

○○論文

マルチメディア
集積回路システム
処理方法の研究

（A Study on Multimedia data processing LSI-system
method）

熊木 武志

立命館大学理工学部電子情報工学科

2006年12月

内 容 梗 概

近年，データ通信インフラストラクチャの急速な整備，発展やモバイル機器の普及で，ユーザが高品質な画像や音声等のマルチメディアデータを取り扱う機会が増加している。一般に，LSIによるマルチメディアデータ処理は，算術演算や論理演算を主体とした繰り返し演算の組み合わせによる処理と，データを符号化するテーブルルックアップ処理に分けることができる。従来の汎用プロセッサやDSPでは，繰り返し演算処理を動作周波数や処理の並列度を上げることで性能向上を図ってきた。特にモバイル用途等の場合には，消費電力や面積の点において，仕様を満足するようなマルチメディアデータ処理を実現することが重要な課題となってきており，専用ハードウェアでは，製品の仕様やアルゴリズムの変更に柔軟に対応することができない点が問題となっている。

一方，テーブルルックアップ符号化処理は，代表的なものにハフマン符号化と呼ばれる可逆圧縮アルゴリズムがあり，JPEG，MP3，MPEG2，及びH.264等多くのアプリケーションで利用されている。このアルゴリズムはハードウェアリソースの点で効率的な並列化が容易ではないため，逐次的に処理されることが多い。そのためアプリケーションによっては，処理全体の約30%近くを占める場合もあり，高速化のボトルネックとなっている。

本論文では，マルチメディアデータ処理LSIの高性能化を図るために，機能メモリの一種であり，一致検索処理を高速に実現することが可能なCAM (Content Addressable Memory) を用い，テーブルルックアップ処理の並列化による高速処理，ハフマン符号化圧縮アルゴリズムを効率よく処理することによる高圧縮化，及び汎用性に重点を置いたプログラマビリティの3つに焦点を定めLSIアーキテクチャの開発を行った。

高圧縮化に関しては，CAMを用いたパイプラインハフマン符号化処理を行いながら，符号化テーブルをリアルタイムにデータの出現頻度に応じて適応的に再構築するアーキテクチャを開発した。符号化テーブルは，アクティブ用とアップデート用に分け，適時切り替えることによってデータの高圧縮化を実現する。開発アーキテクチャをFPGAに実装し，性能を評価した後に，画像圧縮の代表的なアルゴリズムであるJPEGアプリケーションに対して評価を行った。その結果，最大約28%の圧縮率の向上が得られ，処理速度に関しても携帯電話やデジタルカメラの仕様を十分に満足することができた。

高速処理に関しては，テーブルルックアップ符号化処理の並列化に着目した。従来の技術では，高速にテーブル変換処理を行うことのできるCAM等のアーキテクチャを並列に配置した場合，面積や消費電力が増大するため実現が難しい。そこで，符号化テーブルを1つだけ持ち，面積増加を抑えた上で，並列にテーブル変換処理を実現することのできるマルチポートCAMを新たに開発した。マルチ

ポート CAM は、並列度が 4 の場合、CAM を並列に 4 つ配置した場合と比較して、面積の増加率は約 1.57 倍で済むため、AT (Area Time) 積も通常の CAM と比較して約 37.5% と低い値を示すことが分かった。更に、符号化アプリケーションの特長をうまく活かすことで、マルチポート CAM の処理能力を更に向上させた後に、ハフマン符号化に適用し、性能を評価した。その結果、符号化にかかるクロックサイクル数は従来の汎用 DSP と比較して最大約 93% のクロックサイクルの削減を実現し、面積当たりの処理能力も DSP と比較して最大 3.8 倍の値を得ることができた。また、FPGA に実装、及び ASIC 向けに論理合成した結果、ポート数が 16 までの増加に対して、動作周波数は、大幅な減少が見られず、面積は線形に変化するなどスケーラビリティが高いことを立証した。

テーブルルックアップ処理のボトルネックは、CAM を用いた新しいアーキテクチャを適用することで高圧縮処理と高速処理を実現した。本論文では、テーブルルックアップ処理の性能向上に加えて、繰り返し演算処理を、従来の汎用プロセッサや DSP と比較して、大幅に並列度を向上させた SRAM ベースの超並列 SIMD 型プロセッサを開発することで高速処理を実現した。SRAM ベース超並列 SIMD 型プロセッサは、データの処理ベクトルを従来の方式から変更することで並列度の向上を得る。また、演算器 (PE : Processing Element) とデータレジスタ領域を密結合することによって PE、メモリ間の I/O 転送にかかる消費電力を削減した。90 nm 7Cu CMOS テクノロジの実装結果では、面積 3.1 mm²、消費電力 250 mW であり、16 ビットの加算処理では動作周波数 200 MHz で 40 GOPS (Giga Operation per Second) の性能を得ることができた。

次に、CAM ベーステーブルルックアップ符号化アーキテクチャの開発を元に、超並列 SIMD 型プロセッシングアーキテクチャに、CAM を付加する CAM ベース超並列 SIMD 型プロセッシングアーキテクチャを開発した。これはマルチメディア処理のボトルネックであるテーブルルックアップ符号化処理を高速に行うことを利用とし、繰り返し演算処理を SIMD 型プロセッシングユニットで行うことにより高速化を実現するアーキテクチャである。このアーキテクチャは CAM、SRAM、及び PE を融合した構造をとっているため小面積で高並列を実現し、単位面積当たりの演算能力を上げることで、動作周波数を抑え、低消費電力化を実現している。更に、CAM と SRAM アレイのデータ転送を工夫することにより、パイプライン処理も実現している。提案アーキテクチャはメモリベースであり、データやプログラムを交換することによって柔軟に様々なマルチメディアアプリケーションに適用が可能である。90 nm 7Cu CMOS テクノロジの実装見積もりでは、面積 3.49 mm² となり、新規に追加した CAM 及び周辺回路の消費電力は 32.4 mW であった。FPGA に実装を行い基本動作を検証した後、JPEG アプリケーションにおいては、最大約 86% のクロックサイクル数の削減を可能とした。単位面積当たりの処理性能では、DSP の約 4.4 倍となり、その有効性を示すことができた。

目 次

内容梗概	i
第 1 章 序論	1
1.1 研究背景	1
1.2 本研究の目的	6
第 2 章 CAM ベーステーブルルックアップ符号化処理	9
2.1 はじめに	9
2.2 ハフマン符号化	10
2.2.1 静的ハフマン符号化方式 (Static Huffman Coding)	11
2.2.2 動的ハフマン符号化方式 (Adaptive Huffman Coding)	12
2.3 リアルタイム符号化テーブル最適化アーキテクチャ	13
2.3.1 最適化ハフマンテーブルによる符号化	13
2.3.2 リアルタイム符号化テーブル最適化の原理	16
2.3.3 アーキテクチャ	18
第 3 章 マルチポート CAM ベース並列テーブルルックアップ符号化処理	23
3.1 はじめに	23
3.2 フレキシブルマルチポート CAM	24
3.2.1 CAM のマルチポート化	24
3.2.2 FMCAM の概要	26
3.2.3 アーキテクチャ	28
3.2.4 マルチポートの実現方法	31
3.2.5 FPGA への実装結果	33
3.2.6 性能評価	35
3.3 テーブルルックアップ用マルチポート CAM	40
3.3.1 マルチプル／シングルサーチモード (Multiple/Single Search Mode)	41
3.3.2 カウンタ値設定モード (Counting Value Setting Mode)	42
3.3.3 カテゴリ結合モード (Scalability of Categorization Structure)	43
3.4 アーキテクチャ	46
3.4.1 ポートブロック (Port block)	46
3.4.2 カテゴリブロック (Category block)	47

3.4.3 コントローラ (Controller)	48
3.5 FPGA / ASICへの実装結果	48
3.6 ハフマン符号化への適用と評価	54
3.6.1 並列テーブルルックアップ処理の概念	54
3.6.2 評価・検証	55
3.7 リアルタイム符号化テーブル最適化アーキテクチャとの融合	58
3.7.1 処理の概要	58
3.7.2 ハフマン符号化適用時における性能評価	59
第4章 CAMベース超並列 SIMD型プロセッシングアーキテクチャ	63
4.0.3 動作概要	63
4.0.4 VLSI設計結果	72
4.1 CAMと超並列 SIMD型プロセッシングアーキテクチャの融合	73
4.1.1 融合処理の概要	73
4.1.2 アーキテクチャ	77
4.1.3 命令セット	81
4.1.4 動作概要	83
4.1.5 VLSI実装結果	86
4.2 画像処理アルゴリズムへの適用	90
4.2.1 JPEG方式の概要	90
4.2.2 DCT (Discrete Cosine Transform)	91
4.2.3 ハフマン符号化	100
4.2.4 性能評価	107
4.3 まとめ	111
第5章 結論	113
5.1 研究成果	113
5.2 研究成果の応用と将来展望	118
5.2.1 研究成果の応用	118
5.2.2 将来展望	120
参考文献	120
謝辞	129
発表論文リスト	131

図 目 次

1.1	マルチメディアデータ処理に要求される性能.	5
1.2	ビデオ圧縮伸長技術.	6
2.1	ハフマン木構築例.	11
2.2	リアルタイム符号化テーブル最適化アーキテクチャ構成.	13
2.3	リアルタイム符号化テーブル最適化アーキテクチャブロック図.	14
2.4	最適化された符号化テーブルを用いたリアルタイム圧縮処理動作原理.	16
2.5	ハフマンテーブルの最適化及びアップデートのフローチャート.	17
2.6	アサインモジュールブロック図.	18
2.7	シャドウテーブルのアップデート処理手順.	21
2.8	スワップモジュールブロック図.	21
3.1	ワード長 d -bit, ワード数 2^a の FMCAM.	26
3.2	FMCAM の内部構成.	28
3.3	コンテンツモジュール (CoM) 内のカテゴリ.	29
3.4	セレクタモジュール (SeM).	30
3.5	ポートモジュール (PoM).	31
3.6	リングカウンタを用いた比較方式.	33
3.7	FMCAM, MCAM 及び並列 CAM のハードウェア量の比較.	39
3.8	Adapted FMCAM の全体図.	40
3.9	Adapted FMCAM のブロック図.	41
3.10	Adapted FMCAM のメモリ空間.	42
3.11	Adapted FMCAM による一致検索時の動作波形 : (a) マルチプルサーチモード, (b) シングルサーチモード.	44
3.12	Adapted FMCAM による一致検索時のカウンタ動作 : (a) 通常モード, (b) カウンタ値設定モード.	45
3.13	Adapted FMCAM におけるポートモジュールのブロック図.	47
3.14	配置配線マッピング結果.	49
3.15	90 nm 7Cu CMOS テクノロジによる 16 ポート Adapted FMCAM の論理合成結果.	50

3.16 Adapted FMCAM のポート数増加に対する FPGA (Xilinx XC4VLX160)への実装結果 : (a) 最大動作周波数, (b) ゲート数.	52
3.17 Adapted FMCAM のポート数増加に対する ASIC (90 nm CMOS テクノロジ)への実装結果 : (a) 最大動作周波数, (b) 実装面積.	53
3.18 並列テーブルルックアップ処理の概念図.	54
3.19 ベンチマーク用画像.	55
3.20 FMCAM アーキテクチャと DSP のハフマン符号化処理クロックサイクル数の比較.	56
3.21 リアルタイム符号化テーブル最適化アーキテクチャと FMCAM の融合.	59
3.22 FMCAM 適用時のアサインモジュールブロック図.	60
3.23 ベンチマーク用画像.	60
4.1 超並列 SIMD 型プロセッサの加算処理.	64
4.2 超並列 SIMD 型プロセッサの減算処理 (1/2).	65
4.3 超並列 SIMD 型プロセッサの減算処理 (2/2).	66
4.4 超並列 SIMD 型プロセッサの乗算処理 (1/2).	67
4.5 超並列 SIMD 型プロセッサの乗算処理 (2/2).	68
4.6 超並列 SIMD 型プロセッサの除算処理 (1/3).	69
4.7 超並列 SIMD 型プロセッサの除算処理 (2/3).	70
4.8 超並列 SIMD 型プロセッサの除算処理 (3/3).	71
4.9 超並列 SIMD 型プロセッサのチップ写真.	72
4.10 直交 CAM のブロック図.	75
4.11 テーブルルックアップインターフェースモジュールのブロック図.	77
4.12 テーブルルックアップインターフェースコントローラブロック図.	79
4.13 超並列 SIMD 型プロセッサのメモリ空間.	81
4.14 テーブルルックアップインターフェースモジュールの命令フィールド構成.	82
4.15 テーブルルックアップインターフェースモジュールを利用したデータ処理フロー.	84
4.16 インタフェースモジュールによるハフマン符号化の例.	85
4.17 インタフェースモジュールによるフィードバック処理の例.	86
4.18 直交 SRAM セルのレイアウト.	88
4.19 32 bit × 32 word の直交 SRAM セルアレイ.	89
4.20 JPEG 方式の分類図.	90
4.21 JPEG 基本方式の処理フロー.	91
4.22 画像と周波数.	92
4.23 DCT 基本パターン群.	93

4.24 CAM ベース超並列 SIMD 型プロセッサによる、高速 DCT データフ ロー.	94
4.25 CAM ベース超並列 SIMD 型プロセッサによる、高速 DCT 处理手法.	95
4.26 CAM ベース超並列 SIMD 型プロセッサによる垂直方向高速 DCT 处 理.	96
4.27 CAM ベース超並列 SIMD 型プロセッサによる水平方向高速 DCT 处 理.	97
4.28 CAM ベース超並列 SIMD 型プロセッサ及び DSP による JPEG 处理 クロックサイクル数の比較.	98
4.29 CAM ベース超並列 SIMD 型プロセッサによるハフマン符号化処理.	101
4.30 192×128 , 480×600 及び 512×480 の検証用画像.	103
4.31 256×256 の検証用画像.	104
4.32 512×512 の検証用画像.	104
4.33 600×480 の検証用画像.	105
4.34 720×576 , $1,024 \times 768$ 及び $1,500 \times 1,125$ の検証用画像.	106
4.35 ハフマン符号化処理の信頼性検証手順.	107
4.36 各アーキテクチャによるクロックサイクル数の比較 (2,048 entry, Y, C_b , 及び C_r 画像).	110
5.1 階層並列構造 SIMD 型プロセッシングアーキテクチャ.	121

表 目 次

1.1	IT 市場の予測.	3
1.2	携帯電話, e ビジネスライフ及びプラットフォーム各市場の予測.	4
3.1	実装に用いた FPGA ボード.	34
3.2	FMCAM の FPGA への実装結果.	35
3.3	構成方法による比較器数及びクロックサイクル数.	36
3.4	各種 CAM のスペック及び AT 積.	37
3.5	一致検索時における, ハフマン符号化クロックサイクル数の平均値.	57
3.6	ハフマン符号化における Original/Adapted FMCAM と並列 DSP の性能比較.	58
3.7	処理クロックサイクル数と圧縮データサイズの算出結果.	61
4.1	90 nm CMOS 技術による, 超並列 SIMD 型プロセッサの実装結果.	72
4.2	テーブルルックアップモジュール構築のためのメモリモジュール組み合わせ案.	76
4.3	メモリモジュール組み合わせ案に対する機能実現の評価結果.	76
4.4	ソフトマクロによる, テーブルルックアップインターフェースモジュールの実装結果.	87
4.5	ハードマクロによる, テーブルルックアップインターフェースモジュールの実装結果.	88
4.6	単位面積当たりの処理性能の比較.	98
4.7	JPEG 処理クロックサイクル数の比較 (Y 画像).	109
4.8	単位面積当たりの処理能力の比較.	110

第1章 序論

近年，光ファイバー通信網の普及や地上デジタル放送の開始に伴いデータ通信インフラストラクチャが急速に整備されてきている。これにあわせて，携帯電話に代表されるモバイル機器，及び，デジタルカメラに代表されるデジタル家電製品も普及してきており，高品質な画像や音声等のマルチメディアデータを取り扱う機会が増加している。これに加え，マルチメディアアプリケーションの発展やユーザニーズの多様化により，マルチメディアデータを処理する LSI には，大量のマルチメディアデータを高速，高圧縮かつプログラマブルに処理することが求められている。一般にマルチメディアアプリケーションにおける処理は，算術論理演算を大量のデータに繰り返し行う演算処理と，あらかじめ符号化テーブルを用意し，テーブル引きによりデータの符号化を行う，テーブルルックアップ処理の2つに大きく分けることができる。特にテーブルルックアップ処理は，並列化が難しいため [1]–[3]，大量のマルチメディアデータに対し逐次処理となってしまい，これまで高速処理のボトルネックとなっていた。

本論文では，マルチメディアデータ処理 LSI の性能を向上させるために，繰り返し演算処理を超並列の SIMD (Single Instruction Multiple Data) 型プロセッサで行い，テーブルルックアップ処理及び高圧縮処理を CAM (Content Addressable Memory) によって行う，CAM ベースの新しい小面積，低消費電力な高性能マルチメディアデータ処理 LSI アーキテクチャの開発を行う。

1.1 節では，近年のマルチメディアデータ処理に関する様々な現状を述べ，それらに対して??節にてマルチメディアデータ処理 LSI に求められる技術を述べる。??節では，求められる技術を元に，従来のアーキテクチャによるマルチメディアデータ処理を議論する。この議論を通してマルチメディアデータ処理 LSI が解決すべき課題を示した後，1.2 節にて本論文の目的である高性能なマルチメディアデータ処理 LSI のアーキテクチャについて議論する。最後に??節において，本論文の全体構成を述べる。

1.1 研究背景

近年，光ファイバー通信網の整備による FTTH (Fiber To The Home) の普及や地上デジタル放送の開始に伴うデジタルコンテンツの発展等大容量のデータを取り扱うためのインフラストラクチャ環境が急速に発展している。これにあわせて，

携帯電話, PDA (Personal Digital Assistants) に代表されるモバイル機器, 及びデジタルカメラ, デジタルビデオ等に代表されるデジタル家電製品も普及してきている。表 1.1 に示す国内 IT (Information Technology) 主要市場分析, 及び今後の市場規模予測¹によると, 家庭向け FTTH 分野は, 2010 年には加入世帯数 1,488 万, 市場規模 6,483 億円にまで伸びると予測されている。更に地上デジタル放送は 2011 年に完全移行となるが, 2010 年には, 普及世帯数 3,512 万, 市場規模 1 兆 8,507 億円に達し, 普及率は日本総世帯数の 70% を超える規模となる。次に, 表 1.2 に示すように, 携帯電話, e ビジネスライフ, 及びプラットフォーム市場の予測によると, 2006 年のワンセグ (携帯電話向け地上デジタル放送) 開始に伴い, 携帯電話や MP3 プレーヤをターゲットとした音楽配信サービスが急速に成長している。このことから 2010 年にはこれらのサービスに関わる市場が, 現在の約 5.3 倍にまで膨れ上がるという試算がなされている。

以上の背景から, 今後ユーザが高品質かつ大容量のマルチメディアデータを手軽に取り扱う機会がますます増加することは容易に推測できる。ここで述べるマルチメディアデータとは, “文字だけでなく音声やカラー映像を駆使したリッチ・コンテンツ情報” [4] を意味し, マルチメディアとは, “電話, ラジオ, テレビ及びコンピュータなど, 様々な機器が取り扱うデータを互いにリンクさせ, 融合させたもの” [5] と定義することができる。以下にマルチメディアデータが使用される具体的な例を以下に挙げる。

- ビデオ
Video CD, DVD player, デジタルカメラ等
- オーディオ, 音声
MP3 プレーヤ, 音声入力, 音声自動翻訳等
- グラフィックス
デジタルテレビ, 家庭用ゲーム機, アミューズメントマシン, カーナビゲーションシステム等
- 通信・認識
ロボット, 指紋認証等

これらのアプリケーションが単体, もしくはいくつか組み合わさることにより, これからマルチメディア機器は, オーディオ, 画像, グラフィックス等大量のマルチメディアデータをリアルタイムに処理する高い性能が必要となるのは容易に推測できる。また, これに加えて高スループットの通信機能, 安全性の高いセキュリティ機能, 及び高度な認識機能も実装されると考えられる。

¹ 株式会社野村総合研究所による調査～2010 年までの国内 IT 市場の規模とトレンドを展望～(2005.12.7) より抜粋

表 1.1: IT 市場の予測.

(単位:億円)

市場・分野		2005年度	2006年度	2010年度
ブロードバンド市場	FTTH (Fiber To The Home)	2,113	2,916	6,483
	ケーブルテレビインターネット	1,837	1,946	1,745
	ADSL (Asymmetric Digital Subscriber Line)	5,566	5,779	4,234
	公衆無線LAN	92	171	376
	IP-VPN, 広域イーサネット	5,364	5,985	7,640
	ISP (Internet Service Provider)	7,113	7,286	7,353
	IP電話 (注1)	9,076	11,356	16,236
放送市場	地上デジタル放送	2,299	4,493	18,507
	BSデジタル放送	1,459	2,013	4,918
	ケーブルテレビ	2,252	2,414	2,743
	移動体向け放送	0 (注2)	5	502
	ネット放送	90	190	940
	VOD (Video On Demand)	90	140	620
セキュリティ市場	ウイルス対策	599	676	1,008
	情報漏えい対策ツール	320	360	472
	バイオメトリクス (主体)認証機器	128	158	242
	セキュリティサービス	1,100	1,140	1,370

(注1) ここでは、一般消費向けIP電話の加入者数。単位は千加入

(注2) 2006年4月のワンセグサービス (13セグメントに分割された6MHz帯域の1セグメントを利用するサービス)開始以降に立ち上がる市場であるため

出展:株式会社野村総合研究所による調査～2010年までの国内IT市場の規模とトレンドを展望～ (2005.12.7)

図 1.1 に、現在及び今後求められるマルチメディアデータ処理の要求性能を示す。図中では、マルチメディアアプリケーションに必要とされる基本的な演算性能を、GOPS (Giga Operations Per Second) で示した。MPEG-4 や現行のテレビの解像度を処理する MPEG-2 の MP@ML (Mail Profile at Main Level) の動画画像処理では、1 GOPS 程度の高い性能が要求され、デジタルテレビの機能と解像度を処理する MPEG-2 の MP@HL (Mail Profile at High Level) の処理では、10 GOPS 以上の性能が要求されることになる。最新の研究では、国際固体素子回路会議 ISSCC2005 (International Solid-State Circuits Conference 2005) にて、富士通の研究グループが 32 bit マルチメディアプロセッサコアである FR550 を 4 つマルチコア化し、MPEG-2 の MP@HL を 51 GOPS で処理可能にしたという結果が報告されている [6]。MPEG や、Dolby-AC3 等のオーディオ伸長処理では、それ

表 1.2: 携帯電話, e ビジネスライフ及びプラットフォーム各市場の予測.

(単位:億円)

市場・分野			2005年度	2006年度	2010年度
携帯電話市場	モバイルキャリア (注1)	ケース1 ケース2	69,000 69,000	70,200 68,800	70,200 63,300
	モバイルプラットフォーム		387	645	2,732
	モバイルコンテンツ		2,930	3,210	3,580
	モバイルソリューション		4,721	6,460	17,011
eビジネス・ライフ市場	BtoC EC (Business to Consumer Electric Commerce)		36,000	41,000	56,000
	CtoC (Consumer To Consumer)ネットオークション		11,400	16,200	28,000
	オンラインゲーム		850	1,130	2,370
	音楽配信 (*)		106	170	570
	ネット広告 (*)		2,650	3,590	6,430
	eラーニング		760	850	1,290
プラットフォーム市場	電子認証		253	292	531
	課金・決済		1,510	1,640	1,960
	ICタグ		275	370	1,207
	ICカード		367	464	675
	企業通貨		596	628	700

(注1) 年度ベースではなく年ベース. ケース1はARPU (Average Revenue Per User: 平均利用料) の低下速度が遅くなり前年比-3%で推移する場合. ケース2は料金競争により-5%で推移する場合

(注2) *の数値は、年度ベースではなく年ベース

出展:株式会社野村総合研究所による調査～2010年までの国内IT市場の規模とトレンドを展望～ (2005.12.7)

ほど高い性能は要求されないが、音声自動翻訳では 100 GOPS 程度と非常に高い性能が要求される。主にゲーム機等のアミューズメントアプリケーションで必要となる、3次元グラフィックス処理では 100 Mpps (Mega polygons per second) のスループットを達成するのに、100 GOPS 以上の処理能力を必要とする。現在家庭用ゲーム機の1つとして広く普及している、プレイステーション2に使用されている LSI, Emotion Engine では 66 Mpps の性能で 3次元グラフィックスを処理している [7]。通信認識においては、顔認識や動画像認識で 5~100 GOPS と幅広い性能が求められる。カーナビゲーション等の用途に今年度開発された、ルネサステクノロジの最新マルチメディアプロセッシングのコアである SH7785 は、600 MHz で、1 GOPS というスループットでマルチメディアデータを処理することが可能である [8]。以上より、マルチメディアデータ処理 LSI には、大量のデータを

高速に処理する性能が、今後益々求められることが分かる。

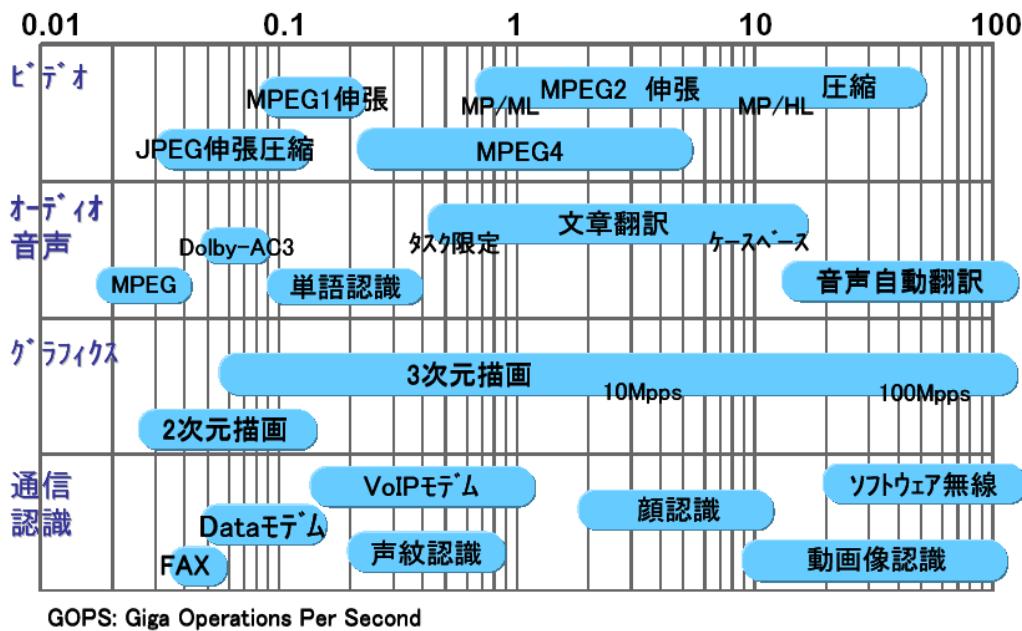


図 1.1: マルチメディアデータ処理に要求される性能。

次に、マルチメディアデータを効率よく処理するためのアルゴリズムについて述べる。一般に、マルチメディアアプリケーションに適用されるアルゴリズムは、音声や動画等のアナログの情報を、処理速度に加え高精度かつ高圧縮に処理することが求められる。マルチメディアデータの中でも、ビデオデータは特に膨大でありデータ圧縮を施さなければ通信速度や処理速度の向上率を必要以上に高めなければならなくなる。図 1.2 に、ビデオ圧縮伸長アルゴリズムの例を示す。横軸がビットレート、縦軸が解像度であり、各アルゴリズムの処理範囲を表している。主なビデオ圧縮伸長技術としては、ITU-T (International Telecommunication Union) の勧告である H.261 や H.263, ISO (International Organization for Standardization) の定めた国際標準である MPEG-1, MPEG-2, 及び MPEG-4 がある。H.261 と H.263 は電話回線を用いた TV 会議用に開発された。特に H.263 はエラー耐性や圧縮率向上を図ったオプションが多数用意されており、移動体通信にも適用可能となっている。MPEG-1 や MPEG-2 は、ビデオ CD や DVD 等の画像観賞用に用いられている。MPEG-4 に関しては、1999 年にバージョン 1 が国際標準となった比較的新しい規格であり、広範囲のアルゴリズムをカバーできる仕様となっている。また図中には示していないが、近年では圧縮率を向上させた H.264 が 2003 年 5 月に勧告として承認されている。以上のように、大量のデータを取り扱うマルチメディアデータ処理には、高圧縮を実現するアルゴリズムが不可欠であり、これらのア

ルゴリズムを効率的に処理することがマルチメディアデータ処理 LSI に求められるのである。

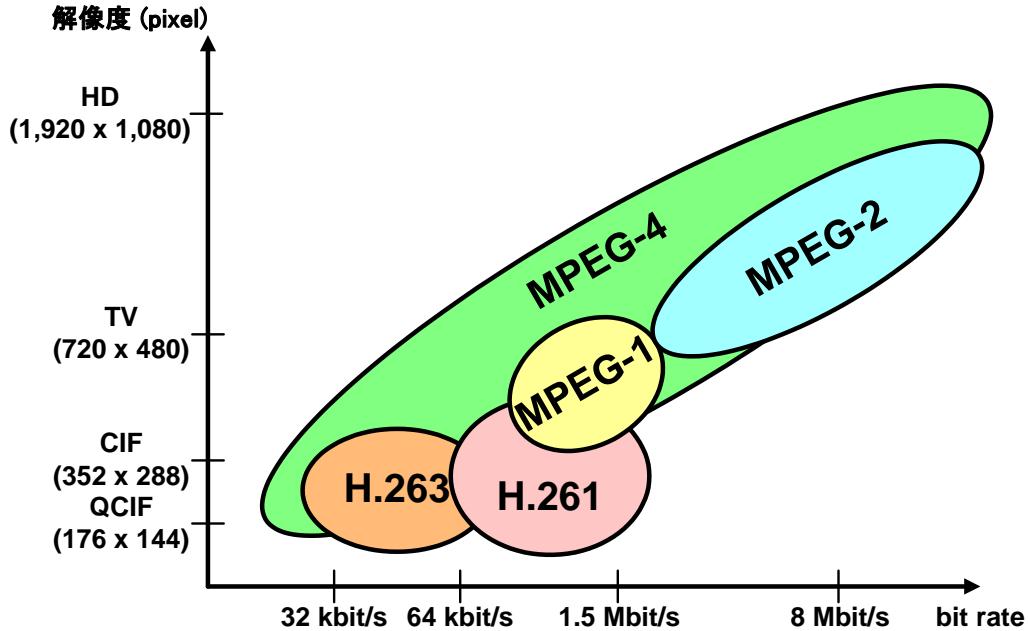


図 1.2: ビデオ圧縮伸長技術。

1.2 本研究の目的

本研究では、??節、及び??節の検討を通して、プログラマブルであるという汎用的性質を持ちながら、処理能力は専用ハードウェアに近いという設計思想の下に、リアルタイムにマルチメディアデータを高速処理、高圧縮でき、小面積かつ低消費電力であるマルチメディアデータ処理 LSI アーキテクチャの開発を研究目的とする。

はじめに、マルチメディアデータ処理のボトルネックとなっているテーブルルックアップ処理に主眼を置く。テーブルルックアップ処理の高速化には、機能メモリの一種であり、主として一致検索処理を高速に実現することができる CAM を利用する。CAM は 1956 年に発表されたクライオトロン・カタログメモリ (Cryotron catalog memory) [9] を起源とする、高速に一致検索処理を行うことのできる機能メモリの一種であり、これまで様々な研究や実用化が行われてきた。本論文では、CAM のアーキテクチャをベースとして、マルチメディアデータ処理 LSI アーキテクチャの開発を行う。また、モバイル機器をターゲットとした 200 MHz 前後の低い動作周波数で高速化を実現する手法として、通常の CAM に複数の入出力ポートを付加した、新しいアーキテクチャである、マルチポート CAM を適用する。こ

の適用によって、これまで実現が困難であった、テーブルルックアップ処理の並列化を検討する。更に、上記のアーキテクチャにデータ圧縮を実現する機構を加え、リアルタイム処理に加え高压縮を実現するアーキテクチャを実現する。

次に、従来のプロセッシングアーキテクチャと比較して、大幅に並列度を向上させることで、繰り返し演算処理を高速に実現できる、SRAM ベースの超並列 SIMD 型プロセッシングアーキテクチャの開発を行う。そして、上記で述べた CAM ベースのテーブルルックアップアーキテクチャと繰り返し演算処理 SIMD 型アーキテクチャを融合することで、マルチメディアデータをリアルタイム、プログラマブル、かつ高压縮に処理することができる、高性能 CAM ベースマルチメディアデータ処理 LSI アーキテクチャの開発を行う。

第2章 CAMベースブルルック アップ符号化処理

本章では、マルチメディアデータを高压縮に処理することのできる、CAMベースのテーブルルックアップアーキテクチャを提案する。提案アーキテクチャは、テーブルルックアップ処理を高速に行いつつ、リアルタイムに圧縮に用いる符号化テーブルを最適化して圧縮効率の向上を実現するものである。また、提案アーキテクチャは本論文の研究目的である、高性能CAMベース超並列 SIMD型プロセッサにおける高压縮処理の基本となる技術となっている。

本章の構成は以下の通りである。2.1節において、提案アーキテクチャの実現目標を示し、2.2節にて、提案アーキテクチャのターゲットアルゴリズムであるハフマン符号化について述べる。2.3節では、提案アーキテクチャの構成について詳述し、??節にて、FPGAへの実装結果について説明する。??節では、ハフマン符号化が用いられているJPEGアプリケーションに提案アルゴリズムを適用した場合の性能評価について言及し、提案アーキテクチャをLSI化した際のハードウェアコストについて、??節にて述べる。最後に??節にて、本章のまとめとする。

2.1 はじめに

近年のマルチディア環境の急激な発展に伴い、データのリアルタイムな高压縮処理の重要性については??節及び??節にて述べた。圧縮処理には大きく分けて、可逆圧縮 (Lossless compression) アルゴリズムと非可逆圧縮 (Lossy compression) アルゴリズムがある。可逆圧縮とはデータの欠落が全く起こらない圧縮方式のことであり、復号によって圧縮前のデータを完全に再現可能とするアルゴリズムである。この圧縮方法は圧縮率は大きくないものの、プログラムや文字データ等、データの欠損が重大な損失を起こすコンテンツに使用される。非可逆圧縮は、この逆でありデータの圧縮率が大きいものの、完全に前の状態には復元できない。そのため、音声や画像のような多少のデータの欠落に対しても品質を大きく損なう恐れの無いコンテンツに使用されることが多い。

本章では、代表的な可逆圧縮アルゴリズムであるハフマン符号化を圧縮アルゴリズムに利用し、可逆圧縮と非可逆圧縮それぞれの長所を実現可能なCAMをベースとしたリアルタイムに高压縮処理が可能なアーキテクチャを提案する。

2.2 ハフマン符号化

ハフマン符号化は、データ圧縮技術の中でも、最も効率のよいロスレス圧縮技法の一つとして知られている [10]. このアルゴリズムは、圧縮対象のデータ（記号）に対して可変長の符号を割り当てるものであり、各データのうち生起確率の大きい、すなわち頻繁に出現する記号には短いビット長の符号を割り当て、生起確率の小さい、稀にしか出現しない記号には長いビット長の符号を割り当てる符号化方式である。一般にこの割り当てはハフマン木と呼ばれる、各記号の生起確率に基づいた木を構築することで行われる。各葉に対応する記号の生起確率（及びそれらの和）を重みとして持たせ、重みの大きい順に記号を組み合わせることでハフマン木を構築し、根から葉にたどることにより符号化を行う。図 2.1 にハフマン木の構築例を示す。ハフマン木の構築法及び符号の割り当ては以下の手順で行う。

1. 各記号の生起確率を求める。
2. 記号を生起確率の大きさ順に並べる。
3. 生起確率の小さいものから 2つ選択し和を取る。この際枝の上から 0, 1 を付加する。
4. 記号ごとに根から葉までたどることで対応する記号の符号化が完了する。

この手順で処理を行うと、図 2.1 中の # で表している順番で木を構築することとなり、例えば記号 E の場合、「1110」のハフマン符号を割り当てることができる。図 2.1 の場合には各記号が ASCII コードで表されているとすると、「ABCDEFG」のデータは 56 bit 必要であるところを、24 bit で済むこととなる。この結果、全体のデータ量を削減することができる。以下にハフマン符号化の特徴をまとめる。

- 可変長符号
圧縮する記号それぞれに長さの異なる符号を割り当てる。
- データの一意性
デコードデータの解釈が一通りである。
- 可逆圧縮（ロスレス圧縮）
符号化されたデータから圧縮前のデータを損失なく、完全に復元することができる。

ハフマン符号化は、圧縮処理を行う前に現れる記号列を基にして、はじめに符号化テーブルを作成し、再び同様の記号列に対し作成した符号化テーブルを用いて圧縮を行う。しかしながらこの方法では、圧縮を行うためにデータを 2 回スキャンする必要があるため、オンラインでのリアルタイム圧縮は困難となる。そこで、

現在のハードウェアによるハフマン符号化の実装形態は、大きく分けて静的ハフマン符号化 (Static Huffman Coding) [10] と動的 (適応型とも呼ぶ) ハフマン符号化 (Adaptive Huffman Coding) [11] の 2 通りに分かれている。以下では、それぞれの現状とその問題点について述べる。

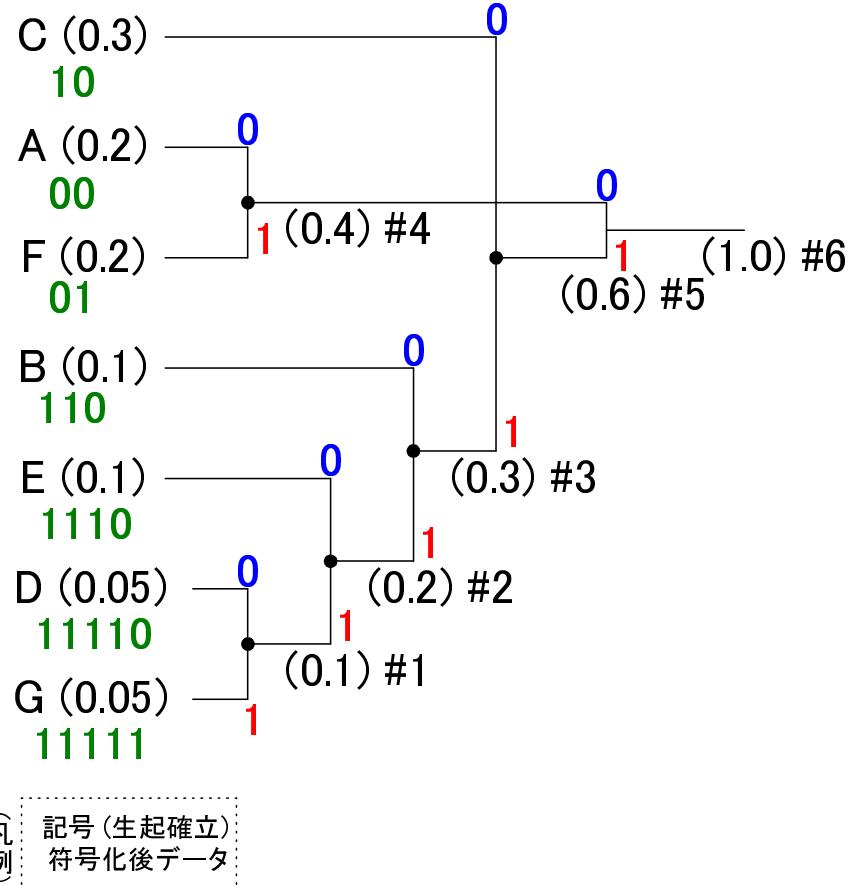


図 2.1: ハフマン木構築例。

2.2.1 静的ハフマン符号化方式 (Static Huffman Coding)

静的ハフマン符号化は最も広く実装されている形態である。符号化をリアルタイムに実行するために、あらかじめ数多くのサンプルデータから、データの出現頻度を調査しておき、この結果を元に作成した既存のテーブル [12], [13] 等を用意して符号化を行う。この既存のテーブル (静的テーブルと呼ぶ) を用いた方法には、汎用プロセッサベース、SRAM ベース [14], PLA (Programmable Logic Array) ベース [15], 並びに CAM [16] ベースのアーキテクチャが提案されている。汎用プロセッサベースのアーキテクチャはソフトウェアで符号化が実現されるため、ハード

ウェアと比較して処理効率は低い。また SRAM ベースアーキテクチャでは符号化する際、符号化テーブルから変換データを見つけるためのアドレスを計算することに数クロックサイクル必要となる。PLA ベースアーキテクチャは、符号化テーブルの規模が大きくなるにつれてハードウェア量が増大する傾向にある。また、CAM ベースアーキテクチャは符号化テーブルを用いて単純に一致検索機能を行うため、高速に処理が可能である。これらの実装方法はいずれも、標準的なデータに基づく既存の符号化テーブルを利用するため、画像の傾向が急激に変化するような動画等のコンテンツの場合、圧縮効率が減少する恐れがあり、更に既存の符号化テーブルが存在しないようなアプリケーションには適用できない等の問題がある。

2.2.2 動的ハフマン符号化方式 (Adaptive Huffman Coding)

従来のハフマン符号化の問題点である、オンラインでの圧縮、静的ハフマン符号化の問題点である既存のテーブルが存在しない、任意の記号列にあわせた圧縮、を改善するために考えられたのが動的(適応型)ハフマン符号化 [11] である。その特徴は記号を処理するごとにハフマン木を更新し、その時点で構成されたツリーによって符号化を行うものである。すなわち、符号化前の記号列全体をスキャンして符号化を行わないため、オンラインでの処理が可能となり、符号化テーブルも作成する必要がない。しかしながら、圧縮効率及びリアルタイム性についていくつかの問題も存在している。すなわち、アルゴリズムの特性上、符号化データに加えて初出現の記号は符号化せずに送らなければならないため、その分データ量が増加することになる。また、圧縮開始後はハフマン木が十分なものでないため、圧縮率が低いことが多い。これは、圧縮中にデータの出現分布が変化した場合にも同様のことが言える。更にデータの局所性により、集中的に出現した記号に短い符号が割り当てられたとしても、それ以降出現しない場合にはデータ量が大きくなるにつれて静的ハフマン符号化よりも圧縮効率が下がる場合がある。これに対しては符号化テーブルの定期的なリフレッシュという技法が取られることがあるが、過去のデータの生起確率を破棄することは逆に圧縮効率を悪化することにもなる。また、符号化の際にその都度ハフマン木を更新するため、処理により多くの時間を必要とする。これによりオンラインでの処理が可能だとしてもリアルタイムアプリケーションへの適用は向かない場合がある。このハフマン木の更新処理時間を改善するためにアルゴリズムを改良する研究 [17] や木の更新に CAM を用いた高速化に関する研究 [18] が行われているが、本質的にハフマン木の更新に関する検索処理やスワップ処理を含んでいるため適応型ハフマン符号化のハードウェアでの実装は難しく [16]、その圧縮効果と比較して利点が少ない。

2.3 リアルタイム符号化テーブル最適化アーキテクチャ

この章では、2.2.1節、及び2.2.2節で述べた静的ハフマン符号化、及び動的ハフマン符号化の問題点を改善し、マルチメディアデータを効率よく圧縮できる新しいアーキテクチャを提案する。提案アーキテクチャは、静的ハフマン符号化に基づくものであり、CAM を用いたテーブルルックアップ処理により、高速に符号化を行う仕組みになっている。従来の CAM ベースアーキテクチャは既存の符号化テーブルにのみ基づいて符号化を行っていたため、符号化は高速であっても圧縮効率に関しては、全く考慮されていなかった。これに対し、提案アーキテクチャでは符号化と同時に、出現する記号列にあわせて出現頻度を算出し、最適な符号化テーブルをアップデート／交換する。これによって既存の符号化テーブルが存在しないアプリケーションに対しても、柔軟に高速なハフマン符号化を提供することが可能となる。

図 2.2 に、提案アーキテクチャの構成を示す。提案アーキテクチャは、主としてエンコーダとオプティマイザという 2 つのブロックから成り立っている。エンコーダは CAM 及び符号化テーブルを内蔵しており、パイプライン処理によって高速に符号化を行うことができる。オプティマイザは、リアルタイムに高い圧縮効率を維持するために用いられる。これら 2 つのブロックが並列に動作することによって、提案アーキテクチャは、最適なテーブルのアップデート及び交換と、それに基づく高速かつ、高压縮な符号化を実現することが可能となる。

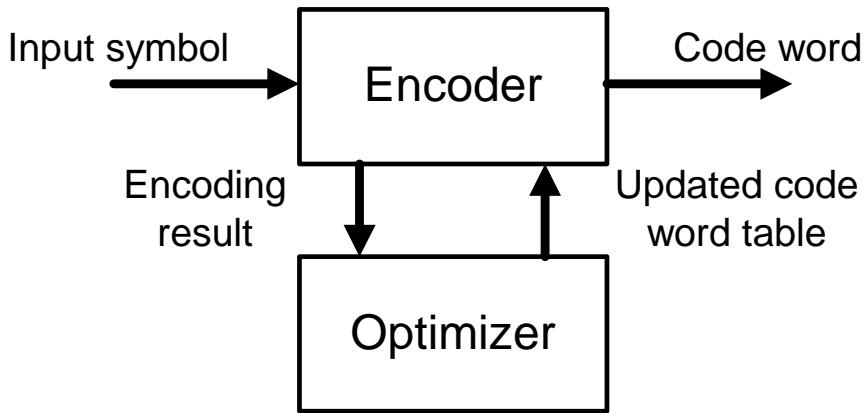


図 2.2: リアルタイム符号化テーブル最適化アーキテクチャ構成。

2.3.1 最適化ハフマンテーブルによる符号化

提案アーキテクチャの、ブロック図を図 2.3 に示す。エンコーダブロックは、大きく分けて以下に示す 4 つのモジュール及び複数個のレジスタから構成される。

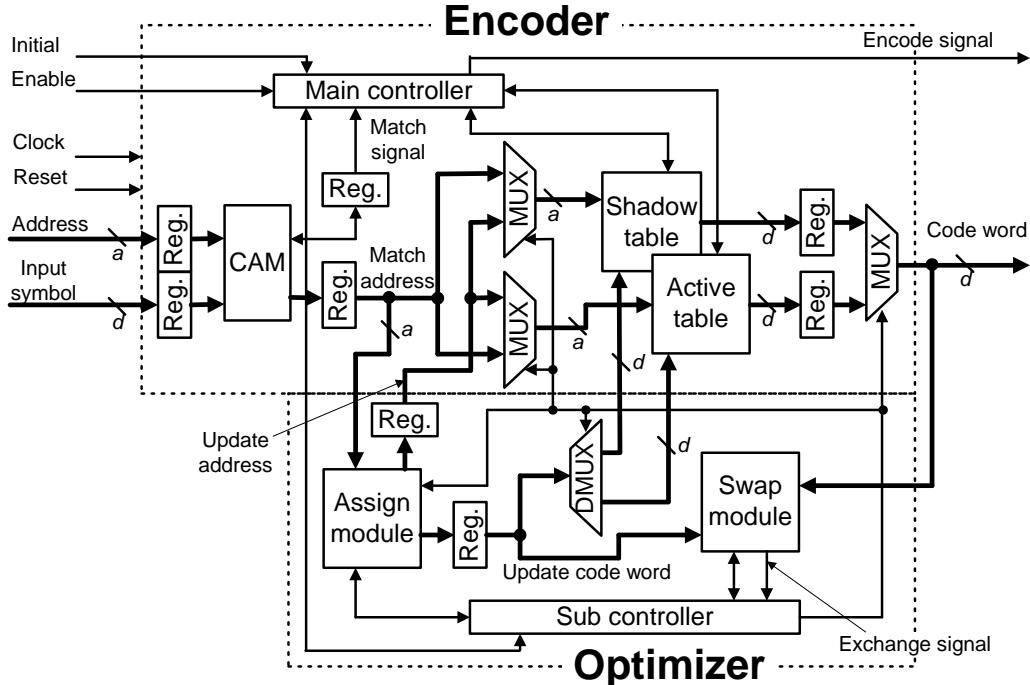


図 2.3: リアルタイム符号化テーブル最適化アキテクチャブロック図.

- CAM (Content Addressable Memory) × 1
- SRAM (Static Random Access Memory) × 2
- マルチプレクサ (MUX: MULTipleXer) × 3
- メインコントローラ (Main controller) × 1

内部の CAM 及び SRAM は、ワード数 2^a (a はアドレス長)，ワード長 d -bit である。また、2つの SRAM は、それぞれアクティブ符号化テーブル（以下、アクティブテーブルと呼ぶ）と、シャドウ符号化テーブル（以下、シャドウテーブルと呼ぶ）を構成する。マルチプレクサは各種信号の切り替えに使用し、メインコントローラがこれらのモジュールを制御する。

オプティマイザブロックは、大きく分けて以下に示す 4 種類のモジュール及び複数個のレジスタから構成される。

- アサインモジュール (Assign module) × 1
- スwapモジュール (Swap module) × 1
- デマルチプレクサ (DMUX: DeMULTipleXer) × 1
- サブコントローラ (Sub controller) × 1

アサインモジュールと、スワップモジュールの働きについては、2.3.3節にて詳述する。デマルチプレクサは信号の振り分けを行い、サブコントローラはオプティマイザブロックの制御を担当する。

提案アーキテクチャは、エンコーダブロックとオプティマイザブロックが並列に動作することによってリアルタイムに符号化テーブルを最適化した圧縮処理を実現する。その動作原理を、図2.4に示す。図中に示すように、CAMには、入力される符号化前データの全パターンが格納されており、アクティブテーブル及びシャドウテーブルには、ハフマン符号化テーブルが格納されている。また、あらかじめCAMに格納されている全符号化前データと、SRAM内のテーブルに格納されているハフマン符号は、符号変換に対応するデータ同士をアドレスで関連付けてある。処理の流れは以下の通り。

1. 符号化前データがCAMに入力されると、CAM内部に格納しているパターンに対して、一致検索処理が1クロックサイクルで実行される。
2. CAMから一致した符号化前データのアドレスが出力され、アクティブテーブルの読み出しポートに入力される。
3. アクティブテーブルは、アドレスに基づき、入力された符号化前データに対応したハフマンコードを1クロックサイクルで出力する。
4. シャドウテーブルは、圧縮処理のバックグラウンドでデータの出現頻度にあわせてテーブルの内容が最適化されているため、圧縮効率が低下したならば即座に、アクティブテーブルとシャドウテーブルの役割を切り替える。これによって圧縮効率の向上を実現することが可能となる。

提案アーキテクチャは、この一連のプロセスを、CAMに備えている検索データを格納するレジスタと、SRAMに備えているアドレスレジスタを経由することで1クロックづつ符号化前データを移動させることができる。そのためパイプライン処理を実現でき、連續して入力される符号化前データは、1クロックサイクルで次々とハフマンコードに変換され出力される。更に、エンコーダブロックは、入力される符号化前データの出現頻度にあわせ、符号化テーブルを選択することができる。通常のハフマン符号化アーキテクチャは、既存のハフマン符号化テーブルを1つ備えているだけであり、符号化前データの出現頻度が変化してもテーブルを選択することはできない。提案アーキテクチャは、符号化処理と並行してシャドウテーブルのアップデートが行われており、データの圧縮率があらかじめ定めてあるしきい値 (Threshold value) を下回った場合に、即座に符号化テーブルを切り替えることによって、再び圧縮率を向上させることが可能となっている。従来のハフマン符号化アーキテクチャは、符号化前データの出現頻度の変化に備えて、複数のハフマン符号化テーブルを用意しているものもあるが[19]、提案アーキテクチャは、アクティブテーブルとシャドウテーブル2つのみで様々な出現データに対応でき、かつ高い圧縮率を実現することが可能となっている。

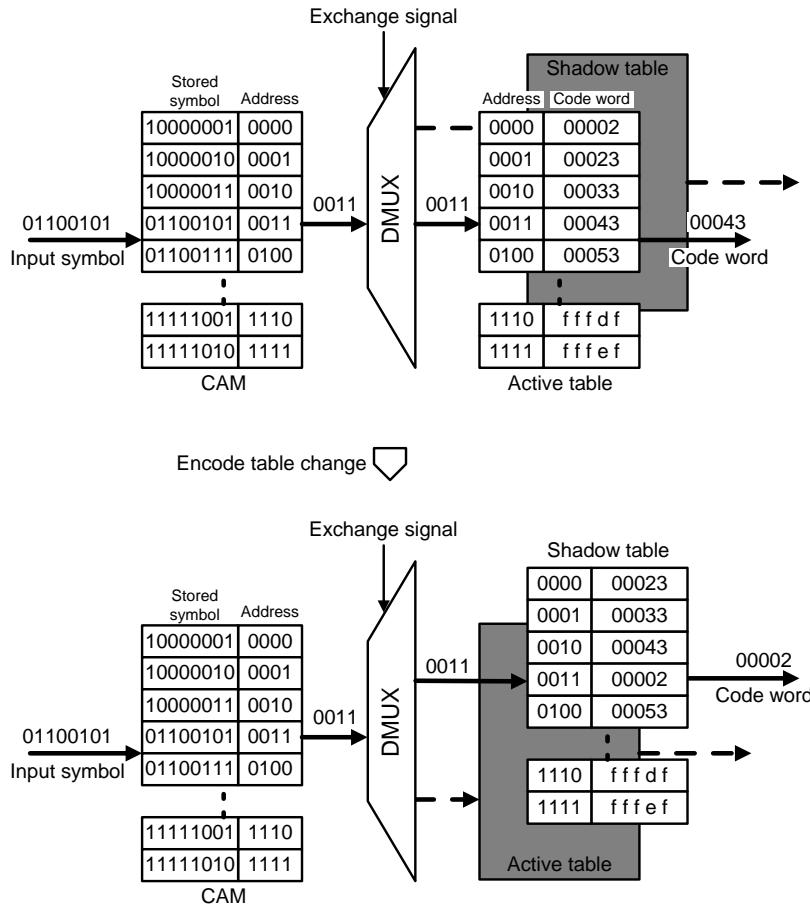


図 2.4: 最適化された符号化テーブルを用いたリアルタイム圧縮処理動作原理.

2.3.2 リアルタイム符号化テーブル最適化の原理

提案アーキテクチャは、符号化テーブルをアクティブ用とアップデート用に分け、符号化と並行してリアルタイムにマルチメディアデータの出現頻度を算出している。そして、算出結果に基づき符号化テーブルをアップデートし、データの圧縮状況にあわせてリアルタイムに符号化テーブルを切り替えることによって、データの高压縮の実現を可能としている。この最適化は、図 2.3 に示しているアシンモジュールとスワップモジュールによって実現される、ハフマンテーブルの最適化及びアップデートのフローチャートを図 2.5 に示す。図 2.5 の右側に示しているのが、オプティマイザブロックによるステップであり、左側に示しているのが、エンコーダブロックによるステップである。エンコーダブロックによるハフマン符号化に関しては、2.3.1 節で述べたとおりである。

オプティマイザブロックによるハフマンテーブルの最適化及びアップデートは符号化と並行して行われ、エンコーダブロックによるパイプライン処理を妨げることは無い。提案アーキテクチャに入力された符号化前データは、CAM で一致検

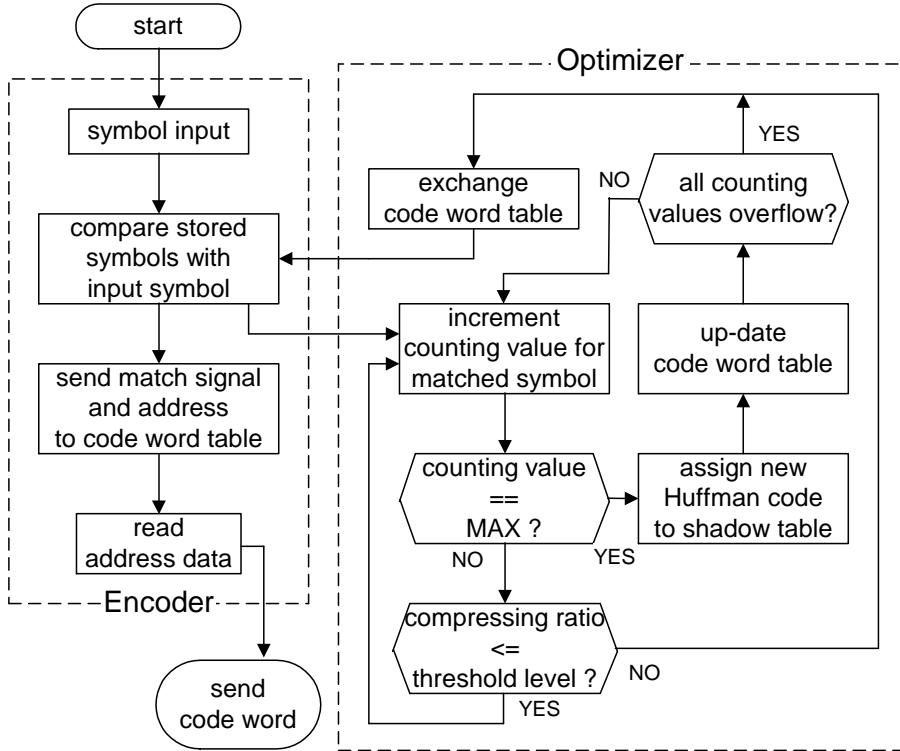


図 2.5: ハフマンテーブルの最適化及びアップデートのフローチャート.

索処理されアドレスに変換された後に、アクティブテーブルへ送信されるが、同時にアサインモジュールにも送信される。アサインモジュールには、CAM に格納されている全符号化前データパターンと同数個のカウンタが用意されており、CAM の一致検索結果に従って、該当するカウンタがインクリメントされる。

アサインモジュールは、1回の符号化毎にカウンタのインクリメントが終了した後、該当したカウンタの値がオーバフローしていないか確認する。ここでオーバフローを確認した場合には、その直前に入力された符号化前データの出現頻度が大きいことを意味するため、現在変換に利用されているハフマンコードより、短いビット長のアップデート用ハフマンコードをシャドウテーブルへ送信する。このアップデート用ハフマンコードは CAM から出力された一致アドレスが示すデータへ上書きされることとなる。続いてアサインモジュールでは、内部に格納しているカウンタが全てオーバフローしているか否かを確認し、全てのカウンタのオーバフローが確認されたならば、サブコントローラに確認信号を送信する。その後、サブコントローラからテーブル切り替え信号 (Exchange signal) がスワップモジュールに送信され、アクティブテーブルとシャドウテーブルの役割を切り替える信号がスワップモジュールからサブコントローラを介して送信されることとなる。

CAM による一致検索処理に該当したカウンタが、オーバフローしていなかった場合は、ハフマンコードのアップデートプロセスは省略されるものの、スワップ

モジュール内であらかじめ設定してあるしきい値と、現在の圧縮率を比較し、しきい値を上回った場合には、アクティブテーブルとシャドウテーブルの役割を切り替える信号がスワップモジュールから送信される。

以上のプロセスによって、出現頻度の高い符号化前データには、よりビット長の短いハフマンコードが割り当てられ、データの圧縮率が低下した場合には、新しく構築したハフマンテーブルが適用される。従って、提案アーキテクチャは、常に画像等の出現データの特徴にあわせて、リアルタイムに高い圧縮率を維持することが可能となる。

2.3.3 アーキテクチャ

この節では、提案アーキテクチャの符号化テーブルアップデート、及び切り替えの役割を担う、オプティマイザブロックのアーキテクチャを中心に述べる。

アサインモジュール

入力される符号化前データの出現頻度に基づいたシャドウテーブルの最適化は、主としてアサインモジュールによって処理される。図 2.6 に、アサインモジュール内部のブロック図を示す。アサインモジュールは、大きく分けて以下に示す 9 種類のユニットが動作して符号化テーブルのアップデート処理を実行する。

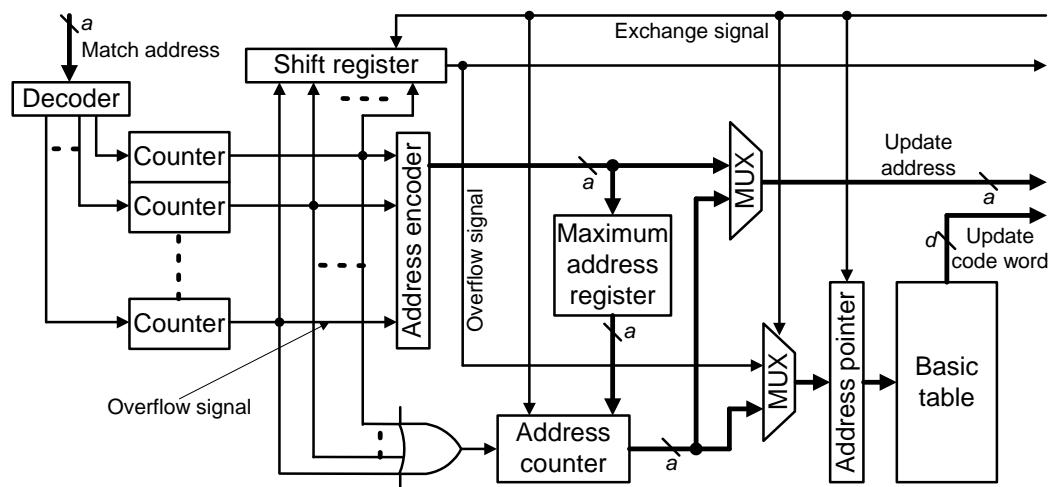


図 2.6: アサインモジュールブロック図。

- カウンタ (Counter) $\times 2^a$
- デコーダ (Decoder) $\times 1$

- アドレスエンコーダ (Address encoder) × 1
- アドレスカウンタ (Address counter) × 1
- アдресスポインタ (Address pointer) × 1
- ベーシックハフマン符号化テーブル (Basic table) × 1
- シフトレジスタ (Shift register) × 1
- 最大アドレス値記憶用レジスタ (Maximum address register) × 1
- マルチプレクサ (MUX: MULTipleXer) × 2

エンコーダブロックに内蔵されている CAM に、符号化前データが入力された後、一致アドレスはアサインモジュールに入力される。一致アドレスはデコーダを経由するため、 2^a 個あるカウンタのうち 1 つが選択され、その値がインクリメントされる。この際、提案アーキテクチャを適用するアプリケーションによって、カウンタの最大値をあらかじめ決定しておく。カウンタがオーバフローを起こした場合、オーバフローシグナル (Overflow signal) は、アドレスエンコーダ (Address encoder) を経由してアドレス値に変換され、シャドウテーブルを更新するためのアップデートアドレス (Update address) として出力される。それと同時に、オーバフローシグナルを OR 演算した結果が、アドレスカウンタに入力されベーシックハフマン符号化テーブル (以下ベーシックテーブルと呼ぶ) 用の、アドレスが生成される。シャドウテーブルを最適化するための、アップデートハフマンコード (Update code word) は、ビット長の短いものからベーシックテーブルに降順で格納されているため、カウンタがオーバフローした順に、先に生成されたアップデートアドレスに基づき、シャドウテーブルへアップデートハフマンコードが送られることとなる。これらのカウンタ及びアドレスカウンタの値は、スワップモジュールによってアクティブテーブルとシャドウテーブルが切り替えられた後、リセットされる。提案アーキテクチャは、この切り替え処理を 2 つの条件、圧縮率があらかじめ設定されたしきい値を上回った場合、もしくは全カウンタがオーバフローした場合に行う。

次にアサインモジュールに内蔵されている各ユニットの働きと共に、シャドウテーブルのアップデート手順を、図 2.7 にて述べる。処理開始時には、アクティブテーブル、シャドウテーブル、及びベーシックテーブルに格納されているデータは同一である。カウンタのオーバフロー時に、アサインモジュールから出力されるアップデートハフマンコードは、シャドウテーブルの別のアドレスに格納されている。そのため、アップデートハフマンコードを書き込む前に、そのアドレスに書き込まれているハフマンコードは、シャドウテーブル内の別のアドレスへ移さなければならない。しかしながら、この入れ替え処理には少なくとも数クロックサイクル程度必要とするため、エンコーダブロックのパイプライン処理の妨げと

なる。この問題を解決するために、提案アーキテクチャはシャドウテーブルのアップデートプロセスを、オーバーライティングフェーズ (Over-writing phase)，及びオーダーライティングフェーズ (Order-writing phase) に分けて行うこととした。

オーバーライティングフェーズでは、シャドウテーブルのデータの整合性を気にすることなく、オーバフローしたカウンタに基づいて、次々にアップデートハフマンコードを上書きしていく。

図 2.7 の例に示すように、アップデートハフマンコード，“00002”及び“00023”は、同一のデータがテーブル内に存在するのに関わらず上書きされているのが分かる。アップデートハフマンコード，“00002”は、2 bit のデータ量であり、上書き前のデータである“001e5”と比較して 3 bit のデータ量削減となる。この例の場合、3 回のアップデートが行われたため、合計 20 (=3+13+4) bit のデータ量削減となる。また、カウンタから出力されるオーバフローシグナルは、信号線が接続されているシフトレジスタの各ビットへ書き込まれ、最大アドレス記憶用レジスタは、アドレス “0010” から “1110” に更新される。以上の処理をアサインモジュールは、カウンタ全てがオーバフローするか、圧縮率がしきい値を上回るまで実行する。

オーダーライティングフェーズは、オーバーライティングフェーズで生じた、シャドウテーブル内のデータの不整合を修正する処理である。オーバーライティングフェーズ終了時には、異なる符号化前データでも、同一のハフマンコードが出力される状態となっている。そのため、降順で読み出されたベーシックテーブルのハフマンコードの残りを、最大アドレス記憶用レジスタが示すアドレスまでシャドウテーブルに書き込まなければならない。オーバーライティングフェーズが終了したならば、図 2.6 に示す、アドレスポインタは最後のアップデートアドレスの位置で停止する。その後アドレスカウンタは、再び 0 からカウントを開始し、シャドウテーブルにアドレスを送信する。この時、シフトレジスタはこれまで保存していたオーバフローの履歴を逐次的に出力する。この信号は、シャドウテーブルへはデータ書込みのイネーブル信号として、ベーシックテーブルへはアドレスポインタをオフセットとして残りのアップデートハフマンコードを読み出すアドレスとして利用される。

図 2.7 の例では、シャドウテーブルには、アドレス “0000” に、“001e5”的続きのハフマンコードである“00fe8”が書き込まれる。続いてアドレス “0001” には、ハフマンコード “00fa9” が書き込まれる。この処理を最大アドレス記憶用レジスタが保存しているアドレス “1110” まで繰り返す。その結果、ハフマンコードが重複すること無く、シャドウテーブルは、入力される符号化前データの出現頻度にあわせてアップデートされる。

以上述べた、オーバーライティングフェーズとオーダーライティングフェーズは、符号化のパイプライン処理を停止することなく実行されるため、提案アーキテクチャは高速にハフマン符号化を実行しながら、ハフマンテーブルの切り替えを実現できる。

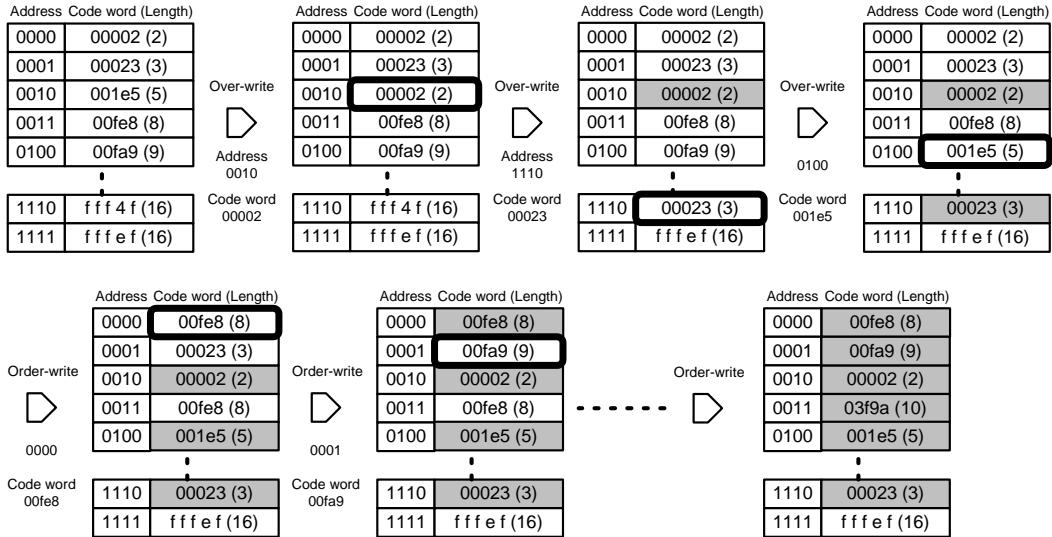


図 2.7: シャドウテーブルのアップデート処理手順.

スワップモジュール

スワップモジュールは、符号化前データの圧縮状況を監視し、アクティブテーブルとシャドウテーブルの切り替えを実行するモジュールである。図 2.8 に、ブロック図を示す。スワップモジュールは、大きく分けて以下に示す 4 種類のユニットから構成される。

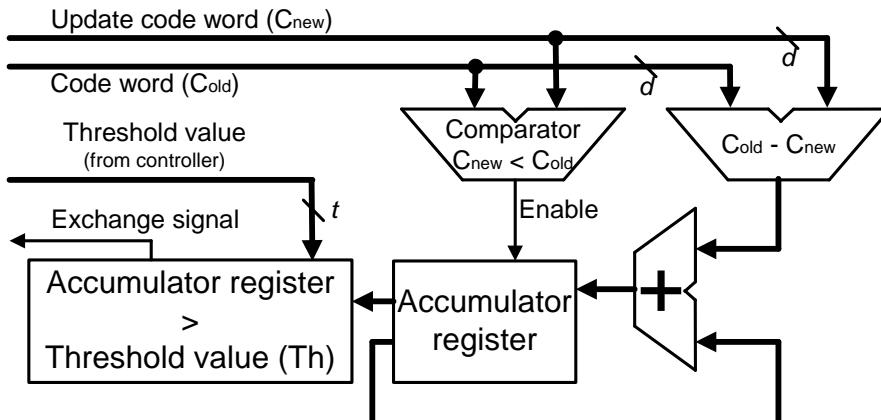


図 2.8: スワップモジュールブロック図.

- アキュムレータレジスタ (Accumulator register) $\times 2^a$
- 加算器 (Adder) $\times 1$

- 減算器 (Subtractor) $\times 1$
- 比較器 (Comparator) $\times 2$

エンコーダブロックにてハフマンコード (Code word: C_{old}) が output され, アシンモジュールがアップデートハフマンコード (Update code word: C_{new}) を生成した場合, スワップモジュールは, これらデータの減算処理 ($C_{new} - C_{old}$) を行う, また, 比較器では, データの大小比較 ($C_{new} < C_{old}$) を行い, アップデートハフマンコードが出力されたハフマンコードより小さい場合, 減算結果が加算器を介してアキュムレータレジスタへ保存される. このアキュムレータレジスタに保存される値は, これまでアップデートされたハフマンコードが, どの程度ビット長を削減しているかの指標として利用できる. よって, この指標があらかじめユーザによって指定されるしきい値を超えた場合, 現在のアクティブテーブルは入力される符号化前データに適応していないことがわかる. その結果, スワップモジュールは直ちにアクティブテーブルとシャドウテーブルの切り替え信号 (Exchange signal) を生成するため, 提案アーキテクチャはどのようなパターンの符号化前データが入力されても, 圧縮効率の向上を実現することが可能となる.

第3章 マルチポートCAMベース並列テーブルルックアップ符号化処理

本章では、マルチメディアデータを高速に処理するための、ボトルネックとなっていたテーブルルックアップ処理を、並列度の向上によって解決する手法を提案する。並列にテーブルルックアップ処理を実現するアーキテクチャとして、フレキシブルマルチポートCAM (FMCAM: Flexible Multi-Ported Content Addressable Memory) を提案する [20]–[26]。FMCAMは、従来効率よい処理が困難であった並列一致検索処理を実現可能にした、新しい機能メモリである。

本章の構成は、3.1節において、テーブルルックアップ処理を並列化することの意義について述べる。3.2節では、FMCAMのアーキテクチャについて詳述し、FPGAへの実装を通して性能評価を行う。次に、3.3節において、汎用向けアーキテクチャであったFMCAMをテーブルルックアップ処理向けに特化した改良について説明する。3.4節では、改良されたFMCAMのアーキテクチャを詳述し、3.5節において、FPGA及びASICへの実装を通して性能評価を行う。3.6節では、テーブルルックアップ処理の1つであるハフマン符号化に着目し、FMCAMの処理能力を検証する。また、3.7節では、提案アーキテクチャと、2章で述べたリアルタイム符号化テーブル最適化アーキテクチャの融合について検討する。最後に、??節にて本章のまとめを述べる。

3.1 はじめに

マルチメディアデータを高速に処理するためにはテーブルルックアップ処理を高速に行うことが必須である。一般に、高速化の方法で代表的なものは並列化が考えられるが、テーブルルックアップ処理を並列化する場合、符号化のためのテーブルを複数用意する必要がある。しかし、この方法では面積及び消費電力が大きくなり、テーブルデータ間のコヒーレンシや同期等の問題も無視することはできない。そのため、これまでのマルチメディアデータ処理LSIは、主にパイプライン処理によって高速化を図っていた [14]–[16], [27]。しかしながら、図 ??で示したように、テーブルルックアップ処理は別モジュールで行われることが多く、デー

タ転送の際に内部バスの混雑を生じさせ、消費電力も増大する結果となる。加えて近年の汎用プロセッサやメディアプロセッサは処理能力向上のために、演算器を並列に複数備えるアーキテクチャやコアを複数備えるマルチコア化が進んでいる[6], [28]。これらのアーキテクチャの発展に、テーブルルックアップ処理を適合させるには、これまで実現が困難であった並列化の実現が求められる。そこで、本章では、テーブルルックアップ処理の高速処理のために、従来のCAMに比較ポートを複数実装させたマルチポートCAMを新たに開発し、マルチポートCAMを適用した並列テーブルルックアップ符号化アーキテクチャと、その有効性を示す。

3.2 フレキシブルマルチポートCAM

本章では、CAMの能力向上の一手法として、マルチポート化に着目し、新しいCAMアーキテクチャであるフレキシブルマルチポートCAMであるFMCAM(Flexible Multi-ported Content Addressable Memory)を提案する。性能評価のためにFMCAMをFPGAに実装することにより、既存のCAMとの比較を通して、処理能力の向上及びハードウェア量の削減効果を示す。

3.2.1 CAMのマルチポート化

マルチポート化への背景

CAMは機能メモリの一種であり、主として一致検索処理を実行する回路である。一般にCAMは、その一致検索能力が高速であることが知られているが、実際には処理速度と比較器の数の関係、及びコストの問題があり、これらが普及の妨げになっている。ここで、CAMの能力を向上させる方法として考えられるのは、最大動作周波数の向上、処理ビット幅の拡大、及びポート数の増加の3点であり、特にポート数の増加は、その他の方法と比較して有効であることが示されている[29]。この観点から既存のCAMをポート数で大別した場合、(1)入力ポート及び出力ポートを1対持ち比較対象データを格納する内部のメモリであるコンテンツテーブルを1つ持つシングルポートCAM、(2)複数の入力ポートと1つの出力ポートを持ちコンテンツテーブルを共有しているマルチインプットCAM[30]、及び(3)シングルポートのCAMを並列に配置した並列CAMの3つがあげられる。特に、大規模なアプリケーションほど並列CAMにより処理能力の向上を図っているものが多い[30]–[35]。しかし、並列CAMは、ハードウェア量及びコストの増大を避け難く、また、コンテンツテーブルを更新する際に、各CAM間でコンテンツの同期を取る必要も生じる上、CAMの間の遅延等も考慮せねばならなくなる等の問題がある。このような背景から本論文ではCAMの新しいアーキテクチャとして、複数の入力ポートを持ち、その各々に対応する出力ポートも複数持つマルチポートCAMを提案する。なお、その構成方式には、処理速度及びハードウェア

量のバランスを考慮して、BPBP (Bit-Parallel Block-Parallel) 方式 [36] を採用する。マルチポート CAM は、シングルポート CAM と比較して並列処理による処理能力の向上を望める。さらに並列 CAM を使用する場面においても、マルチポート CAM を適用することで処理能力の向上を図りながら、ハードウェアコストを削減することができると期待される。

マルチポート化の課題

CAM のマルチポート化にあたっては、次に示す 2 つの課題があり、本論文ではその各々について有効な解決策を示す。

1. マルチポート化による比較器数の増加

複数のポートがコンテンツテーブルを共有すると、比較器数がシングルポート CAM と比較して増大する問題がある。

この解決策として、比較対象データのカテゴリ分け [37], [38] を採用した。これは、通常 BPWP (Bit-Parallel Word-Parallel) 方式 [36] で用いられ、比較対象データをあらかじめ指定した一部のビットパターンによるカテゴリに基づいて、コンテンツテーブル内に分類・格納しておく手法である。一致検索の際には、入力された比較データのビットパターンより、比較データの属するカテゴリを決定し、そのカテゴリ内の比較対象データのみを一致検索処理することで比較器数を削減するものである。本論文ではこの手法を FMCAM に組み込み、BPBP 方式及びマルチポート化に適合するような改良を施すことで、ポート数の増加に伴う比較器数の増加を抑えることを目指した。

2. 比較データの即時処理

一般に、マルチポート化された CAM の場合、各ポートに比較データが到着するタイミングは任意となる。並列 CAM の場合には、各 CAM が個々に比較データを処理できるため、このタイミングのばらつきによるスループットの低下は生じない。一方、単体の CAM、特に BPBP 方式等の CAM を基にして、単純にマルチポート化した場合、処理の開始及び終了を各ポートで合わせることで、同時に複数の比較データを処理できる。しかし、入力データの到着タイミングにずれが生じて、比較開始に間に合わない比較データがあった場合には、次の比較開始までの待ち時間を必要とすることとなる。これによりスループットの低下が生じ、複数のポートを十分に活用できない。

本論文では、特にアドレスカウンタを改良することによって比較データの即時処理を可能とし、スループットの向上を目指した。

3.2.2 FMCAM の概要

図 3.1 にワード長 d -bit, ワード数 2^a 及び p 組の入出力ポートを持つ FMCAM の概念図を示す。各入力ポートには d -bit の比較データ (CD: Comparison Data), 及び d -bit のマスクデータ (MD: Mask Data), 及び h ($\leq d$)-bit のハミング距離 (HD: Hamming Distance) を入力できる。一方、各出力ポートは、1 bit の一致信号 (Match signal) 及び a -bit のアドレス値 (Match address) より成る。FMCAM は、 p 組の入出力ポートが 1 つのコンテンツテーブルを共有しており、最大 p 個の比較データの一一致検索処理を同時に行うことができるほか、どのポートも比較データの到着にあわせて一致検索処理を即時に開始可能である。そのため FMCAM は、同じサイズのコンテンツテーブルを持つ通常のシングルポート CAM と比較して、最大 p 倍の処理能力を持つことになる。また、単体ながら並列 CAM と同様の処理が可能となり、ハードウェア量の消費を少なく抑えられ、かつスループットの高い処理が可能となる。

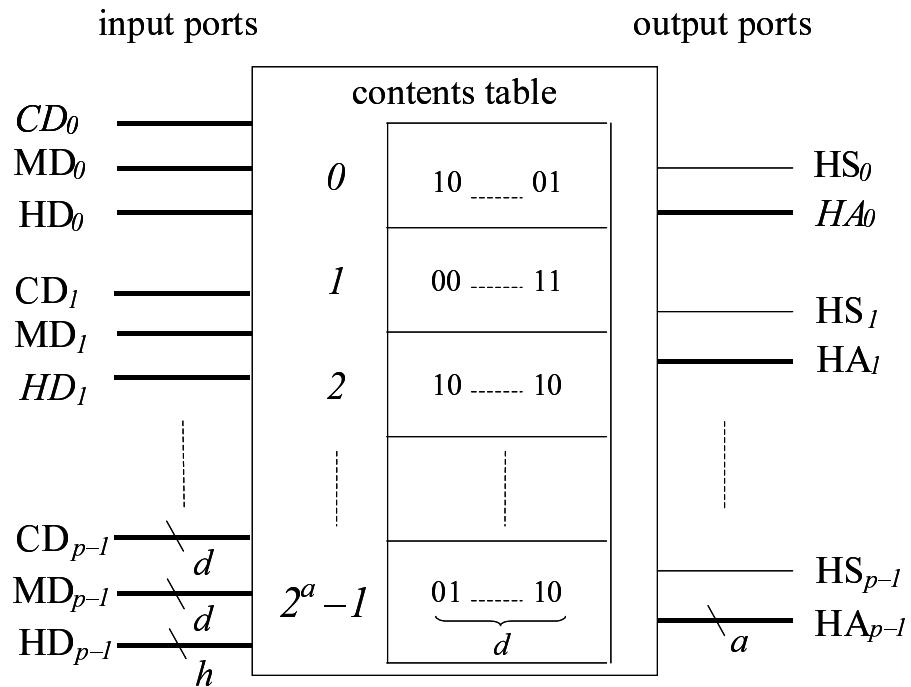


図 3.1: ワード長 d -bit, ワード数 2^a の FMCAM.

柔軟な検索機能

FMCAM はマルチポートであり、ASIC 及び FPGA に実装することで高い処理能力を得ることができ、マルチポートを生かした柔軟な検索機能を利用すること

が可能となる。そのため既存のCAMと比較して次に示すようなことが実現可能である。

1. 適用アプリケーションに合わせてポート数が変更可能

FMCAMはデータを格納しているモジュールと、データの一致検索を行うモジュールが互いに独立しているため、ポート数の増減が容易に可能である。従って、適用するアプリケーションに応じて最適なポート数を持つように構成できるという特徴を有する。

2. ポート毎に独立したマスクを指定可能

FMCAMは、ポート毎に独立してマスクデータを設定することができるため、幅広い検索が可能となる。

一例として、文献 [39], [40] 等で提案されている最小値検索がある。これは、FMCAMの各ポートに長さの異なるマスクデータを用意したうえで、同一の比較データを入力することで一度にポート数分処理できるため、処理時間を短縮することができるようになる。

3. あいまい検索

文献 [41]–[43] では、検索にハミング距離の概念を導入したCAMを提案している。すなわち“あいまいさ (approximate)”も含めて一致とするものである。これによって柔軟な検索を可能としているが、FMCAMは、ポート毎に独立してハミング距離を指定することを可能としている。そのため、より柔軟なあいまい検索が可能となる。例えば、4つのポートに、各々ハミング距離を0(通常の一一致検索), 1, 2及び3と指定した場合、同一の比較データ“00000”を各ポートに入力しても、一致するものはそれぞれ“00000”, “00100”, “11000”, “01011”などというように異なることとなる。

適用アプリケーション例

FMCAMは、高い処理能力と柔軟な検索機能で様々なアプリケーションへ適用できる利点を持つ。本論文では、FMCAMをマルチメディアアプリケーションのボトルネックとなっている、テーブルルックアップ処理へ適用し、その有効性を検証する。特にハフマン符号化は、テーブルルックアップ処理を利用する圧縮アルゴリズムとして知られており、近年のマルチメディアデータ処理の発展から、高効率の圧縮方法の1つとして注目を浴びてきている。これは文献 [10] にて提案された方式で、特に画像、音声及び動画等に向けた研究が盛んである。FMCAMは、従来並列化が困難であったこのアプリケーションについても、高速に処理できる可能性を持っており、ハードウェア量及びコストの削減につながるものと考えている。特にハードウェア量を小さくすることは、携帯端末等への応用につながるものと考えられる。

3.2.3 アーキテクチャ

FMCAMは、マルチポート化、比較対象データのカテゴリ分け処理、及び比較データの即時処理に特化した構成をとっている。

図3.2に、内部構成のブロック図を示す。FMCAMは大きく分けて4つのモジュール、すなわち“カテゴリモジュール(CaM)”, “コンテンツモジュール(CoM)”, “セレクタモジュール(SeM)”及び“ポートモジュール(PoM)”から構成されている。

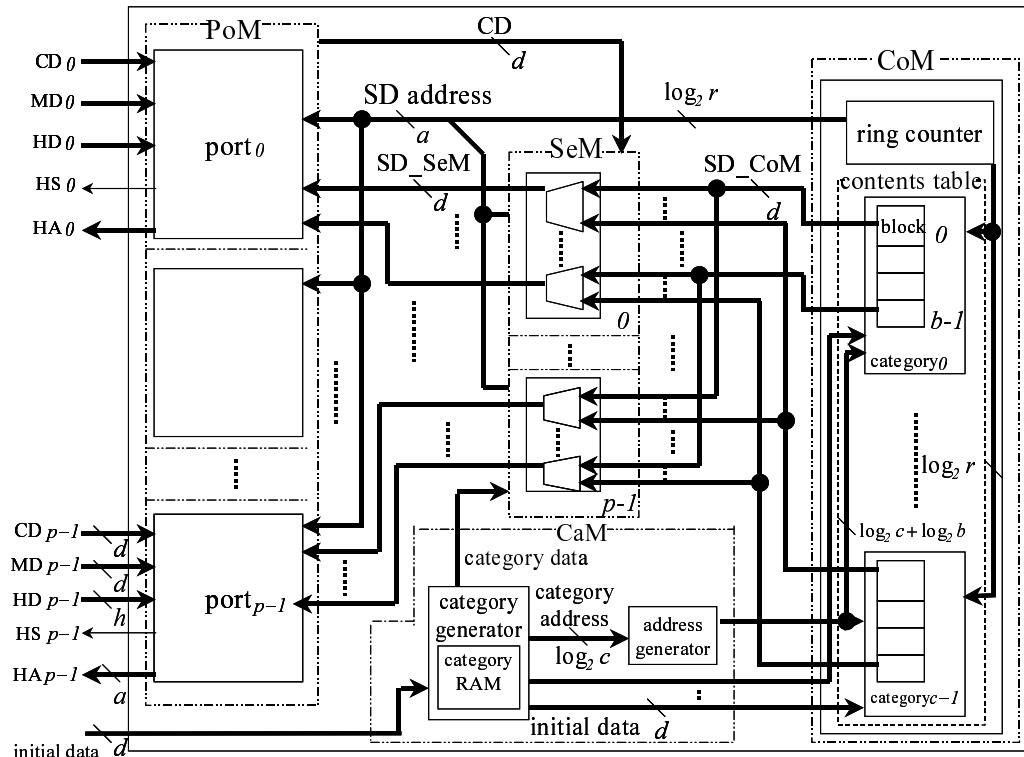


図3.2: FMCAMの内部構成。

1. カテゴリモジュール

内部にはカテゴリRAMを内蔵したカテゴリジェネレータとアドレスジェネレータが格納されている。カテゴリジェネレータ及びカテゴリRAMは、初期化の際に外部からデータを受け取って3.2.4節にて詳述するカテゴリ分けのための処理を行う。またアドレスジェネレータは、この外部からのデータをコンテンツテーブルに格納するためのアドレスを、カテゴリに基づいて生成し、コンテンツテーブルへと送出する。

2. コンテンツモジュール

このモジュールは、図3.2に示すカテゴリを c 個ひとまとめにしたコンテンツテーブル、及び3.2.4節にて詳述するリングカウンタを格納するモジュールである。

FMCAMはBPBP(Bit-Parallel Block-Parallel)方式を採用しており、図3.3に示すように1ブロックは d -bit長の比較対象データ(SD_CoM)を r ワード格納しているため、リングカウンタは0から $r-1$ までのアドレスを1クロックサイクル毎に順次出力することになる。よって各ブロックからは全データを r クロックサイクルで読み出すことができ、FMCAMは、全ての一致検索処理を r クロックサイクルで完了できる。

1カテゴリは、ブロック b 個分に相当しており、コンテンツテーブルにはカテゴリを c 個用意している。従って全ての比較対象ワード数は $2^a = cbr$ と表すことができる。初期化の際には、ライトイネーブル信号、外部からの初期データ及びそのアドレスが送られてきて、各カテゴリに格納される。

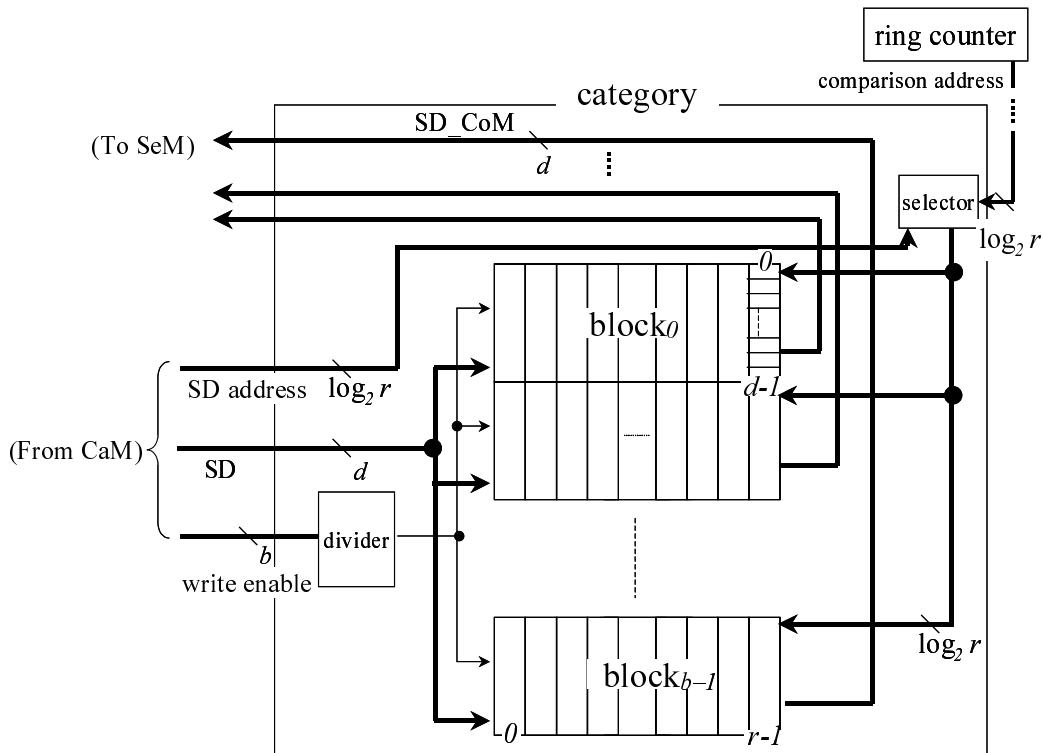


図3.3: コンテンツモジュール(CoM)内のカテゴリ。

3. セレクタモジュール

このモジュールは、各ポートモジュールに対応させてポート数分用意されている。図3.4に示すように、このモジュールの内部には、カテゴリセレクタ

と c 対 1 のマルチプレクサが b 個用意されている。カテゴリセレクタは、比較データ (CD) から対応するカテゴリを判別し、各マルチプレクサに選択信号を送信する。この信号を受けてマルチプレクサは、 c 個の比較対象データ (SD_CoM) のうち対応するカテゴリのデータを選択して、ポートモジュールへ対応する比較対象データ (SD_SeM) を送出することになる。

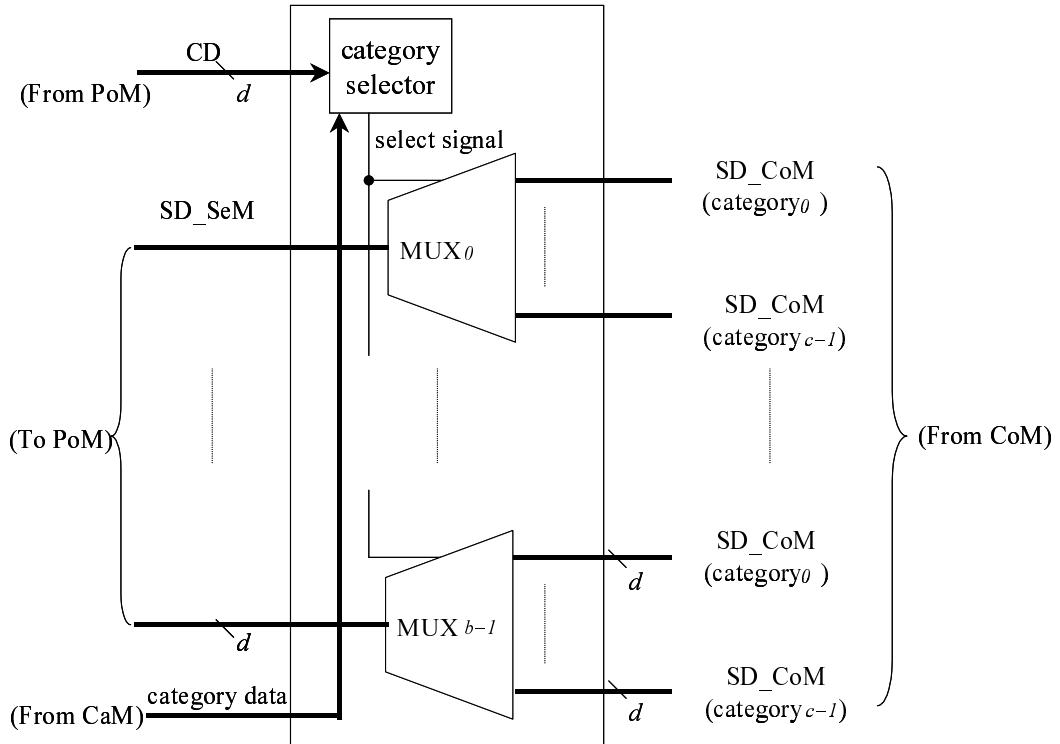


図 3.4: セレクタモジュール (SeM).

4. ポートモジュール

データの入出力を行うモジュールであり、FMCAM はこのポートモジュールをポート数分持つ。図 3.5 にポートモジュールのブロック図を示す。比較データ (CD) を受け取ったポートモジュールは、まずセレクタモジュールへこのデータを送信する。その後、セレクタモジュールから対応する比較対象データ (SD_SeM) が比較器に送られてくるので、ポートモジュールで受け取った比較データと比較処理される。なお、比較データと同時に入力されるマスクデータ (MD) の対応ビットが 1 になっているビットについては、比較結果を反映させないことでマスクしている。また、ハミング距離 (HD) が指定されている場合には、プライオリティエンコーダに比較結果を送信する前に、ファンクション回路であいまい検索等の処理を行う。なお、このプライオリティエンコーダは、複数のヒットがあった場合に優先順位の高い比較対象データ

を選択し、一致信号 (HS) とそのアドレス (HA) を出力するためのものである。また、コンテンツテーブル内の各カテゴリは b 個のブロックに分割されているため、比較器もそれに応じ b 個用意されている。ここで示したポートモジュールは互いに独立しており、再構成によってその数も増減可能であるため、適用アプリケーション毎に必要なポート数を柔軟に確保することが可能となる。

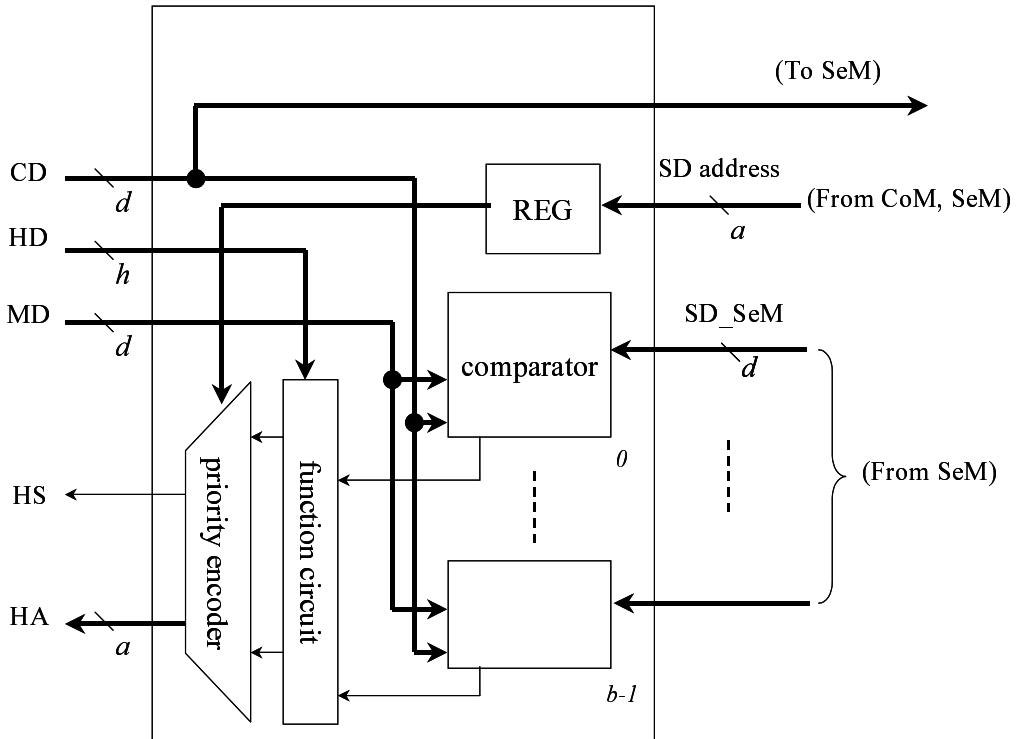


図 3.5: ポートモジュール (PoM).

3.2.4 マルチポートの実現方法

CAMにとって、処理速度の向上と大容量のコンテンツテーブルを両立させることは大きな課題であり、そのために FMCAM はマルチポートでありながら比較器数の増大を抑えている。これを実現するための改良点の中で特徴的なものは、カテゴリ分け処理及び比較データの即時処理であり、それらの詳細を次に述べる。

カテゴリ分け処理

3.2.1 節のマルチポート化の課題 1 で述べた問題を解決するために、比較器の位置を従来のコンテンツテーブル付近からポートモジュールへ移動することを考案

した。そのため、この2つのモジュールの中間にセレクタを配し、カテゴリ分けされた比較対象データの中から対応するもののみをポートモジュール内で比較できるようにしている。

FMCAMは、初期化時に入力される外部からのデータを内部のコンテンツテーブルに書き込む段階でカテゴリ分けを実行し、カテゴリは、そのデータの上位ビットから判定するものとしている。図3.2に示したカテゴリモジュール内では、あらかじめ指定したカテゴリの値をカテゴリRAM内に記憶させておき、入力された比較対象データに基づいてカテゴリジェネレータが対応するカテゴリを決定する。

このカテゴリに基づいて、外部からのデータは対応するコンテンツテーブル内のカテゴリに格納される。FMCAMは、このカテゴリ分けを初期化の段階で実行し、外部から入力される全てのデータに対してカテゴリ分けの処理が終了した時点で、一致検索の準備が完了したこととなる。

カテゴリ分け処理による一致検索は、まず任意のポートモジュールが比較データを受け取ったならば、対応するセレクタモジュール内のカテゴリセレクタへ比較データを送出する。カテゴリセレクタは、初期化の際にカテゴリモジュールからカテゴリデータを受け取り、そのデータを保持しているため、比較データを受け取ったならば、その上位ビットからカテゴリを判断できる。そしてマルチプレクサが、コンテンツモジュールから順次ブロードキャストされてくる比較対象データの中から適するものを選択し、ポートモジュールへと送信する。その後に、ポートモジュールで一致検索が開始されることとなる。この結果、各ポートモジュールで一致検索を並列に行なえるようになったため、コンテンツテーブルの近傍に位置している比較器をポートモジュール固有のものとし、その数を大幅に削減できた。

比較データの即時処理

3.2.1節のマルチポート化の課題2で示した問題を解決するため、新たな機構としてリングカウンタを通常のアドレスカウンタの代わりに採用した。リングカウンタは、コンテンツテーブルの初期化が終了した後、入力データの到着に関わらず常時ループカウントし続けることとした。なお、1サイクルのカウント数はブロック内のワード数である r で決まるため、0から $r-1$ となる。

図3.6に示すように、ある時点においてポート0と $p-1$ に比較データ CD_0 及び CD_{p-1} が到着したならば、その時点のアドレス a_i をそれらのポートモジュール内のレジスタに保持することで検索開始アドレスを記憶した後、比較処理を開始する。その後ポートモジュールには逐次アドレスが、 $a_{i+1}, a_{i+2}, \dots, a_{r-1}, a_0, a_1, \dots, a_{i-1}$ と送られてくるので、次の a_i が来たならばアドレスが一巡したこととなるため、一連の比較処理を終了する。このようにして、コンテンツテーブル内の対応するカテゴリの比較対象データ全ての検索を行うことが可能となる。ポート0及び $p-1$ の比較処理中に、ポート1に比較データ CD_1 が到着した場合でも、その時点の処理開始アドレス a_j をポートモジュール1内のレジスタに保持することで

待ち時間を必要とせず、ポート1も比較処理を即座に開始できる。

このようなリングカウンタの適用で、並列CAMを用いなくとも、FMCAM単体で同様の処理を行なえるようになった。

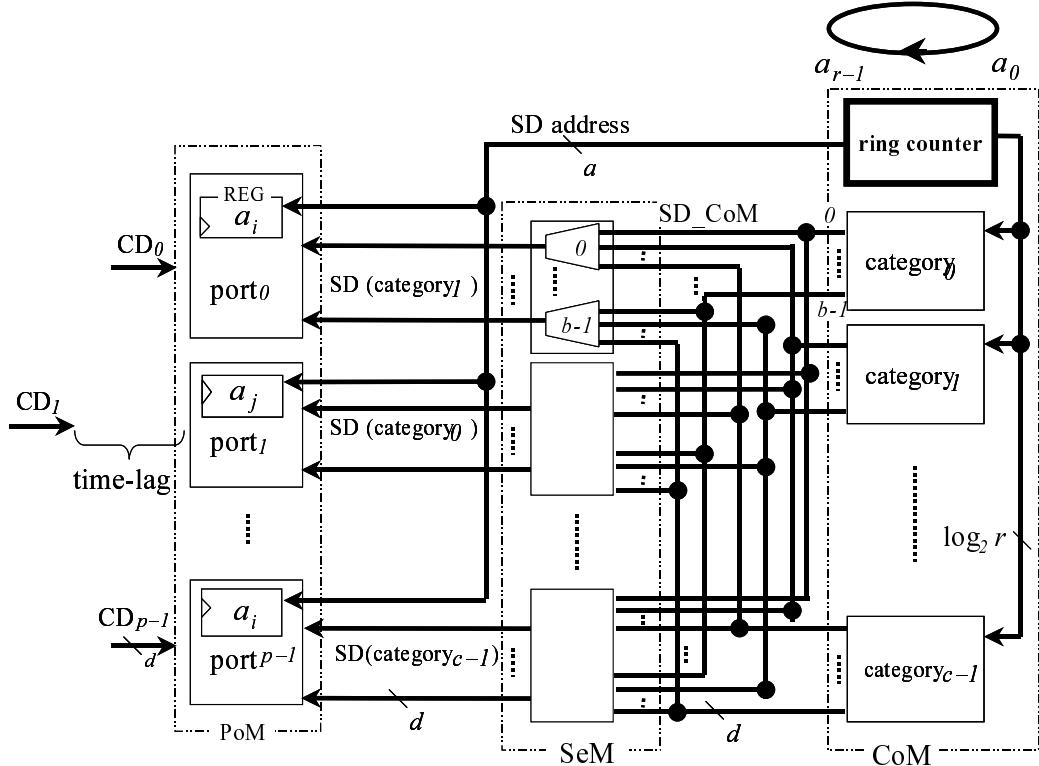


図3.6: リングカウンタを用いた比較方式。

3.2.5 FPGAへの実装結果

本節では、提案したFMCAMをXilinx社製FPGAであるXC2V6000に実装した。図3.1に使用した、FPGAボードである三菱電機エンジニアリング製KAC-02Aの写真を示す。ボード上には、XC2V6000が2石搭載してあり、PCIバスを経由してホストコンピュータとデータのやり取りを行う。また、設計はVerilog-HDLを使用し、設計ツールはXilinx社のISE Foundation 4.2i、論理合成にはXST Verilogを用いた。

表3.2に設計したFMCAMの諸元及び実装結果を示す。今回は、ワード数 2^a の変化に伴う実装結果の違いを確認するため、この値を256及び1,024と異なる2種類のFMCAMを開発した。ただし、ワード長dはどちらも128 bitである。また、RAM及びレイテンシに関してはそれぞれ内訳を破線にて示している。

1. ハードウェア量

表 3.1: 実装に用いた FPGA ボード.



ワード数 1,024 の FMCAM は、ワード数 256 のものと比較して、コンテンツテーブルの容量が 4 倍になっている。このとき、既存の方式の CAM では、比較器も元の 4 倍必要となる。しかし、FMCAM は比較器をコンテンツテーブルと独立させてポートモジュールに配置しているため、ワード数の増加に対して比較器数は変化しない。増加したリソースは、主にコンテンツテーブルの容量を増加させるための増加分及びコンテンツテーブルからセレクタモジュールへのバスエリアである。結果として FMCAM は、比較器数の FPGA の基本ロジックであるスライス数による比較では 1.99 倍程度の増加にとどまった。このように FMCAM は、コンテンツテーブルを拡大させたにも関わらず、ハードウェア量の増加を少なく抑えられていることが確認できた。

2. 構成

FMCAM のレイテンシは、入力データが確定してから、出力データが確定されるまでのクロックサイクル数で測定したものである。比較にかかるクロックサイクル数 (Comparison latency) が、ワード数によって差異があるが、これは BPBP 方式の特性によるものであり、ブロック内のワード数 r が 8 及び 16 と異なるためである。

動作周波数に関しては、回路規模が小さいワード数 256 のものが回路規模の大きい 1,024 のものより通常高くなると考えられるが、今回の実装結果では反対になっている。この理由としては、使用した FPGA である XC2V6000 が大規模回路設計向けのものであることが考えられる。XC2V6000 の全体の使用可能スライス数は 33,792 であり、これより各 FMCAM の使用スライスの

表 3.2: FMCAM の FPGA への実装結果.

CAM size (2^a)	256	1,024
CAM width (d)	128	128
Ports (p)	4	4
Words in a block (r)	8	16
Blocks in a category (b)	4	4
Categories (c)	8	16
Slices	9,107	18,160
Total Block RAMs	33	65
- Contents table RAMs	32	64
- Category RAM	1	1
Total latency	16	24
- Input latency (l_i)	4	4
- Comparison latency (r)	8	16
- Output latency (l_o)	4	4
$f_{\text{MAX}}(\text{MHz})$	22.928	26.577

割合は 1,024 のものが 54%, 256 のものが 27% となる。よって 256 のものは外部ピンから内部のレジスタまでのパスやその他のレジスタ間のパスが 1,024 のものと比較して長くなるため動作周波数は低くなったものと考えられる。

3.2.6 性能評価

FMCAM の性能を評価するために、比較方式及び既存の CAM との比較検討を行なった。尚ここでは、幅広い観点から評価を行うために、ASIC, FPGA 及び並列 CAM を評価対象としている。

1. CAM の比較方式による性能評価

ここでは、各方式の CAM の性能を、比較器数と比較にかかるクロックサイクル数の両面から評価してみることにする。

FMCAM と BSWP (Bit-Serial Word-Parallel) 方式 [36], BPWP 方式, BPBP 方式及びカテゴリ分けを行う CAM, それぞれの比較器数の算出式 N は、表 3.3 の様になる。

ここで、 a はコンテンツテーブルのアドレスのビット幅, d は比較データのビット幅, c はカテゴリの個数, b は 1 カテゴリ内のブロックの個数, r は 1 ブロック内の比較対象データの個数, そして p はポート数である。各式の全

表 3.3: 構成方法による比較器数及びクロックサイクル数.

Construction	Number of comparators (N)	Number of clock cycles
BSWP	2^a	d
BPWP	$2^a d$	1
BPBP	$\frac{2^a}{r} d$	r
Category	$\frac{2^a}{c} d$	1
FMCAM	$p \frac{2^a}{cr} d \quad (= bdq)$	r

体的傾向として、いずれの N も 2^a に比例しているため、 a の増加に対して指數関数的に増加することになる。

BSWP 方式においては、 N の値に d を乗じないため、他の N と比べて最も小さいが、比較に要するクロックサイクル数が d となるため比較処理に多くの時間を要する。BPWP 方式の比較にかかるクロックサイクル数は 1 であり最も高速であるが、他の式と比べて最も N の増加が著しいものとなる。よってこれらの方程式は、適用アプリケーションに対して制限が大きいものといえる。

BPBP 方式及びカテゴリ分け方式においては、 r もしくは c で除された分、 N の増加が抑えられていることが分かる。しかし、BPBP 方式は r を過度に増やすと比較速度は低下し、カテゴリ分けを行う CAM では一般的に BPWP 方式を用いているので、比較にかかるクロックサイクル数が 1 ではあるが、カテゴリ分けのための機構が別に必要となり、この処理に数クロックかかる。また、 c の値は適用するアプリケーションに依存することが多い。

FMCAMにおいては、 r および c の積で除されて N の値が抑えられるため、比較器数を大幅に削減することが可能となっている。また比較器をコンテンツテーブルから独立させているため、 p は 2^a に対して独立変数であり、 2^a や cr の値に対して高々僅かであるため、それを乗ずることによっても比較器の増加数は僅かで済む。FMCAMは、BPBP 方式とカテゴリ分けのデメリットを抱えることにはなるが、マルチポート化しているため処理能力が高く、このデメリットを凌駕すると考えられる。以上より FMCAM は、マルチポート化により処理能力を向上しつつ、それに伴う比較器数の増加を大幅に抑えている。またコンテンツテーブルが拡大しても、比較器数がそれほど増加しないように抑制することに成功している。

2. 既存の CAM との比較

FMCAM を含めた各 CAM の諸元、対象 LSI 及び比較方式を表 3.4 に示す。比較対象の CAM としては、各ベンダから現在リリースされている標準的な CAM を選択した。これらのうち FPGA をベースとしたものとしては、次の 3 種類を選択した。Xilinx 社のアプリケーションノートで公開されている XAPP CAM のうち、BPWP 方式であり大規模用途向けの XAPP202 [44]、BPWP 方式であり小型で高速な XAPP 203 [45]、そして ALTERA 社から FPGA 内に組み込み済みの専用 CAM の APEX CAM [46] である。また ASIC としては、IPv6 ルーティング向けの大規模高速な CAM であるミュージックセミコンダクタ社の LANCAM MP [47] を選択した。

表 3.4: 各種 CAM のスペック及び AT 積。

Design	Device	Construction	CAM width (bits)	CAM size (words)	f_{\max}	Ports	Area (A)	Access time (T) [ns]	AT	
XAPP203	FPGA (XCV400)	BPWP	24	256	115	1	1	8.70	8.70	
APEX CAM	FPGA (20KE)		48	256	110	1	1	9.09	9.09	
LANCAM MP	ASIC (MU9C1965-A/L)		128	1,024	120	1	1	8.33	8.33	
XAPP202	FPGA (XCV400)	BPBP	16	4,096	80	1	0.06	200.00	12.5	
FMCAM	FPGA (XC2V6000)			256	23		0.06	170.46	10.9	
			128	1,024	27	4	0.02	225.76	3.53	

評価方法として、CAM 自体の総合的な性能を AT (Area-Time) 積で表したもののが文献 [32] で示されており、本論文ではこれを用いて既存の CAM と FMCAM の性能比較を行なった。以下にその算出式を示す。なお、各変数は表 3.2 に使われているものとなっている。

$$AT = \frac{N}{2^a d} \times \frac{r + li + lo}{pf_{\max}} \quad (3.1)$$

一般に AT 積は回路規模と処理時間の積であり、この値が小さいほど優れている実装であるということが言える。この式を用いて様々な実装形態の CAM を比較するため、使用するチップの面積に相当するエリア (A) として比較対象ワード数を比較器数で正規化したものを用い、一方動作周波数と一致検索に必要なクロックサイクル数から算出された 1 ポートあたりのアクセスタイム (T) を処理に要する時間として、両者の積 (AT) を求めたものである。な

お、正規化に関しては、実際にCAMのハードウェア量のうち、特に比較器の数がボトルネックとなっているという背景を考慮したものとなっている。これらのAT積を算出した結果も表3.4に併せて示す。

まずエリアの値に関しては、BPWP方式の場合、全比較対象データの各ビットに比較器を持っているため全て1となる。これに対してBPBP方式では、比較器数を削減できるため、XAPP202及びワード数256のFMCAMの場合に0.06、ワード数1,024のFMCAMの場合に0.02とそれぞれ低い値を示していることが分かる。特にFMCAMは、比較器がコンテンツテーブルから独立してポートモジュールに配置してあるので、コンテンツテーブルの容量を4倍に拡大しても、直接影響は受けず、比較器数に変化はない。そのため今回の設計では、ワード数が異なる2種類のFMCAMのNの値は同値となり、エリア値はワード数1,024のほうが低い値を示している。

アクセスタイムに関しては、BPBP方式の場合、BPWP方式の値と比較してどれも高い値にならざるを得ない。また、今回作成した2種類のFMCAMは、遅延パスの改善等配置配線時の制約条件がほぼ初期設定であるため、動作周波数は既存のものと比較して低いものとなっている。しかし、マルチポート化されており複数の処理を並列に実行できるので、アクセスタイムを算出する際、ポート数pで除することができる。そのためアクセスタイムがある程度抑えられる。

このようにして算出されたAT積を各CAMで比較すると、ワード数1,024のFMCAMが、最も良い結果を示していることが分かった。また、ワード数256のFMCAMは、同じBPBP方式のXAPP202より優れた値を出しているものの、BPWP方式のCAMと比較すると劣っていることが分かる。この結果よりFMCAMは、ワード数を大きくするほどAT積が低くなり、有利となる事が分かった。これは、カテゴリ分け処理によって比較器の数を徹底して削減しているため、コンテンツテーブルの拡大に対して比較器の数の増加が抑えられていることが反映したためである。

3. マルチポートCAM及び並列CAMとの比較

FMCAMにおけるカテゴリ分けによる比較器の数の削減効果を詳しく検証するため、並列CAMとFMCAMからカテゴリモジュール、セレクタモジュール及びリングカウンタ等の機構を取り除き、BPBP方式のCAMをマルチポート化したマルチポートCAM(MCAM)を作成して、ハードウェア量の比較を行なった。これら3種類のCAMのハードウェア量のポート数に対する平均増加率を算出した結果を図3.7に示す。ここでは、ポート数pを1から4まで変化させつつ算出した値を示している。MCAMは並列CAMと比較してハードウェア量が少なくなり、ポート数pを1から4まで変化させた場合、平均増加率の差は最大で1.64倍となった。これはコンテンツテーブルを

共有化することによるハードウェア量の削減の効果である。一方、FMCAMの場合にはさらにハードウェア量を少なくでき、並列CAMとFMCAMの平均増加率の差は最大で2.36倍にもなった。これはカテゴリ分けによって比較器をコンテンツテーブルから独立させ、ポートモジュールに格納した結果であると考えられる。

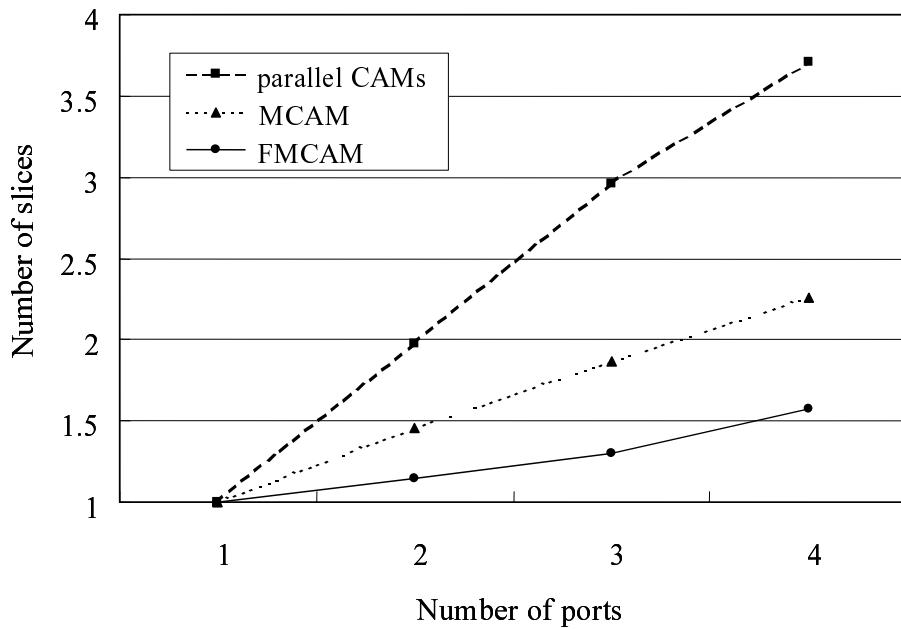


図3.7: FMCAM, MCAM及び並列CAMのハードウェア量の比較。

3.3 テーブルルックアップ用マルチポートCAM

マルチメディアデータ処理におけるボトルネックなっていたテーブルルックアップ処理は、従来のアーキテクチャでは並列化が困難であることをこれまで述べた [2], [3]。そこで、並列一致検索能力の高いFMCAMをテーブルルックアップ処理に利用することで効果的な変換処理の実現が期待できる。

図3.8に、マルチメディアデータ処理向けマルチポートCAM (Adapted FMCAM: Adapted Flexible Multi-ported Content Addressable Memory) の全体図を示す。基本的な概観は、図3.1と同様であるが、テーブルルックアップ処理にハミング距離を検索する必要は無いと判断し、これに関するポートは除いてある。図3.9に、Adapted FMCAMのモジュール構成を示す。図3.2に示した、これまでのFMCAM(以下、Original FMCAMと呼ぶ)の構成と異なり、セレクタモジュールはポートモジュールに組み込み、まとめてポートブロック (Port block) として設計の容易さを実現した。また、カテゴリモジュールはリングカウンタと共に1つのコントローラとし、図3.10に示すように、Adapted FMCAMの制御及びデータの入出力を1つのメモリ空間にまとめた。これによって、データの入出力、制御が全てアドレスを設定することで可能となり、初期データの入力等のピンを不要とし、Adapted FMCAMの制御を一元化することが可能となった。更に、これまでのコンテンツモジュールからリングカウンタ等を除いてカテゴリブロック (Category block) とした。カテゴリブロックは、1ポートのSRAMを用いて多バンク構造としたため、LSI実装時にスタンダードメモリセルを使用できるようになった。

この節では以降、3.2節で述べた、FMCAMをマルチメディアデータ処理に活用するために施した3つの工夫について述べる。

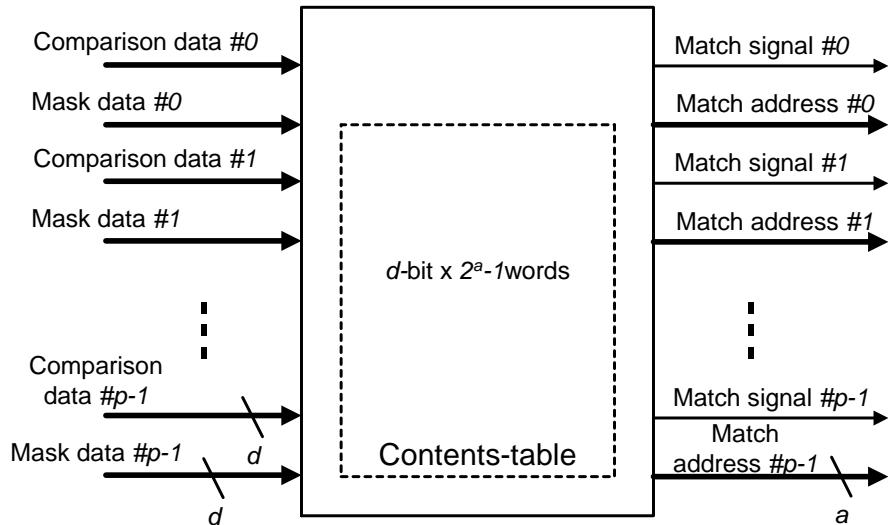


図3.8: Adapted FMCAMの全体図。

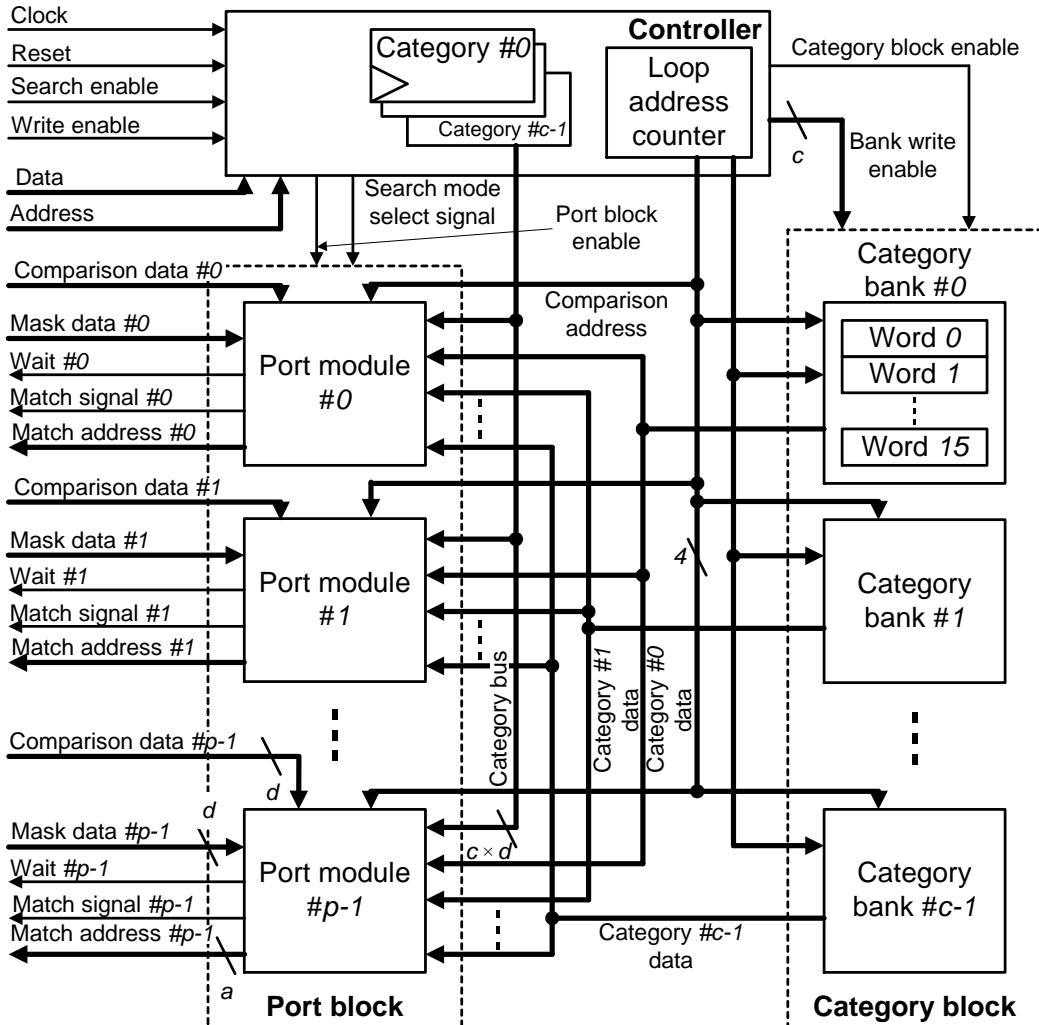


図 3.9: Adapted FMCAM のブロック図。

3.3.1 マルチプル／シングルサーチモード (Multiple/Single Search Mode)

Original FMCAM は、マルチポート化に伴う比較器の増加を押さえるために、一般的の CAM で用いられている BPWP 方式の変わりに BPBP 方式を採用している。これにより、3.2.6 節で示すように、AT 積において優れた値を示すことができている。しかしながら、リアルタイムに大容量のデータが配信される近年のマルチメディアアプリケーションを高速に処理することを考慮すると、比較にかかるクロックサイクル数を更に削減し、BPWP の処理クロックサイクル数に近づける必要がある。ここで、一般にマルチメディアデータ処理におけるテーブルルックアップ処理の特徴を検討すると、符号化前のデータと、符号化後のデータは必ず 1 対 1 であることが分かる。そのため一回の検索処理サイクルで、符号化前のデータに対

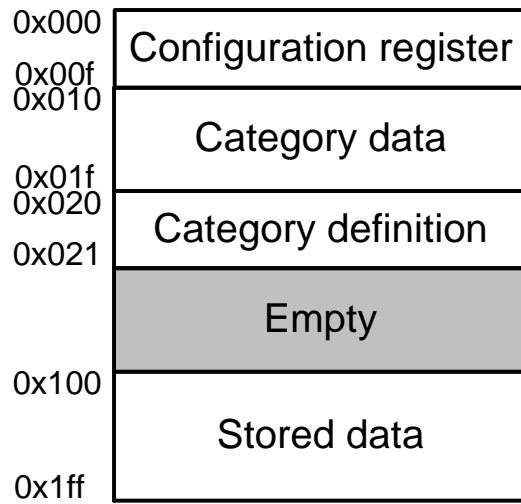


図 3.10: Adapted FMCAM のメモリ空間.

する一致検索結果が出力された後は、その以降のデータを調べる必要がない。以上のことから、Adapted FMCAM は、一致検索処理に 2 種類のモード，“マルチプルサーチモード (Multiple search mode)”と“シングルサーチモード (Single search mode)”を用意した。これらのモードはコントローラからポートブロックを制御することで実現される。

図 3.11 に一致検索時の両モードの動作波形を示す。図 3.11-(a) の例は、マルチプルサーチモード、すなわち Original FMCAM の検索動作である。このモードは比較対象データ内に、検索データと同一のものが複数存在している状況を前提としているため、図中では 1 回のサーチタイム (Search time) で、2 つのマッチシグナル (Match signal) が output されている。このサーチタイムにかかるクロックサイクル数は、アプリケーションによらず一定である。図 3.11-(b) の例は、Adapted FMCAM に、新たに追加された検索動作であるシングルサーチモードの動作波形である。テーブルルックアップ処理は、一度マッチシグナルが output された後は、検索処理をする必要がない。そのため図中で示しているように、マッチシグナルが output された後は、直ちに次の検索動作へ移行することを可能とした。結果として 1 回のサーチタイムが短くなり、次々と新しいデータの検索処理を行うことが可能となった。

3.3.2 カウンタ値設定モード (Counting Value Setting Mode)

Adapted FMCAM は、比較器の削減のために BPBP の採用だけでなく、3.2.4 節で述べた、カテゴリ分け処理も採用している。この工夫を行うことで、各ポートとも、全てのデータを検索する必要が無く、1 つのカテゴリバンクに検索範囲を制限

することが可能となった。しかしながら、1つのカテゴリバンクに格納できるワード数は固定される。そのため、このワード数がそのままループアドレスカウンタの最大値となり、一致検索処理に必要とするクロックサイクル数となっていた。一般に1つのカテゴリに格納するデータの個数はアプリケーションによって異なり、検索処理に無効なワードも保存しなければならない場合が生じる。図3.12-(a)の例に示すように、カテゴリバンクに4つの無効なデータが存在している場合、検索データが入力されたとしても、始めの4クロックサイクルは、検索処理に関係がないため、直接処理時間の増加につながることとなる。この問題を解決するために、Adapted FMCAMは、ループアドレスカウンタによる検索クロックサイクル数を、ユーザが任意に変更できる仕様にした。図3.12-(b)の例に示すように、本来であれば1回の検索処理に16クロックサイクルかかるところを、12クロックサイクルでループするように設定することが可能である。この例の場合は、図3.12-(a)と比較して、サーチタイムが約2分の1となっていることが分かる。

3.3.3 カテゴリ結合モード (Scalability of Categorization Structure)

カテゴリ分け処理を行っているCAM[37],[38]の多くは、1カテゴリの容量を変化させることはできない。そのため、1つのカテゴリに格納するためのデータ数が許容量を超えるアプリケーションには、適用が難しくなる。そこで、Adapted FMCAMは、広範囲のアプリケーションに適用できるように、カテゴリバンクの格納にスケーラビリティを持たせるように工夫した。すなわち、あるカテゴリのデータ群が1つのカテゴリ容量に格納できない場合には、2つのカテゴリを結合して1つのカテゴリとすることを可能とした。これによって適用できるアプリケーションの幅が広がることが期待できる。一致検索処理に必要なクロックサイクル数は倍となるが、シングルサーチモードを適用した場合にはクロックサイクル数を削減することが可能である。

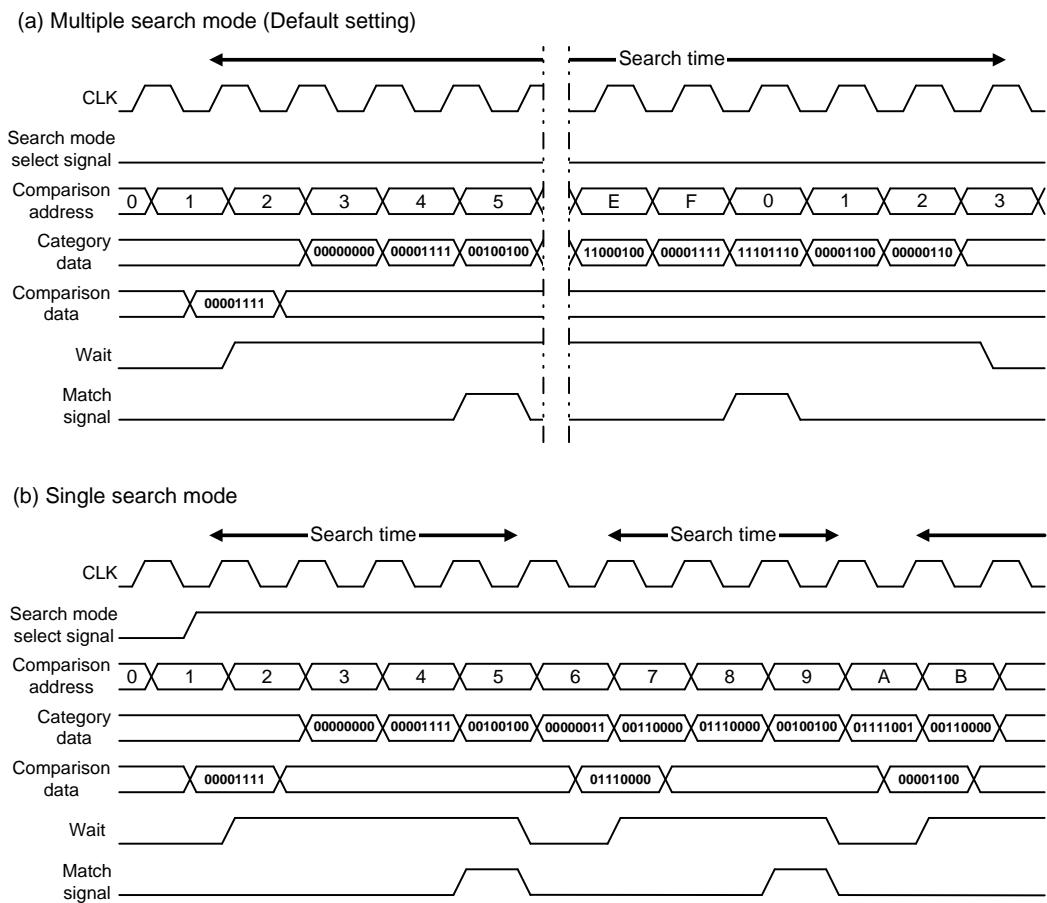


図 3.11: Adapted FMCAM による一致検索時の動作波形 : (a) マルチプルサーチモード, (b) シングルサーチモード.

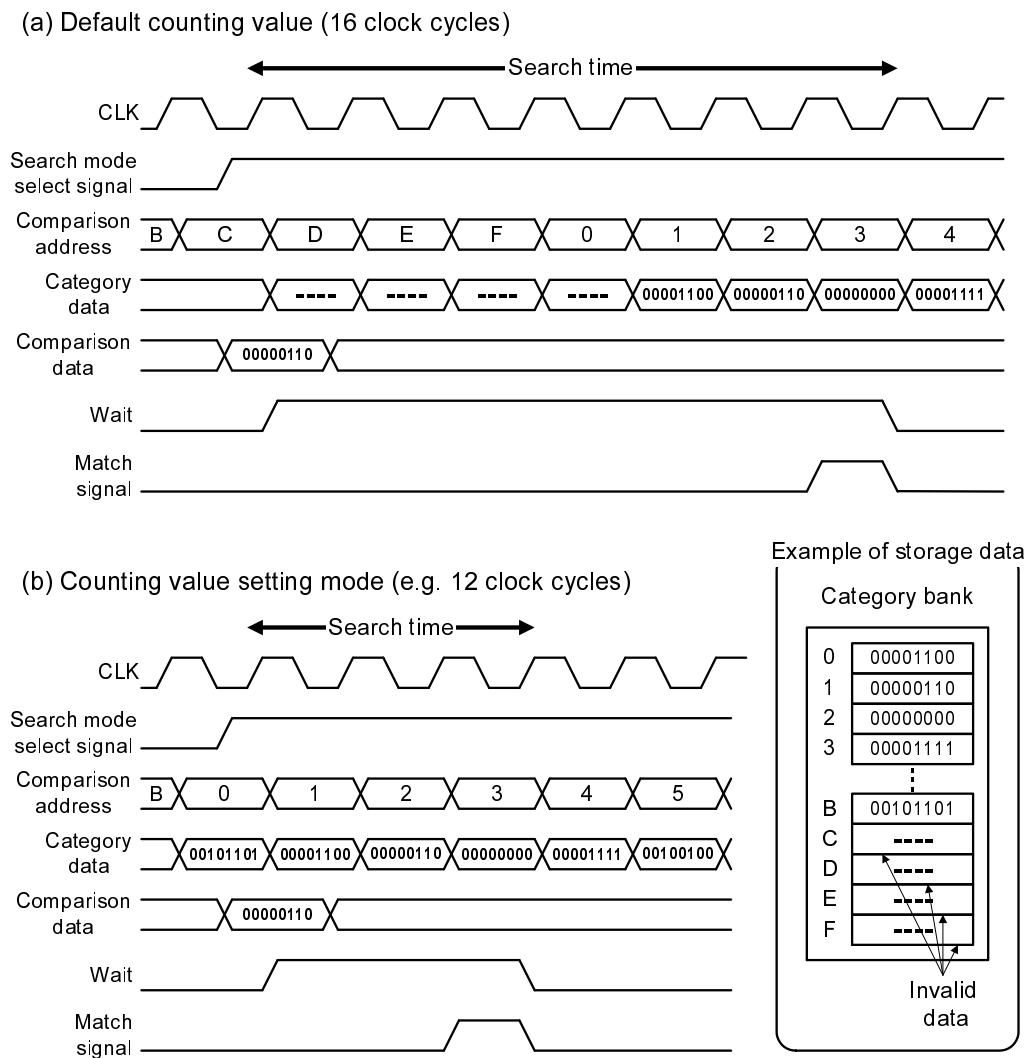


図 3.12: Adapted FMCAM による一致検索時のカウンタ動作 : (a) 通常モード, (b) カウンタ値設定モード.

3.4 アーキテクチャ

Adapted FMCAM は、大きく分けてポートブロック (Port block), カテゴリブロック (Category block), 及びコントローラ (Controller) から構成される。これらは、互いに機能を独立させて設計されているため、ポート数の増減はポートモジュールの個数を変化させるだけでよく、スケーラビリティ性が高い。以下の節では、各ブロックの動作概要及び構成を述べる。

3.4.1 ポートブロック (Port block)

図 3.9 に示すように、ポートブロックは p 個のポートモジュールから構成される。このモジュールの個数が Adapted FMCAM のポート数及び並列処理能力に反映されるため、ポートモジュールの個数が多いほど検索能力が向上する。図 3.13 に、ポートモジュールの構成を示す。このモジュールは、 d -bit の検索データ及びマスクデータを入力することができ、1 bit の検索結果信号と、そのデータが格納されている a -bit のアドレスを出力することができる。一致検索処理に関して、ポートモジュールは他のポートモジュールと同期を取ることなく動作させることができるので、検索データを入力させたならば、直ちに検索処理に取り掛かることが可能である。

Adapted FMCAM のポートモジュールは、大きく分けて以下に示す 9 種類のユニットと複数個のレジスタ及び論理回路から構成される。

- 一致検索用比較器 (Search comparator) $\times 1$
- カテゴリ選択用比較器 (Category comparator) $\times c - 1$
- ループアドレスカウンタ用比較器 (Comparator) $\times 1$
- カテゴリデコーダ (Category decoder) $\times 1$
- マルチプレクサ (MUX: MULTipleXer) $\times 1$
- デマルチプレクサ (DMUX: DeMULTipleXer) $\times 1$

ポートモジュールに検索データが入力されたならば、カテゴリ選択用比較器は、データ内のカテゴリを決定するビット箇所とコントローラ内の中のカテゴリデータを比較し、カテゴリ選択信号 (Definition signal) を出力する。この信号はカテゴリデコーダにてカテゴリアドレスに変換され、マルチプレクサを制御する。マルチプレクサは、検索データに属しているカテゴリのデータを選択し、これらのデータは順次一致検索用比較器に送られる。一致検索用比較器は送られてきたカテゴリのデータと検索データを比較し、一致したものがいれば一致信号を及びそのアドレスを出力する。上記の処理と並行してループアドレスカウンタから送られてく

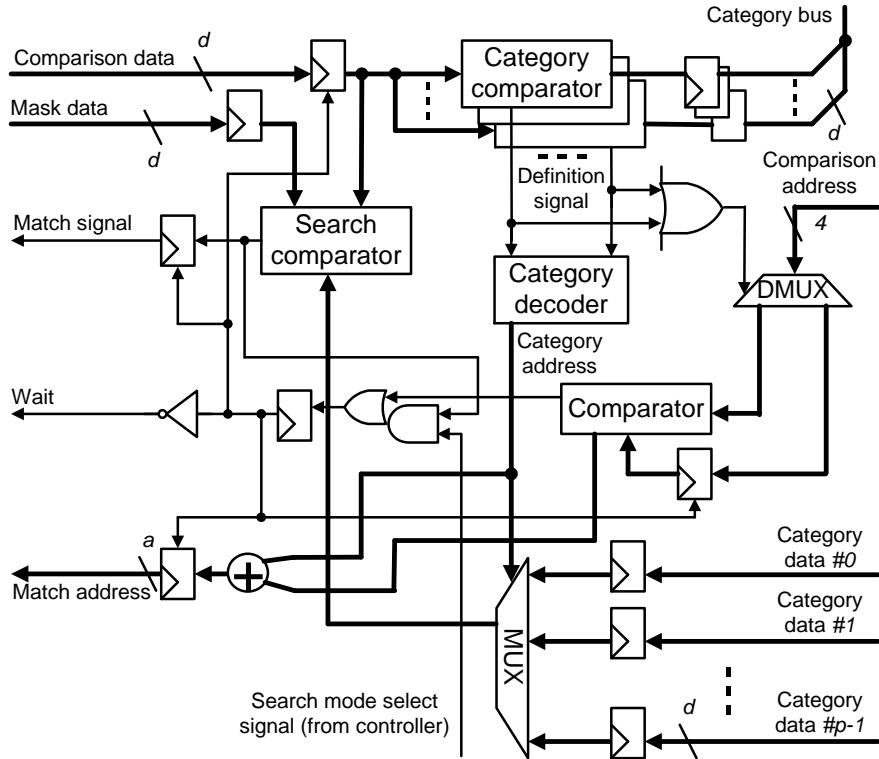


図 3.13: Adapted FMCAM におけるポートモジュールのブロック図.

るアドレスは、カテゴリ選択信号が出力された際にレジスタに格納される。ここで格納されたレジスタはループアドレスカウンタが再び同じアドレスを送信するまで保持されることとなる。この間が1回の一一致検索処理にかかる時間となる。また、ループアドレスカウンタは全てのポートモジュールにブロードキャストされているが、格納するレジスタは各ポートモジュール固有なため、各ポート一致検索に同期を取る必要が無い。以上は、マルチプルサーチモードの検索行程となるが、シングルサーチモードが選択されている場合、そのイネーブル信号 (Search mode select signal (from controller)) が、コントローラから入力され、一致検索結果と AND 演算された後に、直ちに各レジスタはリセットされることとなる。

3.4.2 カテゴリブロック (Category block)

Adapted FMCAM は、比較器の配置を工夫することで、実装に関し様々な利点を有している。従来の CAM は、検索データと全比較対象データを並列に検索するため、メモリセル内に比較のためのトランジスタを追加しなければならない。しかしながら、この手法は以下に示す2つの問題がある。

1. CAM を LSI として実装する際に、専用の CAM セルを設計、もしくは使用し

なければならない。そのため、動作速度及び面積において最適にカスタマイズされた、多くのスタンダード SRAM セルを使用することができない。よって、実装面積も大きくなり、適用アプリケーションに制限が生じる。

2. マルチポート化が困難となる。通常の CAM セルをマルチポート化する場合ポート数分 XNOR を構成するトランジスタを追加する必要が生じる。これは実装面積及び消費電力等の面で問題がある。

以上の問題を克服すべくいくつかの研究が行われているが [30], [48]、実現された例はない。そこで、Adapted FMCAM は、比較器をメモリセルから分離し、ポートモジュールとしてまとめて配置することで、この問題を解決した。ポートモジュール内の比較器には、各メモリセルからデータがブロードキャストされており、検索データの入力にあわせて一致検索処理を行うことが可能である。この方式を採用することによって、Adapted FMCAM は FPGA, ASIC と実装ターゲットにとらわれることなく、組み込みメモリやスタンダードメモリセルを使用することが可能となり、比較器全体の面積も削減している。また、設計に要する時間も、通常の CAM と比較して短期間になることが期待され、実装面積も縮小されると考えられる。

3.4.3 コントローラ (Controller)

コントローラは、主としてカテゴリ情報を持ったレジスタとループアドレスカウンタによって構成される。ユーザから格納されたカテゴリのデータはマスクによってビット位置を指定された後、各ポートモジュールにブロードキャストされる。ループアドレスカウンタは 3.2.4 節で述べたリングカウンタの動作と同一であり、アドレスを降順にカウントし、最大値に達した後、0 から再びカウントを開始する。ただし、カウンタ値設定モードでカウンタの最大値が指定された場合は、その値でループを繰り返すことになる。

3.5 FPGA / ASICへの実装結果

Original FMCAM は、既存の CAM と比較して AT 積で優れており、並列 CAM と比較した場合にも面積の増加度は低く抑えられていることを示した (表 3.4, 3.7 参照)。この特徴は Adapted FMCAM にも備わっている。この節では更に、FPGA 及び ASIC への Adapted FMCAM の実装結果をもとに、動作周波数、及び面積のスケーラビリティ性について述べる。

Adapted FMCAM は、ハードウェア記述言語である Verilog-HDL で設計した。FPGA への実装については、Xilinx 社の開発ツールである、ISE Foundation 7.1i を用い、論理合成は Synplicity 社の Synplify Pro 8.1 を使用した。ターゲットデバ

イスは、Xilinx 社の FPGA である、 XC4VLX160 を使用している。これらの環境で、最大動作周波数と、換算ゲート数を算出した。図 3.14 に配置配線後のマッピング状況を示す。ASICへの実装については、Synopsys 社の Design Compiler (バージョン 2005.09) によって 90 nm CMOS テクノロジで論理合成を行った。図 3.15 に、16 ポートの合成結果を示す。

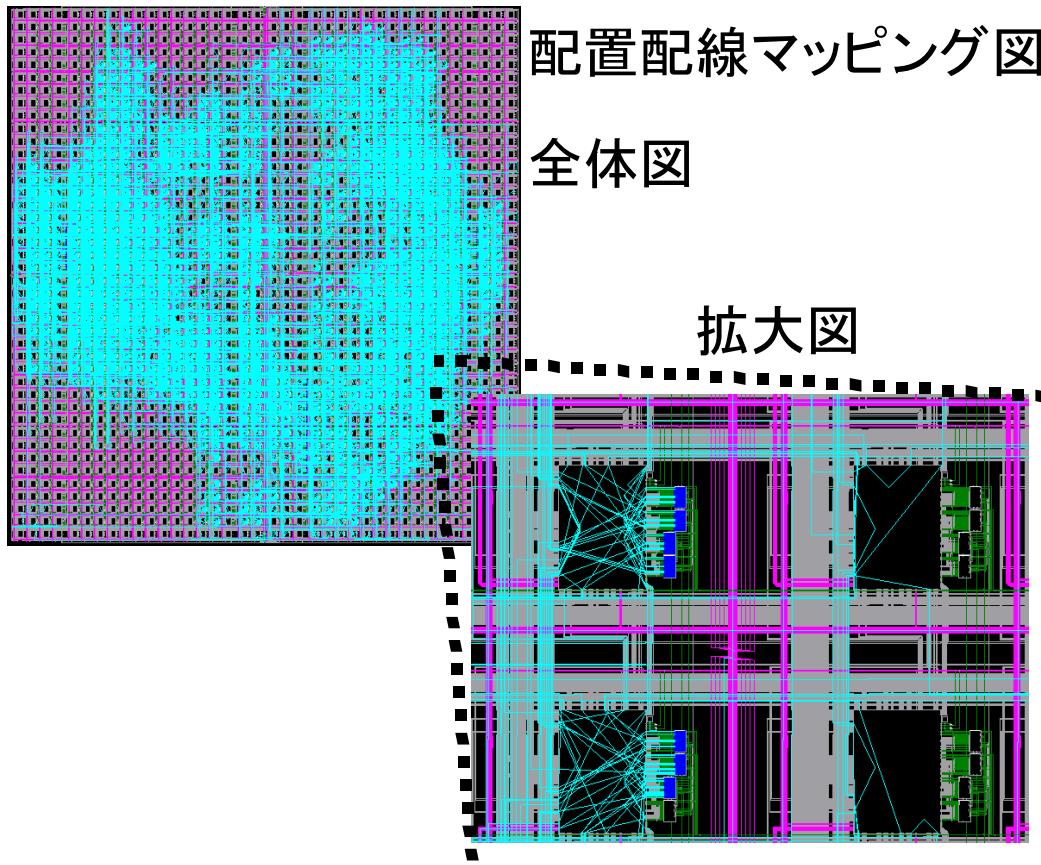


図 3.14: 配置配線マッピング結果。

以上の結果を元に、最大動作周波数及び実装面積を見積もった。Adapted FM-CAM の各パラメータの緒元は、以下の通りである。

- アドレス値: $a = 8$
- 1 ワードのビット幅: $d = 32$
- カテゴリ数: $c = 16$
- ポート数: $p = 1 \sim 16$

図 3.16-(a) は、FPGA における論理合成後の最大動作周波数 (Synthesis maximum frequency), 及び配置配線後の最大動作周波数 (P&R maximum frequency)

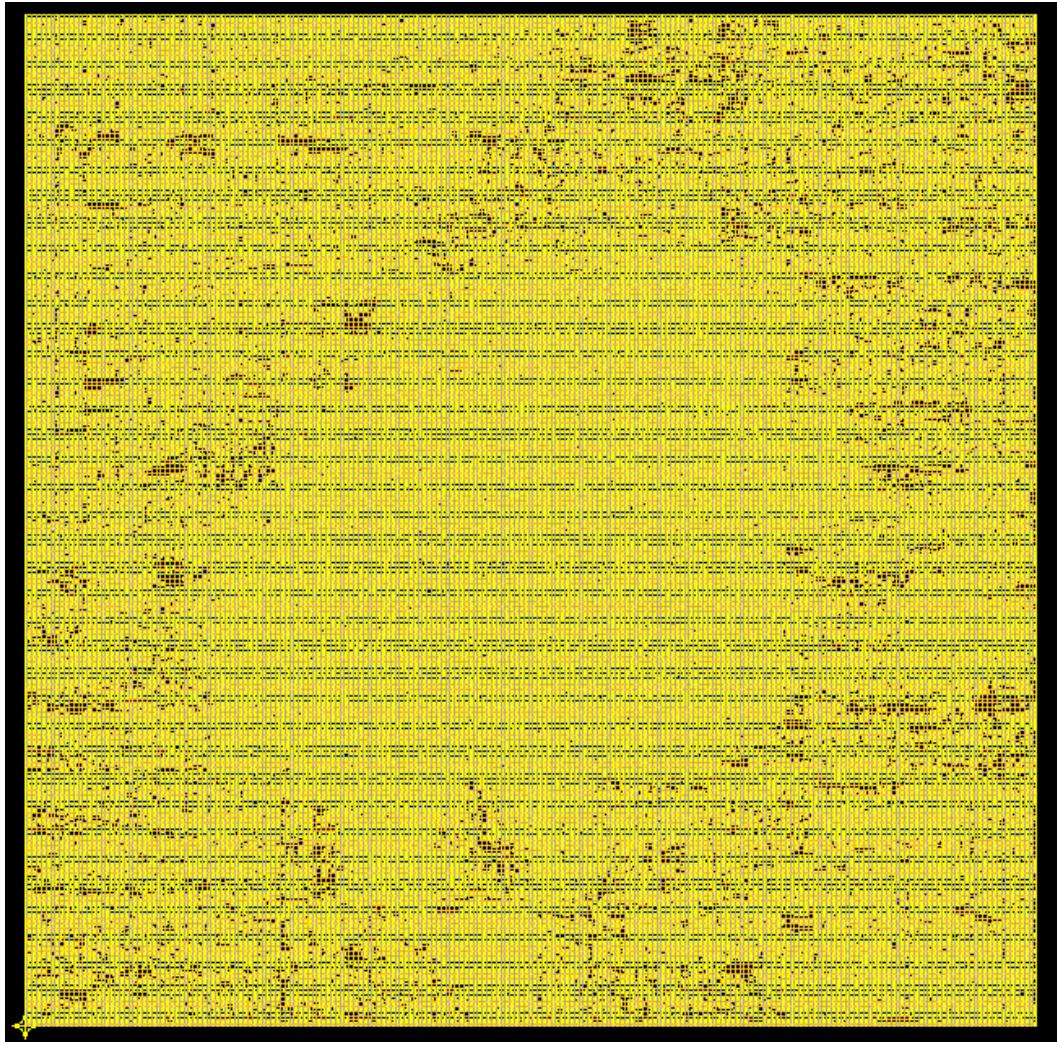


図 3.15: 90 nm 7Cu CMOS テクノロジによる 16 ポート Adapted FMCAM の論理合成結果.

を示している。また、図 3.17-(a) は、ASIC における論理合成後の最大動作周波数を示している。これらから分かるように、FPGA, ASIC どちらに実装した場合にも。ポート数の増加と比較して、最大動作周波数の低下は少ないことが分かる。特に ASIC の場合には、論理合成の結果ではあるが、ポート数の増加に対し、最大動作周波数の制約条件である 200 MHz に対し制約を満たさないことはなかった。Adapted FMCAM は、ポート数の増加に対し、ポートモジュールを追加するだけで、コントローラ、及びコンテンツブロックには影響を及ぼすことがない。ポートモジュールを増加するとポートブロックとコンテンツブロックをつなぐ信号線が増加するため、FPGA における配置配線後の動作周波数は多少の減少が見られるが、大きいものではないと言える。また、FPGA における配置配線後の動作周

波数は、ポート数が8より大きい場合FPGAのリソースが不足したため、破線で示す予測値となっている。ASICへの実装に関しては、近年の多層配線技術の進歩により、90 nm CMOSテクノロジで7層程度まで使用できる。よってポート数の増加に伴う信号線の増加に対しては、多層配線を効率よく利用することが重要となる。

図3.16-(b)は、FPGAにおける配置配線後の換算ゲート数を示している。また、図3.17-(b)は、ASICにおける論理合成後の面積を示している。ただし、ASICにおける面積見積もりは論理合成時のものではあるが、仮想配線の面積を加味したものとなっている。今回の実装ではカテゴリバンク内のメモリセルをフルカスタムによるハードマクロのものではなく、Flip-Flopベースのソフトマクロを使用した。そのため、全体の面積は大きくなっている。一般に、マルチポートアーキテクチャ、例としてマルチポートSRAMはポートの増加に対して実装面積が、2乗で増加することが知られている[49]。このことがマルチポートSRAMの普及の妨げにもなっているが、Adapted FMCAMはFPGA及びASICの実装結果共に、線形で増加することが確認できる。その増加率は1ポートの増加に対し約9%と低い。これは、ポート数の増加がポートモジュールの増加、及び配線のみにとどまるからであると考えられる。

以上の結果より、Adapted FMCAMは、FPGA、ASICどちらのデバイスでも、マルチポートの有効性を十分に発揮できる、実装にスケーラブルなアーキテクチャである見通しを得た。

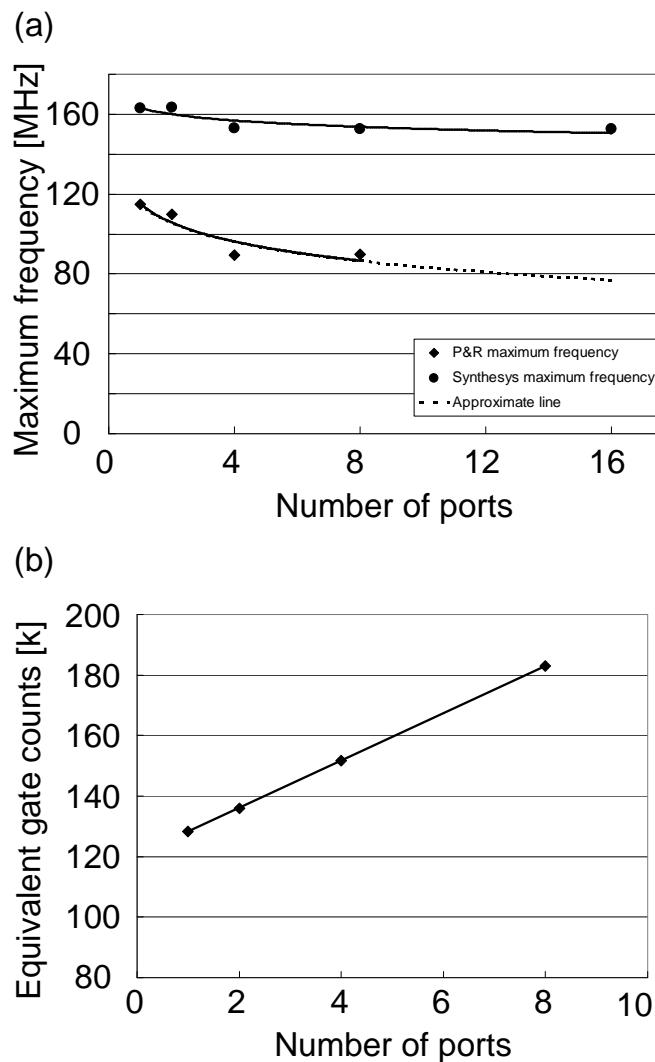


図 3.16: Adapted FMCAM のポート数増加に対する FPGA (Xilinx XC4VLX160)への実装結果：(a) 最大動作周波数, (b) ゲート数.

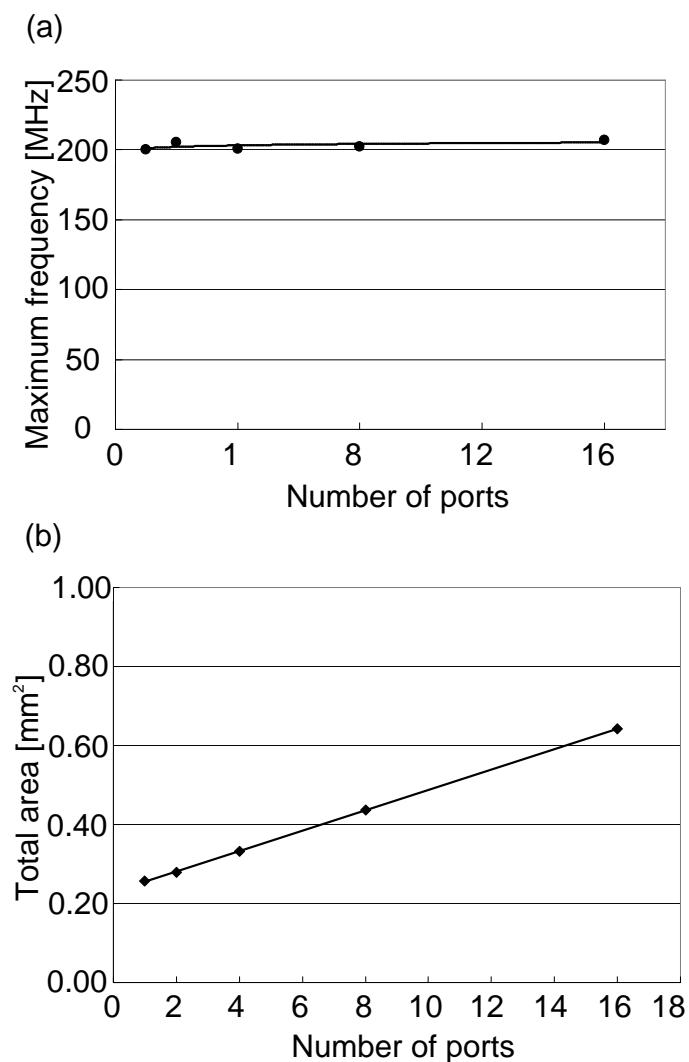


図 3.17: Adapted FMCAM のポート数増加に対する ASIC (90 nm CMOS テクノロジ)への実装結果 : (a) 最大動作周波数, (b) 実装面積.

3.6 ハフマン符号化への適用と評価

マルチメディアデータ処理を行う従来のプロセッサにおいて、テーブルルックアップ処理の効率的な並列化が困難であることは??節、及び3.1節で述べた。この節では、マルチメディアデータ処理におけるAdapted FMCAMの並列テーブルルックアップ能力を検証する。

3.6.1 並列テーブルルックアップ処理の概念

Adapted CAM の持つ複数の入出力ポートを利用して、並列テーブルルックアップ処理を行うためには、マルチポート SRAM もしくは複数の SRAM と組み合わせて処理を実現する。マルチポート SRAM には、消費電力及び面積において有効であることが報告されている階層構造バンク型多ポートメモリ (HMA: Hierarchical Multi-port Memory) [50] 等を使用することによって面積の増加を抑えることが可能である。

図 3.18 に、並列テーブルルックアップの概念図を示す。ここでは、Adapted FMCAM を用いてハフマン符号化を行う例を示している。ハフマン符号化の準備として、Adapted FMCAM には、符号化前の全データパターンを格納し、マルチポート SRAM にはハフマン符号化テーブルを格納する。Adapted FMCAM の各ポートには、任意のタイミングで符号化前データが入力される。その後、符号化前データの属するカテゴリからデータパターンがロードされるため、比較処理を行い、一致したものが見つかり次第、一致アドレスが出力される。Adapted FMCAM の各ポートとマルチポート SRAM の各ポートは 1 対 1 で対応しているため、マルチポート SRAM は、一致アドレスを受け取ったならば、直ちにハフマンコードを出力する。以上の処理を行うことによって、準備する符号化前テーブルとハフマンテーブルは 1 つづつであるにもかかわらず、これまで符号化が困難であったテーブルルックアップ処理を p 倍の並列度で実行することが可能となった。

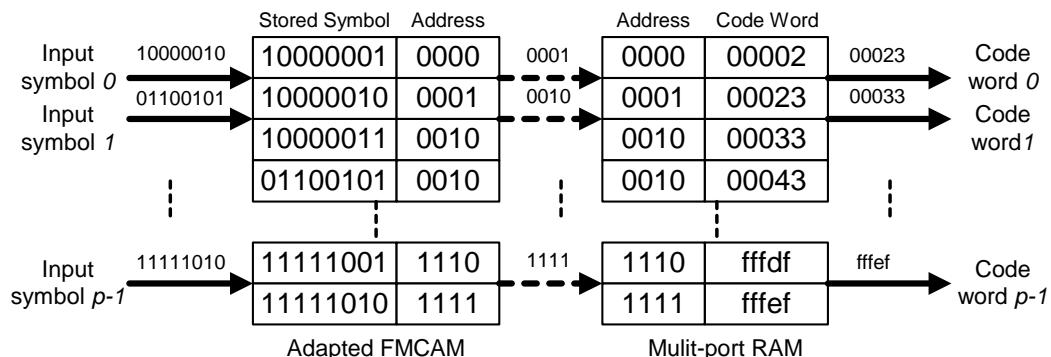


図 3.18: 並列テーブルルックアップ処理の概念図.

3.6.2 評価・検証

3.6.1節で述べた、Adapted FMCAMベースの並列テーブルルックアップアーキテクチャをハフマン符号化に適用した結果を示す。圧縮処理に用いた画像を、図3.19に示す。これら4つの画像は、 154×144 から $1,024 \times 768$ と、大小様々なサイズを用意し、人物、動物、物体、及び風景と一般に取り扱われることが多い被写体を選択した。これらの画像に、Adapted FMCAMベースの並列テーブルルックアップアーキテクチャを用いて、ハフマン符号化処理を行った。

(a) 154×144 (b) 256×256 (c) 600×480 (d) $1,024 \times 768$ 

図3.19: ベンチマーク用画像。

得られた結果を図3.20に示す。各グラフの横軸はポート数、縦軸はクロックサイクル数を示している。比較対象アーキテクチャはOriginal FMCAM、及び現在モバイル機器に搭載されている汎用品である、2命令同時発行VLIW (Very Long Instruction Word)アーキテクチャ16 bitのDSP [51]–[53]である。比較結果から分かるように、Original FMCAMは、DSPと比較してハフマン符号化処理におけるクロックサイクル数の削減を実現している。DSPは処理の並列度を変化させることはできないため、常に一定であるのに対し、FMCAMはポート数を増加することでその削減率は比例して向上する。ただし、Original FMCAMはBPBP方式をそのまま適用して一致検索処理を行っているため、1回の比較に必要な処理クロックサイクル数は16クロックと一定である。そのため画像(a)における並列度が低

い1ポートに関しては、DSPによるクロックサイクル数が小さい。これは、(a)の検証用画像がベンチマーク用として有名な被写体であり、静的ハフマン符号化テーブルによく適合しているためと考えられる。

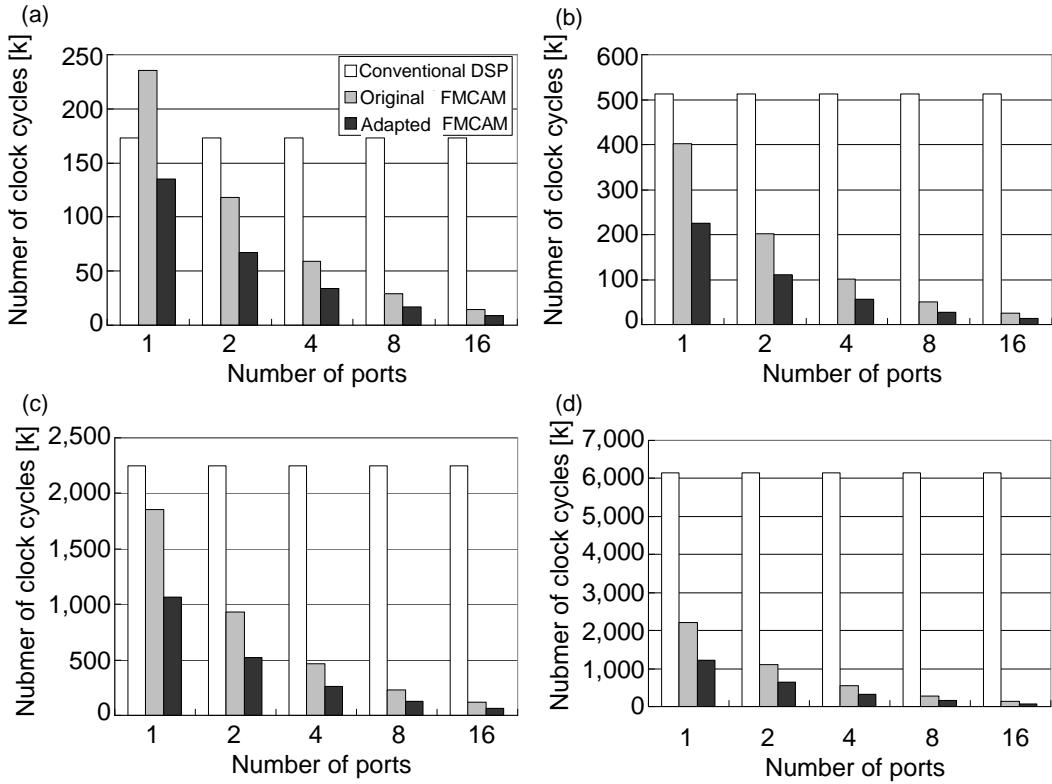


図 3.20: FMCAM アーキテクチャと DSP のハフマン符号化処理クロックサイクル数の比較。

これらの結果に対し、Adapted FMCAM は、3.3.1節、及び3.3.2節で示した、シングルサーチモード、及びカウンタ値設定モードの追加によって一致検索処理にかかるクロックサイクル数の削減を実現している。そのため、DSP 及び Original FMCAM と比較して大幅な比較クロックサイクル数の削減を実現した。画像(d)のポート数 16 の場合には DSP と比較して約 93% のクロックサイクル数削減を達成しており、Adapted FMCAM の処理クロックサイクル数は各ポート数とも Original FMCAM に対して平均 43% の削減を実現している。表 3.5 に、Original FMCAM と Adapted FMCAM をポート数 1 から 16 におけるハフマン符号処理クロックサイクル数で比較したものを示す。どのポートにおいても、クロックサイクル数の大幅な削減を達成していることから、Adapted FMCAM は、テーブルルックアップ処理というマルチメディアデータアルゴリズムで頻繁に行われる処理において、BPBP アーキテクチャの本質的な問題である、一致検索時の一定クロックサイクル数をうまく解決し、クロックサイクル数削減を達成していることが分かる。

表 3.5: 一致検索時における、ハフマン符号化クロックサイクル数の平均値.

Number of ports	Average of encoding clock cycles	
	Original FMCAM	Adapted FMCAM
1	19.00	10.79
2	9.50	5.39
4	4.75	2.69
8	2.38	1.36
16	1.19	0.68

クロックサイクル数の削減効果を実証できたため、次に動作周波数、及び実装面積を考慮して性能比較を行う。性能比較の指標は1秒あたりの処理回数である、MOPS (Mega Operations Per Second) を実装面積で割った値を使用する。また、ここで扱う1回の処理とは、1つの符号化前データに対するテーブルルックアップ処理と定義する。以下に算出式を示し、表 3.6 に算出結果を示す。

$$MOPS/mm^2 = \frac{1 \text{ クロックサイクルあたりの処理回数} \times \text{最大動作周波数 [MHz]}}{\text{実装面積 [mm}^2\text{]}} \quad (3.2)$$

表 3.6 は、ポート数を 1 から 16 まで変化させた場合の動作周波数、処理性能 (MOPS)、実装面積、及び単位面積当たりの性能 ($MOPS/mm^2$) 算出結果である。比較アーキテクチャは Original FMCAM と圧縮処理の評価で用いた汎用 DSP である。ただし、DSP はポートを複数持たないため、比較条件を統一する目的でポート数を増加した場合は並列に実装して処理することとしている。両 FMCAM と DSP の動作周波数は、約 200 MHz で動作する。ここで、Original FMCAM と Adapted FMCAM の動作周波数が同一なのは、マルチプル・シングルサーチモード、及びカウンタ値設定モードを切り替えることで、Adapted FMCAM はどちらのアーキテクチャも実現可能であるためである。今回の評価では Adapted FMCAM の検索モードをマルチプルサーチモードへ変更し、カウンタ値の設定は行っていない状態で Original FMCAM としている。性能 (MOPS) は、全アーキテクチャともポート数の増加に比例して増加することが分かる。性能 (MOPS) のみを検討すると、並列配置 DSP が優れている。しかしながら、面積の増加率がポート数の増加に比例して増加するのに対し、両 FMCAM は、3.5 節で示したように、面積の増加が緩やかなため、単位面積当たりの性能 ($MOPS/mm^2$) では Adapted FMCAM による

テーブルルックアップ符号化アーキテクチャが効果的であることが分かる。ポート数 16 の場合では、Original FMCAM と比べて 1.7 倍、並列配置 DSP と比べて 3.8 倍の向上率であった。

以上より、Adapted FMCAM はテーブルルックアップ処理を並列に行うアーキテクチャとして、有効な構成であると考えられる。

表 3.6: ハフマン符号化における Original/Adapted FMCAM と並列 DSP の性能比較。

Number of ports	Maximum frequency [MHz]		Comparison operation [MOPS]			Total area [mm ²]		MOPS / mm ²		
	Original/Adapted FMCAM	Parallel DSPs	Original FMCAM	Adapted FMCAM	Parallel DSPs	Original/Adapted FMCAM	Parallel DSPs	Original FMCAM	Adapted FMCAM	Parallel DSPs
1	200	200	10.55	18.58	25.60	0.26	0.21	41	72	122
2	205	200	21.61	38.11	51.20	0.28	0.42	78	137	122
4	201	200	42.26	74.61	102.40	0.33	0.84	127	224	122
8	202	200	85.18	149.14	204.80	0.44	1.68	195	342	122
16	207	200	174.11	302.92	409.60	0.64	3.36	271	472	122

3.7 リアルタイム符号化テーブル最適化アーキテクチャとの融合

3.5 節、及び 3.6 節等の検討で、マルチポート CAM はテーブルルックアップ処理に対して非常に有効なアーキテクチャであることが示せた。この節では、マルチポート CAM ベーステーブルルックアップ符号化アーキテクチャの高性能化のために、2 章で示した、リアルタイム符号化テーブル最適化アーキテクチャとの融合を検討する。これらのアーキテクチャを融合することで、マルチメディアデータを並列にリアルタイムで符号化しつつ、データサイズを小さくできるアーキテクチャの実現が期待できる。

3.7.1 処理の概要

3.6.1 節にて述べた FMCAM による並列テーブルルックアップ処理を、リアルタイム符号化テーブル最適化アーキテクチャに適用するには、2.3.1 節で示したエンコーダ内に FMCAM とマルチポート SRAM を配置する。図 3.21 に、そのブロック図を示す。マルチポート CAM には、FMCAM を、アクティブテーブルとシャドウテーブルには、それぞれマルチポート SRAM を利用する。エンコーダブロックのアーキテクチャ、及び処理の概要は、ほぼ 2 章で述べたものと同一であるが、

オプティマイザブロックはFMCAMからの出力ポートが複数あるためデータの出現頻度を算出するために工夫が必要となる。図3.22に並列テーブルルックアップ処理のためのアサインモジュールブロック図を示す。FMCAMの各エントリに対応しているカウンタには、それぞれORロジックを付加し、一致アドレスを論理演算してカウントすることとした。このロジックを追加することによって、データの出現頻度を計算可能となる。別々の出力ポートから同タイミングで、同一のアドレスが出力された場合は出現頻度がまとめられることとなるが、この影響に関しては、3.7.2にて述べることとする。その他のシャドウテーブルアップデート及び切り替え処理アルゴリズムに関しては同一である。

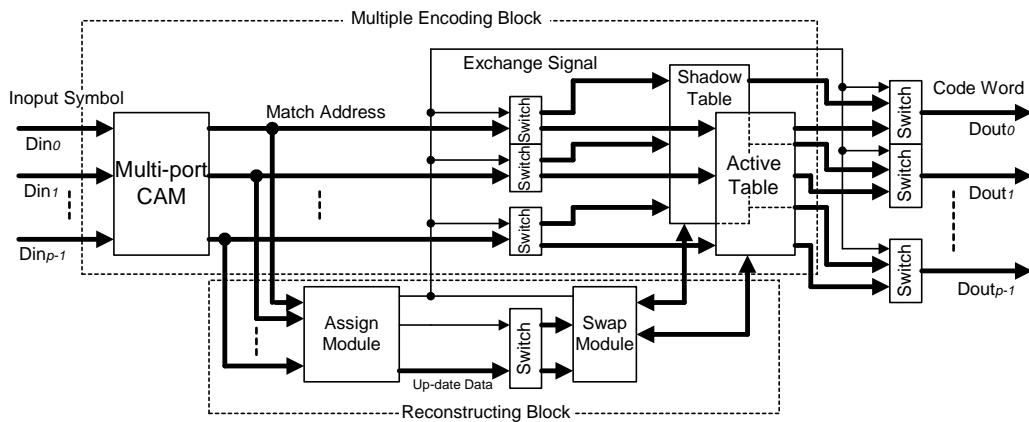


図3.21: リアルタイム符号化テーブル最適化アーキテクチャとFMCAMの融合。

3.7.2 ハフマン符号化適用における性能評価

並列テーブルルックアップアーキテクチャによる、符号化テーブル最適化効果を検証するために3.6節と同様に、JPEGアプリケーションにおけるハフマン符号化を対象として性能評価を行った。なお、ハフマン符号化の圧縮率を主として検証するために、マルチポートCAMはOriginal FMCAMのみを適用している。図3.23に検証に用いた画像を示す。各画像の特徴は、 192×128 から $1,024 \times 768$ までのサイズが異なる、(A)人物が被写体の画像、(B)使用されている色の種類が少なく、単純な画像、(C)景色の画像、及び(D)コンピュータグラフィックである。各画像とも3.6.1節での検証と異なる条件で検証を試みた。検証条件は色差のファクターである C_b 及び C_r のサンプリングを行わず、量子化係数は100としている。また、Y, C_b 及び C_r 全てを圧縮対象のデータとしている。

表3.7に検証結果を示す。比較対象アーキテクチャは、MUSIC社のシングルポートCAMであるMU9C1480A、Xilinx社のIPコアエンコーダであるXAPP616、及びAMPHION社のSRAMベースエンコーダであるCS6100とした。FMCAMは

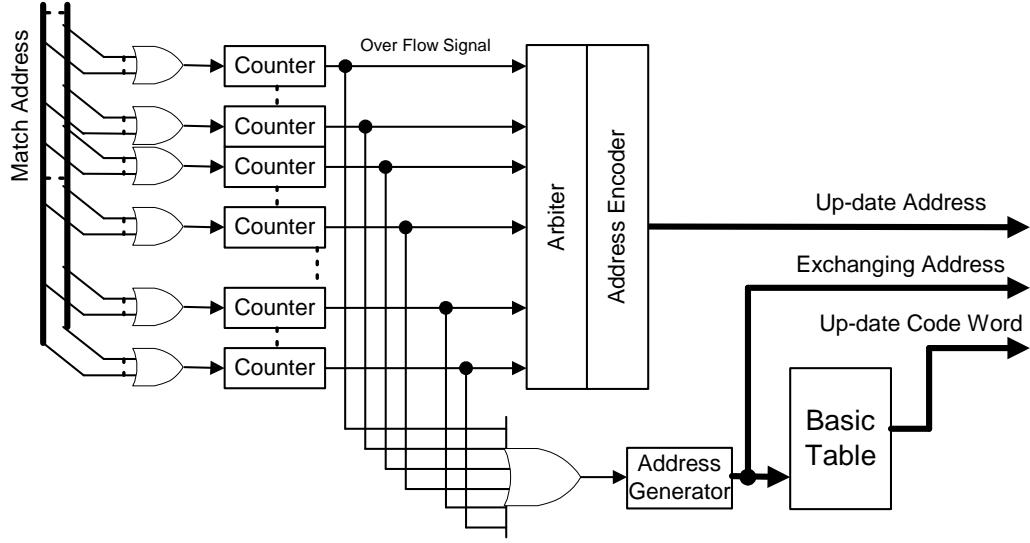


図 3.22: FMCAM 適用時のアサインモジュールブロック図.

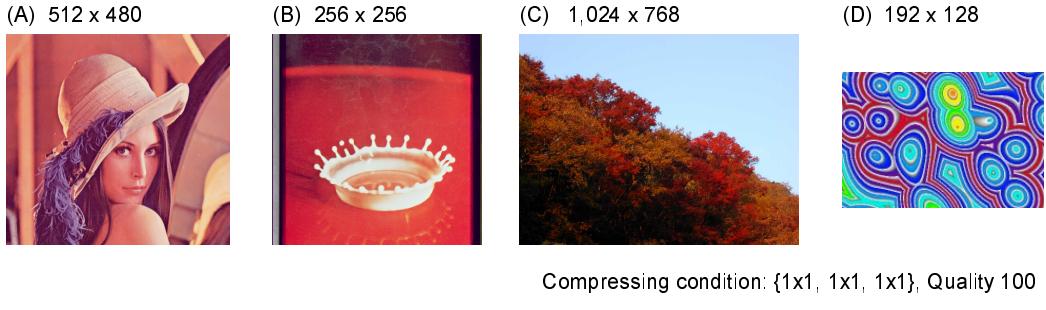


図 3.23: ベンチマーク用画像.

8 ポートを実装している。表 3.7-(i)より, SRAM ベースである CS6100 は, テーブルルックアップ処理を逐次的に比較して行わなければならないため, クロックサイクル数は最も多くなっている。MU9C1480A は CAM ベースであるため, 比較処理を 1 クロックサイクルで処理でき, XAPP616 は, 専用回路であるため比較にかかるクロックサイクル数は小さいものとなっている。これらに対し, Original FMCAM は, 複数のポートで処理できるため CAM や専用ハードウェアと比較しても約 4 分の 1 のクロックサイクル数で処理であることが分かった。

次に, 表 3.7-(ii)より, データの圧縮サイズの検証を行った。Original FMCAM 以外のアーキテクチャは, 静的ハフマン符号化を実装しているため同一の値となる。今回の検証で用いた画像は, 単純な画像, もしくはサンプル画像が多いため静的ハフマン符号化に用いている既存のテーブルに適している画像が多い。それにもかかわらず, どの画像もリアルタイム符号化テーブル最適化アルゴリズムの

表 3.7: 处理クロックサイクル数と圧縮データサイズの算出結果.
 (i) Encoding clock cycles

Device	Architecture	Number of clock cycles			
		(A)	(B)	(C)	(D)
Proposed architecture	Multi-port CAM (8 port)	184,321	49,153	589,825	18,433
MUSIC MU9C1480A	Single port CAM	737,281	196,609	2,359,297	73,729
XILINX XAPP616	PLA	737,284	196,612	2,359,300	73,732
AMPHION CS6100	SRAM	2,974,448	839,853	8,445,927	393,232

(ii) Compressed data size

Device	Picture size [byte]			
	(A)	(B)	(C)	(D)
Proposed architecture	430,945	101,238	465,129	47,089
Conventional architecture	499,940	110,984	526,810	56,019

効果が現れており、画像サイズの圧縮効果が確認できている。特に画像 (D) に関しては、約 20% の削減を達成している。これより、別々の出力ポートから同タイミングで、同一のアドレスが出力された場合の影響は少ないものと考えられる。

以上の検証よりマルチポート CAM ベースのテーブルルックアップ符号化アーキテクチャと、リアルタイム符号化テーブル最適化アーキテクチャとの融合は、テーブルルックアップ処理を高速、高圧縮で処理できることが確認できた。これにより、マルチメディアデータ処理においてボトルネックとなっていたテーブルルックアップ処理を高速かつ高圧縮に処理できることが期待できる。

第4章 CAMベース超並列 SIMD型 プロセッシングアーキテクチャ

本章では、マルチメディアデータ処理におけるテーブルルックアップ処理と繰り返し演算処理の高速化を両立する、新しいメモリベースのマルチメディア LSI アーキテクチャを提案する。テーブルルックアップ処理は、2章、及び3章にて示したCAMベースのテーブルルックアップ符号化アーキテクチャを利用する。繰り返し演算の高速な処理については、SRAMをベースとして、演算器の配置とデータの処理ベクトルを工夫することにより並列度を大幅に向上させることを実現した超並列 SIMD型演算プロセッサを新たに開発した。本章では、上記2つのメモリベースアーキテクチャを融合することで、プログラマブルに様々なマルチメディアアプリケーションの高速な処理を可能としながら、小面積、低消費電力である、CAMベース超並列 SIMD型プロセッサのアーキテクチャを提案し、その性能評価について述べる。

本章の構成は、??節において、CAMベース超並列 SIMD型プロセッサの基本概念について述べる。??節では、繰り返し演算処理を高速に処理することが可能な SRAMベース超並列 SIMD型プロセッサについて詳述し、4.1節において、CAMベースのテーブルルックアップ符号化アーキテクチャとの融合について論じるとともに、CAMベース超並列 SIMD型プロセッサについて詳述する。4.2節では、提案アーキテクチャを JPEG アプリケーションに適用した場合の性能を評価し、4.3節にて、本章のまとめを述べる。

4.0.3 動作概要

超並列 SIMD型プロセッサは、SRAM領域から演算器にデータを送り込むことによって四則演算等の処理を行うことが可能である。以下に加算、減算、乗算、及び除算の例を示す。尚、図中に示す X はテンポラリレジスタ、CB はキャリーを格納するレジスタ、M はマスクレジスタである。加算の例を図 4.1、減算の例を図 4.2、図 4.3、乗算の例を図 4.4、図 4.5、及び除算の例を図 4.6、図 4.7、図 4.8 に示す。

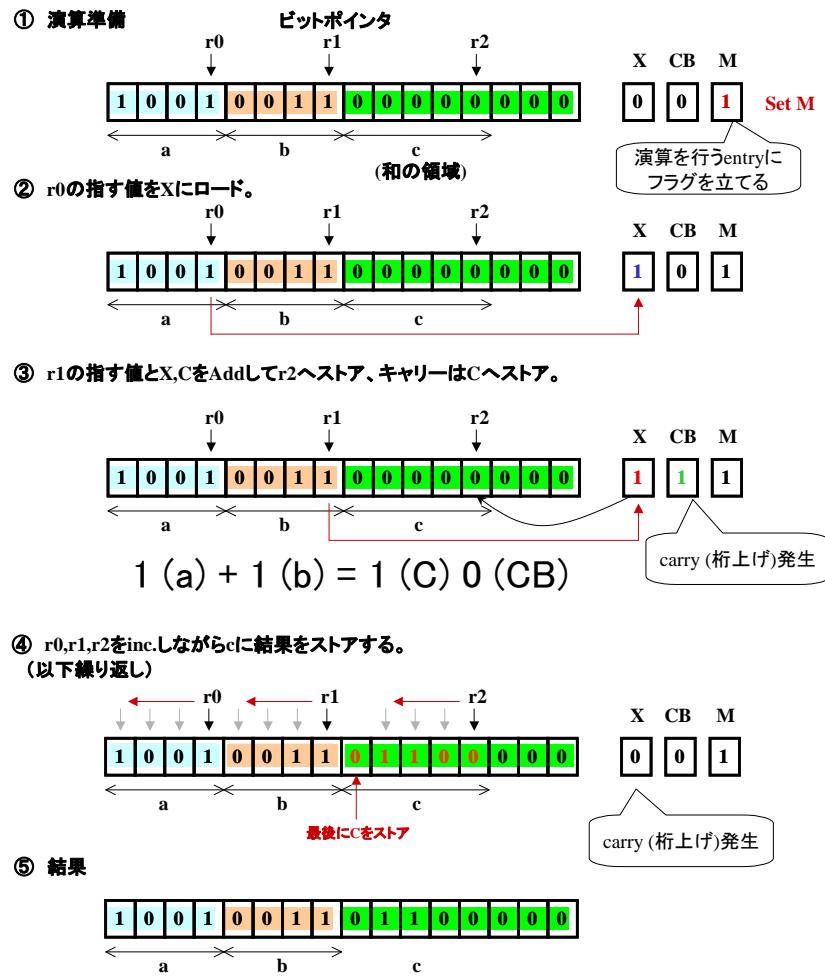


図 4.1: 超並列 SIMD 型プロセッサの加算処理.

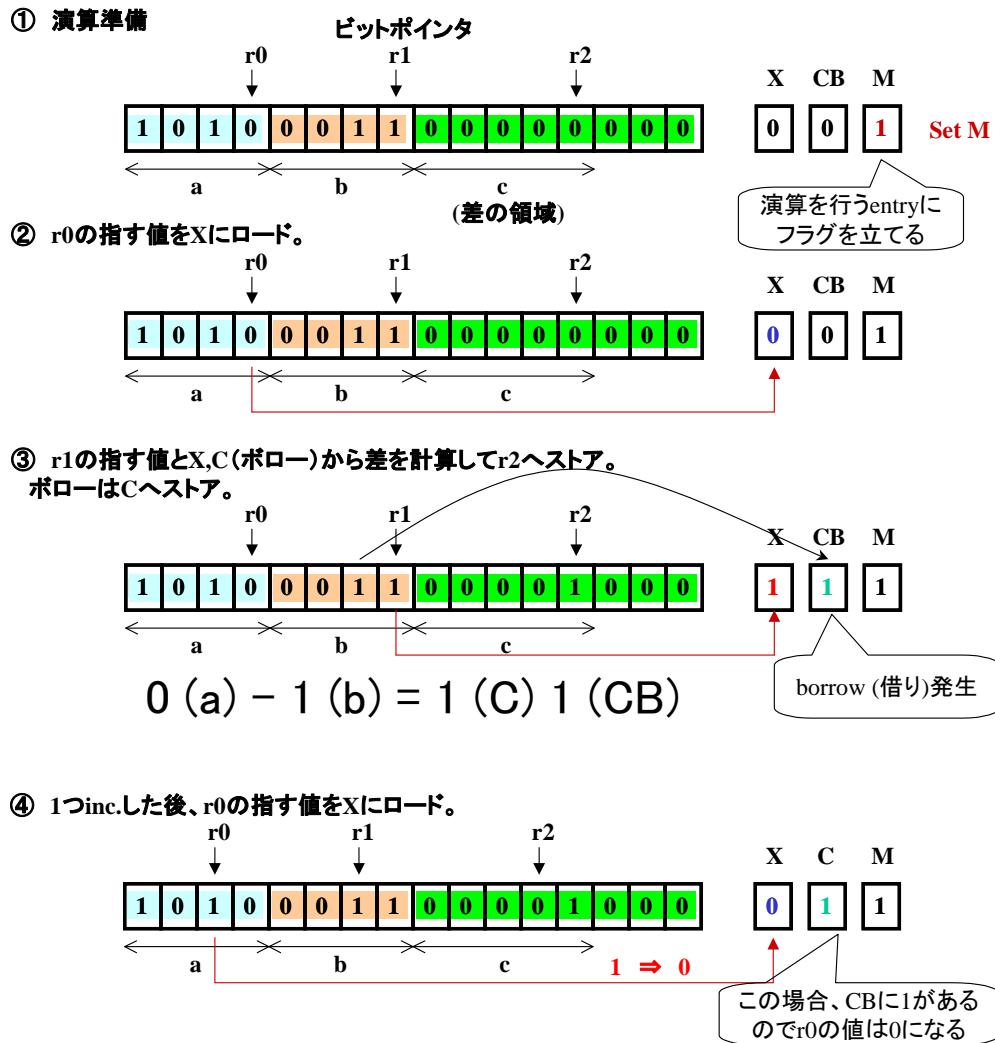
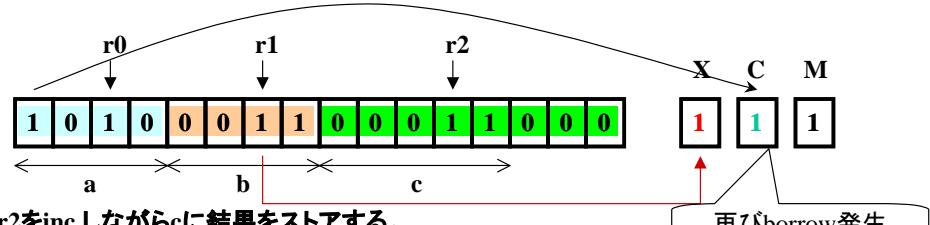
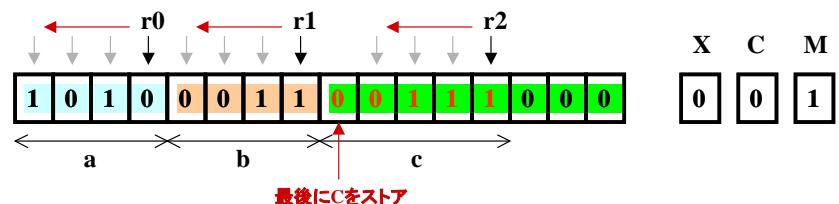


図 4.2: 超並列 SIMD 型プロセッサの減算処理 (1/2).

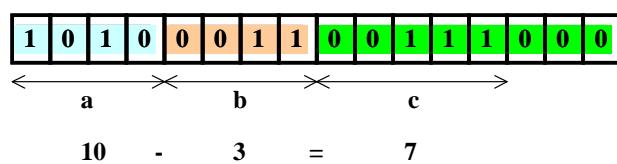
⑤ 1つincした後、r0の指す値をXにロード。



⑥ r0,r1,r2をincしながらcに結果をストアする。
(以下繰り返し)



⑦ 結果



$$10 - 3 = 7$$

図 4.3: 超並列 SIMD 型プロセッサの減算処理 (2/2).

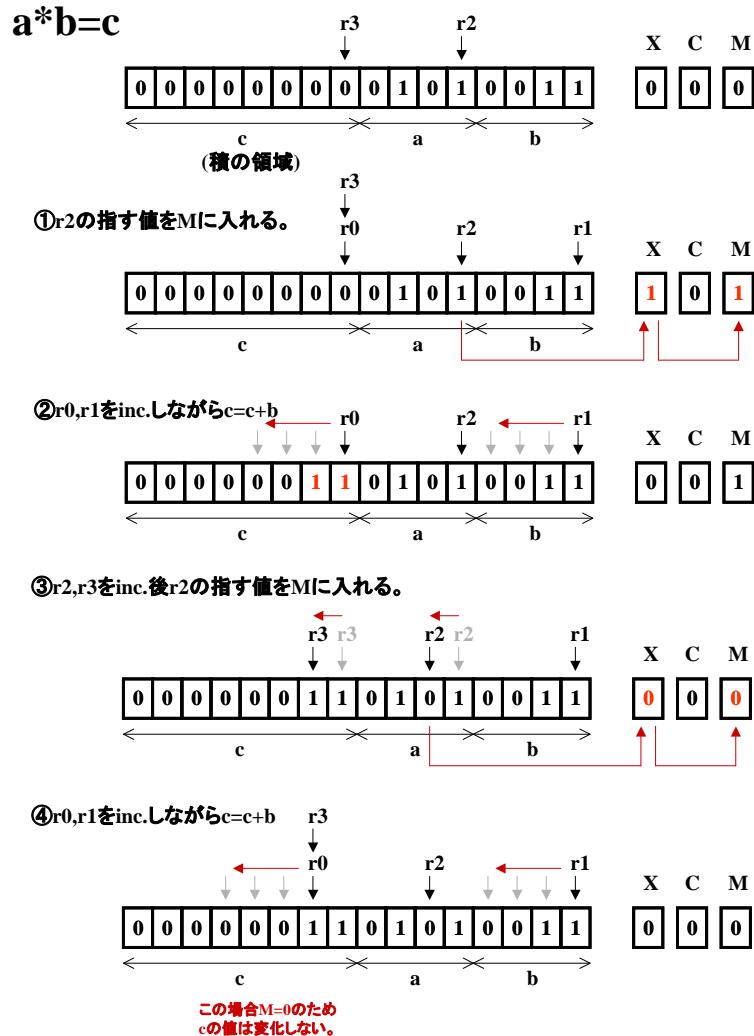
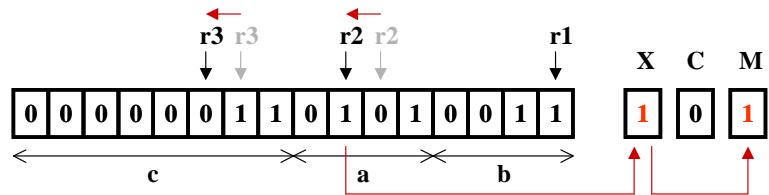
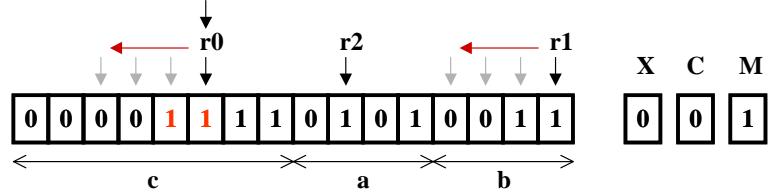


図 4.4: 超並列 SIMD 型プロセッサの乗算処理 (1/2).

⑤r2,r3をinc.後r2の指す値をMに入れる。

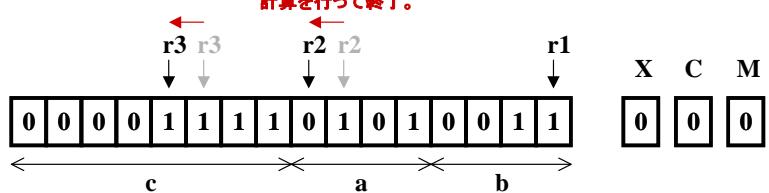


⑥r0,r1をincしながら $c=c+b$ r3



⑦以下同様の繰り返し

ビットポインタが a の範囲の
MSBまで到達した場合の
計算を行って終了。



(結果)

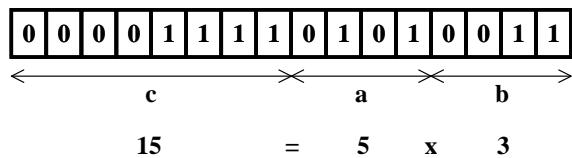
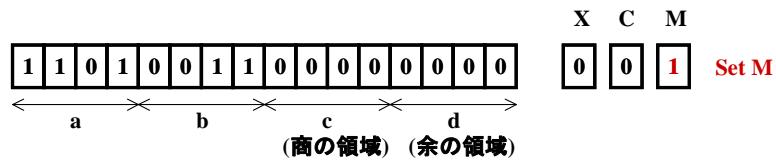
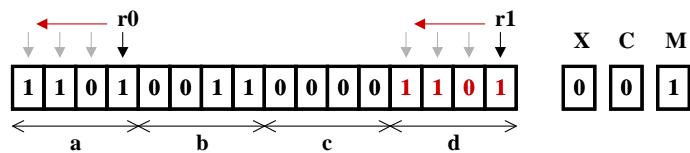


図 4.5: 超並列 SIMD 型プロセッサの乗算処理 (2/2).

a/b=c..d

①aをdにコピー



②(d>b)?をSubにより判定

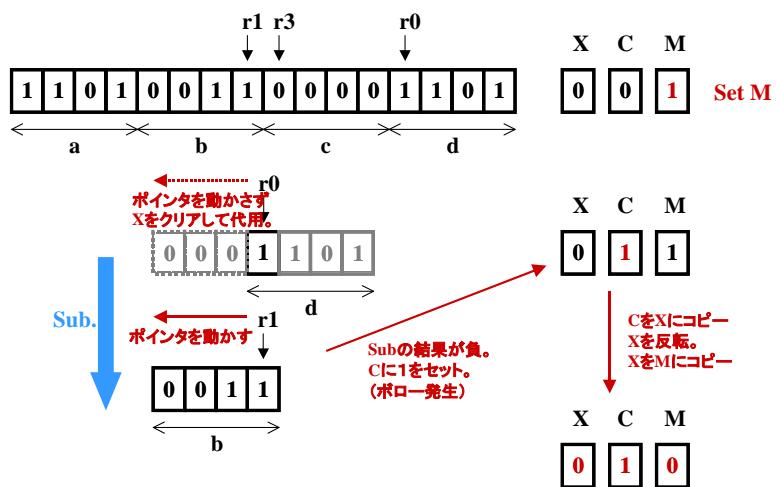
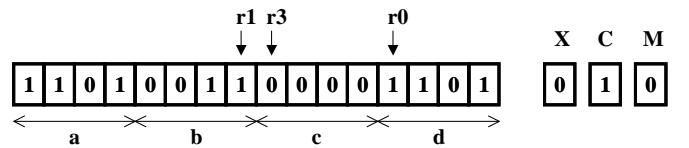


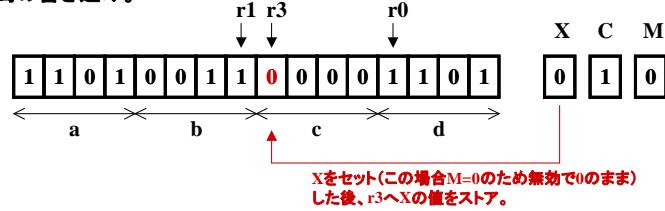
図 4.6: 超並列 SIMD 型プロセッサの除算処理 (1/3).

③d-bを実行して結果をdに書き込み。



この場合M=0のためd-bは実行されない。
⇒ dの値は変化しない。

④cへの商の書き込み。



Xをセット(この場合M=0のため無効で0のまま)
した後、r3へXの値をストップ。

⑤dのピットポインタr0をdec.して(d>b)?をSubにより判定

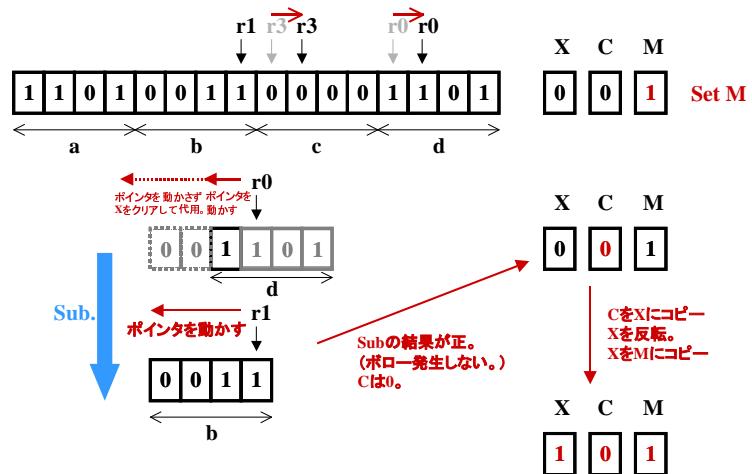
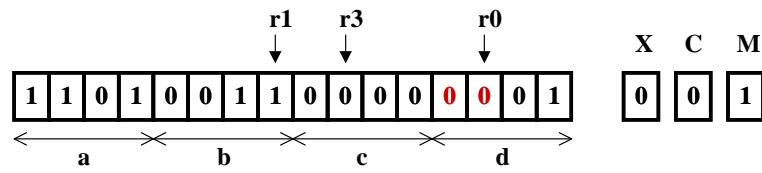


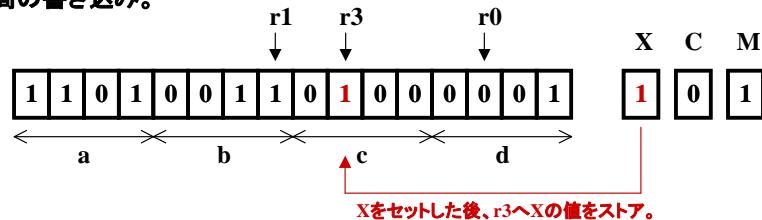
図 4.7: 超並列 SIMD 型プロセッサの除算処理 (2/3).

⑥d-bの結果をdに書き込み。



この場合M=1のためd-bが実行され、結果がストアされる。

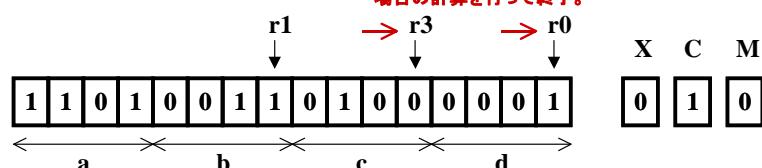
⑦cへの商の書き込み。



Xをセットした後、r3へXの値をストア。

⑧以下同様の繰り返し

ビットポインタがLSBまで到達した場合の計算を行って終了。



(結果)

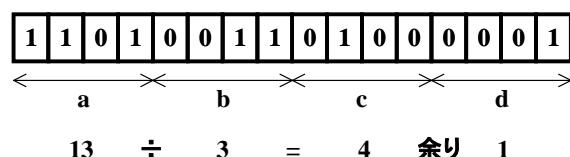


図 4.8: 超並列 SIMD 型プロセッサの除算処理 (3/3).

4.0.4 VLSI 設計結果

超並列 SIMD 型プロセッサを、90 nm CMOS 技術によって共同開発した [54]。開発結果を表 4.1 に、チップ写真を図 4.9 に示す。チップの概要は、2,048 並列の演算器と、500 Kbit (256 bit, 2,048 entry) の SRAM を 2 面備えている。実装面積は 3.1 mm^2 であり、1.2 V 時に 200 MHz で動作する。消費電力は 1.2 V 時、250 mW であり、モバイル機器向けの数値としては十分低いものであるといえる。また、演算能力は 16 bit の加算を行った場合、40 GOPS という高い値を示した。

表 4.1: 90 nm CMOS 技術による、超並列 SIMD 型プロセッサの実装結果。

Process technology	90 nm 7Cu CMOS Low Standby Process
Chip size	3.1 mm^2
SRAM cell size (Single port)	$0.99 \mu\text{m}^2$
Operation frequency	200 MHz (1.2 V時)
Power consumption	250 mW
Maximum performance	40 GOPS

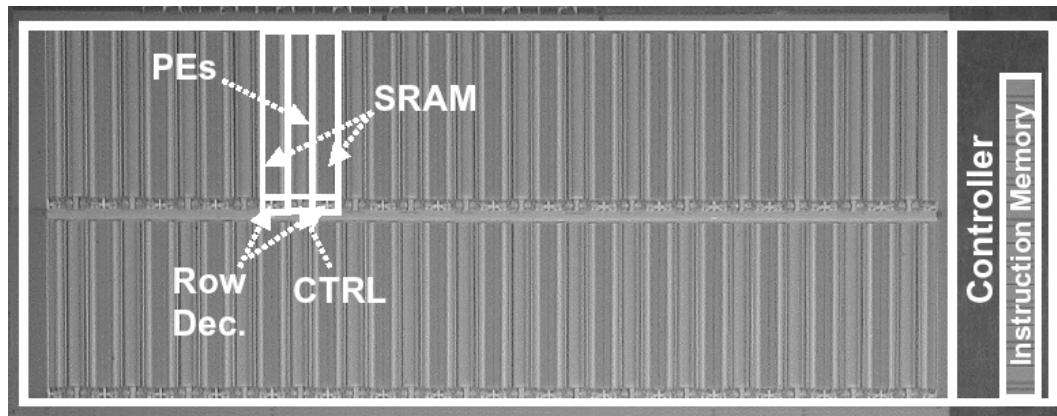


図 4.9: 超並列 SIMD 型プロセッサのチップ写真。

4.1 CAMと超並列 SIMD型プロセッシングアーキテクチャの融合

マルチメディアデータ処理の主要なボトルネックとなっていたテーブルルックアップ処理については、2章、及び3章にて、CAMベースのテーブルルックアップ符号化を適用することで高速かつ高圧縮に処理ができる事を示した。もう一方の処理である繰り返し演算は??節にて、SRAMベースの超並列 SIMD型プロセッサアーキテクチャによって高速に処理できる事を示した。この章では、上記2つのアーキテクチャを融合させて、マルチメディアアプリケーションを高速、高圧縮かつプログラマブルに処理する事のできる、新規性の高い高性能なCAMベースの超並列 SIMD型プロセッサアーキテクチャについて述べる。

4.1.1 融合処理の概要

CAMベーステーブルルックアップ符号化アーキテクチャとSRAMベース超並列 SIMD型プロセッサアーキテクチャを融合するためには、まず、各アーキテクチャのデータ処理手法の違いに着目しなければならない。図 ??に示しているように、CAMベーステーブルルックアップ符号化アーキテクチャのテーブルルックアップ処理は図 ??-(A)の処理であり、超並列 SIMD型プロセッサアーキテクチャによる繰り返し演算は、図 ??-(B)の処理となるからである。そのため、両アーキテクチャが相互にデータを取り取りし、協調動作をするためには、マルチメディアデータの直交変換を行うインターフェースモジュールを中心に両アーキテクチャで挟み込む形にすることが、データ転送の際の消費電力や、配線長に伴う面積及び配線遅延の削減の面で効果的であると考えられる。インターフェースモジュールとSRAMベース超並列 SIMD型アーキテクチャの配置については、図 ??で示した構成を用いる。CAMベーステーブルルックアップ符号化アーキテクチャの融合に関しては、以下に求められる機能と、使用可能なメモリアーキテクチャについて述べた後、構成を決定する。

CAMベーステーブルルックアップ符号化アーキテクチャの融合に関して求められる機能は、以下の通りである。

1. CPUバスから送信されるデータを直交変換し、SIMD型演算モジュールに転送する機能
2. SIMD型演算モジュールから出力されるデータを直交変換し、CPUバスに送信する機能
3. CPUバスから送信されるデータを処理し、再度CPUバスへ送信する機能
4. SIMD型演算モジュールから出力されるデータを処理し、再度SIMD型演算モジュールに出力する機能

5. SIMD 型演算モジュールのボトルネックであるテーブルルックアップ処理を行う機能
6. SIMD 型演算モジュールのボトルネックである一致検索処理を行う機能
7. 汎用性
8. 小面積及び低消費電力

これらの求められる機能を実現するために必要な処理は、大きく分けてデータの直交変換を行う処理と、入力データと同一のデータを符号化テーブルから一致検索する処理である。そのため融合化には、以下に示すメモリアーキテクチャを組み合わせて実現する方法が考えられる。

1. Static Random Access Memory (SRAM)

入力ポート及び出力ポートを一对もつメモリであり、データの書き込み及び読み出しが可能である。

2. 直交 SRAM

水平方向の入力ポート及び出力ポートを一対、垂直方向の入力ポート、及び出力ポートを一対もつメモリである。ブロック図は図 ?? に示してある。SRAM 同様データの書き込み、及び読み出しが可能である。また、水平方向から入力されるデータを垂直方向に出力する、もしくは垂直方向から入力されるデータを水平方向に出力することによってデータの直交変換を可能にすることができる。ただし、直交変換処理を行った場合、数十ビットのワード単位で入力されるデータは、1 bit づつ出力されることとなる。

3. Content Addressable Memory (CAM)

通常の読み書きを行うことのできるメモリに、特定の機能を付加した機能メモリの一種であり、内部に格納されているデータと入力されるデータとの間で一致検索処理を行うことのできるものである。データを格納しているメモリセルの付近に比較器を配置することで、検索処理を実現する構成をとっており、通常この一致検索処理にかかる時間は、他のソフトウェア及びハードウェアより高速である。

4. 直交 CAM

直交 SRAM、及び CAM 両方の機能を有する機能メモリである。ブロック図を図 4.10 に示す。

融合の際には、これらのメモリアーキテクチャを組み合わせて、前述した機能の充足度を評価したうえで、テーブルルックアップインターフェースモジュールを

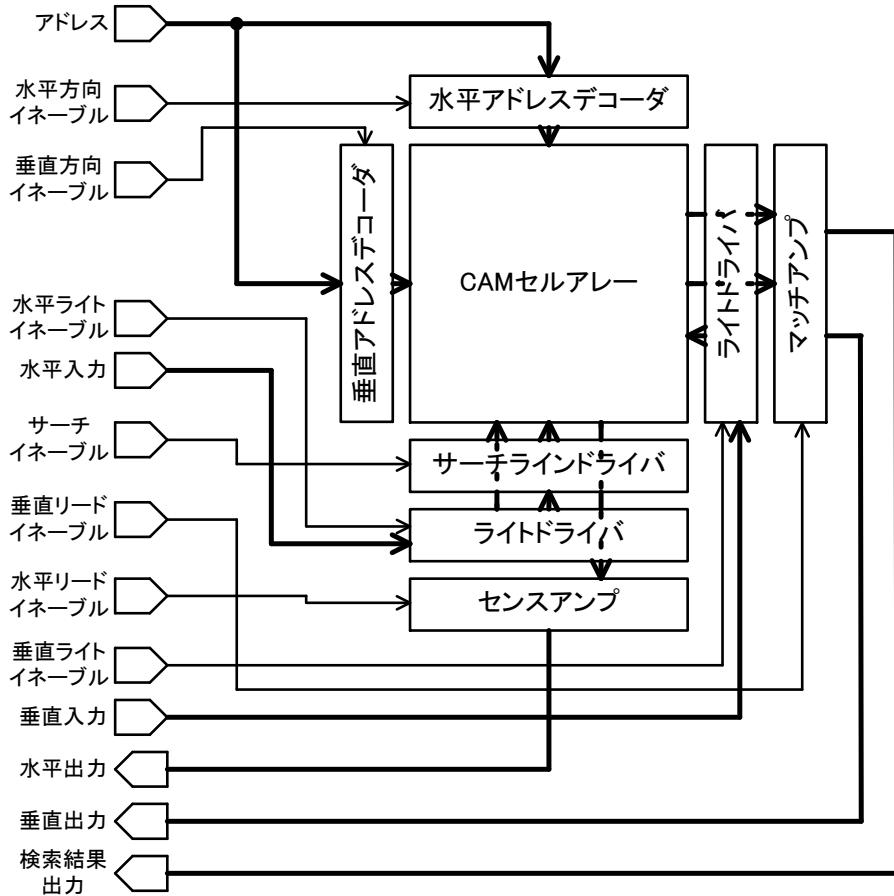


図 4.10: 直交 CAM のブロック図。

実装することとなる。表 4.2 に、検討したメモリアーキテクチャの組み合わせを示す。各組み合わせ案については、インターリーブを動作とテーブルルックアップの両立を踏まえつつ、冗長度が少ないものを選出してある。これらの組み合わせ案を元に、各機能の実現を評価した結果を、表 4.3 に示す。各項目 A (高性能に実現可能), B (実現可能), 及び C (実現は困難) の 3 段階で評価した。ただし、面積の見積もりに関しては組み合わせ 1 の実装面積を 3 として、各組み合わせ案の面積を算出している。検討の結果、機能の実現性、将来性、及び実装面積等で最も有効なのは CAM を直交 SRAM の近傍に配置した構成をとる組み合わせ 3 の案となった。

表 4.2: テーブルルックアップモジュール構築のためのメモリモジュール組み合わせ案.

	SRAM	直交SRAM	CAM	直交CAM
組み合わせ1	-	2	-	-
組み合わせ2	-	-	-	2
組み合わせ3	-	2	2	-
組み合わせ4	2	-	-	2
組み合わせ5	-	-	2	2
組み合わせ6	-	2	-	2

表 4.3: メモリモジュール組み合わせ案に対する機能実現の評価結果.

	機能1	機能2	機能3	機能4	機能5	機能6	機能7	機能8	総合評価
組み合わせ1	A	A	C	C	C	C	C	3	4
組み合わせ2	A	A	A	B	B	A	B	6	3
組み合わせ3	A	A	A	A	A	A	B	6	1
組み合わせ4	A	A	B	B	B	A	B	8	2
組み合わせ5	A	A	A	A	A	A	B	9	3
組み合わせ6	A	A	A	A	A	A	A	9	2

4.1.2 アーキテクチャ

4.1.1節の検討結果より、マルチメディアデータを高速かつ高圧縮に処理できる、CAMベーステーブルルックアップ符号化アーキテクチャとSRAMベース超並列 SIMD型プロセッサアーキテクチャを融合化したアーキテクチャは大きく分けて以下に示す4つの処理回路及び複数個のセレクタから構成されたものとした。

- 直交 SRAM (Orthogonal SIMD) × 2
- 2バンク CAM × 1
- コントローラ (Controller) × 1
- バスインターフェース (Bus interface) × 1

ブロック図を図4.11に示し、アーキテクチャを構成する各回路について述べる。

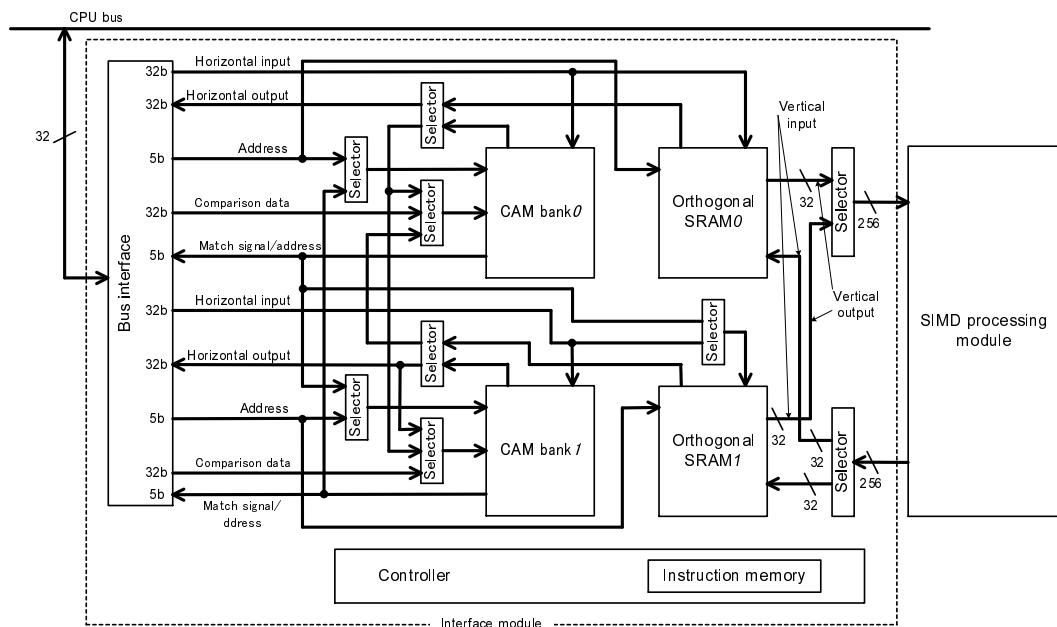


図4.11: テーブルルックアップインターフェースモジュールのブロック図。

直交 SRAM

基本動作及び構成は、??節で述べたものと同様に、直交 SRAM を 2つ用意し、インターリープ動作を実行する。周辺にはCAMに対して、水平に読み出したデータを転送し、検索結果を入力する配線が追加されている。

直交 SRAM を 2つ配置することで、第 1 案と同様のインターリープ直交変換を行うことが可能であり、さらにテーブル検索処理を行う CAM を独立して配置することにより、直交変換とテーブル変換を同時にを行うことを可能とした。

CAM

CAM を用意して、テーブルルックアップ符号化を高速に行う。CAM のサイズはハフマン符号化を主体としたテーブルルックアップ符号化を考慮して $32 \text{ bit} \times 512 \text{ word}$ となっており、高速に一致検索処理を行うことができる。CAM の内部は 2 バンク構成となっており、テーブルルックアップ処理の際には 1 バンクごと独立の処理を行うことも可能である。すなわち、同時に別々の検索データを受け取り一致検索処理が可能であり、一方を CAM として、もう一方を RAM として動作させることができる。従ってテーブルルックアップ処理の際にはバンク 0 に符号化前の全パターンを格納し、CAM として動作させ、バンク 1 にはハフマンコードテーブル等のデータを格納し、RAM として動作させる。また、この時処理される符号後のデータは、直接バスインターフェースを介して CPU へ送信する、もしくは再び直交メモリへ返すことが可能である。

コントローラ

直交 SRAM と CAM の組み合わせは、高速な一致検索能力を利用して、テーブルルックアップ処理をはじめとする様々な処理を行えることが期待できる。そのため、アーキテクチャをプログラマブルな構成とし、各種のアプリケーションを処理できる仕様としたほうが望ましい。従って、コントローラ内部には命令メモリを備え、直交 SRAM 及び CAM を自在に制御できる仕様とした。図 4.12 に、コントローラのブロック図を示す。

コントローラは以下に示す 9 種類の回路から構成される。

- ステートマシン (State machine) × 1

命令メモリから読み出されるデータを解釈し、コントローラ内部のその他の回路を制御する。

- 命令メモリ (Instruction memory) × 1

$32 \text{ bit} \times 256 \text{ word}$ のメモリである。各ワードは、命令セット、デスティネーションアドレス、各種フラグ、及びその他の制御信号から構成されている。

- 32 ビットレジスタ (32 bit register) × 16

ステートマシンが使用するレジスタが用意されている。命令メモリに基づいて各回路、及びモジュールの制御信号やテンポラリデータが書き込まれる。

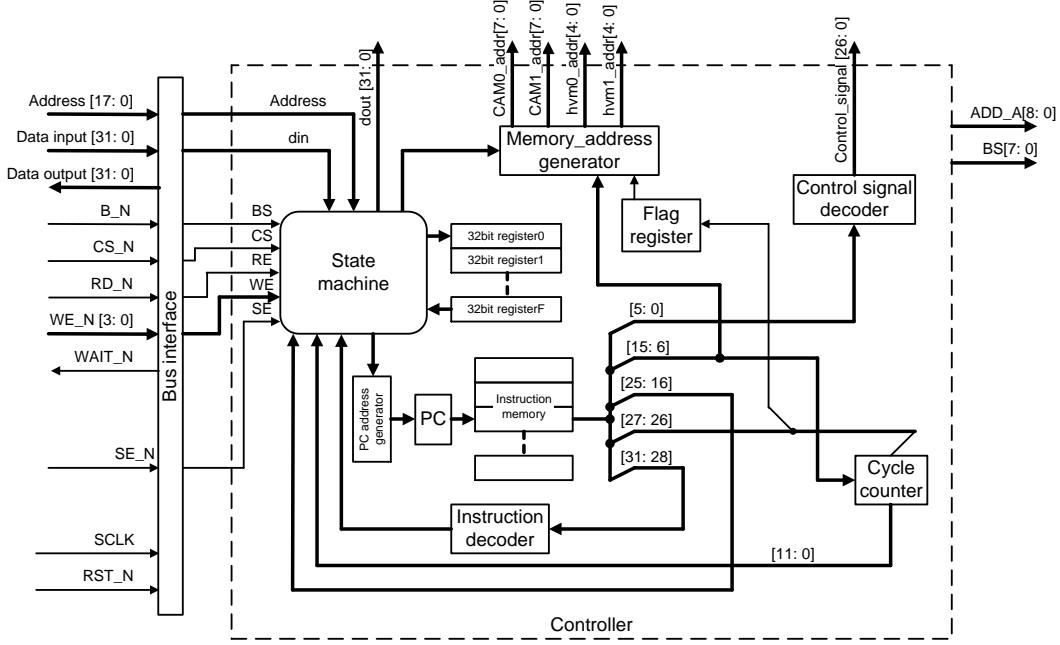


図 4.12: テーブルルックアップインターフェースコントローラブロック図。

- プログラムカウンタ及びプログラムカウンタアドレスジェネレータ (Program counter & Program counter address generator) × 1
プログラムカウンタから逐次アドレスが出力され、命令メモリ内のデータを読み出す。命令メモリから出力されたデータはステートマシンで解釈され、4.1.3 節にて示すジャンプ命令等の場合には、プログラムカウンタアドレスジェネレータがカウンタ値を制御することとなる。
- メモリアドレスジェネレータ (Memory address generator) × 1
ステートマシンから出力されたデータに基づき、直交 SRAM、及び CAM に入力されるアドレスをデコードして出力する。
- フラグレジスタ (Flag register) × 1
メモリアドレスジェネレータ制御用フラグ。
- 命令セットデコーダ (Instruction decoder) × 1
命令メモリから出力されたデータがこのデコーダを介すことにより、命令セットに解釈され、ステートマシンの制御信号となる。
- コントロールシグナルデコーダ (Control signal decoder) × 1
直交 SRAM、及び CAM の周辺回路を制御するための信号を出力する。

- サイクルカウンタ (Cycle counter) × 1

ループ回数等を記憶するカウンタ.

主な入出力ピンの役割は以下の通りである.

- SCLK (入力)

CAM ベース超並列プロセッサのマスタークロック

- RST_N (入力)

CAM ベース超並列プロセッサのグローバルリセット (非同期)

- Address (入力)

CAM ベース超並列 SIMD 型プロセッサのアドレスである。テーブルルックアップインターフェースのコントローラは、内部の CAM、及び直交 SRAM を制御すると同時に、SIMD 型演算モジュール内のコントローラを制御しなければならない。図 4.13 に、CAM ベース超並列 SIMD 型プロセッサ全体のアドレス空間を示す。外部とのアクセスはこのアドレス空間を介して行う。色づけしてある箇所は、テーブルルックアップインターフェースのアドレス空間で、色付けされていない箇所は超並列 SIMD 型演算モジュールのアドレス空間である。

- Data input (入力)

CPU から CAM ベース超並列 SIMD 型プロセッサへの 32 bit データ入力

- Data output (出力)

CAM ベース超並列 SIMD 型プロセッサから CPU への 32 bit データ出力

- CS_N (入力)

CPU が、CAM ベース超並列 SIMD 型プロセッサを選択するための信号

コントローラのアーキテクチャは、命令メモリにプログラムを格納し、取り扱うデータは直交 SRAM 及び CAM に格納される構成とした。これは、ハーバードアーキテクチャの 1 種であり、ファン・ノイマンボトルネックの解消を実現している。

バスインターフェース

基本動作及びその処理は、?? にて述べたものと同様である。

BASE + H'0 0000	連想メモリベース超並列SIMDプロセッサ 制御レジスタ空間
BASE + H'0 7FFF	
BASE + H'0 8000	予約 (1KB)
BASE + H'0 83FF	
BASE + H'0 9000	テーブルルックアップインターフェース 命令メモリ空間 (固定プログラム) (0.5KB)
BASE + H'0 9200	
BASE + H'0 A000	テーブルルックアップインターフェース 命令メモリ空間 (ユーザプログラム領域) (0.5KB)
BASE + H'0 A200	
BASE + H'0 C000	予約 (1KB)
BASE + H'0 C3FF	
BASE + H'0 D000	CAM空間 (2KB)
BASE + H'0 D800	
BASE + H'1 0000	MTCレジスタ空間 (58B)
BASE + H'1 7FFF	
BASE + H'1 8000	SIMD演算モジュール命令メモリ空間 (8KB)
BASE + H'1 FFFF	
BASE + H'2 0000	SIMD演算モジュールデータ空間 (128KB)
BASE + H'3 FFFF	

図 4.13: 超並列 SIMD 型プロセッサのメモリ空間.

4.1.3 命令セット

テーブルルックアップインターフェースモジュールは、内部の命令メモリにプログラムを格納することで、直交 SRAM、及び CAM を用いた様々な処理を行うことができるよう設計した。命令長は、今後の拡張を考慮して冗長度の高い 32 bit としている。図 4.14 に示すのが、1 命令のフィールド構成となる。LSB から、5 bit は直交 SRAM、及び CAM のイネーブル信号を制御するフィールドであり、コントロールシグナルデコーダにて制御信号に変換される。6 bit から 15 bit までは、直交 SRAM、及び CAM に入力するアドレスとなる。16 bit から 25 bit までは、後述する命令セットで使用される、デスティネーションアドレスやループ回数を格納する。26 bit から 27 bit までは、フラグを格納するフィールドである。27 bit から 31 bit までは、命令コードを格納し、命令セットデコーダによって変換されコントローラの動作を決定する。

以下にプログラムを作成するための、基本命令セットを示す。

1. NO OPERATION :

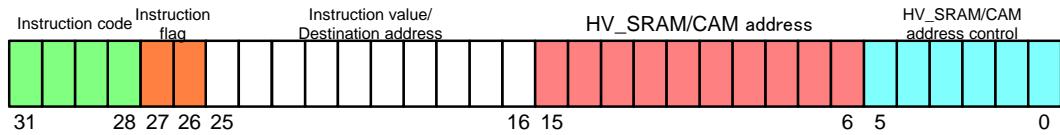


図 4.14: テーブルロックアップインターフェースモジュールの命令フィールド構成.

テーブルロックアップインターフェース制御コードを出力し, PC (Program Counter) を 1 つ進ませる.

2. INITIALIZE

コントローラ内のレジスタを初期化し PC のアドレスを 0 にする.

3. LOOP

この命令が格納されているアドレスの次のアドレスから, Return 命令 (後述) が格納されているアドレスまで指定された回数分処理を繰り返す.

4. RETURN

ループ命令における繰り返し処理の最後に実行する. この命令が実行されるとループ命令が位置しているアドレスの次のアドレスに PC が移動する. また, この際にループ回数が 1 つ減る.

5. JUMP (NO CONDITION)

デスティネーションアドレスフィールドで示されたアドレス (プログラム開始アドレスをベースアドレスとする) に無条件でジャンプする.

6. JUMP (CONDITION)

条件が真である場合, デスティネーションアドレスフィールドで示されたアドレスにジャンプする. 偽である場合はアドレスをインクリメントする.

7. STAY

この命令が格納されている動作を実行し, PC の値は保持する. 主に CPU からの制御信号で使用 (4.1.4 節で示す, スルーモード等).

8. HALT

この命令が実行された場合, プログラムの処理が終了となる.

4.1.4 動作概要

4.1.2 節にて、CAM ベーステーブルルックアップ符号化アーキテクチャをインターフェースモジュールに組み込むアーキテクチャを示した。この章では、テーブルルックアップインターフェース動作の基本となる、データ入出力の手順やテーブルルックアップ符号化処理の手順を示す。

データ入出力

CPU からテーブルルックアップインターフェースを介して、CAM ベース超並列 SIMD 型プロセッサにデータを書き込む、もしくは読み出す場合は、データを直交変換する動作(以下、直交モードと呼ぶ)と、直接書き込む動作(以下、スルーモードと呼ぶ)の 2通りが考えられる。従ってコントローラはそれぞれの場合に応じて直交 SRAM、CAM 及び SIMD 型演算処理部を制御しなければならない。以下に処理の流れを示す。

- 直交モード

これは、通常使用されるモードで、CPU と SIMD 型演算モジュールとのデータの入出力及び、テーブルルックアップインターフェースモジュールを使用してのアプリケーション処理に利用される。処理の流れを、図 4.15 に示す。処理の開始時には CPU から、CAM ベース超並列 SIMD 型プロセッサ制御レジスタ空間にあるコンフィギュレーションレジスタを “0” に設定する。次に、書き込みもしくは読み出し用のアドレス設定レジスタに、処理開始アドレスを書き込む。その後、ライトバッファレジスタカリードバッファレジスタに “1” を書き込むことで、CPU からバースト書き込み、もしくは読み出しが可能となる。

- スルーモード

このモードは、CAM と SIMD 型演算モジュール間で、直交変換を施さずデータを送受信する、及び各種レジスタプログラムの書き込みに使用されるモードである。図 4.15 に示すように、始めにコンフィギュレーションレジスタを “1” に設定する。その後、リードイネーブルもしくはライトイネーブルを制御しアドレスもしくはアドレスと書き込みデータを入力することで、データの読み出し、書き込みが可能となる。

テーブルルックアップ符号化

テーブルルックアップ処理は、符号後のデータを直接 CPU へ送信する処理と再び、SIMD 型演算モジュール内にフィードバックする処理の 2通りが考えられる。以下にそれについて詳述する。

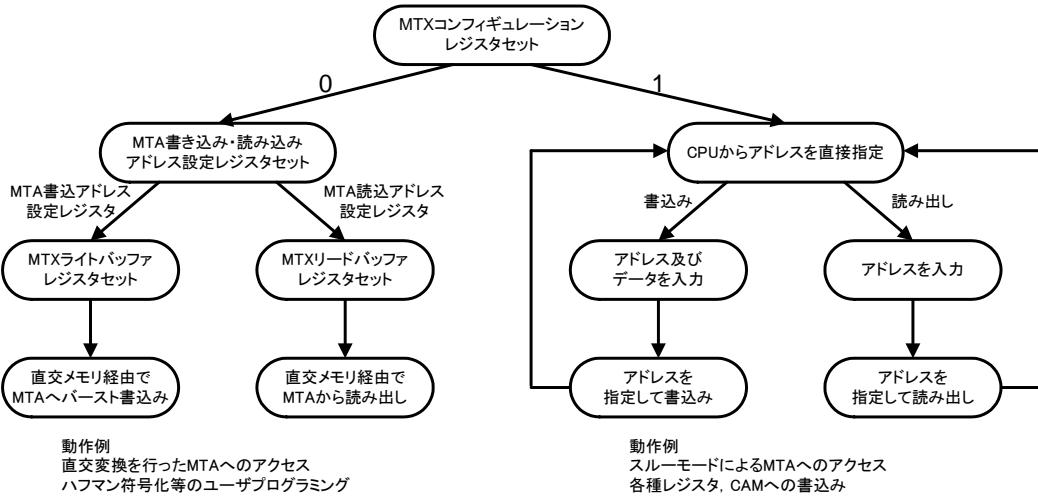


図 4.15: テーブルルックアップインターフェースモジュールを利用したデータ処理フロー。

- ハフマン符号化処理の例

ハフマン符号化は、画像処理のプロセスにおける最後のアルゴリズムとして使用されることが多い。従って、超並列 SIMD 型プロセッサで処理されたマルチメディアデータは、符号化前データとして直交 SRAM を介して CAM へ転送される。その後テーブルルックアップ処理が行われて、CPU へ出力される。具体的な処理は、図 4.16 に示す、①から⑥の順で行われる。直交 SRAM0 及び直交 SRAM1 をデータのインターリープ直交変換に使用し、CAM バンク 0 及び CAM バンク 1 をテーブル検索に利用することによって完全パイプラインのハフマン符号化が実現される。CAM バンク 0 には、入力される符号化前データの全パターンを格納し、CAM バンク 1 には、ハフマン符号化テーブルを格納する。なお、ここで示すハフマン符号化は基礎技術となる静的ハフマン符号化の例であり、リアルタイム符号化テーブル最適化アルゴリズムを適用していない。

- ① 超並列 SIMD 型プロセッサから、1 bit 每データを読み出し直交 SRAM0 に垂直入力する
- ② ①で格納されたデータを、アドレスの降順に水平読み出しし、読み出されたデータを CAM バンク 0 に検索データとして入力する。
- ③ CAM バンク 0 には、あらかじめハフマンコードが格納されているため、入力される検索データと一致するハフマンコードが検索され、そのアドレスが出力される。
- ④ ③で出力されたアドレスを CAM バンク 1 に、水平方向のアドレスと

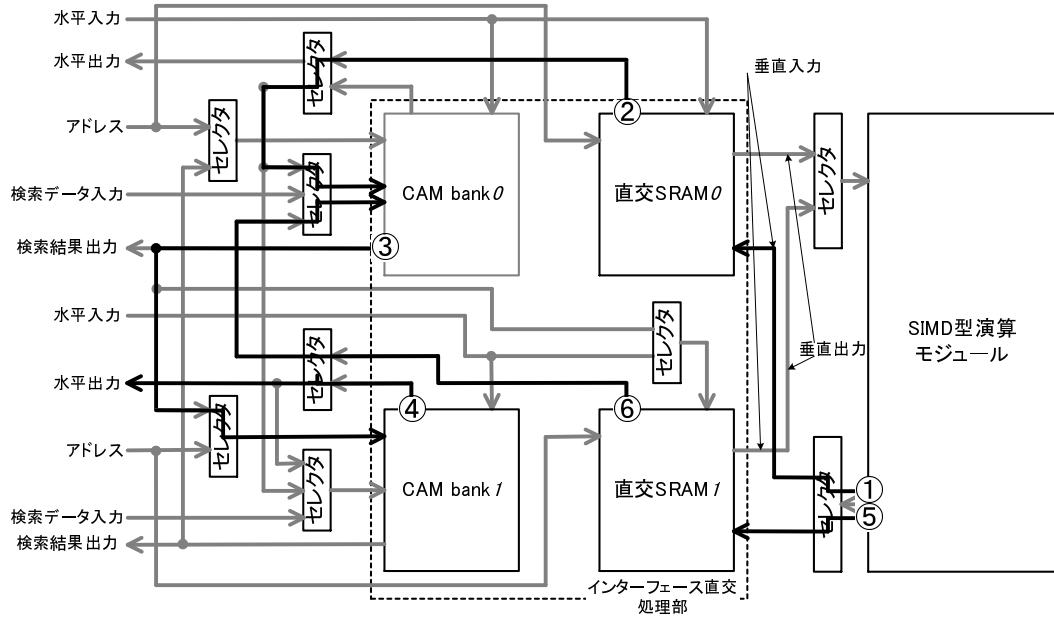


図 4.16: インタフェースモジュールによるハフマン符号化の例.

して入力する。CAM バンク 1 には、あらかじめ符号後のデータが CAM バンク 0 のハフマンコードとアドレスで対応付けされて格納されており、その出力から、符号後データを得る。

- ⑤ 直交 SRAM0 が符号前データを CAM バンク 0 に出力している間に、直交 SRAM1 には超並列 SIMD 型プロセッサから 1 bit 每データが垂直入力される。
- ⑥ ②と同様の操作で直交 SRAM1 から、CAM バンク 0 に検索データを出力する。その後は③と同様のプロセスを経て符号後のデータが出力される。なお、この間直交 SRAM0 には、①の処理が行われることとなる。

- 暗号処理の例

前述の例では、CAM による一致検索機能をテーブルルックアップ処理の一種であるハフマン符号化処理に応用し、その処理結果を CPU へ直接出力する方法について述べた。一般に、マルチメディアアプリケーションは、ハフマン符号化の様に、CAM の処理結果を直接 CPU へ出力するものと、アルゴリズムの途中にテーブルルックアップ符号化を行い、その結果を再度超並列 SIMD 型プロセッサへフィードバックするものがある。この処理で代表的なものは、暗号化があり AES アルゴリズムは、繰り返し演算処理の間に Sub byte 変換というテーブルルックアップ処理が行われる。具体的な処理は、図 4.17 に示す、①から④の順で行われる。

- ① 超並列 SIMD 型プロセッサから、1 bit 每データを読み出し、直交 SRAM0 に垂直入力する。
- ② ①で格納されたデータを、アドレスの降順に水平読み出しし、読み出されたデータを CAM バンク 0 に検索データとして入力する。
- ③ CAM バンク 0 に格納されているデータと検索処理が行われ、入力される検索データと一致した場合、そのアドレスが出力されることとなる。出力されたアドレスは、直交 SRAM1 に、水平入力のデータとして入力される。
- ④ 直交 SRAM1 の全アドレスに、③の処理で入力されたデータが格納されたならば、超並列 SIMD 型プロセッサへ、1 bit 每データを出力する。

なお、上記の処理において、CAM バンク 0 は一致検索処理を行ったが、単なるメモリとして機能させても良い。

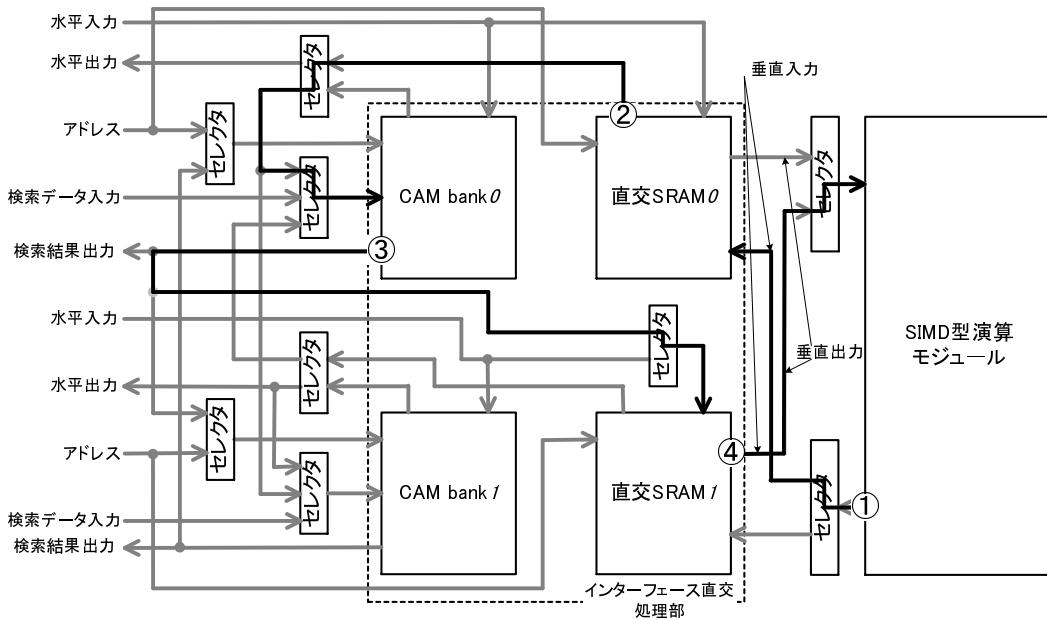


図 4.17: インタフェースモジュールによるフィードバック処理の例。

4.1.5 VLSI 実装結果

CAM ベーステーブルルックアップ符号化アーキテクチャを融合させた、超並列 SIMD 型プロセッサのインターフェースモジュールを、90 nm CMOS 技術によって開発した。アーキテクチャはハードウェア技術言語 (HDL: Hardware Description Language) の一種である、Verilog-HDL を用いて開発した。表 4.4 に、ソフトマク

口での論理合成結果を示す。使用ツールは、シノプシス社の Design Compiler (バージョン 2005.09) を使用した。動作周波数の制約は、超並列 SIMD 型プロセッサの動作周波数に合わせるために 200 MHz とした。また、最小面積を目標として合成されるように制約を施している。論理合成後の、全体面積は約 0.75 mm² となった。直交 SRAM は 32 bit × 32 word とサイズが小さいため、約 0.03 mm² と、最も小さかった。次いでコントローラが約 0.21 mm² というサイズであり、最も大きい 32 bit × 256 word の CAM サイズ約 0.24 mm² と近い値になったが、これは、命令メモリサイズ (32 bit × 256 word) を、大きめに備えているためである。

ここで直交 SRAM と CAM をフルカスタムで設計した場合の実装に関して面積の見積もりを算出する。ベースとなる Verilog ソースに、フルカスタムによるメモリ類を実装した場合の論理合成見積もりを表 4.5 に示す。直交 SRAM は図 4.18 に示す直交 SRAM セルのレイアウト、及び図 4.19 に示す直交 SRAM セルアレイを元に面積を算出した。一般的にメモリモジュールのセル以外の周辺回路はセルエリアの約 50%程度と見積もることができるため。算出した結果、約 0.01 mm² と見積もることができた。また、CAM に関しては直交 SRAM セルのサイズと CAM セルのサイズがほぼ 1 対 1 であることが分かっているので、面積を約 0.08 mm² と見積もった。その結果、全体の面積は約 0.39 mm² となり、ソフトマクロ実装の場合の約 2 分の 1 となった。なお、命令メモリをフルカスタム化することで、更なる面積の削減を実現することが可能である。超並列 SIMD 型プロセッサの面積が、3.1 mm² であるので CAM ベース超並列 SIMD 型プロセッサは、約 3.49 mm² となった。一般にモバイル機器向けの SoC のコアが約 1 cm² であることを考慮すると、十分実装できる大きさであることがわかる。

表 4.4: ソフトマクロによる、テーブルルックアップインターフェースモジュールの実装結果。

Process technology	90nm 7Cu CMOS Low Standby Process
Total area size (*)	0.75 mm ²
- Horizontal-Vertical SRAM	0.03 mm ²
- CAM	0.24 mm ²
- Controller & etc	0.21 mm ²
Operation frequency	200 MHz

(*) 直交SRAM2つとCAM2つを含む

表 4.5: ハードマクロによる、テーブルルックアップインターフェースモジュールの実装結果。

Process technology	90nm 7Cu CMOS Low Standby Process
Total area size	0.39 mm^2
- Horizontal-Vertical SRAM	0.01 mm^2
- CAM	0.08 mm^2
- Controller & etc	0.21 mm^2
Operation frequency	200 MHz

(*) 直交SRAM2つとCAM2つを含む

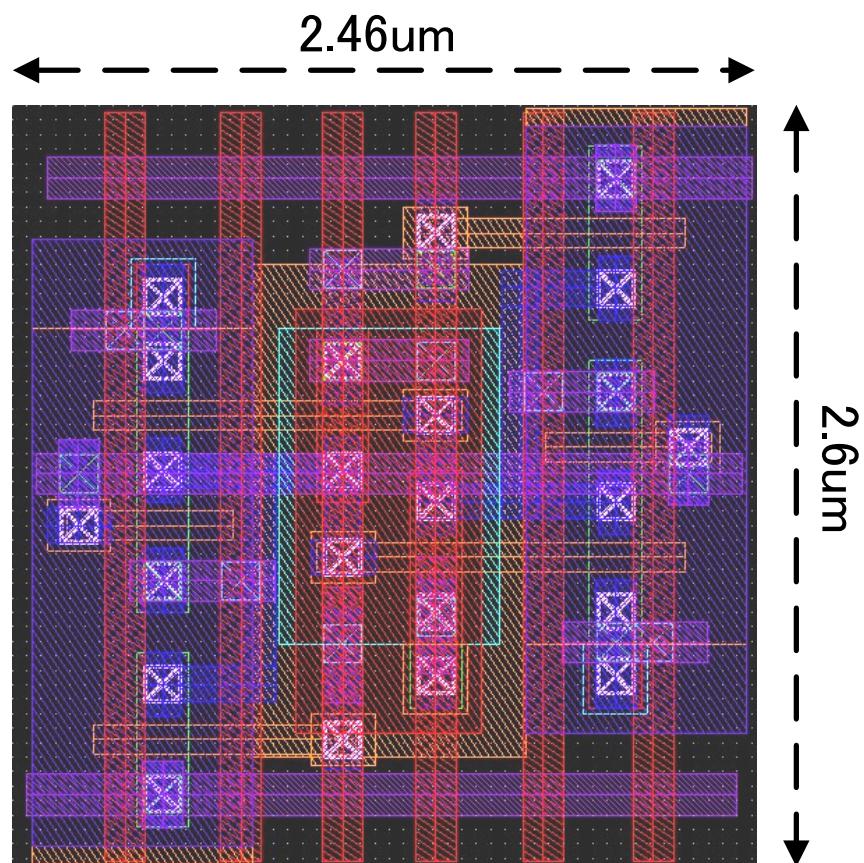


図 4.18: 直交 SRAM セルのレイアウト。

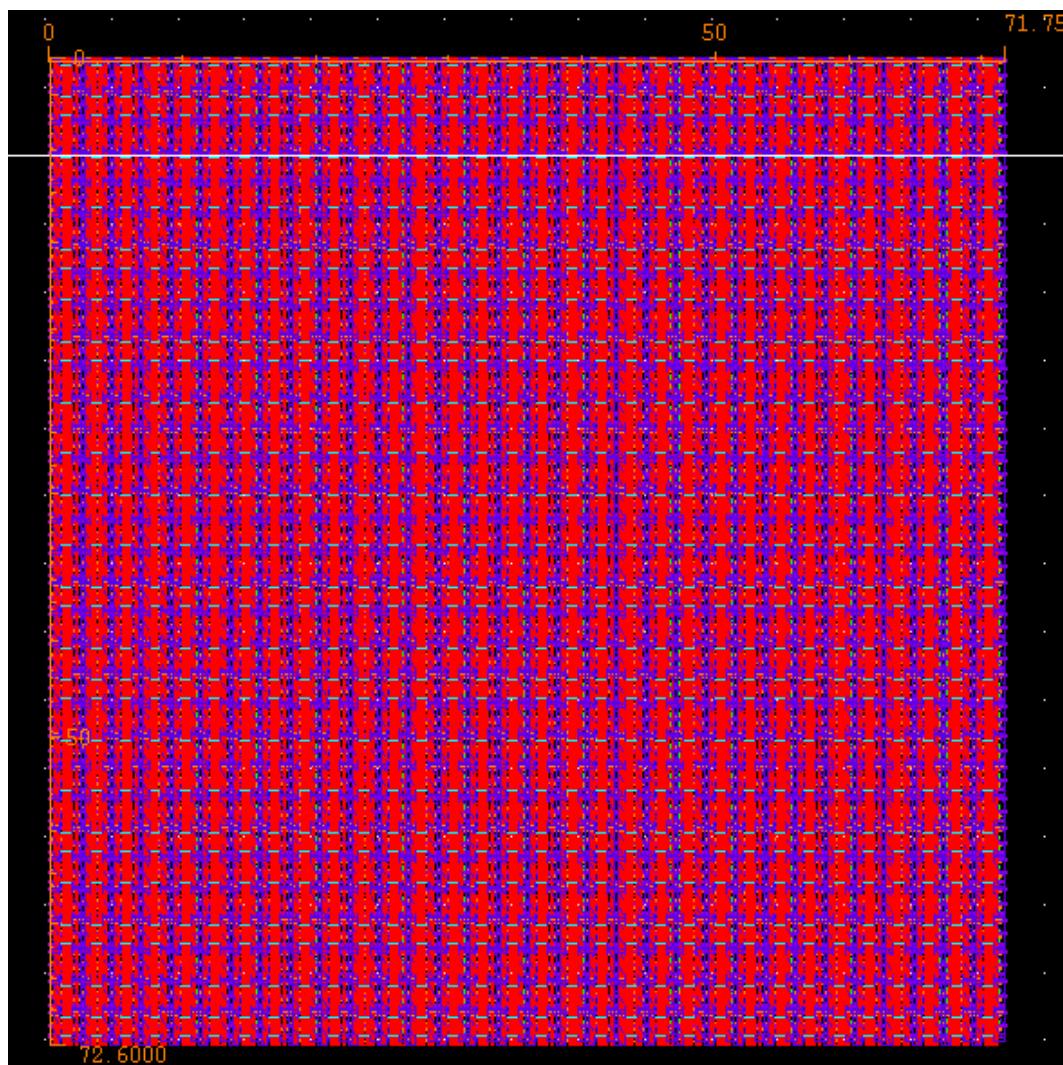


図 4.19: 32 bit × 32 word の直交 SRAM セルアレイ.

4.2 画像処理アルゴリズムへの適用

この節では、CAM ベース超並列 SIMD 型プロセッサを用いて、マルチメディアデータ処理を行い、性能を評価する。性能評価に用いるマルチメディアアプリケーションは、静止画像圧縮技術の 1 つであり、カラー画像の国際標準方式に制定されている、JPEG (Joint Photographic Experts Group) アルゴリズムを用いる。

4.2.1 JPEG 方式の概要

JPEG 方式 (以下、JPEG と呼ぶ) は、デジタルカメラ等のデジタル家電や携帯電話等のモバイル機器で幅広く用いられており、繰り返し演算やテーブルルックアップ符号化を含むマルチメディアアプリケーションである。JPEG は、図 4.20 に示すように基本方式 (Baseline system)，拡張方式 (Extended system)，及び DPCM (Differential Pulse Code Modulation) と呼ばれる差分パルス符号化方式に分けることができる。一般のモバイル機器や、デジタル家電で多く用いられているのは基本方式であり、DCT (Discrete Cosine Transform : 離散コサイン変換) をベースにした DCT 方式を利用して、高い圧縮率を実現する。ただし、情報の一部が省略され、復号時には非可逆となるため、完全な画像の再現はできない。これに対して、オプション機能として用意されているのが、拡張方式と DPCM 方式である。拡張方式は、算術符号化 [55] 等を用いて、高精彩画像、及び高能率符号化を実現するものである。DPCM 方式は、画像の可逆方式を実現する、すなわちデータの欠損がない復号処理を行うアルゴリズムである。これは、DCT を用いず、空間関数アルゴリズムを用いることによって実現され、主に医療診断用等の特殊な用途で使用される。今回の評価では、CAM ベース超並列 SIMD 型プロセッサのマルチメディアデータへの適用を踏まえ、最も一般的な基本方式に基づいて性能評価を行う。図 4.21 に、JPEG の基本処理フローを示す。以下の節では、マルチメディアデータ処理 LSI にとって処理の負担が大きい、DCT とハフマン符号化を中心として各処理の説明及び CAM ベース超並列 SIMD 型プロセッサアーキテクチャによる処理の概要について述べる。

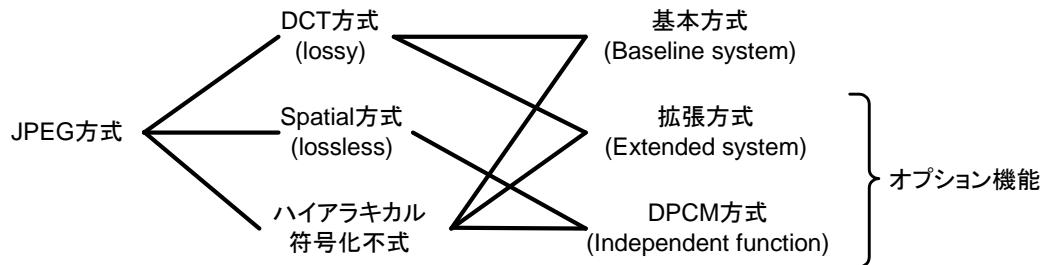


図 4.20: JPEG 方式の分類図。

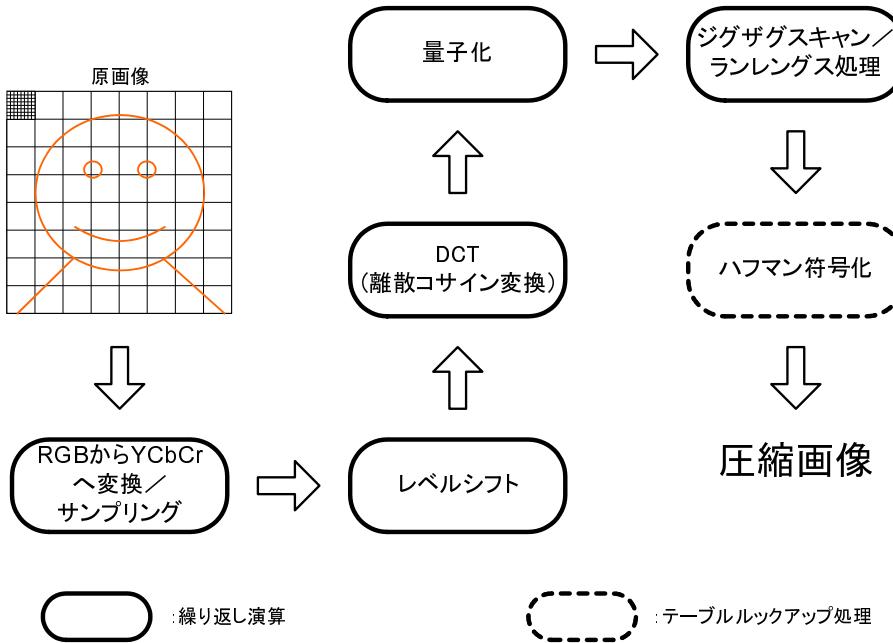


図 4.21: JPEG 基本方式の処理フロー.

4.2.2 DCT (Discrete Cosine Transform)

DCT (Discrete Cosine Transform) は、離散コサイン変換とも呼ばれ、画像内の各画素をコサイン波で表される周波数成分に変換して、各画素値を各周波数成分にかかる係数 (DCT 係数値) で表す座標変換処理である。

ここで、今後の性能評価に關係のある、画像と周波数の関係について図 4.22 を例として述べることにする。この例では簡単のために、画像は白か黒どちらかのみ使用している。図中の上部にある画像を周波数 (空間周波数と呼ぶ) に変換すると、下部のようになる。縦軸は色の濃淡を表し、黒に近いほど値は小さくなり、白に近いほど値は大きくなる。また横軸は距離を表す。従って、左の周波数は色の変化が少ないため低い周波数となり、右の周波数は色の変化が激しく、高い周波数となる。音が正弦波の重ねあわせで表現できるように、画像も基本周波数の重ねあわせで表現できることになる。

DCT 処理は、 $N_x \times N_y$ (N_x, N_y は、x, y 方向の画素数) の画素値の集合を対象として演算が行われる。この集合に施した結果は、低周波数成分が左上に、高周波数成分が右下まとめられ、各画素値は DCT 係数値で表される。図 4.23 に、x 及び y が 8 の場合の DCT の基本パターン群を示す。これらのパターンが画像の中にどの程度含まれているかが DCT の結果表される。その後、各画素の冗長な成分が量子化によって圧縮される。

DCT 処理は、 $N_x \times N_y$ の画素値の集合全てに演算を施す必要があるため、JPEG に含まれるアルゴリズムの中でも、マルチメディアデータ処理 LSI による処理時

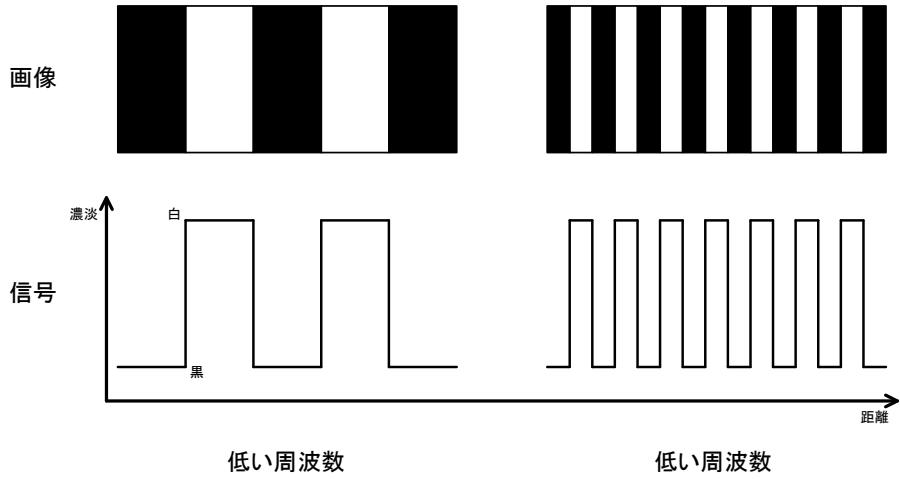


図 4.22: 画像と周波数.

間が大きく、少なくとも全体の処理時間の 26%を超えることが多い [1], [28]. そのため、マルチメディアデータ処理 LSI の処理性能は、テーブルルックアップ処理と共に DCT の処理時間によっても決まることがある.

これまでの研究により、マルチメディアデータ処理 LSI が高速に DCT を処理するためには、以下に示す 3 つの要点を満たす必要があることが分かっており、それぞれについて CAM ベース超並列 SIMD 型プロセッサによる処理方針を述べる.

1. 高速 DCT の適用

一般に、 $N_x \times N_y$ の集合には、1 回の処理で x 方向と y 方向に演算を施さなければならない. しかしながら、演算自体が複雑な上に、データ間の移動が多く、ハードウェア実装向けのアルゴリズムとはいい難い. そのため高速化のために式を変形して、 x 方向及び y 方向に別々に分解し、対象データに順番に施すことによって 1 次元 DCT として高速化を実現することが可能である. 特に、 $N_x=8$, $N_y=8$ のとき、積和演算の回数を削減することが分かっており、1 次元 DCT が高速に行えることが分かっている. これは高速 DCT 処理と呼ばれ、ハードウェア実装向けのアルゴリズムとして盛んに研究が行われている [56], [57]. 高速 DCT は、データフローで処理の手法を視覚的に表すことが可能であり、図 4.24 に、CAM ベース超並列 SIMD 型プロセッサ向けに最適化した高速 DCT 処理のデータフローを示す. CAM ベース超並列 SIMD 型プロセッサは、データの移動に垂直チャネル、及び水平チャネルを用いる. 垂直チャネルの移動度は、2 の累乗であることは、??節にて述べた. そのため、図 4.24 のフローには各ステップに様々なデータの移動量が含まれているのだが、STEP2 や STEP4 にあるように移動距離が近いものをまとめ、処理の効率化を図った. また、STEP3 や STEP5 では、演算をまとめて SIMD 処理の際に、動作しないエントリが極力無いように工夫した.

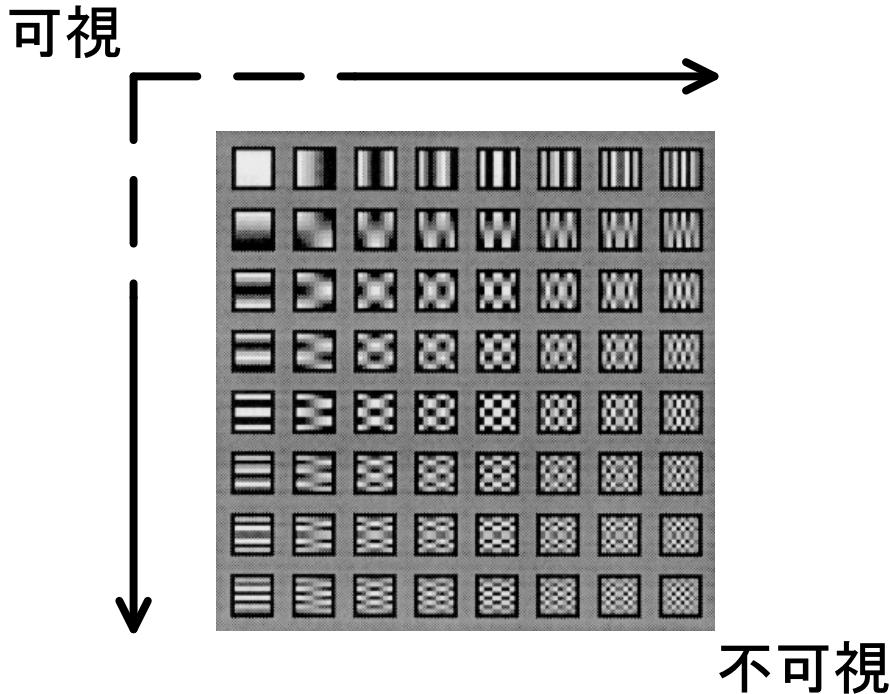


図 4.23: DCT 基本パターン群.

CAM ベース超並列 SIMD 型プロセッサは、この高速 DCT 処理を、図 4.25 に示すように、垂直方向及び水平方向に順番に施す。水平方向の処理は $6 \text{ ステップ} \times 8 = 48 \text{ ステップ}$ で、水平方向に関しては、各 PE が垂直に配列してあるため、6 ステップで完了できる。

2. 並列処理の適用

CAM ベース超並列 SIMD 型プロセッサは、図 4.25 に示すように、 $N_x=8$, $N_y=8$ の画素の集合を、原画像の構成のまま SRAM へ格納することが可能である。JPEG は、処理画像を $N_x=8$, $N_y=8$ を 1 つの単位として、処理するため、一般的のマルチメディアプロセッサは、図 ?? に示すようにデータを分割して処理する。これに対し CAM ベース超並列 SIMD 型プロセッサは、垂直に配置することで、高い並列処理を実現することが可能である。CAM ベース超並列 SIMD 型プロセッサの並列度が 2,048 である場合には、256 並列に処理を行うことが可能である。

3. データ処理幅の変動に対する柔軟性

DCT 処理が行われるデータは、通常 8 bit から始まる。このデータにコサインの乗算を繰り返すことで、データ長は、8 bit, 16 bit, 24 bit, 32 bit, … と増加する。通常のマルチメディアデータ処理プロセッサは、図 ?? に示すよ

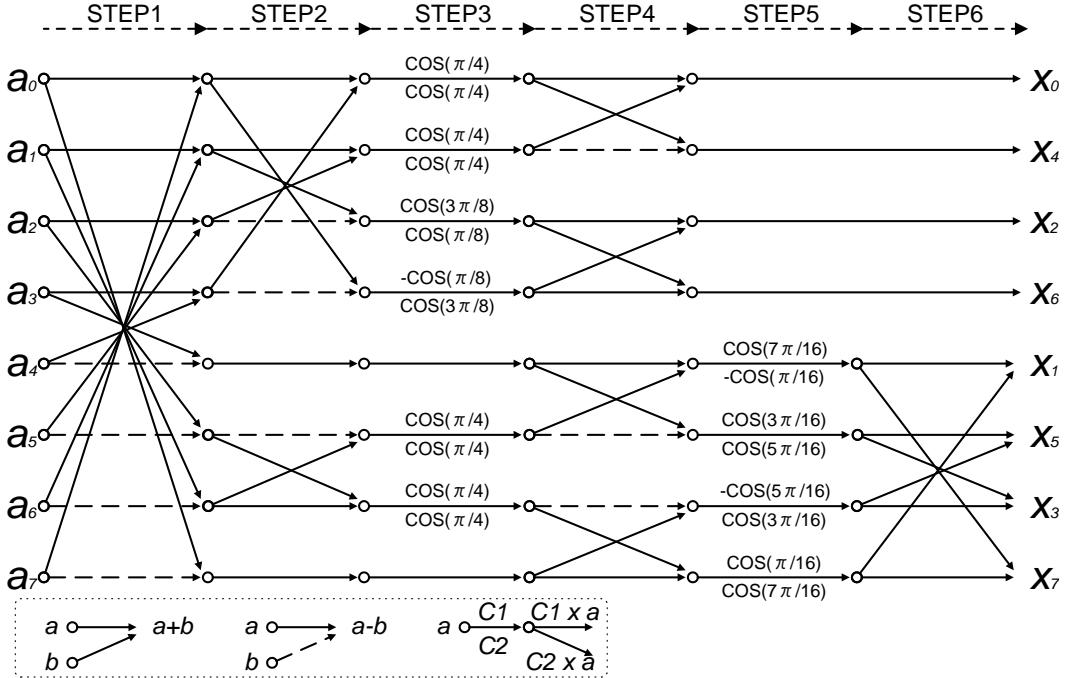


図 4.24: CAM ベース超並列 SIMD 型プロセッサによる、高速 DCT データフロー。

うな並列処理の手法をとっているため、データ長の変動は直接並列度の低下を招くこととなる。しかしながら、超並列 SIMD 型プロセッサの構成は、水平方向に 256 bit 以上と十分な長さを備えており、データの処理方向がビットシリアルであるため並列度の低下は無い。

以上の検討より、CAM ベース超並列 SIMD 型プロセッサによる DCT 処理を詳述する。

垂直方向高速 DCT 処理

CAM ベース超並列 SIMD 型プロセッサによる、垂直方向高速 DCT 処理の流れを述べる。高速 DCT 処理は、図 4.25-(a) に示すデータフローによって行われ、この処理を 8 回 (8 画素列分) 行う。ここでは、図 4.24 に示すデータフロー中の STEP2 及び STEP3 について図 4.26 を用いて説明する。

STEP2 は、2種類のバタフライ演算によって処理される。上方への移動を正、下方への移動を負とすると各画素 (a_0, a_1, \dots, a_7) の移動度は、 ± 1 移動、 ± 3 移動から構成される。前述したように垂直チャネルの移動量は 2 の累乗であるから、SIMD 型アーキテクチャを生かすために、この移動量をうまく組み合わせて共通に移動するデータを多くすることで処理回数を削減できる。例えば a_0 の場合には、バリッドフラグを用いてマスク処理を行い、始めに -4 移動する。その後は $+1$ 移

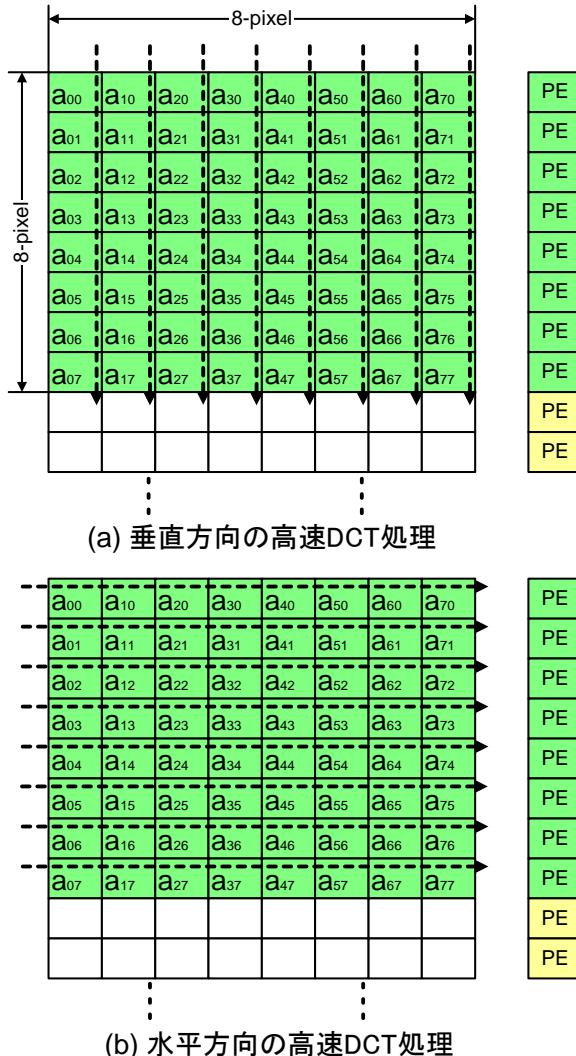


図 4.25: CAM ベース超並列 SIMD 型プロセッサによる、高速 DCT 処理手法.

動が共通である a_0 , a_2 及び a_6 を移動し、減算することによって a_0-a_3 , a_6-a_5 を同時に処理している。STEP3では、乗算が主体となるので、データの移動後、各エントリにコサイン値を一斉に乗じることによって処理の共通化を図っている。なお、コサインの定数値やテンポラリの領域はポインタの指定で任意に扱うことが可能である。

水平方向高速 DCT 処理

CAM ベース超並列 SIMD 型プロセッサによる、水平方向高速 DCT 処理の流れを述べる。水平方向の処理は、前述した垂直方向高速 DCT 処理後のデータに対して行われる。高速 DCT 処理は、図 4.25-(b) に示すデータフローによって行われ

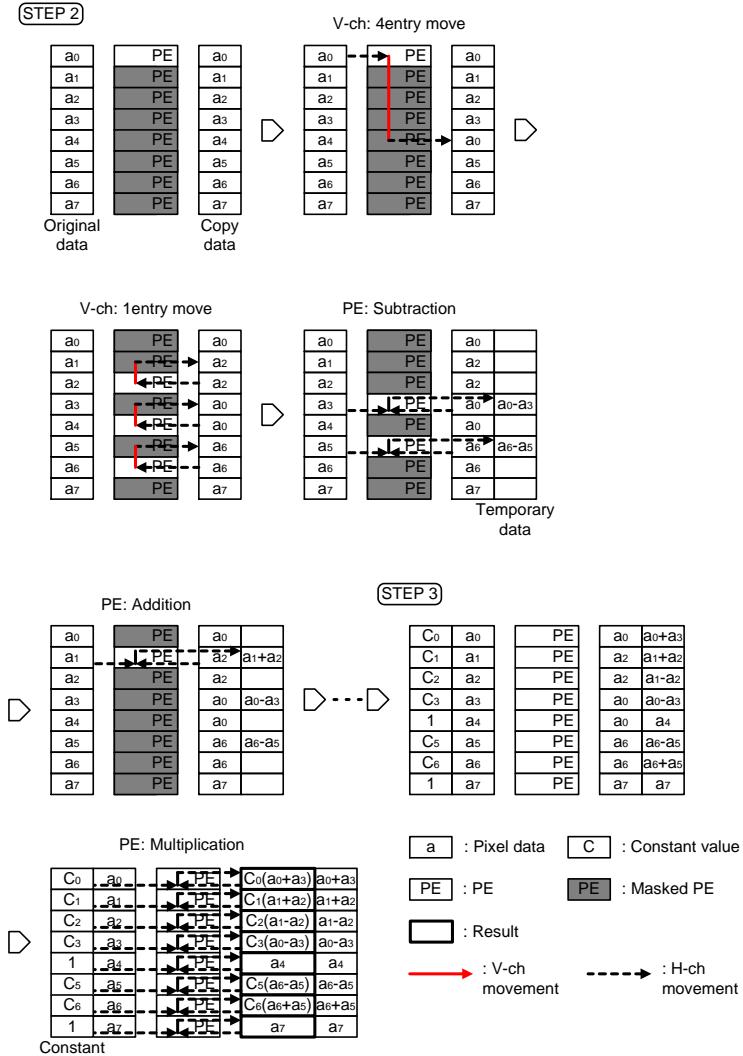


図 4.26: CAM ベース超並列 SIMD 型プロセッサによる垂直方向高速 DCT 处理.

る。なお、水平方向の処理では全画素行に対しデータの全並列に処理を行うことができるためデータフローは1回のみ行えばよい。ここでは、データフロー中の STEP4 及び STEP5 について図 4.27 を用いて説明する。

STEP4 は、1種類のバタフライ演算が各エントリで行われる。水平方向の処理は水平チャネルのみ用いるため、移動度の制限は無い。そのためデータフローで示されている、加算もしくは減算を順次行うだけでよい。例では、STEP4-2 で、 a_0 と a_1 の加算を行い、STEP4-3 で a_0 と a_1 の減算を行っている。なお、必要に応じてテンポラリの領域を使用する。STEP5 に関しても、STEP4 と同様に水平チャネルのみを行い、別領域に格納されているコサイン値と乗算を行うことで実現可能である。

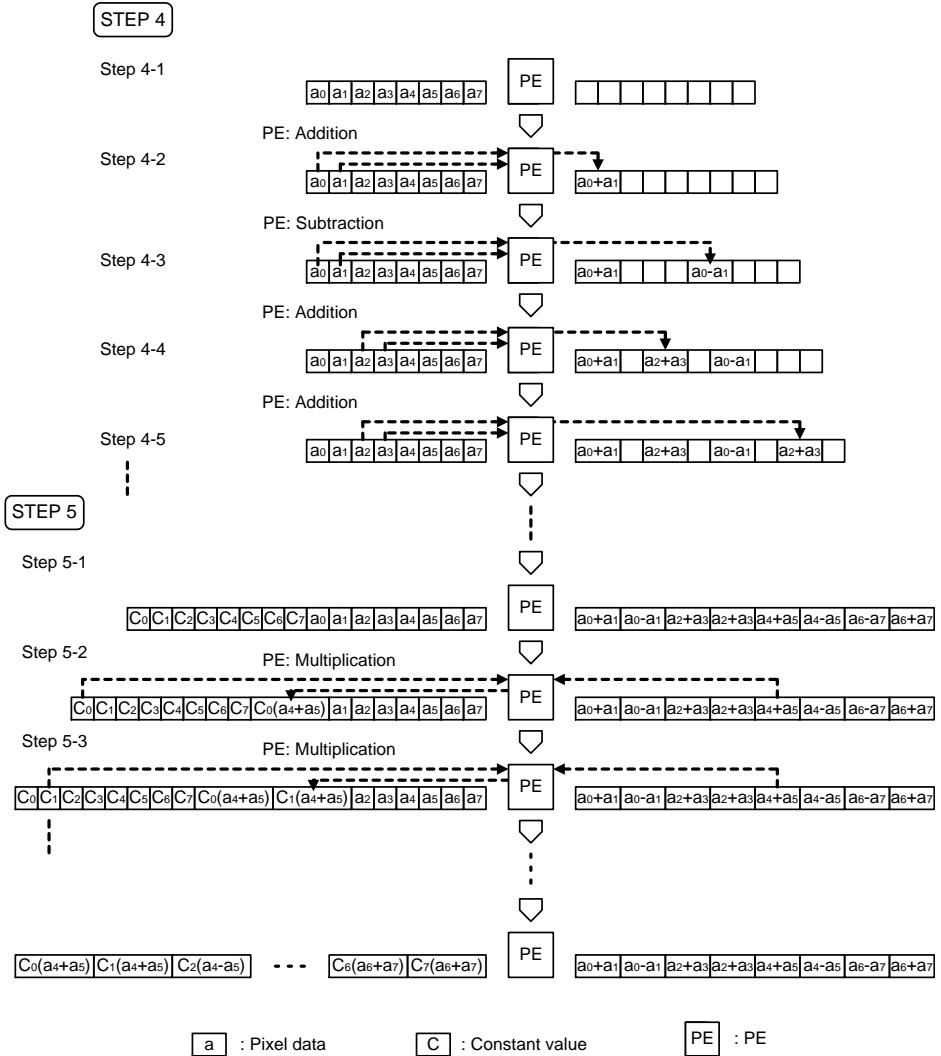


図 4.27: CAM ベース超並列 SIMD 型プロセッサによる水平方向高速 DCT 処理.

DCT 処理の評価

CAM ベース超並列 SIMD 型プロセッサによる、DCT 処理の性能評価について述べる。DCT は、コサインの乗算によってデータ長が変化するため、性能評価には $512 \text{ word} \times 1,024 \text{ entry}$ の SRAM を 2 面用意して評価を行う。比較対象のプロセッサは、90 nm CMOS テクノロジで製作された、2 命令同時発行 VLIW (Very Long Instruction Word) アーキテクチャ 16 bit の DSP [51]–[53] を選択した。動作周波数はどちらも 200 MHz である。図 4.28 に処理クロックサイクル数のグラフを示す。横軸は処理クロック数である。処理画像の総数は、1,024 entry に対して高速 DCT における 1 つのブロックが、 $N_y=8$ ピクセルであることから、128 ブロックとしている。なお、画像の解像度の変化に伴い、CAM ベース超並列 SIMD 型プロセッサ、及び DSP 共に、総クロックサイクル数は線形で増加する。グラフよ

り、CAMベース超並列 SIMD 型プロセッサは、DSP と比べてクロックサイクル数を約 87% 削減することが示せた。この結果を元に、スループット (MOPS: Mega Operation Per Second) と実装面積で単位面積当たりの処理能力を評価する。表 4.6 に算出結果を示す。両アーキテクチャとも 200 MHz で動作することから、スループットは約 8 倍の性能向上となった。また、単位面積あたりの処理能力についても、5.6 倍となった。

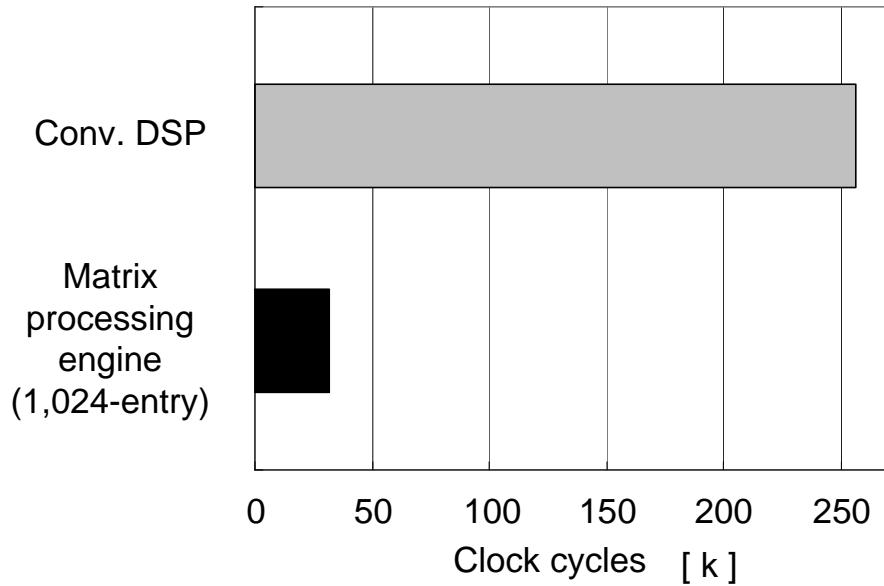


図 4.28: CAM ベース超並列 SIMD 型プロセッサ及び DSP による JPEG 処理クロックサイクル数の比較。

表 4.6: 単位面積当たりの処理性能の比較。

	MOPS	MOPS/mm ²
Conv. DSP	6.40 (1)	2.94 (1)
Matrix processing engine (1,024-entry)	51.24 (8.0)	16.53 (5.6)

スループットの算出結果に関しては、CAMベース超並列 SIMD 型プロセッサが 128 並列に画像ブロックを高速 DCT にて処理できることから差がついたものと考えられる。単位面積辺りの処理能力に関しては、SIMD 型演算モジュールが SRAM

第4章

データレジスタバンクと演算器を密結合していること、及び集積度の高いメモリをベースにしていることに起因していると考えられる。

4.2.3 ハフマン符号化

この節では、CAM ベース超並列 SIMD 型プロセッサによるハフマン符号化について述べる。ハフマン符号化は図 4.21 に示した通り、ランレングス処理を経た、符号化前データがハフマンコードへ変換されることでデータ長が圧縮されるアルゴリズムである（アルゴリズムの詳細は、2.2 節を参照）。この処理は、JPEG 方式の最後の処理であり、全体の処理の 30% を占めている [1], [28]。従って、このアルゴリズムを高速に処理することは画像処理全体の処理時間を大きく縮めることに直結し、マルチメディア処理 LSI のパフォーマンスを大きく左右する。

CAM ベース超並列 SIMD 型プロセッサは、4.1.4 節に示した、CAM を用いた高速なテーブルルックアップパイプライン処理によって、高速処理を実現できる。

テーブルルックアップインターフェースによるハフマン符号化

テーブルルックアップインターフェースは、直交 SRAM、及び CAM を用いてプログラマブルにデータの処理が行えるようなコントローラを有している。図 4.29 に、テーブルルックアップインターフェースによるハフマン符号化のフローチャートを示し、その概要を説明する。フローチャート前半の流れは、始めにコンフィギュレーションレジスタを “1” へセットすることにより、CPU からのデータ入力に対してスルー モードへ設定することから始まる。その後、CPU から、アドレス及びデータを逐次的に入力し、CAM バンク 0 に符号化前データの全パターンを、CAM バンク 1 にハフマンコードテーブルを格納する。ここまでが、符号化のための準備となる。次に、コンフィギュレーションレジスタを “0” へセットした後に、バースト書き込みのアドレスを指定して、SIMD 型演算モジュールへ、圧縮対象となる画像データを書き込む。SIMD 型演算モジュールのエントリ分画像データを格納したならば、SIMD 型演算モジュール内で、図 4.21 に示した、R, G 及び B から、Y, C_b 及び C_r への変換処理からランレングス処理まで繰り返し演算処理を行う。その後、SIMD 型演算モジュール内の符号化前データを読み出しつつ、ユーザプログラムを命令メモリから読み出し、直交 SRAM、及び CAM を制御して、4.1.4 節にて述べた処理によりハフマン符号化を実行する。CAM ベース超並列 SIMD 型プロセッサは、CAM にハフマンコードテーブルを格納した後の処理を、1,024、もしくは 2,048 entry 単位で実行でき、これを画素数分行うことで、JPEG 処理が完了する。

信頼性評価

テーブルルックアップインターフェースモジュールの開発については、4.1.5 節にて述べた。このモジュールは、ユーザが作成したプログラムを、コントローラ内の命令メモリに格納し、SIMD 型演算部からのデータをコントローラ、直交 SRAM

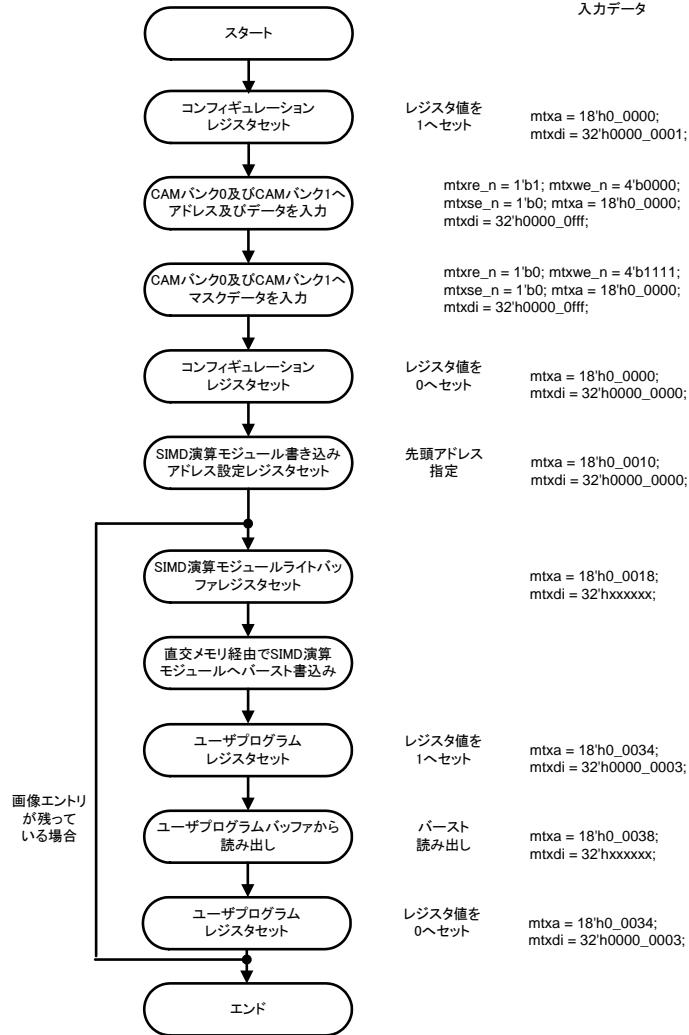


図 4.29: CAM ベース超並列 SIMD 型プロセッサによるハフマン符号化処理.

及びCAMによって処理する。そのため、性能を評価する前に、正確に各モジュールが協調動作しているかどうかの信頼性を検証する必要がある。ここでは、テーブルルックアップインターフェースモジュールにて処理したデータと、あらかじめソフトウェアで処理したデータを比較することにより、アーキテクチャの信頼性を検証する。信頼性の検証はテーブルルックアップインターフェースと、SIMD型演算モジュールのデータレジスタバンクを用意して行った。

検証環境は、Xilinx社のISE Foundation 7.1iを用いて論理合成及び配置配線し、Mentor Graphics社のModelSimSE 5.8Cを使用してタイミング(遅延)シミュレーションを行った。タイミングシミュレーションにてハフマン符号化処理を実施することにより、ハードウェア実装時の遅延を考慮した環境を模擬した。処理対象画像は結果が偏るのを防ぐため、サンプルデータは、カラー、モノクロ、人物、動

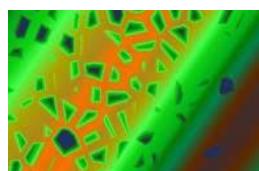
物，及び背景等，様々な被写体を選択した。サンプルデータを図 4.30，図 4.31，図 4.32，図 4.33，図 4.34 に示す。これらは， 192×128 から， $1,500 \times 1,125$ までの解像度である画像 25 枚から構成される。次に，信頼性検証の手順を図 4.35 に，その説明を以下に示す。

- ① 検証用ビットマップ画像を，2,048 entry 単位に分割し，テーブルルックアップインターフェースモジュールに入力。
- ② テーブルルックアップインターフェースモジュール内で直交 SRAM が，インターリープ動作をしながら連続して SIMD 型演算モジュールへデータを転送。
- ③ SIMD 型演算モジュール全エントリにデータが入力されたならば，再びテーブルルックアップインターフェースモジュールにデータを送信する（本来は，この間にハフマン符号化以外の処理が行われる）。
- ④ 直交 SRAM がインターリープ動作をしつつ，CAM にデータを送信。CAM によりテーブルルックアップ動作が行われ，ハフマンコードが生成される。
- ⑤ ハフマンコードが符号後データとして，CAM ベース超並列 SIMD 型プロセッサより出力されテキストデータとして保存される。
- ⑥ あらかじめ用意していた，ソフトウェアの結果と比較する。

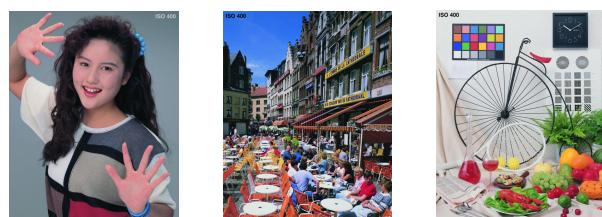
全ての画像についてデータの整合性を確認した結果，一致率は 100% となり，テーブルルックアップインターフェースアーキテクチャの信頼性が確認できた。

また，この検証と共に処理クロックサイクル数の測定も行ったが，この評価を含めた処理性能の評価は，4.2.4 節にて，JPEG に含まれる他のアルゴリズムとあわせて行うこととする。

192 × 128



480 × 600



512 × 480



図 4.30: 192×128 , 480×600 及び 512×480 の検証用画像.



図 4.31: 256×256 の検証用画像.

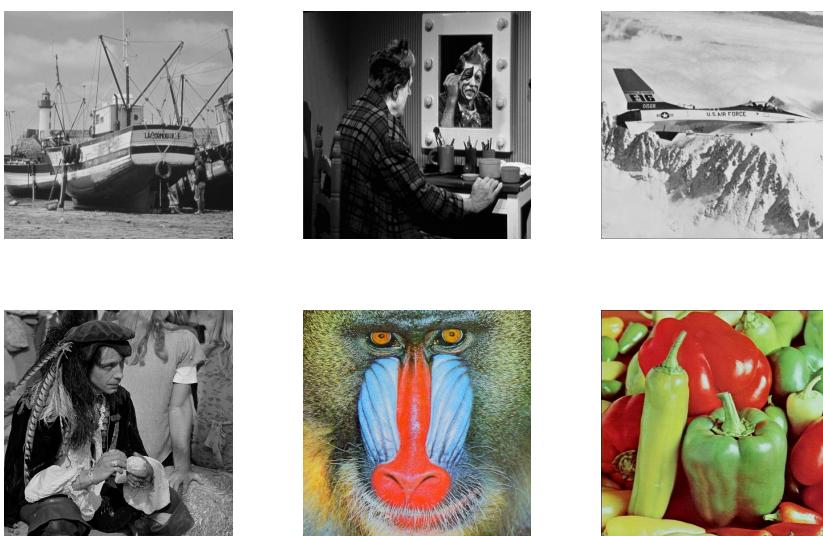


図 4.32: 512×512 の検証用画像.



図 4.33: 600×480 の検証用画像。

720 × 576



800 × 600



1,024 × 768



1,500 × 1,125

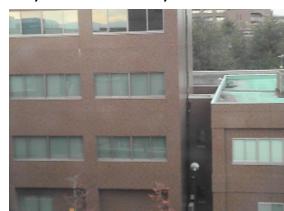


図 4.34: 720 × 576, 1,024 × 768 及び 1,500 × 1,125 の検証用画像.

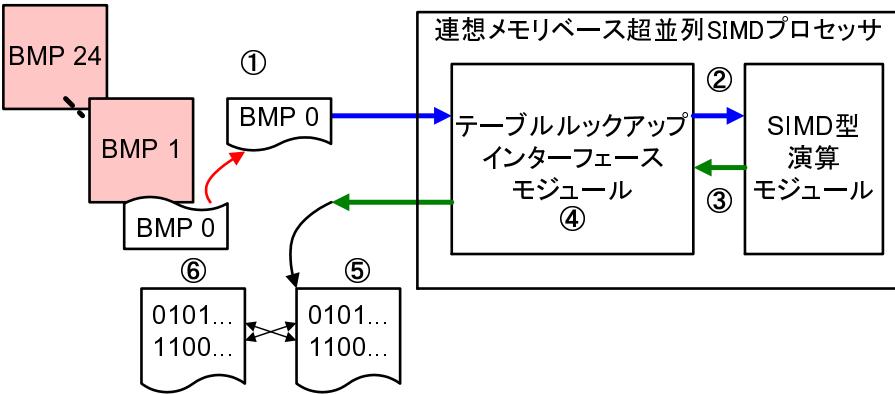


図 4.35: ハフマン符号化処理の信頼性検証手順.

4.2.4 性能評価

この節では、CAM ベース超並列 SIMD 型プロセッサによる各アルゴリズムの処理方法を元にして、実画像を処理した結果を示す。また、その結果を元に既存の DSP と比較することで性能評価を行う。

はじめに、ビットマップ画像から JPEG 画像へと変換した際の、総処理クロック数について比較する。対象画像は、4.2.3にて検証用に用いた JPEG 画像である。比較対象アーキテクチャは、SRAM ベース超並列 SIMD 型プロセッサと一般的な 2 命令同時発行 VLIW (Very Long Instruction Word) アーキテクチャである 16 bit の DSP [51]–[53] である。処理にかかった総クロックサイクル数を表 4.7 に示す。今回は、圧縮にオプションを指定しない通常の JPEG アルゴリズムを使用している。また、R, G 及び B から Y, C_b 及び C_r を生成する際に、サンプリングを考慮しないようにするために Y 画像のみを取り扱った。そのため、画像の視覚的特長が異なる場合でも、処理データ自体は異なるが、ピクセル数が同一であればクロックサイクル数は同じ値となる。比較の結果、DSP は DCT 処理やハフマン符号化において並列度が低く、テーブルルックアップ処理が逐次的な処理のため、どの画像においても、最も大きい値となった。SRAM ベースの超並列 SIMD 型プロセッサは、DCT 等の処理を高速に行えるため、DSP に比べて処理クロックサイクル数は少ないことが分かった。これらの結果に対して、SRAM ベースの超並列 SIMD 型プロセッサに、CAM ベーステーブルルックアップ符号化アーキテクチャを融合させた CAM ベース超並列 SIMD 型プロセッサは、DCT 等の処理を超並列に処理しつつ、テーブルルックアップ処理も高速に行えるため、最もクロックサイクル数が少なく、高速に処理を行えることが分かる。ここで、図 4.36 に、DCT とハフマン符号化、及びその他の処理のクロックサイクル数を視覚的に表したグラフを示す。なお、一般にランレンジス処理後は、データの個数自体が減少するが、超並

列 SIMD 型アーキテクチャでエントリを削減するためには、垂直チャネルでエントリを移動せねばならず、クロックサイクル数を消費することになる。そのため、ブランクなエントリが存在するものとして、処理を行っているので、各処理クロックサイクル数は総ピクセル数と比例関係にある。また、DSP のクロックサイクル数に関しても 8×8 のピクセルブロックで正規化した値を使用しているので、ピクセル数と比例関係にある。従って、ここでは 256×256 サイズの画像を例として使用する。表 4.7 の結果を元に、4:2:2 のカラー画像のクロックサイクル数を見積もった値を使用した。横軸は、各アーキテクチャの名前、縦軸は処理クロックサイクル数である。各グラフとも上から DCT, ハフマン符号化、及びその他の処理の順でクロックサイクル数を分けています。グラフから分かるように、SRAM ベース超並列 SIMD 型アーキテクチャは並列度を増加させることで、DSP と比較して総クロックサイクル数を約 48% 削減している。特に DCT に関しては、97% もの削減を実現している。しかしながら、並列度を向上させた代わりに、逐次処理が主体となるハフマン符号化のボトルネックが顕著に現れる結果となり、ハフマン符号化のクロックサイクル数は約 2.4 倍となった。CAM ベース超並列 SIMD 型プロセッサは、インターフェースモジュールに CAM を融合させることでテーブルルックアップ処理を高速に行うことが可能となったため、SRAM ベース超並列 SIMD 型プロセッサのボトルネックである、ハフマン符号化を大幅に削減することに成功した。削減率は約 92% にまで達している。また、DSP と比較しても 74% の削減となった。総クロックサイクル数で比較すると、SRAM ベースの超並列 SIMD 型プロセッサと比較して、約 73% の削減を達成し、DSP との比較では、約 86% の削減を実現した。

CAM ベース超並列 SIMD 型プロセッサは、SRAM ベース超並列 SIMD 型プロセッサのインターフェース部に CAM やコントローラを融合させたアーキテクチャとなっている。この構成をとることによって、クロックサイクル数の削減を実現できたが、同時に面積も増加することになる。そこで面積を考慮した上での、性能評価を表 4.8 に示す。評価は、単位面積当たりの処理ピクセル数とした。各アーキテクチャは全て 200 MHz で動作する。また、テーブルルックアップインターフェースモジュールの実装状況にあわせて、(a) をソフトマクロ使用時、(b) をハードマクロ (直交 SRAM、及び CAM) 使用時とした。表より、ソフトマクロ使用時では、CAM ベース超並列 SIMD 型プロセッサは、CAM とコントローラの面積増加にも関わらず、SRAM ベースの場合と比較して、約 3 倍の処理性能を持つことが分かった。また、DSP との比較では約 4 倍もの処理性能を持つことが分かった。ハードマクロ使用時では、CAM ベース超並列 SIMD 型アーキテクチャの値は 6.20 となり、SRAM ベースの場合と比較して、約 3.3 倍、DSP との比較では約 4.4 倍になることが分かった。なお、消費電力に関しても考察すると、SIMD 型演算モジュールが繰り返し演算を行っている間は、テーブルルックアップインターフェースモジュールはデータを転送するのみである。すなわち CAM による一致検索処理は行わない。また、反対にテーブルルックアップ処理を行っている際には、SIMD 型演算

モジュールで演算処理は行われない。そのため、CAMベース超並列 SIMD型プロセッサの消費電力は、SRAMベースと同様の 250 mW であると見積もることができる。

以上の検証より、SRAMベースの超並列 SIMD型プロセッサに、CAMベーステーブルルックアップ符号化アーキテクチャを融合させた CAMベース超並列 SIMD型プロセッサは、マルチメディアデータの繰り返し演算処理とテーブルルックアップ処理を、どちらも効率的に処理できるアーキテクチャであることが分かった。また、アーキテクチャの融合に伴う面積の増加率も、処理性能と比較した場合、わずかなものであることが分かった。

表 4.7: JPEG 処理クロックサイクル数の比較 (Y 画像)。

Width (Pixel)	Height (Pixel)	Number of pixels	Number of clock cycles		
			CAMベース 超並列 SIMD型プロセッサ	SRAMメモリベース 超並列 SIMD型プロセッサ	DSP
192	128	24,576	113,824	414,637	800,064
256	256	65,536	302,584	1,105,697	2,133,504
480	600	288,000	1,329,368	4,860,598	9,375,750
512	480	245,760	1,133,128	4,146,361	8,000,640
512	512	262,144	1,208,632	4,422,785	8,534,016
600	480	288,000	1,329,368	4,860,598	9,375,750
720	576	414,720	1,913,871	6,999,092	13,501,080
800	600	480,000	2,215,234	8,100,995	15,626,250
1,024	768	786,432	3,624,760	13,268,353	25,602,048
1,500	1,125	1,687,500	7,798,427	28,491,983	54,936,035

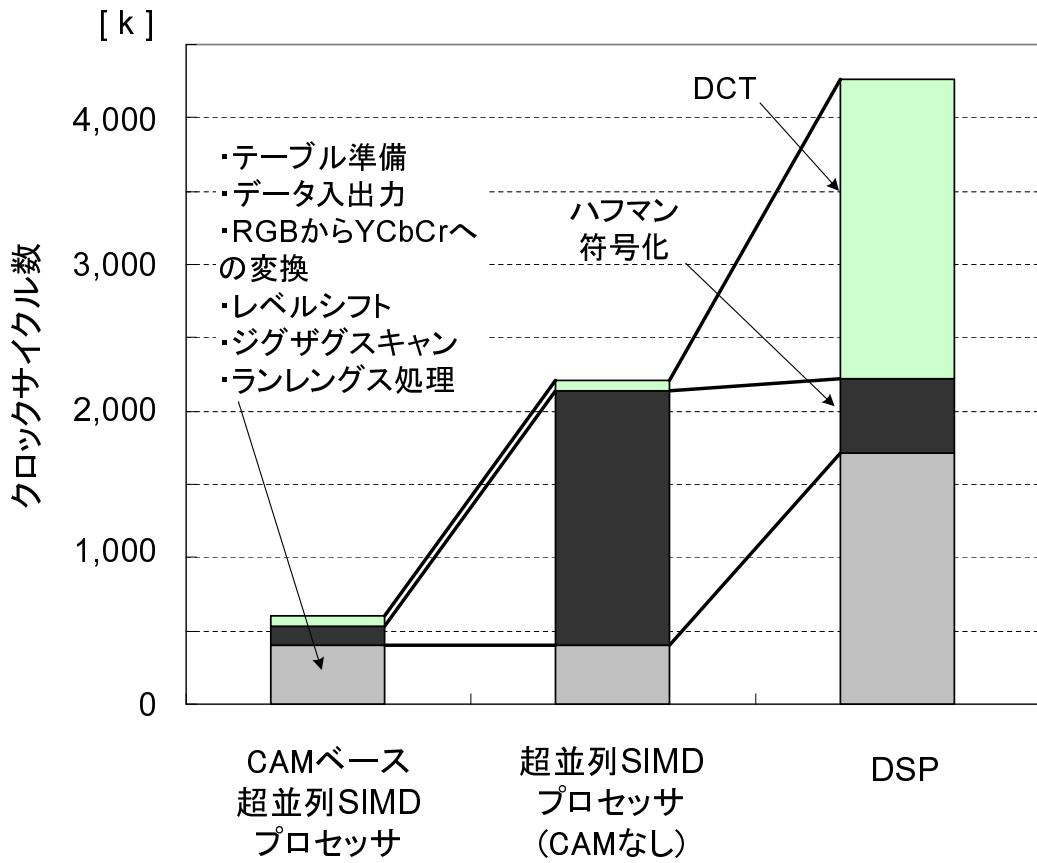


図 4.36: 各アーキテクチャによるクロックサイクル数の比較 (2,048 entry, Y, C_b, 及び C_r 画像).

表 4.8: 単位面積当たりの処理能力の比較.

(a) ソフトマクロ使用時

	連想メモリベース 超並列 SIMD プロセッサ	SRAMメモリベース 超並列 SIMD プロセッサ	DSP
Mpixel/mm ²	5.63	1.88	1.41

(b) ハードマクロ使用時

	連想メモリベース 超並列 SIMD プロセッサ	SRAMメモリベース 超並列 SIMD プロセッサ	DSP
Mpixel/mm ²	6.20	1.88	1.41

4.3 まとめ

本章では、テーブルルックアップ処理、及び繰り返し演算処理をどちらも高速に処理することのできるCAMベースの新しいアーキテクチャを提案、開発した。これはマルチメディア処理のボトルネックであるテーブルルックアップ符号化処理を高速に行うことを利用とし、繰り返し演算処理を2,048並列のSIMD型プロセッsingユニットで行うことにより高速化を実現するアーキテクチャである。提案アーキテクチャはCAM、SRAM、及びPEを融合した構造をとっているため小面積での高並列処理を実現し、単位面積当たりの演算能力を上げることで、動作周波数を抑え、低消費電力化を実現している。更に、CAMとSRAMアレイのデータ転送を工夫することにより、パイプライン処理も実現している。従来のDSP等と比較した結果、JPEGアプリケーションにおいては、最大87%のクロックサイクル数の削減を可能とし、単位面積あたりの処理能力では、約4.4倍の能力を有することがわかった。消費電力に関しても、モバイル用途向けに問題とならない程度であることを議論した。

以上より、提案アーキテクチャは従来のマルチメディアデータ処理LSIでは実現が難しかったテーブルルックアップ処理、及び繰り返し演算処理の効率的な両立を、アーキテクチャを工夫することで可能とした。また、マルチメディアデータ処理のみならず、プログラマブルに様々なアプリケーションに適用できる構成であることも示した。

第5章 結論

本論文では、マルチメディアデータ処理を効率的に行う CAM ベースマルチメディアデータ処理 LSI アーキテクチャの研究を行った。従来のマルチメディアデータ処理 LSI では、処理全体の 30%から 40%を占めるためボトルネックとなっていたテーブルルックアップ処理を符号化テーブル最適化ブロックを有する CAM、及びマルチポート CAM を用いることにより、高速かつ、高压縮に実現できるアーキテクチャを提案した。また繰り返し演算処理に対しては、2,048 並列の演算器で処理できる CAM を有する超並列 SIMD 型アーキテクチャを提案した。この 2つのアーキテクチャを融合させることで、マルチメディアアプリケーションの処理効率を全体的に向上させることができた。

本論文で得られた研究成果を、各章ごとにまとめて、5.1 節に示す。その後、5.2 節で本研究に関する今後の応用や将来展望を示す。

5.1 研究成果

第2章では、マルチメディアアプリケーションにおける代表的なテーブルルックアップ処理であるハフマン符号化について、符号化テーブルの切り替えを行い従来の方式よりも高压縮にデータを処理することのできるアーキテクチャを提案、その性能を評価した。

従来の研究は静的ハフマン符号化適用時において、データの圧縮率を考慮しないもののが多かった。また、考慮した場合においても、あらかじめ複数枚のテーブルを用意する手法を採用し、ハードウェア量の増加が問題となっていた。本章の研究は、静的ハフマン符号化における高压縮処理を 2つのテーブルを用意することでどのようなデータに対しても柔軟に対応できるようにした、実用性に富むものである。

(2-1) ハフマン符号化処理と並列に、符号化テーブルをマルチメディアデータの出現頻度にあわせてカスタマイズし、データの圧縮率を向上させることのできるアーキテクチャを開発した。

提案アーキテクチャは、CAM を用いて高速にハフマン符号化を行いながら、データの出現頻度をカウンタによって算出する。この算出結果を元に、出現頻度の高いデータは別に用意しているカスタマイズ用の符号化テーブルに最

適な符号が割り当てられる。この処理を符号化と並行して行い、適時切り替えることによってデータの圧縮率を向上させることが可能となる。

(2-2) 提案アーキテクチャを FPGA に実装し SoC 実装が可能であることを示した。

提案アーキテクチャを FPGA に実装し、エンコーダ（符号化モジュール）とオプティマイザ（テーブル更新／切り替えモジュール）の面積比を確認した。従来のアーキテクチャはエンコーダのみの場合が多いが、提案アーキテクチャの全体の面積は、エンコーダの約 1.9 倍であることが分かった。これをもとに、90 nm 7Cu CMOS テクノロジでのフルカスタム実装面積を見積もった結果、約 0.35 mm^2 であり、一般にモバイル機器向け SoC コアの面積は 1 cm^2 程度であるため、十分実装可能であることを示した。

(2-3) JPEG アプリケーションに適用し、画像の圧縮率向上の有効性を示した。

解像度が、 598×512 から、 $1,500 \times 1,125$ のベンチマーク画像を用意し、従来の静的ハフマン符号化アーキテクチャと提案アーキテクチャによる、リアルタイム符号化テーブル最適化処理の圧縮サイズを比較した。JPEG におけるハフマン符号化をターゲットアプリケーションとし、圧縮サイズを算出した結果、最大 28% の画像サイズ圧縮効果を確認できた。この差異は、画像の解像度が大きくなるにつれてより明確に効果があるものといえる。また、今回用いたベンチマーク画像を 4 Gbyte のフラッシュメモリに格納する場合、提案アーキテクチャによって圧縮した画像は 240 Mbyte 多く格納できることになり、その有効性を示した。

(2-4) JPEG アプリケーションに適用時の処理速度向上を実現した。

上記の (2-3) と同一の条件で、ハフマン符号化時の処理速度を算出した結果デジタルカメラや携帯電話等で、求められている速度仕様である 2, 3 fps を十分に満たしており、将来的に要求されると考えられる 5 fps 以上も満たしていることが分かった。

第 3 章では、マルチメディアアプリケーションにおいて高速処理のボトルネックとなっている、テーブルルックアップ処理を並列化によって高速化するアーキテクチャを提案、その有効性を示した。

本章の研究によって、従来実現することが難しかったマルチポート CAM を、他の研究に先駆けて開発することに成功した。また開発のみにとどまらず、実アプリケーションに適用することでマルチポート CAM の効果を示すことができた。

(3-1) CAM に複数の入出力ポートを実装したマルチポート CAM を提案した。

单一のメモリセルアレイを複数のポートで共有し、柔軟に効率的な並列一致検索処理が実現できる、FMCAM (Flexible Multi-ported Content Addressable Memory) を提案した。従来、並列一致検索処理を実現する場合、CAM 等の検索モジュールを複数の用意する必要があったが、ハードウェア量の増加が

著しく、モジュール間のデータコヒーレンシをとるのが難しいため実現は困難であった。FMCAMはビットパラレル・ブロックパラレル方式、格納データのカテゴリ分け、及び検索時のループカウンタの採用でマルチポート化に伴う比較器の増加を最小限にし、柔軟かつ効率的な並列検索処理を実現した。また、比較器とメモリセルを分離することで、CAMの特徴を持ちながら、通常のカスタマイズされたSRAMセルの使用を可能とし、小面積化、低消費電力化が可能であることを示した。

(3-2) FMCAMをFPGAに実装し処理性能の高さを示した。

FMCAMをFPGAに実装し、処理速度、及び面積の両面から総合的な性能を評価した。評価対象のCAMはソフトマクロ、及びハードマクロから様々な仕様のものを選択し、評価指標はAT(Area Time)積を使用した。検証の結果、ビット幅128、ワード数1,024のFMCAMが、他のCAMと比べて約60%以上小さい値となり、高性能であることを示した。

(3-3) FMCAMのポート数に伴うハードウェア増加率は小さいためスケーラビリティがあることを示した。

FMCAMは、比較器とメモリセルアレイを分離していることで、ポート数の増加に関わらずハードウェア量の増加が抑えられている。カテゴリ分けを施していない単純なマルチポートCAMと通常のシングルポートCAMを並列に配置した並列CAMを用意し、ポート数増加に伴うハードウェア量の増加をFMCAMと比較したところ、ポート数4の場合、並列CAMはFMCAMの約2.36倍ハードウェア量が増加する。そのため提案FMCAMは面積効率が高いことを示せた。

(3-4) テーブルルックアップ処理向けにFMCAMのアーキテクチャを開発した。

FMCAMをテーブルルックアップ処理に適用するために、比較クロックサイクル数の削減を実現する工夫を施したAdapted FMCAMを提案した。テーブルルックアップ処理は、符号化前のデータと符号化後のデータが1対1であることに着目し、一致検索結果が出力された後、直ちに次の検索動作を実行できるシングルサーチモードを導入した。また、適用アプリケーションによって1カテゴリに格納されるデータ数が異なるため、ループアドレスカウンタ値を任意に設定できる仕様にした。これらのアプリケーションオリエンティッドの改良の結果、これまでのFMCAMと比較して約50%のクロックサイクル数削減を実現し、その有効性を示した。

(3-5) Adapted FMCAMをFPGA／ASICに実装し、スケーラビリティの高さを示した

Adapted FMCAMをFPGAに実装し、次に90 nm 7Cu CMOSテクノロジでのASICの論理合成を行った。ポート数を1から16まで変化させて動作周波数を確認したところ、FPGA、及びASIC共に、ハードウェア量は線形

に増加することが分かった。通常のマルチポートメモリは、ポートの増加に伴い面積が 2 乗に比例した形で増加する。しかしながら、Adapted FMCAM は、ポートの増加時に追加されるのはポートモジュールのみであるため、面積の増加が抑えられているといえる。また、ポート数の増加に関わらず動作周波数の減少が少ないことを示した。特に ASICにおいてはハードウェア量の増加が線形で一定であるにもかかわらず、制約条件の 200 MHz を下回ることは無い。この結果によって、Adapted FMCAM は、ポートの増加に対しスケーラブルであり、その拡張性の高さを立証した。

(3-6) 並列テーブルルックアップ処理を JPEG アプリケーションに適用し、その高速処理を実現した。

Adapted FMCAM を JPEG アプリケーションに適用し、ハフマン符号化における並列符号化処理の効果を検証した。解像度が、 154×144 から、 $1,024 \times 768$ のベンチマーク画像を用意し、従来の静的ハフマン符号化アーキテクチャと Adapted FMCAM による、ハフマン符号化に必要なクロックサイクル数を算出した。その結果、従来の FMCAM に対しクロックサイクル数を約 43% 削減し、DSPとの比較では約 93% の削減を実現した。これにより並列テーブルルックアップ処理の効果の高さを示した。

(3-7) Adapted FMCAM の小面積実装と単位面積当たりの高性能を示した。

(3-6)において、Adapted FMCAM の並列テーブルルックアップ効果を示したが、実装時の面積も考慮することで、更に検証を行った。ポート数の増加に伴う単位面積当たりの処理能力を評価に用い、従来の FMCAM と複数個の DSP を並列に配置した並列 DSP とで比較した。比較の結果、ポート数 16 の場合において従来の FMCAM の約 1.7 倍、DSP の約 3.8 倍単位面積当たりの処理能力が高いことを示した。また、Adapted FMCAM は、ポート数を増加するほどこの値は向上するため、マルチメディアデータ処理 LSI に並列テーブルルックアップ処理用のコアとして組み込むことの有効性を示した。

(3-8) Adapted FMCAM とリアルタイムテーブル符号化アーキテクチャとの融合アーキテクチャを提案した。

Adapted FMCAM の並列テーブルルックアップアーキテクチャに、第 2 章で有効性を示したリアルタイムテーブル符号化アーキテクチャを融合した、テーブルルックアップを高速かつ高压縮に処理できるアーキテクチャを提案した。提案アーキテクチャは、FMCAM とマルチポートメモリを組み合わせて、並列なテーブルルックアップ処理を行い、同時にマルチポート向けに改良を加えたオプティマイザによって高压縮処理を行うことが可能である。

(3-9) Adapted FMCAM とリアルタイムテーブル符号化の融合アーキテクチャによる JPEG アプリケーションの高速処理を実現した。

(3-8)で提案したアーキテクチャを用いて、解像度が、 192×128 から、 $1,024$

× 768 のベンチマーク画像に対して処理速度と圧縮サイズの算出を行った。その結果、従来の DSP、もしくは SRAM ベース及び CAM ベースのエンコーダと比較して約 3 倍から 20 倍の処理速度の向上を確認した。また画像の圧縮サイズに関しては、平均 20% の圧縮率向上を確認できた。

第 4 章では、マルチメディアアプリケーションにおいてテーブルルックアップ処理と並んで、もう 1 つの処理である、繰り返し演算処理の高速化を実現するアーキテクチャを提案した。そして、提案アーキテクチャを第 2 章及び第 3 章で示した CAM ベースのテーブルルックアップアーキテクチャと融合することでマルチメディアアプリケーションを高効率に処理することのできるアーキテクチャを実現した。

本章の研究によって、性質が異なる上記 2 つの処理を、並行して効率よく処理できるマルチメディアデータ処理 LSI の開発を実現することができた。効率よい両立化の実現は、従来のマルチメディアアプリケーション処理のボトルネックを解消した新規性の高いものである。

(4-1) 超並列 SIMD 型演算アーキテクチャを提案した。

従来の DSP や専用ハードウェアは、ワードシリアルにデータの処理を行っていたが、データの処理ベクトルを変更することで、ビットシリアルにデータを処理する超並列 SIMD 型演算アーキテクチャを提案した。提案アーキテクチャは、メモリレジスタ部と演算器を混載することにより広いバンド幅と低消費電力化を実現し、1,024 から 2,048 という高い並列度を実現した。

(4-2) 超並列 SIMD 型演算アーキテクチャを ASIC へ実装し、高い処理性能を示した。

超並列 SIMD 型演算アーキテクチャを共同研究で開発した。90 nm 7Cu CMOS テクノロジを用いて開発した結果、面積が 3.1 mm²、消費電力が 250 mW@200 MHz であり、処理能力を従来の DSP と比較した。その結果 16 bit 加算時において、単位面積当たりの処理能力が約 70 倍、消費電力当たりの処理能力が約 12 倍と、共に高い値を示すことができ、マルチメディアデータ処理 LSI の繰り返し演算を行うコアとして有効であることを示した。

(4-3) 超並列 SIMD 型演算アーキテクチャと CAM ベースのテーブルルックアップアーキテクチャの融合を実現した。

超並列 SIMD 型演算アーキテクチャの直交インターフェース部に CAM を組み込むことにより、マルチメディアアプリケーションを構成するテーブルルックアップ処理と繰り返し演算処理を、どちらも高効率に処理することのできる、アーキテクチャである CAM ベース超並列 SIMD 型演算アーキテクチャを提案した。融合に際しては、SRAM、CAM、直交 SRAM、及び直交 CAM のうち要求性能を満たす組み合わせを検討し、直交 SRAM が 2 つ、2 バンクの CAM が 1 つの構成が最適であることが分かった。また、これらの回路を

プログラマブルに動作させるため、アドレス空間、命令セットやコントローラの仕様を決定し、プログラマブルなアーキテクチャを実現した。

(4-4) CAM ベース超並列 SIMD 型演算アーキテクチャを FPGA / ASIC に実装し高い処理性能を示した。

提案アーキテクチャを FPGA に実装し、USB インターフェースを用いて PC からのコンソールアプリケーションより基本動作の確認を行った。次に、90 nm 7Cu CMOS テクノロジを用いて論理合成を行った。直交 SRAM セルアレイをフルカスタムで作成し、CAM、直交 SRAM、及びコントローラの面積及び消費電力の見積もりを算出した結果、約 0.39 mm²、約 32.4 mW@200 MHz と高性能を実現した。この結果と、(4-2) で示した結果を考慮した場合、全体の面積は約 3.49mm²、消費電力は、200 mW@200 MHz となり、一般にモバイル機器向け SoC コアの面積は約 1 cm² 程度であるため、実装には十分小さい面積で実現可能であることを示した。

(4-5) CAM ベース超並列 SIMD 型演算アーキテクチャによる JPEG アプリケーションの高速処理を実現した

提案アーキテクチャを用いて、JPEG アプリケーションにおける DCT や量子化等の繰り返し演算処理を、超並列 SIMD 型演算アーキテクチャで、ハフマン符号化を CAM によるテーブルルックアップ符号化によって処理した。解像度が、192 × 128 から、1,500 × 1,125 のベンチマーク画像に対して JPEG 全体の処理クロックサイクル数を算出した結果、従来の DSP と比較して、約 87% のクロックサイクル数削減を実現した。また、単位面積当たりの処理能力は 200 MHz で DSP の約 4.4 倍となり、CAM ベース超並列 SIMD 型演算アーキテクチャはマルチメディアデータ処理 LSI 向けの SoC コアとして大変有効であることを立証した。

5.2 研究成果の応用と将来展望

本研究では、大量のマルチメディアデータを高速、高圧縮に処理することができる CAM ベースの高性能マルチメディアデータ処理 LSI アーキテクチャを開発した。以下に、今後の研究成果の応用と将来展望について示す。

5.2.1 研究成果の応用

本研究成果の今後の応用として、主として以下の 3 つが挙げられる。

- (1) 暗号や誤り訂正処理等のアプリケーションに対する効率的な圧縮処理。
- (2) ルーティング処理やマルチコアキャッシュへの並列一致検索処理の応用。

- (3) CAM ベース超並列 SIMD 型演算アーキテクチャのモバイル機器用アプリケーションへの応用
- (1) 近年携帯電話をはじめとするモバイル機器、家庭用ゲーム機、及びデジタル家電等は多機能化が進んでいる。そのためこれらの機器に搭載される LSI は、音声、動画像の処理を始め、安全な通信のための暗号や誤り訂正処理等のアルゴリズムを効率よく処理できる能力が求められる。特にモバイル機器においては、取り扱うデータ量の増大に対し効率よく圧縮処理を行い転送データ量を抑える工夫が必要となる。しかしながら、暗号や誤り訂正処理においては従来の動画像に用いられている不可逆な圧縮（復号時にデータの欠損が生じる）方式は適用できない。従って、可逆圧縮方式をうまく用いて高圧縮を実現する必要がある。本研究で提案した CAM ベースのリアルタイム符号化テーブル最適化アーキテクチャは、ハフマン符号化を圧縮アルゴリズムに用いているため可逆圧縮であり、CAM を用いているためリアルタイムに符号化を実現しながら、圧縮効率の向上も実現できる。このアーキテクチャを先に述べた LSI のコアとして応用することで、様々はアプリケーションに対して今後増大するデータ量の圧縮を効率よく実現することが期待される。
- (2) 現在、並列一致検索処理が有効であると考えられる応用例としてルーティング、マルチコアに対するキャッシュ等がある。ルーティングに関しては、事業所やプロバイダにおける基幹ルータへの応用が挙げられる。近年、FTTH 等の普及により、基幹ルータにはより高速なルーティングが求められている。従来このルーティングには CAM を複数配置して処理を行うことが多かったが、ここに FMCAM を応用することで、実装面積、処理能力、消費電力、及びルーティングテーブルのメインテナンス等様々な利点が得られると考えられる。実際の応用に関しては、FMCAM をターナリ（3 値）化し、TCAM として実装することが考えられる。またキャッシュに関しては、CPU 等のプロセッサでマルチコア化が急激に進む中、これまでキャッシュに使用されていた CAM に FMCAM を応用することが考えられる。これによって各 CPU のコアによるキャッシュテーブルの共有が可能となり、更に CAM の能力を用いて目的のデータをすばやく検索することが可能になると考えられる。
- (3) 本研究では提案した CAM ベース超並列 SIMD 型演算アーキテクチャを JPEG アプリケーションによって性能を評価した。今後はより広範囲で性能を評価するために動画像や暗号処理、誤り訂正等のアプリケーションへの適用を検討する。具体的なターゲットアプリケーションとしては、携帯電話の動画像配信に利用されている MPEG-4、携帯電話の地上デジタル放送視聴に利用されている H.264 がある。暗号化に関しては、ISO (International Organization for Standardization)/IEC (International Electrotechnical Commission) にて国際標準暗号に制定されている、AES (Advanced Encryption Standard) や

Triple DES (Data Encryption Standard) 等がある。誤り訂正符号方式に関しては、次世代のアルゴリズムである LDPC (Low Density Parity Check) 符合がある。

5.2.2 将来展望

本研究で提案した高性能 CAM ベースマルチメディアデータ処理 LSI アーキテクチャの将来展望として、更に処理性能を向上図ったプロセッシングアーキテクチャが考えられる。近年、CPU 等のプロセッサにおいてマルチコア化の研究、開発が盛んになっており、ユーザが容易に入手できるようになってきた。Intel の Core Duo 等のデュアルコアプロセッサは、ノートパソコンにも使用されており、次世代ゲーム機のプレイステーション 3 に使われている Cell と呼ばれるプロセッサは、9 個のコアを内蔵している。また、2006 年にサンフランシスコで開催された Intel Developer Forum では、コア数が 80 個であるメニイコアプロセッサの試作品も発表されており、毎秒 1 Tbyte/s の浮動小数点演算を可能としている [58]。このような背景からも、マルチコア化は高性能化への 1 手法として大変有効であると考えられる。そこで、今後の研究の拡張として第 4 章にて述べた、超並列 SIMD 型演算モジュールをマルチコア化して、FMCAM を実装させた階層並列構造 SIMD 型プロセッシングアーキテクチャが考えられる。図 5.1 にその概念図を示す。階層並列構造 SIMD 型プロセッシングアーキテクチャは、SIMD 型演算モジュールを並列もしくはインタリーブ動作させることにより大量のデータを並列に処理することが可能であり、各コアがテーブルルックアップ処理を行う場合には FMCAM を用いて一致検索処理を行うことが可能となる。また、圧縮が必要なアプリケーションに対してはオプティマイザを用いることによって高圧縮な可逆圧縮処理也可能としている。このアーキテクチャは、並列度を増加することで容易に処理性能を向上させることができるために、モバイル機器の用途のみならず、デスクトップ PC やサーバ向けのコアとしても十分に威力を発揮することが期待できる。また、最小距離検索連想メモリを導入することによるあいまい検索処理を用いて、高度並列知能システムアーキテクチャへ発展することが期待できる。

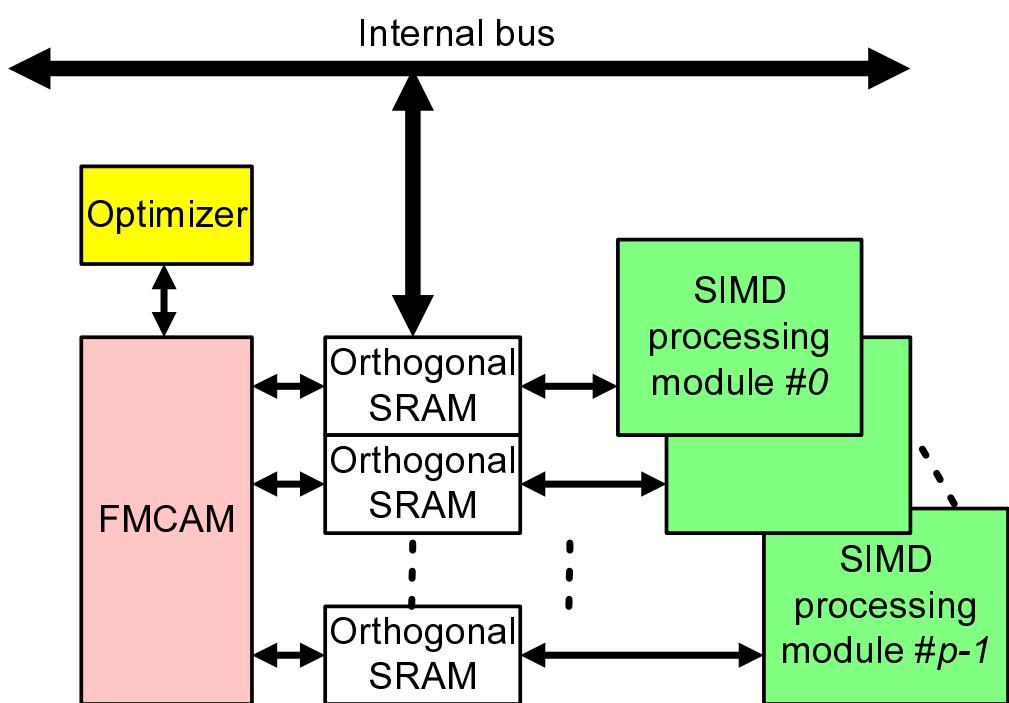


図 5.1: 階層並列構造 SIMD 型プロセッシングアーキテクチャ.

参考文献

- [1] M. Anderson and P. Karlström, “Parallel JPEG processing with a hardware accelerated DSP processor,” Examensarbete utfört i Datorteknik vid Tekniska Högskolan I Linköping, May 2004.
- [2] T. Miyazaki and I. Kuroda, “Video codec implementation on programmable processors,” IEICE Trans., vol. J83-A, no. 12, pp. 1339–1348, Dec. 2000.
- [3] <http://edevice.fujitsu.com/fj/MARCOM/find/19-1j/pdf/j19-1-1.pdf>.
- [4] 篠原文彦, 岡田賢治, “ブロードバンド教科書,” IE インスティテュート, pp. 178–213, July 2001.
- [5] 亀山 渉, 花村 剛, “改訂版デジタル放送教科書 上,” インプレス, pp. 246–247, Oct. 2004.
- [6] <http://pr.fujitsu.com/jp/news/2005/02/7.html>.
- [7] 斎藤光男, 田胡治之, 岡田豊史, “ゲーム機向けシステム LSI,” 電子情報通信学会誌, vol. 84, no. 8, pp. 559–564, Aug. 2001.
- [8] http://japan.renesas.com/fmwk.jsp?cnt=sh7785_root.jsp&fp=/products/mpumcu/superh_family/sh7780_series/sh7785_group/.
- [9] A. E. Slade and H. O. McMahon, “A cryotron catalog memory system,” Proc. EJCC, pp. 115–120, 1956.
- [10] D. A. Huffman, “A method for the construction of minimum redundancy codes,” Proc. IRE, no. 40, pp. 1098–1101, Sept. 1952.
- [11] D. E. Kunth, “Dynamic Huffman coding,” J. Algorithms, vol. 6, pp. 163–180, 1985.
- [12] ISO/IEC 10918.
- [13] ISO/IEC 13818-2.

- [14] S. J. Lee, K. H. Yang, J. S. Song, and C. W. Lee, “An efficient memory allocation scheme for Huffman coding of multiple sources,” *Signal Process.: Image Commun.*, vol. 14, pp. 311–323, Mar. 1997.
- [15] S. M. Lei and M. T. Sun, “An entropy coding system for digital HDTV applications,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 1, pp. 147–155, Mar. 1991.
- [16] “Data compression with MUSIC CAMs,” *MUSIC Semiconductors Application Note AN-N6*, Nov. 1998.
- [17] W. W. Lu and M. P. Gough, “A fast-adaptive Huffman coding algorithm,” *IEEE Trans. Commun.*, vol. 41, no. 4, pp. 535–538, Apr. 1993.
- [18] L. Y. Liu, J. F. Wang, R. J. Wang, and J. Y. Lee, “CAM-based VLSI architectures for dynamic Huffman coding,” *IEEE Trans. on Consumer Electronics*, vol. 40, no. 3, pp. 282–289, Aug. 1994.
- [19] S. Iwasaki, H. Inoue, and T. Katura, Japan patent laid-open publication, no. 9-65334, Mar. 1997.
- [20] 熊木武志, 岩井啓輔, 黒川恭一, “多機能マルチポートCAMとその応用例,” 情報処理学会第64回全国大会, vol. 1, pp. 55–56, Mar. 2001.
- [21] 熊木武志, 岩井啓輔, 黒川恭一, “改良型多機能マルチポートCAMの提案,” Forum on Information Technology 2002 (FIT2002), vol. 1, no. C-9, pp. 205–206, Sept. 2002.
- [22] T. Kumaki, K. Iwai, and T. Kurokawa, “A proposal of MFM-CAM and its applications,” *Proc. ITC-CSCC2002*, vol. 1, pp. 224–227, July. 2002.
- [23] 熊木武志, 岩井啓輔, 黒川恭一, “フレキシブルマルチポート連想メモリ,” 電子情報通信学会論文誌, vol. J87-D-I, no. 1, pp. 12–21, Jan. 2004.
- [24] T. Kumaki, Y. Kuroda, T. Koide, H. J. Mattausch, H. Noda, K. Dosaka, K. Arimoto, and K. Saito, “Multi-port CAM based VLSI architecture for Huffman coding with real-time optimized code word table,” *Proc. IEEE International Midwest Symposium on Circuits And Systems (MWSCAS’05)*, pp. 55–58, Aug. 2005.
- [25] T. Kumaki, K. Iwai, and T. Kurokawa, “A flexible multi-port content addressable memory,” *Systems and computers in Japan*, vol. 37, no. 11, pp. 57–67, Oct. 2006.

- [26] T. Kumaki, Y. Kono, M. Ishizaki, T. Koide, and H. J. Mattausch, “Scalable FPGA/ASIC implementation architecture for parallel table-lookup-coding using multi-ported content addressable memory,” IEICE Trans. Inf. & Syst., vol. E90-D, no. 1, pp. 346–354, Jan. 2007.
- [27] CS6100 Motion JPEG Encoder, <http://www.amphion.com/cs6100.html>.
- [28] J. Redford, “Parallelizing JPEG,” ChipWrights Inc., Apr. 2003.
- [29] S. Fukae, N. Omori, H. J. Mattausch, T. Koide, T. Inoue, and T. Hironaka, “Comparison of the hierarchical and crossbar-based architectures for the construction multibank multiport memory,” IEICE Technical Report, vol. VLD2002-62, pp. 37–42, June 2002.
- [30] M. Hariyama and M. Kameyama, “Collision detection VLSI processor for highly - safe intelligent vehicles using a multiport content-addressable memory,” Interdisciplinary Information Sciences, vol. 5, no. 2, pp. 109–116, 1999.
- [31] <http://cco-sj-2.cisco.com/japanese/warp/public>.
- [32] M. Hariyama, K. Sasaki, and M. Kameyama, “Collision detection VLSI processor for intelligent vehicles using a hierarchically-content-addressable memory,” IEICE trans. Electoron, vol. E82-C, no. 9, pp. 1722–1729, Sept. 1999.
- [33] M. Hariyama, F. Yamaguchi, and M. Kameyama, “Implementation of an ultra-high-speed path planning VLSI processor using ROM-type content-addressable memory,” Trans. SICE, vol. 37, no. 3, pp. 235–241, 2001.
- [34] T. Hayashi and T. Miyazaki, “FLASH: Fast and scalable table-lookup engine architecture for telecommunications,” IEICE Trans. Inf. & Syst., vol. E85-D, no. 10, pp. 1636–1644, Oct. 2002.
- [35] M. Hirata, H. Yamada, H. Nagai, and K. Takahashi, “A versatile data string-search VLSI,” IEEE J. Solid-State Circuits, vol. 23, no. 2, pp. 329–334, Apr. 1988.
- [36] K. Kobayashi, K. Tamaru, H. Yasuura, and H. Onodera, “A bit-parallel block-parallel functional memory type parallel processor architecture,” IEICE Trans. Electron, vol. E76-C, no. 7, pp. 1151–1158, July 1993.
- [37] M. Motomura, J. Toyoura, K. Harata, H. Ooka, H. Yamada, and T. Enomoto, “A 1.2-million transistor, 33-MHz, 20-b dictionary search processor (DISP) ULSI with a 160-kb CAM,” IEEE J. Solid-State Circuits, vol. 25, no. 5, pp. 828–834, Oct. 1990.

- [38] K. J. Schultz and P. G. Gulak, “Architectures for large-capacity CAMs,” INTEGRATION the VLSI Journal, vol. 18, pp. 151–172, June 1995.
- [39] S. Okugawa, “The content addressable memory and it’s application,” bit, vol. 15, no. 4, pp. 318–329, 1983. (in Japanese).
- [40] 小林和淑, 寺田一彦, 田丸啓吉, “動画像圧縮システムのためのベクトル量子化機能メモリ型並列プロセッサ,” 1999 年第 1 回 LSI IP デザイン・アワード IP 優秀賞, 1999. <http://ne.nikkeibp.co.jp/IPJapan/ipaward/990525ipa3.html>.
- [41] 近越一真, 濱田芳隆, 吉田正廣, 鈴木八十二, “ハミング距離検索機能を有する CMROM の照合特性,” 信学論 (C), vol. J83-C, no. 7, pp. 658–659, July 2000.
- [42] 池田 誠, 浅田邦博, “最近傍データ探索機能を有する CAM マクロ,” 1999 年第 1 回 LSI IP デザイン・アワード 開発奨励賞, 1999. <http://ne.nikkeibp.co.jp/IPJapan/ipaward/990624ipa10.html>.
- [43] S. M. S. Jalaleddine and L. G. Johnson, “Associative IC memories with relational search and nearest-match capabilities,” IEEE J. Solid-State Circuits, vol. 27, no. 6, pp. 892–900, June 1992.
- [44] M. Defossez, “Content addressable memory (CAM) in ATM applications,” Xilinx Application Note XAPP 202, Feb. 2000.
- [45] J. L. Brelet and B. New, “Designing flexible, fast CAMs with Virtex family FPGAs,” Xilinx Application Note XAPP 203, Sept. 1999.
- [46] “CAM comparison: APEX 20KE vs. Virtex-E devices,” ALTERA, Technical Brief 61, Dec. 1999.
- [47] “MU9C1965A/L LANCAM MP,” MUSIC Semiconductors Preliminary Data Sheet, Oct. 1998.
- [48] K. J. Schultz and P. G. Gulak, “Fully parallel integrated CAM/RAM using preclassification to enable large capacities,” IEEE J. Solid-State Circuits, vol. 31, no. 5, pp. 689–699, May 1996.
- [49] 佐々木敬泰, 井上智宏, 大森伸彦, 弘中哲夫, マタウシュ ハンス, 小出哲士, “オンチップマルチプロセッサ用共有キャッシュの実現方式の検討とその性能面積評価,” 電子情報通信学会論文誌, vol. J87-D-I, no. 3, pp. 350–363, Mar. 2004.

- [50] H. J. Mattausch, K. Kishi, and T. Gyohten, “Area-efficient multi-port SRAMs for on-chip data-storage with high random-access bandwidth and large storage capacity,” IEICE Trans. Electron., vol. E84-C, no. 3, pp. 410–417, Mar. 2001.
- [51] H. Sato, T. Yoshida, M. Matsuo, T. Kengaku, and K. Tsuchihashi, “A dual-issue RISC processor for multimedia signal processing,” IEICE Trans. Electron., vol. E81-C, no. 9, pp. 1374–1381, Sept. 1998.
- [52] 圓山俊幸, 松尾雅仁, 寺岡栄一, “メディアプロセッサ D10V,” 三菱電機技報, vol. 74, no. 3, Mar. 2000.
- [53] 吉田豊彦, “省電力携帯電話用メディアプロセッサ D10V,” bit, Jan. 2000.
- [54] M. Nakajima, H. Noda, K. Dosaka, K. Nakata, M. Higashida, O. Yamamoto, K. Mizumoto, H. Kondo, Y. Shimazu, K. Arimoto, K. Saitoh, and T. Shimizu, “A 40GOPS 250mW massively parallel processor based on matrix architecture,” ISSCC Dig. Tech. Papers, pp. 410–412, Feb. 2006.
- [55] 越智 宏, 黒田英夫, “JPEG & MPEG 図解でわかる 画像圧縮技術,” 日本実業出版社, pp. 112–113, Oct. 2003.
- [56] W. H. Chen, C. H. Smith, and S. C. Fralick, “A fast computational algorithm for the discrete cosine transform,” IEEE Trans. Communications, vol. COM-25, no. 9, pp. 1004–1009, 1977.
- [57] 酒井善則, 吉田俊之, “映像情報符号化,” オーム社, pp. 118–120, Dec. 2001.
- [58] <http://developer.intel.com/idf/us/spring2006/index.htm>.

謝辞

本研究の遂行にあたり、終始御懇切な御指導と御鞭撻を賜りました広島大学ナノデバイス・システム研究センター 小出 哲士 助教授、マタウシュ ハンスユルゲン 教授に深く感謝の意を表します。また、本研究全般にあたり御指導並びに有益な御討論を頂きました広島大学ナノデバイス・システム研究センター長 岩田 穆 教授、同センター 角南 英夫 教授、吉川 公磨 教授、横山 新 教授、芝原 健太郎 助教授、中島 安理 助教授、広島大学大学院先端物質研究科 半導体集積科学専攻三浦 道子 教授、宮崎 誠一 教授、佐々木 守 助教授、東 清一郎 助教授、江崎 達也 助教授、村上 秀樹 助手、吉田 毅 助手、同大学院 量子物質科学専攻 樋口 克彦 助教授、並びに量子物質科学専攻の教員各位に深く感謝の意を表します。

超並列 SIMD 型プロセッシングアーキテクチャの研究全般に当たり多大な御協力と御助言を頂きました、株式会社ルネサステクノロジ 斎藤 和則 氏、有本 和民 氏、堂阪 勝己 氏、中田 清 氏、野田 英行 氏、行天 隆幸 氏、矢野 裕二 氏、黒田 泰斗 氏に心から感謝の意を表します。

フレキシブルマルチポート連想メモリの研究に当たり、御指導、御鞭撻を賜りました防衛大学校 情報工学科 黒川 恒一 助教授、岩井 啓輔 助手に心から感謝の意を表します。

高性能 CAM ベースマルチメディアデータ処理 LSI アーキテクチャの開発全般にあたり昼夜を問わずに御協力、及び有益な御意見を頂きました幸野 豊 氏、石崎 雅勝 氏、田上 正治 氏に心から感謝の意を表します。

日頃から御協力を頂き、お世話になった桐山 治 博士、アリ アーマディ 博士、森本 高志 博士、アベディン ムハマドアノワルル 氏、上口 光 氏、末吉 徹也 氏、白川 佳則 氏、藤井 崇之 氏、碧山 賢一 氏、足立 英和 氏、上村 一弘 氏、山岡 功佑 氏、椋田 佑也 氏、粟根 和俊 氏、田中 裕己 氏、リトンガ ムハマド アリフィン 氏、和泉 伸也 氏、岡崎 啓太 氏、榎原 尚吾 氏、広島大学ナノデバイス・システム研究センターの先輩、同輩、後輩、事務の方々、広島大学大学院先端物質科学研究科の事務の方々、並びに独立行政法人 日本学術振興会 研究者養成課の方々に深く感謝します。また、研究に関する事務手続き等で特にお世話になりました広島大学ナノデバイス・システム研究センターの久良 佳都子 氏、葦原 千秋 氏、久保田 秋子 氏、國貞 尚子 氏、門前 智美 氏、淀川 恭子 氏、広島大学学術部 藤井 優江 氏、広島大学大学院先端物質研究科 中田 伸明 氏に感謝します。

本研究におけるチップ試作において使用した CAD ツールは東京大学大規模集積システム設計教育研究センターを通し、シノプシス株式会社、日本ケイデンス株

謝辞

式会社、メンター株式会社の協力で行われたものである。

FPGA を用いた FMCAM、及び CAM を有するテーブルルックアップアーキテクチャのシステム開発にあたり、いろいろご助言をいただきました三菱電機エンジニアリング株式会社 立崎 賢治 氏、東京エレクトロン デバイス株式会社 富田 和寿 氏、に感謝いたします。また、FPGA 設計ツールはザイリンクス株式会社、シンプリシティ株式会社、及びメンター株式会社のアカデミックプログラムによるものである。

本研究の一部は 21 世紀 COE プログラム「テラビット情報ナノエレクトロニクス」、独立行政法人 日本学生支援機構、独立行政法人 日本学術振興会 特別研究員奨励費 (No.175303)、の助成を受けて行われたものであり、ここに感謝の意を表します。

研究に対する心構えなどを教えていただきました、防衛大学校 数学教育室 寺澤 順 教授、航空自衛隊 梶崎 浩嗣 氏、海上自衛隊 野毛 寛之 氏、陸上自衛隊 清家 秀律 氏、神原 公仁 氏、鈴木 諭司 氏、大韓民国海軍 権 五俊 氏、ベトナム社会主義共和国陸軍 グン チュオン ソン 氏に心から感謝の意を表します。

最後になりましたが、日常生活から研究までいろいろ支えていただいた妻 美沙と息子達 慧弥、熙弥、並びに友人の方々に心から感謝の意を表します。

2006 年 12 月 熊木 武志

研究業績リスト

【公表論文】

- 熊木武志, 岩井啓輔, 黒川恭一, “フレキシブルマルチポート連想メモリ,” 電子情報通信学会論文誌, Volume J87-D-I, No. 1, pp. 12–21, Jan., 2004.
- Takeshi Kumaki, Keisuke Iwai and Takakazu Kurokawa, “A flexible multiport content-addressable memory,” Systems and computers in Japan, Volume 37, No. 11, pp. 57–67, Oct., 2006.
- Takeshi Kumaki, Yasuto Kuroda, Masakatsu Ishizaki, Tetsushi Koide, Hans Jürgen Mattausch, Hideyuki Noda, Katsumi Dosaka, Kazutami Arimoto and Kazunori Saito, “Real-time Huffman encoder with pipelined CAM-based data path and code-word-table optimizer,” IEICE Transactions on Information & Systems, Volume E90-D, No. 1, pp. 334–345, Jan., 2007.
- Takeshi Kumaki, Yutaka Kono, Masakatsu Ishizaki, Tetsushi Koide and Hans Jürgen Mattausch, “Scalable FPGA/ASIC implementation architecture for parallel table-lookup-coding using multi-ported content addressable memory,” IEICE Transactions on Information & Systems, Volume E90-D, No. 1, pp. 346–354, Jan., 2007.

【国際会議】

- Takeshi Kumaki, Keisuke Iwai and Takakazu Kurokawa, “A proposal of MFM-CAM and its applications,” Proceedings of International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Volume 1, pp. 224–227, July, 2002.
- Takeshi Kumaki, Yasuto Kuroda, Tetsushi Koide, Hans Jürgen Mattausch, Hideyuki Noda, Katsumi Dosaka, Kazutami Arimoto and Kazunori Saito, “CAM-based VLSI architecture for Huffman coding with real-time optimization of the code word table,” Proceedings of IEEE International Symposium on Circuits And Systems (ISCAS), pp. 5202–5205, May, 2005.
- Takeshi Kumaki, Yasuto Kuroda, Tetsushi Koide, Hans Jürgen Mattausch, Hideyuki Noda, Katsumi Dosaka, Kazutami Arimoto and Kazunori Saito, “Multi-port CAM based VLSI architecture for Huffman coding with real-time optimized code word table,” Proceedings of IEEE International MidWest Symposium on Circuits And Systems (MWSCAS), pp. 55–58, Aug., 2005.
- Takeshi Kumaki, Yutaka Kono, Masakatsu Ishizaki, Tetsushi Koide and Hans Jürgen Mattausch, “Application of multi-ported CAM for parallel coding,” Proceedings of IEEE Asia pacific Conference on Circuits and Systems (APCCAS), pp. 1861–1864, Aug., 2006.

発表論文リスト

【国内研究会等発表】

- 熊木武志, 岩井啓輔, 黒川恭一, “多機能マルチポート CAM とその応用例,” 情報処理学会第 64 回全国大会, Volume 1, pp. 55–56, Mar., 2001.
- 熊木武志, 岩井啓輔, 黒川恭一, “改良型多機能マルチポート CAM の提案,” Forum on Information Technology 2002 (FIT2002), Volume 1, No. C-9, pp. 205–206, Sept., 2002.
- 鈴木諭司, 西山 怜, 熊木武志, 梶崎浩嗣, 岩井啓輔, 黒川恭一, “CAM を用いた不正検知型侵入検知システム実装手法の提案,” 情報処理学会第 65 回全国大会, Volume 3, pp. 555–556, Mar., 2002.
- Takeshi Kumaki, Yutaka Kono, Masakatsu Ishizaki, Tetsushi Koide and Hans Jürgen Mattausch, “CAM-based Huffman coding architecture for real-time applications and code-word-table optimizer,” Hiroshima International Symposium on Nanoelectronics for Tera-Bit Information Processing, June, 2006.

【特許】

- 小出哲士, 熊木武志, 黒田泰斗, マタウシュハンスユルゲン, 野田英行, 堂阪勝己, 有本和民, 斎藤和則, “符号化装置,” 日本特許, 特願 2005-146-211, 2005 年 5 月 19 日出願.
- 行天隆幸, 堂阪勝己, 野田英行, 有本和民, 黒田泰斗, 熊木武志, 小出哲士, マタウシュハンスユルゲン, 石崎雅勝, “プロセッサおよびそれを用いたテーブル変換方法,” 日本特許, 特願 2006-283803, 2006 年 10 月 18 日出願.

【その他研究活動】

- 小出哲士, 熊木武志, 黒田泰斗, マタウシュハンスユルゲン, 野田英行, 堂阪勝己, 有本和民, 斎藤和則, “符号化装置,” 日本特許, 特願 2005-146-211, 2005 年 5 月 19 日出願.
- 行天隆幸, 堂阪勝己, 野田英行, 有本和民, 黒田泰斗, 熊木武志, 小出哲士, マタウシュハンスユルゲン, 石崎雅勝, “プロセッサおよびそれを用いたテーブル変換方法,” 日本特許, 特願 2006-283803, 2006 年 10 月 18 日出願.