次世代の高性能システムを実現する新しい HyperFlex アーキテクチャ

Mike Hutton, Architect, IC Design Engineering

WP-01231-1.0 ホワイトペーパー

次世代の高性能システムには、いっそう広い帯域幅が必要です。この要件を満たすために、設計者は様々な手法を使用してデザインを最適化し、クロック周波数性能を可能な限り向上させようと注力しています。従来の FPGA コア・アーキテクチャでもこうした最適化手法は利用できますが、実現可能な周波数の向上幅には限界があります。従来のアプローチとは異なり、Stratix® 10 FPGA & SoC は、新しい HyperFlex™ アーキテクチャを採用して、前世代の高性能 FPGA の 2 倍のコア・クロック周波数性能を実現します。このホワイトペーパーでは、新しい HyperFlex アーキテクチャと、設計者がこのアーキテクチャを使用して総合的な性能の目標を達成する方法について解説します。

はじめに

次世代の高性能システムの増大する帯域幅要件を満たすために、FPGA ベンダーは継続的改善によりデバイス・アーキテクチャを進化させ続けています。しかし、こうした先進アーキテクチャを採用する場合であっても、設計者は非常に広いオン・チップ・バスを使用してデザインを実装することが少なくありません。事実、512 ビット、1,024 ビット、または 2,048 ビット幅のオン・チップ・バスがますます一般的になっています。この手法では FPGA コアのデータ・スループットは向上するものの、広いバスによって大量のファブリック・リソースと電力が消費されることになります。さらに、FPGA 内のリソース使用率が上昇すると、配線リソースが密集し、コア・クロック周波数が制限される可能性があります。

帯域幅を拡大するもう 1 つの方法は、最先端のプロセス・ノードを使用して製造された FPGA にデザインを実装して、最新テクノロジで実現されるトランジスタのスイッチング速度の向上を活用することです。しかし、プロセスの微細化に伴って、ロジック・セル間のインタコネクト遅延が FPGA の全遅延の中で大きな部分を占めるようになっているため、トランジスタ・スイッチング速度の向上の効果も頭打ちになっています。根本的に、従来の FPGA アーキテクチャは将来の性能要件を満たすことができません。

帯域幅に対するニーズ

オプティカル・トランスポート・ネットワーク (OTN)、有線通信、防衛、および高性能コンピューティング・アプリケーションでは、ますます広い帯域幅が必要になっています。転送する必要があるデータの量が増加したことで、FPGA 内部のデータパス幅が広がりました。配線アーキテクチャを介して転送できるデータの量は、使用されるワイヤーの数とワイヤーの速度 (f_{MAX}) によって決まります。使用可能なワイヤーの数は、テクノロジに依存します。つまり、そのテクノロジにおけるデバイスのサイズとワイヤーの最小ピッチによって決まります。

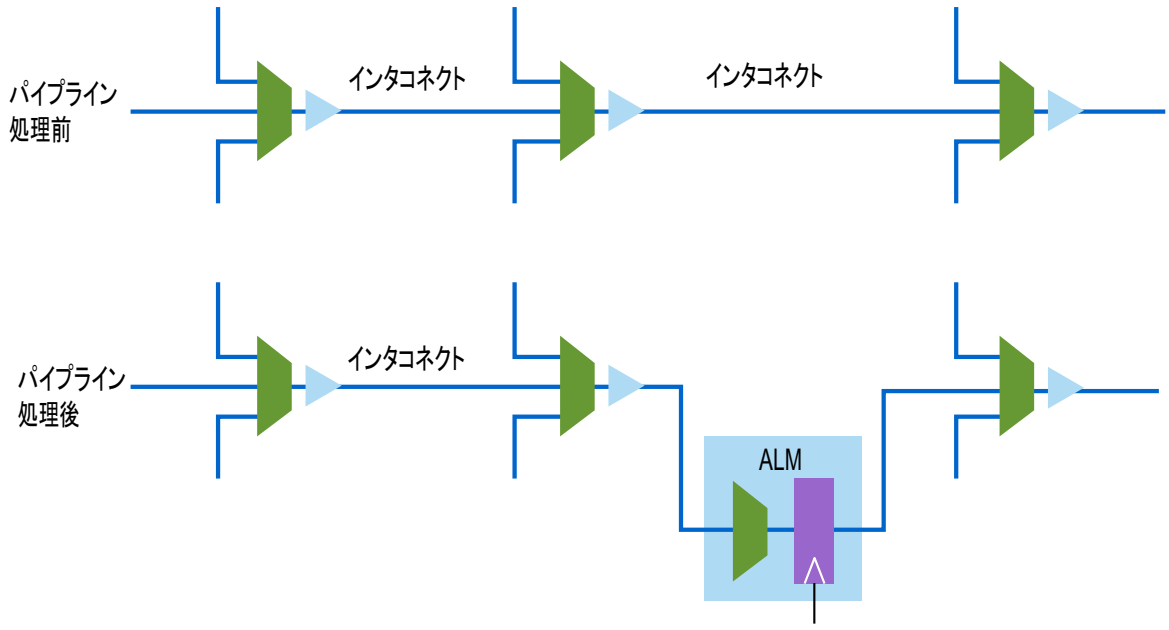
101 Innovation Drive
San Jose, CA 95134
www.altera.com

配線アーキテクチャは、階層 (例えば、ロジック・アレイ・ブロック (LAB) 内のローカル配線と水平および垂直インタコネクト・ライン上のグローバル配線) と最適化手法を使用することでワイヤー効率を高めることができます。しかし、ワイヤーの数を 2 倍にしても、ダイ面積が増大し、電力消費が増加するだけです。配線ワイヤーの速度はテクノロジ (ワイヤーの RC 遅延) に依存し、FPGA のアーキテクチャとデザイン実装によって決まります。例えば、デザインをパイプライニングすることにより、ワイヤー数を増やさずにクロックを高速化し、同じリソースに対する帯域幅を拡大できます。

効率に対するニーズ

設計者がデザインをパイプライニングして性能を向上させる場合は、デザインにレジスタを追加します。すべての既存 FPGA コア・アーキテクチャに採用されている、レジスタとルック・アップ・テーブル (LUT) ペアの構築という既存の手法では、追加されたパイプライン・レジスタに到達するためにロジックが犠牲になります。従来のアーキテクチャでのパイプライニングは、信号をロジック・ブロック経由でルーティングする必要があるため、遅延コストの発生にもつながります。この結果、特に配線遅延が全遅延の大きな部分を占める場合、パイプライニング手法の効果が薄れてしまいます。図 1 に、従来のパイプライニングの前と後、および追加されたレジスタに対するルーティングによって増加した遅延の例を示します。

図 1. 従来のパイプライニングによる遅延の増加



クロック向上に対するニーズ

クロックの高速化に伴って、クロック・スキューがますます重大になっています。従来の FPGA コア・アーキテクチャは、クロック・ツリーのバランスを調整することで確定的なスキューを最小化することに重点を置いてきました。この手法は、500 MHz までのデザインには効果的でした。しかし、500 MHz の障壁を打ち破り 1 GHz の速度に達するには、次世代のクロック・ソリューションが必要となります。このソリューションは、クロックをローカライズしてローカルのばらつきとスキューを最小化するとともに、高性能デザイン内で共通する多数のクロックを供給する柔軟性の高いネットワークを提供する必要があります。

HyperFlex ソリューション

これらの課題を解決するために、アルテラの Stratix 10 FPGA & SoC は、まったく新しいコア・アーキテクチャである HyperFlex アーキテクチャを採用しています。この革新的な HyperFlex アーキテクチャは、前世代の高性能 FPGA の 2 倍のコア性能という、これまで想像できなかったレベルの性能を提供します。このような性能レベルは、既存のアーキテクチャでは不可能です。HyperFlex アーキテクチャの利点を活用するには、よく知られている手法であるレジスタ・リタイミング、パイプライニング、およびデザイン最適化を使用します。これらの手法では、従来のアーキテクチャのデザインも高速化できます。しかし、HyperFlex アーキテクチャと組み合わせて使用すれば、最高 1 GHz という驚異的なコア・クロック・レートで動作するデザインを実現できます。

HyperFlex アーキテクチャ

Stratix 10 デバイスは、再設計されたコア・アーキテクチャを採用しています。このアーキテクチャでは、Hyper-Register と呼ばれる追加のレジスタがコア・ファブリック全体に満遍なく配置されます。Hyper-Register は、すべてのインタコネクト配線セグメントと、すべてのファンクション・ブロックの入力で使用できます。こうした細粒度の Hyper-Register により、帯域幅を拡大し、面積効率と電力効率を向上させるという課題を解決できます。満遍なく配置され、簡単にアクセスできるレジスタを活用することで、設計者はレジスタのリタイミングによってクリティカル・パスを削除し、パイプライン・レジスタの追加によって配線遅延を解消し、デザインの最適化によってクラス最高のパフォーマンスを実現できます。Hyper-Register を使用してこれらの手法を実装する場合、すべての FPGA ロジック・リソースはロジック・ファンクションとして使用でき、既存の LUT レジスタに到達するためのフィードスルー・セルとして犠牲になることはありません。

コア・ファブリックの高性能に追いつくために、FPGA コア内の専用ファンクション・ブロック (M20K メモリや浮動小数点デジタル信号処理 (DSP) ブロックなど) が再設計され、最高 1 GHz のクロック速度での動作をサポートします。

Hyper-Register を使いやすくするために、Quartus® II 開発ソフトウェアには以下の利点を提供する Hyper-Aware デザイン・フローが含まれています。

- 配置配線後のパフォーマンス・チューニングによるタイミング・クロージャの加速化
- Hyper-Aware 合成および配置配線によるパイプライニングの効率化
- Fast Forward Compile による性能向上オプションの識別

柔軟なクロック・ネットワークに対するニーズを満たすために、Stratix 10 FPGA & SoC は、プログラマブル・クロック・ツリー・シンセシスをサポートします。この ASIC ライクなクロッキングを使用すると、スキューと不確定性を減少できます。また、クロック・ネットワーク分岐をインテリジェントに有効にすることで電力消費を低減できます。

Stratix 10 FPGA & SoC は、インテルの 14nm トライゲート (FinFET) プロセス・テクノロジを採用しています。新しい HyperFlex アーキテクチャと最先端の FinFET プロセス・テクノロジの組み合わせにより、Stratix 10 デバイスは前世代の高性能 FPGA の 2 倍のコア性能を実現します。

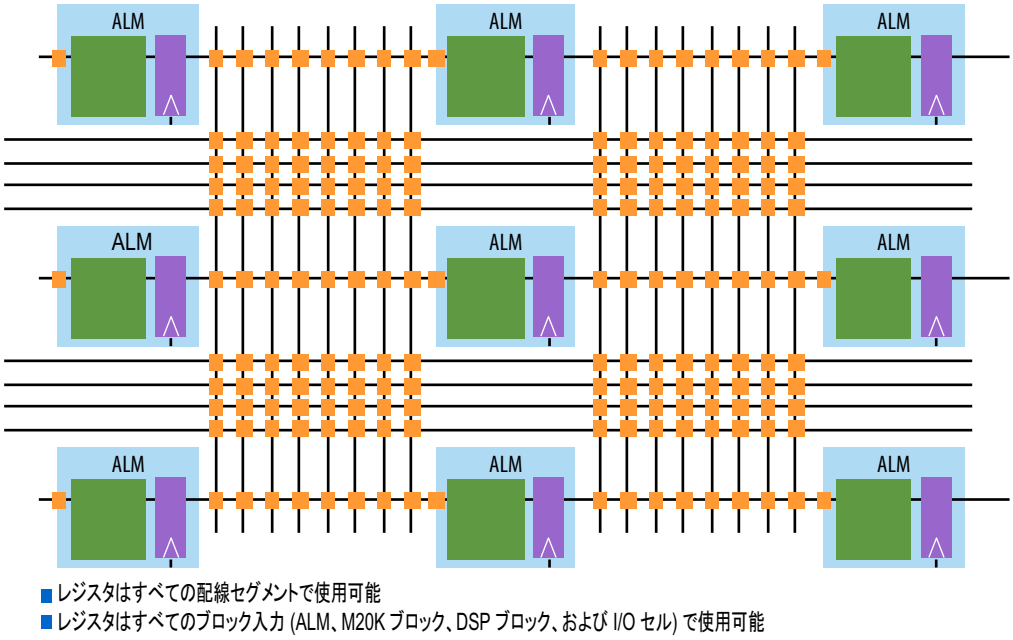
Hyper-Register

アルテラは、90nm Stratix II FPGA において、FPGA ベンダーとして初めて 8 入力の分割可能な LUT 構造 によりクリティカル・パスを短縮しました。28nm Stratix V FPGA では、タイム・ボローイング・ラッチの採用により、クロック信号とデータ信号の自動マイクロ・リタイミングを実現しました。

さらに 14nm Stratix 10 デバイスでは、FPGA ベンダーとして初めて、バイパス可能なリタイミング／パイプライン用レジスタを敷き詰めた革新的な「レジスタ・エブリウェア (どこでもレジスタ)」コア・アーキテクチャを採用しました。この手法では、アダプティブ・ロジック・モジュール (ALM) 内のファンクション・レジスタと、クリティカル・パスのリタイミングとパイプライニングによってデザイン効率を向上させる Hyper-Register 間のリンクが解除されます。

HyperFlex アーキテクチャは、高性能デザインのリタイミングとパイプライニングを実現するように設計されています。すべての配線セグメントには、オプションの Hyper-Register が組み込まれたプログラマブル配線マルチプレクサが配置されています。これにより、配線セグメントをレジスタ付きまたは組み合わせロジックにすることができます。これらの Hyper-Register は、コア・ファブリック内のあらゆる場所で使用できます (図 2 を参照)。Hyper-Register は、すべての水平および垂直配線セグメントの交点に位置する小さな正方形として示されています。

図 2 :「レジスタ・エブリウェア」HyperFlex アーキテクチャ



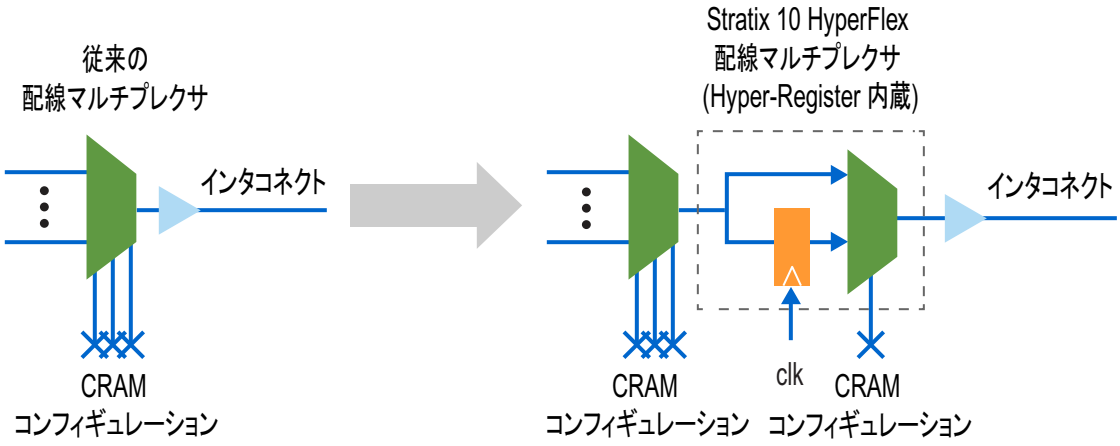
このアーキテクチャでは、ALM のレジスタをパイプライン・レジスタとして使用する必要がありません。デバイス内のすべての水平および垂直インタコネクト・ラインには、FPGA を構成することでオンまたはオフにできる Hyper-Register が組み込まれています。

Hyper-Register は、シンプルな 1 入力 1 出力のバイパス可能レジスタです。Hyper-Register は、コンフィギュレーション・ビットで制御します。Hyper-Register はデバイスのシリコン面積を著しく増大させることはありません。Hyper-Register はコア・ファブリックに遍在するため、設計者はデザイン内のレジスタ数の制約を受けることがありません。追加の LAB リソースを消費せずに、必要に応じてリタイミングとパイプライニングを実行できます。また多くの場合、デザインで使用する LAB リソースを減らすことができます。これは、Hyper-Register を使用してレジスタを配線に実装できるため、レジスタを使用するためだけに ALM を部分的に消費する必要がないためです。

HyperFlex の利点

Hyper-Register はインタコネクト配線アーキテクチャに埋め込まれているため、配置配線後、デザインの配線を変更せずにタイミング最適化を実行できます。Quartus II 開発ソフトウェアは、リタイミング操作中に Hyper-Register を簡単に見つけて使用できます。図 3 は、従来の配線マルチプレクサと、Hyper-Register が組み込まれた HyperFlex 配線マルチプレクサを比較したものです。

図 3. 従来の配線マルチプレクサと HyperFlex 配線マルチプレクサの比較



Hyper-Register を使用すると、新しく効果的な方法で実装される従来の性能向上手法 (リタイミング、パイプライニング、およびオプティマイゼーション) を活用できます。ALM レジスタの代わりに Hyper-Register を使用して実装されるこれらの手法を、それぞれ Hyper-Retiming、Hyper-Pipelining、および Hyper-Optimization と呼びます。表 1 に、これらの手法を順番に適用した場合に達成される性能向上を示します。HyperFlex アーキテクチャを使用すると、この 3 ステップ・プロセスにより性能を最大化できます。

表 1. HyperFlex アーキテクチャを使用して性能を最大化する 3 ステップ・プロセス

ステップ	アーキテクチャのメリット	必要な作業	コア性能 (前世代の高性能 FPGA との比較)
1	Hyper-Retiming	変更なし、または RTL の細かい変更	1.4 倍
2	Hyper-Pipelining	さらなるパイプライニング	1.6 倍
3	Hyper-Optimization	デザインによる	2 倍以上

Hyper-Retiming

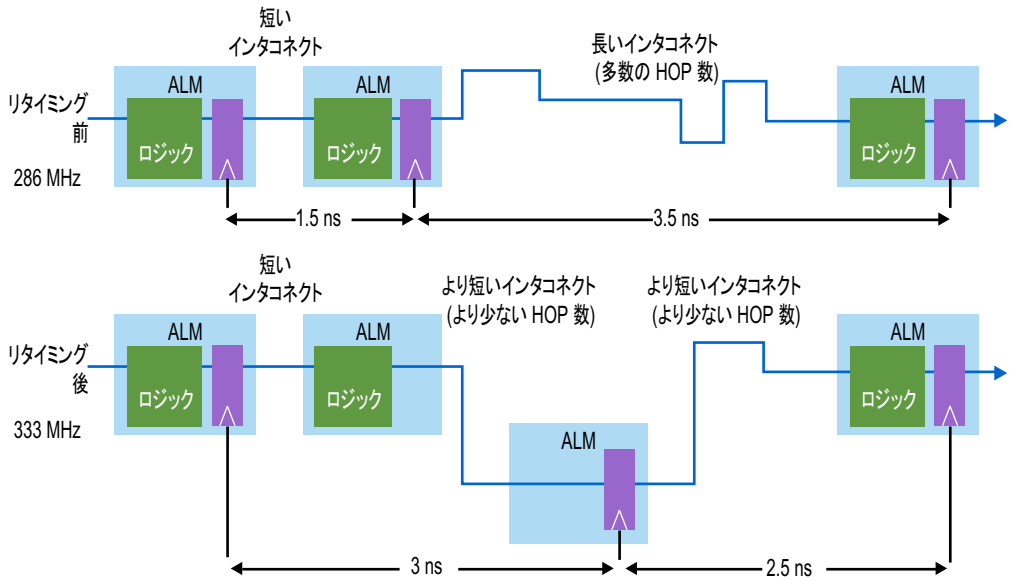
従来のアーキテクチャでは、ソフトウェアは隣接する未使用の ALM レジスタを見つけてこれを回路に組み込むことでリタイミングを実行します。このリタイミング手法は、ALM レジスタを使用するため、以下の制約を受けます。

- 未使用の ALM は最適な場所に存在するとは限らないため、デザイン内の遅延が増加する場合がある。
- ALM を経由してレジスタにルーティングするため、遅延オーバーヘッドが発生する。

- ソフトウェアが広いバス (512 ビット、1,024 ビット、またはこれ以上) のリタイミングを実行しようとした場合、多数のロジック・セルがさらに必要となる。
- リタイミングされるレジスタの最適な位置を決定するために必要なアルゴリズムが困難である。

図 4 に、従来のアーキテクチャを使用したリタイミングの前と後の配線例を示します。

図 4：従来の FPGA アーキテクチャでのリタイミング



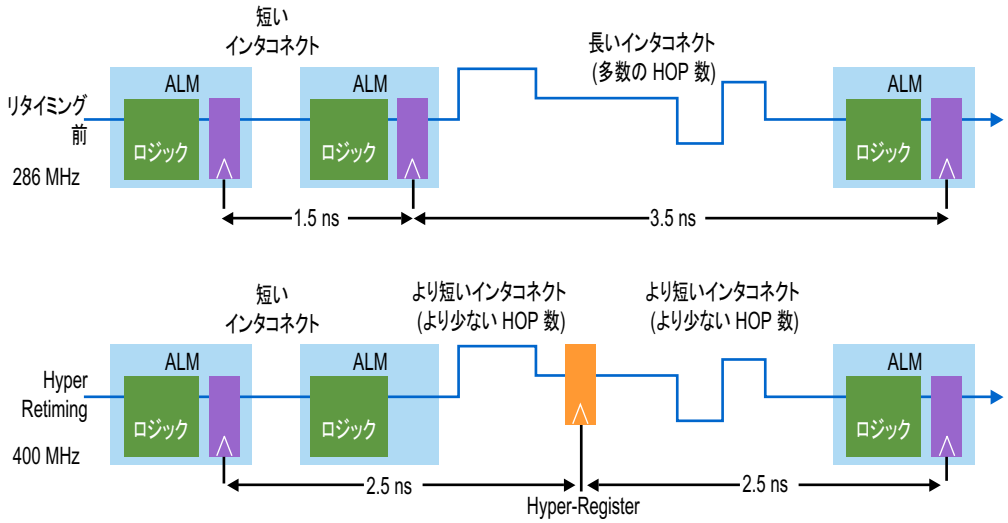
新しい HyperFlex コア・アーキテクチャでは、Hyper-Register を使用することで高精度な Hyper-Retiming を実現します。Quartus II 開発ソフトウェアは、ロジック・セル内のレジスタをインタコネクトに移動することでパスのリタイミングを実行します。すべての配線セグメントに Hyper-Register が存在するため、多数のレジスタの中から最適な位置のレジスタを選択することができ、最適化が容易になります。

Hyper-Register を使用すると、リタイミングの精度が非常に細かくなります。これは個々の配線ワイヤーの遅延となりますが、この遅延は数十ピコ秒です。従来のアーキテクチャでリタイミング・レジスタを配置しようとする場合は、図 4 に示すような妥協が必要になりますが、HyperFlex アーキテクチャではこれが不要です。このため、図 5 に示すように、Hyper-Retiming 中に数ナノ秒のパスを完全に分割できます。

Hyper-Retiming は既存の LAB と ALM に影響を与えないため、追加の配置配線は不要になり、コンパイル時間に重大な影響が及ぶこともありません。レジスタのリタイミングを実行するには、配置配線後にレジスタを配線上のバランスの良い位置に移動するだけです (図 5 を参照)。この機能は、広いデータ・バスを使用するデザインに非常に有効です。従来のアーキテクチャでこうしたデザインを実装する場合、リタイミングを実現するために数百ないし数千の ALM が追加で必要となり、一般的に大がかりな再配線も必要になるからです。

f Quartus® II 開発ソフトウェアを使用して Hyper-Retiming を実行する方法の詳細については、テクニカル・ホワイトペーパー「Quartus II 開発ソフトウェアで実現する HyperFlex アーキテクチャ性能の最大化」をご覧ください。

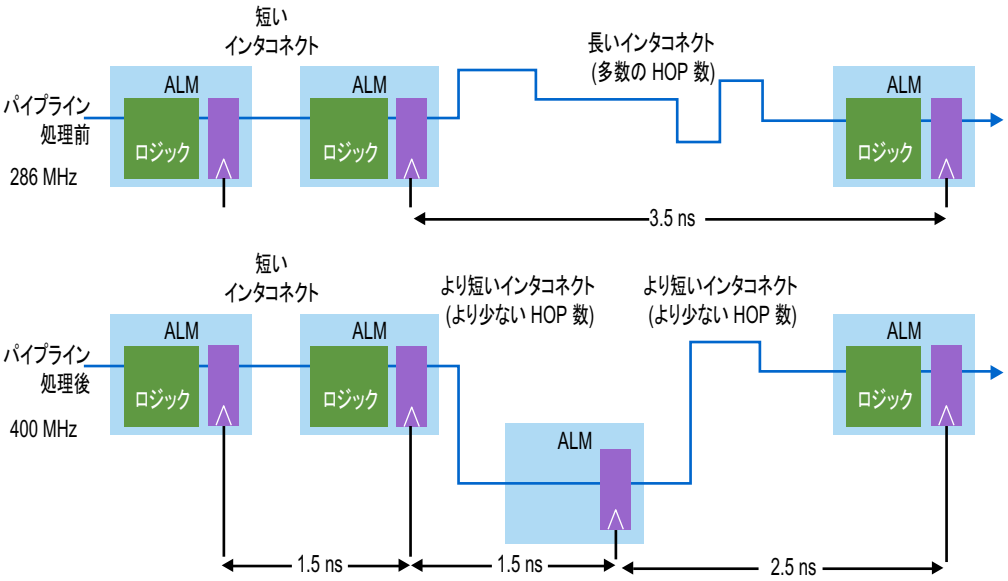
図 5：HyperFlex アーキテクチャの Hyper-Retiming



Hyper-Pipelining

従来のパイプライニングには、従来のリタイミングと同じ欠点があります。また、レジスタの粒度が粗いため、最適化の効果が低減されます。従来のパイプライニングは、本質的に反復的なプロセスです。必要なパイプライン・ステージ数とその最適な位置が開始時に不明であるためです。このため、デザインの配置配線を繰り返しながら、性能目標を満たすパイプライン・ソリューションへと収束させていく必要があります。図 6 に、従来のパイプライニングの前と後のシンプルな例を示します。

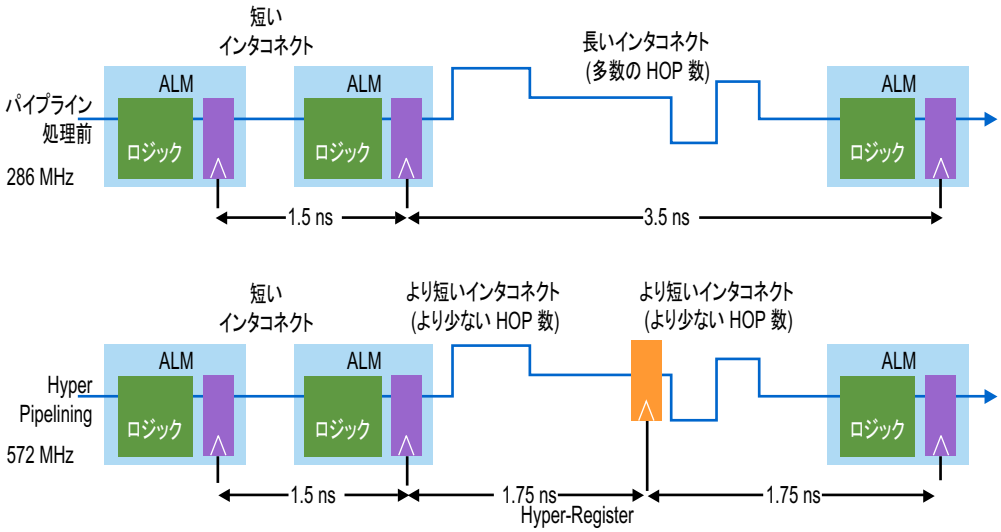
図 6：従来の FPGA アーキテクチャのパイプライニング



HyperFlex アーキテクチャでは、Hyper-Register を使用することでデザインを肥大化させることなく自由にパイプライニングを実行できます。このプロセスは、Hyper-Pipelining と呼ばれています。多くの場合、レジスタを大量に使用するデザインでは、必要な ALM 数が減少します。

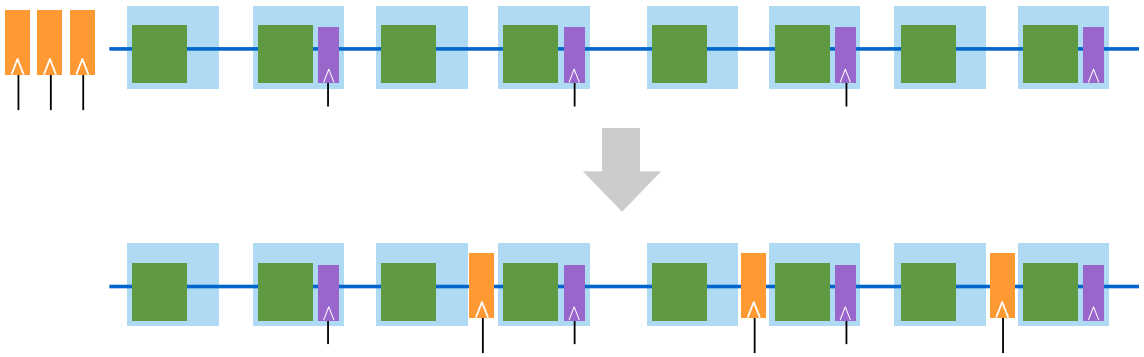
コストを発生させずにパイプライニングを実行するには、特にデータパスとフィードフォワード・ロジックでこの手法を積極的に使用します。図 7 に、Hyper-Pipelining の例を示します。

図 7：HyperFlex アーキテクチャでの Hyper-Pipelining



ソフトウェアはレジスタをインタコネクトに移動することでリタイミングを自動的に実行するため、設計者は、クロック・ドメインへの入力またはサブ・デザインのピン・ロジックで必要なパイプライン・レジスタの数を指定するだけです。Quartus II 開発ソフトウェアは、必要に応じて配置配線後にレジスタを配線に移動します。これにより、従来のアーキテクチャのパイプライニングで発生する設計イタレーションの問題が解決されます。IP (Intellectual Property) ライブラリが複数のクロック周波数 (f_{MAX}) をターゲットにしている場合、RTL 上でレジスタをまとめて配置することで容易にパラメタライズも可能です。図 8 に、クロック・ドメインの入力に追加のパイプライン・レジスタを配置し、これらのレジスタをインタコネクト配線の最適な位置に移動する例を示します。

図 8. クロック・ドメインの入力へのパイプライン・レジスタの配置



パイプラインに適したデザインを実装する設計者は、HyperFlex アーキテクチャの利点を有効に活用できます。例えば、レイテンシの影響を受けないフロー制御形式 (高ファンアウト・クロック・イネーブルの代わりにデータ有効信号を使用するなど) により、ソフトウェアは FPGA コア・ファブリック内でレジスタを簡単に移動できるようになります。

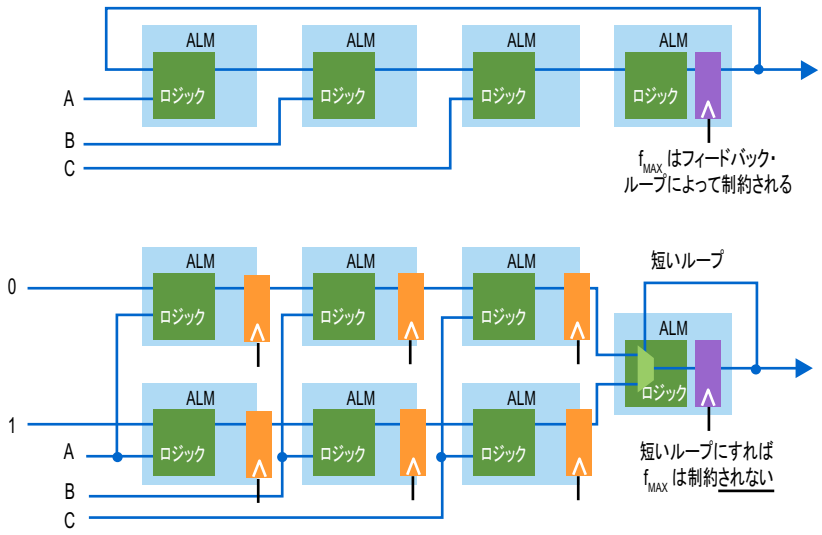
f HyperFlex アーキテクチャを活用したデザイン最適化の詳細については、テクニカル・ホワイトペーパー「HyperFlex アーキテクチャで最適な性能を得るための、RTL デザインの調整」をご覧ください。

Hyper-Optimization

Hyper-Retiming と Hyper-Pipelining が完了すると、デザインの一部のセクションでは著しい性能向上が達成されますが、さらなる向上を妨げるボトルネックとなるセクションも存在します。ボトルネックとなる可能性があるのは、クロック・サイクルごとに判定する必要がある長いフィードバック・ループや複雑なステート・マシンなどの回路です。

デザインの性能を向上させるための一般的な手法は、特定の部分を最適化することです。例えば、デザインに長いフィードバック・ループが存在すると、最大周波数 (f_{MAX}) が制限される可能性があります。最高周波数を上げるには、回路を再設計して使用し得るフィードバック値を事前に計算し、短いフィードバック・ループでそれらフィードバック値を選択します。Hyper-Register を使用すると、事前計算されたパスを Hyper-Retiming と Hyper-Pipelining を使用して最適化できるため、従来のアーキテクチャを使用する場合よりもこのプロセスを高速化できます。図 9 に、Hyper-Optimization の例を示します。この例では、シャノン展開 (またはブール代数の因数分解) によりループを短縮することで、最高周波数が上げられます。一般に、これらの最適化は、(展開するために必要な追加ロジックによる) 面積コストより性能のメリットが非常に重視される制御ループに対して実行します。

図 9：長いフィードバック・ループの Hyper-Optimization



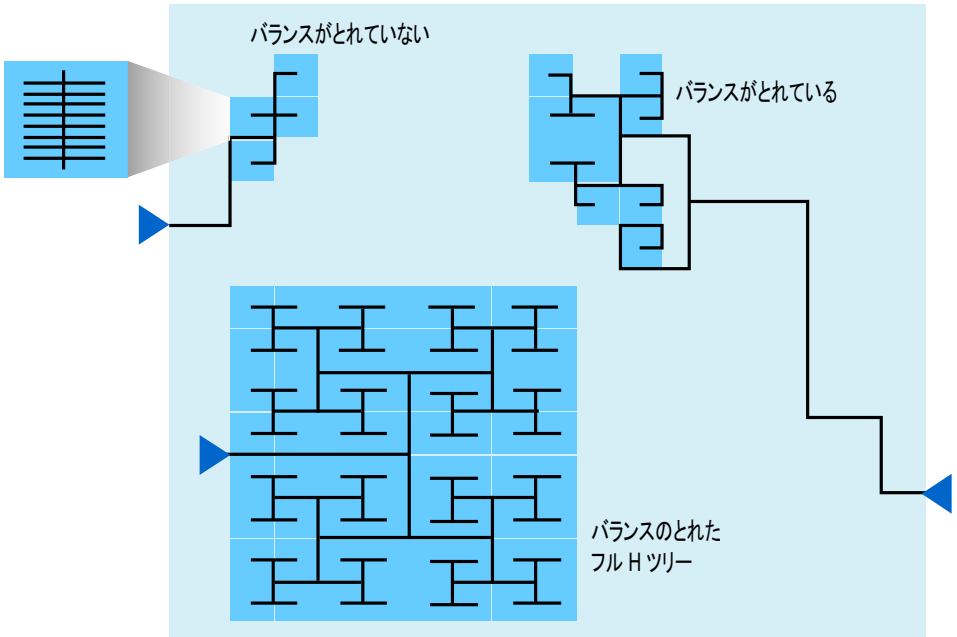
柔軟かつ高速なプログラマブル・クロック・ツリー・シンセシス

高性能 FPGA デザインのクロッキングは、設計者にとってますます困難な課題になっています。従来の FPGA は、高ファンアウト、チップ全体に渡るグローバル・クロック・ドメインをサポートするように設計されている固定グローバル・クロック・ツリー・ネットワークを実装しています。しかし、GHz レベルの性能では、より柔軟なクロック・ネットワークが必要です。設計者は、性能バランスとクロック・クロッシングのためにタイム・シフト・クロックを生成し、レート・マッチングとシステム電源管理のためにゲーテッド・クロックを動的に生成したいと考えます。

これらのニーズに対応するために、HyperFlex アーキテクチャには、まったく新しいクロック構造が組み込まれています。このクロック構造には事前配線済みのクロック・パスが実装されており、デザインのクロック領域はそのパスに生成されます (ASIC クロック・ツリー・シンセシスの場合と同様)。この構造により、ローカライズされた小規模なクロック・ドメインをこれまでにないほど柔軟に作成できます。また、ソフトウェアでスキューを管理できます。つまり、メリットがあるときはスキューを活用し、必要なときはスキューを最小化できます。さらに、必要な場合は、下位互換性のためにこのクロック構造を使用して、従来のバランスのとれたグローバルおよびリージョナル H ツリー・クロックを生成できます。

Quartus II 開発ソフトウェアは、プログラマブル・クロック・ツリー・シンセシスを管理し、配置配線中にクロック・ツリーを生成します。図 10 に、このアプローチで生成されるバランスのとれたクロック・ツリーとバランスのとれていないクロック・ツリーの例を示します。

図 10 : バランスのとれたクロック・ツリーとバランスのとれていないクロック・ツリーの合成



電力効率

Stratix 10 FPGA & SoC は、インテルの 14nm トライゲート (FinFET) プロセス・テクノロジをデバイスに採用して、旧ファミリに比べて電力効率を飛躍的に向上させています。さらに、HyperFlex アーキテクチャは、大幅な節電効果を実現します。HyperFlex アーキテクチャの性能向上により、350 MHz の 1,024 ビット・データパスを 700 MHz の 512 ビット・データパスとして実装できます。これにより、半分のサイズのデバイスにデザインを組み込むことができるようになります。この変更により、ダイナミック消費電力は変化しませんが、スタティック消費電力が半分に削減され、より小型のデバイスを使用することでコストも大幅に低減されます。また、設計者は性能向上の一部をクロック速度の向上に利用し、残りの性能向上をコア電源電圧の低下またはより低速グレードのデバイスの使用によって節電に利用することもできます。

生産性

HyperFlex アーキテクチャによってもたらされるコア性能向上のメリットは、コアの動作クロック・レートの高速化だけに留まりません。コア性能の向上により、タイミング・クロージャが容易になり、デザイン・チームの生産性が向上し、製品の市場投入までの期間が短縮します。

結論

従来の FPGA コア・アーキテクチャでは、次世代の高性能デザインのニーズを満たすことは困難です。リタイミング、パイプライニング、最適化などの手法の価値は、アーキテクチャそのものによって限定されます。Stratix 10 HyperFlex アーキテクチャは、「レジスタ・エブリウェア」アプローチによりこれらの最適化手法を新たなレベルへと引き上げ、前世代の高性能 FPGA の 2 倍のコア性能を実現します。

詳細情報について

- ホワイトペーパー：「次世代システム要件に対応する新しい FPGA アーキテクチャおよび最先端の FinFET プロセス技術」
  https://www.altera.co.jp/content/dam/altera-www/global/ja_JP/pdfs/literature/wp/wp-01220-hyperflex-architecture-fpga-socs_j.pdf
- テクニカル・ホワイトペーパー：「Quartus II 開発ソフトウェアで実現する HyperFlex アーキテクチャ性能の最大化」
  https://www.altera.co./jp/content/dam/altera-www/global/ja_JP/pdfs/literature/wp/wp-01218-quartus-ii-software-to-maximize-performance-in-hyperflex-architecture_j.pdf

文書改訂履歴

表 2 に、本資料の改訂履歴を示します。

表 2. 文書改訂履歴

日付	版	変更内容
2015 年 6 月	1.0	初版