

この資料は英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。こちらの日本語版は参考用としてご利用ください。設計の際には、最新の英語版で内容をご確認ください。

QII51006-6.0.0

はじめに

今日の FPGA アプリケーションは、ASIC に匹敵する複雑さおよび性能要件に達しています。このような複雑なシステム・デザインの開発では、適切なデザイン手法が、デバイスのタイミング性能、ロジック利用率、およびシステム信頼性にきわめて大きな影響を与えます。適切にコーディングされたデザインは、ターゲットを異なるファミリやスピード・グレードに変更した場合でも、予測可能な高い信頼性のもとで動作します。適切なデザイン手法を用いれば、FPGA と HardCopy® またはプロトタイプ作成および生産用 ASIC 実装の間で、デザインの移行を成功させるのにも役立ちます。

アルテラ・デバイスで設計する際は、最適な性能、信頼性、および迅速な「time-to-market」を達成するために、以下の指針に従う必要があります。

- 同期デザイン方式の影響を把握する。
- 階層デザイン分割を含む推奨デザイン手法に従う。
- ターゲット・デバイスのアーキテクチャ上の特徴を活用する。



メモリや DSP ブロックなどの専用デバイス・ハードウェアをターゲットとする場合のコーディング・ガイドラインを含む、特定の HDL コーディング例と推奨事項については、「Quartus II ハンドブック Volume 1」の「Recommended HDL Coding Styles」の章を参照してください。HardCopy デバイスへのデザインの移行についての情報は、「HardCopy Series Handbook」の「HardCopy Series Design Guidelines」の章を参照してください。

同期 FPGA デザイン手法

適切なデザイン手法における最初のステップは、実施するデザイン作業およびデザイン手法の意味を理解することです。この項では、最適な同期デザイン作業の利点と、その他の手法に伴う危険性の概要を説明します。適切な同期デザイン方法は、常にデザインの目標達成に役立ちます。その他のデザイン手法の問題としては、デバイスの伝播遅延への依存、不完全なタイミング解析、グリッチの可能性などがあります。

同期デザインでは、クロック信号がすべてのイベントをトリガします。すべてのレジスタのタイミング要件が満たされている限り、同期デザインはすべてのプロセス、電圧、および温度 (PVT) 条件で、予測可能かつ信頼性の高い方法で動作します。同期デザインでは、簡単に異なるデバイス・ファミリやスピード・グレードをターゲットにすることができ

ます。さらに、同期デザイン手法は、デザインをアルテラ HardCopy デバイスなどの量産ソリューションに移行する場合や ASIC のプロトタイプを作成している場合に、移行を確実に成功させるのに役立ちます。


同期デザインの基礎

同期デザインでは、すべての要素がクロック信号を基準にしています。クロックのすべてのアクティブ・エッジ（通常は立ち上がりエッジ）で、レジスタのデータ入力がサンプリングされ、出力に転送されます。アクティブ・クロック・エッジに続いて、レジスタのデータ入力に供給されている組み合わせロジック出力が値を変化させます。信号は多くの遷移を経て最終的に新しい値に整定するため、この変化によってロジックを通じた伝播遅延による不安定期間が発生します。レジスタのデータ入力で起こる変化は、次のアクティブ・クロック・エッジまで、それらの出力の値に影響しません。

レジスタの内部回路がデータ出力を入力から分離するため、以下のタイミング要件が満たされている限り、組み合わせロジックの不安定性がデザインの動作に影響を与えることはありません。

- アクティブ・クロック・エッジの前に、少なくともレジスタのセットアップ時間の間データ入力が安定している。
- アクティブ・クロック・エッジの後で、少なくともレジスタのホールド時間の間データ入力が安定している。

すべてのクロック周波数およびその他のタイミング要件を指定すると、Quartus® II タイミング・アナライザは、デザインのすべてのピンのセットアップ時間 (t_{SU}) およびホールド時間 (t_H) に対する実際のハードウェア要件を発行します。これらの外部ピンの要件を満たすこと、かつ同期デザイン手法に従うことによって、アルテラ・デバイス内のすべてのレジスタのセットアップ時間とホールド時間が確実に満たされます。

 すべての入力ピンのセットアップ時間およびホールド時間要件を満たすには、レジスタに信号を供給するどの組み合わせロジックの入力も、レジスタのクロックと同期関係を持つ必要があります。信号が非同期の場合、アルテラ・デバイスの入力で信号をラッチして、必要なセットアップ時間およびホールド時間に違反しないようにすることができます。

レジスタのセットアップ時間またはホールド時間に違反した場合、出力は High レベルと Low レベルの間の準安定状態と呼ばれる中間電圧レベルに設定される可能性があります。この不安定状態では、電源レールでのノイズのような小さな動揺によって、レジスタが High または Low 電圧レベルになり、結果として予測不能な有効状態が生じます。伝播遅延

の増加や不正な出力状態など、さまざまな有害な影響が発生する可能性があります。場合によっては、出力が比較的長時間にわたって、2 つの有効な状態の間で発振することさえあります。



Quartus II ソフトウェアにおけるタイミング要件と解析について詳しくは、「Quartus II ハンドブック Volume 3」の「クラシック・タイミング解析」または「TimeQuest タイミング解析」の章を参照してください。

非同期デザインのハザード

過去において、設計者は、プログラマブル・ロジック・デバイス (PLD) のデザインにリップル・カウンタやパルス・ジェネレータなどの非同期手法を多用して、デバイス・リソース節約の「近道」をすることができました。非同期デザイン手法には、デバイスにおける伝播遅延への依存などの特有の問題があり、タイミング制約が不完全になったり、グリッチやスパイクが発生する可能性があります。今日の FPGA は、多くの高性能ロジック・ゲート、レジスタ、およびメモリを搭載しており、リソースと性能のトレードオフが変化してきました。現在では、問題のある非同期手法を使用してデバイス・リソースを節約するよりも、一貫したデザイン目標の達成に有効なデザイン手法に重点を置くことが重要です。

一部の非同期デザイン構造は、適切に機能するために信号の相対的伝播遅延に依存しています。これらのケースでは、信号の順番が変わるとロジック出力に影響が及ぶ箇所、競合状態が発生する可能性があります。PLD デザインにはさまざまなタイミング遅延があり、各コンパイルでデザインがデバイス内に配置配線される方法に応じて値が異なります。したがって、特定のロジック・ブロックに関連するタイミング遅延を、前もって確定させることはほとんど不可能です。デバイス・プロセスの改善に伴ってデバイスの高速化が進むと、非同期デザインでの遅延が減少し、その結果デザインが期待どおりに動作しない場合があります。5-4 ページの「デザイン・ガイドライン」に具体例を示します。また、特定の遅延への依存によって、非同期デザインを異なるアーキテクチャ、デバイス、またはスピード・グレードに移行することも非常に困難になります。

非同期デザイン構造のタイミングは、タイミング・アサインメントや制約でモデル化することが難しいか、あるいは不可能な場合も少なくありません。完全なまたは正確なタイミング制約がない場合、合成および配置配線ツールで使用されるタイミング・ドリブン・アルゴリズムは、最高の最適化を実行できず、レポートされた結果も完全でない場合があります。

非同期デザイン構造によっては、クロック周期に比べて非常に短いパルスである有害なグリッチを生成する可能性があります。大部分のグリッチは組み合わせロジックによって生成されます。組み合わせロジックの入力が変化すると、出力には新しい値に整定するまでに多数のグリッチ

デザイン・ ガイドライン

が現れます。これらのグリッチは、組み合わせロジックを通じて伝播し、非同期デザインの出力に不正値が現れる可能性があります。同期デザインでは、クロック・エッジまでデータが処理されないため、レジスタのデータ入力のグリッチは悪い症状ではなく正常な現象です。

ハードウェア記述言語 (HDL) コードでデザインする場合、合成ツールが異なる HDL デザイン手法を解釈する方法と、期待される結果を理解することが重要です。デザイン手法は、ロジック利用率とタイミング性能、およびデザインの信頼性に影響を与えます。この項では、信頼性の欠如や不安定性の原因となるいくつかの一般的な原因を回避しながら、アルテラ・デバイスをターゲットとするデザインに対して最適な合成結果を達成する、いくつかの基本的なデザイン手法について説明します。潜在的な問題を回避するために、組み合わせロジックを慎重に設計し、クロッキング方式に注意を払うことにより、同期式機能を維持し、タイミング問題を回避することができます。

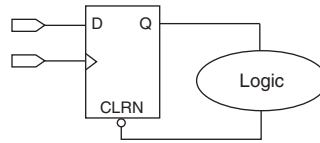
組み合わせロジックの構造


組み合わせロジックの構造は、入力の現在の状態にのみ依存するロジック・ファンクションで構成されています。アルテラ FPGA では、これらのファンクションは、デバイスのアーキテクチャのルック・アップ・テーブル (LUT) に実装されており、ロジック・エレメント (LE) またはアダプティブ・ロジック・モジュール (ALM) のいずれかを使用します。組み合わせロジックがレジスタに信号を供給するいくつかのケースでは、レジスタ・コントロール信号をロジック・ファンクションの一部を実装するのに使用して、LUT リソースを節約することもできます。この項の推奨事項に従うことにより、組み合わせデザインの信頼性を向上させることができます。

組み合わせループ

組み合わせループは、デジタル・デザインにおける不安定性および信頼性の欠如の最も一般的な原因であり、可能であれば常に回避すべきです。同期デザインでは、フィードバック・ループにレジスタが含まれていない限りなりません。組み合わせループにおける一般的な同期デザイン原理への違反は、レジスタを含まないダイレクト・フィードバック・ループが確立されることです。例えば、組み合わせループは HDL コードで演算式の左辺が右辺にも現れる場合に発生します。また、組み合わせループは、レジスタの出力を組み合わせロジックを通して、同じレジスタの非同期ピンにフィードバックするときにも発生します (図 5-1 を参照)。

図 5-1. 非同期コントロール・ピンを介した組み合わせループ



 clear または reset などの非同期ポートで、Quartus II ソフトウェアによるタイミング解析を実行するには、Assignments メニューの **Settings** をクリックします。Setting ダイアログ・ボックスで、**Timing Requirements & Option** を選択し、**More Settings** をクリックします。**Enable Recovery/Removal Analysis** をオンにします。

組み合わせループは、以下の理由から本質的に危険性の高いデザイン構造です。

- 組み合わせループの動作は、一般にループに関連するロジックを介した相対的な伝播遅延に依存しています。これまでに説明したように、伝播遅延は変化する可能性があるため、ループの動作は予測不能です。
- 組み合わせループは、多くのデザイン・ツールにおいて無限演算ループの原因となる可能性があります。大部分のツールは、デザインを処理するために組み合わせループをこじ開けます。デザイン・フローで使用されるさまざまなツールは、ループを開く方法が異なり、オリジナルのデザインの意図とは合致しない方法でループを処理する場合があります。

ラッチ

ラッチは、新しい値が割り当てられるまで信号の値を保持する、小さな組み合わせループです。ラッチの使用を意図していなくても、HDL コードの記述からラッチが生成されることがあります。FPGA アーキテクチャはレジスタをベースにしています。FPGA デバイスでは、ラッチは実際にはレジスタよりも多くのロジック・リソースを使用するため、性能の低下が生じます。これは、レジスタよりもラッチのほうが遅延が小さく、より狭いシリコン面積に実装できる他のデバイス・アーキテクチャとは異なります。

ラッチはデザインでさまざまな困難をもたらす原因になる可能性があります。ラッチはメモリ・エレメントですが、レジスタとは根本的に異なります。ラッチがフィード・スルーまたはトランスペアレント・モードの場合、データ入力と出力の間にダイレクト・パスがあります。データ

入力のグリッチは出力を通過できます。また、ラッチのタイミングも本質的に明確ではありません。例えば、D ラッチのデザインを解析するとき、ソフトウェアはデータをクロックの前方エッジまたは後方エッジのいずれで出力に転送したいか判断できません。多くの場合、デザインのすべての意図を理解しているのは元の設計者だけなので、別の設計者が簡単にデザインを修正したり、コードを再利用することはできません。

合成ツールでグリッチの問題を発生しないラッチを推測できる場合もあります。アルテラの `lpm_latch` ファンクションを推測すれば、アルテラ・アーキテクチャでグリッチのない実装を行うことができます。いくつかのサードパーティ合成ツールは、推測される多数の `lpm_latch` ファンクションをリストします。Quartus II 合成機能を使用すると、これらのラッチは **User-Specified and Inferred Latches** と呼ばれるコンパイルレポートのセクションに報告されます。デザイン内のラッチまたは組み合わせループがこのレポートにリストされていない場合、ソフトウェアでは安全なラッチとして推測されず、グリッチなしと判断されていないことを意味します。

ただし、グリッチのないラッチでもタイミング解析時に完全に解析できない場合もあります。Quartus II ソフトウェアには、**Analyze latches as synchronous elements** と呼ばれるオプションがあり、このオプションを使用して、ラッチをタイミング解析 (FPGA デザイン・ツールで実行される通常の解析) の開始ポイントおよび終了ポイントとして扱うことができます。このオプションがオンになっていると、ラッチは (反転クロックにより) レジスタとして解析されます。Quartus II ソフトウェアは、Synopsys PrimeTime などのサードパーティ・タイミング解析ツールで実行されるような、サイクル-ボロウイング解析は実行しません。

さらに、ラッチのフォーマル検証ツールのサポートは限定的なものです。したがって、フォーマル検証を使用するときは、ラッチを使用しないことが特に重要です。

アルテラでは、ラッチの使用を避け、デザインのタイミング性能と信頼性を完全に解析し、検証できるようにすることを推奨しています。

遅延チェーン

遅延チェーンは、単一ファン・インと単一ファン・アウトを持つ 2 つ以上の連続したノードが使用され遅延の原因となっている場合に発生します。遅延を追加するために、インバータが連結されていることがよくあります。遅延チェーンは、他の非同期デザイン手法によって作成された競合状態を解決するために、ときどき使用されます。

前述したとおり、PLD デザインでの遅延は各配置配線サイクルで変化することがあります。立ち上がり / 立ち下がり時間の違いやオンチップでのバラツキの影響は、特にクロック・パスで生じた遅延チェーンがデザ

インで重大な問題を引き起こす可能性があることを意味します。5-3ページの「非同期デザインのハザード」に、遅延チェーンが原因となる問題の種類例を示します。遅延チェーンの使用を避けて、この種類の問題を防止してください。

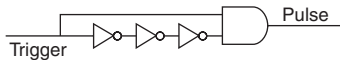
ASIC デザインによっては、デバイスの周囲に配線される信号をバッファするために遅延が使用されます。配線構造がデバイス全体でバッファを提供するため、この機能は FPGA デバイスでは必要ありません。

パルス・ジェネレータおよびマルチバイブレータ

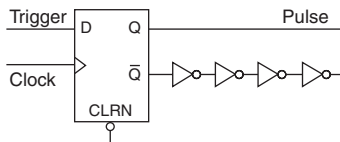
遅延チェーンは、1 個のパルス（パルス・ジェネレータ）または一連のパルス（マルチバイブレータ）の生成に使用されることがあります。図 5-2 に示すように、パルスの生成には 2 つの一般的な方法があります。これらの手法は純粋に非同期であり、必ず回避しなければなりません。

図 5-2. 非同期パルス・ジェネレータ

Using an AND Gate



Using a Register



「AND ゲート使用時」(図 5-2) には、2 入力 AND ゲートの両入力にトリガ信号が供給されますが、デザインではそれらの入力の 1 つに入る遅延チェーンを反転させるか、それに追加します。パルス幅は、ゲートに直接供給されるパスと遅延を通過するパスの相対的な遅延によって決まります。これは、入力値の変化に続いて組み合わせロジックでグリッチが生成されるのと同じメカニズムです。この手法では遅延チェーンの使用により、人工的にグリッチの幅を拡張します。

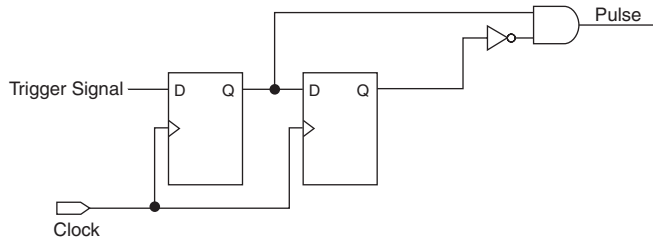
「レジスタの使用」(図 5-2) では、レジスタの出力は遅延チェーンを通して、同じレジスタの非同期リセット信号をドライブします。このレジスタは一定の遅延後に、非同期で自身をリセットします。

この方法で生成されたパルス幅で、合成および配置配線ソフトウェアに判断、設定、および検証させるのは困難です。実際のパルス幅は、配線と伝播遅延が既知のとき、配置配線後のみ決定されます。HDL コードの作成時に、パルス幅を確定することはできず、EDA ツールで設定することもできません。すべての PVT 状態において、パルス幅がアプリケーションにとって不十分な場合があります、また別のデバイスに変更するとパルス幅が変化します。また、スタティック・タイミング解析は、パルス幅の検証には使用できないため検証は非常に困難です。

マルチバイブレータは、回路をオシレータに変える組み合わせループと共に、「グリッチ・ジェネレータ」を使用してパルスを作成します。これにより多数のパルスが関係するため、新たな問題が生じます。さらに、この構造が複数のパルスを生成すると、デザインに新しい人工的なクロックも作成されます。このクロックはデザイン・ツールで解析する必要があります。

図 5-3 に示すように、パルス・ジェネレータを使用する必要がある場合は、同期手法を使用します。


図 5-3. 推奨されるパルス生成手法



このデザインでは、パルス幅は常にクロック周期と等しくなります。このパルス・ジェネレータは予測可能であり、タイミング解析によって検証でき、他のアーキテクチャ、デバイス、またはスピード・グレードに容易に移行できます。

クロッキング方式

組み合わせロジックと同様、クロッキング方式もデザインの性能と信頼性に大きな影響を与えます。デザインで機能的およびタイミング問題の原因になるおそれがあるため、可能な箇所では内部生成クロックの使用は避けてください。組み合わせロジックで生成されるクロックは、機能上問題となるグリッチを発生することあり、また組み合わせロジックに固有の遅延は、タイミング問題を引き起こす可能性があります。以下の項では、これらの問題を回避するためのいくつかの具体例と推奨事項について説明します。

 フィットニング中に最良のタイミング・ドリブンを最適化を実行し、適切なタイミング解析を実行するために、Quartus II ソフトウェアですべてのクロックの関係を指定します。派生クロックまたは内部クロックにクロック設定アサインメントを使用して、ベース・クロックとの関係を指定します。

アルテラでは、すべての内部生成クロックに対しては、通常の配線ライン上の配線クロックではなく、グローバル・デバイス・ワイドで低スキューの専用配線を使用することを推奨しています。詳しくは、5-16 ページの「クロック・ネットワーク・リソース」を参照してください。

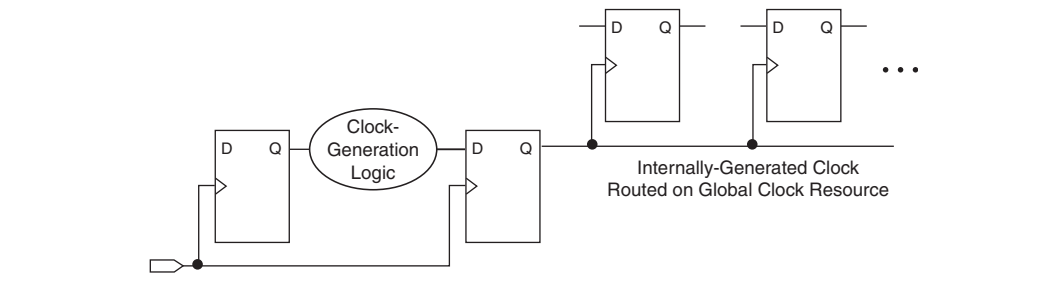
可能な場合は常に、異なるクロック間でのデータ転送を避けます。異なるクロック間でデータ転送が必要な場合は、FIFO 回路を使用します。クロック・ドメイン間のさまざまな遅延を補正するために、Quartus II ソフトウェアでクロック不確実性 (Clock Uncertainty) 機能を使用することができます。クロック・セットアップ不確実性とクロック・ホールド不確実性を、クロック遅延の 10% から 15% の値に設定することを検討してください。

内部生成クロック

組み合わせロジックからの出力をクロック信号または非同期リセット信号として使用する場合、デザインにグリッチが発生することを予測しておく必要があります。同期デザインでは、レジスタのデータ入力のグリッチは、悪い症状ではなく正常な現象です。ただし、レジスタのクロック入力（または非同期入力）上のグリッチまたはスパイクは、重大な結果を引き起こすことがあります。狭いグリッチはレジスタの最小パルス幅要件に違反する可能性があります。グリッチがクロック入力に達したときにレジスタのデータ入力に変化した場合、セットアップ時間およびホールド時間にも違反する可能性があります。デザインがタイミング要件に違反しない場合でも、レジスタ出力が不意に値を変化させる可能性があります。これがデザインの他の場所で機能ハザードの原因となります。

これらの問題のために、組み合わせロジックの出力はクロック信号として使用する前に必ずラッチしておきます。図 5-4 を参照してください。

図 5-4. 推奨されるクロック生成手法



組み合わせロジックの出力をラッチすると、組み合わせロジックで生成されるグリッチはレジスタのデータ入力でブロックされます。

分周クロック

デザインに、マスタ・クロックを分周して作成されたクロックが必要な場合がよくあります。大部分のアルテラ FPGA は、クロック分周のための専用 PLL (Phase-Locked Loop) 回路を備えています。専用 PLL 回路を使用すると、非同期クロック分周ロジックで生じる可能性がある多くの問題を回避するのに有効です。

マスタ・クロックを分周するロジックを使用する必要があるときは、必ず同期カウンタまたはステート・マシンを使用します。さらに、5-9 ページの「内部生成クロック」で説明するように、レジスタが常に分周クロック信号を直接生成できるようにデザインを作成し、グローバル・クロック・リソース上でクロックを配線します。グリッチを避けるために、カウンタまたはステート・マシンの出力をデコードしてクロック信号を生成しないでください。

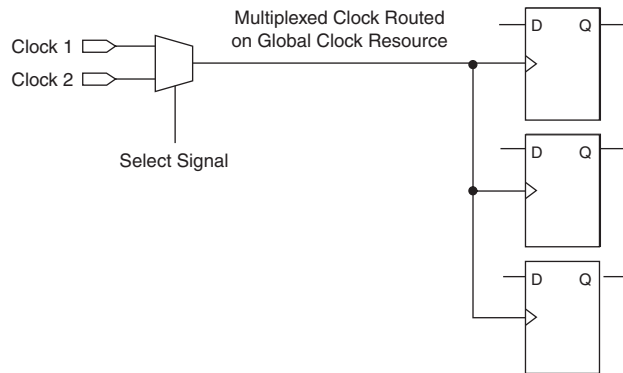
リップル・カウンタ

アルテラでは、検証を簡略化するために、デザインでリップル・カウンタを使わないことを推奨しています。これまで、FPGA の設計者はリップル・カウンタの方が簡単に設計でき、同期式カウンタの場合より使用ゲート数が少なくすむという理由から、リップル・カウンタを実装し、2 のべき乗でクロックを分周していました。リップル・カウンタは、各レジスタの出力ピンが次のステージのレジスタのクロック・ピンに供給されるカスケード接続レジスタを使用します。カウンタが各ステージでリップル・クロックを生成するため、このカスケード接続は問題の発生原因となる可能性があります。これらのリップル・クロックはタイミング解析時に適切に処理しなければなりません。これが容易でなく、合成および配置配線ツールで複雑なタイミング・アサインメントが必要になる場合もあります。

多重化クロック

クロックの多重化は、同じロジック・ファンクションを別のクロック・ソースで動作させるのに使用できます。これらのデザインでは、[図 5-5](#)に示すように、マルチプレキシングでクロック・ソースを選択しています。例えば、複数の周波数規格を扱う通信アプリケーションでは、しばしば多重化クロックを使用します。

図 5-5. 多重化ロジックおよびクロック・ソース



多重化ロジックをクロック信号に追加すると、前の項で説明した問題が発生する可能性があります。多重化クロックの要件はアプリケーションによって大きく異なります。クロック多重化はクロック信号がグローバル・クロック配線リソースを使用するとき、以下の条件を満たせば使用できます。

- 最初のコンフィギュレーション後、クロック多重化ロジックが変更されていない。
- デザインで多重化ロジックを使用してテスト用クロックを選択している。
- クロック切り替え時に常にレジスタがリセットされる。
- クロック切り替えに続く一時的な不正応答で、悪い結果が発生していない。

デザインがクロックをリセット信号なしでリアルタイムに切り替え、かつデザインが一時的な不正応答に耐えられない場合は、同期デザインを使用して、レジスタにタイミング違反がなく、クロック信号にグリッチがなく、競合状態やその他のロジックの問題がないようにする必要があります。デフォルトでは、Quartus II ソフトウェアは、マルチプレクサを経由する可能なすべてのパス、およびマルチプレクサから来る可能性がある両方の内部クロック間で可能なすべてのパスの最適化と解析を行います。マルチプレクサが常に特定の1つのクロックを選択する場合は、

これによって必要以上に制約の多い解析が行われることがあります。これ以上完成度の高い解析が必要がない場合は、マルチプレクサの出力を Quartus II ソフトウェアのベース・クロックとして割り当てることができます。それによって、すべてのレジスタ間パスがクロックを使用して解析されます。

アルテラでは、使用可能な場合は多重化ロジックを使用する代わりに、クロック多重化を実行する専用ハードウェアの使用を推奨しています。例えば、Stratix® デバイス・シリーズの PLL のクロック切り換え機能、または Stratix II デバイスおよび Cyclone™ II デバイスのクロック・コントロール・ブロックを使用することができます。これらの専用ハードウェア・ブロックによって、グローバル低スキュー配線ラインを使用して、クロック・ライン上のロジック遅延によりデバイスで発生する可能性があるホールド時間の問題を回避します。

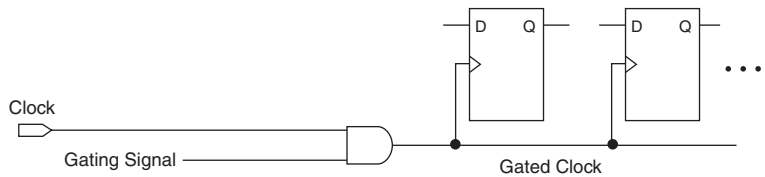


デバイス固有のクロッキング構造に関する情報については、該当するデバイスのデータ・シートまたはハンドブックを参照してください。

ゲート付きクロック

ゲート付きクロックは、図 5-6 に示すように、ある種のゲーティング回路を制御するイネーブル信号を使用して、クロック信号をオン・オフします。クロックがオフになると、対応するクロック・ドメインがシャット・ダウンされ、機能的に非アクティブになります。

図 5-6. ゲート付きクロック



一部のデバイス・アーキテクチャでは、ゲート付きクロックを使用して消費電力を低減することができます。クロックがゲートされると、クロック・ネットワークとそれによってドライブされるレジスタの両方がトグルを停止し、消費電力への影響がなくなります。ただし、ゲート付きクロックは同期方式の一部ではないため、デザインの実装と検証に必要な労力が大幅に増える可能性があります。ゲート付きクロックは、クロック・スキューを増やすため、デバイスの移行が困難になります。これらのクロックはグリッチにも敏感で、デザインが失敗する原因になる場合があります。

アルテラでは、多重化ロジックを使用する代わりに、ターゲット・デバイスで使用可能な場合は、クロック・ゲーティングを実行する専用ハードウェアの使用を推奨しています。例えば、Stratix II および Cyclone II デバイスでクロック・コントロール・ブロックを使用して、クロック・ネットワーク全体をシャット・ダウンすることができます。専用ハードウェア・ブロックによって、グローバル低スキュー配線ラインを使用して、クロック・ライン上のロジック遅延によりデバイスで発生する可能性があるホールド時間の問題を回避します。



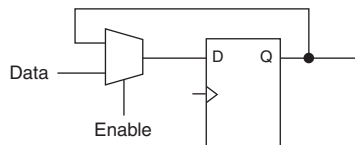
デバイス固有のクロッキング構造に関する情報については、該当するデバイスのデータシートまたはハンドブックを参照してください。

機能的な観点からは、同期クロック・イネーブル信号を使用した純粹に同期式の方法で、クロック・ドメインをシャット・ダウンできます。ただし、同期クロック・イネーブル方式を使用する場合、クロック・ネットワークはトグルを継続します。この方法では、ソースにおけるクロックのゲーティングほどには消費電力を削減しません。「同期クロック・イネーブル」の項で説明するように、ほとんどの場合は同期方式を使用する必要があります。ロジックでクロックをゲーティングする際に、さらに消費電力を削減するには、5-14 ページの「推奨されるクロック・ゲーティング方法」を参照してください。

同期クロック・イネーブル

同期方式でクロック・ドメインをオフにするには、同期クロック・イネーブル信号を使用します。すべてのデバイス・レジスタで専用クロック・イネーブル信号が使用可能なため、FPGA は効率的にクロック・イネーブル信号をサポートします。この方式では、クロック・ネットワークはトグルを継続するため、ソースでクロックをゲーティングしたほど消費電力は削減されませんが、一連のレジスタをディセーブルすることによってゲート付きクロックと同じ機能を実行します。各レジスタのデータ入力の前にマルチプレクサを挿入して、新しいデータをロードするかまたはレジスタの出力をコピーします (図 5-7)。

図 5-7. 同期クロック・イネーブル

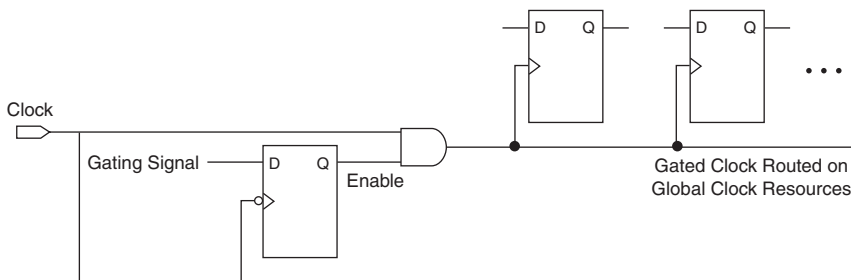


推奨されるクロック・ゲーティング方法

ターゲット・アプリケーションで消費電力の削減が必要なとき、ゲート付きクロックによってデバイス・アーキテクチャで必要な削減を達成できる場合は、ゲート付きクロックのみを使用します。ロジックでゲート制御されるクロックを使用する必要がある場合は、[図 5-8](#) に示す堅牢なクロック・ゲーティング手法を使用してこれらのクロックを実装し、ゲート付きクロック信号が確実に専用グローバル・クロック配線を使用するようにします。

クロック・ネットワークのソース、各レジスタ、またはその間のどこかで、クロック信号をゲート制御することができます。クロック・ネットワークは、スイッチング消費電力に影響を及ぼすため、可能な場合はソースでクロックをゲート制御します。これにより、レジスタでクロック・ネットワークと共にさらにゲーティングする代わりに、クロック・ネットワーク全体をシャット・ダウンできます。

図 5-8. 推奨されるクロック・ゲーティング手法



[図 5-8](#) に示した手法では、信号にグリッチやスパイクが発生しないように、レジスタがイネーブル信号を生成します。イネーブル信号を生成するレジスタは、ゲートされるクロックの非アクティブ・エッジでトリガされます ([図 5-8](#) に示すように、立ち上がりエッジでアクティブなクロックをゲーティングするときは、立ち下がりエッジを使用します)。この手法を使用するには、クロックをオン・オフするゲートの 1 入力だけを同時に変更し、出力にグリッチやスパイクが乗らないようにします。AND ゲートを使用して、立ち上がりエッジでアクティブなクロックをゲートします。立ち下がりエッジでアクティブになるクロックの場合は、OR ゲートを使用してクロックをゲート制御し、正エッジ・トリガ・レジスタでイネーブル・コマンドをレジスタに入れます。

この手法を使用するときは、クロック・サイクルの半分でイネーブル信号を生成しなければならないため、クロックのデューティ・サイクルとイネーブル信号を生成するロジックを通じた遅延に注意してください。この状況は、イネーブル・コマンドを生成するロジックが特に複雑な場

合、またはクロックのデューティ・サイクルが極端にアンバランスな場合は、問題の発生原因になることがあります。ただし、デューティ・サイクルとロジック遅延を注意深く管理すると、その他のゲーティング・クロックの方法で生じる問題と比べた場合に、有効な解決策になることがあります。

Quartus II ソフトウェアで、ゲート付きクロックにクロック設定を適用していることを確認します。図 5-8 に示すように、AND ゲートの出力にクロック設定を適用します。それ以外の場合、タイミング・アナライザはレジスタを経由するクロック・パスを最長クロック・パスとし、レジスタをスキップするパスを最短クロック・パスとして使用して人工的なクロック・スキューを生成する回路を解析することができます。

階層デザインの の分割

階層デザインは、階層内で互いにリンクする複数のデザイン・ブロックによって構成されています。デザインが階層に分割されている場合は、個々のデザイン・ブロックを別々にコンパイル、最適化、およびシミュレートすることができます。インクリメンタル・コンパイルまたは LogicLock™ デザイン・フローを使用して、各ブロックが独立して配置配線されるブロック・ベースのデザイン手法に従うと、階層内のすべてのブロックがトップ・レベルで結合されます。合成ツールには、個別のネットリスト・ファイルの作成、またはデザインの異なる部分に対するネットリスト・ファイルの個々の部分の維持に役立つ機能を備え、ブロック・ベースのデザイン手法またはインクリメンタル・コンパイルをサポートするものもあります。



インクリメンタル・コンパイルについて詳しくは、「Quartus II ハンドブック Volume 1」の「Quartus II Incremental Compilation for Hierarchical & Team-Based Design」の章を参照してください。LogicLock デザイン手法について詳しくは、「Quartus II ハンドブック Volume 2」の「LogicLock Design Methodology」の章を参照してください。合成ツールでのインクリメンタル合成フローについて詳しくは、「Quartus II ハンドブック Volume 1」の「合成」の項の該当する章を参照してください。

階層またはインクリメンタル・デザイン手法を使用する場合、良い結果を得るためのデザインの分割方法を検討する必要があります。

アルテラでは、デザインの分割に以下の方法を推奨しています。

- 機能的な境界でデザインを分割します。
- 異なるパーティション間の I/O 接続を少なくします。
- 各ブロックのすべての入力と出力を登録します。これによって、ロジックが同期し、グリッチを回避して、パーティション間を通過する信号の遅延ペナルティを回避します。I/O を登録することによって、通常、異なるブロック間を接続する信号のタイミング要件を指定する必要がなくなります。

- 階層ブロック間に「グルー・ロジック」または接続ロジックは使用しないでください。階層境界を維持する場合、グルー・ロジックは階層ブロックにマージされません。合成ソフトウェアは、グルー・ロジックを個別に最適化する場合があり、それによって合成の結果が低下する可能性があり、LogicLock デザイン手法で使用した場合は効率的ではありません。
- ロジックはパーティション境界を越えて合成または最適化されないことを忘れないでください。これは、定数値（GND に設定された信号など）がパーティションを越えて伝播されないことを意味します。
- 階層境界では、トライ・ステート信号または双方向ポートは使用しないでください。低レベル・ブロックで境界トライ・ステートを使用する場合、アルテラ・デバイスの出力ピンのトライ・ステート・ドライバを活用するために、合成は階層を通じてトライ・ステートをトップ・レベルに押し上げます。これには階層を通じた最適化が必要なため、低レベルの境界トライ・ステート信号には、ブロック・レベル・デザイン手法での制約があります。
- クロックを1ブロックあたり1個に制限します。デザインをクロック・ドメインに分割することによって、合成とタイミング解析が簡単になります。
- 最適化を高速化し、エンコーディング・コントロールを強化するために、ステート・マシンを個別のブロックに配置します。
- タイミングがクリティカルなブロックとクリティカルでないブロックを分離します。
- クリティカル・タイミング・パスを1つの階層ブロックに制限します。複数のデザイン・ブロックからのロジックをグループ化し、クリティカル・パスが1つのブロックに集中するようにします。



Quartus II インクリメンタル・コンパイルでデザイン・パーティションを作成するためのガイドラインについて詳しくは、「Quartus II ハンドブック Volume 1」の「Quartus II Incremental Compilation for Hierarchical & Team-Based Design」の章を参照してください。

クロックおよび レジスタ・ コントロール・ アーキテクチャ 機能の ターゲット化

下記の一般的なデザイン・ガイドラインに加えて、デバイスのアーキテクチャを念頭においてデザインをコード化することが重要です。FPGA は、性能を向上させることができるデバイス・ワイドのクロックおよびレジスタ・コントロール信号を提供します。

クロック・ネットワーク・リソース

アルテラ FPGA は、デバイス・ワイドのグローバル・クロック配線リソースと専用入力を提供しています。可能な場合は、FPGA の低スキュー、高ファン・アウト、専用配線を使用する必要があります。クロッ

ク入力将这些の専用クロック・ピンの1本に割り当てるか、Quartus II ロジック・オプションを使用してグローバル配線を割り当てることにより、クロック信号に使用可能な専用配線をうまく利用できます。

ASIC デザインでは、クロック遅延はデバイス全体に分散されるため、クロック遅延のバランスを図ることが重要になります。アルテラ FPGA では、デバイス・ワイドのグローバル・クロック配線リソースと専用入力を提供しているため、手動でクロック・ネットワークでの遅延のバランスを図る必要がないためです。

アルテラでは、デザイン内のクロック数を FPGA で使用可能な専用グローバル・クロック・リソース数に制限することを推奨しています。グローバル配線を使用しないで複数の位置に供給するクロックは、デバイス全域でクロック・スキューを生じることがあり、タイミング問題を引き起こす可能性があります。さらに、組み合わせロジックを使用して内部クロックを生成するときは、クロック・ラインに遅延が追加される場合があります。また、クロック・ラインの遅延によって、クロック・スキューが2個のレジスタ間のデータ・パス長より長くなることもあります。クロック・スキューがデータ遅延よりも大きい場合、レジスタのタイミング・パラメータ（ホールド時間要件など）に違反するため、デザインが正常に機能しません。

多くのクロック・ドメインを持つ大規模なデザインに対応するために、今日の FPGA が提供するグローバル・クロック数は増え続けています。多くの大規模 FPGA デバイスは、専用グローバル・クロック・ネットワーク、領域クロック・ネットワーク、および専用高速領域クロック・ネットワークを提供しています。これらのクロックは通常、各デバイス領域に小さなスキューと遅延を持つ多くのクロックを配置できる階層的クロック構造に編成されます。これらは通常、グローバルまたは領域クロック・ネットワークのいずれかをドライブする何本かの専用クロック・ピンになっており、PLL 出力と内部クロックの両方がさまざまなクロック・ネットワークをドライブできます。

クロック・ドメイン内のクロック・スキューを低減するには、ホールド時間が当該クロック・ドメインの範囲内になるようにし、各クロック信号をグローバル高ファン・アウトおよび FPGA デバイス内の低スキュー・クロック・ネットワークのいずれか1つに割り当てます。Quartus II は、高ファン・アウト・コントロール信号、PLL 出力、およびデバイスのグローバル・クロック・ピンに供給する信号に対しては自動的にグローバル配線を使用します。明示的にグローバル信号ロジック・オプションを設定できます。明示的なグローバル信号ロジック・オプション設定を行うには、Assignment メニューで **Assignment Editor** をクリックします。ソフトウェアに特定の信号にグローバル配線を使用するよう強制する必要がある場合は、このオプションを使用します。

これらの配線リソースをフルに活用するには、デザイン内のクロック信号のソース（入力クロック・ピンまたは内部生成クロック）は、レジスタのクロック入力ポートのみドライブする必要があります。従来のアルテラ・デバイス・ファミリ（FLEX[®] 10K や ACEX[®] 1K など）では、クロック信号がレジスタのデータ・ポートに供給されている場合、この信号には専用配線を使用できない場合があります、それによって性能低下やクロック・スキュー問題を引き起こす可能性があります。一般に、クロック信号でレジスタのデータ・ポートをドライブできる場合は同期デザインとは考えられず、タイミング解析が複雑になる可能性があります。これは推奨されている方法ではありません。

リセット・リソース

ASIC デザインは、ローカル・リセットを使用して、信号上の長い配線遅延を回避できます。大部分の FPGA で使用可能なデバイス・ワイドの非同期リセット・ピンを活用して、これらの問題を解決する必要があります。このリセット信号は、デバイス全域に低スキュー配線を提供します。

レジスタ・コントロール信号

デザインのターゲット・デバイスのアーキテクチャに非同期ロード用の専用回路を持つレジスタが含まれていない場合は、非同期ロード信号の使用は避けてください。また、アーキテクチャがこれらのコントロール信号のうち1つしか提供していない場合は、非同期クリアおよびプリセットの両方の使用も避けてください。例えば、APEX[™] デバイスは、非同期クリア機能を直接サポートしていますが、プリセットまたはロード機能はサポートしていません。ターゲット・デバイスが信号を直接サポートしない場合、配置配線ソフトウェアが組み合わせロジックを使用して、同じ機能を実装する必要があります。さらに、デバイス・アーキテクチャでの固有の優先順位以外の優先順位で信号を使用する場合、希望のコントロール信号を実装するために、組み合わせロジックが必要になる場合があります。組み合わせロジックは効率が悪く、グリッチやその他の問題の原因となる可能性があるため、これらの実装を回避することが得策です。



各種コントロール信号を持つレジスタの Verilog HDL および VHDL の例、およびアルテラ・デバイス・アーキテクチャにおけるレジスタ・コントロール信号の固有の優先順位に関する情報は、「Quartus II ハンドブック Volume 1」の「Recommended HDL Coding Styles」の章を参照してください。

まとめ

この章で概説したデザイン手法に従えば、確実にデザイン目標を達成するのに役立ちます。非同期デザイン手法では、タイミング解析が不完全になったり、データ信号にグリッチを発生させる可能性があります。また、デバイスで伝播遅延に頼って、競合状態や予測不能な結果を引き起こす可能性があります。FPGA デバイスのアーキテクチャ上の特徴を活用することによっても、結果の質を向上させることができます。

