Vol.2017-ARC-225 No.28
Vol.2017-SLDM-179 No.28
Vol.2017-EMB-44 No.28
2017/3/9

マルチコアニューラルネットワークアクセラレータにおけるデータ転送のブロードキャスト化

大場 百香[1,a)]　三輪 忍[1]　進藤 智司[2]　津邑 公暁[2]　八巻 隼人[1]　本多 弘樹[1]

概要：マルチコアニューラルネットワークアクセラレータでは, メモリとコア間のデータ転送時間がボトルネックとなっており，ニューラルネットワーク計算を効率良く行うことができない．そこで本論文では，このデータ転送をブロードキャスト化することでボトルネックを解消するアクセラレータを提案し，性能分析およびハードウェアコストの評価を行った．

1. はじめに

画像認識や音声認識の分野で広く使われているニューラルネットワーク (Neural Network： NN) は，DeepLearning[3] の登場によって近年急速に大規模化が進み，その計算に要する時間や消費エネルギーが問題となっている．このような背景から，現在，NN 計算のためのハードウェアアクセラレータ（Neural Network Accelerator：NNA）が多数開発されている [1], [2], [4]．NNA は，オフチップ GPU と同様，アクセラレータとしてホストコンピュータに PCIe 等の手段で接続し，ホストコンピュータから NN 計算をオフロードすることを想定して開発が進められている．NNA は CPU と比べて 100 倍程度高速に NN 計算を行うことができる上，その消費エネルギーは GPU の 1/100 程度で済むことが知られている [4]．しかし，既存の NNA はエネルギー効率に優れている一方，処理できるニューロンモデルや学習アルゴリズムが制限されており，その用途が限定的である．このような既存の NNA が持つ制約は，人工知能研究者などの NN 自体を研究しているユーザにとっては受け入れ難いものである．

このような背景から，我々は，ユーザによってニューロンモデルや学習アルゴリズムを変更することができ，また，高速かつ低消費エネルギーで NN 計算を行える NNA を開発している [6]．我々の NNA（再構成可能な NNA（Reconfigurable NNA：RNNA））は DaDianNao[2] のアーキテクチャをベースに開発しており，結合重みやニューロン値を格納するためのメモリ，ニューロンの出力値を計算するための積和演算器といった，NN 計算に必要とされるカスタムロジックの他に，再構成可能ロジックを搭載する．これにより各ユーザは，ニューロンモデル毎の出力計算アルゴリズムや学習アルゴリズムを再構成可能ロジック上に実装することによって，所望の NN 計算をアクセラレータ上で実行できる．

DaDianNao[2] や RNNA のようなマルチコア NNA（Multicore NNA：MNNA）では，全コアが共有するメモリ上にニューロン値が格納されており，各コアが NN 計算を行う際は共有メモリへのアクセスが発生する．我々の以前の研究ではこの共有メモリへのアクセスが MNNA の性能上のボトルネックとなることを明らかにした [6]．このボトルネックを解消するため，本稿では，全コアへの同一データの転送をブロードキャスト化したアーキテクチャ（MNNA-bc）を提案する．

本論文の構成は以下のようになっている．まず 2 章では MNNA について詳しく述べる．続く 3 章では MNNA-bc について述べる．4 章で提案アクセラレータの性能とハードウェアコストの評価を行い，5 章で本論文をまとめる．

2. マルチコアニューラルネットワークアクセラレータ

2.1 DaDianNao の概要

DaDianNao[2] は，DianNao[1] の問題点である計算ユニット（NFU）とメインメモリ間のバンド幅の不足を解消し，大規模 NN 計算の更なる高速化と省エネルギー化を目的として設計されたマルチコア NNA である．

DaDianNao は複数のコアと共有メモリによって構成されており（**図 1**），各コアが複数ニューロンの出力計算を並列に行う．ニューロンの出力計算を行うユニット (NFU) はカスタムロジックによって構成されている．結合重みは

[1] 電気通信大学
[2] 名古屋工業大学
[a)] ohba@hpc.is.uec.ac.jp

Vol.2017-ARC-225 No.28
Vol.2017-SLDM-179 No.28
Vol.2017-EMB-44 No.28
2017/3/9

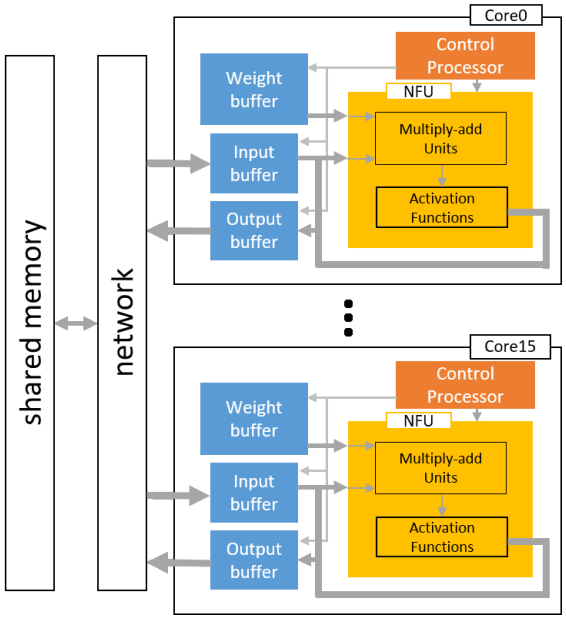

図 1 DaDianNao のアーキテクチャ

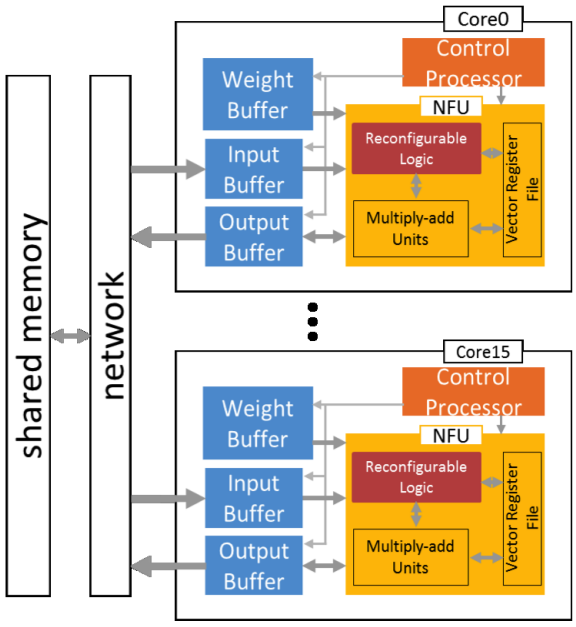

図 2 RNNA アーキテクチャ

各コアが備える結合重みバッファ（Weight Buffer）に格納されている．

一方，ニューロンの入力値は図 1 の左側に位置する共有メモリ（shared memory）に格納されており，必要なデータが制御プロセッサ（Control Processor）によって入力バッファ（Input Buffer）にロードされる．NFU は結合重みバッファと入力バッファの値（どちらもベクトル・データ）を用いて複数ニューロン（16 個程度）の出力計算を並列に行い，計算結果（ベクトル・データ）を出力バッファ（Output Buffer）へ出力する．なお，NFU 内は 3 つのパイプライン・ステージに分かれており，出力計算はパイプライン処理される．このように，DaDianNao のコア内の動作は，共有メモリからの結合重みのロードが不要な点を除いて DianNao とほぼ同じである．

DaDianNao は，このように計算に必要なデータを格納するメモリを NFU の近くに配置することで，これらのデータのアクセスレイテンシとアクセスエネルギーを抑制している．

2.2 再構成可能なニューラルネットワークアクセラレータの概要

我々は，任意の NN 計算を可能とする再構成可能ロジックを搭載した NNA として RNNA を提案した [6]．MNNA のアーキテクチャを**図 2** に示す．

基本的なアーキテクチャは DaDianNao と同様であり，複数コアがネットワークを介して接続されており，1 つのメモリを共有した構成となっている．各コアは，NFU および制御プロセッサと入力，出力ニューロン，結合重みそれぞれの値を保持するバッファによって構成される．RNNA が DaDianNao と異なる点は，再構成可能ロジックを採用した点と，演算途中の結果を一時的に格納するためのベクトル・レジスタを NFU 内に配置した点である．

再構成可能ロジックは，積和演算以外の NN 計算（例えば，活性化関数の計算，膜電位の減少量とスパイク発生の計算など）をハードウェアで実現する．積和演算は多くの NN 計算で必要とされるため，カスタムロジックとして構成する．また，ベクトル・レジスタはニューロンの状態や膜電位を保持する．これら NFU 内のロジックの動作はすべてパイプライン化される．

制御プロセッサは，他のコアへのデータ転送や推論処理と学習処理のどちらを行うかをコアに対して指示する．DaDianNao と同様の構成にすることで，共有メモリと NFU 間の結合重みの転送を省略し，転送に必要なレイテンシと消費エネルギーを抑制する．それに加え，一部のスカラ命令（分岐命令や整数系の四則演算命令）も実行できる．このようにすることで，ループ構造を持った制御プログラムの記述を可能にし，プログラム中の静的命令数を削減できる．図 2 には示していないが，制御プロセッサ内にはプログラム格納用のメモリが存在している．また，上記のスカラ命令を実行するために，制御プロセッサ内には少量のスカラ・レジスタが存在する．

2.3 MNNA のネットワークアーキテクチャ

DaDianNao[2] や RNNA[6] のような MNNA の詳細アーキテクチャを**図 3** に示す．MNNA は，入力，重み，出力それぞれに対しコア数分のオンチップ・ネットワークを有し，コア毎，またデータ毎に独立したデータ転送を可能とする．また，DMAC（DMA Controler）を各コアのバッファ毎に設置している．各 DMAC には格納できるメモリリクエスト数に上限がある．共有メモリはオンチップ・ネットワー

Vol.2017-ARC-225 No.28
Vol.2017-SLDM-179 No.28
Vol.2017-EMB-44 No.28
2017/3/9

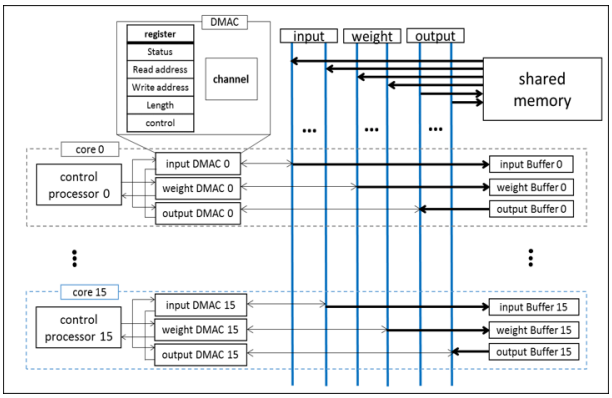

図 3 MNNA ネットワークアーキテクチャ

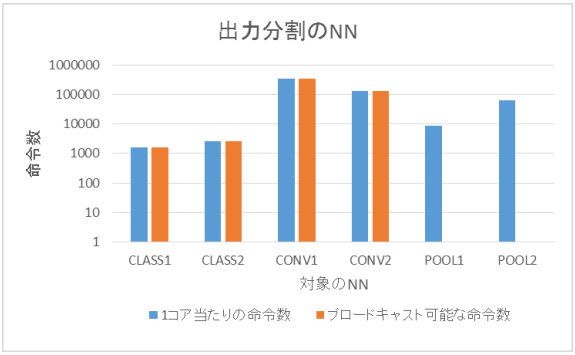

図 4 出力分割されている NN のブロードキャスト数

クに合わせた 48 ポート，各バッファは 1 ポートを有する．

また，MNNA の動作は，共有メモリからニューロン値を入力バッファへロードし，コア内で結合重みと入力バッファの値を用いて計算を行い，結果を出力バッファまたは共有メモリに書き込む，という一連の処理を行う長命令を実行する．

MNNA では，コア毎にオンチップ・ネットワークを有することで 1 本あたりのオンチップ・ネットワーク・バンド幅が小さくなるので，メモリとコア間のデータ転送時間がボトルネックになる [6]．このボトルネックを解消するために，我々はデータ転送のブロードキャスト化を行うことにした．

3. データ転送のブロードキャスト化

3.1 予備実験

ブロードキャスト可能なデータ転送命令がどの程度存在するかを調べるため，DaDianNao[2] の評価で使用された NN 計算に含まれる命令数を調査した．この NN 計算はすべて出力分割 [7] されている．結果を**図 4** に示す．CLASS 層と CONV 層では大部分のデータ転送命令が全コアで同じデータに対する処理をするため，データを全コアへブロードキャストできることがわかった．一方で，全コアでニューロンの出力を共有する入力分割されている NN ではデータをブロードキャストできるデータ転送の命令数は 0 だった．

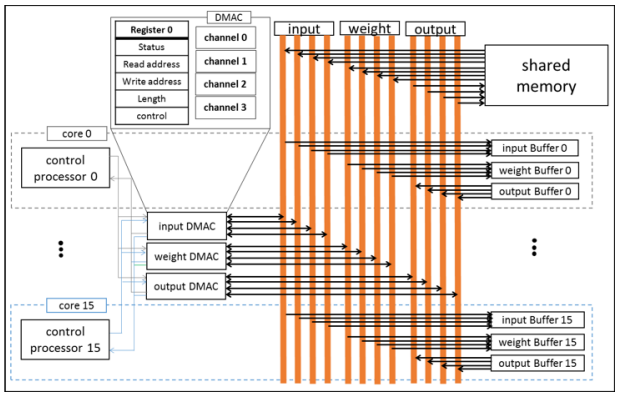

図 5 MNNA-bc アーキテクチャ

3.2 ブロードキャスト化のためのアーキテクチャ

入力，重み，出力それぞれのオンチップ・ネットワークと DMAC を全コア共有にし，データ転送をブロードキャスト化した MNNA（MNNA-bc）を提案する．提案アクセラレータのアーキテクチャを**図 5** に示す．

基本的なアーキテクチャは 3.1 で述べた MNNA と同様だが，MNNA と異なる点は，オンチップ・ネットワークのバンド幅と本数，DMAC 内アーキテクチャである．オンチップ・ネットワークの本数を 16 本から 4 本へ変更するのに伴い，各ネットワークのバンド幅も 6.25GB/s から 25GB/s に変更する．総バンド幅は 100GB/s と変更はない．DMAC は各コアのバッファ毎に存在していたので，DMAC 内チャネルは 1 チャネルであったが，MNNA-bc では，オンチップ・ネットワークの数に合わせて 4 チャネルを要する．

MNNA-bc の動きは MNNA とは異なる．相違点は，1 つのリクエストでロードするデータの数である．MNNA では 1 つのリクエストで常に 1 つのデータをロードしていたが，MNNA-bc では，各コアからリクエストされたデータがブロードキャストできる場合であればまとめて共有メモリからデータをロードする．データのブロードキャストの可否は DMAC 内で各コアからのリクエストをチェックし，判断する．

DMAC 内での判断は，DMAC から共有メモリにロードする際に参照する表（**表 1**）を作成し，各コアからリクエストを受け取る際にリクエストが共有メモリの同じアドレスであれば DMAC 内で作成した表のエントリに 1 を書き込むことにより行う．エントリの初期値は 0 とする．全コア

表 1 DMAC 内の表のイメージ

	Core 0	Core 1	Core 2	…	…	Core 15
address 0	1	0	1	…	…	1
address 1	1	1	1	…	…	1
address 2	1	1	1	…	…	1
…	1	1	0	…	…	1
…	1	0	0	…	…	1
…	1	0	0	…	…	1

のリクエストが DMAC に到着したら，DMAC はチャネルの 1 つを使用して，これらのリクエストが要求する共有メモリ上のデータに対するブロードキャストを開始する．この表を使ってリクエストを処理するのはブロードキャスト用の命令からのリクエストのみである．そのため，ブロードキャスト用の命令を新たに追加した．

4. 評価

4.1 評価内容と評価方法

MNNA-bc の性能及びハードウェアコストについて評価し，MNNA との比較を行った．評価には，我々が開発した NNA-Sim（ニューラルネットワークアクセラレータ・シミュレータ）を使用した．NNA-Sim は，結合重み，プログラム，ニューロン値の初期データが，それぞれ，重みバッファ，プログラム格納用メモリ，共有メモリにセットされた状態からシミュレーションを開始する．アクセラレータ内の各メモリに初期データをセットする処理は実行サイクル数としてカウントしない．

今回評価に使用した NN（**表 2**）は，画像認識等で実際に使用されている CNN（Convolutional Neural Network）の 1 つの層を取り出したものであり，Convolutional 層（CONV），Pooling 層（POOL），Classifier 層（CLASS）の 3 種類からなる．

CONV 層は，入力データの特徴要素を識別する層である．入力ニューロン（$N_x \times N_y \times N_z^i$）に対して 2 次元のカーネル（$K_x \times K_y$）を 1 ニューロンずつスライドさせて出力計算を行う．そのため，出力ニューロンは $((N_x - K_x + 1) \times (N_y - K_y + 1) \times N_z^o)$ の 3 次元構造となる．POOL 層は CONV 層の直後に配置される層であり，入力されたデータ（$N_x \times N_y \times N_z^i$）にカーネル（$K_x \times K_y$）を K_x または K_y ニューロンずつスライドさせながら出力を計算する．そのため，出力ニューロンは $(N_x/K_x) \times (N_y/K_y) \times N_z^o$ の 3 次元構造となる．CLASS 層は CONV 層と POOL 層をまとめる層であり，1 次元に配置された入力ニューロン（N_z^i）と，同じく 1 次元に配置された出力ニューロン（N_z^o）が全結合されている．

NNA-Sim ではコア数，共有メモリ・サイズ等のアーキテクチャ・パラメータを設定することができる．これらの値は DaDianNao の論文を参考に**表 3** の値とした．また，NFU 全体のレイテンシは 2 サイクル（積和演算に 1 サイクル，活性化関数の計算に 1 サイクル）とした．

DaDianNao と同様，NNA-Sim では，共有メモリへのアクセスが DMAC によってパイプライン処理されることを想定している．今回の評価では，入力バッファの DMAC が処理可能なメモリリクエスト数の上限をコアあたり 64 とした．メモリリクエスト数がこの上限を超えると，DMAC は新たなメモリリクエストを受けつけることができなくなり，ロードを行う命令がデコードされた時点で制御プロセッサがストール（Load Blocking）する．なお，今回の評価では，出力バッファの DMAC には処理可能なメモリリクエストの上限値を設けていない．これは，出力バッファの DMAC が処理する書き込みリクエスト数は，入力バッファの DMAC が処理する読み出しリクエスト数よりも少ないためである．

MNNA-bc において積和演算回路のレイテンシを 1, 10, 100, 1000 サイクルと変化させた場合，またオンチップ・ネットワーク総バンド幅を変化させた場合の性能評価を行い，MNNA との比較を行った．

更に，DMAC アーキテクチャの変更に伴うハードウェアコストの差を，シミュレータを用いて評価した．面積とアクセス時間，動的電力，静的電力などを評価するシミュレータには，CACTI5.3 [5] を使用した．今回の実験に使用したパラメータ（共有メモリやバッファサイズ，ポートの数など）の値を**表 4** に示す．なお，今回は MNNA-bc のデータのブロードキャストの可否を判断する DMAC 内の表（表 1）に関する評価は行っていない．これは，DMAC 内の表のハードウェアコストは無視できるほど小さいと考えられるからである．現在は，DMAC 内の表のエントリ数は全リクエストの共有メモリのアドレスを格納できるように設定している（160 エントリ）が，実際はブロードキャストを行ったアドレスのエントリを別のアドレスのリクエストが再利用できるため数エントリで良いと考えられる．実際にこの表のエントリ数を数エントリに減らした場合の性能への影響は今後評価する必要がある．

4.2 性能分析

積和演算回路のレイテンシを変化させた場合の MNNA の実行サイクル数の内訳を**図 6** に示す．グラフの横軸は NN を，縦軸は実行サイクル数を表している．NN ごとの 4 本の積み上げ棒グラフは，左から順に，積和演算回路の

表 2 評価対象の NN（文献 [2] より）

layer	N_x	N_y	K_x	K_y	N_z^i	N_z^o
CLASS1	-	-	-	-	2560	2560
CLASS2	-	-	-	-	4096	4096
CONV1	256	256	11	11	256	256
CONV2	500	375	9	9	32	48
POOL1	492	367	2	2	12	12
POOL2	256	256	2	2	256	256

表 3 シミュレーション・パラメータ

Parameters	Remarks
The number of Cores	16
Shared Memory	16 ports 4MB 10 cycles
Input Buffer	16 entry 1cycle
Output Buffer	16 entry 1cycle
Weight Buffer	2M entry
Frequency	606MHz

Vol.2017-ARC-225 No.28
Vol.2017-SLDM-179 No.28
Vol.2017-EMB-44 No.28
2017/3/9

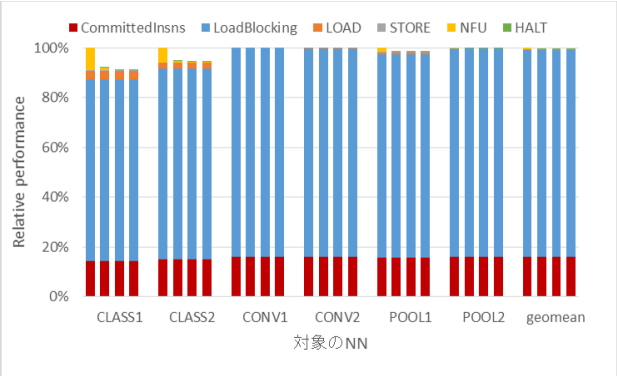

図 6 MNNA の積和演算回路のレイテンシを変更した場合の実行サイクル数の内訳（オンチップ・ネットワーク・バンド幅：100GB/s）左から 1000, 100, 10, 1 サイクル

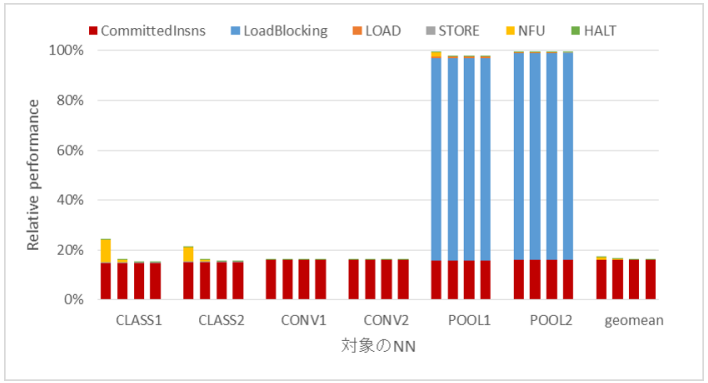

図 7 MNNA-bc の実行サイクル数の内訳の比較

レイテンシを 1000, 100, 10, 1 サイクルに設定した場合の結果である．各積み上げ棒グラフは 6 つの項目からなり，下から順に，総実行命令数，Load Blocking のサイクル数，ロードの実効サイクル数，ストアの実効サイクル数，NFU の実効サイクル数，プログラムが終了して停止していた（他コアの計算が終了するのを待っていた）サイクル数を表している．なお，各サイクル数は，16 コアの平均値であり，1000 サイクルの時の総実行サイクル数で正規化されている．

図 6 より，Load Blocking は 100GB/s のオンチップ・ネットワークを仮定した場合，総実行サイクル数の平均 86%を占める結果となった．これより，MNNA では共有メモリに対してリクエストを待つ時間が性能上のボトルネックとなっていることがわかる．

図 7 は，MNNA-bc の実行サイクル数を，1000 サイクルの積和演算回路のレイテンシを有する MNNA の実行サイクル数（図 6）で正規化し，各積和演算回路のレイテンシ毎に表したグラフである．グラフの横軸は NN を，縦軸は実行サイクル数を表している．

MNNA に対する MNNA-bc の総実行サイクル数の減少率は，積和演算回路のレイテンシが 1 の場合，最大 83%（CONV1），平均 70%，積和演算回路のレイテンシが 10 の場合，最大 84%（CONV1），平均 70%，積和演算回路のレイテンシが 100 の場合，最大 84%（CONV1），平均 69%，積和演算回路のレイテンシが 1000 の場合，最大 84%（CONV1），平均 67%であった．CLASS，CONV において，データ転送をブロードキャスト化した MNNA-bc の総実行サイクル数が MNNA 総実行サイクル数の約 1/5 になった．この結果より，メモリとコア間のデータ転送をブロードキャスト化することで，性能上のボトルネックを改善できることがわかった．

更に，バンド幅を減らして MNNA-bc の性能の変化を評価した．オンチップ・ネットワークバンド幅を 1 本あたり 9.5GB/s，総バンド幅を 38GB/s に設定した場合の MNNA-bc の総実行サイクル数の内訳を**図 9** に，オンチップ・ネットワーク総バンド幅を 1 本あたり 9GB/s，総バンド幅 36GB/s に設定した場合の MNNA-bc の総実行サイクル数の内訳を**図 8** に示す．また，CLASS，CONV に関しては，オンチップ・ネットワークの総バンド幅を 100GB/s から 38GB/s， 36GB/s まで減らしても MNNA の性能が大きく変わらないことから，低いバンド幅でも高速に NN

表 4 cacti で使用したパラメータ

MNNA				
Parameters	Shared Memory	Input/Output Buffer	Weight Buffer	DMAC register
RAMsize (byte)	8,000,000	1,024	2,097,152	512
Nr. of Banks	4	1	4	1
Read Ports	32	1	1	1
Write Ports	16	1	1	1
Read Bits size	64	64	64	64
MNNA-bc				
Parameters	Shared Memory	Input/Output Buffer	Weight Buffer	DMAC register
RAMsize (byte)	8,000,000	1,024	2,097,152	64
Nr. of Banks	4	1	4	1
Read Ports	8	4	1	16
Write Ports	4	4	1	4
Read Bits size	256	256	256	256

Vol.2017-ARC-225 No.28
Vol.2017-SLDM-179 No.28
Vol.2017-EMB-44 No.28
2017/3/9

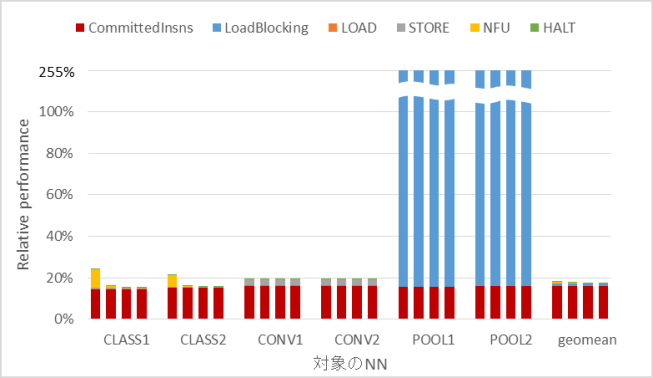

図 8 総バンド幅 36GB/s, レイテンシ 1（1 本あたりのバンド幅 9.5GB/s）の場合

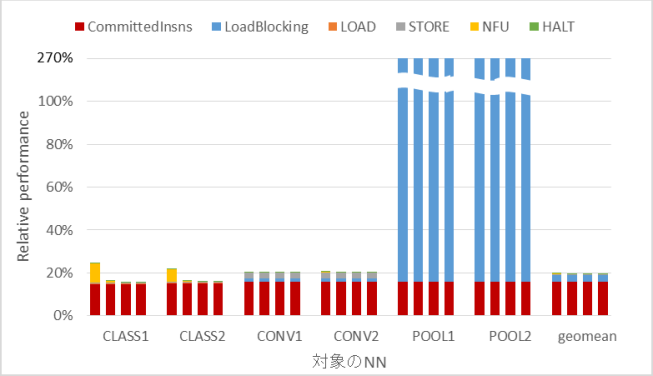

図 9 総バンド幅 36GB/s, レイテンシ 1（1 本あたりのバンド幅 9GB/s）の場合

計算を行うことができることがわかった．

4.3 ハードウェア規模の評価

MNNA と MNNA-bc のハードウェア規模の比較結果を**表 5** に示す．改良前のアーキテクチャでは，コア毎の各バッファと，各バッファ用に 3 つの DMAC レジスタを持っており，それらを合計したものを表 5 の Total としている．一方で，改良後のアーキテクチャでは，コア毎の各バッファと，コア共通で各バッファ用に 4 チャネル分の計 12 個の DMAC レジスタを持っており，それらを合計したものを表 5 の Total としている．各バッファの消費電力が増加しているが，共有メモリの消費電力は削減できることがわかった．結果として，全体のエリアは約 16%程度増加し，共有メモリの消費電力は約 4.1%程度減少した．

表 5 ハードウェア規模の比較

	エリア (mm^2)		アクセス時間 (ns)	
	MNNA	MNNA-bc	MNNA	MNNA-bc
Total	59.529	69.390	-	-
Shared Memory	39.923	16.918	8.487	5.797
Input Buffer	0.124	6.288	0.827	1.020
Weight Buffer	19.088	35.312	2.342	2.615
Output Buffer	0.124	6.288	0.827	1.020
DMAC register	0.089	1.528	0.753	1.067
	動的電力 (W)		静的電力 (W)	
	MNNA	MNNA-bc	MNNA	MNNA-bc
Shared Memory	0.097	0.093	6.80E-03	5.63E-03
Input Buffer	0.003	0.029	3.71E-08	1.30E-07
Weight Buffer	0.031	0.08	3.90E-04	4.05E-04
Output Buffer	0.003	0.029	3.71E-08	1.30E-07
DMAC register	0.003	0.009	1.98E-08	2.94E-07

5. おわりに

本研究では，MNNA の性能上のボトルネックであるメモリとコア間のデータ転送時間を短縮するために，各コアへのデータ転送をブロードキャストすることで任意の NN 計算をより高速かつ低消費エネルギーで行うことのできる MNNA-bc を提案し，性能およびハードウェアコストについて評価を行った．

メモリと各コア間のデータ転送をブロードキャスト可能にすることで，総実行サイクル数の減少率は MNNA に比べて最大 84%，平均 70%となり，ボトルネックを解消できることがわかった．また，MNNA と MNNA-bc のハードウェアコストを比較した結果，共有メモリの消費電力を約 4.1%程度減らすことができた．今後は，DMAC 内の表のハードウェアコストの評価及び SNN など様々な NN の評価も行う予定である．

謝辞 本研究の一部は JST CREST による．

参考文献

[1] Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y. and Temam, O.: DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine- learning, Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS ’14, pp. 269-284 (2014).

[2] Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N. and Temam, O.: DaDianNao: A Machine-Learning Supercompu ter, Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-47, pp. 609-622 (2014).

[3] Le, Q. V., Monga, R., Devin, M., Chen, K., Corrado, G. S., Dean, J. and Ng, A. Y.: Building high-level features using large scale unsupervised learning, In Inter- national Conference on Machine Learning (2012).

[4] Liu, S., Du, Z., Tao, J., Han, D., Luo, T., Xie, Y., Chen, Y. and Chen, T.: Cambricon: An Instruction Set Architecture for Neural Networks, Proceedings of the 43rd Annual International Symposium on Computer Architecture, ISCA ’16, pp. 393-405 (2016).

[5] CACTI 5.3, http://quid.hpl.hp.com:9081/cacti/.

[6] M. Ohba, S. Miwa, S. Shindo, T. Tsumura, H. Yamaki, H. Honda, Initial Study of Reconfigruable Neural Network Accelerators, Proc. 7th International Workshop on Advances in Networking and Computing (poster presentation), pp.707-709, (Nov 2016).

[7] S. Shindo, M. Ohba, T. Tsumura, S. Miwa, Evaluation of Task Mapping on Multicore Neural Network Accelerators, Proc. 4th Int’l Workshop on Computer Systems and Architectures, pp.415- 421, (Nov 2016).