NAIST-IS-MT1651118

修士論文

マルチスレッド化による
シストリックリングアクセラレータの面積削減手法

山野 龍佑

2018年2月1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。

山野 龍佑

審査委員：

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

マルチスレッド化による シストリックリングアクセラレータの面積削減手法*

山野 龍佑

内容梗概

機械学習アルゴリズムの成功や，VR 技術のコモディティ化が大きな注目を集めている．それに伴い組み込み機器に要求される演算性能が高まっている．一方で，半導体微細化による周波数向上やコストダウンの恩恵がなくなりつつある．これまで汎用プロセッサが享受してきた周波数向上とマルチコア化による性能向上は今後望めない．DCNN の社会実装に向けて GPU を搭載した大規模サーバで学習を行い，組み込み機器では DSA を搭載し認識のみを行う垂直分業が確立しつつある．しかし，DSA はその特性上応用範囲が限定的であり，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出来ない問題点がある．

CGRA やサブセットであるシストリックアレイはプログラマビリティを持つアクセラレータとして研究されてきた．しかし構造上の欠点として，配線混雑と演算器およびメモリを使い切ることの困難さが指摘され，面積効率向上が課題であった．本研究では，シストリックリングアクセラレータである EMAXV をもとに，マルチスレッド化によるシストリックリングアクセラレータの面積効率向上手法を提案する．行列積，3×3 畳み込み演算，Lightfield 画像処理の 3 つのアプリケーションについて EMAXV と演算性能の比較を行い，マルチスレッド化した IMAX においても同等の演算性能を達成可能であることを確認した．更に 28nm テクノロジを利用した論理合成による回路面積の評価を行った．この結果 IMAX は EMAXV と比べて面積当たり性能を 2.25 倍から 4.34 倍改善できることを示した．

*奈良先端科学技術大学院大学 情報科学研究科 修士論文, NAIST-IS-MT1651118, 2018 年 2 月 1 日.

キーワード

CGRA, 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

Area Reduction Method of systolic ring Accelerator by multithreading*

Ryusuke Yamano

Abstract

The success of machine learning algorithms and the commoditization of VR technology have attracted a lot of attention. Along with that, the computing performance required for embedded devices is increasing. On the other hand, the benefits of frequency improvement and cost reduction due to miniaturization of semiconductors are getting rid of. Performance improvement by frequency improvement and multicore which general-purpose processor has got so far, cannot be expected in the future. For the social implementation of DCNN, vertical division of labor is being established that learns with a large scale server equipped with GPU and just recognition in embedded system be incorporated DSA. However, due to its characteristics, the DSA has limited scope of application, and there are problems that it cannot respond to changes in the machine learning algorithm and the appearance of killer applications other than machine learning.

CGRAs and systolic arrays that are subsets have been studied as programmable accelerators. However, as a structural disadvantage, it is pointed out that there is congested wiring and it is difficult to use up the computing unit and memory, and improvement of the area efficiency has been a problem. In this research, the author propose a method of improving the area efficiency of systolic ring accelerator by multithreading based on systolic ring accelerator EMAXV. The author compared

*Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1651118, February 1, 2018.

the computational performance with EMAXV for the three applications of matrix multiplication, convolution operation, and lightfield image processing, and confirmed that the same computing performance can be achieved with multithreaded IMAX. The circuit area by logic synthesis using 28 nm technology was evaluated. As a result, it was shown that IMAX can improve performance per area compared with EMAXV from 2.25 times to 4.34 times.

Keywords:

CGRA, 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，組み込み機器向けの計算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．大きな要因の一つはDNN（Deep Neural Network）による機械学習の進歩が挙げられる [1, 2, 3]．特にDCNN（Deep Convolutional Neural Network）の応用範囲が広く社会実装の期待が高まっているが，要求される計算資源が莫大なため低消費電力かつ高い演算性能を持つ組み込み機器が希求されている．もう一つの大きな要因は，VR（Virtual Reality）技術のビジネス化がある．特に2016年はVR元年と呼ばれ，これまでVRを体験を可能にするHMD （Head Mounted Display）は，一部の研究機関や特殊作業に従事する人しか入手が困難なデバイスであったが，複数のサプライヤーが消費者向けのHMDを市場に投入したため，これからますますコモディティ化が進むことが予想される．これに伴い，3D動画像データがコンテンツとして価値を高めていると同時に，3D動画像処理を行うためデバイスに高い演算性能が要求されている．

しかし一方で，半導体微細化による低電力化やコストダウンの恩恵がなくなりつつある．図1はプロセスルールごとにトランジスタの単価を示している．文献 [4, 5] では半導体，特に28nm以降のテクノロジで微細化は進んでいるものの，周波数は上がらずトランジスタの単価が増大していることが示されている．周波数向上による従来のノイマン型コンピュータの演算性能向上は，物理的な制約により限界を迎えつつある．近年汎用プロセッサはマルチコア化による性能向上が行われていたが，上記のとおり面積当たりの半導体製造コストの増大により，価格的な制約の厳しい組み込み機器に向けた実装はますます困難になると考えられる．つまりこのままでは，組み込み向けに搭載される計算資源の性能が今後大きく向上する保証はないことを意味している．

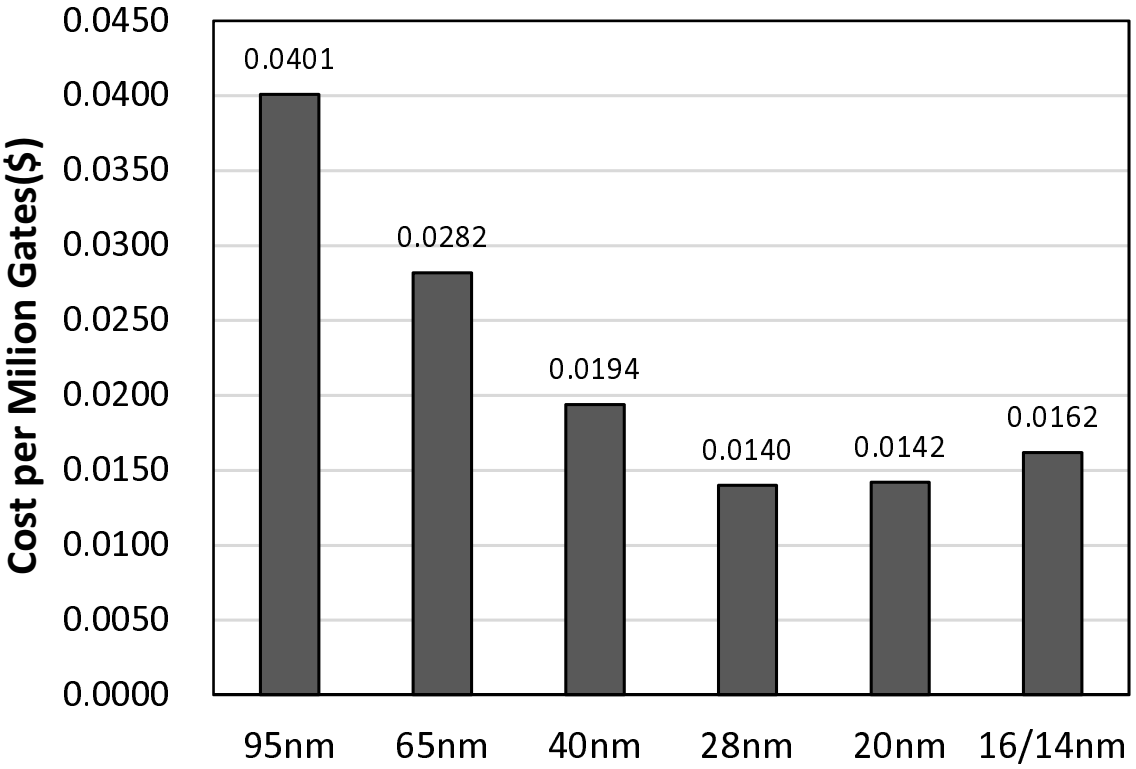


図 1 半導体製造コストの増加

1.2 研究目的

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 DCNN 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 [6]．DCNN だけでなく，VR に必要なステレオ画像生成にも応用可能な Lightfield 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，EMAXV が提案されている [7]．EMAXV はシストリックアレイの接続関係を一方向だけトーラスとして接続したシストリックリングアクセラレータである．

しかし，CGRA およびシストリックアレイは演算ユニット間を接続するための配線が多く，動作周波数を挙げることが困難である．Lightfield 画像処理のようなデータの参照パターンが複雑なアプリケーションに対応するためには多ポートメモリが必要である．EMAXV においては各演算ユニットのローカルメモリに同一内

容を格納して読み出しポート数を増やしているが，メモリマクロは回路面積の大部分を占め，面積効率を悪化させる問題点がある．このことから本研究は EMAXV より面積が 4 分の 1 で演算性能が同等となるシストリックリングアクセラレータの開発を目的とする．

1.3 構成

本論文では，本研究の背景と目的を述べた．次に 2 章にて GPU と DSA による社会実装に向けた DCNN 実装の問題点と，EMAXV の課題を述べる．そして 3 章にて面積効率を改善したシストリックリングアクセラレータとして IMAX（In-Memory Accelerator eXtension）を提案し，面積削減と配線混雑を解消するためのアイデアと諸手法について説明する．4 章では，DCNN の実装に必要とされる行列積と畳み込み演算，および Lightfield 画像処理をシストリックリングアクセラレータにプログラミングする際の設計戦略を説明する．5 章にて EMAXV と IMAX の演算性能と面積および配線混雑そして面積当たり性能の評価を行い．6 章を本論文のまとめとする．

2. 先行研究と課題

背景で DCNN 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 DCNN 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（Domain Specific Accelerator）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 と DSA を説明したのち，現行のシステムにおける DSA の問題点を説明する．そして DSA に変わる計算デバイスとして CGRA をとりあげ CGRA の一種であるシストリックリングアクセラレータ型のアクセラレータである EMAXV と課題を説明する．

2.1 GPU

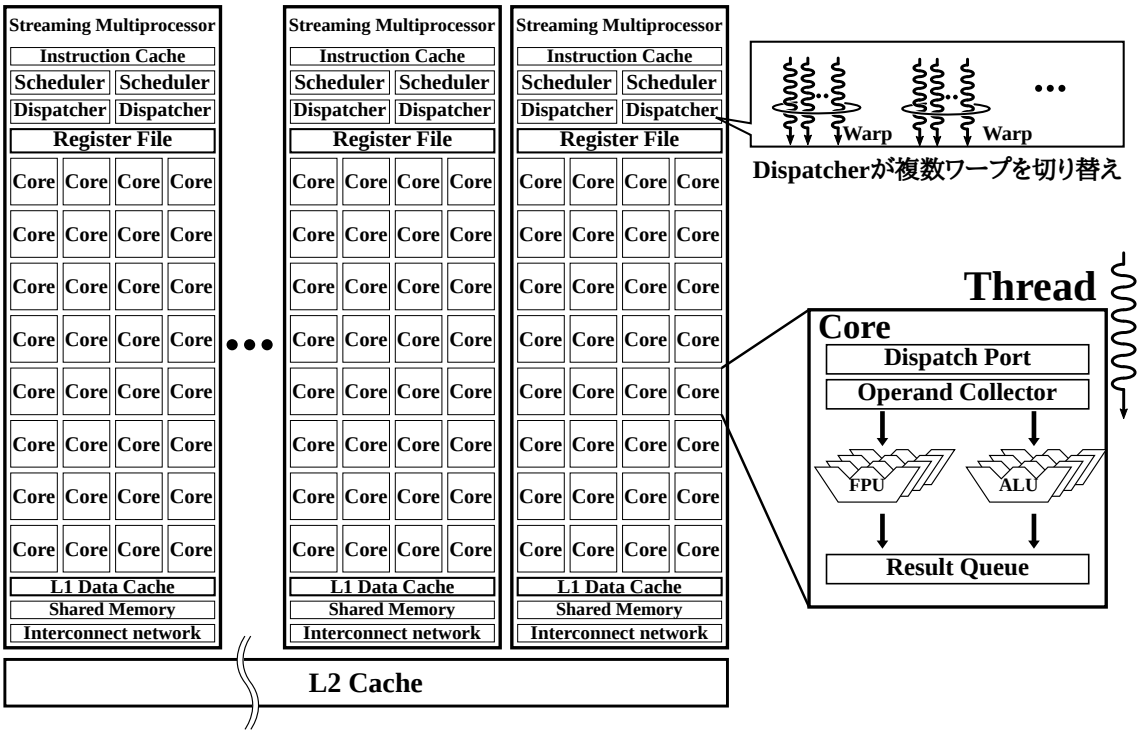


図 2 GPU の構造

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 OoO（Out-of-Order）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 ROB（Re-order buffer）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 DCNN 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，DCNN 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 2 に一般的な GPU の構造を示す．それぞれの演算ユニットは SIMD （Single Instruction Multiple Data）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．NVIDIA はこのグループを SM（Streaming Multiprocessor）と呼び [8]，複数演算ユニットに同一の命令を発行する手法を SIMT（Single Instruction Multiple）方式と呼んでいる．また簡単な分岐は条件付き実行命令でサポートする．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．これらの機構はデータの流れをハードウェアが制御する．さらに Warp はそれぞれのプログラムカウンタを持つため，マルチスレッドプログラミングが一般的になってきた現在，プログラミングを容易にする．しかしながら，組み込み機器への実装には面積，電力面でオーバヘッドとなるためトレードオフの関係といえる．そこで近年は DCNN 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．

2.2 DSA

図 3 は DSA の一つである Eyeriss[9] の構成図である．このほかにも FlexFlow[10] や EIE（Efficient Inference Engine）[11] といった数多くの推論専用 DSA が提案されている．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 [12]，重みの情報量を削減し 16bit-8bit の固定小数積和演算を行う．また，主記憶へのアクセス

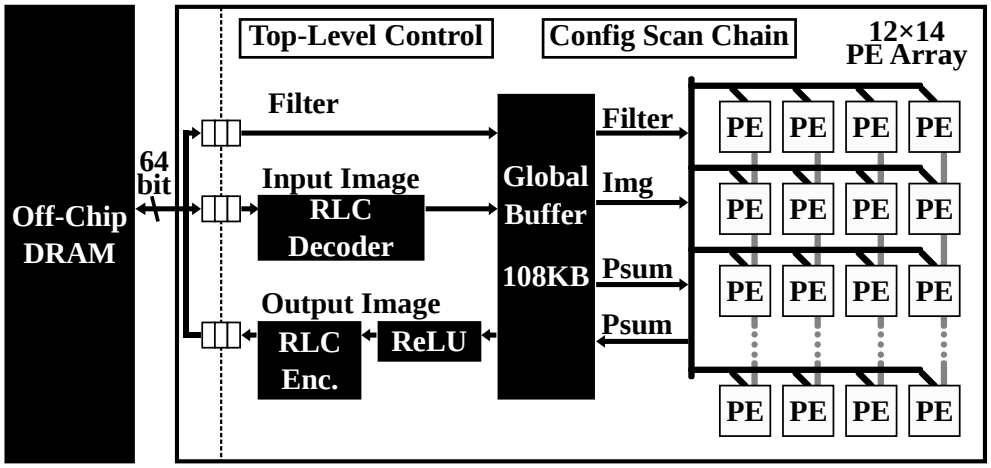


図 3 Eyeriss

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 [13]．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．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，背景で述べたよう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開発コスト面で現実的ではない．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残したアクセラレータが必要である．

2.3 シストリックリングアクセラレータ

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（Coarse-Grained Reconfigurable Architecture）が注目されている．図 4 に一般的な CGRA の構造を示す．特に基本演算ブロックを隣接するブロックに制限した構成の CGRA

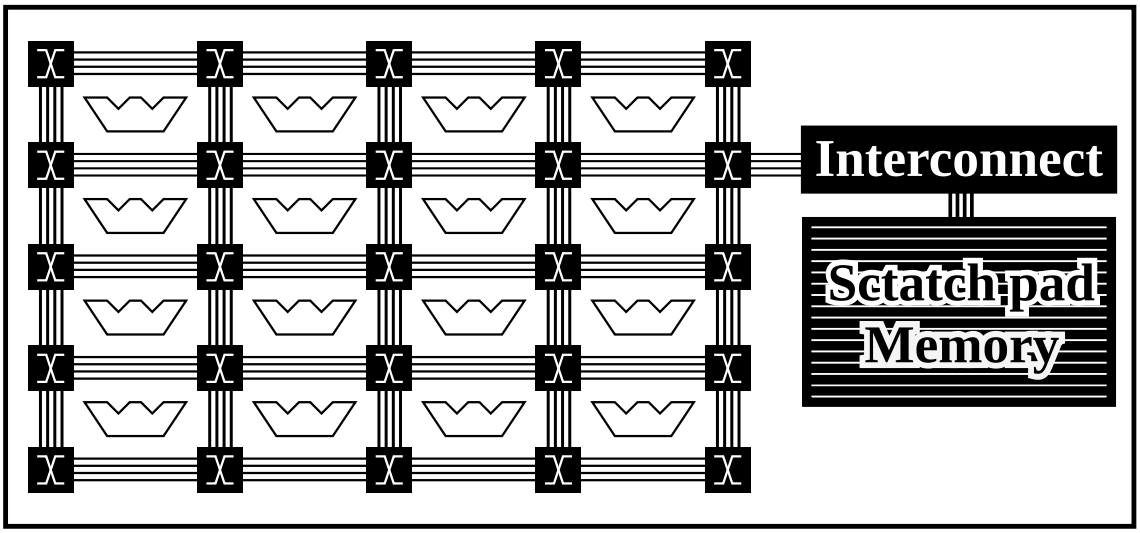


図 4 一般的な CGRA

をシストリックアレイとよび，Google の TPU[6] は高い性能を示している．しかし TPU は 8bit の積和演算のみに対応しており，DSA と同様の問題を抱えている．

画像フィルターをはじめ科学術計算でも頻出し，近傍アドレスを決まったアクセスパターンでロードする演算をステンシル演算とよぶ．近年の DCNN でよく用いられる畳み込み演算もこれにあたる．ベクトル演算と大規模離散ステンシルを高速化可能な，EMAX（Energy-aware Multimode Accelerator eXtension）[14, 15] が提案されている．EMAX 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 LMM（Local Memory）とアドレス計算用の演算回路を持つ．図 5 に EMAX の最新バージョンである EMAXV[16] の概要図を示す．EMAXV は，AMBA AXI4 バスインターフェースを持ち，(A）FSM ユニット内部の DMA エンジンが，主記憶対 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（B）境界レジスタの共有バスと（C）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 は 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 をステージ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．ユニット内部の動作はアドレス計算用 ALU

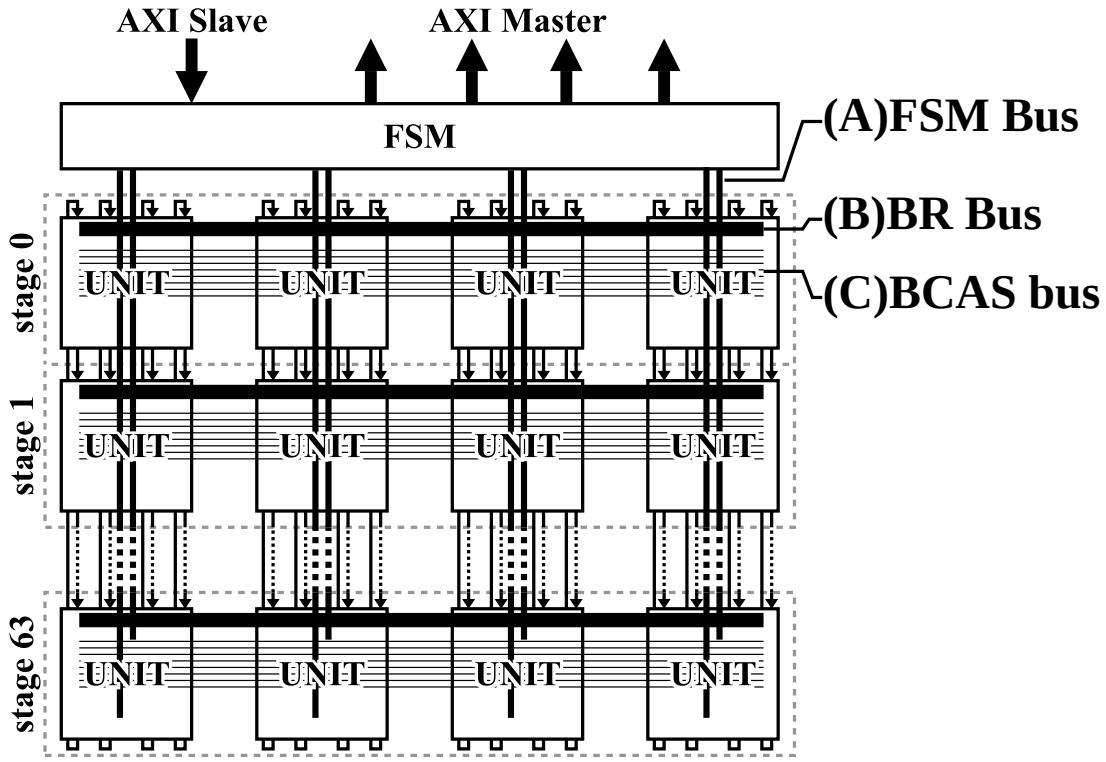


図 5 EMAXV の概略図

を配置したことにより，演算を行いながら主記憶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．

参考文献 [7] で複雑なメモリアクセスパターンを持つ離散ステンシル演算に対しても EMAX が有効であることが報告されている．参考文献 [17] では，EMAXV へ DCNN（Deep Convolutional Neural Network）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．

しかしこれまで EMAX を含むシストリックアレイから発展した 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，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 への実装の困難が予想されると指摘されてきた．EMAXV でも同様に図 5 内の（A）（B）（C）で示した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す．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，回路面積当たりの費用が増大している昨今，データのコピーを置くため物理的に 4 つの LMM を用意することはコストが大きい．

3. 提案手法

LSI の動的消費電力は次式で近似される．α は信号変化率，f は動作周波数，C_L は負荷容量，Vdd は電源電圧を表している．このうち負荷容量は回路面積に比例する．IMAX は EMAXV から動作周波数 4 倍で動作し，面積を 4 分の 1 にすることにより動的消費電力はそのままであると考える．

$$P[W] = \alpha \times f[Hz] \times C_L[F] \times Vdd^2[V] \tag{1}$$

IMAX では面積効率を改善する方法として FPU（浮動小数点演算器）と LMM を含む基本演算ブロックを複数のスレッドで共有する機構を実装する．小節 3.1 にてマルチスレッド化による FPU の回路面積削減手法を説明し，小節 3.2 にて時分割アクセスによる多ポートメモリマクロの面積削減手法を説明する．また，配線混雑を改善する方法として，主記憶と LMM 間のパスをパイプライン化する手法を小節 3.3 にて説明する．さらに IMAX では LMM と命令のコンフィギュレーションを行うレジスタを含めたほぼすべての記憶領域をホストプロセッサから直接ア

クセス可能なアドレス空間にマッピングすることにより，EMAXV に実装されていた 4 基の DMA エンジンを削減する．これについては小節 3.4 にて説明する．

3.1 マルチスレッド化による FPU の削減

図 6 に先行研究である EMAXV のアーキテクチャを示す．EMAXV において，演算ブロック間を接続する MUX から FPU の演算回路を通ってメモリまでのパスがクリティカルパスである．予備評価として FPGA へ実装した際の結果より，主記憶から LMM へデータを転送するバスも配線混雑のため，大きな配線遅延を持つことが明らかとなっている．

ここで改めて，面積当たりの演算効率について考える．同じ面積回路で同等の演算を行うのであれば，より高い周波数で動作する回路の方が面積当たりの演算効率が高い．さて，IMAX では上記の点を踏まえ，回路をパイプライン化することにより面積当たりの演算効率を改善する．図 7 に IMAX のブロック図を示す．黒い箱がレジスタを表している．EMAXV では演算ブロック内で 2 サイクルかけて演算結果を下の演算ブロックに伝達していた．IMAX では演算自体は 4 サイクルかけて行う．まず，MUX から FPU までを 1 ステージ化する．さらに，FPU の積和演算を分解し，3 つの段階に分けて演算する．積については，高速な実装として知られる Booth Encoder と Wallace Tree を用いることにより，$n$bit 積の論理段数が $log2(n)$ のオーダに削減される． 部分積の段階で 1 ステージ化し，下段で CSA（Carry Save Adder）と CLA （Carry Look-ahead Adder）を組み合わせた 3 入力加算を行うことにより，CLA の数を削減している．さらに，昨今の論理合成コンパイラは Retiming という最適化を行うことが可能であり，最終的なモジュールの出力は変わらないまま，クリティカルパス周辺の Flip-Flop をゲートを跨いで複製，削減および移動が可能である．この最適化は演算器等の局所的なモジュールに対して特に有効であることが知られており，今回のレジスタで 1 ステージ化した FPU に対しても有効であると考えている．

ところが，アキュムレート型の自己ループを持つ演算をマップすると，FPU の演算結果が 4 サイクルに一度しか出力されないため，演算効率が 4 分の 1 になる場合があることに気づく．そこで，IMAX では 4 サイクルの演算のスケージュール

に EMAXV の 4 ユニットそれぞれで行っていた演算をスレッドとして割り当てる．これにより，真の依存を持つ演算を 4 サイクルごとに行ったとしても FPU にバブルが発生しない．OoO 実行の CPU であれば ROB によってバブルの発生を抑制するが，ROB は回路面積が大きい．IMAX はコンパイルの段階でバブルが発生しない最適化を行うことが可能である．しかしまだ，下段の演算ブロックで演算結果を利用するためには 4 サイクルごとに 4 サイクルのバブルが発生する．これを解決するために，IMAX は演算ブロック間のレジスタをダブルバッファとして実装する．最終的に演算ブロックは 8 サイクルかけて演算結果を出力する．

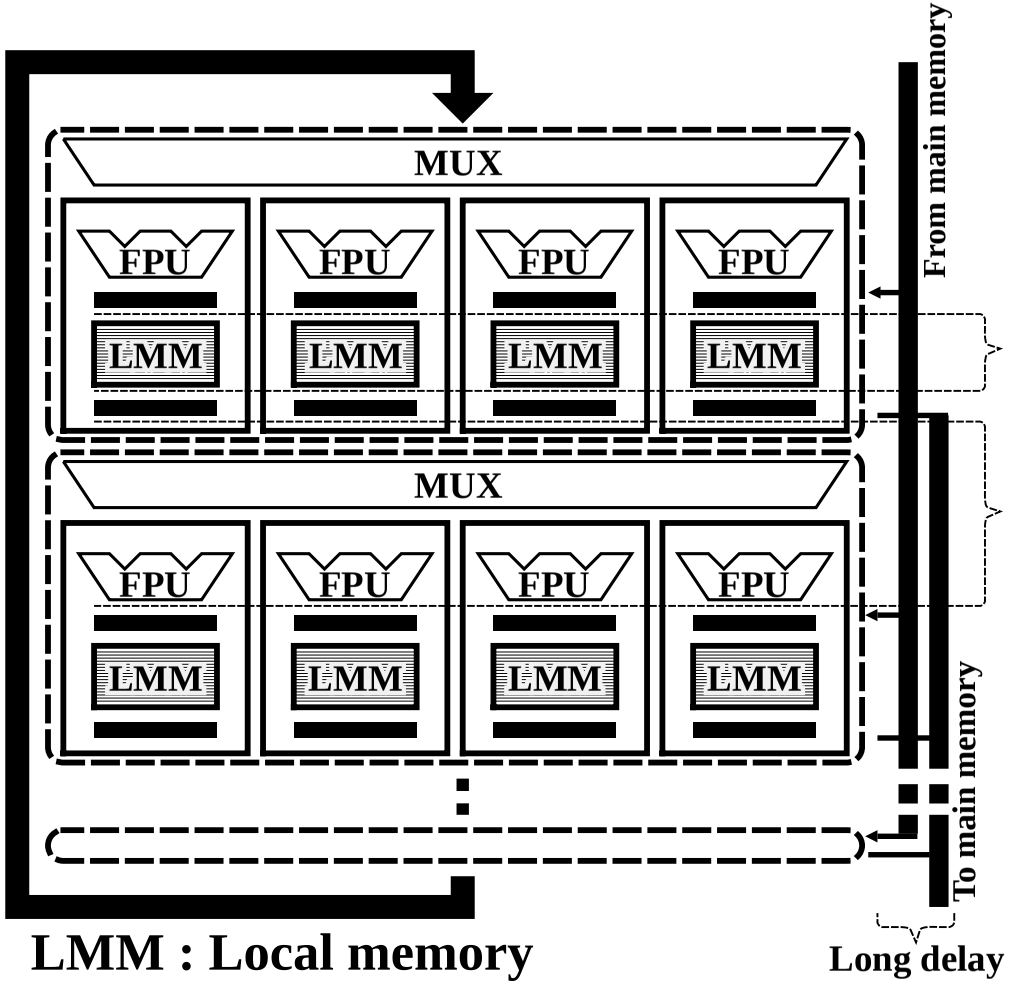


図 6 EMAXV の構成図

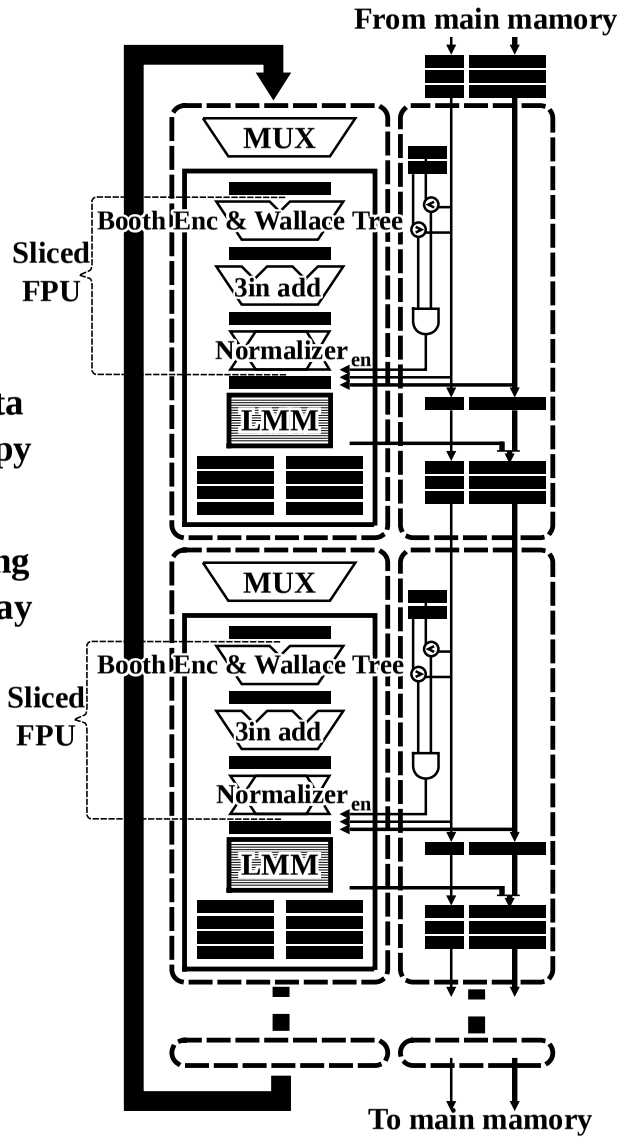


図 7 IMAX の構成図

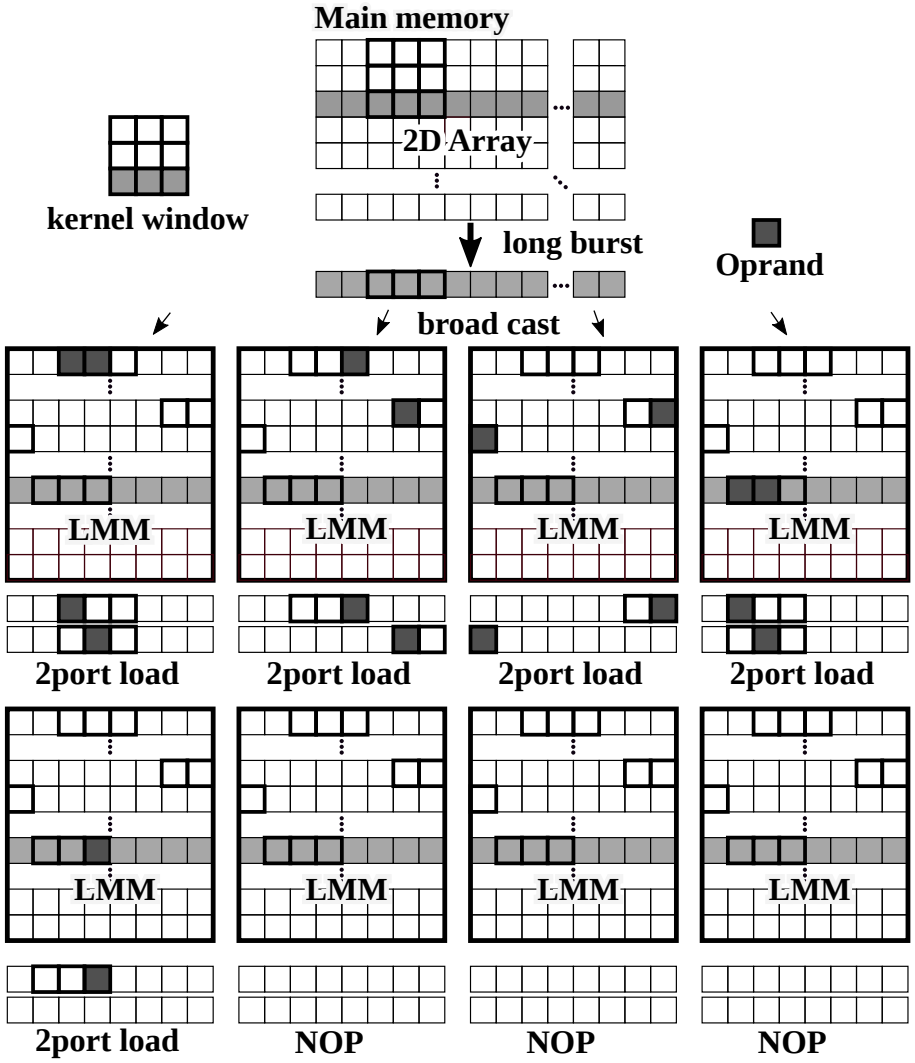


図 8 EMAXV における LMM 参照

3.2 時分割アクセスによる多ポートメモリの面積削減

EMAXV は LMM を備えた演算ブロックを横に 4 つ並べたグループを最小構成とする．横に 4 つ並んだ演算ブロックを縦に増やす方向にスケーラビリティを持つ．図 8 は単純なステンシル演算である 3×3 の畳み込み演算を行う際の EMAXV のデータアクセスを示している．演算を行うためには 9 点のデータをロードする必要がある．広帯域なバスを持つ LMM であれば，一度のロードでアドレスの連続したデータを読み出し可能であるが，前述のように離散アドレスに対するデー

タロードを行うためにはメモリのポートが複数必要になる．EMAXV では横に並べた 4 つのメモリにデータをブロードキャストし，それぞれ 2 ポートから異なるアドレスのデータを読み出すことにより, 実質 8 ポートのメモリとして利用する．更に縦方向にもデータをコピーし 9 点のデータをロードする．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，物理的に 4 つの LMM を用意することはコストが大きい．

そこで，IMAX では演算をマルチスレッド化することにより，1 つの演算ブロックで 4 つの演算ブロックで行っていた演算を行い，LMM を備えた演算ブロックを 4 分の 1 に削減する．EMAXV をでは 4 つの 2 ポートメモリにブロードキャストし，それぞれからロードしていたところを IMAX では，4 サイクルかけて 1 つの 2 ポートメモリから 8 点のデータをロードする．これにより，複数の LMM にブロードキャストされていたデータは同一のメモリから読み出されるようになり，同一行の演算ブロックの LMM へデータをブロードキャストする必要はなくなる．

3.3 主記憶間パスのパイプライン化による配線混雑改善

演算ブロックに対するパイプライン化について説明した．次に主記憶から LMM へのデータパスのパイプライン化について説明する．EMAXV では主記憶からの Load および Store は専用の DMA エンジンとコントローラによって行われていた．コントローラは EMAXV のステートを管理し，逐次 Load と Store 対象の演算ブロックを特定し，膨大な MUX ツリーを制御することにより，DMA エンジンのアドレスバスとデータバスを対象の演算ブロックへと接続する．FPU をパイプライン化し，周波数向上させたのちに MUX ツリーが次のクリティカルパスとなる．そして MUX ツリーは EMAXV において配線混雑の原因となっていた．

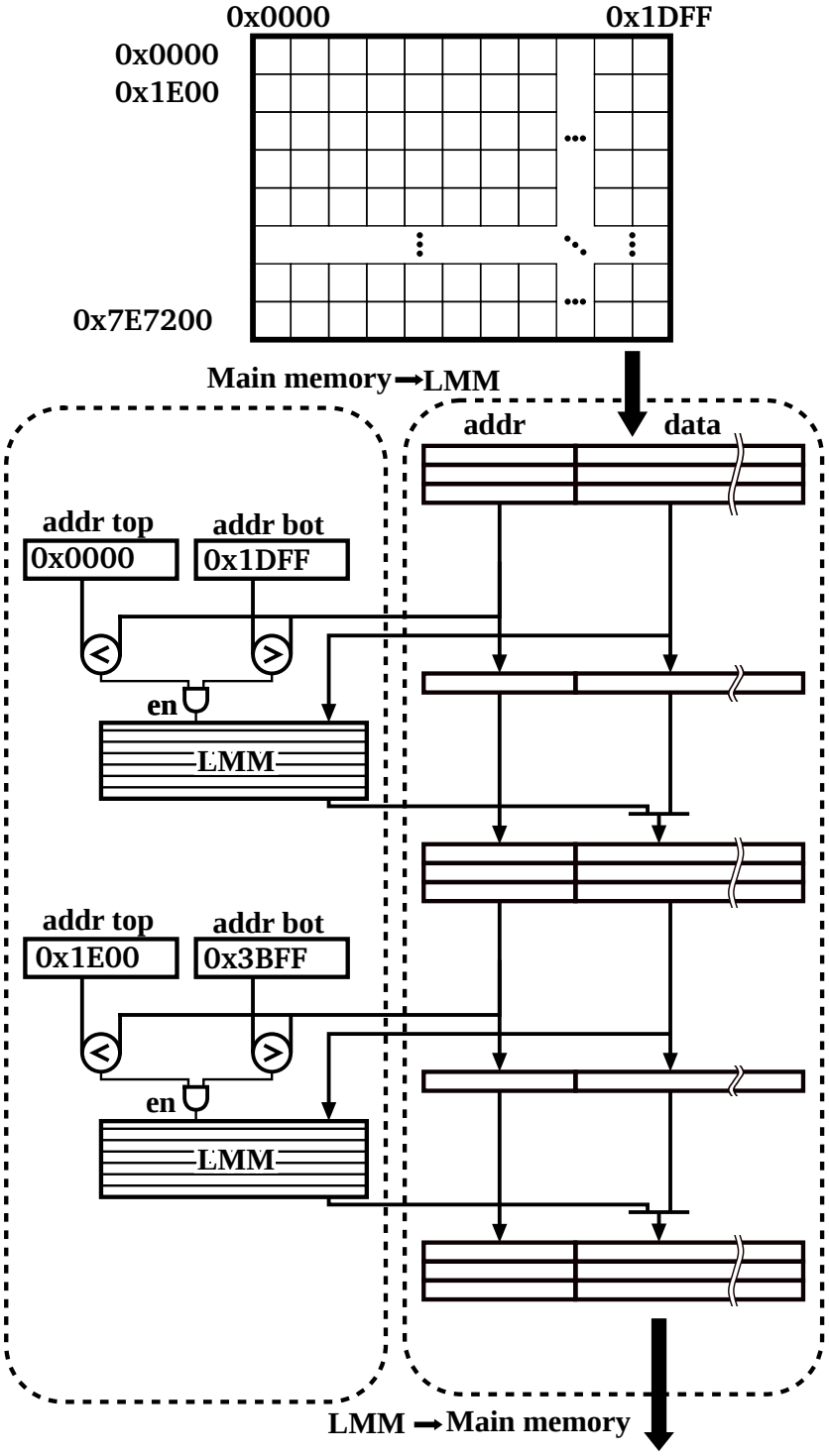


図 9 IMAX における LMM と主記憶間のデータ転送

IMAX では主記憶から LMM へのパスをパイプライン化する．ここで問題となるのが IMAX が演算している最中のデータ転送である．具体的には演算に使用するデータを 2 ポートでロードしているときに，データの書き込みができない．これは読み出すアドレスと書き込むアドレスが異なるため起きる．しかし 4 サイクルのうちデータのロードを行わない命令がマップされている場合，書き込みが可能である．そこで，バックプレッシャーをかけることが可能なバッファを置くことにより，LMM への Load と Store の衝突を回避する．またメモリへのアクセスは 1 サイクル消費するため，タイミングを合わせるために 1 段レジスタを設ける．つまり，演算ブロックは 8 サイクルでデータを伝搬し，1 演算ブロックに対応する主記憶から LMM へのパスは 2 サイクルかけてデータを伝播する．これにより，演算ブロックが増大してもデータ転送のレイテンシ増加は 2 サイクルに収まる．

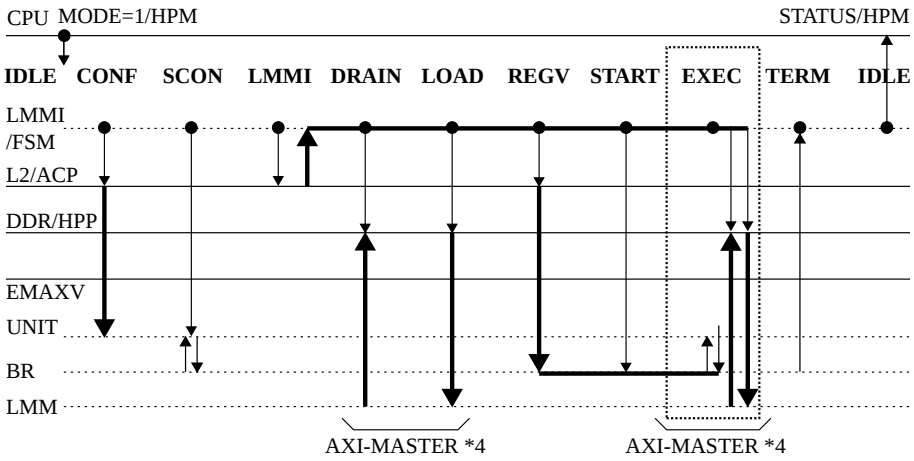
IMAX の段数を増加させると主記憶と LMM 間のレイテンシが大きくなる．そこで，主記憶とローカルメモリ間のバースト転送をさらに効率化する．EMAXV では，縦方向の演算器間でのブロードキャスト機能は持ち合わせておらず，同じデータを複数回転送する必要があった．そこで，IMAX は LMM の対応するアドレスを柔軟に変更可能な機構を持つことにより，縦方向の演算器間に対しても一度の転送でデータコピーを可能にする．図 9 に IMAX での主記憶と LMM 間データ転送の例を示す．まず，各演算ユニットには対応するアドレスの領域を指示するため addr top と addr bot の二つのレジスタを設定する．基本的に LMM はほとんどのアプリケーションでラインバッファとして動作するため，演算データの一行分に注目した際の下位アドレスと上位アドレスを設定することになる．そして，データがバースト転送されると，これらのアドレスと比較して取り込むべきアドレスであれば enable 信号を生成する．このとき複数演算ブロックの addr top と addr bot に同じアドレスを割り当てると自動的にデータがブロードキャストされる．各演算ユニットが独立してアドレス比較した上でデータを取り込むため，バースト転送の際にデータの 2 次元配列であってもラインの端を気にすることなく最大バースト長を利用可能になった．さらに EMAXV は専用の DMA エンジンとコントローラを備えていたが，IMAX ではすべての記憶領域を CPU と同じアドレス空間にマップすることにより， システムに搭載されている汎用の DMA エンジンを利用し，制御レジスタも CPU に設定させる．これにより，EMAXV では演算を

行う度に命令マップデータ及び各レジスタの初期値を主記憶から取り出していたが，IMAX は必要な初期値のみをピンポイントで書き込むことが出来き，オーバーヘッドの削減につながる．

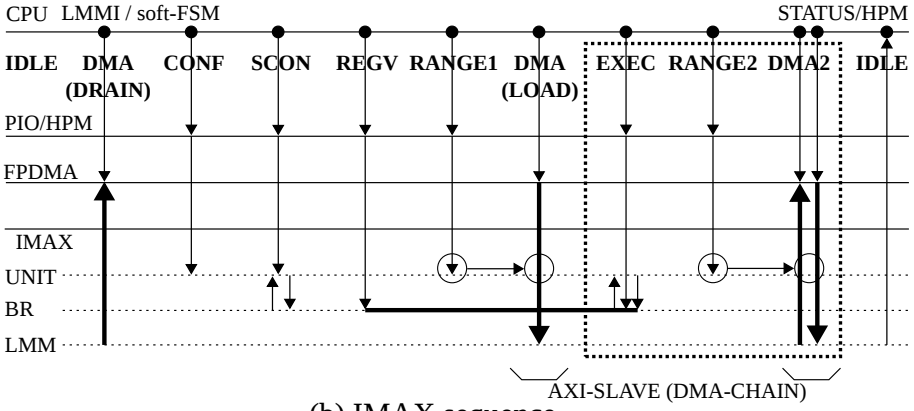
3.4 メモリマップデバイス化による DMA エンジンの削減

図 10 に EMAXV と IMAX の初期化から演算まで一連のステータスの遷移を示す．EMAXV は入出力チャネルを合計 5 本持っていた．そのうち 1 本がホストプロセッサからの制御を受け付けるチャネルであり，それ以外は主記憶へのアクセスを自律的に行う広帯域インターフェースである．そのため，ホストプロセッサから直接アクセス可能な領域は制御用のレジスタ空間のみで，ローカルメモリ内のデータを取り出すためには一度 LMM から主記憶へのデータ転送を行う DRAIN シーケンスを跨ぐ必要があった．さらに主記憶へのアクセスを自律的に行うために，4 基の DMA エンジンを内蔵する必要があり，これも面積効率を悪化させていた．IMAX では入出力チャネルは 1 本しか持たず，データのトランザクションはすべてホスト CPU で制御する．さらに DMA については最近の SoC（System On a Chip）の実装を見る限り，システムのインターコネクト上に汎用の DMA エンジンが実装されていることがほとんどであり，バースト転送が必要な大きなペイロードの転送についてはこれを利用する．つまり，ホストプロセッサは IMAX のほとんどのアドレス空間にアクセス可能となり，ペイロードが小さいデータ転送についてはプロセッサのロードストア命令でカバー可能なため，EMAXV よりもデータ転送のオーバヘッド削減が可能となる．

図 11 に EMAXV の動作モデルと比較した IMAX のメモリとしての動作モデルを示す.（イ）は EMAXV でのロードストアが演算サイクルに完全にオーバーラップした場合の動作シーケンスであり，高い演算性能を発揮することが可能であった．しかし,（ロ）に示すように EMAXV では演算までにデータが間に合わずデータのロードストアペナルティーが発生する場合が存在した．主な原因として，各 LMM で同じ内容を保持する場合でも stage をまたいだデータのブロードキャストが不可能であったため，同じ内容のデータ転送が複数回発生していたことが挙げられる．IMAX では LMM が 4 ユニット分統合した形となるため，LMM 上の参照



(a) EMAXV sequence



(b) IMAX sequence

図 10 EMAXV と IMAX のシーケンス比較

可能な範囲が拡張される．更に，複数の LMM に同じアドレスを割り当てることが可能な機構により，データ転送の回数を大きく 削減可能であるため,（ハ）に示すように EMAXV よりも高速にデータをバースト転送することが可能である．更に EMAXV と比較して有効な点として，内部レジスタの初期化を短縮すること挙げられる．EMAXV では FSM がバースト転送で内部のレジスタを設定を行っていたため，本来であれば更新不要なレジスタのデータも転送されていた．IMAX では PIO のロードストア命令によって個別にレジスタの更新が可能となるため，変更が必要なレジスタのみピンポイントで更新する．

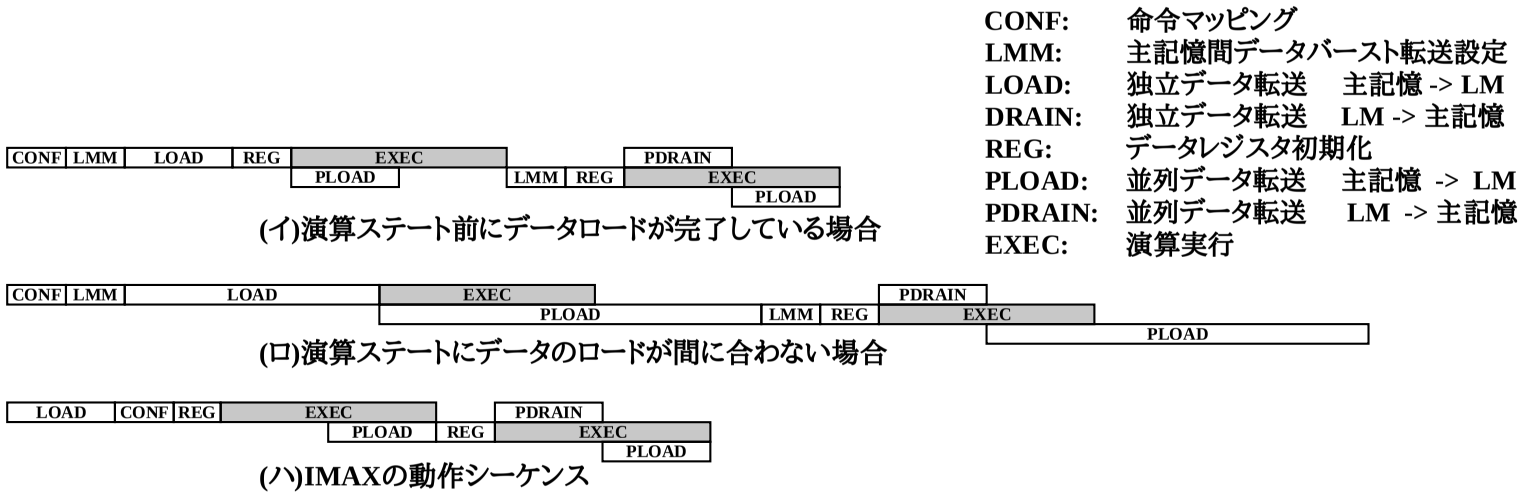


図 11　メモリ動作モデル

4. 高速化対象とするプログラム

シストリックリングアクセラレータを用いたプログラムの高速化ではGPU等のSIMD方式に見られる水平方向の並列性を最大限利用するだけでなく，データが上から下へ流れていくパイプラインとして垂直方向の並列性も積極的に適用していく．更に演算に使用するデータを主記憶から転送する回数を最小限にするために，多重ループを出来るだけアンローリングし，一度のアクセラレータの起動で参照する入力データを大きくする．さらにデータの転送は可能な限り演算とオーバラップさせる．以上がシストリックリングアクセラレータにおけるアプリケーション高速化の基本戦略となる．

以後の小節において，行列積，3×3畳み込み演算，Lightfield画像処理を例に，各プログラムのシストリックリングアクセラレータにおける実装の詳細を説明する．

4.1 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DCNNのFC（Full Connect）層も行列積で実装される．図12に基本となるソースコードを示す．最も初歩的な高速化は図内のループ変数jのfor文とkのfor文を交換する方法である．配列b[*]へのアクセスが連続アドレスとなるためキャッシュのヒット率が向上するうえに，SIMDロード命令によって一度のロードでSIMD幅分のデータがレジスタに乗り，そのまま演算が行われる．この手法はkjループ交換と呼ばれている．次に行われる高速化手法として，テンポラルブロッキングがある．これは入力データと出力データのサイズがキャッシュに収まるサイズに問題を分割し，キャッシュのヒット率を向上させる手法である．この際の途中演算結果を置いておく記憶領域をテンポラル領域と呼ぶ．

アプリケーションの実装は単精度浮動小数点を対象とする．IMAXではテンポラルブロッキングを参考にし，LMMのサイズを一般的なプロセッサのL1キャッシュサイズである32Kbyteとして実装していく．入力データの転送回数を最小限にするため，テンポラル領域が32Kbyteであることを踏まえ,64×64の行列にする．

```
for(i=0; i<N; i++)
  for(j=0; j<M; j++)
    for(k=0; k<L; k++)
      c[i*M+j] += a[i*L+k] * b[k*M+j];
```

図 12 行列積ソースコード

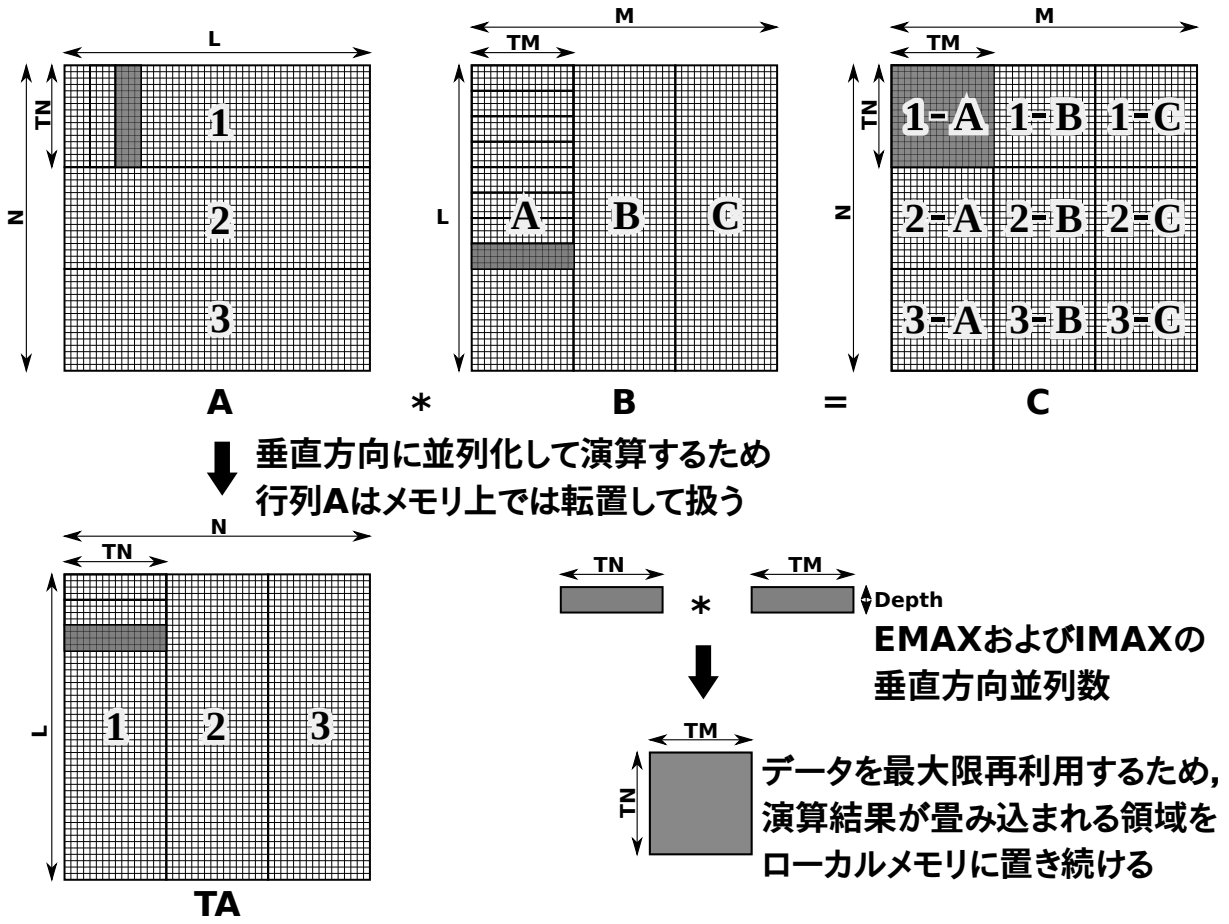


図 13 行列積の実装方法

一般的なプロセッサでは kj ループ交換により，配列 b[*] に保存されている行列 B は実質転置行列の様に扱われるが，シストリックリングアクセラレータでは行列 A が深さ方向に連続させることが最もアクセス効率が良い．そのことを図 13 で説

```
for(kk=0; kk<L; kk+=DP) /* blocking exection */
  for (ij=0; ij<N*M; ij+=SIMDW)
    for (ii=0; ii<DP; ii++) /* vertical (pipelined) */
      for (jj=0; jj<SIMDW; jj++) /* horizontal (parallel) */
        c[ij+jj] += ta[N*kk+N*ii+(ij/M)]
                        * b[M*kk+N*ii+(ij&(M-1))+jj];
```

図 14 IMAX 用行列積疑似コード

明する．まず行列を途中演算結果が LMM に乗り切るサイズにまで分割することを考える．この結果 LMM のテンポラル領域が TN×TM だとすると行列 A は TN の長さずつ縦に分割する．行列 B は TM の長さずつ横に分割する．分割して出来た帯はそれぞれ，TN×TM のテンポラル領域に途中演算結果を出力していく．シストリックリングアクセラレータはこの帯を垂直方向の並列数で分割して，分割した領域を高速に処理していく．

図 14 に並列化を考慮した行列積の擬似コードを示す．DP は垂直方向の並列数を表し，SIMDW は垂直方向の並列数を表している．今回の実装では IMAX32 段に演算の写像を行うため DP は 32 となる．IMAX での SIMDW は EMAXV で横に並んだ演算ユニット 4 基が行っていた演算を単一の演算ユニットで 4 サイクルかけて実行することで同等の演算を行う．また，シミュレータのモデルでは EMAXV と IMAX のレジスタのワード長は 64bit であり，単精度浮動小数点演算は 2 並列で行う．よって SIMDW は 8 となる．IMAX の LMM には転置した行列 A の一部として TN×Depth サイズのデータと，行列 B の一部として TM×Depth サイズのデータが展開される．

図 15 に IMAX へ実際に写像した際の各レジスタと LMM の振る舞いを示す．テンポラル領域には EMAXV であれば 4 つのデータ C10，C11，C12，C13 が 1 サイクルで同時に書き込まれ，IMAX では 4 サイクルかけて書き込まれる．それぞれの演算器は，上段の演算器から出力された値に加算するように浮動小数点積和演算を行う．行列 B のデータは毎サイクル 4 データずつずらしながらレジスタに展開され，行列 A のデータは TM サイクルごとにレジスタに展開するアドレスを加算する．

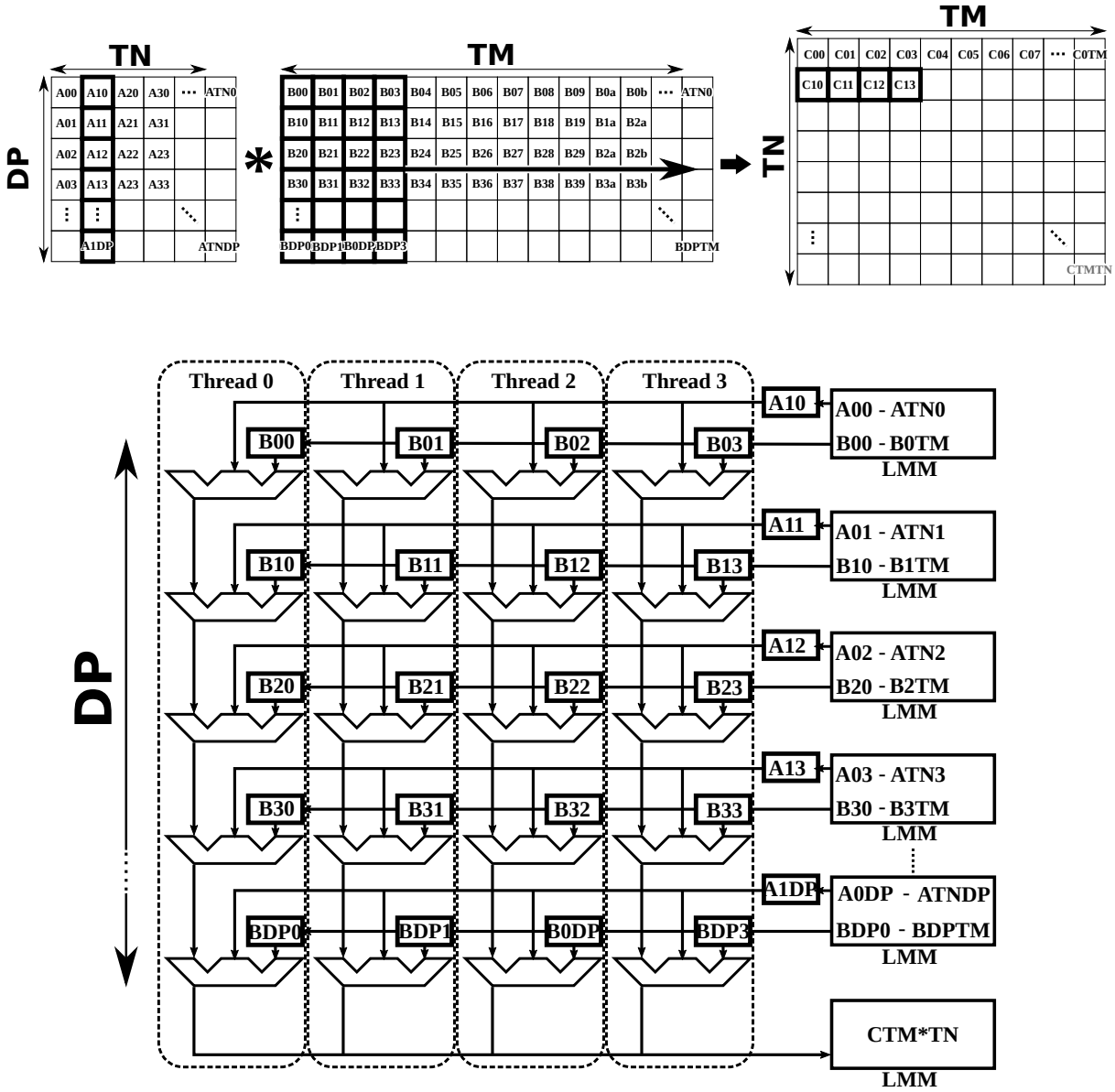

TN
TM
TM
DP
TN
A00 A10 A20 A30 ⋯ ATN0
A01 A11 A21 A31
A02 A12 A22 A23
A03 A13 A23 A33
A1DP ATNDP
B00 B01 B02 B03 B04 B05 B06 B07 B08 B09 B0a B0b ⋯ ATN0
B10 B11 B12 B13 B14 B15 B16 B17 B18 B19 B1a B2a
B20 B21 B22 B23 B24 B25 B26 B27 B28 B29 B2a B2b
B30 B31 B32 B33 B34 B35 B36 B37 B38 B39 B3a B3b
BDP0 BDP1 B0DP BDP3 BDPTM
C00 C01 C02 C03 C04 C05 C06 C07 ⋯ C0TM
C10 C11 C12 C13
CTMTN
Thread 0
Thread 1
Thread 2
Thread 3
DP
A10
A00 - ATN0
B00 - B0TM
LMM
A11
A01 - ATN1
B10 - B1TM
LMM
A12
A02 - ATN2
B20 - B2TM
LMM
A13
A03 - ATN3
B30 - B3TM
LMM
A1DP
A0DP - ATNDP
BDP0 - BDPTM
LMM
CTM*TN
LMM

図 15 行列積の実装

4.2 近接ステンシル計算

3×3 のステンシル計算は画像処理における平滑化，尖鋭化さらにエッジ検出や，物理モデルのシミュレーション，そして DCNN の畳み込み層においても利用されている応用範囲の広い計算パターンである．図 16 に 3×3 のステンシル計算の基本となるソースコードを示す．3×3 のステンシル計算は 3 点ずつに連続したアドレスに存在するデータを参照する．そしてすべてのデータに対してカーネルと呼ばれるデータとの積を演算し，積の結果の和を出力とする．これを出力データ × カーネルサイズ分繰り返す．

図 20 にシストリックリングアクセラレータへの高速な 3×3 ステンシル計算の実装を擬似コードで示す．シストリックリングアクセラレータにおける高速化ではパイプラインとして垂直方向の並列性を最大限に生かすため，3 点の連続したアドレスを参照する計算を垂直方向に配置する．水平方向の並列性は j をインクリメントした次のループで演算が行われる予定であった演算を配置する．IMAX では前節で説明したように，縦に並んだ複数ユニットへのデータブロードキャストはほぼゼロコストで行われるため，垂直方向に段数が増えてもデータ転送のレイテンシは演算に必要なラインバッファサイズである 3×W のままである．さらに，1 列分の演算が終わると次に転送の必要な演算データは最後に転送したデータのアドレスから W だけ進んだアドレスまでのデータであり，それ以外のデータは LMM にすでに転送されているデータを再利用する．

図 18 に IMAX へ実際に写像した際の各レジスタと LMM の振る舞いを示す．垂直方向に 3 ユニットずつ同じデータが LMM に転送される．各演算器は事前にレジスタに転送される重みデータ k0-k8 と 4 点連続したアドレスにあるデータをロードし積和演算をパイプラインとして処理していく．同じデータを持つ LMM は上段に存在するユニットで参照したアドレスよりデータ分インクリメントしたアドレスから連続して 4 点データをロードする．そして上から流れてくる積和演算の結果も入力として積和演算を行う．これを垂直方向に並べたカーネルデータの回数分行い最終的に 4 点が並んだ演算結果を LMM にストアする．

```
for(i=0; i<H; i++)
  for(j=0; j<W; j++){
    c[i*W+j] += a[(i-1)*W+j-1] * k[0];
    c[i*W+j] += a[(i-1)*W+j  ] * k[1];
    c[i*W+j] += a[(i-1)*W+j+1] * k[2];
    c[i*W+j] += a[ i   *W+j-1] * k[3];
    c[i*W+j] += a[ i   *W+j  ] * k[4];
    c[i*W+j] += a[ i   *W+j+1] * k[5];
    c[i*W+j] += a[(i+1)*W+j-1] * k[6];
    c[i*W+j] += a[(i+1)*W+j  ] * k[7];
    c[i*W+j] += a[(i+1)*W+j+1] * k[8];
  }
```

図 16 3×3 畳み込みソースコード

```
for(i=0; i<H; i++)
 for(j=0; j<W; j+=SIMDW)
  for (jj=0; jj<SIMDW; jj++){ /* horizontal (parallel) */
   c[i*W+j]+=a[(i-1)*W+jj-1]*k[0]; /* vertical (pipelined) */
   c[i*W+j]+=a[(i-1)*W+jj  ]*k[1]; /* vertical (pipelined) */
   c[i*W+j]+=a[(i-1)*W+jj+1]*k[2]; /* vertical (pipelined) */
   c[i*W+j]+=a[ i   *W+jj-1]*k[3]; /* vertical (pipelined) */
   c[i*W+j]+=a[ i   *W+jj  ]*k[4]; /* vertical (pipelined) */
   c[i*W+j]+=a[ i   *W+jj+1]*k[5]; /* vertical (pipelined) */
   c[i*W+j]+=a[(i+1)*W+jj-1]*k[6]; /* vertical (pipelined) */
   c[i*W+j]+=a[(i+1)*W+jj  ]*k[7]; /* vertical (pipelined) */
   c[i*W+j]+=a[(i+1)*W+jj+1]*k[8]; /* vertical (pipelined) */
  }
```

図 17 3×3 畳み込み EMAX 擬似コード

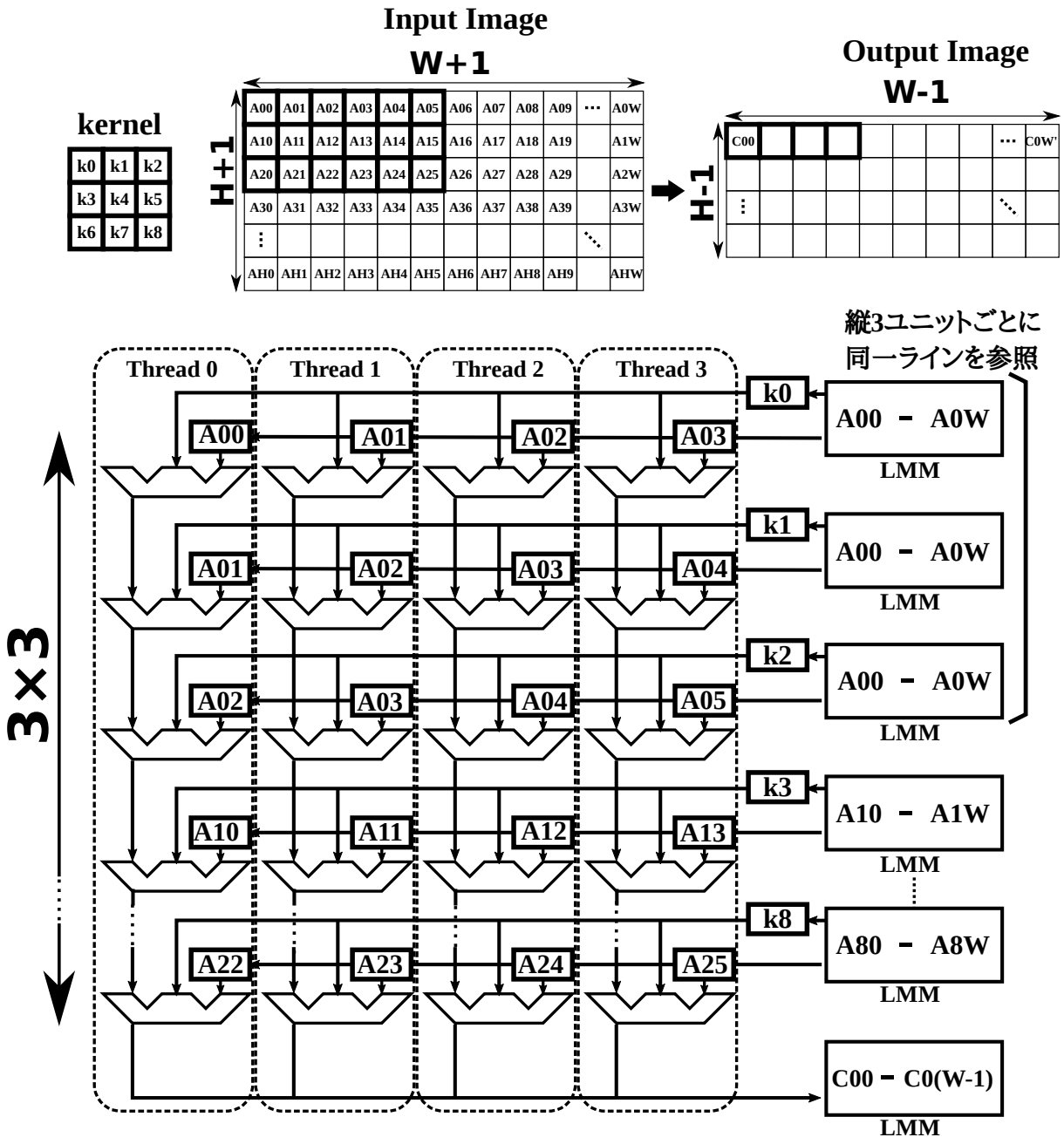
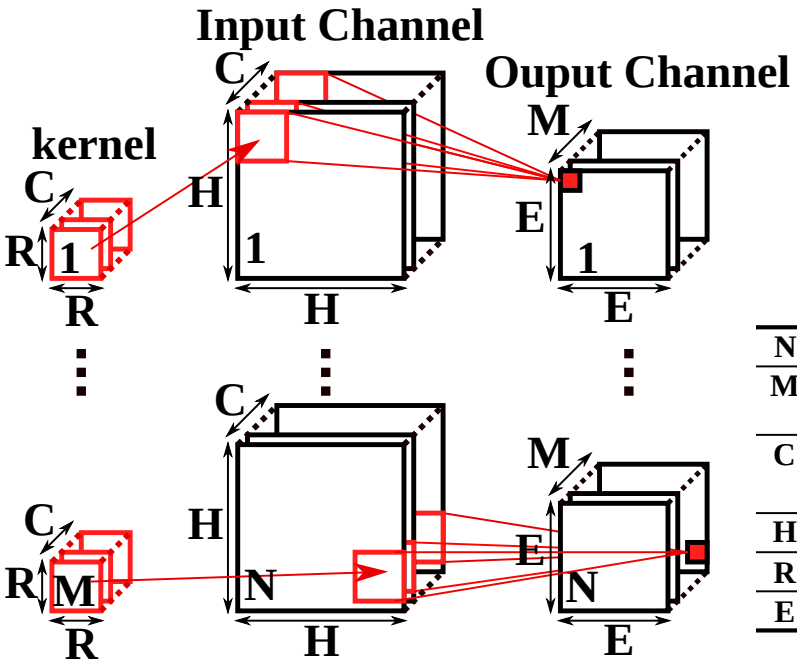

Input Image
W+1
H+1
kernel
k0 k1 k2
k3 k4 k5
k6 k7 k8
A00 A01 A02 A03 A04 A05 A06 A07 A08 A09 ⋯ A0W
A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A1W
A20 A21 A22 A23 A24 A25 A26 A27 A28 A29 A2W
A30 A31 A32 A33 A34 A35 A36 A37 A38 A39 A3W
AH0 AH1 AH2 AH3 AH4 AH5 AH6 AH7 AH8 AH9 AHW
Output Image
W-1
H-1
C00 ⋯ C0W'
縦3ユニットごとに
同一ラインを参照
Thread 0
Thread 1
Thread 2
Thread 3
3×3
k0
k1
k2
k3
k8
A00 – A0W
A10 – A1W
A80 – A8W
C00 – C0(W-1)
LMM

図 18 3×3 畳み込実装

ここでDCNNへの実装までを考慮したLMMの使用率について注目する．まず，行列積と畳み込み演算の場合で一時的に利用するLMMのサイズに大きな差がある．N×Nの行列積では垂直方向の並列性を利用するために，入力データを保存する各LMMのサイズはNでよいのに対して，演算結果を一時的に保存するLMMのサイズはN×N必要になる．そのため入力データを置いているLMMはほとんど使いきれていないことになる．畳み込み演算において各LMMは入力データと出力データのどちらもラインバッファとして扱われるため，それぞれでLMMを使い切ることが可能である．しかし図19に示すように，DCNNでは複数の入力画像と重みデータ間で3次元方向にも畳み込みが発生する．図20に示すDCNNの擬似コードでは入力データを最大限利用するように最適化すると出力データを保存する領域には広いアドレス空間が必要になる．これらは多重ループをアクセラレーションする場合，垂直，水平方向（変数i，j，k0，k1に相当）に並列化するため最内ループからアンローリングする戦略を取る際，複数画像間つまりさらに外側のループ（変数cに相当）をアンローリングするとアドレスの参照範囲が大きく増加するため，一時的に必要な記憶領域が増えることを意味している．プロセッサであれば階層記憶がうまく作用し，内側のループでアクセスされるアドレスが上位のキャッシュに乗り，外側のループに行くにつれL2，L3キャッシュに乗るようになる．しかし，前述のとおり，Lightfield画像処理の様にキャッシュのラインサイズを超えるような参照が最内ループに存在すると性能は著しく劣化する．これらを考慮すると，IMAXでも畳み込まれるデータを置く記憶領域をLMMのみでなくプログラマが管理可能な階層を持つ記憶領域に置くことにより，入力データサイズと出力データサイズのアンバランスを解消し，さらにメモリ使用率を向上させることが可能であると考えられる．



N	batch size
M	# of 3D kernels or output channels
C	# of input channels or kernel channels
H	input width/height
R	kernel width/height
E	output witdh/height

図 19 DCNN の畳み込み演算

```
for n in 1..N /* 選択batch */
 for m in 1..M /* 出力チャネル選択 */
  for c in 1..C /* 入力チャネル選択 */
   for i in 1..E /* 出力データ選択 */
    for j in 1..E /* 出力データ選択 */
     for k1 in 1..R /* kernel データ選択 */
      for k0 in 1..R /* kernel データ選択 */
       oc[n*M*E*E+m*E*E+i*E+j]
        += ic[n*M*E*E+c*H*H+(i+k1)*H+j+k0] * k[k1*R+K0];
```

図 20 DCNN 擬似コード

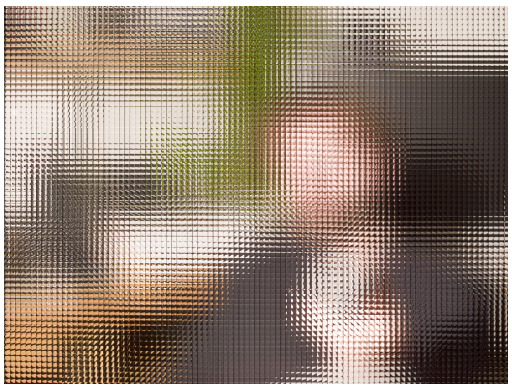


図 21　Lightfield 画像

図 22　リフォーカス画像

4.3　Lightfield 画像処理

図 21 に Lightfield 画像を載せる．Lightfield 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，奥行き画像生成が可能な画像である．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 22 に図 21 をリフォーカス処理した画像を示す．Lightfield 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著しくアプリケーションの性能が落ちることが分かっている．しかし，EMAX 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 23 に示すシストリックリングアクセラレータによる Lightfield 画像のリフォーカス処理の簡易的な実装例を示し，IMAX に同等のアプリケーションを実装する．

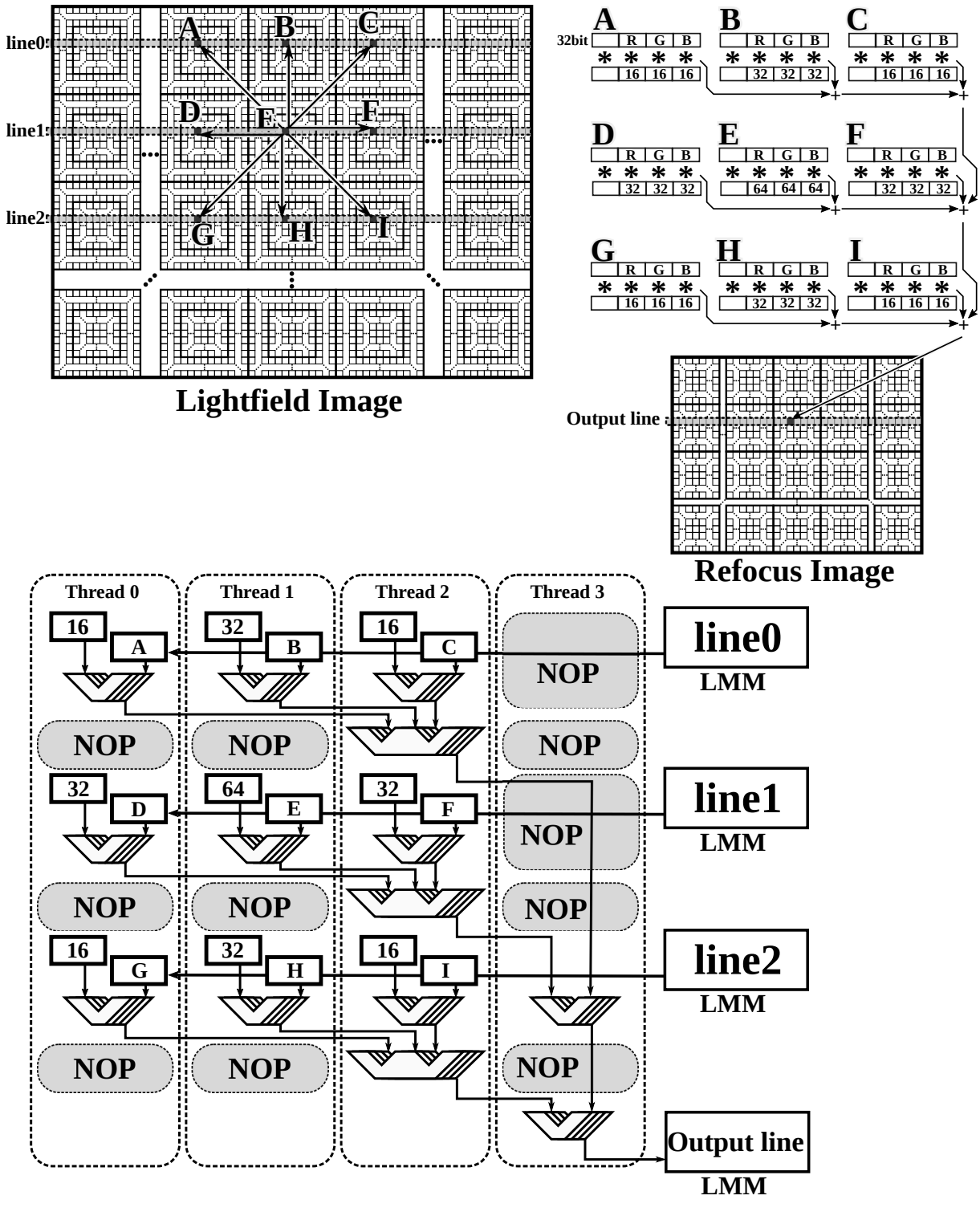


図 23 Lightfield 画像リフォーカス処理の実装

5. 評価

まずEMAXVおよびIMAXのサイクルアキュレートシミュレータを用いて，行列積，3×3畳み込み演算，ライトフィールド画像処理の3種類のアプリケーションについてサイクル数を計測し，性能比較を行う．行列積については基本的な浮動所数点演算性能を評価，3×3畳み込み演算は連続していないアドレスへのアクセスを持つ浮動小数点演算の性能を評価，Lightfield画像処理については広く離散的なアドレスへアクセス性能を評価する．この結果より，IMAXがEMAXVと同等な性能を持つことを示す．次にVerilog-HDLによるRTLモデルを作成し，EMAXVとIMAXの回路面積の見積もりを行い，面積削減の効果を明らかにする．最終的に面積で正規化した性能をEMAXVとIMAXで比較を行い，提案手法の有効性の評価を行う．

5.1 シミュレーションによる性能評価

表1にサイクルアキュレートシミュレータのプロセッサモデルを示す．ホストプロセッサはARMv8プロセッサを想定している．これはXilinxのZynq UltraScale+というCortex-A53プロセッサとFPGAが実装されているSoC上でFPGAにIMAXをエミュレートさせることを想定している．評価アプリケーションはシングルコア，シングルスレッド，2.4Ghzで動作する．表2に評価を行うサイクルアキュレートシミュレータで表現するEMAXVのモデルとIMAXのモデルのパラメータを示す．LMMのサイズは64×64の行列積の途中演算結果が収まるサイズである32KByte

表1 ホストプロセッサのシミュレータモデル

ホストプロセッサ	ARMv8
命令セット	AArch64
最大コア数	4（本評価では1コアのみ使用）
最大スレッド数	16（本評価では1スレッドのみ使用）
ARMv8動作周波数 [GHz]	2.4

表 2 EMAXV/IMAX のシミュレータモデル

	EMAXV	IMAX
段数	64	64
動作周波数 [MHz]	300	1,200
LMM サイズ [KByte]	32	32

表 3 各アプリケーションの問題サイズ

	行列積	3 × 3 畳み込み	Lightfield 画像処理
size of cyctolic ring	34	17	11
Input data size	NxNx2	(W+2)x(H+2)+K^2	LWxLH+K^2
Output data size	NxN	WxH	RWxRH
FMA/OPs	NxNxN	WxHxK^2	RWxRHxK^2

†N:64 Matrix size

†W:2048 Output width, H:100 Output height, K:3 kernel size

†LW:7240 Lightfield width, LH:5433 Lightfield height

†RW:1000 Refocus width, RH:765 Refocus height

を設定している．これは一般的なプロセッサの L1 キャッシュの容量に対応しており，プロセッサの行列積の高速化手法であるテンポラルブロッキングを参考にしている．

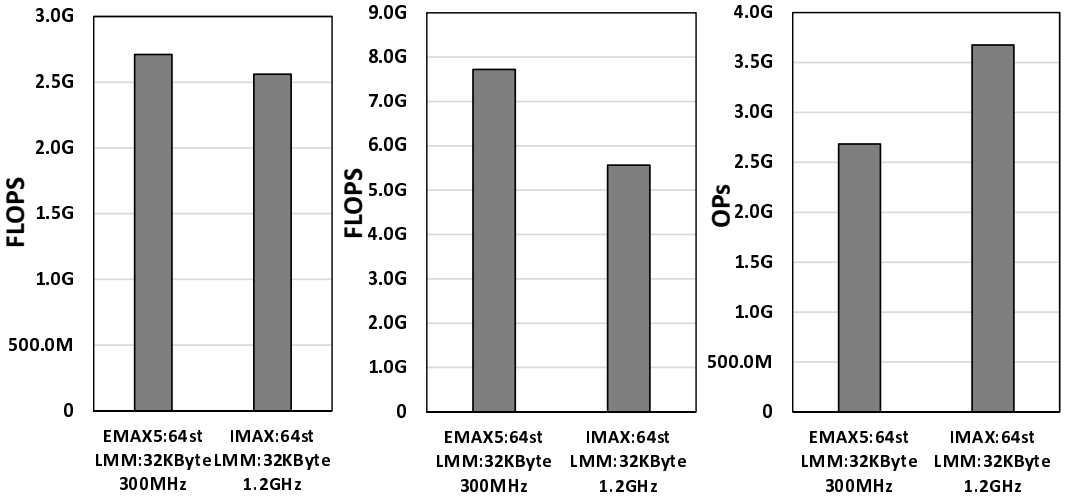
表 3 に各アプリケーションの問題サイズを示す．行列積は 64×64 の浮動小数積和演算の回数であり，3×3 畳み込みは 2048×100 のサイズの配列に対し 3×3 のステンシル演算を行う際の浮動小数積和演算の回数である．Lightfield 画像処理の OPs は離散的なアドレスへのメディア演算命令である．

表 4 にシミュレーションによるホスト CPU サイクル数の評価結果を示す．図 24 にサイクル数の評価結果より求めた演算性能を示す．行列積と畳み込み演算について，EMAXV よりの性能が落ちているのは，トランザクションの制御をプロセッサで行うようにしたため，シングルスレッドで動作させた際のオーバーヘッドが表れ

ている．更に，現在の IMAX 用のカーネルコンパイラで提案手法で述べたステージ間でのブロードキャストとラインサイズを超える連続アドレスのバースト転送を行うための最適化が実装できていないため，コンパイル時に決定できるデータ転送の要否判定を実行時に行っていることが原因であることが分かっている．そこで，3×3 畳み込みアプリケーションのデータトランザクションを手動で記述した場合の性能を図 25 に示す．この結果より，トランザクションのオーバーヘッドをコンパイル時に解決すれば，IMAX が EMAXV と同等の演算性能を持つといえる．

表 4　畳み込み演算と Lightfield 画像処理の実行サイクル数

	EMAXV	IMAX
行列積	464,173	491,283
3 × 3 畳み込み演算	1,145,464	1,590,418
Lightfield 画像処理	24,632,286	17,986,415



(a)行列積　　(b)3×3畳み込み　　(c)Lightfeild画像処理

図 24　各アプリケーションの性能

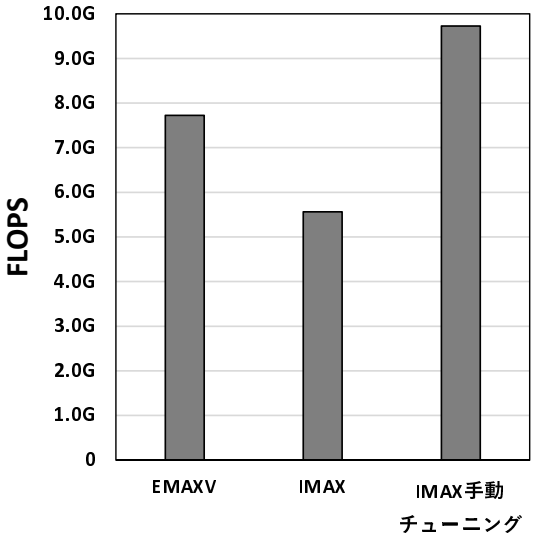


図 25　3 × 3 畳み込みの手動チューニング

5.2 論理合成による面積評価

IMAX の 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と配線混雑の評価を行った．表 5 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 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 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．表 6 と図 26 に論理合成結果を示す．組み合わせ回路の面積を示している Combination area に注目すると，およそ半分ほどの面積に削減できたことが分かる．半分しか下がっていないのは主記憶と LMM 間のバスをパイプライン化したうえで，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 are に注目すると，Combination area 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．

LMM に注目するとメモリマクロの面積削減の効果が大きいことが分かる．ここで注意が必要なのは，IMAX は入力データの種類が複数ある場合，LMM をスレッド間で分割して利用する． 行列積の場合，入力データが A 行列と B 行列の 2 種類存在し，EMAXV は IMAX の 2 倍 LMM を利用できるため，IMAX のメモリマクロによる面積効率は高々2 倍になる．しかし，今回の畳み込み演算の場合はカーネルの重みデータはレジスタに置かれるため，入力データは 1 種類であり，この場合，EMAXV は 4 つのローカルメモリに同一のデータをコピーすることになるので確かに IMAX ではメモリマクロの面積を 4 分の 1 に削減できることになる．

表 5 評価に使用した環境

ASIC 向け論理合成環境	Design Compiler M-2016.12-SP2
テクノロジ	28nm

表 6 EMAXV/IMAX 64stage の回路面積

	EMAXV	IMAX	IMAX with RMM
Clock constraints[MHz]	300	1,200	1,200
Combinational area[μm^2]	4,074,459	1,890,315	1,890,315
Buf/Inv area[μm^2]	173,855	189,093	189,093
Noncombinational area(Register)[μm^2]	773,047	1,104,633	1,104,633
Macro/Black Box area(LMM)[μm^2]	22,374,142	5,593,536	3,058,733
Macro/Black Box area(RMM)[μm^2]	0	0	488,832
Total cell area[μm^2]	27,221,649	8,588,483	6,731,607

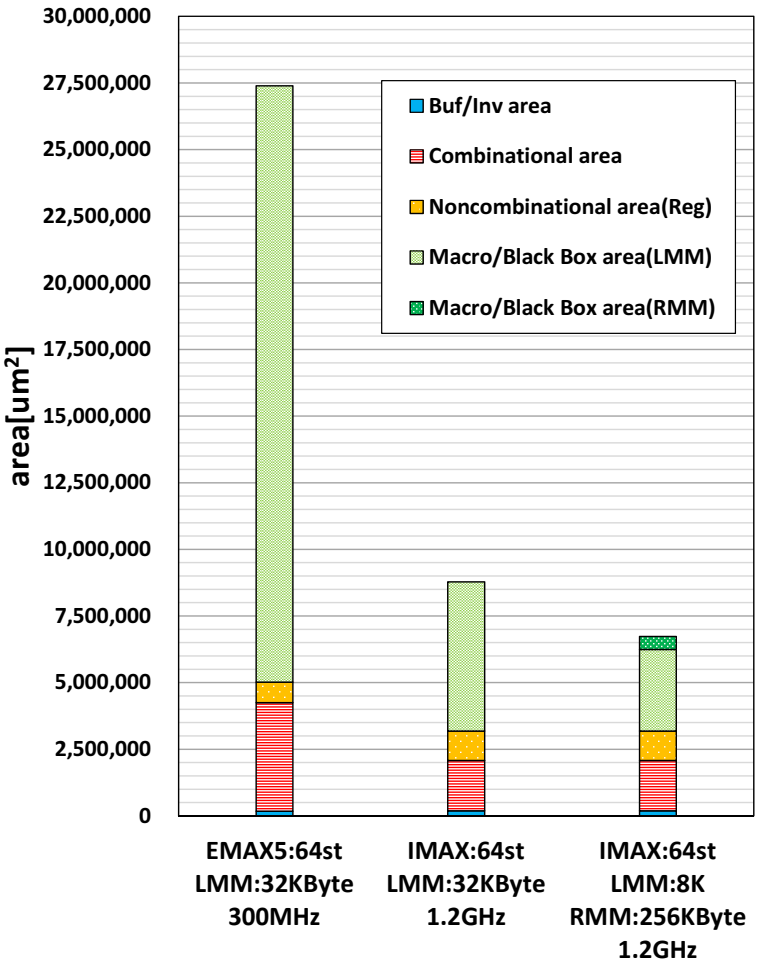


図 26 EMAXV/IMAX 64stage の回路面積

面積評価では，DCNN の実装を考慮した畳み込みデータが置かれるメモリとして RMM（Recurrent Memory）を実装した場合の面積の見積もりを載せる．RMM を実装した場合の IMAX は LMM 8KByte，RMM 256KByte，さらに RMM はシングルポートのメモリとして見積もりを行った．LMM 小型化の恩恵を考えると，容量が 4 分の 1 なのに対して面積が約半分となっている．これはメモリマクロは小型化するにつれて，SRAM の構成要素であるセンスアンプ回路の占める面積が支配的になることが原因として考えられる．

表 7 に EMAXV と IMAX の配線混雑を評価として，fanout が 100 を超える配線の本数を示す．fanout の高い配線は，敷き詰め率を悪化させ，周波数向上の障壁となる．この結果より，周波数の制約を厳しくしても IMAX のもつ fanout の高い配線は EMAXV に比べ 300 本以上減少しており，配線混雑を改善しているといえる．

表 7 EMAXV/IMAX 8stage の配線混雑評価

	EMAXV	IMAX
Clock constraints[MHz]	300	1,200
# of high fanout（More than 100） nets	750	441

5.3 面積当たり性能

図 27 に（a）行列積,（b）3×3 畳み込み演算,（c）lightfield 画像処理の演算性能を面積で正規化した演算性能を示す．行列積では 2.94 倍，畳み込み演算では 2.25 倍，lightfield 画像処理では 4.34 倍の面積当たり性能を達成している．畳み込み演算の面積当たり性能は，手動チューニング前の実装による性能を正規化した．

これらの結果より，CGRA の問題点であった面積効率の悪さを提案手法であるマルチスレッド化によって改善可能であることを明らかにした．しかしながら，畳み込み演算の例にもあるように，データのトランザクションの管理をホストプロセッサに動的に行わせることによるオーバーヘッドが無視できないことが明らかとなっている．この問題を解決するためにはコンパイラの静的な解析をさらに改善するか，あるいはプログラマにトランザクションの管理を書かせるか，いずれかの方法をとる必要があり，今後の課題となっている．

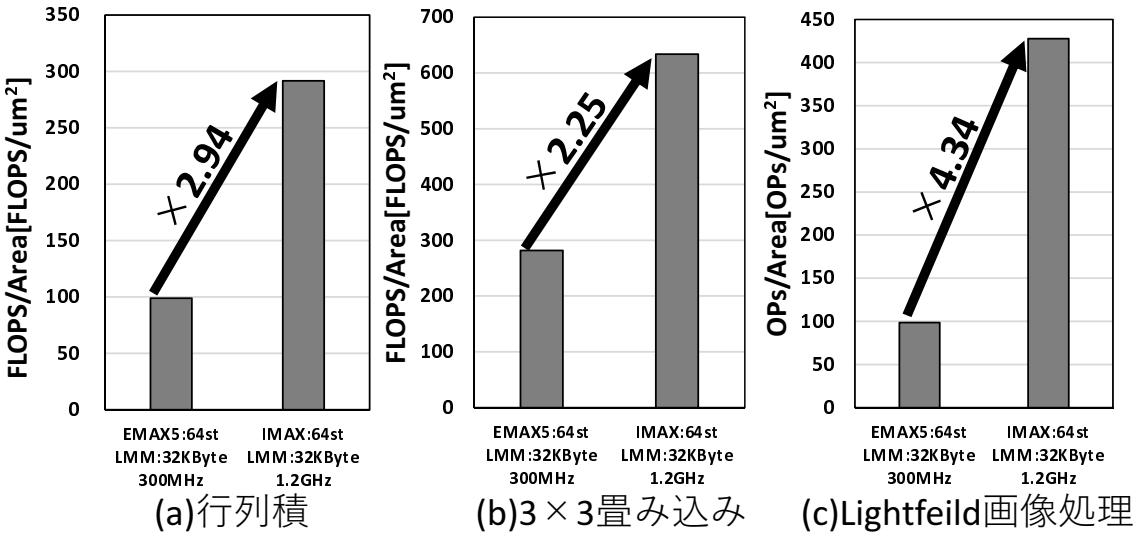


図 27 面積当たり性能

ほかにも，入力データを置くための記憶領域と演算結果の畳み込むための記憶領域サイズの関係がアンバランスであるアプリケーションや，多重ループの写像写像において可能な限りループアンローリングして，入力データの主記憶と LMM 間の通信量を削減するために RMM が有効であることが分かっているため，RMM を利用してアプリケーションを実装した場合の演算性能の評価も課題である．しかし，RMM と LMM 間のデータトランザクションを考えると，コンパイラで静的に解析させるよりも，プログラマに管理させるほうが性能の見積もりが容易であると考えている．もちろんこれらはアプリケーション実装のコストを増大させるためトレードオフの関係にある．

6. おわりに

本稿ではCGRAの問題点あった，面積効率を改善する手法としてマルチスレッド機構を実装したシストリックリングアクセラレータとしてIMAXを提案した．従来のCGRAの面積効率の悪さは配線混雑による最高周波数の低下とデータコピーを保持するためメモリを複数実装しなければならないことが原因であった．そこでIMAXでは4スレッドで単一の演算ブロックを使いまわし，FPUのパイプライン化とメモリへの時分割アクセスを行う方法について説明した．さらに主記憶とLMM間をつなぐバスの遅延についても注目し，連続アドレスを各ユニットで分割して受け取り可能な機構を持ったパイプラインを新たに提案した．これにより，2次元データをラインサイズが超えるサイズであっても一度のバースト転送に載せることが可能となり，また垂直方向にデータをブロードキャストもほとんどオーバーヘッドなしで可能となり，現段階ではプログラマーがトランザクションを手動で記述可能であれば，アプリケーションをさらに高速化できることを明らかにした．

本稿では先行研究であるEMAXVと面積当たり性能を比較するため，サイクルアキュレートシミュレータを用いて，行列積，3×3畳み込み演算，Lightfield画像処理の3種類のアプリケーションについて性能比較した．そして，Verilog-HDLによるRTLモデルも実装し，28nmテクノロジを利用した場合の回路面積の見積もりを行った．これらの結果より，IMAXはEMAXVと比べたとき2.25倍から4.34倍の面積当たり性能を持つことが分かった．

謝辞

研究活動はもちろんのこと，研究室での交流に至るまで常日頃より的確なご助言やご指導，ご支援を頂いた本学の中島康彦教授に心より深く感謝の意を表します．本論文をご精読いただき，また発表に対して貴重なご意見を頂いた本学の井上美智子教授に深く感謝したします．研究室ミーティングにおいて平素より鋭い視点からご意見いただき，本論文の執筆にご助言を頂いた本学の中田尚准教授に深く感謝いたします．研究者としての姿勢や計算機科学の面白さをご教授頂いた元本学助教，現北海道大学の高前田信也准教授に深く感謝いたします．Renyuan ZHANG 助教，TRAN Thi Hong 助教には講義において丁寧なご指導ご鞭撻を賜りましたことを深く感謝いたします．就職活動から研究活動に至るまでを支えてくださった一倉孝宏氏，2 年間の研究生活で多くを支えてくださった三谷剛正氏，同期の福岡久和氏に深く感謝いたします．研究だけでなく日々の生活や就職活動においても様々なご助言を頂いた加藤大真氏，亀田友哉氏，嶋谷知氏，藤本啓輔氏を始めとするコンピューティング・アーキテクチャ研究室 OB の方々に深く感謝いたします．そして普段より研究室の運営維持のために多くの係を引き継いでくださった平賀由利亜氏，菊谷雄真氏，上竹規之氏，山根弘樹氏 ，吉田怜矢氏ら研究室の後輩たちに深く感謝いたします．

最後に支えてくれた家族に心より感謝申し上げます．

参考文献

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "imagenet classification with deep convolutional neural networks". *Commun. ACM*, 60(6):84–90, May 2017.

[2] Karen Simonyan and Andrew Zisserman. "very deep convolutional networks for large-scale image recognition". *CoRR*, abs/1409.1556, 2014.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "going deeper with convolutions". *CoRR*, abs/1409.4842, 2014.

[4] "technology quarterly after moore's law". http://www.economist.com/technology-quarterly/2016-03-12/after-moores-law.

[5] "lones, why migration to 20nm bulk cmos and 16/14nm finfets is not best approach for semiconductor industry". https://pdfs.semanticscholar.org/61f7/2c4be61632090e728ff7dc29f77678568582.pdf.

[6] Norman P. Jouppi and et al. Young. "in-datacenter performance analysis of a tensor processing unit". In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 1–12, New York, NY, USA, 2017. ACM.

[7] 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.

[8] "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.

[9] Chen, Yu-Hsin and Krishna, Tushar and Emer, Joel and Sze, Vivienne. ”eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks”. In *IEEE International Solid-State Circuits Conference, ISSCC 2016, Digest of Technical Papers*, pages 262–263, 2016.

[10] 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.

[11] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. ”EIE: efficient inference engine on compressed deep neural network”. *CoRR*, abs/1602.01528, 2016.

[12] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. ”deep learning with limited numerical precision”. *CoRR*, abs/1502.02551, 2015.

[13] 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, Sept 2013.

[14] 稲垣慶和, 原祐子, 姚駿, and 中島康彦. ”リング型アレイアクセレータ向け演算ライブラリの実装と性能評価”. In **研究報告計算機アーキテクチャ（*ARC*）**, volume 2013, pages 1–6. 一般社団法人情報処理学会, jul 2013.

[15] 藤原知広, 姚駿, 原祐子, and 中島康彦. In *”***リング型アレイアクセラレータのマクロパイプライン化による性能見積もり***”*, volume 2013, pages 1–6. 一般社団法人情報処理学会, jul 2013.

[16] 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.

[17] 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.

発表リスト

1. 福岡久和, 山野龍佑, 高前田伸也, 中島康彦, “高位合成ツールを用いたFPGA並列コンピューティングの可能性検討 “, 信学技報, CPSY2016-26, pp.181-186, Aug. (2016)

2. 福岡久和, 山野龍佑, 中島康彦, “各種FPGAによる畳み込み演算向けシストリックリングの実装と評価 “, CPSY研究会, 2017-05-23, May. (2017)

3. 山野龍佑, 中島康彦, “時分割多重実行によるシストリックリングの面積効率向上手法 “, 信学技報, vol.117, no.44, CPSY2017-6, pp.27-32, May. (2017)

4. 一倉孝宏, 山野龍佑, 福岡久和, 中島康彦, “DCNNに最適なCGRAの探索と予備評価 “, 信学技報, vol.116, no.416, CPSY2016-114, pp.49-54, Jan. (2017)

5. 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, pp.31-36, Jan. (2018)