]
 - [AXI Memory Mapped only]
 - [AXI Stream with Completion]
 - [AXI Memory Mapped with Completion]
- [AXI Lite Slave Interface]: AXI4-Lite スレーブ インターフェイスをイネーブルにする場合は選択します。
- [Enable Bridge Slave Mode]: AXI メモリ マップド スレーブ インターフェイスをイネーブルにする場合は選択します。
- [Enable PIPE Simulation]: より高速なシミュレーション用の PIPE シミュレーションをイネーブルにします。これはシミュレーション専用です。
- [Enable GT DRP Ports]: GT チャネルの DRP ポートをイネーブルにします。
- [Enable PCIe DRP Ports]: PCIe の概要 DRP ポートをイネーブルにします。
- [Additional Transceiver Control and Status Ports]: 追加ポートをイネーブルにする場合は選択します。
- [Tandem Configuration or Partial Reconfiguration]: デザインに該当する場合はこの機能を選択します。

[Capabilities] タブ

次の図に [Capabilities] タブを示します。

図 26: [Capabilities] タブ

Basic	Capabilities	PCIe : BARS	PCIe : MISC	PCIe : DMA	Debug Options	Shared Logic	GT Settings
SRIOV Capabilities							
<input type="checkbox"/> SRIOV Capability							
<input type="checkbox"/> Enable FLR							
<input type="checkbox"/> Enable Mailbox among functions							
Physical Functions							
Total Physical Functions <input type="text" value="4"/>							
PF - ID Initial Values							
PF#	Vendor ID	Device ID	Revision ID	Subsystem Vendor ID	Subsystem ID		
PF0	10EE	903F	00	10EE	0007		
PF1	10EE	913F	00	10EE	0007		
PF2	10EE	923F	00	10EE	0007		
PF3	10EE	933F	00	10EE	0007		
Class Code							
PF#	Use Classcode Lookup Assistant	Base Class Menu	Base Class Value	Subclass Interface Menu	Subclass Value	Interface Value	Class Code
PF0	<input type="checkbox"/>	Memory controller	05	Other memory cont...	80	00	058000
PF1	<input type="checkbox"/>	Memory controller	05	Other memory cont...	80	00	058000
PF2	<input type="checkbox"/>	Memory controller	05	Other memory cont...	80	00	058000
PF3	<input type="checkbox"/>	Memory controller	05	Other memory cont...	80	00	058000

- [SRIOV Capability]: SR-IOV (Single Root Port I/O Virtualization) 機能をイネーブルにします。拡張型の SR-IOV PCIe は統合ブロックによりインプリメントされます。このオプションがオンになっていると、選択したすべての物理ファンクション (PF) に SR-IOV がインプリメントされます。SR-IOV 機能がイネーブルになっている場合は、MSI-X 割り込みのみがサポートされます。
- [Enable Mailbox among functions]: ファンクション間で通信するためのメールボックス システムです。[SR-IOV Capability] (上記) がオンになっていると、このオプションはデフォルトでオンになります。
- [Enable FLR]: ファンクション レベル リセット (FLR) ポートをイネーブルにします。[SR-IOV Capability] (上記) がオンになっていると、このオプションはデフォルトでオンになります。
- [Total Physical Functions]: 最大 4 つの PF をイネーブルにできます。
- [PF - ID Initial Values]:
- [Vendor ID]: デバイスまたはアプリケーションの製造者 ID です。各 ID の重複を避けるため PCI Special Interest Group によって有効な ID が割り当てられています。デフォルト値「10EEh」はザイリックスのベンダー ID です。ここにベンダー ID を入力します。「FFFFh」は予約されています。
- [Device ID]: アプリケーションの ID です。選択されている設定によって異なりますが、デフォルト値は 70h です。このフィールドには任意値を指定できます。アプリケーションに合わせて変更してください。

このパラメーターは、次に基づいて評価されます。

- デバイス ファミリ。UltraScale+™ の場合は 9、UltraScale™ の場合は 8、7 シリーズ の場合は 7 です。
- EP または RP モード
- リンク幅
- リンク スピード

上記の値のいずれかが変更された場合、[Device ID] の値が再評価され、現在設定されている値が上書きされます。



推奨: リンク幅、リンク スピード、デバイス ポート タイプを最初に変更してから [Device ID] の値を変更することを推奨します。IP を生成する前に、[Device ID] の値が正しく設定されていることを確認してください。

- [Revision ID]: デバイスまたはアプリケーションのリビジョンを指し、デバイス ID に付随しています。デフォルト値は「00h」です。アプリケーションに適した値を入力してください。
- [Subsystem Vendor ID]: デバイスまたはアプリケーションの製造者をさらに細分化させるための ID です。ここにサブシステム ベンダー ID を入力します。デフォルト値は「10EEh」です。通常、この値はベンダー ID と同じです。この値を「0000h」に設定すると準拠テストで問題が発生します。
- [Subsystem ID]: デバイスまたはアプリケーションの製造者をさらに細分化させるための ID です。この値は通常デバイス ID と同じです。デフォルト値は選択されているレーン幅およびリンク スピードにより異なります。この値を「0000h」に設定すると準拠テストで問題が発生します。
- [Class Code]: デバイスの一般ファンクションを識別します。
- [Use Classcode Lookup Assistant]: 選択されているデバイスの一般機能のベース クラス、サブクラス、およびインターフェイス値が表示されます。選択されている機能に対し 3 つの値しか表示されないため、これらの値をデバイス設定に変換するには、ユーザー自身が [Class Code] の値を手動入力する必要があります。
- [Base Class]: デバイスにより実行される機能のタイプを広義に特定します。
- [Subclass]: デバイス機能をさらに細かく特定します。
- [Interface]: レジスタ レベルのプログラミング インターフェイスを定義し、デバイスに依存したソフトウェアがデバイスとインターフェイスできるようにします。

[PCIe BARs] タブ

次の図に [PCIe BARs] タブを示します。

図 27: [PCIe BARs] タブ

Basic Capabilities **PCIe : BARs** SRIOV Config SRIOV VF BARs PCIe : MISC PCIe : DMA Debug Options Shared Logic GT Settings

Base Address Registers (BARs) serve two purposes. Initially, they serve as a mechanism for the device to request blocks of address space in the system memory map. After the BIOS or OS determines what addresses to assign to the device, the Base Address Registers are programmed with addresses and the device uses this information to perform address decoding.

Bar	Type	64 bit	Prefetchable	Size	Scale	Value (Hex)	PCIe to AXI Translation
<input checked="" type="checkbox"/>	DMA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	128	Kilobytes	FFFFFFFFFFE000C	0x0000000000000000
<input type="checkbox"/>	AXI Bridge Master			128	Megabytes	00000000	0x0000000000000000
<input checked="" type="checkbox"/>	AXI Lite Master	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4	Kilobytes	FFFFFFFFFFF00C	0x0000000000000000
<input type="checkbox"/>	AXI Bridge Master			128	Kilobytes	00000000	0x0000000000000000
<input type="checkbox"/>	AXI Bridge Master	<input type="checkbox"/>	<input type="checkbox"/>	128	Kilobytes	00000000	0x0000000000000000
<input type="checkbox"/>	AXI Bridge Master			128	Kilobytes	00000000	0x0000000000000000
<input type="checkbox"/>	Expansion ROM			2	Kilobytes	00000000	0x0000000000000000

Copy PF0

Bar	Type	64 bit	Prefetchable	Size	Scale	Value (Hex)	PCIe to AXI Translation
<input checked="" type="checkbox"/>	DMA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	128	Kilobytes	FFFFFFFFFFE000C	0x0000000000000000
<input type="checkbox"/>	AXI Bridge Mas...			128	Megabytes	00000000	0x0000000010000000
<input checked="" type="checkbox"/>	AXI Lite Master	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4	Kilobytes	FFFFFFFFFFF00C	0x0000000010000000
<input type="checkbox"/>	AXI Bridge Mas...			128	Kilobytes	00000000	0x0000000000000000
<input type="checkbox"/>	AXI Bridge Mas...	<input type="checkbox"/>	<input type="checkbox"/>	128	Kilobytes	00000000	0x0000000000000000
<input type="checkbox"/>	AXI Bridge Mas...			128	Kilobytes	00000000	0x0000000000000000
<input type="checkbox"/>	Expansion ROM			2	Kilobytes	00000000	0x0000000000000000

PF2

PF3

- ベース アドレス レジスタ (BAR) の概要: エンドポイント コンフィギュレーションのコアでは、最大 6 つの 32 ビット BAR または 3 つの 64 ビット BAR、および拡張 ROM (read-only memory) BAR がサポートされます。ルートポート コンフィギュレーションのコアでは、最大 2 つの 32 ビット BAR または 1 つの 64 ビット BAR、および拡張 ROM (read-only memory) BAR がサポートされます。BAR は次のいずれかのサイズに設定できます。
 - 32 ビットの BAR: アドレス空間は 128 バイトから 2 ギガバイトまでです。[DMA]、[AXI Lite Master]、または [AXI Bridge Master] に使用されます。
 - 64 ビットの BAR: アドレス空間は 128 バイトから 8 ギガバイトまでです。[DMA]、[AXI Lite Master]、または [AXI Bridge Master] に使用されます。

すべての BAR レジスタに次のオプションがあります。



重要: ファンクションおよびキューをサポートするため、DMA には大きな空間が必要です。デフォルトでは、DMA BAR には 64 ビットの BAR 空間が選択されています。これは PF および VF のどちらの BAR にも当てはまります。64 ビットまたは 32 ビットの BAR 空間を選択する前に、まずはデザイン ニーズを計算する必要があります。

BAR のオプションは選択可能です。デフォルトでは、DMA は BAR 0 (64 ビット)、AXI-Lite マスターは BAR 2 (64 ビット) です。これらのオプションはユーザーのニーズに応じて変更できます。

- [Bar]: チェック ボックスをオンにすると BAR がイネーブルになります。チェック ボックスをオフにすると BAR がディスエーブルになります。
- [Type]: [DMA] (BAR0 に固定)、[AXI Lite Master] (イネーブルになっている場合は BAR1 に固定)、[AXI Bridge Master] (イネーブルになっている場合は BAR2 に固定) から選択します。それ以外の BAR に関しては、[AXI Lite Master] または [AXI Bridge Master] を選択します。BAR6 を選択すると、拡張 ROM がイネーブルになります。

64 ビットの BAR の場合は (デフォルト)、[DMA] は BAR0 に固定、[AXI Lite Master] は BAR1 に固定 (イネーブルの場合)、[AXI Bridge Master] は BAR2 に固定 (イネーブルの場合) されています。BAR6 を選択すると、拡張 ROM がイネーブルになります。

- [DMA]: すべての PF で DMA は BAR0 空間に固定されています。[DMA] ではなく [DMA Mailbox Management] を選択できますが、DMA 操作ができなくなります。[DMA Mailbox Management] を選択すると、ホストが拡張メールボックス空間にアクセスします。詳細は、[QDMA_PF_MAILBOX \(0x2400\)](#) レジスタ空間を参照してください。
- [AXI Lite Master]: AXILite インターフェイスの BAR 空間を選択するには、このチェック ボックスをオンにします。サイズ、スケール、アドレス変換は設定可能です。
- [AXI Bridge Master]: AXI ブリッジ マスター インターフェイスの BAR 空間を選択するには、このチェック ボックスをオンにします。サイズ、スケール、アドレス変換は設定可能です。
 - [Size]: サイズ範囲は、BAR が 32 ビットなのか 64 ビットなのかによって異なります。DMA の場合は 128 キロバイトの空間が必要で、これがデフォルト設定になっています。BAR の割り当てが変更になると、128 キロバイトの空間を割り当てることができます。ほかの BAR サイズも選択できますが、値を指定する必要があります。
 - [Value]: 現在の選択に基づいて BAR に割り当てられている値です。
- [Expansion ROM]:
- [Disabling Unused Resources]: 最良の結果を得るには、未使用のベース アドレス レジスタをディスエーブルにしてシステム リソースを節約します。[Customize IP] ダイアログ ボックスで未使用 BAR のチェック ボックスをオフにすると、その BAR はディスエーブルになります。

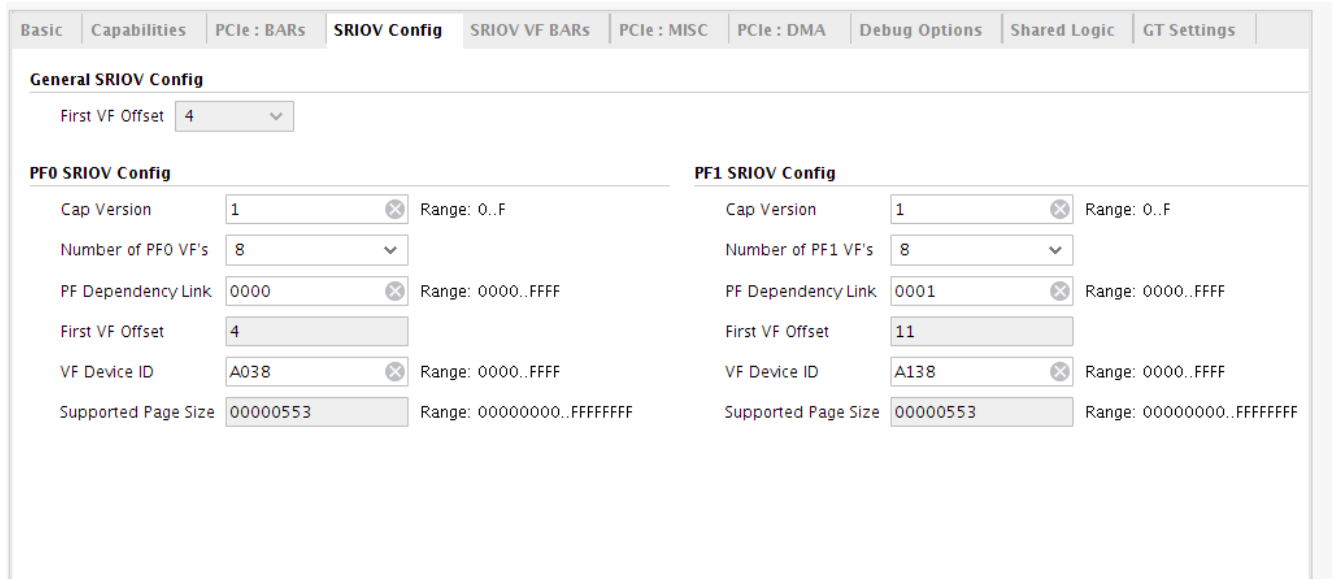
[SRIOV Config] タブ

[SRIOV Config] タブでは、物理ファンクション (PF) の SR-IOV 機能を指定できます。ここで設定した情報は、SR-IOV 機能構造の構築に使用されます。仮想ファンクション (VF) は電源投入時には存在しません。VF はシステム ソフトウェアのシステム機能に基づいてイネーブルになります。VF サポートは、各 PF の SR-IOV 機能構造をスキャンすると確認できます。

注記: [\[Capabilities\] タブ](#) で [SRIOV Capability] を選択すると、[SRIOV Config] タブが表示されます。

次の図に [SRIOV Config] タブを示します。

図 28: [SRIOV Config] タブ



- [General SRIOV Config]: 少なくとも 1 つの VF がイネーブルになっている最初の物理ファンクション (PF) のオフセットを指定します。ARI がイネーブルになっている場合の許容値は 'd4 または 'd64 で、すべての PF にある VF の合計数にこのフィールドの値を足した数は 256 以下である必要があります。ARI がディスエーブルになっている場合は、1 PF + 7 VF の非 ARI SRIOV 設定のみをサポートするため、このフィールドは 1 に設定されます。
- [Cap Version]: PF 用の 4 ビットの SR-IOV 機能バージョンを示します。
- [Number of PFx VF's]: PF に関連付けられている VF の数を示します。VF は合計 252 個あり、4 つの PF にまたがって柔軟に使用できます。
- [PFx Dependency Link]: PF の SR-IOV 機能依存リンクを示します。デバイスのプログラミング モデルのファンクションセットには、ファンクション間にベンダー特定の依存性を持たせることができます。その依存性をこのフィールドを使用して定義します。
- [First VF Offset]: PF の最初の VF のオフセットを示します。PF0 は常にオフセット 0、PF1 は常にオフセット 1 に存在します。Gen3 Integrated Block for PCIe コアでは 6 つの VF を使用でき、そのファンクション番号は 64 ~ 69 です。VF は、PF0 の VF から順にマップされます。たとえば、PF0 に 2 つの VF があり、PF1 には 3 つの VF がある場合は、マップは次のようになります。

PFx_FIRST_VF_OFFSET は、VF の最初のオフセットから PF のオフセットを差し引いて計算されます。

$$\text{PFx_FIRST_VF_OFFSET} = (\text{PFx の最初の VF オフセット} - \text{PFx オフセット})$$

上記の例では、次のオフセットが使用されます。

$$\begin{aligned} \text{PF0_FIRST_VF_OFFSET} &= (64 - 0) = 64 \\ \text{PF1_FIRST_VF_OFFSET} &= (66 - 1) = 65 \end{aligned}$$

PF0 は常に 64 で、PF0 には 1 つ以上の VF があることが前提です。PF1 の最初のオフセットは、PF0 に接続されている VF 数で計算され、次のような疑似コードで定義されます。

$$\text{PF1_FIRST_VF_OFFSET} = 63 + \text{NUM_PF0_VFS}$$

- [VF Device ID]: PF に関連付けられている VF すべての 16 ビットのデバイス ID を示します。

- [SRIOV Supported Page Size]: PF でサポートされるページ サイズを示します。32 レジスタのビット n がセットされている場合、この PF では $2n+12$ のページ サイズがサポートされます。

[SRIOV VF BARs] タブ

次の図に [SRIOV VF BARs] タブを示します。

図 29: [SRIOV VF BARs] タブ

Basic	Capabilities	PCIe : BARs	SRIOV Config	SRIOV VF BARs	PCIe : MISC	PCIe : DMA	Debug Options	Shared Logic	GT Settings																																																								
Base Address Registers (BARs) serve two purposes. Initially, they serve as a mechanism for the device to request blocks of address space in the system memory map. After the BIOS or OS determines what addresses to assign to the device, the Base Address Registers are programmed with addresses and the device uses this information to perform address decoding.																																																																	
PF0 <table border="1"> <thead> <tr> <th>Bar</th> <th>Type</th> <th>64 bit</th> <th>Prefetchable</th> <th>Size</th> <th>Scale</th> <th>Value (Hex)</th> <th>PCIe to AXI Translation</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/></td> <td>DMA</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>16</td> <td>Kilobytes</td> <td>FFFFFFFFFC00C</td> <td>0x0000000000000000</td> </tr> <tr> <td><input type="checkbox"/></td> <td>AXI Bridge Mas...</td> <td></td> <td></td> <td>4</td> <td>Kilobytes</td> <td>00000000</td> <td>0x0000000040000000</td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>AXI Lite Master</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td>4</td> <td>Kilobytes</td> <td>FFFFFFFFF00C</td> <td>0x0000000040000000</td> </tr> <tr> <td><input type="checkbox"/></td> <td>AXI Bridge Mas...</td> <td></td> <td></td> <td>4</td> <td>Kilobytes</td> <td>00000000</td> <td>0x0000000000000000</td> </tr> <tr> <td><input type="checkbox"/></td> <td>AXI Bridge Mas...</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>4</td> <td>Kilobytes</td> <td>00000000</td> <td>0x0000000000000000</td> </tr> <tr> <td><input type="checkbox"/></td> <td>AXI Bridge Mas...</td> <td></td> <td></td> <td>4</td> <td>Kilobytes</td> <td>00000000</td> <td>0x0000000000000000</td> </tr> </tbody> </table>										Bar	Type	64 bit	Prefetchable	Size	Scale	Value (Hex)	PCIe to AXI Translation	<input checked="" type="checkbox"/>	DMA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	16	Kilobytes	FFFFFFFFFC00C	0x0000000000000000	<input type="checkbox"/>	AXI Bridge Mas...			4	Kilobytes	00000000	0x0000000040000000	<input checked="" type="checkbox"/>	AXI Lite Master	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4	Kilobytes	FFFFFFFFF00C	0x0000000040000000	<input type="checkbox"/>	AXI Bridge Mas...			4	Kilobytes	00000000	0x0000000000000000	<input type="checkbox"/>	AXI Bridge Mas...	<input type="checkbox"/>	<input type="checkbox"/>	4	Kilobytes	00000000	0x0000000000000000	<input type="checkbox"/>	AXI Bridge Mas...			4	Kilobytes	00000000	0x0000000000000000
Bar	Type	64 bit	Prefetchable	Size	Scale	Value (Hex)	PCIe to AXI Translation																																																										
<input checked="" type="checkbox"/>	DMA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	16	Kilobytes	FFFFFFFFFC00C	0x0000000000000000																																																										
<input type="checkbox"/>	AXI Bridge Mas...			4	Kilobytes	00000000	0x0000000040000000																																																										
<input checked="" type="checkbox"/>	AXI Lite Master	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4	Kilobytes	FFFFFFFFF00C	0x0000000040000000																																																										
<input type="checkbox"/>	AXI Bridge Mas...			4	Kilobytes	00000000	0x0000000000000000																																																										
<input type="checkbox"/>	AXI Bridge Mas...	<input type="checkbox"/>	<input type="checkbox"/>	4	Kilobytes	00000000	0x0000000000000000																																																										
<input type="checkbox"/>	AXI Bridge Mas...			4	Kilobytes	00000000	0x0000000000000000																																																										
<input checked="" type="checkbox"/> Copy PF0																																																																	
PF1																																																																	
PF2																																																																	
PF3																																																																	

[SRIOV VF BARs] タブでは、仮想ファンクショングループ (VFG) 内のすべての仮想ファンクション (VF) のベース アドレス レジスタ (BAR) を設定できます。同じ VFG 内の VF はすべて、同じベース アドレス レジスタ (BAR) 設定を共有します。エンドポイント仮想ファンクションでは、最大 6 個の 32 ビット BAR または 3 個の 64 ビット BAR がサポートされます。仮想ファンクション BAR は、関連付けられている物理ファンクション BAR の設定に依存せずに設定できます。



重要: ファンクションおよびキューをサポートするため、DMA には大きな空間が必要です。デフォルトでは、DMA BAR には 64 ビットの BAR 空間が選択されています。これは PF および VF のどちらの BAR にも当てはまります。64 ビットまたは 32 ビットの BAR 空間を選択する前に、まずはデザイン ニーズを計算する必要があります。

BAR のオプションは選択可能です。デフォルトでは、DMA は BAR 0 (64 ビット)、AXI-Lite マスターは BAR 2 (64 ビット) です。これらのオプションはユーザーのニーズに応じて変更できます。

- [Bar]: チェック ボックスを使用して、該当する BAR を選択します。
- [Type]: 次の中から、該当するタイプを選択します。
 - [DMA]: BAR 0 空間に固定されています。
 - [AXI Lite Master]: BAR 1 空間に固定されています。
 - [AXI Bridge Master]: BAR 2 空間に固定されています。それ以外の BAR に関しては、[AXI Lite Master] または [AXI Bridge Master] を選択します。

注記: 現在の IP では、1 つの VF に対し、最大 1 つの DMA BAR (またはメールボックスだけが必要な場合は管理 BAR 1 つ) がサポートされています。AXI4-Lite バスを介して割り当てられたメモリ空間にアクセスするには、ほかの BAR を [AXI Lite Master] に設定できます。仮想ファンクション BAR では I/O 空間がサポートされていないので、適切なメモリ空間にマップする必要があります。

- [64-bit]:

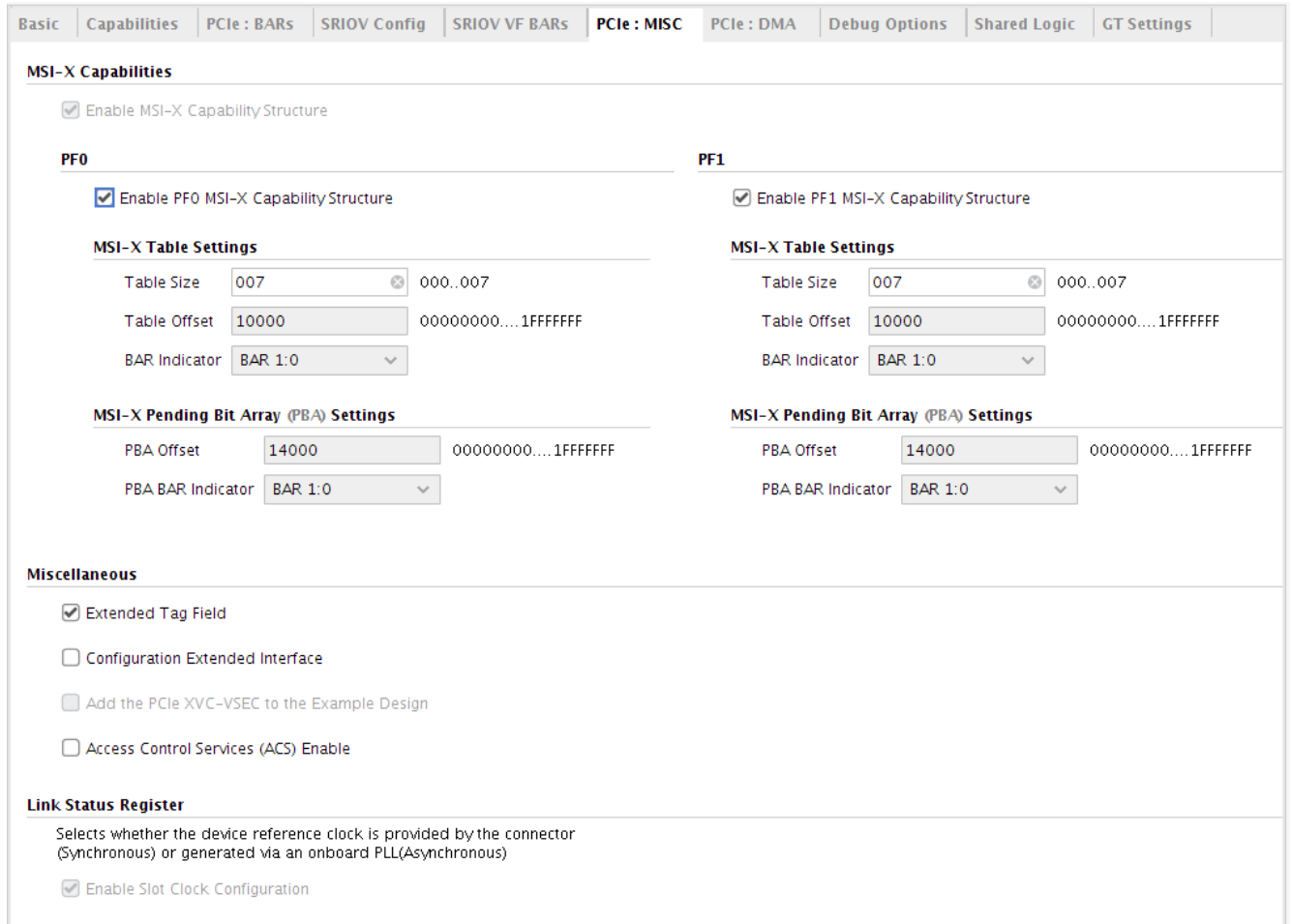
VF BAR は 64 ビットまたは 32 ビットに設定できます。デフォルトは 64 ビットです。

- DMA BAR には 64 ビットのアドレス指定がサポートされています。
- BAR が 64 ビットに設定されている場合、拡張アドレス空間用に次の BAR が使用されるので、その BAR にはアクセスできなくなります。
- VF BAR は、[Prefetchable] には設定できません。
- [Size]: 使用可能なサイズ範囲は、BAR が 32 ビットまたは 64 ビットに選択されているかによって左右されます。サポートされているページ サイズ フィールドには、SR-IOV 仕様で規定されている PF でサポートされているページ サイズがすべて示されます。このフィールドに基づき、VF BAR メモリ アドレスのマッピングに使用されるシステム ページ サイズ フィールドがシステム ソフトウェアによって設定されます。各 VF BAR アドレスがシステム ページの境界に揃えられます。デフォルトでは、DMA は 16 K バイトです。VF でのキュー割り当てに基づいて、さらに空間を追加できます。
- [Value]: 現在の選択内容に基づいて BAR に値が割り当てられます。

[PCIe MISC] タブ

次の図に [PCIe MISC] タブを示します。

図 30: [PCIe MISC] タブ



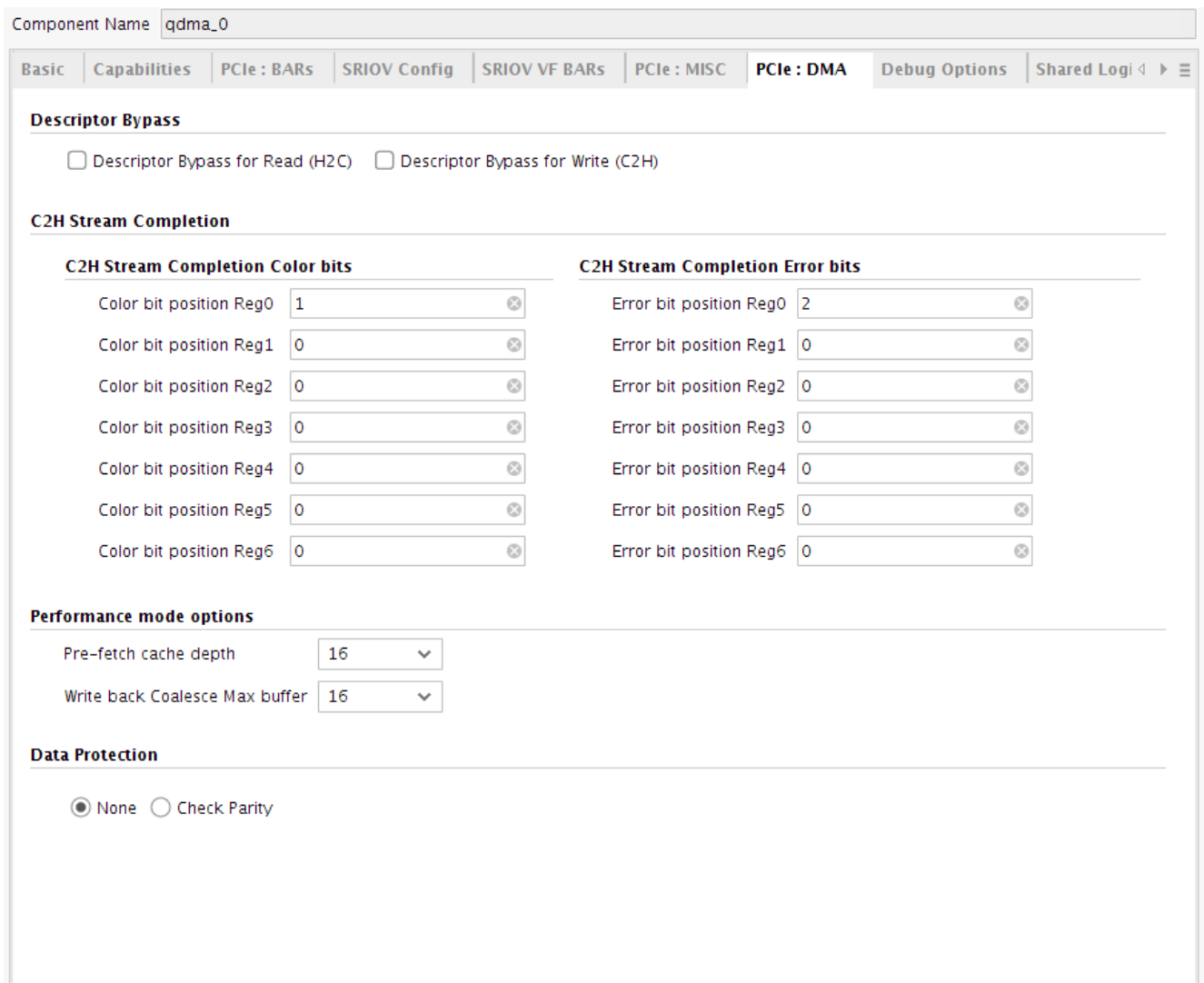
- [MSI-X Capabilities]: MSI-X がデフォルトでイネーブルになっています。必要に応じて異なる物理ファンクション用に MSI-X を設定できます。
- [MSI-X Table Settings]: MSI-X のテーブル構造を定義します。
 - [Table Size]: MSI-X テーブルのサイズを指定します。デフォルトは 8 です (ファンクションごとに割り込みベクターが 8 個)。より多くのベクターを 1 つのファンクションに追加することが可能です。サポートに関しては、ザイリンクスまでお問い合わせください。
 - [Table Offset]: MSI-X テーブルのファンクションをメモリ空間へマップするために使用される DMA コンフィギュレーション空間のベース アドレス レジスタ (BAR) からのオフセットを指定します。テーブル空間はオフセット 0x10000 で固定されています。
 - [BAR Indicator]: DMA コンフィギュレーション BAR に固定されています。
- [MSI-X Pending Bit Array Settings]:
 - [PBA Offset]: MSI-X PDB のベースをポイントする DMA BAR レジスタからのオフセットを指定します。テーブル空間はオフセット 0x14000 で固定されています。
 - [PBA BAR Indicator]: DMA コンフィギュレーション BAR に固定されています。

- [Extended Tag Field]: UltraScale+™ デバイスの場合、デフォルトで [Extended] タブのオプションが 256 個のタグになっています。[Extended] タブのオプションが選択されていない場合、DMA で 32 個のタグが使用されます。
- [Configuration Extended Interface]: PCIe 拡張インターフェイスは、コンフィギュレーション空間がさらに必要な場合に選択できます。[Configuration Extended Interface] が選択されている場合は、インターフェイスが正しく機能するように、それを拡張するためのロジックをユーザーが追加してください。
- [Access Control Server (ACS) Enable]: ACS がデフォルトで選択されています。

[PCIe DMA] タブ

次の図は [PCIe DMA] タブです。

図 31: [PCIe DMA] タブ



Component Name: qdma_0

Basic | Capabilities | PCIe : BARs | SRIOV Config | SRIOV VF BARs | PCIe : MISC | **PCIe : DMA** | Debug Options | Shared Log |

Descriptor Bypass

Descriptor Bypass for Read (H2C) Descriptor Bypass for Write (C2H)

C2H Stream Completion

C2H Stream Completion Color bits		C2H Stream Completion Error bits	
Color bit position Reg0	1	Error bit position Reg0	2
Color bit position Reg1	0	Error bit position Reg1	0
Color bit position Reg2	0	Error bit position Reg2	0
Color bit position Reg3	0	Error bit position Reg3	0
Color bit position Reg4	0	Error bit position Reg4	0
Color bit position Reg5	0	Error bit position Reg5	0
Color bit position Reg6	0	Error bit position Reg6	0

Performance mode options

Pre-fetch cache depth: 16

Write back Coalesce Max buffer: 16

Data Protection

None Check Parity

- [Descriptor Bypass for Read (H2C)]: H2C 転送用にディスクリプター バイパス出力および入力ポートをイネーブルにします。ディスクリプターが送信されるかどうかはコンテキスト設定で決まります。

- [Descriptor Bypass for Write (C2H)]: C2H 転送用にディスクリプター バイパス出力および入力ポートをイネーブにします。ディスクリプターが送信されるかどうかはコンテキスト設定で決まります。
- [C2H Stream Completion]:
 - [C2H Stream Completion Color bits]: 完了エントリの完了カラー ビット位置。ビット 0 から 511 まで (64 バイトの完了の場合) をプログラムするのに 7 つのレジスタがあります。これらのビットをプログラムして、BIT ファイルを生成できます。DMA 転送中、入力ピン `s_axis_c2h_cmpt_ctrl_color_idx[2:0]` により、どのカラー ビット位置を使用できるかが決まります。デフォルトのビット位置 1 はレジスタ 0 で選択されま
 - [C2H Stream Completion Error bits]: 完了エントリの完了エラー ビット位置。ビット 0 から 511 まで (64 バイトの完了の場合) をプログラムするのに 7 つのレジスタがあります。これらのビットをプログラムして、BIT ファイルを生成できます。DMA 転送中、入力ピン `s_axis_c2h_cmpt_ctrl_err_idx[2:0]` により、どのエラー ビット位置を使用できるかが決まります。デフォルトのビット位置 2 はレジスタ 0 で選択されます。
- [Performance mode options]:
 - [Pre-fetch cache depth]: プリフェッチ キャッシュでは、最大 64 個のキューがサポートされ、8、16、32、64 のいずれかを選択します (デフォルトは 16)。プリフェッチ キャッシュでは、常に多くのアクティブ キューがサポートできるようになっています。アクティブ キューがフェッチを完了し、そのキューのパケットのディスクリプターをすべて送信すると、ほかのアクティブ キューのためにキャッシュ エントリが解放されます。キャッシュ サイズが大きいと、より多くのキューをサポートできますが、そうするとエリアも増大します。
 - [CMPT Coalesce Max buffer]: CMPT 連結最大バッファでは、最大 64 個のバッファがサポートされ、8、16、32、64 のいずれかを選択します (デフォルトは 16)。CMPT 結合バッファの各エントリは、帯域幅の使用率を改善するため、ホストに書き込む前に複数の完了 (64B まで) を結合して 接続元 つのキューにします。CMPT 結合バッファがより深いと、複数のキューをさらに結合できますが、一方でエリアが増えてしまうのがマイナスです。
- [Data Protection]: パリティ チェックに関する設定です。デフォルトは [None] で、チェックは実行されません。[Check Parity] をオンにすると、PCIe からの読み出しデータで QDMA Subsystem for PCIe によりパリティがチェックされ、PCIe にデータを書き込むためのパリティが生成されます。

ユーザー パラメーター

このセクションは、このサブシステムには適用されません。

出力ファイルの生成

詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』 (UG896: [英語版](#)、[日本語版](#)) を参照してください。

サブシステムへの制約

必須の制約

QDMA Subsystem for PCIe には、PCI Express® 用の特定パフォーマンス要件を満たすため、タイミングなどの物理的なインプリメンテーション制約を指定する必要があります。これらの制約は、ザイリンクス デザイン制約 (XDC) ファイルで提供されています。生成される XDC ファイルにあるピン配置および階層名は、配布されているサンプル デザインに対応しています。



重要: サンプル デザインの最上位ファイルを使用しない場合は、基準クロック用に IBUFDS_GTE4 インスタンス、`sys_rst` 用に IBUF インスタンス、さらにこれらに関連付けられているタイミング制約をローカル デザインの最上位にコピーします。

一貫したインプリメンテーション結果を得るには、ザイリンクス ツールでデザインを実行するときに、変更されていない元の制約を含んだ XDC ファイルを使用する必要があります。XDC または特定の制約の定義および使用方法は、『Vivado Design Suite ユーザー ガイド: 制約の使用』(UG903: [英語版](#)、[日本語版](#)) を参照してください。

PCIe の統合ブロック ソリューションで提供されている制約は、ハードウェア上でテスト済みで、一貫した結果を提供します。制約は変更可能ですが、各制約をよく理解した上で変更してください。また、提供されている制約を規定から逸脱した形で使用したデザインに対してはサポートを提供していません。

デバイス、パッケージ、スピード グレードの選択

XDC のデバイス選択セクションには、ターゲット パーツ、パッケージ、スピード グレードが記述されており、この情報がインプリメンテーション ツールに渡されます。

このセクションには常にパーツを選択する行が含まれていますが、パーツまたはパッケージ別のオプションも含まれている場合があります。たとえば、次のような行です。

```
CONFIG PART = xcvu9p-flgb2104-2-i
```

クロック周波数

クロック要件の詳細は、『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』(PG213: [英語版](#)、[日本語版](#)) を参照してください。

クロック管理

クロック要件の詳細は、『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』(PG213: [英語版](#)、[日本語版](#)) を参照してください。

クロック配置

クロック要件の詳細は、『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』(PG213: [英語版](#)、[日本語版](#)) を参照してください。

バンク設定

このセクションは、この IP サブシステムには適用されません。

トランシーバーの配置

このセクションは、この IP サブシステムには適用されません。

I/O 規格と配置

このセクションは、この IP サブシステムには適用されません。

統合ブロック コアの移動

IP コア レベルの制約により、ブロック RAM、トランシーバー、および PCIe ブロックが推奨ロケーションにデフォルトでロックされています。これらのブロックを移動するには、XDC 制約ファイルでこれらブロックの制約を上書きする必要があります。上書きするには、次の手順に従います。

1. コア レベルの XDC 制約ファイルから上書きする必要のあるブロックの制約をコピーします。
2. それらの制約をユーザー XDC 制約ファイルに貼り付けます。
3. その制約に新しいロケーションを指定し、アップデートします。

ユーザー XDC 制約ファイルは一般的にデザインの最上位にスコープされるため、制約によって参照されるセルをコピー アンド ペーストしても、なお有効であることを確認します。通常はモジュール パスにフル階層名を使用してアップデートする必要があります。

注記: 入れ替える必要のあるロケーションが存在する場合 (つまり、別のモジュールが新しいロケーションを使用している場合)、次の 2 つの方法でロケーションを入れ替えることができます。

- 一時的なロケーションとして使用できる場所がある場合は、最初のモジュールを新しいそこへ移動します。その後、2 つ目のモジュールを最初のモジュールが使用していたロケーションに移動します。次に、最初のモジュールを 2 つ目のモジュールのロケーションに移動します。上記の手順は、XDC 制約ファイルで実行できます。
- 一時的なロケーションとして使用できる場所がほかにない場合は、2 つ目のモジュールをこのロケーションに移動させる前に、Tcl コマンド ウィンドウで最初のモジュールに `reset_property` コマンドを使用します。
`reset_property` コマンドは XDC 制約ファイルで実行できないので、Tcl コマンド ファイルから呼び出すか、Tcl コンソールに直接入力します。

シミュレーション

Vivado® シミュレーション コンポーネントについて、またサポートされているサードパーティ ツールについては、『Vivado Design Suite ユーザー ガイド: ロジック シミュレーション』 (UG900: [英語版](#)、[日本語版](#)) を参照してください。

基本的なシミュレーション

AXI-MM および AXI-ST のシミュレーション モデルは生成およびシミュレーションできます。この単純なシミュレーション モデルのオプションを使用すれば、複雑なデザインを開発しやすくなります。

AXI-MM モード

AXI4 メモリ マップド (AXI-MM) モードのサンプル デザインには、ユーザー サイドに 512 KB のブロック RAM があり、データをそのブロック RAM に書き込むことができ、また、ブロック RAM からホストへと読み出すことができます。

H2C 転送開始後、DMA がホストメモリからデータを読み出して、ブロック RAM に書き込みます。転送が完了したら、DMA はライトバック ステータスをアップデートし、割り込みを生成します (イネーブルになっている場合)。次に、C2H 転送が開始すると、DMA がブロック RAM からデータを読み出し、ホスト メモリに書き込みます。元のデータは、C2H 書き込みデータと比較されます。H2C および C2H はそれぞれに 1 つのディスクリプターを使用して設定されるので、合計の転送サイズは 128 バイトになります。

詳細手順は、[リファレンス ソフトウェア ドライバーのフロー](#) を参照してください。

AXI-ST モード

AXI4-Stream (AXI_ST) モードのサンプル デザインには、H2C 転送からのデータをチェックするデータ チェッカーと、C2H 転送用のデータ ジェネレーターがあります。

H2C 転送開始後、DMA がホストメモリからデータを読み出して、ユーザー サイドに書き込みます。転送が完了したら、DMA はライトバック ステータスをアップデートし、割り込みを生成します (イネーブルになっている場合)。ユーザー サイドのデータ チェッカーは、あらかじめ定義されているデータがあるかどうかを確認し、その結果は、ユーザーが読み出す、定義済みのアドレスにポストされます。

C2H 転送開始後、データ ジェネレーターはあらかじめ定義されているデータおよび関連付けられている制御信号を生成し、それらを DMA に送信します。DMA はデータをホストに転送し、完了 (CMPT) リングのエントリ/ステータスをアップデートし、割り込みを生成します (イネーブルになっている場合)。

H2C および C2H はそれぞれに 1 つのディスクリプターを使用して設定されるので、合計の転送サイズは 128 バイトになります。

詳細手順は、[リファレンス ソフトウェア ドライバーのフロー](#)を参照してください。

PIPE モード シミュレーション

QDMA Subsystem for PCIe は、コアの PIPE インターフェイスがリンク パートナーの PIPE インターフェイスに接続される PIPE モード シミュレーションをサポートします。このモードを選択すると、シミュレーション スピードが加速します。

現在の Vivado® Design Suite ソリューションのサンプル デザイン (エンドポイント モードまたはルート ポート モードのいずれか) で PIPE モード シミュレーションを有効にするには、[Customize IP] ダイアログ ボックスの [Basic] タブで [Enable PIPE Simulation] オプションを使用します。コアの境界で外部 PIPE インターフェイス信号が生成され、外部デバイスへのアクセスが可能になります。この機能を有効にすると、サンプル デザインで提供されるルート ポート モデルの代わりにサードパーティの PCI Express® VIP/BFM を使用するために必要な接続も提供されます。

次の表は、コアの最上位に含まれる PIPE バス信号と、EP コア (pcie_top) 内でマップされている PIPE 信号を示します。

表 315: 入力コマンドおよびエンドポイントの PIPE 信号のマップ

入力コマンド	エンドポイント PIPE 信号のマップ
common_commands_in[25:0]	使用されていない

表 316: 出力コマンドおよびエンドポイントの PIPE 信号のマップ

出力コマンド	エンドポイント PIPE 信号のマップ
common_commands_out[0]	pipe_clk ¹
common_commands_out[2:1]	pipe_tx_rate_gt ²
common_commands_out[3]	pipe_tx_rcvr_det_gt
common_commands_out[6:4]	pipe_tx_margin_gt
common_commands_out[7]	pipe_tx_swing_gt
common_commands_out[8]	pipe_tx_reset_gt
common_commands_out[9]	pipe_tx_deemph_gt

表 316: 出力コマンドおよびエンドポイントの PIPE 信号のマップ (続き)

出力コマンド	エンドポイント PIPE 信号のマップ
common_commands_out[16:10]	使用されていない ³

注記:

1. pipe_clk は、コアの設定に基づく出力クロックです。Gen1 レートの場合、pipe_clk は 125 MHz、Gen2 および Gen3 の場合は 250 MHz です。
2. pipe_tx_rate_gt は、パイプレート (2'b00-Gen1、2'b01-Gen2、および 2'b10-Gen3) を示します。
3. このポートの機能は廃止されており、未接続のままでもかまいません。

表 317: 入力バスおよびエンドポイントの PIPE 信号のマップ

入力バス	エンドポイントの PIPE 信号のマップ
pipe_rx_0_sigs[31:0]	pipe_rx0_data_gt
pipe_rx_0_sigs[33:32]	pipe_rx0_char_is_k_gt
pipe_rx_0_sigs[34]	pipe_rx0_elec_idle_gt
pipe_rx_0_sigs[35]	pipe_rx0_data_valid_gt
pipe_rx_0_sigs[36]	pipe_rx0_start_block_gt
pipe_rx_0_sigs[38:37]	pipe_rx0_syncheader_gt
pipe_rx_0_sigs[83:39]	使用しない

表 318: 出力バスおよびエンドポイントの PIPE 信号のマップ

出力バス	エンドポイントの PIPE 信号のマップ
pipe_tx_0_sigs[31: 0]	pipe_tx0_data_gt
pipe_tx_0_sigs[33:32]	pipe_tx0_char_is_k_gt
pipe_tx_0_sigs[34]	pipe_tx0_elec_idle_gt
pipe_tx_0_sigs[35]	pipe_tx0_data_valid_gt
pipe_tx_0_sigs[36]	pipe_tx0_start_block_gt
pipe_tx_0_sigs[38:37]	pipe_tx0_syncheader_gt
pipe_tx_0_sigs[39]	pipe_tx0_polarity_gt
pipe_tx_0_sigs[41:40]	pipe_tx0_powerdown_gt
pipe_tx_0_sigs[69:42]	使用されていない ¹

注記:

1. このポートの機能は廃止されており、未接続のままでもかまいません。

合成およびインプリメンテーション

合成およびインプリメンテーションの詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』 (UG896: [英語版](#)、[日本語版](#)) を参照してください。

サンプル デザイン

この章では、Vivado® Design Suite で提供されている 5 つのサンプル デザインについて説明します。

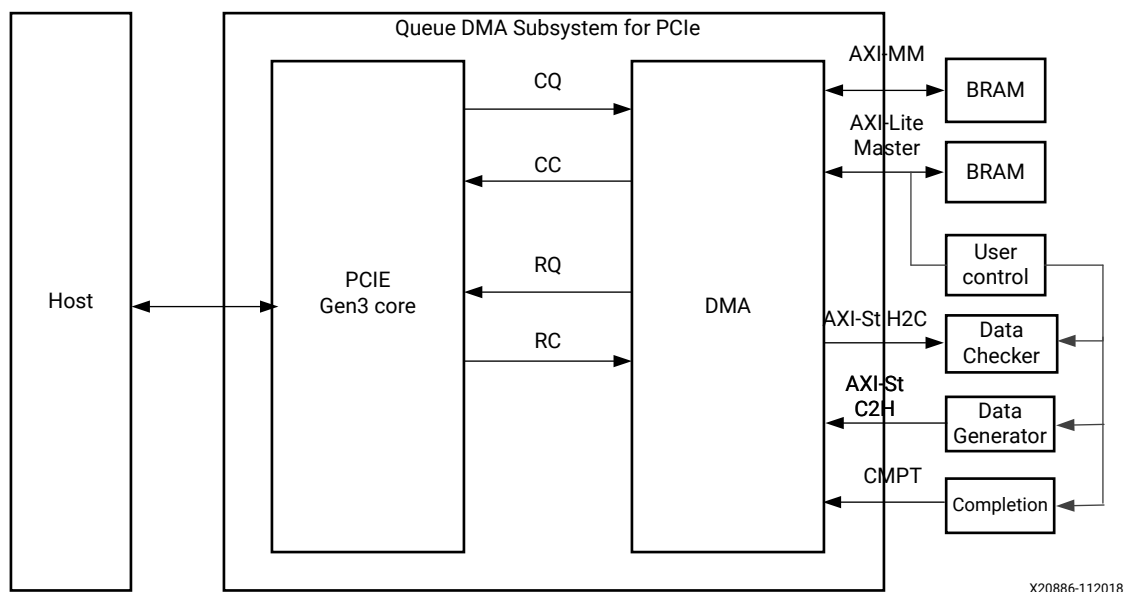
- AXI4 メモリ マップドおよび AXI-Stream (完了付き)
- AXI メモリ マップド
- AXI メモリ マップド (完了付き)
- AXI-Stream (完了付き)
- ディスクリプター バイパス入力/出力ループバック

上記のオプション用に生成されたサンプル デザインはテスト専用です。これらのサンプル デザインは随時変更できます。

完了のある AXI4 メモリ マップドおよび AXI4-Stream のデフォルト サンプル デザイン

次のサンプル デザインは、DMA インターフェイスの選択オプションが [Basic] タブで [AXI4 Memory Mapped and AXI4-Stream with Completion] に設定された状態で生成されています。

図 32: デフォルトのサンプル デザイン



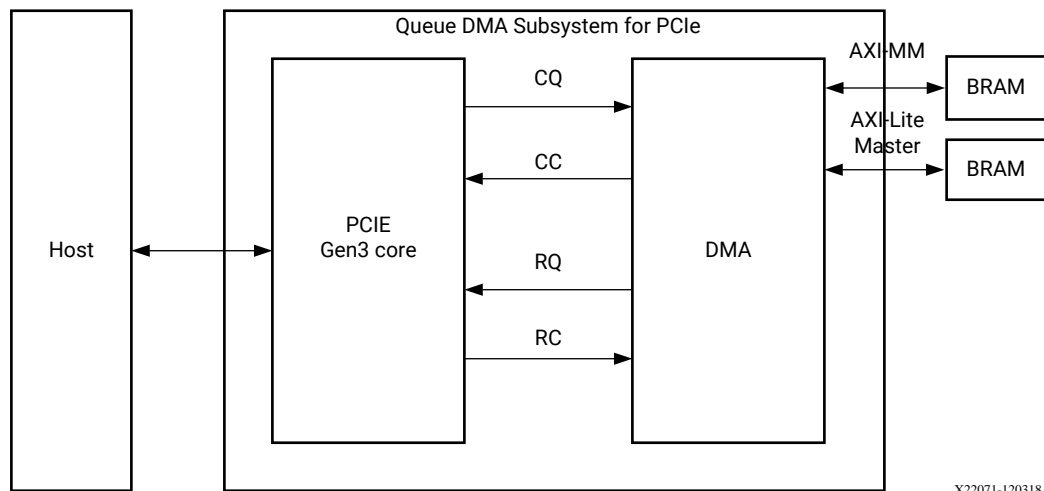
生成されたサンプル デザインは、AXI4 メモリ マップドおよび AXI4-Stream インターフェイスを使用するためのブロックを提供します。

- AXI-MM インターフェイスは 512 KB のブロック RAM に接続されます。
- AXI4-Stream インターフェイスは、カスタム データ ジェネレーター、データ チェッカー モジュールに接続されません。
- CMPT インターフェイスは、完了ブロック ジェネレーターに接続されます。
- データ ジェネレーターおよびチェッカーは、あらかじめ定義された、0 から開始する 16 ビットのインクリメンタルパターンでのみ機能します。このデータ ファイルは、ドライバー パッケージに含まれています。

パターン ジェネレーターおよびチェッカーは、[サンプル デザインのレジスタ](#)にあるレジスタを使用して制御できません。これらのレジスタは、AXI4-Lite マスター インターフェイスを介してのみ制御できます。QDMA Subsystem for PCIe の AXI4-Stream インターフェイスをテストするには、AXI4-Lite マスター インターフェイスがあることを確認します。

AXI メモリ マップド サンプル デザイン

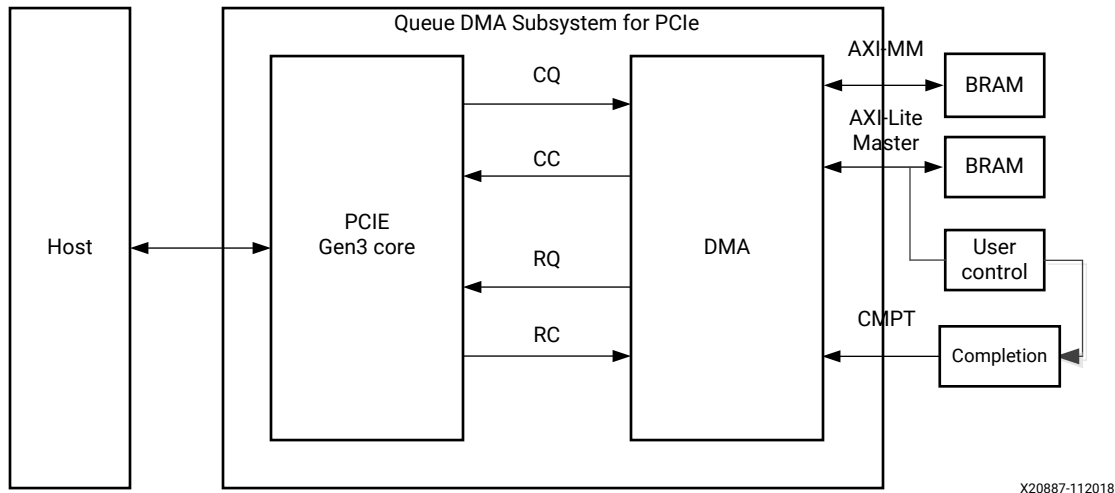
図 33: AXI4 メモリ マップド サンプル デザイン



上記のサンプル デザインは、DMA インターフェイスの選択オプションが [Basic] タブで [AXI-Memory Mapped only] に設定された状態で生成されています。このモードでは、AXI4-MM インターフェイスは 512 KB のブロック RAM に接続されます。上記の図では、AXI4-Lite マスターが 4 KB のブロック RAM に接続されています。H2C 転送の場合、DMA がホストからデータを読み出して、ブロック RAM に書き込みます。C2H 転送の場合、DMA がブロック RAM からデータを読み出して、ホストに書き込みます。

完了のある AXI メモリ マップドのサンプル デザイン

図 34: 完了のある AXI4 メモリ マップドのサンプル デザイン

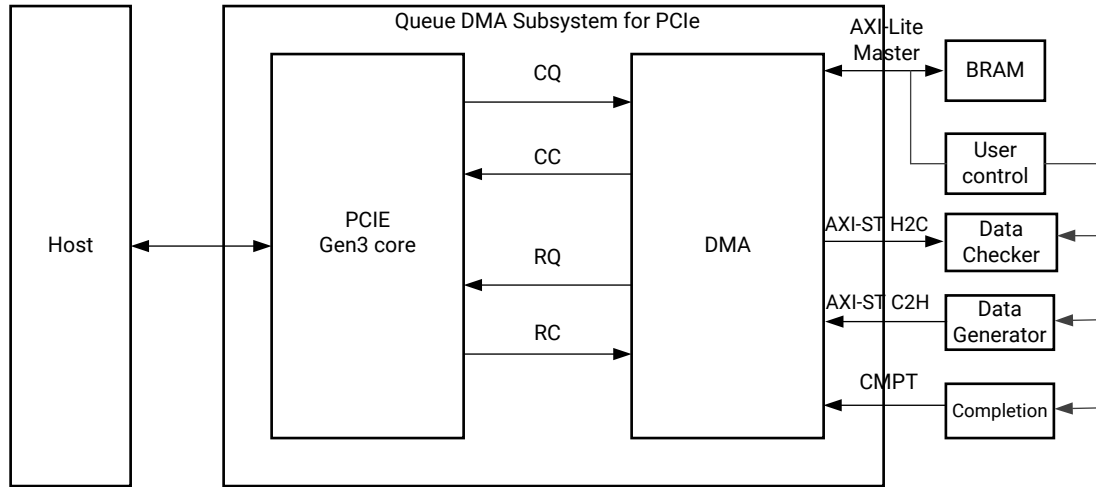


上記のサンプル デザインは、DMA インターフェイスの選択オプションが [Basic] タブで [AXI-MM with Completion] に設定された状態で生成されています。このモードでは、AXI-MM インターフェイスは 512 KB のブロック RAM に、CMPT インターフェイスは完了ジェネレーター ブロックに接続されます。上記の図では、AXI-Lite マスターが 4 KB のブロック RAM およびユーザー制御ロジックに接続されています。H2C 転送の場合、DMA がホストからデータを読み出して、ブロック RAM に書き込みます。C2H 転送の場合、DMA がブロック RAM からデータを読み出して、ホストに書き込みます。

完了ブロックは、[サンプル デザインのレジスタ](#)にあるレジスタを使用して制御できます。

完了のある AXI ストリームのサンプル デザイン

図 35: AXI4-Stream のサンプル デザイン



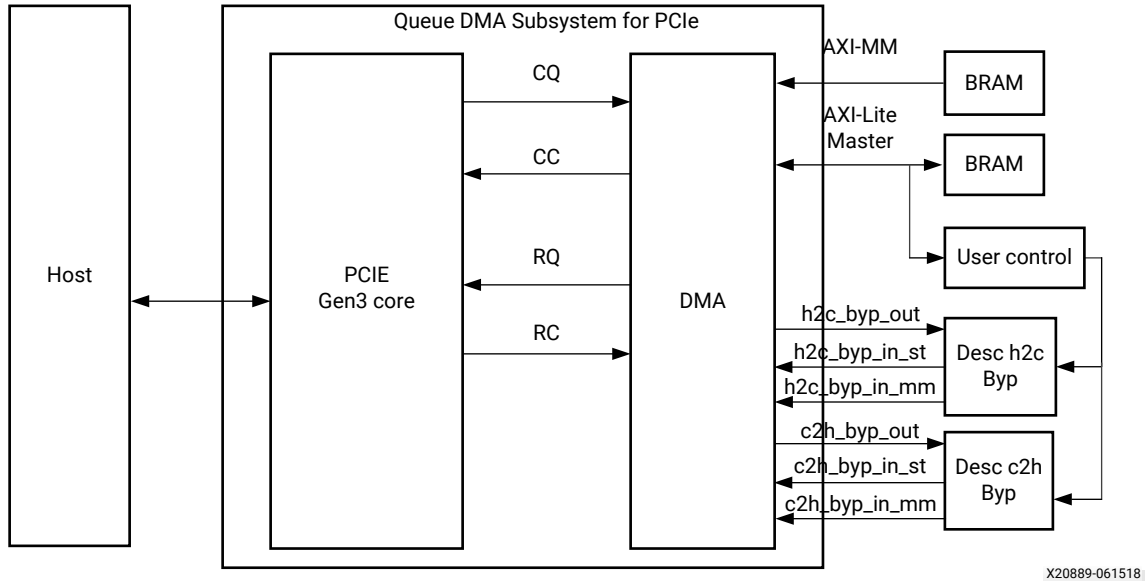
X20888-120718

上記のサンプル デザインは、DMA インターフェイスの選択オプションが [Basic] タブで [AXI Stream with Completion] に設定された状態で生成されています。このモードでは、AXI-ST H2C インターフェイスはデータ チェッカーに、AXI-ST C2H インターフェイスはデータ ジェネレーターに、CMPT インターフェイスは完了ジェネレーターモジュールにそれぞれ接続されます。上記の図では、AXI-Lite マスターが 4 KB のブロック RAM およびユーザー制御ロジックに接続されています。ソフトウェアは、AXI4-Lite マスター インターフェイスを介して、データ チェッカーおよびデータ ジェネレーターを制御できます。データ ジェネレーターおよびチェッカーは、あらかじめ定義された、0 から開始する 16 ビットのインクリメンタル パターンでのみ機能します。このデータ ファイルは、ドライバー パッケージに含まれています。

パターン ジェネレーターおよびチェッカーは、[サンプル デザインのレジスタ](#)にあるレジスタを使用して制御できません。

ディスクリプター バイパス入力/出カループバックのあるサンプル デザイン

図 36: AXI4 メモリ マップドおよびディスクリプター バイパスのサンプル デザイン



上記のサンプル デザインは、[PCIe DMA] タブの [Descriptor Bypass for Read (H2C)] および [Descriptor Bypass for Write (C2H)] のオプションが選択されていると、生成されます。これらのオプションは、[Basic] タブにある次の DMA インターフェイス オプションと共に選択可能です。

- [AXI Memory Mapped and AXI Stream with Completion]
- [AXI Memory Mapped only]
- [AXI Stream with Completion]
- [AXI Memory Mapped with Completion]

ディスクリプター バイパスの入力/出カループバックは、サンプル デザイン レジスタの [DESCRIPTOR_BYPASS \(0x090\)](#) ビット 0 および 1 に書き込みを実行することで AXI4-Lite マスターにより制御されます。

ディスクリプター バイパス出力をイネーブルにするには、正しいコンテキストのプログラミングが必要です。詳細は、[コンテキスト プログラミング](#) を参照してください。

サンプル デザインのレジスタ

表 319: サンプル デザインのレジスタ

レジスタ	アドレス	説明
C2H_ST_QID (0x000)	0x000	AXI-ST C2H キュー ID

表 319: サンプル デザインのレジスタ (続き)

レジスタ	アドレス	説明
C2H_ST_LEN (0x004)	0x004	AXI-ST C2H 転送の長さ
C2H_CONTROL_REG (0x008)	0x008	AXI-ST C2H パターン ジェネレーター制御
H2C_CONTROL_REG (0x00C)	0x00C	AXI-ST H2C 制御
H2C_STATUS (0x010)	0x010	AXI-ST H2C ステータス
C2H_PACKET_COUNT (0x020)	0x020	AXI-ST C2H 転送するパケットの数
C2H_COMPLETION_DATA_0 (0x030) から C2H_COMPLETION_DATA_7 (0x04C)	0x4C-0x030	AXI-ST C2H 完了データ
C2H_COMPLETION_SIZE (0x050)	0x050	AXI-ST 完了データ型
SCRATCH_REG0 (0x060)	0x060	スクラッチ レジスタ 0
SCRATCH_REG1 (0x064)	0x064	スクラッチ レジスタ 1
C2H_PACKETS_DROP (0x088)	0x088	破棄された AXI-ST C2H パケットの数
C2H_PACKETS_ACCEPTED (0x08C)	0x08C	受信された AXI-ST C2H パケットの数
DESCRIPTOR_BYPASS (0x090)	0x090	C2H および H2C ディスクリプターのバイパス ループバック
USER_INTERRUPT (0x094)	0x094	ユーザー割り込み、ベクター番号、ファンクション番号
USER_INTERRUPT_MASK (0x098)	0x098	ユーザー割り込みマスク
USER_INTERRUPT_VECTOR (0x09C)	0x09C	ユーザー割り込みベクター
DMA_CONTROL (0x0A0)	0x0A0	DMA 制御
VDM_MESSAGE_READ (0x0A4)	0x0A4	読み出された VDM メッセージ

C2H_ST_QID (0x000)

表 320: C2H_ST_QID (0x000)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:11]	0	NA		予約
[10:0]	0	RW	c2h_st_qid	AXI4-Stream C2H キュー ID

C2H_ST_LEN (0x004)

表 321: C2H_ST_LEN (0x004)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:16]	0	NA		予約
[15:0]	0	RW	c2h_st_len	AXI4-Stream パケット長

C2H_CONTROL_REG (0x008)

表 322: C2H_CONTROL_REG (0x008)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:3]	0	NA		予約
[2]	0	RW		即値データ。 セットされている場合、データ ジェネレーターにより即値データが送信されます。このビットは自動クリアされます。1 を書き込むと転送が開始します。
[1]	0	RW		AXI-ST C2H 転送を開始します。このビットは自動クリアされます。1 を書き込むと転送が開始します。
[0]	0	RW		ストリーミング ループバック。セットされている場合、カード側の H2C ストリーミング ポートからのデータ パケットが C2H ストリーミング ポートにループバックされます。

標準の C2H ストリーム パケット転送の場合、アドレス オフセット 0x08 を 0x2 に設定します。

C2H 即値データ転送の場合、アドレス オフセット 0x8 を 0x4 に設定します。

C2H/H2C ストリーミング ループバックの場合、アドレス オフセット 0x8 を 0x1 に設定します。

H2C_CONTROL_REG (0x00C)

表 323: H2C_CONTROL_REG (0x00C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:12]	0	NA		予約
[11:10]	0	RW	c2h_st_at	c2h_byp_in_st_sim_at[1:0] および c2h_byp_in_csh_at[1:0] アドレス タイプ。 2'b00: リクエスト内のアドレスは変換されません。 2'b01: 予約。 2'b10: リクエスト内のアドレスは変換されます。 2'b11: 予約。
[9:8]	0	RW	c2h_mm_at	c2h_byp_in_mm_at[1:0] アドレス タイプ。 2'b00: リクエスト内のアドレスは変換されません。 2'b01: 予約。 2'b10: リクエスト内のアドレスは変換されます。 2'b11: 予約
[7:6]	0	RW	h2c_st_at	c2h_byp_in_st_at[1:0] アドレス タイプ。 2'b00: リクエスト内のアドレスは変換されません。 2'b01: 予約。 2'b10: リクエスト内のアドレスは変換されます。 2'b11: 予約。

表 323: H2C_CONTROL_REG (0x00C) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[5:4]	0	RW	h2c_mm_at	h2c_byp_in_mm_at[1:0] アドレス タイプ。 2'b00: リクエスト内のアドレスは変換されません。 2'b01: 予約。 2'b10: リクエスト内のアドレスは変換されます。 2'b11: 予約。
[0]	0	RW		H2C 転送の一致ビットをクリアします。

H2C_STATUS (0x010)

表 324: H2C_STATUS (0x010)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:15]	0	NA		予約
[14:4]	0	R		H2C 転送キュー ID
[3:1]	0	NA		予約
[0]	0	R		H2C 転送一致

C2H_PACKET_COUNT (0x020)

表 325: C2H_PACKET_COUNT (0x020)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:10]	0	NA		予約
[9:0]	0	RW		AXI-ST C2H の転送するパケット数

C2H_COMPLETION_DATA_0 (0x030)

表 326: C2H_COMPLETION_DATA_0 (0x030)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	NA		AXI-ST C2H 完了データ [31:0]

C2H_COMPLETION_DATA_1 (0x034)

表 327: C2H_COMPLETION_DATA_1 (0x034)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	NA		AXI-ST C2H 完了データ [63:32]

C2H_COMPLETION_DATA_2 (0x038)

表 328: C2H_COMPLETION_DATA_2 (0x038)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	NA		AXI-ST C2H 完了データ [95:64]

C2H_COMPLETION_DATA_3 (0x03C)

表 329: C2H_COMPLETION_DATA_3 (0x03C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	NA		AXI-ST C2H 完了データ [127:96]

C2H_COMPLETION_DATA_4 (0x040)

表 330: C2H_COMPLETION_DATA_4 (0x040)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	NA		AXI-ST C2H 完了データ [159:128]

C2H_COMPLETION_DATA_5 (0x044)

表 331: C2H_COMPLETION_DATA_5 (0x044)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	NA		AXI-ST C2H 完了データ [191:160]

C2H_COMPLETION_DATA_6 (0x048)

表 332: C2H_COMPLETION_DATA_6 (0x048)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	NA		AXI-ST C2H 完了データ [223:192]

C2H_COMPLETION_DATA_7 (0x04C)

表 333: C2H_COMPLETION_DATA_7 (0x04C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	NA		AXI-ST C2H 完了データ [255:224]

C2H_COMPLETION_SIZE (0x050)

表 334: C2H_COMPLETION_SIZE (0x050)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:13]	0	NA		予約
-12]	0	RW		完了タイプ。 1'b1: NO_PLD_BUT_WAIT 1'b0: HAS PLD 詳細は、 AXI4-Stream C2H 完了ポート を参照してください。
[10:8]	0	RW		s_axis_c2h_cmpt_ctrl_err_idx[2:0] 完了エラー ビット インデックス。 3'b000: 0 番目のレジスタを選択します。 3'b111: エラー ビットがレポートされません。
[6:4]	0	RW		s_axis_c2h_cmpt_ctrl_col_idx[2:0] 完了エラー ビット インデックス。 3'b000: 0 番目のレジスタを選択します。 3'b111: カラー ビットがレポートされません。
[3]	0	RW		s_axis_c2h_cmpt_ctrl_user_trig 完了ユーザー トリガー
[1:0]	0	RW		AXI4-Stream C2H 完了データ サイズ。 00: 8 バイト 01: 16 バイト 10: 32 バイト 11: 64 バイト

SCRATCH_REG0 (0x060)

表 335: SCRATCH_REG0 (0x060)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW		スクラッチ レジスタ

SCRATCH_REG1 (0x064)

表 336: SCRATCH_REG1 (0x064)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW		スクラッチ レジスタ

C2H_PACKETS_DROP (0x088)

表 337: C2H_PACKETS_DROP (0x088)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	R		転送ごとに破棄された AXI-ST C2H パケット (ディスクリプター) の数

各 AXI-ST C2H 転送には、転送サイズおよび C2H バッファ サイズによりますが、1 つ以上のディスクリプターが含まれます。このレジスタは、現在の転送で破棄されたディスクリプターの数を表示します。このレジスタは、転送開始時に 0 にリセットされます。

C2H_PACKETS_ACCEPTED (0x08C)

表 338: C2H_PACKETS_ACCEPTED (0x08C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	R		転送ごとに受領された AXI-ST C2H パケット (ディスクリプター) の数

各 AXI-ST C2H 転送には、転送サイズおよび C2H バッファ サイズによりますが、1 つ以上のディスクリプターが含まれます。このレジスタは、現在の転送で受領されたディスクリプターの数を示します。このレジスタは、転送開始時に 0 にリセットされます。

DESCRIPTOR_BYPASS (0x090)

表 339: ディスクリプター バイパス (0x090)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:3]	0	NA		予約

表 339: ディスクリプター バイパス (0x090) (続き)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[2:1]	0	RW	c2h_dsc_bypass	C2H ディスクリプター バイパス ループバック。セットされている場合、C2H ディスクリプターのバイパス出力ポートは、C2H ディスクリプターのバイパス入力ポートにループバックされます。 2'b00: バイパス ループバックなし。 2'b01: C2H MM ディスクリプター バイパス ループバックおよび C2H ストリーム キャッシュ バイパスループバック。 2'b10: C2H ストリームの単純ディスクリプター バイパスループバック。 2'b11: H2C ストリームの 64 バイト ディスクリプターは完了インターフェイスにループバックされます。
[0]	0	RW	h2c_dsc_bypass	H2C ディスクリプター バイパス ループバック。セットされている場合、H2C ディスクリプターのバイパス出力ポートは、H2C ディスクリプターのバイパス入力ポートにループバックされます。 1'b1: H2C MM および H2C ストリームのディスクリプター バイパス ループバック 1'b0: ディスクリプター ループバックなし

USER_INTERRUPT (0x094)

表 340: ユーザー割り込み (0x094)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:20]	0	NA		予約
[19:12]	0	RW	usr_irq_in_fun	ユーザー割り込みファンクション番号
[11:9]	0	NA		予約
[8:4]	0	RW	usr_irq_in_vec	ユーザー割り込みベクター番号
[3:1]	0	NA		予約
[0]	0	RW	usr_irq	ユーザー割り込み。設定されていると、サンプル デザインによりユーザー割り込みが生成されます。

ユーザー割り込みを生成するには:

- ビット [19:12] にファンクション番号を書き込みます。これは、usr_irq_in_fnc ユーザー割り込みを生成するファンクションに対応します。
- ビット [8:4] に MSI-X ベクター番号を書き込みます。これは、usr_irq_in_vec ユーザー割り込み用に設定される MSI-X テーブルのエントリに対応します。
- ユーザー割り込みを生成するため、ビット [0] に 1 を書き込みます。このビットは、DMA からの usr_irq_out_ack が生成された後に、セルフ クリアします。

1 つの書き込みで、この 3 手順を同時に実行できます。

USER_INTERRUPT_MASK (0x098)

表 341: ユーザー割り込みマスク (0x098)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW		ユーザー割り込みマスク

USER_INTERRUPT_VECTOR (0x09C)

表 342: ユーザー割り込みベクター (0x09C)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:0]	0	RW		ユーザー割り込みベクター

`user_interrupt_mask[31:0]` および `user_interrupt_vector[31:0]` レジスタは、ファンクションのユーザー割り込みを生成できるユーザー割り込みアグリゲーションのサンプル デザインとして提供されています。
`user_interrupt_mask[31:0]` は `user_interrupt_vector[31:0]` と AND 接続されていて (ビットワイヤ)、ユーザー割り込みが生成されます。`user_interrupt_vector[31:0]` は読み出しレジスタでクリアです。

ユーザー割り込みを生成するには:

1. `user_interrupt[19:12]` にファンクション番号を書き込みます。これは、`usr_irq_in_fnc` ユーザー割り込みを生成するファンクションに対応します。
2. `user_interrupt[8:4]` に MSI-X ベクターを書き込みます。これは、`usr_irq_in_vec` ユーザー割り込み用に設定される MSI-X テーブルのエントリに対応します。
3. `user_interrupt_mask[31:0]` レジスタにマスク値を書き込みます。
4. `user_interrupt_vector[31:0]` レジスタに割り込みベクター値を書き込みます。

これで、DMA ブロックにユーザー割り込みが生成されます。

ユーザー割り込みの生成方法には、次の 2 つがあります。

- `user_interrupt[0]` に書き込みます。または、
- マスクを設定して `user_interrupt_vector[31:0]` レジスタに書き込みます。

DMA_CONTROL (0x0A0)

表 343: DMA 制御 (0x0A0)

ビット	デフォルト	アクセス タイプ	フィールド	説明
[31:1]		NA		予約
[0]	0	RW	gen_qdma_reset	<code>soft_reset</code> がセットされている場合は、DMA ブロックへのソフトリセットを生成します。このビットは、100 サイクル後にクリアになります。

DMA_control[0] に 1 が書き込まれると、soft_reset_n (アクティブ Low) でソフトリセットが生成されます。リセットが 100 サイクル間アサートされ、その後続く信号はディアサートされます。

VDM_MESSAGE_READ (0x0A4)

表 344: VDM メッセージ読み出し (0x0A4)

ビット	デフォルト	アクセスタイプ	フィールド	説明
[31:0]		RO		読み出された VDM メッセージ

ベンダー定義のメッセージ (VDM)、st_rx_msg_data、はサンプル デザインの FIFO に格納されます。このレジスタ (0x0A4) への読み出しは、1 回につき 32 ビットのメッセージ 1 つです。

アップグレード

v2.0 から v3.0 への変更点

QDMA Subsystem for PCIe v2.0 から v3.0 の変更リストは、アンサー [71737](#) を参照してください。

DMA/Bridge Subsystem for PCI Express との比較

次の表に、DMA/Bridge Subsystem for PCI Express® と QDMA Subsystem for PCI Express の違いを示します。

表 345: サブシステムの比較

	DMA/Bridge Subsystem	QDMA Subsystem
コンフィギュレーション	最高 Gen3x16 まで。	最高 Gen3x16 まで。
チャンネル/キュー	4 つの H2C チャンネル、4 つの C2H チャンネル (1PF)。	最高 2000 個のキューまで (すべて 1 PF に割り当て可能、または 4 チャンネル間に分散可能)。
SR-IOV	サポートされていません。	サポートされています (4 個の PF および 252 個の VF をサポート)。
ユーザー インターフェイス	AXI-MM または AXI-ST を使用してコンフィギュレーション可能 (ただし両方は使用できない)。	各キューのコンテキストから、AXI4 メモリに行くのか、AXI4-Stream に行くのかは判断可能。
ユーザー割り込み	最大 16 個のユーザー割り込み。	ファンクションごとに割り込みアグリゲーション。
デバイス サポート	7 シリーズ Gen2 から UltraScale+™ デバイスのサポート。	UltraScale+ デバイスのみのサポート。
割り込み	レガシ、MSI、MSI-X をサポート。	PF の場合は MSI-X をサポート。 VF の場合は MSI-X のみをサポート。
ドライバー サポート	Linux、Windows サンプル ドライバー。	Linux、DPDK、Windows。

デバッグ

この付録では、ザイリンクス サポート ウェブサイトより入手可能なリソースおよびデバッグ ツールについて説明します。

ザイリンクス ウェブサイト

サブシステムを使用した設計およびデバッグでヘルプが必要な場合は、[ザイリンクス サポート ウェブ ページ](#)から製品の資料、リリース ノート、アンサーなどを参照するか、テクニカル サポートでサービス リクエストを作成してください。また、[ザイリンクス コミュニティ フォーラム](#)もご活用ください。このフォーラムは、ザイリンクス ソリューションについてメンバーが学び、参加し、情報を共有し、質問できる場です。

資料

この製品ガイドは、サブシステムに関する主な資料です。このガイドおよび設計プロセスをサポートする各製品の関連資料はすべて、[ザイリンクス サポート ウェブ ページ](#)またはザイリンクス Documentation Navigator から入手できます。ザイリンクス Documentation Navigator は、[ダウンロード ページ](#)からダウンロードできます。このツールの詳細および機能は、インストール後にオンライン ヘルプを参照してください。

ソリューション センター

デバイス、ツール、IP のサポートについては、[ザイリンクス ソリューション センター](#)を参照してください。デザイン アシスタント、デザイン アドバイザリ、トラブルシューティングのヒントなどが含まれます。

QDMA Subsystem for PCIe コアに関するソリューション センターは、[PCI Express のソリューション センター](#)を参照してください。

アンサー

アンサーには、よく発生する問題の解決方法、ザイリンクス製品に関する既知の問題などの情報が記載されています。アンサーは、ユーザーが該当製品の最新情報にアクセスできるよう作成および管理されています。

このサブシステムに関するアンサーの検索には、[ザイリンクス サポート ウェブ ページ](#)にある検索ボックスを使用します。よりの確な検索結果を得るには、次のようなキーワードを使用してください。

- 製品名
- ツールに表示されるメッセージ
- 問題の概要

検索結果は、フィルター機能を使用してさらに絞り込むことができます。

このサブシステムに関するマスター アンサー

アンサー [70927](#).

テクニカル サポート

ザイリンクスでは、製品資料の説明に従って使用されている LogiCORE™ IP 製品に対して、テクニカルサポートを[ザイリンクス コミュニティ フォーラム](#)で提供しています。ただし、次に該当する場合、ザイリンクスではタイミング、機能、サポートは保証できません。

- 資料で定義されていないデバイスにソリューションをインプリメントした場合。
- 資料で定義されている許容範囲を超えてカスタマイズした場合。
- 「DO NOT MODIFY」とされているデザイン セクションに変更を加えた場合。

ザイリンクス テクニカル サポートへのお問い合わせは、[ザイリンクス サポート ウェブ ページ](#)を参照してください。

デバッグ ツール

QDMA Subsystem for PCIe デザインの問題を解決するためのツールは多数あります。どのツールがどの状況のデバッグに適しているのかを理解しておくことが重要です。

Vivado Design Suite のデバッグ機能

Vivado® Design Suite のデバッグ機能は、Logic Analyzer および Virtual I/O コアをユーザー デザインに直接挿入します。デバッグ機能を使用すると、トリガー条件を設定して、アプリケーションおよび統合ブロックのポート信号をハードウェアに取り込むことができます。取り込まれた信号は、その後解析できます。Vivado IDE のこの機能は、ザイリンクス デバイスで実行されるデザインの論理デバッグおよび検証に使用されます。

Vivado ロジック解析は、次の LogiCORE IP コアに使用されます。

- ILA 2.0 (およびそれ以降のバージョン)
- VIO 2.0 (およびそれ以降のバージョン)

『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908: [英語版](#)、[日本語版](#)) を参照してください。

ハードウェア デバッグ

ハードウェアの問題は、リンク立ち上げ時の問題から、テスト後に生じる問題までさまざまです。ここでは、一般的な問題のデバッグ手順を説明します。Vivado® のデバッグ機能は、ハードウェア デバッグに有益なリソースです。次の各セクションに示す信号を Vivado のデバッグ機能でプローブすることで、個々の問題をデバッグできます。

一般的なチェック

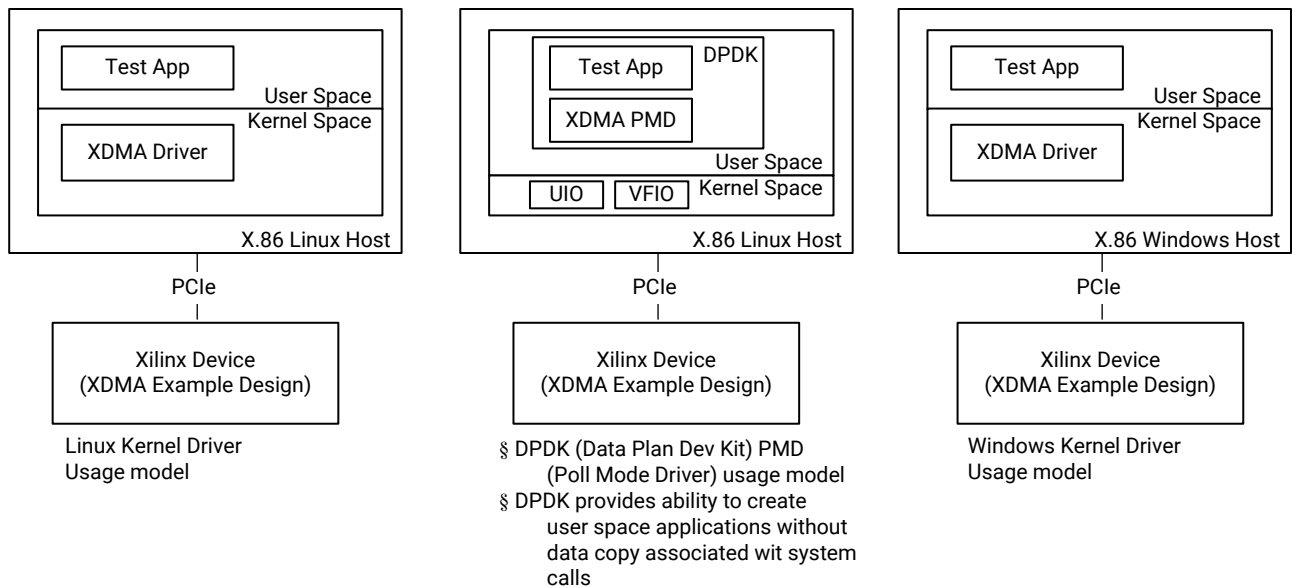
コアに対するタイミング制約がサンプル デザインからすべて適切に取り込まれていること、さらにインプリメンテーション時にこれらの制約がすべて満たされていることを確認します。

- 配置配線後のタイミング シミュレーションで正しく動作しているかを確認します。タイミング シミュレーションでは発生しない問題がハードウェアで発生する場合、PCB の問題である可能性があります。すべてのクロックソースがアクティブでノイズのないものであることを確認してください。
- デザインで MMCM を使用している場合、locked ポートをモニターして、すべての MMCM がロックしていることを確認します。
- 出力が 0 になった場合は、ライセンスを確認してください。

アプリケーションソフトウェア開発

デバイス ドライバー

図 37: デバイス ドライバー



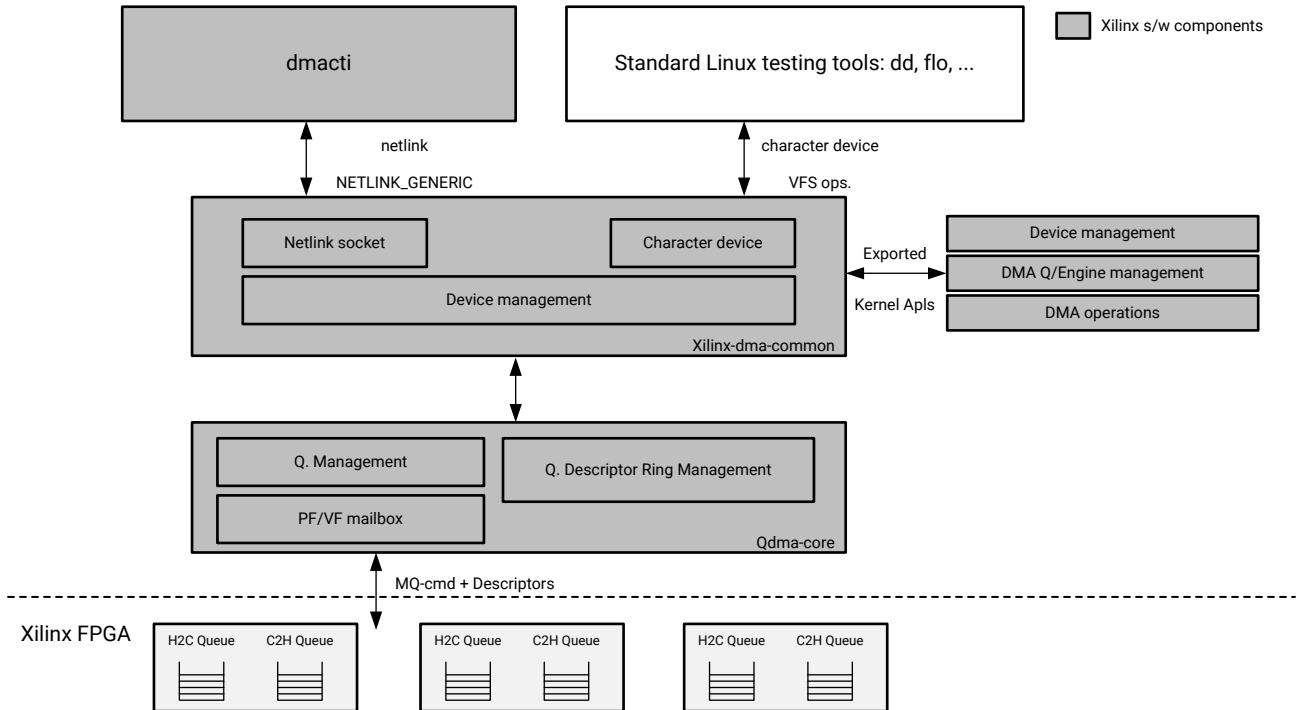
X20600-090418

上の図は、Linux および Windows の QDMA ソフトウェア ドライバーの使用モデルを表しています。この QDMA Subsystem for PCIe サンプル デザインは、PCI Express を介して X86 のホストに接続されているザイリンクス FPGA にインプリメントされます。

- 最初の使用モードでは、QDMA ドライバーは Linux のカーネル空間で実行しますが、テスト アプリケーションはユーザー空間で実行します。
- 2 番目の使用モードでは、ユーザー空間のみで実行する QDMA ポール モード ドライバー (PMD) を開発するため、また FPGA との通信に UIO および VFIO カーネル フレームワークを使用するために、データ プラン開発キット (DPDK) が使用されます。
- 3 番目の使用モードでは、QDMA ドライバーは Windows のカーネル空間で実行しますが、テスト アプリケーションはユーザー空間で実行します。

Linux DMA ソフトウェア アーキテクチャ (PF/VF)

図 38: Linux DMA ソフトウェア アーキテクチャ



X20598-040218

QDMA ドライバーは、次の 3 つのコンポーネントで構成されています。

- デバイス制御ツール: PCIe デバイス クエリ、キュー管理、キューのコンテキストの読み出しなどの目的でネットワーク ソケットを作成します。
- DMA ツール: DMA トランザクションを開始するためのユーザー空間アプリケーションです。標準 Linux ユーティリティの `dd` または `fio` を使用できます。または、ドライバー パッケージのサンプル アプリケーションを使用できます。
- カーネル空間ドライバー: ディスクリプターを作成し、FPGA デバイスとの通信用にユーザー空間機能を低レベル コマンドに変換します。

ドライバーの使用

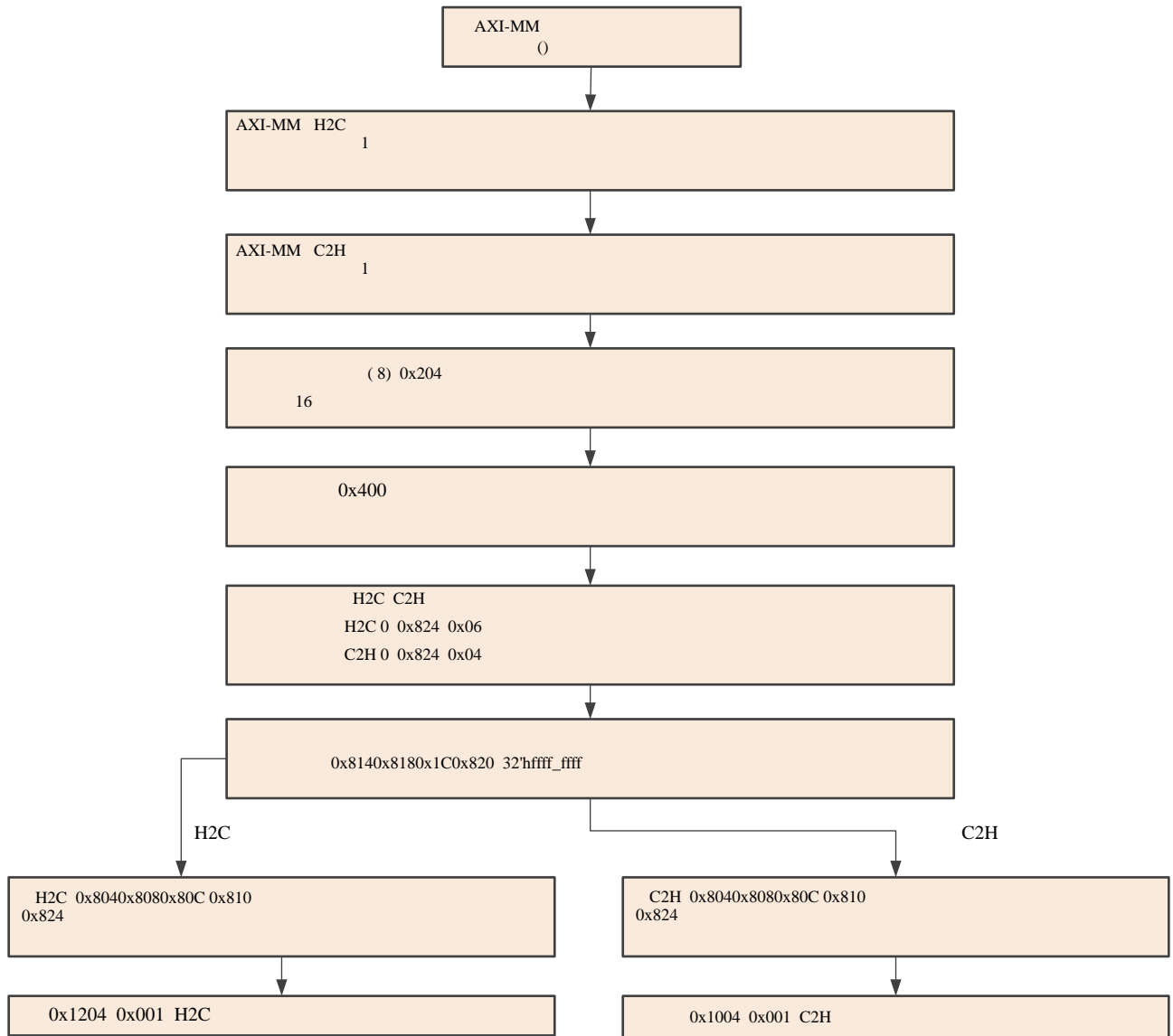
QDMA ドライバーおよびその資料は、次の URL からダウンロードできます。

- Linux および DPDK ドライバーの場合は、ザイリンクス DMA IP ドライバー (https://github.com/Xilinx/dma_ip_drivers) を参照してください。
- Windows ドライバーの場合は、QDMA Windows ドライバー ラウンジ (https://japan.xilinx.com/member/xdma_windows_driver.html) を参照してください。

リファレンス ソフトウェア ドライバーのフロー

AXI4 メモリ マップのフロー チャート

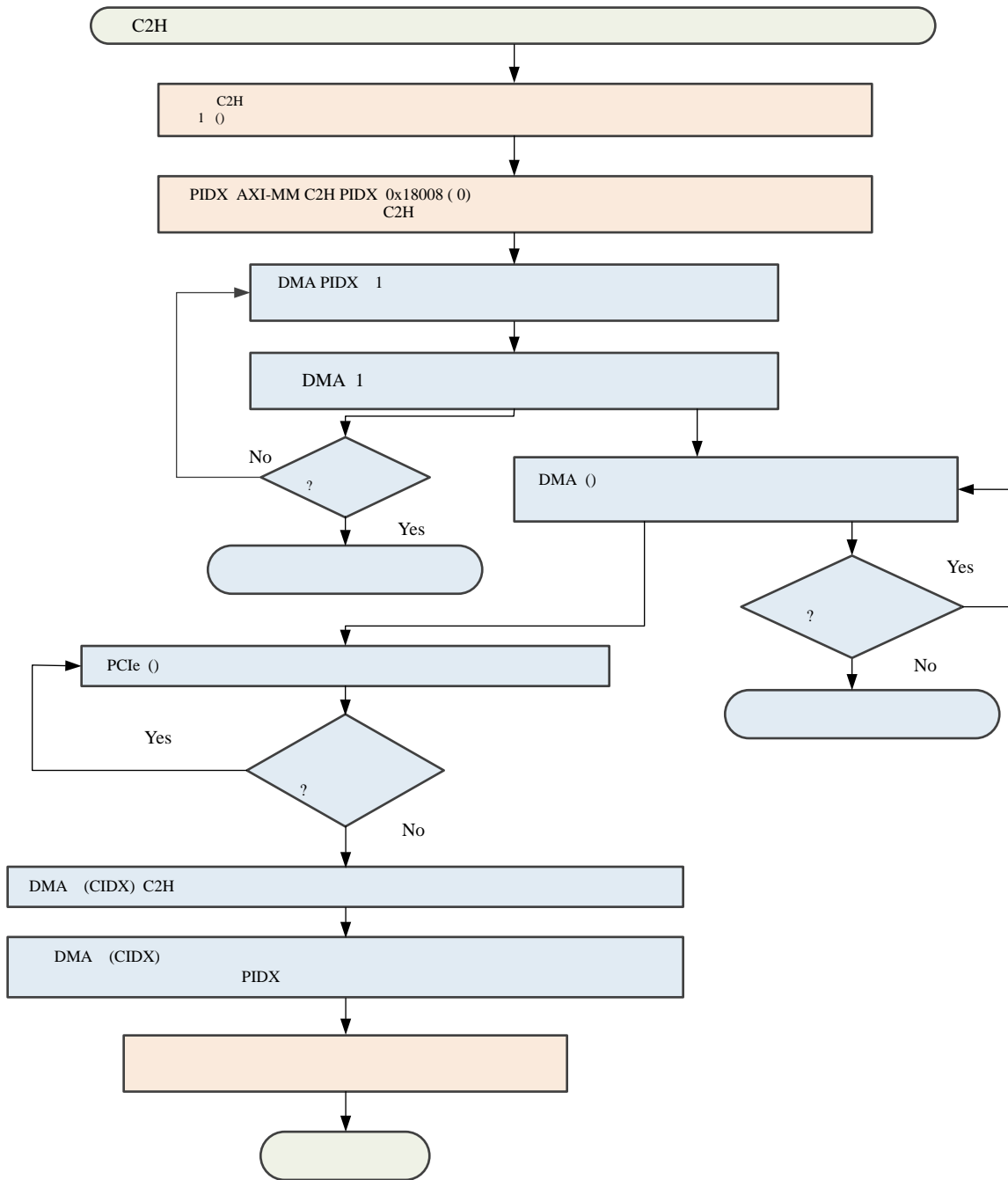
図 39: AXI4 メモリ マップのフロー チャート



X20550-041418

AXI4 メモリ マップド C2H フロー

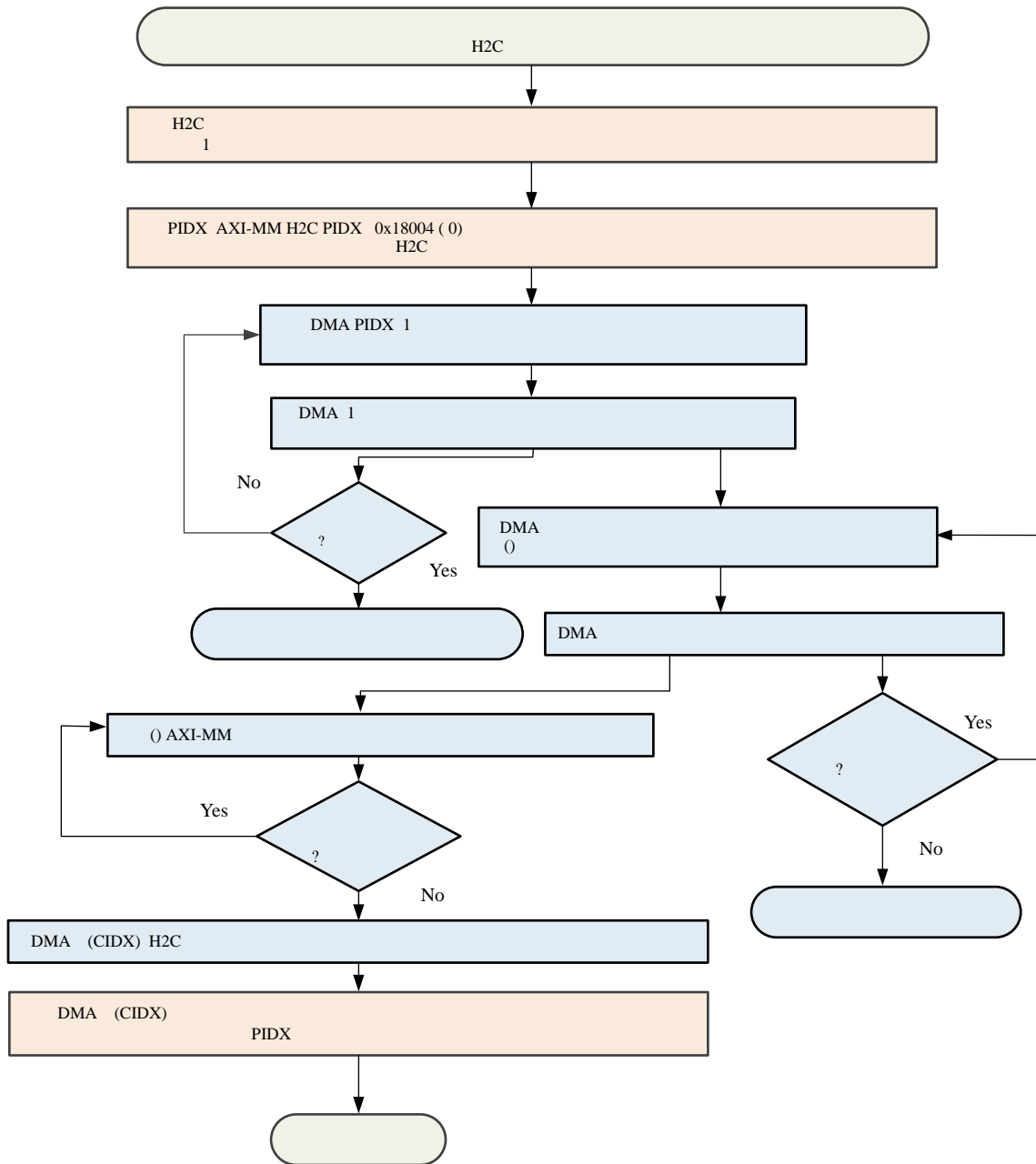
図 40: AXI4 メモリ マップド フローの図



X20525-120518

AXI4 メモリ マップド H2C フロー

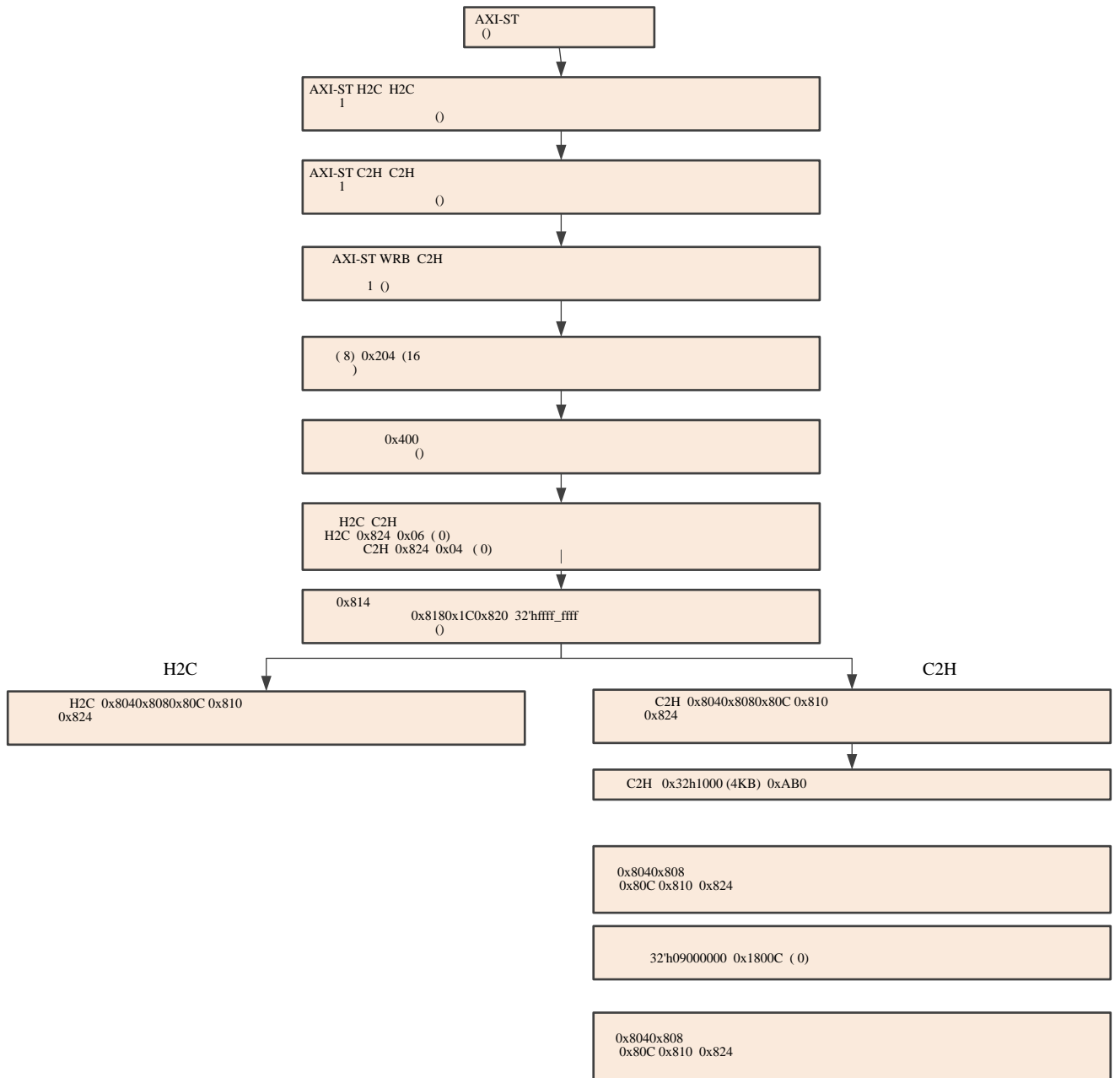
図 41: AXI4 メモリ マップド H2C フローの図



X20526-120518

AXI4-Stream フロー チャート

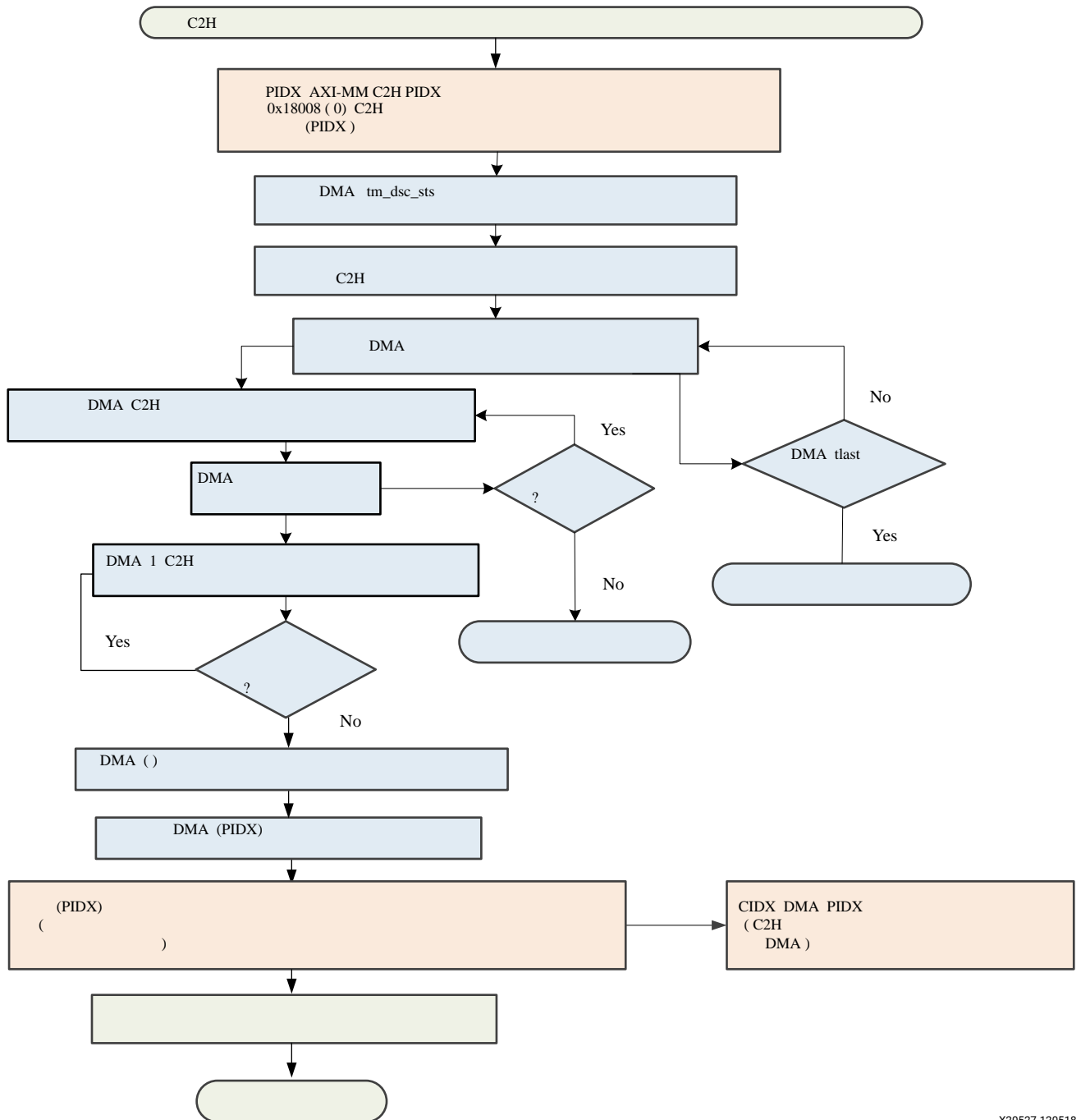
図 42: AXI4-Stream フロー チャート



X20551-120518

AXI4-Stream C2H フロー

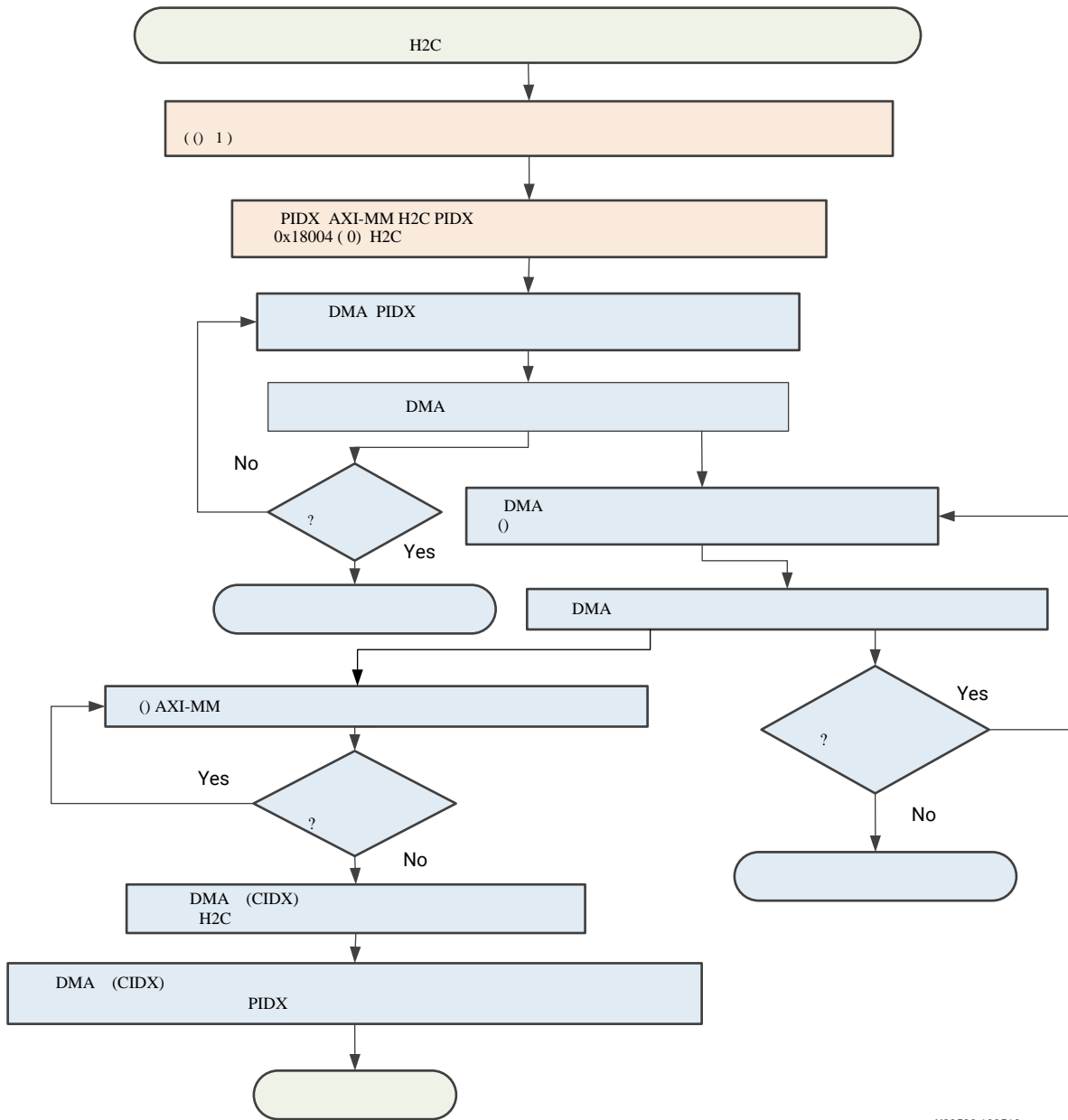
図 43: AXI4-Stream C2H フローの図



X20527-120518

AXI4-Stream H2C フロー

図 44: AXI4-Stream H2C フローの図



X20528-120518

その他のリソースおよび法的通知

ザイリンクス リソース

アンサー、資料、ダウンロード、フォーラムなどのサポート リソースは、[ザイリンクス サポート](#) サイトを参照してください。

Documentation Navigator およびデザイン ハブ

ザイリンクス Documentation Navigator (DocNav) では、ザイリンクスの資料、ビデオ、サポート リソースにアクセスでき、特定の情報を取得するためにフィルター機能や検索機能を利用できます。DocNav を開くには、次のいずれかを実行します。

- Vivado® IDE で [Help] → [Documentation and Tutorials] をクリックします。
- Windows で [スタート] → [すべてのプログラム] → [Xilinx Design Tools] → [DocNav] をクリックします。
- Linux コマンド プロンプトに「docnav」と入力します。

ザイリンクス デザイン ハブには、資料やビデオへのリンクがデザイン タスクおよびトピックごとにまとめられており、これらを参照することでキー コンセプトを学び、よくある質問 (FAQ) を参考に問題を解決できます。デザイン ハブにアクセスするには、次のいずれかを実行します。

- DocNav で [Design Hub View] タブをクリックします。
- ザイリンクス ウェブサイトで [デザイン ハブ](#) ページを参照します。

注記: DocNav の詳細は、ザイリンクス ウェブサイトの [Documentation Navigator](#) ページを参照してください。



注意: DocNav からは、日本語版は参照できません。ウェブサイトのデザイン ハブ ページをご利用ください。

参考資料

この製品ガイドの補足情報は、次の資料を参照してください。

1. AMBA AXI4-Stream Protocol Specification ([ARM IHI 0051A](#))
2. PCI-SIG 仕様 (www.pcisig.com/specifications)

3. 『Virtex-7 FPGA Integrated Block for PCI Express LogiCORE IP 製品ガイド』 (PG023)
4. 『7 Series FPGAs Integrated Block for PCI Express LogiCORE IP 製品ガイド』 (PG054: [英語版](#)、[日本語版](#))
5. 『UltraScale Devices Gen3 Integrated Block for PCI Express LogiCORE IP 製品ガイド』 (PG156: [英語版](#)、[日本語版](#))
6. 『AXI Bridge for PCI Express Gen3 Subsystem 製品ガイド』 (PG194)
7. 『DMA/Bridge Subsystem for PCI Express 製品ガイド』 (PG195)
8. 『UltraScale+ Devices Integrated Block for PCI Express 製品ガイド』 (PG213: [英語版](#)、[日本語版](#))
9. 『Vivado Design Suite: AXI リファレンス ガイド』 (UG1037: [英語版](#)、[日本語版](#))
10. 『Vivado Design Suite ユーザー ガイド: IP インテグレーターを使用した IP サブシステムの設計』 (UG994: [英語版](#)、[日本語版](#))
11. 『Vivado Design Suite ユーザー ガイド: IP を使用した設計』 (UG896: [英語版](#)、[日本語版](#))
12. 『Vivado Design Suite ユーザー ガイド: 入門』 (UG910: [英語版](#)、[日本語版](#))
13. 『Vivado Design Suite ユーザー ガイド: ロジック シミュレーション』 (UG900: [英語版](#)、[日本語版](#))
14. 『Vivado Design Suite ユーザー ガイド: 制約の使用』 (UG903)
15. 『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908: [英語版](#)、[日本語版](#))

改訂履歴

次の表に、この文書の改訂履歴を示します。

セクション	変更内容
12/05/2018 v3.0	
IP の概要およびドライバーの使用	Windows ドライバーのサポートを追加。
レジスタ空間	レジスタを追加および更新。
[PCIe MISC] タブ および [PCIe DMA] タブ	2018.3 リリース用に内容を更新。
第 6 章: サンプル デザイン	サンプル デザインを 2 つ追加、レジスタを更新。
付録 A: アップグレード	コアバージョンの変更を確認するアンサー レコードへのリンクを追加。
09/04/2018 v2.0	
ポートの説明	tm_dsc_sts_rdy (VDM ポート) および st_rx_msg_rdy (QDMA トラフィック マネージャー クレジット出力ポート) に対し、このインターフェイスが使用されていない場合は、レディ信号を 1 に接続する必要があることを強調。
レジスタ空間	未処理データ量がプログラムされたしきい値を超えた場合、H2C ストリーム エンジンからの読み出しリクエストをストールするレジスタを追加。 ユーザー トリガー、タイマー時間切れ、またはしきい値を超えるカウントを含む、新しい C2H 完了割り込みトリガー モードを追加。
06/22/2018 v2.0	
「概要」の章	全体的に更新。
「ポートの説明」セクション	表の内容および構成を一部変更。

セクション	変更内容
「レジスタ空間」セクション	メモリ マップ レジスタ空間および AXI4-Lite スレーブ レジスタ空間のセクションを追加。
「コンテキスト構造定義」および「キュー エントリ構造」のセクション	これらのセクションを削除し、コンテンツを「概要」の「QDMA 操作」セクションに移動。
「デザイン フローの手順」の章	[Basic] タブ、[Capabilities] タブ、[PCIe BARs] タブ、[PCIe Misc] タブ、[PCIe DMA] タブの説明を更新。
「サンプル デザイン」の章	サンプル デザインを 2 つ新規追加、サンプル デザインのレジスタを更新。
04/17/2018 v1.0	
初版。	

お読みください: 重要な法的通知

本通知に基づいて貴殿または貴社 (本通知の被通知者が個人の場合には「貴殿」、法人その他の団体の場合には「貴社」。以下同じ) に開示される情報 (以下「本情報」といいます) は、ザイリンクスの製品を選択および使用することのためにのみ提供されます。適用される法律が許容する最大限の範囲で、(1) 本情報は「現状有姿」、およびすべて受領者の責任で (with all faults) という状態で提供され、ザイリンクスは、本通知をもって、明示、黙示、法定を問わず (商品性、非侵害、特定目的適合性の保証を含みますがこれらに限られません)、すべての保証および条件を負わない (否認する) ものとし、また、(2) ザイリンクスは、本情報 (貴殿または貴社による本情報の使用を含む) に関係し、起因し、関連する、いかなる種類・性質の損失または損害についても、責任を負わない (契約上、不法行為上 (過失の場合を含む)、その他のいかなる責任の法理によるかを問わない) ものとし、当該損失または損害には、直接、間接、特別、付随的、結果的な損失または損害 (第三者が起こした行為の結果被った、データ、利益、業務上の信用の損失、その他あらゆる種類の損失や損害を含みます) が含まれるものとし、それは、たとえ当該損害や損失が合理的に予見可能であったり、ザイリンクスがそれらの可能性について助言を受けていた場合であったとしても同様です。ザイリンクスは、本情報に含まれるいかなる誤りも訂正する義務を負わず、本情報または製品仕様のアップデートを貴殿または貴社に知らせる義務も負いません。事前の書面による同意のない限り、貴殿または貴社は本情報を再生産、変更、頒布、または公に展示してはなりません。一定の製品は、ザイリンクスの限定的保証の諸条件に従うこととなるので、<https://japan.xilinx.com/legal.htm#tos> で見られるザイリンクスの販売条件を参照してください。IP コアは、ザイリンクスが貴殿または貴社に付与したライセンスに含まれる保証と補助的条件に従うこととなります。ザイリンクスの製品は、フェイルセーフとして、または、フェイルセーフの動作を要求するアプリケーションに使用するために、設計されたり意図されたりしていません。そのような重大なアプリケーションにザイリンクスの製品を使用する場合のリスクと責任は、貴殿または貴社が単独で負うものです。<https://japan.xilinx.com/legal.htm#tos> で見られるザイリンクスの販売条件を参照してください。

自動車用のアプリケーションの免責条項

オートモーティブ製品 (製品番号に「XA」が含まれる) は、ISO 26262 自動車用機能安全規格に従った安全コンセプトまたは余剰性の機能 (「セーフティ設計」) がない限り、エアバッグの展開における使用または車両の制御に影響するアプリケーション (「セーフティ アプリケーション」) における使用は保証されていません。顧客は、製品を組み込むすべてのシステムについて、その使用前または提供前に安全を目的として十分なテストを行うものとし、セーフティ設計なしにセーフティ アプリケーションで製品を使用するリスクはすべて顧客が負い、製品の責任の制限を規定する適用法令および規則にのみ従うものとし、また、

商標

© Copyright 2018 Xilinx, Inc. Xilinx、Xilinx のロゴ、Alveo、Artix、ISE、Kintex、Spartan、Versal、Virtex、Vivado、Zynq、およびこの文書に含まれるその他の指定されたブランドは、米国およびその他の各国のザイリンクス社の商標です。PCI、PCIe、および PCI Express は PCI-SIG の商標であり、ライセンスに基づいて使用されています。すべてのその他の商標は、それぞれの所有者に帰属します。

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、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。フィードバックは日本語で入力可能です。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。