大型 FPGA 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

UG872 (v13.4) 2012 年 1 月 18 日

XILINX®

本資料は英語版 (v13.4) を翻訳したもので、内容に相違が生じる場合には原文を優先します。
資料によっては英語版の更新に対応していないものがあります。
日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

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

改訂履歴

日付	バージョン	改訂内容
2012 年 1 月 18 日	13.4	初版

目次

XILINX®

第 1 章

概要

ザイリンクスの『大型 FPGA 手法ガイド』(UG872) では大型 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。このガイドでは「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「大型 FPGA デバイス手法」にあるように、これらのストラテジは次の点において大型 FPGA から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」

大型 FPGA デバイス

「大型 FPGA デバイス」という表現は、このガイドではザイリンクス Virtex®-6 および Virtex-7 デバイス ファミリの大型デバイスを指します。

表 1-1 : ザイリンクス大型 FPGA デバイス

デバイス ファミリ	デバイス
Virtex-6	• LX550T • LX760
Virtex-7	• 580HT • 870HT • 980XT • 1140XT • 1500T • 2000T

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。

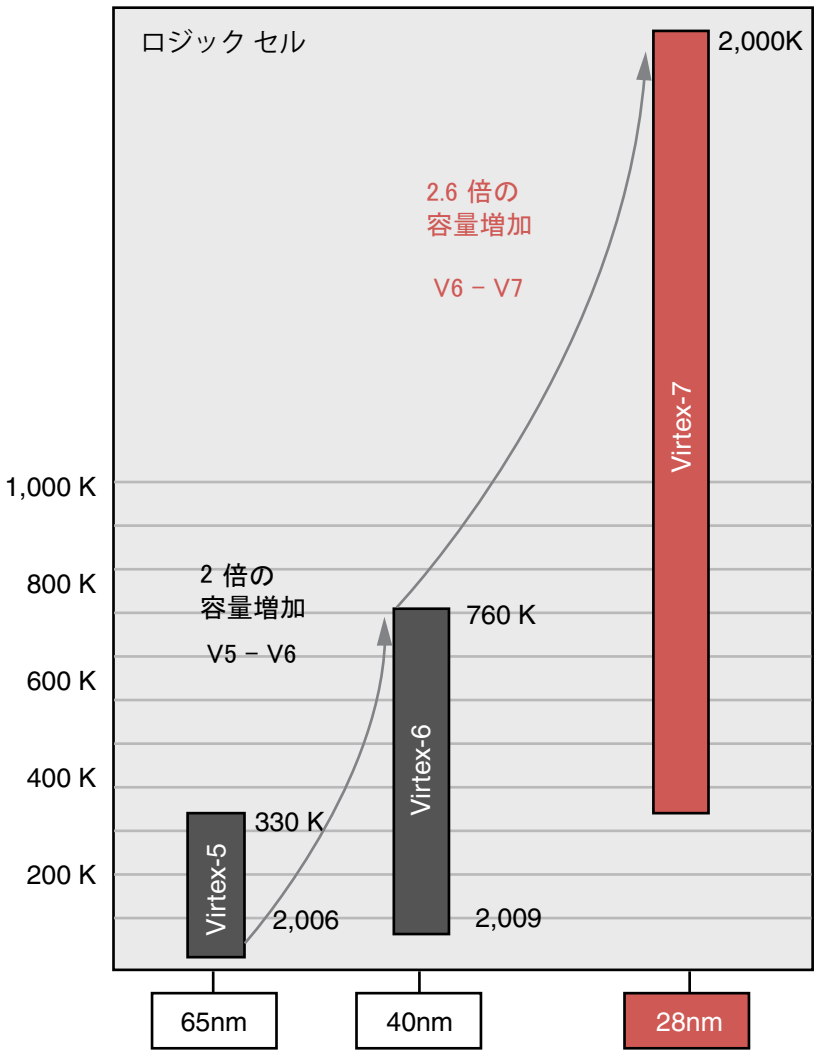

図 1-1 : FPGA の機能は 5 年未満で 6 倍以上向上

現時点での最大 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) テクノロジといわ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きも、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、手法が多く用いられます。しかし、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。

XILINX®

第 2 章

大型 FPGA デバイス手法

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、このガイドでは次の項目に関する FPGA デバイス設計手法を説明します。

- 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 デバイスは著しく変化を遂げてきたため、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。

ザイリンクス大型 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適に活か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、大型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。

本書で説明するトピックの多くは特に新しいものではなく、大型デバイスに限られた内容のものでもありません。しかし、次の大型 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、それを上回ることもできるのです。

配線使用率

FPGA デバイスの配線が固定された有限リソースであると常に認識されているわけではありません。

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軽減

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

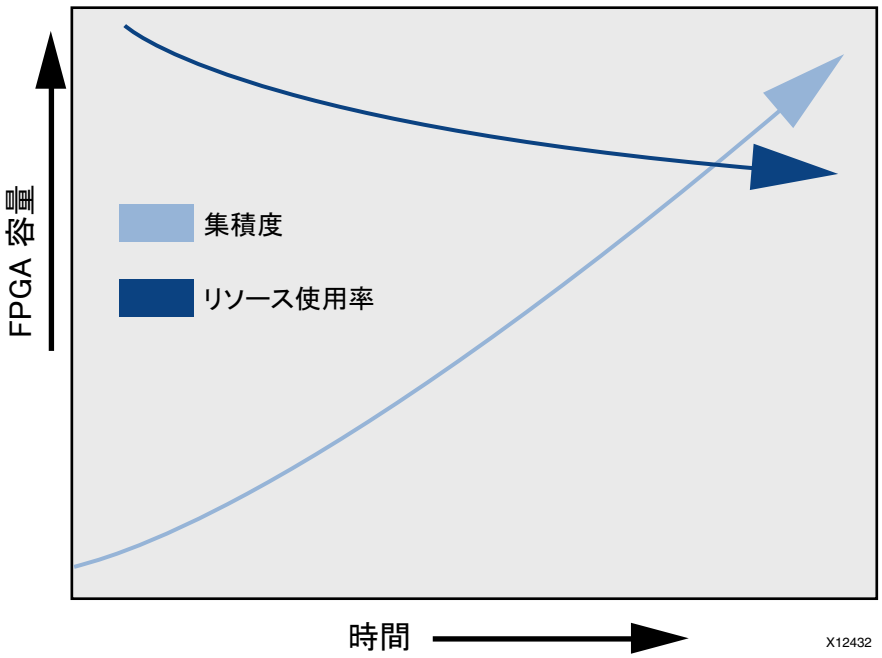

図 2-1：集積度が高いほど大型 FPGA デバイスで高いリソース使用率が求められる

配線リソースの非効率な使用

配線リソースを非効率に使用すると、次のような結果を招く可能性があります。

- 配線の混雑状態を引き起こす
- ツールの配線選択肢が限られ、次のような結果を招く
  - 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)

大型デバイスではロジック アレイが大きく、次のような傾向があります。

- 関連ロジック ファンクションが互いに隔離される
- ファンアウトがかなり大きくなってしまう信号が出てくる
- 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

本書で説明するテクニックを利用すると、配線使用率を向上させることができ、使用可能なロジックの数が増えます。

また、次の点も可能になります。

- 一部の FPGA リソースの必要性を抑える
- アプリケーションのデバイス使用率を抑える
- 今後の拡張のためリソースに余裕を持たせる
- 小型デバイスへ移行し、コスト、消費電力などの面で改善を図る

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、次のような問題を招くことがあります。

- 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

大型デザインには次のようなさまざまな要因が重なり、最適とはいえない配置を行わなければならない状況が発生することがあります。

- 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

最適でない配置が行われる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なのです。

消費電力

ザイリンクス FPGA デバイスの消費電力は近年著しく低下してきました。

こうした低減がなければ、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への電源供給を抑える努力が必要です。

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、大変な消費電力量に達してしまうことがあります。

本書の推奨事項の多くは、特定デザインやファンクションの消費電力を低減するのに役立ちます。

電力解析および消費電力低減テクニックの詳細は、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うことがあります。

- 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、さらに大きな FPGA デバイスへの移行や、複数の大型 FPGA デバイスへの移行を迫られることになりかねません。

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はさらに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を踏まえ、全体的なコストの上昇につながります。

XILINX®

第 3 章

スタックド シリコン インターコネクト (SSI)

ザイリンクスの『大型 FPGA 手法ガイド』(UG872) では大型 FPGA デバイスをターゲットにしたデザインについて説明します。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。

- さらに大型
- さらに多くの専用機能
- 低電力

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。

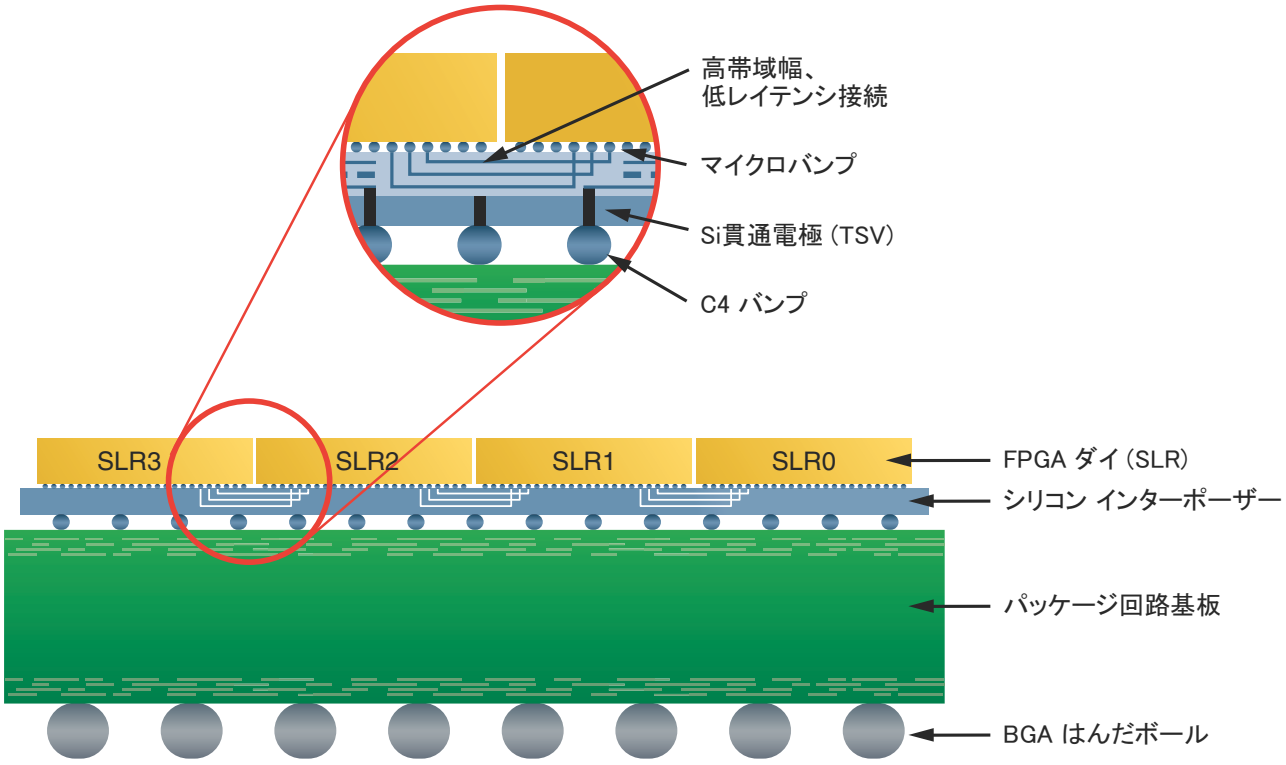

図 3-1 : SSI デバイスの構成

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。

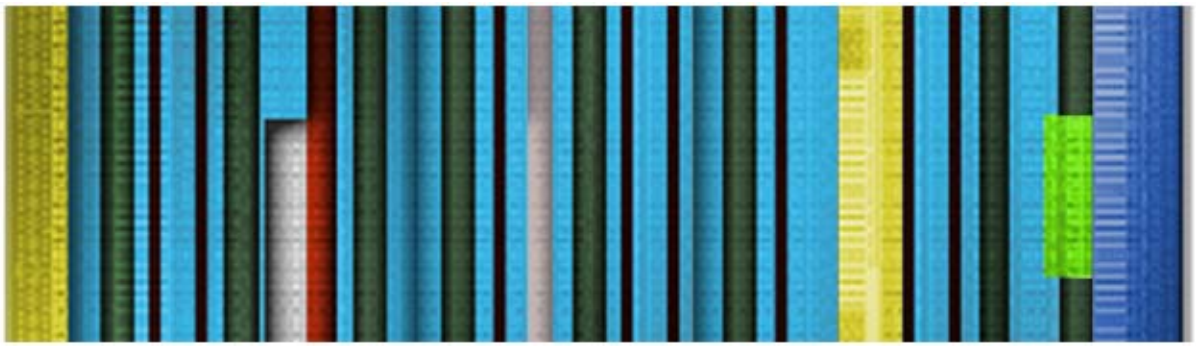

図 3-2 : SSI デバイスに含まれる SLR

SSI デバイスを作成するには、複数の SLR コンポーネントが垂直方向に重ねられます。

- 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 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

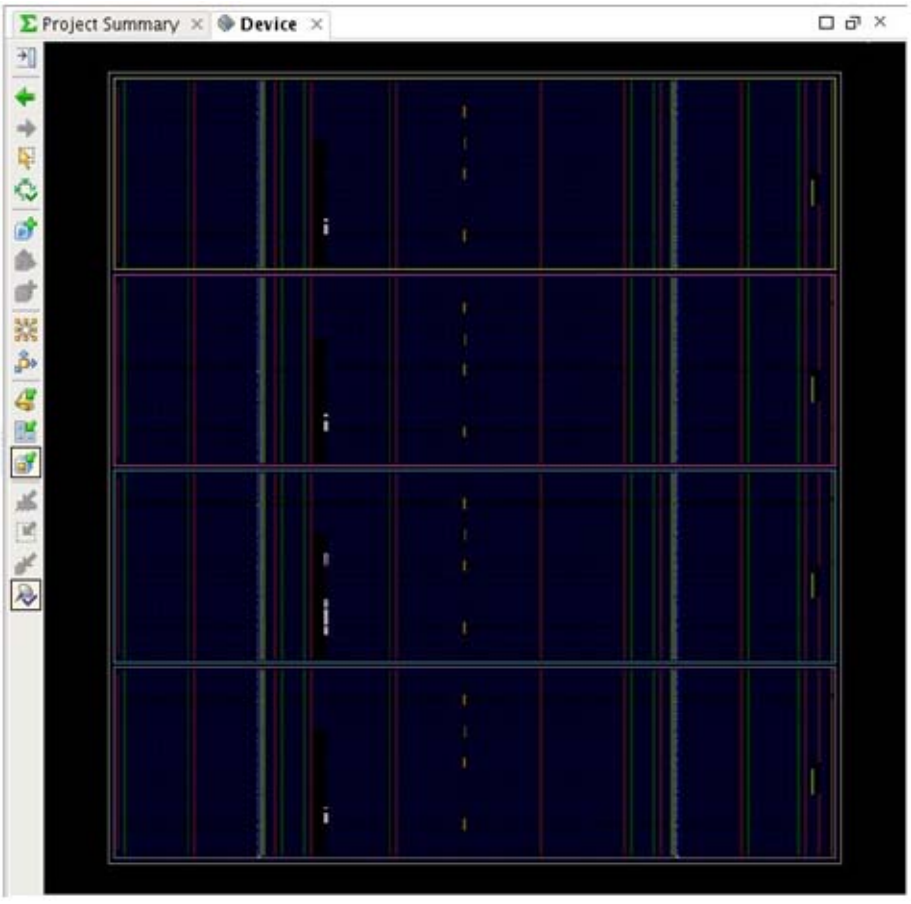

図 3-3 : PlanAhead ツールでの 2000T デバイスの表示

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 ファミリには 2 つの異なる SLR コンポーネントが使用されています。

- 「xc7v1500t および xc7v2000t デバイス」
- 「xc7vx1140t および Virtex-7 HT デバイス ファミリ」

xc7v1500t および xc7v2000t デバイス

xc7v1500t および xc7v2000t デバイスでは、次のものを含む同じタイプの 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
- xc7v1500T および 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	13,270	10,800

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 デバイスの受動レイヤーです。

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。

- 電力およびグランド
- 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。FPGA デバイスの回路をパッケージのボールに接続するため、シリコン インポーザーは Si 貫通電極 (TSV) 使用してパッケージ基板に実装されています。

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、デバイス パッケージに次を接続します。

- 電力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)

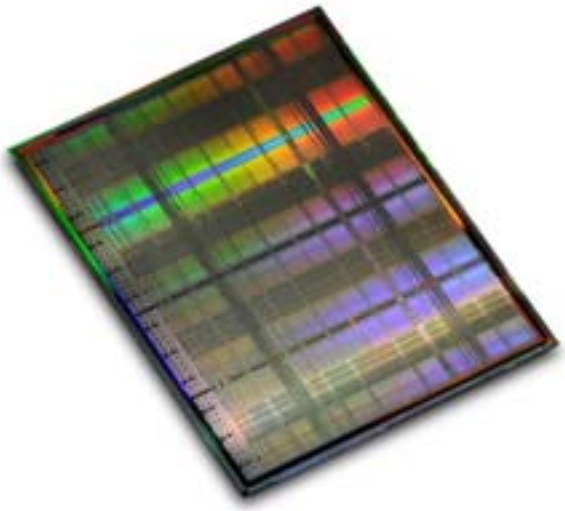

図 3-4 : シリコン インターポーザー

SLL (Super Long Line) 配線

SLL (Super Long Line) 配線の詳細は、次のとおりです。

- ある SLR から別の SLR に伝送される信号の一般的な接続を行う
- 「シリコン インターポーザー」にある
- 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る
- SLR の垂直方向の 12 本の配線の中央に接続する

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 に隣り合わせ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
7V1500T	13,270
7V2000T	13,270
7VX1140T	10,800

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。

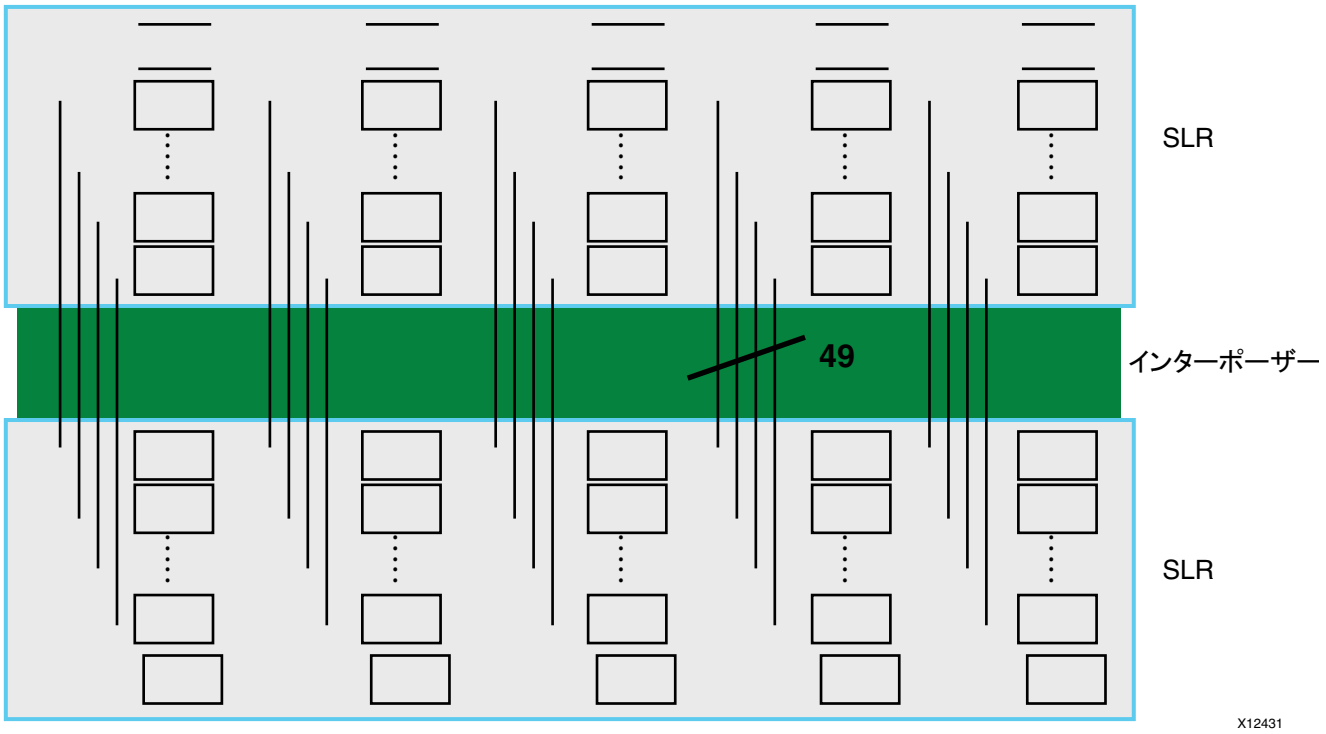

図 3-5 : SSI デバイスで少しずつずらされた SLL が交差

この SLR コンポーネント間の比率や差は図解用に書かれたもので、実値を反映するものではありません。実際の差は比較的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 に接続します。

この接続の役割は、次のとおりです。

- SLR とその隣の 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を提供する
- 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な配置を行う

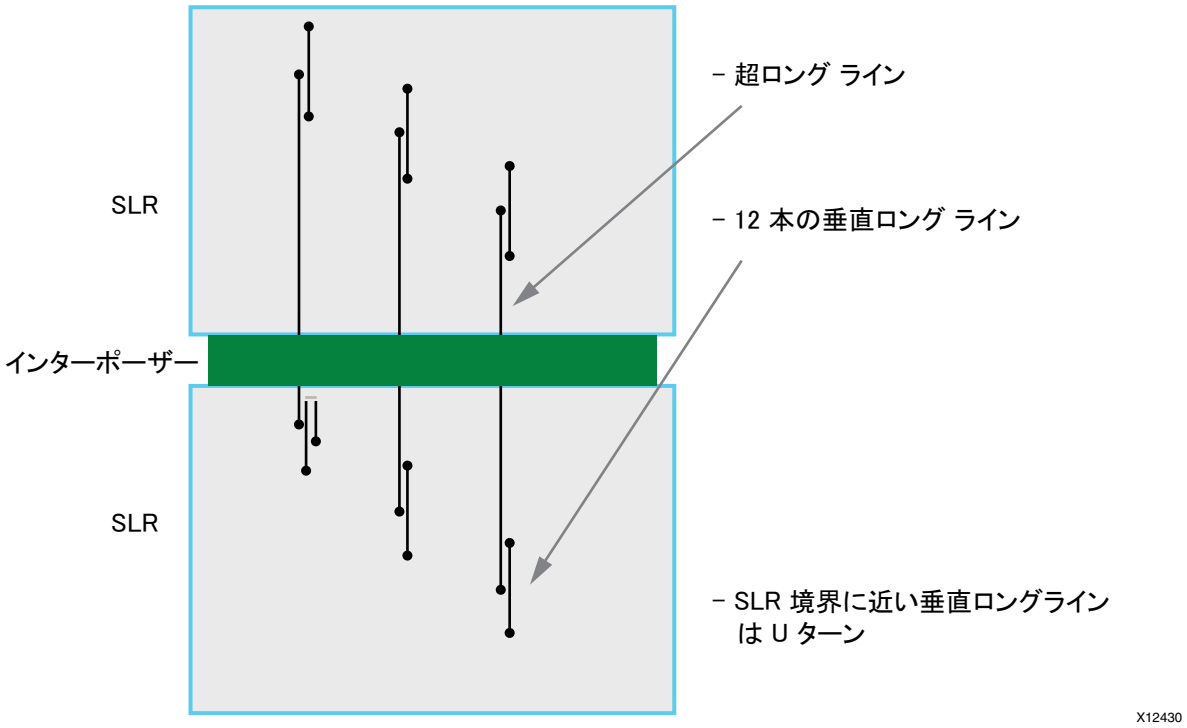

図 3-6 : SLR での SLL 接続

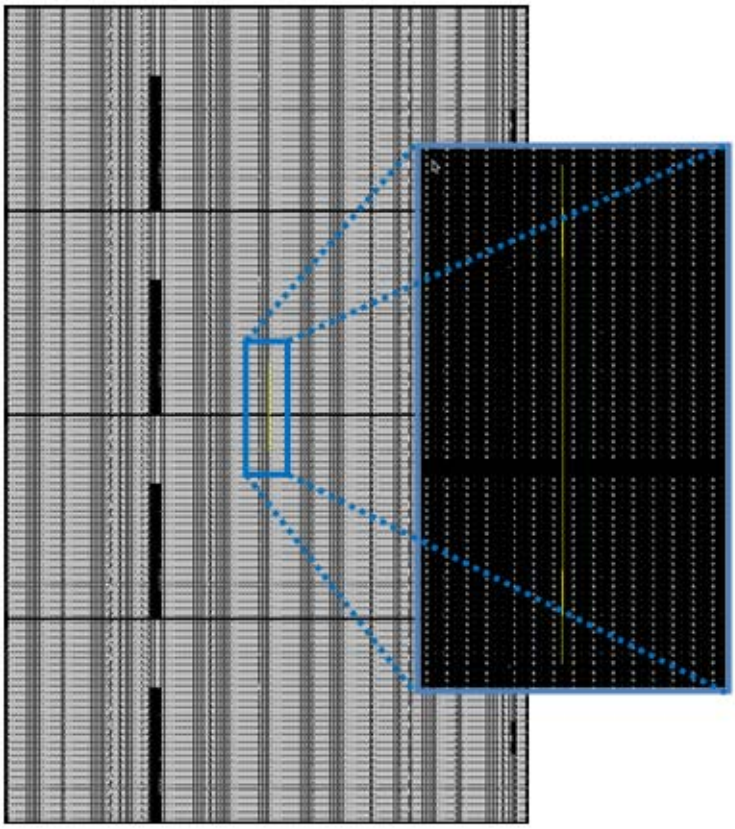

図 3-7 : FPGA Editor での SLL (ハイライト表示)

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ようにするには、次の作業を行うときにもこの制限を考慮する必要があります。

- 非常に長い DSP カスケードを構築する
- SLR の境界付近に手動でそのようなロジックを配置する
- デザインのピン配置を指定する。

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とき、配置配線ツールでは自動的に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。

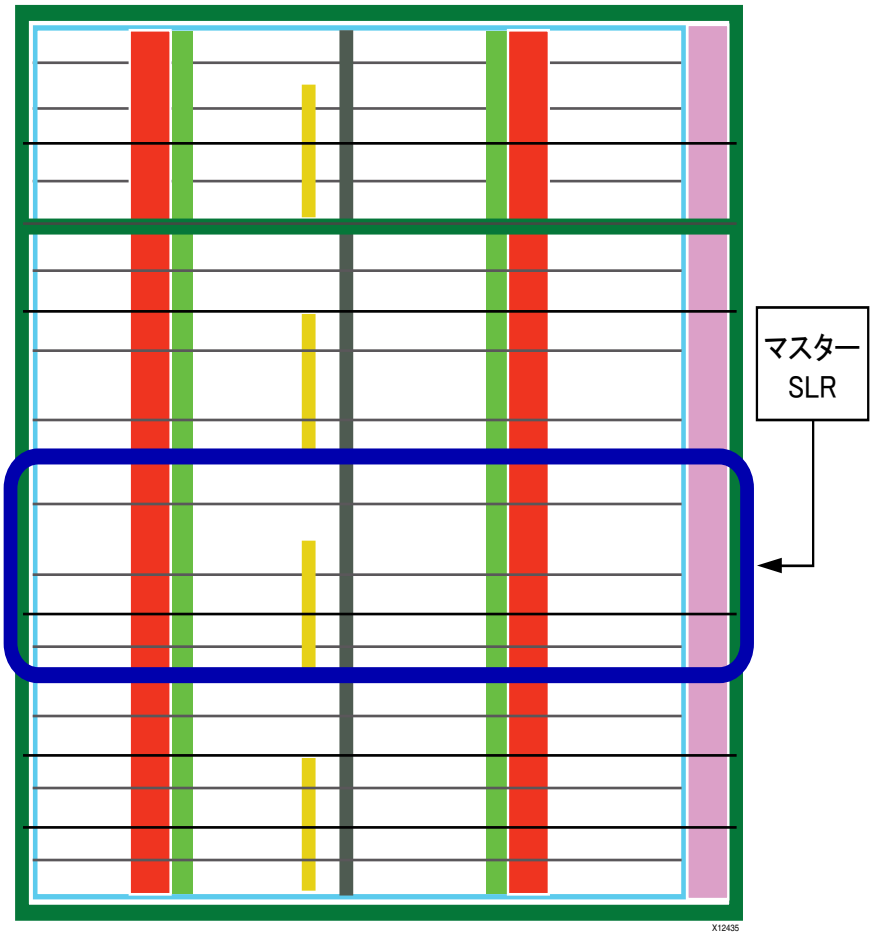

図 3-8 : xc7v2000t デバイスに含まれるマスター SLR

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

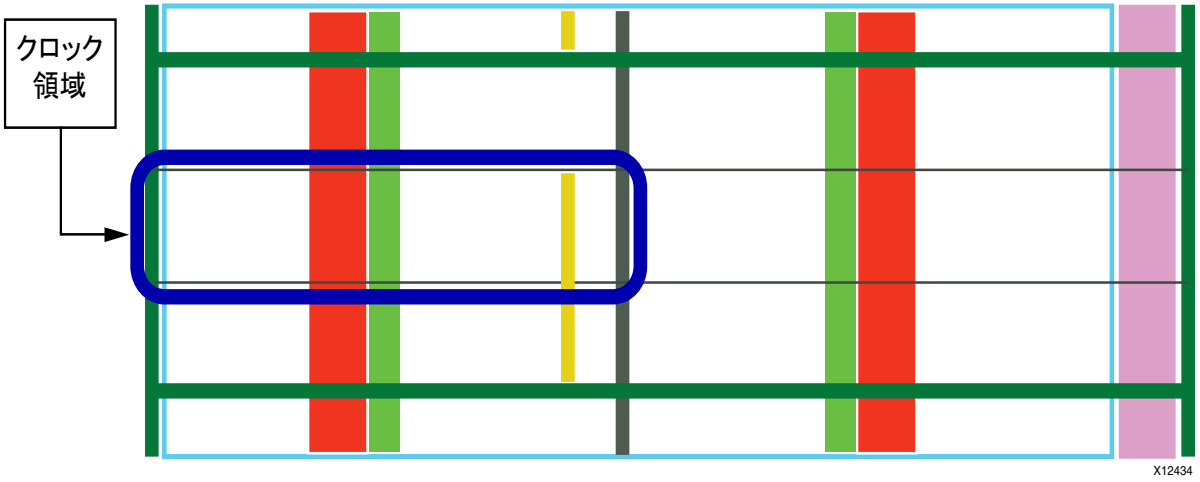

図 3-9 : クロック領域を含む SLR (拡大表示)

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 の場合、バッファーは複数の SLR コンポーネントをまたぐことがありません。BUFMR または BUFMRCE が別の SLR と直に接している SLR のクロック領域にある場合は、次のようになります。

- BUFMR または BUFMRCE は次のものにのみアクセス可能
  - BUFMR または BUFMRCE が配置されているクロック領域
  - 同じ SLR に直に接しているクロック領域
- BUFMR または BUFMRCE はこの近接している 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フル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提供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) もザイリンクス 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 SLR のクロック領域 1 個に含まれている 12 個の水平クロック (BUFH) の 1 つを駆動できる

1 つの SSI デバイスに含まれるすべての SLR コンポーネントで、クロッキングを含み、ほかのザイリンクス 7 シリーズ FPGA デバイスの場合と同様に想定してください。

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、次のようになっています。

- 各 BUFG の SLR の一番上から下までを接続
- 水平行クロッキングへの接続を可能

これらの垂直クロッキング ラインは、SLR の境界で非常に短いインターポーザーに接続され、インターポーザーを介して近接 SLR の対応ラインに接続されます。

このプロセスの詳細は、次のようになっています。

- SLR の水平クロッキング リソースを駆動
- すべての SLR コンポーネントが接続されるまで継続
- グローバル クロッキング リソースを作成

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ある程度管理することが必要になります。

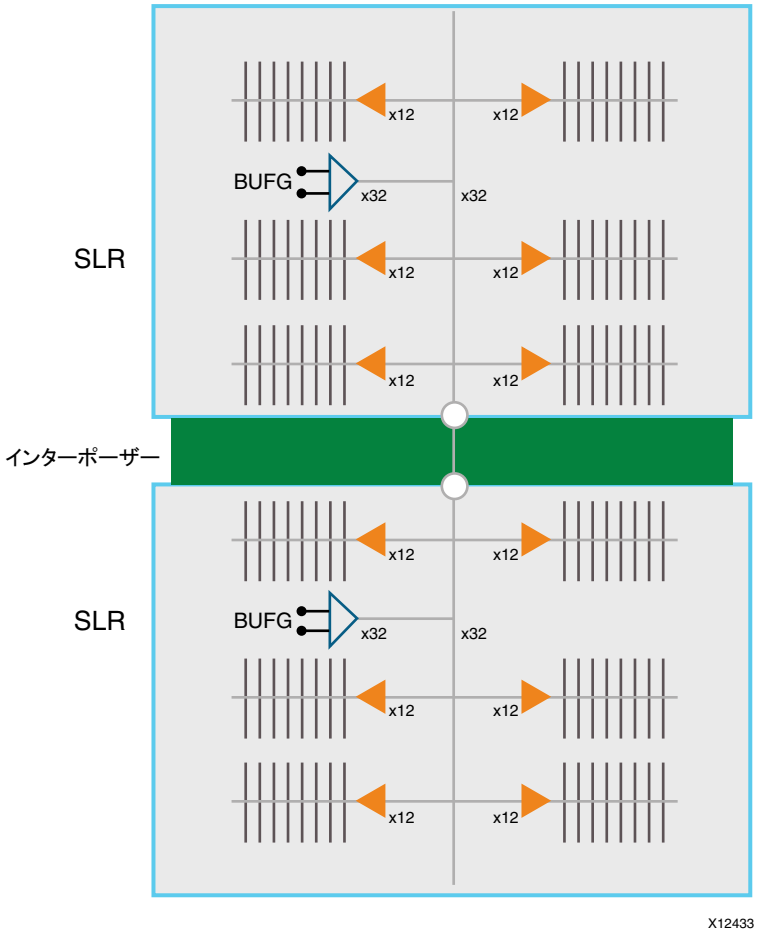

図 3-10 : SSI デバイスでのグローバル クロッキング接続

この SLR コンポーネント間の比率や差は図解用に書かれたもので、実値を反映するものではありません。実際の差は比較的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、ある程度の管理が必要になります。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作業が行われます。

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 を特定のリソースでいっぱいにしてしまわないように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沿うことで、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るようになります。

SLL の交差を制限する利点

SLL の交差を制限する利点は、一般的には次のようなものがあります。

- 消費電力の低減
- SLR 間の接続を損なうことなくデザインに拡張性を持たせる

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 (PBlocks) および LOC 制約などの物理的制約
5. タイミング制約
6. I/O 規格およびその他の制約

SLR コンポーネントを自動で割り当てる際は、ピン配置、クロック選択、およびその他のデザイン選択などを正しく行ってください。

フロアプラン

フロアプランはデザインの高速パフォーマンスが求められる部分に必要な場合があります。フロアプランは適宜行ってください。

手動 SLR 割り当て

次の場合に手動 SLR 割り当てが必要になることがあります。

- 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られない場合
- 実行の繰り返しが重要な場合

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。

近接する複数の SLR コンポーネントにロジックを割り当てることはできるのですが、必ず PBlock に SLR 全体が含めら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 を特定のリソースでいっぱいにしてしまわないように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」にある「配置ストラテジ」と同じです。ガイドラインに従うと、割り当てが現時点または今後のデザイン 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 デザインの階層がしっかりと定義されていると SLR が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、この推奨事項に沿ってください。

- 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 デバイスで非常に高速なデザインを達成するには、さらに手動で作業を行う必要がある場合があります。

SLR の速度は、同等スピード グレードの Virtex-7 デバイスものと同じです。

パイプライン処理によりパスのロジック レベルを少数に抑えたり、削除すると、SLR の境界を超えるときの速度を 400MHz 以上にできます。

パイプライン処理

SLR の境界を超えるときにレジスタ間の接続が高速になるように設計する場合、次のようにパイプライン処理を実行する必要があります。

- HDL コードで記述
- 合成時に制御

これで、SLR の境界を越える必要のあるロジック パスで SLR 自動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

次のもの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デスティネーション 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。

こうしたテクニックは常に必要なものではありませんが、ハイ 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いて、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を読み込むと、ビットストリームの一部を該当する SLR に割り当てるよう、個々の SLR コンポーネントが順次コンフィギュレーションされます。

共に接続される信号

インターポーザーでは、次の信号が共に接続されています。

- INIT
- DONE
- KEYCLEAR

これで、たとえば「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。シングル キーの利点は、次のとおりです。

- キーおよびコンフィギュレーション データの管理が簡単になる
- 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る

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

これでデータの取り込みに要する時間や、ECC の場合は破損データを修正する時間を短縮できるようになります。

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るのですが、マスター SLR のものが優先されるようになっています。

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の場合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 によりこの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。

XILINX®

第 4 章

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」
- 「ピン配置選択でのザイリンクス ツールの使用」
- 「一般的なピン配置選択の推奨事項」
- 「具体的なピン配置選択の推奨事項」
- 「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。配置が不適切の場合は配線が長くなり、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、これは特に大型 FPGA デバイスに言えることです。大型 FPGA デバイスは大規模な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。

ピン配置選択でのザイリンクス ツールの使用

デザイン プランニングおよびピン配置選択にザイリンクス ツールを使用することができます。しかし、ツールを効果的に利用するには、ツールに必要な情報を入力する必要があります。

ザイリンクス PlanAhead™ デザイン ツールなどのツールでは、ピン配置に対して次の利点があります。

- 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ための 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、ピン配置の選択もクロック供給に大きな影響を与えます。

一般的なピン配置選択の推奨事項

ピンの選択にあたっては、一般的に次の点に留意します。

- 関連信号は互いに近くに配置
- 関連信号は駆動するロードの近くに配置

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。しかし、よい FPGA デザインを目指すにはしっかりとしたピン配置を作成し維持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 ピンを使用する大型 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。内部的に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の方が重要です。

具体的なピン配置選択の推奨事項

次の点に関して具体的なピン配置選択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、近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、I/O インターフェイスを含むザイリンクス IP への推奨事項は、次のとおりです。

- インターフェイスを生成
- 推奨されているピン配置を使用

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、次の点に留意してください。

- 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制御信号を共有しているため、共通制御セットの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。

- 低いデバイス使用率
- 限られた配置のオプション

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、配線のオプションが多くなり柔軟性も増えので、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 エレメントになる可能性があります。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汎用同期コードから作成されることもあります。しかし、リセットを使用すると、こうしたコンポーネントの選択に影響し、デザインに対しリソースが最適に使用されないこともあります。

- 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 つが推論されるのと、数千個のレジスタが推論されるほどの差が出ることがある
- 遅延ラインに不必要にリセットが記述されていると、SRL LUT 数個とレジスタ数百個の差が出ることがある
- 乗算器の入力または出力に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る

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、ロジック 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で各レジスタは既知のステートになっています。デバイスの初期化のためだけにグローバル リセットのコード記述をする必要はありません。

リセットの使用の限定

リセットの使用を限定すると、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軽減
- 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。

- 各同期ブロックの見直し

  リセットが必要がどうか不明な場合は、リセットを記述しないでください。

- 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。

- SRL 1 つ
- 1 つの SRL と複数のレジスタの組み合わせ
- すべてレジスタ
- 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、HDL コード指定された初期値にすべてのレジスタを初期化します。

- 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は 0 または 1 に割り当てられる
- 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る

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が定義されています。

すべての同期エレメントを設定に従って初期化するようにしてください。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ようにしてください。

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 アーキテクチャに吹くまえっるリソース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、レジスタをブロックに直接推論できなくなり、機能に差が生じる可能性があります。

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よう、次のようなファンアウトの大きい制御信号で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

極性にはアクティブ High を使用するようにしてください。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。極性を混合すると混合制御セットが作成されます。混合制御セットを使用すると、次のような影響が出る可能性があります。

- 配置および配線に悪影響
- 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

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がすべてアクティブ 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追加 LUT が使用される可能性があります。

HDL コーディング手法

HDL コードの記述がしっかりしていると、次のような利点があります。

- 合成ツールの作業が簡単
- シミュレーションおよび合成での実行が高速
- FPGA アーキテクチャ間での移植性が高まる
- ターゲット アーキテクチャにうまく変換
- 読みやすく、デバッグがしやすい

よい HDL コード記述を行うには、次の点に留意します。

- デバイス リソースへ効率よく推論される HDL 構文
- 次の点に注意 :
  - ロジック
  - クロッキング
  - 制御信号
- 階層の決定
- 合成属性の使用

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を立てることで、潜在的な問題を回避できます。

次は、ザイリンクス FPGA リソースに RTL コードがどのようにマップされるかを示すガイドラインです。

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、加減算は、次のような特徴になります。

- 一般的にキャリー チェーンが使用される
- 2 ビットの加算ごとに LUT が 1 つ使用される

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間にあるレジスタを使用せずに別の値に追加されるとき)、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。3 値インプリメンテーションを生成することで、デバイス使用率を半分にカットすることができるからです。

乗算

乗算器は一般的に DSP ブロックで使用されます。

- 幅が 18 X 25 未満の符号付きビットは 1 つの DSP ブロックにマップされる
- 幅が 17 X 24 未満の符号なしビットは 1 つの DSP ブロックにマップされる
- 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る

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。

- 幅が 16 ビット以下の SRL コンポーネント 2 つ
- 最大 32 ビット幅の 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、消費電力に関してより望ましい結果が得られる可能性があります。

最大 64 ビットのメモリ アレイ

深さが最大 64 ビットまでのメモリ アレイは、通常 LUTRAM コンポーネントにインプリメントされます。

- 深さが 32 ビット以下の場合 1 つの LUT に 2 ビットマップされる
- 深さが最大 64 ビットの場合 LUT ごとに 1 ビット マップできる

深さの値が大きい RAM も、次の条件次第で LUTRAM 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場合があります。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、LUTRAM 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。

深さのビット数が 256 よりも大きいメモリ アレイ

深さのビット数が 256 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 デバイスの場合、幅と深さのさまざまな組み合わせでこうしたメモリをマップする柔軟性を備えています。こうした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 の数や構造が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ていただけでも、間違ったリソースが使用される場合があります。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、LUTRAM 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-to-1	• LUT 1 個	1
8-to-1	• LUT 2 個 • MUXF7 1 個	1
16-to-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、次の利点があります。

- リソースのよりよい管理
- デザインのデータ パスのロジック レベルの理解および制御

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。

早期にロジック レベル数を検討しておくと、タイミング クロージャ中の後半ではなく初期段階で変更がしやすくなります。

適切なデザイン階層の選択

デザイン階層は、次の点で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
- 古いコードの階層定義

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、次のようになります。

- 「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

できれば、大型の階層モデル、特に階層が上の方のモデルの出力にレジスタを付けます。

特に次の理由で階層の最適化が行われない場合に、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のではなく、モジュール内またはモジュール間の境界に常に含まれています。タイミングまたは機能に問題が発生すると、解析や修復が難しくなります。

階層の上位にクロック エレメントを配置

クロック エレメントを階層の上位に配置すると、次の利点があります。

- モジュール間のクロック共有が簡単
- 必要なクロック リソースが少ない
- リソース管理がしやすい
- パフォーマンスおよび消費電力が改善される

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。

- インスタンシエーションが必要がときは、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。
- 回路が上の方の階層にあると I/O の問題をデバッグしやすくなります。

階層デザイン

しっかりと境界を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。

次のような要因に影響を受けやすい箇所は個別に分けておく必要があります。

- タイミング クロージャ
- 論理デバッグ

階層をしっかりと定義しておくとフロアプランがしやすくなり、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。SLR にすべてのロジックが割り当てられるよう階層を定義しておくと次の利点があります。

- 割り当てがしやすくなる
- 配置配線が早く完了する
- デザイン解析がしやすくなる

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 デバッグが簡単になります。

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。これで大型 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てタイミング制約を満たすことができない
- ロジック レベル数が多すぎて目的の速度でデザインが動作できない

パイプライン処理には、次のような利点があります。

- タイミング要件を満たすのに十分なロジック レベル数に抑える (この実現にはパイプライン処理しか方法がない場合がある)
- 早期にパイプライン処理をプランしておくと、タイミング クロージャが非常に簡単になる

  パスにパイプラインを追加すると、レイテンシの差を回路全体に伝搬できます。この若干の変更でコードの大部分を変更できます。
- 設計の早期段階にパイプライン処理できる箇所を確認しておくと、次の利点があります。
  - タイミングクロージャが達成しやすくなる
  - インプリメンテーション ランタイムを短縮 (タイミングの問題を修正しやすくなるため)
  - デバイスの消費電力を低減 (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。

デザインへの影響を予測するため、負荷が数千あるネットを確認します。

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。

- 「負荷の低減」
- 「BUFG および BUFH の使用」
- 「ツールで重複の管理」

負荷の低減

負荷を必要としないデザインの部分に負荷を移動させます。

- ファンアウトが大きい制御信号の場合は、コード記述されているすべての箇所でそのネットが正しく機能する必要があるかどうかを確認する。負荷要求を抑えるとタイミングを著し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する。

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 バッファーを使用」を参照してください。

ツールで重複の管理

ツールで重複を管理します。

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重複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。

XILINX®

第 5 章

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すら、デザインにおける決定がデザイン目標に大きく影響を与えます。早期から賢明に計画することで、正しい決断ができ、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

ピン配置を選択する前に、最初のステップの 1 つとしてクロック 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、特定のピン配置やそのロジックの配置が決まってきます。正しいクロック供給の選択は良好な結果につながります。

ザイリンクスの Virtex®-6 および Virtex-7 には BUFG というグローバル クロック バッファーが 32 個含まれています。

BUFG を使用することで、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 コンポーネントには、次のような特徴があります。

- 合成で簡単に推論
- 制限事項が少なく、一般的なクロック供給が可能

しかし、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の点を推奨します。

- 使用可能なクロック リソースに対するクロック ニーズを解析する
- 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 バッファーにはクロック供給以外にも機能があります。この追加機能は、次の場合にのみ使用できます。

- デザイン コードの手動編集中
- 合成中

グローバル クロック バッファーには、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 :

- 同期した、グリッチのないクロック イネーブル (ゲーティング) 機能にアクセス可能
- 追加ロジックやリソースが不要

BUFGCE は、次の目的に使用します。

- 一定期間クロックを停止する
- 周波数の高いベース クロックから周波数を 2 分割または 4 分割したクロックなど、ロー スキューおよび低消費電力のクロック分周を行う

  これは回路動作の時間ごとに異なる周波数を生成するときに特に便利です。

BUFGMUX

BUFGMUX は、次の目的に使用します。

- あるクロック ソースから別のクロック ソースまでの間でグリッチまたはタイミングの問題を発生させずに、クロックを変更する
- 異なる時間または動作条件に対しクロック周波数を生成する

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 バッファーでは、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 および合成でも高度なクロック供給機能を使用することができます。

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 バッファーを使用できます。

クロック供給アーキテクチャのプラン中は、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シエート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ー (BUFH、BUFHCE)」
- 「リージョナル クロック バッファー (BUFR)」
- 「I/O クロック バッファー (BUFIO)」
- 「マルチリージョナル クロック バッファー (BUFR)」

水平クロック領域のバッファー (BUFH、BUFHCE)

水平クロック領域のバッファー (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 バッファーとして使用

水平クロック領域のバッファーの使用

水平クロック領域のバッファーは、次の目的に使用します。

- 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
- クロック ドメイン数が多いデザインに追加クロック リソースを提供する

BUFH および BUFHCE リソースを使用すると、任意の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 には同一のグリッチなしのクロック イネーブルがあり、特定クロック 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 の単純なバッファーとなり、つまり同じリソースになります。

クロック イネーブルが不要な場合は BUFH を使用するようにしてください。

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。

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 (負荷が数百から数千の範囲) に対し、BUFHCE はクロック リソースとして効果的です。

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 から独立して BUFH および BUFHCE を使用

BUFH および BUFHCE は、別のクロック機能を追加するため BUFG からは独立して使用することもできます。

独立して使用する場合、BUFH に接続されている負荷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 (負荷が少ない) 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。

- 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
- 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。

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へと駆動されている場合、例外が 1 つ発生します。この場合、同じクロック ソースにより両方の BUFH コンポーネントが駆動されているときの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で、2 つの BUFH クロック ドメインをまたいでデータを伝送することができます。

リージョナル クロック バッファー (BUFR)

リージョナル クロック バッファー (BUFR) は、スピードの遅い I/O およびデバイス クロックとして、よりスピードの速い I/O データのキャプチャおよび伝送に通常使用されます。

リージョナル クロック バッファーの使用

リージョナル クロック バッファー (BUFR) は、次の目的に使用します。

- クロックのイネーブルまたはディスエーブル (ゲート)
- 公約数によるクロック分周

BUFR は、Virtex-6 デバイスで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します。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 は使用しないようにしてください。BUFR は中速から低速のクロック供給に適してします。

ビルトインのクロック分周機能を使用する場合、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 BUFR は適しています。

I/O クロック バッファー (BUFIO)

I/O クロック バッファー (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への出力ロジックに供給する
- 高速、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のデータ伝送が確実にでき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 バッファー (BUFR)

マルチリージョナル クロック バッファー (BUFMR) には、次の特徴があります。

- ザイリンクス 7 シリーズ FPGA デバイスでの新機能です。
- 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、シングル 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 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。しかし、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 は 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。

バンクまたはクロック領域へと BUFMR を駆動するクロックを SLR の中央にあるクロック領域に配置することを推奨します。これで、SLR の 3 つのすべてのクロック領域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は特に注意事項はありません。
- 競合を避けるため BUFG コンポーネントは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、クロック負荷の配置、またはその両方が基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。

ほかのザイリンクス 7 シリーズ FPGA デバイスと同様、クロック対応 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して制限が設けられています。

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。

競合なくすべての SLR コンポーネントにすべてのクロックが駆動できるように、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 ドメインの数を低減することができます。

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。

水平方向に並ぶクロック領域では、左右両方の HROW バッファーをロー 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。

SLR コンポーネントの近くにある垂直グローバル クロック ラインに隔離バッファーがあり、同じ垂直グローバル クロック ラインにある別の SLR で 2 つの BUFG コンポーネントを競合なしに使用できます。

追加ユーザー制御および介入

この機能には、追加ユーザー制御および介入が必要です。

- BUFG の使用にあたっては、選択に注意し、LOC 制約を使用して手動で配置する必要があります。
- 各クロック ドメインの負荷すべてを手動でグループにまとめ、クロッキングの競合を避けるため適切な SLR に配置する必要があります。

33 個以上のグローバル クロック リソースを使用できるのは、次のような場合です。

- グローバル クロックがすべて配置されている
- クロック競合が発生しないようにすべての負荷が管理されていて、すべての負荷にクロックがアクセスできるようになっている

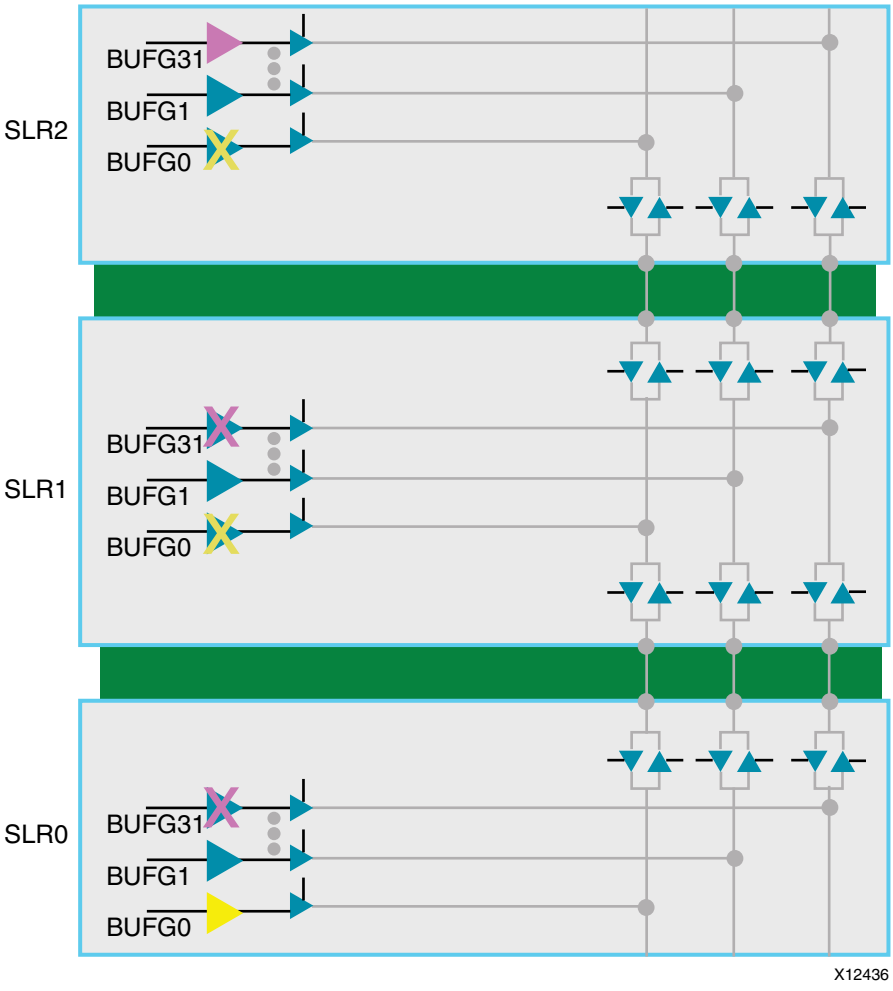

図 5-1 : クロック スキュー

SSI デバイスのクロック スキュー

大型 FPGA デバイスのクロック スキューは、任意パスのタイミング予測における大きな一因です。

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点に関し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 デバイスもほかの大型 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

デバイス 1 つに含まれるマルチダイ SLR は、次のようになっています。

- PVT のプロセス部分を誇張する
- ザイリンクス アセンブリ プロセスによって管理され、同じレベルのスピードのダイのみが一緒にパッケージされる

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のような差が考慮されます。ツールではパス解析中に次の作業が行われます。

- 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を解析
- 指定要件に対するパス遅延の一部としてレポート

SSI デバイスの場合、別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、パス遅延の差も誇張され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、2 つ以上の SLR を駆動する必要があるグローバル クロックを中央の SLR に配置するようにしてください。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」
- 「インスタンシエーション」

推論

特に指定がない限り、アーキテクチャで使用可能な数のクロックにグローバル バッファー (BUFG) が合成ツールにより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。パーツでの BUFG コンポーネントの機能や使用可能な数を超えるクロッキングではない限り、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

クロック リソースを制御するには、正しく合成属性を使用することが一番簡単です。

合成属性の使用

合成属性の役割は、次のとおりです。

- BUFG 推論を防ぐ
- BUFG をほかのクロック供給構造に置き換える
- クロック バッファーがない場合に指定する
- HDL ソース コードを変更せずにクロック リソースを制御する

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 バッファーを指定しない
- BUFH や BUFR などの特定バッファー 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。
- XCF (Xilinx Constraint File) 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 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 にクロック供給構造が含まれていることもあります。

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

こうした IP は考慮の上で追加クロック リソースとして使用できます。

しっかり検討せずに使用すると、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。

インスタンシエートされた 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。

インスタンシエーション

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

- 「インスタンシエーションの利点」
- 「最上位でのクロック リソースのインスタンシエーション」
- 「重複クロック リソースのリスク」

インスタンシエーションの利点

HDL デザインにクロック リソースをインスタンシエートすると、デバイスのすべての機能にアクセスおよび制御することができます。

一般的に、インスタンシエーション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

単純なバッファーの場合であっても、デザインにそれをインスタンシエート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シエーション

コードの最上位またはその近くにある別のエンティティやモジュールにクロック リソースを配置するようにしてください。

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散させることができます。

重複クロック リソースのリスク

クロック リソースが重複していると、次のような問題が発生します。

- FPGA リソースの浪費
- 消費電力の浪費
- 次の結果を招く、競合および配置が発生：
  - インプリメンテーション ツールのランライムが長くなる
  - タイミング シナリオが複雑になる

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」
- 「ゲート化クロックの使用」
- 「ダイナミック消費電力の低減」

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しく制御することができます。

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 が次の状態にあるようにするためにいくつかの属性を設定する必要があります。

- 仕様どおりに動作している
- 出力で目的のクロック特性が見られる

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 インスタンシエーション

MMCM または PLL は直接インスタンシエートすることもできます。

直接インスタンシエート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、次のような問題が発生する可能性があります。

- ジッターが大きくなりクロックのばらつきが増える
- 間違った位相関係になる
- 他の問題を引き起こし不必要にタイミングを満たすことが困難になる

位相制御のための IDELAY の使用

若干の位相調整が必要な場合にのみ、MMCM および PLL ではなく、IDELAY または ODELAY を使用することができます。

IDELAY または ODELAY を使用すると、次のことが可能になります。

- 追加遅延を追加
- 任意の関連データに関しクロックの位相オフセットを増加

ゲート化クロックの使用

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 リソースを提供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するために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 テクニック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上の理由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を使用している場合、合成ツールを使用してクロック パスに配置されたゲートをデータ パスにマップし直すことができます。この利点は、次のとおりです。

- クロック リソースへよりよいマッピングが可能
- ゲート化されているドメインへ出入力するデータの回路のタイミング解析が簡単になる

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遅いクロックへとクロック 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低減

上記のどのテクニックでもダイナミック消費電力を効果的に低減できます。どのテクニックが最も効果的になるかは、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、次の場合に最も効果的です。

- クロックが外部生成されている
- 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) でこのテクニックを使用するには Virtex-7 デバイスが必要な場合があります。

BUFHCE

BUFHCE は、シングル 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、最適な消費電力を得るにはベストではありません。

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 サイクルに対してよく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 つの入力を High にする
- 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。

外部クロックに対しさらに位相制御が必要な場合は、次のいずれか、または両方を使用して MMCM または PLL 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緩和させることができます。

クロック ドメインの交差

プロセス、電圧、温度 (PVT) に差があるため、2 つのクロック ドメインを交差する場合はクロックおよび位相のばらつきが増加します。

クロック ドメインを交差すると、次の状況が発生します。

- セットアップおよびホールド計算に追加が発生する
- スピードが遅いデザインでもタイミングを満たすのが困難になる場合がある

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。

- クロック ドメイン数を定義する
- 次のコンポーネントの 1 つに汎用クロック イネーブルを使用する
  - 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、マルチ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、ドメイン間でデータが正しく伝送されていることを確認する特別なテクニックを利用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、ある程度の注意事項に沿えば問題なくドメイン間でデータ伝送が可能です。

クロック スキュー

コモン ノードおよびソースからデスティネーションへのバッファーでの伝送距離によりますが、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 レベル数の増減を検討するようにしてください。

同期ドメイン交差の例

- チップの同じサイド (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、間違ったバス キャプチャやメタステーブル状態など、パスのデータ 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、非同期クロック ドメインを問題なくデータが交差できるようにするに、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 ビット以上伝送するのにグレー 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 (つまりバス) 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伝送を行います。

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。しかし、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シエートしたり、CORE Generator™の FIFO Generator を使用して FIFO を作成することができます。

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

ディアサートが行われるとき、デザインは既知の初期ステートにあり操作できるよう解放されます。

未知のステート

次の場合、デザインの一部が未知のステートになる場合があります。

- リリース 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
- GWE が安全にリリースできる速度よりも速いスピードでクロックが動作している場合
- この両方の状態が発生している場合

デザインの中には未知のステートになっても問題がないものもありますが、それ以外のデザインは未知のステートになると、次のような問題が発生する可能性があります。

- 不安定になる
- 初期データ セットが間違って処理される
- この両方の状態が発生する

既知のステート

デザインが既知のステートにならなければならない場合、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ます。つまりデザインに同期化して解放されます。

これは、次のように設定します。

- STARTUP コンポーネントをインスタンシエートします。
- 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に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。

これは、次のように設定します。

- 次のコンポーネントをインスタンシエートします。
  - BUFGCE
  - BUFHCE
  - BUFR
- 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
- クロック イネーブルとローカル リセットの両方

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することも可能です。

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、次の点に留意してください。

- 「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 スキューに対応しており、短い伝搬遅延に対応している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、その信号はデスティネーションに向けグローバル クロック ネットから送信される必要があります。

- 2 つ以下の BUFG コンポーネントがグローバル クロック ネットのある 1 つのポイントからその信号を出力する場合は、競合はありません。どのような状況でもデザインを配線することができます。
- 3 つ以上の BUFG コンポーネントが同じロケーションに信号を送信する場合は、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。

配線競合を避けるよう r 、非クロック信号には 3 つ以上の BUFG コンポーネントを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のある 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 を使用するときに伝搬遅延を軽減するには、次のように設定します。

- 入力をロジック 1 にする
- 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。

- CE_TYPE 属性を ASYNCに設定する
- 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。

- 入力をグランド接続します。
- 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

バッファーの選択

バッファーの選択では、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 を使用する場合 :

- 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る
- 要件が厳しい箇所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

バッファー配置の指定

特に高速パスで求める結果を得るには、バッファー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」
- 「IDELAY および IODELAY」

BUFG

BUFG は、次の場合に使用します。

- ファンアウトの大きいクロックをデバイス全体にいくつかのクロック領域に供給する必要がある場合
- 制御クロックを手動でインスタンシエートしない場合
- 極性が混合していない中速から低速のクロックに対しグローバル　リセットなどのファンアウトが非常に大きな非クロック ネット。この使用は 1 デザインにつき 2 箇所に制限するようにしてください。

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、スキューを軽減できます。

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、次の場合に使用します。

- シングル クロック領域に含めることができるロジックのクロック ドメインが小さい場合
- クロック ドメインが超高速である場合
- クロック ドメインが 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場合

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、次の目的に使用します。

- BUFG によって駆動されているとき、シングル 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
- シングル クロック領域に含めることが可能なリセットなどファンアウトの大きい非クロック信号に対して使用する

BUFR

BUFR は次のものに使用します。

- 200MHz を超える速度が必要ではない小規模から中規模のクロック ネットワーク
- クロック分周が必要な垂直方向に接するクロック領域 (最大 3 領域まで) に制約できる外部供給クロック

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クロック領域複数にある 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、次の場合に使用します。

- クロックを定期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クロック領域複数に BUFR または BUFIO コンポーネントを使用する必要のある場合
- 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。接続されている BUFR コンポーネントすべての位相スタートアップを確実にするため BUFMRCE を使用可能

MMCM

MMCM は、次の目的に使用します。

- 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 (入力データにクロックの位相を揃える)
- 正しくデータを取り込むため、クロック位相制御でソース同期データをクロック揃える
- 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
- クロック ジッターをフィルターする

PLL

PLL は、次の目的に使用します。

- 高速入力クロックの位相を揃える

IDELAY および IODELAY

IDELAY および IODELAY は、次の目的に使用します。

- 入力クロックにわずかな追加位相オフセット (遅延) を追加する
- 入力データにデータへの追加遅延を追加するこれでデータに対しクロック位相オフセットを効果的に低減できます。

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。

XILINX

付録 A

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
- 用語集 : http://japan.xilinx.com/company/terms.htm
- 『ISE Design Suite : 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/xilinx13_4/iil.pdf
- 『ISE Design Suite : リリース ノート ガイド』(UG631) :
  http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/irn.pdf
- 製品のサポートおよび資料 **:** 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

ISE 資料

- ライブラリ ガイド :
  http://japan.xilinx.com/support/documentation/dt_ise13-4_librariesguides.htm
- ISE Design Suite のマニュアル :
  http://japan.xilinx.com/support/documentation/dt_ise13-4.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/xilinx13_4/devref.pdf
  - 『制約ガイド』(UG625) :
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/cgd.pdf
  - 『ISim ユーザー ガイド』(UG660) :
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/plugin_ism.pdf
  - 『消費電力手法ガイド』(UG786)
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/ug786_PowerMethodology.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/xilinx13_4/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/xilinx13_4/ug612.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/xilinx13_4/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
- 『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/xilinx13_4/ug702.pdf

PlanAhead 資料

- PlanAhead 各種ユーザー ガイド http://japan.xilinx.com/support/documentation/dt_planahead_planahead13-4_userguides.htm
  - 『フロアプラン手法ガイド』(UG633) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/Floorplanning_Methodolgy_Guide.pdf
  - 『階層デザイン手法ガイド』(UG748) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/Hierarchical_Design_Methodolgy_Guide.pdf
  - 『ピン配置手法ガイド』(UG792) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/ug792_pinplan.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/xilinx13_4/ug789_pa_tcl_commands.pdf
  - 『PlanAhead ユーザー ガイド』(UG632) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_4/PlanAhead_UserGuide.pdf