■10 群（集積回路）－ 5 編（演算・信号処理 LSI）

3 章 機能ごとの実現法

（執筆者：天野文雄）［2010 年 3 月 受領］

■概要■

本章では，画像符号化・画像認識などの画像信号処理，音声符号化・音声合成などの音声信号処理，グラフィックス技術，誤り訂正技術，MIMO 技術，暗号の各技術について実現法を紹介している．

【本章の構成】

3-1 節では，画像符号化について，JPEG，JPEG 2000，MPEG-2 中核処理，H.264 中核処理，の各実現方法を詳細に説明している．また，併せてレートコントロールの実現方法，及び異なるアーキテクチャによるリアルタイムコーデック LSI のいくつかの構成法とトレンドも紹介している．

3-2 節では，画像処理・画像認識について，処理の特徴，LSI 設計に向けた留意事項，LSI の実現例，今後の展望を述べている．

3-3 節では，音声符号化，音楽符号化，及び音声認識・音声合成，ノイズキャンセル・エコーキャンセル，マイクアレイなどの音声処理技術について，処理の概要と構成上のポイントを述べている．関連してシリコンマイクや省電力化にも触れている．

3-4 節では，グラフィックス技術の動向を紹介している．3D グラフィックスでは大規模なシステムからコモディティ化の方向へ向かっていること，専用 LSI，プログラマブルシェーダ，頂点処理と画素処理の統合へとアーキテクチャが進化していること，最近の NVIDIA 社および AMD 社の GPU の例などが紹介されている．

3-5 通信では，3-5-1 項で誤り訂正符号としてビタビ復号器，ターボ符号の実現方法を，3-5-2 項で MIMO システムとその LSI 化を性能の比較を交えながら紹介している．

最後に，3-6 暗号では，共通鍵暗号技術と公開鍵暗号技術について，それぞれ実装技術の現状を解説している．

■10 群 － 5 編 － 3 章

3-1 画像符号化

本節では静止画符号化の中で，ディジタルカメラなどで幅広く応用されているベースライン JPEG と，高符号化効率を有する JPEG 2000 の実現方法について説明する．静止画符号化 LSI は，要求される演算速度，限られた回路リソースやメモリバンド幅で実現するために，高効率なアルゴリズムやアーキテクチャを実装することが重要である．

3-1-1 JPEG の実現方法

（執筆者：山内英樹）［2009 年 1 月 受領］

JPEG [1] のベースラインシステムでは，8 画素×8 ラインのブロック単位で符号／復号化する．**図 3･1** で示されるように，符号化ではピクセルインタフェースがメモリバスを介して，外部メモリに保存されている原画像データから 8 画素×8 ラインのブロックごとにデータを切り出す．このブロックデータを DCT／逆 DCT 部，量子化／逆量子化部，ハフマン符号化／復号化部が処理しビットストリームに加工する．生成された JPEG ビットストリームは，ビットストリーム I/F 部を介して，外部メモリに保存される．ホストバスに接続されている内部 CPU/DSP 部は，内部バスを介して各ブロックに接続されており，各ブロックの制御，量子化テーブルやハフマンテーブルを設定する．符号量制御は，ビットストリーム I/F 部で生成されたストリームの符号量を計測し，量子化テーブルにフィードバックすることで行う．

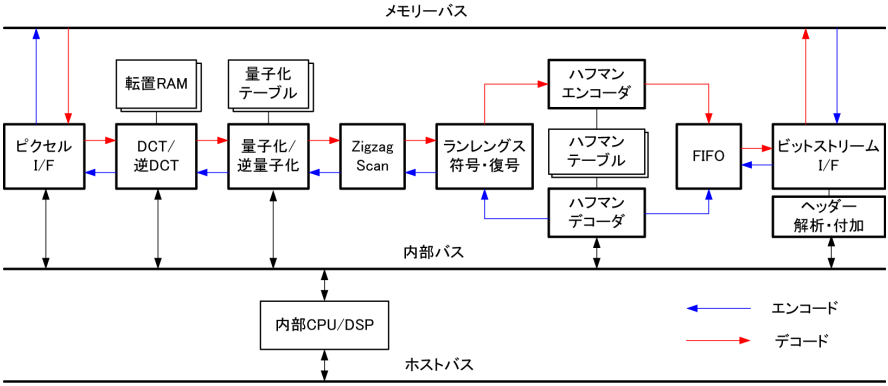


図 3･1　ベースライン JPEG の構成

復号化動作では矢印で示されるように，符号化動作の逆順で処理を行う．ヘッダ解析部では，復号化時にビットストリームに埋め込まれている各種のヘッダの解析する．またデコードしたハフマンテーブルと量子化テーブルは LSI 内のメモリにロードする．

JPEG 符号／復号の高速処理を実現するにはパイプラインアーキテクチャの適用が有効である．ブロック単位で DCT／逆 DCT 部，量子化／逆量子化部，ハフマン符号化／復号化部をパイプライン動作させることで，1 サイクル当たり 1 画素以上の処理が可能である．

(1) 2 次元 DCT/IDCT 回路

2 次元 DCT 係数は，8×8 個の画素値から作られる線形変換であるので，単純に計算すると 8×8 個の DCT 係数を得るのに 4096 回の乗算と加算が必要になる．このような膨大な演算は，回路面積や消費電力の面で不利である．一方，DCT 変換及び逆 DCT 変換は式(1)，(2)で表される．括弧内の演算は 1 次元 DCT であるので，1 次元 DCT を 2 回繰り返すことで 2 次元 DCT が求められることがわかる．この場合，1024 回の乗算と加算で実現できる．

$$\text{DCT変換}：F_{uv}=\frac{1}{4}C_uC_v\left\{\sum_{y=0}^{7}\left\{\sum_{x=0}^{7}f_{xy}\cos\frac{(2x+1)u\pi}{16}\right\}\cos\frac{(2y+1)v\pi}{16}\right\} \tag{1}$$

$$\text{逆DCT変換}：f_{xy}=\frac{1}{4}\left\{\sum_{v=0}^{7}C_v\left\{\sum_{u=0}^{7}C_uF_{uv}\cos\frac{(2x+1)u\pi}{16}\right\}\cos\frac{(2x+1)u\pi}{16}\right\} \tag{2}$$

$$\text{ここで，}C_uC_v=\begin{cases}1/\sqrt{2} & \textit{for } u, v = 0\\ 1 & \textit{for otherwise}\end{cases} \tag{3}$$

更に，1 次元 DCT の 8×8 行列を小行列に分解し，因数分解することで計算量を減らす高速 DCT 演算が適用できる．Chen のアルゴリズム [2] はこの代表的なものであり，乗算が 256 回，加算が 416 回で DCT 演算が実現できる．汎用 CPU や DSP を用いて DCT 演算を行う場合には，この手法が広く用いられる．

・2 次元 DCT 回路の構成

高性能 LSI の実現には，多くの面積を占める 2 次元 DCT 回路の小型化やスループットの向上が重要である．2 組の 1 次元 DCT 演算器とバッファ RAM で実現する 2 次元 DCT 回路において，1 次元 DCT 演算回路を再利用すれば回路が小型化できる．また，1 次元 DCT 演算回路はチェンのアルゴリズムで構成できるが，ASIC などの専用ハードウェアで実装する場合は回路面積の大きい乗算器を，入力の一方がコサイン係数である定数乗算器への置き換えや，定数乗算器を加算器と減算器のみで構成することで，回路規模の小型化と低消費電力が可能になる．これを**図 3･2** の IDCT 回路 [3] を用いて説明する．

直列・並列変換回路に入力された DCT 係数は，8 個分の DCT 係数が揃ってから 4 通りに並べ替えられ，1 次元 IDCT 回路に入力される．演算器では二つの IDCT 係数を同時に出力するので，並べ替えを 4 回繰り返すことで 8 個の入力データに対応した 8 個の 1 次元 IDCT の演算結果が出力される．8 個分の IDCT 係数が転置 RAM に書込まれた後，次の 8 個分の DCT 係数が直列・並列変換回路から 1 次元 IDCT 演算回路に入力される．同様の処理を 8 回繰り返すことで，8×8 個の 1 次元 IDCT 係数が転置 RAM に保存される．次に，並べ替え回路の入力を転地 RAM からの出力に切り替え，2 回目の 1 次元 IDCT 演算を行う．2 次元 IDCT 演算結果は丸め処理回路から出力される．同様な処理を 8 回繰り返さすことで，8×8 画素の 2 次元 IDCT 演算が完了する．この場合，1 ブロックの IDCT 演算を 34 サイクルで処理できる．

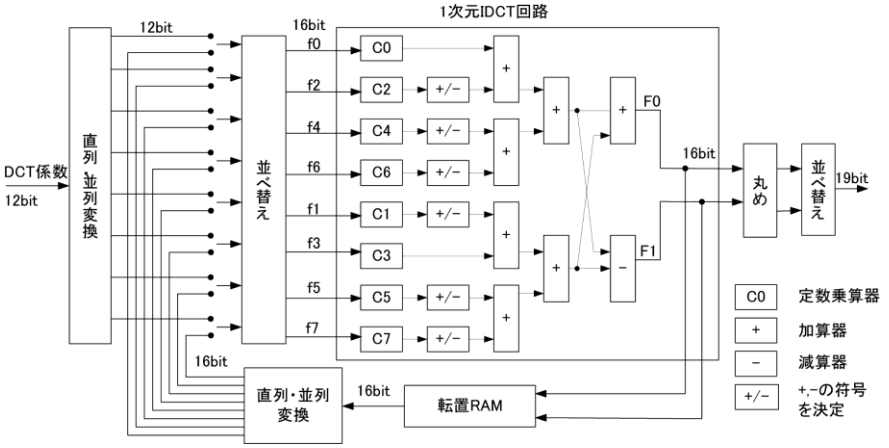


図 3・2　2 次元 DCT アーキテクチャ

(2)　ハフマン符号・復号部

2 次元 DCT 係数は，1 個の DC 成分と残りの 63 個の AC 成分に対して，異なるアルゴリズムでハフマン符号･復号化される. DC 成分の符号化は, 絶対値の大きさによりカテゴリ（SSSS）に分類する．次にハフマンテーブルを用いてカテゴリ番号を符号化し，それに続いてカテゴリから一意に決まるビット長で 2 進数表現した DC 成分を連結する．AC 成分の符号化については，ジグザクスキャン部で 2 次元 DCT 係数を 1 次元データに変換した後，ランレングス符号部で連続する零の個数である RUN 値（RRRR）と非零の係数である LEVEL 値（LLLL）を求める．次に指定されたハフマンテーブルから RRRR と LLLL の組合せに対応する符号が決定される．更にこの符号と LLLL より定まる可変ビット長で 2 進数表現した非ゼロの DCT 係数を連結する．これをすべての DCT 係数に施す.

DC 成分の復号は，ハフマンデコーダでビットストリームを DC 成分のハフマンテーブルと比較する．一致すれば，カテゴリと可変ビット長が求まるので，このビット長のデータを切り出して DC 成分を復号する．AC 成分の復号は，ビットストリームと RRRR/LLLL のハフマンテーブルを比較し RUN/LEVEL の組合せを求める. RUN 値により連続するゼロの個数が求まる．LEVEL 値により非ゼロ係数の可変符号長が求まるので，その可変符号長の AC 係数をビットストリームから切り出して復号する. ランレングス符号部で RUN 値と AC 係数から，AC 係数を再生し IQ 部に出力する．ハフマン復号器の実装はバレルシフタと比較器により構成され，LSI で実現する場合には，1 ビットから 16 ビットの比較器を 16 個並列に実装することで高速処理することができる.

3-1-2 JPEG 2000 の実現方法

（執筆者：山内英樹）［2009 年 1 月 受領］

JPEG 2000[4), 5), 6)] は，JPEG で用いられている DCT やハフマン符号に変わり，DWT（Discreet Wavelet Transformation）と EBCOT（Embedded Block Coding with Optimized Truncation）を用いる．これらの演算は，DCT やハフマン符号化に比べると膨大な演算量や大容量メモリが必要

とされる．したがって，高性能 JPEG 2000 の実現には，高効率なアルゴリズムやアーキテクチャが必要となる．**図 3･3** に示される JPEG 2000 エンコーダでは，メモリに格納されている画素データを 2 次元 DWT 演算器（2D-DWT）で四つのサブバンド（*LL*; *LH*; *HL*; *HH*）に分解し，ウェーブレット係数をメモリに保存する．更に *LL* サブバンドに対しては，指定回数まで 2D-DWT を繰り返す [7]．量子化器（QTZ）では，メモリからコードブロック単位でウェーブレット係数を読み込み量子化する．算術符号化器（EBCOT）では量子化後のウェーブレット係数を符号化し，パケット単位にコードブロックの並べ替えを行う．ヘッダ生成部でコードブロックの情報を表すパケットヘッダを生成し，各パケットに付加し，JPEG 2000 のストリームを形成する．画像サイズが大きい場合には画像全体をタイルに分割し，タイルごとに 2D-DWT を行う場合がある．この場合，タイル境界が不連続になりタイルノイズが発生する場合がある．JPEG 2000 を高速処理するには 2D-DWT と EBCOT をフレームレベルでパイプライン動作させ，EBCOT 符号化の処理中に次のフレームの DWT 処理を行い，高スループットを得る．

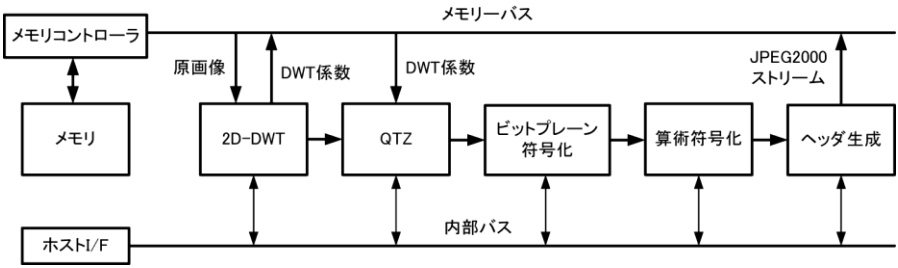


図 3・3　JPEG 2000 エンコード部の構成

(1)　2 次元 DWT 演算

2D-DWT では画像全域に対して, 水平または垂直の 1D-DWT 処理を行った後の途中結果を保存する大容量の転置 RAM が必要となる．また，メモリバンド幅も JPEG と比べて高くなる．これを解決するために，様々な VLSI 向けのアーキテクチャが提案されている．水平と垂直の DWT 演算を同時処理するラインベース変換法 [9] では, **図 3･4** に示す (9-7)フィルタの場合，9 ライン分のラインメモリ，1 個の垂直ウェーブレットフィルタ（V-filter），2 個の水平ウェーブレットフィルタ（H-filter(L)，H-filter(H)）から構成される．V-filter に，垂直方向に連続する 9 画素（p0-p8）を入力し，2 個のウェーブレット係数（L^V, H^V）を得る．同時に L^V は H-filter(L) へ，同様に H^V は H-filter(H) へ入力される．V-filter に入力する 9 画素の位置を水平（列）方向にスキャンしてすべての列について同様の処理を行う．次に，ラインメモリの 2 ラインを新たな行データに更新し，同様な処理をすべての行が処理されるまで行う．H-filter(L) では，L0-L8 を用いて *LL*, *HL* を生成し，H-filter(H) では，H0-H8 を用いて *LH*, *HH* のウェーブレット係数を生成する．ラインベース変換法では画像サイズに比例してラインメモリの容量が大きく，また複数の DWT 演算器が必要となる．

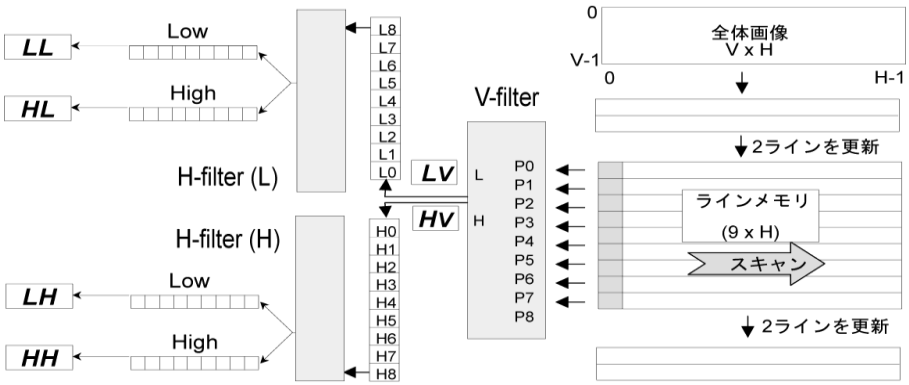


図 3・4　ラインベース 2D-DWT

一方，**図 3･5** のオーバーラップブロックベース DWT [10] では，水平及び垂直方向に，互いに 7 画素分オーバラップする任意の大きさの矩形領域に画像全域を分割し，分割した矩形領域ごとに 2D-DWT を行う．オーバラップはブロック間の連続性を維持し，矩形領域境界でのノイズ発生を抑えるためである．オーバーラップブロックベース DWT は小容量のメモリで 2D-DWT を実現できるが，矩形領域のサイズが小さいほどメモリアクセスが頻繁になる．1 次元 DWT のハードウェアでの実現方法として，systlic アーキテクチャに DWT をマッピングした構造 [12] や，リフティング [8] のデータフローをパイプラインアーキテクチャに適用した構造 [11] などがある．これらについては参考文献を参照されたい．

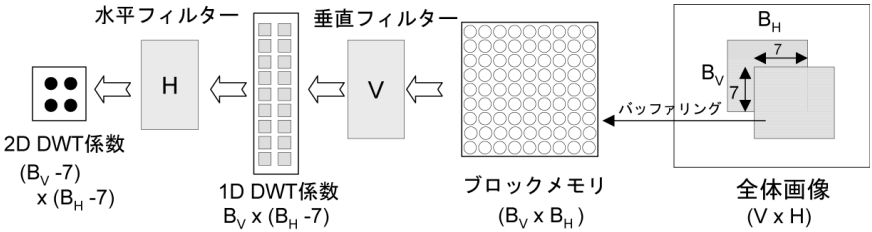


図 3・5　オーバーラップブロックベース DWT

改良オーバーラップブロックベース DWT [13] では DWT 計算時に生じる中間データを保存することで，回路の小型化と計算効率を向上させている．**図 3･6** に示されるように，垂直，水平それぞれのウェーブレットフィルタ演算部と，それに接続されるレジスタアレイから構成される．レジスタアレイのサイズは，垂直方向用が $4\times B_H$，水平方向用が 4×2 である．

- 画像データはブロック単位でブロックメモリにバッファリングされ，ブロックメモリから縦方向に隣り合った 2 個単位でラスタスキャン順に垂直フィルタへ入力される．
  [(0,0),(1,0)] - [(0,1),(1,1)] - [(0,2),(1,2)] ･･･････ [(2*n*,*i*),(2*n*+1,*i*)] ･･･････
- 垂直フィルタでは，2 個の入力画素と，レジスタアレイに格納されている中間データ C_n,

D_n, E_n, F_n を用いて，2 個のウェーブレット係数 W_H, W_L と新たな中間データ C_{n+1}, D_{n+1}, E_{n+1}, F_{n+1} を計算する．W_H と W_L は，H-filter へ出力される．レジスタアレイに格納されている中間データ C_n, D_n, E_n, F_n は，新たな中間データ C_{n+1}, D_{n+1}, E_{n+1}, F_{n+1} で更新される．

- 水平フィルタでは，2 個の W_H と $C_n^H, D_n^H, E_n^H, F_n^H$ から，2 個のウェーブレット係数 W_{HH}, W_{LH} と新たな中間データ $C_{n+1}^H, D_{n+1}^H, E_{n+1}^H, F_{n+1}^H$ を計算する．続いて，2 個の W_L と $C_n^L, D_n^L, E_n^L, F_n^L$ から，2 個のウェーブレット係数 W_{LL}, W_{HL} と新たな中間データ $C_{n+1}^L, D_{n+1}^L, E_{n+1}^L, F_{n+1}^L$ を計算する．レジスタアレイに格納されている中間データは新たな中間データで更新される．

なお，垂直フィルタでは，低周波成分 L と高周波成分 H を同時に生成するが，これらのデータは内部で並び替えられ，L 成分，H 成分単位で 2 個同時に出力する．水平フィルタに入力される垂直成分のウェーブレット係数は DWT 演算の順番となっているため，垂直フィルタと水平フィルタをメモリを介さずに接続できる．

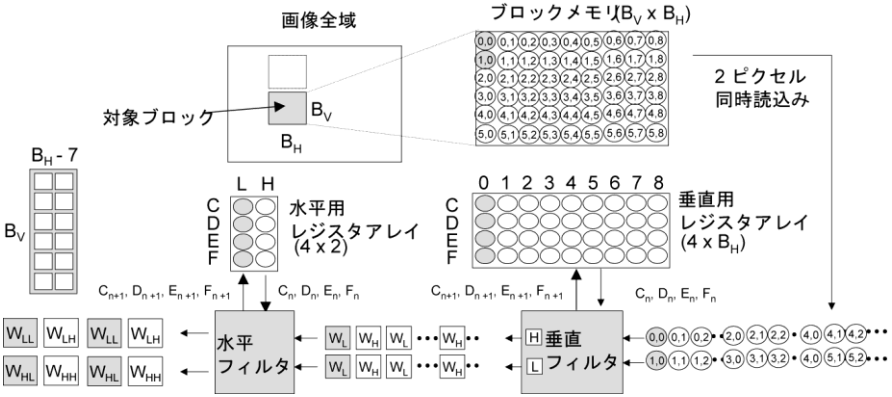


図 3・6　改良オーバーラップブロックベース DWT

(2)　量子化回路

メモリに保存されている DWT 係数は，コードブロック単位で量子化器（QTZ）で量子化される．コードブロックのサイズが大きいと符号化効率は高いが，回路規模や動作速度の面では不利になる．複数サンプルを並列に量子化することでスループット向上させる．1 コードブロック分の量子化が終了後，バッファ RAM に保存され，後段の EBCOT 部で使用される．

(3)　EBCOT 符号化

EBCOT 符号化の処理手順は BPC（ビットプレーン符号化）と算術符号化のプロセスからなる．ビットプレーン符号化では，(1) 量子化された DWT 係数をコードブロックに分割，(2) 各コードブロックをビットプレーンに分解，(3) 各プレーンの解析を行い，3 通りの符号化パ

ス（Significance pass，Refinement pass，Cleanup pass）に分割，という三つの処理により行われる. 算術符号化プロセスでは, それぞれの符号化パスを 2 値算術符号化器 MQC（MQ-Coder）で符号化する．**図 3･7** に示す EBCOT 符号化のアーキテクチャ例では，BP & MQ Coder で構成される二つの EBCOT 回路を並列動作させ，上位プレーンと下位プレーンを同時に処理させる．また，各 BP & MQ Coder は 3 個の符号化パス処理回路の並列化や，MQC を 4 並列で処理することで，高速処理を実現できる．

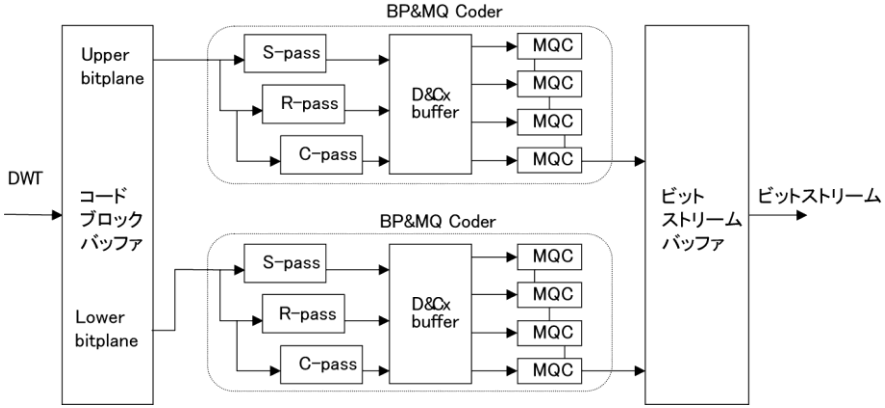


図 3・7　EBCOT のブロック構成

■参考文献

1) ISO/IEC IS 10918-1, “Information Technology - Digital compression and coding of continuous-tone still images: Requirement and guidlines,” ISO/IEC JTC1/SC9/WG10, Feb. 1994.
2) W. H. Chen, C. H. Smith and S. C. Fraick, “Afast computation algorithm for the discrete cosine transform,” IEEE Trans. On Communication, vol.25, no.9, pp.1004-1009, 1977.
3) 河原桂太，山内英樹，岡田茂之，松田　優, “4 倍速の早送り再生ができる MPEG 復号化 LSI を開発,” 日経エレクトロニクス, no.639, pp.107-121, 1995.
4) ISO/IEC IS 15444-1, “Information Technology - JPEG2000 image coding system - Part 1: Core coding system,” ISO/IEC JTC1/SC29/WG1, Dec. 2000.
5) ISO/IEC IS 15444-1, “Information Technology - JPEG2000 image coding system - Part 1: Core coding system,” AMENDMENT 1: Codestream restrictions, Mar. 2002.
6) David S. Taubman and MichaelW. Marcellin, “JPEG2000 Image Compression Fundamentals, Standard and Practice,” KAP, 2002.
7) S. G. Mallat, “A theory for multiresolution signal decompositopn: The wavelet representation,” IEEE Trans. PAMI, vol.11, pp.674-693, 1989.
8) I. Daubechies and W.Sweldens, “Factoring Wavelet Transforms into Lifting Steps,” J. Fourier. Appl., vol.4, pp.247-269, 1998.
9) C. Chrysafis and A. Ortega, “Line-based, reduced memory, wavelet image compression,” IEEE Trans. on Image Processing, vol. 9, issue 3, pp.378-389, Mar. 2000.
10) T. C. Denk and K. K. Parhi, “Calculation of mini-mum number of registers in 2-D discrete wavelet trans-form using lapped block processing,” in Proc. IEEE ISCAS’94, vol. 3, pp.77-80, May 1994.
11) K. Andra, and T. Acharya, “A VLSI Architrcture for Lifting-based Forward and Inverse Wavelet transform,” IEEE Trans. on Signal processing, vol. 50, no. 4, pp.966-977, Apr. 2002.

12) T. Acharya and P. Chen, “VLSI of a DWT Arhiteture,” Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Monterey, CA, May 31-Jun. 3, 1998.
13) H. Yamauchi, et al., “1440×1080 pixel, 30 frames per second motion-JPEG 2000 codec for HD-movie transmission,” IEEE Journal of Solid-State Circuits, vol.40, pp.331-341, 2005.

3-1-3 MPEG-2 中核処理の実現法

（執筆者：笠井良太）［2009 年 11 月 受領］

(1) 処理の概要

MPEG-2 を含む殆どの動画像符号化／復号化の処理は DCT などの直交変換と量子化, 動き補償，エントロピー符号化の組合せで実現しており，その処理は 16 画素×16 画素のマクロブロック（MB）単位で実行される．MPEG-2 の符号化処理(1) を**図 3･8** に示す．既に符号化された画像フレームを参照してそこからの動きを推定して予測信号を生成し（動き補償という），その予測信号と符号化対象の画像との差分データを DCT，量子化（Q）したのち，可変長符号化（VLC）というエントロピー符号化を行っている．復号化処理は逆の処理フローとなり，可変長復号化（VLD），逆 DCT（IDCT），逆量子化（IQ）と動き補償（MC）を経て再生画像を出力する．差分データを正しくデコーダに伝えてきれいな再生画像を得るために，符号化器内で復号化器と同じ再生画像を作って参照画像とする必要があり，IDCT と IQ からなるローカルデコーダが配置される．

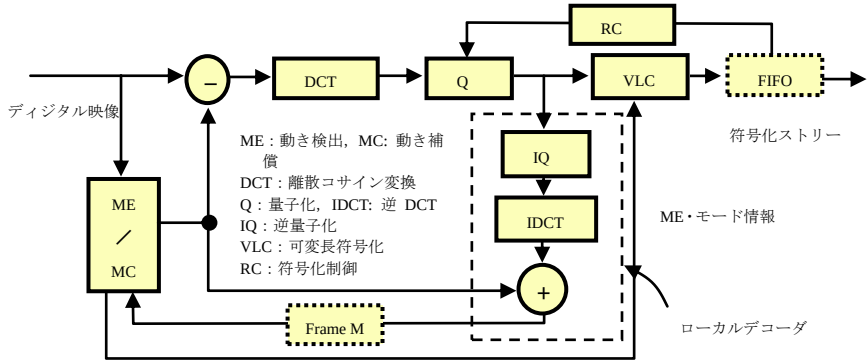


図 3・8 MPEG-2 ビデオ符号化処理フロー

高い圧縮効率を得るためには，現 MB とベストマッチの参照 MB を見つける動きベクトル検出（ME）が必要となる．MPEG-2 では 0.5 画素精度の探索行うことが求められており，全探索法のような力任せのアルゴリズムでは膨大な演算量と ME とフレームメモリ間に大きなメモリバンド幅が必要となる．

(2) ブロックマッチング演算器の構成

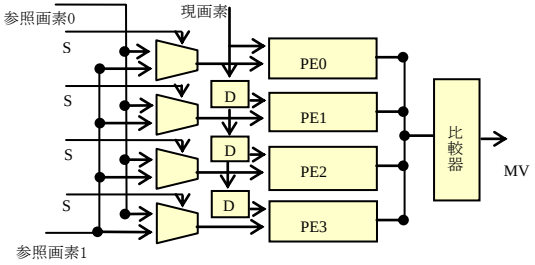
MV 探索（ME）の基本処理は，符号化対象 MB（以下，現 MB という）と参照 MB との間で画素差分絶対値和（SAD : Sum of Absolute Difference）を求めるブロックマッチング（BM）処理である．探索範囲が水平，垂直ともに±P，MB のサイズが N×N のとき，探索ポイント

m, n の SAD 値は次式で表される．

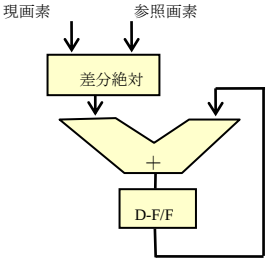
$$SAD(m,n) = \sum_{i=1}^{N} \sum_{j=1}^{N} Distortion(i, j, m, n) \quad -P \leqq m,n < P \tag{1}$$

$$Distortion\ (i,j,m,n) = |cur(i,j) - ref(i+m, j+n)| \tag{2}$$

ここで，$cur\ (i, j)$ と $ref\ (i+m, j+n)$ は現 MB と参照 MB の画素値，(m, n) は探索範囲のひとつの探索候補点，$Distortion\ (i, j, m, n)$ はその候補点における現行画素と参照画素の差分絶対値，$SAD\ (m\ ,n)$ は差分の総和を表す．



（a）1 次元シストリックアレイ構成



（b）PE の回路構成

	PE0		PE1		PE2		PE3	
clk	Diff	SAD	Diff	SAD	Diff	SAD	Diff	SAD
0	C00-R00							
1	C10-R10		C00-R10					
2	C20-R20		C10-R20		C00-R20			
3	C30-R30		C20-R30		C10-R30		C00-R30	
4	C01-R01		C30-R40		C20-R40		C10-R40	
--	---		---		---		---	
14	C23-R23		C13-R23		C03-R23		C32-R62	
15	C33-R33	SAD00	C23-R33		C13-R33		C03-R33	
16	C00-R01		C33-R43	SAD10	C23-R43		C13-R43	
17	C00-R11		C00-R11		C33-R53	SAD20	C23-R53	
18	C00-R21		C00-R21		C00-R21		C33-R63	SAD30

（c）1 次元シストリックアレイのデータフロー

図 3・9　1 次元シストリックアレイ型 ME の構成例

BM は画素ごとの差分絶対値とそれを MB 単位に積算する演算の繰り返しであることから，同じプロセッサエレメント（PE）を配列する高並列演算器である SIMD またはシストリックアレイ（SA）型構成が適している．高い演算効率が得られることから VLSI では SA 型が主に採用されている．SA 型 BM 演算器は大きく分類して PE の配列が 1 次元型と 2 次元型（デー

タ供給方法による分類ではないことに注意），PE で SAD まで実行する一括型と Distortion のみで積算は PE 外の木構造加算器で行う分割型とに分類できる [2].

図 3･9(a) は 1 次元一括型構成の代表例 [3] を示す．探索範囲の水平画素数分の PE を 1 次元に配置して，各 PE に現 MB のデータをつづら折に伝播供給し，参照データを放送供給する．図(b) に示すように PE は差分絶対値とそのクロックごとの積算を実行する．図(c) のデータフローに示すように $N \times N$ クロック後には SAD 値が PE から出力される．

図 3･10 は 2 次元アレイの構成例を示す [3]．図(a) は現 MB の画素マトリクスに対応した 2 次元 PE アレイに参照画像データをつづら折に供給していくタイプである [4]．この構成は，参照画像を水平方向にずらしていくときには効率よくデータ供給ができるが，垂直方向にずらせるときには，無効演算期間が生じる．しかし，物理イメージに合った構成であるため，データ供給制御が簡単であるという利点を有する．図(b) の構成は，現 MB 画素に対応した 2 次元 PE アレイに参照画像を放送供給して，差分絶対値をつづら折にシフト加算していくタイプである [5]．この構成では，無効演算期間をなくすことができるうえに総和演算回路が不要であるため，高い稼働率とハードウェア規模の削減が可能である．しかし，図(a) に比べると，データフロー制御が複雑になる．

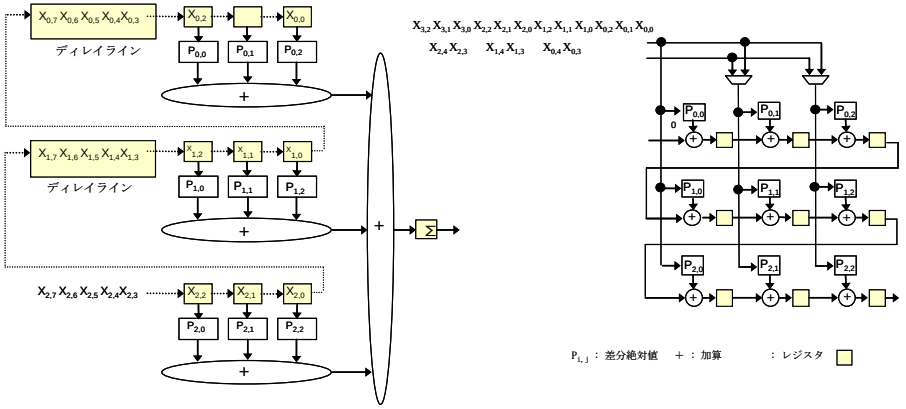


図 3・10　2 次元シストリックアレイの構成例

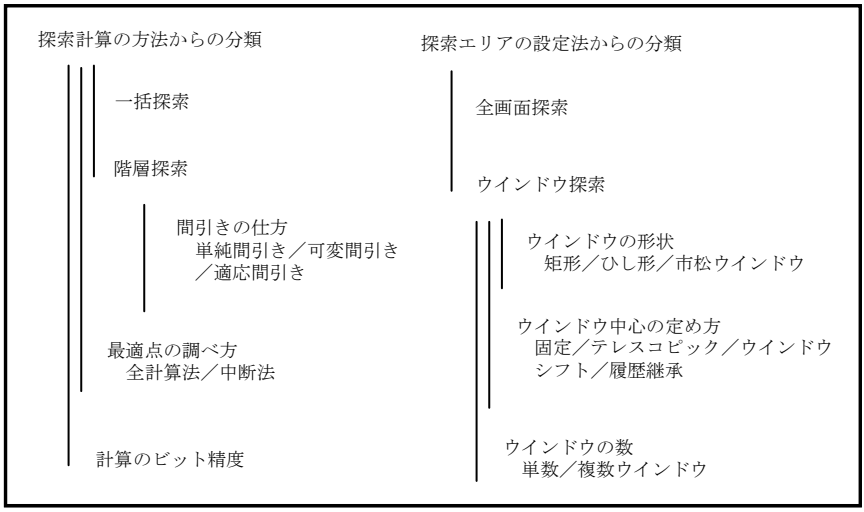
(3)　動き探索の演算量削減アルゴリズム

HD 画像の圧縮で要求される±200 画素にも及ぶ広い探索範囲をしらみつぶしに探索したのでは 1012/秒をはるかに超える演算量となり，(2) 項の高効率演算器をもってしても LSI 化は現実的ではない．したがって，高効率演算器に加えて広い探索範囲を高精度に探索し，なおかつ演算量を抑えることのできるアルゴリズムの開発が必須となる．

表 3･1 に VLSI 向きの動き探索アルゴリズムの体系を簡単に示す．探索エリア設定の仕方からの分類と最適探索点計算方法からの分類に分けて示した．最適探索点計算法で広く使われるのは，粗い画素ピッチの 1 次探索による広範囲探索から SAD 最小に近い 1 画素精度の擬似最適探索点を見つけ，その近傍の 2 次探索以降で 0.5 画素精度の最適点を探索するという

階層探索である．画素の間引き方には単純間引き法，中心位置よりも周辺を多く間引く [5]，または複数種類の間引きを組み合わせる可変間引き法 [13]，最も輝度値の変化が激しいラインのみを探索する適応間引き法 [6] などがある．また，最適点の調べ方には SAD 値の減少が最大の方向を探索していく傾斜法 [7),10)]，SAD 値が設定した値を下回った時点で探索を打ち切る中断法 [8] などがある．更には，計算のビット精度を削減する方法も提案されている [9]．

表 3・1　動きベクトル探索アルゴリズムの体系



探索エリア設定法からの分類では，1 次探索のある領域に当たりをつけて狭いウインドウで広い探索を行えるようにするテレスコピック法 [11]，ウインドウシフト法 [12),13)] などが効果的である．また，符号化済み隣接 MB の MV，前フレームの同位置 MB の MV，ゼロベクトルなどから探索範囲を絞り込む履歴継承法 [13] も有効である．

(4)　高効率動きベクトル検出器の構成

(a)　階層的テレスコピック探索 [11]

テレスコピック探索法は，隣接画面間ではオブジェクトの動きに強い相関があるという前提に基づいたアルゴリズムである．例えば，$M = 3$ の場合における P ピクチャ間の動きベクトル検出では，間に 2 枚のピクチャがあるため，動きの激しい画像の動きを直接検出するには広い探索エリアの設定が必要である．テレスコピック探索は，間にある 2 枚のピクチャの動きベクトルを順次トレースして，探索範囲の広がりを抑えるものである．ちょうど望遠鏡で遠くを覗いているような格好になるところからこの名前が付けられている．ピクチャ間の距離 D に比例して，探索範囲の中心位置がずれていく．このため，ピクチャごとの動き検出演算量を変えることなく，等価的に最大 D 倍のエリアを探索したことになる．動きを予測して投機的に探索ウインドウをシフトしたりホッピングしたりする方式とは異なり，ピクチャ間で動きが急に変わっても対応しやすい方式であり，このことがテストモデルに採用された

大きな理由にもなっている．このテレスコピックに階層探索を組み合わせたものが階層的テレスコピック探索法である．その様子を**図 3･11** に示す．

ここでは，2 画素（水平，垂直とも 2 : 1 サブサンプリング），単画素，0.5 画素の 3 段階の階層探索を実行している．第 1 段階探索は，現 MB，参照画像ともに 4 : 1 サブサンプリングを行うので，演算量は 1/16 に減少する．しかし，メモリバンド幅が大きくなる欠点を有する．標準画像を使って調べた PSNR 劣化量は画像によって異なるが，最も大きい場合でも全探索より 0.3 dB ほどの劣化で抑えられている．

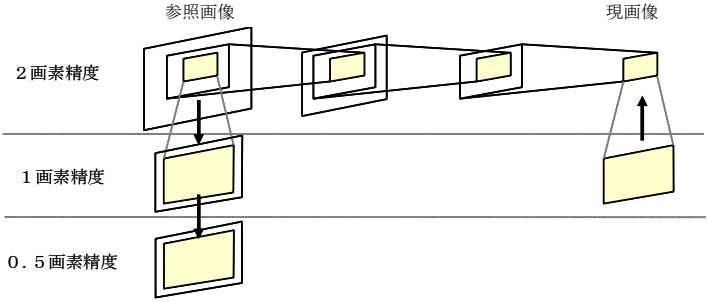


図 3・11　階層テレスコピック探索の概念図

(b)　適応的ウインドウシフト動き探索 [13)]

2 階層探索と過去に求めた動きベクトルの統計を参照して，探索範囲の形や場所をピクチャ単位に適応的に変える探索法を組み合わせている．1 次探索では，水平方向のみの 2 : 1 サブサンプリングを現 MB と参照画像の両方に適用して，水平 2 画素精度，垂直単画素精度の探索を実行する．これで，演算量を 1/4 に抑えている．**図 3･12** に示すように，過去に求めた動きベクトルを参照する適応型探索ウインドウを採用して広い範囲を探索する．

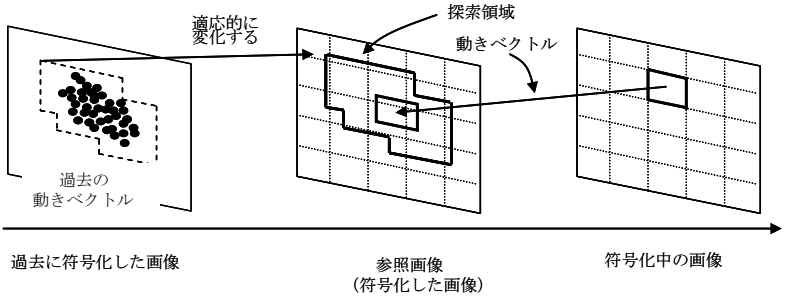


図 3・12　履歴適応型動きベクトル探索法

画像に動きが少ないときには，ウインドウを小さくし，動きが激しいときには大きくする．また，ウインドウの形状を矩形ではなくひし形に設定して，その分演算量を小さくしている．PSNR は MPEG テストモデルに比べて約 0.4～0.8 dB も向上させることができるとしている．

2 次探索は，1 次探索で得られたベクトルを中心に，0.5 画素単位で，水平 5 点，垂直 3 点で並列に SAD 値を求める．この 0.5 画素精度の演算器を 3 セット用意して，フィールド予測の top 側，bottom 側及びフレーム予測の三つの動きベクトルを同時計算することができる構成になっている．

以上，MPEG-2 における動き探索演算器構成法については詳しく述べたが，DCT 変換・量子化と可変長符号化の構成法については，3-1-1 節の JPEG 構成法ところで記述したのと大きく変わらないためにここでは省略した．

3-1-4 H.264 中核処理の実現法

（執筆者：笠井良太）［2009 年 11 月 受領］

(1) 処理の概要

H.264 符号化処理 [14] の概要を**図 3･13** に示す．MPEG-2 と異なる点は，画面内（イントラ）予測を使うこと，DCT の代わりに整数変換を用いること，メインプロファイルやハイプロファイルではエントロピー符号化に算術符号化（CABAC）を使うこと，ループ内フィルタとしてデブロッキングフィルタを導入したことなどである．更に，可変ブロックサイズ，1/4 画素精度の動き補償，多様なイントラ予測モードを有し，すべてのブロックサイズ，予測モードについて符号化コストを評価して最適な組合せを選ぶ RD 最適化が組み込まれており，MPEG-2 に比べて約 2 倍の高い圧縮効率を実現できる半面，10 倍以上の膨大な演算量を必要とする．なかでも，ME/MC と CABAC 処理が演算量も大きく複雑な処理であるため，高度な並列処理手法による回路規模や消費電力削減の実現が VLSI 化にあたり大きなテーマとなっている．

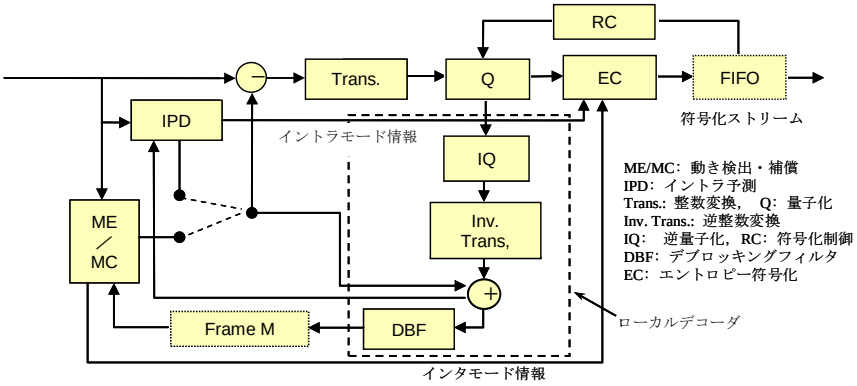
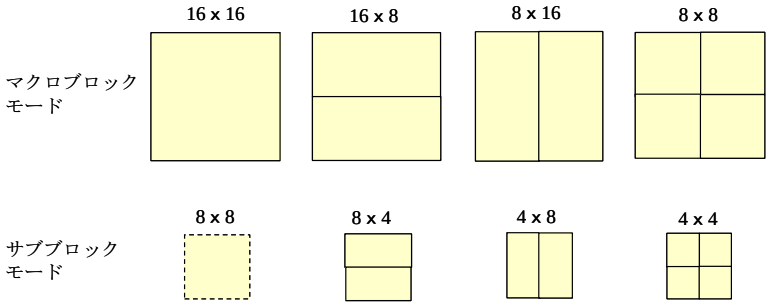


図 3・13　H.264 ビデオ符号化処理フロー

(2) 可変ブロックサイズの ME/MC

MPEG-2 では，16×16 のサイズに固定された 1/2 画素精度の ME/MC 処理で，参照先フレームも固定されていたのに対して，H.264 では**図 3･14** に示すように，16×16，16×8，8×16，8×8 の 4 種類ブロックと 8×4，4×8，4×4 の 3 種類のサブブロックの合計 7 種の可変ブロックサイズ，41 個 MV に対して 6 タップフィルタを使った 1/4 の画素精度の ME/MC

を行い，ブロックごとに異なるフレームを参照（MRF：Multiple Reference Frames）するというように MPEG-2 よりも大幅に演算量が増える．このため，単画素精度までの 1 次探索（これを IME，Integer ME という）と 1/4 画素までの 2 次探索（FME：Fractional pixel ME）に分けた階層探索が一般的となっている. FME は MV コスト計算に基づくインターモード選択と動き補償を合わせて実行する．



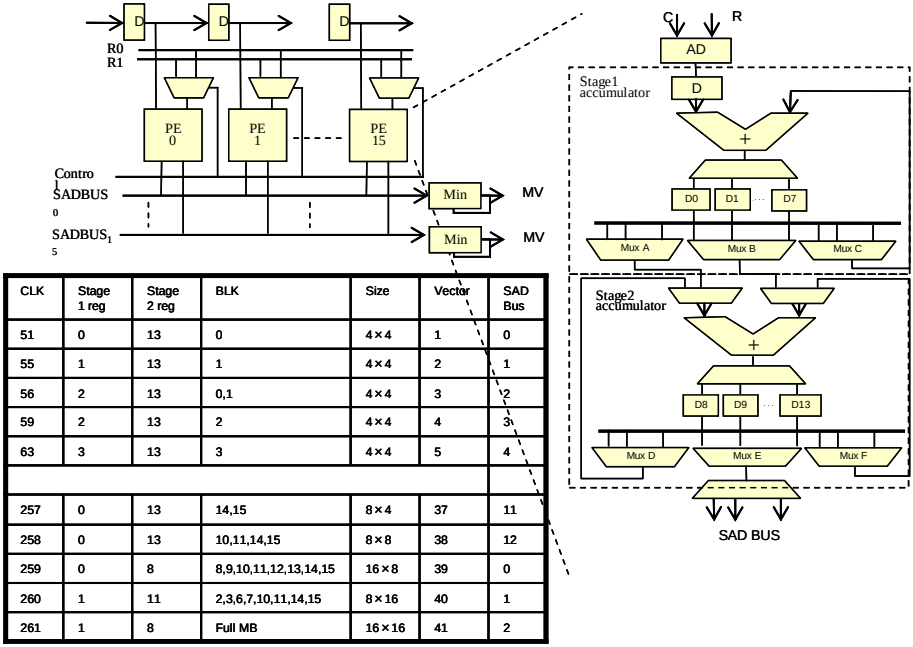
モードの数（ブロックサイズの種類）：7，　MVの数：1+2+2+4+4×(2+2+4) = 41

図 3・14　H.264 動き補償における可変ブロックサイズ

可変ブロックサイズの IME をブロックごとに順次処理を行ったのでは実時間処理 LSI は実現できないため，並列処理技術を駆使して一つの演算器内でブロックサイズの異なる複数の BM を同時に実行するような工夫が各所で試みられている．例えば，16×16 サイズの BM を実行する性能をもつ SA 演算器に，4×4 ブロックを単位とした 16 個の部分 SAD 値に分割して求める機能を付加して, 部分 SAD の組合せで 41 個すべての SAD 値を求める方法などがある．

図 3･15 に 1 次元 SA 型で 41 個の ME 計算を並列に求める BM 演算器と PE の構成を示す [15]. 図 3･9 と同様の 1 次元 SA 構成であるが，各 PE は各ブロックサイズの SAD を無駄なくすべて算出できるように 8 個のレジスタをもつ 1 段目の累算器と 6 個のレジスタをもつ 2 段目の累算器からなる複雑な構成をとる．1 段目の 8 個のレジスタに 4×4 ブロックの部分 SAD 値が保存されたのち 2 段目の 6 個のレジスタに転送され，2 段目累算器でより大きなブロックの SAD 値が生成される．各 PE では 261 サイクル（内部レイテンシー5 サイクルを含む）の間に 41 個の SAD 値が生成され，16 個の PE から水平方向 16 画素の探索範囲の 16×41 個の SAD 値が SADBUS に出力され，比較器で各ブロックの最小 SAD をもつ ME を求める．探索範囲 16×16 の可変ブロックサイズ BM が，ほぼ 256×16 の 4096 サイクルで完了する．同様に，2 次元 SA を用いた高効率な可変ブロックサイズ BM も数多く提案されている [16), 17)].

これらの BM 演算器は，探索範囲が 32×32 程度までの探索であれば，妥当なコストで LSI 化できるので，SDTV クラスの TV 会議や監視装置の応用では，隣接ブロックの MV から予測値を中心点とした探索範囲削減策と組み合わせて使われている．しかし，HD スポーツ画像のように探索範囲の大幅な拡大が必要な場合や，HD カメラのように消費電力の大幅な削減が必要な場合には，3-1-2 節の(3)で述べたと同様の様々な処理量削減アルゴリズムが合わせて適用されている．また，SDTV 以上の解像度を対象にする場合は圧縮効率向上に効果の少ない 8×8 未満のサブブロックの ME を省略するなどの削減策もよく使われる．



CLK	Stage 1 reg	Stage 2 reg	BLK	Size	Vector	SAD Bus
51	0	13	0	4×4	1	0
55	1	13	1	4×4	2	1
56	2	13	0,1	4×4	3	2
59	2	13	2	4×4	4	3
63	3	13	3	4×4	5	4
257	0	13	14,15	8×4	37	11
258	0	13	10,11,14,15	8×8	38	12
259	0	8	8,9,10,11,12,13,14,15	16×8	39	0
260	1	11	2,3,6,7,10,11,14,15	8×16	40	1
261	1	8	Full MB	16×16	41	2

図 3・15　1 次元シストリックアレイ型可変ブロックサイズ MV 探索器

FME では，IME で求めた 41 個の単画素精度 MV に対して，その周辺を 1/2 画素，続いて 1/4 画素精度の探索を±1 の探索範囲で行う．単画素精度 MV はそれぞれ異なる座標と参照フレームを指すため参照データの再利用性に乏しいこと，隣接 MB からの PMV（MV 予測値）を使った MV のコスト計算とそれに基づくモード判定と MC に対して，通常の MB パイプライン処理では左隣の MB の MV は確定していないことなど，FME の高効率処理実現には注意点が多い．

(3)　高効率 ME/MC の構成例

(a)　再構成可能なリング接続型シストリックアレイを用いた IME 構成例 [18)]

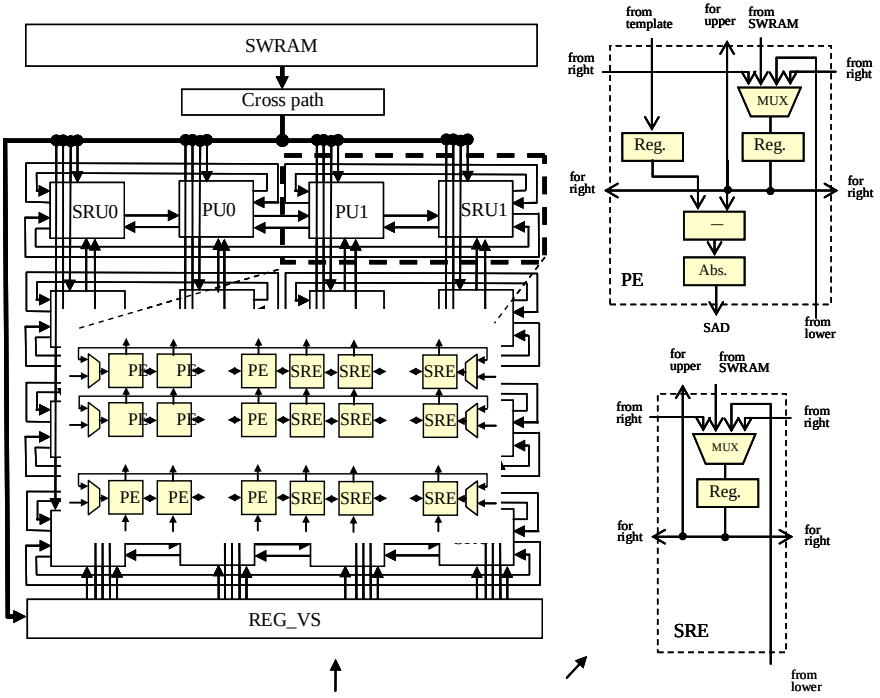
可変ブロックサイズと MBAFF に適用可能な IME を提案している．探索アルゴリズムは粗い探索と細かい探索の 2 段階方式で，粗い探索には VLSI 向きに考案した勾配法である CRCS（Complementary Recursive Cross Search）を導入，細かい探索には粗い探索の MV の分布状態を見て MB ペアの探索と小ブロックに分けた狭範囲全探索を適応的に切り替えて用いる方法を提案している．

粗い探索の初期値にはゼロベクトル，上，右上，左の隣接ブロックの IME の結果ベクトルの 4 候補のうち一つを使う．CRCS は，水平と垂直の 1 次元ダイヤモンドサーチ（1-DS）を相互に繰り返して最小 SAD 点を見つける RCS 法に，水平から探索開始のものと垂直から開始のものを 2 通り実行してより小さい SAD 点を見つけるという改良を加えている．また，探

索は水平方向に 1/2 サブサンプルされた field-16×8，field-8×8，frame-16×16，frame-16×8，frame-8×16，frame-8×8 の 6 種類のブロックに対して実行し，それらの部分 SAD 値をそれより小さいサイズのブロックの探索に再利用する．

field-16×8 の Top Field と Bottom field, MB Pair の上下 MB の合計 8 本の MV が時間的及び空間的に局所分布しているかどうかの判定条件により，局所分布していると判定される場合は MB ペアの細かい探索（FSMP）に，そうでない場合は個別の小ブロックの探索（FSSB）にブランチする．FSSB では，ゼロ，隣接 3 ブロックのベスト IME，粗い探索のベスト 3 ポイントの合計 7 ポイントをランダムブロックマッチング（RBM）して，最適のポイントから±4 で全探索（FS）を実行する．一方，FSMP では MB ペア（16×32）の±4 の全探索を実行する．ほかのブロックサイズはその部分 SAD を再利用して求める．

提案した IME は探索範囲が±128（H），±64（V）と比較的広く，JM に組込まれている UMHS 法に比べて処理負荷が 41 %まで軽減でき, 様々な HD 映像に対しても PSNR は 0.06 dB 低下するにとどまっている．



(a)再構成可能なリング接続型SA　　(b)PEおよびSRE回路構成

図 3・16　再構成可能なリング接続型シストリックアレイを用いた IME 構成例

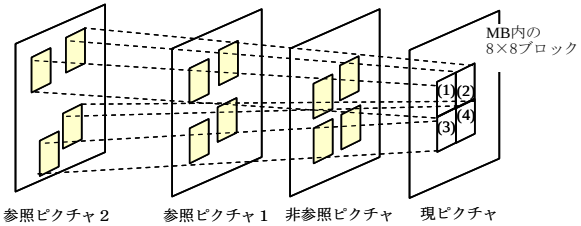
この IME のアルゴリズムを効率よく実行できる VLSI アーキテクチャとして，再構成可能なリング接続型のシストリックアレイ（RRSA）と探索ウインドウ RAM（SWRAM）からなる構成を提案している．**図 3･16**(a) に RRSA のブロック構成と RRSA の構成要素のサブブ

ロック SA（SBSA）を，図(b) に SBSA の構成要素である PE とシフトレジスタ（SR）の構成を示す．SBSA は 8×8 の PE と 8×8 の SR からなっており，8×8 ブロックサイズの SAD が図 3･3(a) の SA と同様な方法で計算できる．RRSA は 2×4 個の SBSA をもっており，隣り合う SBSA 間は双方向にリング接続されているので，8×8 より大きいサイズの SAD 計算のための再構成ができる．また，SWRAM は矩形領域のデータを水平・垂直方向任意の場所から連続アクセス可能な構成になっていて，無駄のない参照データ供給ができる．これとリング接続によって参照データの転送が左右，上下に効率よく切替えできるため，1-DS，RBM，FS などの高効率演算も可能である．

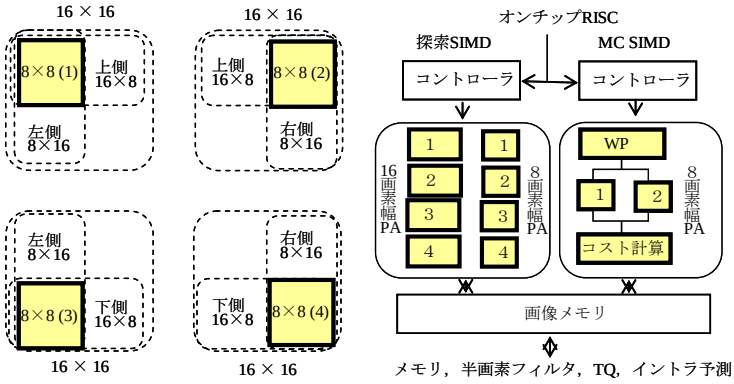
(b)　マルチレファレンス・マルチブロックサイズ ME/MC の構成例 [19)]

本構成例は，4：2：2 プロファイル対応の放送用途 HDTV コーデックに向けて，HD 映像の圧縮に効果の少ないサブブロック以下の ME/MC は省略し，テレスコピックサーチを使って比較的狭い範囲の BM で広い探索範囲を実現したものである．

図 3･17(a) に示すように, IME では 2 画素精度のテレスコピック探索で 4 つの 8×8 ブロックの BM を並列に実行し，マルチプルレファレンスの MV を効率よく求めている．この IME 演算器は図 3･10(b) の 2 次元 SA 型の 4×4 PE アレイを 2 並列動作させている．



（a）2 画素精度テレスコピックによる 4 ブロック並列探索



（b）8×8 ブロックベースの包括的 FME　　（c）FME モジュールの構成

図 3・17　マルチレファレンス・マルチブロックサイズ ME/MC の構成

IME で求めた四つの MV と参照フレームを使って，図 3･17(b) に示すように 16×16，16

×8，8×16，8×8 ブロックの 1/2 画素と 1/4 画素の FME を順次実行する．例えば，上側の 16×8 ブロックは IME で求めた 8×8 (1) と 8×8 (2) を含んでいるので，それぞれの MV を中心とする 16×8 の FME を 2 回実行する．この FME の構成を図 3･17(c) に示す．

IME の結果が揃うと，対応する参照データは 1/2 画素補間されイメージメモリに蓄積される．探索制御用の SIMD プロセッサの指示に従い，8 つの PE アレイが各可変ブロックサイズの 1/2 画素，続いて 1/4 画素の 9 点探索を並列に実行する．PE アレイは 16 幅と 8 幅の 2 種類をもっており，前者は 16×16，16×8 ブロックに，後者は 8×16，8×8 ブロックの L0，L1 探索に使われる．すべてのブロックの ME 候補が計算されるとその中から SAD と MV コストの最小のベスト MV が決定される．

このベスト MV に基づいて各ブロックサイズの双方向 MC が，MC 制御用 SIMD の指示のもと，二つの 8 幅の PE アレイを使って FME と並列実行される．この処理フローを**図 3･18** に示す．IME と FME，モード選択と MC までの処理が 1 MB サイクル内で完了する．

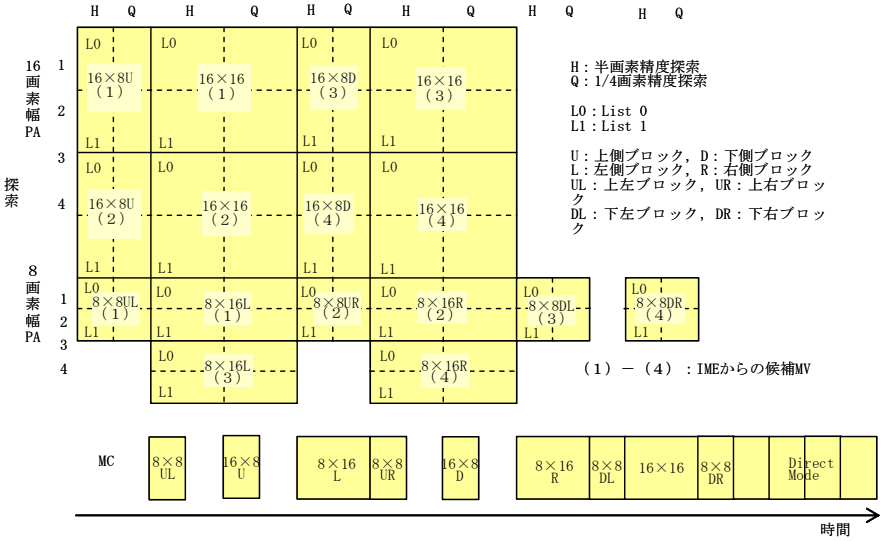


図 3･18　マルチブロックサイズの MV・MC 計算フロー

この ME/MC 演算器は，二つの SIMD プログラムによって PE アレイのデータパスの動作を変更することができ，PAFF，MBAFF，4：2：2，重み付け予測などの様々な符号化モードに対応できるようになっている．また，Intra/Inter 判定やモード選択のための符号化コストの計算に対しても，外部からコスト計算のオフセットやしきい値設定などができる柔軟性をもたせている．この結果，探索範囲が水平 －271.75 / +199.75，垂直 －109.75 / +145.75 という広い探索範囲と JM に対して PSNR がほぼ同等になる探索精度を実現している．

(4) 整数変換・量子化とイントラ予測の構成法

H.264 の直交変換は，MPEG-2 の DCT の実数演算手順の不一致にともなう ICD ミスマッチの問題を避け，16 ビット精度の整数の加減算のみで実現可能となっている．また，量子化も簡単なテーブル参照とシフト演算で行えるように改良されている．このため，ハードウェア実現に特別なアーキテクチャやアルゴリズム導入は必要とされない．

イントラ予測では，周辺ブロックの画素値から 4×4 ブロック／16×16 ブロックの画素値を予測する．予測モードが 4×4 で 9 通り，16×16 で 4 通りあり，その中から最適なモード選択をすることが要求されている．インター予測（MC のモード選択）と合わせて最適なモード選択が必要となり，この複雑な処理を MB サイクル内で実現するために様々な工夫がなされている．

(5) デブロッキングフィルタ（DF）の構成

DF は DCT 量子化処理で発生するブロック歪が他ピクチャに伝搬しないように動き補償のループから再生画像をフレームメモリに格納する前に設けられたブロック歪除去フィルタである（図 3･6 参照）．処理対象のブロック境界がブロック歪の発生しやすい条件であるかどうかを判定して適応的にフィルタ強度を変更させてすべてのブロック境界に適用する．1MB 内には輝度 32，色差 16，合計 48 のブロック境界が存在し，それぞれに水平または垂直フィルタ処理を行う．ブロック画素の縦横を入れ替えるトランスポーザや特別な 2 ポートメモリを用いて 1 回のメモリアクセスで水平・垂直フィルタ処理用のデータをもってくる構成と効率的なパイプラインを組むことにより，MB 当たり 192 フィルタサイクル（1 フィルタサイクル:メモリのリードライトとフィルタ処理)で実行できる DF 実現例が報告されている [20), 21)].

(6) CABAC の構成

CABAC は適応的算術符号化の一種であり，図 3･19 に示すように多値信号を 2 値に変換する 2 値化部と 2 値算術符号化部に加えて，符号化すべき 2 値信号の出現確率を符号化の状況に応じて計算・更新するコンテキスト計算部からなる [22), 23)].

2 値化部は符号化のシンタックス要素（多値信号）を一意の 2 値信号（Bin 系列）に変換する．シンタックス要素に応じて複数の 2 値化方式が単独または組み合わされて適用される．

2 値算術符号化部では，符号化の状態を Bin 系列のシンボル（0 または 1）の存在区間の幅（Range）と下端（Low）で表す．Range と Low のビット幅は 9 ビットと 10 ビットであり，符号化の先頭では 0x1FE と 0x000 にそれぞれ初期化される．シンボルのうち出現確率の高い方を MPS, 低い方を LPS に割り当てて, LPS の出現確率(rLPS)はその状態を示す番号(state)と Range によってコンテキスト計算部のテーブルから引くことにより求まる．MPS は Range－rLPS で求まる．

図 3･19 に概略処理フローを示す．シンボルが LPS であるとき，Range には rLPS が設定され，Low は MPS の出現確率を加算することで更新される．State が 0 のときには，次の符号化状態で MPS と LPS が逆転すると予想されるため, 対応するシンボルを入れ替える．一方，MPS の場合は，Low は変化せず，Range には MPS の出現確率が設定され，MPS に対応する遷移テーブルを引くことにより state を更新する．Range < 0x100 のときには再正規化処理が行われ，Range と Low が拡大される．

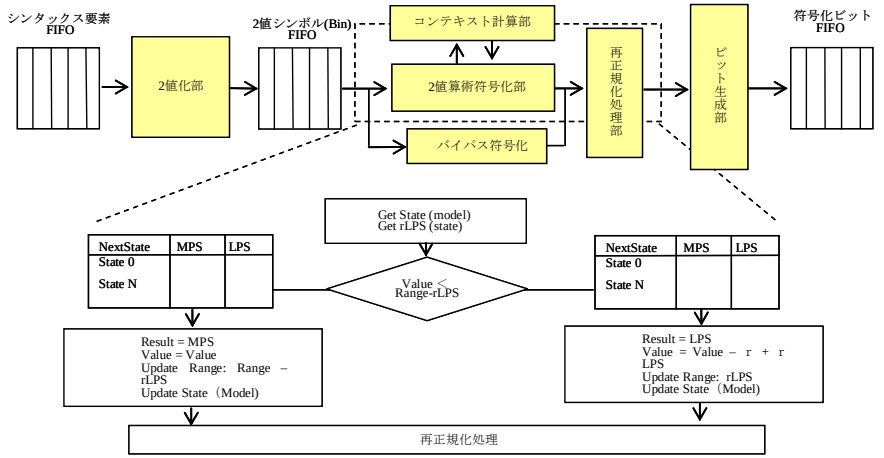


図 3・19　CABAC の概略処理フロー

再正規化処理では，Low < 0x100 のとき出力は 0 に確定し，0x200 ≦ Low のとき 1 に確定して Low から 0x200 が減算される．しかし，その中間ではその後の符号化状態に出力が依存するため，未確定値（Outstanding bit）を設定して Low から 0x100 が減算される．その後，Range と Low は左 1 ビットシフトで 2 倍に拡大される．Range が 0x100 以上になるまで再正規化処理が繰り返され，その回数分の Outstanding bits が出力される．再正規化処理終了後，確定した 0 または 1 を出力すると同時に Outstanding bits の出力を確定（出力シンボルの反転シンボルを出力）する．このような処理では，再正規化処理ごとに終了判定をし，Outstanding bits をその後に確定するため，高速な符号化ができないという問題がある．例えば，Bin 系列一つ当たり最大8回の再正規化処理が連続して最悪16サイクルの処理ステップが必要となってしまう．

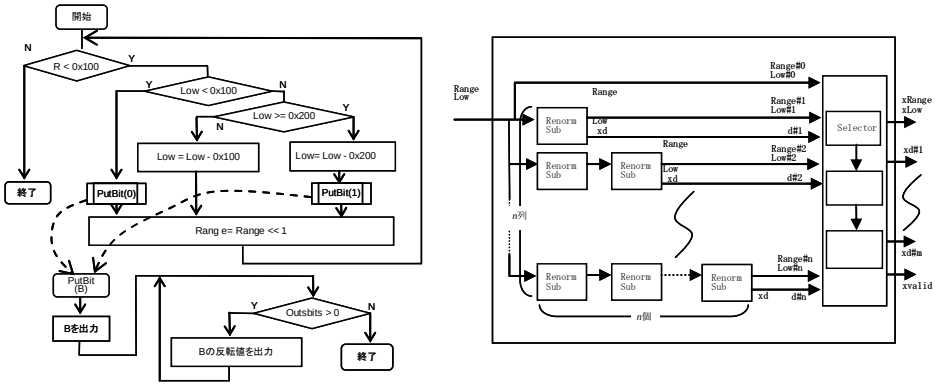


図 3・20　再正規化処理の並列化による CABAC 高速化

CABAC の高速化は各所で試みられているが，その代表例を**図 3･20** に示す．再正規化処理の回数は Range の最大シフト数 8 回であるので，1 回から 8 回までの再正規化処理回路を 8 個並列に並べられ，各回路からは 0，1 または未確定 S の 3 値が出力される．Range のシフト回数に対応した各回路からの出力を集めて Low を選択するようにすることで，1 サイクル内で再正規化処理を終了することができる．

このほか，符号化処理ではスライス分割をして並列度を上げることができるが，応用範囲が限られてしまう．また，投機的な処理や複数の bin を並列に処理する工夫を施している例もある．復号化処理においても，符号化処理とほぼ共通の処理になるため，これらの高速化手法は同様に利用できる．

3-1-5 レートコントロールの実現方法

（執筆者：笠井良太）［2009 年 11 月 受領］

レートコントロール（符号量制御）とは，最適な符号化モード（フレーム予測／フィールド予測，イントラ予測／インター予測，参照フレーム選択，ブロックサイズ選択など）の決定と量子化係数を調整して出力ストリームのビットレートと映像品質を制御することをいう．用途によって，ビットレートを一定に保つ CBR とビットレートを可変にして品質を一定に保つ VBR とがある．

リアルタイムエンコーダの場合，符号化モードの決定のために実際に符号化と復号化を実行して最適なものを選ぶというマルチパス方式は使えないため，MB の処理時間内に各符号化モード選択時の SAD 値とヘッダ符号量推定値を比較して各々が最小となるモードを選択する必要がある．すべてのモードを比較するのは困難なために，入力映像の性質を符号化前に分析（入力映像のシーンチェンジやフェード状態の自動検出やノイズカットなど）して比較すべきモードを絞り込むような方法がとられる．また，放送用途のような高品質を要求される場合，実時間 2 パス符号化（1 段目の符号化情報を使って 2 段目の符号化モードを決定する）方式が採用されることもある．

ビットレート制御は，GOP レイヤ，ピクチャレイヤ，MB レイヤの 3 階層に分けて行われる．GOP レイヤでは，残りの GOP 内ピクチャにどの程度の符号量を割り当てることができるかを計算し，次ピクチャの目標符号量を指示する（ステップ 1）．ピクチャレイヤでは，仮想バッファのフルネスを評価して，処理すべき MB の平均量子化係数を計算する（ステップ 2）．MB レイヤでは，上記量子化係数の平均値を守りながら MB ごとの空間アクティビティ（画素輝度値と MB 平均輝度値との差分）を使って符号量を変化（微調整）させる（ステップ 3）．仮想バッファのフルネス評価とは，VBR，CBR ともにピクチャごとの発生符合量の変化をエンコーダ出力部とデコーダ入力部のバッファメモリ（FIFO）で吸収して滑らかな符号化，復号化が実行できるようにする制御のことをいう．この制御は符号化処理のレイテンシー（遅れ時間）に関係しており，低遅延のコーデックシステムを実現するうえで重要なファクタとなる．

3-1-6 リアルタイムコーデック LSI の構成法

（執筆者：笠井良太）［2009 年 11 月 受領］

映像コーデック LSI のアーキテクチャは，3-1-3 節，3-1-4 節で詳しく述べたように個々の符号化処理に対して高性能な専用ハードウェアを実現して，これらを組み合わせるアプローチ（以下，専用ハードウェアビルディングブロック方式：DHB と呼ぶ）と MPU や DSP の新

規アーキテクチャを取り入れた高性能プロセッサを活用するマルチプロセッサ型アプローチ（MPA）に分けることができる．MPA 型には，符号化処理に特化した DSP や SIMD などの異種のプロセッサを組み合わせるメディアプロセッサアプローチ（以下，HMA と呼ぶ），符号化処理に特化した同種のベクトルプロセッサをアレイ状に並べるアプローチ（VPA と呼ぶ），コンフィギャラブルプロセッサ＋専用エンジン構成（CPA）などがある．DHB はハード規模，消費電力を最小化できる半面フレキシビリティに乏しい，一方，MPA はハード規模や電力が大きくなるがソフトウェアによるプログラマビリティの確保ができる．

(1) 専用ハードウアビルディングブロック（DHB）型

低コスト・低消費電力が要求される携帯機器向けから放送プロフェッショナル向けまでの幅広い応用に採用例が数多く見られる．MPEG-2 エンコーダ LSI の代表的なブロック構成を**図 3･21** に示す [24]．映像入力処理部，ME 部から VLC 部までは MB レベルのパイプライン処理が行われ，処理モジュール間のデータ転送は MB 単位で FIFO または外部フレームメモリ（SDRAM）を介して行われる．

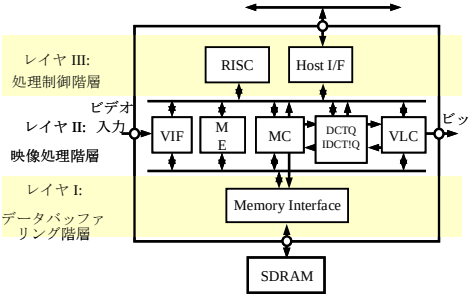


図 3・21　DHB 型 MPEG-2 エンコーダ LSI

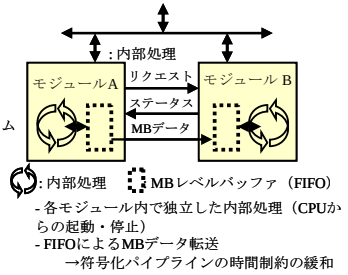


図 3・22　モジュール独立性保持機構

図 3･22 に示すとおり各モジュールの処理は独立して実行できるので，符号化パイプラインの時間制約を大幅に緩和することができる．最も処理の重い動きベクトル検出には階層探索法などを採用して回路規模と消費電力削減を行っている．各モジュールの起動・停止を含む全体の処理コントロールと符号量制御（レートコントロール：RC）には，RISC や簡単なシーケンサなど用いる例が多い．制御用パラメータを外部からレジスタ指定することと内部の量子化情報，発生情報量などを参照可能にすることにより，一定レベルのユーザカスタマイズを可能としている．この意味ではプロセッサとの組合せになっており，最近の符号化 LSI では純粋な DHB 方式採用例はないといっても過言ではない．

H.264 エンコーダ LSI への適用例を**図 3･23** に示す [25]．M-Core は ME/MC と IPD（イントラ予測），TQ（整数変換・量子化）の各処理を MB パイプラインで動作させており，C-Core では，LF（ループフィルタ）と EC（エントロピー符号化）をスライス単位に処理をする．いずれも隣接 MB 情報に基づいた複雑な適応処理を含めて 32 bit RISC で制御を行っている（MRISC，CRISC）．全体制御とスライスより上位の符号量制御を TRISC が行っており，制御階層が 2 階層になっている．フレームメモリは再生画像をオンチップの DRAM（eDRAM）に，原画像は外部 DDR メモリに割り付けることにより，外部メモリバンド幅の大幅な削減

を行っている．

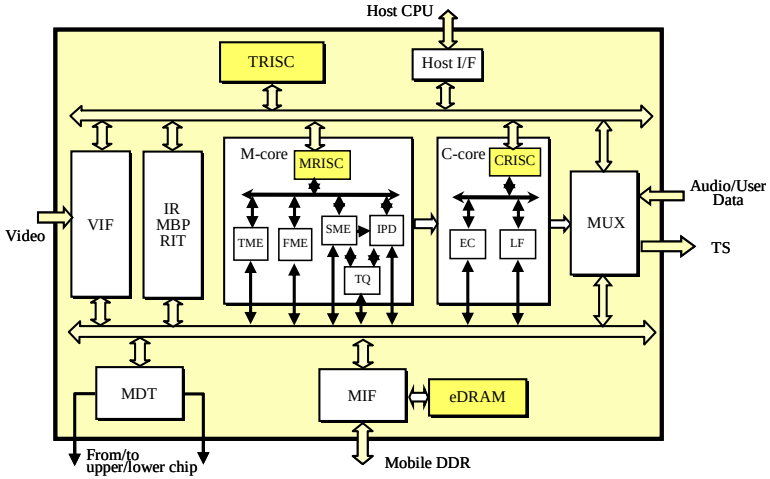


図 3・23　DHB 型 H.264/MPEG-2 エンコーダ LSI

(2)　ヘテロジーニアスマルチプロセッサ／メディアプロセッサ方式（HMA）

代表的な LSI のブロック構成を**図 3･24** に示す [26]．全体制御に MIPS，レートコントロールに SPARC という 32 ビット RISC プロセッサコアを，画素レベルの画像処理実行にメディアプロセッサ／ビデオ DSP を搭載し，ME とエントロピー符号化には専用ハードエンジンを備える．メディアプロセッサ部は DCT，IPD，MC などの符号化処理とノイズカットフィルタなどの映像処理を高効率に行われるように，VLIW などのインストラクションレベルの並列化や SIMD などによるベクトル演算などが取り入れられている．H.264 における隣接 MB 情報の参照，符号化モードの決定などベクトル演算処理能力の向上だけでは対応できない処理のために様々な工夫がされている [27]．しかし，可変ブロックサイズ ME や CABAC などに対応するには処理能力が不足するため専用ハードウェアエンジンを搭載している．

このほか，プロセッサコアと外部のシステム（ホスト CPU，ネットワーク，記憶媒体など）を結ぶ PCI インタフェース，HDTV などへの拡張のためのマルチチップ構成を可能とする IPC インタフェース，外付けのフレームメモリ（DDR メモリ）とのインタフェースをつかさどるメモリコントローラを搭載する．

この LSI は MPEG-2，H.264，VC1 などのマルチ規格の符号化／複合化ができ，マルチチップ構成によって 422 プロファイルや 1080/60p などの広いアプリケーションに適応できる．この柔軟性は，CPU と DSP のプログラマビリティによる．

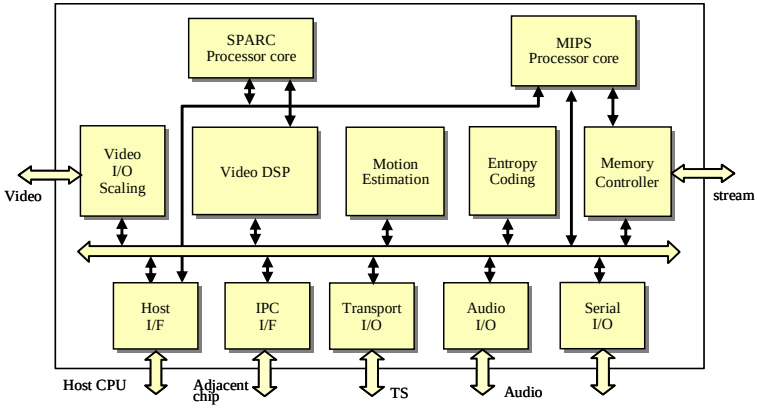


図 3・24　ヘテロジーニアスマルチプロセッサ型 H.264 コーデック LSI

(3)　VLIW プロセッサアレイ型（VPA）

図 3・25 に代表例を示す [28]．この例では，ベクトルプロセッサタイル（VP Tile）とスカラープロセッサタイル（SP Tile）の 2 種類の PE をタイル状に連結することでスケーラブルに処理能力を変更できるアーキテクチャを実現しており，各 PE は VLIW 型並列プロセッサと 2 次元ビデオデータに効率よくアクセスする DMA システムからなる．LSI 構成時に並列度や記憶容量などを目的に合わせてカスタマイズすることで電力効率，面積効率を高めることができる．また，AHB，OCP，AXI のような標準的なバスプロトコルをサポートしている．VP Tile は命令セットとアーキテクチャが符号化，復号化やその前処理，後処理に最適化されており，再構成可能な SIMD 型ベクトル演算器とスカラーデータパスを有する．SP Tile はエントロピー符号化と符号化制御に合わせた命令セットアーキテクチャとなっている．

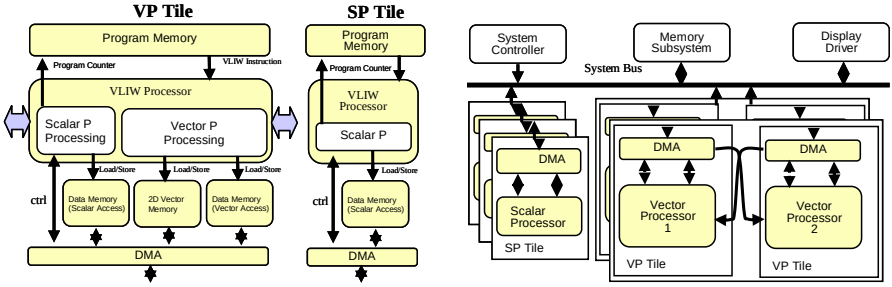


図 3・25　VLIW プロセッサアレイ型構成例

(4)　コンフィギャラブルプロセッサ＋専用エンジン方式（CPA）

設計ごとにユーザが定義する拡張命令に対応して再構成可能な RISC プロセッサを中心として，この拡張命令に対応した SIMD プロセッサアクセラレータまたはハードウェアエンジ

ンを比較的簡単に付加できるようにして，実時間の映像コーデックが実現できるプラットフォームである．汎用性を追及して再構成の自由度に重点を置いた例 [30] とアプリケーションをメディア処理に絞って専用コアを提供した例 [29] とがある．

図 3･26 に示した例では，コンフィギャラブル RISC コアと 128 ビットデータ幅までの SIMD 並列処理が可能な SIMD プロセッサを組み合わせた再構成可能なビデオ処理向けコアが提供される．SIMD 命令セットは，このコア内に格納されており，実行の際には命令が SIMD エクステンション部のコードキューに蓄積され，パイプライン実行される．符号化処理の場合，ここで DCT や MC など実行される．ME や CABAC には，SIMD とは別のハードウェアエンジンを付加することができる構成になっている．DMA 部には参照画像，現画像，再生画像などを効率よく SIMD やほかのエンジンに供給できる構成が組み込まれている．この SIMD 拡張命令を使用すれば，使用していない場合に比べ MPEG-4 や H.264 などのマルチメディア処理を 10 倍程度まで加速できるとしている．

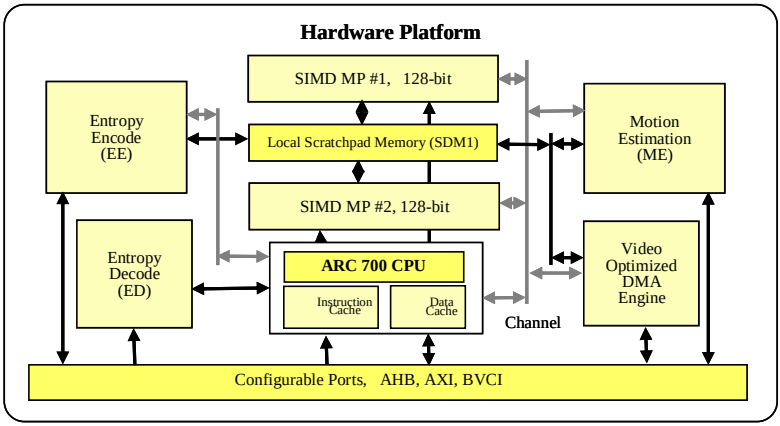


図 3・26　再構成可能なプロセッサコアをベースとした構成例

(5)　汎用プロセッサ（GPP：General Purpose Processor）による映像コーデック処理

事務処理や簡単な科学計算を実行するのに適した汎用マイクロプロセッサ上で映像や音声処理能力を拡大するアプローチとして，SIMD 拡張命令セット MMX（Multi Media eXtension）[31] やその延長上にある SSE（Streaming SIMD Extension）の導入とベクトルコプロセッサとの組合せがある．

MMX は命令セットアーキテクチャに SIMD 命令を拡張した x86 アーキテクチャのことを指すが，IBM の PowerPC ファミリでは AltiVec と呼ばれる同様な拡張を行っている．これらは汎用的なメディア処理を対象にしており必ずしも映像コーデックに特化したものではないこととアラインされていないデータやストリームデータに対するアドレス生成がサポートされていないため，(1)～(4)項で述べたコーデック専用アーキテクチャと比較すると効率面では大きく見劣りをする．しかし，圧倒的な普及と汎用性，それに支えられた GPP 自体の処理能力拡大とグリッドコンピューティングのようなネットワークを介した並列処理の発展によっ

て，MMX や SSE を使った実時間のソフトウェアコーデックの実現例が数多くみられるようになっている．MPEG-2 や H.264 の実時間 HDTV コーデックをクワッドコア搭載 PC 上で実現した例が報告されている [32]．

一方，MMX の非効率なメディアデータアクセスを改良する試みとして，GPP に密結合したベクトルプロセッサを付加する RSVP（Reconfigurable Streaming Vector Processor）[33] のような試みがあるが，アカデミックな試行が主体で実用化には至っていない．実用的には，グラフィックプロセッサ（GPU）にメディア処理機能を付加するアプローチが注目されている．今後，GPP と GPU が集積化される方向にあり，この方式が GPP による映像コーデック実現法の主流に発展する可能性がある．

(6) アーキテクチャのトレンド

実時間映像コーデック処理アーキテクチャ間のトレードオフ考察として，応用の広さ，開発コスト，半導体コスト，消費電力，開発環境，処理能力（効率），開発期間（Time-to-Market）の評価尺度で捉えるのが妥当と考える．応用の広さ（ソフトウェアによる機能拡張性，柔軟性）では，GPP が最も広く，DHB は最も狭い．一方，半導体コストや消費電力では，DHB が最有力で，GPP は最も不利になる．しかし，開発コストでは，H.264 エンコーダのように極めて複雑な処理を想定すると，DHB は妥当な期間でバグなしの開発完了が極めて困難という側面があって，GPP や MPA などに遠く及ばないという指摘がある．一方，H.264 の普及とともにハードウェア IP の流通が進みコストダウンできるのでその差は縮まっていくとも考えられる．開発環境では，GPP が安定したインフラが普及しているので最も有利である．DHB はもともとソフトウェアに大きく依存しないのと汎用の組込み CPU を利用する方向にあるので問題になることは少ない．MPA，特に CPA や VPA などは特定の企業からのサポートに頼らざるを得ないので開発環境という視点からは有利とはいえない．処理能力の観点では，DHB が最も高いが，GPP はコストを掛ければ，また MPA は並列処理の最適化と半導体の進歩により，HDTV の H.264 エンコード処理が手に届くところまできている．開発期間は，GPP が最も有利であり，MPA が後に続く．DHB は専用ハードウェアブロックの RTL 設計・検証が非常に重いために不利である．半導体コストと消費電力が重視される応用では DHB が有利ではあるが開発期間の長さが障害になりやすく，応用の初期段階ではほかの方式で立上げ，後で DHB に置き換えるという方法が現実的になっている．

以上のトレードオフ比較からいえることは，特定の応用における最適なアーキテクチャは単一の方式ではなく，その組合せで実現される方向にあるということと種々の符号化方式への対応や符号化の様々な周辺処理（イメージ処理，グラフィックス，高速通信処理など）への対応が重要視されるようになり，GPP と MPA 方式が重要な位置を占める方向にあるということであろう．GPP に DHB や HMA，CPA を組み合わせるようなアプローチも今後増えてくるものと思われる．このようなトレンドを考察するにあたって，アーキテクチャのトレードオフだけではなく映像コーデックのアルゴリズムの進展とトレードオフにも目を向けなければならないことはいうまでもない．

■参考文献

1) ISO/IEC 13818-1/2/3 International Standard, “Information technology － Generic Coding of Moving Pictures

and Ass0ciated Audio: Systems/Visual/Audio," Nov. 1994.
2) Ching-Yeh Chen, Shao-Yi Chien, Yu-Wen Huang, Tung-Chien Chen, Tu-Chih Wang, and Liang-Gee Chen, "Analysis and Architecture Design of Variable Block-Size Motion Estimation for H.264/AVC," IEEE Trans. Circuits and Systems, vol. 53, no. 2, pp.578-593, Feb. 2006.
3) T.Minami, T. Kondo, K. Suguri, and R. Kasai, "A Proposal of a one-dimensional Array Architecture for Full-search Block Matching Algorothm," IEICE Trans. Inf. & Syst. (Japanese Edition), vol. J78-D-1, no. 12, pp. 913-925, Dec. 1995.
4) S. Uramoto, A. Takabatake, M. Suzuki, H.Sakurai, and M. yoshimoto, "A Half-pel Precision motion estimation processor for NTSC-resolution video," IEICE Trans. Electron., vol.E77-C, no.12, pp.1930-1936, Dec. 1994.
5) 石原和也, 花見充雄, S. Scotznivosky, 松村哲哉, 吉田豊彦 他, "高画質対応 MPEG-2 動き検出 LSI（ME3）の開発," 電子情報通信学会技術研究報告, ICD98-116, pp.29-36, Aug. 1998.
6) E. Ogura, M. Takashima, D. Hiranaka, T. Ishikawa, Y. Yanagita, S. Suzuki, T. Fukuda, and T. Ishii, "A 1.2W Single-Chip MPEG-2 MP@ML Video Encoder LSI including Wide Search Range Motion Estimation and 81MOPS Controller," Digest of ISSCC, pp.32-33, Feb. 1998.
7) L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," IEEE Trans. Circuits Syst. Video Technology, vol. 6, pp. 419-423, Aug. 1996.
8) 笹島靖正, 大沢徹也, 廣部厚紀, 榎本忠儀, "中断法動きベクトル検出アルゴリズムの開発と動きベクトル検出アレイへの応用," 電子情報通信学会技術研究報告, ICD97-94, pp.1-8, Aug. 1997.
9) Z. He and M.-I. Liou, "Reducing hardware complexity of motion estimation algorithms using truncated pixels," in Proc. IEEE Int. Symp.Circuits Syst., pp. 2809-2812, 1997.
10) ISO/IEC ITU-T VCEG, Fast Integer Pel and Fractional Pel Motion Estimation for JVT, JVT-F017, 2002.
11) Kazuhito Suguri, Toshihiro Minami, Hiroaki Matsuda, Ritsu Kusaba, Toshio Kondo, Ryota Kasai, et al., "A real-time motion estimation and compensation LSI with wide search range for MPEG2 video encoding," IEEE Micro, vol.16, no.2, pp. 51-58, Apr. 1996.
12) 大塚竜志, 酒井潔, 浜野崇, 渡辺英明 他, "1 チップ MPEG-2 ビデオエンコーダ LSI の開発とシステム展開," 電子情報通信学会技術研究報告, ICD98-50, pp.1-8, Jun. 1998.
13) 水野正之, 柴山充文, 林直哉, 仙田裕三, 大井康, 民谷一郎, 山品正勝, "MPEG-2 符号化 LSI における履歴適応型動きベクトル探索とそのハードウェア実現," 電子情報通信学会技術研究報告, ICD97-163, pp.33-40, Oct. 1997.
14) ISO/IEC 14496-10:2003, "Information technology − Coding of audio-visual objects − Part 10: Advanced Video Coding," Dec. 2003.
15) S. Y. Yap and J. V. McCanny, "A VLSI Architecture for Variable Block Size Video Motion Estimation," IEEE Trans. Circuits and Systems, vol. 51, no. 7, pp.384-389, Jul. 2004.
16) Tung-Chien Chen, Yu-Wen Huang, and Liang-Gee Chen, "Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04), V-9-12, vol. 5, May 2004.
17) Y.W. Huang, T. C. Chen, C. H. Tsai, C. Y. Chen, T.W. Chen, C. S. Chen, C. F. Shen, S.Y. Ma, T. C.Wang, B.Y. Hsieh, H. C. Fang, and L. G. Chen, "A 1.3 tops H.264/AVC signle-chip encoder for HDTV applications," in Proc. Int. Solid-State Circuits Conf., 2005, pp. 128-129.
18) Y. Murachi, J. Miyakoshi, M. Hamamoto, T. Iinuma, T. Ishihara, F. Yin, J. Lee, H. Kawaguchi, and M Yoshimoto, "A Sub 100mW H.264 MP@L4.1 Integer-pel Motion Estimation Processor core MBAFF Encoding with Reconfigurable Ring-Connected Systolic Array and Segmentation-Free, Rectangle Access Search-Window Buffer," IEICE Trans. Electron., Vol. E91-C, No.4, pp.465-478, Apr. 2008.
19) Takayuki Onishi, Takashi Sano, Koyo Nitta, Mitsuo Ikeda, and Jiro Naganuma, "Multi-Reference and Multi-Block-Size Motion Estimation with Flexible Mode Selection for Professional 4:2:2 H.264/AVC Encoder LSI," IEEE International Symposium on Circuits and Systems (ISCAS 2008), pp.800-803, May 2008.
20) L. Li, S. Goto, and T. Ikenaga, "A Highly Parallel Architecture for Deblocking Filter in H.264/AVC," IEICE Trans. on Information and System, vol. E88-D, no. 7, pp.1623-1629, 2005.
21) S. Y. Shih, C. R. Charng, and Y. L. Lin, "A Near Optimal Deblocking Filter for H.264 Advanced Video Coding," Asia South Pacific Design Automation Conference, 2006, pp. 170-175

22) D. Marpe, T. Wiegand, and H. Schwarz, “Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard,” IEEE Transactions on Circuits and Systems for Video Technology, 13(7):620-636, Jul. 2003.
23) 粕谷滋, 永井律彦, 「2 値算術符号化装置」特願 2007-214068, 2007 年 8 月.
24) Mitsuo Ikeda, Toshio Kondo, Koyo Nitta, Kazuhito Suguri, Takeshi Yoshitome, Toshihiro Minami, Hiroe Iwasaki, Katsuyuki Ochiai, Jiro Naganuma, Makoto Endo, Yutaka Tashiro, Hiroshi Watanabe, Naoki Kobayashi, Tsuneo Okubo, Takeshi Ogura, and Ryota Kasai, “SuperENC MPEG-2 Video Encoder Chip,” IEEE Micro, vol. 19, no. 4, pp. 56-65, Jul. 1999.
25) Koyo Nitta, Mitsuo Ikeda, Hiroe Iwasaki, Takayuki Onishi, Takashi Sano, Atsushi Sagata, Yasuyuki Nakajima, Minoru Inamori, Takedhi Yoshitome, Hiroaki Matsuda, Ryuichi Tanida, Atsushi Shimizu, Ken Nakamura, and Jiro Naganuma, “An H.264/AVC High422 Profile and MPEG-2 422 Profile Encoder LSI for HDTV Broadcasting Infrastructures,” 2008 Symposium on VLSI Cicuits Digest of Technical Papers, pp.106-107, Jun. 2008.
26) S. Bewick, “An Advanced Media Processor Architecture for Consumer Applications,” Microprocessor Forum 2006, Oct. 2006.
27) J.W. van de Waerdt, S. Vassiliadis, et al., “The TM3270 media-processor,” In MICRO '05 Proceedings of the 38th International Symposium on Micro-architecture, pp. 331-342, Nov. 2005.
28) J. Leijten, “New Core Enables Programmable Camera Sensor Signal Processing,” Microprocessor Forum 2006, Oct. 200
29) Nigel Topham, “Efficient Multi-Processing for Media Applications,” Spring Pprocessor Forum 2006, May 2006.
30) Dennis Moolenaar, “A Dual-Core Video Decoder/Encoder,” Microprocessor Forum 2007, May 2007.
31) A. Peleg and U. Weiser, “MMX Technology extension to the Intel Architecture,” IEEE Micro, vol.16, no.4, pp.42-50, Aug. 1996.
32) Iwasaki, H., Naganuma, J., Ochiai, K., Endo, M. and Ogura, T., “Real-Time Software MPEG2 Video Encoder on Parallel and Distributed Computer System,” International Conference on Computer Communication (ICCC) 1999, pp.361-369, 1999.
33) J. Ciricescu, R. Essick, B. Lucas, P. May, K. Moat, J. Morris, M. Schuette, and A. Saidi, “The Reconfigurable Streaming Vector Processor (RSVP),” In MICRO'03: Proceedings of the 36th International Symposium on Micro-architecture, pp.141-150, Dec. 2003.

■10 群 － 5 編 － 3 章

3-2 画像処理・画像認識

（執筆者：京　昭倫）［2009 年 4 月 受領］

画像信号はほかの信号と比べ，処理すべきデータ（画素）が 2 次元状に配置されている点，及び画素数が多くの場合膨大である（数十万～数千万）点に特徴がある．画像処理・画像認識プロセッサはこうした特徴をうまく利用することで，汎用プロセッサよりも大幅に優れたコスト性能比を実現させようとするものであり，これまで多くの設計例がある．本節ではまず，画像処理・画像認識の処理的特徴についてまとめる．次に，そうした特徴の存在によって，汎用プロセッサの場合と比べどのような設計留意事項が存在するかについて説明する．最後に，これまでの実現例や今後の設計トレンドを説明する．

3-2-1 処理的特徴

画像に対する主要な信号処理には，写真・図形などの 2 次元画像やテレビ映像などの動画像の見た目の性質を操作することで画質を高めたり，あるいは逆にぼかしたりする画像処理（Image Processing）と，どのような物体・情景が映っているかといった画像の内容を把握する画像認識（Image Recognition）とに大きく分けられる．前者はディジタルカメラや TV モニタなどの付加価値を高める重要な機能であり，後者は工場ラインでの欠陥検査装置，知能ロボット視覚装置，交通監視装置，そして車両搭載される予防安全装置などの構築に必要な機能である．これらの機能の実現に向けた処理をステップに分けると，画像認識の場合は，低レベル処理→中レベル処理→高レベル処理に分解でき，そして画像処理はその中の低レベル処理のステップに対応付けられる．以下，これら低～高レベル処理のそれぞれがもつ一般的な処理的特徴についてまとめる（**図 3･27** 参照）．

- 低レベル処理では，画素を入力とし（撮像系に起因する）ひずみの補正やノイズ除去などの画質補正的な処理を行った後に，画素値を別の（複数）種類の特徴量（エッジ，テクスチャ，色情報，動き情報など画素特徴量と呼ぶ）に変換する．処理的特徴としては，通常全画素位置を対象とし，かつ互いに依存がない（並列度大），同一の処理を適用する（データ並列性），8～16 ビット整数演算が中心，参照データは対象画素位置からみて平面的（動画像なら空間的）近傍内に存在する（参照の面内近傍性大），などがあげられる．
- 中レベル処理では，（複数の）画素特徴量を入力とし，その存在位置を中心とする平面的（あるいは空間的）な近傍内での連結性，類似性，または所定の統計量による評価に基づき，多次元ベクトルとしての高次元特徴量に変換する．処理的特徴としては，画素特徴量同士に依存関係はないが所定基準値を超える画素特徴量のみが処理対象となるため，低レベル処理と比べると数桁程度並列度が低下する（並列度中），16～32 ビット整数演算が中心，各画素特徴量の位置や値に応じ相異なる処理パスをたどる（タスク並列性），低レベル処理と同様に参照データは平面的（あるいは空間的）近傍内に存在する（参照の面内近傍性大），などがあげられる．
- 高レベル処理では，高次元特徴量を入力（＝入力ベクトル）とし，多くの場合，識別器 [1)~3)] を用いて辞書データ（参照ベクトル群）との距離計算を行い，そのうえ対象物体モデルや拘束条件などの先験的知識により検定を行うことで，最終的な判断としての認識結果を出力す

る．処理的特徴としては，入力ベクトル同士は互いに独立，入力ベクトル数は中レベル処理でのそれと同程度だが個々の次元数が大きく増加する（並列度大），32 ビット以上の整数演算が中心，全入力ベクトルに同一の処理を適用するフェーズ（データ並列性）と，入力ベクトルごとが相異なる処理パスを通るフェーズ(識別率向上目的で識別器を多段カスケード接続する場合など）とが存在（タスク並列性），参照データに特に面内近傍性は見られない（参照の面内近傍性小），などがあげられる．

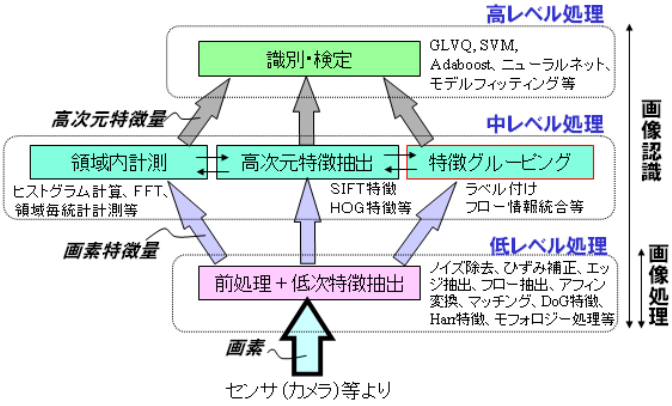


図 3・27　画像処理・画像認識の処理ステップ

3-2-2 画像処理・画像認識 LSI の設計に向けた留意事項

表 3･2 に示すように，中レベル処理が低レベル処理と比べほぼ相反する性質をもつのに対し，高レベル処理は低レベル処理と中レベル処理の双方の性質を併せもつ．このように，低レベル処理が主たるターゲットの画像処理プロセッサの場合，低レベル処理がもつ処理的特徴がそのまま設計上での留意事項となるのに対し，低～高レベル処理の全般がターゲットとなる画像認識プロセッサの場合，相反する幾つかの処理的特徴を同時に設計上で考慮するための工夫が必要になる．

表 3・2　画像処理・画像認識の処理的特徴のまとめ

	低レベル処理	中レベル処理	高レベル処理
処理の複雑度	8～16 ビット整数	16～32 ビット整数	32 ビット整数以上
並列性の規模	大	中	大
並列性の性質	データ並列	タスク並列	データ＋タスク並列
参照の面内近傍性	大	大	小
具体的な処理例	平滑化，中間値フィルタ，エッジ抽出，テンプレートマッチング，フロー抽出，アフィン変換，DoG 特徴，Harr 特徴など	ラベル付け，フロー情報統合，ヒストグラム計算，キーポイント抽出，FFT など	SVM[2]，GLVQ[3]，Adaboost[1]，ニューラルネット，モデルフィッティングなど

表 3･3 に，汎用プロセッサ設計と対比させる形で，前記の処理的特徴を考慮した場合の画像処理・画像認識 LSI それぞれの設計留意事項をまとめる．

表 3・3　汎用プロセッサと対比させた場合の画像処理・画像認識 LSI の設計留意事項

項目名	汎用プロセッサの場合	画像処理 LSI の場合	画像認識 LSI の場合
メモリアクセス制御部	セットアソシアティブ方式キャッシュ制御	「参照の面内近傍性」を活用するには，アドレス値の連続性だけで局所性を判定する既存のキャッシュ制御方式では効率が悪いため，他方式の検討が必要．	
演算器の機能	32 ビット整数，倍精度浮動小数	8～16 ビット整数 浮動小数点は基本的には演算不要．	16～32 ビット整数 浮動小数点演算は基本的には不要．
命令並列性利用方法	スーパースカラ方式＋アウトオブオーダ命令発行	データ並列性とタスク並列性の顕著さと比べ，高い並列度が期待できない命令並列性の活用優先度は低くなる．そのため，活用する場合でも実現コストがあまりかからない方式（例えば VLIW 方式）を検討すべき．	
データ並列性利用方法	4～16 並列の SIMD 命令拡張	単純な SIMD 命令拡張は，そのままでは「参照の面内近傍性」をもつ多くの低レベル処理に対し有効でないため，SIMD 命令拡張の更なる高度化・高並列化も含め，より積極的にデータ並列性を活用できる方式を検討すべき．	
タスク並列性利用方法	ハイパースレッディング	そもそもタスク並列性が顕著でないため不要．	中～高レベル処理がもつタスク並列性の顕著さを考えると，より積極的な活用方式を検討すべき．

3-2-3 画像処理・画像認識 LSI の実現例

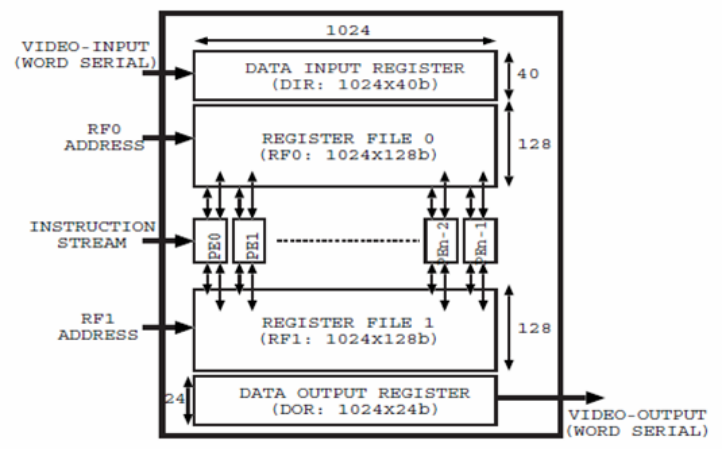
画像処理 LSI は専用回路として設計するケースと，多様な機能を柔軟に実現するために並列プロセッサとして設計するケースとがあり，後者の場合，低レベル処理がもつ高いデータ並列性と参照の面内近傍性の処理的特徴を低コストで活用するのに適したパイプライン方式や SIMD 方式に基づく設計例が多く，一方画像認識 LSI は，認識対象や認識アルゴリズムの多様性から，専用回路よりも並列プロセッサとして設計する例が多いが，並列プロセッサ設計の場合，設計に際しどの処理的特徴の高速化を重視するかでパイプライン方式（低レベル処理重視），SIMD 方式（低レベル処理重視），MIMD 方式（中～高レベル処理重視），そして専用回路＋SIMD や SIMD + MIMD といったように複数方式の組合せ(全体のバランス重視)，に基づくものとに大きく分けられる．

- 専用回路方式に基づくものでは，低レベル処理に属するエッジ抽出や雑音除去のような空間フィルタ処理に特化した構成をもつもの [4] や，空間フィルタ処理に加えテンプレートマッチング用の正規化相関回路やヒストグラム計算用回路なども合わせて集積したもの [5] などが例としてあげられる．画像認識向けでは，特定の画像認識アルゴリズム全体に特化させることで低消費電力化を狙うものが提案されている [6]．
- パイプライン方式に基づくものでは，単一演算から関数レベルまで，処理をパイプライン的に行えるように演算要素（PE）を直列に接続したもの [7),8)]，PE 間をネットワーク結合させたもの [9)]，データの待ち合わせを動的に行わせるもの [10)] などの設計例があげられる．パイプライン各段の処理負荷をほぼ均等にできるケースでは，ハードウェアリソースを非常に効率よく利用できる有効な方式といえる．

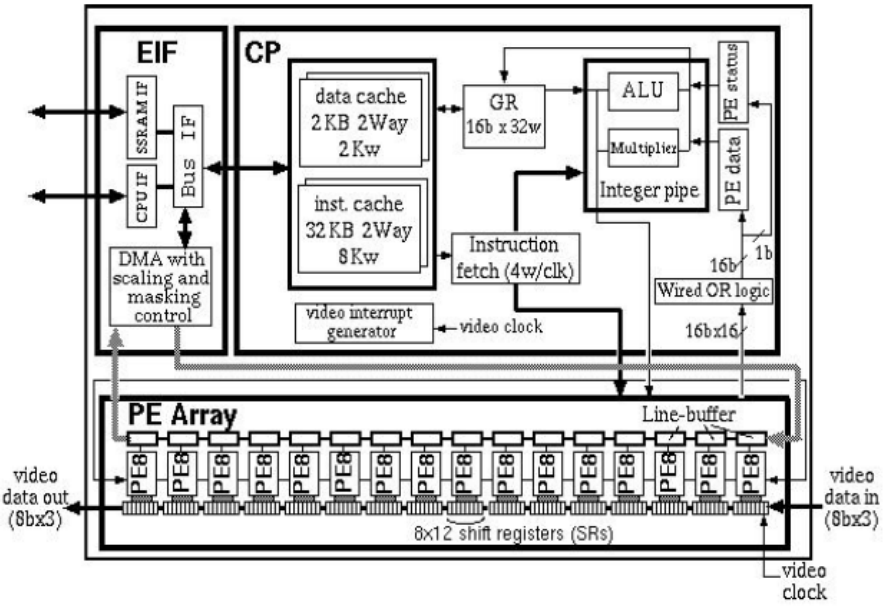
- SIMD 方式に基づくものは，多数の PE が同期して相異なるデータを対象に一斉に同一の処理を行う．PE 数によらず制御ユニットは一つだけで済むため構造的に低コストであることから，低レベル処理が有する大量なデータ並列性を非常に効率よく活用できる構成方法である．特に 1 次元状に PE を結合させた設計例が多く，スキャンライン単位で入力されてくる画像をそのまま処理するタイプのもの [11), 12)]，個々の PE に数 K バイトのメモリと動作やアドレッシングの自律性などをもたせることで，一部の中～高レベル処理にも対応可能にしたもの [13), 14)] などに分けられる．
- MIMD 方式に基づくものは，個々の PE にそれぞれ独立した制御ユニットやプログラムメモリが存在し，各 PE がそれぞれ独立に動作可能であるため，「タスク並列性」を有する中～高レベル処理に対し適合性が高い [15), 16)]．ただし，そのままでは低レベル処理に対しては SIMD 方式と比べると個々PE の実現コストが数倍も高くなることから，低レベル処理対策として SIMD 命令拡張 [17)]PE を利用するケース [18)] も幾つか見られる．
- 専用回路＋SIMD 方式に基づくものには，実現する画像認識アルゴリズムの種類を固定化させることで，中～高レベルに対応する処理は専用回路で実現しつつ，低レベル処理の一部分にある程度の柔軟性を残すために SIMD 方式の並列プロセッサを採用した構成のものが幾つか提案されている [19), 20)]．現状，標準的とされる画像認識アルゴリズムがまだ存在しないため，こうしたアプローチはある程度のリスクをともなうが，所定の高い認識性能を低電力で実現したい場合では一つの選択肢となる．
- SIMD＋SIMD 方式に基づくものには，MIMD 方式のプロセッサに SIMD 動作モードを追加したもの [21)]，SIMD 方式と MIMD 方式の 2 種類のプロセッサを階層結合させたもの [22)]などがあり，低レベル処理のみならず，中レベル以上の処理にも対応可能な構成である点に利点があるが，コスト的に高くなるという問題点がある．これに対し，例えば SIMD 方式の並列プロセッサを MIMD 方式にも動作切り替え可能とする代わりに，MIMD 方式実行時は並列度を下げることでコスト性能比の低下を防ぐアプローチ [23)] などが提案されている．

上記実現例のうち，画像処理プロセッサの製品化例として，コピー機や TV モニタなどでの画質改善目的で利用された TI 社の SVP（Serial Video Processor）[11)]，また画像認識プロセッサの製品化例として，自動車の衝突防止システムでの障害物検出機能を実現する用途で利用された NEC エレ社の IMAPCAR（Integrated Memory Array Processor for CAR）[14)] のブロック構成を**図 3･28** に示す．

SVP は SIMD 方式に基づき，スキャンライン単位で入力されてくる画像をそのまま処理するタイプの画像処理プロセッサである．SVP は図 3･28(a) を参照すると，ビデオデータの入力部（DIR）と出力部（DOR）がそれぞれ 1024 ワード×40 ビットと 1024 ワード×24 ビットのデュアルポートメモリで構成されており，ビデオ画像の入出力と PE からのアクセスを並列に行えるようになっている．演算部は 1024 ワード×128 ビットのレジスタファイル 0 と 1（RF 0 と RF 1）に加え，1 ビット ALU と四つのワークレジスタ（M, A, B, C）のセットを計 1024 個（PE 0～PE 1023）含む構成をもつ．命令は外部より供給される．その他，各 PE はその隣接 4 PE までと直接データ交換可能であるように内部結線されている．こうした構成のもと，各 PE はそれぞれに対応して存在する 128×2 ビット容量のレジスタ（RF 0 と RF 1）を演算の作業領域として利用しつつ，DIR よりスキャンラインごとに入力されてくる画像をその場で処理しては，処理結果を DOR にそのまま外部出力する．



(a) SVP



(b) IMAPCAR

図 3・28　画像処理・画像認識プロセッサ製品例のブロック構成図

一方，IMAPCAR は図 3･28(b) に示すように，個々が 2 KB のローカルメモリをもつ PE 計 128 個からなる PE アレイ部，PE アレイ部へ命令などを供給する制御プロセッサである CP 部，そして外部メモリインタフェース回路や DMA 回路などを含む EIF 部とからなる．SVP

が行単位の低レベル画像処理をターゲットとしているのに対し，同じ SIMD 方式に基づきながら画像認識応用をターゲットとしている IMAPCAR は，低レベル処理のみならず，一部の主要な中～高レベル処理にも対応可能となるように，PE ごとが動作を行うか否かという「動作の自律性」，及び PE ごとが相異なるメモリアドレスにアクセスできるという「アドレッシングの自律性」をもつように構成されている．これら 2 種類の自律性の付加により，画像をライン単位で処理すればよいケース（**図 3･29**(a)）にとどまらず，PE アレイによる並列的な画素参照についてはより多様な順序性を実現できるようになる [24]（図 3･29(b)～(d)）．そうした工夫のもとで，実現コストの低い SIMD 方式に基づきつつも中レベル処理に分類される幾つかの主要な処理で必要となる画素参照パターンの制約を満たしながら，並列に処理が行えるようになった．

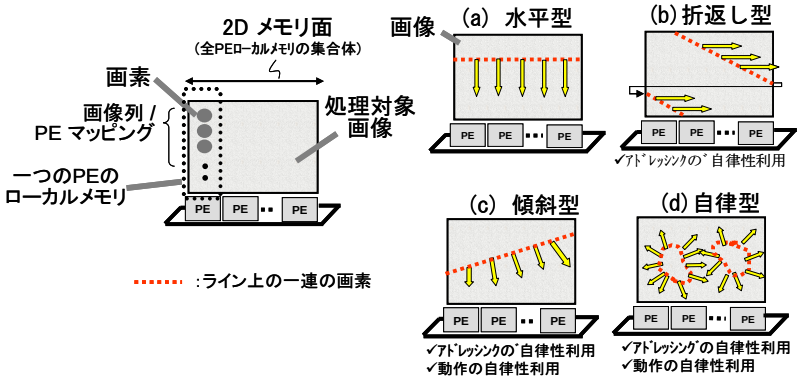


図 3・29　動作とアドレッシングの自律性の利用により実現可能となる画素参照パターン例

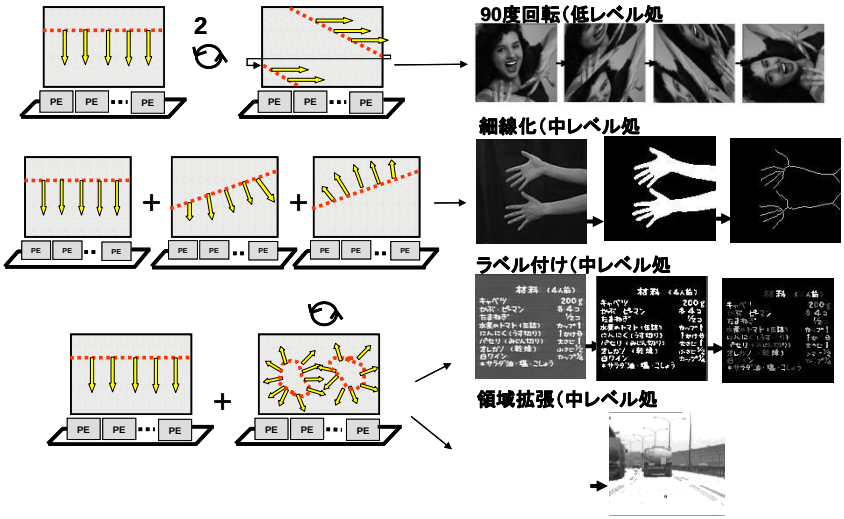


図 3・30　動作とアドレッシングの自律性を活用した低～中レベル処理の並列化例

低～中レベル処理に分類される幾つかの画像処理・画像認識タスクについて，IMAPCAR を利用した場合の並列化のイメージを**図 3･30** に模式的に示す.

3-2-4 今後の展望

画像処理･画像認識 LSI の設計に際しては, 画像処理･画像認識がもつ処理的特徴に対し, いかにアーキテクチャ的適合性を高めるかが大きなポイントとなる．画像処理 LSI は今後, より大きな画像サイズへの対応と共に，その低コスト実現手法のさらなる追求が行われるだろう．一方，画像認識 LSI は発展初期では大量の画素の処理を必要とする低レベル処理の処理時間が支配的だったが, 画像処理技術の進展にともないそうした課題も解決されつつあり, 最近では中～高レベル処理の処理時間全体に占める比率が高くなりつつある．そこで，今後は低～高レベル処理全体に対しアーキテクチャ的適合性の高いものをいかに実現していくかが，画像認識 LSI の研究開発における焦点となると予想される.

■参考文献

1) P. Viola, M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” IEEE Computer Vision and Pattern Recognition Conference (CVPR), pp.I-511-518, 2001.
2) C. Cortes, V. Vapnik, “Support-Vector Networks”, Machine Learning,” vol.20, no.3, pp.273-297, 1995.
3) A. Sato, K. Yamada, “Generalized Learning Vector Quantization,” Advances in Neural Information Processing Systems, vol.8, pp.423-429, MIT Press, 1996.
4) 伊藤潔人, 小川　誠, 柴田　直, “フラッシュコンボリューション型画像フィルタ演算プロセッサ,” 信学技報, ICD2003-28, pp.13-18, 2003.
5) 村松彰二, 小林芳樹, 大塚裕史, 清水英志, 今井聡彦, 広瀬健二, 菊池博幸, “IP5000 ビジョンシステムと高速化を実現した画像処理機能,” 第 5 回画像センシングシンポジウム, pp.45-50, 1999.
6) Y. Hanai, Y. Hori, J. Nishimura and T. Kuroda, “A Versatile Recognition Processor Employing Haar-Like Feature and Cascaded Classifier,” IEEE International Solid-State Circuits Conference (ISSCC), pp.148-149, 2009.
7) P. P. Jonker, E. R. Komen, “A scalable real-time image processing pipeline,” Proc. of IAPR Conf. on Pattern Recognition (ICPR’92), vol.4, pp.142-146, 1992.
8) S. M. Chai, D. S. Wills, “Systolic opportunities for multidimensional data streams,” IEEE Transactions on Parallel and Distributed Systems, vol.13, no.4, pp.388-398, 2002.
9) 直井　聡, 古明地正俊, 太田善之, 尾崎　暢, 佐々木繁, 後藤敏行, 吉田真澄, “構造可変型ビデオレートカラー画像処理システム「韋駄天」,” 信学論(D), vol.J73-D2, no.10, pp.1751-1760, 1990.
10) T. Temma, T. Iwashita, K. Matsumoto, H. Kurokawa, and T. Nukiyama, “Data Flow Processor Chip for Image Processing,” IEEE Trans. on Electron Devices, vol.ED-32, no.9, pp.1784-1791, 1985.
11) J. Childers, P. Reinecke, H. Miyaguchi, S. Yamamoto, Y. Takahashi, Y. Yaguchi, and M. Takeyasu, “SVP: Serial Video Processor,” IEEE 1990 Custom Integrated Circuits Conference, 17.3, pp.1-4, 1990.
12) R. P. Kleihorst, A. A. Abbo, A. van der Avoird, M. J. R. Op de Beeck, L. Sevat, and P. Wielage, “Xetal: A Low‐power High‐Performance Smart Camera Processor,” IEEE Int. Symp. Circuits System, vol.5, pp.215−218, 2001.
13) D. W. Hammerstrom and D. P. Lulich, “Image Processing Using One-Dimensional Processor Arrays,” Proceedings of the IEEE, vol.84, no.7, pp.1005-1018, 1996.
14) S. Kyo, S. Okazaki, T. Koga, F. Hidano, “A 100 GOPS In-vehicle Vision Processor for Pre-crash Safety Systems Based on a Ring Connected 128 4-Way VLIW Processing Elements,” Digest of Technical Papers of Symp. on VLSI Circuits, pp.28-29, 2008.
15) K. Deguchi, K. Tago, I. Morishita, “Integrated parallel image processings on a pipelined MIMD multi-processor system PSM,” Proc. of 10th International Conference on Pattern Recognition, vol.2, pp.442-444, 1990.

16) T. Matsuyama, N. Asada, M. Aoyama, “Parallel Image Analysis on Recursive Torus Architecture,” Proc. of Second IEEE Workshop on Computer Architecture for Machine Perception (CAMP), pp.202-214, 1993.
17) S. K. Raman, V. Pentkovski, J. Keshava, “Implementing streaming SIMD extensions on the Pentium III processor,” IEEE Micro, vol.20, no.4, pp.47-57, 2000.
18) Y. Kondo et al.: “4 GOPS 3 way-VLIW image recognition processor based on a configurable media-processor,” ISSCC Digest of Technical Papers, pp.148-149, 2001.
19) Chih-Chi Cheng, Chia-Hua Lin, Chung-Te Li, Samuel Chang, Chia-Jung Hsu, and Liang-Gee Chen, “iVisual: An Intelligent Visual Sensor SoC with 2790fps CMOS Image Sensor and 205GOPS/W Vision Processor,” ISSCC Digest of Technical Papers, pp.306-307, 2008.
20) Kwanho Kim, Seungjin Lee, Joo-Young Kim, Minsu Kim, Donghyun Kim, Jeong-Ho Woo, and Hoi-Jun Yoo, “A 125GOPS 583mW Network-on-Chip Based Parallel Processor with Bio-inspired Visual-Attention Engine,” ISSCC Digest of Technical Papers, pp.308-309, 2008.
21) H. J. Siegel, L. J. Siegel, F. C. Kemmerer, P. T. Mueller Jr., H. E. Smalley Jr., and S. D. Smith, “PASM: A partitionable SIMD/MIMD system for image processing and pattern recognition,” IEEE Trans. on Computer, vol.C-30, pp.934-947, 1981.
22) C. C. Jr. Weems, “The Second Generation Image Understanding Architecture,” Proc. of Second IEEE Workshop on Computer Architecture for Machine Perception (CAMP), pp.276–285, 1993.
23) S. Kyo, T. Koga, H. Lieske, S. Nomoto, and S. Okazaki, “A Mixed-Mode Parallel Processor Architecture for Embedded Systems, ACM Int. Conf. on Supercomputing (ICS), pp. 253-262, 2007.
24) S. Kyo, S. Okazaki, and T. Arai, “An Integrated Memory Array Processor for Embedded Image Recognition Systems,” IEEE Trans. on Computers, vol.56, no.5, May 2007.

■10 群 － 5 編 － 3 章

3-3 音　声

（執筆者：天野明雄）［2009 年 12 月 受領］

携帯電話はその登場時点と比べて大幅に小さくなり，連続通話時間も格段に伸びた．更にインターネット接続機能をはじめとして，カメラ機能，ワンセグ TV 視聴機能など多彩な機能が搭載されてきている．これらは携帯電話用のプロセッサをはじめとする半導体技術の進歩の賜物である．また，携帯型音楽プレーヤの普及も目覚しい．音楽プレーヤの場合は携帯電話と比べると音楽再生機能に特化され，特に小型化が進んでいる．更に，最近の携帯電話，携帯型音楽プレーヤでは通話音，音楽再生音の品質向上を求めてノイズ抑圧機能を搭載したり，複数マイクを装備するものも出てきている．一方で通信回線の広帯域化にともない，音声，映像共に圧縮せずに送信するようなテレビ会議／音声会議システムなども普及し始めている．こうしたテレビ会議／音声会議では拡声通話が前提となるため，エコーキャンセラも必要となる．また，多数のマイクを搭載して周囲ノイズを抑圧したり，またスピーカもアレイ化することにより相手音声の聞こえる方向を制御する会議システムも出てきている．

本節ではこれら多岐にわたる音声関連処理について，音声符号化，音楽符号化，音声処理（音声認識，音声合成，ノイズキャンセル，マイクアレイなど）などの機能の実装，省電力化の観点から解説する．

3-3-1 音声符号化

携帯電話がディジタル化された当初から，音声符号化方式としては CELP（Code Excited Linear Prediction Coding）を基本とする方式が採用されている．**図 3･31** に CELP の基本構成を示す．合成フィルタは入力音声の線形予測分析結果から求め，合成フィルタを駆動するための音源情報は符号長の中から最適なものを選択することによって得る．符号化時には符号長からの最適符号選択処理をし，復号化時は得られた符号に基づいて合成フィルタにより音声を再生する．

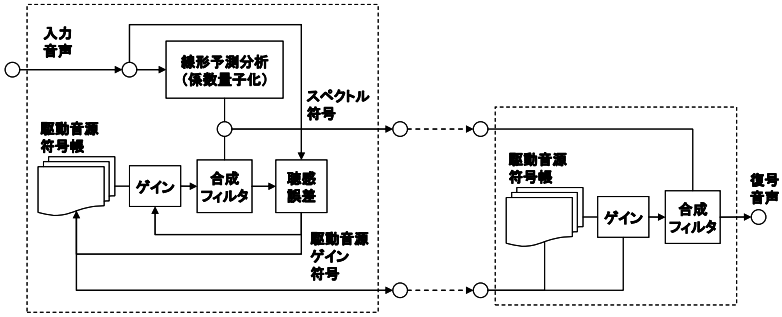


図 3・31　CELP の基本構成

音声符号化，復号化に必要な所要処理量は数十 MIPS もかからず，専用 LSI を用いることなく携帯電話向けの汎用 DSP を用いてソフトウェアで実装されるのが一般的である．

当初はフルレートの VSELP で 11.2 kbps，ハーフレートの Psi-CELP で 5.6 kbps の伝送レートとなっており，2.5G，3G に入ってからは AMR，EVRC などベースの符号化方式は 2G と同様であるが，更に改良が施され，更に低いビットレートあるいは可変レートで同等の品質の音声伝送がなされるようになってきている．

最近の携帯電話では音声通話機能以外にインターネット接続機能，カメラ機能，決済機能などと多彩な機能を搭載するようになっている．**図 3･32** に携帯電話の一般的な構成ブロック図を示す．こうした多彩な機能の実行のために CPU と DSP を搭載するのが一般的な構成であり，音声符号化の処理は DSP で実行されるのが一般的である．

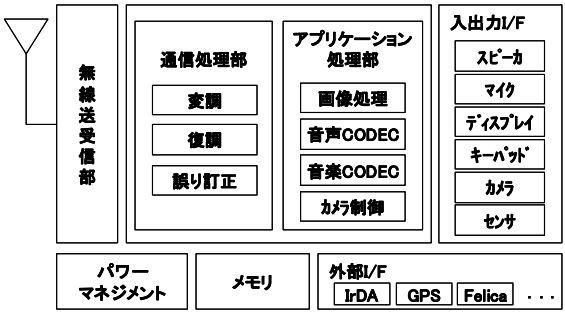


図 3・32　携帯電話の構成ブロック図

音声符号化は携帯電話ばかりでなく IP 電話などでも使われる．IP 電話で使われる音声符合化は G.711（PCM），G.723（ADPCM），G.729（CS-ACELP）などである．IP 電話機能は PC 上のソフトで実現できる．最近では PC に接続しなくても直接使える IP 電話端末も出てきている．こうした IP 電話端末では無線 LAN 機能を搭載し，無線 LAN にて IP 接続して通話する．この場合の構成は図 3･32 の構成のうち，無線送受信部と通信処理部を無線 LAN 接続機能に置き換え，カメラ制御や画像処理を削除したような構成となる．

3-3-2 音楽符号化

2001 年の iPod 登場以来携帯型音楽プレーヤの普及には目覚しいものがある．携帯型音楽プレーヤでは MP3 に代表される音楽符号化方式が使用されている．音楽符号化では音声符号化の場合とは異なり人間の聴覚特性を利用することにより音楽データの圧縮を行う．**図 3･33** に MP3 の基本構成図を示す．

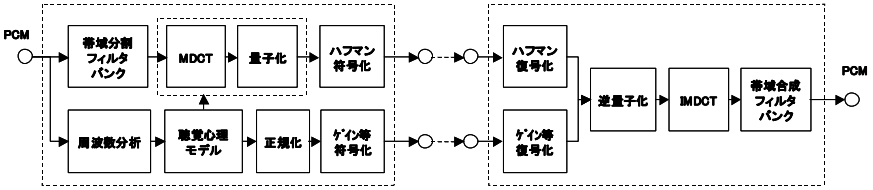


図 3・33　MP3 の基本構成図

携帯型音楽プレーヤの場合，携帯電話の場合とは異なり，符号化回路を搭載する必要はなく，復号化回路（デコーダ）のみ搭載すればよい．フィルタバンク，MDCT などいわゆる周波数分析処理が中心となり，積和演算の回数が多く必要となるが，画像処理などに比べると所要演算量ははるかに低く，組込み向けプロセッサをコアとし，オーディオ出力周辺回路を組み込んだ，数 mm 角の大きさの低消費電力専用 LSI が構成され，携帯型音楽プレーヤなどで使われている．**図 3･34** に携帯型音楽プレーヤの一般的なハードウェア構成ブロック図を示す．

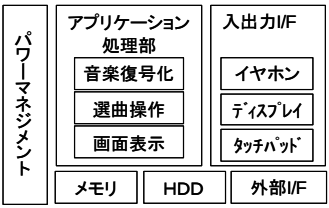


図 3・34　携帯型音楽プレーヤの一般的なハードウェア構成ブロック図

音楽符号化には MP3 のほかに，AAC（Advanced Audio Coding），ATRAC（Adaptive Transform Audio Coding），WMA（Windows Media Audio）などがあるが，データ圧縮の基本的な原理は同様である．複数種類の音楽ファイルフォーマットに対応してもなお低消費電力で小型の専用 LSI の構成が可能である．図 3･34 の構成でもメモリスティック程度の大きさになってきている．携帯電話に音楽再生機能を搭載する場合に，携帯電話のメインのプロセッサに負荷をかけないようにするために専用 LSI としてメインプロセッサからの制御で動作するような音楽再生 LSI も登場している．携帯型音楽プレーヤの場合，バッテリーパックも交換しないことを前提に，半田で直付けした作りとなっているものが多い．

携帯電話，携帯型音楽プレーヤのように小型低消費電力な組込み応用が進展する一方で，薄型大画面 TV の普及やプロジェクタの低価格化，DVD，BD の普及により家庭でも映画館並み，あるいは映画館以上の臨場感をもった映画を楽しむことができるようになってきている．こうしたいわゆるホームシアターに対応するためのオーディオ機器の普及も始まっている．HD-DVD, HD-Audio に対応するためのオーディオ機器では 5.1 ch サラウンドのみならず，7.1 ch，11.2 ch 対応なども出てきており，対応するオーディオフォーマットとしてもドルビーディジタル，DTS（Digital Theater System），ドルビーTrueHD などが出てきている．こうしたホームシアターでのオーディオ機器の高級機の価格は数十万円から百万円にも達し，マルチチャンネルでの音場制御などに対応するために数百 MHz 級の DSP を数チップ搭載するような構成となっている．

3-3-3 音声処理

音声処理といっても範囲は多岐にわたる．ここでは音声認識，音声合成，EC（エコーキャンセル）NC（ノイズキャンセル）を取り上げ，それらについて機器実装の観点から簡単に触れる．最近ではマイクを複数搭載するマイクアレイ処理が TV 会議システムや電話会議システムに採用されてきているが，マイクアレイ処理の実装についても簡単に触れる．また，マイク自体は回路アーキテクチャとはいえないが，従来のエレクトレットコンデンサマイク

（ECM）に代わって，最近採用が増えているシリコンマイクについても簡単に触れる．

(1) 音声認識・音声合成

音声認識は入力された音声をテキストに，音声合成は入力されたテキスト（漢字かな混じり文）を音声に変換する処理である．**図 3･35** に音声認識の処理構成図を示す．

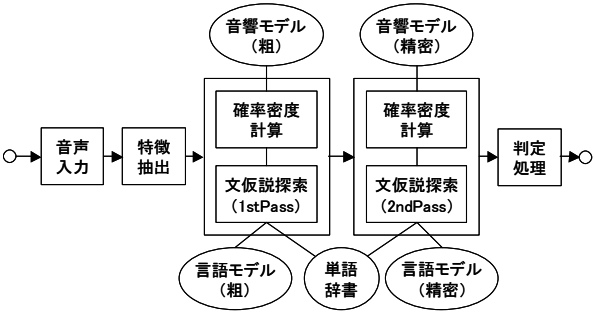


図 3・35　音声認識の処理構成図

図 3･35 中で最も処理量を要するのが仮説探索処理部である．仮説探索処理部は膨大な仮説空間の中から正解テキストを探索する処理であり，図 3･35 に示すように 2 パス構成とすることが多く，第 1 パスの処理量の少ない粗い探索にて仮説を絞り込み，その後，第 2 パスにて精密な探索を実施する．1990 年代のハードウェアでは複数のワークステーションをネットワークで接続し，各ワークステーションに音声分析処理，確率計算処理，仮説探索処理などを分担して初めてリアルタイムでの音声認識処理が実現されていた．2000 年代になるとワークステーションあるいは PC 1 台でこうした処理が実現されるようになってきた．最近ではノート PC 1 台でリアルタイムの音声認識が実現できるようにまでなってきている．

しかしながら，カーナビ，ゲーム機，携帯電話などの組込み型機器に音声認識機能を搭載しようとすると語彙サイズを制限するなどの制約が必要となる．カーナビでの目的地設定などを想定すると必要な語彙は地名，駅名など目的地を設定するのに必要な語彙に制限され，また，言い回しも目的地設定に適した言い回しに制限される．こうした状況では図 3･35 の単語辞書のサイズ及び言語モデルの複雑度が大分制限され，組込み向けプロセッサのフルパワーの 20～30 %程度の演算量で実時間音声認識が可能となる．2G の当初の携帯電話でも電話をかける相手の名前を音声で発生すると発信できる音声ダイアル機能が多数搭載された．この場合，対象となる語彙は電話帳に登録されている名前のみであり，大幅に語彙が制限される．

また，携帯電話端末では音声認識処理の中の特徴抽出処理までを実施し，この結果をパケットに載せてサーバに送り，音声認識処理の中で処理量の多くかかる仮説探索処理はサーバ側で実施し，結果を端末に返すという実装方式もある．この方式を分散型音声認識（Distributed Speech Recognition）といい，ETSI（The European Telecommunications Standard Institute）で 2000 年頃この方式が標準化されており，2006 年頃から一部のキャリアで実際のサービスも開始されている．**図 3･36** に分散型音声認識の構成を示す．メーカによっては携帯電話用のマルチコアアプリケーションプロセッサのマルチコアの構成を音声認識処理の各

処理に均等な負荷としてかかるように構成し，携帯電話単体でフル機能の音声認識の実現へ向けて開発している例もある．

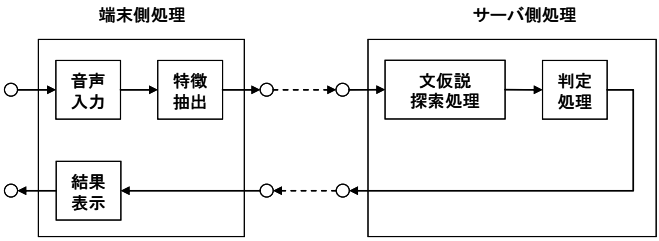


図 3・36　分散型音声認識の構成図

図 3･37 に音声合成の処理構成図を示す．メモリ容量，CPU の処理能力が不十分であった 1990 年代では音声素片を分析パラメータで保持する分析合成方式が主流であり，合成される音声として鼻がつまったようなロボット音声感が避けられなかった．メモリがふんだんに使え，プロセッサの処理能力が飛躍的に向上した現在では音声素片を波形データのまま保持する素片合成方式が主流となっている．しかもメモリ容量の拡大により，同じ音素を表す素片でも前後の音素環境に依存して複数の素片をもつこと，また複数の候補から最適な音声素片列をリアルタイムで選択することが可能となってきており，ノート PC クラスの処理能力で肉声にかなり近い品質の音声を合成することができるようになってきている．

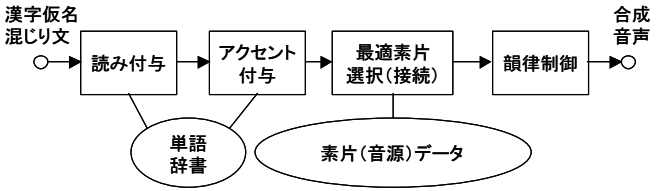


図 3・37　音声合成の処理構成図

図 3･37 中，音源データにどれだけの種類の音声素片をもつか，また，それらを接続して最適素変系列を探索する処理にどれだけの CPU パワーを割くことができるかによって，合成される音声の品質に差が出てくる．カーナビやゲーム機のような組込み型の機器の場合，PC ほどふんだんにメモリ量や CPU パワーを割くことができないので，可能なメモリ容量や音声合成に割くことのできる CPU パワーを考慮して音源データサイズを低減せねばならない．合成音声の品質と音源データサイズの間にはトレードオフの関係がある．PC 搭載を考えた場合には数百 MB の音源データが容易に搭載されるが，カーナビやゲーム機などの組込み機器の場合は数十 MB から数 MB 程度まで音源データを低減する必要が出てくる．

(2)　ノイズキャンセル，エコーキャンセル

携帯電話や携帯型音楽プレーヤが普及し，街中や電車の中で通話したり，音楽を聴いたりすることが多くなってきた．また，会議録作成などのための IC レコーダも広く使われるよう

に.なってきた．このような様々な場面での通話，音楽聴取，録音において高音質化することへのニーズも高いし，プロセッサの発展によりノイズキャンセル機能の搭載も容易になってきた．ノイズキャンセルの最も基本的な方式は，スペクトルサブトラクション方式である．入力信号に含まれているノイズのスペクトルを予測し，入力信号からそのノイズ成分を周波数領域において差し引くような処理を施す．本処理は音声符号化処理や音声認識処理における周波数分析処理，特徴抽出処理の一部分として構成され，処理量の増加もさほど大きくない．ノイズスペクトルの予測には，音声入力用のマイクを兼用する場合と，ノイズ専用マイクを用意して 2 ch マイクとする場合があるが，処理内容，処理量としては大きく変化することはない．最近では携帯電話や携帯型音楽プレーヤでもノイズ推定用のマイクが搭載される場合が多い．

1999 年の道交法改正により運転中の携帯電話の使用について危険を生じた場合に罰則が科されるようになり，更に 2004 年の改正により危険の有無に関係なく罰則が科されるようになり，より罰則が強化された．こうした状況を踏まえ自動車内での通話のためのハンズフリー通話のニーズが高まった．ハンズフリー通話の場合，口元とマイクの間に距離が生じ，またスピーカも受話器の場合と異なり耳との間に距離が生ずる．こうした拡声通話の場合には音響エコーキャンセルが必須となる．拡声通話の場合，スピーカから発生される相手側の音声をマイクが拾って相手側にエコーとして戻してしまい，通話に支障が生じるばかりでなく，悪い場合にはハウリングを起こしてしまう．こうした場合には**図 3･38** に示すようなエコーキャンセル回路を必要とする．

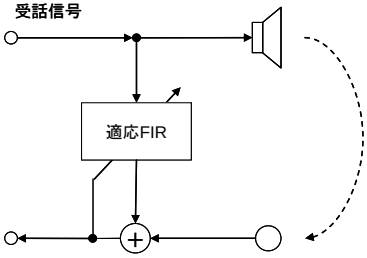


図 3・38　エコーキャンセル回路

スピーカからマイクまでの伝達関数を推定し，これに基づいてマイクが拾うエコーを適応フィルタの出力により差し引いてエコーを抑圧する．音響エコーキャンセル回路は DSP とプログラムという構成でも実現できるが DSP と CODEC 回路など必要コンポーネントを 1 チップ化して音響エコーキャンセル専用 LSI として構成される場合も多い．また，組込み向け CPU コアも内蔵してノイズキャンセル機能も合わせて搭載される場合も多い．これらの LSI はハンズフリー通話用としてカーオーディオシステムの追加コンポーネント，あるいはカーナビシステムの追加コンポーネントとして自動車内に実装される．

(3)　マイクアレイ

ブロードバンド化が進み，TV 会議システム，音声会議システムでも映像の高精細化，音声の高音質化が進んできた．従来の TV 会議システムでは相手の表情を読み取ることなどは

困難であったが，高精細化にともない，表情を読むといったことも可能になってきている．音声についても従来の電話帯域でなく広帯域の高音質な音声が求められるようになってきており，16 kHz～22 kHz サンプリングのシステムなども出てきている．TV 会議，電話会議では拡声通話が使われ，前述の音響エコーキャンセルは必須である．TV 会議を実施する会議室の広さも従来に比べて広くなってきており，出席者とマイクの間の距離も大きくなってきている．

こうした状況で集音機能を高めるためにマイクを複数搭載するシステムも出てきている．マイクを複数搭載することにより，指向性を制御したり音源方向を同定することが可能となり，例えば出席者の居る方向のゲインを大きくし，雑音源が存在する方向のゲインを低くするといったことができる．ただし，マイクが複数になると各マイクごとのエコーキャンセルも必要となり，所要処理量はかなり増加する．マイク 32 ch，スピーカ 12 ch といった構成の電話会議システムが数十万円といった価格で市販されるようになってきている．この価格の大部分がマイク，スピーカの通話モジュール部分であり，多チャネル音響エコーキャンセルなどは高性能の DSP により処理されるのが一般的である．

マイクアレイ技術に関してはまだ開発途上の技術も多く，消費電力の問題なども特に大きくないので専用 LSI 化するよりは高性能の汎用 DSP にて実装されるのが一般的である．一方で，多チャネルマイク入力信号のデータを DSP などに取り込むために，多チャネル並列入力信号データを直列信号化する回路が重要となってきており，こうした部分については専用 LSI 化あるいは FPGA などにより構成される場合が多い．

(4) シリコンマイク

マイクアレイ技術は携帯電話などにも搭載される傾向にある．携帯電話の場合にはマイク数は 2 ch から多くても 4 ch 程度である．小型組込み機器にマイクを実装する場合，実装面積の観点もさることながら従来の ECM（Electlet Condensor Microphone）ではアナログ信号線を引き回すためにノイズを拾いやすいという問題があった．数年前からディジタルマイクあるいはシリコンマイクといって MEMS（Micro Electro Mechanical System）技術を利用し，元々はメカニカルな機構であったマイクのダイアフラムをシリコン基板上に作成するようなマイクが登場してきている．シリコンマイクの場合，同一パッケージ上に A/D 変換器を搭載することも可能となり，上記アナログ信号線に基づくノイズの問題が回避できるだけでなく，センサ部のサイズは 1 mm 角を切っており，ECM に比べてかなり小さく，携帯電話でのマイク実装やマイクアレイにおけるマイク実装など実装面積や実装厚さでクリティカルになる応用場面で有利になっている．

3-3-4 省電力化

音声処理に関しての省電力について，特に処理アルゴリズムの観点からの省電力ということでは，音声符号化で使われている VAD（Voice Activity Detection）が代表例としてあげられる．VAD は音声パワーや音声の基本周波数などの情報に基づいて，音声が発話されているかどうかを検出する処理である．一般に会話において，実際には音声がなされていない部分も大分ある．発話のない無音部分については符号化処理を省略し，背景雑音相当データを送信することにより省電力化を行う．この考え方は音声認識，マイクアレイ処理ほか，入力音声

信号に対して何らかの処理を施す音声処理技術では同様に扱うことができ，無音と判定される部分に対しては音声処理を施さないことにより省電力化を図る．

音声処理については周波数分析をはじめとして，積和演算が多用されるが，ハードウェアの観点から見ると，こうしたケースでは汎用 CPU に比べて DSP の方が省電力となっている．また，更に省電力化を図るには，基本的には低電圧での低いクロック周波数での動作が必要となる．特に超小型の携帯型音楽プレーヤなどでは，低電圧，低クロック周波数で動作させるために専用 LSI を構成する例が見られる．

■10 群 － 5 編 － 3 章

3-4 グラフィックス

（執筆者：森　健一）［2009 年 4 月 受領］

3-4-1 グラフィックス概要

グラフィック（Graphic）とは図や絵を意味する．これをコンピュータで処理するのが CG（Computer Graphics）である．特に，奥行きのある 3 次元空間を表現したものを，3 次元コンピュータグラフィックス，または 3D（3 Dimension）CG と呼ぶ．コンピュータが処理できるデータの状態で 3 次元空間の情報の保持と計算処理が行われ，人間が見ることができるように描画（Rendering）処理され，ディスプレイや紙に表示，印刷される．そのための専用の LSI を GPU（Graphics Processing Unit）と呼ぶ．

(1) グラフィックスデータの持ち方

CG で原子に相当する基本要素は，点（Point），線（Line），面（Surface）である．極論すれば点だけですべてを表現でき，そのような手法も存在するが，データ扱いと処理の利便性から，線と面が活用される．線の代表は直線，特に二つの点（両端）で定義される線分（Line Segment）である．曲線は数式とパラメータでデータ化され，描画時に点や線分に分解されて描画される．

面の代表は平面，特に三つの点（頂点）で定義される三角形（Triangle）である．三角形を含めた多角形をポリゴン（Polygon）と呼び，処理の基本単位に用いられる．三角形は，独立したものよりも，隣接して頂点を共有して連なることで効率的に多角形を表す Triangle Strip が多く用いられる．曲面は数式とパラメータでデータ化され，描画時に三角形や点に分解されて描画される．

このように，頂点とそれをつなぐ直線や曲線で構成されているデータをベクタ型と呼んでいる．

(2) 表示バッファ

アナログテレビ放送のように，送信側からラスタで送られてきた映像を，そのままの順序でラスタ方式にて表示するだけであれば，小さなバッファがあれば動作できる．しかし，CG では各データを描画する順序は画面のラスタ順ではなく，ほとんどランダムな位置と大きさのポリゴンを描画していく．CG をラスタ表示するためのハードウェアは，メモリや処理速度が小さい時代のスプライト技術やラインバッファ技術などを経て，現在はフレームバッファ（Frame Buffer）方式が主流となっている．

フレームバッファは，メモリ上に表示 1 画面分の画素（ピクセル，Pixel）データを 2 次元格子状に並べたバッファを構成するものである．通常は，描画中の不完全な画面を見せないために，描画用と表示用とフレームバッファをもち，描画後にそれを切り替えるダブルバッファ（Double Buffer）方式が用いられる．

(3) ラスタライズ（Rasterize）

2 点で定義される線分などのベクタデータの描画は，ベクタスキャンのディスプレイや XY

プロッタであれば，その線分を直接描画するだけでよい．しかし，現在主流のフレームバッファにベクタ型の直線やポリゴンを書き込むには，それらが 2 次元格子状のフレームバッファのどの画素を占めるかを計算する，ラスタライズ（Rasterize）と呼ばれる処理が行われる．

(4) API（Application Programming Interface）

グラフィックスを含め，アプリケーションプログラムが各種機能を使うための，関数ライブラリ群を API と呼ぶ．ハードウェアを使うための API も，実現できる機能の定義から API を設計し，個々のハードウェア固有の操作を API の内部で実装することで，ハードウェアを抽象化できる．これにより，異なるハードウェアでも同じアプリケーションプログラムを動作させることができ，ソフトウェア資産の活用が可能となるので，大変重要である．

3-4-2 2D グラフィックス

2 次元グラフィックス（2 Dimension Graphics）は，文字やイラストなど奥行き方向の情報をもたない 2 次元の描画処理である．

(1) データの形式

(a) ビットマップ（Bitmap）

2 次元の格子状に並べた正方形の点で画を表す方式．複数の大きさや解像度に対応するには，それぞれの大きさのデータをもつか，構成する画素データすべての拡大縮小が必要になる．

(b) ベクタ（Vector）

線分と曲線で画を表す方式．描画時に頂点位置を拡大縮小計算してからラスタライズするので,一つのデータセットさえ定義しておけば任意の解像度に対応できるという特徴をもち，よってスケーラブル（Scalable）と呼ばれる．

(2) フォント（Font）

漢字を含めて文字を表示するためのデータがフォントである．各サイズごとのデータをもつビットマップフォント（Bitmap Font）形式と，スケーラブルなフォントであるアウトラインフォント（Outline Font）が用いられている．オープンソースのフォント描画ライブラリとして FreeType などがある．アウトラインフォントの描画処理では，同じサイズ・同じ傾きの同じ文字が何度も現れるので，ラスタライズ後のデータをキャッシングすることで処理量を減らせる．

(3) 2 次元 CG のデータ形式と API

データ形式と API について，代表的なものをそれぞれ一つ紹介する

(a) SVG（Scalable Vector Graphics）

2D グラフィックスとグラフィカルアプリケーションを記述する言語のひとつで，W3C 勧告となっている．XML で記述されるベクタグラフィック言語と，画像フォーマットが定義されている．ビットマップデータとベクタデータを扱え，レイヤ機能でそれらを重ね合わせた表示を定義できる．文字はベクタデータのアウトラインフォントで扱われる．近年は多くの

ブラウザが描画対応してきている．

(b) OpenVG

ローレベルなハードウェアレベルでの 2D 描画機能を定義する API で，標準化団体クロノスグループ（The Khronos Group）が管理している．オープンソースによる実装として OpenVG がある．

(4) ハードウェア化

現在の PC 分野では，2D グラフィックス処理は CPU や GPU で処理され，専用ハードウェアをもつことはない．家電などの組込み用途では，CPU の性能がそれほど高くないので，システム LSI に内蔵された専用のハードウェアブロックや GPU が 2 次元処理を実施するケースが多い．

3-4-3 3D グラフィックス

3 次元コンピュータグラフィックス（3 Dimension Computer Graphics：3DCG）のハードウェア化の目的としては，以前は CG 映像制作向けのアクセラレータという分野もあったが，現在はリアルタイム CG 処理のための専用ハードウェアという位置づけが主である．

(1) リアルタイム 3D グラフィックス概要

リアルタイム 3 次元グラフィックスは，ユーザの入力を反映して更新される 3 次元空間データを，1/30 秒間隔や 1/60 秒間隔で 2 次元の画面に描画し続け，スムーズな動画生成する技術である．映画などのように前もって計算された CG 映像を再生するのではなく，上記の短い時間で映像を生成している点が大きく異なる．

ゲームやユーザインタフェースで，ユーザの入力（キーボード，マウス，コントローラなど）に対応して，視点の位置や物体の位置といった 3 次元空間の構成データが変更される．変更された 3 次元データを表示するには，それをユーザの視点から見た映像を 2 次元の画面に透視投影して描画処理する．

表示できる画像の画質は時代とともにワイヤフレーム，フラットシェーディング，スムーズシェーディング，テクスチャマッピング，複雑なテクスチャマッピング，プログラマブルシェーダへと進歩し，一秒間に描画できるラインやポリゴン（Polygon，多角形）の数と画素（ピクセル，Pixel）数も増加してきた．

(2) 必要とされる演算量とメモリバンド幅

MPEG-2 などの映像ストリームの表示であれば，ストリームのビットレートと解像度などから，必要な処理能力とメモリバンド幅を決めることができる．これに対し，3 次元 CG では必要な性能に上限というものがない．より多くの演算能力とメモリバンド幅があれば，より複雑な映像を描画することが可能になる．このため，半導体技術（LSI 技術）の進歩を活用し，常により高い性能のグラフィックスシステムが生まれ続けてきている．

(3) 大規模なシステムからコモディティへ

リアルタイム 3D CG システムは，1963 年の Sketchpad を始祖として，1970 年代からフラ

イトシミュレータなどの大型のシステムで実用化されてきた．1980 年代には，LSI 技術の進歩を応用し，3 次元形状に対する対話的（インタラクティブ）な操作が求められる機械系や建設系の CAD のために，WS（ワークステーション）分野で特に高い 3 次元 CG 能力をもった GWS（グラフィックスワークステーション）が生まれた．GWS により，デザイナが対話的操作でモデリングすることが可能となり，映画など CG 映像を作成するためにも用いられ，CG 作成のための専用ソフトウェアも開発された．

ゲームセンタのアーケードゲーム機も，1990 年代はじめにレースゲームなどから，ポリゴンを用いた本格的な 3 次元 CG システムが用いられ，すぐに家庭用ゲーム機も 3 次元 CG 化してきた．家庭用ゲーム機は出荷台数が多いので量産効果が得られることから最先端の半導体技術が投入された．平行して，PC（パーソナルコンピュータ）の CPU 性能が向上し，1990 年代半ばに 3 次元処理できる PC 用の GPU が現れるに従い，フライトシミュレータをはじめとした 3 次元ゲームも PC 上で動作するようになり，先の GWS 的な用途も PC で可能になってきた．家庭用ゲーム機の 3 次元 CG 技術は，パチンコ・パチスロといったアミューズメント機器にも応用され，携帯電話の 3 次元 CG 機能と性能も進んできている．

(4) レンダリングパイプラインとハードウェア化

現在のところリアルタイム 3D CG のレンダリング手法はポリゴン系が主体である．特に，処理の最小単位である多量の三角形を高速に処理するハードウェアとなっている．ポリゴン系の CG の処理の流れはレンダリングパイプラインと呼ばれ，以下の処理ステージをもつ．

(a) 頂点処理部（Vertex Processing）

直線やポリゴンの頂点単位の処理を行う．視点から見た位置への座標変換や，頂点単位の光源処理を行うため，T&L（Transform and Lighting）やジオメトリ処理部（Geometry Processing）とも呼ばれる．

(b) ラスタライズ部

先に述べた，頂点データによる直線やポリゴンが，フレームバッファ上のどの画素を占めるか計算する処理部．ハードウェア演算回路として実装されることが多い．

(c) 画素処理部（Pixel Processing）

画素単位の処理を行う．フレームバッファ上の画素（Pixel）に対し，そこに至るまでの処理単位を Fragment と呼ぶ場合もある．画素単位に，画像を貼り付けるテクスチャマッピング処理，画素の色を決定する処理，そして隠面処理や半透明処理を行って，フレームバッファの画素情報を更新する．

(d) 表示部

フレームバッファのデータをビデオインタフェースから出力する．

(5) アーキテクチャの進化 －固定パイプラインからシェーダへ

GWS やパソコンで実現されてきたハードウェア構成の変化の概要は下記である．

(a) 専用 LSI 化の流れ

GWS 時代には，ジオメトリ処理専用の LSI と，レンダリング専用の LSI が用いられていた．パソコン向けの初期の GPU には，2 次元のビデオカード技術の延長から画素処理部の LSI 化が行われ，浮動小数点処理を含む頂点処理は CPU が実施した．

LSI 技術の微細化の進歩により搭載できる回路規模が大きくなり，1999 年頃から，パソコン向け GPU に頂点処理部も組み込まれるようになった．LSI 化された頂点処理部も画素処理部も，当初は API などで規定された固定処理だけをユーザに公開していた．頂点処理部は DSP 的な部分をもつ構成だったが，そのプログラムはファームウェアとして非公開であった．ユーザは API で定義された各種パラメータによって動作モードを変えることができ，年々そのパラメータ指定のバリエーションが増えていった．

(b) プログラマブルシェーダの登場

2001 年に，パラメータ指定だけでなく，処理内容や処理手順をユーザがプログラマブルに与えられるように進化した，プログラマブルシェーダ（Programmable Shader）をもつ LSI が登場した．頂点処理部をシェーダ化したのが Vertex Shader で，画素処理部をシェーダ化したのが Pixel Shader または Fragment Shader と呼ばれる．当初の LSI で可能な回路規模では画像処理部のプログラムに大きな制約が残っていたが，さらなる LSI の微細化により可能な処理の汎用性が進んできた．近年の GPU は，シェーダ構成のみが主体となり，過去の固定パイプライン処理もシェーダのプログラムとして実装されるようになってきている．

(c) 頂点処理と画素処理の統合

シェーダ登場以前のハードウェアは，頂点処理は浮動小数点計算で，画素処理は整数演算であり，処理内容も大きく異なっていたが，画素処理での浮動小数点利用や汎用性が進につれ，両者を統合する方向になってきている．モデルと実装の二つの方向から統合が行われている．

- Unified Shader Model：ユーザから見えるシェーディング言語と計算モデルを統合したものである．ハードウェア実装として頂点処理部と画素処理部が異なるアーキテクチャということもあり得る．
- Unified Shading Architecture：頂点処理と画素処理が同じハードウェアブロックで実行されるアーキテクチャである．上位と逆に，ユーザから見えるシェーディング言語が異なることもあり得るが，今後はこのハードウェア上で(a)のモデルが実現されていく．

(6) CG 技法のリアルタイム処理

下記のような CG 特有の技法をリアルタイムに実現するため，専用のハードウェア回路，またはソフトウェアとの組合せで実現されている．

(a) 陰面除去

遠くの物体は手前の物体の陰になって見えないことがある．このことを計算するのが陰面除去である．フレームバッファのほかに，画素ごとの遠近情報を格納するデプスバッファ（Depth Buffer または Z Buffer）をもち，画素単位に一番手前のデータだけが残るように処理される．

(b) テクスチャマッピング（Texture Mapping）

看板をモデル化するとき，そこに描かれている絵を詳細にポリゴンでモデル化するより，絵をそのまま貼り付けた方が簡単である．このように，ポリゴンの表面に画像を貼り付けるのがテクスチャマッピングであり，そのための回路を GPU はもつ．林や茂みなどの大量の樹木も，枝や葉を一つずつモデル化するのではなく，1 本の木を 1 枚のテクスチャ画像としてモデル化されることが多い．リアルタイム CG の歴史の中では，(遠くが小さく見えるという)

透視変換の中で，正しく描画するためには，画素単位での割り算が必要であり，十分な演算能力を GPU がもてるまでは実現できなかった．

(c) 反射，屈折

正確な反射や屈折の計算は，ポリゴン型描画ではできず，擬似的な手法が開発されてきたが，近年のシェーダの進歩で，より正確なモデルの実装が可能になってきた．

(d) 陰（Shade）と影（Shadow）

どちらもリアリティ向上に必要なものであるが，特に影は物体の位置関係を認識するヒントとしての役割が大きい．しかし，各物体で光源の反対側が暗くなる「陰」は，その物体と光源の関係だけで計算されるのに対して，「影」は他の物体に影が落ちるのであり，個々の物体独立ではなく，複数の物体が関係するので，その計算は複雑になる．影の計算には，擬似的で簡便な方法から，より高度な方法まで様々なアルゴリズムがあり，近年の CPU では，影の計算のための特殊機能の実装と処理の高速化，シェーダプログラムの開発が進んでいる．

(e) アンチエリアシング

ポリゴンの辺やラインを斜めに描画する際に現れるギザギザであるジャギーを軽減する処理．各ピクセルの色を決定する際に，細かいサブピクセル単位でポリゴンの境界部分が入るかどうか判定し，その重みを用いた色計算によりジャギーを弱められる．

(7) 最近の GPU の例

2009 年はじめ時点のハイエンド GPU LSI の例を記載する．近年，GPU の高集積化が続いており，その回路規模は CPU を凌駕している．更にハイエンドのボード構成として，これら GPU を 2 個乗せたものも製品化されている．

- **NVIDIA 社 GeForce GTX 285**：トランジスタ数は約 14 億個．55 nm プロセスにおいて，シリコンの面積が約 470 mm^2 という巨大な LSI で，最大消費電力も 183 W である．スカラ型で，約 1.5 GHz で動作するシェーダプロセッサ 240 基をもち，演算性能は約 1 Tflops である．
- **AMD 社 Radeon HD 4870**：トランジスタ数 9.56 億個で，55 nm プロセスを用いて，シリコンの面積 282 mm^2 とのこと．TDP（Thermal Design Power）で定義される消費電力は 160 W．VLIW 型で，750 MHz で動作するシェーダプロセッサ 800 基をもち，演算性能は 1 Tflops を超える．

上記二つの GPU は，シェーダプロセッサの個数が大きく違うが，これはどのような処理単位や回路単位で一つのプロセッサを構成するかのアーキテクチャの違いによる．

・並列計算機による GPU 機能

Intel 社が Larrabee（ララビー）アーキテクチャを発表している．これは，汎用の並列計算機であり，また GPU 機能も実現するものである．従来の GPU と異なり，CPU として実績のある x86 命令セットの派生型を用いたシェーダコアを多数搭載した構成となっている．GPU がシェーダの汎用性向上で CPU 化しているのに対し，CPU の並列化を進めた結果 GPU としても使えるものを目指すという見方もできる．従来のポリゴン系 CG よりも，レイトレーシングなどのグローバルイルミネーション処理を目指しているともいわれている．

(8) メモリ

リアルタイム 3 次元 CG では，高速で大容量のメモリが必要とされる．そのため，通常の

メインメモリである DDR 系をベースにしつつ，よりバンド幅を大きく取れる仕組みを加えた高速な GDDR 系メモリが用いられる．HD 解像度の CG のためには，最低 2 枚のフレームバッファに，デプスバッファ，そして高解像度でもリアリティを出すための高品位なテクスチャ画像を多数もつためのメモリが必要となる．

(9) リアルタイム CG 用 API とシェーディング言語

CG 技術が複雑になり，また次々と新しい機能を追加した新しい製品がでる時代には，それらをプログラマが使いこなすためのライブラリ API が重要度を増している．特にハードウェアの抽象化が重要であり，一度作成したアプリケーションやライブラリが，新しく出たハードウェアでも動くことが求められている．これと同時に，新しいハードウェア機能に対応した拡張も求められ続けている．従来との互換維持と新しい技術の導入をどう進めるかがポイントとなる．また，グラフィックスパイプラインのシェーダ処理をユーザが記述するための，シェーディング言語も開発されている．シェーディング言語は，ピクサー社の RenderMan の Shading Language を手本にスタートしてきた．どれも高級言語の枠組みとして C 言語をベースに CG に適した工夫がなされている．

OpenGL と DirectX Graphics の二つが広く使われている CG 用 API である．

(a) OpenGL

OpenGL は SGI 社の IrisGL ライブラリをベースに規格化されたグラフィックス API である．現在は，標準化団体クロノスグループ（The Khronos Group）が管理している．OpenGL は複数の OS 上で実装されており，CAD からゲームまで，広い範囲で用いられている．OpenGL 1.x では，グラフィックスパイプラインが定義され，OpenGL 2.0 で本格的にシェーディング言語主体となった．OpenGL のシェーディング言語は，GLSL（OpenGL Shading Language）と呼ばれている．OpenGL 3.0 では，今までのオプショナルな拡張機能が標準化される一方，現在の GPU 性能の向上に適していない初期時代の API を，将来の削除に向けて非推奨としたことに今後の方針が見える．

フリーウェアの OpenGL API 準拠の実装には，Mesa3D ライブラリがある．下位レイヤのソフトウェアライブラリとして，Gallium3D の開発が進んでいる．組込み用途向けのサブセット規格として OpenGL ES（OpenGL for Embedded Systems）が提案され，実装されている．OpenGL ES 2.0 では OpenGL 2.0 と同様に，シェーディング言語に対応している．

(b) DirectX Graphics

DirectX はマイクロソフト社が仕様を策定する Windows OS 用の API である．ゲームをターゲットとして開発され，現在は各種メディア処理に広く使われている．DirectX 全体としては，音声・ネットワーク・入力デバイスの処理など幅広いコンポーネントから構成されている．その中に 3 次元グラフィックス API である Direct3D を主体とした DirectX Graphics がある．DirectX Graphics のシェーディング言語は，低レベルなプログラマブルシェーダと，高級言語化した HLSL（High Level Shading Language）がある．

(c) その他

nVIDIA 社が提供するシェーディング言語の Cg（C for Graphics）は，同社の GPU であれば，OpenGL でも DirectX でも使えるものである．

(10) 上位 CG ライブラリ

前記 CG ライブラリより上位のレイヤに，シーングラフライブラリ（Scene Graph Library）やゲームエンジン（Game Engine）と呼ばれるものがある．これらは，3 次元モデルを意味のあるオブジェクト単位で管理することを基本に，影（Shadow）の管理と生成，アニメーションの管理などを提供する．リアルタイム CG で用いられ CG 技術が多彩になるにつれ，先の CG 用 API でゼロから実装することは工数がかかりすぎるため，こちらのライブラリを用いたアプリケーション開発が重視されている．例として，ある PC 用ゲームのために開発されたゲームエンジンが外販され，ほかの PC 用ゲームがそのエンジンを用いて開発されることも増えている．これらのライブラリには，商用のものやフリーウェアがいくつかある．

(1) OpenSG

(2) OpenSceneGraph

3-4-4 GPGPU (General Purpose Computing on GPU)

GPU のもつ莫大な演算能力を CG 以外にも活用しようというのが GPGPU である．

(1) 演算要素

GPU のシェーダやテクスチャユニットがもつ演算器は，4 次元ベクトルの浮動小数点の積和演算器を主体としており，これを汎用演算に用いることができる．なお，3 次元 CG でもちられる行列演算は，RGB の 3 原色や 3 次元の回転計算まで可能な 3×3 の行列ではなく，透過度のアルファ値を含めた RGBA の 4 要素，及び，平行移動と透視変換計算まで含めた 4×4 の行列演算を基本としているので,4 並列の浮動小数点演算を得意とするハードウェアが多い．

(2) GPGPU の API と言語

シェーディング言語は CG 処理をターゲットとしているため，汎用の数値演算には向かない面もある．そこで，GPU のハードウェアの特性を反映しつつ，より汎用に近い言語が開発されている．

(a) CUDA (Compute Unified Device Architecture)

NVIDIA 社の提唱する GPGPU 用言語. 同社からの GPU 向け物理演算ライブラリ PhysX は，GPU に移植された際に CUDA で書き直されている．

(b) OpenCL (Open Computing Language)

C 言語をベースとした，並列コンピューティングのためのフレームワークで，GPU だけでなく，CPU や DSP でも動作する構想である．アップル社が提案し，現在は標準化団体クロノスグループ（The Khronos Group）が仕様策定を管理している．

(c) その他

AMD 社も ATI Stream という，CUDA と同じ方向性の言語を提供している．マイクロソフト社は，DirectX に，Direxst3D に統合する形の，DirectX Compute Shader を追加しようとしている．

(3) 用 途

GPGPU の具体的な用途は多く考えられているが，実用が進んでいるものは下記がある．

全般に，並列処理に向いているうえで，処理量が多く，現在の CPU でも処理時間がかかるアプリケーションがターゲットであり，GPU により低コストで性能を得ることを狙っている．

(a) 数値演算

浮動小数点の加算積算性能を活かして，従来はスーパーコンピュータで行っていたような大規模な行列演算やフーリエ変換などを演算要素に，様々な用途のシミュレーション計算に用いられる．

(b) 映像のエンコード処理

処理量が多く CPU でも時間のかかる HD サイズの MPEG-4 AVC（H.264）のエンコードも，ハイエンドの GPU であれば高速に処理できるようになっている．

(c) 画像処理フィルタ

画像に対して鮮鋭化などの多彩なフィルタ処理を行うプログラムとアプリケーションが開発されている．画像フィルタは並列化しやすいものが多く，GPU の並列な演算器を活用することができる．高度なフィルタは複雑なプログラムとなるが，汎用処理が十分に進化した GPU のシェーダで実行できるようになってきた．

(d) ゲームの高度な支援

物体の衝突計算や流体計算などの物理シミュレーションの計算や，ゲーム中の人間などの行動をシミュレートする人工知能の計算に用いられる．

3-4-5 高度な 3D グラフィックスやその派生

従来は CPU 上のプログラムで計算されていた高度なアルゴリズムも，シェーダによるプログラマビリティを獲得した現在の GPU で処理できるようになってきている．

(a) Global Illumination

3 次元空間中の物体の相互反射など，もっと複雑に光の通り方を計算する手法で，Ray Tracing，Radiosity，Photon Mapping などのアルゴリズムが開発されている．ポリゴン系のグラフィックスパイプラインでは，ポリゴン同士の相互関係がないので，ストリーム的に処理を流し込むが，Global Illumination では相互関係の計算が行われるので，3 次元空間のデータベースを何度もアクセスすることとなる．

(b) Non Photo Realistic

リアルさの追求ではない描画のスタイルを狙うものである．筆で描いた絵画調の画像を生成する手法や，トゥーンレンダリングという 3 次元の物体モデルデータからアニメのような 2 次元化した映像を描画する手法などがある．

(c) Image Based Rendering

多数の実写の映像をより高度に利用する手法である．

(d) Volume Rendering

bitmap を 3 次元化したような 3 次元の格子状のデータを扱い，描画する手法である．医療用の画像処理などで用いられる．

3-4-6 今後の方向性

今後，リアルタイム 3 次元コンピュータグラフィックスや GPU がどうなっていくかを考えてみる．まず，リアルタイム 3 次元 CG の用途が依然としてゲーム中心でゲームの映像の高品位化だけが進歩しており，CG の恩恵を受けるユーザ層を広げられないでいる．ゲームの CG が進歩すれば他の用途に広がるという議論は行われてきたが，今までは大きな成果は出ていなかった．ユーザインタフェースへの応用も，幾度か試みられ，そのようなフレームワークが提案されてきたが，まだ普及に至っていない．しかし，GPU が CPU のチップセットに内蔵されるなどのコモディティ化が進み，すべてのパソコンが高性能な CG 処理ができつつある現在，温故知新的な試みが大きな変革をもたらす可能性は十分にあると考えられる．

■参考文献

1) Real-Time Rendering http://www.realtimerendering.com/
2) Khronos Group http://www.khronos.org/
3) OpenGL http://www.opengl.org/
   http://www.khronos.org/opengl/
4) OpenGL ES http://www.khronos.org/opengles/
5) OpenCL http://www.khronos.org/opencl/
6) Mesa3D http://www.mesa3d.org/
7) SVG http://www.w3.org/Graphics/SVG/
8) Wikipedia （GPU）http://ja.wikipedia.org/wiki/Graphics_Processing_Unit

■10 群 － 5 編 － 3 章

3-5 通　信

3-5-1 誤り訂正符号

（執筆者：宮本直人）［2009 年 8 月 受領］

ディジタル衛星通信，移動通信などの通信システムにおいては，情報が正確に伝送されることが極めて重要である．しかしながら，実際には通信路で雑音や減衰などにより信号に誤りが発生する．誤り訂正は通信システムにおいてディジタル信号の伝送の誤りを軽減し，信頼性を高めるための技術である．誤り訂正をするため，送信側では信号を符号化して送信する．一方，受信側ではこれを受信して復号しなくてはならない．

(1)　ビタビ復号器

ビタビ復号器は畳み込み符号で符号化された信号をビタビアルゴリズム[1)]に基づいて復号する LSI である．本 LSI は，受信した信号からハミング距離ないしユークリッド 2 乗距離を計算するブランチメトリック計算回路，1 レベル前の各ステートの生き残りパスの尤度を記憶するパスメトリックメモリ，生き残りパスの尤度を計算する ACS（Add-Compare-Select）回路，生き残りパスを蓄積し復号結果を出力する回路から構成される．ビタビ復号では，生き残りパスを確定する方法としてトレースバック方式とレジスタエクスチェンジ方式がある．後者は短い拘束長で，高スループット低レイテンシが要求される用途に向いている．

(a) トレースバック方式（図 3･39）

生き残りパスの決定には，生き残りパスメモリとトレースバック回路が用いられる．生き残りパスメモリは，全レベル・全ステートでの ACS 回路の選択結果を記憶する．生き残りパスメモリを元に，トレースバック回路は最終レベルから初期レベルへと順に生き残りパスを確定していく．したがって，トレースバック回路には LIFO（Last In First Out）が必要となる．

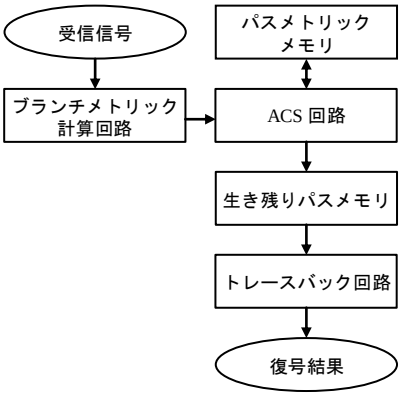


図 3・39　トレースバック方式

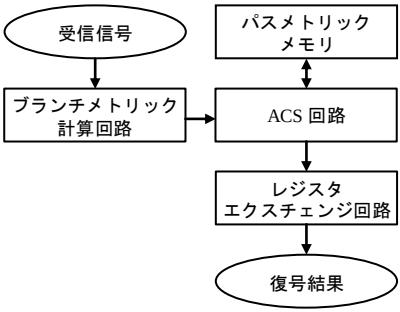


図 3・40　レジスタエクスチェンジ方式

(b) レジスタエクスチェンジ方式（図 3･40）

レジスタエクスチェンジ方式では，パスメモリを使用せず，代わりに各ステートに生き残りパスの履歴を保持するレジスタを割り当てる．レベル k におけるレジスタの値は，生き残りパスのレベル $k-1$ におけるステートのレジスタの値を継承する．レジスタエクスチェンジ回路は，継承したレジスタの値とレベル k におけるステート固有のビットとを合わせて生き残りパスの履歴を生成し，レジスタの値を上書きする．最終レベルにおける生き残りパスの履歴そのものが生き残りパスとして確定される．したがって，トレースバックは必要ない．

(2) ターボ符号

ターボ符号は 1993 年に開発されたシャノン限界に近い高性能な誤り訂正符号である [2), 3)]．符号化には 2 個の RSC（Recursive Systematic Convolutional）符号化器とインターリーバが用いられる（**図 3･41**）．ターボ符号化器の入力信号 r_0，及び 2 個の RSC 符号化器の出力信号 r_1，r_2 が送信される．なお，RSC 符号化器の出力をパンクチャリングすることで高符号化率の符号を得ることもできる．

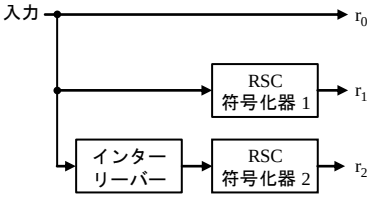


図 3・41　ターボ符号化器

ターボ復号器は 2 個の要素復号器，2 個のインターリーバ，2 個のデインターリーバと硬判定出力回路から構成される（**図 3･42**）．要素復号器 1 では受信信号 r_0，r_1 とデインターリーブされた外部値 Λ_{e2} を用いて外部値 Λ_{e1} を計算する．外部値とは受信信号の対数尤度比（事後値）の一部であり，受信ビットが 1 である可能性が高い場合は正となり，0 である可能性が高い場合は負となる．要素復号器 2 ではインターリーブされた Λ_{e1}，インターリーブされた r_0 と r_2 を用いて外部値 Λ_{e2} と事後値 Λ を計算する．軟出力値である Λ は硬判定回路により最終的な復号ビットに変換される．このようにインターリーバ，デインターリーバを介して外部値のフィードバックを繰り返すことでビットエラー率特性を向上することができる．

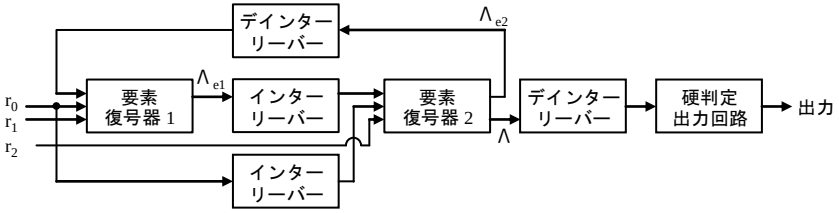


図 3・42　ターボ復号器

要素復号器の復号アルゴリズムは, MAP（Maximum A Posteriori）[4], Log-MAP, Max-Log-MAP, SOVA（Soft-Output Viterbi Algorithm）がある [5]．それぞれの計算量を**表 3･4** に示す．k は RSC 符号の入力数，ν は RSC 符号のメモリ数である．演算の種類は大別して加算，乗算，最大値計算（MAX），ルックアップ（LU），指数計算（EXP）がある．MAP が最も計算量が多く，Log-MAP，Max-Log-MAP，SOVA の順に計算量が少ない．一方，ビットエラー率特性では MAP と Log-MAP が最も良く，Max-Log-MAP，SOVA の順に悪化する．

表 3･4　要素復号器の計算量

	MAP	Log-MAP	Max-Log-MAP	SOVA
加算	$2\times 2^{k+\nu}+6$	$6\times 2^{k+\nu}+6$	$4\times 2^{k+\nu}+8$	$2\times 2^{k+\nu}+9$
乗算	$5\times 2^{k+\nu}+8$	$2^{k+\nu}$	$2\times 2^{k+\nu}$	$2^{k+\nu}$
MAX		$4\times 2^{\nu}-2$	$4\times 2^{\nu}-2$	$2\times 2^{\nu}-1$
LU		$4\times 2^{\nu}-2$		
EXP	$2\times 2^{k+\nu}$			

(a) Sliding Window

ターボ復号器の構成では SRAM などのメモリが LSI の面積の大部分を占める．符号長 N が大きい場合，この問題は特に深刻となる．ターボ復号アルゴリズムでは前方確率（α）と後方確率（β）を漸化式に基づいて計算する．対数尤度比の計算には α と β の両方が必要となるため，β を先に求めた後，α と対数尤度比が計算される（**図 3･43**）．したがって，ターボ復号器には β 保存用にサイズ N のメモリが必要となる．

β 保存用メモリのサイズを縮小するために Sliding Window 方式 [6] が提案されている．この方式では符号長 N におけるすべての β を保存するのではなく，学習期間 L だけ保存する．一般的に L は高速長 $K = \nu + 1$ の 5～6 倍であり，N よりも圧倒的に小さいため，大幅な β メモリのサイズ縮小が可能である．Sliding Window 方式の計算フローとこの方式を用いた場合のターボ復号器のアーキテクチャを**図 3･44** と**図 3･45** に示す．β トレーニング，β 計算，α 計算を並列演算することで，復号スループットが約 2 倍に増加する．

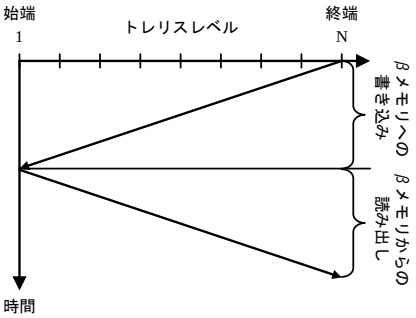


図 3･43　ターボ復号フロー

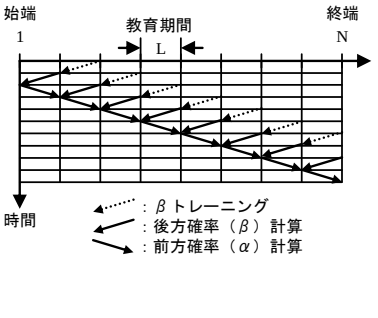


図 3･44　Sliding Window ターボ復号フロー

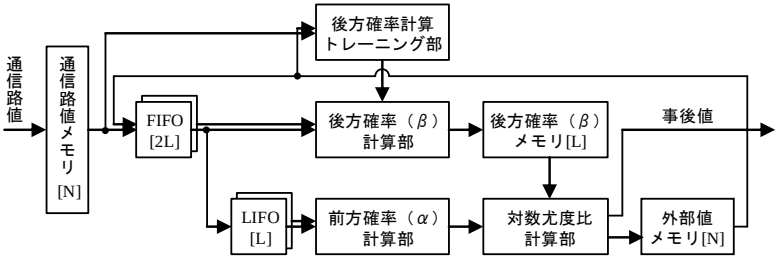


図 3・45　Sliding Window ターボ復号器のアーキテクチャ

(b) Block-Interleaved Pipelining

更に高スループットなターボ復号器を実現するために提案された方式が Block-Interleaved Pipelining（BIP）である[7]．BIP では復号トレリスを 1～$N/2$（ブロック 1）と $2/N+1$～N（ブロック 2）に分割し，ブロック 1 とブロック 2 を交互にインターリーブすることで，漸化式でありながらパイプライン処理を可能とする．ブロック 2 の α 計算の前にαトレーニングが必要となる（**図 3･46**）．BIP により復号スループットが更に約 2 倍増加する．

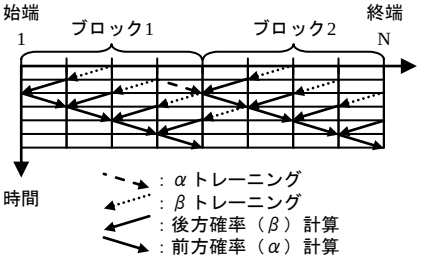


図 3・46　BIP ターボ復号フロー

ブロックインターリーブした確率計算器とその入出力回路を**図 3･47** に示す．ブロック 1 とブロック 2 から交互にインターリーブされたデータが確率計算器に入力される．確率計算器の出力は 2 度レジスタ（D）で遅延された後，漸化式の一部として再度確率計算器に戻る．ここで，1 個のレジスタをリタイミングにより確率計算器の中間に配置することで，動作周波数 $f'=2\times f$（$f=1/\tau$）の 2 段パイプライン演算器が得られる（**図 3･48**）．

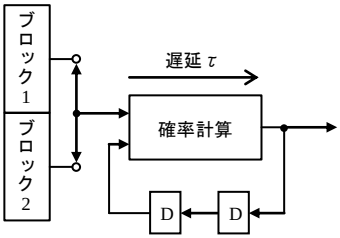


図 3・47　ブロックインターリーブした確率計算器

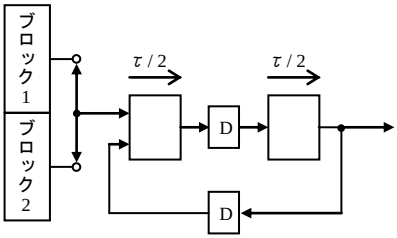


図 3・48　レジスタをリタイミングした確率計算器

BIP のブロック分割数を増やすことにより，更に深いパイプライン化が考えられる．しかしながら，それはαトレーニング計算回数の増大を招いてしまう．富士通研究所の雁部らは，前回のターボイタレーションで得られた後方確率を次回のターボイタレーションの初期値に利用することで，ビットエラー率特性を劣化させることなくトレーニングを省略可能な Improved Sliding Window 方式を提唱した[8]．この方式をαトレーニングに応用することで，深いパイプライン構造をもつ超高スループットの BIP が実現可能となる．

■参考文献

1) Andrew J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” IEEE Trans. Information Theory, vol.IT-13, no.2, pp.260-269, 1967.
2) Claude Berrou, Alain Glavieux and Punya Thitimajshima: Claude Berrou: “Near Shannon Limit error-correcting coding and decoding: Turbo-codes,” Proc. IEEE Int. Conf. Comm., pp.1064-1071, 1993.
3) Claude Berrou and Alain Glavieux, “Near Optimum Error Correcting Coding and Decoding: Turbo-Codes,” IEEE Trans. Communications, vol.44, no.10, pp.1261-1271, 1996.
4) L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, “Optimal Decoding of Linear Codes for minimizing symbol error rate,” IEEE Trans. Information Theory, vol.IT-20, no.2, pp.284-287, 1974.
5) Branka Vucetic and Jinhong Yuan, “Turbo Codes: Principle and Applications,” Kluwer Academic Publishers
6) S. Benedetto, D. Divsalar, G. Montorsi and F. Pollarab, “Soft-Output Decoding Algorithms in Iterative Decoding of Turbo Codes,” The Telecommunications and Data Acquisition Progress Report 42-124, NASA Jet Propulsion Laboratory, pp.63-87, 1996.
7) Seok-Jun Lee, Naresh R. Shanbhag and Andrew C. Singer, “A 285-MHz Pipelined MAP Decoder in 0.18-um CMOS,” IEEE J. Solid-State Circuits, vol.40, no.8, pp.1718-1725, 2005.
8) Hiroshi Gambe, Yoshinori Tanaka, Kazuhisa Ohbuchi, Teruo Ishihara and Jifeng Li, “An Improved Sliding Window Algorithm for Max-Log-MAP Turbo Decoder and Its Programmable LSI Implementation,” IEICE Trans. Electron., vol.E88-C, no.3, pp.403-412, 2005.

3-5-2 MIMO システムとその LSI 化設計

（執筆者：尾知　博）[2009 年 12 月 受領]

本節では，MIMO（Multi Input Multi Output）ワイヤレス通信方式について，その各種デコードアルゴリズムと LSI 化設計について説明する．MIMO システムを大別すると以下の 2 種類がある．

A. 時空間符号化 STC-MIMO

直交する符号系列よりなる時空間符号（Space Time Coding：STC）を用いて，送受信ダイバシチゲインを得る方法である[1]．この方法は，言い換えれば MIMO を用いて SISO システムより通信距離を拡大する方法である．伝送レートは高速化できない．

B. 空間分割多重 SDM-MIMO

空間分割多重（Space Division Multiplexing：SDM）MIMO システムを用いると，通信路容量の増大化すなわち伝送レートの高速化が周波数帯域を広げることなく可能である．SDM-MIMO は，更に送信側でアンテナビームフォーミングを利用する方式と，受信側で多重化された符号をデコードする干渉キャンセラを利用する方式の二つに区分される．いずれも一定の周波数帯域内でマルチストリーム伝送を行うことで，伝送レートを上げる方式となっている．

無線 LAN の規格 IEEE 802.11n や WiMAX で知られる IEEE 802.16e 無線 MAN の国際標準規格，及び LTE などの最近のセルラーシステムでは，この SDM-MIMO アーキテクチャが積

極的に採用され，100 Mbps 以上もの高速伝送レートが実用化され，将来的には 1 Gbps を超える超大容量を目指している．STC-MIMO もこれらの規格でオプションとして採用され，実用化されている．

実装面で考えると STC-MIMO は，送信側で直交符号列に時空間符号化し，受信側では最大比合成（MRC）を行うだけなので，LSI 化設計は比較的簡単である．そこで，本節ではより LSI 設計が困難な SDM-MIMO システムの復号方式（干渉キャンセラ：Interference Canceller または MIMO デコーダ：MIMO Decoder と呼ぶ）について解説する．

まず，3-5-1 節で MIMO システムのベースバンドチャネルモデルについて述べ，3-5-2 項でチャネルの特異値分解を用いた送信ビームフォーミングによる SDM-MIMO 方式について説明する．送信ビームフォーミングを行うためには，送信側にチャネル行列をフィードバックする必要があり，スループットの低下を招くことや複雑な通信プロトコルとなる問題がある．そこで，3-5-3 節では受信側だけで復号する MIMO デコーダに関して，まず線形復号である ZF/MMSE アルゴリズムの説明とそれらの復号性能及び LSI 化設計の例を示す．3-5-4 節では，復号性能を追求した最尤推定 MLD アルゴリズムを用いた MIMO デコーダの各種アルゴリズムと LSI 化設計の例を幾つか紹介する．

(1) MIMO チャネルモデルと MIMO システム

(a) MIMO ベースバンドチャネルモデル

図 3･49(a) に送信アンテナ M 本，受信アンテナ N 本の MIMO 通信システムのベースバンドモデルを示す．図(b) はチャネルモデルを表す．

図 3･49 の MIMO 通信システムのモデル化において，本節では以下の仮定を考える．

仮定 1　送信アンテナ数：M 本

時刻 n における送信符号を次式で定義しておく．

$$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_M]^T \tag{1}$$

ここで，$i = 1, 2, \cdots, M$ は，QAM などにマッピングされた狭帯域信号とする．すなわち，パスバンドでは，シングルキャリア変調を対象とする．また今後，時刻 n は明示しない．

仮定 2　フラットフェージング（メモリレス）チャネル，($N \times M$)

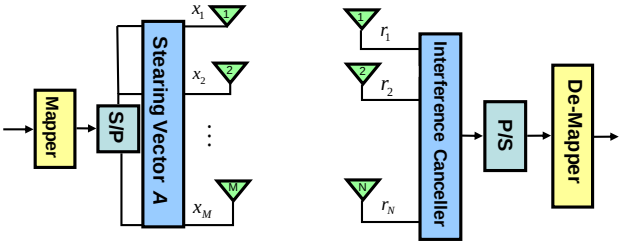
$$H = [h_{nm}] = \begin{bmatrix} h_{1,1} & h_{2,1} & \cdots & h_{M,1} \\ h_{1,2} & h_{2,2} & \cdots & h_{M,2} \\ \vdots & \vdots & \ddots & \vdots \\ h_{1,N} & h_{2,N} & \cdots & h_{M,N} \end{bmatrix} \tag{2}$$

ただし，h_{nm} は i.i.d.チャネルとする．

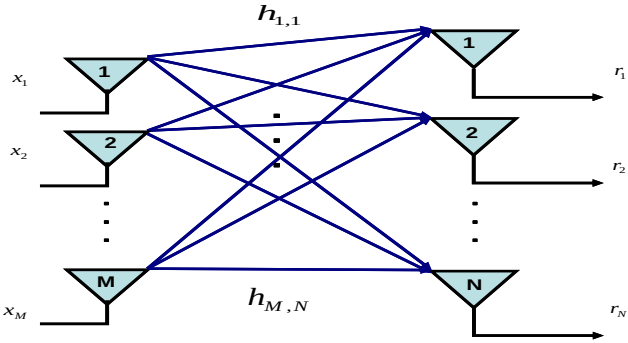
仮定 3　受信アンテナ数：N 本

時刻 n における受信信号を以下で定義する．

$$\mathbf{r} = [r_1 \quad r_2 \quad \cdots \quad r_N]^T \tag{3}$$



（a）送受信モデル



（b）チャネルモデル

図 3・49　MIMO 通信システム

以上より，MIMO 通信システムのベースバンドモデルは次式で表現できる．

$$\mathbf{r} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{4}$$

ただし，**n** は雑音ベクトル（雑音電力σ^2の白色ガウス雑音）とする．

ここで MIMO システムの特徴は，仮定 1 より各送信アンテナから別々の符号 x_i が M 個同時刻 n かつ同一周波数で送信することである．このような送信方法を階層符号化（Layered Space Time Coding：LST）と呼ぶ．階層符号化の利点は，SISO システムに比べて伝送レートを最大で M 倍に高速化できる点である（$N \geqq M$ 時）．一方，STC で得られるダイバシチゲインは，LST では基本的に得られない．

図 3･49 において Mapper は，QPSK や QAM などのシンボルマッピングを行う．マッピングされた符号は，SP 変換により並列化され，送信信号 **x** としてそれぞれのアンテナから送信される．送信ビームフォーミングを行う場合は，SP 変換後にステアリングベクトルとして，ある特定の $M \times M$ 正方行列 $\boldsymbol{A}$ を乗算する．すなわち SP 変換されたシンボルに対し行列 $\boldsymbol{A}$ により，ある符号化を行っていると考えてもよい．このように行列 $\boldsymbol{A}$ で符号化を行う場合を空間分割多重方式（SDM）と呼び，階層符号化 LST の特殊な場合である．ステアリングベクトル $\boldsymbol{A}$ や送信ビームフォーミング SDM については文献 2) などを参照されたい．送信ビームフォーミングを行わない場合は，階層時空間符号化 LST に相当し，$M \times M$ 単位行列 **I** をかけておく．なお，本節では慣例として LST も SDM と今後呼ぶことにする．

送信側である時刻 n に送信された $\boldsymbol{x}$ は MIMO チャネルで干渉し，受信信号 $\boldsymbol{y}$ となる．受信側では，$\boldsymbol{y}$ から送信信号 $\boldsymbol{x}$ を復号するために，干渉キャンセルを行う．干渉キャンセラすなわち MIMO デコーダについては以下で詳しく述べる．

なお，MIMO チャネルの統計的モデルは，文献 3) が詳しい．また，Matlab で記述された MIMO チャネルや，次に述べる各種 MIMO デコーダ，及び MIMO 無線 LAN システムのシミュレーションプログラムも必要に応じて利用されたい [4].

(2) 線形 MIMO デコーダ

階層符号化 MIMO システムの性能は, MIMO デコーダすなわち干渉キャンセラの復号性能に依存する．また，実装上最も演算量・ハードウェア量が多くなる部分が干渉キャンセラである．本節では，まず線形復号方式の Zero Forcing（ZF）法，Minimum Mean Squared（MMSE）法について述べ，次にこれらの線形復号法を利用して SN 比の良好な信号順に順序付けして復号する V-BLAST アルゴリズムを示す.

(a) Zero Forcing（ZF）法

$\mathbf{I}$ を $N\times N$ の単位行列として

$$\mathbf{WH}=\mathbf{I} \tag{5}$$

となる $M\times N$ 重み行列 $\mathbf{W}$ を式(4)の両辺に掛けることで干渉の抑制・除去を行い，受信信号 $\mathbf{x}$ を検出することを考える．すなわち,

$$\begin{aligned}\mathbf{y}&=\mathbf{Wr}\\&=\mathbf{W}\left(\mathbf{Hx}+\mathbf{n}\right)\\&=\mathbf{x}+\mathbf{Wn}\end{aligned} \tag{6}$$

チャネル行列 H は一般に M 行 N 列の非正方行列である．したがって，重み行列 $\mathbf{W}$ としてチャネル行列 $\mathbf{H}$ の Moore-Penrose 擬似逆行列 $\boldsymbol{W}_{ZF}$ を用いる．$N \geqq M$ の場合，$\boldsymbol{W}_{ZF}$ は次式で与えられる（脚注 *[1] 参照）．ここで，$\mathbf{H}^{\mathbf{H}}$ は $\mathbf{H}$ のエルミート転置行列である．

$$\mathbf{W}_{\mathbf{ZF}}=\left(\mathbf{H}^{\mathbf{H}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathbf{H}} \tag{7}$$

このように, 重み行列 $\mathbf{W}$ を受信信号にかけることにより, 干渉の抑制･除去が可能となり，受信信号 $\mathbf{x}$ を復号することができる．このような意味から，重み行列 $\mathbf{W}$ を干渉キャンセラ（Interference Canceller）あるいは MIMO デコーダ（MIMO Decoder）と呼んでいる. 式(7)の $\boldsymbol{W}_{ZF}$

---

*[1]**擬似逆行列**　式(7)の証明

[証明]

ノイズレス環境と仮定すると，式(6)より

$$\mathbf{H}^{\mathbf{H}}\mathbf{r}=\mathbf{H}^{\mathbf{H}}\mathbf{Hx}$$

上式においては正方行列（M×M）となり，正則である場合は逆行列が存在する．よって

$$\mathbf{x}=\left(\mathbf{H}^{\mathbf{H}}\mathbf{H}\right)^{-1}\mathbf{H}^{H}\mathbf{r}$$

したがって，式(6)の重み行列の定義より次式を得る.

$$\mathbf{W}_{\mathbf{ZF}}=\left(\mathbf{H}^{\mathbf{H}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathbf{H}}$$ （Q.E.D.）

は，Zero Forcing（ZF）法と呼ばれる干渉キャンセラである．ZF 法では，式(6)のように行列積 **WH** の対角要素以外をゼロに強制するので，Zero Forcing という名前が付けられている．ZF 型 MIMO デコーダの問題点としては，式(6)に示すように復号時にノイズ強調が生じる欠点がある．

(b) Minimum Mean Squared（MMSE）法

MMSE 法は，送信信号 **x** とデコード信号 **y** の自乗平均誤差の期待値を最小にする干渉キャンセル方法である．ある受信アンテナ $i(i \le N)$ における送信信号 x_i と復号信号 y_i の平均自乗誤差ε は，E を期待値とすると次式となる．

$$\varepsilon = E\left|x_i(n) - y_i(n)\right|^2,\ i = 1,2,...,M \tag{8}$$

MMSE 法における評価関数εを最小にすることは，式(8) を見てわかるように，復号信号と干渉成分を含む雑音との比（Signal to Noise Interference Ratio：SINR）を最大にしていると考えることができる．一方，ZF 型は単純に SNR を最大にしていることとなる．

このε を最小にする重み行列 **W** は，$N \geqq M$ の場合に次式となる．導出は，文献 5) などを参照されたい．

$$\mathbf{W}_{MMSE} = \left(\mathbf{H^H H} + \sigma^2 \mathbf{I_M}\right)^{-1} \mathbf{H^H} \tag{9}$$

ただし，σ^2 は AWGN 付加雑音信号の分散である．

ここで，$\underline{\mathbf{r}} = [\mathbf{r} \quad \mathbf{0}], \underline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sigma^2 \mathbf{I_M} \end{bmatrix}$とすると，式(6)，式(9) より次式が成り立つ．

$$\mathbf{y} = \left(\underline{\mathbf{H}}^{\mathbf{H}} \underline{\mathbf{H}}\right)^{-1} \underline{\mathbf{H}}^{\mathbf{H}} \underline{\mathbf{r}} \tag{10}$$

上式の$\left(\underline{\mathbf{H}}^{\mathbf{H}} \underline{\mathbf{H}}\right)^{-1} \underline{\mathbf{H}}^{\mathbf{H}}$ の項は$\underline{\mathbf{H}}$ の Moore-Penrose 擬似逆行列であり，したがって MMSE 法は ZF 法と同様な式で復号していることがわかる．相違点は，雑音信号を考慮した重み行列 W かどうかという点である．式(9) の雑音分散σ^2 は，信号が送信されていない時間で測定するか，あるいは既知の信号（例えば，無線 LAN ではプリアンブル）が送出されている時間に，受信信号電力から既知信号電力を引いて求める．

ところで，雑音分散σ^2 の推定が精度よく行われたとすると，MMSE 法の方が ZF 法より BER 特性が改善され性能がよい．理由は，MMSE 法は式(8) の平均自乗誤差を最小すなわち SINR を最大にする規範であり，一方 ZF 法は単に復号信号の SNR を最大にするため，雑音強調が生じる場合があるからである．性能比較は本節 (4) 項で示す．

(c) ストラッセン算法を用いた MMSE デコーダ

次に LSI 化に適したストラッセン算法を用いた MMSE デコーダについて述べる [6]．

・アルゴリズム

ストラッセン算法は，複素行列の乗算や逆行列計算における乗算数を低減する算法である．4×4 行列Ω の逆行列の場合は，まずΩを 2×2 のブロック行列 A～D に分割する．

$$\Omega = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \tag{11}$$

ここで，ストラッセン算法を用いると，逆行列は以下のように求められる．

$$\Omega^{-1} = \begin{pmatrix} F & -A^{-1}BE^{-1} \\ -E^{-1}CA^{-1} & E^{-1} \end{pmatrix} = \begin{pmatrix} A' & B' \\ C' & D' \end{pmatrix} \tag{12}$$

ただし，$E = D - CA^{-1}B$, $F = A^{-1} + A^{-1}BE^{-1}CA^{-1}$

ここで，行列Ωはエルミート行列であるので，$\boldsymbol{C} = \boldsymbol{B}^{H}$, $\boldsymbol{B}' = \boldsymbol{C}'^{H}$が成立する．この性質より，$\boldsymbol{A}^{-1}\boldsymbol{B} = (\boldsymbol{C}\boldsymbol{A}^{-1})^{H}$ となり，$\boldsymbol{A}^{-1}\boldsymbol{B}$ を計算すれば $\boldsymbol{C}\boldsymbol{A}^{-1}$ の計算が不要となる．同様に，$-A^{-1}BE^{-1}$を用いて $\boldsymbol{F}$ の計算に必要な$A^{-1} + A^{-1}BE^{-1}CA$を求めることができる．以上のように重複する計算は一度しか行わず，計算値を流用することで全体の演算量を低減している．

以上の算法に基づく逆行列計算のフルパイプライン回路を**図 3･50** に示す．左から 2×2 行列ブロック $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ を入力し，Ω^{-1} の計算に必要な値が式(12)に基づいて順に計算されているのが理解できる．

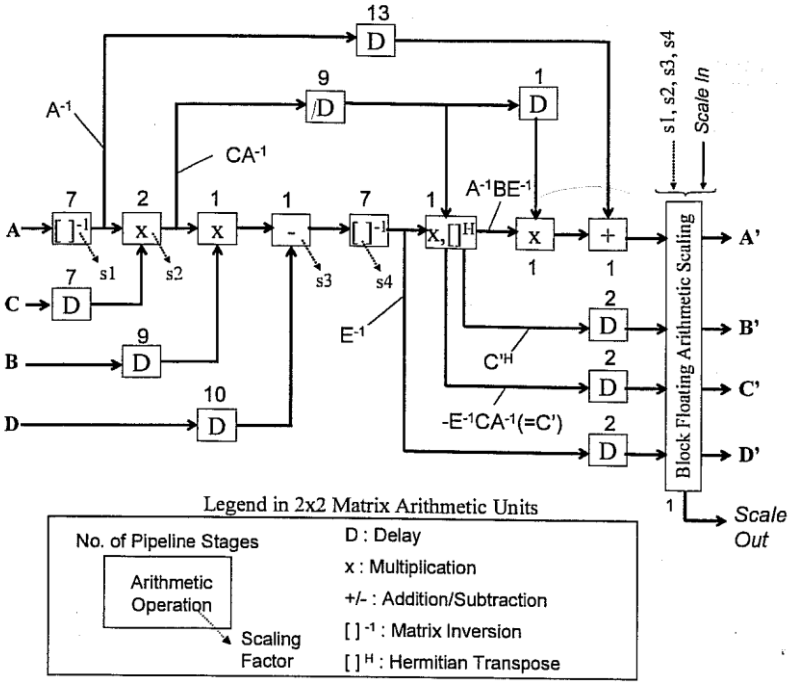


図 3･50　ストラッセン型 MMSE MIMO デコーダ [6]

(d) 順序付け干渉キャンセル法（V-BLAST）

次に，Vertical-Bell Laboratory Space-Time Coding（V-BLAST）と呼ばれる，SN 比順に信号の順序付けを行う干渉キャンセル法について紹介する [9]．ここでは，ZF 法を用いて説明する

が，MMSE 法でも同様にデコードできる．

式(6) より，雑音成分の期待値は次式で表される．

$$N_{ZF} = E\left\{\left(\mathbf{W}_{ZF}\right)_i \mathbf{n}\right\}\left\{\left(\mathbf{W}_{ZF}\right)_i \mathbf{n}\right\}^* = \sigma^2 \left\|\left(\mathbf{W}_{ZF}\right)_i\right\|^2 \tag{13}$$

ここで，E は期待値，$(\mathbf{W}_{ZF})_i$ は行列 $\mathbf{W}_{ZF}$ の i 行を意味している．

よって，

$$k_i = \arg \min_{i=1,2\cdots M} \left(\left\| \left(\mathbf{W}_{ZF}\right)_i \right\|^2 \right) \tag{14}$$

となる番目の受信アンテナを選択すれば SN 比が最大となり復号精度が高まる．

今，最適な復号手順が $k_1, k_2, \cdots, k_M$ であるとすると，復号過程は次のようになる．

(1) 重みベクトル $\mathbf{w}_{k_i}$ を用いて，式(13)の第 2 式より k_i 番目 y_{k_i} の信号を求める．

(2) 硬判定を行い，k_i 番目の信号 $\hat{x}_{k_i}$ を推定する．

(3) $\hat{x}_{k_i} = x_{k_i}$ として，受信ベクトル $\mathbf{r}_i$ から推定済みの信号を除去する．

このように(1)～(3)の手順を $k_1, k_2, \cdots, k_M$ の順に行い，受信ベクトル $\mathbf{r}_i$ を修正しながら送信信号を推定する．

次に，復号手順 $k_1, k_2, \cdots, k_M$ の選択方法について説明する．ZF 法を用いた場合，k_i 番目の重みベクトルは次式を満たす．

$$\mathbf{w}_{k_i} [\mathbf{H}]_{k_j} = \begin{cases} 0 & (j \neq i) \\ 1 & (j = i) \end{cases} \tag{15}$$

ここで，$[\ ]_{k_j}$ は行列の k_i 番目の列ベクトルを表す．また，$\mathbf{w}_{k_i}$ はそれに対応する信号がキャンセル（除去）される前の $\mathbf{r}_i$ によってスパンされる部分空間であり．式(15) を満たす唯一のベクトルが $H^{+}_{\bar{k}_{i-1}}$ の k_i 番目の行である．ここで，$\mathbf{H}_{\bar{k}}$ は $\mathbf{H}$ の k 列要素をゼロにした行列である．

以上をアルゴリズムとしてまとめると，

V－BLAST Algorithm:
init:
$i = 1;$
ZF criterion:
MMSE criterion:
$\mathbf{R}_i = \mathbf{R};$
recursion:
$k_i = \arg \min_{j \notin k_1, k_2, \cdots, k_{i-1}} \left| |(\mathbf{W})_j|^2 \right|;$
$y_{k_i} = \mathbf{W}_{k_i} \mathbf{R}_i;$
$\hat{x}_{k_i} = Q(y_{k_i});$
$\mathbf{R}_{i+1} = \mathbf{R}_i - [\mathbf{H}]_{k_i} \hat{x}_{k_i};$
$\mathbf{W}_{i+1} = (\mathbf{H})^{+}_{k_i};$
$i = i + 1;$

(e) 線形 MIMO デコーダの LSI 設計例

ストラッセン算法を用いた 4×4 MMSE デコーダと比較対象の設計例を**表 3･5** に示す.

表 3･5　線形 MIMO デコーダの LSI 化設計例（文献 6) より一部抜粋）

参考文献	(7)	(8)	(6)	(10)
アルゴリズム	ZF	MMSE	ストラッセン型 MMSE	MMSE V-BLAST
伝送速度〔bps〕			2.6 G	212 M
ASIC プロセス・ゲート数（面積）	90 nm 43 k	0.25μm 89 k	90 nm 1.86M	0.18μm （1 mm^2）
レイテンシ〔ns〕	180	600	187.5	－
K サブキャリア当たりのレイテンシ〔ns〕*	180×K	600×K	187.5	－

*は MIMO-OFDM の場合である.

(3)　最尤推定 MIMO デコーダ

最尤推定（Maximum Likelihood Decoder : MLD）法は，MIMO における最適な信号復号方法である.

MLD 法は，式(4)において，受信信号 $\boldsymbol{R}$ と受信レプリカ信号 $\boldsymbol{HX}_j$ との二乗ユークリッド距離を最小とする送信レプリカ信号系列 $\hat{X}$ を選択することで行い，次式で表せられる.

$$\hat{X} = \arg \min_{X_j \in X_1, X_2, \cdots X_{M^{*2^{K*M}}}} \|R - HX_j\|^2 \tag{16}$$

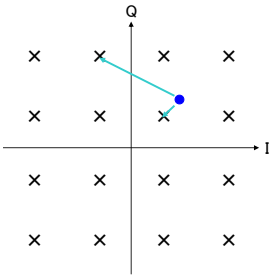


図 3･51　MLD の概念

例えば**図 3･51** の 16 QAM において，受信信号 $\boldsymbol{R}$ の最適レプリカ信号は左下のシンボル点となる．MLD は，MIMO における最適な復号法であるが，すべての送信レプリカ信号系列候補との二乗ユークリッド距離を計算しなくてはならず, 例えば, 送信アンテナ数 4, 64 QAM 変調の場合，2^{24} 個もの送信レプリカ信号系列候補が存在し，膨大な演算量となりハードウェア化は困難である．そのため，QRM-MLD，Sphere Decoding，K-best などの低演算量の手法が提案研究されており，順に以下で説明する.

(a) QR 分解法

QR 法では，チャネル行列 **H** の QR 分解を行い，以下のようにする．

$$\mathbf{H}=\mathbf{QR} \tag{17}$$

ここで，**Q** はユニタリー行列，**R** は上三角行列であり，式(17)の左から $\mathbf{Q}^H$ をかけると式(18)となる．

$$\mathbf{y}=\mathbf{Q}^H\mathbf{r}=\mathbf{Q}^H\mathbf{QRx}+\mathbf{Q}^H\mathbf{n}=\mathbf{Rx}+\mathbf{n}' \tag{18}$$

よって，QR 分解では **R** の最下位行より送信ベクトル **x** の最後の要素 x_M をまず求め，ついで x_M を前方代入して残りの $\boldsymbol{x}$ の要素を順に推定していく．

4×4 MIMO システムを例にとり，式(18)を要素別に書き下すと，

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}=\begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{14} \\ 0 & R_{22} & R_{23} & R_{24} \\ 0 & 0 & R_{33} & R_{34} \\ 0 & 0 & 0 & R_{44} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{19}$$

となり，最初に x_4 の候補を以下の最尤推定により求める．64 QAM の場合は，64 回のユークリッド距離の計算と比較を行う必要がある．

$$\arg\min\left\|y_4-R_{44}\tilde{\mathbf{x}}\right\|^2 \tag{20}$$

続いて，x_4 の候補 64 個を用いて以下の最尤推定により x_3 を求める．この場合，64^2 回の比較演算が必要となる．

$$\arg\min\left\|y_3-R_{34}\tilde{\mathbf{x}}-R_{33}\tilde{\mathbf{x}}\right\|^2 \tag{21}$$

同様な手順で，x_2, x_1 も求める．x_1 に至っては，64^4 回もの距離計算と比較演算が必要となり，このままではほとんどチップ化は困難な演算量であることがわかる．

以下では，このユークリッド距離の比較演算を少なくしてチップ化を目指したデコードアルゴリズムを幾つか設計例と共に紹介する．

(b) Sphere Decoder

説明を簡単にするため送信アンテナ数 M = 3, BPSK 変調の場合について説明する．ここで，シンボル候補を S_1, S_2 とする．

(i) アルゴリズム

最適な MLD では，送信アンテナそれぞれで異なるシンボルが送信されるので，2^3 = 8 通りの送信シンボル候補の組合せがある．受信側では，受信信号とシンボル候補とのユークリッド距離（以下，メトリック）の計算が必要であるが，Sphere Decoder では木探索法を用い，探索を打ち切るメトリックの値（SC : Sphere Constraint）を更新しながら探索することで，ユークリッド距離の計算回数を削減している．

図 3･52 はそのアルゴリズムを示している．各ノードは送信シンボル候補を，各パスに割り当てられた数字は，次のノードへのメトリックを表している．このアルゴリズムは，各ステージごとに最小のパスを選びながら葉ノードまで探索する（深さ優先探索）．図 3･52 では $s_{1,2,1}$ へたどり着いており,この葉ノードまでのパスの積算メトリック 8 を保存し，これを SC とする．

以降は，探索が葉ノードに到達するか，またはその時点のノードまでの積算メトリックが SC を超えた場合に，未探索ノードのうち現在のノード直近の上位ステージのノードに戻り，同様に深さ優先探索を行う．葉ノードまで探索し，それまでの積算メトリックが SC 未満の場合，SC をその値に更新する．これを探索するノードがなくなるまで繰り返す．**図 3･52** の例では，$s_{2,1,1}$, $s_{2,1,2}$, $s_{2,2,1}$, $s_{2,2,2}$ へのメトリック計算が省略され，演算量が削減されている．

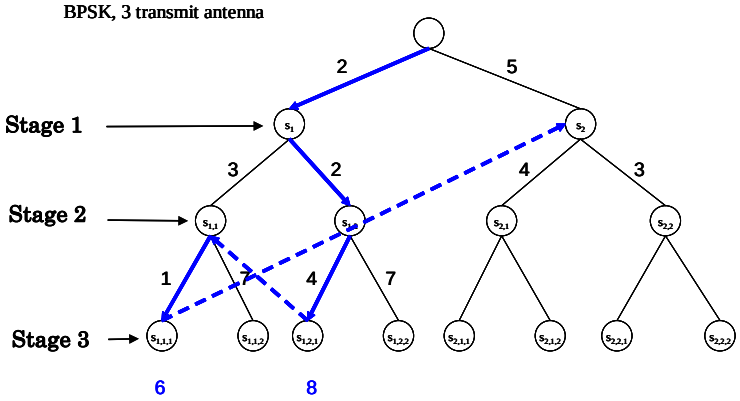


図 3・52　Sphere Decoding アルゴリズム

(ii) アーキテクチャ

次に，Sphere Decoder のチップ化の例について紹介する [10]．**図 3･53** はそのアーキテクチャであり，各ブロックを①～④に分けて説明する．

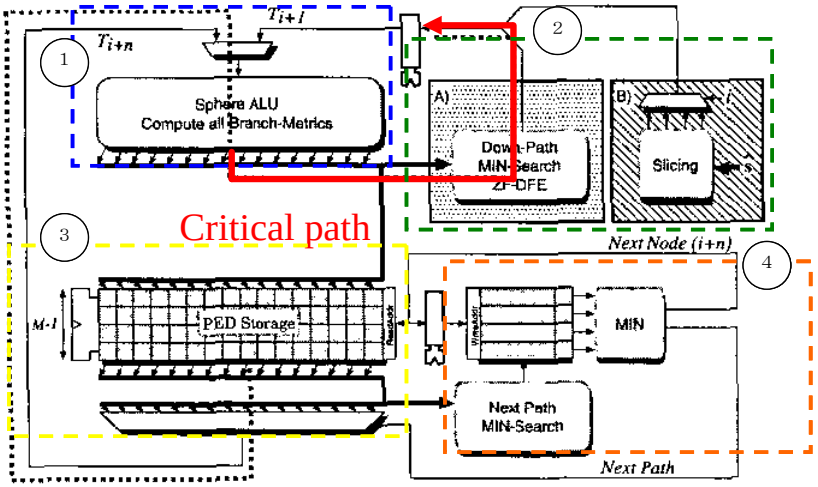


図 3・53　Sphere Decoding アーキテクチャ

① このブロックでは,②または③からフィードバックされたノードから一つ下の子ノードへのすべてのパスのメトリックを計算し，現在のノードまでの積算メトリックと足し合わせる．そして，子ノードのインデックスと積算メトリックの組を②，③へ渡す．

② B) のブロックは，このアーキテクチャ特有のもので，一般のスフィアデコーダが行わない処理をする部分なので省略する．ここでは A) のブロックについて述べる．このブロックでは，①から受け取ったデータから，最小のメトリックをもつ一つの子ノードを選択して①へ戻す．現在のノードの最小メトリックが SC（Sphere Constraint：積算メトリックがこの値を超えるとそれ以下のノードの探索を打ち切る）を超えている場合はこの処理を行わない．また，現在のノードが葉ノードであり，かつ，積算メトリックが SC より小さい場合，このパスを記憶しておき，その積算メトリックを新しい SC の値とする．

③ このブロックでは現在のノードに至る経路上のノードに対する全子ノードの積算メトリックを PED Storage に保存する．葉ノードへのパスはここには保存しない．例として，図 3･52 において青の線で書いた経路で探索した場合，その時点で PED Storage に保存されているメトリックは図 3･52 中の数字を書いたパスになる．数字はそのパスのメトリックを表している．

④ ここでは，①の探索が終わった後 PED Storage 中の最もステージ数の大きいノードのうち SC 以下で, 最小のメトリックをもつノードを取り出し①へ戻す. 取り出したノードは PED Storage から消去する．例として図 3･52 で葉ノード $s_{1,2,1}$ のまで探索した場合，PED Storage から取り出すのは一つ上のステージの $s_{1,1}$ である. PED Storage が空になると探索終了となる．

このアーキテクチャでは，②から①へのフィードバックを必要とすることから，この部分の処理のパイプライン化ができないため, 図 3･53 の赤で示すパスがクリティカルパスとなる．このため，本アルゴリズムでは，現在のデバイス技術では，数百 Mbps を超える高スループットを得ることは難しい．

(c) K-best

(i) アルゴリズム

K-best では，式(10)の QR 分解後において，各ステージの探索ノードを K 個に制限することでユークリッド距離計算回数を削減している．K-best では Sphere Decoder と同様に，木探索を行うが幅優先の探索である．

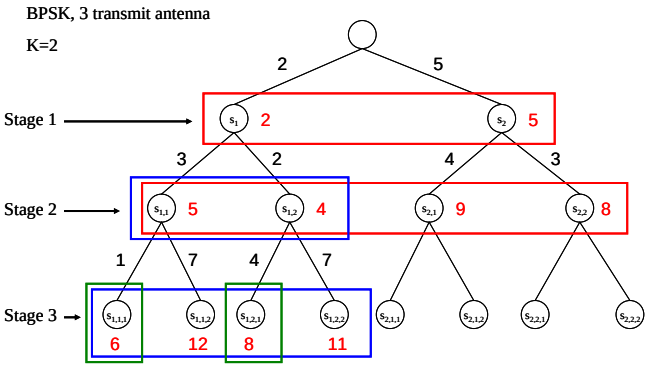


図 3･54　K-best アルゴリズム

図 3･54 はそのアルゴリズムを示している．各ステージではメトリック計算済みのパスのうち，積算メトリック最小の K 個を選択し，それらについてのみ，次ステージのノードへのメトリックを全候補計算する．図 3･54 では各ステージで探索ノードを 2 個ずつに絞りながら探索を行っている．

K-best アルゴリズムにおいても，探索ノード数 K が大きいと演算料が低減できない．そこで，各ステージごとに探索ノード数を変える Modified K-best アルゴリズムも開発されている．

(ii) **アーキテクチャ**

図 3･55 は K-best の 1 ステージ分を計算するアーキテクチャを表している．この Computation Unit を直列にステージ数分つなぐことでパイプライン動作させる．この図 3･52 のアーキテクチャを説明する．前のステージから受け取った K 個の候補ノードに対し，MCU（Metric Computation Unit）では 1 ステップで一つのノードのすべての子ノードへのメトリックを計算し，積算メトリックを KBU（K-best Unit）の PEDs に保存する．それと同時に PEDs の要素のうちメトリックが最小の K 個が常にバッファ L_K に入っているようにする．MCU，KBU の処理を K ステップ行うと KBU のバッファ L_K には前のステージから受け取った K 個の親ノードに対する全子ノードへのメトリック中最小の K 個が入るので，これを次のステージの Computation Unit に渡す．最終ステージの出力が求めるパスとなる．

このアーキテクチャでは一つの Computation Unit で K ステップかかり，これがステージ数分つながってパイプライン動作するので，レイテンシは K ステップである．

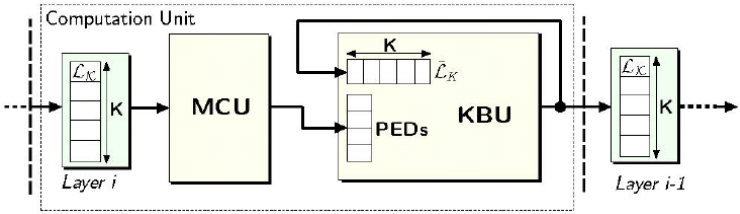


図 3・55　K-best **アーキテクチャ**

(iii) LSI **化設計**

MLD を用いた MIMO デコーダの LSI 化設計の例を**表 3･6** に示す．

表 3・6　MLD-MIMO **デコーダの** LSI **化設計例**

Algorithm	Sphere Decoder [11]	K-best [13]
Number of antenna Modulation	4×4, 16 QAM	4×4, 16 QAM
Throughput〔Mbps〕	80	424
Latency〔cycle〕	11.2	5
Process Technology	0.25μm	0.25μm
Number of gates	117 k	68 k
Area〔mm^2〕	3.55	－

(4)　各種干渉キャンセラの性能比較

ここでは各種 MIMO デコーダの BER 特性の性能比較を示す．

(a) 線形デコーダの BER 特性

まず**表 3･7** に示す伝搬路環境で，OFDM-SDM における各種干渉キャンセラの性能比較を行う．

まず，BER 特性の比較を行う．表 3･7 にシミュレーションの諸条件を示す．なお，畳み込み符号化や Viterbi 誤り訂正は行っていない．

表 3・7　シミュレーション条件

Data modulation	QPSK
Number of sub-carriers	512
Number of data-carriers	480
Gurard Interval length	64
Channel estimation	Ideal
Channel model	Quasi-static multipath fading(18-path)
Number of iteration	10000

図 3･56 より ZF 法より MMSE 法の特性がよいことがわかる．また，BLAST アルゴリズムの方が順序付けの分，ダイバシチゲインが得られている．更に，MLD が最もダイバシチゲインが得られていることがわかる．この場合の MLD のアルゴリズムは，原理的な総当り法であり硬判定で実施した．

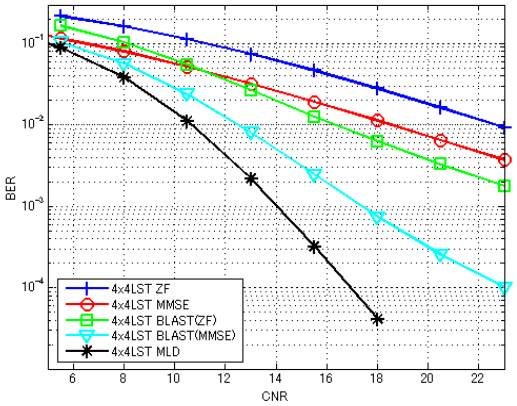


図 3・56　4×4 MIMO 各種干渉キャンセラの BER 特性

(b) MLD デコーダの BER 特性

次に，**図 3･57** に各種 MLD アルゴリズムの比較を示す．ここでは，参考のために MLD 法の中で比較的演算量が少ない ASSESS [14] や Viterbi アルゴリズムについても特性を示してい

る．このように K-best アルゴリズムに比べて，その他の MLD アルゴリズムは若干 BER が劣化しているが，ZF 法よりはるかに性能は良好である．

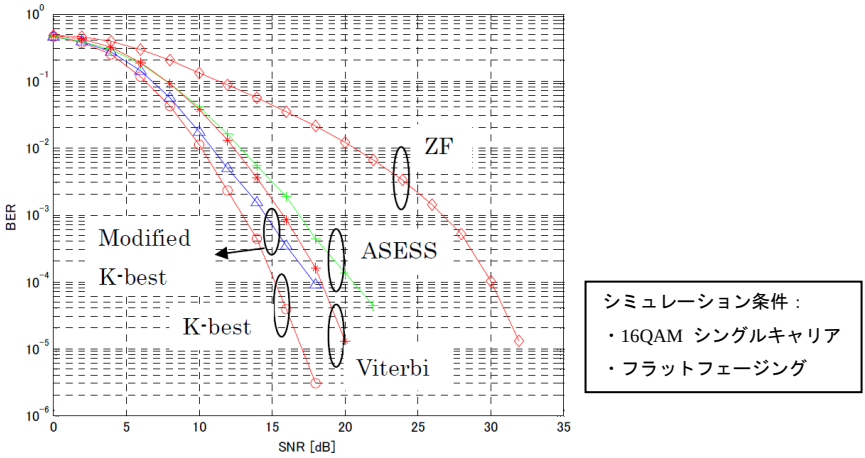


図 3・57　MLD アルゴリズムの BER 特性

(c) 演算量とダイバシチゲインの比較

表 3･8 に各種 MIMO デコーダに必要な演算量（乗算数）を示しておく．V-BLAST において擬似逆行列の演算自体は $O(M^3)$であるが，順序付けを行うことで $O(M^4)$ になる．また，ML については，K-best 法 [13] により硬判定最尤推定を行う場合の演算量である．

一方, 表 3･8 の演算量は SDM における干渉キャンセルアルゴリズム自身の演算量であり，OFDM-SDM の場合は，1 サブキャリア数当たりの演算量に相当する．すなわち，OFDM-SDM においては，全サブキャリア数を L とすると必要な演算量は表 3･8 に対し L 倍となり，更に増加することに注意する．

表 3・8　MIMO デコーダの演算量

アルゴリズム名	ダイバシチ	演算量（$M = N$）	演算量の根拠
線形復号	$N-M+1$	$O(M^3)$	逆行列
V-BLAST	$N-M+1+\alpha$（α の値は順序付けに依存）	$O(M^4)$	逆行列演算の M 回反復計算
MLD(K-best)	N	$O(M^6)$	候補選択アルゴリズム

以上のように，SDM-MIMO では，演算量と性能にはトレードオフの関係があることが，表 3･8 及び図 3･56，図 3･57 の結果から理解できる．

(5)　まとめ

本節では，MIMO システムにおいて，高速レート化を実現する空間分割多重 SDM 方式に

ついて述べた．MIMO デコーダ（干渉キャンセラ）として，ZF 法や MMSE 法などの線形復号法，及び線形復号を利用してダイバシチゲインを得る順序付け復号方式の V-BLAST 法，更に干渉キャンセル能力に最も優れた最尤推定法（MLD）についてそれぞれアルゴリズムについて述べた．また，各種アルゴリズムの演算量と性能にはトレードオフの関係があることを説明した．

本節では，MIMO-SDM に関する基礎的な内容に限定して記述している．更に発展的な内容として，

・送信ダイバシチゲインを得る時空間符号化（Space Time Coding：STC）[1)]
・ブロック時空間符号化 STBC との併用 [15)]
・最適ビットローディング法：Water Filling [16)]
・トレリス符号化変調との併用 [17)]

などがあげられる．1 Gbps を超える大容量 MIMO システムの標準化や開発 [18)] も現在進んでおり，今後の発展に期待したい．

■参考文献

1) S. M. Alamouti, “A simple transmit diversity technique for wireless communications,” IEEE Journal on Selected Areas in Communications, vol.16, no.8, pp.1451-1458, Oct. 1998.
2) 唐沢, “ディジタル移動通信の電波伝搬基礎,” コロナ社.
3) J. P. Kermoal, L. Schumacher, K. I. Pedersen, P. E. Mongensen, and F. Fredriksen, “A stochastic MIMO radio channel model with experimental validation,” IEEE Jour. Selec. Areas Commun., vol.20, no.6, pp.1211-1226, 2002.
4) http://www.radrix.com/?page_id=158
5) 大鐘武雄, 小川恭孝著, “わかりやすい MIMO システム技術,” オーム社, 2009.
6) S. Yoshizawa, Y. Yamauchi and Y. Miyanaga, “VLSI Implementation of a Complete Pipeline MMSE Decoder for a 4x4 MIMO-OFDM Receiver,” IEICE Trans.Fundamental, vol.EA91-A, no.7, pp.1757-1762, 2005.
7) J. Eilert, D. Wu, and D. Liu, “Efficient complex inversion for MIMO software defined radio,” IEEE ISCAS, pp.2610-2613, May 2007.
8) A. Burg, and et.al, “Algorithm and VLSI architecture for linear MMSE detection in MIMO-OFDM systems,” IEEE ISCAS pp.4102-4105, May 2006.
9) G. D. Golden, C. J. Foschini, R. A. Valenzuela and P. W. Wolniansky, “Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture, ” Electronics Letters, vol.35, no.1, pp.14-15, Jan. 7, 1999.
10) F. Sobhanmanesh, S. Nooshabadi, K. Kiseon, “A 212 Mb/s Chip for 4 × 4 16-QAM V-BLAST decoder,” Proc. of MWSCAS 2007, pp.1437-1440, 5-8 Aug. 2007.
11) A. Burg, et.al. , “VLSI implementation of the sphere decoding algorithm,” ESSCIRC 2004. Proceeding of the 30th European Solid-State Circuits Conference, 2004. pp. 303-306, Sep. 2004.
12) 中雅嗣, 久保博嗣, 村上圭司, “複数の超球を用いる Sphere Decoding による MLD の演算量削減に関する検討,” 電子情報通信学会論文誌, vol.J87-B, no.12, Dec. 2004.
13) M. Wenk, M. Zellweger, A. Burg, N. Felber, W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424 Mbps,” Proc. of ISCAS 2006, vol. 3, pp. 1151-1154, May 21-24, 2006.
14) K. Higuchi, H. Kawai, N. Maeda, M. Sawahashi, "Adaptive selection of surviving symbol replica candidates based on maximum reliability in QRM-MLD for OFCDM MIMO multiplexing,” IEEE Proc. of GLOBECOM 2004, vol. 4, pp. 2480-2486, Nov. 29 - Dec. 3, 2004.
15) S. Wahyul, M. Kurosaki and H. Ochi, “Design of 600 Mbps MIMO Wireless LAN System using GLST Coding and Its FPGA Implementation,” 2009 IEEE Radio and Wireless Symposium, San Diego, Jan. 2009.
16) S. Haykin, M. Moher, “Modern Wireless Communications,” Prentice-Hall, 2005.

17) Mohinder Jankiraman, “Space-time Codes And MIMO Systems,” Artech House Universal Personal Communications, 2004.
18) W. A. Syafei, Y. Nagao, M. Kurosaki, B. Sai, and H. Ochi, “Design of 1.2 Gbps MIMO WLAN for 4K Digital Cinema Transmission,” The 20th IEEE Personal Indoor and Mobile Radio Communications Symposium (PIMRC) 2009, Tokyo, Japan, Sep. 13-16, 2009.

■10 群 － 5 編 － 3 章

3-6 暗　号

（執筆者：反町　亨・鈴木大輔）［2009 年 1 月 受領］

暗号機能を実現するハードウェア回路には，対象となるセキュリティシステムの要求仕様により，高速性や小実装，省電力などの様々な異なる機能に対応することが要求される．このためには，一つの暗号機能をその特性に応じた複数の異なるアーキテクチャで実装が実現できることなどが重要である．本節では，一般的な暗号方式の概略について述べ，代表的な暗号技術として利用される共通鍵暗号方式と公開鍵暗号方式に関する機能（アルゴリズム）のハードウェア実装技術に関して説明する．また，暗号実装において，ハードウェア回路自身の安全な実装技術に関して説明する．

3-6-1 暗号方式概説

暗号技術は，軍事や外交目的で非常に長い歴史をもっているが，本節で対象とする技術は，1970 年以降に研究が活発化した現代暗号である．現代暗号の最大の特徴は，「暗号化・復号」を実施するためのアルゴリズムを一般に公開し，通信時に使用する鍵のみを秘密に扱うことで秘匿性を実現することである．

暗号方式には，**図 3･58** のように主に守秘目的で利用される共通鍵暗号方式と，署名，認証，鍵共有目的で利用される公開鍵暗号の二つに大別される．それぞれの暗号方式は，**表 3･9** のような特徴をもつため，お互いの特徴を補完しあうように組み合わせて利用されることが多い．例えば，公開鍵暗号技術を用いて秘匿通信したい相手の認証を行い，認証が成立した後に，秘匿通信を行うために利用する共通鍵暗号技術で使用する秘密鍵の共有を公開鍵暗号技術で行う．共有された秘密鍵を用いて，実際に送信したい情報を共通鍵暗号技術で暗号化して秘匿通信を行う．また，通信情報の正当性を保証するための署名や，正当な通信相手であるかを確認する手段である認証を実現するためには公開鍵暗号技術を使用する．

共通鍵暗号技術は，データの処理単位をある決まったブロック（64 ビット，128 ビットなど）によって処理されるブロック暗号と，データ系列として処理されるストリーム暗号に分けられる．公開鍵暗号技術は，素因数分解問題の困難性に安全性を依存した RSA 暗号と楕円曲線上の離散対数問題の困難性に安全性を依存した楕円曲線暗号が代表的である．

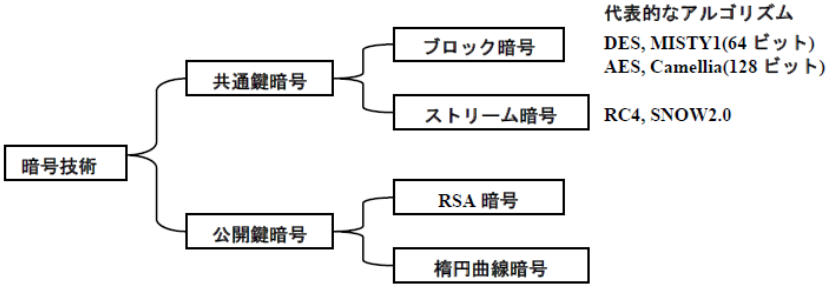


図 3・58　暗号技術の分類

表 3・9　共通鍵暗号技術と公開鍵暗号技術の比較

	共通鍵暗号技術	公開鍵暗号技術
暗号化鍵と復号鍵の関係	暗号化鍵＝復号鍵	暗号化鍵≠復号鍵
暗号化鍵	秘密	公開
復号鍵	秘密	秘密
鍵の共有	必要（秘匿）	不要（公開可）
秘密に保管すべき鍵	多数（相手ごとに必要）	自分の秘密鍵のみ
認証機能の実現	困難	容易
暗号化速度	高速	低速

3-6-2 共通鍵暗号の実装技術

共通鍵暗号の実装技術として，現在広く普及しているブロック暗号のハードウェア実装技術を取り上げる．ブロック暗号は，同じ構成のラウンド関数を N 段（$N>1$）積み重ねて処理されるアルゴリズムが一般的であり，実装アーキテクチャとして，以下の三つのタイプに大別される．なお，ブロック暗号アルゴリズムには，暗号化／復号を行うデータランダム化部と秘密鍵から拡大鍵を生成する鍵スケジュール部の二つの機能をもつが，本説明はデータランダム化部の実装の違いを示したものである．

(1) Fully Loop Unrolled（FLU）アーキテクチャ
(2) Loop アーキテクチャ
(3) Pipeline アーキテクチャ

(1)　Fully Loop Unrolled（FLU）アーキテクチャ

図 3･59 のように，アルゴリズムを構成する N 段のラウンド関数回路をすべて実装する方式である．N 段すべて処理するまで一度もレジスタに格納しないため，クリティカルパスは長くなるが，高速処理が可能である．ただし，すべての論理を回路で実現するため回路規模は大きくなる．回路規模や動作周波数の制約が厳しくなく，処理性能の要求が高いセキュリティシステムに適したアーキテクチャである．また，拡大鍵の入力などの制御回路が他のアーキテクチャと比較して容易に実現可能である．

(2)　Loop アーキテクチャ

図 3･60 のように，ラウンド関数回路を一つだけ実装し，N 回ループさせて 1 回の暗号化処理を実行する方式である．セレクタ回路や制御回路の増加が考えられるが，ラウンド関数の実装を 1 段分のみとすることにより，FLU アーキテクチャの実装と比較して約 $1/N$ の回路規模となることが期待される．また，1 段分処理するごとにレジスタに格納するため，クリティカルパスは FLU アーキテクチャの実装と比較して約 $1/N$ と非常に短くなる．しかし，レジスタに格納される回数が N 倍となること，セレクタや制御の複雑化から，FLU アーキテクチャの実装と比較して処理性能が低下する．ただし，レジスタやセレクタよりもラウンド関数の処理遅延の方が一般的に大きいため，性能が $1/N$ までは低下しない．処理性能よりも回

路規模や動作周波数の制約が厳しいセキュリティシステムに適したアーキテクチャである．Loop アーキテクチャは，実装するターゲットのアルゴリズムが偶数段であれば，2 段分のラウンド関数を実装し *N*/2 回ループにより実現可能なように，システムの要求仕様に沿った構成が可能である．

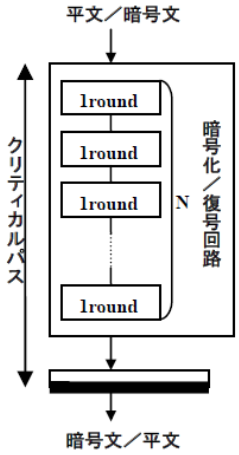


図 3・59　FLU アーキテクチャ

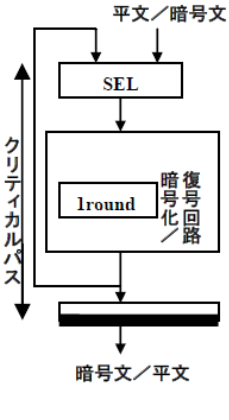


図 3・60　Loop アーキテクチャ

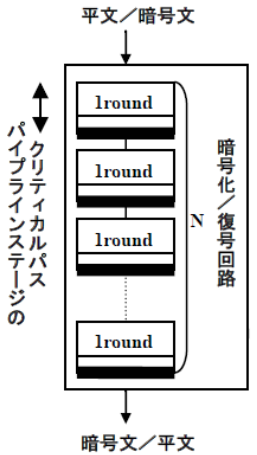


図 3・61　Pipeline アーキテクチャ

(3)　Pipeline アーキテクチャ

図 3･61 のように，FLU アーキテクチャと同様にアルゴリズムを構成する *N* 段のラウンド関数回路をすべて実装する方式である．ただし，1 段のラウンド関数ごとにレジスタを挿入し，パイプライン処理を行うことで，1 クロックごとに処理データ入力が可能となる．レジスタや制御回路の増加が考えられ，FLU アーキテクチャの実装以上に回路規模は大きくなることが予想される（実装ターゲットが FPGA のように論理回路とレジスタがセットの構成となっている場合は，Pipeline アーキテクチャで各段に挿入するレジスタの増加は実際の回路規模に影響しないこともある）．

しかし，クリティカルパスが 1 段のラウンド関数の処理（パイプラインステージ）と短く，かつ，1 クロックごとに処理可能であることから，1 ブロックの暗号化処理遅延は FLU アーキテクチャより若干増加するが，処理性能（スループット）は *N* 倍に近い高速処理を実現可能である．

回路規模の制約があり，かつ，処理性能要求の厳しい場合の実現方法には，inner-Pipeline アーキテクチャといった方式も存在する．inner-Pipeline アーキテクチャは，Loop アーキテクチャのループする暗号化回路部分をパイプライン化する方式である．制御回路はより複雑になるが，回路規模と処理性能のトレードオフを考えた場合，有効な方式である．

表 3・10　ブロック暗号の実装アーキテクチャの比較

	FLU アーキテクキチャ	Loop アーキテクチャ	Pipeline アーキテクチャ
回路規模	1	約 1/*N*	1＋α
クリティカルパス	1	約 1/*N*	約 1/*N*
処理性能（スループット）	1	1/*N*～1 の間	約 *N*
暗号利用モード対応	5 モードとも○	5 モードとも○	ECB, CTR モードのみ○

※ LU アーキテクキチャの規模，性能を 1 としたときの相対値

ブロック暗号をシステムで使用する場合，1 ブロックサイズ以上のデータを暗号化するための暗号利用モードを用いる．暗号利用モードは，ISO で ECB（Electronic CodeBook）モード, CBC（Cipher Block Chain）モード, OFB（Output FeedBack）モード, CFB（Cipher FeedBack）モード，CTR モードの五つが標準化されている．CBC，OFB，CFB モードは，処理時に前のブロックの暗号化情報などが必要となるため，並列処理不可である．したがって，FLU アーキテクチャと Loop アーキテクチャはすべてのモードに対応可能であるが，Pipeline アーキテクチャは並列処理可能な ECB，CTR モードのみに対応可能である．

また，IC カードや TPM のようなセキュリティチップに組み込む暗号化回路の要求仕様には，単純に 1 段のラウンド関数を実装して実現する Loop アーキテクチャでは，回路規模を満足しない場合が多い．この場合には，実装するアルゴリズムのラウンド関数の内部構成ロジックに踏み込んだ最適化実装が必要である．ブロック暗号の重要な構成要素として，非線形変換な置換表（S-box）があげられる．

S-box がアルゴリズム全体の処理の大きな部分を占めていることが多く，S-box をどのように実現するかが，回路規模や処理性能に大きく影響する．標準化され多くのシステムで利用されている AES や MISTY のようなブロック暗号の S-box は，暗号の安全性評価が可能な GF(2^m) のガロア体上の演算により構成されていることが多い．

ハードウェア実装では，文献 1), 2), 3) などのように，ソフトウェア実装のようにテーブル参照のタイプの回路記述を行い，論理合成ツールによる最適化を期待するより，S-box が GF(2^m) のガロア体上の演算であることに着目し，ビット単位の論理演算展開や，部分体への分割による回路実装を行うことが回路規模や処理性能に有効である場合が多い．

3-6-3 公開鍵暗号の実装技術

まず，公開鍵暗号の具体的な回路アーキテクチャについて検討する前に，設計者が整理すべき項目について述べる．以下にその項目を示す．

(1) 演算対象とする体（GF(p)，GF(2^m) など）
(2) 演算対象とする体のビット長（固定，可変など）
(3) コプロセッサで実行する演算範囲（多倍長加減乗算，モンゴメリ乗算，楕円加算/2 倍算，べき乗剰余演算，スカラ倍算など）

(1)は楕円曲線暗号を構成するために必要な基本演算に係わる項目である．GF(2^m) の演算は XOR 演算が基本となるため，加算時に桁上がりが発生しない．このため小型で高速な実装が可能となる．一方，GF(p) の演算は通常の加算あるいは乗算器を基本として構成される．このため，回路規模や処理性能は GF(2^m) に劣るが，RSA 暗号などのほかの暗号アルゴリズ

ムと親和性が高い．ハードウェア構成を工夫すれば，複数の暗号アルゴリズムの高速化に寄与するコプロセッサを構成できる．

(2)はセキュリティパラメータである鍵長に影響を与える項目である．体のビット長を固定し，特殊な素数（例えば，一般化メルセンヌ素数）や既約多項式に限定した演算を行うコプロセッサを考えた場合，ある範囲で体のビット長を可変とするコプロセッサよりも回路規模や処理性能を改善できる場合がある．ただし，演算可能な楕円曲線やその鍵長が限定されるため，アプリケーションも限定される．

(3)は CPU とコプロセッサ間のデータ転送能力や，コプロセッサが占有可能な回路面積に応じて設定する必要がある．多倍長の加減乗算やモンゴメリ乗算 [4] などの基本演算のみをサポートするコプロセッサを考えた場合，演算結果を格納する記憶領域として CPU との共有メモリを使用することになる．このため，コプロセッサの回路規模は専用の記憶領域をもつより構成よりも縮小されるが，メモリとコプロセッサ間のデータ転送がオーバヘッドとなる．また，次節 3-6-4 で述べるように，安全な実装を実現する際には注意が必要となる．

以下の説明では，最も基本的でかつ汎用性の高い公開鍵暗号の回路アーキテクチャについて述べる．

(1) モンゴメリ乗算

剰余乗算 $A \cdot B \bmod M$（A, B, M は整数）は，RSA 暗号や楕円曲線暗号などの公開鍵暗号における処理で重要となる演算の一つである．例えば，RSA 暗号では，べき乗剰余演算 $Y = X^E \bmod M$ により暗号化や復号を行うが，このべき乗剰余乗算は剰余乗算の繰り返しによって実行できる．また，楕円曲線暗号においても，剰余乗算を含む剰余演算にて暗号化や復号処理が行われる．したがって，公開鍵暗号のハードウェア実装では，剰余乗算を処理する回路アーキテクチャが重要なポイントとなる．

モンゴメリ乗算は，剰余乗算を効率的に処理するアルゴリズムの一つである．このアルゴリズムは，通常の法演算で必要な除算処理をビットシフトなどに置き換え，単調な積和演算の繰り返しで剰余乗算を処理できるため，ハードウェア実装に適している．モンゴメリ乗算には様々なバリエーションが存在するが，以下に最も基本的なアルゴリズムを示す．

アルゴリズム 1 モンゴメリ乗算

入力　$0<=A,B<M$ となる整数 $M=(m_{n-1}...m_1m_0)_K$，$A=(a_{n-1}...a_1a_0)_K$,
$B=(b_{n-1}...b_1b_0)_K$, $\gcd(M,K)=1$ となる $R=K^n$，$m' = -M^{-1} \bmod K$

出力　$S = \mathrm{MM}(A,B)=ABR^{-1} \bmod M$, $0<=S<M$

1: $S \leftarrow 0$（$S = (s_{n-1}...s_1s_0)_K$ とする）
2: **for** $i = 0$ to n-1 **do**
3: $u_i \leftarrow (s_0+b_i a_0)\, m' \bmod K$.
4: $S \leftarrow (S + u_i M + b_i A) / K$.
5: **end for**
6: **if**($S \geqq M$) **then**
7: $S \leftarrow S - M$
8: **end if**
9: **return** S

RSA 暗号や楕円曲線暗号では，M として大きな素数やその合成数が用いられるので，K を 2 のべき（2^k）とするのが一般的である．これによりアルゴリズム 1 における mod K の処理や K による除算はそれぞれビットの切捨てやシフト処理で実現できる．

具体的な例として $Y=A^{11} \bmod M$ $(1 \leqq A < M)$ を計算することを考える．まず，あらかじめ $R'= R^2 \bmod M$ を計算しておく．ここで，R は定数なので，同じ法 M に対して R' は 1 回だけ計算すればよい．次に A_1= MM(A, R')=$A \cdot R^2 \cdot R^{-1} \bmod M=AR \bmod M$ を計算する．同様に A_2= MM(A_1, A_1)，A_3＝MM(A_2, A_2)，A_4= MM(A_1, A_3)，A_5= MM(A_4, A_4)，A_6= MM(A_1, A_5)= $A^{11}R \bmod M$ の順に求める．最後に Y = MM(1, A_6) より $A^{11} \bmod M$ を得ることができる．なお，A_1 を求める処理をモンゴメリ変換，MM(1, A_6) のように R 倍されたデータに対して R^{-1} を乗ずる処理をモンゴメリ逆変換と呼ぶことがある．

桁数の大きい指数（例えば 1024 bit）に対してべき乗剰余を考えた場合，R' の算出，モンゴメリ変換，及びモンゴメリ逆変換によって増加する計算量よりも，除算処理が不要となることで削減される計算量の方が圧倒的に大きい．また，一般に乗算回路は除算回路よりも小さい回路規模で実装可能である．

(2) モンゴメリ乗算回路

次にアルゴリズム 1 を処理する回路アーキテクチャについて考える．前述のように $K = 2^k$ とするならば，アルゴリズム 1 の主な処理は多倍長の積和演算となる．積和演算を処理する回路アーキテクチャの例を**図 3･62** に示す．図 3･62 は Digital Signal Processor（DSP）などの高速積和演算を目的としたプロセッサにおいて基本となる回路アーキテクチャであり，広く用いられている．

以下では，図 3･62 の積和演算器を用いたモンゴメリ乗算の具体例を示す．まず，図 3･62 の各入力（a, b, c）は 16 ビットと仮定し，1 024 ビットのモンゴメリ乗算を処理することを想定する．この場合，$K = 2^{16}$ と考え，$R = 2^{1024}$ ＝ $(2^{16})^{64}$ とし，入力 A, B, M をそれぞれ 16 ビットごとに 64 分割してモンゴメリ乗算を行う．3, 4 行目の処理は 2 回の乗算があるので，それぞれ図 3･62 の積和演算器を逐次的に用いて行い，演算結果の下位 16 ビットを切り捨てることで K による除算処理を行う．また，6, 7 行目の処理は $S－M$ を実行し，符号に応じて S か $S－M$ を選択すれば実現できる．

この例では，64 回のループ処理に 16×16 ビットの積和演算が 2 回，1024×16 ビットの積和演算が 2 回，1024 ビットの減算が 1 回あるので，およそ 13000 サイクルで 1 回のモンゴメリ乗算が可能となる．

(3) モンゴメリ乗算回路の高速化

ここで，より高速なモンゴメリ乗算を実現するために，積和演算の並列度を高めることを考える．その方法の一つとして高基数化が考えられる．これは，例えば k ビットの乗算器を 2 k ビットにし，一度に乗算するビット長を増やすことを意味する．もう一つの方法として，**図 3･63** に示すように，図 3･62 の積和演算器をカスケードに接続する方法である．この二つのアプローチにはそれぞれ一長一短があるが，汎用 LSI として流通している DSP や Field ProgrammableGate Array（FPGA）のアーキテクチャは後者のアプローチと親和性が高い．例として文献 5）や文献 6）などの実装例があげられる．

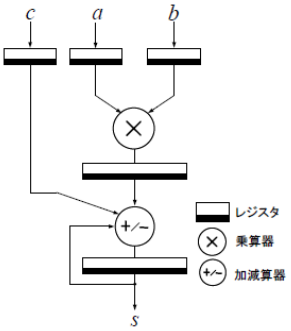


図 3・62　積和演算器

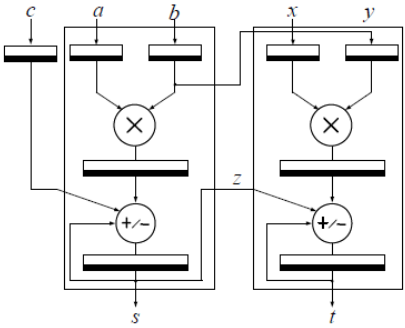


図 3・63　積和演算器のカスケード接続

3-6-4 安全な暗号アルゴリズムの実装

暗号アルゴリズムと他の信号処理の実装で大きく異なる実装要件の一つに，“安全な実装”がある．これは暗号アルゴリズムの安全性ではなく，ソフトウェアやハードウェアによる実装そのものの安全性である．特に，暗号アルゴリズムが実装されたモジュールから生じる副次的な情報，すなわち，処理時間や消費電力といった情報（サイドチャネル情報）を解析する“サイドチャネル攻撃”と呼ばれる攻撃法は，強力な攻撃法として注目されている．いかに数学的に安全な暗号アルゴリズムを用いても，その実装次第では，これらの攻撃によって，秘密情報が漏洩してしまう可能性がある．

サイドチャネル攻撃の簡単な例として，前述したべき乗剰余演算に対する電力情報による攻撃について述べる．モンゴメリ乗算の入力が同じ場合，すなわち MM(X, X) となる自乗算と，MM(X, Y) となる乗算が**図 3･64** のように電力波形から区別可能とする．この場合，電力波形から指数の情報が特定可能となる．RSA 暗号の復号処理では，指数は秘密情報であり，電力情報から秘密情報が漏洩してしまうことになる．

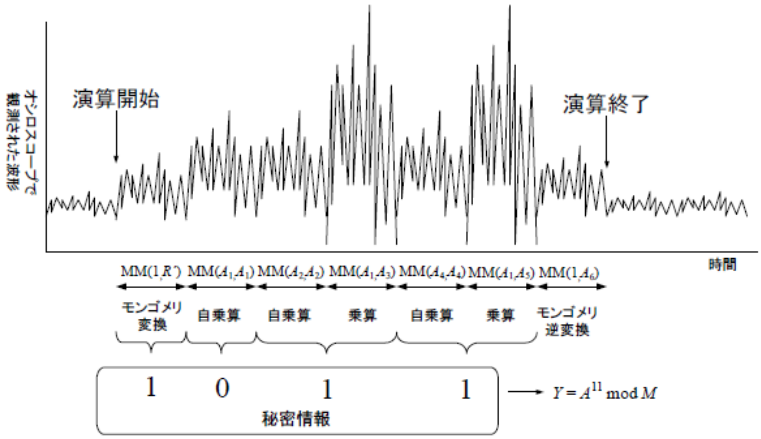


図 3・64　べき乗剰余演算の単純電力解析

このほかにも着目するサイドチャネル情報の種類や解析の種類に応じて，様々な攻撃手法が存在するが，代表的なものを**表 3･11** に幾つか示す．表 3･3 以外にも，これらを組み合わせた攻撃や LSI の誤動作を利用した攻撃など多くの攻撃手法が知られており，今後もその種類は増えていくと予想される．IC カードや組込み機器における暗号の実装では，このような攻撃に対する対策が必要である．

表 3・11　サイドチャネル攻撃のバリエーション

タイミング攻撃（TA）	条件分岐の成立の有無，キャッシュのヒット/ミスヒットなどによって変化する処理時間の違いから鍵を推定する攻撃法である．
電力攻撃（PA）	LSI の消費電力の変化を解析して鍵を推定する攻撃手法．単純電力攻撃（SPA），差分電力攻撃（DPA）などがある．
電磁波攻撃（EMA）	LSI から発生する電磁波を解析する攻撃手法．電力攻撃と同様に，単純電磁波攻撃（SEMA），差分電磁波攻撃（DEMA）などがある．

■参考文献

1) M. Matsui, “New Block Encryption Algorithm MISTY,” FSE97, Jan. 1997.
2) A. Satoh and S. Morioka, “Unified Hardware Architecture for 128-bit Block Ciphers AES and Camellia,” CHES 2003, LNCS2779, pp. 304-318, 2003.
3) A. Satoh and S. Morioka, “Hardware-Focused Performance Comparison for the Standard Block Ciphers AES, Camellia, and Triple-DES,” ISC 2003, LNCS2851, pp. 252-266, 2003.
4) P. L. Montgomery, “Modular Multiplication without Trial Division,” Mathematics of Computation, vol.44, no.170, pp.519-521, 1985.
5) K. Itoh, M. Takenaka, N. Torii, S. Temma, and Y. Kurihara, “Fast Implementation of Public-Key Cryptography on a DSP TMS320C6201,” CHES’ 99, LNCS1717, pp. 61-71, 1999.
6) D. Suzuki, “How to Maximize the Potential of FPGA Resources for Modular Exponentiation,” CHES 2007, LNCS 4727, pp. 272-288, 2007.