Vol.2017-SLDM-181 No.30
Vol.2017-EMB-46 No.30
2017/11/7

複数 FPGA を用いたスパイキングニューラルネットワークシミュレーションの高速化

岡本朋大[†1]　川尾太郎[†2]　河野崇[†3]　藤田昌宏[†4]

概要：近年，深層学習をはじめとして CPU の計算量が増大しており，並列計算能力や消費電力の面で優れた FPGA がアクセラレータとして用いられることがある．先行研究では，DSSN という人間の脳の活動を数学的に表した複雑なネットワークのモデルを取り上げ，1 つの FPGA に実装することによって 768 個のニューロンについてシミュレーションを行うことができた．しかし 1 つの FPGA ではチップ上の記憶容量が不十分であるため，複数の FPGA を用いることでさらに多くのニューロンを実装することを目指す．その際，ボトルネックとなるのが FPGA 間の通信遅延である．この研究では通信量を減らすために，リング状に FPGA を接続し，各 FPGA に計算分割を行う手法を提案する．

Spiking Neural Network Simulation Accelerator Using Multiple FPGA Chips

TOMOHIRO OKAMOTO[†1]　TARO KAWAO[†2]
TAKASHI KOHNO[†3]　MASAHIRO FUJITA[†4]

1. はじめに

近年，機械学習や精緻なシミュレーションなどに代表されるような科学技術計算における計算量の増大から，CPU のみで計算を行うのではなく，アクセラレータとして GPU や FPGA(Field Programmable Gate Array)，ASIC を用いることが増えてきている．GPU(Graphical Processing Unit)は多数のコア，スレッドによる並列処理，階層的なメモリなどの特徴から計算性能が高く，ディープラーニングの計算にも用いられる一方で，消費電力が大きい．ASIC(Application Specific IC)は，特定の計算用のチップを作ることで，高性能かつ低消費電力な計算が可能であるが，開発に時間的，金銭的コストを要する．FPGA はプログラムを書き換えることによって再構成可能であるという柔軟性と，ハードウェア的な高速処理，低消費電力であるという性能から注目を浴びている．ASIC における設計の前段階としても用いられることがある．FPGA をアクセラレータとして用いた例としては，Neuroflow と呼ばれるニューラルネットワーク計算専用のプラットフォーム，Microsoft の「Catapult」などが挙げられる[1][2]．特に後者は，縦横 6×8 個の FPGA をトーラス状に並べた構造になっており，ディープラーニングなどの例題に対しても高速化を実現し,高いエネルギー効率を示している．このように FPGA を用いた高速化において，複数のボードをつなげて計算性能を上げていくような研究も増えてきている．複数の FPGA を用いる場合，FPGA ボードのゲート数が一般的に 100 万ゲート程であるのに対し，入出力のピンは 1000 程しかない．このことより複数の FPGA を用いる際には FPGA 間のデータ転送遅延がボトルネックとなり得る．

本研究室で行われた先行研究として，DSSN モデル[3]というスパイキングニューラルネットワークのモデルについて，FPGA を用いてシミュレーションを行ったものがある[4][5]．この先行研究は，人間の脳に近いモデルを，FPGA で高速に計算することにより，脳機能の解明や，ロボットへの応用などを目指すものであった．このモデルでは，シミュレーションを行う神経細胞の数を増やすことによってより正確なシミュレーションを目指しているが，１つの FPGA のみしか用いていなかったため，ボード上のメモリの不足により，実装できる神経細胞の数に限界があった．

そこで本研究では，より多くのニューロンを実装するため，複数の FPGA を用いることとした．その際 FPGA 内のデータ転送に比べ FPGA 間の通信は低速となるため，本論文ではなるべく FPGA 間の通信量が少なくなるような分割手法を提案する．今回扱うモデルに含まれる密行列計算は他のニューラルネットワークのモデルでも扱うような汎用な計算であり，その効率的な分割手法は応用の分野が広いと考える．

2. 提案手法

2.1 FPGA

FPGA は主に 3 つの基本要素からなる．論理回路部分であるロジックセル，入出力ポート，それらの要素を配線でつなぐプログラマブルスイッチである．ロジックセルは，

†1 東京大学大学院工学系研究科電気系工学専攻
Department of Electrical Engineering, The university of Tokyo.
†2 ルネサスエレクトロニクス(株)
Renesas Electronics Corporation.
†3 東京大学生産技術研究所
Institute of Industrial Science, The University of Tokyo.
†4 東京大学大規模集積システム設計教育研究センター
VLSI Design and Education Center, The University of Tokyo.

ルックアップテーブル（LUT）とフリップフロップ(FF)によって構成される．ルックアップテーブルとは SRAM を用いた値の対応表であり，例えば Xilinx の Virtex7 というデバイスでは 6 入力の LUT が一般的に用いられている．1 つの LUT では 2^{2^6}通りの論理関数を実現することが可能である．Virtex7 では 4 つの 6 入力 LUT と 8 つの FF が 1 つのスライスを構成し，このスライスが 1 つの FPGA ボードの中に 16000 枚搭載されている[6]．また 3 つの基本要素以外にも DSP ブロック(Digital Signal Processing)が演算性能を上げるために組み込まれている．この DSP ブロックではディジタル信号処理のように頻繁に行われる乗算，加算に対応したもので，LUT ではなく，乗算器や加算器がハードウェアとして固定で置かれている．乗算器を LUT で実装しようとした場合，多くの LUT とそれをつなぐ複雑な配線を要するが，固定の乗算器と加算器をうまく用いることにより，効率の良い回路構成が見込める．もちろん，DSP の面積分，LUT のリソースが減ってしまうため，DSP を活かすような設計が設計者には求められている．このように FPGA でも CPU チップなどと同様に，集積化と高速化のための工夫が現在も行われている．

2.2 DSSN モデル

本研究で扱うモデルは，DSSN(Digital Spiking Neural Network)モデルというニューロンの活動電位の再現性の高さが特徴的なモデルである．計算量が多いが，その名の通り，ハードウェアでの実装が用意となるように定数の値が決められている．ニューロンは図 1 のようにお互いが全結合したネットワークを形成していて，ニューロン数を N とするとN^2個のシナプスがある．データの受け渡しが毎ステップ行われるため複雑なネットワークとなる．DSNN モデルの計算は，ニューロン内部での活動電位の計算，シナプスへの出力，シナプスからの入力の部分に分かれている．

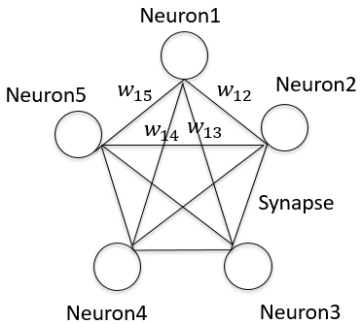

図 1　全結合ネットワーク

図 2 で表されるように各ニューロンは他のニューロンへシナプスを通して $Is^1, Is^2 .. Is^N$ を出力する．式(1)に従ってニューロン j からニューロン i への結合強度 w_{ij} と加重和を取ることで，ニューロン i への入力である刺激電流 I_{sim}^i が得られる．この式はニューラルネットワークに共通である部分であるが，w_{ij}が全結合であるため計算量が $O(N^2)$ であること，この結合強度はほとんどの値が 0 ではない密行列であることが DSNN の特徴的である．

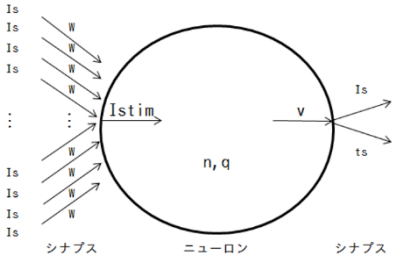

図 2 ニューロンの構造と入出力

$$I_{stim}^i = \frac{c}{N}\sum_{N}^{j=1} w_{ij} I_s^j \quad (1)$$

$$\frac{dv}{dt} = \frac{\varphi}{\tau}(f(v) - n - q + I_0 + I_{stim}) \quad (2)$$

$$\frac{dn}{dt} = \frac{1}{\tau}(g(v) - n) \quad (3)$$

$$\frac{dq}{dt} = \frac{\varepsilon}{\tau}(v - v_0 - \alpha q) \quad (4)$$

$$\mathrm{f(v)} = \begin{cases} a_n(v - b_n)^2 - C_n & (v < 0) \\ a_n(v - b_p)^2 - C_p & (v \geq 0) \end{cases} \quad (5)$$

$$\mathrm{g(v)} = \begin{cases} k_n(v - l_n)^2 + m_n & (v < r) \\ k_p(v - l_p)^2 - m_p & (v \geq r) \end{cases} \quad (6)$$

続いてニューロンの内部では(2)~(6)のように I_{sim}^i，他の定数を入力として，v, n, q の内部変数が計算され，最終的に I_s^i を出力する．以上(1)から(6)の式までを一定回数繰り返す．また，結合強度 w_{ij} の学習方法のモデルも様々あるが，まずこのシミュレーションにおいては簡単化のためw_{ij}は定数とする．

2.3 先行研究における高速化の工夫

DSSN モデルの全体の計算の中で最も計算量が多いのは(1)式の加重和計算であり，この部分の計算を高速化する必要がある．(1)式で行われている計算のうちあるニューロン i についての加重和 I_{stim}^i は図 3 のようなデータフローグラフで表すことができる．この計算を逐次実行した場合，$I_{stim}^1 \sim I_{stim}^N$ の計算の総量は N^2 のオーダーとなる．これに対し，高速化の手法としてパイプライン処理を行った．パイプライン処理では，図 3 のように N 個の加算器と N 個の乗算器からなる回路を用意する．

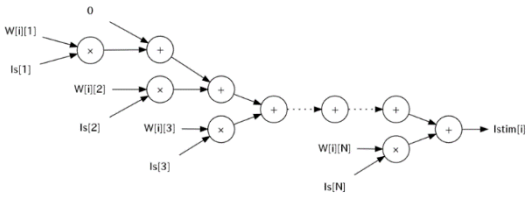

図 3　荷重和部分の回路

Vol.2017-SLDM-181 No.30
Vol.2017-EMB-46 No.30
2017/11/7

図 3 の加算器，乗算器を縦に見ると，N+1 ステージに区切ることができる．まずニューロン 1 が 1 ステージ目で 1 サイクルかけて計算され，続いてのサイクルではニューロン 1 がステージ 2 で，ニューロン 2 がステージ 1 でというようにニューロンのデータが流れていくことになる．N=256 のときについて考えた場合のサイクルと実行されるニューロンの番号を図 4 に示す．

図 4 各サイクルにおける演算

オレンジ色で示されているのが加重和の計算である．横がサイクル数，縦が計算しているニューロンの番号，四角の中にある番号が先ほど説明したステージ数となっている．ニューロン 1 については 256 サイクルで加重和 I^1_{stim} の計算が終わり，その後この回路からは，1 サイクルごとに $I^1_{stim}, I^2_{\mathrm{stim}}, \ldots I^N_{stim}$ を出力されることになる．つまりこのパイプライン処理のスループットは 1，レイテンシは 256 となる．出力された I^i_{stim} は，図 4 において緑色の部分であらわされるように(2)～(6)式で示されるニューロン内各変数の計算を行う回路へとデータが移っていく．もともと N^2 のオーダーの計算時間がかかった加重和の計算は，N 個の計算資源を用意し，同時に計算を行う N 段のパイプライン化によって $\mathrm{O}(\mathrm{N}^2)/N = O(\mathrm{N})$となり，計算時間が格段に減少した．

また，Is や W などの入出力は 16bit の固定小数点（小数点部分 13bit)，その他の変数には 18bit の符号付き固定小数点（小数点部分 13bit）が用いられている．このように必要なところで bit 幅を変えることができる点も FPGA の大きな強みとなっている．

以上のような高速化の工夫によって，先行研究では 768 個のニューロンを実装することができた．しかし，ボード上のメモリの不足により，それ以上のニューロンを実装することはできなかった．先行研究におけるこの問題に対する解決策は 2 通り考えることができる．1 つ目はオフボードのメモリ(DRAM)を使用すること，2 つ目は複数の FPGA を用いて各ボード上のメモリを使用してシミュレーションを行うことである．前者について，実行環境におけるオフボードメモリのバンド幅は，38GB/s である．DSSN の計算においては，図 3 の通り毎サイクル各 2(Bytes)の結合強度 w_{ij}を合計 N 個呼び出す必要がある．200MHz で動作させる場合，1 秒間に呼び出さなければいけないデータ量は，

$$2(\mathrm{Bytes}) \times \mathrm{N} \times 200 \times 10^6(\mathrm{Hz})$$

と計算することができ，768 個のニューロンを実装した場合は，

$$2 \times 768 \times 200 \times 10^6 = 307.2GB/s \gg (38GB/s)$$

とオフボードメモリのバンド幅を大きく超えてしまう．これでは，加重和計算の部分のパイプライン処理も 1 サイクルずつ行うことができず，大幅な遅延が予想される．一方，複数の FPGA によって実装を目指す後者は FPGA 間の伝達遅延がボトルネックとなるが，もし効率的なデータ転送を可能にする計算分割手法を考えることができれば，増やしていく FPGA ボードの枚数に応じたメモリの容量，つまり多数のニューロンの実装を望むことができる．

2.4 計算分割手法

まずは，FPGA の接続方法について考える．接続方法としては，直線状，リング状，メッシュ状などが考えられる．リング状，メッシュ状に FPGA を配置した場合の方が，端から端の FPGA までのデータの転送距離が短くなるので，データの転送遅延が少なくなり，直線状の実装よりも効果的である．リング状とメッシュ状の違いについて，リング状の構造では，通信の方向を一方向に限定することができ，コネクションのバンド幅を目いっぱい使うことができる点が挙げられる．メッシュ状の構造では，さらにデータの転送距離を減らすことのできる可能性があるが，どのサイクルでどの転送路を使うのか衝突しないよう自由度の高い中で考えていく必要があり，FPGA を増やしていった際に複雑な配置問題を解く必要が生じてしまう．以上のことから本研究ではリング状でスケーラブルな構造を提案することとする．

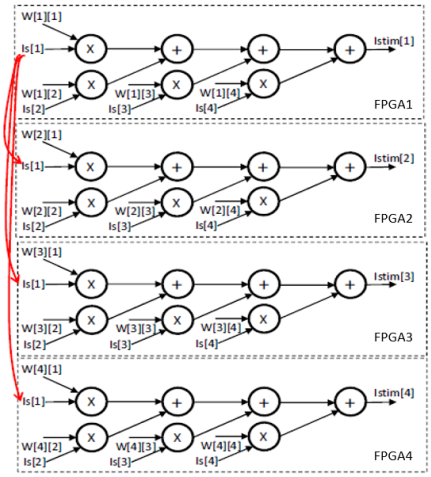

図 5 通常の計算順序による分割

リング状に接続した場合の効率の良い計算の分割を考える．このとき，伝達遅延を考慮し，データ転送が最小となるようなデータ分割が好ましい．さらに，計算の並列性，また各 FPGA の動作が等しくなるような対称性もプログラミングしていく上で重視する必要がある．ここでリングを構成するバスも時計回り，もしくは反時計回りの向きに固定した方が，対称性がよくデータの衝突も抑えられる．

Vol.2017-SLDM-181 No.30
Vol.2017-EMB-46 No.30
2017/11/7

まずはニューロン数 N=4 の時を考える．FPGA の数を M として，ここでは M=4 のときについて考える．(1)の加重和計算は式(7)のように行列の積で表すことができる．

$$\begin{pmatrix} I_{stim}1 \\ I_{stim}2 \\ I_{stim}3 \\ I_{stim}4 \end{pmatrix} = \begin{pmatrix} w11 & w12 & w13 & w14 \\ w21 & w22 & w23 & w24 \\ w31 & w32 & w33 & w34 \\ w41 & w42 & w43 & w44 \end{pmatrix} \cdot \begin{pmatrix} I_s1 \\ I_s2 \\ I_s3 \\ I_s4 \end{pmatrix} \quad (7)$$

まずは単純に横方向に分割することを考えると，その回路図は図 5 の様に表すことが出来る．

図 5 のように分割した場合，各 FPGA でパイプライン処理を行うため，各 FPGA は 1 個の乗算器と 1 個の加算器を持てばよい．一般的には，M 枚のボードを用意した場合，1 枚の FPGA 内には N/M 個の乗算器, 加算器を用意する必要がある．この手法では，各 FPGA が図 5 のようにそれぞれ $\mathrm{Istim}^{\mathrm{i}}$ を出力し，そこから Is^{i} が式(2)～(6)の差分方程式によって計算されるが，その Is^{i} を出力後に図 5 内赤矢印のように他の全ての FPGA へとまとめて送らなければならない．この多くのデータが送り終わって，Is^{i} が更新されなければ次のステップの計算に進むことはできない．ここのデータ転送の部分にサイクル数が多くかかってしまう．1 ステップにかかるサイクル数を見積もると以下のようになる．実験により MaxRing は 1 サイクルに 1 つのデータを十分に転送可能であることが分かっているので，転送に 1 サイクル要すると見積もることとする．

加重和部分：N サイクル

差分方程式部分：12 サイクル

他への Is^{i} の転送：$\mathrm{N} \times (\mathrm{M}-1)/\mathrm{M}$サイクル

合わせて$\mathrm{N}+12+\ \mathrm{N} \times (\mathrm{M}-1)/M$サイクルかかることになる．他への Is^{i} の転送については，N 個のIs^{i}のデータをリング状に接続された他の FPGA へ（M－1）回隣へ送る必要があり，その転送を M 本の転送路で処理することからこのような式となった．

続いて，本研究で提案する分割手法を図 6 に示す． (8)の式を縦方向に 4 つに分割する方法を考える．FPGA1 は結合強度 w_{ij} の 4 行 4 列の行列のうち 1 列目と Is1 の積を担当し，FPGA2 は 2 列目と Is2，FPGA3 は 3 列目と Is3，FPGA4 は 4 列目と Is4 の積の計算をそれぞれ行う．そのためには，加重和を計算した途中までの部分和を隣の FPGA に転送し，続いて，隣から受け取った部分和に積を計算した荷重和を計算した結果をたしていくという操作をすればよい．ここで図 6 において，Is^{i} について注目してみると分かるように，このように分割を行えば部分和が FPGA のリングを一周したときに，Is1~Is4 が更新され，その値を各 FPGA1~FPGA4 がそれぞれ用いるため，毎ステップごとに他の FPGA に渡す必要がなく，通常の計算順で分割するときと比べてデータの転送量が少なくなる上，転送路が常に埋まっている状態になる．例えばニューロン 1 の加重和計算は，まず FPGA4 から加重和計算が始まり，隣への転送を繰り返して FPGA3→FPGA2→FPGA1 と戻ってくると(9)式の計算は完了したことになる．図 7 のように実行順を工夫すると，隙間なく演算を各サイクルに詰め込むことができる．

$$\mathrm{I}^1_{\mathrm{stim}} = w11Is1 + w12Is2 + w13Is3 + w14Is4 \quad (9)$$

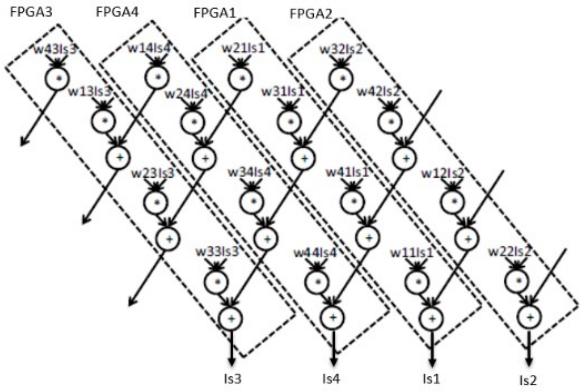

図 6 提案する計算順による分割

図 6 を見てみると，ロジック，メモリリソースの使用量に関しては，図 6 で示す通常の計算順序での実装と変わらない．続いて先ほどの手法と同様にサイクル数を見積もると，部分和としては(M-1)回転送されることで，全ての演算を終えることになるため，

加重和部分：N サイクル

差分方程式部分：12 サイクル

隣への FPGA への転送：（M - 1）サイクル

と見積もることが出来る．合わせて N + 12 + (M - 1)サイクルかかることとなる．表 1 には先行研究と通常順序の分割，提案する分割法について，実行サイクルと各ボードのロジック，メモリリソースの使用量をまとめた．

表 1 各手法のサイクル数とロジック/メモリ使用量

手法	サイクル	ロジックリソース	メモリ
先行研究	N+12	N	N
通常分割	$N+12+N\times\frac{M-1}{M}$	N/M	N^2/M
提案手法	N+12+(M-1)	N/M	N^2/M

ニューロン数 N は FPGA の数 M に対して十分大きいので，通常分割のサイクル数は 2N サイクル，提案手法のサイクル数は N サイクル数とみなすことができ，サイクル数がおよそ 2 倍となった．

提案手法は，FPGA の数を増やしていくとその数に対しあくまで線形にサイクル数が増えていく．このことにより FPGA の数は消費電力にこだわらなければいくらでも増やすことができ，スケーラビリティがある．ロジックリソースの使用量は，ニューロンの数 N に対して線形に増大して

いくが，一方，メモリについては，ニューロンの数 N に対して 2 乗のオーダーで増大していくため，依然として実装できるニューロンの数はメモリと FPGA ボードの枚数によることがわかる．ここで例えば，ニューロンの数を N，結合強度 w_{ij} を 2(Bytes)，ボード 1 枚あたりのメモリ容量を C(Bytes)とおくと，ボードが 1 枚のとき，

$$2 \times N^2 < C(\text{Bytes})$$

$$N < \sqrt{C/2}(\text{Bytes})$$

を満たす．M 枚のボードを使用した場合，容量の関係式は以下の様に表される．

$$2 \times N^2 < MC(\text{Bytes})$$

$$N < \sqrt{MC/2}(\text{Bytes})$$

この式より，ボードを M 枚用意したとき，現在 1 枚の FPGA で実装できているニューロンの数 768 個の $\sqrt{M}$ 倍実装可能であると見積もることができる．4 枚のボードでは$\sqrt{4}$ 倍となる 1536 個のニューロンが実装できると予想される．

FPGA 数が増大した際の問題として，大量の FPGA 間のクロック同期をどのように取るかという問題が考えられる．また，提案手法では毎サイクル MaxRing という FPGA 間の通信を使っているが，通常分割では一通り計算した後にまとめてデータを転送している．サイクル数では，提案手法でサイクル数が減少したが，配置配線の後の動作周波数があまり上がらない可能性もある．それも含めて実験によって明らかにしていく．

3. 実験

3.1 実験環境

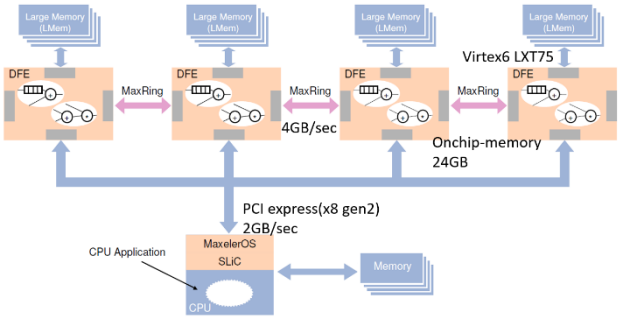

図 7　Maxcompiler の構成

実験には Maxcompiler という高位合成ツールを用いる．Maxcompiler とは，Maxeler 社が提供している高位合成ツールである．ユーザーは Java 文法でデータフローグラフと CPU と FPGA 間のインターフェースの動作を記述し，C 言語でホスト CPU の動作を記述する．Maxcompiler がデータフローグラフを RTL 記述に変換した後，Xilinx のツール群が FPGA の設計データを生成する．高度なパイプライン化を含むデータフローグラフの最適化，演算器の割り当てなどは論理合成ツールによって行われる．配置配線が最も時間を要するプロセスであり，1 時間を超えることも多い．

最新の Maxcompiler のシリーズでは，8 枚の FPGA ボードが MaxRing という伝送路でつながれリング状の構造を成している．本研究室にある Maxcompiler の構成は，図 7 のように CPU が Xeon X5650，FPGA は Virtex6 LXT75 の 4 つのボードによって構成されている．そのうち 3 つの接続は，Maxeler のサポートする Maxring(4GB/sec)によって接続されていて，各ボードは CPU と PCIexpress(2GB/sec)で接続されている．

3.2 実験内容

実験では，まず 1 つの FPGA ボードに提案するリング状の構造を再現することを試みる．高位合成ツールでは，全ての構造を指定することはできないものの，1 つの FPGA 内でもある程度 4 つのブロックに分かれたような実装ができるのではないかと予想し，全ての計算，メモリを 4 分割して行うように記述した．また，リング状となるように計算順序を変更した．従来実装でニューロンの個数が 256 個,768 個のもの，提案する実装について 256,768 個のものについて，それぞれリソースの使用量を比較した．また，先ほど図 5 で示した横方向の行列分割について 256 個のニューロンを実装した．実験の主目的は 2 つあり，1 つ目は 4 つの FPGA に拡張したときのために，リソース使用量などに大きな問題がなく 1 ボードへの実装が可能かどうか確認することである．2 つ目は，この分割によって，FPGA ボード内における Is のデータ転送距離が減少することが期待され，そのことにより，回路が簡単化され，さらには動作周波数が上がるではないかという仮説を検証することにある．

3.3 実験結果

実験結果が表 2，表 3 に示した通りである．(ⅰ)～(ⅴ)すべての条件で，配置配線を行うことができ，問題なく動作した．(ⅰ)従来実装(N=256)，(ⅱ) 4 分割で，提案する計算順序での実装(N=256)，(ⅲ)4 分割で，横方向分割での実装(N=256)，(ⅳ)従来実装(N=768)，(ⅴ) 4 分割で，提案する計算順序での実装(N=768)となっている．

表 2　256 個のニューロンについて実行順による比較

項目	(ⅰ) 従来	(ⅱ)通常	(ⅲ)提案
周波数 (MHz)	250	200	200
ニューロン数	256	256 (64*4)	256 (64*4)
サイクル数	256+12	256+12	256+12
ロジック リソース (最大 297600)	47044 （15.81%）	46032 (15.5%)	56846 (19.1%)
LUT (最大 297600)	20617 (6.93%)	36138 (12.1%)	36980 (12.4%)

FF (最大 297600)	42271 (14.2%)	43059 (14.5%)	52689 (17.7%)
乗算器 (最大 2016)	257 (12.8%)	260 (12.9%)	1028 (51.0%)
DSP ブロック (最大 2016)	257 (12.8%)	260 (12.9%)	1028 (51.0%)
BRAM (最大 2128)	291 (13.7%)	678 (31.9%)	679 (31.9%)

表 3　768 個のニューロンについて実行順による比較

項目	(ⅳ) 従来	(ⅴ) 提案
周波数(MHz)	200	200
ニューロン数	768	768(192*4)
サイクル数	768+12	768+12
ロジックリソース (最大 297600)	91441(30.7%)	133056(44.7%)
LUT (最大 297600)	48563(16.3%)	87813(29.51%)
FF (最大 297600)	86840(29.2%)	126391(42.5%)
乗算器 (最大 2016)	769(38.1%)	772(38.3%)
DSP ブロック (最大 2016)	769(38.1%)	772(38.3%)
BRAM (最大 2128)	805(37.83%)	1873(88.0%)

3.4 考察

実験の結果について考察を行う．まず主目的の 2 つについて考察を加える．1 点目のメモリの容量の問題について、(ⅱ)の実装においては，BRAM（ブロック RAM）や LUT 使用量の増加はあるものの 4 分割する上では問題ない増加であった．つまり現在関数やメモリなどは分割された状態にあるため，このまま，(ⅱ)を複数 FPGA で実装すれば，256 個のニューロンの実装に対し，各 FPGA の BRAM の使用量は 32%/4=8%に抑えることが出来るはずである．この数字は従来実装(ⅰ)の 14%よりも BRAM の使用量を下げることが出来ている．また，(ⅴ)についても同様に各 FPGA の BRAM の使用量は 88%/4=22%となるはずで，従来実装(ⅳ)よりも BRAM の使用量を下げることが出来ている．また，(ⅰ)と(ⅳ)，(ⅱ)と(ⅴ)の BRAM の使用料を見ると，結合強度w_{ij}のメモリに占める割合が支配的で，ニューロン数を 3 倍にしたときには BRAM の使用量は3^2=9 倍になると予想されるが，おおむね線形の 3 倍程度で済んでいる．

続いて 2 つ目の目的について，提案する(ⅱ)の実装と(ⅰ)の実装を比べると，関数やメモリを分割したせいか，LUT や BRAM の使用量は(ⅱ)が多くなってしまった．動作周波数を見ても，従来実装(ⅰ)では 250MHz で動作したが，提案手法(ⅱ)では 200MHz までしか動作周波数をあげることができなかった．2 つ目については，予想と異なる結果となったが，それは，関数や変数を 4 倍に増やしたため，回路が複雑化したからだと考えられる．

その他，DSP と乗算器の数は(ⅰ)では 257 個となっていて，これはもともと加重和計算に N と同個数の乗算器を用意していたことに由来する．(ⅱ)でも同様に加重和計算に対しては N と同個数の乗算器を用意していることが結果から読み取れる．(ⅲ)の通常分割においては乗算器，加算器の組を 4 回ループの様に回して計算しなければならないところを，乗算器加算器を FPGA の数である 4 倍用意してしまったためロジックリソースが 4 倍となった．

4. おわりに

本論文では、提案する分割手法によってデータの転送量、サイクル数ともに減らすことができると、数式による見積もりで示した。実験では，複数 FPGA に実装する前段階として，提案する分割を 1 つの FPGA に実装を行った．その結果，1 ボードの中で分割されたブロックあたりの BRAM 使用量は減少した．複数 FPGA に実装する際に，大きく回路図の変更はないため，同様に 1 ボードあたりの BRAM 使用量が減少することが予想される．現在実際に MaxRing を用いて複数の FPGA に対して実装を行っていて，データ転送自体の部分でサイクルに大きな後れが生じてしまったり，動作周波数が 200MHz より落ちてしまったりすることはない．今後，複数の FPGA でシミュレーション全体が動かせるよう目指し，さらに FPGA の数も増やしていきたい．

参考文献

[1] Kit Cheung, et al., “Neuroflow: A general purpose spiking neural network simulation platform using customizable processors” frontiers in Neuroscience, 9. 516(2016)
[2] K Ovtccharov et al., “Accelerating Deep Convolutional Neural Ntworks Using Specialized Hardware.”, Microsoft Research(2015)
[3] Jing Li, Yuichi Katori and Takashi Kohno “An FPGA-based silicon neuronal network with selectable excitability silicon neurons” Frontiers in NEUROSCIENCE, 2012, Volume 6,Article 183.
[4] Taro Kawao, et al., “Spiking neural network simulation on FPGAs with automatic and intensive pipelining”
[5] 川尾 太郎, “FPGA を利用した科学技術計算の高速化”, 東京大学工学系研究科修士論文, 2016.
[6] Xilinx,Inc. ,Virtex7 datasheet https://japan.xilinx.com/support/documentation/data_sheets/j_ds180_7Series_Overview.pdf