部分積の効率的な累算によりメモリ容量を削減した畳み込み演算アクセラレータ

徐 宏傑[1,a)]　塩見 準[1]　小野寺 秀俊[1]

概要：畳み込みニューラルネットワーク（CNN）では，メモリと PE 間の畳み込み層内でのデータ移動が最もエネルギーを消費する．本稿では，畳み込み処理のためのメモリアクセス数とオンチップバッファ容量の両方を最適化する畳み込み処理データフローを提案する．このデータフローに基づいて，オンチップバッファ容量を削減した CNN アクセラレータを設計した．評価実験を行った結果，提案する CNN アクセラレータは，既存の CNN アクセラレータと同じデータスループットにおいて，オンチップバッファ使用量を 2.5 分の 1 に削減し，エネルギー効率は 3.1 倍に向上した．提案アーキテクチャは，ほぼ一定のオンチップバッファ容量でより高いデータスループットを実現することが可能である．

1. 序論

近年，ネットワーク構造の改善に伴い，畳み込みニューラルネットワーク（CNN）の予測精度は着実に向上している．その結果，CNN は画像処理やパターン認識など数多くの分野で広く利用されている．そのため，多くのエッジ側端末において，ASIC（Application-Specific Integrated Circuits）等を用いて端末内で実時間推論を実行する事が求められている．一方，CNN ネットワークの深化・複雑化に伴い，必要な計算能力とエネルギー消費量も増加している [1, 2]．エッジ側端末の多くはバッテリ駆動であるため，エネルギー効率の高い ASIC が強く求められている．畳み込み (CONV) 動作は，CNN ネットワークが消費するエネルギーの 90%以上を消費する事が知られている [3]．そのため，低消費エネルギーかつ高スループットの CONV 演算用アクセラレータの開発に注目があつまっている [4–14]．CONV 演算時に必要となるデータへのアクセス特性に注目し，入力データ，フィルタデータ，部分和 (部分積) 結果の再利用性を向上させる検討が行われている [4–14]．これらの研究では，オフチップメモリへのアクセスとオンチップバッファへのアクセス回数の削減が図られている．しかし，メモリアクセスに必要なエネルギーは，アクセス回数だけでなく，メモリ自体の容量にも関係している [15]．処理中の中間結果を保存するために，多くの CNN アクセラレータは大量のオンチップバッファを必要とする．メモリアクセス回数の削減と同時に，CONV 演算に必要なオンチップバッファ容量の削減も重要である．

表 1　畳み込み演算のパラメータ．

パラメータ	説明
H_i /W_i	Ifmap 高さ / 幅
H_w /W_w	フィルター 高さ / 幅
H_o /W_o	Ofmap 高さ / 幅
C_i	Ifmap チャネル数
C_o	Ofmap チャネル数
S	ストライド数

本稿では，CONV 演算において活用可能な各種データとそれらの保存容量に関する再利用性について検討する．CONV 演算における保存容量の再利用とデータの再利用を組み合わせることで，オンチップバッファのアクセス回数とオンチップバッファの容量の双方を削減する計算方法を提案する．

本稿の構成を以下に示す．**2.** では，CONV 演算での乗算と加算におけるデータの再利用性について議論し，従来の取り組みを紹介する．**3.** では，CONV 演算におけるデータの再利用性とともに中間データの保存量削減を考慮したデータフローを提案する．**4.** でシミュレーションによる評価結果を示す．**5.** にて結論を述べる．

2. CNN 概要

CNN ネットワークは, 数層 [16] から数百層 [16] の CONV 層で構成されている．各 CONV 層は，相対的に小さなフィルタ群を用いて，入力特徴マップ (ifmaps) から出力特徴マップ (ofmap) と呼ばれる高レベルの特徴を抽出する．従来の CNN では，予測精度を高めるために，多チャンネル

[1] 京都大学大学院情報学研究科
[a)] xuhongjie@vlsi.kuee.kyoto-u.ac.jp

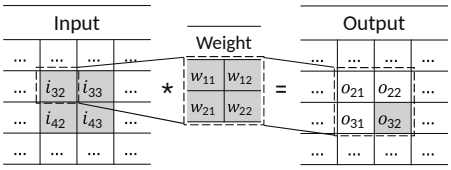

図 1 2 次元畳み込み演算．フィルタ中の個々の重みが参照される乗算の回数は $H_o \times W_o$ である．各 ifmap ピクセルが参照される乗算の総数は $H_w \times W_w$ である．各 ofmap ピクセルに対し，$H_w \times W_w$ の部分積の累算が必要である．

の 2 次元 (2D) ifmap や多チャンネルの 2D フィルタを用いることが多かった [17]．CNN アルゴリズムの改良に伴い，畳み込みニューラルネットワークにおいて，1 チャネルの 2 次元入力と 1 チャネルの 2 次元出力を用いた深さ方向畳み込み [18] も開発され，CONV の次元を小さくして省エネルギー化を図る試みもある．2 次元 ifmap と 2 次元フィルタの間の 2 次元 CONV は，CNN ネットワークにおける基本演算となる．畳み込みの形状を表すパラメータとして，表 1 を考えると，2 次元 CONV は次式のように表現できる．

$$O[x][y] = \sum_{i=1}^{H_w} \sum_{j=1}^{W_w} I[Sx+i][Sy+j] \times W[i][j], \quad (1)$$
$$1 \le x \le H_o, 1 \le y \le W_o.$$

ここで，O，I，W は，それぞれ ofmap，ifmap，重みを示す．

2.1 畳み込み演算におけるデータおよび保存空間の再利用

図 1 に，ストライド数 1 で処理される 2 次元畳み込み演算を示す．通常，S は ifmap ピクセルと重みの間の基本的なデータ依存性を変えるものではないため，以下では簡単のために S を 1 に固定する．

2 次元畳み込み演算には，$H_o \times W_o \times H_w \times W_w$ 回の乗算が必要である．乗算の結果 (部分積) に対して，同じ回数の累算も必要である．各フィルタの重みに対して，$H_o \times W_o$ 回の乗算が必要である．図の破線で示すように，各 ifmap ピクセルは $H_w \times W_w$ 回の乗算に出現する．同一の ifmap ピクセルとフィルタ重みが多数の演算に出現する事に関して，**データ再利用（data reuse）** [2] と呼ばれる概念が知られている．データ再利用は，1 個の ifmap ピクセルまたは 1 個の重みがオンチップバッファからレジスタにフェッチされてから，乗算のために再利用される回数を表している．図内の灰色の領域は，各 ofmap ピクセルが $H_w \times W_w$ 個の部分積を累算していることを示している．各部分積の物理的な記憶場所を考えた場合，1 個の ofmap 画素領域を中間データ格納領域として複数回使用することを**空間再利用（space reuse）**と定義する．空間再利用は，1 つの ofmap 画素用レジスタのデータがオンチップバッファに送られるまでに，累算のために当該レジスタがアクセスされる回数を示している．本稿では，2 次元畳み込み演算におけるリソースの再利用が，乗算時のデータ再利用と累算時の空間再利用の 2 種類に分けられることに注目する．

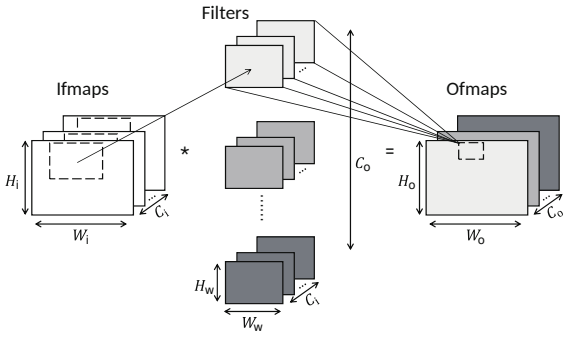

図 2 マルチチャンネル 2 次元コンボリューション演算．各 ifmap pixel はさらに C_o 倍の乗算を必要とする．各 ifmap pixel はさらに C_i 倍の加算を必要とする．

(1) 2D CONV の**データの再利用**：各 ifmap ピクセルは $H_w \times W_w$ 回再利用され，フィルターの重みは $H_o \times W_o$ 回再利用される．

(2) 2D CONV の**空間の再利用**：部分積を ofmap ピクセルに累算する際，各 ofmap ピクセルには $H_w \times W_w$ 回アクセスする．

データ再利用をまったく行わない場合，1 個の ifmap ピクセルに $H_w \times W_w$ 回アクセスしなければならない．また，1 個の重みデータに $W_o \times W_o$ 回アクセスしなければならない．同様に，畳み込みでの空間再利用を無視すると，$H_w \times W_w$ 倍のメモリ容量が必要となる．マルチチャンネルの 2D 畳み込みは，基本的には 2D CONV の次元を 3D CONV に拡張した処理と考えることができる．そのため，図 2 に示すように，データ再利用と空間再利用の概念を同様に適用することができる．

(1) Ofmap チャネル間のデータ再利用：各 ifmap ピクセルを，C_o 回余分に再利用可能である．

(2) Ifmap チャネル間での空間再利用：ofmap ピクセルの各要素は，部分積の累算が完了するまで余分に C_i 回アクセスされる．

2.2 既存研究

畳み込み演算におけるデータの再利用と部分積の効率的蓄積に焦点をあて，オフチップメモリのアクセス回数削減を図る研究が精力的に行われている [4–14]．図 3 で代表的な CNN アクセラレータを 2 例紹介する．図中の WM，IM，PM は，それぞれ重みデータ，ifmap ピクセル，ofmap ピクセル用のオンチップバッファを示す．図 3(a) に，Eyeriss [8] が採用している Row-Stationary (RS) データフローを示す．各処理要素 (PE) は独立した WM，IM，PM を持つ．各 PE は，重み行と入力行の間の部分積を計算する．重み行は PE 間で水平方向に共有され，入力行は斜め方向に共有され，部分積は垂直方向に蓄積される．理

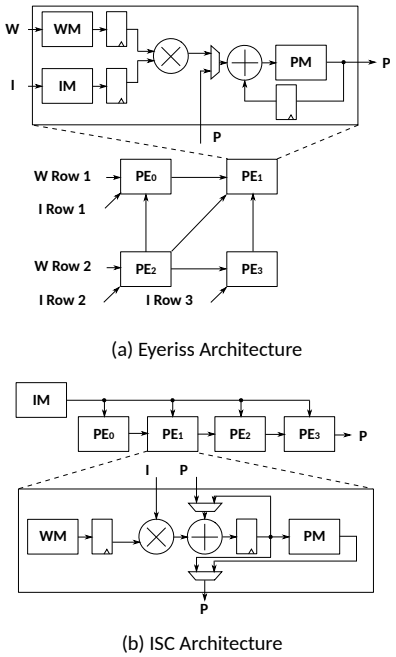

図 3 既存研究．Eyeriss アーキテクチャ [8](a) は，各 PE で同じ列の重みで部分積を計算し，上位の PE に結果を渡すことで部分積を蓄積する．ISC アーキテクチャ [6](b) は，全部の PE に ifmap ピクセルをブロードキャストし，その結果を次の PE に渡すことで部分積を蓄積する．

想的な状況では，RS データフローは ifmap ピクセルと重みを最大限に再利用して，オフチップメモリアクセスを最小限に抑え，高いスループットを実現する．様々な形状の CONV 演算に適応し，柔軟性を維持するために，PE は独立して動作するコンピューティングユニットとして設計されている．その結果，各 PE は，隣接する PE が同じ入力データを使用する場合でも，近い将来にデータを再利用するために，同一のデータを複数のローカルレジスタに保存する．重複データの保管は，オンチップバッファのエネルギー損失につながる．

図 3(b) に ISC アーキテクチャを示す．Ifmap と重みデータアクセスの冗長性を考慮して，ISC アーキテクチャは ifmap ピクセルによって必要とされる乗算を可能な限り迅速に計算する．単一の ifmap ピクセルは，すべての PE にブロードキャストされ，必要なすべての重みデータと乗算し，PE アレイのレジスタに一時的に格納される．同じ行の重みに対応する部分積は，PE 内の PM に一時的に格納される．次の ifmap ピクセルの結果が到着すると，隣接する PE に部分積をユニキャストして蓄積する．Ifmap を効率的に再利用することで，ISC は Eyeriss よりも優れたエネルギー性能を実現する．しかし，ISC の部分積の空間再利用の能力は同一行に限定されているため，列間の空間再利用は無視される．その結果，PM の容量が H_{w} 倍に増加する．さらに，1 つの CONV 処理内の重みサイズが小さいため，同一の ifmap を用いる複数の ofmap チャネル方向に処理を並列化し，ifmap の利用率をさらに向上させている [6]．しかし，このような ofmap チャネルの並列計算では，ifmap チャネル間の空間再利用機能を有効に利用することができない．最悪の場合，ISC は，合計で $H_{\mathrm{w}} \times C_{\mathrm{o}}$ 倍の PM 容量を要求することになり，エネルギーのオーバーヘッドが大きくなる．

これまでのアプローチは，いずれもデータ再利用を最大化することでメモリアクセス数を最適化することを目的としているが，オンチップメモリ容量のデータフロー依存性について十分に議論されていない．一般に，メモリアクセスに必要な消費エネルギーはメモリ容量に強く依存するため，CONV 処理において，空間再利用性を最大化して，メモリ容量の削減を図る事よりで，エネルギー効率をさらに向上できる．

3. 提案データフロー

データの再利用と空間の再利用の両方を考慮するために，2D CONV のデータフローでは，重みの水平・垂直方向の再利用と ifmaps の水平・垂直方向の再利用の両方を考慮する必要がある．図 4(a) に ISC の ifmap アクセスパターンを示す．これまでの CNN アクセラレータは，CONV でのデータ再利用を最大化することを目的としている．したがって，ISC のデータフローは，1 つの ifmap ピクセルと，複数の ofmap チャネル間のすべてのフィルタ重みの間の操作を並列に処理する．この並列データフローにより，ifmap と重みの両方のデータ再利用を最大化できる．しかし，異なる ofmap チャネル間の部分積には空間再利用機能がない．したがって，アクセラレータ設計にあたり，性能オーバーヘッドに繋がるオフチップメモリアクセスを避けるために，演算途中の部分積を格納するための十分なオンチップストレージスペースを確保しなければならない．

この問題を解決するために，提案するデータフローでは，図 4(b) に示すように，複数の行で同時に ifmap のピクセルにアクセスする．Ifmap ピクセルは，一度アクセスした乗算でのデータ再利用を最大化するために，2D CONV 内の重みに ifmap ピクセルをマルチキャストする．2D CONV 内の蓄積が終了すると，提案するデータフローでは，2D CONV を ifmap チャネルごとに 2D CONV の処理を開始し，部分積に割り当てるメモリ容量を削減する．以上により，部分積に必要なメモリ容量を削減することが可能である．

提案のデータフローをアルゴリズム 1 に示す．第 5 行目は，各クロックサイクルで PE が処理するタスクの範囲を示している．4 行目と 5 行目は，レジスタに格納するデータの範囲に対応する．3 行目，4 行目，5 行目は，オンチップバッファに格納するデータの範囲に対応する．データフローは，5 行目に示すように，多数のコラムで同時に ifmap ピクセルにアクセスする．ALU 内の ifmap のデータ再利用を確保するために，ifmap ピクセルは 1 つのチャネル内

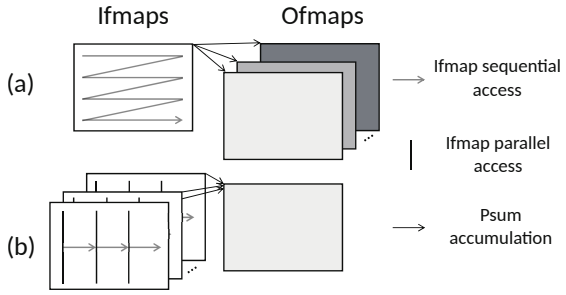

図 4 (a)ISC と (b) 提案データフロー における ifmap データ再利用と ofmap 空間再利用．ISC では 1 つの ifmap ピクセルを取得し，そのピクセルを ofmap チャネル間で処理する．Ifmap データの再利用は最大化されるが，空間再利用を活用できない部分積が生成される．提案データフローでは，同じ列にある ifmap の画素をフェッチし，累算可能な部分積を生成することで，メモリ容量を節約する．

Algorithm 1 提案データフロー．

Require: W, I
Ensure: O
1: Initialization;
2: **for** $u \in [1 : C_{\mathrm{o}}]$ **do**
3: **for** $k \in [1 : C_{\mathrm{i}}]$ **do**
4: **for** $x \in [1 : W_{\mathrm{o}}]$, $y_1 \in [1 : H_{\mathrm{o}}/n]$ **do**
5: **for** $y_2 \in [1 : n]$, $i \in [1 : W_{\mathrm{w}}]$, $j \in [1 : H_{\mathrm{w}}]$ **in parallel do**
6: $\mathbf{O}[u][x][y_1y_2] += \mathbf{I}[k][x + i][y_1y_2 + j] \times \mathbf{W}[u][k][i][j]$;
7: **end for**
8: **end for**
9: **end for**
10: **end for**
11: Return O

の全ての重みにマルチキャストされる．同様に，ALU の利用率を維持するために，データフローは n カラムの ifmap ピクセルを並列に処理する．積和累算の中間結果は PM に格納され，最終の ofmap ピクセル値が得られた段階でオフチップメモリに転送される．

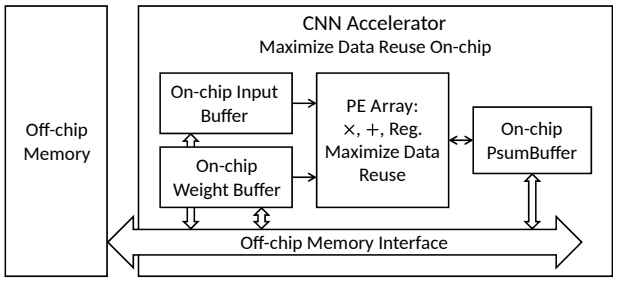

図 5 アクセラレータのアーキテクチャ．MAC 演算は提案された CNN アクセラレータで処理される．重みと ifmap のためのオンチップバッファがデータを格納し，乗算でのデータ再利用を待つ．PE 配列には，可能な限り部分積を蓄積するために，複数の乗算器，加算器，レジスタが含まれている．計算された ofmap ピクセルは，オフチップ・メモリ・インターフェースを介して直接オフチップ・メモリに送られる．PM は時間内に ofmap ピクセルに蓄積できなかった部分積を蓄積する．

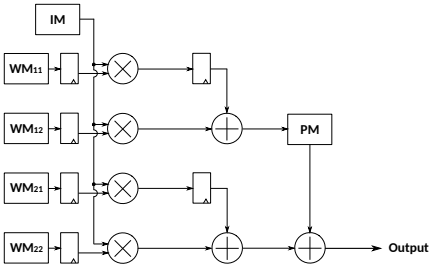

図 6 PE アレイアーキテクチャ．可能な限り早く部分積を加算して，PM と部分積レジスタの容量を減らす．フィルタサイズを 2×2 とする．IM，PM の容量は入力幅と等しい．重みは全て WM に格納する．

提案する CNN アクセラレータのアーキテクチャを図 5 に示す．空間再利用を活用してオンチップバッファの容量を減らし，データの再利用に応じてオンチップバッファへのアクセス数を制限することを目的としている．2D CONV における提案データフローの 1 実装例として，$H_{\mathrm{w}} \times W_{\mathrm{w}} \times H_{\mathrm{o}}$ 個の PE を使用して並列計算を行う構造が考えられる．この場合には ofmap ピクセル数以上の PM は不要となる．W_{o} クロックサイクルで部分積を蓄積することで，データ再利用と空間再利用を同時に最大化している．しかし，アプリケーションによっては，ofmap の H_{o} が大きく，ハードウエア実装が出来ないことがある．その一方で，別のアプリケーションでは $H_{\mathrm{w}}W_{\mathrm{w}}$ が小さく，PE アレイで利用可能な乗算器を十分活用できない状況も考えられる．そこで，提案するアーキテクチャでは，2D CONV の再利用を利用するために，n の値を 1 から H_{o} までの適切な値に設定する事を想定している．そのため，n が H_{o} 以外の場合には, ofmap ピクセルの部分的累算結果を PM に格納することになる．

図 6 では，n を 1 とした基本的な PE 配列を提案しているが，n を 1 とすると，1 クロック周期の全ての部分積が異なる ofmap 空間に対応する．このとき，乗算が必要な部分積はレジスタ配列に一時的に格納される．次の ifmap ピクセルが計算されると，累算器は部分積レジスタに部分積を累算し，現在の部分積を加算する．積算結果が次の ifmap 行のピクセルを必要とする場合，PE 配列はその結果を PM に送信する．ifmap のアクセスが次の行に来ると，PM に格納されている前の部分積を再読み込みして，最終的な ofmap のピクセルに蓄積する．

図 7 では，n を 2 とした場合の PE 配列を示している．n が 2 になると，1 クロックサイクルで空間の再利用が可能になり，関連する部分積を追加して部分積レジスタに送ることが可能である．累算が完了した ofmap ピクセルの値は，直接オフチップメモリに送られる．これにより，PM へのアクセス数と PM の必要容量の両方が削減される．

提案するデータフローは，空間とデータを同時に再利用することで，より少ないメモリ容量とメモリアクセスで

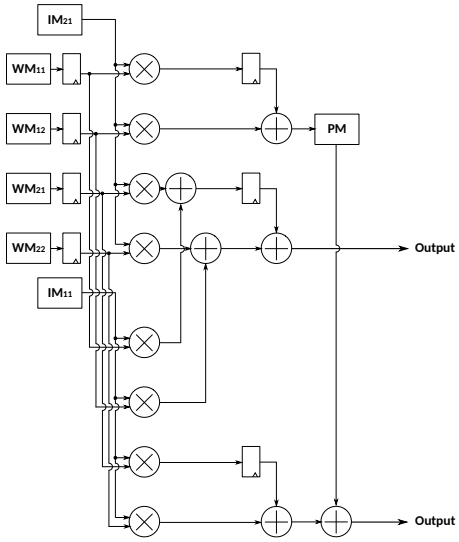

図 7 $n = 2$ のときの PE アレイのアーキテクチャ．フィルタサイズを 2×2 と仮定する．ほとんどの PM は，レジスタアレイの最終的な ofmap ピクセルに蓄積することが可能である．これにより，PM アクセスと PM 容量の両方の要件を削減することが可能である．

表 2 比較パラメータ．

a	フィルター 高さ / 幅
b	Ifmap 高さ / 幅
N	Ofmap チャネル
Na^2	PE 数

同等以上のスループットを実現している．次節では，提案アーキテクチャと ISC アーキテクチャの類似点と相違点を，2D CONV 条件下で定量的に比較・解析する．

4. 実験検証

本節では，提案データフローと ISC データフローの比較を行う．ハードウェアの利用可能な資源はアプリケーションに応じて大幅に変化するため，幅広い設計環境に対応するために，アーキテクチャに搭載する PE 数と性能の依存関係を評価する．ISC の構造は，ofmap チャネル数が 1 個の場合には効果的な並列化が実行できない．公正な比較を行うために，比較環境を単一の ifmap チャネル，複数の ofmap チャネルでの畳み込み演算に設定した．詳細な畳み込み形状を表 2 にまとめる．

表 3 で，ifmap サイズが b^2，フィルタサイズが a^2，ofmap チャネルが N の畳み込み演算を行う場合にそれぞれのデータフローが必要なレジスタパラメータを示す．N = 1 の場合のメモリ要件については，文献 [6] にまとめられている．ここでは，それを $N > 1$ の場合に拡張する．ISC は，ifmap の 1 ピクセルに対するすべての乗算を 1 回で計算することを保証している．ofmap チャネル数が N の場合，ISC は Na^2 個の PE を必要とする．PE の数に比例して，Na^2 個のの重みレジスタと部分積レジスタが必要になる．Ifmap レジスタは直接全 PE にピクセルをブロードキャストするので，ifmap レジスタは不要である．レジスタアクセス数を考えると，ISC も提案アーキテクチャも Weight Stationary (WS) データフローを採用しているため，重みレジスタへのアクセス数は処理時間 b^2 のオーダーに相当する．提案アーキテクチャでは，2 次元 CONV の ifmap ピクセルを $n = N$ に設定し，並列処理のために N 個の ifmap ピクセル取得している．その結果，提案アーキテクチャにおいては，2D CONV の垂直空間再利用機能を利用して，a 個の垂直部分積を 1 個の部分積レジスタにマージする．

表 3 データフローに必要な最小レジスタ数．データの転送時間を無視すると，両方のアーキテクチャのデータ処理時間は b^2 である．

アーキテクチャ	PE	レジスターサイズ		レジスターアクセス回数	
		重み	部分積	重み	部分積
ISC [6]	Na^2	Na^2	Na^2	b^2	Na^2b^2
This work	Na^2	a^2	$a^2 + aN$	b^2	Nab^2

表 4 データフロー実装に必要な最小のバッファ量．

アーキテクチャ	バッファサイズ			バッファアクセス回数		
	Ifmap	重み	部分積	Ifmap	重み	部分積
ISC [6]	b	Na^2	Na^2b	Nb^2	Na^2	Nab^2
This work	Nb	a^2	a^2b	Nb^2	Na^2	ab^2

表 5 演算器とメモリのエネルギー消費量．

コンポーネント		エネルギー
16 ビットレジスターアクセス* (E_{REG})		0.18 pJ
16 ビット乗算 (E_{MAC})		0.21 pJ
16 ビットオンチップバッファアクセス* (E_{BUF})	1 kB	2.04 pJ
	8 kB	6.63 pJ
16 ビット 8 MB オフチップ SRAM アクセス (E_{OFF})		104.45 pJ

*: 読み出し動作と書き込み動作の平均エネルギー消費量．

表 4 に，オンチップバッファの要件を示す．ISC は，ofmap チャンネル間で並列化を図るのに対し，提案アーキテクチャは 1 ofmap チャネル内での並列化を図っている．ofmap チャンネルの並列計算では，空間再利用が考えられないため，ISC では N 倍の WM と PM が必要になる．バッファへのアクセス回数を考えると，ISC が利用可能な空間再利用性は重みの水平方向の累積であるため，PM には Nab^2 倍のアクセスが必要となる．

本研究では，提案構造の優位性を示すために，RTL (Register-Transfer Level) シミュレーションを行い，提案アーキテクチャと既存のアクセラレータである ISC の比較を行う．まず，両アーキテクチャの RTL 設計を行う．次に，本研究では，65 nm プロセス技術のスタンダードセルライブラリから演算器とレジスタのエネルギーを評価する．CACTI Ver.7.0 [19] の SRAM モデルからオンチップバッファとオフチップメモリへのアクセスの消費電力を 65 nm プロセス，電源電圧 1.2 V で評価する．表 5 は，1.2 V の電源電圧で評価したエネルギーをまとめたものである．すべてのハードウェアコンポーネントは 16 ビット固定小数点表現で構築されている．ベンチマークを実行している両

アクセラレータの RTL シミュレーションによってコードトレースログを生成する．エネルギー消費は，以下の式に基づき評価した．

$$E = E_{\mathrm{MAC}} * C_{\mathrm{MAC}} + E_{\mathrm{REG}} * C_{\mathrm{REG}} + E_{\mathrm{BUF}} * C_{\mathrm{BUF}} + E_{\mathrm{OFF}} * C_{\mathrm{OFF}}, \quad (2)$$

ここで，E，E_{MAC}，E_{REG}，E_{BUF}，E_{OFF} は，それぞれ，1 回の積和処理，レジスタへの 1 回のアクセス，オンチップバッファへの 1 回のアクセス，オフチップメモリへの 1 回アクセスの消費エネルギーである．C_{MAC}，C_{REG}，C_{BUF}，および C_{OFF} は，それぞれ，積和処理，レジスタ，オンチップバッファ，およびオフチップメモリの総アクセス数である．

両データフローの RTL シミュレーションに基づき，Mobilenet で CONV を処理する際のオフチップメモリアクセスを最小化するために必要なバッファ容量を図 8 に示す．3 章の議論に基づき，提案するデータフローでは，全ての処理タスクのデータを全ての ifmap チャネルと少なくとも 1 つの ofmap チャネルに格納する必要があり，少なくとも 1114 KB のオンチップバッファが必要となる．同様に，ISC データフローでは，すべての ofmap チャネルと少なくとも 1 つの ifmap チャネルにデータ用のデータを格納する必要があり，少なくとも 2792 KB のオンチップバッファを必要とする．本稿では，Mobilenet の ifmap チャネル数よりも ofmap チャネル数の方が多いため，ISC のバッファ要件よりも 2.5 倍小さくなっている．

図 9 に，Mobilenet での depthwise CONV 層を 64 個の演算器により処理した場合の ISE の消費エネルギーを示す．図 10 に，Mobilenet で depthwise CONV 層以外のレイヤを 64 個の演算器で処理した場合の消費エネルギーを示す．部分積バッファへの高価なアクセスを制限し，提案アーキテクチャは大幅なエネルギー削減を実現している．その結果，不要な PM 空間を削除することで，消費エネルギーを 3.1 分の 1 に削減した．同時に，提案するアーキテクチャは，PM にアクセスすることなく，ほとんどの部分積を最終的な ofmap ピクセルに蓄積可能であることを保証している．空間再利用とデータ再利用を考慮することで，提案アーキテクチャは必要なバッファ容量とバッファアクセス数の両方を考慮している．さらに，提案するアーキテクチャは，必要なバッファ容量と必要なエネルギー消費の両方を削減する．

実環境でのスケーラビリティと消費電力の優位性をさらに示すために，乗算器の数を変えて，ハードウェア記述言語に基づき提案するアクセラレータを設計した．CNN 評価モデルとして Alexnet [20] を用いる．実験では，4 枚の画像をバッチ処理する．提案アーキテクチャの性能を表 6 にまとめる．ISC は理論的には ifmap ピクセルへの読み込みアクセスを最適化しているが，ofmap ピクセルの潜在的な空間再利用性を無視している．1 チャネルの 2 次元 CONV では，理論的には 4.2 kB のオンチップバッファで空間再利用を最大化することが可能である．しかし，チャネル間での資源再利用のために，オンチップバッファの容量を 60.4 kB に設定することで，オフチップアクセスの数を減らすことができる．これにより，同等のスループットを得ながら，オンチップバッファ容量を 2.3 分の 1 に削減

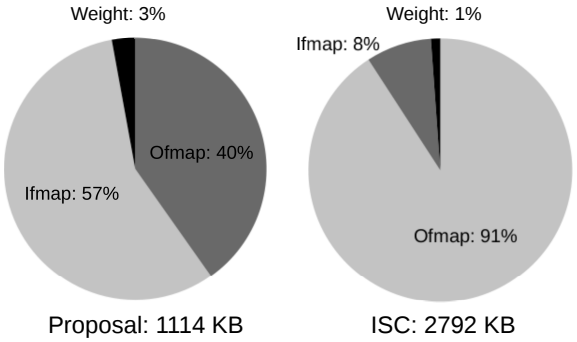

図 8 Mobilenet の CONV 処理タスクのオフチップメモリアクセスを最小化するためのオンチップバッファの容量．

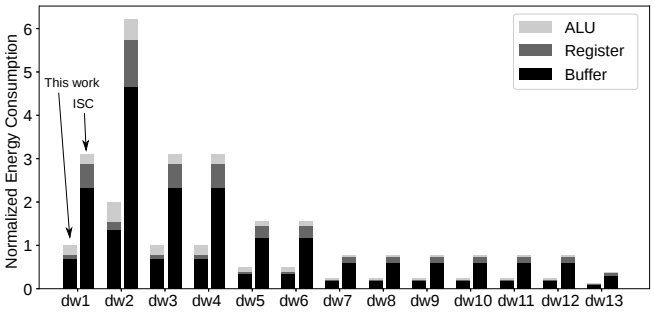

図 9 Mobilenet の CONV 処理タスクの性能評価結果．横軸のラベルは，Mobilenet のすべての depthwise の CONV レイヤーで構成されている．縦軸は，正規化されたエネルギー消費量を示している．エネルギー消費量は，図の最初の深さ方向の CONV レイヤーでの提案作業のシミュレーション結果によって正規化されている．

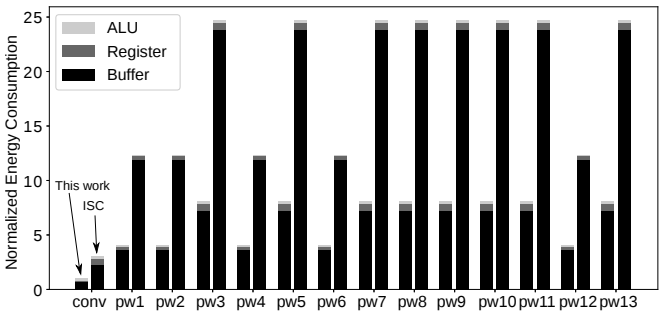

図 10 Mobilenet の CONV 処理タスクの性能評価結果．横軸のラベルは，Mobilenet の CONV レイヤーと pointwise の CONV レイヤーで構成されている．縦軸は正規化されたエネルギー消費量を表している．エネルギー消費量は，図中の最初の CONV レイヤーでの提案データフローのシミュレーション結果で正規化されている．

表 6　Alexnet でのアクセラレータ性能.

アーキテクチャ	ISC [6]	This	Eyeriss2 [8]	This
PE 数	64	72	384	432
周波数 [MHZ]	250	250	200	200
スループット [GMACS]	11.7	13.9	61.8	73.8
オンチップメモリ [kB]	139.6	60.7	492*	66.0
パワー効率 [mW]	93.4	42.9	450	96.4

オンチップメモリは，すべてのバッファとレジスタで構成されている.
*: メモリ容量は 16 ビットで正規化されている.

し，エネルギーを 2.2 分の 1 に削減することができる．提案構造は，ifmap チャネル間の空間再利用を最大化可能であるので，このアーキテクチャは，$H_{\mathrm{o}} \times H_{\mathrm{w}} \times W_{\mathrm{w}} \times C_{\mathrm{i}}$ までの大規模な PE スケールに容易に拡張することが可能である．同じデータフローを用いて，より大規模なアーキテクチャを設計し，Eyeriss v2 [8] との性能比較を行う．その結果，Eyeriss v2 に対して，オンチップバッファの容量を 7.5 分の 1 に削減し，消費エネルギーを 4.7 分の 1 に削減できることが判明した.

5. 結論

畳み込み計算では，メモリと PE 間のデータ移動がエネルギー全体の大部分を消費する．メモリが消費するエネルギーは，アクセス数とメモリの容量の両方によって決定される．本論文では，畳み込み計算におけるデータ再利用と空間再利用を考慮し，メモリアクセス数とメモリサイズを最適化する CONV データフローを提案した．既存の CNN アクセラレータ [6]，[8] と比較して，Alexnet において同一のスループットを得るためのオンチップバッファ量は，それぞれ 2.3 分の 1，7.5 分の 1 に削減できる．消費エネルギーも，それぞれ 2.2 分の 1 と 4.7 分の 1 に削減可能であることが判明した.

6. 謝辞

本研究は，文部科学省「卓越大学院プログラム」の助成を受けている．また，本研究一部は，東京大学大規模集積システム設計教育研究センターを通し，シノプシス株式会社，日本ケイデンス株式会社，メンター株式会社の協力で行われた.

参考文献

[1] 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)*, June 2015, pp. 1–9.

[2] V. Sze, Y. Chen, T. Yang, and J. S. Emer, “Efficient processing of deep neural networks: A tutorial and survey,” *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec 2017.

[3] J. Cong and B. Xiao, “Minimizing computation in convolutional neural networks,” in *Artificial Neural Networks and Machine Learning – ICANN 2014*. Cham: Springer International Publishing, 2014, pp. 281–290.

[4] J. L. *et al*, “Unpu: A 50.6tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision,” *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 218–220, 2018.

[5] N. P. J. *et al*, “In-datacenter performance analysis of a tensor processing unit,” in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, June 2017, pp. 1–12.

[6] J. Jo, S. Kim, and I. Park, “Energy-efficient convolution architecture based on rescheduled dataflow,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4196–4207, Dec 2018.

[7] B. Moons and M. Verhelst, “An energy-efficient precision-scalable convnet processor in 40-nm cmos,” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, April 2017.

[8] Y. Chen, J. S. Emer, and V. Sze, “Eyeriss v2: A flexible and high-performance accelerator for emerging deep neural networks,” *CoRR*, vol. abs/1807.07928, 2018. [Online]. Available: http://arxiv.org/abs/1807.07928

[9] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, “Cnvlutin: Ineffectual-neuron-free deep neural network computing,” in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 1–13.

[10] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, “Shidiannao: Shifting vision processing closer to the sensor,” in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, June 2015, pp. 92–104.

[11] A. Ardakani, C. Condo, M. Ahmadi, and W. J. Gross, “An architecture to accelerate convolution in deep neural networks,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1349–1362, April 2018.

[12] S. Park, K. Bong, D. Shin, J. Lee, S. Choi, and H. Yoo, “4.6 a1.93tops/w scalable deep learning/inference processor with tetra-parallel mimd architecture for big-data applications,” in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, Feb 2015, pp. 1–3.

[13] S. Yin, P. Ouyang, S. Tang, F. Tu, X. Li, L. Liu, and S. Wei, “A 1.06-to-5.09 tops/w reconfigurable hybrid-neural-network processor for deep learning applications,” *2017 Symposium on VLSI Circuits*, pp. C26–C27, 2017.

[14] M. Cho and D. Brand, “MEC: memory-efficient convolution for deep neural network,” *CoRR*, vol. abs/1706.06873, 2017. [Online]. Available: http://arxiv.org/abs/1706.06873

[15] H. Xu, J. Shiomi, T. Ishihara, and H. Onodera, “Maximizing Energy Efficiency of on-Chip Caches Exploiting Hybrid Memory Structure,” in *2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. IEEE, July 2018, pp. 237–242.

[16] 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)*, June 2016, pp. 770–778.

[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv 1409.1556*, 09 2014.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and

H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[19] S. J. E. Wilton and N. Jouppi, “CACTI: an Enhanced Cache Access and Cycle Time Model,” *Journal of Solid State Circuits*, vol. 31, no. 5, pp. 677–688, May 1996.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.