

# LogiCORE IP Aurora 64B/66B v9.2

## 製品ガイド

Vivado Design Suite

PG074 2014 年 6 月 4 日

本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

## IP の概要

### 第 1 章: 概要

機能概要 .....	6
アプリケーション .....	6
サポートされていない機能 .....	7
ライセンスおよび注文情報 .....	7

### 第 2 章: 製品仕様

規格 .....	9
パフォーマンス .....	9
リソース使用状況 .....	10
ポートの説明 .....	11
機能の詳細説明 .....	35

### 第 3 章: コアを使用するデザイン

一般的なデザイン ガイドライン .....	65
共有ロジック .....	65
クロッキング .....	69
コアの機能 .....	75

### 第 4 章: デザイン フローの手順

コアのカスタマイズおよび生成 .....	78
UltraScale アーキテクチャ特有デザインのコアのカスタマイズ オプション .....	88
コアへの制約 .....	92
シミュレーション .....	94
合成およびインプリメンテーション .....	95

### 第 5 章: サンプル デザインの詳細

ディレクトリとファイルの内容 .....	97
サンプル デザインのクイック スタート .....	97
サンプル デザインの詳細 .....	98
サンプル デザインの実装 .....	110
サンプル デザインのハードウェア リセット FSM .....	110

### 第 6 章: テストベンチ

#### 付録 A: 検証、互換性、相互運用性

#### 付録 B: 移行およびアップグレード

デバイスの移行 .....	117
Vivado Design Suite への移行 .....	117
Vivado Design Suite でのアップグレード .....	117

レガシー (LocalLink ベース) Aurora コアから AXI4-Stream Aurora への移行 ..... 119

**付録 C: デバッグ**

    ザイリンクス ウェブサイト ..... 126  
    デバッグ ツール ..... 127  
    シミュレーション デバッグ ..... 129  
    ハードウェア デバッグ ..... 131  
    評価ボードでデザインを実行 ..... 133  
    インターフェイスのデバッグ ..... 135

**付録 D: Transceiver Wizard でラッパー ファイルを生成**

**付録 E: 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 ..... 137  
    参考資料 ..... 137  
    改訂履歴 ..... 138  
    法的通知 ..... 139

## はじめに

ザイリンクスの LogiCORE™ IP Aurora 64B/66B コアは、スケラブルで軽量な高速データ レートの高速シリアル通信用リンク レイヤー プロトコルです。プロトコルはオープンで、ザイリンクス デバイス テクノロジーを使用して実装できます。

Aurora 64B/66B コアのソース コードは Vivado® Design Suite で生成されます。コアはシンプレックスまたはフル デュプレックスで、シンプルで 2 つのユーザー インターフェイスのいずれかを選択し、オプションのフロー制御を使用できます。

## 機能

- Vivado Design Suite でサポートされる
- 500Mb/s ~ 200Gb/s を超えるスループットの汎用データ チャンネル
- 任意の連続する 16 個の GTX トランシーバーまたは 16 個の Virtex®-7 FPGA GTH トランシーバー、および 16 個の UltraScale™ デバイス GTH トランシーバーをサポート
- Aurora 64B/66B プロトコル仕様 v1.2 準拠 (64B/66B エンコード)
- 伝送オーバーヘッドが非常に低いため (3%) リソース コストが低い
- 使いやすい AXI4-Stream (フレーミング) またはストリーミング インターフェイス、およびオプションのフロー制御
- チャンネルの自動初期化および管理
- フルデュプレックス (複信) または単方向 (単信)
- ユーザー データ用の 32 ビット CRC (巡回冗長検査)
- RX 極性反転をサポート
- ビッグ エンディアン形式/リトル エンディアン形式の AXI4-Stream ユーザー インターフェイス

この LogiCORE IP について	
<b>コアの概要</b>	
サポートされる デバイス ファミリー <sup>(1)</sup>	UltraScale アーキテクチャ、Zynq®-7000 All Programmable SoC、Virtex-7 <sup>(2)</sup> 、Kintex®-7 <sup>(2)</sup>
サポートされる ユーザー インターフェイス	AXI4-Stream
リソース <sup>(3)</sup>	詳細は、表 2-1 および表 2-2 を参照
<b>コアに含まれるもの</b>	
デザイン ファイル	RTL
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 デザイン制約ファイル (.xdc)
シミュレーション モデル	なし
サポートされる ソフトウェア ドライバー	なし
<b>テスト済みデザイン フロー<sup>(4)</sup></b>	
デザイン入力	Vivado Design Suite Vivado IP インテグレーター
シミュレーション	サポートされるシミュレータについては、 <a href="#">『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』</a> を参照
合成	Vivado 合成
<b>サポート</b>	
<a href="http://japan.xilinx.com/support">japan.xilinx.com/support</a> 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. 詳細は、『7 シリーズ FPGA 概要』(DS180) [\[参照 1\]](#) および『UltraScale アーキテクチャおよび製品概要』(DS890) [\[参照 2\]](#) を参照してください。
3. 性能データの詳細は、9 ページの「パフォーマンス」を参照してください。
4. 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』](#)を参照してください。

## 概要

この製品ガイドは、LogiCORE™ IP Aurora 64B/66B v9.2 コアの機能および動作について説明し、またコアの設計、カスタマイズ、および実装に関する情報を提供しています。

Aurora 64B/66B はマルチギガビット リンクの軽量なシリアル通信プロトコルです (図 1-1)。1 つまたは複数の GTX/GTH トランシーバーを使用するデバイス間のデータ伝送に使用されます。通信方式は、フルデュプレックス (双方向データ通信) またはシングルデュプレックス (単方向データ通信) のいずれかに指定できます。

LogiCORE IP Aurora 64B/66B コアは、AMBA® プロトコルの AXI4-Stream ユーザー インターフェイスをサポートします。対象となる UltraScale™、Zynq®-7000、Virtex®-7、および Kintex®-7 デバイスで高速シリアル GTX/GTH トランシーバーを使用して Aurora 64B/66B プロトコルを実装します。500Mbps から 200Gbps を超えるスループットで、低コスト、汎用のデータ チャンネルを提供するため、サポートされているライン レートで実行している場合、最大 16 個の連続デバイスの GTX/GTH トランシーバーをサポートできます。

Aurora 64B/66B コアは、自動化されたシミュレーション テストを使用してプロトコルに準拠しているかどうかを検証されます。

注記 : Aurora 64B/66B v9.2 は、UltraScale、Zynq-7000、Virtex-7、および Kintex-7 デバイスをサポートします。

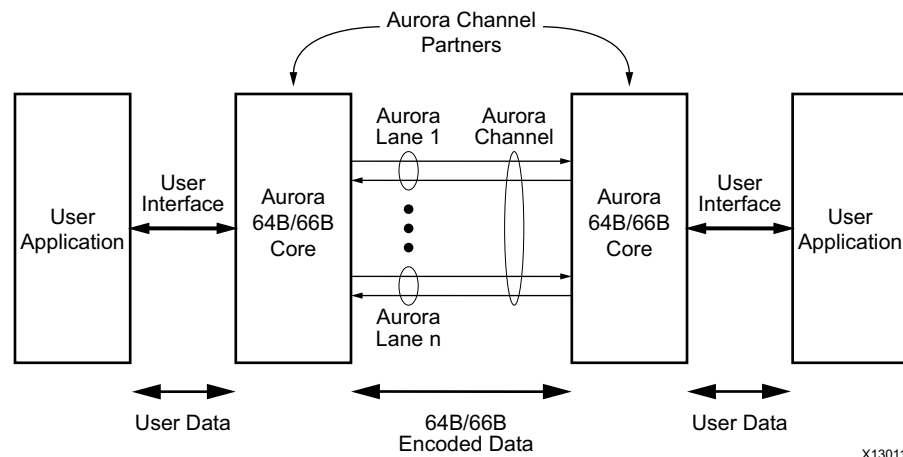


図 1-1 : Aurora 64B/66B チャンネルの概要

Aurora 64B/66B コアは、Aurora 64B/66B チャンネル パートナーに接続されると、チャンネルを自動的に初期化します。初期化後、アプリケーションはこのチャンネルを介してフレームまたはストリームとしてデータを送受信できます。Aurora 64B/66B のフレーム サイズは任意のサイズに設定でき、優先要求による割り込みが可能です。ロックの維持および過度な電磁干渉の防止のため、有効なデータ バイト間のギャップは自動的にアイドル信号で埋められます。フロー制御は Aurora 64B/66B ではオプションで、リンク パートナーの送信データ レートを高めるため、またはチャンネルで短い優先メッセージを送信するのに使用するために使用できます。

ストリームは Aurora 64B/66B では 1 つの無限フレームとしてインプリメントされます。データが送信されていないときは常に、リンクを有効にしておくためアイドルが送信されます。ビット エラー、接続のない状態、または装置エラーが多発すると、コアがリセットして新しいチャンネルの再初期化が行われます。Aurora 64B/66B コアでは、マルチレーン チャンネルの受信側で最大 2 つのシンボル スキューをサポートできます。Aurora 64B/66B プロトコルには 64B/66B エンコードが使用されます。64B/66B エンコードの伝送オーバーヘッドは、8B/10B エンコードの 25% のオーバーヘッドと比較すると非常に低いため (3%)、パフォーマンスを向上させることができます。



**推奨 :** Aurora 64B/66B コアは完全検証されたソリューションですが、完全デザインをインプリメントする際の要件は、アプリケーションのコンフィギュレーションや機能によって異なります。ベストな結果を得るには、ザイリンクス インプリメンテーション ツールおよびデザイン制約ファイル (XDC) を使用して、ハイパフォーマンスのパイプライン化された FPGA デザインの構築に関する知識が望まれます。

詳細は、第 2 章の「ステータス、制御、およびトランシーバー インターフェイス」を参照してください。『UltraScale FPGA GTH トランシーバー ユーザー ガイド』(UG576) [参照 3] および『7 シリーズ FPGA GTX/GTH トランシーバー ユーザー ガイド』(UG476) [参照 4] の PCB デザイン要件に関する情報もあわせて参照してください。特定要件の詳細な検証および評価を行う場合は、お近くのザイリンクス販売代理店へお問い合わせください。

## 機能概要

LogiCORE IP Aurora 64B/66B コアは、UltraScale、Zynq-7000、Virtex-7、および Kintex-7 に搭載されている高速シリアル トランシーバーを使用して Aurora 64B/66B プロトコルを実装します。このコアは、AMBA® プロトコルの AXI4-Stream ユーザー インターフェイスをサポートします。

Aurora 64B/66B コアは、『Aurora 64B/66B プロトコル仕様』(SP011) [参照 5] に準拠し、高速シリアル GTX または GTH トランシーバーを使用します。コアはオープンソース コードとして提供され、Verilog デザイン環境をサポートしています。各コアには、サンプル デザインとサポーティング モジュールが伴います。

## アプリケーション

Aurora 64B/66B コアは、低リソース コスト、スケーラブルなスループット、および柔軟なデータ インターフェイスという特徴から、さまざまなアプリケーションで使用できます。Aurora 64B/66B コアのアプリケーション例は次のとおりです。

- チップ間のリンク : チップ間のパラレル接続を高速シリアル接続に置き換えることで、PCB に必要なトレースおよびレイヤーの数を著しく抑えることができます。Aurora 64B/66B コアは、GTX、および GTH トランシーバーの使用に必要なロジックを最低限の FPGA リソース コストで提供します。
- ボード間のリンクおよびバックプレーンのリンク : Aurora 64B/66B では標準 64B/66B エンコードが使用されます。これは 10 ギガビット イーサネット用の優先エンコードで、ケーブルおよびバックプレーンの既存ハードウェア規格との互換性を提供します。Aurora 64B/66B はライン レートおよびチャンネル幅の両方が調整可能で、廉価で古いハードウェアでも新しい高パフォーマンス システムで使用することができます。
- シンプレックス接続 (単方向) : 一部のアプリケーションでは、高速バック チャンネルが不要です。Aurora 64B/66B シンプレックス プロトコルは、単方向のチャンネル初期化を実行するオプションを提供し、バック チャンネルがない場合でも GTX、および GTH トランシーバーの使用を可能にします。またフルデュプレックスのリソースを使用しないため、コストを抑えることができます。

- ASIC アプリケーション : Aurora 64B/66B は FPGA に限定されるわけではなく、プログラマブル ロジックと高性能 ACIS との間にスケーラブルで高性能なリンクを作成するためにも使用できます。Aurora 64B/66B プロトコルはシンプルであるため、ASIC でも FPGA でもリソースコストを抑えやすく、また、Aurora 64B/66B バス ファンクション モデル (BFM) のようなデザイン リソースを自動化適合テストと併用することで、Aurora 64B/66B 接続を確立しやすくなっています。ASIC アプリケーション用 Aurora のライセンスについては、ザイリンクス販売担当者または [auroramkt@xilinx.com](mailto:auroramkt@xilinx.com) までお問い合わせください。

---

## サポートされていない機能

Aurora 64B/66B には、サポートされていない機能はありません。

---

## ライセンスおよび注文情報

このザイリンクス LogiCORE IP モジュールは、[ザイリンクス エンド ユーザー ライセンス規約](#)のもとザイリンクス 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 IP およびその他のザイリンクス LogiCORE IP モジュールは、[ザイリンクス IP コア](#) ページから入手できます。その他のザイリンクス LogiCORE IP モジュールおよびツールの価格や提供状況については、[ザイリンクス販売代理店](#)にお問い合わせください。

ASIC (application specific integrated circuit) で Aurora 64B/66B コアを使用する場合は、[ザイリンクス コア ライセンス契約](#)に基づいた別途有料ライセンス契約が必要です。詳細は、Aurora マーケティング ([auroramkt@xilinx.com](mailto:auroramkt@xilinx.com)) へお問い合わせください。

詳細は、[Aurora 64B/66B 製品ページ](#)を参照してください。

# 製品仕様

図 2-1 は Aurora 64B/66B コアのインプリメンテーションのブロック図です。

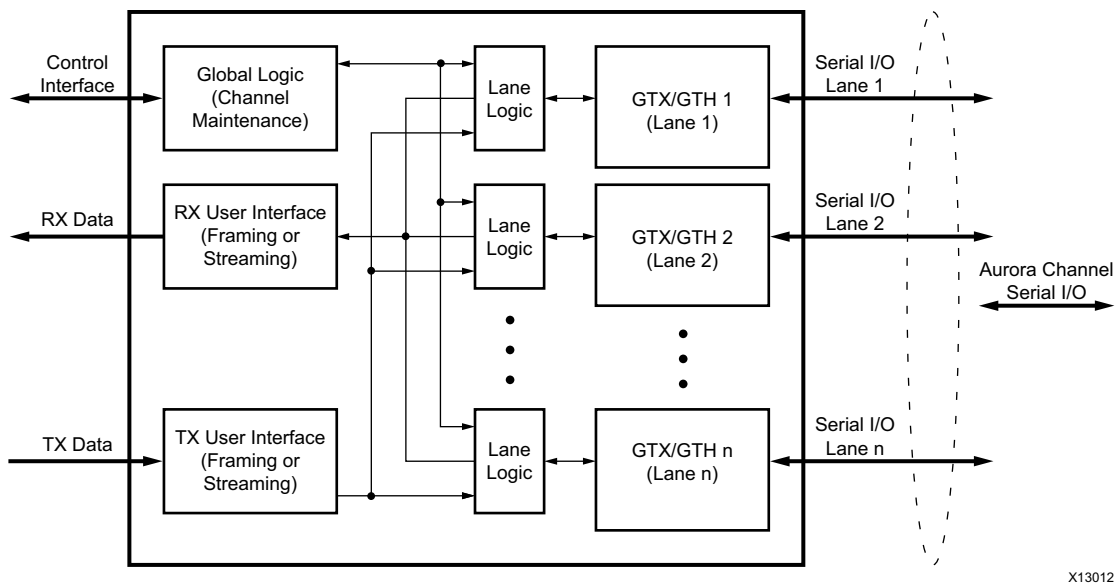


図 2-1 : Aurora 64B/66B コアのブロック図

Aurora 64B/66B コアの主な論理モジュールには次があります。

- レーン ロジック : 各 GTX および GTH トランシーバーはレーン ロジック モジュールのインスタンスで駆動されます。このモジュールは、各 GTX および GTH トランシーバーを初期化し、制御文字のエンコードおよびデコードとエラー検出を処理します。
- グローバル ロジック : Aurora 64B/66B コアのグローバル ロジック モジュールは、チャンネル初期化のチャンネル ボンディングを実行します。チャンネルが動作している間、Aurora 64B/66B プロトコルで定義されている not ready アイドル文字を追跡し、エラーがないかすべてのレーン ロジック モジュールを監視します。
- RX ユーザー インターフェイス : RX (受信) ユーザー インターフェイスは、チャンネルからアプリケーションヘッダーデータを伝送します。ストリーミング データは、データバスと、フロー制御用の valid 信号および ready 信号を装備した単純なストリーム インターフェイスを使用して伝送されます。フレームは標準の AXI4-Stream インターフェイスを使用して伝送されます。このモジュールはフロー制御機能も実行します。

- TX ユーザー インターフェイス : TX (送信) ユーザー インターフェイスは、アプリケーションからチャンネルヘッダを送ります。valid 信号および ready 信号のあるストリーム インターフェイスはストリーミング データに使用されます。標準の AXI4-Stream インターフェイスはデータ フレームに使用されます。このモジュールはフロー制御の TX 機能も実行します。このモジュールにはクロック補正を制御するためのインターフェイスがあります (接続されている Aurora 64B/66B コア間のクロック周波数の若干の差が原因で発生するエラーを防止するため、特殊文字が定期的に送信される)。通常、このインターフェイスは Aurora 64B/66B コアに含まれる標準クロック補正管理モジュールで駆動されますが、オフにしたり、特殊なニーズに対応するためにカスタム ロジックで駆動することもできます。

## 規格

Aurora 64B/66B コアは、『Aurora 64B/66B プロトコル仕様 v1.2』(SP011) [参照 5] に準拠しています。

## パフォーマンス

このセクションでは、さまざまなコア コンフィギュレーションの性能情報について詳しく説明します。

### 最大周波数

コアの最大周波数は、サポートされるラインレートおよびデバイスのスピード グレードに依存します。

### レイテンシ

デフォルトのシングル レーン コンフィギュレーションの場合、Aurora 64B/66B コアのレイテンシは、プロトコル エンジン (PE) を通るパイプライン遅延や GTX/GTH トランシーバーを通る遅延によるものです。AXI4-Stream インターフェイス幅が増加すると、PE パイプライン遅延が増加します。GTX/GTH トランシーバーの遅延は、GTX/GTH トランシーバーの各機能で一定となります。

このセクションでは、UltraScale™、Zynq®-7000、Virtex®-7、および Kintex®-7 デバイスの GTX/GTH トランシーバーをベースとするデザインで user\_clk サイクルを単位として、Aurora 64B/66B コア AXI4-Stream ユーザー インターフェイスのレイテンシを測定する方法を説明します。レイテンシを説明するにあたって、Aurora 64B/66B モジュールは、GTX/GTH トランシーバー ロジックと FPGA ロジックにインプリメントされたプロトコル エンジン (PE) ロジックに分割されます。

図 2-2 は、フレーム パスのレイテンシを表しています。

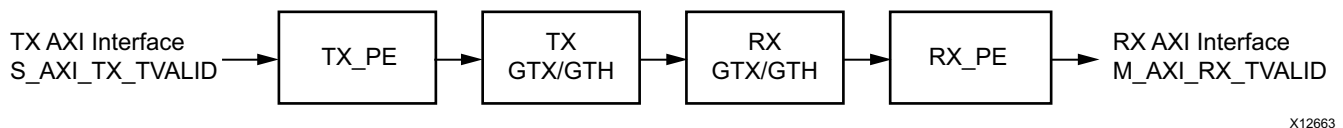


図 2-2 : フレーム パスのレイテンシ

注記 : 図 2-2 には、Aurora 64B/66B チャンネルの各側におけるシリアル接続の長さによって生じるレイテンシは考慮されていません。

GTX/GTH トランシーバーを使用するデザインで、s\_axi\_tx\_tvalid 信号と s\_axi\_tx\_tready 信号が最初にアサートされてから m\_axi\_rx\_tvalid がアサートされるまでの最大レイテンシは、シミュレーションで user\_clk の約 53 サイクルとなります。

パイプライン遅延は、クロック スピードを維持することを目的としています。

## スループット

Aurora 64B/66B コアのスループットは、トランシーバーの数および選択したトランシーバーのターゲット ライン レートによって異なり、シングル レーン デザインから 16 レーン デザインでそれぞれ 0.48Gb/s ~ 203.3Gb/s の範囲で変化します。スループットは、Aurora 64B/66B プロトコル エンコードの 3% のオーバーヘッドと 0.5Gb/s ~ 13.1Gb/s ライン レート 範囲を使用して計算されました。

## リソース使用状況

表 2-1 および表 2-2 では、xc7vx485tffg1157-1 デバイスに Vivado® Design Suite を使用してインプリメントした場合、選択された Aurora 64B/66B のフレーミングまたはストリーミングで使用されるルックアップ テーブル (LUT) およびフリップフロップ (FF) の数を示しています。Aurora 64B/66B コアは、これらの表にはないコンフィギュレーションでも使用可能です。表にはフロー制御に使用される追加リソースは含まれていません。また、FRAME\_GEN や FRAME\_CHECK などのサンプル デザイン モジュールで使用される追加リソースも含まれていません。ここで提供する値は、特定のコンフィギュレーションで取得した正確な値です。サポート ロジックを含むデフォルト コンフィギュレーション (3.125G) での値を示しています。

表 2-1: Virtex-7 ファミリの GTX トランシーバーのリソース使用量 (ストリーミング)

Virtex-7 ファミリ (GTX トランシーバー)		ストリーミング		
		デュプレックス	シンプレックス	
レーン	リソース タイプ	フルデュプレックス	TX のみ	RX のみ
1	LUT	549	315	377
	フリップフロップ	1359	476	957
2	LUT	1044	467	686
	フリップフロップ	2379	761	1721
4	LUT	1971	711	1452
	フリップフロップ	4347	1380	3139
8	LUT	3610	1256	2805
	フリップフロップ	8219	2539	5973
16	LUT	6656	1949	5496
	フリップフロップ	15966	4825	11641

表 2-2 : Virtex-7 ファミリの GTX トランシーバーのリソース使用量 (フレーミング)

Virtex-7 ファミリ (GTX トランシーバー)		フレーミング		
		デュプレックス	シンプレックス	
レーン	リソース タイプ	フルデュプレックス	TX のみ	RX のみ
1	LUT	873	315	597
	フリップフロップ	1398	499	975
2	LUT	1475	471	1106
	フリップフロップ	2442	799	1748
4	LUT	2628	764	2012
	フリップフロップ	4444	1425	3182
8	LUT	4997	1566	3896
	フリップフロップ	8391	2623	6046
16	LUT	9418	2874	7560
	フリップフロップ	16273	5018	11771

注記 : UltraScale デバイスのリソース使用量は、7 シリーズ デバイスの場合と類似すると予想されます。

## ポートの説明

各 Aurora 64B/66B コアを生成するために使用されるパラメーターはその特定コアで使用可能なインターフェイス (図 2-3) を決定します。Aurora 64B/66B コアには、4 ~ 8 個のインターフェイスがあります。

- 12 ページの「ユーザー インターフェイス」
- 15 ページの「ユーザー フロー制御インターフェイス」
- 17 ページの「ネイティブフロー制御インターフェイス」
- 17 ページの「ユーザー K ブロック インターフェイス」
- 24 ページの「GTX および GTH トランシーバーのインターフェイス」
- 34 ページの「クロック インターフェイス」
- 62 ページの「DRP インターフェイス」
- 64 ページの「クロック補正インターフェイス」

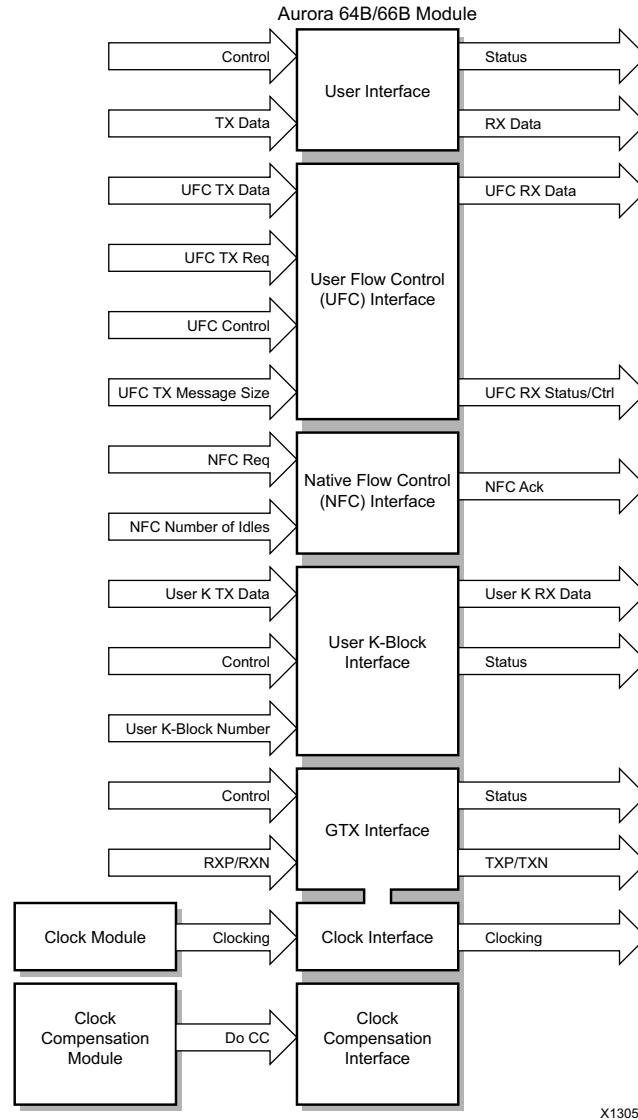


図 2-3 : 最上位インターフェイス

## ユーザー インターフェイス

このインターフェイスには、Aurora 64B/66B コアに入出力されるストリームまたはフレーム データの読み出しおよび書き込みに必要なポートがすべて含まれています。フレーミング インターフェイスで Aurora 64B/66B コアが生成される場合は、AXI4-Stream ポートが使用されます。ストリーミング モジュールの場合、インターフェイスは valid および ready ポートを含む単純なデータ ポートで構成されます。フルデュプレックス コアには送信 (TX) と受信 (RX) の両方のポートが含まれ、シンプレックス コアではサポートされている方向に必要なポートのみが使用されます。すべてのインターフェイスのデータ ポート幅は、コアで使用される GTX/GTH トランシーバーの数に依存します。CRC オプションが選択されている場合は、フレーミング インターフェイスのすべてのフレームに対して、CRC がデータ インターフェイス上で計算されます。

## フレーミング インターフェイスのポート (AXI4-Stream)

表 2-3 では、AXI4-Stream TX データ ポートとそれらの説明を示しています。詳細は、37 ページの「フレーミング インターフェイス」を参照してください。コアには、Vivado® IDE から AXI4-Stream User I/O をリトル エンディアン形式で構成するオプションがあります。デフォルトはビッグ エンディアン形式です。

表 2-3 : AXI4-Stream のユーザー I/O ポート (TX)

名前	方向	説明
<code>s_axi_tx_tdata[0:(64n-1)]</code> または <code>s_axi_tx_tdata[(64n-1):0]</code> <sup>(1)</sup>	入力	出力されるデータ (昇ビット順) です。[Little Endian Support] がオンの場合、コアは <code>s_axi_tx_tdata[(64n-1):0]</code> を使用します。
<code>s_axi_tx_tready</code>	出力	ソースからの信号が受信されると ( <code>s_axi_tx_tvalid</code> もアサートされている場合)、クロック エッジでアサート (High) されます。ソースからの信号が無視される場合には、クロック エッジでディアサート (Low) されます。
<code>s_axi_tx_tlast</code>	入力	フレームの終わりを示します (アクティブ High)。
<code>s_axi_tx_tkeep[0:(8n-1)]</code> または <code>s_axi_tx_tkeep[(8n-1):0]</code> <sup>(1)</sup>	入力	最後のデータ ビートで有効なバイト数を示します (有効なバイト数 = <code>tkeep</code> 内の「1」の数)。 [Little Endian Support] がオンの場合、コアは <code>s_axi_tx_tkeep[(8n-1):0]</code> を使用します。 例 : <code>s_axi_tx_tkeep = FF</code> は、8 バイトが有効という意味 ( <code>s_axi_tx_tlast</code> がアサートされた場合のみ有効)。 Aurora コアは、連続的に位置合わせされたストリームおよび連続的に位置合わせされていないストリームをサポートし、LSB から MSB へと連続的なデータの埋め込みを要求します。有効な <code>s_axi_tx_tdata</code> バスの間に無効なバイトが挿入されることはありません。
<code>s_axi_tx_tvalid</code>	入力	ソースからの AXI4-Stream 信号が有効な場合にアサート (High) されます。ソースからの AXI4-Stream 制御信号またはデータが無視される場合にはディアサート (Low) されます。

1. n は、レーン数を表します。

表 2-4 では、AXI4-Stream RX データポートとそれらの説明を示しています。詳細は、37 ページの「フレーミング インターフェイス」を参照してください。

表 2-4 : AXI4-Stream のユーザー I/O ポート (RX)

名前	方向	説明
m_axi_rx_tdata[0:(64n-1)] または m_axi_rx_tdata[(64n-1):0] <sup>(1)</sup>	出力	チャンネルパートナーから入力されるフレーム データです (昇ビット順)。[Little Endian Support] がオンの場合、コアは m_axi_rx_tdata[(64n-1):0] を使用します。
m_axi_rx_tkeep[0:8n-1] または m_axi_rx_tkeep[8n-1:0] <sup>(1)</sup>	出力	最後のデータ ビートで有効なバイト数を示します。m_axi_rx_tlast がアサートされている場合のみ有効です。[Little Endian Support] がオンの場合、コアは m_axi_rx_tkeep[8n-1:0] を使用します。
m_axi_rx_tvalid	出力	Aurora コアからのデータおよび制御信号が有効の場合にアサート (High) されます。 Aurora コアからのデータまたは制御信号を無視する場合にはディアサート (Low) されます。
m_axi_rx_tlast	出力	入力されるフレームの最後を示します (アクティブ High で、user_clk の 1 サイクル間アサートされる)。

1. n は、レーン数を表します。

## ストリーミング ポート

表 2-5 では、ストリーミングの TX データポートについて説明しています。

表 2-5 : ストリーミングのユーザー I/O ポート (TX)

名前	方向	説明
s_axi_tx_tdata[0:(64n-1)] または s_axi_tx_tdata[(64n-1):0]	入力	出力されるデータ (昇ビット順) です。[Little Endian Support] がオンの場合、コアは s_axi_tx_tdata[(64n-1):0] を使用します。
s_axi_tx_tready	出力	ソースからの信号が受信されると (s_axi_tx_tvalid もアサートされている場合)、クロック エッジでアサート (High) されます。 ソースからの信号が無視される場合には、クロック エッジでディアサート (Low) されます。
s_axi_tx_tvalid	入力	ソースからの AXI4-Stream 信号が有効な場合にアサート (High) されます。 ソースからの AXI4-Stream 制御信号またはデータが無視される場合にはディアサート (Low) されます。

表 2-6 では、ストリーミングの RX データポートについて説明しています。これらのポートは、フルデュプレックスおよびシンプレックス RX フレーミング コアに含まれます。詳細は、44 ページの「ストリーミング インターフェイス」を参照してください。

表 2-6 : ストリーミングのユーザー I/O ポート (RX)

名前	方向	説明
m_axi_rx_tdata[0:(64n-1)] または m_axi_rx_tdata[(64n-1):0]	出力	チャンネルパートナーから入力されるデータです (昇ビット順)。[Little Endian Support] がオンの場合、コアは m_axi_rx_tdata[(64n-1):0] を使用します。
m_axi_rx_tvalid	出力	Aurora 64B/66B コアからのデータおよび制御信号が有効の場合にアサート (High) されます。 Aurora 64B/66B コアからのデータまたは制御信号を無視する場合にはディアサート (Low) されます。

### 注記 :

すべてのフロー制御の中で最も優先される要求は次のとおりです。

- GT TX インターフェイスからの TXDATAVALID 信号のディアサート (1 サイクル)
- CC の送信 (6 サイクル)

## ユーザー フロー制御インターフェイス

ユーザー フロー制御 (UFC) を有効にしてコアを生成した場合、UFC インターフェイスが作成されます。UFC インターフェイスの TX 側には、UFC メッセージを開始するのに使用される req (要求)、tvalid、および tready ポート、そしてメッセージの長さを指定するためのポート (ms) があります。UFC インターフェイスの valid および ready ポートにより変わりますが、UFC 要求直後にユーザーが UFC の DATA ポートにメッセージ データを送信します。これを受けて、ユーザー インターフェイスの ready ポートがディアサートされ、コアが標準データを受け付けられないことを示します。これで、UFC データが UFC データ ポートに書き込み可能になります。

UFC インターフェイスの RX 側には、UFC メッセージをフレームとして読み出すことができる AXI4-Stream ポートセットがあります。フルデュプレックス モジュールには TX と RX の UFC ポートがありますが、シンプレックス モジュールには、サポートされる方向にデータ送信するために必要なインターフェイスのみが含まれます。表 2-7 では UFC インターフェイスのポートについて説明しています。詳細は、48 ページの「ユーザー フロー制御」を参照してください。

表 2-7: UFC の I/O ポート

名前	方向	説明
ufc_tx_req	入力	チャンネル パートナーへの UFC メッセージ送信が要求されると、アサートされます (アクティブ High)。別の UFC メッセージが進行中で、最後のサイクルの途中でない限り、1 サイクル後に要求が処理されます。要求後、優先順位の高いイベントによって割り込みされない限り、2 サイクル以内に s_axi_ufc_tx_tdata バスはデータ送信可能な状態となります。
ufc_tx_ms[0:7] または ufc_tx_ms[7:0]	入力	UFC メッセージ内のバイト数を指定します (メッセージ サイズ)。最大の UFC メッセージ サイズは 256 です。ufc_tx_ms に指定する値は、転送される実際のバイト数より 1 つ少なくなります。たとえば、この値が 3 の場合、実際には 4 バイトのデータが送信されます。値が 0 の場合、1 バイトが送信されます。 [Little Endian Support] がオンの場合、コアは ufc_tx_ms [7:0] を使用します。
s_axi_ufc_tx_tready	出力	Aurora 8B/10B コアが s_axi_ufc_tx_tdata インターフェイスからデータを読み出す準備が整うとアサート (アクティブ High) されます。この信号は、優先順位の高い要求が進行中でない場合、ufc_tx_req がアサートされてから 1 クロック サイクル後にアサートされます。コアが直前に要求された UFC メッセージのデータを待機する間、s_axi_ufc_tx_tready はアサートを維持します。CC、CB、および NFC 要求は優先順位が高いため、これらが進行中の場合、この信号はディアサートされます。s_axi_ufc_tx_tready がアサートされる間、s_axi_tx_tready はディアサートされます。
s_axi_ufc_tx_tdata[0:(64n-1)] または s_axi_ufc_tx_tdata[(64n-1):0]	入力	Aurora チャンネルへ送信される UFC メッセージの入力バスです。s_axi_ufc_tx_tvalid および s_axi_ufc_tx_tready の両方が user_clk の立ち上がりエッジでアサートされる場合のみ、データがバスから読み出されてチャンネルへ送信されます。メッセージ内のバイト数がバスのバイトの整数倍でない場合、最後のサイクルで、バスの左から開始するメッセージの終了に必要なバイトのみ使用されます。 [Little Endian Support] がオンの場合、コアは s_axi_ufc_tx_tdata [(64n-1):0] を使用します。

表 2-7：UFC の I/O ポート (続き)

名前	方向	説明
s_axi_ufc_tx_tvalid	入力	s_axi_ufc_tx_tdata 上のデータが有効の場合にアサートされます (アクティブ High)。s_axi_ufc_tx_tready がアサートされている間にこの信号がディアサートされると、UFC メッセージにアイドルブロックが挿入されます。
m_axi_ufc_rx_tdata[0:(64n-1)] または m_axi_ufc_rx_tdata[(64n-1):0] <sup>(1)</sup>	出力	チャネルパートナーから送られる UFC メッセージデータです。[Little Endian Support] がオンの場合、コアは m_axi_ufc_rx_tdata[(64n-1):0] を使用します。
m_axi_ufc_rx_tvalid	出力	m_axi_ufc_rx_tdata ポートの値が有効な場合にアサートされます (アクティブ High)。この信号がアサートされない場合、m_axi_ufc_rx_tdata ポートのすべての値は無視されます。
m_axi_ufc_rx_tlast	出力	入力される UFC メッセージの終わりを示します (アクティブ High)。
m_axi_ufc_rx_tkeep[0:(8n-1)] または m_axi_ufc_rx_tkeep[(8n-1):0] <sup>(1)</sup>	出力	UFC メッセージの最後のワードで m_axi_ufc_rx_tdata ポートに現れる有効なバイト データ数を指定します。m_axi_ufc_rx_tlast がアサートされている場合のみ有効です。最大の UFC サイズは 256 バイトです。 [Little Endian Support] がオンの場合、コアは m_axi_ufc_rx_tkeep[(8n-1):0] を使用します。

1. n は、レーン数を表します。

## ネイティブ フロー制御インターフェイス

ネイティブ フロー制御 (NFC) を有効にしてコアを生成した場合、NFC インターフェイスが作成されます。このインターフェイスには、NFC メッセージを送信するために使用する要求ポートと肯定応答ポートがあります。

**注記 :** NFC 完了モードはストリーミング デザインには使用できません。

詳細は、[46 ページ](#)の「ネイティブ フロー制御」を参照してください。

[表 2-8](#) では NFC インターフェイスのポートについて説明しています。

表 2-8 : NFC の I/O ポート

名前	方向	説明
s_axi_nfc_tx_tvalid	入力	チャンネル パートナーへの NFC メッセージ送信が要求されると、アサートされます (アクティブ High)。s_axi_nfc_tx_tready がアサートされるまで High を保持する必要があります。
s_axi_nfc_tx_tready	出力	Aurora 64B/66B コアが NFC 要求を受信するとアサートされます (アクティブ High)。
s_axi_nfc_tx_tdata[0:15] または s_axi_nfc_tx_tdata[15:0]	入力	チャンネル パートナーから送られる UFC メッセージ データです。 [Little Endian Support] がオンの場合、コアは s_axi_nfc_tx_tdata[15:0] を使用します。 詳細は、「ネイティブ フロー制御」を参照してください。

## ユーザー K ブロック インターフェイス

ユーザー K ブロックをオンにしてコアを生成すると、ユーザー K インターフェイスが作成されます。ユーザー K ブロックは、制御ブロックを含む特殊なシングル ブロック コードで、Aurora インターフェイスではデコードされず、ユーザーに直接渡されます。これらのブロックは、アプリケーション固有の制御ファンクションをインプリメントするために使用できます。TX 側にはユーザー K ブロック送信を開始するのに使用する valid および ready ポートと、9 つあるユーザー K ブロックのうち送信する必要のあるものを示すためのブロック番号ポート (DATA) があります。ユーザー K データは、ユーザー K ブロック インターフェイスで ready 信号がアサートされた後に送信されます。また、これは標準データを受け付けなくなったことをユーザー インターフェイスに示し、ユーザー K データのユーザー K データ ポートへの書き込みが可能になります。ユーザー K ブロックはシングル ブロック コードです。

ユーザー K ブロック インターフェイスの RX 側には、ユーザー K ブロックの受信を示す RX valid 信号があります。フル デュプレックス モジュールには TX および RX 両方のユーザー K ポートがありますが、シングル デュプレックス モジュールには、サポートされる方向にデータ送信するために必要なインターフェイスのみが含まれます。

[表 2-9](#) ではユーザー K ブロック インターフェイスのポートについて説明しています。詳細は、[17 ページ](#)の「ユーザー K ブロック インターフェイス」を参照してください。

表 2-9: ユーザー K ブロックの I/O ポート

名前	方向	説明
<code>s_axi_user_k_tx_tdata[0:(64n-1)]</code> または <code>s_axi_user_k_tx_tdata[(64n-1):0]</code> <sup>(1)</sup>	入力	ユーザー K ブロック データは 64 ビットにアラインされています。[Little Endian Support] がオンの場合、コアは <code>s_axi_user_k_tx_tdata[(64n-1):0]</code> を使用します。 各レーンの信号マップは次のとおりです。 デフォルト: <code>s_axi_user_k_tx_tdata={4'h0, user_k_blk_no[0:4n-1], s_axi_user_k_tx_tdata[0:56n-1]}</code> リトルエンディアン形式: <code>s_axi_user_k_tx_tdata={s_axi_user_k_tx_tdata[56n-1:0], 4'h0, user_k_blk_no[4n-1:0]}</code>
<code>s_axi_user_k_tx_tvalid</code>	入力	<code>s_axi_user_k_tx_tdata</code> ポート上のユーザー K データが有効の場合にアサートされます (アクティブ High)。
<code>s_axi_user_k_tx_tready</code>	出力	Aurora 8B/10B コアが <code>s_axi_user_k_tx_tdata</code> インターフェイスからデータを読み出す準備が整うとアサート (アクティブ High) されます。
<code>m_axi_rx_user_k_tvalid</code>	出力	<code>m_axi_rx_user_k_tdata</code> ポート上のユーザー K データが有効の場合にアサートされます (アクティブ High)。
<code>m_axi_rx_user_k_tdata</code> または <code>m_axi_rx_user_k_tdata[(64n-1):0]</code> <sup>(1)</sup>	出力	Aurora レーンから受信するユーザー K ブロックは、64 ビットにアラインされています。[Little Endian Support] がオンの場合、コアは <code>m_axi_rx_user_k_tdata[(64n-1):0]</code> を使用します。 各レーンの信号マップは次のとおりです。 デフォルト: <code>m_axi_rx_user_k_tdata={4'h0, rx_user_k_blk_no[0:4n-1], m_axi_rx_user_k_tdata[0:56n-1]}</code> リトルエンディアン形式: <code>m_axi_rx_user_k_tdata={m_axi_rx_user_k_tdata[56n-1:0], 4'h0, rx_user_k_blk_no[4n-1:0]}</code>

1. n は、レーン数を表します。

## ステータスおよび制御ポート

表 2-10 では、フルデュプレックス コアのステータス ポートと制御ポートの機能について説明しています。

表 2-10: フルデュプレックス コアのステータス ポートと制御ポート

名前	方向	説明
<code>channel_up</code>	出力	Aurora 8B/10B チャンルの初期化が完了し、チャンネルがデータ送受信可能な状態になるとアサートされます (アクティブ High)。
<code>lane_up[0:m-1]</code> <sup>(1)</sup>	出力	レーンの初期化が正常に完了すると、各レーンに対してアサートされます (アクティブ High)。各ビットがそれぞれのレーンを表しています。Aurora 64B/66B コアは、すべての <code>lane_up</code> 信号が High に遷移した後にのみデータを受信できます。
<code>hard_err</code>	出力	ハード エラーを検出したことを示します (アクティブ High で、Aurora 64B/66B コアがリセットされるまでアサートされる)。詳細は、55 ページの表 2-21 を参照してください。
<code>loopback[2:0]</code>	入力	詳細は、『7 シリーズ FPGA GTX/GTH トランシーバー ユーザー ガイド』(UG476) [参照 4] または『UltraScale アーキテクチャ GTH トランシーバー ユーザー ガイド』(UG576) [参照 3] を参照してください。付録 E の「その他のリソースおよび法的通知」を参照してください。

表 2-10: フルデュプレックス コアのステータスポートと制御ポート (続き)

名前	方向	説明
power_down	入力	Aurora 64B/66B コアをリセット状態にします (アクティブ High)。
reset	入力	Aurora 64B/66B コアをリセットします。このポートは、デバウンス回路を介して最上位レベルへ接続され、すべての Aurora コア ロジックを組織的にリセットします。この信号は、user_clk の 6 サイクル間 user_clk を使用してデバウンス処理されます。詳細は、この製品ガイドの「リセットおよびパワーダウン」を参照してください。
soft_err	出力	入力されるシリアルストリームにソフトエラーが検出されたことを示します (アクティブ High で、user_clk の 1 サイクル間アサートされる)。詳細は、55 ページの表 2-21 を参照してください。
rxp[0:m-1]	入力	差動シリアルデータ入力ピンの正側です。
rxn[0:m-1]	入力	差動シリアルデータ入力ピンの負側です。
txp[0:m-1]	出力	差動シリアルデータ出力ピンの正側です。
txn[0:m-1]	出力	差動シリアルデータ出力ピンの負側です。
pma_init	入力	シリアルトランシーバー用の pma_init (アクティブ High) リセット信号は、デバウンス回路を介して最上位レベルへ接続されます。このポートは、トランシーバーのすべての PCS (物理コーディングサブレイヤー) と PMA (物理媒体アタッチメント) サブコンポーネントを組織的にリセットします。この信号は、init_clk の最低 6 サイクル間 init_clk_in を使用してデバウンス処理されます。詳細は、該当するトランシーバーのユーザーガイドの「リセット」セクションを参照してください。

表 2-10: フルデュプレックス コアのステータスポートと制御ポート (続き)

名前	方向	説明
init_clk	入力	<p>init_clk 信号は、pma_init 信号のレジスタへの格納およびデバウンス処理に使用されます。init_clk 信号は GT TX/RX のリセット FSM で使用され、リセット モードを初期化および実行、あるいは user_clk 生成の MMCM リセット機能を管理します。init_clk のレートは、50 ~ 200MHz の範囲が理想的です。デフォルトの init_clk 周波数は、コアによって 50MHz に設定されています。この値は、ご利用になるシステムに応じて XDC で変更する必要があります。</p> <p>Zynq-7000 および 7 シリーズ デバイスを使用するデザインの場合 : XDC サンプル デザイン ファイル (&lt;component name&gt;_clocks.xdc および &lt;component name&gt;_ooc.xdc) では init_clk が 200MHz に制約されています。さらに、STABLE_CLOCK_PERIOD パラメーターが 5ns に設定され、&lt;component name&gt;_core に同じ条件を反映します。&lt;component name&gt;_TB の INIT_CLOCKPERIOD パラメーターは 5 に設定され、init_clk を生成します。init_clk の周波数範囲は 50MHz ~ 200MHz に制約される必要があります。init_clk 周期の変更は、IP コアを適切に動作させるためにサンプル XDC (&lt;component name&gt;_clocks.xdc、&lt;component name&gt;_ooc.xdc、&lt;component name&gt;_core、および &lt;component name&gt;_TB) 内で行う必要があります。[Include Shared Logic in Core] をオンに選択してコアを生成した場合、init_clk ポートは差動 (init_clk_p、init_clk_n) になります。</p> <p>UltraScale アーキテクチャ デザインの場合 : init_clk の周波数は、TXUSERCLK の周波数と同じになる必要があります。値は 200MHz 以下にしてください。TXUSERCLK の周波数は、ラインレートと内部データパス幅に依存します。詳細は、『UltraScale FPGA GTH トランシーバー ユーザー ガイド』(UG576) [参照 3] を参照してください。Aurora 64B/66B コアは、GT の内部データパス幅を 32 ビットとして構成します。この init_clk_in は、GTHE3_CHANNEL の DRP ポートの DRPCLK にも接続されます。IP コアを適切に動作させるには、init_clk_in 周期の変更はサンプル XDC ファイル (&lt;component name&gt;_clocks.xdc、&lt;component name&gt;_ooc.xdc、および &lt;component name&gt;_TB) 内で行う必要があります。</p>

**注記 :**

1. m は、GTX または GTH トランシーバーの数を表しています。

表 2-11 では、シンプレックス TX コアのステータスポートと制御ポートの機能について説明しています。

**注記 :** Aurora 64B/66B チャネルでは、CHANNEL\_UP 信号がアサートされた後、ユーザー主導の要求を受けるにはより長い時間が必要です。所要時間は、コア コンフィギュレーションによって異なります。ただし、コアはチャネルがデータ転送可能な状態になるまで、いかなるインターフェイスに対しても TREADY をアサートしません。



**推奨 :** Aurora チャネルを介して要求を開始する前に、データ インターフェイスの s\_axi\_tx\_tready 信号がアサートされていることを確認してください。s\_axi\_tx\_tready 信号がアサートされる前に開始されたフロー制御要求は、コアで処理されません。

表 2-11 : シンプレックス TX または TX/RX シンプレックス コアのステータス ポートと制御ポート

名前	方向	説明
tx_channel_up	出力	Aurora 8B/10B チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
tx_lane_up[0:m-1] <sup>(1)</sup>	出力	レーンの初期化が正常に完了すると、各レーンに対してアサートされます (アクティブ High)。各ビットがそれぞれのレーンを表しています。Aurora 64B/66B コアは、すべての tx_lane_up 信号が High に遷移した後にのみデータを送信できます。
tx_hard_err	出力	ハード エラーを検出したことを示します (アクティブ High で、Aurora 64B/66B コアがリセットされるまでアサートされる)。詳細は、55 ページの表 2-21 を参照してください。
power_down	入力	Aurora 64B/66B コアをリセット状態にします (アクティブ High)。
reset	入力	Aurora 64B/66B コアをリセットします (アクティブ High)。この信号は、user_clk と同期する必要があり、少なくとも user_clk の 6 クロック サイクル間アサートされる必要があります。
tx_soft_err	出力	送信ロジックにソフト エラーが検出されたことを示します (アクティブ High で、user_clk の 1 サイクル間アサートされる)。詳細は、55 ページの表 2-21 を参照してください。
txp[0:m-1]	出力	差動シリアル データ出力ピンの正側です。
txn[0:m-1]	出力	差動シリアル データ出力ピンの負側です。
pma_init	入力	シリアル トランシーバーの pma_init (アクティブ High) リセット信号は、デバウンス回路を介して最上位レベルへ接続されます。このポートは、トランシーバーのすべての PCS (物理コーディング サブレイヤー) と PMA (物理媒体アタッチメント) サブコンポーネントを組織的にリセットします。 この信号は、init_clk の最低 6 サイクル間 init_clk_in を使用してデバウンス処理されます。詳細は、該当する トランシーバーのユーザー ガイドの「リセット」セクションを参照してください。

表 2-11 : シンプレックス TX または TX/RX シンプレックス コアのステータス ポートと制御ポート (続き)

名前	方向	説明
init_clk	入力	<p>init_clk 信号は、pma_init 信号のレジスタへの格納およびデバウンス処理に使用されます。init_clk 信号は GT TX のリセット FSM で使用され、リセット モードを初期化および実行、あるいは user_clk 生成の MMCM リセット機能を管理します。init_clk のレートは、50 ~ 200MHz の範囲が理想的です。デフォルトの init_clk 周波数は、コアによって 50MHz に設定されています。この値は、ご利用になるシステムに応じて XDC で変更する必要があります。</p> <p>Zynq-7000 および 7 シリーズ デバイスを使用するデザインの場合 : XDC サンプル デザイン ファイル (&lt;component name&gt;_clocks.xdc および &lt;component name&gt;_ooc.xdc) では init_clk が 200MHz に制約されています。さらに、STABLE_CLOCK_PERIOD パラメーターが 5ns に設定され、&lt;component name&gt;_core に同じ条件を反映します。&lt;component name&gt;_TB の INIT_CLOCKPERIOD パラメーターは 5 に設定され、init_clk を生成します。init_clk の周波数範囲は 50MHz ~ 200MHz に制約される必要があります。init_clk 周期の変更は、IP コアを適切に動作させるためにサンプル XDC (&lt;component name&gt;_clocks.xdc、&lt;component name&gt;_ooc.xdc、&lt;component name&gt;_core、および &lt;component name&gt;_TB) 内で行う必要があります。[Include Shared Logic in Core] をオンに選択してコアを生成した場合、init_clk ポートは差動 (init_clk_p、init_clk_n) になります。</p> <p>UltraScale アーキテクチャ デザインの場合 : init_clk の周波数は、TXUSERCLK の周波数と同じになる必要があります、値は 200MHz 以下にしてください。TXUSERCLK の周波数は、ライン レートと内部データパス幅に依存します。詳細は、『UltraScale FPGA GTH トランシーバー ユーザー ガイド』(UG576) [参照 3] を参照してください。Aurora 64B/66B コアは、GT の内部データパス幅を 32 ビットとして構成します。この init_clk_in は、GTHE3_CHANNEL の DRP ポートの DRPCLK にも接続されず、IP コアを適切に動作させるには、init_clk_in 周期の変更はサンプル XDC ファイル (&lt;component name&gt;_clocks.xdc、&lt;component name&gt;_ooc.xdc、および &lt;component name&gt;_TB) 内で行う必要があります。</p>

注記 :

1. m は、GTX または GTH トランシーバーの数を表しています。

表 2-12 では、シンプレックス RX コアのステータス ポートと制御ポートの機能について説明しています。詳細は、18 ページの「ステータスおよび制御ポート」を参照してください。

表 2-12 : シンプレックス RX または TX/RX シンプレックス コアのステータス ポートと制御ポート

名前	方向	説明
rx_channel_up	出力	Aurora 8B/10B チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
rx_lane_up[0:m-1] <sup>(1)</sup>	出力	レーンの初期化が正常に完了すると、各レーンに対してアサートされます (アクティブ High)。各ビットがそれぞれのレーンを表しています。Aurora 64B/66B コアは、すべての rx_lane_up 信号が High に遷移した後にデータを受信できます。
rx_hard_err	出力	ハード エラーを検出したことを示します (アクティブ High で、Aurora 64B/66B コアがリセットされるまでアサートされる)。詳細は、55 ページの表 2-21 を参照してください。
power_down	入力	Aurora 64B/66B コアをリセット状態にします (アクティブ High)。

表 2-12 : シンプレックス RX または TX/RX シンプレックス コアのステータス ポートと制御ポート (続き)

名前	方向	説明
reset	入力	Aurora 64B/66B コアをリセットします (アクティブ High)。この信号は、user_clk と同期する必要があり、少なくとも user_clk の 6 クロック サイクル間アサートされる必要があります。
rx_soft_err	出力	受信ロジックにソフト エラーが検出されたことを示します (アクティブ High で user_clk の 1 サイクル間アサートされる)。詳細は、55 ページの表 2-21 を参照してください。
rxp[0:m-1]	入力	差動シリアル データ入力ピンの正側です。
rxn[0:m-1]	入力	差動シリアル データ入力ピンの負側です。
pma_init	入力	シリアル トランシーバー用の pma_init (アクティブ High) リセット信号は、デバウンス回路を介して最上位レベルへ接続されます。このポートは、トランシーバーのすべての PCS (物理コーディング サブレイヤー) と PMA (物理媒体アタッチメント) サブコンポーネントを組織的にリセットします。この信号は、init_clk の最低 6 サイクル間 init_clk_in を使用してデバウンス処理されます。
init_clk	入力	<p>init_clk 信号は、pma_init 信号のレジスタへの格納およびデバウンス処理に使用されます。init_clk 信号は GT TX のリセット FSM で使用され、リセット モードを初期化および実行、あるいは user_clk 生成の MMCM リセット機能を管理します。init_clk のレートは、50 ~ 200MHz の範囲が理想的です。</p> <p>Zynq-7000 および 7 シリーズ デバイスを使用するデザインの場合 : XDC サンプル デザイン ファイル (&lt;component name&gt;_clocks.xdc および &lt;component name&gt;_ooc.xdc) では init_clk が 200MHz に制約されています。さらに、STABLE_CLOCK_PERIOD パラメーターが 5ns に設定され、&lt;component name&gt;_core に同じ条件を反映します。&lt;component name&gt;_TB の INIT_CLOCKPERIOD パラメーターは 5 に設定され、init_clk を生成します。init_clk の周波数範囲は 50MHz ~ 200MHz に制約される必要があります。init_clk 周期の変更は、IP コアを適切に動作させるためにサンプル XDC (&lt;component name&gt;_clocks.xdc、&lt;component name&gt;_ooc.xdc、&lt;component name&gt;_core および &lt;component name&gt;_TB) 内で行う必要があります。[Include Shared Logic in Core] をオンに選択してコアを生成した場合、init_clk ポートは差動 (init_clk_p、init_clk_n) になります。</p> <p>UltraScale アーキテクチャ デザインの場合 : init_clk の周波数は、TXUSERCLK の周波数と同じになる必要があります、値は 200MHz 以下にしてください。TXUSERCLK の周波数は、ライン レートと内部データパス幅に依存します。詳細は、『UltraScale FPGA GTH トランシーバー ユーザー ガイド』(UG576) [参照 3] を参照してください。Aurora 64B/66B コアは、GT の内部データパス幅を 32 ビットとして構成します。この init_clk_in は、GTHE3_CHANNEL の DRP ポートの DRPCLK にも接続されます。IP コアを適切に動作させるには、init_clk_in 周期の変更はサンプル XDC ファイル (&lt;component name&gt;_clocks.xdc、&lt;component name&gt;_ooc.xdc、および &lt;component name&gt;_TB) 内で行う必要があります。</p>

**注記 :**

1. m は、GTX または GTH トランシーバーの数を表しています。



注意 : デフォルトの `init_clk` 周波数は、7 シリーズ デバイスのコアによって 50MHz に設定されています。この値は、ご利用になるシステムに応じて XDC で変更する必要があります。

## GTX および GTH トランシーバーのインターフェイス

このインターフェイスには、GTX/GTH トランシーバーのシリアル I/O ポートおよび Aurora 64B/66B コアの制御ポートとステータスポートがあります。このインターフェイスは、リセット、ループバック、およびパワーダウンなどの制御機能へのユーザーアクセスを提供します。DRP インターフェイスを利用し、AXI4-Lite または Native DRP インターフェイスを介してシリアル トランシーバーのパラメーターや設定にアクセスあるいは、それらを変更できます。

表 2-13 : トランシーバーの DRP ポート

名前	方向	説明
<code>rxp[0:m-1]</code> <sup>(1)</sup>	入力	差動シリアル データ入力ピンの正側です。
<code>rxn[0:m-1]</code>	入力	差動シリアル データ入力ピンの負側です。
<code>txp[0:m-1]</code>	出力	差動シリアル データ出力ピンの正側です。
<code>txn[0:m-1]</code>	出力	差動シリアル データ出力ピンの負側です。
<code>loopback[2:0]</code>	入力	トランシーバーのループバック ポートです。ループバック テスト モード コンフィギュレーションに関しては、該当するトランシーバーのユーザー ガイドを参照してください。
<code>pma_init</code>	入力	トランシーバー用の非同期リセット信号です。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。
<code>tx_lock</code>	出力	入力されるシリアル トランシーバーの <code>refclk</code> がトランシーバーの PLL によってロックされていることを示します。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。
7 シリーズおよび UltraScale FPGA トランシーバーの DRP ポート <sup>(2)</sup>		
<code>drpaddr_in</code>	入力	DRP アドレス バス
<code>drp_clk_in</code>	入力	DRP インターフェイス クロック
<code>drpdi_in</code>	入力	FPGA ロジックからトランシーバーへコンフィギュレーション データを書き込むためのデータ バスです。
<code>drpdo_out</code>	出力	トランシーバーから FPGA ロジックへコンフィギュレーション データを読み出すためのデータ バスです。
<code>drpen_in</code>	入力	DRP のイネーブル信号です。
<code>drprdy_out</code>	出力	DRP 書き込み処理が完了し、読み出しデータが有効であることを示します。
<code>drpwe_in</code>	入力	DRP の書き込みイネーブルです。

1. `m` は、GTX または GTH トランシーバーの数を表しています。
2. DRP の詳細は、該当するトランシーバーのユーザー ガイドを参照してください。
3. Vivado IDE で [Additional transceiver control and status ports] チェックボックスをオンにした場合、トランシーバー デバッグ ポートが有効になります。

表 2-14 に示すポートは、Aurora 64B/66B コアをコンフィギュレーションする際にダイアログ ボックスで [Additional transceiver control and status ports] をオンに選択した場合のみ表示されます。詳細は、『7 シリーズ FPGA GTX/GTH トランシーバー ユーザー ガイド』(UG476) [参照 4] および 『UltraScale アーキテクチャ GTH トランシーバー ユーザー ガイド』(UG576) [参照 3] を参照してください。

表 2-14 : 7 シリーズおよび Zynq-7000 デバイス トランシーバーのデバッグ ポート

名前	方向	説明
gt<lane>_eyescandataerror_out	出力	COUNT または ARMED ステートのときに (マスクされていない) エラーが発生すると、rec_clk の 1 サイクル間 High になります。 デュプレックスおよび RX のみシンプレックスの場合に利用可能です。詳細は、該当するトランシーバーのユーザーガイドを参照してください。
gt<lane>_eyescanreset_in	入力	EYESCAN のリセット シーケンスを開始するため、このポートは High 駆動された後ディアサートされます。 デュプレックスおよび RX のみシンプレックスの場合に利用可能です。詳細は、該当するトランシーバーのユーザーガイドを参照してください。
gt<lane>_eyescantrigger_in	入力	トリガー イベントを発生させます。 デュプレックスおよび RX のみシンプレックスの場合に利用可能です。詳細は、該当するトランシーバーのユーザーガイドを参照してください。
gt<lane>_rxcdrhold_in	入力	CDR 制御ループを停止状態に保持します。 デュプレックスおよび RX のみシンプレックスの場合に利用可能で、Zynq-7000 および 7 シリーズ デバイスの GTX および GTH トランシーバーにのみ適用できます。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。
gt<lane>_rxlpmhfovrden_in	入力	OVRDEN RX LPM <ul style="list-style-type: none"> <li>2'b00 : KH の周波数増加ループを適応</li> <li>2'b10 : 現在の適応値を固定</li> <li>2'bx1 : RXLPM_HF_CFG 属性に応じて KH 値を上書き</li> </ul> デュプレックスおよび RX のみシンプレックスの場合に利用可能で、Zynq-7000 および 7 シリーズ デバイスの GTX および GTH トランシーバーにのみ適用できます。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。
gt<lane>_rxdfegchold_in	入力	HOLD RX DFE <ul style="list-style-type: none"> <li>2'b00 : AGC (Automatic Gain Control : 自動ゲイン制御) ループを適応</li> <li>2'b10 : 現在の AGC 適応値を固定</li> <li>2'bx1 : RX_DFE_GAIN_CFG 属性に応じて AGC 値を上書き</li> </ul> デュプレックスおよび RX のみシンプレックスの場合に利用可能で、Zynq-7000 および 7 シリーズ デバイスの GTX および GTH トランシーバーにのみ適用できます。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。

表 2-14 : 7 シリーズおよび Zynq-7000 デバイス トランシーバーのデバッグ ポート (続き)

名前	方向	説明
gt<lane>_rxdfegcovrden_in	入力	<p>OVRDEN RX DFE</p> <ul style="list-style-type: none"> <li>• 2'b00 : AGC (Automatic Gain Control : 自動ゲイン制御) ループを適応</li> <li>• 2'b10 : 現在の AGC 適応値を固定</li> <li>• 2'bx1 : RX_DFE_GAIN_CFG 属性に応じて AGC 値を上書き</li> </ul> <p>デュプレックスおよび RX のみシンプレックスの場合に利用可能で、Zynq-7000 および 7 シリーズ デバイスの GTX および GTH トランシーバーにのみ適用できます。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>
gt<lane>_rxdfelfhold_in	入力	<p>1'b1 に設定すると、現在の低周波ブースト値が保持されます。1'b0 に設定すると、低周波ブースト値が適用されます。</p> <p>デュプレックスおよび RX のみシンプレックスの場合に利用可能で、7 シリーズ デバイスの GTP トランシーバーにのみ適用できます。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>
gt<lane>_rxdfelpmreset_in	入力	<p>DFE のリセット シーケンスを開始するため、このポートは High 駆動された後ディアサートされます。</p> <p>デュプレックスおよび RX のみシンプレックスの場合に利用可能で、Zynq-7000 および 7 シリーズ デバイスの GTX および GTH トランシーバーにのみ適用できます。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>
gt<lane>_rxlplmfkloverden_in	入力	<p>OVRDEN RX LPM</p> <ul style="list-style-type: none"> <li>• 2'b00 : KL の周波数低減ループを適応</li> <li>• 2'b10 : 現在の適応値を固定</li> <li>• 2'bx1 : RXLPM_LF_CFG 属性に応じて KL 値を上書き</li> </ul> <p>デュプレックスおよび RX のみシンプレックスの場合に利用可能で、Zynq-7000 および 7 シリーズ デバイスの GTX および GTH トランシーバーにのみ適用できます。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>
gt<lane>_rxlpmen_in	入力	<p>RX データパス</p> <ul style="list-style-type: none"> <li>• 0 : DFE</li> <li>• 1 : LPM</li> </ul> <p>デュプレックスおよび RX のみシンプレックスの場合に利用可能で、Zynq-7000 および 7 シリーズ デバイスの GTX および GTH トランシーバーにのみ適用できます。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>

表 2-14 : 7 シリーズおよび Zynq-7000 デバイス トランシーバーのデバッグ ポート (続き)

名前	方向	説明
gt<lane>_rxmonitorout_out	入力	<p>GTX トランシーバー :</p> <ul style="list-style-type: none"> <li>• RXDFEVP[6:0] = RXMONITOROUT[6:0]</li> <li>• RXDFEUT[6:0] = RXMONITOROUT[6:0]</li> <li>• RXDFEAGC[4:0] = RXMONITOROUT[4:0]</li> </ul> <p>GTH トランシーバー :</p> <ul style="list-style-type: none"> <li>• RXDFEVP[6:0] = RXMONITOROUT[6:0]</li> <li>• RXDFEUT[6:0] = RXMONITOROUT[6:0]</li> <li>• RXDFEAGC[3:0] = RXMONITOROUT[4:1]</li> </ul> <p>デュプレックスおよび RX のみシンプレックスの場合に利用可能で、Zynq-7000 および 7 シリーズ デバイスの GTX および GTH トランシーバーにのみ適用できます。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>
gt<lane>_rxmonitorsel_in	入力	<p>rxmonitorout[6:0] の信号を選択します。</p> <ul style="list-style-type: none"> <li>• 2'b00 : 予約</li> <li>• 2'b01 : AGC ループを選択</li> <li>• 2'b10 : UT ループを選択</li> <li>• 2'b11 : VP ループを選択</li> </ul> <p>デュプレックスおよび RX のみシンプレックスの場合に利用可能で、Zynq-7000 および 7 シリーズ デバイスの GTX および GTH トランシーバーにのみ適用できます。詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>
gt<lane>_txpostcursor_in	入力	<p>トランスミッターのポストカーソル TX プリエンファシスを指定します。</p> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p> <p>詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>
gt<lane>_txdiffctrl_in	入力	<p>ドライバーの強度を指定します。</p> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p> <p>詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>
gt<lane>_txmaincursor_in	入力	<p>TX_MAINCURSOR_SEL 属性が 1'b1 に設定されている場合、メインカーソルの係数を直接設定できます。</p> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p> <p>詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>
gt<lane>_txpolarity_in	入力	<p>出力データの極性の反転に使用します。</p> <ul style="list-style-type: none"> <li>• 0 : 反転しない。TXP は正、TXN は負。</li> <li>• 1 : 反転する。TXP は負、TXN は正。</li> </ul> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p> <p>詳細は、該当するトランシーバーのユーザー ガイドを参照してください。</p>

表 2-14 : 7 シリーズおよび Zynq-7000 デバイス トランシーバーのデバッグ ポート (続き)

名前	方向	説明
gt<lane>_txpmareset_in	入力	TX PMA のリセットに使用します。TX PMA のリセットシーケンスを開始するため、このポートは High 駆動された後ディアサートされます。このポートを High 駆動すると TX PMA と TX PCS の両方がリセットされます。
gt<lane>_txpcsreset_in	入力	TX PCS のリセットに使用します。PCS のリセットシーケンスを開始するため、このポートは High 駆動された後ディアサートされます。このポートを High 駆動すると TX PCS のみリセットされます。
gt<lane>_txresetdone_out	出力	GTX/GTH トランシーバー TX がリセットを完了して使用可能になるとアクティブ High になります。gtxtxreset が High 駆動すると、このポートは Low に遷移し、GTX/GTH トランシーバーの TX で txuserddy 信号の High 駆動が検出されるまで High にはなりません。
gt<lane>_rxpmareset_in	入力	RX PMA のリセットシーケンスを開始するため、このポートは High 駆動された後ディアサートされます。詳細は、『7 シリーズ FPGA GTX/GTH トランシーバー ユーザーガイド』[参照 4] を参照してください。
gt<lane>_rxpcsreset_in	入力	RX PMA のリセットシーケンスを開始するため、このポートは High 駆動された後ディアサートされます。詳細は、『7 シリーズ FPGA GTX/GTH トランシーバー ユーザーガイド』[参照 4] を参照してください。rxpcsreset は、rxuserddy が High に遷移するまでリセットプロセスを開始しません。
gt<lane>_rxbufreset_in	入力	RX エラスティックバッファのリセットシーケンスを開始するため、このポートは High 駆動された後ディアサートされます。rxbufreset を High 駆動すると RX エラスティックバッファのみがリセットされます。
gt<lane>_rxresetdone_out	出力	GTX/GTH トランシーバー RX がリセットを完了して使用可能になると、High になります。gtxrxreset が High の場合は Low 駆動します。また、rxuserddy が High に遷移するまで High 駆動しません。
gt<lane>_txbufstatus_out[1:0]	出力	txbufstatus [1] : TX バッファのオーバーまたはアンダーフロー txbufstatus [1] は High になると、TX バッファがリセットされるまで High が保持されます。 <ul style="list-style-type: none"> <li>• 1 : TX FIFO のオーバーフロー/アンダーフロー</li> <li>• 0 : TX FIFO のオーバーフロー/アンダーフローエラーなし txbufstatus [0] : TX バッファのステータスです。</li> <li>• 1 : TX FIFO は 1/2 以上</li> <li>• 0 : TX FIFO は 1/2 未満</li> </ul>
gt<lane>_rxbufstatus_out[2:0]	出力	RX バッファのステータスです。 <ul style="list-style-type: none"> <li>• 000b : 通常条件</li> <li>• 001b : バッファのバイト数が CLK_COR_MIN_LAT 未満</li> <li>• 010b : バッファのバイト数が CLK_COR_MAX_LAT より多い</li> <li>• 101b : RX エラスティックバッファがアンダーフロー</li> <li>• 110b : RX エラスティックバッファがオーバーフロー</li> </ul>

表 2-14 : 7 シリーズおよび Zynq-7000 デバイス トランシーバーのデバッグ ポート (続き)

名前	方向	説明
gt<lane>_cplllock_out	出力	High の場合、この PLL 周波数ロック信号は、PLL 周波数があらかじめ判断した耐性範囲内であることを示します。この条件が満たされるまで、トランシーバーおよびそのクロック出力は信頼できません。
gt_qplllock<quad>	出力	High の場合、この PLL 周波数ロック信号は、PLL 周波数があらかじめ判断した耐性範囲内であることを示します。この条件が満たされるまで、トランシーバーおよびそのクロック出力は信頼できません。
gt<lane>_precursor_in	入力	トランスミッターのプリカーソル TX プリエンファシスを指定します。 デュプレックスおよび TX のみシンプレックスの場合に利用可能です。詳細は、該当するトランシーバーのユーザーガイドを参照してください。
gt<lane>_txprbsforceerr_in	入力	High に駆動されると、PRBS トランスミッターでエラー挿入が有効になります。アサート中は、出力データパターンにエラーが挿入されます。txprbsssel が 000 に設定されている場合は、TXDATA への影響はありません。
gt<lane>_txprbsssel_in[2:0]	入力	トランスミッター PRBS ジェネレーターのテストパターンを制御します。 <ul style="list-style-type: none"> <li>• 000 : 標準動作モード (テストパターン生成はオフ)</li> <li>• 001 : PRBS-7</li> <li>• 010 : PRBS-15</li> <li>• 011 : PRBS-23</li> <li>• 100 : PRBS-31</li> <li>• 101 : PCI Express 準拠パターン。20 ビットおよび 40 ビットモードの場合のみ</li> <li>• 110 : 2UI の方形波 (0 と 1 を交互に配列)</li> <li>• 111 : 16-UI、20-UI、32-UI、または 40-UI の方形波 (データ幅に基づく)</li> </ul>
gt<lane>_rxprbsssel_in[2:0]	入力	レシーバーの PRBS チェッカーのテストパターンを制御します。 有効な設定は次のとおりです。 <ul style="list-style-type: none"> <li>• 000 : 通常動作モード (PRBS チェッカーはオフ)</li> <li>• 001 : PRBS-7</li> <li>• 010 : PRBS-15</li> <li>• 011 : PRBS-23</li> <li>• 100 : PRBS-31</li> </ul> PRBS 以外のパターンに対してチェックは実行されません。PRBS チェッカーは、現在のサイクルからのデータを使用して、次のサイクルで想定されるデータを生成するため、シングルエラーが PRBS のバーストエラーとなって生じます。
gt<lane>_rxprbserr_out	出力	PRBS エラーが発生したことを示すステータス出力です。
gt<lane>_rxprbscntreset_in	入力	PRBS エラー カウンターをリセットします。
GTX トランシーバー : gt<lane>_dmonitorout_out[7:0] GTH トランシーバー : gt<lane>_dmonitorout_out [14:0]	出力	デジタル モニター出力バスです。

## 注記 :

1. <lane> には、0 から AURORA\_LANES までの値が入ります。
2. UltraScale™ デバイスを使用するデザインの場合、シングル レーン コアではオプションのトランシーバー ポートの接頭語が gt<lane> から gt へ変更され、接尾語 \_in および \_out は削除されます。マルチレーン コアの場合、オプションのトランシーバー デバッグ ポートの接頭語 gt(n) が 1 つのポートに集約されます。

表 2-15 : UltraScale アーキテクチャのトランシーバー デバッグ ポート (1)

名前	方向	説明
gt_cpplllock	出力	High の場合、この PLL 周波数ロック信号は、PLL 周波数があらかじめ判断した耐性範囲内であることを示します。この条件が満たされるまで、トランシーバーおよびそのクロック出力は信頼できません。
gt_dmonitorout	出力	デジタル モニター出力バスです。 デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_eyescandataerror	出力	COUNT または ARMED ステートのときに (マスクされていない) エラーが発生すると、REC_CLK の 1 サイクル間 High になります。 デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_eyescanreset	入力	EYESCAN のリセット プロセスを開始するため、このポートは High 駆動された後ディアサートされます。 デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_eyescantrigger	入力	トリガー イベントを発生させます。 デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_gtrxreset		gtx_wrapper の PMA_INIT に接続されます。
gt_gttxreset		gtx_wrapper の PMA_INIT に接続されます。
gt_loopback		gtx_wrapper の LOOPBACK に接続されます。
gt_rxbufreset	入力	RX エラスティック バッファのリセットプロセスを開始するため、このポートは High 駆動された後ディアサートされます。 シングル モードまたはシーケンシャル モードのいずれかでこのポートを High 駆動すると RX エラスティック バッファのみがリセットされます。 デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxbufstatus	出力	RX バッファのステータスです。 <ul style="list-style-type: none"> <li>• 000b : 通常条件</li> <li>• 001b : バッファのバイト数が CLK_COR_MIN_LAT 未満</li> <li>• 010b : バッファのバイト数が CLK_COR_MAX_LAT より多い</li> <li>• 101b : RX エラスティック バッファがアンダーフロー</li> <li>• 110b : RX エラスティック バッファがオーバーフロー</li> </ul> デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxcdrhold	入力	CDR 制御ループを停止状態に保持します。 デュプレックスおよび RX のみシンプレックスの場合に利用可能です。

表 2-15 : UltraScale アーキテクチャのトランシーバー デバッグ ポート (続き)<sup>(1)</sup>

名前	方向	説明
gt_rxdfelprmreset	入力	DFE のリセット シーケンスを開始するため、このポートは High 駆動された後ディアサートされます。デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxlpmen	入力	RX データバス 0 : DFE 1 : LPM デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxpcsreset	入力	RX PCS のリセット シーケンスを開始するため、このポートは High 駆動された後ディアサートされます。デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxpmareset	入力	RX PMA のリセット シーケンスを開始するため、このポートは High 駆動された後ディアサートされます。デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxpmaresetdone	出力	RX PMA リセットが完了すると、アクティブ High になります。GTRXRESET または RXPMARESET がアサートされると、Low に駆動します。デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxpolarity		内部で使用されます。
gt_rxprbscntreset	入力	PRBS エラー カウンターをリセットします。デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxprbserr	出力	PRBS エラーが発生したことを示すステータス出力です。デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxprbsssel	入力	トランシーバー チャンネル プリミティブの RXPRBSSEL へ接続されます。デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxrate	入力	GTH トランシーバー RX で有効な PLL 分周器の値を自動的に変更するダイナミック ピンです。これらのポートは、PCI® Express やその他の規格で使用されます。デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_rxresetdone	出力	アサートされると、GTH トランシーバー RX がリセットを完了して使用可能になったことを示す、アクティブ High になります。シーケンシャル モードでは、GTRXRESET が High の場合に Low 駆動します。また、RXUSERRDY が High に遷移するまで High 駆動しません。シングル モードでは、RX リセットがアサートされた場合に Low 駆動します。この信号は、すべての RX リセットがディアサートされ、RXUSERRDY がアサートされるまでアサートされません。デュプレックスおよび RX のみシンプレックスの場合に利用可能です。

表 2-15 : UltraScale アーキテクチャのトランシーバー デバッグ ポート (続き)<sup>(1)</sup>

名前	方向	説明
gt_txbufstatus	出力	<p>TXBUFSTATUS は TX バッファまたは TX 非同期ギアボックスのステータスを提供します。TX 非同期ギアボックス使用時のポートのステータスは次のとおりです。</p> <p>ビット 1 :</p> <ul style="list-style-type: none"> <li>• 0 : TX 非同期ギアボックスの FIFO オーバーフローなし</li> <li>• 1 : TX 非同期ギアボックスの FIFO オーバーフローが発生</li> </ul> <p>ビット 0 :</p> <ul style="list-style-type: none"> <li>• 0 : TX 非同期ギアボックスの FIFO アンダーフローなし</li> <li>• 1 : TX 非同期ギアボックスの FIFO アンダーフローが発生</li> </ul> <p>ポートは High になると、TX 非同期ギアボックスがリセットされるまで High が保持されます。</p> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p>
gt_txdiffctrl	入力	<p>ドライバーの強度を指定します。デフォルト値はユーザーが指定します。</p> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p>
gt_txpcsreset	入力	<p>TX PCS のリセットに使用します。PCS のリセットシーケンスを開始するため、このポートは High 駆動された後ディアサートされます。シーケンシャルモードの場合、このポートを High 駆動すると TX PCS のみリセットされます。</p> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p>
gt_txpmareset	入力	<p>TX PMA のリセットに使用します。TX PMA のリセットシーケンスを開始するため、このポートは High 駆動された後ディアサートされます。シーケンシャルモードの場合、このポートを High 駆動すると TX PMA と TX PCS の両方がリセットされます。</p> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p>
gt_txpolarity	入力	<p>出力データの極性の反転に TXPOLARITY を使用します。</p> <ul style="list-style-type: none"> <li>• 0 : 反転しない。TXP は正、TXN は負。</li> <li>• 1 : 反転する。TXP は負、TXN は正。</li> </ul> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p>
gt_txpostcursor	入力	<p>トランスミッターのポストカーソル TX プリエンファシスを指定します。デフォルト値はユーザーが指定します。</p> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p>
gt_txprbsforceerr	入力	<p>High に駆動されると、PRBS トランスミッターでエラー挿入が有効になります。アサート中は、出力データパターンにエラーが挿入されます。TXPRBSSEL が 4'b0000 に設定されている場合は、TXDATA への影響はありません。</p> <p>デュプレックスおよび TX のみシンプレックスの場合に利用可能です。</p>

表 2-15 : UltraScale アーキテクチャのトランシーバー デバッグ ポート (続き)<sup>(1)</sup>

名前	方向	説明
gt_txprbssel	入力	トランスミッター PRBS ジェネレーターのテスト パターンを制御します <ul style="list-style-type: none"> <li>4'b0000 : 標準動作モード (テスト パターン生成はオフ)</li> <li>4'b0001 : PRBS-7</li> <li>4'b0010 : PRBS-9</li> <li>4'b0011 : PRBS-15</li> <li>4'b0100 : PRBS-23</li> <li>4'b0101 : PRBS-31</li> <li>4'b1000 : PCI Express 準拠パターン。内部データ幅が 20 ビットおよび 40 ビットのモードのときにのみ有効。</li> <li>4'b1001 : 2UI の方形波 (0 と 1 を交互に配列)</li> <li>4'b1010 : 16UI、20UI、32UI、または 40UI の方形波 (内部データ幅に基づく)</li> </ul> デュプレックスおよび TX のみシンプレックスの場合に利用可能です。
gt_txprecursor	入力	トランスミッターのプリカーソル TX プリエンファシスを指定します。デフォルト値はユーザーが指定します。 デュプレックスおよび TX のみシンプレックスの場合に利用可能です。
gt_txresetdone	出力	GTH トランシーバー TX がリセットを完了して使用可能になるとアクティブ High になります。GTTXRESET が High 駆動すると、このポートは Low に遷移し、GTH トランシーバー TX で TXUSERRDY 信号の High 駆動が検出されるまで High になりません。 デュプレックスおよび TX のみシンプレックスの場合に利用可能です。
gt_dmonitorout	出力	デジタル モニター出力バスです。 デュプレックスおよび RX のみシンプレックスの場合に利用可能です。
gt_qplllock	出力	High の場合、この PLL 周波数ロック信号は、PLL 周波数があらかじめ判断した耐性範囲内であることを示します。この条件が満たされるまで、トランシーバーおよびそのクロック出力は信頼できません。

1. これらのデバッグ ポートの詳細は、『UltraScale FPGA GTH トランシーバー ユーザー ガイド』(UG576) [参照 3] を参照してください。

## クロック インターフェイス



**重要** : このインターフェイスは、正しい Aurora 64B/66B コア動作を得るのに最も重要です。クロック インターフェイスには、GTX/GTH トランシーバーを駆動する基準クロック用のポートと、Aurora 64B/66B コアがアプリケーションロジックと共有するパラレルクロック用のポートがあります。

表 2-16 では、Aurora 64B/66B コアのクロック ポートについて説明しています。GTX および GTH トランシーバー デザインでは、基準クロックを GTXQ/GTHQ から供給可能で、これらは GTX/GTH トランシーバーの差動入力クロックとなります。GTX/GTH トランシーバー用の基準クロックは、clk<sub>in</sub> ポートを介して供給されます。クロック インターフェイスの詳細は、69 ページの「クロッキング」を参照してください。

表 2-16 : GTX/GTH ベース Aurora 64B/66B コアのクロック ポート

名前	方向	説明
mmcm_not_locked	入力	<p>7 シリーズおよび Zynq-7000 デバイスの場合 : Aurora 64B/66B コア用のクロック信号の生成に MMCM を使用する場合、mmcm_not_locked 信号をシリアル トランシーバー PLL の反転された locked 信号に接続する必要があります。Aurora 64B/66B コアで提供されているクロック モジュールは、クロックの分周に PLL を使用します。クロック モジュールからの mmcm_not_locked 信号は、Aurora 64B/66B コアの mmcm_not_locked 信号に接続する必要があります。</p> <p>UltraScale デバイスの場合 : mmcm_not_locked は、&lt;=:USER_COMPONENT_NAME:&gt;_ultrascale_tx_userclk モジュールで駆動される gt<sub>wiz</sub>_userclk_tx_active_out へ接続されます。このポートは、クロッキング ヘルパー コアのステータスに基づいて駆動され、ヘルパー コアがリセット状態から回復したことを示します。クロック モジュールがコア用に user_clk および sync_clk を生成するため、このポートは High にアサートされる必要があります。このポートの機能は、7 シリーズ デバイスに生成されるものとは異なります。</p> <ul style="list-style-type: none"> <li>• 1 : クロッキング ヘルパーはアクティブ状態</li> <li>• 0 : クロッキング ヘルパーは非アクティブ状態であり、通常動作の準備が整っていない</li> </ul>
user_clk	入力	<p>Aurora 64B/66B コアとユーザー アプリケーションで共有されるパラレルクロックです。user_clk は、BUFG の出力であり、BUFG の入力は tx_out_clk から派生します。これらのクロック生成は、&lt;component name&gt;_clock_module ファイルで対応します。user_clk は txusrclk2 として トランシーバーへ入力されます。詳細は、該当する トランシーバーのユーザー ガイドまたはデータシートを参照してください。</p>
tx_out_clk	出力	<p>GTX/GTH トランシーバーは、PLL のスピード設定に基づいて、基準クロックから tx_out_clk を生成します。このクロックはバッファを介す必要があり、Aurora 64B/66B コアへ接続されるロジック用のユーザー クロックを生成するために使用されます。</p>
sync_clk	入力	<p>Aurora 64B/66B コアのシリアル トランシーバーの内部同期ロジックで使用されるパラレルクロックです。このクロックは、txusrclk として トランシーバー インターフェイスへ入力されます。sync_clk は、user_clk の 2 倍のレートです。詳細は、該当する トランシーバーのユーザー ガイドまたはデータシートを参照してください。</p>
gt_pll_lock	出力	<p>tx_out_clk が安定すると High にアサートされます (アクティブ High)。ディアサート (Low) 駆動されている場合は、tx_out_clk を使用する回路をリセット状態に保持する必要があります。</p>
gt_refclk	入力	<p>gt_refclk (clkp/clkn) ポートは、オシレーターで生成された専用外部クロックです。このクロックは、専用の IBUFDS を介して供給されます。</p>

## 機能の詳細説明

Aurora 64B/66B コアは、フレーミングまたはストリーミング ユーザー データ インターフェイスのいずれかで生成できます。その他、フレーミング インターフェイスを使用するデザインにはフロー制御オプションがあります。「[フロー制御](#)」を参照してください。

フレーミング ユーザー インターフェイスは、AXI4-Stream プロトコル仕様 (『AMBA AXI4-Stream プロトコル仕様』) に準拠しており、フレーム化されたユーザー データの送受信に必要な信号を構成します。ストリーミング インターフェイスでは、フレーム区切り文字を使用せずにデータを送信できます。動作が単純で、フレーミング インターフェイスより少ないリソースを使用します。

## 最上位アーキテクチャ

Aurora 64B/66B コアの最上位 (ブロック レベル) ファイルでは、Aurora レーン モジュール、TX/RX AXI4-Stream モジュール、グローバル ロジック モジュール、および GTX/GTH トランシーバー用ラッパーがインスタンス化されます。この最上位ラッパー ファイルは、クロック、リセット回路、およびフレームとチェッカー モジュールと共にサンプル デザイン ファイルにインスタンス化されています。

図 2-4 に、デュプレックス コンフィギュレーションでの Aurora 64B/66B コアの最上位アーキテクチャを示します。この最上位ファイルは、ユーザー デザインの基本となります。

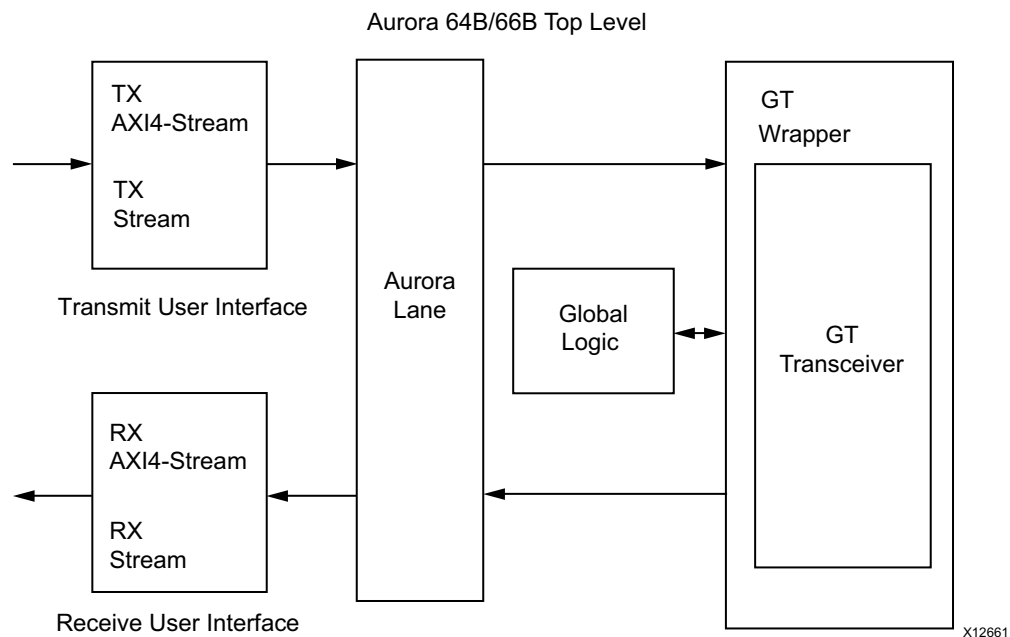


図 2-4: 最上位アーキテクチャ

このセクションでは、ストリーミング インターフェイスとフレーミング インターフェイスについて詳しく説明します。ユーザー インターフェイス ロジックは、次の各セクションで説明する各インターフェイスのタイミング要件を満たすように設計する必要があります。

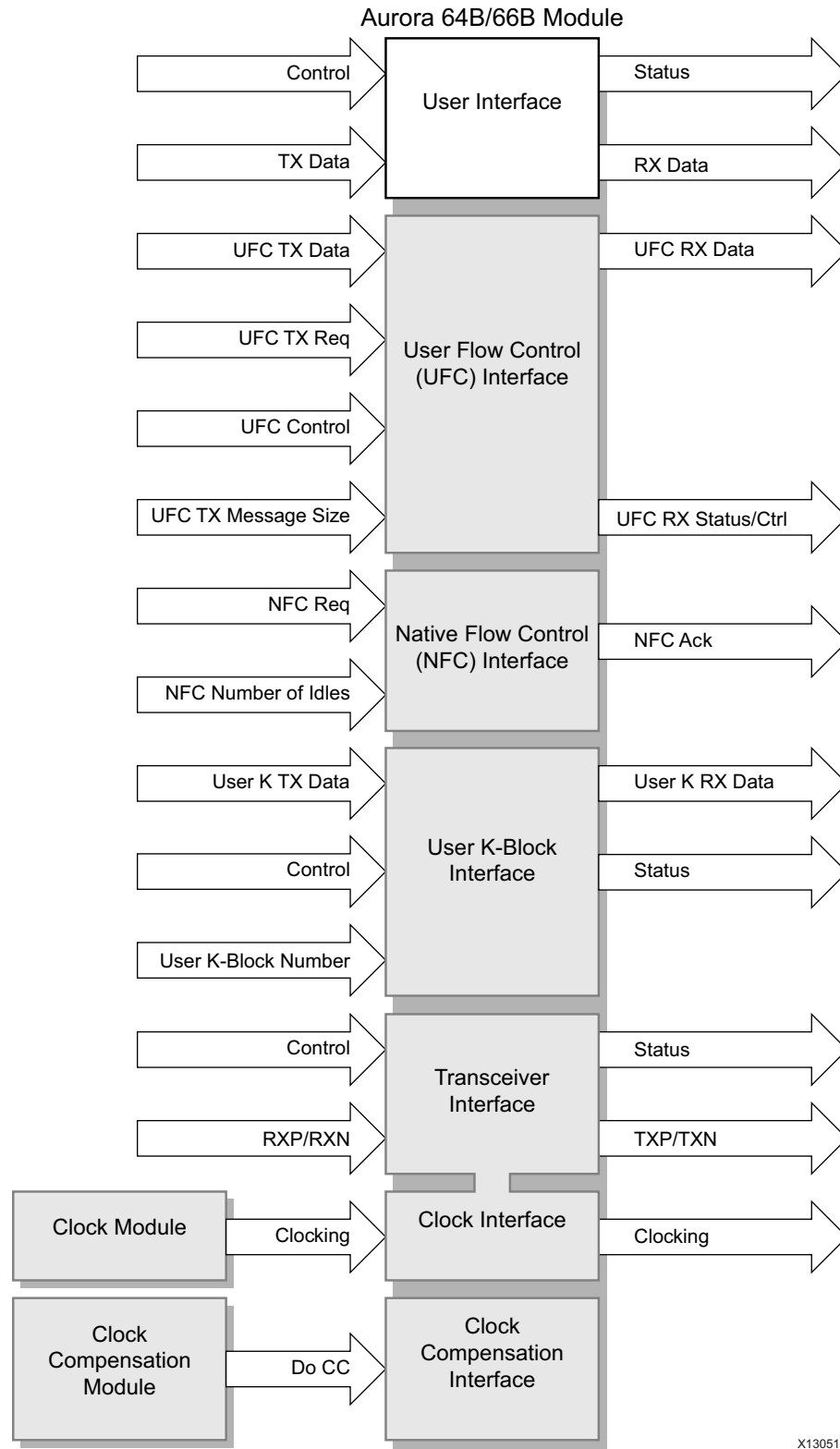
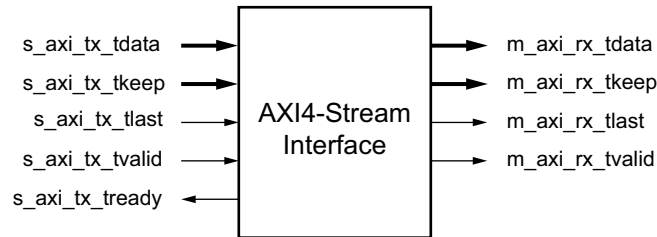


図 2-5 : 最上位ユーザー インターフェイス

注記 : ユーザー インターフェイス信号は、IP カタログで Aurora 64B/66B コアを生成する際の選択項目に基づいて異なります。

## フレーミング インターフェイス

図 2-6 に、Aurora 64B/66B コアのフレーミング ユーザー インターフェイスと TX/RX データ用の AXI4-Stream に準拠するポートを示します。コアには、Vivado IDE から AXI4-Stream User I/O をリトル エンディアン形式で構成するオプションがあります。デフォルトはビッグ エンディアン形式です。



X13013

図 2-6 : Aurora 64B/66B コアのフレーミング インターフェイス (AXI4-Stream)

**注記 :** ユーザー インターフェイスは、Aurora 64B/66B Vivado IDE の設定に基づいて、ビッグ エンディアンまたはリトル エンディアンのいずれかになります。

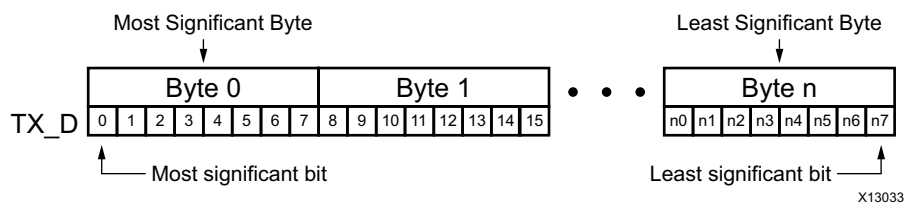
データを送信する場合、ユーザー アプリケーションは制御信号を操作してコアに次を実行させます。

- ユーザー アプリケーションの s\_axi\_tx\_tdata バスからデータを取得します。
- s\_axi\_tx\_tlast のアサートがフレームの最後であることを示し、Aurora チャンネルの複数レーンにデータをストライピングします。
- ユーザー アプリケーションは s\_axi\_tx\_tvalid 信号をデアサートすることによって、シリアル ラインにアイドルまたはポーズを挿入できます。

データを受信する場合、コアは次を実行します。

- 制御バイト (アイドル、クロック補正) を検出して破棄します。
- フレーミング信号 (m\_axi\_rx\_tlast) をアサートします。
- 複数レーンからのデータを回復します。
- ユーザー アプリケーションへデータを送信するために、m\_axi\_rx\_tdata バス上にあるデータを有効なバイト数 (m\_axi\_rx\_tkeep) とアセンブルして、m\_axi\_rx\_tlast サイクル中に m\_axi\_rx\_tvalid がアサートされます。

Aurora 64B/66B コアの AXI4-Stream ユーザー インターフェイスは、昇順を適用します。コアは、LSB 順の最上位ビットを最初に送受信します。図 2-7 に、Aurora 64B/66B コアの AXI4-Stream データ インターフェイスの n バイトの順序を示します。



X13033

図 2-7 : AXI4-Stream インターフェイスのビット順

### データ送信

AXI4-Stream は同期インターフェイスです。Aurora 64B/66B コアは、s\_axi\_tx\_tready と s\_axi\_tx\_tvalid の両方がアサート (High) されているサイクルで、user\_clk の立ち上がりエッジでのみインターフェイス上のデータをサンプルします。

AXI4-Stream 信号のサンプリングでは、`s_axi_tx_tvalid` および `s_axi_tx_tready` がアサートされている場合のみ有効としてみなされます。ユーザー アプリケーションは任意のクロック サイクルで `s_axi_tx_tvalid` 信号をデアサートできます。これによって、Aurora コアはそのサイクルの AXI4-Stream 入力を無視します。フレームの途中でこの信号がデアサートされると、アイドル シンボルが Aurora チャネル経由で送信され、結果として RX ユーザー インターフェイスで受信されるフレームの途中でアイドル サイクルに遷移します。

AXI4-Stream データは、フレームの中に含まれる場合のみ有効です。フレームの外にあるデータは無視されます。フレーム送信を終了する場合、データの最後のワード (または一部のワード) が `s_axi_tx_tdata` ポートに現れる間に `s_axi_tx_tlast` をアサートします。CRC が選択されている場合は、CRC が計算されて、データ ストリームの最後のデータワードの後に挿入されます。これが、有効な CRC バイト数に基づいて `s_axi_tx_tkeep` を再計算し、それに応じて `s_axi_tx_tlast` をアサートします。

## データ ストローブ

AXI4-Stream では、フレームの最後のワードは不完全なワードになることがあります。したがって、ワード サイズとは関係なく 1 つのフレームに任意のバイト数が含まれます。`s_axi_tx_tkeep` バスを使用して、フレームの最後のワードに含まれる有効なバイト数を示します。このバスは、`s_axi_tx_tlast` がアサートされる場合のみ使用されます。TKEEP は、`s_axi_tx_tdata` バス内の有効なバイト数を示し、フレームの最後のデータ ビートにおける特定バイトの有効性を表します。`s_axi_tx_tlast` がアサートされたときに最後のデータ ビートで TKEEP が「0F」の場合、8 バイトのうち 4 (LSB バイト) が有効で、byte4 ~ byte7 が無効となります。`s_axi_tx_tkeep` ですべてが「1」の場合は、`s_axi_tx_tdata` ポートのすべてのバイトが有効です。`s_axi_tx_tkeep` は、有効なバイトの位置を特定しませんが、`s_axi_tx_tlast` がアサートされた時点での最後のデータ ビートの有効なバイト数を示します。コアでは、TKEEP が LSB から左に揃えられることが要求されます。コアでサポートされているデータ ストリーム タイプの制約については、付録 B 「移行およびアップグレード」を参照してください。

## Aurora 64B/66B フレーム

TX サブモジュールは、TX インターフェイスから受信した各ユーザー フレームを Aurora 64B/66B フレームに変換します。コアは、最初のデータワードを含むデータブロックを送信して Aurora 64B/66B フレームを開始し、フレームの最後のバイトを含むセパレーターブロックを送信してフレームを終了します。有効なデータがない場合は常にアイドルブロックが挿入されます。ブロックは、8 バイトのスクランブル データ、または 2 ビットの制御ヘッダーを含む制御情報 (合計 66 ビット) です。Aurora 64B/66B のすべてのデータは、データブロックとして、またはセパレーターブロックとして送信されます (セパレーターブロックは、特定ブロックに含まれる有効なバイト数を示すためのカウント フィールドで構成される)。

表 2-17 に、偶数のバイト数を含む標準的な Aurora 64B/66B フレームを示します。

## 長さ

ユーザー アプリケーションでは、`s_axi_tx_tvalid` および `s_axi_tx_tlast` 信号を操作してチャネル フレームの長さを制御します。それに対して Aurora 64B/66B コアは、表 2-17 に示すようにこれらをデータブロック、アイドルブロック、およびセパレーターブロックに変換します。

表 2-17: 標準的なチャネル フレーム

Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	...	Data Byte n-2	Data Byte n-1	Data Byte n
SEP (1E)	Count (4)	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	x	x

## 例 A : シンプルなデータ送信

図 2-8 に、AXI4-Stream インターフェイス (n バイト幅) におけるシンプルなデータ送信の例を示します。この場合、送信されるデータ数は  $3n$  バイトとなるため、3 データ ビート が必要です。`s_axi_tx_tready` がアサートされると、AXI4-Stream インターフェイスがデータ送信可能な状態になったことを示します。Aurora 64B/66B コアは、データを送信しない間アイドルブロックを送信します。

データ送信を開始するには、ユーザー アプリケーションが `s_axi_tx_tvalid` とユーザー フレームの最初の `n` バイトをアサートします。`s_axi_tx_tready` 信号はすでにアサートされているため、データ送信は次のクロック エッジで開始します。データ バイトは、データ ブロックに配置され、Aurora チャンネルを介して送信されます。

データ送信を終了するには、ユーザー アプリケーションが `s_axi_tx_tlast`、`s_axi_tx_tvalid` と共に最後のデータ バイトと `s_axi_tx_tkeep` バス上の適切な値をアサートします。この例では、`s_axi_tx_tkeep` が `FF` に設定されてすべてのバイトが最後のデータ ビートで有効であることを示しています。Aurora 64B/66B コアは、データ ブロックの最後のワードを送信し、フレームの最後であることを示すために次のサイクルで空のセパレーター ブロックを送信する必要があります。次のサイクルで `s_axi_tx_tready` が再度アサートされるため、データ送信を続けることができます。新しいデータがなければ、Aurora 64B/66B コアはアイドルを送信します。

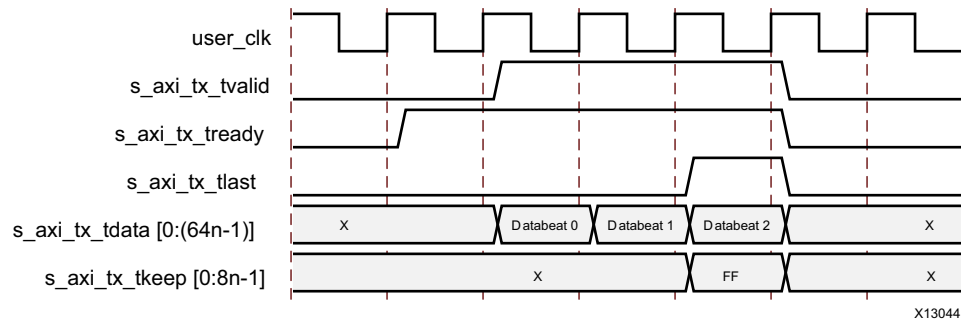


図 2-8 : シンプルなデータ送信

### 例 B : ポーズを含むデータ送信

図 2-9 では、フレーム転送中にユーザー アプリケーションがデータ送信を中断 (ポーズ) するプロセスを示しています。この例では、ユーザー アプリケーションが  $3n$  バイトのデータを送信し、最初の  $n$  バイトの後でデータフローを中断しています。最初のデータワードの後、ユーザー アプリケーションが `s_axi_tx_tvalid` をディアサートし、これによって TX Aurora 64B/66B コアはバス上のすべてのデータを無視して、その代わりにアイドルブロックを送信します。中断状態は、`s_axi_tx_tvalid` がディアサートされるまで続きます。

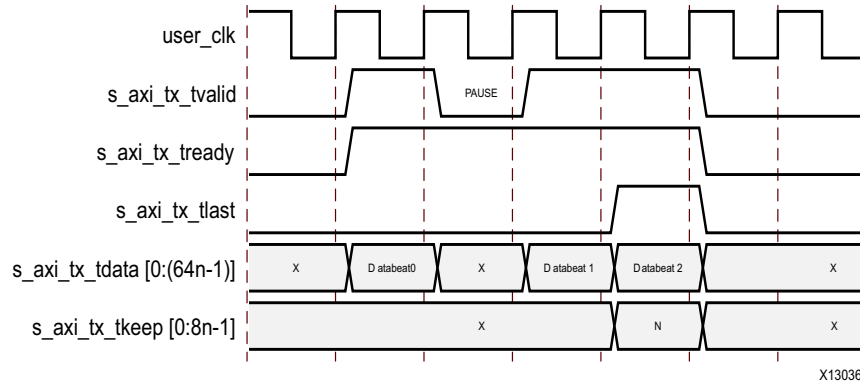
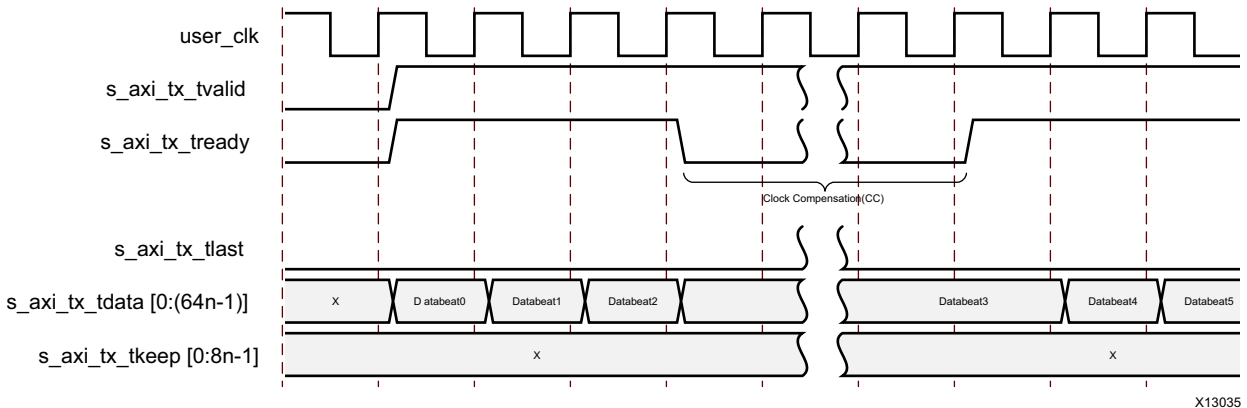


図 2-9 : ポーズを含むデータ送信

### 例 C : クロック補正を含むデータ送信

クロック補正シーケンスを送信する場合、Aurora 64B/66B コアは自動的にデータ送信を中断します。クロック補正シーケンスは、10,000 バイトごとに 3 サイクルのポーズを与えます。

図 2-10 では、クロック補正シーケンス中に Aurora 64B/66B コアがデータ送信を中断するプロセスを示しています。



X13035

**注記 :**

1. クロック補正機能を使用する場合、連続するデータ送信はできません。クロック補正機能が必要な場合の詳細情報は、64 ページの「クロック補正インターフェイス」を参照してください。

図 2-10 : クロック補正で中断されるデータ送信

### TX インターフェイスの例

このセクションでは、送信 FIFO と Aurora 64B/66B コアの AXI4-Stream インターフェイス間のシンプルなインターフェイスの例について説明します。

データ送信について復習すると、ユーザー アプリケーションが `s_axi_tx_tvalid` をアサートし、`s_axi_tx_tvalid` がアサートされたまま保持されていることを条件に、`s_axi_tx_tdata` バス上のデータが次のクロックの立ち上がりエッジで送信されることを示す `s_axi_tx_tready` がアサートされます。

図 2-11 には、Aurora 64B/66B コアとデータ ソース (この場合は FIFO) 間の一般的な接続を示しています。また、一般的な FIFO バッファ ステータスから `s_axi_tx_tvalid` 信号と `s_axi_tx_tlast` 信号を生成するために必要なシンプルなロジックも示しています。reset が FALSE の間、サンプル アプリケーションは FIFO に書き込みが行われるまで待機します。その後、`s_axi_tx_tvalid` 信号を生成します。Aurora 64B/66B コアは、これらの信号を受けて `s_axi_tx_tready` 信号をアサートし、FIFO の読み出しを開始します。

Aurora 64B/66B コアは、FIFO が空になるまで FIFO データをカプセル化して送信します。その後、サンプル アプリケーションは、`s_axi_tx_tlast` 信号を使用して Aurora 64B/66B コアに送信終了を伝えます。

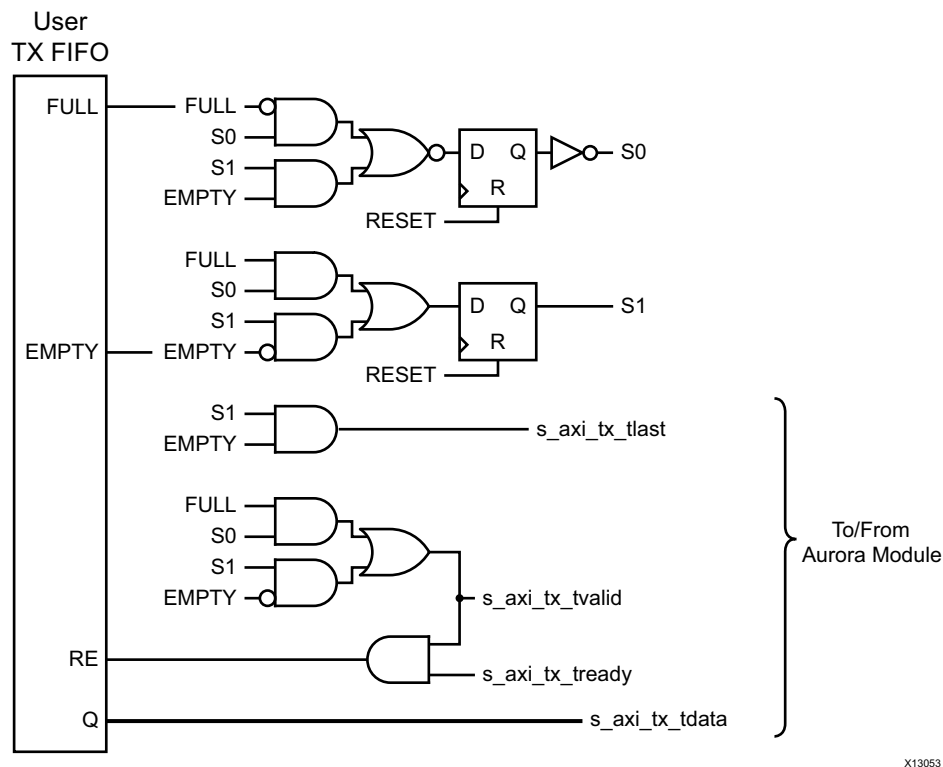


図 2-11 : データ送信

## データ受信

Aurora 64B/66B コアが Aurora 64B/66B フレームを受信する場合、制御情報、アイドルブロック、およびクロック補正ブロックを破棄した後に RX AXI4-Stream インターフェイスを介してユーザー アプリケーションにこれらを表示します。

Aurora 64B/66B コアには、ユーザー データ用のビルトインバッファがありません。その結果、RX AXI4-Stream インターフェイスには `m_axi_rx_tready` 信号がありません。ユーザー アプリケーションが Aurora チャネルからのデータフローを制御する唯一の方法は、オプションでコアのフロー制御機能を使用することです。ほとんどの場合は、RX データパスに FIFO を追加して、フロー制御メッセージが送信されている間にデータが失われないようにします。

Aurora 64B/66B コアは、RX AXI4-Stream インターフェイスの信号が有効の場合に `m_axi_rx_tvalid` 信号をアサートします。`m_axi_rx_tvalid` がディアサート (アクティブ Low) されている間に RX AXI4-Stream ポートでサンプルされた値は無視します。

`m_axi_rx_tvalid` 信号は、Aurora 8B/10B コアからの各フレームの最初のワードと同時にアサートされ、`m_axi_rx_tlast` 信号は、各フレームの最後のワードまたは部分的なワードと同時にアサートされます。`m_axi_rx_tkeep` ポートは、各フレームの最後のワードに含まれる有効なバイト数を示します。`s_axi_tx_tkeep` と同じバイト表示手順を使用し、`m_axi_rx_tkeep` がアサートされていない場合はすべてのバイトが有効 (すべて「1」) であることを示し、`m_axi_rx_tkeep` がアサート (アクティブ High) されている場合には有効なバイト数を正確に示します。

CRC オプションが選択されている場合は、想定される CRC 値に対して受信データ ストリームが計算されます。このブロックは、`m_axi_rx_tkeep` 値を再度計算し、それに応じて `m_axi_rx_tlast` をアサートします。

Aurora 64B/66B コアは、フレームの途中であっても常に `m_axi_rx_tvalid` をディアサートできます。

「例 A : ポーズを含むデータ受信」では、標準的な Aurora 64B/66B フレームの受信について説明しています。

### 例 A : ポーズを含むデータ受信

図 2-12 に、中断される  $3n$  バイトのデータ受信の例を示します。データは、`m_axi_rx_tdata` バス上に現れます。このバスに最初の  $n$  バイトが配置されると、`m_axi_rx_tvalid` がアサートされてユーザー アプリケーションにデータが有効であることを示します。最初のデータ ビートの後のクロック サイクルで、コアは `m_axi_rx_tvalid` をディアサートして、データ フローが中断されることをユーザー アプリケーションへ示します。

中断後、コアは `m_axi_rx_tvalid` をアサートして `m_axi_rx_tdata` バス上の残りのデータを引き続き集めて処理します。フレームの最後で `m_axi_rx_tlast` をアサートします。また、コアは `m_axi_rx_tkeep` の値も計算し、フレームの最後のワードに含まれる有効なバイト数を考慮して、それらをユーザー アプリケーションに提供します。

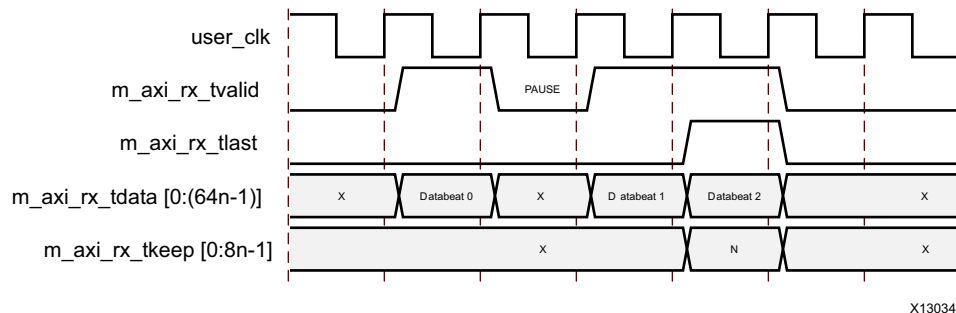


図 2-12 : ポーズを含むデータ受信

### RX インターフェイスの例

Aurora 64B/66B コアの RX AXI4-Stream インターフェイスは、シンプルな FIFO を使用して実装できます。データ受信では、FIFO が `m_axi_rx_tvalid` 信号をモニターします。有効なデータが `m_axi_rx_tdata` に現れると、`m_axi_rx_tvalid` がアサートされます。`m_axi_rx_tvalid` は FIFO WE ポートに接続されているため、データとフレーミング信号が FIFO に書き込まれます。

### フレーミングの効率性

Aurora 64B/66B コアのフレーミングの効率性に影響を与える要素は次の 2 つです。

- フレーム サイズ
- `user_clk` の 32 サイクルごとに生じるギアボックスからのデータ無効化要求

`user_clk` の 10,000 サイクルごとにすべてのレーンで 3 `user_clk` サイクルを使用するクロック補正 (CC) シーケンスは、総チャネル帯域幅の約 0.03% を使用します。

GTX/GTH トランシーバーのギアボックスは、クロック分周値や 64B/66B エンコードのために定期的に中断する必要があります。これは、AXI4-Stream インターフェイスでバック プレッシャーとして現れ、ユーザー データは 32 サイクルごとに 1 サイクル間停止します (図 2-13)。ユーザー インターフェイスには、32 サイクルごとに 1 サイクル間ディ

アサート (アクティブ Low) される Aurora コアからの `s_axi_tx_tready` 信号があります。ポーズ サイクルを使用して、64B/66B エンコード用のギアボックスを補正します。



X13039

図 2-13 : フレーミングの効率性

GTX/GTH トランシーバーにおけるギアボックスのポーズについては、『7 シリーズ FPGA GTX/GTH トランシーバー ユーザー ガイド』(UG476) [参照 4] または『UltraScale アーキテクチャ GTH トランシーバー ユーザー ガイド (UG576) [参照 3] を参照してください。

Aurora 64B/66B コアは、Aurora 64B/66B プロトコルの [Strict Aligned] オプションを実行します。特定サイクル上で、アイドルブロックまたは SEP ブロックの後にデータブロックが挿入されることはありません。SEP ブロックの後はデータブロックを挿入しないという制限があるため、マルチレーンの Aurora 64B/66B コアではフレーミングの効率性が低下します。

表 2-18 には、クロック補正用のオーバーヘッドを考慮した後の計算例を示しています。この例は、シングル レン チャンネルの効率性を示しており、チャンネル フレームの長さが増加すると効率性が増すことを確認できます。

表 2-18 : 効率性の例

ユーザー データ バイト	フレーミングの効率性 %
100	96.12
1,000	99.18
10,000	99.89

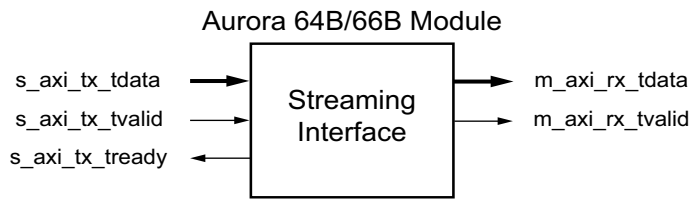
表 2-19 は、256 バイトのフレーム データを送信する際のシングル レン チャンネルでのオーバーヘッドを示しています。フレームの終わりを示す SEP ブロックが使用されるため、最終的なデータ ユニットは 264 バイトとなります。この値は、トランスミッターのオーバーヘッドの 3.03% に相当します。さらに、クロック補正ブロックが 10,000 サイクルごとに 3 サイクル間送信される必要があるため、0.03% のオーバーヘッドが追加されます。

表 2-19 : 256 データ バイトを送信する場合の標準的なオーバーヘッド

レーン	クロック	機能
[D0:D7]	1	チャンネル フレーム データ
[D8:D15]	2	チャンネル フレーム データ
.	.	.
[D248:D255]	32	チャンネル フレーム データ
制御ブロック	33	SEP0 ブロック

## ストリーミング インターフェイス

図 2-14 に、ストリーミング ユーザー インターフェイスで構成された Aurora 64B/66B コアの例を示します。



X13023

図 2-14 : Aurora 64B/66B コアのストリーミング ユーザー インターフェイス

注記 : `s_axi_tx_tdata` および `m_axi_rx_tdata` の幅は、Vivado IDE の設定 ([Little Endian Support] または [Big Endian Support]) に依存します。

## データの送受信

ストリーミング インターフェイスでは、Aurora チャンネルをパイプとして使用できます。チャンネルの TX 側に書き込まれたワードは、RX 側へ順番に送られます (レイテンシが生じる)。初期化後、チャンネルは常に書き込み可能な状態ですが、`do_cc` 信号がアサートされてクロック補正シーケンスが送信される場合は例外です。アプリケーションは、`s_axi_tx_tdata` ポートを介してデータを送信し、`s_axi_tx_tvalid` ポートを使用してデータが有効であることを示します (アクティブ High にアサート)。ストリーミング Aurora インターフェイスでは、`s_axi_tx_tdata` ポートのすべての幅に対してデータが埋め込められていることが求められます。Aurora 64B/66B コアは、チャンネルがデータを受信できる状態でない場合に `s_axi_tx_tready` をディアサート (アクティブ Low) します。それ以外の場合、`s_axi_tx_tready` はアサートされたままとなります。

`s_axi_tx_tvalid` がディアサートされると、ワード間にギャップが生じます。これらのギャップは、クロック補正シーケンスが送信される場合以外はそのまま残されます。Aurora チャンネルの両側における周波数差を補正するために、トランシーバーによってクロック補正シーケンスが CC ロジックによって複製または削除されます。これにより、`DO_CC` のアサートによってできたギャップが縮小/拡大します。`do_cc` 信号の詳細は、64 ページの「クロック補正インターフェイス」を参照してください。

Aurora チャンネルの RX 側にデータが到達すると、`m_axi_rx_tdata` バス上に現れて `m_axi_rx_tvalid` がアサートされます。このデータはすぐに読み出されなければ失われます。これが不可能な場合は、RX インターフェイスにバッファを接続して、読み出し可能になるまでデータを保持する必要があります。

図 2-15 に、ストリーミング データの標準的な例を示します。この例は、いずれの `ready` 信号もアサートされていない状態、つまりユーザー ロジックと Aurora 64B/66B コアが共にデータ転送の準備が整っていない状態で開始されています。次のクロック サイクルで、Aurora 64B/66B コアは `s_axi_tx_tready` をアサートし、データを転送できる状態を示しています。その 1 サイクル後、ユーザー ロジックは `s_axi_tx_tvalid` 信号をアサートして `s_axi_tx_tdata` バス上にデータを配置し、データを転送できる状態を示しています。これで両方の `ready` 信号がアサートされて、データ D0 がユーザー ロジックから Aurora 8B/10B コアへ転送されます。次のクロック サイクルでデータ D1 が転送されます。

この例では、Aurora 64B/66B コアが `ready` 信号の `s_axi_tx_tready` をディアサートし、`s_axi_tx_tready` 信号が再びアサートされる次のクロック サイクルまでデータは転送されません。そして、次のクロック サイクルでユーザー アプリケーションが `s_axi_tx_tvalid` をディアサートし、両方の `ready` 信号がアサートされるまでデータは転送されません。

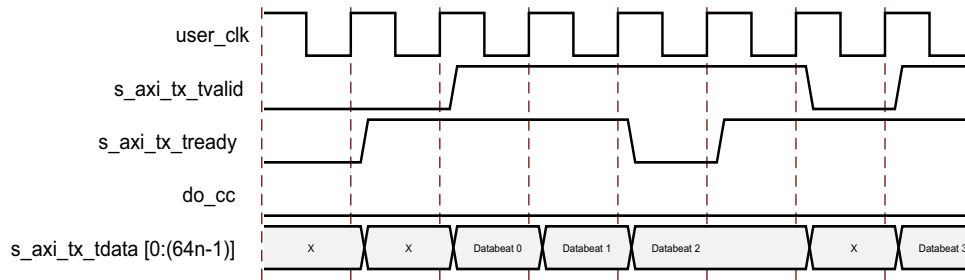


図 2-15 : 標準的なストリーミング データ送信

図 2-16 に、ストリーミング データ受信の標準的な例を示します。

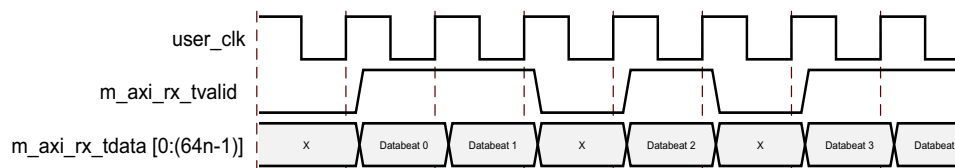


図 2-16 : 標準的なストリーミング データ受信

## フロー制御

このセクションでは、Aurora コアのフロー制御方法について説明します。次に示す 2 つのフロー制御インターフェイスから選択できます。ネイティブ フロー制御 (NFC) は、受信側のフルデュプレックス チャネルでデータ転送レートを制御する場合に使用されます。ユーザー フロー制御 (UFC) は、動作を制御する際に優先順位の高いメッセージに対応するために使用されます。

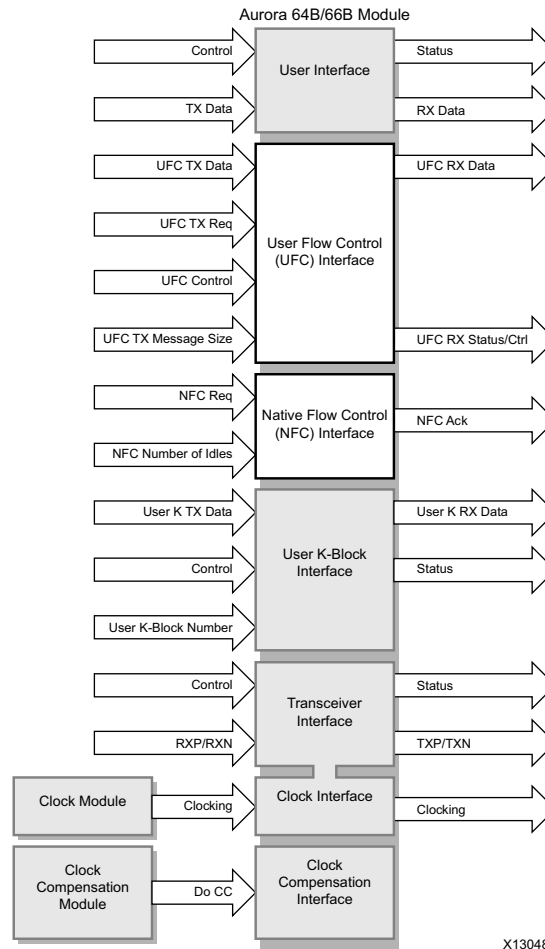


図 2-17 : 最上位のフロー制御

### ネイティブ フロー制御

Aurora 64B/66B プロトコルに含まれるネイティブ フロー制御 (NFC) では、チャネル パートナーがデータを送信できないサイクル数を指定することによって、レシーバー側でデータ送信されるレートを制御できます。トランスミッターに一時的にアイドルのみを送信するように要求することで、データ フローを完全に無効にすることもできます (XOFF)。通常、NFC は FIFO のオーバーフローを防ぐために使用されます。NFC 動作の詳細は、『Aurora 64B/66B プロトコル仕様 v1.2』[参照 5]を参照してください。

図 2-18 および図 2-19 には、NFC メッセージのデフォルト (ビッグ エンディアン) 形式とリトル エンディアン形式を示します。

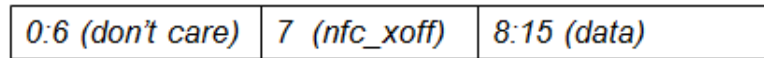


図 2-18 : NFC メッセージの形式 (デフォルト)

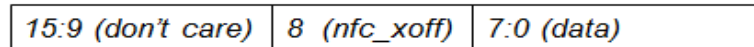


図 2-19 : NFC メッセージの形式 (リトル エンディアン形式)

### デフォルト モードの NFC メッセージ

NFC メッセージをチャネル パートナーへ送信するには、ユーザー アプリケーションが `s_axi_nfc_tx_tvalid` をアサートし、`s_axi_nfc_tx_tdata[8:15]` に 8 ビットの PAUSE カウントを書き込みます。ポーズ コードは、NFC メッセージを受信してからデータ送信を再開できるようになるまでチャネル パートナーが待機する必要がある最少サイクル数を示します。データ送信を含まない `user_clk` サイクル数は、`s_axi_nfc_tx_tdata` に 1 を加えた数に相当します。

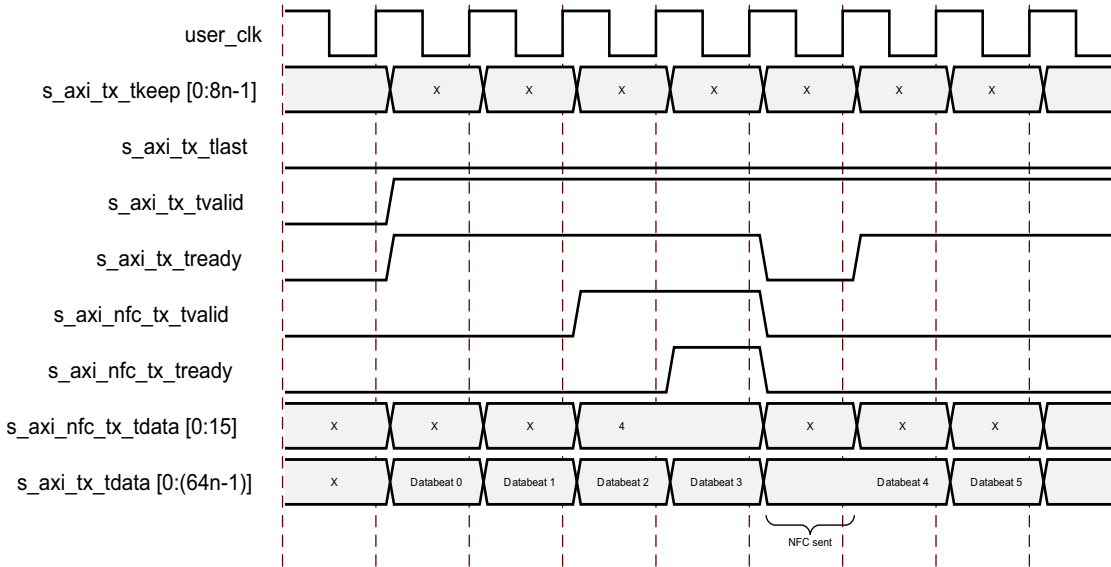
信号 `s_axi_nfc_tx_tdata[7]` は、NFC\_XOFF を表します。アサートして NFC\_XOFF メッセージを送信すると、XOFF NFC 以外のメッセージを受信またはリセットが実行されるまで、チャネル パートナーはデータ送信を中断することが要求されます。PAUSE と XOFF が共に 0 に設定されて要求が送信される場合、NFC は XON モードに設定されます。XOFF モードをオフにするには、XON メッセージ (すべて 0) が送信される必要があります。つまり、この XON 要求を受信した後、コアは任意の新しい NFC 要求を受信します。`s_axi_nfc_tx_tready` が `user_clk` の立ち上がりエッジでアサートされ、Aurora 64B/66B コアが NFC メッセージを送信することを示すまで、ユーザー アプリケーションは `s_axi_nfc_tx_tvalid`、`s_axi_nfc_tx_tdata[8:15]`、および `s_axi_nfc_tx_tdata[7]` (`nfc_xoff`) (使用している場合のみ) をホールドします。

Aurora 64B/66B コアは、NFC メッセージを送信している間、データを送信できません。`s_axi_tx_tready` は、`s_axi_nfc_tx_tready` のアサート後のサイクルで常にディアサートされます。NFC Completion モードは、フレーミング Aurora 64B/66B インターフェイスでのみ使用できます。

### 例 A : NFC メッセージの送信

図 2-20 は、ユーザー アプリケーションが AXI4-Stream インターフェイスを使用してチャネル パートナーへ NFC メッセージを送信する際のタイミングの例をしています。

注記 : `s_axi_tx_tready` 信号が 1 サイクル間ディアサートされ、データ フローにギャップが生じています。このとき、NFC メッセージが送信されます。

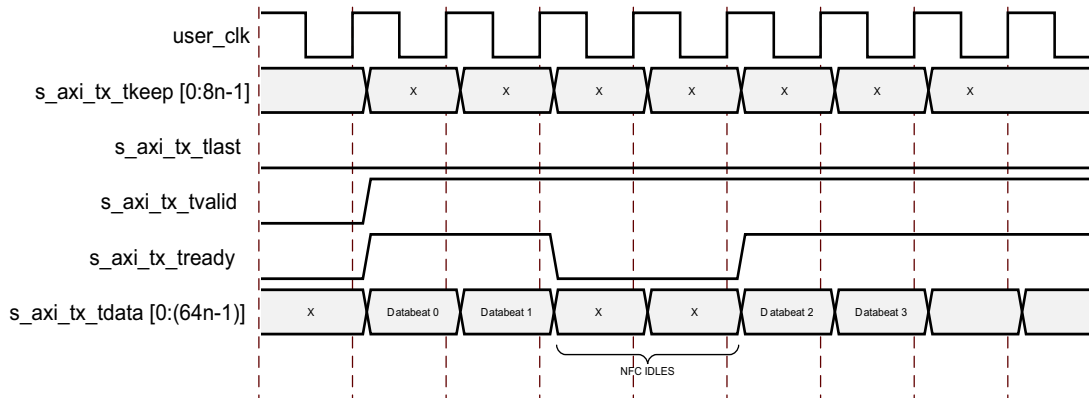


X13055

図 2-20 : NFC メッセージの送信

### 例 B : NFC アイドルが挿入されたメッセージの受信

図 2-21 は、NFC メッセージが受信される場合の TX ユーザー インターフェイスでの信号の例を示しています。この場合、NFC メッセージは  $8'b01$  を伝搬し、データ送信を除いて 2 サイクルが必要です。コアは、ユーザー インターフェイスで `s_axi_tx_tready` をディアサートして、2 サイクル間データが送信されないようにします。この例では、コアは Immediate NFC モードで動作しています。Aurora 64B/66B コアは、Completion モードでも動作できます。この場合、NFC アイドルは、新しいフレームの最初のデータ バイトの前にも挿入されます。Completion モードでフレーム送信中にコアが NFC メッセージを受信する場合は、フレームの送信を終了した後に `s_axi_tx_tready` をディアサートしてアイドルを挿入します。



X13054

図 2-21 : NFC アイドルが挿入されたメッセージの送信

### ユーザー フロー制御

Aurora 64B/66B プロトコルには、チャンネル パートナーが独立したインバンド チャンネルを使用して制御情報を送信できるようにするユーザー フロー制御があります。動作中のフレームの最後が現れるまで待機しなくても、ユーザー アプリケーションはチャンネル パートナーへ短い UFC メッセージを送信できます。UFC メッセージは、標準のフレーム データとチャンネルを共有しますが、フレーム データより高い優先順位で処理されます。UFC メッセージは、CC/NR/CB/NFC ブロックなどの優先順位の高い制御ブロックによって中断されます。

## UFC メッセージの送信

UFC メッセージは、1 ~ 256 データ バイトを伝搬できます。ユーザー アプリケーションは、`s_axi_ufc_tx_ms` ポートに必要なバイト数から 1 を引いた数を駆動してメッセージの長さを指定します。たとえば、この値が 3 の場合、実際には 4 バイトのデータが送信されます。値が 0 の場合、1 バイトが送信されます。

UFC メッセージを送信するため、任意の SIZE コードで `ufc_tx_ms` ポートを 1 サイクル間駆動している間、ユーザー アプリケーションが `ufc_tx_req` をアサートします。ある要求が実行されてから、その要求の最後のサイクルで `s_axi_ufc_tx_tready` がアサートされるまで、次の要求は送信できません。UFC メッセージ用データは `s_axi_ufc_tx_tdata` ポートに配置される必要があり、バスに有効なメッセージデータが含まれている場合は常に `s_axi_ufc_tx_tvalid` 信号がアサートされます。

コアは、UFC データを送信する間 `s_axi_tx_tready` をディアサートし、要求されているメッセージが完成するのに十分なデータが含まれるまで `s_axi_ufc_tx_tready` のアサートを保持します。UFC メッセージの間に `s_axi_ufc_tx_tvalid` がディアサートされると、チャンネルにアイドルが送信され、`s_axi_tx_tready` はディアサートを保持して `s_axi_ufc_tx_tready` はアサートを保持します。CC 要求、CB 要求、または NFC 要求がコアに送信されると、これらは優先順位が高いため、要求された動作が実行される間 `s_axi_ufc_tx_tready` がディアサートされます。

### 例 A : シングル サイクル UFC メッセージの送信

図 2-22 に、シングル サイクル UFC メッセージの送信プロセスを示します。この場合、8 バイトのインターフェイスに 4 バイトのメッセージが送信されています。

注記 : コアがメッセージを受信する前に `s_axi_tx_tready` and `s_axi_ufc_tx_tready` 信号が 1 サイクル間ディアサートされます。このサイクルで UFC ヘッダーが送信されます。

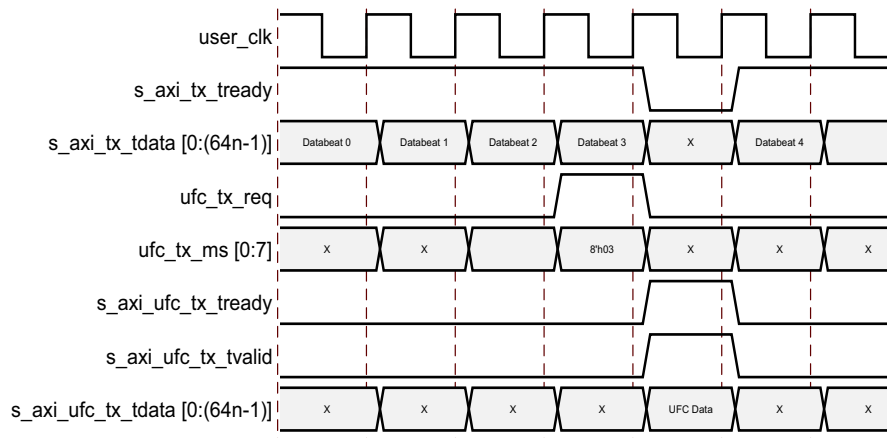


図 2-22 : シングル サイクル UFC メッセージの送信

### 例 B : マルチサイクルの UFC メッセージの送信

図 2-23 に、2 サイクルの UFC メッセージ送信プロセスを示します。この場合、ユーザー アプリケーションは 8 バイトのインターフェイスを使用して 16 バイトのメッセージを送信しています。

UFC データを送信するためには、`s_axi_ufc_tx_tready` 信号が 2 サイクル間アサートされます。

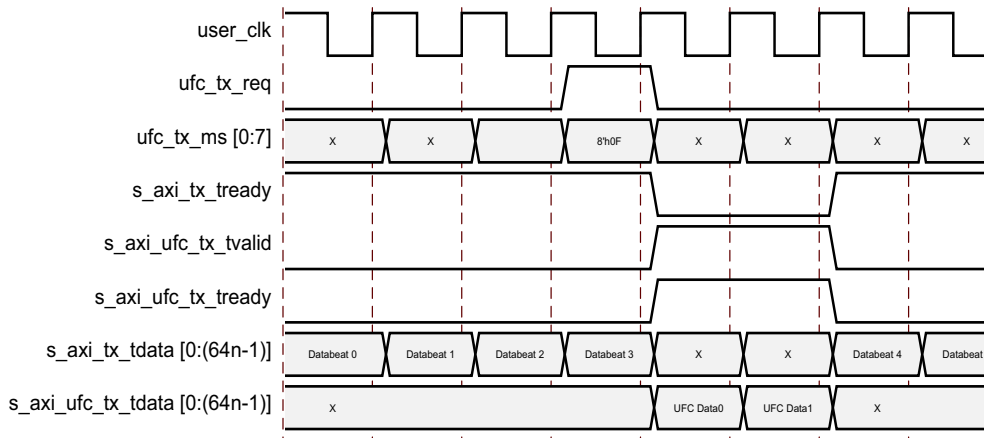


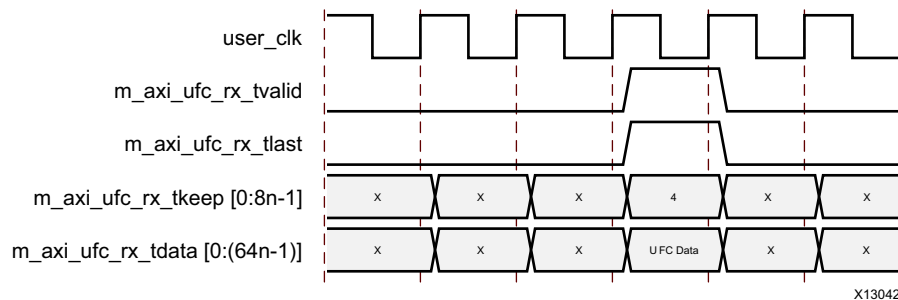
図 2-23 : マルチサイクルの UFC メッセージの送信

### ユーザー フロー制御メッセージの受信

Aurora 64B/66B コアが UFC メッセージを受信する場合、専用の UFC AXI4-Stream インターフェイス経由でメッセージ データをユーザー アプリケーションへ渡します。データは `m_axi_ufc_rx_tdata` ポートに現れます。`m_axi_ufc_rx_tvalid` のアサートがメッセージ データの開始を示し、`m_axi_ufc_rx_tlast` が終わりを示します。`m_axi_ufc_rx_tkeep` を使用して、メッセージの最後のサイクル (例:`m_axi_ufc_rx_tlast` がアサートされている間) に `m_axi_ufc_rx_tdata` 上で有効となるバイト数を表します。`ufc_rx` AXI4-Stream インターフェイス上の信号は、`m_axi_ufc_rx_tvalid` がアサートされている場合のみ有効です。

### 例 C : シングル サイクルの UFC メッセージの受信

図 2-24 は、4 バイトの UFC メッセージを受信する 8 バイト データ インターフェイスの Aurora 64B/66B コアを示しています。コアは、`m_axi_ufc_rx_tvalid` と `m_axi_ufc_rx_tlast` をアサートしてシングル サイクル フレームであることを示し、ユーザー アプリケーションにこのデータを送信しています。`m_axi_ufc_rx_tkeep` バスは 4 に設定され、インターフェイスの最高位バイト 4 つのみが有効であることを示しています。



X13042

図 2-24 : シングルサイクルの UFC メッセージの受信

### 例 D : マルチサイクルの UFC メッセージの受信

図 2-25 は、15 バイトの UFC メッセージを受信する 8 バイト データ インターフェイスの Aurora 64B/66B コアを示しています。

**注記** : 最終的なフレームの長さは 2 サイクル分となり、2 つ目のサイクルで `m_axi_ufc_rx_tkeep` が 7 に設定され、データ インターフェイスの 7 バイトすべてのデータが有効であることを示しています。

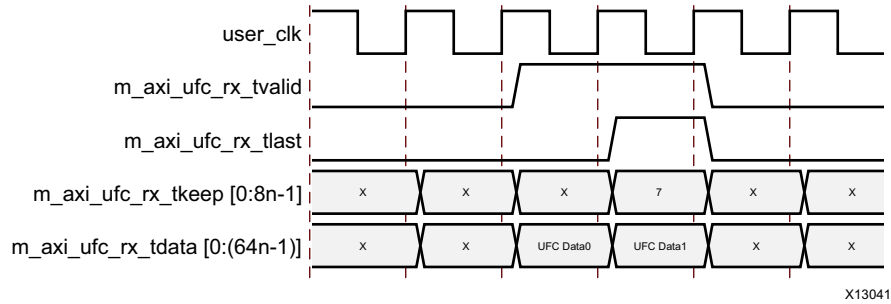


図 2-25 : マルチサイクルの UFC メッセージの受信

## ユーザー K ブロック インターフェイス

このセクションでは、短いブロック データの送信および受信について説明します。

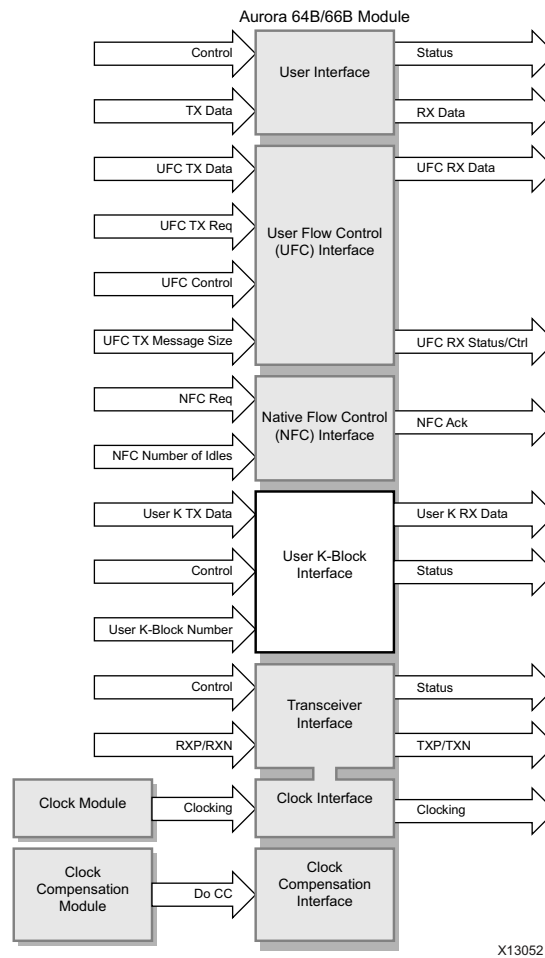


図 2-26 : 最上位ユーザー K ブロック インターフェイス

ユーザー K ブロックは、制御ブロックを含む特殊なシングルブロック コードで、Aurora インターフェイスではデコードされず、ユーザーに直接渡されます。これらのブロックは、アプリケーション固有の制御ファンクションをインプリメントするために使用できます。ユーザー K ブロックは、合計 9 個あります (表 2-20)。これらの優先順位は、UFC より低く、ユーザー データより高くなります。

表 2-20: ユーザー K ブロックの有効な BTF (ブロック タイプ フィールド)

ユーザー K ブロック名	ユーザー K ブロックの BTF
ユーザー K ブロック 0	0xD2
ユーザー K ブロック 1	0x99
ユーザー K ブロック 2	0x55
ユーザー K ブロック 3	0xB4
ユーザー K ブロック 4	0xCC
ユーザー K ブロック 5	0x66
ユーザー K ブロック 6	0x33
ユーザー K ブロック 7	0x4B
ユーザー K ブロック 8	0x87

ストリーミング デザインとフレーミング デザインで、ユーザー K ブロックは区別されません。各ユーザー K ブロックのコードは 8 バイト幅で、ユーザー K BTF を使用してエンコードされます。これは、ユーザー アプリケーションの `s_axi_user_k_tx_tdata` にユーザー K ブロック番号として示されます。ユーザー K ブロックはシングルブロック コードで、常にユーザー K ブロック番号で表されます。ユーザーは、18 ページの表 2-9 に示すとおり、ユーザー K ブロック番号を与える必要があります。7 バイトの `s_axi_user_k_tdata` のみ使用できます。

図 2-27 および図 2-28 には、ユーザー K のデフォルト (ビッグ エンディアン) 形式とリトル エンディアン形式を示します。



図 2-27: デフォルトのユーザー K 形式



図 2-28: リトル エンディアンのユーザー K 形式

## ユーザー K ブロックの送信

s\_axi\_user\_k\_tx\_tready 信号は Aurora コアによってアサートされますが、CC、CB、NFC、および UFC 要求が実行されるとデアサートされます。ユーザー K ブロック番号と共に s\_axi\_user\_k\_tx\_tdata を配置し、s\_axi\_user\_k\_tx\_tvalid がアサートされた後、ユーザー アプリケーションは必要に応じて s\_axi\_user\_k\_tx\_tready のアサート後に s\_axi\_user\_k\_tx\_tdata を変更できます (図 2-29)。これによって、Aurora コアは 9 個のユーザー K ブロックの中から適切なユーザー K BTF を選択できます。s\_axi\_user\_k\_tx\_tready がアサートされている間に現れるデータは常に処理されます。

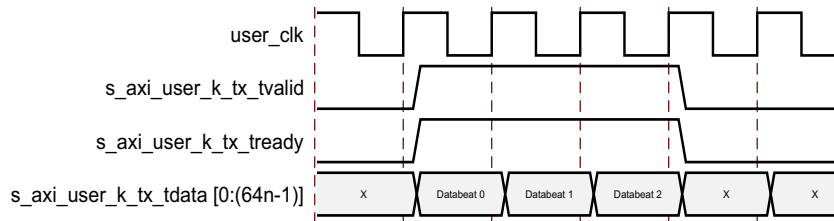


図 2-29 : ユーザー K データおよびユーザー ブロック番号の送信

## ユーザー K ブロックの受信

受信 BTF はデコードされて、対応する BTF のブロック番号がユーザー アプリケーションへ渡されます (図 2-30)。ユーザー アプリケーションは、m\_axi\_rx\_user\_k\_tvalid がアサートされているときにバス上に現れる m\_axi\_rx\_user\_k\_tdata を有効な値として認識します。

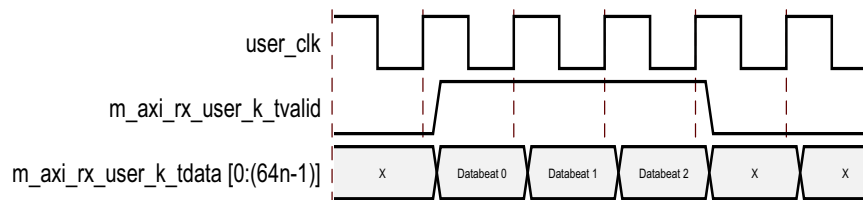


図 2-30 : ユーザー K データおよびユーザー ブロック番号の受信

## ステータス、制御、およびトランシーバー インターフェイス

Aurora 64B/66B コアのステータスおよび制御ポートによって、ユーザー アプリケーションは Aurora チャンネルをモニターでき、またシリアル トランシーバー インターフェイスのビルトイン機能を使用できるようになります。このセクションでは、Aurora 64B/66B コアのステータスおよび制御インターフェイスの図を示し、それらのポートについて説明します。また、GTX および GTH シリアル I/O インターフェイスについても説明します。

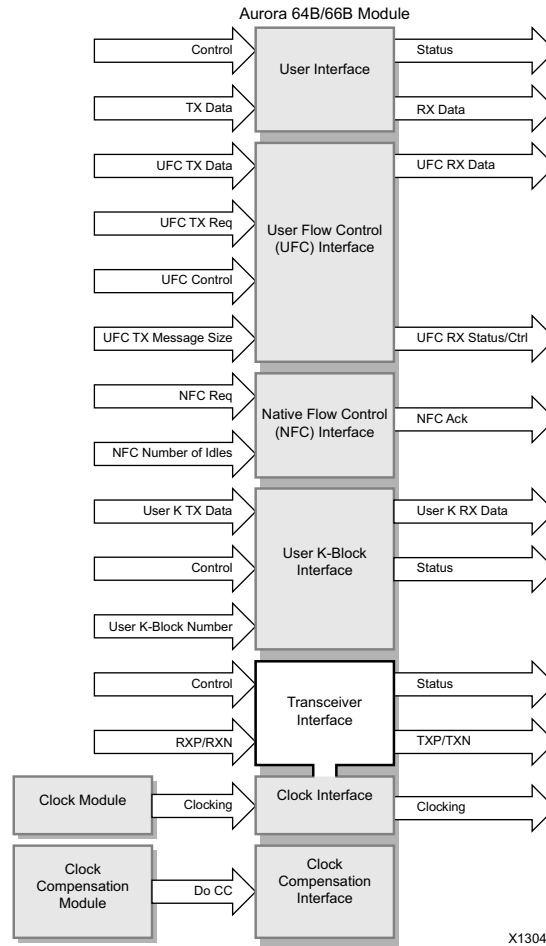


図 2-31 : 最上位 GTX インターフェイス

## ステータスおよび制御ポート

Aurora 64B/66B コアは、フルデュプレックス/シンプレックスとなり、送信および受信の Aurora 8B/10B チャネル接続を提供します。シンプレックス モード動作の場合、Aurora 64B/66B コアは、いかなるサイドバンド信号も必要ありません。図 2-32 に、Aurora 64B/66B コアのステータスおよび制御インターフェイスを示しています。

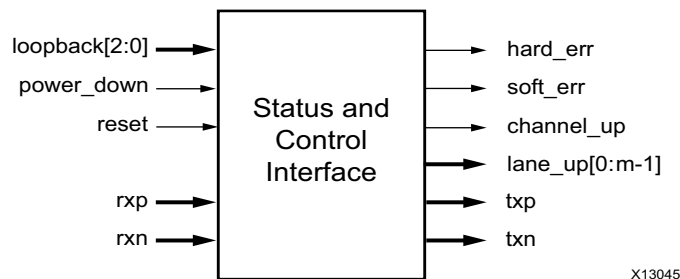


図 2-32 : Aurora 64B/66B コアのステータスおよび制御インターフェイス

## Aurora 64B/66B コアのエラー信号

装置の問題やチャネル ノイズが原因となり、Aurora チャネル動作中にエラーが生じる場合があります。64B/66B エンコードによって、Aurora 64B/66B コアはチャネル内で生じたビット エラーを検出できます。コアは、検出したすべてのサイクルで `soft_err` 信号をアサートして、これらのエラーをレポートします。

また、各高速シリアル GTX/GTH トランシーバーをモニターして、バッファのオーバーフローやロックの損失などのハードウェア エラーも検出します。この場合、`hard_err` 信号をアサートしてハードウェア エラーをレポートします。多数のソフト エラーが生じた場合も致命的なハードウェア エラーとなります。コアは、『Aurora 64B/66B プロトコル仕様 v1.2』(SP011) [参照 5] で説明している Block Sync アルゴリズムを使用して、多数のソフト エラーをハード エラーとして対応するべきかを判断します。

ハード エラーが検出されると常に、Aurora 64B/66B コアが自動的にリセットをトリガーして再初期化を行います。通常、このプロセスによってハード エラーの原因が修正されるとすぐに Aurora チャネルが再構築されます。ソフト エラーの場合は、短時間に多数のエラーが生じて、ブロック同期ステート マシンがトリガーされない限り、リセットは実行されません。

表 2-21: フルデュプレックス コアのエラー信号

信号	説明
<code>hard_err/</code> <code>tx_hard_err/</code> <code>rx_hard_err</code>	<p>TX のオーバーフロー/アンダーフロー: TX データ用のエラスティック バッファのオーバーフロー/アンダーフローを示しています。これは、ユーザー クロックと基準クロックのソースが同じ周波数で動作していない場合に生じます。</p> <p>RX のオーバーフロー/アンダーフロー: RX データ用のクロック補正およびチャネル ボンディング FIFO のオーバーフロー/アンダーフローを示しています。これは、2 つのチャネルパートナーのクロック ソース周波数が <math>\pm 100\text{ppm}</math> 範囲外の場合に生じます。</p>
<code>soft_err/</code> <code>tx_soft_err/</code> <code>rx_soft_err</code>	<p>ソフト エラー: 短い期間に多数のソフト エラーがあることを示しています。無効な同期ヘッダーが多数検出された場合には、アライメント用のブロック同期ステート マシンが自動的に再アラインを実行します。ソフト エラーがハード エラーに変わることはありません。</p> <p>無効な SYNC ヘッダー: 64 ビット ブロックで 2 ビットのヘッダーが無効の制御またはデータヘッダーであることを示しています。</p> <p>無効な BTF: 制御ブロックの受信で、ブロック タイプ フィールド (BTF) 内に認識されない値が含まれていることを示します。通常、これはビット エラーが原因となります。</p>

## 初期化

Aurora 64B/66B コアは、電源投入後、リセット後、またはハード エラー発生後に自動的に初期化を実行します。チャンネルの両側の Aurora 64B/66B コアが、チャンネルの使用準備が整うまで Aurora の初期化プロセスを実行します。lane\_up バスは、チャンネル内のどのレーンが初期化プロセスのレーン初期化プロセスを完了したかを示します。この信号は、マルチレーン チャンネルで装置問題をデバッグする際に役立ちます。channel\_up は、コアがすべての初期化プロセスを完了した場合のみアサートされます。

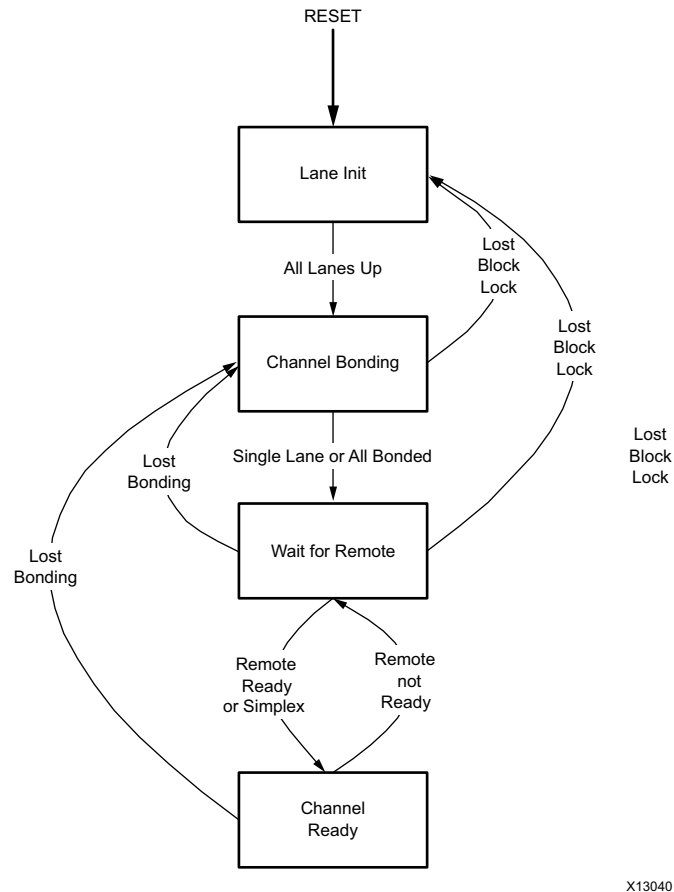


図 2-33 : 初期化の概要

Aurora 64B/66B コアは、channel\_up がアサートされる前にデータを受信できます。入力されるデータの適性判断には、ユーザー インターフェイスの m\_axi\_rx\_tvalid 信号のみ使用されます。channel\_up がアサートされるまでデータ転送は行われないため、channel\_up 信号を反転させて使用し、フルデュプレックス チャンネルの TX 側を駆動するモジュールをリセットできます。データを受信する前にユーザー アプリケーション モジュールをリセットする必要がある場合は、いずれかの lane\_up 信号を反転して使用できます。データは、すべての lane\_up 信号がアサートされるまで受信されません。

## Aurora のシンプレックス動作

シンプレックス Aurora 64B/66B コアにはサイドバンド接続がなく、チャンネル パートナーが初期化を完了してデータ送信可能な状態になったことを知らせるためにタイマーを使用します。

シンプレックス TX/RX コアは、GT の送信部と受信部を両方備えており、それぞれ独立して動作するよう構成されていますが、コアの送信パスと受信パス間に共通のリセット信号と `pma_init` があります。

ユーザー アプリケーションは、チャンネル要件に基づいてタイマー値を変更できます。シンプレックス リンクの場合、`tx_channel_up` がアサートされる前に `rx_channel_up` がアサートされる必要があります。これによって、シンプレックス TX が動作する前にシンプレックス RX の受信可能な状態になります。

TX のレーン アップ信号は、シンプレックス RX リンクのブロック ロック時間および CDR ロック時間を示す 24 ビット カウンターに基づいてアサートされます。TX/RX RESET 間または PMA\_INIT 間のディアサート時間の差に応じて、シンプレックス TX の `SIMPLEX_TIMER_VALUE` パラメーターを前述の基準を満たすように調整する必要があります。 `SIMPLEX_TIMER_VALUE` パラメーターは、`<user_component_name>_core.v` 内で変更できます。

- `rx_reset` の後に `tx_reset` がディアサートされる場合、リンク動作にはデフォルト値の 12 ビットで十分対応できます。
- `rx_reset` の前に `tx_reset` がディアサートされる場合、シンプレックス TX の `SIMPLEX_TIMER_VALUE` パラメーターはリセット信号のディアサート時間の差に応じて調整する必要があります。

## リセットおよびパワー ダウン

### リセット

制御およびステータス インターフェイスの `reset` 信号を使用して、Aurora 64B/66B コアを既知のデフォルト状態に設定します。コアをリセットすると、現在動作しているすべてのチャンネルが停止します。リセット後、コアはチャンネルを初期化します。Aurora チャンネル パートナー 1 のリセット信号がアサートされると、チャンネル パートナー 2 もロックを失います。パートナー 1 がリセット状態から回復し、有効なパターンを送信すると、チャンネル パートナー 2 のロックも回復します。

フルデュプレックス モジュールの場合、`user_clk` の立ち上がりエッジで `reset` 信号がアサートされると、チャンネルの両側 (TX および RX) がリセットされます。シンプレックス Aurora コアには、両側にリセット ポートがあります。`pma_init` がアサートされると、シリアル トランシーバー全体がリセットされ、最終的に Aurora コアもリセットされます。

## リセット シーケンス

サンプル デザイン レベルで推奨される Aurora 64B/66B コアの リセット シーケンスは次のとおりです。図 2-34 を参照してください。

1. リセット信号をアサートします。少なくとも user\_clk の 128 サイクル間待機します。
2. pma\_init をアサートします。pma\_init 信号と reset 信号のアサート状態を 1 秒以上保持します。これによって、CC 文字の送信を防ぐことができ、リモート エージェントがホット プラグ イベントを確実に検出します。第 3 章の「ホットプラグ ロジック」を参照してください。
3. pma\_init をディアサートします。
4. reset をディアサートします。

### 注記 :

1. 前述のリセット シーケンスは、参照用として <user\_component\_name>\_exdes.v 内にインプリメントされています。
2. シンプレックスの場合、TX リセットは RX リセットより先にアサートする必要があります。これによって、TX リセットがアサートされると常にシンプレックス TX が NA アイドル文字を送信するようになります。これらの文字をチャネル パートナー (シンプレックス RX) が受信することによって、リンクが切断されます。
3. TX/RX シンプレックス コアの場合、reset および pma\_init 入力ポートのアサートによって、コアの TX と RX が両方および GT がそれぞれリセットされます。reset 信号と pma\_init 信号の接続は、デュプレックス コアの場合と同じです。

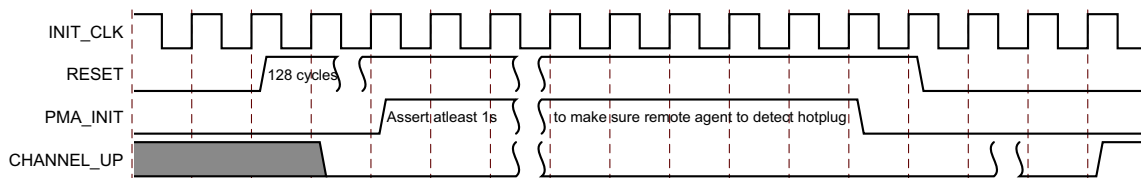


図 2-34: リセット シーケンス

## pma\_init のステージング

サンプル デザイン レベルの最上位 pma\_init 入力、128 サイクル遅延されます (pma\_init\_stage)。この信号は、24 ビット カウンター時間延長されます (pma\_init\_assertion)。これらを統合した信号が、pma\_init 入力としてコアに提供されます。したがって、コアで pma\_init がアサートされると、コア全体に対して reset がアサートされるようになります。

<user\_component\_name>\_support\_reset\_logic.v 内のデバウンサー ロジック (reset\_debounce\_r) は、gt\_reset\_in signal (pma\_init\_assertion) 信号が High になるまでリセット状態を保持します。これによって、最上位の pma\_init 信号がアサートされると常に、内部生成されたリセットが生じることになります。

図 2-35 に pma\_init のビヘイビアを示します。

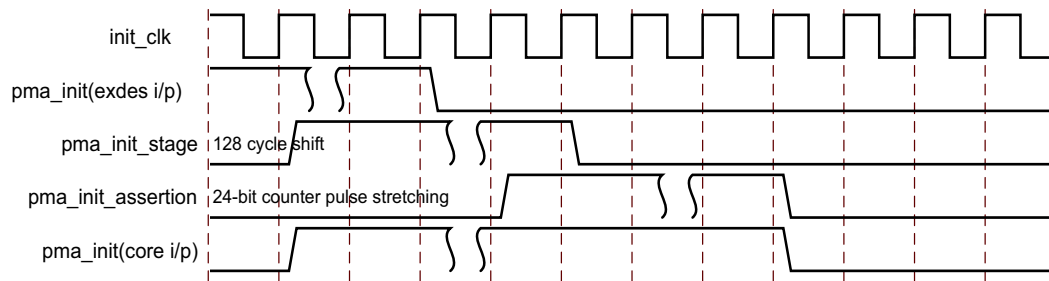


図 2-35 : pma\_init のステージング

コアに対して pma\_init 信号をアサートすると、チャンネルパートナー コアでホットプラグ リセットがアサートされます。図 2-36 に、ホットプラグ リセットのアサート後のリセット シーケンスを示します。

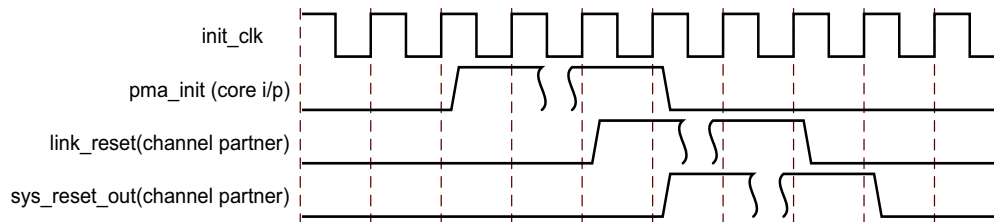


図 2-36 : リモート システム リセットの pma\_init

## リセット フロー

サンプル デザインの場合、最上位の reset 入力はデバウンス処理が行われ、コアへ接続されます (reset\_pb)。この信号は、シリアル トランシーバーのリセット ステータスとコアから送信されるホットプラグ リセットをコア リセット ロジックで統合して、コアのリセット信号を生成します (sys\_reset\_out)。この信号は、コアの reset 入力へ接続される必要があります。図 2-37 に、リセット フローを示します。

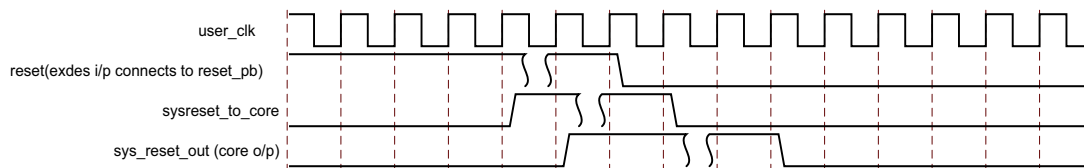


図 2-37 : リセット フロー

注記 :

1. 上記の要件を満たすには、コアの入力で reset\_pb 信号と reset 信号を相互接続しないでください。
2. コアの reset 入力を駆動するには、sys\_reset\_out および必要に応じてその他のシステム固有のリセットを使用してください。

## パワー ダウン

power\_down がアサートされると、Aurora 64B/66B コア ロジックのみリセット状態となります。これによって、デザインで使用されている GTX または GTH トランシーバーの電源は切断されません。

## タイミング

図 2-38 に、reset 信号のタイミングを示します。ノイズが少ない環境では、一般的に  $t_{CU}$  は 500 クロック未満となり、ノイズが多い環境の  $t_{CU}$  は長くなります。

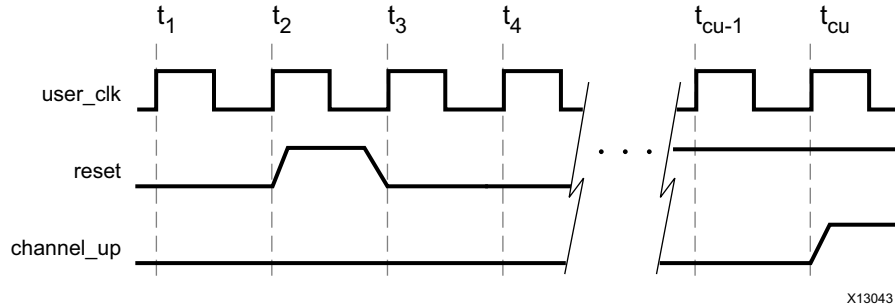


図 2-38: リセットおよびパワー ダウンのタイミング

## リセットの使用ケース

### 使用ケース 1: デュプレックス コアにおける reset のアサート

デュプレックス コアでの reset 信号のアサートは、少なくとも user\_clk 信号の 128 サイクル分必要です。これを受けて、図 2-39 に示すように channel\_up がディアサートされます。

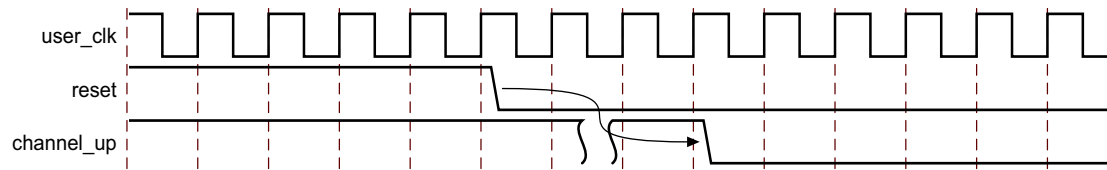


図 2-39: デュプレックス コアにおけるリセットのアサート

### 使用ケース 2: デュプレックス コアにおける PMA\_INIT のアサート

図 2-40 では、デュプレックス コアにおける pma\_init 信号のアサートを示しています。この信号のアサートは、少なくとも init\_clk の 128 サイクル分必要です。その結果、トランシーバーからの txoutclk がなくなり、channel\_up がディアサートされるため、数クロック サイクル後には user\_clk が停止します。

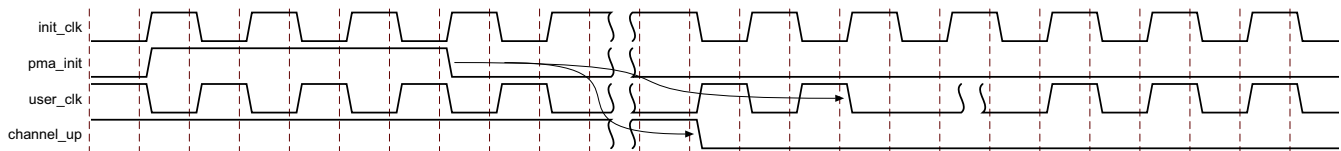


図 2-40: デュプレックス コアにおける pma\_init のアサート

### 使用ケース 3 : シンプレックス コアにおけるリセットのアサート

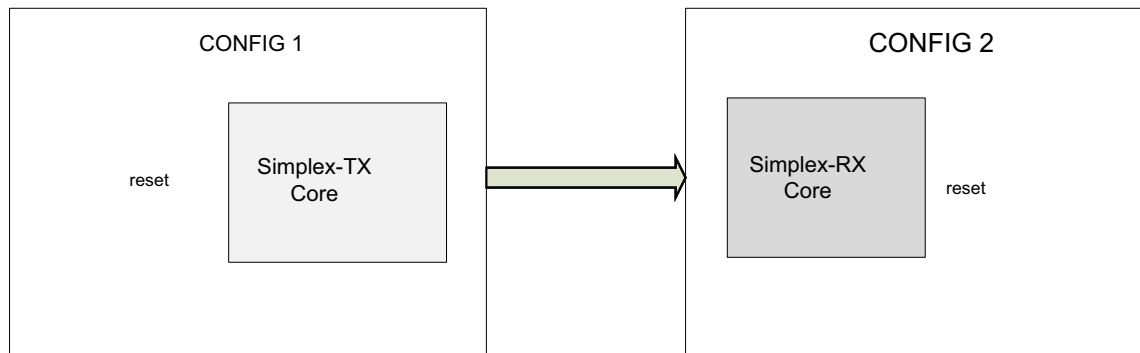


図 2-41 : シンプレックス コアを含むシステム

図 2-41 は、システム内で接続されたシンプレックス TX コアとシンプレックス RX コアを示しています。CONFIG1 と CONFIG2 は、同じデバイスまたは複数のデバイス内に含めることができます。

シンプレックス コアにおける TX コア リセットおよび RX コア リセットの推奨されるアサート手順を次に示します。

1. RX コアの reset 信号が user\_clk の 128 サイクル間アサートされ、その後 RX シンプレックス コアの reset 信号が user\_clk の 128 サイクル間アサートされます。
2. 少なくとも user\_clk が 5 サイクル経過した後に tx\_channel\_up および rx\_channel\_up がディアサートされます。
3. RX シンプレックス コアの reset 信号は、TX シンプレックス コアの reset 信号がディアサートされる前にディアサートまたはリリースされます。これによって、シンプレックス TX コアが TX\_CHANNELUP に達する前に、シンプレックス RX コアのトランシーバーが CDR 用に十分な時間を確保できます。

4. tx\_channel\_up がアサートされる前に rx\_channel\_up がアサートされます。この条件は、シンプレックス RX コアで満たされなければならないため、シンプレックス TX コアのシンプレックス タイマー パラメーター (SIMPLEX\_TIMER\_VALUE) を調整してこの条件を満たす必要があります。SIMPLEX\_TIMER\_VALUE パラメーターは、<user\_component\_name>\_core.v 内で変更できます。
5. シンプレックス TX コアが Aurora チャンネル初期化シーケンスの送信を設定された時間で完了すると、tx\_channel\_up がアサートされます。tx\_channel\_up が最後にアサートされることによって、シンプレックス RX コアの準備が整った状態でシンプレックス TX コアが確実に Aurora 初期化シーケンスを送信します。

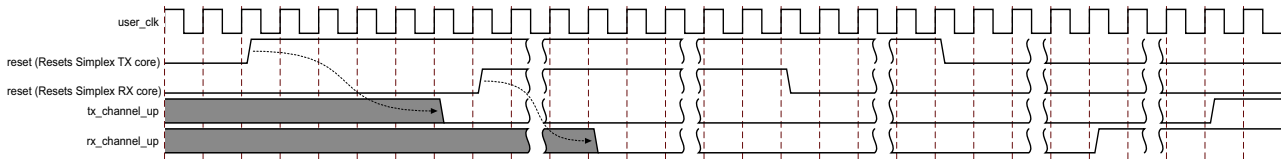


図 2-42: シンプレックス コアにおける RESET のアサート

6. TX/RX シンプレックス コアの場合、reset と pma\_init 信号がアサートするため、デュプレックス コアのリセットシーケンスが続きますが、コアの使用モデルに基づいて SIMPLEX\_TIMER\_VALUE を調整する必要があります。

## DRP インターフェイス

DRP インターフェイスは、トランシーバーブロックのステータスを制御またはモニターします。ユーザー アプリケーションは、DRP ポートを介して値を書き込む/読み出すことによって、シリアルトランシーバーの設定へのアクセスや変更が可能です。ネイティブ インターフェイスは、ネイティブ トランシーバー DRP インターフェイスを提供します。DRP ポートへのアクセスには、AXI4-Lite インターフェイスを選択することも可能です。

表 2-22 : AXI4-Lite 信号の定義

名前	方向	説明
s_axi_awaddr	入力	DRP 用の AXI4-Lite 書き込みアドレス
s_axi_awvalid	入力	書き込みアドレスの valid 信号
s_axi_awready	出力	書き込みアドレスの ready 信号
s_axi_araddr	入力	読み出しアドレス
s_axi_arvalid	入力	読み出しアドレスの valid 信号
s_axi_arready	出力	読み出しアドレスの ready 信号
s_axi_wdata	入力	書き込みデータ
s_axi_wvalid	入力	書き込みの valid 信号
s_axi_wready	出力	書き込みの ready 信号
s_axi_bvalid	出力	書き込み応答の valid 信号
s_axi_rdata	出力	読み出しデータ
s_axi_rvalid	出力	読み出しの valid 信号
s_axi_rready	入力	読み出しの ready 信号
s_axi_bready	入力	書き込み応答の valid 信号

注記 : DRP の読み出し動作の開始時点で、コアはユーザー AXI4-Lite インターフェイスがデータ取得可能な状態であることを求めます。

表 2-23 : DRP ポート信号の定義

ポート	方向	クロックドメイン	説明
drpaddr[8:0]	入力	DRPCLK	DRP アドレスバス
drpclk	入力	N/A	DRP インターフェイスクロック
drpen	入力	DRPCLK	DRP のイネーブル信号です。 0 : 読み出しまたは書き込み処理が無効 1 : 読み出しまたは書き込み処理が有効 書き込み処理の場合、drpwe および drpen を drpclk の 1 サイクル間のみ High に駆動する必要があります。正しい動作については、 <a href="#">図 2-31</a> を参照してください。
drpdi[15:0]	入力	DRPCLK	FPGA ロジックからトランシーバーへコンフィギュレーションデータを書き込むためのデータバスです。
drprdy	出力	DRPCLK	DRP 書き込み処理が完了し、読み出しデータが有効であることを示します。
drpdo[15:0]	出力	DRPCLK	GTX/GTH トランシーバーから FPGA ロジック リソースへコンフィギュレーションデータを読み出すためのデータバスです。
drpwe	入力	DRPCLK	DRP の書き込み イネーブル信号です。 0 : drpen が 1 のときに読み出し処理を実行 1 : drpen が 1 のときに書き込み処理を実行 書き込み動作を行う場合、drpwe および drpen は drpclk の 1 サイクル間のみ High に駆動する必要があります。

注記 : UltraScale デバイスの場合、DRP ポート名は gt<lane>\_drp\* から始まります (lane = レーン数)。

AXI4-Lite インターフェイスからの Write Address または Read Address チャンネルが各 valid/ready 信号のアサートによってアクティブの場合、DRP インターフェイスは drpen をアサートします。書き込み動作の drpwe 信号は、AXI4-Lite インターフェイスからの Write Data チャンネルがアクティブの場合に有効となります。AXI4-Lite インターフェイスからの Read Data チャンネルが有効の場合、drpdo には drpaddr で指定されたアドレスのデータが含まれます。

## クロック補正インターフェイス

このインターフェイスは、データを送信するモジュールに含まれていて、クロック補正の管理に使用されます。do\_cc ポートが High に駆動されるたびに、コアはデータのフローおよびフロー制御メッセージを停止し、その後にクロック補正シーケンスを送信します。各 Aurora 64B/66B コアには、『Aurora 64B/66B プロトコル仕様 v1.2』(SP011) [参照 5] に従って、クロック補正インターフェイスの駆動に使用されるクロック補正管理モジュールがあります。チャンネルの両側で同じ物理クロックが使用され、ホットプラグ ロジックが無効に設定されている場合は、do\_cc を Low に接続する必要があります。ただし、信頼性の高いリンク動作を求める場合には、CC ロジックを有効にすることを強く推奨します。

すべての Aurora 64B/66B コアには、クロック補正シーケンスの伝送を制御するクロック補正インターフェイスがあります。表 2-24 では、クロック補正インターフェイス ポートの機能について説明しています。

表 2-24: クロック補正 I/O ポート

名前	方向	説明
do_cc	入力	この信号がアサートされている場合、Aurora 64B/66B コアはすべてのクロック サイクルですべてのレーンに CC シーケンスを送信します。CC モジュールの do_cc 出力に接続します。

# コアを使用するデザイン

この章では、コアを使用してより簡単に設計するためのガイドラインおよび追加情報を紹介します。

---

## 一般的なデザイン ガイドライン

すべての Aurora 64B/66B コアのインプリメンテーションでは、システム性能の要件に注意を払う必要があります。パインライン処理、ロジック マップ、配置制約、およびロジック複製は、システム性能を向上させる最適な手段です。

### レジスタの使用

FPGA デザインのタイミングをシンプルにしたり、システム性能を向上させるには、ユーザー アプリケーションとコア間のすべての入力と出力にレジスタを使用してください。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。信号のレジスタへの格納はすべてのパスで可能とは限りませんが、これによってタイミング解析が容易になり、またザイリンクス 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を認識

コアのサンプル デザインに付属する XDC ファイルは、クリティカルな信号を識別して適用すべきタイミング制約を特定するのに役立ちます。

### サポートされているデザイン フローを使用

コアは、Verilog ソース コードとして提供されます。サンプルのインプリメンテーション スクリプトでは、合成ツールとして XST を使用しており、このツールはコアに同梱されています。その他の合成ツールも使用可能です。

### 許可された変更のみ実行

Aurora 64B/66B コアはユーザーが変更を加えることができません。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。IP カタログのオプション選択を使用して、Aurora 64B/66B コアのサポートされたユーザー コンフィギュレーションのみ利用できます。

---

## 共有ロジック

コアのバージョン 8.1 までは、RTL 階層が固定されていました。このため、共有可能なクロッキングやリセット ロジックはコアのサンプル デザインから抽出してからコアの単一/複数インスタンスで使用する必要があり、難点がありました。

共有ロジックは、より柔軟なアーキテクチャを提供する新しい機能であり、スタンドアロン コアとして、または 1 つ以上のインスタンスを含むより大規模なデザインの一部として使用されます。この機能は、必要な HDL の変更を最小限に抑えると同時に、多くの使用ケースに対応できる柔軟性を備えています。

新しい階層レベルは、<user\_component\_name>\_support と呼ばれています。図 3-1 および図 3-2 に、共有ロジックブロックがコアまたはサンプル デザインに含まれる 2 つの階層を示します。図中の <user\_component\_name>には生成されたコアの名前が入ります。この 2 つの階層の違いは、コアの境界線です。これは、Vivado® IDE の [Shared Logic] を使用して指定します。

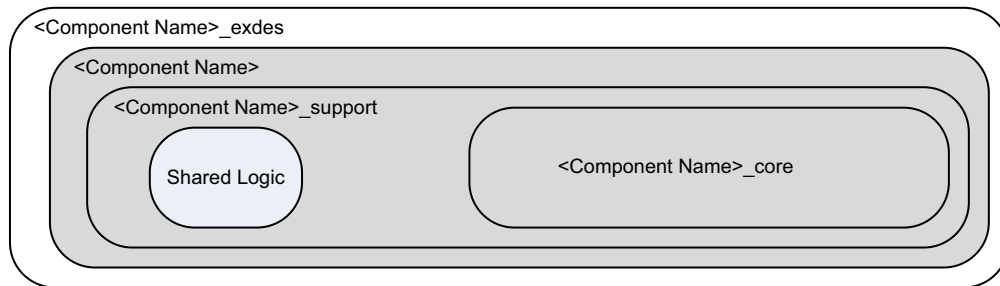


図 3-1: コアに含まれた共有ロジック (グレイ表示部分が xci top)

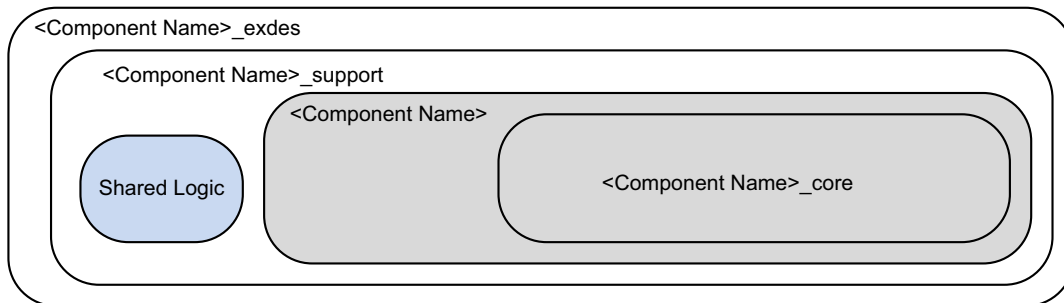


図 3-2: サンプル デザインに含まれた共有ロジック (グレイ表示部分が xci top)

共有ロジックの内容は、物理インターフェイスとターゲット デバイスによって異なります。共有ロジックには、GT 差動バッファのインスタンス (IBUFDS\_GTE2)、サポート リセット ロジック、および <=:USER\_COMPONENT\_NAME:>\_CLOCK\_MODULE のインスタンスが含まれます。これらのブロックのほかに、トランシーバーの COMMON ブロック インスタンスも含まれます。選択したトランシーバーの種類 (GTX または GTH) に応じて、トランシーバーの共有ブロックがインスタンス化されます。サポート リセット ロジックには、reset および gt\_reset ポート用のデバウンス ロジックが含まれます。

表 3-1 に [Shared Logic] オプション設定によるポート変更について説明しています。

表 3-1 : [Shared Logic] によるポート変更

NAME	方向	説明	備考
gt_refclk1_p gt_refclk1_n	入力	差動トランシーバーの基準クロック 1	[Include Shared Logic in Core] がオンの場合に有効になります。
gt_refclk2_p gt_refclk2_n	入力	差動トランシーバーの基準クロック 2	[Shared Logic in Core] がオンで、2 つ以上の基準クロックが必要な場合に有効になります。
refclk1_in	入力	シングルエンド トランシーバーの基準クロック 1	[Shared Logic in Example Design] がオンの場合に有効になります。
refclk2_in	入力	シングルエンド トランシーバーの基準クロック 2	[Shared Logic in Example Design] がオンで、2 つ以上の基準クロックが必要な場合に有効になります。
user_clk_out	出力	ユーザー クロック出力	[Include Shared Logic in Core] がオンの場合に有効になります。
init_clk_out	出力	INIT クロック出力	[Include Shared Logic in Core] がオンの場合に有効になります。7 シリーズデバイスでのみ有効です。
sync_clk	入力	サポート ロジックからの同期クロック入力です。	[Shared Logic in Example Design] がオンの場合に有効になります。
sync_clk_out	出力	サポート ロジックで使用される同期クロック出力です。	[Include Shared Logic in Core] がオンの場合に有効になります。
reset_pb	入力	プッシュ ボタン リセットであり、サンプル デザイン レベルの最上位 reset 入力です。サポート リセット ロジックがコア内に含まれているため、この入力が必要になります。	
gt_reset_out	出力	gt_reset ポート用デバウンス ロジックの出力	[Include Shared Logic in Core] がオンの場合に有効になります。
gt_refclk1_out	出力	シングルエンド トランシーバーの基準クロック	[Include Shared Logic in Core] がオンの場合に有効になります。
gt_refclk2_out	出力	シングルエンド トランシーバーの基準クロック	[Include Shared Logic in Core] がオンの場合に有効になります。
mmcm_not_locked_out	出力	クロック モジュールからの mmcm_not_locked 信号	
gt_rxcdrvrdn_in	入力	ループバック モードで GT をコンフィギュレーションする場合に使用される RXCDR オーバーライド	

表 3-1 : [Shared Logic] によるポート変更 (続き)

NAME	方向	説明	備考
gt_qpllclk_quad<quad>_in gt_qpllrefclk_quad<quad>_in	入力	GTXE2_COMMON、 GTHE2_COMMON、 GTHE3_COMMON で生成され るクロック入力	<quad> は 1 から 12 までのアク ティブトランシーバーのクワッド を表します。 [Shared Logic in Example Design] が オンの場合に有効になります。 GTX または GTH トランシーバー デザインに適用されます。これら のポートは、Vivado Design Suite で コア コンフィギュレーション中に Vivado IDE で選択してクワッドご とに有効化されます。
gt_qpllclk_quad<quad>_out gt_qpllrefclk_quad<quad>_out	出力	GTXE2_COMMON、 GTHE2_COMMON、 GTHE3_COMMON で生成され るクロック出力	<quad> は 1 から 12 までのアク ティブトランシーバーのクワッド を表します。 [Include Shared Logic in Core] がオ ンの場合に有効になります。GTX または GTH トランシーバー デザ インに適用されます。これらの ポートは、Vivado Design Suite でコ ア コンフィギュレーション中に Vivado IDE で選択してクワッドご とに有効化されます。
gt_to_common_qpllreset_out	出力	スレーブ共有ロジックで使用 される QPLL 共有リセット出力	[Shared Logic in Example Design] が オンで、QPLL が使用されている場 合に有効になります。
gt_qplllock_quad<quad>_in gt_qpllrefclklost_quad<quad>_in	入力	マスター共有ロジックから入 力される QPLL のロック信号お よび refclock のロスト信号	[Shared Logic in Example Design] が オンで、QPLL が使用されている場 合に有効になります。<quad> は 1 から 12 までのアクティブ トラン シーバーのクワッドを表します。
gt_qplllock_quad<quad>_out gt_qpllrefclklost_quad<quad>_out	出力	スレーブ共有ロジックへ出力 される QPLL のロック信号およ び refclock のロスト信号	[Shared Logic in Core] がオンで、 QPLL が使用されている場合に有 効になります。<quad> は 1 から 12 までのアクティブ トランシーバー のクワッドを表します。
init_clk_p init_clk_n	入力	差動のフリーランニング シス テム/ボード クロック	[Include Shared Logic in Core] がオン の場合に有効になります。7シリー ズ デバイスでのみ有効です。
sys_reset_out	出力	サンプル デザイン レベルのロ ジックで使用されるシステム リセット出力	
init_clk	入力	フリーランニングのシステム/ ボード クロック	7 シリーズ デバイスでのみ有効で す。

## クロッキング

UltraScale™、Zynq®-7000、Virtex®-7、および Kintex®-7 デバイスの Aurora 64B/66B コアを正常に動作させるには、最良のクロッキングが不可欠です。コアは、GTX/GTH トランシーバーの高速 TX クロックおよびクロック リカバリ回路を駆動するために、低ジッターの基準クロックが必要です。また、ユーザー アプリケーションとの同期動作のために、1 つ以上の周波数ロックされたパラレルクロックが必要です。

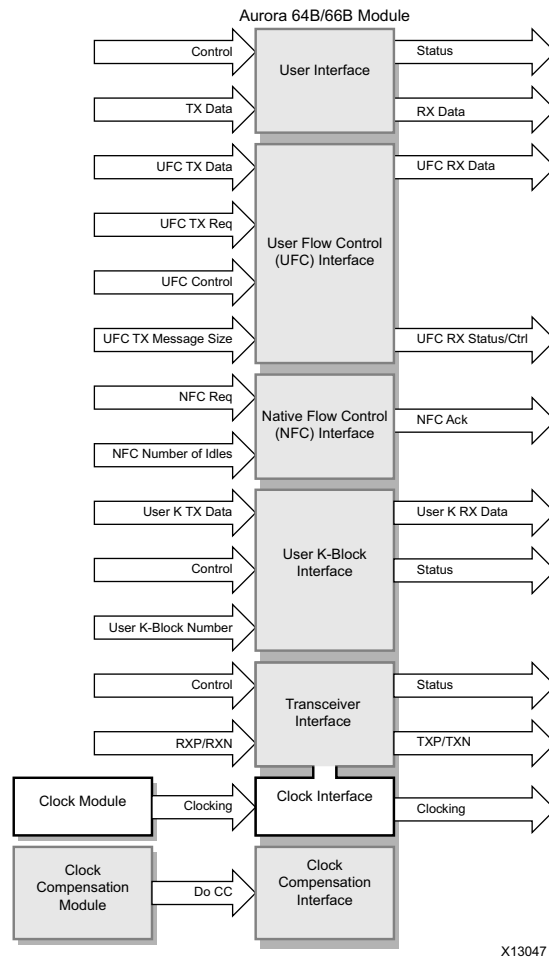


図 3-3 : 最上位のクロッキング

各 Aurora 64B/66B コアは、`aurora_example` というデザインを含む `example_project` ディレクトリに生成されます。このサンプル デザインは生成された Aurora 64B/66B コアをインスタンス化することで、コアで有効なクロック コンフィギュレーションを実証します。Aurora コアを初めて使用する場合は、サンプル デザインを検証して、クロック インターフェイスを接続する際のテンプレートとして使用してください。

## クロック インターフェイスおよびクロッキング

### Aurora 64B/66B のクロッキング アーキテクチャ

図 3-4 に、Zynq-7000、Virtex-7、および Kintex-7 デバイスの GTX または GTH トランシーバーの Aurora 64B/66B コアのクロッキング アーキテクチャを示します。

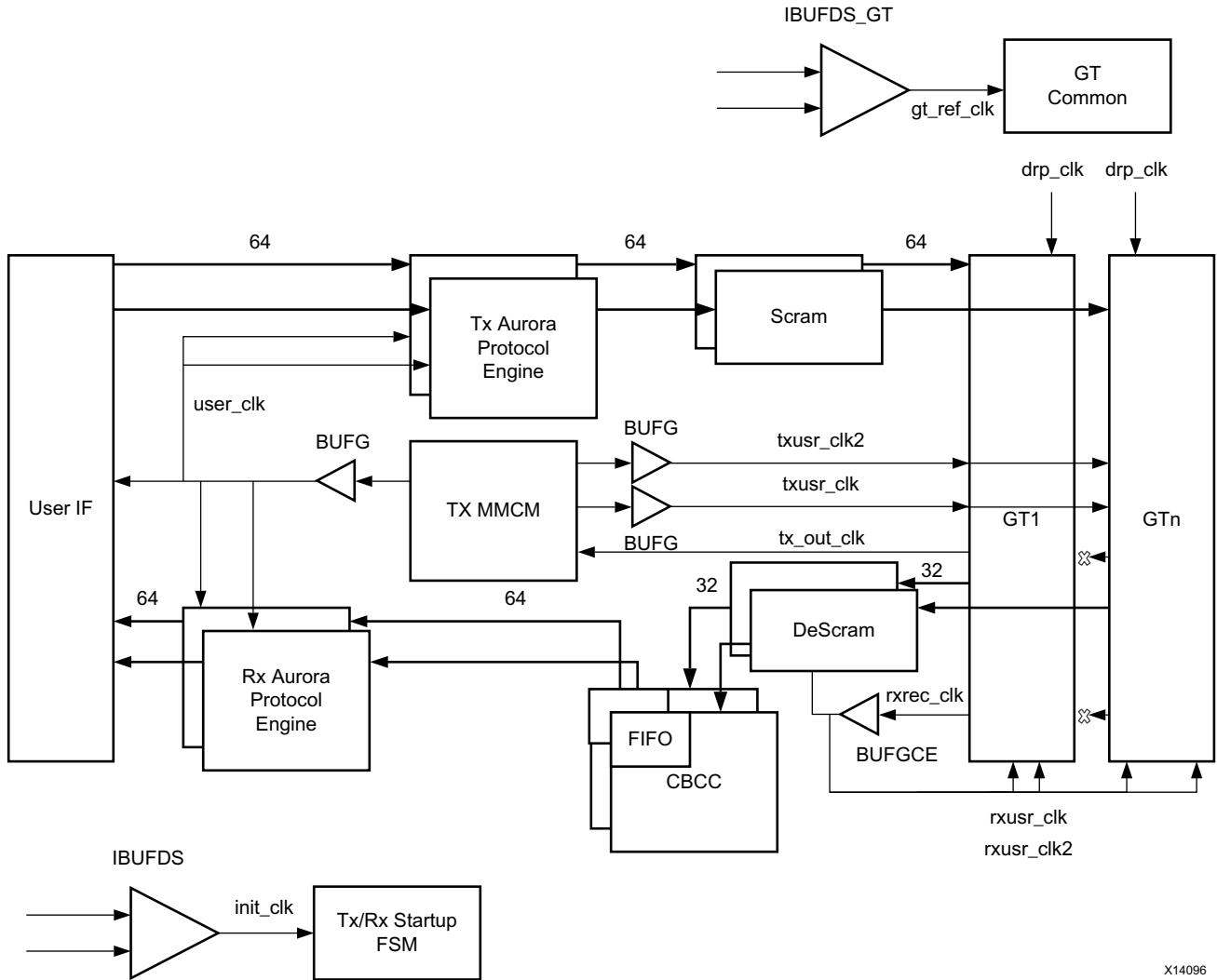


図 3-4 : Aurora 64B/66B のクロッキング (Zynq-7000、Virtex-7、および Kintex-7 デバイスの GTX/GTH トランシーバーの場合)

## user\_clk、sync\_clk、および tx\_out\_clk の接続

Aurora 64B/66B コアは、位相ロックされた3つのパラレルクロックを使用します。1つ目の user\_clk 信号は、コアとユーザーアプリケーション間のすべての信号を同期化します。コアに接続されるすべてのロジックは、user\_clk で駆動される必要があり、この信号はグローバルクロックバッファ (BUFG) を介す必要があります。

user\_clk 信号は、シリアルトランシーバーの txusrclk2 ポートを駆動するために使用します。64B/66B エンコードとデコードを考慮して、モジュールのパラレル側のデータレートがモジュールのシリアル側のデータレートと一致するように、tx\_out\_clk が選択されます。

3つ目の位相ロックされたパラレルクロックは、sync\_clk です。このクロックも BUFG を介す必要があり、シリアルトランシーバーの txusrclk ポートを駆動するために使用されます。また、このクロックは Aurora 64B/66B コアにも接続されて、シリアルトランシーバーの内部同期化ロジックを駆動します。

より簡単に2つのパラレルクロックを使用できるように、example\_design/support または src の下 (共有ロジックの設定によって異なる) にある clock\_module というサブディレクトリにクロックモジュールが提供されています。このモジュールの各ポートについては、34 ページの表 2-16 で解説しています。このクロックモジュールを使用する場合は、クロックモジュールの mmcm\_not\_locked 出力へ mmcm\_not\_locked 信号を接続する必要があります。つまり、tx\_out\_clk をクロックモジュールの clk へ接続し、pll\_lock をクロックモジュールの pll\_not\_locked ポートへ接続します。このモジュールを使用しない場合は、mmcm\_not\_locked 信号をいずれかのパラレルクロックの生成に使用される PLL からの locked 信号の反転バージョンへ接続し、tx\_out\_clk を PLL ソースクロックとして使用する場合は、安定をはかる間、pll\_lock 信号を使用して PLL をリセット状態に保持します。pma\_init がアサートされている間、txusrclk は安定していません。したがって、コアは MMCM の同期化に安定したクロック (init\_clk) を使用します。安定したクロックを使用してサンプルすることで、リンクに高い堅牢性が備わります。

MMCM を使用して安定したクロック (init\_clk) を生成する場合は、MMCM がロックされるまで Aurora コアに pma\_init を適用する必要があります。これによって、安定したクロックがコアで利用できるようになるまで、コアは既知のステートを維持します。

## Aurora 64B/66B コアにおける BUFG の使用

Aurora 64B/66B コアは、GTX/GTH トランシーバーを使用する特定のコアコンフィギュレーションで4つの BUFG を使用します。Aurora 64B/66B は8バイトにアラインされたプロトコルであり、ユーザーインターフェイスからのデータバスは8バイトにアラインされています。GTX/GTH トランシーバーの場合、コアは送信バスを8バイトとして構成し、受信バスを4バイトとして構成します。

CB/CC ロジックはコアの内部に含まれ、シリアルトランシーバーから受信するリカバリクロックに基づいています。BUFG の使用は、すべてのコアコンフィギュレーションで同じであり、コア機能によって増加することはありません。

## FPGA デザインの基準クロック

Aurora 64B/66B コアは、GTX/GTH トランシーバーで高速シリアル クロックを生成および回復するために、低ジッターの基準クロックを必要とします。各基準クロックは基準クロック入力ポート (gtxq/gthq) に設定できます。ジッターを低減してビット エラーを回避するには、できる限り基準クロックを高品質のクロック ソースに接続する必要があります。DCM はジッターを多く発生させるため、基準クロックの駆動には使用しないでください。

Zynq-7000、Virtex-7、および Kintex-7 デバイスのマルチレーン デザインの場合、Aurora 64B/66B ウィザードでは、上下クロッキング条件に従って、選択したクワッドの上下に位置するクワッドを選択できます。クワッドの選択が3つのクワッド境界を超える場合は、2つ目の基準クロック ソースを選択できます。上下クロッキングの詳細は、『7 シリーズ FPGA GTX/GTH トランシーバー ユーザー ガイド』(UG476) [参照 4] を参照してください。

UltraScale デバイスの場合、ザイリンクスのインプリメンテーション ツールは、上下配線への必要な調整を行い、また必要に応じて GTHE3 トランシーバー クロック入力へのピンを切り替えて別のクワッドへクロックを配線します。

1 組のクロック ピン ペアでは、最大 20 の GTH トランシーバーへクロックを供給できます。



**重要 :** 基準クロックを共有する場合、このようなコンフィギュレーションで発生するジッターを高速デザインのジッター マージン要件内に抑えるには、次の規則に従う必要があります。ソースとなっているクワッドの上にある GTH トランシーバー クワッドの数は最大 2 つまでです。ソースとなっているクワッドの下にある GTX /GTH トランシーバー クワッドの数は最大 2 つまでです。

## クロック補正

クロック補正は、Aurora チャネルの両側で使用される基準クロック周波数を  $\pm 100\text{ppm}$  の精度で補正する機能です。この機能は、チャネルで接続された各デバイスに独立した基準クロック ソースを使用し、データの送信と受信に同じ `user_clk` を使用するシステムで使用されます。

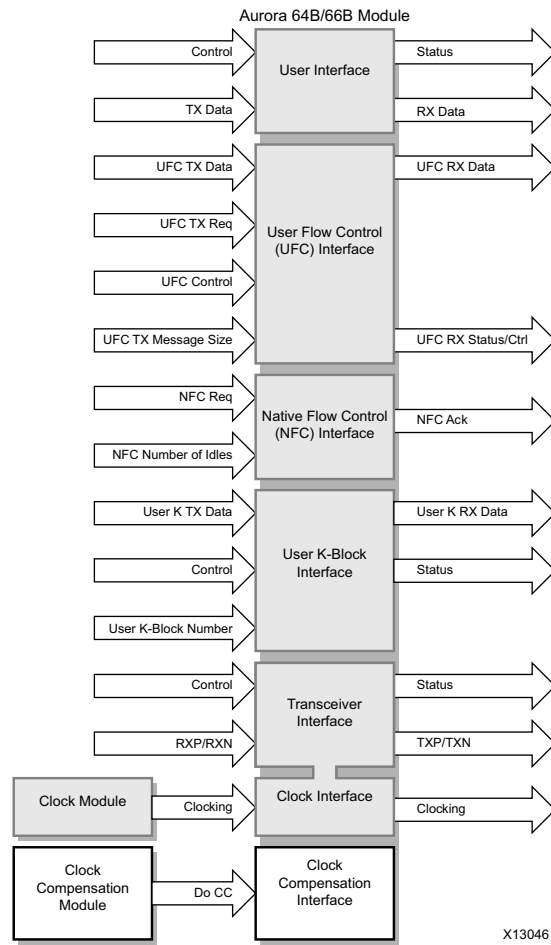


図 3-5 : 最上位のクロック補正インターフェイス

Aurora 64B/66B コアのクロック補正インターフェイスによって、コアのクロック補正機能全体が完全に制御されま。標準のクロック補正モジュールは Aurora 64B/66B コアと共に生成され、独立した基準クロック ソースを使用して Aurora 準拠のクロック補正機能をシステムに提供します。特殊なクロック補正要件がある場合は、カスタム ロジックを使用してインターフェイスを駆動できます。チャネルの両側に対して同じ基準クロック ソースが使用される場合は、インターフェイスをグランドに接続してクロック補正機能を無効にできます。

図 3-6 および図 3-7 に、`do_cc` 信号の動作を波形図で示しています。

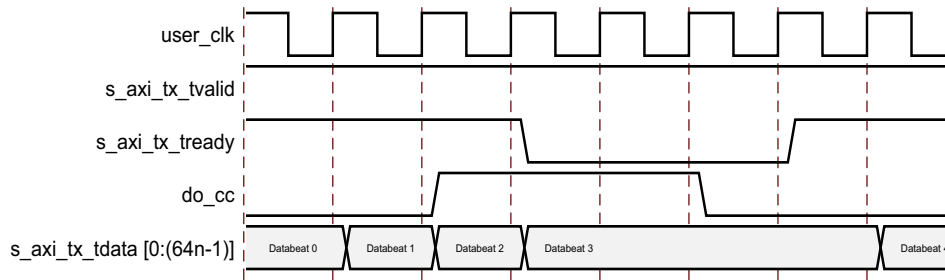


図 3-6: クロック補正シーケンスが挿入されるストリーミング データ



図 3-7: クロック補正によって中断されるデータ受信

Aurora プロトコルは、Aurora チャンネルの各側における基準クロックの差を  $\pm 100\text{ppm}$  以内にすることを規定しています。Aurora 準拠のクロック補正機能を実行するには、`user_clk` の 10,000 サイクルごとに 3 サイクル間 `do_cc` をアサートする必要があります。`do_cc` がアサートされている間、TX ユーザー インターフェイス上で `s_axi_tx_tready` がデアサートされると、チャンネルはクロック補正シーケンスを送信するために使用されます。

Vivado 設計ツールでは各 Aurora 64B/66B コアが生成されると同時に、標準のクロック補正 (CC) モジュールが `example_design` の下にある `cc_manager` サブディレクトリに生成されます。このモジュールは、`do_cc` ポートに自動的にパルスを生成して Aurora 準拠のクロック補正シーケンスを生成します。このモジュールは、例外を除いて常に Aurora モジュールのクロック補正ポートへ接続される必要があります。表 3-2 では標準 CC モジュールのポートについて説明しています。

表 3-2: 標準の CC モジュールの I/O ポート

名前	方向	説明
<code>do_cc</code>	出力	Aurora 64B/66B コアの <code>warn_cc</code> 入力へ接続します。
<code>channel_up</code>	入力	フルデュプレックス コアの <code>channel_up</code> 出力へ、または TX のみシンプレックスのポートの <code>tx_channel_up</code> 出力へ接続します。

Aurora チャンネルの両側が同じクロックで駆動されている場合 (74 ページの図 3-7 参照)、モジュールの両側で基準クロック周波数がロックされるため、クロック補正は必要ありません。この場合、`do_cc` はグラウンドに接続してください。

その他、標準のクロック補正モジュールが適合しない特殊な例があります。そのような場合は `do_cc` ポートを使用して、特定チャンネルの要件を満たすために任意のタイミングと長さでクロック補正シーケンスを送信できます。この機能の最も一般的な用法は、フレームの外、そしてデータフローを中断しないようにストリーム中の特定時に、クロック補正イベントを生じさせるようスケジューリングすることです。



**重要：**一般的にクロック補正ロジックのカスタマイズは推奨していませんが、カスタマイズが必要な場合は、詳しい解析とテストを実施して次のガイドラインに従って注意深く設計する必要があります。

- クロック補正シーケンスは、すべてのレシーバーで確実に認識されるように少なくとも `user_clk` の 3 サイクル間アサートする必要があります。
- 使用するクロック周波数の最大差を十分補正できる期間と周期が選択されている必要があります。
- 8 サイクル間に複数のクロック補正シーケンスを続けて実行しないでください。
- ホットプラグロジックが有効の場合には、クロック補正機能を無効にしないでください。

## コアの機能

このセクションでは、次に示す Aurora 64B/66B コアの機能について説明します。

- 「CRC」
- 「Vivado ラボ ツールの使用」
- 「ホットプラグロジック」
- 「リトルエンディアン形式のサポート」

### CRC

フレーミング ユーザー データ インターフェイス用にインプリメントされている 32 ビットの CRC は `<component name>_crc_top.v` モジュールにあります。 `crc_valid` 信号と `crc_pass_fail_n` 信号が、受信した CRC と送信した CRC の結果を示します (表 3-3 参照)。

表 3-3：CRC モジュールのポート

ポート名	方向	説明
<code>crc_valid</code>	出力	<code>crc_pass_fail_n</code> 信号をサンプルするアクティブ High 信号です。
<code>crc_pass_fail_n</code>	出力	受信した CRC が送信した CRC と一致する場合に、 <code>crc_pass_fail_n</code> 信号がアサートされます。受信した CRC が送信した CRC と一致しない場合、この信号はアサートされません。 <code>crc_pass_fail_n</code> 信号は、常に <code>crc_valid</code> 信号を使用してサンプルされます。

### Vivado ラボ ツールの使用

ボード デザインのデバッグおよび検証に非常に有効な ILA コアと VIO コアが Aurora 64B/66B コアと共に提供されています。Aurora 64B/66B コアは、対応する信号を VIO コアに接続して、デザインの構築およびデバッグを容易にします。コアの Vivado IDE 環境で [Vivado Lab Tools] をオンにして、サンプル デザインの一部に含めます (79 ページの図 4-1 参照)。

Vivado ラボ ツールを有効にして生成されたコアには、3 つの VIO インターフェイスと 1 つの ILA インターフェイスが備わります。

- `viol_inst` – コアの Lane Up、Channel Up、Data Error カウント、Soft Error カウント、Channel Up トランジション カウント、System Reset、GT Reset、および Loopback ポートが含まれる

- vio2\_inst – リセット クオリティ カウンターのステータスが含まれる
- vio3\_inst – リピート リセット テストの合格/不合格ステータスが含まれる

## ホットプラグ ロジック

Aurora 64B/66B デザインのホットプラグ ロジックは、受信したクロック補正文字に基づきます。Aurora の RX インターフェイスでクロック補正文字を受信するという事は、通信チャンネルが有効つまり破損していないことを意味します。あらかじめ指定した時間にクロック補正文字を受信されない場合は、ホットプラグ ロジックがコアとトランシーバーをリセットします。Aurora 64B/66B デザインには、クロック補正モジュールを必ず使用してください。

ホットプラグ ロジックを無効にする場合は、`<component name>_cbcc_gtx_6466.v` モジュールの `ENABLE_HOTPLUG` パラメーターを 0 に設定してください。ホットプラグ ロジックが無効の場合、デュプレックスでクロック補正文字を検索したり、受信データでシンプレックス RX 用の有効な BTF 文字を検索している場合に、コアが繰り返しリセットされることはありません。



**重要 :** リンクに予想可能な動作を求める場合には、ホットプラグ ロジックを有効にすることを強く推奨します。

次にホットプラグ シーケンスについて説明します。

1. 要件 : カードの置き換え、特定システムの電源切断、またはビット ファイルの再プログラミング前は、`reset` をアサートしてからホットプラグを実行する必要があります。これによって、リモート エージェントのチャンネルが確実に無効になり、接続を解除してプラグインしたときに確実に準備が整います。
2. 動作 : ホットプラグを実行する前に `reset` 信号が 128 サイクル以上アサートされると、十分な `NA_IDLE` が生成されるため、リモート リンクがエラーなしに `Channel Up` をディアサートできます。
3. 制限 : 前述のシーケンスに従わない場合は、`SOFT/DATA` エラーが生じ、リンクが適切に切断されない可能性があります。

## リトルエンディアン形式のサポート

Aurora 64B/66B IP コアは、デフォルトでビッグ エンディアン形式のユーザー インターフェイスをサポートしています。その他、リトルエンディアン形式もサポートしているため、AXI4-Stream 準拠の IP デザインヘシームレスに接続できます。リトルエンディアン形式を指定する場合は、Vivado IDE で [Little Endian Support] をオンにします。この設定は、ユーザー データ、UFC、NFC、およびユーザー K インターフェイスへ適用されます。ポートの変更については、該当するインターフェイスを参照してください。

## デザイン フローの手順

この章では、Aurora コアのカスタマイズと生成、制約、およびシミュレーション/合成/インプリメンテーション手順について説明します。一般的な IP インテグレーターの Vivado® デザイン フローについては、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) [参照 6]
- 『Vivado Design Suite ユーザー ガイド : IP を使用した設計』(UG896) [参照 7]
- 『Vivado Design Suite ユーザー ガイド : 入門』(UG910) [参照 8]
- 『Vivado Design Suite ユーザー ガイド : ロジック シミュレーション』(UG900) [参照 9]

Vivado IP インテグレーターでコア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) [参照 6] を参照してください。IP インテグレーターは、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を自動的に計算する場合があります。値が変更されるか否かを確認するには、この章のパラメーターの説明を参照してください。また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。

## コアのカスタマイズおよび生成

このセクションでは、LogiCORE™ IP Aurora 64B/66B コアの生成およびカスタマイズにあたっての Vivado Design Suite の使用方法について説明します。

**注記：**このコアには、IP インテグレーターの基本的なサポートが含まれますが、パラメーターの伝搬はサポートされていません。

### Vivado 統合設計環境 (IDE)

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 コア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 ([IP Catalog] → [Communication & Networking] → [Serial Interfaces] → [Aurora 64B66B])。
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 メニューで、[Customize IP] コマンドを選択します。

詳細は、『Vivado Design Suite ユーザー ガイド : Designing with IP』(UG896) [参照 7] および『Vivado Design Suite ユーザー ガイド : 入門』(UG910) [参照 8] を参照してください。

Aurora 64B/66B コアは、IP カタログを使用して、さまざまな要件に対応するようにカスタマイズできます。この章では、カスタマイズできるパラメーターについて説明し、また IP カタログのインターフェイスでこれらのパラメーターを指定する方法を説明します。

### IP カタログの使用

Vivado IP カタログで Aurora 64B/66B コアを選択すると、Aurora 64B/66B IP カタログが表示されます。79 ページの [図 4-1](#) および 80 ページの [図 4-2](#) にそれらの画面を示し、各セクションで詳しく説明します。

### IP カタログ

[図 4-1](#) および [図 4-2](#) に、IP カタログ画面を示しています。画面左側には、設定された Aurora 64B/66B コアのブロック図が表示されます。右側には、ユーザーが指定できるパラメーターが表示されます。カスタマイズ オプションの詳細については、この後のサブセクション (80 ページの「[Component Name]」) で説明します。

**注記：**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。

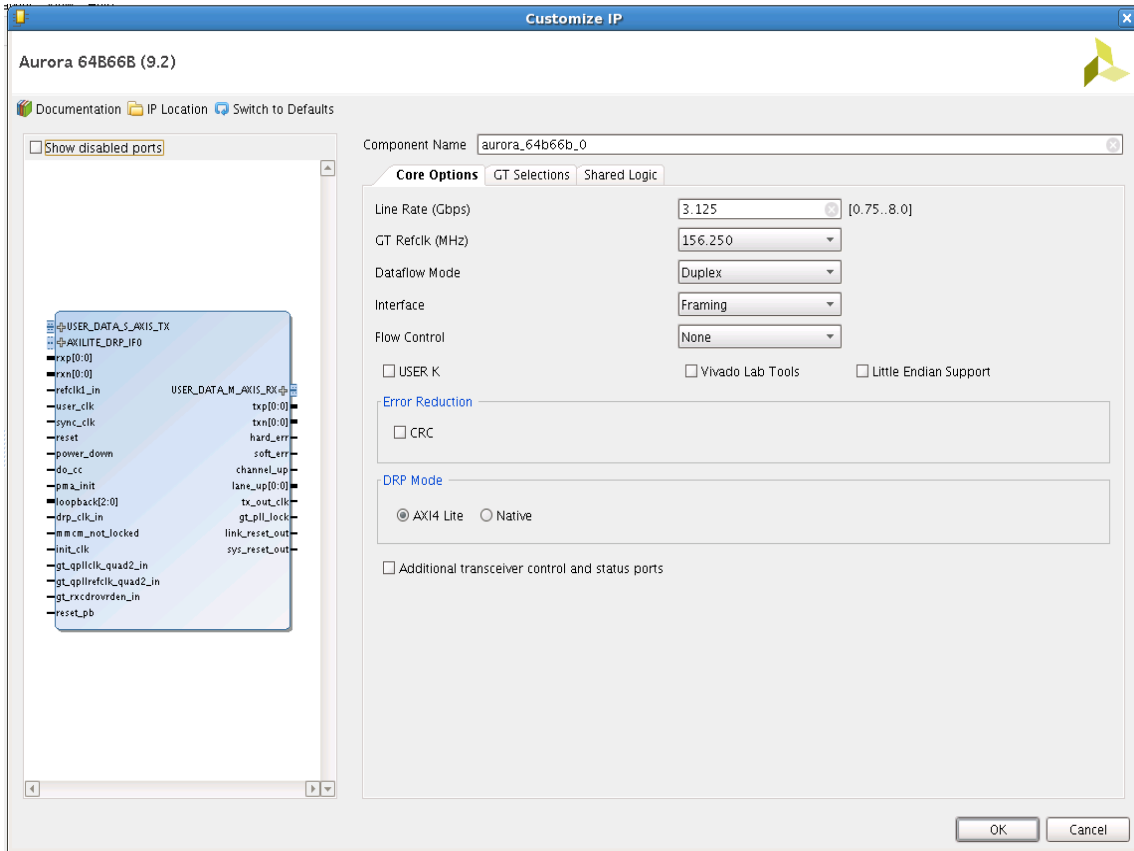


図 4-1 : Aurora 64B/66B IP カタログのページ 1 (7 シリーズ FPGA)

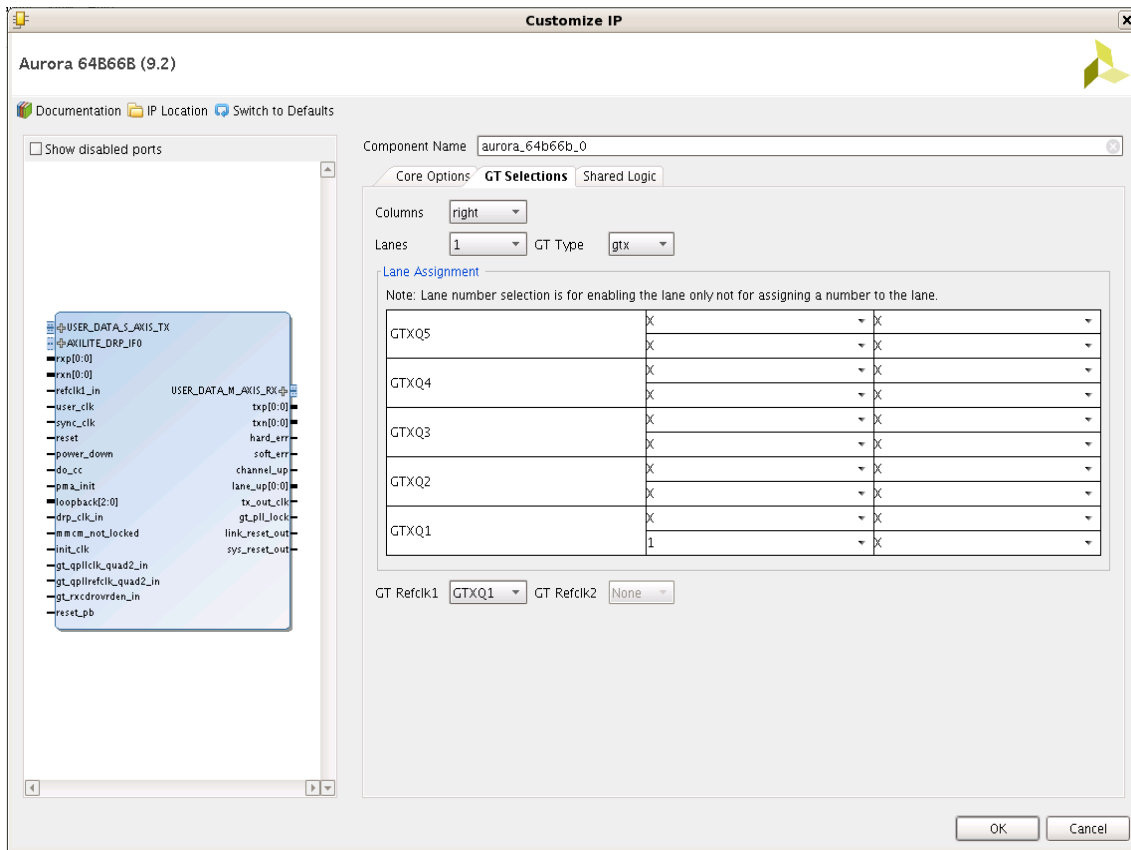


図 4-2 : Aurora 64B/66B IP カタログのページ 2 (7 シリーズ FPGA)

### [Component Name]

このテキスト ボックスには、コアの最上位の名前を入力します。規則外の名前が入力されると、修正されるまで赤色表示されます。生成されたコアのすべてのファイルは、この名前が付いたサブディレクトリに配置されます。コアの最上位モジュールにも、この名前が使用されます。

デフォルト : aurora\_64b66b\_0

### [Line Rate]

浮動小数点の値を入力します (Gb/s)。有効な範囲内の値を入力してください。これによって、シリアルリンクにデータが転送される際のエンコードされないビット レートが決定します。

デフォルト : GTX トランシーバーおよび Virtex®-7 FPGA GTH トランシーバーの場合は 3.125Gb/s

### [GT Refclk]

ドロップダウン リストから基準クロックの周波数を選択します。これらの基準クロック周波数はメガヘルツ (MHz) 単位で表示され、選択したライン レートによって異なります。最良の結果を得るには、ターゲット デバイスの基準クロック入力に実際に適用できる最大レートを選択します。

デフォルト：156.25MHz

### [Dataflow Mode]

Aurora 64B/66B コアがサポートするチャネルの方向を選択します。シンプレックス Aurora 64B/66B コアには、相補関係にあるシンプレックス 64B/66B コアに接続する単方向のシングルシリアルポートがあります。RXのみシンプレックスまたはTXのみシンプレックスとして、2つのオプションがあります。これらのオプションによって、Aurora 64B/66B コアがサポートするチャネルの方向が選択されます。

デュプレックス - Aurora 64B/66B コアには、通信用に TX と、もう一方にそれに対応する RX があります。

デフォルト：Duplex

### [Interface]

コアに使用されるデータパス インターフェイスの種類を選択します。任意の長さのデータ フレームを送信できる AXI4-Stream インターフェイスを使用する場合は、[Framing] を選択します。データ valid 信号を使用して Aurora チャネルを介してデータを転送するシンプルなワード ベースのインターフェイスを使用する場合は、[Streaming] を選択します。

デフォルト：Framing

### [Flow Control]

必要なオプションを選択して、コアにフローの制御を追加します。ユーザー フロー制御 (UFC) の場合は、アプリケーションは Aurora チャネルを介して高優先順位の短いメッセージを互いに送信できます。ネイティブ フロー制御 (NFC) の場合は、フルデュプレックス レシーバーが送信されるデータのレートを調節できるようになります。[Immediate Mode] の場合は、データ フレームの途中にアイドル コードを挿入できますが、[Completion Mode] の場合は完了したデータ フレーム間にのみアイドル コードを挿入できます。

利用可能なオプションは次のとおりです。

- None
- UFC only
- Immediate Mode – NFC
- Completion Mode – NFC
- UFC + Immediate Mode – NFC
- UFC + Completion Mode – NFC

ストリーミング インターフェイスでは、Immediate Mode のみ有効です。フレーミング インターフェイスでは Immediate Mode と Completion Mode の両方が有効です。

デフォルト：None

### [USER K]

コアにユーザー K インターフェイスを追加する場合は、このオプションをオンにします。ユーザー K ブロックは、ユーザー アプリケーションへ特殊なシングルブロック コードを直接渡します。これらのブロックは、アプリケーション固有の制御機能をインプリメントするために使用します。

デフォルト：未選択 (オフ)

#### [CRC]

データ ストリームに CRC32 を挿入する場合は、このオプションをオンにします。

デフォルト：未選択 (オフ)

#### [Little Endian Support]

すべてのインターフェイスをリトル エンディアン形式に変更する場合は、このオプションをオンにします。詳細は、[第3章の「リトルエンディアン形式のサポート」](#)を参照してください。デフォルトでは、ビッグ エンディアン形式を使用します。

デフォルト：未選択 (オフ)

#### [DRP Mode]

ダイナミック リコンフィギュレーション ポート (DRP) を使用してトランシーバーを制御およびモニターする場合は、必要なインターフェイスを選択します。

利用可能なオプションは次のとおりです。

- Native
- AXI4\_Lite

デフォルト：Native

#### [Columns]

ドロップダウン リストから適切な GT カラムを選択します。

デフォルト：left

#### [Lanes]

コアで使用されるレーン数 (GTX および GTH トランシーバー) を選択します。有効な範囲は、選択したターゲット デバイスによって異なります。

デフォルト：1

### [GT Type]

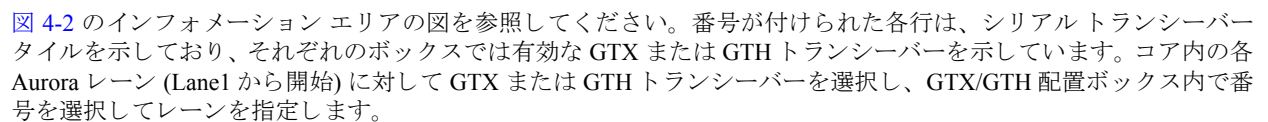
ドロップダウン リストから、シリアル トランシーバーの種類を選択します。このオプションは、Virtex-7 XT デバイスにのみ適用されます。その他のデバイスの場合、ドロップボックスは表示されません。

利用可能なオプションは次のとおりです。

- GTX
- V7GTH

デフォルト : gtx

### [Lane Assignment]

 **図 4-2** のインフォメーション エリアの図を参照してください。番号が付けられた各行は、シリアル トランシーバー タイルを示しており、それぞれのボックスでは有効な GTX または GTH トランシーバーを示しています。コア内の各 Aurora レーン (Lane1 から開始) に対して GTX または GTH トランシーバーを選択し、GTX/GTH 配置ボックス内で番号を選択してレーンを指定します。

- ドロップボックス メニューの「X」は、レーンが選択されていないことを意味します。
- ドロップボックス メニューの「1 - 16」は、特定レーンが選択されていることを意味します。物理的なレーンへその番号を割り当てるものではありません。



**推奨 :** 複数 GT デザインの場合は、連続する/物理的に隣接するレーンを選択してください。

**注記 :** Aurora コアは、あらかじめ定義された方法でトランシーバーを配置します。また、トランシーバーの配置制約 (LOC) を昇順に生成します。Vivado IDE でカーソルを移動して、7 シリーズおよび Zynq®-7000 ファミリーベースのデザインで選択されているトランシーバーを確認できます。レーン選択でどのように番号が入力されるかによって、トランシーバーの LOC やコアのインプリメンテーションが変更されることはありません。[Lane Assignment] は、UltraScale™ アーキテクチャベースのデザインにはありません。タイミング クロージャを達成するには、連続的なレーン選択にすることを強く推奨します。

### [GT Refclk1] および [GT Refclk2]

このセクションのドロップダウン リストから GTX または GTH トランシーバー タイルの基準クロック ソースを選択します。

デフォルト : GT REFCLK Source 1: GTXQn/ GTHQn、GT REFCLK Source 2 : None

**注記 :** n は、シリアル トランシーバー (GTX または GTH) の位置によって異なります。

### [Vivado Lab Tools]

Aurora 64B/66B コアに Vivado ラボ ツールを追加する場合は、このオプションをオンにします (75 ページの「Vivado ラボ ツールの使用」参照)。このオプションを選択することでデバッグ インターフェイスが提供され、コアのステータス信号が表示されます。

デフォルト : 未選択 (オフ)

[Shared Logic]

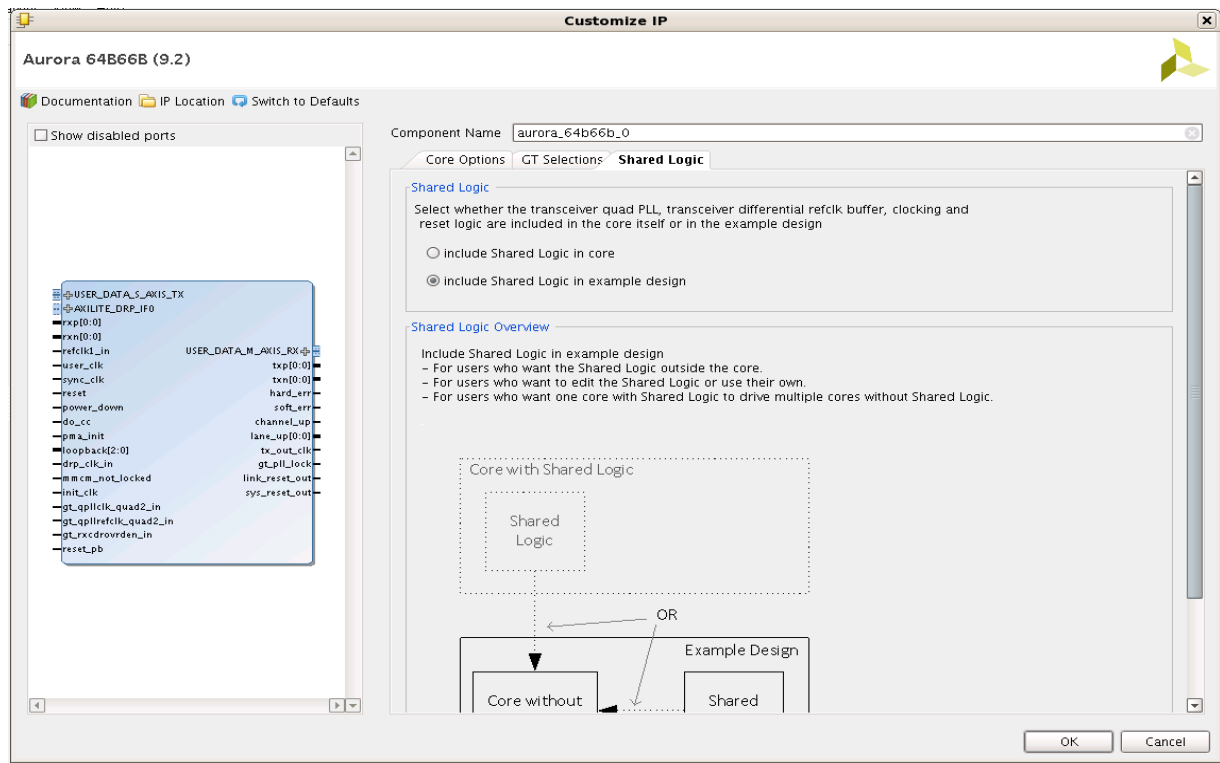


図 4-3：7 シリーズ FPGA の共有ロジック

トランシーバー共有 PLL ブロックとそのロジックを IP コアまたはサンプル デザインに含める場合、このオプションを選択します。

使用可能なオプション：

- [Include Shared Logic in Core]
- [include shared logic in example design]

デフォルト：[include shared logic in example design]

[Additional transceiver control and status ports]

コアの最上位にトランシーバーの制御ポートとステータス ポートを含める場合には、このオプションをオンにします。

デフォルト：未選択 (オフ)

[OK]

[OK] をクリックしてコアを生成します (96 ページの「コアの生成」参照)。Aurora 64B/66B コアのモジュールは、コアの最上位と同じ名前 IP カタログ ツールのプロジェクト ディレクトリに書き込まれます。

## ユーザー パラメーター

表 4-1 (7 シリーズ デバイス) および表 4-2 (UltraScale™ アーキテクチャ デバイス) では、Vivado IDE の GUI フィールドと XCI ファイルのユーザー パラメーター (Tcl コンソールに表示可能) の関係を示しています。バッチ駆動型の Tcl フローの場合は、表内の情報を使用して GUI パラメーターを設定し、Aurora 64B/66B コアを生成してください。

表 4-1：7 シリーズ<sup>(1)</sup> の GUI パラメーターとユーザー パラメーターの値

GUI パラメーター / 値	ユーザー パラメーター / 値	デフォルト値
[Core Option] タブ		
Line Rate (Gbps)	C_LINE_RATE	3.125
GT Refclk (MHz)	C_REFCLK_FREQUENCY	156.250
Dataflow Mode	Dataflow_Config	Duplex
Interface	Interface_Mode	Framing
Flow Control	Flow_Mode	None
User K	C_USER_K	false
Vivado Lab Tools	C_USE_CHIPSCOPE	false
Little Endian Support	C_USE_BYTESWAP	false
Error Reduction		
CRC	CRC_MODE	NONE
DRP Mode		
AXI4 Lite (デフォルト モード)	drp_mode	AXI4_LITE
Native		
Additional transceiver control and status ports	TransceiverControl	false
[GT Selection] タブ <sup>(2)</sup>		
Columns	C_COLUMN_USED	右 <sup>(3)</sup>
Lanes	C_AURORA_LANES	1
GT Type	C_GT_TYPE	gtx <sup>(4)</sup>
Lane Assignment <sup>(5)(6)</sup>		
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y4 を含める <sup>(7)</sup>	C_GT_LOC_5 <sup>(8)</sup>	1
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y5 を含める	C_GT_LOC_6	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y5 を含める	C_GT_LOC_7	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y7 を含める	C_GT_LOC_8	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y8 を含める	C_GT_LOC_9	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y9 を含める	C_GT_LOC_10	X

表 4-1: 7 シリーズ<sup>(4)</sup> の GUI パラメーターとユーザー パラメーターの値 (続き)

GUI パラメーター / 値	ユーザー パラメーター / 値	デフォルト値
Lane Assignment (続き)		
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y10 を含める	C_GT_LOC_11	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y11 を含める	C_GT_LOC_12	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y12 を含める	C_GT_LOC_13	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y13 を含める	C_GT_LOC_14	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y14 を含める	C_GT_LOC_15	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y15 を含める	C_GT_LOC_16	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y16 を含める	C_GT_LOC_17	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y17 を含める	C_GT_LOC_18	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y18 を含める	C_GT_LOC_19	X
トランシーバーを選択して、 デザインに GTXE2_CHANNEL_X1Y19 を含める	C_GT_LOC_20	X
GT Refclk (MHz)		
GT Refclk1	C_GT_CLOCK_1	GTXQ1
GT Refclk2	C_GT_CLOCK_2	None
[Shared Logic] タブ		
Include Shared Logic in core	SupportLevel <sup>(9)</sup>	0
Include Shared Logic in example design (デフォルト モード)		
注記： <ol style="list-style-type: none"> <li>この表の値は、デフォルト デバイス (xc7vx485tffg1157-1) の場合です。</li> <li>X0Y0 の GT 選択は、カラムに基づきます。</li> <li>両側に GT があるデバイスの場合、左側 (left) がデフォルト値です。</li> <li>GTX トランシーバーを備えるデバイスでは、gtx がデフォルト値です。GTH トランシーバーを備えるデバイスでは、v7gth がデフォルト値です。</li> <li>レーン番号の選択は、レーンを有効にするのみで、レーンに番号を割り当てるためではありません。</li> <li>レーンの選択は、7 シリーズ FPGA にのみ適用され、UltraScale デバイスには適用されません。</li> <li>デフォルト デバイスでは、GT は GTXE2_CHANNEL_X1Y4 から開始します。その他は、GTXE2_CHANNEL_X0Y0 から開始します。</li> <li>C_GT_LOC_i の i は 1 ~ 48 の範囲です。デフォルトでは、最も低い i の C_GT_LOC_i が指定されます。</li> <li>[Include Shared Logic in core] がオンの場合、SupportLevel は 1 となります。</li> </ol>		

表 4-2 : UltraScale の GUI パラメーターとユーザー パラメーターのマッピング

GUI パラメーター / 値	ユーザー パラメーター / 値	デフォルト値
[Core Options] タブ		
Physical Layer		
Line Rate (Gbps)	C_LINE_RATE	10.3125
Lanes	C_AURORA_LANES	1
GT Type	C_GT_TYPE	gth
GT Refclk (MHz)	C_REFCLK_FREQUENCY	156.250
Link Layer		
Dataflow Mode	Dataflow_Config	Duplex
Interface	Interface_Mode	Framing
Flow Control	Flow_Mode	None
User K	C_USER_K	false
CRC	CRC_MODE	NONE
Little Endian Support	C_USE_BYTESWAP	false
Debug and Control		
DRP Mode		
AXI4 Lite (デフォルト モード)	drp_mode	AXI4_LITE
Native		
Additional transceiver control and status ports	TransceiverControl	false
Vivado Lab Tools	C_USE_CHIPSCOPE	false
[Shared Logic] タブ		
Include Shared Logic in core	SupportLevel <sup>(1)</sup>	0
Include Shared Logic in example design (デフォルト モード)		
注記 :		
1. [Include Shared Logic in core] がオンの場合、SupportLevel は 1 となります。		

# UltraScale アーキテクチャ特有デザインのコアのカスタマイズオプション

このセクションでは、Vivado IDE で UltraScale アーキテクチャ特有デザインに適用するコアのカスタマイズ オプションについて説明します。

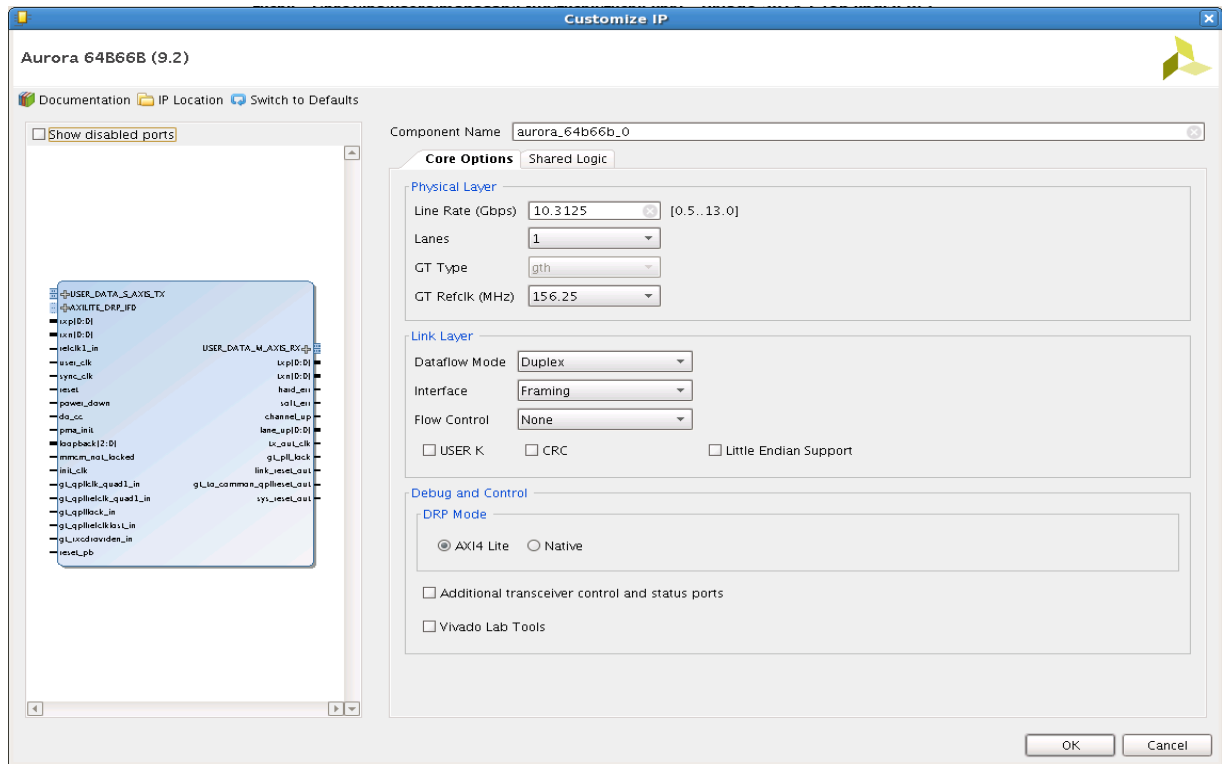


図 4-4 : Aurora 64B/66B IP カタログのページ 1 (UltraScale デバイス)

図 4-4 に、UltraScale デバイスをターゲットとしている Aurora 64B/66B コアの Vivado IDE を示しています。これは、[Customize IP] ウィンドウで GT コンフィギュレーションが設定されている状態を示しています。このコアは、0.5Gb/s ~ 13.0Gb/s のライン レートをサポートします。GT 用にコンフィギュレーション可能なパラメーターは、[Line Rate]、[Lanes]、および [GT Refclk] です。ライン レートの設定に基づき、GT の基準クロックの選択肢が自動的に変更されます。同様に、ライン レートに基づいて [GT Refclk] 値の範囲の設定が可能になります。ユーザー コンフィギュレーションに従って、パラメーター リストが XCI ファイルに生成されます。この XCI ファイルは、その後の Aurora 64B/66B および GT Wizard コンフィギュレーションのベースとして使用されます。

## UltraScale デバイスの GT インプリメンテーション

UltraScale デバイスの GT インプリメンテーションは、階層デザインフローとして知られているダイナミック コンフィギュレーションの呼び出しを通してサポートされています。詳細は、『7シリーズ GTZ トランシーバー ユーザーガイド』(UG478) [参照 10] を参照してください。GT 関連のユーザー コンフィギュレーションは、Vivado IDE で Aurora コアをコンフィギュレーションする際に渡されます。UltraScale デバイスの GT Wizard の使用については、『UltraScale FPGA トランシーバー ウィザード製品ガイド』(PG182) [参照 11] を参照してください。Aurora 64B/66B コア デザインの場合、UltraScale デバイスの GT Wizard は、サブ コアのリファレンス呼び出しを通じて参照されます。UltraScale アーキテクチャのアップデートに伴い、GT Wizard 独自のサブモジュール (セット コントローラーやデータ幅サイズ調整モジュールなど) が GT Wizard の中に含まれるように設計され、送信/受信ユーザー クロッキング モジュール ヘルパー コアは常に GT Wizard の外に配置されるように設計されます。GT コモンの場所は、レーン コンフィギュレーション速度やターゲット UltraScale デバイスに基づきます。8.0Gb/s より高速の場合、GT コモンは GT Wizard の外に配置されます。コアが非共有モードでコンフィギュレーションされている場合、GT コモンは Aurora 64B/66B サンプル デザインの一部になります。一方、Vivado IDE オプションを使用してコアが共有モードでコンフィギュレーションされている場合の GT コモンはコアの一部になります。Aurora 64B/66B コアは、ラインレート 0.5Gb/s ~ 8.0Gb/s の場合に CPLL を構成し、ラインレート 8.1Gb/s ~ 13.0Gb/s の場合に QPLL を構成します。

## コアに含まれる UltraScale デバイスの GT チャネル インスタンス

レーン速度、レーン数、基準クロック、CPLL/QPLL1 (ラインレートに依存) の選択、および GT の位置などの GT パラメーターは、階層 IP フローによって自動的に GT Wizard へ渡されます。これらのパラメーターに基づいて GT コンフィギュレーションが完了し、Aurora 64B/66B コアに GT インスタンスが生成されます。前述したとおり、GT Wizard にはリセット コントローラーおよびユーザー データ幅サイズ調整モジュールが含まれます。

## コアに含まれる UltraScale デバイスの GT クロッキングストラクチャ

コアのメイン クロッキング モジュールは、ユーザー クロック、同期クロック、および初期化クロックを生成します。同期クロックとユーザー クロックは、コア ロジックと同様に GT チャネル インターフェイスの基準クロックとなります。コアは、常に送信ユーザー クロッキング モジュールをインスタンス化しますが、受信クロッキング モジュールのインスタンス化に関しては、コアのコンフィギュレーションに基づきます。Aurora 64B/66B コアが共有モードでコンフィギュレーションされる場合、クロッキング モジュールはコアの一部となり、ポートは共有可能な出力ポートとして利用できます。非共有モードでコンフィギュレーションされる場合のクロッキング モジュールはサンプル デザインの一部となり、コアはこれらのポートを入力ポートとして備えます。

## コアに含まれる UltraScale デバイスの GT コモン インスタンス

UltraScale アーキテクチャの GT Wizard からの GT コモンは、Aurora 64B/66B コアの共有ロジックの一部となります。これは、8.0Gb/s を超えるレーン速度が選択された場合のみ該当します。コアは、レーン数に基づいて、GT コモン クワッドの数を自動的に挿入します。各 GT クワッドは、最大 4 つの GT チャネルへ基準クロックを提供します。コアは、GT コモン モジュールに対して、クロック、リセット、ロック信号用のインターフェイスを提供します。また、デフォルトの GT 位置は連続したものになります。

クロック モジュールが GT コモンへ基準クロックを提供し、GT コモンは各クワッドの各 GT チャネル用にクロック、基準クロック、クロック ロック、および基準クロックのロスト信号を提供します。

Aurora 64B/66B コアが共有モードでコンフィギュレーションされる場合、GT コモンはコアの一部となり、ポートは出力ポートとして利用可能になります。非共有モードでコンフィギュレーションされる場合 (8Gb/s より高速) の GT コモンは Aurora 64B/66B サンプル デザインの一部となり、コアはこれらのポートを入力ポートとして備えます。

8Gb/s 以下のレーン速度が選択される場合、GT コモンは GT Wizard IP コア内に属し、それらのポートは共有モードのときにのみコアの周辺で有効になります。非共有モードの場合、これらのポートはコア内部に含まれます。

注記：8Gb/s より低速の場合、GT コモン モジュールはコア (共有/非共有モード) の一部に含まれません。

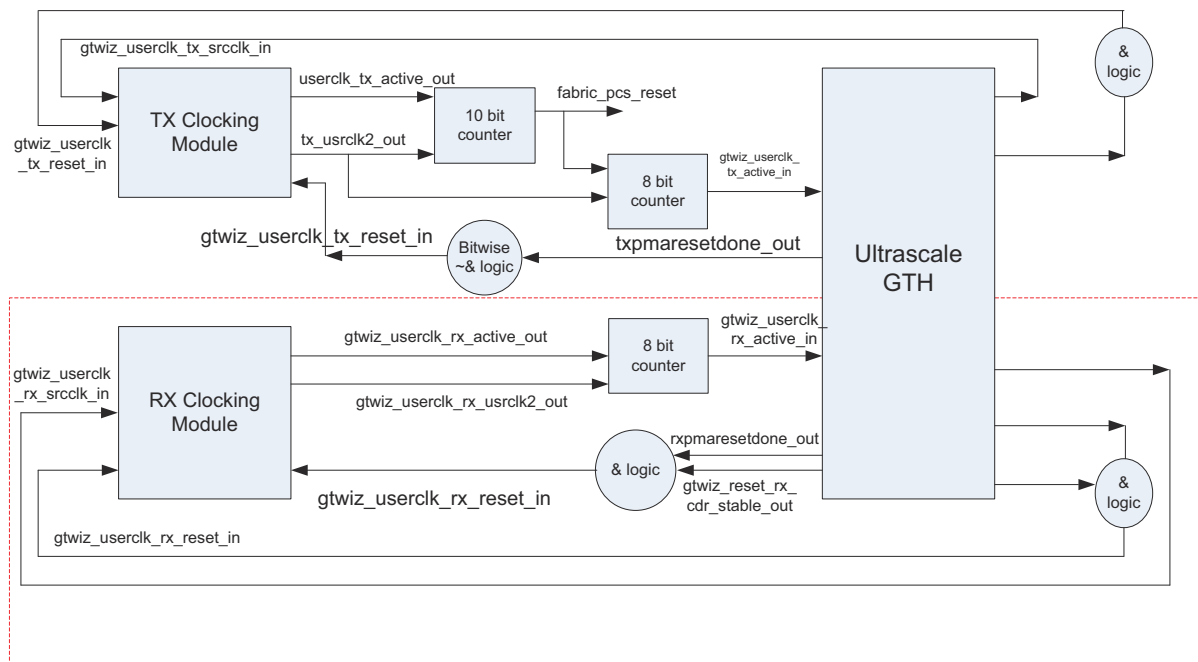


図 4-5: リセット シーケンス ロジックのインプリメンテーション (説明用に描写)

拡張されたリセット アクティブ信号によって、GT チャネルと GT コモン間の信頼性の高いリセット シーケンスが保証されます。

## GT チャネルの位置

Aurora 64B/66B コアの UltraScale アーキテクチャの GT インプリメンテーションでは、GT の位置が連続している必要があります。選択したレーン数およびターゲット UltraScale デバイスに基づいて、コアは連続する GT チャネルの位置を提供します (GT Wizard でデフォルト設定)。QPLL1 (ライン レートが 8.0Gb/s より高速) をベースとするデザインの場合、GT コモンはコアの共有/非共有ロジックの一部となります。GT コモンと GT チャネル間の接続は、レーン数によって異なります。



**推奨 :** デザイン生成後に絶対的に必要な場合でない限り、デフォルトの位置を変更しないでください。変更したデザインの機能は保証されません。8.0Gb/s より低速のライン レートを選択した場合、CPLL が GT Wizard 階層コアの一部となります。

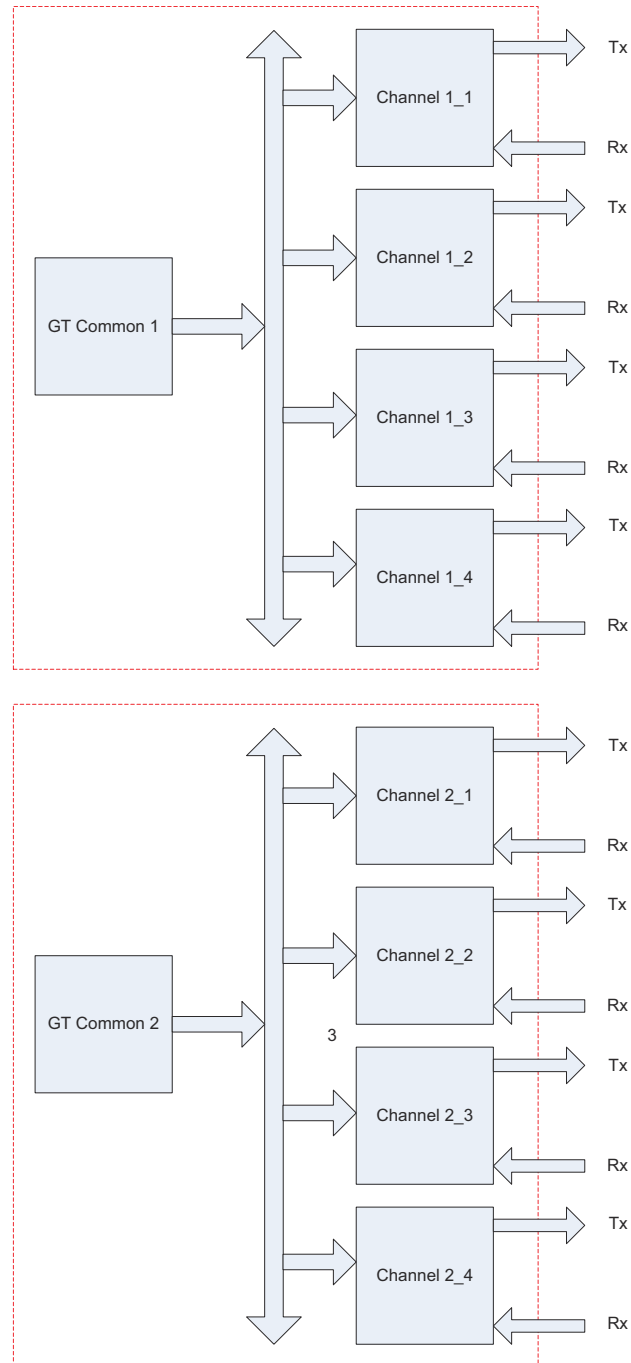


図 4-6 : UltraScale アーキテクチャの GT コモンと GT チャネルのインターフェイス  
(Aurora 64B/66B コンフィギュレーション : 9Gb/s、8 レーン)

## Aurora 64B/66B コアによる GT のマップ

Vivado IDE では、Aurora 64B/66B コアのラインレート、レーン数、およびデータフローモードなどを指定できます。これらの入力、階層的な IP 呼び出しメカニズムによって、GT Wizard へ渡されます。GT Wizard は、選択した UltraScale デバイスに対して、CPLL または QPLL ベースのデザインに応じて基準クロック範囲およびデバイスで有効なデフォルトの GT 位置など、適切な情報を提供します。図 4-6 を参照し、GT チャンネルがそれぞれどのように接続されているかを確認してください。これらのデフォルト位置は、UltraScale デバイスの GT Wizard インスタンスによって提供される XDC に指定されています。変更が必要な場合には、ここで位置を確認できます。これらの位置は変更しないことを推奨していますが、デザイン要件に応じて異なるチャンネル位置を選択できます。選択した GT チャンネル位置は連続する必要がある、最小限のクワッド数を使用するように割り当てます。たとえば、3 つのレーンを持つ 2 つの Aurora デザインをコンフィギュレーションする場合、これら 2 つの Aurora デザインは 2 つの異なるクワッドに配置される必要があります。各クワッドグループには、1 つの GT コモンと 4 つの GT チャンネルのほかロジックが含まれます。クワッド構造の詳細説明は、『UltraScale FPGA GTH トランシーバー ユーザーガイド (UG576) [参照 3]』を参照してください。UltraScale デバイス用の現在の Aurora 64B/66B コア インプリメンテーションでは、各 GT コモンが同じクワッドにある最大 4 つの GT チャンネルに、クロック信号、基準クロック、クロック ロック信号、および基準クロック ロスト信号を提供します。コアは、必要なチャンネル数に応じてクワッドを推論し、各クワッドに割り当てられた GT チャンネルへ適切なインターフェイスを提供します。

ラインレートが 0.5Gb/s ~ 8.0Gb/s の範囲で選択される CPLL ベースのインプリメンテーションの場合、CPLL は GT Wizard コアのインスタンス内に含まれ、デフォルトですべての内部接続が提供されます。

## 出力生成

カスタマイズされた Aurora 64B/66B コアは、Verilog の HDL ソース モジュールセットとして提供されます。これらのファイルは、あらかじめ定義されたディレクトリ構造の中に配置されます。プロジェクト ディレクトリ名は、このセクションで説明したとおりにプロジェクト作成時に IP カタログで入力します。

詳細は、『Vivado Design Suite ユーザーガイド：IP を使用した設計』(UG896) [参照 7] を参照してください。

## コアへの制約

ここでは、Vivado Design Suite でコアに制約を指定する方法について説明します。

## デバイス、パッケージ、スピード グレードの選択

該当なし

## クロック周波数

Aurora 64B/66B サンプル デザインのクロック制約は、次の 3 つのカテゴリに分類されます。

- GT 基準クロックの制約

Aurora 64B/66B コアは、デザインに最小値の基準クロックを 1 つと最大値の基準クロックを 2 つ使用します。GT 基準クロックの数は、トランシーバーの選択に基づいて決定されます (Vivado IDE の 2 ページ目にある [Lane Assignment])。Vivado IDE の 1 ページ目で選択した GT REFCLK 値を使用して、GT 基準クロックに制約を与えます。GT 基準クロックの制約には create\_clock XDC コマンドが使用されます。

- CORECLK クロックの制約

CORECLK は、コアの機能に基づくクロックです。USER\_CLK および SYNC\_CLK などの CORECLK は、適用された基準クロックと GT トランシーバーの分周値に基づいて、GT トランシーバーによって生成される TXOUTCLK から派生します。ラインレートと GT インターフェイス幅に基づいて、Aurora 64B/66B コアが

USER\_CLK/SYNC\_CLK 周波数を算出します。すべての CORECLK の制約には、create\_clock XDC コマンドが使用されます。

- INIT\_CLK の制約

Aurora 64B/66B サンプル デザインは、デバウンス回路を使用して、init\_clk クロックによって非同期でクロック供給される PMA\_INIT 信号をサンプルします。init\_clk クロックの制約には、create\_clock XDC コマンドが使用されます。



**推奨**：7 シリーズおよび Zynq デバイスの場合は、システム クロック周波数を GT 基準クロック周波数より低く、また 50 ~ 200MHz の範囲に設定することを推奨しています。UltraScale デバイスの場合、推奨範囲は 6.25MHz ~ line\_rate/64 または 200MHz のいずれか低い方) となります。

## 注意事項

- 7 シリーズ FPGA の場合、デフォルトの init\_clk 周波数はコアによって 50MHz に設定されています。XDC ファイル内のシステムおよび <user\_component\_name>\_core.v ファイルの STABLE\_CLOCK\_PERIOD に対して、この値を変更してください。
- CPLL を使用する UltraScale アーキテクチャ デザインで、init\_clk 周波数が line\_rate/64 以外の場合は、C\_FREERUN\_FREQUENCY パラメーター値を ip フォルダ内の <user\_component\_name>\_gt/synth/<user\_component\_name>\_gt.v ファイルにある周波数値に変更してください。

## フォルス パス

フォルス パス制約は、CDC モジュールの最初のステージのフリップフロップに定義されます。

## サンプル デザイン

生成されたサンプル デザインとサポート ロジックは、ライン レートが 10.3125Gb/s で基準クロックが 156.25MHz です。KC724 ボードに搭載された XC7K325T-FFG900-2 デバイス用に生成された XDC ファイルは、次のとおりです。

```
<user_component_name>_exdes.xdc

##### CLOCK CONSTRAINTS #####
##User Clock Constraint: the value is selected based on the line rate of the module
create_clock -name TS_user_clk_i -period 6.206 [get_pins
<user_component_name>_block_i/clock_module_i/user_clk_net_i/O]

##SYNC Clock Constraint
create_clock -name TS_sync_clk_i -period 3.103 [get_pins
<user_component_name>_block_i/clock_module_i/sync_clock_net_i/O]

##Reference clock constraint for GTX
create_clock -name GTXQ0_left_i -period 6.400 [get_ports GTXQ0_P]
create_clock -name GTXQ0_left_i -period 6.400 [get_ports GTXQ0_N]

##INIT_CLK board Clock Constraint
create_clock -name TS_INIT_CLK -period 20 [get_ports INIT_CLK_P]
create_clock -name TS_INIT_CLK -period 20 [get_ports INIT_CLK_N]

##False path constraint to the first D input pin of the synchronizer stages
set_false_path -to [get_pins -hier *<user_component_name>_cdc_to*/D]

##PIN LOCATION CONSTRAINTS
set_property LOC C25 [get_ports INIT_CLK_P]
```

```

set_property LOC B25 [get_ports INIT_CLK_N]
set_property LOC G19 [get_ports RESET]
set_property LOC K18 [get_ports PMA_INIT]
set_property LOC A20 [get_ports CHANNEL_UP]
set_property LOC A17 [get_ports LANE_UP]

##### GT CLOCK Locations #####
##Differential SMA Clock Connection
set_property LOC R8 [get_ports GTXQ0_P]
set_property LOC R7 [get_ports GTXQ0_N]

set_property LOC GTXE2_CHANNEL_X0Y0 [get_cells
<user_component_name>_block_i/<user_component_name>_i/inst/<user_component_name>_wr
apper_i/<user_component_name>_multi_gt_i/<user_component_name>_GTX_INST/gtxe2_i]
  
```

前述のサンプル XDC は、参照用としてのみご利用いただけます。この XDC は、Vivado デザイン ツールでコアが生成される際に自動的に生成されます。

## クロック管理

該当なし

## クロック配置

該当なし

## バンキング

該当なし

## トランシーバーの配置

GT トランシーバーの配置制約には、`set_property` XDC コマンドが使用されます。Vivado IDE の 2 つ目のページにツールチップとして表示されます。参照用にサンプル XDC が提供されています。

## I/O 規格および配置

正側の差動クロック入力ピン (末尾に `_P` が付く) と負側の差動クロック入力ピン (末尾に `_N` が付く) が GT 基準クロックとして使用されます。GT 基準クロック ピンの制約には、`set_property` XDC コマンドが使用されます。

---

## シミュレーション

このセクションでは、Vivado Design Suite 環境でのシミュレーションについて説明します。詳細は、『Vivado Design Suite ユーザー ガイド：ロジック シミュレーション』(UG900) [参照 9] を参照してください。

Aurora IP コアは、サンプル デザイン用のデモ テストベンチを提供します。シミュレーションのステータスは、メッセージでレポートされます。「TEST COMPLETED SUCCESSFULLY」というメッセージは、サンプル デザインのシミュレーションが完了したことを示します。

注記:「Reached max. simulation time limit」というメッセージは、シミュレーションが正常に完了しなかったことを意味します。詳細は、付録 C 「デバッグ」を参照してください。

デュプレックス コアのシミュレーションは、サンプル デザイン 生成後にシングル ステップで実行できます。シングル コアのシミュレーションには、パートナー コアの生成が必要です。パートナー コアは自動生成され、[Open IP Example Design] をクリックすると、シミュレーション ファイル セットの下に合成済み ネットリストが生成されます。シングル コアのサンプル デザインを開く場合、パートナー コアを合成する必要があるため、デュプレックス サンプル デザインの生成よりも多少時間がかかります。

シミュレーションの高速化：

C\_EXAMPLE\_SIMULATION パラメーターは、合成/インプリメンテーション後のネットリストの論理シミュレーションを高速化するために使用されます。

1. バッチ モードでコア生成を生成する場合、`set c_example_simulation true` コマンドをコア生成の一部に含めます。
2. Tcl コマンドを実行してシミュレーションを高速化します。前述のコマンドで生成されたコアは、シミュレーション専用です。
3. Vivado IDE でコアを生成する場合、これらのファイル (<USER\_COMPONENT\_NAME>\_exdes.v および <USER\_COMPONENT\_NAME>\_core.v) に生成された RTL で EXAMPLE\_SIMULATION パラメーターを 0 に変更して、シミュレーションを高速化します。

---

## 合成およびインプリメンテーション

このセクションでは、Vivado® Design Suite 環境での合成およびインプリメンテーションについて説明します。

合成と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) [参照 7] を参照してください。

### インプリメンテーション

クイックスタート サンプルには、次のコンポーネントが含まれます。

#### 概要

- デフォルト パラメーターを使用して生成された Aurora 64B/66B コアのインスタンス
  - 単一 GTX または GTH トランシーバーを使用するフルデュプレックス
  - AXI4-Stream インターフェイス
- サンプル デザインの 2 つのインスタンスをシミュレーションするためのデモ用テストベンチ

Aurora 64B/66B サンプル デザインは、合成は Vivado Design Suite で検証され、シミュレーションは Mentor Graphics Questa® で検証されています。

## コアの生成

Vivado デザイン ツールを使用して、デフォルト値で Aurora 64B/66B コアを生成する場合の手順は次のとおりです。

1. 作業ディレクトリから Vivado デザイン ツールを起動します。Vivado デザイン ツールの使用方法については、『Vivado Design Suite ユーザー ガイド：IP を使用した設計』(UG896) [参照 7] を参照してください。
2. [Create New Project] → [New Project] ページで [Next] をクリックします。
3. 新しいプロジェクト名とプロジェクトの場所を入力します。
4. [Project Type] に [RTL Project] を選択して、[Next] をクリックします。
5. デバイスには、[xc7vx485tffg1157-1] を選択します。
6. プロジェクト作成後、[Project Manager] パネルで [IP catalog] をクリックします。
7. /Communication\_&\_Networking/Serial\_Interfaces の下にある IP カタログで Aurora 64B/66B v9.2 コアを指定します。
8. コアをダブルクリックします。
9. [OK] をクリックします。

## サンプル デザインの実装

サンプル デザインは、IP コアから生成する必要があります。

1. 生成された IP を右クリックします。右クリックで表示されたメニューから [Open Example Design] をクリックします。これで、生成した IP コア用のサンプル デザインが開きます。
2. [Run Implementation] をクリックして、合成とインプリメンテーションを実行します。その他、[Generate Bitstream] をクリックして、ビットストリームを生成できます。

**注記：**XDC でデザインのすべての入力および出力ポートに LOC および IO 規格を指定する必要があります。

合成と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) [参照 7] を参照してください。

# サンプル デザインの詳細

この章では、Vivado® Design Suite 環境で提供されているサンプル デザインについて説明します。

---

## ディレクトリとファイルの内容

サンプル デザインのディレクトリ構造およびファイル内容の詳細は、[92 ページの「出力生成」](#)を参照してください。

---

## サンプル デザインのクイック スタート

このクイック スタート ガイドでは、Aurora 64B/66B コアの生成、サンプル デザインを使用したハードウェアへのコアの実装、そしてデモ用テスト ベンチ (demo\_tb) でのコアのシミュレーションの手順を追って説明します。Aurora 64B/66B コアと共に提供されるサンプル デザインの詳細は、「[サンプル デザインの詳細](#)」を参照してください。

クイックスタート サンプル デザインには、次のコンポーネントが含まれます。

- デフォルト パラメーターを使用して生成された Aurora 64B/66B コアのインスタンス
  - 単一 GTX トランシーバーを使用するフルデュプレックス
  - AXI4-Stream ユーザー インターフェイス
- シンプルなデータ転送動作にコアをコンフィギュレーションするための XDC ファイル付き最上位サンプル デザイン (<component name>\_exdes)
- サンプル デザインの 2 つのインスタンスをシミュレーションするためのデモ用テストベンチ

## サンプル デザインの詳細

各 Aurora 64B/66B コアには、シンプルなデータ転送システムを使用するサンプル デザイン (<component name>\_exdes) が含まれています。example\_design ディレクトリの詳細は、第 4 章の「出力生成」を参照してください。

サンプル デザインは、主に 2 つのコンポーネントで構成されています。

- TX インターフェイスへ接続されるフレーム ジェネレーター (「FRAME\_GEN」)
- RX ユーザー インターフェイスへ接続されるフレーム チェッカー (「FRAME\_CHECK」)

図 5-1 に、フルデュプレックス コアのサンプル デザイン ブロック図を示します。98 ページの表 5-1 は、サンプル デザインのポートについて説明しています。

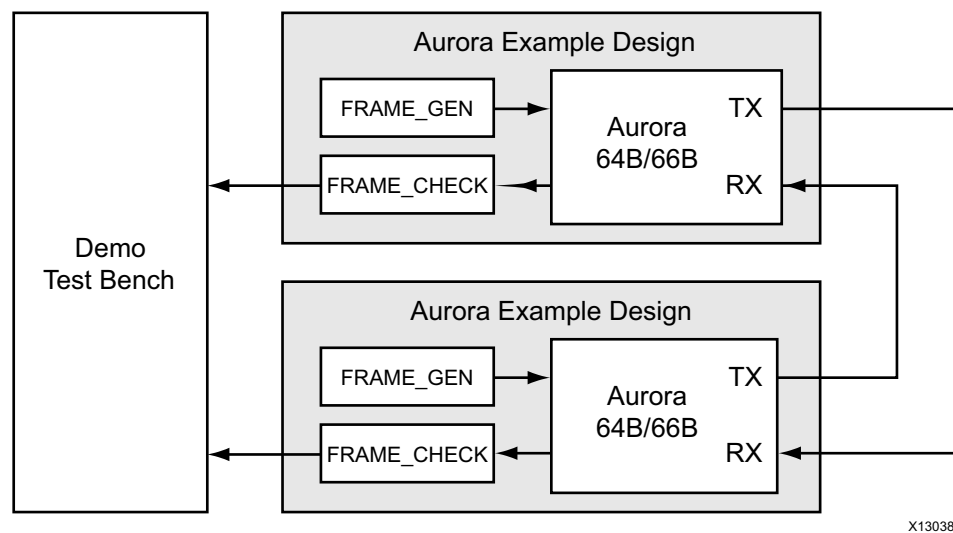


図 5-1 : サンプル デザイン

サンプル デザインは、コアのすべてのインターフェイスを使用します。オプションのフロー制御用に個別の AXI4-Stream インターフェイスがあります。TX または RX インターフェイスのないシンプレックス コアには、FRAME\_GEN または FRAME\_CHECK ブロックがありません。フレーム ジェネレーターが、ストリーミング/フレーミング インターフェイスを使用してコアにランダムなデータ ストリームを生成します。

また、サンプル デザインを参照用として利用し、クロッキング インターフェイスなど Aurora 64B/66B コアの難しいインターフェイス接続を容易に行うことができます。

ボード上でサンプル デザインを使用する場合は、example\_design サブディレクトリの <component name>\_exdes ファイルで適切なピンの割り当てやクロック制約を作成/変更する必要があります。表 5-1 は、サンプル デザインのポートについて説明しています。

表 5-1 : サンプル デザインの I/O ポート

ポート	方向	説明
rxn[0:m-1]	入力	差動シリアル データ入力ピンの負側です。
rxp[0:m-1]	入力	差動シリアル データ入力ピンの正側です。
txn[0:m-1]	出力	差動シリアル データ出力ピンの負側です。
txp[0:m-1]	出力	差動シリアル データ出力ピンの正側です。

表 5-1: サンプル デザインの I/O ポート (続き)

ポート	方向	説明
reset	入力	サンプル デザインのリセット信号です。このアクティブ High のリセット信号は、基準クロック入力から生成される user_clk 信号を使用してデバウンス処理されます。
<reference clock(s)>	入力	Aurora 64B/66B コアの基準クロックは、サンプル デザインの最上位に配線されています。基準クロックの詳細は、第 3 章の「クロック インターフェイスおよびクロッキング」を参照してください。
<core error signals>	出力	Aurora 64B/66B コアの Status および Control インターフェイスからのエラー信号は、サンプル デザインの最上位に現れ、レジスタに格納されます。詳細は、第 2 章の「ステータス、制御、およびトランシーバー インターフェイス」を参照してください。
<core channel up signals>	出力	コアのチャンネルアップステータス信号は、サンプル デザインの最上位に現れ、レジスタに格納されます。詳細は、第 2 章の「ステータス、制御、およびトランシーバー インターフェイス」を参照してください。
<core lane up signals>	出力	コアのレーンアップステータス信号は、サンプル デザインの最上位に現れ、レジスタに格納されます。コアには、使用する各 GTX/GTH トランシーバーに 1 つのレーンアップ信号があります。詳細は、第 2 章の「ステータス、制御、およびトランシーバー インターフェイス」を参照してください。
pma_init	入力	GTX/GTH トランシーバーの PCS および PMA モジュール用リセット信号は、デバウンス回路を介して最上位レベルへ接続されます。信号は、init_clk を使用してデバウンス処理されます。GT RESET の詳細は、『7 シリーズ FPGA GTX/GTH トランシーバー ユーザー ガイド』(UG476) [参照 4] を参照してください。
init_clk_p/ init_clk_n	入力	init_clk 信号は、PMA_INIT 信号のレジスタへの格納およびデバウンス処理に使用されます。この信号は、GTX/GTH トランシーバーを介さずに、低速レートに設定する必要があります。基準クロックよりも低速にすることを推奨します。UltraScale™ デバイスの場合、init_clk 信号はシングルエンドです。
data_err_count[0:7]	出力	FRAME_CHECK が受信した想定値と異なるフレーム データ ワード数を示します。
ufc_err	出力	FRAME_CHECK が想定値と異なる UFC データ ワードを受信するとアサート (アクティブ High) されます。
user_k_err	出力	FRAME_CHECK が想定値と異なるユーザー K データ ワードを受信するとアサート (アクティブ High) されます。

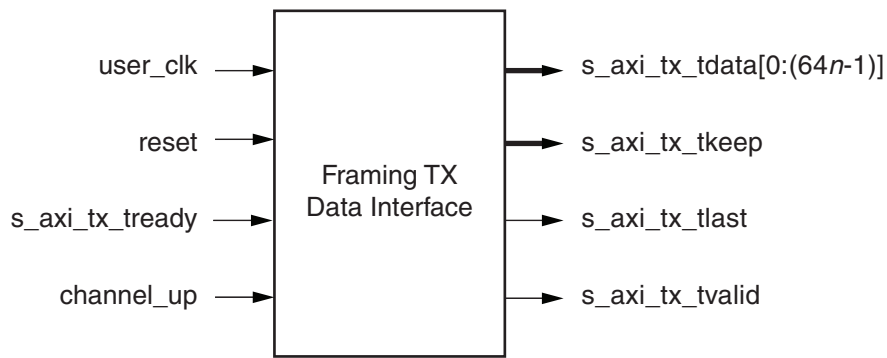
## FRAME\_GEN

### フレーミング TX データ インターフェイス

ユーザー データを送信する場合、FRAME\_GEN ユーザー データ ステート マシンが制御信号を操作して、次を実行します。

- Aurora インターフェイスが RESET から回復して CHANNEL\_UP ステートに到達すると、ユーザー データ リニア フィードバック シフト レジスタ (LFSR) を使用して疑似ランダムデータが生成され、s\_axi\_tx\_tdata バスへ接続されます。
- 2 つのカウンターに基づいて、現フレームに対して s\_axi\_tx\_tlast を生成します。8 ビット カウンターを 1 つ使用してフレーム サイズを決定し、また別の 8 ビット カウンターを使用して、送信されたユーザー データ バイトの数を追跡します。フレーム サイズ カウンターは初期化され、フレームごとに 1 つインクリメントされます。
- s\_axi\_tx\_tkeep バスはユーザー データ LFSR の下位ビットへ接続され、SEP および SEP7 条件を生成します。
- AXI4-Stream プロトコル仕様に従って、s\_axi\_tx\_tvalid 信号がアサートされます。
- ユーザー データ ステート マシンのステート遷移は、Aurora AXI4-Stream インターフェイスの s\_axi\_tx\_tready 信号で制御されます。
- 単一サイクルフレームなど多様なフレーム トラフィックが生成されます。

図 5-2 に、Aurora 64B/66B コアの FRAME\_GEN フレーミング ユーザー インターフェイスと TX データ用の AXI4-Stream に準拠するポートを示します。



UG775\_c10\_02\_050211

図 5-2 : Aurora 64B/66B コアのフレーミング TX データ インターフェイス (FRAME\_GEN)

表 5-2 では、FRAME\_GEN フレーミング TX データ ポートとそれらの説明を示しています。

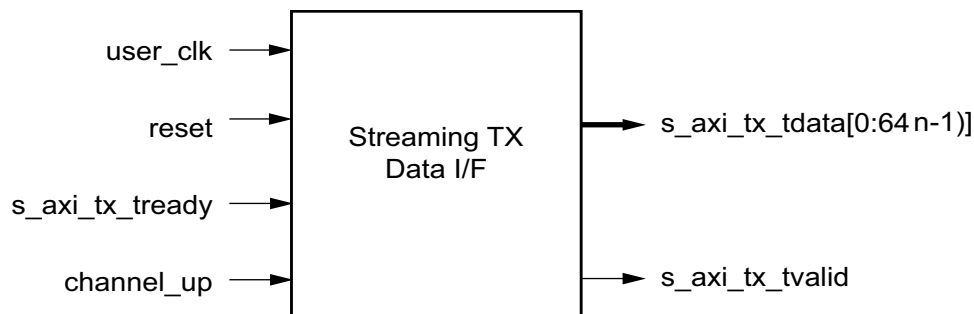
表 5-2 : FRAME\_GEN フレーミング ユーザー I/O ポート (TX)

名前	方向	説明
s_axi_tx_tdata[0:(64n-1)]	出力	ユーザー フレーム データです。幅は 64*n (n はレーン数を表す) です。
s_axi_tx_tkeep[0:n-1]	出力	最後のデータ ビートで有効なバイト数を示します (s_axi_tx_tlast が High にアサートされている場合のみ有効)。
s_axi_tx_tvalid	出力	ソースからの AXI4-Stream 信号が有効な場合にアサート (High) されます。ソースからの AXI4-Stream 制御信号またはデータが無視される場合にはディアサート (Low) されます。
s_axi_tx_tlast	出力	フレーム データの終わりを示します (アクティブ High)。
s_axi_tx_tready	入力	ソースからの信号が受信されると (s_axi_tx_tvalid もアサートされている場合)、クロック エッジでアサート (High) されます。ソースからの信号が無視される場合には、クロック エッジでディアサート (Low) されます。
channel_up	入力	Aurora チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
user_clk	入力	Aurora 64B/66B コアとユーザー アプリケーションで共有されるパラレル クロックです。
reset	入力	Aurora コアをリセットします (アクティブ High)。

## ストリーミング TX データ インターフェイス

ストリーミング TX データ インターフェイスは、フレーミング TX データ インターフェイスと同じですが、フレーム区切り文字、s\_axi\_tx\_tlast、および s\_axi\_tx\_tkeep がありません。ユーザー データを送信する場合、FRAME\_GEN ユーザー データ ステート マシンが制御信号を操作して次を実行します。

- Aurora インターフェイスが RESET から回復して CHANNEL\_UP ステートに到達すると、ユーザー データ リニア フィードバック シフト レジスタ (LFSR) を使用して疑似ランダムデータが生成され、s\_axi\_tx\_tdata パスへ接続されます。
- LFSR は、s\_axi\_tx\_tready がアサートされるたびに新しいデータを生成します。
- s\_axi\_tx\_tvalid 信号は常にアサートされます。



X13022

図 5-3 : Aurora 64B/66B コアのストリーミング TX データ インターフェイス (FRAME\_GEN)

表 5-3 では、FRAME\_GEN ストリーミング TX データ ポートとそれらの説明を示しています。

表 5-3 : FRAME\_GEN ストリーミング ユーザー I/O ポート (TX)

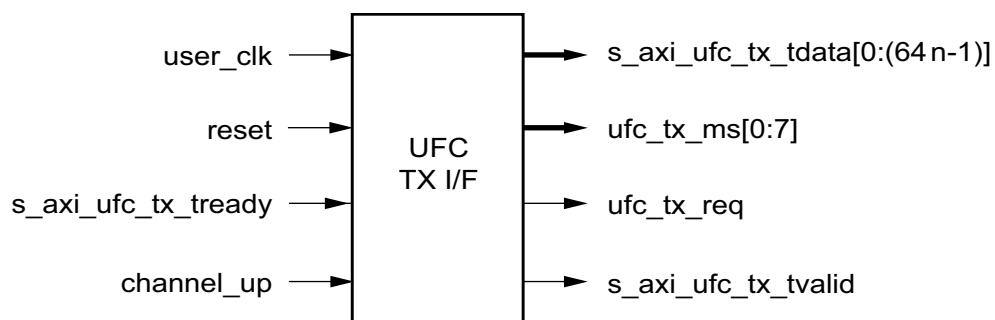
名前	方向	説明
s_axi_tx_tdata[0:(64n-1)]	出力	送信されるフレーム データです。幅は 64*n (n はレーン数を表す) です。
s_axi_tx_tvalid	出力	ソースからの AXI4-Stream 信号が有効な場合にアサート (High) されます。ソースからの AXI4-Stream 制御信号またはデータが無視される場合にはディアサート (Low) されます。
s_axi_tx_tready	入力	ソースからの信号が受信されると (s_axi_tx_tvalid もアサートされている場合)、クロック エッジでアサート (High) されます。ソースからの信号が無視される場合には、クロック エッジでディアサート (Low) されます。
channel_up	入力	Aurora チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
user_clk	入力	Aurora 64B/66B コアとユーザー アプリケーションで共有されるパラレル クロックです。
reset	入力	Aurora コアをリセットします (アクティブ High)。

## UFC TX インターフェイス

UFC データを送信する場合、FRAME\_GEN UFC ステート マシンが制御信号を操作して次を実行します。

- Aurora TX インターフェイスで CHANNEL\_UP がアサートされると、ufc\_tx\_req をアサートします。
- ufc\_tx\_req と共に ufc\_tx\_ms も送信されます。ufc\_tx\_ms 信号は、最初の UFC フレームには 0 を送信し、次の UFC フレームから 255 (最大値) まで 1 つずつインクリメントされます。
- ufc\_tx\_req が配置された後に s\_axi\_ufc\_tx\_tvalid 信号がアサートされます。
- Aurora TX インターフェイスから s\_axi\_ufc\_tx\_tready を受信すると、s\_axi\_ufc\_tx\_tdata 信号が送信されます。
- UFC フレーム送信の周波数は、UFC\_IFG パラメーターで指定されます。

図 5-4 に、Aurora 64B/66B コアの FRAME\_GEN UFC TX インターフェイスと UFC TX データ用の AXI4-Stream に準拠するポートを示します。



X13027

図 5-4 : Aurora 64B/66B コアの UFC TX インターフェイス (FRAME\_GEN)

表 5-4 では、FRAME\_GEN UFC TX データ ポートとそれらの説明を示しています。

表 5-4 : FRAME\_GEN UFC ユーザー I/O ポート (TX)

名前	方向	説明
ufc_tx_req	出力	チャンネル パートナーへの UFC メッセージ送信が要求されると、アサート (アクティブ High) されます。別の UFC メッセージが進行中で、最後のサイクルの途中でない限り、1 サイクル後に要求が処理されます。要求後、優先順位の高いイベントによって割り込まれない限り、2 サイクル以内に s_axi_ufc_tx_tdata バスはデータ送信可能な状態となります。
ufc_tx_ms[0:7]	出力	UFC メッセージ内のバイト数を指定します (メッセージ サイズ)。最大の UFC メッセージ サイズは 256 です。ufc_tx_ms に指定する値は、転送される実際のバイト数より 1 つ少なくなります。たとえば、この値が 3 の場合、実際には 4 バイトのデータが送信されます。
s_axi_ufc_tx_tdata [0:(64n-1)]	出力	Aurora チャンネルへ送信する UFC メッセージの出力バスです。s_axi_ufc_tx_tvalid および s_axi_ufc_tx_tready の両方が user_clk の立ち上がりエッジでアサートされる場合のみ、データがバスから読み出されてチャンネルへ送信されます。メッセージ内のバイト数がバスのバイトの整数倍でない場合、最後のサイクルで、バスの左から開始するメッセージの終了に必要なバイトのみ使用されます。
s_axi_ufc_tx_tvalid	出力	s_axi_ufc_tx_tdata 上のデータが有効の場合にアサートされず (アクティブ High)。s_axi_ufc_tx_tready がアサートされている間にこの信号がディアサートされると、UFC メッセージにアイドルブロックが挿入されます。
s_axi_ufc_tx_tready	入力	64B/66B コアが s_axi_ufc_tx_tdata インターフェイスからデータを読み出す準備が整うと High にアサートされます。この信号は、その他に優先順位の高い要求が進行中でない場合、ufc_tx_req がアサートされてから 1 クロック後にアサートされます。コアが直近に要求された UFC メッセージのデータを待機する間、s_axi_ufc_tx_tready はアサートを維持します。CC および NFC 要求は優先順位が高いため、これらが進行中の場合、この信号はディアサートされます。s_axi_ufc_tx_tready がアサートされている間、s_axi_tx_tready はディアサートされます。
channel_up	入力	Aurora チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
user_clk	入力	Aurora 64B/66B コアとユーザー アプリケーションで共有されるパラレル クロックです。
reset	入力	Aurora コアをリセットします (アクティブ High)。

## NFC TX インターフェイス

NFC フレームを送信する場合、FRAME\_GEN NFC ステート マシンが制御信号を操作して、次を実行します。

- NFC ステート マシンは、TX ユーザー データが送信されるまで待機し、その後 NFC XON モードに遷移します。
- s\_axi\_nfc\_tx\_tdata 値は、s\_axi\_nfc\_tx\_tvalid 信号と共に送信されます。
- あらかじめ指定した時間を経過すると、NFC ステート マシンは NFC XOFF モードに切り替わります。
- NFC ステート 遷移は、s\_axi\_nfc\_tx\_tready によって制御されます。
- UFC フレーム送信の周波数は、NFC\_IFG パラメーターで指定されます。

図 5-5 に、Aurora 64B/66B コアの FRAME\_GEN NFC TX インターフェイスと NFC TX データ用の AXI4-Stream に準拠するポートを示します。

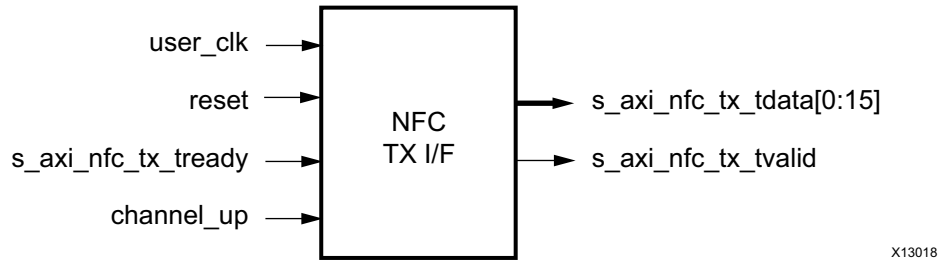


図 5-5 : Aurora 64B/66B コアの NFC TX インターフェイス (FRAME\_GEN)

表 5-5 では、FRAME\_GEN NFC TX データ ポートとそれらの説明を示しています。

表 5-5 : FRAME\_GEN NFC ユーザー I/O ポート (TX)

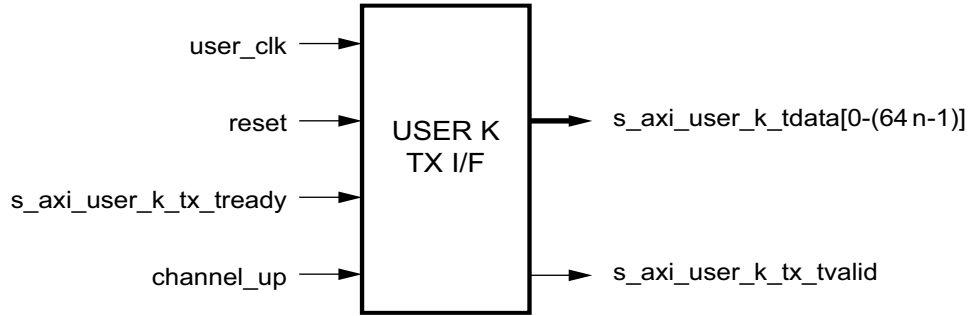
名前	方向	説明
s_axi_nfc_tx_tvalid	出力	チャンネルパートナーへの NFC メッセージ送信が要求されると、アサート (アクティブ High) されます。s_axi_nfc_tx_tready がアサートされるまで High を保持する必要があります。
s_axi_nfc_tx_tdata [0:15]	出力	NFC メッセージを受信したときにデータを送信できるようになるまで、チャンネルパートナーが待機する user_clk 信号のサイクル間数を示します。s_axi_nfc_tx_tready がアサートされるまで High を保持します。データ送信を含まない user_clk サイクル数は、s_axi_nfc_tx_tdata [8:15] に 1 を加えた数に相当します。s_axi_nfc_tx_tdata [7] (アクティブ High) は nfc_xoff にマップされ、XOFF NFC 以外のメッセージを受信するリセットされるまで、チャンネルパートナーにデータ送信を停止するよう要求します。 信号マップ : s_axi_nfc_tx_tdata = {7'h0, NFC XOFF bit, NFC Data}
s_axi_nfc_tx_tready	入力	Aurora コアが NFC 要求を受信するとアサートされます (アクティブ High)。
channel_up	入力	Aurora チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
user_clk	入力	Aurora 64B/66B コアとユーザー アプリケーションで共有されるパラレルクロックです。
reset	入力	Aurora コアをリセットします (アクティブ High)。

## ユーザー K の TX インターフェイス

ユーザー K データを送信するには、FRAME\_GEN が制御信号を操作して次を実行します。

- ユーザー K のフレーム内ギャップの後に s\_axi\_user\_k\_tx\_tvalid がアサートされます。
- あらかじめ定義されたユーザー K データは、ユーザー K ブロック番号と共に送信されます。ユーザー K ブロック番号は、最初のユーザー K ブロックに対して 0 に設定され、次のユーザー K ブロックから 8 に到達するまで 1 つずつインクリメントされます。
- User K の送信周波数は、USER\_K\_IFG パラメーターで指定されます。

図 5-6 に、Aurora 64B/66B コアの FRAME\_GEN ユーザー K TX インターフェイスとユーザー K TX データ用の AXI4-Stream に準拠するポートを示します。



X13032

図 5-6 : Aurora 64B/66B コアのユーザー K の TX インターフェイス (FRAME\_GEN)

表 5-6 では、FRAME\_GEN ユーザー K の TX データ ポートとそれらの説明を示しています。

表 5-6 : FRAME\_GEN ユーザー K ユーザー I/O ポート (TX)

名前	方向	説明
s_axi_user_k_tdata [0:(n*64-1)]	出力	ユーザー K ブロック データです。s_axi_user_k_tx_tdata = {4'h0, USER K BLOCK NO, USER K DATA[0:56n-1]}
s_axi_user_k_tx_tvalid	出力	s_axi_user_k_tdata ポート上のユーザー K データが有効の場合にアサートされます (アクティブ High)。
s_axi_user_k_tx_tready	入力	Aurora 8B/10B コアが s_axi_user_k_tx_tdata インターフェイスからデータを読み出す準備が整うとアサート (アクティブ High) されます。
channel_up	入力	Aurora 8B/10B チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
user_clk	入力	Aurora 64B/66B コアとユーザー アプリケーションで共有されるパラレルクロックです。
reset	入力	Aurora コアをリセットします (アクティブ High)。

## FRAME\_CHECK

### フレーミング RX データ インターフェイス

フレーム RX データの想定値が LFSR で計算されます。受信したユーザー データは、次の AXI4-Stream プロトコル規則に対してチェックされ有効/無効が判断されます。

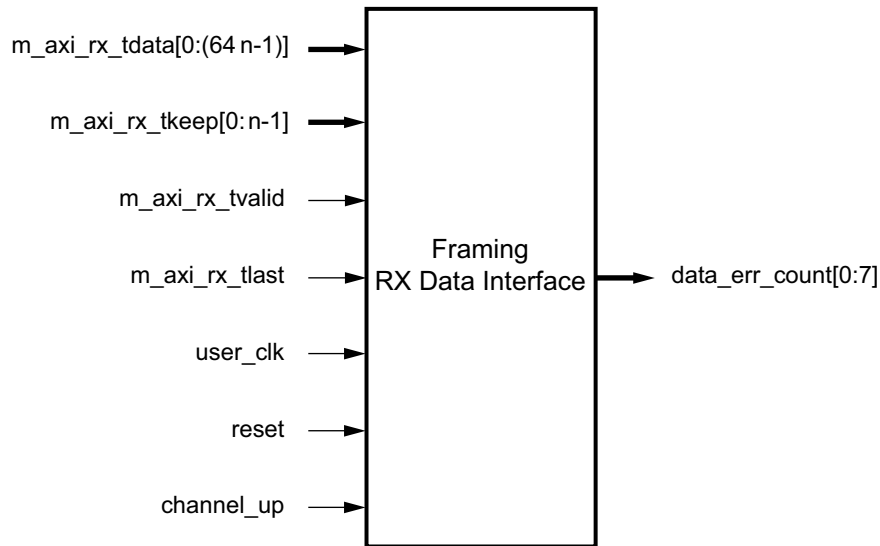
m\_axi\_rx\_tvalid がアサートされると、フレームが開始します。

1. m\_axi\_rx\_tkeep バスは、m\_axi\_rx\_tlast がアサートされている間有効となります。
2. m\_axi\_rx\_tvalid 信号は、想定値と実際の値が比較されている間アサートされる必要があります。

m\_axi\_rx\_tdata ポートに入力される RX データがレジスタに格納されて、FRAME\_CHECK 内にある計算された RX データと比較されます。入力された RX データが想定した RX データと一致しない場合、8 ビット カウンターがインクリメントされます。このエラー カウンターは、data\_err\_count ポートを介してユーザー アプリケーションへ伝えられます。エラー カウンターは、255 に達するとカウントを停止します。

注記 : カウンターは、リセットして 0 に戻すことができます。

図 5-7 に、Aurora 64B/66B コアの FRAME\_CHECK フレーミング ユーザー インターフェイスと RX データ用の AXI4-Stream に準拠するポートを示します。



X13015

図 5-7 : Aurora 64B/66B コアのフレーミング RX データ インターフェイス (FRAME\_CHECK)

表 5-7 では、FRAME\_CHECK フレーミング RX データ ポートとそれらの説明を示しています。

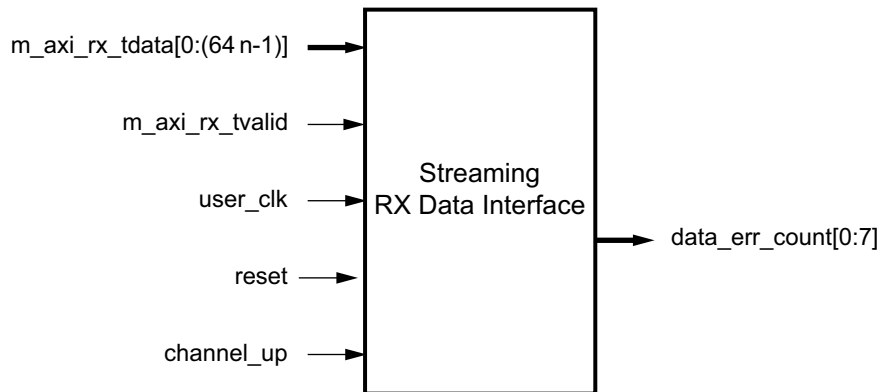
表 5-7 : FRAME\_CHECK フレーミングのユーザー I/O ポート (RX)

名前	方向	説明
m_axi_rx_tdata[0:(64n-1)]	入力	チャンネル パートナーから入力されるフレーム データです (昇ビット順)。
m_axi_rx_tkeep[0:n-1]	入力	最後のデータ ビートで有効なバイト数を示します。m_axi_rx_tlast がアサートされている場合のみ有効です。
m_axi_rx_tvalid	入力	Aurora コアからのデータおよび制御信号が有効の場合にアサート (High) されます。Aurora コアからのデータまたは制御信号を無視する場合にはディアサート (Low) されます。
m_axi_rx_tlast	入力	入力されるフレームの最後を示します (アクティブ High で、user_clk の 1 サイクル間アサートされる)。
data_err_count[0:7]	出力	フレーム チェッカーが受信した想定値と異なる RX フレーム データワード数を示します。
channel_up	入力	Aurora 8B/10B チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
user_clk	入力	Aurora 64B/66B コアとユーザー アプリケーションで共有されるパラレルクロックです。
reset	入力	Aurora コアをリセットします (アクティブ High)。

## ストリーミング RX データ インターフェイス

- ストリーミング モードの場合、入力される RX データが想定された RX データに対して比較されます。
- RX データは、m\_axi\_rx\_tvalid がアサートされている場合のみ比較されます。

図 5-8 に、RX データ用の Aurora 64B/66B コアの FRAME\_CHECK ストリーミング ユーザー インターフェイスを示します。



X13020

図 5-8 : Aurora 64B/66B コアのストリーミング RX データ インターフェイス (FRAME\_CHECK)

表 5-8 では、FRAME\_CHECK ストリーミング RX データ ポートとそれらの説明を示しています。

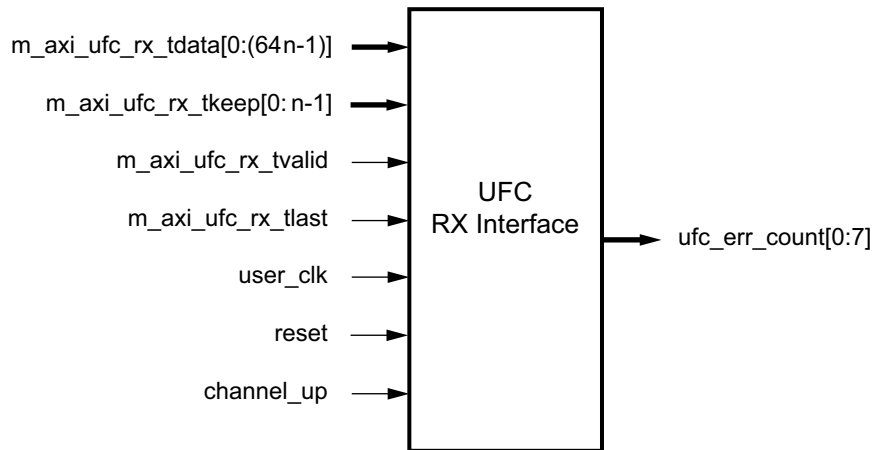
表 5-8 : FRAME\_CHECK ストリーミングのユーザー I/O ポート (RX)

名前	方向	説明
m_axi_rx_tdata[0:(64n-1)]	入力	チャンネル パートナーから入力されるフレーム データです (昇ビット順)。
m_axi_rx_tvalid	入力	Aurora コアからのデータおよび制御信号が有効の場合にアサート (High) されます。Aurora コアからのデータまたは制御信号を無視する場合にはディアサート (Low) されます。
data_err_count[0:7]	出力	フレーム チェッカーが受信した想定値と異なる RX データワード数を示します。
channel_up	入力	Aurora 8B/10B チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
user_clk	入力	Aurora 64B/66B コアとユーザー アプリケーションで共有されるパラレル クロックです。
reset	入力	Aurora コアをリセットします (アクティブ High)。

## UFC RX インターフェイス

- UFC RX データの想定値が LFSR で計算されます。
- エラー チェック機能およびカウンタ ロジックは、「[フレーミング RX データ インターフェイス](#)」と同じです。
- 入力された `m_axi_ufc_rx_tdata` が想定した RX UFC データと一致しない場合、8 ビット カウンタがインクリメントされます。
- このエラー カウンタは、`ufc_err_count` ポートを介してユーザー アプリケーションへ伝えられます。

図 5-9 に、Aurora 64B/66B コアの FRAME\_CHECK UFC RX インターフェイスと UFC RX データ用の AXI4-Stream に準拠するポートを示します。



X13025

図 5-9 : Aurora 64B/66B コアの UFC RX インターフェイス (FRAME\_CHECK)

表 5-9 では、FRAME\_CHECK UFC RX データ ポートとそれらの説明を示しています。

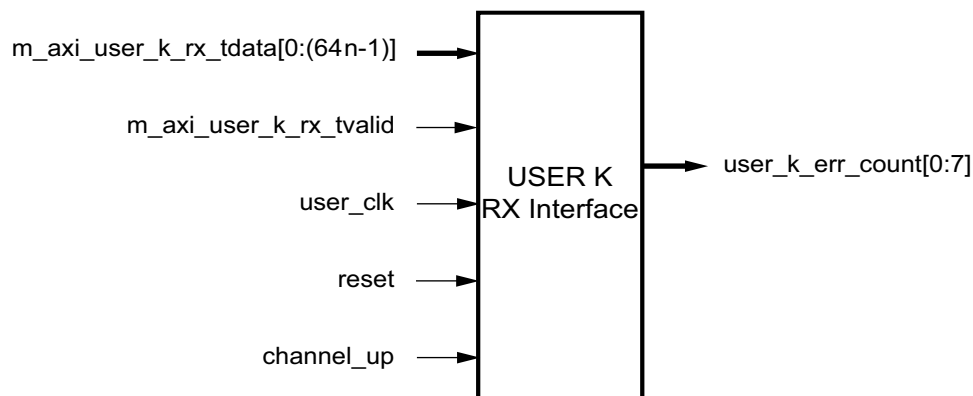
表 5-9 : FRAME\_CHECK UFC のユーザー I/O ポート (RX)

名前	方向	説明
m_axi_ufc_rx_tdata [0:(64n-1)]	入力	チャンネル パートナーから送られる UFC メッセージ データです。
m_axi_ufc_rx_tkeep [0:n-1]	入力	UFC メッセージの最後のワードで m_axi_ufc_rx_tdata ポートに現れる有効なバイト データ数を指定します。m_axi_ufc_rx_tlast がアサートされている場合のみ有効です。n = 最大 256 バイト
m_axi_ufc_rx_tvalid	入力	m_axi_ufc_rx_tdata ポートの値が有効な場合にアサートされます (アクティブ High)。この信号がアサートされない場合、m_axi_ufc_rx_tdata ポートのすべての値は無視されます。
m_axi_ufc_rx_tlast	入力	入力される UFC メッセージの終わりを示します。
ufc_err_count[0:7]	出力	フレーム チェッカーが受信した想定値と異なる RX UFC データワード数を示します。
channel_up	入力	Aurora 8B/10B チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
user_clk	入力	Aurora 64B/66B コアとユーザー アプリケーションで共有されるパラレル クロックです。
reset	入力	Aurora コアをリセットします (アクティブ High)。

## ユーザー K の RX インターフェイス

- 想定値と実際のユーザー K データが比較されている間、m\_axi\_rx\_user\_k\_tvalid がアサートされます。
- 入力される m\_axi\_rx\_user\_k\_tdata があらかじめ定義されたユーザー K データに対して比較されます。
- これらが一致しない場合、8 ビットの user\_k\_err\_count がインクリメントされます。
- このエラー カウンターは、user\_k\_err\_count ポートを介してユーザー アプリケーションへ伝えられます。

図 5-10 に、Aurora 64B/66B コアの FRAME\_CHECK ユーザー K RX インターフェイスとユーザー K RX データ用の AXI4-Stream に準拠するポートを示します。



X13029

図 5-10 : Aurora 64B/66B コアのユーザー K RX インターフェイス (FRAME\_CHECK)

表 5-10 では、FRAME\_CHECK ユーザー K の RX データ ポートとそれらの説明を示しています。

表 5-10 : FRAME\_CHECK ユーザー K のユーザー I/O ポート (RX)

名前	方向	説明
m_axi_rx_user_k_tvalid	入力	m_axi_rx_user_k_tdata ポート上のユーザー K データが有効の場合にアサートされます (アクティブ High)。
m_axi_rx_user_k_tdata[0:(64n-1)]	入力	Aurora レーンからユーザー K ブロックを受信します。 各レーンの信号マップ: m_axi_rx_user_k_tdata={4'h0, User K Block No, User K Data}
user_k_err_count[0:7]	出力	フレームチェッカーが受信した想定値と異なる RX ユーザー K データワード数を示します。
channel_up	入力	Aurora チャンネルの初期化が完了し、チャンネルがデータ送信可能な状態になるとアサートされます (アクティブ High)。
user_clk	入力	Aurora 64B/66B コアとユーザーアプリケーションで共有されるパラレルクロックです。
reset	入力	Aurora コアをリセットします (アクティブ High)。

Aurora 64B/66B サンプル デザインは、合成は XST で検証され、シミュレーションは Mentor Graphics Questa® で検証されています。

## サンプル デザインの実装

サンプル デザインは、IP コアから生成する必要があります。これを行うには、生成した IP を右クリックします。右クリックメニューから [Open Example Design] をクリックします。これで、生成した IP コア用のサンプル デザインが開きます。[Run Implementation] をクリックすると、合成とインプリメンテーションを実行できます。その他、[Generate Bitstream] をクリックして、ビットストリームを生成できます。

注記 : XDC でデザインのすべての入力および出力ポートに LOC および IO 規格を指定する必要があります。

## サンプル デザインのハードウェア リセット FSM

Aurora 64B/66B v9.2 コア サンプル デザインには、反復リセットの実行やリンクの堅牢性をモニターするハードウェア リセット FSM が統合されています。この FSM には、リセット信号のアサート間隔を異なる値に設定するオプションもあります。また、channel\_up と link\_reset の状態遷移カウンタが連続的にモニターされて、VIO を介してテスト ステータスがレポートされます。

リンクをプローブするために、次の信号がデフォルトの ILA および VIO に追加されています。

i\_ila

- tx\_d\_i [0:15] : LocalLink Frame Gen モジュールからの TX データ信号
- rx\_d\_i [0:15] : LocalLink Frame チェック モジュールへの RX データ信号
- data\_err\_count\_o : 8 ビットのデータ エラー カウント値 (通常動作では 'd0)
- lane\_up\_vio\_usrclk : lane\_up 信号
- channel\_up\_i : channel\_up 信号
- soft\_err\_i : ソフト エラー モニター

- `hard_err_i`: ハード エラー モニター

`vio1_inst`:

- `sysreset_from_vio_i`: サンプル デザインの `reset` 入力
- `gtreset_from_vio_i`: サンプル デザインの `pma_init`
- `vio_probe_in2`: リンク ステータス用のクオリティ カウンター
- `rx_cdovrden_i`: ループバック モードを有効にする間使用
- `loopback_i`: ループバック モードを有効にする間使用

`vio2_inst`:

- `reset_quality_cntrs`: サンプル デザインのすべてのクオリティ カウンターをリセットする際に使用
- `reset_test_fsm_from_vio`: ハードウェア リセット テスト FSM をリセットする際に使用
- `reset_test_enable_from_vio`: ハードウェア上で VIO ポートを介して反復リセット テストを有効化/開始する場合に使用
- `iteraion_cnt_sel_from_vio`: 開始する反復リセットの反復回数。固定の反復回数を示すエンコードされた 4 ビットの値で、[Vivado lab tools] がオンに設定されている場合にサンプル デザインに表示
- `lnk_reset_in_initclk`: `link_reset` のアサートをモニターするための入力プローブ
- `soft_err_in_initclk`: `soft_err` ステータスをモニターするための入力プローブ
- `chan_up_transcnt_20bit_i [15:8]`: `channel_up` のトランザクション回数。完了したリセット反復回数をモニターするために使用

注記:

- a. `chan_up_transcnt_20bit_i` は、[15:8] ビットのみプローブされます。したがって、このプローブはステータスの更新に多少時間がかかります。
- b. リセット反復回数を変更する場合は、`iteraion_cnt_sel_from_vio` のそれぞれの値を変更し、それに応じて `chan_up_transcnt_20bit_i` を選択してステータスをプローブします。

`vio3_inst`:

- `test_passed_r`: リセットが問題なく完了した場合、それぞれの反復回数が終了した後に、このテスト パス ステータスがアサート
- `test_failed_r`: `channel_up` 信号の欠如またはデータ エラーが生じた場合に、このテスト フェイル ステータスがアサート
- `lnkrst_cnt_20bit_vio_i`: `link_reset` がアサートされる回数をモニターするためのプローブ信号
- `reset_test_fsm_chk_time_sel`: リセット信号がディアサートされた後、`channel_up` のアサート用ハードウェア `reset_fsm` チェック タイムを選択するための、エンコードされた 3 ビットのプローブ信号

ハードウェア FSM オペレーション:

サンプル デザイン (<user\_component\_name>\_exdes.v) には、反復リセット時のリンクの堅牢性をテストするために、ハードウェア主導の反復リセット FSM が追加されています。この FSM には、IDLE、ASSERT\_RST、DASSERT\_RST、WAIT、WAIT1、CHECK、FAIL、および DONE ステートがあります。

1. IDLE ステート時には、リセット テストに合格したことを示す `test_passed_r`、リセット テストに合格しなかったことを示す `test_failed_r`、およびリセットの反復回数を示す `timer_r` がデフォルトの 0 になります。
2. `vio` からの `reset_test_enable_from_vio` 信号がアサートされると、ハードウェア FSM が ASSERT\_RST ステートへ遷移し、`pma_init` が指定時間 (28 ビットのカウンタ数) アサートされます。
3. この `pma_init` がアサートされることによって、リンク パートナーがホットプラグを検出します。その後、ハードウェア FSM が DEASSERT\_RST ステートへ遷移し、`pma_init` がディアサートされてタイマーにデフォルト値があらかじめロードされます。この値は、`reset_test_fsm_chk_time_sel` `vio` 信号を使用して指定できます。

4. その後、FSM は、選択した時間を経過するまで WAIT ステートへ遷移します。このステートでは、データエラーやソフト エラーなどのすべてのエラーが確認されます。また `channel-up` が High にアサートされており、`pma_init` の繰り返しに対して 2 回以上トグルしていないことが確認されます。
5. これらの条件が満たされない場合、FSM はフェイル ステートへ遷移し、反復リセットの実行を停止します。満たされている場合は、パケットがほとんど送受信されない WAIT1 ステートへ遷移します。
6. 次の CHECK ステートでは、`channel-up` の遷移が再び確認されます。2 回以上の遷移がない場合、FSM は要求された繰り返し動作が完了するまで IDLE ステートへ戻ります。これによって、リンクの堅牢性が保証され、リンクの複数の反復リセットから確実に回復します。

## テストベンチ

Aurora IP コアは、サンプルデザイン用のデモ テストベンチを提供します。この章では、Aurora テストベンチおよびその機能について説明します。テストベンチには、次のモジュールが含まれます。

- 被試験デバイス (DUT)
- クロックおよびリセット ジェネレーター
- ステータス モニター

Aurora テストベンチのコンポーネントは、選択した Aurora コア コンフィギュレーションによって異なりますが、基本的な機能はすべてのコア コンフィギュレーションで共通です。

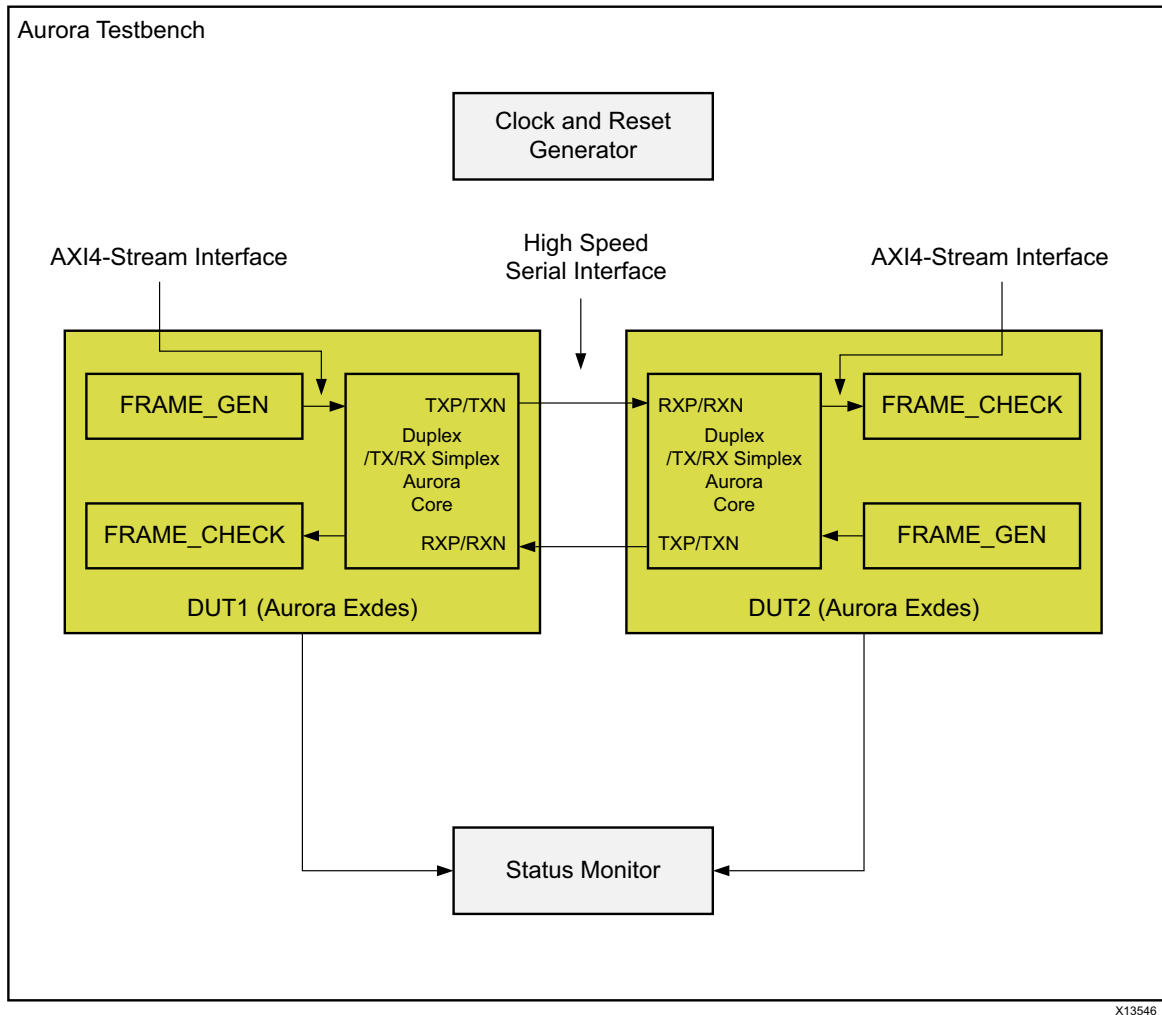


図 6-1: デュプレックス コンフィギュレーション用の Aurora テストベンチ

Aurora テストベンチ環境では、高速シリアル インターフェイスを使用してループバック モードでデュプレックス/TX/RX シンプレックス コアを接続します。図 6-1 に、デュプレックス/TX/RX シンプレックス コンフィギュレーション用の Aurora テストベンチを示します。

テストベンチは、チャンネルのステータスを確認し、その後、あらかじめ定義したシミュレーション期間のユーザーデータ、UFC データ、User-K データの整合性を検証します。channel\_up のアサーション メッセージによって、リンク トレーニングやチャンネル ボンディング (マルチレーン デザインの場合) が正常に行われたことを確認できます。FRAME\_CHECK モジュールがカウンターを管理しているため、エラー データの受信を確認できます。エラー データが受信されると、テストベンチがエラーをフラグします。

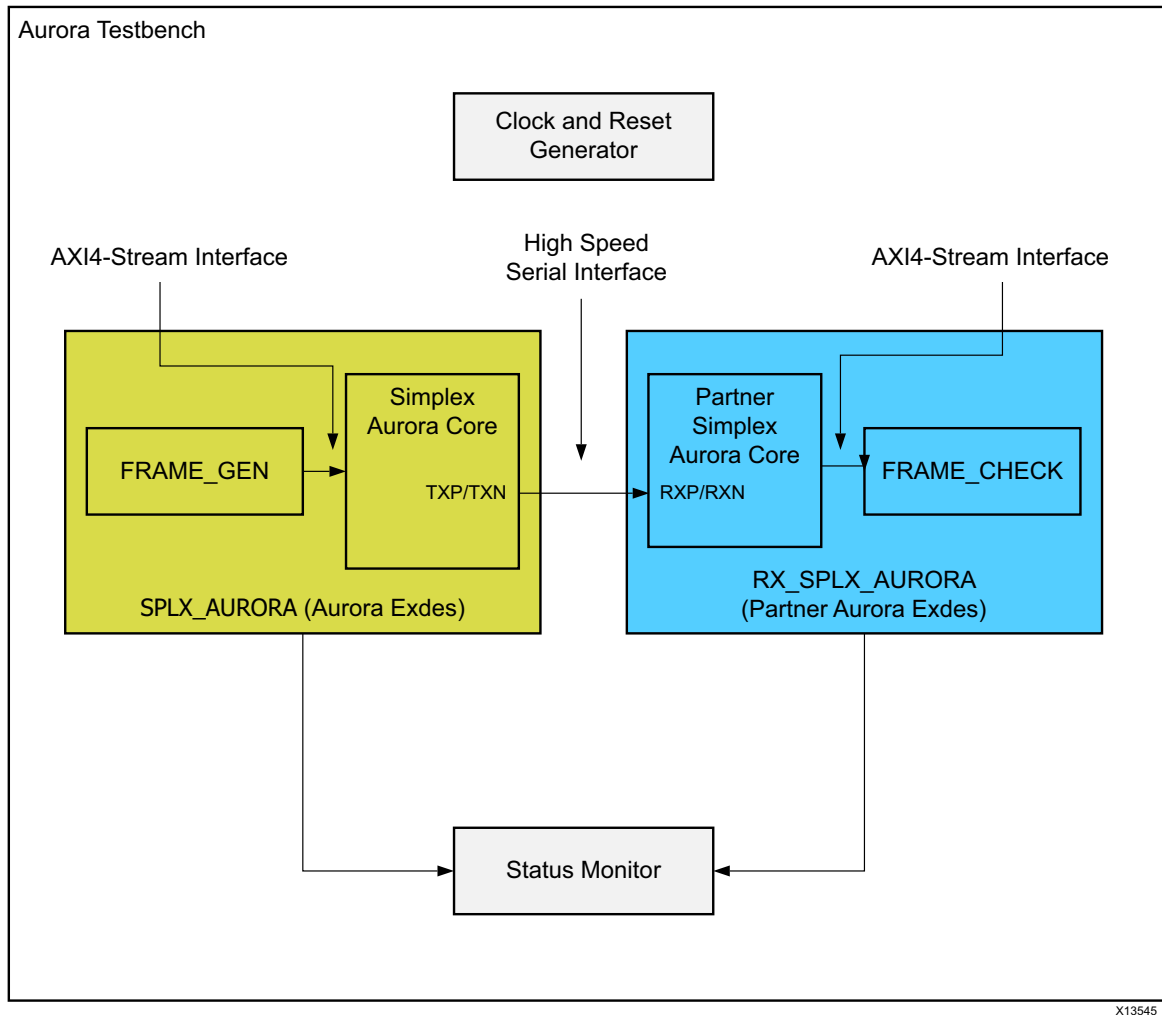


図 6-2: シンプレックス コンフィギュレーション用の Aurora テストベンチ

Aurora テストベンチ環境では、高速シリアル インターフェイスを使用してシンプレックス Aurora コアとパートナー シンプレックス Aurora コアを接続します。図 6-2 に、DUT1 が TX のみのシンプレックスとしてコンフィギュレーションされ、DUT2 が RX のみのシンプレックスとしてコンフィギュレーションされたシンプレックス コンフィギュレーションの Aurora テストベンチを示します。

テストベンチは、あらかじめ定義したシミュレーション期間の送信チャネルと受信チャネルのステータスを検出し、ユーザー データの整合性を検証します。tx\_channel\_up および rx\_channel\_up のアサーション メッセージによって、リンク トレーニングやチャネル ボンディング (マルチレーン デザインの場合) が正常に行われたことを確認できます。

## 検証、互換性、相互運用性

付録 A では、この IP コアに対して実行された互換性テストについて説明します。

Aurora 64B/66B コアは、自動化されたハードウェアおよびシミュレーション テストを使用してプロトコルに準拠しているかどうかを検証されます。このコアには、コアの機能の理解/検証に役立つ、LFSR (リニア フィードバック シフトレジスタ) を使用して実装されたサンプル デザインがあります。

Aurora 64B/66B コアは Aurora 64B/66B BFM (バス ファンクション モデル) と独自のカスタム テストベンチを使用して検証されています。Aurora 64B/66B BFM では、インターフェイス レベルのチェックおよびエラー シナリオに加えて、プロトコル準拠も検証されます。自動テストシステムにより、ランダムに選択された、一般に広く使用されているデザイン コンフィギュレーションで一連のシミュレーション テストが実行されています。また、Aurora 64B/66B コアは、ザイリンクスの GTX トランシーバーのデモ ボードを使用して、機能、性能、信頼性に関してハードウェアでテストされています。すべてのモジュールを対象とする Aurora 64B/66B 検証環境は、個々のモジュールのパラメーターすべてをテストできるよう、常に変更されています。

Aurora 8B/10B コアのハードウェア テストには、KC724、KC705、VC7203、および ZC723 ボードが使用され、一連のテスト シナリオが実証されています。

7 シリーズ FPGA GT トランシーバー用 Aurora 64B/66B コアのバージョン間の相互運用性を備えるため、新たにユーザー レベルのパラメーターがコアのバージョン v9.2 で導入されました。バージョン間で動作可能にするには、パラメーターを表 A-1 のように設定する必要があります。

表 A-1 : Aurora 64B/66B の相互運用性

Aurora 64B/66B V8_1 と V9_2 の相互運用性		
V9_2\V8_1	V8_1 GTX	V8_1 GTH
V9_2 GTX	√	√
V9_2 GTH	√	√
Aurora 64B/66B V7_3 と V9_2 の相互運用性		
V9_2\V7_3	V7_3 GTX	V7_3 GTH
V9_2 GTX	√	x
V9_2 GTH	√	x

初期のコア バージョンとの後方互換性を管理するため、2 つのパラメーター (BACKWARD\_COMP\_MODE1 および BACKWARD\_COMP\_MODE2) が <user\_component\_name>\_core.v モジュールに含まれています。

### BACKWARD\_COMP\_MODE1/BACKWARD\_COMP\_MODE2

- デフォルト値は 0 に設定されています。これにより、v9.2 コアと v9.1 コア間、および v9.2 コアと v9.0 コア間の相互運用性が保証されます。
- v9.2 コアと v8.1/v7.3 コア間の相互運用性を備えるには、これらのパラメーターを共に 1 に設定してください。

# 移行およびアップグレード

付録 B には、ISE® デザインを Vivado® Design Suite へ移行する際の情報、最新版 IP コアへのアップグレード、およびレガシー (LocalLink ベース) Aurora コアから AXI4-Stream Aurora コアへの移行に関する情報が記載されています。

Vivado Design Suite でアップグレードする場合のポート変更およびユーザー ロジックへの影響といった重要な情報もここに記載されています。

---

## デバイスの移行

7 シリーズ GTX または GTH デバイスから UltraScale™ GTH デバイスへ移行する場合、シングルレーン コアでは、オプションのトランシーバー デバッグ ポートの接頭語が gt0、gt1 から gt へ変更され、接尾語 \_in および \_out が削除されます。マルチレーン コアの場合、接頭語が付いたオプションのトランシーバー ポート gt(n) は 1 つのポートに集約されます。たとえば、gt0\_gtrxreset および gt1\_gtrxreset は、gt\_gtrxreset [1:0] となります。これはすべてのポートに対して適用されますが、例外として DRP バスは、gt (n)\_drpxyz の規則に従います。

新しいトランシーバー デバッグ ポート名を使用するようにデザインをアップデートする必要があります。UltraScale デバイスへの移行については、『UltraScale アーキテクチャへの移行手法ガイド』(UG1026) [参照 12] を参照してください。

---

## Vivado Design Suite への移行

Vivado Design Suite への移行方法については、『Vivado Design Suite 移行手法ガイド』(UG911) [参照 13] を参照してください。

---

## Vivado Design Suite でのアップグレード

このセクションでは、Vivado Design Suite でこの IP コアの最新版にアップグレードする際の、ユーザー ロジックおよびポートの変更について説明します。

コアの最新リビジョンでは、旧バージョンとのピン互換性を備えるためにいくつかの変更が加えられました。これらの変更は、使いやすさの向上を目的とし、通常の階層的変更の一部として適用されました。今後、このような変更はありません。

### 共有ロジック

コアの階層的変更の一環として、複数コアで共有できるすべてのロジックをコアに含めることが可能になり、すでにコアのサンプル デザインに含まれています。



**推奨:** 旧バージョンから共有ロジックを含む新バージョンへ簡単にアップグレードする方法はありません。詳細は、この資料の「共有ロジック」セクションを参照してください。

## v9.1 コアからのアップデート

表 B-1 では、v9.2 Aurora 64B/66B コアに追加された新しいポートについて説明し、v9.1 ベースの既存デザインにこれらのポートを追加する際の影響について説明しています。

表 B-1: 2014.1 Aurora 64B/66B に追加された新しいポート

新ポート	方向	追加の理由
gt_refclk1_out gt_refclk2_out	出力 (マスター)	共有ロジック デザインの場合、差動の GT 入力を持つマスター (共有ロジックがコアに含まれる) は IBUFDS をインスタンス化し、シングルエンドの refclk を GT へ渡します。スレーブ (共有ロジックがサンプル デザインに含まれる) はシングルエンドの refclk 入力を要求しますが、V9.1 バージョンのマスターでは対応できません。その結果、差動の GT refclk を追加し、IBUFDS を外部にインスタンス化して、スレーブの入力として使用するか、マスターから手動で gt_refclk [1,2] を生成します。このような理由から、v9.2 にはこれらの 2 つの出力ポートが追加されています。
gt_reset_out	出力 (マスター)	マスターおよびスレーブ デザインの適切な GT リセット シーケンスを保証するため、スレーブの pma_init 入力ポートへ接続される gt_reset_out が提供されています。
mmcm_not_locked_out	出力 (マスター)	スレーブ デザインには、入力として mmcm_not_locked ポートがあり、TX のスタートアップ FSM で使用されます。マスター デザインには、MMCM インスタンスがあり、出力として mmcm_not_locked を駆動します。 Aurora 64B/66B v9.1 コアには、この出力ポートはありません。
s_axi_bready s_axi_bready_lane[1..15]	入力 (DRP モードは AXI4-Lite)	AXI4-Lite との互換性を確保するため、DRP モードが AXI4-Lite の場合には、この新しいポートが追加されています。アップグレードを容易にするため、この入力はデフォルト値 1 に接続され、ユーザー ロジックで駆動される必要はありません。

IP をアップグレードする場合、これらのポートの追加に関する重大な警告メッセージが表示されます。新しいポートで提供される機能を使用しない場合は、これらのメッセージを無視しても問題ありません。

## v9.0 コアからのアップデート

- TX のスタートアップ FSM で、`mcm_lock_count` のカウント メカニズムは `txuserclk` で実行されました。これはリカバリ クロックであるため、制限がありました。新バージョンでは、MMCМ ロック同期化に `stable_clock` が使用されます。
- CBCC モジュールまでの RX データパスが 32 ビットになり、幅変換ロジックや `clk_en` の生成が不要になります。つまり、FIFO ヘデータを書き込む前に CBCC モジュールで制御されます。
- レーンのスキュー耐性が強化されました。より大きなレーン間スキューに対応できるようになりました。
- レーンの `init` が有効の間、極性の反転を検出し、極性を反転させるロジックがあります。
- 内部でコアが Aurora TX ロジック用に `tx_channel_up` を生成し、Aurora RX ロジック用に `rx_channel_up` を生成します。これによって、TX ロジックより先に RX ロジックがアクティブになり、受信可能な状態になります。`rx_channel_up` は、`channel_up` として提供されます。
- すべてのレーンに対して共通のリセットと制御信号があります。
- トランシーバー ユーザー ガイドで推奨されているとおり、RX CDR ロック タイムが 50KUI から 37MUI に増加しました。
- リンクの堅牢性向上のため、ブロックの `sync` ヘッダーの最大カウントが 64 から 60K に増加しました。
- チャンネルの初期化中におけるリンクの堅牢性を備えるために、より多くのアイドル文字が送信できるようになりました。
- スクランブラーのリセットを削除し、フリーランニングにして CDR ロックを高速化しました。スクランブラーから送信されるデフォルト パターンは NA アイドル文字のスクランブル値です。
- GTH トランシーバーの QPLL 属性がアップデートされました。(ザイリンクス アンサー [56332](#) を参照)
- 共有ロジック、オプションのトランシーバー制御ポートおよびステータス デバッグ ポートが追加されました。
- クロック乗せ換え用に同期装置をアップデートし、メタスタビリティによる MTBF (平均故障間隔) を削減しました。現在、共通の同期装置を使用し、最初のステージのフロップにのみフォールス パス制約を適用しています。
- IES (Cadence 社) および VCS シミュレーター (Synopsys 社) のサポートが追加されました。
- デバッグ用に Vivado ラボ ツールのサポートが追加されました。
- テスト品質向上のため、サンプル デザインにクオリティ カウンターが追加されました。
- 反復リセット テストの実行のため、サンプル デザインにハードウェア リセット ステート マシンが追加されました。

## レガシー (LocalLink ベース) Aurora コアから AXI4-Stream Aurora への移行

### 事前に必要なもの

- AXI4-Stream プロトコルをサポートする 64B/66B v9.x コアを含む Vivado デザイン ツールのビルド
- Aurora ディレクトリ構造の知識
- Aurora サンプル デザインの実行知識
- AXI4-Stream および LocalLink プロトコルに関する基本的な知識
- AXI4-Stream アップデートを含む、コアの最新製品ガイド (PG074)
- レガシー製品関連の資料: 『LogiCORE IP Aurora 64B/66B v4.2 データシート』(DS528) [[参照 14](#)]、『LogiCORE IP Aurora 64B/66B v4.1 スタートアップ ガイド』(UG238) [[参照 15](#)]、および 『LogiCORE IP Aurora 64B/66B v4.2 ユーザー ガイド』(UG237) [[参照 16](#)]

- ・ 移行ガイド (この付録資料)

## 主な変更点

主な変更点は、AXI4-Stream インターフェイスの追加です。

- ・ ユーザー インターフェイスが従来型 LocalLink (LL) から AXI4-Stream に変更されます。
- ・ すべての AXI4-Stream 信号はアクティブ High であるのに対して、LocalLink 信号はアクティブ Low です。
- ・ サンプル デザインのユーザー インターフェイスとデザインの最上位ファイルは AXI4-Stream です。
- ・ AXI4-Stream Aurora コアには新たにシム モジュールが追加され、AXI4-Stream 信号を LL へ変換し、また LL を AXI4-Stream 信号へ戻すために使用されます。
  - 送信インターフェイスの AXI4-Stream - LL 間シム モジュールは、すべての AXI4-Stream 信号を LL へ変換します。
  - シム モジュールは、AXI4-Stream と LocalLink 間での信号のアクティブ High とアクティブ Low の変更に対応します。
  - SOF\_N ビットと REM ビットのマップの生成は、シム モジュールで行われます。
  - 受信インターフェイスの LL - AXI4-Stream 間シム モジュールは、すべての LL 信号を AXI4-Stream へ変換します。
- ・ コアの最上位には、各インターフェイス (PDU、UFC、および NFC) に個別の AXI4-Stream - LL 間および LL - AXI4-Stream 間シム モジュールがインスタンス化されます。
- ・ Aurora サンプル デザインのフレーム ジェネレーターとチェッカーには、生成された AXI4-Stream デザインとインターフェイスするため、LL - AXI4-Stream 間および AXI4-Stream - LL 間シム モジュールがそれぞれインスタンス化されます。

## ブロック図

図 B-1 に、従来型 LocalLink インターフェイスを使用する Aurora サンプル デザインを示します。図 B-2 に、AXI4-Stream インターフェイスを使用する Aurora サンプル デザインを示します。

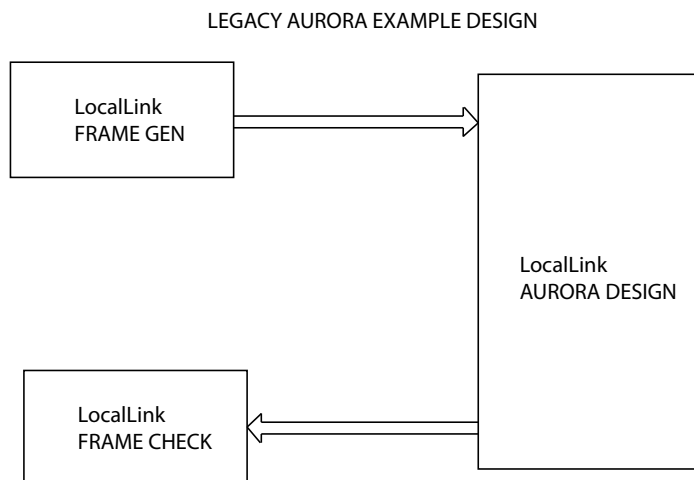


図 B-1: 従来型 LocalLink の Aurora サンプル デザイン

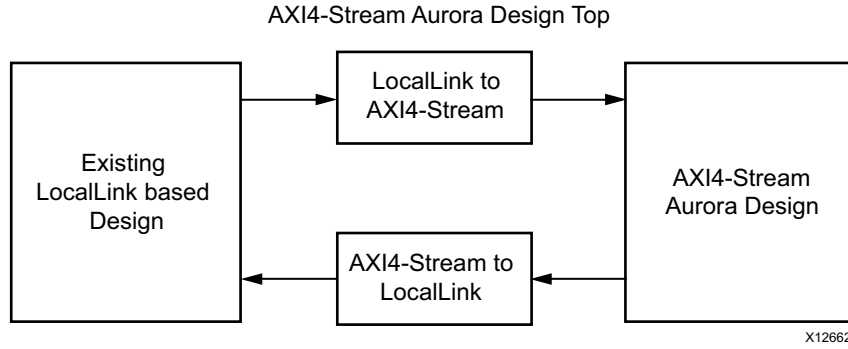


図 B-2: AXI4-Stream の Aurora サンプル デザイン

## 信号の変更

表 B-2: インターフェイスの変更

LocalLink 名	AXI4-S 名	相違点
TX_D	s_axi_tx_tdata	名前の変更のみ
TX_REM	s_axi_tx_tkeep	名前の変更。 機能的な変更の詳細は、13 ページの表 2-3 を参照。
TX_SOF_N		内部で生成
TX_EOF_N	s_axi_tx_tlast	名前の変更、極性
TX_SRC_RDY_N	s_axi_tx_tvalid	名前の変更、極性
TX_DST_RDY_N	s_axi_tx_tready	名前の変更、極性
UFC_TX_REQ_N	ufc_tx_req	名前の変更、極性
UFC_TX_MS	ufc_tx_ms	変更なし
UFC_TX_D	s_axi_ufc_tx_tdata	名前の変更のみ
UFC_TX_SRC_RDY_N	s_axi_ufc_tx_tvalid	名前の変更、極性
UFC_TX_DST_RDY_N	s_axi_ufc_tx_tready	名前の変更、極性
NFC_TX_REQ_N	s_axi_nfc_tx_tvalid	名前の変更、極性
NFC_TX_ACK_N	s_axi_nfc_tx_tready	名前の変更、極性
NFC_PAUSE	s_axi_nfc_tx_tdata	名前の変更。 信号マップの詳細は、17 ページの表 2-8 を参照。
NFC_XOFF		
USER_K_DATA	s_axi_user_k_tdata	名前の変更。 信号マップの詳細は、18 ページの表 2-9 を参照。
USER_K_BLK_NO		
USER_K_TX_SRC_RDY_N	s_axi_user_k_tx_tvalid	名前の変更、極性
USER_K_TX_DST_RDY_N	s_axi_user_k_tx_tready	名前の変更、極性
RX_D	m_axi_rx_tdata	名前の変更のみ
RX_REM	m_axi_rx_tkeep	名前の変更。 機能的な変更の詳細は、13 ページの表 2-3 を参照。
RX_SOF_N		削除
RX_EOF_N	m_axi_rx_tlast	名前の変更、極性

表 B-2: インターフェイスの変更 (続き)

LocalLink 名	AXI4-S 名	相違点
RX_SRC_RDY_N	m_axi_rx_tvalid	名前の変更、極性
UFC_RX_DATA	m_axi_ufc_rx_tdata	名前の変更のみ
UFC_RX_REM	m_axi_ufc_rx_tkeep	名前の変更。 機能的な変更の詳細は、 <a href="#">15 ページの表 2-7</a> を参照。
UFC_RX_SOF_N		削除
UFC_RX_EOF_N	m_axi_ufc_rx_tlast	名前の変更、極性
UFC_RX_SRC_RDY_N	m_axi_ufc_rx_tvalid	名前の変更、極性
RX_USER_K_DATA	m_axi_rx_user_k_tdata	名前の変更。 機能的な変更の詳細は、 <a href="#">18 ページの表 2-9</a> を参照。
RX_USER_K_BLK_NO		
RX_USER_K_SRC_RDY_N	m_axi_rx_user_k_tvalid	名前の変更、極性

## 移行手順

Vivado デザイン ツールで AXI4-Stream Aurora コアを生成します。

### コアをシミュレーションする

1. /simulation/functional ディレクトリにある vsim -do simulate\_mti.do ファイルを実行します。
2. Questa® SIM が起動し、モジュールをコンパイルします。
3. wave\_mti.do ファイルが自動的に AXI4-Stream 信号をロードします。
4. シミュレーションを実行します。これには多少の時間を要する場合があります。
  - a. 最初にレーン アップ信号がアサートされます。
  - b. 次にチャンネル アップ信号がアサートされ、データ転送が開始します。
  - c. すべてのフロー制御インターフェイスからデータ転送が開始します。
  - d. フレーム チェッカーが受信したデータを連続的にチェックし、データの不一致をすべてレポートします。
5. テストのステータスを示す TEST PASS または TEST FAIL ステータスが Questa SIM コンソールに表示されます。

### コアを実装する

1. /implement ディレクトリにある /implement.sh (Linux の場合) を実行します。
2. インプリメント スクリプトがコアをコンパイルし、Vivado デザイン ツール環境でファイルを実行して、コアのビット ファイルとネットリストを生成します。

### 既存の LocalLink ベース Aurora デザインへ統合する

1. Aurora コアには、既存の LL ベース インターフェイスと接続するための軽量なシム モジュールがあります。このシムは、aurora\_64b66b\_v8\_0 バージョンからコアに含まれています。
2. AXI4-Stream Aurora コアからの LL Aurora コアのエミュレーションは、[121 ページの図 B-2](#) を参照してください。
3. AXI4-Stream Aurora コアの src ディレクトリに、2 つのシム (<user\_component\_name>\_ll\_to\_axi.v および <user\_component\_name>\_axi\_to\_ll.v) が提供されています。
4. LL ベース デザインの最上位に、この 2 つのシムと <user\_component\_name>.v をインスタンスエートします。
5. [121 ページの図 B-2](#) のように、シムと AXI4-Stream Aurora デザインを接続します。
6. 最新の AXI4-Stream Aurora コアは、既存のすべての LL ベース デザインにプラグインできます。

## Vivado IDE の変更

図 B-3 の IP シンボルの図に AXI4-Stream 信号を示しています。

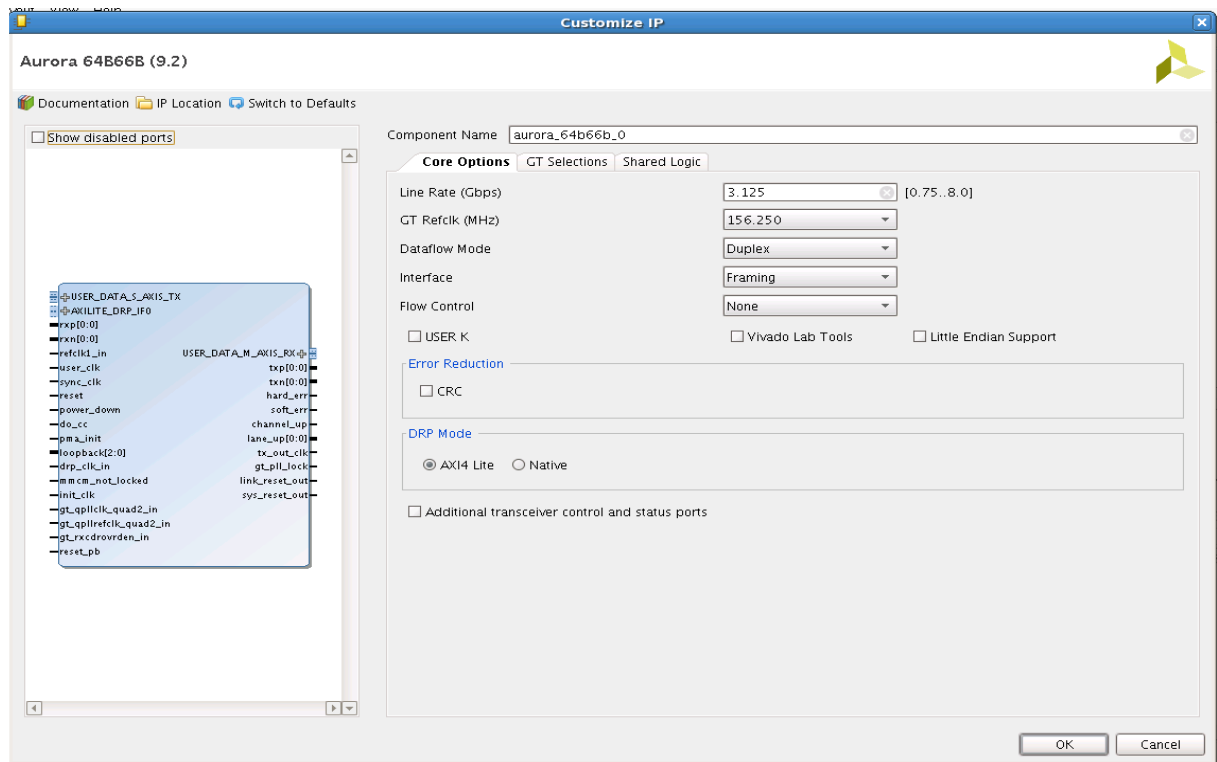


図 B-3 : AXI4-Stream 信号

## 制限

このセクションでは、AXI4-Stream をサポートする Aurora 64B/66B コアの制限事項について説明します。



---

**重要:** Aurora 64B/66B コアを AXI4-Stream 準拠のインターフェイス コアへ接続する際には、次に示す制限事項に注意する必要があります。

---

### 制限 1:

AXI4-Stream 仕様は、4 種類のデータ ストリームをサポートします。

- バイト ストリーム
- 連続的に位置合わせされたストリーム
- 連続的に位置合わせされていないストリーム
- スパース ストリーム

Aurora 64B/66B コアは、連続的に位置合わせされたストリームおよび連続的に位置合わせされていないストリームのみをサポートします。位置バイトは、パケットの最後でのみ有効です。

### 制限 2:

AXI4-Stream プロトコルは、パケットの最後でデータがない転送をサポートしますが、Aurora 64B/66B コアではパケットの最後で 1 バイト以上が有効である必要があります。

# 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

Aurora 64B/66B コア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 サポートでケースを開いてください。[Aurora ページ](#)も参照してください。

### 資料

この製品ガイドは Aurora 64B/66B コアに関する主要資料です。このガイド並びに全製品の設計プロセスをサポートする資料はすべて、ザイリンクス サポート ウェブ ページ ([japan.xilinx.com/support](http://japan.xilinx.com/support)) またはザイリンクスの Documentation Navigator から入手できます。

Documentation Navigator は、ダウンロード ページ ([japan.xilinx.com/download](http://japan.xilinx.com/download)) の [デザイン ツール] タブ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。

### アンサー

アンサーには、よく発生する問題についてその解決方法、および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、ユーザーが正確な情報にアクセスできるようにするため、随時作成および更新されています。

このコア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
- ツールで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。

アンサー データベースの検索機能の使用方法は次のとおりです。

1. [japan.xilinx.com/support](http://japan.xilinx.com/support) にアクセスします。検索ボックスは、このウェブ ページの上部にあります。
2. 検索ボックスにキーワードを入力して [検索] をクリックします。
  - 検索可能なキーワードの例には、製品名、エラー メッセージ、問題の概要などがあります。
  - Aurora 64B/66B コアに関するすべてのアンサーを表示させる場合は、「Aurora 64B66B」と入力して検索してください。

Aurora 64B/66B コアのマスター アンサー

AR : [54368](#)

ザイリンクスでは、さらにヘルプが必要なカスタマーに対して、テクニカル サポートを提供しています。

## テクニカル サポート

ザイリンクスでは、製品資料に記述されているように、[japan.xilinx.com/supoorot](http://japan.xilinx.com/supoorot) からこの LogiCORE™ IP 製品のテクニカル サポートを提供しています。資料で定義されていないデバイスにインプリメントしたり、製品資料で記述されている範囲を超えてカスタマイズしたり、あるいは「DO NOT MODIFY」と記述されているデザイン セクションに変更を加えたりした場合、タイミング、機能、製品サポートは保証されません。

テクニカル サポートへのお問い合わせ方法は、次のとおりです。

1. <http://japan.xilinx.com/support/> にアクセスします。
2. 「その他のリソース」の下の [\[ウェブケースを作成\]](#) リンクをクリックし、ウェブケースを開きます。

ウェブケースを作成する際は、次の情報を含めてください。

- パッケージおよびデバイス スピード グレードを含むターゲット FPGA の情報
- 該当するすべてのザイリンクス デザイン ツールとシミュレータのソフトウェア バージョン
- Aurora 64B/66B コア生成中に作成された XCI ファイル
- 問題によっては、ファイルの追加を求められる場合があります。ウェブケースに含める特定ファイルについては、この資料の関連セクションを参照してください。

**注記 :** すべての問題がウェブケースの利用対象になるわけではありません。ウェブケース ツールにログインしてサポート オプションを確認してください。

---

## デバッグ ツール

Aurora 64B/66B コア デザインの問題を解決するには、数多くのツールを利用できます。さまざまな状況をデバッグするのに有益なツールを理解しておくことが重要です。

## Vivado ラボ ツール

Vivado® ラボ ツールは、Logic Analyzer (ILA) および Virtual I/O (VIO) コアをユーザーのデザインに直接挿入します。Vivado ラボ ツールを使用すると、トリガー条件を設定して、ハードウェアで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と検証に使用されます。

Vivado ロジック アナライザーは次の論理デバッグ IP コアと共に使用されます。

- ILA 3.0 (およびそれ以降のバージョン)
- VIO 3.0 (およびそれ以降のバージョン)

このオプションの使用方法については、『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』(UG908) [\[参照 17\]](#) を参照してください。

## リファレンス ボード

Aurora 64B/66B コアはさまざまなザイリンクス開発ボードでサポートされています。これらのボードを使用してデザインのプロトタイプを作成し、コアがシステムと通信できるようにします。

- 7 シリーズ FPGA 評価ボード
  - KC705
  - KC724
  - VC7203
  - ZC723

## シミュレーション デバッグ

### シミュレーションでレーンアップとチャネルアップが確認できない

- これらの問題を解決する最も簡単な方法は、動作していない GTX または GTH トランシーバー インスタンスの出力信号を確認します。
- 基準クロックとユーザー クロックがすべてトグルしていることを確認します。

**注記:** 基準クロックは 1 つのみトグルする必要があり、残りは Low に接続されます。

- recclk および txoutclk 信号がトグルしているかをチェックします。これらがトグルしていない場合、PMA がロックを完了するまでしばらく待機する必要があります。通常、レーンアップおよびチャネルアップには約 6 ~ 9µs が必要です。シンプレックス/7 シリーズ FPGA デザインの場合は、もう少し時間がかかる可能性があります。
- txn と txp がトグルしているかを確認します。これらがトグルしていない場合、待機時間が十分であったか (前の項目を参照)、また別の信号で tx 信号を駆動していないかを確認してください。
- <user\_component\_name>\_support モジュールをチェックし、pll/mmcm\_not\_locked 信号および reset 信号がデザインに含まれていることを確認します。これらの信号がアクティブ状態で保持されている場合、Aurora モジュールは初期化を実行できません。
- power\_down 信号がアサートされていないことを確認します。
- 各 GTX または GTH トランシーバーからの txn および txp 信号が、チャネルの反対側にある対応する GTX/GTH トランシーバーの rxn および rxp 信号にそれぞれ接続されているかを確認します。
- glbl モジュールをインスタンス化し、このモジュールを使用してシミュレーション開始時に power\_up リセットを駆動してコンフィギュレーション後に生じるリセットをシミュレーションする必要があります。このリセット信号は、数サイクル間保持する必要があります。例として次のコードを使用できます。

```
//Simulate the global reset that occurs after configuration at the beginning
//of the simulation.
assign glbl.GSR = gsr_r;
assign glbl.GTS = gts_r;

initial
begin
    gts_r = 1'b0;
    gsr_r = 1'b1;
    #(16*CLOCKPERIOD_1);
    gsr_r = 1'b0;
end
```

- 複数のチャネルを使用している場合は、チャネルの両側のすべての GT が正しい順序で接続されていることを確認してください。

### シミュレーションでチャネルアップは確認できるが、S\_AXI\_TX\_TREADY がアサートされない (High にならない)

- モジュールにフロー制御が含まれているが使用していない場合、要求信号が High 駆動していないことを確認してください。s\_axi\_nfc\_tx\_tvalid と ufc\_tx\_req はアクティブ High です。つまり、これらが High の場合、チャネルがフロー制御用に割り当てられるため、s\_axi\_tx\_tready は Low のままとなります。
- do\_cc が継続的に High 駆動されていないことを確認します。クロックの次の立ち上がりエッジで do\_cc 信号が High 駆動されると常に、クロック コレクション文字を送信するためにチャネルが使用されるため、s\_axi\_tx\_tready がディアサートされます。

- モジュールにユーザー K ブロックが含まれているが使用していない場合、`s_axi_user_k_tx_tvalid` 信号が High 駆動していないことを確認してください。High の場合、チャンネルがユーザー K ブロック用に割り当てられるため、`s_axi_tx_tready` は Low のままとなります。
- NFC が有効の場合は、チャンネルの反対側のデザインが NFC XOFF メッセージを送信していないことを確認してください。反対側で NFC XON メッセージが送信され、再びフローが再開されるまで、通常データ用のチャンネルは無効となります。詳細は、ug775 (PDF) を参照してください。

## Aurora チャンネルを通過する際にバイトとワードが失われる

- AXI4-Stream インターフェイスを使用している場合、データの記述が正しいことを確認してください。最もよくある間違いとして、`s_axi_tx_tready` を確認せずにワードは書き込まれるものとする場合があります。また、`s_axi_tx_tlast` がアサートされているときにどのバイトが有効であるかを示すため、`s_axi_tx_tkeep` 信号を使用する必要があることにも注意してください。
- RX インターフェイスから正しく読み出しを実行しているかを確認してください。データおよびフレーミング信号は、`m_axi_rx_tvalid` がアサートされている間のみ有効です。

## デザイン コンパイル中の問題

コンパイル時、src ディレクトリのすべてのファイルを含めるようにしてください。

## 次の手順

サポート ケースを開いて、ザイリンクス サポート 担当者に問題について質問してください。

ウェブケースからテクニカル サポート ケースを作成するには、次のザイリンクス ウェブサイト ([japan.xilinx.com/support/clearexpress/websupport.htm](http://japan.xilinx.com/support/clearexpress/websupport.htm)) を参照してください。

ケースを作成する際に必要なもの:

- 問題の詳細説明
- 前述した手順を行った結果
- 観測した VCD または WLF ダンプの添付

## ハードウェア デバッグ

トランシーバーは、Aurora コアの重要な構築ブロックであるため、トランシーバー動作をデバッグして正常な動作を保証することが何よりも大切です。図 C-1 は、トランシーバー関連の問題をデバッグする際の手順を示しています。

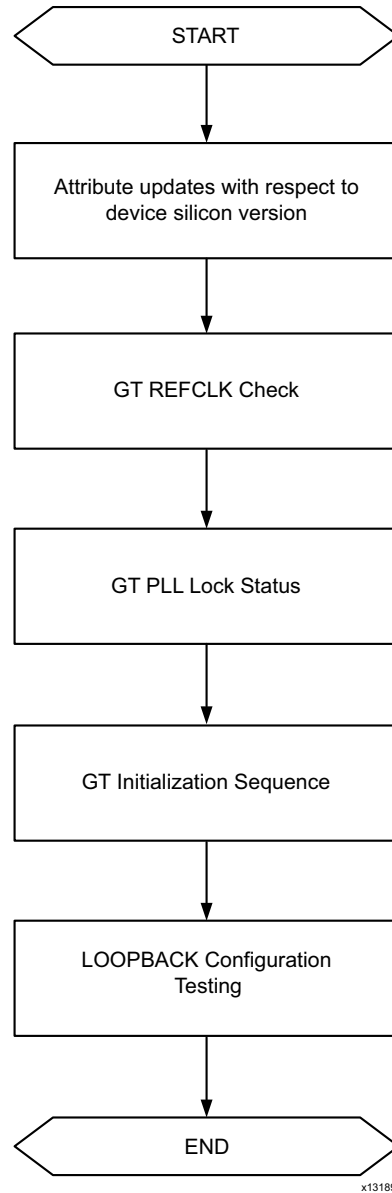


図 C-1: トランシーバー デバッグのフロー図

1. デバイス シリコン バージョンのトランシーバー属性に関するアップデートは、ボードで使用されているデバイスのシリコンバージョンに対応する必要があります。各シリコンのバージョンに与えられた適切な回避策およびアンサーのソリューションをすべて適用します。

2. GT REFCLK の確認

トランシーバーの基準クロックには低ジッターの差動クロックを供給する必要があります。オンボードの差動クロックをトランシーバーへ接続することによって、外部クロック生成の問題やトランシーバーへ接続される外部クロック ケーブルの問題に限定できます。

3. GT PLL ロックの確認

トランシーバーは、入力される GT REFCLK にロックし、p11lock 信号をアサートします。この信号は、Aurora サンプル デザインでは tx\_lock 信号として現れます。GT PLL 属性が適切に設定されており、トランシーバーが、指定したラインレートとデータパス幅に基づいて適切な周波数の txoutclk 信号を生成することを確認します。Aurora コアは、GTX または GTH トランシーバー用に生成されたコアの Channel PLL/Quad PLL (CPLL/QPLL) を使用するという点に留意してください。

4. GT 初期化シーケンス

Aurora コアは、シーケンシャル モードでリセット モードを使用するため、すべてのトランシーバー コンポーネントが順次リセットされます。トランシーバーの初期化が完了すると、txresetdone および rxresetdone 信号がアサートされます。通常、rxresetdone のアサート時間は、txresetdone よりも長くなります。gt\_reset 信号のパルス幅がそれぞれのトランシーバー ガイドラインに準拠していることを確認してください。txresetdone および rxresetdone 信号は、Aurora サンプル デザインに含まれており、モニターできます。

5. LOOPBACK コンフィギュレーション テスト

ループバック モードは、トランシーバー データパスの特殊なコンフィギュレーションです。Aurora サンプル デザインの loopback ポートがループバック モードを制御します。ループバック モードには 4 種類あり、それらのガイドラインおよび詳細は、該当するトランシーバーのユーザー ガイドを参照してください。図 C-2 に、4 つのループバック モードを備えたループバック テストのコンフィギュレーションを示します。

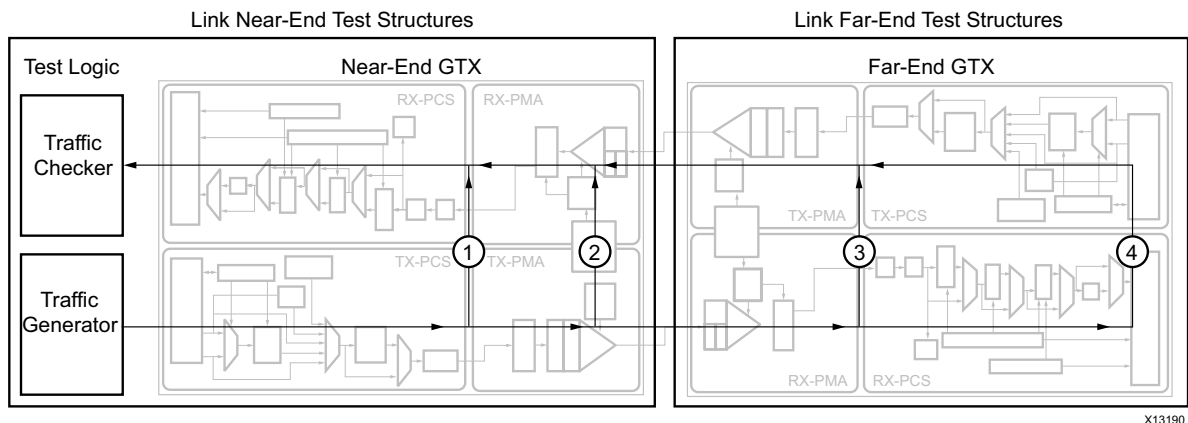


図 C-2: ループバック テストの概略図

# 評価ボードでデザインを実行

## KC705 ボードで Aurora を検証

セットアップ要件:

- ソフトウェア: Vivado Design Suite
- 必要なハードウェア コンポーネント:
  - Kintex-7 FPGA KC705 評価キットのベース ボード
  - 電源アダプター付き KC705 ボード (x2)

検証およびコア生成の手順:

1. Vivado Design Suite を起動して、新規プロジェクトを作成 (通常、デバイス番号は xc7k325tffg900-2 を使用) します。その他、ボード オプションの選択も可能です。[Finish] をクリックします。
2. Vivado の [Project Manager] で [IP Catalog] → [Communication & Networking] → [Serial Interfaces] → [Aurora 64B66B] を選択します。
3. Aurora 64B/66B コアをカスタマイズする場合は、[Core Option] タブで [Vivado Lab Tools] をオンにします。そして、[GT Selections] タブで GTXQ2 の [GTXE2\_X0Y8] を選択します。
4. コアを生成して、プロジェクトの IP サンプル デザインを開きます。

5. <user\_component\_name>\_exdes.xdc を開いて、Aurora コアのすべてのポートのピン位置が適切であることを確認します。評価ボードでは、hard\_err、soft\_err、および data\_err\_count が使用されないため、このファイルに次の行を追加できます。

```
set_property BITSTREAM.General.UnconstrainedPins {Allow} [current_design]
```

6. ファイルを保存します。

表 C-1: ピンの位置

ピン名	ボード上の位置	備考
init_clk_p	AD12	
init_clk_n	AD11	
reset	AG5	
pma_init	AC6	
lane_up	A8	
channel_up	AA8	
hard_err		LOC 制約されていない
soft_err		LOC 制約されていない
data_err_count		LOC 制約されていない
refclk_p	J8	GTXQ2_P
refclk_n	J7	GTXQ2_N

7. 合成とインプリメンテーションを実行して、ビットストリームを生成します。
8. ボードの接続手順は、次のとおりです。
  - a. ボード 1 の txp は ボード 2 の rxp に接続し、ボード 1 の txn はボード 2 の rxn へ接続してください。
  - b. 同様に、ボード 2 の txp はボード 1 の rxp に接続し、ボード 2 の txn はボード 1 の rxn へ接続してください。
9. ビット ファイルでボードをプログラムします。ila/vio を使用することによって、lane\_up、channel\_up、および data\_err\_count をモニターできます。

## インターフェイスのデバッグ

AXI4-Stream インターフェイスでデータが送信または受信されていない場合は、次を確認します。

- `s_axi_tx_tvalid` 入力のアサートされた後、送信の `s_axi_tx_tready` が Low のままになる場合、コアはデータを送信できません。
- 受信の `s_axi_tx_tvalid` が Low のままになる場合、コアはデータを受信しません。
- `user_clk` 入力が接続されており、トグルしていることを確認します。
- AXI4-Stream の波形に従っていることを確認します。(図 2-8 参照)。
- コアのコンフィギュレーションを確認します。
- コア特定のチェックを追加します。

# Transceiver Wizard でラッパー ファイルを生成

トランシーバーの属性は、Aurora 64B/66B コアの機能において重要な役割を果たします。最新の Transceiver Wizard を使用して、トランシーバーのラッパー ファイルを生成してください。



**推奨 :** Transceiver Wizard はアップデートされているが、Aurora コアはアップデートされていない場合、Design Suite ツール リリースでトランシーバーのラッパー ファイルをアップデートすることを推奨しています。

ここでは、トランシーバー ラッパー ファイルの生成方法について説明します。

これらの手順に従って、7 シリーズ FPGA トランシーバー ウィザードを使用してトランシーバー ラッパー ファイルを生成します。

1. IP カタログを使用する場合、[7 Series FPGA Transceivers Wizard] の最新バージョンを実行します。トランシーバー ウィザードのコンポーネント名は、Aurora 64B/66B コアのコンポーネント名と同じにしてください。
2. プロトコル テンプレートを選択します。(Aurora 64B/66B)。
3. アプリケーション要件に基づいて、TX と RX の [Line Rate] を変更します。
4. アプリケーション要件に基づいて、ドロップダウン メニューから TX と RX の基準クロックを選択します。
5. アプリケーション要件に基づいて、トランシーバーとクロック ソースを選択します。
6. 3 ページ目で、RX の [External Data Width] を 32 ビットに、[Internal Data Width] を 32 ビットに設定します。TX の外部データ幅が 64 ビット、内部データ幅が 32 ビットに設定されていることを確認します。
7. その他の設定はデフォルトを使用します。
8. コアを生成します。
9. Aurora 64B/66B コアの example\_design/gt/ ディレクトリにある <component name>\_gtx.v ファイルを 7 シリーズ FPGA Transceivers Wizard で生成された <component name>\_gt.v に置き換えます。

これで Aurora 64B/66B コアのトランシーバー設定が最新となります。

**注記 :** UltraScale™ アーキテクチャの Aurora 64B/66B IP コアは、階層的なコア呼び出し方法で UltraScale デバイスの GTWizard IP コアを呼び出します。これにより、すべてのトランシーバーの属性、パラメーター、および必要な回避策が最新となります。ほとんどの場合、UltraScale デバイスのトランシーバー ファイルを手動で変更する必要はありません。

# 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。

ザイリンクスの資料で使用されている技術用語については、[ザイリンクス用語集](#)を参照してください。

---

## 参考資料

次の資料は、この製品ガイドの補足資料として役立ちます。

1. 『7 シリーズ FPGA 概要』([DS180](#))
2. 『UltraScale アーキテクチャおよび製品概要』([DS890](#))
3. 『UltraScale アーキテクチャ GTH トランシーバー ユーザー ガイド』([UG576](#))
4. 『7 シリーズ FPGA GTX/GTH トランシーバー ユーザー ガイド』([UG476](#))
5. 『Aurora 64B/66B プロトコル仕様 v1.2』([SP011](#))
6. 『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](#))
7. 『Vivado Design Suite ユーザー ガイド : IP を使用した設計』([UG896](#))
8. 『Vivado Design Suite ユーザー ガイド : 入門』([UG910](#))
9. 『Vivado Design Suite ユーザー ガイド : ロジック シミュレーション』([UG900](#))
10. 『7 シリーズ GTZ トランシーバー ユーザー ガイド』([UG478](#))
11. 『UltraScale FPGA トランシーバー ウィザード製品ガイド』([PG182](#))
12. 『UltraScale アーキテクチャへの移行手法ガイド』([UG1026](#))
13. 『Vivado Design Suite ユーザー ガイド : 移行手法ガイド』([UG911](#))
14. 『LogiCORE IP Aurora 64B/66B v4.2 データ シート』([DS528](#))
15. 『LogiCORE IP Aurora 64B/66B v4.1 スタートアップ ガイド』([UG353](#))
16. 『LogiCORE IP Aurora 64B/66B v4.2 ユーザー ガイド』([UG237](#))
17. 『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』([UG908](#))
18. 『Vivado AXI リファレンス ガイド』([UG1037](#))
19. 『Virtex-7 FPGA データシート : DC 特性およびスイッチ特性』([DS183](#))
20. 『Kintex-7 FPGA データシート : DC 特性およびスイッチ特性』([DS182](#))
21. 『合成/シミュレーション デザイン ガイド』([UG626](#))
22. 『ARM® AMBA® 4 AXI4-Stream プロトコル仕様 v1.0』([ARM IHI 0051A](#))

## 改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	内容
2014年6月4日	9.2	<ul style="list-style-type: none"> <li>ユーザーパラメーター情報の追加。</li> </ul>
2014年4月2日	9.2	<ul style="list-style-type: none"> <li>合成/インプリメンテーション後のネットリストの論理シミュレーションを高速化するための C_EXAMPLE_SIMULATION の追加。</li> <li>UltraScale™ デバイスのサポート追加。</li> <li>IP インテグレーターの強化サポート。</li> <li>データおよびフロー制御インターフェイスにリトル エンディアン形式のサポートが追加 (デフォルトではなく、Vivado® IDE で選択)。</li> <li>相互運用性の説明追加。</li> <li>特定条件で特定のフレーム長にみられる機能的問題について。</li> </ul>
2013年12月18日	9.1	<ul style="list-style-type: none"> <li>init_clk_p、initclk_n、および INIT_CLK の説明にデフォルト情報を追加。</li> <li>リセットシーケンスの手順と波形を更新。</li> <li>pma_init ステージングに関する情報を追加。</li> <li>スクリーンショットを変更。</li> <li>ハードウェア FSM リセットの説明と手順を追加。</li> </ul>
2013年10月2日	9.0	<ul style="list-style-type: none"> <li>新しい章を追加: 「シミュレーション」、「テストベンチ」、「合成およびインプリメンテーション」。</li> <li>共有ロジックおよびトランシーバー デバッグ機能を追加。</li> <li>ディレクトリおよびファイル構造を変更。</li> <li>信号名とポート名の大文字信号表記を小文字に変更。</li> <li>Zynq®-7000 デバイスのサポートを追加。</li> <li>RX データパスアーキテクチャを更新。</li> <li>Aurora シンプレックス動作の説明を更新。</li> <li>第4章の図 3-2 およびスクリーンショットを更新。</li> <li>ホットプラグロジックの説明を更新。</li> <li>Vivado IP インテグレーターのサポートを追加。</li> <li>サンプルデザインの XDC ファイルを更新。</li> <li>評価ボード上でのデザイン起動情報を追加。</li> </ul>
2013年6月19日	8.1	<ul style="list-style-type: none"> <li>コアのバージョン番号と一致するようにリビジョン番号を 8.1 に変更。</li> <li>2013.2 リリースおよびコアバージョン 8.1 用に内容を変更。</li> <li>2 つ目の NFC 要求と共にクロックコレクションが送信される際の NFC 送信エラーシナリオを変更。これらのシナリオに対応して、NFC ステートマシンが更新。</li> </ul>
2013年3月20日	2.0	<ul style="list-style-type: none"> <li>2013.1 リリースおよびコアバージョン 8.0 用に内容を変更。</li> <li>すべての ISE® デザインツールと Virtex®-6 関連のデバイス情報を削除。</li> <li>リセット波形を追加。</li> <li>コアのデバッグガイドおよびトランシーバー デバッグの詳細説明を更新。</li> <li>Verilog 用の小文字表記のポートを追加。</li> <li>シンプレックス TX/RX のサポートを追加。</li> <li>プロトコルが強化されて Channel Init 時間が増加。</li> <li>GT リセットシーケンスを制御するために TXSTARTUPFSM および RXSTARTUPFSM モジュールが統合。</li> </ul>
2012年12月18日	1.0.1	<ul style="list-style-type: none"> <li>14.4 および 2012.4 リリース用に内容を更新。</li> <li>TKEEP の説明を追加。</li> <li>デバッグに関する付録を更新。</li> </ul>

日付	バージョン	内容
2012年10月16日	1.0	製品ガイドの初版リリース。この資料は、『LogiCORE IP Aurora 64B/66B ユーザーガイド』(UG775) および『LogiCORE IP Aurora 64B/66B データシート』(DS815) の置き換えになります。 <ul style="list-style-type: none"> <li>コアの制約に関する説明セクションを追加。</li> <li>コアのデバッグに関する説明セクションを追加。</li> </ul>
2014年6月4日		

## 法的通知

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>.

© Copyright 2012-2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. AMBA, AMBA Designer, ARM, ARM1176JZ-S, CoreSight, Cortex, and PrimeCell are trademarks of ARM in the EU and other countries. All other trademarks are the property of their respective owners.

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](mailto: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