マルチメディア通信と分散処理 76－24
（１９９６．５．１７）

任意の有限状態機械を学習するニューラルネットワークによる通信プロトコルの実装

西垣 正勝　佐藤 文明　水野 忠則
静岡大学情報学部情報科学科
〒432 静岡県浜松市城北 3-5-1
Tel:053-478-1467　Fax:053-475-4595
e-mail:nisigaki@cs.inf.shizuoka.ac.jp

本稿は任意の有限状態機械を学習するニューラルネットワークを提案する. 一般に, 通信プロトコルの動作は有限状態機械で表されるため, 本ネットワークはプロトコルのハードウェアによる実装を可能にする.

高速な通信が要求されるマルチメディア通信や, メモリ領域の少ない計算機を端末として通信を行うモーバイル通信では, プロトコルのハードウェアによる実装が不可欠となる. しかし, FDT(形式記述) からの定式的な回路合成手法は未だ確立されていない. 本稿では, 自己組織化能力を有するニューラルネットワークを活用し, 状態遷移表から所望の動作を学習する回路を構築する. プロトコル回路は学習により自動合成される.

ソフトウェアによるプロトコルの実装が主流となっていた理由の一つに, その複製・変更が容易であるという点が挙げられる. ニューラルネットワークの学習は素子間の結合荷重を調節することにより行われるので,(トポロジ的には) 同一のハードウェア構成で任意のプロトコルの実現が可能である. 本稿では, 結合荷重をプログラマブルなスイッチで実装することにより, 書き換え可能なプロトコル回路を実現する.

本稿ではプロトコルの制御部の実装に焦点を当てる.

Hardware Implementation of Communication Protocols By Neural Networks Representing Finite State Machines

Masakatsu NISHIGAKI, Fumiaki SATO and Tadanori MIZUNO
Faculty of Information, Shizuoka University
3-5-1, Johoku, Hamamatsu, 432, JAPAN

This paper describes a neural network which can represent the behavior of any finite state machine. Proposed network is applied to hardware implementation of communication protocols.

Recent trend toward high speed communication of multimedia sources requires the hardware implementation of communication protocols. In addition, protocol hardware must be preferable for mobile computing, since the mobile host is usually small and has limited memory capacity. So far, however, translation of protocol specifications described by formal description techniques to VLSI circuits has not been matured yet. This paper proposes to apply neural network to hardware implementation of protocols. Neural network is self-organized by training to obtain the functions of given protocol. That is, protocol circuit is synthesized automatically during the learning process.

The software implementation has a big advantage, i.e., its modification and copy are very easy. Thus protocols have been often implemented by software. Neural network can realize any kind of protocol by adjusting the Synaptic weights. Therefore, its hardware configuration remains unaltered. This paper suggests to fabricate Synaptic weights by programmable switches. The protocol circuit presented here can be easily modified and copied just by rewriting the Synaptic weights.

1. 序論

情報通信システムや分散システムにおいて, 計算機間を接続するために通信プロトコルが必須である. 近年は, LAN や ISDN 等の高速通信網が発展し, また, 各種情報処理装置の高機能化・高性能化が著しい. これにともない, ネットワーク上のサービスが拡大し, 大量なマルチメディアデータを高速に通信する必要性が急速に高まっている. 更に, 最近は移動体通信が隆盛を迎えつつあり, 無線通信の連続性 (コネクティヴィティの問題やシームレス通信) をサポートするプロトコルが強く望まれている. 今後, 社会の高度情報化が益々進むにつれ, より高速かつ信頼性の高い通信プロトコルが要求される.

高速な通信が要求される場合やメモリ領域の少ない計算機を端末として通信を行う場合には, プロトコルはハードウェアで実装されるべきである [1]. 画像などの大容量データを扱うマルチメディア通信には高速なプロトコルが不可欠であり, また, 小型化を強いられる移動体端末ではメモリ容量が現実に制限されることになる. すなわち, ハードウェアによるプロトコルの実装は, マルチメディア通信, モーバイル・コンピューティングの実現に対する鍵と考えられる.

従来までに, FDT(Formal Description Technique) を用いたプロトコルの形式記述から回路を自動合成する研究が行われているが, その手法は未だ完全に確立されてはいない [2]. また, 通常, 回路の自動合成では, NP 完全である論理関数の最小化が問題となる [3].

本稿では, 自己組織化能力を有するニューラルネットワークを活用し, 状態遷移表から所望の動作を学習する回路を構築する. 一般に, 通信プロトコルの動作は有限状態機械で表されるため, 本ネットワークはプロトコルのハードウェアによる実装を可能にする. ニューラルネットワークの自己組織化過程は回路合成に対応しており, プロトコル回路は学習により自動合成される. 更に本稿で提案するネットワークは論理関数のフーリエ級数展開に基づいており, 効率の良い回路合成が期待できる.

動作の無矛盾性・完全性が要求されるプロトコルにおいては実装後にその動作改善のための仕様変更が行われることが少なくない. そして, 修正の度に, プロトコルは計算機の数だけ複製され, 全ての計算機に再インストールされる. ソフトウェアによるプロトコルの実装が主流となっていた大きな理由の一つに, その複製・変更が容易であるという点が挙げられる. 仕様変更の度にハードウェア構成が変わってしまうという柔軟性に対する問題, また, ハードウェアでは部分的な修正が非常に難しい (例えば, あるゲートの論理を修正するとそのゲートにつながる全ての出力が変更されることになるため, 結局は回路全体の再合成が必要となる場合が多い) 点などが, ハードウェアによるプロトコルの実装に対する障害となっている.

ニューラルネットワークは素子間の結合荷重を調節することにより所望の動作を獲得するため, (トポロジ的には) 同一のハードウェア構成で任意のプロトコルを実装できる. また, 学習をし直すことにより回路は容易に変更が可能である. 更に本稿では, 結合荷重を {1,0,−1} に制限したネットワークを提案する. {1,0,−1} に離散化された結合荷重はプログラマブル・スイッチで実装することができる. この結果, プロトコル回路は PLA(プログラマブル・ロジック・アレイ) に見られるようなアレイ型の書き換え可能な回路として実現される. 本稿ではこれをプログラマブル・ニューロン・アレイ (PNA) と名付ける.

ニューラルネットワークによるプロトコル回路の実装には多くの長所が存在する.

1. 回路は学習により自動合成されるため, 設計者は仕様を決定するだけで良く, 詳細設計から開放される.
2. より効率の良い合成が期待できる. 回路の自動合成では, NP 完全である論理関数の最小化が問題となる [3]. 本回路の論理合成のベースであるフーリエ級数展開は線形変換であるので, 最急降下アルゴリズムにより任意の初期点から最適な結合荷重 (最小解) に有限時間内に到達可能であると推測される [5].
3. ニューラルネットワークは回路全体が並列に動作するため, 応答速度においてディジタル回路に勝る潜在能力を有する [4]. (論理回路では多数のゲートが縦続接続している部分の遅延が問題となる.)
4. 仕様変更などに柔軟な回路が実現する. プロトコル回路の修正や複製は, PNA のスイッチ・マトリクスの書き換えを行うだけでよい.

2. ニューラルネットワークによる万能順序回路

プロトコルの制御は, 一般に, 有限状態機械で表される. そして, 任意の有限状態機械は順序回路により実装することが可能である. 順序回路は論理回路 (組合せ回路) と記憶回路から成る. 本章では, 論理関数, 順序関数を学習するニューラルネットワークについて述べる.

2.1 論理回路の実現

次式で表される N_{in} 入力の論理関数 (ブール関数) を学習するニューラルネットワークを考える.

$$y = f(x_{N_{in}}, \cdots, x_2, x_1) \tag{1}$$

入力の各要素 x_i 及び出力 y は 0 または 1 である.

ここで,

$$u = \sum_{i=1}^{N_{in}} 2^{i-1} x_i \tag{2}$$

を用いて入力に対する 2 進－ 10 進変換を行うことにより，N_{in}次元の $\{x_i\}$ 空間は一次元の u 空間に写像される．u 空間には $2^{N_{in}}$ $(=N_{data})$ 個の点が存在し，各点には出力 $y(u)$ (0 または 1) が対応している．今，u を横軸に y を縦軸にプロットすると，$\{y(u) \mid u=0,1,\cdots,2^{N_{in}}-1\}$ はサンプリングされた N_{data} $(=2^{N_{in}})$ 個の点列と捉えることができる．この点列が N_{data} ごとに繰り返す周期的な点列であると仮定すると，いかなる点列 $\{y(u)\}$ もフーリエ級数

$$y(u) = a_0 + \sum_{i=1}^{N_f}\{a_i \cos(\frac{2\pi iu}{N_{data}}) + b_i \sin(\frac{2\pi iu}{N_{data}})\}$$
$$\text{for} \quad u=0,1,\cdots,N_{data}-1 \qquad (3)$$

により完全に表現することが可能である．ここで，点列 $\{y(u)\}$ の最高調波 $\{y(u)\}=\{\cdots,0,1,0,1,\cdots\}$ が $\cos(\pi u)$ で表されることから，最高調波数 N_f は $N_{data}/2=2^{N_{in}-1}$ である．

従って，任意の論理関数は図 1 に示される 1 層のニューラルネットワークにより実現される．図中，各素子は入力 u に対して $\cos(\frac{2\pi iu}{N_{data}})$ または $\sin(\frac{2\pi iu}{N_{data}})$ の大きさの出力を呈する．これらの素子をトリグロンと呼ぶ．本ネットワークに必要なトリグロン数は $2N_f=2^{N_{in}}=N_{data}$ である．各結合荷重 $\{a_i\},\{b_i\}$ は次章に示される学習アルゴリズムにより自動的に最適化される．教師信号は真理値表から与えられる．

図 1 では関数近似の能力を向上させるために出力部にコンパレータが挿入されている．コンパレータは 0.5 以上の入力に対し 1 を出力し，その他の場合は出力 0 を呈する．コンパレータにより出力が補正されるため，ある程度

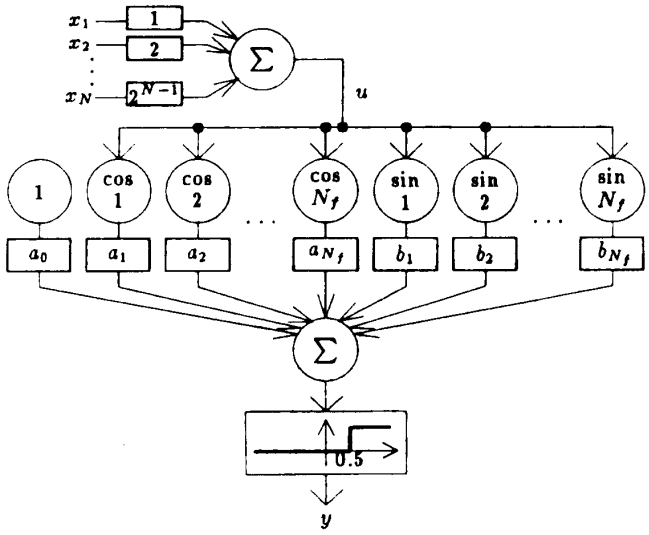


Fig.1. Neural network to approximate general logic functions, where $\overset{\cos}{i}$ and $\overset{\sin}{i}$ express trigrons to output $\cos(\frac{2\pi iu}{N_{data}})$ and $\sin(\frac{2\pi iu}{N_{data}})$, respectively.

なら結合荷重 $\{a_i\},\{b_i\}$ の調整は粗くても問題ない．

2.2 順序回路の実現

順序回路の動作は

$$\boldsymbol{Y}^t = g(\boldsymbol{X}^t, \boldsymbol{Q}^{t-1}) \qquad (4)$$
$$\boldsymbol{Q}^t = h(\boldsymbol{X}^t, \boldsymbol{Q}^{t-1}) \qquad (5)$$

により表される．ここで，$\boldsymbol{X}^t$, $\boldsymbol{Y}^t$, $\boldsymbol{Q}^t$はそれぞれ入力ベクトル，出力ベクトル，内部状態変数ベクトルである．式 (4),(5) は$\boldsymbol{Q}^{t-1}$を外部入力であると仮定することにより複数の論理関数の組に分解することができる．

$$Y_i^t = g_i(X_{N_{in}}^t, ..., X_2^t, X_1^t, Q_{N_s}^{t-1}, ..., Q_2^{t-1}, Q_1^{t-1})$$
$$i=1,2,...,N_{out} \qquad (6)$$

$$Q_i^t = h_i(X_{N_{in}}^t, ..., X_2^t, X_1^t, Q_{N_s}^{t-1}, ..., Q_2^{t-1}, Q_1^{t-1})$$
$$i=1,2,...,N_s \qquad (7)$$

ここで，N_{in}, N_{out}, N_sはそれぞれ$\boldsymbol{X}^t$, $\boldsymbol{Y}^t$, $\boldsymbol{Q}^t$の次数である．

従って，図 1 のネットワークを図 2 のように構成することにより順序回路を構築することができる．図 2 の回路には，次クロックが入るまで内部状態を保つための 0 次ホールド回路が投入されている．

設計者はプロトコルの仕様を状態遷移表 (または状態遷移図) により記述するだけで良い．本ネットワークは与えられた状態遷移表を基に自己組織化を行い，所望の動作を学習する．

3. 順序回路の PNA による実装

3.1 結合荷重の離散化

図 1, 図 2 の回路には，関数近似の能力を高めるためにコンパレータが付加されていた．コンパレータにより出力が補正されるため，ある程度なら結合荷重の調整は粗くても問題ない．ここで，コンパレータのしきい値を可変にすることにより，更に近似能力が向上することが期待できる．つまり，可変しきい値のコンパレータを用いた場合，結合荷重の値としていくつか離散値のみを用意すれば十分に論理関数を実現できると考えられる．

本稿では，結合荷重を {1,0,−1} に制限したネットワークを提案する．{1,0,−1} に離散化された結合荷重はプログラマブル・スイッチで実装することができる．結合荷重=1 は対応するトリグロンが加算器に接続されていることを意味する．結合荷重=−1 は内部で生成されているトリグロンの反転出力が加算器に接続されていることを意味する．結合荷重=0 は接続がないことを意味する．この結果，プロトコル回路は PLA(プログラマブル・ロジック・アレイ) に見られるようなアレイ型の書き換え可能な回路として実現される．本稿ではこれをプログラマブル・

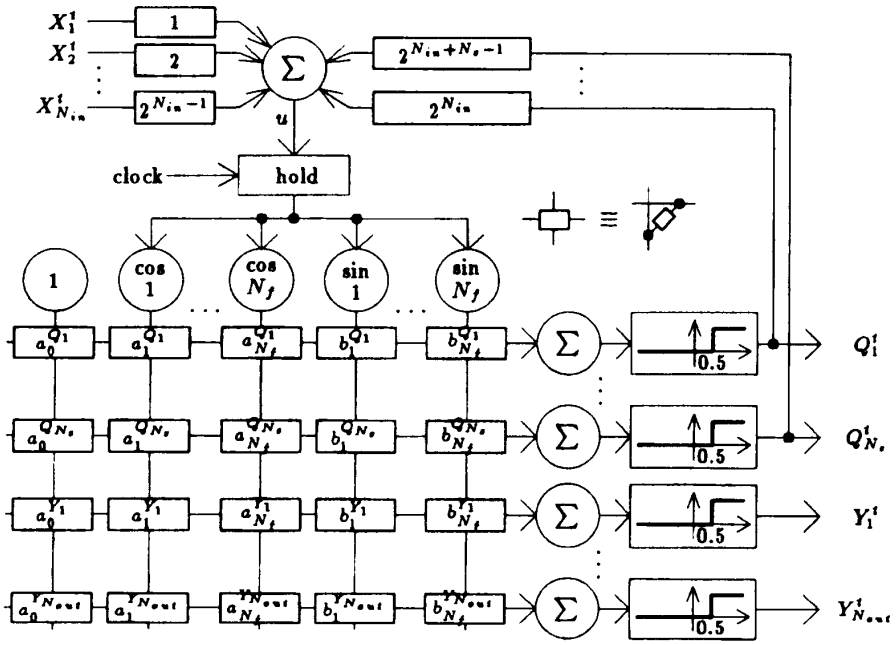


Fig.2. Neural network to approximate general sequential logic functions.

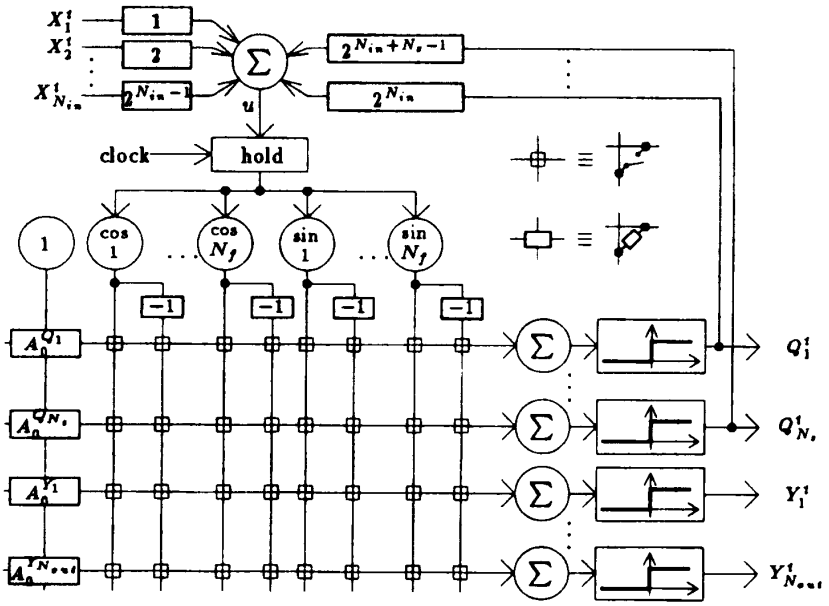


Fig.3. PNA implementation of universal digital sequential circuit.

ニューロン・アレイ (PNA) と呼ぶ. プロトコル回路の修正や複製は, PNA のスイッチ・マトリクスを書き換えるだけでよい. 回路を図 3 に示す.

なお, コンパレータの可変しきい値 θは, フーリエ級数の直流成分 a_0とまとめることができる. (つまり, $A_0^i = a_0^i - \theta^i$. ここで, $\{A_0^i\}$ のみが連続値の結合荷重である.)

3.2 PNA の学習アルゴリズム

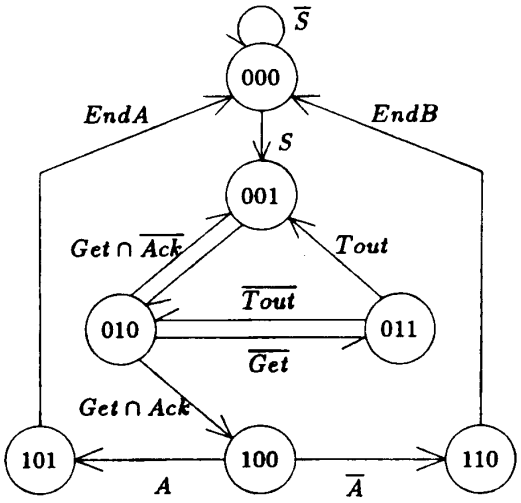
PNA の離散化荷重の学習には, 連続値の結合荷重が利用される.

2.2 で述べたように, 式 (1) で表される論理関数の近似が本回路の基盤である. したがって, ここでは簡略化のため, 式 (1) の学習に対して示す. 学習の目的は, 式 (8) を満たすように, 離散化荷重 $\{A_i\}$,$\{B_i\}$ を調整することである.

$$y(u) = \tilde{y}(u), \qquad \text{for} \qquad u = 0, 1, ..., N_{data} - 1 \quad (8)$$

ここで,

$$\tilde{y}(u) = sgn(A_0 + \sum_{i=1}^{N_f}\{A_i \cos(\frac{2\pi iu}{N_{data}}) + B_i \sin(\frac{2\pi iu}{N_{data}})\})$$
$$\text{for} \qquad u = 0, 1, ..., N_{data} - 1 \quad (9)$$



State 0 (000): idle.
State 1 (001): sending of request to a terminal.
State 2 (010): waiting to get a messege from the terminal.
State 3 (011): timer increment.
State 4 (100): decoding of the messege.
State 5 (101): processing of job_A.
State 6 (110): processing of job_B.

Fig.4(a). The state transition diagram for job selecter.

S	Get	Ack	$Tout$	A	$EndA$	$EndB$	Q_3^{t-1}	Q_2^{t-1}	Q_1^{t-1}	Q_3^t	Q_2^t	Q_1^t	$\overline{Busy}$	$Send$	$SetT$	$IncT$	E_D	E_A	E_B
0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	0	0	0	0	0	0	1	0	0	0	0	0	0
1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	0	0	0	0	0	1	1	0	0	0	0	0	0
ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	0	0	1	0	1	0	0	1	1	0	0	0	0
ϕ	0	ϕ	ϕ	ϕ	ϕ	ϕ	0	1	0	0	1	1	0	0	0	0	0	0	0
ϕ	1	0	ϕ	ϕ	ϕ	ϕ	0	1	0	0	0	1	0	0	0	0	0	0	0
ϕ	1	1	ϕ	ϕ	ϕ	ϕ	0	1	0	1	0	0	0	0	0	0	0	0	0
ϕ	ϕ	ϕ	0	ϕ	ϕ	ϕ	0	1	1	0	1	0	0	0	0	1	0	0	0
ϕ	ϕ	ϕ	1	ϕ	ϕ	ϕ	0	1	1	0	0	1	0	0	0	1	0	0	0
ϕ	ϕ	ϕ	ϕ	0	ϕ	ϕ	1	0	0	1	0	1	0	0	0	0	1	0	0
ϕ	ϕ	ϕ	ϕ	1	ϕ	ϕ	1	0	0	1	1	0	0	0	0	0	1	0	0
ϕ	ϕ	ϕ	ϕ	ϕ	0	ϕ	1	0	1	1	0	1	0	0	0	0	0	1	0
ϕ	ϕ	ϕ	ϕ	ϕ	1	ϕ	1	0	1	0	0	0	0	0	0	0	0	1	0
ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	0	1	1	0	1	1	0	0	0	0	0	0	0	1
ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	1	1	1	0	0	0	0	0	0	0	0	0	0	1

Fig.4.(b). The state table for job selecter.

である. また,

$$A_i = tre(a_i, Th), \qquad B_i = tre(b_i, Th)$$
$$i = 1, 2, ..., N_f \tag{10}$$

である. なお, 関数 $sgn(\cdot)$, $tre(\cdot)$ はそれぞれ,

$$sgn(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

$$tre(x, Th) = \begin{cases} 1, & \text{if } x \geq Th \\ -1, & \text{if } x \leq -Th \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

である. 学習は, 連続値の結合荷重 $\{a_i\}$,$\{b_i\}$ に対して行われる. そして, 関数 $tre(\cdot)$ が $\{a_i\}$,$\{b_i\}$ を離散化する.

連続値の結合荷重 A_0, $\{a_i \mid i = 1, 2, .., N_f\}$, $\{b_i \mid i = 1, 2, ..., N_f\}$ の調整には, 最急降下法により基づく学習アルゴリズムが用いられる. 学習式は, それぞれ,

$$\Delta A_0 = -\alpha_0 \cdot (\tilde{y}(u) - y_{true}(u)) \tag{13}$$

$$\Delta a_i = -\alpha_a \cdot (\tilde{y}(u) - y_{true}(u)) \cdot \cos(\frac{2\pi i u}{N_{data}}) \cdot \partial tre/\partial a_i$$
$$i = 1, 2, ..., N_f \tag{14}$$

$$\Delta b_i = -\alpha_b \cdot (\tilde{y}(u) - y_{true}(u)) \cdot \sin(\frac{2\pi i u}{N_{data}}) \cdot \partial tre/\partial b_i$$
$$i = 1, 2, ..., N_f \tag{15}$$

と導かれる. ここで, $y_{true}(u)$ は真理値表から得られる教師信号, α_0, α_a, α_bは学習パラメータである. なお, 学習の収束性を向上させるため,

$$\partial tre/\partial x = \begin{cases} 1 - |Th - |x||/\sigma, & \text{if } |Th - |x||/\sigma \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

とした. (正確には, $\partial tre/\partial x = \delta(x - Th) + \delta(x + Th)$; $\delta(x)$ はインパルス関数.) ここで, σは定数である.

回路を実装する前に, 結合荷重値を決定するために本アルゴリズムによる学習が計算機上で行われる. つまり, 学

習過程はPNAの接続パターン(スイッチ・マトリクス)の設計に対応している.

4. プロトコル実装例

係数を{1,0,−1}に離散化したフーリエ級数をコンパレータを通して補正することにより,全ての論理関数が実現できるという数学的な裏付けはない.プログラマブル・ニューロン・アレイ(PNA)の関数近似能力を測るために,数多くのシミュレーションを行った.その結果,$N_{in}+N_s \leq 4$の場合には全ての論理関数をPNAで実現可能であることを確認した.しかし残念ながら,$N_{in}+N_s > 5$の場合には実現不可能な論理関数が見つかった.そして,$N_{in}+N_s$が大きくなるとともに,実現不可能な関数の数も増加した.

これに対し,荷重結合の離散化をより細かくすることにより,PNAの近似能力が向上することを確認した.

通信プロトコルの制御部論理を実装する例として,「端末の指示に従いジョブAまたはBを実行させる」という手続きを行う回路をPNAに学習させた.本回路の状態遷移図,状態遷移表を図4(a),(b)に示す.状態0はアイドリングである($\overline{Busy}$信号が出力されている).スタート信号Sを受け,状態1で端末にジョブの選択をうながす要求メッセージが送信される(送信器をenableにする信号$Send$を出力).同時にタイマがセットされる(タイマに対し$SetT$信号を出力).その後,回路は状態2に移り,端末からジョブの指定メッセージが返送される(着信フラグGetが立つ)のを待つ.指定メッセージの着信がない場合,逐次,状態3でタイマがインクリメントされる(タイマに対し$IncT$信号を出力).タイムアウト($Tout$フラグが立つ)とともに回路は状態1に戻り,端末への要求メッセージを再送する.状態2で端末から受け取ったジョブの指定メッセージが正しい場合,着信フラグGetとともに受理フラグAckが真となる.Ackが偽の場合には状態1に戻り,端末への要求メッセージを再送する.Ackが真の場合,回路は状態4に移る.着信メッセージには,指定ジョブのみが記述されているわけではなく,当然,送信時に必要となるヘッダなどが付加されている.状態4では,メッセージからジョブ記述部を抽出するために,着信メッセージをデコードする(デコーダのenable信号E_Dが出力される).指定されたジョブがAの場合,フラグAが真となり,状態5でジョブAが開始される(E_Aを出力).ジョブAが終了する(ジョブA終了フラグ$EndA$が立つ)と回路は状態0に戻る.同様にジョブBが指定された(フラグAが偽)場合,回路は状態6に移り,ジョブBが実行される.ジョブBの終了とともに回路は状態0に戻る.

PNAの結合荷重の学習を*3.2*のアルゴリズムにより行った.結合荷重の値は{5,1,0,−1,−5}とした.各パラメータは経験的に,α_0=0.015(式(13)),α_a=0.015(式(14)),α_b=0.015(式(15)),Th=0.4(式(10)),σ=1.0(式(16))とした.全ての結合荷重を0.0に初期化した後,学習を行った.数十回の学習で適切な結合荷重へ収束し,本回路の動作を獲得することを確認した.

5. 結論

本稿は任意の有限状態機械を学習するニューラルネットワークを提案した.ニューラルネットワークは,応答が高速である,自己組織化能力を有する,同一のハードウェア構成により任意の動作を実現する等の長所を持ち,プロトコルを実装するに適した回路であると言える.更に本稿では,ネットワークの結合荷重を離散化し,これをプログラマブル・スイッチで実装することにより,書き換え可能なプロトコル回路を実現した.ハードウェアによるプロトコルの実装が不可欠となるマルチメディア通信やモーバイル通信における,本ネットワークの寄与は大きい.

本稿のニューラルネットワークはフーリエ級数展開を利用して論理関数を実装する.ここではプログラマブル・ニューロン・アレイ(PNA)の回路構成と学習アルゴリズムについて述べた.

実際に,簡単なプロトコルの制御部をPNAにより実装し,その適用可能性を示した.

入力数,状態数が多くなると,素子(トリグロン)数や学習データ数が爆発する.将来的に実用的な規模のプロトコルを実装する場合には,回路分割などの対処が必要になると考える.

参考文献

[1] 井戸上,加藤,鈴木,パーソナルコンピュータおよびワークステーションのためのOSI7層ボードの実装と評価,情報処理学会誌,**vol.36**, 3, pp.763-774, Mar. 1995.

[2] 黄,安本 他,LOTOS風言語で書かれた同期式順序回路の要求仕様記述と回路自動合成,情報処理学会研究報告 マルチメディア通信と分散処理,**66-9**, pp.49-54, July. 1994.

[3] K.A.Bartlett, R.K.Brayton, et al., *Multilevel logic minimization using implicit don't cares*, IEEE Trans. Computer-Aided Design, **7**, 6, pp.723-740, June 1988.

[4] M.Morisue, K.Sakai and T.Iizuka, *Neural networks for digital adder*, IEEE proc. ISCAS'91, pp.1605-1608, June 1991.

[5] J.J.Hopfield and D.W.Tank, *'Neural' Computation of Decisions in Optimization Problems*, Boil. Cybern., **52**, pp.141-152, 1985.