**修士論文**

# 時分割多重実行型<br>線形アレイアクセラレータの開発と評価

菊谷 雄真

2019年3月14日

奈良先端科学技術大学院大学
情報科学研究科

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

菊谷 雄真

審査委員：

| | |
|---|---|
| 中島 康彦 教授 | （主指導教員） |
| 井上 美智子 教授 | （副指導教員） |
| 中田 尚 准教授 | （副指導教員） |
| Renyuan Zhang 助教 | （副指導教員） |
| Tran Thi Hong 助教 | （副指導教員） |

# 時分割多重実行型<br>線形アレイアクセラレータの開発と評価*

菊谷 雄真

## 内容梗概

ビッグデータやIoTという言葉が流行する昨今，膨大な数のセンサが収集した一次データをクラウドセンターに集約し学習・判断するシステムが次々に発表されている．エッジには低価格組込用マイクロプロセッサ，クラウドセンターにはGPUや大規模Field Programmable Gate Array (FPGA) を搭載することが主流である．しかし，このような中央集権型システムでは，ネットワークとクラウドセンターの負荷増大が招く応答性や可用性の低下が問題となる．そのため，エッジが学習・判断の一端を担い，それらの負荷を軽減することが必要である．そこで，エッジに適合する低価格・低電力・高効率な計算基盤が希求されており，盛んに研究されており，専用ハードウェア (Domain Specific Accelerator: DSA) やFPGAによる実装方法が数多く提案されている．しかし，前者は汎用性に欠け大量生産によるコストダウンが見込めず，後者は動作周波数が一桁低いという構造的問題を抱えている．上述の問題を解決できるアーキテクチャとして，Coarse Grain Reconfigurable Architecture (CGRA) 注目されている．ただし，従来型CGRAでは，2次元構造が基本であるため配線混雑が発生しやすく，小型化が難しい．また，要求性能に合わせてスケーラブルに性能向上するには，GPUを始めとするDMA Masterアクセラレータと同様に，チップ数の増加に見合った十分な帯域の外部メモリインターフェースを備える必要があり，同様に小型化が難しい．

本研究では，CGRAの演算ユニットを時分割4重実行と浮動小数点演算パイプラインの導入によって性能低下を抑えつつ1次元構造化（演算器数は1/4）し，さ

*奈良先端科学技術大学院大学 情報科学研究科 修士論文, 2019年3月14日.

らに，外部メモリインターフェースを増やすことなくマルチチップ化可能なリニアアレイアクセラレータ In-Memory Accelerator eXtension (IMAX) を提案する．まず，FPGA 上にプロトタイプシステムを開発し，動作周波数が 2 次元構造の同等機能 CGRA (EMAXV) の 50MHz に対し，150MHz に向上したことを確認した．そして，プロトタイプを用いて実アプリケーション (行列積，畳み込み演算，Light-field 画像処理) の実行性能を評価した．その結果，IMAX は EMAXV 対して 4.1, 7.3, 3.7 倍の実行性能を有することが明らかとなった．さらに，プロトタイプにおけるボード間通信のボトルネックを解消した場合，EMAXV に対して 6.3, 9.2, 6.6 倍の実行性能となる見積もりが得られた．そして，4 チップ構成の見積では，1 チップ構成に対して，プロトタイプの場合で最大 3.3 倍，ボトルネックを解消した場合の見積もりで最大 3.1 倍のスケールアウトが可能であるという見積もりが得られた．最後に，28nm テクノロジを利用した論理合成による回路面積の見積もりを行い，EMAXV の面積 27.40mm$^2$ に対し，8.40mm$^2$ で実装できることが明らかになった．ASIC においても同様の周波数比となることを仮定した場合，面積当たりピーク性能は，EMAXV に対し 2.44 倍となった．

**キーワード**

CGRA, シストリックリング，アクセラレータ, マルチスレッディング, エッジコンピューティング, ステンシル計算

# Development and Evaluation of a Liner Array Accelerator with Multithreading*

Yuma Kikutani

**Abstract**

In recent years when the word big data and IoT are prevalent, systems for collecting primary data collected by an enormous number of sensors in the cloud center and learning and inferencing are launching one after another. It is mainstream to mount low-cost embedded microprocessor on the edge, GPU and large-scale Field Programmable Gate Array (FPGA) in the cloud center. However, in such a centralized system, the reduction in responsiveness and availability resulting from increasing in the load on the network and the cloud center poses a problem. Therefore, it is necessary for edges to play a part of learning and inference and to reduce their load. Thus, low cost, low power, high efficiency computing infrastructure adapting to the edge is being demanded and studied vigorously, and a lot of implementation methods by dedicated hardware (Domain Specific Accelerator: DSA) and FPGA have been proposed. However, the former lacks general versatility and cost reduction due to mass production cannot be expected, and the latter has a structural problem that the operating frequency is one digit lower. Coarse Grain Reconfigurable Architecture (CGRA) has attracted attention as an architecture that can solve the above problem. However, in the conventional CGRA, wiring congestion tends to occur because the two-dimensional structure is basic and it is difficult to reduce the size. Further, in order to improve the performance

*Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, March 14, 2019.

scalability according to the required performance, it is necessary to provide an external memory interface of a sufficient bandwidth corresponding to increasing in the number of chips, as with the GPU and other DMA master accelerators. It is difficult to fabricate it.

In this research, we introduce CGRA operation unit into time division quadruple execution and introduce a floating-point operation pipeline to introduce one-dimensional structuring (the number of computing units is reduced to 1/4) while suppressing performance degradation. Further, increasing the number of external memory interfaces we propose a linear array accelerator In-Memory Accelerator eXtension (IMAX) that can be multi-chipping. First, we developed a prototype system on the FPGA and confirmed that the operating frequency improved to 150 MHz compared with 50 MHz of the equivalent function CGRA (EMAXV) with two-dimensional structure. We evaluated the performance of real applications (matrix multiplication, convolution operation, light-field image processing) using the prototype. As a result, it became clear that IMAX has execution performance of 4.1, 7.3 and 3.7 times EMAXV. Furthermore, when the bottleneck of board-to-board communication in the prototype is solved, an estimate of 6.3, 9.2 and 6.6 times execution performance was obtained for EMAXV. In estimating the 4-chip configuration, it demonstrated that the scale-out of up to 3.3 times in the prototype case and 3.1 times in the case of eliminating the bottleneck is possible for one chip configuration. Finally, we estimate the circuit area by logic synthesis using 28 nm technology and found that it implemented with 8.40 $mm^2$ for EMAXV area of 27.40 $mm^2$. Assuming that ASIC also has the same frequency ratio, the peak performance per area was 2.44 times compared to EMAXV.

**Keywords:**

CGRA, Systolic Ring, Accelerator, Multithreading, Edge Computing, Stencil Calculation

# 目 次

# 図 目 次

# 表 目 次

# 1. はじめに

## 1.1 背景

近年，機械学習アルゴリズムの発展が目覚ましいものとなっている．特に，Deep Convolutional Neural Networks (DCNNs) は画像処理をはじめとする諸分野でブレイクスルーを引き起こしている [1, 2, 3, 4]．DCNNs は，その性能と引き換えに膨大な演算が必要となるため，GPU を用いて演算の高速化を行う General Purpose GPU (GPGPU) が用いられることが一般的である．アルゴリズムの進歩のためには，プログラミング容易性が必要不可欠である．そのため，GPU はプログラミング容易性を保証するために，従来の計算機と同じノイマン型のプログラミングパラダイムを採用しながらも，多数のスレッドをマルチスレッド実行することで高い演算性能を実現しており，膨大な主記憶帯域幅が必要となる．GPU の消費電力の大きさは，演算性能とプログラミング容易性を両立することの代償であるといえる．一方で，機械学習が進歩するにつれて，組み込み機器であるエッジデバイスに対する演算性能の要求がますます高まっている．しかし，半導体プロセスの微細化によって支えられてきた計算機の演算性能の向上は物理的に困難になっている．そこで，特定のアプリケーションに特化したアクセラレータである，Domain-Specific Accelerator (DSA) の研究が盛んになっている [5, 6, 7]．汎用プロセッサは OS のような複雑なプログラムを動作させることができるのに対し，DSA は実行可能なタスクの領域は限定的であるが，非常に効率的かつ高速に動作可能である．しかし，DSA は特定のアルゴリズムを処理することを前提として設計されているため，新しいアルゴリズムの出現などに臨機応変に対応することができず，ライフサイクルが短いといった欠点がある．社会実装を考えた場合，新しいアルゴリズムやキラーアプリケーションに対応可能な，DSA よりもプログラマブルなアクセラレータが必要とされる．また，IoT のワークロードを担うこととなるエッジデバイスで活用する場合，狭バンド幅でもワークロードの規模に応じてスケーラブルな機構が望ましい．

## 1.2 目的

高い電力あたり性能を持つアクセラレータとして CGRA が研究されている．CGRA の一種であるシストリックアレイは DCNNs においても高い電力あたり性能を持つことが示されている [8]．DCNNs だけでなく，ステレオ画像生成や距離の取得ににも応用可能な Light-field 画像処理の高速化を行うことができるアクセラレータとして，我々は CGRA アクセラレータ EMAXV を提案してきた [9, 10, 11]．そして，EMAXV の演算ユニットを時分割 4 重実行と浮動小数点演算パイプラインの導入によって，性能低下を抑えつつ 1 次元構造化 (演算器数と局所記憶数は 1/4) した，リニアアレイアクセラレータである EMAX6 を提案してきた．

本研究では，FPGA を用いたプロトタイプによる EMAX6 の評価を行い，動作周波数の向上と起動に伴うオーバヘッドの削減が課題となることが明らかとなった．そこで，それらの課題を解決することで面積あたり性能をさらに向上し，加えて，EMAX6 の DMA Slave デバイスであるという特性を利用し，メモリインターフェース幅の限られるエッジデバイスにおいて，外部メモリインターフェースを増やすことなくマルチチップ化が可能なアクセラレータ IMAX の開発を目的とする．

## 1.3 構成

本章では，本研究の背景と目的を述べた．次に 2 章にて GPU や DSA，CGRA による社会実装に向けた DCNNs 実装の現状と課題を述べる．その後，3 章にて我々がこれまで研究を行ってきた EMAXV とその CNNs 向け拡張である EMAXVR，IMAX とその課題について説明する．そして，4 章にて明らかとした課題を解決するためのアイデアと手法について説明する．5 章では，IMAX のアプリケーションとして行列積，CNNs の畳み込み演算，Light-field 画像距離抽出を取り上げ，IMAX への効果的な実装について説明する．その後，FPGA を用いて構築した ARMv8 + EMAXV / IMAX プロトタイプについて説明し，それを用いたアプリケーション実行性能評価や，論理合成による面積の評価を行う．最後に，6 章を本論文のまとめとする．

# 2. 関連研究

背景で DCNNs の社会実装のために高い演算性能を持つ組み込み向けの計算デバイスが希求されているが，従来のノイマン型コンピュータのみでは実現が困難であることを述べた．そこで DCNNs の社会実装では全てを従来型計算基盤によりカバーするのではなく，GPU を用いてアルゴリズム開発および機械学習を行い，DSA や Field Programmable Gate Array (FPGA) を用いて学習データを利用した認識システムを実装および運用する垂直分業が確立されつつある．本章では，GPU や DSA および FPGA について説明した後，我々が着目しているアーキテクチャである CGRA について説明する．

## 2.1 GPU

汎用プロセッサはスーパースカラによって拡張された複数演算の同時実行機構の演算器使用率をさらに高めるため Out-of-Order (OoO) 実行機構と，複雑な分岐を持つコンテキストを高速に実行するため投機実行機構と分岐予測器を備えている．これらの機構は Re-order buffer (ROB) や分岐ヒストリテーブルといった多くのレジスタを必要とする回路が必要なため，面積的にも消費電力的にもオーバヘッドが大きい．しかし，背景で述べたような DCNNs や規模の大きい画像処理などは，ほとんどがループのアンローリングが可能な複雑な分岐を持たないプログラムであり，OoO 実行機構や分岐予測器の恩恵をほとんど受けない．ここでいう OoO 実行機構の恩恵を受けないの意味は，DCNNs のようなアプリケーションでは並列に実行可能な命令は静的に解決されるため，ROB が必要ないことを言っている．そこで，GPU は投機実行機構をあきらめ，その代わりに莫大な数の演算器を載せた構成を持つ．

図 1 に一般的な GPU の構造を示す．それぞれの演算ユニットは Single Instruction Multiple Data (SIMD) 演算を行う．複数の演算ユニットは 32 コアで一つのグループとして扱われ，共通の命令が発行される．GPU メーカーの最大手である NVIDIA はこのグループを Streaming Multiprocessor (SM) と呼び [12]，複数演算ユニットに同一の命令を発行する手法を Single Instruction Multiple (SIMT) 方式と呼んで

いる．簡単な分岐は条件付き実行命令でサポートしている．それぞれのコアで走るコンテキストはスレッドとして表現され，SM に発行されるスレッドの束は Warp や Wavefront と呼ばれる．GPU は ROB を持たないが Load 命令などで Warp がストールすると演算器稼働率が低下するためディスパッチャが異なる Warp を割り当て演算器を稼働させる．GPU は SIMT 方式とディスパッチャによって膨大な数の演算器を高い稼働率で動作させることが可能である．しかし，Warp のストールによるペナルティを隠蔽するためにコンテキストスイッチが必要であるため，深いパイプラインを持った演算器を持つことが不利になる．そこで，膨大な数の演算器へのデータ供給は莫大な広さを持つ主記憶帯域とデータのバースト転送効率を上げるためのコアレッシング機構，および大きなレジスタファイルとキャッシュによりカバーする．これらの機構のデータフローをハードウェアが制御する．さらに Warp はそれぞれプログラムカウンタを持つため，プログラミングが比較的容易である．しかしながら，組み込み機器への実装は面積，電力面でオーバヘッドとなるためトレードオフの関係である．そこで近年は DCNNs の学習を GPU で行い，学習結果の重みパラメータを用いて推論のみを低消費電力で高速に行う DSA が数多く提案されている．

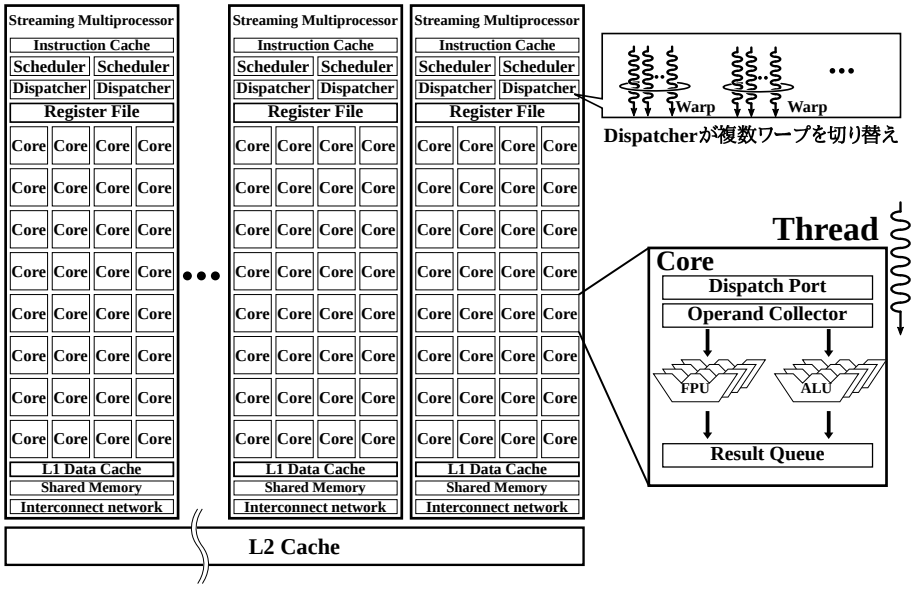



図 1: GPU の構造

## 2.2 DSA

深層学習の推論処理を専用ハードウェア (DSA) にて高速化する取り組みが流行している．推論専用 DSA の例として，Eyeriss[5]，FlexFlow[7] や Efficient Inference Engine (EIE) [6] などがあげられる．これらの特徴として，ディープラーニングの推論には演算の精度を落としても，認識の正誤率はそれほど下がらないことが示されているため [13]，重みの情報量を削減し 16bit-8bit の固定小数積和演算を行うことで演算性能，面積効率および電力効率を向上している．また，主記憶へのアクセスは計算システム全体の電力の約 25%を占めることが示されている [14]．そこで，多く DSA は畳み込み演算に特化した深い演算パイプラインとして動作するとにより，データを可能な限り再利用し主記憶へのアクセス回数を大きく削減する．

図 2 に DSA の一つである Eyeriss[5] の構成図を示す．Eyeriss では画像をランレングス圧縮することにより主記憶間のデータ転送量を削減している．これらの最適化によって面積と消費電力を大幅に削減し高い電力あたりの性能を持つことが分かっている．

しかしながら，DSA はその特性上応用範囲が限定的であるため，進歩の速い機械学習アルゴリズムの変化や，機械学習以外のキラーアプリの登場に対応することが出来ない問題点がある．さらに，半導体の製造コストはますます上がっていくと考えられ，新しい DSA が出てくるたびに新しい ASIC として設計しなすことは

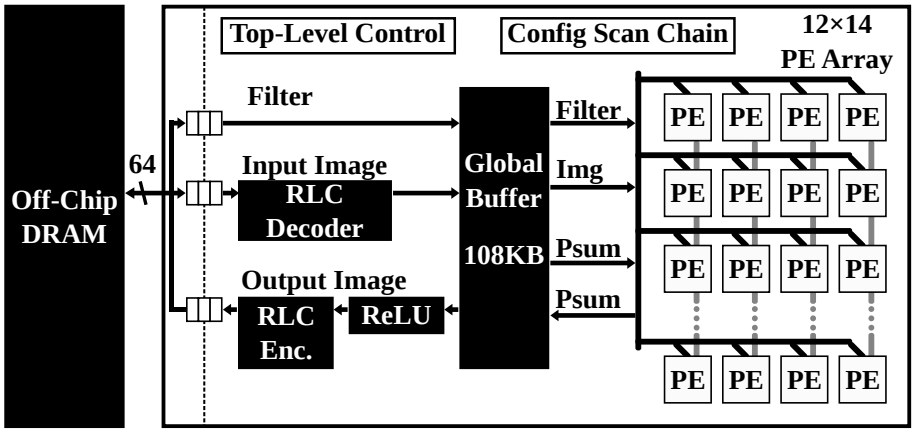



図 2: Eyeriss

開発コストを回収することが難しく，現実的ではない．こういった理由からGPUよりも省電力低コストでかつ，プログラマビリティを有するアクセラレータが必要である．

## 2.3 FPGA

FPGAは回路を何度でも変更可能な再構成可能デバイスである．図3に，FPGAの主流なスタイルであるアイランドスタイルFPGAの構成を示す．SRAMによって構成され，論理ゲートとして振る舞うLogic Block，外部とのI/Oを行うI/O Block，それらの接続を行うConnection BlockがFPGAの基本的な構成要素である．かつてはその動作周波数の低さから，専らASICのプロトタイプ開発や，性能を要求されないグルーロジックや万能ICとして用いられていたが，半導体プロセス技術の進歩に伴い，集積度と動作周波数が向上し，DSPが搭載されているものも多く，近年では計算用デバイスとしても活用されている．主記憶アクセスを最小限に抑える専用回路を構成できることから，CPUやGPUに比べて電力効率は大幅に優れているが，ASICとして製造されたDSAには及ばない．しかしながら，その汎用性から大量生産が見込めるため，LSI製造の初期費用の回収が比較的容易

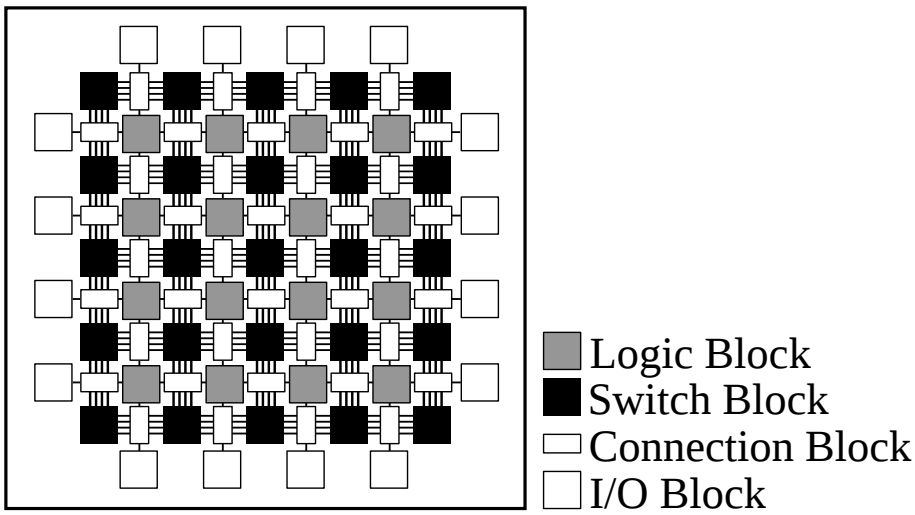



図 3: アイランドスタイルFPGA

であることから，最先端プロセスで製造されることが多く，ASIC に比べて最先端プロセスルールの恩恵を受けやすい．本研究では，我々の提案するアクセラレータのプロトタイプを FPGA に実装し，その動作の検証に用いた．

## 2.4 CGRA

CGRA はシストリックアレイ [15] を起源とするアーキテクチャで，FPGA と同様に再構成可能なアーキテクチャである．GPU より高い電力当たり性能を達成する計算デバイスとして CGRA は注目されている．図 4 に一般的な CGRA の構造を示す．CGRA は演算器およびレジスタ等からなる基本ユニットとメモリから構成されることが多い．再構成の粒度が FPGA ではゲートレベルであるのに対し，CGRA では演算器レベルであり，同等プロセスルールで同等機能を実現する場合，FPGA に比べて動作周波数や回路面積で優位である．さらに，FPGA を用いた開発では，Register Transfer Level (RTL) 記述からゲートレベルに合成したネットリストを実際の FPGA の構造に落とし込む配置・配線のプロセスがあり，大規模な最適化問題を解く必要があり，回路の規模と制約によっては合成と合わせて数十分から数十時間を要する．一方で，CGRA では上述のように再構成の粒度が粗く，探索範囲が大幅に削減できるため，コンパイル時間は数秒から数十秒程度である．そのため，CGRA は FPGA に比べて開発における turnaround time の面でも優位

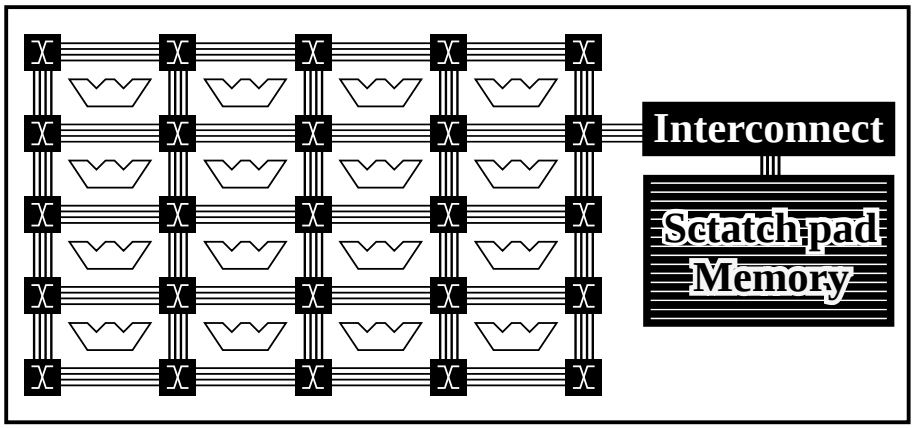



図 4: CGRA の構造

である．また，CGRA は GPU のような SIMD 型水平方向並列処理ではなくパイプライン型垂直方向並列処理であり，演算器群に毎サイクル供給すべきデータを抑えることができる．このため，大規模なメモリバスやコアレシング機構は不要であり，メモリバス幅を確保しづらいエッジにおいて有望なアーキテクチャである．

図 5 に CGRA アクセラレータの例として ADRES[16] のインスタンス例を示す．ADRES は VLIW view と CGA view から成る密結合アーキテクチャとなっている．VLIW contorl unit (CU) が CGA のループ制御を VLIW の関数コールとして行う．図 5 の例では，VLIW の命令発行幅は 4 命令であり，CGA section では 4x3=12 命令を発行可能である．CGA 実行時には，データの通信は VLIW Section の Functional Unit (FU) を通じて行う．

ADRES の他に CGRA アクセラレータとして，PipeRench[17]，CMA[18]，LAPP[19] 等が提案されている．また，シストリックアレイをコアとなる行列積エンジンのアーキテクチャに採用した Google の TPU[8] は，高い面積あたり性能および電力

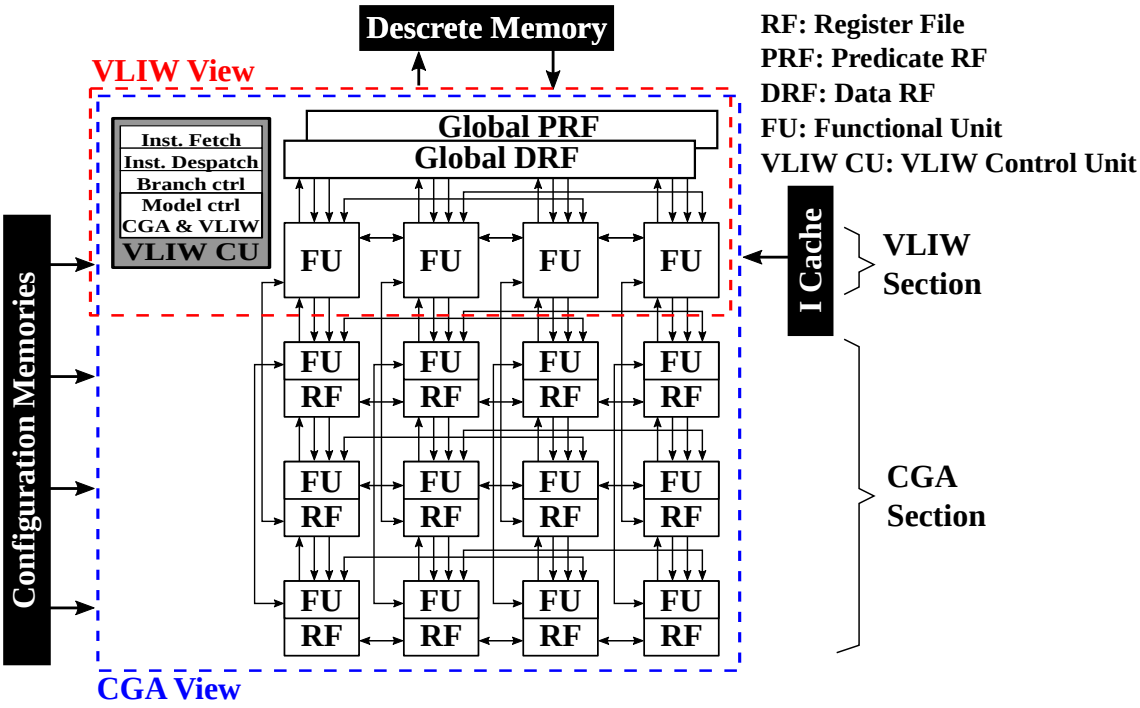



図 5: ADRES のインスタンス例

効率を達成し，話題となったことは記憶に新しい．

複数の CGRA を用いた関連研究としては，2017 年に開催された Hot Chips29 において Wave Comuting 社が発表した Dataflow Processing Unit (DPU)[20] があげられる．DPU はクラスタ化を前提とした高性能計算基盤向け CGRA アクセラレータであり，64-DPU のクラスタで AlexNet[1] 90 epochs の学習を 40 分で完了できると報告されている．本研究においても，スケールアウトによる性能向上を目指すが，サーバサイドを想定している DPU におけるスケールアウトは，広帯域幅のメモリインターフェースと Network on Chip (NoC) を使用しておりエッジデバイスには適さない．本研究では，エッジデバイスの挟メモリインターフェース環境下においてもスケールアウトが可能な機構を模索する．

# 3. 先行研究と課題

画像フィルタをはじめ，科学技術計算や近年流行している Deep Convolutional Neural Networks (DCNNs) などのアプリケーションで頻出するステンシル演算は，同一アドレスの参照回数が多いものの，一度の演算に使用するデータのアドレスが不連続となる．サイズの大きいステンシル演算や，Light-field 画像処理に用いられる離散ステンシル演算を扱う場合，一度の演算におけるアドレス参照範囲が大きく，局所記憶へのアクセスに膨大な帯域が要求される．特に，離散ステンシル演算はアドレス参照範囲がひときわ大きく，キャッシュラインよりも参照範囲が大きい場合は演算器稼働率が低下する．そこで，我々は，離散ステンシル演算のような複雑なアクセスパターンを持つ演算にも対応するために，演算器ごとにローカルメモリ (LMM) を配置した CGRA アクセラレータである EMAXV[9, 10, 11]，EMAXV をベースとして，DCNNs 向けの機能追加と改良を行った EMAXVR[21]，そして，EMAXV の演算ユニットをマルチスレッド化することで面積効率の改善を行った IMAX[22] を提案してきた．本章では，我々がこれまで提案してきた CGRA アクセラレータである EMAXV，EMAXVR および IMAX について説明する．

## 3.1 EMAXV

EMAXV の概要図を図 6 に，演算ユニットを図 7 に示す．EMAXV では組み込み機器の制約である限られた主記憶帯域においても演算強度の低い計算を高速化するため，データ再利用の重要性に着目し演算ユニット内部に Local Memory Module (LMM) とアドレス計算用 ALU (Effective Address Generator: EAG) を持つ．EAG を配置することにより，演算を行いながら主記憶-LMM 間のデータバースト転送をオーバーラップすることが可能となり，高い演算器稼働率を保ったままステンシル演算を行うことが可能である．EMAXV は AMBA AXI4 Bus インターフェース (FSM) を持つ DMA Master デバイスであり，FSM ユニット内部の DMA エンジンが主記憶-LMM 間の DMA 転送の調停を行う．演算ユニット間の接続は 4 ユニットを境界レジスタの共有バスとブロードキャストバスで相互接続し，stage という単位で扱う．stage 間を 2 次元メッシュの両端を接続したリングとして構築し，

演算ユニット内の接続を決定する Configuration resgister を stage 間で伝播することにより，回路構成情報の書き換え回数を削減する．stage の段数はスケールすることが可能である．

参考文献 [9] では，EMAXV へ DCNNs の畳み込み層を分割して写像し，高々 52stage を利用した演算で GPU を超える消費電力対性能を実現可能であることを示している．3×3 の畳み込みを行う層では 16 並列の畳み込み演算器として EMAXV を動作させている．これは，stage の段数が写像可能な演算の規模に強く影響しており，段数を拡張することで演算性能のスケールアップが可能であることを示している．

しかし，CGRA ベースのアクセラレータに対しその構造の抱える問題として，

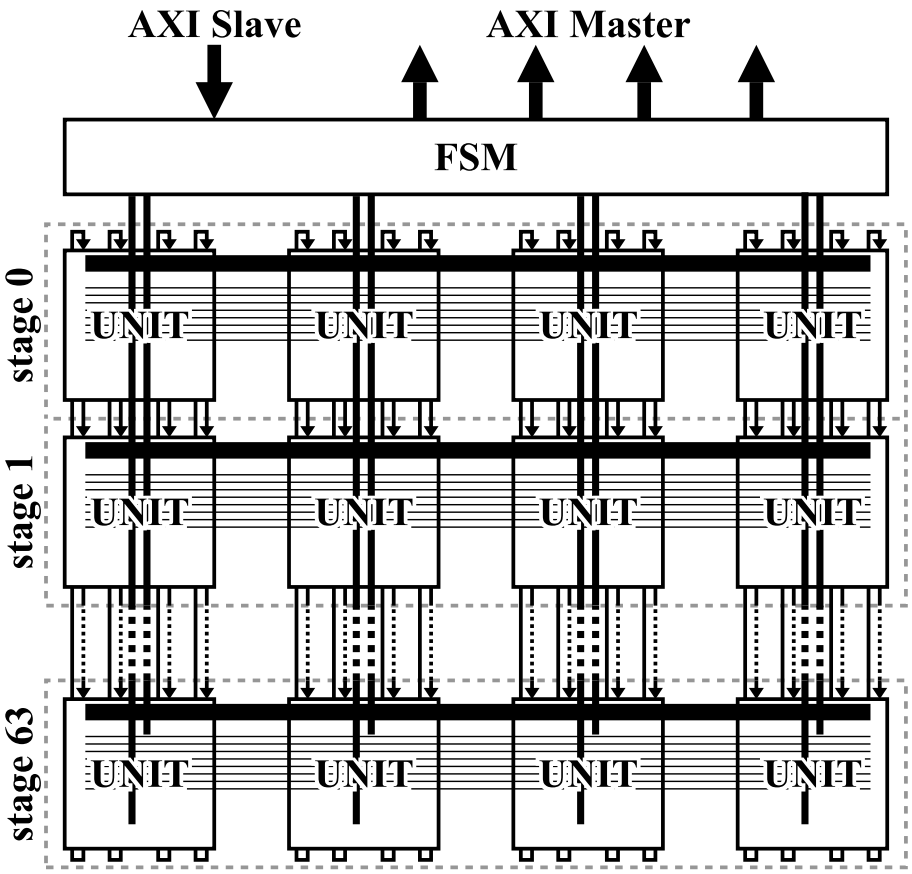



図 6: EMAXV の概略図

膨大な本数の配線が必要であり，CGRA の配線の複雑さから面積効率が低く，LSI への実装が難しいことが指摘されてきた．EMAXV でも同様にバスが配線負荷の原因となることが明らかになっている．配線負荷の大きい回路は LSI にする際の敷き詰め率を悪化させ回路面積を増大させる．特に半導体プロセスの微細化によるコストの減少を期待できなくなった今日では，回路面積はコストに直結するため，改善の必要性は大きい．また，ステンシル演算では不連続なアドレスへのアクセス効率を上げるためには LMM に多ポートのメモリが必要となる．ところが，メモリマクロは高々2 ポートである．そこで，EMAXV ではデータを同一 stage の複数の LMM にブロードキャストすることでアドレスが不連続なデータを一度に取り出し，最大で 8 ポートのメモリとして扱うことでこの問題に対処している．しかし，メモリマクロは回路面積の中でも大きなウェイトを占めるため，回路面積

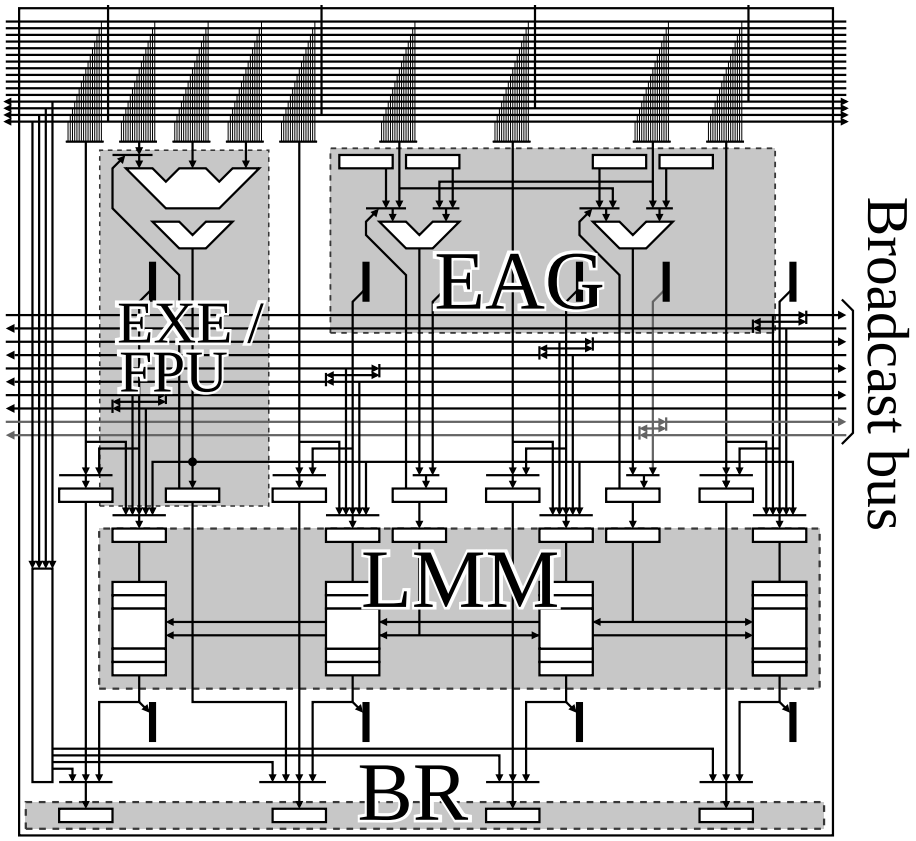



図 7: EMAXV のユニット

当たりのコストが減少しなくなった昨今，データのコピーを置くためだけに物理的に 4 つの LMM を用意することはコストの面から望ましくない．

## 3.2 EMAXVR

図 8 に EMAXVR[21] の概要図を示す．前節で紹介した DSA (Eyeriss[5]) は，CNNs に特化することで，高い演算器利用率を実現し，少ない演算器でも高い計算能力を確保している．また，データの再利用性を高めることで，外部メモリアクセスの削減にも成功している．外部メモリアクセスの電力消費は，回路内の演算やデー

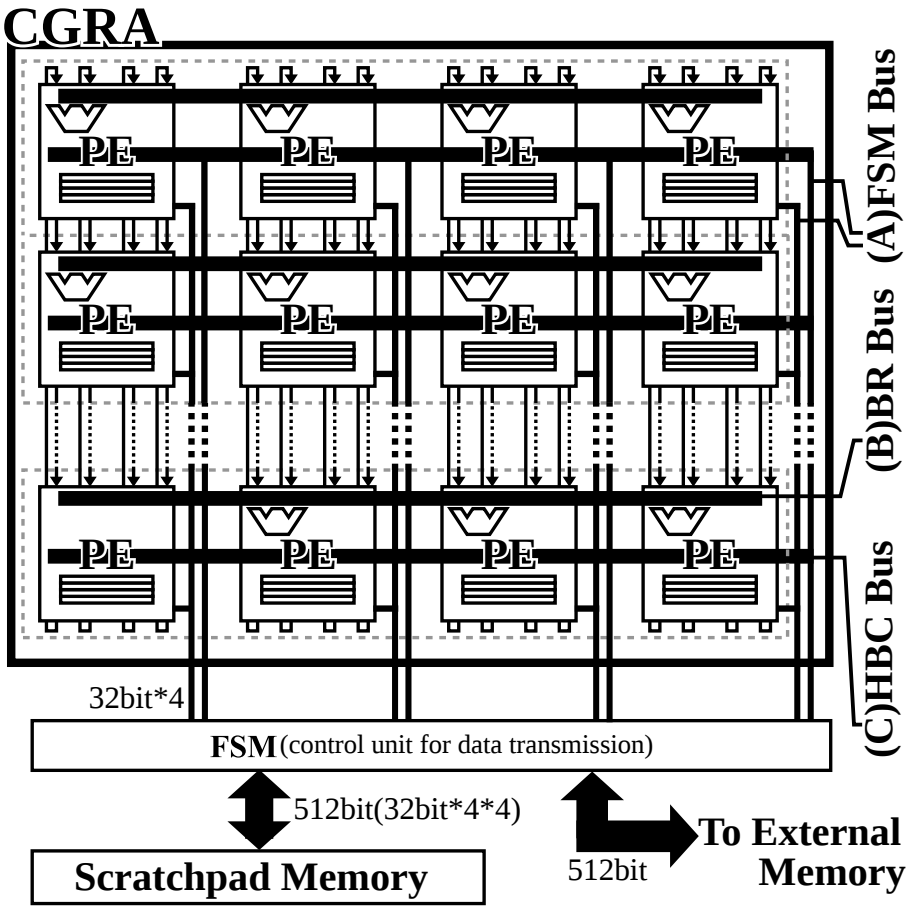



図 8: EMAXVR の概略図

タ移動と比較して非常に大きいこと知られている [14]．そのため，メモリアクセス量を減らすことで，消費電力を低く抑えることができる．そこで，EMAXV に対して，条件付き加算による多重ループ制御機構，stage 間ブロードキャストバスとスクラッチパッドメモリを加えることで，畳み込み演算時の演算器利用率とデータの再利用性を向上したアクセラレータである．参考文献 [21] では，Eyeriss[5] に対して，演算器利用率とデータの再利用性を示す外部メモリとの通信量がともに優れていることを報告している．

## 3.3 EMAX6

EMAXV と EMAX6 の比較を図 9 に，演算ユニットを図 10 に示す．EMAX6 は EMAXV の 1stgae を構成していた 4 ユニットを時分割多重実行によって 1 ユニットに統合し，4×N stages のアレイ構造から N stage のカスケード構造とすることで面積効率を向上したアーキテクチャである．この構造を従来型 CGRA に対し，Coarse Grained reconfigurable Linear Array (CGLA) と称し，提唱している．

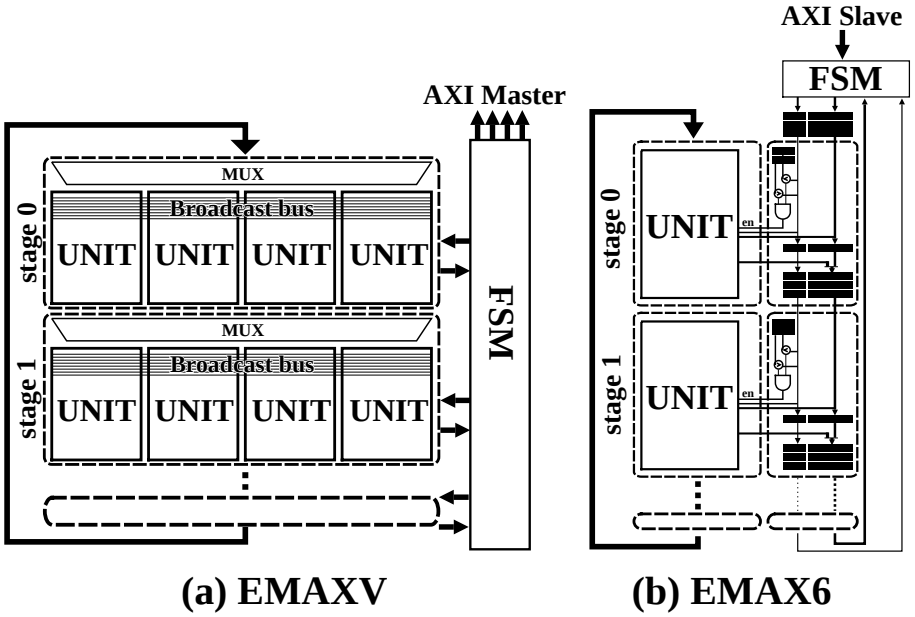



図 9: EMAXV と EMAX6 の比較

EMAX6 では，EMAXV において各ユニットの LMM にブロードキャストすることで実質 8 ポートのメモリとして実現していたところを，同一 LMM の時分割 Read が可能となり，1/4 の LMM で仮想 8 ポートメモリを実現できる．さらに，同一 stage の演算ブロック間でデータをブロードキャストする必要はなくなり，ブロードキャストバスも不要となり配線混雑が緩和された．また，EMAX6 では 4 サイクルの演算スケジュールに EMAXV の 4 ユニットそれぞれが行っていた演算を割り当てる．これにより，浮動小数点演算器をパイプライン化し，真の依存を持つ演算を 4 サイクルごとに行った場合でもバブルが発生せず，浮動小数点演算器のパイプライン化によって動作周波数を向上させることで，演算器およびメモリ搭載量を 1/4 に削減しながらも EMAXV からの性能低下を軽減した．

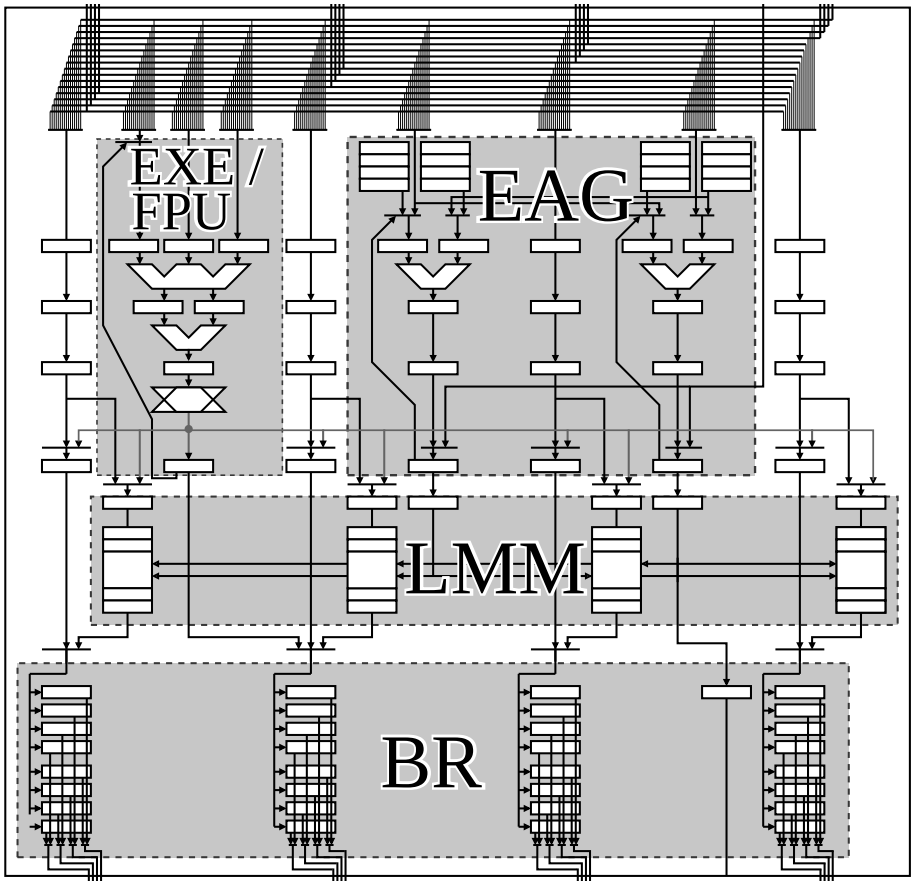



図 10: EMAX6 の演算ユニット

ARMv8 + EMAX6 環境におけるの動作シーケンスを図 11 に示し，各ステートの動作を表 1 に示す．CPU 側が Master となり，EMAX6 は Slave として動作する．まず，DMA(DRAIN) によって前回実行時のローカルメモリの内容を読み込み，CONF または SCON によって各ユニットへ命令をマップし，REGV でレジスタを初期化する．次に，RANGE によって有効となるローカルメモリを設定し，演算に必要なデータを DMA(LOAD) でローカルメモリへ転送し，EXEC で演算

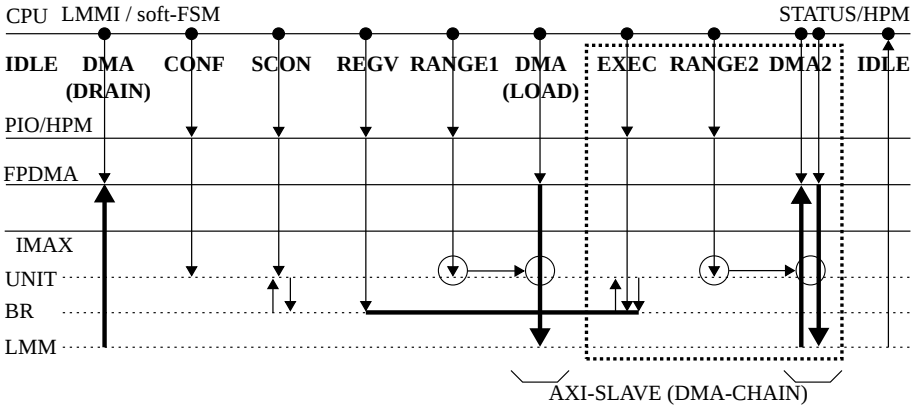



図 11: EMAX6 の動作シーケンス

表 1: EMAX6 動作シーケンス ステート一覧

| ステート名 | 動作の説明 | PIO/DMA |
|---|---|---|
| CONF | 命令マッピング | PIO |
| SCON | CONF 情報シフト | PIO |
| REGV | レジスタ情報設定 | PIO |
| RANGE | LMM 情報設定 | PIO |
| DMA(DRAIN) | DMA 転送 (LMM-主記憶) | DMA |
| DMA(LOAD) | DMA 転送 (主記憶-LMM) | DMA |
| EXEC | 演算実行 | PIO |
| IDLE | アイドル状態 | PIO |

を実行する．DRAIN，LOAD には PS の FPDDMA の DMA-CHAIN 機能を使用し，CONF，SCON，REG，RANGE，EXEC には PIO を使用する．また，CONF，SCON，REGV および RANGE は単独動作のみ可能であるが，EXEC と DMA は同時動作可能であり，DMA 転送と実行のオーバラップが可能である．

EMAXV は FSM に専用の DMA エンジンとコントローラを備えるアクセラレータとしては一般的な DMA Master デバイスであったが，EMAX6 では DMA Slave デバイスとすることで，Host に搭載されている汎用 DMA エンジンを利用する．この変更に伴い，図 9 に示すように，EMAXV の stage はすべて FSM と接続されていたのに対し，EMAX6 では 1stage あたり 2 サイクルをかけて stage 間を伝搬していくリング構造とし，目的の stage に到達した際にユニットが自律的にデータを取り込む仕組みに変更した．これにより，EMAXV で問題となっていた FSM の巨大なファンアウトが改善され，複数 stage に対するデータのブロードキャストが可能となった．また，ユニット制御情報の書き込みについて，ポストプロセッサの PIO によって前回起動時から変更の差分のみをピンポイントで書き込むことが可能となり，DMA バースト転送によって全体を書き換えていた EMAXV に比べて，初期化のオーバーヘッドを削減した．ただし，DMA Master デバイスからの Read に関しては，トランザクションの到着から N stages 構成の場合で 2N サイクルの遅延が生じる．

まとめると，EMAXV から EMAX6 への改良点は次のとおりである．

- 時分割多重実行によるユニットの削減
- 時分割多重実行による FPU のパイプライン化
- 時分割アクセスによる仮想多ポートメモリの実現
- 複数 stage へのブロードキャスト機能の追加
- DMA Slave デバイス化とそれに伴う FSM の小型化

## 3.4 EMAX6 の課題

前節で述べた EMAX6 について，評価を行った結果，次に挙げる課題が明らかとなった．課題 A: FPU をパイプライン化し，LMM の容量を削減した結果，パイプライン化されていない EXE (整数演算器) がクリティカルパスとなることが明らかとなった．そのため，新たに EXE のパイプライン化を行うことでクリティカルパスの削減が可能であると考えられる．課題 B: LMM からの Read が低速である．これは，FSM が EMAXV の全結合構造からリング構造となったことによって発生する，トランザクションの到着から N stages 構成の場合で 2N サイクルの遅延によると考えられる．課題 C: 起動に際するオーバヘッドが大きく，演算実行時間が短い．これは，起動時の初期化オーバヘッドは EMAXV や EMAX6 の仕様上避けられないため，EMAXVR で採られた多重ループ制御機構のように，そもそもの起動回数を削減する必要がある．

まとめると，EMAX6 の課題は次のとおりである．

課題 A: EXE がクリティカルパスであること

課題 B: LMM からの Read が低速であること

課題 C: 起動に際する初期化オーバヘッドが大きいこと

これらの課題を解決すべく，次章では改善に向けた手法を提案する．

# 4. 提案手法

前章では，我々が研究してきたCGRAアクセラレータについて説明し，時分割多重実行によって1次元構造としたCGLAアクセラレータEMAX6については課題について述べた．本章では，EMAX6の課題を解決すべく，改善のための手法を提案する．特に，課題Cについては EMAXVR[21] における多重ループ制御機構を参考に，異なるアプローチで解決を試みる．

## 4.1 動作周波数の向上

課題Aについて，EMAXVおよびEMAX6では，EXE (整数演算器) はEX1 (算術演算)，EX2 (論理演算)，EX3 (シフト演算) の3段から構成されており，命令の種別が異なればカスケーディングが可能である．これまで，EXEはクリティカルパスでなかったため，パイプライン化されていないなかった．ところが，前節で述べたようにFPUをパイプライン化することでクリティカルパスを削減し，また，アプリケーションを実装していくにつれてLMMの容量が削減可能であることが明らかとなった結果，パイプライン化されていないEXEがクリティカルパスとなることが明らかとなった．したがって，EXEのパイプライン化によって達成可能な動作周波数の向上が可能であると考えられる．

## 4.2 スケーラビリティの向上

課題B，課題Cとして述べた，EMAX6におけるオーバーヘッドの解決を試み，さらに，費用対効果に優れたスケールアウト手法として，複数チップ接続によるマルチチップ構成について説明する．

### 4.2.1 チップ内バス並列化と複数チップ Slave 接続機構

3章で述べた課題Bは，従来のIMAXではチップ内バスが1列に接続されているために発生する，2N (64 stages構成で128サイクル) の遅延が原因の一つであると考えられる．そこで，チップ内バスを8 stagesごとに分割し，8行×8列の構

成とし，分割したチップ内バスそれぞれから並列に読み出すように変更することで LMM からの Read の遅延を 1/8 に抑える．

また，エッジデバイスにおいて，狭小なメモリバス幅制約の下でいかに性能を確保するかが重要となる．一方で，米 AMD が Multi-Chip Module (MCM) 構成によって競合他社を寄せ付けない費用対効果を実現した Zeppelin アーキテクチャ [23] を発表し話題となり，世界中の大規模クラウドサーバを有する企業が導入を表明し，シェアを大きく拡大したことは記憶に新しい．このことからもわかるように，半導体微細化速度の鈍化と LSI 製造コストの増大が叫ばれる今日，マルチチップ構成はコストを抑えつつ性能をスケールアウト可能な有望な方法である．そこで，IMAX の DMA Slave 構造を利用し，メモリインターフェースを増設することなく，マルチチップカスケーディングにより性能を向上する手法を提案する．

GPU をはじめとするアクセラレータにおいて，DMA Master デバイスであることが通常である．DMA Master デバイスの場合，それぞれが独立したメモリ空間を持ち，ホストデバイスとの接続は，PCI-Express を始めとする高速 I/O インターフェースであり，マルチデバイス構成によって性能をスケールアウトしようとした場合，アクセラレータの数だけインターフェースが必要となり，単一ノードにおける大規模なスケールアウトは難しく，物理的制約の厳しいエッジデバイスには適さない．一方，DMA Slave デバイスである IMAX では，1 チップ分のメモリインターフェースに接続し，複数のチップを数珠つなぎにすることで，メモリインターフェースを拡張することなく，マルチチップ構成を実現できる．

図 12 に，2 チップ構成の場合の unit 間接続方法を示す．PIO/DMA 機能を有するホストおよび DDR と，チップ#0 およびチップ#1 が，1 組の AXI4 バスによりカスケード接続されている．64 個の unit を内蔵する各チップは，演算用に 1 列構成のリング接続（後述のように各ユニット通過時間は 8 サイクル），DDR-LMM 間転送用に 8 行 x8 列構成に分割したデータパス（同様に各ユニット通過時間は 2 サイクル）を備えている．演算用と DDR-LMM 間転送用を共有しないのは，演算と転送の同時動作に加え，チップ数増加時に問題となる 64unit 通過時間の大幅削減を狙うためである．AXI-WRITE-IF は，ホストからの書き込み要求をチップ内および次チップに伝搬する．また AXI-READ-IF は，チップ内 8 列および次チップからの応答を待ち合わせて結果を戻す．

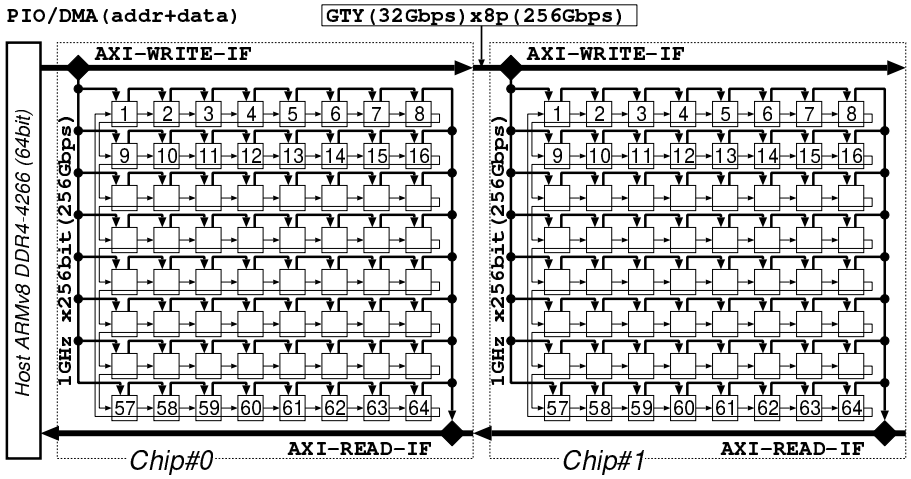



図 12: IMAX2 チップ構成

### 4.2.2 多重ループ一括実行機構

課題Bに対応するためには，IMAX起動回数の削減が必要である．そこで，EMAXVR[21]のように多重ループを一括実行することで起動回数を削減する．多重ループ一括実行のためには，複数のループ制御変数に対応できるアドレス計算機構と，多重ループ実行終了まで演算結果データをストアしない仕組みが必要である．多重ループ一括実行のための改良を加えたIMAXのユニットを図13に示す．

多重ループ制御を実装したC言語によるIMAX制御記述を図14に示す．EMAXV/E-MAX6では使用可能なループ制御変数は1種類のみで1重ループのみが使用可能であり，起動時にループカウンタにループ制御変数をセットし，それをデクリメントしていき0になった時点で実行終了としていた．そのため，1度の起動で演算を実行できる回数がループ1重分と少なく，その結果起動回数が増え，それに伴う起動のオーバーヘッドが問題であった．多重ループの一括実行に対応するには，複数のループカウンタを実装し，アドレス計算時に使用するループカウンタを選択できる機構が必要である．EMAX6ではユニットの4wayマルチスレッド化にあわせて演算器を4段パイプライン化しており，ループカウンタのデクリメントにそのままでは4サイクルかかる．しかし，デクリメント自体は1サイクルにて実行できるため，4サイクルを使用すれば4列分の依存関係のあるデクリメントが

可能である．そこで，LOOP0, LOOP1 をループ制御変数として定義し，LOOP0, LOOP1 をそれぞれループの終了判定に用い，全てが 0 になった際に演算を終了する．そのために，図 13 において赤色で示したループ終了判定結果を 1 サイクルで通知するためのパスを追加した．また，ループ先頭であることを示すフラグとして INIT0, INIT1 を実装し，アドレス計算時にこれらを用いてロードする値を選択し，多重ループにおけるアドレス計算を可能とした．3 重ループ一括実行を実装することも可能であるが，アドレス参照範囲が膨大になることが予想され，大容量 LMM が必要となり面積の増大に繋がることから，3 重ループ目はマルチチップ構成に対応付ける．

複数のループ制御変数に対応するアドレス計算機構について説明したが，従来

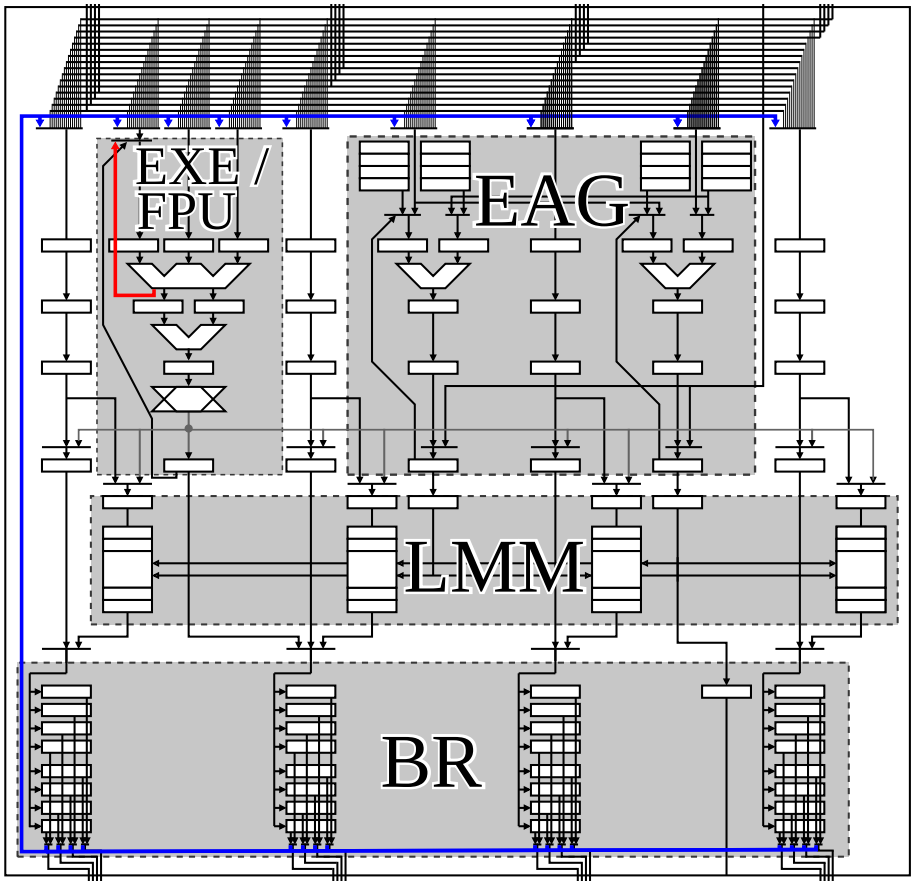



図 13: 改良した IMAX の演算ユニット

では一般的な CGRA と同様に，演算結果や LMM からロードしたデータは下段へ伝搬することしかできず，行列積や畳込み演算では，1 重ループ実行終了時に演算結果を一度主記憶にストアし，次のループ開始時にロードすることで部分和の累算を行っていた．IMAX は DMA Slave デバイスであるため，主記憶へのアクセスはホストによる制御が必要であり，アドレス計算機構が多重ループに対応しても，従来のままでは最内ループ実行が終了する度にホストへ通知し DMA を待つ必要がある．そこで，多重ループ実行が終了するまで主記憶へのストアを回避できる機構が必要である．EMAXVR[21] では，CGRA 部分の外側にスクラッチパッドメモリを設置することでこの問題に対処していた．一方，本研究では，ユニットの出力を同一ユニットの入力に戻す，図 13 にて青色で示したフィードバックパスを

```
/* BR: Boundary Register, AR: ALU output Register, W=4, M=480, RMGRP=8 */
/* LOOP0: loop counter for loop0, LOOP1: loop counter for loop1 */

/* stage#0 mapped to FOR() on BR[63][1][0] */
loop1: for (INIT1=1,LOOP1=RMGRP,rofs=0-M*4; LOOP1--; INIT1=0) {
   /* stage#0 mapped to FOR() on BR[63][0][0] */
   loop0: for (INIT0=1,LOOP0=M/W,cofs=(0-W*4)&0x00000000ffffffffLL; LOOP0--; INIT0=0) {
      /* calculation of address offsets */
      exe(OP_ADD, &cofs, INIT0?cofs:cofs, ..., W*4, ... ); /* stage#0 */
      exe(OP_ADD, &rofs, rofs, ..., INIT0?M*4:0, ... ); /* stage#0 */
      exe(OP_ADD, &oofs, rofs, ..., cofs, ... ); /* stage#1 */

      /* stage#1 ~ stage#62: FMA */
      /* load A and B */
      mop(OP_LDUWR, 1, &BR[1][0][1], (Ull)b000, (Ull)cofs, ... ); /* stage#1 */
      mop(OP_LDUWR, 1, &BR[1][0][0], (Ull)b001, (Ull)cofs, ... ); /* stage#1 */
      mop(OP_LDUWR, 1, &BR[1][1][1], (Ull)b002, (Ull)cofs, ... ); /* stage#1 */
      mop(OP_LDUWR, 1, &BR[1][1][0], (Ull)b003, (Ull)cofs, ... ); /* stage#1 */
      mop(OP_LDUWR, 1, &BR[1][2][1], (Ull)a00, (Ull)rofs, ... ); /* stage#1 */
      /* floating-point multiplication of A and B */
      exe(OP_FML, &AR[2][0], BR[1][0][1], ..., BR[1][2][1], ... ); /* stage#2 */
      exe(OP_FML, &AR[2][1], BR[1][0][0], ..., BR[1][2][1], ... ); /* stage#2 */
      exe(OP_FML, &AR[2][2], BR[1][1][1], ..., BR[1][2][1], ... ); /* stage#2 */
      exe(OP_FML, &AR[2][3], BR[1][1][0], ..., BR[1][2][1], ... ); /* stage#2 */
                :
      /* update partial sum of C with read-modify-write */
      mop(OP_LDUWR, 1, &BR[62][0][1], (Ull)c00, (Ull)oofs, ... ); /* stage#62 */
      mop(OP_LDUWR, 1, &BR[62][1][1], (Ull)c01, (Ull)oofs, ... ); /* stage#62 */
      mop(OP_LDUWR, 1, &BR[62][2][1], (Ull)c02, (Ull)oofs, ... ); /* stage#62 */
      mop(OP_LDUWR, 1, &BR[62][3][1], (Ull)c03, (Ull)oofs, ... ); /* stage#62 */
      exe(OP_FAD, &AR[62][0], AR[61][0], ..., BR[62][0][1], ... ); /* stage#62 */
      exe(OP_FAD, &AR[62][1], AR[61][1], ..., BR[62][1][1], ... ); /* stage#62 */
      exe(OP_FAD, &AR[62][2], AR[61][2], ..., BR[62][2][1], ... ); /* stage#62 */
      exe(OP_FAD, &AR[62][3], AR[61][3], ..., BR[62][3][1], ... ); /* stage#62 */
      mop(OP_STWR, 1, &AR[62][0], (Ull)oofs, (Ull)c00, ... ); /* stage#62 */
      mop(OP_STWR, 1, &AR[62][1], (Ull)oofs, (Ull)c01, ... ); /* stage#62 */
      mop(OP_STWR, 1, &AR[62][2], (Ull)oofs, (Ull)c02, ... ); /* stage#62 */
      mop(OP_STWR, 1, &AR[62][3], (Ull)oofs, (Ull)c03, ... ); /* stage#62 */
   }
}
```

図 14: IMAX の 2 重ループ制御記述例 (行列積)

追加することにより，同一ユニットの LMM から Read したデータを用いて演算を行い，LMM のデータを更新する Read-Modify-Write を可能とすることで，面積の大きい LMM を増設することなく多重ループを一括実行する．

### 4.2.3 アダプティブ命令シフト

EMAXV/EMAX6 は，再利用可能な LMM に合わせて命令写像を下方にシフトし LMM への転送を最小化できる機能を有している [24]．再利用の可否が不連続である場合，命令写像の下方シフト量をアドレス計算結果に応じてアダプティブに変化させる機構が必要である．アダプティブ命令シフトは，従来のコンパイラが連続実行起動前にホストが常時命令シフト指示を発行するコードを生成していた点を見直し，連続実行起動前に，現在の LMM のアドレス範囲と次の連続実行に必要なアドレス範囲を比較し，同一の場合に命令シフト指示を発行しないよう改良した機能である．コンパイラの改良であり，ハードウェアの変更はない．

# 5. 評価と考察

4 章では，CGLA アクセラレータ EMAX6 を改良した IMAX について説明した．本章では，IMAX の評価と考察を行う．まず，評価対象とするアプリケーションプログラムとその IMAX へのマッピングについて説明し，その後，IMAX の FPGA プロトタイプシステムについて説明する．そして，FPGA プロトタイプシステム上でアプリケーションを実行し，性能およびオーバヘッドの解析を行い，さらにマルチチップ構成にした場合の性能見積もりを行う．

## 5.1 アプリケーションプログラムの準備

CGLA (CGRA) である IMAX を用いたプログラムの高速化では，GPU 等の SIMD 方式のような水平方向の並列性を利用するだけでなく，データが上から下へと流れていく演算パイプラインとして，垂直方向の並列性も積極的に活用する．さらに，ループをアンローリングし，データの再利用性を高めるようにマップすることで，主記憶アクセスを削減する．

本節では，IMAX の評価対象アプリケーションとして，DCNNs の全結合層に使用する行列積 (MM)，同じく特徴抽出のための畳み込み層で使用する畳み込み演算 (CNN)，画像切り出し時の背景消去に使用するライトフィールド距離画像生成 (LF) を取り挙げ，IMAX への効率的なマッピングについて説明する．

### 5.1.1 行列積

行列積は古くから計算機での高速化が盛んに研究されているアプリケーションである．古典的なニューラルネットモデルであるパーセプトロンや DCNNs の全結合層も行列積で実装される．

図 15(a) に MM のデータ構造，(b) に C 言語による一般的な実装，(c) に各 unit に積和演算器を 1 つ備える H+1（height）x W（width）構成の IMAX の同一時刻におけるスナップショットを示す（M≥H, M≥W を仮定）．出力 C[row][col] に対応する A[row][*] x B[*][col] の積和演算を H 組ごとに分割し，1 列の unit[*mod H][col mod W] に対応させてパイプライン実行すると，HxW の全演算器を利用で

き効率がよい．例えば左端列では，結果 C0c, C08, C04, C00 に必要な乗算 A00 x B0c，A01 x B18, A02 x B24, A03 x B30 を同時刻に行い，最終段から C00, C04, C08, C0c, ..., C0(M-4) の部分和を毎サイクル出力し LMM に累算する．すなわち，HxW の演算スループットにより，毎サイクル W 個の部分和を出力する．これを M/H 回繰り返すことにより，完全な C00, C01, ..., C0(M-1) が得られる．(d) はさらに，配列 A を N チップに行分割して並列処理し，各 LMM が，配列 A の 1 行（M 要素）x GRP 行分と，配列 B の 1 行（M 要素）を収容できる場合の実装である．矩形内の 5 重ループが全 IMAX チップを 1 回起動する処理に対応し，chip ループの各イタレーションが，各チップにおける配列 A の GRP 行分と配列 B 全体の行列積に対応する．外側ループの blk が 0 から M-1 に増加する間，各チップの最終段を除く LMM は配列 A を GRP 行分保持し，最終段の LMM は Cx0, Cx1, ..., Cx(M-1) を GRP 行分保持しつつ更新する．一方，配列 B は，blk を更新する度に対応する H 行分を全チップにブロードキャストする必要がある．以上の演算に要する理論的サイクル数は，IMAX 起動時の遅延（H+1 サイクル）および LMM の入れ換え時間を除くと $M^3$/(HxWxN) となる．また，LMM が A，B，C 全体を収

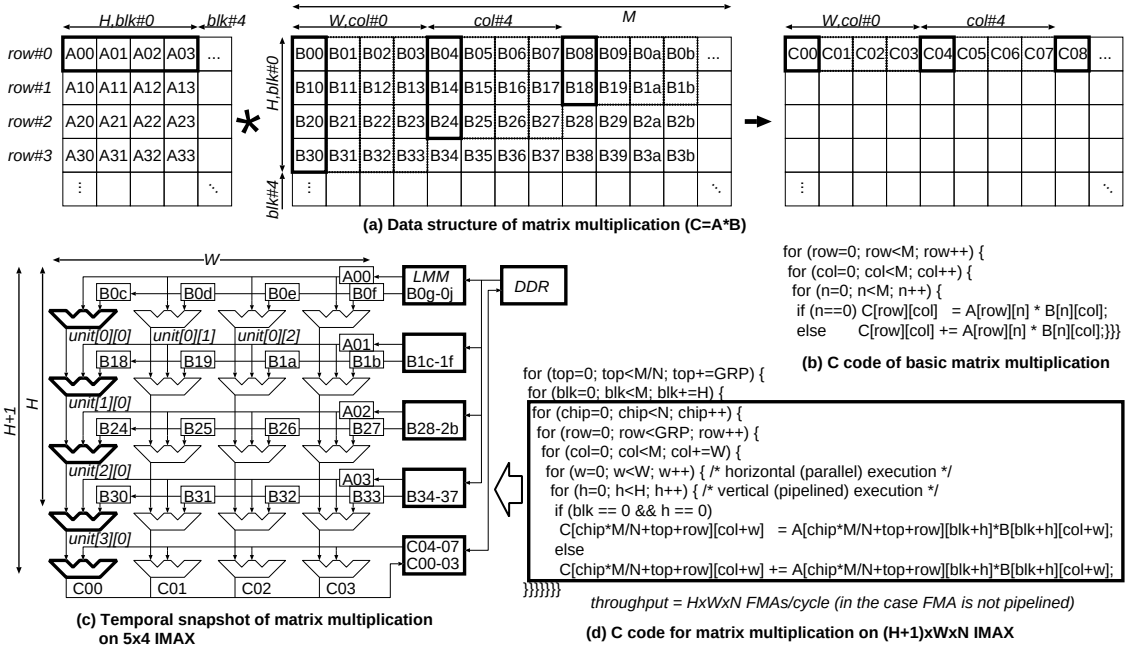
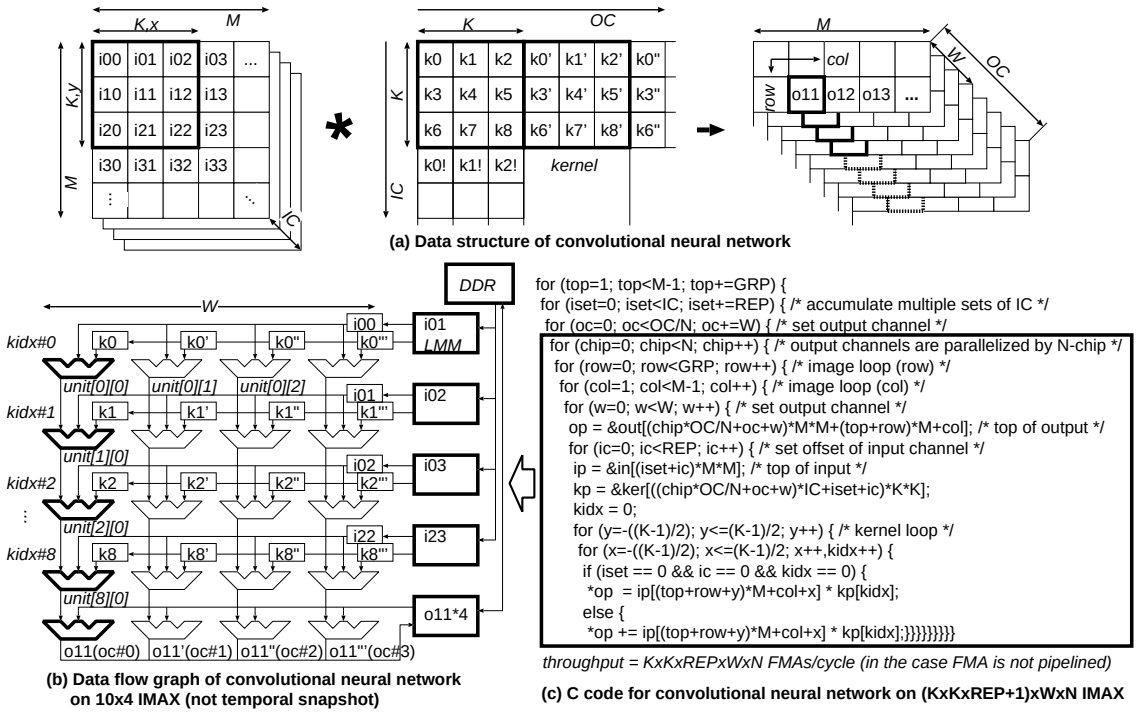



図 15: 行列積の IMAX への実装

容できる場合の理論的 DDR-LMM 間転送量が $M^2$x3 であるのに対し，各 LMM が A と C を MxGRP，B を MxH のみ収容できる場合の転送量は，A と C が各々$M^2$，B が $M^2$xM/GRP（M=GRP の場合 $M^2$ に一致．N に非依存．ブロードキャスト前提）と最適になる．

### 5.1.2 畳み込み演算

図 16(a) に，CNN のデータ構造，(b) に各 unit に積和演算器を 1 つ備える ($K^2$+1)xW 構成の IMAX を示す（M≥K, OC≥W を仮定）．(b) は簡単のために，同一時刻のスナップショットではなく，データフローグラフを示している．実際のハードウェアでは図 15(c) のように各段のデータがパイプライン処理される．CNN は，入力 in の一部（KxK）と kernel（KxK）の積和結果を全入力チャネル分累算し，出力 out に格納する．入力の総ワード数は $M^2$xIC，kernel の総ワード数は，$K^2$ に入力チャネル数（IC）と出力チャネル数（OC）を乗じた $K^2$xICxOC，出力の総ワード




**(a) Data structure of convolutional neural network**

**(b) Data flow graph of convolutional neural network on 10x4 IMAX (not temporal snapshot)**

**(c) C code for convolutional neural network on (KxKxREP+1)xWxN IMAX**

図 16: 畳み込み演算の IMAX への実装

数は $M^2$xOC となる．CNN に用いられる K は奇数であり，偶数が適するハードウェアパラメタ W に対応させると演算器稼働率が低下する．KxK の積和演算を 1 列の unit[*][oc mod W] に対応させてパイプライン実行し，W 個の出力チャネルを同時に計算すると $K^2$xW の全演算器を利用でき効率がよい．このとき，K 段の LMM に対する列方向ブロードキャスト機能を用いる．(c) はさらに，OC を N チップに分割して並列処理し，各チップが $K^2$ 段の積和演算を REP 組 xW 列写像できる段数を備え，さらに各 LMM が，in の 1 行（M 要素）x GRP 行分と，全 kernel（IC*OC*K*K 要素）を収容できる場合の実装である．矩形内の 7 重ループが全チップの IMAX を 1 回起動する処理に対応し，chip ループの各イタレーションが，各チップにおける REP 組の入力チャネルと kernel との畳み込み演算により，W 組の出力チャネルを GRP 行分計算する処理に対応する．外側ループの oc が 0 から OC/N-1 まで増加する間，各チップの最終段を除く LMM は in を GRP 行分保持する．一方，最終段の LMM は oc を更新する度に入れ替える必要がある．以上の演算に要する理論的サイクル数は，IMAX 起動時の遅延（$K^2$xREP サイクル）および LMM の入れ換え時間を除くと，$K^2$x$(M-2)^2$xICxOC/($K^2$xREPxWxN) = $(M-2)^2$xICxOC/(REPxWxN) となる．また，LMM が in，kernel，out 全体を収容できる場合の理論的 DDR-LMM 間転送量が $M^2$xIC + $K^2$xICxOC + $M^2$xOC であるのに対し，各 LMM が in を MxGRP，out を MxGRPxW，kernel のみ全部収容できる場合の転送量は，in と kernel が各々$M^2$xIC と $K^2$xICxOC，out は 1 回分の入れ換え量が (MxGRPxW)x2，矩形外の回転数 (OC/N/W)x(IC/REP)x(M/GRP) に N を乗じた回数の入れ替えが発生するため，$M^2$x2xOCxIC/REP（IC=REP の場合 out は一度で計算でき入れ換え不要のため DDR 出力のみの $M^2$xOC．N に非依存）となる．

### 5.1.3 Light-field 画像処理

図 17 に Light-field 画像を示す．Light-field 画像は高解像度のイメージセンサとマイクロレンズアレイを用いて撮影された画像である．近年スマートフォンにも搭載されることが多くなったステレオカメラの発展として，撮影後に様々な演算を行うことにより，撮影後にリフォーカスをかけることや，仮想視点画像生成お

よび距離抽出が可能である．コンピュータの処理によって生成されたこれらの画像はコンピューテーショナルフォトグラフィと呼ばれ，様々な応用が期待されている．

図 18 に図 17 から距離抽出処理を行った画像を示す．Light-field 画像は隣接したマイクロレンズが結像した画像を集めて一枚の画像を生成する．そのため，参照するアドレスがマイクロレンズアレイ間の距離の分だけ離散的になる．汎用 CPU や GPU では，アドレスの参照範囲がキャッシュサイズを超えるとアプリケーションの実行性能が著しく低下することが分かっている．しかし，IMAX では離散的なアドレスに対して，多ポートメモリを利用することで毎サイクル必要なデータを取り出すことが可能である．

図 19(a) に，LF のデータ構造，(b) に各 unit に Sum of absolute difference（SAD）演算器を 1 つ備える 12x4 構成の IMAX を示す．(b) は図 16 と同様，データフローグラフを示している．75x75 画素のマイクロレンズ画像が 100x100 の格子点上に配置され，全体で 7500x7500 画素の画像（in）が得られると仮定する．各マイクロ画像（上下左右が反転）の一部（3x3）について，75+ofs 画素離れた複数のマイクロ画像間の SAD を求め，最小となる ofs を距離情報（out）として生成する．具体的には，中央（p0）の 3x3 画像のうち 6 画素と，周囲 6 箇所（p1）の 3x3 画像のうち 6 画素に対して SAD を求め，合計 36 画素の SAD を累算している．IMAX の先頭段（unit[0][*]）に，y#0 の位置における 3x3 画素の最上行（i#-1 に対応）を割り当

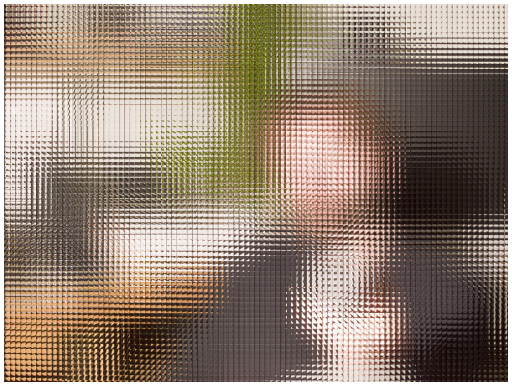


図 17: Light-field 画像

図 18: 距離抽出画像

て，1 行分の画像を保存する先頭 LMM から 6 箇所の離散的ロード（p0 と p1 を含む）を行う（点線矩形部分）．第 2 段（unit[1][*]）と第 3 段（unit[2][*]）は，各々，y#-1 および y#1 の位置（p1）からのロードを行い，先頭段にて取得した p0 と比較する．IMAX を 9 段通過した後に，さらに 2 段を用いて，4 列の SAD を 1 つの SAD に累算する．最外ループにおいて ofs を変更し，過去の最小 sad を保存しておき，より小さい SAD 値が得られた場合に，条件付きストアを用いて sad と out を

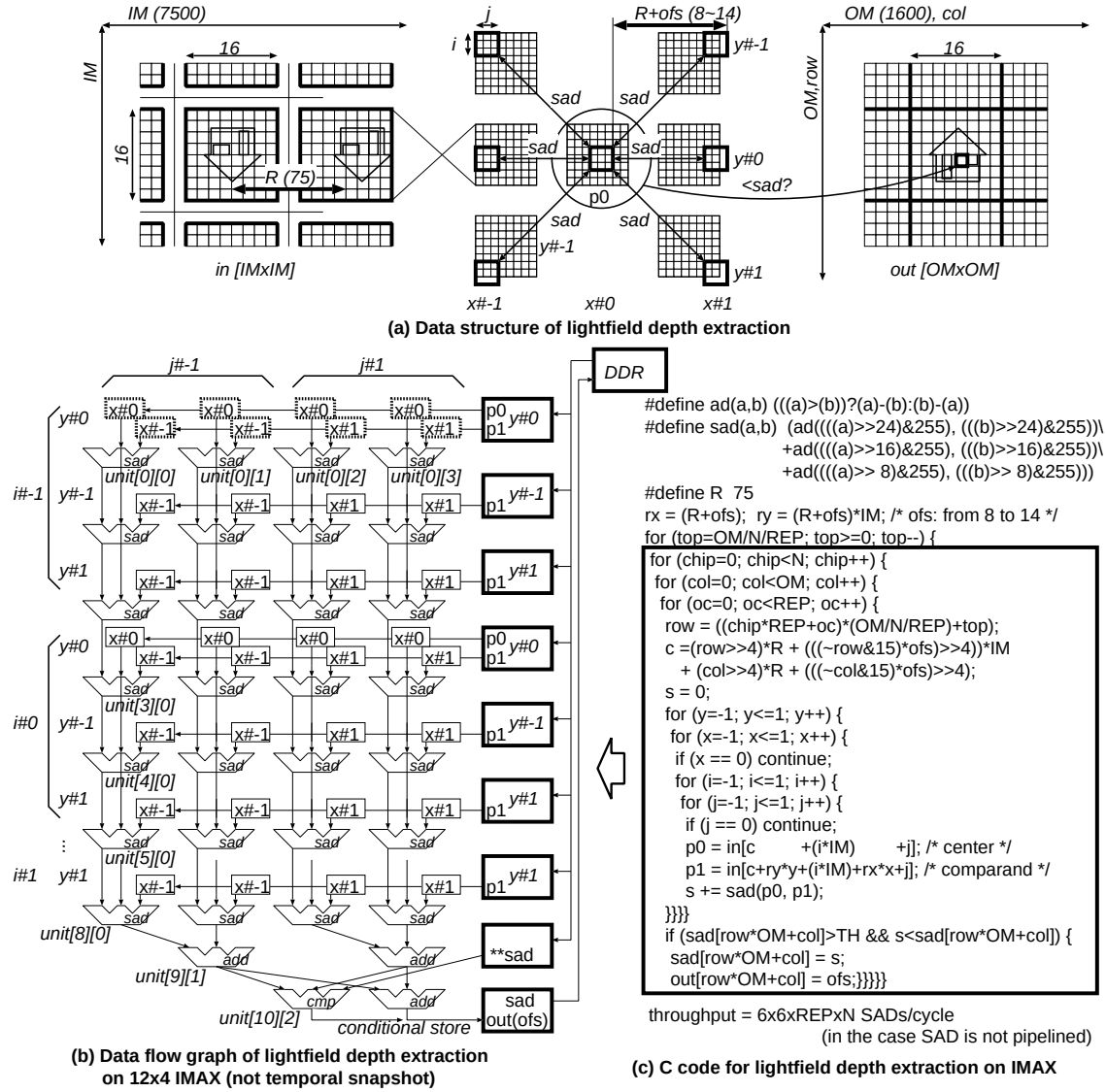



図 19: Light-field 画像距離抽出処理の IMAX への実装

更新する．このように，36 画素の SAD 値を求めるために 11 段を使用し，44 個中 40 個の演算器がアクティブとなる．全体の段数は，(b) に示したストア 1 段を含む 12 段に，c のアドレス計算等に必要な 6 段を前置した合計 18 段となる．(c) はさらに，画面を N チップに分割して入力画像を並列処理し，各チップが 36 画素の SAD 演算を REP 組写像できる段数を備える場合の実装である．なお ofs を更新する最外ループは省略している．以上の演算に要する理論的サイクル数は，IMAX 起動時の遅延（REP≥2 の場合 14xREP+3 サイクル）および LMM の入れ換え時間を除くと，ofs あたり $40xOM^2/(REPxN)$ となる．out を 1 画素生成するために p0 から 6 画素，p1 から 36 画素必要であるものの，p0 と p1 の水平方向参照は重複と考えると，理論的 DDR-LMM 間転送量は，$in:9xOM^2 + out:OM^2$ である．

### 5.1.4 アプリケーションプログラムの総括

表 2 に各アプリケーションの総演算数と，各アクセラレータにマップした際の占有 stage 数を示す．前述したとおり，MM は 480×480 の行列積であり，110M 回の Fused Multiply-Add (FMA) を行う．CNN は入力チャネル数 (IC)18，出力チャネル数 (OC)16，入力サイズ 242×242，カーネルサイズ 3×3 の畳み込み演算であり，152M 回の FMA を行う．LF は，7500×7500 pixel の Light-field 画像から 1600×1600 pixel の距離画像を生成するアプリケーションであり，93M 回の Sum of Absolute Difference (SAD) 命令を行う．このとき，IMAX(DLE) では RMGRP=8 としており，DLE を用いない場合に比べて起動回数は 1/8 となる．また，EMAX/IMAX では，アクセラレートしたい演算をループアンローリングしマップするが，その際に stage に余裕がある場合，複数のループイタレーションに相当する演算をマップし，演算性能を向上することができる．このときの単位となる演算のマップを set と呼び，64stages 構成の際に何 set をマップできるかを表 2 に示している．

図 20 に，1 チップの場合の理想的なダイアグラム（横軸が時間，縦軸が unit 番号）を示す．(a)MM における先頭 unit の連続実行サイクル数は，前述の 960 にマルチスレッディングの影響 4 を乗じた 3840，実行完了までのサイクル数は 64unit の通過に要する 64x8 サイクルを加えた 4352 である．これに B の DDR-LMM 転送（LOAD）時間 3600 と 8 行 x8 列の通過に要する 8x2 サイクルを加えた 7968 が blk

ループ 1 回あたりのサイクル数であり，M/H=8 倍したものにさらに A の LOAD 時間と C の LMM-DDR 転送（DRAIN）時間を加えた 64720（ホスト PIO によるレジスタ初期化（REGV）および LMM アドレス範囲初期化（RANGE）時間は除く）が top ループ 1 回あたりのサイクル数である．最外ループの回転数は，N=1 の場合，M/GRP=60 であるため，MM の理想的総サイクル数は 3.88M となる．

(b)CNN における先頭 unit の連続実行サイクル数は前述の通り 1920 にマルチスレッディングの影響 4 を乗じた 7680，実行完了までのサイクル数は 64x8 サイクルを加えた 8192 である．これに out の入れ換え（LOAD+DRAIN）時間 968+16+968+16 を加えた 10160 が oc ループ 1 回あたりのサイクル数であり，OC/W=4 倍したものにさらに in の LOAD 時間を加えた 42092（REGV+RANGE は除く）が iset ループ 1 回あたりのサイクル数である．iset および最外ループの回転数は，(M-2)/GRP=30 および IC/REP=3 であるため，CNN の理想的総サイクル数は 3.79M となる．

(c)LF における先頭 unit の連続実行サイクル数は 1600 に列方向マルチスレッディングの影響 4 を乗じた 6400，実行完了までのサイクル数は 64x8 サイクルを加えた 6912 である．これに sad,out の入れ換え（LOAD+DRAIN）時間 400+400+16 と in の LOAD 時間 33750+16 を加えた 41494 が top ループ 1 回あたりのサイクル数である．最外ループの回転数は，OM/REP=400 であるため，LF の理想的総サイクル数は 16.6M となる．

表 2: アプリケーションと命令マップ

| | MM | CNN | LF |
|---|---|---|---|
| 演算数 | $M^3$ | $K^2M^2\times IC\times OC$ | $40\times(OM\text{-}PAD)^2$ |
| | =110M(FMA) | =152M(FMA) | =93M(SAD) |
| stages / set | H+1=61 | 初段:$K^2$+1=10 | 初段:18 |
| | | 後続:9 | 後続:14 |
| sets / 64stages | 1 | 6 | 4 |
| 使用 stage 数 | 60 | 55 | 60 |

†MM: H=60, M=480

†CNN: K=3, M=242, IC=18, OC=16

†LF: IM=7500, OM=1600, PAD=32

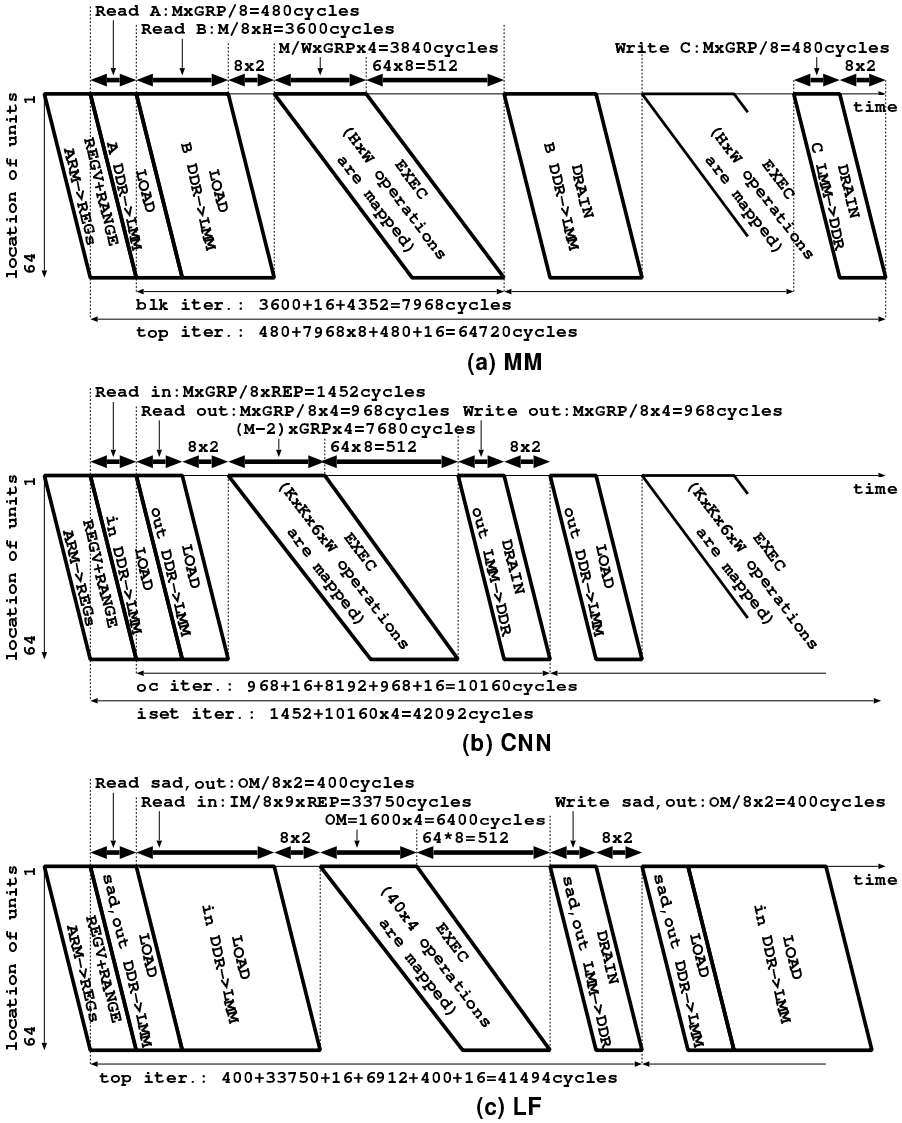



図 20: 理想モデルにおけるアプリケーション実行ダイアグラム

## 5.2 FPGA による性能見積もり

EMAXV / IMAX を RTL 記述し動作検証を行うため，FPGA SoC である Xilinx Zynq UltraScale+ ZU9EG を搭載する ZCU102 と，大規模 FPGA である Virtex Ultra Scale XCVU440 を搭載する S2C Single VU440 Prodigy Logic Module (VU440) を用いて ARMv8+IMAX のプロトタイプを実装した．本節では，FPGA SoC と大

規模 FPGA を用いた ARMv8+IMAX プロトタイプシステムの実装について説明する．プロトタイプのブロック図と実機の写真を図 21 および，図 22 に示す．図 21 では，矢印の始点が Master，終点が Slave としている．FPGA SoC である ZCU102 は，ARM を中心とする Processing System (PS) と，FPGA である Progmrammable Logic (PL) から成る．プロトタイプシステムは ZCU102 の ARM をホストとして，VU440 に実装した IMAX にアクセスする形をとっている．これは，ZCU102 の PL の規模では 64 段構成の IMAX を実装することが出来ず，大規模な FPGA が必要であったためである．ZCU102 - VU440 間は Xilinx の高速差動シリアル IO トランシーバーである GTH を 3 レーン用いて接続している．

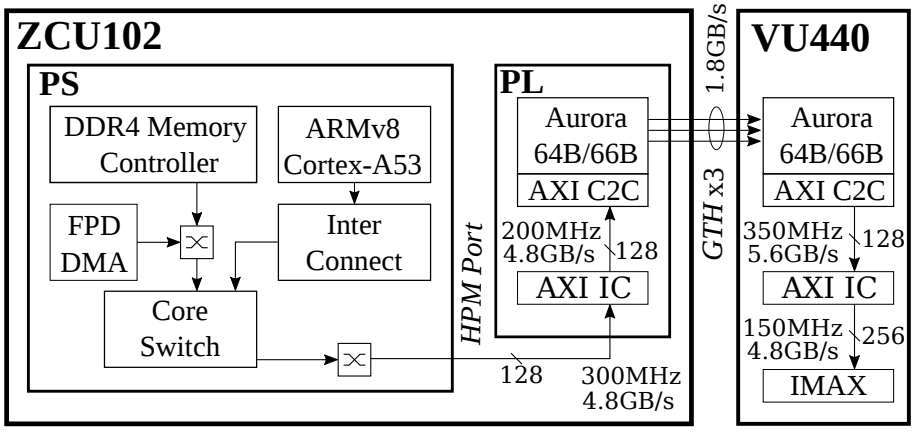



図 21: ARMv8 + IMAX FPGA プロトタイプシステムのブロック図

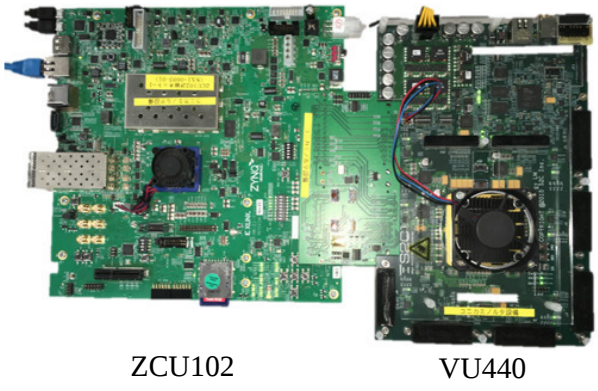



図 22: ARMv8 + IMAX FPGA プロトタイプシステム

プロトタイプシステムの開発に使用した開発環境と，合成・配置配線の Strategy を表 3 に示し，実装に用いた IP コアのうち，主要なものの一覧を表 4 に示す．これらの IP コアは，Xilinx 社から提供されており，Xilinx 社の FPGA を用いた開発において自由に利用することができる．

表 3: 開発環境と使用した Vivado Strategy

| | |
|---|---|
| ワークステーション | Intel Core-i9 7980XE<br>DDR4-2400 128GB CentOS 7.6 |
| FPGA 開発環境 | Vivado System Edition 2018.3 |
| Synthesis Strategy | Flow_PerfOptimized_high (EMAX6, IMAX)<br>Flow_AlternateRoutability (EMAXV) |
| Implementation Strategy | Performance_ExplorePostPhysOpt |

表 4: 開発に使用した主要な IP コア

| IP コア名 | バージョン |
|---|---|
| Zynq UltraScale+ MPSoC | 3.0 |
| AXI Interconnect | 2.1 |
| AXI Chip2Chip Bridge | 4.3 |
| Aurora 64B66B | 11.2 |

### 5.2.1 ZCU102 へのホスト機能実装

まず，ZCU102 側の FPGA デザインについて説明する．

PS には，FPDDMA という DMA エンジンが搭載されており，これを利用することで，広帯域幅な主記憶アクセスを実現する．ZCU102 の PL は，AXI Interconnect (AXI IC) を介して，チップ間通信を行うための IP コアである AXI Chip2Chip (AXI

C2C) に接続し，64B/66B エンコーディングで GTH を使用できる Aurora 64B/66B を用いて外部と高速 IO を行うことができるデザインとなっている．C2C は ZCU102 側が Master である．AXI IC は PS 側，C2C 側共に 128bit で接続，300MHz で動作しており，理論帯域幅は 4.8GB/s となっている．ボード間の接続は，C2C の制約から，1 レーン 5Gbps で動作する GTH を 3 レーン使用しており，64B/66B エンコーディングなので，理論帯域幅は 5/8[GB/s]×3×64/66=1.818...[GB/s] となり，このボード間通信が本プロトタイプにおけるボトルネックである．

### 5.2.2 VU440 への IMAX 実装

次に，VU440 側の FPGA デザインについて説明する．VU440 側も ZCU102 側と同様に，Aurora 64B/66B を用いて GTH を 3 レーン使用し，C2C から AXI IC を介して IMAX に接続するデザインとなっている．理論帯域幅は C2C - AXI IC 間が 128bit 接続 350MHz 動作なので 5.6GB/s，AXI IC - IMAX 間が 256bit 接続 150MHz 動作なので 4.8GB/s である．

VU440 に EMAXV，EMAX6 および IMAX を実装した際のリソースの使用率を表 5 に示す．EMAXV は論理合成および配置は可能であったが，配線できなかったため，動作周波数は N/A としている．ただし，EMAXV についてもアプリケーション実行評価を行うため，バス幅を 128bit 演算器を 32bit に半減し，各アプリケー

表 5: VU440 における FPGA リソース使用量

| | EMAXV | ENAX6 | IMAX | VU440 利用可能リソース |
|---|---|---|---|---|
| Fmax | N/A (配線不能) | 120 MHz | 150MHz | N/A |
| LUT | 2421366 | 1390086 | 1567474 | 2532960 |
| LUTRAM | 416 | 284 | 676 | 459360 |
| F/F | 298680 | 449309 | 457572 | 5065920 |
| BRAM | 2084 | 522 | 522 | 2520 |
| DSP | 2048 | 256 | 256 | 2880 |

ション実行に必要のない演算器（MM,CNN ではメディア演算器および EX2,EX3, LF では FPU）を廃することで VU440 上に実装した．このとき，動作可能周波数は MM, CNN 仕様で 50MHz，LF 仕様で 30MHz であった．IMAX は EMAX6 に対して，LUT12.8%，LUTRAM138%，F/F1.8%リソース使用量が増加した．このうち，LUTRAM について，増加率は大きいが絶対値は他のリソースに比較して大幅に小さい．また，IMAX の最大動作周波数は 150MHz であり，EXE のパイプライン化によって EMAX6 の 120MHz から 25%の向上に成功した．IMAX を実装した際のフロアプランを図 23 に示す．図 23 では，8 x 8stages に分割したグループごとに異なる色となるように彩色した．

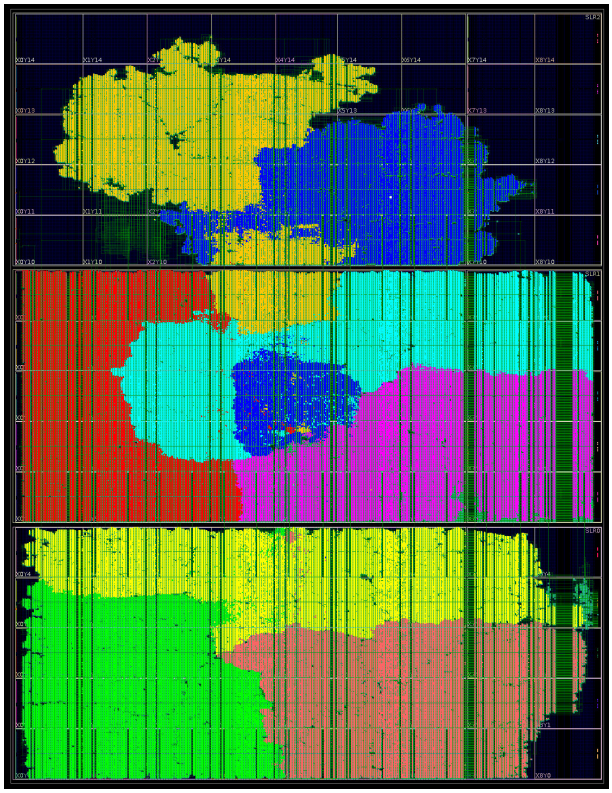


図 23: IMAX を実装した VU440 のフロアプラン

### 5.2.3 アプリケーション実行性能

前述したFPGAプロトタイプシステムを用いて，アプリケーション実行性能の評価を行った．アプリケーション実行性能評価における諸条件を表6に示す．評価はOS上で行うため，外部のアクセラレータとデータを共有するには，データの位置する物理アドレスが必要となる．そこで，CMA(Continuous Memory Allocator)を用いて，主記憶上に物理アドレス上に連続していることが保証されている領域を確保した．CMA領域に対してmemset()は使用できず，Segmentation Faultが発生するため，配列の初期化にmemset()が用いられるgccの最適化オプションO3以上は使用することができない．このため，CPUとアクセラレータの場合でコンパイルオプションが異なる．

図24に，アプリケーションを実行した際のEMAX6で正規化した演算性能，図25および図26にEMAX6の実行時間で正規化したステート (表1) 別プロファイリン

表 6: 評価条件

| | |
|---|---|
| OS | Debian 9.1 |
| カーネル | 4.9.0-xilinx-v2017.2 |
| コンパイラ | gcc version 6.3.0 |
| CPU | ARMv8 Cortex-A53 1.2GHz |
| メインメモリ | DDR4-2133 single channel |
| CPU評価時gccオプション | -Ofast -mstrict-align |
| GPU | NVIDIA Pascal 256 CUDA Cores (Jetson TX2) |
| GPU用コンパイラ | nvcc V8.0.72 |
| アクセラレータ動作周波数 | EMAXV: 50MHz (MM, CNN) / 30MHz (LF),<br>EMAX6: 120MHz, IMAX: 150MHz |
| アクセラレータ段数 | 64 stages |
| アクセラレータLMM容量 | 32KB / unit |
| アクセラレータ評価時gccオプション | -O2 -mstrict-align |

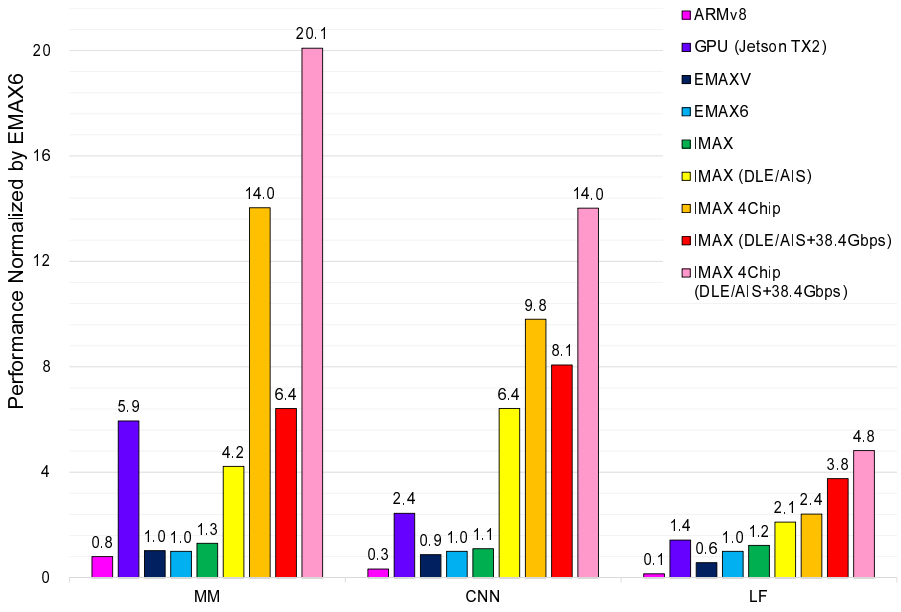



図 24: アプリケーション実行性能

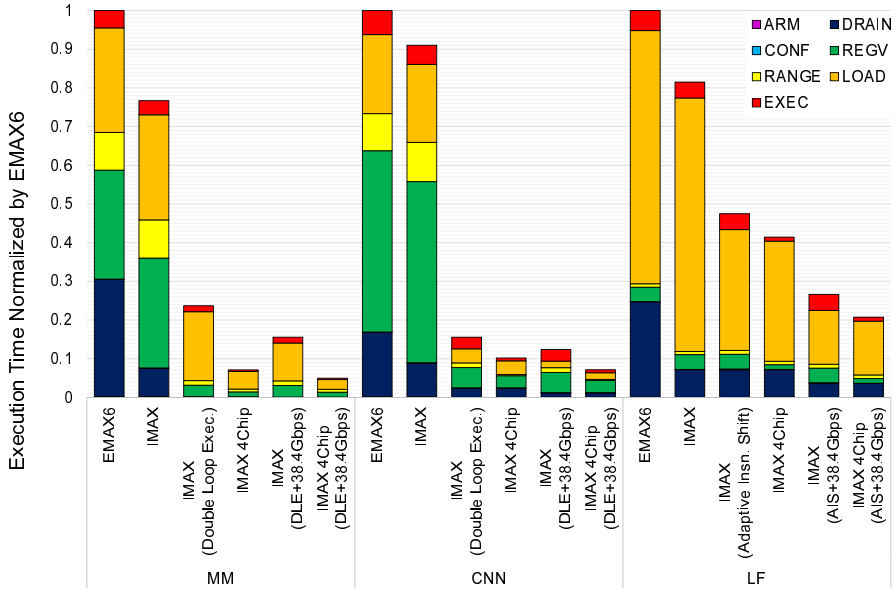



図 25: ステート別プロファイリング結果

グ結果を示す．評価は，比較対象として，ZCU102 の PS に搭載されている ARMv8 Cortex-A53 シングルコア (ARMv8) および，NVIDIA の組み込み向け GPU 搭載 SoC である Jetson TX2 を用意し，EMAXV，EMAX6，IMAX，2 重ループ一括実行 (Double Loop Execution: DLE) またはアダプティブ命令シフト (Adaptive Instruction Shift: AIS) を使用した IMAX，および，DLE/AIS を用いた 4 チップ構成 IMAX の見積もりで行った．2 重ループ一括実行は MM および CNN で，アダプティブ命令シフトは LF で用いた．4 チップ構成のプロトタイプは未完成であるが，コンパイラは完成しており，4 チップ構成時に対応した DMA やユニット設定 PIO が可能であるため，これを用いて見積もりを行った．このとき，実際のマルチチッププロトタイプにおいて発生すると考えられる FPGA-FPGA 間通信のオーバーヘッドは見積もりに含まれないことに注意する．また，前述したとおり，FPGA プロトタイプシステムでは，ボード間通信の 1.8GB/s がボトルネックとなっている．256bit AXI4 インターフェースを持ち 150MHz で動作する IMAX の帯域幅は 256bit×150MHz=38.4Gbps=4.8GB/s であり，主記憶アクセス速度が律速されている．そこで，各アプリケーションにおいて図 20 に基づいて総 DMA 転送量

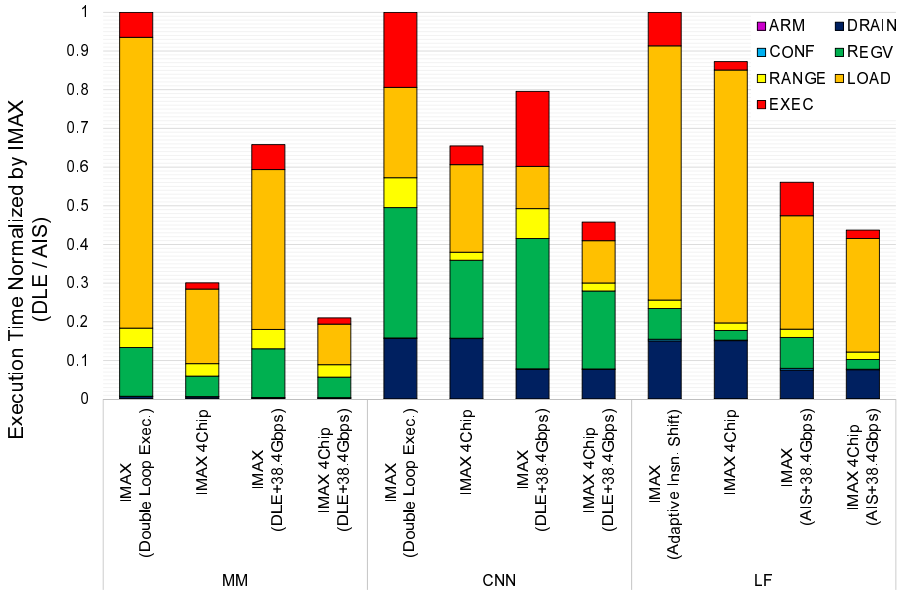



図 26: ステート別プロファイリング結果（詳細）

を計算し，ホスト-IMAX 間通信が 38.4Gbps で行われた場合の DMA を転送時間を見積もり，図 26 の DMA 転送時間と差し替え，理想帯域幅下における IMAX の性能を見積もった．表 7 に理想帯域幅下における DMA 転送時間見積もりの詳細を示す.

表 7: 各アプリケーションにおける DMA 総転送量と理想帯域幅下における所要時間

| 転送データ | バースト長 [Byte] | DMA 起動回数 | DMA 総転送量 [Byte] | 帯域幅 38.4Gbps における所要時間見積もり [ns] |
|---|---|---|---|---|
| MM | | | | |
| A (LOAD) | 15,360 | 60 | 921,600 | 192,000 |
| B (LOAD) 1 チップ構成 | 1,920 | 28,800 | 55,296,000 | 11,520,000 |
| B (LOAD) 4 チップ構成 | 1,920 | 7,200 | 13,824,000 | 2,880,000 |
| Total LOAD 1 チップ構成 | | 28,860 | 56,217,600 | 11,712,000 |
| Total LOAD 4 チップ構成 | | 7,260 | 14,745,600 | 3,072,000 |
| C (DRAIN) | 15,360 | 60 | 921,600 | 192,000 |
| Total DRAIN | | 60 | 921,600 | 192,000 |
| CNN | | | | |
| ker (LOAD) | 10,368 | 1 | 10,368 | 2,160 |
| in (LOAD) | 9,680 | 540 | 5,227,200 | 1,089,000 |
| out (LOAD) | 7,744 | 1,440 | 11,151,360 | 2,323,200 |
| Total LOAD | | 1,981 | 16,388,928 | 3,414,360 |
| out (DRAIN) | 7,744 | 1,440 | 11,151,360 | 2,323,200 |
| Total DRAIN | | 1,440 | 11,151,360 | 2,323,200 |
| LF | | | | |
| sad (LOAD) | 6,144 | 1,536 | 9,437,184 | 1,966,080 |
| out (LOAD) | 6,144 | 1,536 | 9,437,184 | 1,966,080 |
| in (LOAD) | 30,000 | 3,168 | 95,040,000 | 19,800,000 |
| Total LOAD | | 6,240 | 113,914,368 | 23,732,160 |
| out (DRAIN) | 6,144 | 1,536 | 9,437,184 | 1,966,080 |
| sad (DRAIN) | 6,144 | 1,536 | 9,437,184 | 1,966,080 |
| Total DRAIN | | 3,072 | 18,874,368 | 3,932,160 |

図 25 において，ARM と CONF は図 20 では省略した起動までの ARM 側処理時間と命令写像に要する時間であり，いずれも無視できるレベルである．EMAX6 と IMAX を比較すると，チップ内バス並列化により，特に AXI-READ-IF の遅延が短縮され，DRAIN 時間が大幅に削減できていることがわかる．IMAX と IMAX (DLE/AIS) を比較すると，MM および CNN では 2 重ループ一括実行により起動回数を削減できたことで RANGE+REGV を含む起動に伴うオーバーヘッドを，LF に関してはアダプティブ命令シフトにより余分な LOAD を大幅に削減できていることがわかる．3 つのアプリケーションいずれについても，2 重ループ一括実行またはアダプティブ命令シフト以降の貢献が大きい．さらに，38.4Gbps-AXI を仮定すると，３つのアプリケーションでオーバーヘッドの多くを占めている LOAD が削減され，理想的な実行モデル (図 20) に近づく．

図 24 より，FPGA 上に実装した IMAX (DLE/AIS) は，ARMv8 に対して 5.3, 19.5, 14.6 倍，GPU に対して 0.7, 2.6, 1.5 倍，EMAXV に対して 4.1, 7.3, 3.7 倍，EMAX6 に対して 4.2, 6.4, 2.1 倍の性能となることが確認できた．また，4 チップ構成 (IMAX 4Chip) は 1 チップ構成 (IMAX (DLE/AIS) ) に対して 3.3, 1.5, 1.2 倍の性能となる見通しが得られた．特に，マルチチップ構成時の MM に関しては行列 B をブロードキャストすることで LOAD を削減できるため，性能上昇率が大きい．そして，プロトタイプにおけるボトルネックを解消し，IMAX の AXI インターフェースの全帯域を活用してホストと通信できた場合，IMAX (DLE/AIS + 38.4Gbps) は，ARMv8 に対して 8.1, 24.5, 26.1 倍，GPU に対して 1.1, 3.3, 2.6 倍，EMAXV に対して 6.3, 9.2, 6.6 倍，EMAX6 に対して 6.4, 8.1, 3.9 倍の性能となる見通しが得られた．このとき，4 チップ構成 (IMAX 4Chip (DLE/AIS+38.4Gbps)) は 1 チップ構成 (IMAX (DLE/AIS+38.4Gbps) ) に対し，3.1, 1.7, 1.3 倍の性能となる見通しが得られた．

これらのことから，IMAX は組み込み向け CPU や GPU と比較しても優れたアプリケーション実行性能を有することが明らかとなった．また，4 チップ化によって最大で 3 倍程度の性能が得られることから，メモリインタフェースを増設することなく性能のスケールアウトが可能である見通しが得られた．

## 5.3 論理合成による面積見積

IMAX を Verilog HDL により RTL モデル化し，論理合成を Synopsys Design Compiler を用いて回路面積の見積もりを行った．表 8 に論理合成を行った環境を示す．スタンダードセルライブラリと SRAM ライブラリについては TSMC 28nm プロセスのライブラリを利用した．しかし，配線負荷のモデルは含まれておらず，配線遅延と消費電力の見積もりについては配置配線が必要となるため本論文では扱わない．EMAXV, EMAX6 においても同等の環境で論理合成を行い，回路面積を算出した．このとき，周波数制約は，EMAXV で 300MHZ，EMAX6 / IMAX で 900MHz とした．EMAXV は，以前に実施した TSMC28nm ライブラリ，配線負荷モデルおよびメモリジェネレータを用いた評価により，動作周波数 300MHz，回路面積 7.3$mm^2$（8KBLMM），24.9$mm^2$（32KBLMM）であることがわかっている．そのため，今回は配線負荷モデルが使用できなかったため，EMAX の 300MHz に FPGA の動作周波数比 (3.0) を乗じた 900MHz にて提案手法の面積評価を行った．また，64 stages 構成の場合は合成時間が膨大であるため，16 stages と 8 stages の結果を用いて 64 stages の面積を算出した．

表 9 に論理合成結果を示す．まず，従来型 CGRA である EMAXV と提案している CGLA である EMAX6,IMAX について比較を行う．組み合わせ回路の面積を示している Combinational に注目すると，EMAXV に対して，EMAX6 および IMAX はおよそ半分ほどの面積に削減できたことが分かる．4 列を 1 列に統合したにもかかわらず，半分となったのは主記憶と LMM 間のバスをパイプライン化し，アドレスの比較と LMM アクセスの衝突を防止するためのバッファリングを行うために，バックプレッシャーをかける信号の生成をする回路が増えたことが原因として挙げられる．レジスタとして実装される Noncombintional に注目する

表 8: 評価に使用した環境

| | |
|---|---|
| ASIC 向け論理合成環境 | Design Compiler N-2017.9-SP1 |
| テクノロジ | TSMC 28nm |

と，Combinational と同じ理由で，バスをパイプライン化した分少しだけ増加していることが読み取れる．一方，LMM に注目すると，メモリマクロはユニット数の比のまま 1/4 となっており，面積削減効果が大きいことが分かる．LMM の面積が支配的であり，特に CGRA から CGLA となったことによる stgae の一列化の効果が大きい．

EMAX6 と IMAX の比較では，アドレス計算機構とフィードバックパスを追加したにもかかわらず，Combinational，Noncombinational は微減，Buf/Inv は微増という結果となり，トータルでは面積はわずかに減少した．これは，LMM の面積が支配的であることに加えて，EXE のパイプライン化によってタイミング制約の満足が容易となったことによる面積削減と，多重ループ一括実行機構追加分の面積増加が相殺した結果であると考えられる．結果として，IMAX は EMAXV に対して，動作周波数 3 倍の 4way マルチスレッディングであることから，3/4/0.307=2.44 倍の面積あたりピーク性能を多重ループ一括実行機能を追加しながらも実現できることが明らかとなった．

表 9: EMAXV / EMAX6 / IMAX 64 stages の回路面積

| | EMAXV | EMAX6 | IMAX |
|---|---|---|---|
| Clock constraints [MHz] | 300 | 900 | 900 |
| Combinational [$\mu m^2$] | 4,074,459 | 1,647,054 | 1,705,263 |
| Buf/Inv [$\mu m^2$] | 173,855 | 165,545 | 278,722 |
| Noncombinational (registers) [$\mu m^2$] | 773,047 | 1,016,211 | 826,694 |
| Macro/Black Box (LMM) [$\mu m^2$] | 22,374,142 | 5,593,536 | 5,593,536 |
| Total cell area [$\mu m^2$] | 27,395,504 | 8,422,346 | 8,404,215 |
| ratio (normalized by EMAXV) | 1.00 | 0.307 | 0.307 |

# 6. おわりに

本研究では，CGRA の演算ユニットを時分割 4 重実行と浮動小数点演算パイプラインの導入によって性能低下を抑えつつ 1 次元構造化（演算器数は 1/4）した EMAX6 に対して，多重ループ実行機能の追加と整数演算器のパイプライン化を施し，さらに，外部メモリインターフェースを増やすことなくマルチチップ化可能なリニアアレイ (CGLA) アクセラレータ IMAX を提案した．

FPGA 上に IMAX 64 stages のプロトタイプシステムを開発し，動作周波数が 2 次元構造の同等機能 CGRA (EMAXV) の 50MHz に対し，150MHz に向上したことを確認した．そして，プロトタイプを用いて実アプリケーション (行列積，畳み込み演算，Light-field 画像処理) の実行性能を評価した．その結果，IMAX は EMAXV に対して 4.1, 7.3, 3.7 倍の実行性能を有することが明らかとなった．さらに，プロトタイプにおけるボード間通信のボトルネックを解消した場合，EMAXV に対して 6.3, 9.2, 6.6 倍の実行性能となる見積もりが得られた．加えて，4 チップ構成の見積では，1 チップ構成に対して，プロトタイプの場合で最大 3.3 倍，ボトルネックを解消した場合の見積もりで最大 3.1 倍のスケールアウトが可能であるという見積もりが得られた．最後に，28nm テクノロジを利用した論理合成による回路面積の見積もりを行い，EMAXV の面積 27.40mm$^2$ に対し，IMAX は 8.40mm$^2$ で実装できることが明らかとなった．ASIC においても FPGA プロトタイプと同様の周波数比となることを仮定した場合，面積当たりピーク性能は EMAXV に対し 2.44 倍となった．

# 謝辞

僅かな進捗でも欠かさずに褒めてくださり，進捗が芳しくないときは隣に付き添ってサポートしてくださるなど，常に暖かく的確にご指導くださり，また，研究以外でも社会においてよしなに事を運ぶための立ち回り方など，今後の人生において大いに役立つ助言を数多く賜りました中島康彦教授に心より御礼申し上げます．本論文をご精読頂き，また，研究室内のみでは得られない異なった目線からのご助言を賜りました井上美智子教授に深く感謝いたします．日常的に学生に近い目線で接してくださり，また，本論文の執筆においてもご助言を賜りました中田尚准教授に深く感謝いたします．講義において，アナログ回路の基礎からシミュレーションまで丁寧にご教授くださり，また，ご多忙にも関わらず事務手続きや英文添削を快く引き受けてくださったRenyuan ZHANG助教に深く感謝いたします．講義においてご指導を賜りましたTRAN Thi Hong助教に感謝いたします．

至らないことばかりで多大なるご迷惑をお掛けしたにも関わらず，見捨てずに粘り強くご指導・ご助言くださり，また，研究以外でも日頃の息抜きから食事に至るまで手厚くお世話をしてくださり，今もなお，お忙しいにも関わらずお相手してくださる山野龍佑先輩に心より御礼申し上げます．日々暖かく接してくださり，本論文の執筆におきましてもツールの使用方法やRTL記述のLINTチェックなど，多大なるご支援を賜りました一倉孝宏氏に心より御礼申し上げます．テストやウェブなど異分野の知識も豊富で，数多くご助言を賜りました福岡久和先輩に深く感謝いたします．入学前のインターンシップからお世話になり，その後も継続して相談事や休日のお相手をしてくださっている研究室OBの加藤大真氏および嶋谷知氏に深く感謝いたします．修士生活をよき思い出として残る2年間としてくださった同輩の上竹規之氏，平賀由利亜氏，山根弘樹氏，吉田怜矢氏に深く感謝いたします．宴会や旅行を計画してくださり，2年目も楽しい年としてくださった後輩の池田裕哉氏，岩本淳氏，西本宏樹氏，新谷隆太氏に深く感謝いたします．

末筆となりましたが，本学への進学に際しては快諾してくださり，心身から経済面まで支えてくださり，そして，24年間，ここまで育ててくださった両親に篤く御礼申し上げます．

# 参考文献


[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet Classification with Deep Convolutional Neural Networks". In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[3] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". *CoRR*, abs/1409.1556, 2014.

[4] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going Deeper with Convolutions". In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[5] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks". In *Proceedings of the 43rd International Symposium on Computer Architecture*, ISCA '16, pages 367–379, Piscataway, NJ, USA, 2016. IEEE Press.

[6] 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 *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 243–254, June 2016.

[7] W. Lu, G. Yan, J. Li, S. Gong, Y. Han, and X. Li. "FlexFlow: A Flexible Dataflow Accelerator Architecture for Convolutional Neural Networks". In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 553–564, Feb 2017.

[8] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon. "In-Datacenter Performance Analysis of a Tensor Processing Unit". In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12, June 2017.

[9] 中島康彦. "EMAXV における複数バースト転送と複数ベクトル演算のオーバラップ手法 (コンピュータシステム)". In *IEICE technical report :* **信学技報**, volume 116, pages 71–76. 電子情報通信学会, aug 2016.

[10] Y. Yuttakonkit and Y. Nakashima. "Performance Comparison of CGRA and Mobile GPU for Light-Field Image Processing". In *2016 Fourth International Symposium on Computing and Networking (CANDAR)*, pages 174–180, Nov 2016.

[11] 一倉孝宏, 山野龍佑, 福岡久和, and 中島康彦. "DCNN に最適な CGRA の探索と予備評価"(リコンフィギャラブルシステム)". In *IEICE technical report :* **信学技報**, volume 116, pages 49–54. 電子情報通信学会, jan 2017.

[12] "Prepared under contract with NVIDIA Corporation, NVIDIA's Fermi: The First Complete GPU Computing Architecture". http://www.nvidia.com/content/PDF/fermi_white_papers/P.Glaskowsky_NVIDIA%27s_Fermi-The _First_Complete_GPU_Architecture.pdf.

[13] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. "Deep Learning with Limited Numerical Precision". In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1737–1746. JMLR.org, 2015.

[14] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie. "Quantifying the Energy Cost of Data Movement in Scientific Applications". In *2013 IEEE International Symposium on Workload Characterization (IISWC)*, pages 56–65, Sep 2013.

[15] H.T. Kung. "Why Systolic Architectures?". *Computer*, 15(1):37–46, Jan 1982.

[16] Francisco-Javier Veredas, M. Scheppler, W. Moffat, and Bingfeng Mei. "Custom Implementation of the Coarse-grained Reconfigurable ADRES Architecture for Multimedia Purposes". In *International Conference on Field Programmable Logic and Applications, 2005.*, pages 106–111, Aug 2005.

[17] S. C. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. R. Taylor, and R. Laufer. "PipeRench: A Coprocessor for Streaming Multimedia Acceleration". *ACM SIGARCH Computer Architecture News*, 27(2):28–39, May 1999.

[18] N. Ozaki, Y. Yasuda, M. Izawa, Y. Saito, D. Ikebuchi, H. Amano, H. Nakamura, K. Usami, M. Namiki, and M. Kondo. "Cool Mega-Arrays: Ultralow-Power Reconfigurable Accelerator Chips". *IEEE Micro*, 31(6):6–18, Nov 2011.

[19] N. Devisetti, T. Iwakami, K. Yoshimura, T. Nakada, J. Yao, and Y. Nakashima. "LAPP: A Low Power Array Accelerator with Binary Compatibility". In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pages 854–862, May 2011.

[20] Chris Nicol. "A Dataflow Processing Chip for Training Deep Neural Networks", 2017. https://www.hotchips.org/wp-

content/uploads/hc_archives/hc29/HC29.22-Tuesday-Pub/HC29.22.60-NeuralNet1-Pub/HC29.22.610-Dataflow-Deep-Nicol-Wave-07012017.pdf.

[21] T. Ichikura, R. Yamano, Y. Kikutani, R. Zhang, and Y. Nakashima. “EMAXVR: A programmable accelerator employing near ALU utilization to DSA”. In *2018 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*, pages 1–3, April 2018.

[22] 菊谷雄真, 山野龍佑, 一倉孝宏, and 中島康彦. “時分割多重実行型シストリックリングの実装と評価”. In *IEICE technical report :* **信学技報**, volume 117, pages 31–36. 電子情報通信学会, jan 2018.

[23] N. Beck, S. White, M. Paraschou, and S. Naffziger. “‘Zeppelin’: An SoC for Multichip Architecture”. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 40–42, Feb 2018.

[24] Masakazu Tanomoto, Shinya Takamaeda-Yamazaki, Jun Yao, and Yasuhiko Nakashima. “A CGRA-Based Approach for Accelerating Convolutional Neural Networks”. In *Proceedings of the 2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip*, MCSOC ’15, pages 73–80, Washington, DC, USA, 2015. IEEE Computer Society.

# 発表リスト

1. 菊谷雄真, 山野龍佑, 一倉孝宏, 中島康彦. “時分割多重実行型シストリックリングの実装と評価”, 信学技報, vol.117, no.377, CPSY2017-111, pages 31-36, Jan 2018

2. 菊谷雄真, 山野龍佑, 一倉孝宏, 中島康彦. “エッジコンピューティング向けアクセラレータの実装と評価”, 電子情報通信学会関西支部第 23 回研究発表講演会, Mar 2018

3. Takahiro Ichikura, Ryusuke Yamano, Yuma Kikutani, Renyuan Zhang, and Yasuhiko Nakashima. “EMAXVR: A Programmable Accelerator Employing Near ALU Utilization to DSA”, IEEE Symposium on Low-Power and High-Speed Chips 2018, Apr 2018

4. 菊谷雄真, 中島康彦. “ニアメモリ型アクセラレータ IMAX のプロトタイプ実装と分析”, 信学技報, vol.118, no.92, CPSY2018-2, pages 35-40, Jun 2018

5. 岩本淳, 菊谷雄真, 中島康彦. “ユニット内フィードバックによるリニアアレイの多重ループ対応手法”, 信学技報, vol.118, no.339, CPSY2018-40, pages 33-38, Dec 2018

# 受賞

1. 菊谷雄真, 山野龍佑, 一倉孝宏, 中島康彦. “エッジコンピューティング向けアクセラレータの実装と評価”, 電子情報通信学会関西支部第 23 回研究発表講演会, Mar 2018, 電子情報通信学会関西支部学生会研究発表講演会奨励賞