修士論文

シストリックアレイと高速リンクの組み合わせによる高効率マルチチップシステムの評価

岩本 淳
奈良先端科学技術大学院大学
先端科学技術研究科
情報理工学プログラム

主指導教員: 中島 康彦 教授
コンピューティング・アーキテクチャ研究室（情報科学領域）

令和 2 年 3 月 13 日提出

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

岩本 淳

審査委員：

中島 康彦 教授	(主指導教員，情報科学領域)
井上 美智子 教授	(副指導教員，情報科学領域)
中田 尚 准教授	(副指導教員，情報科学領域)
Renyuan Zhang 助教	(副指導教員，情報科学領域)
Tran Thi Hong 助教	(副指導教員，情報科学領域)

シストリックアレイと高速リンクの組み合わせによる高効率マルチチップシステムの評価*

岩本 淳

内容梗概

機械学習での活用を目的とし，膨大なセンサが収集した一次元データをクラウドセンターに集約し，学習・推論するシステムが多く報告されている．ここで，エッジデバイスには低コストかつ低電力，高効率な計算基盤が求められている．そこで，演算器を格子状に配置するシストリックアレイ・アーキテクチャが注目されているが，エッジ側においては，要求性能に応じたスケーラビリティが求められる一方，コスト削減のために外部メモリバスの増設が容易ではない．また，格子構造のまま実装すると長距離配線や配線混雑が動作周波数および面積効率の低下をもたらし，パイプライン実行を妨げてしまう．さらに，複雑なデータ参照パターンを持つ演算の場合，外部メモリ参照が増加し，ローカルメモリ (LMM) の再利用効率の悪化や読み出しの遅延などが問題となる．最後に，最内ループの高速化だけではベクトル長の短い行列積や畳み込み演算の高速化に向かないなど解決すべき課題は多い．

本研究では，上記課題を改善しつつ，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シストリックアレイである In-Memory Accelerator eXtension (IMAX) を提案する．まず，複数の外部メモリバスや高機能な DMA マスタを持つことによる消費電力の増大化を改善するためのマルチチップカスケード接続機構を提案する．次に，二次元格子構造による長距離配線や配線混雑を改善し，複雑なデータ参照パターンを持つ演算への対応を可能とする列マルチスレッディング，そして読み出し遅延を改善するチップ内バス並列化を提案する．最後に，ベクトル長が短い演算に対しても十分な性能を出すことを可能とする多重ループ一括実行機構を提案する．この IMAX のプロトタイプシステムを FPGA 上に実装し，スター型，ツリー型，デイジーチェイン型の 3 種のマルチチップ構成を形成

*奈良先端科学技術大学院大学 先端科学技術研究科 修士論文, 令和 2 年 3 月 13 日.

し，評価する．評価方法は行列積や畳み込み演算，Light-Field 画像処理，ステンシル演算などから成る 10 種の評価アプリケーションを実行することにより，最大 4 チップで評価する．また，デイジーチェイン型においては，最大 8 チップでの評価を加えて行う．評価の結果，最大性能では IMAX の 1 チップ構成と比較して，スター型，ツリー型，デイジーチェーン型の 4 チップ構成はそれぞれ 2.9 倍，2.69 倍，2.54 倍の実行性能を得られることが明らかとなった．また，デイジーチェーン型の最大 8 チップ構成の評価では最大 2.9 倍 (行列積，7 チップ接続時) となった．加えて，畳み込み演算では 5 チップ接続，Light-Field 画像処理では 2 チップ接続，ステンシル演算では大気シミュレーションとして用いられる grapes は 5 チップ接続，その他のステンシル演算では 2 チップ接続が最も高速であった．行列積や畳み込み演算のようにチップ間共有データが多いとマルチチップ化の効果が大きく，デイジーチェン型においても安定した性能を出すことができるが，ステンシル演算のように単純な空間分割による演算ではマルチチップ化の効果が小さく，スター型やツリー型の方がオーバーヘッドが小さく性能が出しやすい．しかし，デイジーチェーン型においてもエッジ向け GPU である Jetson TX2 と比較して，ステンシル演算 (resid) 実行時には最大 5.4 倍の実行性能を出しつつ，他のアプリケーションにおいても高い実行性能を実現することができた．結論としては，電力制約下にあるエッジデバイス用途を前提とし，限られたメモリバンド幅であるデイジーチェーン型においても十分な性能が出せることが明らかとなった．

キーワード

シストリックアレイ, 再構成可能メモリ領域, メモリバンド幅, エッジコンピューティング，マルチスレッディング

Evaluation of High Efficiency Multi-Chip System by Combining Systolic Array and High-Speed Links*

Jun Iwamoto

Abstract

Many systems have been reported for collecting and learning one-dimensional data collected by a huge number of sensors in a cloud center for use in machine learning. Low-cost, low-power, and high-efficiency computing platforms are required for edge devices. Therefore, systolic array architecture, in which arithmetic units are arranged in a grid pattern, has been attracting attention. On the other hands, scalability performance is required, but it is not easy to add an external memory bus to reduce costs. In addition, long-distance wiring and wiring congestion cause a decrease in operating frequency and area efficiency when implemented with a lattice structure, which hinders pipeline execution. Furthermore, in the case of operations with complex data reference patterns, external memory references increase. The reuse efficiency of the local memory (LMM) deteriorates, and the readout delay becomes dominant. Finally, there are many issues that need to be solved, such as speeding up the innermost loop alone is not suitable for accelerating matrix multiples and convolutional operations with short vector length.

In this research, this paper proposes In-Memory Accelerator eXtension (IMAX), which is a systolic array that can be multi-chip without increasing the external memory interface to improve the above problems. First, this paper proposes a multi-chip cascade connection mechanism to improve power consumption due to having multiple external memory buses and sophisticated DMA masters. Next, this paper proposes column multi-threading to improve long-distance wiring and

*Master's Thesis, Graduate School of Science and Technology, Nara Institute of Science and Technology, March 13, 2020.

wiring congestion by using a two-dimensional lattice structure, enable operations with complicated data reference patterns, and parallelize buses in a chip to improve readout delay. Finally, this paper proposes a multi-loop execution mechanism that can provide sufficient performance for operations with short vector lengths. The prototype system of this IMAX is mounted on an FPGA, and three types of multi-chip configurations of star type, tree type, and daisy chain type are formed and evaluated. The evaluation method is to evaluate with up to 4 chips by executing 10 kinds of evaluation applications consisting of matrix multiples, convolution operation, Light-Field image processing, stencil operation and so on. In addition, since only 5 chips or more on the prototype system can be connected in the daisy chain type, evaluation is performed with a maximum of 8 chips. As a result of the evaluation, it is clear that the 4 chip configuration of the star type, the tree type, and the daisy-chain type can achieve 2.9 times, 2.69 times, and 2.54 times the execution performance compared to the 1 chip configuration at the maximum performance. In the evaluation of a daisy-chain type with a maximum of 8 chips, the maximum performance is 2.9 times (with matrix multiples 7 chip configuration). In addition, 5 chip configuration in the convolution operation, 2 chip configuration in the light-field image processing, 5 chip configuration to the grapes used for atmospheric simulation in the stencil calculation, and 2 chip configuration were the fastest in other stencil calculations. If there is a lot of shared data between chips such as matrix multiplication and convolution operation, the effect of multichip is large and stable performance can be obtained even in daisy-chain type. However, in operation by simple space division such as stencil operation, The effect of multi-chip implementation is small, and the star type and tree type have smaller overhead and are easier to achieve high performance. However, compared to Jetson TX2, which is a GPU for edge side, the daisy chain type can achieve up to 5.4 times the execution performance when executing stencil operation (resid), and can achieve high execution performance in other applications. In conclusion, it is clarified that sufficient performance can be obtained even in a daisy chain type with a limited memory bandwidth, assuming the use of edge devices under power constraints.

Keywords:

Systolic Array, Reconfigurable memory space, Memory bandwidth, Edge computing, Multithreading

Contents

1. はじめに **1**
1.1 背景 1
1.2 研究目的 2
1.3 構成 2

2. 関連研究 **3**
2.1 GPU 3
2.2 DSA 5
2.3 FPGA 5
2.4 シストリックアレイ 6

3. 従来型シストリックアレイの課題とその改善策 **8**
3.1 課題 8
3.2 IMAX の基本構造 9
3.3 IMAX の動作シーケンス 11

4. 提案手法 **13**
4.1 列マルチスレッディング 13
4.2 チップ内バス並列化 14
4.3 多重ループ一括実行機構 14
4.4 カスケード接続機構 17
4.5 アダプティブ命令シフト 20

5. FPGA プロトタイプシステムと評価アプリケーションの準備 **21**
5.1 ZCU102 へのホスト機能実装 23
5.2 VU440 への IMAX 実装 23
5.3 マルチチップ構成 23
5.4 アプリケーションプログラムの準備 26
5.4.1 評価アプリケーション 27
5.4.2 行列積 28
5.4.3 畳み込み演算 29
5.4.4 Light-field 画像処理 31
5.4.5 3 種類のアプリケーションプログラムの総括 32

6. 評価と考察 **37**

7. おわりに **44**

謝辞 **45**

参考文献 **46**

発表リスト **49**

受賞 **49**

List of Figures

1 GPU の構造 3
2 Eyeriss の構造 4
3 アイランドスタイル FPGA の構造 6
4 TPU の構造 7
5 IMAX の演算ユニット 9
6 命令とメモリアクセスパターンの対応例 10
7 IMAX の動作シーケンス 11
8 一般的シストリックアレイと IMAX のマルチスレッディング 13
9 チップ内バス並列化 15
10 多重ループ実行機構を適用した IMAX の演算ユニット 16
11 For 文によるマルチチップおよび多重ループ制御 17
12 Read-Modify-Write 機能 18
13 IMAX マルチチップ構成 19
14 ARMv8 + IMAX FPGA プロトタイプシステムのブロック図 21
15 ARMv8 + IMAX FPGA プロトタイプシステム 21
16 IMAX を実装した VU440 のフロアプラン 24
17 マルチチップ構成 25
18 IMAX の 8 チップ接続時 26
19 行列積の IMAX への実装 28
20 畳み込み演算の IMAX への実装 30
21 Light-field 画像距離抽出処理の IMAX への実装 31
22 理想的なダイアグラム 34
23 シングルチップ実行性能 36
24 アプリケーション実行性能 (star) 38
25 アプリケーション実行性能 (tree) 38
26 アプリケーション実行性能 (daisy chain) 39
27 ステート別プロファイリング結果 (star) 39
28 ステート別プロファイリング結果 (tree) 40
29 ステート別プロファイリング結果 (daisy chain) 40
30 ステート別プロファイリング結果 (daisy chain, 最大 8 チップ接続時) 42

List of Tables

1 IMAX 動作シーケンス ステート一覧 12
2 開発環境と使用した Vivado Strategy 22
3 開発に使用した主要な IP コア 22
4 VU440 における FPGA リソース使用量 24
5 評価アプリケーションの概要 27
6 アプリケーションと命令マップ 33
7 評価条件 36

1. はじめに

1.1 背景

近年，計算機の演算性能向上に伴い，Deep Learning 等機械学習アルゴリズムの性能が向上し，画像処理分野をはじめとした，あらゆる分野でブレークスルーを起こしている [1, 2, 3, 4]．この機械学習での活用を目的とし，膨大なセンサが収集した一次元データをクラウドセンターに集約し，学習・推論するシステムが多く発表されている．エッジデバイスには低価格かつ低消費電力組み込み用マイクロプロセッサ，クラウドセンターには高性能かつ多重並列処理が可能である Graphics Processing Unit (GPU) や大規模 Field Programmable Gate Array (FPGA) を搭載することが主流となっている．しかし，ネットワークとクラウドセンターの負荷増大は，応答性と可用性を低下させる．膨大なエッジデバイスが学習・推論の一端を担うためには，エッジデバイスに適合する低価格かつ低電力，高効率計算基盤を開発することが極めて重要であるといえる．ここで，プロセッサの性能向上は，半導体の微細化と並列処理の効率化により支えられてきた．近年では，ムーアの法則の終焉などにより前者に限界が見えており，後者のさらなる進歩に期待がかけられている．現在広く普及している特徴的な計算基盤は 3 種類あり，高性能プロセッサ，低電力メニーコアプロセッサ，GPU である．高性能プロセッサは内臓キャッシュへのデータを閉じ込め，GPU は巨大なレジスタファイルと膨大なマルチスレッドにより 1000 サイクル以上の主記憶遅延隠蔽を可能としている．いずれにしても各コアが要求に応じて主記憶参照の規則性をハードウェアが解析かつ予測して演算器稼働率を最大化している．GPU の消費電力の多さは，従来型プログラマビリティに必要不可欠となっている．一方，前述の通り，機械学習アルゴリズムが開発されるにつれて，エッジデバイスに対する演算性能の要求はますます高まっている．エッジデバイスに期待されるワークロードは，ネットワーク負荷を下げる動画像フィルタや圧縮，クラウドセンターからのフィードバックを用いた動画像認識等である．しかし，演算で多用される複雑な離散ステンシル演算やベクトル長の短い畳み込み演算は，要求に応じて参照順序を規則化するチューニングが非常に困難である．そのため，最新型 GPU の場合でピーク性能の数分の一程度しか出すことができず，要求性能達成のために過大な演算能力と高価な大規模 LSI が必要となっている．そこで，特定のアルゴリズムに特化することを目的とした Domain Specific Architecture (DSA) や FPGA にアクセラレータを実装した例が数多く報告されている．しかし，前者は汎用性に欠け，後者は構造的に動作周波

数が低いといった問題を持っており，エッジデバイスには適さない．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，ノイマン型計算基盤の性能向上に限界が見えている昨今，DSA の低電力と高速性，FPGA の柔軟性，プロセッサのプログラマビリティ，加えてコストダウンのために必要不可欠である汎用性，省面積，拡張性を兼ね備えるエッジデバイス向けの計算基盤が求められている．

1.2 研究目的

従来よりノイマン型アーキテクチャの欠点を補うアクセラレータとして，シストリックアレイが研究されている．このシストリックアレイは Deep Learning において高い電力効率が得られることが報告されている．加えて，Deep Learning などの機械学習だけではなく，VR に必要なステレオ画像生成に応用可能な Light-Field 画像処理やコントラスト調整のためのトーンカーブ画像処理，計算機科学に幅広く用いられるステンシル演算など幅広い演算に対応できるアクセラレータが報告されている．本研究では，従来型シストリックアレイの課題点を確認し，それらの課題を解決することで 1.1 節で述べた特性を兼ね備えたエッジデバイス向けアクセラレータ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．2 章に関連研究を示し，3 章にて従来型のシストリックアレイの課題と研究対象とするシストリックアレイを説明する．4 章に 5 つの提案手法を示す．その後，5 章に FPGA によるプロトタイプシステムと 10 種の評価アプリケーションの説明を行い，6 章にてアプリケーション実行性能評価を行う．最後に 7 章にまとめる．

2. 関連研究

背景でエッジデバイス向けの低コストかつ省電力なアクセラレータが求められているが，従来型のノイマン型計算基盤だけでは対応が困難であることを述べた．そこで，本章ではノイマン型計算基盤以外に現在幅広く用いられている GPU と DSA を説明したのち，現行システムにおける DSA の問題点について説明する．そして，我々が着目しているアーキテクチャであるシストリックアレイについて説明する．

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．対して，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

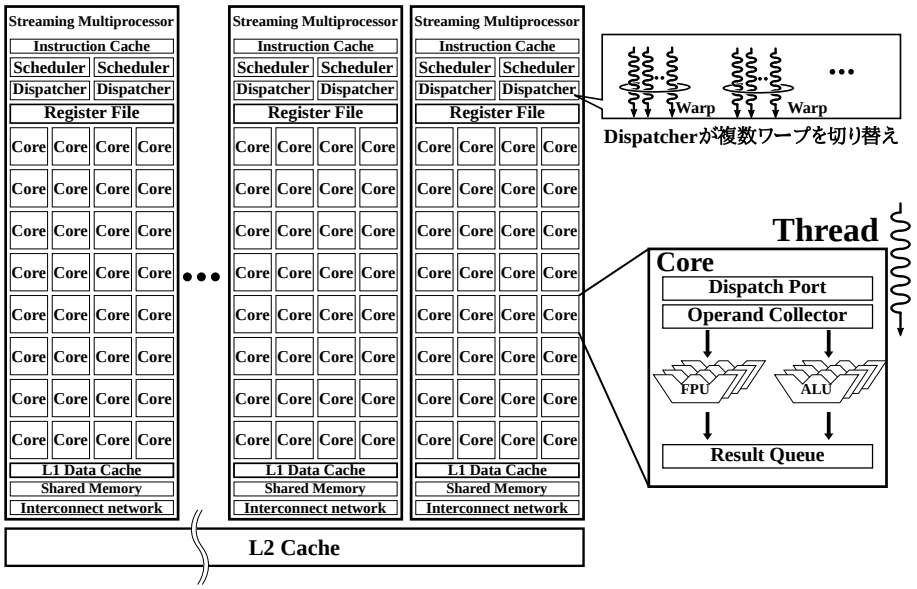

Figure 1: GPU の構造

はこのグループを Streaming Multiprocessor (SM) と呼び，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 Threads (SIMT) 方式と呼んでいる [5]．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 Deep Learning 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や画像処理を低消費電力で高速に行う DSA が数多く提案されている．

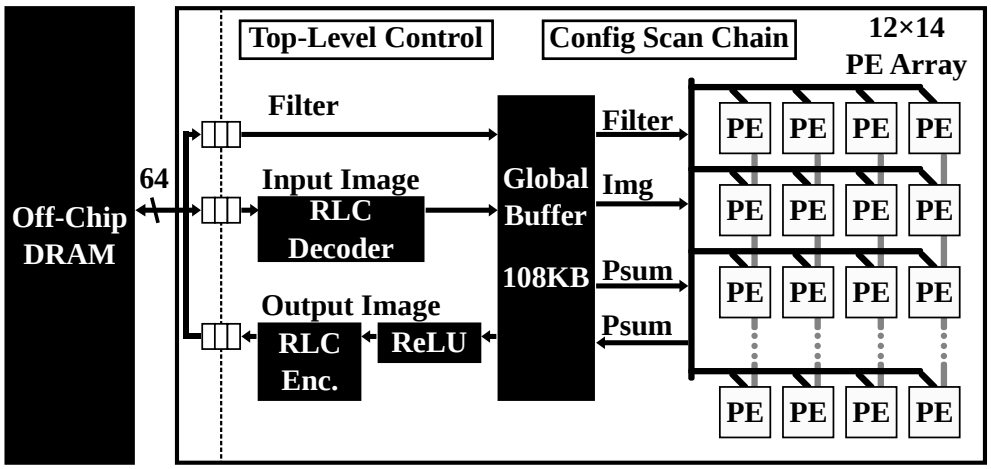

Figure 2: Eyeriss の構造

2.2 DSA

Deep Learning の推論処理やデータベース処理など特定利用のための専用ハードウェア (DSA) を開発し，アプリケーションを高速化する取り組みが流行している．DSA の例として，Eyeriss[6]，Q100-Database Processing Unit[7]，DianNao[8]，FlexFlow[9] などがあげられる．推論処理用の DSA の特徴として，Deep Learning の推論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 [10]，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 [11]．そこで，多くの 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ことにより，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[6] の構造を示す．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，アルゴリズムの変化やその特定アプリケーション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，新しいアルゴリズムの出現とともに ASIC として設計し直すことは開発コストを回収することが難しく，現実的ではない．このような理由から 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 の構造を示す．FPGA は論理要素 (Logic Block)，チップ外周に配置された入出力要素 (I/O Block)，それらを接続する接続要素 (Connection Block, Swicth Block) から成る．隣接する論理要素，入出力要素，接続要素をまとめて論理タイルと呼び，アイランドスタイル FPGA では論理タイルがアレイ状に並んでいる．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 シストリックアレイ

シストリックアレイは 1982 年に H.T.Kung らが提唱した FPGA と同様に再構成可能なアーキテクチャである [12]．GPU より高い消費電力あたりの演算性能を達成する計算基盤として注目されている．シストリックアレイは二次元配置の演算器ネットワークをプログラムの最内ループに合わせて，命令列を固定 (再構成) することで計算する．内部メモリに一時保存したデータと主記憶データを同時に演算器に流し込むことで，汎用プロセッサの演算器の数倍の演算性能を発揮することが可能である．各演算器に命令キャッシュやデコーダーが不要なため単位面積当たりの演算器密度向上が容易である．再構成粒度がゲートより大きい演算器レベル

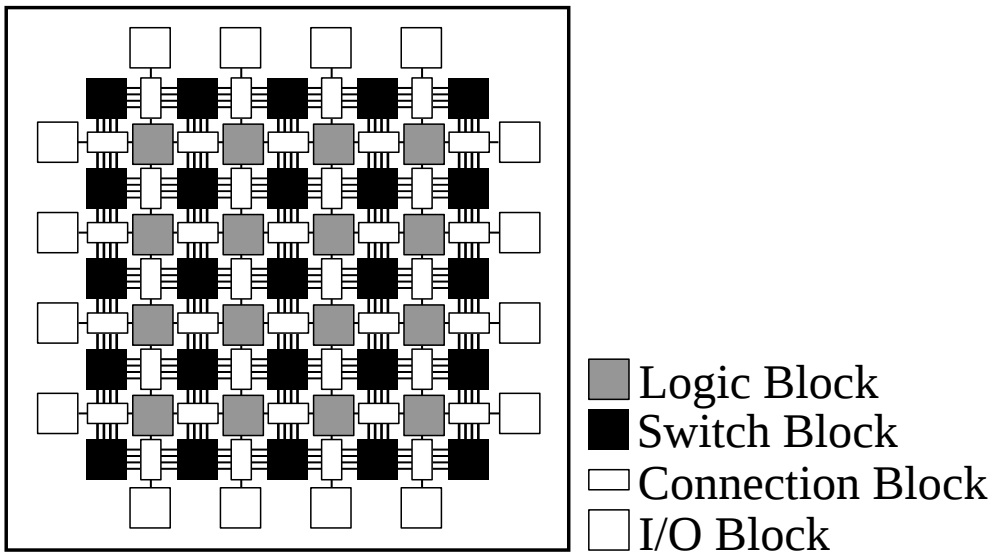

Figure 3: アイランドスタイル FPGA の構造

であるため FPGA より数倍高い動作周波数を実現可能である．また，未使用演算器の長期的電源供給を止めることで省電力化が可能である．

シストリックアレイの例として，Tensor Processing Unit (TPU)[13] や LSSD[14] が挙げられる．図 4 にシストリックアレイの一つである TPU の構造を示す．TPU では大規模な行列演算パイプライン用の演算器をデバイス内に持ち，演算器内で直接データを渡すことでメモリへの読み書きを大幅に削減している．加えて，演算の集約度を高めることで，電力消費を抑えることに成功している．

また，複数のシストリックアレイ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 Computing 社が発表した Dataflow Processing Unit (DPU)[15] が挙げられる．DPU はクラスタ化を前提とした高性能計算基盤向けシストリックアレイであり，64-DPU のクラスタで AlexNet[1] の 90 epochs の学習を 40 分で完了できると報告されている．

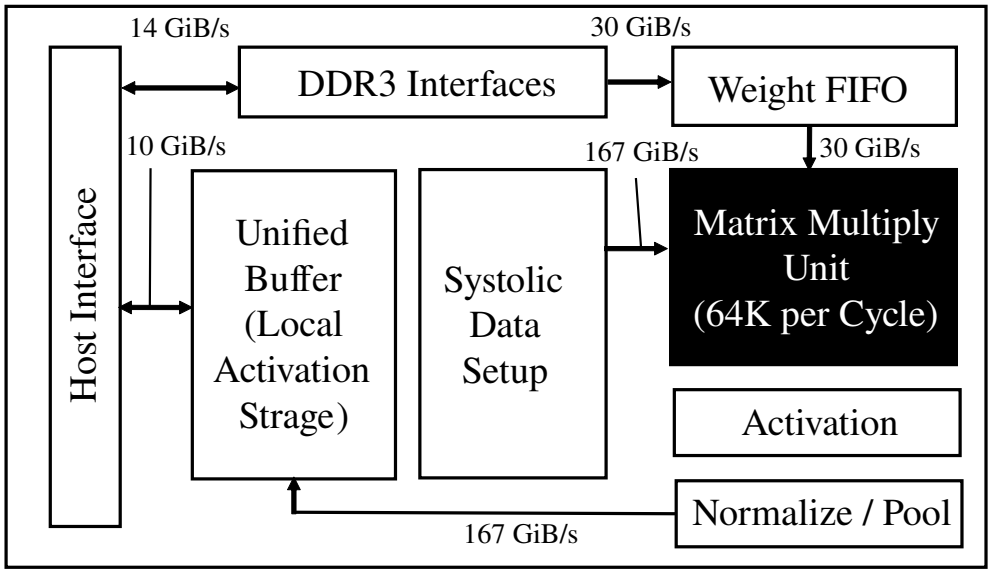

Figure 4: TPU の構造

3. 従来型シストリックアレイの課題とその改善策

前章では現在までに提案されてきたシストリックアレイの特徴について述べ，機械学習アルゴリズムとの親和性が高いことを強調した．しかし，従来型シストリックアレイには本質的欠点があり，そのままではエッジデバイスに適する小型アクセラレータに向かない．そこで，本章では従来型シストリックアレイの課題点を説明し，それを改善したアクセラレータとして我々が提案している IMAX について説明する．

3.1 課題

本節では従来型シストリックアレイの課題について説明する．まず，シストリックアレイの二次元格子をそのまま実装すると長距離配線や配線混雑が動作周波数および面積効率の低下をもたらしてしまう．次に，静的解析が困難なデータ参照パターンを持つ演算において，外部メモリ参照が増加し，ローカルメモリの再利用効率の悪化や読み出し遅延の増大につながる．また，シストリックアレイに写像される最内ループの高速化のみでは，ベクトル長の短い行列積や畳み込み演算において性能を出すことが困難である．浮動小数点アキュムレートなどの本質的に複数サイクルを要する命令を使用しても，パイプライン実行が生じてしまい演算バブルが生じてしまう．最後に，高機能な DMA マスタ機能や外部メモリバスを複数持つことで消費電力の増大を招いてしまう．

まとめると，従来型シストリックアレイの課題点は次の通りである．

課題 A: 二次元格子構造による長距離配線と配線混雑

課題 B: 複雑なデータ参照パターンを持つ演算における外部メモリ参照と読み出し遅延

課題 C: ベクトル長が短い演算における最内ループ実行の高速化

課題 D: パイプライン実行によるバブル

課題 E: 高機能な DMA マスタを内蔵することによる消費電力の増大化

課題 F: 複数の外部メモリバスを持つことによる消費電力の増大化

3.2 IMAX の基本構造

本節では我々が提案しているアクセラレータである IMAX の基本構造とメモリ参照について説明する．IMAX の物理構造は演算ユニットを 1 列 ×64 行配置する構造となっており，実際のプログラミング時の論理構造は 4 列 ×64 行である．図 5 に IMAX の演算ユニットを示す．各ユニットは 3 段パイプラインの 3 入力の Floating Point Unit (FPU) を備えており，64 ビット幅のレジスタを使用して計算する．また，3 段パイプライン演算器の初段は各種マルチメディア演算および固定小数点演算，中段は論理演算，後段はシフト演算を行うことが可能である．

図 6 に命令とメモリアクセスパターンの対応例を示す．各ユニットは 32KB のデュアルポートメモリを備えおり，図 6 に示す様々なメモリ参照パターンの命令に対応可能である．図 6(a) は，1 行のローカルメモリに格納された A，B，C を 4

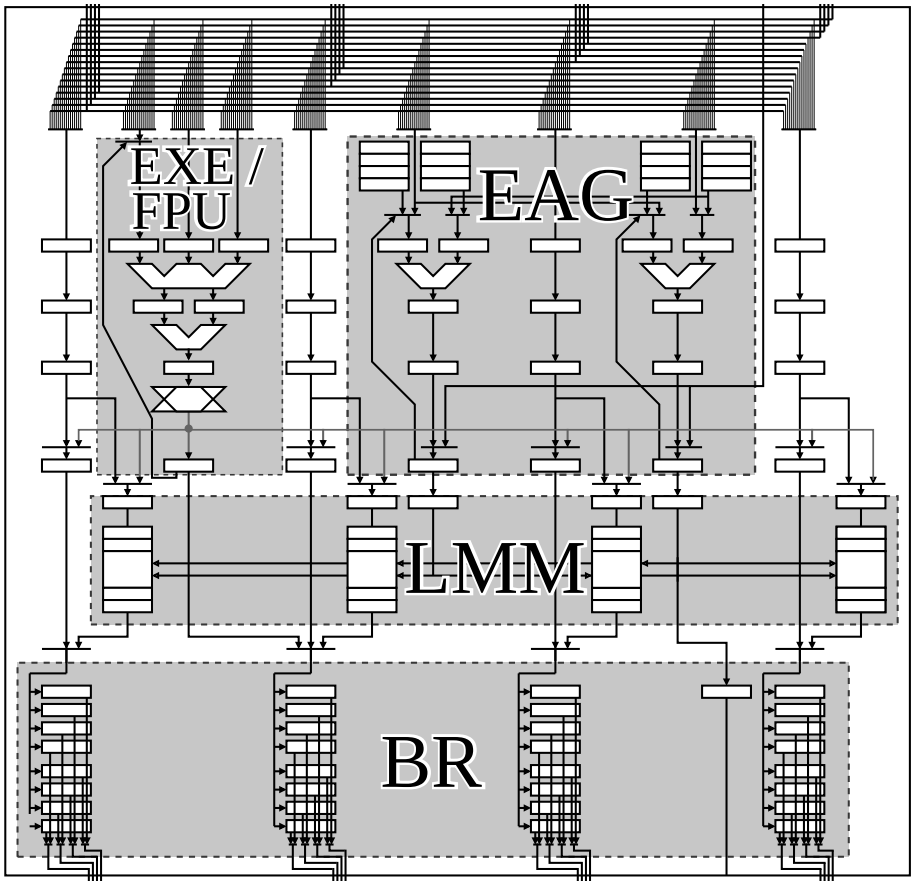

Figure 5: IMAX の演算ユニット

ワードずつ読み出し，4 つの演算器により SIMD 演算を行った後，演算結果 D を次段のローカルメモリに格納する組を 2 セット写像する．A，B，C は外部メモリから供給され，D は外部メモリに取り出される。続いて，図 6(b) は，縦方向に畳

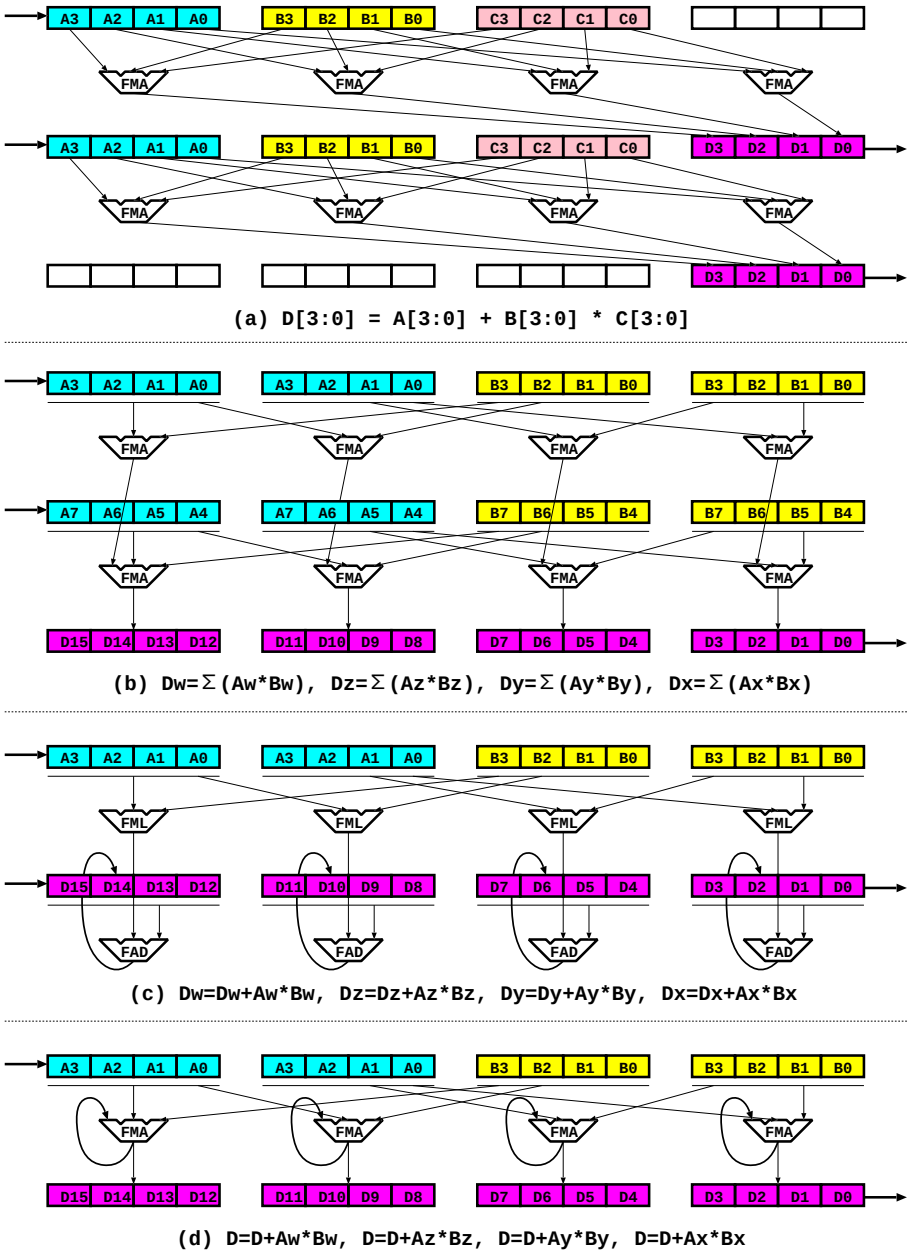

Figure 6: 命令とメモリアクセスパターンの対応例

み込み演算を行う写像である．デュアルポートメモリから各列の演算器に A と B を供給し，畳み込み演算を行う．図 6(c) は，ローカルメモリに D を累算する写像である．一般的なシストリックアレイとは異なり，全体としてはデータを下流へ伝搬しつつ，累算も行うことが可能となっている．図 6(d) は，同様にローカルメモリに D を累算する写像である．ただしこの場合，格納先が移動し，ローカルメモリの Read-Modify-Write により，何度もローカルメモリ内のデータを更新をすることを示している．

3.3 IMAX の動作シーケンス

本節では IMAX の動作シーケンスについて説明する．まず，IMAX における動作シーケンスを図 7 に示し，各ステートの動作を表 1 に示す．ホストとなる CPU 側が Master となり，IMAX 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 で演算を実行する．DRAIN，LOAD には DMA 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．

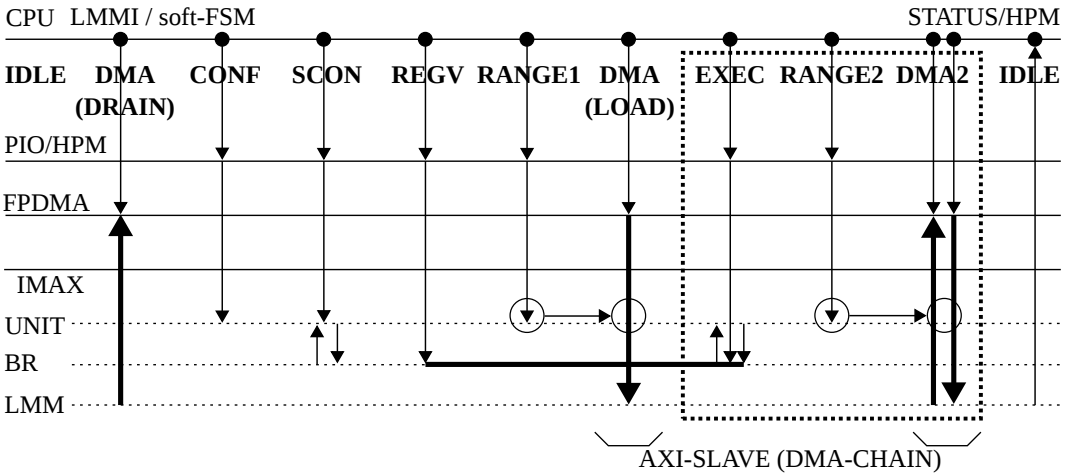

Figure 7: IMAX の動作シーケンス

Table 1: IMAX 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	ローカルメモリ情報設定	PIO
DMA(DRAIN)	DMA 転送 (ローカルメモリ-主記憶)	DMA
DMA(LOAD)	DMA 転送 (主記憶-ローカルメモリ)	DMA
EXEC	演算実行	PIO
IDLE	アイドル状態	PIO

4. 提案手法

3.1 節では，従来型シストリックアレイの課題と欠点について説明した．そこで本章では，従来型シストリックアレイの課題を解決すべく，改善のための手法を提案する．

4.1 列マルチスレッディング

課題 A と B を解決するために，列方向マルチスレッディングとチップ内バス並列化を提案する．チップ内バス並列化については後述する．図 8(a) に列マルチスレッディング導入前の一般的なユニット構成，図 8(b) に導入後の IMAX を示す．従来型シストリックアレイは，演算結果の同期コストを削減するために各演算器が毎サイクル演算結果を出力する前提で設計される．このため，浮動小数点アキュムレートなどの演算時間が長い命令をパイプライン実行することができずに，面積効率が悪化する．また，ローカルメモリから多量の読み出しを行うために，同一内容を保持する複数のローカルメモリを配置して複数ポート化する必要がある．つまり，広帯域なバスを持つローカルメモリであれば，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，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である．そこで，図 8(b) のように 1 つのユニットが 4 サイクルを使用して，論理的に図 8(a) と同じ機能を実現することで

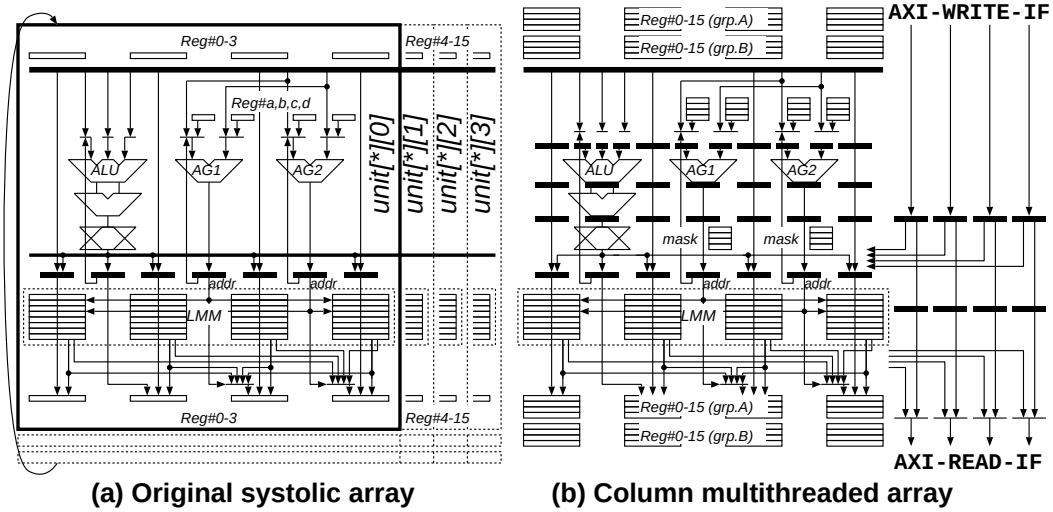

Figure 8: 一般的シストリックアレイと IMAX のマルチスレッディング

改善することができる．演算器を 4 段パイプライン化し，マルチスレッディングを行うことで，1 つの演算ブロックで 4 つの演算ブロックが行っていた演算を行うため，ローカルメモリを備えた演算ブロックを 4 分の 1 に削減することができる．ここで，図 8(a) では 1 つのユニットにおける演算遅延が 2 サイクルであるのに対して，図 8(b) では 8 サイクルである．

4.2 チップ内バス並列化

図 9 に 8 行 ×8 列のチップ内バス並列化を示す．シストリックアレイではチップ内バスが 1 列に接続されている場合，データの読み出し遅延が発生する．例えば，我々が提案する IMAX の場合，2N (64 stages 構成で 128 サイクル) の遅延が発生する．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ローカルメモリからの読み出し遅延を 1/8 に抑える．

4.3 多重ループ一括実行機構

課題 C と D に対応するために，多重ループ一括実行機構とアダプティブ命令シフトを提案する．アダプティブ命令シフトについては後述する．シストリックアレイは多重ループなどの連続実行起動前に命令写像とレジスタ初期化を必要とするため，できる限り連続実行時間を長くし，起動時オーバーヘッドを削減する必要がある．しかし，連続実行時間はローカルメモリに格納可能なデータ量により決定されるため，ローカルメモリの容量にはパイプライン演算器の動作周波数を落とさないようにするための上限が存在する．例えば，画像処理においては入力画像が格納されるデータとなるため，比較的ローカルメモリの容量を使い切ることが可能である．一方で，行列積や畳み込み演算などの場合では連続実行時間が短くなり，ローカルメモリの容量を使い切ることが難しい．そこで，多重ループを一括実行することで起動回数を削減し，連続実行時間を長くする．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機構を適用した IMAX の演算ユニットを図 10 に示す．

図 11 に IMAX の多重ループ実行機構およびマルチチップ実行 (後述) に対応可能な C 言語記述を示す．同一プログラムから，CPU 向けの通常 C コンパイラでも同一の結果を出力するバイナリを生成できるような記述をしている．//EMAX5A に始まる 3 重ループ記述が，IMAX の 1 回の起動においてシステム全体に写像さ

れる機能である．最外ループがマルチチップ，中間および最内ループが各チップにおける多重ループ一括実行機構に対応づけられる．多重ループ一括実行制御においては，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る．ここで，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．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

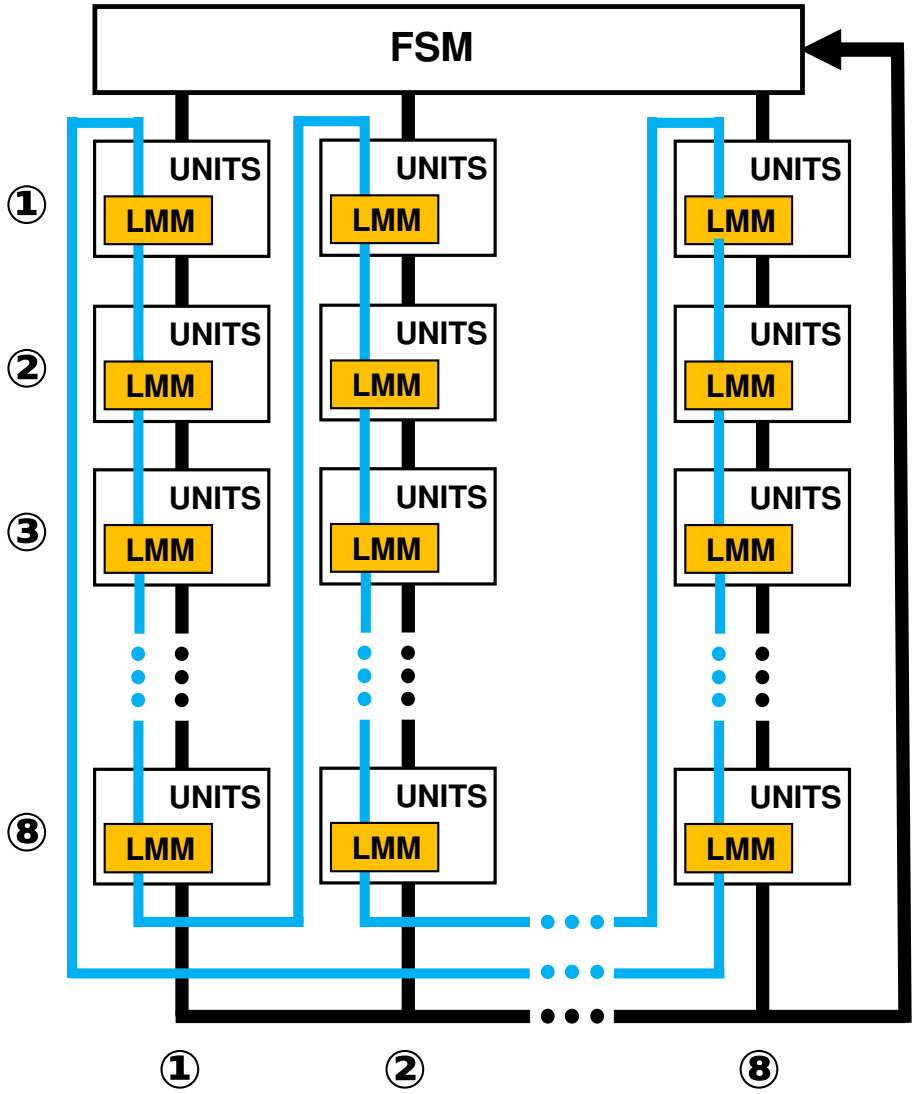

Figure 9: チップ内バス並列化

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．図 10 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することで，これを実現している．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．

複数制御変数に対応可能なアドレス計算機構については説明したが，従来型シストリックアレイでは，演算結果やローカルメモリ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，最内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う必要がある．特に，IMAX は DMA Slave デバイスであるため，主記憶へ

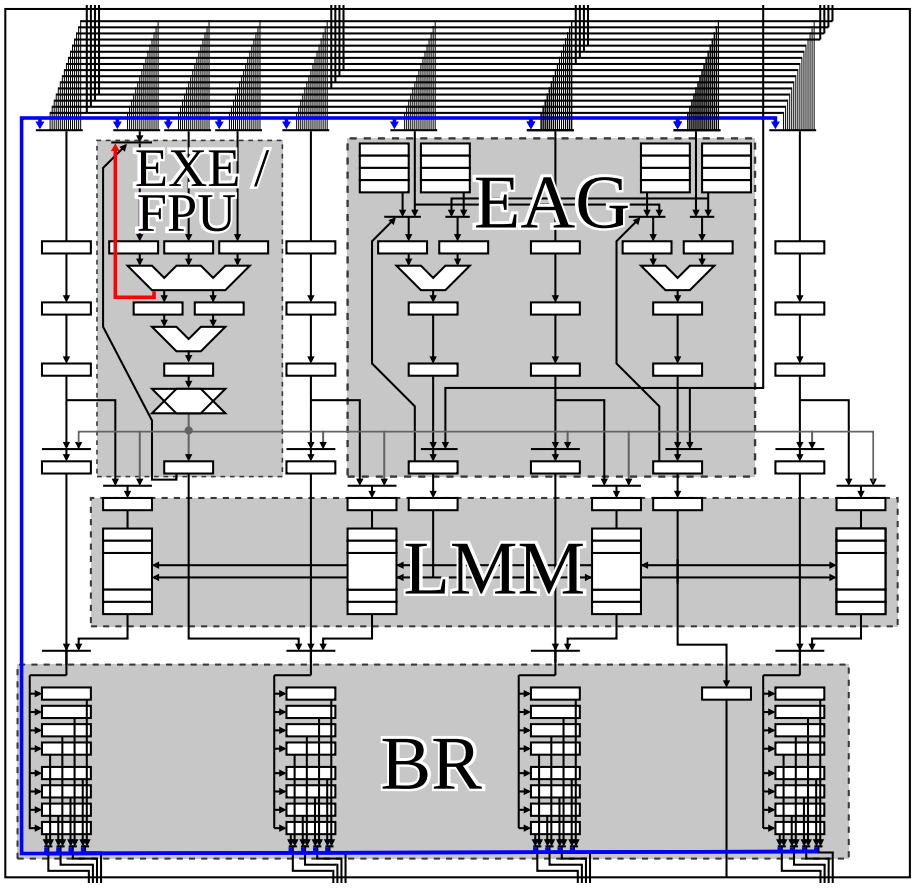

Figure 10: 多重ループ実行機構を適用した IMAX の演算ユニット

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．一方，本研究では，複数制御変数に対応可能なアドレス計算機構により多重ループを一括実行しつつ，最終段のローカルメモリに保存される最内ループの実行結果を一旦 DDR に追い出すことなく Read-Modify-Write 機能により累算することで，これを実現する．Read-Modify-Write 機能はユニットの出力を同一ユニットの入力に戻す，図 10 にて青色で示したフィードバックパスを追加することにより，実装される．図 12 にユニットとの対応を示す．論理的に 4 列分の Read-Modify-write が物理的には単一のユニットに写像される．前段の演算器による累算結果 (sum0-3) が一旦ユニット下端のレジスタに送られ，ロード結果 (o0-3) とタイミングを合わせて演算器入力へフィードバックされ，加算結果 (s0-3) がローカルメモリにストアされる．

4.4 カスケード接続機構

課題 E と F を解決するために，メモリインターフェースを新たに増設する必要がないマルチチップカスケード接続機構を提案する．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[16] 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

```
//EMAX5A begin x1 mapdist=0
 /*3*/ for (CHIP=0; CHIP<NCHIP; CHIP++) {
  /*2*/ for (INIT1=1,LOOP1=RMGRP,rofs=0-M*4; LOOP1--; INIT1=0)
   /*1*/ for (INIT0=1,LOOP0=M-2,cofs=(0-4)&0x00000000ffffffffLL; LOOP0--; INIT0=0) {
       exe(OP_ADD, &cofs, INIT0?cofs:cofs, EXP_H3210, 4, ... );/* stage#0 */
       exe(OP_ADD, &rofs, rofs, EXP_H3210, INIT0?M*4:0, ... ); /* stage#0 */
       exe(OP_ADD, &oofs, rofs, EXP_H3210, cofs, ... ); /* stage#1 */
       mop(OP_LDUWR, 1, &BR[2][0][1], (Ull)kp00[CHIP], 0LL, ... );/* stage#2 */
       mop(OP_LDUWR, 1, &BR[2][0][0], (Ull)kp01[CHIP], 0LL, ... );/* stage#2 */
       mop(OP_LDUWR, 1, &BR[2][1][1], (Ull)kp02[CHIP], 0LL, ... );/* stage#2 */
       mop(OP_LDUWR, 1, &BR[2][1][0], (Ull)kp03[CHIP], 0LL, ... );/* stage#2 */
       mop(OP_LDUWR, 1, &BR[2][2][1], (Ull)ip00, oofs, ... );
```

Figure 11: For 文によるマルチチップおよび多重ループ制御

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．ここで，マルチチップ構成によりコストダウンが図れるのは，単一チップの面積を小さくすることにより歩留まりが改善されるためである．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．

IMAX では，ARMv8 をホストとし，標準の AXI バス上のメモリデバイスとして任意の演算資源を ARM の主記憶空間上に写像可能である．この時，1 つの空間に

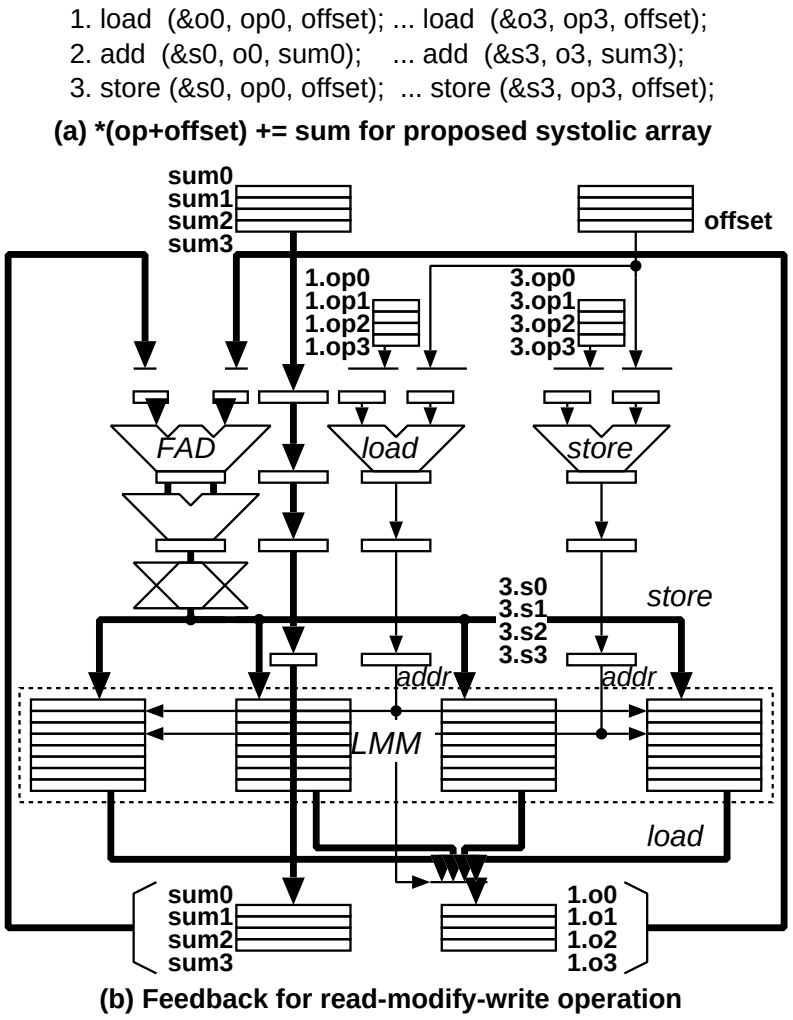

Figure 12: Read-Modify-Write 機能

複数のローカルメモリを写像し，ホストからの PIO/DMA により多数のローカルメモリに画像やパラメータを一度に書き込むことも可能となっている．C 言語によりアプリケーションを記述する場合，各データ構造をどのローカルメモリに対応付けるかを指定し，演算を配置することで任意のデータフローを構成する．ここで，複数チップにまたがって多数のローカルメモリへブロードキャスト PIO/DMA ができ，演算結果の回収も複数チップにまたがって PIO/DMA 転送することで，アプリケーションをスケーラブルに高速化可能である．

図 13 に，2 チップ構成の場合のユニット間接続方法を示す．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 個のユニットを内蔵する各チップは，演算用に 1 列構成のリング接続（前述のように各ユニット通過時間は 8 サイクル），DDR とローカルメモリ間転送用に 8 行 ×8 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 とローカルメモリ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 64 ユニット通過時間の大幅削減を狙うためである．ここで，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．

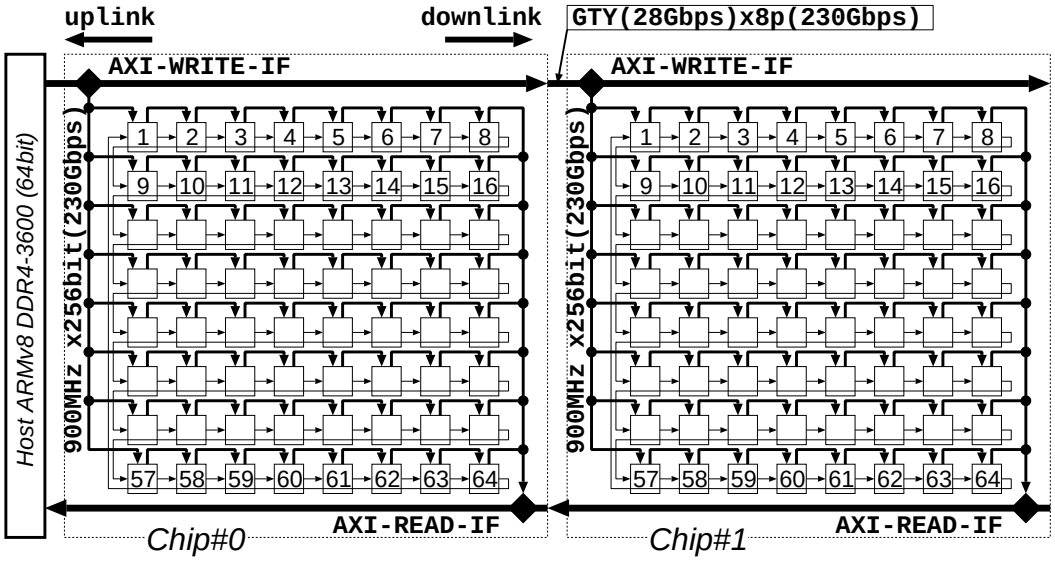

Figure 13: IMAX マルチチップ構成

4.5 アダプティブ命令シフト

再利用可能なローカルメモリに合わせて命令写像を下方にシフトし，ローカルメモリへの転送を最小化できる機能について提案されている [17]．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，現在のローカルメモリ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. FPGA プロトタイプシステムと評価アプリケーションの準備

本章では，IMAX の FPGA プロトタイプシステムについて説明する．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．FPGA SoC と大規模 FPGA を用いた ARMv8+IMAX

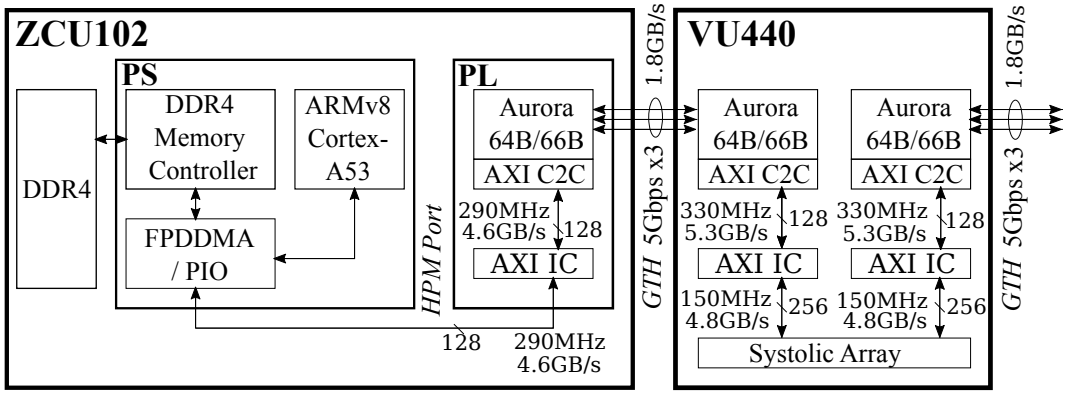

Figure 14: ARMv8 + IMAX FPGA プロトタイプシステムのブロック図

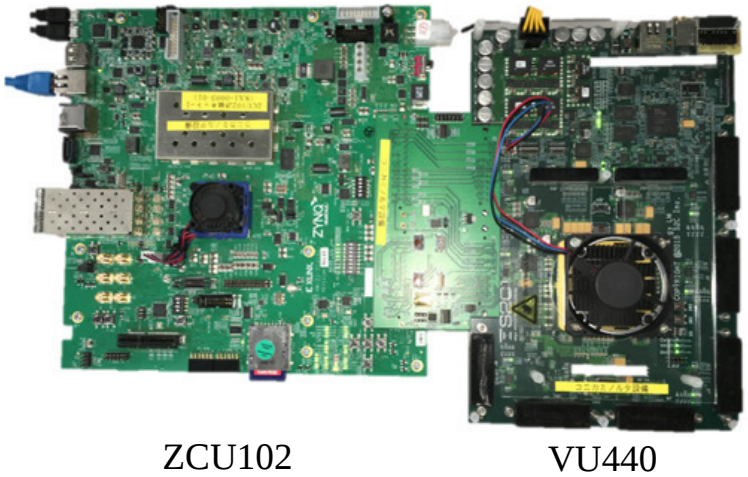

Figure 15: ARMv8 + IMAX FPGA プロトタイプシステム

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 14 および，図 15 に示す．図 14 では，矢印の始点が Master，終点が Slave としている．ここで，Systolic Array は IMAX を指している．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．

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 2 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 3 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．

Table 2: 開発環境と使用した Vivado Strategy

ワークステーション	Intel Core-i9 7980XE DDR4-2400 128GB CentOS 7.6
FPGA 開発環境	Vivado System Edition 2018.3
Synthesis Strategy	Flow_PerfOptimized_high
Implementation Strategy	Performance_ExplorePostPhysOpt

Table 3: 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

5.1 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，330MHz 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 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 330MHz 動作なので 5.3GB/s，AXI IC - IMAX 間が 256bit 接続 150MHz 動作なので 4.8GB/s である．また，マルチチップ接続に対応づけられる VU440 間に関しても ZCU102 側と同様に AXI IC と C2C，そして Aurora64B/66B を用いた GTH の 3 レーンを介して接続される．ここで，VU440 間も理論帯域幅は同様であるが，Master と Slave は接続の度に交互に入れ替わる．

VU440 に IMAX を実装した際のリソースの使用率を表 4 に示す．IMAX は使用可能リソースに対して，LUT が 61.9%，F/F が 9.1%であった．IMAX を実装した際のフロアプランを図 16 に示す．図 16 では，8 × 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．

5.3 マルチチップ構成

図 17 にホストである ZCU102 と IMAX (VU440) を接続する 3 種のマルチチップ構成，図 18 に IMAX の 8 チップ接続時の画像を示す．図 17 について，4 チップ接続時以外の構成においては，Z を ZCU102，V を IMAX として簡略化して表記している．また，デイジーチェインについては最大 8 チップ接続時の評価を行うため，

5 から 8 チップ接続時の構成も載せている．接続リンク数が増加すると，ホストとシストリックアレイ間の通信コストが増大する．スター型は，ホスト上に多数の接

Table 4: VU440 における FPGA リソース使用量

	IMAX	VU440 利用可能リソース
Fmax	150MHz	N/A
LUT	1567474	2532960
LUTRAM	676	459360
F/F	457572	5065920
BRAM	522	2520
DSP	256	2880

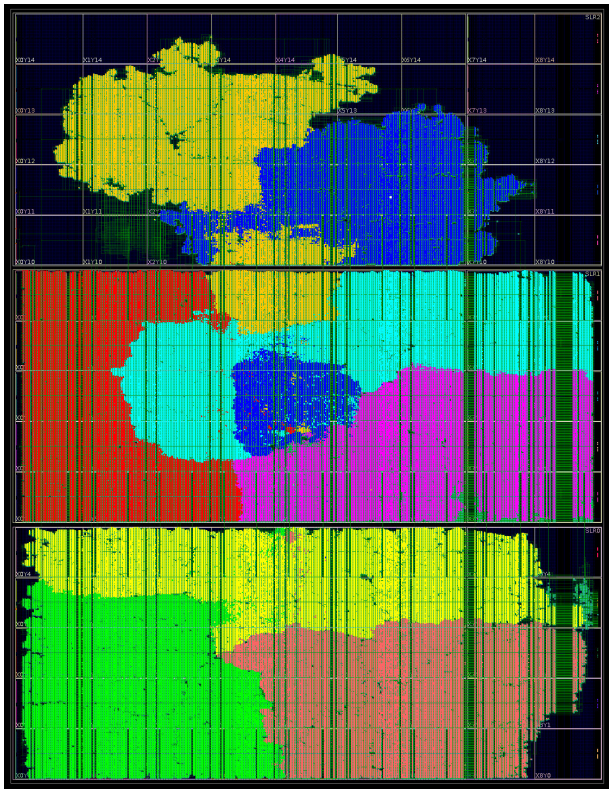

Figure 16: IMAX を実装した VU440 のフロアプラン

続リンクを必要とするが，接続リンクの深さが最小の構成である．次に，ツリー型は，スター型に比べてホスト上の接続リンク数を大幅に節約する構成である．最後に，デイジーチェイン型はホスト上の接続リンク数を最小とし，各ノードに2つの接続リンクのみを持たせた構成で VU440 同士を接続させることが可能である．一方で，接続リンクの速度が重大なボトルネックである場合，スター型構成は最も高い実行性能を得ることが可能である．対して，そうではない場合，ツリー型構成やデイジーチェイン型においても安定した性能を得ることが可能である．

ここで，プロトタイプシステム上においてマルチチップ実行を試験運用した結果，先頭FPGA ボードと2枚目の FPGA ボードとの接続では，先頭 FPGA は複数のバースト送信を連続発行するが，送受信間に長い idle 時間が存在していた．そこで，AXI プロトコルの送受信確認信号 (bvalid/bready) のハンドシェイクを使用しないことでパイプライン送信を可能とし，改善した．また，4枚以上 FPGA ボー

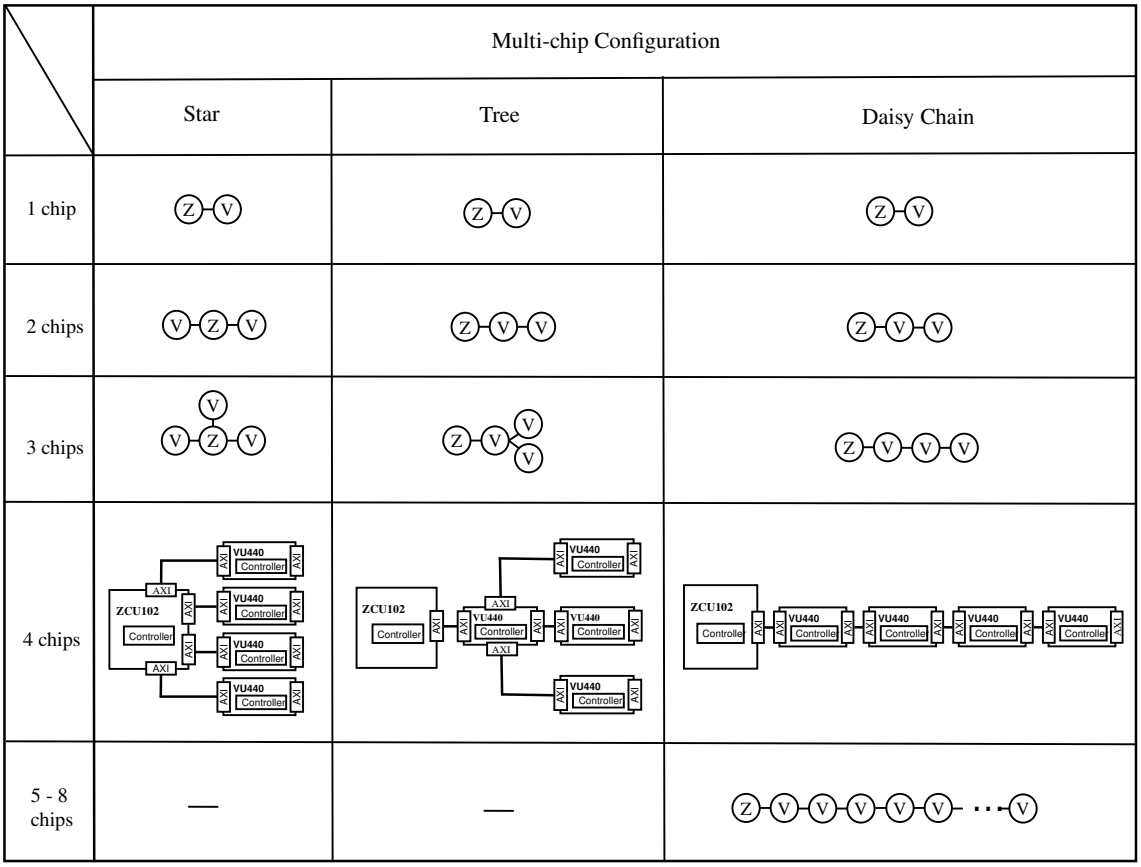

Figure 17: マルチチップ構成

ドを接続した場合，ローカルメモリからの読み出しに使用する AXI-READ が極めて低速であることが判明した．これは，ホストの DMA 機能に対して十分な長さの読み出すバースト長を指定した場合においても，ホストに内蔵されている AXI インターフェースが AXI3 互換であるため，256bit 幅転送の場合，8beat 毎に転送が分割され全チップからの応答が完了するまで次の AXI-READ が待機させられることが問題であった．そこで，ホストの AXI インターフェースは AXI3 のまま使用し，IMAX のチップ間では分割せず内部で元のバースト長を維持することで対応した．

5.4 アプリケーションプログラムの準備

本節では，評価対象とする 10 種のアプリケーションについて説明する．その後，10 種のアプリケーションから機械学習と画像処理で用いられる 3 種のアプリケー

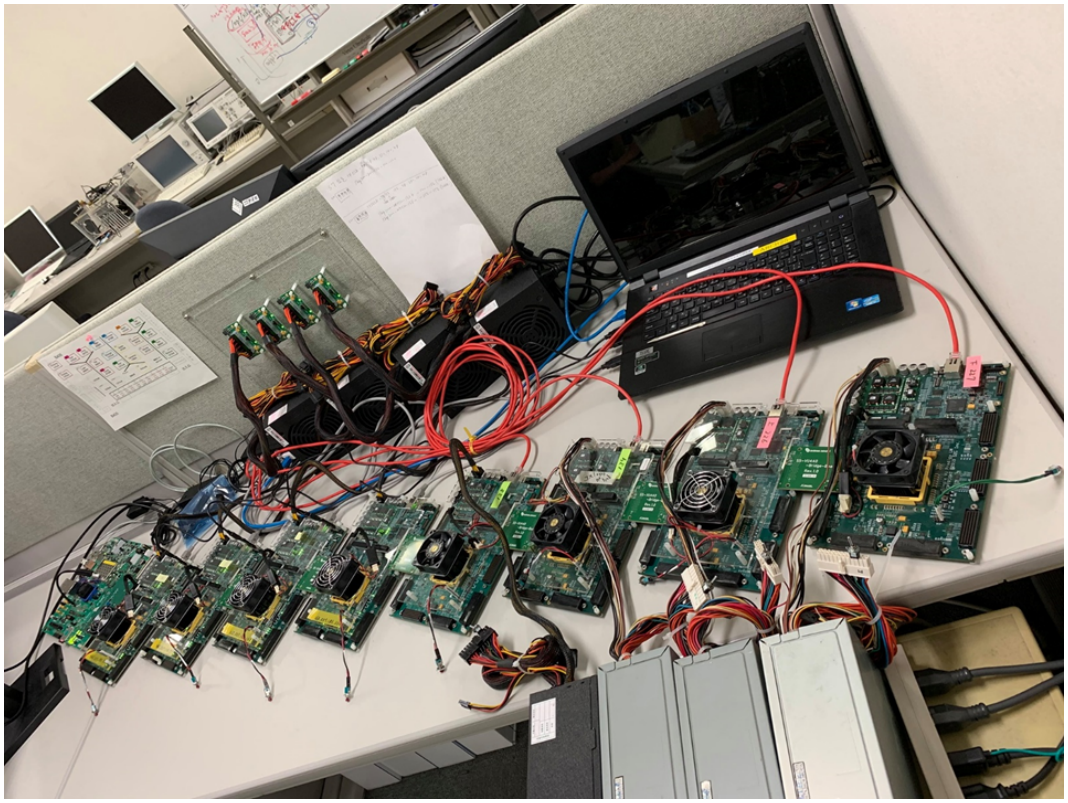

Figure 18: IMAX の 8 チップ接続時

Table 5: 評価アプリケーションの概要

アプリケーション	説明	データサイズ
mm	行列積	480×480
cnn	畳み込み演算	242×242,IC=18,OC=16
gather	Light-Field 画像のレンダリング	7500×1600
gdepth	Light-Field 画像からの深度抽出	7500×1600
grapes	3 次元 19 点ステンシル演算 (degree=1)	320×240×120
jacobi	3 次元 7 点ステンシル演算 (degree=1)	320×240×120
fd6	3 次元 19 点ステンシル演算 (degree=3)	320×240×120
resid	3 次元 27 点ステンシル演算 (degree=1)	320×240×120
wave2d	2 次元 5 点ステンシル演算 (degree=1)	320×240
tone curve	コントラスト調整	320×240

ションについて IMAX へのマッピング例を取り上げる．3 種のアプリケーションは Deep Learning 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 Light-Field 画像からの深度抽出 (gdepth) である．

5.4.1 評価アプリケーション

表 5 に評価対象とする 10 種のアプリケーションの概要を示す．評価アプリケーションには，Deep Learning や画像処理などで頻繁に使用される演算に加えて，Light-Field 画像処理のような複雑なメモリアクセスシーケンスを持つ演算などを選択した．これは，アクセラレータのプログラマビリティや汎用性を実証する目的がある．その中でも，mm と cnn の入力は簡単に再利用でき，物理リンク速度の影響が軽減されることが予想される．Light-Field 画像処理 (gather, gdepth) においては，不規則なアドレス読み出しにより再利用シーケンスが予測できず，ローカルメモリからの読み出し時間の影響が大きくなる．ステンシル演算 (grapes, jacobi, fd6, resid, wave2d) の入力データはローカルメモリ内での再利用効率が高いが，単純に空間分割して演算するため，マルチチップ化の恩恵を受けにくい．コントラスト調整 (tone curve) は基本的な画像処理演算である．ただし，従来型キャッシュシステムにおいて，参照テーブルを参照するための間接メモリアクセスを使用するため，キャッシュラインが不明瞭になる．しかし，IMAX のシストリック構造では，入力画像は再利用することが不可能ではあるが，LUT をローカルメモリに保存して再利用することが可能となっている．

5.4.2 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 Deep Learning の全結合層も行列積で実装される．

図 19(a) に mm のデータ構造，図 19(b) に C 言語による一般的な実装，図 19(c) に各 unit に積和演算器を 1 つ備える H+1 (高さ)× W (幅) 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][*] × 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，H×W 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 × B0c，A01 × B18, A02 × B24, A03 × 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ローカルメモリに累算する．すなわち，H×W 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．図 19(d) はさらに，配列 A を N チップに行分割して並列処理し，各ローカルメモリが，配列 A の 1 行（M 要素）× GRP 行分と，配列 B の 1 行（M 要素）を収容

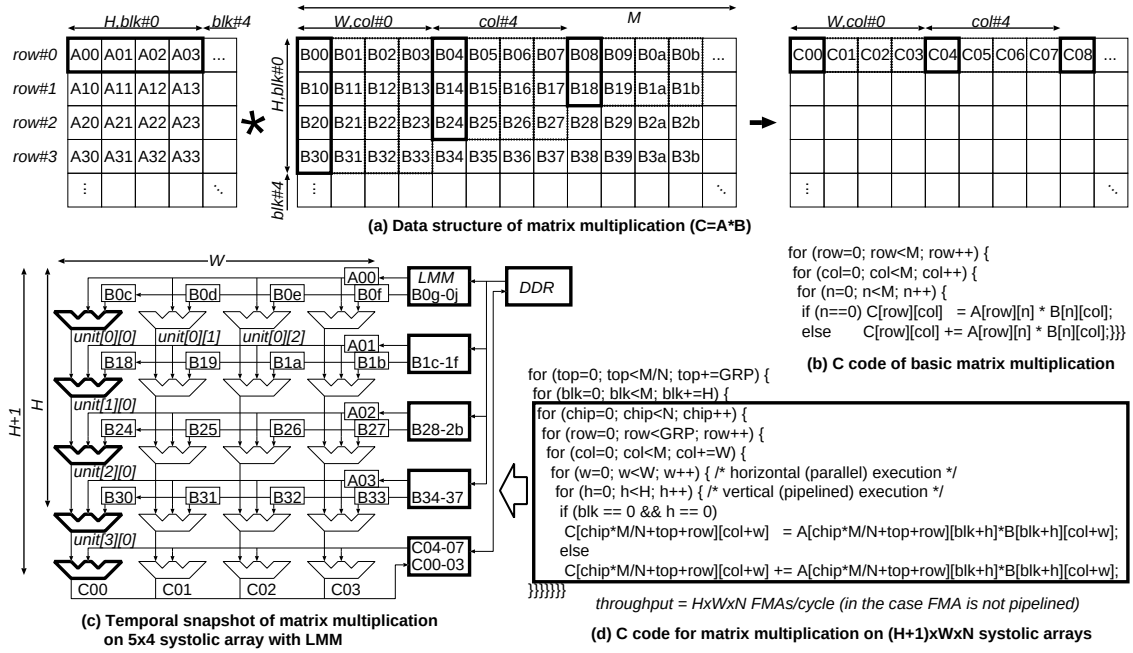

Figure 19: 行列積の IMAX への実装

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ローカルメモリは配列 A を GRP 行分保持し，最終段のローカルメモリは C×0, C×1, ..., C×(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）およびローカルメモリの入れ換え時間を除くと $M^3/(H\times W\times N)$ となる．また，ローカルメモリが A，B，C 全体を収容できる場合の理論的 DDR とローカルメモリ間転送量が $M^2\times 3$ であるのに対し，各ローカルメモリが A と C を M×GRP，B を M×H のみ収容できる場合の転送量は，A と C が各々 M^2，B が $M^2\times M/GRP$（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．

5.4.3 畳み込み演算

図 20(a) に，cnn のデータ構造を示す．M，K，IC，OC，N，W，GRP はそれぞれ，マトリックスのサイズ (各入力および出力チャネル)，カーネルサイズ，入力チャネルの数 (入力画像の数など)，出力チャネルの数，チップ数，演算器の論理幅，およびシングルバースト実行で処理されるグループの行数を示している．また，図 20(b) に各 unit に積和演算器を 1 つ備える $(K^2+1)\times W$ 構成の IMAX を示す（M≥K, OC≥W を仮定）．図 20(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．cnn は，入力 in の一部（K×K）と kernel（K×K）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 $M^2\times IC$，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 $K^2\times IC\times OC$，出力の総ワード数は $M^2\times OC$ 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．K×K 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 $K^2\times W$ の全演算器を利用でき効率がよい．このとき，K 段のローカルメモリに対する列方向ブロードキャスト機能を用いる．一方，最終段のローカルメモリは oc を更新する度に入れ替える必要がある．cnn の演算に要する理論的サイクル数は，IMAX 起動時の遅延（$K^2\times$REP サイクル）およびローカルメモリの入れ換え時間を除くと，$K^2\times(M\text{-}2)^2\times IC\times OC/(K^2\times REP\times W\times N) = (M\text{-}2)^2\times IC\times OC/(REP\times W\times N)$ とな

る．また，ローカルメモリが in，kernel，out 全体を収容できる場合の理論的 DDR とローカルメモリ間転送量が $M^2 \times IC + K^2 \times IC \times OC + M^2 \times OC$ であるのに対し，各ローカルメモリが in を $M \times GRP$，out を $M \times GRP \times W$，kernel のみ全部収容できる場合の転送量は，in と kernel が各々$M^2 \times IC$ と $K^2 \times IC \times OC$，out は 1 回分の入れ換え量が $(M \times GRP \times W) \times 2$，データの回転数 $(OC/N/W) \times (IC/REP) \times (M/GRP)$ に N を乗じた回数の入れ替えが発生するため，$M^2 \times 2 \times OC \times IC/REP$（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 $M^2 \times OC$．N に非依存）となる．図 20(c) に，時間的なメモリアクセスを示す．t は時間が進む方向を示している．また，入力層から出力層への矢印は，畳み込み演算の方向を示している．

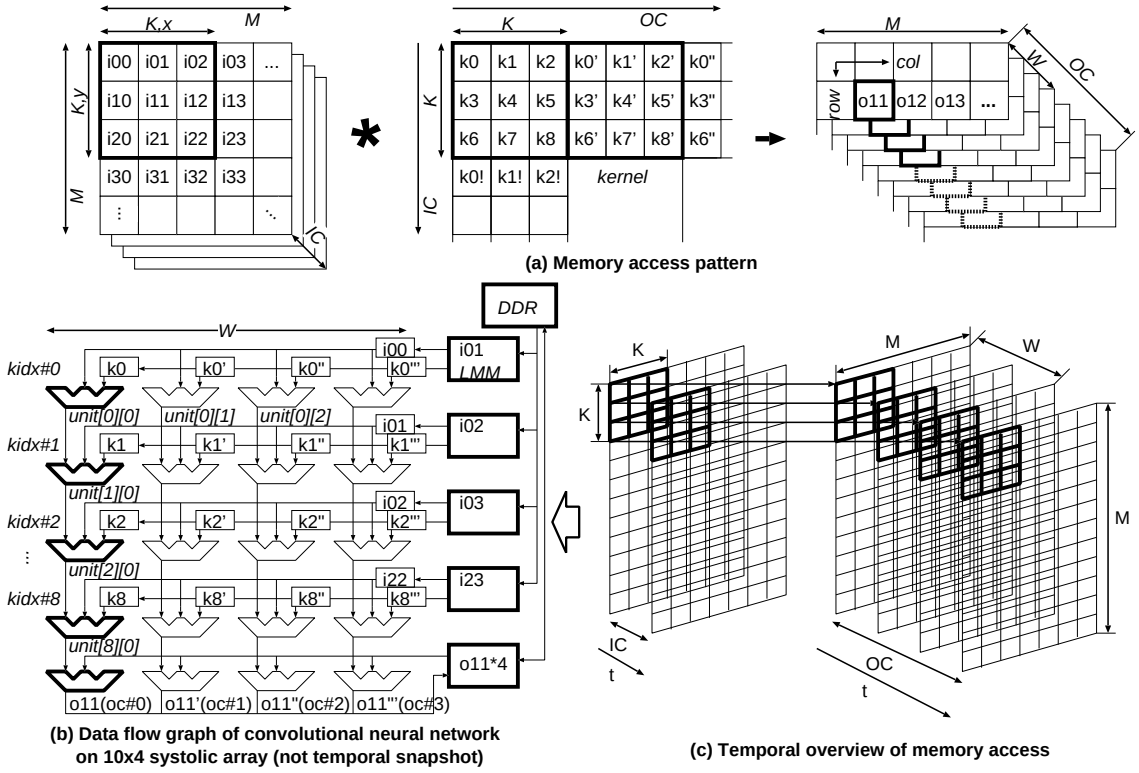

Figure 20: 畳み込み演算の IMAX への実装

5.4.4 Light-field 画像処理

図 21(a) に，Light-Field 画像処理 (gdepth) のデータ構造を示す．IM，OM，RM，N，および W は，それぞれ入力画像のサイズ，出力画像のサイズ，マイクロ画像の距離，チップ数，演算器アレイの論理幅を示している．入力画像には 7500×7500 ピクセルあり，マイクロ画像 (75×75 ピクセル) は 100×100 の 2D メッシュ形式で配置されている．出力画像は，6 つの 3×3 ピクセルのマイクロ画像間で距離 (75+

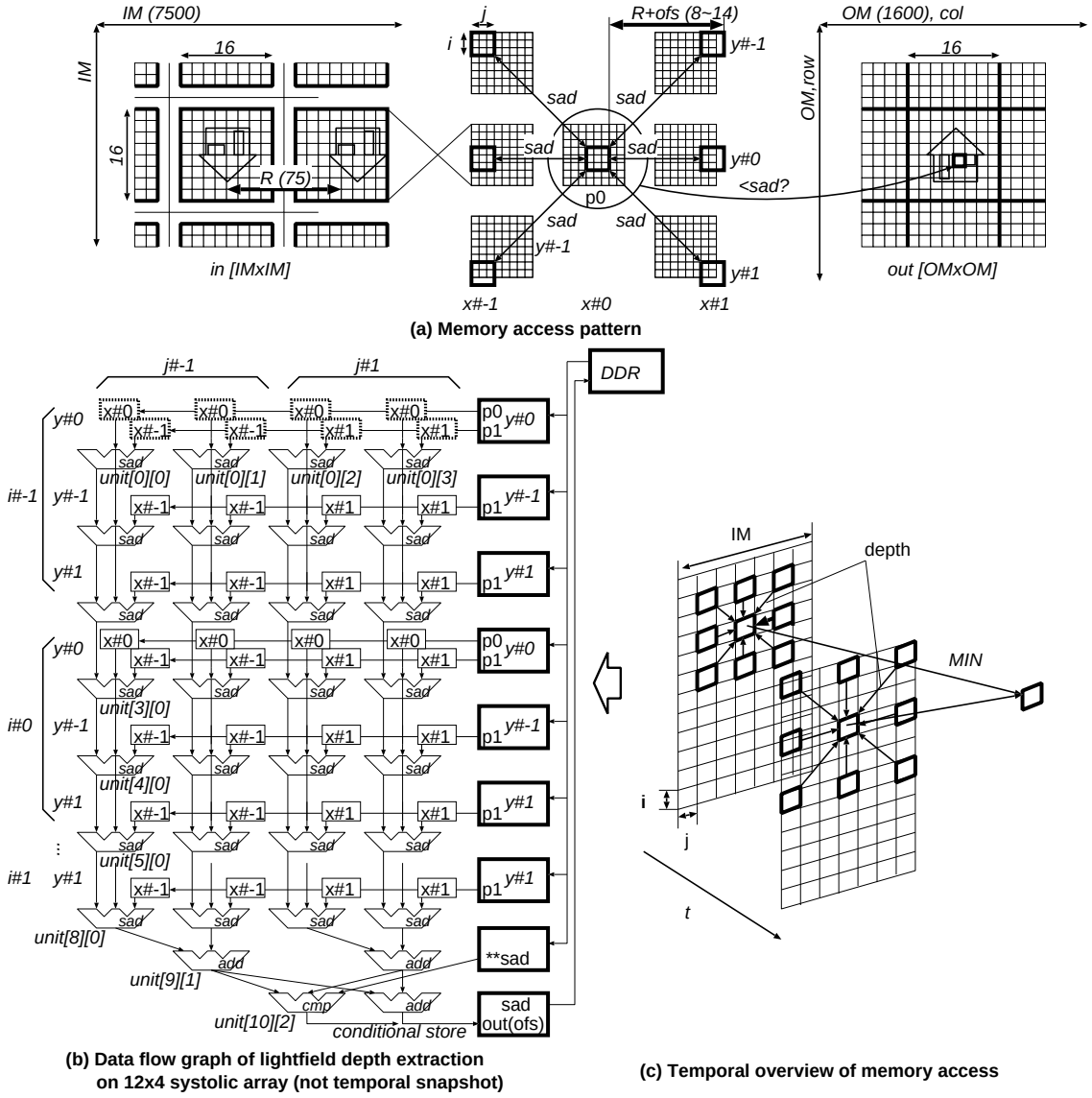

Figure 21: Light-field 画像距離抽出処理の IMAX への実装

深度ピクセル) を変えながら，離散ステンシルパターンの絶対差の最小和 (SAD) を見つけることによって生成された 1600×1600 ピクセルを持つ．SAD は，中央の 6 つのピクセル (p0) と周囲の 6 つの離散位置 (p1) の 36 の差の合計から取得される．図 21(b) に，12×4 ユニットを使用する演算器アレイのデータフローグラフを示す．各ユニットの ALU は SAD 機能を持つ．y#0 位置の 3×3 ピクセル (i#-1) の一番上の行は，最初のステージ (unit[0][*]) にマッピングされ，p0 と p1 を含む 6 つの離散ピクセルは，最上段のローカルメモリからロードされ，入力画像のラインを保持する．2 番目のステージ (unit[1][*]) および 3 番目のステージ (unit[2][*]) は y#-1 および y#1 の位置 (p1) からロードされ，最初のステージで取得した p0 と比較される．演算器アレイの 9 つのステージを通過後，4 列から得られた 4 つの SAD 結果は，2 つの追加ステージを介して，蓄積される．蓄積されている SAD 値よりも，さらに小さな SAD 値を得られた場合には，条件付きストアで保存および更新される．このようにして，36 ピクセルの SAD を取得するために 11 ステージが使用され，44 ユニットのうち 40 ユニットがアクティブになり，ステージ数は 18 となる．図 21(c) に，時間的なメモリアクセスパターンを示す．t は時間の方向を示す．深さは時間とともに増加し，各ウィンドウの SAD が計算され，最小の SAD 値と対応する深さのペアが最終結果として保存される．

gdepth では，DDR とローカルメモリ間の伝送を削減するためにアダプティブ命令シフトが有効的である．図 21(b) において，1，4，7 段目，2，5，8 段目，3，6，9 段目のローカルメモリに格納する入力画像は各々連続している．基準位置 p0 が次行に移動する際，p0 および p1 は下方のローカルメモリから 2 行を調達でき，新規に 1 行分のみ DDR からローカルメモリへ転送すればよい．再利用可能なローカルメモリに合わせて演算写像を下方にシフトしローカルメモリへの転送を最小化できる [17]．ただし mm や cnn とは異なり，再利用の可否が不連続であるため，新たに，演算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命令シフトが必要となる．

5.4.5 3 種類のアプリケーションプログラムの総括

表 6 に各アプリケーションの総演算数と， 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．gdepth 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 SAD 命令を行う．また，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 6 に示している．

図 22 に，単チップの場合の理想的なダイアグラム（横軸が時間，縦軸がユニット番号）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 とローカルメモリ転送（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 のローカルメモリと 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）およびローカルメモリの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

Table 6: アプリケーションと命令マップ

	mm	cnn	gdepth
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

†gdepth: IM=7500, OM=1600, PAD=32

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

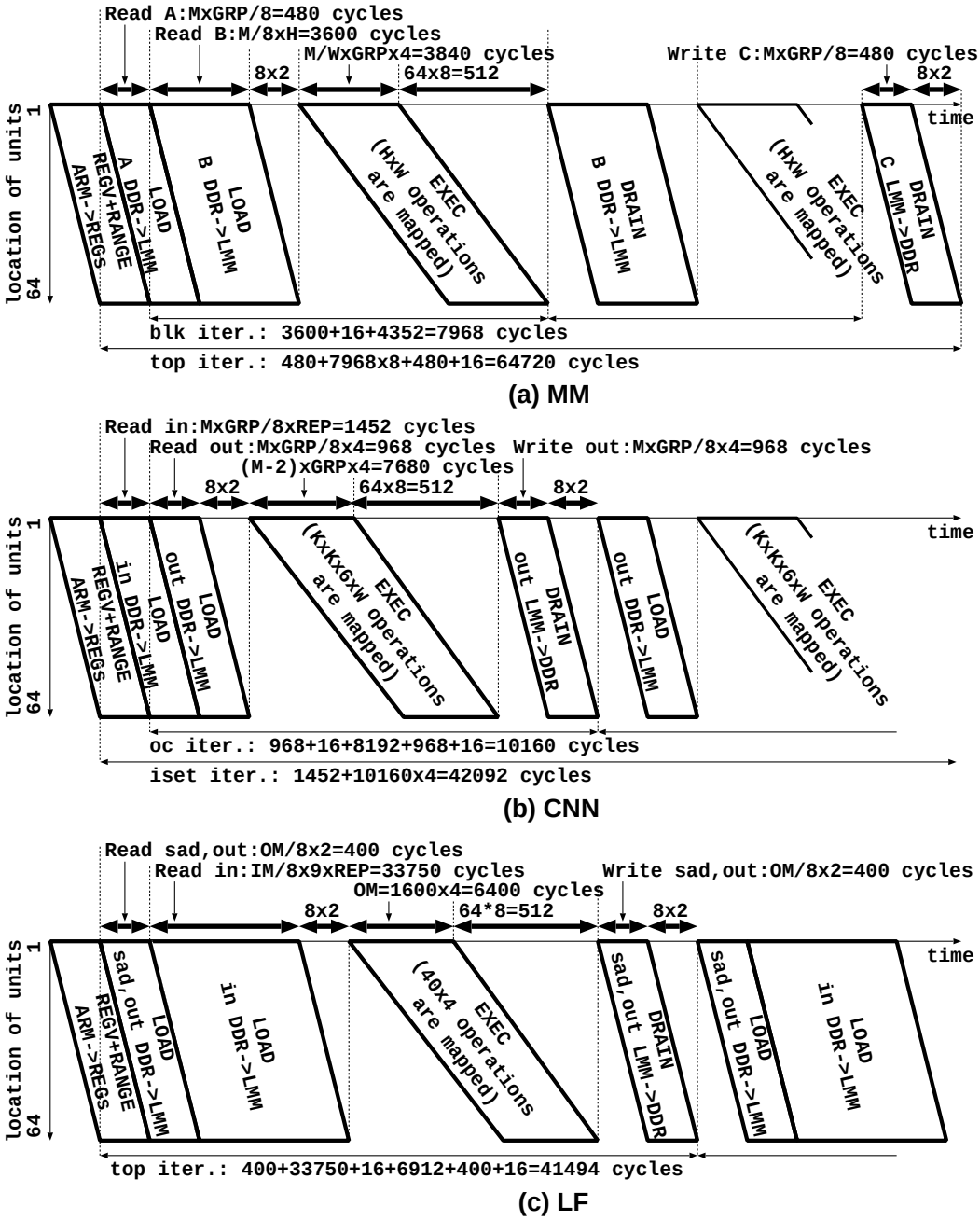

Figure 22: 理想的なダイアグラム

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 (gdepth) 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 (gdepth) の理想的総サイクル数は 16.6M となる．

Table 7: 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
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	150MHz
アクセラレータ段数	64 stages
アクセラレータのローカルメモリ容量	32KB / unit
アクセラレータ評価時 gcc オプション	-O2 -mstrict-align

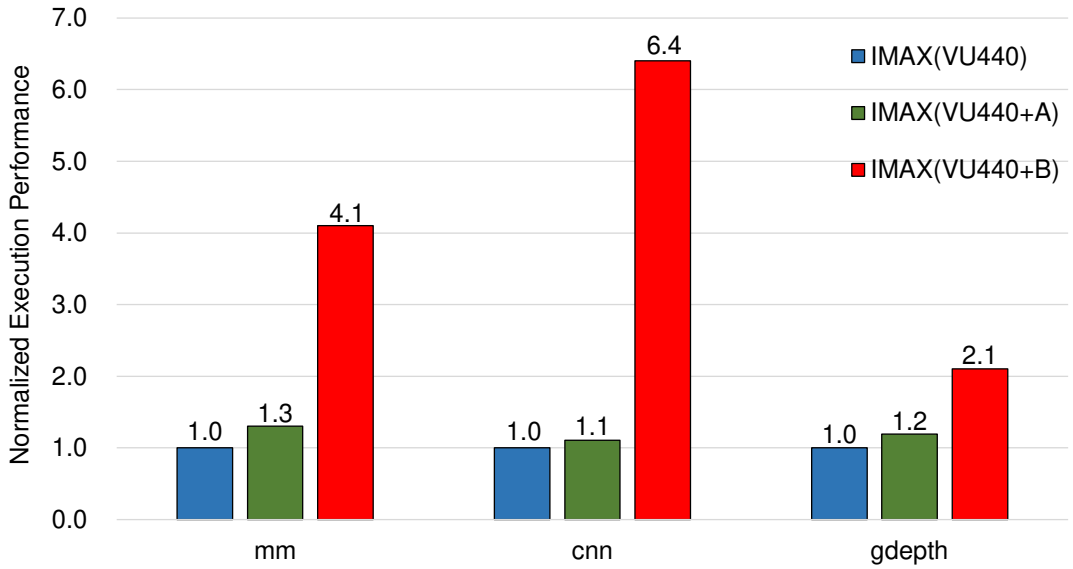

Figure 23: シングルチップ実行性能

6. 評価と考察

本章では，5 章で説明した FPGA 上に実装したプロトタイプシステムを用いて，まず提案手法を段階的に適用して，主要アプリケーションである mm，cnn，gdepth によりシングルチップ実行性能評価を行う．次に，スター型，ツリー型，デイジーチェイン型の 3 種のマルチチップ構成を形成し，評価する．評価方法は，同じく 5 章で説明した 10 種の評価アプリケーションを実行することにより，最大 4 チップで評価する．また，デイジーチェイン型においてのみプロトタイプシステム上での 5 チップ以上の物理接続が可能なため，最大 8 チップで評価を加えて行う．アプリケーション実行性能評価における諸条件を表 7 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．

図 23 は列マルチスレッドを既に適用した IMAX に対して，段階的にチップ内バス並列化と多重ループ一括実行機構を適用した場合のアプリケーション実行性能を示している．それぞれ，列マルチスレッドを適用した IMAX(VU440)，それに加えてチップ内バス並列化を適用した IMAX(VU440+A)，最後に IMAX(VU440+A) に多重ループ一括実行機構を適用した IMAX(VU440+B) を表している．それぞれ，IMAX(VU440) の性能によって正規化している．その結果，IMAX(VU440) と比較して，IMAX(VU440+A) はそれぞれ，1.3 倍，1.1 倍，1.2 倍となった．また，IMAX(VU440+A) に多重ループ一括実行機構を適用した IMAX(VU440+B) はベクトル長の短い mm や cnn で比較的高い効果を発揮し，それぞれ 4.1 倍，6.4 倍となっている．

図 24，図 25，図 26 に，それぞれスター型，ツリー型，デイジーチェーン型のアプリケーション実行性能評価を示す．マルチチップの構成は図 17 で示したスター型，ツリー型，デイジーチェーン型の 3 種類で評価する．それぞれ，1 チップの実行性能によって正規化しており，比較対象として Jetson TX2 の実行性能を載せている．1 チップのスター型，ツリー型，デイジーチェーン型は同じ構成を指すため，それぞれの性能は同じである．IMAX は 1 チップ構成の面積は $28nm$ にて推定 $8.4mm^2$，Jetson TX2 は面積 $16nm$ にて推定 $43.6mm^2$ としている．評価結果から，IMAX は全構成の gather を除いた全アプリケーションにおいて，Jetson TX2 の実

行性能を上回っていることが分かる．mm と cnn は全ての構成において，それぞれ優れたスケーラビリティを示している．ただし，wave2d など，デイジーチェーン型では，1 チップの実行性能よりも低い実行性能を示しているアプリケーションも

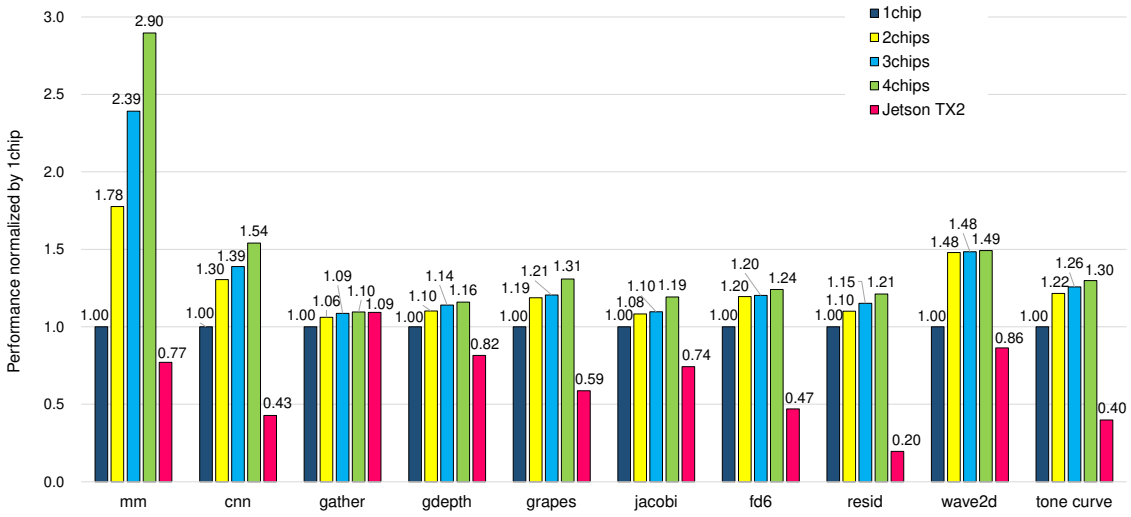

Figure 24: アプリケーション実行性能 (star)

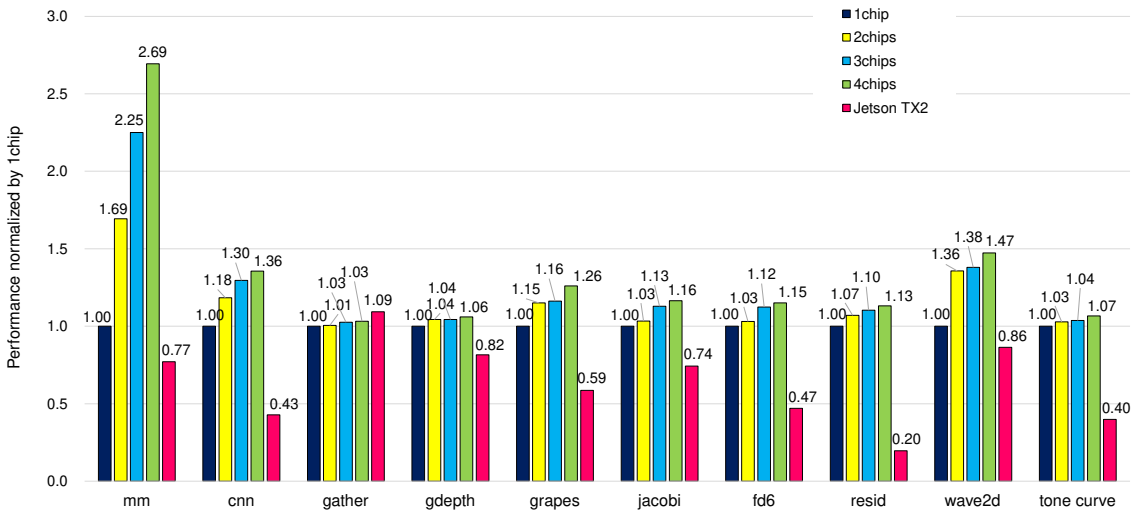

Figure 25: アプリケーション実行性能 (tree)

存在する．図 27， 図 28， 図 29 に，それぞれのステート別プロファイリング結果を示す．スター型においては，ARMv8 でバースト実行の準備するための時間でも

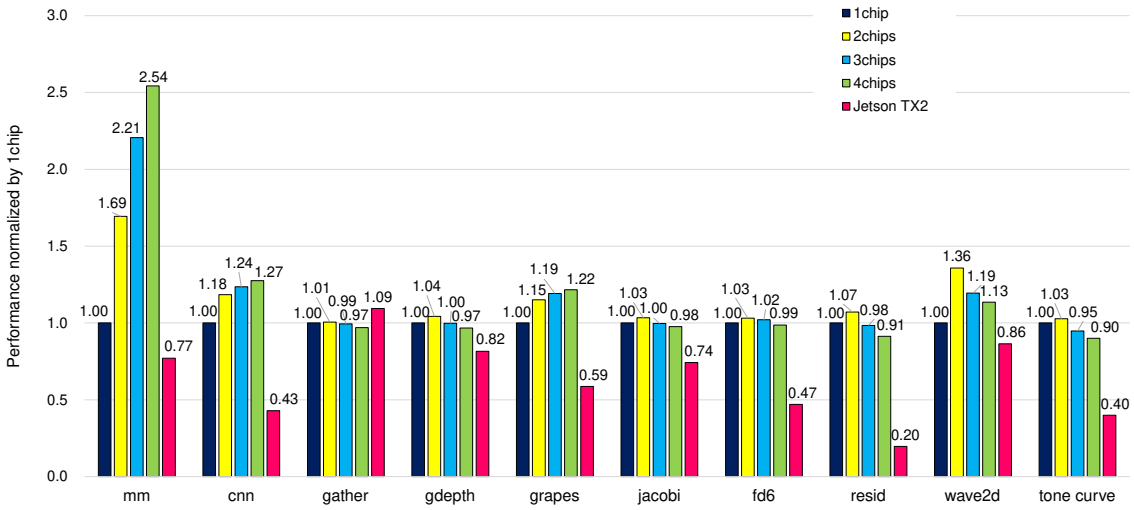

Figure 26: アプリケーション実行性能 (daisy chain)

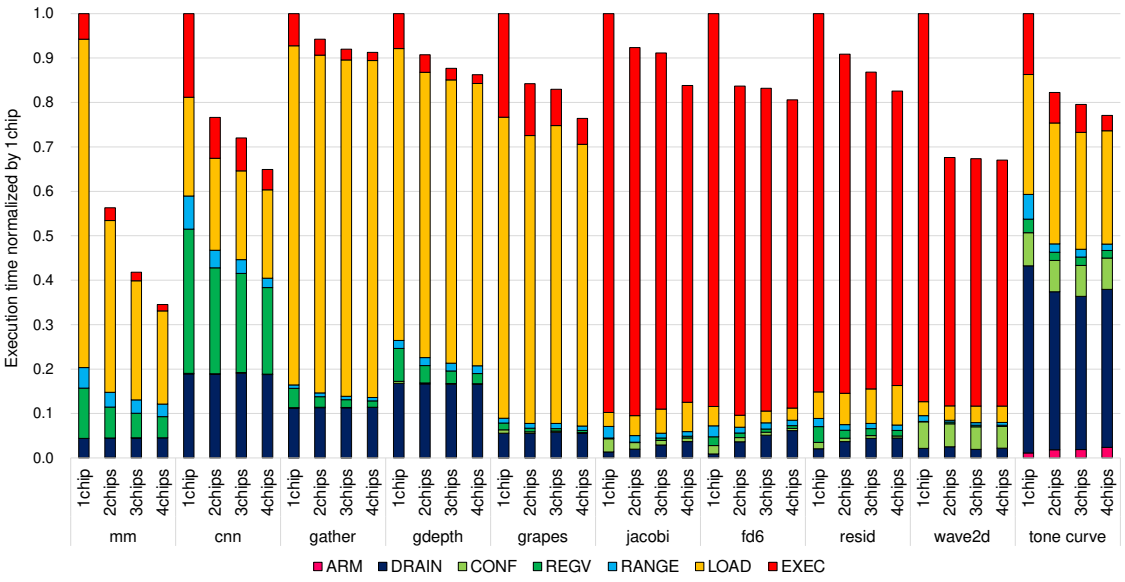

Figure 27: ステート別プロファイリング結果 (star)

ある ARM や，PIO によるマッピング操作である CONF を省略することができる．加えて，LOAD が全体的に支配的であり，DRAIN は比較的小さくなる．対して，

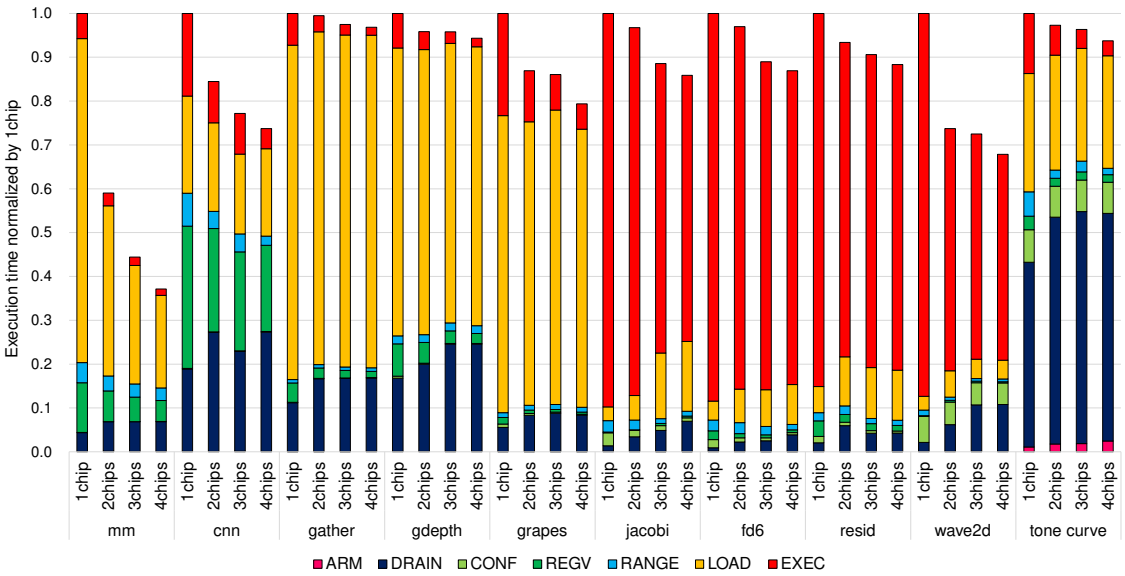

Figure 28: ステート別プロファイリング結果 (tree)

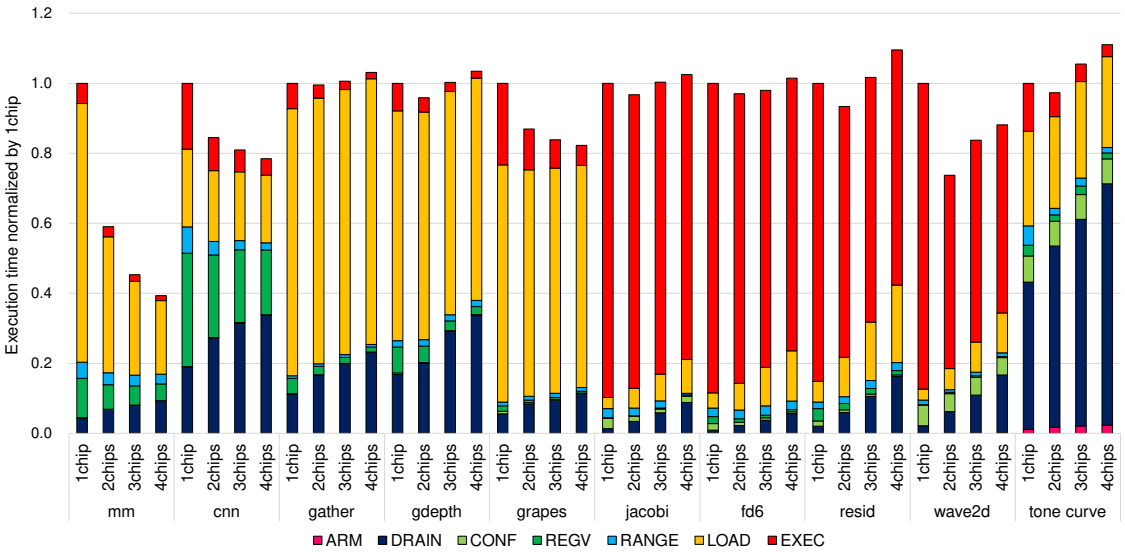

Figure 29: ステート別プロファイリング結果 (daisy chain)

スター型，デイジーチェイン型と接続リンク数が増えるにつれて DRAIN の割合が大きくなる．mm はローカルメモリを効率的に再利用することができるため，チップ数の増加とともに，優れたスケーラブルな性能を達成することができている．また，cnn ではツリー型やデイジーチェーン型において，多くのレジスタを初期化する必要があるため REGV が支配的になるが，チップ間で効率良くデータが共有され，複数の PIO を単一の PIO としてまとめることで，実行時間が削減されている．Light-Field 画像処理 (gather, gdepth) では，EXEC と REGV のみが削減されるため．前述した条件付きストアによる予期しないアドレスの不規則なメモリアクセスにより，ローカルメモリの再利用が不明瞭になり，LOAD が支配的になる．そのため，マルチチップ実行の恩恵である演算時間の削減では性能向上が大きく見られない．grapes では，大量のデータを共有する必要があり，LOAD が支配的になる．その他のステンシル演算 (jacobi, fd6, resid, wave2d) では，EXEC が支配的であり，マルチチップ構成により効果的に削減される．これらのステンシル演算は事前にデータを読み出す，Pre-LOAD を使用するため，LOAD は EXEC として計上されている．そのため，見かけ上は EXEC が削減されていないように見えるが，実際にはチップ枚数の増加に伴い EXEC が減少している．最後に，IMAX では LUT をローカルメモリに保存して再利用することが可能ではあるが，tone curve では LUT 以外のローカルメモリを再利用する機会が無いため，LOAD と DRAIN は同じ時間消費してしまう．

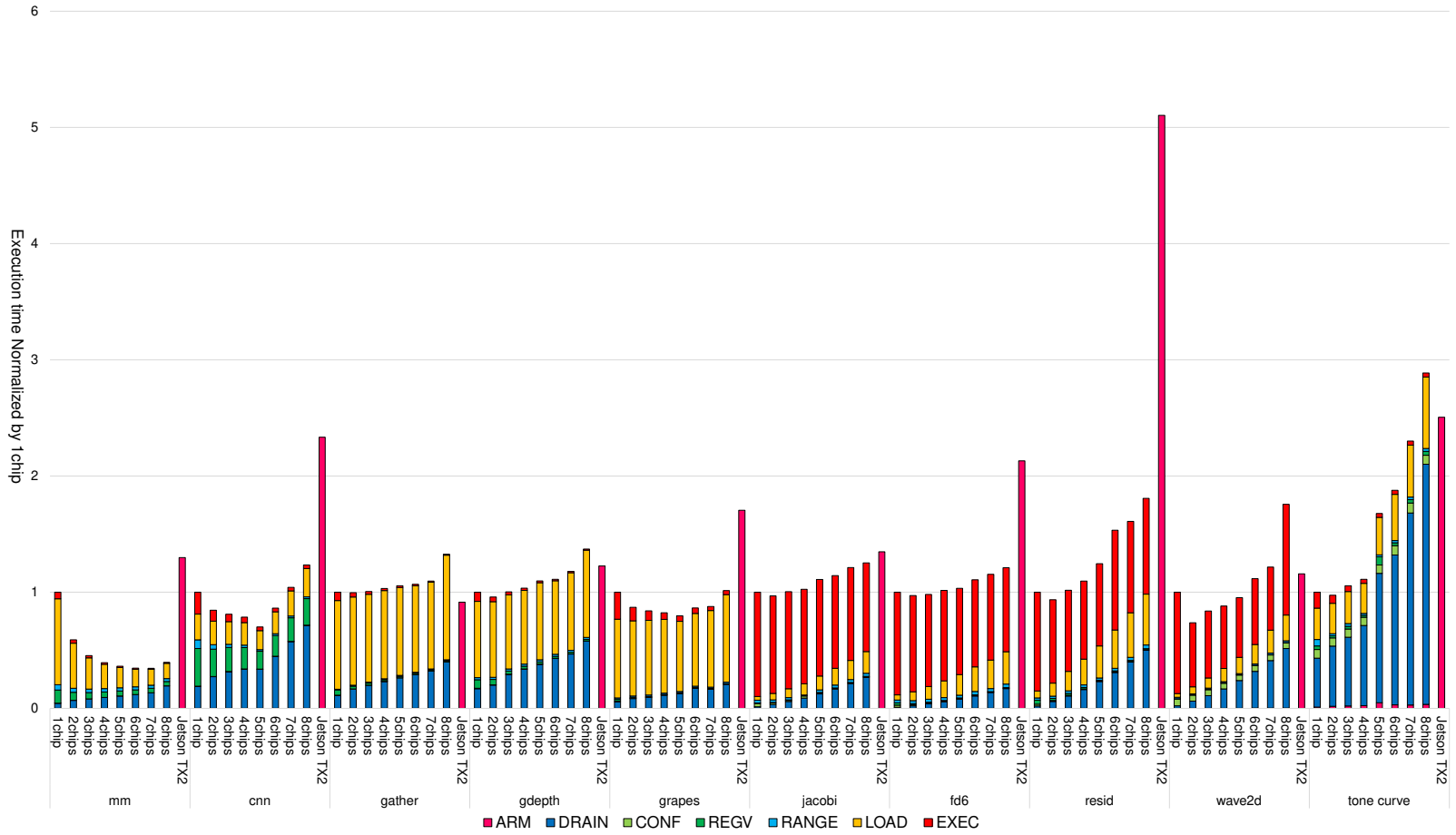

Figure 30: ステート別プロファイリング結果 (daisy chain, 最大 8 チップ接続時)

図 30 にデイジーチェイン型の最大 8 チップのステート別プロファイリング評価結果を示す．比較対象として Jetson TX2 の実行性能を載せ，演算時間を ARM 時間として計上している．mm は 7 チップ接続，cnn は 5 チップ接続，gdepth は 2 チップ接続，ステンシル演算では大気シミュレーションとして用いられる grapes は 5 チップ接続，その他ステンシル演算では 2 チップ接続での実行が最も性能が高くなった．各々のアプリケーションを実行した傾向として，mm や cnn のようにチップ間共有データが多いとマルチチップ化の効果が大きく，デイジーチェン型においても安定した性能を出すことができるが，ステンシル演算のように単純な空間分割による演算ではマルチチップ化の効果が小さく，スター型やツリー型の方がオーバーヘッドが小さく性能が出しやすいことが分かる．IMAX のメモリバンド幅は Jetson TX2 のメモリバンド幅の 32 分の 1 (IMAX は 15.6Gbps であり，Jetson TX2 は 477.6Gbps) であり，エッジデバイスに適したアクセラレータとして十分な性能を出せているといえる．また，IMAX が仮に 28nm の ASIC 化により 900MHz 動作した場合，Jetson TX2 の 17 分の 1 の面積で最大 7.8 倍の性能 (cnn，1 チップ実行時) となり，面積あたりの性能では 130 倍となる．消費電力が面積に比例すると考えた場合，同一性能あたりの消費電力も同じ比率であると考えられる．加えて，マルチチップ評価においても，最大 5.4 倍の実行性能 (resid，2 チップ接続時) を出しつつ，十分なスケーラビリティを達成できているといえる．つまり，電力制約下にあるエッジデバイス用途を前提として，限られたメモリバンド幅であるデイジーチェイン型においても十分な性能を出せることがわかった．

7. おわりに

本研究では，エッジデバイスに適用できるように，外部メモリインターフェース増やすことなくマルチチップ可能なシストリックアレイ IMAX を提案した．さらに，長距離配線や配線混雑を防ぎ，複雑なデータ参照パターンを持つ演算への対応可能とする列マルチスレッディング，読み出し遅延を改善するチップ内バス並列化，そしてベクトル長が短い演算に対しても十分な性能を出すことを可能とする多重ループ一括実行機構を提案し，IMAX に適用した．

FPGA 上に IMAX 64 stages を実装し，ARMv8 をホストとしてアクセスできるプロトタイプシステムを開発した．そして，プロトタイプシステムを用いて，スター型とツリー型，デイジーチェイン型において最大 4 チップのマルチチップ構成を形成し，行列積や畳み込み演算，Light-Field 画像処理，ステンシル演算などから成る 10 種の評価アプリケーションの実行性能を評価した．上記の評価に加えて，デイジーチェイン型においてのみ最大 8 チップのマルチチップ構成で評価を評価した．その結果，最大性能では IMAX の 1 チップ構成と比較して，スター型，ツリー型，デイジーチェーン型の 4 チップ構成はそれぞれ 2.9 倍，2.69 倍，2.54 倍の実行性能を得られることが明らかとなった．また，デイジーチェーン型の最大 8 チップ構成の評価では最大 2.9 倍 (行列積，7 チップ接続時) となった．加えて，畳み込み演算では 5 チップ接続，Light-Field 画像処理では 2 チップ接続，ステンシル演算では大気シミュレーションとして用いられる grapes は 5 チップ接続，その他のステンシル演算では 2 チップ接続が最も高速であった．行列積や畳み込み演算のようにチップ間共有データが多いとマルチチップ化の効果が大きく，デイジーチェン型においても安定した性能を出すことができるが，ステンシル演算のように単純な空間分割による演算ではマルチチップ化の効果が小さく，スター型やツリー型の方がオーバーヘッドが小さく性能が出しやすい．しかし，デイジーチェーン型においてもエッジ向け GPU である Jetson TX2 と比較して，ステンシル演算 (resid) 実行時には最大 5.4 倍の実行性能を出しつつ，他のアプリケーションにおいても高い実行性能を実現することができた．結論としては，電力制約下にあるエッジデバイス用途を前提とし，限られたメモリバンド幅であるデイジーチェーン型においても十分な性能が出せることが明らかとなった．

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．学会発表やミーティングなどでお世話をしていただきました本学の中田尚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．研究活動から私生活に至るまでを支えてくださった菊谷雄真氏，2 年間の研究生活や私生活で多くを支えてくださった同期の池田裕哉氏，新谷隆太氏，西本宏樹氏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本田卓氏，滝下雄太氏，野村武司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と小南真帆氏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". 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] "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.

[6] 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.

[7] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y.Chen, and O. Temam. The q100 database processing unit. In *IEEE Micro*, volume 35, pages 34–36, May-June 2015.

[8] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y.Chen, and O. Temam. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. In *2016 ACM Architectural Support for Programming Languages and Operating System (ASPLOCS'14)*, pages 269–284, Febuary 2014.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] T. Nowatzki, V. Gangadhan, K. Sankaralingam, and G. Wright. Pushing the limits of accelerator efficiency while retaning programmability. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 12–16, Febuary 2016.

[15] 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.

[16] 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.

[17] 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. 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

2. Jun Iwamoto, Yuma Kikutani, Renyuan Zhang and Yasuhiko Nakashima. “CGRA Cascading for Narrow Memory Bandwidth and Low Cost”, xSIG2019 - The 3rd cross-disciplinary Workshop on Computing Systems, Infrastructures, and Programming, Mar 2019

3. Jun Iwamoto, Renyuan Zhang and Yasuhiko Nakashima. “Evaluation of a Chained Systolic Array with High-Speed Links”,IEEE 2019 Seventh International Symposium on Computing and Networking Workshops, Nov 2019.

4. Jun Iwamoto, Yuma Kikutani, Renyuan Zhang and Yasuhiko Nakashima. “Daisy-chained Systolic Array and Reconfigurable Memory Space for Narrow Memory Bandwidth”, IEICE Trans., Vol.E103-D, No.03, to appear, Mar 2020

受賞

1. Jun Iwamoto, Yuma Kikutani, Renyuan Zhang and Yasuhiko Nakashima. “CGRA Cascading for Narrow Memory Bandwidth and Low Cost”, xSIG2019 - The 3rd cross-disciplinary Workshop on Computing Systems, Infrastructures, and Programming, Mar 2019, Outstanding Originality Award