社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－ARC－160 (14)
2004／12／2

MMPDCLアルゴリズムに基づくコードブック生成専用プロセッサ

高木 知陽†, 佐野 健太郎†, 鈴木 健一†, 中村 維男†

近年，画像，動画，音声，ボリュームデータ等のデータの効率的な保存や転送を実現する上で，高効率的なデータの圧縮を実現するベクトル量子化が注目されている．これまで，誤差の少ない量子化のための最適コードブックを生成する様々な手法が提案されてきた．しかし，最適コードブックを生成するための膨大な計算時間が実用化の際に障害となっている．本報告では，誤差を最小とするベクトル量子化のためのコードブックをMMPDCLアルゴリズムに基づき生成する専用プロセッサを提案する．本報告では，ソフトウェアシミュレーションとFPGAを用いた試作により，提案するプロセッサの性能評価を行う．

A Processor Dedicated to Codebook Design based on MMPDCL Algorithm

Chiaki TAKAGI†, Kentaro SANO†, Ken-ichi SUZUKI†, Tadao NAKAMURA†

Long computational time for optimal codebook design has limited its practical use for vector quantization applications. This paper presents a dedicated processor to high-speed codebook design based on MMPDCL algorithm. This processor is based on systolic memory architecture to exploit parallelism in competitive learning. We evaluate the performance of the proposed processor through simulation results and FPGA-based prototype implementation.

1 はじめに

近年，画像，動画，音声，ボリュームデータ等のデータの効率的な保存や伝送を実現する上で，情報損失を抑えながら効率的にデータ圧縮を実現するベクトル量子化が注目されている．ベクトル量子化によるデータ圧縮は，JPEG圧縮方式等に用いられているスカラ量子化を用いた圧縮手法と比較して理論的に高圧縮時の劣化が少ないと言われている．しかし，ベクトル量子化により誤差の小さな圧縮を行うには圧縮時の歪みが最小となる最適コードブックが必要であり，これを求めるための膨大な計算時間が実用化の障害となっている．

この問題を解決するために，これまで，ベクトル量子化のための専用プロセッサが幾つか提案されている[1]-[3]．Tsangらによって提案されたベクトル量子化のためのプロセッサ[1]は，4分木を用いて階層的にコードブックを生成する．このプロセッサでは，ベクトル間の距離を計算する演算器をパイプライン化することにより，高スループットを目指している．しかし，演算回路から独立したコードブックメモリ，及びインデックスメモリとのデータ入出力が，性能向上のボトルネックとなる．

一方，Davidsonらは，ベクトル量子化のためのシストリックアーキテクチャを提案した[4]．このアーキテクチャでは，シストリックアレイにより最近傍探索を行う．この方法は高い並列性を活かした演算が可能となり処理性能の点で非常に優れた方法である．しかしながら，これまで提案されてきたこれらの専用プロセッサの多くは，局所的な最適解に陥りやすいコホネン競合学習に基づいている．このため，これまでと同様に高い並列性を維持しつつ，最適なコードブックを生成するためにコホネン競合学習アルゴリズムより優れたアルゴリズムに基づいた専用プロセッサが必要とされている．

本報告では，量子化誤差を最小とするコードブックを高速に生成するために，部分歪み定理を利用したMinimax Partial Distortion CompeititiveLearning(MMPDCL)アルゴリズム[5]に基づいたコードブック生成専用プロセッサを提案する．提案するプロセッサは，シストリックアレイの構成要素に局所的記憶要素を持たせたシストリックメモリアーキテクチャに基づいており，コードブックを記憶したシストリックメモリアレイにおいて，局所メモリの広帯域幅を利用しながら高速に競合計算を行う．本研究では，ベクトル量子化の応用例として画像圧縮を取り上げる．ソフトウェアシミュレーションとFPGAを用いた試作により，提案するプロセッサの処理性能と，生成したコードブックの量子化誤差について評価を行う．

本稿の構成は以下の通りである．2節では，コホネン競合学習アルゴリズムの問題点とMMPDCLアルゴリズムについて述べる．3節では，MMPDCLによるコードブック生成専用プロセッサを提案する．4節では，ソフトウェアシミュレーションとFPGAを用いた試作により，提案プロセッサの性能評価を行う．5節は結論である．

2 ベクトル量子化のためのコードブック生成

2.1 最適コードブック生成

ベクトル量子化では，k次元のユークリッド空間$\boldsymbol{R}^k$中のベクトルを有限個のコードワードによって近似する．コードワードの集合をコードブックという．ベクトル量子化器は，コードブック$\boldsymbol{Y}=\{\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_N\}$と，それぞれのコードベクトル$\boldsymbol{y}_i$に対応した領域$S_i$の集合$\boldsymbol{S}=\{S_1,S_2,\ldots,S_N\}$により定義される．ベクトル量子化器$Q(\boldsymbol{x})$は，次式に示すように，領域$S_i$に含まれる入力ベクトル$\boldsymbol{x}$をコードワード$\boldsymbol{y}_i$によって近似する．

$$Q(\boldsymbol{x})=\boldsymbol{y}_i \quad \text{if} \quad \boldsymbol{x}\in S_i \quad \text{for} \quad i=1,2,3,\ldots,N \tag{1}$$

ここで$\boldsymbol{x}=\{x_1,x_2,\ldots,x_k\}\in\boldsymbol{R}^k$は，$k$次元の入力ベクトルである．$\boldsymbol{x}$を$\boldsymbol{y}_i$により置き換えたことによる歪み$d(\boldsymbol{x},\boldsymbol{y}_i)$は，通常，ユークリッド距離として次式により与えられる．

$$d(\boldsymbol{x},\boldsymbol{y}_i)=\|\boldsymbol{x}-\boldsymbol{y}_i\| \tag{2}$$

$p(\boldsymbol{x})$を入力ベクトルの確率密度関数とすると，ベクトル量子化結果の平均2乗誤差(MSE)は，次式により

†東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University

定義される．

$$\begin{aligned} MSE(Q) &= \sum_{i=1}^{N} \int_{S_i} d^2(\boldsymbol{x}, \boldsymbol{y}_i) p(\boldsymbol{x}) d\boldsymbol{x} \\ &= \sum_{i=1}^{N} E[d^2(\boldsymbol{x}, \boldsymbol{y}_i) \mid \boldsymbol{x} \in S_i] \cdot p(\boldsymbol{x} \in S_i) \\ &= \sum_{i=1}^{N} pd_i \end{aligned} \tag{3}$$

ここで $E[\cdot]$ は期待値を表し，pd_i はコードワード $\boldsymbol{y}_i$ に対応する領域 S_i の部分歪みである．

最適なベクトル量子化を行うためには，MSE が最小となるような領域 $\boldsymbol{S}$ とコードワード $\boldsymbol{Y}$ を決定する必要があるが，多くの応用分野では確率密度関数 $p(\boldsymbol{x})$ が不明である．この場合，与えられた入力ベクトルに基づき，MSE が最小となるようなコードワードの配置を決定する必要がある．これまで，与えられた入力ベクトルを用いた競合学習を行う様々なベクトル量子化手法が提案されている [5] –[10].

2.2 コホネン競合学習アルゴリズムとその問題点

コホネン競合学習アルゴリズムは，コードブック生成のための代表的な手法である [6]．コホネン競合学習では，個々の入力ベクトルに対して，コードブックの中から最近傍のコードワードを探索する．次に，探索により求まった最近傍コードワードを入力ベクトル方向に近づける．コードワードの移動は，移動後のコードワードを $\boldsymbol{y}_i'$，学習率係数を α とすると，次式により表される．

$$\boldsymbol{y}_i' = \boldsymbol{y}_i + \alpha(\boldsymbol{x} - \boldsymbol{y}_i) \tag{4}$$

この最近傍コードワードの探索，及び位置更新からなる一連の処理を学習ステップと呼び，入力ベクトルに対する最近傍コードワードを競合の勝者と呼ぶ．この学習ステップを繰り返すことにより，コードワードの配置は入力ベクトルの分布を反映したものに変化する．

コホネン競合学習アルゴリズムでは，初期コードブックが適切でない場合，利用されないコードワードが存在し続け，MSE の増加につながることが問題となっている．そこで，より小さな MSE を持つコードブックを生成するために部分歪み定理を利用した Minimax Partial Distortion Competitive Learning(MMPDCL) が提案されている [5].

2.3 MMPDCL アルゴリズム

部分歪み定理によると，最適なベクトル量子化を行うためには，十分に大きなコードブックサイズにおいて，各分割領域 S_i の部分歪み pd_i を等しくする必要がある．このために，MMPDCL では，次式により表されるような，部分歪みの重み付き距離を用いて最近傍探索を行う．

$$d_M(\boldsymbol{x}, \boldsymbol{y}_i) = pd_i(t-1)e^{-m/T} + \|\boldsymbol{x} - \boldsymbol{y}_i\|^2 \tag{5}$$

ここで t は入力ベクトルの入力時の時刻，m は sweep 回数，また $T()0)$ は実験的に定まる定数である．sweep とは，入力ベクトルセットを使った競合学習の 1 周期である．

この重み付き距離に基づき，コホネン競合学習アルゴリズムと同様に勝者 $\boldsymbol{y}_i^*$ を決定する．勝者は次のように更新される．

$$\boldsymbol{y}_{i^*} := \boldsymbol{y}_{i^*} + \alpha_{i^*}(t)(\boldsymbol{x}_k - \boldsymbol{y}_{i^*}) \tag{6}$$

学習率 $\alpha_{i^*}(t)$ は，$1/u_{i^*}$ である．u_{i^*} はコードワード $\boldsymbol{y}_{i^*}$ が時刻 t までに勝った回数である．ただし，u_{i^*} は Sweep 回数 m が 2^I と等しくなったとき，$2+(int)(m/10)$ にリセットされる．ここで I は任意の整数である．勝者の更新後，部分歪みが次の近似式により更新される．

$$pd_{i^*}(t) = pd_{i^*}(t-1) + \|\boldsymbol{x}_k - \boldsymbol{y}_{i^*}\|^2 \tag{7}$$

MMPDCL アルゴリズムでは，式 (5) に基づいて距離に部分歪みを加えることにより，初期の学習ステップでは部分歪みが均等かつ最小化されるようにコードブックが変化する．学習が進むにつれて，部分歪みの効果は弱められ，最終的に最適なコードブック生成を実現するためにコホネン競合学習が行なわれる．

本報告では，MMPDCL アルゴリズムに基づきコードブックを生成する専用プロセッサを提案する．高速なコードブック生成を実現するために，MMPDCL アルゴリズムにおける並列性の抽出が重要である．MMPDCL アルゴリズムでは，入力ベクトルに対する最近傍コードワード，及び最近傍コードワードの部分歪みの更新が完了するまで，次の入力ベクトルに対する最近傍探索を行わない．このため，異なる入力ベクトルの競合学習間で並列に処理ができない．本研究では，異なる入力ベクトルに対する競合学習の並列性を高めるために MMPDCL アルゴリズムの一部に変更を行った上で，競合学習そのものに内在する高い並列性を活かし高速にコードブックを生成可能な専用プロセッサアーキテクチャを提案する．

3 MMPDCL プロセッサ

3.1 シストリックメモリアーキテクチャ

本研究では，競合学習の高い並列性を利用するために，シストリックメモリアーキテクチャに基づく MMPDCL プロセッサを提案する．シストリックメモリアーキテクチャは，シストリックアレイによる規則的な超並列演算と，機能メモリとして各演算回路に付加された局所的記憶要素による広帯域のデータ入出力を特長とする．また，これらに加えてシストリックアレイには，処理要素間のデータの流れが規則的かつ局所的であり大域的な構造が不必要であるという利点がある．

シストリックアレイに入力される入力データは，処理要素を通過しながら並列に処理されるため，アレイの入出力に対して多数の演算要素を稼働させることができる．また，処理要素自身がデータを保持する機能メモリとして実装されており，広帯域のデータ参照が可能となる．この特長は，競合学習の並列性を引き出す上で，非常に適している．従来のメモリと処理ユニットが分離したアーキテクチャでは，バンド幅に限界があり，十分な性能向上は得られない．一方，大域的な構造を持つアーキテクチャは，将来のディープサブミクロン時代の CMOS テクノロジにおいて無視できない配線遅延の影響が大きくこれが性能向上の妨げとなるが，シストリックメモリアーキテクチャでは，このような問題が緩和されると考えられる．

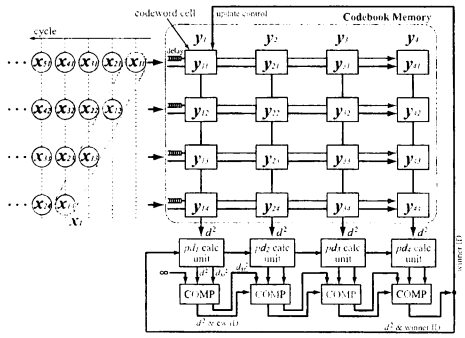

図 1: シストリックメモリアーキテクチャに基づくコードブック生成プロセッサ

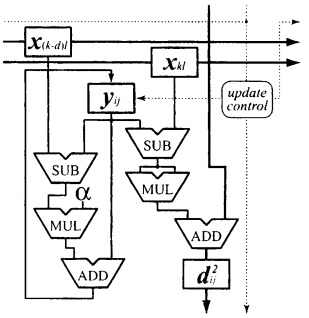

図 2: コードワードセル

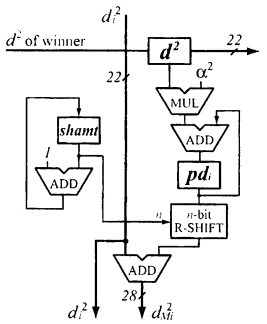

図 3: 部分歪み演算ユニット

3.2 MMPDCL プロセッサ

前節での考察を基に MMPDCL によるコードブック生成専用プロセッサを設計する．図 1 に MMPDCL プロセッサの概念図を示す．本プロセッサは，競合学習における距離計算の高い並列性を利用し，また，データ入出力のボトルネックを排除するために，コードワードの要素を格納する演算ユニットをシストリックアレイ状に配置した構造を持つ．各列の末端には，2.3 節で述べた部分歪みによる重み付き距離計算のための部分歪み演算ユニットを接続する．さらに部分歪み演算ユニットの出力には，最近傍コードワード探索のための比較ユニットを接続する．MMPDCL プロセッサは，このような構成によりコードワードと部分歪みの更新と競合学習を並列に行いながら MMPDCL に基づく競合学習を実現する．次節ではシストリックメモリアレイの要素であるコードワードセルの動作，MMPDCL の実装のために追加した部分歪み演算ユニット，及び比較ユニットについて説明する．

3.2.1 コードワードセル

コードワードセルの構成を図 2 に示す．コードブック生成専用プロセッサは，式 (2) のユークリッド距離を並列に計算するために図 1 のシストリックアレイによりパイプライン処理を行なう．入力ベクトルの各要素はクロックサイクル毎に遅らせながら各行に入力される．シストリックアレイの行方向には，入力ベクトルの要素がクロックサイクル毎に伝搬する．各コードワードセルでは，自分の持つコードワードの要素と伝搬した入力ベクトルの要素との差の二乗を計算する．列方向には，この計算結果が伝搬し，各セルの計算結果が次々と加算されて，最終的に列の末端からは入力ベクトルとその列のコードワードとの二乗距離が出力される．このようにシストリックアレイ中の全コードワードセルを常に稼動させながら，このように，絶え間なく距離計算を行うことが可能である．

3.2.2 部分歪み演算ユニット

部分歪み演算ユニットの構成を図 3 に示す．この部分歪み演算ユニットでは，各コードワードの部分歪みの計算，及び更新を行う．また，2.3 節で述べたように，シストリックアレイから入力される二乗距離に重みとしてそれぞれの部分歪み演算ユニットが格納する部分歪みを加算し重み付き距離を計算し，比較ユニットへ出力する．

ここで，シストリックアレイから入力される二乗距離を d_i^2．比較ユニットへ出力される重み付き二乗距離を $d_{M_i}{}^2$ とする．また，この演算器が格納するコードワードの部分歪みを pd_i とすると，部分歪み演算ユニットは，式 (5) の演算を行い重み付き距離を比較ユニットに出力する．

式 (5) における $e^{-m/T}$ は，時間共に pd_i の値を指数関数的に減衰させるための項である．この指数関数を実現するためのハードウェア量は膨大なものとなる．そこで，本研究では．右算術ビットシフトにより pd_i 固定小数点値を時間と共に減衰させ，同様の効果を実現する．

3.2.3 比較ユニット

比較ユニットの構成を図 4 に示す．比較ユニットは，部分歪み演算ユニットから入力された重み付き距離を比較し，最近傍コードワード探索を行う．MMPDCL アルゴリズムに基づく競合学習では，コードワードの更新のための最近傍のコードワードのインデックス，及び，部分歪みの更新のための入力ベクトルと最近傍コードベクトルの間の二乗距離 $d_{M_i}{}^2$ が必要である．ここで，前段からの重み付き距離の入力を $d_{M_{i-1}}{}^2$，上段の部分歪み演算ユニットから入力される重み付き距離

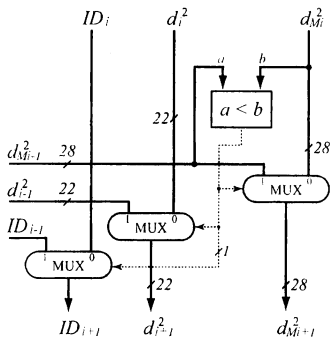

図 4: 比較ユニット

を $d_{M_i}^2$ とする．比較ユニットでは，これらの2つの値を比較し，次段の比較ユニットに $d_{M_i}^2$ の小さい方のコードワードインデックス ID_i と共に二乗距離 d_i^2 を出力する．

このようにして求めた最近傍コードワードのインデックス $ID_{i'}$，及び最近傍コードワード間の二乗距離 $d_{M_{i'}}^2$ が最も右の比較ユニットから出力される．最上段かつ最左端のコードワードセルには最近傍コードワードインデックス ID_i が，最左端の部分歪み演算ユニットには最近傍コードワードインデックス ID_i と二乗距離がそれぞれ入力され，コードワードと部分歪みの更新に用いられる．

3.2.4 遅延更新

提案する MMPDCL プロセッサのシストリックメモリアレイにおいて競合計算を絶え間なく行うために，本研究では，コードワードと部分歪みの遅延更新を導入する．これにより，コードワード，及び，部分歪みの更新完了を待たずに，次の入力ベクトルに対する最近傍探索を並行して行うことが可能となる．

まず，コードワードの遅延更新について述べる．コードワードの更新は，勝者が決定された後，勝者インデックスを最上段かつ最左端のコードワードセルから各セルに伝搬させて行う．勝者インデックスが各コードワードセルのインデックスと一致した時のみ，そのセルにおいてコードワードの要素の更新が行われる．

各コードワードの更新の際には，インデックスと同時に入力ベクトルの値も各セルに伝搬しておかなければならない．そこで，各行に入力された入力ベクトルの要素を，ある一定の遅延を持たせて伝搬させることとする．ここで n をベクトルの次元数，m を列数とする．入力ベクトル $\boldsymbol{x}_1$ を入力してから $n+m$ クロックサイクル後にも，$\boldsymbol{x}_1$ がコードワードセル y_{11} から行方向に伝搬すると，比較器から y_{11} へ出力された勝者インデックスと同じタイミングで更新に必要なデータが揃うこととなる．更新と同時に次の入力ベクトルも各行に入力され，連続して最近傍探索が行われる．

本研究では，以上のように入力ベクトルの入力から数サイクル遅れた更新を，遅延更新と呼ぶ．提案する MMPDCL プロセッサでは，遅延は $n+m+1$ サイクルである．提案する MMPDCL プロセッサでは，遅延更新により，$n+m$ 個の入力ベクトルの処理と，$n+m+1$ クロックサイクル前のコーベワードの更新を並行に行う事が可能である．

次に，部分歪み pd_i の遅延更新について述べる．部分歪みの更新は，勝者が決定された後，勝者インデックスと二乗距離を最左端の部分歪み演算ユニットから各ユニットに伝搬させて行う．勝者インデックスが各部分歪み演算ユニットのインデックスと一致した時のみ，そのユニットの部分歪み pd_i が式 (7) により更新される．ただし，式 (6) より，

$$\|\boldsymbol{x}_k - \boldsymbol{y}_{i\cdot}\|^2 = (1-\alpha)^2 d_{i\cdot}^2. \tag{8}$$

である．部分歪みの遅延更新も，コードワードの更新と同様に $n+m+1$ クロックサイクル遅延して更新される．

4 性能評価

4.1 ソフトウェアシミュレーションによる量子化誤差の評価

本報告で提案する MMPDCL プロセッサでは，演算の高速化と，ハードウェア資源の有効利用を考慮して，固定小数点演算を用いる．現段階では，画像圧縮等への応用を考慮した演算精度を実現するために，コードワード更新においては，整数部 8bit，小数部 4bit，距離計算においては，整数部 18bit，小数部 4bit．また，部分歪みの計算においては，整数部 24bit，小数部 4bit の固定小数点演算を用いている．一般に，固定小数点演算は浮動小数点演算に比べダイナミックレンジが狭く，丸め誤差が発生しやすい．また，遅延更新は，元の MMPDCL アルゴリズムと異なる競合結果を与える可能性がある．以上の理由から，遅延更新と固定小数点演算の導入により量子化誤差が増大することが考えられる．このような量子化誤差増大を調査するため，ソフトウェアシミュレーションにより，画像圧縮のためのコードブック生成実験を行った．実験では，図 5 に示す 256×256 画素の Lenna 画像を 4×4 画素のブロックに分割して得られた 4096 個の 16 次元ベクトルを入力ベクトルとして用いた．また，コードワード数を 512 とした．以下，次の 3 つのアルゴリズムにより生成したコードブックの MSE を比較する．

1. コネン競合学習アルゴリズム（浮動小数点，学習率係数固定）
2. MMPDCL アルゴリズム (浮動小数点，式 (5) の減衰項，式 (6) の学習率係数)
3. MMPDCL (遅延更新，固定小数点演算，学習率係数固定，算術ビットシフトによる減衰項)

遅延更新では，入力ベクトルが入力されてから 529 サイクル後に更新が行なわれる．固定の学習率係数として，0.125 を用いた．アルゴリズム 3 は，提案する MMPDCL プロセッサによるコードブック生成に対応する．Sweep 数を 256 とした競合学習によりコードブックを生成した．図 6 に，10 回のコードブック生成結果の平均による sweep 数と MSE の関係を示す．

MMPDCL アルゴリズムは，コホネンの競合学習に比べて最終 MSE が低く，優れたコードブックを生成している．また，アルゴリズム 2 と 3 の結果を比較すると，MMPDCL プロセッサを模擬したコードブック生成では，MSE の収束は 10sweep 程遅れるものの，この実験条件では最終 MSE に大きな違いが無いことが確認された．

図 5: Lenna 標準画像

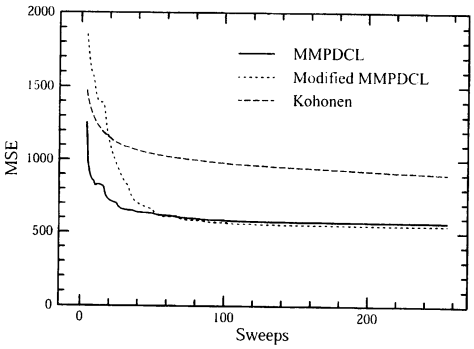

図 6: Sweep 数と MSE (512 コードワード)

4.2 FPGA を用いた MMPDCL プロセッサの実装

4.2.1 FPGA プロトタイプボード

DiNI Group 社の FPGA プロトタイプボード DN5000K10S を用いて提案プロセッサの試作を行った．このボードには，Altera 社の FPGA Stratix EP1S80 が搭載されており，およそ 60 万ゲート規模の回路を構成することができる．Stratix EP1S80 には 64KB の SRAM が 9 個内蔵されている．本試作では，この SRAM をアレイに流す入力ベクトルのためのメモリとして使用した．

本ボードは PCI バスに接続が可能である．本研究では，MMPDCL プロセッサ，入力ベクトルメモリ，及びそれらの制御回路を PCI ターゲットデバイスとして実装した．

4.2.2 コードブック生成システム

図 7 に試作を行なったコードブック生成システムの概要を示す．試作システムは，MMPDCL プロセッサ，入力ベクトルメモリ，制御回路とそのためのレジスタ，PCI コントローラにより構成されている．MMPDCL プロセッサは，16×16 コードワードセルのシストリックメモリアレイと、部分歪み演算ユニットの 1 次元アレイ，及び比較ユニットの 1 次元アレイから成る．入力ベクトルメモリは，2 次元分の要素を格納する 32 ビット幅の SRAM8 個により構成されており，16 次元のベクトルデータを同時に読み出すことが可能である．入

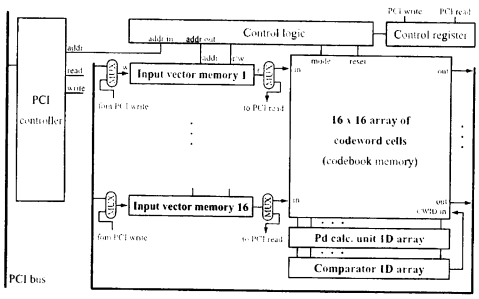

図 7: 試作システムの概要

力ベクトルメモリと制御レジスタは PCI アドレス空間にマッピングされており，PCI バスを通してホスト PC から読み書きができる．制御レジスタの値を書き換えることより，MMPDCL プロセッサの動作を以下の 4 つのモードから選ぶことができる．

1. コードブック書き込みモード
2. 競合学習モード
3. コードブック読み出しモード
4. 無動作モード

起動時には，MMPDCL プロセッサは無動作モードになっている．以下の手順により，MMPDCL プロセッサを用いてコードブック生成を行なう．まず，ホスト PC から入力ベクトルメモリに初期コードブックを書き込む．この後に，MMPDCL プロセッサをコードブック書き込みモードにし，入力ベクトルメモリ中の初期コードブックをシストリックメモリアレイに書き込む．次に，ホスト PC から入力ベクトルメモリに入力ベクトルセットを書き込み，MMPDCL プロセッサを競合学習モードにすることによりコードブック生成を行なう．最後に，コードブック読み出しモードにより入力ベクトルメモリに読み出したコードブックを，ホスト PC に転送する．試作システムは，PCI バスと同じ 33MHz の周波数で動作する．

4.2.3 コードブック生成結果

試作システムにより，画像データ圧縮のためのコードブック生成を行った．4.1 節で述べたソフトウェアシミュレーションと同じ Lenna 画像から 4096 個の 16 次元残差ベクトルを生成し，入力ベクトルとして使用した．ここで，残差ベクトルとは，4×4 の画像ブロックの各画素値から画素値の平均を引いた値を要素としたベクトルである．コードブックのサイズは，シストリックメモリアレイの列数と同じ 16 である．Sweep 数を 192 として競合学習を行なった．

33MHz で動作する試作システムでは，2.4×10^{-2} 秒で上記のコードブック生成を行なうことができる．一方，2GHz で動作する Intel Celeron プロセッサを用いた場合には，MMPDCL アルゴリズムに基づき浮動小数点演算によりコードブック生成を行なうには 9.0 秒が必要であった．浮動小数点演算と固定小数点演算の違い等はあるが，試作システムは汎用プロセッサのおよそ 400 倍の処理速度で同様のコードブック生成が可能であることが確認された．

図 8 に，浮動小数点による MMPDCL アルゴリズム，MMPDCL プロセッサのソフトウェアシミュレー

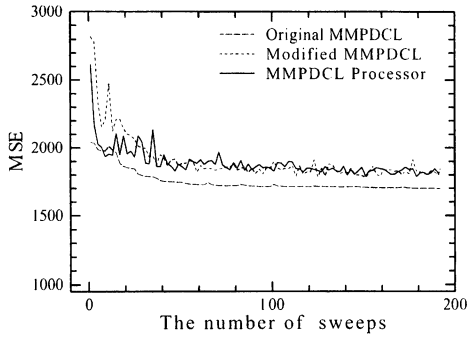

図 8: Sweep 数と MSE (16 コードワード)

図 9: 復元画像 (ソフトウェアシミュレーション, MSE=1845)

ション, 試作システムにより生成されたコードブックの MSE を示す. 浮動小数点による MMPDCL アルゴリズムには及ばないものの, 試作システムはソフトウェアシミュレーションと同等の MSE を有するコードブックを生成可能であった. 図 9, 10 に, それぞれ, ソフトウェアシミュレーションと試作システムにより生成されたコードブックを用いた画像圧縮の復元画像を示す. 同等の MSE を持つこれらの画像には, 視覚的差異は殆んど見られない.

5 おわりに

本報告では, 誤差を最小とするベクトル量子化のためのコードブックを MMPDCL アルゴリズムに基づき生成する専用プロセッサを提案し, ソフトウェアシミュレーションと FPGA を用いた試作により, その性能評価を行った. 遅延更新と固定小数点演算のために元のアルゴリズムと比べて若干の MSE 増加が見られたものの, 提案するプロセッサはほぼ同等のコードブックを生成可能であった. また, 汎用プロセッサにより同様のコードブック生成を行なった場合と比べて, 数百倍の速度向上が得られた. 今後の課題は, さらなる動作検証と, より多くのコードワードを処理するための改良である.

参考文献

[1] Kevin Tsang and Belle W. Y. Wei. A vlsi architecture for a real-time code book generator and encoder of a vector quantizer. *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, 2(3):360–364, September 1994.

[2] K. Kobayashi, N. Nakamura, K. Terada, H. Onodera, , and K. Tamaru. An lsi for low bit-rate image compression using vector quantization. *IEICE Transactions on Electron*, (5):718–724, May 1998.

[3] Toshiyuki Nozawa, Masahiro Konda, Masanori Fujibayashi, Makoto Imai, Koji Kotani, and Shigetoshi Sugawara Tadahiro Ohmi. A parallel vector-quantization processor eliminating redundant calculation for real-time motion picture compression. *IEEE Journal of Solid-State Circuits*, 35(11):1744–1751, November 2000.

[4] Grant A. Davidson, Peter R. Cappello, and Allen Gersho. Systolic architectures for vector quantization. *IEEE transactions on Acoustics, speech and signal processing*, 36(10):1651–1664, October 1988.

[5] Ce Zhu and Lai-Man Po. Minimax partial distortion competitive learning for optimal codebook design. *IEEE Transactions on Image Processing*, 7(10):1400–1409, October 1998.

[6] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin Heidelberg, 1989.

[7] A.Gersho and R.M.Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[8] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, (1):84–95, January 1980.

[9] S.P.Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, IT-28(2):129–137, March 1982.

[10] S.Grosberg. Adaptive pattern classification and universal recoding:i.parallel development and coding of neural feature detectors. *Biological Cybernetics 23*, pages 121–134, March 1976.

図 10: 復元画像 (試作システム, MSE=1826)