博士論文

CGRA 型プログラマブルアクセラレータの画像認識応用に関する研究

一倉　孝宏

2019 年 06 月 06 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
博士 (工学) 授与の要件として提出した博士論文である.

一倉　孝宏

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)
トラン・ティ・ホン 助教	(副指導教員)
張 任遠 助教	(副指導教員)

CGRA 型プログラマブルアクセラレータの画像認識応用に関する研究*

一倉　孝宏

内容梗概

高速・高精度画像認識のために，組み込み機器に Convolutional Neural Networks (CNNs) を搭載することが求められている．ただし，CNNs の計算量は膨大であり，従来機器に搭載される CPU や GPU では十分な性能が得られない．CNNs に特化したアクセラレータ（Domain Specific Accelerators (DSAs)) が多数報告されているものの，微細化によるコストダウンの恩恵がなくなった現状において LSI 開発費を回収するには，様々なメモリ参照パタンに対応できる汎用的アクセラレータが望ましい．そこで本論文では，Coarse-Grained Reconfigurable Architecture (CGRA) とローカルメモリを備え，様々な離散ステンシル計算を効率よく計算できる既存プログラマブルアクセラレータ EMAXV を CNNs 向けに改良した EMAXVR を提案する．EMAXVR は，CNNs 計算を効率化するために，1）多重ループ制御機構，2）マルチディレクションブロードキャストバス，及び，3）スクラッチパッドメモリを搭載した．評価の結果，AlexNet と VGG16 における畳み込み層の計算時に，ARM CoretexA9 に比べて 60 倍以上，NVIDIA Tegra K1 に比べて最大 35 %高速化できることを確認した．さらに DSAs と比較すると，演算器利用率が 18 %低いものの，消費電力指標である計算回数あたり DRAM アクセス量では，同等にまで迫れることを確認した．

キーワード

深層学習, 畳み込みニューラルネットワーク, 粗粒度リコンフィギャラブルアーキテクチャ, 演算器利用率, データ再利用

*奈良先端科学技術大学院大学 情報科学研究科 博士論文, 2019 年 06 月 06 日.

A Study on the programmable accelerator of CGRA type for image recognition application*

Takahiro Ichikura

Abstract

For high-speed and high-precision image recognition, it is required to install Convolutional Neural Networks (CNNs) in embedded devices. However, the computational complexity of CNNs is enormous, and sufficient performance can not be obtained with CPUs and GPUs installed in conventional equipments. Although numerous accelerators specialized for CNNs (Domain Specific Accelerators (DSAs)) have been reported, in the present situation that cost reduction benefits of miniaturization are lost, in order to recover LSI development costs, it is desirable to design accelerators so as to manage various memory reference patterns. In this paper, I propose EMAXVR that can efficiently calculate various discrete stencil calculations for CNNs with Coarse-Grained Reconfigurable Architecture (CGRA) and local memory based on an existing programmable accelerator EMAXV. EMAXVR has 1) multiple nested loop manager, 2) multidirectional broadcast bus, and 3) scratch pad memory for efficient CNN calculation. As a result of the evaluation, it was confirmed when computing the convolution layer in AlexNet and VGG 16, EMAXVR can execute 60 times faster than ARM Coretex A9 and up to 35% faster than NVIDIA Tegra K1. Furthermore, compared with DSAs, I confirmed that even though the computing unit utilization

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, June 06, 2019.

rate is 18% lower, the amount of DRAM access per calculation, which is a power consumption metric, is almost the same.

Keywords:

Deep Learning, Convolutional Neural Networks, CGRA, ALU utilization, data reuse

目 次

図 目 次

表 目 次

1. 序論

1.1 研究の背景

Deep Learning を用いた画像認識技術がめざましい進歩を遂げている．国際的な画像認識の大会である ILSVRC (ImageNet Large Scale Visual Recognition Challenge)[1] の 2012 年大会において，従来の人間が特徴量を設計した方式では，エラー率が 26 %台であったものを，Deep Learning を用いたチームが 16 %台のエラー率を出したことで注目を集めた [2]．そして，2015 年の同大会では，エラー率が 3.6 %にまで低下した [3]．同じ画像認識のテストを人間が行うとエラー率が 5 % [4] になると言われていることから，Deep Learning を用いた画像認識の精度が，人間以上の精度を実現したことになる．

ここで使用されている Convolutional Neural Networks (CNNs) は，顔認証 (Face Recognition)[5], 人物の姿勢推定 (Human Pose Estimation)[6, 7]，リアルタイム物体検出 (Real-Time Object Detection)[8, 9] や，セグメンテーション (Segmentation)[10, 11] などの色々なアプリケーションで使用できる．このようなアプリケーションを組み込みシステムに搭載し，リアルタイム処理できれば，色々な分野での活用が期待できる．

表 1 に一般的な CNNs の画像 1 枚あたりの積和演算量を示す．リアルタイム処理にはこれらの演算を 1 秒間に 1 回以上実行できる必要がある．表 2 に現在の組み込みシステムで使用されている NXP 社の SoC i.MX6QuadPlus Processor[12] に搭載されている CPU と GPU の積和演算性能を示す．これらの表から演算量が最大の VGG16 では，理論最大性能で計算できた場合でも 1 秒間に 1 回も計算できないことがわかる．CNNs の計算量の 90 %以上が畳み込み演算である．そのため，畳み込み演算を効率よく計算するための Domain Specific Accelerators (DSA) [13][14] が多数提案されている．これらの DSA は，畳み込み演算に特化することで，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を減らしている．外部メモリアクセスの電力消費は，回路内の演算やデータ移動と比較して非常に大きいことがわかっている [13]．そのため，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

表 1: 画像一枚あたりの CNNs の積和演算回数

CNNs	Number of operations [Giga operations]
AlexNet[2]	0.74
VGG16[23]	15.4
GoogleNet[24]	1.5
ResNet34[3]	3.64

表 2: NXP 社の SoC i.MX6QuadPlus Processor の積和演算性能

Processor		Frequency	Number of Arithmetic units	Cycle number of multiply-add operations	Maximum number of multiply-add operations per sec.
ARM CoretexA-9	1core	1 GHz	4	2	2 GOPS
	2core		8		4 GOPS
	4core		16		8 GOPS
Vivante GC2000+		0.72 GHz	16	1	11.5 GOPS

る．一方，微細化によるコストダウンの恩恵がなくなった現状において特定の機能のみに特化した LSI では，開発費の回収を図るのは難しいことが予想される．

そこで，様々な離散ステンシル計算を効率よく計算することを目的に開発した CGRA[15, 16] 型のプログラマブルアクセラレータ EMAX シリーズ [17, 18, 19, 20, 21] の EMAXV[22] に対して，多重ループ制御機構, マルチディレクションブロードキャストバス及びスクラッチパッドメモリを加えることで，CNNs の計算の多数を占める畳み込み演算の計算を効率よく実行できるように，演算時の演算器利用率とデータの再利用性を高めたアクセラレータ EMAXVR を提案する．

1.2 研究の成果

本研究で得られた主な成果は以下の通りである．

1. CNNs の畳み込み演算を効率よく計算する CGRA 型プログラマブルアクセ

レータの仕組みを確立した

CNNs の計算時間の大部分 (90 %以上) を占める多重ループの畳み込み演算を効率よく計算するために多重ループの計算を 1 回の起動で実行する仕組みを提案し，シングルループの計算と比較して畳み込み演算の計算速度を 20 倍高速化した．さらに，電力消費が大きな外部メモリへのアクセスを抑制するために，マルチディレクションブロードキャストによるデータ転送回数の低減と，スクラッチパッドメモリの追加により，外部メモリ通信量を従来の 15 %に低減した．

2. CNNs の畳み込み演算を効率よく計算する CGRA 型プログラマブルアクセレータの命令マップを確立した．

   カーネルサイズなどの各種パラメータが異なる畳み込み演算において，アクセラレータの演算器利用率を最大化する命令写像を示し，従来の命令マップより計算速度を 1.5 倍高速化した．

3. CGRA 型プログラマブルアクセレータの CNNs の計算における有効性を実証した．

   本プロセッサをクロックサイクルアキュレートのソフトウェアシミュレータにより性能評価を行い，組み込み CPU や GPU より高い性能を実現することを示した．さらに，プログラマビリティを有したアクセラレータでありながら，CNNs の計算に特化した DSAs に対して，演算器利用率は少し劣るものの，外部メモリの通信量を同等に迫れることを示した．

2. 関連研究

背景においてCNNsの組み込みシステムへの実装が期待されているものの，従来の組み込み機器に搭載されているCPUとGPUでは実現が困難であることを述べた．これに対して，CNNsのアルゴリズム開発及び機械学習において広く使用されているNVIDIA社のGPUを搭載したSoC[25, 26]を用いることが提案されている．このGPUは，組み込み機器向けのため演算器数は抑えられているものの，前述した現在の組み込みシステムに搭載されているGPUに対して10倍以上の演算器数が搭載されている．他に，CNNsの計算に特化することで高効率に計算可能なCNNs向けDSAを使用する方法も提案されている．本章では，GPUとDSAの構成と課題について説明する．

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の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の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要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，CNNsの畳み込み計算や規模の大きい画像処理などは，ほとんどがループアンローリング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OoO実行機構の恩恵を受けないとは，CNNsのようなアプリケーションでは並列実行可能な命令は静的に解決されるため，ROBが必要ないことを指す．そこでGPUは投機実行機構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[27]，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 Thread）方式と呼んでいる．また簡単な分岐

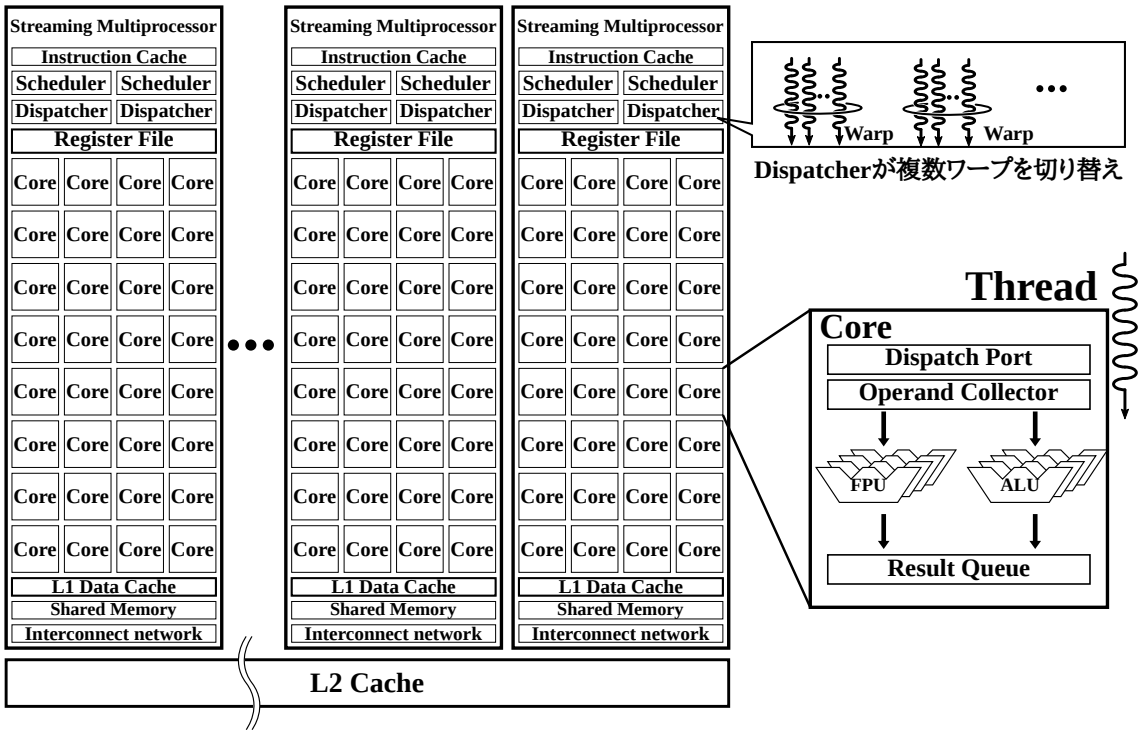


図 1: GPU の構造

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 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，及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．これらの機構により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

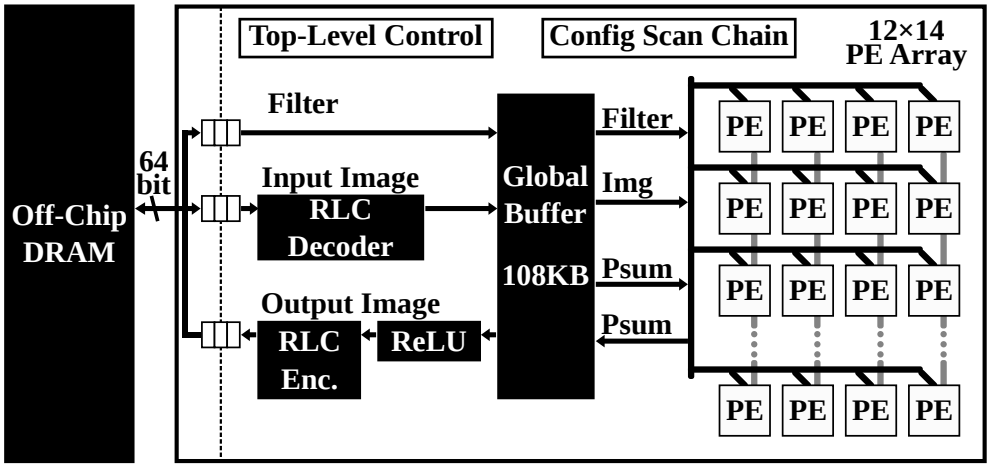


図 2: Eyeriss

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．また，サーバー機器向けの GPU では広い主記憶帯域を確保するために GPU 専用の GDDR が搭載されているが，組み込み機器向けの SoC の場合は，主記憶は CPU と共通のものを使用するため広い帯域を確保することが難しい．

2.2 DSA

図 2 は DSA の一つである Eyeriss[13] の構成図である．この他にも FlexFlow[14] などの数多くの推論専用 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答率はそれほど下がらないことが示されているため [28]，重みの情報量を削減し 16bit の固定小数積和演算を行う．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 [29]．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動作により，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性能を達成できる．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できない問題点がある．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行くと考えられ，新しい DSA が出てくる度に新しい ASIC を設計しなすことは開発コスト面で現実的ではない．複数の用途に使用できるように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．

3. EMAXVの概要とCNNs応用の予備評価

プログラマビリティを残したアクセラレータとしてCGRA型プログラマブルアクセラレータEMAXシリーズが提案されており，グラフ処理[21]やLight Fieldのレンダリングの計算[22]で高い性能が発揮できることが確認されている．本章では，EMAXシリーズのEMAXVについて概観し，EMAXVでCNNsの計算を行った予備評価結果を示す．

3.1 EMAXVの全体構成

図3に，EMAXVのブロック図を示す．EMAXVは，CGRA部とFSM部の2つのブロックで構成される．CGRA部は，2次元格子状に演算器とローカルメモリ(LMM)を有するProcessing Element (PE)を配置し，それらを様々なバスによって接続している．PEは，列方向に4つ配置され，行方向については設計時に決定する任意の数を搭載することが可能になっている．バスは3種類あり，(A)FSM Busは，CGRA部とFSM部を接続するBusであり，列毎に備えている．(B)HBC Busは，FSM Bus経由で転送されるデータを水平方向にブロードキャストするためのBusである．同じ行の異なるPEに，同じデータを転送する際に，効率的に転送するために使用する．(C)BR Busは，行間を接続するためのBusであり，各PEが持つ4つの出力レジスタが接続され，次の行のPEはこれらから任意のものを選択して使用できる．

次に，FSM部はCGRA部と外部メモリを接続するインターフェースとシステム全体のコントロールを行うブロックである．FSM部では，計算に使用するデータだけでなく，CGRA部がどのように動作するかを決定する設定データの転送も行う．FSM部にはDMA機能が搭載されており，メモリ間の転送や設定データをバースト転送することで効率よくデータ転送を行う．

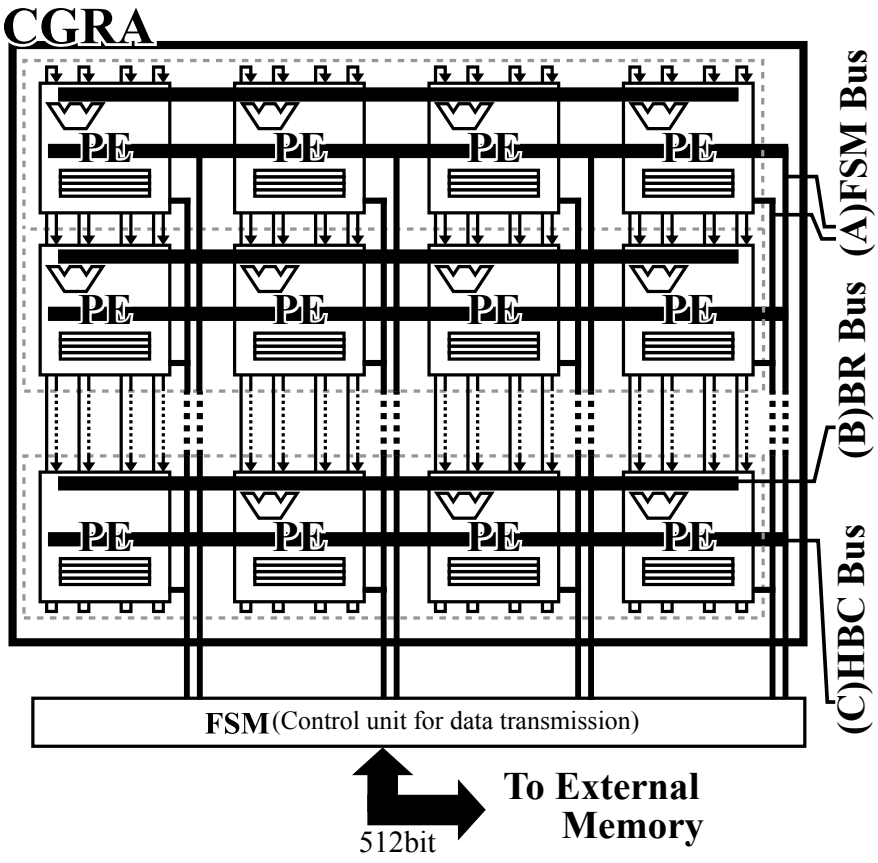


図 3: EMAXV のブロック図

3.2 EMAXV の Processing element の構成

図 4 に，Processing element (PE) の詳細を示す．本 PE の最大の特徴は，ローカルメモリ (LMM）を持っていることである．計算で使うデータは，外部メモリから LMM に転送され，LMM から読み出して計算する．計算結果も LMM に格納され，後から，外部メモリへ転送する．LMM はランダムアクセスメモリであり，非連続なアドレスへのアクセスでもペナルティはなく，読み書きすることが可能である．LMM にアクセスするために実効アドレス生成器 (Effective Address Generator (EAG)) を 2 つ備えており，LMM からの読み出し，又は LMM へ書き込みを最大 2 種類同時実行できる．1 つの演算器は複数ユニットから構成され，算術演算，論理演算及び独自演算が実行できる．これらのユニットはパイプライン実行が可能で，加算とシフトを別のユニットに搭載することで，加算後にシフト演算するといった，算術演算と論理演算の連続実行が可能である．Configuration register に，PE がどのように動作するかの設定が格納されている．各行は，各 PE が持つ 4 つの Boundary register (BR) と，それらをつなぐ BR Bus を介して接続される．これらを使って，LMM から読み出した値，計算結果及び次行に渡すデータを受けとる．BR は初期値を設定することが可能で，固定値と LMM から読み出した値で演算することが可能である．LMM にデータ転送する書き込み用の FSM Bus は，Horizontal Broadcast Bus (HBC Bus) を介して接続されている．HBC Bus には，同一行の PE の FSM Bus が全て接続されており，各 PE の FSM Bus からのデータを同一行の他 PE にブロードキャストできる．これにより，同一行の PE に同じデータを転送する場合に，転送回数と使用する FSM Bus 数を減らすことができる．読み出すための FSM Bus は，各 PE の LMM に直接接続している．EAG を用いた LMM へのアクセスと，FSM Bus を用いた LMM へのアクセスは同時に実行でき，これを用いて計算中にデータの入替えを行う．

3.3 EMAXV の状態遷移

図 5(a) に，EMAXV の状態遷移を示す．各状態は，表 3 に示している．状態は，CONF，LMMI，LOAD，REGV，EXEC と DRAIN という 6 つで，PLOAD

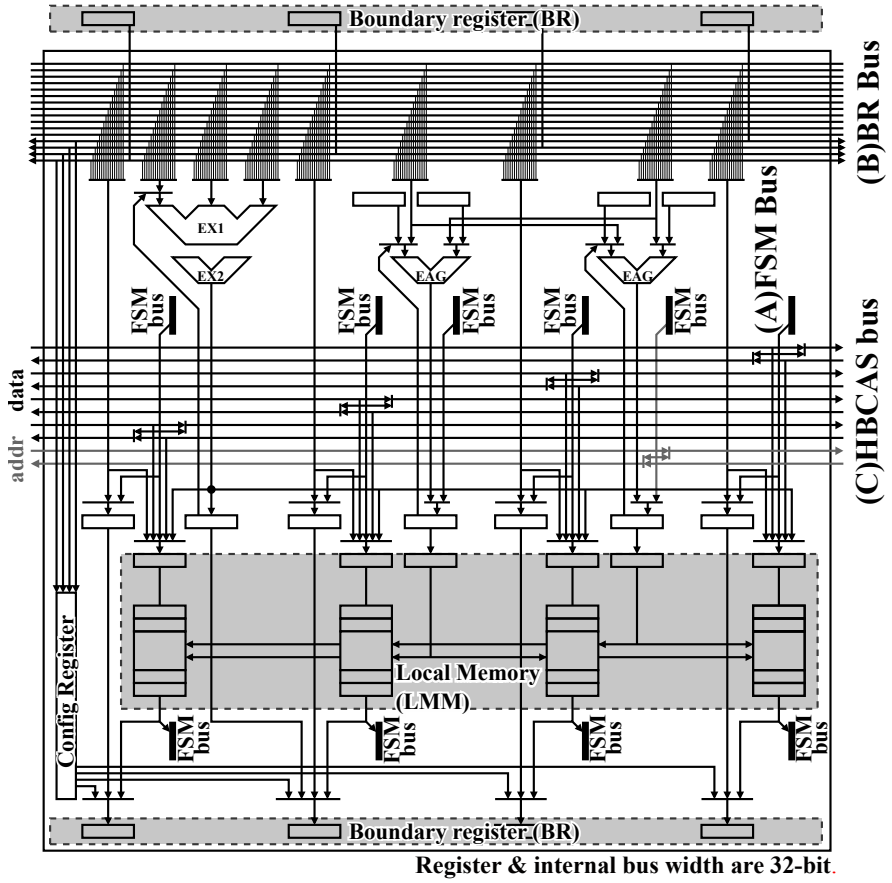
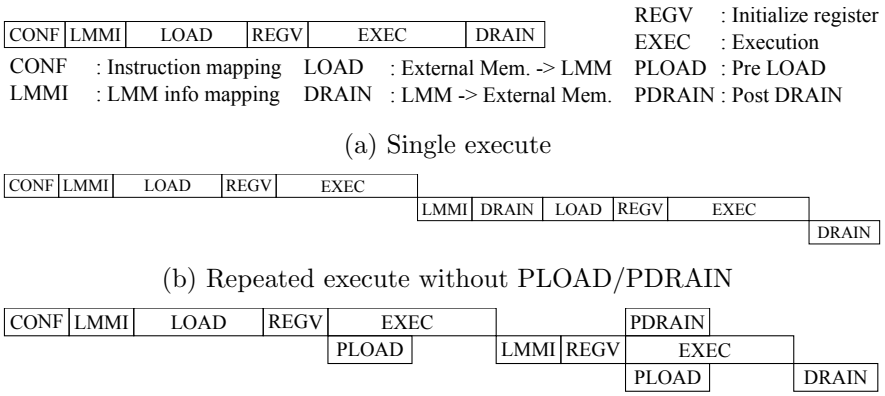


図 4: Processing element (PE) の詳細構造

表 3: EMAXV の状態

State	Action
CONF	Instruction mapping.
LMMI	Set information for data transfer between LMMI and External Memory.
LOAD	Data transfer to LMMI from External Memory.
REGV	Initialize register.
EXEC	Execute calculation.
DRAIN	Data transfer from LMMI to External Memory.
PLOAD	Indicates that the data of the next calculation is being loaded during state of EXEC.
PDRAIN	Indicates that the previous calculation result is being drained during EXEC state.

と PDRAIN は，EXEC 中の特別な動作を実行していることを示すためのものである．これらの状態を遷移して計算を実行する．CONF では，CGRA 部に命令をマッピングする．すなわち，各 PE の動作設定 (演算，LMM の LOAD/STORE) と PE 間のルーティング設定を行っている．LMMI では，LMM と外部メモリ間で転送するデータの先頭アドレスと転送サイズ及びブロードキャスト情報を FSM に設定する．LOAD では，LMMI で設定された情報に基づいて LMM へデータを転送する．REGV では，各 PE が持つレジスタを初期化する．EXEC では，事前に設定された繰り返し回数だけ計算を実行する．DRAIN では，LMMI で設定された情報に基づいて LMM からデータが転送される．図 5(b) に，EMAXV を繰り返し起動して計算する場合の状態遷移を示す．この場合は，DRAIN は次の起動時に実行される．例えば，1 回目の計算結果の転送は，2 回目に起動した時の DRAIN 状態で実行される．DRAIN の前に，LMMI が実行されているが，1 つ前の設定を保持しており，それに基づいて実行される．また，CONF について，一つ前に実行された時と同じ命令がマッピングされる場合，CONF には遷移せず，前回実行時のマッピングを保持する．さらに，図 5(c) に示す通り，EMAXV には，効率よく計算を行うために，計算中に次回計算に使うデータを事前に LMM へ転送すること (PLOAD) と，計算中に前回計算した結果を LMM から転送すること (PDRAIN) が可能である．LOAD/DRAIN の時間が，EXEC の時間に比べて短い場合に，LOAD/DRAIN の時間を隠蔽することが可能で，計算効率を向上させることができる．



(a) Single execute

(b) Repeated execute without PLOAD/PDRAIN

(c) Repeated execute with PLOAD/PDRAIN

図 5: EMAXV の状態遷移

3.4 EMAXV のプログラマビリティ

EMAXV は独自命令セットで動作し，ユーザープログラムを EMAXV 命令セットに書き換える必要がある．回路規模や全体のクリティカルパスを削減するために，除算や剰余演算のような複雑な命令は搭載せず，単純な命令のみをサポートする．独自命令セットの利点は，特定のアプリケーションに対して最も効率的な命令を持てることにある．ベクトルプロセッサのように，複数の算術演算を同時実行する SIMD 命令を備えたり [22]，CNNs の開発で用いられるフレームワークと同様の精度で計算を行うための浮動小数点演算命令を備えたり [19]，CNNs 向け DSA で用いられる 16bit 演算命令を備えることができる．また，データサイズに合わせたビット幅のロード/ストア命令も搭載可能である．命令を限定して，回路規模を抑えることもできるし，様々なアプリケーションに対応するために，限定せず搭載することもできる．

3.5 予備評価

EMAXV における CNNs の畳み込み演算性能を明らかにするため予備評価として，計算時間と外部メモリの通信量を計測した．表 4 に評価で使用した AlexNet の畳み込み層の構成と積和演算回数を示す．データタイプは単精度浮動小数点数

表 4: 評価で使用した AlexNet の構成．(C_x : 畳み込み層．Kernel : C_x : 入力チャネル数 × 出力チャネル数 × グループ数@カーネル幅 × カーネル高さ, P_x:カーネル幅 × カーネル高さ．Layer Size : 出力チャネル数@出力画像幅 × 出力画像高さ Number of Ops. : 積和演算回数)

Layer	Kernel	Layer Size	Number of Ops. [MOPS]
Input		3@227 × 227	
C1	3 × 96 × 1@11 × 11	96@55 × 55	105.4
C4	48 × 128 × 2@ 5 × 5	256@27 × 27	223.9
C7	256 × 384 × 1@ 3 × 3	384@13 × 13	149.5
C8	192 × 192 × 2@ 3 × 3	384@13 × 13	112.1
C9	192 × 128 × 2@ 3 × 3	256@13 × 13	74.8

表 5: EMAXV のパラメータ

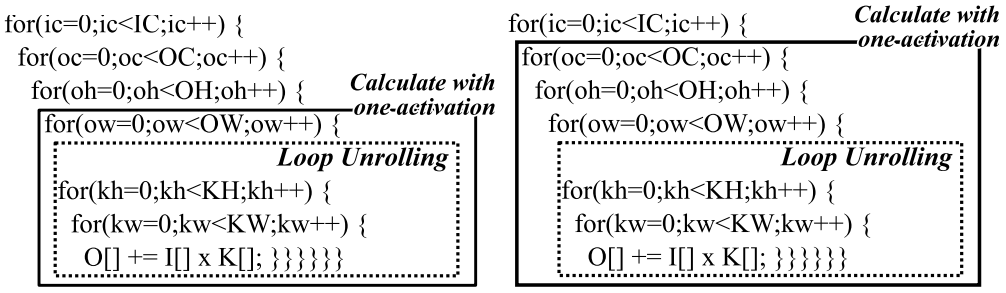
Parameter	Value
Number of PEs	64 rows × 4 columns
Size of LMM	2 KByte
Frequency	240 MHz

表 6: 並列数と理想の計算時間．(Number of parallel : カーネルサイズ × 入力チャネルの並列数)

Layer	Number of parallel	Ideal calculation time
C1	121 (121 × 1)	3.6 msec
C4	100 (25 × 4)	9.3 msec
C7	144 (9 × 16)	4.3 msec
C8	144 (9 × 16)	3.2 msec
C9	144 (9 × 16)	2.2 msec

を使用した．評価には EMAXV のサイクルベースシミュレータを使用した．評価時の EMAXV のパラメータを表 5 に示す．CNNs の畳み込み演算は図 6 に示すように多重ループの積和演算である．今回の評価で用いた命令マップは，カーネルのループをアンローリングしてカーネルサイズ分の積和演算命令を配置し，さらに最外ループの入力チャネルの異なるものを配置可能な数だけ配置した．各層の並列数と理想の計算時間を表 6 に示す．理想の計算時間は，積和演算回数 ÷ 並列数 ÷ EMAXV の動作周波数 の式で算出した．

図 7 に計測した計算時間を示す．C1 層は 33.1 msec，C4 層は 152.5 msec，C7 層は 144.2 msec，C8 層は 108.5 msec，C9 層は 72.6 msec という結果で，理想の計算時間に対して，C1 層は 3.6 倍，C4 層は 16.3 倍，C7-C9 層は 33.3 倍の計



(a) Single loop (b) Multiple loop

図 6: 畳み込み演算の擬似コード. (IC は入力チャネル数. OC は出力チャンネル数. OH は出力の高さ. OW は出力の幅. KH はカーネルの高さ. KW はカーネルの幅.)

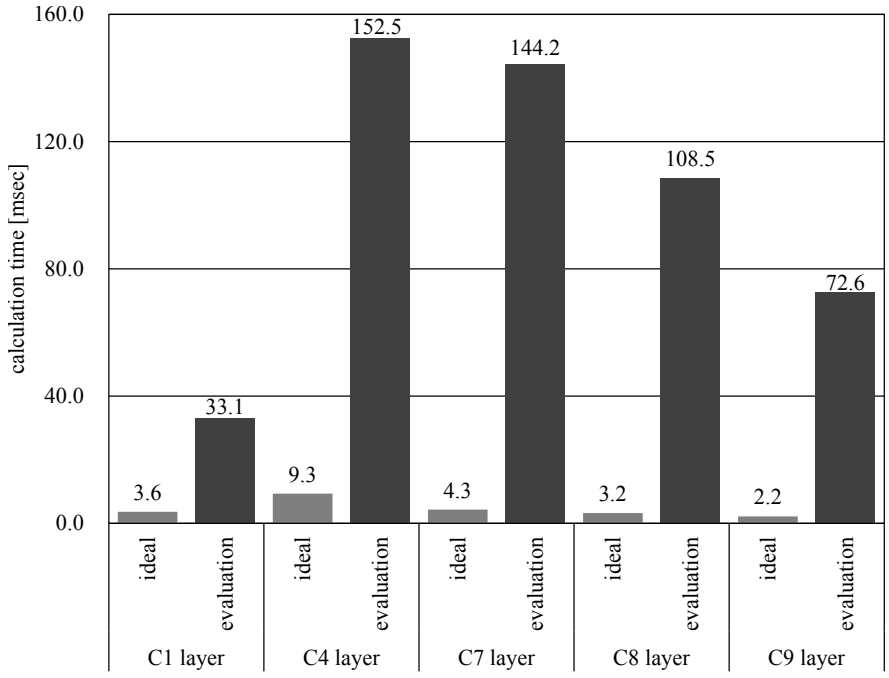


図 7: 予備評価の結果 (畳み込み層の計算時間)

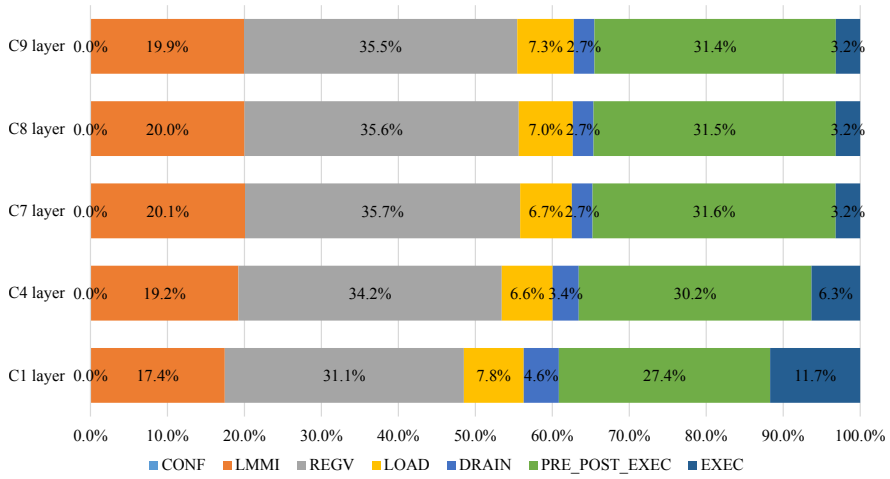


図 8: 予備評価の結果 (計算時間に占める EMAXV の状態別の割合)

表 7: 計算に必要な起動回数

Layer	Start count
C1	15840
C4	82944
C7	79872
C8	59904
C9	39936

表 8: 層毎の入出力サイズ [MByte]

Layer	Input	Output
C1	0.57	1.11
C4	0.27	0.71
C7	0.17	0.25
C8	0.25	0.25
C9	0.25	0.17

表 9: 外部メモリ通信量 [MByte]

Layer	Read		Write	
	Input	Partial sum	Output	Partial sum
C1	3.20	2.34	1.11	2.22
C4	1.68	8.70	0.71	7.83
C7	0.69	4.57	0.25	3.71
C8	1.03	3.35	0.25	2.72
C9	1.03	2.23	0.17	1.82

算時間となっている．計算時間に占める EMAXV の状態別の割合を図 8 に示す．PRE_POST_EXEC とは EXEC 状態におけるパイプラインを埋めるまでの時間と，全計算終了後の後処理の時間を示したものである．EXEC 状態で実際に計算を行っている時間に比べて，非常に多い割合を占めているため，分離して示した．図 8 の通り，起動毎に発生するオーバヘッド時間の内，命令を配置する CONF は命令の再利用ができており，占める割合はおおよそ 0 %であるが，起動毎に必ず必要な LMMI，REGV 及び PRE_POST_EXEC で 70 %〜80 %程度占めてしまっている．これは，畳み込み演算は多重ループであるが，EMAXV は 1 回の起動毎に最内のループの計算しかできないため，図 6(a) に示す通り，1 回の起動では出力画像幅分の計算しか行えない．そのため，全ての計算を行うには，表 7 に示す回数起動しなければならず，オーバヘッド時間が長くなってしまっている．起動回数は，入力チャネル数 ÷ 入力チャネル並列数 × 出力チャネル数 × 出力画像高さ の式で計算できる．また，LOAD と DRAIN も 7 %〜9 %程度占めている．今は，命令配置の簡略化のために使用していていないが，PLOAD と PDRAIN を使用することで計算時間に隠蔽できる見込みである．

次に外部メモリ通信量について確認する．計測したのは，LOAD 及び DRAIN の時に通信される LMM と外部メモリ間の通信量である．予備評価で採用した命令マップでは，重みとバイアスの値は出力画像幅のループでは普遍のため，重みとバイアスはレジスタに固定値を設定し，LMM には格納していない．入力画像はループ中に変化するため外部メモリから読み出して LMM に格納し，計算結果は一旦 LMM に格納された後に外部メモリへ書き込む．表 8 に示す AlexNet の層

毎の入出力サイズを示しており，これが理想的な通信量になる．表 9 に外部メモリ通信量を示す．入力出力以外に，部分和の通信が発生していることがわかる．CNNs の畳み込み演算は，異なる入力チャネルの計算結果を足し合わせなければならない．EMAXV では LMM から LMM へのデータ転送はできないため，部分和を足し合わせるためには，一度外部メモリに転送し，LMM に取り込む必要がある．この通信量は，出力画像サイズ × (入力チャネル数 ÷ 入力チャネル並列数 - 1) の式で計算され，入力チャネル数が大きくなる C4 以降では，出力サイズの 10 倍以上の読み出しと書き込みが発生している．読み出しと書き込みでサイズが異なるのは，書き込みの通信ではストローブ信号を使用することで，バス幅に対して端数の通信に対して，余分書き込みを発生しないようにしているが，読み出しの場合はバス幅のまま通信するため，余分の通信が発生しまうためである．そして，入力画像の読み出しでも，実際のサイズより多く通信している．カーネルのストライドが 1 の場合に，入力画像の同じ行は，カーネルの高さの回数だけ計算に使用される．そのため，C4 では 5 回，C6–C8 では 3 回入力画像の同じ行を外部メモリから読み出しており，ストライドが 4 の C1 は，11 ÷ 4 で，2.7 回読み出している．また，PE に配置できるメモリアクセス命令は最大 2 つのため，CGRA 部の 1 行にメモリアクセス命令の最大数は 8 である．列方向の LMM にはブロードキャストで転送できるため，同じ行の異なる列の LMM に同じデータを LMM に配置する場合でも余分な通信は発生しない．これに対して，C1 のカーネルのサイズは 11 のため，カーネル 1 行分の入力画像を読み出す命令は 11 個必要である．そのため，入力画像を読み出すロード命令を CGRA 部の 2 行に渡って配置しなければならず，2 行分の LMM へ同じ入力画像を格納しなければならない．行方向にはブロードキャストできないため，2 回同じでデータを転送する必要があり，C1 では 2.7 回の同じ入力画像の読み出しをさらに 2 倍実行している．これが，入力画像サイズに比べて多くの通信が発生している原因である．さらに，部分和の時の同様の端数による余分な通信も発生している．

4. CNNs向けに改良したEMAXVR

本章では，CNNsの畳み込み演算を効率化するために，EMAXVに改良を加えて開発したEMAXVRについて説明する．まず，予備評価で見つかった課題に対する解決策を提案し，次に全体構成，EMAXVからの改良点である多重ループ処理，LMMマルチディレクションブロードキャスト，及びスクラッチパッドメモリについて説明する．

EMAXVの予備評価の結果から，起動回数が多いことにより計算の時間の内オーバヘッド時間が大部分を占めてしまい計算が遅い課題と，部分和の読み出しと書き込み及び同じ入力画像データの読み出しによる外部メモリ通信量が多い課題が見つかった．それぞれについて解決策を提案する．まずは，計算が遅い課題について多重ループ処理を導入し，起動回数を削減することで高速化する．具体的には，図6(b)に示すように，1回の起動で出力画像幅，出力画像高さ及び出力チャネルの3重ループを計算できるようする．これにより起動回数をEMAXVの起動回数 ÷ 出力画像高さ ÷ 出力チャネルにできる．EMAXVRの起動回数とEMAXVの起動回数との比率を表10に示す．計算時間は，起動のためのオーバヘッド時間をt_o，LMMと外部メモリ間の転送の時間をt_m及び実際に計算を実行している時間をt_eとすると以下の式で表すことができる．

$$t = t_o + t_m + t_e$$

起動回数をNとすると1回起動する毎のオーバヘッド時間t'_oは，

$$t'_o = t_o/N$$

となり，

$$t = t'_o \times N + t_m + t_e$$

になる．この式からEMAXVRの計算時間を見積もる．EMAXVRの起動回数をN_{EMAXVR}とし，EMAXVRの各時間がEMAXVと同じであると仮定するとEMAXVRの計算時間は以下のようになる．

$$t_{EMAXVR} = t'_{oEMAXV} \times N_{EMAXVR} + t_{mEMAXV} + t_{eEMAXV}$$

EMAXV の起動回数を N_{EMAXV} として，さらに式を変形すると

$$t_{EMAXVR} = t_{oEMAXV} \times \frac{N_{EMAXVR}}{N_{EMAXV}} + t_{mEMAXV} + t_{eEMAXV}$$

となる．t_{oEMAXV} は，図 8 の LMMI，REGV 及び PRE_POST_EXEC を足したもので全体の 76〜88 %であり，$\frac{N_{EMAXVR}}{N_{EMAXV}}$ は，表 10 より，1000 分の 1 以下である．そのため，$t_{oEMAXV} \times \frac{N_{EMAXVR}}{N_{EMAXV}}$ は，全体の 0.1 %以下で無視できる値になることから，以下の式になる．

$$t_{EMAXVR} = t_{mEMAXV} + t_{eEMAXV}$$

これに予備評価の結果である図 8 の値を当てはめると，EMAXV と比較して，C1 層は 100/24 で 4.2 倍，C4 層は 100/16.3 で 6.1 倍，C7 層は 100/12.6 で 7.9 倍，C8 層は 100/12.9 で 7.8 倍及び C9 層は 100/13.2 で 7.6 倍だけ高速に計算できることになる．その結果，計算時間は C1 層は 7.5msec，C4 層は 24.1msec，C7 層は 17.1msec，C8 層は 13.2msec 及び C9 層は 9.1msec で，合計は 71.0msec となり，1 秒間に 14.1 枚の画像を計算できる．また，PLOAD と PDRAIN を使用して t_{mEMAXV} を t_{eEMAXV} の時間に隠蔽することでさらなる高速化を目指す．

次に，外部メモリの通信量が多い課題について，スクラッチパッドメモリの追加と LMM マルチディレクションブロードキャストを導入し，通信量を削減する．通信量を増やす要因の 1 つ目は部分和を LMM に格納しきれず，外部メモリへの退避と LMM への復旧を繰り返すためである．部分和は 1 枚の入力画像に対し，出力画像サイズ × 出力チャネル数 分のサイズであり，入力画像サイズと出力画像サイズが同じ場合は，入力画像サイズに対して最大で出力チャネル数倍のサイズになる．部分和を収めるサイズの LMM を搭載することも可能であるが，全ての PE に搭載する LMM サイズが増加するため，回路規模が増大してしまう．並列度を上げるために，入力画像は多数の LMM に格納する必要があるが，部分和を格納する LMM は計算結果であるため少なくすることができる．予備評価の結果，使用した命令マップの場合，C1 層は入力画像が格納される LMM が 66 個，部分和が格納される LMM は 1 個，同様に C4 層は入力画像用が 60 個，部分和用が 1 個，及び C7–C9 層は入力画像用が 96 個，部分和用が 1 個であった．そのため，部分和を収めることができるサイズの LMM とした場合，部分和を収める LMM

表 10: AlexNet 計算時の EMAXVR の起動回数と EMAXV の起動回数との比率

Layer	Start count	EMAXV / EMAXVR
C1	3	5280
C4	24	3456
C7	16	4992
C8	24	2496
C9	24	1664

は搭載されているメモリをほとんど使用することになるが，数十倍多く使用することになる入力画像を格納する LMM のメモリは，出力チャネル数分の 1 程度しか使用しておらず，搭載しているメモリを有効利用できないことになる．こうした理由から，LMM を大きくするのではなく，EMAXV の LMM と外部メモリというメモリ構成に，中規模サイズのメモリのスクラッチパッドメモリを追加し，LMM，スクラッチパッドメモリ及び外部メモリという 3 階層構成に変更する．さらに，部分和の退避先を外部メモリからスクラッチパッドメモリに変更することで，部分和の退避と復旧のための外部メモリ通信を無くす．通信量を増やす要因の 2 つ目は入力画像の同じデータを LMM に格納するためである．これに対し，CGRA 部の行方向にもブロードキャストできるようにする．そして，予備評価では，カーネル 1 行分の入力画像を都度 LMM に格納する方法を用いたが，計算で使用する入力画像を全て LMM に格納するように変更する．このようにすると例えば AlexNet の C7–C9 層の場合は，3 行 2 列の PE の LMM に同じデータを格納することになるものの，行方向にもブロードキャストできるため，1 度の通信で全て LMM にデータを格納できる．LMM に全ての入力画像が格納できない場合は，追加したスクラッチパッドメモリに一度入力画像データを格納し，スクラッチパッドメモリから LMM へ読み出すようにすることで，外部メモリから複数回読み出さないようにする．こうすることで，入力画像の同じデータを外部メモリから読み出さないようにする．

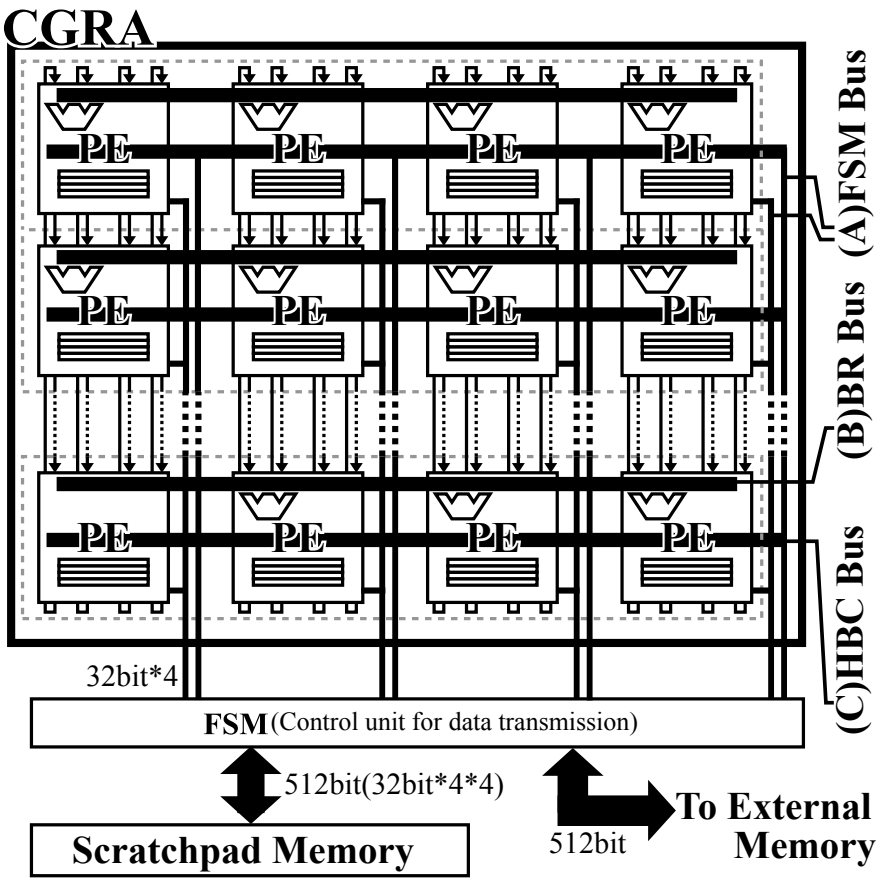


図 9: EMAXVR のブロック図

4.1 全体構成

図 9 に，EMAXVR のブロック図を示す．EMAXVR は，EMAXV と同様の CGRA 部と FSM 部にスクラッチパッドメモリを加えた 3 つのブロックから構成される．PE の配置やバス接続は EMAXV と同じである．FSM 部は，CGRA 部と外部メモリ及びスクラッチパッドメモリを接続するインターフェースと，システム全体のコントロールを行う機能を有する．新たに搭載したスクラッチパッドメモリは，ローカルメモリのサイズを超えるデータを格納するための中規模メモリとして利用する．

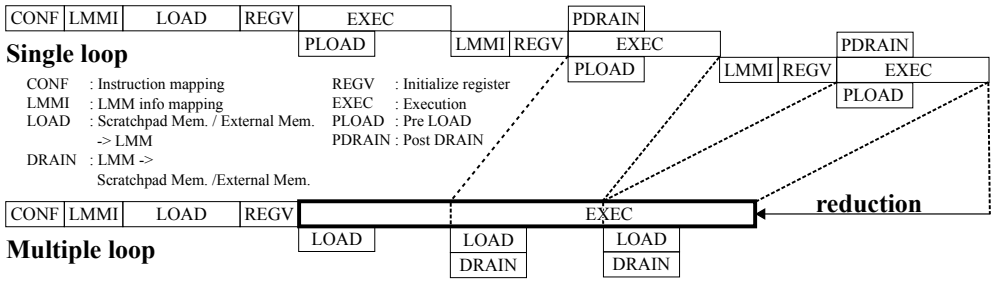


図 10: 多重ループを実行する EMAXVR の状態遷移

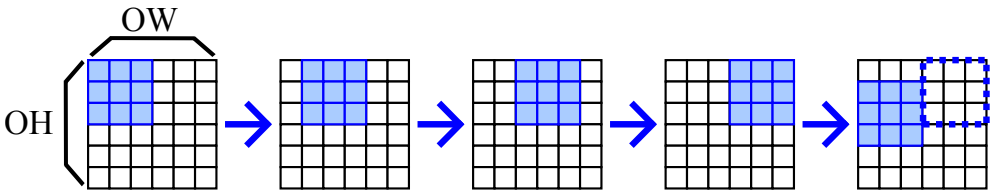


図 11: 多重ループ時の参照パタン

4.2 多重ループ制御

畳み込み演算は図 6(a) に示した通り多重ループ構造である．KH と KW のループを展開して 1 回起動する毎に OW 回計算し，これを OH × OC × IC 回起動する計算方法を [19] で提案した．EMAXV で行った予備評価でも同様の方法計算を行った．しかし，近年の画像認識で使用する CNNs では，入力に近い層では OC と IC は小さいものの，OW と OH は大きく，出力に近い層では OW と OH は小さいものの，OC と IC が大きく，全体を通して OH × OC × IC は大きな値になる．このため，EMAXV では全ての計算を行うために何度も起動しなければならなかった．EMAXV では，起動する度に，LMMI 状態における LMM と外部メモリ間データ転送のための情報の受け取り，また，REGV 状態における内部レジスタの初期化及び EXEC 状態でもパイプラインを埋める必要があるため，起動回数が多くなるとオーバヘッドが増加し，性能低下を招く．特に出力に近い層では，OW $\ll$ (OH × OC × IC) となり，起動毎の計算量が少なくなり，起動回数が多くなる．計算時間 (EXEC) に対して，オーバヘッドとなる LMMI，REGV 及び PRE_POST_EXEC の時間が増加する．そこで，EMAXVR では，図 6(b) に示すように，1 回の起動で最大 3 重ループを計算できるよう改良した．多重ループ実行

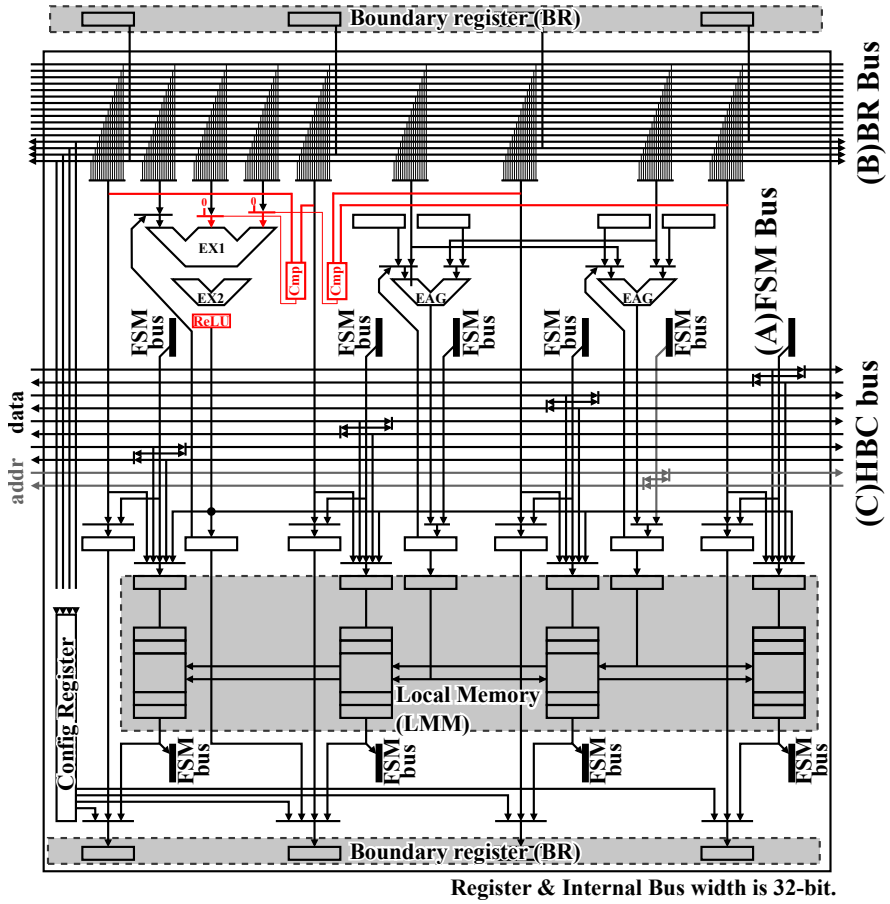
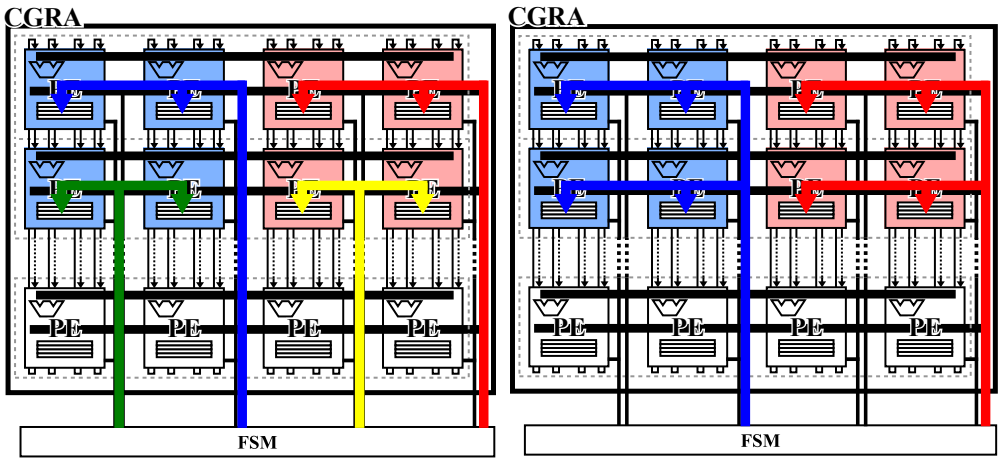

Boundary register (BR)
(B)BR Bus
(A)FSM Bus
(C)HBC bus
data
addr
EX1
EX2
ReLU
Cmp
Cmp
EAG
EAG
FSM bus
Config Register
Local Memory (LMM)
Boundary register (BR)
Register & Internal Bus width is 32-bit.

図 12: 改良した PE



(a) Horizontal only　　(b) Horizontal & vertical

図 13: マルチディレクションブロードキャスト．同じ色の LMM に対する同一データの転送．

時の状態遷移を図 10 に示す．多重ループが必要とするデータを全て LMM に格納することはできないため，ループの動きに合わせて LOAD と DRAIN を実行し，データを入替えながら計算する．データ入替え機能を実現するために，LOAD 又は DRAIN を実行するタイミング (どのループが終了する度に実行するか) と，実行する度に加えるアドレスのオフセットを設定できるよう LMMI を拡張した．また，FSM も 3 重ループの動きに合わせて，データの入替えを行うよう改良を加えた．多重ループに対応するためには，図 11 に示すような，内側ループ終了時の LMM に対する非連続なアドレス参照に対応しなければならない．そこで，ループの状態に合わせて (最終か最終以外) 加算する値を切り替える条件付き加算命令を追加した．本命令に対応するために，改良した PE を図 12 に示す．PE には，比較器及び比較結果によって入力を切り替える機構を追加した．本命令で計算した値を使用して LMM 参照アドレスを切り替えることで 3 重ループに対応した．さらに，活性化関数 ReLU を実行できるユニットを演算器の最終段に追加した．

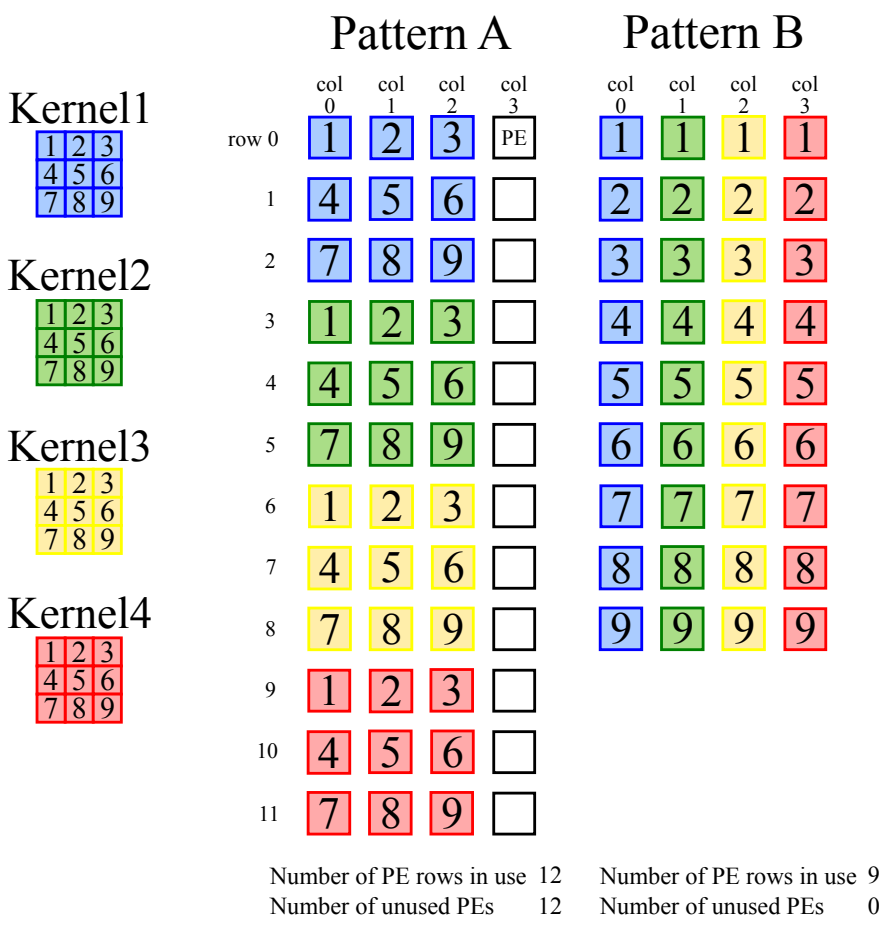


図 14: 命令マッピングパターン．Pattern A：カーネルの同じ行を，CGRA 部の同じ行に配置．Pattern B：一つのカーネルを 1 列に配置．

4.3 マルチディレクションブロードキャスト

予備評価の結果，複数行に同一の入力画像を転送する場合に，余分な通信が発生することに対して，EMAXV の水平方向ブロードキャストに加えて垂直方向ブロードキャストを可能とし，全てのカーネル列のためのメモリアクセス命令が配置される PE の LMM に計算で使用する入力画像を全て格納することで，余分な転送を無くすことを提案した．この転送方法は，入力画像を 1 行ずつ転送するものに比べて全ての行を 1 度に転送するため，バスのハンドシェイクやメモリアクセスの遅延も 1 度で済み転送効率が向上する．マルチディレクションブロードキャスト対応による効果を図 13 に示す．また，畳み込み演算を多重ループで計算するようにしたことで，入力画像と重みの両方を LMM に格納することになり，畳み込み演算のカーネルサイズが 4 を超える場合は，複数行を利用してカーネル 1 行分のメモリアクセス命令を配置なければならない．AlexNet の畳み込み層をシングルループで計算した時は，カーネルサイズが 11 の C1 層のみ同一の入力画像データを複数行の LMM に転送が必要であったが，カーネルサイズが 5 の C4 層でも複数行への転送が必要になる．しかし，垂直方向のブロードキャストを備えるため，この転送についても余分な通信を発生することがない．入力画像データと異なり，重みデータは要素毎に使用するデータが異なるため，LMM への転送を考慮したメモリ配置にしておくことで余分な通信をすることなく LMM へ転送できる．若しくは，異なる要素の重みデータも含めて LMM へ格納できる場合は，全て重みデータをメモリアクセス命令が配置される全ての LMM へブロードキャスト転送するようにすれば，転送効率を向上させることもできる．また，図 14 に示す通り，カーネルサイズが 4 を超えない場合でも，垂直方向に同一行のカーネル計算を配置するマッピングの方が，演算器利用率が高い場合がある．演算器利用率を上げることで計算時の並列度を上げることが可能になる．この配置を利用する場合でも，垂直方向ブロードキャストにも対応したことで，余分な通信をすることなく LMM へ転送することが可能である．垂直方向ブロードキャストに対応するにあたり，LMMI で転送される情報に垂直方向のブロードキャストの情報を加えた．FSM 部に備わっていた行毎にアクセスするためのイネーブル信号を，この情報を使って同時にアサートすることで垂直方向ブロードキャストを実現し

た．ブロードキャストするかどうかの判定は，命令配置から静的に判断することが可能である．命令配置からLMMI情報を生成するコンパイラを改良し，垂直方向に同一データのメモリアクセス命令があるかどうかを判定し，ある場合には垂直方向ブロードキャストを有効化できるようにした．

4.4 スクラッチパッドメモリ

予備評価の結果，部分和のLMMからの退避とLMMへの復旧による大量の外部メモリとの通信が発生することに対して，中規模サイズメモリを持つスクラッチパッドメモリを搭載することを提案した．部分和の入替えはスクラッチパッドメモリを使用することで，外部メモリとの通信はしないようにした．また，入力画像サイズが大きく，必要な行数分のデータをLMMに格納できない場合は，1行ずつLMMに格納し，入替えながら計算する．この時，一度LMMにロードしたデータを別のLMMに再ロードしなければならないことがある．外部メモリから同じデータを複数回転送することになり，余分な通信が増えてしまう．この場合も必要データをスクラッチパッドメモリに事前に格納しておき，スクラッチパッドメモリから転送することで，外部メモリとの余分なデータ転送をしないようにした．外部メモリとスクラッチパッドメモリは転送アドレスによって自動的に判定するようにした．そのため，メモリアクセス命令はどちらのメモリを使用する場合でも同一のものが使用できるようにした．

表 11: 各ハードウェアの評価環境

Hardware	Environment	Frequency	Data type
CPU (ARM Coretex-A9 4Core)	MCIMX6QP-SDB	1GHz	32bit Floating point
GPU (NVIDIA Tegra K1)	Jetson TK1	852MHz	32bit Floating point
EMAXVR (Target Process 28nm)	Simulator	240MHz	16bit Fixed point

5. 評価

提案通りに計算時間及び外部メモリの通信量が改善していることを確認するために，EMAXVR のサイクルベースシミュレータを開発し，計算時間と外部メモリの通信量を計測した．EMAXVR の状態毎の割合も測定し，問題のオーバヘッドの改善効果も確認した．比較のために，組み込み CPU と NVIDIA 社のモバイル向け GPU について，評価ボードを用い同様の演算を実行し計測した．各ハードウェアの評価環境を表 11 に示す．EMAXVR（ターゲットプロセス 28nm）の動作周波数を 240MHz とする根拠は，国内半導体メーカの 28nm HVT プロセスを使用して EMAXV（EMAXVR のベース）の配置配線を意識した論理合成（使用ツール：Synopsys 社 Design Compiler Graphical）した回路の動作周波数が 244MHz であること，TSMC 社 28nmHPC+を使用して EMAXVR が動作周波数 250MHz で論理合成（使用ツール：Synopsys 社 Design Compiler）できていることによる．EMAXVR のデータタイプは [13, 14] と同じ 16bit 固定小数点を採用した．CPU と GPU のデータタイプは単精度浮動小数点数である．評価で使用した CNNs は AlexNet[2] と VGG16[23] である．これら畳み込み演算の計算時間を各評価環境において測定した (表 12).

5.1 EMAXVR シミュレータ

評価で使用した EMAXVR シミュレータを説明する．EMAXVR シミュレータは，クロック毎の動作を再現するクロックアキュレートシミュレータである．シミュレータをベースに開発した Verilog-HDL（サイクル内の処理内容はシミュレータと一致しているもの）で論理合成し，目標とする動作周波数でタイミング収束

表 12: 評価した CNNs の構成. (C_x : 畳み込み層. P_x : プーリング層. Kernel : C_x : 入力チャネル数 × 出力チャネル数 × グループ数@カーネル幅 × カーネル高さ, P_x:カーネル幅 × カーネル高さ. Layer Size : 出力チャネル数@出力画像幅 × 出力画像高さ)

Network	Layer	Kernel	Layer Size
AlexNet	Input		3@227 × 227
	C1	3 × 96 × 1@11 × 11	96@55 × 55
	P3	3 × 3	96@27 × 27
	C4	48 × 128 × 2@ 5 × 5	256@27 × 27
	P6	3 × 3	256@13 × 13
	C7	256 × 384 × 1@ 3 × 3	384@13 × 13
	C8	192 × 192 × 2@ 3 × 3	384@13 × 13
	C9	192 × 128 × 2@ 3 × 3	256@13 × 13
	P10	3 × 3	256@6 × 6
VGG16	Input		3@224 × 224
	C1	3 × 64 × 1@3 × 3	64@224 × 224
	C2	64 × 64 × 1@3 × 3	64@224 × 224
	P3	2 × 2	64@112 × 112
	C4	64 × 128 × 1@3 × 3	128@112 × 112
	C5	128 × 128 × 1@3 × 3	128@112 × 112
	P6	2 × 2	128@56 × 56
	C7	128 × 256 × 1@3 × 3	256@56 × 56
	C8	256 × 256 × 1@3 × 3	256@56 × 56
	C9	256 × 256 × 1@3 × 3	256@56 × 56
	P10	2 × 2	256@28 × 28
	C11	256 × 512 × 1@3 × 3	512@28 × 28
	C12	512 × 512 × 1@3 × 3	512@28 × 28
	C13	512 × 512 × 1@3 × 3	512@28 × 28
	P14	2 × 2	512@14 × 14
	C15	512 × 512 × 1@3 × 3	512@14 × 14
	C16	512 × 512 × 1@3 × 3	512@14 × 14
	C17	512 × 512 × 1@3 × 3	512@14 × 14
	P18	2 × 2	512@7 × 7

表 13: EMAXVR シミュレータのメモリモデル

Memory model	Memory type	Bus width [bit]	Frequency [MHz]	Bandwidth [MByte/s]	Read latency [EMAX clock=240MHz]
Scratchpad Memory	Single port SRAM	256	480	15360	1
External Memory	DDR3 2133 16bit x 4	64	2133	17064	15 (Asynchronous CDC (2clock), Delay of memory controller (10clock), CAS Latency (3clock))

できていることを確認している．
メモリモデルの帯域は表 13 に示す通りである．スクラッチパッドメモリのリードレイテンシは，SRAM を想定しているので 1 クロックとした．外部メモリのリードレイテンシは，メモリコントローラの想定が難しいため余裕のある数値にしている．ライトレイテンシは，アクセスするバスの帯域に対して両メモリ共に帯域を超えていないことから未考慮とした．
バスのモデルは，アクセス競合は未考慮とした．外部メモリ及びスクラッチパッドメモリと接続される FSM Bus の帯域は，15306MByte/s (128bit x 4 x 240MHz) と各メモリの帯域を越えていないこと，異なる FSM Bus が同じアドレスにアクセスしないようにプログラムすることで，アクセス競合は未考慮としている．レイテンシは，アドレスチャネルのハンドシェイク後，データチャネルのハンドシェイク行うモデルとし，EMAX の 2 クロック分とした．

5.2 CPU と GPU の実行方法

CPU は自身で開発しコンパイルした実行コードを用いて実行し，GPU は CNN フレームワークを使用して実行した．
CPU の実行コードは，CNNs で必要な層や活性化関数を実行するコードを独自に作成し，g++コンパイラ (Version:4.9.2, Option:-O3 -mfloat-abi=hard -mfpu=neon -fopenmp) により生成した．計測対象は，畳み込み層，プーリング層及び活性化関数 ReLU である．この中で高速化の工夫をした畳み込み層の実装方法について説明する．具体的には，OpenMP を用いた並列実行と NEON 命令による演算の高速化を行った．OpenMP では，多重ループとして記述する畳み込み演算の最外

表 14: カーネルの計算において使用した NEON 命令

kernel	vmulq _fp32	vaddq _fp32	vmul _fp32	vadd _fp32
11x11	36	36	0	2
5x5	0	0	15	16
3x3	3	3	0	2

ループに#pragma omp paralle for を設定し for ループを並列化している．NEON 命令には組み込み関数を使用し，カーネルサイズのループを展開してサイズ分の積和演算を実行するよう最内ループにおいて使用している．使用した NEON 命令は，単精度浮動小数点数を 4 並列計算する vmulq_fp32，vaddq_fp32 及び，2 並列計算する vmul_fp32, vadd_fp32 である．カーネルサイズ毎に使用した命令数を表 14 に示している．部分和との加算は 2 画素毎に vadd_fp32 命令によって行っている．

GPU の計算は CNN フレームワークの NVIDIA Caffe 0.12.2 を，JetPack2.3.1 (cuda6.5, cuDNN 2) の環境においてビルドし使用した．JetPack は, Jetson TK1 に対応する最新版，NVIDIA Caffe は cuDNN 2 に対応する最新版を選択した．

実行速度は，CPU では C 標準ライブリ gettimoofday 関数を用いて計測し，GPU では NVIDIA 製のプロファイリングツール (nvprof) を用いた．

5.3 CNNs に合わせた EMAXVR の構成

評価に用いた EMAXVR のパラメータを表 15 に示す．PE は 64 行 × 4 列の計 256 個，LMM は 1KB とした．外部メモリとの通信量の評価には，スクラッチパッドメモリのサイズを 128KB と 256KB の 2 種類を用いた．畳み込み演算に使用した命令を表 16 に示す．EX1 には，加算，乗算，積和命令，2 入力 MAX 選択命令，2 つの 16bit データを 32bit に結合する命令や条件付き加算命令 (AEX) を搭載した．AEX 命令を EX1 に搭載した理由は，加算器を従来から搭載している加算命令用の加算器と共用するためである．EX2 には右シフト命令を搭載し，活性化関

表 15: EMAXVR のパラメータ

Parameter	Value
Number of PEs	64 rows × 4 columns
Size of LMM	1KByte
Size of Scratchpad Memory	128, 256 KByte
Frequency	240 MHz

数 ReLU 命令も搭載した．LMM 参照命令として，16bit を読み出す LDHR 命令，連続 16bit×2 を読み出す LDDHR 命令，32bit をストアする STR 命令と 16bit をストアする STHR 命令を搭載した．LDDHR 命令は連続する 16bit のデータを異なるレジスタに読み出す命令である．この命令を使用することで，読み出された値を使用する算術命令などにおいて，どちらのレジスタの値を使用する場合でも，bit マスクや論理シフトを使用することなく，16bit の値として使用できる．EX1，EX2，ReLU，LMM 参照命令は全てパイプライン処理される．

5.4 AlexNet C1 層のマッピング

AlexNet における C1 層の畳み込み演算をマッピングした様子を図 15(a) に示す．第 1 行にはループ回数をカウントする加算命令 (最内ループから順に，出力画像幅，出力チャネル，出力画像高さ) をマップしている．第 2 行にはループ順に合わせて LMM の参照アドレスを計算するための AEX 命令をマップしている．AEX 命令の 3 つのパラメータは任意の値が設定可能である．ループの動きに合わせたアドレス参照ができるように，これらのパラメータを設定することで，C1 層のように入力画像のストライドの値が 4 の場合でも，AEX 命令でアドレスを計算することができる．第 3 行以降，カーネル幅分の 11 個の積和を 3 行使用してマップしている．入力データとカーネルのロードには計 22 命令必要であり，1 つの PE に最大 2 つのロード命令をマップできるため (LDDHR 命令は 1 つ)，積和命令と同じ 3 行でマップできる．ロード命令で取り出した値は，後続の PE しか使用できないため，演算命令とロード命令は 1 行ずらしてマップしている．こ

表 16: 実装した命令 (追加した命令は **太字**)

Unit	Inst.	Description
EX1	ADD	32bit integer A + B
	MUL	16bit integer A x B
	MAD	16bit integer A x B + 32bit integer C
	MAX	16bit integer (A $\geq$ B) ? A : B
	CCAT	concatenate (16bit,16bit $\rightarrow$ 32bit)
	AEX	conditional addition
		32bit integer A
		(first loop == end) ? + B : + 0
		(second loop == end) ? + C : + 0
EX2	SAR	arithmetic shift to right
ReLU	**ReLU**	Rectified Linear Unit
Memory	LDHR	Load 16bit from LMM
	LDDHR	Load 16bit x2 from LMM
	STR	Store 32bit to LMM
	STHR	Store 16bit to LMM

表 17: EMAXVR の状態毎の FSM Bus の使途

(a) kernel 11x11

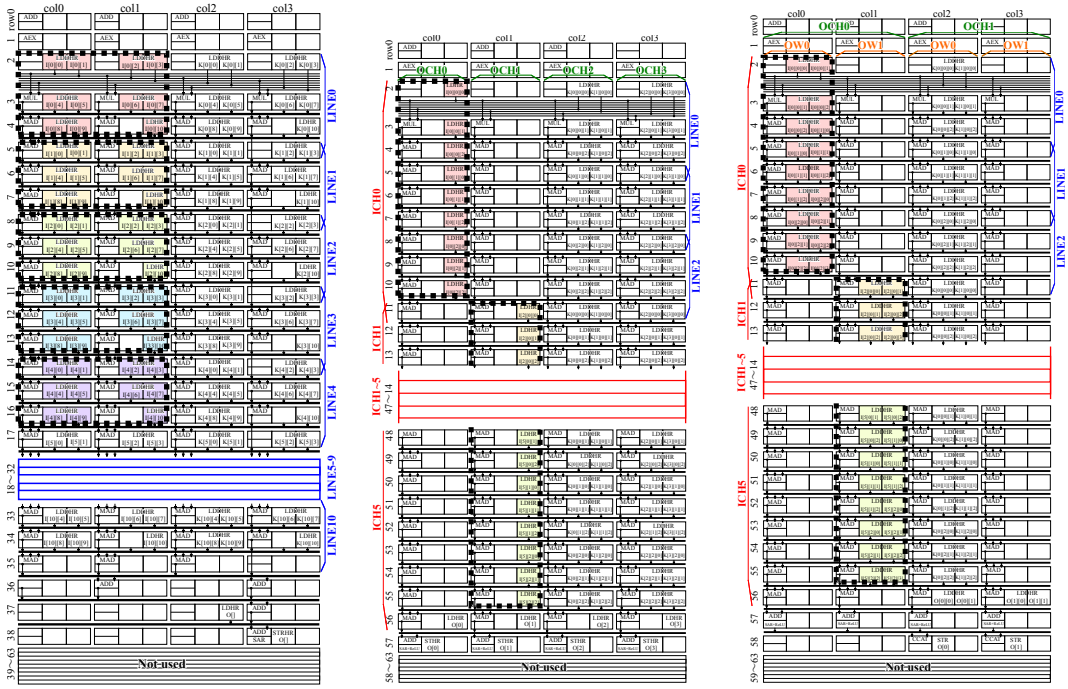
State of EMAXVR	FSM Bus[0]	FSM Bus[1]	FSM Bus[2]	FSM Bus[3]
LOAD	Input	Input	Kernel, Bias / Partial Sum	Kernel
DRAIN				Output
EXEC	Input	Input	Partial Sum	Output

(b) kernel 3x3(1)

State of EMAXVR	FSM Bus[0]	FSM Bus[1]	FSM Bus[2]	FSM Bus[3]
LOAD	Input, Partial Sum	Input, Partial Sum	Kernel, Bias / Partial Sum	Kernel, Bias / Partial Sum
DRAIN	Output	Output	Output	Output
EXEC	Partial Sum, Output	Partial Sum, Output	Partial Sum, Output	Partial Sum, Output

(c) kernel 3x3(2)

State of EMAXVR	FSM Bus[0]	FSM Bus[1]	FSM Bus[2]	FSM Bus[3]
LOAD	Input, Partial Sum	Input, Partial Sum	Kernel, Bias / Partial Sum	Kernel, Partial Sum
DRAIN			Output	Output
EXEC	Input	Input	Partial Sum, Output	Partial Sum, Output



(a) kernel 11x11 (b) kernel 3x3(1) (c) kernel 3x3(2)

図 15: EMAXVR への畳み込み演算のマッピング

れをカーネルの高さ分 (11 セット) 並べている．積和命令を使用することで，加算も同時に行うことに加え，全ての積和結果を加算するための加算命令をマップし，カーネルサイズ分の 121 個の乗算と加算を並列化している．最下段の PE には，121 個の積和結果に，部分和又はバイアスの加算命令と 16bit にクリップするための右シフト，及び活性関数 ReLU 命令をマップしている．さらに結果を LMM に格納するストア命令も同一 PE へマップする．ただし，ReLU は最後の入力チャネルを計算する時のみマップし，それ以外ではマップしない．図 15 の点線枠内の PE は，LMM に同一データが転送されていることを示す．図 15(a) の通り，3 行 2 列の合計 6PE に，同一データの転送が必要であるが，従来から可能な水平方向ブロードキャストに加えて垂直方向ブロードキャストに対応したことで，1 回の送信により全 PE に転送できる．表 17(a) には，各 EMAXVR の状態における FSM Bus の使途を示している．この層では，LOAD 時は FSM Bus の第 1 列と第 2 列は入力データの取り込み，第 3 列はカーネルとバイアス又は部分和の取り込

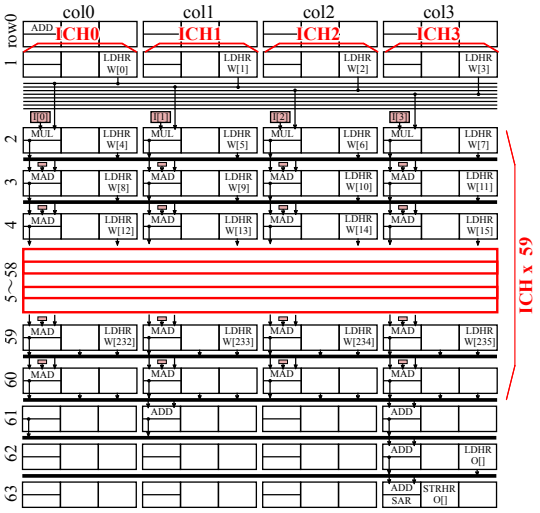


図 16: EMAXVR における全結合層のマッピング

み，第 4 列はカーネルの取り込みに使用する．DRAIN 時は，FSM Bus の第 4 列のみを出力の取り出しに使用する．EXEC 時は，FSM Bus の第 1 列と第 2 列は入力データの取り込み，第 3 列と第 4 列は部分和の取り込みと出力データの取り出しに使用する．EXEC 時に取り込みと取り出しを行っているのは，使用する入力データ，部分和，及び出力データそれぞれの合計サイズが LMM のサイズを超えており，計算中にデータを入替えるためである．入力データは，出力チャネルのループが終了する毎に入替え，部分和と出力データは，出力画像幅のループが終了する毎に入替える．

5.5 AlexNet C7–C9 層のマッピング

AlexNet における C7–C9 層の畳み込み演算をマッピングした様子を図 15(b) に示す．第 1 行にループ回数をカウントする加算命令 (最内ループから順に，出力画像幅，出力画像高さ，出力チャネル) をマップしている．第 2 行にループ順に合わせて LMM 参照アドレスを計算する AEX 命令，第 3 行以降，9 行 1 列分の PE を使用して，カーネルサイズ分の 9 個の積和命令をマップしている．各列には異なる出力チャネル 4 つ分の計算をマップしている．入力データは，出力チャネルが異なっても共通であるため，ロード命令を 9 行 1 列の PE にマップする．カーネル

は出力チャネル毎に異なるため，LDDHR 命令を 2 つマップし，4 種類のデータをロードしている．以上を異なる入力チャネル 6 セット分マップし，合計 216 個の乗算と加算を並列化している．最後に，出力チャネル毎の積和結果に，部分和又はバイアスの加算命令，16bit にクリップする右シフト命令及び活性関数 ReLU 命令をマップしている．ReLU を最後の入力チャネル計算時のみマップするのは C1 層と同様である．点線枠内の PE には，C1 層と同様同一データを転送する．表 17(b) には，各 EMAXVR の状態における FSM Bus の使途を示している．この層では，LOAD 時は FSM Bus の第 1 列と第 2 列の入力データと部分和の取り込み，第 3 列と第 4 列のカーネルとバイアス又は部分和の取り込みを行う．DRAIN 時は，全列の FSM Bus が出力の取り出しを行う．EXEC 時は，全列の FSM Bus が部分和の取り込みと出力の取り出しを行う．C1 層と異なり，計算に必要な入力は LMM に全て格納できるため，EXEC 時に取り込む必要はない．ただし，出力チャネルの並列化に伴い，部分和の取り込みと出力データの取り出しは 4 列全てに対して行う必要がある．このため，FSM Bus を 4 つ使用し，出力画像高さのループが終了する毎に部分和と出力データを入替えている．C4 層については示していないが，C7–C9 層と同様の考え方により 25 行 1 列分の PE を使用してカーネルサイズ分の積和命令をマップしている．

5.6 VGG16 C1–C2 層, C4–C5 層のマッピング

VGG16 における畳み込み層は，AlexNet における C7–C9 層と同様に，カーネル幅と高さが 3 である．ただし，入力側に近く入力画像のサイズが大きい C1-C2 層と，C4-C5 層は，入力データを全て LMM へ取り込むことができず，EXEC 時に入替える必要がある．入力データの入替えタイミング (どのループの終了時に取り込むか) と，部分和と出力データの入替えタイミングが異なるため，AlexNet の C7–C9 層とは異なるマッピングを採用した (図 15(c))．第 1 行にはループ回数をカウントする加算命令 (最内ループから順に，出力画像幅，出力チャネル，出力画像高さ) をマップした．第 2 行はループ順に合わせて LMM 参照アドレス計算の AEX 命令をマップしている．第 3 行以降，9 行 1 列分の PE を使用して，カーネルサイズ分の 9 個の積和命令をマップし，第 1–2 列と第 3–4 列に異なる出力チャ

ネル 2 つ分をマップし，同じ出力チャネルがマップされる第 1 列と第 2 列，及び第 3 列と第 4 列に，隣り合う出力画像 2 つ分の計算をマップする．入力データは，出力チャネルが異なっても共通であるため，ロード命令を 9 行 1 列の PE にマップする．隣り合う 2 画素分の入力データが必要なため，LDDHR 命令を使用している．カーネルは出力チャネル毎に異なるため，LDDHR 命令を 1 つマップし，2 種類のデータをロードしている．以上を異なる入力チャネルの 6 セット分をマップし，合計 216 個の乗算と加算を並列化している．入力チャネルが 6 に満たない場合は，6 以下の最大数をマップする．最終行直前に，出力チャネル毎の積算結果に，部分和又はバイアスの加算命令と 16bit にクリップする右シフト命令，及び活性関数 ReLU 命令をマップしている．最後行には 16bit データ 2 つを連結する CCAT 命令を配置し，同じ出力チャネルのデータを連結して 1 つの LMM へストアしている．これにより，4 つの計算結果を 2 つの LMM に格納できる．表 17(c) には，各 EMAXVR の状態における FSM Bus の使途を示している．この層では，LOAD 時は FSM Bus の第 1 列と第 2 列は入力データの取り込み，第 3 列はカーネルとバイアス又は部分和の取り込み，第 4 列はカーネルの取り込みに使用する．DRAIN 時は，第 3 列と第 4 列の FSM Bus を出力の取り出しに使用する．EXEC 時は，FSM Bus の第 1 列と第 2 列は入力データの取り込み，第 3 列と第 4 列は部分和の取り込みと出力データの取り出しに使用する．入力データは，出力チャネルのループが終了する毎に入替えを行い，部分和と出力データは出力画像幅のループが終了する毎に入替える．C1-C2 層と，C4-C5 層以外の畳み込み層については，AlexNet の C7–C9 層と同じマッピングである．

なお，AlexNet の P10 と，VGG16 の P3，P6，P10，P14，P18 に該当するプーリング層は，畳み込み層の直後にあるため，EMAXVR では畳み込みに連続して実行している．スクラッチパッドメモリに格納されている畳み込み層の結果を使用してプーリングし，外部メモリに転送するデータをプーリング後のサイズにして通信量を削減する．

5.7 全結合層のマッピング

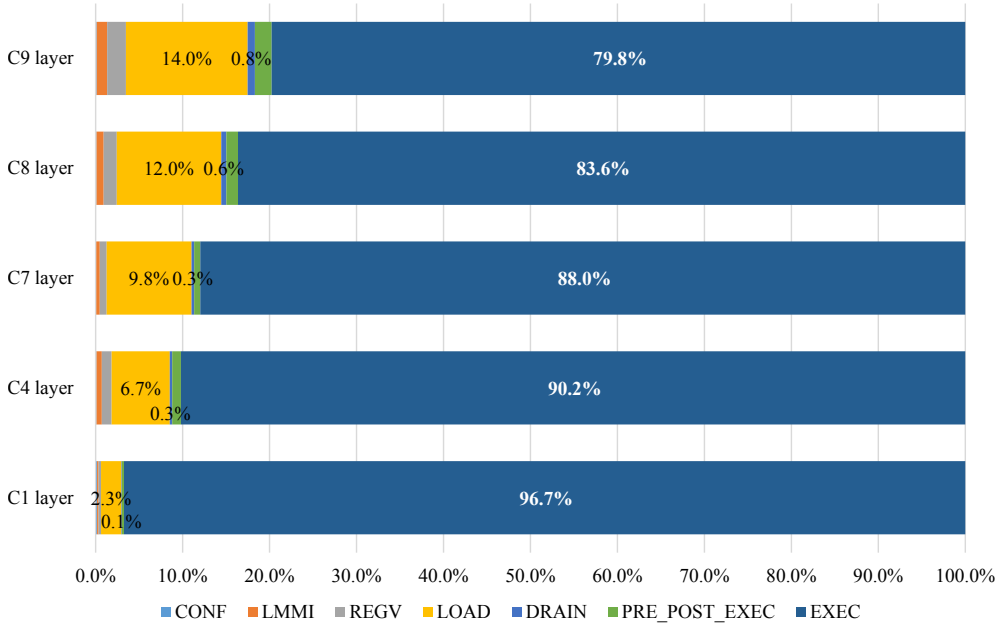
全結合層のマッピングを図 16 に示す．第 1 行にはループ回数をカウントする加算命令をマップしている．全結合層には，重みデータの再利用性が無く，計算で消費するデータ量が LMM の読み込み帯域を超えてしまい，計算中のデータ入替えができない．そのため，全結合層では畳み込み演算と異なり，出力チャネルの 1 重ループになっている．1 重ループの場合，AEX 命令は不要なので，2 行目以降から全結合層の演算をマップしている．1 行に 4 つの入力チャネルの積和命令と重みデータのロード命令を配置している．出力チャネルのループでは入力画像の値が不変のため，入力画像はレジスタに固定値を設定し，重みデータのみ LMM へ格納している．そのため，ロード命令は重みデータの分だけ配置して，これを搭載できる最大数の 59 セット分並べている．積和命令を使用することで加算も同時に行うことに加え，全ての積和結果を加算するための加算命令をマップし，入力チャネル 236 個分の乗算と加算を並列化している．最下段の PE には，236 個の積和結果に，部分和又はバイアスの加算命令と 16bit にクリップするための右シフト，及び活性関数 ReLU 命令をマップしている．さらに結果を LMM に格納するストア命令も，同一 PE へマップする．ただし，ReLU は最後の入力チャネルを計算する時のみマップし，それ以外ではマップしない．もちろん活性化関数が不要な場合は，マップしないことで ReLU 無しの全結合層の演算が可能である．全結合層では，各 LMM に格納されている値は全て異なるため，ブロードキャスト転送は行われない．LOAD 時は全ての FSM Bus を，重みデータの取り込みに使用し，第 4 列だけは重みデータに加えてバイアス又は部分和の取り込みに使用する．DRAIN 時は第 4 列の FSM Bus を使用して出力を取り出す．EXEC 時はデータの入替えを行わないため FSM Bus を使用したデータ転送は行わない．

6. 結果

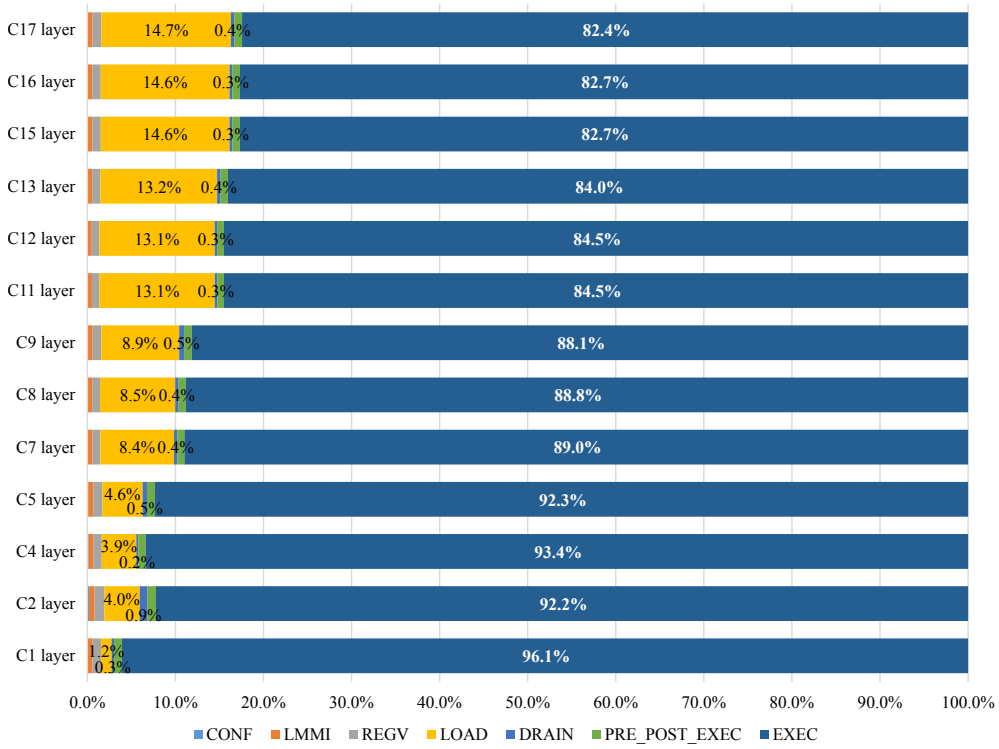
6.1 計算時間の比較

図 17 に，各 CNNs の畳み込み層を計算した時の計算時間に占める EMAXVR の状態の割合を示す．図 17(a) は AlexNet の畳み込み層計算時の EMAXVR の状態の割合である．LMM と外部メモリ間の転送時間（LOAD と DRAIN）の割合と実際に計算を行っている時間（EXEC）の割合を足した値は，C1 層が 99 %，C4 層が 97 %，C7 層が 98 %，C8 層が 96 %及び C9 層が 95 %とオーバヘッドは 5 %以下に抑えられており，多重ループ化による削減効果が確認できた．DRAIN は全ての層で 1 %以下であり PDRAIN により隠蔽できている．LOAD は，予備評価時は EXEC と同等若しくは 2 倍程度の時間を占めていた．EMAXVR では，多重ループ化により重みデータも LMM へ格納する必要があるためシングルループ時から転送に必要な時間は増している．しかしながら，LOAD の時間は，C1 層が EXEC の 2 %，同様に C4 層が 7 %，C7 層が 11 %，C8 層が 14 %及び C9 層が 18 %であり，PLOAD による隠蔽ができている．層が進むにつれて大きくなっているのは，計算量に対する必要なデータ量が増えているためである．VGG16 についても図 17(b) に示す通り，オーバヘッドは全ての層で 3 %以下であり，オーバヘッドによる性能低下は起きていない．DRAIN も 1 %であり，LOAD についても最大で EXEC の 18 %であり，PDRAIN と PLOAD による隠蔽ができている．

図 18 に，CPU，GPU 及び EMAXVR の各 CNNs の畳み込み層とプーリング層の計算時間を示す．AlexNet の計算時間は，CPU 2Core，CPU 4Core，GPU と EMAXVR で，それぞれ，1.05 秒，0.55 秒，0.023 秒と 0.017 秒であり，おおよそ CPU 2Core の 62 倍，CPU 4Core の 32 倍，GPU の 1.35 倍の速度で計算できた (図 18(a))．VGG16 の計算時間は同様に，それぞれ，25.8 秒，13.2 秒，0.37 秒と 0.35 秒であり，CPU 2Core の 74 倍，CPU 4Core の 38 倍，GPU の 1.06 倍の速度で計算できた (図 18(b))．GPU とは同等の計算性能であるが，GPU は EMAXVR と比較して周波数が 3.6 倍，演算器数は 0.75 倍で，理論最大性能としては 2.7 倍である．このことから EMAXVR が 2.7 倍高い効率で計算できている．



(a) AlexNet



(b) VGG16

図 17: 計算時間に占める EMAXVR の状態の割合

表 18: 演算器利用率

(a) AlexNet

Layer	Utilization[%]	
	EMAXVR	EMAXV
C1	52	49
C4	82	41
C7	89	58
C8	89	58
C9+P10	89	58
Average	80	53

(b) VGG16

Layer	Utilization [%]
C1	47
C2+P3	87
C4	87
C5+P6	88
C7	88
C8	88
C9+P10	88
C11	88
C12	88
C13+P14	88
C15	88
C16	88
C17+P18	88
Average	88

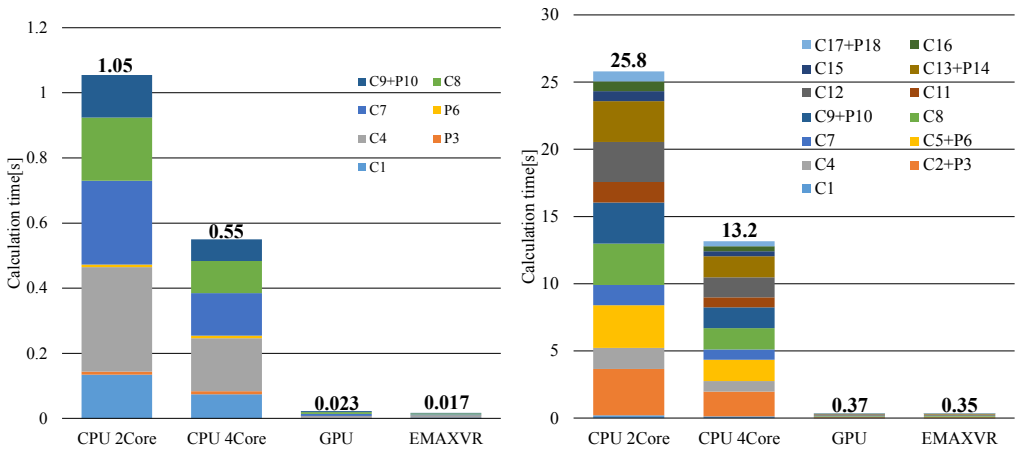
表 19: 外部メモリ通信量. (128K と 256K はスクラッチパッドメモリをサイズの表している．)

(a) AlexNet

Layer	Transaction [MByte]	
	128K	256K
C1	1.34	1.09
C4	1.68	1.08
C7	1.90	1.77
C8	1.52	1.27
C9+P10	0.99	0.86
Sum	7.43	6.10

(b) VGG16

Layer	Transaction [MByte]	
	128K	256K
C1	6.6	6.49
C2+P3	14.7	11.2
C4	9.3	7.0
C5+P6	13.2	8.5
C7	10.5	6.6
C8	19.5	11.5
C9+P10	18.4	10.2
C11	17.1	10.2
C12	33.4	19.7
C13+P14	32.8	19.1
C15	9.42	4.88
C16	9.42	4.88
C17+P18	9.28	4.74
Sum	203.7	125.1



(a) AlexNet (b) VGG16

図 18: 計算時間

表 20: 既発表アクセラレータとの比較

	EMXVR	Eyeriss[13]	FlexFlow[14]
Number of PEs	256	168	256
Local Memory / PE	1024 Byte	512 Byte	512 Byte
Scratchpad Memory	128, 256 KByte	108 KByte	64 KByte
Utilization of ALU (AlexNet)	80 %	88%	97%
Utilization of ALU (VGG16)	88 %	94%	100%
DRAM-Access / Op (AlexNet)	0.0059, 0.0048	0.006	0.0049
DRAM-Access / Op (VGG16)	0.0070, 0.0043	0.008	-

6.2 演算器利用率と外部メモリ通信量の比較

表 18 に，各 CNNs の層毎の演算器利用率を示す．AlexNet が 80 %，VGG16 が 88 %となった．表 18(a) には，予備評価で使用した命令マップの演算器利用率も示している．提案した命令マップを使用することで，利用率が平均で 53 %から 80 %まで向上した．表 20 に示す通り，DSA の利用率と比較すると，Eyeriss[13] に対しては 9 %と 6 %の低下と，10 %以下の差に収まった．FlexFlow[14] に対しては，18 %と 12 %の低下となった．表 19 に，各 CNNs の層毎の外部メモリ通信量を示す．AlexNet では，スクラッチパッドメモリのサイズが 128 KByte の時に 7.43 MByte，256 KByte の時に 6.10 MByte であった．VGG16 では，スクラッチパッドメモリのサイズが 128 KByte の時に 203.7 MByte, 256 KByte の時に 125.1 MByte となった．スクラッチパッドメモリのサイズを 128 KByte から 256 KByte に増やすことで，AlexNet では 18 %，VGG16 では 39 %通信量を削減できた．各 DSA と演算数で正規化した値で比較すると，Eyeriss には，スクラッチパッドメモリが 128 KByte と 256 KByte のいずれでも勝ることができ，FlexFlow には 256 KByte により勝ることができた．

6.3 スクラッチパッドメモリによる電力削減効果

スクラッチパッドメモリ追加による電力削減効果について説明する．電力削減効果の見積りに使用した外部メモリの仕様は以下の通りである．

- メモリ: Micron 1Gb DDR3L SDRAM with 16 DQs and a -093 Speed Grade
- 動作周波数: 1066MHz
- 動作電圧: 1.35V
- Read 時の消費電力: 262.3mW
- Write 時の消費電力: 546.6mW

DDR3 を 64bit 幅にするため，これを 4 つ使用した条件にて見積りを行った．消費電力は Activate や Refresh などで消費する電力は含まず，Read/Write で実行

時に消費する電力のみである．消費電力の見積りは，Micron 社が提供している DDR3_DDR3L_Power_Calc.xlsm を使用した．表 21 に，スクラッチパッドメモリを追加したことと，メモリサイズを 128KByte から 256KByte に拡張したことで削減できた外部メモリとの通信量及び通信時間と，それにより削減できた電力量を示している．削減できた通信量は，スクラッチパッドメモリの通信量を EMAXVR シミュレータにて計測した．通信時間は，削減できた通信量を通信するために必要なデータ転送の時間を見積りに使用した外部メモリの条件下で算出した．各ネットワークの計算に必要な時間 (図 18) から削減できた通信時間の割合を算出し，外部メモリの消費電力に掛け合わせることで，電力削減量を算出した．電力削減量を算出するには，スクラッチパッドメモリの通信のための消費電力やリーク電力を考慮する必要があるが，TSMC28nmHPC+の SRAM の仕様を確認し，未考慮でも問題ないことを確認している．

スクラッチパッドメモリの追加によって，AlexNet では 264mW，VGG16 では 318mW の電力削減効果が見込めることを確認した．組み込みシステムの搭載に向けて，1W〜数 W 以下の消費電力が期待されていると考えており，この削減効果は，20〜30 %程度の電力削減効果と非常に大きなものであり，チップ面積を増やしてでも追加すべきと考える．一方で，DSA 以下の通信量を目指すためにスクラッチパッドメモリを 128KByte から 256KByte に増やしたことによる削減量は，AlexNet では 7mW，VGG16 では 15mW であった．これはスクラッチパッドメモリの追加による効果の 20 分の 1 と非常に少ない削減量であることから，電力削減のみの目的にサイズを拡張することは難しいと言える．

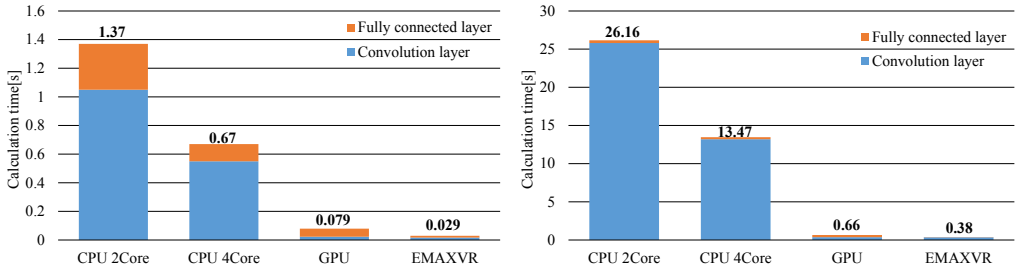
6.4 全結合層も含めた計算時間

図 19 に，CPU，GPU 及び EMAXVR の各 CNNs の畳み込み層，プーリング層と全結合層の計算時間を示す．全結合層の時間は EMAXVR のみ机上計算で算出した．CPU と GPU は畳み込み層とプーリング層を計測したものと同じ環境で計測した．AlexNet の計算時間は，CPU 2Core，CPU 4Core，GPU と EMAXVR で，それぞれ，1.37 秒，0.67 秒，0.079 秒と 0.029 秒であり，おおよそ CPU 2Core の 46 倍，CPU 4Core の 23 倍，GPU の 2.7 倍の速度で計算できた (図 19(a))．VGG16 の

表 21: スクラッチパッドメモリによる電力削減量

Addition, Expansion	Network	Reduced transaction [MByte]		Reduced transaction time (Data transfer only) [s]		Reduced power [mW]
		Read	Write	Read	Write	
Addition of memory	AlexNet	24.5	21.0	0.0015	0.0013	264
	VGG16	573.8	539.8	0.035	0.033	318
Expansion of memory	AlexNet	1.1	0.24	0.00007	0.00002	7
	VGG16	79.0	0	0.0005	0	15

計算時間は同様に，それぞれ，26.16 秒，13.47 秒，0.66 秒と 0.38 秒であり，CPU 2Core の 70 倍，CPU 4Core の 36 倍，GPU の 1.7 倍の速度で計算できた (図 19(b))．全結合層を含めた場合も，AlexNet と VGG16 共に CPU や GPU より高速に計算できている．しかし，1 秒あたりの積和演算できる回数 (OPS (Operation Per Sec)) は，畳み込み層やプーリング層と比較して，全結合層は低下している．それぞれ，AlexNet は畳み込み層とプーリング層は 39.2GOPS で全結合層は 4.8GOPS であり，VGG16 は畳み込み層とプーリング層は 43.8GOPS で全結合層は 4.9GOPS である．この理由は，全結合層の計算は畳み込み層の計算と比べて演算強度が低いためである．EMAXVR の状態別の時間の割合でも図に示すように，LOAD が計算時間の 80 %以上を占めている．そのため，ネットワークを変更せずに全結合層の計算を高速にするには，より低い bit に量子化して演算強度を上げたり [30]，データ転送の帯域を広げるといった畳み込み層の高速化で用いたものとは異なる手法が必要である．



(a) AlexNet (b) VGG16

図 19: 全結合層も含めた計算時間

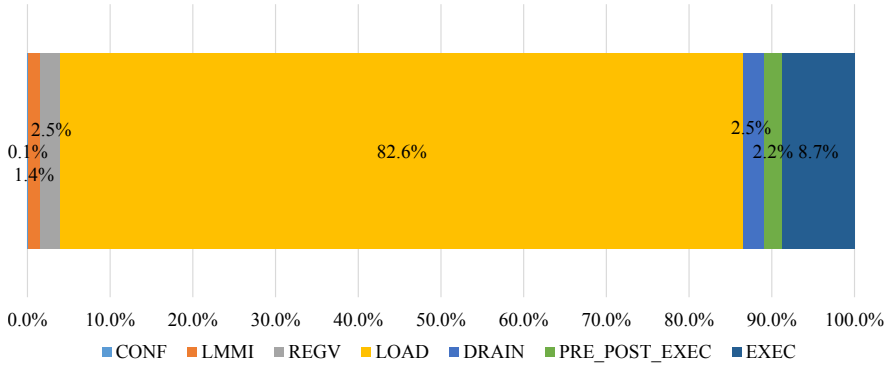


図 20: 全結合層の計算時間に占める EMAXV の状態別の割合

7. 結論

高精度な画像認識を実現することが可能な CNNs を組み込みシステムへ搭載するために，高速に計算するアクセラレータが必要とされている．本稿では，ローカルメモリを備える CGRA を採用したプログラマブルアクセラレータを CNNs にも効率よく適用できる EMAXVR を提案した．様々なステンシル計算におけるプログラマビリティを有し，従来の組み込みシステムに搭載されている CPU と比較して数十倍以上，NVIDIA 社のモバイル向け GPU と比較しても同等以上の CNNs 計算性能を発揮し，DSA の特徴である高い演算器利用率と低い外部メモリ通信量に迫れることを確認した．

謝辞

本研究を進めるにあたり，多くの方々にご指導，御協力，ご支援を頂きました．ここに誠意を添えてお名前を記させて頂きます．

奈良先端科学技術大学院大学 情報科学研究科 コンピューティングアーキテクチャ研究室 中島 康彦 教授には，本研究の全過程において熱心な御指導を賜りました．研究方針だけでなく，論文執筆や発表方法についても多くのご助言を頂きました．心より厚くお礼申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 ディペンダブル研究室 井上 美智子 教授には，本研究に対して貴重な御指導，御助言を賜りました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 コンピューティングアーキテクチャ研究室 中田 尚 准教授には，本研究ならびに論文の執筆において多大なる御指導，御助言を賜りました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 コンピューティングアーキテクチャ研究室 トラン・ティ・ホン助教には，本論文審査において貴重な御指導，御助言を賜りました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 コンピューティングアーキテクチャ研究室 張 任遠 助教には，英語論文の執筆及び国際会議の発表において多大なる御指導，ご助言を賜りました．心より感謝申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 コンピューティングアーキテクチャ研究室 OB の山野 龍佑氏, 福岡久和氏及び菊谷 雄真氏には，本研究において多大なる協力を頂きました．心より感謝申し上げます．また，同研究室で共に研究させて頂いた多くの OB やメンバーの皆様にも感謝申し上げます．

株式会社ベリフォア 久保田 雄三取締役及び宝 純氏には，EMAXVR のシミュレータ開発において多大なる御協力を頂きました．心より感謝申し上げます．

コニカミノルタ株式会社 IoT サービス PF 開発統括部 江口 俊哉 統括部長，同戦略推進部 田中 正浩 部長，同アーキテクチャ開発部 岸 恵一部長ならびに同エッジコントローラ開発部 木村 和伸グループリーダーには，社会人博士として，本学に入学し研究させて頂くという貴重な機会を与えて頂きました．また，要所要

所で貴重な御指導，ご助言を賜りました．心より感謝申し上げます．

コニカミノルタ株式会社 IoT サービス PF 開発統括部 エッジコントローラ開発部 内野 浩志氏，藤澤 慎也氏，戸川 智史氏，藤崎 修一氏，西上 直孝氏及び同データサイエンス技術部 矢口 義孝氏には，業務との両立において多大なるご配慮を頂くだけでなく，研究への御協力及び様々な場面で貴重な御指導，御助言を賜りました．心より感謝申し上げます．

最後に筆者の社会人としての会社での業務遂行と，博士後期課程の学生としての研究活動に対し，快く理解を示し支えてくれた妻 恵美，長男 想士と次男 優士に深く感謝致します．

参考文献

[1] Stanford Vision Lab. Imagenet large scale visual recognition challenge (ilsvrc).

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, USA, 2012. Curran Associates Inc.

[3] 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)*, pp. 770–778, June 2016.

[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, Vol. 115, No. 3, pp. 211–252, 2015.

[5] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, June 2014.

[6] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1653–1660, June 2014.

[7] Z. Cao, T. Simon, S. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1302–1310, July 2017.

[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox

detector. 2016. To appear.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, June 2016.

[10] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 12, pp. 2481–2495, Dec 2017.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, Oct 2017.

[12] NXP. i.mx6quadplus processor. `https://www.nxp.com/docs/en/data-sheet/IMX6DQPIEC.pdf`.

[13] Y. Chen, T. Krishna, J. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 262–263, Jan 2016.

[14] 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)*, pp. 553–564, Feb 2017.

[15] Joo M.P. Cardoso and Pedro C. Diniz. *Compilation Techniques for Reconfigurable Architectures*. Springer US, 2009.

[16] Yoonjin Kim and Rabi N. Mahapatra. *Design of Low-Power Coarse-Grained Reconfigurable Architectures*. CRC Press, 2010.

[17] 清水怜, 田ノ元正和, 高前田 (山崎) 伸也, 姚駿, 中島康彦. メモリネットワークベースアクセラレータの試作と評価. 信学技法 CPSY2014-81, pp. 51–56, 2014.

[18] Y. Inagaki, S. Takamaeda-Yamazaki, J. Yao, and Y. Nakashima. Performance evaluation of a 3d-stencil library for distributed memory array accelerators. *IEICE TRANSACTIONS on Information and Systems*, Vol. E98-D, No. 12, pp. 2141–2149, Dec 2015.

[19] M. Tanomoto, S. Takamaeda-Yamazaki, J. Yao, and Y. Nakashima. A cgra-based approach for accelerating convolutional neural networks. In *2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip*, pp. 73–80, Sept 2015.

[20] 田ノ元正和, 高前田 (山崎) 伸也, 姚駿, 中島康彦. メモリネットワークベースアクセラレータを用いた畳み込みニューラルネットワーク処理. 信学技法 CPSY2014-82, pp. 57–62, 2014.

[21] 清水怜, 高前田 (山崎) 伸也, 姚駿, 中島康彦. メモリインテンシブアレイアクセラレータを用いた高性能グラフ処理. 信学技報 CPSY2014-11, pp. 7–12, 2014.

[22] 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)*, pp. 174–180, Nov 2016.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, Vol. abs/1409.1556, pp. 1–14, 2014.

[24] 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)*, pp. 1–9, June 2015.

[25] NVIDIA Corporation. Nvidia tegra x1. https://www.nvidia.com/object/tegra-x1-processor.html.

[26] NVIDIA Corporation. Nvidia tegra k1. https://www.nvidia.com/object/tegra-k1-processor.html.

[27] NVIDIA Corporation. Nvidia's fermi: The first complete gpu computing architecture. https://www.nvidia.com/content/PDF/fermi_white_papers/P.Glaskowsky_NVIDIA's_Fermi-The_First_Complete_GPU_Architecture.pdf.

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

[29] 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)*, pp. 56–65, Sept 2013.

[30] 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. *ACM SIGARCH Computer Architecture News*, Vol. 44, No. 3, pp. 243–254, June 2016.

付録

A. 業績

査読付学術論文

[1] 一倉孝宏, 菊谷雄真, 中島康彦: ”DSA 並みの効率を達成する CNNs 拡張機能付き CGRA の提案と評価”, 電子情報通信学会論文誌,Vol.J102-D,No.07,pp.-,Jul. 2019.(第 4-6 章に対応する)

査読付国際会議発表

[1] 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 and Systems, COOL Chips 21, 18-20 April 2018

国内研究会

[1] 山野龍佑, 福岡久和, 一倉孝宏, 中島康彦: ”低電力姿勢認識のための CGRA プロトタイプシステム”, 電子情報通信学会コンピュータシステム研究会 (CPSY) ポスター, 電子情報通信学会技術研究報告 CPSY2016-65, pp.37-37, 2016 年 12 月 15-16 日.

[2] 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”, 電子情報通信学会コンピュータシステム研究会 (CPSY), 電子情報通信学会技術研究報告 CPSY2016-114, pp.49-54, 2017 年 1 月 23-25 日.

[3] 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”, 電子情報通信学会集積回路研究専門委員会 (ICD), LSI とシステムのワークショップ 2017 ポスター, 2017 年 5 月 15-16 日.

[4] 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”, 電子情報通信学会コンピュータシステム研究会 (CPSY), 電子情報通信学会技術研究報告 CPSY2017-111, pp.31-36, 2018 年 1 月 18-19 日.

[5] 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, 2018 年 3 月 1 日.

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, 2018 年 3 月 1 日. 支部長賞 奨励賞 受賞