社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Sw・Hw混載システムを用いたパルスニューラルネットワークのハードウェア化

茂野聡登志† 金丸 隆志† 入谷 勝† 関根 優年†

† 東京農工大学 工学部 〒184-8588 東京都小金井市中町 2-24-16
E-mail: kanamaru@sekine-lab.ei.tuat.ac.jp

あらまし 「アプリケーション開発の容易さ」を主眼にディジタルニューロ回路の試作を行う. ターゲットは FPGA とし, 我々が開発している HwModule ボードと市販の PCI ボードの 2 種類を考慮する. 設計する回路は, 2 入力ニューロンとそれを利用した XOR 回路, そして 9 ニューロン 81 シナプスの連想記憶回路である. HwModule はハードウェアを通常のオブジェクト指向言語のオブジェクトとしてアプリケーションから利用可能であり (HwObject), 設計したニューロ回路をソフトウェアから容易に利用可能である. (信学技報, vol.103, No.732, pp.135–140 (2004)).

キーワード パルスニューラルネットワーク, Sw・Hw 混載システム, FPGA, ハードウェアオブジェクト, HwModule

Hardware implementation of pulse neural networks with Sw/Hw heterogeneous system

Akitoshi SHIGENO†, Takashi KANAMARU†, Masaru IRITANI†, and Masatoshi SEKINE†

† Department of Electrical and Electronic Engineering, Faculty of Technology, Tokyo University of Agriculture and Technology, Tokyo 184-8588, Japan
E-mail: kanamaru@sekine-lab.ei.tuat.ac.jp

Abstract Digial neural-network circuits are designed to facilitate the development of applications which use neural circuits from the software. As targets, we use FPGAs and consider both our custom-made HwModule and a PCI board on market. A neuron with two inputs, an XOR circuit, and an associative-memory circuit with 9 neurons and 81 synapses are designed. Using the HwModule, application developers can use hardwares as usual objects in their applications, thus they can easily use neural circuits from softwares.

Key words pulse neural networks, Sw/Hw heterogeneous system, FPGA, hardware object, HwModule

1. はじめに

ニューラルネットワークは本質的に並列分散処理システムであるため, 従来のノイマン型コンピュータで実装すると膨大な計算時間がかかり, ニューラルネットワークが持つ並列性を十分に活かすことは難しい. そのため, ニューラルネットワークのハードウェア化が望まれている. ハードウェア化方式にはアナログ回路方式とディジタル回路方式があるが, 近年, FPGA (field-programmable gate array) のような書き換え可能なデバイスの大規模化, 高速化が進んで来たことにより, ディジタル回路方式のニューロハードウェア [1]〜[10] が注目を集めている. さらに, ディジタル回路方式には「他のシステムとのインターフェイスが容易である」, 「ノイズに強い」などの利点も多い.

そのようなニューロハードウェアが現実に広く応用されるようになるためには, 「大規模なネットワークを作成できること」, 「そのハードウェアをアプリケーションから容易に利用できること」が必要であるが, 現状ではそれらの条件が満たされているとは言いがたい.

FPGA をターゲットにした場合, 大規模ネットワークを作成するには

- 小規模な回路で済むような回路方式, 情報表現を工夫する
- 複数の FPGA を実装し, それらを連係させる
- 今後のさらなる FPGA の大規模化に期待する

などが考えられる.

一方, ハードウェアをアプリケーションから容易に利用できるようにするためには, ハードウェアとソフトウェアのインターフェイス部分に工夫が必要となる. 我々はオブジェクト指向のパラダイムに基づき, ハードウェアをアプリケーションからオブジェクト (HwObject) として利用可能となるような枠組み (Object Manager) を提供しており, そのためのプラット

フォームとして FPGA を 3 つ実装した HwModule を開発している [11]. 図 1 および 2 に HwModule の模式図と写真をそれぞれ示した． HwModule は通常の PC の PCI バスに接続

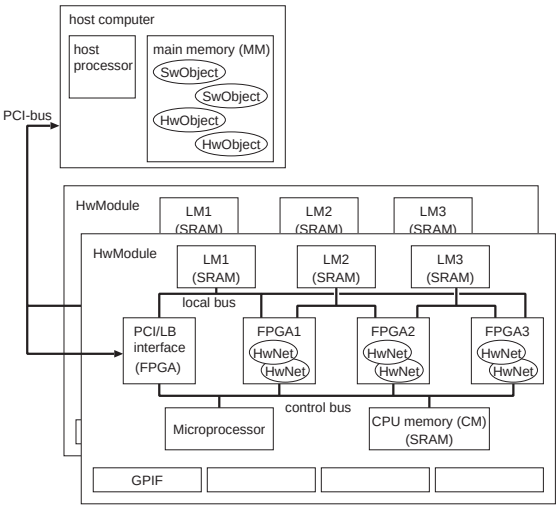


図 1　HwModule の模式図.

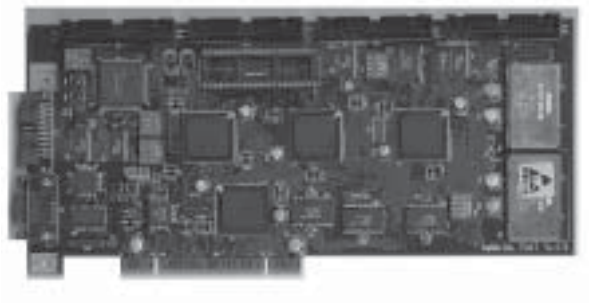


図 2　HwModule の写真. ユーザ用の FPGA が 3 つ搭載されている. FPGA は Xilinx 社製の Spartan2 (xc2s200fg456) である.

されるデバイスであり, HwModule と HwObject の枠組みを用いるとアプリケーション開発者は, 既に HDL 言語などで設計されたニューロ回路を通常の C++ 言語から利用可能になる. 例えば, new 演算子で HwObject を確保すると, 回路データが FPGA にダウンロードされる, などである. また, HwModule は一枚につき 3 つの FPGA が載り, さらに複数の HwModule を 1 つの PC に接続することも可能であるので, 今後の回路の大規模化にも対応可能であると考えている.

本研究は主に「アプリケーション開発の容易さ」を主眼にディジタルニューロ回路の試作を行う. ターゲットとしては我々が開発している HwModule ボードと, 市販の PCI ボード [12] の 2 種類を考慮して比較検討を行う.

2. ニューロンの設計例

ディジタル回路を用いたニューロンモデルの模式図を図 3 に示した. 0/1 のパルス列を入力として受取り, 0/1 を出力するため, これはパルスのやりとりで情報処理を実現しているという意味ではパルスニューロンのモデルである. ただし, 情報のキャリアとして発火頻度を用いるかパルスのタイミングを用い

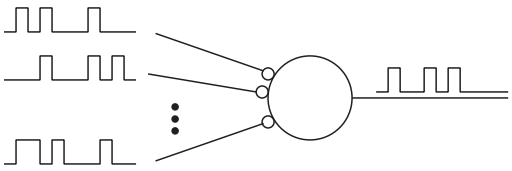


図 3　ディジタル回路を用いたニューロンモデルの模式図.

るかは設計する回路次第となる. パルスのタイミングを重視するいわゆる「パルスニューロン」も設計可能であるが, 以下では発火頻度を情報とするニューロンモデルを取り扱う. これは, 発火頻度を情報としてニューロンの振舞いのほうが現状では広く研究されており, アプリケーション開発が容易であろうと考えられるからである.

以下では, FPGA を対象としたニューロンの設計例を示す. 任意の入力数に対応した素子を設計できるが, まず簡単のため 2 入力素子についての解説を行う.

2.1　全体のブロック図

設計したニューロンモデルの模式図を図 4 に示す. 処理の大

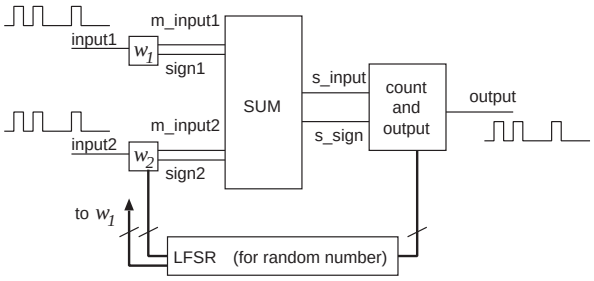


図 4　設計したニューロンモデルの模式図.

まかな流れは以下のようになる.

- 結合係数 w による入力パルスの変調.
- 複数の入力を空間加算.
- 空間加算したパルスの数をある時間幅 (timeslot) で計数し, その数に応じてパルスを出力する.

なお,「結合係数 w による変調」と「パルスの出力」には乱数が必要であるので, 乱数生成用に LFSR (linear feedback shift register) が各素子に一つ存在する.

2.2　入力表現

図 5 のように, 入力パルスは一パルスあたりクロック周期の幅を持っている. また, パルスの位置は乱数で決めることと

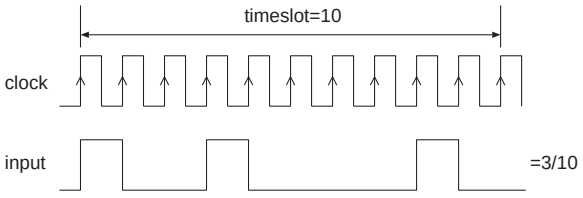


図 5　入力表現.

する. その模式図を図 5 に示した. “timeslot” の時間幅におけるパルス数の割合が情報を担っているとする. 図 5 の場合, timeslot=10 の中に 3 発のパルスが存在するので, 入力は 3/10

であると考える. この情報表現により, [0,1] 区間の実数が表現できる. 実際に設計した回路では timeslot=255, 511, 1023 などを試している.

このように, ランダムなパルスによって情報が表現されているとする方式に文献 [1], [3] がある. 一方, パルスが周期的であるとする (周波数変調) 方式に文献 [2], [4], [6] があるが, ここでは採用しなかった.

2.3 結合係数によるパルス変調

入力パルスに対して, 結合係数 w を乗算するという操作を考える. 結合係数 w は正負の値を取り得るが, ここでは $-1 \leqq w \leqq 1$ と考え, その絶対値 $|w|$ が確率を表すと考えよう.

そして, 入力パルスが到達するたびに, 確率 $|w|$ で入力パルスを後段に 1 クロック遅れで伝えることとする. このとき, 入力のパルス数が N であれば, 変調されたパルス数の期待値は $N|w|$ 個となる. このような確率によるシナプス乗算は文献 [1] で用いられている.

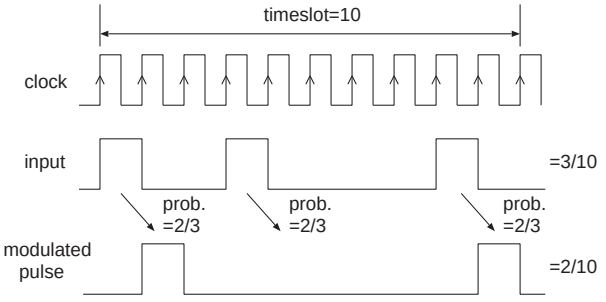


図 6 結合係数 w による入力パルスの変調.

今, $|w| = 2/3$ である場合の模式図を図 6 に示した. $N = 3$ であったパルス数が変調によって $N = 2$ になっていることがわかる.

なお, 後段に w の符号を伝える必要があるため, w の符号用の 1 bit 出力が存在する (図 4).

実際に設計した回路では w を 8 bit の 2 の補数表現で表し (すなわち $-128 \leqq w \leqq 127$), $|w|/128$ が確率を表すとしている.

2.4 空 間 加 算

結合係数 w により変調された複数のパルスを加算 (空間加算) するための回路は組み合わせ回路で実現できる.

いま, 図 4 のように符号 sign1 のパルス m_input1 と符号 sign2 のパルス m_input2 の 2 入力が加わっているとする. ただし, 符号 sign は正のとき 0, 負のとき 1 をとるとする.

いま, ある時刻における正のパルス数 pos_pulse と負のパルス数 neg_pulse はそれぞれ

$$\text{pos\_pulse} = \overline{\text{sign0}} \cdot \text{m\_input0} + \overline{\text{sign1}} \cdot \text{m\_input1} \qquad (1)$$

$$\text{neg\_pulse} = \text{sign0} \cdot \text{m\_input0} + \text{sign1} \cdot \text{m\_input1} \qquad (2)$$

とあらわされる. このとき,

- pos_pulse > neg_pulse なら s_pulse = 1, s_sign = 0
- pos_pulse < neg_pulse なら s_pulse = 1, s_sign = 1
- pos_pulse = neg_pulse なら s_pulse = 0, s_sign = 0

と出力を定める. この模式図を図 7 に示した.

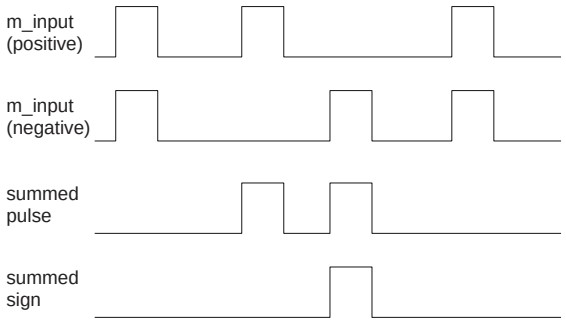


図 7 入力の空間加算.

2.5 パルスのカウントとパルスの出力

空間加算されたパルスと符号を用いて timeslot の時間間隔内のパルス数をカウントし, その値に基づいてパルスを出力する回路が必要である. 図 8 に従って解説する. 入力パルス s_input

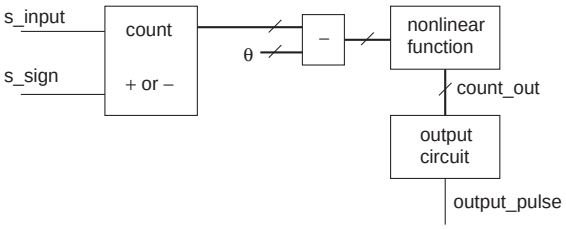


図 8 パルス数のカウントと出力.

と入力符号 s_sign に従って, レジスタ count の値を +1 または −1 する. この計数は timeslot の時間間隔だけ行う. レジスタ count は timeslot のビット幅に符号ビットと桁あふれを考慮しただけのビット幅が必要である. timeslot=255 を例とすると, count のビット幅は 8 + 1 (符号) + 3 (桁あふれ用) の 12 ビットとなる.

計数が終了したらあらかじめ設定してあった閾値 θ を count から引き, その値に非線形関数を施し, その出力を count_out とする.

非線形関数の出力 count_out は, 「count を 4 倍し, 最大値 (=timeslot) を超えていたら timeslot の値に, 0 以下であったら 0 にセットする」こととする. これにより, count_out は 0〜timeslot の値を取るが, これを timeslot で割るものと考えると, 出力は [0,1] 区間の実数値を取ることになる. この実数値を再び確率と考え, 出力パルスを生成する. この出力パルスは次段のニューロンへの入力となる.

2.6 制 御 回 路

ニューロンモデルの動作のタイムチャートは図 9 のようになる. パルスの計数とパルスの出力は並列に動作する.

このようなタイムチャートを実現させるために, 図の count_ctrl, update_ctrl, output_ctrl のような制御信号を生成する回路を作成した. この制御回路はネットワークにつき一つ存在する.

2.7 インターフェイス回路

作成したニューロン回路を複数接続する場合, 各ニューロンは図 10 のように 0/1 のパルスで情報をやり取りする. この

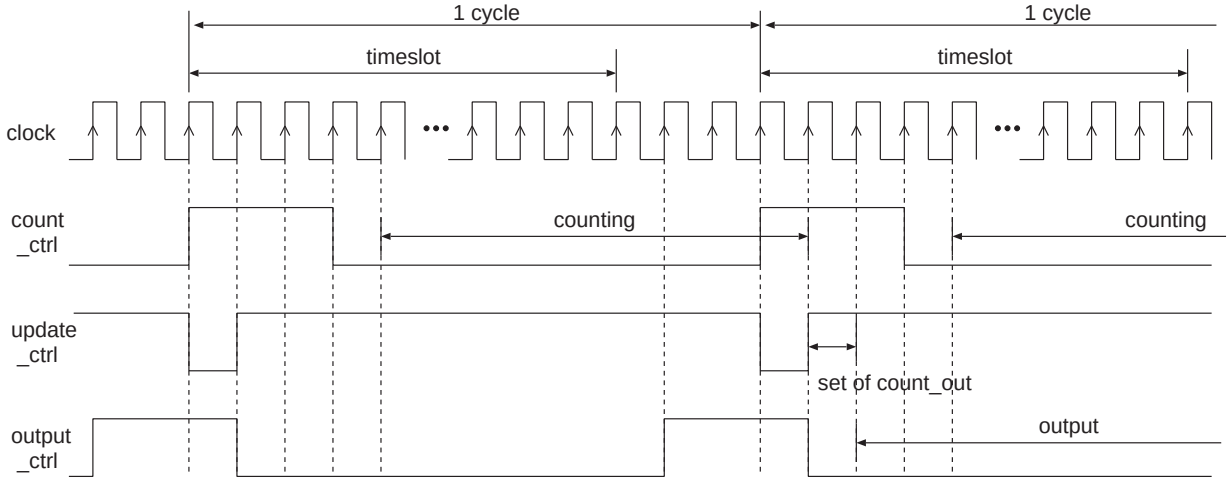


図 9　パルス数のカウントと出力のタイムチャート, およびそれを実現するための制御信号.

とき, 外部とのデータのやりとりのために, インターフェイス回路が必要になる. 具体的には, 入力値をパルス列に変換する MakeInputPulse という回路を作成した. MakeInputPulse モジュールは確率を用いてパルス数が入力値がほぼ等しくなるようにパルスを出力する. ニューロンの出力は, パルス数をカウントするモジュールの count_out (図 8 参照) を観測することにする.

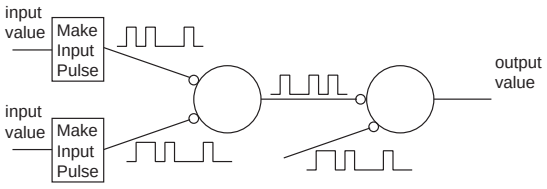


図 10　作成したニューロンと外部とのインターフェイス.

2.8　シミュレーション結果 1: 入出力関係

以上のようにして作成した 2 入力 1 出力のニューロンモデルのシミュレーションを行った. シミュレーションには Solaris 9 上で動作する Synopsys 社製の VCS を用いた.

まず, 2 入力のうち 1 系統のみにパルスを加え, 素子の入出力関係を調べる. 外部との情報のやりとりには 2.7 節のインターフェイス回路を用いる.

シミュレーションは以下のステップで行う.

（1） 結合強度 w, 閾値 θ, 乱数の seed をニューロンにセットする (seed はシミュレーション中一定とする).

（2） ニューロンに入力を加える.

（3） ランダムな回数だけニューロンに出力を計算させる (この回数はソフトウェア上で乱数を用いて決める).

（4） ニューロンの出力を得る.

(3) のステップがないと, ニューロンは毎シミュレーションごとに同じ乱数系列で出力を出すことになり, きれいな出力が得られる傾向がある. (3) のステップを加えることでより実機での動作に近いシミュレーションになると言える.

timeslot を 255, 511, 1023 に変えてシミュレーションを行った結果が図 11 である. 入出力の値は最大値が timeslot になるように表示されている. timeslot が短いと入出力関係に揺らぎが大きいことがわかる. さらに, w が小さいとやはり揺らぎが大きくなる.

なお, $w = 1$ のグラフの傾きは 4 である. これは count を 4 倍して非線形処理を施して count_out にセットしたことによる. より急峻な傾きが欲しい場合は 4 倍ではなく 8 倍, 16 倍などを施せば良いであろう.

2.9　シミュレーション結果 2: XOR

次に, 3 素子を組み合わせて図 12 のような構成の XOR 回路を作成した. この回路の入出力関係をシミュレーションで調べて図示すると図 13 のようになった. 確かに XOR 回路として動作している. なお, 傾きが急峻になるところでの揺らぎが大きいのは確率演算による素子の特徴である.

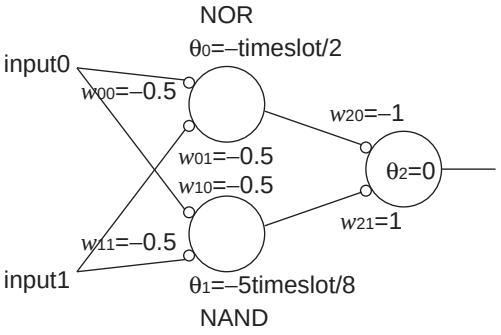


図 12　XOR 回路の構成.

2.10　ゲート数

ここで作成したの「2 入力ニューロン」および「XOR 回路」を Xilinx 社製の WebPack ISE で Implement したところ, 以下のようなゲート数になった. ただし, ターゲットは Xilinx 社製の Spartan2 (xc2s200fg456) とした. 255, 511, 1023 の 3 種の timeslot について調べている. まず, 「2 入力ニューロン」は図 4 のような構成で Implement したもので, 制御回路などは含めていないゲート数を表す. 2.7 節で扱ったインターフェイス回路のゲート数は MakeInputPulse に対応する. また, 「XOR(full)」は 3 ニューロンに制御回路, MakeInputPulse 2

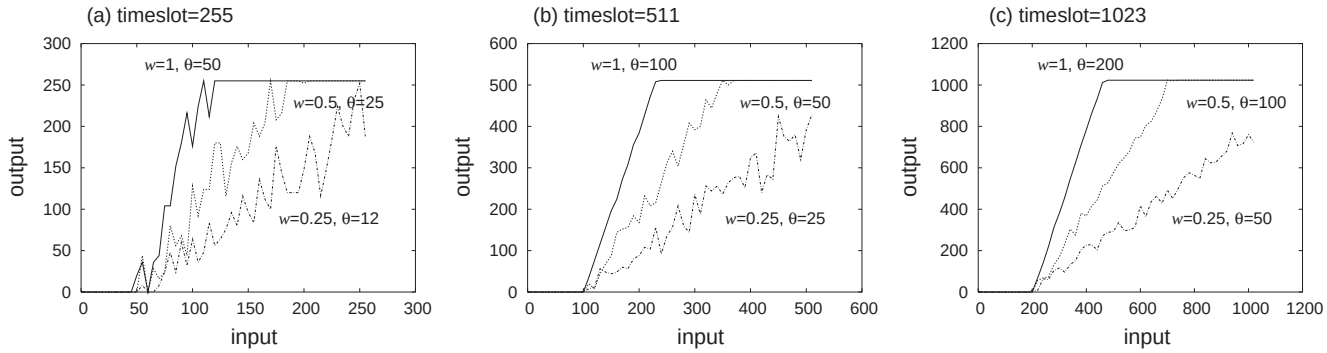


図 11　設計した素子の入出力関係. 一つのパラメータあたり, およそ 50 のデータをプロットしている.

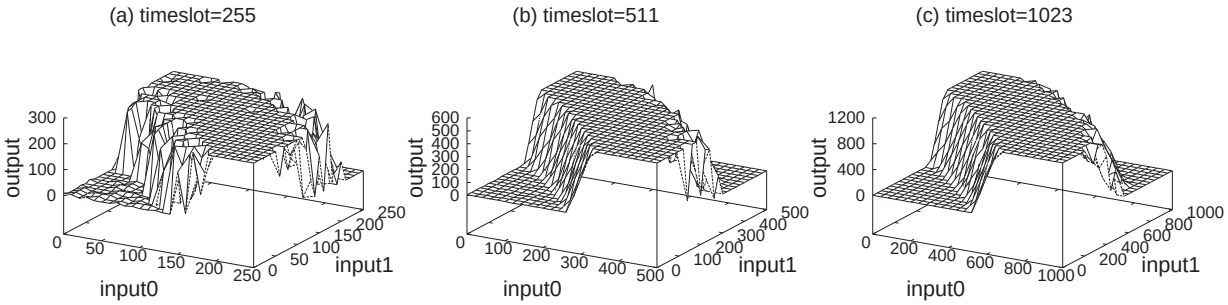


図 13　XOR 回路の入出力関係. 3 つの timeslot に関して調べた.

回路 \timeslot	255	511	1023
2 入力ニューロン (a)	1,372	1,427	1,482
制御回路 (b)	348	374	394
MakeInputPulse (c)	162	179	196
XOR (full)	4,876	5,116	5,332

つを加えて Implement したもので, ほぼ $3a + b + 2c$ に近いゲート数になっていることがわかる.

ターゲットの Spartan2 は 200,000 ゲート であるので, 数十〜百程度のニューロンが実装可能であることがわかる. ただし, 上記のデータは 2 入力ニューロンについて行ったもので, 入力数を増やすと実装可能な素子数も減ることになるだろう.

3.　9 ニューロン 81 シナプス連想記憶回路

前章で取り扱った 2 入力ニューロンを同様の方式で「9 ニューロン 81 シナプス連想記憶回路」を設計した. この素子数にしたのは, 比較的設計が容易であろうと考えたからである. timeslot は 1023 を用いる. 全ての素子から入力が加わるので, 1 ニューロンに対し 9 つの入力が存在する.

このネットワークに対してパターン ξ_i^μ $(i = 1, 2, \cdots, 9, \mu = 1, 2, \cdots, p, \xi_i^\mu \in \{0, 1\})$ を埋め込む. パターン数 p は 3 とし, その 3 つのパターンは図 14 のように定める.

このとき, j 番目の素子から i 番目の素子への結合 w_{ij} を以下で定める.

$$w_{ij} = \frac{1}{Na(1-a)} \sum_{\mu=1}^{p} \xi_i^\mu (\xi_j^\mu - a). \tag{3}$$

ただし, $N = 9$, $a = 0.5$ とした.

図 14　ネットワークに埋め込む 3 つのパターン. 黒が 1, 白が 0 を表す.

以上のようなネットワークを Verilog-HDL で記述し, 前章と同様に Synopsys 社の VCS でシミュレーションした. まず, パターン想起の様子を図 15 に示す. 時間ステップ 0 におけるパターンは, ネットワークの初期状態である. 2 ステップで一つ目のパターンが想起されていることがわかる.

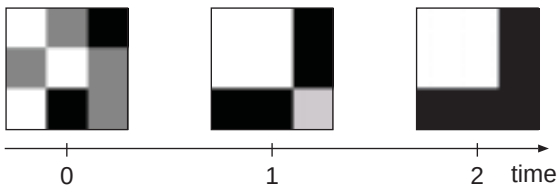


図 15　設計した回路によるパターンの想起. 2 ステップで一つ目のパターンが想起されていることがわかる.

このときのパターンとネットワークの状態のオーバーラップ

$$m^\mu = \frac{1}{Na(1-a)} \sum_{i=1}^{N} (\xi_i^\mu - a)(x_i - a). \tag{4}$$

を図示したのが図 16 である. やはり 2 ステップで想起に成功していることがわかる.

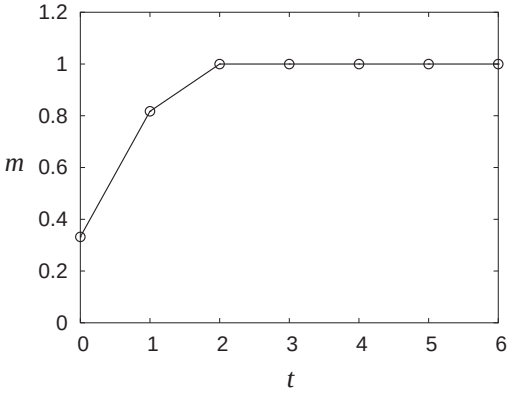
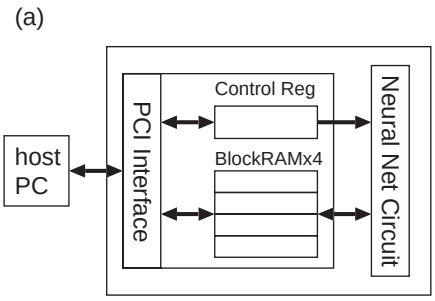


図 16　図 15 に対応するオーバーラップ m の時間変化. やはり 2 ステップで想起に成功していることがわかる.

4. 実機への実装

まず, 設計した回路の市販の評価用 PCI ボード [12] への実装に関して述べる. このボードには Xilinx 社製の Spartan2 (xc2s150-5fg456) が 1 つ搭載されている. このボードへ実装するために作成した回路のブロック図を図 17 (a) に示す. な



(b) Arrangement of Control Register

31 24	23 16	15 8	7 0	
w_{01}	w_{00}	set	enable	00h
w_{21}	w_{20}	w_{11}	w_{10}	04h
seed0		seed_in		08h
seed2		seed1		0ch
θ_1		θ_0		10h
		θ_2		14h

図 17　評価用 PCI ボードへ実装するために作成した回路.

お, 実装したのは timeslot=255 の 2 入力 1 出力の回路である. ニューラルネットへの変数の受渡しや制御信号 (set, enable) のために Control Register が, 入出力データのために dual port BlockRAM が使用されている. Control Register は図 17 (b) のように配置されており, メモリアクセスと同様にこのレジスタに対して host PC から書き込み・読み込みが可能である. データ用の BlockRAM へのアクセスも PCI メモリアクセスで行う. この回路において, 図 11(a) のような入出力関係を PC 上で得ることができた.

また, 我々が製作した HwModule への実装例については講演にて触れる.

5. ま と め

「アプリケーション開発の容易さ」を主眼にディジタルニューロ回路の試作を行った. ターゲットとしては我々が開発している HwModule ボードと, 市販の PCI ボード [12] の 2 種類を考慮した. 設計した回路は, 2 入力ニューロンとそれを利用した XOR 回路, そして 9 ニューロン 81 シナプスの連想記憶回路である.

設計には, ディジタル回路によるニューロン回路にしばしば用いられる確率演算方式を用いた. この場合, Xilinx 社の Spartan2 を用いると 2 入力ニューロンが数十〜百程度の素子が載ることがわかった. これでは現実の問題に対するニューラルネットワークのアプリケーションを作成するには十分とは言いがたく, 効率の良い回路方式や情報表現を追求する必要があるだろう.

また, 設計したニューロ回路を用いたアプリケーション作成を容易にするため, 我々が研究している HwObject モデルと HwModule ボードを利用し, その有効性を講演にて示す.

本研究は東京大学大規模集積システム設計研究センターを通しシノプシス株式会社の協力で行われたものである.

文　献

[1] 江口裕俊, 古田俊之, 堀口浩幸, 樗木杉高, “学習機能をもつパルス密度形ニューロンモデルとそのハードウェア,” 電子情報通信学会論文誌 C-II, vol.J74-C-II, no.5, pp.369–376 (1991).
[2] 平井有三, “PDM ディジタルニューラルネットワークシステム,” 電子情報通信学会論文誌 C-II, vol.J74-C-II, no.5, pp.267–280 (1991).
[3] Young-Chul Kim and Michael A. Shanblatt, “Random Noise Effects in Pulse-Mode Digital Multilayer Neural Networks,” IEEE Transactions on Neural Networks, vol.6, no.1, pp.220–229 (1995).
[4] 肥川宏臣, “周波数変調パルスと多数決ニューロンによる学習機能付きニューラルネットワーク,” 電子情報通信学会論文誌 A, vol.J82-A, no.7, pp.1005–1015 (1999).
[5] 川島毅, 石黒章夫, 大熊繁, “小規模回路で実現可能なニューラルネットワークのハードウェア化手法,” 電子情報通信学会技術報告, NC99-90, pp.23–28 (2000).
[6] 平井有三, 西澤邦宜, “実時間 PCA 学習回路のハードウェア化,” 電子情報通信学会論文誌 D-II, vol.J84-D-II, no.4, pp.699–707 (2001).
[7] 田中愛之, 黒柳奨, 岩田彰, “FPGA のためのニューラルネットワークのハードウェア化手法,” 電子情報通信学会技術報告, NC2000-179, pp.175–182 (2001).
[8] 市川道教, “脳型コンピュータとニューラル・ネット・プロセッサ,” Computer Today 特集 「脳と情報処理 − 脳はどこまで創れるのか」, no.90, pp.10–17 (1999).
[9] 市川道教, “小型ロボットを用いた脳型コンピュータの開発研究,” Computer Today 特集 「脳を創る − 脳型コンピュータの実現に向けて」, no.106, pp.10–15 (2001).
[10] 桃井昭好, 秋元俊祐, 佐藤茂雄, 中島康治, “ストカスティックロジックを使った連続時間ハードウェアニューロンモデルと非単調活性化関数の改良,” 日本神経回路学会第 13 回全国大会講演論文集, pp.122–123 (2003).
[11] K. Kudo, Y. Myokan, W. Chan Than, S. Akimoto, T. Kanamaru, and M. Sekine, “Hardware object model and its application to the image processing,” IEICE Transactions on Fundamentals, March (2004) in press.
[12] “PCI デバイス設計入門,” CQ 出版社, (2000).