自己組織化学習アルゴリズムのFPGAシステムによる高速化

竹谷史生† 岩崎理典† 吉永努†
荒井正之‡ 大津金光† 馬場敬信†

†宇都宮大学工学部、‡帝京大学理工学部

〒321 栃木県宇都宮市陽東7-1-2
TEL:028-689-6262
E-mail:takeya@lynx.is.utsunomiya-u.ac.jp

あらまし

多段のニューラルネットで構成するアプリケーションに対して専用プロセッサを開発することは，多大なコストを要すると共にアルゴリズムの変更に対する柔軟性に乏しいなどの問題がある．我々は，このようなアプリケーションに対し，書き換え可能なFPGAをベースとしたFCCB(FPGA-based Custom Computing Board)を用いたシステムを提案している．本稿では，FCCBのアーキテクチャとその応用例として，2階層ニューラルネットの自己組織化学習アルゴリズムの適用について述べる．また，その予備評価について考察する．その結果，演算サイクル数はワークステーションUltra2上のCプログラムと比べて約1/531であることなどがわかった．

キーワード RCシステム，FPGA，ニューラルネット，手書き漢字認識

High-Performance Implementation of a Learning Algorithm based on Self-Organization using an FPGA System

Fumio TAKEYA†, Mitinori Iwasaki†, Tsutomu YOSHINAGA†
Masayuki ARAI‡, Kanemitsu OOTSU†and Takanobu BABA†

†Faculty of Engineering, Utsunomiya University,
‡School of Science & Engineering, Teikyo University

7-1-2, Yoto, Utsunomiya-shi, Totigi 321, Japan
TEL:+81-28-689-6262
E-mail:takeya@lynx.is.utsunomiya-u.ac.jp

Abstract

We have proposed a reconfigurable computing system which consists of a host computer and an FCCB, FPGA-based Custom Computing Board. In this paper, we describe the architecture of the FCCB and its application to a learning algorithm based on self-organization. We show its implementation and the preliminary evaluation results. A comparison of execution cycles between the FCCB and a C program on a SUN Ultra-2 workstation shows that the FCCB reduces the cycles approximately one five hundredth of the C program .

key words RC system, FPGA, neural network, hand-written kanji recognition

1 はじめに

ニューラルネットワークの学習計算は，膨大な計算量を必要とする．そのため，ASICを用いたニューロチップ[1][2][3][4]やニューロシステム[5]などが開発されている．しかし，このようなニューロシステムは，ネットワークの形態や規模などの制約から，種々のアプリケーションやアルゴリズムの大幅な変更に対応できない場合も多い．これに対して，我々はFPGA-based Custom Computing Board(以下FCCB)を用いたシステムを提案している．

書き換え可能なFPGAを用いたシステム，リコンフィギュラブル・コンピューティング・システム[6]は，1990年頃から研究・開発が行われている[7][8][9][10][11]．これらの中で最も有名なのは，1989年に開発されたSPLASH[7]である．SPLASHは，汎用計算機が苦手とするパターンマッチング問題などが，専用エンジンにより大きく高速化可能であることを示した．近年のFPGAの技術的な発展は，より小規模のハードウェアによって種々のアプリケーションを実装することを可能としている．本研究は，単一ボードに最新のFPGAを複数搭載したFCCBを活用することにより，負荷の大きいニューラルネット・アプリケーションがどの程度高速化可能であるか示すことを目的とする．これまでに，BPニューラルネットをXILINX社のXC40125XV-1を用いたFCCBに実装し，Sun Ultra2 ワークステーション(以下WS)上のCプログラムと比較した結果を報告している[12]．本稿では，2段のニューラルネットワークで構成する手書き漢字認識アルゴリズムの自己組織化学習アルゴリズムへの適用について述べる．

2 システムの全体構成

図1にシステム全体の構成図を示す．本システムは，ホストコンピュータとその拡張ボードとして実装するFCCBで構成する．FCCBは，ホストとのデータ転送やFCCB全体の動作を制御するCU，ニューラルネットの学習計算などの計算集約的な処理を行う9個のPU，各PUのローカルメモリ(LM)，ホストからFCCBへのデータ転送を仲介する2個のFIFOで構成する．CUとPUは再構成可能なFPGAとし，様々なアルゴリズムを実装可能とする．また，PU-PU間のデータ通信を柔軟にするため，隣接するPU間にリング状のデータバスを設けると共に，全PUにブロードキャストやマルチキャスト可能なデータバスを設ける．

本システムは，ホストとFCCBの両方で処理を行う協調型システムである．そのため，ホスト-FCCB間は高速なI/Oバスで接続する．

3 自己組織化学習アルゴリズム

3.1 Honeycombネット

本研究で適用している手書き漢字認識ニューラルネットHoneycombネット[13]の構成を図2に示す．Honeycombネットは，ベクトル量子型ニューラルネットとBPニューラルネットで構成している．前者は，多数のパターンを類似文字群に大分類する．後者は，ベクトル量子型ニューラルネットによって分類した類似文字群から文字を同定する．

3.2 ベクトル量子型ニューラルネット

Honeycombネットにおけるベクトル量子型ニューラルネットの自己組織化学習アルゴリズムには，コホーネンの自己組織化学習アルゴリズム[14]とK-means法[15]を用いる．コホーネンの自己組織化学習は，図3に示すように，教師信号がなくても入力信号の性質に応じてシナプス荷重を自動的に修正することにより，入力信号のパターン分類ができるアルゴリズムである．K-means法は，クラスタ数を予め決めておき，類似するデータを同じクラスタに分類する手法である．

Honeycombネットのベクトル量子型ニューラルネットの入力には，電総研の手書き文字データベースETL9Bの漢字2965文字を使用する．はじめに，各文字の学習サンプルデータ100セットから平均特徴ベクトルを求める．次に，コホーネンの自己組織化学習により，各文字の平均特徴ベクトルから2965文字を約500個の類似文字群に分割する．そして，分類率向上のため，K-means法により，各ニューロンに1個以上の参照ベクトルを作成し配置する．Honeycombネットでは，クラスタリングした学習サンプルデータの平均特徴ベクトルを参照ベクトルとする．

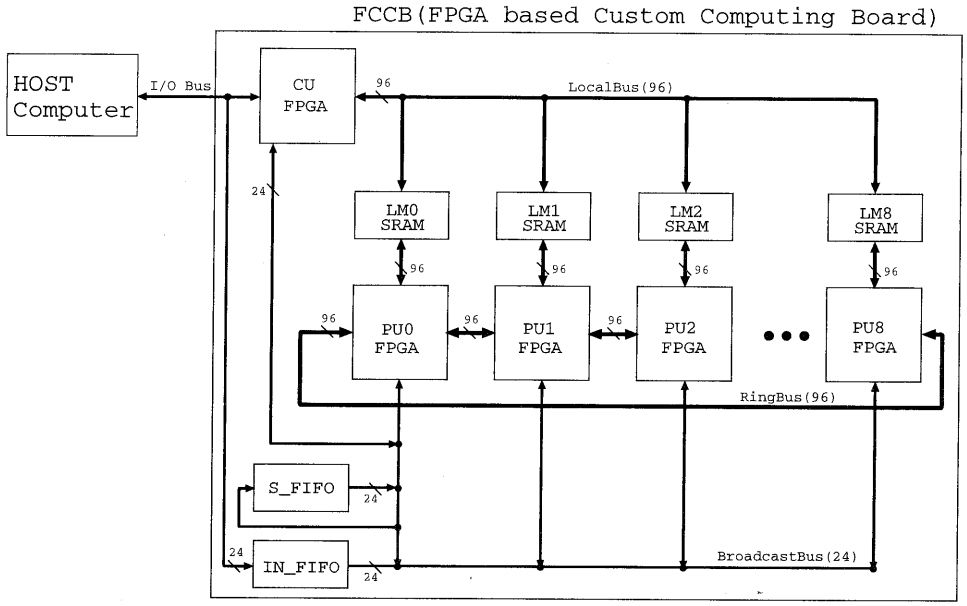

図 1: システムの全体構成

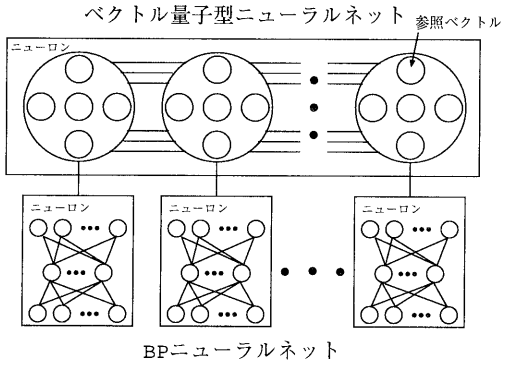

図 2: Honeycomb ネット

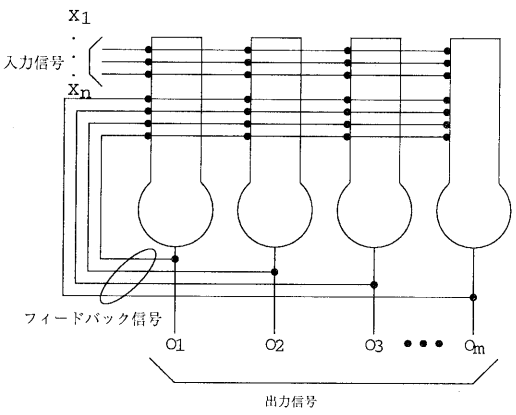

図 3: コホーネンのニューラルネットの一次元モデル

3.3 K-means 法による参照ベクトルの作成

次に，K-means 法による参照ベクトルの作成アルゴリズムを示す．

1. ある文字の学習サンプルデータ 100 セットを設定したクラスタと同数に均等に分配し，初期参照ベクトル $t_0(0)$，$t_1(0)$，…，$t_{p-1}(0)$ を作成する．ここで p はクラスタ数である．

2. k 回目の繰り返しにおいて，各学習サンプルデータを最も距離の近い参照ベクトルをもつクラスタに割り当てる．学習サンプルデータ x と参照ベクトルのユークリッド距離 d が，

$$d(x, t_j(k)) < d(x, t_i(k)) \qquad (1)$$

$$i = 0, 1, \ldots, p-1, i \neq j$$

のとき，x をクラスタ j に移動する．学習サンプルデータ x の抜けたクラスタ，x が加わったクラスタに対し，クラスタに属する学習サンプルベクトルから平均特徴ベクトルを再計算し，新しい参照ベクトルとする．

3. 全ての参照ベクトル i において，

$$t_i(k+1) = t_i(k) \tag{2}$$

であれば，終了．そうでなければ，2へ．

全2965文字に対して上記の処理を行う．論文[13]において，クラスタが5つで良好な結果が得られているので，$p = 5$ とする．Honeycomb ネットの自己組織化学習アルゴリズムにおいては，K-means 法による参照ベクトルの作成に膨大な時間が必要となる．よって，本稿ではこの部分のハードウェア化について検討する．

4 アルゴリズムの実装

先に説明した K-means 法による参照ベクトルの作成アルゴリズムを本システムに実装する．ホストと FCCB には，次のように計算を分割する．なお，FCCB 上の処理は Verilog-HDL を用いて記述する．

- ホスト
  ETL9B 2965 文字のデータから 1 文字分の 100 学習サンプルデータを抽出する．次に，その学習サンプルデータを 5 つのクラスタに均等に割り当てる．そして，FCCB 上の LM に各クラスタの学習サンプルデータを割り当てる情報を生成する．このとき，FCCB 上の PU の負荷分散を考慮して，LM0 から 7 までに 12 個のデータを，LM8 に残り 4 個のデータを割り付ける．

- FCCB
  K-means 法による参照ベクトルの計算を行なう．

次に，システム全体の流れを説明する．

- Step1
  ホストは，抽出した学習サンプルデータと割り当て情報を FCCB に転送する．

- Step2
  CU は，ホストから転送されるデータを割り当て情報に従って，図4に示すように LM に格納する．これにより，参照ベクトルを求める計算を各 PU に分散することが可能となる．各 LM には，1 語に最大 4 つの学習サンプルデータを格納する．このとき，全 LM を通して同一のアドレスに同じクラスタに属する学習サンプルデータが並ぶようにする．これにより，一つのクラスタに属する全学習サンプルデータを同時に PU にロード可能とする．

- Step3
  CU は，K-means 法に従って移動する学習サンプルデータを選択する．その学習サンプルデータを LM に持つ PU は，そのデータをどのクラスタにも含まれないようにする．

- Step4
  各 PU は，LM に格納しているクラスタ i に含まれる学習サンプルデータから合計ベクトルを求める．そして，各 PU で計算した結果を加算し，選択している学習サンプルデータを除いた合計ベクトル (以下 **Non 合計ベクトル**) を S_FIFO に格納する．また，各 PU の LM に格納しているクラスタ i に含まれる学習サンプルデータ数を合計し，学習サンプル数 k を求める．そして，CU の内部レジスタに格納する．すべての Non 合計ベクトルを求め，S_FIFO に格納したら Step5 へ．

- Step5
  S_FIFO に格納しているクラスタ i の Non 合計ベクトルと CU が求めた $1/k$ から，各 PU は参照ベクトルを求める．はじめに，その参照ベクトルとその参照ベクトルを持つクラスタに含まれる学習サンプルデータのユークリッド距離 (以下 Non 距離) を求める．また，選択している学習サンプルデータを持つ PU は，選択している学習サンプルデータと Non 合計ベクトルを加算し，LM に格納する．次に，LM に格納している合計サンプルデータと CU が求めた $1/(k+1)$ から各 PU は，選択した学習サンプルデータがクラスタ i に含まれる場合の参照ベクトルを求める．最後に，他の PU にブロードキャストし，選択している学習サンプルデータを含むユーク

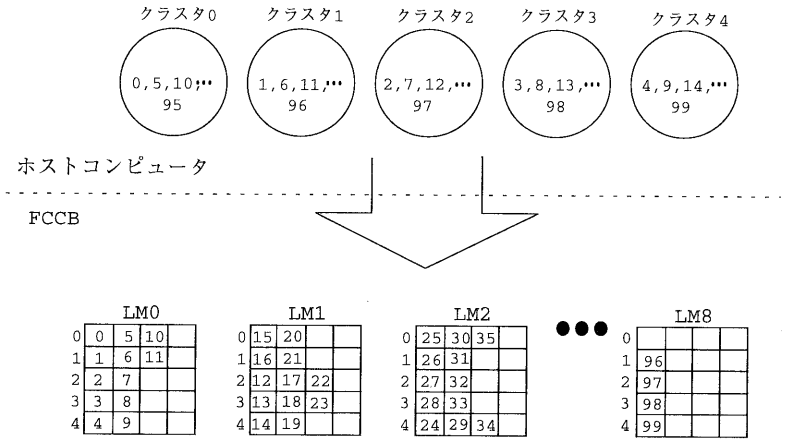

図 4: 初期状態の 1 例

リッド距離 (以下 Inc 距離) を求める．すべての Non 距離と Inc 距離を求めたら，Step6 へ．

- Step6
  各 PU で計算した Non 距離と Inc 距離を加算しながら PU8 に集める．それらの距離の値から各クラスタのユークリッド距離 $d(x, t_i(k))$ を求める．そして，式 (1) により，クラスタ j に選択している学習サンプルデータを移動する．

- Step7
  100 セットの学習サンプルデータの移動が終了したら，式 (2) を評価し，条件が成立すれば，求めた参照ベクトルをホストに転送し Step1 へ．そうでなければ，Step3 へ．

1 学習サンプルデータは，1 ワードを 3 バイトの浮動小数点データとする 768 ワードで構成する[1]．したがって，本学習で扱うデータ量は，2965 文字 × 100 セット × 768 ワード × 3 バイト = 683.136M バイトとなる．この大規模データをいかに効率的に処理するかが高速化のポイントとなる．FCCB では，学習サンプルデータを次のように処理する．

1. ギャザ処理
   Step4 において，各 PU で計算した Non 合計ベクトルは，図 5のように，RingBus を使って，隣接する PU に転送する．そして，最終的に，S_FIFO に各クラスタの Non 合計ベクトルを格納する．

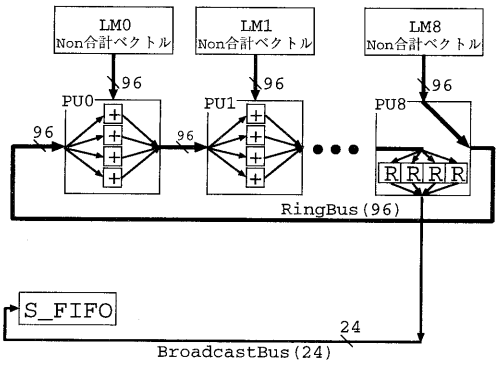

図 5: ギャザ処理

2. スキャタ処理
   Step5 において，S_FIFO は格納している Non 合計ベクトルをすべての PU にブロードキャストする．また，選択している学習サンプルデータを持つ PU は，合計ベクトルを他の PU にブロードキャストする．

[1] 学習サンプルデータは，外郭方向寄与度特徴 [16] で特徴を表現しているため，ワード数は 768 となる．

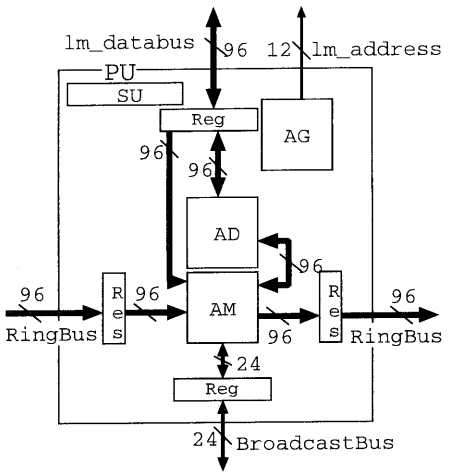

図 6: PU のモジュール構成

5 ハードウェア構成

5.1 FCCB 構成

以上のような計算を実現するために，FCCB 上の各ブロックを次のように用いる．

1. CU
   FCCB 全体の動作を規定するステートマシンを構成する．また，クラスタに含まれる学習サンプルデータ数 k を入力とし 1/k 浮動小数点データを出力とするルックアップテーブル (以下 LUT) を搭載する．

2. PU
   PU は，図 6に示すように，以下のモジュールに機能分割する．

   - State Controller(SC)
     PU 内部の動作を規定するステートマシンを構成する．
   - Address Generator(AG)
     LM に格納している学習サンプルデータを管理する．
   - ADders(AD)
     5 個の 24 ビット浮動小数点加算器を搭載し，累積演算を行なう．
   - Adder & Multiplier(AM)
     24 ビット浮動小数点加算器 4 個と乗算器 5 つを搭載し，和積演算を行なう．

3. LM
   学習サンプルデータを格納する．

4. S_FIFO
   Non 合計ベクトルを格納する．

表 1: 24 ビット除算器の仕様

ターゲットデバイス	XC4000XV-1
CLB 数	928
クリティカルパス	1487.33ns

5.2 参照ベクトルの計算

Step5 におけるの参照ベクトルを求める計算は以下に示すように，除算器を必要とする．

$$
\begin{aligned}
for \quad & (j = 0; j < 768; i++) \\
& ref.feat[j] = sum\_smp[i].feat[j]/k; \\
& k \text{ はクラスタ } i \text{ に含まれる} \\
& \text{学習サンプルデータの数}
\end{aligned}
$$

しかし，除算器は，表 1に示すように，回路規模とクリティカルパスが大きい．演算サイクル時間は，加算器，乗算器の動作速度を 1 とすると，除算器は 5 以上となり，大きなディレイを伴う．よって，除算器の代わりに LUT を用いる．LUT は，k を入力値とし，$1/k$ を出力するように設計する．

5.3 可変パイプライン処理

Step5 において，AM と AD は，LM に格納している学習サンプルデータの配列により，1 サイクルで処理する並列度が異なる．そのため，入力するワード数に応じて，演算器の内部接続とパイプライン数を適宜変更する．これを可変パイプライン処理と呼ぶ．入力データ数が 4 の場合と 2 の場合について，AM と AD の内部構成を図 7に示す．1 サイクルで入力するワード数の情報は，AG で生成する．図 7 の (a) の場合，ワード情報により，AM は 4 つのパイプラインを構成する．また，AD は入力 4 の累積演算器を構成する．同様に，図 7の (b) の場合，AM は 2 つのパイプライン，AD は入力 2 の累積演算器を構成する．

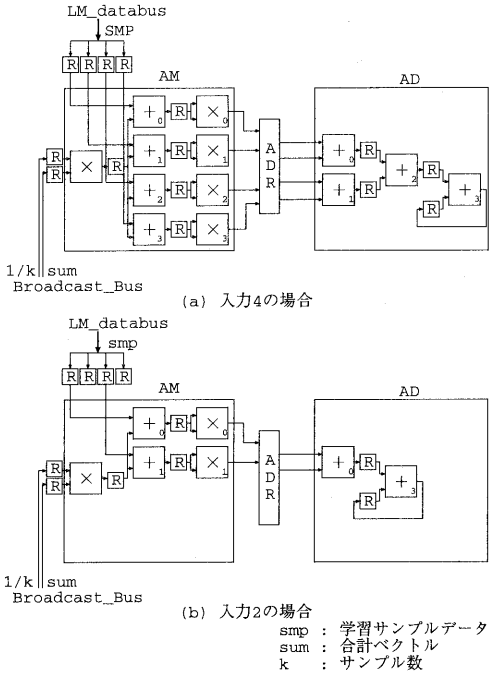

図 7: 可変パイプライン処理

6 予備評価

4 章で述べた設計で得られる演算サイクル数の比を示す.

ギャザ処理において, 学習サンプルデータを用いた C プログラムと FCCB で実行する演算サイクル数を同一とし, 768 と仮定する. この計算で得られる演算サイクル数の比は,

$$768 * 100 * 5/((768 + 192) * 5 + 192) = 76.9$$

となる. また, スキャタ処理において, 同様に仮定すると, この計算で得られる演算サイクル数の比は,

$$768 * 100 * 5/(768 * 10) = 50$$

となる. よって, 学習サンプルデータを用いた計算で得られる演算サイクル数の比は,

$$768 * 200 * 5/(((768 + 192) * 5 + 192) + 768 * 10)$$
$$= 60.6$$

となり, C プログラムと比較すると, 約 1/60 の演算サイクル数となる.

また, 本システムの性能を評価する為に, 初期状態から 1 つの学習サンプルデータの移動までの演算サイクル数を計算した. そして, WS 上で C 言語プログラムを実行する場合の演算サイクル数と比較した. その結果を表 2に示す.

表 2: 演算サイクルの比較

WS(Ulata2)	FCCB	比率
7,155,498	13,458	531.7:1

本システムの演算サイクル数は, WS の約 1/531 となった. WS の演算サイクルは, C 言語プログラムコンパイル結果から, 1 命令を 1 サイクルと仮定して求めた. なお, コンパイルには gcc を利用し, その最適化には O2 を用いた.

この結果, ギャザ/スキャタ処理の演算サイクル比 60.6 以上に, カスタム・コンピューティングにおける可変パイプライン処理や高いバンド幅のデータ供給などによる高速化が期待できる.

7 むすび

本稿では, FPGA を用いたカスタム計算ボード FCCB のアーキテクチャと 2 段のニューラルネットで構成する Honeycomb ネットの自己組織化学習アルゴリズムへの適用について述べた. また, その中心的な部分の演算サイクル数を予備評価し, WS 上の C 言語プログラムと比較した. その結果, FCCB は演算サイクル数を約 1/531 に短縮できることがわかった.

今後の課題として, Honeycomb ネット全体を FCCB とホストとの協調処理により実現することが挙げられる. その場合, 2 段のニューラルネット全体を FCCB に実装することはハードウェア規模の観点から困難である. そこで, Honeycomb ネットを BP ニューラルネット, ベクトル量子型ニューラルネットとその自己組織化学習を形成する 2 つの処理の 4 つに分割する. そして, FCCB 上で高速化可能なアルゴリズムに対して, FPGA の動的再構成機能を利用して実装することを検討中である. BP ニューラルネットについては, XILINX 社の XC40125XV-1 を用いた FCCB により, Sun Ultra2 ワークステーション上の C プログラムに対して約 95 倍の高速化

が可能であることを確認している．今後は，ベクトル量子型ニューラルネットのハードウェア化についての検討とその実装を進めると共に，動的再構成型FCCBの利害得失について考察する予定である．

謝辞日頃より御指導、御助力を頂いた宇都宮大学工学部情報工学科馬場、吉永、大津研究室の諸氏に感謝する。

本研究の一部は東京大学 大規模集積システム設計教育研究センターより提供して頂いたCADツールを使用した。深く感謝する。

本研究は一部文部省科学研究費 基盤研究(C)課題番号09680324、基盤研究(B)課題番号10558039、奨励研究(A)課題番号09780237の援助による。

参考文献

[1] 広瀬佳生，安仏英明，山下公一，後藤源助，“バックプロパゲーション専用プロセッサのアーキテクチャ”，信学技報ICD92-17，pp.40-46，1992.

[2] Y. Kondo, Y, koshiba, Y. Arima,M. Murasaki, T. Yamada, H. Amishiro, et.al., “A 1.2FLOPS Neural Network Chip Exhibiting Fast Convergence”, ISSCC '94 IEEE International Solid-State Circuits Conference, TP 13.1,pp.218-219,Feb.1994.

[3] 齋藤修，相原公久，藤田修，内村国治，“大規模ニューラルネットワークに対応可能な学習機能内蔵ディジタルニューロチップ”，信学技報CPSY98-4，pp.25-32，1998.

[4] M. Murray, M. Laung, K. Boonyanit, K. Kritayakirana, G. J. Burr, J. B. nad Wolff, T. Watanabe, E. Schwartz, D. G. Stork, A. M. Peterson, “ Digital Boltzmann VLSI for constraint satisfaction and learning. ” Advances in Neural Network Systems,Vol.6:896-903,1994.

[5] 柴田克成，安永守利，大山光男，益田昇，柳生正義，浅井光男，山田稔，坂口隆宏，橋本雅，“高速学習型ニューロWSIのシステム設計”，信学技報ICD90-127，pp.49-56，1990.

[6] 末吉敏則，“Reconfigurable Computing Systemの現状と課題-Computer Evolutionへ向けて-”，信学技報CPSY96-91，pp.111-118，1996-12.

[7] Maya Gokhale,Bill Holmes,Andrew Kopser, Dick Kunze,Dan Lopresti,Sara Lucas,Ron Minnich, “SPLASH:A Reconfigurable Linear Logic Array”, January 26,1994.

[8] M.Wazlowski, L.Agarwal, T.Lee, A. Smith, E.Lam, P.Athanas, H.Silverman, S.Ghosh, “ PRISM-II Compiler and Architecture ”, Proc. IEEE Workshop on FPGAs for Custom Computing Machines,pp.147-157,1994.

[9] Jean-Luc Beuchat,Jacques-Oliver Haenni and Eduardo Sanchez, “Hardware Reconfigurable Neural Networks”, LNCS1388. Rolim Parallel and Distributed Processing, 1998.

[10] Michael J . Wirthlin , Brad L . Hutchings, “DISC:The dynamic instruction set computer”, Proc. SPIE-Int. Soc. Opt. Eng. (USA), vol. 2607, pp.92-103.1995.

[11] David Abramson,Paul Logothetis,Adam Postula,Marcus Randall, “FPGA Based Custom Computing Machines for Irregular Problems”, HPCA,pp324-333,1998-4.

[12] 竹谷史生，吉永努，荒井正之，大津金光，馬場敬信，“FPGAを用いた手書き漢字認識ニューラルネットのハードウェア支援”，情処研報ARC98-7 1998-8.

[13] 荒井正之，王晋申，奥田健三，宮道壽一，“Honeycomb ネットによる多字種の手書き漢字認識”，信学論 (D-II)，J76-D-II，11，pp.2316-2323，1993-11.

[14] T. kohonen, “ Self - Organization Map ” , Proc. IEEE, 78, 9, 1990-9.

[15] ハイ，森下，蕪山，伊崎，山本，“手書き漢字認識におけるテンプレート複数化の検討”，信学技報，PRL81-42，pp.49-56，1981-9.

[16] 萩田泰一，内藤誠一郎，増田功，“外郭方向寄与度特徴による手書き漢字の識別”，信学論，J66-D，10，pp.1185-1192，1983-10.