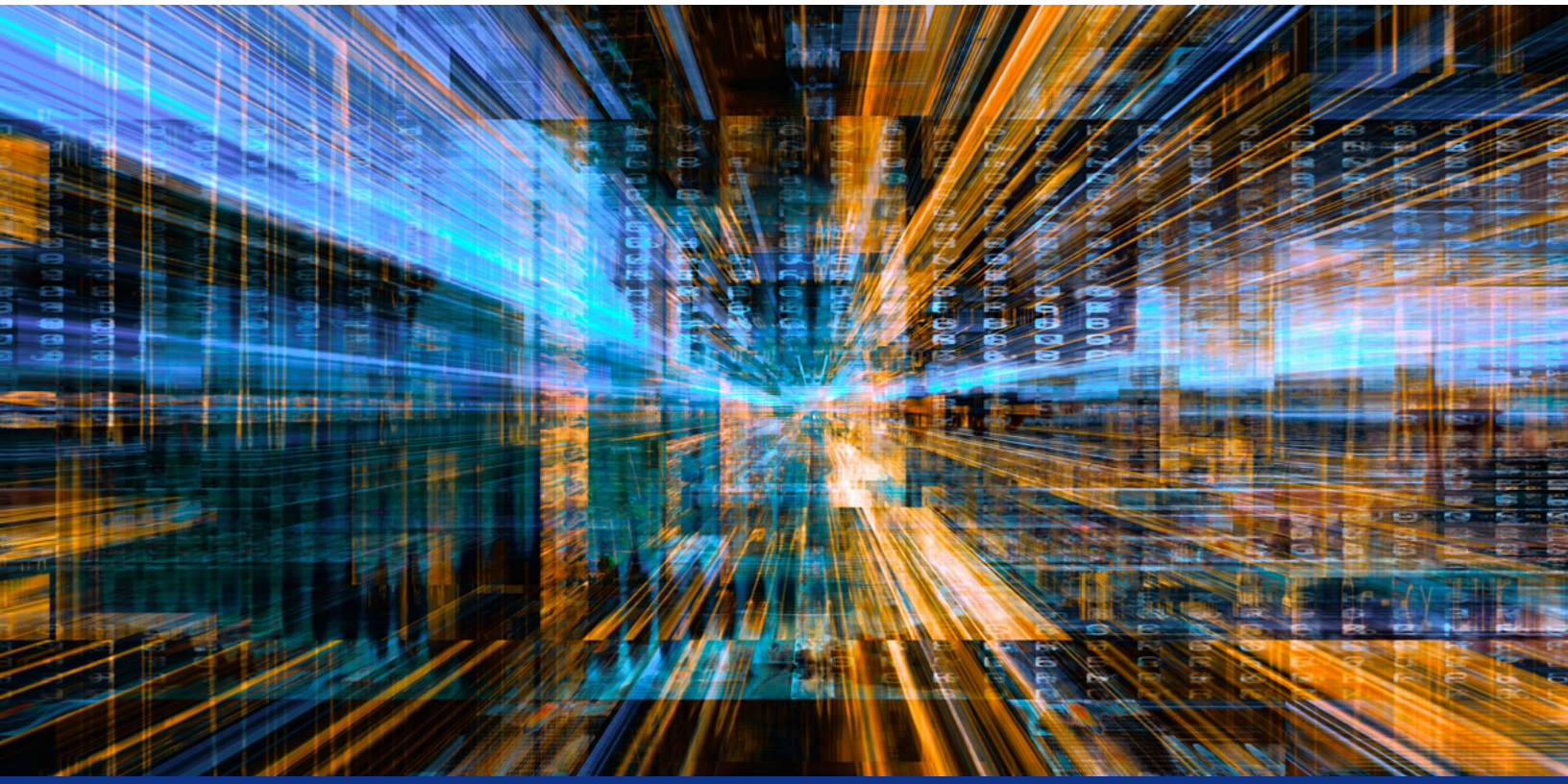


OpenCL* による FPGA プログラミング

FPGA のプログラム方法の理解と実践

James Reinders James Reinders Consulting LLC コンサルタント & HPC エンスージアスト
Tom Hill インテル コーポレーション プログラマブル・ロジック・グループ DSP 製品ライン・マネージャー

FPGA が注目されている理由

フィールド・プログラマブル・ゲート・アレイ (FPGA) は、低レイテンシーで高いパフォーマンスと電力効率を実現できる優れたハードウェアです。これらの長所は、FPGA の大規模な並列機能と再構成の容易性により実現されます。FPGA は、次のような処理を行うことができる、再構成可能なチャネルレス型の構造を提供します。

- カスタム・ハードウェア・アクセラレーターを**設計**する。
- 単一アプリケーションに**配置**する。
- 異なるアプリケーションの新しいアクセラレーターとしてデバイスを**素早く再構成**する。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

この記事では、独自の設計を行う方法と、それらの設計を使用してアプリケーションを高速化する方法を説明します。ハードウェア・エンジニアは、さまざまなアプリケーションで ASIC の代わりに FPGA を長年使用してきました。これまで、FPGA の構成 (プログラミング) は、Verilog* や VHDL* のような高水準定義言語を利用して行われてきました。これらの設計手法はハードウェア・エンジニアにはよく知られていますが、ソフトウェア開発者にとっては全く新しいものです。

OpenCL* を中心とする新しいツールは、このギャップを埋め、FPGA ハードウェア・プラットフォームの恩恵をソフトウェア開発者にもたらします。これらの新しいツールを使用すると、C のような構文で記述されたカスタム・アルゴリズムを高速で電力効率の良いハードウェアに変換できる、高度に構成可能なデバイスとして FPGA を活用することができます。これらのソリューションの高い柔軟性とパフォーマンスは、最新世代の FPGA デバイスのおかげです。

大規模な並列処理

インテル® Stratix® 10 FPGA は、最大 10 TFLOPS の単精度浮動小数点パフォーマンスを提供します。また、標準数値形式に制限されません。例えば、2 ビットのディープ・ニューラル・ネットワークの実装は困難でしたが、2017 年 11 月にスーパーコンピューター上で行われたデモで紹介された、インテルのディープラーニング・アクセラレーターの構成可能なワークロードを利用すると簡単に実装できます。

インテル® FPGA は、大規模な並列処理とハードウェアの柔軟性によりレイテンシーを軽減します (**図 1**)。FPGA はデータフロー (またはパイプライン) の構成がプログラムされるため、データの移動が最適化され、ホスト CPU と FPGA 間のデータ転送は排除されます。

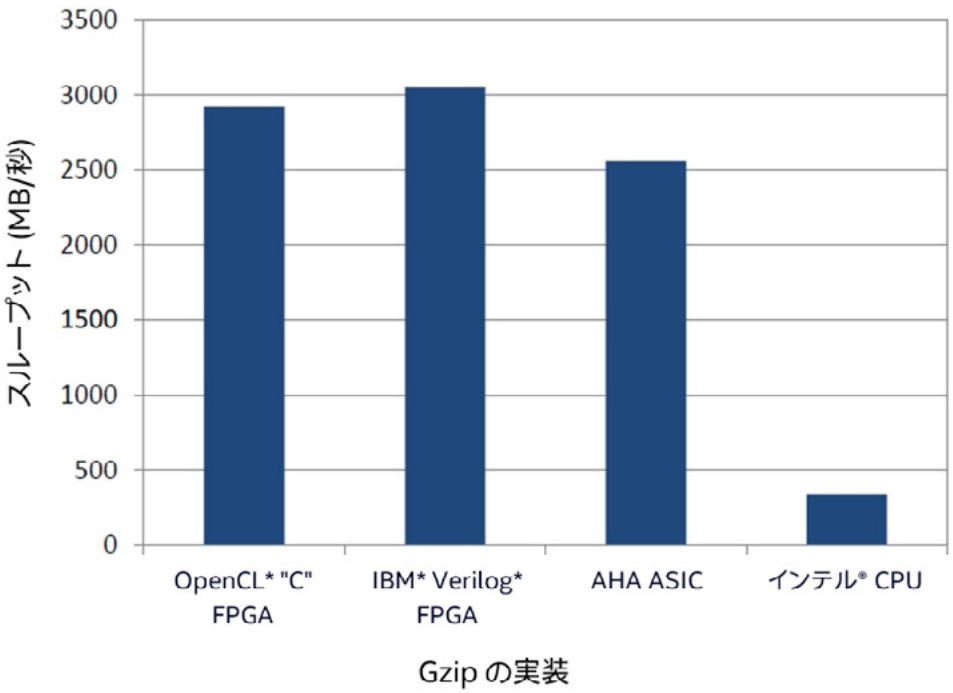
注意すべき点

現在では、C プログラマーが FPGA プログラミングを試してみることを妨げるものは何もありません。ただし、OpenCL* (新しい開発環境と新しい手法を含む) を使用して FPGA のプログラミングを行う場合も、ほかのすべてのプログラミングと同様に、学習曲線が存在します。幸い、オンライン・トレーニングと OpenCL* コーディング・ガイドを利用することで、簡単に FPGA プログラミングに取り組むことができます。

FPGA の開発環境

OpenCL* ソフトウェア開発環境は Eclipse* または Visual Studio* ベースであるため、C プログラマーには馴染み深いものです。この記事では、OpenCL* で最初のプログラムを記述する方法と、そのプログラムを FPGA で実行する方法を説明します。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。



1 FPGA への対応が進む OpenCL*: 非常に生産的な OpenCL* FPGA プログラミングと、生産性が低く、アプローチの種類が少ない 3 つの従来のプログラミングを使用した結果の比較。(この記事の最後の**関連情報**セクションの「Gzip on a Chip」を参照。)

FPGA の最適化

新しいデバイスとプログラミング手法には、最高のパフォーマンスを達成するさまざまな秘訣があります。これらの最適化は、アルゴリズムのデータ構造組織やクリーン思考などのよく知られているアプローチから、OpenCL* でパイプラインを表現する最も効果的な方法までさまざまです。よくある状況ではありませんが、CPU のインライン・アセンブリーを使用するのと同じように、ハードウェア記述言語プログラミング (Verilog* または VHDL*) も使用可能です。FPGA の最適化は CPU や GPU の最適化とは異なりますが、簡単でも難しくもありません。

CPU や GPU と同様に、FPGA プログラマーは FPGA 向けに記述された (ハードウェア・エンジニアが通常 IP ブロックと呼ぶ) ライブラリーを使用できます (FPGA の最適化についてさらに詳しく知りたい方は、この記事の次に、**インテル® FPGA SDK for OpenCL* ベスト・プラクティス・ガイド**をお読みになることを推奨します。)

FPGA プログラミングを試す 3 つのアプローチ

FPGA プログラミングを試す 3 つの方法について説明します。

- **エミュレーション**。FPGA ハードウェアを購入することなく FPGA プログラミングを試すことができます。FPGA プログラムを開発およびデバッグする際に推奨する最初のステップです。SDK は Windows* または Linux* で利用できます。
- **オフラインコンパイル**。エミュレーションを利用してデバッグした後、FPGA なしで OpenCL* 環境を使用してコンパイルを行い、FPGA プログラミング・ファイルを生成します。OpenCL* カーネルの正確なエリアレポートとパフォーマンス・レポートを調べて、FPGA 上で実際に実行する前にコードを改良します。
- **クラウド**。クラウドの **Nimbix*** (英語) システムでブラウザーからツールと FPGA を使用して、エミュレーションから FPGA 上での実行まで、すべての開発を行うことができます。現在の構成は、インテル® ツールのみを含む CPU ベースのマシン、ツールとインテル® Arria® 10 FPGA を含むマシンです。コストは、プログラム開発が 1 時間あたり 36 セントから、FPGA を搭載したマシンの使用が 1 時間あたり 3 ドルからです。Nimbix* システムは Linux* を実行します。

ターゲット・プラットフォームとして FPGA 開発ボードを使用することもできます。最新の FPGA のすべての機能にアクセスできます。1 枚の PCIe* カードのコストは、**インテル® Arria® 10 FPGA** を搭載した場合は約 5,000 US ドル、**インテル® Stratix® 10 FPGA** を搭載した場合は約 11,000 US ドルになります。技術的には、**Cyclone®** FPGA 開発ボードで OpenCL* を使用することも可能です。**DE1-SoC ボード** (英語) はわずか 250 US ドルと非常に魅力的です。しかし、**表 1** を見れば分かるように、このボードは計算の高速化に適しているとはお世辞にも言えません。FPGA 上で OpenCL* を使用した計算の高速化を 250 US ドル以下で試すには、上記のオプション 1、2、3 の組み合わせを推奨します。ここでは、あえて DE1-SoC ボードでインテル® FPGA SDK for OpenCL* を使用しました。問題なく動作しましたが、予想通り計算能力は非常に低いものでした。

表 1. OpenCL* をサポートしているインテル® FPGA プラットフォーム (一部)

プラットフォーム	製品番号	最大ロジック エレメント数 (百万)	最大エンベデッド・メモリー (M ビット)	最大ピーク TFLOPS (単精度、IEEE)
Cyclone® V SoC 開発ボード	**DE1-SoC ボード** (英語) (Cyclone® V - 5CSEMA5F31C6N)	0.085	3.9	0.001 未満
インテル® Arria® 10 FPGA 開発キット (英語)	**DK-DEV-10AX115S-A** (英語)	1.1	53	1.4
インテル® Stratix® 10 FPGA 開発キット (英語)	**DK-DEV-1SGX-H-0A** (英語)	5.5	229	9.2
Nimbix* クラウド	**BittWare A10PL4 ボード** (英語) および **インテル® Arria® 10 GX 1150** (英語)	1.1	53	1.4

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® Arria® 10 FPGA ファミリーおよびインテル® Stratix® 10 FPGA ファミリーはアプリケーションの高速化を、Cyclone® FPGA ファミリーは低消費電力の組込み機器での利用を、それぞれ想定しています。OpenCL* は、インテル® Arria® 10 FPGA とインテル® Stratix® 10 FPGA の両方でサポートされています。また、一部の Cyclone® V SoC FPGA でもサポートされています。どのボードを選んだ場合でも、インテル® FPGA SDK for OpenCL* が特定のシステム構成を適切にターゲットするために使用する、ボード・サポート・パッケージ (BSP) が必要です。

なぜ OpenCL* なのか

OpenCL* は、ヘテロジニアス･システムのアルゴリズムを高速化する C を拡張した標準規格であり、プログラミング･モデルです。CUDA* のように、OpenCL* の中心となるのは、デバイス (ハードウェア・アクセラレーター) 上で実行するカーネルの概念です。カーネルは C99 のサブセットで記述されています。CUDA* とは異なり、OpenCL* は可搬性がある、ロイヤルティー・フリーのオープンソース標準規格であり、CPU、GPU、および FPGA を含むあらゆる計算デバイスをサポートするように設計されています。

OpenCL* には、ホスト (CPU) が PCIe* 経由でデバイス (今回のケースでは FPGA) と通信するため、またはカーネルがホストを介することなく別のカーネルと通信するためのアプリケーション･プログラミング･インターフェイス (API) も用意されています。これに加えて、インテルは、10Gb イーサネットなどのストリーミング I/O インターフェイスからカーネルへデータを直接送る I/O チャネル API も提供しています。

OpenCL* プログラミング・モデルでは、ユーザーがコマンドキューにタスクをスケジュールします。各デバイスには少なくとも 1 つのコマンドキューがあります。OpenCL* ランタイムは、データ並列タスクを分割して、デバイスのプロセシング・エレメントに送ります。これがホストが任意のデバイスと通信する方法です。ベンダー固有の実装を行わないようにすることは、個々のデバイスベンダーの責任です。**インテル® FPGA SDK for OpenCL*** (英語) は固有の実装を行わず、OpenCL* 1.2 標準規格に準拠しています。

インテル® FPGA SDK for OpenCL*

インテル® FPGA SDK for OpenCL* を使用すると、ユーザーは、従来のハードウェア FPGA 開発フローではなく、より迅速で高レベルのソフトウェア開発フローを利用できます。OpenCL* のアクセラレーター・コードを、x86 ベースのホストで数秒エミュレートして、特定のアルゴリズムのパイプライン依存関係に関する情報を含む詳細な最適化レポートを取得できます。このレポートにより、長いコンパイル時間をかけて FPGA 上で実行する最終的なコードを生成する前に、プログラムをデバッグすることができます。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® FPGA SDK for OpenCL* は、FPGA 設計の複雑さを排除することにより開発を促進します。ソフトウェア・プログラマーは、ANSI C ベースの言語である OpenCL* C と追加の OpenCL* 構文を使用して、ハードウェア・アクセラレーションを活用したカーネル関数を記述できます。

インテルは、SDK の一部として、通常ソフトウェア・プログラマーが利用する開発フローに近いツールのスイートを提供します。

- **エミュレーター**。x86 CPU 上でコードをステップ実行し、コードが機能的に正しいことを保証します。
- **詳細な最適化レポート**。内部ループのロードとストアの依存関係を理解するのに役立ちます。
- **プロファイラー**。適切なメモリー結合とストールフリーのハードウェア・パイプラインを保証するためカーネルのパフォーマンス情報を提供します。
- **OpenCL* コンパイラー**。カーネルコードに 300 を超える最適化を行い、1 ステップで FPGA イメージ全体を生成できます。

インテル® FPGA SDK for OpenCL* は、**インテル® Xeon® プロセッサー**、SoC の組込み **ARM* Cortex*-A9 プロセッサー・コア** (英語)、および **IBM* Power Systems* プロセッサー** (英語) を含む、さまざまなホスト CPU をサポートしています。複数の FPGA と複数のボードにおけるスケーラブルなソリューションに加えて、DDR SDRAM (シーケンシャル・メモリー・アクセス)、QDR SRAM (ランダム・メモリー・アクセス)、内部 FPGA メモリー (低レイテンシー・メモリー・アクセス) などのさまざまなメモリーターゲットをサポートしています。半精度、単精度、および倍精度浮動小数点演算もサポートしています。

FPGA プログラミングを試す

では、FPGA プログラミングを試す 3 つの手法を順に紹介します。

エミュレーション

エミュレーション手法では FPGA は必要ありません。FPGA 向けの OpenCL* プログラムを記述する場合、エミュレーションは OpenCL* コードを効率的にデバッグするための重要な最初のステップです。FPGA 向けのコンパイルは非常に遅く、デバッグオプションも制限されているため、FPGA で確認する前にエミュレーションを使用してできるだけデバッグしておくことが非常に大切です。

これらの FPGA プログラミングを試す 3 つの手法を説明するため、単純なベクトル加算について考えてみましょう。**インテル® FPGA SDK for OpenCL*** をダウンロードしてインストールした後、**インテル® FPGA ソフトウェア・デザイン・センター** (英語) からベクトル加算のサンプルをダウンロードします。サンプルをダウンロードした後 .zip ファイル (Windows*) または .tar ファイル (Linux*) を展開して、README.html の内容を確認します。SDK およびサンプルコードは無料でダウンロードして使用できます。Windows* と Linux* の両方でサンプルが正しく動作することを確認し、インテル® FPGA SDK for OpenCL* は、**VBox*** (英語) 内で Linux* を実行している macOS* システムで効率良く動作することが分かりました。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

コードのパスにスペースを含めないように注意してください。ツールがパスのスペースを正しく処理できなかったために、いくつかの問題が発生しました。FPGA ツールのドキュメントの一部にはこの情報が記載されていますが、OpenCL* にしか興味がなければ読むことはないでしょう。(現在、これらの制限を解決する作業を進めています。)

エミュレーション手法には 3 つのステップがあります。

1. FPGA コードをビルドする。
2. ホストコードをビルドする。
3. プログラムを実行する。

FPGA コードをビルドする

FPGA コードは OpenCL* で記述されたカーネルから構成されます。Altera OpenCL* コンパイラー (aoc) で **-march=emulator** オプションを指定してエミュレーションを使用します。このオプションを指定しない場合、コンパイラーは FPGA のすべての合成、配置、および配線を行うため、コンパイル時間が非常に長くなります。**-march=emulator** オプションを指定すると、コンパイルは短時間で完了し、x86 コードが生成されます。生成されたプログラムは FPGA なしで実行できます。ここでの目標はエミュレーションであるため、このオプションを指定します。**vector_add** ディレクトリーから、README.html の指示に従って、次のコマンドを使用します。

```
aoc -march=emulator device/vector_add.cl -o bin/vector_add.aocx
```

ホストコードをビルドする

ホストコードは C で記述されており、OpenCL* API を呼び出して、FPGA をセットアップし、カーネルをロードして FPGA で使用します。Windows* では、Microsoft* Visual Studio* を利用します (無償の Community エディションで動作することを確認済み)。Linux* では、Makefile を利用して “make” を行います (GNU* C++ コンパイラーを使用)。

プログラムを実行する

ホストプログラムがエミュレートされた FPGA コードを実行するよう、README.html の指示に従って **CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA=1** に設定します。この指示のほかに、次の操作を行います。

最初に (ほかのステップを行う前に)、インテル® FPGA SDK for OpenCL* の変数を初期化します。

- **Windows*:** `%INTELFPGAOCLSDKROOT%\init_opencl.bat`
  (テストマシンでは、`C:\intelFPGA\17.1\hld\init_opencl.bat` でした)。
- **Linux*:** `$INTELFPGA_ROOTDIR/hld/init_opencl.sh`
  (テストマシンでは、`/opt/intelFPGA_pr/17.1/hld/init_opencl.sh` でした)。

Windows* では、すべての処理が 64 ビットで行われるように Visual Studio* の環境変数も初期化します。(テストシステムでは、この初期化を行わないとコンパイルに失敗しました。)

- `vcvars64.bat` (テストマシンでは、`C:\Program Files (x86)\Microsoft Visual Studio\Shared\14.0\VC\bin\amd64\vcvars64.bat` でした)。

オフラインコンパイル

OpenCL* 環境を使用して FPGA プログラミング・ファイルとレポートを生成するコンパイルを実行できます。これらのレポートは、`aoc` コマンドを実行すると自動的に作成されます。FPGA のパフォーマンスを多少 (10 ~ 20 パーセント) 落としてプログラミング・ファイルの出力とレポートを高速に生成する `-fast-compile` オプションもあります。これらのレポートはカーネルの最適化に役立ちます。レポートはコンパイル中に静的に作成されるため、この出力を得るために FPGA でプログラムを実際に実行する必要はありませんが、特定の FPGA 向けにコンパイルする必要があります (`-march=emulator` オプションはレポートを生成しません)。`vector_add` ディレクトリーから、FPGA コードをビルドしたときと同じコマンドを使用します。

```
aoc -march=emulator device/vector_add.cl -o bin/vector_add.aocx
```

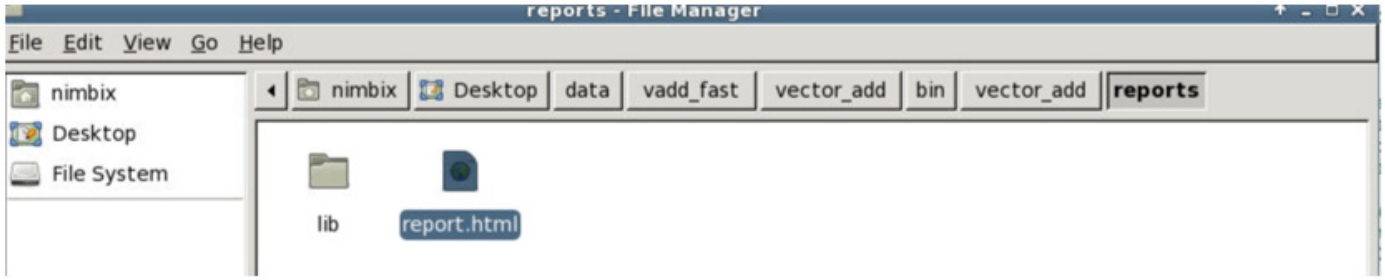
コンパイラーは、次のような使用状況レポートを出力します。

```
aoc: Optimizing and doing static analysis of code...

!===========================================================================
! The report below may be inaccurate. A more comprehensive
! resource usage report can be found at vector_add/reports/report.html
!===========================================================================

+-------------------------------------------------------------------+
; Estimated Resource Usage Summary ;
+----------------------------------------+--------------------------+
; Resource + Usage ;
+----------------------------------------+--------------------------+
; Logic utilization ; 18% ;
; ALUTs ; 9% ;
; Dedicated logic registers ; 9% ;
; Memory blocks ; 9% ;
; DSP blocks ; 0% ;
+----------------------------------------+--------------------------;
Compiling for FPGA. This process may take a long time, please be patient.
```

これらのリソース使用状況は、OpenCL* プログラムが FPGA で利用可能な並列処理を適切に行っているか判断するのに役立ちます。これは重要なチューニング情報であり、長いコンパイル時間をかけることなく取得できます。非常に多くのリソースを使用している場合、そのプログラムは特定の FPGA には適さないことがあります。逆に、ほとんどリソースを使用していない場合でも、コードをチューニングすることでパフォーマンスを向上できる可能性があります。デバイスの (例えば、浮動小数点処理能力の) 10 パーセントしか使用していない場合、最大限のパフォーマンスを引き出していないと理解するのは容易です。

フルコンパイルを行うと、より詳細なレポートが作成されます。これらのレポートはより正確であり、微調整には役立ちますが、作成するには長い時間がかかります。そのため、FPGA がない場合でも、レポートを高速に (約 1/4 の時間で) 作成する高速コンパイルオプションを指定すると時間を節約できます。(レポートの使用方法は、**インテル® FPGA SDK for OpenCL* ベスト・プラクティス・ガイド**を参照してください。)

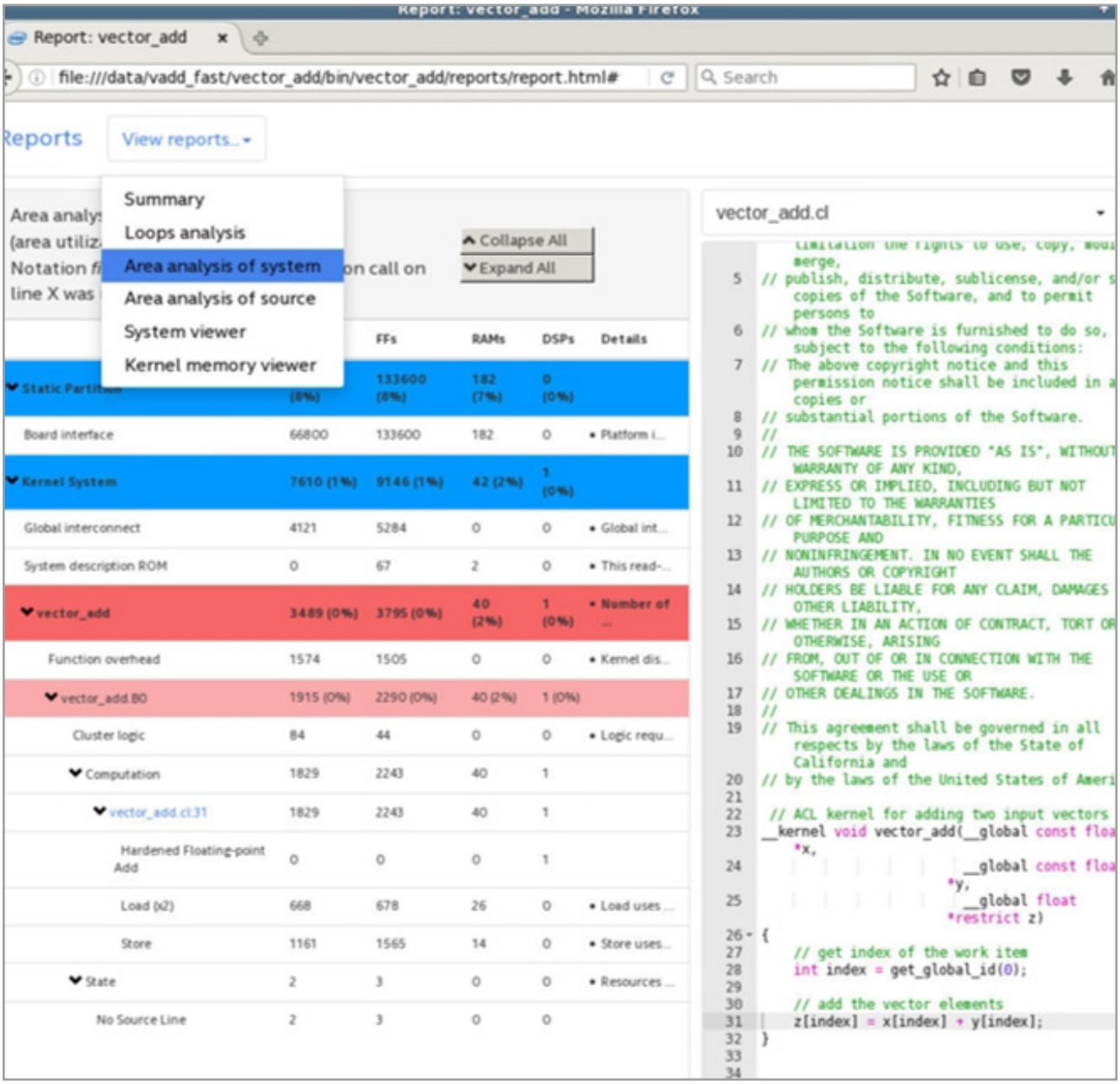
現在は、コンパイラーで複数の処理を自動的に行い、結果をまとめてブラウザーに表示することができます (以前は、手動で処理を行う必要がありました)。コンパイル中に作成された report.html ファイルを見つけることから始めます (**図 2**)。(ファイルの場所は使用状況レポートに含まれています。)



2 レポートを開く

結果ページから複数のレポートを参照できます。**図 3** は、ベクトル加算のサンプルのサマリーを示しています。約 1 時間のコンパイルの後に作成された詳細レポートで、コンパイル開始から数秒でレポートされた使用状況を確認するというのはなかなか面白いことです (ALUT 使用 9 パーセント、専用論理レジスター 9 パーセント、メモリーブロック 9 パーセント、DSP なし)。**図 4** は、エリア解析レポートを示しています。このレポートでは、プログラムにドリルダウンして、ソースコードとリソース使用状況の関係を理解することができます。

3 `-fast-compile` を使用したコンパイルのサマリーページ



4 **-fast-compile** を使用したコンパイルのシステムレポートのエリア解析

最後のヒント : "**aocl env aocxfilename.aocx**" コマンドは aocx ファイルがどのようにコンパイルされたか詳細な情報を出力します (**report.html** サマリーの情報よりも詳細です)。

クラウド

クラウド手法では、各自が FPGA を所有する必要はありません。また、ソフトウェアをダウンロードしたりセットアップする必要もありません。代わりに、Nimbix* のアカウントをセットアップし、使用量に応じて料金を払います。macOS*、Windows*、および Linux* ホストでブラウザーから、このアプローチを使用して動作することを確認しました。レートは 1 時間当たり 36 セントからで、秒単位で比例配分して計算されます。ごくわずかな料金で OpenCL* を試すことができ、ハードウェアやソフトウェアのセットアップが必要ないため大幅に時間を節約できました。最終的に、Nimbix* でベクトル加算のサンプルを試すために支払った料金は 5 ドル以下でした。

ほかのクラウドベースの FPGA 開発システムとは異なり、Nimbix* とインテルは、クラウドで実行するためツールの特別バージョンを作成していません。つまり、FPGA コードのビルドに費用をかけたくなければ、手法 1 と 2 の無料のツールを使用して、クラウドで実行するバイナリーをビルドすることができます。FPGA 上で実行できる準備が整うまで、ローカルで開発、デバッグ、およびビルドするのはより自然なことに感じられるでしょう。また、クラウドには最新のツールが用意されています。クラウドで使用されているボードと同じボードを持っていれば、ローカルシステムでもコンパイルできます。

Nimbix* クラウドにはあらゆる環境が事前にセットアップされているため、OpenCL* ツールをインストールする必要はありません (**`init_opencl`** スクリプトを source する必要もありません)。ただし、指示に従ってサンプルを実行する前に、Nimbix* のアカウントを作成し、インスタンスをアクティベートして、サンプルコードをダウンロードする必要があります。

Nimbix* クラウドを利用する手順は次のとおりです。

- **このリンク (英語) を使用してアカウントをセットアップします。**顧客情報と課金情報のみ入力してください。**`PushToCompute`** オプションは選択しないでください。料金が $0.00 になっていることを確認して SUBMIT をクリックします。
- **「Nimbix* クラウドでインテル® FPGA 開発ツールを使用するためのクイック・スタート・ガイド」(英語) を開きます。**このガイドには多くの役立つ情報が含まれていますが、OpenCL* を使用するためには必要ない情報も含まれています。このガイドを読み、マシンの **`/data`** ディレクトリーにストレージがマウントされているため、マシンをアクティベートすることなく読み書きできる点に特に留意します (「Persistent Storage and Data Transference (永続ストレージとデータ転送)」セクションを参照)。

前述のエミュレーションの手順に従って開始することを推奨します。ただし、作業を行うのはクラウド上です。具体的な手順を次に示します。

- **ダッシュボード (https://www.nimbix.net (英語)) で** [Login] をクリックしてシステムにログインします。[Compute] (右側のメニュー) をクリックし、検索フィールドに Quartus と入力して、使用するマシンを絞り込みます。利用可能な最新バージョンの Quartus (ここでは 17.1) および [PAYG] を選択します。
- **次に、[Desktop] を選択し、**CPU とメモリーを選択します。例えば、エミュレーションのコンパイルと実行のみ行う場合は 2 CPU で十分です。非エミュレーション・ビルドを行う場合は、追加のメモリーが必要になるため、8 CPU を推奨します。
- **マシンの準備ができたら、**クリックして接続します。これで、マシンにアクセスして使用できます。マシンを終了 (左下の [N] をクリックし、[Log Out] を選択して [Shutdown] を選択) するまで課金は継続されていることに注意してください。端末ウィンドウを切断してもマシンは終了しないため、課金は継続されたままです。ダッシュボード (**https://www.nimbix.net** (英語)) からマシンを終了 (kill) して、すべてのアクティビティーが停止していることを確認することを推奨します。
- **マシンで、**右クリックしてメニューを表示し、端末ウィンドウを開きます。
- 作業が保存される **`/data`** ディレクトリーで作業を行うことを**推奨します**。そのため、端末ウィンドウで最初に次のコマンドを入力します。

```
cd /data
```

- **OpenCL* ベクトル加算のサンプル (英語) で** tar ファイル (**`.tgz`**) のリンクを見つけます。**`wget`** を使用して tar ファイルをダウンロードします。

```
wget https://www.altera.com/content/dam/altera-www/global/en_US/others/support/
examples/download/exm_opencl_vector_add_x64_linux.tgz
```

- **tar ファイルを展開します。**

```
tar xvf exm_opencl_vector_add_x64_linux.tgz
```

- **README.html** には詳細な説明が含まれています。ここで必要なのは、3 つのコマンド (FPGA コードのビルド、ホストコードのビルド、実行) のみです。これらのコンパイルと実行にかかった時間は約 30 秒でした。

```
cd vector_add
```

```
aoc -march=emulator device/vector_add.cl -o bin/vector_add.aocx
make
```

```
CL_CONTEXT_EMULATOR_DEVICE_ALTERA=1 bin/host -n=10000
```

これで第 1 段階は完了です。この時点では、OpenCL* プログラムはエミュレーションによりクラウド上で実行されています。FPGA 上で実行するには、(FPGA 向けに) リビルドして実行する必要があります。

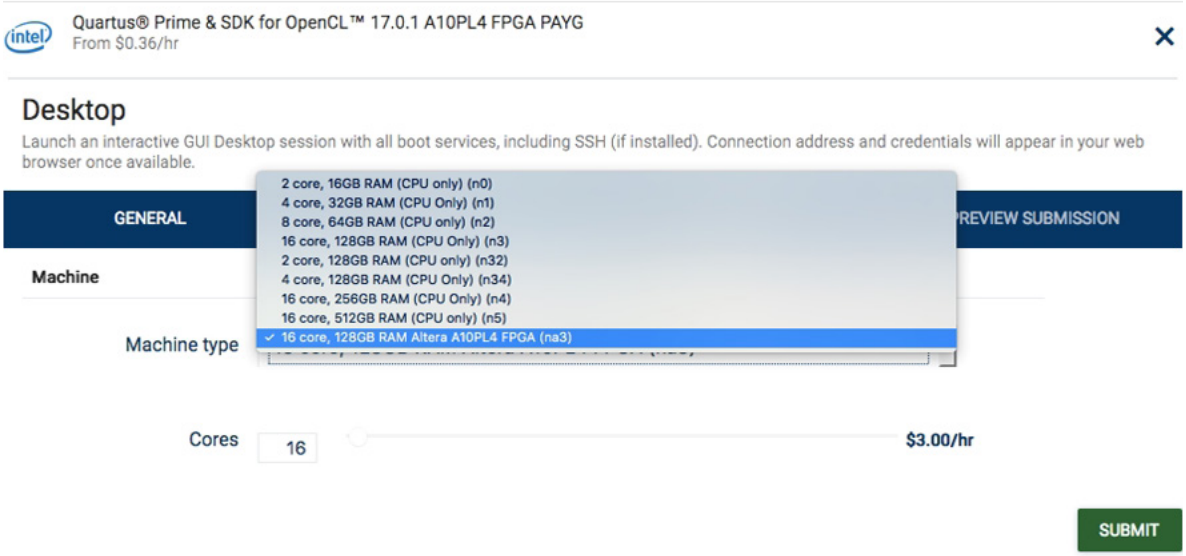
- **リビルド。**`-march=emulator` オプションを指定しないで `aoc` を使用してコンパイルします。次のコマンドを使用して 8 CPU で完了するまで約 3 時間かかりました。

```
aoc device/vector_add.cl -o bin/vector_add.aocx
```

  `-fast-compile` オプションを追加して、次のコマンドを使用した場合、17.1 コンパイラーで aocx ファイルのビルドにかかる時間は 1 時間以下になりました。

```
aocl -fast-compile device/vector_add.cl -o bin/vector_add.aocx
```

- **実行。**FPGA を搭載しているマシン上で実行する必要があります。Nimbix* はインテル® Arria® 10 FPGA を搭載した **BittWare** (英語) の A10PL4 ボードを使用しているため、名前に A10PL4 を含む [Application] を選択します。A10PL4 ボードに実際にアクセスできるマシンタイプを選択する必要があります。プルダウンメニューで、(CPU only) ではなく A10PL4 を含むマシンタイプを選択していることを確認します (**図 5**)。少なくとも 1 台の FPGA が有効なマシンを選択できるはずです。



5 [Machine type] プルダウンメニューから FPGA が有効なマシンを選択している例

- コマンドを使用して (**CL _ CONTEXT _ EMULATOR _ DEVICE _ ALTERA** を設定しないで) アプリケーションを実行します。

```
bin/host -n=10000
```

これで第 2 段階も完了です。正常に実行されると、次のようなメッセージが表示されます。

```
Initializing OpenCL
Platform: Intel(R) FPGA SDK for OpenCL(TM)
Using 1 device(s)
a10pl4_dd4gb_gx115 : A10PL4 (acla10pl40)
Using AOCX: vector_add.aocx
Reprogramming device [0] with handle 1
Launching for device 0 (10000 elements)
Time: 0.409 ms
Kernel time (device 0): 0.062 ms
Verification: PASS
```

aocl diagnose コマンドはシステムで利用可能なカードを調べるのに役立ちます。

```
aocl diagnose
aocl diagnose: Running diagnose from /opt/intelFPGA_pro/17.0/hld/board/a10pl4/
linux64/libexec
------------------------- acl0 -------------------------
Vendor: BittWare Inc

Phys Dev Name Status Information
acla10pl40 Passed A10PL4 (acla10pl40)
PCIe dev_id = 2494, bus:slot.func = 02:00.00, Gen3 x8
FPGA temperature = 52.6523 degrees C.
DIAGNOSTIC_PASSED
---------------------------------------------------------
```

2017 年 11 月以降の 17.1 ビルドを使用して新しい **`-fast-compile`** オプションを指定すると、コンパイル時間は大幅に短縮されます。これは、FPGA の非常に長いコンパイル時間を短縮するインテル独自の機能です。**`-fast-compile`** オプションを指定すると、コンパイル時間は大幅に (通常は 1/4 ~ 1/5 に) 短縮されますが、FPGA プログラムの実行効率はわずかに (通常は 10 ~ 20 パーセント) 低下します。複数のカーネルで構成される設計の場合は、変更されたカーネルを自動的に識別して変更点のみを再コンパイルする、インクリメンタル・コンパイル・オプションが利用できます。

OpenCL*: ハイパフォーマンス、オープンソース、広範なサポート

この記事では、OpenCL* による FPGA プログラミングの優れた点について概要を紹介しました。例えば、インテルの OpenCL* は、FPGA 上で実際の実行をプロファイルして結果を調べるためにフックをサポートしています。また、ホストと I/O チャネルを含む FPGA パフォーマンスを広範にサポートしています。チャネルは、I/O またはホストから FPGA カーネルへの直接データ・ストリーミングをサポートするため、インテルにより作成された OpenCL* 言語の拡張です。

この記事をお読みになった皆さんが、OpenCL* による FPGA プログラミングの新しい世界を探求されることを期待しています。インテルと Altera の Web サイトでは、無料のトレーニング・リソースを含む、多くのレポート、ツール、ライブラリー、および拡張を提供しています。

関連情報

- **インテル® FPGA SDK for OpenCL***
- **インテル® FPGA SDK for OpenCL* のドキュメント**
- **インテル® FPGA SDK for OpenCL* ベスト・プラクティス・ガイド** - OpenCL* で FPGA プログラミングを行う際の優れたガイド。レポートとその意味についての情報も含まれています。
- **Nimbix* クラウドでインテル® FPGA 開発ツールを使用するためのクイック・スタート・ガイド** (英語)
- **インテル® FPGA ソフトウェア・デザイン・センター** - コードサンプルをダウンロードできます。
- **インテル® FPGA プラットフォーム** - アクセラレーター・ボードのリンクが含まれています。
- **OpenCL* 標準規格** (英語)
- **インテル® Cyclone® 10 LP ファミリー**、**インテル® Arria® 10 ファミリー**、および**インテル® Stratix® 10 ファミリー**の製品情報
- OpenCL* 2014 の国際ワークショップの OpenCL* + FPGA による gzip アクセラレーションに関する論文 : **Gzip on a Chip: High Performance Lossless Data Compression on FPGAs using OpenCL*** (英語)

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。