平成 14 年度

学士学位論文

DDMP 向けマルチメディア処理用ライブラリに関する検討

A Study on Multimedia Processing Library for DDMP

1030320　　山岡 正明

指導教員　　岩田 誠 教授

高知工科大学 情報システム工学科

要 旨

DDMP 向けマルチメディア処理用
ライブラリに関する検討

山岡 正明

開発環境を向上させる方法の一つにライブラリ化がある．システム中で頻繁に使用されるプログラム・コードを登録し，再利用することにより，重複するプログラムを作成する時間を省略し，開発効率を向上することができる．

DDMP(Data-Driven Multimedia Processor) はデータ駆動型アーキテクチャを自己同期型パイプラインにより実現しており，ストリーミング処理に適したアーキテクチャを持っていることからマルチメディア処理に適していることがいえる．

データフローグラフは処理ノード間データ依存関係を明示的に表現できるので，超並列実行に適しているだけでなく，システム機能の中核的な表現法としても優れた性質を持っている．しかしながら，現行の信号処理向きのデータ駆動型マルチプロセッサ DDMP 用ソフトウェア開発ツール DADT(DDMP Application Designer's Toolkit) は，マシン命令を対象にしたフローグラフの開発しかできない．このため，より高次な記述水準で応用システム開発が可能な環境の構築が急務となっている．

高水準記述の実現の一環として，既存のマルチメディア拡張命令セットを調査し，抽出した命令セットをライブラリとして実装し，性能評価を行った．評価した結果，マルチメディア処理用ライブラリを適用することにより，スループットは半減したものの，大幅にプログラム作成時間を削減できることがわかった．

キーワード　データフローグラフ，ライブラリ，マルチメディア処理，高水準記述

Abstract

A Study on Multimedia Processing Library for DDMP

Masaaki Yamaoka

With growth of micro-fabrication technology, data network system for ubiquitous computing is within reach. In such a system, multimedia-oriented processing program becomes important and very huge, the productivity of software would be unmanageable.

In order to improve the production efficiency, high-level description of software is key.With DFG (Data-Flow Graph), the specification can be converted into object code directly, and the comprehension for module interface can be clearly cognized by its diagramatic representation, thus high-parallel execution subtended in multimedia processing is naturally described. Moreover DFG can be executed by DDMP (Data-Driven Multimedia Processor) with dynamic data-driven scheme. DDMP, which is realizes high-parallel processing by its self-timed pipelines, is hopeful architecture for multimedia processing.However DADT (DDMP Application Designer's Toolkit), current software development tool, is designed to target assembler-level description, thus the development of application systems in a high level is extremely difficult.

As one sphere of the study on high-level development environment, higher-level description of software using library is studied. In this paper, a multimedia processing library for DDMP is developed based on the existing multimedia instruction sets of Intel's IA-32.

key words dataflow graph,library,multimedia processing,high-level development environment

目次

第 1 章　序論　1

第 2 章　マルチメディア処理用ライブラリの要件　3

2.1　緒言 3

2.2　DDMP アーキテクチャの概要 3

2.2.1　自己同期型パイプライン 3

2.2.2　データ駆動型プログラム 5

2.3　DADT の課題 5

2.4　ライブラリ実現の課題 7

2.5　結言 7

第 3 章　既存マルチメディア命令セットの調査　9

3.1　緒言 9

3.2　MMX 命令セット 9

3.2.1　SIMD 9

3.2.2　機能とデータ型 10

3.2.3　オペレーションモード 11

3.3　SSE ／ SSE2 命令セット 12

3.4　結言 17

第 4 章　ライブラリの実現法　18

4.1　緒言 18

4.2　FIS の導入 18

4.3　データ駆動型プログラムへの変換 18

4.4　マルチメディア処理用ライブラリ 19

4.5 DDMP 向けマルチメディアライブラリ 20
4.6 結言 20

第 5 章 性能評価 24

5.1 緒言 24
5.2 FFT を用いた評価 24
5.3 評価結果 25
5.4 考察と課題 26
5.5 結言 26

第 6 章 結論 27

謝辞 30

参考文献 31

付録 A マルチメディア処理用ライブラリ表 32

図目次

2.1 環状パイプライン 4
2.2 パケット転送の抽象図 4
2.3 フローグラフ 5

3.1 SIMD 実行モデル 10
3.2 MMX 命令で追加されたデータ型 11
3.3 128 ビット・パックド単精度浮動小数点データ型 12
3.4 パックド単精度浮動小数点の操作 13
3.5 スカラ単精度浮動小数点の操作 13
3.6 SSE2 で追加されたデータ型 15
3.7 パックド倍精度浮動小数点の操作 16
3.8 スカラ倍精度浮動小数点の操作 16

4.1 ライブラリ情報の例：複素乗算モジュール 19

5.1 周波数間引き FFT の流れ図表示 (N = 8) 25

表目次

4.1 MMX，SSE，SSE2 命令表:その 1 21
4.2 MMX，SSE，SSE2 命令表:その 2 22
4.3 MMX，SSE，SSE2 命令表:その 3 23

5.1 評価結果 26

A.1 ライブラリ検索用命令表:その 1 32
A.2 ライブラリ検索用命令表:その 2 33
A.3 ライブラリ検索用命令表:その 3 34

第 1 章

序論

近年，携帯電話等の情報システムに求められる要求として高度かつ多様化したサービスが挙げられる．そのようなサービスには動画，音声，信号等のマルチメディア処理が大半を占める．より快適なマルチメディア処理の高速化が求められる．マルチメディア処理の特徴に大量のデータを扱うことが多いという点が挙げられる．このマルチメディア処理の対策として以下の二つが挙げられる．

- ネットワークの広帯域化
- 処理ノードの高速化

ネットワークには広帯域化の手段として，ディジタル化によりチャネルを増やす ISDN(Integrated Services Digital Network)，既存の電話回線を利用した ADSL(Asymmetric Digital Subscriber Line)，光ファイバケーブルなど伝送媒体が主である．一方，処理ノードを高速化する手段には，プロセッサの性能向上，およびソフトウェアのアルゴリズムの改良等が挙げられる．その方法の一つとして，ライブラリ化が挙げられる．システムのファイル入出力やメモリ管理機能など，多くのアプリケーションで共通して利用できるプログラムコードは，ライブラリとして独立させておき，プログラムからこれをリンクして利用するのが一般的である．これにより，逐次にプログラムを作成せずともプログラムにリンク情報を記載しておけば，いつでも呼び出すことができ，より複雑なシステムにも対応できる．もう一つは，CPU ベンダが個々に用意したマルチメディア用の拡張命令セットを使用する方法である．既存のノイマン型 CPU の代表的なマルチメディア向け拡張命令セットには，x86 系 CPU 用の MMX (MultiMedia eXtension)

や SSE(Streaming SIMD Extensions)/SSE2 命令，SPARC 系 CPU の VIS (Visual Instruction Set)，PA-RISC 系 CPU の MAX (Multimedia Acceleration eXtensions)，MIPS 系 CPU の MDMX (Digital Media eXtension) などがある．これらの命令セットを使用することにより，マルチメディア処理部分の性能の向上，プログラミングの生産性を向上させている．しかし，現在のノイマン型アーキテクチャではクロック同期などの特徴からマルチメディア処理には不向きであり，アーキテクチャを変更しない限り，飛躍的な性能向上は望めないことから，マルチメディア処理に指向したアーキテクチャを持つプロセッサの開発が求められている．

データ駆動型マルチメディアプロセッサ（以下，DDMP:DataDriven Multimedia Processor）[1] データ駆動原理と，自己同期型パイプラインを採用していることからストリーミング処理に適しているという特徴を持っている．この特徴は高度なメディア及び信号処理などの通信アプリケーションに適しているということがいえる．また，データ駆動型プログラムの記述法としてデータフローグラフを採用している．これは処理ノード間のデータ依存関係を明示的に表現できるので，超並列実行に適しているだけではなく，ソフト／ハードを含むシステム機能の中核的な表現法としてもすぐれた性質を持っている．しかし，現行の DDMP 用ソフトウェア開発ツール DADT（DDMP Application Designer's Toolkit）は、マシン命令を対象としたフローグラフの記述に特化されており，マルチメディアアプリケーション等のより高度なプログラムを記述することは容易ではない．さらに DDMP の実装は多岐にわたり，固有の命令セットが多く存在する．この課題に対応するには，より高次な記述水準かつ柔軟に応用システム開発が可能な環境の構築が急務となっている．

本研究では，上記のような高水準な開発環境を実現する方法に関する検討の一環として，既存のマルチメディア拡張命令セットから，DDMP 向けマルチメディア処理用ライブラリの拡充を行う．第 2 章では上述の DDMP 向けマルチメディア処理用ライブラリの要件を整理する．第 3 章では既存のマルチメディア命令セットを調査し，整理した DDMP 向けマルチメディア処理用ライブラリの要件と照らし合わせ分析結果を記述する．それを第 4 章でライブラリとして実装する方法を提案する．第 5 章で性能評価を行い，その結果を記述する．

第 2 章

マルチメディア処理用ライブラリの要件

2.1 緒言

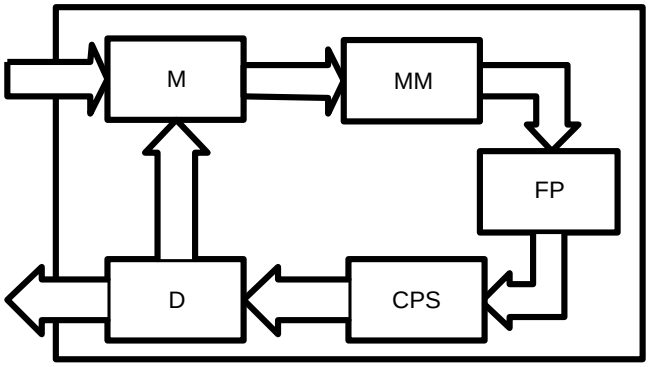
本章では DDMP 向けのアーキテクチャの説明とともに，ライブラリ実現の際に求められる要件を記述する．

2.2 DDMP アーキテクチャの概要

2.2.1 自己同期型パイプライン

データ駆動型プロセッサ (DDMP) の特徴として，動的データ駆動型処理方式が挙げられる．その処理方式を実現したものが環状構造パイプライン (図 2.1 である．各処理部分の説明を以下に示す．

- M 部外部からの入力パケットと Branch 部からのパケットを調停し，合流させたパケットは MM 部に渡される．
- MM 部 M 部から渡されたパケットを一時的に内部のメモリに保持する．もう一つのパケットが揃ったら FP 部へ渡す．
- FP 部発火パケットに対して，パケットのタグで指定された演算を実行する．必要ならば，メモリにもアクセスする．演算が終了したパケットは CPS 部に渡される．



MM　:Matching Memory(待ち合わせメモリ)
FP　:Functional Processing Unit(演算部)
CPS　:Cache Program Storage(プログラム記憶部)
M　:Flow Merging Module(合流部)
D　:Flow Diverting Module(分岐部)

図 2.1　環状パイプライン

- CPS 部内部に保有した
- D 部 D 部ではパケットの行き先に基づいて，ナノ PE 内外へパケットを分流する．

この環状構造の内部に自己同期型パイプライン (図 2.2 参照) が組み込まれている．このパイプラインは先行ステージが空の場合にパケットが転送され，空でない場合は，パケットは現ステージに停滞し，先行のステージが空になるまで待つ．この自己同期型パイプラインは，柔軟かつ遅延耐性が強いという特徴を有している．よって、同期型パイプラインよりも，マルチメディア処理に適しているということが言える．

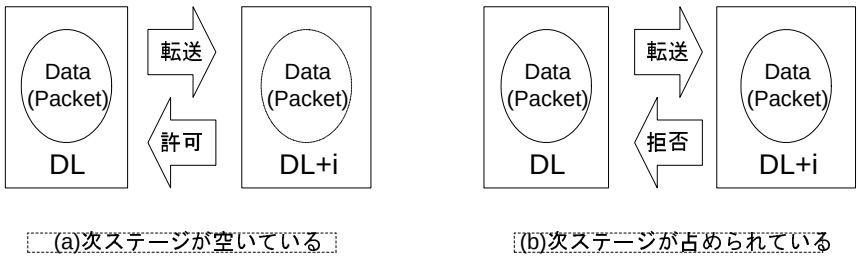


図 2.2　パケット転送の抽象図

2.2.2 データ駆動型プログラム

データ駆動型プログラムは処理部分を示すノードとデータの流れを示すアークで構成されている．このデータフローグラフは視覚的にプログラムの超並列実行を表現できるので大量のデータを処理するマルチメディアアプリケーションなどの高度なプログラム構築に向いている．

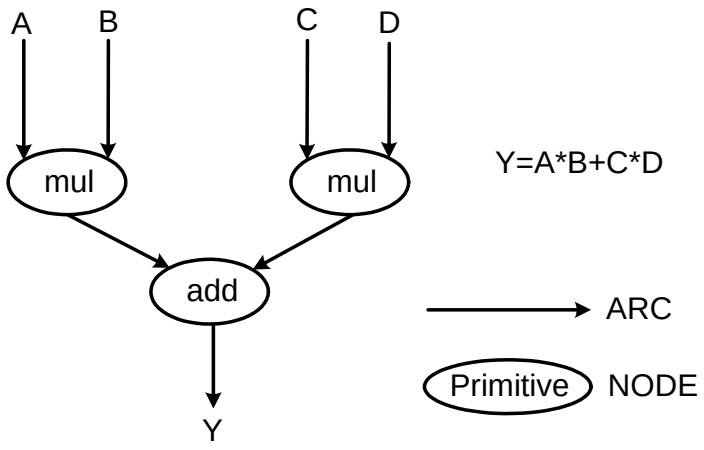


図 2.3 フローグラフ

2.3 DADT の課題

DADT(DDMP Application Designer's Toolkit) は，プログラム記述，プログラムの接続チェック，オブジェクトファイルの生成，シミュレーション，デバッグに至るまでプログラム開発に関わる作業を一貫してサポートする GUI ベースのツールキットである．そして以下のツールから構成される．

1. FED(Flowgraph EDitor) データ駆動型プログラムは DFG(Data Flow Graph) によって表現される．FED は DFG を記述するためのグラフィカルツールである．
2. Validator FED で記述された DFG の接続情報を調べ，プログラムに文法的な誤りがないかどうかをチェックするツールである．

3. Allocator FED で記述された DFG の PE 接続チェックを行います．マルチ PE 時に使用する．
4. Estimator 作成された DFG(プログラム) を DDMP の各 nPE にどれくらいの処理負荷がかかるかを定量的に見積もるツールである．
5. Assembler 作成された DFG をオブジェクトファイルへ変換する．
6. Simulator Assembler から得られたプログラムオブジェクトファイルの内容に沿って，入力パケットファイルの内容を時刻に応じて処理していき，DDMP の動作をシミュレートする．

次に，プログラムの作成手順を以下に示す．

1. プログラムソース記述

   プログラムソースファイルを記述する．FED を用いて DFG の記述を行い，記述が完了したら，Validator を用いて DFG の文法チェックを行う．少し複雑なプログラムを記述したいときは，プログラムの一部をモジュール化して階層記述することもできる．本研究ではこのモジュール化したプログラムをライブラリとして提供する．
2. プログラムアセンブル

   Allocator,Estimator,Assembler を用いてプログラムオブジェクトファイルを生成する．プログラムオブジェクトファイルは DDMP のプログラムメモリの内容を表しており Simulator で使用する．
3. プログラム入力データパケットファイル作成

   プログラムに投入する入力データパケットファイルをテキストエディタを用いて記述する．また，入力データパケットファイルには，DDMP の内臓データメモリの初期値を記述することもできるので，ルックアップテーブルなどを作成することができます．
4. パラメータ設定とファイル実行

   Simulator の GUI において，DDMP 内部のパラメータレジスタの設定を行う．次にオブジェクトファイルと入力パケットファイルを指定してシミュレーションを実行する．

5. シミュレーション結果の確認

   シミュレーション実行の結果得られた出力パケットファイルを確認する．また，Simulator の GUI にもシミュレーション結果が表示されるので，その内容も合わせて確認する．

以上の手順でプログラムの開発を進めていく．しかし，現在の開発環境では，極めて単純なマシン命令しか実装されておらず，より高度なアプリケーションを開発しようとすると，非常に大規模かつ複雑になり，取り扱いが煩雑になってしまう．本研究ではこの課題を解決する方法として頻繁に使用されるマルチメディア処理プログラムをライブラリとして提供することで，より高水準な開発環境の実現を目指す．

2.4　ライブラリ実現の課題

ライブラリを実装することを考える場合，以下の課題が考えられる．

1. どのような機能を提供すべきか

   まずマルチメディア処理の種類を調査し，そこから頻繁に使用されるものを抽出する作業を行う．そのなかからさらに DDMP のアーキテクチャと照らし合わせ，アーキテクチャの利点を損なわない処理を抽出する．
2. 使用されるデータ型にはどのような種類のものがあるか同じくマルチメディア処理の種類の中から，どのようなデータが頻繁に使われているかを調査する．

本研究では，上記の二つの課題を軽減するためすでに検討されている既存のマルチメディア拡張命令セットを調査し，その結果をマルチメディア処理用ライブラリとして実装する．

2.5　結言

本章では DDMP アーキテクチャを説明するとともに，DDMP の応用に適したマルチメディア処理用ライブラリの要件を明らかにした．次章では既存マルチメディア命令セットの

2.5 結言

調査を行い，DDMP に将来的に実装すべき命令があるかどうかを検討する．

第 3 章

既存マルチメディア命令セットの調査

3.1 緒言

本章では既存のマルチメディア命令セットを調査し前章のライブラリの要件と照らし合わせ，DDMP に将来的に実装すべき命令を抽出する．

3.2 MMX 命令セット

3.2.1 SIMD

マルチメディア処理では 1 つの命令で，複数のデータを同時に処理することが多い．そこで，MMX 命令 [5] では SIMD(Single Instruction Multiple Data) を採用した．この方式はマルチメディア・データを取り扱うマイクロプロセッサや、DSP、スーパー・コンピュータなどにおいて実装されている．この SIMD 実行モデル（図 3.1 を参照）がターゲットとしているのはメディア，通信，グラフィック関係のアプリケーションである．このようなアプリケーションではサイズの小さいデータ型に対して同一演算を膨大な回数実行するアルゴリズムを使用している．例えば，オーディオ・データは 16 ビットで表現されていることが多いので MMX 命令を使用すれば，一つの命令で 4 つのワード・データを処理することが出来る．また，ビデオやグラフィック・データは 8 ビットであることが多いので，個々でも MMX 命令を使用することにより，一つの命令で 8 個のバイトデータを同時に演算できる．

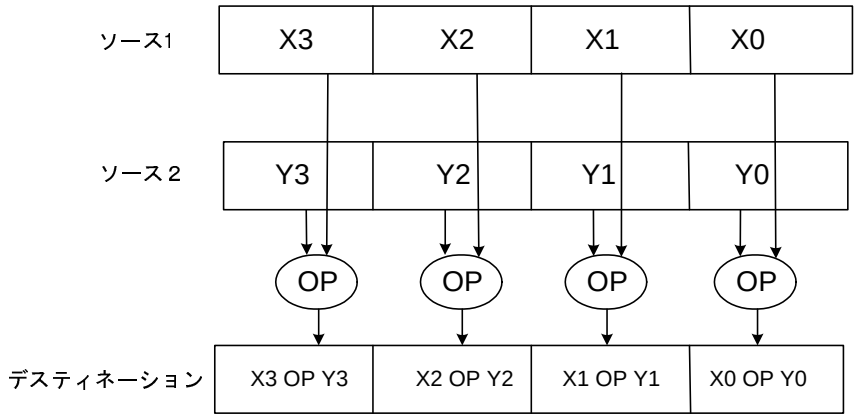


図 3.1　SIMD 実行モデル

3.2.2　機能とデータ型

MMX 命令セットは IA − 32 アーキテクチャで拡張命令セットとして実装されたマルチメディア向けの命令セットである．まず MMX 命令が提供している機能を以下に記述する．

- データ転送命令
- 算術命令
- 比較命令
- 論理命令
- 変換命令
- アンパック命令
- シフト命令
- EMMS 命令

算術演算命令ではパックド・データに対して，加算，減算，乗算，乗算および加算の各演算を実行する．比較命令ではパックド・データに対して，大小判定，一致判定を行い，1 と 0 からなるマスクビットを出力する．論理命令では両オペランドに対してビットごとの論理演算を行う．変換命令では大きいデータ型を小さいデータ型，例えばダブルワードならばワードに，ワードならばバイトにデータの変換を行う．アンパック命令では変換命令の逆でバイ

トならばワードに，ワードならばダブルワードに変換する．シフト命令ではパックド・データに対して，算術右シフト，論理右シフト，論理左シフトの各演算を実行する．EMMS 命令では MMX テクノロジステートをクリアする命令である．

この命令セットでは図 3.2 に示される以下の 3 種類のデータ型を追加している．

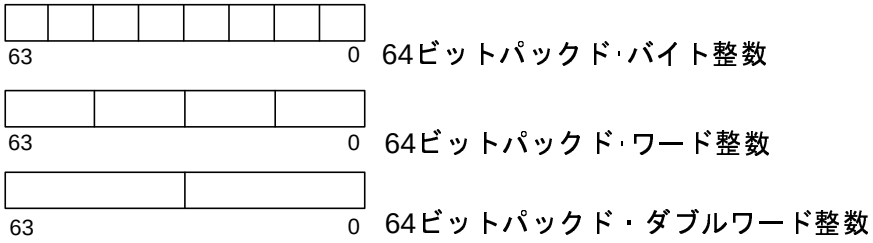


図 3.2　MMX 命令で追加されたデータ型

3.2.3　オペレーションモード

整数算術演算を実行したとき，計算結果が範囲外になる場合がある．このような場合範囲外処理をしなければならないのだが，MMX 命令では命令を記述する際に，以下の 3 種類から範囲外処理を選択することが出来る．

- ラップアラウンド算術
- 符号あり飽和算術
- 符号なし飽和算術

ラップアラウンド算術では範囲外の真の結果は切り捨てられる．例えば加算命令で正のオーバーフローが発生した場合，オーバーフロービットは無視され，演算結果の下位ビットが出力される．これはオペランドの範囲を制御できるアプリケーションに向いている．符号あり飽和算術は範囲外の結果が出た場合，表現できる符号付整数の値に制限される．例えば符号付ワードで正のオーバーフローが出た場合，演算結果は 16 ビットで表される最大の数である 7FFFH となる．負のオーバーフローが出た場合，8000H に飽和される．符号なし飽和算術はは範囲外の結果が出た場合，表現できる符号付整数の値に制限される．例えば符号なし

ワードで正のオーバーフローが出た場合，演算結果は 16 ビットで表される最大の数である FFFFH となる．負のオーバーフローが出た場合，0000H に飽和される．

3.3　SSE ／ SSE2 命令セット

SSE 命令セットは高度な 2 D および 3 D グラフィックス，モーション・ビデオ，画像処理，音声認識，音声合成，およびビデオ会議などのアプリケーションに対するパフォーマンスを向上させるために拡張された命令セットである．SSE では新たに 128 ビットパックド単精度浮動小数点データ型が追加された (図**??**参照)．

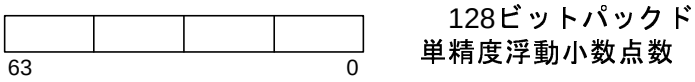


図 3.3　128 ビット・パックド単精度浮動小数点データ型

次に，SSE で追加された機能は，以下の 4 つに分類される．

- パックド，およびスカラ単精度浮動少数点命令
- 64 ビット SIMD 整数命令
- ステート管理命令
- キャッシュ制御命令，プリフェッチ命令，メモリアクセス順序命令

パックド，およびスカラ単精度浮動少数点命令は以下の機能に分類される．

- データ転送命令
- 算術演算命令
- 論理命令
- シフト命令
- シャッフル命令
- 変換命令

パックド単精度浮動小数点命令は 4 つの 32 ビット単精度浮動小数点に対して SIMD 演算を実行する (図 3.4 参照)．スカラ単精度浮動小数点命令は最下位ダブルワードに対して演算を

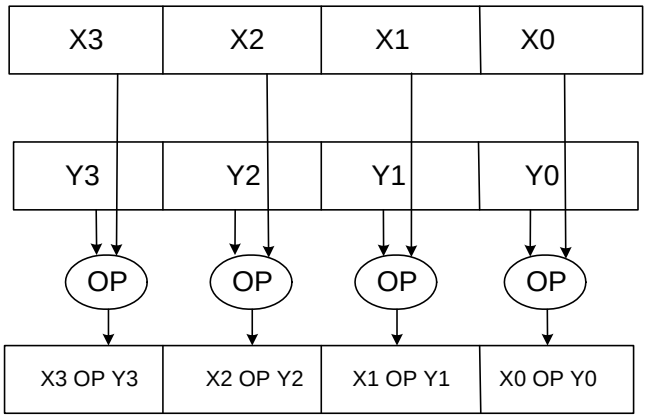


図 3.4　パックド単精度浮動小数点の操作

実行し，上位 3 つのダブルワードはそのままディスティネーションオペランドに渡される (図 3.5 参照)．

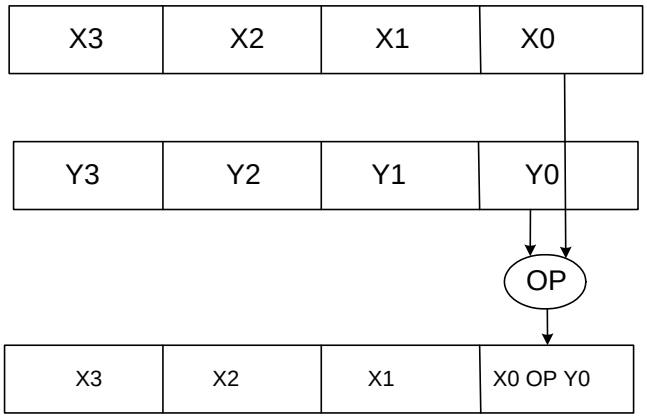


図 3.5　スカラ単精度浮動小数点の操作

SSE データ転送命令は，レジスタ同士またはレジスタとメモリの間でパックド単精度浮動小数点データを転送する．SSE 算術演算命令は，パックドおよびスカラ単精度浮動小数点値に対して，加算，減算，乗算，除算，逆数計算，平方根計算，平方根の逆数計算，および最大値／最小値計算を実行する．SSE 論理命令はパックド単精度浮動小数点値の AND,

AND NOT，OR，および XOR 演算を実行する．SSE 比較命令はパックド単精度浮動小数点値同士を比較し，その結果の 0 と 1 からなるマスクビットをデスティネーション・オペランドに返す．SSE シャッフル命令とアンパック命令は 2 つのパックド単精度浮動小数点オペランドの内容をシャッフルまたはインターリーブし，その結果をデスティネーション・オペランドに返す．SSE の変換命令は単精度浮動小数点フォーマットとダブルワード整数フォーマットとの間で，パックド変換またはスカラ変換を実行する．64 ビット SIMD 整数命令は 64 ビットパックド・データに対して平均値計算，コピー，最大値／最小値計算，ビットマスク，乗算，絶対差，およびシャッフル演算を実行する．MXCSR ステート管理命令は MXCSR レジスタのロードとストアを行う．キャッシュ制御命令は非テンポラルなヒントを使用してレジスタからメモリにデータをストアする命令である．非テンポラルなヒントは，可能な限りデータをキャッシュ階層内に書き込まないようにプロセッサに指示する．プログラムが参照するデータはテンポラルと非テンポラルなデータに分けることが出来る．近い将来に参照される可能性が高いデータをテンポラル，近い将来には参照されないデータは非テンポラルとなる．例えば，一般的にプログラムデータはテンポラルであるが，３ D グラフィックス・アプリケーションの表示リストなどのマルチメディア・データは非テンポラルである事が多い．テンポラルなデータはキャッシュへ，非テンポラルなデータはキャッシュしないことが望ましい．非テンポラルなデータがキャッシュにストアされることをキャッシュの汚染という．次にプリフェッチ命令はプロセッサ内の指示されたキャッシュ・レベルにデータをロードする．これにより，アプリケーション・コード内の高いパフォーマンスが要求される部分で，データ・アクセスのレイテンシを最小限に抑えることができる．

SSE2 命令セットは，高度な 3D グラフィックス，ビデオ・デコーディング／エンコーディング，音声認識，電子商取引，インターネット，科学計算，工学計算などのアプリケーション向けに開発された拡張命令セットである．以下に新たに追加されたデータ型を図 3.6 に表す．そして，SSE2 は以下の 4 つの機能に分類される．

- パックドおよびスカラ倍精度浮動小数点命令

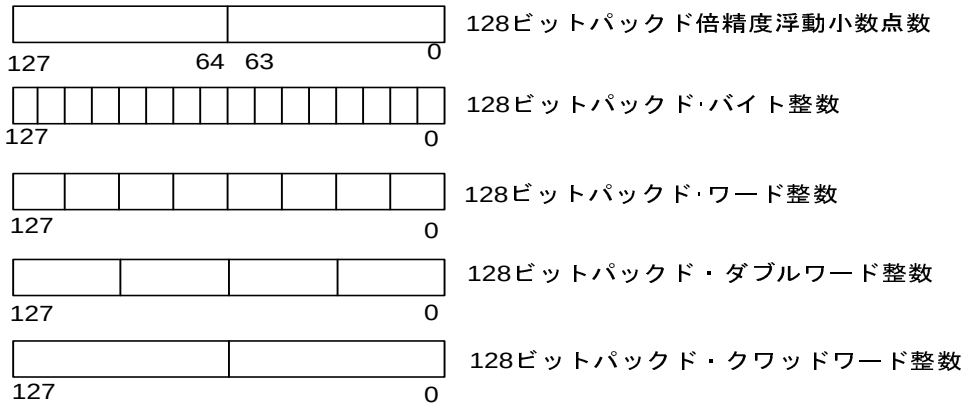


図 3.6 SSE2 で追加されたデータ型

- 64 ビットおよび 128 ビット SIMD 整数命令
- 128 ビット拡張
- キャッシュ制御命令および命令順序命令

パックド倍精度浮動小数点命令の機能は以下のグループに分類される.

- データ転送命令
- 算術演算命令
- 論理命令
- シフト命令
- シャッフル命令
- 変換命令

パックド単精度浮動小数点命令は 4 つの 32 ビット単精度浮動小数点に対して SIMD 演算を実行する (図 3.7 参照).

スカラ単精度浮動小数点命令は最下位ダブルワードに対して演算を実行し，上位 3 つのダブルワードはそのままディスティネーションオペランドに渡される (図 3.8 参照).

SSE2 データ転送命令は，レジスタ同士またはレジスタとメモリの間でパックド倍精度浮動小数点データを転送する．SSE2 算術演算命令は，パックドおよびスカラ倍精度浮動小数

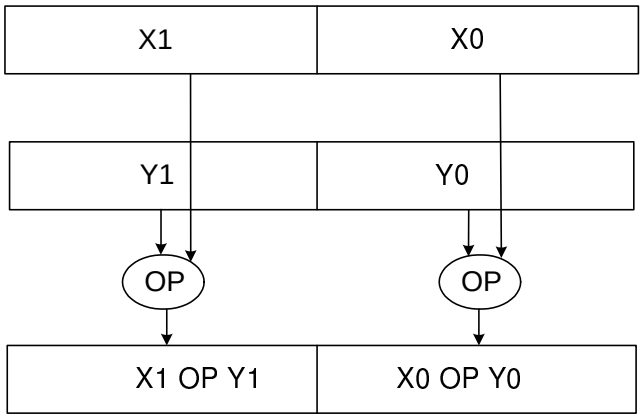


図 3.7　パックド倍精度浮動小数点の操作

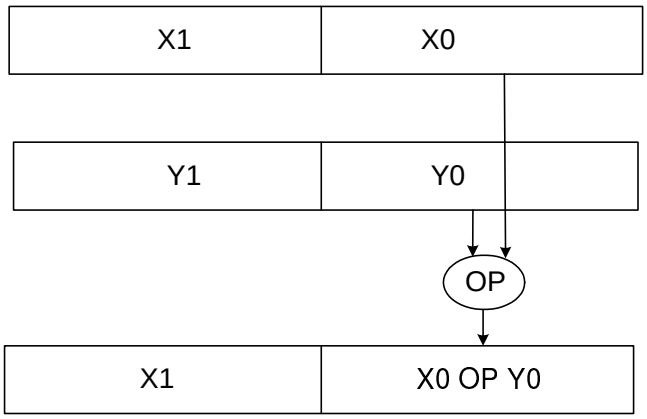


図 3.8　スカラ倍精度浮動小数点の操作

点値に対して，加算，減算，乗算，除算，平方根計算，および最大値／最小値計算を実行する．SSE2 論理命令はパックド倍精度浮動小数点値の AND，AND NOT，OR，および XOR 演算を実行する．SSE2 比較命令はパックド倍精度浮動小数点値同士を比較し，その結果の 0 と 1 からなるマスクビットをデスティネーション・オペランドに返す．SSE2 シャッフル命令とアンパック命令は 2 つのパックド倍精度浮動小数点オペランドの内容をシャッフルまたはインターリーブし，その結果をデスティネーション・オペランドに返す．SSE2 の変換命令は以下のデータ型の間でパックド変換またはスカラ変換を実行する．

- 倍精度浮動小数点フォーマットと単精度浮動小数点フォーマット

- 倍精度浮動小数点フォーマットとダブルワード整数フォーマット
- 単精度浮動小数点フォーマットとダブルワード整数フォーマット

128 ビット SIMD 整数命令では MMX，SSE で実装されていた 64 ビット SIMD 整数命令が自動的に 128 ビットに拡張される．例えば PADDB は 64 ビットの MMX 命令であるが，SSE2 ではこれが自動的に 128 ビットの加算命令に拡張されるキャッシュ制御では新たに倍精度浮動小数点とダブルクワッドワードのデータ型の命令が新たに追加された．機能で追加された点はない．フェンス命令では新たに LFENSE 命令と MFENSE 命令が追加された．LFENSE 命令ではこの命令が実行される前のデータのロードの実行がすべて終わるまで見込み的なデータをロードできないようにする．MFENSE 命令は LFENSE 命令と SFENSE 命令を組み合わせた命令である．この命令を実行すると，この前のロードとストアの実行が終了するまで見込み的なデータのロードとストアをできないようにする．PAUSE 命令は時間待ちループを改善するための命令である．この命令は時間待ちループの実行中のプロセッサの消費電力を軽減する効果がある．

3.4 結言

本章では IA − 32 アーキテクチャの拡張命令セットである MMX，SSE，SSE2 を調査を行った．．次の章では抽出した命令セットを基にライブラリの実装を行う．

第 4 章

ライブラリの実現法

4.1 緒言

本章では既存の命令セットを FIS，DADT に実装する方法を記述する

4.2 FIS の導入

DDMP が実装している命令は種類ごとに異なるので，同じプログラムを作る際にも，命令の違いを考慮しなければならない．それはライブラリを作成する際も同様である．そこで，ライブラリの作成には FIS(Fundamental Instruction Set) を使用する．FIS は使用頻度の高い命令群を抽象化したハードウェアに非依存な命令セットである．FIS でライブラリを作成することにより，プロセッサの違いによる命令の変更する時間を軽減することができ，ライブラリの再利用性の向上を図れる．

4.3 データ駆動型プログラムへの変換

まず，前章で検討を行い，実装を必要と決定した MMX 命令を FIS，DADT でプログラムを作成し，モジュール化を行う．MMX 命令はレジスタ同士で演算を行うわけだが，その方法では動的データ駆動処理方式の利点を損なってしまう．そこでプログラムではレジスタ構成をパケットに見立てて，実装を行っていくことにする．さらに FIS のパケット・データ長は 32 ビットなので，その範囲で実装を行う．

4.4　マルチメディア処理用ライブラリ

マルチメディア向けライブラリとしてまず複素加算と複素乗算 (図 4.1 参照) のモジュール化を行い，ライブラリ情報の登録を行った．登録したデータを以下に示す．

- 入力データ
- 出力データ
- ノード数
- クリティカルパス

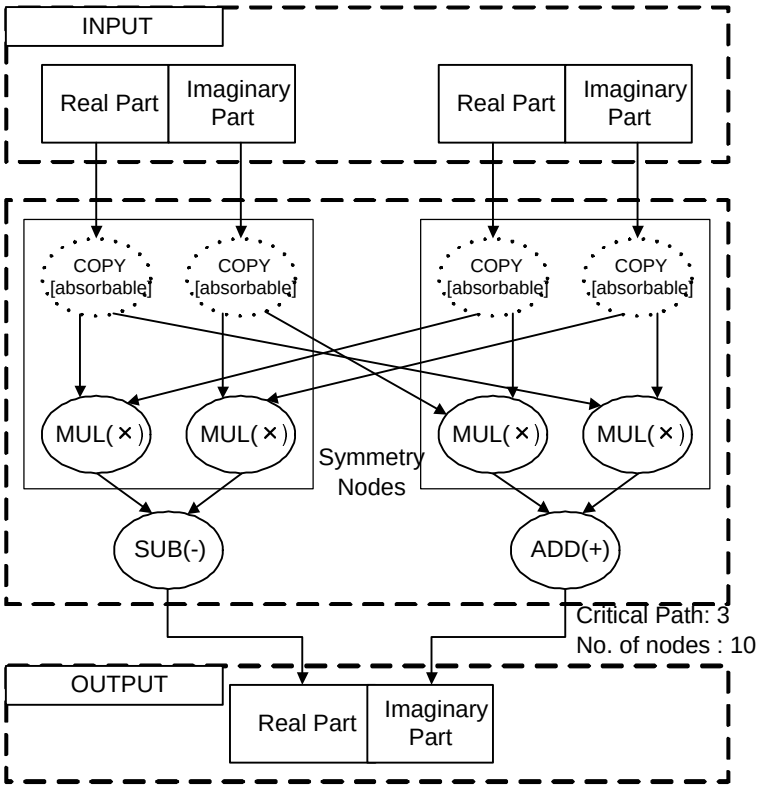


図 4.1　ライブラリ情報の例：複素乗算モジュール

4.5　DDMP 向けマルチメディアライブラリ

前章の既存拡張命令セットから将来的に DDMP に実装する必要があるかどうかを検討し，その結果を表にまとめた．表 4.1 － 3 は，それぞれ命令の機能，命令セットの種類，ニーモニックおよび DDMP に実装する必要性とその理由を記載している．さらに，ライブラリ検索用としてデータ型，オペレーションモードを組み込み，命令セット別にソートしたライブラリ表付録 A に記載する．

抽出した 174 命令中，MMX 命令のデータ範囲外の命令を除いた結果，37 命令をライブラリとして FIS に実装を行った．

4.6　結言

本章では MMX 命令セットを FIS，DADT でそれぞれ実装し，モジュール化する作業を行った．次の章では実際にアプリケーションに適用することで性能評価を行う．

カテゴリ	命令セット	命令名	実装の必要性と理由
加算	SSE	ADDPS ADDSS	一つの命令で複数を演算できる ことからノード数を削減する効果 が期待できるので導入の必要 性がある
	SSE2	ADDPD ADDSD	
	MMX	PADDB PADDW PADDD PADDSB PADDSW PADDUSB PADDUSW	
減算	SSE	SUBPS SUBSS	
	SSE2	SUBPD SUBSD	
	MMX	PSUBB PSUBW PSUBD PSUBSB PSUBSW PSUBUSB PSUBUSW	
乗算	SSE	MULPS MULSS	
	SSE2	MULPD MULSD	
	MMX	PMULL PMULH	
乗算および加算	MMX	PMADD	
除算	SSE	DIVPS DIVSS	
	SSE2	DIVPD DIVSD	
逆数	SSE	RCPPS RCPSS	
平方根	SSE	SQRTPS SQRTSS	
	SSE2	SQRTPD SQRTSD	
平方根の逆数	SSE	RSQRTPS RSQRTSS	
最大値	SSE	MAXPS MAXSS	
	SSE2	MAXPD MAXSD	
最小値	SSE	MINPS MINSS	
	SSE2	MINPD MINSD	
比較	SSE	CMPPS CMPSS COMISS UCOMISS	算術演算と同様にノード数を削 減する効果が期待できるので 導入する必要性がある
	SSE2	CMPPD CMPSD COMISD UCOMISD	
一致比較 大小比較	MMX	PCMPEQB PCMPEQW PCMPGTPB PCMPGTPW PCMPGTPD	
シャッフル	SSE	SHUFPS	データの入れ替えを省略できる ので実装する必要がある
	SSE２	SHUFPD	

表 4.1 MMX，SSE，SSE2 命令表:その 1

変換	SSE	CVTPI2PS CVTSI2SS CVTPS2PI CVTTPS2PI CVTSS2SI CVTTSS2SI	データ型を変換する命令である ため，必要性がある
	SSE2	CVTPS2PD CVTPD2PS CVTSS2SD CVTSD2SS CVTPD2PI CVTTPD2PI CVTPI2PD CVTPD２DQ CVTTPD2DQ CVTDQ2PD CVTSD2SI CVTTSD2SI CVTSI２SD CVTPS2DQ CVTTPS2DQ CVTDQ2PS	
パック	MMX	PACKSSWB PACKSSDW PACKUSWB	
アンパック	SSE	UNPCKHPS UNPCKLPS	
	SSE2	UNPCKHPD UNPCKLPD	
	MMX	PUNPCKHBW PUNPCKHWD PUNPCKHDQ	
	MMX	PUNPCKLBW PUNPCKLWD PUNPCKLDQ	
整数命令	SSE	PAVGB PAVGW PEXTRW PINSRW PMAXUB PMINUB PMAXSW PMINSW PMOVMSKB PMULHUW PSADBW PSHUFW	算術演算,比較演算，などＳＳＥ、ＳＳＥ２ での整数命令なので実装の必要性はある
	SSE2	MOVDQA MOVDQU PADDQ PSUBQ PMULUDQ PSHUFLW PSHUFHW PSHUFD PSLLDQ PSRLDQ PUNPCKHQDQ PUNPCKLQDQ MOVQ2DQ MOVDQ2Q	
ステート管理	SSE	LDMXCSR STMXCSR	レジスタに関する命令なので実装する 必要はない

表 4.2 MMX，SSE，SSE2 命令表:その 2

キャッシュ制御	SSE	MOVNTQ MOVNTPS MASKMOVQ	キャッシュの概念がないため実装する必要はない
	SSE 2	CLFLUSH MOVNTDQ MOVNTPD MOVNTI MASKMOVDQU	
プリフェッチ	SSE	PREFETCH0 PREFETCH1 PREFETCH2 PREFETCH1	命令の発行機構が違うため実装する必要はない
フェンス	SSE	SFENSE	メモリのブロッキング処理である．データのアクセスを制限できるので実装の必要がある
	SSE2	LFENSE MFENSE	
	SSE2	PAUSE	データ駆動型処理であるため，不必要である
AND ANDNOT OR XOR	SSE	ANDPS ANDNPS ORPS XORPS	算術演算等と同様に，ノード削減が期待できるため導入の必要性がある
AND ANDNOT OR XOR	SSE2	ANDPD ANDNPD ORPS XORPS	
AND ANDNOT OR XOR	MMX	PAND PANDN POR PXOR	
左に論理シフト	MMX	PSLLW PSLLD PSLLQ	算術演算，比較演算と同様にノード数を削減する効果が期待できるので導入の必要性はある
右に論理シフト	MMX	PSRLW PSRLD PSRLQ	
右に算術シフト	MMX	PSRAW PSRAD	
データ転送	SSE	MOVAPS MOVUPS MOVSS MOVLPS MOVHPS MOVLHPS MOVHLPS MOVMSKPS	２世代化操作などの世代操作命令で実装する必要がある
	SSE2	MOVAPD MOVUPD MOVSD MOVLPD MOVHPD MOVMSKPD	
	MMX	MOVW MOVW MOVW MOVQ MOVQ MOVQ	
EMMS		EMMS	レジスタの状態をクリアする命令であるため，不必要

表 4.3 MMX，SSE，SSE2 命令表:その 3

第 5 章

性能評価

5.1 緒言

本章では実装した MMX 命令をプログラムで評価する方法とその結果を記述する．

5.2 FFT を用いた評価

今回の研究はマルチメディア処理用ライブラリを適用できるマルチメディアに特化したプログラムでなければならない．そこで，評価プログラムには波形分析，計測，画像や音声処理などのディジタル信号処理の各分野の基礎技術として頻繁に使用されている高速フーリエ変換 (Fast Fourier Transform:FFT[3]) を採用する．FFT は離散フーリエ変換 (Discrete Fourier Transform:DFT) の膨大な計算量をアルゴリズムを工夫することによって，大幅に削減することでより実用的にしたものである．例えば，は取得するデータ数を 8 とすると複素加算回数は 56 回，複素加算回数は 64 回の計算量を必要とする．しかし，図 5.1 のようにバタフライ演算 [4] に置き換えると，大幅に計算量を削減できる．

評価方法としては DADT でフローグラフを作成し，手作業で最適化したプログラムと MMX 命令を適用した FIS で記述し，DADT 移し変えたプログラムとで性能比較を行った．

本研究の目的はマルチメディア処理用ライブラリを拡充することによるフローグラフ記述の高次化である．評価すべき項目を下記の 3 種類にした．理由はフローグラフ記述の高次化に大きく反映される部分だからである．

- プログラム作成時間

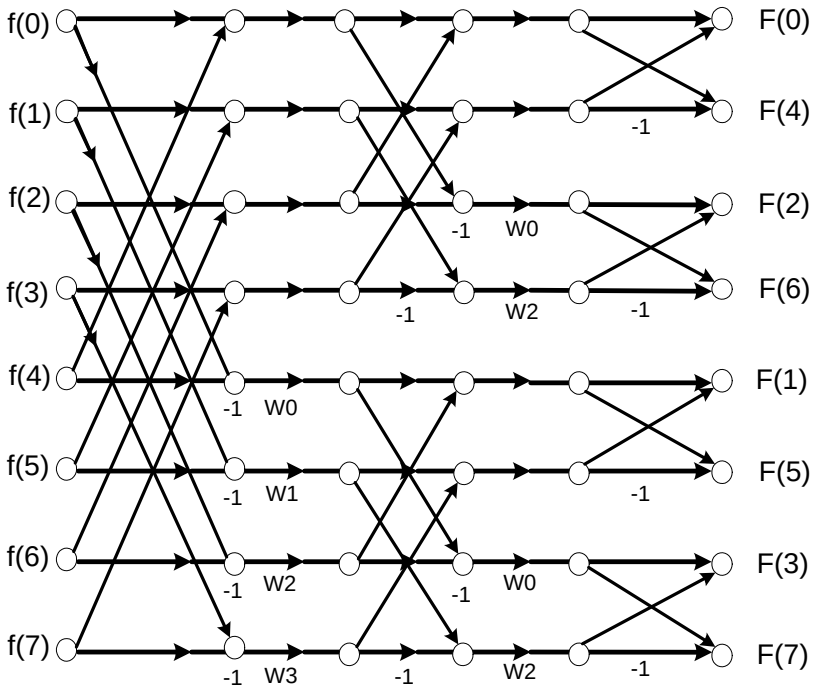


図 5.1　周波数間引き FFT の流れ図表示 (N = 8)

- スループット
- ノード数

本研究では 64 点 FFT を記述して性能評価を行い，最適化 FFT と MMX 命令適用 FFT とで，スループット，作成時間，ノード数を比較する．

5.3　評価結果

まず，評価結果を表 5.1 に示す．

表 5.1 を見ると，最適化 FFT よりも，ノード数が若干増加し，スループットは 55 %減少したが，作成時間は約 70 %削減できた．

表 5.1　評価結果

プログラム	ノード	スループット	作成時間
最適化 FFT	230	10054	50
MMX 命令適用 FFT	268	22353	15

5.4　考察と課題

ノード数が最適化 FFT より MMX 命令適用 FFT が上回ったのにはタグ操作に費やすノード数が原因と考えられる．FIS では一度タグをデータとして取り出し，操作を行ってからもう一度タグとしてデータに付与しなければならない．一方，DADT ではタグをデータとして取り出すことなくタグを操作することが可能なので，そこでノード数に差が出たということがいえる．特に FFT のようなプログラムはデータの計算よりもタグの操作の方にノードを費やしており，またタグの操作がこのプログラムの中核的部分であったことも原因の一つである．今後，世代操作などの DDMP 固有の特性を考慮し，ライブラリに反映させる課題が残されている．

5.5　結言

本章ではマルチメディア処理用ライブラリの利便性について検証した．その結果，手作業で最適化した DADT のプログラムを作成するよりも，FIS で MMX 命令を使用して記述し，DADT に機械的に置換する方法をとると，スループットは半減し，ノード数は若干増加したが,，作成時間を大幅に削減できることがわかった．

第 6 章

結論

高水準な開発環境の実現の手段の一つとしてライブラリ化がある．システムのなかで頻繁に使用される部分をライブラリ化することで，システム開発に関する冗長な部分を省略し，開発環境を向上させることができる．

本研究では，マルチメディア処理に適した自己同期型パイプラインを採用した DDMP のプログラム開発手法であるデータフローグラフ記述の高次化を目指し，既存マルチメディア命令セットを調査し，DDMP のアーキテクチャと照らし合わせた結果，抽出した命令セットをライブラリとして，ハードウェア非依存な命令セットである FIS に実装を行った．これにより，将来的に命令セットが拡張された場合でも，FIS のライブラリを改良のみで対応することができる．実装の際には，データの形態をレジスタからパケットに変更することにより，動的データ駆動型処理方式の利点を失うことなく実装することを可能にした．そして実際に FFT 演算プログラムで評価をする際に DADT に手作業で実装と最適化を行ったプログラムと MMX 命令を適用した FIS を DADT に機械的に置換したプログラムとで，スループット，ノード数，作成時間の比較を行った．その結果，マルチメディア処理用ライブラリをプログラムに適用することにより，ノード数，スループットは悪化したものの，手作業で最適化したプログラムよりも大幅にプログラム作成時間を削減できるということがわかった．

今後の課題に以下のことが挙がった．

- 世代操作を考慮したライブラリの実装
- これまでに実装したライブラリの最適化

- 作成時間に差が出た原因の更なる調査

第 5 章でも述べたとおり，MMX 命令を適用した FFT と手作業で最適化を行った FFT とでノード数をみると MMX 命令を適用した FFT の方が増加していた．その理由は，マルチメディア処理用ライブラリに世代操作を取り入れていなかったことと，FFT 演算プログラムが世代番号の操作を頻繁に行う特徴をもつアルゴリズムを持っていたことが挙げられる. このようなプログラムに対応するために，柔軟に世代番号の変更可能なライブラリの実装が必要である．

既存のマルチメディア命令セットである MMX，SSE，SSE2 についての検討は終了したもののまだ個々のライブラリの性能評価は行われてはいない．FFT 演算プログラムでは，ライブラリを 4 種類しか使用しておらず，まだ多くのライブラリに対して，定量的な評価を行い，最適化を行う必要がある．例えば，検討した命令セットではノイマン型アーキテクチャなので SIMD(Single Instruction Multiple Data) が採用されているが，データ駆動型プロセッサの並列型アーキテクチャならば，MIMD(Multiple Instruction Multiple Data) の概念をマルチメディア処理用ライブラリに組み込むことにより更なる開発環境の向上も可能であると考えられる．

本研究ではマルチメディア処理用ライブラリを拡充したことにより，大幅な計算時間を削減できた．しかし，一つのプログラムでしか検証がなされていないため削減できた原因が上記の理由だけとは必ずしも断定できない．この課題を解決するには，多岐にわたるマルチメディア処理用プログラムで検証を行い，原因を詳細に分析する必要があるだろう．

今後の展望として，次のものが挙げられる．

- ライブラリに対する資源割り当ての最適化
- 更なるマルチメディア命令セットの調査

現在，ライブラリを実装したのみでライブラリに対する資源割り当ての枠組みの検討はまだな去れていないので，資源割り当ての最適化を実現する手法を考える必要がある．

現在，検討がなされているのは MMX 命令，SSE 命令，SSE2 命令のみで SPARC 系 CPU

の VIS (Visual Instruction Set)，PA-RISC 系 CPU の MAX (Multimedia Acceleration eXtensions)，MIPS 系 CPU の MDMX (Digital Media eXtension) に対しても検討する必要がある．

謝辞

本研究を行うにあたり，ご懇篤なご指導とご鞭撻を賜りました岩田 誠 教授に心より感謝いたします.

本研究を行うにあたり，お忙しい中，貴重なご意見を賜りました大森 洋一 助手に深く感謝します.

本研究の核としているデータ駆動型アーキテクチャを提唱された寺田 浩詔 教授に心より感謝の意を表します.

本研究論文の副査をお引き受け頂きました，岡田 守 教授，竹田 史章 教授に心より感謝します.

岩田研究室において，温かいご支援，並びにご助言を頂きました大学院生の林 秀樹 氏，橋本 正和氏，別役 宣奉氏，森川 大智氏，小倉 通寛氏，三宮 秀次氏，志摩 浩氏，
中村 勲二 氏に，心より感謝します.

日頃から，研究室の同輩として温かいご支援を頂きました，荒木 俊介 氏，岩井 秀樹 氏，大石 裕子 氏，そね田 紘貴 氏，西山 直人 氏，宮崎 康徳 氏に心より感謝します.

参考文献

[1] H. Terada, et al., “ DDMP’s: self-timed super-pipelined data-driven multimedia processors, ” Proc. of IEEE, 87(2), 282-296, 1999.

[2] 三宮他, “ シミュレーション対象のハードウェア非依存な命令セット FIS の提案, ” 平成 13 年度電気関係学会四国連合大会.

[3] 佐川雅彦，貴家仁志，辻井重雄 “ ディジタル信号処理シリーズ 第 2 巻 高速フーリエ変換とその応用, ” 株式会社昭光堂, April 1996.

[4] 松尾 博, “ やさしいフーリエ変換, ” 森北出版株式会社, September 1994.

[5] 株式会社インテル, “ IA-32 インテルアーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル. ”

付録 A

マルチメディア処理用ライブラリ表

命令セット	データ型	演算モードまたは形式	命令名
SSE	単精度浮動小数点数		ADDPS ADDSS
SSE2	倍精度浮動小数点数		ADDPD ADDSD
MMX	パックド・データ	ラップアラウンド演算 符号付き飽和演算 符号なし飽和演算	PADDB PADDW PADDD PADDSB PADDSW PADDUSB PADDUSW
SSE	単精度浮動小数点数		SUBPS SUBSS
SSE2	倍精度浮動小数点数		SUBPD SUBSD
MMX	パックド・データ	ラップアラウンド演算 符号付き飽和演算 符号なし飽和演算	PSUBB PSUBW PSUBD PSUBSB PSUBSW PSUBUSB PSUBUSW
SSE	単精度浮動小数点数		MULPS MULSS
SSE2	倍精度浮動小数点数		MULPD MULSD
MMX	パックド・データ	ラップアラウンド演算	PMULL PMULH
MMX	パックド・データ	ラップアラウンド演算	PMADD
SSE	単精度浮動小数点数		DIVPS DIVSS
SSE2	倍精度浮動小数点数		DIVPD DIVSD
SSE	単精度浮動小数点数		RCPPS RCPSS
SSE	単精度浮動小数点数		SQRTPS SQRTSS
SSE2	倍精度浮動小数点数		SQRTPD SQRTSD
SSE	単精度浮動小数点数		RSQRTPS RSQRTSS

表 A.1　ライブラリ検索用命令表:その 1

SSE	単精度浮動小数点数		MAXPS MAXSS
SSE2	倍精度浮動小数点数		MAXPD MAXSD
SSE	単精度浮動小数点数		MINPS MINSS
SSE2	倍精度浮動小数点数		MINPD MINSD
SSE	単精度浮動小数点数		CMPPS CMPSS COMISS UCOMISS
SSE2	倍精度浮動小数点数		CMPPD CMPSD COMISD UCOMISD
MMX	パックド・データ	ラップアラウンド演算	PCMPEQB PCMPEQW
	パックド・データ	ラップアラウンド演算	PCMPGTPB PCMPGTPW PCMPGTPD
SSE	単精度浮動小数点数		SHUFPS
SSE 2	倍精度浮動小数点数		SHUFPD
SSE			CVTPI2PS CVTSI2SS CVTPS2PI CVTTPS2PI CVTSS2SI CVTTSS2SI
SSE2			CVTPS2PD CVTPD2PS CVTSS2SD CVTSD2SS CVTPD2PI CVTTPD2PI CVTPI2PD CVTPD 2 DQ CVTTPD2DQ CVTDQ2PD CVTSD2SI CVTTSD2SI CVTSI 2 SD CVTPS2DQ CVTTPS2DQ CVTDQ2PS
MMX	パックド・データ	符号付き飽和演算 符号なし飽和演算	PACKSSWB PACKSSDW PACKUSWB
SSE			UNPCKHPS UNPCKLPS
SSE2			UNPCKHPD UNPCKLPD
MMX	パックド・データ	ラップアラウンド演算	PUNPCKHBW PUNPCKHWD PUNPCKHDQ
MMX	パックド・データ	ラップアラウンド演算	PUNPCKLBW PUNPCKLWD PUNPCKLDQ
SSE			PAVGB PAVGW PEXTRW PINSRW PMAXUB PMINUB PMAXSW PMINSW PMOVMSKB PMULHUW PSADBW PSHUFW

表 A.2　ライブラリ検索用命令表:その 2

SSE2			MOVDQA MOVDQU PADDQ PSUBQ PMULUDQ PSHUFLW PSHUFHW PSHUFD PSLLDQ PSRLDQ PUNPCKHQDQ PUNPCKLQDQ MOVQ2DQ MOVDQ2Q
SSE			LDMXCSR STMXCSR
SSE			MOVNTQ MOVNTPS MASKMOVQ
SSE 2			CLFLUSH MOVNTDQ MOVNTPD MOVNTI MASKMOVDQU
SSE			PREFETCH0 PREFETCH1 PREFETCH2 PREFETCH1
SSE			SFENSE
SSE2			LFENSE MFENSE
SSE2			PAUSE
SSE	単精度浮動小数点数		ANDPS ANDNPS ORPS XORPS
SSE2	倍精度浮動小数点数		ANDPD ANDNPD ORPS XORPS
MMX	パックド・データ	クワッドワード全体	PAND PANDN POR PXOR
MMX		パックド・データ クワッドワード全体	PSLLW PSLLD PSLLQ
MMX		パックド・データ クワッドワード全体	PSRLW PSRLD PSRLQ
MMX		パックド・データ	PSRAW PSRAD
SSE	単精度浮動小数点数		MOVAPS MOVUPS MOVSS MOVLPS MOVHPS MOVLHPS MOVHLPS MOVMSKPS
SSE2	倍精度浮動小数点数		MOVAPD MOVUPD MOVSD MOVLPD MOVHPD MOVMSKPD
MMX	パックド・データ	ダブルワード転送 クワッドワード転送	MOVW MOVW MOVW MOVQ MOVQ MOVQ
			EMMS

表 A.3　ライブラリ検索用命令表:その 3