Vol.2017-ARC-225 No.29
Vol.2017-SLDM-179 No.29
Vol.2017-EMB-44 No.29
2017/3/9

ゲートウェイにおける オンラインニューラルネットワーク学習器 ハードウェアの提案

澤口 聡太† 西 宏章†

†慶應義塾大学理工学部システムデザイン工学科 〒223-8522 神奈川県横浜市港区日吉 3-14-1
E-mail: †{sawaguchi, west}@west.sd.keio.ac.jp

あらまし 今後 Internet of Things や Trillion Sensor，Smart City といった用語に代表されるデータ社会の到来により，数多のセンサデータが取得できるようになり，多種多様なサービスの展開が予見される．斯様なサービスの創出にはデータ解析が不可欠であるが，その中でも特にニューラルネットワークが近年注目されており，クラウドのみならずゲートウェイでのアプリケーションにおいてもその利用が検討されている．しかし，通常ゲートウェイでニューラルネットワーク学習を行うことは計算リソースの観点から困難である．そこで本報告では，SoC FPGA をゲートウェイとして想定し，ニューラルネットワーク学習のハードウェア実装による電力効率向上を提案する．

キーワード Neural Network，SoC FPGA，Stochastic Gradient Descent，Gateways，Dynamic Partial Reconfiguration

Hardware Accelerator for Online Neural Network Training at Gateway

Sota SAWAGUCHI† and Hiroaki NISHI‡

†Department of System Design Engineering 3-14-1 Hiyoshi, Yokohama-shi Kanagawa, 223-8522 Japan
E-mail: †sawaguchi@west.sd.keio.ac.jp, ‡west@sd.keio.ac.jp

Abstract As the growing discussion of Internet of Things(IoT), Trillion Sensor, and Smart City, the advent of data society is predestined. A plethora of data will come out and engender the emergence of new services and applications. Under these circumstances, the advance of data analytics is prerequisite condition for pursuing the new services applications. In particular, neural network is one of the most attractive technologies that many researches have validated its usability for the applications not only in the Cloud but also at gateways. However, neural network training is highly compute-intensive for gateways from computation resources perspective. In this report, we propose that SoC FPGA should be assumed as a gateway and hardware implementation of neural network training will improve its energy efficiency

Keywords Neural Network，SoC FPGA，Stochastic Gradient Descent，Gateways，Dynamic Partial Reconfiguration

1. はじめに

Internet of Things(IoT)の進展に伴い，取得可能なデータ数が増加し，ビッグデータ解析を利用した新規サービスやアプリの創出が想定されている．IoT では数多くのセンサデバイスを我々の身の回りのモノに設置することから，個人や限られた範囲へのアプリ展開がより加速されると考えられる．実際にゲートウェイでの展開を想定した Neural Network を用いたアプリやサービスの有効性が数多く示されている．例えば，[1]ではスマートホームにおいて Neural Network を用いたユーザの行動推定による家電機器制御を提案している．またスマートホームでの位置推定において，従来手法である Support Vector Machine(SVM)と比較して，独自アルゴリズムと低コスト学習機構を用いた 1 層 Neural Network によって，精度を落とさず 50 倍のスループット向上を実現している[2]. Matlab シミュレーションにより，Neural Network を用いて無線センサネットワークのライフタイムを 43%改善できることが論文[3]にて示されている．しかし，これらの研究ではクラウドにて学習を行うことやゲートウェイの処理性能が十分であることを前提としている．上記のようなローカルを対象としたアプリは本来ゲートウェイで処理されるべきである．

IoT によりローカルを対象とした新たなアプリの展開が進展する際，ゲートウェイで処理を完結するのが好ましい．現状のゲートウェイでは多数アプリへ対応

Vol.2017-ARC-225 No.29
Vol.2017-SLDM-179 No.29
Vol.2017-EMB-44 No.29
2017/3/9

する動的性を備えているものの，処理性能が低いといった問題がある．処理性能を補うために[2]のように処理ノードを増やすことが一手段として考えられるが，消費電力の増加が懸念されるため，処理ノードを増やすことはなるべく避けたい．したがって，次世代ゲートウェイは，より高性能で低消費電力であることが求められる．

以上を踏まえ，本報告では CPU の柔軟性と FPGA の電力効率の高さを兼ね備えた SoC FPGA をゲートウェイとして利用することを想定し，Neural Network 学習のハードウェア実装を行う．FPGA へのオフロードを利用した電力効率の向上(低消費電力化と高スループット化)によるアプリのクラウドからゲートウェイへの移行可能性を，Neural Network 学習を例に示す．他の機械学習手法と比較して，Neural Network は入力と出力の関係を学習させることでその関係を数式として表現することができることから，データ解析手法の中でも汎用性が高く，今後さまざまなアプリへの利用が期待される[4]．本報告では Neural Network 学習器(逆誤差伝搬アルゴリズムのひとつである Stochastic Gradient Descent (SGD)アルゴリズム)を実装するにあたり，FPGA の動的部分再構成の利用を想定したアーキテクチャを考える．これは，次世代ゲートウェイでは Neural Network 学習に限らず，匿名化処理やプロトコル変換などさまざまなアプリを展開する必要があり，限られた FPGA リソースになるべく多くのアプリをハードウェア実装するためである．

2. 関連研究

Neural Network を FPGA に実装した既存研究を紹介する．処理高速化と低リソース化を実現するために，固定小数点を用いた研究が多く存在する[4][5][6]．Neural Network の分類[4][5]では，8-16bit のデータ精度に落とし低コスト算術演算器を利用することで，分類精度を著しく落とすことなく，処理高速化と低リソース化を行っている．Neural Network 学習[6]では，16bit のデータ精度に加え，stochastic rounding(S.R.)を丸め手法に用いることで，重み係数がゼロになり生じうる過学習を防ぎ，学習精度を落とさずに処理高速化と低リソース化を行っている．また，活性化関数に用いられる Sigmoid 関数を，固定小数点を用いた線形区分近似関数により表現することで，高性能化と低リソース化を実現している[4][7]．16bit 固定小数点のデータ精度で加算器とシフト演算器により Sigmoid 関数を表現し，分類・学習精度に大きく影響しないことを示している．本報告においてもこれらの高速化・低リソース化手法を用いる．

Neural Network に関する既存研究の問題点を挙げる．Gaowei らは，クラウドにて Hadoop の MapReduce を用いたユーザ行動推定のための Neural Network 学習を行い，ゲートウェイから家電機器制御を行っている[1]．しかし，本来ローカルで閉じたアプリであり，ゲートウェイで行われるべきである．各家庭でこのアプリが利用される場合，クラウドでの学習は現実的ではないと言える．

Hantao らは，独自アルゴリズムと低コスト学習器を用いた 1 層ニューラルネットワークによるスマートホームでのユーザ位置推定を提案している[2]．分散ゲートウェイで処理を行い，従来手法の SVM による位置推定と比較して，精度を著しく落とさずに 50 倍の性能向上を実現した．しかし，分散ゲートウェイを用いることによる消費電力増加が問題となる．

Raja らは，20 個のニューロンを用いた Neural Network により，無線センサネットワークのライフタイムを 43%延長できることが Matlab シミュレーションによって示されている[3]．この研究では十分な処理性能がゲートウェイに備わっていることが想定されている．本報告ではゲートウェイに SoC FPGA を用いることで，処理性能の向上と低消費電力化の提案を行う．

Neural Network の SGD アルゴリズムのハードウェア実装に関する既存研究について述べる．Ernesto らは，3 層 Neural Network をサポートしており，活性化関数には 3 次多項式で近似した Tanh を実装している[8]．実際の Neural Network ではアプリに応じて，Tanh のみならず，Sigmoid 関数や ReLU(Rectified Linear Unit)，線形写像がよく用いられる．また，SGD のハードウェア実装ではパイプライン機構を実装しているが，まだ空間的並列性の獲得余地があると考えられる．

一方 Alin らは，3 層 Neural Network をサポートしており，論理合成前にユーザが各層のニューロンの数を指定できる抽象化を行っている[7]．活性化関数には Sigmoid 関数を用いており，SGD のハードウェア実装では，各層の逆誤差伝播が 1 サイクルで行われるように並列化している．しかし，本報告で用いる FPGA は低リソースであり，逆誤差伝播において完全に並列化することが難しい．以上より，本報告では活性化関数に Tanh，Sigmoid，ReLU，線形写像の 4 つからユーザが指定できる 3 層 Neural Network 学習器の実装を行う．

Vol.2017-ARC-225 No.29
Vol.2017-SLDM-179 No.29
Vol.2017-EMB-44 No.29
2017/3/9

3. Neural Network

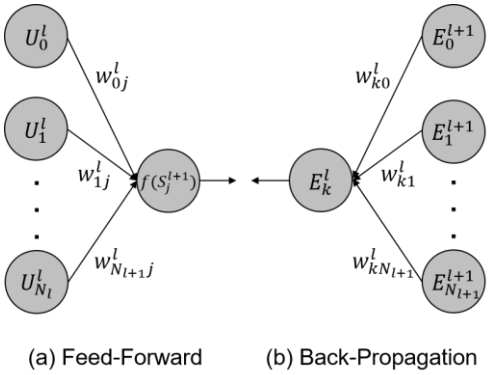

図 1. Neural Network 概要図

Neural Network の概要図を図 1 に示す．学習は，Feed-Forward, Back-Propagation の 2 つの処理から成る．まず Feed-Forward の処理フローを図 1(a)を用いて説明する．層 l の i 番目のニューロンの入力を$U_i^l(i = 0,1,\dots,N_l)$と表現する．層 l+1 の j 番目のニューロンの出力を$U_j^{l+1}(j = 0,1,\dots,N_{l+1})$とし，層 l の各ニューロンから層 l+1 の j 番目のニューロンへの重み係数w_{ij}^l，合計値S_j^{l+1}，活性化関数 f とすれば，式(1)，(2)を得る．

$$\boldsymbol{U_j^{l+1} = f(S_j^{l+1})} \qquad (1)$$

$$\boldsymbol{S_j^{l+1} = \sum_i^{N_l} U_i^l w_{ij}^l} \qquad (2)$$

次に Back-Propagation について説明する．提案機構では，計算コストと性能の改善と過学習の回避を目的とし，SGD アルゴリズムにミニバッチ学習を採用する．サンプル数をN_mとし，n 個目のサンプルにおける出力層のニューロンの出力を$y_k^n(k = 0,1,\dots,N_o)$，出力の目標値をt_k^nとすれば，逆誤差伝播に実際に用いる平均誤差E_k^oは，以下の式(3)で得られる．

$$\boldsymbol{E_k^o = \frac{1}{N_m}\sum_n^{N_m}(t_k^n - y_k^n)} \qquad (3)$$

層 l+1 から l への誤差伝播と重み更新について，層 l+1 の各ニューロンから層 l の k 番目のニューロンへの重み係数w_{kj}^l，伝播誤差E_k^lは，以下の式(4)，(5)で得られる(図 1(b))．但し，δ_j^{l+1}は式(6)で与えられるとし，ε は学習係数を表す．

$$\boldsymbol{E_k^l = \sum_j^{N_{l+1}} w_{kj}^l \delta_j^{l+1}} \qquad (4)$$

$$\boldsymbol{w_{kj}^l = w_{kj}^l - \varepsilon U_k^l \delta_j^{l+1}} \qquad (5)$$

$$\boldsymbol{\delta_j^{l+1} = f'(S_j^{l+1}) E_j^{l+1}} \qquad (6)$$

以上に示したアルゴリズムをもとに提案機構の説明を行う．

4. 提案ハードウェア機構

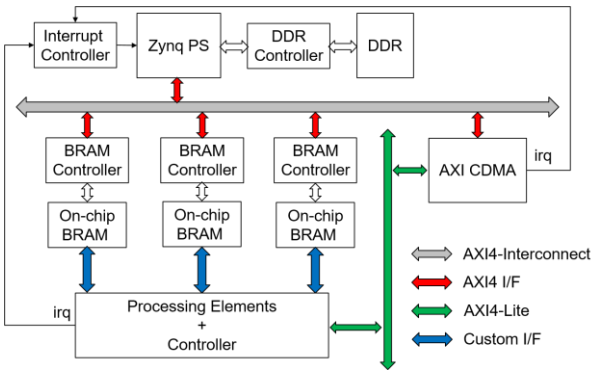

図 2. 提案機構のシステム概要図

実装環境には Zynq ZC702 の使用を想定しており，図 2 に提案機構の概要図を示す．Zynq Processing System(PS)，BRAMs，Processing Elements (PEs)，Controller の 4 モジュールからなる粗粒度アーキテクチャ[9]を実装し，PE を動的部分再構成により再構成領域として利用することを想定している．

Neural Network に用いる各 BRAM と Processing Elements(PEs) の Feed-Forward，Average Error，Back-Propagation 機構を図 3 に示す. BRAM に関して，Feature BRAM，Weight BRAM，Intermediate BRAM，Target & Error BRAM を実装し，それぞれ特徴データ，重み係数，中間データ，目標値・誤差値を格納するためのデュアルポート BRAM とする．各値は 16bit 固定小数点(1bit 符号，3bit 整数部，12bit 小数部)を用いて表現し，BRAM は 64bit 幅で実装する．これにより，データレベル並列性を獲得する．

本段落では$N_f - N_h - N_o$ Neural Network の場合の各 BRAM へのデータ格納方法について説明する．入力層と隠れ層のニューロン i，j について，n 個目の特徴ベクトルデータ・隠れ層の中間データをf_i^n, h_j^nと表現し，重み係数と誤差は，層の表現に h(隠れ層)，o(出力層)を用いている．それ以外は 3 節で定義した記号を用い

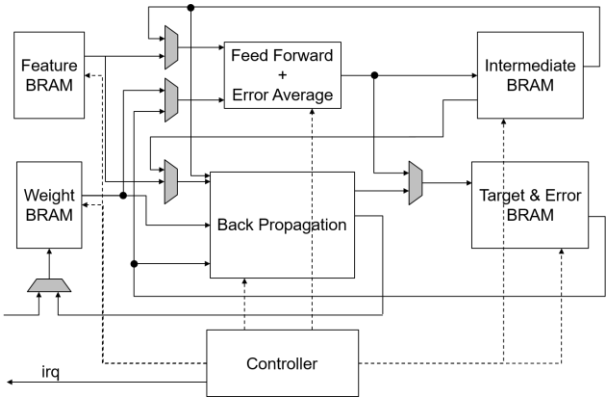

図 3. Neural Network の PEs 機構

Vol.2017-ARC-225 No.29
Vol.2017-SLDM-179 No.29
Vol.2017-EMB-44 No.29
2017/3/9

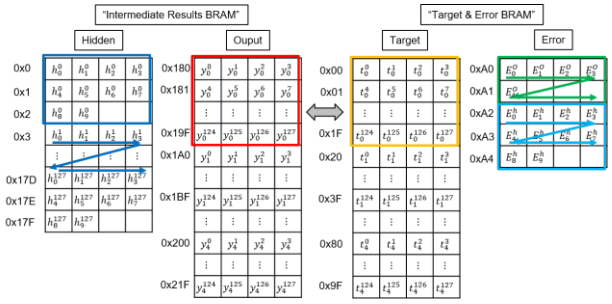

図 4. BRAM へのデータ格納例

る．特徴データ，重み係数(隠れ層と出力層)，隠れ層の中間データ，誤差(隠れ層と出力層)はそれぞれ $f_0^n, f_1^n, \ldots, f_{(N_f-1)}^n$, $w_{0k}^h, w_{1k}^h, \ldots, w_{(N_f-1)k}^h$, $w_{0m}^o, w_{1m}^o, \ldots, w_{(N_h-1)m}^o$, $h_0^n, h_1^n, \ldots, h_{(N_f-1)}^n$， $E_0^o, E_1^o, \ldots, E_{(N_o-1)}^o$， $E_0^h, E_1^h, \ldots, E_{N_h}^h$ ($n = 0, \ldots, N_m - 1$, $k = 0, \ldots, N_h - 1$, $m = 0, \ldots, N_o - 1$)の順に，あるアドレスから順に格納していく．n+1, k+1, m+1 の，もしくは異なる層のデータ群の格納に切り替わる際，そのデータ群は必ずアドレスを 1 つ移動して，順に格納していく．これを繰り返し行うことでデータをすべて格納する．7-10-5 Neural Network の場合にサンプル数 128 個でミニバッチ学習を行ったときの Intermediate BRAM と Target & Error BRAM へのデータ格納状態を図 4 に示す．出力層の出力値と目標値は図 4 に示す通り，上述した格納方法とは異なり，出力層のあるニューロンからの出力値・目標値を連続して保存することで，平均誤差算出時のデータ読み出しと並列処理の容易化を図る．

以下，PEs のアーキテクチャを説明する．図 5 に Feed-Forward+Average Error 機構を示す．MUX によって乗算器を選択すれば Feed-Forward 機構，減算器を選択すれば Average Error 機構として動作する．まず，Feed-Forward 機構に関して説明する．特徴データもしくは中間データと重み係数を入力として，Multiply-Accumulator(MAC)を用いた並列処理により式(2)の処理を行う．Adder Tree にて算出された合計値を Tanh，Sigmoid，ReLU，および線形写像いずれかの活性化関数により処理し，次層の値を取得(式(1))して Intermediate BRAM に格納する．図 3 に示すように，MUX を用いて入力層から隠れ層へのデータ伝播の場合は Feature BRAM からの特徴データ，隠れ層から出力層の場合は Intermediate BRAM からの中間データを選択する．

Average Error 機構の説明を行う．Feed-Forward 機構により得られた出力値と目標値を入力とし，ミニバッチ学習のために式(3)に示す平均誤差を取得する．ミニバッチを 16，32，64，128，256 に制約することで，シフト演算による平均誤差の取得を可能とし，性能向上とリソースの削減を図った．

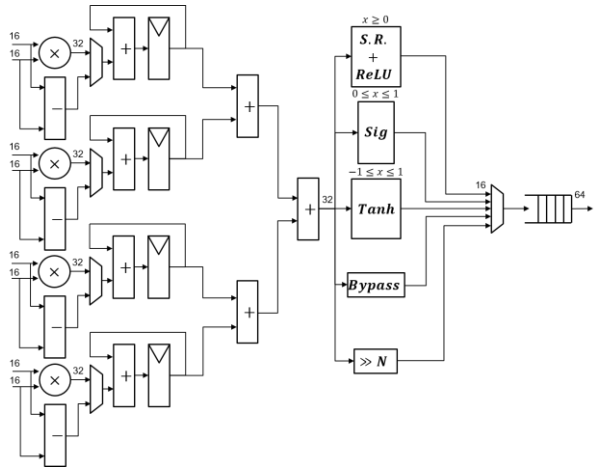

図 5. Feed-Forward+Average Error 機構

Back-Propagation 機構を図 6 に示す．前層の伝播誤差と重み更新を並列に行うために，パイプライン並列性とアルゴリズムレベル並列性(空間的並列性)を実現する．パイプライン機構は図 6 に示すように S.R.モジュールの後ろにレジスタを配置して実現する．後者の並列性は，式(4)，(5)に含まれる，2 つの値の乗算結果と他の乗算結果または被乗算数を並列実行できるように実装した．式(6)の δ_j^{l+1}内の $f'(S_j^{l+1})$に関して，Tanh と Sigmoid の微分は式(7)で表現できるので，図 6 に示す赤枠内の回路により，Tanh，Sigmoid，ReLU，線形写像をサポートする．なお，実際は図 6 に示す機構を 4 つ並列に配置し，前層の伝播誤差算出と重み更新の並列処理を行う．各層のニューロン数次第では重み更新が遅れる場合があるが，SGD は誤差を最小化する学習手法であり，最適化が行われると考えている．

$$f'(S_j^{l+1}) = \begin{cases} f(S_j^{l+1})\left(1 - f(S_j^{l+1})\right) & (sigmoid(x)) \\ 1 - f(S_j^{l+1})^2 & (\tanh(x)) \end{cases} \quad (7)$$

最後に Neural Network 学習の処理フローを説明する．まず，学習に使用する特徴データ・目標値・重み係数を含むすべてのデータを DDR に格納する．ミニバッチに用いるデータ転送分だけ特徴データと重み係数の Buffer Descriptor (BD)を作成する．

作成した重み係数の BD を AXI CDMA に書き込み，

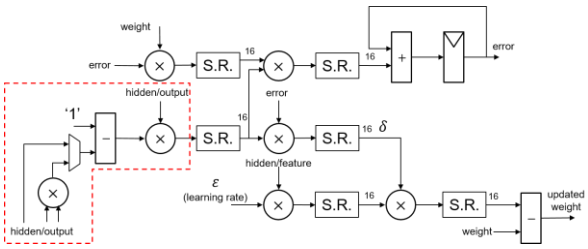

図 6. Back-Propagation 機構

Vol.2017-ARC-225 No.29
Vol.2017-SLDM-179 No.29
Vol.2017-EMB-44 No.29
2017/3/9

各 BRAM へデータ転送を開始する. 転送が終了すると DMA から割り込み(irq)を受け，PS 側は特徴データの BD を DMA に書き込み，データ転送を行う．これらデータ転送の間に PS 側で並列して，AXI4-Lite バスを介した制御情報の PL 内の Controller への書き込みや目標値の BD の作成などを行う．特徴データのデータ転送完了を知らせる DMA 割り込み後，Controller に開始ビットをアサートすることで，PL の処理を開始する．その後，目標値の BD を DMA に書き込み，目標値データをロードする．また，PL 側の処理終了までに次のミニバッチで使用する特徴データの BD を準備する．1Epoch の処理が終了すると，Controller から PS へ irq が生じ，次のミニバッチに必要なデータのロードを行うために，DMA に新たな BD を書き込む．データ転送が終了して DMA が irq を生成すると，学習処理を再開する．以上に示した制御情報の Controller への書き込みを除くプロセスを，ユーザが指定した Epoch 数回だけ繰り返し行った後，Controller の irq によって学習終了とする．

5. 論理合成と配置・配線

本報告では，現在のハードウェア実装状況と Vivado による論理合成，配置配線結果について述べる．現在実装済みのデザインでは，活性化関数として Sigmoid, ReLU，線形写像をサポートしている．図 7 に提案機構のフロアプランを示す．また，図 8 に本提案システム全体で使用されるリソース量，**エラー! 参照元が見つかりません。**にリソースを多く使用している主要なモジュールを示す．LUT に関して，システム全体の使用率は 18%程度であり，それに対して Controller，FFNN(Feed-Forward Neural Network)，BPNN(Back-Propagation Neural Network)での使用量はそれぞれ 3.9%, 2.0%, 1.4%となっている. 図 8 で BRAM 使用率が 57%であることを考慮すると，それぞれが互いに独立している特徴データを格納することから

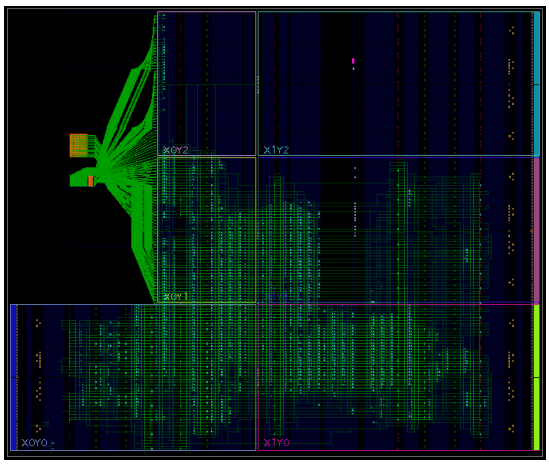

図 7. 提案機構のフロアプラン

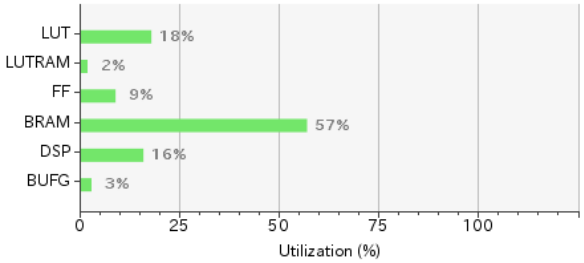

図 8. 全体の HW リソース使用量

表 1. モジュール別 HW リソース使用量

	LUTs (106,400)	Registers (53,200)	DSPs (220)
AXI CDMA	1,524(2.9%)	2,300(2.2%)	0
AXI Interconnect	2,021(3.8%)	2,180(2.1%)	0
FFNN	704(1.4%)	461(0.5%)	4(1.9%)
BPNN	1,048(2.0%)	1,392(1.4%)	32(14.6%)
Controller	2,073(3.9%)	737(0.7%)	0

Feature BRAM を分割し，それぞれの BRAM に対して，FFNN や BPNN，Controller モジュールを用意することでプロセスレベル並列性を獲得でき，更なるスループットの向上が期待できる．それぞれの学習データが独立しているため，複数の Feature BRAM それぞれに Controller を一つずつ用意することで，プロセスレベル･パイプライン並列性を獲得することも可能である．また，動的部分再構成を用いることで，ヘテロな機械学習アプリケーションの切り替えの提案も可能である．

6. 電力・スループット

次に電力効率評価に関して説明する．まず，電力計測結果を述べる．例として，Zynq ZC702 では図 9 に示すように，直流電源装置を接続し，ボード全体の電圧・電流を計測した．Zynq ZC702 チップの電力は Vivado の静的電力解析により取得した．比較対象に Raspberry Pi3 Model B(以下 Raspi3)を使用し，同様に直流電源装置にて電力測定を行った．表 2 に現在の評価環境，表 3 と表 4 に Zynq ZC702 と Raspi3 それぞれの電力測定結果を示す．表中の NN は Neural Network を意味し，MNIST 学習は 784-10 Neural Network で行い，Epoch 数は 1000 とした．それぞれのボードの消費電力を比較すると，Zynq は Raspi3 と比較して IDLE 時でおよそ 6 倍程度の電力を消費していることがわかった．Zynq はあくまでも評価ボードであることも考慮し，今後の電力比較では両者のチップの電力消費量で比較することを考えた．

スループットについて述べる．本提案機構の実機動作はまだ確認できていないが，FFNN，Average Error，BPNN 機構のレイテンシはそれぞれ 5，5，3cycle であり，784-10 Neural Network にて 128 個のデータでミニバッチ学習を 1000Epoch 行う場合，概算値でそれぞれ 250,885, 325, 250,883 cycle/Epoch であった．動作周波

Vol.2017-ARC-225 No.29
Vol.2017-SLDM-179 No.29
Vol.2017-EMB-44 No.29
2017/3/9

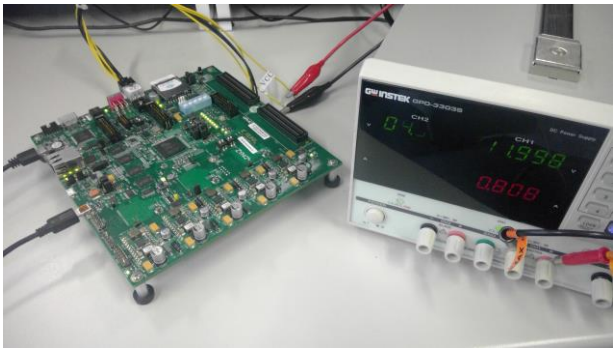

図 9. Zynq ZC702 と直流電源装置

数を 100MHz とすると，処理時間は 5.02s と算出された．データ転送時間に関して，AXI HP Port を用いた場合，最大で 400MB/s のバンド幅であり，データ転送量は概算値で 204MB であることから 1000Epoch で約 0.6s かかると予想される．データ転送と処理を含めて合計で 6s 程度となり，表 4 から Raspi3 と比較して 3 倍スループットの向上が見込める．

表 2. 実験環境

	Voltage(V)	OS	NN
Zynq ZC702	11.998	None (Bare Metal)	FPGA
Raspi3	4.998	Raspbian GNU/Linux8.0	Tensorflow

表 3. Zynq ボードとチップの消費電力

Zynq ZC702	Current(A)	Power(W)	On-chip Power(W)
IDLE (Only Zynq PS)	0.808	9.696	1.717
IDLE (NN FPGA config.)	0.879	10.546	1.889

表 4. Raspi3 の各条件下での消費電力

Raspi3	Current(A)	Power(W)	Proc. Time(s)
IDLE(OS/HDMI)	0.317	1.584	-
MNIST(784-10) Learning	0.360-0.454	1.999 (0.400A)	18.324

7. 今後の課題

本論文では，ゲートウェイに動的部分再構成が可能な SoC FPGA を利用することを想定し，Neural Network 学習のハードウェア実装による高速化と電力効率の向上を図る提案を行った．今後の課題として，Tanh を含めた実装を行い，MNIST 評価指標を用いて提案機構の動作を確認する．その後，更なるシステム改善を行うために，BRAM を分割してプロセスレベル並列性やプロセスレベルのパイプライン並列性の獲得を目指す．また，匿名化処理やプロトコル変換のハードウェア実装を行い，FPGA の動的部分再構成を用いた多様なアプリの高速化を目指す．

謝　辞

本研究は、公益財団法人セコム科学技術振興財団助成、文部科学省科学技術研究費基盤研究 B (JP16H04455)、平成２８年度総務省委託研究開発「スマートコミュニティサービス向け情報通信プラットフォームの研究開発」、国土交通省 住宅・建築物技術高度化事業ならびに慶應義塾大学グローバルスマート社会創造プロジェクト研究の一環としてなされた。

文　献

[1] X. Gaowei, L. Min, L. Fei, Z. Feng , S. Weiming, “User Behavior Prediction Model For Smart Home Using Parallelized Neural Network Algorithm,” Proceedings of the 2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design, 2016.

[2] H. Hantao, C. Yuehua , Y. Hao, “ Distributed-neuron-network based Machine Learning on Smart-gateway Network towards Real-time Indoor Data Analytics,” 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016.

[3] V. Raja , P. Rajalakshmi, “Neural Network based Short Term Forecasting Engine To Optimize Energy And Big Data Storage,” 2015 IEEE 39th Annual International Computers, Software & Applications Conference, 2015.

[4] L. Sicheng, L. Xiaoxiao, M. Mengjie, L. Hai, C. Yiran, L. Boxun , W. Yu, “Heterogeneous Systems with Reconfigurable Neuromorphic Computing Accelerators,” 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 2016.

[5] Q. Jiantao, W. Jie, Y. Song, G. Kaiyuan, L. Boxun, Z. Erjin, Y. Jincheng, T. Tianqi, X. Ningyi, S. Sen, W. Yu , Y. Huazhong, “Going Deeper with Embedded FPGA Platform for Convolutional Neural Network,” FPGA'16: Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Array, 2016.

[6] G. Suyog, A. Ankur, G. Kailash , N. Pritish, “Deep learning with limited numerical precision, ” Proceedings of the 32nd International Conference on Machine Learning, 2015.

[7] T. Alin , C. Jeannette, “An End-User Platform for FPGA-Based Design and Rapid Prototyping of Feedforward Artificial Neural Networks With On-Chip Backpropagation Learning, ” IEEE Transactions on Industrial Informatics Vol. 12, No.3, 2016.

[8] O.-C. Ernesto , d.-T. Rene, “MLP Neural Network And On-Line Backpropagation Learning Implementaion In A Low-Cost FPGA,” GLSVLSI'08 Proceedings of the 18th ACM Great Lakes symposium on VLSI, 2008.

[9] K. Abhishek, L. Douglas , A. Suhaib, “ Are Coarse-Grained Overlays Ready for General Purpose Application Acceleration on FPGAs?,” 2016 IEEE 14th Intl Conf on DASC /PiCom /DataCom /CyberSciTech, 2016.