90 号 2015

# Xcell journal

SOLUTIONS FOR A PROGRAMMABLE WORLD

## 1 ワット当たり性能を 2 倍～5 倍に向上させる 16nm UltraScale+ デバイス

60GHz ミリメートル波 バックホール リンクによる 大容量通信の実現

2 つのコアを利用して Zynq SoC の性能を 最大限に引き出す方法

ザイリンクス FPGA に PetaLinux を移植する方法

ザイリンクス FPGA を使用したソーラー オービターのオンボード データ処理

ページ 12

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 90 号

2015 年 4 月 13 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



# Get Ready for More Innovations from Xilinx
# ザイリンクスが可能にする新たなイノベーション

2011 年にザイリンクスが業界初の All Programmable SoC の発売を開始して以来、ユーザーの皆様は、ますます増え続ける市場で多様な革新的製品を開発してきました。自動車、産業用機器、科学技術計算、有線 / 無線通信、航空宇宙、検査 / 測定機器、放送、家電製品などの多彩な市場で、これまでに Zynq® SoC を活用した多くのイノベーションが生まれてきましたし、これからも生み出されていくことでしょう。過去数年間の Xcell Journal をお読みになり、新しい Xcell Daily Blog をご覧になった皆様は、デバイスの用途に関連する記事の割合が増えていることにお気付きのことと思います。

Zynq SoC に関連するすべての記事に共通するテーマは、ARM® デュアルコア Cortex™-A9 MPCore プロセッサと 7 シリーズ FPGA ロジックを単に 1 つのデバイスに統合し相互接続することで実現される、Zynq SoC の 驚異的なシステム性能です。Zynq SoC は、プロセッシング システムとプログラマブル ロジックを接続する 3,000 を超えるインターコネクトにより、ASSP/ASIC + FPGA の 2 チップ デバイスでは扱いきれない高性能を実現します。ディスクリートの FPGA と ASSP の外部 I/O では、チップ間の高速通信を処理するには不十分です。プロセッシング システムとプログラマブル ロジックの統合は、消費電力 ( および BOM コスト ) の削減というもう 1 つのメリットをもたらします。2 チップを 1 チップで置き換えれば、システムに必要な電力回路も削減されるからです。世界中の一流業界誌のイノベーション賞を受賞してきた Zynq SoC は、過去 4 年間でそれに見合う価値を実証してきました。

本号で、来年初頭に発売予定の次世代 All Programmable SoC である Zynq UltraScale+™ MPSoC のオンチップ機能の詳細がついに発表されるのは喜ばしいことです。このデバイスの詳細と、新たに公開されたその他の 16nm UltraScale+ 製品については、カバー ストーリーをお読みください。第 1 世代の Zynq SoC から得られた教訓、ユーザーからのご意見、顧客の製品ロードマップの分析を活かしてザイリンクスが開発した All Programmable MPSoC は、注目を集めた第 1 世代 Zynq SoC に比べてシステム集積度と 1 ワット当たりのシステム性能が飛躍的に向上しています。

カバー ストーリーでは、新しい UltraScale+ ポートフォリオのすべてのデバイス (FPGA、3D IC、MPSoC) で、TSMC 社の 16nm FFT+ プロセス技術の採用により、どのように 1 ワット当たりのシステム性能が前世代のシステムの 2 倍以上に向上しているかについて説明します。さらに、ザイリンクスがほとんどのデバイスに採用している UltraRAM と呼ばれる新しい大容量メモリと、SmartConnect と呼ばれる新しいシステム レベルのインターコネクト テクノロジも、1 ワット当たりの性能を向上させています。

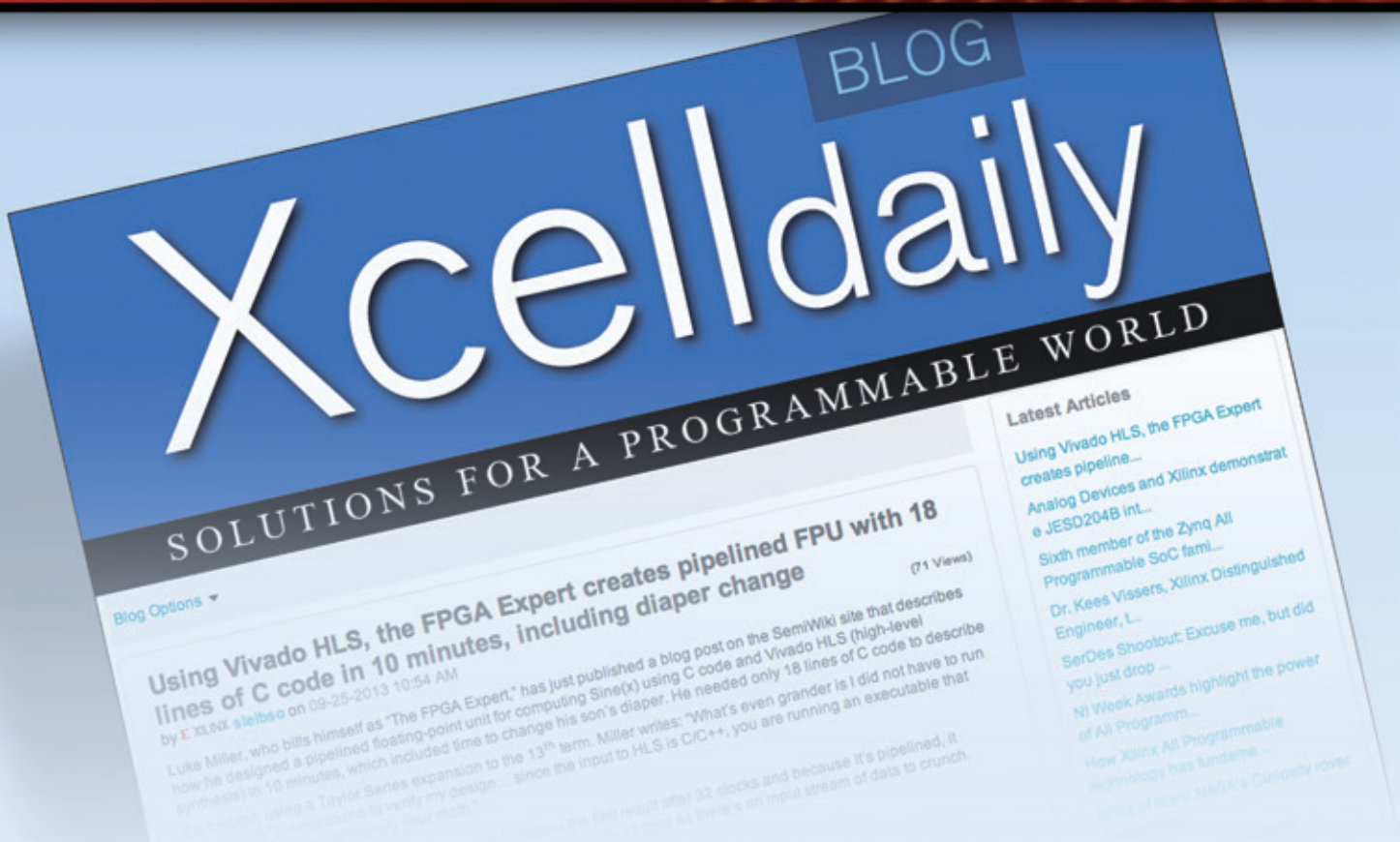
Zynq UltraScale+ MPSoC をシステムに採用すれば、1 ワット当たりのシステム性能は最も大きく向上します。Zynq MPSoC は全機能搭載型の All Programmable SoC であり、クワッドコア 64 ビット APU、デュアルコア RPU、グラフィックス プロセッサに加え、多数のペリフェラル、セキュリティ機能、電力管理機能をすべてワンチップに統合した製品です。Zynq MPSoC システムの 1 ワット当たりの性能は、28nm Zynq SoC システムの 5 倍に達します。

これまで Zynq SoC を使って数々の素晴らしいシステムを開発してきた皆様が、今度は Zynq UltraScale+ MPSoC を使ってどのような製品を開発されるのかを心待ちにしております。ザイリンクスのこの注目すべきデバイスが発売されたら、ぜひ本誌に寄稿して、各社の開発者と設計体験を共有していただければ幸いです。

Mike Santarini
発行人

長らく Xcell Journal の広告担当 兼 クリエイティブ ディレクターを勤めた Dan Teie 氏が 2015 年 1 月に逝去されました。氏は闘志あふれるスポーツマン、冒険家であるとともに紳士であり、ご家族や友人への深い愛情をもって最後まで情熱的に生きた素晴らしい人でした。ここに謹んでお悔やみを申し上げます。

# 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）**

- *A Double-Barreled Way to Get the Most from Your Zynq SoC*
- *New video shows successful 400G and 25G Ethernet interoperability demos at last week's OFC 2015*
- *How do you boost the floating-point performance of a quad-core Intel Core i7 PC by 1.7x? Add Zynq*
- *How to cut through the thicket of multiple interface standards in your Broadcast and Pro A/V designs*
- *Testing 10Gbps Optical Modules: Simplify Your "Hot" Testing with Xilinx's Zynq SoC*

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

XILINX
ALL PROGRAMMABLE™

## VIEWPOINTS



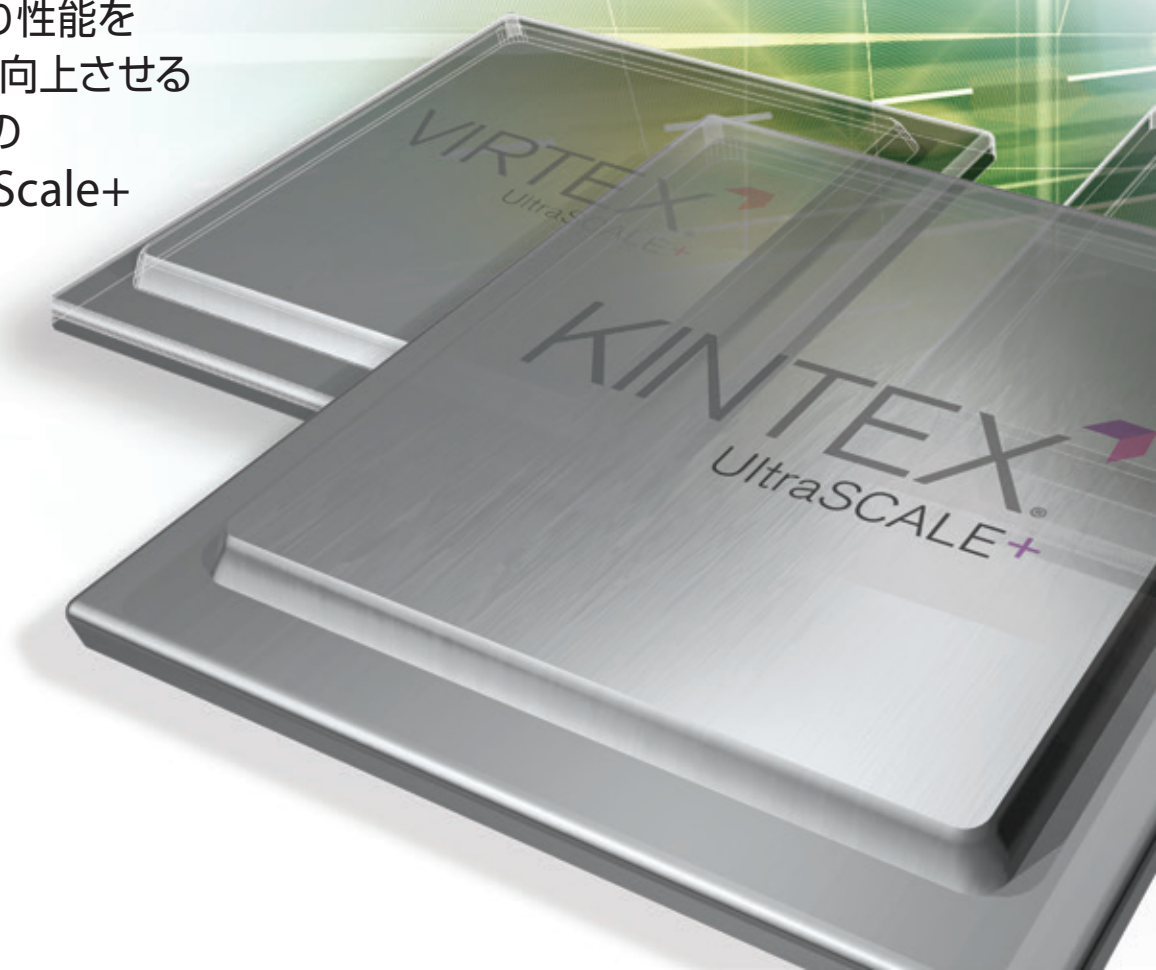

## XCELLENCE BY DESIGN APPLICATION FEATURES





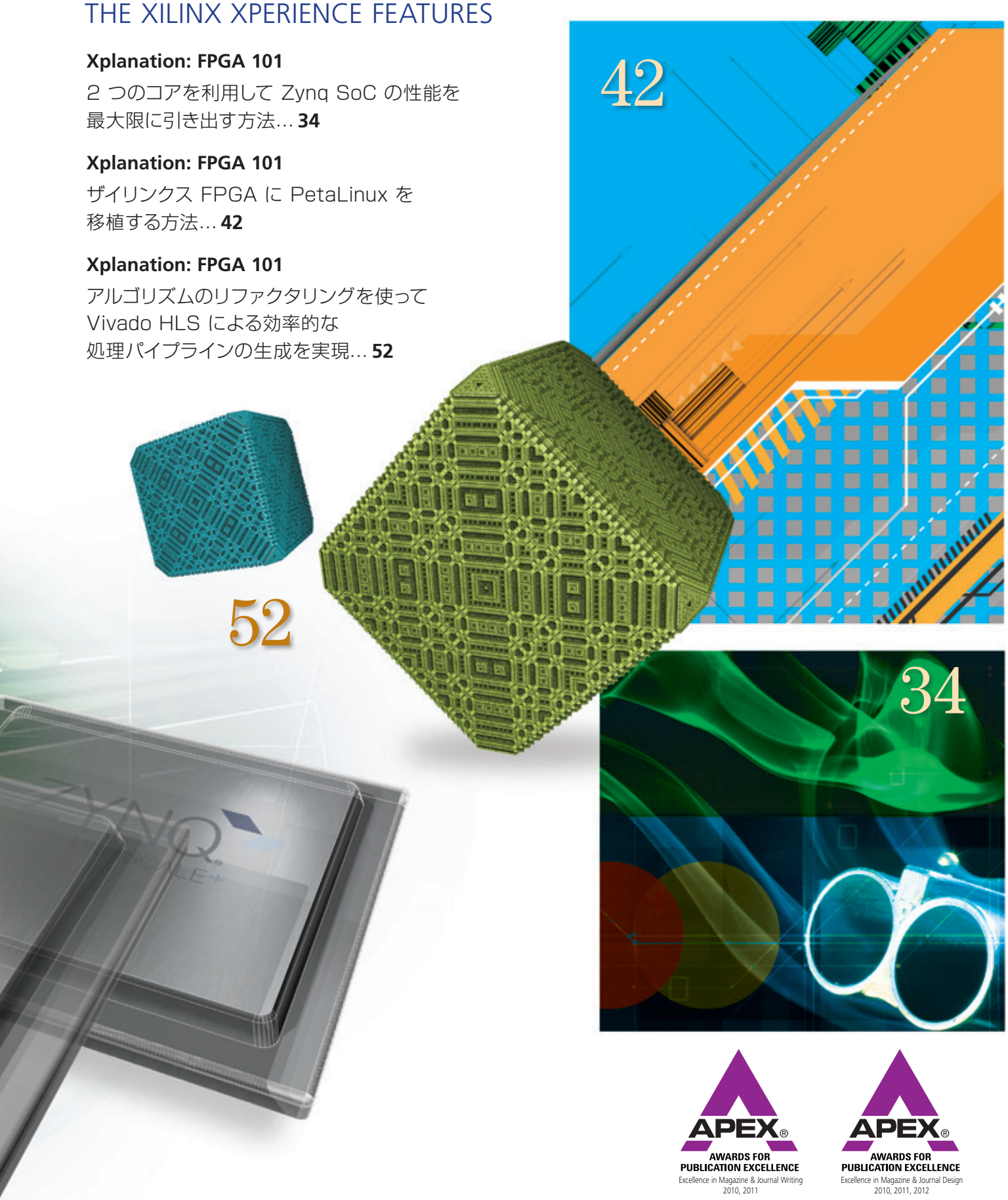
# Cover Story

4 1 ワット当たり性能を
2 倍～5 倍に向上させる
ザイリンクスの
16nm UltraScale+
デバイス



# THE XILINX XPERIENCE FEATURES





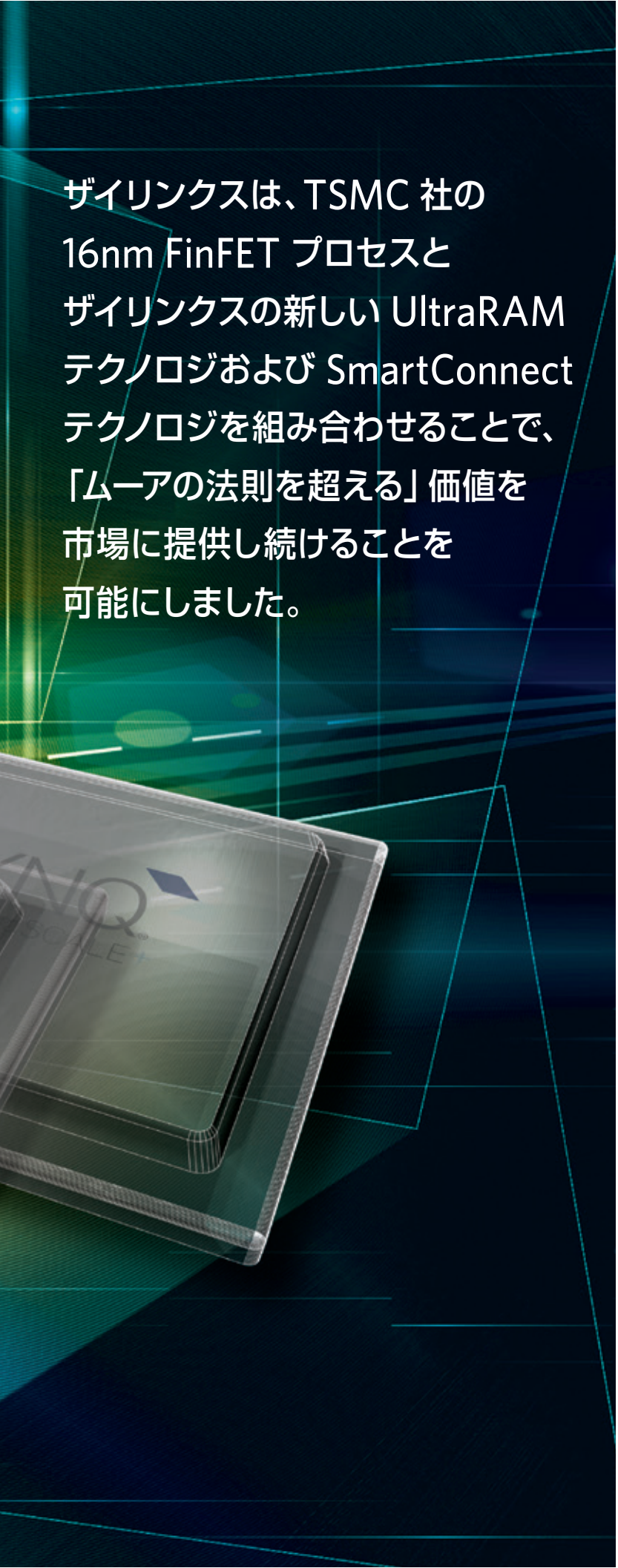

APEX® AWARDS FOR PUBLICATION EXCELLENCE
Excellence in Magazine & Journal Writing
2010, 2011

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

# Xilinx 16nm UltraScale+ Devices Yield 2-5X Performance/Watt Advantage

# 1ワット当たり性能を2倍～5倍に向上させるザイリンクスの16nm UltraScale+ デバイス

**Mike Santarini**
Publisher, Xcell Journal
Xilinx, Inc.
*mike.santarini@xilinx.com*




ザイリンクスは、一世代先をリードする 28nm 7 シリーズ All Programmable デバイス ファミリと業界初の 20nm UltraScale™ ポートフォリオに加え、このたび 16nm UltraScale+™ 製品ラインを公開しました。16nm UltraScale+ デバイス ポートフォリオを使って設計したシステムは、ザイリンクスの 28nm デバイスを使用した同等クラスのシステムに比べて、1 ワット当たり性能が 2 倍～5 倍に向上します。

この優れた 1 ワット当たり性能を生み出す主な要因は、TSMC 社の 16FF+ (16nm FinFET Plus) プロセス技術によるデバイスのインプリメンテーション、ザイリンクスのオンチップ UltraRAM メモリ、SmartConnect と呼ばれる革新的なシステム レベルのインターコネクト最適化テクノロジの 3 つです。

また、ザイリンクスは第 2 世代 Zynq® All Programmable SoC も発表しました。Zynq UltraScale マルチプロセッシング SoC (MPSoC) は、クワッドコア 64 ビット ARM® Cortex™ -A53 アプリケーション プロセッサ、32 ビット ARM Cortex -R5 リアルタイム プロセッサ、ARM Mali-400MP グラフィックス プロセッサと、16nm FPGA ロジック (UltraRAM 搭載)、多数のペリフェラル、セキュリティ機能および信頼性機能、革新的な電力制御テクノロジを 1 つのデバイスに統合した製品です。新しい Zynq UltraScale+ MPSoC を使えば、28nm Zynq SoC を使って設計されたシステムに比べて、1 ワット当たり性能が 5 倍に向上したシステムを開発できます。

## FinFET による UltraScale ポートフォリオのシステム価値の向上

ザイリンクスのシリコン製品管理およびマーケティング担当シニア ディレクタである Dave Myron は、「16nm UltraScale+ ポートフォリオで、ザイリンクスは従来のムーアの法則の先を行くシステム価値を創出します。ザイリンクスは、LTE Advanced や初期の 5G ワイヤレス、テラビットのワイヤード通信、先進運転支援システム (ADAS)、産業用 IoT ( モノのインターネット ) アプリケーションなど、広範囲にわたる次世代

アプリケーションに対応します。UltraScale+ ポートフォリオにより、お客様は各社の市場で競合他社の一世代先をリードし、優れたイノベーションを実現できます」と述べています。

UltraScale 世代の製品は、TSMC 社の 20nm プレーナ プロセス ( 既に発売中 ) と 今回の TSMC 社の 16FF+ プロセス (2015 年第 4 四半期に発売予定 ) という 2 つのプロセス ノードのデバイスが並行して提供されます。ザイリンクスは、16nm UltraScale+ 版の Virtex® FPGA および 3D IC ファミリ、Kintex® FPGA ファミリ、そして新しい Zynq UltraScale+ MPSoC を提供する予定です。

新製品採用およびソリューション マーケティング担当ディレクタの Mark Moran によると、2013 年にザイリンクスは TSMC 社の 16FF+ プロセスを待たずに 20nm プロセスの UltraScale の発売を開始しました。これは、一部のアプリケーション空間で、( 基本的に性能と容量が 28nm よりも高い ) 20nm デバイスを 1 年半早く提供する必要があったためです。

Moran は次のように述べています。「ザイリンクスのポートフォリオ全体が市場のニーズを念頭に置いて設計されています。UltraScale+ で実現される 1 ランク上の 1 ワット当たり性能を必要としない市場および最終アプリケーションの次世代製品には、20nm UltraScale アーキテクチャ デバイスの機能がより適しています。私たちは 16nm がすぐ後に控えていることを意識しながら 20nm FinFET を開発しました。そのため、16nm で市場ニーズに応える 1 ランク上の性能と価値を実現できるように、16nm の基盤になりそうなさまざまなアーキテクチャの改良を 20nm にインプリメントしました。お客様は、現在供給されている 20nm デバイス上で一歩先を行く形で開発を進められるため、16nm UltraScale+ デバイスが利用可能になった時点で、デザインを迅速に移植し、短期間で市場に投入できます」

Myron によると、Virtex UltraScale+ デバイスの多くは 20nm Virtex UltraScale デバイスとピン互換性があり、1 ワット当たり性能の向上を必要とするデザインへの移行が容易に行えます。

「ツール側から見ると、20nm UltraScale デバイスと 16nm UltraScale+ デバイスはほぼ同じに見えます。したがって、16nm UltraScale+ デバイスを使用すると、1 ワット当たり性能が向上するため、性能と消費電力の目標値の達成が容易になるというメリットもあります」と Myron は述べています。

Myron によると、UltraScale+ FPGA および 3D IC では、28nm 7 シリーズ デバイスに比べて 1 ワット当たり性能が 2 倍以上に向上します。一方、Zynq UltraScale+ MPSoC は、統合型ヘテロジニアス プロセッシング機能の追加により、28nm Zynq SoC で構築された同等クラスのシステムに比べて、1 ワット当たりのシステム性能が 5 倍以上に向上します ( 図 1)。

## TSMC 社の 16FF+ プロセスによる 1 ワット当たり性能の向上

ザイリンクスのデバイスは、16nm FinFET プロセスへの移行のみによって、28nm 7 シリーズ デバイスに比べて 1 ワット当たり性能が 2 倍に向上しました。Myron は次のように述べています。「TSMC 社の 16FF+ は極めて効率的なプロセス技術であり、プレーナ トランジスタのインプリメントに使用

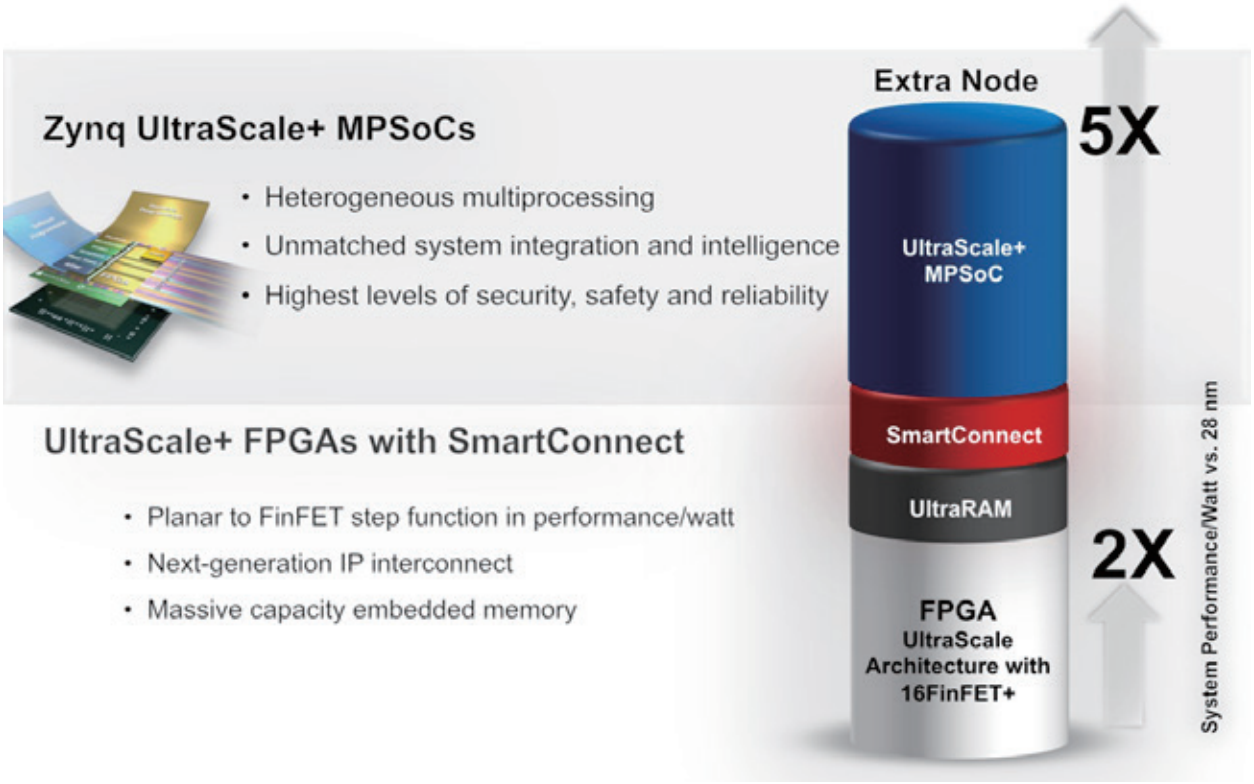



図 1 – ザイリンクスの 16nm UltraScale+ FPGA および Zynq UltraScale+ MPSoC は、設計チームに 1 ランク上の価値を提供します。

される従来のシリコン プロセスに関連するトランジスタ漏れ電力は事実上ゼロになります。さらに、ザイリンクスは TSMC 社との入念な共同作業により、新しいプロセス技術を十分に利用できるように UltraScale+ デバイスを改良しました。少なくとも ( 新しいプロセス技術のイノベーションだけで )、UltraScale+ デザインは、28nm 7 シリーズ デバイスでインプリメントされたデザインに比べて、1 ワット当たり性能が 2 倍以上に向上する見込みです」

ザイリンクスの 20nm UltraScale アーキテクチャの詳細と、プレーナ トランジスタ プロセスと比べた FinFET の利点については、Xcell Journal 日本語版 84 号のカバーストーリー (ページ 6) を参照してください。

ザイリンクスは、UltraScale+ ファミリで業界初の 3D-on-3D デバイスも提供しています。これは TSMC 社の 16FF+ 3D トランジスタ テクノロジ上にインプリメントされる、第 3 世代スタックド シリコン インターコネクト 3D IC です。

Myron によると、数々の受賞歴がある 7 シリーズ 3D IC は、1 つの IC 上に複数のダイを実装することにより、ムーアの法則の性能と容量の限界を乗り越えました。

「ホモジニアス 3D IC では、ザイリンクスはムーアの法則の容量の限界を打ち破り、最大の 28nm モノリシック FPGA が提供するものに比べて 2 倍の容量をもつデバイスを実現しました。さらに、最初のヘテロジニアス デバイスにおいて、ザイリンクスは FPGA ダイと高速トランシーバー ダイを組み合わせて、28nm モノリシック デバイスでは不可能なシステム性能と帯域幅を実現しました。UltraScale+ 3D IC により、ザイリンクスはムーアの法則を超える容量と性能を提供し続けます」と Myron は述べています。

## UltraRAM による 1 ワット当たり性能の向上

Myron によると、多くの UltraScale+ デザインは、UltraRAM と呼ばれる新しい大容量オンチップ メモリにより、28nm デバイスを使ったデザインに比べて 1 ワット当たり性能がさらに向上します。ザイリンクスは、大部分の UltraScale+ デバイスに UltraRAM を追加しています。

Myron は次のように述べています。「基本的に、現状では LUT RAM/ 分散 RAM やブロック RAM などのオンチップ メモリと、DDR やオフチップ SRAM などのオフチップ メモリの間のギャップが広がっています。異なる種類のメモリを必要とする、プロセッサに負荷の高いアプリケーションが増えています。特に大規模で複雑なデザインを設計する場合は、高速メモリをオンチップに搭載する必要性が高まります。粒度の細かいブロック RAM では、容量が小さすぎます。一方、メモリをオフチップに配置すると、消費電力の増大、I/O の複雑化、BOM コストの増加を招きます」

これらの理由で、ザイリンクスは UltraRAM を開発しました。「私たちはオンチップ メモリの階層を 1 レベル増やすとともに、デザインに大容量メモリ ブロックを簡単にインプリメントできる機能を追加しました。これにより、設計者は適切なサイズのメモリをオンチップに簡単に配置し、タイミングを保証できます」と Myron は述べています。

LUT/ 分散 RAM は、設計者がビットおよびキロビット単位で RAM を追加できます。BRAM は、数十メガビット単位でメモリ ブロックを追加できます。UltraRAM は、UltraScale+ デバイスを使用する設計者が、数百メガビット単位でオンチップ SRAM ブロックをインプリメントできます ( 図 2)。これにより、設計者は、オフチップ RAM (SRAM、RLDRAM、TCAM) をあまり必要としない、より高性能で電力効率の高いシステムの開発が可能となります。その結果、BOM コストが削減されます。最大の UltraScale+ デバイスである VU13P には、432 メガビットの UltraRAM が搭載されます。

## SmartConnect による 1 ワット当たり性能の向上

もう 1 つの新しいテクノロジである SmartConnect は、UltraScale+ デザインの 1 ワット

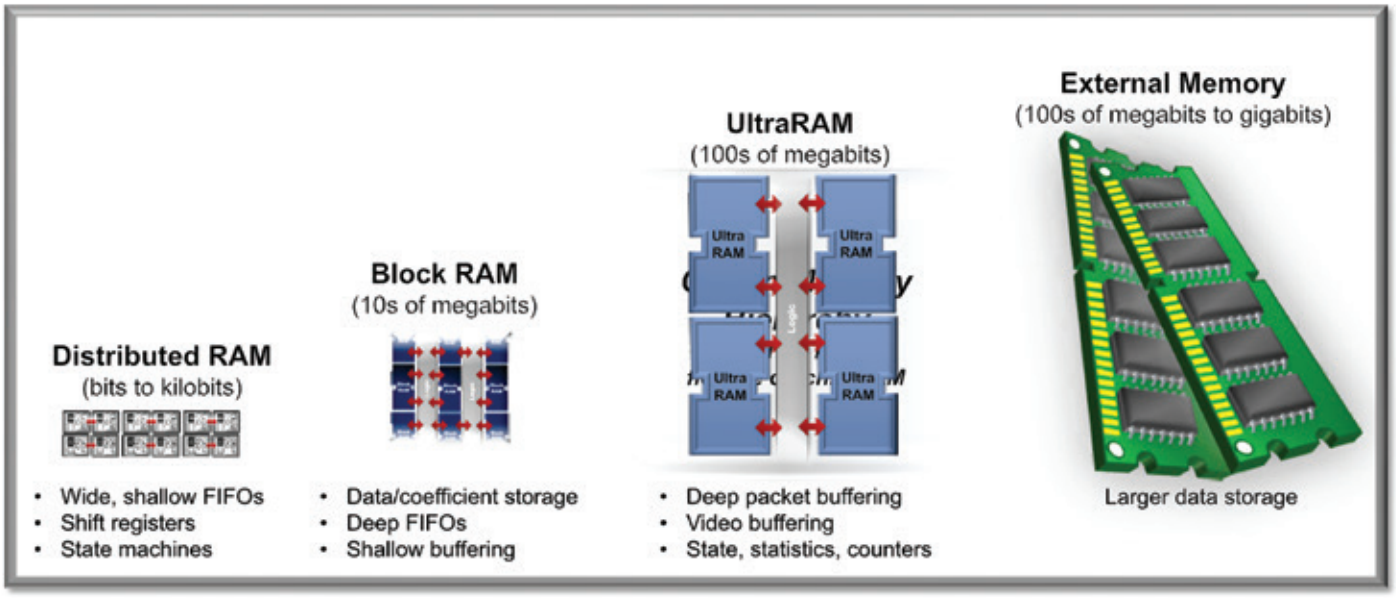



図 2 – UltraRAM はオンチップ メモリとオフチップ メモリのギャップを埋め、大容量ローカル メモリ ブロックを使用した高性能、低消費電力のシステム開発を可能にします。

当たり性能をさらに向上させます。

「SmartConnect は、ツールとハードウェアの協調最適化により、複雑化したデザインでも簡単にインプリメントできる高度な手法です」と Myron は述べています。

従来は、技術者がより多くの IP ブロックをデザインに詰め込もうとすると、(電力と面積の要件に関して) オーバーヘッドが増加していました。Myron によると、SmartConnect では、システム レベルからデザイン全体を検討する最適化機能が Vivado® Design Suite に追加されました。SmartConnect は、AXI インターコネクトの新たな拡張機能と 16nm UltraScale+ シリコンを利用して、最も効率的なインターコネクト トポロジを見つけ出し、最小限の面積と最大限のパフォーマンスを達成します。

「16nm UltraScale+ デバイスは、配線レベルだけでなく、このような高レベルのプロトコルで効率が向上します。つまり、16nm FinFET の利点の上に、1 ワット当たり性能を引き上げる機能がさらに加わるのです」と Myron は述べています。

図 3 に、8 個のビデオ処理エンジンがすべてプロセッサおよびメモリとインターフェイスする実際のデザインを示します。Myron は次のように述べています。「驚くべきことに、このような実際のデザインでは、インターコネクト ロジックが実際にデザインの総面積の 2 分の 1 を使用することがあります。このために消費電力が増えるだけでなく、周波数も制限されます。SmartConnect は、インターコネクト ブロックを自動的に再構築し、同じ性能で消費電力を 20% 削減します」

## 16nm UltraScale+ FPGA のベンチマーク

FPGA デザインのシナリオで 1 ワット当たり性能の向上を示すために、次の例を考えます。48 ポート無線 CPRI 圧縮 / ベースバンド ハードウェア アクセラレータを 28nm Virtex-7 FPGA にインプリメントした場合、消費電力は 56W (ワット) です (図 4)。同じ性能で動作する同じデザインを 16nm Virtex UltraScale+ FPGA にインプリメントすると、消費電力は 27W となり (55% 削減)、1 ワット当たり性能は 2.1 倍に向上します。UltraRAM と SmartConnect によって 1 ワット当たり性能がさらに向上するため、Virtex UltraScale+ 版のデザインの 1 ワット当たり性能は、28nm Virtex-7 FPGA インプリメンテーションに比べて 2.7 倍以上に向上し、消費電力は 63% 削減されます。

同様に、FPGA の電力バジェットが 15W の画像処理 PCI モジュールを 28nm Virtex-7 にインプリメントした場合、525Ops (Operations Per Second) の性能で動作します。比較のために同じデザインを 16nm UltraScale にインプリメントすると、モジュールは 1,255Ops で動作し、1 ワット当たり性能は 2.4 倍に向上します。UltraRAM と

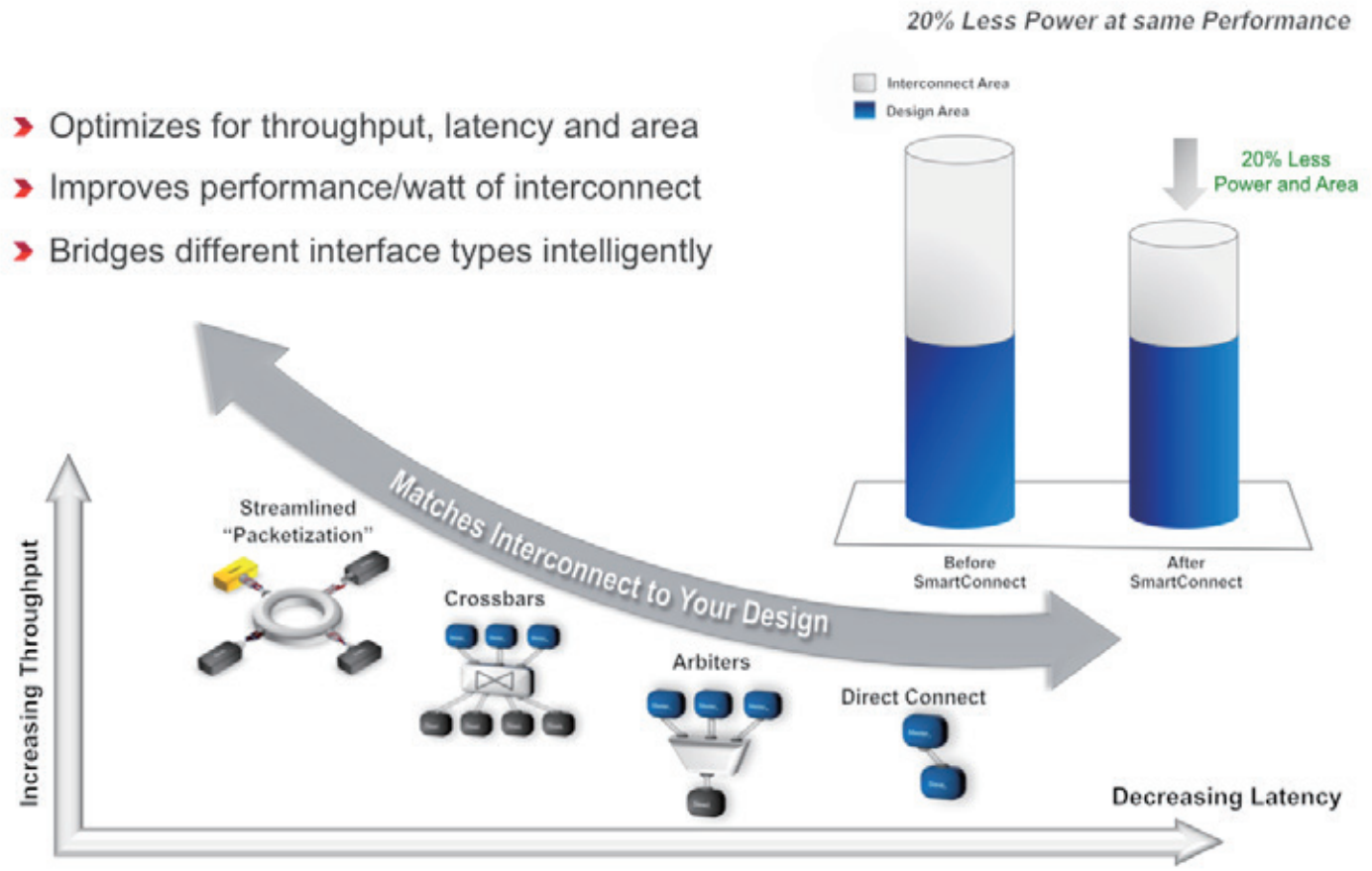



図 3 – SmartConnect テクノロジは必要なインターコネクトの面積を最大 20% 削減します。これにより、同じ性能レベルで消費電力は 20% 削減されます。

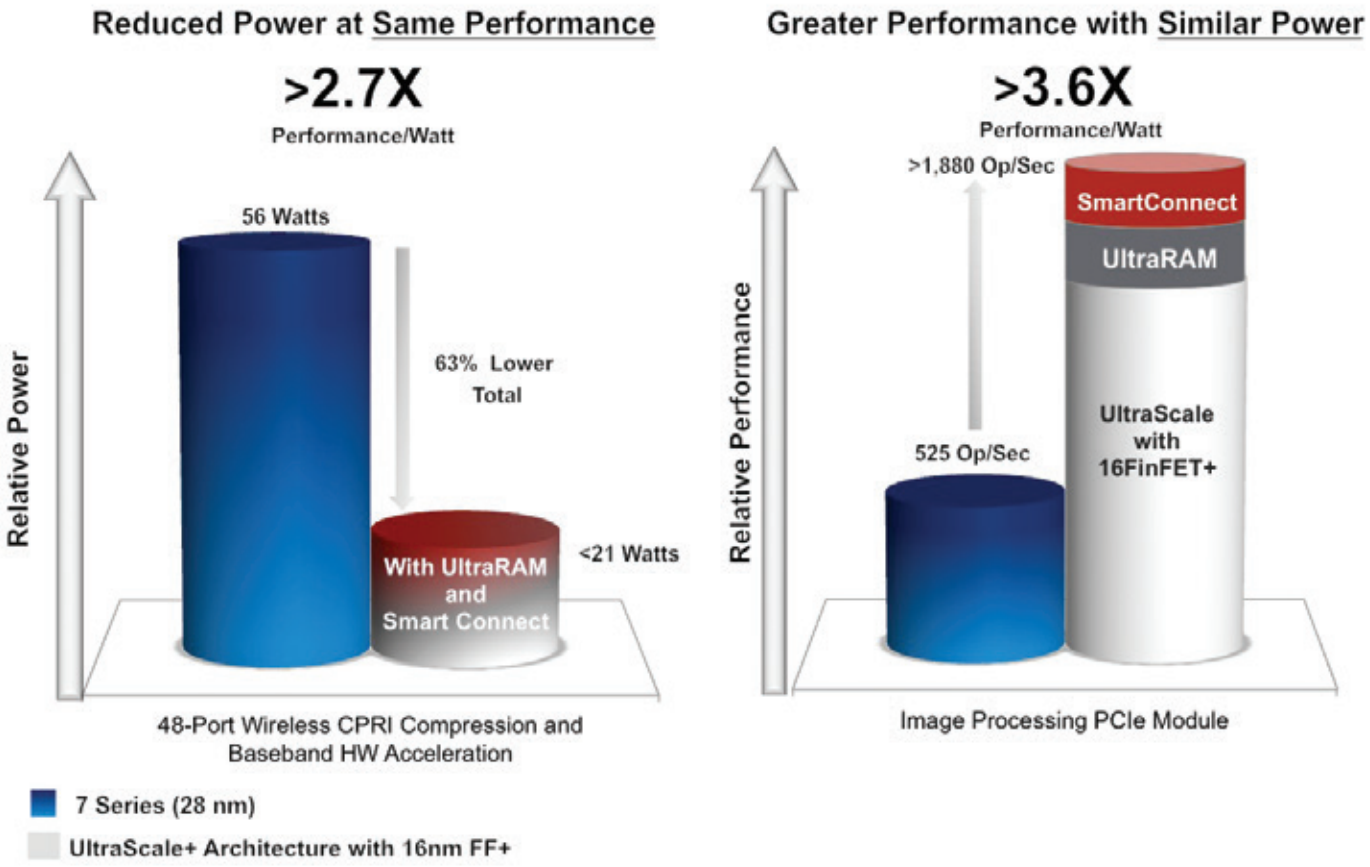



図 4 – 電力バジェットを増やさずにデザインを高速化したい場合や、性能を落とさずに消費電力を大幅に削減したい場合、16nm UltraScale+ デバイスの優れた 1 ワット当たり性能が効果的です。

SmartConnect による向上分を加算すると、Virtex UltraScale+ 版の 1 ワット当たり性能は、28nm Virtex-7 FPGA インプリメンテーションの 3.6 倍以上に向上します。

## Zynq UltraScale+ MPSoC は 1 ワット当たり性能が 5 倍以上に向上

第 2 世代 All Programmable SoC は、TSMC 社の 20nm プロセスでインプリメントすることも可能でしたが、ザイリンクスは TSMC 社の 16nm FinFET プロセスが利用可能になるのを待つことにしました。16nm Zynq UltraScale+ MPSoC は、デバイスのヘテロジニアスなマルチプロセッシング機能セットと、16nm UltraScale アーキテクチャによる 1 ワット当たり性能の向上を組み合わせて、非常に効率的な中央演算処理システム コントローラとして機能します。16nm Zynq UltraScale+ MPSoC は、28nm Zynq SoC の 5 倍を超えるパフォーマンスを実現します。

2014 年、ザイリンクスは UltraScale MPSoC アーキテクチャの「タスクごとに最適なエンジン」の利用モデルを公開しましたが、Zynq UltraScale+ MPSoC デバイスに搭載されるコアの詳細については発表を控えました。今回ザイリンクスは、Zynq UltraScale+ MPSoC の全機能セットを公開しました ( 図 5)。

オリジナルの 28nm Zynq SoC の最大の価値は、ARM プロセッシング システムとプログラマブル ロジックを 1 つのデバイスに統合したことです。( 約 84Gbps のピーク帯域幅で動作する) 3,000 以上のインターコネクトが、Zynq SoC のプロセッシング システム (PS) とプログラマブル ロジック (PL) ブロックに接続します。PS と PL がこのように緊密に接続されるため、1 つの FPGA と別個の ASSP で構成される 2 チップ システム アーキテクチャでは実現できない、高いスループットとパフォーマンスを提供できます。

16nm UltraScale+ MPSoC では、デバイスに 500Gbps のピーク帯域幅で動作する 6,000 以上のインターコネクトを付与することにより、プロセッシング システムとプログラマブル ロジック間の性能が飛躍的に向上しました。ザイリンクスの All Programmable SoC 製品マーケティングおよび管理担当ディレクタの Barrie Mullins は「Zynq UltraScale+ MPSoC のプロセッシング システムとロジック システム間の接続は、28nm Zynq SoC の 6 倍に高速化されます。これにより、このデバイスは ASSP + FPGA の 2 チップ アーキテクチャをはるかに超えるシステム性能を発揮します」と述べています。

Mullins によると、Zynq UltraScale+ MPSoC の中核となるのは、64 ビット クワッドコア ARM Cortex-A53 プロセッサです。このプロセッサは 28nm Zynq SoC のデュアルコア Cortex-A9 プロセッシング システムの 2 倍以上の性能を提供します。このアプリケーション処理システムは、ハードウェア仮想化と非対称プロセッシングに対応し、ARM の TrustZone® セキュリティ機能を

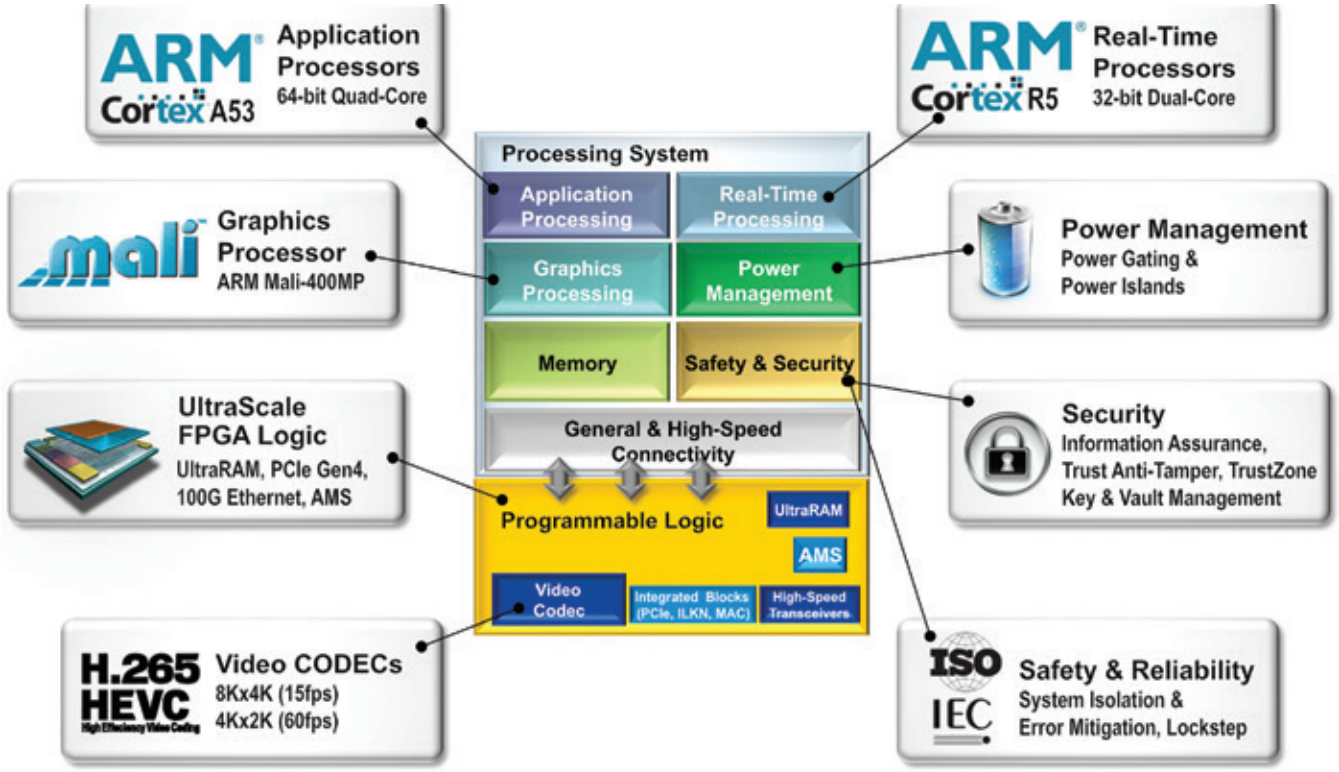



図 5 – 16nm Zynq UltraScale+ MPSoC は豊富な処理エンジンを搭載しており、設計チームはこれらを必要に合わせて調整して比類のないシステム性能を実現し、システムの価値を飛躍的に高められます。

完全にサポートします。

Zynq UltraScale+ MPSoC には、システムの決定論的演算用にデュアルコア ARM Cortex-R5 リアルタイム プロセッシング サブシステムも搭載されています。このリアルタイム プロセッサは、極めて高いレベルのスループット、安全性、信頼性が要求されるアプリケーションに必要なシステムの即応性を保証します。

Zynq UltraScale+ MPSoC には多数のグラフィックス専用エンジンが組み込まれており、処理能力がさらに向上します。ARM Mali™-400MP 専用グラフィックス アクセラレーション コアは、グラフィックスを多用する作業からメイン CPU を解放します。GPU を補完するため、H.265 ビデオ規格に対応するハード化されたビデオ圧縮 / 伸張用ビデオ コーデック コアがプログラマブル ロジック ブロックに追加され、8Kx4K (15 フレーム / 秒) および 4Kx2K (60 フレーム / 秒) を処理します。また DisplayPort ソース コアにより、システムに外部 DisplayPortP TX トランスミッター チップを使用せずに、ビデオ データのパケット化を高速で処理できます。

Zynq UltraScale+ MPSoC では、オンチップ メモリも大幅に強化されています。UltraScale+ 製品ファミリの最大のデバイスである Zynq UltraScale+ MPSoC のプログラマブル ロジックには、ブロック RAM に加えて UltraRAM を搭載しています。一方、Zynq UltraScale+ MPSoC のプロセッシング コアは、L1 キャッシュおよび L2 キャッシュを共有します。

Zynq UltraScale+ MPSoC には、広帯域の ECC 付き 72 ビット DDR インターフェイス コア (64 ビット + ECC 用 8 ビット) も搭載されます。このインターフェイスは大容量 32GB の DRAM をサポートし、DDR4 のデータ転送速度が最大 2,400Mbps に向上します。

Zynq UltraScale+ MPSoC の専用セキュリティ ユニットは、M2M 通信やコネクテッド制御アプリケーションで標準的に要求される、セキュア ブート、キー & ヴォールト管理、改ざん防止機能といった軍用クラスのセキュリティを実現します。さらに、Zynq UltraScale+ MPSoC のプログラマブル ロジック システムには、150G Interlaken、100G イーサネット MAC、PCIe® Gen4 用の統合型コネクティビティ ブロックも含まれます。オンボードの AMS (アナログ ミックスド シグナル) コアにより、設計チームは設計したシステムを System Monitor (システム モニター) でテストできます。

これだけの豊富な機能を備えた MPSoC では、アプリケーションが利用可能なエンジンをすべて使用することはおそらくありません。このため、Zynq UltraScale+ MPSoC は、極めて柔軟性の高い専用の電力管理ユニット (PMU) を搭載しています。このコアを利用して、(粗粒度および細粒度の) パワー ドメインおよびパワー アイランドを制御し、システムが使用しているプロセッシング ユニットにのみ電力を供給できます。さらに、このコアをダイナミック動作用にプログラムすれば、指定されたタスクに必要な機能のみを実行し、その後パワーダウンするようにシステムを設定できます。また、PMU は、信号およびエラーの検出と軽減、セーフステート モード、システムの隔離と保護など、多くの安全機能および信頼性機能を駆動します。

Myron によると、前述のように、16nm デバイスの 1 ワット当たり性能の機能にこれらのプロセッシング機能を追加することにより、Zynq UltraScale+ MPSoC でデザインをインプリメントすれば、28nm Zynq SoC に比べて平均 5 倍の 1 ワット当たり性能のメリットを享受できます。

## 16nm Zynq UltraScale+ MPSoC のベンチマーク

Zynq UltraScale+ MPSoC の優れた 1 ワット当たり性能を示すために、このデバイスの多くのアプリケーションの中から 3 種類のベンチマークを検討します。図は各種の処理エンジンの相違点を示すように色分けされています ( 図 6)。

フル 1080p ビデオを使ったビデオ会議システムの開発には、これまで 1 つの Zynq SoC と別個の H.264 ASSP を組み合わせて使用していましたが、現在では Zynq UltraScale+ MPSoC の利点を活用して、同じ電力バジェットの 1 個の Zynq UltraScale+ MPSoC に 4K x 2K UHD システムをインプリメントすることで、2 チップ システムの 5 倍の 1 ワット当たり性能が得られるようになりました。

SoC 製品ライン担当シニア マネージャーの Sumit Shah は、「1 個の Zynq SoC と 2 個の ASSP が必要だった警察・消防無線アプリケーションは、現在では 1 個の Zynq UltraScale+ MPSoC にデザイン全体をインプリメントできるようになりました。その結果、以前のコンフィギュレーションに比べてシステム消費電力は 47% 削減、パフォーマンスは 2.5 倍に向上し、1 ワット当たり性能は 4.8 倍に向上しました」と述べています。

同様に、Shah によると、従来 2 個の 28nm Zynq SoC にインプリメントされていた自動車用マルチカメラ運転支援システムは、1 個の Zynq UltraScale+ MPSoC に縮小されます。このワンチップ システムは、2 チップ デザインの 2.5 倍のパフォーマンスを発揮し、消費電力は 50% 削減されます。したがって、1 ワット当たり性能のトータル メリットは従来のインプリメンテーションに比べて 5 倍に向上します。

UltraScale+ ファミリのすべての製品には、早期から顧客の関与が進められています。ザイリンクスでは、最初のテープアウトと設計ツールのアーリー アクセス リリースを 2015 年第 2 四半期に予定しています。UltraScale+ デバイスのお客様への販売は、2015 年第 4 四半期からとなる見込みです。

16nm UltraScale ポートフォリオの優れた 1 ワット当たり性能の詳細は、http://japan.xilinx.com/ultrascale を参照してください。Zynq UltraScale+ MPSoC の詳細は、http://japan.xilinx.com/products/technology/ultrascale-mpsoc.html を参照してください。


図 6 – Zynq UltraScale+ MPSoC の広範囲にわたる処理ブロック、豊富なペリフェラル セット、16nm ロジック ブロックを利用して、28nm Zynq SoC を使用したデザインに比べて 1 ワット当たり性能が 5 倍に向上した革新的なシステムを開発できます。

# Solar Orbiter Will Process Data Onboard Using Xilinx FPGAs

# ザイリンクス FPGA を使用したソーラー オービターのオンボード データ処理

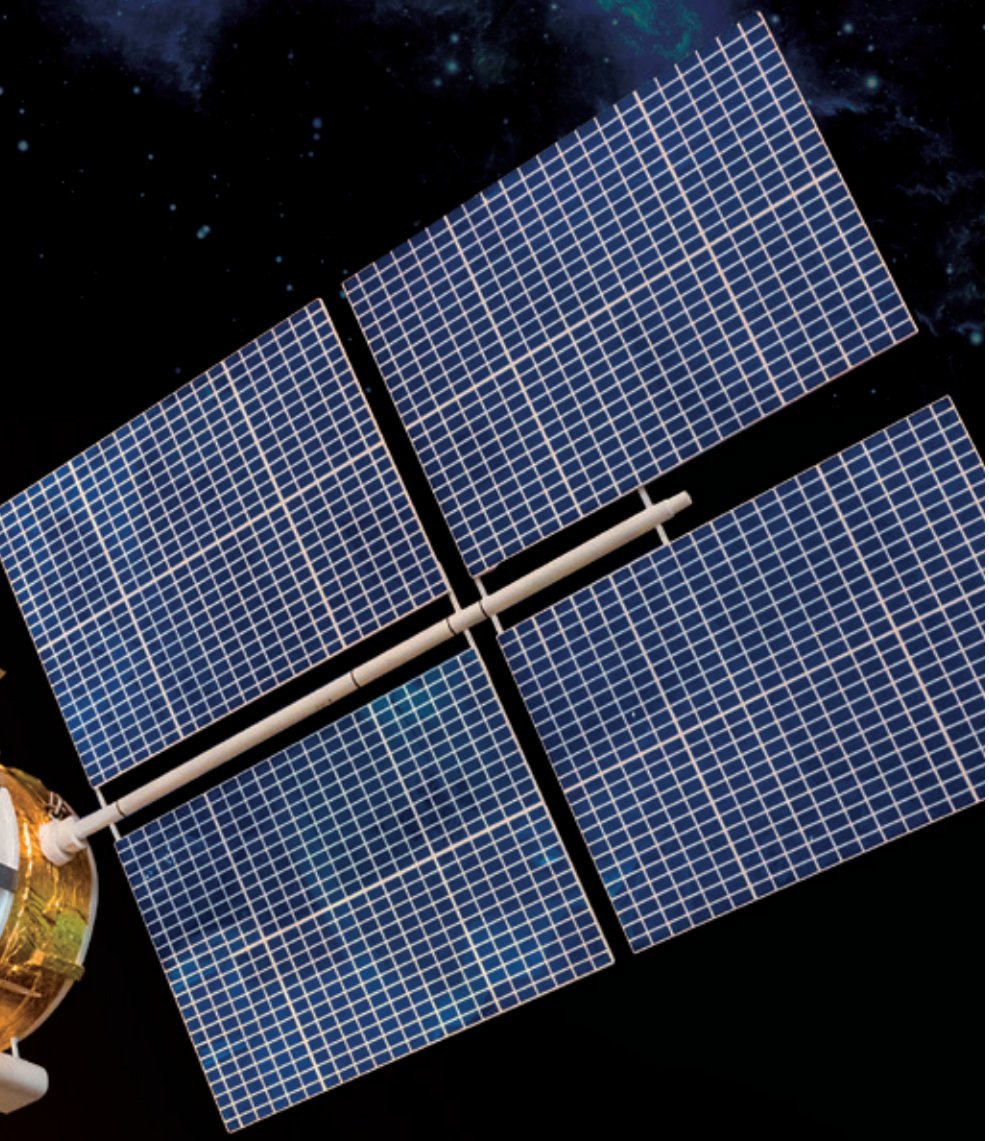
**Tobias Lange**
Design Engineer
Braunschweig University of Technology (Germany), Institute of Computer and Network Engineering (IDA)
*tobias.lange@tu-bs.de*

**Holger Michel**
Design Engineer
Braunschweig University of Technology, IDA

**Björn Fiethe**
Senior Engineer
Braunschweig University of Technology, IDA

**Harald Michalik**
Professor
Braunschweig University of Technology, IDA

# 航空宇宙グレード品の Virtex FPGA は、画像データの収集を高速化し、宇宙観測機器上のインフライト処理を実現します。



宇宙探査機に搭載された最先端のリモートセンシング機器は、大量の高解像度画像データを生成します。従来の地球観測ミッションでは、通常は地上で受信した後、その収集したデータを科学者が評価します。深宇宙探査ミッションでは高レートの画像データを処理する必要がありますが、テレメトリ レートは非常に限られています。

要求の厳しい例の 1 つとして、2017 年開始予定の欧州宇宙機関のソーラー オービター ミッションで科学ペイロードの一部に選ばれた偏光 · 日震撮像 (PHI) 装置が挙げられます。ドイツのマックス プランク太陽系研究所 (MPS) が主に開発したこの PHI 装置は、太陽光球内の連続体強度、磁場ベクトル、視線速度のマップを提供します。

大量のデータが収集されるけれどもダウンリンク性能が限られているため、科学パラメーターを宇宙探査機内のボード上で抽出できれば、データ量は大幅に削減されます。その結果、科学者たちは太陽光球をより詳細に分析できます。

このようなオンボード処理の要求に対応する魅力的なソリューションが、ゲート数の高いザイリンクス SRAM ベース FPGA です。筆者らの研究チームは、ドイツのブラウンシュバイク工科大学において、既にアクティブ宇宙観測ミッションにザイリンクス FPGA を活用してきました。筆者らは、金星探査機 Venus Express のモニタリング カメラ (VMC) や宇宙探査機 Dawn のフレーミング カメラ (DawnFC) の処理ユニットで古典的な画像データ圧縮処理用にザイリンクス FPGA を採用しましたが、いずれも数年間にわたって正常に動作しています。ソーラー オービターの PHI 処理ユニットには、2 つの航空宇宙グレード Virtex®-4 FPGA を使うことにしました。これらの FPGA はフライト中にリコンフィギュレーションされます。

この重要なミッションで FPGA がどのようにデータ収集を合理化するかについて詳しく説明する前に、PHI の機能とその動作について説明しましょう。

## ソーラー オービターの PHI

PHI 装置は、アクティブ ピクセル センサーから一連の画像を収集します。光路内のフィルター ホイールは、さまざまな波長と偏光の設定をこれらの画像に適用します。収集した画像データを前処理し ( ダーク フィールドおよびフラット

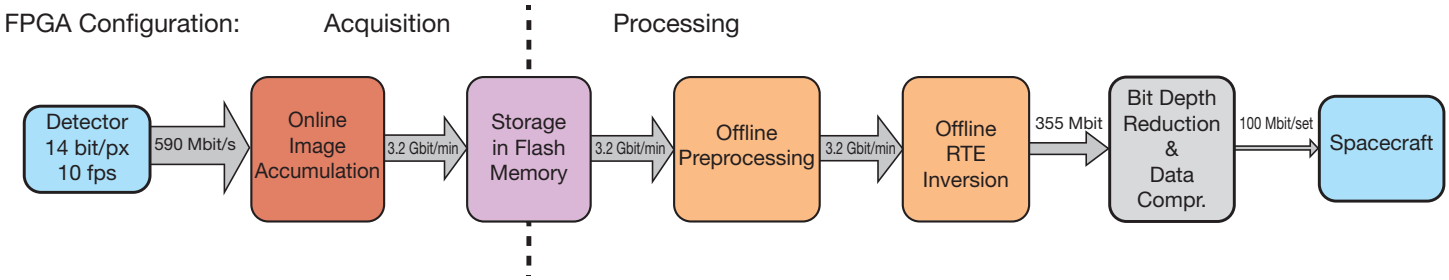



図 1 – 偏光・日震撮像 (PHI) データ処理パイプライン

フィールドの補正など)、放射伝達方程式 (RTE) の反転を実行すれば (この処理は演算量が多い)、ピクセル データから磁場ベクトルを計算できます。この手法と標準的なデータ圧縮を組み合わせると、各データ セットのデータ量を 3.2 ギガビットから 100 メガビットに削減できます。この量は生のデータ入力の 64 分の 1 です。

PHI の処理フローは、2 つの動作モードに分けられます (図 1)。これらの 2 つのモードを切り替えるには、Virtex FPGA のインフライト リコンフィギュレーションが最適です。このプロセスの仕組みを次に示します。

収集フェーズ (図 1 の左側の赤いボックス) では、検出器は解像度 2,048 x 2,048 ピクセルの画像を生成します。データ収集用の FPGA は、さまざまなフィルター設定を適用した複数の画像のセットを蓄積し、大容量の NAND フラッシュ メモリに直接保存します。宇宙探査機の残留ジッターを軽減するため、画像安定化システムの専用コントローラーが同時に動作します。

データ収集後、2 個の Virtex-4 FPGA をプリプロセッシング コアと RTE コア でリコンフィギュレーションします (図 1 のオレンジ色のボックス)。プリプロセッシング コアは、以前に保存されたデータをフラッシュ メモリから取得し、ダーク フィールドおよびフラット フィールド補正、フレームの加算、乗算、たたみ込みを実行します。続いて、RTE コアが放射伝達方程式の反転を計算します。

RTE 反転用の FPGA の設計には、スペインのグラナダにあるアンダルシア宇宙物理学研究所が貢献しています。画像安定化システムのコントローラーの設計は、バルセロナ大学が担当しました。

## PHI 処理ユニットのアーキテクチャ

図 2 に、データ処理ユニットのアーキテクチャを示します。宇宙探査機およびシステム コントローラーとの通信には、50MHz のクロック周波数で動作する信頼性の高い専用の GR712 LEON3-FT プロセッサ ASIC を使用します。この CPU は専用の 2 ギガビット SDRAM メモリを持ち、またソフトウェアと FPGA ビットストリーム コンフィギュレーション ファイルを格納する 1 ギガビットの不揮発性 NOR フラッシュ メモリに接続されています。画像収集、画像安定化、

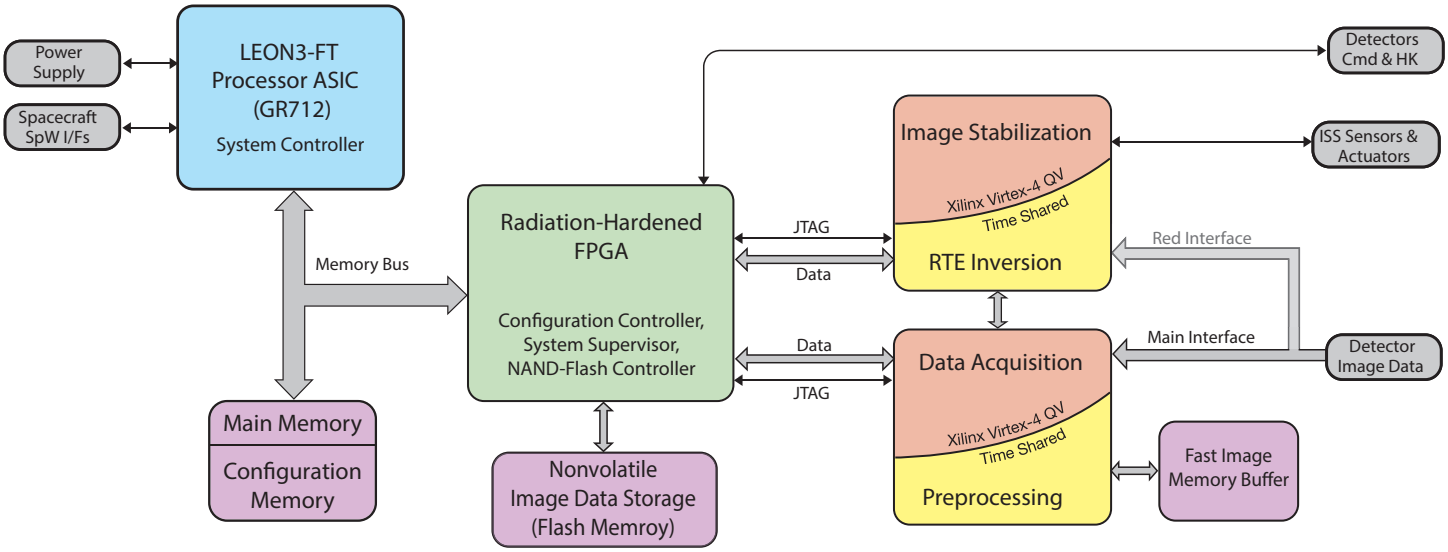



図 2 – PHI データ処理ユニットのアーキテクチャ


図 3 – はんだ付けされたデイジー チェーン デバイスを搭載した検証用ボード

プリプロセッシング、RTE 反転には、2 個の Virtex-4QV FPGA を使用します。インフライト リコンフィギュレーションが可能なため、これらの 2 個のデバイスはタイム シェアリング方式で効率的に利用できます。この方法で、深宇宙探査ミッションで非常に重要な要因となる、プラットフォームの質量、体積、消費電力を削減できます。

放射線防護を強化したワンタイム プログラマブル FPGA が、LEON3 システム コントローラーと 2 個の Virtex-4 デバイスを接続します。さらに、この FPGA はシステム スーパーバイザーとしても機能します。この FPGA が提供する I/O 信号およびインターフェイスは、ハードウェア パーツおよび外部電源の制御と、センサーおよびアクチュエーターとの通信に使用されます。この FPGA は、2 つの JTAG インターフェイスを介して、2 個の Virtex-4 デバイスのコンフィギュレーション ビットストリームの書き込みとリードバックを実行します。

大量の画像データを格納するために、筆者らは多数の NAND フラッシュ デバイスをベースとする総容量 4 テラビットのメモリ ボードを設計しました。別のボード上に配置されるこのメモリ セットを管理するため、システム スーパーバイザー FPGA 内に配置される NAND フラッシュ コントローラーを開発しました。NAND フラッシュ アレイとプロセッシング FPGA 間の比較的低速なデータ レートに対処するため、データ収集とプリプロセッシングに高速の外部バッファー メモリを利用します。専用のネットワークにより、システム コントローラー FPGA とオフチップの 2 個の Virtex-4 FPGA を接続し、NAND フラッシュ メモリ コントローラーとオンチップのプロセッシング コアを接続します。

## 放射線の影響への対処

ザイリンクス Virtex-4QV は耐放射線 FPGA であり、放射線の影響による物理的損傷を受けません。しかし、ビット反転が発生する可能性があるので、デザインはそれを軽減する必要があります。放射線は、コンフィギュレーション層とアプリケーション層の 2 層で SRAM ベースの FPGA に影響を与える可能性があります。

コンフィギュレーション層のビット反転は、主に配線ロジックと組み合わせファンクションを変化させます。この層に発生したエラーを修復する 1 つの方法は、一定の時間間隔でコンフィギュレーション SRAM を上書きすることです ( この手法はスクラビングと呼ばれる )。筆者らは、ビットストリーム上でリードバックを実行し、反転が検出されたときにのみ特定のコンフィギュレーション フレームをリコンフィギュレーションすることにより、スクラビング プロセスを最適化しました。

アプリケーション層への放射線の影響は、制御ロジックのフォルト ( ステート マシンの停止など ) や単にデータ パス内の誤った値をもたらします。筆者らは、アプリケーション

層のビット反転を軽減するために、トリプルモジュラー冗長性 (TMR) 手法とエラー検出および訂正 (EDAC) 手法を使用します。

FPGA デザインのビット反転を上手に軽減するには、コンフィギュレーション層とアプリケーション層の両方を保護するメカニズムを作成する必要があります。スクラビングと TMR のいずれか一方の軽減方式を組み込むだけでは不十分です。

## Virtex-4QV の組み立ての検証

商用品の Virtex-4 FPGA はフリップチップ BGA パッケージで提供されるのに対して、航空宇宙グレード品の Virtex-4QV デバイスはセラミック パッケージで提供されます。互換性のあるフットプリントを維持するために、Virtex-4QV デバイスは、はんだカラムを搭載しています。2012 年に筆者らが Virtex-4QV デバイスの使用を決めたとき、欧州ではこれらの CF1140 パッケージを組み立てる資格認定プロセス メーカーは見つかりませんでした。この理由で、筆者らはミッション固有のパッケージ組み立ての検証を始める必要がありました。

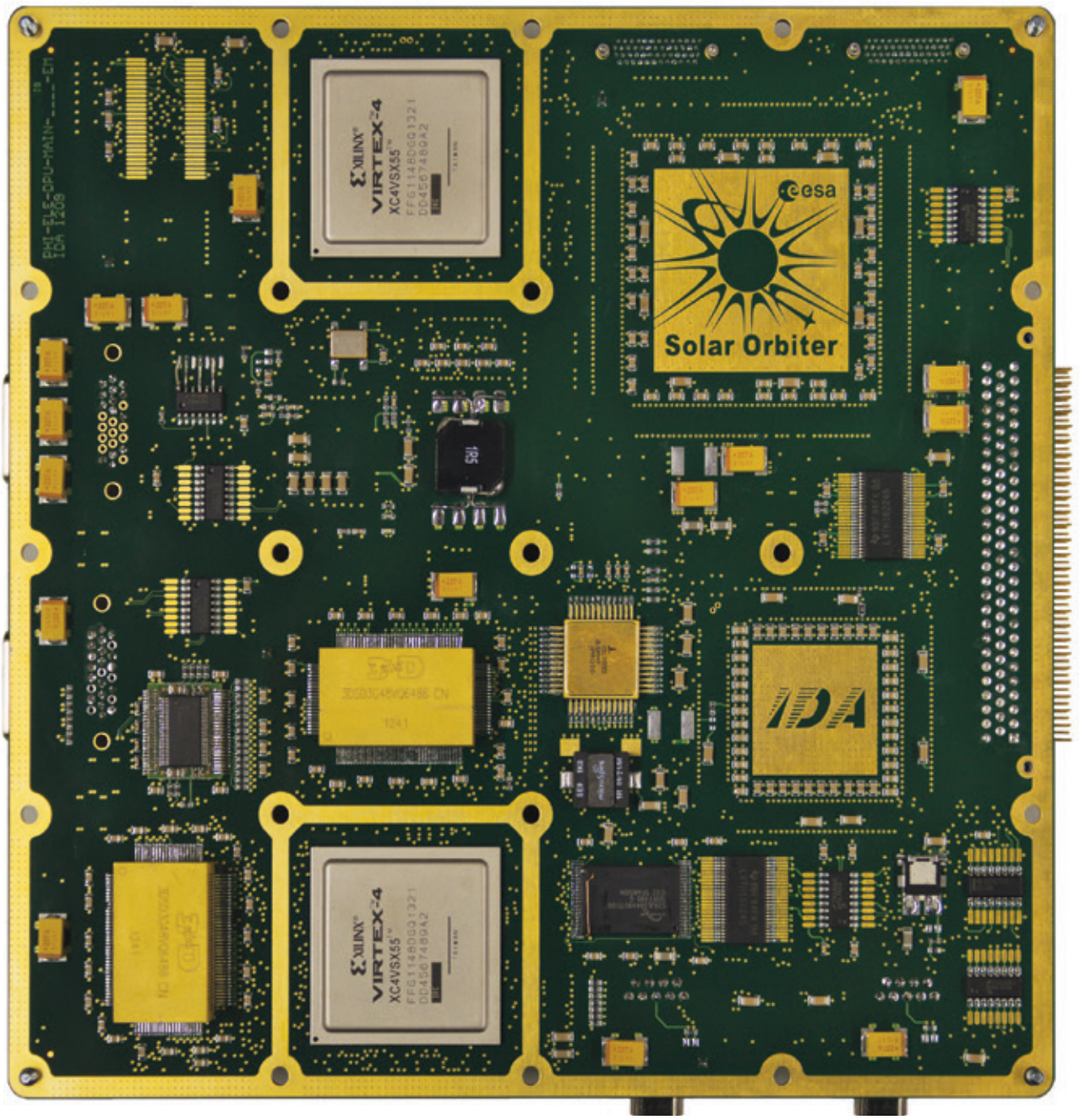



図 4 – PHI データ処理ユニットのエンジニアリング モデルの裏側

この目的のために、筆者らは全体で 6 個の CF1140 デイジー チェーン デバイスを搭載した代表的な検証用ボードを 3 個組み立てました ( 図 3)。専用の衝撃振動試験の実行後、熱サイクル試験を開始し、デイジー チェーン パッケージの耐性を詳細にモニタリングしました。各テスト段階の前後に、デバイスの光学検査と X 線検査を行い、致命的な物理的損傷が発生していないことを確認しました。筆者らは、一枚の PCB の破壊的マイクロセクショニング ( 微細切断撮影 ) を行って、検証プロセスを終えたところです。

## 現状と展望

エラーの軽減と Virtex-4 の組み立て検証を含めてデザインの基本アーキテクチャを定義した後、筆者らは商用品デバイスをベースとするデータ処理ユニットの試作を開始しました。このエンジニアリング モデルは、電子機器ハウジングの 20cm x 20cm の最終的形状に適合し、大きな問題なく既に稼働しています。このユニットの質量は 550g (NAND フラッシュ メモリ ボードを除く )、消費電力は 15W 未満です。2 個の Virtex FPGA を搭載したこのモデルの裏側を図 4 に示します。筆者らは現在、検証済みデバイスを搭載したボードの検証用モデルの仕上げに取り組んでいます。

これまでのことを要約すると、インフライト リコンフィギュレーションに対応する、ゲート数の高い Virtex-4 デバイスにより、サイズ、質量、消費電力を削減した、コンパクトで高性能なデータ処理プラットフォームの開発が可能になりました。データの収集と処理の流れは、2 個のリコンフィギャラブル FPGA を備えたシステムに完璧に適合します。

このシステムは、深宇宙探査ミッションにインフライト リコンフィギュレーション テクノロジを導入する試みの最初の一歩です。航空宇宙業界向け電子機器は、通常は商用テクノロジの進歩に 2、3 年遅れています。現在では、ザイリンクス Zynq®-7000 All Programmable SoC ファミリは、SRAM ベースの FPGA テクノロジと複数のプロセッサ コアを 1 つのシステム オン チップに統合しています。今後数年のうちに、この種の SoC ソリューションが航空宇宙業界のニーズに合うかどうかを検証するのが楽しみです。

# 60G Millimeter-Wave Backhaul Link Is Poised to Boost Cellular Capacity

# 60GHz ミリメートル波バックホール リンクによる大容量通信の実現

**John Kilpatrick**
Consulting Engineer
Analog Devices
*John.Kilpatrick@analog.com*

**Robbie Shergill**
Strategic Applications Manager
Analog Devices
*Robbie.Shergill@analog.com*

**Manish Sinha**
Product Marketing Manager
Xilinx, Inc.
*manish.sinha@xilinx.com*

# ザイリンクスの Zynq SoC をベースとする 60GHz 双方向データ通信方式は、スモール セル バックホール市場に最適な高性能と柔軟性を備えています。

世界の携帯電話網に対する増え続けるデータ需要に応えるため、通信事業者は、2030 年までに通信容量を 5,000 倍に拡大する方法を模索しています [1]。この目標を達成するには、現状の 5 倍のチャネル性能、20 倍の割り当てスペクトラム、50 倍の基地局数が必要です。

新設される基地局の多くは、トラフィックの大部分が発生する屋内に配置されるため、光ファイバーが、ネットワークにトラフィックを送り戻す媒体の第一の選択肢になります。しかし、屋外では光ファイバーが利用できなかったり、接続が高価すぎたりするため、屋外に基地局を設置する場合は、無線バックホールが最も現実的な代替手段です。

ライセンス不要の 5GHz スペクトラムが利用可能で、視線経路も不要です。しかし、5GHz の帯域幅は限られており、大量のトラフィックと広いアンテナ パターンのため、ほぼ確実にこのスペクトラムの他のユーザーからの干渉を受けます。

60GHz 通信リンクは、大容量通信の需要に応えなければならない数千の屋外基地局のバックホール リンクの有力な候補として登場しました。このスペクトラムもライセンス不要ですが、6GHz 以下の周波数帯とは異なり、最大 9GHz の利用可能な帯域幅を含んでいます。しかも、60GHz の高周波数では、非常に狭い焦点を絞ったアンテナ パターンが可能になるため、耐干渉性がある程度向上します。

ザイリンクスと ( 現在は Analog Devices 社の一部門である ) Hittite Microwave 社が開発した完全な 60GHz 双方向データ通信リンクは、スモール セル バックホール市場の要件を満たす優れた性能と柔軟性を示しています ( 図 1)。このプラットフォームのデジタル モデム部はザイリンクスが開発し、ミリメートル波無線通信部は Analog Devices 社が開発しました。

図 1 に示すように、このリンクを作成するには 2 つのノードが必要です。各ノードは、( 変調器付き ) トランスミッターおよびそれに関連するアナログ Tx チェーンと、( 復調器付き ) レシーバーおよびそれに関連するアナログ Rx チェーンを含んでいます。

モデム カードは、アナログ デバイスおよびディスクリート デバイスと統合されます。モデム カードは周波数合成の精度を確保するための発振器 (DPLL モジュール ) を搭載し、すべてのデジタル機能を FPGA または SoC 内で実行します。このシングル キャリア モデム コアは、最大 500MHz までのチャネル帯域幅で QPSK から 256QAM までの変調方式をサポートし、最大

3.5Gbps のデータ レートを実現します。また、このモデムは、周波数分割複信 (FDD) 伝送方式と時分割複信 (TDD) 伝送方式の両方をサポートします。堅牢なモデム設計手法によってローカル発振器の位相ノイズを軽減し、性能とリンク バジェットの向上のために強力な LDPC コーディングを組み込んでいます。

## ミリメートル波モデム

ザイリンクスのミリメートル波モデム ソリューションにより、インフラストラクチャベンダーは、コストが最適化された、柔軟でカスタマイズ可能な無線バックホール ネットワーク用リンクを開発できます。このソリューションは、ザイリンクスの「一世代先へリードする」28nm 製品ファミリであるザイリンクス Zynq®-7000 All Programmable SoC または Kintex®-7 FPGA デバイスをターゲットにしてインプリメントされます。

完全適応型で低消費電力かつ小型フットプリントであるザイリンクスのソリューションは、屋内および完全な屋外におけるポイント ツー ポイントのリンクおよびポイント ツー マルチポイントのマイクロ波リンクの展開に使用できます。ザイリンクスのミリメートル波モデム ソリューションのロードマップは、シリコン製品のロードマップと同様に非常に積極的であり、通信事業者には、現場でのアップグレードが可能なスケーラブルなシステムを展開できるユニークな機能を提供します。

図 2 に、Zynq SoC プラットフォーム上にインプリメントされるデジタル モデムの詳細を示します。このプラットフォームのスケーラブルなプロセッシング システム (PS) には、プログラマブル ロジック (PL) はもとより、デュアル ARM® Cortex™-A9 コアと、統合型メモリ コントローラー、ペリフェラル用マルチスタンダード I/O が含まれます。

このシステム オン チップ (SoC) プラットフォームは高度な柔軟性を備えているため、さまざまなデータ処理機能および制御機能の実行と、ハードウェア アクセラレーションの用途で使用されます。図 2 は、PHY、コントローラー、システム インターフェイス、パケット プロセッサのすべてを搭載した統合型ミリメートル波モデム ソリューションを示しています。しかし、開発者は、必要なアーキテクチャに応じて、各種のモジュールを挿入、更新、または削除できます。たとえば、XPIC コンバイナーをインプリメントするのであれば、このモデムを他のモデムと組み合わせる交差偏波モードで使用できます。このモデム ソリューションは PL 内にインプリメントされ、PL 内では、モデムとパケット プロセッサの間、パケット プロセッサとメモリの間、モデム間、DAC/ADC など、各種のデータ パス インターフェイスに SerDes および I/O が使用されます。

ザイリンクス モデム IP コアのその他の重要な機能には、適応変調符号化 (ACM) を使用したヒットレス (中断のない) でエラーのない自動ステート スイッチングによるリンク動作の維持、閉ループ適応型デジタル プリディストーション (DPD) による RF パワーアンプの効率と線形性の向上、シンクロナス イーサネット (SyncE) によるクロック同期の維持、リードソロモンまたは低密度パリティ検査 (LDPC) 方式のフォワード エラー訂正 (FEC) などが挙げられます。FEC の方法はデザインの要件に基づいて選択されます。無線バックホール アプリケーションでは LPDC 方式の FEC がデフォルトの選択肢であり、リードソロモン方式の FEC はフロントホールなどの低レイテンシのアプリケーションに向いています。

LDPC のインプリメンテーションは高度に最適化されており、エンコーダーとデコーダーによって実行される演算に FPGA の並列処理を利用します。その結果、顕著な SNR ゲインが得られます。LDPC コアの反復回数を変えることにより、並列処理をさまざまなレベルで適用し、デコーダーのサイズと電力を最適化できます。また、チャネル帯域幅とスループットの制約に基づいた、このソリューションのモデル化も可能です。

ザイリンクスのモデム ソリューションは、表示およびデバッグ用の強力なグラフィカル ユーザー インターフェイス (GUI) を搭載し、チャネル帯域幅や変調方式の選択といった高レベルの機能と、ハードウェア レジスタの設定などの低レベルの機能を実行できます。図 1 に示したソリューションでは、3.5Gbps のスループットを実現するために、モデム IP コアは 440MHz のクロック レートで動作します。ADC と DAC の接続インターフェイス用に 5 個のギガビット トランシーバー (GT) を使用し、10GbE ペイロードまたは CPRI インターフェイス用にさらに数個の GT を使用します。

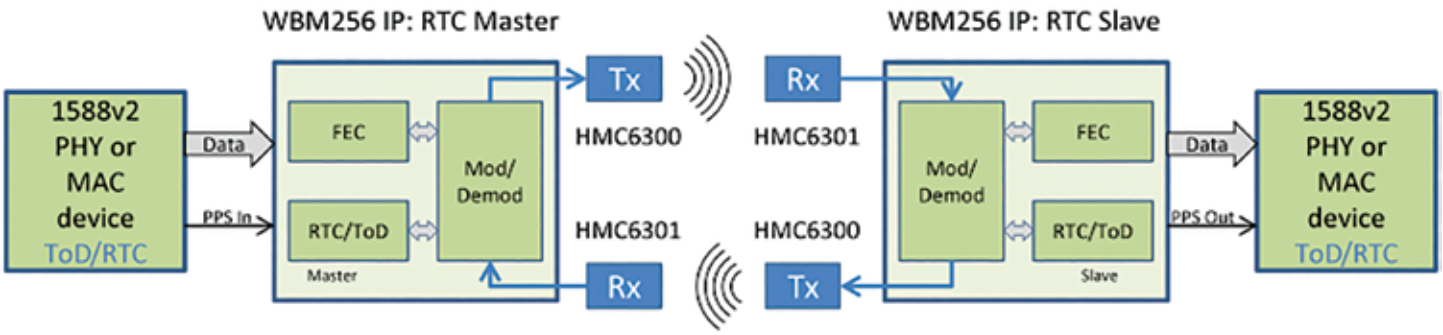



図 1 – 完全な双方向通信リンクの高レベルのブロック図

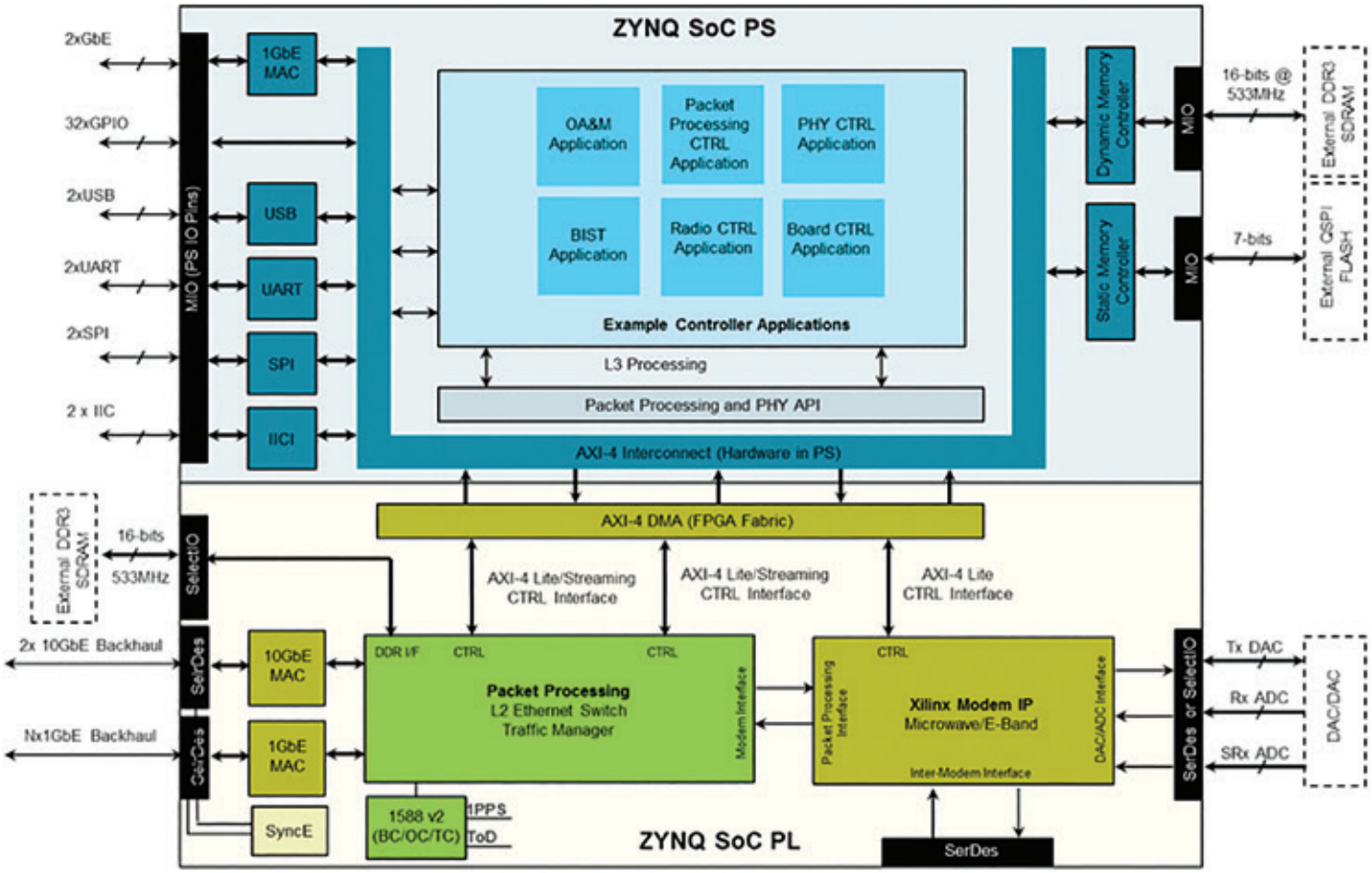



図 2 – 無線モデム アプリケーション用の All Programmable SoC

## ミリメートル波トランシーバー チップセット

2014 年後半、Analog Devices 社は、スモール セル バックホール アプリケーション向けの大幅な機能強化と最適化を実現した、第 2 世代シリコン ゲルマニウム (SiGe) 60GHz チップセットを発表しました。HMC6300 トランスミッター チップは、アナログ ベースバンドからミリメートル波への完全なアップコンバーターです。改良された低位相ノイズの周波数シンセサイザは、57GHz ～ 66GHz を 250MHz ステップでカバーし、少なくとも 64QAM までの変調方式をサポートします。出力電力はおよそ 16dBm の線形電力まで増加しましたが、各種規制値を超えないように、内蔵電力検出器が出力電力を監視します。

このトランスミッター チップは、IF および RF ゲインをアナログ制御またはデジタル制御できます。アナログ ゲイン制御は、高次変調方式の使用時にしばしば必要とされます。これは離散ゲインの変化が振幅変調として誤認識され、ビット エラーを発生させることがあるためです。デジタル ゲイン制御は、内蔵 SPI インターフェイスを使ってサポートされます。

狭いチャネルでさらなる高次変調を必要とするアプリケーションでは、さらに低位相ノイズの外部 PLL/VCO をトランスミッターに挿入し、内部シンセサイザをバイパスできます。図 3 に、HMC6300 のブロック図を示します。

トランスミッターは最大 1.8GHz の帯域幅をサポートします。MSK 変調器のオプションを使用すれば、消費電力の大きい高価な DAC を使用せずに、低コストで最大 1.8Gbps のデータ伝送が可能です。

このトランスミッターは、スモール セル バックホールの厳しい要件を満たすよう、同じように最適化された HMC6301 レシーバー チップと組み合わせて使用されます。パラボラ アンテナの高ゲインによってレシーバー入力の信号レベルが高くなる短距離リンクを扱えるように、このレシーバーの入力 P1dB が -20dBm、IIP3 が -9dBm に大きく向上しています。

レシーバーのその他の特長として、最大ゲイン設定で 6dB の低雑音指数、調整可能なローパスおよびハイパス ベースバンド フィルター、57GHz ～ 66GHz 帯で 64QAM 変調をサポートする新しいシンセサイザ ( トランスミッター チップに搭載されるものと同じ )、IF および RF ゲインのアナログ制御またはデジタル制御が挙げられます。

図 4 に、HMC6301 レシーバー チップのブロック図を示します。このレシーバーには、オン / オフ キーイング (OOK) などの振幅変調を復調する AM 検出器も含まれていることに注意してください。また、FM 識別回路は、単純な FM 変調または MSK 変調

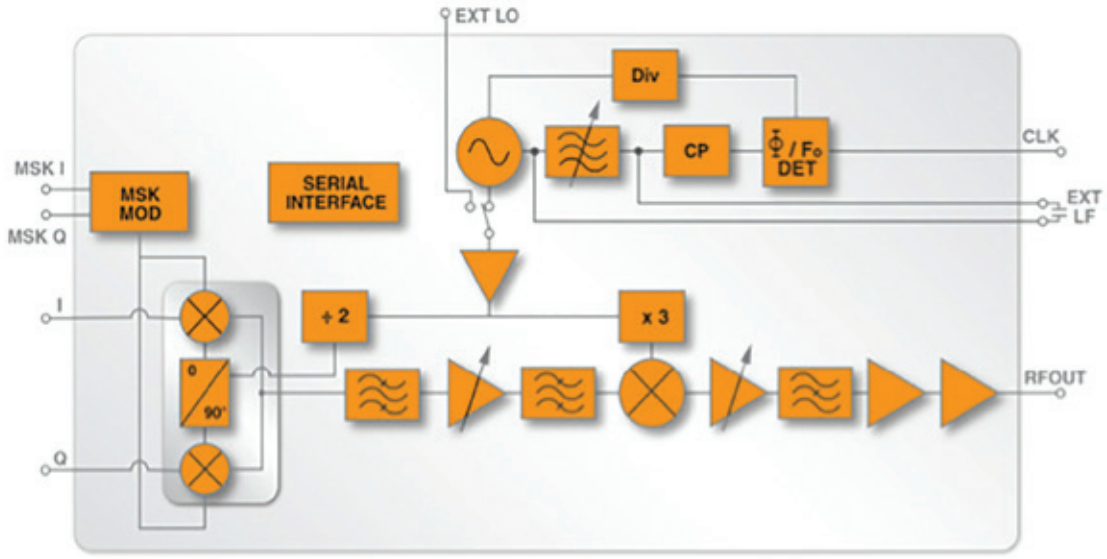



図 3 – HMC6300、60GHz トランスミッター IC のブロック図

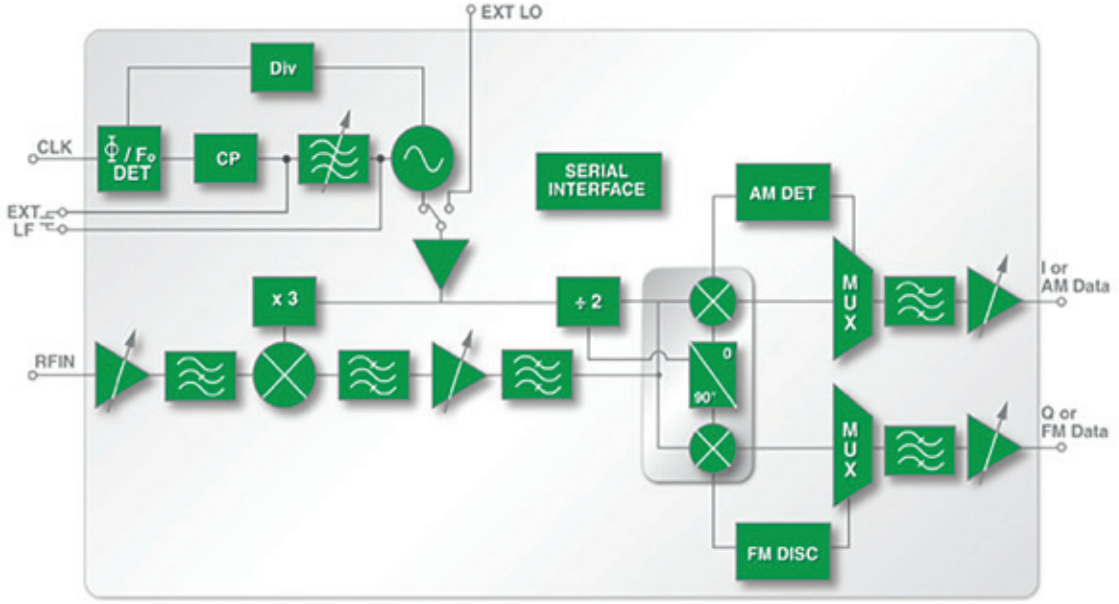



図 4 – HMC6301、60GHz レシーバー IC のブロック図

を復調します。それに加えて、QPSK 変調およびより複雑な QAM 変調で直交ベースバンド出力の復元に使用される、IQ 復調器があります。

HMC6300 トランスミッターと HMC6301 レシーバーは、いずれも 4 x 6mm ウェハー レベル BGA パッケージで供給されます。製品番号は HMC6300BG46 と HMC6301BG46 で、2015 年前半にサンプル出荷が予定されています。これらの表面実装デバイスを使って、無線ボードを低コストで製造できるようになります。

ミリメートル波モデム / 無線通信システムのサンプル デザインのブロック図を図 5 に示します。このデザインには、FPGA、モデム ソフトウェア、ミリメートル波チップセット以外のコンポーネントが多数含まれています。これには、AD9234 デュアル チャネル 12 ビット 1 ギガサンプル / 秒 ADC、AD9144 クワッド チャネル 16 ビット最大 2.8GSPS ( ギガサンプル / 秒 ) Tx-DAC、HMC7044 超低ジッター クロック シンセサイザと (ADC IC と DAC IC の両方で使用される ) JESD204B シリアル データ インターフェイスのサポートが含まれます。

## デモンストレーション プラットフォーム

ザイリンクスと Analog Devices 社は、ザイリンクス KC705 開発ボード (ADC、DAC、クロック チップ、2 個の無線モジュール評価ボードからなる業界標準規格の FMC ボード ) 上に FPGA ベースのモデム機能を搭載

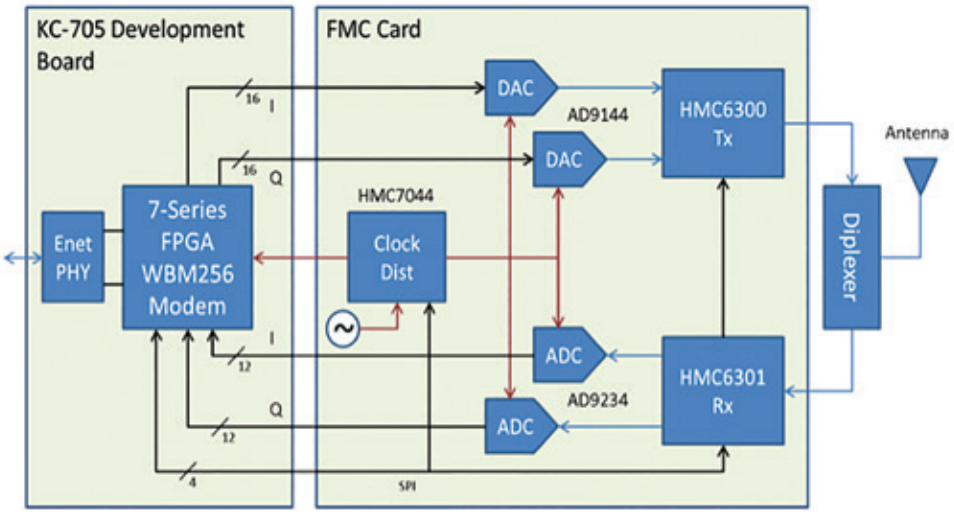



図 5 – ザイリンクスと Analog Devices 社の IC を使用した
サンプル リファレンス デザイン

したデモンストレーション プラットフォームを共同で開発しました ( 図 6)。このデモ プラットフォームには、モデムの制御およびビジュアル表示用のノートブック PC と、標準的なミリメートル波リンクのパス損失を複製する可変 RF 減衰器が含まれます。ザイリンクス KC705 開発ボードは、WBM256 モデム ファームウェア IP を実行する Kintex-7 XC7K325T-2FFG900C FPGA を搭載しています。KC705 開発ボード上に業界標準規格の FMC メザニン コネクタを使用して、ベースバンド ボードとミリメートル波無線ボードを接続します。

ミリメートル波モジュールはベースバンド ボードに装着されます。このモジュールは、60GHz インターフェイス用の MMPX コネクタと、オプションで使用される外部ローカル発振器用の SMA コネクタを搭載しています。

このプラットフォームは、周波数分割複信接続の各方向に対して 250MHz チャネルで、最大 1.1Gbps のポイント ツー ポイント バックホール接続をデモンストレーションするのに必要なすべてのハードウェアとソフトウェアを含んでいます。

## カスタマイズ可能なモジュラー構成

FFPGA ベースのプラットフォームは高度なモジュール化とカスタマイズが可能であり、OEM の総保有コストが削減されるため、各種の無線バックホール ソリューションへの FPGA の採用がしだいに増えています。7 シリーズ FPGA/SoC ファミリの消費電力の大幅な向上と高性能のワイドバンド IP コアを活用したザイリンクスのミリメートル波モデム ソリューションは、スモール セル バックホール アプリケーションの有力な候補として期待されています。ザイリンクスの FPGA および SoC は電力効率の高い高速デザインに最適で、高速 GT はワイドバンド通信およびスイッチング機能に使用すると効果的です。ザイリンクスのソリューションは、数百メガビット / 秒で動作するローエンドのスモール セル バックホール製品から、3.5Gbps で動作するハイエンド製品に至るまで、同じハードウェア プラットフォーム上で多彩な製品をサポートするスケーラビリティを備えています。

無線通信部については、トランシーバーがシリコン ベースの IC に組み込まれ、表面実装デバイスにパッケージされているため、低コストにて製造が可能です。Analog Devices 社のミリメートル波チップセットは、スモール セル基地局を展開するための無線バックホールの要件を満たし、消費電力、サイズ、柔軟性、機能の面で市場をリードする性能を備えています。Analog Devices 社は、この完全なソリューションの重要なコンポーネントである業界最高クラスのデータ コンバーターと クロック管理 IC も提供しています。ザイリンクスと Analog Devices 社は協力して、このエキサイティングなテクノロジの普及促進に取り組んでいます。

## 参考資料

1. “Evolutionary and Disruptive Visions Towards Ultra High Capacity Networks,” IWPC, April 2014

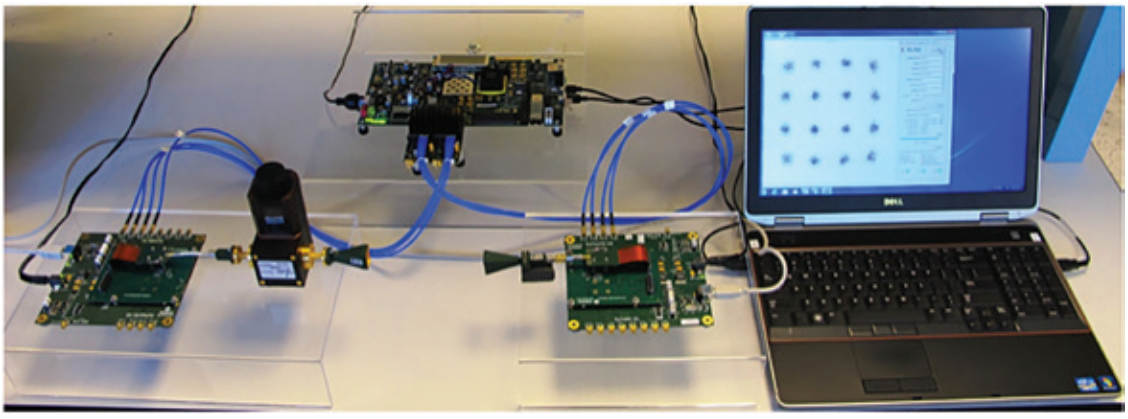


図 6 – 動作中のデモンストレーション プラットフォーム

# MACsec IP Improves Data Center Security

# MACsec IP コアによって データ センターのセキュリティを向上

**Paul Dillien**
Consultant
High Tech Marketing
*paul@high-tech-marketing.co.uk*

**Tom Kean, PhD**
Managing Director
Algotronix Ltd.
*tom@algotronix.com*

# データ センター機器の設計者は、FPGA ベースのコアを利用して高性能でセキュアなイーサネット リンクを実現できます。

クラウド ストレージと IT サービスのアウトソーシングは、コストの削減とサポート作業の軽減を可能にするため、IT マネージャーにとって多くの利点があります。しかし、企業ファイアウォールの外側に機密データを置く場合、セキュリティの不安が大きな問題になります。財務データ、顧客データ、製造関連データなどの情報は、多くの企業にとって最も価値ある資産の 1 つですから、躊躇するのも無理はありません。

しかし現在、データ センター機器メーカーは、ザイリンクス FPGA ベースのソリューションを使って機器の性能を向上させ、セキュリティの新たな上位基準を確立しています。MACsec と呼ばれる新しいイーサネット標準に適合する Algotronix 社の包括的なセキュリティ サブシステムは、高性能かつ低レイテンシで電力効率の高いザイリンクス FPGA の IP ( 知的設計資産 ) コアを使用します。

FPGA ベースのソリューションは、ソフトウェア ベースのソリューションよりもはるかに高速です。さらに、専用のハードウェアでシステム プロセッサの作業負荷を取り除くことで、処理能力をディープ パケット検査などの他のタスクに解放できます。あるいは、設計者がより低価格のプロセッサを採用することも可能です。

## 暗号化と認証

ネットワーク上およびデータ センター内を移動するデータを暗号化することは、疑問の余地のない情報保護対策と言えるでしょう。データが暗号化されていれば、万一不正な関係者がリンクを盗聴してデータを傍受しても、情報は読み取れません。理想的には、データを認証し、保全性を確保する必要もあります。メッセージ認証機能は、伝送エラーが原因で、あるいは不正な利益を追求する攻撃者が故意に改ざんしたために、オリジナルの暗号化データが改変された場合、どこで改変されたかを検出するように設計されています。

イーサネット通信は、高速伝送に最適な効率性と拡張性を備えているため、ネットワーク通信の大半を占めるほどに成長しました。イーサネット標準は広く利用されることでコストが低下し、さらに魅力を増しました。この好循環によって、イーサネットは最適なレイヤー 2 テクノロジとしての地位を確保しています。しかし、つい数年前までイーサネット仕様には暗号化が含まれていなかったため、通信プロトコル スタックの上位層で動作する IPsec などのテクノロジが暗号化処理を担当していました。

現在、IEEE 802.1AE 仕様の下で新たに拡張されたイーサネットには、多数のセキュリティ対策が追加されています。数年前に定義されたこの技術は、メッセージの暗号化と認証を行うとともに、ネットワーク上のさまざまな攻撃を検出し、システムを攻撃から保護する統合型セキュリティ システムを備えています。この仕様はメディア アクセス コントロール セキュリティ標準 ( 一般には MACsec) と呼ばれています。

Algotronix 社は、数年前から、ハードウェア アクセラレーションによる暗号化機能をさまざまなデータ レートで提供する IP コアの開発に着手しました (Algotronix 社は、MACsec 製品に非常によく似たインターフェイスを備えた IPsec 用 IP コアも供給します。MACsec と IPsec の両方の標準をサポートする必要があるシステムには、この IP コアが最適です )。

MACsec システムの概要を簡潔に説明すれば、この仕様の包括的な性質と、インプリメント作業の複雑性がよく理解できると思います。

## 信頼されたエンティティ

MACsec の構想では、ネットワーク上のノードは一連の信頼されたエンティティを形成します。各ノードは暗号化されたメッセージとプレーンテキスト メッセージの両方を受信でき、それぞれのメッセージをどのように処理するかはシステム ポリシーによって決まります。コアには、認証や検証の対象にならないプレーンテキスト メッセージをバイパスするオプションが含まれています。エンドツー エンドのテクノロジとしてレイヤー 3/4 で動作する IPsec のようなプロトコルとは異なり、MACsec は、パケットがイーサネット LAN を出入りするたびに各パケットの復号化と検証を行います。MACsec は、スター型接続 LAN またはバス型 LAN などのイーサネット トポロジとポイント ツー ポイント システムに適しています。

MACsec 仕様は、セキュリティ エンティティ (SecY) と呼ばれる手法を使用します。この手法では、各ノードまたはエンティティは各自のイーサネット ソース アドレスに関連付けられる固有のキーを持ちます。筆者らは、複数の仮想 SecY をサポートする MACsec コアの 1G 版を設計しました。その結果、マルチアクセス LAN などのアプリケーションで、1 つのイーサネット MAC がそれに関連付けられる複数の MACsec SecY を持つことが可能になります。MACsec は、通常は IEEE 801.1X-2010 規格すなわち Internet Key Exchange (IKE) プロトコルと連携して動作します。IKE は全ネットワーク内のセキュア キー分配機能を提供します。

データ センター内のパケット転送にレイヤー 2 接続が選ばれる理由は、レイテンシとパケット内のオーバーヘッド データを最小限に抑えながらセンター内で高速通信を実現できる点にあります。それに対して、セキュアなレイヤー 3 テクノロジを使用する IPsec などの通信では、処理のためにメッセージをスタックの上位層に渡す必要があるため、レイテンシが大きくなります。

レイヤー 2 ソリューションを使用すると、レイヤー 3 のセキュリティ ポリシーを作成する複雑な手順も不要になります。データ センターは、ファイアウォールの内側を保護するために MACsec を採用することも、データ センター間の直接リンク上で MACsec を使用することもできます。システム管理者は、セキュアな方法で通信する機器を認証できます。この機器はエラーまたはサービス拒否攻撃 (DOS) などの誤使用を検出できます。

## 優れたプログラマビリティ

MACsec 市場は多様な要件で細分化されているため、MACsec にはカスタマイズ可能な FPGA ソリューションが理想的です。元来 MACsec はメトロポリタン エリア ネットワークに利用されるテクノロジと考えられていましたが、その用途は現在データ センターにも広がり、FPGA ベースのソリューションの全体的な需要は拡大しています。

Algotronix 社は既に AES-GCM と呼ばれる各種の暗号化エンジンを開発していたため、MACsec コアの開発に向かうのは自然な展開でした。これらのコアは、1G、10G、および 40G で動作します。筆者らは、パイプライン化、クロック周波数の向上、ザイリンクス Artix® デバイスから Kintex® デバイスへ、さらに Virtex® FPGA への段階的な移行によってこのようなスピードを実現しました。さらに、これらの手法を採用して、Virtex UltraScale™ デバイス上で 100G のスループットの実現に取り組んでいます。

FPGA 内の IP コアを使って達成できる性能は選択可能であり、ギガビット イーサネットから 10GbE までの任意の容量 ( ワースト ケース条件でのコアの実際のスループット ) をサポートします (40G 版と 100G 版は計画中です )。この速度はソフトウェア ベースのシステムよりもはるかに高速です。MACsec コアは、通常はハードウェア MAC に直接接続されます ( 図 1 を参照 )。これは、FPGA チップ上のエンベデッド プロ

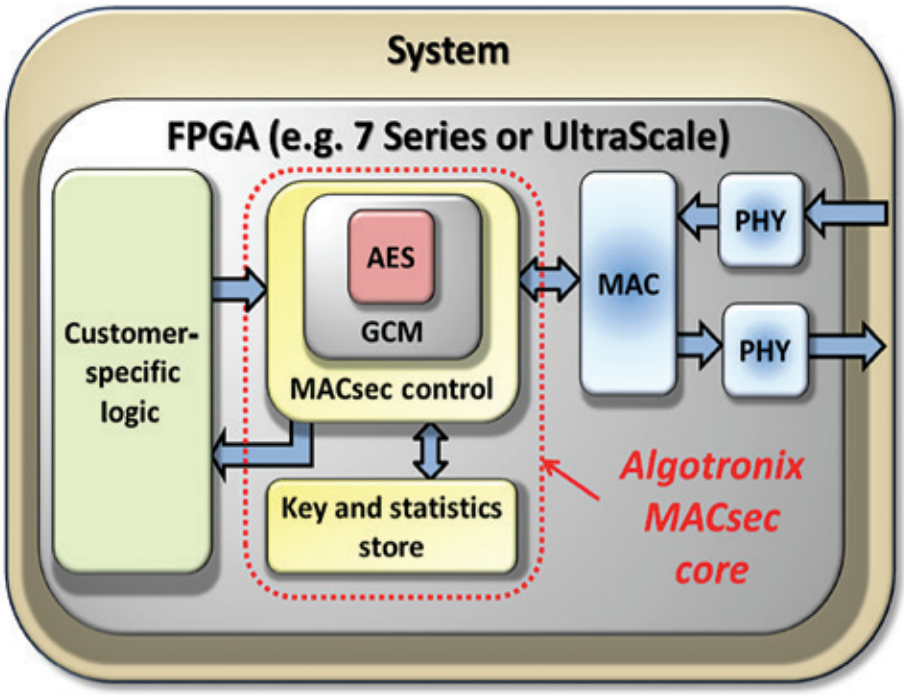



図 1 – 最大限のセキュリティを確保するため、MACsec IP コアは完全に FPGA 内部に配置されます。

セッサのソフトウェアでは、スループットの処理に必要な高速データ転送に対応できずに苦戦する可能性があるためです。ハードウェア内にセキュリティ機能をインプリメントした上に、ソフトウェアから非暗号化キーを利用できないようにすれば、トロイの木馬やウィルスなどの一般的なソフトウェアベースの攻撃に対するシステムの脆弱性を軽減できます。これにより、IT 担当者がシステムのソフトウェア側全体を考慮に入れなければならない場合に比べて、脆弱性の分析が容易になります。

考慮すべきもう 1 つの重要な点は、ソフトウェアにインプリメントされる暗号化機能などのアルゴリズムを FPGA にインプリメントした高速化システムでは、消費電力が劇的に削減されることです。FPGA を使用すると、ソフトウェア ソリューションに比べて電力効率が飛躍的に向上します。

Algotronix 社のすべての暗号化コアは、S-Box と呼ばれる重要なブロックを、FPGA ファブリックのブロック RAM 内またはルックアップ テーブル (LUT) 内の何れかにインプリメントする便利な機能を搭載しています。このオプションを利用して、2 種類のリソースのトレードオフにより、利用可能なリソースにデザインを凝縮してインプリメントできます。たとえば、MACsec コアの外側のデザインがブロック RAM を大量に使用していなかった場合は、ブロック RAM を使って S-Box をインプリメントし、デザインが BRAM を大量に使用していた場合は、LUT を使って S-Box をインプリメントできます。

## MACsec の入力と出力

MACsec システムの構想では、各データ ソースが異なる暗号化キーを使用します。メッセージを受信すると、受信側はオンチップ CAM に保持されたリスト内でそれを検索し、そのパケットの復号化に使用する適切なキーを決定します。反復または再生されるパケットを検出し、拒否できるように、各パケットには番号が付けられます。これは「中間者攻撃」を防ぐための手法です。

また MACsec は、拒否されたパケットの数と拒否された理由に関する統計情報を収集します。統計情報によって攻撃の検出を支援することで、基本的な暗号の機密性、認証、再生防止を超えたレベルでセキュリティを強化し、システム管理者が進行中の攻撃に対して予防的に対応できるようにします。

筆者らは、定評ある AES-GCM コアの周囲を MACsec ロジックで「包む」手法を採用しました。つまり、効率的な高速暗号化コアを設計することは、設計上の課題の一部にすぎません。MACsec 仕様は広範囲にわたり、多くの変数を含んでいます。たとえば、MACsec 標準は当初は 128 ビット暗号化キーのみを指定していました。128 ビット キーでは、データが 10 回変換されて ( 各回はラウンドと呼ばれる ) コア内の暗号化プロセスが完了します。MACsec 標準はその後改訂され、256 ビット キーのオプションが追加されました。256 ビット キーは 14 ラウンドの処理で暗号化を完了します。このオプションは、パイプラインの段数を増やし、キーの格納に使用されるメモリの幅を広げることによって実現されます。

MACsec はイーサネット トラフィックのタイプに依存せず、上位層のプロトコルに対して透過的です。これらのコアの導入により、システムに簡単に MACsec を追加してネットワーク内の保護レイヤーを強化できます。MACsec を導入したサイトも他のサイトと通信できますが、MACsec のセキュリティ強化機能は利用できません。

イーサネット パケットは、メディア アクセス コントローラー (MAC) から MACsec コアに供給されます。たとえば、1G MACsec コアとオンチップ トランシーバーおよびトライモード イーサネット MAC (TEMAC) を組み合わせて、小型で効率的なソリューションを構築できます。各パケットには、デスティネーションと、パケットの伝送を開始したソースのアドレスが含まれます。この標準は MACsec システム内でも維持されますが、重要な点は、マルチホップ通信では最後にパケットを転送した機器のアドレスが「ソース」になることです。したがって、エンド ツー エンドの方式と考えられる IPsec とは異なり、MACsec はホップごとに作用します。各ホップについて、MACsec は、フォワード転送の条件として、入力ポート上のすべての暗号化データが、その機器に割り当てられた固有のキーで復号化され、再暗号化されることを要求します。復号化されたプレーンテキストにより、図 2 に示すように、各ステージでのパケット検査のオプションが可能になります。復号化されたプレーンテキストは、トラフィック マネージャーによってデータのフローの調整に使用されることもあります。

MACsec 標準では、図 3 に示すヘッダーには、MAC Security TAG (SecTAG) と呼ばれる追加のフィールドが含まれます。SecTAG は、EtherType と、パケットの暗号化の有無を示すフラグを定義します。ICV フィールドのメッセージの最後にデータを付加することにより、認証が実現されます。ICV は暗号化キーと連携して ( ヘッダーと MACsec タグを含む ) フレームを認証し、フレームのソース アドレスまたはデスティネーション アドレスが操作可能でないことを保証します。筆者らは、このロジックが高速で予測可能なタイミングで動作し、レイテンシが最小限に抑えられるように、FPGA ファブリックにこのロジックをインプリメントしました。

MACsec コアには、各ソース アドレスにリンクされるルックアップ テーブルが含まれます。このテーブルには、メッセージを正しく復号化するのに必要なキーが含まれています。この機能は、デバイスの LUT およびブロック RAM に効率的にインプリメントされるように設計されています。筆者らは、FPGA ソリューションの柔軟性を利用して、128 ビット キーと 256 ビット キーの選択や、コアでサポートされる仮想 SecY 数の変更機能などのインプリメンテーション オプションを備えたコアを設計しました。

新しい MACsec 標準のもう 1 つの便利な機能は、MACsec によるパケット レベルでの統計情報の照合です。システム管理者は、たとえば、到着の遅延、無効な復号化キーによる保全性チェックの失敗、誤ったキーの使用などが原因で拒否されたパケットの数を確認し、これらの統計情報と、正常に伝送されたパケットの数を比較できます。

MACsec 標準は、ポイント ツー ポイント アプリケーション用に簡略化されたオプションを備えています。このオプションでは、CAM はパケット内の明示的なセキュア チャネル識別子 (Secure Channel Identifier) からキーを判別する必要がなくなり、ポイント ツー マルチポイント動作のオプションは不要になります。筆者らのコアは、1 つのイーサネットに関連付けられる複数の仮想 SecY をサポートします。これにより、その MAC から異なるデスティネーションに送信されるデータの暗号化に、それぞれ

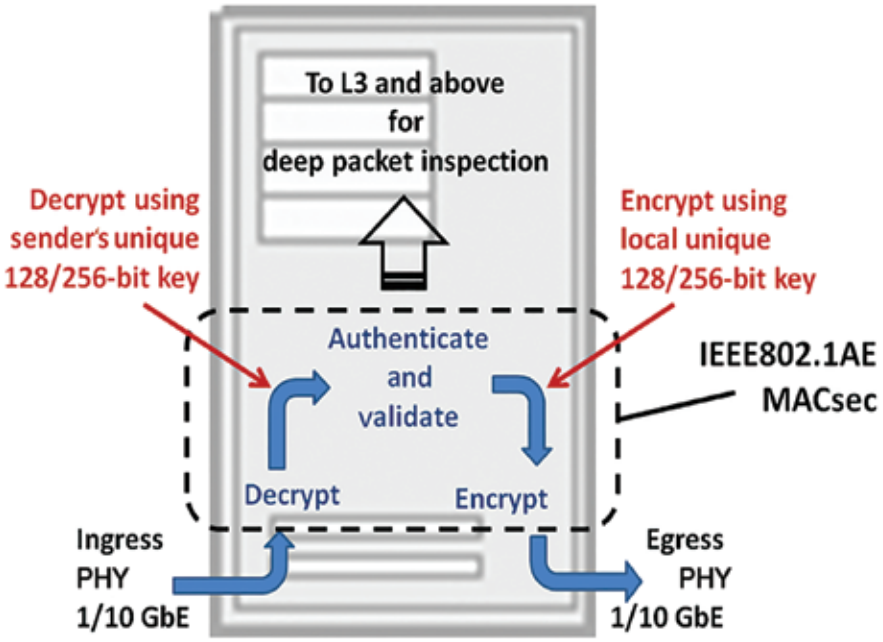



図 2 – メッセージは入力ポート上で復号化され、出力ポート上で暗号化されます。

異なるキーを使用できます。あたかも各デスティネーションが異なるイーサネット LAN 上にあるかのように見えるため、MACsec 標準は、このようなコンフィギュレーションをマルチアクセス ローカル エリア ネットワークとして定義しています。この機能により、システムは、異なるキーを使って出力を暗号化することによって受信側機器を分割できます。

データ センターでは、複数の SecY を使って仮想ディビジョンを作成し、顧客 A からのデータと顧客 B のデータを固有の暗号化キーによって分割できます。必要に応じて、選択したラックを隔離するようにデータ センター内部の通信を構成し、仮想隔離エリアを構築できます。この機能により、データセンターおよびクラウド アプリケーション内のデータの保全性と分離に関する懸念に対処できます。MACsec システムは誤った接続や悪意ある行動による不正なパケットを検出し ( 図 4 を参照 )、システム管理者はこれらのパケットを隔離または削除するためのポリシーを設定できます。

データの暗号化と復号化は、すべてポート レベルで実行されます。ポート レベルの暗号化を有効にした場合、追加の MACsec ヘッダーとレイテンシの多少の増加以外に、オーバーヘッドや性能への影響は発生しません。

機器ベンダーは、今すぐにこれらのコアを使用して、IEEE 802.1AE 準拠のイーサネット レベル 2 暗号化方式を組み込むことで自社のシステムを差別化できます。他のユーザーを互いに信頼できないクラウド ベースのユーザーは、MACsec が提供する機密性とデータ ソース認証のメリットを享受し、データが確実に保護されているという安心感を高めます。機器メーカーは、1 ギガビットおよび 10 ギガビット イーサネットのスループット要件に対応する IP コアの中から選択できます。Kintex または Virtex FPGA デバイス上で 10Gbps の IP コアを簡単に実現できるように、アーキテクチャ デザインは設計されています。このデザインは、各パケット上のキーの変更によってジャンボ フレームと最小サイズのパケットの両方をサポートします。このシナリオはシステムのワースト ケースの条件を示しています。これらのコアは MACsec のフル仕様に準拠し、各 MACsec コアは一般的な FPGA ファミリを幅広くサポートします。

## ソース コードが付属

Algotronix 社は、業界の慣例を破り、すべてのライセンス付きコアの HDL ソース コードを供給しています。その主な目的は、コードにウィルスやトロイの木馬が含まれて

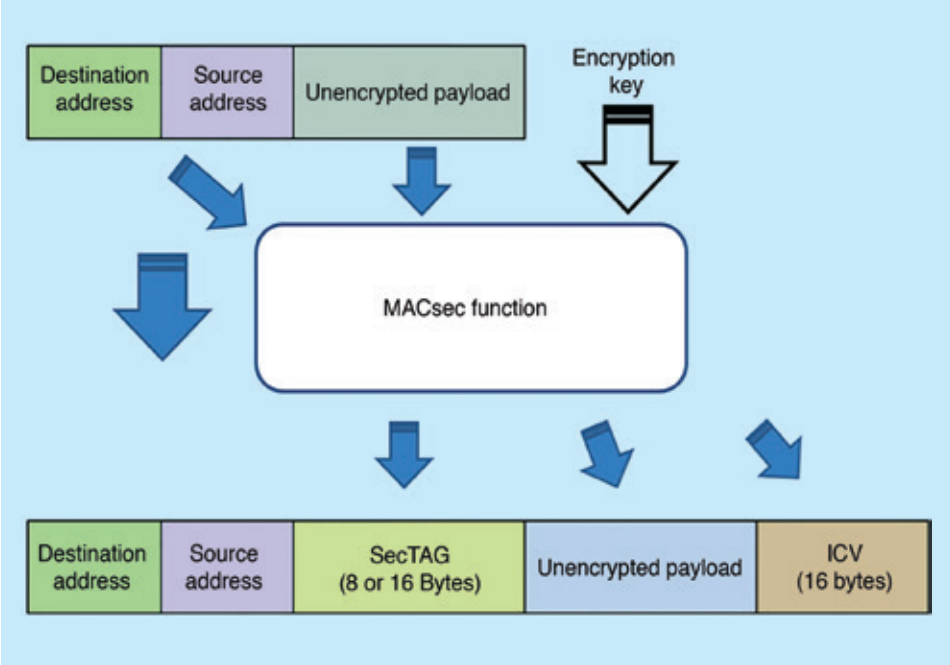



図 3 – MACsec フレーム構造には、MAC Security TAG (SecTAG) と呼ばれるフィールドが含まれています。このフィールドは、EtherType と、パケットの暗号化の有無を示すフラグを定義します。

いないこと、コードが強制的に不正な状態や動作に陥らないことを、お客様自身で検査できるようにすることです。ソース コードがあれば、セキュリティ監査のコストと複雑性が軽減されます。また、暗号化、復号化、または暗号化 / 復号化といったコンフィギュレーション変数とキー長を簡単に実験でき、独自のシミュレーションを行ってコア内の信号の状態を確認できるため、デザイン プロセスを加速できます。コアのコンフィギュレーション次第で、広いデータ パスをインプリメントして高いスループットを得ることも、狭いデータ幅を選択して FPGA のフットプリントを最小限に抑えることも可能です。ソース コードを入手することのさらなる利点として、コアの動作を簡単に理解できること、文書化と保管が簡単に素早く行えることが挙げられます。

コアには大規模な検証テストベンチが含まれ、ModelSim などのツール内でコアの正常な動作を確認できます。このテストベンチは MACsec の動作モデルと MACsec IP コアのセルフチェック版で構成され、合成可能なハードウェアの出力が動作モデルに対してチェックされます。このセルフチェック デザインはユーザーのシミュレーション環境にインスタンシエートできるため、実際のユーザー デザインの文脈で簡単にコアをテストし、コアが正常に駆動しない場合は有益な診断メッセージを生成します。

このコアは多数のオプションを利用できるので、正確なリソース数は、データ レート、キー長、選択した SecY の数などのパラメーターの選択によって異なります。ザイリンクスのウェブサイトの IP のセクションに掲載されている 10G MACsec コアは、6,638 スライス、20,916 LUT、53 BRAM ブロックを使用します。ライセンス オプションについては Algotronix 社までお問い合わせください。

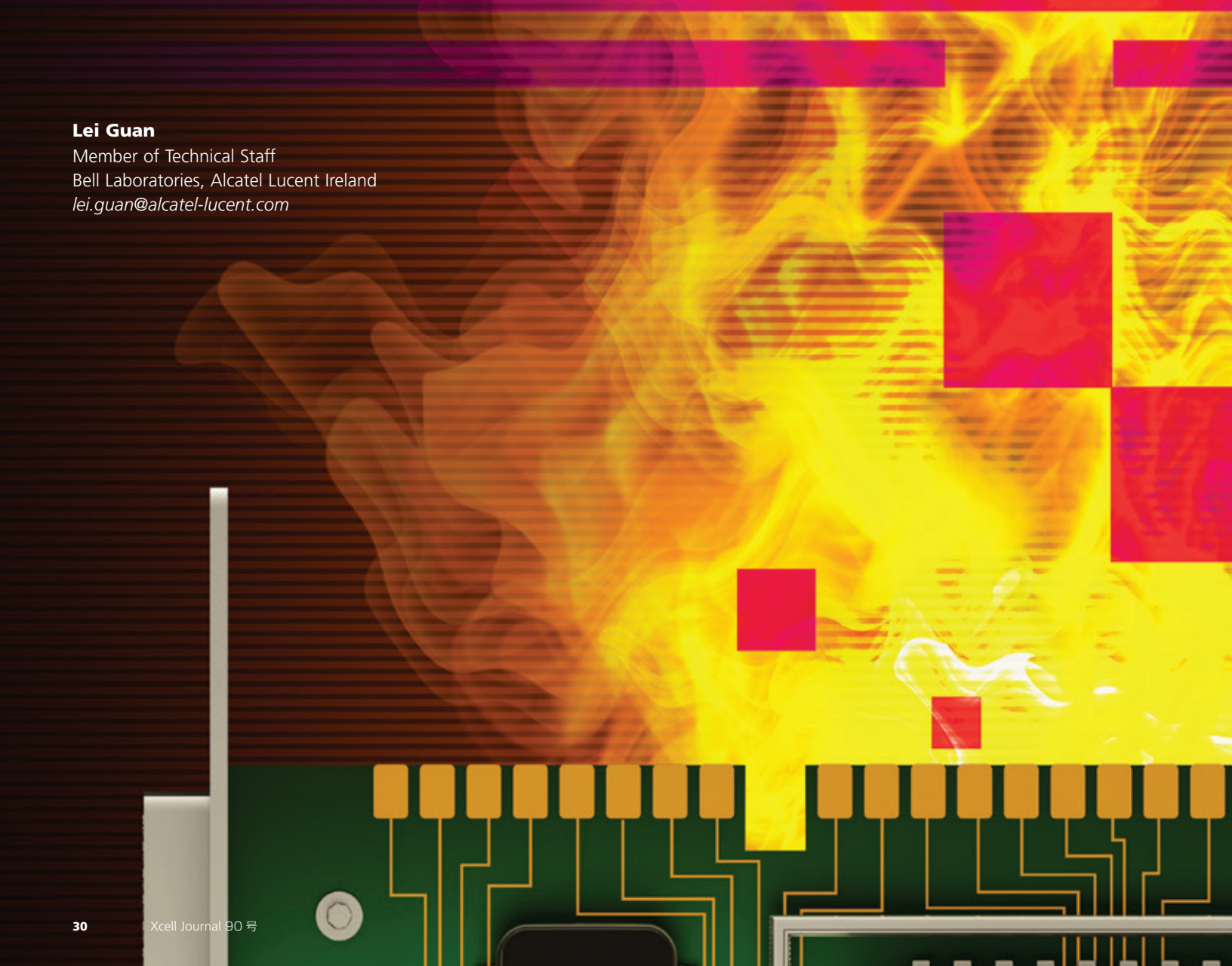
低消費電力のザイリンクス FPGA と Algotronix 社の MACsec コアを組み合わせた、高性能で低レイテンシのソリューションを使って、機器メーカーは自社の製品を差別化できます。このセキュリティ機能は、データ センターの機密性を顧客に保証すると同時に、セキュリティ管理者が悪意ある行動を検出し、システムを攻撃から保護することを可能にします。

図 4 – MACsec は、偶然または故意によって生じた誤った接続を介して到着したパケットを拒否します。

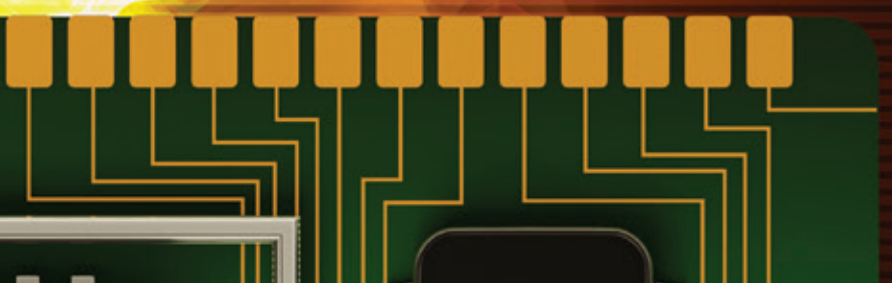
# Simplify Your 'Hot' Testing with Xilinx's Zynq SoC

# ザイリンクスの Zynq SoC を使用し、「ホット テスト」を簡略化

**Lei Guan**
Member of Technical Staff
Bell Laboratories, Alcatel Lucent Ireland
*lei.guan@alcatel-lucent.com*



# Zynq SoC とザイリンクス IP コアを使用して高速光トランシーバー モジュールの熱テストを効率化する方法を説明します。



データ センター内の光トランシーバー モジュールの伝送速度が向上するにつれて、データ センター内の各シャーシの温度は急激に上昇します。縦横かつ高密度に配置されたラック内に高温の高速モジュールが積み重ねられると、温度の上昇は複合的になりそれに拍車がかかります。このような複合的で急激な温度上昇によって、チップの温度が上限を超えることでチップの致命的な障害を引き起こし、データ センター システム全体に悪影響を与えるおそれがあります。したがって、光トランシーバー モジュールは熱特性を考慮に入れて設計しなければなりません。設計者は熱源に注意を集中し、モジュール レベルとラック レベルで効果的な冷却手段を使用して発熱を抑制する必要があります。

光モジュールの熱特性のテストには、従来 2 つの手法が使用されてきました。1 つは、複雑なネットワーク データ ジェネレーターを使用して高速 (10Gbps) リンクを作成し、光モジュールの熱特性をテストする方法です。もう 1 つは、調整可能な電圧および電流を設定済みの、いわゆる「熱的に等価な」モジュールを利用して熱的状況を再現し、実際の高速データを使用せずに熱特性を評価する方法です。

どちらの手法も最適とは言えません。第 1 の手法には専門的な高速ネットワーク データ ジェネレーターが必要なため、運用にコストがかかります。他方、第 2 の手法は抽象的すぎます。熱的に等価なモジュールは、物理的スイッチング動作によって生じる温度変化を完全に再現できません。

しかし先頃、Alcatel Lucent Ireland 社のベル研究所において、筆者らのチームは、ザイリンクス Zynq® 7000 All Programmable SoC プラットフォームとザイリンクスの IP ( 知的設計資産 ) コアを使用して同じジョブを実行することにより、このプロセスを根本的に簡略化しました。この記事では、筆者らがどのようにテストを簡略化したかを詳しく説明します。

## 設計前の分析

この種の熱テストの基本的要件は、XFP 光トランシーバーを 10Gbps データで継続的に刺激しながら、IR カメラを使用して温度変化の追跡と特性評価を行うことです。

筆者は、開発ホストとしてザイリンクス ZC706 評価ボードを選びました。これは、ZC706 ボードのメイン デバイスである Zynq-7000 SoC XC7Z045 (スピード グレード 2) 上の GTX トランシーバーによって、シングルラインの 10Gbps データ伝送を簡単に実現できるからです。この Zynq SoC デバイスは、ARM® コアをベースとするプロセッシング システム (PS) と Kintex®-7 FPGA プログラマブル ロジック (PL) ファブリックを統合した製品です。当初は PL ダイ側のリソースだけで 10Gbps 全 / 半二重データ伝送を処理できます。将来特定のユーザー データ パターンが必要になったら、PS を使用してそのパターンを生成できます。

筆者らの熱テスト グループは、光トランシーバーのハウジングとして Finisar 社の XFP 評価ボードを用意しました。この FDB-1022 評価ボードは、最先端の 10Gbps XFP 光トランシーバーを評価するための高性能ホストで、差動データ入力および出力用の SMA コネクタを搭載しています。このボードは、光トランシーバー モジュールへのクロック供給用に SMA コネクタを介して 1/64 クロック (すなわち、156.25 MHz = 10GHz/64) を直接接続するようにコンフィギュレーションできます。

## システム デザイン

筆者は、7 年間におよぶ FPGA の 開発経験から、できる限り多くのザイリンクス コアを使用することで設計サイクルを大幅に短縮できることを学びました。筆者はこのデザインにも同じ戦略を採用し、Integrated Bit Error Ratio (IBERT) コアから設計を始めました。このコアを使用してパターン生成と検証を実行し、Zynq SoC 上の GTX トランシーバーを評価できます。次に、デザインを適切に配線するために、ミックスド モード クロック マネージャー (MMCM) コアをベースとする位相整合されたクロック分配ユニットを作成し、FPGA ファブリック上の GTX トランシーバーと XFP 評価ボード上の光トランシーバーに同時にクロック供給しました。図 1 にシステム図を示します。

このデザイン プロジェクトでは、ザイリンクスの古い ISE® Design Suite ツールを使用して、次の 3 つの手順で作業を進めました。

第 1 の手順では、CORE Generator™ ツールを使用して IBERT コアを作成しました。IBERT 7 シリーズ GTX (ChipScope™ Pro) IBERT コアの主要な設定の一部を次に示します。このデザインでは、IBERT のシステム クロッキングはボード上の外部クロック ソース (P pin location = H9、N pin location = G9 の 200MHz 差動クロック) から入力されます。GTX のクロッキング モードは独立クロッキング モードで QUAD 111、ライン レートは Max Rate = 10Gbps に設定しました。GTX のリファレンス クロックは、Refclk = 156.25 MHz および Refclk source = MGTREFCLK1 111 に設定しました。

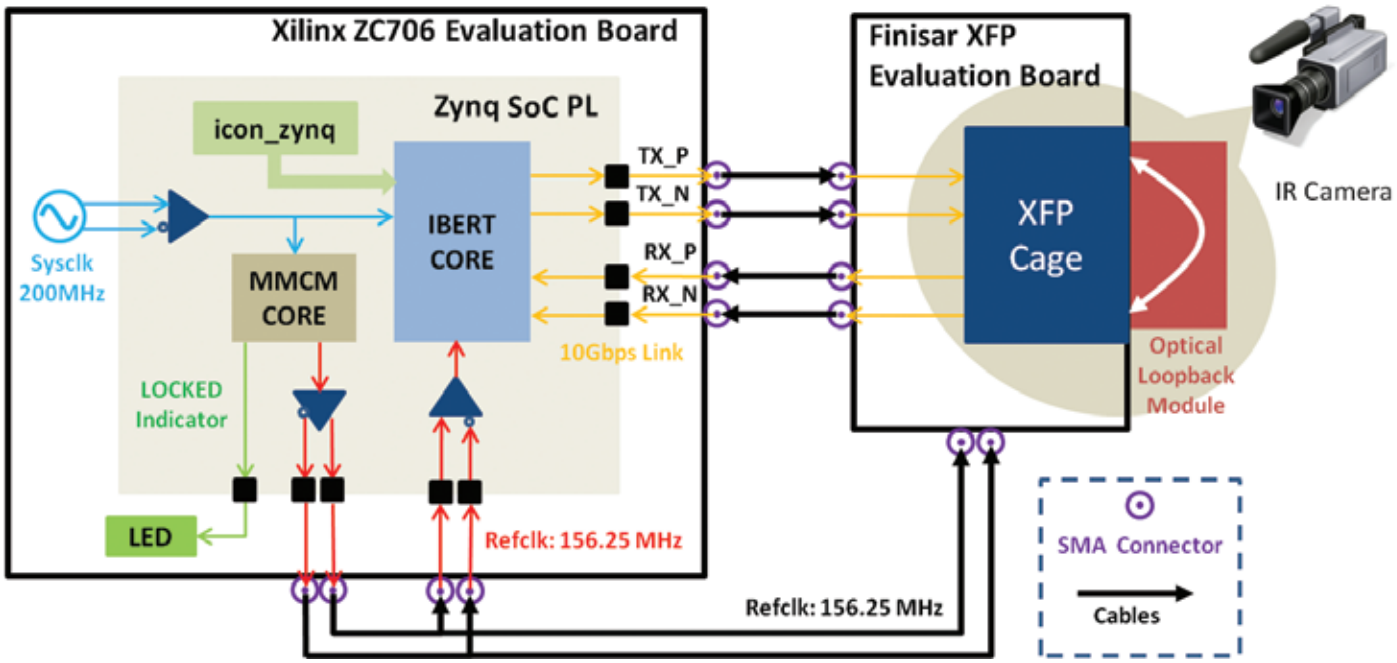



図 1 – 提案されたシステムと接続例のブロック図

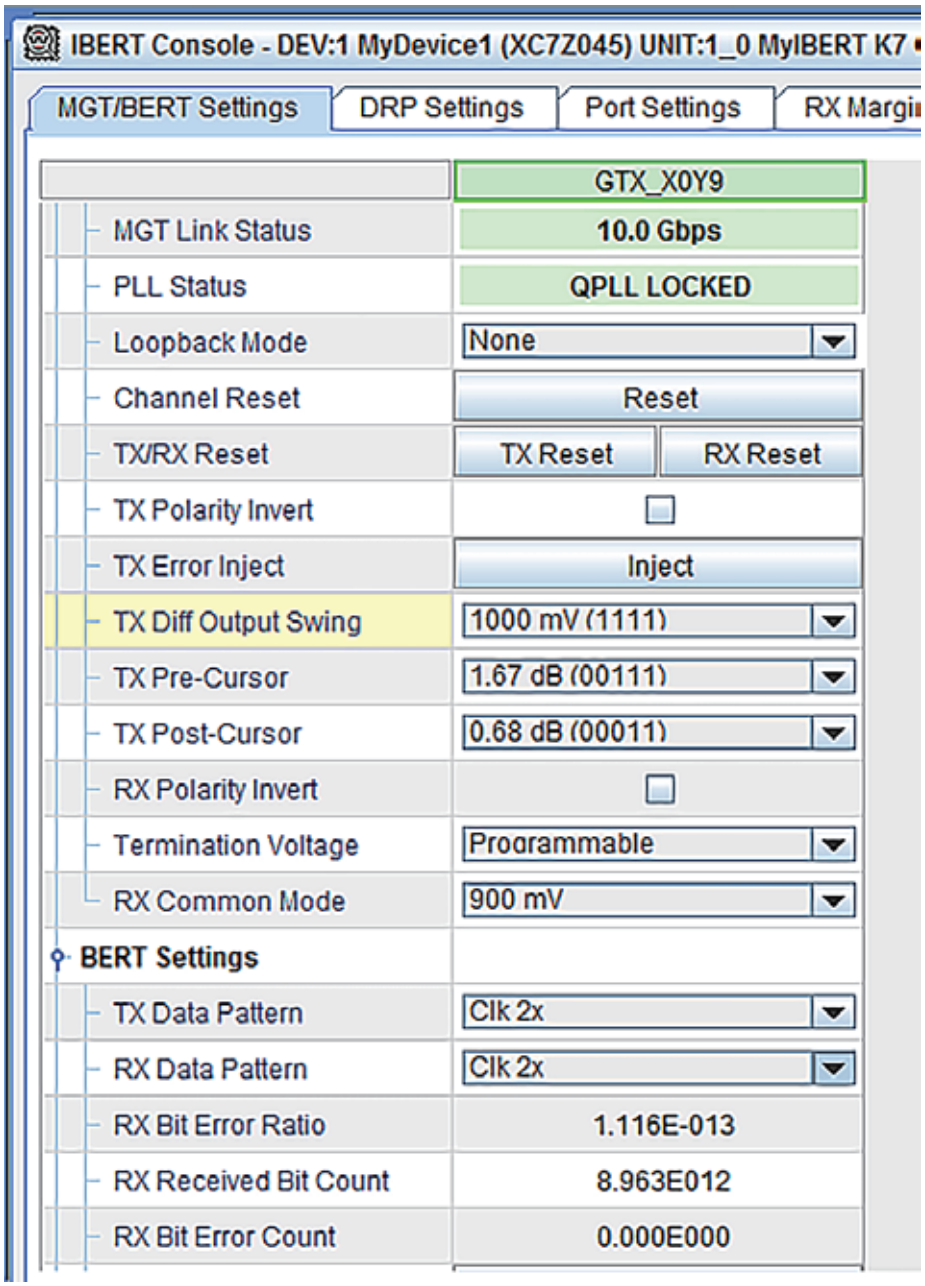



図 2 – ChipScope Pro 画面のスナップショット

第 2 の手順では、CORE Generator ツールを使用して MMCM コアを作成しました。CORE Generator ツールの Clocking Wizard を正しく設定する必要があり、そのためにクロック フィーチャを周波数合成および位相アライメントとして設定しました。入力クロックはボード上のシステム クロック (200MHz) と同じでなければなりません。ターゲット派生クロックは、50% のデューティ サイクルで 156.25MHz に設定しました。MMCM コアの制御および表示用に、2 つの追加の信号 (RESET および LOCKED) を使用しました。

第 3 の手順では、ザイリンクスのツールを使用してすべてのコンポーネントを組み立てました。このプロジェクトでは ISE Design Suite 14.4 を使用しましたが、チップの性能を最大限に引き出すために、今後どこかの時点で Vivado® Design Suite に切り替える予定です。

筆者はまず ISE ツール内で新規プロジェクトを作成し、IBERT コア フォルダー (example_ibert_gtx.vhd、ibert_ gtx_top.ucf、ibert_core.ngc、icon_zynq.ngc) を ISE プロジェクトに移動しました。次に、MMCM コア フォルダー ( 手順 2) から mmcm_core.vhd を ISE プロジェクトに追加しました。次に、example_ ibert_gtx.vhd を最上位モジュールとして使用して、mmcm_core をインスタンシエートし、3 つの新しい信号 (CLK_ OUTPUT_P、CLK_OUTPUT_N、LED_REFCLK) をデザインに追加して、対応するピン割り当てを ibert_gtx_top.ucf 内で行いました。

## システム テスト

.bit ファイルの生成後、FPGA デザインが 10Gbps リンクを使用して XFP 光トランシーバーを刺激する準備が完了しました。筆者は 2 つのボードを接続して ( 図 1 を参照 )、ChipScope Pro アナライザーを開き、新たに作成された .bit ファイルを使用してデバイスをコンフィギュレーションしました。次に、IBERT コンソールをダブルクリックして、新しいグラフィカル ユーザー インターフェイスを表示しました ( 図 2 を参照 )。この画面で、Clk 2x (1010…) などの定義済みデータ パターンと PRBS ( 擬似ランダム バイナリ シーケンス ) を調整し、光トランシーバーの熱的性能を徹底的に評価できます。

ザイリンクス コアと ZC706 評価ボードを組み合わせて使用すれば、高速光トランシーバー評価用のテスト プラットフォームを簡単に構築できます。このデザインでは 1 個の XFP モジュールの評価の例を示しましたが、読者はこの設計手法をそのまま応用して、複数の光トランシーバー モジュールのテスト用のロジック コアを迅速に作成できます。

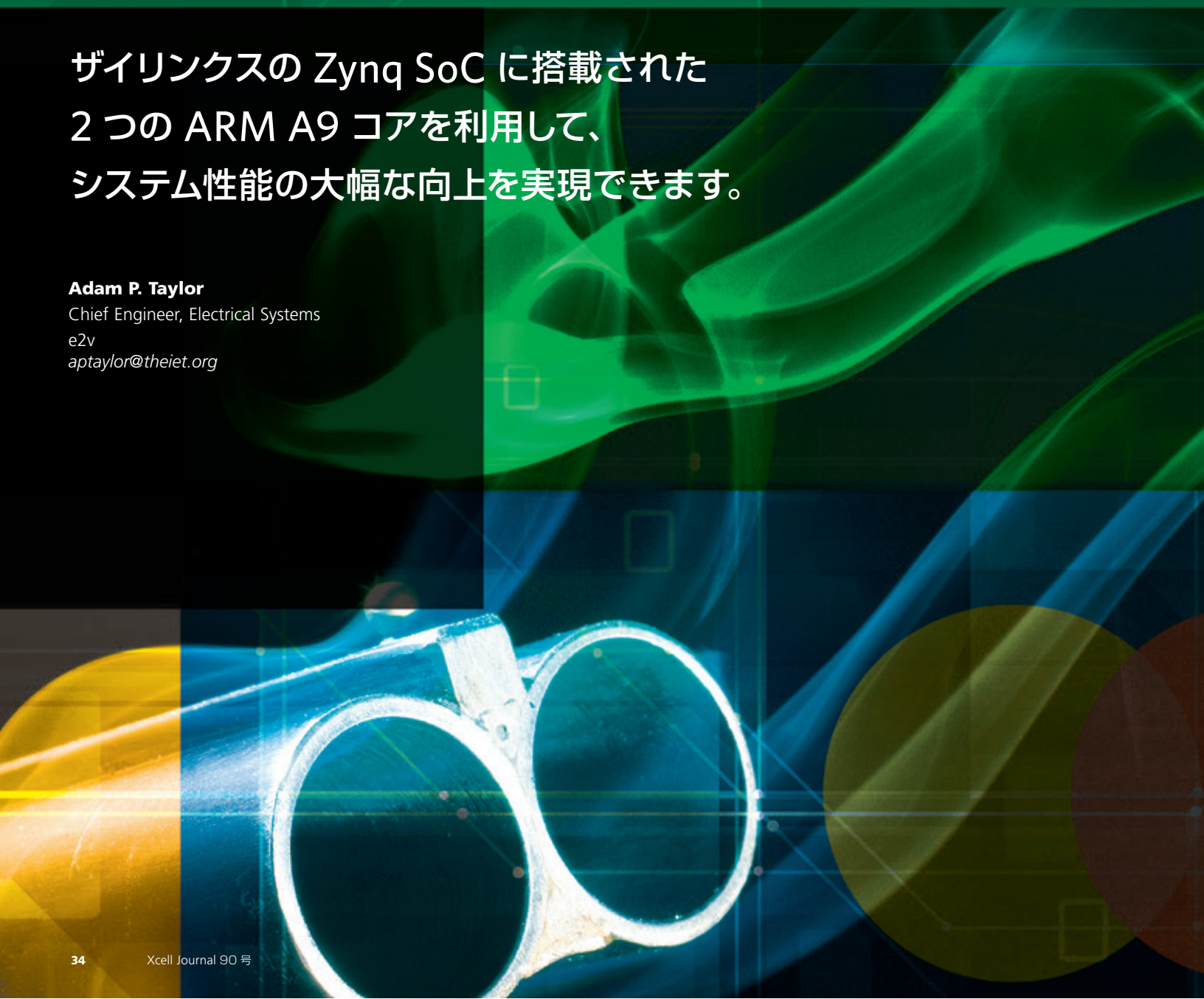
詳細情報は、lei.guan@alcatel-lucent.com までお問い合わせください。

# A Double-Barreled Way to Get the Most from Your Zynq SoC

# 2つのコアを利用して Zynq SoC の性能を最大限に引き出す方法

ザイリンクスの Zynq SoC に搭載された
2 つの ARM A9 コアを利用して、
システム性能の大幅な向上を実現できます。

**Adam P. Taylor**
Chief Engineer, Electrical Systems
e2v
*aptaylor@theiet.org*



ザイリンクスの Zynq®-7000 All Programmable SoC の多くの利点の 1 つとして、2 つの ARM® Cortex™ -A9 プロセッサをボード上に搭載していることが挙げられます。しかし、多くのベアメタル アプリケーションや簡単なオペレーティング システムは、Zynq SoC のプロセッシング システム (PS) 内の 2 つの ARM コアのうち 1 つしか使用しません。このようなデザインの選択では、システム性能が制限される恐れがあります。

開発中のアプリケーションによっては、両方のプロセッサでベアメタル アプリケーションを実行したり、それぞれのプロセッサで異なるオペレーティング システムを動作させたりしなければならない場合があります。たとえば、第 1 のプロセッサはベアメタル /RTOS アプリケーションを使用して重要な計算を行い、第 2 のプロセッサは Linux を使用して HMI と通信機能を提供することが考えられます。

## マルチプロセッシングとは

これらのシナリオは、いずれもマルチプロセッシングの例です。簡潔に定義すると、マルチプロセッシングとは、システム内で 2 つ以上のプロセッサを使用することです。マルチプロセッシング アーキテクチャでは、( 常に必要とは限りませんが ) 複数の命令を同時に実行することが可能です。

マルチコア プロセッシングには、対称型と非対称型の 2 種類があります。

対称型マルチプロセッシングは、負荷を複数のコアに分散することにより、複数のソフトウェア タスクを同時に実行できるようにします。非対称型マルチプロセッシング (AMP) は、特定のアプリケーションまたはタスク用に、複数の異なるプロセッサを使用するか、または複数の同一プロセッサ上で異なるアプリケーションを実行します。

Zynq SoC の 2 つのコアとベアメタルまたは異なるオペレーティング システムを組み合わせて使用するのが、定義上、非対称型マルチプロセッシングの例になります。Zynq SoC 上の AMP は、次の組み合わせを処理できます。

- コア 0 およびコア 1 上の異なるオペレーティング システム
- コア 0 上のオペレーティング システムとコア 1 上のベアメタル ( またはその逆 )
- 異なるプログラムを実行する、コア 0 およびコア 1 上のベアメタル

Zynq SoC 上で AMP システムを作成する必要がある場合は、次の事実を考慮に入れる必要があります。すなわち、ARM プロセッサ コアはプライベート リソースと共有リソースの両方を合わせて備えており、それらを適切に扱わなければなりません。2 つのプロセッサは、プライベート リソースとして、L1 命令キャッシュおよびデータ キャッシュ、タイマー、ウォッチドッグ、割り込みコントローラー ( 共有割り込みとプライベート割り込みの両方を制御する ) を備えています。また共有リソースも多数あり、一般的な例として、I/O ペリフェラル、オンチップ メモリ、Interrupt Controller Distributor (ICD)、L2 キャッシュ、DDR メモリ内に置かれるシステム メモリなどが挙げられます ( 図 1 を参照 )。これらのプライベート リソースと共有リソースは、注意深く管理する必要があります。

各 PS コアは専用の割り込みコントローラーを持ち、ソフトウェア割り込みを使用して、それ自身、もう 1 つのコア、または両方のコアに割り込みをかけることができます。これらの割り込みは、ARM の分散割り込みコントローラー技術によって分配されます。

各コアで実行中のプログラムは DDR メモリ内に置かれるため、これらのアプリケーションが適切に分割されるよう、よく注意する必要があります。

## AMP の起動

Zynq SoC 上で AMP を起動するのに必要な主要コンポーネントは、ブート ローダーです。ブート ローダーは、最初のアプリケーションをメモリにロードした後、第 2 の実行ファイルを探します。ザイリンクスは、便利なアプリケーション ノートおよびソース コードを XAPP1079 で提供しています。この文書には、AMP システムの作成に使用できる変更済みファースト ステージ ブート ローダー (FSBL) および変更済みスタンドアロン OS が付属しています。

最初の手順では、このアプリケーション ノートに付属の ZIP ファイルをダウンロードした後に、2 つの要素 (FSBL と OS) を希望の作業ディレクトリに抽出します。次に、SRC “design” というフォルダーの名前を変更する必要があります。ここで、変更済み FSBL と変更済みスタンドアロン OS の両方を含むこれらの新しいファイルの存在を、ソフトウェア開発キット (SDK) に認識させる必要があります。したがって、次の手順では、SDK がこれらのファイルの存在を

認識するように、SDK リポジトリを更新します。

これを実行する手順は簡単です。SDK 内で [Xilinx tools] メニューの下にある [repositories] → [new] をクリックし、<ユーザーの作業ディレクトリ >\app1079\design\work\sdk_repo に移動します ( 図 2 を参照 )。

## プロセッサ間の通信

AMP デザイン用のアプリケーションを作成する前に、必要に応じて、アプリケーション同士が通信する方法について検討する必要があります。最も簡単な方法は、オンチップ メモリを使用することです。Zynq SoC は、次の 4 つのソースの何れかからアクセス可能な 256 キロバイトのオンチップ SRAM を備えています。

- いずれか一方のコアから、スヌープ制御ユニット (SCU) を介して
- AXI ACP ( アクセラレータ コヒーレンシ ポート ) を使用するプログラマブル ロジックから、SCU を介して
- 高性能 AXI ポートを使用するプログラマブル ロジックから、オンチップ メモリ (OCM) インターコネクトを介して
- 中央インターコネクトから、OCM を介して

オンチップ メモリの読み出しと書き込みが可能なさまざまなソースがあるため、OCM の動作を使用前に詳しく理解しておくことが特に重要です。

OCM にアクセスするリソースは複数あるので、アービトレーションと優先順位の形式を定義しておくのが賢明でしょう。最小のレイテンシはスヌープ制御ユニット (SCU) ( プロセッサ コアまたは AXI ACP インターフェイスのいずれか ) によって要求されるため、これらのソースからの SCU 読み出しが最高の優先順位を持ち、次に SCU 書き込み、次に OCM インターコネクトの読み出しと書き込みの順になります。ユーザーは、オンチップ メモリ制御レジスタで SCU 書き込みの優先順位を low に設定することで、SCU 書き込みと OCM インターコネクトへのアクセスの優先順位を逆にすることができます。

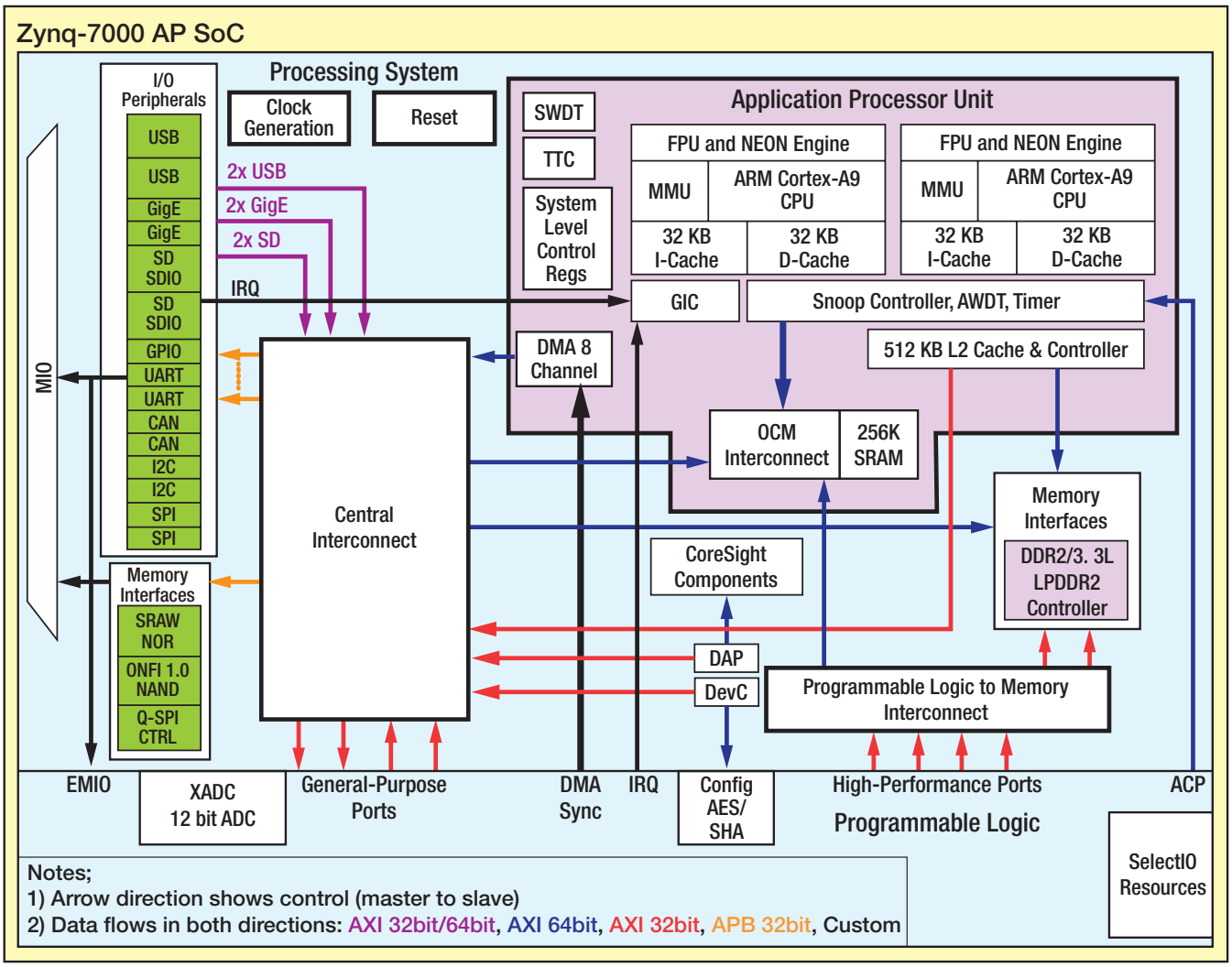



図 1 – Zynq SoC のプロセッシング システム (PS)、プライベート リソースと共有リソースを示す

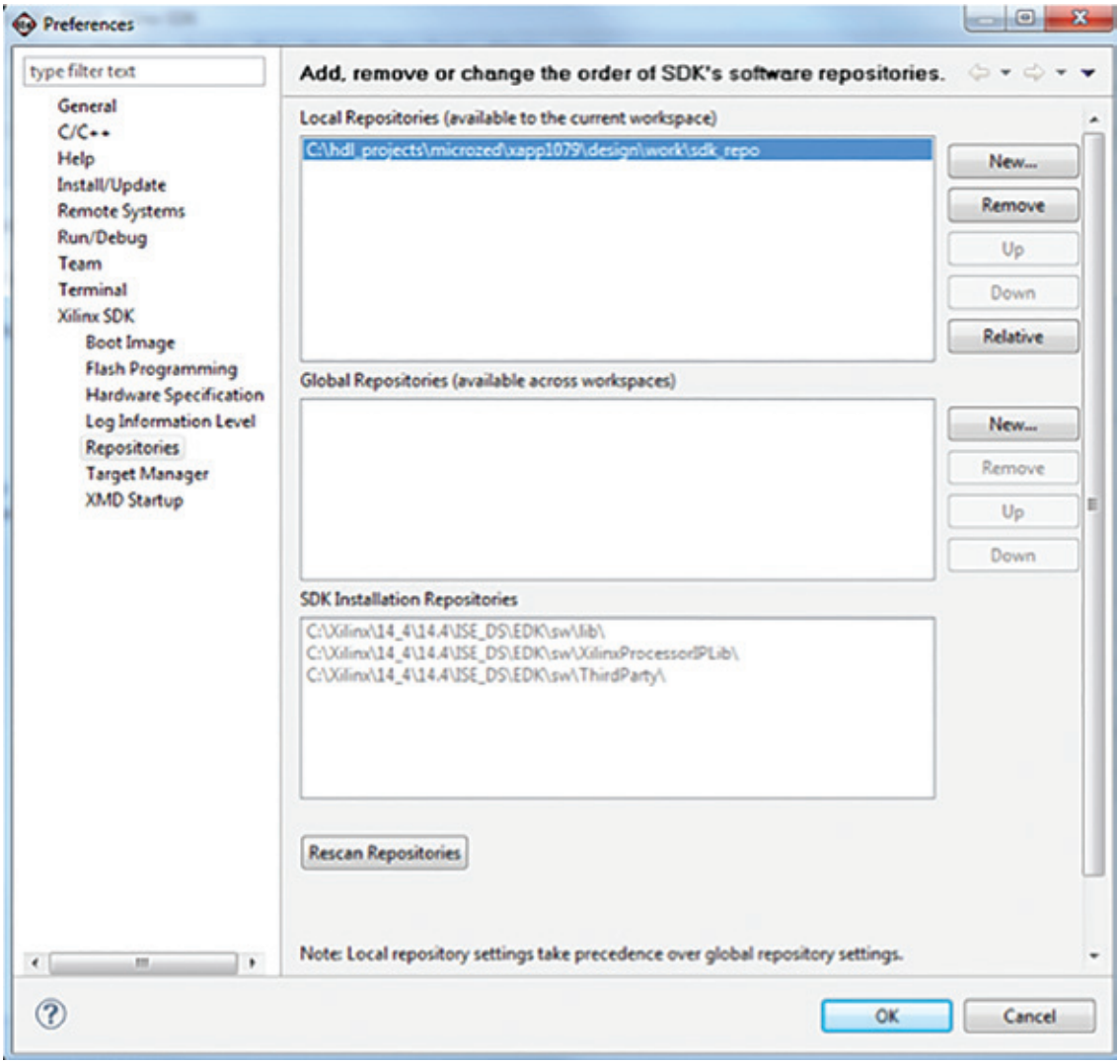



図 2 — リポジトリへの新しいファイルの追加

OCM それ自体は 128 ビット ワード単位で構成され、PS のアドレス空間の異なる位置にある 4 つの 64 キロバイト領域に分割されます。初期コンフィギュレーションでは、最初の 3 つの 64 キロバイト ブロックがアドレス空間の最初に配置され、最後の 64 キロバイト ブロックがアドレス空間の最後に配置されます ( 図 5)。

## 簡単なオンチップ メモリの例

ザイリンクスの I/O 関数を使用して OCM にアクセスし、選択したメモリ アドレスに対して読み出しと書き込みを実行できます。Xil_IO.h ファイルに含まれているこれらの関数を使って、CPU アドレス空間内の 8 ビット /16 ビット /32 ビットの char 型 /short 型 /int 型データの格納とアクセスが可能です。これらの関数を使用するには、単にアクセスするアドレスとそのアドレスに格納する値を指定する必要があります。書き込みの場合は、たとえば次のようになります。

```
Xil_Out8(0xFFFF0000,0x55);
read_char = Xil_In8(0xFFFF0000);
```

特に複数の設計者が異なるコア プログラムを開発している場合に、この手法を使って、2 つのアドレスがいずれもオンチップ メモリ内の同じ位置をターゲットとしていることを保証するには、共通のヘッダー ファイルを用意するのが良い方法と言えます。たとえば、このファイルには、特定の転送の対象となるアドレスのマクロ定義が含まれます ( 次の例を参照 )。

```
#define LED_PAT 0xFFFF0000
```

代替的な手法として、両方のプログラムがポインターを使用してメモリ位置にアクセスする方法があります。これを行うには、次のように、マクロを使用して ( 通常は C 言語で ) 定数のアドレスを指すポインターを定義します。

```
#define LED_OP (*(volatile
  unsigned int *)(0xFFFF0000))
```

この場合、このアドレスを示すもう 1 つのマクロ定義を使用して、アドレスが両方のアプリケーション プログラムに共通であることを保証することもできます。この手法では、ザイリンクス I/O ライブラリを使用する必要はなく、ポインターを介して簡単にそのアドレスにアクセスできます。

## プロセッサ間の割り込み

Zynq SoC の各コアは、ソフトウェアで生成される 16 の割り込みを備えています。前述のように、各コアは、それ自身、もう 1 つのコア、または両方のコアに割り込みをかけることができます。ソフトウェア割り込みの使い方は、ハードウェア割り込みの使い方とそれほど違いはありませんが、もちろん割り込みをトリガーする方法は異なります。ソフトウェア割り込みを使用する場合、割り込みをかけられるアプリケーションは、予想されるメモリ位置で更新された値をポーリングする必要はありません。

ハードウェア割り込みの場合と同様に、両方のコア内で汎用割り込みコントローラー (Generic Interrupt Controller) をコンフィギュレーションする必要があります。詳細は、Xcell Journal 日本語版 87 号の 34P「Zynq SoC の割り込み使用方法」を参照してください。

これで、xscugic.h 内に提供される XScuGic_SoftwareIntr 関数を使用して、割り込みをかける側のコア内でソフトウェア割り込みをトリガーできます。このコマンドは、指定されたコアに対してソフトウェア割り込みを発行します。割り込みをかけられたコアは、適切な動作を実行できます。

```
XScuGic_SoftwareIntr(<GIC
Instance Ptr>, <SW Interrupt
ID>, <CPU Mask>)
```

## アプリケーションの作成

リポジトリにファイルを追加したら、次の段階では、AMP ソリューションの 3 つの重要な要素である AMP ファースト ステージ ブート ローダー (FSBL)、コア 0 のアプリケーション、コア 1 のアプリケーションを生成します。これらのアイテムのそれぞれに、異なるボード サポート パッケージを生成する必要があります。

最初に、SDK を使用して新しい FSBL を作成します。[file new application project] を選択すると、AMP をサポートする FSBL プロジェクトを作成できます。これは通常の FSBL を作成する際のプロセスと変わりありません。ただし、ここでは [Zynq FSBL for AMP] テンプレートを選択します ( 図 3 を参照 )。

AMP FSBL の作成に続いて、第 1 のコアのアプリケーションを作成します。必ずコア 0 と希望のオペレーティング システムを選択し、コア 0 専用の BSP を作成します ( 図 4 を参照 )。

アプリケーションを作成したら、アプリケーションの実行が開始される DDR メモリ内の位置を正しく定義する必要があります。

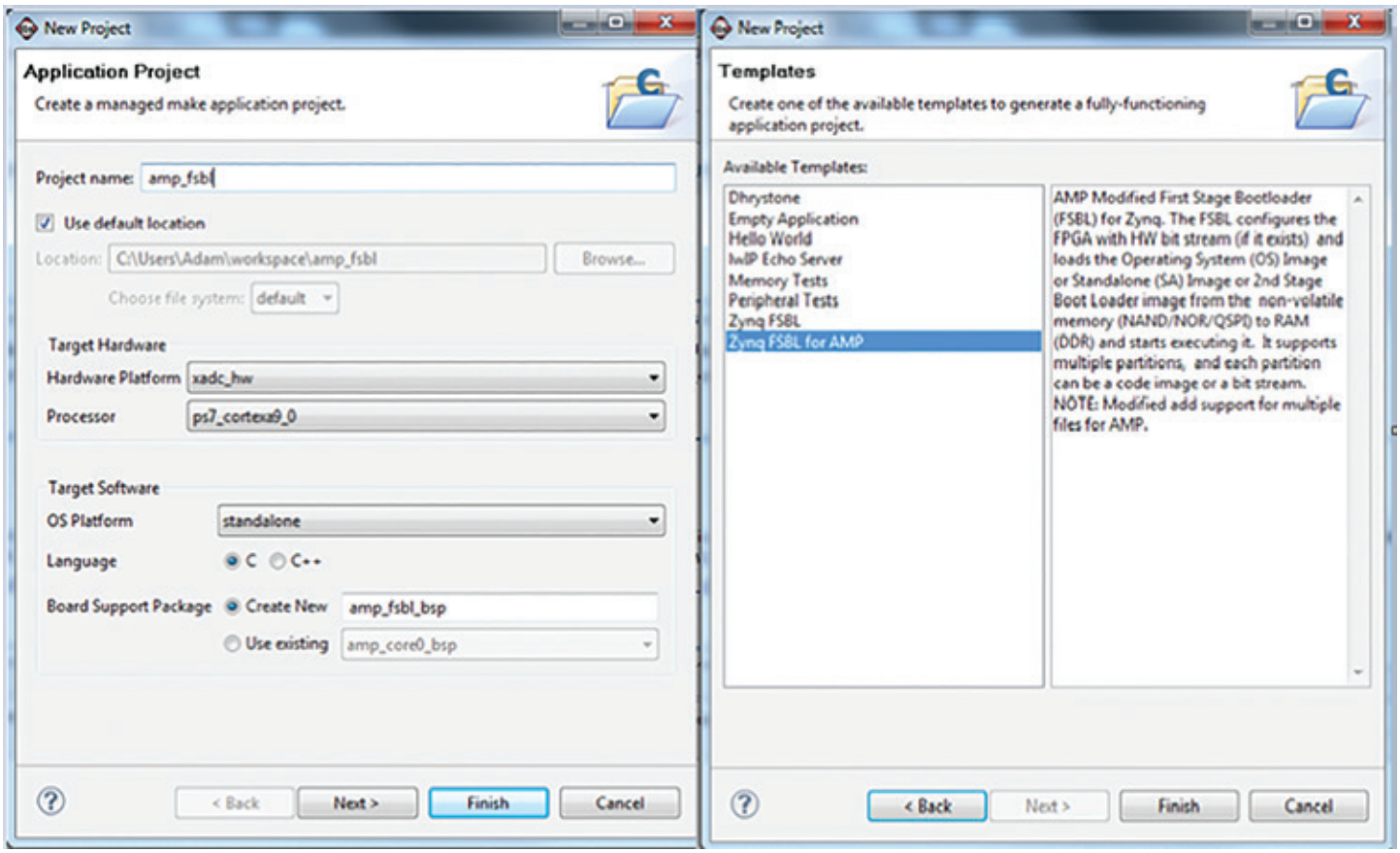



図 3 – AMP デザイン用のファースト ステージ ブートローダーの選択

これを行うには、図 5 に示すようにリンカー スクリプトを編集し、DDR のベース アドレスとサイズを表示します。DDR メモリ内でコア 0 アプリケーションとコア 1 アプリケーションを適切に分割しないと、一方が他方を誤って破壊するリスクが生じるため、この作業は重要です。

分割が完了したら、コア 0 上で実行するアプリケーションを作成できます。コア 0 は AMP システム内で監督機能を持つコアなので、コア 0 のアプリケーションを先に作成します。コア 0 はコア 1 のアプリケーションの実行を開始しなければなりません。コア 0 のアプリケーション内に図 6 に示すコード セクションを追加する必要があります。このコードは、オンチップ メモリ上のキャッシュを無効にし、コア 1 プログラムの開始アドレスを、コア 1 がアクセスするアドレスに書き込みます。コア 0 が Set Event (SEV) コマンドを実行すると、コア 1 はコア 1 プログラムの実行を開始します。

次の手順では、コア 1 用の BSP を作成します。PS のスヌープ制御ユニットの再初期化を防ぐ、変更済みのスタンドアロン OS (standalone_amp、図 7 を参照 ) を使用する必要があります。したがって、コア 0 のときとは異なり、プロジェクトの作成中に BSP の自動生成を有効にしないでください。CPU 選択オプションで、必ずコア 1 を選択します。

これでコア 1 用の BSP が作成されたら、コア 1 上で実行するアプリケーションの作成に進む前に、BSP の設定を変更する必要があります。この作業は非常に簡単で、BSP のドライバー セクションのコンフィギュレーションに追加のコンパイラ フラグ -DUSE_ AMP=1 を追加するだけです。

この手順が完了したら、コア 1 のアプリケーションを作成できます。プロセッサとして必ずコア 1 を選択し、コア 1 用に作成した BSP を使用します。ここでも、新しいアプリケーションを作成したら、コア 1 プログラムの実行が開始される DDR メモリ内の適切なメモリ位置をもう一度定義する必要があります。これを行うには、前回コア 0 で行ったように、コア 1 のアプリケーションのリンカー スクリプトを編集します。コア 0 のときと同様に、コア 1 のアプリケーション内でオンチップ メモリ上のキャッシュを無効

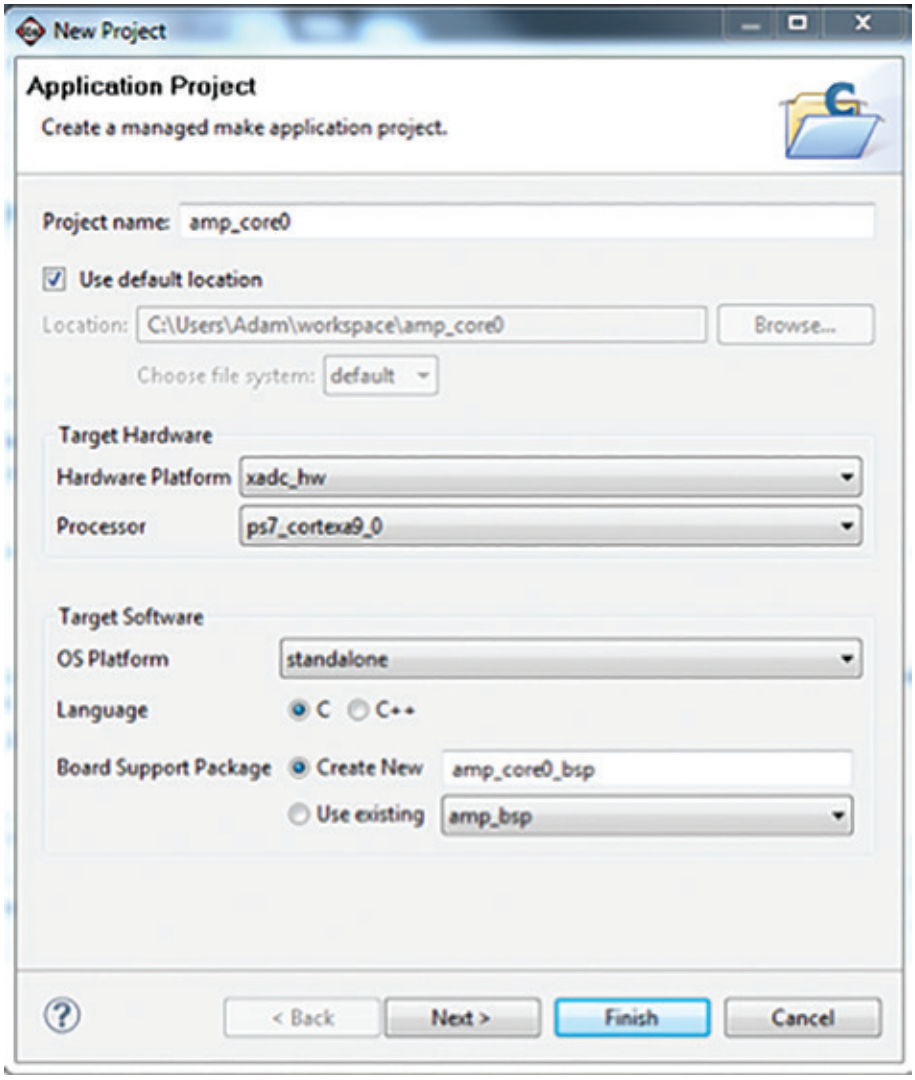



図 4 – コア 0 用のアプリケーションと BSP の作成

Available Memory Regions

| Name | Base Address | Size |
|---|---|---|
| ps7_ddr_0_S_AXI_BASEADDR | 0x00100000 | 0x00100000 |
| ps7_ram_0_S_AXI_BASEADDR | 0x00000000 | 0x00030000 |
| ps7_ram_1_S_AXI_BASEADDR | 0xFFFF0000 | 0x0000FE00 |

図 5 – コア 0 の DDR 内の位置とサイズ

```
#include <stdio.h>
#include "xil_io.h"
#include "xil_mmu.h"
#include "xil_exception.h"
#include "xpseudo_asm.h>"
#include "xscugic.h>

#define sev() __asm__("sev")
#define CPU1STARTADR 0xffffffF0
#define COMM_VAL  (*(volatile unsigned long *)(0xFFFF0000))


int main()
{

     //Disable cache on OCM
     Xil_SetTlbAttributes(0FFFF0000,0x14de2);      // s=b1 TEX=b100 AP=bll, Domain=bllll, C=b0, B=b0
     Xil_Out32(CPU1STARTADR, 0x00200000);
     dmb(); //waits until write has finished
     sev();
```

図 6 – オンチップ メモリ上のキャッシュを無効にするコード

にして、2 つのプロセッサ間の通信に使用できるようにする必要があります。

## システムの各要素の統合

アプリケーションの作成とプロジェクトの構築が完了すると、次のコンポーネントが作成されているはずです。

- AMP FSBL の ELF
- コア 0 の ELF
- コア 1 の ELF
- AMP がインプリメントされる Zynq デバイスのコンフィギュレーションを定義する BIT ファイル

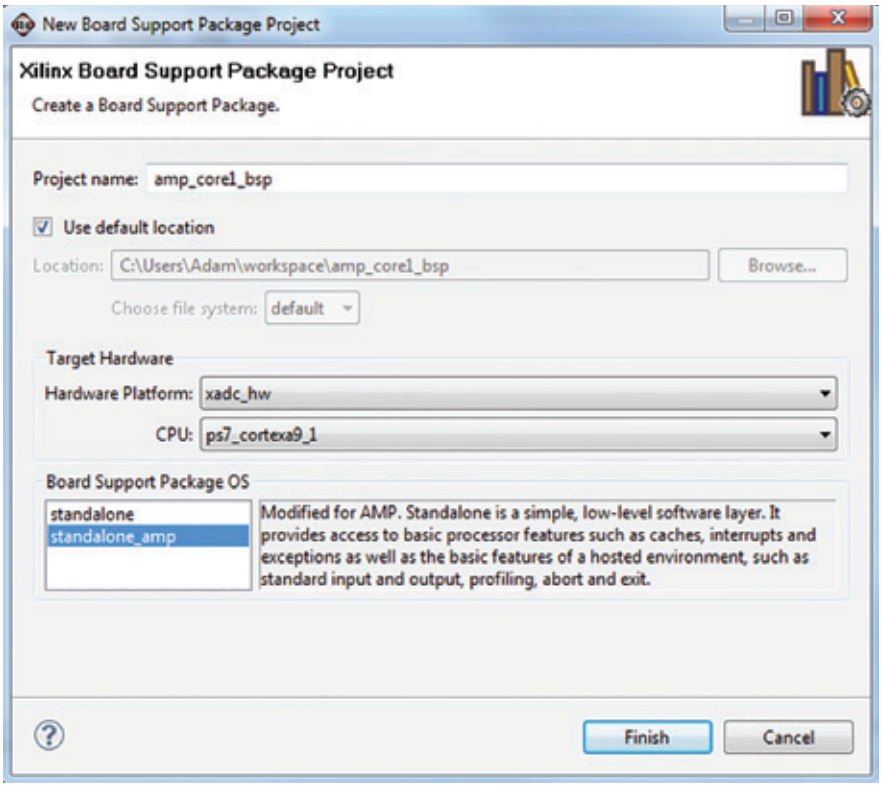



図 7 – コア 1 の BSP の作成

Zynq SoC が選択されたコンフィギュレーション メモリからブートできるようにするには、.bin ファイルを作成する必要があります。BIN ファイルを作成するには、BIF ファイルも必要です。BIF ファイルは、この BIN ファイルの作成に使用される複数のファイルとそれらの順番を定義します。SDK 内の "create Zynq" ブート イメージを使用するのではなく、ISE® Design Suite のコマンド プロンプトと、(ダウンロードされたディレクトリ \design\work\bootgen の下にある) XAPP1079 の一部として提供される BAT ファイルを使用します。このディレクトリには、BIF ファイルと cpu1_bootvec.bin が含まれます。cpu1_bootvec.bin は、変更済みの FSBL の一部として使用され、FSBL がロードするアプリケーションをこれ以上探すのを停止します。

BIN ファイルを生成するには、生成された 3 つの ELF ファイルを bootgen ディレクトリにコピーし、BIF ファイルを編集して、ファイル内の ELF の名前が正しいことを確認する必要があります (図 8 を参照)。

ここで ISE コマンド プロンプトを開き、bootgen ディレクトリに移動できます。bootgen ディレクトリで、createboot.bat を実行します。この手順により、図 9 に示すような boot.bin ファイルが作成されます。

このファイルは Zynq SoC 上の不揮発性メモリ内にダウンロードできます。Zynq SoC デバイスをブートすると、両方のコアが起動し、それぞれのプログラムを実行します。

利用可能なツールを使用すれば、Zynq SoC 上で非対称型マルチプロセシング アプリケーションを作成する手順は非常に簡単です。2 つのコア間の通信は、オンチップ メモリまたは分割された DDR 領域を使用して簡単に実現できます。

```
the_ROM_image
{

      [bootloader] amp_fsbl.elf
                   download.bit
                     amp_cpu0.elf
                     app_cpu1.elf

   //write start vector address 0xFFFFFFF0 with 0xFFFFFFF00
   //This load address triggers fsbl to continue
   [load = 0xFFFFFFF0] cpu1_bootvec.bin
}
```

図 8 – BIF ファイルの変更

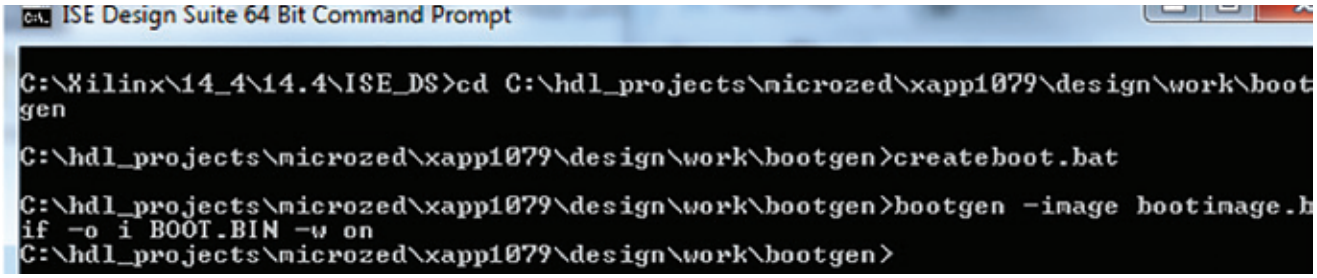



図 9 – Zynq SoC 上で動作する boot.bin ファイルの作成

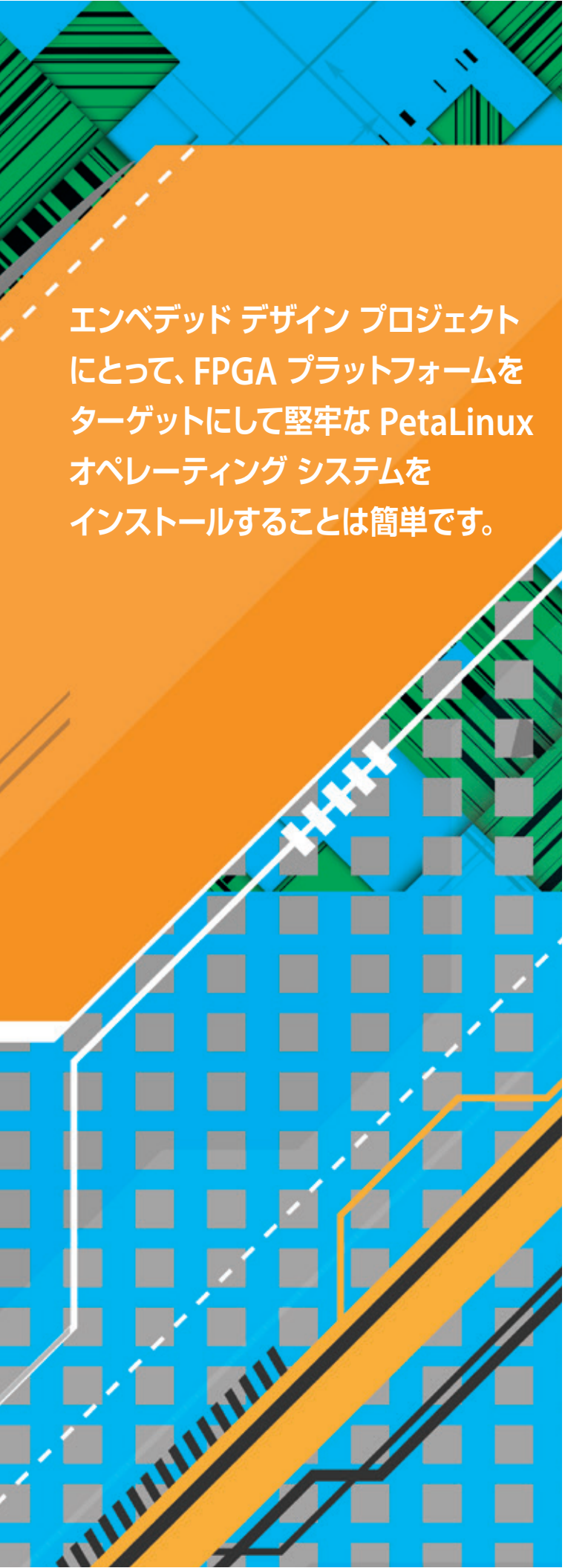
# How to Port PetaLinux Onto Your Xilinx FPGA

# ザイリンクス FPGA に PetaLinux を移植する方法

**Sweta**
Postgraduate Student, Department of Telecommunication
PES Institute of Technology, Bangalore, India
*sweta.v.walikar@gmail.com*

**Srikanth Chintala**
Research Engineer, Satellite and Wireless Group
Centre for Development of Telematics (C-DOT), Bangalore, India
*chintala@cdot.in*

**Manikandan J**
Professor and Domain Head, Signal Processing Domain
Department of Electronics and Communication (EC) and
Professor, Crucible of Research and Innovation (CORI)
PES University, Bangalore, India
*manikandanj@pes.edu*



**エンベデッド デザイン プロジェクトにとって、FPGA プラットフォームをターゲットにして堅牢な PetaLinux オペレーティング システムをインストールすることは簡単です。**

現在の FPGA は、初期の簡単なグルー ロジックから長い時間をかけて大きく進歩し、大容量ロジックと高い柔軟性によってエンベデッド デザインの中心的な位置を占めるようになりました。現在では、1 つのプログラマブル チップにシステム全体を実装できるアーキテクチャにより、ハードウェア / ソフトウェア協調設計が容易になり、ハードウェアとソフトウェア アプリケーションの統合が進んでいます。

このような FPGA ベースのエンベデッド デザインには、堅牢なオペレーティング システムが必要です。PetaLinux は組込み機器設計者に好評な OS として注目されています。PetaLinux はオープン ソースとして無償で入手可能であり、ザイリンクス MicroBlaze® CPU や ARM® プロセッサなど各種のプロセッサ アーキテクチャをサポートします。特定の FPGA に PetaLinux を移植するには、ターゲット プラットフォームに合わせて、カーネル ソース コード、ブート ローダー、デバイス ツリー、ルート ファイル システムのカスタマイズ、コンフィギュレーション、構築を行う必要があります。

筆者らのチームは、PES 大学および C-DOT の設計プロジェクトで、Kintex®-7 XC7K325T FPGA を搭載したザイリンクスの KC705 評価ボード上に PetaLinux を移植し、複数の PetaLinux ユーザー アプリケーションを実行しました。その結果、これは非常に簡単なプロセスであることがわかりました。

## PetaLinux の利点

移植の手順を詳しく説明する前に、FPGA ベースのエンベデッド システムで利用可能な各種の OS について簡単に説明しておく必要があります。FPGA 上で広く使用される OS には、PetaLinux のほか、µClinux や Xilkernel があります。µClinux は、小規模な Linux カーネルを含む Linux ディストリビューションまたは移植済みの Linux OS であり、メモリ管理ユニット (MMU) を持たないプロセッサ向けに設計されています [1]。µClinux には、ライブラリ、アプリケーション、ツール チェーンが付属しています。一方 Xilkernel は、小型で堅牢なモジュラー形式のカーネルで、µClinux よりも高度なカスタマイズが可能であり、カーネルを調整することでデザインのサイズと機能を最適化できます [2]。

一方 PetaLinux は、FPGA ベースの システム オン チップ (SoC) デザインをターゲットとする統合型 Linux ディストリビューションおよび開発環境です。PetaLinux は、コンフィギュレーション済みのブート可能なバイナリ イメージ、ザイリンクス デバイス向けに完全にカスタマイズ可能な Linux、付属の PetaLinux ソフトウェア開発キット (SDK) [3] ( コンフィギュレーション、構築、展開までの複雑な作業を自動化するツールとユーティリティを含む ) で構成されます。ザイリンクスのウェブサイトから無償でダウンロード可能な PetaLinux 開発パッケージには、各種のザイリンクス FPGA 開発キット向けに設計されたハードウェア リファレンス プロジェクトが含まれています。また、このパッケージには、ザイリンクス FPGA 用のカーネル コンフィギュレーション ユーティリティ、クロス コンパイラなどのソフトウェア ツール、ハードウェア デザイン作成ツール、数多くの設計支援ツールも含まれています。

Xilkernel の性能は µClinux を上回り [4]、PetaLinux の性能は Xilkernel よりもさらに優れていることが報告されています [5]。この理由から、また特に、ザイリンクスのターゲット ボード向けのパッケージがただちに利用可能だったため、筆者らはこのプロジェクトに PetaLinux を選びました。PetaLinux を移植するもう 1 つの利点は、ユーザーがリモート プログラミング機能を利用できることです。これは、リモート アクセスを使用して、Telnet を介して新しいコンフィギュレーション ファイル ( またはビットストリーム ファイル ) を FPGA ターゲット ボードにロードできるという意味です。

## インストールの開始

ここでは、筆者らのチームがどのように PetaLinux をインストールしたかを詳しく説明します。最初の手順では、PetaLinux パッケージ 12.12 と Kintex-7 ターゲット ボード用ボード サポート パッケージ (BSP) をダウンロードしました。PetaLinux SDK インストーラーを実行し、コンソール内で次のコマンドを使用して、同じ内容を /opt/Petalinux-v12.12-final ディレクトリにインストールしました。

```
@ cd /opt
@ cd /opt/PetaLinux-v12.12-final-full.tar.gz
@ tar zxf  PetaLinux-v12.12-final-full.tar.gz
```

次に、ザイリンクスのウェブサイトから取得した PetaLinux SDK ライセンスをコピーし、.xilinx および .Petalogix フォルダーに貼り付けました。次に、以下のコマンドを使用して適切な設定をソースし、SDK 作業環境を設定しました。

```
@ cd /opt/PetaLinux-v12.12-final
@ source settings.sh
```

作業環境が設定されたかどうかを確認するために、次のコマンドを使用しました。

```
@ echo $PETALINUX
```

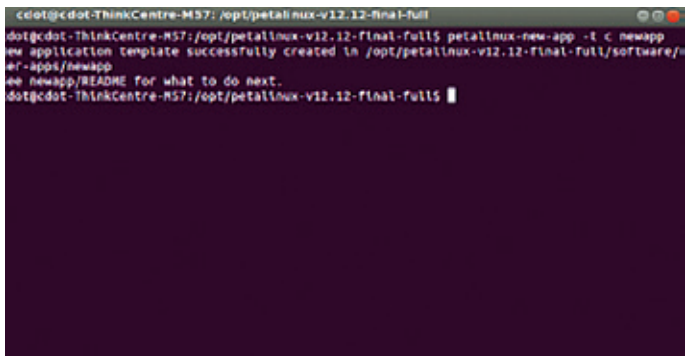



図 1 – ユーザー設定用の Linux ターミナル ウィンドウのスナップショット

環境が正しく設定されていれば、PetaLinux がインストールされたパスが表示されます。この例では、PetaLinux がインストールされたパスは /opt/PetaLinux-v12.12-final です。

次の手順では、BSP をインストールしました。BSP には、必要なデザイン ファイル、コンフィギュレーション ファイル、事前に構築済みのハードウェアおよびソフトウェア パッケージ ( 既にテスト済みで、ターゲット ボード上にダウンロードしてただちに利用可能 ) が含まれています。Quick Emulator (QEMU) システム シミュレーション環境内でのブート用のパッケージも用意されています。BSP をインストールするために、パス /opt に “bsp” という名前のフォルダーを作成し、次のコマンドを使用して KC705 BSP の ZIP ファイルをコピーしました。

```
@ cd  /opt/PetaLinux-v12.12-final-full
@ source settings.sh
@ source /opt/Xilinx/14.4/ISE_DS/settings32.sh
@ PetaLinux-install-bsp /bsp/Xilinx-KC705
    -v12.12-final.bsp
```

新しいハードウェア プラットフォームに合わせてカスタマイズされた PetaLinux システムを構築するためのソフトウェア プラットフォームは、次の 2 つの手法で作成およびコンフィギュレーションできます。第 1 の手法は、Linux ターミナルを使用して、PetaLinux コマンドをそれに対応するパス ロケーション内で使用します ( 図 1 を参照 )。第 2 の手法は、プルダウン メニューを備えた GUI を使用します ( 図 2 を参照 )。これらいずれかの手法を使用して、プラットフォームの選択、Linux カーネルのコンフィギュレーション、ユーザー アプリケーションのコンフィギュレーション、イメージの構築を実行できます。PetaLinux コンソールは PetaLinux OS のインストール後に利用可能になり、GUI は PetaLinux SDK プラグインのインストール後に利用可能になります。PetaLinux SDK プラグインのインストールが完了したら、PetaLinux Eclipse SDK 内にある PetaLinux GUI を使用してコンフィギュレーションを設定できます ( 図 2)。この GUI は、ユーザー アプリケーションおよびライブラリの開発と、PetaLinux およびハードウェア プラットフォームのデバッグ、構築、コンフィギュレーションの機能を備えています。

## ハードウェアの構築

筆者らは、このプロジェクトに Kintex-7 FPGA ベースの KC705 評価ボードを使用しました。このデザインに必要なハードウェア インターフェイスは、出力のモニター用の RS232 インターフェイス、FPGA のプログラミング用の JTAG インターフェイス、リモート プログラミング用のイーサネット インターフェイスです。PetaLinux SDK 以外に、提案されたデザインに必要なソフトウェアは、Xilinx Platform Studio (XPS) [6,7] およびザイリンクス ソフトウェア開発キット (SDK) [7] です。

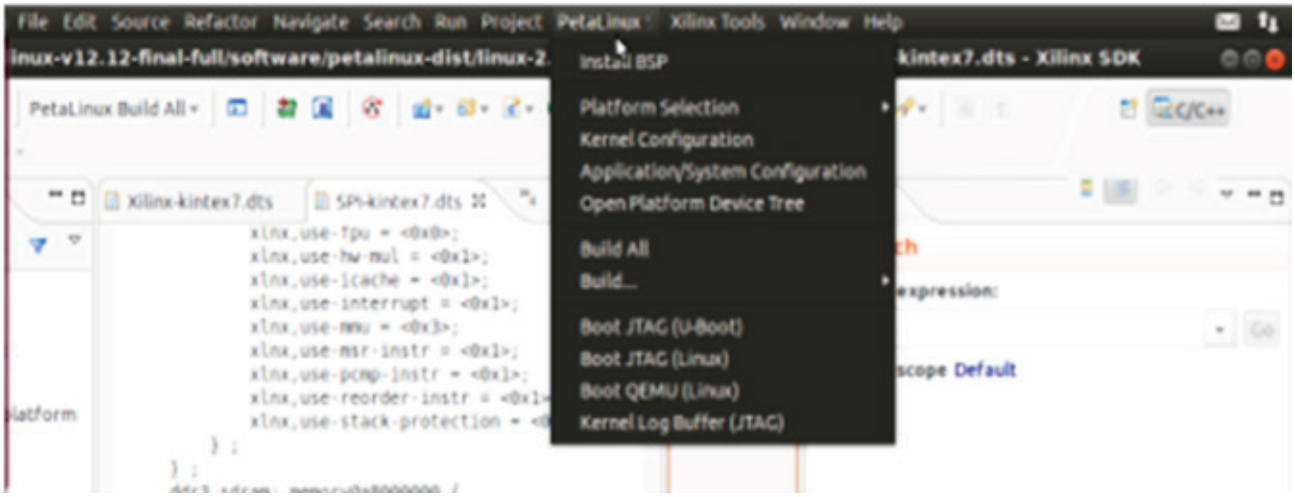



図 2 – ユーザー設定用の PetaLinux SDK メニューのスナップショット

エンベデッド デザインのハードウェア部分については、最初の作業は、XPS 内の Base System Builder (BSB) を使用して、MicroBlaze プロセッサ ベースのハードウェア プラットフォームを設計することでした。BSB を使って、ターゲット ボード上で利用可能な一連のペリフェラルを選択できます。アプリケーションの必要に応じて、ペリフェラルを追加または削除できます。提案されたアプリケーションに採用した一連のコアまたはペリフェラルは、外部メモリ コントローラーと 8 メガバイトのメモリ、割り込みに対応したタイマー、ボー レート 115,200bps の RS232 UART、イーサネット、不揮発性メモリ、LED でした。ペリフェラルの選択後、ハード

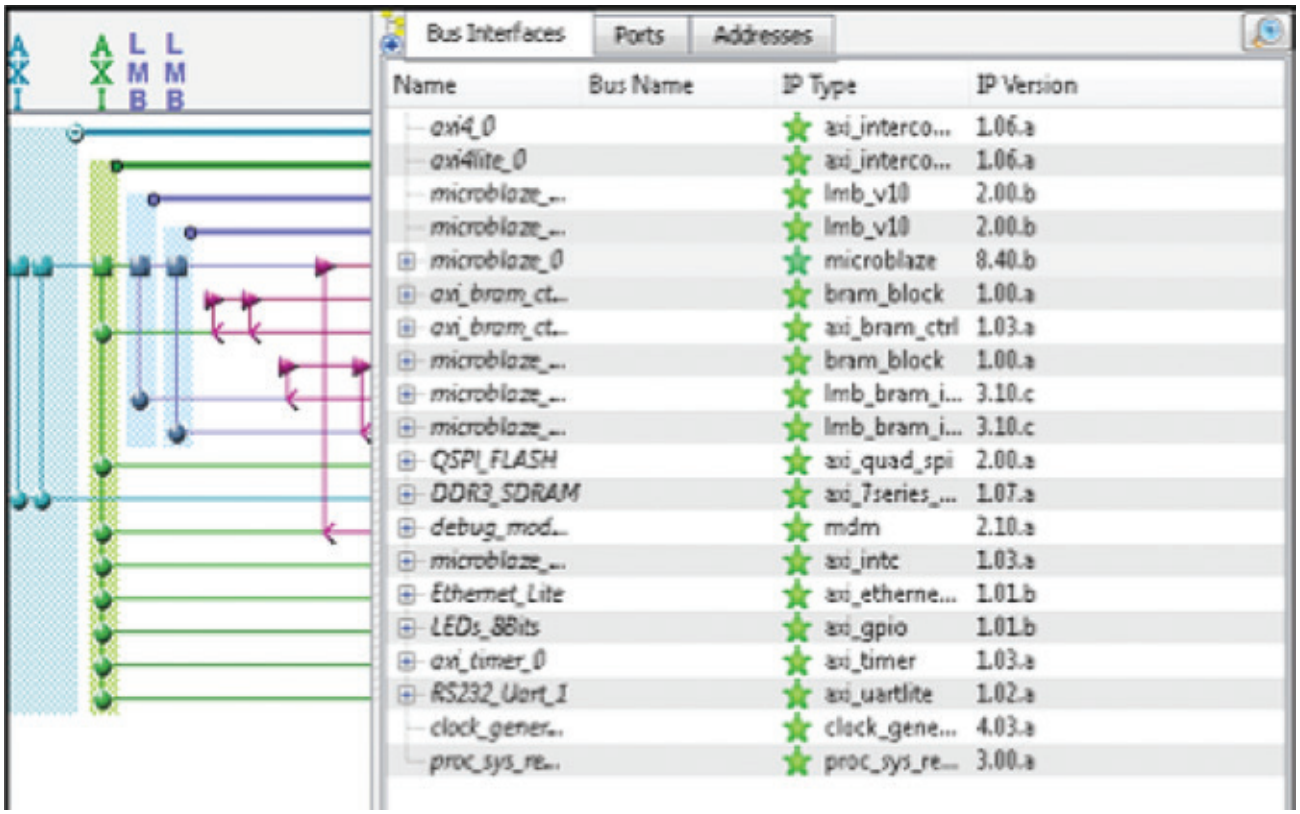



図 3 – FPGA のハードウェア コンフィギュレーション

ウェア ペリフェラルとそれに対応するバス インターフェイスを取得しました ( 図 3)。MicroBlaze プロセッサ ベースのデザインでは、PetaLinux は MMU に対応した CPU を必要とします。したがって、筆者らは、XPS ウィンドウで microblaze_0 インスタンスをダブルクリックし、MMU 付きのローエンド Linux を選択しました。

次に、3 段階の変換プロセスを使用して、ハードウェア コンフィギュレーションをビットストリームに変換しました。最初に、XPS を使用して、エンベデッド ハードウェア プラットフォームを表現するネットリストを生成しました。次に、デザインを FPGA ロジックにマッピングしました。最後に、インプリメントされたデザインを、FPGA 上にダウンロード可能なビットストリームに変換しました。XPS の最終的な出力は、system.bit ファイルと system_bd.bmm ファイルでした。

ビットストリームの生成後、SDK 内でターゲット ハードウェア プラットフォームを観察するために、ハードウェア プラットフォームの記述を SDK にエクスポートしました。エクスポートされた system.xml ファイルには、SDK がターゲット ハードウェア プラットフォーム上でのアプリケーション ソフトウェアの書き込みとデバッグに必要とする情報で構成されています。次の作業は、[Xilinx Tools] → [Repository] → [New] を使用して SDK 内に PetaLinux リポジトリを追加し、PetaLinux がインストールされたパスを選択することでした。この例では、パスは $PetaLinux/Hardware/edk_user_repository でした。

次に、[File] → [Board support package] → [PetaLinux] を使用して、PetaLinux BSP を作成しました。必要なアプリケーションに基づいて必要なドライバーを選択し、PetaLinux BSP をコンフィギュレーションしました。次に、BSP を構築し、カーネルを立ち上げるためのファースト ステージ ブート ローダー アプリケーション (fs-boot) を作成してコンフィギュレーションしました。BSP は、ハードウェアとブート アプリケーションの間の相互作用を確立します。SDK の出力は fsboot.elf です。データからメモリへの変換コマンド data2mem が利用可能です。このコマンドは、system.bit、system_bd.bmm、fsboot.elf を、最終的な FPGA ビットストリームとして機能する download.bit という名前の 1 つのビットストリーム ファイルに結合します。

この時点で、PetaLinux OS が動作する MicroBlaze コアを含むハードウェア デザインが完成しました。これで、ファースト ステージ ブート ローダー アプリケーションを使用してカーネルを立ち上げることができます。

## ソフトウェアの構築

ハードウェア プラットフォームの構築後、このハードウェアをターゲットとするカスタマイズされた PetaLinux ソフトウェア プラットフォームを、次のコマンドを使用して作成しました。

```
$ cd/opt/PetaLinuxv12.12
$  PetaLinux-new-platform -c <CPU-ARCH> -v
   <VENDOR> -p <PLATFORM>
```

このコマンドでは、-c <cpu-arch> はサポートされる CPU のタイプ ( ここでは MicroBlaze プロセッサ )、-v <vendor> はベンダー名 ( ここでは Xilinx)、-p <platform> は製品名 ( ここでは KC705) です。ソフトウェア プラットフォームのコンフィギュレーション ファイルは、PetaLinux がインストールされたディレクトリ (すなわち /opt/PetaLinuxv12.12/software/PetaLinux-dist/vendors/Xilinx/KC705) に生成されます。

ソフトウェア プラットフォームのテンプレートをハードウェアに合わせてカスタマイズするために、コマンド PetaLinux-copy-autoconfig を使用して、既存のプラットフォームのコンフィギュレーションとカーネルのコンフィギュレーションを結合しました。このコマンドは、ハードウェア コンフィギュレーション ファイル Xilinx-KC705.dts、xparameters.h、config.mk を生成します。

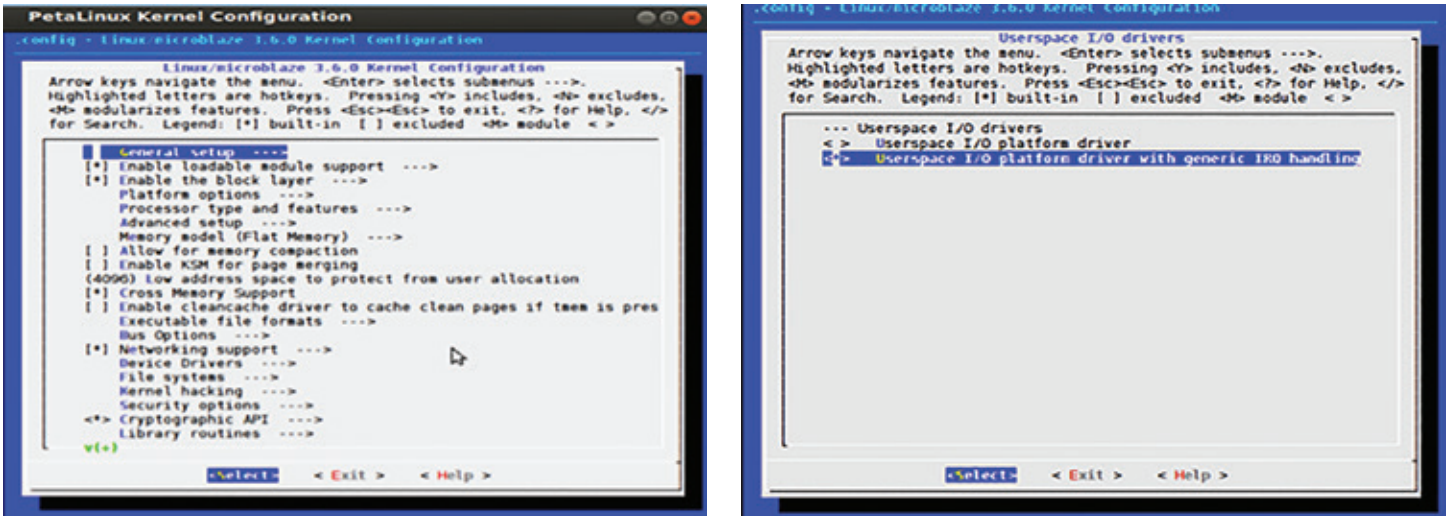



図 4 – カーネル コンフィギュレーション メニュー

GUI を使用してカーネル コンフィギュレーション メニューを開き ([PetaLinux SDK] → [Kernel Configuration])、Linux カーネルをコンフィギュレーションしました。この操作は、Linux ターミナルで次のコマンドを使用して実行することもできます。

```
$ cd /opt/PetaLinux_v12.12
$  PetaLinux-config-kernel
```

[Kernel Configuration] ポップアップ ウィンドウ ( 図 4 を参照 ) でアプリケーション用のドライバーを有効にしました。提案された作業でユーザー空間入力 / 出力 (UIO) インターフェイスを介してデバイスにアクセスするために、カーネル コンフィギュレーション メニューで UIO ドライバーを有効にしました。

カーネルをコンフィギュレーションした後、数種類のアプリケーションを設計しました。PetaLinux は、C および C++ プログラミング用のユーザー アプリケーション テンプレートを提供します [8]。これらのテンプレートにはアプリケーションのソース コードと Makefile が含まれているため、ターゲット チップ用にアプリケーションのコンフィギュレーションとコンパイルを行い、ルート ファイル システムにアプリケーションをインストールすることは簡単でした。新しい PetaLinux ユーザー アプリケーションを作成するには、GUI ([File] → [PetaLinux New Application]) を使用するか、または Linux ターミナルに次のコマンドを入力します。

```
$ cd /opt/PetaLinux_v12.12
$  PetaLinux-config-apps
```

次に、ユーザー アプリケーションにファイル名を付けました。この例では、gpio-dev-mem-test および gpio-uio-test ユーザー アプリケーションを作成し、アプリケーションの要件に基づいてテンプレート ソース コードを修正しました。

次に、GUI を使用して PetaLinux システム イメージを構築しました ( 図 2 を参照 )。この操作は、次のように Linux ターミナルで make コマンドを使用して実行することもできます。

```
$ cd $PETALINUX/software/ PetaLinux-dist
$ make
```

これで、OS とカスタマイズ済みのユーザー アプリケーションを含むソフトウェア プラットフォームを、既に説明したハードウェア デザインと一緒に使用する準備ができました。

## デバイス上での PetaLinux の動作テスト

PetaLinux のブートアップ プロセスは次のとおりです。MicroBlaze プロセッサがブロック RAM 内のコードを実行します。ファースト ステージ ブート ローダー (fs-boot) が基本ハードウェアを初期化し、fs-boot.elf を実行し、フラッシュ パーティション内で U-Boot (Universal Bootloader) のアドレスを検索します (U-Boot のアドレスは fs-boot のコンフィギュレーション中に指定されるので )。fs-boot がフラッシュ内の U-Boot パーティションから U-Boot イメージをフェッチして、デバイスの DDR3 メモリに送信し、カーネルを実行します。ブートに必要なすべてのイメージを構築したら、JTAG、イーサネット、または Quick Emulator (QEMU) を介して、ハードウェア上でそれらのイメージをテストできます。QEMU は、PetaLinux OS を実行できるエミュレーター / 仮想マシンです [9]。ここでは、3 つのソリューションすべてのブート手法を説明します。

JTAG は、FPGA デザインのプログラミングとテストに用いられる伝統的な手法です。JTAG を使って FPGA をプログラムするために、筆者らはプルダウン メニュー [Xilinx Tool] → [Program the FPGA] を使用し、既に生成した download.bit ファイルをダウンロードしました。次に、GUI ([PetaLinux SDK] → [BOOT JTAG [Linux]]) を使用して、ボード上にイメージをダウンロードしました ( 図 2 を参照 )。Linux ターミナルで次のコマンドを使用することもできます。

```
$ cd/opt/PetaLinux_v12.12/software/
   PetaLinux-dist
$  PetaLinux-jtag-boot -i images/image.elf
```

あるいは、U-Boot を使用して間接的カーネル ブートを実行し、PetaLinux をブートすることもできます。GUI ([PetaLinux SDK] → [BOOT JTAG [U-Boot]]) または次のコマンドを使用して、JTAG インターフェイスを介して U-Boot をダウンロードすることで、システムは最初にブートストラップされます。

```
$ cd $PETALINUX/software/ PetaLinux-dist
$  PetaLinux-jtag-boot -i images/u-boot.elf
```

図 6 に U-Boot コンソールのスナップショットを示します。

FPGA ボードはイーサネット インターフェイスに接続されていることに注意してください。XPS のハードウェア リソース部でイーサネット インターフェイスを選択する必要があります。U-Boot がブートしたら、サーバーとホストの IP アドレスが同一かどうかを確認します。アドレスが同一でない場合は、U-Boot ターミナルで次のコマンドを使用して、ホストの IP を設定します。

```
u-boot>print serverip // prints 192.168.25.45(server ip)
u-boot>print ipaddr   // prints IP address
of the board as  // 192.168.25.68
u-boot>set serverip <HOST IP> // Host IP 192.168.25.68
u-boot>set serverip 192.168.25.68
```

これでサーバー (PC) とホスト (KC705 ボード ) は同一の IP アドレスを持ちます。サーバーから netboot コマンドを実行し、

PetaLinux イメージをダウンロードしてブートします。

```
u-boot> run netboot
```

netboot の実行後、PetaLinux コンソールが表示されるはずです ( 図 5 を参照 )。

大事なことを言い忘れていましたが、GUI ([PetaLinux SDK] → [BOOT QEMU [Linux]]) または次のコマンドを使用して、QEMU によってカーネル ブートを実行できます。

```
$ cd $ PETALINUX/software/ PetaLinux-dist
$ PetaLinux-qemu-boot -i images/image.elf
```

この高速な手法を使用すると、図 7 に示す画面が表示されます。

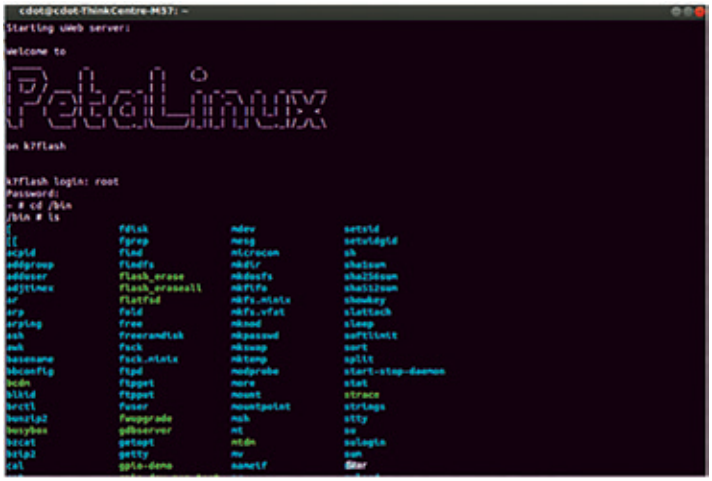


図 5 – OS がブートされたことを確認する PetaLinux コンソールのスナップショット

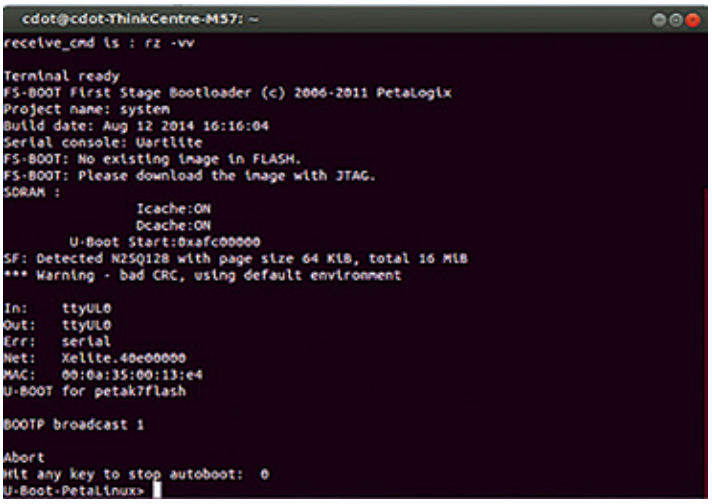



図 6 – U-Boot (Universal Bootloader) による間接的カーネル ブート

## デザイン上でのアプリケーションの動作テスト

PetaLinux のブート テストが完了したら、次の作業は、PetaLinux 向けに設計されたユーザー アプリケーションをテストすることです。MicroBlaze プロセッサは、Kintex-7 FPGA ボード上のハードウェア ペリフェラルを一連のメモリ レジスタとして認識します。各レジスタはそれぞれのベース アドレスと終了アドレスを持ちます。ペリフェラルにアクセスするには、ユーザーはペリフェラルのベース アドレスと終了アドレスを知っている必要があります。これらのアドレスの詳細は、デバイス ツリー ソース (*.dts) ファイルで確認できます。このデザインでは、DDR3 へのアクセス、/dev/mem を使用した GPIO へのアクセス、UIO を使用した GPIO へのアクセス、ファイル転送の 4 つのアプリケーションを開発し、テストしました。

### *1. DDR3 へのアクセス*

DDR3-test.c という名前の PetaLinux アプリケーションを使用して、DDR3 メモリにアクセスしました。このアプリケーションは、DDR3 のメモリ位置に対するデータの書き込みと読み出しを行うように設計されています。DDR3 は、ユーザー コードおよびデータの格納用の SDRAM を備えたデュアル インライン メモリ モジュールです。前述のように、ユーザーは DDR3 メモリの開始アドレス (0xC0000000) と終了アドレス (0xC7FFFFFF) を知っている必要があります。メモリのサイズは 512 メガバイトです。Linux カーネルは DDR3 メモリの初期メモリ位置にあります。したがって、DDR3 メモリに対する書き込み位置は、Linux カーネルが壊れないように選択されます。DDR3 メモリへのデータの書き込みには、次のコマンドを使用しました。

```
#DDR3-test -g 0xc7000000 -o 15
```

このコマンドでは、DDR3-test はアプリケーション名、-g は

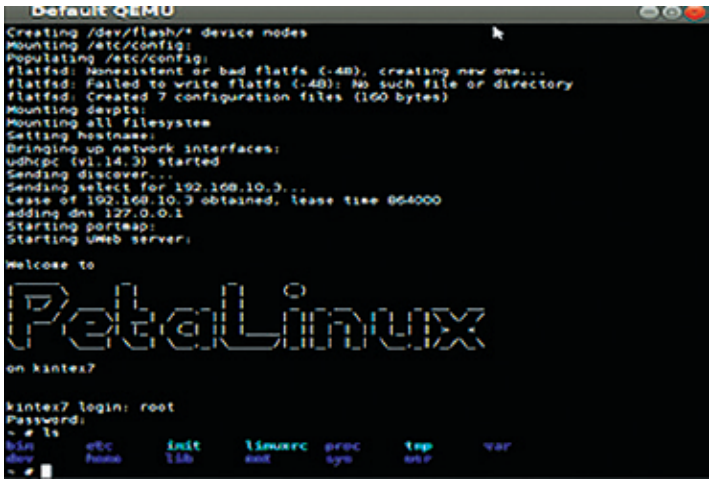



図 7 – QEMU を使用した PetaLinux の実行

DDR3 メモリの物理アドレス、-o は出力、15 は DDR3 メモリの位置 0xc7000000 に書き込まれると予想される値です。予想される位置に値が書き込まれるかどうかをテストするために、次のコマンドを使用して、DDR3 メモリからデータを読み出しました。

```
#DDR3-test -g 0xc7000000 -i
```

値 15 がターミナル内で観察されました。これにより、DDR3 メモリの読み出しおよび書き込み動作が完璧に機能していたことを確認できます。

### 2. /dev/mem を使用した GPIO へのアクセス

次のアプリケーション テストでは、gpio-dev-mem-test.c という名前の PetaLinux アプリケーションを使用して汎用 I/O (GPIO) にアクセスしました。このアプリケーションは、8 ビット ディスクリート出力を制御し、ボード上で LED を GPIO に接続することによってその出力をテストするように設計されています。ユーザー空間からデバイスにアクセスするには、/dev/mem を開き、mmap() を使用してデバイスをメモリにマッピングします。筆者らが使用した LED GPIO の開始アドレスは 0x40000000、終了アドレスは 0x4fffffff です。

GPIO ペリフェラルは、データ レジスタ (GPIO_DATA) と方向レジスタ (GPIO_TRI_ OFFSET) の 2 つのレジスタを持ちます。GPIO のステータスを読み出すために、方向ビットを 1 に設定して ( すなわち、GPIO_TRI_OFFSET=1)、データ レジスタからデータを読み出しました。GPIO にデータを書き込むには、このビットを 0 に設定して、データ レジスタに値を書き込みます。PetaLinux ターミナル上で次のコマンドを使用すると、GPIO にデータが書き込まれます。

```
#gpio-dev-mem-test -g 0x40000000 -o 255
```

このコマンドでは、gpio-dev-mem-test はアプリケーション名、-g は GPIO の物理アドレス、-o は出力、255 は (LED に接続された ) GPIO から送信される値です。テストの結果は、LED がプログラムどおりに点灯したことで検証されました。

### 3. UIO を使用した GPIO へのアクセス

GPIO にアクセスする代替手段として、ユーザー空間入力 / 出力 (UIO) を利用する方法があります。筆者らは、gpio-uio-test.c という名前の PetaLinux アプリケーションを使用して、UIO を介して GPIO にアクセスしました。このアプリケーションは、8 ビット ディスクリート出力を制御するように設計され、ボード上で LED を GPIO に接続することによってテストされます。UIO デバイスは、ファイル システム内で /dev/uioX として表されます。UIO を介して GPIO にアクセスするために、/dev/uioX または sys/class/uio/uiO を開き、mmap() コールを使用しました。筆者らは、UIO をサポートするようにカーネルをコンフィギュレーションし、カーネル内で UIO フレームワークを有効にしました。次に、「Compatibility」というパラメーターを使用して、LED の GPIO が通常の GPIO デバイスではなく UIO デバイスとして制御されるように設定しました。また、デバイスのラベルを gpio@40000000 から leds@40000000 に変更しました。

次に、PetaLinux を再構築し、UIO を使用した GPIO アクセスをテストしました。次のコマンドを使用して、ロードされる UIO モジュールの詳細な情報を取得しました。

```
# ls /sys/class/uio/
 uio0 uio1 uio2
```

UIO の名前とアドレスは、/sys/ class/uio/uioX で確認できます。次のコマンドを使用して、UIO ドライバーを介して GPIO LED にアクセスしました。

```
# cd "/sys/class/uio/uioX
# gpio-uio-test -d /dev/uio1 -o 255
```

このコマンドでは、gpio-uio-test はアプリケーション名、-d はデバイスのパス、-o は出力、255 は UIO を介して GPIO に渡される値です。結果は、上記のコマンドを使用して GPIO ライン上に書き込まれたデータに基づいて点灯する LED によって検証されました。

### 4. ファイル転送アプリケーション

最後のテストでは、サーバーからクライアントにファイルを転送しました ( サーバーはホスト PC、クライアントは KC705 ボード )。このテストでは、イーサネット ケーブルを使用してサーバーとクライアントを接続しました。転送には TFTP (Trivial File Transfer Protocol) を使用しました。このプロトコルの簡潔さはよく知られており、一般的にコンフィギュレーション ファイルまたはブート ファイルの自動転送に使用されます。TFTP を使用したサーバーからクライアントへのファイル転送をテストするために、サーバー PC の /tftpboot 内に test という名前のファイルを作成しました。次のコマンドを使用して、ファイル内に“Hello World”と書き込み、同じファイルの内容を表示しました ( 図 8 を参照 )。

```
@ echo "Hello World" > /tftpboot/test
@ more /tftpboot/test
```

このファイルをサーバーから受信するために、KC705 ボード上でクライアントとして動作している PetaLinux ターミナル ウィンドウに、次の get コマンド (-g) を入力しました。

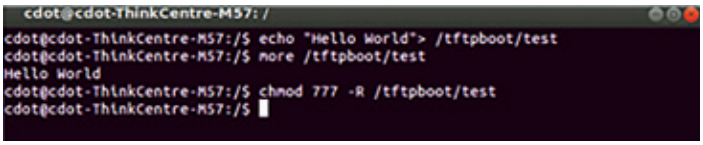



図 8 – サーバー内でのファイル作成のスナップショット

```
# tftp -r test -g 192.168.25.68
# ls -a
```

ファイル名“test”の新しいファイルがクライアント内に作成されました ( 図 9 を参照 )。このファイルの内容は、図 9 に示すように、more コマンドを使用して表示できます。

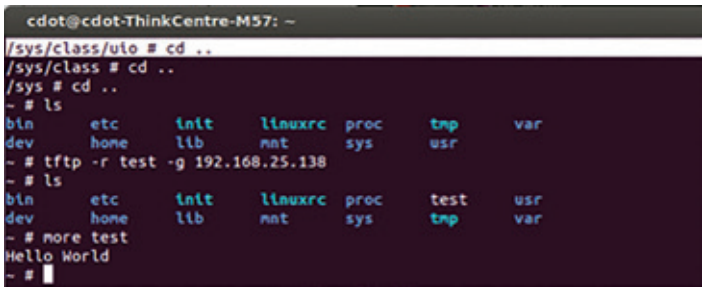



図 9 – クライアント内でのファイル受信のスナップショット

同様に、クライアント マシン内に“PetaLinux OS”という内容で test1 という名前のファイルを作成することで、クライアントからサーバーへのファイル転送を実行します。このファイルをクライアントからサーバーへ送信するには、クライアント上で動作している PetaLinux ターミナルで次の“put”コマンド (-p) を使用します ( 図 10 を参照 )。

```
# tftp -r test1 -p 192.168.25.68
```


図 10 – クライアントからサーバーへのファイル送信のスナップショット

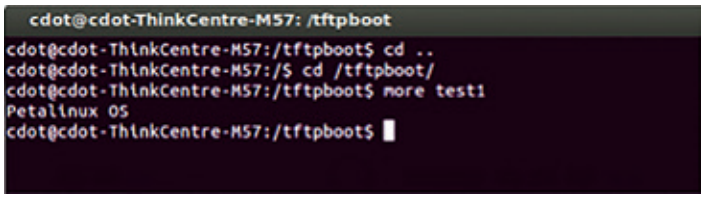



図 11 – サーバー内でのファイル受信のスナップショット

空の test1 ファイルがサーバー内に作成されます。ファイルの内容はファイル転送操作後に読み出され、図 11 に示すように検証されます。

FPGA 上にエンベデッド システムをインプリメントし、PetaLinux を動作させる作業は非常に簡単です。次に筆者らは、リモート プログラミング機能を使用してデザインをインプリメントする予定です。このデザインでは、イーサネットを介してブート ファイルが転送され、クライアントは新しいアプリケーションを実行できます。

## 参考資料


1. Kynan Fraser, “MicroBlaze Running uClinux,” Advanced Computer Architecture from *http://www.cse.unsw.edu.au/~cs4211*
2. Xilkernel Version 3.0 (Xilinx Inc., 2006 年 12 月)
3. 『PetaLinux SDK ユーザー ガイド』(UG976) (Xilinx Inc., 2013 年 4 月)
4. Gokhan Ugurel and Cuneyt F. Bazlamacci, “Context Switching Time and Memory Footprint Comparison of Xilkernel and µC/OS-II on MicroBlaze,” 7th International Conference on Electrical and Electronics Engineering, December 2011, Bursa, Turkey, pp.52-55
5. Chenxin Zhang, Kleves Lamaj, Monthadar Al Jaberi and Praveen Mayakar, “Damn Small Network Attached Storage (NAS),” Project Report, Advanced Embedded Systems Course, Lunds Tekniska Hogskola, November 2008
6. 『Platform Studio ユーザー ガイド』(UG113) (Version 1.0) (Xilinx Inc., 2004 年 3 月)
7. 『EDK コンセプト、ツール、テクニック : 効率的なエンベデッド システム構築をサポートするハンディガイド』(UG683) (Version 14.1) (Xilinx Inc., 2012 年 4 月)
8. 『PetaLinux SDK ユーザー ガイド : アプリケーション開発ガイド』(UG981) (Xilinx Inc., 2013 年 4 月)
9. 『PetaLinux SDK ユーザー ガイド : QEMU システム シミュレーション ガイド』(UG982) (Xilinx Inc., 2013 年 11 月)

# 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。

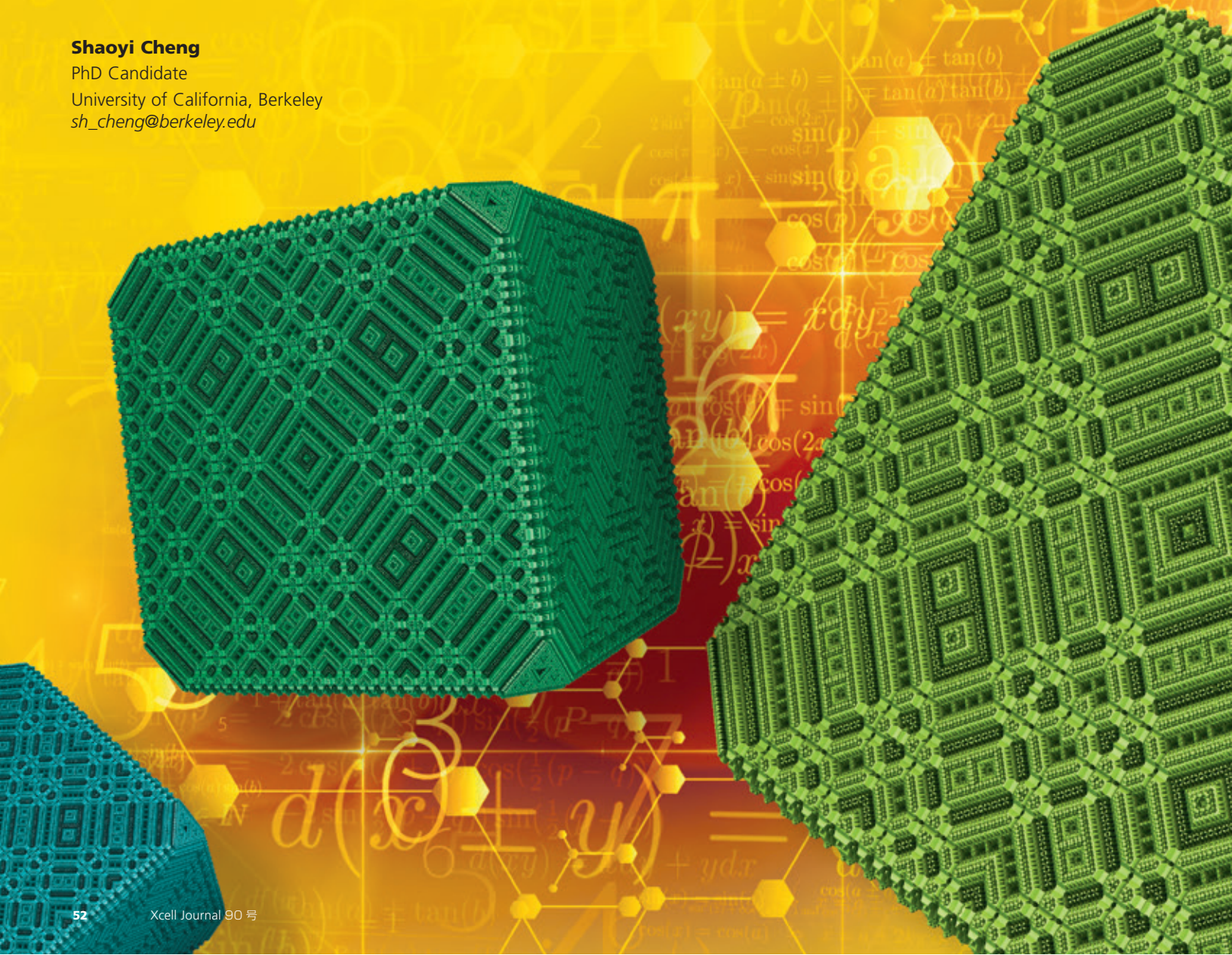
- ▶ UltraScale アーキテクチャ概要
- ▶ Zynq-7000 SoC アーキテクチャとエコシステム
- ▶ 28nm ザイリンクス 7 シリーズ FPGA のアジャイル ミックスド シグナル テクノロジ

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/

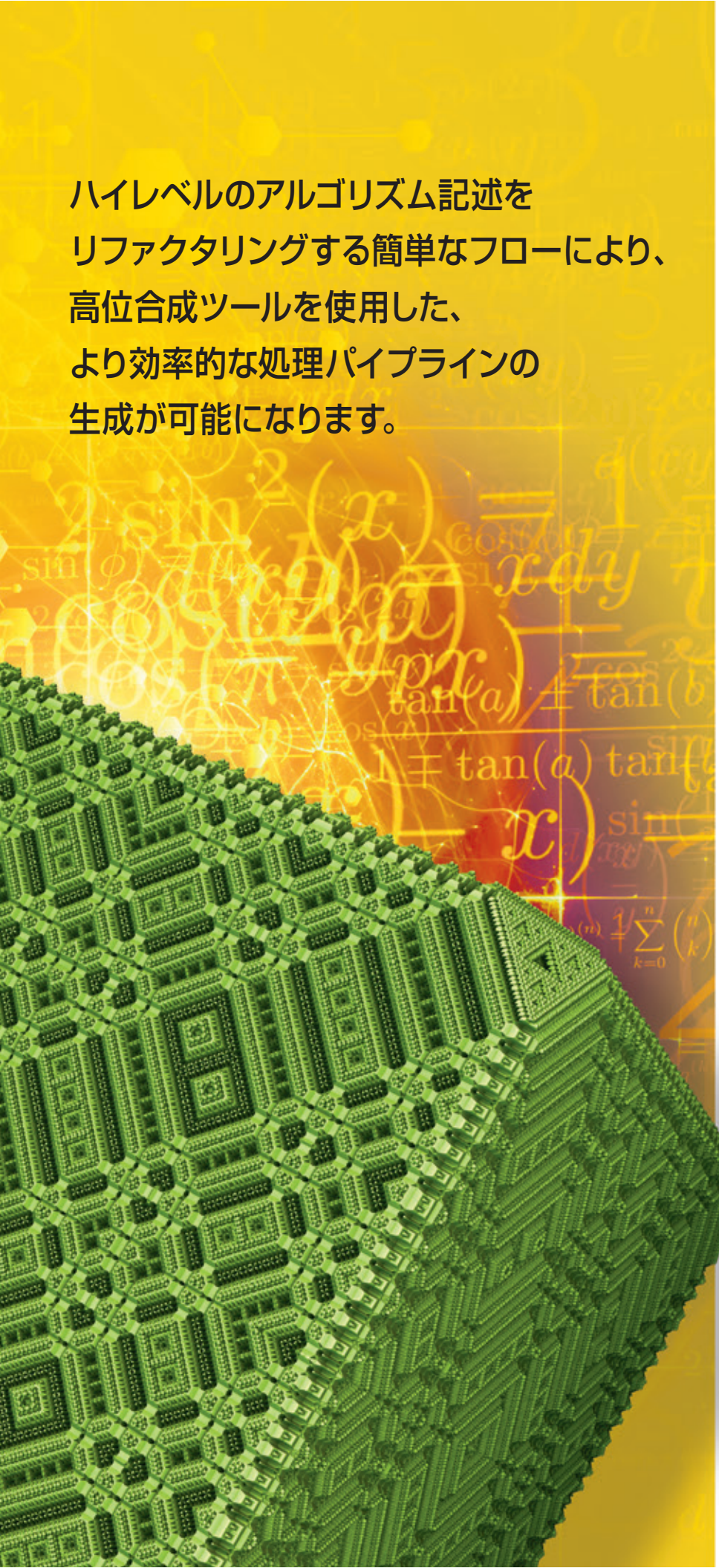
# Try Algorithm Refactoring to Generate an Efficient Processing Pipeline with Vivado HLS

# アルゴリズムのリファクタリングを使って Vivado HLS による効率的な処理パイプラインの生成を実現

**Shaoyi Cheng**
PhD Candidate
University of California, Berkeley
*sh_cheng@berkeley.edu*



**ハイレベルのアルゴリズム記述を
リファクタリングする簡単なフローにより、
高位合成ツールを使用した、
より効率的な処理パイプラインの
生成が可能になります。**



規則的なメモリ アクセス パターンとループ反復間の簡単に抽出可能な並列処理を含む演算カーネルを開発する場合、高性能アクセラレータ作成用の優れたリソースとして Vivado® Design Suite 高位合成 (HLS) ツールを利用できます。C 言語によるハイレベルのアルゴリズム記述に少数のプラグマを追加することにより、ザイリンクス FPGA 上に高スループットの処理エンジンを迅速にインプリメントできます。作成された処理エンジンと、ソフトウェアによって管理される DMA メカニズムを組み合わせることで、汎用プロセッサに比べて桁違いの高速化を実現できます。

しかし、実際のアプリケーションには、多くの場合 ( 特に科学技術計算および信号処理分野以外のアルゴリズムになると ) 簡単には処理できない複雑なメモリ アクセスが含まれます。筆者らは、このような状況で効率的な処理パイプラインを生成できる簡単な手法を考案しました。この手法について詳しく説明する前に、Vivado HLS の動作機能と、( さらに重要なのですが ) このツールが十分に機能しない場合について説明します。

## Vivado HLS ツールの機能

Vivado HLS ( 高位合成 ) ツールは、高級言語で記述された制御データ フロー グラフ (CDFG) 内の並列性を取り込もうとします。演算操作とメモリ アクセスは、それらの間の依存関係の制約とターゲット プラットフォームのリソースの制約に従って割り当てられ、スケジューリングされます。回路内での特定の操作のアクティベーションは特定のクロック サイクルに関連付けられ、データ パスと一緒に合成される中央が CDFG 全体の実行を調整します。

スケジューリングは静的に実行されるため、アクセラレータの実行時の動作は比較的簡単です。生成された回路のさまざまな部分は、相互にロックステップ方式で動作します。高性能 CPU

に見られるような動的な依存関係チェック メカニズムは不要です。たとえば、図 1(a) に示す関数では、ループ インデックスの加算と curInd のロードは並列化できます。また、現在の反復が終了する前に次の反復を開始できます。

一方、浮動小数点乗算は常に前の反復からの乗算結果を使用するため、新しい反復を開始できる最小間隔は、浮動小数点乗算器のレイテンシによって制限されます。この関数の実行スケジュールを図 2(a) に示します。

## この手法で最適な結果が得られない場合とは

この手法の問題点は、データ フロー グラフ全体が厳格なスケジュールで実行されることです。オフチップ通信によって生じるストール（行き詰まり）が処理エンジン全体に伝搬すると、性能が大幅に低下します。メモリ アクセス パターンがあらかじめわかっており、データが必要になる前にそのデータをオンチップで移動できる場合、あるいはデータ セットが十分に小さく、FPGA 上でデータ全体をバッファリングできる場合は、この問題は起こりません。しかし、多くの興味深いアルゴリズムでは、データ アクセスは演算の結果に依存し、メモリ フットプリントが大きいためにオフチップ RAM を使用する必要があります。ここで、「素朴に (naively)」HLS をカーネルに適用すると、命令レベルの並列性の大きいデータ パスが作成されますが、これがアクティベートされると、データの供給を待つ間に頻繁に停止することになります。

図 2(b) に、この例の関数に対して生成されたハードウェア モジュールの実行を示します（データ セットが大きすぎて、オンチップ キャッシュに動的にフェッチする必要がある場合）。すべてのキャッシュ ミス レイテンシの組み合わせが、どのように速度の低下に反映しているかに注意してください。ただし、演算グラフには、メモリ データがただちに利用可能でなくても進行する部分があるため、必ずしもこのようなストールが発生するとは限りません。これらのセクションの処理が進行できるようにする必要があります。これによって実行スケジュールの自由度が多少高まることで、これから説明するように、大きな効果をもたらす可能性があります。

## リファクタリング / 分離のメリット

既に説明した関数の例を考えます。統一的なスケジュールによって浮動小数点乗算の実行とデータ アクセスがすべて関連付けられてはいない場合を考えます。1 つのロード演算子でデータが戻るのを待機している間に、もう 1 つのロード演算子は新しいメモリ要求を開始でき、乗算器の実行も進行します。これを実現するには、独自のスケジュールで動作し、各メモリ アクセスを担当するモジュールが 1 つ必要です。また、乗算器ユニットは、すべてのメモリ操作に対して非同期で実行される必要があります。

異なるモジュール間のデータの依存関係は、ハードウェア FIFO を介して伝達されます。この例で可能なリファクタリングを図 1(b) に示します。ステージ間の通信に使用されるハードウェア キューは、既にフェッチ

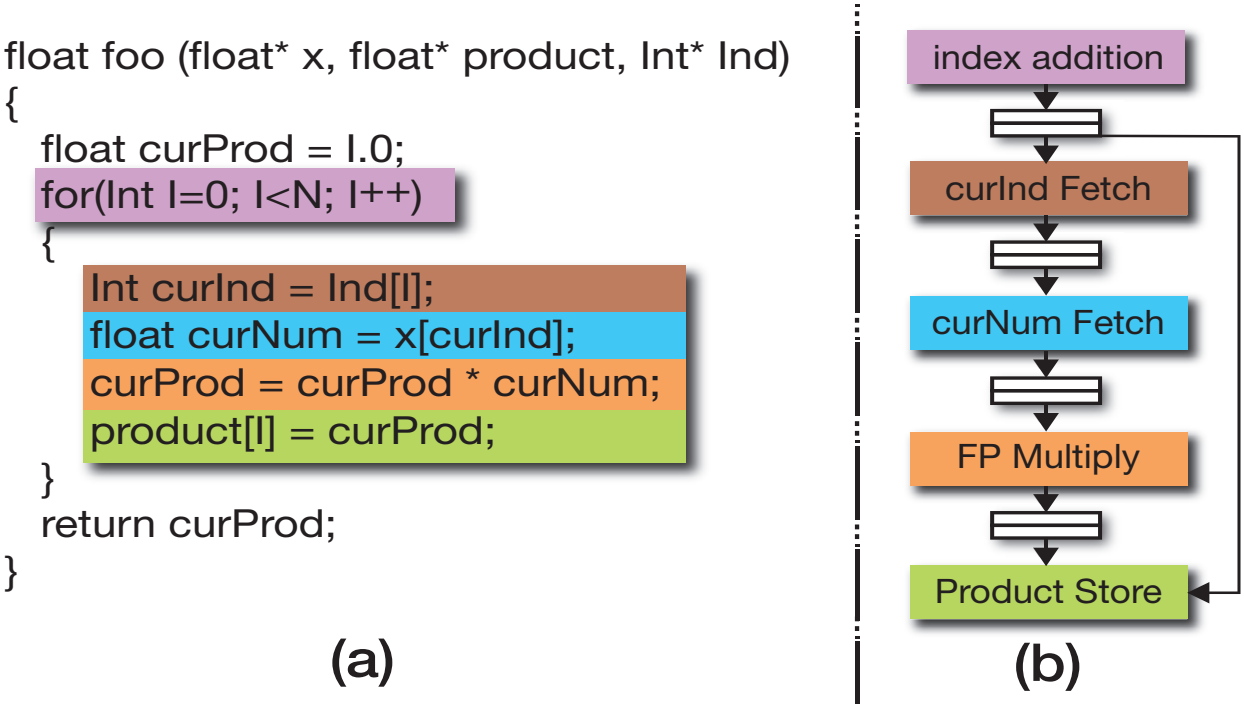



図 1 – サンプル デザイン : (a) 不規則なメモリ アクセス パターンを含む関数、(b) 可能なリファクタリングから得られるパイプライン構造

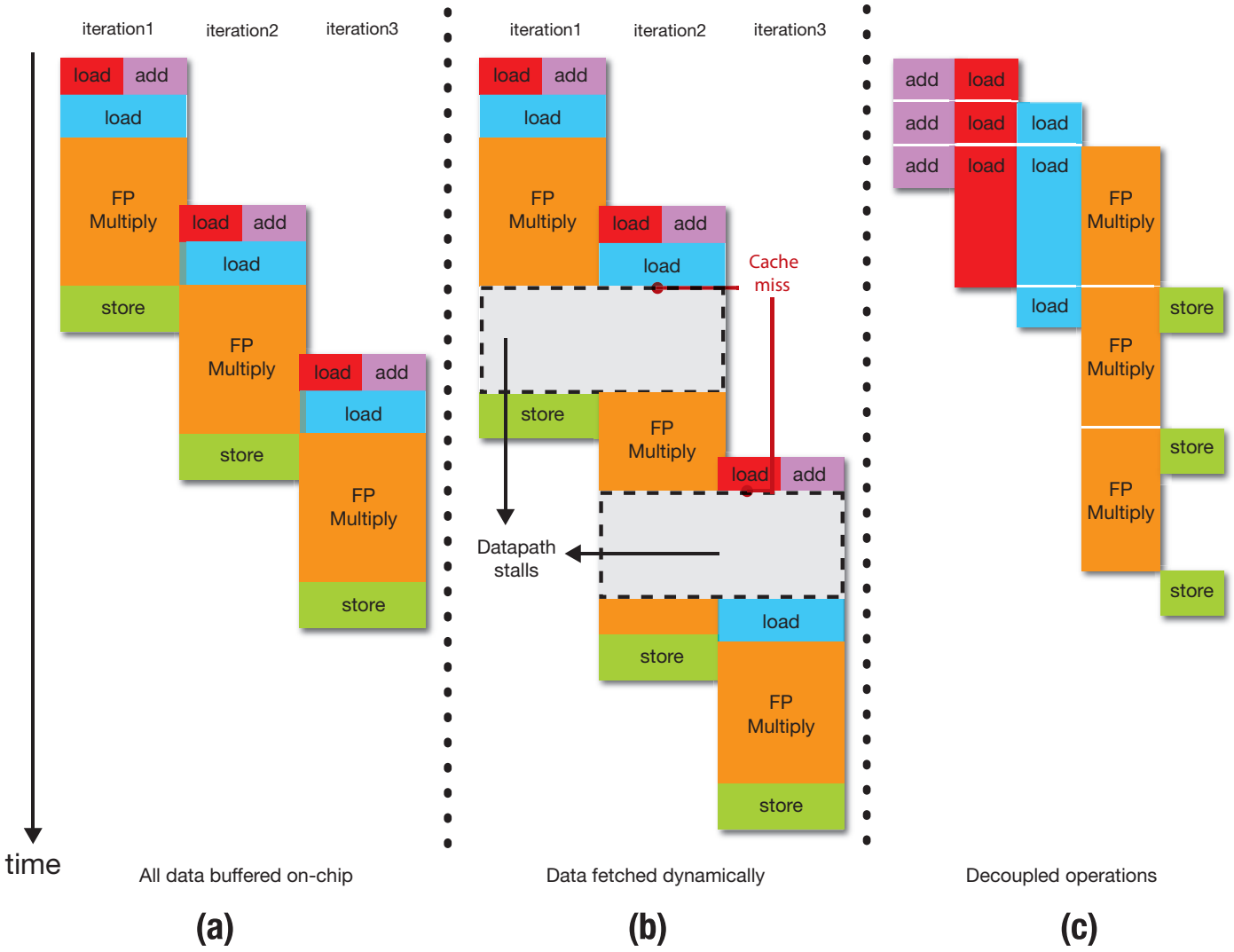



図 2 – さまざまなシナリオの実行スケジュール : (a) すべてのデータがオンチップでバッファリングされる、(b) データが動的にフェッチされる、(c) 分離された操作

されたがまだ使用されていないデータをバッファリングできます。メモリ アクセス部分がキャッシュ ミスのためにストールしたときは、既に生成されたデータのバックログから乗算器ユニットへの供給が続けられます。長い時間で見ると、発生したストールは浮動小数点乗算の長いレイテンシによって隠すことができます。

図 2(c) に、この分離された処理パイプラインを使用した場合の実行スケジュールを示します。ここでは FIFO を介したレイテンシを考慮に入れていませんが、反復の回数が多い場合、その影響は最小限に抑えられます。

## リファクタリングの手法

分離された処理モジュールのパイプラインを生成するには、まず元の CDFG 内の命令をクラスター化してサブグラフを形成する必要があります。得られるインプリメンテーションが最大限の性能を発揮するには、クラスタリングの手法が次の 3 つの要件を満たしていなければなりません。

第 1 に、既に説明したように、Vivado HLS ツールはソフトウェア パイプライン処理を使用して、前の反復が完了する前に新しい反復を開始します。CDFG 内の最も長い循環依存関係のレイテンシによって、新しい反復を開始できる最小間隔が決まります。この間隔は、最終的には、アクセラレータが達成できる全体的なスループットを制約します。したがって、このような依存性サイクルが複数のサブグラフにわたらないことが重要です。それは、モジュール間の通信に使用される FIFO が、常にレイテンシを大きくするからです。

第 2 に、データ消費の遅速性によってキャッシュ ミスが「隠される」ように、長レイテンシ演算を含む依存性サイクルからメモリ操作を切り離すことは有益です。「長レイテンシ」とは、演算に 2 サイクル以上かかる

という意味です。これは筆者らの目的のために、Vivado HLS スケジュールを使用して得られた指標です。したがって、たとえば乗算は長レイテンシ演算ですが、整数の加算は長レイテンシ演算ではありません。

最後に、キャッシュ ミスによって生じるストールの影響を局所に限定するために、各サブグラフ内のメモリ操作の数を最小限に抑えることを推奨します（特に、それらの操作がメモリ空間の異なる部分をアドレス指定する場合）。

第 1 の要件（依存性サイクルが複数のサブグラフにわたらないようにすること）を満たすのは簡単です。そのためには、元のデータ フロー グラフ内の強連結成分 (SCC) を見つけて、それらの成分を異なるクラスターに分離する前にノードにコラプス（分解）します。このプロセスの結果、一部のノードは簡単な命令であり、他のノードは相互に依存する一連の操作である、有向非巡回グラフが得られます。

第 2 および第 3 の要件（メモリ操作を分離し、ストールの影響を局所に限定すること）を満たすには、メモリ操作ノードのトポロジカル ソートを実行してからノードを分割します。最も簡単に分割する方法は、すべてのメモリ操作または長レイテンシの SCC ノードの後に「境界線」を引くことです。図 3 に、どのようにこの手法を筆者らの例に適用するかを示します。このクラスタリングと図 1 のパイプライン構造との対応関係は明らかです。各サブグラフは、互いに独立して HLS を適用できる新しい C 関数です。これらのサブグラフは、互いに非同期で実行されます。

筆者らは、ソースからソースへの簡単な変換ツールを作成して、このリファクタリングを実行しました。生成された個々のモジュールを接続する FIFO には、ザイリンクス IP コアを使用しました。特定の演算カーネルのリファクタリングの方法は、確かに 1 つではありません。デザイン空間の探索は依然として進行中です。

## メモリ アクセスのパイプライン化

分離された処理パイプラインの初期インプリメンテーションが完了したら、最適化を実行して効率を向上させることができます。既に説明したように、HLS を使用して C 関数をマッピングすると、メモリ読み出しが妨げになります。このことは筆者らのパイプラインの個々の段についても当てはまります。たとえば、x[curInd] のロードを担当するモジュールは、次の curInd が既に利用可能であり、FIFO の下流に十分な空間がある場合でも、データを待機する間ストールすることがあります。

この問題を解決するには、変換を実行してメモリ アクセスをパイプライン化します。特定の段について C 関数内で簡単なメモリ ロードを実行する代わりに、そのロードを新しい FIFO へのアドレスのプッシュで置き換えます。これで、新しいハードウェア モジュールが別個にインスタンシエートされ、アドレス FIFO からアドレスを読み出して、それらのアドレスをメモリ サブシステムに送信します。戻ってくるデータは下流の FIFO に直接プッシュされます。これでメモリ アクセスは実質的にパイプライン化されます。

アドレスのプッシュ操作は、Vivado HLS 内で FIFO インターフェイスへのメモリ ストアによって表現できますが、下流の FIFO を監視してメモリ要求を発行するハードウェア モジュールは、Verilog でインプリメントされます。これは、送信アドレスと応答データは、Vivado HLS によって合成可能なメモリ インターフェイス内に一緒にバンドルされないからです。しかし、この簡単なモジュールはさまざまなベンチマークで何回も再利用されるため、デザインに手間をかけるだけの価値は十分にあります。

## 複製と通信のコスト比較

ステージ間のデータ移動のために導入される FIFO は、カーネルのリファクタリングと分離された処理パイプラインの生成における大きなオーバーヘッドになります。最小の深さの FIFO でもかなりの量の FPGA リソースのコストがかかりそうなので、若干の演算命令を複製することによって FIFO の一部を削除することは、多くの場合有益です。

一般的に、このトレードオフにおける最善のデザイン ポイントの探索には、コスト

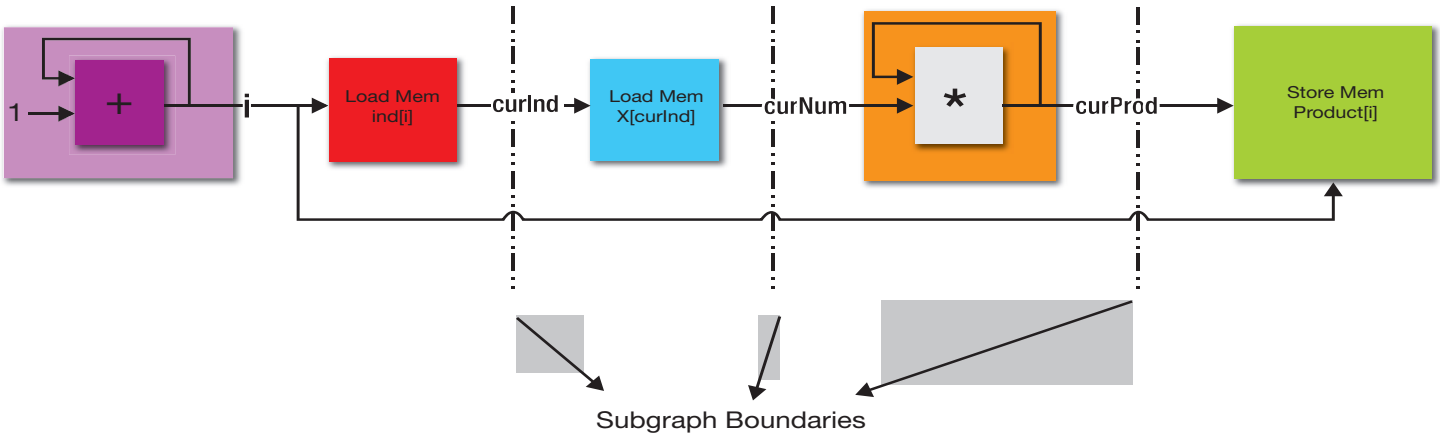



図 3 – サブグラフへのリファクタリング

```
for (w = 1; w <= W; w++) {
  int option1 = opt[n-1][ w];
  int option2 = -999999;
  int opt_without = opt[n-1][ w-cur_weight];
  if (cur_weight <= w)
     option2 = cur_profit + opt_without;
     opt[n][w] = option1> option2? option1:option2;
     sol [n][ w] = option2> oprion1? 1:0;
}
```

図 4 – ナップザック問題

モデリングとフォーマル最適化手法を適用できます。しかし、ほとんどのベンチマークでは、単にすべてのユーザーに対して簡単なループ カウンターを複製することで大きな面積を削減できるため、筆者らはこの手法を採用しました。この例では、この最適化手法に i の整数加算器の複製が含まれるため、Product[i] へのストアが他のモジュールからインデックスを取得する必要はありません。

## バースト メモリ アクセス

第 3 の最適化手法は、バースト メモリ アクセスです。メモリ帯域幅を効率的に利用するには、1 回のメモリ トランザクションで複数ワードのデータを伝達することが望まれます。AXI バス プロトコルは、バースト長の指定が可能です。分離された C 関数とパイプライン化されたメモリ アクセス モジュールを多少変更することで、この機能を活用できます。

連続したメモリ チャンクにアクセスするとき、分離された C 関数内の各メモリ演算子は、アドレスの生成以外に、バースト長の計算も行います。アクセスするワード数はそれぞれの分離された関数内でローカルに決定できるため、ループ カウンターを複製すれば、バースト アクセスの生成にも役立ちます。

## 実験的評価

筆者らは、この記事で説明した手法を応用して 2、3 のケース スタディを実施しました。この手法のメリットを評価するために、筆者らの手法を使って生成された分離型処理パイプライン (DPP) と、素朴に HLS を適用することによって生成されたアクセラレータ (Naïve) を比較しました。Naïve と DPP のインプリメンテーションに対して Vivado HLS を起動する際には、ターゲット クロック周波数を 150MHz に設定し、配置配線後に達成可能な最高のクロック レートを使用しました。また、アクセラレータとメモリ サブシステム間の相互作用のためにさまざまなメカニズムを試しました。ACP ポートと HP ポートが使用され、それぞれについてリコンフィギャラブル アレイ上に 64 キロバイト キャッシュもインスタンシエートしました。

この実験に使用した物理デバイスは、Zed Board 評価プラットフォーム上にインストールされたザイリンクスの Zynq®-7000 XC7Z020 All Programmable SoC です。また筆者らは、Zynq SoC 上の ARM® プロセッサでこのアプリケーションのソフトウェア版を実行し、そのパフォーマンスを実験のベースラインとして使用しました。生成されるすべてのアクセラレータは自己完結型であり、リコンフィギャラブル ファブリックとの間のデータ移動に DMA メカニズムを必要としません。

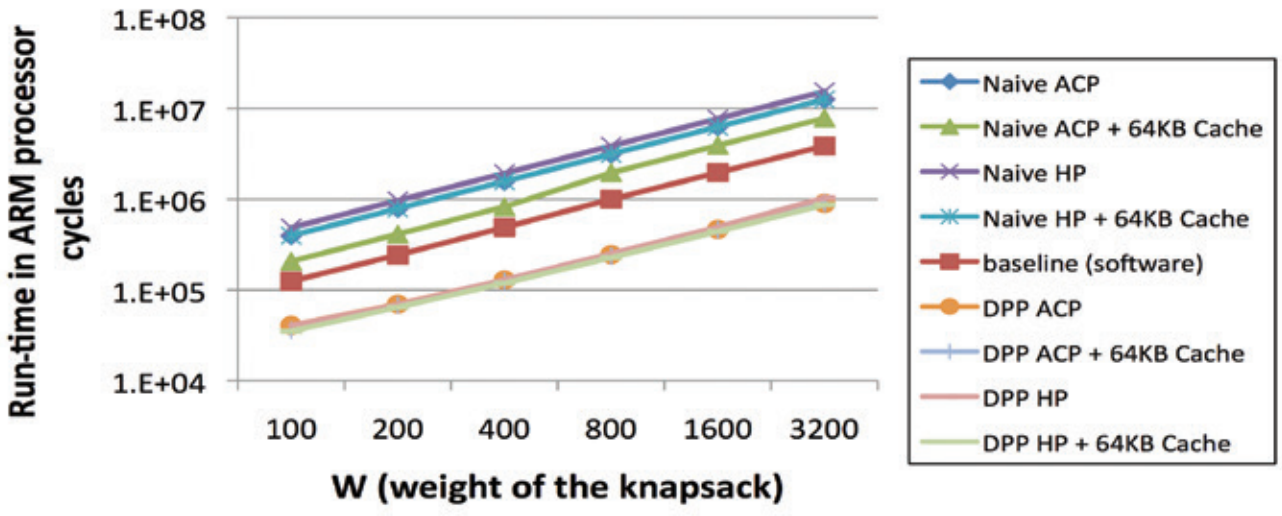



図 5 – ナップザック問題の実行時間の比較

## ケース スタディ 1 : ナップザック問題

ナップザック問題は、動的プログラミングを使って解くことができる有名な組み合わせ問題です。カーネルの概要を図 4 に示します。特に、太字で示した変数はすべて実行時にメモリから読み出されます。したがって、変数 opt_without がどこからロードされるかの正確な位置はあらかじめわかっていません。w と n が大きい場合、opt 配列全体をオンチップでバッファリングすることはできません。配列の必要な部分を演算エンジンにフェッチさせることのみ可能です。

図 5 に、筆者らの手法 (DPP) を使用して生成されるアクセラレータと、素朴に HLS を関数に適用することによって生成されるアクセラレータ (Naïve) の実行時間の比較を示します。このグラフは、ARM プロセッサ上での関数の実行のパフォーマンスも示しています。n ( 品目数 ) を 40 に固定して、w ( ナップザックの総重量 ) を 100 から 3,200 まで変化させます。

Vivado HLS を使用して素朴にソフトウェア カーネルをマッピングすると、ベースラインよりもはるかに性能が低いアクセラレータが生成されることは、この比較から明らかです。Zynq SoC 上のスーパースカラー型アウトオブオーダー ARM コアは、命令レベルの並列性をかなりの程度まで利用でき、高性能なオンチップ キャッシュも備えています。Vivado HLS ツールによって抽出される並列性の増加は、プログラマブル ロジックと比べたハード プロセッサ コアのクロック周波数の高さと、リコンフィギャラブル アレイからの大きなデータ アクセス レイテンシを埋め合わせるには明らかに不十分です。

しかし、カーネルが複数の処理ステージに分離された場合、その性能は ARM プロセッサの性能よりもはるかに高くなります ( 約 4.5 倍 )。また、DPP を使用した場合の各種のメモリ アクセス メカニズム間の違いはかなり小さくなります。つまり、筆者らの手法を使うと、メモリ アクセス レイテンシに対する感度は大きく改善されます。

```
for(s =0; s<dim; s++)
{
   int kend = ptr[s];
   int k;
   float curY = y[s];
   for(k = kbegin; k<kend; k++){
      int curInd = indArray[k];
      curY = curY +valArray[k] * xvec[curInd];
    }
    Y[s] = curY;
    kbegin = kend;
}
```

図 6 – 疎行列ベクトル積

## ケース スタディ 2 : 疎行列ベクトル積

疎行列ベクトル (SpMV) 積は、さまざまな

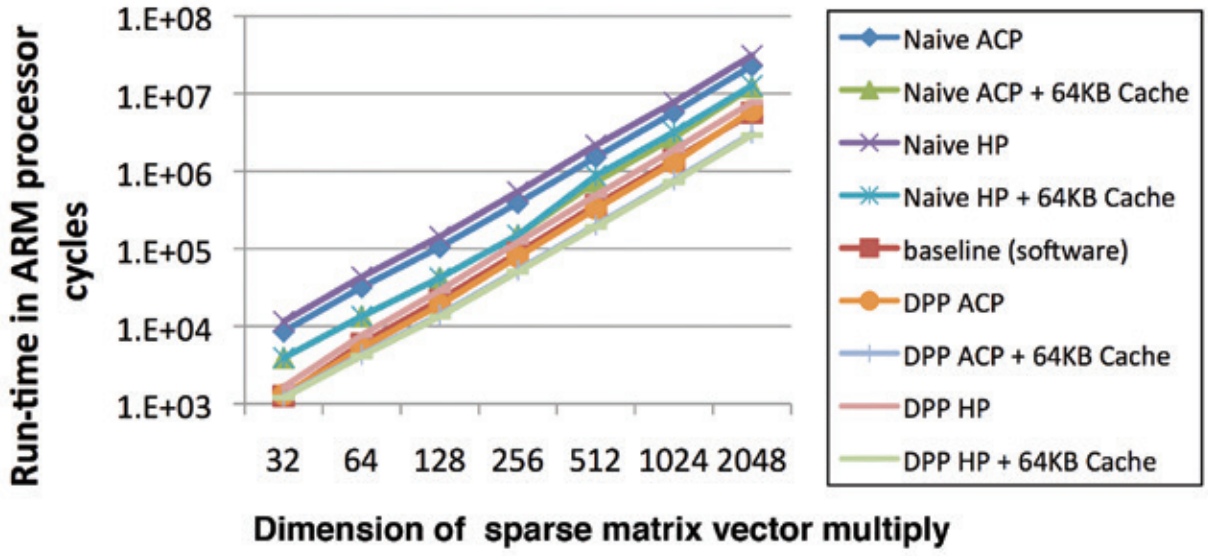



図 7 – 疎行列ベクトル積の実行時間の比較

研究プロジェクトで多くの異なる方法で研究、変形され、ベンチマークの対象となってきた演算カーネルです。筆者らの目的は、特殊なデータ構造とメモリ割り当て方式を使用して最高の性能を持つ SpMV 積を作成することではなく、最も基本的なアルゴリズム記述で、Vivado HLS を使用した場合にリファクタリング パスがどの程度のメリットをもたらすかを確認することです。

図 6 に示すように、この実験では、疎行列は CSR (Compressed Sparse Row) 形式で格納されます。実際の浮動小数点乗算のための数値がフェッチされる前に、インデックス配列からのロードが実行されます。この事例でも、制御フローとアクセス先のメモリ位置は、実行時に初めてわかる値に依存します。

図 7 に示した実行時間の比較では、行列は 1/16 の平均密度で値を与えられ、次元は 32 ～ 2,048 の範囲で変化します。

この事例でも、素朴なマッピングを適用した場合の性能は、ソフトウェア版よりも低速になります。筆者らの手法で生成される分離された処理パイプラインは、FPGA 上のキャッシュが使用されない場合、ベースラインとほぼ同じ性能になります。

リコンフィギャラブル アレイ上にインスタンシエートされた 64 キロバイト キャッシュにより、DPP の性能はベースラインの性能の 2 倍近くに達します。前のベンチマークと比べて、キャッシュの追加は DPP の性能に顕著な影響を与えます。

## ケース スタディ 3 : フロイド - ワーシャル アルゴリズム

フロイド - ワーシャル法は、任意の 2 頂点間の最短経路の発見に使用されるグラフ アルゴリズムです。メモリ アクセス パターンは、前のベンチマークよりも簡単です。したがって、DMA とアクセラレータのセットアップを工夫して、演算とオフチップ通信の最適なオーバーラップを実現する方法があると考えられます。筆者らの手法は、このようなオーバーラップを自動的に実現しようとするものですが、絶対に最適な結果とここで得られる結果とのギャップを示す研究はまだ行われていません。

ただし筆者らは、前の 2 つのベンチマークと同様に、実行時間の比較を行いました。ここでは、グラフのサイズを 40 ノードから 160 ノードまで変化させました。各ノードは、隣接するノードと同様に、平均で全ノードの 3 分の 1 を持ちます。

得られた結果は、ナップザック問題の結果と非常によく似ています。分離された処理パイプラインは、( 素朴なマッピングの 2 倍以上のスループットを持つ) ベースラインとなるソフトウェア版の約 3 倍の性能を達成しました。DPP を使用した場合に FPGA キャッシュに与える影響も小さく、メモリ アクセス レイテンシに対する許容度の高さを示しています。

```
for(k=0; k<V; k++)
   for(i=0; i<V; i++)
    if(i!=k) {
      int dik = dis[i][k];
      for(j=0; j<V; j++)
        if(j!=k) {
          int dkj =  dist[k][j];
          int dij =  dist[i][j];
          if(dik +  dkj < dij )
            dist[i][j] = dik + dkj;
        }
    }
```

図 8 – フロイド-ワーシャル アルゴリズム

この簡単な手法は、メモリ帯域幅を効率的に利用してメモリ レイテンシに対する許容度を高める処理パイプラインを作成することで、Vivado HLS のパフォーマンスを向上させます。この記事で説明した手法は、キャッシュ ミスがアクセラレータの他の部分をストールさせないように、制御データ フローグラフ内のメモリ アクセスと長い依存性サイクルを切り離します。

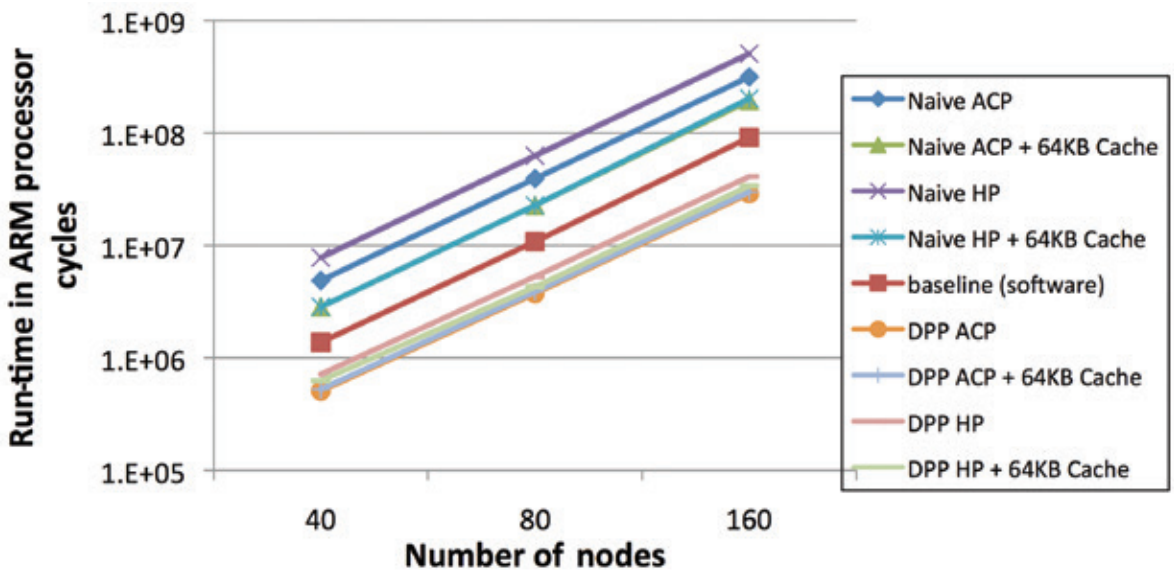
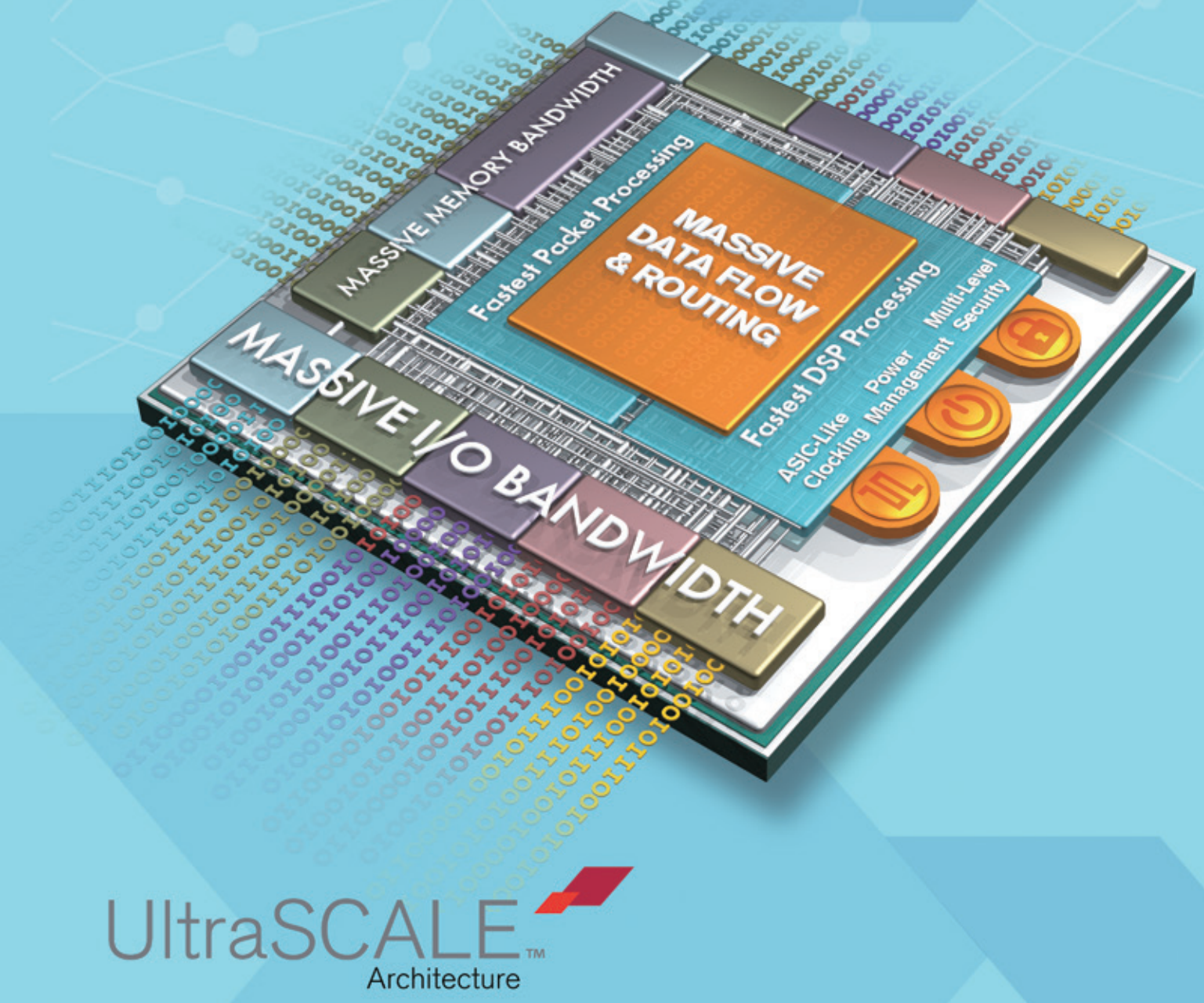



図 9 – フロイド-ワーシャル アルゴリズムの実行時間の比較

# 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
■アヴネット(株) TEL (03) 5792-8210 EVAL-KITS-JP@avnet.co.jp ■(株)PALTEK TEL (045) 477-2001 info_pal@paltek.co.jp

# ザイリンクス トレーニング カリキュラムパス

XILINX

**基本のFPGAアーキテクチャ E-Learning**
- スライス および I/Oリソース
- メモリ および クロッキングリソース
- Architecture Wizard および Foorplan Editor

開発言語

| VHDL 基礎 | Verilog 基礎 | System Verilog での設計 |
|---|---|---|
| VHDL 中級 | Verilog 中級 | System Verilog での検証 |

Vivado デザインツールフロー

VivadoでのFPGA設計導入

VivadoでのFPGA設計実践

- 7シリーズデザイン
  - Chip Scope Pro ツールを使用したデバッグ法
  - FPGA 低消費電力設計のコツ
  - アナログミックスシグナル(AMS)設計
    - Vivado でのアドバンスド FPGA設計
      - パーシャルリコンフィギュレーション
- コネクティビティデザイン基礎
  - メモリインターフェイス設計
    - マルチギガビットシリアルIO
- DSPデザイン基礎
  - Simlink基礎(MathWorks社開催)
    - SystemGenerator を使用したDSPデザイン
      - Cコードベース設計 : Vivado HLS を使用した高位合成
- エンベデッドデザイン基礎
  - Zynq システムアーキテクチャ
    - Zynq エンベデッドシステム開発
      - Zynq エンベデッドシステムソフトウェア開発
        - エンベデッドマイクロブレーズ Linux開発

入門 / 初級 / 中級 / 上級

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**

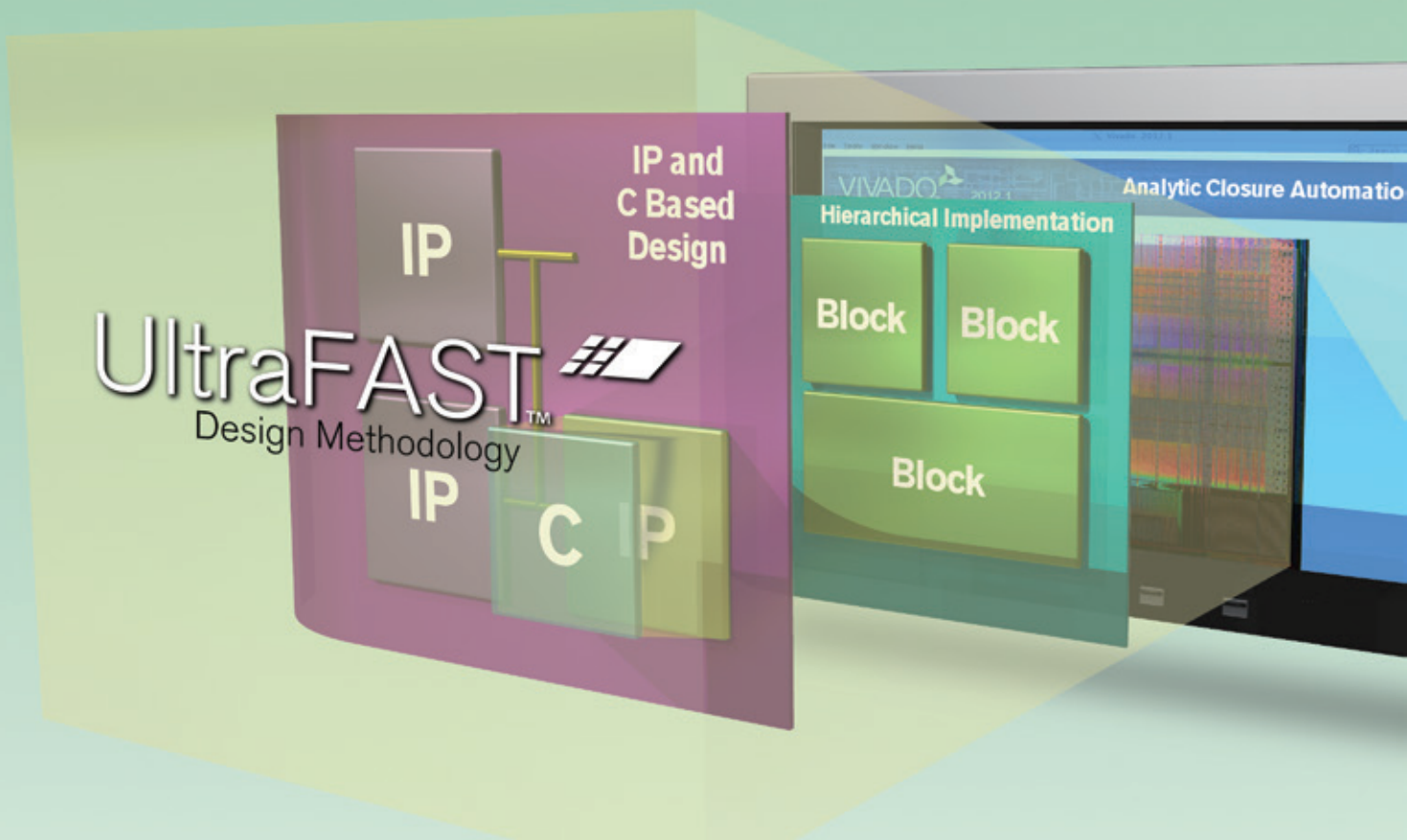
オリジナルトレーニングの内容およびスケジュールは、各社の Web サイトをご覧ください。

| | | |
|---|---|---|
| AVNET | アヴネット(株) | www.avnet.co.jp/training.aspx |
| 新光商事 Shinko Shoji Co.,Ltd. | 新光商事(株) | xilinx.shinko-sj.co.jp/training/index.html |
| PALTEK | (株)パルテック | www.paltek.co.jp/seminar/index.htm |
| hdLab | (株)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
■アヴネット(株) TEL (03) 5792-8210 EVAL-KITS-JP@avnet.co.jp ■(株)PALTEK TEL (045) 477-2001 info_pal@paltek.co.jp


**詳細はこちら： japan.xilinx.com/ultrafast**