

# 高集積度 FPGA 設計手法 ガイド

## スタックド シリコン インターコネクト (SSI) テクノロジー

UG872 (v14.3) 2012 年 10 月 16 日

該当するソフトウェア バージョン : Vivado Design Suite 2012.2 ~ 2013.1 および ISE Design Suite 14.3 ~ 14.5





## Notice of Disclaimer

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 the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

© Copyright 2012 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. All other trademarks are the property of their respective owners.

本資料は英語版 (v14.3) を翻訳したもので、内容に相違が生じる場合には原文を優先します。

資料によっては英語版の更新に対応していないものがあります。

日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

この資料に関するフィードバックおよびリンクなどの問題につきましては、[jpn\\_trans\\_feedback@xilinx.com](mailto:jpn_trans_feedback@xilinx.com) までお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。

## 改訂履歴

日付	バージョン	改訂内容
2012 年 10 月 16 日	14.3	<ul style="list-style-type: none"><li>デバイス特定の情報をアップデート</li></ul>
2012 年 4 月 24 日	14.1	<ul style="list-style-type: none"><li>表 3-1 「各 Virtex-7 SLR に含まれる主なリソース」の SLR 間のインターコネクトをアップデート</li><li>表 3-2 「各 SLR の交差ポイントに使用される SLL コンポーネント数」の SLL コンポーネントをアップデート</li><li>図 3-5 「SSI デバイスで少しずつずらされた SLL が交差」をアップデート</li><li>図 3-6 「SLR での SLL 接続」をアップデート</li></ul>

# 目次

---

改訂履歴.....	2
<b>第 1 章：概要</b>	
デザイン ストラテジ .....	5
高集積度 FPGA デバイス .....	5
SSI テクノロジ .....	7
<b>第 2 章：高集積度 FPGA デバイス設計手法</b>	
利点 .....	9
配線使用率 .....	10
デザイン パフォーマンス .....	11
消費電力 .....	12
プロジェクト コスト .....	12
<b>第 3 章：スタックド シリコン インターコネクト (SSI)</b>	
SSI コンポーネント .....	15
クロッキング .....	23
SLR コンポーネントでのデザイン配置の管理 .....	25
SSI コンフィギュレーション .....	28
<b>第 4 章：システム レベルのデザイン</b>	
ピン配置の選択 .....	31
制御セット .....	34
HDL コーディング手法 .....	38
<b>第 5 章：クロッキング</b>	
クロック リソースの選択 .....	45
グローバル クロッキング .....	46
リージョナル クロッキング .....	47
SSI デバイスのクロッキング .....	49
SSI デバイスのクロック スキュー .....	51
クロック位相、周波数、デューティ サイクル、およびジッターの制御 .....	55
出力クロック .....	57
クロック ドメインの交差 .....	58
非クロック ネットにクロック バッファーを使用 .....	61
クロック リソースのまとめ .....	63
<b>付録 A：その他のリソース</b>	
ザイリンクス リソース .....	67
ハードウェア資料 .....	67
ISE 資料 .....	67

パーシャル リコンフィギュレーション資料.....	68
PlanAhead 資料.....	68

## 概要

---

このガイドでは、高集積度 FPGA デバイスをターゲットにしたデザインについて解説します。[ストタックド シリコン インターコネクト \(SSI\)](#) テクノロジを使用したデザインについても説明します。

### デザイン ストラテジ

このガイドでは、次のストラテジを詳しく説明します。

- システム レベルのプランニング
- デザイン作成
- インプリメンテーション
- 解析

[第 2 章「高集積度 FPGA デバイス設計手法」](#)で説明されているように、これらのストラテジは高集積度 FPGA において次の面で最適な結果を得るのに役立ちます。

- [配線使用率](#)
- [デザイン パフォーマンス](#)
- [消費電力](#)
- [プロジェクト コスト](#)

### 高集積度 FPGA デバイス

「高集積度 FPGA デバイス」という表現は、このガイドではザイリックス Virtex®-6 および Virtex-7 デバイス ファミリの高集積度デバイスを指します。

次の図に示すように、デバイスの機能は新しい FPGA デバイス ファミリごとに向上しています。

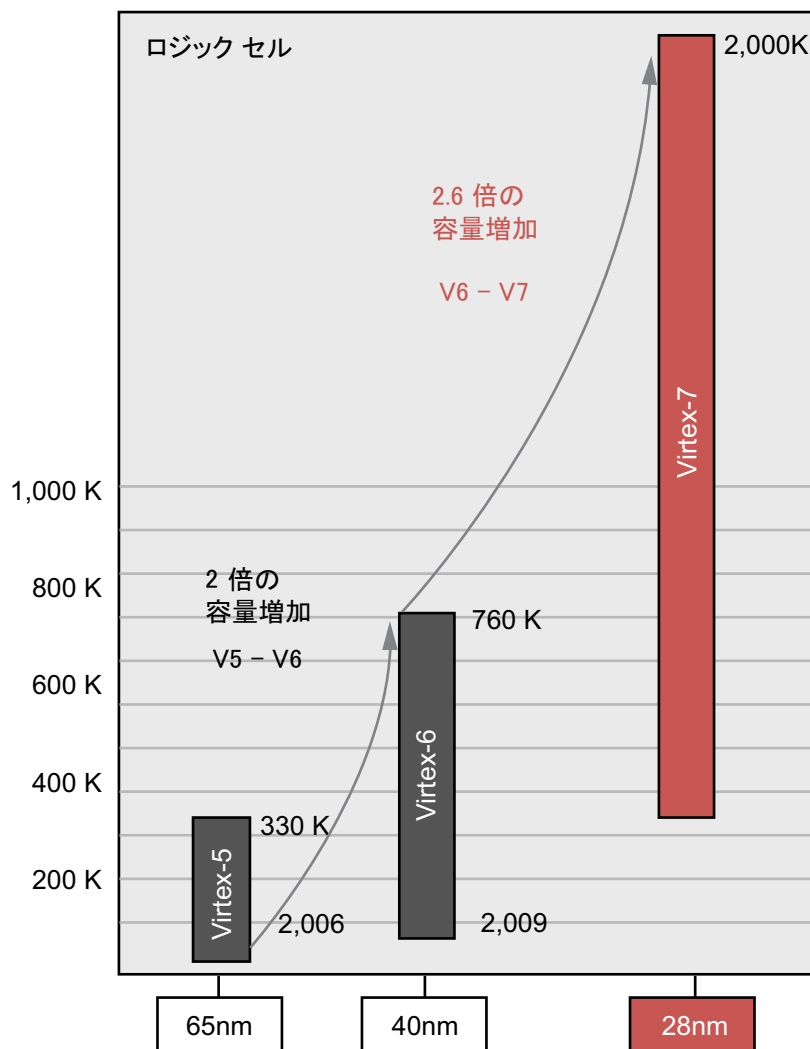


図 1-1 : FPGA の機能は 5 年未満で 6 倍以上向上

現時点で最も集積度の高い FPGA デバイスには、通常次のものが含まれています。

- 100 万個を超える 6 入力 LUT
- 200 万個を超えるレジスタ
- 数千個のブロック RAM コンポーネントおよび DSP ブロック
- 1000 個を超える汎用 I/O コンポーネント
- 最大 96 個のマルチギガビット トランシーバー (GT)
- その他多数のファンクションおよびリソース

機能が大きく飛躍したことで、大型システムを少ないチップに、さらには 1 つのチップに統合することが可能になっています。

## SSI テクノロジ

大容量、高性能を誇る Virtex-7 FPGA は、スタックド シリコン インターコネクト (SSI) テクノロジと呼ばれる製造プロセスを使用して作成されています。

SSI を採用した Virtex-7 をターゲットにするとともに、ほかの大型 FPGA デザインで使用されるのと同じツール、テクニック、手法が多く使用されますが、Virtex-7 アーキテクチャは特殊なため、特別な注意事項がいくつかあります。

詳細は、[第 3 章「スタックド シリコン インターコネクト \(SSI\)」](#)を参照してください。





# 高集積度 FPGA デバイス設計手法

---

合成およびインプリメンテーション ツールは、FPGA デバイスの固定リソースを最適に使用する必要があります。

この目標を達成するため、このガイドでは高集積度 FPGA デバイス設計手法の次の項目について説明します。

- コードの記述方法
- インプリメンテーション手法
- デザイン テクニック

## 利点

FPGA デバイスは急速に向上してきたため、リソース使用率、パフォーマンス、消費電力をはじめとするデザイン目標を達成するには、従来のコード記述方法やインプリメンテーション手法は十分ではありません。

ザイリンクス高集積度 FPGA デバイス設計手法を利用することで、次のようなデバイスやデザインの特性を最大限に活用した設計が可能となります。

- [配線使用率](#)
- [デザイン パフォーマンス](#)
- [消費電力](#)

また、この手法で次の点における効率を高めることができます。

- ソフトウェア ランタイム
- デバッグ
- 移植性

効率の悪いコード記述やインプリメンテーションはデザイン目標の達成を大きく妨げることになります。これらは、デザインまたはデバイスのサイズにかかわらず影響しますが、高集積度デバイスをターゲットにしたデザインには特に影響します。

このガイドで説明するトピックの多くは、特に新しいものでも高集積度デバイスに限られたものでもありませんが、大型の FPGA デザインにこうした手法を採用することで、デザイン目標を満たすだけでなく、それを上回る結果を得ることが可能になります。

## 配線使用率

FPGA デバイスの配線は、固定された有限リソースです。

配線リソースを正しく管理しないと、次のような FPGA デザイン特性に悪影響を与える可能性があります。

- リソース使用率
- パフォーマンス目標の達成
- 電力要件の達成または消費電力の低減

配線の使用は、次の項目に直接関係しています。

- システム レベル デザインの選択
- デザイン入力
- コードの記述方法
- インプリメンテーションおよびデバッグの方法

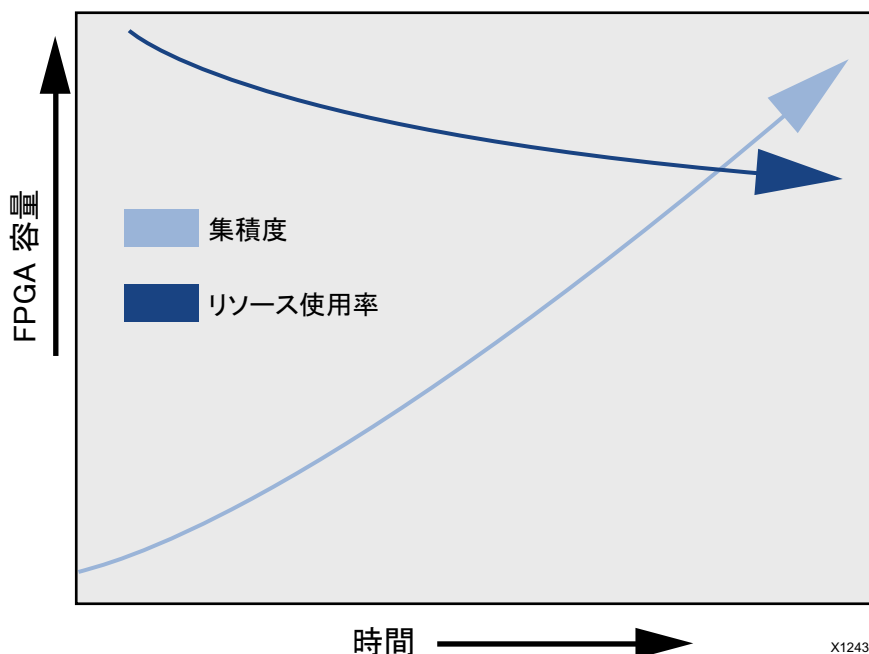


図 2-1：集積度が高いほど高いリソース使用率を達成するのに努力が必要

## 配線リソースの非効率な使用

配線リソースが非効率に使用されると、配線が密集したり、ツールでの配線の選択が制限される原因となり、次のような結果を招くことがあります。

- 配線容量の増加 (消費電力の増加)
- 遅延の増加 (パフォーマンスへの悪影響)
- デバイスにデザインを完全に配線できない (ワースト ケース)

高集積度デバイスではロジック アレイ数が多く、次のような傾向があります。

- 関連ロジック ファンクションが離れて配置される
  - 一部の信号のファンアウトがかなり大きくなる
  - 完全なデザインを実現させるにはさらに多くの配線リソースおよび配線エリア必要となる
- 配線チャンネルを使い切ってしまうと、リソース使用率は低下します。

## 配線使用率の改善

このガイドで説明する手法を利用すると、次が可能になります。

- 配線使用率を向上する
- 使用可能なロジック数が増加する
- 一部の FPGA リソースの必要性を軽減する
- アプリケーションのデバイス使用率を削減する
- 今後の拡張のためリソースに余裕を持たせる
- 集積度の低いデバイスに移行し、コスト、消費電力などを改善する

## デザイン パフォーマンス

デザイン パフォーマンスを管理していないと、次のような問題が発生することがあります。

- タイミング クロージャが困難になる
- ランタイムが長くなる
- 実行の繰り返しが増える
- デザイン パフォーマンスが低下する

大型デザインでは、次のようなさまざまな要因により、配置が最適なものにならないことがあります。

- I/O 接続数が多い
- データ バス幅が非常に大きい
- ファンアウトが大きい信号が多い
- ロジック レベル数が多すぎる

配置が最適でないと配線リソースがさらに必要になり、その結果、配線が長くなり、パフォーマンスの低下を招きます。

大型デザインは範囲やサイズが大きいいため、数え切れないほどのタイミング パスを解析し、クロージャに導く必要があります。大型デザインのパフォーマンス管理は小型デザインよりもさらに重要です。

## 消費電力

新しいザイリンクス FPGA デバイスでは、消費電力が大きく削減されています。

こうした削減がなければ、デザインの消費電力はデザイン サイズに対して過度に増加する可能性があります。消費電力を下げるには、チップ内部の放熱やチップに必要な電源量を削減する必要があります。

システム レベルで消費電力に細心の注意を払っても、デバイス レベルで確認しないと、消費電力が許容できない量に達してしまうことがあります。

このガイドの推奨事項の多くは、特定のデザインやファクションの消費電力を低減するのに役立ちます。

消費電力解析および消費電力低減手法の詳細は、[付録 A「その他のリソース」](#)にリストされている『消費電力手法ガイド』(UG786)を参照してください。

## プロジェクト コスト

大型 FPGA デザインはリソースを多く使用します。リソース使用率は、次のような固定リソースにおいて著しく高くなってしまいます。

- ルックアップ テーブル (LUT)
- フリップフロップ (FF)
- クロック リソース
- I/O コンポーネント
- RAM コンポーネント
- DSP コンポーネント

リソース使用率が高まると、さらに高集積の FPGA デバイスや、複数の高集積度 FPGA デバイスへの移行が必要になる可能性があります。

FPGA リソースが効率よく管理されていないと、大型デザインがさらに大きくなり、プロジェクトのコストが増大することになります。たとえば、大型デザインでリソースが 10% 非効率に使用されていると、小型デザインでの 10% の非効率よりもコストがかなり大きくなる可能性があります。

デザインのサイズや複雑さは、デバイスのコストだけでなく、ボードのコスト、プロジェクト スケジュール上の追加コストも含め、全体的なコストの上昇につながります。

# スタックド シリコン インターコネクト (SSI)

---

このガイドでは、高集積度 FPGA デバイスをターゲットにしたデザインについて解説します。この章では、スタックド シリコン インターコネクト (SSI) テクノロジを使用したデザインについて特に説明します。

このテクノロジは、パッシブ [シリコン インターポーザー](#) に実装された複数の [SLR \(Super Logic Region\)](#) コンポーネントをまとめるものです。

従来のデバイスと比較すると、SSI テクノロジを利用したザイリンクス FPGA デバイスには次のような特徴があります。

- 集積度が高い
- 専用機能が多い
- 消費電力が低い

注記：「従来のデバイス」および「モノリシック デバイス」という表現は、SSI テクノロジを使用していないデバイスのことを指します。

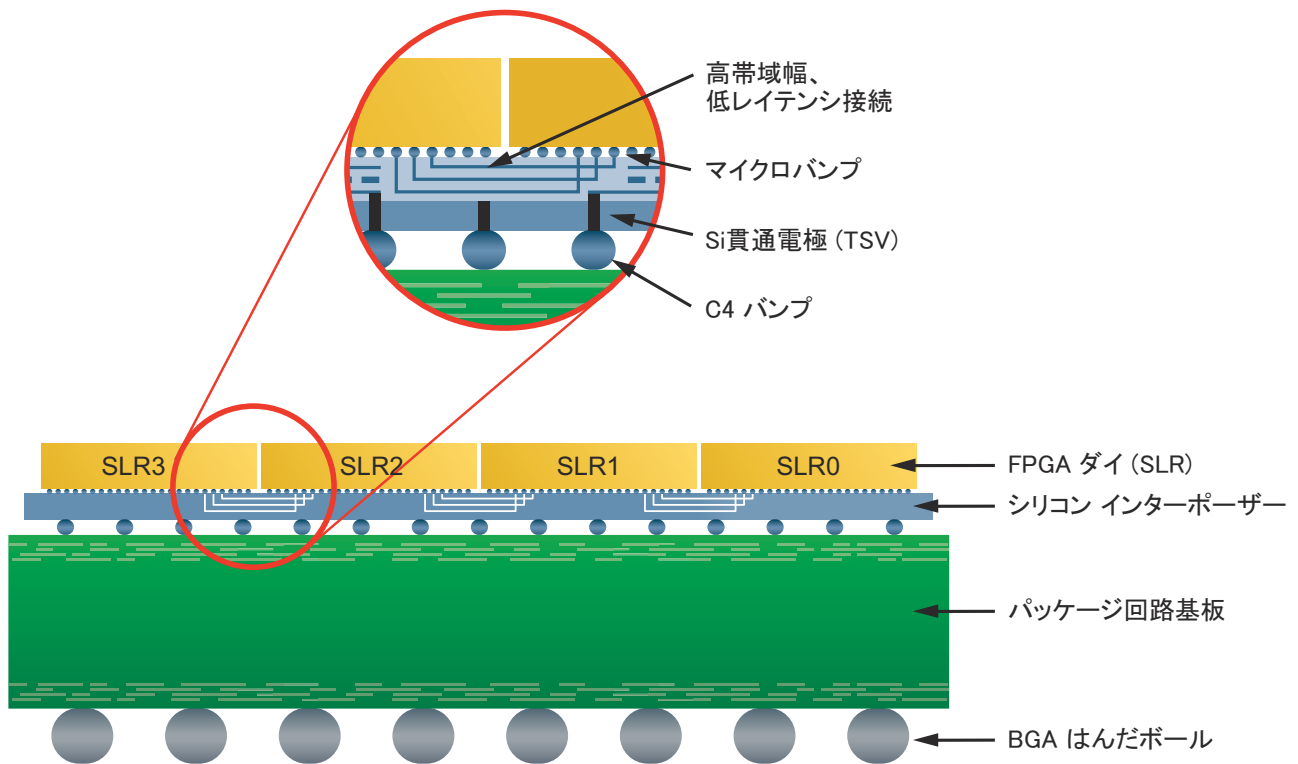


図 3-1：SSI デバイスの構造

## SSI コンポーネント

このセクションでは、次のスタックド シリコン インターコネクト (SSI) コンポーネントについて説明します。

- [SLR \(Super Logic Region\)](#)
- [シリコン インターポーザー](#)
- [SLL \(Super Long Line\) 配線](#)
- [マスター SLR \(Super Logic Region\)](#)

### SLR (Super Logic Region)

SLR (Super Logic Region) とは、SSI デバイスに含まれる FPGA ダイ スライス 1 つを指します。

#### 能動回路

各 SLR には、ほとんどのザイリンクス FPGA に共通の能動回路が含まれています。この回路には次のものが多数含まれています。

- 6 入力 LUT
- レジスタ
- I/O コンポーネント
- ギガビット トランシーバー (GT)
- ブロック メモリ
- DSP ブロック
- その他のブロック

#### SLR コンポーネント

1 つの SSI デバイスは、複数の SLR コンポーネントから構成されています。

SLR のアスペクト比は、高さよりも幅の方が広いのが一般的です。SLR コンポーネントはインターポーザー上に垂直に重ねられています。

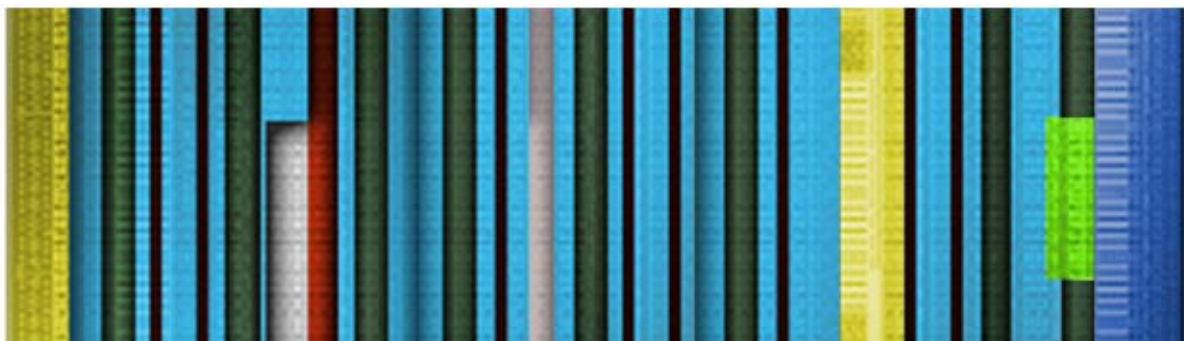


図 3-2 : SSI デバイスに含まれる SLR

SSI デバイスは、複数の SLR コンポーネントを垂直方向に重ねて作成されています。

- 一番下の SLR は SLR0
- SLR コンポーネントが垂直方向に積み重ねられるにつれ、番号が大きくなる

たとえば、XC7V2000T デバイスには 4 つの SLR コンポーネントがあります。

- 一番下の SLR は SLR0
- SLR0 のすぐ上は SLR1
- SLR1 のすぐ上は SLR2
- 一番上の SLR は SLR3

ザイリンクス ツール (PlanAhead™ デザイン解析ツールを含む) では、グラフィカル ユーザー インターフェイス (GUI) およびレポート ファイルで SLR コンポーネントが明確に識別されます。

## SLR の命名規則

次の作業を行う際、ターゲット デバイスの SLR 命名規則を理解しておくことが重要です。

- ピン選択
- フロアプラン
- タイミング レポートをはじめとするレポートの解析
- ロジックの位置およびそのロジックのソースとデスティネーションの特定

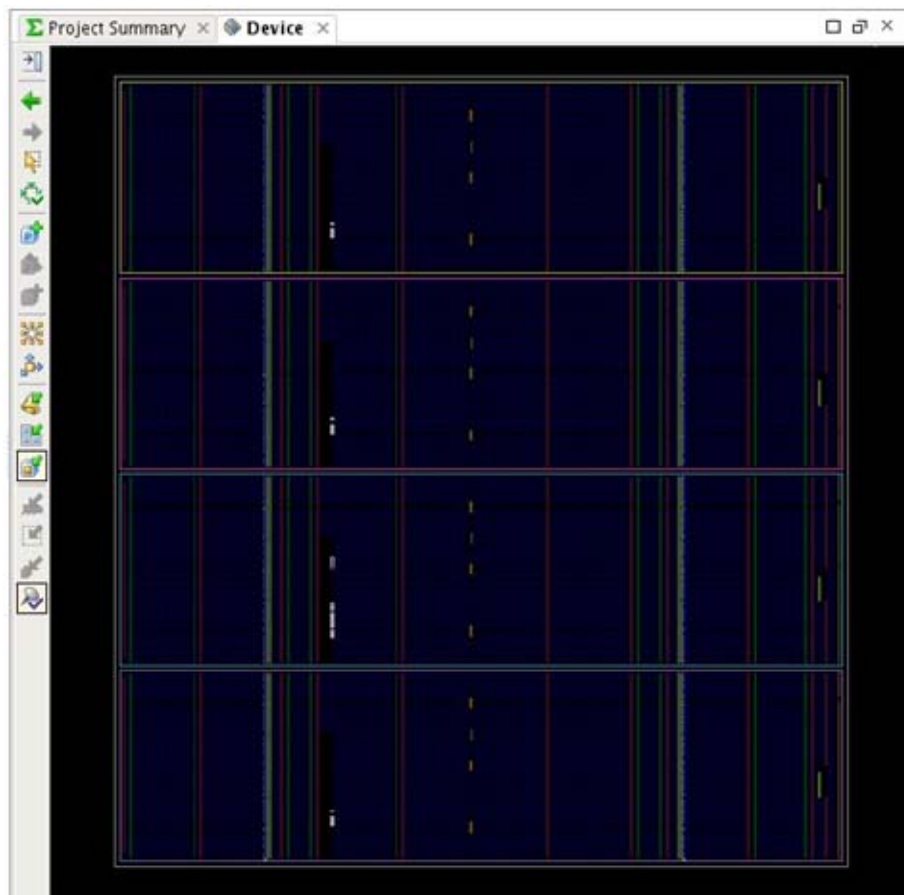


図 3-3 : Vivado ツールでの 2000T デバイスの表示



## Virtex-7 デバイス ファミリの SLR コンポーネント

Virtex®-7 デバイス ファミリでは、2 つの異なる SLR コンポーネントが使用されています。

- [xc7v2000t デバイス](#)
- [xc7vx1140t および Virtex-7 HT デバイス ファミリ](#)

### xc7v2000t デバイス

xc7v2000t デバイスでは、次のものを含む SLR が使用されます。

- 約 50 万個のロジック セル
- 含まれるコンポーネント：
  - I/O
  - ブロック RAM
  - DSP ブロック
  - GTX トランシーバー
  - その他のブロック

### xc7vx1140t および Virtex-7 HT デバイス ファミリ

xc7vx1140t デバイスおよび Virtex-7 HT デバイス ファミリでは、次のコンポーネントを含む SLR が使用されます。

- 約 29 万個のロジック セル
- GTH トランシーバー
- xc7v2000t の SLR コンポーネントよりも多数のブロック RAM および DSP コンポーネント

表 3-1：各 Virtex-7 SLR に含まれる主なリソース

	Virtex-7 T SLR	Virtex-7 XT/HT SLR
ロジック セル	488,640	284,800
スライス	76,350	44,500
ブロック RAM	323	470
DSP スライス	540	840
クロック領域/MMCM	6	6
I/O	300	300
トランシーバー	12	24
SLR 間のインターコネクト	12,864	10,560

## シリコン インターポーザー

シリコン インターポーザーは、SSI デバイスの受動層です。

このレイヤーでは、SLR コンポーネント間に次が配線されます。

- コンフィギュレーション
- グローバル クロック
- 汎用インターコネクト

シリコン インターポーザーでは、次が提供されます。

- 電源およびグランド
- コンフィギュレーション
- デバイス間の接続
- その他必要な接続

能動回路は SLR 上にあります。シリコン インポーザーは Si 貫通電極 (TSV) コンポーネントを使用してパッケージ基板に実装されています。これらのコンポーネントは、FPGA デバイスの回路をパッケージ ボールに接続します。

シリコン インポーザーは、SLR コンポーネントとパッケージ基板の間のパイプ役を果たし、次のものをデバイス パッケージに接続します。

- 電源およびグランドの接続
- I/O コンポーネント
- ギガビット トランシーバー (GT)

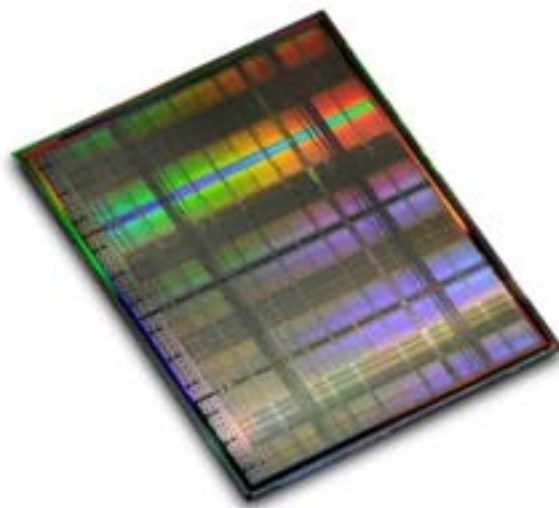


図 3-4：シリコン インターポーザー

## SLL (Super Long Line) 配線

- ある SLR から別の SLR に伝送される信号の汎用接続です。
- [シリコン インターポーザー](#)内にあります。
- SLR のインターコネクトに直接接続されているマイクロバンプにより SLR コンポーネントに接続されています。
- SLR の 12 本の垂直配線の中央に接続されます。

## Virtex-7 デバイスの SLL コンポーネント

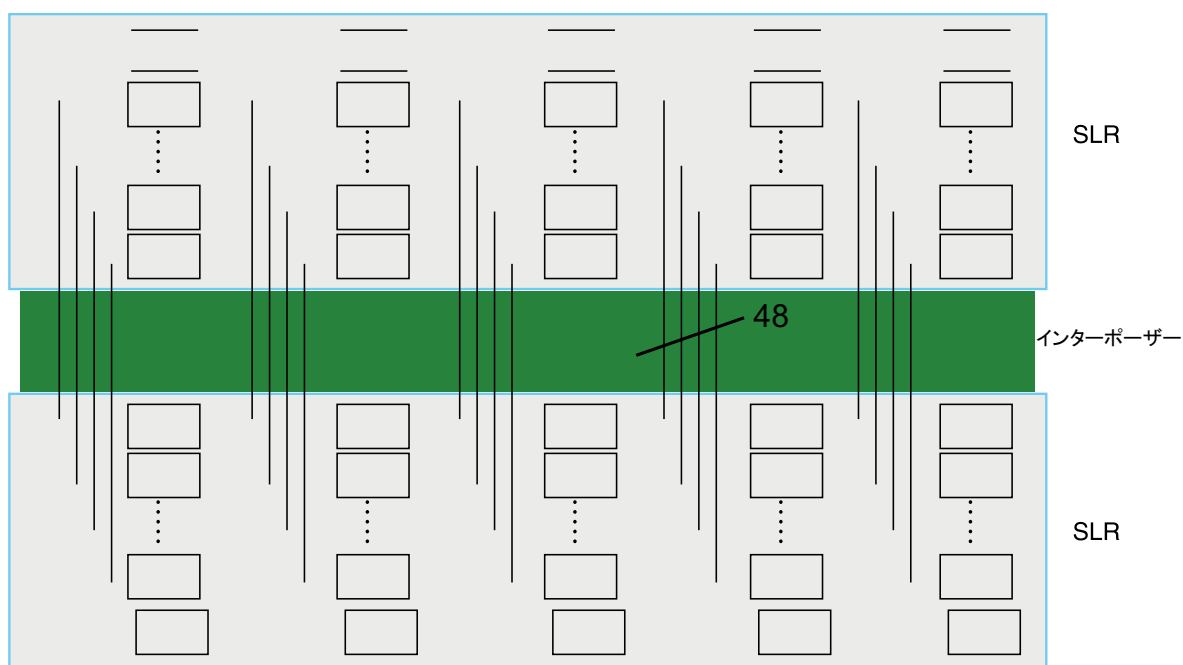
Virtex-7 デバイスでは、各 SLL コンポーネントは垂直方向に 50 個のインターコネクト タイル (50 個のスライス コンポーネントに相当) にまたがっています。これは、ザイリンクス 7 シリーズ FPGA デバイスの 1 クロック領域の高さと同じです。

つまり、SLR に隣接するクロック領域には、クロック領域のインターコネクトごとにその SLR に接続するインターコネクト ポイントが 1 つあります。

表 3-2 : 各 SLR の交差ポイントに使用される SLL コンポーネント数

Virtex-7 デバイス	SLL コンポーネント
7V2000T	13,270
7VX1140T	10,560

7VX1140T デバイスには、DSP およびブロック メモリ列が多く含まれているので、SLL コンポーネントの数が少なくなっています。これらの列では、同じエリアに対するインターコネクト タイル数が少なくなっています。



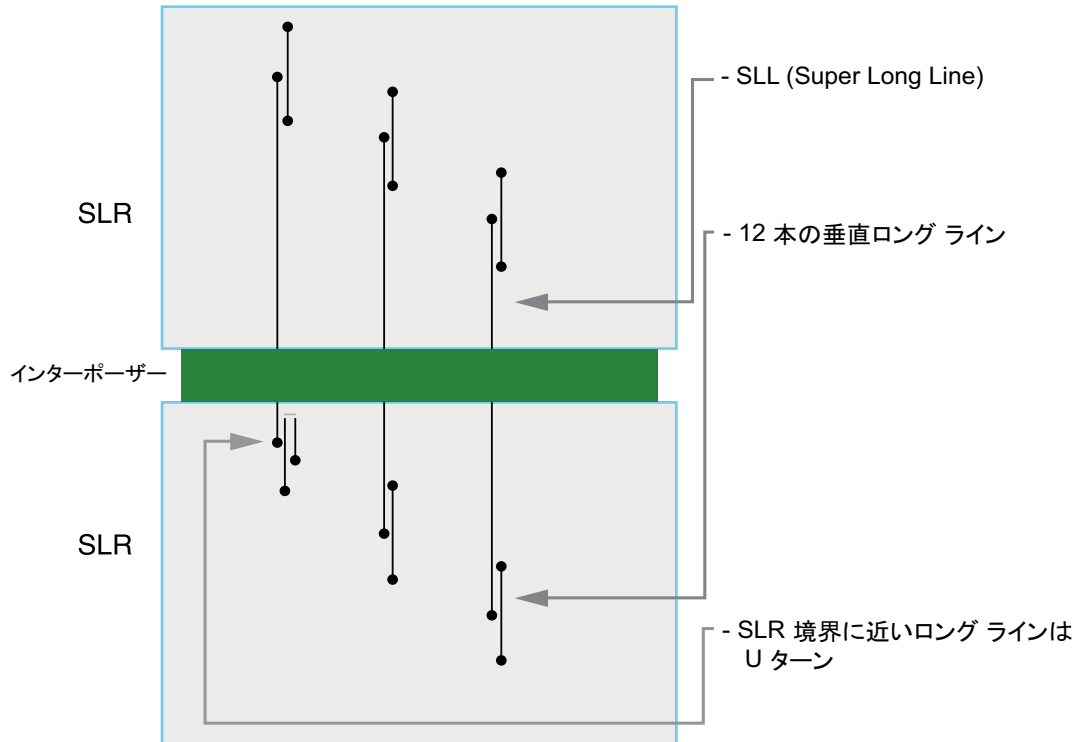
X12431

図 3-5 : SSI デバイスで少しずつずらされた SLL が交差

この SLR コンポーネント間の比率や差は図解用であり、実際の差はこれより小さくなっています。

SLL コンポーネントは、SLR の 12 個のインターコネクト タイルにまたがる 12 本の垂直ロングラインの中央で SLR に接続されます。

この接続は、SLR とその隣接する SLR との間を SLL が接続するための 3 つの最適接続ポイントとなり、パフォーマンスを低下させたり消費電力を増加させずに柔軟に配置できるようにします。



X12430

図 3-6 : SLR での SLL 接続

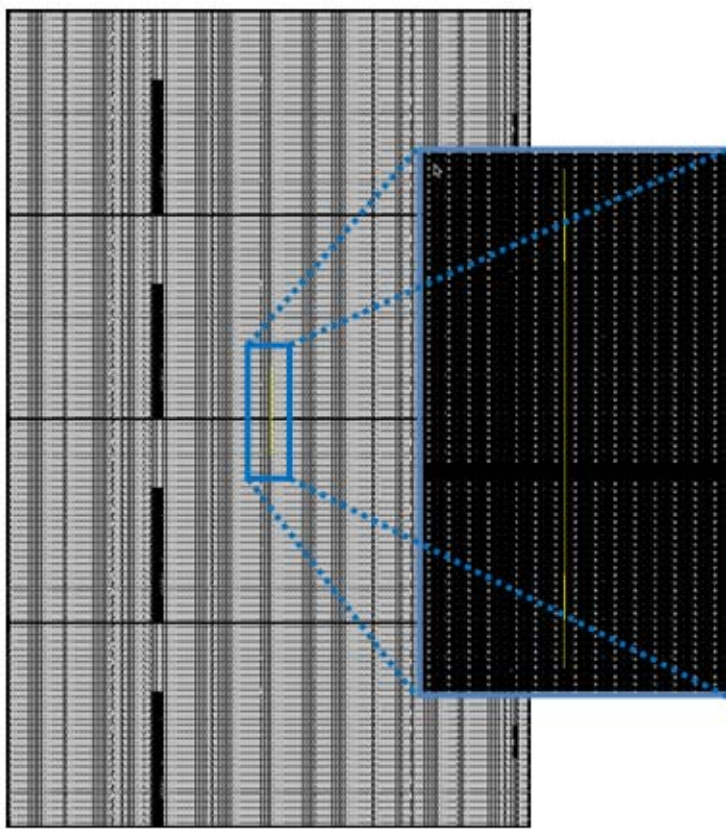


図 3-7 : FPGA Editor での SLL (ハイライト表示)

### 伝搬上の制限

SLL 信号は、SLR コンポーネント間のデータ接続のみを行います。

次のものは SLR コンポーネント間では伝搬されません。

- キャリー チェーン
- DSP カスケード
- ブロック RAM アドレス カスケード
- DCI カスケードなどのその他の専用接続

通常、伝搬上の制限はツールで自動的に考慮されますが、デザインが正しく配線されデザイン目標が満たされることを確実にするには、非常に長い DSP カスケードを構築し、SLR の境界付近にそのようなロジックを手動で配置する場合や、デザインのピン配置を指定する場合にもこの制限を考慮する必要があります。

## マスター SLR (Super Logic Region)

各 SSI デバイスには、マスター SLR が 1 つあります。すべての SSI デバイスにおいて SLR1 がマスター SLR となります。

マスター SLR には、デバイスおよびその他すべてのコンポーネントのコンフィギュレーションを開始するプライマリ コンフィギュレーション ロジックが含まれています。

マスター SLR のみに次のような専用回路が含まれています。

- DEVICE\_DNA
- USER\_EFUSE
- XADC

これらの回路にアクセスするには、デバイスにピンまたはロジックを手動で制約する際、関連ピンまたはロジックをマスター SLR に配置します。これらのコンポーネントを使用する場合、配置配線ツールで関連ピンおよびロジックを適切な SLR に割り当てることができます。通常、その他の作業は必要ありません。

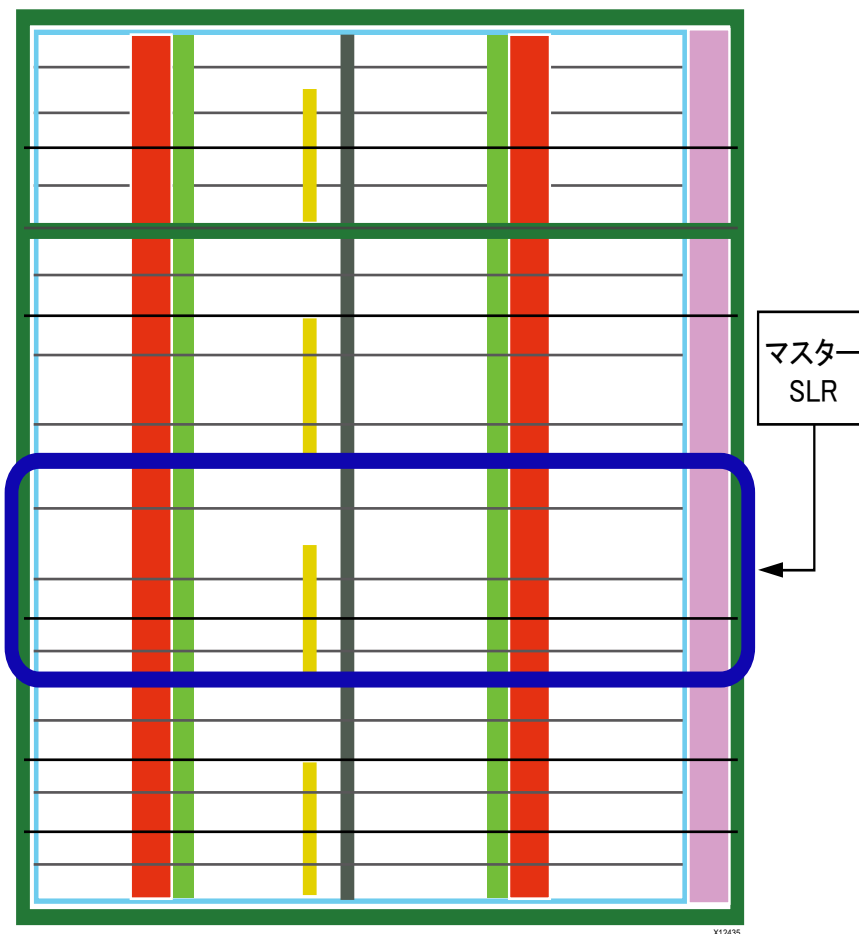


図 3-8 : xc7v2000t デバイスのマスター SLR

## クロッキング

このセクションではクロッキングについて説明します。

- リージョナル クロッキング
- グローバル クロッキング (BUFG)

### リージョナル クロッキング

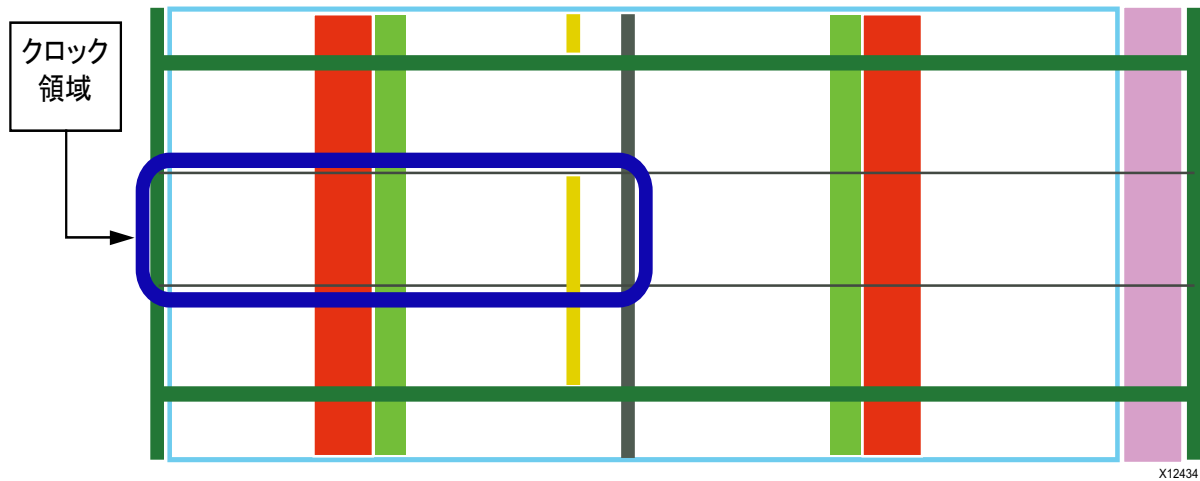


図 3-9 : SLR のクロック領域 (拡大表示)

SSI デバイスのクロック供給アーキテクチャは、ザイリンクス 7 シリーズ FPGA デバイスのものと類似しています。

次のコンポーネントのリージョナル クロッキングの接続および動作は、ザイリンクス 7 シリーズ FPGA デバイスのものと同じです。

- BUFIO
- BUFR
- BUFH

#### 例外

例外が 1 つだけあります。

BUFMR または BUFMRCE が複数の SLR コンポーネントをまたぐことはありません。

BUFMR または BUFMRCE が別の SLR と隣接する SLR のクロック領域にある場合、次の制限が適用されます。

1. BUFMR または BUFMRCE は次のものにのみアクセス可能
  - a. BUFMR または BUFMRCE が配置されているクロック領域
  - b. 同じ SLR 内の隣接するクロック領域
2. BUFMR または BUFMRCE は隣接する SLR にはアクセスできない

BUFMR または BUFMRCE は、SLR の中央クロック領域に配置するようにしてください。そうすると上下両方のクロック領域にアクセスできます。

クロック ドメインの 3 つのクロック領域すべてにアクセスする必要が今はなくても、その必要が生じたときにすべての領域にクロックを供給できるようになります。クロックおよびピン配置中にこの点を必ず考慮してください。

## グローバル クロッキング (BUFG)

SSI デバイスのグローバル クロッキング (BUFG) も、ザイリックス 7 シリーズ FPGA デバイスのものと類似しています。

- SLR でのグローバル クロッキング トポロジは同じ
- デバイス全体で使用できる BUFG コンポーネントは 32 個
- 各 BUFG コンポーネントで 1 つの SLR のクロック領域に含まれている 12 個の水平クロック (BUFH) の 1 つを駆動可能

SSI デバイスに含まれるすべての SLR コンポーネント (クロッキングを含む) は、ほかのザイリックス 7 シリーズ FPGA デバイスと同様です。

SLR の BUFG コンポーネントは、ほかの SLR コンポーネントのクロック同期エレメントにもクロックを供給できます。これは SLR 間のクロック供給の接続およびトポロジによって可能になります。SLR の 32 個の BUFG コンポーネントのそれぞれが、垂直グローバル クロッキング ライン (グローバル クロッキング バックボーンともいう) と呼ばれる 32 本の垂直ラインの 1 つを駆動します。

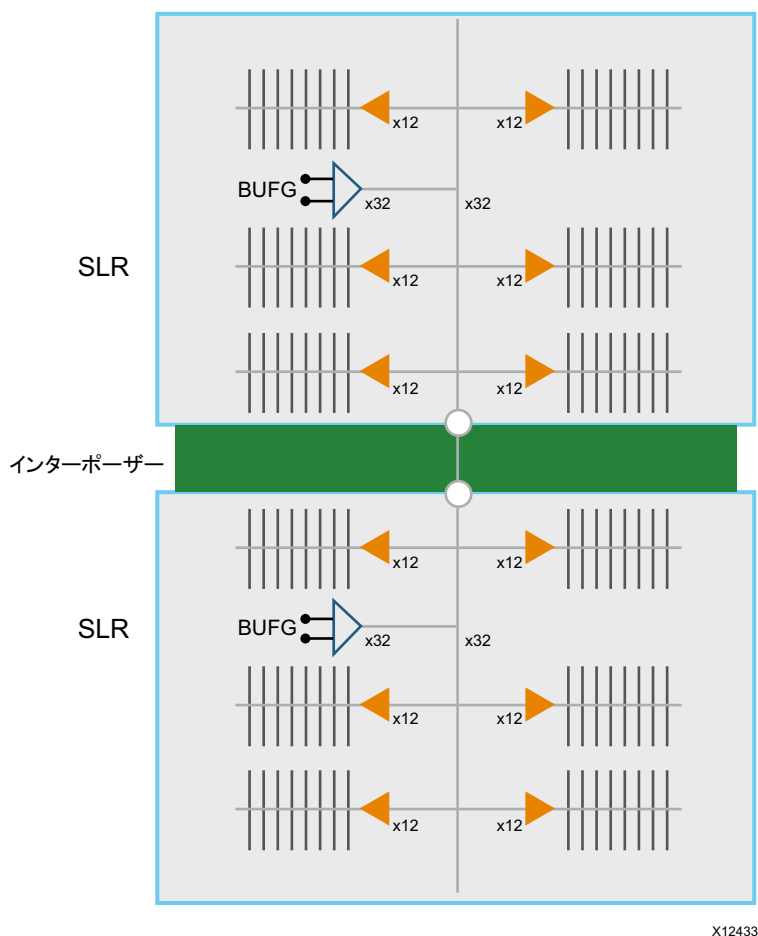
この接続は、各 BUFG の SLR の一番上と下までに到達し、水平方向のクロッキングへの接続を可能にします。

SLR の境界では、これらの垂直クロッキング スパインは非常に短いインターポーザー ホップに接続され、近接 SLR の対応するスパインに接続されます。

このプロセスは、SLR の水平クロッキング リソースを駆動し、すべての SLR コンポーネントが接続されるまで上下に継続し、真のグローバル クロッキング リソースを作成します。

垂直グローバル クロッキング ラインは各 SLR の BUFG コンポーネント間の共有リソースであるため、これらのリソースを管理することが必要になる場合があります。





X12433

図 3-10 : SSI デバイスでのグローバル クロッキング接続

この SLR コンポーネント間の比率や差は図解用であり、実際の差はこれより小さくなっています。  
詳細は、第 5 章「クロッキング」を参照してください。

## SLR コンポーネントでのデザイン配置の管理

このセクションでは、SLR コンポーネントのデザイン配置管理について説明します。

- 自動 SLR 割り当て
- 手動 SLR 割り当て

SSI デバイスは複数の SLR コンポーネントから構成されているため、多少の管理が必要になる場合があります。配線、ファンクション、タイミングに問題がないよう、デザインがデバイスに正しく配置されていることを確認する必要があります。

### 自動 SLR 割り当て

ザイリンクス ツールには、SSI デバイスのリソース配置を管理するビルトイン アルゴリズムがあります。SSI パーツにデザインがフィットするよう、ツールで SLR コンポーネントを使用してロジックが自動的に配置されます。

## 配置ストラテジ

このビルトイン配置アルゴリズムを使用して、ツールで次のような処理が実行されます。

1. SLL リソースが超過しないようにデザインを配置する
2. SLR コンポーネント間をまたぐ必要のあるタイミング クリティカル パスの数を制限する
3. SLR で特定のリソースばかりが多数使用されないようリソースのバランスを取る
4. SLL の交差を最小限に抑える

こうしたストラテジに従うことで、パフォーマンス要件を満たしながらもバランスの取れた配置が実行されます。

## SLL の交差を制限する利点

SLL の交差を制限すると、一般的に消費電力が低減され、SLR 間の接続に影響を与えることなくデザインを拡張できます。

## SLR の選択に影響するその他の要因

SLR の選択に影響を及ぼすデザインおよびインプリメンテーションの要因は、ほかにもあります。この要因には、次のようなものがあります。

1. ピン配置
2. クロック選択
3. リソース タイプ
4. フロアプラン (PBlock) および LOC 制約などの物理制約
5. タイミング制約
6. I/O 規格およびその他の制約

SLR コンポーネントは自動で割り当て、適切なピン配置、クロック選択、およびその他のデザイン選択を実行することをお勧めします。

## フロアプラン

デザインの高パフォーマンスが求められる部分にフロアプランが必要な場合があります。フロアプランは、必要な場合にのみ実行してください。

## 手動 SLR 割り当て

デザイン要件を満たすソリューションがツールで見つけれない場合や、実行間で同じ結果が得られるようにする必要がある場合は、手動 SLR 割り当てが必要になることがあります。

### 手動 SLR 割り当ての実行

手動 SLR 割り当てを実行するには、次の手順に従います。

1. 大きな PBlock (エリア グループ) を複数作成します。
2. これらのエリア グループにデザインの一部を割り当てます。

デザインの大きな部分を 1 つの SLR に割り当てるには、次の手順に従います。

1. 1 つの SLR を含むことができる PBlock を作成します。
2. この PBlock にロジックの関連階層を割り当てます。

隣接する複数の SLR コンポーネントにロジックを割り当てることはできますが、必ず PBlock に SLR 全体が含まれるようにしてください。

SLR 全体に制約を設定せずに SLR の境界を越える PBlock は作成しないでください。自動 SLR 配置アルゴリズムで有効な配置を実行するのが難しくなります。

### 手動 SLR 割り当てのガイドライン

次に、SLR コンポーネントにロジックを手動で割り当てるときの推奨事項を示します。

1. SLL リソースが超過しないようにデザインを配置する
2. SLR コンポーネント間をまたぐ必要のあるタイミング クリティカル パスの数を制限する
3. SLR で特定のリソースばかりが多数使用されないようリソースのバランスを取る
4. SLL の交差を最小限に抑える

これらのガイドラインは「自動 SLR 割り当て」の「配置ストラテジ」と同じです。これらのガイドラインに従うと、割り当てが現在および今後のデザイン ルールに違反する可能性が低くなります。

詳細は、付録 A「その他のリソース」にリストされている『フロアプラン設計手法ガイド』(UG633)を参照してください。

## SSI の階層

SSI デザインの階層が適切に定義されていると、SLR を割り当てやすくなります。信号が SLR コンポーネントの境界を越える必要がある場合は、タイミング要件を緩和させるため階層インスタンスの出力にレジスタを付けます。

次の目的のためにパーティションを使用する場合も、この推奨事項に従ってください。

- デザインの再利用
- チーム デザイン
- パーシャル リコンフィギュレーション

## SSI デバイスで高速デザインを達成

SSI デバイスで超高速デザインを達成するには、さらに手動での作業が必要となる場合があります。

SLR の速度は、Virtex-7 デバイスの同等のスピード グレードのものと同じです。

パスのロジック レベルが少ない場合やない場合は、SLR の境界を越えるときの速度を 400MHz 以上にできます。

### パイプライン処理

SLR 境界を越えるときのためにレジスタ間的高速接続を設計する場合、適切なパイプライン処理を HDL コードに記述し、合成時に制御する必要があります。

このようにすると、SLR の境界を越える必要のあるロジック パスで、シフト レジスタ LUT (SRL) の推論およびその他の最適化が実行されなくなります。

この方法でコードを変更することで SLR の境界を越える位置が定義されます。このようなデザイン変更を反映させるよう SLR 割り当てを定義する必要があります。

### フロアプラン

次が必要になる場合もあります。

- 手動のフロアプラン
- LOC 制約またはエリア グループ制約

LOC 制約およびエリア グループ制約を使用して、SLR の境界を越える場合のソースおよびDestination レジスタに制約を設定します。これで、最大速度を達成する最適な配置がツールで検出されます。

こうした手法は常に必要なものではありませんが、デバイスのパフォーマンス限界付近でピーク パフォーマンスを達成するために必要な場合があります。

## SSI コンフィギュレーション

SSI デバイスのコンフィギュレーションは、従来のデバイスのものと類似しています。ツールで 1 つのビットストリームが生成され、すべてのコンフィギュレーション機能 (暗号化や SEU など) およびコンフィギュレーション モードがサポートされています。

### コンフィギュレーションの詳細

マルチ SLR コンフィギュレーションは、コンフィギュレーション回路およびザイリンクス ツールですべて処理されます。各 SLR にコンフィギュレーション エンジンがあり、これは従来のデバイスのものと実質的には同じです。マスター SLR にはマスター コンフィギュレーション エンジンが含まれており、その他すべての SLR コンポーネントのコンフィギュレーション エンジンはスレーブとして処理されます。

ザイリンクス ツールで 1 つのビットストリームが生成されます。このビットストリームが読み込まれると、個々の SLR コンポーネントが順にコンフィギュレーションされ、ビットストリームの各部分が該当する SLR に割り当てられます。

## 結合される信号

次の信号は、インターポーザーで結合されます。

- INIT
- DONE
- KEYCLEAR

これで、「コンフィギュレーション キーのクリア」などのファンクションがすべての SLR で一度にすばやく一貫して実行されるようになります。「コンフィギュレーションの完了」や「コンフィギュレーション エラー」を通知するコンフィギュレーション フィードバックは、従来のデバイスと同じように動作します。

## ビットストリームの復号化

ビットストリームの暗号化などの操作では、すべての SLR コンポーネントで 1 つのキーが使用されます。1 つのキーを使用すると、キーおよびコンフィギュレーション データの管理が簡略化され、SSI デバイスがその他のザイリンクス FPGA デバイスと同じように表示および動作するようになります。

ビットストリームの復号化は、SLR で実行されます。インターポーザーでの SLR 間のデータ伝送は、コンフィギュレーション データを保護するために暗号化されたままになります。

## SLR ごとの操作

ほとんどの操作は従来のデバイスでのものと同じですが、次の操作は SLR ごとに実行されます。

- CAPTURE
- READBACK
- FRAME\_ECC

これにより、データの収集時間や、ECC の場合は破損データを修正する時間を短縮できます。

## マスター SLR にのみ存在するコンポーネント

コンフィギュレーションやデバイス アクセス関連のコンポーネントの一部は、マスター SLR にのみ存在します。マスター SLR でのみ使用できるコンポーネントは、次のとおりです。

- DEVICE\_DNA
- USER\_EFUSE
- XADC

バウダンリ スキャンはすべての SLR コンポーネントにありますが、マスター SLR のものが優先されます。

## パーシャル リコンフィギュレーション

デバイス コンフィギュレーションと同じように、パーシャル リコンフィギュレーションも従来のデバイスと同様に処理されます。

1. パーシャル リコンフィギュレーションを実行する方法が外部のものであっても内部のものであっても、1 つのビットストリームが生成されます。
2. マスター SLR によりビットストリームが該当する SLR コンポーネントに割り当てられます。



# システム レベルのデザイン

---

この章では、システム レベルのデザインについて説明します。

- [ピン配置の選択](#)
- [制御セット](#)
- [HDL コーディング手法](#)

クロッキングの詳細は、[第 5 章「クロッキング」](#)を参照してください。

## ピン配置の選択

このセクションではピン配置について説明します。

- [ピン配置の重要性](#)
- [ピン配置でのザイリンクス ツールの使用](#)
- [一般的なピン配置の推奨事項](#)
- [具体的なピン配置の推奨事項](#)
- [デバイスの移行](#)

### ピン配置の重要性

ピン配置が適切な場合、デザイン ロジックも正しく配置されます。

ピン配置が不適切な場合、駆動するロジックの配置オプションが限られてしまいます。配置が不適切だと配線が長くなり、配線が長いと消費電力が増えパフォーマンスが低下します。

不適切なピン配置→不適切な配置→長い配線→消費電力の増加、パフォーマンスの低下
---

不適切なピン配置が原因でこのような結果を招いてしまうのですが、これは特に高集積度 FPGA デバイスに言えることです。高集積度 FPGA デバイスは大型のチップに搭載されているので、ピン配置が離れていると、アレイ内に目的のロジック構造を構築するための関連信号の伝送距離が長くなる可能性があります。

## ピン配置でのザイリンクス ツールの使用

デザイン プランニングおよびピン配置にザイリンクス ツールを使用できます。ツールを効果的に利用するには、ツールに必要な情報を正しく入力する必要があります。

PlanAhead™ デザイン解析ツールなどのツールは、ピン配置において次のような利点があります。

- I/O 配置をグラフィカルに表示
- クロックおよび I/O コンポーネント間の関係を表示
- ピン選択を解析するデザイン ルール チェック (DRC) を提供

### 必須情報

ツールを効果的に使用するには、I/O 特性およびトポロジに関する情報をできる限り多く入力する必要があります。

次の電気的特性を指定する必要があります。

- I/O 規格
- 駆動
- スルー

また、次のような関連情報もすべて考慮する必要があります。

- クロック トポロジ
- タイミング制約

特にクロック供給に関する選択は、ピン配置に大きな影響を与えます。その逆も同じで、ピン配置はクロック供給に大きな影響を与えます。

## 一般的なピン配置の推奨事項

一般的には、ピンを選択する際は、関連信号を近くに配置し、駆動するロードの近くに配置するようにします。

ボード レイアウトや最終段階での ECO 変更時にピン配置を見直しているとき、こうした点が見過ごされがちですが、よい FPGA デザインを目指すには適切なピン配置を作成し、管理することが重要です。これは、1000 を超えるインターフェイス ピンを使用する高集積度 FPGA デザインでは特に重要です。

デバイス パッケージ上で近くに配置されているピンが、デバイス アレイでも近くにあるとは限りません。内部タイミングを満たすには、デバイス パッケージ上よりもアレイ上でピンを近くに配置することが重要です。



## 具体的なピン配置の推奨事項

次の点に関して具体的なピン配置の推奨事項があります。

- インターフェイス データ、アドレス、および制御ピン
- インターフェイス制御信号
- ファンアウトが非常に大きい、デザイン全体の制御信号
- I/O インターフェイスを含むザイリンクス IP
- CCIO および CMT の使用
- 特定 SLR にあるコンポーネント (SSI)

### インターフェイス データ、アドレス、および制御ピン

同じインターフェイス データ、アドレス、および制御ピンは同じバンクにまとめます。

- 同じバンクにこれらのコンポーネントをまとめることができない場合は、隣接するバンクにまとめる
- SSI デバイスの場合、特定インターフェイスのピンをすべて同じ SLR にまとめる

### インターフェイス制御信号

次のインターフェイス制御信号は、制御するデータ バスの中央に配置します。

- クロッキング
- イネーブル
- リセット
- ストローブ

### ファンアウトが非常に大きい、デザイン全体の制御信号

ファンアウトが非常に大きい、デザイン全体にわたる制御信号は、デバイス中央付近に配置します。

SSI デバイスの場合、駆動する SLR コンポーネント群の中央にある SLR にこの信号を配置します。

### I/O インターフェイスを含むザイリンクス IP

MIG (Memory Interface Generator) などの I/O インターフェイスを含むザイリンクス IP では、インターフェイスを生成し、推奨されるピン配置を使用します。

### CCIO および CMT の使用

デバイス上部および下部にある BUFG コンポーネントへバランスよくアクセスするには、デバイスの上下で CCIO および CMT をバランスよく使用します。

SSI デバイスの場合、ほかの SLR コンポーネントと比較し、1 つの SLR の上下で CCIO コンポーネントまたは CMT コンポーネントをバランスよく使用します。

### 特定 SLR にあるコンポーネント (SSI)

SSI デバイスの場合、特定 SLR にあるコンポーネントのピン配置のプラン段階で、同じ SLR にピンを配置します。

たとえば、外部インターフェイスの一部としてデバイス DNA 情報を使用するとき、DEVICE\_DNA が存在するマスター SLR にそのインターフェイスのピンを配置します。

## デバイスの移行

多くのザイリンクス デバイスでは、同じパッケージでサイズの異なるデバイスにデザインを移行させることができます。リスクを抑えてデザインを移行させるには、デザイン プロセスの初期段階で次の点に注意してください。

- デバイスの選択
- ピン配置の選択
- デザインの条件

同じパッケージでサイズの異なるデバイスに移行する場合は、次の点に注意してください。

- ピン配置
- クロッキング
- リソースの管理

## 制御セット

このセクションでは制御セットについて説明します。

- [制御セットとは](#)
- [リセット](#)

### 制御セットとは

制御セットとは、特定の RAM またはレジスタを駆動する制御信号をまとめたものです。制御信号には、次のものがあります。

- セット/リセット
- クロック イネーブル
- クロック

制御信号の組み合わせそれぞれに対し制御セットが 1 つ作成されます。

1 つのスライスにあるレジスタは共通の制御信号を使用するため、共通制御セットを持つレジスタのみを同じスライスにパックできます。

制御セットが複数あるデザインには、次のような特徴があります。

- デバイス使用率が低い
- 配置オプションが少ない

これが原因で、消費電力が増加し、パフォーマンスが低下する可能性があります。

制御セットが少ないデザインの場合、配置オプションが多くなり柔軟性も増すので、通常結果が良くなります。

## リセット

このセクションではリセットについて説明します。

- [リセットとは](#)
- [リセットを使用するタイミングと場所](#)
- [推論された同期エレメントの初期ステートの定義](#)
- [同期リセットと非同期リセット](#)
- [アクティブ High およびアクティブ Low のリセット](#)

### リセットとは

リセットは、最も一般的で重要な制御信号の 1 つです。リセットを管理しないと、パフォーマンス、エリア、消費電力に悪影響を及ぼすことがあります。

同期コードが推論されると、LUT やレジスタだけでなく、レジスタ以外の多くのデザイン エレメントが使用される可能性があります。汎用同期コードにより、シフト レジスタ LUT (SRL)、ブロックまたは LUT メモリ、DSP48 レジスタなどのリソースが使用されることもありますが、リセットの使用方法がこれらのコンポーネントの選択に影響し、リソースが最適に使用されなくなることがあります。

- アレイにリセットを誤って配置すると、ブロック RAM コンポーネント 1 つが推論されるはずが、数千個のレジスタが推論されてしまうことがあります。
- 遅延ラインに不要なリセットが記述されていると、数個の SRL LUT ですむところが数百個のレジスタが使用されることがあります。
- 乗算器の入力または出力に非同期リセットが記述されていると、DSP ブロックではなく、スライスにレジスタが配置されることがあります。

こうした点は、次のものに大きく影響します。

- リソース
- 消費電力
- パフォーマンス

### リセットを使用するタイミングと場所

デバイスの初期化には、リセットは必要ありません。

FPGA デバイスには、専用グローバル セット/リセット信号 (GSR) があります。デバイスのコンフィギュレーションの最後に GSR が自動的にアサートされて、すべてのレジスタが HDL コードで指定されている初期ステートに初期化されます。

初期ステートが指定されていない場合は、通常、論理 0 (ゼロ) がデフォルト値になります。HDL コードで指定されているリセット トポロジにかかわらず、コンフィギュレーションの最後に各レジスタは既知のステートになります。デバイスの初期化のためだけにグローバル リセットのコードを記述する必要はありません。

### リセットの使用の制限

リセットの使用を制限すると、次のことが可能になります。

- リセット ネットのファンアウトを制限
- リセットを配線するのに必要なインターコネクトの数を低減
- リセット パスのタイミングを単純化
- パフォーマンスおよび消費電力を改善

### リセットの必要性の決定

リセットが必要かどうかを決定するには、注意が必要です。

#### 1. 各同期ブロックの見直し

リセットが必要かどうかが不明な場合は、リセットを記述しないでください。

#### 2. 論理シミュレーションの使用

論理シミュレーションが正しく動作する場合は、インプリメントされたデザインでも正しく動作するはずです。

### リセットが記述されていない場合

ロジックにリセットが記述されていない場合、ロジックをマップする FPGA リソースの選択肢が大きく広がります。

たとえば、単純な遅延ラインに対しリセットが記述されていると、共通のリセットを持つレジスタセットにマップされる可能性が高くなります。

リセットがない場合、同じロジックは次のものにマップできます。

- シフト レジスタ LUT (SRL) 1 つ
- 1 つの SRL と複数のレジスタの組み合わせ
- すべてのレジスタ
- LUT またはブロック メモリ

その後、次の点を考慮して、合成ツールでそのコードに最適なリソースが選択されます。

- 機能
- パフォーマンス要件
- 使用可能なデバイス リソース
- 消費電力

### 推論された同期エレメントの初期ステートの定義

GSR は、すべてのレジスタを HDL コードで指定された初期値に初期化します。

- 初期値が指定されていない場合、合成ツールによって初期ステートに 0 または 1 が割り当てられます。
- 初期ステートのデフォルト値は通常 0 ですが、ワンホット ステート マシン エンコーディングなど、いくつかの例外があります。

推論されたシフト レジスタ LUT (SRL)、メモリ、その他の同期エレメントにも、コンフィギュレーション時に関連エレメントにプログラムされる初期ステートを定義できます。

すべての同期エレメントを正しく初期化することをお勧めします。レジスタの初期化は、主な FPGA 合成ツールであれば完全に推論できます。コンフィギュレーション後、FPGA デバイスでは同期エレメントがすべて既知の値で開始します。

既知の値で初期化する利点は、次のとおりです。

- 初期化のためだけにリセットを追加する必要をなくす
- RTL コードとインプリメントされたデザインが一致しやすくなる
- 論理 (RTL) シミュレーションが既知のステートになる

## 同期リセットと非同期リセット

リセットが必要な場合、同期リセットを記述することをお勧めします。

同期リセットには、非同期リセットよりも多くの利点があります。同期リセットは、FPGA アーキテクチャの多くのリソース エレメントに直接マップできます。

DSP48 やブロック RAM などのコンポーネントの場合、ブロックのレジスタ エレメントには同期リセットしかありません。

これらのエレメントに関連したレジスタ エレメントに非同期リセットを使用すると、機能を変更しなければレジスタをこれらのブロックに直接推論できません。

たとえば 7v2000t デバイスの場合、非同期リセットが記述されていないければ、約 650,000 個の DSP レジスタおよび 93,000 個のブロック RAM レジスタを使用できます。これらのレジスタでは同期リセットのみがサポートされています。

また、集積度が高かったり、詳細に調整された配置が必要な場合、同期リセットを使用すると柔軟に制御セットをマップし直すことができます。

## アクティブ High およびアクティブ Low のリセット

配線および配置に最大限の柔軟性を持たせるため、次のようなファンアウトの大きい制御信号では極性を混合しないでください。

- クロック イネーブル
- リセット

アクティブ High を使用することをお勧めします。通常、アクティブ High を使用すると、FPGA アーキテクチャにマップするときに必要なロジック数およびロジック レベル数が少なくなります。

### 極性

デザインを通して一定した極性を使用することは、さらに重要です。混合制御セットを使用すると、配置および配線に悪影響が出て、ワースト ケースで FPGA デバイスにタイミングおよびフィットに関する問題が発生することがあります。

## Virtex-6 および Virtex-7 デバイス

Virtex-6 および Virtex-7 デバイスのスライスおよび内部ロジックでは、クロック イネーブルおよびリセットはすべてアクティブ High です。

アクティブ Low のリセットまたはクロック イネーブルを記述すると、これらの配線に単純インバーターとして LUT が追加される可能性があります。

## HDL コーディング手法

HDL コードを適切に記述すると、次のような利点があります。

- 合成ツールでの処理が容易になる
- シミュレーションおよび合成が高速になる
- FPGA アーキテクチャ間での移植性が高まる
- ターゲット アーキテクチャに変換しやすくなる
- 解読およびデバッグしやすい

HDL コードを適切に記述するには、次の点を考慮します。

- デバイス リソースへ効率よく推論される HDL 構文を使用する
- 次の点に注意する
  - ロジック
  - クロッキング
  - 制御信号
- 適切な階層を使用する
- 合成属性を注意して使用する

### デバイス リソースへの推論

ターゲット アーキテクチャの主要な演算、ストレージ、およびロジック エレメントを考慮することが必要です。

デザインのコードを記述するときは、合成マッピングを理解し予測することで、潜在的な問題を回避できます。

次のガイドラインは、RTL コードがザイリンクス FPGA リソースにどのようにマップされるかを示します。

#### 4 ビットを超える加算、減算、加減算

4 ビットを超える加算、減算、加減算では、一般的にキャリー チェーンが使用され、2 ビットの加算ごとに LUT が追加で 1 つ使用されます。

たとえば、8 ビット X 8 ビットの加算器には LUT が 8 個と関連キャリー チェーンが使用されます。

3 値加算の場合 (または加算器の結果がレジスタを介さずに別の値に追加される場合)、3 ビットごとに LUT が 1 個使用されます。たとえば、8 ビット X 8 ビット X 8 ビットの加算にも LUT が 8 個と関連キャリー チェーンが使用されます。

複数の加算が必要な場合は、加算 2 段ごとにレジスタを指定すると有利になる可能性があります。これにより 3 値インプリメンテーションの生成が可能になるので、デバイス使用率を半分に削減できます。

## 乗算

乗算には、通常 DSP ブロックが使用されます。

- ビット幅が 18 X 25 未満の符号付き乗算は、1 つの DSP ブロックにマップされます。
- ビット幅 17 X 24 未満の符号なし乗算は、1 つの DSP ブロックにマップされます。
- 積が大きい乗算の場合は、複数の DSP ブロックにマップされる場合があります。

DSP ブロックに推論されるロジックに対しパイプライン処理を行うと、パフォーマンスおよび消費電力を大きく改善できます。乗算器を記述するときは、その周辺に 3 段のパイプライン処理を行うことで、クロック周波数、セットアップ、clock-to-out、および電力特性において最善な結果を得ることができます。

パイプラインの段数が非常に少ないと (1 段またはなし)、これらのブロックにタイミングの問題が発生したり、消費電力が増加したりすることがあります。

## シフト レジスタまたは遅延ライン

リセットまたは複数のタップ ポイントが不要なシフト レジスタまたは遅延ラインは、通常、シフト レジスタ LUT (SRL) コンポーネントにマップされます。

次のものを 1 つの LUT にマップできます。

- ワード数が 16 ビット以下の SRL コンポーネント 2 つ
- ワード数が 32 ビットまでの SRL コンポーネント 1 つ

SRL コンポーネントを最大限に利用するには、これらのブロックのリセット仕様に注意してください。リセットが不要な場合、デバイス使用率、パフォーマンス、消費電力において望ましい結果が得られる可能性があります。

## 64 ビットまでのメモリ アレイ

ワード数が 64 ビットまでのメモリ アレイは、通常 LUT RAM コンポーネントにインプリメントされます。

- ワード数が 32 ビット以下の場合、1 つの LUT に 2 ビットがマップされます。
- ワード数が 64 ビットまでの場合、LUT ごとに 1 ビットをマップできます。

これよりワード数が大きい RAM も、次の条件しだいで LUT RAM にインプリメントできます。

- 使用可能なリソース数
- 合成ツールでの割り当て

これらのブロックの記述の際にコーディング スタイルが少し異なっていただけでも、間違ったリソースが使用されることがあります。たとえば、アレイの値を変えてしまう非同期の書き込みやリセットを記述すると、LUT RAM ではなく、1 つのレジスタ アレイにインプリメントされる可能性があります。

## ワード数が 256 ビットよりも大きいメモリ アレイ

ワード数が 256 ビットよりも大きいメモリ アレイは、通常ブロック メモリにインプリメントされます。Virtex-6 デバイスおよびザイリンクス 7 シリーズ FPGA デバイスは、さまざまな幅とワード数の組み合わせでメモリをマップする柔軟性を備えています。これらのコンフィギュレーションを理解しておく、と、大型メモリ アレイの宣言に使用されるブロック RAM コンポーネントの数や構造を理解しやすくなります。

これらのブロックの記述の際にコーディング スタイルが少し異なっただけでも、間違ったりリソースが使用されることがあります。たとえば、アレイをリセットする非同期の読み出しやリセットを記述すると、LUT RAM や複数のレジスタ アレイではなく、1 つのレジスタ アレイにインプリメントされる可能性があります。

### 標準マルチプレクサになる条件文

表 4-1：標準マルチプレクサになる条件文

マルチプレクサ	インプリメント先	ロジック (LUT) レベル数
4:1	• LUT 1 個	1
8:1	• LUT 2 個 • MUXF7 1 個	1
16:1	• LUT 4 個 • MUXF7 と MUXF8 リソースの組み合わせ	1

このコードを理解しておくと、リソースをより適切に管理できるようになり、デザインのデータパスのロジック レベルを理解および制御するのに役立ちます。

### 汎用ロジック

汎用ロジックの場合は、レジスタを介した出力 1 つに対する入力数を考慮します。この数に基づいて、LUT 数およびロジック レベル数を予測できます。

- 入力数が 6 以下の場合、ロジック レベル数は 1
- 入力数が 11 以下の場合、ロジック レベル数は 2 またはそれ以下

入力数が多く、論理式が複雑であるほど、より多くの LUT およびロジック レベルが必要になります。

初期段階でロジック レベル数を考慮しておくと、タイミング クロージャ中ではなく早期に簡単に変更できます。

### 適切なデザイン階層の選択

デザイン階層は、次のものによりある程度定義されます。

- デザインの論理セクション
- コアまたは IP の使用
- 以前のコードの階層定義



## デザイン階層のガイドライン

デザイン階層のガイドラインは、次のとおりです。

- データパスの出力にレジスタを付ける
- 階層の上位にクロック エレメントを配置
- I/O コンポーネントを推論

### データパスの出力にレジスタを付ける

可能であれば、大型の階層モデル、特に階層の上の方にあるものにレジスタを付けます。

特に次の理由で階層の境界が最適化されない場合に、タイミングを向上させることができます。

- 階層デザイン手法
- フロアプラン
- デバッグ

この手法では、クリティカルパスが複数の階層をまたぐことはなく、モジュール内またはモジュール間の境界に含まれます。クリティカルパスが複数の階層をまたいでいると、タイミングまたは機能に問題が発生した場合に、解析や修正が難しくなります。

### 階層の上位にクロック エレメントを配置

クロック エレメントを上位階層に配置すると、モジュール間でクロックを共有しやすくなり、リソースを管理しやすくなります。また、必要なクロック リソースの数が少なくなり、パフォーマンスおよび消費電力が改善されることもあります。

### I/O コンポーネントを推論

可能な場合は I/O コンポーネントを推論します。

- インスタンス化が必要がときは、コードの上位に I/O コンポーネントを配置します。
- 階層デザインおよびパーシャル リコンフィギュレーションの場合は、最上位またはその付近に I/O コンポーネントを配置します。
- 回路が階層の上の方にあると、I/O の問題をデバッグしやすくなります。

## 階層デザイン

境界を適切に定義してデザインを分割しておくと、次のような階層デザイン設計手法を利用しやすくなります。

- パーティション
- パーシャル リコンフィギュレーション
- チーム デザイン

チーム デザインの場合は、このような分割が必須条件になることがよくあります。

次のような要因の影響を受けやすい箇所は、分割しておくことが必要となる場合があります。

- タイミング クロージャ
- 論理デバッグ

階層を適切に定義しておく、フロアプランしやすくなり、後でタイミング クロージャを達成しやすくなります。階層レベルでクリティカルパスがまとめられていると、フロアプランが簡単になります。

SSI デバイスの場合は、個々の SLR コンポーネントにロジックを割り当てる必要があることがあります。

- すべてのロジックが SLR に割り当てられるように階層を定義しておく、このような割り当てを簡単に作成できます。
- すべてのロジックが SLR に割り当てられるように階層を定義しておく、配置配線が早く完了します。
- すべてのロジックが SLR に割り当てられるように階層を定義しておく、デザインを解析しやすくなります。

## 論理およびタイミング デバッグ

クリティカル タイミング パスが論理デザイン部分に限定されていると、タイミング デバッグしやすくなります。

ザイリンクス ツールでは階層別にゲート レベルのタイミング ネットリストを書き出すことができます。これにより、大型 FPGA デザインのタイミングおよび論理デバッグが簡単になります。

## パイプライン処理

タイミング クロージャを制限する主な要因は、2 つあります。

- 不適切な配置が原因で配線が長くなりすぎ、タイミング制約を満たすことができない
- ロジック レベル数が多すぎ、デザインが目的の速度で動作しない

パイプライン処理には、次のような利点があります。

- ロジック レベル数を削減してタイミング要件を満たします。
- 初期段階にパイプライン処理を計画しておく、タイミング クロージャが簡単になります。  
一部のパスにパイプラインを追加すると、回路全体にレイテンシの差が伝搬されることがあります。この若干の変化のために、コードを部分的に再設計しなければならないことがあります。
- 設計の初期段階にパイプライン処理できる箇所を確認しておく、次の利点があります。
  - タイミングクロージャを達成しやすくなります。
  - インプリメンテーション ランタイムを短縮できます (タイミングの問題が修正しやすくなるため)。
  - デバイスの消費電力を低減します (ロジック切り替えを低減できるため)。

## ファンアウトが大きい非クロック ネットの管理

ファンアウトが大きいネットの管理は、デザイン プロセスの早い段階で行うのが簡単です。ネットのファンアウトが大きくなるかどうかは、パフォーマンス要件およびパスの構成によって決まります。

ロードが数千あるネットを確認し、デザインへの影響を予測します。

## ファンアウトの大きいネットの管理

ファンアウトが大きいネットを判別できたら、次のテクニックを利用します。

- [ロードの低減](#)
- [BUFG および BUFH の使用](#)
- [ツールでの複製の管理](#)

### ロードの低減

デザインのロードを必要としない部分にロードを移動させます。

- ファンアウトが大きい制御信号の場合、コード記述されているすべての箇所で、正しく機能するためにそのネットが必要かどうかを確認します。ロードの要求を抑えると、タイミングを大きく向上させることができます。
- データパスの場合は、ロジックを制限することでファンアウトを抑えることができるかどうかを確認します。

### BUFG および BUFH の使用

ネットを駆動するには、BUFG および BUFH コンポーネントを使用します。

詳細は、[第 5 章「クロッキング」](#)の「[非クロック ネットにクロック バッファを使用](#)」を参照してください。

### ツールでの複製の管理

ツールでの複製を管理します。

ほとんどの合成ツールには、ファンアウトの大きい信号の影響を抑えるため、レジスタまたはロジックの複製を自動的にまたは手動で管理する機能があります。

この機能を利用すると、場合によってはまったく RTL コードを変更せずに、ファンアウトの大きいネットのタイミングを大幅に改善できます。



# クロッキング

---

合成の初回実行前、または HDL コードの最初の 1 行を書き始める前であっても、デザインにおける決定事項がデザイン目標に大きく影響を与えます。初期段階から賢明に計画し、少し時間をかけることで、正しい判断ができ、プロジェクトの時間を短縮できます。

## クロック リソースの選択

最初のステップの 1 つとして、ピン配置を選択する前に、クロック リソースを選択しておくことを推奨します。クロック供給の選択によって、特定のピン配置やロジックの配置が決まることがあります。適切なクロック供給を選択すると、より良い結果が得られます。

Virtex®-6 および Virtex-7 には、BUFG というグローバルクロックバッファが 32 個含まれています。

BUFG を使用すると、次の点に関して要件がそれほど厳しくないデザインのクロック要件のほとんどを満たすことができます。

- クロック数
- デザイン パフォーマンス
- 低消費電力
- クロック特性：
  - クロック ゲーティング
  - マルチプレクシング
  - その他のクロック制御

BUFG コンポーネントは、合成で簡単に推論でき、制限事項が少ないので、最も汎用のクロック供給が可能です。

ただし、クロック要件が BUFG の機能を上回る場合や、より良いクロック特性が求められる場合、ザイリンクスでは次を推奨します。

1. 使用可能なクロック リソースに対してクロック ニーズを解析する
2. タスクに最善のリソースを選択し制御する

## グローバル クロッキング

グローバル クロック バッファには、クロック供給以外にも機能があります。この追加機能には、デザイン コードを手動で編集するか、または合成中にアクセスできます。

グローバル クロック バッファには、次のものが含まれています。

- [BUFGCE](#)
- [BUFGMUX](#)
- [BUFGCTRL](#)
- [IP および合成](#)

### BUFGCE

BUFGCE プリミティブを使用すると、グリッチのない同期クロック イネーブル (ゲーティング) 機能にアクセスできます。追加のロジックやリソースは使用されません。

BUFGCE は、一定期間クロックを停止する場合や、周波数の高いベース クロックから 2 分周または 4 分周したクロックなど、ロー スキューおよび低消費電力のクロックを作成する場合に使用します。

これは、回路動作の異なる時間に異なる周波数を生成する場合に特に有益です。

### BUFGMUX

BUFGMUX は、グリッチまたはタイミングの問題を発生させずにクロック ソースを別のものに変更する場合や、異なる時間または動作条件に対して異なるクロック周波数を生成する場合に使用します。

### BUFGCTRL

グローバル クロック ネットワークのすべての機能にアクセスするには、BUFGCTRL を使用します。

このクロック バッファでは、クロック切り替え回路が損失したり停止した場合など、クロック供給のシナリオが複雑な場合に、クロック供給を非同期に制御できます。

### IP および合成

IP および合成でも、これらの高度なクロック供給機能を使用できます。

たとえば、MIG (Memory Interface Generator) を使用する場合、I/O での高速データ伝送およびキャプチャ用に特別なクロック バッファを使用できます。

クロック供給を作成および計画する際は、個々の IP に使用されるクロック リソースを考慮に入れておく必要があります。

ただし、通常、目標のクロッキング動作を得るには、コードでコンポーネントをインスタンス化して正しい接続を設定しておく必要があります。

詳細は、[付録 A 「その他のリソース」](#)にあるクロッキングに関するガイドおよびライブラリ ガイドを参照してください。

## リージョナル クロッキング

このセクションでは、リージョナル クロッキングについて説明します。

- 水平クロック領域のバッファ (BUFH、BUFHCE)
- リージョナル クロック バッファ (BUFR)
- I/O クロック バッファ (BUFIO)
- マルチリージョナル クロック バッファ (BUFMR)

### 水平クロック領域のバッファ (BUFH、BUFHCE)

水平クロック領域のバッファ (BUFH、BUFHCE) は、次のように使用できます。

- BUFG と併用
- スタンドアロン バッファとして使用

#### 水平クロック領域のバッファの使用

水平クロック領域のバッファは、クロック供給およびクロックに接続されている関連ロジックの配置をより厳しく制御する場合や、クロック ドメイン数が多いデザインに追加のクロック リソースを提供する場合に使用します。

BUFH および BUFHCE リソースを使用すると、クロック領域に接続されているグローバル クロック ネットワーク (BUFG) 部分をデザインで使用できます。これにより、グローバル クロック ネットワークの未使用部分のロー スキュー リソースを複数の小さなクロック ドメインに使用して、これらのドメインを 1 つのクロック領域に制約できます。

#### グリッチなしのクロック イネーブル

BUFHCE には、BUFGCE と同じグリッチなしのクロック イネーブルがあり、特定のクロック ドメインでクロックを簡単かつ安全にゲート処理できます。

BUFH を BUFG ネットワークとは別に使用するときは、BUFH は BUFHCE の単純なバッファとなり、同じリソースになります。

BUFH は、クロック イネーブルが不要な場合に使用することをお勧めします。

#### 中粒度クロック ゲーティング

BUFHCE が BUFG により駆動されている場合は、中粒度クロック ゲーティングとして使用できます。

BUFHCE は、クロック供給を断続的に停止する必要があるクロック ドメインの一部 (ロードが数百から数千の範囲) に対して有効なクロック リソースとなります。

BUFG は、同じまたは異なるクロック領域の BUFH コンポーネントを複数駆動でき、複数のロー スキュー クロック ドメインでクロック供給を個別に制御できます。

#### BUFG から独立させて BUFH および BUFHCE を使用

BUFH および BUFHCE は、クロック機能を追加するため、BUFG から独立させて使用することもできます。

独立させて使用する場合、BUFH に接続されているロードはすべて同じクロック領域にある必要があります。これは非常に高速で細粒度の (ロードが少ない) クロッキングに適しています。

次の点を必ず確認してください。

1. BUFH によって駆動されるリソースがクロック領域で使用可能な数を超えていない
2. ほかの競合がない

BUFH とクロック ドメイン間の位相関係は、クロック ドメインが何によって駆動されているかによって (BUFG コンポーネント、ほかの BUFH コンポーネント、またはその他のクロック リソース)、異なる可能性があります。

ただし、2 つの BUFH コンポーネントが水平方向に隣接している領域を駆動している場合は例外です。この場合、両方の BUFH コンポーネントが同じクロック ソースにより駆動されていれば、左右のクロック領域間のスキューには厳しく制御された位相関係があるはずなので、2 つの BUFH クロック ドメインをまたいでデータを転送できます。

## リージョナル クロック バッファ (BUFR)

リージョナル クロック バッファ (BUFR) は通常、低速 I/O およびデバイス クロックとして、より高速の I/O データのキャプチャおよび供給に使用されます。

### リージョナル クロック バッファの使用

リージョナル クロック バッファ (BUFR) は、クロックをイネーブルまたはディスエーブルにしたり (ゲート処理)、公約数によるクロック分周を実行するために使用します。

Virtex-6 デバイスでは、BUFR は次のものを駆動します。

- BUFR のあるクロック領域
- その上下クロック領域

### 中粒度クロック ゲーティング

BUFR は、中粒度クロック供給に適しています。Virtex-7 デバイスでは、BUFR は BUFR が含まれる領域のみを駆動できます。このため、規模がやや小さいクロック供給ネットワークに適しています。

BUFR の速度は BUFG や BUFH よりも遅いため、超高速のクロック供給には BUFR は使用しないでください。BUFR は中速から低速のクロック供給に適しています。

ビルトインのクロック分周機能を使用する場合、BUFR は高速 I/O インターフェイス クロックなどの外部クロック ソースからの分周クロック ネットワークに適しています。

## I/O クロック バッファ (BUFIO)

I/O クロック バッファ (BUFIO) は、次の目的に使用します。

- I/O データを入力ロジックにキャプチャする
- 出力クロックをデバイスの出力ロジックに供給する
- 高速のソース同期データをバンクでキャプチャする
- BUFR と、ISERDES または OSERDES を共に使用しているときにデータを扱いやすい速度にする



BUFIO は、次のような ILogic および OLogic にある入力および出力コンポーネントのみを駆動できます。

- IDDR
- ODDR
- ISERDES
- OSERDES
- 単純な専用入力または出力レジスタ

BUFIO を使用するときは、I/O ロジックとデバイスとの間でデータが確実に転送されていることを必ず確認してください。

## マルチリージョナル クロック バッファ (BUFMR)

マルチリージョナル クロック バッファ (BUFMR) は、ザイリックス 7 シリーズ FPGA デバイスの新機能です。Virtex-6 およびそれ以前のデバイスでは使用できず、また不要です。

BUFMR を使用すると、1 つのクロック ピン (MRCC) で次の場所にある BUFIO および BUFR を駆動できます。

- BUFMR のあるバンク
- その上下の I/O バンク (適宜)

SSI デバイスをターゲットにしている場合は、[第 3 章「スタックド シリコン インターコネクト \(SSI\)」](#)を参照してください。

## SSI デバイスのクロッキング

通常、標準のクロック供給ルールが SSI デバイスにも適用されます。ただし、SSI デバイスをターゲットにしている場合は、さらに注意点がいくつかあります。

BUFMR が SLR の境界を越えてクロック リソースを駆動できない点を除き、リージョナル クロッキングは同じです。

BUFMR を駆動するクロックは、SLR の中央にあるバンクまたはクロック領域に配置することをお勧めします。これで、SLR の 3 つのクロック領域すべてにアクセスできます。

## 16 個以下のグローバル クロックが必要なデザイン

16 個以下のグローバル クロック (BUFG コンポーネント) が必要なデザインの場合、グローバル クロッキングに関して特に注意事項はありません。BUFG コンポーネントは、競合を避けるため、ツールにより自動的に割り当てられます。

## 17 以上 32 個未満のグローバル クロックが必要なデザイン

17 以上 32 個未満の BUFG コンポーネントが必要な場合、グローバル クロッキング ラインの競合、クロック ロードの配置、またはその両方が原因で発生する可能性のあるリソースの競合を避けるため、ピンの選択および配置を検討する必要があります。

ほかのザイリックス 7 シリーズ FPGA デバイスと同様、クロック兼用 I/O (CCIO) コンポーネントおよびその関連クロック マネージメント タイル (CMT) には、SLR で駆動できる BUFG コンポーネントに関する制限があります。

SLR の上半分にある CCIO コンポーネントは SLR の上半分にある BUFG コンポーネントのみを駆動でき、SLR の下半分にある CCIO コンポーネントは 同じく下半分にある BUFG コンポーネントのみを駆動できます。したがって、すべての SLR コンポーネントの上半分または下半分で 17 個以上の BUFG コンポーネントが必要とならないように、ピンおよび関連 CMT を選択する必要があります。

すべてのクロックで競合なくすべての SLR コンポーネントを駆動できるように、ツールによって BUFG コンポーネントすべてが自動的に割り当てられます。

## 33 個以上のグローバル クロックが必要なデザイン

33 個以上のグローバル クロックが必要なデザインの場合は、小さなクロック ドメインに対し BUFR および BUFH を使用することを推奨します。これで、グローバル クロック ドメインの数を削減できます。

BUFMR を使用する BUFR コンポーネントは、1 つの SLR の半分を網羅する 3 つのクロック領域にあるリソースを駆動できます。これは、Virtex-7 デバイス クラスの SLR 1 つにある約 250,000 個のロジック セルに相当します。

水平方向に隣接するクロック領域では、左右両方の HROW バッファをロー スキューで駆動し、1 つの SLR の 3 分の 1 にあたるクロック ドメインをイネーブルにできます。これは約 167,000 個のロジック セルに相当します。

これらのリソースを使用すると、次のことが可能になります。

- クロック リソースの競合を避けるための注意点が少なくなる
- デザイン配置を改善する
- パフォーマンスおよび消費電力を改善する

## BUFG グローバル クロッキング スパインの分割

グローバル クロックが 33 個以上必要な場合、BUFG グローバル クロッキング スパインを分割できます。

垂直グローバル クロック ラインの SLR コンポーネント周縁には絶縁バッファがあり、同じ垂直グローバル クロック ラインの別の SLR にある 2 つの BUFG コンポーネントを競合なしに使用できます。

33 個以上のグローバル クロックが必要な場合でも、通常はザイリンクス ソフトウェアにより BUFG コンポーネントの配置と割り当てが自動的に制御されます。ただし、複雑な状況では、BUFG コンポーネントと駆動されるロジックを手動で配置する必要がある場合もあります。

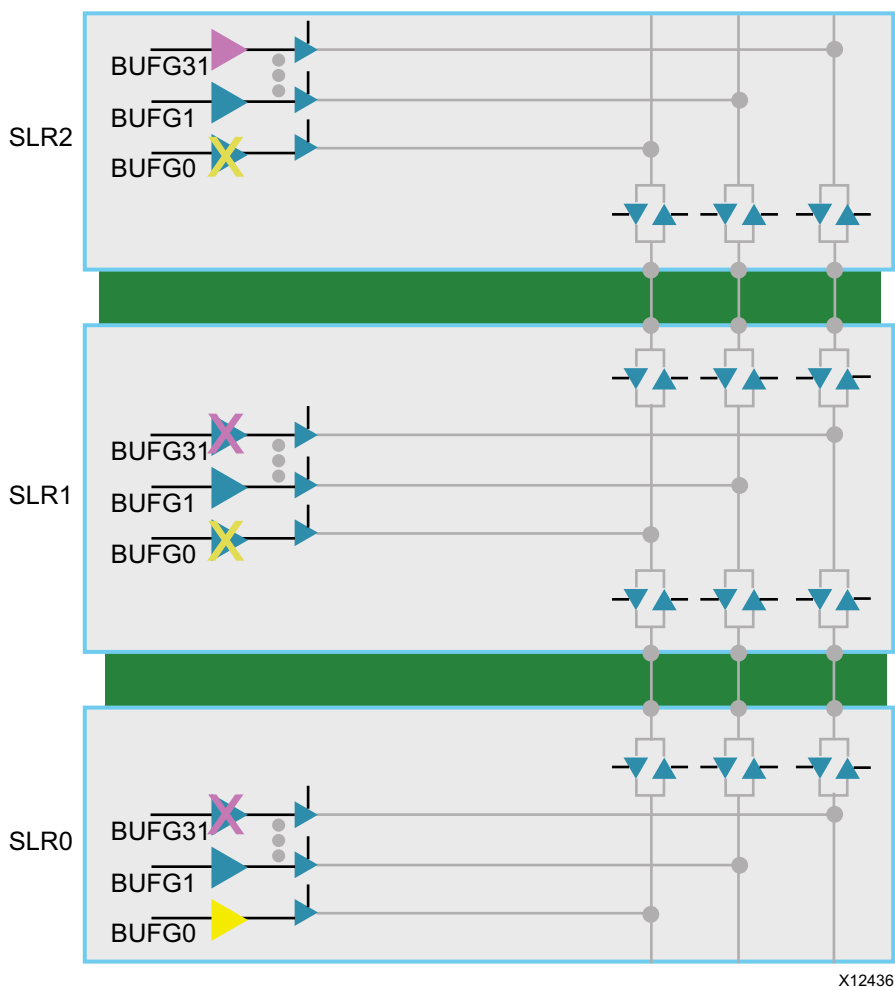


図 5-1 : SSI グローバル クロッキングの分配と共有リソース

## SSI デバイスのクロック スキュー

高集積度 FPGA デバイスのクロック スキューは、パスのタイミングの大部分を占めます。

クロック スキューを管理せずに放置しておくと、過度のスキューが原因で次の面で問題が発生する可能性があります。

- 最大クロック速度
- ホールド タイム

この点に関しては SSI デバイスもほかの高集積度 FPGA デバイスと同じです。スキューの要因は、次のとおりです。

- ソースからデスティネーションまでのクロック ネットワークでのクロック分配が均一でない
- プロセス、電圧、および温度 (PVT) のわずかなばらつきが原因で、データ パスに対しクロック パスの一部の速度が速くなったり遅くなったりする

## マルチダイ

1 つのデバイスに含まれるマルチダイ SLR では、PVT 式のプロセス部分が大きくなります。

マルチダイ SLR はザイリンクス アセンブリ プロセスによって管理され、同様のスピードのダイのみが一緒にパッケージされます。

ザイリンクス タイミング ツールでは、タイミング レポートの一部としてこれらの差が考慮されます。パス解析中、セットアップおよびホールド計算の一部としてこのような点が解析され、指定要件に対するパス遅延の一部としてレポートされます。

SSI デバイスの場合、追加の計算や注意点はありません。タイミング解析ツールで処理されます。

### パス遅延の差

上または下の SLR を使用している場合、パス遅延の差も大きくなる可能性があります。2 点間の距離が大きければ、スキューが大きくなる可能性があります。

このため、複数の SLR を駆動する必要があるグローバル クロックは、中央の SLR に配置してください。これで、パーツ全体でクロック ネットワークのクロック分配がさらに均一になり、クロック スキューを低減できます。

## デザインでのクロッキングの指定

このセクションでは、デザインでのクロッキングの指定について説明します。

- [推論](#)
- [合成属性](#)
- [IP](#)
- [インスタンスエーション](#)

### 推論

特に指定がない限り、合成ツールにより、アーキテクチャで使用可能な数までのクロックにグローバル バッファ (BUFG) が自動的に指定されます。

BUFG は、ほとんどのクロッキング ニーズに適した制御の行き届いたロー スキューのネットワークを提供します。パーツで使用可能な BUFG コンポーネントの機能や数を超えるクロッキングでない限り、これ以外のものは必要ありません。

クロック供給構造を制御すると、次の点において良好な特性を得ることができます。

- ジッター
- スキュー
- 配置
- 消費電力
- パフォーマンス

## 合成属性

合成属性を正しく使用すると、クロック リソースを簡単に制御できます。

### 合成属性の使用

- BUFG が推論されないようにすることができます。
- BUFG をほかのクロック供給構造に置き換えることができます。
- 合成属性で指定しなければ挿入されないクロック バッファを指定できます。
- HDL ソース コードを変更せずにクロック リソースを制御できます。

### XST での BUFFER\_TYPE 属性の使用

XST (Xilinx Synthesis Technology) ツールで BUFFER\_TYPE 属性を次の目的に使用できます。

- クロック バッファを指定しない
- BUFGH や BUFR などの特定バッファ タイプを指定する

BUFFER\_TYPE 属性は、次のように指定できます。

- HDL ソース コードに直接記述  
これで、コード内で制約が有効になります。
- XST 制約ファイル (XCF) で指定  
HDL ソース コードを変更せずにクロック リソースを制御できます。

## IP

一部の IP を使用してクロック供給構造を作成できます。

- [Clocking Wizard および I/O Wizard](#)
- [複雑な IP](#)

### Clocking Wizard および I/O Wizard

Clocking Wizard および I/O Wizard を使用して、次のクロック リソースおよびその構造を作成できます。

- BUFG
- BUFIO
- BUFR
- 次のようなクロック調整ブロック：
  - MMCM (Mixed Mode Clocking Manager)
  - I/O 位相ロック ループ (PLL)

### 複雑な IP

次のような複雑な IP には、クロック供給構造が含まれていることがあります。

- Memory Interface Generator (MIG)
- PCIe
- Transceiver Wizard

こうした IP は、十分に配慮すれば、追加クロック リソースとして使用できます。

適切な配慮なしで使用すると、デザインのほかの部分のクロック オプションを制限してしまうことがあります。

インスタンス化された IP に対しては、デザインのほかの部分のクロック要件、機能、リソースを活用するようにしてください。

## インスタンス化

HDL デザインにクロック リソースをインスタンス化するのが、最も具体的で直接的なクロック供給構造制御方法です。

- [インスタンス化の利点](#)
- [最上位でのクロック リソースのインスタンス化](#)
- [重複クロック リソースのリスク](#)

### インスタンス化の利点

HDL デザインにクロック リソースをインスタンス化すると、デバイスのすべての機能にアクセスし、制御できます。

一般的に、インスタンス化は次の追加ロジックおよび制御が必要なクロック供給構造の唯一のオプションです。

- BUFGCE
- BUFGMUX
- BUFHCE

単純なバッファであっても、デザインにそれをインスタンス化するのが一番簡単です。

### 最上位でのクロック リソースのインスタンス化

クロック リソースは、コードの最上位またはその近くで、個別のエンティティまたはモジュールに配置することをお勧めします。

最上位またはその近くにあるエンティティやモジュールは、デザイン内の複数のモジュールに簡単に分配できます。

### 重複クロック リソースのリスク

- FPGA リソースが無駄に使用されます。
- 消費電力が増加します。
- 次のような結果の原因となる競合や配置が作成されます。
  - インプリメンテーション ツールのランタイムが長くなる
  - タイミングが複雑になる

## クロック位相、周波数、デューティ サイクル、およびジッターの制御

このセクションでは、クロック位相、周波数、デューティ サイクル、およびジッターの制御について説明します。

- [クロック調整ブロックの使用](#)
- [位相制御のための IDELAY の使用](#)
- [ゲーテッド クロックの使用](#)
- [ダイナミック消費電力の削減](#)

### クロック調整ブロックの使用

MMCM または PLL を使用すると、入力クロックの特性を調整できます。

MMCM はクロックの挿入遅延を削除し、入力システム同期データにクロックの位相を揃える際に一般的に使用されます。

MMCM は、次の目的にも使用できます。

- 位相をさらに厳しく制御する
- クロックのジッターをフィルターする
- クロック周波数を変更する
- クロック デューティ サイクルを修正または変更する

これで、デザインの重要な部分を厳密に制御できます。

MMCM および PLL コンポーネントの使用は、クロック リソースの調整および制御には非常に一般的です。

MMCM または PLL を使用するには、MMCM が仕様どおりに動作し、目的のクロック特性が出力されるように、いくつかの属性を設定する必要があります。

### Clocking Wizard

これらのリソースをコンフィギュレーションするには、Clocking Wizard を使用することを推奨します。

### ダイレクト インスタンス化

MMCM または PLL は直接インスタンス化することもできます。

直接インスタンス化することでリソースは制御しやすくなりますが、目標の結果が得られるようにし、さらにタイミング目標も満たすことができるようにするため、正しい設定が使用されていることを確認する必要があります。

PLL での設定が間違っていると、ジッターが大きくなってクロックのばらつきが増加したり、位相関係が不正になったり、タイミングを不必要に困難にする問題が発生したりすることがあります。

## 位相制御のための IDELAY の使用

若干の位相調整のみが必要な場合は、MMCM および PLL の代わりに IDELAY または ODELAY を使用できます。

IDELAY または ODELAY を使用すると、遅延を追加したり、関連データに対してクロックの位相オフセットを増加したりできます。

## ゲーテッド クロックの使用

FPGA デバイスには、ファンアウトが大きく、ロー スキューのクロック リソースを供給できる専用クロック ネットワークがあります。HDL コードで細粒度クロック ゲーティング テクニックを使用すると、この機能および専用リソースへのマッピングに問題が発生する可能性があります。

FPGA デバイスを直接ターゲットにするコードを記述する場合は、クロック パスにクロック ゲーティング構文を記述しないでください。代わりに、クロック イネーブルを推論するコード記述を使用してクロッキングを制御してください。これで、機能または消費電力削減の目的でデザインの一部を停止することができます。

コードにクロック ゲーティング構文が既に含まれている場合、またはそのようなコードが必要な別のテクノロジーを使用している場合、クロック パスに配置されたゲートをデータ パスのクロック イネーブルにマップし直すことができる合成ツールを使用してください。このようにすると、クロック リソースへのよりよいマッピングが可能になり、ゲーテッド クロック ドメインに入出力するデータの回路のタイミング解析が簡単になります。

クロック ネットワークの大部分を一定期間シャットダウンさせる場合、次のコンポーネントのいずれかを使用してクロック ネットワークのオン/オフを切り替えることができます。

- BUFGCE
- BUFHCE
- BUFR
- BUFMRCE

クロックが一定期間遅くなる場合、次のコンポーネントのいずれかと追加のロジックを使用して、クロック ネットを周期的にイネーブルにすることができます。

- BUFGCE
- BUFHCE
- BUFR

または、BUFGMUX を使用して高速クロック信号から低速のクロックにクロック ソースを切り替えることもできます。

## ダイナミック消費電力の削減

上記のどの手法を使用しても、ダイナミック消費電力を効果的に削減できます。どの手法が最も効果的かは、要件およびクロック トポロジによって異なります。

- BUFR
- BUFMRCE
- BUFHCE
- BUFGCE



## BUFR

BUFR は、次の条件を満たす場合に最も効果的です。

- 外部で生成されたクロックである
- 200MHz 未満
- 最高 3 つのクロック領域にクロックを供給する必要がある

## BUFMRCE

複数のクロック領域 (垂直方向に隣接する領域の場合は 3 つまで) でこの手法を使用するには、Virtex-7 デバイスで BUFMRCE が必要な場合があります。

## BUFHCE

BUFHCE は、1 つのクロック領域に含むことができる高速クロックに最も適しています。

## BUFGCE

BUFGCE はデバイス全体で使用でき、最も柔軟性がありますが、消費電力を削減するには適切ではありません。

# 出力クロック

外部クロック供給デバイスを使用している場合、FPGA デバイスからクロックを出力するには ODDR コンポーネントを使用するのが効果的です。

位相関係およびデューティ サイクルが適切に制御されるクロックを作成するには、次のように設定します。

1. 1 つの入力を High にする
2. もう 1 つの入力を Low にする

クロックを停止し、一定期間ある極性に保持するには、セット/リセットおよびクロック イネーブルを使用します。

外部クロックの位相制御がさらに必要な場合は、MMCM または PLL を使用して次のいずれか、または両方の機能を使用します。

- 外部フィードバック補正
- 位相補正:
  - 粗粒度または細粒度
  - 固定または可変

これで、クロック位相およびほかのデバイスへの伝搬時間を制御でき、デバイスからの外部タイミング要件を簡略化できます。

## クロック ドメインの交差

プロセス、電圧、温度 (PVT) は変動するため、1 つのクロック ドメイン内よりも 2 つのクロック ドメインをまたがる場合の方がクロック および位相のばらつきが増加します。

2 つのクロック ドメインをまたがると、セットアップおよびホールド 計算に加算され、低速のデザインでもタイミングを満たすのが困難になることがあります。

デザインの箇所によって異なるクロック 周波数が必要な場合、クロック を次のように制御します。

1. クロック ドメイン数を定義する
2. 汎用クロック イネーブルか、次のコンポーネントのいずれかを使用する
  - BUFGCE
  - BUFGMUX
  - BUFHCE
  - BUFR
  - BUFMRCE

クロック が同期していない、または各エッジで揃っていない場合は、複数サイクル信号が正しく転送されていることを確認します。非同期信号 (位相関係が既知の関係ではない) の場合は、特別な手法を使用してドメイン間でデータが正しく確実に転送されるようにする必要があります。

## 同期ドメインの交差

同期ドメインとは、次のようなドメインを指します。

- クロック 間の位相関係が既知のものである
- 位相関係がインプリメンテーション実行ごとに変化しない

同期ドメインは、次の場合に発生します。

- 互いのドメインから派生している
- 同じ内部または外部ソースから供給されている
- この両方の条件が当てはまる

このようなケースでは、データを解析することが可能で、ある程度の注意を払えば、ドメイン間で問題なくデータを転送できます。

## クロック スキュー

共通ノード、およびソースおよびデスティネーションに到達するまでバッファ内を通過する距離によりですが、クロック スキュー 計算は重要です。

レジスタ間のパスなど小さなデータ パスの場合、クロック スキュー はデータ遅延よりも大きくなる可能性があります。これを修正しないとホールド タイムに影響します。

ロジック レベルがいくつかある場合は、追加スキューが発生してタイミングを満たすのが困難になる可能性があります。こうしたクロック が交差する状況ではロジック レベルをよく監視し、ロジック レベル数が多すぎる場合や少なすぎる場合の影響を考慮してください。

### 同期ドメイン交差の例

- チップの同じ側 (上または下) に 2 つの BUFG コンポーネントがあり、同じ MMCM、PLL、またはデバイスピンから駆動される BUFG ネットワークからもう一方の BUFG ネットワークへの交差
  - 水平方向に隣接する BUFH ネットワークからもう一方の BUFH ネットワークへの交差
  - 同じ BUFMR によって駆動されていて、BUFR から同様にコンフィギュレーションされているもう 1 つの BUFR への交差
- BUFR コンポーネントが BYPASS モードでない場合、すべての関連 BUFR コンポーネントでリセットを同期させて両方のコンポーネントの位相を揃える必要もあります。
- 同じクロックソースの同じクロック領域にある BUFIO と BUFR 間の交差

### 非同期ドメインの交差

非同期ドメインを交差する場合、不正なバスキャプチャやメタステーブル状態など、バスのデータインテグリティに影響する要因をできる限り取り除く必要があります。

一般的に、データが非同期クロックドメインを問題なくまたぐことができるようにするには、次の 2 つの方法があります。

- 1 ビットのみが必要な場合、または関連データを 2 ビット以上転送するのにグレイコーディングなどの方法を使用する場合、回路の平均故障間隔 (MTBF) を抑えるためにレジスタ同期化回路を挿入します。
- 複数ビット (バス) の場合、独立クロック (非同期) FIFO を使用してドメイン間のデータ転送を行います。

このような FIFO がソフトロジックから作成されている場合は推論できます。専用ハード FIFO の使用が望まれる場合や、特性化済みまたは定義済みの FIFO ロジックを使用した方が目的を達成しやすい場合は、FIFO を FIFO プリミティブからデザインに直接インスタンスシートしたり、CORE Generator™ の FIFO Generator を使用して FIFO を作成できます。

### 非同期ドメイン交差の例

- 位相関係がないクロックネットワークから別のクロックネットワークへの交差
- MMCM または PLL を使用しているクロックネットワークから MMCM または PLL を使用していないクロックネットワークへの交差
- BUFH ネットワークからこのネットワークが水平方向に隣接していない別のネットワークへの交差
- 専用クロックリソース (外部クロックピン、MMCM、PLL など) によって直接駆動されていないクロックネットワーク間の交差
- デバイスの上半分にある BUFG から下半分にある BUFG への交差

## デバイス スタートアップの制御および同期化

FPGA デバイスのコンフィギュレーションが完了すると、デバイスはコンフィギュレーション ステートから一般操作の状態になります。

ほとんどのコンフィギュレーション シーケンスで、最終ステップの 1 つは次のようになります。

1. グローバル セット/リセット (GSR) のディアサート
2. 続いて、グローバル イネーブル (GWE) のディアサート

ディアサートされると、デザインは既知の初期ステートになり、操作できるようになります。

### 未知のステート

解放ポイントが指定クロック ドメインに同期していない場合や、クロックが GWE が安全に解放できる速度よりも速いスピードで動作している場合、デザインの一部が未知のステートになることがあります。

デザインの中には未知のステートになっても問題ないものもありますが、通常デザインが未知のステートになると、不安定になったり、初期データ セットが不正に処理される可能性があります。

### 既知のステート

デザインが既知のステートにならない場合、スタートアップ同期化プロセスを制御する必要があります。

#### 単一クロック ドメイン

デザインにクロック ドメインが 1 つのみある場合、コンフィギュレーション後シーケンスをシステム クロックに同期化させることができるので、デザインに同期化して解放できます。

コンフィギュレーション後シーケンスをシステム クロックに同期化するには、次のようにします。

1. STARTUP コンポーネントをインスタンスエートします。
2. システム クロックを CLK ピンに接続します。

これで、この同じクロックによりコンフィギュレーション シーケンスが駆動され、デバイス スタートアップの同期化が簡単になります。

#### その他のケース

その他のケースでは、GWE がアサートされた後一定期間すべてのデザイン クロックを遅延させることができます。

デザイン クロックを遅延させるには、次のようにします。

1. 次のコンポーネントをインスタンスエートします。
  - BUFGCE
  - BUFHCE
  - BUFR
2. これらのコンポーネントのイネーブルを使用し、コンフィギュレーション後に数クロック サイクル間クロック供給を遅らせます。

MMCM の場合、フィードバック クロックではなく出力クロックでこの設定を行います。

## 別の方法

ステート マシンなどデザインのクリティカルな部分で次のものを使用する方法もあります。

- クロック イネーブル
- ローカル リセット

これでクリティカルな部分のスタートアップが制御され既知の状態になります。

## 非クロック ネットにクロック バッファーを使用

クロック バッファーはクロックを効果的に供給できるよう、均一でロー スキューの信号を供給します。クロッキングにクロック バッファーが不要な場合は、ファンアウトの大きな信号の追加配線リソースとして使用できます。

クロック バッファーを非クロック信号に選択する場合は、次の点に注意してください。

- [デザイン パフォーマンス](#)
- [非クロック信号に 3 つ以上の BUFG または BUFH コンポーネントを使用](#)
- [混合極性信号に BUFG コンポーネントを使用](#)
- [イネーブル信号を効果的に使用](#)
- [バッファーの選択](#)
- [バッファー配置の指定](#)

## デザイン パフォーマンス

グローバル クロック バッファーはロー スキューになるよう設計されており、必ずしも伝搬遅延が短いわけではありません。目的のクロック周波数、およびタイミング パスによっては、BUFG を使用するとタイミングを満たすことができない可能性があります。

## 非クロック信号に 3 つ以上の BUFG または BUFH コンポーネントを使用

BUFG で非クロック信号を駆動する場合、その信号がデスティネーションに到達するためにグローバル クロック ネットから抜ける必要があります。

- 2 つ以下の BUFG コンポーネントからの信号がグローバル クロック ネットの 1 つのポイントから抜ける場合は競合はなく、どのような状況でもデザインを配線できます。
- 3 つ以上の BUFG コンポーネントからの信号が同じロケーションに送信される場合、競合が発生してデザインを配線できなくなる可能性があります。

配線競合を避けるため、3 つ以上の BUFG コンポーネントを非クロック信号に使用しないでください。

次の場合にも同じ制限事項が適用されます。

- 非クロック信号で同じクロック領域にある 2 つの BUFH コンポーネントを駆動する場合
- 同じクロック領域にある同じロジックを駆動する必要がある BUFH コンポーネントと BUFG コンポーネントの組み合わせを駆動する場合

## 混合極性信号に BUFG コンポーネントを使用

BUFG コンポーネントは、次の信号には使用しないでください。

- 混合極性信号
- ネットの大部分に追加ロジックが必要な信号

たとえば、次のようなファンアウトの大きなリセット信号があるとします。

- デザインの一部でアクティブ High
- デザインのほかの部分でアクティブ Low

この場合、アクティブ Low の部分には反転を実行するための LUT またはロジックが必要なので、BUFG を挿入しても利点がないどころか、悪影響が及ぶ可能性があります。

## イネーブル信号を効果的に使用

BUFHCE を使用するときには伝搬遅延を削減するには、次のように設定します。

1. 入力を論理 1 に接続する
2. ネットのロジック値を変更するためにイネーブル信号を使用する

これには次の設定が必要です。

1. CE\_TYPE 属性を ASYNC に設定する
2. INIT\_OUT を 0 に設定する

追加コストなしにこのパスに反転をエンコードするには、次のように設定します。

1. 入力をグラウンドに接続する
2. INIT\_OUT を 1 に設定する

同じことを BUFGMUX でも行うには、次の場合にセレクト (S) ピンに接続します。

- CLK\_SEL\_TYPE が ASYNC に設定されている
- 制約がマルチプレクサーへの入力に配置されている

## バッファの選択

バッファの選択により、ロジックのグループ化および配置を制御できます。

BUFH または BUFR を使用すると、接続されているコンポーネントは 1 つのクロック領域に制限され、要件が厳しい部分でタイミングを満たすためより最適な配置が可能になることがあります。

## バッファ配置の指定

特に高速パスで望ましい結果を得るため、バッファの配置を指定する必要がある場合があります。

## クロック リソースのまとめ

このセクションでは次のコンポーネントのクロック リソースをまとめます。

- BUFG
- BUFGCE
- BUFGMUX および BUFGCTRL
- BUFH
- BUFG
- BUFHCE
- BUFR
- BUFIO
- BUFMR
- BUFMRCE
- MMCM
- PLL
- IDELAY および IODELAY
- ODDR

### BUFG

BUFG は、ファンアウトの大きいクロックをデバイス全体の複数のクロック領域に供給する必要がある場合に使用します。

BUFG は、クロックをインスタンスエートしたり手動で制御しない場合に使用します。

BUFG は、極性が混合していない中速から低速のクロックのグローバル リセットなど、ファンアウトが非常に大きな非クロック ネットに使用できます。このような使用は、1 つのデザインにつき 2 つに制限してください。

クロックが複数の SLR にまたがる SSI デバイスの場合、中央の SLR にクロックを配置します。これでデザイン全体にクロック供給ネットを偏りなく分配でき、スキューを低減できます。

### BUFGCE

BUFGCE は、ファンアウトが大きい複数領域からなるクロック ドメインを停止するために使用します。

### BUFGMUX および BUFGCTRL

BUFGMUX および BUFGCTRL は、クロック周波数またはクロック ソースを変更するために使用します。

## BUFH

BUFH は、1 つのクロック領域に含めることができるロジックの小さいクロック ドメインに使用します。

BUFH は、超高速のクロック ドメインに使用します。

BUFH は、BUFG コンポーネントとクロック リソースを奪い合う可能性が低いクロック ドメインで使用します。

## BUFG

SSI デバイスの場合、上または下の SLR コンポーネントで BUFG を使用します。これで中央の SLR にある BUFG コンポーネントでリソースを奪い合う可能性が低くなります。

## BUFHCE

BUFHCE は、BUFG で駆動されている場合、1 つのクロック領域に配置可能なクロック ネットワークの中粒度の部分を停止するために使用します。

BUFHCE は、1 つのクロック領域に含めることが可能なりセットなどのファンアウトの大きい非クロック信号に使用します。

## BUFR

BUFR は、パフォーマンスが 200MHz 以下の小規模から中規模のクロック ネットワークに使用します。

BUFR は、クロック分周が必要な垂直方向に隣接するクロック領域 (3 領域まで) に制約できる外部供給クロックに使用します。

SSI デバイスの場合、上または下の SLR コンポーネントで BUFR を使用します。これで中央の SLR にある BUFG コンポーネントでリソースを取り合う可能性が低くなります。

## BUFIO

BUFIO は、通常ソース同期データ キャプチャでの外部供給の高速 I/O クロッキングに使用します。

## BUFMR

ザイリンクス 7 シリーズ FPGA デバイスのみで使用できます。

1 つのクロック ソースに対し垂直方向に隣接する複数のクロック領域にある BUFR または BUFIO コンポーネントを使用する必要がある場合に、BUFMR を使用します。

SSI デバイスの場合、SLR の中央クロック領域に BUFMR および関連ピンを配置します。これで、BUFMR から 3 つすべてのクロック領域にアクセスできるようになります。



## BUFMRCE

ザイリンクス 7 シリーズ FPGA デバイスのみで使用できます。

BUFMRCE は、クロックを定期的に停止する必要があるとき、1 つのクロック ソースに対し、垂直方向に隣接する複数のクロック領域にある BUFR または BUFIO コンポーネントを使用する必要があります。

BUFMRCE は、クロック分周が使用されている BUFR を複数使用している場合に使用します。接続されている BUFR コンポーネントすべての開始時の位相を確実にするために使用できます。

## MMCM

MMCM は、システム同期入力および出力のクロック挿入遅延を削除するために使用します (入力データにクロックの位相を揃える)。

MMCM は、データを正しく取り込むためにソース同期データをクロック揃えるクロック位相制御に使用します。

MMCM は、入力クロックの周波数またはデューティ サイクルを変更するために使用します。

MMCM は、クロック ジッターをフィルターするために使用します。

## PLL

PLL は、高速入力クロックの位相を揃えるために使用します。

## IDELAY および IODELAY

IDELAY および IODELAY は、入力クロックにわずかな位相オフセット (遅延) を追加するために使用します。

IDELAY および IODELAY は、入力データに遅延を追加するために使用します。これにより、データに対しクロック位相オフセットを効果的に削減できます。

## ODDR

ODDR は、デバイスからの外部転送クロックを作成するために使用します。



## その他のリソース

---

### ザイリンクス リソース

- デバイス ユーザー ガイド :  
[http://japan.xilinx.com/support/documentation/user\\_guides.htm](http://japan.xilinx.com/support/documentation/user_guides.htm)
- ザイリンクス用語集 : <http://japan.xilinx.com/company/terms.htm>
- 『ザイリンクス デザイン ツール：インストールおよびライセンス ガイド』(UG798) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/iil.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/iil.pdf)
- 『ザイリンクス デザイン ツール：リリース ノート ガイド』(UG631) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/irn.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/irn.pdf)
- 製品サポートと資料 : <http://japan.xilinx.com/support>

### ハードウェア資料

- 7 シリーズ デバイスの資料 :  
[http://japan.xilinx.com/support/documentation/7\\_series.htm](http://japan.xilinx.com/support/documentation/7_series.htm)

### ISE 資料

- ライブラリ ガイド :  
[http://japan.xilinx.com/support/documentation/dt\\_ise14-3\\_librariesguides.htm](http://japan.xilinx.com/support/documentation/dt_ise14-3_librariesguides.htm)
- ISE Design Suite のマニュアル :  
[http://japan.xilinx.com/support/documentation/dt\\_ise14-3.htm](http://japan.xilinx.com/support/documentation/dt_ise14-3.htm)
  - 『コマンド ライン ツール ユーザー ガイド』(UG628) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/devref.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/devref.pdf)
  - 『制約ガイド』(UG625) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/cgd.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/cgd.pdf)
  - 『Data2MEM ユーザー ガイド』(UG658) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/data2mem.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/data2mem.pdf)
  - 『ISim ユーザー ガイド』(UG660) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/plugin\\_ism.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/plugin_ism.pdf)
  - 『合成/シミュレーション デザイン ガイド』(UG626) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/sim.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/sim.pdf)
  - 『タイミング クロージャ ユーザー ガイド』(UG612) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/ug612.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/ug612.pdf)

- 『ザイリンクス/Cadence PCB ガイド』(UG629) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/cadence\\_pcb.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/cadence_pcb.pdf)
- 『ザイリンクス/Mentor Graphics PCB ガイド』(UG629) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/mentor\\_pcb.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/mentor_pcb.pdf)
- 『XPower Estimator ユーザー ガイド』(UG440) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/ug440.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/ug440.pdf)
- 『XST ユーザー ガイド (Virtex-4、Virtex-5、Spartan-3、および CPLD デバイス用)』(UG627) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/xst.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/xst.pdf)
- 『XST ユーザー ガイド (Virtex-6、Spartan-6、7 シリーズ デバイス用)』(UG687) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/xst\\_v6s6.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/xst_v6s6.pdf)

## パーシャル リコンフィギュレーション資料

- パーシャル リコンフィギュレーション ウェブサイト :  
[japan.xilinx.com/tools/partial-reconfiguration.htm](http://japan.xilinx.com/tools/partial-reconfiguration.htm)
- 『パーシャル リコンフィギュレーション ユーザー ガイド』(UG702) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/ug702.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/ug702.pdf)

## PlanAhead 資料

- PlanAhead 各種ユーザー ガイド :  
[http://japan.xilinx.com/support/documentation/dt\\_planahead\\_planahead/14-3\\_userguides.htm](http://japan.xilinx.com/support/documentation/dt_planahead_planahead/14-3_userguides.htm)
- 『フロアプラン設計手法ガイド』(UG633) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/Floorplanning\\_Methodolgy\\_Guide.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/Floorplanning_Methodolgy_Guide.pdf)
- 『階層デザイン設計手法ガイド』(UG748) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/Hierarchical\\_Design\\_Methodolgy\\_Guide.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/Hierarchical_Design_Methodolgy_Guide.pdf)
- 『ピン配置設計手法ガイド』(UG792) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/ug792\\_pinplan.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/ug792_pinplan.pdf)
- 『消費電力手法ガイド』(UG786) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/ug786\\_PowerMethodology.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/ug786_PowerMethodology.pdf)
- 『PlanAhead Tcl コマンド リファレンス ガイド』(UG789) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/ug789\\_pa\\_tcl\\_commands.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/ug789_pa_tcl_commands.pdf)
- 『PlanAhead ユーザー ガイド』(UG632) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_3/PlanAhead\\_UserGuide.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_3/PlanAhead_UserGuide.pdf)
- PlanAhead チュートリアル  
[http://japan.xilinx.com/support/documentation/dt\\_planahead\\_planahead14-3\\_tutorials.htm](http://japan.xilinx.com/support/documentation/dt_planahead_planahead14-3_tutorials.htm)