WP514 (v1.0) 2019 年 10 月 28 日

DNN 推論の最新動向と FPGA の位置付け

FPGA はデバイスをユーザー環境に完全に適応させるために必要な機能を備えており、任意のニューラル ネットワーク トポロジに合わせてコンピューティング アーキテクチャを調整するうえで基盤的な役割を果たします。

概要

アプリケーション要件がさまざまであることを反映し、最新の DNN 推論ワークロードおよびハードウェア アクセラレータ アーキテクチャでは、多様性と急速な進化が大きな傾向となっています。このホワイト ペーパーでは、アルゴリズムとアーキテクチャの最新動向を概観し、こうした環境の変化への適応性という観点から FPGA の位置付けを見ていきます。

はじめに

現在、さまざまな機械学習 (ML) アルゴリズムの登場によって新しい産業革命が起こっている中で、その中心的役割を果たしているのが、ディープ ニューラル ネットワーク (DNN) です。DNN はコンピューター ビジョンおよび音声認識において目覚ましい成果を上げており、その他の分野への応用も拡大しています。DNN は、まずラベル付きデータセットを使用して学習させた後、アプリケーションに組み込んで、未知のデータに対して推論を実行します。この推論の部分は「運用フェーズ」とも呼ばれます。このホワイト ペーパーでは主にこの運用フェーズについて論じます。

DNN の運用には非常に多くの演算とストレージが必要とされるため、アクセラレーションが欠かせません。しかも、精度、コスト、消費電力、モデル サイズ、スループット、レイテンシに対する制約はユース ケースによって異なります。拡張現実 (AR)、ドローン制御、自動運転など、リアルタイム性が求められ、かつセーフティ クリティカルなアプリケーションは、レイテンシに対する要求が厳しく、データ転送のオーバーヘッドも生じるため、クラウドへのオフロードには適しません。クラウド コンピューティングや MLaaS (ML-as-a-Service) では、データセンターは天文学的な規模のデータを処理する必要があり、スループットの要求は高まる一方です。しかも、運転コストを最小化するために電力効率を改善することも重要な課題となっています。クラウド サービスにはエンベデッド アプリケーションほど厳しいレイテンシの要求はありませんが、対話型アプリケーションの場合はレイテンシによってユーザー体験が直接左右されます。たとえば Jouppi ら [参照 1] は、クラウド ベース サービスの対話型ユーザー体験における応答時間の上限を 7ms と見積もっています。

こうした課題を解決するため、近年、さまざまな種類の DNN モデルおよびアクセラレータが爆発的に誕生しています。アプリケーション要件がさまざまであることを反映し、最新の DNN 推論ワークロードおよびハードウェア アクセラレータ アーキテクチャでは、多様性が大きな傾向となっています。このホワイト ペーパーでは、DNN 推論ワークロードとハードウェア アクセラレータ アーキテクチャの最新動向を概観し、こうした環境の変化への適応性という観点から FPGA の位置付けを見ていきます。

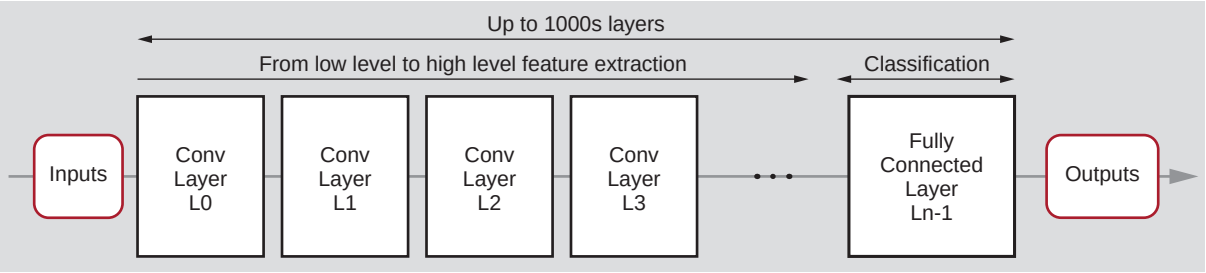

図 1: 基本的なたたみ込みニューラル ネットワーク (CNN) のトポロジ

ディープ ニューラル ネットワークの簡単な概要

一般に、DNN は 1 つ以上の層で構成されるフィードフォワード型の計算グラフであり、大規模なネットワークになると層の数は数百から数千にも達します。各層は多数のニューロンで構成され、層と層はそれぞれに重みが付けられたシナプスで相互に接続されています。各ニューロンがそれぞれの受容野の重み付き合計を計算した後、非線形の活性化関数が実行されます。一般に、コンピューター ビジョンの場合はたたみ込み層を使用します。たたみ込み層の受容野は、通常 2 次元の特徴マップ複数個分のサイズがあり、通常 2 次元のフィルターを複数使用してたたみ込み処理を実行します。図 2 に、この演算の擬似コードを示します。

```
for each layer l
 for each output feature map plane o
  for each output feature map row r
   for each output feature map col c
    for each filter plane p
     for each filter row x
      for each filter col y
       A[l][o][r][c] += A[l-1][p][r-x][c-y] * w[l][o][p][x][y]
```

WP51_02_091919

図 2: 一般的な DNN 演算の擬似コード

PyTorch、TensorFlow、Caffe などの機械学習 (ML) フレームワークは、これらの計算グラフに基づいた表現を使用して計算をスケジューリングし、学習および推論用ハードウェアにマップします。

推論用 DNN モデルの傾向

従来の機械学習研究は、推論のコストを考慮せず、モデルの精度を高めることを追求していました。これは初期の ImageNet で優勝したネットワークで特に顕著で、AlexNet や VGG は現在の基準からすると非常に大型で、パラメーターの数が多い部類に属します [参照 2]。機械学習や DNN が実用的なアプリケーションへと広がっていく中、演算量とメモリ量が大きな課題となっています。これを受けて、最近では精度と演算の複雑さの両面から DNN の推論効率をどのように高めるかという研究が相次いでいます。

DNN の効率を高めるための手法

このホワイト ペーパーでは、DNN の効率を高めるために提案されたいくつかの手法について、簡単な概要を示します。これらの手法は互いに独立したものがほとんどで、複数の手法を組み合わせて使用することもできます。ただし、DNN の種類によっては適用が難しい手法もあります。

トポロジの効率化

DNN のトポロジとは、ネットワークを構成する層の数、各層のタイプとサイズ、各層の接続方法を定義したものです。トポロジによっては、1 個のトポロジ パラメーターに従って層のサイズと数を定義する構成ルールを使用して定義するものもあります。最近では、パラメーターの数および積和演算 (MAC) の数を削減しつつ高い精度を達成した DNN トポロジが、数多くの研究者から提案されています (MobileNets [参照 3]、ShiftNet [参照 4]、ShuffleNet [参照 5]、Deep Expander Networks [参照 6] など)。通常、これらのネットワークには、精度と演算量のトレードオフを制御するトポロジ パラメーターがあります。FPGA なら、シフトやシャッフルなどの新しい種類の演算子もデバイスのプログラマブル インターコネクトをリコンフィギュレーションするだけでインプリメントできるため、演算リソースをほとんど消費することがありません。これは、FPGA ならではの大きな優位性です。

量子化

一般に、DNN の学習には浮動小数点演算を使用しますが、学習に使用する値を制限することもできます。通常、これらは 8 ビットに直接量子化することも可能ですが (図 3)、使用するビット数をさらに減らして再学習させて、量子化済みニューラル ネットワーク (QNN) を得ることもできます。量子化の方式には一様量子化と非一様量子化があり、1 つのネットワークの部分ごとに量子化方式を変えることもできます。使用するビット数を減らすと演算量とメモリ量は大幅に減少しますが、精度も低下することがあります。最近は、学習時量子化手法の改良に関する研究が数多く発表されています [参照 7] [参照 8] [参照 9]。LQ-Nets [参照 10] など、最近の手法には浮動小数点と 4 ビット QNN の精度の差を 1% 未満に抑えたものもあります。

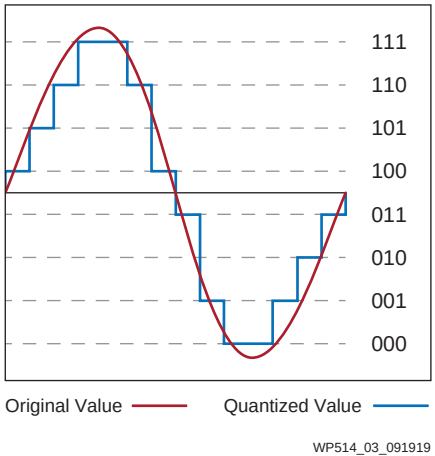

図 3: 正弦波の 3 ビット量子化関数

FPGA のプログラマブルロジックを使用すると、算術演算に使用するリソース量をアプリケーションで必要なビット数に合わせて非常にきめ細かくカスタマイズできます。これにより、アプリケーションで必要な演算およびメモリのハードウェア コストを削減できる可能性があります。

プルーニング

ニューラル ネットワークの一部を刈り込むことをプルーニングといい、層によっては精度を大きく低下させることなく最大 90% のネットワークを削減できます [参照 11]。図 4 を参照してください。プルーニング対象の選択基準 (重みの大きさ、二階微分値など) や、プルーニングの粒度 (個々のシナプス、隣接するシナプスのグループ、たたみ込みに使用する特徴マップ全体など) は、プルーニング手法によって異なります。個々のシナプスに対してプルーニングを実行すると不規則な構造となり、これを効率よく処理するには専用のハードウェアが必要とされます [参照 12]。プルーニングは多くの場合、粗粒度で実行されますが、細粒度のプルーニングの方が性能のスケーラビリティが向上します。FPGA であれば、演算エンジンの実装をサポートしながら、スパース表現を効率よく格納できるようにメモリ サブシステムを調整し、細粒度のプルーニングを利用できます。

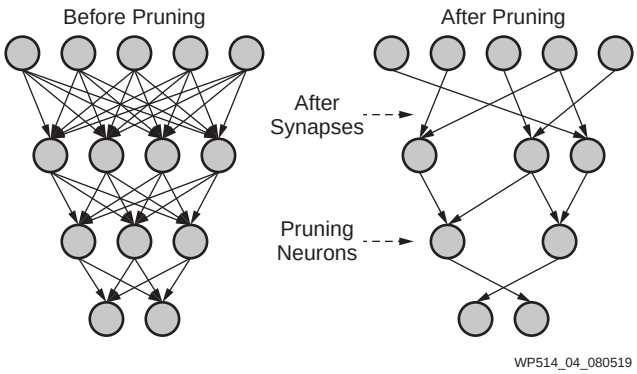

図 4: シナプスのプルーニング

層の融合と分解

DNN の各層の演算量およびメモリ量は、数学的等価性 (近似) を使って削減できます。たとえば、バッチ正規化の演算処理は、直前の線形変換層 (たたみ込み層または全結合層) に融合できます。たたみ込みは空間方向の分離可能なフィルターによって近似でき [参照 13]、全結合層は特異値分解 (SVD) によって近似できます [参照 14]。

その他の手法: 知識の蒸留と呼ばれる手法 [参照 15] を使用すると、効率的なモデルの学習が容易になります。Hoffer ら [参照 16] は、DNN の最後の分類層として値が+1/-1 の固定パターンとしたアダマール行列を使用する手法を提案し、ImageNet のいくつかのネットワークにこの手法を適用した結果、まったく精度低下が見られなかったことを報告しています。

精度と演算量のトレードオフ: 量子化の例

ニューラル ネットワークとは関数近似器であり、近似の品質が高くなるほどコストが増大します。つまり、ニューラル ネットワークを使用した推論の実行に必要なメモリおよび演算リソース量と、推論結果の品質 (ネットワークが未知の入力画像のクラスをどれだけ正確に予測できるか、など) はトレードオフの関係にあります。リソースと精度の関係を厳密に調べることは困難ですが、必要な演算リソースの異なる各種ニューラル ネットワークに学習をさせて、それぞれの精度を観察することで経験的に設計空間を探索できます (図 5)。

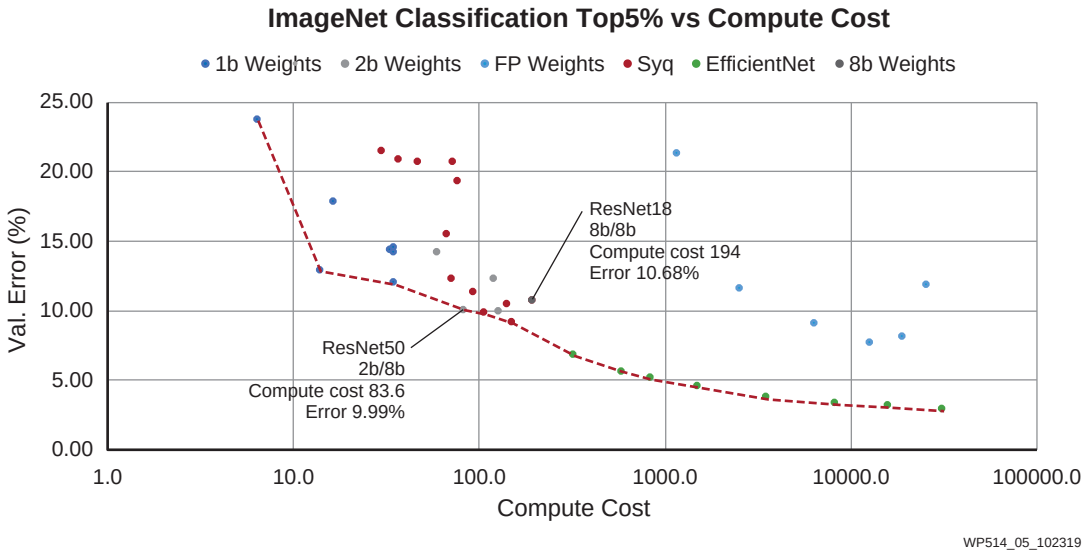

図 5: ImageNet における各種量子化ネットワークの演算コストと Top-5 分類エラーの関係

図 5 は、そのような設計空間探索の結果例を示したものです (図 5 は、x 軸 y 軸共に値が小さいほど良好)。適用する量子化方式が異なると、生成されるネットワークの演算コスト (x 軸。数値は、すべての演算に必要な FPGA の LUT および DSP スライスの概算値) と精度 (y 軸) もさまざまに変化します。赤の点線はパレート フロンティアで、演算コストと精度の条件が最も良いデザインを線で結んだものです。この例では、精度が低く「深い」ネットワーク (ResNet-50、重み 2 ビット、活性化関数 8 ビット) の方が、精度が高く「浅い」ネットワーク (ResNet-18、重み 8 ビット、活性化関数 8 ビット) よりも演算コストと誤認識率の両面で優れた結果を示しています。

推論アクセラレータ アーキテクチャの傾向

前述のとおり、ニューラル ネットワークでは非常に多くの演算量とメモリ量が必要とされます。たとえば、一般的な DNN の 1 つである ResNet-50 で画像分類を実行するには、1 つの入力画像に対して 77 億回もの演算が必要です。ただし DNN には並列性が非常に高いという性質があります。これらのアルゴリズムの運用を可能にするため、並列性という性質を利用したカスタム ハードウェア アーキテクチャが多数提案されています。

DNN の推論演算には、図 6 に示すようにさまざまなレベルの並列性が存在します。具体的には、次のような並列性があります。

- GoogLeNet や DNN アンサンブルに見られるような、隣接する層同士、および並列ブランチ同士の粗粒度のトポロジ並列性
- たたみ込み層の複数の入力/出力特徴マップ チャネルおよびピクセルなど、同じ層内のニューロンとシナプスの並列性
- 重みと活性化関数のそれぞれのビットを別々に見た場合の、演算内のビット レベルの並列性

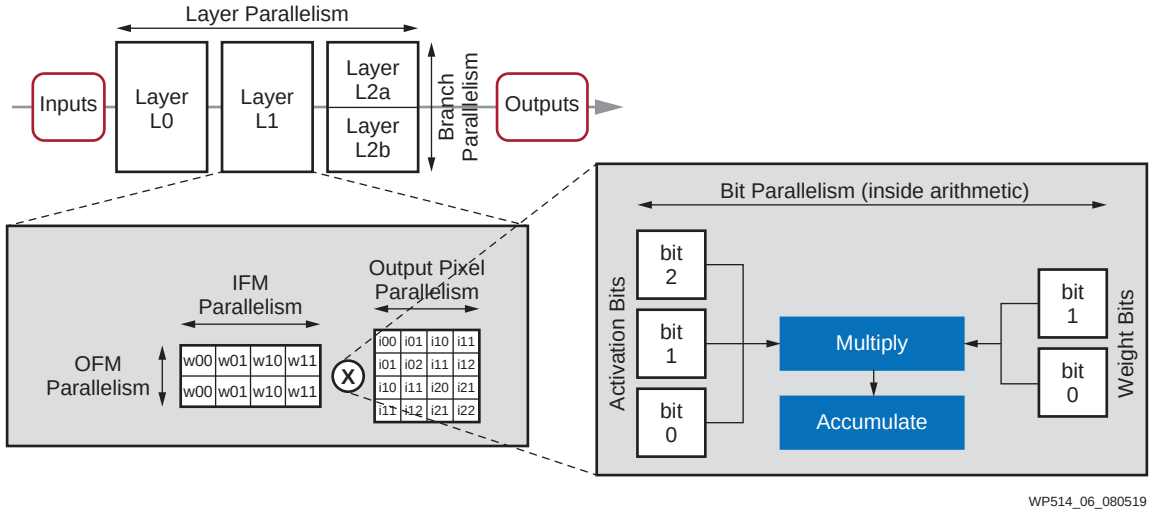

図 6: DNN の推論の演算に利用できるさまざまなレベルの並列性

推論アクセラレータ アーキテクチャの最新動向

このように非常に多くの演算およびメモリ リソースを必要とするアルゴリズムに対してハードウェア アーキテクチャを最適化しようとすると、次のような疑問が生じます。

- データ再利用と演算効率を最大限に高め、メモリ ボトルネックを最小に抑えるには、どのようにループを変換およびアンフォールドするのが最善か。
- テクノロジ ノードの微細化による性能向上がそれほど期待できない中で、どのようにして性能のスケーラビリティを確保するか。
- 消費電力の制約が厳しいエンベデッド アプリケーションで運用するために、どのようにして消費電力を抑えてリアルタイム応答を可能にするか。

標準的な CPU 以外に、GPU、FPGA、AI ASIC など多くの特化型のハードウェア アーキテクチャがアプリケーション固有の制約に合わせた最適化を試みています。このようにカスタマイズしたアーキテクチャを総称して、Microsoft は「DNN プロセッシング ユニット (DPU)」という用語を使用しています [参照 16]。図 7 は一般的な DPU アーキテクチャの概念図を示したもので、代表的な「弱点」を赤で示しています。

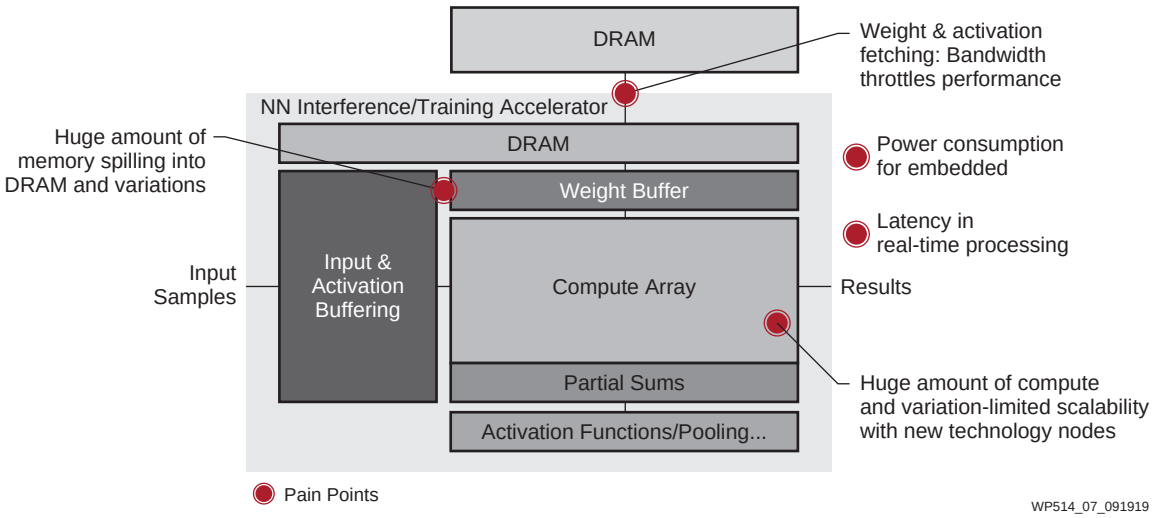

図 7: 一般的な DPU アーキテクチャとその「弱点」

アーキテクチャは、基本的な演算タイプ、メモリ帯域幅、並列性のレベル、特化の度合い、固有にサポートされる精度によって大きく分類できます。GPU はもともとゲームやグラフィックス処理を主要ターゲットとしていましたが、その後、しだいに高性能コンピューティング (HPC) でも採用が増え、今では AI への取り組み強化により、学習フェーズのアクセラレーションの事実上の標準となっています。GPU はベクター ベースの SIMD プロセッサと見なすことができ、Tensor コアを採用した Nvidia の Volta ファミリ [参照 17] で深層学習向けのカスタマイズが進み、Nvidia の最も新しい Turing アーキテクチャ [参照 18] によって固定小数点整数演算 (特に INT4 と INT8) へのカスタマイズが進んでいます。DPU をカスタム ASIC に実装して、ハードウェア コストの最小化と性能の最大化を図ろうとしているのが、Google の Tensor Processing Unit (TPU) [参照 19] です。TPU は単なるベクトルではなくテンソルに対する演算が可能で、前述の量子化を利用できるようにカスタマイズしたメモリ階層と算術演算をサポートしています。TPU 以外にも、このようなカスタム ハードウェアは多くの企業によって開発されています。この中には Arm のほか、Intel 傘下の 3 社 (Nervana、MobilEye、Movidius)、および GraphCore、Cerebras、Groq、Wave Computing など多数のスタートアップが含まれます。このように、推論アクセラレータのアーキテクチャはめまぐるしく変化しています。

FPGA を使用した高効率な DNN 実装の優位性

前述のように、DNN は各レベルでどれだけの並列性を利用できるかという点でも多様性があります。並列な演算素子を一定数だけインスタンシエートして、これらが一定の方法で通信するような固定されたハードウェア アーキテクチャでは、DNN の実行効率を引き上げようとしても限界があります。たとえば、入力特徴マップと出力特徴マップ (IFM-OFM) の並列性を利用できるように固定アーキテクチャを開発した場合、空間方向の分離可能なたたみ込みでは十分な使用率に達しない可能性があります [参照 2]。特に、高効率な DNN を作成する手法は急速に進化しつつあるため、DNN 推論を取り巻く状況が変化する中で高い効率性を維持するには、適応性が鍵となります。

その点、ザイリンクス FPGA は演算およびメモリ リソースの高い適応性と細粒度の超並列性が大きな優位性となります。ザイリンクス デバイスは幅広い種類の DPU アーキテクチャをサポートでき、さまざまなレベルの並列性を利用しながら、任意の DNN トポロジやアプリケーション固有のデザイン制約に合わせることが可能です。FPGA にソフト DPU を実装すると、メモリと算術演算を明示的に管理して任意のニューラル ネットワーク用にカスタマイズし、上記の構成をすべてサポートできます。

ソフト DPU の例

図 8、図 9、および図 10 に、ソフト DPU の例を示します。このように、ソフト DPU のアーキテクチャにはさまざまな種類があります。それぞれの図の下に、各アーキテクチャの主な特長を示します。

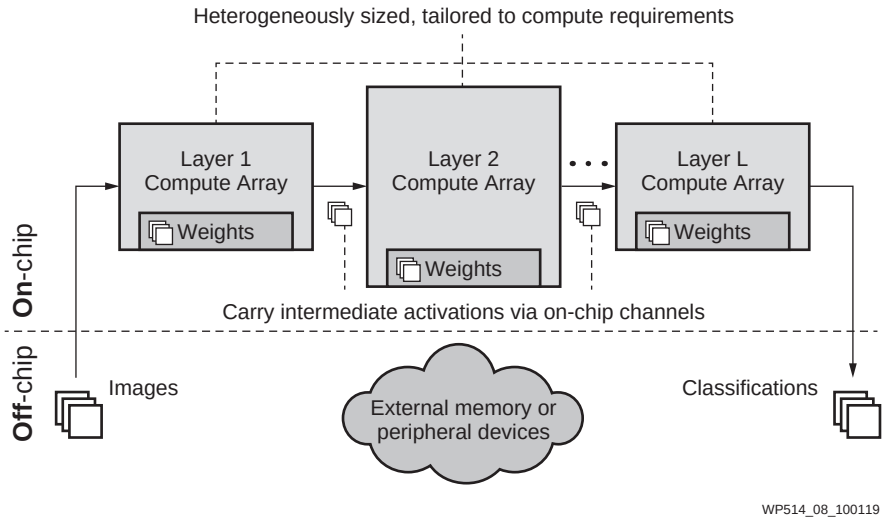

図 8: FINN — 層ごとに専用の演算リソースを使用し、層間にオンチップ データ フローを使用したソフト DPU

- **カスタム アーキテクチャ、複数層の並列性**

  FINN [参照 20] [参照 21] と呼ばれるツールに任意の QNN を入力すると、デバイス サイズに応じて各層を専用ハードウェアで実装し、オンチップ チャネルを使用して層と層を接続したカスタム DPU が生成されます。このアーキテクチャでは層ごとに精度と演算リソースを調整でき、効率のよいデザインを実現できます。層間のデータ フロー並列性を利用すると、低レイテンシと高スループットを達成できます。FINN はオープン ソースとして提供されています [参照 22]。

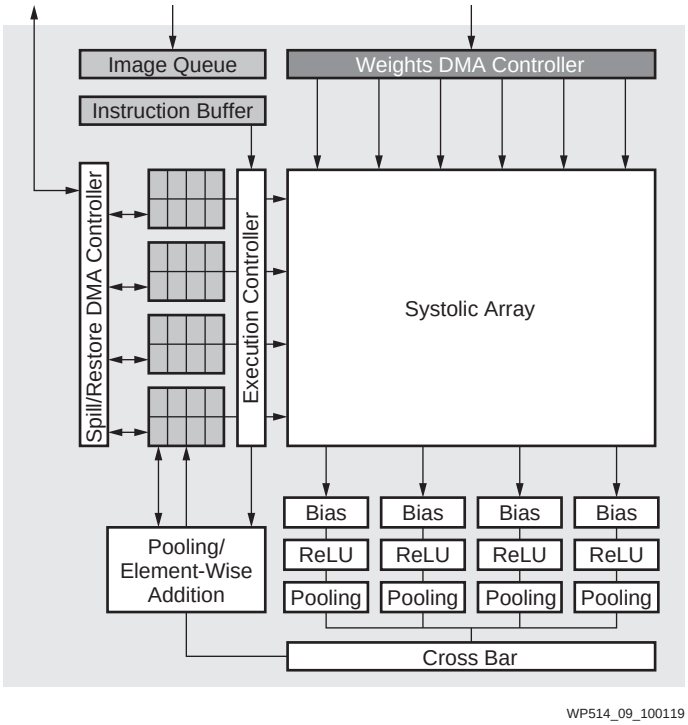

図 9: xDNN — 柔軟なプログラミングと高度な性能最適化が可能なソフト DPU

- **プログラマブルな汎用アーキテクチャ、単一層の並列性、固定精度**

  xDNN [参照 23] は、固定精度のシストリック アレイを使用したプログラマブル オーバーレイ アーキテクチャです。規則的な構造をしたこのアレイにより、高度な性能最適化が可能です。任意の DNN をこのアーキテクチャにマップするためのツール フローが用意されているため、新しいビットストリームを生成する必要はなく、FPGA に関する専門知識も不要です。xDNN は評価用として提供されています [参照 24]。

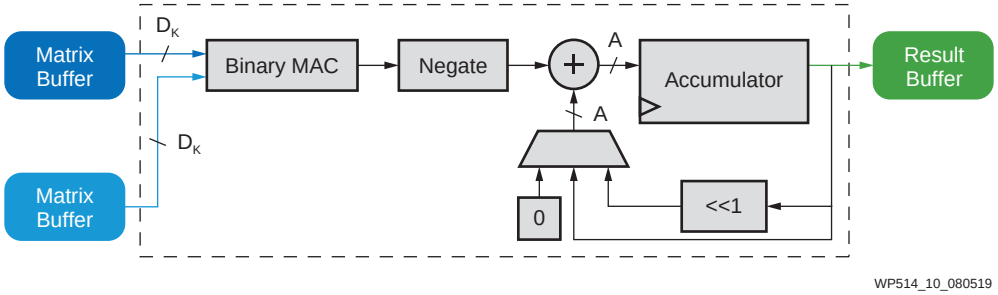

図 10: BISMO — リコンフィギュレーションなしで演算精度を変更できるソフト DPU

- **プログラマブルな汎用アーキテクチャ、単一層の並列性、精度を実行時に変更可能**

  BISMO [参照 25] [参照 26] は、ビット シリアル行列乗算のためのプログラマブル オーバーレイです。ビット精度の次元を逐次化すると同時に、その他の複数の次元を並列化することにより、実行時の可変精度をいかしながら高い性能を達成する固定アーキテクチャを可能にしています。精度の高い層ほど実行に必要なクロック サイクルが増大します。BISMO はオープン ソースとして提供されています [参照 27]。

ムーアの法則の終焉を契機とし、半導体業界の全体的な傾向に沿う形でザイリンクスは個々の垂直市場に特化した専用デバイスの開発を増やしています。これを可能にしているのが、ソフトウェア プログラマブルな AI エンジンとカスタム命令セットで構成された AI 用の革新的なコンピューティング ファブリックです。また、NoC ベースのインターコネクトにより、デバイス使用率を高めるうえで欠かすことのできないリソース配線の柔軟性が、かつてないレベルにまで向上しています。さらに、FPGA はニューラル ネットワークだけでなく、センサー フュージョンや柔軟な I/O もサポートし、コンピューター ビジョンの前処理と後処理を追加できるほか、ハードウェアにインテリジェンスを統合したり、デバイスをユーザー環境に完全に適応させるために必要な機能も備えています。

まとめ

機械学習アルゴリズムはさまざまな分野のアプリケーションへの採用が急速に拡大していますが、その演算負荷は従来のコンピューティング アーキテクチャにはきわめて大きなものとなっています。この課題に対処するため、半導体業界は DPU と呼ばれる革新的なアーキテクチャを数多く開発しています。こうした中、非常に高い柔軟性を備えた FPGA は、機械学習タスク全般に対してだけでなく、任意のニューラル ネットワーク トポロジに合わせてコンピューティング アーキテクチャを調整するうえで基盤的な役割を果たします。プログラマブル デバイスを使用して演算をカスタマイズすることによって、必要なストレージと演算リソースが最小限に抑えられ、性能のスケーラビリティの向上や、厳しいレイテンシ要件に合わせた最適化が可能になります。さらに、FPGA は柔軟な I/O やセンサー フュージョンをサポートし、コンピューター ビジョンの前処理と後処理を追加できるのも、ほかのデバイスにはない利点です。このような特長を兼ね備えた FPGA はその適応性によって、ユーザー要件に合わせたソリューションを提供します。

参考資料

注記: 日本語版のバージョンは、英語版より古い場合があります。

1. N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al., “In-Datacenter Performance Analysis of a Tensor Processing Unit,” in Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium, 2017.
2. Y.-H. Chen, J. Emer, and V. Sze, “Eyeriss v2: A Flexible and High-Performance Accelerator for Emerging Deep Neural Networks,” arXiv preprint arXiv:1807.07928, 2018.
3. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv preprint arXiv:1704.04861, 2017.
4. B. Wu, A. Wan, X. Yue, P. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. Gonzalez, and K. Keutzer, “Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions,” arXiv preprint arXiv:1711.08141, 2017.
5. X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. arXiv 2017,” arXiv preprint arXiv:1707.01083, 2017.
6. A. Prabhu, G. Varma, and A. Namboodiri, “Deep Expander Networks: Efficient Deep Networks from Graph Theory,” arXiv preprint arXiv:1711.08757, 2017.
7. A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, “WRPN: Wide Reduced-Precision Networks,” arXiv preprint arXiv:1709.01134, 2017.
8. J. Faraone, N. Fraser, M. Blott, and P. H. W. Leong, “SYQ: Learning Symmetric Quantization For Efficient Deep Neural Networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.
9. J. Choi, P. I.-J. Chuang, Z. Wang, S. Venkataramani, V. Srinivasan, and K. Gopalakrishnan, “Bridging the Accuracy Gap for 2-bit Quantized Neural Networks (QNN),” arXiv preprint arXiv:1807.06964, 2018.
10. D. Zhang, J. Yang, D. Ye, and G. Hua, “LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks,” arXiv preprint arXiv:1807.10029, 2018.
11. S. Han, H. Mao, and W. J. Dally, “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding,” arXiv preprint arXiv:1510.00149, 2015.

12. S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, “EIE: efficient inference engine on compressed deep neural network,” in Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium, 2016.
13. J. Guo, Y. Li, W. Lin, Y. Chen, and J. Li, “Network Decoupling: From Regular to Depthwise Separable Convolutions,” arXiv preprint arXiv:1808.05517, 2018.
14. E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, “Exploiting Linear Structure within Convolutional Networks for Efficient Evaluation,” in Advances in Neural Information Processing Systems, 2014.
15. G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” arXiv preprint arXiv:1503.02531, 2015.
16. E. Hoffer, I. Hubara, and D. Soudry, “Fix Your Classifier: The Marginal Value of Training the Last Weight Layer,” arXiv preprint arXiv:1801.04540, 2018.
17. Nvidia, Nvidia Volta and Tensor Core GPU Architecture.Available: https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/.
18. Nvidia, Nvidia Turing.Available: https://www.nvidia.com/en-us/geforce/turing/.
19. Google, Cloud TPU.Available: https://cloud.google.com/tpu/.
20. Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, “Finn: A Framework for Fast, Scalable Binarized Neural Network Inference,” in Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017.
21. M. Blott, T. Preusser, N. Fraser, G. Gambardella, K. O'Brien, and Y. Umuroglu, “FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks,” arXiv preprint arXiv:1809.04570, 2018.
22. Xilinx Research Labs, Machine Learning on Xilinx FPGAs with FINN, [Internet].Available: https://xilinx.github.io/finn.
23. ザイリンクス ホワイト ペーパー『ザイリンクス Alveo アクセラレータ カードを使用した DNN の高速化』(WP504: 英語版、日本語版)、2018
24. Xilinx, Xilinx ML Suite, [Internet].Available: https://github.com/Xilinx/ml-suite.
25. Y. Umuroglu, L. Rasnayake, and M. Sjalander, “BISMO: A Scalable Bit-Serial Matrix Multiplication Overlay for Reconfigurable Computing,” arXiv preprint arXiv:1806.08862, 2018.
26. Y. Umuroglu, D. Conficconi, L. Rasnayake, T. B. Preusser, and M. Sjalander, “Optimizing Bit-Serial Matrix Multiplication for Reconfigurable Computing,” in ACM Transactions on Reconfigurable Technology and Systems, 2019.
27. NTNU, Xilinx Research Labs,BISMO: A Scalable Bit-Serial Matrix Multiplication Overlay for Reconfigurable Computing, [Internet].Available: https://github.com/EECS-NTNU/bismo

関連資料

1. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.
2. S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra, “Why M Heads Are Better than One: Training a Diverse Ensemble of Deep Networks,” arXiv preprint arXiv:1511.06314, 2015.

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	内容
2019 年 10 月 28 日	1.0	初版

免責事項

自動車用のアプリケーションの免責条項

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。