Xcell journal

75・76 合併号　2012 SPRING

SOLUTIONS FOR A PROGRAMMABLE WORLD

イノベーションの新時代を担う Zynq-7000 EPP

7 シリーズ FPGA で消費電力を半減

複数の FPGA でコンピューティング フィールドを構築した リコンフィギャラブル システム

FPGA 内の SerDes デザインを 検証するアプローチを比較

ARM AXI4 を サポートする FPGA で 核融合実験への 道筋を切り開く

ページ 28

XILINX®

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 秋山 一雄
kazuo.akiyama@xilinx.com

制作進行 竹腰 美優紀
miyuki.takegoshi@xilinx.com

日本語版 制作・広告・印刷 有限会社エイ・シー・シー

Xcell PUBLICATIONS

japan.xilinx.com/xcell/

Xcell Journal 日本語版 75·76 合併号

2012 年 3 月 1 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

ザイリンクスのベテラン研究者が ACM フェローに

「私は『ものづくり』を目指してエンジニアになりましが、FPGA によって可能なことそのものが『ものづくり』であると強く認識しています」。こう述べるのは、ザイリンクスの特別研究員であり、プログラマブル ロジックとリコンフィギャラブル アーキテクチャの開発、そしてそれらを実現する設計自動化ツールの開発に貢献した Steve Trimberger です。彼は、こういった功績が認められ、2011 年 6 月に ACM (Association for Computing and Machinery：米国計算機学会) フェローの栄誉を受けています。

ザイリンクスの特別研究員
Steve Trimberger、
昨年開催された ACM アウォードにて

Trimberger は 1988 年にザイリンクスに入社して以来、FPGA アーキテクチャおよび EDA ソフトウェアの開発で、複数の世代にわたって中軸的な役割を果たしてきました。現時点で 175 件以上の特許を保有するだけでなく、その他にも数多くの特許を出願中です。

Trimberger が EDA に初めて触れたのはカリフォルニア工科大学の学生時代だったと言います。なお同校では、工学に関する学位と、コンピューター サイエンスに関する博士号を取得しています（修士号はカリフォルニア大学アーバイン校で取得）。「カリフォルニア工科大学在学中に半導体チップの設計に励んでいましたが、チップは必ずしも期待したように動作するとは限りませんでした。そこで、自分でツールを開発すれば問題の原因を把握できるのではないかと考えたのです。そして、チップの開発とデバッグ作業に数回携わったのち、ツールの開発に本格的に取り組むようになりました」(Trimberger)。

Trimberger は、有数な ASIC ベンダーである VLSI Technologies 社に勤めていたときも同様の開発を手掛けてきました。「その当時はツールがほとんど存在しなかったため、ツールを開発さえすればそれだけで感謝されるような時代でした。おそらく誰もが設計したチップに問題を抱えていたはずです」。

その数年後、Trimberger は FPGA (Field Programmable Gate Array) と呼ばれる新しいデバイスを開発した当時株式公開前のザイリンクスへ赴き面接を受けました。

「再プログラム可能なロジック チップを販売している会社があるらしいという話を耳にしたのです。当時の私はこういった会社が長く続くとは思わなかったので、前から欲しいと思っていたチップの開発にかかわれるのであれば、すぐにでも転職したほうがいいと考えました」(Trimberger)。しかし彼の予想に反して、「ASIC では実現できなかった無制限の作り直しができるという FPGA の強み」によって、ザイリンクスには業界にとどまるだけのパワーがありました。

ザイリンクスに入社した当初の Trimberger は、FPGA をプログラミングするソフトウェアを主に担当していましたが、当時のエンジニアリング担当副社長であった Bill Carter の助言もあって、次世代シリコン アーキテクチャの開発に携わるようになりました。Trimberger は次のように述べます。「FPGA と、FPGA をプログラムするツールとを密接に連携させることが、FPGA のビジネスにおける成功の鍵の 1 つです」。

Trimberger が所有する 175 件の特許は、ツール、回路、さらには IC 全体のアーキテクチャなど、幅広い技術分野に及んでいます。初期の特許の中には、シミュレーションを実質的に逆方向に実行して、エラーの原因をピンポイントで速やかに特定できる「後方シミュレータ」といった発明もあります。また、電源が切断されたときにメモリ データが消失する現象を利用して、データの消失率からデバイスが電源オフとなっていた時間を求める方法を編み出したのも彼の功績によるものです。

Trimberger は、『Automated Performance Optimization of Custom Integrated Circuits』、『An Introduction to CAD for VLSI』、および『Field Programmable Gate Array Technology』という 3 冊の本を執筆しています。また、サンタクララ大学にて大学院生を対象に EDA ソフトウェア設計に関する教鞭を取っています。プライベートでは、「FIRST Robotics」コンテストを目指す高校生チームを指導しています (Trimberger が指導するチームの 1 つが 2011 年に一位を獲得)。

ACM は教育分野および科学分野を対象とした世界最大規模の計算機学会です。同学会フェローの栄誉を受けた Steve Trimberger に祝意を表すと共に、最先端の FPGA 技術の発展に尽くし、お客様の新たな技術革新に大きく貢献した同氏の功績を称えたいと思います。

Mike Santarini
発行人

XILINX® ウェブセミナー Web Seminar

プログラマブルロジック ソリューションのリーダーであるザイリンクスの
FPGA/CPLD/EPP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
Zynq-7000 EPP アーキテクチャとエコシステム

FPGA入門編

FPGAをこれから始める方に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、技術管理者の方へなぜ今FPGA /CPLDなのかをご説明します。

- 30分で判る! FPGA入門
- 15分で判る! FPGA採用理由

FPGA活用編

ザイリンクスFPGA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。

- 7シリーズ FPGAで消費電力を大幅に削減
- ISE12を使用したパーシャル リコンフィギュレーションでシステムのコストと消費電力を最適化
- ザイリンクスで創造するコンシューマ製品の新しい価値
- ザイリンクス最新ロードマップ
- Virtex®-6、Spartan®-6 FPGAでの低消費電力デザインの実現

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
- 製品の差別化を実現する開発ツール:ISE Design Suite

FPGA/CPLD/EPP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。

- エクステンシブル プロセッシング プラットフォーム Zynq-7000 ファミリのご紹介
- 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/**

VIEWPOINTS

12

28

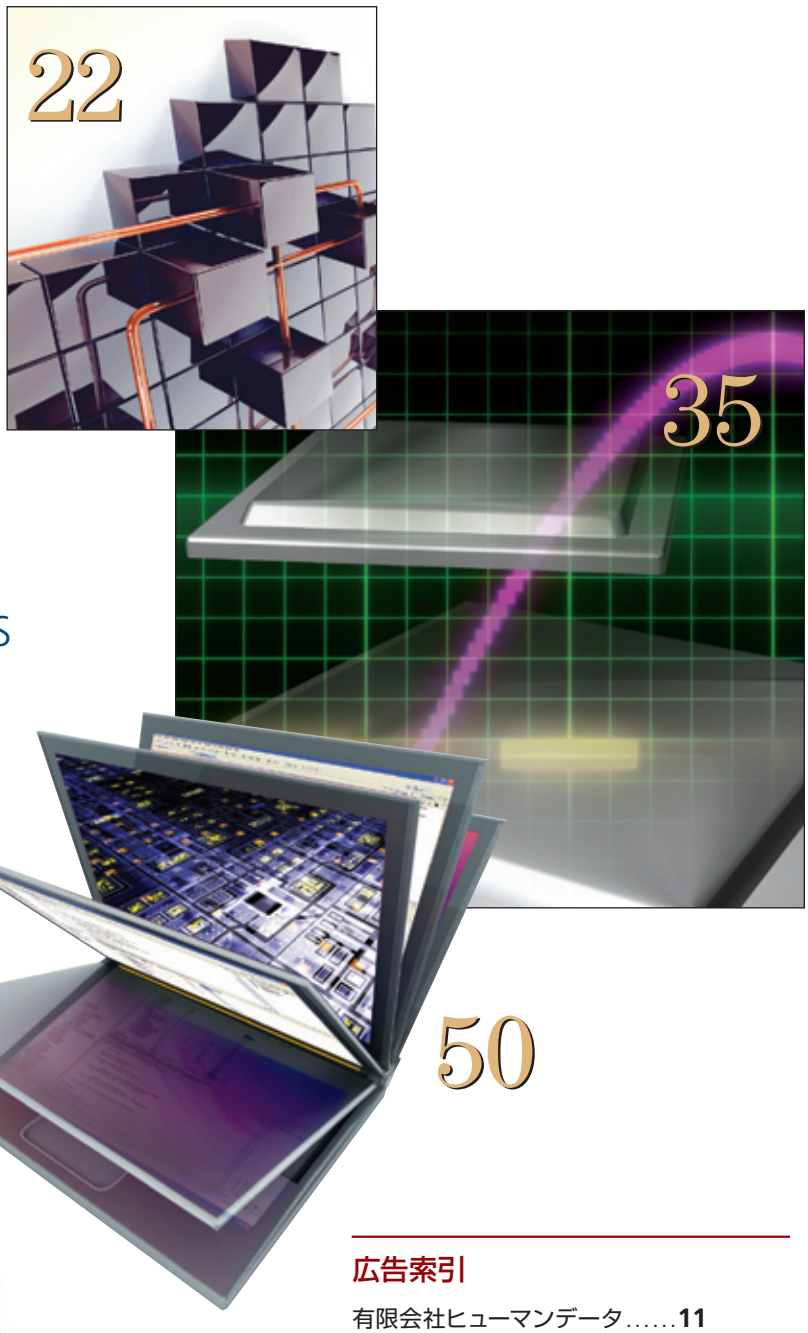
Cover Story

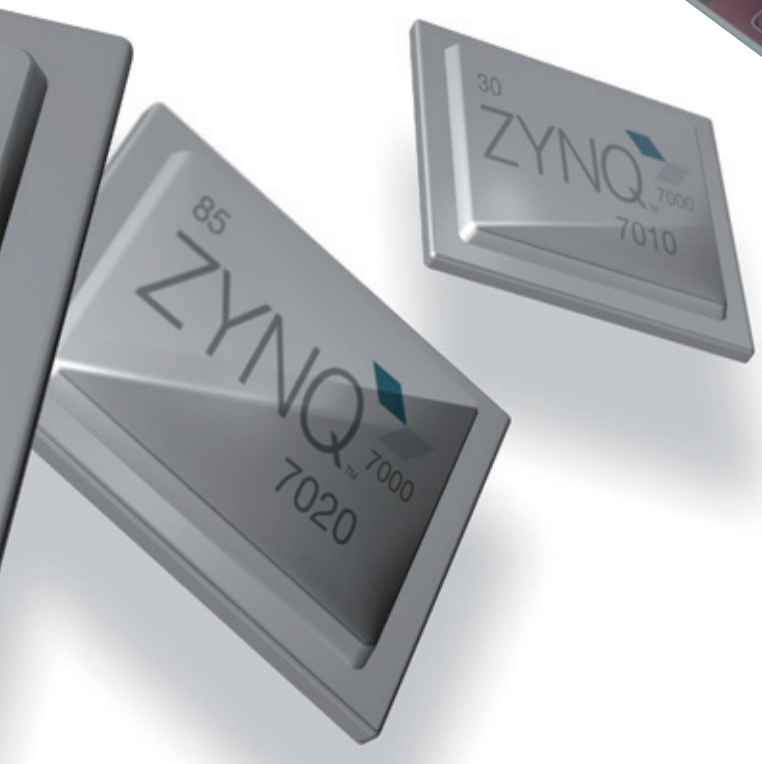
イノベーションの新時代を拓く
Zynq-7000 EPP

4

XCELLENCE BY DESIGN APPLICATION FEATURES

THE XILINX XPERIENCE FEATURES





広告索引

Xcell Journal のご送付先住所等の変更は：
http://japan.xilinx.com/xcell/henko/

Xcell Journal の新規定期購読のお申込みは：
http://japan.xilinx.com/xcell/toroku/

Zynq-7000 EPP Sets Stage for New Era of Innovations

イノベーションの新時代を拓く Zynq-7000 EPP

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

ザイリンクスの Zynq-7000 エクステンシブル プロセッシング プラットフォーム ファミリは、2 つの ARM Cortex-A9 MPCore プロセッサを搭載したプロセッシング システムとプログラマブル ロジック、そしてハードマクロの IP ペリフェラルをワンチップに集積し、柔軟性、コンフィギュレーション性、パフォーマンスにおいてベスト バランスを実現

ザイリンクスはこれまで、2 つの ARM Cortex™-A9 MPCore プロセッサと低消費電力のプログラマブル ロジック、およびハードマクロのペリフェラル IP をすべてワンチップに集積した革新的なアーキテクチャを発表してきました (Xcell Journal 71 & 72 合併号のカバー ストーリーを参照 : http://japan.xilinx.com/publications/archives/xcell/issue71-72/xcell71_72.pdf)。2011 年 3 月、このエクステンシブル プロセッシング プラットフォーム (EPP) に基づいて構築された新しいファミリの第一弾として Zynq™-7000 EPP ファミリと名付けられた 4 つのデバイスを正式に発表しました。

28nm プロセス テクノロジで製造された Zynq-7000 デバイスには、いずれも ARM デュアルコア Cortex-A9 MPCore プロセッシング システムが搭載されています。このプロセッシング システムには、NEON メディア処理エンジン、倍精度浮動小数点ユニット、レベル 1 (L1) およびレベル 2 (L2) キャッシュ、マルチ メモリ コントローラー、そして一般的なペリフェラルが数多く用意されています (図 1)。これまで、ハードワイヤード プロセッサやソフト オンボード プロセッサを搭載した FPGA デバイスについてはベンダー各社から販売されていましたが、Zynq-7000 EPP はプログラマブル ロジックではなく ARM プロセッサ システムが中心的な役割を果たしているという点でこれらの製品とは一線を画しています。つまり、このデバイスは電源を投入すると FPGA ロジックよりも先にプロセッシング システムが起動し、プログラマブル ロジック ファブリックから独立してさまざまなオペレーティング システム (OS) を実行できるように設計されています。その後の動作は、設計者がプロセッシング システムをプログラムして適宜プログラマブル ロジックをコンフィギュレーションします。

このアプローチでは、通常の完全な機能を備えた ARM プロセッサを使用したシステム オン チップ (SoC) とまったく同じソフトウェア プログラミング モデルを使用できます。従来のインプリメンテーションでは、オンボード プロセッサを動作させるには設計者が FPGA ロジックをプログラムする必要があり、これらのデバイスを使用できるのは事実上 FPGA 設計者に限られていました。しかし、このような状況は Zynq-7000 EPP には当てはまりません。

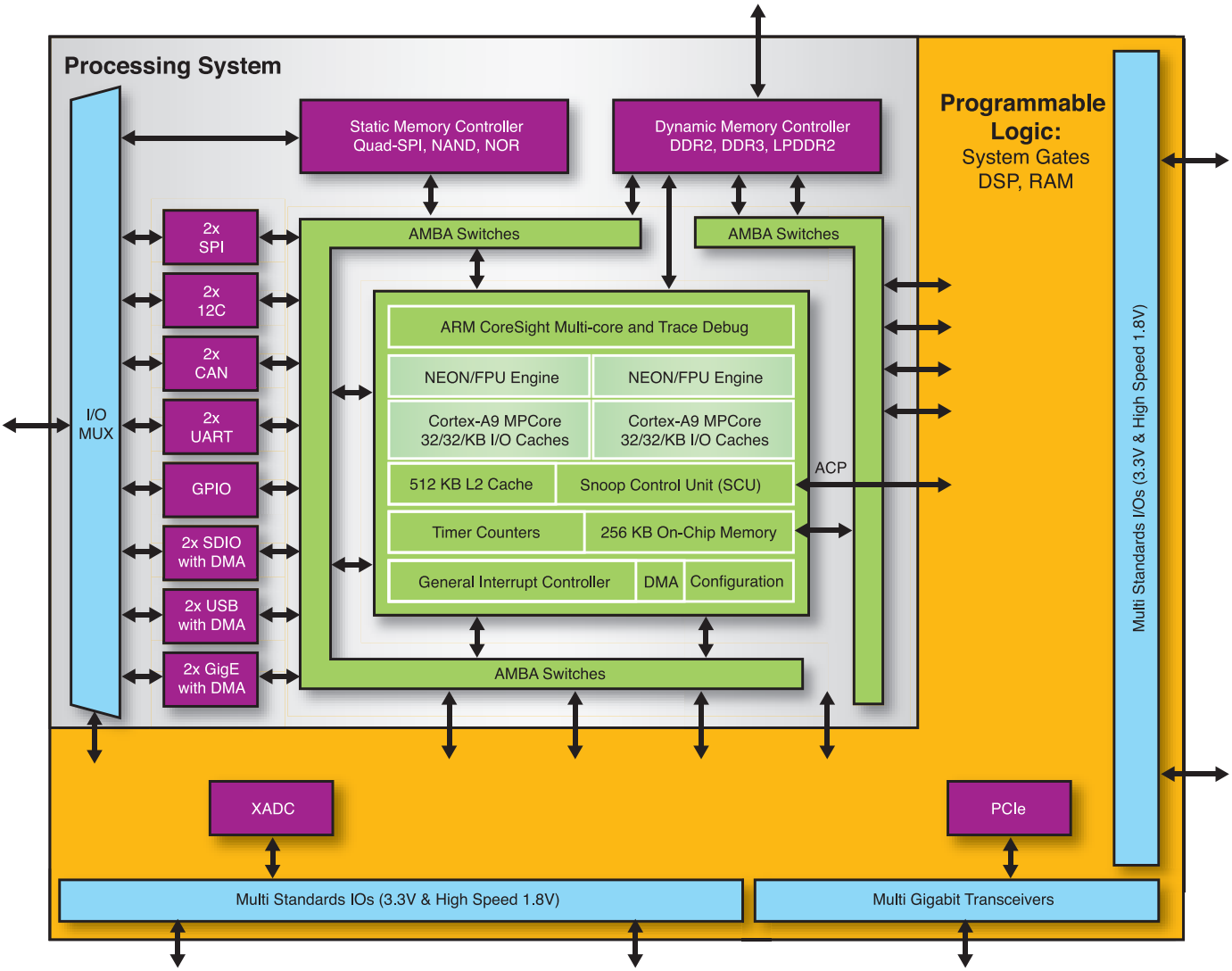


図 1 – FPGA ファブリック内で複数の MPU を統合した従来のチップとは異なり、ザイリンクスの新しい Zynq-7000 EPP ファミリではプログラマブル ロジックではなく ARM プロセッサが中心的な役割を果たします。

この新しい製品ファミリには、これまでの半導体チップにはなかったハードウェアとソフトウェアの高度かつ汎用的なプログラミング性が備わり、チップを最初から設計することに伴うスケジュールの遅れやリスクを気にすることなく、短期間で革新的な SoC を開発できます。このため、Zynq-7000 EPP を使用すればハードウェア設計、ソフトウェア開発、システム開発の専門家だけでなく、メーカーも含めたより多くのイノベーターたちがプロセッシング システムとプログラマブル ロジックの組み合わせによって実現するさまざまな可能性を探り、まだ誰も想像したことがないアプリケーションを開発できるようになります。

ザイリンクスのプロセッシング プラットフォーム担当副社長、Larry Getman は次のように述べています。「Zynq-7000 EPP は、単なるプロセッサでもなければ、単なる FPGA でもないという点で根本的に今までの半導体製品とはまったく異なっています。これら 2 つの長所を組み合わせることで、従来の 2 チップ構成のソリューションや ASIC が抱えていた制約の多くを解消しています」。

現在のエレクトロニクス システムの多くは、1 つのプリント基板上に FPGA とスタンドアロン プロセッサ、または FPGA とオンボード プロセッサを備えた ASIC を組み合わせた構成になっている

と Getman は指摘しています。ザイリンクスから Zynq-7000 が登場したことにより、これまで 2 チップ構成のソリューションを使用していたユーザーもワンチップで次世代システムを構築できるようになり、部材コストやプリント基板面積、そして全体的な消費電力を抑えることができます。また、プロセッサと FPGA が同じファブリック上にあるため、パフォーマンスも飛躍的に向上します。

さらに、Getman は現在市場で進行している ASIC から FPGA への移行が加速すると指摘しています。増大するアプリケーションの種類に対して、ASIC を最新のプロセス技術で製造することは費用がかかり過ぎ、リスクも甚大です。これを受けて多くの企業では FPGA を採用する動きが広がっています。ASIC での開発を続けたい場合は枯れたプロセスを使用する傾向があり、アナリストはこうした製品を「バリュー志向の SoC ASIC」と呼びます。それでも ASIC には長い設計サイクルがつきもので、何度もリスピン（設計のやり直し）が発生するリスクもあります。リスピンが発生すると開発コストが非常に高くつき、製品のタイムリーな市場投入の妨げとなる場合があります。「28nm プロセスで製造されている Zynq-7000 EPP は、プログラマブル ロジック部分も以前のテクノロジ ノードに比べてサイズやパフォーマンスの面でまったく引けをとりません。しかもプロセッシング サブシステム内に 28nm の SoC をハードマクロとして集積できる利点もあります。15 ドル未満からという価格設定の前では、非常にボリュームの大きい量産製品を除き、ASIC 開発のコストとリスクを甘受する理由はほとんど存在しません。ソフトウェア チームもハードウェア チームもプロジェクト初日からフル稼働できます。この点だけでも、エンジニアリング チームが ASIC を使用し続ける正当な理由を見つけるのは困難でしょう」(Getman)。

ザイリンクスがこのアーキテクチャを発表して以来、Zynq-7000 EPP には非常に大きな反響と問い合わせがあったとした上で、Getman は次のように述べています。「すでに、アルファ カスタマーの間では Zynq-7000 デバイスを使用したシステムのプロトタイプ作成が始まっています。このテクノロジには非常に大きな魅力があります」。

スマートなアーキテクチャを採用

Zynq-7000 EPP の設計チームは、ザイリンクスのプロセッシング ソリューション担当副社長 Vidya Rajagopalan の指揮のもと、熟考を重ねてこのまったく新しいデバイスのアーキテクチャを設計しました。抜群の普及率を誇る ARM プロセッサ システムを採用したことはもちろんですが、それ以上にアーキテクチャの鍵となったのは、プロセッシング システムとプログラマブル ロジックの接続に広帯域の AMBA® AXI™ (Advanced Extensible Interface) インターコネクトを使用したことでした。これにより ARM デュアル コア Cortex-A9 MPCore プロセッシング システムとプログラマブル ロジック間で消費電力を非常に少ないレベルに抑えながらマルチギガビットのデータ転送が可能となり、制御、データ、I/O、メモリ間の一般的なボトルネックが解消されています。

ザイリンクスは、ARM 社と共同で ARM アーキテクチャを FPGA アプリケーションにより適した形に改良する作業にも取り組みました。「AXI4 にはメモリ マップ方式とストリーミング方式の 2 種類があります。広帯域ビデオなどのアプリケーション用に開発されている IP の多くはストリーミング IP であるため、ARM 社に働きかけてストリーミング方式の定義を整備してもらいました。同社にはこのストリーミング インターフェイスに対応した製品がなかったため、ザイリンクスが協力する形で開発を進めました」(Rajagopalan)。

Getman は、このアーキテクチャのもう 1 つの重要な特長として、標準的なインターフェイス IP を理想的な組み合わせでハードマクロとして Zynq-7000 EPP シリコンに組み込んだ点を挙げています。「Zynq-7000 EPP には、USB、イーサネット、SDIO、UART、SPI、I2C、GPIO など、広く普及している標準的なインターフェイスのみをペリフェラルとして選択しました。唯一の例外は CAN ペリフェラルです。CAN はハードマクロとして組み込むにはやや用途が限られますが、ザイリンクスが重要なターゲット市場と考える産業および車載分野では広く使用されています。CAN ペリフェラルをハードマクロとしてデバイスに組み込んだことで、Zynq-7000 EPP はより一層使い勝手のよい製品に仕上がっています」(Getman)。

メモリに関しては、Zynq-7000 デバイスには 2 つのプロセッサで共有する最大 512KB の L2 キャッシュがあります。「Zynq-7000 EPP デバイスには 256KB のスクラッチパッドがあり、これをプロセッサと FPGA の両方からアクセス可能な共有メモリとして使用します」(Getman)。

DDR コントローラーは、3 種類の DDR (Double Data Rate) メモリをサポートした非常にユニークなものです。「ASSP は特定の市場をターゲットにしたものがほとんどですが、Zynq-7000 デバイスは LPDDR2、DDR2、DDR3 をサポートしているため、性能と消費電力のどちらを重視するかによってユーザーによる柔軟なトレードオフが可能です。このように複数の DDR 規格をサポートしたコントローラーはまだ市場にほとんど存在しません」(Rajagopalan)。

Zynq-7000 EPP は新世代のデバイスとしてだけでなく、ベース開発ボード、ソフトウェア、IP、各種ドキュメントを含めた最新のザイリンクス ターゲット デザイン プラットフォームとしても提供されます。また、今後は特定の垂直市場やアプリケーション分野に特化した Zynq-7000 EPP ターゲット デザイン プラットフォームの提供も予定されており、ボード / ドーターカード、IP、各種ドキュメントによって製品の迅速な市場投入を強力に後押しします (Xcell Journal 67 & 68 合併号の「ターゲット デザイン プラットフォームで技術革新を加速」を参照 : http://japan.xilinx.com/publications/archives/xcell/xcell67_68.pdf 参照)。

ザイリンクス アライアンス プログラムのメンバーや ARM Connected Community からも、一般的な OS、デバッガー、IP、リファレンス デザイン、その他の学習用および開発用資料を含め、Zynq-7000 EPP に関するリソースが豊富に提供されます。

このようにシリコンの優位性とツールの充実に加え、ザイリンクスは Zynq-7000 EPP の設計フローおよびプログラミング フローに関してもユーザー フレンドリーとなるように細心の注意を払いました。

プロセッサ中心型の開発フロー

Zynq-7000 EPP はごく一般的なツール フローを採用しており、組み込みソフトウェアやハードウェアのエンジニアはザイリンクスの ISE® Design Suite やサードパーティのツールで提供されていた馴染みのあるエンベデッド機器設計メソドロジを利用できるため、開発、デバッグ、インプリメンテーションは現在とほとんど同じ方法で行えます (図 2)。

Getman は、ソフトウェア アプリケーション エンジニアが過去のデザインで使用した開発ツールと同じものを使用できることにも触れています。エンジニアは、組み込みソフトウェア アプリケーション プロジェクト向けにザイリンクスが提供している Eclipse ベースのツール スイートであるソフトウェア開発キット (SDK) を利用できるほか、サードパーティから提供されている開発環境を使用することもできます。たとえば、ARM Development Studio 5 (DS-5™) や ARM RealView Development Suite (RVDS™) など、ARM エコシステムから提供されている各種開発ツールはいずれも利用できます。 Linux アプリケーションを開発する際は、Zynq-7000 デバイスに搭載された 2 つの Cortex-A9 CPU コアを対称型マルチプロセッサ モードで利用することにより、最大限のパフォーマンスが得られます。もしくは、これらの CPU コアをユニプロセッサや非対称型マルチプロセッサとしてセットアップし、Linux または VxWorks などのリアルタイム OS (RTOS)、あるいはこれら両方を実行することもできます。ソフトウェア開発をスムーズに開始できるよう、ザイリンクスからはオープンソース Linux ドライバー、およびプロセッシング システムのすべてのペリフェラル (USB、イーサネット、SDIO、UART、CAN、SPI、I2C、GPIO) に対応するベアメタル ドライバーが提供されています。ミドルウェアとアプリケーション ソフトウェアを完備した OS/RTOS ボード サポート パッケージ (BSP) も、ザイリンクスをはじ

同一のプロセッシング システムで 4 つのデバイスを展開

Zynq-7000 EPP ファミリの 4 種類のデバイス (右図参照) はすべて同じ ARM プロセッシング システムを搭載していますが、プログラマブル ロジックのリソースが異なるため、用途に応じて使い分けることができます。

Cortex-A9 マルチプロセッサ コア (MPCore) は 2 つの CPU で構成され、各 Cortex A9 MPCore プロセッサは専用の NEON コプロセッサ (オーディオ、ビデオ、3D グラフィックス、イメージ、音声処理用の命令を追加したメディア処理 / 信号処理アーキテクチャ) と倍精度浮動小数点数ユニットを搭載しています。Cortex-A9 プロセッサは、L1 キャッシュ サブシステムを備えた高性能で低消費電力の ARM マクロセルで、仮想メモリ機能を完全にサポートしています。このプロセッサは ARMv7™ アーキテクチャを実装し、32 ビット ARM 命令や 16 ビット /32 ビット Thumb 命令のほか、Jazelle ステートで 8 ビット Java バイト コードを実行できます。このほかにも、プロセッシング システムにはスヌープ制御ユニット、L2 キャッシュ コントローラー、オンチップ SRAM、タイマーとカウンター、DMA、システム制御レジスタ、デバイス コンフィギュレーション、ARM CoreSight™ システムが含まれます。デバッグ用には、ARM 社から提供されるエンベデッド トレース バッファー (ETB)、インスツルメンテーション トレース マクロセル (ITM)、クロス トリガー モジュールのほか、ザイリンクスの AXI モニターとファブリック トレース モジュールが含まれます。

4 つのデバイスのうち大型の Zynq-7030 と Zynq-7045 デバイスは、最大 12.5Gbps で動作可能なマルチギガビット トランシーバーを内蔵した高速で低消費電力のシリアル インターフェイスを備えています。これらのデバイスはそれぞれ約 1.9M および 5.2M ASIC ゲート相当 (125k および 350k ロジック セル) で、DSP リソースのピーク性能はそれぞれ 480 GMAC と 1080 GMAC です。小型デバイスの Zynq-7010 と Zynq-7020 はそれぞれ約 430k および 1.3M ASIC ゲート (28k および 85k ロジック セル) に相当し、DSP のピーク性能はそれぞれ 58 GMAC と 158 GMAC です。

各デバイスには、12 ビット、1MSPS ADC (x2)、オンチップ センサー、外部アナログ入力チャネルを備えた汎用の A/D コンバーター (XADC) インターフェイスがあります。XADC は、これまでの Virtex® FPGA のシステム モニターよりも機能が強化されています。2 つの 12 ビット ADC は最大 17 の外部入力アナログ チャネルをサンプリングでき、500kHz 未満の帯域幅でアナログ信号を処理したいという幅広いアプリケーションのニーズに応えます。

Zynq-7000 EPP の第一弾として発表された 4 つのデバイスは同一の ARM プロセッシング システムを搭載し、さまざまなプログラマブル ロジックのリソースを提供します。ゲート数は 430K ～ 3.5M ASIC ゲート相当までと多岐にわたっています。

め ARM パートナー各社から提供される予定です。

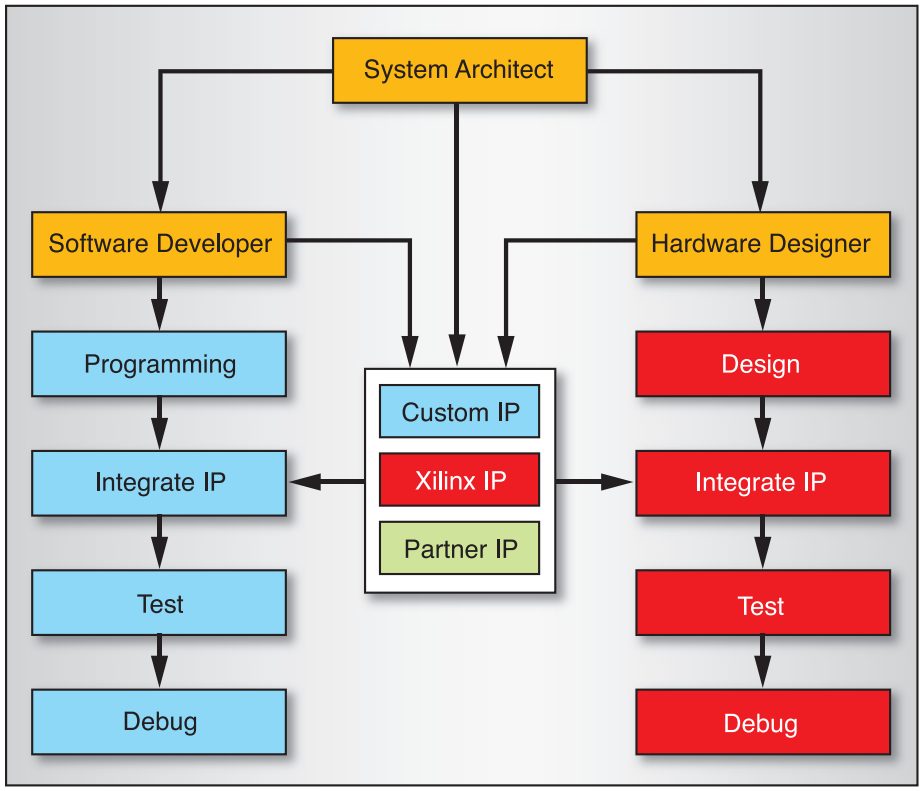
ハードウェア設計フローに関しても、ISE Design Suite を使用してエンベデッド プロセッサを設計する場合のフローとほぼ同じで、EPP 用の手順をいくつか追加するだけで完了します。プロセッシング サブシステムは、完全なデュアル プロセッサ システムによく使用されるペリフェラルを数多く組み合わせて構成されています。ハードウェア設計者は、プログラマブル ロジックのソフト IP ペリフェラルをプロセッシング サブシステムに接続することで処理能力を強化できます。一般的なハードウェア開発工程の多くはハードウェア開発ツールの Xilinx Platform Studio によって自動化され、デバイスのピン配置も容易に最適化できます。「ISE には、ハードウェア ブレークポイントやクロス トリガーなど、協調デバッグ向けの機能をいくつか追加しました。今回最も重視したのは、ソフトウェア開発者とハードウェア設計者の両者に快適な設計開発環境を提供することでした」(Getman)。

考え抜かれたプログラミング メソドロジ

ザイリンクスは、コンフィギュレーションされたプログラマブル ロジックを AXI インターコネクト ブロックを経由して ARM コアに接続することにより、プロセッシング システムの機能と性能を拡張できる仕組みを用意しています。ザイリンクスと ARM パートナー エコシステムからは、FPGA のプログラマブル ロジックにインプリメント可能な AMBA インターフェイスのソフト IP コアが数多く提供されています。設計者はこれらの IP コアを使用することで、ターゲット アプリケーションで必要なカスタム機能を自由に構築できます。Zynq-7000 デバイスのプログラマブル ロジックは 7 シリーズ FPGA と同様の構造となっているため、設計者は 1 種類の固定したプログラマブル ロジック コンフィギュレーションまたは複数のコンフィギュレーションをロードできるほか、パーシャル リコンフィギュレーションを利用してプログラマブル ロジックの機能を必要に応じてオンザフライで書き換えることもできます。

プロセッシング システムとプログラマブル ロジックを結ぶインターコネクトの動作が意識されることはほとんどありません。マスターとスレーブ間のアクセスは、各スレーブ デバイスに割り当てられたアドレス範囲に基づき AXI インターコネクトを介して配線されます。複数のマスターから複数のスレーブへの同時アクセスが可能で、各 AXI インターコネクトは 2 つのレベルのアービトレーションを使用して競合を回避します。

図 2 – Zynq-7000 EPP には、システム アーキテクト、ソフトウェア開発者、ハードウェア設計者が使い慣れたツール フローをそのまま使用できます。



お乗り遅れのなきよう

アーリー アクセス プログラムに参加しているユーザーは、直ちに Zynq-7000 EPP ファミリの評価を始めることができます。2011 年後半にファースト シリコン デバイスの出荷が開始され、2012 年前半にはエンジニアリング サンプル (ES) が提供される予定です。設計者は、ARM 対応のツールや開発キットを使用して Cortex-A9 MPCore アーキテクチャを体験し、コードの移植に取り掛かれます。

Zynq-7000 EPP ファミリの価格はデバイスの種類や数量によって異なりますが、量産開始時には最も安価なもので 15 ドル未満の価格設定となる見通しです。詳細は、japan.xilinx.com/zynq をご覧いただくか、ザイリンクス販売代理店までお問い合わせください。

組み込みプロセッサの新しいカタチ、ZYNQをご紹介します。プロセッサはついにFPGAと統合され、これまで私たちがエクステンシブル プロセッシング プラットフォームと呼んでいた製品はZynq™と名づけられました。おなじみのプログラム手法がさらに直観的になり、お客様の要求を完全にカスタマイズでき、すばやく実装し、いち早く市場に投入できます。あなたがソフトウェアエンジニアで、ARM® Cortex™をご存知ならば、すでにZynqを知っているといえます。あなたがザイリンクスをご存知であれば、これが期待できるイノベーションであるとおわかりになると思います。
詳細は、http://japan.xilinx.com をご覧ください。

XILINX
ZYNQ™

ザイリンクス株式会社
製品のお問い合わせは下記の販売代理店へどうぞ

■東京エレクトロン デバイス(株) TEL(045)443-4016 x2web@teldevice.co.jp ■アヴネット ジャパン(株) TEL(03)5792-8210 EVAL-KITS-JP@avnet.com ■(株)PALTEK TEL(045)477-2005 info_pal@paltek.co.jp
■富士通エレクトロニクス(株) TEL(045)415-5825 fei-xinq@cs.jp.fujitsu.com ■新光商事(株) TEL(03)6361-8086 X-Pro@shinko-sj.co.jp

HUMANDATA

FPGA / CPLD 評価ボード

すぐに使えるFPGAボードで、設計や試作コストが下げられると思います。

- FPGAの動作に必要な最低限の機能を搭載単一電源ですぐに活用できます
- ACM/XCMシリーズはそれぞれ外形やコネクタ位置が同一で置き換えが可能です
- 豊富なラインナップで100種類以上の製品をご用意しています
- 回路図、マニュアルは購入前でも自由に参照できます
- 豊富な納入実績で安心してお使いいただけます
- 基本的に即納体制で最短翌日からご活用いただけます
- スピードグレード変更などのカスタマイズもご相談ください

Spartan-6使用製品例

Spartan-6 FGG484 搭載 FPGA ボード
XCM-110/110Z シリーズ

ハーフカードサイズ / MRAM 搭載 / DDR2 SDRAM 搭載

XC6SLX45-2FGG484C、XC6SLX75-2FGG484C、XC6SLX100-2FGG484Cまたは XC6SLX150-2FGG484Cを搭載

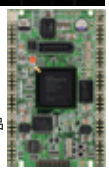
RoHS 指令対応品

¥44,000(税込46,200)～

Spartan-6 LXT FGG484 搭載 FPGA ボード
XCM-020 シリーズ

クレジットカードサイズ / Rocket I/O / DDR2 SDRAM 搭載 / MRAM 搭載

XC6SLX45T-2FGG484C、XC6SLX75T-2FGG484C XC6SLX100T-2FGG484CまたはXC6SLX150T-2FGG484Cを搭載



RoHS 指令対応品

¥66,000(税込69,300)～

Spartan-6 FGG484 搭載 FPGA ボード
XCM-018/018Z シリーズ

クレジットカードサイズ / MRAM 搭載 / DDR2 SDRAM 搭載

XC6SLX45-2FGG484C、XC6SLX75-2FGG484C、XC6SLX100-2FGG484CまたはXC6SLX150-2FGG484Cを搭載

RoHS 指令対応品

¥40,000(税込42,000)～

Spartan-6 FGG484 搭載 FPGA ボード
XCM-019 シリーズ

クレジットカードサイズ

XC6SLX45-2FGG484CまたはXC6SLX75-2FGG484Cを搭載

RoHS 指令対応品

¥39,000(税込40,950)～

Virtex-5使用製品例

Virtex-5 LXT FFG665 搭載 ブレッドボード
XCM-107 シリーズ

ハーフカードサイズ / Rocket I/O

XC5VLX30T-1FFG665CまたはXC5VLX50T-1FFG665Cを搭載

RoHS 指令対応品

¥118,000(税込123,900)～

Virtex-5 FFG676 搭載ブレッドボード
XCM-109 シリーズ

ハーフカードサイズ / SDRAM 搭載

XC5VLX30-1FFG676C、XC5VLX50-1FFG676C、XC5VLX85-1FFG676CまたはXC5VLX110-1FFG676Cを搭載

RoHS 指令対応品

¥77,000(税込80,850)～

Virtex-5 FFG676 搭載 ブレッドボード
XCM-011 シリーズ

クレジットカードサイズ / FRAM 搭載 / SDRAM 搭載

XC5VLXの676ピンBGAチップ搭載

RoHS 指令対応品

¥108,000(税込113,400)～

FPGA Virtex-5 搭載 USB-FPGA ボード
EDX-006

クレジットカードサイズ / USB Config / USB Comm / MRAM 搭載

XC5VLX30-1FFG676C搭載

RoHS 指令対応品

¥99,000(税込103,950)

Spartan-3A/3AN使用製品例

Spartan-3A VQG100 搭載ブレッドボード
XCM-304

セミカードサイズ

XC3S200A-4VQG100Cを搭載

RoHS 指令対応品

¥19,000(税込19,950)

Spartan-3A DSP FGG676 搭載ブレッドボード
XCM-016 シリーズ

クレジットカードサイズ / FRAM 搭載 / SDRAM 搭載

XC3SD1800A-4FGG676CまたはXC3SD3400A-4FGG676Cを搭載

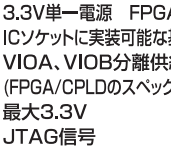
RoHS 指令対応品

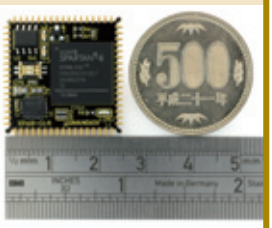
¥49,000(税込51,450)～

PLCC68シリーズ

ICソケットに実装できるFPGA/CPLDモジュール

PLCC68シリーズは、ICソケットに搭載できるように設計された、FPGA・CPLDモジュールです。
市販のICソケットに実装できるので、ユニバーサル基板でのFPGAやCPLDの利用に便利です。
また、シリーズ別にピン割付が共通で、FPGA/CPLDのベンダーやタイプによらず電源やI/Oが共通となっています。
単電源動作も、IO電源を2系統分離して使用することも可能となっています。





- 50本のI/O(内4本はクロック兼用を優先的に割付)
- 3.3V単一電源　FPGA/CPLDが必要な補助電源を内蔵
- ICソケットに実装可能な基板サイズ25.3 × 25.3 [mm]
- VIOA、VIOB分離供給可能 (FPGA/CPLDのスペックによる) 最大3.3V
- JTAG信号
- 全シリーズ共通ピン割付
- 全シリーズRoHS指令対応品

twitter Follow me on Twitter

PLCC 68PIN Spartan-6 FPGA モジュール
XP68-01 シリーズ

- XC6SLX16-2CSG225Cを搭載
- 50本のI/Oを外部引き出し
- 3.3V単一電源動作
- オンボードクロック搭載(50MHz)
- コンフィギュレーションROM搭載
- 6層基板を採用
- ICソケットに実装可能

RoHS指令対応品

¥12,800(税込13,440)

PLCC 68ピン Spartan-3AN FPGA モジュール
XP68-02 シリーズ

- XC3S200AN-4FTG256Cを搭載
- 50本のI/Oを外部引き出し
- 3.3V単一電源動作
- オンボードクロック搭載(50MHz)
- コンフィギュレーションROMはFPGAに内蔵
- 6層基板を採用
- ICソケットに実装可能

RoHS指令対応品

¥12,000(税込12,600)

PLCC 68ピン Spartan-6 FPGA モジュール
XP68-03 シリーズ

- XC6SLX45-2CSG324Cを搭載
- 50本のI/Oを外部引き出し
- 3.3V単一電源動作
- オンボードクロック搭載(50MHz)
- コンフィギュレーションROM搭載
- 6層基板を採用
- ICソケットに実装可能

RoHS指令対応品

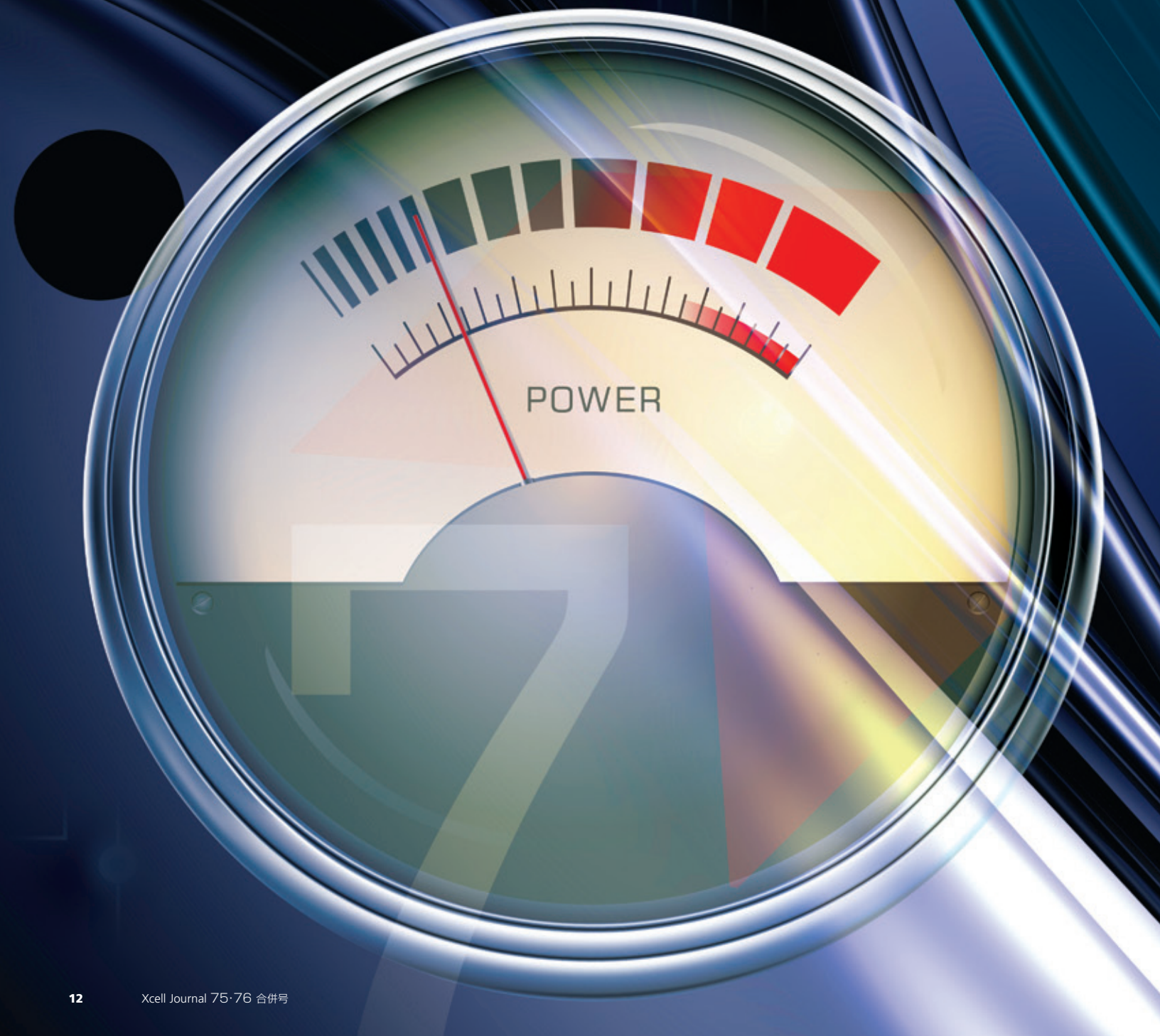
¥19,500(税込20,475)

※その他 FPGA Boardやアクセサリを100種類以上ラインナップしています。詳しくはウエブをご覧ください。

有限会社ヒューマンデータ　大阪府茨木市中穂積1-2-10 ジブラルタ生命茨木ビル 〒567-0034
072-620-2002　072-620-2003　s2@hdl.co.jp
URL www.hdl.co.jp

How Xilinx Halved Power Draw in 7 Series FPGAs

7 シリーズ FPGA で消費電力を半減



FPGA 専用のシリコン プロセスと革新的な統一アーキテクチャを開発し、従来世代比で 50% 以上の消費電力削減を実現

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

ザイリンクスは 7 シリーズ FPGA の仕様策定にあたって数多くの顧客企業と打ち合わせを重ねてきましたが、その中で消費電力（パワー）という言葉が繰り返し登場することに気付きました。その明確な顧客ニーズを踏まえて、2011 年 3 月に出荷を開始した 最新の 28nm FPGA の開発では、消費電力の削減と電力管理を最優先課題に据えました。実際に 7 シリーズ FPGA では、ロジック性能、I/O 性能、およびトランシーバー性能（最高 28Gbps）の向上を図ると共に、これまでで最大のロジック容量を実現しながら、消費電力を従来世代と比較して半分以下に抑えることができました。

このような大幅な消費電力削減を実現できた最大の要因として、7 シリーズ FPGA を製造するために、ザイリンクスと Taiwan Semiconductor Manufacturing Company（TSMC）社とで共同開発した FPGA 専用の 28nm HPL プロセスを採用したことが挙げられます。このプロセスは低消費電力というだけでなく、ほかのプロセスで製造した FPGA では不可能な、パワービニングや電圧スケーリングを実現します。また 7 シリーズ FPGA では、FPGA に最適なプロセスを選択したことと合わせて、消費電力のさらなる削減を図るためにデバイスアーキテクチャの再定義も行っています。

なお、ザイリンクス製 FPGA の消費電力プロファイルを評価する解析ツールの新しいバージョンがリリースされる予定です。

最も求められている機能

多くの FPGA ユーザーが最も求める機能として電力管理機能がリストアップされています。その理由をあらためて取り上げてみましょう。これまで電源を AC コンセントから取るようなシステムを設計する場合は、使用する FPGA の消費電力を特に考慮する必要はなく、FPGA を選択するときに性能とロジック容量だけを考えれば充分でした。では、なぜこれが変化したのでしょうか。

半導体業界はここ 10 年以上の間に、トランジスタのリーク電流が大きくなるというリスクが伴うことはわかっていながらも、さらなる高速性を求めて、プロセス微細化への道を歩んできました。同時にシステム メーカーは、消費電力の小さい製品で総所有コストや運用コストの低減を実現し、製品を差別化しようと努めてきました。また、DC 電源 (バッテリ) で動作するモバイル機器の開発にも取り組んでいます。すなわちシステム メーカーは、対象とする製品のいかんを問わず、消費電力の削減とシステムの電力管理に開発リソースを割かなければならない状況に置かれています。望むと望まざるとに関わらず、システム設計者にとって消費電力の問題は避けて通れなくなっているのが実情です。

効果的な電力削減を模索する

130nm プロセス ノード以降 IC 内部のトランジスタは、システムがスタンバイまたはスリープ モード時でも電流を引き込みむようになりました。スタティック消費電力あるいはスタティック リーク電流と呼ばれるこのような意図しない電流の引き込みは、90nm、65nm、45nm と、微細プロセスが導入されるにつれてますます大きくなっています。45nm ノードで作られた平均的な動作をするチップの場合、全消費電力のうちワーストケースでは 30% から 60% がスタティック消費電力として失われていきます。それ以外に、デバイスの通常動作でダイナミック消費電力も生じます。また、チップの性能をさらに高めようとすると、リークがより大きくなる高性能トランジスタが必要になってきます。

無駄な電力消費にはデメリットしかありませんが、なかでもスタティック消費電力はきわめて深刻な結果をもたらします。すなわち、スタティック消費電力により熱が発生し、その熱はダイナミック消費電力で発生した熱に加算され、トランジスタは高温となってリーク電流が増え、結果的に発熱量はさらに増加します。つまり、温度の上昇によってリーク電流が増え、さらなる温度上昇を招くという悪循環に陥るのです。冷却や電力バジェットを適切に管理しなければ、リーク電流による発熱増大と発熱によるリーク電流増大の悪循環によって、IC の寿命が短くなったり、致命的なシステム障害に至る熱暴走が発生してしまいます。
Microsoft 社の初期の Xbox 360 に搭載された NVIDIA 社製 ASIC で熱暴走が起きた原因でもあり、再設計と大規模なリコールが必要になったことはよく知られています。

スタティック消費電力に起因する影響を軽減するために、多くの設計者がさまざまな対策に取り組んできました (EDN 誌の記事「Taking a Bite out of Power」参照。
http://www.edn.com/article/460106-taking_a_bite_out_of_power_techniques_for_low_power_asic_design.php)。たとえば、クロック ゲーティングやパワー ゲーティングの実装に加え、回路の中にパワー アイランドを設けるなどの方法が考案されてきました。また、チップ外部にヒート シンクやファンを追加したり、さらには大容量電源と冷却装置とを組み合わせるなどして、リーク電流の問題に対処してきました。しかしこれらの方法はいずれも部品点数の増加と設計プロジェクトの工数増大を招いてしまいます。

半導体業界全体がリーク電流を懸念する一方で、一部のシステム メーカーは別の動機によって低消費電力を追求しようとしています。最近では多くの企業が「グリーン化」の流行に乗っているほか、単純に競合製品と比較して消費電力を削減し、電力料金を含めた総所有コストすなわち運用コストを下げて、自社製品の優位性を高めようと考えているからです。このような傾向は、大規模かつ重要なシステムの一部を構成し、かつ 24 時間 365 日の稼動が求められるネットワーク機器や高性能コンピューターで顕著です。こうした機器が設置されるデータ センターの電力料金や空調費用は莫大な額にのぼり、チップあたりで数ワットを節減するだけでも大きな効果が得られます。また、バッテリで動作するすべての機器では、バッテリの充電または交換を必要とするまでの動作時間に影響が及ぶため、消費電力の削減が最優先課題となっていることは言うまでもありません。

FPGA が一般の携帯電話 (ASIC を設計に使用するのが妥当とされる数少ない市場の一つ) に広く採用されるまでにはまだ時間が必要ですが、FPGA を利用した低消費電力アプリケーションは急速に増加しています。一例としては、オートモーティブ インフォテイメントや運転支援システム、軍用の携帯型通信装置、小型あるいは可搬型の医療機器、3D テレビやビデオ カメラ、航空機や宇宙探査機などが挙げられます。

FPGA 専用の HPL プロセスを開発

ザイリンクスは、2010 年に発表した 7 シリーズ FPGA (Xcell Journal 71 & 72 合併号の「最新技術に新たな定義を加えるザイリンクスの 7 シリーズ FPGA」参照 : http://japan.xilinx.com/publications/archives/xcell/issue71-72/xcell71_72.pdf) の開発にあたって複数の 28nm ファウンドリ プロセスの評価を行い、最終的に、FPGA に特化したプロセスを TSMC 社と共同で開発することを決定しました。「High Performance, Low power (HPL)」と呼ぶ新しいプロセスには、トランジスタのリーク電流が少なく、かつ消費電力とパフォーマンスのバランスに優れた High-k メタル ゲート (HKMG) テクノロジを採用しています。「HPL が登場するまで、ザイリンクスをはじめとする FPGA ベンダーは、ファウンドリから提供される低消費電力 (LP) プロセスかハイパフォーマンス (HP) プロセスのいずれかを選択するしかありませんでした」と、ザイリンクスでプロダクト マネージメント ディレクターを務める Dave Myron は振り返ります。なお、LP プロセスの主な対象はパフォーマンスの低いモバイル アプリケーションであり、HP プロセスは大規模なグラフィクス チップや MPU を主な対象として開発されています。

「いずれも FPGA にとって最適なプロセスとはいえません。LP プロセスを採用すればパフォーマンスが犠牲になり、HP プロセスを採用すれば消費電力が目標値を超えてしまうからです。プロセスを調整する余地はわずかに残されてはいたものの、当社が望むレベルではありませんでした」(Myron)。

「FPGA は幅広いアプリケーションに採用され始めてはいますが、グラフィクス チップのようなスピードや、商用携帯電話機で用いられる ASIC のような超低消費電力が求められているわけではありません」と Myron は続けます。ザイリンクスと TSMC 社は、FPGA 専用のプロセスを共同開発するにあたって、トランジスタのスピードとリーク電流とのバランスが最適となるように努めました。「FPGA アプリケーションが求める消費電力とパフォーマンスのスイート スポットにはまるようにプロセスを特化したのが HPL です（図 1 参照）。したがって、この HPL プロセスを採用した 7 シリーズ FPGA は消費電力とパフォーマンスのバランスに優れているため、これらのどちらかを妥協する必要はありません」(Myron)。

HPL プロセスの大きな特長の一つは、28nm HP プロセスに比べて電圧マージンがあるところです。デバイスの Vcc を広い電圧範囲の中から選択できるため、28nm HP プロセスでは得られない電力とパフォーマンスの高い自由度が実現します。図 2 から判るように、28nm HPL プロセスは 28nm HP プロセスと比較してより高い電力性能比を示し、FPGA がターゲットとする性能範囲において、高性能モード (Vcc=1V) でスタティック消費電力が半分になっています。低消費電力モード (Vcc=0.9V) では、28nm LP プロセスと比べてスタティック消費電力が 70% も少なくなっています。

HPL プロセスは電圧マージンが大きいため、Vcc=0.9V においても優れた性能を発揮するチップを数多く選別できます。なお電圧を 0.9V に下げるとダイナミック消費電力はおよそ 20% 低下します。

7 シリーズ FPGA には Voltage ID (VID) と呼ばれるモードも搭載されています。このモードでは Vcc を制御することで消費電力の削減やパフォーマンスの向上が図れます。すべての 7 シリーズデバイスには VID が格納されています。この情報は読み出し可能で、性能仕様を満たしながらデバイスが動作できる最低電圧を示します。「VID モードでは、7 シリーズ FPGA の設計マージンを生かした選択肢がユーザーに提供されます。既存の回路を 7 シリーズ デバイスに実装する場合、回路はそのままで消費電力を実質的に半分に削減する方法もあれば、もともとの電力バジェットは減らさずにより多くの機能を追加する方法もあります。システム全体の消費電力の削減と基板の省スペース化を図りながら、性能の向上とシステム コストの大幅な削減が可能というわけです」と Myron は述べます。

ザイリンクスでは FPGA に最適化した HPL プロセスを、7 シリーズを構成する 3 種類のファミリすべてに適用すると共に、Zynq™-7000 エクステンシブル プロセッシング プラットフォーム (EPP) にも採用しています。これらすべてのデバイスの FPGA ファブリックは、小型かつ電力効率の高いブロックを組み合わせた、統一された ASMBL™ アーキテクチャで構成されています。これにより、7 シリーズの全デバイス（低コストかつ最も消費電力の小さい Artix™-7 FPGA、コスト パフォーマンスに優れた Kintex™-7 FPGA、最高性能と最大容量の Virtex®-7 ファミリ、および ARM® デュアルコア Cortex™-A9 MPCore™ プロセッシング システムを内蔵し、エンベデッド アプリケーションを対象とした Zynq™-7000 EPP）間で、デザインを簡単に移行できます（本合併号の「イノベーションの新時代を拓く Zynq-7000 EPP」参照）。

競合ベンダーは未だ、1 つのアーキテクチャから派生した複数のアーキテクチャを HP プロセスまたは LP プロセス上に実装し続けています。その一方でザイリンクスは、FPGA 専用プロセス上に統一されたシリコン アーキテクチャを実装するという戦略が、システム ソリューションそのものでなく、基盤の役割を果たすプログラマブル プラットフォームとしての FPGA テクノロジの成熟を牽引していくものと考えています。なお、Virtex-6 および Spartan®-6 FPGA 世代から導入されたザイリンクスのプログラマブル プラットフォーム戦略 (Xcell Journal 67 & 68 合併号の「ターゲット デザイン プラットフォームで技術革新を加速」参照 : http://japan.xilinx.com/publications/archives/xcell/xcell67_68.pdf) においては、最先端のシリコンだけではなく、マーケットあるいはアプリケーションに特化した開発ボード、IP、ツール、およびドキュメントが提供されており、新製品の開発期間短縮を支援しています。

「最適化されたプロセスと統一されたアーキテクチャを採用した 7 シリーズ FPGA の開発方針は、5 年以上前にインテル社が先陣を切って実装に成功した戦略 (ホワイト ペーパー「Inside Intel Core

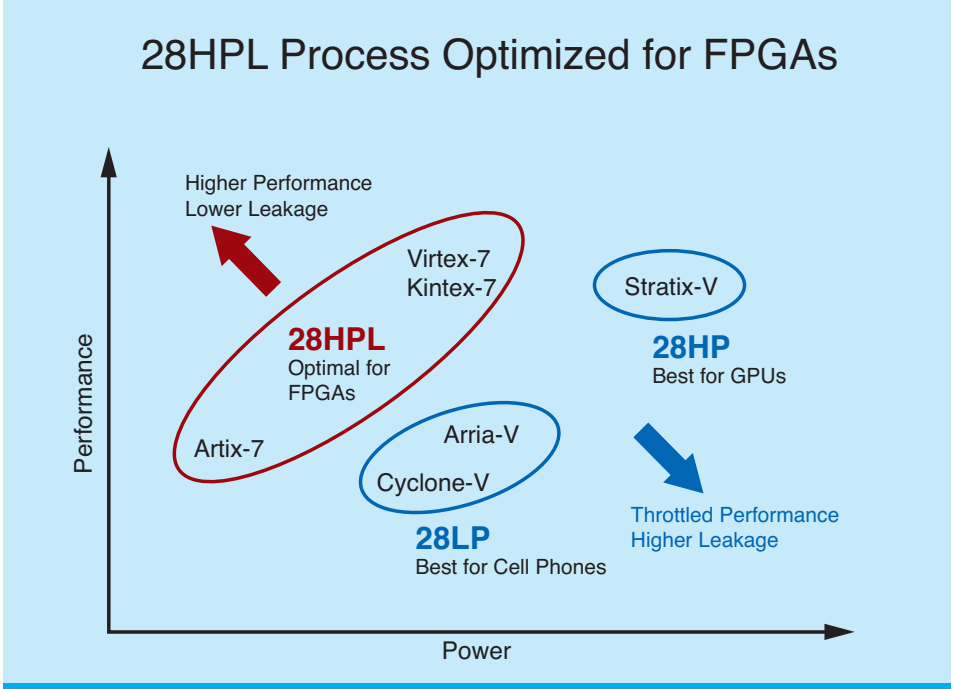


図 1. FPGA アプリケーションの消費電力とパフォーマンス要件のスイート スポットに位置する HPL プロセス

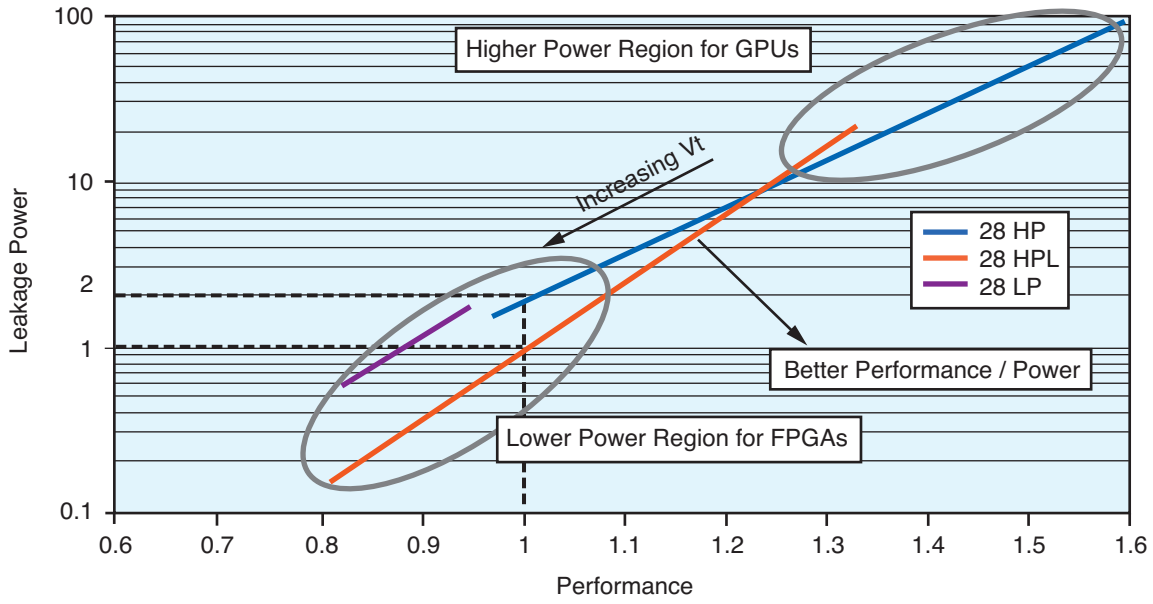


図 2. 28nm HPL、28nm HP、および 28nm LP 各プロセスにおけるパフォーマンスとリーク電流の関係

Microarchitecture」参照 : http://software.intel.com/file/18374/) を FPGA に適用したものであり、半導体業界においては先進的な初の試みというわけではありません」(Myron)。

「インテル社は 2006 年に、複数のマイクロプロセッサ アーキテクチャの使用をやめ、INTEL CORE マイクロ アーキテクチャと呼ぶ唯一かつ強力なハードウェアを 1 つのシリコン プロセス上に実装し、サーバー向けハイエンド プロセッサからモバイル ノート向けプロセッサまでの複数の製品ラインに適用する戦略へと転換しました」と、かつてインテル社に勤めていた Myron は説明します。「開発リソースを無制限に使用できると思われるインテル社が、ハードウェア アーキテクチャをたった 1 つに絞り込んだ理由は何でしょうか。その答えは、さまざまなニーズに対応できる 1 つのアーキテクチャに開発リソースを集中させたいという狙いと、アーキテクチャを統一したほうが複数のアプリケーションを横展開するときに有利になると考えたからです」。

「同様のことがザイリンクスにも言えます。7 シリーズ FPGA に同じアーキテクチャを採用したことで、ソフトウェア開発チームのリソースを複数の製品ファミリに分散させることなく品質の向上に集中させることができるようになりました。また、IP の再利用を効率化したいと常に望んでいたユーザーにとっては、異なるアーキテクチャをまたがる場合と比較して、設計工数の削減と効率的な IP の再利用が図れるというメリットがもたらされます」。

High-k メタルゲートを採用した HPL プロセスの開発は、FPGA のスタティック消費電力を減らす第一段階としての取り組みです。次の段階として、7 シリーズ デバイスのアーキテクチャの刷新を進めました。たとえば、以前のザイリンクス FPGA には、未使用のトランシーバー、PLL、デジタル クロック マネージャー、および I/O の各電源を遮断できるようにパワー ゲーティング機能が搭載されていましたが、7 シリーズでは未使用のブロック RAM に対しても同様の制御が可能です。ブロック RAM はデバイス全体のリーク電流の 30% を占めることもあるため、パワー ゲーティングは大きな効果をもたらします。

システム全体の消費電力削減

「High-k メタルゲートを採用した HPL シリコン プロセスの開発を通じてスタティックおよびダイナミック消費電力を大幅に減らしたのち、7 シリーズ デバイスのシステム全体の消費電力を削減するために、次の段階に取り組みました」(Myron)。ここでシステム全体の消費電力とは、スタティック リーク電流、ダイナミック消費電力、I/O 電力およびトランシーバー電力で構成されます (図 3)。

ダイナミック消費電力の削減

ザイリンクス社内で消費電力に関するエキスパートとして知られ、ザイリンクス FPGA の消費電力削減に貢献してきた優秀なエンジニアである Matt Klein によれば、FPGA ロジックのダイナミック消費電力は「容量 × 電圧の二乗 × 周波数」という一般的な数式で得ることができるとされています。

ダイナミック消費電力 = $\mu \times f_{clk} \times C_L \times V_{DD}^2$

数式内の容量 C に関しては、ダイナミック消費電力を抑える目的で、FPGA 内のほとんどのブロックを対象にデザインの最適化を適用し、容量の大幅な削減を実施済みです。さらに一部のブロックについては、より小型かつ低容量になるようにアーキテクチャの再設計を行っています。「DSP48 を含む一部のブロックのダイナミック消費電力

は、公称電圧が 1V の場合でも、0.85V を公称電圧とする他社の 28nm FPGA よりも下回っています。この消費電力をさらに削減するために、ザイリンクスでは電圧スケーリング オプションも提供しています」(Klein)。なお Klein は、周波数 fclk に比例してダイナミック消費電力は影響を受けると述べています。

ダイナミック消費電力をさらに減らすには、高度なクロック ゲーティング制御を行って稼働率「α」を変える方法もあります。制御回路を追加して、特定のブロックの稼動率を制御するという方法です。ただし、FPGA の設計規模が大きくなるほど開発に時間を要するため、実際に取り入れているユーザーはあまり多くありません。

また、別の削減方法もあります。すべての 7 シリーズ FPGA は階層化クロックを採用しているため、必要なクロック リソースのみが有効となるように回路をプログラムできます。この方法を用いるとクロック回路で消費される電力が大幅に低下します。また、グローバル クロック ゲーティングとリージョナル クロック ゲーティングのほかに、フリップフロップなどのローカル リソースに対してはクロック イネーブル (CE) を介したクロック ゲーティングが使用でき、3 段階でクロックのゲーティング制御が可能です。

「ザイリンクスの FPGA には基本的に 1 つのスライスに 8 個のフリップフロップが存在します。それらのフリップフロップは 1 つの CE 信号を共有していますが、過去のアーキテクチャとは異なり、クロックは CE によってローカルにゲートされ、フリップフロップのトグル動作が停止します。このようなハードウェアを利用し、ISE® デザインツールは出力が下流で使用されていないフリップフロップを探し出し、フリップフロップの不必要なスイッチングを自動的に抑制します。この処理はロジック検証と合成の完了後に行われます。続いて ISE はローカル クロック イネーブルを生成します。これらの機能をマップで有効にするには -power high または -power XE オプションを使用します」と、さらに Klein は続けます。

「この自動かつ高度なクロック ゲーティングを活用すると、ロジック部分のダイナミック消費電力を最大で 30%、平均では 18% 削減できます。このクロック ゲーティングの実装に必要な追加ロジック量は 1% にも満たないため、ダイナミック消費電力の削減に大きく貢献します」(Klein)。

高度なクロック ゲーティングはブロック RAM に対しても有効です。多くの場合、設計者だけでなく合成ツールにおいてもブロック RAM のイネーブル信号は 1 に固定した状態で使用されています。Klein はブロック RAM のアドレス入力とデータ出力の検討を勧めています。データ出力は下流側で使用される前に sel と呼ばれるマルチプレクサ制御信号を介して選択されることがあります。まず、書き込みの対象となっていないブロック RAM や、最後の読み出しサイクルから読み出しアドレスが変更されていないブロック RAM を有効にする必要はありません。次に、ある時点でシステムがブロック RAM 出力を使用していない場合は、読み出し動作に備えてブロック RAM を有効にしておく必要はありません。

ISE は、フリップフロップの CE 信号生成と同様の方法を用いて、ブロック RAM の CE 信号をサイクルごとに自動的に生成します。「ブロック RAM は節電効果がきわめて大きく、わずかなロジックの追加だけで、最大で 70%、平均でも 30% の消費電力を削減できることが判りました。また、電力効率の高いブロック RAM アレイを構成できるように、CORE Generator™ および XST にオプションを設けています。アレイ内のブロック RAM のダイナミック消費電力

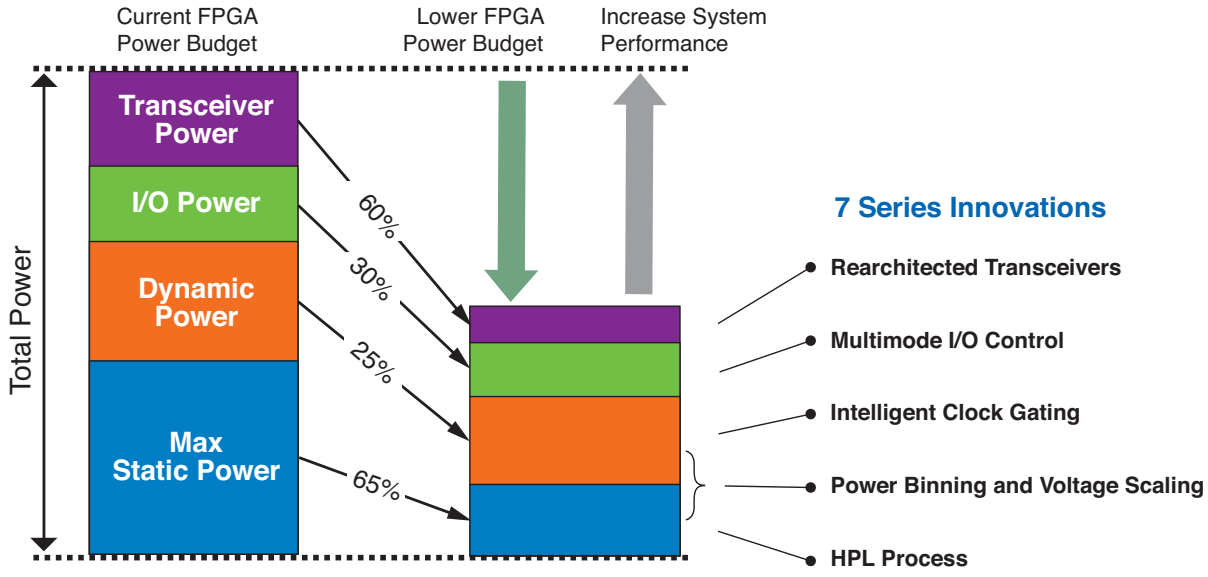


図 3. 従来世代に比べて 50% 以上の消費電力を削減するために、
7 シリーズの消費電力をあらゆる観点から最適化

を最大で 75% も節減できます」(Klein)。

I/O 消費電力の削減

消費電力について考慮しなければならないのはスタティック消費電力とダイナミック消費電力だけではありません。I/O とトランシーバーが消費する電力も FPGA の全消費電力の一部を占めています。

高速 I/O が消費する電力を抑えるために、マルチモード I/O 制御を 7 シリーズ FPGA に追加し、トランシーバーのアーキテクチャを再設計しています。マルチモード I/O 制御によって、メモリ インターフェイスでは特に、メモリへの書き込み動作時に最大 50%、メモリ アイドル状態では最大 75% もの大幅な省電力が可能です。

DDR2 や DDR3 などの外部メモリへの書き込み中に、I/O ハードウェアによって IBUF (入力バッファー) を自動的に無効にする機能が新しく追加されました。これにより、メモリへの書き込み動作時に省電力効果が得られます。「入力バッファーは電圧参照型レシーバーなので、トグル レートとは関係なく DC 電力を消費します。メモリへの書き込み動作時にこの DC 電源を遮断することで、書き込みの多さに比例して削減効果が得られるというわけです。メモリへの書き込み動作時に終端のみを無効にする方法と比較すると、全消費電力の 50% がさらに節減されます」(Klein)。

これとは別に、メモリ バスがアイドル状態のときに IBUF と終端を無効にする省電力機能も追加され、すべての 7 シリーズ FPGA の I/O に搭載されています。「バスアイドル動作中はバスをオフにするのが一般的ですが、アイドル動作はメモリからの読み出し動作に似ているため、この機能がなければ終端と IBUF の両方が電力を消費してしまいます。終端と IBUF を無効にすることで、それらをオンにしたままの状態と比較して、7 シリーズ I/O の消費電力は 75% も少なくなります」(Klein)。

また、7 シリーズでは、V_{CCAUX} 電圧を 2.5V から 1.8V に引き下げています。その結果、PLL、IDELAY、入力および出力バッファー、コンフィギュレーション ロジックなど、V_{CCAUX} で動作するすべてのブロックが消費する電力がおよそ 30% 下がりました。

以上の新たな省電力機能は、Virtex-6 をはじめとするほかの FPGA の高速メモリ インターフェイスと比較した場合、一歩先を進んだ優位性を発揮します。

トランシーバー消費電力の削減

デバイスの全消費電力を左右するもう一つの要因がトランシーバーによって消費される電力です。XPower Estimator (XPE) ツールで 7 シリーズの電力を見積もった際、当初はトランシーバー電力をやや少なく算出していました。その後、GTP および GTH トランシーバーで消費される電力を再検討し、現在はツールの計算値がシリコンの実測と一致するように変更しました。XPE の最新リリース (バージョン 13.2) には、より正確な値が得られるように、こうした変更が反映されています。

前出の Myron は次のように述べています。「6.75Gbps の性能を持つ Artix-7 の GTP と同じ性能条件で Spartan-6 の GTP を比べたとき、トランシーバー全体の消費電力は 60% 以上も低くなっています。絶対的な低消費電力と低コストを求めるローエンド マーケットのニーズに応えるために、このような取り組みを行いました。また、Virtex-7 の GTH の消費電力も大幅に削減されています」。最大で 96 個のトランシーバーを搭載する Virtex-7 は、トランシーバー電力が全消費電力の大きな割合を占める広帯域アプリケーションに利用されます。「競合の 28nm FPGA のトランシーバー電力と同等のレベルです」(Myron)。

パワー ビニングと電圧スケーリング

7 シリーズ デバイスの画期的な省電力技術として、標準品と同じ性能を維持しながらさらなる省電力が得られるパワー ビニング (消費電力による選別) と電圧スケーリングが挙げられます。Myron は次のように説明します。「これらは 28nm HPL プロセスの電圧マージンが増加したことで可能となったオプションであり、通常の 28nm プロセスを採用するほかのベンダーでは実現できません。では、ザイリンクスはどのようにしてこれを可能にしたのでしょうか。1V で動作する標準スピード グレード デバイス (スタティック消費電力およびダイナミック消費電力ともに公称値) のスピード分布は図 4 のようになります。ここから、ス

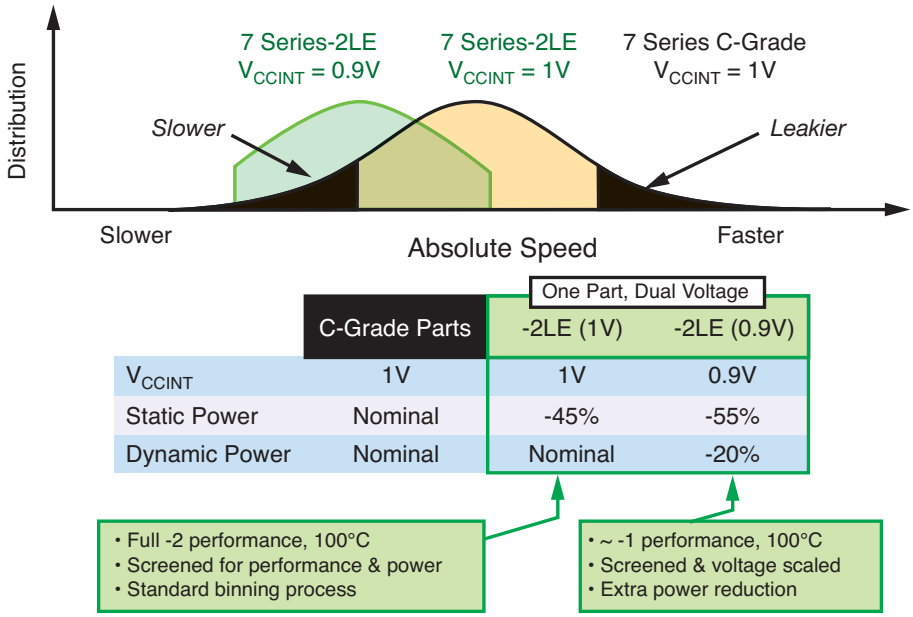


図 4. パワー ビニングと電圧スケーリングを実現する 28-nm HPL プロセスの電圧マージン

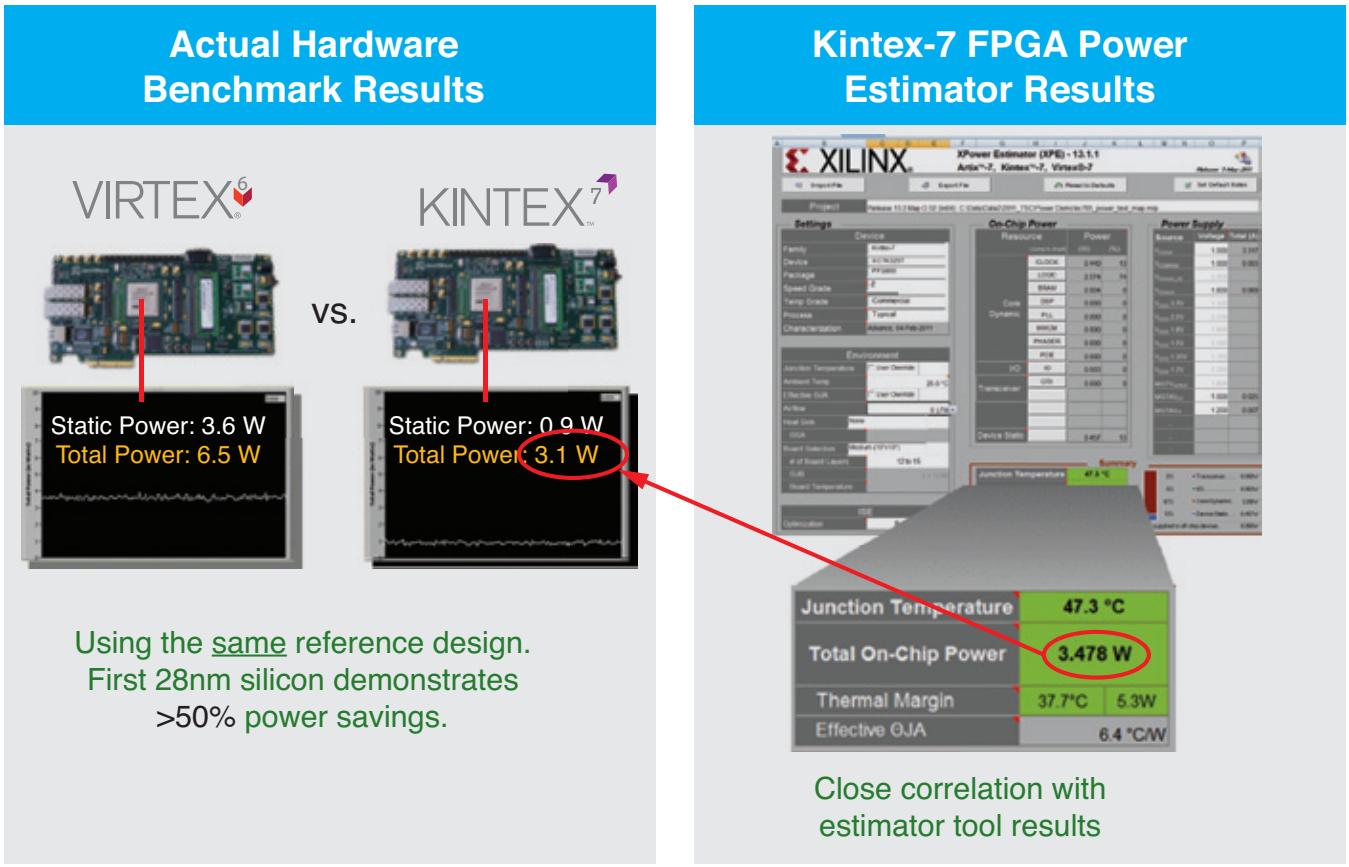


図 5. ザイリンクス FPGA の消費電力プロファイルの評価や比較が可能な
XPower Estimator (XPE) ツール

ピードが充分ではないものとリーク電流が大きいものを選別して取り除くと -2L グレード デバイスが得られます。-2L グレードは同じくコア電圧 1V で動作し、ゆえに性能は民生用 C グレードまたは産業用 I グレードと同じ -2 グレードに相当します。動作温度範囲は最高 100℃ です。-2LE グレード デバイスは、C グレードに比べて 45% も消費電力が少なく、かつ標準デバイスのビニングの過程で得られるため供給も特に問題はありません。次に -2LE グレードから 0.9V で動作するものを選別します。コア電圧を 0.9V に下げることで、標準的な C グレードと比較してスタティック消費電力では最大で 55%、ダイナミック消費電力では 20% もの電力削減を可能としています」。

ベンチマーク評価

ザイリンクスの 28nm ノードの戦略に対して競合ベンダーは、「一つのサイズの服であらゆる体型をカバーしようとしている」などと主張していますが、FPGA を発明したザイリンクスは、7 シリーズ デバイスが画期的な技術開発の証明となると自負しています。ザイリンクスは、FPGA の採用が検討されているさまざまなアプリケーションを対象に、7 シリーズがそれぞれのアプリケーション要件に応じて最適なバランスを備えたデバイスであることを総合的なベンチマークで示してきました。各分野を対象としたベンチマーク評価は、http://www.xilinx.com/publications/technology/power-advantage/7-series-power-benchmark-summary.pdf で公開しているほか、http://seminar2.techonline.com/registration/wcIndex.cgi?sessionID=xilinx_jun1411 にて TechOnline ウェブ セミナを配信しています。

消費電力のさらなる改善に向けて

最新の ISE 13.2 リリースで利用できる消費電力見積もりツール XPE（図 5）では、7 シリーズ デバイスの最新のアップデート データが提供されます。また、アーキテクチャが再設計された GTP トランシーバーおよび GTH トランシーバーの省電力データも反映されています。さらに、お客様から寄せられる多くのご要望にお応えして、電源設計や熱設計でワーストケースを検討する際に必要となる最大消費電力のデータも提供します。

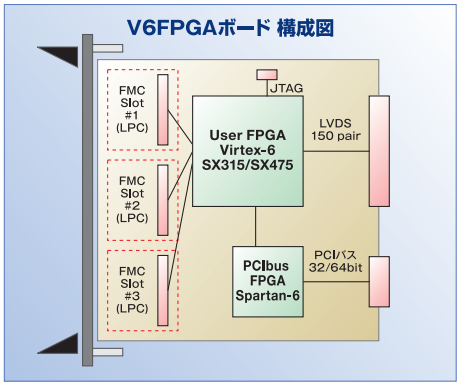
7 シリーズ デバイスの電力管理とベンチマーク情報の詳細は、7 シリーズのホワイトペーパー『28 nm プロセスを採用した 7 シリーズ FPGA で消費電力を削減』(WP389 : http://japan.xilinx.com/support/documentation/white_papers/j_wp389_Lowering_Power_at_28nm.pdf) を参照してください。

また、7 シリーズの消費電力に関する詳細は、http://japan.xilinx.com/products/technology/power/index.htm を参照してください。

FPGAボード
DASH-BBシリーズ
型名：V6FPGA

PCDAQ：Data Acquisition System
iDAQS

Virtex-6搭載 コンパクトPCI FPGAボード
大規模な無線信号処理を複数ボードでリアルタイム処理



仕様

FPGA	Xilinx, Virtex-6 SX315T / SX475T
FMCコネクタ	3スロット
クロック	内部・外部
低ジッタクロック	225 Fs rms
バックプレーン接続	300ピン (LVDS 150 pair)
ボード規格	コンパクトPCI 6U
サイズ	233 x 160 [mm]

FMCモジュール

※アイダックス製FMCモジュール
（V6FPGAボードに搭載して使用可能）

型名：ADC16F
アナログ-デジタル変換
4ch, 16bit, 160MHz

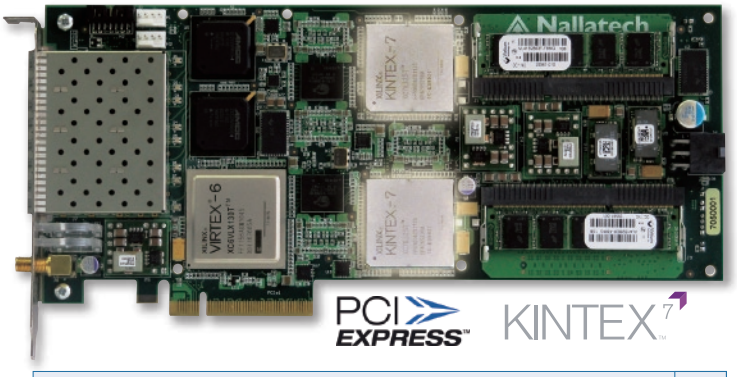
型名：DAC16F
デジタル-アナログ変換
4ch, 16bit, 500MHz

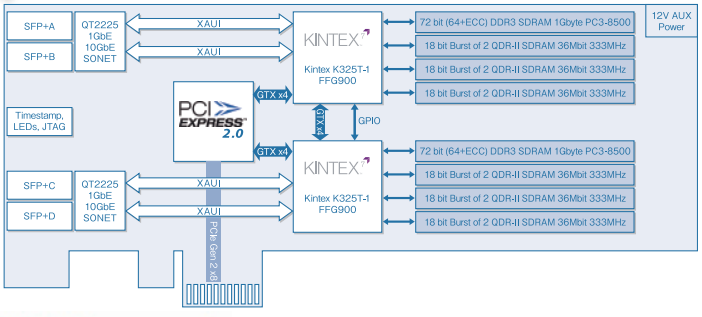
特徴

- **大規模ロジックに対応**
  - >Virtex-6 FPGAを搭載
  - >SX315T/SX475Tから選択
  - >FFG1759パッケージのLXTも対応
- **高速AD&DA向き　FMCコネクタ**
  - >FMCを3スロット用意
    例）4ch DAC&ADC
  - >LPC(Low Pin Count)のピン数対応
  - >MGT高速シルアル：8ペア/各スロット
- **高精度クロック生成**
  - >クロックチップ：AD9516
  - >各FMC、バックプレーンに供給
  - >内部(10MHz)・外部を切替
  - >低ジッタ： 225 Fs rms
  - >PLLによる周波数設定
- **バックプレーン接続**
  - >LVDS150pairを入出力
    ボード間を高速接続し、
    複数FPGAの同期を実現
- **コンパクトPCI**
  - >Spartan-6がPCIバスをインターフェース
  - >クロック設定やコンフィグを実行

PCIe-287N
Quad SFP+, Kintex-7 and Memory Card

Kintex-7搭載 10GbE ネットワーク・ボード
サイバー攻撃、暗号化、フィルタなどの研究向け





Nallatech
a subsidiary of Interconnect Systems Inc.

仕様

FPGA	Kintex-7 K325T 2個搭載
10GbE	SFP+ 4ポート
PCIe転送	RD&WR 2.5GB/s
形状	PCI-Express フル・ハイト フル・レングス
SRAM	4.5MB QDR-II 6個独立
SDRAM	1GB DDR3 2個独立
サポートOS	Linux 64bit

特徴

- **10GbEの高速データをFPGA処理**
  - >フル・ライン・レートでの処理
  - >ゼロ・パケット・ロス
  - >トラフィック・キャプチャー
  - >リアルタイム暗号化
  - >圧縮
  - >カスタム・プロトコル
- **評価済みIPが開発期間を短縮**

株式会社アイダックス
〒167-0042 東京都杉並区西荻北 3-13-5 M&Iビル 2F TEL 03-6762-1210 FAX 03-6762-1211
www.idaqs.jp

Kintex™-7 FPGA ACDC
（Acquisition, Contribution, Distribution and Consumption）1.0 ベースボード

XILINX KINTEX 7™

inrevium
A Revolutionary Innovation Platform

TB-7K-325T-IMG

ビデオデザインの開発に最適な画期的プラットフォーム

4K2K TV、プロジェクター/モニタ、4K2K ビデオカメラ、ブロードキャストスイッチャー/ルーター、アップ/ダウン変換器、SMPTE 2022/EthernetAVB のvideo over IP システムなど、ビデオアプリケーションで求められる高バンド幅で高性能なビデオバッファリング要件に対応しビデオデータの入出力にはHDMI、DVI、DisplayPort、V-by-OneHS、Ethernet、LVDS、PCIe、などのさまざまなインターフェイスの利用が可能

- FPGA：Kintex-7 FPGA XC7K325T-FFG900
- Memory：DDR3 2Gbit x 4
- FMC connector
  - HPC（High Pin Count）x 2
  - LPC（Low Pin Count）x 2
- SERDES：12.5Gbps（GTX）×8ch×2HPC

東京エレクトロン デバイス株式会社
PLD事業部
〒221-0056 神奈川県横浜市神奈川区金港町1番地4 横浜イーストスクエア
TEL：045-443-4016 FAX：045-443-4058
http://ppg.teldevice.co.jp/

XILINX ALLIANCE PROGRAM PREMIER MEMBER

東京エレクトロンデバイスでは自社の持つ情報・技術・サービスを「inrevium（インレビアム）」として商品化しております

Reconfigurable System Uses Large FPGA Computation Fields

複数の FPGA でコンピューティング フィールドを構築した リコンフィギャラブル システム

Igor A. Kalyaev
Director
Kalyaev Scientific Research Institute
of Multiprocessor Computer Systems
Southern Federal University
kaliaev@mvs.sfedu.ru

Ilya I. Levin
Deputy Director of Science
Kalyaev Scientific Research Institute
of Multiprocessor Computer Systems
Southern Federal University
levin@superevm.ru

Evgeniy A. Semernikov
Head of Laboratory
Southern Scientific Centre of the
Russian Academy of Sciences
semernikov@mvs.tsure.ru

数多くのザイリンクス FPGA を相互接続することで、幅広い応用分野のジョブに対応する新しいタイプのスーパーコンピューターを可能に

過去 20 年ほどの間、多くのスーパーコンピューターにはシステムの中心的頭脳としてのマイクロプロセッサと、CPU のコンピューティング タスクを一部負担する FPGA を組み合わせたアーキテクチャが採用されていました。実際、このように FPGA とスタンドアロンのマイクロプロセッサを組み合わせる方法は何世代にもわたってスーパーコンピューターの主流となり、多くの実績を残してきました。

しかし先ごろ、筆者らが所属する南部連邦大学 (ロシア、タガンログ) の Kalyaev Scientific Research Institute of Multiprocessor Computer Systems の科学者グループは、ほぼ全面的にザイリンクス FPGA をベースにしたリコンフィギャラブル コンピューター システム (RCS) を設計しました。複数の FPGA を相互接続することで、幅広い応用分野の問題解決に最適化した専用のコンピューティング構造を構築できます。

FPGA ベースの RCS には、いくつかの設計ポイントを配慮する必要がありますが、ほかでは得られない独自の優位性があります。RCS の心臓部には、多数の FPGA を連結して 1 つのコンピューティング システムを形成しました。この RCS では直交性の高い通信システムを使用し、FPGA を長方形の格子状に配置して隣接する FPGA をダイレクト リンクで相互接続しています。

また、CAD (Computer-Aided Design) ツールおよび構造プログラミング ツールを用いて FPGA のコンピューティング フィールドに並列パイプラインのコンピューティング構造を構築しました。このコンピューティング構造のアーキテクチャはタスクの構造に似せて設計されており、この類似性によって RCS の性能を最大限に引き出すことに成功しています。

ハードウェアの制約により、実行したいタスクまたは解きたい問題の情報グラフ全体をマッピングできない場合は、システムによって自動的にグラフが数多くのサブグラフに分割されます。システムのコンピューティング フィールドはこれらの各サブグラフを個別のコンピューティング構造として認識し、逐次的に並べて順に解いていきます。

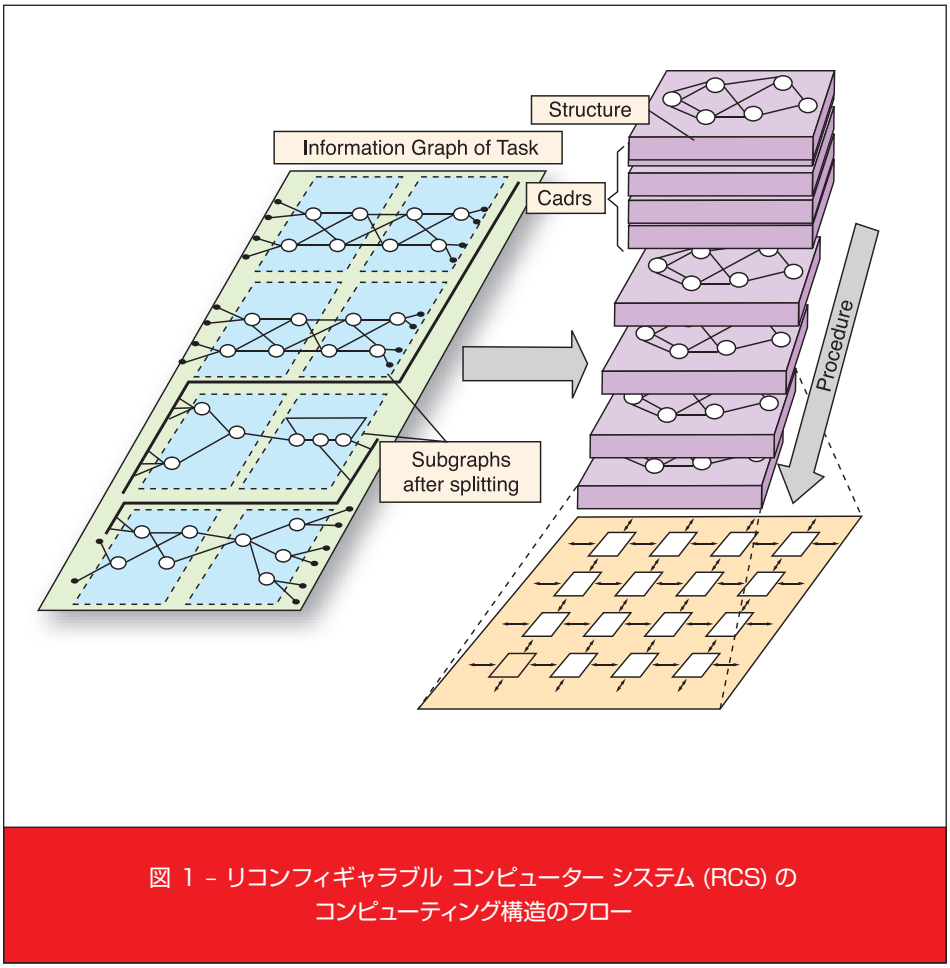


図 1 – リコンフィギャラブル コンピューター システム (RCS) の
コンピューティング構造のフロー

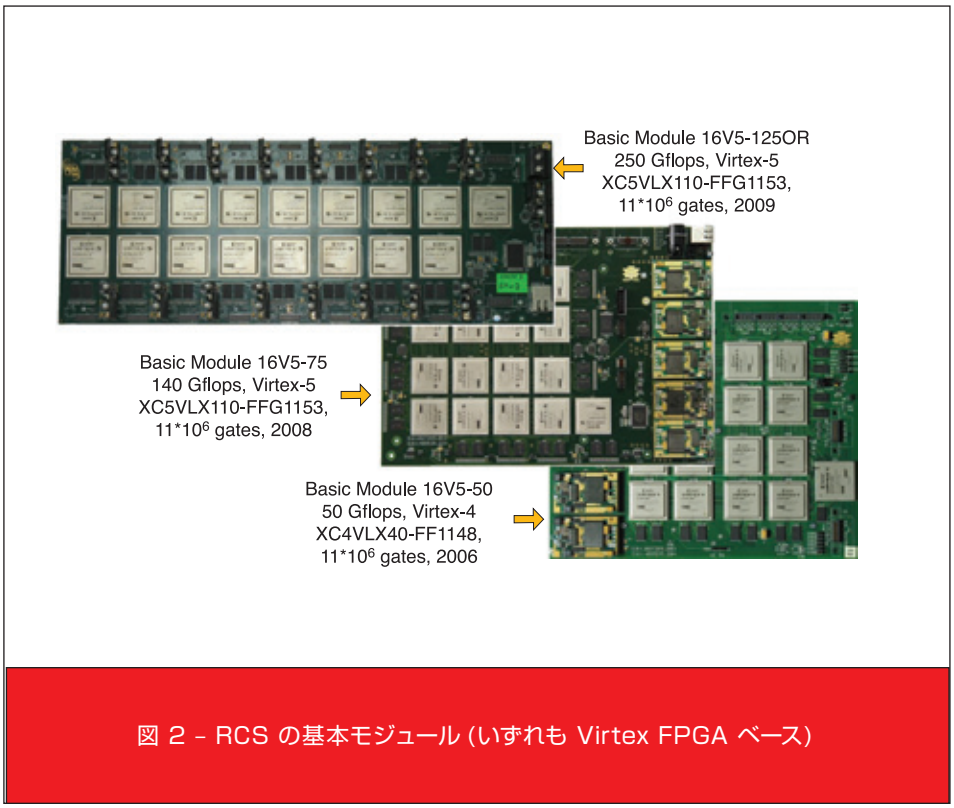


図 2 – RCS の基本モジュール (いずれも Virtex FPGA ベース)

図 1 は、RCS のコンピューティング構造のフロー、およびデータ入出力のプロシージャを示したものです。これを見るとわかるように、各サブグラフのサイズは FPGA フィールドのサイズに比例しています。筆者らが構造プログラミング用に開発したツールは、1 つの問題を少数の大きめなサブグラフに自動的に分割し、これらのサブグラフをハードウェアにマッピングすることで効率よく問題を解き、RCS のパフォーマンスを飛躍的に引き上げるように設計されています。

もう少し詳しく説明すると、このシステムは与えられたタスクまたは問題の情報グラフを、同一構造のサブグラフのシーケンスとして解釈します。これらのサブグラフは、互いに独立している場合と依存関係にある (情報が関連している) 場合があります。各情報サブグラフは、システムによって cadr と呼ばれる特別なコンピューティング構造に変換されます。cadr とはタスクのサブグラフをハードワイヤードで接続してそこでオペランドのフローを実行したものです。オペランド (または結果) の各グループが、シーケンス内の特定のサブグラフの入力 (または出力) ノードに対応します。cadr のシーケンスは、特別な並列プログラム (プロシージャと呼ぶ) で実行します。RCS 全体で必要となる並列プログラムは 1 つのみです。

筆者らは、HPC 用の RCS をモジュラー型アーキテクチャで設計しました。RCS は複数の基本モジュールで構成され、コンピューティング フィールド全体を各モジュールに分割して実行します。また、各モジュールはほかのモジュールや補助サブシステムと接続されています。図 2 に、南部連邦大学、Kalyaev Scientific Research Institute of Multiprocessor Computer Systems (SRI MCS SFU) で設計したいくつかの基本モジュールを示します。

基本モジュールに搭載されている FPGA は、動作周波数 1.2GHz の LVDS チャネルを使用して通信します。RCS 内のモジュールの通信にはすべてこれらのチャネルが使用されています。各基本モジュールには分散型メモリ ユニットやリソース管理用の基本モジュール コントローラー (分割した並列アプリケーションを分散型メモリ コントローラーにロードしたり、データ入出力を処理する) も含まれます。

RCS ファミリのメイン モジュールとなるのが、16V5-75 基本モジュールです。RCS シリーズは、50 GFLOPS (16 個のザイリンクス Virtex®-5 FPGA を使用) から 6 TFLOPS (1,280 個の Virtex-5 FGPA を使用) までさまざまな性能レベルのシステムで構成されています。ハイエンドのモデルは 5 つの ST-1R ラックで構成され、各ラックには 4 つの 16V5-75 基本モジュールで構成される 6U の RCS-0.2-CB ブロックが 4 つ搭載されています。基本モジュールのピーク性能は、単精度データの場合で最大 200 GFLOPS です。このため、ST-1R ラックのコンピューティング フィールドに 256 個の Virtex-5 FPGA (XC5VLX110) を搭載し、これらを高速 LVDS で接続しています (図 3)。

一方、Orion RCS (図 4) のピーク性能は 20 TFLOPS を達成しています。このシステムは 19 インチ ラックに 1,536 個の Virtex-5 FPGA を搭載しています。このシステムのメイン モジュールとなるのは、ピーク性能 250 GFLOPS の 16V5-1250R 基本モジュールです。これらの基本モジュールを高速 LVDS で接続して 1 つのコンピューティング リソースを形成しています。1U ブロック内に 4 つの 16V5-1250R 基本モジュールが接続されています。16V5-75 基本モジュールでは、使用する 16 個の FPGA のうちほかの基本モジュールと接続できるのは 4 つのみですが、16V5-1250R 基本モジュールの FPGA は 16 個すべてを LVDS で接続してコンピューティング フィールドを拡張できます。これにより、Orion ブロック内部では基本モジュール間で非常に高いデータ転送レートを実現しています。

表 1 は、デジタル信号処理、線形代数、数理物理学などさまざまな問題を解いた場合の RCS-0.2-CB と Orion の実性能を比較したものです。

RCS のプログラミング

RCS のプログラミングは、伝統的なクラスター ベースのスーパーコンピューター アーキテクチャとは大きく異なります。従来のスーパーコンピューター アーキテクチャはハードウェア回路が固定されており、プログラミングといえばソフトウェア エンジニアがソフトウェア レベルで行うものに限定されていました。一方、FPGA ベースの RCS はソフトウェアだけでなくハードウェアもプログラミング可能です。そこで、筆者らは RCS のプログラミングを構造プログラミングと手続きプログラミングの 2 種類に分割しました。構造プログラミングは、情報グラフを RCS 構造のコンピューティング フィールドにマッピングする作業、すなわち FPGA のプログラミングを指します。手続きプログラミングは、ユーザーがシステムで実行する計算手順の構成を記述する伝統的なプログラミングです。一般に、ソフトウェア エンジニアは FPGA のプログラミングが苦手とされているため、完全なプログラミングを行うにはハードウェア設計のスキルが要求されます。この点を考慮して、筆者らは RCS のプログラミングを容易化する特別なソフトウェア スイートを開発しました。このスイートを利用すると、FPGA やハードウェア設計に関する

タスク	単位容積当たり性能 (GFLOPS/dm³)	問題を解いた場合の実性能 (GFLOPS)		
		DSP	線形代数	数理物理学
RCS-0.2-CB	16.7	647.7	423.2	535.2
Orion	64.9	809.6	528.7	669.0

表 1 – RCS-0.2-CB ブロックと Orion ブロックの性能

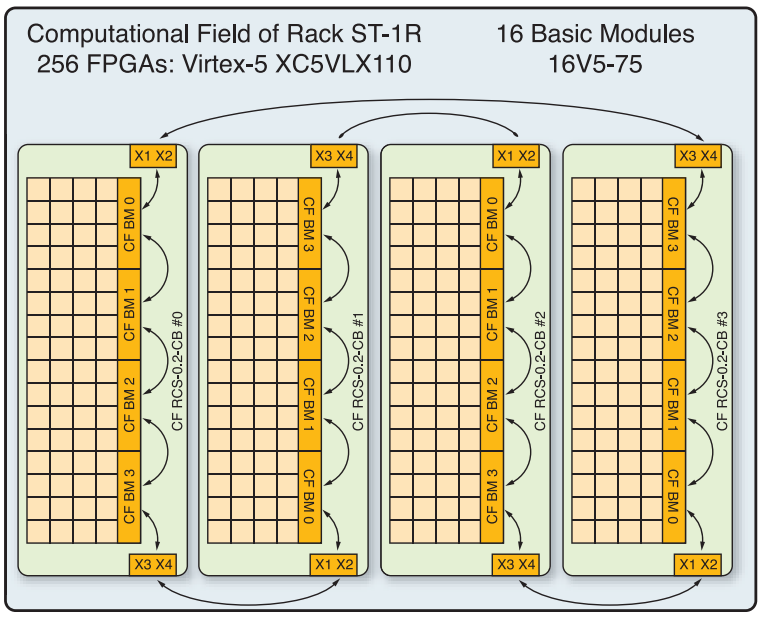


図 3 – 256 個の Virtex-5 FPGA を高速 LVDS で接続した ST-1R ラックのコンピューティング フィールド

専門知識がないユーザーでもシステムのプログラミングが可能です。事実、ソフトウェア開発者がこのスイートを使用してプログラミングを行うと、エンジニアがタスクに合わせて回路を設計し、伝統的な手法やツールを利用した場合にと比較して、1/2 ～ 1/3 の時間でソリューションが完成します。

回路設計用の Fire!Constructor 開発環境をベースにしたこのスイートには、COLAMO と呼ばれる高級プログラミング言語と Argus IDE（統合開発環境）が含まれます。ユーザーが COLAMO でプログラムを作成すると、内蔵のコンパイラ / 変換 / 合成ツールがコンピューティング ユニットおよびインターフェイスの IP ライブラリを使用して構造プログラミングおよび手続きプログラミング用のコードを自動的に作成します。

基本モジュール、ユニット、および RCS 全体の記述（RCS パスポート）がライブラリとして用意されているため、各種アーキテクチャ プラットフォームから RCS へ並列アプリケーションを容易に移植できます。COLAMO 変換ツールは、システム記述を制御、手続き、ストリーム、構造の 4 つのデータ カテゴリに分割します。

制御データは Pascal に変換され、マスター コントローラーで実行されます。マスター コントローラーは、RCS ハードウェア プラットフォームの種類によって基本モジュールに含まれる場合とコンピューティング ブロックに含まれる場合があります。制御データは、対象となるコンピューティング タスクに合わせて FPGA をコンフィギュレーションし、分散型メモリをプログラミングするだけでなく、ブロックと基本モジュール間のデータ交換をマッピングする役割も果たします。

一方、手続きデータとストリーム データは Argus アセンブラ言語に変換されます。これらのデータは分散型メモリ コントローラーによって実行されます。分散型メモリ コントローラーは、cadr の実行内容を指定して、cadr のコンピューティング構造内に並列データ フローを構築します。

このスイートで最も注目すべき機能が Fire!Constructor と呼ばれる合成ツールで、これは RCS の中枢となる FPGA のプログラミングを支援するために開発されました。Fire!Constructor は、IP ライブラリを使用して RCS の各 FPGA の VHDL 記述（*.vhd ファイル）を生成します。これらのファイルを使用して、ザイリンクス ISE® スイートはすべての FPGA のコンフィギュレーション ファイル（*.bit）を生成します。

効果的な問題解決

構造的計算と手続き的計算を組み合わせたことで、RCS はさまざまな問題を効果的に解くことができます。中でも特に適しているのが、アルゴリズムの情報グラフが不変で、入力データ セットが可変な問題です。たとえば薬剤設計、デジタル信号処理、数理シミュレーションなどが RCS での処理に向いています。

これらのリコンフィギャラブル システムは、スピンオフ会社である Scientific Research Centre of Supercomputers and Neurocomputers（本社：ロシア、タガンログ）が製造と販売を手がけています。製造は発注時の仕様に基づいて BTO 形式で行われ、毎月約 100 台の生産能力があります。RCS には Virtex-5 および Virtex-6 FPGA が搭載されており、性能は 100 GFLOPS から 10 TFLOPS までと多岐にわたります。

FPGA ベースのリコンフィギャラブル コンピューター システムは、さまざまな分野の問題解決に独自の強みを発揮します。RCS を使用することでユーザーは、コンピューティング構造を自由に調整して短時間で問題に取り組むことができます。RCS の詳細については、http://parallel.ru/ を参照してください。

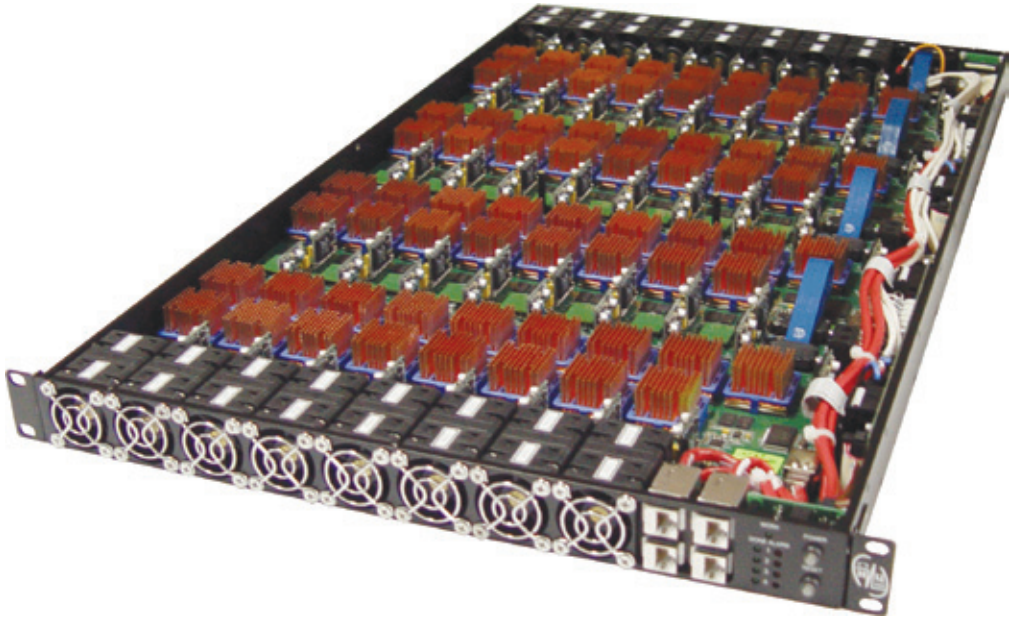


図 4 – 20 TFLOPS の Orion コンピューティング ブロック

MICREL®
Innovation Through Technology®

ノイズにセンシティブな製品に圧倒的なPSRR性能を!

リップル・ブロッカ (MIC94300/MIC94310)は、あらゆる電源のリップルノイズを激減する。

・1KHzでのPSRR：80dB
・5MHzでのPSRR：60dB

システムの電源のクリーン化による総合性能の向上。

・より高いRF送信信号強度
・音質、画像の向上

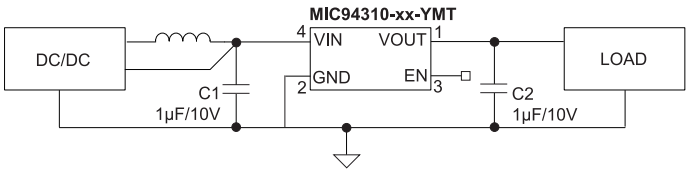
多機能システム・ソリューションでのサイズとコスト削減。

・ディスクリート回路比で63%以上の小型化
・0.8mmx0.8mm 4 バンプ WLCSP

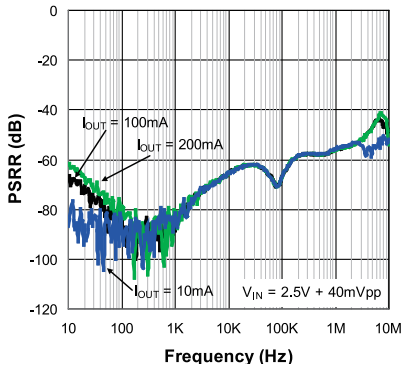
応用例：

- ◆ 車載カメラ、GPS
- ◆ タブレット PC/ ノートブック・コンピュータ
- ◆ WEB カメラ、デジカメ、ビデオカメラ
- ◆ セキュリティ・監視カメラ
- ◆ ビデオ会議システム
- ◆ バーコード・スキャナ
- ◆ スマートフォン・カメラ、RF 回路
- ◆ 医療向け画像処理機器
- ◆ その他車載、工業向け回路

標準的な利用例



PSRR C_{OUT} = 0.47µF



販売代理店

伯東株式会社
〒160-8910
東京都新宿区新宿1-1-13
TEL:03-3355-7629

ユニ電子株式会社
〒140-0001
東京都品川区北品川3-6-17
TEL:03-6811-7478

株式会社リョーサン
〒101-0031
東京都千代田区東神田2-3-5
TEL:03-3862-2635

詳しい情報についてはマイクレル代理店にお尋ねください。
また下記 WEB ページで検索できます。
http://www.micrel.jp/ripple-blocker/

マイクレル・セミコンダクタ・ジャパン株式会社
〒220-6014 神奈川県横浜市西区みなとみらい2-3-1
クイーンズタワーA14階
TEL: 045-224-6616 FAX: 045-224-6716

Bottling a Star Using ARM's AXI4 in an FPGA

ARM AXI4 をサポートする FPGA で核融合実験への道筋を切り開く

Billy Huang
PhD Researcher
Durham University / CCFE
billy.huang@ccfe.ac.uk

Dr. Roddy Vann
Assistant Professor
University of York
roddy.vann@york.ac.uk

Dr. Graham Naylor
Head of MAST Plasma Diagnostics and Control
Culham Centre for Fusion Energy (CCFE)
graham.naylor@ccfe.ac.uk

Dr. Vladimir Shevchenko
Senior Physicist
Culham Centre for Fusion Energy (CCFE)
vladimir.shevchenko@ccfe.ac.uk

Simon Freethy
PhD Researcher
University of York / CCFE
simon.freethy@ccfe.ac.uk

イギリスの核融合研究者が、ARM AXI4 インターフェイスを組み込んだザイリンクスの FPGA を使用し、合成開口イメージング システム用のデータ収集システムを試作

核融合エネルギーは超高温の条件下で水素原子がより大きな原子へと融合するときに生まれるエネルギーです。太陽を含むあらゆる恒星が核融合反応によってエネルギーを生み出しています。地球上で核融合エネルギーを得るには、トカマクといわれる磁気容器 (図 1) の中で、イオン化した水素ガス (プラズマ) を 1 億ケルビン以上に加熱する必要があります。

オックスフォード近郊で核融合エネルギーの研究開発を行う世界的な機関の 1 つ、Culham Centre for Fusion Energy (CCFE) に所属する核融合研究者である筆者らの最終目標は、地球上に大量に存在する水素を燃料とする核融合エネルギー発電所を誕生させることです。核融合を実用化できれば、地球上の水素を使用して人類のエネルギー需要を 100 万年以上にわたって満たすことができるでしょう。しかし核融合は、ガラス瓶の中に星を閉じ込めるのと同じくらいに実現が難しいとされています。世界最大規模の 200 億ドルの予算が組まれている国際的な ITER (International Thermonuclear Experimental Reactor) 事業において、核融合発電を世界で初めて実用規模で実証しようと試みています。20 年以内の稼動を目指して、国際熱核融合実験炉 ITER (ラテン語で「道」あるいは「旅」の意味) をフランス南部に建設中です (http://www.iter.org/)。

核融合研究における重要な技術課題の 1 つが核融合プラズマのリアルタイムでの計測です。各計測にはそれぞれの要件が定められています。CCFE (http://www.ccfe.ac.uk/) では、プラズマ内部を流れる電流を計測するためにプラズマから放射されるマイクロ波を画像化する観測装置を開発しました。これを実現するために着手したのが、合成開口レーダーの画像処理システムの設計です。

マイクロ波の位相を解析

合成開口の画像処理には人間の耳と同じような働きを持つフェーズド アレイ アンテナ (図 2) を使用します。音源が右側にある場合、音の波は左の耳よりも右の耳のほうに早く達します。別の言い方をすると、音の波は左右の耳に異なる位相 (フェーズ) で届くことになります。このような位相差から脳は音の方向を知覚します。これと同じ方法で、アンテナ アレイによって検出されたマイクロ波の位相を解析すればどの方向からマイクロ波が放射されたのかが特定できます。この原理を用いたフェーズド アレイ アンテナで得られたデータをもとに、プラズマ周縁の画像を合成します。

各アンテナで受信された周波数範囲が 6GHz から 40GHz の信号は、無線周波数 (RF) システム (図 3 参照) を介して、FPGA データ収集ボックスが処理できる 250MHz 帯域の信号へとダウンコンバートされます。信号帯域を 250MHz にしたことで、アナログ デジタル コンバーター (ADC) のクロック要件が決定します。今回は 8 個のアンテナを使用し、デジタル変換が必要なチャネル数は 16 チャネルとなります (信号を、実数要素と位相が 90° シフトした虚数要素に分解して処理するため、チャネル数はアンテナ数の 2 倍となる)。

このシステムでは、16 本のアナログ チャネルから、サンプリング レート 250MHz かつ分解能 14 ビットで、0.5 秒間データをキャプチャする必要がありました。データ レート要件は、14 ビットデータを 2 バイトに割り当てると 32 バイト ×0.25GB/s = 8GB/s となります。0.5 秒間に 4GB ものデータをキャプチャする必要があるだけでなく、ADC ベンダーの選択に自由度を得たいことと将来の移植性を考慮して、FPGA メザニンカード (FMC) インターフェイスを備えた FPGA ボードを使用することを考えました。また、内製した FMC デジタル I/O ボードが使用できることも条件の 1 つでした。

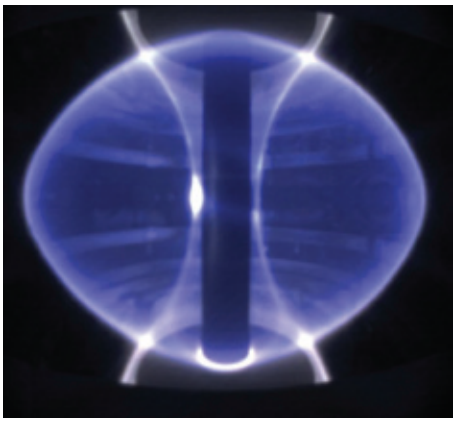


図 1. 従来のドーナツ型ではなく芯のあるリンゴのような独自の形状をした CCFE の Mega Amp Spherical Tokamak (MAST) で、核融合が装置内部で起きている様子を示す

2010 年夏の時点で、ザイリンクスの Virtex®-6 LX240T ML605 ボード 2 枚と、4DSP 社が提供する FMC108 (8チャネル) ADC ボード 2 枚を組み合わせて使用することが最適なソリューションであるとの結論を得ていました。8GB/S というデータ レートは、現在もそうですが、その当時きわめて高い値でした。そこで、より数多くの FPGA ボードを使用し、それぞれに割り当てるチャネル数を少なくして、分割統治法で対処することも考えました。しかし、この方法ではシステムのコストとサイズが増加してしまいます。

2011 年 1 月頃に、ARM 社の AMBA® AXI4 インターフェイス プロトコルをサポートする ISE® デザイン ソフトウェアの新バージョンをザイリンクスがリリースしたという情報が筆者らの元にも届きました。それより以前に AMBA AXI4 インターフェイスはハードウェアとしてはすでに存在していましたが、その機能をフルに発揮するツールはありませんでした。

図 2. 新しい PCB アンテナを活用したマイクロ波イメージ処理用のフェーズド アレイ アンテナ

AXI4 登場以前

筆者らが開発を試みたシステムでは、キャプチャしたリアルタイム データに Linux からもアクセスできるように、Virtex-6 内の MicroBlaze™ プロセッサから DDR3 SDRAM メモリへアクセス可能にする必要がありました。そのため、MicroBlaze バスと開発したリアルタイム ストリーミング IP の両方からアクセスできるメモリ コントローラーを使用しなければならないという制約が生じました。当初 PLB バスの使用を検討しましたが、筆者らが要求する動作周波数では、PLB バス用のメモリ コントローラーには 64 ビット幅ではなく 32 ビット幅のインターフェイスしか接続できませんでした。設計の過程で、低レベル NPI プロトコルを介してメモリ コントローラーと直接通信するコア部分をプログラムした後で、2GB/S しか達成できないことが判明したのです。このデータ レートはそれ自体高い数値で、調べた限りではほかの方式よりも高速でしたが、まだ充分な速度とは言えませんでした。

こういった中で、400MHZ ダブルデータ レートで 64 ビット アクセス (毎秒 8 億トランザクション) が可能な AXI4 インターコネクトとメモリ コントローラーがザイリンクスから発表されました。各ボードに求められていた 4GB/S という要件を充分に上回る、6.4GB/S という実質的なスループットを実現するということで、まさに筆者らが必要としていたものでした。

このスピードを実現するには、2 つの方法があると考えられていました。1 つは AXI_V6_DDRX メモリ コントローラー (AXI インターコネクト レイヤー下に隠れている) を変更する方法、もう 1 つは System Generator で生成される AXI Master PCore を使用する方法です。PCore は AXI External Master の 1 つとして Xilinx Platform Studio (XPS) 内で MicroBlaze システムに接続できます。

どちらの方法でもデータは DDR3 メモリに 5GB/S で転送されます。AXI のプログラムは簡単で、読み出しと書き込みが個別のチャネルで構成できるため、きわめて高いメモリ速度を実現できます。また、XPS ツールによって AXI デザインの自由度が広がります。たとえば、書き込みチャネルが必要であればそのチャネルのみを選択することが可能です。このような設計の自由度を活用して、ロジック設計の簡略化とリソースの軽量化を図ることにしました。

ソフト プロセッサ インターフェイス

ザイリンクスのツール セットには MicroBlaze というソフトウェア プロセッサが独自の機能として備わっています。ソフトウェア プロセッサの「ソフト」は、FPGA のロジックを用いて実現されるということを表しています。ザイリンクスやパートナーの努力もあり、MicroBlaze プロセッサは Linux カーネルのメイン ブランチでサポートされています。彼らの努力に応えるため、筆者らは Linux コミュニティにてさらにこの開発が進むよう支援を行っています。

MicroBlaze 上で Linux カーネルが動作するということは、FPGA システムに PC のようなインターフェイスを実装できることを意味します。たとえばウェブ サーバーや SSH サーバーが FPGA 上で利用できるようになり、応用範囲が広がります。筆者らは、ファームウェアを遠隔でアップデート可能な System ACE™ フラッシュ メモリを (MS-DOS タイプとしてフォーマットされている場合) Linux 下に実装することにしました。

ネットワーク ストリーミング

各 FPGA ボードで 0.5 秒間以内に 2GB をキャプチャできる場合、このようなデータ量を標準的なインターフェイスを介して許容できる時間内にどのように読み出すかという問題に直面しました。Linux と UDP のような単純なプロトコルの組み合わせでは、Gigabit Ethernet を介した MicroBlaze プロセッサのネットワーク速度はおよそ 0.5MB/s にしか達せず、あまりにも低速であることが判りました。この速度では 0.5 秒間でキャプチャしたデータをダウンロードするだけで 1 時間以上も待たなければなりません。

性能を得るにはデザインの下位レベルの回路で対策を講じなければならないことは明らかでした。そこで筆者らが開発した FireStark

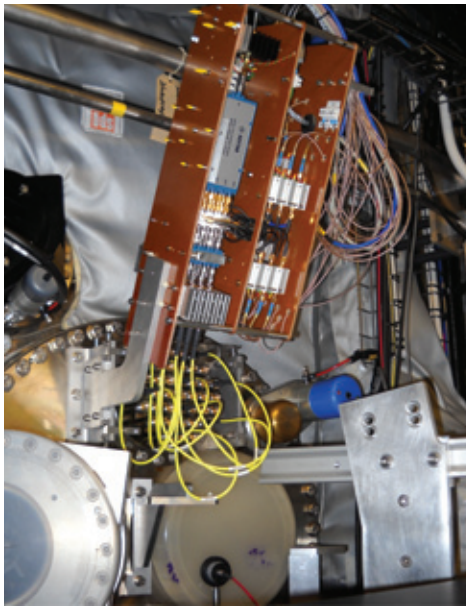


図 3. MAST トカマクに接続された RF システムが、6GHz から 40GHz の入力信号を FPGA データ収集システムによって処理される 250MHz 帯域信号へダウン コンバートする

という UDP ベースの独自プロトコルを AXI Ethernet DMA ドライバー内に実装することにしました。MicroBlaze Linux カーネル ドライバーの変更と合わせ、FPGA に専用のプライベート ネットワークを実装することで、2GB のデータ全体を 60 秒以内にダウンロードできるようになり、70 倍もの速度向上を達成できました。最大 6KB のジャンボフレームでテストを行ったところ速度は 2 倍になり、すなわち、70MB/s 以上が得られました。MicroBlaze のクロック周波数は 100MHz とあまり高くはありませんが、DMA を使用すればメモリからネットワークに対して高いストリーミング スループットが得られるという点は重要です。

FPGA から PC までのレイテンシは実測で 129μs±13μs でした（この計測値には、スイッチ、PC カーネルを介し、ネットワーク スタックを上ってユーザー スペースに入るまでのレイテンシのオーバーヘッドが含まれているため、実際のレイテンシはもう少し短くなる）。また、FPGA から FPGA へのレイテンシも計測する予定ですが、これはより短いレイテンシになると見込んでいます。

クロックの同期制御

MAST トカマクには多くの診断装置やシステムがつながっていて、それらはすべて実験用の 10MHz グローバル クロックに同期して動作する必要があります。筆者らは、250MHz 捕捉クロックをこの 10MHz クロック信号から生成し、得られたクロックを ADC ボードに供給しています。FPGA のその他のロジックはオンボードの水晶クロックで駆動しています。

筆者らのシステムは、およそ 10 秒間のトリガー イベントを除いて、実験用クロックを連続的に使用しないという比較的珍しい構成となっています。トリガー イベントの 10 秒間以外は内部生成クロックを使用します。つまり、相互に切り替わる外部クロックと内部クロックの 2 系統のクロックが存在します。

2 枚の FPGA ボードの主な要件として、クロックを正確に同期させなければなりません。サンプリング周期は 4ns なので、最高周波数で動作する ADC の入力仕様に適合した 8ns 周期の正弦波が入力に必要です (8ns 周期は、位相として 360° に相当)。5° の位相精度が必要な場合、スキューの最大値は 8×(5/360)= 111ps です。このレベルで精度を維持することはきわめて難しい課題です。光でさえ 3.3cm しか進めません。

ファームウェアは両方のボードで同じものを使用しました。それぞれのボードに固有の機能は DIP スイッチを用いて有効または無効にできるようにしています。この方法により、ファームウェアを一度合成させるだけで済むようになり、開発期間が大幅に短縮されました。

一方のボードで生成されたクロックは、隣接する 2 個の SMA ポートから外部に出力され、各 FPGA ボードの FMC ポートに (等長ケーブルを用いて) 接続されている ADC ボードへそれぞれ供給されます。このような構成により、FPGA ボードの 2 個の SMA ポートまでの配線長差に相当する位相の違いはありますが、その点を除いてはまったく同じクロックで 2 枚のボードは動作することになります。図 4 に概略構成を示します。

外部の 10MHz クロックが受信後送信されて両方の ADC ボードに与えられているように、外部トリガーについても同じ方法で両方のボードが同時に受け取れるようにしています。

独自機能がもたらす効果

ザイリンクス FPGA が備えるさまざまな新機能をデザインに利用しました。たとえば、各ピンのパス遅延を微調整する IODELAY プリミティブを使用してトラック長の違いを補正しています。FMC に搭載される ADC のデータ パス長は完全な等長ではないため、この機能は必須でした。パス遅延を補正しなければ、ADC の出力データの有効性が失われる可能性があったからです。ADC からは 250MHz クロックのダブルレートでデータが出力されるため、データの有効な期間はわずか 2ns しかありません。IODELAY を使用すると、データ パスを 125ps ステップというきわめて高い分解能で調整できます。

同様に重要な機能が、逓倍や位相シフト

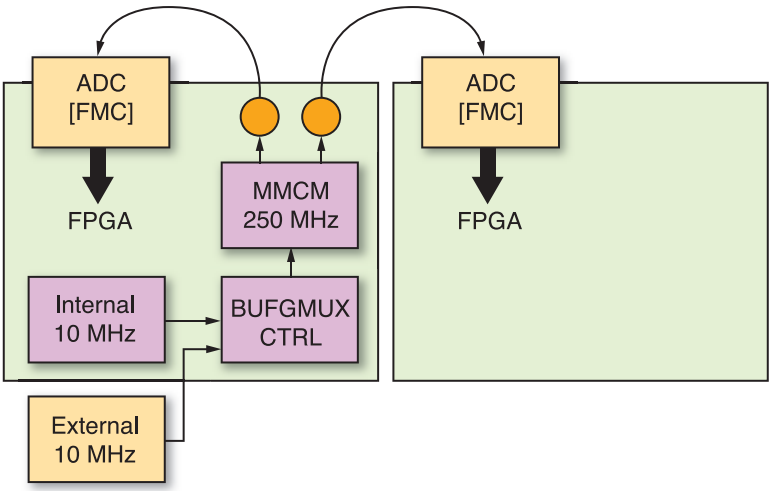


図 4. 厳格な同期が必要な 2 枚の FPGA ボードと正確な同期を実現したクロック系の構成

図 5. ザイリンクスの ML605 評価ボードと 4DSP 社の FMC108 ADC ボード、そして筆者らが開発した FMC/PMOD ヘッダー ボードで構成される FPGA データ収集ボックス。ADC アナログ接続の寿命を延長するため、ADC SSMC 内部コネクタをフロント パネル SMA バルクヘッドに接続

などのクロック マネージメント処理を行う MMCM (Mixed Mode Clock Manager) です。1 つの MMCM をほかの MMCM に接続するカスケード モードを用いて、オリジナルの 10MHz クロック、250MHz ADC サンプリング クロック、別の目的に使用したクロックなどさまざまな周波数のクロックを生成しています。

BUFGMUX_CTRL プリミティブと IDDR プリミティブも同様に活用しました。今回開発したシステムでは内部 10MHz クロックと外部 10MHz クロックとを切り替えて使用するため、2 系統のクロック信号をグリッチなしで切り替える必要があります。これには BUFGMUX_CTRL を用いました。なお、このプリミティブは (クロック制御だけでなく) トリガー信号のような標準ロジックにも使用できますが、その場合はロジック信号が通過するように、属性 IGNORE0 と IGNORE1 を 1 に設定してグリッチ除去回路をバイパスする必要があります。

ADC はデータを DDR フォーマットで出力します。つまり、データはクロックの立ち上がりエッジと立ち下がりエッジの両方で有効です。このデータをシングル データ レート (SDR) のデータとして復元するために、I/O パッドにハードワイヤされる IDDR プリミティブを用いました。このプリミティブは 1 本のデータ ピン入力と 2 本のデータピン出力で構成されています。SAME_EDGE_PIPELINED 属性を使用して両方のピン上のデータが同時に有効であることを保証し、ロジックを追加する必要性をなくしました。この属性にはレイテンシ サイクルが増えるというデメリットがありますが、これは問題になりませんでした。

筆者らのデザインに役立ったザイリンクス アーキテクチャのもう 1 つの特徴が FPGA メザニン カード (FMC) コネクタです。厳密に言えば FPGA の特徴ではなく FPGA ボードの特徴ですが、このコネクタは Virtex-6 FPGA に搭載されており、十分な性能を発揮して設計に大きく貢献してくれました。FMC コネクタには、ML605 ボード上の Virtex-6 のクロック対応ピンに接続されている高周波クロック ピンが割り当てられています。これにより、クロックを FMC を介して FPGA へ送信することができました。クロックの入力ポイントを一箇所に限定できるという点でも、この機能は非常に有益でした。

ザイリンクスのツール スイートの使用

ザイリンクスから FPGA システムの開発を支援するさまざまなツールが提供されています。筆者らはこれらのツールを複数活用して設計を進めました。

VHDL および Verilog の手動によるコーディングには Project Navigator を使用しました。ロジックを視覚化して回路図を作成する、グラフィカルなインターフェイスも搭載されています。ただし、このツールによってシングル ビットのフリップフロップは簡単に操作できますが、範囲を拡大して多くのビットを扱おうとすると操作が複雑になってしまいます。そのため、筆者らは Project Navigator をロー レベル ツールと捉えることにしました。このツールはロー レベルなクロック デザインにとても有効で、特定のロジックを駆動する特定のクロックを正確に制御できます。

ハイ レベルのロジック デザインには System Generator を用いました。System Generator は、ロジックが単一のクロック周波数で駆動されるデザインに特に適しています (ただし、これに制限されるわけではない)。このツールは、使用が簡単というだけでなく、たとえば FFT、除算器ジェネレーター、フィルターなどの幅広い IP を利用できます。また、設計したロジックを、読み出し / 書き込みレジスタや共有メモリとして簡単に MicroBlaze へ接続できます。System Generator によってペリフェラル コア (PCore) が自動的に生成され、XPS プロジェクトに追加されます。

ADC FIFO のパラメーターの微調整には CORE Generator™ を活用しました。FIFO は 256 ビット幅とし、書き込みクロックは 125MHz、読み出しクロックは 200MHz に設定しました。生成された NGC ファイルは PCore として XPS にインポートしました。この処理は、必要な .mpd ファイル、.pao ファイル、.bbd ファイルを生成して手動で行いました。

FPGA のプログラミングと、ファームウェアを CompactFlash 上に恒久的に配置する SystemACE™ ファイルの生成には、Impact ツールを用いました。CompactFlash は正しく機能しましたが、筆者らのシステムにはさらに要件を追加しました (以下の SDK 参照)。

システムに MicroBlaze プロセッサを組み

込むため、Xilinx Platform Studio (XPS) を使用してプロセッサ システムを生成しました。XPS は、プロセッサを中核としたシステムを構築できる包括的なツール スイートです。必要なリンクはウィザードを介してセットアップできるようになっています。Create IP ウィザードを使用すれば、CORE Generator で生成した IP を利用することもできます。AXI4 オンチップ インターコネクトも利用可能な IP として追加されています。

最後に、プロセッサ上で動作するソフトウェアを開発するためにザイリンクスのソフトウェア開発キット (SDK) を使用しました。実際、電源投入後に実行すべきプログラムは SREC ブートローダーのみでした。CompactFlash には FAT ファイル システムが使用されているため、SREC プログラムへアクセスするために必要なライブラリ (同じフラッシュ上にある) によって最終的な実行ファイルのサイズは増大してしまいます。そこで、デバッグ機能をオフにして最適化をオンにし、ポストコンパイル コマンドとして mbstrip -g <elf_file_name> をインクルードすることで、サイズをできるだけ抑えようと考えました。ただし、これらをすべて適用した後も実行ファイルのサイズは 91KB という大きさでした。そのため、このサイズの実行ファイルを含むビット ストリームを初期化できるように、内部 BRAM のサイズを増やす必要がありました。

ここで直面した問題の 1 つが Virtex-6 のコンパイル時間の長さでした。これにはザイリンクスのソフトウェア PlanAhead™ が非常に役立ちました。コンパイル時間を短縮するために PlanAhead の機能をフルに活用しました。

新たな Zynq™-7000 エクステンシブル プロセッシング プラットフォーム (Xcell Journal 75 & 76 合併号の「イノベーションの新時代を拓く Zynq-7000 EPP」を参照) にも多くの期待を寄せています。ただし、Zynq が MicroBlaze を置き換えてしまうのか、それともそのソフトウェアという性質と 10 年以上の開発の蓄積を生かして MicroBlaze が生き残るのかは現時点では判断できません。MicroBlaze は将来、キャッシュ コヒーレントなマルチプロセッサ システムとなって、デュアルコアの ARM® Cortex™-A9 MPCore™ の性能を上回るようになるのでしょうか。Zynq または MicroBlaze の Physical Address Extension 機能は、32 ビットを超えるアドレス空間、つまり 4GB を上回る RAM を提供するような、より強力な性能をシステムにもたらしてくれるのでしょうか。今後の動きが気になりますが、その答えは時間の経過とともに明らかとなるでしょう。

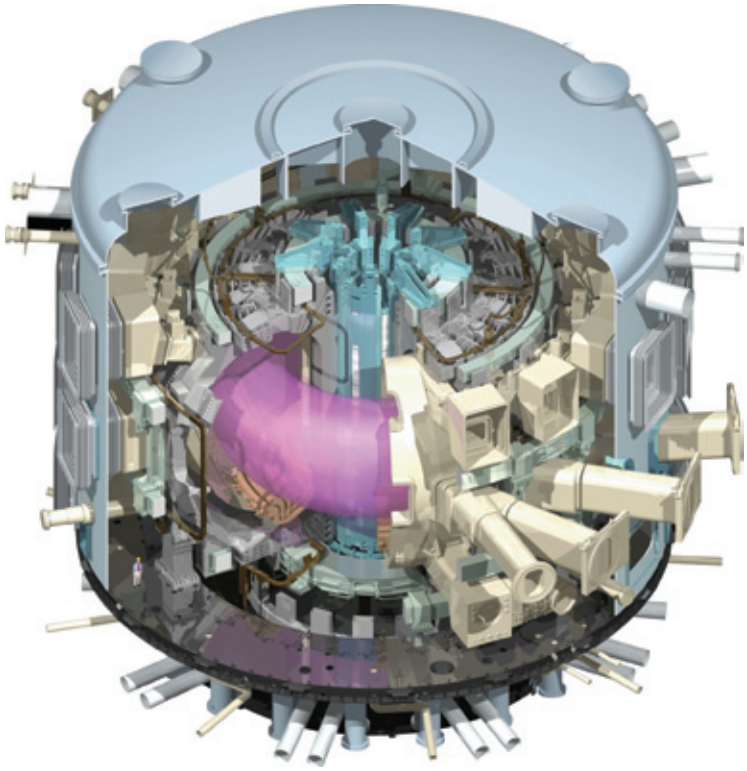


図 6. フランス南部に現在建設中の ITER トカマク - 核融合エネルギーによって 500MW を発電し、実用核融合炉への道を切り開く

最先端の研究を支えるシステム

以上のように、最新のザイリンクス テクノロジを活用して、FPGA の世界でも最先端となるデータ収集システムを完成させることができました (図 5)。このシステムは 10GB/s (80Gbit/s) というデータ レートでリアルタイムのデータ収集が可能です。最終的な費用は 15,000 ドル以下で収まっています。筆者らは、ITER プロジェクトのような大規模の核融合実験への道筋が切り開かれることを願いながら、システムの実証実験を行いました。

核融合エネルギーは人類にとってきわめて難しい技術的挑戦の 1 つです。独自の特徴を備えた FPGA は、困難な障壁をこれまでとは異なるアプローチで乗り越えようとしている筆者らの一助となっています。以上のように、最新の AXI4 インターコネクトをサポートする Virtex-6 FPGA およびザイリンクス ツール フローを活用して、きわめて高いデータ レートを実現するとともに、高密度なロジックをコンパクトに実装した、核融合研究向けデバイスを構築できました。

謝辞

CCFE は英ダラム大学(Durham University) の応用計測センター (Centre for Advanced Instrumentation) および英ヨーク大学 (University of York) のプラズマ研究所 (Plasma Institute) と連携して研究を行う機関です。

この研究は、EPSRC 認可番号 EP/H016732、ヨーク大学、RCUK Energy Programme 認可番号 EP/I501045、および EURATOM と CCFE 間の契約に基づき European Communities から、それぞれ資金提供を部分的に受けています。

ザイリンクスのオープン ソース Linux のエンジニアである John Linn 氏をはじめ、MicroBlaze プロセッサの Linux サポートについて貢献してくれたザイリンクス社員およびザイリンクス パートナーに謝意を表します。

www.aldec.com/jp

Your Global Xilinx® Verification Solutions Provider

VHDL, Verilog, SystemVerilog
OVM/UVM, VMM
コードおよびファンクション・カバレッジ
DSPコ・シミュレーション（MATLAB® & Simulink®）
エミュレーション（ハードウェア・アシステッド検証）

アルデック・ジャパン株式会社
〒160-0022 東京都新宿区
新宿1-34-15　新宿エステートビル9F

Phone: 03-5312-1791
Fax: 03-5312-1795
Email: sales-jp@aldec.com
www.aldec.com/jp

Europe
Mercia House
51 The Green, South Bar
Banbury, OX16 9AB
United Kingdom

Phone: +44.1295. 20.1240
Email: sales-eu@aldec.com

Israel
Even Yehuda 40500
6 Macabi St.
POB 2521
Israel

Phone: +972.5.2257.3422
Email: sales-il@aldec.com

China
Suite 2004, BaoAn Building
#800 DongFang Road
PuDong District
Shanghai City, 200122, P.R. China

Phone: +86.21.6875.2030
Email: info@aldec.com.cn

India
#4123, Ist Floor
6th Cross, 19th Main
HAL II Stage Indira nagar
Bangalore, 560008, India

Phone: +91.80.3255.1030
Email: sales-in@aldec.com

Taiwan
Room 920, 8f, no.8
Lane 360, sec.1
Neihu Rd., Taipei
Taiwan

Phone: +886.2.2659.9119
Email: sales-tw@aldec.com

Headquarters-US
2260 Corporate Circle
Henderson, NV 89074
USA

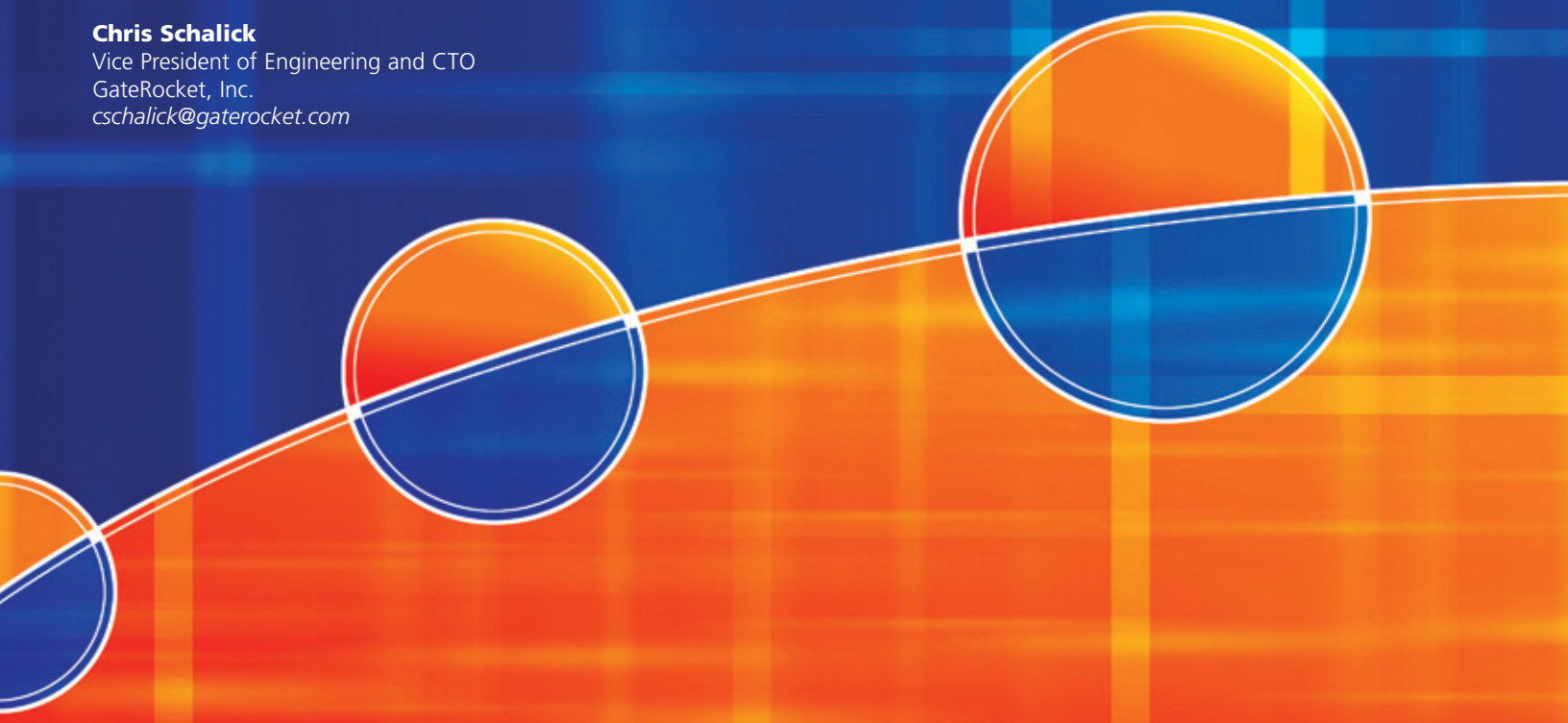
Phone: +702.990.4400
Email: sales@aldec.com

More Than One Way to Verify a Serdes Design in FPG

FPGA 内の SerDes デザインを検証するアプローチを比較

アプリケーションの複雑さ、開発期間 / シミュレーション時間 / 精度のトレードオフによって決定する最適なアプローチ

Chris Schalick
Vice President of Engineering and CTO
GateRocket, Inc.
cschalick@gaterocket.com



FPGA の性能および容量が向上するにつれて、メディア、信号処理、通信など幅広いアプリケーションで FPGA が接続インターフェイスとして使用される機会が増えています。並行して、オンチップおよびチップ間通信には、従来のパラレル バスよりもはるかに高いデータ レートを実現できる高速シリアル接続への移行も進んでいます。このシリアル インターフェイスで重要な役割を果たすのが、SerDes (シリアライザー / デシリアライザー) テクノロジです。SerDes アプローチに基づいたプロトコルは、少ないデバイス ピンで高いデータ レートでの動作を可能にします。

SerDes はマルチギガヘルツのライン レートを実現しますが、これが導入されたことで FPGA デザインは新たな課題に直面しています。中でも特に大きいのがシグナル インテグリティの問題ですが、それに勝るとも劣らず難しいのは、このテクノロジ自体の複雑さに起因する機能検証です。SerDes ベース デザインのロジック シミュレーションには長いシリアル テスト シーケンスが必要で、通常よりも 1 ～ 2 桁長いシミュレーション時間がかかります。しかも、SerDes テクノロジは複雑な階層化プロトコルを採用しており、内部ロジックを徹底的に使用することがますます難しくなっています。さらに、SerDes は使用に慣れていないサードパーティ提供の IP ブロックとしてデザインに組み込まれることが多いため、最終システムのデバッグは困難を極めます。

機能検証のアプローチを工夫すれば、こうした SerDes シミュレーションのボトルネックを解消できます。ただし、機能検証の方法によって検証におけるパフォーマンス、精度、エンジニアリングの生産性が左右されるため、アプローチを選択する際は、開発期間、シミュレーション時間、シミュレーション精度のトレードオフを検討する必要があります。具体的に次のようなアプローチが考えられます。

- シミュレーションから SerDes を除外して、それ以外の部分のチップをパラレル通信で検証する
- テストベンチにもう 1 つ SerDes を追加し、2 つの SerDes を back-to-back で接続する
- デザインのうち、SerDes 部分のみをラボでシステム内検証を行う
- エミュレーションと同様のアプローチで、デザイン全体をネイティブ FPGA ハードウェアで実行する

シミュレーションの複雑さ

最近の FPGA デバイスにはコンフィギュレーション可能な高性能 SerDes 回路が内蔵されており、省ピン化に貢献するチップ間データ転送プロコトルといった簡単なものから、最新のコンピューター マザーボードに接続する標準規格バスといった高性能なものまで、幅広いアプリケーションで SerDes テクノロジを利用できるようになっています。通常、これらの SerDes 回路はハード IP ブロックとしてエンドユーザーに提供されます。現在使用できる FPGA の一般的な SerDes テクノロジは、10Gbps を超えるビット レートに達しています。

ザイリンクス Virtex®-5 の GTP_DUAL セルは最近の代表的な SerDes です。このセルには 100Mbps から 3.75Gbps までの動作が可能な 8 つのシリアル I/O 信号に加え、342 のコア側の信号があり、そのうちの一部はオプションでアクティブになります。さらに、184 個のコンフィギュレーション可能なパラメーター、9 つのクロック入力、5 つのクロック出力などの充実した機能セットを備えています。

これほど多くの機能を搭載したデザインを設計、検証する場合、SerDes 回路に関する詳細な情報がなければ大変な作業になります。このトランシーバーに関する資料 [1] には、SerDes モジュールによってサポートされている 17 種類の通信規格の一欄が記載されています。これらの規格で使用される基準クロックの周波数は 15 種類にも及び、パラメーターのコンフィギュレーションもそれぞれ異なります。

標準化されていない独自プロトコルでは、FPGA SerDes の任意の設定とクロック周波数を使用でき、それらは標準規格に基づいたプロトコルと同じ場合もあればそうでない場合もあります。コンフィギュレーション可能なパラメーターについては、2 値をとるものが 68 個、数値型が 70 個で合計 730 の可変ビットがあり、8 つは非数値の複数の値をとります。このシミュレーション モデルでは、パラメーターの設定の組み合わせだけでもコンフィギュレーションの種類が 2730 を超え、途方もない数字になります。また、SerDes トランシーバーはさまざまなモードで動作することが考えられるほか、幅広いユーザー デザインにも対応します。

この SerDes デザインの動作を正確にモデル化するには、非常に複雑なシミュレーション モデルが必要です。シミュレーション モデルが複雑になると、デザインのロジック シミュレータにかかる負荷も大きくなります。実際、FPGA SerDes を使用したデザインでは FPGA SerDes モデルのシミュレーションに最も多くの時間がかかります。

トランザクターの問題点

デザインの種類を問わず、基本的にインターフェイスをテストする際はトランザクターを使用するのが一般的な検証方法です。トランザクターとは、インターフェイスのピンに接続してインターフェイスのデータ プロコトルを提供 / 使用するモデルをいいます。通常、トランザクターはサイクル精度のピン レベル機能を抽象化し、より理解しやすく扱いやすい精度の機能に変換する役割を果たします。FPGA SerDes システムの接続をモデル化した完全なトランザクターを開発するには、非常に高い柔軟性と機能が要求されます。

FPGA SerDes の外部インターフェイスをモデル化する際は、開発期間、シミュレーション時間、精度のトレードオフを考える必要があります。最もシンプルなインプリメンテーションは、トランザクター内に SerDes シミュレーション モデルをもう 1 つ追加する方法です。この方法では開発期間は短かくなりますが、SerDes シミュレーション モデルのロジック シミュレータにかかる負荷が 2 倍になります。それとは別に、ごく簡単なビヘイビア モデルを作成して実行時

間を短縮する方法もありますが、この場合、機能の完全性と精度が犠牲になります。これらの中間に位置する方法として、FPGA デザインで実際に使用する機能のみをモデル化し、その他の SerDes 機能はテストから除外するというアプローチがあります。

システム機能の検証

SerDes ベース デザインの検証手法について詳しく論じる前に、検証漏れ（機能シミュレーションで発見できないバグ）が発生する主な原因と、これらのバグを検出する難しさについて説明します。つまり、なぜ FPGA SerDes の機能検証が必要なのか、という点です。

FPGA ベース システムの検証は、3 年前の ASIC ベース システムの検証の規模にほぼ匹敵します。現在一般的な FPGA デバイスには、500,000 個のフリップフロップ、マルチメガバイトのオンボード RAM、ハードおよびソフト マイクロプロセッサ コア、そして特定用途向けの通信 / データ処理 / バス インターフェイス IP を含むロジック デザインをインプリメントできます。このようなデバイスを組み込んだシステムを検証するには、厳格な方針が必要です。インターフェイスの動作、サブシステム同士の相互作用、ロジックとインプリメンテーションの正確さに関する前提的なデータの妥当性を実証しなければなりません。FPGA SerDes テクノロジを使用すると、これらすべての作業が複雑になります。

インターフェイスの動作

機能検証漏れの原因としてまず考えられるのが、シリアル側またはパラレル側にある SerDes との直接接続の部分です。通常、シリアル側のデータは、ユーザー データをユーザーのコア側ロジックと設計中のデバイス外部の回路間にあるデータ管理スタックによってカプセル化してエンコードした形となっています。ユーザー データに対する変換スタックは、浅い場合と深い場合があります。

図 1 に示したデザインは、簡単な浅いスタックの例です。ここでは、FPGA SerDes デバイスの 8b10b エンコーディングを使用して 8 ビットのパラレル データを 10 ビットのシリアル データに変換しています。この場合、ユーザー パスの検証は、比較的単純な機能パターンで実行できます。つまり、256 のコア側サイクルに対してインクリメンタル パターンを用意するだけで、可能性のあるすべてのデータ ワードがインターフェイスを介して伝搬可能であることを検証できます。構造的には非常にシンプルな変換ですが、ネイティブ FPGA SerDes デバイスを用いて実際にこうした変換を実行するには、数百もの信号を接続し、さらに数百ものパラメーターを設定しなければなりません。このように簡単な例でさえ、接続や設定を間違えたり、デバイス仕様を読み違える可能性は十分にあります。したがって、どれほどシンプルな用途であっても、SerDes を使用する際は適切な検証を行わなければ機能検証漏れが生じます。

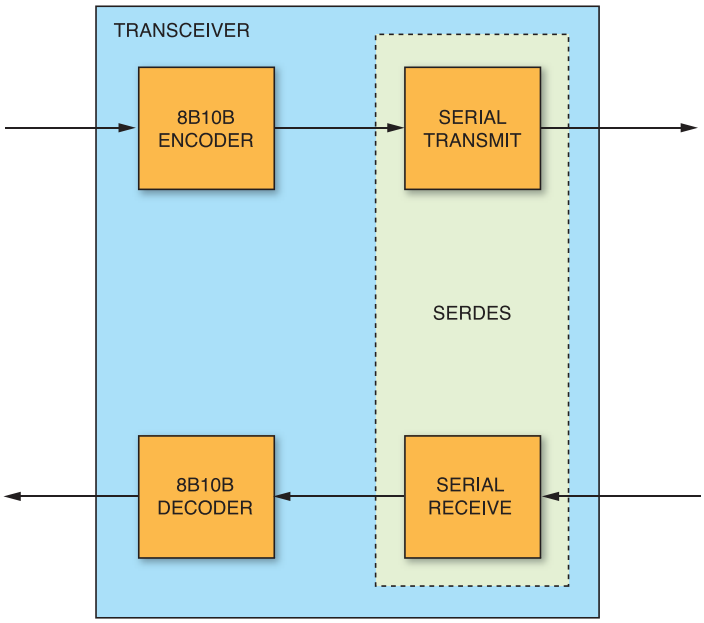


図 1 – シンプルな 8b10b の FPGA ブロック図

さらに複雑な例としては、XAUI、PCI Express®、RapidIO などパケット ベースのバス プロトコルがあります。一般に、これらのインターフェイスは FPGA デバイス内部にハードマクロとして用意されている SerDes IP とソフトマクロとして提供されている IP（プログラマブル ロジックのみ）を組み合わせて構成されます。こうした組み合わせの FPGA IP を、システム要件に合わせてコンフィギュレーションします。これらの外部シリアル インターフェイスは、標準バスへの接続に使用します。検証済みの IP を使用してバス インターフェイスをインプリメントすると、バス部分での基本的な機能エラーを防ぐことはできますが、アップストリーム インターフェイス（ソフトマクロとして提供される IP のコア側のパラレル データ インターフェイス）での検証が必要になります。これらインターフェイスでの制御およびデータ操作はベンダーによって異なり、仕様の誤解釈の余地が残るほか、標準バス インターフェイスに対してベンダー固有のデザインおよび検証作業が必要になります。

コア側のインターフェイスはベンダー独自の仕様に基づいていて標準化されていないため、デザインのコア側ロジックを動作させるには、目的のイベントがもたらされるベンダー固有のアクティビティを標準バス上に生成する以外に方法がありません。このテスト検証は、IP ベンダー間で移植可能なものではありません。IP のコア側のインターフェイス信号をアクティブにするには長いシミュレーション シーケンスが必要となることもあり、その場合、相互接続しているユーザー ロジックの検証に必要なシミュレーション時間はさらに長くなります。

図 2 に示した XAUI デザインでは、シミュレーションにかかる時間は 59 秒ですが、リンク上でのデータ転送をテストするのにかかった時間は 4 秒のみで、55 秒は XAUI リンクの初期化に費やされています。この初期化シーケンスは、テストのたびに実行しなければなりません。このように、FPGA SerDes シミュレーション モデルには非常

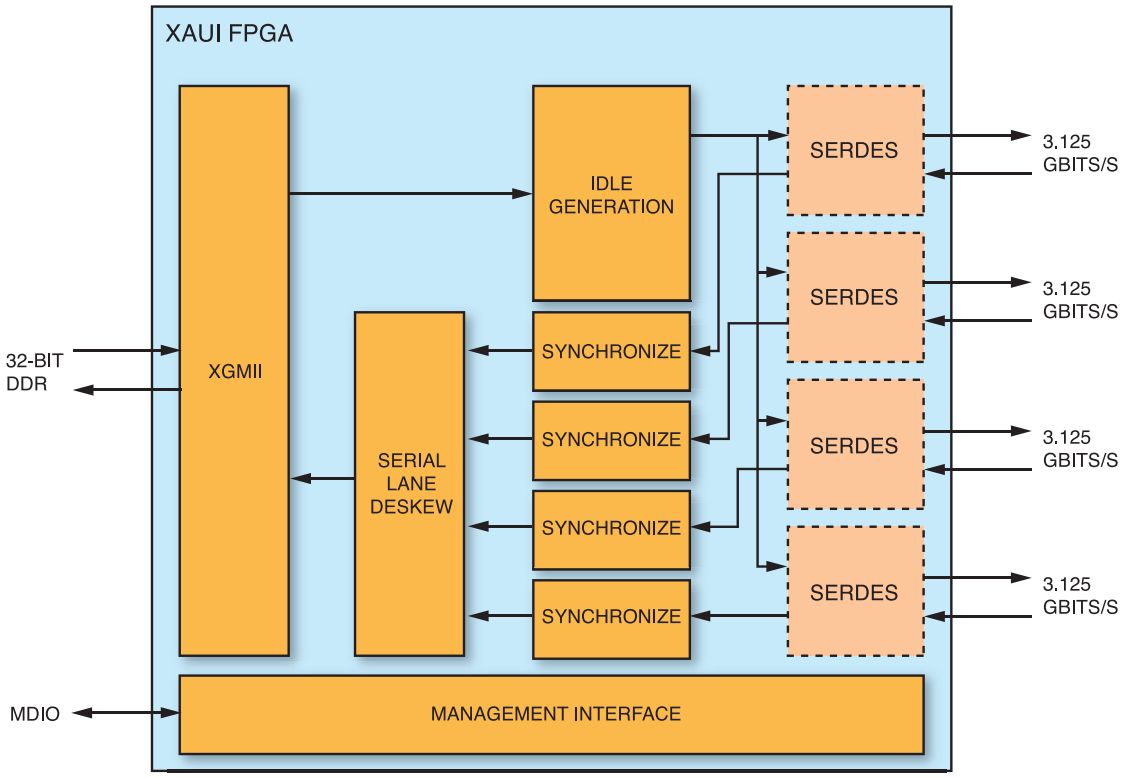


図 2 – XAUI FPGA のブロック図

に長いシミュレーション時間がかかる上、プロトコル IP のコア側インターフェイスが独自であるため、インターフェイスの動作検証は困難を極めます。

サブシステムどうしの相互作用

SerDes ベース システムで機能検証漏れが発生する一般的な原因としてもう 1 つ挙げられるのが、シリアル リンク両端にあるサブシステムどうしの相互作用です。チップ間通信に SerDes を使用した FPGA を複数搭載したシステム デザインは、個々のサブシステムだけでなくサブシステムどうしの連携動作においても仕様条件を満たし、複雑なシステム機能を実現する必要があります。SerDes コンポーネントの設計上の前提条件によってサブシステム検証の作業量が左右されます。

たとえば、1 つの FPGA デバイスから SerDes ベースのリンクを経由してもう 1 つの FPGA にデータを送信し、そこから元の FPGA にデータが戻ってくる場合、シリアライズとデシリアライズに要する時間が 2 回ずつ発生します。これらの時間はデバイスの温度や電圧によってばらつきがあるため、システムにはばらつきへの耐性が求められます。そこで、モデルの状態を選択的にベストケースまたはワースト ケースに変化させ、これらの条件下でデザインが正しく動作することを検証する必要があります。

たとえば図 3 の回路で、シリアル リンクで接続されたリモート デバイスに対して 1 サイクルに 1 回リクエストを発行した場合を考えてみます。各リクエストは、リモート デバイスから受信の応答確認 (ACK) があるまで FIFO に格納されます。シリアライズとデシリアライズの時間にばらつきがあると、この FIFO の使用量 (ワード数 (深さ)) が変わってきます。FIFO がオーバーフローする前に FIFO のワード数 (深さ) が最大に達することを確認するには、FPGA SerDes の遅延を正確にモデル化する必要があります。

先に述べたとおり、シミュレーション モデルの精度とシミュレーション時間はトレードオフの関係にあります。精度の低いモデルを使用するとシミュレーション時間は短くなりますが、FIFO オーバーフローのような機能検証漏れが発生する可能性が大きくなります。こうした検証漏れは、デザインがラボや顧客サイトに届くまで発見することができません。

インプリメンテーションとツール フロー

RTL シミュレーションでは見られないビヘイビアがゲート シミュレーションで顕在化することは一般的に知られています。ロジック シミュレーションでは多くの場合、テスト初期化シーケンスやデバイス シミュレーション モデルはデザインの初期状態を無視するか、または正しいものとして認識します。また、正しく記述された HDL コードでも、ゲート レベルのロジック シミュレーションでハイ インピーダンスや不定値を入力すると異なる動作となる場合があります。これらの状態は RTL シミュレーションでは機能検証の対象として検出されず、ラボ環境で初めて確認されます。これを受けて、設計者はインプリメント後の FPGA デザインの初期化およびゲート動作に対してサニティ チェックを行うため、ゲート シミュレーションを実行します。SerDes ベースの FPGA デザインも例外ではありません。

ゲートレベルでのロジック シミュレーションには時間がかかります。一般に、RTL デザインよりもゲートレベル デザインの方がシミュレータで処理するイベントの

数は 1 桁多くなります。ゲートレベル シミュレーションに SerDes シミュレーション モデルが加わると、パフォーマンスへの影響はさらに大きくなり、シミュレーションは非常に時間のかかる作業になります。モデル化の精度によっては、耐性の問題を捉えることさえできません。たとえば図 3 に示したデザイン例の RTL 部分は、RTL シミュレーションよりもゲート シミュレーションの方が 30 倍時間がかかります。

インプリメンテーションとツール フローで最も機能検証漏れの対象となりやすいのが、初期化エラーです。SerDes モデルに不定値が入力された場合、RTL で検出されないことがあります。このような問題は、ラボ環境でのゲート シミュレーションで明らかになります。

検証へのアプローチ

FPGA SerDes ベースのデザインを検証するには、これまで述べてきた一般的な検証漏れを考慮しながら SerDes シミュレーション モデルの柔軟性と複雑さが及ぼす影響に対応する必要があります。シミュレーション時間が増大すると、実用的なテストを実行する前に長時間の初期化シーケンスが必要となります。ただし、こうした長時間のシーケンスに対処する方法もあります。

次のセクションからは、実際にいくつかの検証アプローチを紹介します。いずれも、シンプルな双方向 8b10b シリアル リンク (図 1) と 10Gbps XAUI インターフェイス (図 2) の 2 つのデザインの場合について考察します。XAUI のデザイン例では、データの変換後にコア側の XGMII パラレル インターフェイスと業界標準のシリアル XAUI インターフェイス間でデータ転送を実行しています。このデザインは、バージョン 8.1 のコアを使用するザイリンクスの CORE Generator™ で生成しました。XAUI コアの資料はザイリンクスから提供されています。[2]

では、これら 2 つのデザインを使用して機能検証アプローチを比較してみましょう。ここで紹介する実験結果は筆者が所属する GateRocket 社でシミュレーションを行ったもので、使用したマシン環境はすべて同じです。[3]

アプローチ 1：SERDES モデルを除外する

FPGA SerDes モデルを使用したデザインのシミュレーションで一般的なアプローチとしてまず挙げられるのが、SerDes モデルを完全に除外し、その代わりにコア側のパラレル データを発信元（送信側）から宛先（受信側）へ直接接続したシェルでそれらのモデルを置き換えるという方法です。これは、通常シミュレーション モデルのスコープに基づいて、テストベンチのトランザクターから SerDes モデルのパラレル出力を直接駆動してコア側のパラレル入力を観

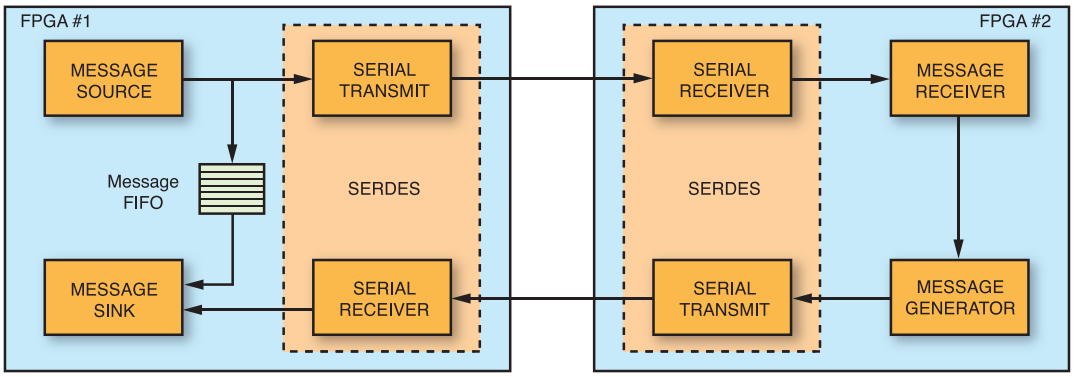


図 3 – 遅延のモデル化が不正確になりやすいデザイン例

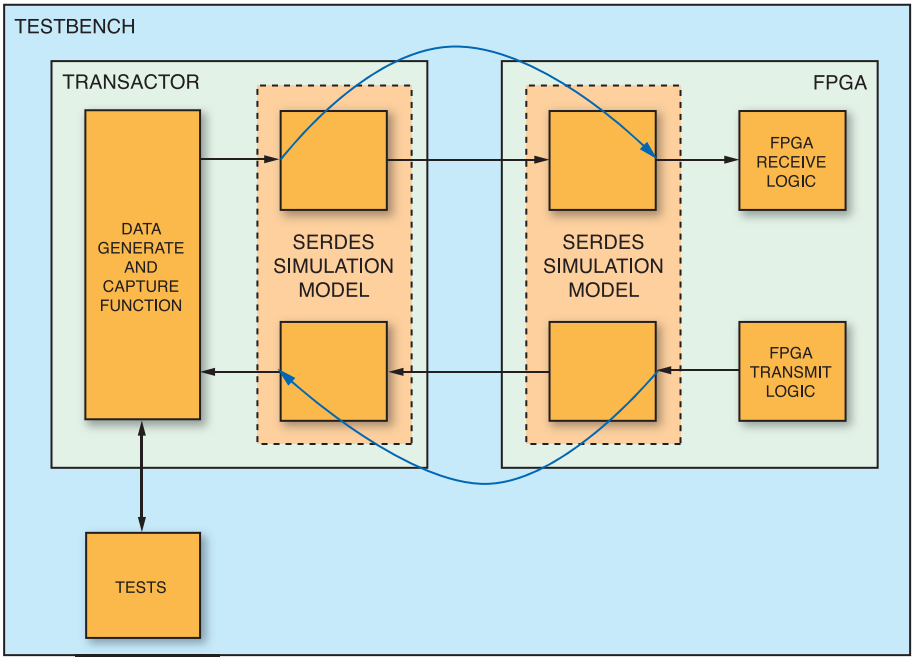


図 4 – デザイン内の SerDes を迂回したアプローチ

察するという方法です。図 4 に、このアプローチを示します。

このアプローチの長所は、複雑なシリアル トランザクターの作成が不要という点ですが、シリアル リンクの動的特性と機能の精度は犠牲になります。たとえばシリアライズ時間やデシリアライズ時間がコア側のロジックに与える影響や、デシリアライズ時に発生するエラーは適切にモデル化できません。ただし、図 1 に示したシンプルな 8b10b のデザイン例では、エンド ツー エンドでパラレル データを転送するためだけに SerDes を使用しており、シリアル リンク上を制御情報が流れていないため、このアプローチが単純明快なソリューションといえます。

一方、図 2 に示した XAUI デザインでこのアプローチを使用する場合、SerDes のコア側のコンフィギュレーションと I/O に関して豊富な知識が必要です。また、XAUI コア ロジックは GTP_DUAL SerDes の制御出力で同期状態が遷移することを前提としているため、送信データとカンマ文字の制御信号をレシーバーにループしただけでは不十分です。

このため、SerDes モデルの精度が非常に重要になります。簡略化したモデルで正常に動作するコア側の回路を作成しても、動作が正確な実際の物理デバイスに接続するとうまく動作しないことがあります。このように、コア側の信号の最小条件を迂回してロジック シミュレーションに合格するようなモデルを作成しても、完成した FPGA デバイスがラボで正しく動作するとは限りません。

インターフェイスの動作検証には 2 つの側面があります。1 つはコア側のロジックとインターフェイスの検証、そしてもう 1 つは SerDes の動作検証です。SerDes モデルを除外するこのアプローチでは、SerDes 自体のコンフィギュレーションやインプリメンテーションは検証できません。

図 1 のデザインでこのアプローチを使用するとシミュレーションが非常に高速に行え、テストは 1 秒以内で完了します。同じアプローチを図 2 の XAUI デザインで使用した場合は 18 秒かかります。モデル作成に要する時間は、8b10b デザインでは数分から数時間、XAUI デザインでは数日から数週間です。

アプローチ 2：テストベンチで 2 つの SERDES を使用する

次に紹介する FPGA SerDes デザインの機能検証アプローチは、テストベンチで FPGA SerDes シミュレーション モデルをトランザクターとして使用するという方法です。このアプローチでは、開発期間を短縮できますが、SerDes モデルによるシミュレータへの負荷が 2 倍となるため、シミュレーション 1 回当たりの実行時間が長くなります。

このアプローチは、基本的にはテスト対象のデザインでインスタンス化するのとまったく同じように SerDes モデルをトランザクター内にインスタンス化するという方法です（図 5 参照）。この方法だと、すべてのコンフィギュレーション オプションが維持されます。テストベンチでは、トランザクターのシリアル送信用出力をテスト対象デザインのシリアル受信用入力に接続し、テスト対象デザインのシリアル送信用出力をトランザクターのシリアル受信入力に接続します。

表 1 は、図 2 のデザインでシミュレーションを実行した場合のシミュレータ プロファイルをまとめたものです。このテストベンチは、シリアル側のスティミュラスとキャプチャした応答に対してビヘイビア トランザクターを使用しています。XAUI デザインでは、59 秒のシミュレーション時間のほとんどを SERDES が占めています。テスト

エンティティ	CPU 負荷の内訳（%）	シミュレーション時間（秒）
GTP_DUAL	71%	41.8
XAUI コア	11%	6.5
トランザクター	10.6%	6.3
テストベンチ	7.4%	4.4

表 1 – XAUI コアのシミュレーション プロファイル

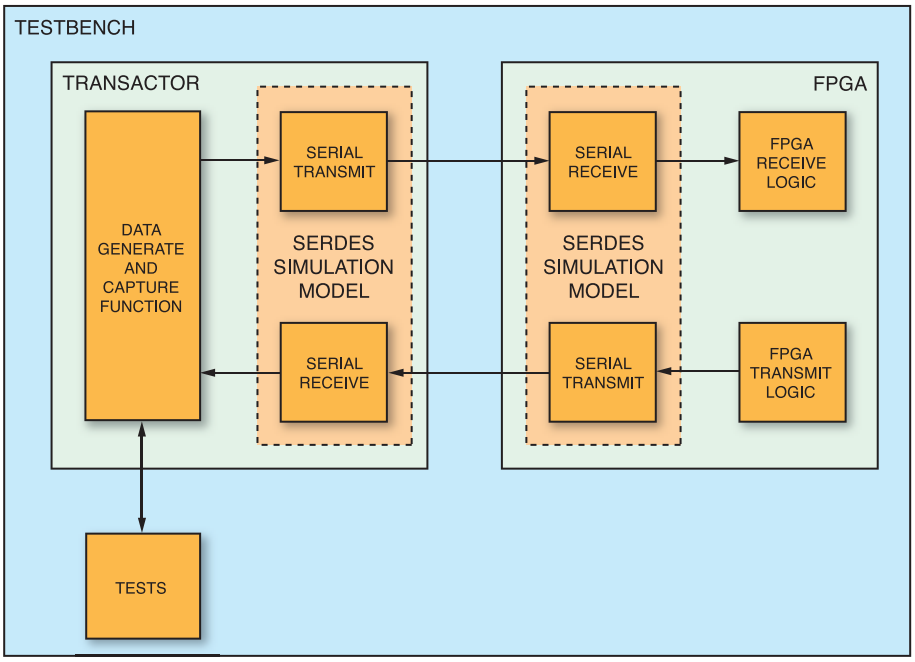


図 5 – テストベンチ内の SerDes の接続

アプローチ	開発時間	精度	シミュレーション速度
SerDes を除外	わずか	低	1 秒
テストベンチで SerDes を使用	短い	中	64 秒
ラボで検証	中程度	高	–
ネイティブ ハードウェアのエミュレーション	長い	高	1 秒
カスタム ビヘイビア モデルを作成	中程度	低	1 秒

表 2 – 8b10b における各アプローチの比較

アプローチ	開発時間	精度	シミュレーション速度
SerDes を除外	中程度	低	18 秒
テストベンチで SerDes を使用	短い	中	101 秒
ラボで検証	長い	高	–
ネイティブ ハードウェアのエミュレーション	長い	高	18 秒
カスタム ビヘイビア モデルを作成	中程度	低	59 秒

表 3 – XAUI における各アプローチの比較

ベンチでビヘイビア トランザクターの代わりに SerDes シミュレーション モデルを使用すると、シミュレータへの負荷が 2 倍になります。すると、1 回のシミュレーション全体にかかる時間が 59 秒から 101 秒になり、ほぼ 2 倍に増えてしまいます。図 1 の 8b10b デザインのシミュレータ プロファイルはさらに極端で、SerDes シミュレーション モデルがシミュレーション時間の 100% を占めています。この場合にテストベンチで 2 つの SerDes を使用すると、32 秒のシミュレーション時間が 2 倍の 64 秒となります。

アプローチ 3 : ラボで SerDes を検証する

FPGA SerDes デザイン検証のアプローチとして次に紹介するのは、ロジック シミュレーションで SerDes のシリアルとコア側の接続は検証せず、ラボで直接この部分を検証するという方法です。

このアプローチの長所は、リアルタイム システムで SerDes インターフェイスに大量のデータを転送できるという点です。一般に、シミュレーションのクロック周波数はフリー ランニングの物理的な FPGA デバイスと比較して 5 ～ 7 桁遅くなります。したがって、シリコンをラボで検証すると、テスト サイクルはソフトウェア シミュレーションに比べて数桁多くなります。しかし、ラボ環境では高周波数のシリアル信号やコア側のロジックの不透明な部分を検出できないため、デバッグは非常に難しくなります。

アプローチ 4 : ネイティブ ハードウェアで検証する

次に紹介する検証アプローチは、FPGA デザインと周辺回路を組み込んだエミュレーション プラットフォームを構築し、これにスティミュラスを与えて応答を観察するというものです。テスト対象の FPGA を含む特定用途向けのハードウェア プラットフォームをロジック シミュレータまたはソフトウェアに接続してパターンを提供し、FPGA デザインからの応答を確認します。

この方法では、既存の RTL テスト向けにシミュレータへのシームレスなインターフェイスを定義および生成するか、またはテストおよびトランザクターから FPGA デザインを切り離す代わりに既存の RTL テストベンチを破棄する必要があります。

ソフトウェア シミュレータ内部で動作する回路の同期性は厳密に維持されます。ハードウェア記述言語 (HDL) のビヘイビアとスケジューリングのルールは厳しく定義されているため、ソフトウェアのみのデザインやシミュレーションの予測性および生産性が向上します。しかし、ロジック シミュレータとハードウェア プラットフォームを接続し、ハードウェアおよびシミュレーション済みの HDL 間でセットアップとホールドの関係を維持するのは容易ではありません。

同様に、デザインをテストベンチから分離して検証するとなると、設計工程が非常に複雑になります。典型的な FPGA 検証フローは、まずシンプルなブロック レベルの機能検証から開始して、フルチップ機能のテストへ進みます。デザインをテストベンチから分離することは、一般的な FPGA 検証の環境とはいえません。

どちらの場合においても、SerDes デザインの動作をエミュレートするにはカスタム ハードウェア プラットフォームを構築する必要があります。デバイスの種類やベンダー、あるいはピン配置の変更など、少しでもデザインに変更があると PCB を配線し直す必要があるため、同じプラットフォームを再利用することはできません。たとえば、図 1 のデザイン用に構築したハードウェア プラットフォームは図 2 のデザインに再利用できません。これは、両者で使用している SerDes シリアル接続が大きく異なるためです。

図 2 の XAUI デザインを FPGA エミュレーション システムで実行すると、SerDes がシリコン精度で動作し、シミュレーション時間も 59 秒から 18 秒に短縮されます。ただし、ロジック シミュレータと結合した

柔軟な特定用途向けハードウェア エミュレーション ソリューションを FPGA デザイン プロジェクトごとに作成するのは非常に骨の折れる作業です。幸い、EDA ツール業界からさまざまな支援ツールが提供されており、筆者の所属する GateRocket 社もデザインに依存しない RocketDrive という定評あるソリューションでこうした機能を提供しています。この記事で紹介しているデータも、このソリューションを用いて得たものです。

アプローチ 5 : カスタム ビヘイビア モデル

アプローチ 1 で説明したように、SerDes シミュレーション モデルを簡略化したモデルで置き換えると動作が不正となる場合があります。そこで、デザインで使用する特定のモードに合わせてビヘイビア モデルを作成するという手法がよく用いられます。SerDes のすべての機能をシミュレーション モデルにインプリメントするのではなく、一部のパラメーター設定やポート接続のみを選択してインプリメントすることで、モデル開発の全体的な規模を抑えることができます。

このようにして作成されたモデルは、パラメーターとポート接続が同じであれば新規デザインで再利用できます。たとえば図 1 に示すようなシンプルな 8b10b デザインの場合、簡単な 8b10b エンコーダー / デコーダーのみでモデルを作成できます。この単純な動作にのみ対応する SerDes デザインであればこのシミュレーション モデルを再利用できます。

ただし、このようなモデルはデザインの機能が変わると再利用できません。新しいデザインで別の SerDes 機能を使用する場合は、モデルを作成し直す必要があります。

このアプローチのよい例が、XAUI デザイン用にザイリンクス CORE Generator で作成したテストベンチです (図 2)。ここでは、XAUI プロトコルに特化したビヘイビア シリアル トランザクターがインプリメントされています。このため、図 1 のように非常にシンプルな 8b10b デザインを含め、ほかのデザインではこのモデルを再利用できません。

テストベンチでは、ベンダーが提供する高精度な SerDes モデルをデザインで使用し、特定用途向けの SerDes ビヘイビア モデルをトランザクターで使用することにより、シミュレーションが 59 秒で完了しています。

アプローチの比較

この記事で提案した 5 つのアプローチを図 1 の 8b10b デザインと図 2 の XAUI デザインに分けてそれぞれ比較しました。その結果を、表 2 と表 3 にまとめています。これらの結果から、推奨されるアプローチを探ることができます。

たとえば、2 番目のアプローチ (テストベンチで 2 つの SerDes を使用) が適しているのは、シミュレーション時間全体に占める SerDes モデルの割合が小さい場合、およびテストの数が非常に少ない場合に限られます。シミュレーション プロファイルで SerDes モデルに費やす時間の割合が大きい場合は、シミュレーション時間が非常に長くなります。

ここで紹介した検証アプローチは、複数を組み合わせて使用されることもあります。たとえば、SerDes モデルをシミュレーションから除外し (アプローチ 1)、SerDes 機能をラボで検証する (アプローチ 3) することができます。

ネイティブ ハードウェア エミュレーションを使用したアプローチでは、カスタム ビヘイビア モデルと同じパフォーマンスでシリコンと同等の動作精度が得られます。

一般に、ソフトウェア シミュレーション モデルは精度とパフォーマンスがトレードオフの関係にあります。ハードウェア ソリューションの場合は優れたパフォーマンスと精度が得られますが、開発期間は長くなります。

シミュレーション時間が長くなると検証工程でテスト サイクルを減らさざるを得ず、当然、機能検証漏れも多くなります。

評価ボードやターゲット システムを用いて SerDes デザインを検証する場合、これらボードやシステムの完成を待つ必要があり、スケジュールに遅れが発生するというリスクが高まります。

まとめ

SerDes ベースの FPGA を検証するプロセスは非常に複雑です。特に難しいのが、デザインのコア ロジックをデバッグして SerDes が正しくコンフィギュレーションされていることを確認すること、必要な機能をインプリメントできるコンフィギュレーションを 2730 通りの組み合わせから選ぶこと、そして SerDes モデルのシミュレーションまたはゲートレベル シミュレーションに要するシミュレーション時間の長期化に対応してデザインの検証精度を高めることです。

GateRocket 社では、検証プロジェクトの入念な計画と厳格な方針に基づいた実行を推奨していますが、現在使用しているツールでシミュレーション時間が非常に長い場合や、テストをスキップしなければスケジュールに間に合わない場合、プロジェクトのリスクが非常に高くなります。検証はどれだけ行っても行いすぎることはなく、ほとんどのプロジェクトで十分に実行されていないのが現状です。検証プロジェクトを計画する際は、十分な検証を行うために検証を 2 回繰り返すくらいの時間が確保できるかどうかを確認するとよいでしょう。重要なのは、SerDes ベース デザインの検証に適したツールを使用するということです。ASIC 設計チームがハードウェアによる検証を利用して設計時点での正確性を確実にするのと同様、ハードウェア エミュレーションによる検証アプローチは高度な SerDes ベースの FPGA デザインに多くの利点をもたらします。

GateRocket 社がインプリメントしたこのテクノロジの詳細は、http://www.GateRocket.com を参照してください。

参考文献

[1] ザイリンクス、Virtex-5 FPGA RocketIO™ GTP トランシーバー ユーザー ガイド (2008 年 12 月)
http://japan.xilinx.com/support/documentation/virtex-5_user_guides.htm

[2] ザイリンクス、XAUI ユーザー ガイド (2010 年 4 月)
japan.xilinx.com/support/documentation/ip_documentation/xaui_ug150.pdf

[3] システム構成 : Intel Xeon CPU X3363 @ 2.83GHz、6MB キャッシュ、16GB DDR3 メイン メモリ (PC2-5300/667MHz)

Cortex-A9 ソフトウェアセミナー

ザイリンクス社 Zynq のCortex-A9MPCore ソフトウェア開発担当者必見セミナー

2012年 初夏開催予定

XILINX ZYNQ™

プログラム概要

1日目
- ARM アーキテクチャ
- ARM アセンブラプログラム
- 割り込みと例外
- NEON

2日目
- TrustZone
- キャッシュとメモリマネジメント
- Cortex-A9
- Cortex-A9 MP

XILINX

SOLUTION SUPPLIER PALTEK

先行ご案内ご希望の方はこちらまで
E-Mail：Info_pal@paltek.co.jp

FPGA 向け RTL 設計 スタイルガイドセミナー

FPGA設計に特化したRTL設計ノウハウと RTL設計スタイルガイドを解説

好評につき 第2回 開催決定

プログラム概要
- FPGA設計でのLVDS、PLL周辺回路作成ノウハウ
- 外部からの初期リセット入力とFPGA初期化
- 非同期クロックドメイン間の転送ノウハウ
- RTL記述ノウハウ
- 演算の記述方法
- テストベンチ記述の基本
- 期待値照合とランダム、アサーション

hdLab

SOLUTION SUPPLIER PALTEK

日時：2012年4月13日(金)10:00～18:00
場所：PALTEK新横浜本社
参加対象者：
・RTL設計における記述スタイルを身につけたいFPGA設計者
・更なるステップアップを求める初中級者

http://www.paltek.co.jp/seminar/extra/fpga/1216FPGA_RTL.htm

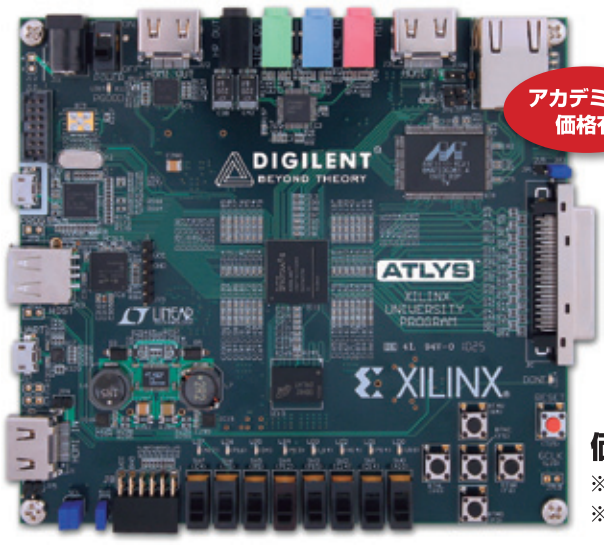
PALTEK セミナー 検索

PALTEK BOARD SELECTION

PALTEK Board 検索

DIGILENT® BEYOND THEORY

FPGA初心者に人気
Spartan-6 LX45 搭載 FPGA 評価ボード



アカデミック価格有

価格:¥38,700-（税抜）
※輸入運賃含む
※教育機関・研究機関様には、別途アカデミック価格有

FPGAボードはPALTEKへ

ウェブにて最新情報掲載中
http://www.paltek.co.jp/board/index.htm

株式会社PALTEK

■新横浜本社　横浜市港北区新横浜2-3-12　TEL.045-477-2005
URL　http://www.paltek.co.jp/　E-mail　info_pal@paltek.co.jp
■西日本支社　大阪府吹田市江坂町1-14-33 大町ビル5　TEL.06-6384-2281

SOLUTION SUPPLIER PALTEK

How Do I Reset My FPGA?

FPGA のリセット方法

デザインの集積度、性能、消費電力を改善する最適なリセット回路

E. Srikanth
Solutions Development Engineer
Xilinx, Inc.
serusal@xilinx.com

FPGA デザインでは、リセットはすべての記憶素子を既知の状態にセットする同期信号としての役割を果たします。通常、デジタル デザインではグローバル リセットは外部ピンとしてインプリメントされ、このピンで電源投入時にデザインを初期化します。グローバル リセット ピンはほかの入力ピンと同様に、FPGA に対して非同期的に駆動されるのが普通です。設計者は、この信号を使用して FPGA 内部でデザインを非同期的にリセットするか同期的にリセットするかを選択します。

リセット回路の構造は FPGA デバイスの使用率、タイミング、消費電力に影響しますが、ここで紹介するいくつかのポイントを押さえれば、最適な構造を選択できるようになるでしょう。

フリップフロップのリセット動作を理解する

リセット方法を詳しく見ていく前に、FPGA スライス内のフリップフロップの動作について理解しておきましょう。ザイリンクス 7 シリーズ アーキテクチャのデバイスは、1 スライスに 8 つのレジスタがあります。これらのレジスタはすべて D 型フリップフロップで、制御信号を共有しています。

フリップフロップの制御信号にはクロック入力 (CLK)、アクティブ High のチップイネーブル (CE)、アクティブ High の SR ポートがあります。フリップフロップの SR ポートは、同期セット / リセットまたは非同期プリセット / クリア ポートとして機能します (図 1 参照)。

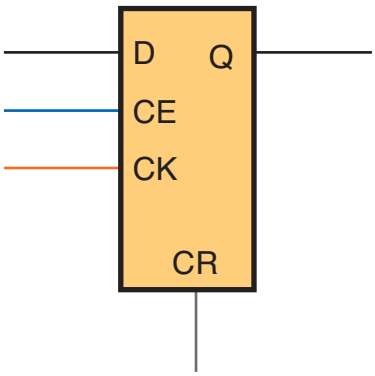


図 1 – スライスのフリップフロップ制御信号

フリップフロップを推論する RTL コードは、フリップフロップが使用するリセットの種類についても推論します。RTL コード内で、RTL プロセスのセンシティビティリストにリセット信号が含まれている場合、コードは非同期リセットを推論します (図 2a)。この場合、合成ツールは、SR ポートをプリセットまたはクリア ポートとして構成したフリップフロップ (FDCE または FDPE フリップフロップ プリミティブとして表わされる) を生成します。SR ポートをアサートすると、フリップフロップ出力はただちにフリップフロップの SRVAL 属性の値となります。

同期リセットの場合、合成ツールは SR ポートをセットまたはリセット ポートとして構成したフリップフロップ (FDSE または FDRE フリップフロップ プリミティブとして表わされる) を推論します。SR ポートをアサートすると、フリップフロップ出力は次の立ち上がりクロック エッジでフリップフロップの SRVAL 属性の値となります。

また、フリップフロップ出力を INIT 属性で指定した値に初期化することもできます。INIT の値は、コンフィギュレーション時およびグローバル セット リセット (GSR) 信号のアサート時にフリップフロップにロードされます。

ザイリンクス FPGA のフリップフロップは非同期および同期のリセット / セット制御信号をどちらもサポートしていますが、元になるフリップフロップでネイティブにインプリメントできるのは、セット / リセット / プリセット / クリアのいずれか 1 つに限られます。RTL コードで複数のセット / リセット / プリセット / クリア状態をコーディングすると、1 つの状態がフリップフロップの SR ポートを使用してインプリメントされ、それ以外がファブリック ロジックでインプリメントされるため、FPGA リソースの使用量が多くなります。

```
signal Q:std_logic:='1';
....
async: process (CLK, RST)
begin
  if (RST='1')then
    Q <= '0';
  else
    if (CLK'event and CLK='1') then
      Q <= D;
    end if;
  end if;
end process async;
```

(a)

INIT

SRVAL

```
signal Q:std_logic:='1';
....
sync: process (CLK)
begin
  if (CLK'event and CLK='1') then
    if (RST='1') then
      Q <= '0';
    else
      Q <= D;
    end if;
  end if;
end process sync;
```

(b)

図 2 – SRVAL 属性と INIT 属性がフリップフロップのリセットと初期化を定義。VHDL コードによって (a) 非同期リセットおよび (b) 同期リセットを推論する。

これらの状態のうち 1 つが同期でその他が非同期の場合、非同期状態は SR ポートでインプリメントされ、同期状態はファブリック ロジックでインプリメントされます。通常は、複数のセット / リセット / プリセット / クリア状態を回避するようにしてください。また、フリップフロップの SR ポートが同期か非同期かは、スライス内の 4 つのフリップフロップのうちの 1 つの属性によって決定します。

リセットの方法

使用するリセットの種類 (同期か非同期か) にかかわらず、リセットはクロックに同期させる必要があります。グローバル リセット パルスが十分長く持続していれば、デバイスのすべてのフリップフロップがリセット状態に入ります。しかしリセット信号のディアサートがフリップフロップのタイミング条件を満たしていなければ、フリップフロップはリセット状態から通常の状態に正しく遷移せず、メタステーブル状態となります。

さらに、ステート マシンやカウンターなど一部のサブシステムを正しく動作させる

には、すべてのフリップフロップを同じクロック エッジでリセット状態から解除する必要があります。1 つのステート マシンがビットによって異なるタイミングでリセット解除された場合、ステート マシンが不正な状態に遷移する可能性があります。このことからも、リセットのディアサートをクロックに同期して行うことが重要であることがわかります。

あるクロック領域で同期リセットを使用するデザインでは、標準的なメタスタビリティ防止回路（連続する 2 つのフリップフロップを接続したもの）を用いてグローバル リセット ピンを特定のクロック領域に同期させるだけで十分です。この同期したリセット信号は、フリップフロップの同期 SR ポートを使用してそのクロック領域のすべての記憶素子を初期化します。同期回路とリセット対象のフリップフロップはどちらも同じクロック領域にあるため、標準のクロック制約 PERIOD でこれら 2 つの間のパスのタイミングに対応できます。デバイス内の各クロック領域は、それぞれ専用の同期回路を使用してそのクロック領域に同期したグローバル リセットを生成する必要があります。

では、ここからは実際のデザインに最適なリセット方法の見つけ方について具体的に説明します。

アドバイス 1：フリップフロップの同期 SR ポートを駆動する場合、すべてのクロック領域でそのクロック領域に同期したローカルなグローバル リセットが必要です。

デザインの一部で有効なクロックが保証されていないことがあります。これは、リカバリ クロックを使用するシステムや、ホットプラグ可能なモジュールからクロックを供給するシステムで起こります。このような場合、デザイン内の記憶素子をリセットするにはフリップフロップの非同期 SR ポートを用いて非同期リセットを行う必要があります。記憶素子は非同期 SR ポートを使用しますが、リセットをディアサートするエッジはクロックに同期していなければなりません。具体的には、フリップフロップのリセットからリカバリまでのタイミングアークの条件を満たす必要があります。これは、非同期 SR のディアサート エッジからクロックの立ち上がりエッジまでのセットアップ条件と似ています。このタイミングアークの条件を満たすことができなければ、フリップフロップがメタステーブル状態となり、同期サブシステムが予期しない状態になることがあります。

図 3 に示したリセット ブリッジ回路を使用すると、リセットを非同期的にアサートし（したがって有効なクロックがない場合でも正しい動作が可能）、同期的にディアサートできます。この回路では、2 つのフリップフロップの SR ポートの機能が非同期プリセット（SRVAL=1）であると仮定しています。

このリセット ブリッジ回路の出力を使用して、特定のクロック領域の非同期リセットを駆動できます。このように同期したリセットは、フリップフロップの非同期 SR ポートを使用することによりクロック領域内にあるすべての記憶素子を初期化できます。ここでも、デバイス内の各クロック領域は、個々のリセット ブリッジによって生成された、その領域のクロックに同期するグローバル リセットが必要です。

アドバイス 2：リセット ブリッジ回路は、クロックに同期して非同期リセットをディアサートさせるための安全メカニズムの役割を果たします。すべてのクロック領域には、リセット ブリッジ回路を使用してそのクロック領域のローカル クロックに同期したグローバル リセットが必要です。

図 3 の回路は、リセット ブリッジおよび関連するロジックに供給されるクロック (clk_a) が安定しておりエラーがないことを前提としています。FPGA では、クロックはオフチップのクロック ソース（理想的には Clock-Capable ピンを経由）から直接供給される場合と、MMCM または PLL (Phase-Locked Loop) を使用して内部で生成される場合があります。MMCM または PLL を使用してクロックを生成した場合、これらの MMCM や PLL にはリセット後にキャリブレーションが必要です。このため、クロックを安定させるためにグローバル リセットにロジックを追加しなければならないことがあります。

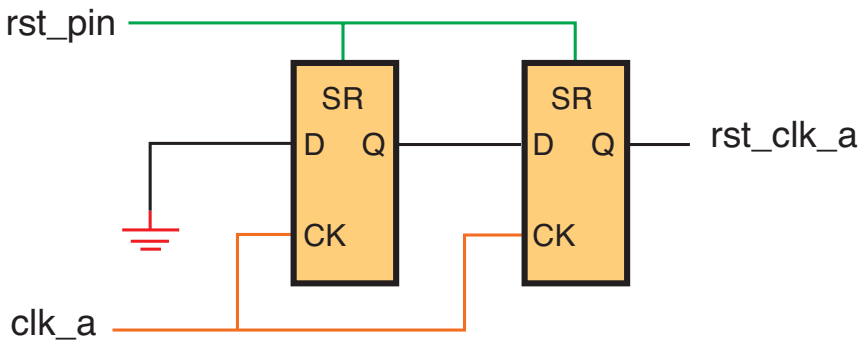


図 3 – 非同期的にアサートし、同期的にディアサートするリセット ブリッジ回路

アドバイス 3：MMCM または PLL でクロックを生成した場合、FPGA に対してグローバル リセットをディアサートする前にクロックが安定してロック状態であることを確認します。

図 4 に、FPGA の典型的なリセットのインプリメンテーションを示します。

ザイリンクス FPGA のレジスタの SR 制御ポートはアクティブ High です。RTL コードでアクティブ Low のセット / リセット / プリセット / クリア機能が記述されている場合、合成ツールはレジスタの制御ポートを直接駆動する前にインバーターを推論します。このインバーターにはルックアップ テーブルが必

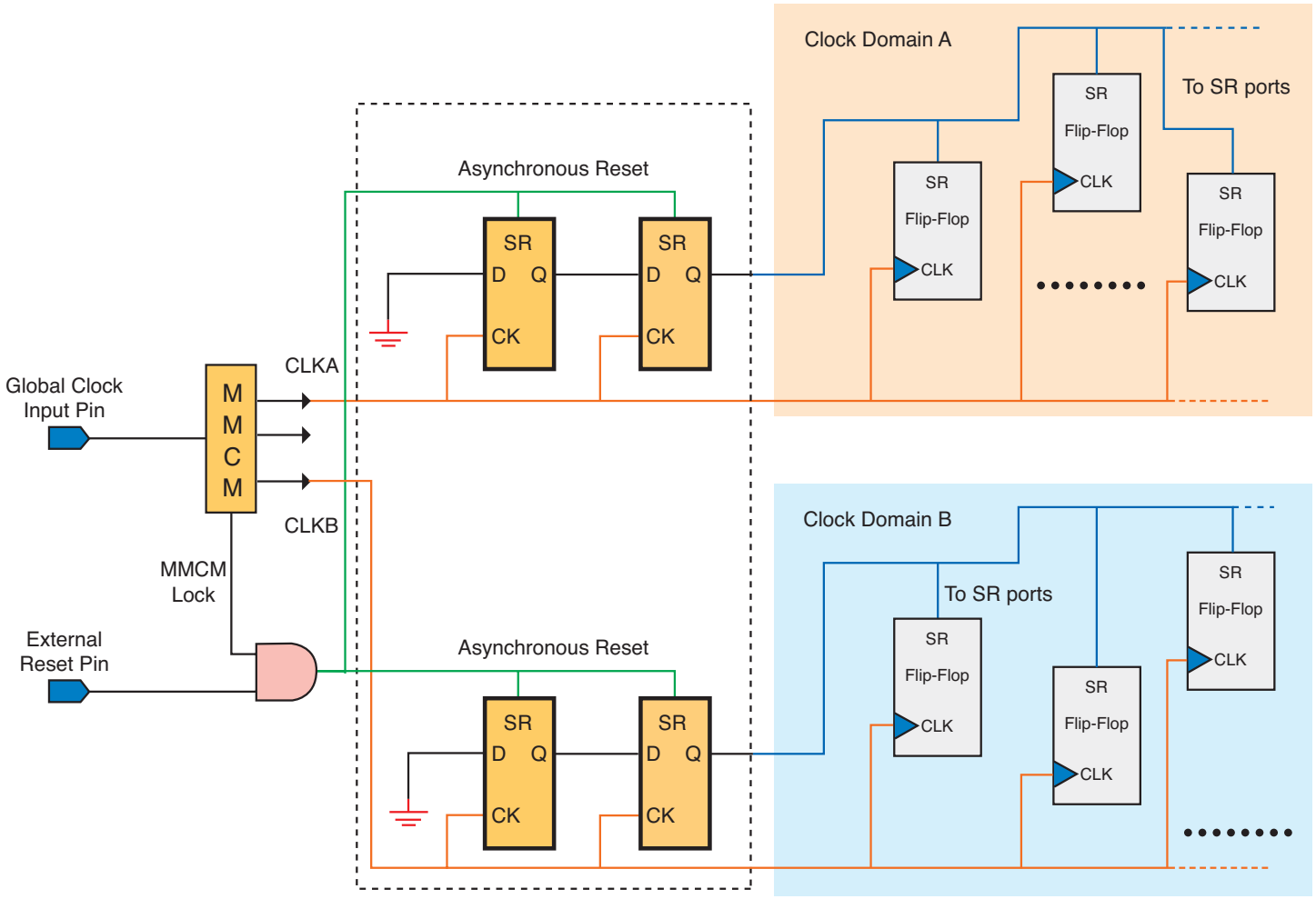


図 4 – FPGA での典型的なリセットのインプリメンテーション

要になるため、LUT 入力が 1 つ占有されます。アクティブ Low の制御信号によって余分なロジックが推論されると、実行時間が長くなるだけでなくデバイス使用率も低下します。また、タイミングと消費電力にも影響が及びます。

結論として、HDL コードやインスタンシエートされたコンポーネントではなるべくアクティブ High の制御信号を使用するようにします。デザイン内で制御信号の極性を制御できない場合は、コードの最上位階層で信号を反転しておく必要があります。このように記述しておくと、インバーターが推論されても I/O ロジックに吸収されるため、FPGA ロジックや配線リソースを余分に消費することがありません。

アドバイス 4：アクティブ High のリセットでは、デバイス使用率とパフォーマンスがより向上します。

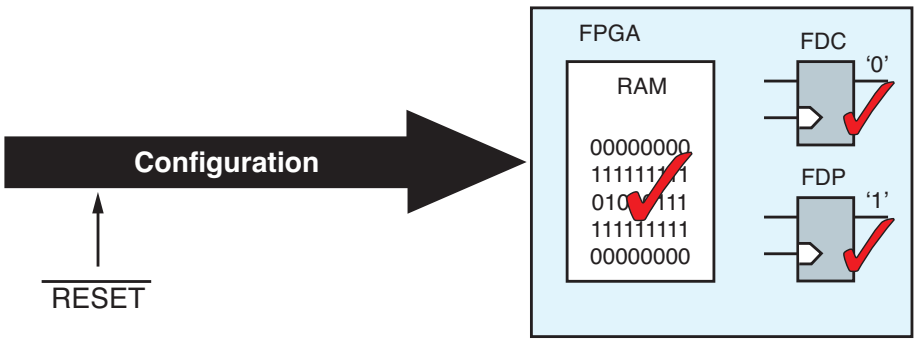


図 5 – コンフィギュレーション後の FPGA の初期化

```
signal reg: std_logic_vector (7 downto 0) := (others <= '0');
....
process (clk) begin
    if (clk'event and clk= '1') then
      if (rst= '1') then
         reg <= '0';
      else
         reg <= D;
      end if;
    end if;
end process;
```

図 6 – RTL コードでの信号初期化 (VHDL)

FPGA には必ずしもグローバル リセットは必要ありません。グローバル リセットは、デザイン内のほかのネットで使用する配線リソースと競合します。また、デザイン内のすべてのフリップフロップに伝搬する必要があるため、一般にファンアウトが大きくなります。このため非常に多くの配線リソースを消費し、デバイス使用率とタイミング パフォーマンスに影響を与えることがあります。したがって、完全なグローバル リセットは使用せず、ほかのリセット方法を検討することも大切です。

ザイリンクス FPGA のコンフィギュレーションまたはリコンフィギュレーションを実行すると、すべてのセル (フリップフロップとブロック RAM を含む) が図 5 のように初期化されます。このため、FPGA コンフィギュレーションには FPGA 内のすべての記憶素子の初期状態を既知の状態にセットするという点で、グローバル リセットと同じ効果があります。

フリップフロップの初期化値は RTL コードから推論できます。図 6 に示した例は、RTL コードでのレジスタの初期化方法を示したものです。一般に、FPGA ツールは信号の初期化を合成できないと考えられていますが、それは誤解です。元になる signal (VHDL の場合) または reg (Verilog の場合) の初期化値が、推論されたフリップフロップの INIT 値となり、この値がコンフィギュレーション時にフリップフロップにロードされます。

レジスタの場合と同様、ブロック RAM もコンフィギュレーション時に初期化できます。プロセッサ ベース システムで使用するエンベデッド RAM の増大に伴い、BRAM の初期化が有用な機能となっています。その理由は、あらかじめ定義された RAM によってシミュレーションのセットアップが簡単に行え、エンベデッド デザインのメモリをクリアするためのブートアップ シーケンスを用意する必要がなくなるからです。

グローバル セット リセット (GSR) 信号は、FPGA のコンフィギュレーションの間、デザインを初期状態に保持する特別な配線済みリセット信号です。コンフィギュレーションが完了すると GSR は解除され、フリップ

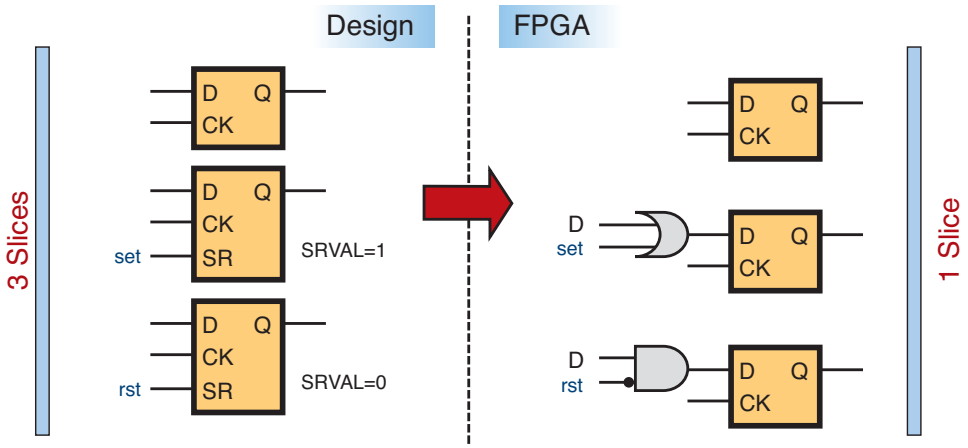


図 7 – SR に対するコントロール セットの削減

フロップおよびその他のリソースがすべて INIT 値となります。GSR 信号を利用できるのはコンフィギュレーション時だけではありません。STARTUPE2 モジュールをインスタンシエートして GSR ポートに接続することで、ユーザー デザインから GSR ネットを利用することもできます。このポートを使用して、デザインが GSR ネットを再度アサートすると、FPGA 内のすべての記憶素子がそれぞれの INIT 属性で指定した状態に戻ります。

GSR のディアサートは非同期であり、デザイン内のすべてのフリップフロップに到達するには数クロックを要します。ステート マシンやカウンターなど、自律的に状態が変化するロジックには、ユーザー クロックに同期してディアサートする明示的なリセットが必要です。このため、GSR を唯一のリセット手段として使用すると、システムが不安定になります。

このような場合は、ハイブリッド アプローチを採用してスタートアップを適切に管理できるようにする必要があります。

アドバイス 5：デザインの一部が自律的に開始するような場合、その部分には明示的なリセットを使用して GSR による内臓型の初期化メカニズムと組み合わせると、デバイスの使用率とパフォーマンスが向上します。

GSR を用いてデザイン全体を初期状態にした後、非同期リセットが必要なステート マシンなどのロジック素子には明示的なリセットを使用します。この場合、標準的なメタスタビリティ防止回路またはリセット ブリッジを使用してクロックに同期した明示的なリセットを生成します。

デバイスの使用率を最大にするリセット方法

RTL コードで記述したリセットのスタイルによって、内在する FPGA リソースにデザインをマッピングする際の効率が大きく異なります。RTL コードを作成する際は、合成ツールが FPGA のリソースに効率よくマッピングできるようにサブデザインのリセット方式を適切に記述する必要があります。

SRL、LUTRAM、ブロック RAM を初期化できるのは GSR によるリセットのみで、明示的リセットは使用できません。したがって、これらリソースにマッピングするコードを作成する際は、リセットを使用しないでコーディングを行う必要があります。たとえば 32 ビット シフト レジスタの RTL コードで、シフト レジスタの 32 ステージに対して明示的なリセットを記述した場合、コーディングしたリセットの条件をこのリソースでは満たすことができないため、合成ツールはこの RTL コードを SRL32E に直接マッピングできません。必要なリセット回路をインプリメントするには、32 個のフリップフロップを推論するか、または SRL32E の周囲に余分な回路を推論するかのどちらかが必要です。いずれにしても、必要なリソース量はリセットなしで RTL をコーディングした場合よりも多くなります。

アドバイス 6：SRL、LUTRAM、ブロック RAM にマッピングする場合は、SRL または RAM アレイのリセットをコーディングしないこと。

7 シリーズ デバイスでは、フリップフロップを別の制御信号と同じスライスにパックすることができません。低ファンアウトのリセットでは、このことが全体的なスライスの使用率に影響を与える可能性があります。同期リセットの場合、合成ツールはフリップフロップの制御ポートではなく LUT（図 7）を使用してリセット機能をインプリメントできるため、制御ポートとしてのリセットを排除できます。このようにして合成した LUT/ フリップフロップは、SR ポートを使用しないほかのフリップフロップと一緒にパックされます。この場合、LUT の使用量は増えますが、スライスの使用率は向上します。

アドバイス 7：同期リセットを使用すると FPGA 使用率が向上するため、非同期リセットよりも同期リセットを使用するようにします。

一部の大規模な専用リソース（ブロック RAM と DSP48E1 セル）には、専用リソースの機能の一部として推論可能なレジスタが含まれます。ブロック RAM のオプションの出力レジスタを用いると、遅延クロックを追加してクロック周波数を高めることができます。また、DSP48E1 セルにもパイプライン化による最大クロック周波数の向上や、サイクル遅延（Z-1）のために使用できるレジスタが数多くあります。ただし、これらのレジスタには同期セット / リセットの機能しかありません。

アドバイス 8：同期リセットを使用すると、合成ツールは DSP48E1 スライスやブロック RAM などの専用リソース内のレジスタを利用できます。これにより、全体的なデバイス使用率、そしてデザインの一部の性能が向上します。また、全体的な消費電力も削減されます。

RTL コードで非同期セット / リセットを記述した場合、合成ツールはこれらの内部レジスタを使用できません。その代わりに、要求された非同期セット / リセット機能をインプリメントできる、スライスのフリップフロップを使用します。これはデバイス使用率の悪化を招くだけでなく、性能と消費電力にも影響を与えます。

多くの選択肢

FPGA のリセット方法にはさまざまな種類があり、それぞれに長所と短所があります。ここで紹介したアドバイスを参考に、各デザインに最も適したリセット構造を選択してください。最適なリセット構造を選択することで、FPGA のデバイス使用率、タイミング、消費電力が改善します。

紹介したアドバイスの多くは、トレーニング コース「7 シリーズ ファミリ デザイン」でも説明しています。ザイリンクスのトレーニング コースの詳細は、japan.xilinx.com/training をご覧ください。

筆者紹介

Srikanth Erusalagandi

ザイリンクスのグローバル トレーニング ソリューション チームのソリューション開発エンジニアとして、ザイリンクスのトレーニング コースのコンテンツ制作を担当。専門は FPGA デザインとコネクティビティ。2010 年 1 月にザイリンクス入社。以前は MosChip Semiconductors 社で約 6 年間アプリケーション エンジニアとして活躍。

Demystifying FPGAs for Software Engineers

ソフトウェア エンジニアリングにおける FPGA の誤解と真実

Glenn Steiner
Senior Manager of Technical Marketing
Xilinx, Inc.
glenn.steiner@xilinx.com

Dan Isaacs
Director of Technical Marketing
Xilinx, Inc.
dan.isaacs@xilinx.com

FPGA エンベデッド プロセッサ用ソフトウェアの開発に役立つ実践的ヒントを紹介

製品の複雑化が進むにつれ、ASSP (APPLICATION-SPECIFIC STANDARD PRODUCT) などの集積回路を使用しなければデザインの要件を満たすことが難しくなっています。エンジニアは従来、プロセッサ、メモリ、ペリフェラルなどの単体部品を選んで、これらをディスクリート ロジックで接続していましたが、最近は ASSP の一覧からシステム要件に最も合ったプロセッシング システムを選択する方法が一般的です。ロジックやペリフェラルを追加する必要がある場合は、ASSP と FPGA を組み合わせてソリューションを完成させるという手法がとられています。実際、今日ではエンベデッド システム全体の 50 ～ 70% に FPGA が使用されているという調査報告もあります。

ここ数年の傾向として FPGA の大容量化が進み、プロセッサとロジックを備えた完全なシステムを 1 個の FPGA に集積できるようになりました。これを受けて、ソフトウェア エンジニアが FPGA 内部のエンベデッド プロセッサ用コードの開発とデバッグを手がける機会が増えています。中には FPGA エンベデッド プロセッサに苦手意識を持つソフトウェア エンジニアもいますが、FPGA の基礎を理解し、FPGA エンベデッド プロセッサ用コードの作成およびデバッグ方法を把握してしまえば、懸念する必要はありません。

FPGA とは

FPGA (Field-Programmable Gate Array) は、製造後にフィールドでコンフィギュレーションや接続を変更できるロジックを内蔵した IC です。以前は、エンジニアがさまざまなロジック デバイスをカタログから選んで購入し、これらをプリント基板 (PCB) 上で接続してロジック デザインを設計していましたが、現在は 1 個のデバイスの内部に完全なデザインをインプリメントできるようになっています。基本的に、FPGA は次の要素で構成されます。

- AND、OR、NOT など多くのロジック回路で構成されたコンフィギャラブル ロジック ブロック (CLB)
- ロジック ブロックを相互接続するコンフィギャラブル インターコネクト
- I/O インターフェイス

これらの要素を使用して、任意のロジック デザインを作成します。

通常、ハードウェア エンジニアは HDL (一般には Verilog または VHDL) でコードを記述し、これをコンパイルして作成したオブジェクト ファイルをデバイスにロードして実行します。一見すると、HDL プログラムは C などの高級言語に類似しています。たとえば、以下に示したのは Verilog で記述した 8 ビット カウンターのインプリメンテーション例ですが (www.asic-world.com/ 提供)、多くの構文が現在の高級言語と共通していることがわかります。

```
//-----------------------------------------
// Design Name : up_counter
// File Name   : up_counter.v
// Function    : Up counter
// Coder       : Deepak
//-----------------------------------------
module up_counter    (
out     ,  // Output of the counter
enable  ,  // enable for counter
clk     ,  // clock Input
reset      // reset Input
);
//----------Output Ports--------------------
    output [7:0] out;
//------------Input Ports--------------------
     input enable, clk, reset;
//------------Internal Variables-------------
    reg [7:0] out;
//-------------Code Starts Here--------------
always @(posedge clk)
if (reset) begin
  out <= 8'b0 ;
end
 else if (enable) begin
  out <= out + 1;
end
endmodule
```

FPGA の技術的優位性

マスク費用の高騰により ASIC の使用が難しくなった今、FPGA はデータ処理システムをインプリメントする手段として柔軟性、性能、コスト パフォーマンスの面で最も優れた選択肢となっています。FPGA のアーキテクチャは非常に柔軟性が高く、ハードウェア設計者は並列化とパイプラインの両要素で構成された処理システムをインプリメントできます。このため、パフォーマンスとレイテンシの両方を最適化したシステム設計が可能です。通常、これらのデータ処理システムでは、汎用プロセッサを使用した場合と比較して低コストで高いパフォーマンスが得られます。

制御用に外部マイクロプロセッサを FPGA 内部のデータ処理システムと組み合わせて使用することも可能ですが、プロセッサを FPGA 内部にインプリメントした方が有利な点があります。まず、FPGA 内部にプロセッサを持つことでプロセッサとデータ処理システム間の制御遅延が飛躍的に減少し、多くのプロセッサ サイクルを削減できます。また、プロセッサとデータ処理システム間には 32 ビット以上の通信チャネルに加え、アドレス用と制御用の配線も必要ですが、外部プロセッサを使用した場合はこれらの配線のためにプロセッサと FPGA 両方のパッケージが大型化し、システム コストの上昇を招きます。ここで PCI EXPRESS® (PCIE®) を使用すればピン数は大幅に削減できますが、PCIE は比較的新しいインターフェイスであるためすべてのプロセッサや FPGA でサポートされているとは限りません。また、PCIE はパフォーマンス自体は高いものの、シリアル インターフェイスであるためプロセッサとデータ処理システム間の遅延は大きくなります。

データ処理システムとプロセッサの両方を FPGA 内部にインプリメントすれば、部品点数、基板面積、そして場合によっては消費電力も削減できます。このため、ソリューション全体として大幅な低コスト化が可能です。FPGA 内部には、ARM® CORTEX™ -A9 プロセッサのようなハードマクロとしてのプロセッサや、ザイリンクス MICROBLAZE™ プロセッサのようなソフト プロセッサをインプリメントできます。また、FPGA ベースのプロセッサはアプリケーションの要件に基づいたコンフィギュレーションも可能です。FPGA ベース システムでは、制御機能と計算機能の分担をプロセッサと FPGA ロジック間で自由に割り振りできるため、システム レベルで容易に調整できます。

インプリメンテーションの方法

FPGA エンベデッド処理システムをインプリメントする方法は複数ありますが、大きく分けると (1) まったく新しいシステムを構築する、(2) ウィザードを使用する、(3) 既存のデザインを一部修正する、という 3 つの方法があります。

(1) は、FPGA ツールを利用してリストから必要な IP を選び、これらをバスや配線で接続することでまったく新しい処理システムを構築するという方法です。この方法では効果的なシステム構築が可能ですが、長い時間がかかります。

(2) は、FPGA ツールのウィザード機能を使用する方法で、マイクロプロセッサの構築にかかる時間を大幅に短縮できます。ターゲット デバイスや必要なプロセッサおよびペリフェラルはドロップダウン リストやチェック ボックスから簡単に指定できます。図 1 は、ウィザードの開始画面とウィザードを利用して完成した最終システムを示したものです。同様に、MATLAB® ソフトウェアなどのツールを用いて制御用のプロセッサ バス インターフェイスを備えたデータ処理システムを構築することも可能です。プロセッサとデータ処理システムは、バス インターフェイスを一致させるだけで接続できます。

(3) は、既存のリファレンス デザインを一部修正したり、既存のハード プロセッサ システムを拡張してエンベデッド処理システムをインプリメントする方法です。最近は FPGA リファレンス デザインやハード プロセッサ システムの進歩が著しく、市場ニーズに焦点を合わせたものが増えています。これらのデザインは非常に完成度が高く、別途コンポーネントを追加しなくてもそのまま使用できることも少なくありません。多くの場合、これらのリファレンス デザインには完全なドライバーと構築済みの OS が付属しています。

(1) と (2) の方法はどちらもプロセッサ システムを効果的に構築

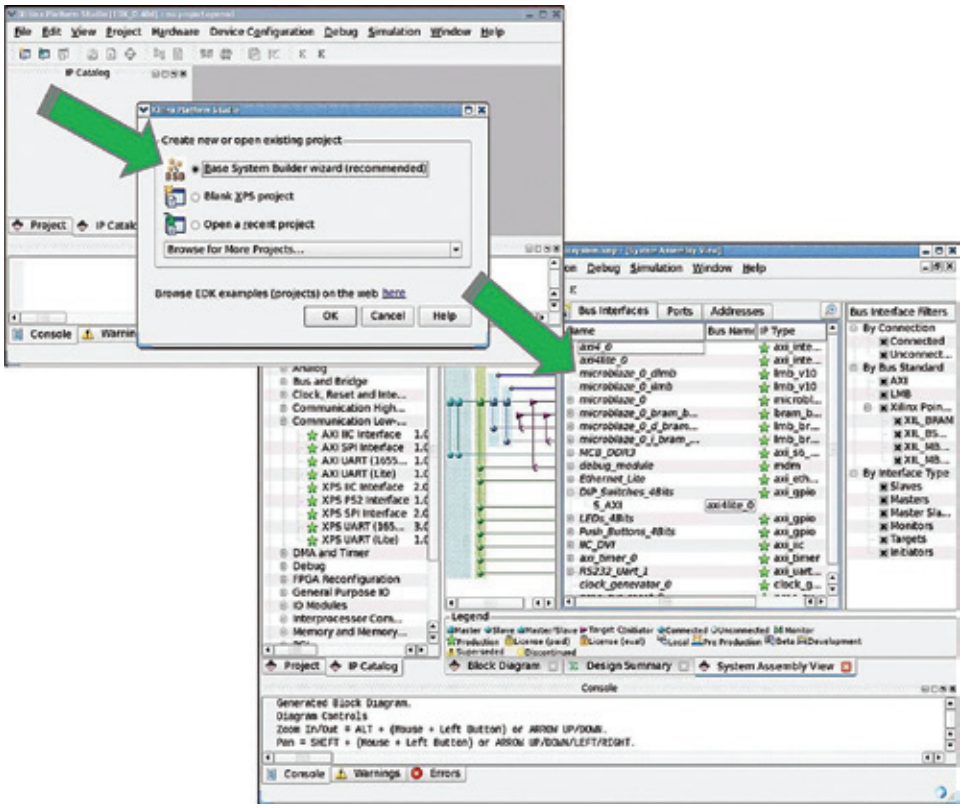


図 1 – ウィザードの開始画面と完成したシステム

ディターで hello world プログラムを作成した様子を示したものです。

誤解：FPGA には ASSP ライクなプロセッサ システムがない。

真実：現在、ASSP ライクなペリフェラル セットを備えた既製の FPGA ソフト エンベデッド プロセッサ デザインやハード プロセッサ デザインが提供されています。

FPGA の機能はソフト プロセッサやハード プロセッサを搭載することで大きく拡張されます。FPGA エンベデッド ソフト プロセッサのリファレンス デザインには、32 ビット RISC プロセッサ、メモリ インターフェイス、業界標準のペリフェラルが用意されています。これらのプロセッサは柔軟性が非常に高く、最新の OS をサポートする MMU を追加するなど、ロジックとパフォーマンス機能をユーザーが自由にトレードオフできます。FPGA の選択肢は非常に幅広く、プロセッサの構成、ペリフェラル、データ処理ロジック、ロジックのパフォーマンスなどはシステム要件に合わせてユーザーがさまざまなレベルから選択できます。

できます。一方、(3) は既存の検証済みリファレンス デザインを利用して作業を開始するため、ハードウェアおよびソフトウェア エンジニアは開発期間を大幅に短縮できます。

誤解を解く

エンジニアの大半が、FPGA 内のエンベデッド プロセッサ用にコードを開発するのは難しいと考えていますが、そこには多くの誤解が存在します。ここからは、これらの誤解を 1 つずつ解いていきたいと思います。

誤解：FPGA エンベデッド プロセッサ用のコードを作成するのは難しい。

真実：FPGA エンベデッド プロセッサ用のコード開発は、現在主流のソフトウェア開発環境で C または C++ を使用して行えることがほとんどです。

最近は、FPGA サプライヤーの多くが Eclipse を利用したソフトウェア開発をサポートしています。Eclipse はプラグインに対応した柔軟なソフトウェア開発環境で、テキスト エディター、コンパイラ、リンカー、デバッガー、トレース モジュール、コード管理などの機能が標準で用意されています。

Eclipse はオープンな環境であり、大規模な開発者コミュニティによって日々新しい機能が追加されています。たとえば、コーディングに使用するエディターも標準のものが使いにくければ、好みに合ったものをインストールできます。図 2 は Eclipse のコード エ

また、ASSP ライクな既製のリファレンス デザインを利用すれば、ソフトウェア開発者はハードウェア エンジニアがプロセッサ システムのインプリメンテーションを完成させる前にコーディングを開始できます。既製のデザインだけでエンベデッド プロセッサのシステム要件が完全に満たされることも多く、その場合、ハードウェア エンジニアはさらなるプロセッサ システム設計から解放されます。そうでない場合も、既製のデザインをベースにしてペリフェラルを追加したり、カスタム ハードウェア アクセラレータを接続するだけでエンベデッド プロセッサ システムの設計が完了します。

誤解：FPGA エンベデッド プロセッサ用のコードはデバッグが難しい。

真実：FPGA エンベデッド プロセッサのソフトウェア デバッグは、エンベデッドではない通常のプロセッサの場合と同様に簡単に実行できます。デバッガーには、コードのダウンロード、プログラムの実行、ソース コードおよびオブジェクト コード レベルでのステップ実行、ブレークポイントの設定、メモリとレジスタの検証などの機能があります。その他、コードのプロファイリングやトレースを行うためのツールも用意されています。

誤解：使用したい OS がサポートされていない。

真実：一般的なエンベデッド プロセッサではほとんどの OS がサポートされており、そのサポート数は現在も増え続けています。ザイリンクスの MicroBlaze は Linux、ThreadX、MicroC/OS-II、eCos をはじめ、数多くの OS をサポートしています。

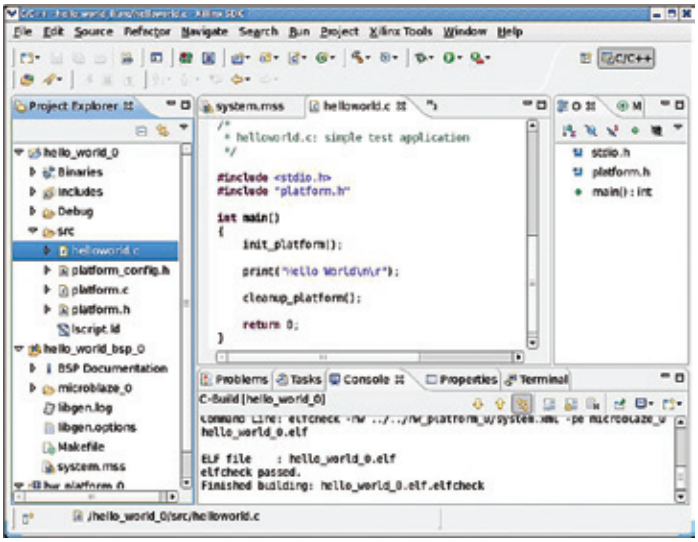


図 2 – Eclipse IDE のコード作成画面

誤解：ドライバーが存在しない。
真実：FPGA エンベデッド プロセッサには、ドライバーが付属されたペリフェラルが数多くライブラリとして提供されています。表 1 は、FPGA ソフト プロセッサ用に提供されている代表的なソフト ペリフェラルの一覧です。これらのデバイスには、いずれも適切なドライバーが存在します。

誤解：ハードウェア エンジニアによるエンベデッド プロセッサの開発が完了しなければコーディングを開始できない。
真実：既製のテスト済みプロセッサ システム デザインも提供されており、これらを利用すればすぐにソフトウェア開発に着手できます。

これらの ASSP ライクな既製プロセッサ システムには、プロセッサ、メモリ コントローラーとメモリ、フラッシュ メモリ コントローラー、各種ペリフェラル (UART、GPIO、イーサネット インターフェイスなど) が含まれます。また、Linux ブートのデモが含まれたサンプルのリファレンス ソフトウェア デザインも付属しています。

誤解：FPGA エンベデッド プロセッサはプロファイリングやトレースが行えない。
真実：プロファイリングやトレース用のツールも用意されています。プロファイリング ツールを利用すると、プロセッサが各関数の実行に費やした時間や、特定の関数を呼び出した回数を調べることができます。

誤解：FPGA ソフトウェア開発ツールは高価である。
真実：ASSP および FPGA の両サプライヤーから提供されるエンベデッド ソフトウェア開発機能の価格は、どちらも 200 ～ 500 ドル程度で大差ありません。また、多くのベンダーが試用版や無償版、機能制限版を提供しているほか、評価キットが優待価格で販売されていることもあります。

コードの作成とデバッグ

FPGA エンベデッド プロセッサ システムのソフトウェア開発は、一般に次の手順で行います。

1. ソフトウェア開発ワークスペースを作成し、ハードウェア プラットフォームをインポートする
2. ソフトウェア プロジェクトとボード サポート パッケージ (BSP) を作成する
3. ソフトウェアを作成する
4. ソフトウェア プロジェクトを実行してデバッグを行う
5. (オプション) ソフトウェア プロジェクトのプロファイリングを行う

手順 3、4、5 はほとんどの開発者にとって馴染み深いものです。手順 1 と 2 は経験のない開発者もいると思いますが、内容自体は非常に簡単です。ここでは、Eclipse 開発環境を例にとって各手順を詳しく見ていきます。

ワークスペースの作成とハードウェア プラットフォームのインポート

Eclipse の起動後、まず使用するワークスペースを指定します。ワークスペースとは、プロジェクト ファイルを格納するディレクトリのパス名です。次に、ハードウェア プラットフォーム (デザイン) を指定します。このファイルはプロセッサ システム (メモリ インターフェイスとペリフェラル、メモリ マップを含む) を記述したもので、

通 信	ペリフェラル	システム
PCI	汎用 I/O	デバッグ モジュール
PCI Express	UART 16550	割り込みコントローラー
USB	ΔΣ ADC、DAC	タイマー /PWM
I2C	外部ペリフェラル コントローラー	タイムベース / ウォッチドッグ
SPI	ADC	メールボックス
Gigabit Ethernet MAC	**メモリ**	ミューテックス
Ethernet MAC Lite	マルチポート メモリ コントローラー (SDRAM、DDR、DDR2、DDR3)	クロック ジェネレーター
CAN	外部 メモリ コントローラー (フラッシュ、SRAM)	システム リセット ブロック
FlexRay	Compact Flash	セントラル DMA
MOST	オンチップ メモリ	

表 1 – ソフト プロセッサのペリフェラル一覧 (例)

ハードウェア開発ツールによって自動的に生成されます。通常は、ハードウェア開発ツールから出力されたこのファイルをハードウェア エンジニアがソフトウェア開発者に提供します。このファイルを指定すると、ハードウェア プラットフォームがインポートされ、この手順は終了します。

ソフトウェア プロジェクトとボード サポート パッケージの作成

ボード サポート パッケージ (BSP) とは、ソフトウェア アプリケーションで使用するライブラリとドライバー一式を集めたもので、ソフトウェア プロジェクトはソフトウェア アプリケーションのソースと設定をまとめたものです。

ザイリンクスのプロジェクト用にカスタマイズされたバージョンの Eclipse では、メニューで [File] → [New] → [Xilinx C Project] をクリックします。ザイリンクス C プロジェクトの場合、Eclipse によって自動的に Makefile が作成されます。この Makefile に基づき、ソース ファイルをコンパイルしてオブジェクト ファイルを生成し、オブジェクト ファイルをリンクして実行可能ファイルを生成します。この際、ユーザーは [Project Name] を定義し、手順 1 で作成したハードウェア プラットフォーム名を入力してプロジェクトとハードウェア プラットフォームを関連付けた後、プロジェクトに名前を付けます。

次に、システムが BSP の生成を確認し、定義されたハードウェア プラットフォームと OS に基づいて適切なドライバーを自動的にロードします。引き続きこれらのドライバーがコンパイルされ、BSP が生成されます。

ソフトウェアの作成

サンプル ソフトウェアをインポートするか、または新規コードを作成します。コードを保存すると Eclipse によって自動的にコードのコンパイルとリンクが実行され、コンパイラやリンカーにエラーがあるとエラー レポートが生成されます。

ソフトウェア プロジェクトの実行とデバッグ

FPGA 特有の手順として、コードを実行する前に FPGA へのプログラミングを完了しておく必要があります。Eclipse のメニューで [Tools] → [Program FPGA] をクリックします。これにより、ハードウェア エンジニアが作成したハードウェア デザインが FPGA にダウンロードされます。以上の手順が完了したら、次にビルドするソフトウェアの種類を選択します。[Debug] を選択するとコード最適化が無効になり、デバッグ シンボルが挿入されます。[Release] を選択するとコード最適化が有効になります。プロファイリングを行う場合は、コンパイル オプションに -pg を指定します。

最後に、[Run] をクリックして実行時の設定やコンパイラ オプションを定義し、コードを実行します。[Release] を選択すると、プロセッサはただちにコード実行を開始します。それ以外の場合、プロセッサはブート命令をいくつか実行した後、ソース コードの最初の行で停止し、Eclipse のデバッグ画面が表示されます。

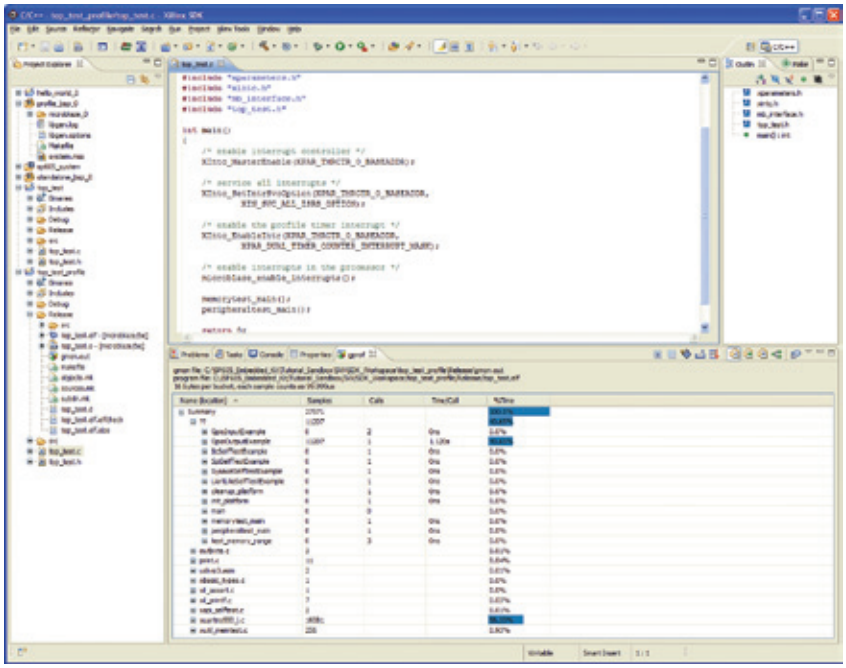


図 3 – Eclipse のプロファイラー画面

デバッグ画面ではソースまたはオブジェクト コード、レジスタ、メモリ、変数が確認できるほか、ソース レベルまたはオブジェクト レベルでコードをステップ実行したり、ブレークポイントを設定してコードを実行することも可能です。

ソフトウェア プロジェクトのプロファイリング

必要に応じて、コードのプロファイリングを実行し、関数の呼び出し回数や各関数の実行にかかった時間を調べることができます。図 3 は、Eclipse のプロファイラー画面でプロファイリングを実行した結果を示したものです。

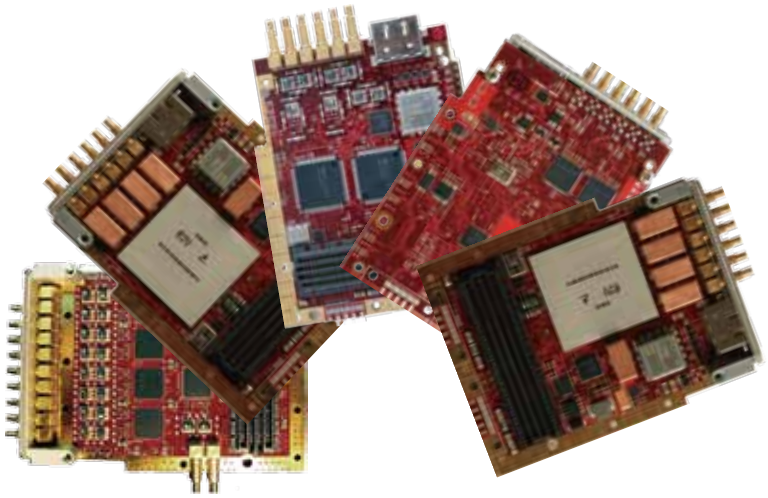
FPGA の利点

コスト、消費電力、サイズ、全体的なシステム効率のすべての面で、FPGA エンベデッド プロセッサは最も有力な選択肢となりつつあります。ソフトウェア エンジニアにとって FPGA エンベデッド プロセッサは馴染みが薄く、外部プロセッサのプログラミングよりも難しいのではと敬遠されているかもしれませんが、実際はそのようなことはありません。

FPGA ベンダーからは Eclipse のような業界標準の開発環境を FPGA エンベデッド プロセッサ用にカスタマイズしたものが低価格で提供されています。これらの開発環境を利用すれば、これまで外部プロセッサで行ってきたように FPGA エンベデッド プロセッサ用コードの作成、コンパイル、リンク、ダウンロードが可能です。さらに既製のプロセッサ リファレンス デザインを使用することで、ハードウェア エンジニアがプロセッサ設計を完成させていなくてもソフトウェア エンジニアはコーディングとテストを開始できます。また、FPGA エンベデッド プロセッサは IP ライブラリ、ドライバー、OS サポートも充実しています。

ラインナップ充実!!

A/D・D/A・DSP FMCモジュール



FMCモジュール搭載可能なFPGAボードも各種取り揃えています。

＊FMCは「FPGA Mezzanine Card]の略で、VITA57で規格化されたメザニンカードの一つです。

FMCモジュールはFPGAのI/O信号にダイレクトに接続され、高速でデータの送受信を実現します。

高速A/D＆D/A FMCモジュール

型番	Sampling	分解能	ch数
FMC110	1GHz 1GHz	12bit 16bit	2ch A/D 2ch D/A
FMC150	250MHz 800MHz	14bit 16bit	2ch A/D 2ch D/A
FMC103	210MHz	12bit	4ch A/D
FMC104	250MHz	14bit	4ch A/D
FMC107	65MHz	12bit	8ch A/D
FMC108	250MHz	14bit	8ch A/D
FMC112	125MHz	14bit	12ch A/D
FMC116	125MHz	14bit	16ch A/D
FMC122	2.5GHz	8bit	2ch A/D
FMC125	5GHz	8bit	4ch A/D
FMC126	5GHz	10bit	4ch A/D
FMC204	1GHz	16bit	4ch D/A
FMC210	8GHz	12bit	1ch D/A
ADF-2500	2.5GHz	10bit	1ch A/D
ADF-1600	1.6GHz	12bit	2ch A/D
ADF-Q55	550MHz	12bit	4ch A/D
ADF-Q40	400MHz	14bit	4ch A/D
SHMEZ-AD-FMC	400MHz	14bit	2ch
SHMEZ-DA-FMC	400MHz	14bit	2ch
AF201	2GHz	12bit	1ch A/D
AF202	1.5GHz	12bit	2ch A/D
AF301	QSFP Interface *		
AF101	FMC Loopback test mezzanine		
FMC645	DSP (TMS320C6455)		

* QSFP = Quad Small Formfactor Pluggable

FMCカード2枚搭載可能！組込シャーシもリリース

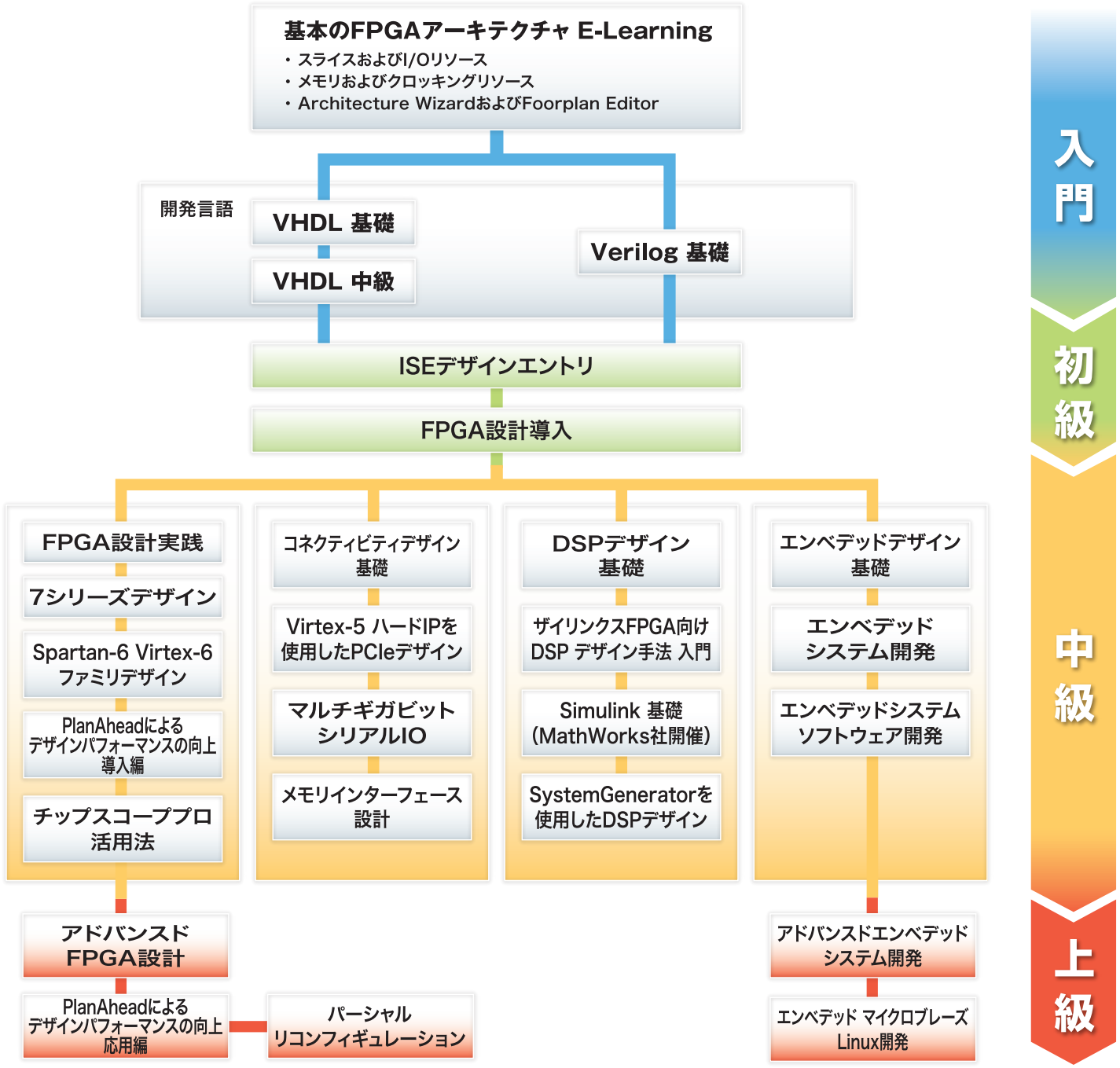
シャーシ内部にはXilinx社製FPGA評価ボード（ML605 / VC707）を据付けています

外形寸法：280(W) x 220(D) x 50(H) mm

株式会社ミッシュインターナショナル
〒190-0004 東京都立川市柏町4-56-1 TEL:042-538-7650
sales@mish.co.jp http://www.mish.co.jp

トレーニング カリキュラムパス

XILINX



ザイリンクス販売代理店 オリジナル トレーニング

販売代理店各社の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 ジャパン	www.jp.avnet.com/design/training/
新光商事	新光商事	xilinx.shinko-sj.co.jp/training/index.html
東京エレクトロンデバイス	東京エレクトロンデバイス	ppg.teldevice.co.jp/
PALTEK	パルテック	www.paltek.co.jp/seminar/index.htm
FUJITSU	富士通エレクトロニクス	jp.fujitsu.com/fei/services/maker/xilinx/xtraining.html

詳細とご登録はこちらから

Japan.xilinx.com/training

XILINX®

2倍のパフォーマンス、半分の消費電力

ザイリンクスの 7 シリーズ FPGA で妥協のない革新を。
新しい 7 シリーズ デバイスは統一されたアーキテクチャ上に構築しているため、
選択肢が広がりお客様のコンセプトをより現実的に設計できます。
また、高性能、低消費電力などさまざまなニーズを満たします。
次世代のISE® Design Suiteで開発時間をスピートアップし、
この高性能と柔軟性で世界を変える革新をおこしてください。

http://japan.xilinx.com/7

ARTIX 7™ 最も低いシステムコスト

KINTEX 7™ 最高の価格対性能

VIRTEX 7® 最高のシステム性能

ザイリンクス株式会社

製品のお問い合わせは下記の販売代理店へどうぞ

■東京エレクトロン デバイス(株) TEL(045)443-4016 x2web@teldevice.co.jp ■アヴネット ジャパン(株) TEL(03)5792-8210 EVAL-KITS-JP@avnet.com ■(株)PALTEK TEL(045)477-2005 info_pal@paltek.co.jp
■富士通エレクトロニクス(株) TEL(045)415-5825 fei-xinq@cs.jp.fujitsu.com ■新光商事(株) TEL(03)6361-8086 X-Pro@shinko-sj.co.jp