SOM^{2}のFPGA化:
ハードウェア化に適したアルゴリズムの実装と検証

SOM^2 on FPGA:
Implementation and verification of suitable algorithm for hardware

○田中 利康，金子 宗司，田向 権，徳永 憲洋，古川 徹生
Toshiyasu Tanaka, Syuji Kaneko, Hakaru Tamukoh, Kazuhiro Tokunaga, Tetsuo Furukawa
九州工業大学大学院生命体工学研究科
Email： tanaka-toshiyasu@edu.brain.kyutech.ac.jp

2007年3月29日（March 29th, 2007）

1 はじめに

我々は自己組織化マップ (SOM) の参照ベクトルユニットの代わりにSOMを用いるSOMのSOM，すなわちSOM^{2}を提案してきた[1][2][3]．SOMはデータベクトルの集合を類似関係を保持したまま低次元の特徴空間へ写像する．この特徴空間はデータベクトル空間では低次元の多様体を表現している．SOM^{2}では各SOMが作り出す多様体の集合の自己組織化マップを作ることができる．

現在SOM^{2}は顔画像分類，風景識別など様々な分野への応用が期待されている．しかしこれらは演算量が膨大であり，高速化が望まれる．そこで我々はSOM^{2}の高速演算が可能なハードウェアの作成を目指している．本稿ではSOM^{2}のベースとなるBatch-Learning SOM (BL-SOM) およびSOM^{2}のハードウェア化について検討をする．それぞれについてハードウェア化に適したアルゴリズムを考え回路を設計し，ソフトウェアによるシミュレーションで学習精度の違いを検討したのでこれを報告する．

2 BL-SOMのハードウェア化

2.1 BL-SOMの学習アルゴリズム

SOM^{2}はBL-SOMの学習アルゴリズムをベースにしている．したがってまずBL-SOMをハードウェアに適した形にする必要がある．

BL-SOMの学習は競合過程，協調過程，適応過程という3つの過程の繰り返し計算である．

最初の競合過程では，全ての入力データベクトルについて，全てのユニットの参照ベクトルとの間の誤差を求め，それが最小となるユニットを勝者（Best Matching Unit: BMU）に決定する．ここで誤差とは2つのベクトル間のユークリッド距離，すなわちL_2ノルムである．i番目の入力データベクトルを$\boldsymbol{x}_i(i = 1, ..., I)$，$k$番目ユニットの参照ベクトルを$\boldsymbol{w}^k$とすると，誤差は次式によって求められる．

$$E_i^k = L_2\left(\boldsymbol{x}_i, \boldsymbol{w}^k\right) \tag{1}$$

式(1)で求めた誤差E_i^kが最も小さくなるk番目ユニットがi番目データに対する勝者となる．勝者のユニット番号をk_i^*とすれば，BMUの番号は以下の式で求められる．

$$k_i^* = \arg\min_k E_i^k \tag{2}$$

次に協調過程では，各ユニットの学習量を計算する．k番目ユニットがi番目データを学習する量（学習分配率）$\alpha_i^k(T)$は次式で求められる．Tは学習回数，$h[\cdot,\cdot]$は近傍関数を表し，一般的にはガウシアン関数が用いられる．

$$\alpha_i^k(T) = \frac{h\left[d\left(k, k_i^*\right), T\right]}{\sum_{i'=1}^{I} h\left[d\left(k, k_{i'}^*\right), T\right]} \tag{3}$$

ここでオンライン型のSOMとの違いは，式(3)のように近傍関数を正規化するかしないかである．正規化することでユニットの全データに対する学習量が1になる．

最後に適応過程では，各ユニットの参照ベクトルを次式に従って更新する．参照ベクトルは $\alpha_i^k(T)$ を重みとする入力データベクトルの重み付き平均で更新される．

$$\begin{aligned}\boldsymbol{w}^k(T+1) &= \sum_{i=1}^{I} \alpha_i^k(T)\boldsymbol{x}_i \\ &= \frac{\sum_{i=1}^{I} h\left[d\left(k,k_i^*\right),T\right]\boldsymbol{x}_i}{\sum_{i'=1}^{I} h\left[d\left(k,k_{i'}^*\right),T\right]}\end{aligned} \tag{4}$$

2.2 BL-SOM のハードウェアに適したアルゴリズム

SOM のユニットを多く実装することと高速演算を実現することを重視し，並列処理およびパイプライン処理ができるハードウェアアーキテクチャを考えた．学習アルゴリズムの計算式の一部で通常の SOM より単純な式を用いることにより回路面積の縮小と同時に高速演算も実現できる．提案手法では競合過程のユークリッド距離演算をマンハッタン距離演算とし，近傍関数をガウシアン関数から矩形関数に変更した．

距離演算の変更

ユークリッド距離演算をする場合，乗算器が必要になることから回路面積が大きくなる．マンハッタン距離演算にすることで乗算器が不要になる．したがって競合過程の式 (1) をマンハッタン距離，すなわち L_1 ノルムに変更した．

$$E_i^k = L_1\left(\boldsymbol{x}_i, \boldsymbol{w}^k\right) \tag{5}$$

更新式の簡単化

BL-SOM アルゴリズムは全てのデータが入力されてから一括で更新するため，式 (3) の α_i^k の値を全て保持しなければならない．α_i^k のため（SOM のユニット数）×（入力データベクトル数）分のメモリ領域を確保しなければならないことになり，回路面積が非常に大きくなる．

そこで近傍関数として次式に示す矩形関数を用いることで式 (3)，(4) を合わせて次のように更新式を変更した．

$$h\left[d\left(k,k_i^*\right),T\right] = \begin{cases} 1 & d\left(k,k_i^*\right) \le \sigma(T) \\ 0 & d\left(k,k_i^*\right) > \sigma(T) \end{cases} \tag{6}$$

$$\begin{aligned}\boldsymbol{w}^k(T+1) &= \frac{\sum_{i\in H}^{I} \boldsymbol{x}_i}{\sum_{i'\in H}^{I} h\left[d\left(k,k_{i'}^*\right),T\right]} \\ &= \frac{\boldsymbol{S}^k(T)}{N^k(T)}\end{aligned} \tag{7}$$

ここで，$d(k,k_i^*)$ は BMU と他のユニットとの特徴空間でのマンハッタン距離，$\sigma(T)$ は近傍半径，H は BMU およびその近傍として選ばれたユニットの集合である．

矩形関数ではガウシアン関数の値を生成する回路を実装する場合と比べて回路規模を節約できる．また矩形関数を採用することにより式 (7) の $\boldsymbol{S}^k(T)$ はスイッチ付きの加算回路，$N^k(T)$ はカウンタ回路により実装できる．必要とするメモリは (SOM のユニット数) 分となり，回路面積を大幅に節約できる．

2.3 BL-SOM のハードウェア実装

回路設計の概要

ハードウェアの実装には FPGA (Field Programmable Gate Array) を用いる．FPGA は IC の一種であり，通常の IC は機能を決めて工場で製造されると後から変更できないが，FPGA は機能の書き換えが可能という特徴を持つ．また機能の設計には一般的な PC を用いることができるため，研究室などで短期間で設計・実装して実験することができる．

前述のアルゴリズムを実現する回路を図 1，2 に示す．BL-SOM 回路の構成は大きく分けると制御器，各ユニット回路の 2 つである (図 1)．制御器から出力される信号によってその他の回路が動作し，回路間で信号のやりとりが行われる．

ユニット回路 (図 2) は SOM の 1 つのユニットに相当する．参照ベクトルブロック，距離演算ブロック，Winner-Take-All (WTA) ブロック，入力データ加算ブロック，近傍関数値加算ブロック，除算器の 6 つがある．このうち参照ベクトルブロックと入力データ加算ブロックは入力データベクトルの次元数分のメモリ回路となっている．

パフォーマンス

このハードウェアアーキテクチャとアルゴリズムにより BL-SOM を実行する場合，1 つの入力データベクトルを学習する学習モードに必要なクロックサイクル数 Clock は，距離演算ブロック，WTA ブロック，入力データ加算ブロックおよび近傍関数値加算ブ

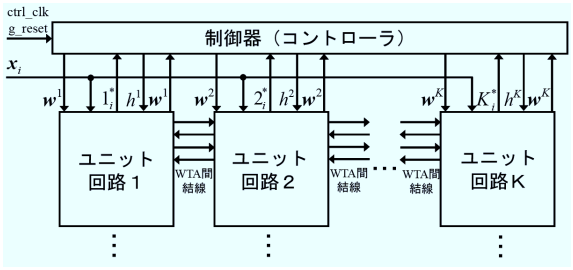


図 1: BL-SOM のグローバル回路

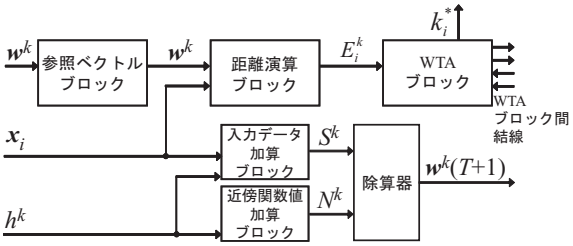


図 2: BL-SOM のユニット回路

ロック，除算器を用いた参照ベクトルの更新，初期化に必要なクロックサイクル数 (C_{Dist} ，C_{WTA}，C_{Sum}，C_{Update}，C_{Init}) を用いて次のように求められる．

$$\begin{aligned}\mathrm{Clock} &= C_{\mathrm{Dist}} + C_{\mathrm{WTA}} + C_{\mathrm{Sum}} + \frac{C_{\mathrm{Update}} + C_{\mathrm{Init}}}{i} \\ &= m + \log_2(m \cdot 2^a) + m + \frac{\log_2(i \cdot 2^a) + 6}{i}\end{aligned} \tag{8}$$

ここで m は入力データ数，d は入力データベクトルの次元数，a は演算の精度 (bit) を表している．したがって，$m = 256$ ， $d = 4$ ， $a = 8$ とすると，1 入力データの学習あたりに必要なクロックサイクル数は 18 となる．

また，実行モードに必要なクロックサイクル数は以下の式で求められる．

$$\begin{aligned}\mathrm{Clock} &= C_{\mathrm{Dist}} + C_{\mathrm{WTA}} \\ &= m + \log_2(m \cdot 2^a)\end{aligned} \tag{9}$$

3 SOM^2 のハードウェア化

3.1 SOM^2 の学習アルゴリズム

SOM^2 は SOM の参照ベクトルユニットを SOM に置き換えたものである．SOM ではユニットが格子状に並んでいるのに対し，SOM^2 では SOM が格子状に並んでいる．本稿では SOM^2 の中にある SOM を

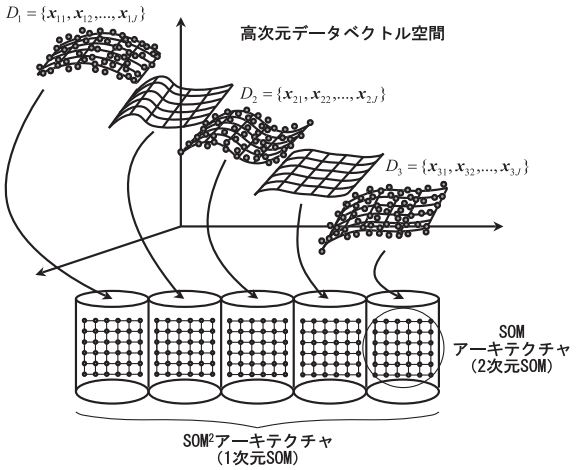


図 3: SOM 特徴空間の自己組織化マップ

SOM モジュールと表現する．SOM^2 のアーキテクチャを図 3 の右に示す．

SOM が多様体を表現するアーキテクチャとするならば，SOM^2 は多様体間のトポロジーを保存したまま多様体集合を表現するアーキテクチャである．3 つのクラスのデータ集合 D_1，D_2，D_3 が図 3 のように存在するとき，SOM^2 は 3 つの SOM モジュールが 3 つのクラスの多様体を表現し，その他の SOM モジュールが 3 つのクラスを内挿し，多様体の集合を特徴空間に写像している．

ベースとなった SOM のアルゴリズムと同様，SOM^2 のアルゴリズムは競合過程，協調過程，適応過程の 3 つの過程からなる．

J 個の多次元データベクトルで 1 つのクラス（多様体）を構成する入力データが I 個あるとする．その時の i 番目のクラスのデータセットは $D_i = \{\boldsymbol{x}_{i,1}, ..., \boldsymbol{x}_{i,J}\}$ と表現される．次に，1 つの SOM が L 個の参照ベクトルを持ち，K 個の SOM モジュールから構成される SOM^2 を考える．このときの参照ベクトルは $W^k = \left\{\boldsymbol{w}^{k,1}, ..., \boldsymbol{w}^{k,L}\right\}$ で表される．

最初の競合過程では，全ての入力データベクトル $\boldsymbol{x}_{i,j}$ について， 全 SOM モジュール内の全参照ベクトル $\boldsymbol{w}^{k,l}$ との誤差 $e_{i,j}^{k,l}$ を求める．

$$e_{i,j}^{k,l} = L_2\left(\boldsymbol{x}_{i,j}, \boldsymbol{w}^{k,l}\right) \tag{10}$$

$e_{i,j}^{k,l}$ が最も小さくなるユニットが i 番目クラスの j 番目データに対する勝者ユニット（Best Matching Unit: BMU）となる．BMU の番号を $l_{i,j}^{k,*}$ とすれば，BMU の番号は次式で記述できる．

$$l_{i,j}^{k,*} = \arg\min_{l} e_{i,j}^{k,l} \tag{11}$$

BMU になったユニットの距離は式 (10)，(11) から $e_{i,j}^{k,l_{i,j}^{k,*}}$ と書ける．1 クラス分のデータセットに対する $e_{i,j}^{k,l_{i,j}^{k,*}}$ の総和により各 SOM モジュールの誤差を求める．

$$E_i^k = \sum_{j=1}^{J} e_{i,j}^{k,l_{i,j}^{k,*}} \tag{12}$$

式 (12) で求めた E_i^k が一番小さいものを勝者モジュール（Best Matching Module: BMM）として，あるクラスをもっともよく表現している SOM モジュールと考える．この時の BMM の番号は以下の式で求められる．

$$k_i^* = \arg\min_k E_i^k \tag{13}$$

次に協調プロセスでは，SOM モジュール内の参照ベクトルの学習分配率 $\beta_{i,j}^l$ と SOM モジュール自体の学習分配率 α_i^k を求める．

$$\beta_{i,j}^l(T) = \frac{h\left[d\left(l, l_{i,j}^{*,*}\right), T\right]}{\sum_{j'=1}^{J} h\left[d\left(l, l_{i,j'}^{*,*}\right), T\right]} \tag{14}$$

$$\alpha_i^k(T) = \frac{g\left[d\left(k, k_i^*\right), T\right]}{\sum_{i'=1}^{I} g\left[d\left(k, k_{i'}^*\right), T\right]} \tag{15}$$

最後に適応プロセスでは，各 SOM モジュール内の全ての参照ベクトルを学習分配率 α_i^k と $\beta_{i,j}^l$ に従い更新する．ここで $V_i = \left\{\boldsymbol{v}_i^1, ..., \boldsymbol{v}_i^L\right\}$ は i 番目クラスのための参照ベクトルと定義する．すなわち V_i は i 番目クラスの多様体を推定した特徴空間ということである．

$$\boldsymbol{v}_i^l(T+1) = \sum_{j=1}^{J} \beta_{i,j}^l \boldsymbol{x}_{i,j} = \frac{\sum_{j=1}^{J} h\left[d\left(l, l_{i,j}^{*,*}\right), T\right] \boldsymbol{x}_{i,j}}{\sum_{j'=1}^{J} h\left[d\left(l, l_{i,j'}^{*,*}\right), T\right]} \tag{16}$$

この $\boldsymbol{v}_i$ と α_i^k を用いて以下の更新を行うことで，特徴空間の自己組織的なマップができる．

$$\boldsymbol{w}^{k,l}(T+1) = \sum_{i=1}^{I} \alpha_i^k \boldsymbol{v}_i^l = \frac{\sum_{i=1}^{I} g\left[d\left(k, k_i^*\right), T\right] \boldsymbol{v}_i^l}{\sum_{i'=1}^{I} g\left[d\left(k, k_{i'}^*\right), T\right]} \tag{17}$$

3.2 SOM^2 のハードウェアに適したアルゴリズム

ハードウェア化のため，BL-SOM の場合と同様の考えのもと次の 2 点を変更した．

まず競合過程の式 (10) をユークリッド距離演算からマンハッタン距離演算に変更した．

$$e_{i,j}^{k,l} = L_1\left(\boldsymbol{x}_{i,j}, \boldsymbol{w}^{k,l}\right) \tag{18}$$

次に近傍関数 $h[\cdot,\cdot]$ と $g[\cdot,\cdot]$ は式 (6) にある矩形の近傍関数を用い，式 (14) と (16)，(15) と (17) を合わせ更新式を次のようにした．

$$\boldsymbol{v}_i^l = \frac{\sum_{j\in H}^{J} \boldsymbol{x}_{i,j}}{\sum_{j\in H}^{J} h\left[d\left(l, l_{i,j}^{*,*}\right), T\right]} = \frac{\boldsymbol{S}_i^l(T)}{N^l(T)} \tag{19}$$

$$\boldsymbol{w}^{k,l} = \frac{\sum_{i\in G}^{I} \boldsymbol{v}_i}{\sum_{i\in G}^{I} g\left[d\left(k, k_i^*\right), T\right]} = \frac{\boldsymbol{S}\boldsymbol{S}^{k,l}(T)}{NN^{k,l}(T)} \tag{20}$$

3.3 SOM^2 のハードウェア実装

回路設計の概要

SOM^2 のハードウェア実装には BL-SOM の時と同様に FPGA を用い図 4，5，6 のような回路構成とした．図 5 は図 4 の SOM モジュール回路の中を，図 6 は図 5 のローカル回路の中を示したものである．

グローバル回路（図 4）の構成は大きく分けて制御器，各 SOM モジュール回路の 2 つである．制御器から出力される信号によってその他の回路が動作し，回路間で信号のやりとりが行われる．

各モジュール回路はローカル回路から出力される $e_{i,j}^{k,l_{i,j}^{k,*}}$ を貯めておくメモリ回路とモジュール間での勝者を決める WTA 回路，そして各ローカルユニットからなる．

各ローカルユニット回路は SOM モジュール内の 1 つの SOM ユニットに相当する．それぞれ参照ベクトルブロック，距離演算ブロック，WTA ブロック，2 つの入力データ加算ブロック，2 つの近傍関数値加算ブロック，マルチプレクサ，除算器からなる．このうち参照ベクトルブロックと入力データ加算ブロックは入力データベクトルの次元数のメモリ回路となっている．

パフォーマンス

このハードウェアアーキテクチャとアルゴリズムにより SOM^2 を実行する場合，一つの入力データベクトルを学習する学習モードに必要なクロックサイクル数 Clock を求める．ローカルユニット回路の距離演算ブロック，WTA ブロック，入力データ加算と近傍

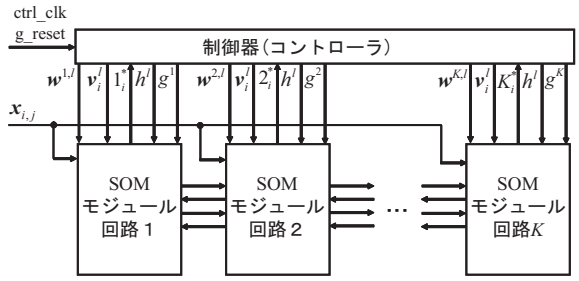


図 4: SOM^2 のグローバル回路

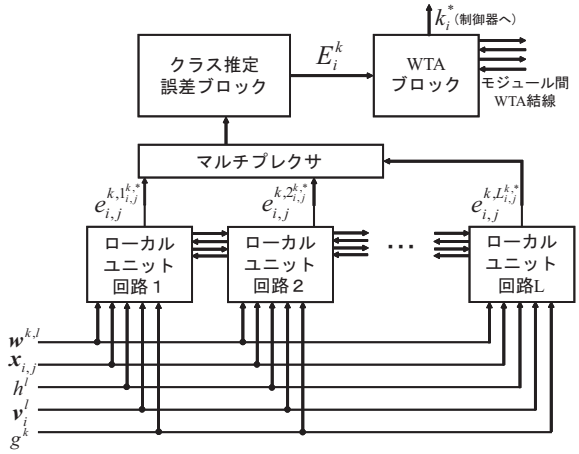


図 5: SOM^2 の SOM モジュール回路

関数値加算ブロック，SOM モジュール回路の WTA ブロック，$\boldsymbol{S}_i^l$ と N^l の除算器，入力データ加算と近傍関数値加算ブロック，$\boldsymbol{SS}^{k,l}$ と $NN^{k,l}$ の除算器，初期化に必要なクロックサイクル数 (C_{Dist}，C_{WTA1}，C_{S1}，C_{WTA2}，C_{DIV1}，C_{S2}，C_{Update}，C_{Init}) を用いて次のように求められる．

$$\begin{aligned}\mathrm{Clock} &= C_{\mathrm{Dist}} + C_{\mathrm{WTA1}} + C_{\mathrm{S1}} \\ &+ \frac{C_{\mathrm{WTA2}} + C_{\mathrm{Div1}} + C_{\mathrm{S2}}}{j} + \frac{C_{\mathrm{Update}} + C_{\mathrm{Init}}}{i \times j} \\ &= m + log_2(m \cdot 2^a) + m \\ &+ \frac{log_2(j \cdot m \cdot 2^a) + log_2(j \cdot 2^a) + m}{j} \\ &+ \frac{log_2(j \cdot 2^a) + 6}{i \times j}\end{aligned} \tag{21}$$

ここで i は入力データのクラス数，j は 1 クラスあたりの入力データベクトルの数，m は入力データベクトルの次元数，a は演算の精度 (bit) を表している．したがって，$i = 5$，$j = 16$，$m = 4$，$a = 8$ とすると，1 入力データの学習あたりに必要なクロックサイクル数は約 19 となる．

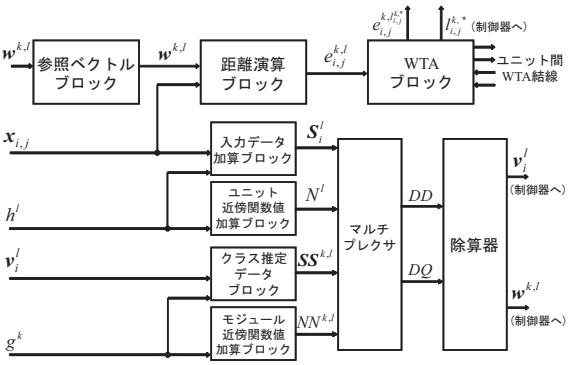


図 6: SOM^2 のローカルユニット回路

また，実行モードに必要なクロックサイクル数は以下の式で求められる．

$$\begin{aligned}Clock &= i\{j\,(C_{Dist1} + C_{WTA1}) + C_{WTA2}\} \\ &= i\{j\,(m + \log_2(m \cdot 2^a)) + \log_2(j \cdot m \cdot 2^a)\}\end{aligned} \tag{22}$$

4 シミュレーション

BL-SOM と SOM^2 それぞれについて，変更前後のアルゴリズムで学習した結果を示す．ここでは計算精度の違いが学習精度にどのような影響を与えるか比較検証する．

4.1 Batch-Learning SOM

入力データとして 2 次元平面上に等間隔に置かれた 256 点の座標を用意し，8 × 8 の 2 次元 BL-SOM で学習を行った．シミュレーション結果を図 7，8 に示す．

図 7 が通常の BL-SOM，図 8 がアルゴリズム変更後のハードウェア型 BL-SOM のシミュレーション結果である．図中の×印の点が入力データベクトルを，・印の点が SOM の参照ベクトルを，・印の点を線で結んだものが SOM が表す特徴空間である．

ハードウェア型の方では，SOM の特徴空間が少しひずんでいる所がある．これは近傍関数を矩形関数としたことでガウシアン関数の場合ほど滑らかに学習率の分配が行われないことが原因と考えられる．このひずみを除けばほぼ等価な結果が得られたと言える．

4.2 SOM^2

3 次元空間上に 2 次関数を 36 点でプロットする．これを 1 クラスとして同様のクラスを 5 つ用意して入力データとする．SOM^2 の構造は，SOM モジュール

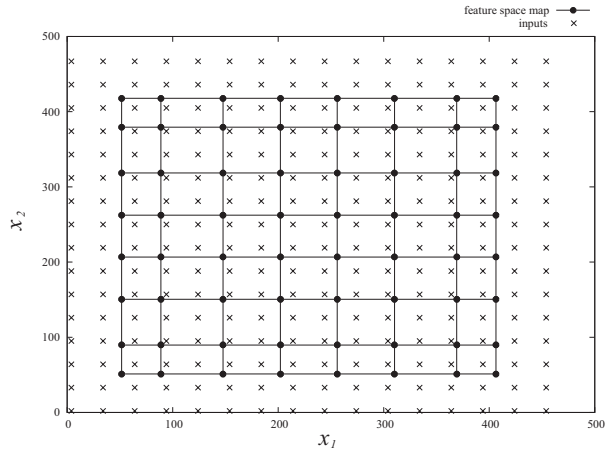


図 7: ソフトウェアアルゴリズムのシミュレーション

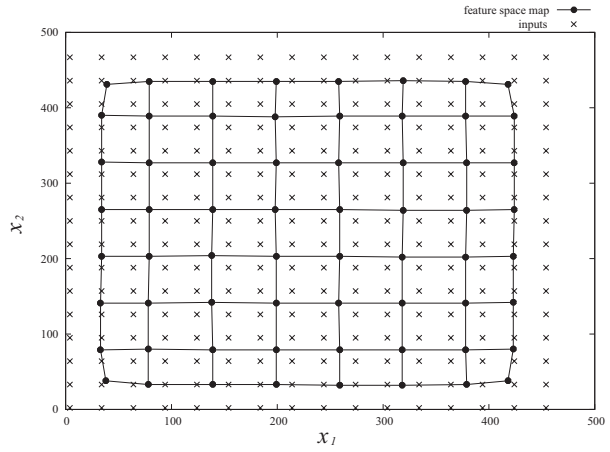


図 8: ハードウェアアルゴリズムのシミュレーション

が 9 個で 1 × 9 の 1 次元配置とし，一つの SOM モジュールは 1 × 9 の 1 次元配置 SOM となっている．シミュレーション結果を図 9，10 に示す．

図 9 が通常の SOM^2，図 10 がアルゴリズム変更後のハードウェア型 SOM^2 のシミュレーション結果である．図中の点線が入力データとして与えた 2 次関数のデータ分布表しており，・印の点が SOM モジュールの参照ベクトルを，・印の点を線で結んだものが 1 つの SOM モジュールが表す特徴空間である．この課題では入力クラス数より SOM モジュール数が多いため，SOM モジュールがクラス間を内挿していることが確認できる．またハードウェア型の SOM^2 では端にあるクラスに SOM モジュールの特徴空間が重なっている点が通常の SOM^2 と異なる．これは近傍関数が矩形関数で近傍の最小半径を 1 としたため通常の SOM^2 と違い他のクラスの学習分配率が 0 になったためである．この点を除けばほぼ等価な結果が得られたと言える．

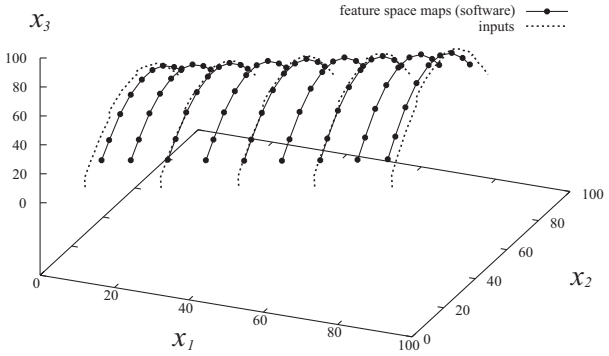


図 9: ソフトウェアアルゴリズムのシミュレーション

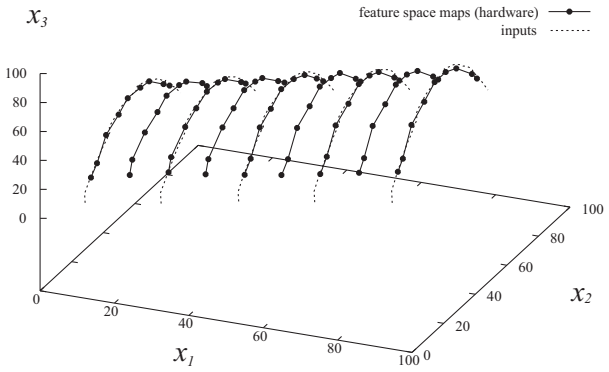


図 10: ハードウェアアルゴリズムのシミュレーション

5　おわりに

本稿では BL-SOM，SOM^2 のハードウェアを実現するため，学習アルゴリズムの変更と回路設計を行い，シミュレーションで動作を確認した．今回のシミュレーション結果から，アルゴリズムとハードウェアアーキテクチャが従来の BL-SOM，SOM^2 と遜色のない結果を出すことがわかった．

今後の課題としてパイプライン処理の実現や実データを扱った課題，そのための回路規模の拡大が挙げられる．

参考文献

[1] T. Furukawa, “SOM of SOMs : An Extension of SOM from ’Map’ to ’Homotopy’,” Lecture Notes in Computer Science, Vol.4232, pp.950-957, 2006

[2] 古川徹生, “SOM の SOM：自己組織化マップ集合をマップする自己組織化マップ,” 第 7 回自己組織化マップ研究会 2006 講演論文集, pp.1-6, 2006.3.

[3] T. Furukawa, “SOM of SOMs : Self-organizing map which maps a group of self-organizing maps,” Lecture Notes in Computer Science, Vol.3696, pp.391-396, 2005.9.