89 号 2014

Xcell journal

SOLUTIONS FOR A PROGRAMMABLE WORLD

ザイリンクスの UltraScale アーキテクチャで生産性を飛躍的に向上

FPGA を利用した Massive MIMO チャネルの特性評価

ザイリンクス FPGA を利用した効率的な並列リアルタイム アップサンプリング

Vivado IP インテグレーターと ザイリンクス IP を利用した 迅速なデザイン エントリ

PicoBlaze マイクロコントローラーを 最大限に活用

Zynq SoC を利用して 4K テレビの開発を容易化

ページ 22

XILINX
ALL PROGRAMMABLE™

japan.xilinx.com/xcell

Xcell journal

発行人 Mike Santarini
mike.santarini@xilinx.com
+1-408-626-5981

編集 Jacqueline Damian

アートディレクター Scott Blair

デザイン/制作 Teie, Gelwicks & Associates

日本語版統括 神保 直弘
naohiro.jinbo@xilinx.com

制作進行 周藤 智子
tomoko.suto@xilinx.com

日本語版 制作・広告 有限会社エイ・シー・シー

Xcell PUBLICATIONS

japan.xilinx.com/xcell/

Xcell Journal 日本語版 89 号

2014 年 12 月 24 日発行

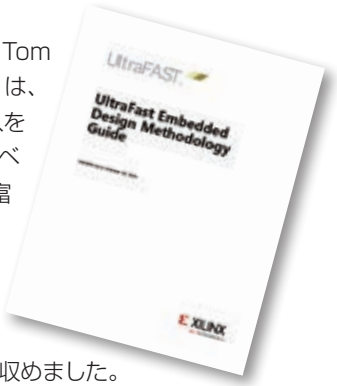
Xilinx, Inc
2100 Logic Drive
San Jose, CA 95124-3400

ザイリンクス株式会社
〒 141-0032
東京都品川区大崎 1-2-2
アートヴィレッジ大崎セントラルタワー 4F

設計チームの Zynq SoC デザインを加速する UltraFast エンベデッド デザイン設計手法マニュアル

Zynq®-7000 All Programmable SoC を次の設計プロジェクトに使用する設計者の皆様に、大事なお知らせがあります。このたびザイリンクスは、設計者必読の設計手法マニュアルである『UltraFast ™ エンベデッド デザイン設計手法ガイド』(UG1046) を発行しました。この資料は無償でダウンロードできます。

このマニュアルは 211 ページで、ザイリンクスが 2013 年に発行した『UltraFast 設計手法 (Vivado Design Suite 用)』(UG949) に対する比較ドキュメントになっており、これと併せてお読みください。『UltraFast 設計手法 (Vivado Design Suite 用)』については、Xcell Journal 日本語版 85 号のカバー ストーリーで紹介しています。



ザイリンクスの設計手法マーケティング担当シニア ディレクターである Tom Feist によると、『UltraFast 設計手法 (Vivado Design Suite 用)』は、すべての Vivado ツール ユーザーが Vivado ツールのベスト プラクティスを利用することに焦点を合わせているのに対して、新しい『UltraFast エンベデッド デザイン設計手法ガイド』は、設計チーム全体でザイリンクスの豊富なエンベデッド デザイン ツールを最大限に活用することを目的としています。このマニュアルのガイドに従えば、設計チームはデザイン プラクティスを最適化し、Zynq SoC ベースのデザインの time-to-market を短縮できます。

「『UltraFast 設計手法 (Vivado Design Suite 用)』は大きな成功を収めました。このマニュアルにより、ザイリンクス製品のユーザーは開発期間を数カ月から数週間に短縮できることが証明されました。また、多くの設計チームが Vivado ツールの ASIC 設計クラスの機能を利用するようになりました」と Feist は述べています。「『UltraFast エンベデッド デザイン設計手法ガイド』により、ザイリンクスの Zynq SoC を使用するエンベデッド デザイン チームの生産性はさらに向上するはずです」。

エンベデッド デザイン チームは、通常はシステム設計者、ハードウェア技術者、ソフトウェア技術者などのさまざまなスキルとメンバーを組み合わせて構成されます。「技術者にはタイプごとに特定の作業の進め方があり、特定のソフトウェアやツールを使用して設計を進めます」と Feist は述べています。「もちろん、スキルのレベルもさまざまです。この設計手法マニュアルは、あらゆる技術者に有益な情報を提供します」。

Feist によると、このマニュアルは、設計チームとしての Zynq SoC 設計プロジェクトの構成の改善と、個々の Zynq SoC 設計スキルの向上に役立ちます。「このマニュアルは、エンベデッド デザイン チームの各メンバーに、設計プロセスの最適化に役立つ主要な原則、具体的な推奨事項と禁止事項、およびベスト プラクティスを示しています」。

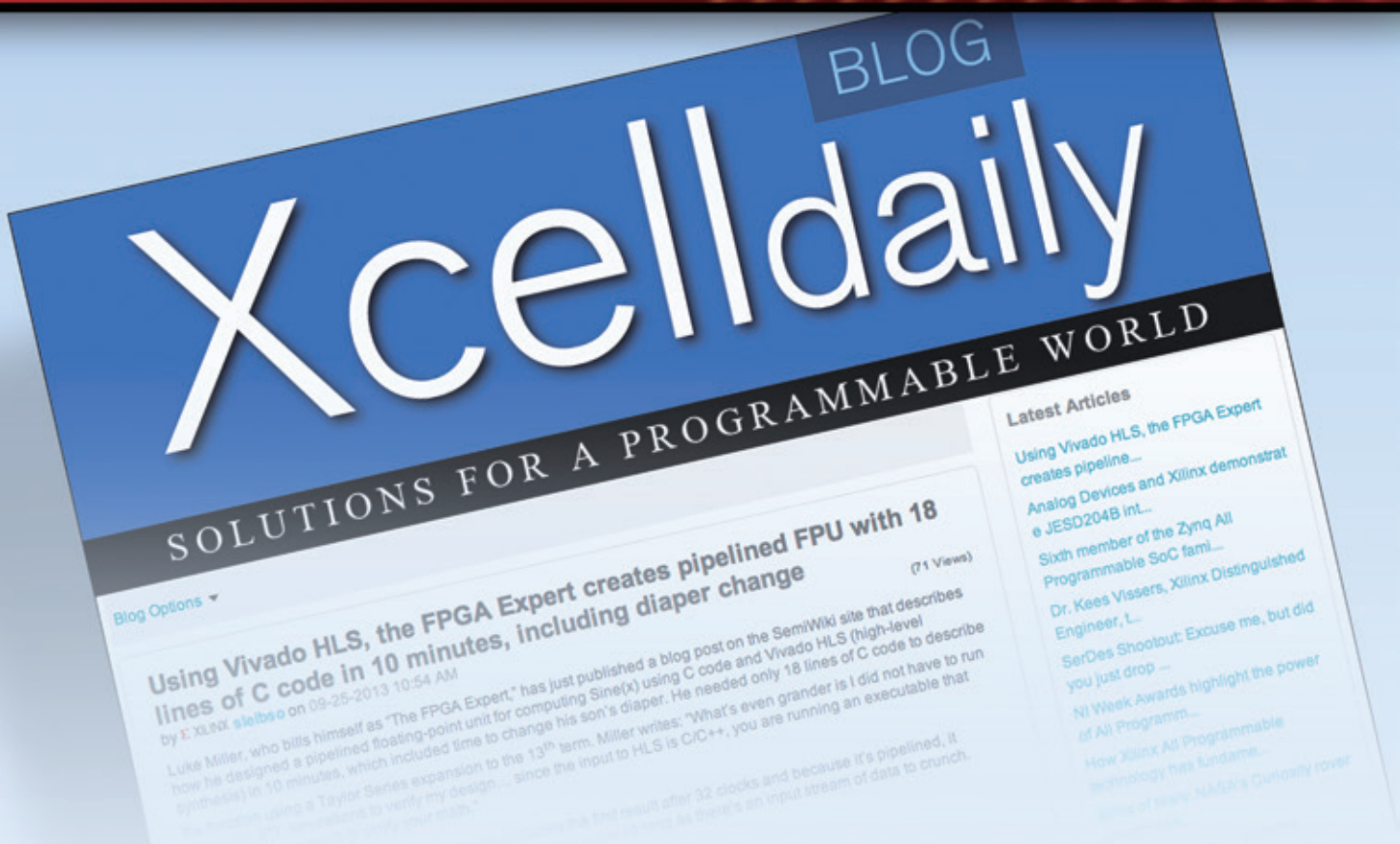
マニュアルの内容は、既に多くの Zynq SoC ベースのプロジェクトを市場に提供してきたザイリンクスの専門家および設計チームのベスト プラクティスに基づいています。このマニュアルは技術分野ごとにセクションが分かれていますが、Feist によると、設計チームのすべてのメンバーがマニュアル全体を読み通すことにより、価値ある情報を習得し、プロジェクトの構成を改善できます。

このマニュアルは、システムレベルの考慮事項、ハードウェア デザインの考慮事項、ソフトウェア デザインの考慮事項、ハードウェア デザイン フロー、ソフトウェア デザイン フロー、およびデバッグの全 6 章で構成されます。さらに、このマニュアルは各フローの切り替えと相互関係についても説明しています。

『UltraFast エンベデッド デザイン設計手法ガイド』をぜひご一読ください。情報の宝庫です。また、今回の Xcell Journal にも興味深い記事がたくさん掲載されていることと思いますのでお楽しみいただけたら幸いです。

Mike Santarini
発行人

Xcell Journal を拡充。新たに Daily Blog を追加



ザイリンクスは、数々の受賞歴がある Xcell Journal をさらに拡充し、エキサイティングな Xcell Daily Blog（英文）を始めました。このブログでは、コンテンツを頻繁に更新し、技術者の皆様がザイリンクスの製品とエコシステムの多岐にわたる機能が活用でき、All Programmable システムおよび Smarter System の開発に役立つ情報を提供します。

Recent（最近の記事）

- *Lighthouse Imaging uses PLDA's Zynq-based SoMZ-7045 to bring commercial products to market faster*
- *12 more companies join NBASE-T alliance for 2.5 and 5Gbps Ethernet standards*
- *Adam Taylor's MicroZed Chronicles Part 62: Answers to a question on the Zynq XADC*
- *Dave Jones does a video review of the Digilent Analog Discovery, which is based on a Spartan-6 FPGA*
- *Next-gen wireless basestation design in your future? New Xilinx White Paper offers some design insights*

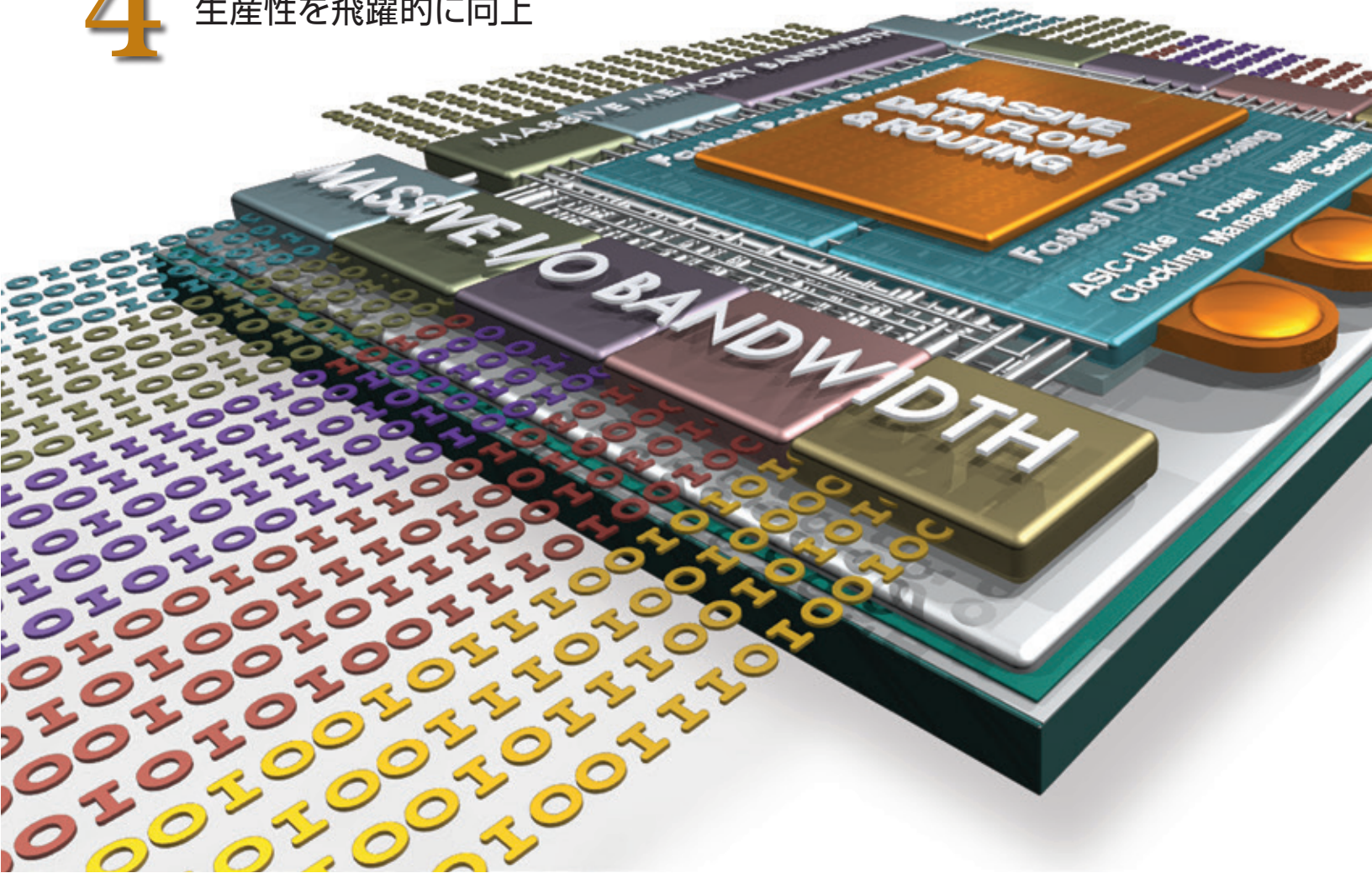
ブログ : www.forums.xilinx.com/t5/Xcell-Daily/bg-p/Xcell

XILINX
ALL PROGRAMMABLE™

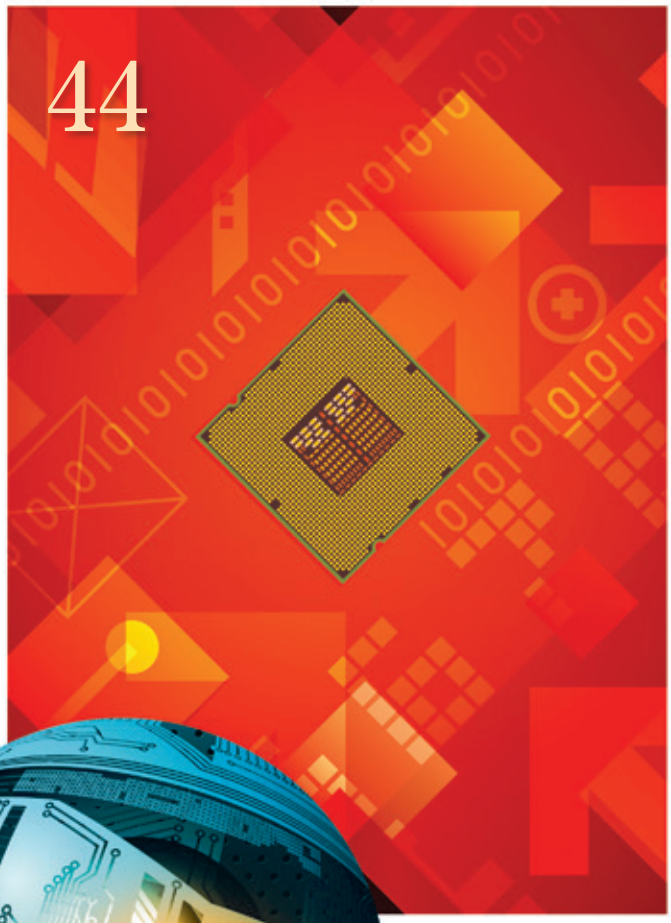
VIEWPOINTS

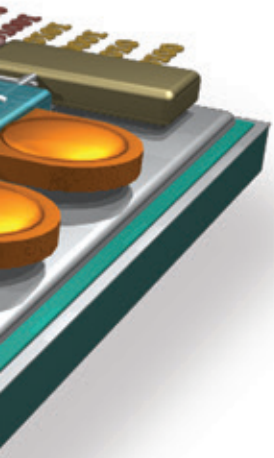
XCELLENCE BY DESIGN APPLICATION FEATURES

Cover Story



THE XILINX XPERIENCE FEATURES





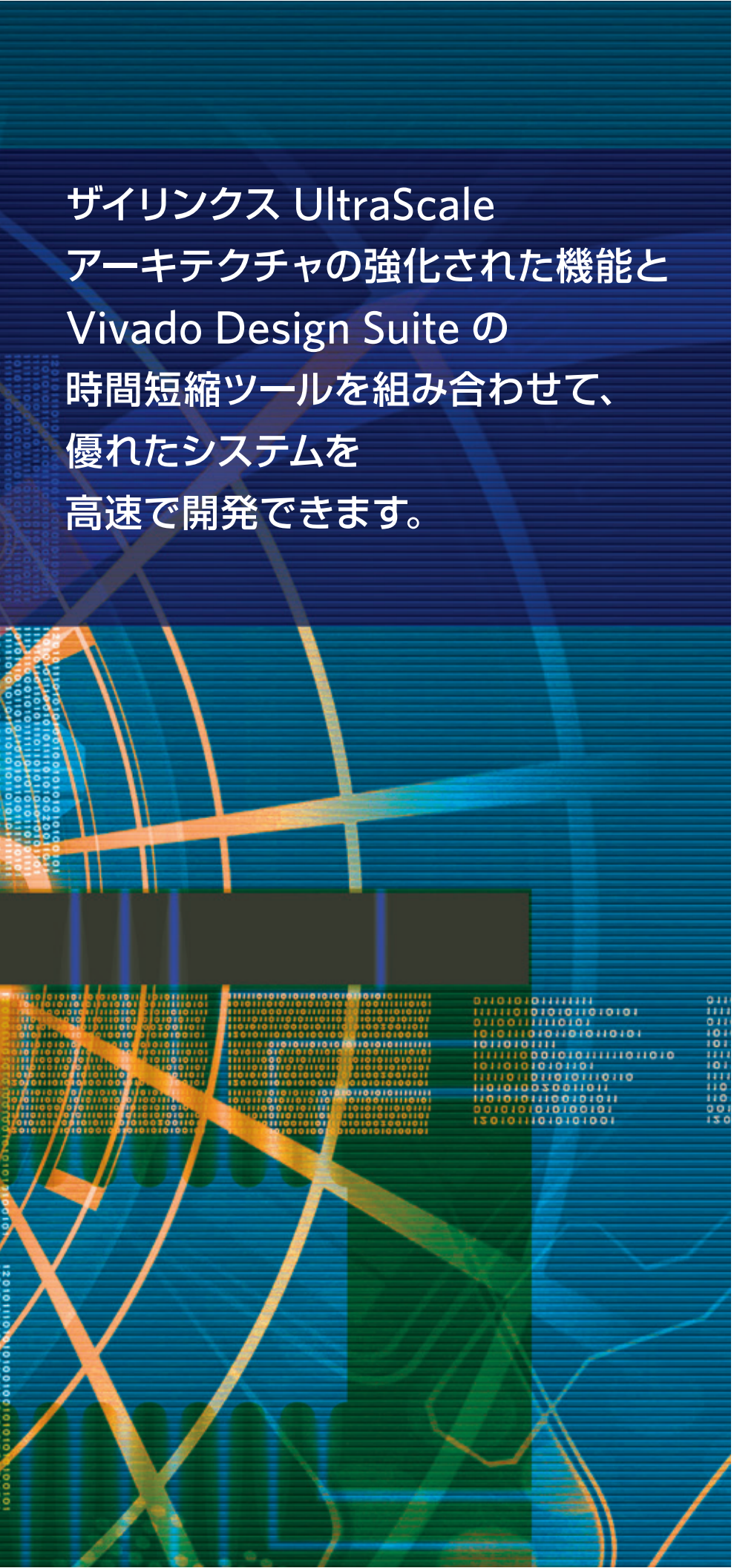
APEX®
AWARDS FOR PUBLICATION EXCELLENCE
Excellence in Magazine & Journal Writing
2010, 2011

APEX®
AWARDS FOR PUBLICATION EXCELLENCE
Excellence in Magazine & Journal Design
2010, 2011, 2012

Productivity Skyrockets with Xilinx's UltraScale Architecture

ザイリンクスの UltraScale アーキテクチャで生産性を飛躍的に向上

Nick Mehta
Senior Technical Marketing Manager
Xilinx, Inc.
nick.mehta@xilinx.com



ザイリンクス UltraScale アーキテクチャの強化された機能と Vivado Design Suite の時間短縮ツールを組み合わせて、優れたシステムを高速で開発できます。

多くの市場およびアプリケーションで、システム帯域幅と処理能力の大幅な向上が求められています。有線通信、無線通信、デジタル ビデオ、画像処理などのさまざまな用途でデータ スループットの要件が高まると、その結果としてトラフィックが増え、すべてのシステム コンポーネントへの要求が増大します。パラレル I/O とシリアル I/O を介してチップに到達するデータの量が増え、そのデータはパラレル I/O を介して DDR メモリの形式で、またシリアル I/O を介してハイブリッド メモリ キューブ (HMC) や MoSys 社の Bandwidth Engine などのシリアル メモリ標準の形式で、バッファー処理しなければなりません。データはロジックと DSP で処理された後、再びパラレル I/O とシリアル I/O を介して次の宛先に送られます。

システム処理の要件も、次のような理由で複雑化しています。データ パケットの大容量化とデータ レートの高速化の結果、パラレル データ バスのバス幅拡大化と動作周波数の向上が進みます。このデータを効率的に処理するために、システム全体を 1 つのデバイスで構築し、2 つの FPGA 間の大量のデータ送信に伴うレイテンシと消費電力を削減することがしばしば必要になります。その結果、より多くの機能を備えた、より高密度な FPGA へのニーズが生まれます。これらの高機能 FPGA の使用率が高くなるに従い、デバイスの容量が一杯になった状態でも性能の低下を回避し、最大限の性能で動作を維持することが必須となります。

複雑な大容量のデバイスを高い使用率に維持することは、設計者には手間のかかる作業に思われるかもしれません。ザイリンクスは、設計者が市場における製品の差別化に集中できるように、設計時間の短縮を目的とする多くのソリューションを提供しています。

UltraScale アーキテクチャ

昨今の市場における課題に対処するため、先頃ザイリンクスは、これまでにないレベルのシステム統合、性能、機能を実現した UltraScale™ アーキテクチャ (図 1) を発表しました。ザイリンクスは、この新しいアーキテクチャを使用して、ザイリンクス Virtex® UltraScale ファミリおよび Kintex® UltraScale ファミリという 2 つの高性能 FPGA 製品ファミリを開発しました。2 つの製品ファミリは、数多くの革新的な先進技術によって特に総消費電力の削減を実現し、幅広いシステム要件に対応します。多くのビルディング ブロックを共有する UltraScale テクノロジは、さまざまな市場の要求に合わせて最適化される、スケーラブルなアーキテクチャを提供します。

システム帯域幅の向上

信号処理やデータ操作を実行するには、データが宛先に到着する必要があります。現在、ターゲット アプリケーションの特定要件に応じて、多数のシリアルおよびパラレル プロトコルと標準が存在します。ほとんどの標準に共通するテーマは、総データ スループットの向上であり、システムを通る大量の情報の高速データ レートでの移動を実現することです。

データは、UltraScale アーキテクチャをベースにした FPGA との間で、高性能パラレル SelectIO™ 接続と高速シリアル トランシーバー接続を組み合わせて転送されます。I/O ブロックは、I/O 標準と電圧の柔軟なサポートによって、最先端のメモリ インターフェイスとネットワーク プロトコルを可能にします。UltraScale アーキテクチャの 2 種類のシリアル トランシーバーのデータ転送速度は、GTH で最大 16.3Gbps および GTY で最大 32.75Gbps です。GTH トランシーバーは一般的なシリアル プロトコルに必要なすべての性能を提供し、GTY トランシーバーは前世代のトランシーバーに比べてビット当たり消費電力を大幅に削減し、25G+ バックプレーン デザインを実現します。UltraScale FPGA のすべてのトランシーバーは、PCI Express® Gen3 および Gen4 に必要なデータ レートをサポートします。また PCI Express 用統合ブロックにより、UltraScale アーキテクチャをベースとする FPGA は、最高 x8 までの Gen3 エンドポイントおよびルート ポート デザインをサポートします。

データのクロッキングとバッファリング

すべてのシンクロナス システムは、回路の同期に 1 つ以上のクロック信号を使用します。システム性能の向上には、大容量デバイス機能でのクロック周波数の向上を必要とするため、クロックの柔軟性の向上と総クロック電力の削減が求められます。

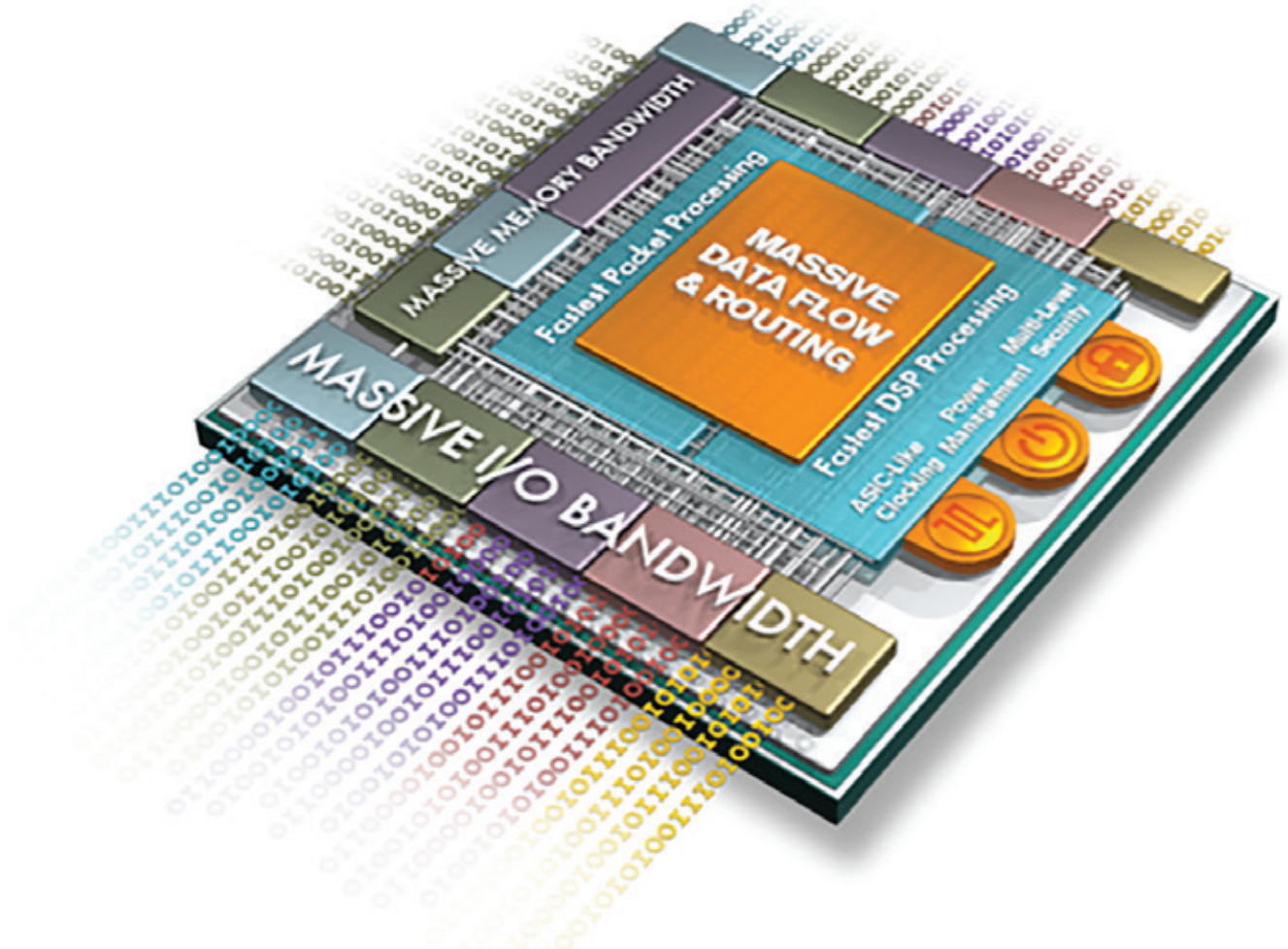


図 1 - UltraScale アーキテクチャの主な利点は速度と帯域幅です。

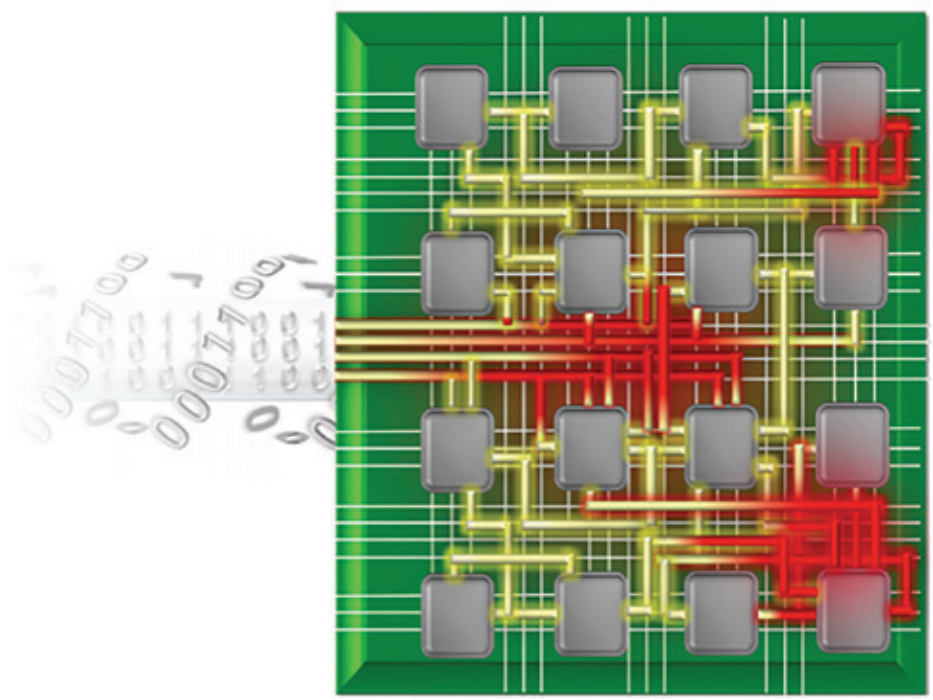


図 2 - UltraScale アーキテクチャは大量のデータを処理できます。

UltraScale アーキテクチャは、再設計された強力なクロック管理回路を搭載しています。これにはクロック合成、バッファリングおよび配線用のコンポーネントが含まれ、各コンポーネントが連携して、設計要件に対応する高性能フレームワークを提供します。このクロック ネットワークは、FPGA 内で極めて柔軟なクロックの分配を可能にし、クロック信号に関連するスキュー、消費電力、遅延を最小限に抑えます。また、クロック管理テクノロジは専用のメモリ インターフェイス回路と緊密に統合され、DDR4 などの高性能外部メモリのサポートを可能にします。クロック セグメンテーションと新しい細粒度化クロック ゲーティングにより、既存の FPGA と比べてクロック消費電力の制御が強化されます。

UltraScale アーキテクチャでは、グローバル クロック バッファーの数が従来の FPGA や競合 FPGA と比べて大幅に増え、設計者の生産性に大きなメリットをもたらします。従来のアーキテクチャでは、グローバル クロック バッファーは FPGA の中心に配置され、32 個しかなかったため、バッファーの使用が制限されていました。UltraScale アーキテクチャでは、アーキテクチャ全体にグローバル クロック バッファーを自由に配置でき、必要な場所にリソースを提供できるため、バッファーの使用を制限する必要は少なくなります。また、ザイリンクスは、旧世代の FPGA と比べてクロック バッファーの種類を大幅に簡略化しながらも、クロック切り替え、クロック分割、クロック イネーブル機能をすべて維持しました。その結果、柔軟で高性能なクロック バッファーが大量に確保され、必要な場所に必要な機能を提供できます。

データの格納、処理、配線

あらゆるシステムの鍵となる要素は、受信したデータを処理、操作、変換する機能です (図 2)。システムの複雑性が増すにつれて、汎用ファブリックと、特定のデータ処理専用の専門的な機能を組み合わせることが必要になります。

現在の FPGA のファブリックは、6 入力ルックアップ テーブル (LUT) およびフリップフロップを含むコンフィギャラブル ロジック ブロック (CLB)、27x18 乗算器を備えた DSP スライス、内蔵 FIFO および ECC をサポートする 36 キロビット ブロック RAM などの多くの要素で構成されます。これらのリソースはすべて、数多くの高性能、低レイテンシのインターコネクトによって互いに接続されます。

CLB は、論理機能以外に、シフト レジスタ、マルチプレクサー、キャリー ロジック機能や、LUT を分散型メモリとして構成して高性能なコンフィギャラブル ブロック RAM を補完する機能を提供します。DSP スライスは、新しい 96 ビット幅の XOR 機能、幅が拡張された 27 ビット前置加算器および 30 ビット入力により、乗算累算、乗算加算、パターン検出など、多くの独立した機能を実行します。またデバイス インターコネクト以外に、第 2 世代 SSI 3DIC テクノロジを採用したデバイス内では、信号が低レイテンシの専用インターフェイス タイルを使用して SLR (Super Logic Region) 境界をまたぐことができます。これらの配線リソースの組み合わせにより、次世代のデータ バス幅を容易にサポートし、90% を超えるデバイス使用率を実現できます。

デザインの課題を簡単に解決

UltraScale アーキテクチャで実現されたアーキテクチャの強化により、設計者が同一面積にパックできるデザインは増えますが、それに加えて、デバイスのサイズが大きくなります。1 つのデバイスに収められるデザインが増えることは大きな利点ですが、設計チームには、最終製品の time-to-market をできる限り短縮するために、指定されたデザインを迅速にインプリメントすることを求める圧力が課せられます。ザイリンクスは、UltraScale アーキテクチャおよび協調最適化された Vivado® Design Suite により、設計時間の短縮と生産性の向上を可能にするソリューションを提供します。

コアの機能の統合

柔軟なプログラマビリティは有益な資産ですが、その他の事例と同様に、それには代償が伴います。プログラマブル リソースから作成された機能は、同じ機能を持つ専用ブロックよりもサイズが大きく、低速になることがあります。もちろん、FPGA は基本的にはプログラマブルなデバイスです。しかし、ザイリンクスの FPGA には、専用の機能を持つ統合された IP コアがバランスよく含まれているため、ユーザーはよく利用される機能を迅速にインプリメントできます (図 3)。UltraScale アーキテクチャには、各種の一般的な通信プロトコル用の

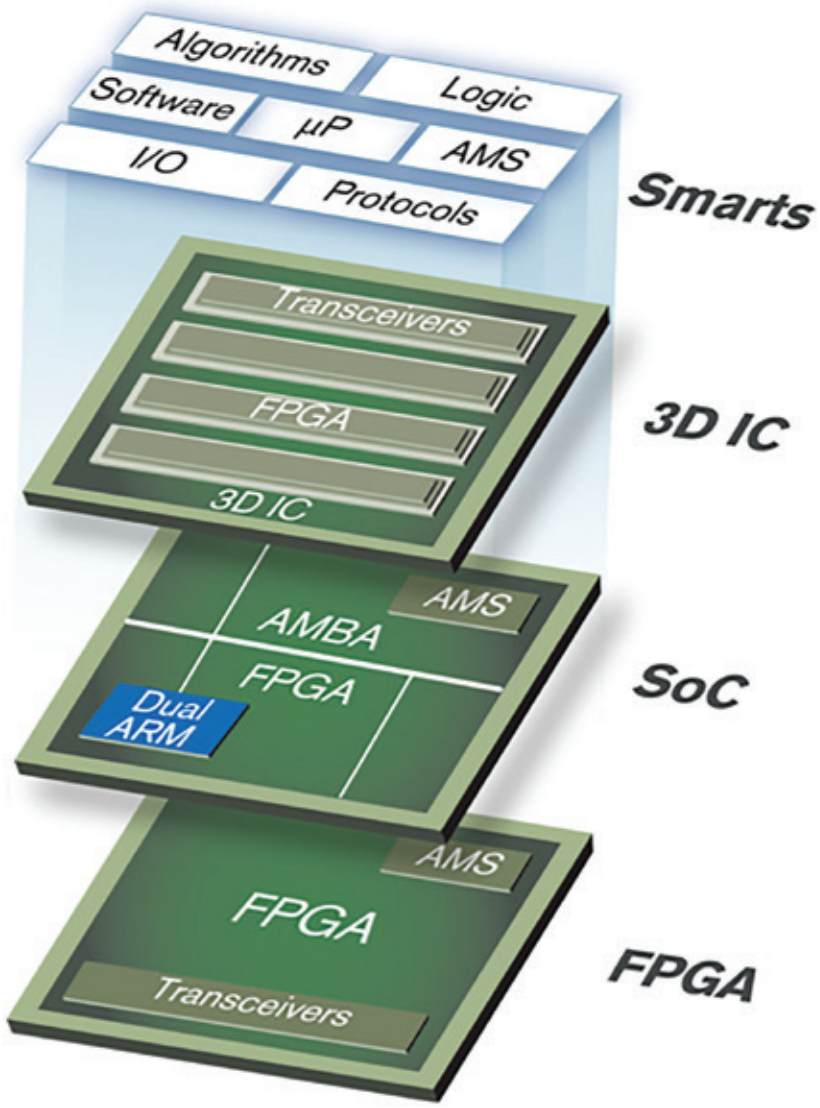


図 3 - UltraScale デバイスは、業界最先端のテクノロジに、よく利用される機能を追加します。

統合ブロックが含まれています。Kintex UltraScale および Virtex UltraScale デバイスには、PCI Express 用、100G イーサネット用、および 150G Interlaken 用の複数の統合ブロックが組み込まれ、すべてのブロックは完全にテストおよび検証済みであり、正常な機能が保証されています。

通信プロトコル以外に、各 I/O バンクには、メモリ インターフェイス ジェネレーター (MIG) ツールによって構成可能なプログラマブル メモリ PHY が含まれています。これは必要に応じたコアの統合の優れた例です。メモリ PHY と一部の制御ロジックはプログラマブルな専用機能として作成されますが、メモリ インターフェイスのデジタル部分はデバイス ファブリックから構築され、必要なすべてのカスタマイゼーションと (専用回路では提供するのが難しそうな) 各種モードへのサポートを提供します。

デバイス ファブリック内には、このほかにも、プログラマビリティの要素を維持しながら特定の機能を実行するように設計された多数のブロックがあります。設計者は、さまざまな深さと幅でブロック メモリを構成し、それらをカスケード接続して、低消費電力の大容量アレイを作成できます。DSP スライスには多くのモードがあり、ユーザーは選択された機能に応じて DSP ブロックの各種のコンポーネントにアクセスできます。このように、UltraScale アーキテクチャ全体に、単なるゲートとレジスタを超えた豊富な機能が備わっています。

カスタマイズ可能で反復利用可能な IP コアで生産性を向上

すべてのデザインは、相互に接続されてシステムを構築する、各種のアーキテクチャ ビルディング ブロックで構成されます。専用の固定機能ブロックとして提供する方が経済的に合理的であるとして常時使用されている機能は、この業界にはわずかしかありません。むしろ最適な手法は、プログラマブル ロジックから作成されるものとして機能を設計し、その機能を検証して、必要に応じて再利用することです。この形式の IP コアの概念は既に数世代にわたって提供されていますが、先頃ザイリンクスは、生産性を強化するいくつかのツールを発表しました (図 4)。

プラグ アンド プレイ IP

2012 年に、ザイリンクスは、プラグ アンド プレイ IP 用の標準インターフェイスとして ARM® AMBA® AXI4 インターフェイスを採用しました。単一の標準インターフェイスを使用すると、従来よりも IP コアの統合がはるかに容易になり、各種のインターフェイスを 1 つに統合できるため、設計者が多くの異なるインターフェイスを理解する必要がなくなります。UltraScale アーキテクチャは AXI4 インターコネクトの柔軟性と拡張性のメリットを従来どおり利用できるので、設計者は早い time-to-market を実現すると同時に、AXI4-Lite や AXI4-Stream など各種の AXI4 インターコネクト プロトコルを使用して、IP コアの性能、面積、消費電力を最適化できます。

Vivado IP Packager と Vivado IP Catalog は、IP-XACT 標準を利用します。IP-XACT は、SPIRIT コンソーシアムがツール フロー内での IP コアのパッケージング、統合、再利用のために標準構造として策定した規格で、現在は承認済みの IEEE 標準 (IEEE1685-2009) となっています。

Vivado IP Packager は、ローカル ドライブまたは共有ドライブ上の拡張可能な IP

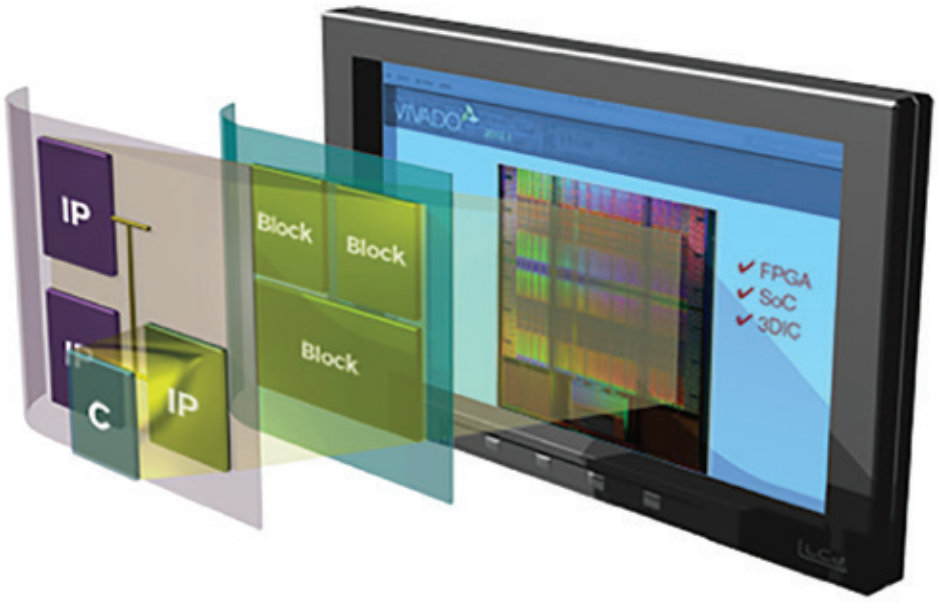


図 4 - Vivado ツールは複雑なデザインの作成とインプリメンテーションを加速します。

カタログ内で利用できる、デバイスの制約、テストベンチ、ドキュメントを使用してデザインを作成します。Vivado IP Catalog は、ユーザー独自の IP コアとザイリンクスおよびサードパーティ製の IP コアを組み合わせて、設計チーム内ですべての IP コアを一貫性のある使いやすい形で共有できるようにします。

Vivado IP Integrator

Vivado IP Integrator (IPI) (図 5) は、システム統合の時間を短縮し、構成要素からシステムをすばやく簡単に組み立てられる、IP 中心型のデザイン フローです。IPI は、インタラクティブなグラフィカル ユーザー インターフェイスにより、IP インターフェイスのインテリジェントな自動接続、ワンクリックによる IP サブシステムの生成、強力なデバッグ機能を提供し、設計者は IP カタログ内のすべての IP コアを素早く簡単に接続できます。この機能により、設計者は、すべてのビルディング ブロックが正しく構成されているという認識のもとで、(無償、有償、自社制作などの) 多くのソースから得られた設計要素から複雑なシステムを迅速に組み立てることができます。これにより、概念設計からデバッグまでの期間がこれまでにないほど短縮されます。

このように、UltraScale アーキテクチャは、数多くの重要な領域に革新的なアーキテクチャを組み込み、次世代の高性能デザインの厳しい要求を満たすことに成功しています。このパズルの重要なピースとなるのは、UltraScale アーキテクチャの使用で得られるように、広帯域データ バスとますます高速化するシステム動作周波数に対応するデザインをインプリメントできる技術です。ただし、デバイスのサイズと複雑性の増大に応じて、設計者の生産性が向上し続けることは非常に重要です。ザイリンクスは、統合ブロックと事前検証済み IP コアを組み合わせて、優れたソリューションを迅速にインプリメントするのに必要なすべてのツールを設計者の皆様に提供します。

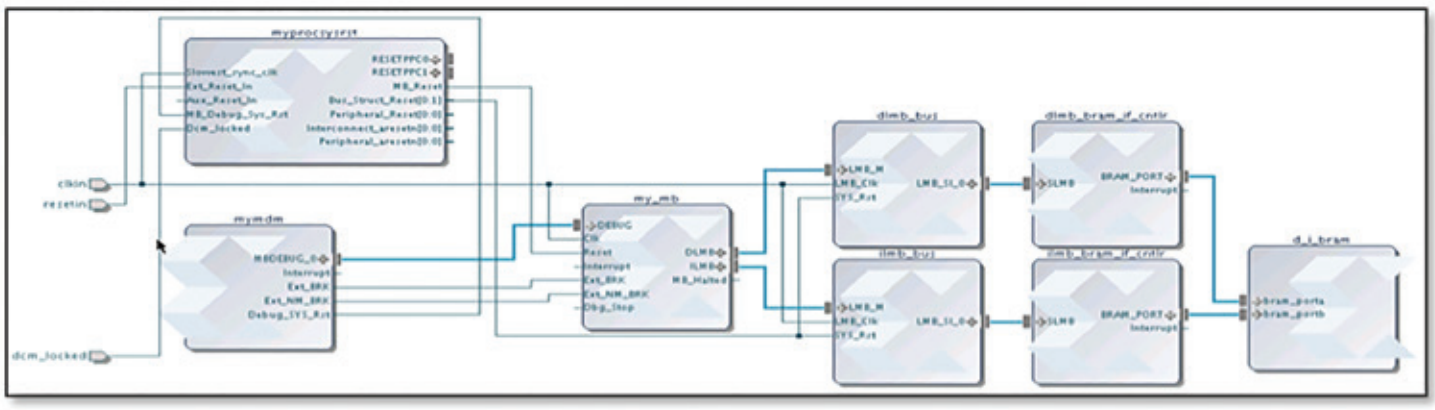


図 5 - IP Integrator 内では、IP ブロックを接続するだけで簡単にデザインを作成できます。

UltraScale Brings Interlaken Onboard for High-Bandwidth Designs

Interlaken IP コアを統合した UltraScale FPGA で高帯域幅のデザインを実現

ザイリンクスは UltraScale FPGA に Interlaken IP コアを統合し、パケット処理システムのインターコネクトを簡略化しました。

Martin Gilpatric
Transceiver Technical Marketing Manager
Xilinx, Inc.
martin.gilpatric@xilinx.com

帯域幅とはおかしなものです。つい 10 年前には、ほとんどの人は「1 秒毎のバイト数 (byte per second)」という用語を理解していませんでした。しかし現在、オンライン ビデオ、スマートフォン、そして相互接続された現代社会の多様な機器によって、帯域幅は一般に広く意識されるようになりました。これらのアプリケーションは、なぜ帯域幅が重要か、なぜ帯域幅の拡張が必要なのかを明るみにしました。

残念なことに、こうした大まかな説明は、より多くのデータを家庭および携帯型の機器に供給するのに必要なテクノロジの進化を覆い隠してしまいます。データ量の増大には、インフラストラクチャが追従する必要があります。従来 10Gbps オプティクスに依存していたインターコネクトは、既に 40 ～ 100Gbps に移行しており、近い将来には 400Gbps を模索する状況です。

現在の業界における INTERLAKEN の位置付け

これらのプロセスを通じて、基本的なパケット処理タスクとアーキテクチャには、大きな変化は起きていません。同様に、100Gbps システムが現在直面しているパッケージング、消費電力、熱に関する制約の多くは、以前の製品が直面していた制約と同じものです。これは、システムのすべてのコンポーネントがはるかに高いレートで動作する必要があることを意味しています。さらに重要なことに、これらのデバイス間のインターコネクトは、ただちに拡張可能である必要があります。この課題を複雑にしているのは、システム内で使用される FIC、NPU、MAC などの ASSP のベンダー数の多さです。これらのデバイスは、目的のスループットを達成するために、さまざまな幅とレートのインターコネクトを使用します。

現在、ザイリンクスと Interlaken プロトコルは、パケット処理機能間でのシリアル インターコネクトのボトルネック解消に取り組んでいます (図 1)。Cisco 社と Cortina 社の共同研究から生まれた Interlaken は、これらの課題に積極的に応えるために開発されたプロトコルです。ザイリンクス FPGA は、高性能プログラマブル ロジックと高性能トランシーバーによって、この標準規格をただちにインプリメントできます。このような状況で Interlaken プロトコルが最適である理由として、高い拡張性、OTN およびイーサネット システムとの適合設計性、プロトコルのオープン性の 3 つが挙げられます。

Interlaken は、最も基礎的なレベルから、最大限の柔軟性と拡張性が得られるように設計されています。Interlaken 仕様は、定義済みのライン レートやレーン幅を義務付けていません。つまり、たとえば 100G イーサネット データ パス (と追加のパケット ヘッダー) をカプセル化するのに十分な 150Gbps インターフェイスをインプリメントする場合、12.5Gbps のレーンを 12 個使用しても、25Gbps のレーンを 6 個使用してもかまいません。いずれの場合も Interlaken への論理インターフェイスは維持されるため、デザインのその他の部分は簡略化されます。さらに、こうした柔軟性により、TCAM メモリをインプリメントする場合のルックアサイド インターフェイスのようなニッチ アプリケーションに使用できるよう、Interlaken を適応させることが可能です。

データが転送されているレーンの数に関係なく、レーン上でデータのセグメンテーションとストリッピングが行われます。この手法には、OTN ストリームまたはイーサネット パケットを上手に取り扱いながら、レイテンシを低減できる利点があります。Interlaken は、このセグメンテーションの範囲内でチャネル化を許容します。これにより、設計者は、Interlaken プロトコルの既存の機能セットの枠内で作業しながら、パケット

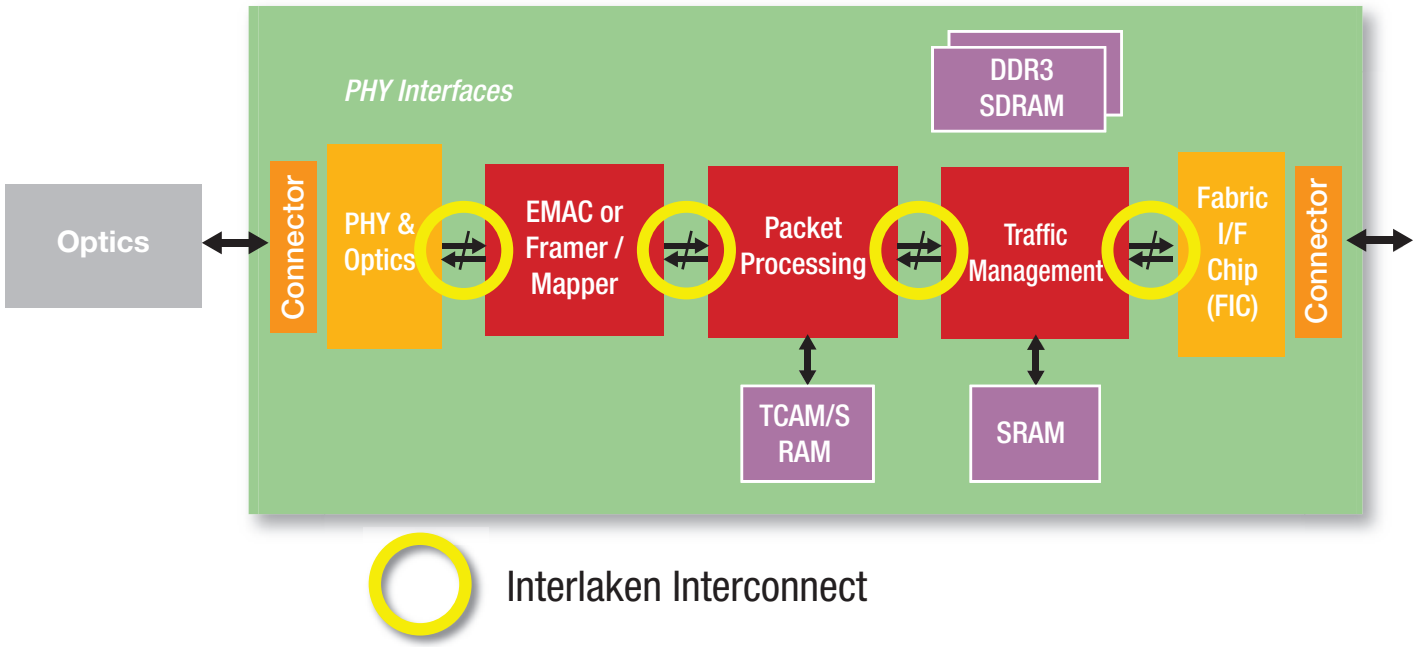


図 1 - パケット処理において Interlaken が使用可能な場所

図 2 - Virtex UltraScale の 32.5Gbps GTY 高性能トランシーバー (このビデオのデモンストレーション) と統合された Interlaken ブロックを使用して、柔軟性を損なわずに、緊密に統合された、高性能、低消費電力のネットワークを作成できます。

の優先順位の指定などの機能を柔軟にインプリメントできます。

Interlaken の豊富な機能セットと無償ライセンスは、このプロトコルの導入を容易にしました。また、Interlaken プロトコルの最初の定義に参加した関係企業の幅広さは、Interlaken の機能セットが幅広いニーズにすぐに対応できることを意味していました。さまざまな ASSP と IP コアがすばやく提供され、これらの採用と、(言うまでもなく) ザイリンクス FPGA の使用が容易になりました。

ザイリンクスと INTERLAKEN : 完璧な組み合わせ

ザイリンクスの製品は、設計者が最新の高性能な標準規格を簡単に導入できることに定評があります。100Gbps 以上への移行に際して、ザイリンクスのデバイスは、主要なパケット処理システムとテスト機器内で計り知れない価値があることをを実証しました。100G イーサネット MAC、OTU4、Interlaken ソリューション用の高品質な IP コアが提供されており、これはザイリンクスの高速 FPGA ファブリックおよびワールドクラスのトランシーバーを補完するもので、柔軟で強力なソリューションを顧客に提供します。

ザイリンクス FPGA 向け Interlaken IP コアは Virtex®-5 の世代から利用可能であり、当初はサードパーティ ベンダーから提供されていましたが、Sarance 社の買収後はザイリンクスから提供されるようになりました。UltraScale™ ファミリの FPGA は、Interlaken ソリューションを提供するザイリンクスのデバイスとしては第 4 世代に当たり、1 つの重要な進化を実現しています。

UltraScale FPGA では、ザイリンクスは Interlaken IP コアを FPGA のシリコン自体に統合しました。Interlaken を固定された機能にすることで、利用可能なファブリック ロジックを増やして、ソフト IP コアのインプリメンテーションに伴うタイミング調整の手間を減らし、設計者の負担を軽減しています。また、この統合されたソリューションは、柔軟性を損なわずに、ダイナミック消費電力とスタティック消費電力の両方を削減します。1 つの Interlaken ブロックで、最大 12.5Gbps までの任意のライン レートで最大 12 レーン、最大 25Gbps までのライン レートで最大 6 レーンをインプリメントできます。

INTERLAKEN の利点を活用

統合された Interlaken IP コアは、ザイリンクスの UltraScale FPGA の魅力の一端にすぎません。幅広いライン レートに対応する、豊富な機能を備えた多数のトランシーバー、先進の高速プログラマブル ロジック、MAC ソリューション用の統合されたソフト IP コア、新しい標準規格をインプリメントする性能は、ほかの FPGA や ASSP の追随を許しません。

UltraScale FPGA に搭載された GTH および GTY トランシーバーは、幅広い状況で高い性能を発揮する多くの機能を備えています。GTH トランシーバーは 500Mbps から 16.3Gbps までのライン レート、GTY トランシーバーは最大 32.75Gbps までの任意のライン レートで動作するため、FPGA はリンク先の機器が要求するあらゆるライン レートをサポートします (図 2 の YouTube のビデオ デモを参照)。これらのトランシーバーの等化機能 (連続時間線形等化と判定帰還等化を含む) により、ボード上のほかのデバイスからオプティクスまであらゆるものへの接続、あるいはバックプレーンの反対側のデバイスへの接続が可能となります。

100G イーサネット、OTU 4.4、広帯域 Interlaken インターフェイスなどの高速インターコネクトの性能発揮を容易にするため、レシーバーの等化機能はすべて自動適応型です。つまり、GTY トランシーバー内の 3 タップの CTLE と 15 タップの DFE の間に生じる数百万種類の組み合わせは、トランシーバーそれ自体によって処理されます。プロセス、電圧、温度の変化に対してリンク上のマージンを維持するために、設計者が各リンクを手作業で調整する必要はありません。Interlaken は限りなく幅広い各種のインターフェイスをサポートし、自動適応機能が簡単

で確実な性能発揮を実現します。

統合された Interlaken ブロックは、UltraScale 統合型 100G イーサネット MAC (C-MAC) によって補完されます。統合された Interlaken ブロックと同じ利点をすべて備えた CMAC は、現在の 100G オプティクス (CFP、CXP、CFP2、CFP4、QSFP28) に即時対応するインターフェイスを提供し、消費電力、複雑性、インプリメンテーション時間の削減を実現します。これらのオプティクスに接続するには、10x10.3125Gbps または 4x25.78Gbps シリアル インターフェイスが必要です。CMAC と UltraScale トランシーバーは、これらの 2 種類のインターフェイス (10.3125Gbps 用の GTH インターフェイスと 10.3125Gbps および 25.78Gbps 両用の GTY インターフェイス) をサポートします。CMAC はライン カード出力のインターフェイス用のプロトコルを 1 つだけ提供します。ザイリンクスは、OTN アプリケーションまたはさまざまなイーサネット規格 (1G、10G、40G、100G、新しい 400G 規格) 用に、ファブリック内にインプリメント可能な IP コアを提供します。

FPGA の重要な用途として、多数の Interlaken コンフィギュレーションを利用する ASSP 間のブリッジングと、MAC 機能のインプリメンテーションの 2 つが挙げられます。しかし、FPGA デザイン開発にはさらに大きな可能性があります。先進的な設計者は、ASIC や ASSP で実行するにはコストがかかる機能を FPGA に処理させて、システムをインテリジェントに最適化する試みに取り組んでいます。これらの機能には、さまざまなレベルのパケット処理、TCAM に対するルックアサイド インターフェイスのインプリメンテーション、あるいは製品の競争力を高めるその他の機能が含まれます。

すべてのテクノロジと同様に、高帯域幅の世界も進化を続けています。このことはスループットの向上と新しい標準規格への対応を意味しますが、この 2 つの分野でザイリンクスの UltraScale デバイスはほかの追随を許しません。ザイリンクスは、UltraScale FPGA に Interlaken を統合することにより、既存の 100G システムおよび将来のすべてのシステムへの新しい標準規格の即時導入、既存のアーキテクチャの機能強化、ボード全体でのシステム統合の向上を実現します。

FPGAs Help Characterize Massive-MIMO Channels

FPGA を利用した Massive MIMO チャネルの特性評価

Patrick Murphy
President
Mango Communications
patrick@mangocomm.com

Clayton Shepard
Graduate Student
Rice University
cws@rice.edu

Lin Zhong
Associate Professor
Rice University
lzhong@rice.edu

Chris Dick
Chief DSP Scientist
Xilinx, Inc.
chris.dick@xilinx.com

Ashutosh Sabharwal
Professor
Rice University
ashu@rice.edu

マルチユーザー MIMO (MU-MIMO) とは、基地局やアクセス ポイントなどのインフラストラクチャ ノード側で複数のアンテナを使用して多数のクライアントに同時にサービスを提供する無線通信技術です。MU-MIMO は今後のワイヤレス規格に不可欠の要素であり、通信量の多いネットワークのパフォーマンスを大幅に改善することが期待されます。

無線システムの新しい世代が登場するたびに基地局のアンテナの数は増えていき、ついには「Massive MIMO」システムが実現されるものと考えられています。Massive MIMO とは、MU-MIMO 基地局のアンテナ数を数十本または数百本に増やし、パフォーマンスの向上と基地局の信号処理の簡素化の可能性を追求する手法です。拡張性の高い Massive MIMO 手法の 1 つは、共役ビームフォーミングと呼ばれています [1]。この手法の早期インプリメンテーションは、実際にパフォーマンスが向上する可能性を示しています [2]。

マルチユーザー MIMO 手法は、無線伝搬環境の正確な情報に依存します。MU-MIMO インフラストラクチャの各ノードは、各ユーザーへの無線チャネルに関する最新の正確な測定値を取得している場合にのみ、複数のユーザーに同時にサービスを提供できます。このチャネル情報をリアルタイムで収集するのは簡単ではありません。情報が古かったり不正確だったりすると、パフォーマンスに大きな影響を与えることがあります。

筆者らは、研究者が Massive MIMO チャネルのダイナミクスをリアルタイムで実験できるように、Massive MIMO チャネル特性評価用の統合型システムを開発しました。このシステムは、ザイリンクス FPGA ベースの WARP ハードウェア プラットフォームと Mango Communications 社の 802.11 リファレンス デザインを中心部に採用しており、ライス大学の Argos プラットフォーム [2] を使用して、96 本のアンテナに接続された 24 個の FPGA にスケールアップできます。Mango Communications 社が開発したカスタム Python フレームワークは、Argos アレイの各ノードからのデータをリアルタイムで制御、収集します。このように Mango Communications 社が開発したツールとライス大学が開発したツールを組み合わせて、Massive MIMO の特性評価に必要な生のチャネル データを含む、ワイヤレス スタックの状況を詳細に可視化できます。

Mango Communications 社によるカスタム 802.11 インプリメンテーションの主な機能は、すべての受信アンテナから、AGC ゲイン、チャネル推定値、生のパケットの内容 (エラーありのパケットを含む) などの低レベルのベースバンド パラメーターをリアルタイムでストリーミングする機能です。リファレンス デザインのこうした機能により、ライス大学の Argos アレイは、802.11 標準規格に準拠したアクセス ポイント (AP) として動作し、商用 Wi-Fi デバイス (スマートフォン、タブレット、ノートブック PC など) にインターネット機能を付与すると同時に、アレイ アンテナと各クライアント間のチャネル データをリアルタイムで収集します。各アンテナでリアルタイム処理を実現するには、ザイリンクスの FPGA が非常に重要です。FPGA は、各アンテナから収集したデータを集約し、カスタム アプリケーションによるストリーミングと分析が可能な、クライアントごとのチャネル特性に変換します。

ここで、Mango Communications 社が開発した WARP ハードウェア プラットフォームおよびカスタム 802.11 インプリメンテーションと、MU-MIMO 向けの共役ビームフォーミング手法について詳しく説明します。次に、無線チャネルの測定値を Wi-Fi クライアントからリアルタイムで収集する方法、チャネルデータを処理して達成可能な MU-MIMO パフォーマンスを推定する方法など、特性評価プロセスについても検討します。

システム コンポーネント

WARP (Wireless Open-Access Research Platform) は、先進の無線ネットワークのプロトタイピング用にゼロから開発された、スケーラブルで拡張性の高いプログラマブル無線プラットフォームです。WARP は、高性能なプログラマブル ハードウェアと、リファレンス デザインおよびサポート資料からなるオープンソース リポジトリを組み合わせたものです。

WARP プロジェクトは、2006 年にライス大学の Ashu Sabharwal 教授によって設立されました。当初アメリカ国立科学財団の資金提供を受け、ザイリンクスからも継続的な支援を受けています。その後、WARP プロジェクトは世界中のユーザーの協力のもとに自立的なオープンソース活動に成長しました。Mango Communications 社は、ライス大学の WARP ハードウェアの製造と販売を初期の目的として、2008 年にライス大学の WARP プロジェクトから分離独立した企業です。2012 年、Mango 社は、設計を一新した WARP v3 ハードウェアを発表しました。現在、Mango 社の技術者は、WARP リポジトリおよび各フォーラムに積極的に貢献し、オープンソースの WARP デザインの継続的な開発とサポートに取り組んでいます。

筆者らの Massive MIMO チャネル測定用システムの中心となるコンポーネントは、Mango Communications 社の WARP v3 ハードウェア プラットフォームです。WARP v3 は、新しいワイヤレス デザインをリアルタイムで迅速に試作できるように設計されています。WARP v3 ハードウェアは、高性能なザイリンクス Virtex®-6 FPGA、2 つの柔軟な RF インターフェイス、複数のペリフェラル (DDR3 DRAM および 2 つの 1Gbps イーサネット インターフェイスなど) を統合した製品です。WARP v3 ボードは、Mango 社のデュアル無線 FMC モジュールを使用して、4 つの RF インターフェイスに拡張できます。図 1 に示したハードウェア コンフィギュレーションは、FPGA に対する互いに独立したデジタル ベースバンド接続ポートを備えた、完全にプログラマブルな 4 つの RF インターフェイスを提供します。

Massive MIMO システムについて研究するには、電源、クロッキング、イーサネット接続を共有する複数の WARP v3 ノードを同じ場所に配置する必要があります。この条件は、ライス大学の Argos プロジェクトによって提唱されてきたものです。Argos v2 アレイは、4 本のアンテナを搭載した WARP v3 ノードを 24 個集めたものです

図 1 – デュアル無線 FMC モジュールを搭載した WARP v3 ハードウェアは、大規模 FPGA、4 つの RF インターフェイス、メモリ、2 つのイーサネット接続ポートを備えています。

図 2 – ライス大学の Argos v2 アレイは、クロッキングとイーサネット接続を共有するクワッド無線アンテナ装備の WARP v3 ノードを 24 個組み合わせたものです。

(図 2 を参照)。Argos アレイは、多種多様の Massive MIMO の実験サポート用に設計され、全 96 本のアレイ アンテナからチャネル測定値を同時に収集するのに最適です。

Argos アレイ内の各 WARP v3 ノード上の FPGA は、RF インターフェイスの近くで大きな処理能力を提供します。Argos などの Massive MIMO コンフィギュレーションでは、非常に大量のデータを処理する必要があります。たとえば、40MHz の帯域幅を受信する場合、WARP v3 上の各 RF インターフェイスは、960Mbps のサンプル ストリーム (12 ビット、40 メガサンプル / 秒の ADC の 2 基分のデータ) を生成します。Argos アレイ全体ではこの量の 96 倍のデータを生成しますが、この量は PC にストリーミングしてリアルタイム処理するには大きすぎます。そこで、このシステムは FPGA を使用してデータをローカルでリアルタイム処理し、上流側のプロセッサの負担を大幅に軽減します。筆者らの Massive MIMO チャネル特性評価デザインでは、このリアルタイム処理が非常に重要です。リアルタイム処理により、この特性評価システムはチャネルを継続的に測定し、チャネル特性のミリ秒以下の変動を高い信頼性で観察できます。この処理を実行するカスタム FPGA デザインが、Mango Communications 社の WARP v3 向け 802.11 リファレンス デザインです。

このリファレンス デザインは、802.11a/g 標準規格の媒体アクセス制御 (MAC) 層および物理 (PHY) 層をリアルタイムで FPGA にインプリメントしたものです。このデザインは、標準 Wi-Fi デバイスと相互運用性があり、AP (Wi-Fi クライアントにサービスを提供する)、クライアント (Wi-Fi AP にアクセスする)、またはモニター (ネットワーク アクティビティの受信専用の受動的オブザーバー) として動作します。MAC と PHY の両方をカスタマイズして、標準規格の新たなバリエーションを検討できます。優れた相互運用性と拡張性を組み合わせて、無線通信とネットワーキングの幅広い実験が可能です。WARP v3 ハードウェアのユーザーには、この 802.11 リファレンス デザインの完全なソースが無償で提供されます。

図 3 に、リファレンス デザインのアーキテクチャを示します。このデザインは、2 つのザイリンクス MicroBlaze™ コアを使用して、高水準および低水準の MAC プロトコルをソフトウェアにインプリメントします。MAC は 2 つの FPGA コアに接続され、これらのコアが PHY トランスミッターとレシーバーをインプリメントします。筆者らは、これらの PHY コアを Xilinx System Generator でインプリメントしました。トランスミッター コアは、バイトから波形までのパイプライン全体をインプリメントします。このコアは、MAC からパケット ペイロードを読み出し、OFDM 波形を生成し、波形を RF インターフェイスの DAC にドライブします。このパイプラインには、エンコード、スクランブル、

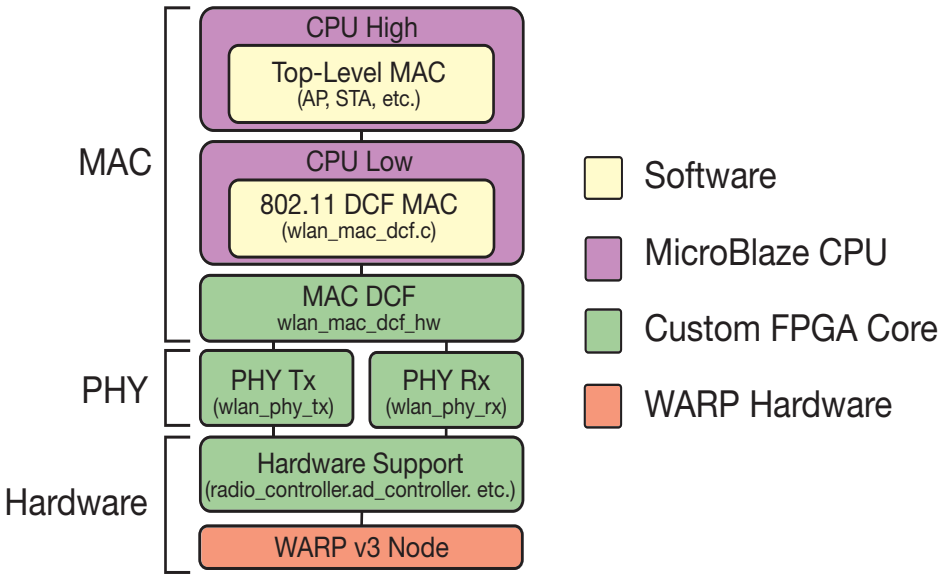


図 3 – Mango 社の 802.11 リファレンス デザインのアーキテクチャは、MAC 用に2 つのザイリンクス MicroBlaze CPU を、PHY トランスミッター/レシーバー用にカスタム System Generator コアを搭載しています。

インターリーブ、IFFT、プリアンブル挿入が含まれます。MAC はパケットごとに変調方式とコーディング レートを指定します。802.11a/g 標準規格で指定された 8 種類のデータ レートをすべてサポートします。

レシーバー デザインは、AGC、同期、FFT、チャネル推定、等化 (イコライゼーション)、検出、デコードを含む、波形からバイトまでのパイプライン全体をインプリメントします。レシーバーは、パケットの SIGNAL フィールドの RATE の値を使用して、パケットごとに復調方式とデコード ブロックを自動的に設定します。レシーバーは、パケット受信時の肯定応答 (ACK) の送信について規定した 802.11a/g 標準規格の厳格な Rx-Tx 間ターンアラウンド要件を満たす、十分に高速な任意のレートのパケットをデコードします。

特性評価用 Massive MIMO チャネルの中心となる、筆者らのレシーバー デザインの 1 つの特徴は、チャネル推定 (エスティメーター) サブシステムです。標準的な OFDM レシーバーでは、チャネル エスティメーターは副搬送波ごとに複素チャネル係数を生成します。イコライザーは、これらの推定された係数を使用して、受信した各データ シンボルについてチャネルの振幅と位相の劣化を補正します。これ以外に、筆者らのデザインは、受信した各パケットのチャネル推定値のコピーをオンチップ メモリ領域に保存します。MAC は、Rx 電力、AGC ゲインの選択、チェックサム ステータス、アンテナの選択などの標準的な情報とともに、受信したフレームに関する剰余のメタデータとして、これらのチャネル推定値を扱います。チャネル推定値は、さらなる処理のために、より高レベルの MAC にコピーされます。筆者らの特性評価プラットフォームは、Argos アレイ内の各ノードが受信したすべてのパケットからこれらの推定値を収集し、Massive MIMO 伝搬環境のリアルタイム ビューを生成します。

WARPnet 実験用フレームワーク

筆者らの Massive MIMO 特性評価システムの最後の要素は、WARP ノードの大規模なネットワークの実験に使用される WARPnet と呼ばれるフレームワークです。WARPnet は、複数の WARP ノードに対する専用の制御接続を使用するカスタム Python パッケージです。WARPnet フレームワークでは、PC 上で実行される Python スクリプトが遠隔からリアルタイムで実験用パラメーターを設定し、実験データを取得できます。WARPnet は、各 WARP v3 ボードのセカンダリ イーサネット接続を介して、Mango 社の 802.11 リファレンス デザインと相互作用します。上位の MicroBlaze デバイスは、WARPnet コマンドを処理し、WARPnet フレームワークが、WARP ノードの高レベル MAC ステートと、低レベル MAC および PHY から渡されるすべてのデータに、直接アクセスできるようにします。

筆者らの Massive MIMO チャネル特性評価デザインでは、WARPnet フレームワークが Argos アレイ内の各ノードへの接続を維持します。各ノードは 802.11 モニターとして構成され、各受信パケットからチャネル推定値を収集し、これらのパケットを詳しい分析のためにイーサネットを介してオフロードします。

WARPnet 用の完全な Python ソースコードは、WARP リポジトリにオープンソースとして公開されています。

マルチユーザー MIMO の概要

マルチユーザー MIMO 手法を採用した基地局は、無線チャネルと組み合わされた時に複数のユーザーに同時にデータを送信する多数の送信アンテナのための波形を生成しようとします。マルチユーザー向けの波形を生成するには、基地局側で高度な処理が必要です。この目的で多くの MU-MIMO 手法が提案されてきました。MU-MIMO デザインの一般的な要件として、各基地局アンテナから各クライアント機器への無線伝搬特性に関する正確な情報が必要です。

MU-MIMO の 1 つの手法は、ゼロフォーシングと呼ばれています。この手法は、理論的には (最近の実施例 [3]) 単一ユーザー手法に比べて性能が大幅に向上することがわかっています。ゼロフォーシング手法は、各クライアントの受信アンテナでの SINR (Signal-to-Interference-plus-Noise Ratio) を最大化するのを目的にしています。SINR を最大化するには、一人のユーザーのアンテナに着信する波形内で、そのユーザーのペイロードを表す信号電力 (SINR の「S」) を最大化し、その他のユーザーのペイロードの電力 (SINR の「I」) を最小化する必要があります。ゼロフォーシングは、基地局側で非常に高度な処理を必要とします。ゼロフォーシングを使用する場合、1 つの特定の

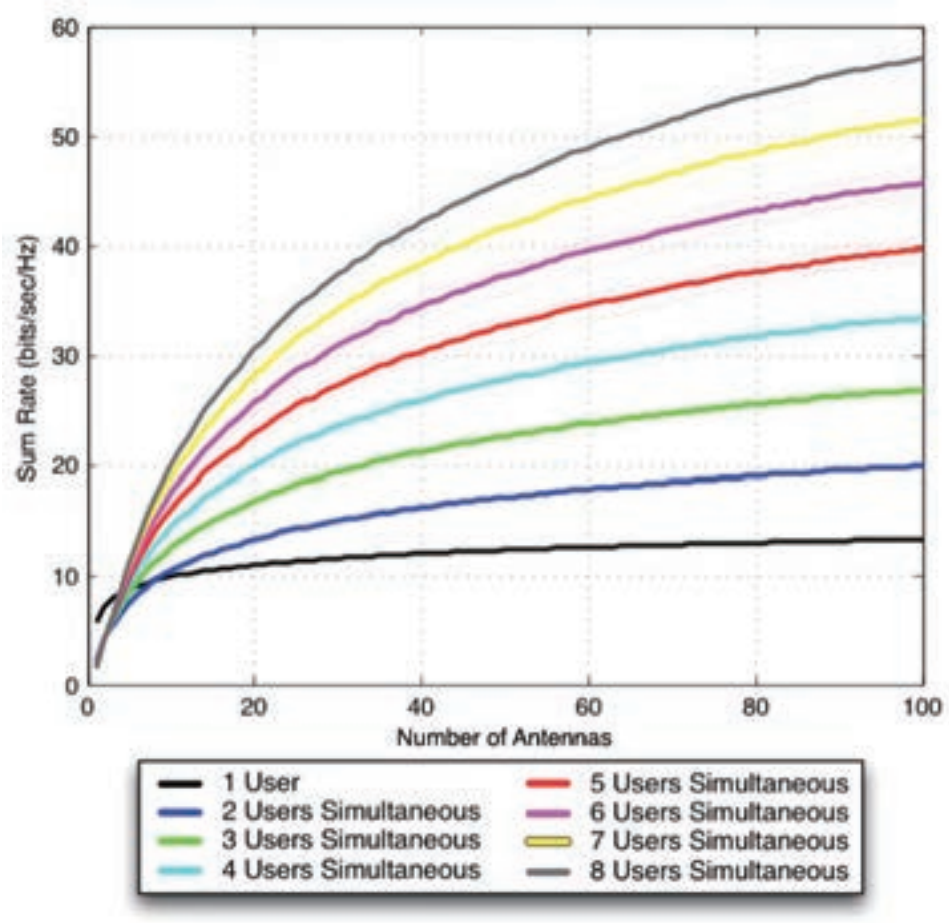


図 4 – マルチユーザー MIMO ネットワークのシミュレーションは、アクセス ポイントに十分な数のアンテナが設置されている場合、多数のユーザーに対する通信速度が大幅に向上することを示しています。

基地局アンテナの送信波形を計算するには、すべてのユーザーのペイロードと、ほかのすべてのアンテナからすべてのユーザーへの無線チャネルのペイロード情報が必要です。基地局アンテナの数が増えるにつれて、この計算の複雑性は大きく増大します。

共役ビームフォーミング [1] は、代替的な MU-MIMO 手法です。この手法では、基地局は、干渉電力を積極的に最小化せずに、各クライアント機器に送信される良好な信号電力を最大化しようとします。理論的には、信号電力 (SINR の「S」) を最大化し、干渉電力 (SINR の「I」) を無視することによって各ユーザー側の SINR を向上させるこの共役ビームフォーミング手法は、アンテナの数が多くなるほど効果が上がります。その上、共役ビームフォーミングによる各送信アンテナの波形の計算には、その他のアンテナのチャネル特性に関する情報は不要です。これらの要因により、共役ビームフォーミングは、基地局がユーザーよりもはるかに多くのアンテナを備える Massive MIMO システムに最適と言えます。

シャノンによる古典的なチャネル容量の式 C = log(1+SINR) を考えます。無線チャネルの容量 (ビット / 秒 /Hz 単位) は、SINR に対して対数的に増加します。システムのユーザー数とアンテナ数が増えた場合、共役マルチユーザー ビームフォーミングは、2 つの競合する効果をもたらします。第 1 に、アンテナが複数あると、各アンテナはユーザーのレシーバー側で送信信号が構造的に結合されるように位相を回転するため、受信信号電力が大きくなります。第 2 に、互いに独立したユーザーに対して複数の送信信号が送られると、干渉電力が大きくなります。重なり合う干渉信号はランダムに結合されます。アンテナの数が増えるにつれて、構造的に結合される信号電力の増大がランダムに結合される干渉電力を上回り、全体として SINR が向上します。

図 4 のシミュレーション結果は、共役ビームフォーミングの使用時に基地局のアンテナ数の増加が全体的なネットワーク容量に与える影響を示しています。このシミュレーションでは、1 つの基地局と 8 人のユーザーからなるネットワークを想定し、互いに独立して同一の分布に従うレイリー フェージングによって無線チャネルをモデル化します。このシミュレーションは、基地局で使用するアンテナの数に対して、1 ～ 8 人のユーザーに同時にサービスを提供する場合の全体的なネットワーク通信速度を示します。少数のアンテナでは、一度に複数のユーザーに対して共役ビームフォーミングを使用してもメリットがないことがわかります。基地局のアンテナが数本しかない場合は、従来のタイム シェアリングに基づくシングルユーザー ビームフォーミングの方が、マルチユーザー共役ビームフォーミングよりも優れています。アンテナの数が増えるにつれて、より多くのユーザーのサポートが可能となり、ネットワーク通信速度は全体として大幅に向上します。

このシミュレーションでは、理想的なチャネル モデルを使用して、マルチユーザー共役ビームフォーミングによる性能向上が可能なことを示しています。実際のシステムで性能向上が可能かどうかは、基地局とクライアント機器間の実際の無線チャネルによって決まります。筆者らの MU-MIMO チャネル特性評価プラットフォームは、基地局から実際のユーザー デバイスまでのチャネルをリアルタイムで測定できるため、MU-MIMO 手法の実際の性能を評価する強力なツールとなります。

システムの各要素を統合

ここまで、Massive MIMO チャネルの測定が必要な理由と、ライス大学の Argos アレイ、WARP ハードウェア、Mango 社の 802.11 リファレンス デザインによって提供されるツールについて説明してきました。次に、これらの要素を組み合わせて完全なリアルタイムの Massive MIMO チャネル特性評価プラットフォームを構築する方法を説明します。

Argos アレイ内の 24 個の WARP v3 ノードは、Mango 社の 802.11 リファレンス

デザインのカスタム バージョンで構成されます。このバージョンは、受信専用モニター モードで動作し、各ノードの 4 本のアンテナすべてでパケットを受信しようとします。パケットを受信するたびに、WARP ノードは各副搬送波について複素チャネル係数を推定し、パケットをデコードして、分析のためにイーサネットを介してパケット ヘッダーとチャネル推定値を送信します。この処理フローはアレイ内の 24 ノードすべてにインプリメントされ、すべてのノードは並列に動作します。

このチャネル測定プラットフォームが標準 Wi-Fi デバイスと通信するには、標準規格に準拠した 802.11 アクセス ポイント (AP) もインプリメントする必要があります。Mango 社 の 802.11 リファレンス デザインを AP モードで実行する、もう 1 つの WARP v3 ノードがこの目的で使用されます。この AP ノードは、Argos アレイ内の 25 番目のノードとして機能します。この AP はオープンな Wi-Fi ネットワーク上に公開され、商用 Wi-Fi デバイスからの接続を受け入れ、プライマリ イーサネット接続を介してインターネット アクセス サービスを提供します。

これは Mango 社の 802.11 リファレンス デザインの AP プロファイルの標準動作です。リアルタイムのチャネル測定を可能にするために、この AP に 1 つの追加機能をインプリメントします。AP ノードは、WARP v3 ボードのセカンダリ イーサネット接続を使用して、Wi-Fi クライアントが無線ネットワークに参加または離脱するたびにイーサネット パケットを送信します。次に説明するチャネル分析アプリケーションは、これらの関連付け (接続) の更新を使用して、アクティブ クライアントのローカル テーブルを維持します。

クライアントの送信頻度

商用 Wi-Fi デバイスから受信したパケットからチャネル測定値を収集する際の主な課題の 1 つは、デバイスに十分な頻度で送信させることです。現在の Wi-Fi デバイスは、しばしば積極的な省電力機能を採用しており、アプリケーションがネットワーク アクセスを要求しないときは Wi-Fi 無線通信を無効にします。これらのデバイスは定期的に AP にチェックインしますが、アレイ側で最新のチャネル測定値を確実に取得するには、おそらく間隔が空きすぎます。

筆者らは、2 つの手段でクライアントの不十分な送信頻度の問題に対処しています。第 1 に、接続しているすべてのクライアントに新しいデータ パケットがキューに入れられたことを知らせるためにプラットフォーム AP によって送信されるビーコンの TIM (Traffic Indication Map) フィールドを変更します。TIM フィールドは、通常はクライアント側で省電力機能のサポートに使用されます。クライアントは、受信専用モードで短時間ウェークアップしてビーコンを受信し、TIM をデコードし、トラフィックが待機していなければ省電力モードに戻ります。すべてのビーコンの TIM フィールドにすべてのノードをリストすれば、ノードがスリープに入る頻度は下がります。

第 2 の手法は、クライアント機器が送信した ACK パケットを使用して、クライアントが送信することをを要求します。アレイは、クライアントが送信する任意のパケット (短い ACK 信号を含む) からチャネルの推定値を抽出できます。しかし、802.11 規格の ACK パケットにはデスティネーション MAC アドレスしか含まれていないため、通常アレイは送信側クライアントを識別できません。

筆者らは、802.11 MAC 仕様の「特異な性質」を利用してこの問題を回避しています。この標準仕様は、802.11 クライアント デバイスが自分を宛先とするユニキャスト パケットの受信に成功したときは肯定応答パケットを送信することを求めています。この「ACK を送信しなければならない」要件は、パケットのソース アドレスを認識できない場合にも適用されます。したがって、クライアントがクライアントに固有の識別子を含む ACK を送信するように、AP は偽の (しかし固有の) ソース アドレスを持つデータ パケットを送信します。クライアントは、パケットを受信すると、AP が使用した偽の (しかし固有の) アドレスに ACK を送信します。アレイ ノードはこの ACK を受信し、得られたチャネル推定値と送信側クライアントを一義的に関連付けることができます。この手法は、アレイ側のチャネル推定値の頻繁な更新をトリガーするのに非常に効果的ですが、Mango 社の 802.11 リファレンス デザインに完全なプログラマビリティがなければ不可能でした。

リアルタイム分析

Massive MIMO チャネル測定プラットフォームの最後の構成要素は、アレイチャネルの推定値を収集し、達成可能なマルチユーザー容量を計算し、結果をリアルタイムで表示するカスタム アプリケーションです。筆者らは、このアプリケーションを Objective-C

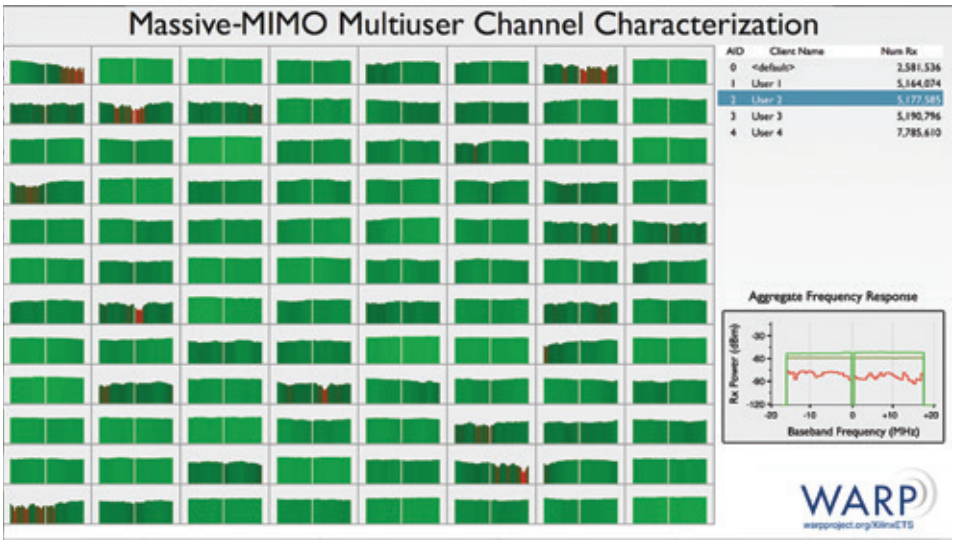


図 5 – 筆者らのカスタム MU-MIMO チャネル分析アプリケーションのチャネル振幅ビュー。それぞれの棒グラフは、1 つのアンテナおよび 1 つの副搬送波あたりの振幅を示します。

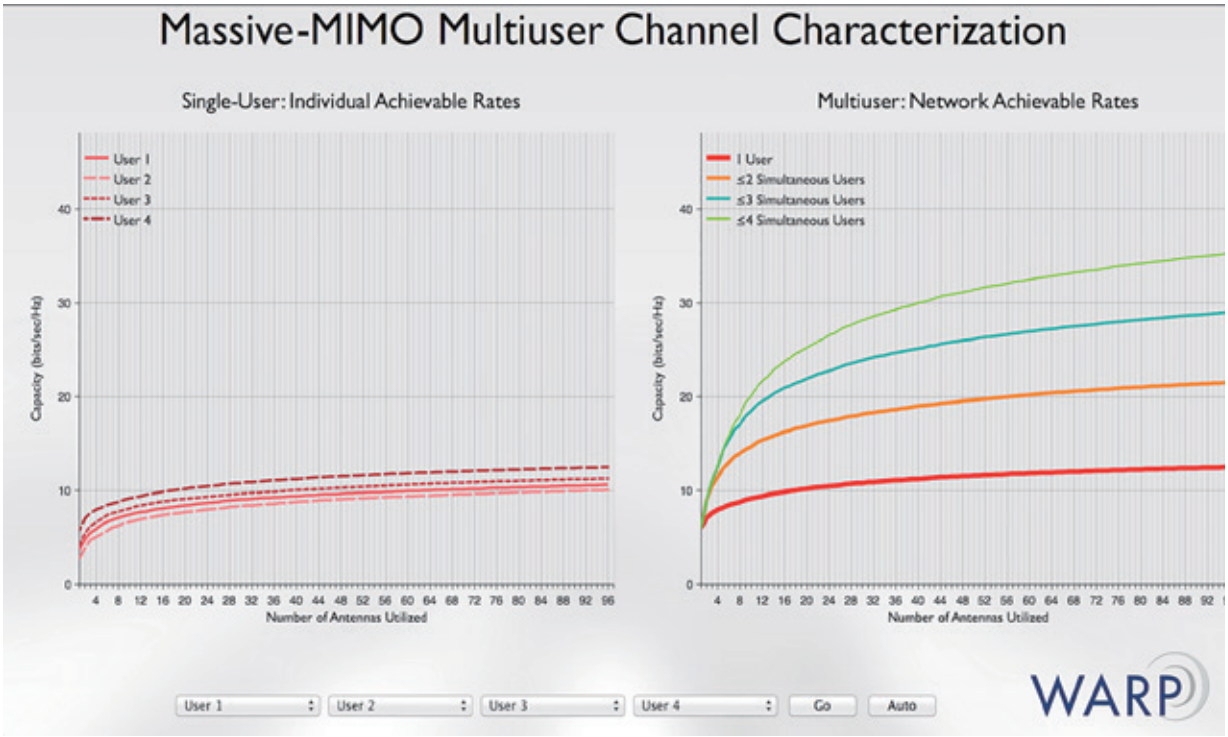


図 6 – シングルユーザーおよびマルチユーザー手法でのネットワーク容量の推定値は、筆者らのカスタム分析アプリケーションが表示しているように、アレイが収集した実際の MU-MIMO チャネル測定値から計算しました。

で開発しました。ネイティブ UDP ソケットを使用してアレイ内の WARP v3 ノードとのインターフェイスをとり、OS X グラフィックス フレームワークを使用して結果をプロットすることにしました。

このアプリケーションには 2 つの主要なビューがあります。第 1 のビューは、各アレイ アンテナがすべての副搬送波について収集する生のチャネル振幅を表示します。データ ポイントの数は全部で 4,992 個 (52 個の副搬送波 x 96 本のアレイ アンテナ) になります。このビューは、アレイによって収集されたチャネル データの生の表示で、主に個々のアレイ アンテナによって観察された幅広いチャネル値を伝えるのに役立ちます。図 5 に、このビューのスクリーン ショットを示します。実際にはこのビューはリアルタイムで (アクティブな Wi-Fi クライアントでは 1 秒当たり 10 フレーム) 更新されます。

アプリケーションの第 2 のビューは、アレイのチャネル測定値に基づいた容量計算の結果を表示します。図 6 にこのビューを示します。2 種類の容量計算が実行されます。第 1 の計算では、使用したアレイ アンテナの数に対する各ユーザーへの容量をプロットします。このプロット上のそれぞれの線は、アレイがアレイ アンテナの一部を従来のシングルユーザー ビームフォーミング コンフィギュレーションで使用したと仮定して、シングル ユーザーに対する達成可能なダウンリンク容量を近似的に示します。アンテナの数が増えるにつれて各容量曲線の傾きが小さくなることから、従来のシングル ユーザー ワイヤレス手法では多数のアンテナの効果が弱まっていくことがはっきりとわかります。

第 2 のプロットは、アレイがアレイ アンテナの一部を使用してダウンリンク マルチユーザー ビームフォーミング手法をインプリメントした場合の総ネットワーク容量を示しています。4 本のプロットの傾向は、マルチユーザー手法を採用した場合の追加アンテナのメリットをはっきりと示しています。ユーザーの数が増えたときに傾きが大きくなることから、マルチユーザー ビームフォーミングでは (MIMO の文献でしばしば「pre-log」と呼ばれる)「log の外側」でネットワーク容量が向上することがよくわかります。

筆者らは、2014 年 2 月のザイリンクス エマージング テクノロジ シンポジウム (ETS) で Massive MIMO チャネル測定プラットフォームのデモを行いました。このデモの詳細 (ビデオおよび補助資料へのリンクを含む) は、http://warpproject.org/XilinxETS/ をご覧ください。

参考資料

1. T.L. Marzetta, “Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antennas,” IEEE Transactions on Wireless Communications, vol. 9, no. 11, pp. 3590–3600, 2010

2. C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang and L. Zhong, “Argos: Practical Many-Antenna Base Stations,” Proceedings of ACM MobiCom, pp. 53–64, 2012

3. Q. Yang, X. Li, H. Yao, J. Fang, K. Tan, W. Hu, J. Zhang and Y. Zhang, “Bigstation: Enabling Scalable Real-time Signal Processing in Large MU-MIMO Systems,” Proceedings of ACM SIGCOMM, pp. 399–410, 2013

4K TV Development Made Easy with the Zynq SoC

Zynq SoC を利用して 4K テレビの開発を容易化

Roger Fawcett
Managing Director
OmniTek
roger.fawcett@omnitek.tv

ザイリンクスの All Programmable テクノロジは、4K ビデオ システムの設計者にとって非常に有益です。FPGA デザインに慣れていないユーザーには、関連ツール、IP コア、リファレンス デザインが役に立ちます。

超高精細 (UHD) テレビ (解像度に基づいて 4K とも呼ばれる) は既に広く利用可能になり、消費者の間で 4K は 3D テレビよりもはるかによく利用されるテクノロジを提供しています。しかし、標準規格はこうした急速な普及に追いついていません。4K60 ビデオをサポートする 6Gbps および 12Gbps SDI の SMPTE (Society of Motion Picture & Television Engineers) 規格はつい先頃発表されたばかりで、同じ解像度をサポートする HDMI ™ 2.0 および DisplayPort の普及はまだ初期段階にあります。4K UHD テレビに対する消費者の大きな需要を見越して、この空隙を埋めようとする多くの暫定的な規格が登場しています。

確かに 4K UHD テレビについては非常に多くの内容がまだ流動的であるため、システムは発展途上の標準規格に適応できる柔軟性が要求されます。柔軟性を確保する方法としては、長い間 4K システムのデザインに使用されてきた従来のチップ セットと ASSP を、FPGA とザイリンクスの Zynq®-7000 All Programmable SoC などの All Programmable システムオンチップ (SoC) で置き換えることです。これらのソリューションは、必要とされる柔軟性と、ASIC と同等クラスの性能を提供します。

同時に、最新の FPGA と SoC のサイズと性能では、特に FPGA の経験が浅い技術者は、多くの設計上の課題に直面します。ハードウェアの設計と FPGA のインプリメンテーションには多くの共通点があるとは言え、FPGA ベースのシステムは通常、はるかに多くのコンポーネントで構成されます。その上、FPGA ではファームウェア デザインの柔軟性が高いため、設計作業はさらに複雑になります。

幸いにも、ザイリンクスは多くの手段で 4K テレビの設計者をサポートしています。したがって、コスト的にはシステムをゼロから設計するよりも、時間と予算の両面ではるかに少なくなります。しかし、FPGA テクノロジを利用した 4K アプリケーションの開発方法について詳しく説明する前に、4K システムが短期間にこれほど広く普及した理由と、4K システムが解決しなければならない課題について簡単に説明しましょう。

4K の普及要因と課題

テレビが発明されて以来、映し出される映像を現実に近づけようとする試みは常に続けられてきました。この試みは、一般に、画面の解像度、フレーム レート、またはダイナミック レンジの向上 (すなわち、どれだけ画面を明るくできるか) による画面の大型化、高画質化、ビデオの高速化に帰着します。それに加えて、(言うまでもなく) 本格的な 3D 効果または (少なくとも) より臨場感あふれる感覚の実現も試みられます。

解像度が向上するにつれて、映像はより精細になり、ピクセレーションが目立たないように大画面に表示できるようになります。画面が大型化すると、臨場感が高まります。これはユーザーが簡単に楽しめる改善点なので、お金をかける気持ちになりやすくなります。一方、フレーム レートの向上 (より滑らかな動き) またはダイナミック レンジの向上 (明るい箇所はより明るく、黒い箇所はより暗く) による改善は、魅力的ではありますが、これまでのところあまり顧客の関心を引き付けてはこなかったようです。

新しい 4K UHD テレビでは、これまで好評だった HD 規格のピクセル数が 4 倍になります。おそらくユーザーにとって最も重要なのは、4K への移行により、画質に明らかな影響がみられなくても、はるかに臨場感あふれる、はるかに大画面のテレビにアップグレードできることです。

しかし、4K ビデオをサポートするシステムの開発には、多くの技術的課題があります。まず第 1 に、3,840 x 2,160 ピクセルのフレーム サイズを最大 60Hz のフレーム レートで実現するには、600MHz のピクセル レートが必要になります。これだけのレートをリアルタイムで処理するには、非常に高性能のシステムが必要です。また、4K についてはさまざまな配信コンフィギュレーションが定義されていますが、そのすべてに複数のデータ ストリームが含まれ (一部は同じケーブル上で、一部は異なるケーブル上で多重化される)、このデータ ストリームを供給するさまざまなテクノロジ (4x3G、6G-SDI および 12G-SDI、HDMI 1.4 および HDMI 2.0、DisplayPort 1.2、V-by-One HS) が登場しています。

設計者のもう 1 つの課題は、4K ビデオ システムは、4K 規格だけでなく、現在使用されている (すべてではないにしても) 多くのビデオ規格 (SD を含む) を処理する必要があることです。また、4K ビデオ システムは、各種の規格間の変換と、それに関連するアップ / ダウン / クロス コンバージョン、カラー スペースの不一致、カラー補正、インターレースおよびインターレース解除、ケイデンス処理などのすべての問題をサポートしなければなりません。さらに問題を複雑にしているのは、通常はアップコンバージョンの結果として不可避的に発生する画像の平滑化 (スムージング) に対抗するため、いわゆる「超解像度」強化技術のアプリケーションを実行する必要があることです。

その他の必要な処理には、ノイズ リダクション、クロッピング、リサイジングなどがあり、すべてをリアルタイムで実行する必要があります。システムによっては、HDCP (High-bandwidth Digital Content Protection) の処理も必要です。

さらに、ブロードキャスト送信の品質を確認する必要がある場合には、適切なアイおよびジッター表示を生成する必要がありますが、このためのテクノロジのインプリメンテーションは、ビット レートが高くなるほど難しくなります。

最初の段階でのサポート : 4K IP コア

あらゆるシステム デザインの最初の手順は、デザインに利用できる既製のブロックを見つけることです。FPGA の世界では、PCB デザインに利用できる各種のチップと等価なビルディング ブロックは、IP (知的設計資産) コアと呼ばれます。最初に、どのような IP コアが 4K UHD デザインに利用可能であるかを確認する必要があります。

OmniTek 社は、あらゆるタイプのビデオ システム デザインに適した IP コアの優れた供給元です。同社はザイリンクス アライアンス プログラムの認定メンバーで、自社のビデオ試験測定 (T&M) システムの開発から出発して、ビデオ処理の分野で豊富な経験を積んできました。T&M システムに必要な専用ハードウェアの開発は、専用ファームウェア ブロックの開発につながりました。これらのファームウェア ブロックは、現在でも IP コアとしても利用可能です。OmniTek 社の最新の T&M システム (新たに発表された Ultra 4K Tool Box) の開発は、各種の 4K 対応 IP コアの開発につながり、現在ではサードパーティの開発者が利用できるようになっています。

4K システムの設計者には、OmniTek 社の OSVP v2 Scalable Video Processor と Multi-Channel Streaming DMA Controller の 2 種類の IP コアが特に有益です。これらの IP コアは、いずれもザイリンクス 7 シリーズ FPGA および Zynq SoC をターゲットとして提供され、ARM® AMBA® AXI4 システム インターコネクト規格を採用しています。

OSVP v2 の機能には、6 軸カラー補正、動き適応型およびエッジ適応型インターレース解除 (3:2 および 2:2 フィルム ケイデンス検出および処理機能を完全装備)、画像の鮮明化およびスムージング機能付きリサイズおよびクロップ機能、ノイズ リダクション

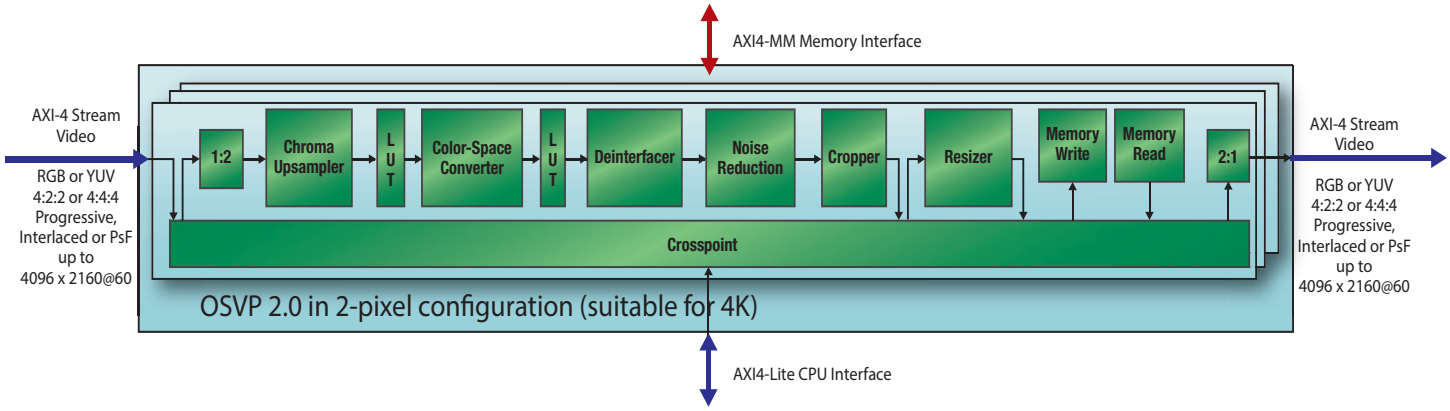


図 1 – OmniTek 社の OSVP v2 Scalable Video Processor コアの入力チャネル アーキテクチャ

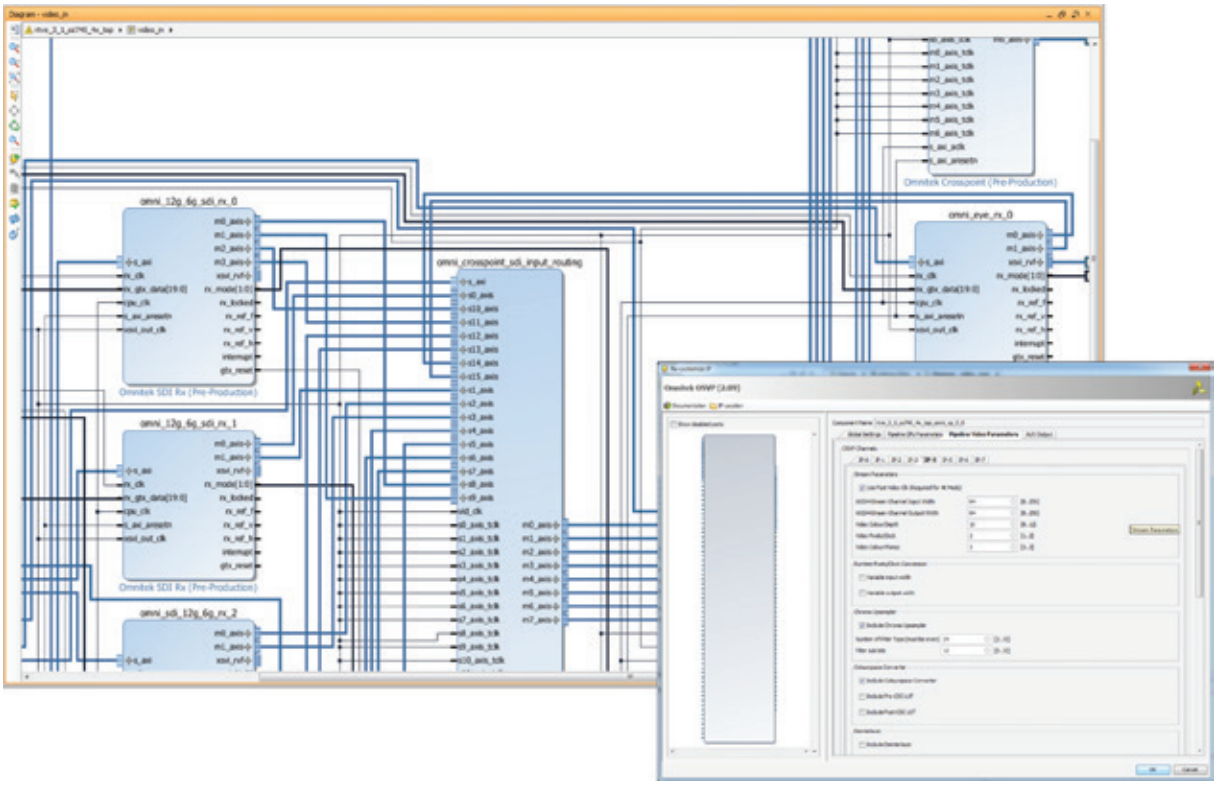


図 2 – Vivado IP Integrator 内でレイアウトされた RTVE 3.1 ビデオ デザインと、[Configuration] 画面

が含まれます。図 1 に、このコアのブロック図を示します。設計者は、コンパイル時に、搭載されている処理機能を選択してコンフィギュレーションします。OSVP v2 コアによって実行される処理の詳細は、実行時に設定することも、ソフトウェアから制御することも可能です。

OSVP v2 コアは、コア スイートの一部として提供されます。このスイートには、複数のビデオ ストリームを結合するコンバイナー、インターレース フォーマットで出力を生成するインターレーサー、専用のクロスポイント、4:4:4、4:2:2 および 4:2:0 間で YCbCr の移行に使用できるクロマ リサンプラが含まれています (1 つのクロマ リサンプラ コアで、4:4:4 から 4:2:2、4:2:2 から 4:2:0、4:2:2 から 4:4:4、または 4:2:0 から 4:2:2 への変換が可能です)。

1 つの OSVP v2 コアで複数のビデオ チャネルを処理できます。制約要因となるのは、コアがインプリメントされる FPGA または SoC が備えているリソースと、利用可能な SDRAM 帯域幅です。たとえば、8 種類の HD ビデオ規格、8 つのカラー スペースなどでビデオを処理する最大 8 つの入力用 に、Kintex®-7 XC7K325T FPGA 上にインプリメントされた OSVP コアをコンフィギュレーションできます。同時に、最大 16 のプログレッシブ HD 出力用に、出力ブロックをコンフィギュレーションできます。また、1 つの 4K チャネル用、あるいは 4 チャネルのセットによるスクエア ディビジョン (「クワッド」) または 2 ピクセル サンプル インターリーブ 4K 用に、出力ブロックをセットアップすることもできます。

複雑な 4K システムを設計する際のもう 1 つの課題は、ビデオの処理に必要となる、高帯域幅メモリへの大量のアクセスをどのように管理するかということです。場合によっては、必要なビデオ処理機能がビデオ処理ブロックとともに提供されます。たとえば、OSVP v2 コアには、ビデオ入力 / 出力処理用の高効率エンジンを備えたマルチポート ビデオ DMA ブロックが含まれています。

ただし、PCI Express® 上で 1 つ以上の 4K60 チャネルの取り込みと送信を行うには、PCIe® インターフェイス上のストリーミング データの処理に最適化された DMA コントローラーが必要です。OmniTek 社の Multi-Channel Streaming DMA Controller は、この目的に役立つ主な機能を備えています。1 つの機能は、メモリ間のデータ転送を不要にする、FIFO ベースの DMA (FDMA) です。もう 1 つの機能は、スキャッター ギャザー モード ディスクリプタのプリフェッチや、TLP パケットのバック トゥ バック パッキングなど、コントローラーによる PCIe 帯域幅の利用効率を高めることが可能な、一連のデザイン最適化機能です。

OmniTek 社が 4K UHD ビデオ用に開発したもう 1 つの IP コアは、2 サンプル インターリーブ形式の 4K ビデオを構成する、複数の異なるストリームを分解するためのブロックです。また、基本的 MIG SDRAM コントローラーの交換用のドロップイン モジュールにより、UHD TV ビデオ アプリケーションのパフォーマンスはさらに向上します。

優れたプログラマビリティ

ザイリンクスは、FPGA および SoC ベースの 4K ビデオ システムの設計者を、次の

3 つの形でさらにサポートします。

第 1 の利点は、Zynq SoC それ自体にあります。Zynq SoC は、高性能なビデオ処理または画像処理用に強力なハードウェアとソフトウェアの処理機能を組み合わせたデバイスです。Zynq SoC は、豊富な機能を備えたデュアルコア ARM Cortex™ -A9 プロセッシング システムと 7 シリーズ (28 ナノメートル) FPGA のプログラマブル ロジック部を 1 つのデバイスに統合しています。ユーザーは、処理アルゴリズムを ARM プロセッサ上で実行することも、あるいはリアルタイム動作を実現するために処理を高速化する必要がある場合、その処理を FPGA ハードウェアに振り分けることもできます。

Kintex-7 FPGA と Zynq SoC のプログラマブル ロジックは、いずれも持続的な 300MHz のビデオ処理速度を提供し、64 ビット DDR3 の 1,600Mbps のメモリ性能と合わせて、4K ビデオ処理と 4K フレーム バッファーの処理に重要な役割を果たします。豊富な DSP 機能を備えた Zynq SoC のプログラマブル ロジック ファブリックは、柔軟性の高いプラットフォームを DSP 設計者に提供し、設計者はこのプラットフォーム上に信号処理アルゴリズムをインプリメントできます。同時に、プロセッサとプログラマブル ロジックの緊密な結合により、両方のドメインにまたがるコーデック アルゴリズムの開発が可能です。Zynq SoC ベースのデザインでは、複数の ASSP を必要とする機能を 1 つのデバイスに統合できるため、電力とコストの削減も可能となります。

第 2 に、ザイリンクスは、FPGA および SoC 上に搭載される各種の内蔵トランシーバーと、広範囲にわたるインハウスのコネクティビティ IP コアにより、幅広いコネクティビティをサポートし、4K ビデオ システムの開発をサポートします。たとえば、Zynq 7045 SoC は最大 16 個の 12.5Gbps トランシーバーを搭載し、12G-SDI、6Gbps HDMI 2.0、5.4Gbps DisplayPort 1.2、10Gbps イーサネットの各規格と組み合わせて使用できます。

第 3 に、ザイリンクスは、Vivado® Design Suite に含まれる IP Integrator (IPI) ツールによって設計者に貢献します。図 2 に示すように、IPI ツールを使用すれば、プリント回路基板上でチップを接続するのと同じような作業で IP ブロックを接続できます。OmniTek 社の OSVP ブロックと DMA ブロックのように、IP ブロック上のインターフェイスが、ザイリンクスが標準規格として採用した AMBA AXI4 インターコネクト プロトコルに準拠している場合、この作業は特に簡単に行えます。

ザイリンクスの新しい UltraScale™ (16nm /20nm) テクノロジの登場とともに、さらに大きな処理能力が利用可能になります。UltraScale テクノロジは、最大で 1 秒当たり数百ギガビットのクロック速度をサポートし、「ASIC クラス」と呼ばれています (詳細は、http://japan.xilinx.com/products/

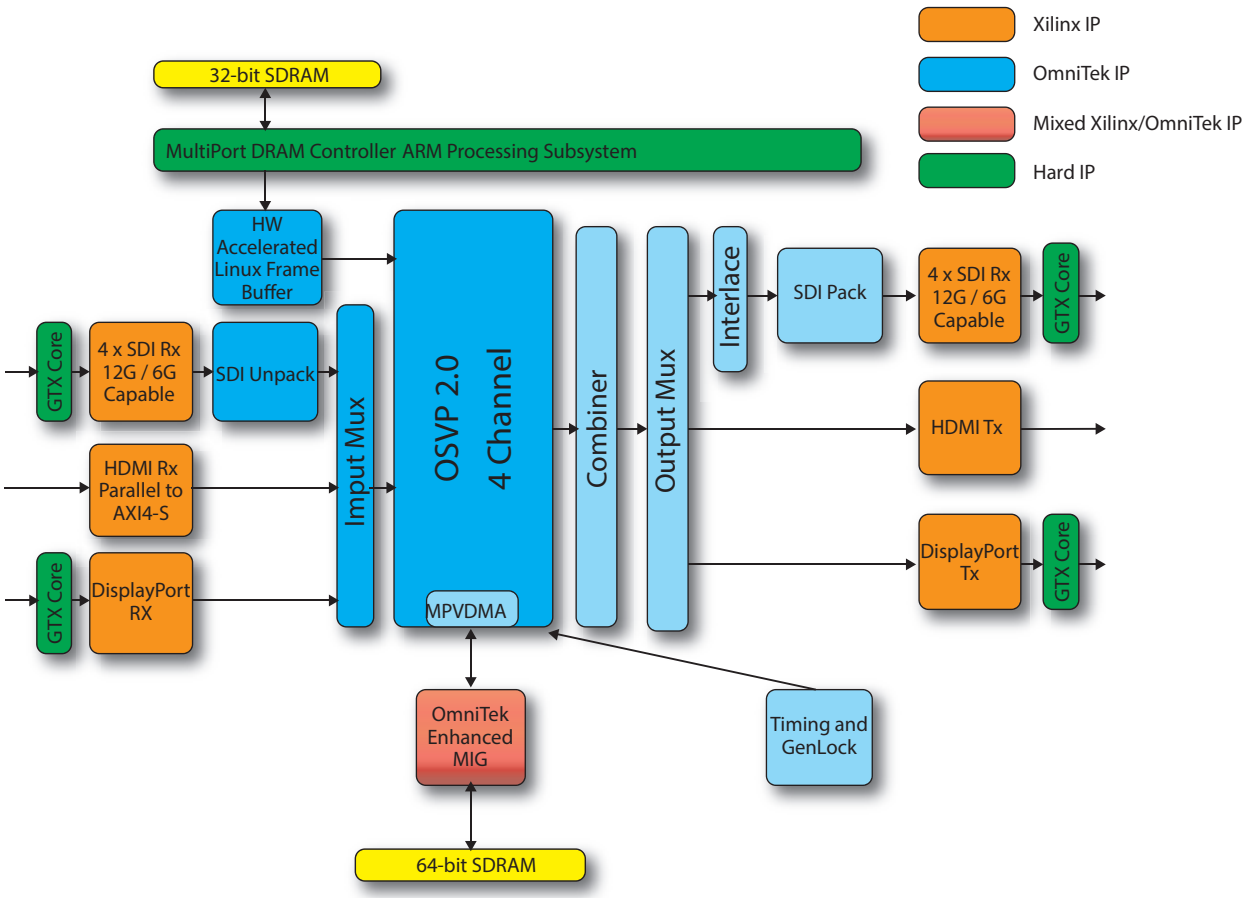


図 3 – RTVE 3.1 リファレンス デザインの構造の概要

図 4 – OmniTek 社の新しい Ultra 4K Tool Box
(搭載されているコンポーネントが見えるように開いた状態)

technology/ultrascale.html を参照)。UltraScale アーキテクチャにより、4K ビデオ システムの開発にとどまらず、将来的には 8K システムの開発も視野に入ってきます。

既製のシステムの応用

IP コアのビルディング ブロックを使用すると、ビデオ システム デザインの開発作業を大幅に容易化できますが、既製のシステムを出発点としてアプリケーションに適応させれば、さらに効率的に開発を進められます。しばらく前から、ザイリンクスは、ザイリンクスのビデオおよびコネクティビティ IP コアと OmniTek 社の IP ブロックの両方を組み込んだ、リアルタイム ビデオ エンジン (RTVE) のリファレンス デザインを提供しています。これらのリファレンス デザインは、幅広いビデオ アプリケーションをターゲットにし、高度に実証が可能なブロードキャスト品質のビデオ処理機能を備えています。RTVE デザインは、これらの IP ブロックの機能と容易な相互運用性 (IP ブロックはすべて AXI4 インターコネクト規格に準拠して設計されている) の両方を実証しています。

RTVE は、バージョン更新のたびに、最新の IP ブロックを組み込むことでデザインの機能を拡張してきました。最新バージョンの RTVE 3.1 で は、SMPTE 425-5:2014、DisplayPort 1.2、6G-SDI および 12G-SDI で定義された 4K ビデオ規格のサポートが追加されました。図 3 に、このデザインのブロック図を示します。

RTVE 3.1 デザインには、上記の 2 種類の OmniTek 社製コアとともに、OmniTek 社のインターレーサー、コンバイナー、専用クロスポイントと、ザイリンクスの主要なコンポーネントがいくつか組み込まれています。また、このデザインには、Web ベースのインターフェイスから RTVE エンジンを駆動する API およびアプリケーションが付属します。RTVE 3.1 デザインのファームウェアと RTVE 3.1 アプリケーションのソフトウェアは、いずれもソース形式で顧客に提供されるため、これらのツールを使用したシステム デザインの進め方の実例を学ぶことも、これを出発点として同じようなシステムを開発することも可能です。

また、RTVE 3.1 用のハードウェア プラットフォームも利用可能です。このプラットフォームは、(ザイリンクス Zynq 7045 SoC をベースとする) OmniTek 社の OZ745 開発キットと、FMC 拡張カードで構成されます。FMC カードは、DisplayPort 1.2 互換の入力および出力ポートと、2 つの SD/HD/3G/6G-SDI 入力および出力を追加します。これらの I/O を組み合わせて、サポートされるビデオ規格を拡張し、6G 4K および 12G 4K、3G Level A および 3G Level B スクエア ディビジョン / クワッド 4K、3G Level A および 3G Level B 2 サンプル インターリーブ 4K に対応させることが可能です。

これらのコンポーネントを組み合わせて商業的に実現可能なシステムを開発できることは、OmniTek 社の Ultra 4K Tool Box (図 4) で証明されています。この 4K Tool Box の基本アーキテクチャは、OmniTek 社の OZ745 開発キット、FMC カード、RTVE 3.1 のファームウェアおよび関連アプリケーション ソフトウェアで構成されます。4K Tool Box は、最大 4K60 までのすべてのビデオ規格のアップ / ダウン / クロス コンバージョンおよび関連する画像補正だけでなく、アイおよびジッター、ガモットの各ビュー、4K 画像を構成する多数のデータ ストリームすべてのピクセル データなどに対する各種の表示機能を提供します。

Ultra 4K Tool Box は最近発売された製品ですが、チップセット製造からテストおよび測定、放送に至るまで、4K 処理の全領域にわたる幅広い顧客が既に購入しています。このような急速な普及は、新しい 4K 規格に対するビデオ業界全体の本格的な関心の現われと言えます。

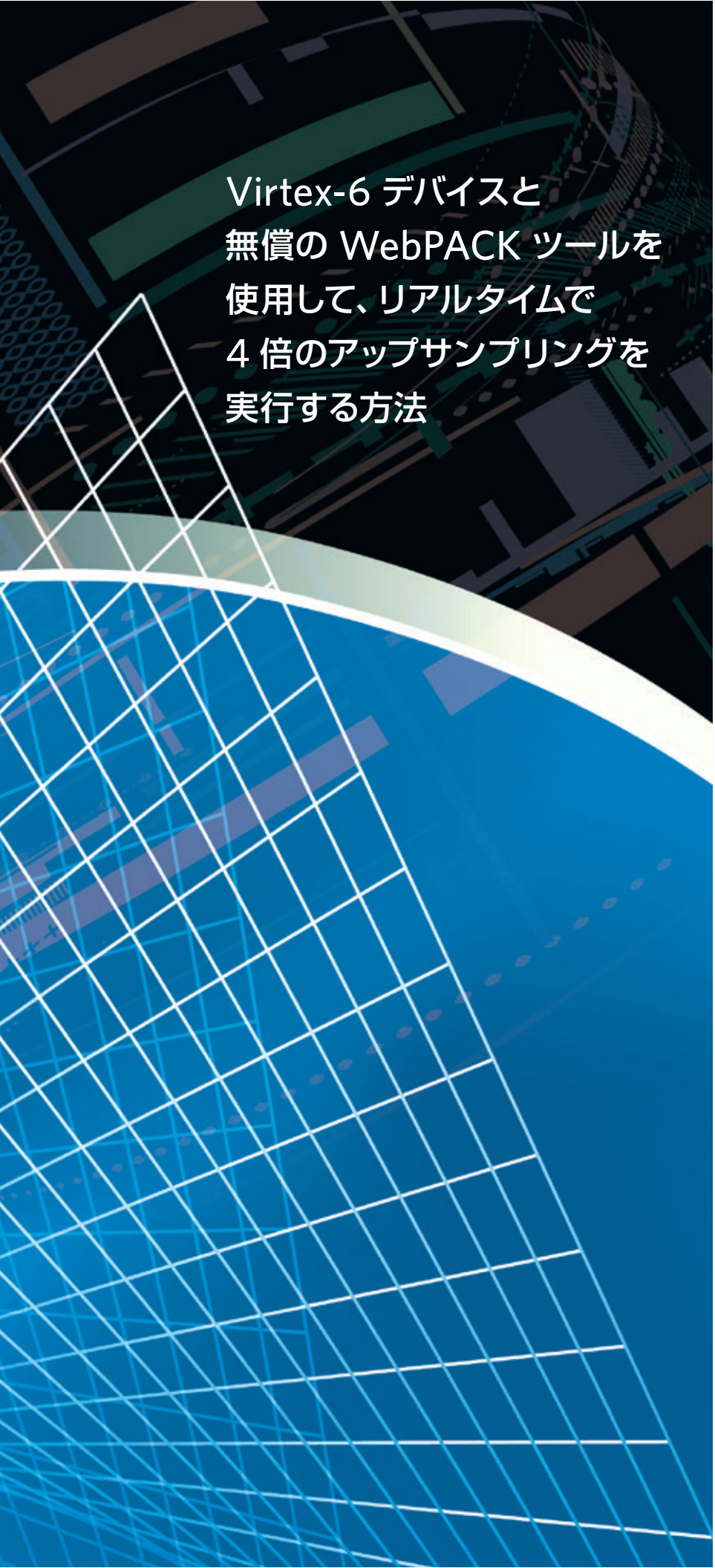
最高レベルのサポートを実現

OmniTek 社は、これらのツールおよび IP コアとともに、顧客の 4K デザインの稼働をお手伝いするコンサルティング サービスを提供しています。ザイリンクスの最先端のシリコン テクノロジおよびソフトウェア ツールと、OmniTek 社のビデオ処理および製造分野の専門知識の組み合わせにより、ビデオ システムの設計者は、完全な開発フレームワークを最初から活用し、予想をはるかに超えた簡単な統合機能とサポートを利用できます。その結果、競争力の高い革新的な製品を、非常に短期間にて市場導入できます。

Efficient Parallel Real-Time Upsampling with Xilinx FPGAs

ザイリンクス FPGA を利用した効率的な並列リアルタイムアップサンプリング

William D. Richard
Associate Professor
Washington University, St. Louis
wdr@wustl.edu



Virtex-6 デバイスと無償の WebPACK ツールを使用して、リアルタイムで 4 倍のアップサンプリングを実行する方法

アップサンプリングは、多くの信号処理アプリケーションで必要とされる処理です。データのベクターを M 倍にアップサンプリングする場合、最も簡単な方法は、概念的には、データ ベクターの離散フーリエ変換 (DFT) [1] を実際の周波数成分の数の (M-1) 倍のゼロでゼロパディングしてから、ゼロパディングしたベクターをタイム ドメインに戻すことです [1, 2]。しかし、この手法は大量の計算を必要とする上、FPGA 内に効率的にインプリメントするのも簡単ではありません。この記事で説明する効率的な並列リアルタイム アップサンプリング回路は、ADC の 1 クロック当たり M 個のアップサンプリングされた値を生成します (ここで、M は希望するアップサンプリング係数)。筆者らのザイリンクス Virtex®-6 XC6VLX75T FPGA のインプリメンテーションは、係数 M=4 でアップサンプリングを実行するもので、より一般的な手法の一例として役立ちます。

筆者らの並列アップサンプリング手法の基盤となる一般的な概念は、「windowed Sinc 補間法」とも呼ばれ、文献 [3, 4] などの優れた論文で解説されています。

説明のために、図 1 に示す 16MHz アナログ信号の例を考えます。この信号の形式は次のとおりです。

$$f(t) = \cos(2 \pi f t) * e^{(t * t)/\text{constant}}$$

式 1

フルスケール入力範囲の 97.7% まで駆動される 12 ビット ADC を使用して、図 1 に示す信号を 80MHz でサンプリング / 量子化した場合、1 信号周期当たり 5 つのサンプルが収集され、図 2 に示すサンプル データ シーケンスが得られます。この例のデータ シーケンスを 4 倍に (320MHz の実効サンプル レートまで) アップサンプリングすると、1 信号周期当たり 20 サンプルが得られます。ここで説明する手法は、より大きな係数を指定することも可能ですが、説明のために M=4 でアップサンプリングを実行します。

もちろん、ADC が生成するデータ シーケンス内の実際の各サンプル値の間に (M-1) 個のゼロを挿入することにより、希望する数のサンプルを使用して (明らかに低品質に) アップサンプリングされたデータ ベクターを生成することは可能です。この「ゼロの挿入手順」は、周波数ドメイン内の元の信号のスペクトラムの複製に対応します。得られる「ゼロパディングされた」タイム ドメイン信号にローパス フィルターを適用し、周波数ドメイン内の希望するスペクトラムの「複製」を削除することにより、アップサンプリングされたデータ ベクターを得ることができます。

FIR フィルターのデザイン

周波数ドメイン内の理想的な (ブリック ウォール) ローパス フィルターは、無限範囲の Sinc 関数を使用したタイム ドメイン内のたたみ込みに対応します。したがって、図 3 に示す 31 タップ FIR フィルターの例のトポロジを使用して、ゼロパディングされたタイム ドメイン信号を ADC クロック レートの M 倍で動作する対称性ローパス FIR フィルターを通過させ、希望するたたみ込み演算の近似計算を実行します。この方法で、アップサンプリングされたデータ ベクターをリアルタイムで生成できます。図 3 の R1、R2、…、R31 は、ADC のクロック レートの M 倍でクロッキングされるレジスタを表し、C0、C1、…、C15 は、FIR フィルターの係数を表します。

図 3 に示す FIR フィルターのレジスタの大半は、どの特定のクロック周期でも実際のサンプル データではなくゼロを保持することに注意してください。たとえば、M=4 の場合、R1 が実際のサンプル データを保持するときは、R2、R3、R4 はゼロを保持します。R1 が実際のサンプル データを保持するときは、R5、R9、R13、R17、R21、R25、R29 もサンプル データを保持し、その他のレジスタはゼロを保持します。次のクロック周期では、R2、R6、R10、R14、R18、R22、R26、R30 が実際のサンプル データを保持します。

図 3 に示す FIR フィルター内を移動していく M 個のサンプルのうち (M-1) 個はゼロになるため、31 タップ FIR フィルター使用時に M=4 の場合、図 4 に示すようにフィルターを折り畳んで、M 個の出力をパラレルに生成できます。このインプリメンテーションでは、パラレル FIR フィルターは、ADC のクロック レートの M 倍ではなく、ADC のベース クロック レートで動作します。

図 4 に示す windowed Sinc 関数係数 Cw(n) を指定して、FIR フィルターのインプリメンテーションに必要な乗算器の数を最小限

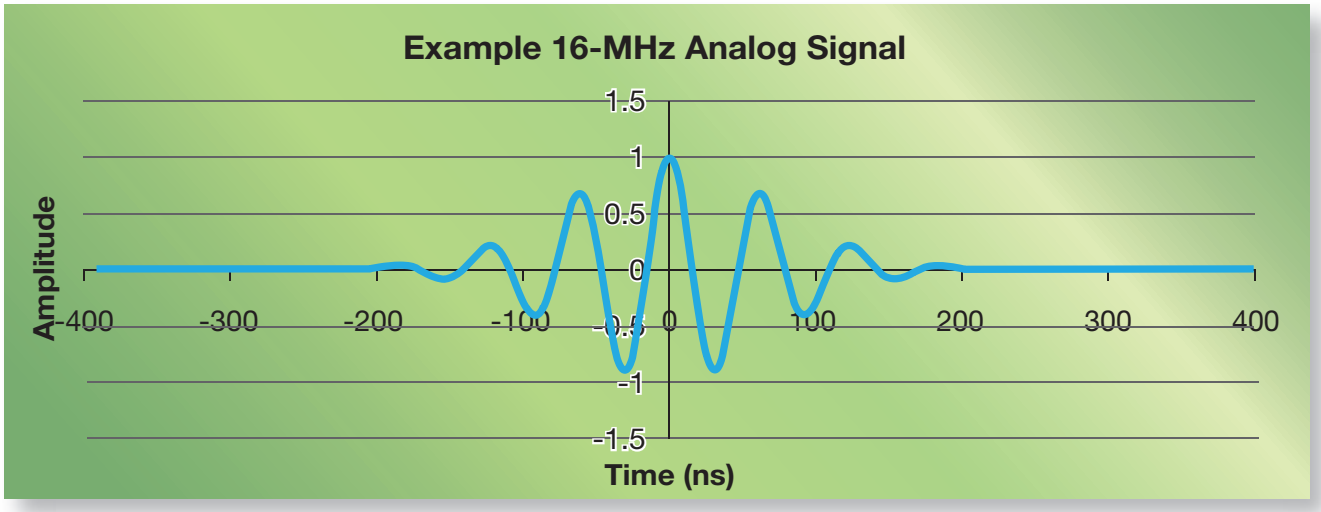


図 1 – この例の 16MHz 信号はアップサンプリング プロセスを示します。

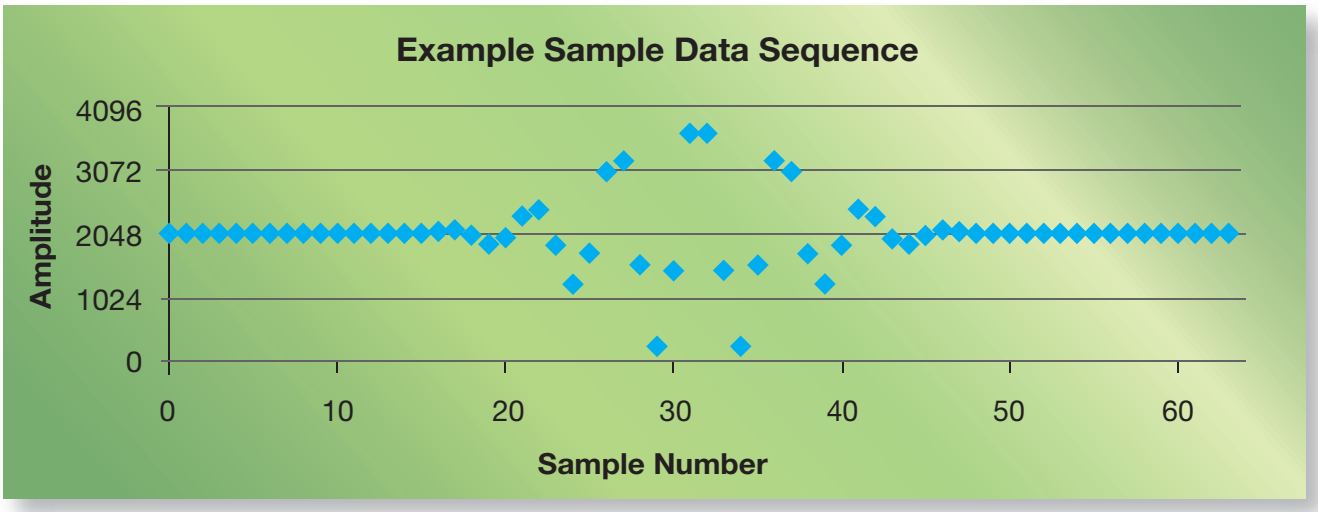


図 2 – フルスケール入力範囲の 97.7% まで駆動される 12 ビット ADC を使用して、
図 1 の例のアナログ信号を 80MHz で (1 周期当たり 5 回) サンプリングして得られるサンプル データ シーケンスの例

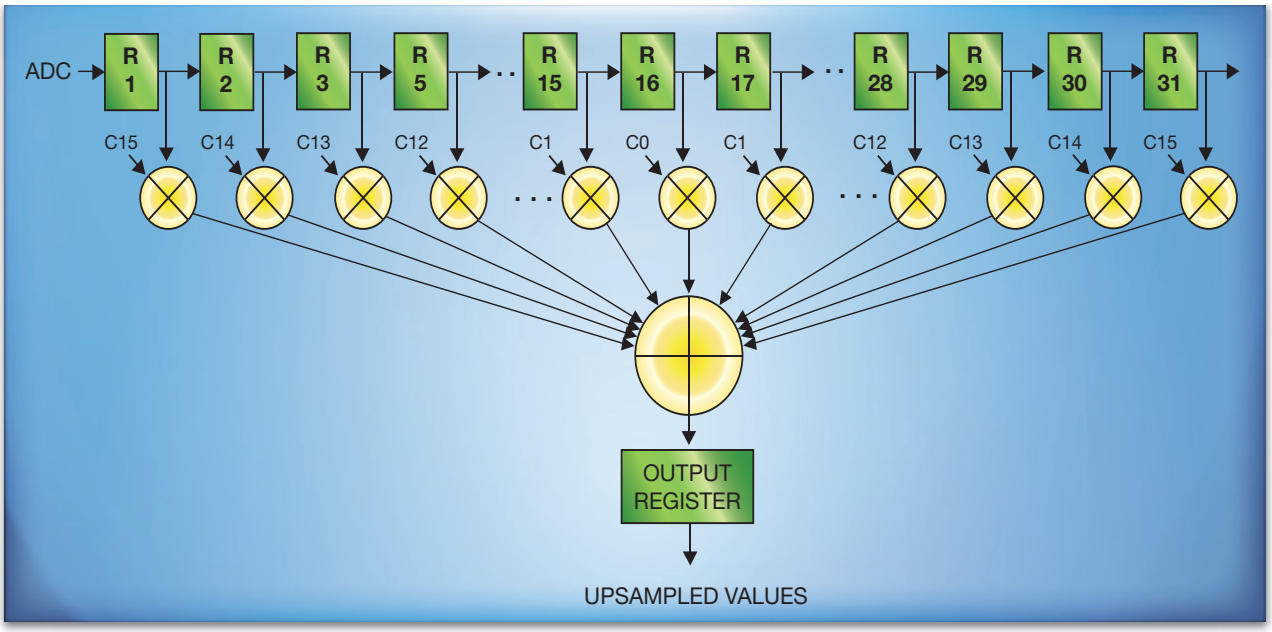


図 3 – 31 タップ FIR フィルターを使用して、1 クロック周期当たり 1 つのアップサンプリングされたデータ値を生成できます (ADC のベース クロック レートの M 倍のクロッキングとゼロの挿入を使用した場合)。

に抑えることができます。T タップのローパス FIR フィルターの場合、最適な係数は次の式で与えられます。

$$C(n) = Sinc[(n * \pi) / M], n = 0 \text{ to } (T-1)/2.$$

式 2

ここで、ハニング窓係数は次の式で与えられます。

$$H(n) = [1-COS(2*pi*(n + ((T-1)/2))/(T-1))]/2, \quad n = 0 \text{ to } (T-1)/2.$$

式 3

windowed Sinc 関数係数 Cw(n) は、C(n) と H(n) の対応する値を乗算することによって求められます。

$$C_w(n) = C(n) * H(n), n = 0 \text{ to } (T-1)/2.$$

式 4

M=4 の場合、31 タップの FIR フィルターの係数が上記のように計算されると (C0 = 1.0、C4 = C8 = C12 = C15 = 0)、図 4 でこれらの係数に関連付けられる 9 個の乗算器は不要になります。さらに、各係数を 2 回ずつ使用して UPSAMPLED VALUE(1) が生成されていることに着目すれば、このインプリメンテーションを「折り畳んで」(たとえば、乗算の前に R1 ～ R8 を加算する)、さらに 4 個の乗算器を削除できます。最終的なデザインでは、1 クロック周期当たり 4 個のアップサンプリングされた値を生成するのに必要な乗算器の数は、合計 18 個で済みます。上記のフィルター設計手法では、元の各サンプル値は変更されずにパラレル フィルターを通過することに注意してください。

筆者らは、図 5 の合成可能な VHDL [5] モデルを使用して、図 4 に示した回路の性能を評価しました。この VHDL インプリメンテーションは、Analog Devices 社の AD9670 8 チャネル超音波フロントエンド集積回路 [6] が生成する、12 ビット サンプル データを想定しています。フィルター係数は、FPGA ダイ上に集積された乗算器のサイズに合わせて、25 ビット固定小数点定数として表現されます。ADC からの入力サンプルは入力ピンに接続されたレジスタ (図 4 の R1) にクロック入力され、アップサンプリングされた出力値は出力ピンに接続されたレジスタを使用します。レジスタ R2 ～ R8 はチップに内蔵されています。合成されたロジックに計算を実行する余裕を与えるために、レジスタ R1 ～ R8 は意図的に 15 ビット幅になっています。このデザインは、オーバーフローやアンダーフローがないかをチェックして、有効範囲内に収まるように結果を制限します。

パイプライン化は不要

図 6 は、無償のザイリンクス WebPACK ™ ツール バージョン 14.7 の ISim シミュレータを使用して VHDL モデルのシミュレーションを行い [8]、図 2 のサンプリング / 量子化された 12 ビット データ シーケンスを供給したときに得られる、アップサンプリングされたデータ シーケンスをプロットしたものです。上記のように、元の各 12 ビット サンプルは変化せず、実際の各サンプル間で元の波形上に 3 つの新しいサンプルが挿入されています。

計算された (アップサンプリングされた) 値と元のアナログ信号の理想値の間のワースト ケースの誤差はフルスケール範囲の 0.464% で、平均誤差はフルスケール範囲の 0.070% です。もちろん、ソースとなるサンプリング / 量子化された 12 ビットのベクター データ値には、初期量子化ステップによる約 ½ LSB (すなわち、フルスケール範囲の 0.012%) の誤差が含まれています。

筆者らは、WebPACK ツール バージョン 14.7 を使用して、ザイリンクス XC6VLX75T-3FF484 Virtex-6 FPGA [7] 内にアップサンプラーをインプリメントしました。配置配線済みのデザインは、FPGA 内の 288 個の DSP48E1 ブロックのうち 19 個を使用しましたが、これは全スライスの 1% 未満です。最終的なアップサンプリング回路は、107MHz で動作しました。この性能を達成するために、フィルターのパイプライン化は不要でした。筆者らは、217MHz 以上で動作するパイプライン版の回路も開発しました。

XC6VLX75T-3FF484 は、ザイリンクス Virtex-6 ファミリで最も小型の製品ですが、ダイ上に集積された 25x18 ビット乗算器付き DSP48E1 ブロック 288 個を搭載しており、理論的には図 4 に示したタイプのパラレル アップサンプリング FIR フィルターを 15 個インプリメントするのに十分なリソースを提供します。筆者らは、80MHz で動作するアップサンプラーの 8 つのコピーを使用するプロトタイプ環状アレイ超音波システムを XC6VLX75T FPGA 内に構築し、ビーム フォーミングの前に、Analog Devices 社の AD9670 8 チャネル 超音波フロントエンド チップからデータをアップサンプリングしました。このシステム内のアップサンプラーは、シミュレーションで予測したとおりに機能し、AD9670 ADC のベース クロック レートの 80MHz で動作しながら、320MHz にアップサンプリングされたデータを使用して、リアルタイムのビーム フォーミングを実現します。

ザイリンクス Virtex-6 FPGA ファミリで最も大型の XC6VSX-475T は、2,016 個の 25x18 ビット乗算器を搭載し、理論的には図 4 に示したタイプのアップサンプリング フィルターをワンチップに 106 個インプリメントできます。

FIR フィルターがこの記事で説明した効率的なパラレル トポロジを利用して設計されている場合、ザイリンクス XC6VLX75T-3FF484 FPGA にインプリメントされた 107MHz で動作する FIR フィルターを使用して、M=4 の係数でリアルタイムにアップサンプリングすることが可能です。元のデータ サンプルは変更されずにフィルターを通過し、(M-1) = 3 個のアップサンプリングされた値がパラレルに生成されます。この簡単な FIR フィルター設計手法により、高度なフィルター設計ツールは不要になり、優れた

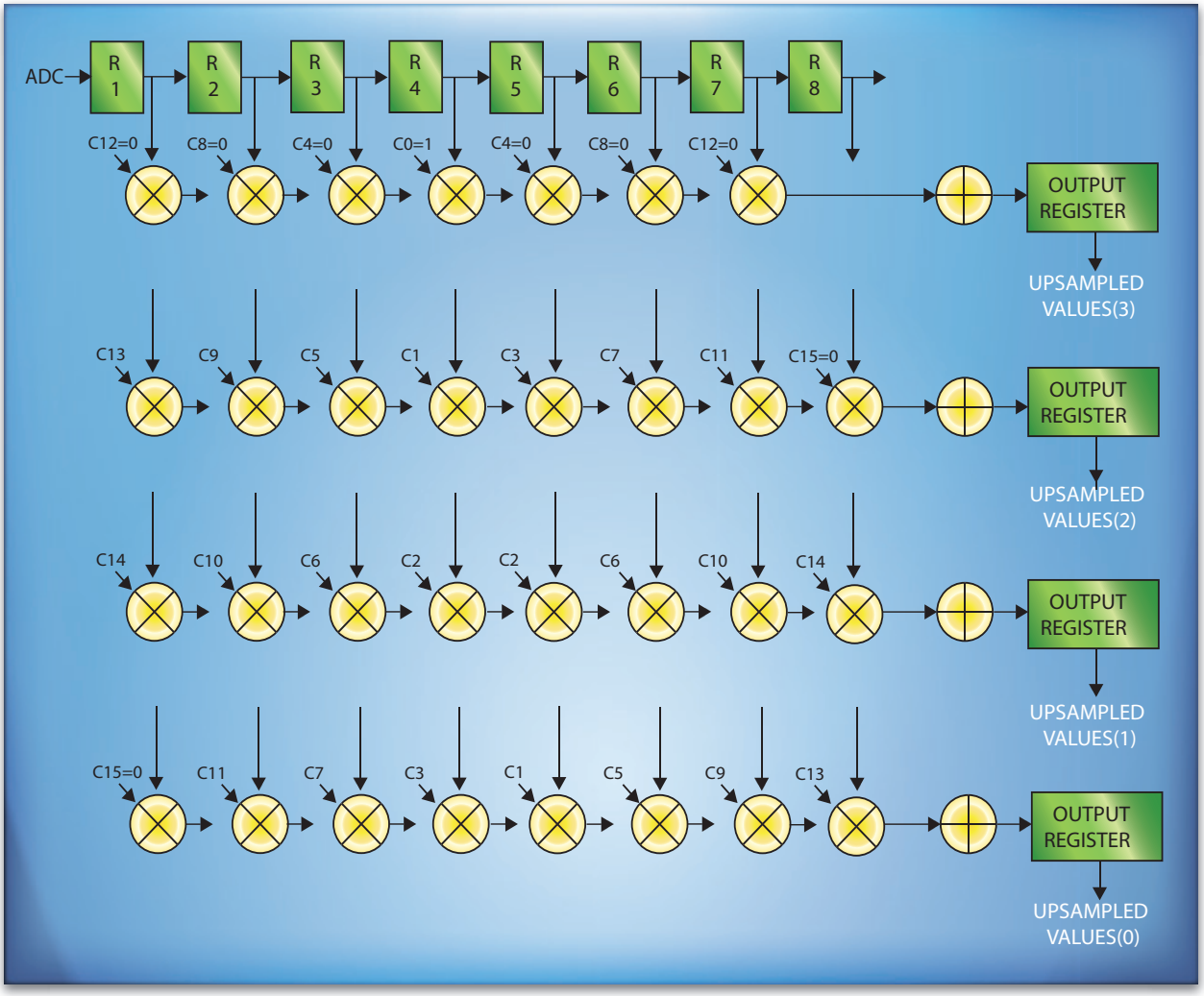


図 4 – 図 3 で、どの特定のクロック周期でも、ゼロでないデータを保持するのはレジスタ 4 つ毎に 1 つだけであることに着目すれば、フィルターが ADC のベース クロック レートで動作している場合、フィルターを折り畳んで 4 つの出力をパラレルに生成することが可能です。

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_ARITH.ALL ;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;

ENTITY upsample IS
   PORT (clk         : IN STD_LOGIC ;
         r_ext       : IN STD_LOGIC_VECTOR(11 DOWNTO 0) ;
         d0,d1,d2,d3 : OUT STD_LOGIC_VECTOR(11 DOWNTO 0)) ;
   END upsample ;

ARCHITECTURE mine OF upsample IS
   SIGNAL   r1,r2,r3,r4,r5,r6,r7,r8             : STD_LOGIC_VECTOR(14 DOWNTO 0) ;
   SIGNAL   d0int,d1int,d2int                   : STD_LOGIC_VECTOR(39 DOWNTO 0) ;
   CONSTANT c1  : STD_LOGIC_VECTOR(24 DOWNTO 0) := "1110001111110100111001100110" ;
   CONSTANT c2  : STD_LOGIC_VECTOR(24 DOWNTO 0) := "1001101111101110000000000011" ;
   CONSTANT c3  : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0100010101111111011001111001" ;
   CONSTANT c5  : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0010001010010010011110000" ;
   CONSTANT c6  : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0010001110001110010111001" ;
   CONSTANT c7  : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0001001000101111000010111" ;
   CONSTANT c9  : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0000100011011001000000101" ;
   CONSTANT c10 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0000100000100110000100111" ;
   CONSTANT c11 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0000001101110111011000010" ;
   CONSTANT c13 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0000000011000100001100100" ;
   CONSTANT c14 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0000000001000001000111111" ;

BEGIN

   flops:PROCESS(clk)
   BEGIN
      IF (clk = '1' AND clk'EVENT) THEN
         r1 <= "000" & r_ext ;
         r2 <= r1 ;
         r3 <= r2 ;
         r4 <= r3 ;
         r5 <= r4 ;
         r6 <= r5 ;
         r7 <= r6 ;
         r8 <= r7 ;
         IF d0int(39) = '1' THEN
            d0 <= "000000000000" ;
         ELSIF d0int(38) = '1' OR d0int(37) = '1' THEN
            d0 <= "111111111111" ;
         ELSE
            d0 <= d0int(36 DOWNTO 25) ;
         END IF ;
         IF d1int(39) = '1' THEN
            d1 <= "000000000000" ;
         ELSIF d1int(38) = '1' OR d1int(37) = '1' THEN
            d1 <= "111111111111" ;
         ELSE
            d1 <= d1int(36 DOWNTO 25) ;
         END IF ;
         IF d2int(39) = '1' THEN
            d2 <= "000000000000" ;
         ELSIF d2int(38) = '1' OR d2int(37) = '1' THEN
            d2 <= "111111111111" ;
         ELSE
            d2 <= d2int(36 DOWNTO 25) ;
         END IF ;
         d3 <= r4(11 DOWNTO 0) ;
      END IF ;
   END PROCESS ;

   d0int <= r2*c11 - r3*c7 + r4*c3 + r5*c1 - r6*c5 + r7*c9 - r8*c13 ;
   d1int <= (r2+r7)*c10 - (r1+r8)*c14 - (r3+r6)*c6 + (r4+r5)*c2 ;
   d2int <= r2*c9 - r1*c13 - r3*c5 + r4*c1 + r5*c3 - r6*c7 + r7*c11 ;

END mine ;
```

図 5 – この VHDL ソースは、1 つのプロセスと 25 ビット固定小数点係数を使用して図 4 のフィルター トポロジをインプリメントします。

結果が得られます。この記事で説明した考え方の延長線上で、さらにタップ数の多い FIR フィルターを使用すれば、より大きな倍率でのアップサンプリングや、算出されるアップサンプリング値の誤差の縮小が可能となります。

参考資料

1. A.V. Oppenheim, R.W. Schafer, *Discrete-Time Signal Processing* (Prentice Hall, Englewood Cliffs, NJ, 1989)
2. H. Stark, J.W. Woods, I. Paul, "An investigation of computerized tomography by direct Fourier inversion and optimum interpolation," IEEE Transactions Biomedical Engineering 28, 496-505 (1981)
3. R.W. Schafer, L.R. Rabiner, "A digital signal processing approach to interpolation," Proceedings of the IEEE 61, 692-702 (1973)
4. R. Crochiere, L.R. Rabiner, *Multirate Digital Signal Processing*, (Prentice-Hall, Englewood Cliffs, NJ, 1983)
5. D. Pellerin, D. Taylor, *VHDL Made Easy!* (Prentice-Hall, Upper Saddle River, NJ, 1997)
6. Analog Devices AD9670 Octal Ultrasound AFE with Digital Demodulator Datasheet Rev Sp0 (Analog Devices, 2013)
7. 『Virtex-6 ファミリ概要』 (DS150) (v2.3) (Xilinx, Inc., 2011)
8. 『ISE アドバンス チュートリアル』 (UG695) (v13.1) (Xilinx, Inc., 2011)

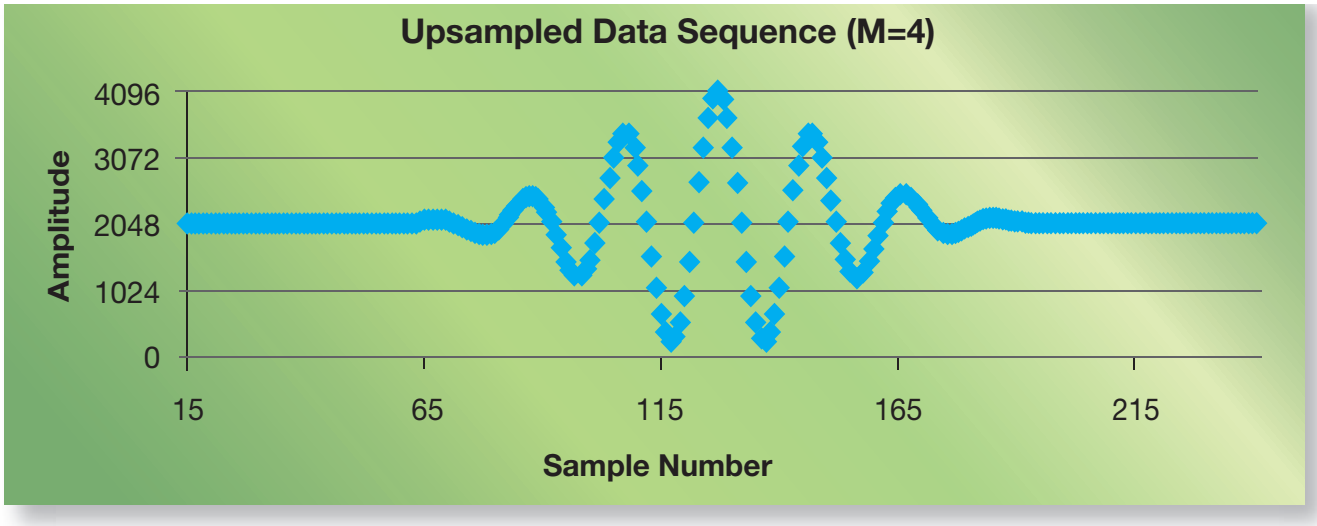


図 6 – このグラフは、VHDL モデルによって生成されたアップサンプリング後のデータ シーケンスを示しています。

XILINX® ウェブセミナー Web Seminar

All Programmable FPGA、SoC、3D IC の世界的なリーディング プロバイダーの
ザイリンクスが提供するプログラマブル ロジックからプログラマブル システム
インテグレーションのさまざまな機能と活用方法をご紹介します。
コストを抑え、最大のパフォーマンスを実現するための最新情報を手に入れてください。

ニーズに合わせたプログラムを各種取り揃えて好評配信中!!

New!! 新セミナー登場
All Programmableで実現する ハイエンド組込みヒューマン マシン インターフェイス

FPGA入門編

FPGA の基本を理解したい方へ FPGA の全体概要を解説した入門編と、ものづくりにチャレンジする経営者、技術管理者の方へ FPGA を採用する利点をご説明します。

- ▶ 30分で判る! FPGA入門
- ▶ 15分で判る! FPGA採用理由

FPGA/SoC 活用編

ザイリンクス FPGA/SoC を使った最先端デザインの設計手法や、さまざまなアプリケーション設計に求められるデザイン チャレンジに対するソリューションをご紹介・解説します。

- ▶ ザイリンクス All Programmable ソリューションで実現する機能安全
- ▶ Zynq SoC を使用したマルチチャンネル リアルタイム ビデオ プロセッサの設計
- ▶ Zynq SoC を使用した最先端 エンベデッド システムの設計 ～アクセレータでのソフトウェア ボトルネックの解消方法～
- ▶ 7 シリーズ ターゲット デザイン プラットフォーム

開発ツール編

プログラマブルデバイスである FPGA の設計には開発ツールがキーになります。ザイリンクスが提供するユーザー フレンドリーな開発ツールの特徴や使い方、先端設計メソドロジについて解説します。

- ▶ 次世代FPGA設計手法セミナー PlanAhead デザイン解析ツール ～ 第1部、第2部、第3部、デモ ～
- ▶ AMBA AXI4 テクニカルセミナー

FPGA/SoC 概要編

FPGA の世界トップシェアを誇るザイリンクスが提案するソリューションや、ザイリンクスの最先端 FPGA の詳細を解説します。

- ▶ UltraScale アーキテクチャ概要
- ▶ Zynq-7000 SoC アーキテクチャとエコシステム
- ▶ 28nm ザイリンクス 7 シリーズ FPGA のアジャイル ミックスド シグナル テクノロジ

セミナー内容の詳細／ご視聴は今すぐこちらから ▶▶▶ http://japan.xilinx.com/webseminar/

A Framework for FPGA Design Planning

FPGA デザイン プランニング用のフレームワーク

Jeffrey Lin
Senior Manager, Global Communications Services Group
Xilinx, Inc.
jeffrey.lin@xilinx.com

この実績のある FPGA 開発フレームワークを利用すれば、プロジェクトのスムーズな運営が可能になります。

FPGA は現在システムの中心的位置に据えられ、多くの製品で主なデータ処理エンジンとして機能しています。FPGA の機能と性能がこのレベルに達してから、既に長い時間がたっています。

多くのアプリケーションで FPGA が果たしている重要な役割を考えると、定式化された体系的な開発プロセスで FPGA デザインに取り組むことは、以前にも増して重要になっています。開発サイクルの後半に入ってから設計上の欠陥が見つかると、問題の解決に大きなコストと時間がかかり、スケジュール、コスト、品質に致命的な影響を与えるおそれがあります。この開発プロセスの目的は、このような事態を回避することです。

ザイリンクスのグローバル通信サービス グループは、長年にわたり、実績のあるデザイン フレームワークを使用して、医療画像処理エンジンから自己学習型ネットワーク スイッチ エンジンまで幅広い製品のターンキー FPGA デザインを開発し、顧客に提供してきました。このフレームワークは、数百種類の FPGA デザインの設計、開発、提供の過程で開発され、進化してきたものです。

筆者らが使用しているフレームワークは、システム アーキテクチャの考慮事項から FPGA の開発とテストのプランニングまで、あらゆる内容を対象とします。この記事では、FPGA のハードウェアに焦点を合わせて、このフレームワークについて詳しく説明します。そしてこのフレームワークは、さまざまなエンジニアリング チームが複雑な FPGA 設計プロジェクトに取り組む際に有益なツールとなることでしょう。

フレームワークの概要

このフレームワークは、FPGA ハードウェアを設計するための反復的トップダウン手法です。最初に、システム アーキテクチャ レベルからプランニングを始めて、FPGA の機能を決定します。次に、FPGA デバイスの既知の機能と性能を利用して、FPGA にインプリメントされる機能を段階的に改善していきます。

また、大規模な FPGA デザインをインプリメントするには、明確に定義された開発プラン、シミュレーション プラン、検証プランが必要です。このフレームワークは、これらのプランを作成する際にも役立ちます。このフレームワークは、図 1 に示したフローチャートにまとめられます。この議論では、プランニングと文書化の部分（最上部）に焦点を合わせます。

システム アーキテクチャ

この議論の文脈では、システム アーキテクチャとは、システム ソフトウェアおよびハードウェア内での機能の分割を指します。製品レベルの要件は既に定義されているものとして（たとえば、マーケティング部門や製品定義部門が既に製品の要件に同意しているとして）、特に FPGA とほかのマイクロチップ コンポーネントへのハードウェア機能の分担に焦点を合わせます。

このシステム アーキテクチャ フェーズ（段階）の目的は、製品の要件を物理的な製品内でどのように実現するかを明確に定義しておくことです。FPGA の場合、主に決定しなければならないのは、どのような機能とファンクションをプログラマブル デバイスにインプリメントするか、またさらに、FPGA 内で何が十分に達成可能であるかということです。

FPGA の上位レベル要件を先に定義しておけば、開発プロセス後半でのコストのかかる設計変更や要件変更を回避できます。この初期段階では、システム アーキテクチャの定義に従えば、開発期間と製品コストの両方に影響を与えるいくつかの重要な決定に導かれます。

このレベルの議論に必要なのは、FPGA の機能の概要だけです。詳細な機能とインプリメンテーション要件は、FPGA 要件の定義の後半の段階で定義されます。この議論には、システムレベルの要件についてよく理解している人、システムレベルのアーキテクチャのデザインに関する知識のある人、FPGA の機能と能力についてよく理解している人が参加します。

FPGA に関連して、次の 10 個の重要な問いに答える必要があります。

1. どのような機能リストを FPGA にインプリメントするか。
2. FPGA への機能のインプリメントと非FPGA コンポーネントの使用の間には、どのような技術的トレードオフがあるか。
3. FPGA へのインプリメンテーションと非 FPGA コンポーネントのインプリメンテーションには、どの程度の設計労力/コストがかかるか。
4. どのようなカスタム機能または処理が必要か。
5. FPGA の柔軟性は機能に何をもたらすか。
6. 将来も有効なリスク軽減手段として何を検討するか。
7. 複数の非 FPGA コンポーネントの機能を FPGA に統合できるか。
8. インプリメントするデザインの機能に基づいて、どのような FPGA デバイスの選択肢があるか。
9. その機能は FPGA にインプリメントできるか。
10. どのような非 FPGA デバイスが必要か。非 FPGA デバイスと FPGA とのインターフェイスをどのようにするか。

FPGA アーキテクチャ

FPGA アーキテクチャは、FPGA デバイス上の物理レベルでのマイクロアーキテクチャおよびチップレベルのデータフロー設計です。設計チームは、デバイスのサイズ、選択、

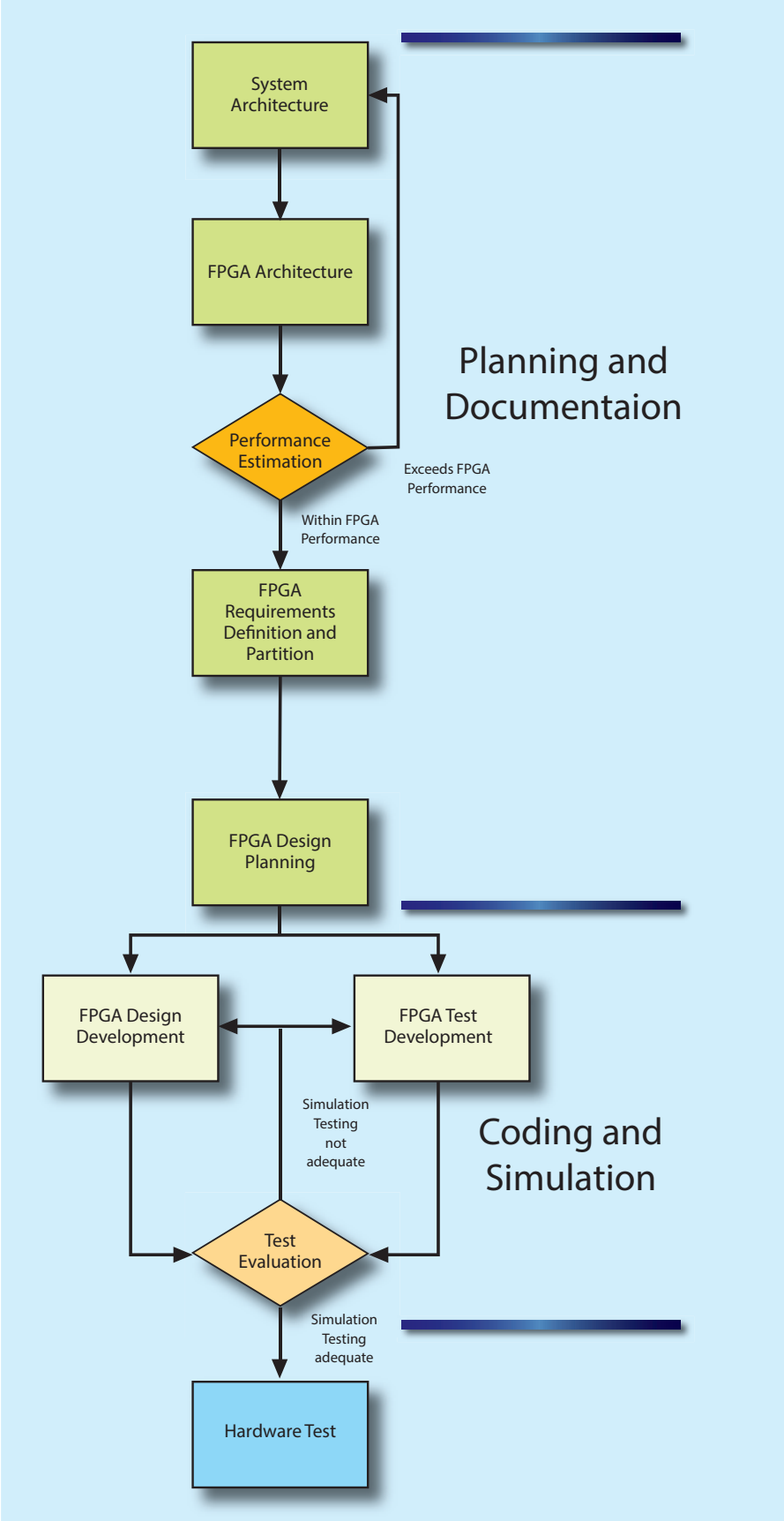


図 1 – FPGA の開発フレームワーク

実現可能性を決めるために、FPGA アーキテクチャを、システムレベルのアーキテクチャと同時に、設計する必要があります。

FPGA アーキテクチャの定義の目的は、システム アーキテクチャの要件を、正確、現実的、かつ達成可能な設計要件として、FPGA 内に確実にインプリメントできるようにすることです。

このレベルの議論には、FPGA のファブリックとリソースの機能と能力に関する詳しい知識が必要です。したがって、経験豊富な FPGA 設計者が議論に参加する必要があります。この段階では、FPGA の性能目標、起こり得るリスクの領域、利用可能な FPGA ファブリック リソースの使用率について検討し、決定する必要があります。

FPGA アーキテクチャの定義中に、システムレベルの要件およびアーキテクチャを FPGA にインプリメントするのは達成不可能または高リスクであることが判明する場合もあります。この場合は、システム アーキテクチャを再評価して更新し、FPGA で達成可能な上位レベルの要件に定義し直す必要があります。

どのような既存の IP コアを使用できるか、何を自分で作成しなければならないかを検討する必要があります。また、I/O の要件や、クロック ドメインとクロック機能を FPGA のクロック リソースにどのようにマッピングするかについても検討する必要があります。その他の主な検討事項として、GT リソースを FPGA にどのように配置するか、SLR の境界にまたがるデータ フローを SSI デバイスで考慮するか、目標クロック周波数はデザインの機能に対して現実的であるか、などが挙げられます。さらに、選択した FPGA に対して目標とする性能が現実的であるかについても評価する必要があります。

FPGA 要件の定義と分割

FPGA 要件の定義と分割のフェーズ (段階) は、システム アーキテクチャおよび FPGA アーキテクチャと密接に関連しており、各段階の決定によって影響を受けます。FPGA 要件の定義とは、FPGA にインプリメントされる詳細な要件のことであり、設計チームとテスト エンジニアリング チームが

設計、インプリメンテーション、達成するべき最終的な機能リストとなります。FPGA 要件の定義は、システム アーキテクチャ要件や FPGA アーキテクチャ要件の定義とは、その緻密さが異なります。この機能リストは、システムの各種コンポーネント間の機能の分担や、FPGA を通るデータ フローではなく、FPGA の機能の詳細な要件を定義します。

この段階の目的は、FPGA エンジニアリング チームが正確にはどのような機能をインプリメントし、テストするかを明確に定義することです。ここでは、上位レベルのシステム アーキテクチャと FPGA アーキテクチャの要件を、厳密なインプリメンテーション要件に変換します。これには 2 つの利点があります。第 1 に、FPGA の要件を個々に定義すると、システム アーキテクチャと FPGA アーキテクチャの制限や、それまで考慮されてこなかったり、予測できなかったりした条件が明確になります。第 2 に、このステップにより、FPGA デザインの開発とテストがスムーズに進められます。

FPGA の要件を正確に記述するには、個々の要件に簡単に変換できる、簡潔で明確な定義が必要です。各要件に名前または番号を付けて、(上位レベルの曖昧な要件としてではなく) 達成可能か否かを簡単に判断できる基本的な記述として定義することをお勧めします。明確で簡潔でさえあれば、業界標準規格も独自の形式も自由に使用できます。

「高速」や「小型」などの厳密に定義されていない曖昧な用語を使用せず、「400MHz」や「4.2k フリップフロップ」のような特定の目標値を記述します。この定義の目的は、システム アーキテクチャや FPGA アーキテクチャに関する予備知識を持たない開発エンジニアリング チームに、いちいち内容を説明せずに文書を配布できるようにすることです。各要件がわかりやすく、簡潔に、そして曖昧さがなく定義されているかどうか、いちいち説明しなくても済むように、すべての必要な情報が含まれているかどうかを確認してください。また、ピン配置と I/O の定義が要件に含まれているか、すべての上位レベル要件が基礎的な設計要素に分解されているか、初期段階でのシステム アーキテクチャの定義に参加していない設計チームがこの要件に基づいて FPGA を開発できるか、テストおよび検証チームがこの文書に従ってテスト プラットフォームとテスト方法を開発し、各要件の最終的な合否を検証できるかを確認してください。

FPGA デザイン プランニング

フレームワークのこの段階は、FPGA ハードウェアの実際の開発作業のプランニングです。全体的な製品開発の中で、FPGA の機能と開発がそれ以外の部分と歩調を合わせて完了するように計画します。

この段階の目標は、システムレベルおよび FPGA レベルの要件とアーキテクチャの現在の状態を、開発プランの中に適切に位置付けることです。既に説明したプランニング段階を完了した後、開発チームは通常 2 つのシナリオに直面します。

シナリオ 1 は、システムおよび FPGA のアーキテクチャと要件が明確に理解され、詳細に記述された結果、FPGA デザイン開発フェーズ (すなわち、HDL コーディング) とテスト開発フェーズ (シミュレーション テストベンチ) が最小限の要件変更でスムーズに進行する場合です。

シナリオ 2 は、システム アーキテクチャと FPGA の要件がまだ流動的な場合です。この場合、開発サイクルのデザインおよびテスト開発フェーズで、複数の設計変更や修正が発生します。

誰もがシナリオ 1 を目指しているにもかかわらず、しばしばシナリオ 2 (明らかに、より管理が難しい状況) に陥ります。

デザイン プランニングの全体的な目標は、開発サイクルのこの段階で、シナリオ 1 に到達することです。シナリオ 1 では、FPGA の開発は簡単であり、デザインの機能のインプリメンテーションとテストのスケジューリングの問題に帰着します。

シナリオ 2 の場合、最も重要な管理タスクは、明確に理解されたプロセスを確立し、どのような変更をインプリメントするか、それぞれの変更が開発スケジュール全体にどのような影響を与えるかを評価し、決定することです。利用できるプログラム管理方針および手法はいくつかありますが、最も重要な点は、このような設計変更の評価とその影響の評価を行うことです。

FPGA 固有のプランニングと開発について言えば、FPGA の利点の 1 つは、ハードウェア プラットフォームを修正してダウンロードし、PCB を何回でも試作できることです。設計チームはこの機能をフルに活用するべきです。したがって、作業用デザインに段階的に機能を追加していく開発プランを推奨します。その趣旨は、一部の要件がインプリメントされていない状態でも主要な通信インターフェイスが機能するような、基本的なデザインから始めることです。

この手法の利点は次の 2 つです。第 1 に、PCB とより大規模なシステムのデバッグに使用できる作業用デザインを常に確保できます。第 2 に、新たに追加された機能をチェックして、現在機能しているデザインに干渉したり、問題を起こしたりしないことを確認できるため、実際の FPGA デザインのデバッグが容易になります。

FPGA デザインの開発と並行して、結果として得られる FPGA デザインに対する適切なシミュレーション環境プランを作成することが非常に重要です。堅牢なシミュレーション環境の開発に投資することにより、デザインのバグが減少するだけでなく、実際のデータ フローを複製してシミュレーションでエラー条件を再現し、根本原因を迅速に切り離して特定できるため、ラボでのデバッグ時間を大幅に短縮できます。

堅牢なテストおよびシミュレーション環境の開発は、FPGA デザインそれ自体の開発と同じくらい複雑であり、周到な計画と配慮が要求されます。

ザイリンクスのグローバル通信サービス グループは、FPGA 開発フレームワークに磨きをかけて、数百種類以上の FPGA デザインに一貫したアプリケーションの形で適用してきました。その結果、優れた結果が得られ、容易に理解され、多様な開発タスクへの幅広い応用を特徴とする、実践的な設計手法が確立されました。このフレームワークを利用して次の FPGA デザインを開発すれば、的確な全体の開発スケジュール、ハードウェアの迅速な開発、最終的には予定どおりの製品発売という形で、確実にメリットをもたらすでしょう。

Faster Design Entry with Vivado IP Integrator and Xilinx IP

Vivado IP インテグレーターとザイリンクス IP を利用した迅速なデザイン エントリ

Duncan Cockburn
Staff Design Engineer
Xilinx, Inc.
duncan.cockburn@xilinx.com

CPRI リモート ラジオ ヘッドのデザインで Vivado IPI と組み合わせて使用される、ザイリンクスの IP コアを最適化する方法を説明します。

今日の FPGA ベースのデザインに使用される IP (知的設計資産) コアは、種類とインスタンス数の両面で増加の一途をたどっています。Vivado® Design Suite の IP インテグレーター (IPI) ツールとザイリンクスの通信 IP を使用して、これらの IP ブロックをすばやく簡単に接続できます。

IPI を利用した手法の優れた効果を示すために、無線リモート ラジオ ヘッド (RRH) の例を考えます。アンテナの近くに設置される RRH は、携帯電話通信網の一部を構成します。RRH は、通常は上流のベースバンド トランシーバー ステーションに光ファイバーで接続されますが、オプションによって下流のほかの RRH にも接続され、マルチホップ トポロジ (図 1) を実現します。

CPRI (Common Public Radio Interface) プロトコルは、これらの RRH の相互接続に広く使用されています。この記事では、1 つのアップリンク CPRI ポートと 3 つのダウンリンク CPRI ポート、およびそれらを接続するためのデザイン例を作成します。この作業の大半は IPI で実行できます。その結果、全体的なデザイン内の主要なコンポーネントが形成されます。ここでは、このアプリケーションに最適な、低消費電力、低コスト、高性能の Kintex®-7 デバイスを使用します。スピード グレードが -2 の All Programmable Kintex FPGA および Zynq®-7000 SoC 内の GTX トランシーバーにより、9.8Gbps の CPRI ライン レートを使用できます。

図 2 は、IPI 内で作成されるデザインを示しています。ブロック デザインを作成し、必要な IP コアを IP カタログからインスタンシエートできます。CPRI コアは標準的なザイリンクス IP カタログ内に用意され、できる限りリソースを共有するように、また IPI 内で使いやすいように最適化されています。スイッチはカスタム IP コアです。

IP コアのリソース共有

ユーザーは、IP コアの複数のインスタンスを使用する場合、リソースをどのように効率的に共有するかという課題に直面します。多くの通信 IP コアは、「共有ロジック」機能をサポートしています。CPRI コアの場合、コア内の共有可能なロジック リソースを使用して IP コアをコンフィギュレーションすることも、これらの共有リソースを省略することもできます。共有リソースがコアに含まれている場合、それらのリソースは、そのロジックを排除したコアへの接続に必要な出力ポートを提供します。

特殊な要件があるユーザーは、すべてのコア上でこのロジックを排除し、独自のロジックをインプリメントできます。筆者らのデザインでは、9.8Gbps で動作するように CPRI コアをコンフィギュレーションしました。このライン レートでは、トランシーバー クロック用に LC タンクベースのオシレーターを使用する必要があります。Kintex-7 デバイス内のトランシーバーは 4 個 1 組で構成され、4 個 1 組のトランシーバーのそれぞれは、4 つのトランシーバー チャネルと 1 つの LC タンクベースのクワッド位相ロック ループ (QPLL) で構成されます。すべてのコアが、QPLL と、アップリンク クロックによって生成されるクロックを共有する必要があります。図 3 は、共有ロジックを使用してカスタマイズされたアップリンク コア上の QPLL とクロック出力ポートが、共有ロジックを排除してカスタマイズされたダウンリンク CPRI コア上の適切な入力ポートに接続される構成を示しています。

CPRI コア間のデータの転送

筆者らは、コア間でデータを転送できるように、IQ スイッチとイーサネット スイッチもインスタンシエートしました。

CPRI ネットワーク内の制御および管理データは、イーサネット サブチャネルを介して送信されます。システム内のイーサネット スイッチにより、遠隔からファームウェア アップデートまたはコマンドを発行し、任意のノードに送信することが可能になります。この状況では全機能装備のイーサネット スイッチは必要ないので、スイッチの IP コアは、できる限り少量のロジック リソースを使用するように設計されています。

IQ スイッチは、CPRI コア間で任意の IQ サンプルを確定的なレイテンシで転送する機能を提供します。マルチホップ無線システムに重要なのは、リンク遅延を正確に測定する

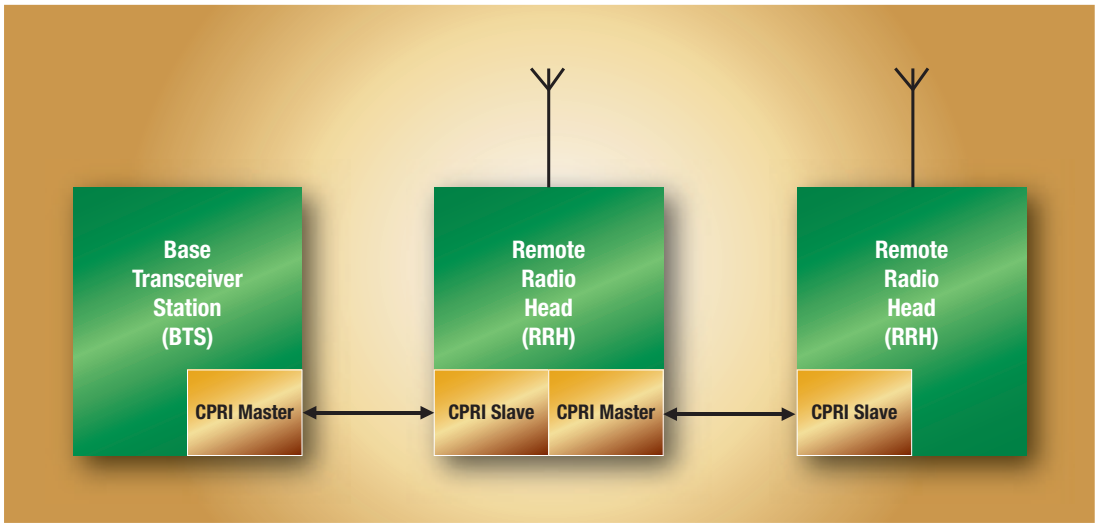


図 1 – マルチホップ トポロジのダイアグラム

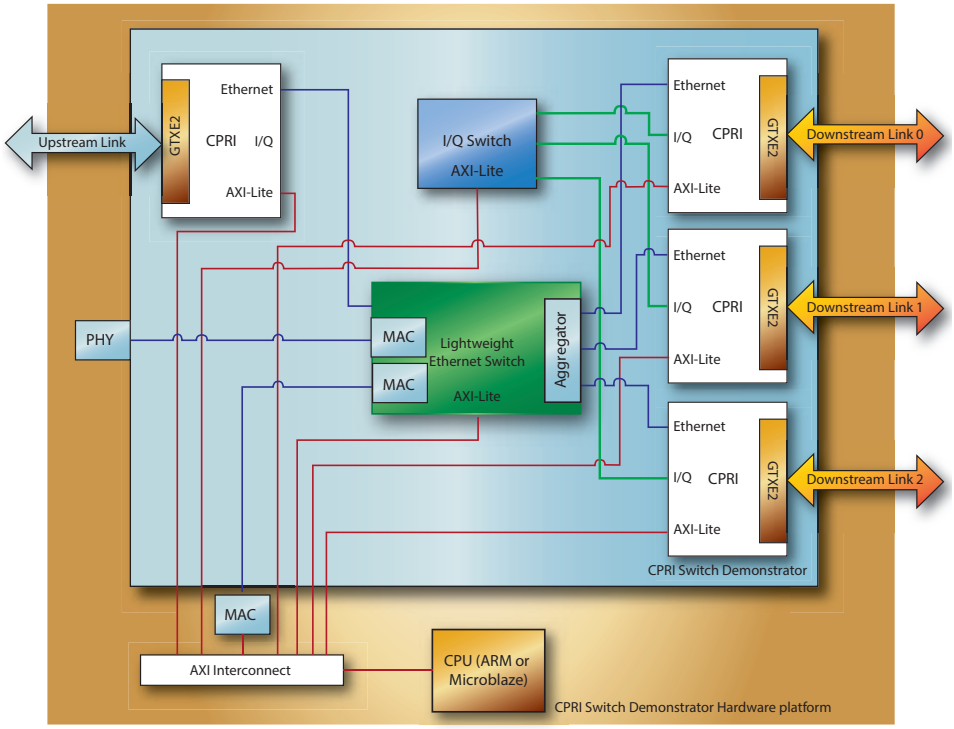


図 2 – CPRI スイッチのハードウェア プラットフォーム

機能です。CPRI 規格は、この測定を容易にする手法を定義しています。

IPI を使用してインターフェイスを接続

IPI バス インターフェイスは、定義済みの一連の論理ポートを、IP コア上の特定の物理ポートにマッピングします。インターフェイスが使用できる場所では、多数の信号を接続する手法から、少数のインターフェイスを接続する手法へと移行します。IP コア上の共通バス インターフェイスは、AXI4-Lite や AXI4-Stream などの ARM® AXI 仕様に準拠しています。このように抽象化のレベルを上げることで、すばやく簡単なデザイン エントリが可能となり、インターフェイスに対してデザイン ルール チェックを利用できます。Vivado IP パッケージャーを使用すれば、IP インテグレーター内でユーザー独自の IP コアを使用することや、ユーザー独自のデザイン内でインターフェイスを活用することができます。

IPI により、インターフェイスを簡単に相互

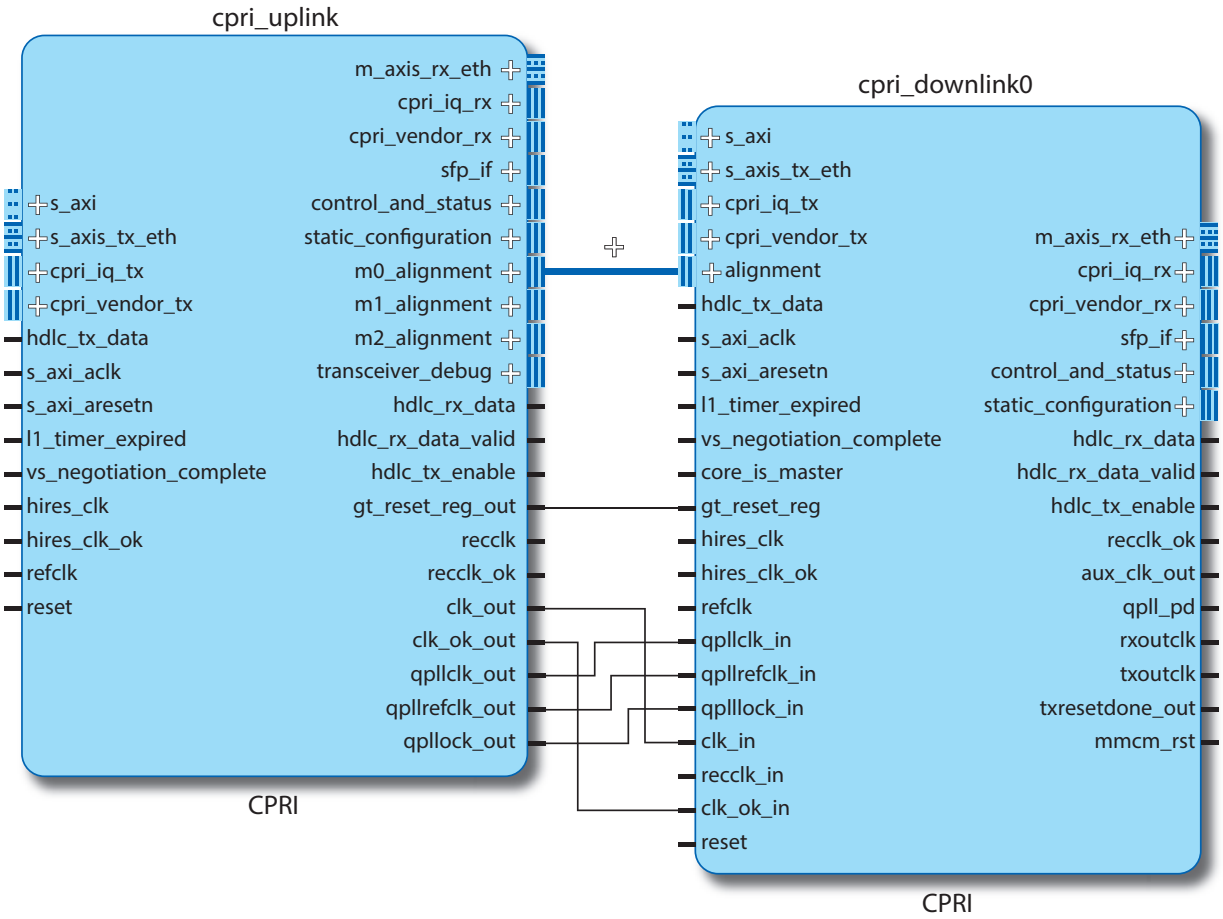


図 3 – QPLL の共有ロジックの接続

接続できます。インターフェイスをクリックすると、IPI は接続可能な対象を指示します。希望するエンド ポイントに接続線をドラッグすると、接続が作成されます。この方法で、わずか数回クリックするだけで多くの信号を接続できます。

図 4 は、多数の AXI4-Stream インターフェイス、2 つの GMII インターフェイス、1 つの AXI4-Lite インターフェイスを提供するイーサネット スイッチを示しています。ストリーミング インターフェイスにより、CPRI コアへの直接接続が可能となり、CPRI コア上の内部バッファリングが不要になります。GMII インターフェイスにより、イーサネット PHY への接続が可能となり、フィールド エンジニアがネットワークの問題をデバッグする際に便利です。AXI4-Lite 管理インターフェイスにより、アドレス テーブルのマッピングと、アドレス テーブルのエージング間隔などのコンフィギュレーション オプションにアクセスできます。

このように作業を続けながら、IPI 内でインターフェイスを接続し、システムを構築できます。ユーザーに最もよく合ったエントリ手法を柔軟に利用できます。GUI を使用してインターフェイスをリンクする以外に、Tcl コンソールからコマンドを直接発行することも、スクリプトからコマンドを供給することもできます。GUI 内で何か操作を行うたびに、その結果生じるコマンドがエコー バックされます。

デザインの作成が完了したら、コマンド「write_bd_tcl」を使用してデザイン全体をエクスポートすることもできます。このコマンドで作成される Tcl ファイルをソースとして、ブロック デザイン全体をゼロから作成できます。またこのファイルは、スクリプトで記述されるビルド フローの一部として簡単に使用できます。デザイン内のすべての IP コアは、これらのコアのホスト プロセッサへの接続を可能にする AXI4-Lite 管理インターフェイスを提供します。IPI に内蔵されたインテリジェンスは、接続の自動化を実現します。このメカニズムにより、IPI は、ユーザー IP コア上の AXI4-Lite インターフェイスが AXI バス インターコネクトに接続することを認識し、適切なアドレス範囲を自動的に設定して、ユーザーのバスを接続します。次に、ユーザーは IPI の助けを借りて、このバスをホスト プロセッサに接続できます。筆者らのデザインではホスト プロセッサは MicroBlaze™ ですが、Zynq SoC シリーズ デバイスを使用する場合は、これを簡単に変更して ARM 社の CPU を使用できます。

今後期待されるメリット

Vivado IP インテグレーターの機能は急速に成長しています。この成長とともに、今後さらに大きなメリットが期待されます。適切な IP コアと組み合わせて、サブシステム全体を迅速に構築し、収益力を高められます。CPRI、イーサネット スイッチ IP コア、あるいは IQ スイッチ IP コア の詳細は、ザイリンクス ワイヤレス コミュニケーションズの Perminder Tumber (permind@xilinx.com) にお問い合わせください。

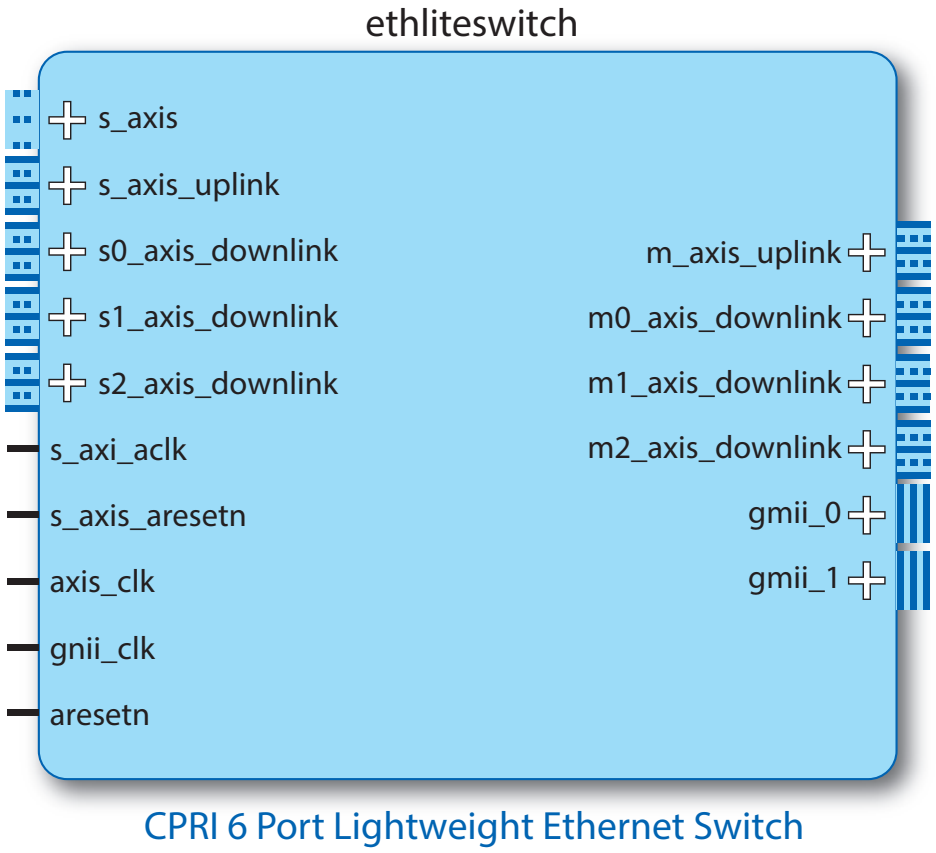
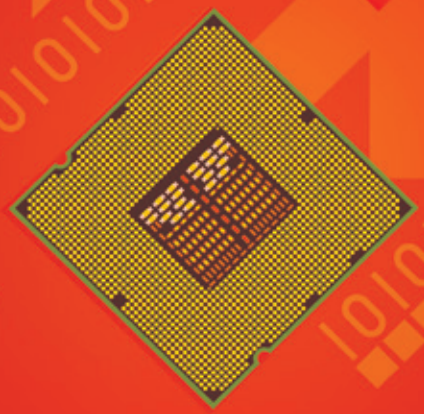


図 4 – インターフェイスを備えたイーサネット スイッチ シンボル

Getting the Most out of Your PicoBlaze Microcontroller

PicoBlaze マイクロコントローラーを最大限に活用

Adam P. Taylor
Head of Engineering – Systems
e2v
aptaylor@theiet.org



多くの FPGA アプリケーションでは、シンプルなソフト コア プロセッサを使用してシーケンシャル制御構造を簡単に生成できます。

PicoBlaze™ は、ザイリンクス FPGA 内にインスタンシエートされるコンパクトな 8 ビット ソフト コア マイクロコントローラーです。インプリメントされた PicoBlaze コアは、FPGA ファブリックに完全に組み込まれ、ロジック スライスとブロック RAM のみを使用します。外部の揮発性メモリや不揮発性メモリは必要ありません。

PicoBlaze は実装面積が小さいので、1 つの FPGA に複数の PicoBlaze インスタンシエーションを実装し、各インスタンシエーションを使用して (通常ならステート マシンによって作成される) 制御構造をインプリメントできます。その結果、開発期間を短縮でき、標準化された手法で制御構造を生成できます。高性能なザイリンクスの FPGA ファブリックを基盤とする PicoBlaze インスタンシエーションは、しばしば個別の 8 ビット マイクロコントローラーを上回る性能を発揮します。

この記事では、この便利なデバイスをデザイン内で上手に利用する方法について説明します。

PicoBlaze アーキテクチャ

PicoBlaze コアを使用する前に、PicoBlaze のアーキテクチャについて簡単に説明します。PicoBlaze は、RISC アーキテクチャ ベースの非常にシンプルな 8 ビット マイクロコントローラーです (図 1 を参照)。このコントローラーは 12 ビット アドレス ポートを搭載し、4,096 のメモリ ロケーションをアドレス指定できます。各アドレス ロケーションは、コアが実行する操作を定義する 18 ビット命令を保持します。コアとの間の入出力は、2 つの 8 ビット ポート (1 つは入力、1 つは出力) を介して可能です。また、PicoBlaze コントローラーは 8 ビット識別ポートを搭載し、最大 256 個のペリフェラルの読み出しと書き込みが可能です。サイズを選択可能なスクラッチ パッド (64 バイト、128 バイト、または 256 バイト) もあります。すべてのマイクロコントローラーと同様に、PicoBlaze は 1 つの論理演算ユニット (ALU) を搭載し、1 つの割り込みをサポートします。これらの機能により、PicoBlaze は FPGA 設計技術者に多くのメリットをもたらします。

PicoBlaze の最も重要な利点の 1 つは、高度な確定的性質です。PicoBlaze は 2 クロック サイクルですべての命令を実行し、最大 4 クロック サイクルで割り込みを処理します (PicoBlaze アーキテクチャの詳細は、ダウンロード可能なザイリンクスのユーザー ガイドを参照)。

PicoBlaze を使用するメリット

FPGA アプリケーションは、通常はパラレル処理とシーケンシャル処理を組み合わせて実行する必要があります。データ フローは主にパラレルで処理され、制御構造 (ステート マシンなど) は主にシーケンシャル構造としてインプリメントされます (Xcell Journal 日本語版 81·82 合併号の記事「FPGA にステート マシンをインプリメント」を参照)。しかし、複雑な制御構造をステート マシンとしてインプリメントすると、非常に扱いづらくなり、また検証に時間がかかる、開発サイクル後半での変更が困難になるなどの問題が起こります。複雑なステート マシンは開発に時間がかかるので、複数のステート マシンを使用する場合はかなりの開発期間が必要になります。

PicoBlaze は、RS232、I2C、および SPI 上のシリアル通信の制御にも使用できます。実際に、標準的な 8 ビット マイクロコントローラーで実行できるあらゆる処理は PicoBlaze にインプリメント可能で、さらに

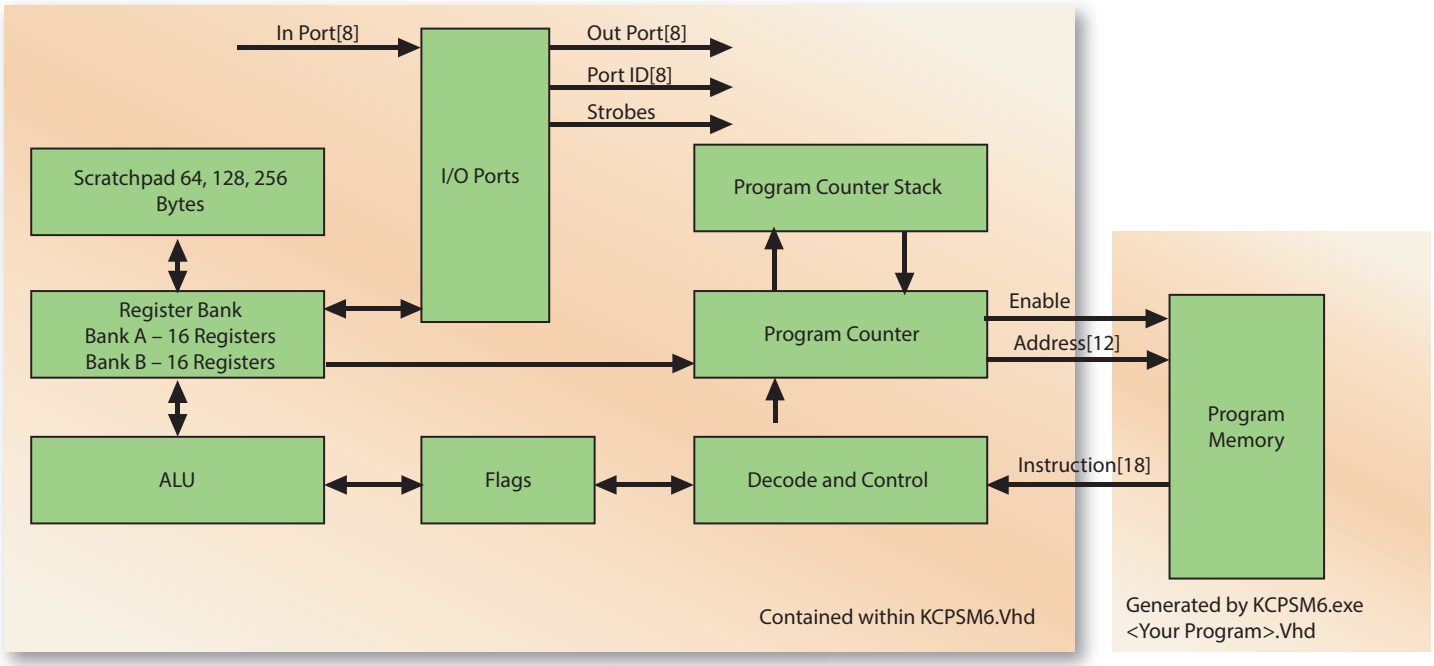


図 1 – PicoBlaze アーキテクチャ (左側のボックスにプロセッサ、右側にメモリを示す)

性能向上という利点も得られます。これまで技術者たちは、PicoBlaze を使用して制御システム内に PID コントローラーをインプリメントしてきました。また、PicoBlaze と I2C、SPI、またはパラレル DAC を組み合わせて、簡単な方形波、のこぎり波、三角波から複雑な正弦波 / 余弦波までのリファレンス波形を生成してきました (CORDIC アルゴリズムのシフト アンド アド (shift-and-add) を使用)。

FPGA 内に PicoBlaze マイクロコントローラーをインスタンシエートし、これらのシーケンシャル機能をインプリメントすれば、開発期間が短縮され、開発サイクル後半での変更も簡単に行えます。もちろん、ソフト コアである PicoBlaze は、陳腐化の問題にも対応しやすく、ASM モジュールが開発された時点で簡単にデザインを再利用できます。

初めての PicoBlaze のインスタンシエーション

次の簡単な初期手順に従って、デザイン内に PicoBlaze をすばやくインプリメントできます。まず、ターゲットとするデバイスに適合する最新バージョンの PicoBlaze マイクロコントローラーを用意します。これはザイリンクスの PicoBlaze ラウンジから入手できます。PicoBlaze ラウンジでは、最新の 7 シリーズ デバイス対応版から以前の Spartan®-3 および Virtex®-4 デバイス対応版まで、さまざまなバージョンを提供しています。

適切なバージョンの PicoBlaze プロセッサをダウンロードしたら、ダウンロードしたファイルを作業ディレクトリに展開し、必ず「readme」ファイルに目を通します。必要な PATH および XILINX 環境変数の設定に十分注意してください。作業ディレクトリには、通常の readme ファイル、ライセンス ファイル、およびユーザー ガイドとともに、次のファイルまたはアプリケーションが作成されます。

- KCPSM6.vhd : PicoBlaze の実際のソース コード。
- KCPSM6.exe : 必要なマシン コードとメモリ ファイルの生成に使用されるアセンブラ プログラム。
- ROM_Form.vhd : このファイルを使用

```
NAMEREG s0,led  ;rename S0 register to led
;As 8 bit processor we need four delay loops 256 * 256 * 256 *
256 = 4294967296
CONSTANT max1, 80 ;set delay
CONSTANT max2, 84 ;set delay
CONSTANT max3, 1e ;set delay
CONSTANT max4, 00 ;set delay
main:  LOAD led, 00; load the led output register with 00
flash:  XOR led, FF; xor the value in led register with FF i.e.
toggle
       OUTPUT led,01; output led register with port ID of 1
       CALL delay_init; start delay
       JUMP flash; loop back to beginning
delay_init: LOAD s4, max4;
                    LOAD s3, max3;
                    LOAD s2, max2;
                    LOAD s1, max1;
delay_loop:   SUB s1, 1'd; subtract 1 decimal from s1
                    SUBCY s2, 0'd; carry subtraction
                    SUBCY s3, 0'd; carry subtraction
                    SUBCY s4, 0'd; carry subtraction
                    JUMP NZ, delay_loop;
                    RETURN
```

図 2 – LED を点滅させるプログラムのアセンブラ コードのスニペット

```
kcpsm6.exe
KCPSM6 Assembler v2.63
  Ken Chapman - Xilinx Ltd - 20th December 2013

Enter name of PSM file: test.psm

Reading top level PSM file...
  C:\hdl_projects\picoblaze\test.psm

A total of 21 lines of PSM code have been read

Checking line labels
Checking CONSTANT directives
Checking STRING directives
Checking TABLE directives
Checking instructions

Writing formatted PSM file...
  C:\hdl_projects\picoblaze\test.fmt

Expanding text strings
Expanding tables
Resolving addresses and Assembling Instructions
  Last occupied address: 00E hex
  Nominal program memory size: 1K (1024)    address(9:0)
  Occupied memory locations: 15
  Assembly completed successfully

Writing LOG file...
  C:\hdl_projects\picoblaze\test.log
Writing HEX file...
  C:\hdl_projects\picoblaze\test.hex
Writing VHDL file...
  C:\hdl_projects\picoblaze\test.vhd

KCPSM6 Options.....
        R - Repeat assembly with 'test.psm'
        N - Assemble new file.
        Q - Quit
```

図 3 – KCPSM6 アセンブラを使用したメモリ ファイルの生成

して、アセンブラの実行ファイルは、ユーザーが作成したプログラムが含まれる VHDL ファイルを生成します。

- KCPSM6_design_template.vhd : PicoBlaze プロセッサのテンプレート インスタンシエーション。
- All_kcpsm6_syntax.psm : このファイルは、すべてのアセンブラ コマンドとシンタックス (構文) の定義です。

筆者らのデザイン例では、最後の手順で ISE® Design Suite 内で新規プロジェクトを作成します。既存プロジェクトに PicoBlaze の機能を追加するのでなければ、この新規プロジェクトで PicoBlaze とプログラム メモリをインスタンシエートできます。

上記の手順が完了したら、アプリケーション内で PicoBlaze プロセッサの作成を開始できます。非常に簡単なレベルでは、図 1 に示すように、デザイン内で 2 つのコンポーネント (プロセッサそれ自体とプログラム メモリ) を宣言するだけです (図 1 ではプロセッサを左側のボックス、メモリを右側のボックスにコンテキスト表示)。もちろん、インスタンシエーションが 2 つ以上ある場合は、複数のメモリ コンポーネントが実装され、それぞれに異なるプログラムが格納されます。しかし、最初に、標準的なプロジェクトの開発フローを理解する必要があります。

開発フロー

初めての PicoBlaze インスタンシエーションの作成は簡単です。最初の手順では、Notepad++ などのエディターを使用して、空のテキスト ファイルを作成します。このファイルの拡張子は .PSM になります (たとえば、test.psm)。マイクロコントローラーのプログラミングには、PicoBlaze アセンブラを使用します。ザイリンクスはこのシンタックス (構文) の詳細を、ダウンロード可能な All_kcpsm6_ syntax.psm ファイルに記載しています。この構文はわかりやすく、簡単に学習できます。図 2 はアセンブラ コードのスニペットの例で、40MHz のクロックを使用して 2Hz の周波数で LED を点滅させる簡単なプログラムです。

アセンブラ プログラムが完成したら、次の段階では、(先にダウンロードした) アセンブラ実行ファイルを使用して、このプログラムを実行します。アセンブラ プログラムを実行すると、メモリ ファイル (FPGA 内で使用される VHDL)、ログ ファイル、16 進数ファイル (このファイルの用途については後述) が生成されます。図 3 は、上記のコード スニペットに対して実行されたアセンブラ プロセスを示しています。アセンブラの実行が完了したら、FPGA 内に PicoBlaze をインスタンシエートする準備は完了です。

これで、必要な 2 つの VHDL ファイル KCPSM6.vhd と、ユーザーのアプリケーションを含むアセンブラ プログラムによって作成された VHDL ファイル (この場合は test.vhd)) を用意できました。第 2 の段階では、VHDL デザイン内で 2 つのコンポーネント (KCPSM6 とメモリ) を宣言し、図 4 に示すようにインスタンシエートします。このシンプルな VHDL の例は、図 5 のコード スニペットに見られます。このコードは、LX9 Spartan® 開発ボード上で LED を点滅させる PicoBlaze をインプリメントするものです。

シミュレーションと検証

アプリケーション内でデザイン ファイルをインスタンシエートしたら、合成とインプリメンテーションに進む前に、もちろん、シミュレーション環境内でシステムまたはモジュールの性能を検証することも可能です。PicoBlaze はロジック スライスとブロック RAM を使用するため、Mentor Graphics 社

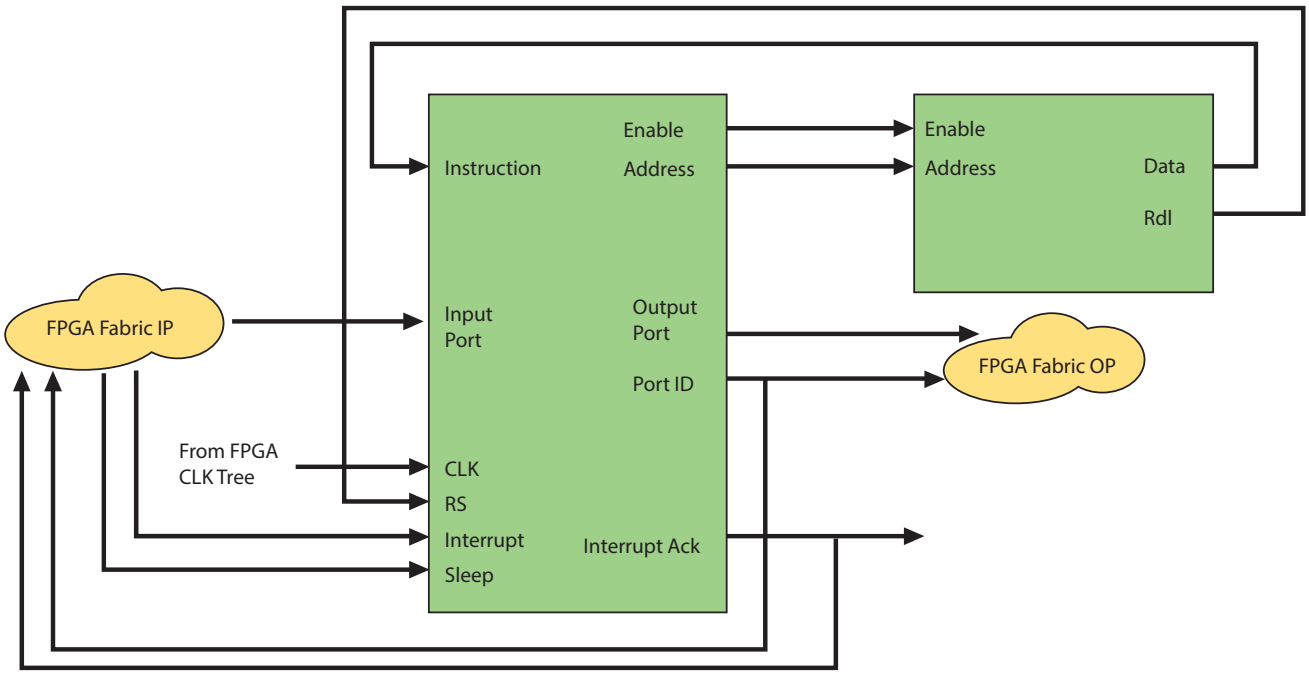


図 4 – PicoBlaze のコンテキスト ダイアグラム

の ModelSim やザイリンクスの ISE ツール内の ISim (あるいは、Vivado® Design Suite で PicoBlaze をインプリメントする場合は Vivado Design Suite 内の Xsim) などのプログラムで、非常に簡単にシミュレートできます。

ブロック RAM にはユーザー プログラム用の命令が含まれているため、シミュレーションは簡単です。基本的に、ユーザーが提供する必要があるのは、インスタンシエーションで必要とされるクロックやほかの入出力だけです。図 6 は、ISim による PicoBlaze のシミュレーション結果と、命令のローディング間隔が 2 クロック サイクルであることを示しています。

プログラムの更新

FPGA (および BIT ファイル) に PicoBlaze を組み込む大きな利点の 1 つは、FPGA のコンフィギュレーションの終了後に、PicoBlaze の IP コアが RAM 内のプログラムの実行を開始することです。しかし、コアが実行しているプログラムの修正が必要になる場合があります。更新されたメモリ ファイルを含めてインプリメンテーション段階を再実行することもできますが、デザインのその他の部分の複雑性によっては、長い時間がかかることがあります。特にラボ内でさまざまな可能性を試行するだけの場合、時間の無駄となります。したがって、インプリメンテーション段階を再実行する前に、ダウンロード可能な JTAG ローダー プログラムを使用して、コアが使用するプログラム メモリを更新し、この更新されたプログラムをテストすることが可能です。

JTAG ローダーを使用する際は、最初の手順として、デザインの設定で JTAG ローダーを有効化する必要があります。1 つのプログラム メモリのインスタンシエーション内で、ジェネリック C_JTAG_LOADER_ENABLE : integer := 1 を使用します。なお、このパラメーターを設定できるのは、ユーザー デザイン内の 1 つのメモリ インスタンシエーションに対して一度だけです。

デザイン内でこの機能を有効化したら、まず JTAG_loader ディレクトリから、ユーザーが使用しているオペレーティング システムに適合するバージョンのローダーを選択し、それを (16 進数ファイルが置かれている) 作業ディレクトリにコピーします。ここで、

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity pico_wave_top is
    Port ( clk : in  STD_LOGIC;
           led  : out  STD_LOGIC_VECTOR (3 downto 0));
end pico_wave_top;

architecture Behavioral of pico_wave_top is

component kcpsm6 is
  generic(   hwbuild : std_logic_vector(7 downto 0) := X"00";
             interrupt_vector : std_logic_vector(11 downto 0) := X"3FF";
             scratch_pad_memory_size : integer := 64);
  port (     address : out std_logic_vector(11 downto 0);
             instruction : in std_logic_vector(17 downto 0);
             bram_enable : out std_logic;
             in_port : in std_logic_vector(7 downto 0);
             out_port : out std_logic_vector(7 downto 0);
             port_id : out std_logic_vector(7 downto 0);
             write_strobe : out std_logic;
             k_write_strobe : out std_logic;
             read_strobe : out std_logic;
             interrupt : in std_logic;
             interrupt_ack : out std_logic;
             sleep : in std_logic;
             reset : in std_logic;
             clk : in std_logic);
  end component kcpsm6;

component test is
  generic(     C_FAMILY : string    := "S6";
               C_RAM_SIZE_KWORDS        : integer := 1;
               C_JTAG_LOADER_ENABLE : integer := 1);
  Port (       address : in std_logic_vector(11 downto 0);
               instruction : out std_logic_vector(17 downto 0);
               enable : in std_logic;
                      rdl : out std_logic;
                      clk : in std_logic);
  end component test;


SIGNAL instruction : std_logic_vector(17 DOWNTO 0);
SIGNAL address : std_logic_vector(11 DOWNTO 0);
SIGNAL enable : std_logic;
SIGNAL rd1 : std_logic;
SIGNAL kcpsm6_output : std_logic_vector(7 downto 0);
SIGNAL port_id : std_logic_vector(7 downto 0);
SIGNAL write_strobe:std_logic;
begin

ram_inst : test PORT MAP  (
             address => address,
             instruction => instruction,
             enable => enable,
             rdl => rd1,
             clk => clk);
```

```
pico_inst :  kcpsm6 PORT MAP (
             address => address,
             instruction => instruction,
             bram_enable => enable,
             in_port  => (OTHERS =>'0'),
             out_port => kcpsm6_output,
             port_id  => port_id,
             write_strobe => write_strobe,
             k_write_strobe => open,
             read_strobe => open,
             interrupt  => '0',
             interrupt_ack => open,
             sleep => '0',
             reset => rd1,
             clk => clk);

output_ports: process(clk)
begin
if rising_edge(clk) then
if write_strobe = '1' then
-- 4 LEDs at port address 01 hex Spartan LX9
   Development board
               if port_id(0) = '1' then
                    led <= kcpsm6_output(3 DOWNTO 0);
               end if;
end if;
end if;
end process;
end Behavioral;
```

図 5 – LX9 Spartan 開発ボード上で LED を点滅させる PicoBlaze のコード スニペット

コマンド ウィンドウを開き、作業ディレクトリに移動して、次のコマンドを使用します。

jtagloader -l < プロジェクト名 >.hex

注記：筆者の OS に適合する実行ファイルのバージョンの名前を、jtagloader.exe に変更しました。

このコマンドを実行すると、最新の PSM ファイル上でアセンブラを実行した際に作成された 16 進数ファイルがダウンロードされ、図 7 に示す結果が得られます。このファイルがダウンロードされると、JTAG ローダーはコアの実行を停止し、新しいプログラムをメモリにダウンロードした後、コアをリセット状態から解放します。この時点で、コアは新しいプログラムの実行を開始します。

PSM ファイルの更新された動作の確認を終えたら、インプリメンテーションと BIT ファイルの生成を再実行できます。これにより、次回デバイスがコンフィギュレーションされるときは、デバイスは更新されたプログラムを実行します。

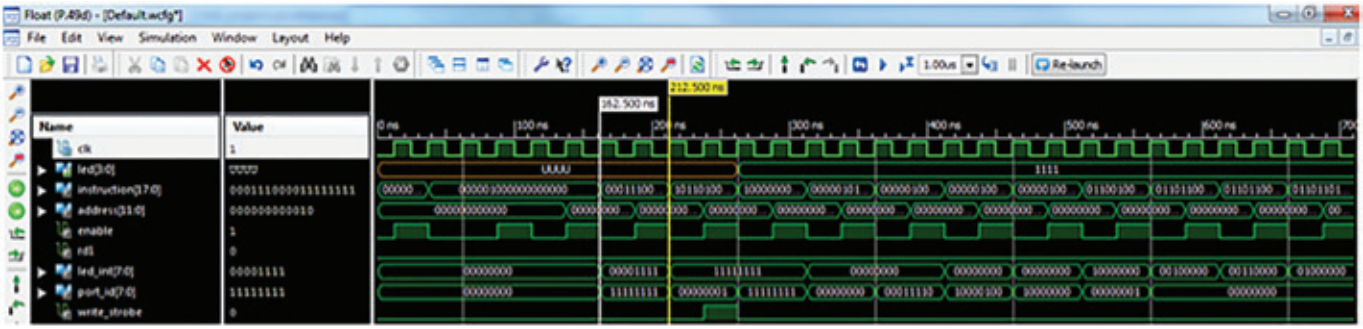
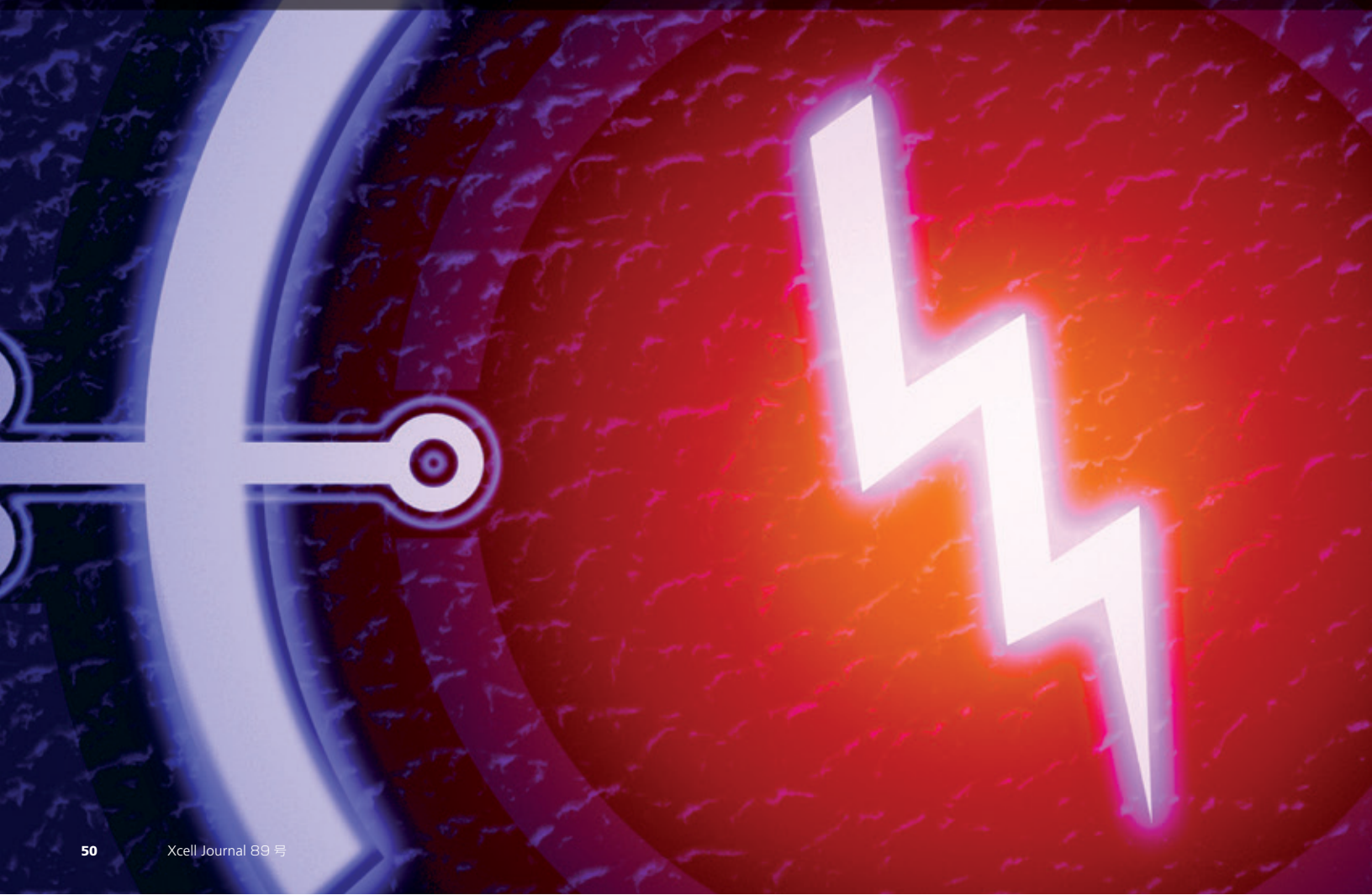


図 6 – ISim のシミュレーション結果

図 7 – 動作中の JTAG ローダー

Changing Utilization Rates in Real Time to Investigate FPGA Power Behavior

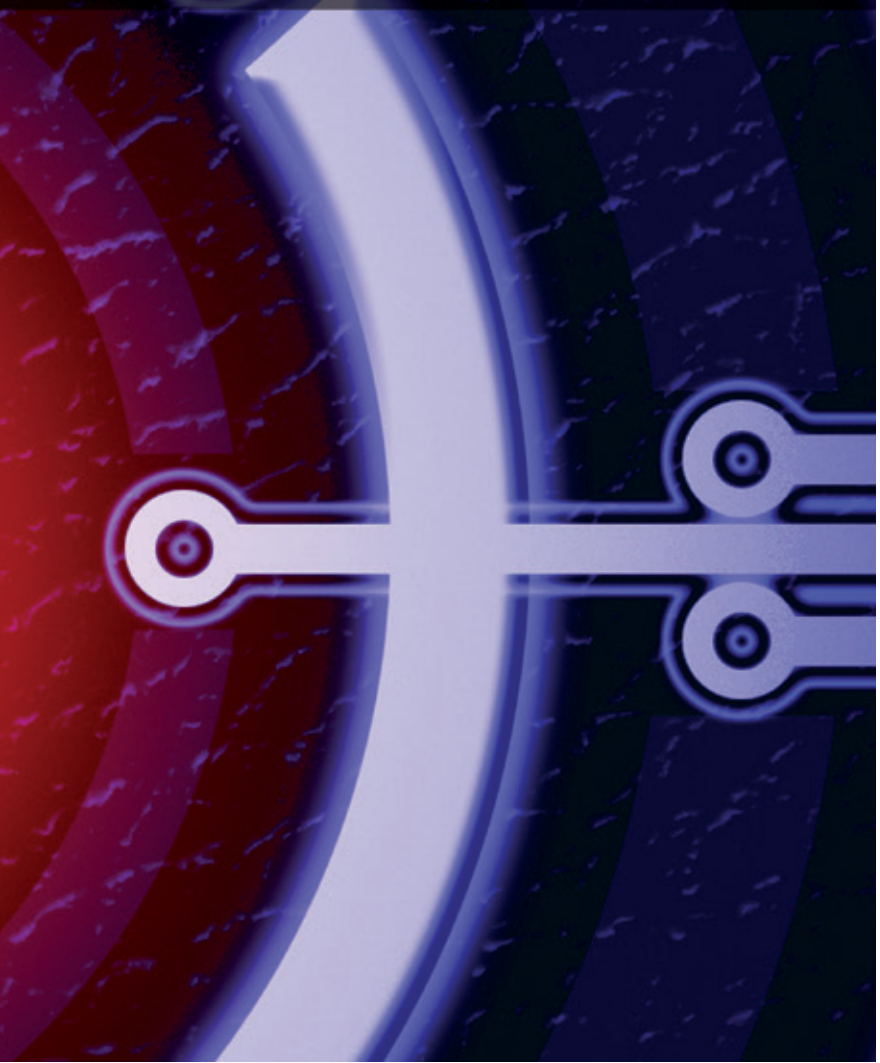
使用率をリアルタイムで変化させて FPGA のパワー ビヘイビアーを推定



FPGA デザインの パワー マネージメントは、常に重要な問題です。実際の FPGA デバイス上で推定消費電力を測定する新たな方法について説明します。

Ahmet Caner Yüzügüler
Hardware Design Engineer
Aselsan
acyuzuguler@aselsan.com.tr

and Emre Sahin
Senior Hardware Design Engineer
Aselsan
emresahin@aselsan.com.tr



今日の FPGA チップは、高性能アプリケーション開発への対応力が非常に優れていますが、デザイン内のパワー マネージメントが制約要因になりがちです。デザインの容量と処理速度は、主に FPGA デバイスのリソース使用率によって決まりますが、リソースを追加すれば消費電力が増加します。消費電力の増加は、運用コストの増大、必要な面積の拡大、ジャンクション温度の上昇をもたらします。設計者は、エアフローと冷却システムの強化でこれに対処しなければなりません。

ボードやシステムの総消費電力は非常に重要であるため、設計者は、リソース使用率と消費電力のトレードオフを調整し、電力バジェットを設定する必要があります。したがって、システムの電力消費量を事前に予測する機能があれば、一歩先を行く対応が可能になります。

ザイリンクスは、インプリメンテーション前の消費電力評価用に、ユーザーの入力または合成レポートに基づいた仮想的な消費電力推定ツールをいくつか用意しています。その 1 つは、ザイリンクス パワー エスティメーター (XPE) スプレッドシートです。この Excel ベースの消費電力推定ツールでは、ユーザーはリソース使用率、トグル レート、クロック周波数などのデザイン プロパティを入力し、デバイスの情報に従って推定消費電力を計算します。よく使用されるもう 1 つのツールは、ザイリンクス パワー アナライザー (XPA) です。配置配線後、XPA は生成された NCD ファイルをインポートし、より正確な推定値が得られるように、ユーザー入力の代わりにインプリメンテーションの詳細とシミュレーション結果を使用して消費電力を推定します。

筆者らは、これらの代替手段として、実際のデバイス上で FPGA デザインの推定消費電力を測定する新しい手法を開発しました。さらに、さまざまなインプリメンテーション シナリオを再現するために、汎用のデバイス フリー VHDL デザインを作成しました。このデザインは、FPGA を動作させながら、アクティブ化されたリソースの数 (すなわち、DSP スライス、ブロック RAM、スライス レジスタ) と各リソースの動作条件 (ジャンクション温度、クロック周波数、トグル レート) を、シリアル チャネルを介して変更できます。筆者らの手法は、さまざまなリソース使用率と周囲温度で、デバイスのダイナミック消費電力特性を簡単に観察できるように、電源レールの電流レベルと電圧レベルを同時にモニターします。

筆者らはこのデザインをザイリンクス KC702 評価ボード上に実装しましたが、デザインに使用される IP コアをサポートする FPGA デバイスであれば、数箇所の簡単な変更で、ほかの任意のデバイスに実装できます。

筆者らの手法を使用したもう 1 つの方法として、FPGA ボードの VHDL デザインをテストすることも考えられます。

汎用 Kintex®-7 ボードを最近設計した場合を考えます。顧客はこのボード上に任意のデザインをインプリメントできますが、刻々と変化する周囲温度でどれだけの量のリソースが使用されるかは不明です。ボードの安定性と堅牢性を確保するには、デバイスの動作を想定される最高 / 最低周囲温度内の動作に限定し、さまざまなリソース使用率で FPGA が正常に動作することを確認する必要があります。しかし、すべてのリソース使用率のシナリオをテストするには、新しい VHDL デザインをゼロから設計する必要がありますが、それでは時間がかかりすぎます。筆者らが推奨する手法では、設計者は必要に応じて柔軟に、リアルタイムでテスト デザインを操作できます。

インプリメンテーションの詳細

筆者らは、ロジックとして使用される (消費電力を制御できない) スライス LUT の生成を防ぐため、インプリメンテーションをできるだけシンプルにしました。スライス レジスタをインプリメントする最も簡単な方法は、それらをシフト レジスタ ブロックとして組み合わせることです。図 1a は、スライス レジスタがどのようにインプリメントされるかを示すブロック図です。ここで注意すべき点は、筆者らがスライス レジスタを作成しようとしても、合成ツールはスライス レジスタの代わりに 32 ビット シフト レジスタ (SRL32) としてスライス LUT を使用しがちであることです。次の VHDL 属性を使用して、合成ツール ソフトウェアがスライス レジスタを使用するように強制できます。

LogiCORE™ Block Memory Generator は、ブロック RAM をシングル ポート RAM

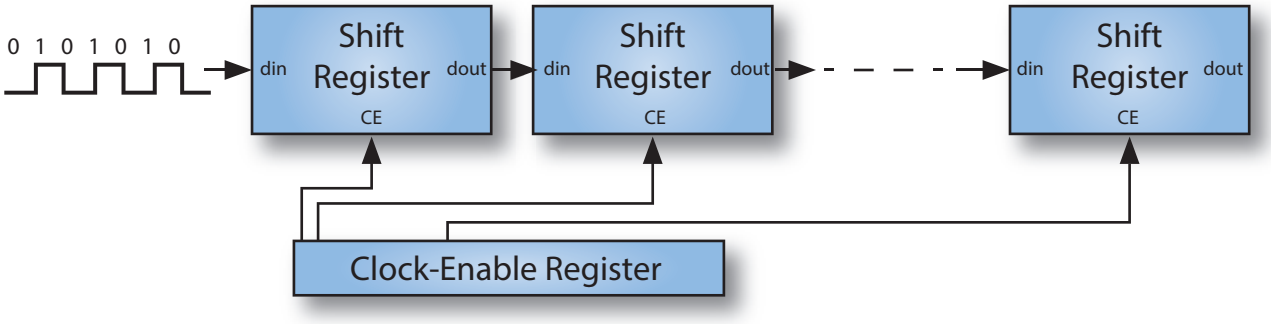


図 1a

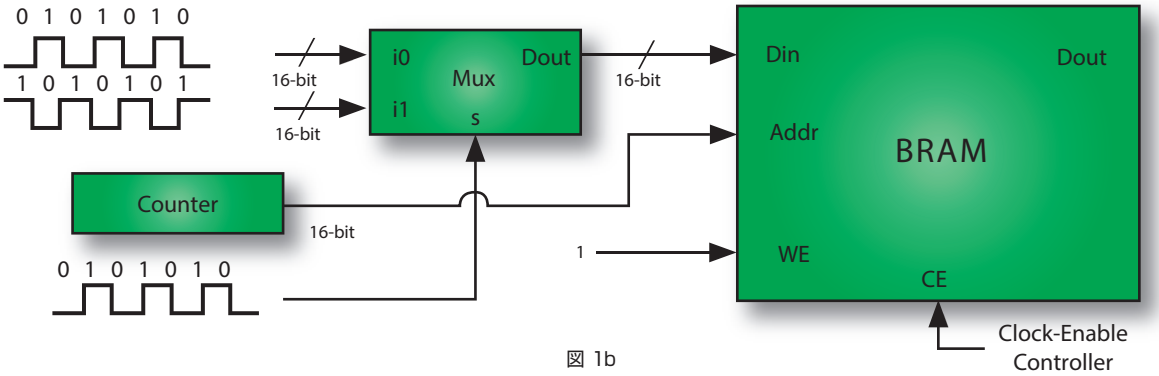


図 1b

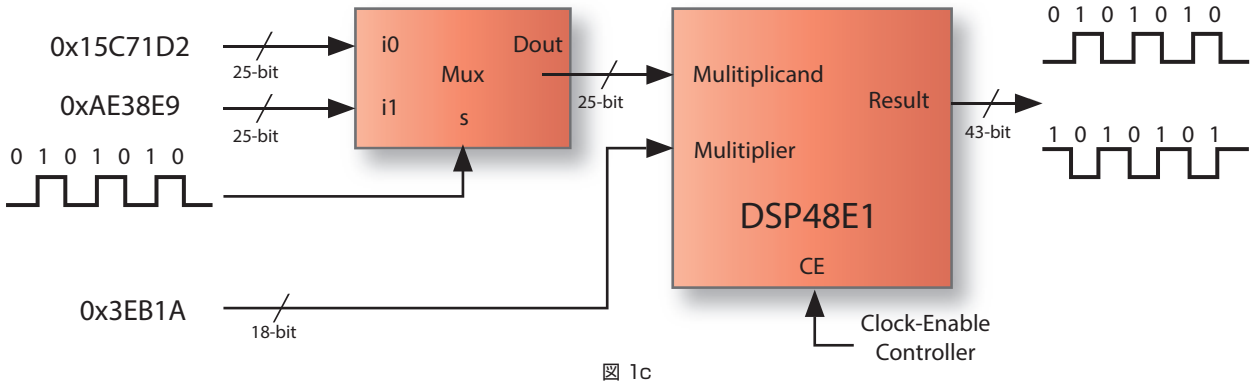


図 1c

図 1 — インプリメンテーションのブロック図 : (a) スライス レジスタ、(b) ブロック RAM、(c) DSP スライス

として作成します。ビットが継続的にトグルされる 16 ビット ワードが、アクティブ化された BRAM のランダム アドレスに常に書き込まれ、それらのアドレスを占有された状態に保ちます。図 1b は、1 つの BRAM コンポーネントのブロック図です。同様に、DSP スライスを使用して 25 ビット被乗数に 18 ビット乗数を掛けて、43 ビットの出力を生成します (これは 1 個の DSP48E1 が処理できる最大ワード幅です)。図 1c に示すように、すべての DSP がダイナミック電力を消費するように、すべての DSP コンポーネントの被乗数は定期的に変更されます。ただし、これらのモジュールの出力はどの出力ピンにも接続されていないため、合成ツールは合成プロセスでこれらのリソースを削除しようとします。次の属性を使用して、リソースが削除されないように指定できます。

```
attribute SHREG_EXTRACT:      string;
attribute SHREG_EXTRACT of <signal_name> : signal is "no";
```

この方式で、リソースのクロック イネーブル信号の制御により、リソースをアクティブ状態にします。クロックがディスエーブルにされたリソースは、ほとんど電力を消費しないため、デザイン内でまったく使用されていないと見なされます。具体的には、トグル レートが 100% の DSP スライス 50 個の消費電力は 0.112W (ワット) ですが、クロック イネーブル信号がディアサートされると 0.001W になります。その結果、さまざまな使用率でほぼ同じ結果が得られます。したがって、各リソース コンポーネントの使用 / 不使用がシリアル通信を介したクロック イネーブル信号で制御されるデザインでは、刻々と変化する使用率を瞬時にシミュレートできますので、新たにデザインしたり、それをインプリメントしたりする必要はありません。

```
attribute KEEP : string;
attribute KEEP of <signal_name>: signal is "TRUE";
```

消費電力に大きな影響を与える重要な要因の 1 つは、リソースの平均トグル レートです。トグル レートとは、特定のリソースの 1 クロック サイクル当たりの出力の遷移回数です。リソースの入力を切り替えて、サイクルごとに出力の状態を変化させれば、常時 100% のトグル レートを実現できます。DSP スライスの場合、たとえば、出力の各ビットがサイクルごとに変化するように、連続的に切り替えられる 2 つの被乗数と、1 つの乗数を選択することにより、トグル レートを調整できます。入力の切り替えレートによって、リソースのトグル レートが決まります。このように、シリアル チャネルを介して、各種リソースのトグル レートをオンザフライで制御します。

また、消費電力は、クロック周波数に直接比例します。筆者らは、混合モード クロック マネージャー (MMCM) を使用して、可変周波数のクロックを生成しました。一組のレジスタにより、MMCM の出力の周波数、位相、デューティ サイクルを指定します。これらの値は、通常はデザインがインプリメントされ、ビットストリーム ファイルが生成されるときにのみ初期化されます。ただし、MMCM のダイナミック リコンフィギュレーション ポートを使用して、FPGA の動作中に、出力クロックとしてのこれらの特性を変更できます。クロックの位相は消費電力に影響を与えませんし、デューティ サイクルは通常は変化しないため、位相とデューティ サイクルは問題になりません。

一方、周波数には、消費電力との強い相関関係があります。MMCM の内部メカニズムは、VCO 周波数を CLKOUT0_DIVIDE レジスタの値で割り、希望する出力周波数を得るように機能します。筆者らは、ザイリンクス アプリケーション ノート XAPP888 で説明されているような、グリッチのない遷移を実現するステート マシン アルゴリズムに従って、このレジスタに新しい値を割り当てます。同時に、グラフィカル ユーザー インターフェイス (GUI) へのユーザー入力に従って、デザインのクロック周波数をリアルタイムで変更します。これで、異なる周波数について、デザインの消費電力特性を簡単に観察できます。

高温条件下でのパワー ビヘイビアーは、もう 1 つの重要な問題であり、注意深い観察と検証が必要です。シリコンのコアの温度は、ボードのデザイン、処理速度、周囲温度、ヒートシンク、ファンのエアフローによって異なります。筆者らは、簡単な PWM オン / オフ コントローラーを使用して、FPGA チップの上部にあるファンの回転速度を変えることにより、ジャンクション温度を部分的に制御しました。ヒート ガンなどの外部ヒーティング ツールを使用して、加熱時間を短縮することもできます。

オンチップ センサーがコアの温度を測定し、LogiCORE XADC ウィザードによってアナログ / デジタル コンバーター (XADC) が生成されます。GUI にリファレンス温度の値を入力すると、その値はシリアル チャネルを介してデバイスに送信され、オンチップ センサーで測定されたコアの温度と比較されます。ジャンクション温度が希望するレベルで安定するように、ファンの回転速度が制御されます。この方法で消費電力を観察し、コアの温度に対してプロットします。これにより、消費電力がデザインの電力バジェットに適合しており、必要なヒーティング プロファイルの許容限界を超えていないことを確認できます。

グラフィカル ユーザー インターフェイスは、FPGA デバイスとの通信によって上記のパラメーターを簡単に変更できるように設計されています。図 2 に、GUI のスクリーンショットを示します。まず、利用可能なポートのうち 1 つを使用して、標準 UART を介してデバイスに接続します。次に、ザイリンクス パワー エスティメーターを開き、消費電力の推定値と測定値を比較します。GUI は、パラメーターのデフォルト値を使用して、推定値と実際の消費電力の両方について、時間に対する消費電力のグラフのプロットをただちに開始します。関連するパネル上でこれらのパラメーターを変更し、[Resource] パネル上で使用するスライス リソースの数とそれらのトグル レートを指定できます。[Utilization Sweep] パネルでは、特定のリソースの使用率を 0 ～ 100% の範囲で

等間隔にスイープし、各区間の消費電力の測定値を使用して、使用率に対する消費電力のグラフをプロットできます。

同様に、[Clock Select] パネル上で MMCM の出力を変更し、クロック周波数をスイープできます。次の列では、FPGA チップの電源電圧が測定され、[Vccint] パネル上に連続的にプロットされます。一方、[Core Temperature] パネル上には、コアの温度が表示され、グラフがプロットされます。ユーザーは、用意されたボックスにリファレンス温度を入力し、ジャンクション温度を変更できます。

電源の電圧および電流レベルは、KC702 ボード上に搭載された Texas Instruments 社の デジタル PWM システム コントローラー UCD9248 によって測定されます。この電源コンバーター用マルチレール / マルチフェーズ PWM コントローラーは、PMbus (Power Management Bus) 通信プロトコルをサポートします。このコントローラーの PWM 信号は、Vccint 電源電圧を安定化する UCD7242 集積回路を駆動します。一組の PMBus コマンドを使用して、IC の機能を設定します。UCD7242 は、オンチップ電圧 / 電流検出回路を搭載し、UCD9248 と通信します。筆者らの GUI ソフトウェアは、対応する PMbus コマンドをこのデバイスに継続的に送信し、DC/DC コンバーターの電圧値および電流値を標準的な PMbus データ フォーマットで受信します。受信したバイトを実際値に変換すると、電源の電圧および電流レベルが得られます。

複雑で密集したデザインを高温で動作させると、電源電圧レベルが変動することがあります。残念なことに、FPGA チップの入力電圧の許容範囲は広くありません。許容されるレベルを超えると、機能が失われたり、チップに永久的な損傷を与えたりする場合があります。したがって、厳しい動作条件下での電圧レベルの安定性についても、ボードの設計後にテストしなければなりません。[Vccint] パネルで、設計したボードの電源システムが、高温下で圧倒的な処理速度で動作する際の電圧の変動に耐えるかどうかを観察、検証できます。

設計者が常に注意するべき 1 つの重要な点は、製造プロセスのばらつきが原因で、消費電力はデバイスごとに異なることです。たとえば、Virtex-7 FPGA チップの製造後、テスト段階でチップを正常に動作させるのに十分な最小電源電圧のレベルが確定され、eFUSE レジスタに書き込まれます。この最小電圧レベルに合わせて電圧レギュレータの出力を調整すると、スタティック消費電力も変化します。したがって、スタティック消費電力のオフセットが原因で、XPE の結果と筆者らのデザインの結果に差が生じることがあります。たとえば、XPE で推定される Kintex-7 チップのスタティック消費電力は、標準値とワースト ケースの計算の間に 0.7W の差があります。

XPE との高い一致度

筆者らは、さまざまな要因を含むさまざまな設計シナリオでこのデザインをテストし、精度と信頼性を確認しました。一方、XPE の推定結果は筆者らの測定値と同じグラフ上にプロットされるので、消費電力の推定値と

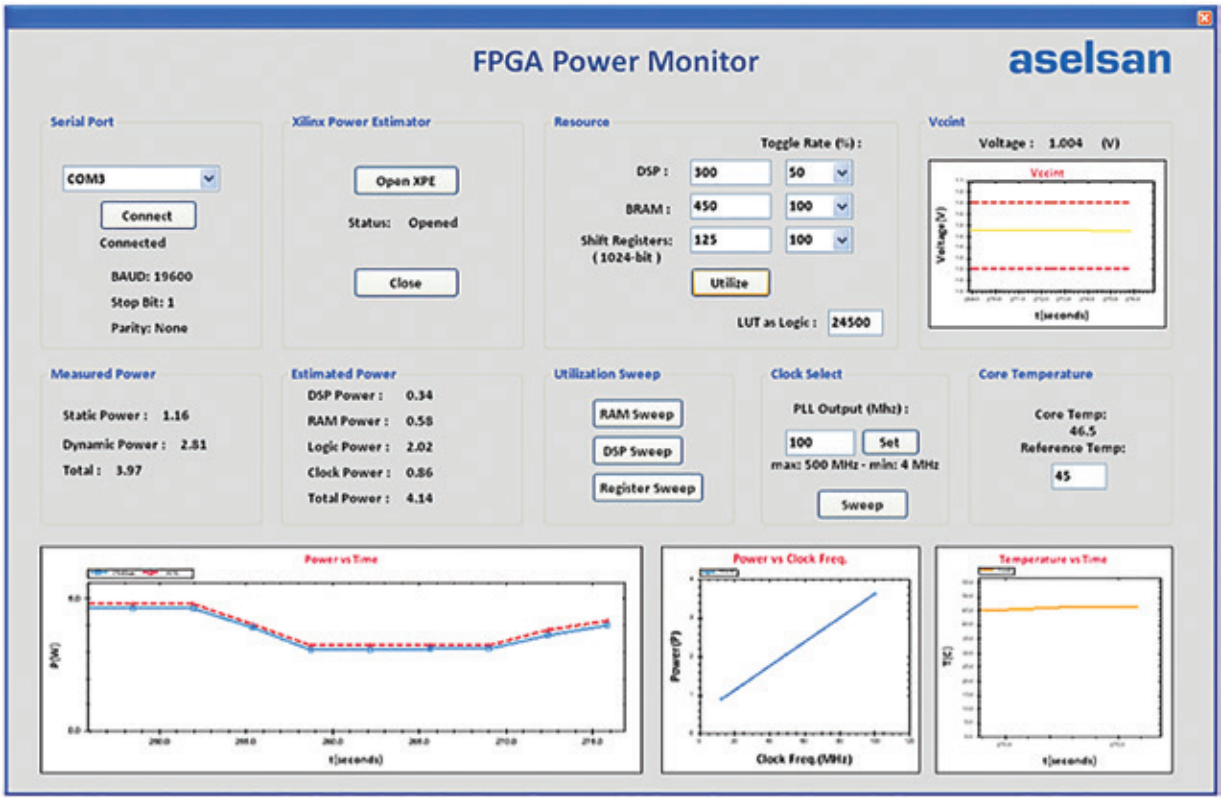


図 2 – 筆者らのグラフィカル ユーザー インターフェイス

実際の測定値を比較できます。

Kintex-7 デバイスのパワー ビヘイビアーを観察するには、0 から (デザイン上に最初にインプリメントされた) 最大値までリソース使用率をスイープします。図 3 は、あるリソース タイプの使用率をスイープしたときの消費電力の変化を示しています。直線は実際の電力の測定値、破線は XPE の推定値を表します。使用率と消費電力の値が大きい領域でも、2 本の曲線はほぼ重なり合っていることがわかります。

これらの結果は、ザイリンクス パワー エスティメーター (XPE) の計算値は実際の測定値と整合性があり、消費電力を正確に予測していることを示しています。また、筆者らが提案した手法は予想どおりに機能しており、XPE の代替手段になり得るという結論が得られます。これらの 2 つの手法を協調的に使用すれば、さらに効果的です。XPE と筆者らの手法の両方でデザインをテストすれば、結果を二重にチェックでき、XPE で数値を変更するときのミスを防止できます。たとえば、XPE でクロックの平均ファンアウト数を正しく入力しなかった場合や、単に概念を正しく理解できず、ボックスに誤った内容を入力した場合は、生成されるプロットが重ならないため、ミスがあることがわかります。そこで入力内容を修正し、誤解を招くような推定を防ぐことができます。

最大消費電力やコアの許容温度範囲など、FPGA ボードの定格仕様は、ボードのデータシートに必ず記載されています。しかし、デザインのリソース使用率とクロック周波数が高い場合、これらの定格値を超えることがあります。したがって、可能な FPGA デザインについて、電力レギュレータがデバイスに十分な電流を供給できるか、また冷却システムが温度を許容範囲内に保てるかを検証することが重要です。筆者らの手法では、リソース使用率とクロック周波数を上げることにより、高性能要件についてボードの信頼性をテストすることが可能です。このようなテストは、このボード上にインプリメントされるデザインで可能な、最大信号処理レートを確定するのに役立ちます。あるいは、このテスト結果に基づいて、ボードの機能やシステムの冷却メカニズムをアップグレードする必要があるかどうかを判断できます。

テストの結果から、筆者らの手法が、リソース使用率を操作して消費電力を評価するための信頼できる方法であることが確認されました。これにより、デザイン前の段階でパワー マネージメントに使用できる方法の選択肢が 1 つ増えました。今後、JTAG インターフェイス、完全にデバイスフリーの VHDL コード、一般的な電流検出システムなどがさらに改善されれば、このデザインは FPGA プロジェクトの主要なツールとなるでしょう。

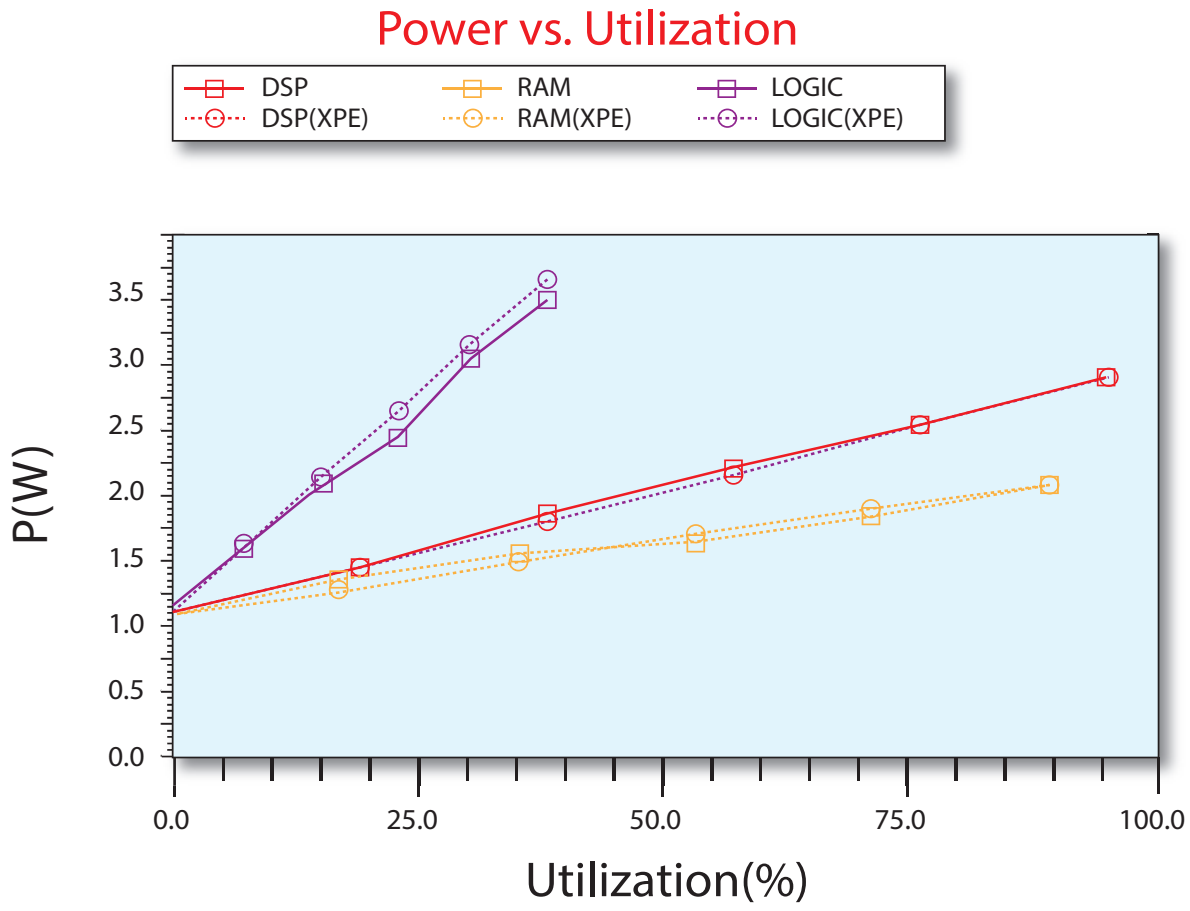
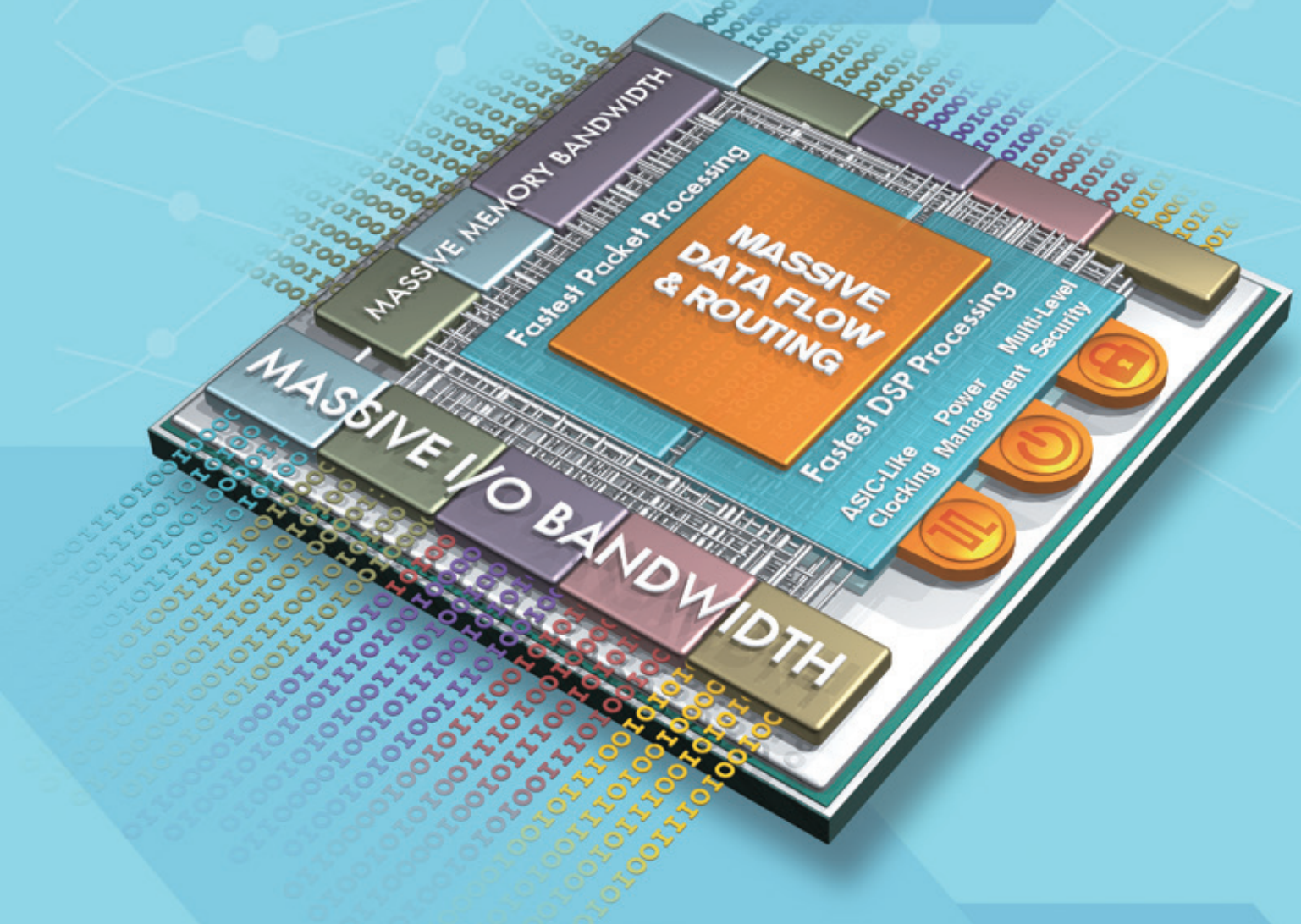


図 3 – 消費電力と使用率の関係をプロットしたこのグラフでは、直線が実際の電力の測定値、破線が XPE の推定値を表します。

A Generation Ahead

業界初、ASIC クラスのプログラマブル アーキテクチャ



UltraSCALE™ Architecture

詳細はこちら

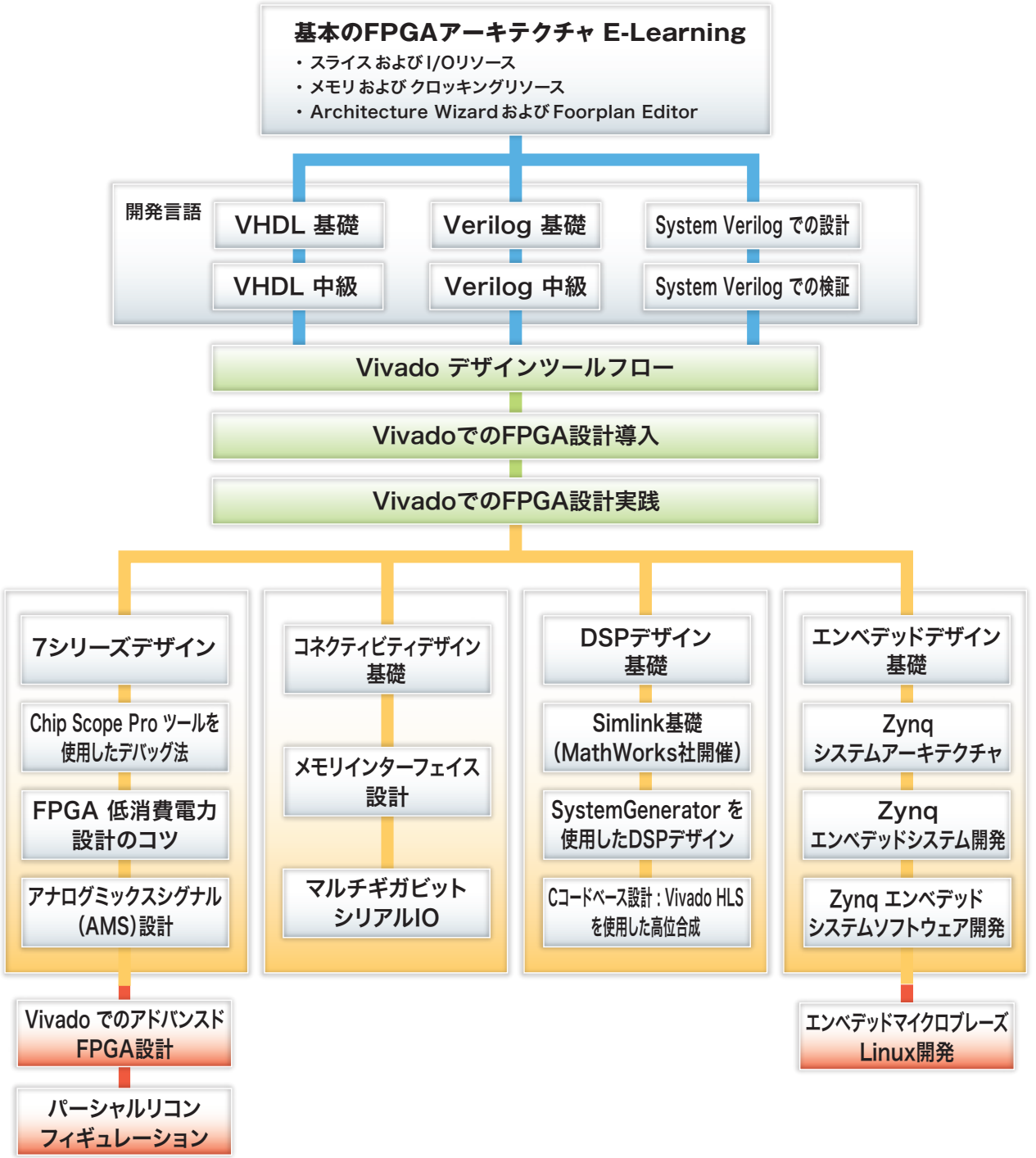
XILINX®
ALL PROGRAMMABLE

ザイリンクス株式会社
製品のお問い合わせは下記の販売代理店へどうぞ

■東京エレクトロン デバイス(株) TEL(045)443-4016 x2web@teldevice.co.jp ■アヴネット・インターニックス(株) TEL(03)5792-8210 EVAL-KITS-JP@avnet.com
■(株)PALTEK TEL(045)477-2005 nfo_pal@paltek.co.jp ■新光商事(株) TEL(03)6361-8086 X-Pro@shinko-sj.co.jp

ザイリンクス トレーニング カリキュラムパス

XILINX



ザイリンクス販売代理店 / 認定トレーニングプロバイダ オリジナル トレーニング

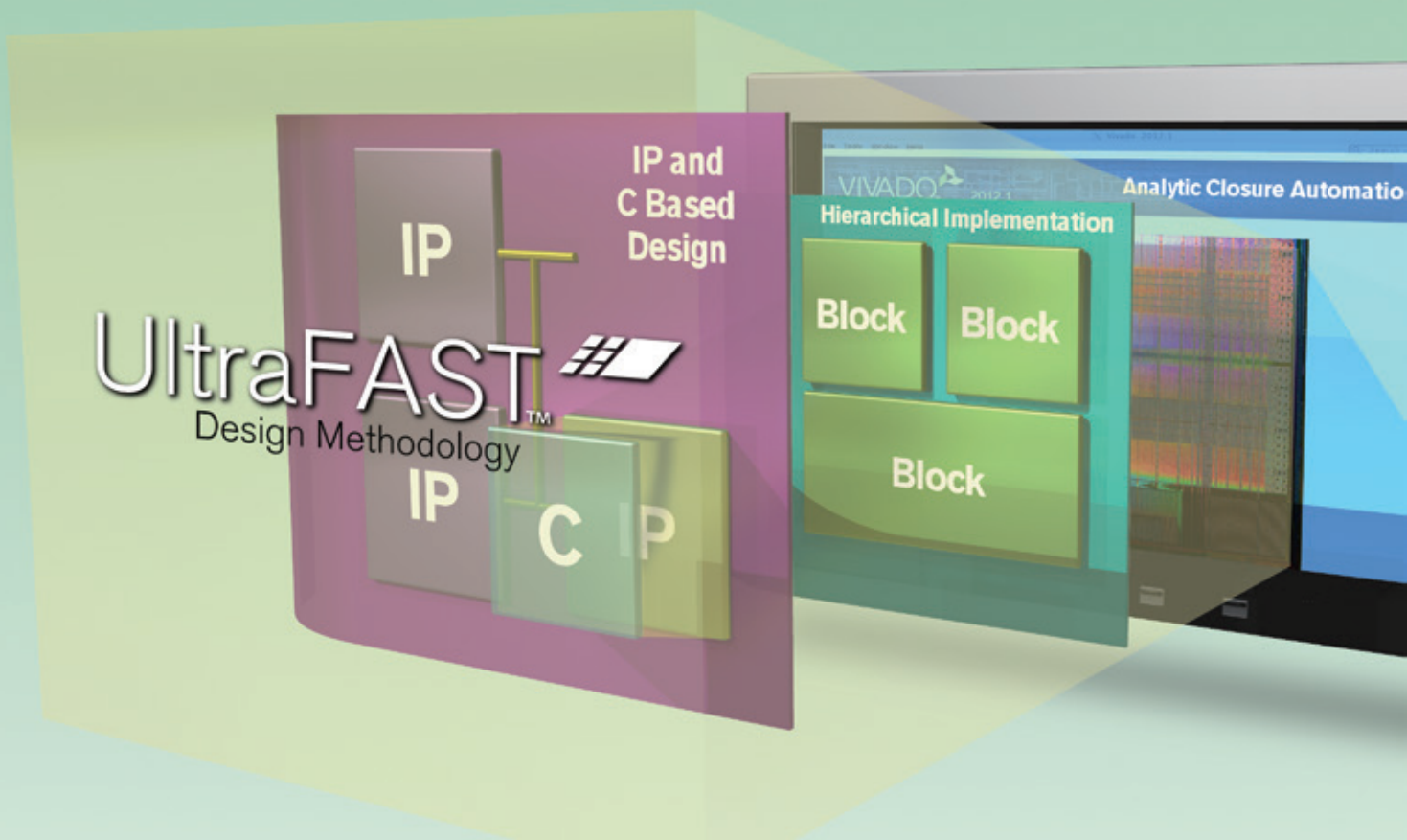
オリジナルトレーニングの内容およびスケジュールは、各社の Web サイトをご覧ください。

AVNET Internix	アヴネット・インターニックス	avnetinternix.co.jp/training.aspx
新光商事 Shinko Shoji Co.,Ltd.	新光商事	xilinx.shinko-sj.co.jp/training/index.html
東京エレクトロン デバイス	東京エレクトロンデバイス	ppg.teldevice.co.jp/
PALTEK	パルテック	www.paltek.co.jp/seminar/index.htm
hdLab	エッチ・ディー・ラボ	www.hdlab.co.jp/web/x500x/

詳細とご登録はこちらから **Japan.xilinx.com/training/**

Xilinx Introduces

Vivado® Design Suite 向け
UltraFast™ 設計手法



ザイリンクス UltraFast 設計手法は迅速で予測可能な設計サイクルを可能にします。

VIVADO

XILINX
ALL PROGRAMMABLE™

ザイリンクス株式会社
製品のお問い合わせは下記の販売代理店へどうぞ
■東京エレクトロン デバイス(株) TEL(045)443-4016 x2web@teldevice.co.jp ■アヴネット・インターニックス(株) TEL(03)5792-8210 EVAL-KITS-JP@avnet.com
■(株)PALTEK TEL(045)477-2005 nfo_pal@paltek.co.jp ■新光商事(株) TEL(03)6361-8086 X-Pro@shinko-sj.co.jp

詳細はこちら： japan.xilinx.com/ultrafast