ニューラルネットワークFPGA実装のための積和演算設計手法

松永 多苗子[1,a)]　松永 裕介[2]

概要：近年，ディープラーニングを用いたシステムが画像認識や手書き文字認識などを含め，幅広い分野で活用されるようになってきた．本稿では，エッジ側で比較的安価にディープラーニングの推論エンジンを実現することを目標として，主たる演算である積和演算を FPGA 上で実現するための設計手法を対象とするものである．学習済みのニューラルネットワークを対象とするため，大量に現れる積和演算は，定数との乗算の総和になることを踏まえ，DSP での実装と多入力加算をアダーツリーでの実装を比較する．

キーワード：ディープラーニング，ニューラルネットワーク，FPGA，積和演算, 多入力加算

An Approach to Implement Multiply-Accumulation Operations for realizing Nerual Network on FPGAs

Matsunaga Taeko[1,a)]　Matsunaga Yusuke[2]

Abstract: Recently, deep learning based systems are widely used for various areas such as vision and speech recognition. In this paper, an approach to implement multiply-accumlation operations on FPGAs are discussed to realize pre-learned Nerual Networks on FPGAs in edge-computing environments. multiply accumulation operations are treated as multiplication with constant values, and multi-operands additions are realized on FPGAs as adder trees without DSP blocks.

Keywords: deep learning, neural network, FPGA, multiply-add function, multi-operand addition

1. はじめに

近年，人工知能や、その一分野であるディープラーニングに大きな関心が集まっている．ディープラーニングは機械学習の一手法であり，大量のデータから何らかの知見を得るための手法である．古くから研究されていたニューラルネットワークを発展させたもので，マシンパワーが向上したことや，インターネットにより大量のデータを扱う環境が発生したことが，その発展の原動力となっている．現在では 様々なディープラーニング用のフレームワークが利用できる状況にあり，画像の判定，手書き文字認識，自動翻訳などを筆頭に，幅広いユーザ層の下で，様々な分野に対して活用されている．近年広がりつつある IoT 分野でも，同様に利用範囲が拡大している．

IoT でセンサから得られたデータを処理する際には，インターネットを通じてデータセンタに集約して処理を行うクラウドピューティングが広く利用されている．クラウドコンピューティングでは集約したデータに対して，計算能力の高い CPU や GPU を用いて処理を行うことが可能である．ただし，データ量が膨大になると通信量が膨大になり，レスポンスの速さの面で課題がある．センサデバイスとクラウド間の通信バンド幅を増やして対応することも考えられるがコストがかかる．一方，クラウド上に集約するのではなく，センサデバイスに近い場所で処理を行う，エッジコンピューティングでは，エッジ側の計算能力は高

1　日本文理大学工学部情報メディア学科
〒 870-0397 大分市一木 1727
2　九州大学大学院システム情報科学研究院

a)　matsunagatk@nbu.ac.jp

ⓒ 2019 Information Processing Society of Japan

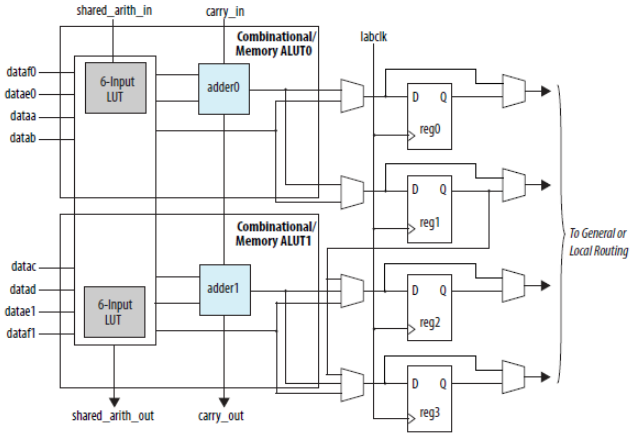

図 1 Cyclone V の ALM の構造 [3]

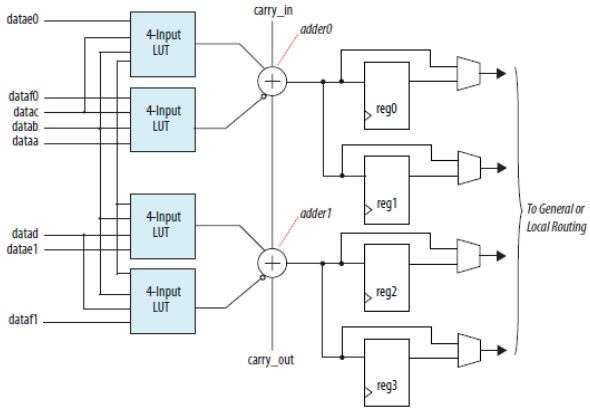

図 2 Arithmetic mode の構成

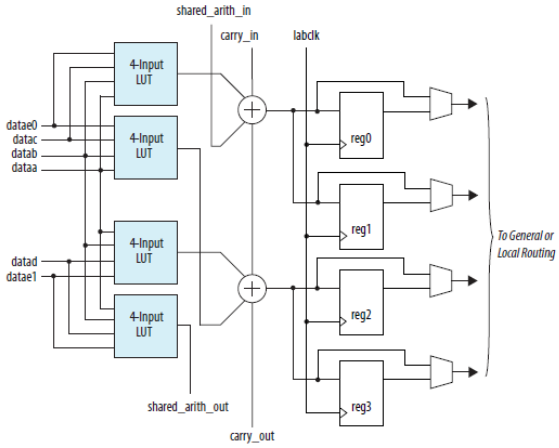

図 3 Shared arithmecit mode の構成

くはないものの，ネットワークコストが抑えられ，リアルタイム処理に向いているというメリットがある．

本研究の目的は，IoT 機器などのエッジ側において，ディープラーニング推論フェイズの処理を FPGA 上に高速かつ安価に実現するための 設計手法を確立する ことである．これまでにも，学習済みニューラルネットワークの推論エンジンを FPGA 上で実現する手法は様々に提案されている [4],[5]．本稿では特に，演算の大半を占める積和演算の実現方法に焦点を当てて，実現方法の比較・検討を行うものである．

2. 対象とする FPGA

本研究では，ハイエンドではない FPGA 上に学習済みニューラルネットワークを実装することを目的として，特にその演算の大きな部分を占める積和演算の実装方法を検討する．対象とする FPGA として，Intel 社の Cyclone V [3] を取り上げる．

Cyclone V の基本的な構成要素はアダプティブ・ロジック・モジュール (ALM) と呼ばれ，図 4 に示す構造をもっている．図中左端にある，上下 2 つの LUT を囲った四角い部分はアダプティブ LUT（ALUT) と呼ばれ，４つのモードが存在する．積和演算の実装においては，特に Arithmetic mode（図 2） と Shared arithmetic mode（図 3）が関係し，それぞれキャリーチェインを高速に実行する構造を活かして 2 入力ないし 3 入力の加算を高速に実行することができる．

一方，図??は Cyclone V の可変精度 DSP ブロックの構造を示している．3 つの 9x9 乗算器，2 つの 18（符号あり・符号なし）x 19（符号あり）乗算器，1 つの 27x27 乗算器，の 3 つの動作モードにコンフィギュレーションすることができる．

3. アプローチ

ディープラーニングが用いる多層のニューラルネットワークを図 5 に示す．ネットワークを構成する丸いノードはニューロンモデルに相当し，各ノードにおいては入力値に重みをかけて足し合わせる積和演算と，ニューロンの興奮状態を表す信号に変換する活性化関数から構成される．したがって，ディープラーニングにおいて実行される主要な関数は，積和演算と活性化関数ということになる．これらを，推論精度に影響のない範囲で，できるだけ簡単な回路になるように近似する手法を確立することが目的となる．

積和演算を実装するにあたって，本稿で取るアプローチのポイントは以下の 2 点である：

- 学習済みニューラルネットワークにおいて入力値の重みはすでに得られていることを踏まえ，定数乗算の多入力加算という枠組で扱う．
- 積和演算は DSP ブロックを用いて実現されるが，DSP ブロックは使わず ALM のみを用いる．ただし，多入力加算器の実装においては，デバイス固有のキャリーチェインを利用してアダーツリーを構成する．

3.1 定数乗算

一般に，2 つの n ビット入力の乗算は，図 6 に示すように部分積を生成し足し合わせることによって $2n$ ビットの

ⓒ 2019 Information Processing Society of Japan

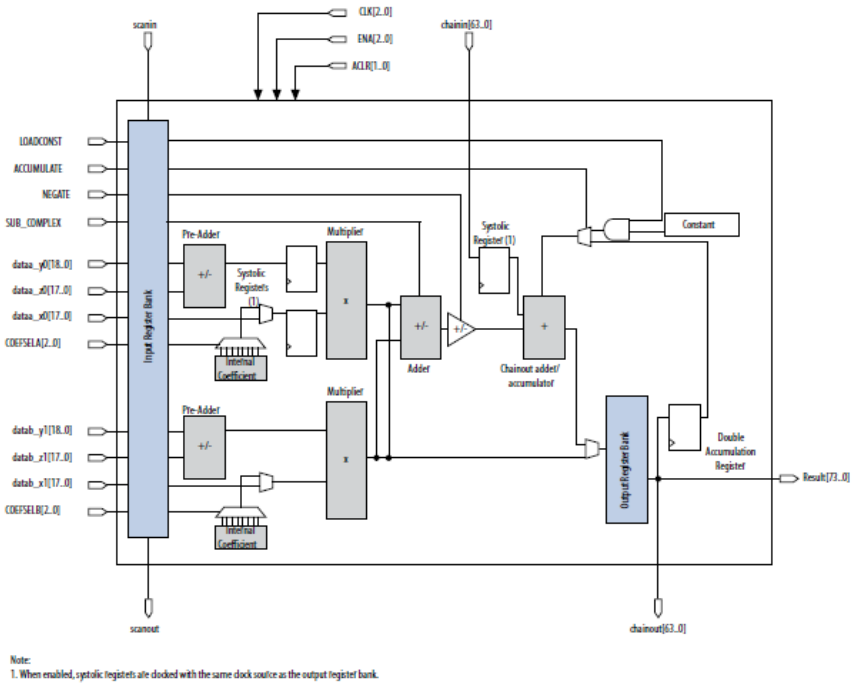

図 4　DSP

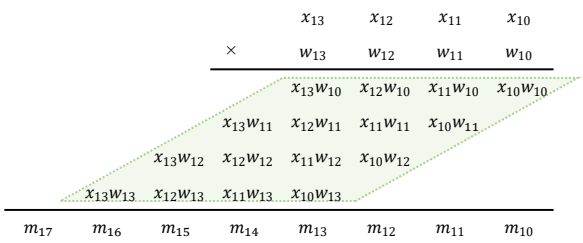

図2　乗算における部分積加算構造

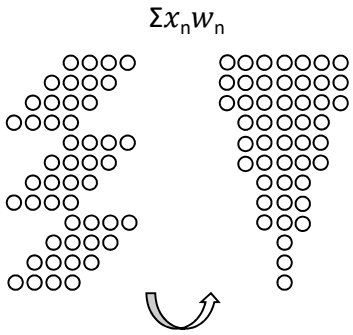

図3　積和演算における部分積加算構造

結果を得る．しかし，入力データの片方が定数であることがわかっている場合は，すべてのビット対の論理積を取ることは不要になり，値が 0 との部分積を削除できるため，より小さな回路で実現できることになる．

実際には，すべての重みを回路の中に埋め込む形で実装することが可能かどうかは，問題の規模や使用するデバイスの種類に依存する，ここでは，それが可能であった場合，あるいは，可能な範囲で埋め込んだ場合の効果を検証することを目的とする．

3.2　多入力加算

図 3 において，乗算における 1 ビットごとの論理積（例えば，a_0b_0）を白丸で表すとすると，積和演算は図 3(b) 左に示すように乗算が複数足し合わされる形になる．

ASIC において 3 個以上のオペランドの和を計算する場合，Wallace tree のようにまず部分積が 2 個になるまで圧縮を行い，最後に桁上げ伝搬加算器で結果を得ることで，桁上げが伝搬することによる遅延の影響を抑える方法が効果を上げることが多い．一方 FPGA においても，部分積数を圧縮する方式 [6] が考案されているが，キャリーチェインや DSP ブロックなどの，デバイスに備わっている算術演算回路用の構造を利用することで，速度・電力消費的にも優れた回路が実現できると言われている．特に多入力加算においては，キャリーチェイン構造を利用してアダーツリーを構成することが効果的であることを筆者らも確認している．したがって，今回は部分積は圧縮せず，アダーツリーで実現すること，特に，Cyclone V で実現されている，3 入力加算の実行モードを利用する方式を取ることと

ⓒ 2019 Information Processing Society of Japan

$$
\begin{array}{cccccccccccccccc}
 & & & & & & & & & & & & & & a_7 & a_6 \\
 & & & & & & & & & & & & & & a_5 & a_4 \\
 & & & & & & & & & & & & & & a_3 & a_2 \\
 & & & & & & & & & & & & & & a_1 & a_0 \\
 & & & & & & & & & & & & & & b_7 & b_6 \\
 & & & & & & & & & & & & & & b_5 & b_4 \\
 & & & & & & & & & & & & & & b_3 & b_2 \\
 & & & & & & & & & & & & & & b_1 & b_0 \\
\hline
 & & & & & & & & & & & & & a_7b_0 & a_6b_0 & a_5b_0 \\
 & & & & & & & & & & & & a_7b_1 & a_6b_1 & a_5b_1 & a_4b_1 \\
 & & & & & & & & & & & a_7b_2 & a_6b_2 & a_5b_2 & a_4b_2 & a_3b_2 \\
 & & & & & & & & & & a_7b_3 & a_6b_3 & a_5b_3 & a_4b_3 & a_3b_3 & a_2b_3 \\
 & & & & & & & & & a_7b_4 & a_6b_4 & a_5b_4 & a_4b_4 & a_3b_4 & a_2b_4 & a_1b_4 \\
 & & & & & & & & a_7b_5 & a_6b_5 & a_5b_5 & a_4b_5 & a_3b_5 & a_2b_5 & a_1b_5 & a_0b_5 \\
 & & & & & & & a_7b_6 & a_6b_6 & a_5b_6 & a_4b_6 & a_3b_6 & a_2b_6 & a_1b_6 & a_0b_6 & \\
 & & & & & & a_7b_7 & a_6b_7 & a_5b_7 & a_4b_7 & a_3b_7 & a_2b_7 & a_1b_7 & a_0b_7 & & \\
\end{array}
$$

(a) 8 ビット乗算

$$
\begin{array}{cccccccc}
a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\hline
a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
\end{array}
$$

(b) 8 ビット乗算で一方が定数の場合の例

図 6　8bit 乗算と定数乗算

図 5　ニューラルネットワーク

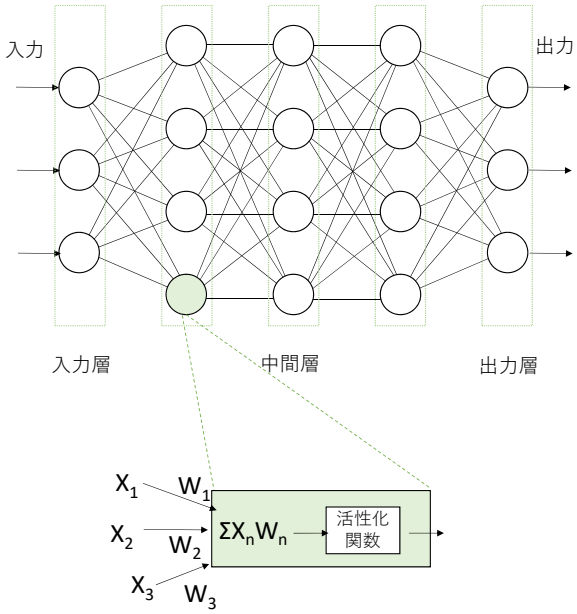

表 1　実験結果

	積和演算	定数乗算の総和
ALM	78	395
DSP block	9	0
Delay(ns)	24.5	13.4
Total power(mW)	977.28	770.99

している．

4.　実験結果

上述したような定数乗算とアダーツリーを組み合わせた手法と，通常の積和演算として DSP ブロックを用いて実現する場合の比較実験を行った．重みを表す定数はランダムに発生させ，8 ビット長とする．加算する個数は 20 とした．両方とも Verilog HDL で記述を行い，設計ツールは Quartus Prime 17.0 を用いた．

表 1 に結果を示す．

表 1 からわかるように，乗数が定数であることを考慮しない場合は，主に DSP ブロックを用いて実現されている．一方，定数を考慮した場合は DSP は使われず，ALM のみを用いて実現された．そのうちおよそ 6 割程度が shared arithmetic mode，残り 4 割程度が arithmetic mode で実現されており，Cyclone V のキャリーチェインおよび 3 入力加算の機能を利用してアダーツリーが構築されていることが確認された．また，今回の実験で一例に過ぎないが，DSP を用いた実装のほうが，最大遅延および消費電力が大きくなることが確認された．

5.　おわりに

本稿では学習済みニューラルネットワークを FPGA 上に実装するにあたって，積和演算を定数乗算の多入力加算

ⓒ 2019 Information Processing Society of Japan

として扱うことによって，DSP で実現するよりも小規模・高速・低電力を実現できる可能性を示した．定数 0 の部分が多いほど，回路規模は縮小できる．このことは，精度をできるだけ落とさずにビット幅を縮小することを考える際に，定数を表す 2 進数において 0 が多くなるような変換方法や，部分積加算構造を変更する方法など，これまでとは異なるアプローチの可能性を示唆している，

参考文献

[1] K. Guo, S. Zeng, J. Yu,Y. Wang, and H. Yang, “A Survey of FPGA-based Nerual Network Inference Accelerators”, in *ACM Transactions on Reconfigurable Technology and Systems*,vol.12, no.1, article 2. , March 2019.

[2] 高野　成幸，「THINKING　MACHINES　機械学習とそのハードウェア実装」，インプレス R&D，2017 年.

[3] “Cyclone V Device Handbook: Volume 1: Device Interfaces and Integration”, CV-52001, 2019.

[4] 中原啓貴，笹尾勤，岩本久，「Nested RNS の定数除算を用いた深層畳込みニューラルネットワークの FPGA 実現について」，信学技法　VLD2015-108,CPSY2015-140,RECONF2015-90，2016 年.

[5] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, “Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks”, in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field Prograaammable Gate Arrays*,FPGA’15, 167-170, 2015.

[6] T. Matsunaga, S. Kimura, and Y. Matsunaga, “Multi-operand adder synthesis targeting LUT-based FPGA”, in *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, vol. E94-A, No. 12, 2579-2568, December 2011

ⓒ 2019 Information Processing Society of Japan