特別研究報告書

距離画像生成処理における
メディアプロセッサの評価

指導教官　　富田 眞治 教授

京都大学工学部情報学科

清水 雄歩

平成 15 年 2 月 3 日

距離画像生成処理における
メディアプロセッサの評価

清水 雄歩

内容梗概

近年、計算機の性能が急速に向上しているものの、膨大な計算量が必要な画像処理にはまだまだ十分とは言えない。このため、マルチプロセッサや専用ハードウェアを用いた大規模なシステムが提案されている。一方、大規模な専用システムは、開発のために時間とコストを必要とするだけでなく、大量の電力を消費するという問題点がある。特に、携帯電話やデジタルカメラ等のモバイル製品や、監視カメラへの応用を考えた場合、電源装置の小型化やファンに頼らない自然空冷が不可欠であり、低消費電力化が重要な課題である。

以上の背景から、本論文では、画像処理において低消費電力かつ高速に処理を行うことのできる汎用プロセッサの利用が今後重要になると考え、特に、自動監視システムに必要となるステレオ距離画像生成処理をアプリケーションプログラムとして、富士通製 VLIW 型プロセッサ FR-V やメディア演算命令を搭載した汎用マイクロプロセッサを用いて距離画像生成を高速化する研究を行った。VLIW アーキテクチャは命令のスケジューリングをコンパイラが行うため、スケジューリングをハードウェアで行うスーパースカラアーキテクチャより回路規模を小さくでき、消費電力も少ないことから、モバイルシステムに適していると予想した。

さて、距離画像生成処理は、ステレオ画像において、物体の視差、すなわち左右の画像における対応点の位置の違い (disparity) を測ることによって、その物体までの距離を求めることを基本としている。左右の画像から対応点を見つけるために用いられるアルゴリズムに SSD (Sum of Squared Difference) がある。これは、対応点では、周囲の画素値の差が 0 に近いという性質を利用したアルゴリズムである。一方、より計算量を減らすためのアルゴリズムとして、画素差の絶対値の総和を用いる SAD (Sum of Absolute Difference) がある。本論文では、SAD を採用し、距離画像生成処理の大部分を占める比較処理をメディア演算命令を用いて高速化することにした。

メディア演算命令を用いた比較処理は、各プロセッサに搭載されている命令セットの種類によって異なるものの、概ね次のように実現することができる。ま

ず、画素データはメモリ上に配列として格納されている。左右のスモールウィンドウから 1 ピクセル分のデータをレジスタにロードし、次に画素差の絶対値を計算し、総和を求めていく。Intel IA-32 プロセッサには、SSE および SSE2 と呼ばれるストリーミング SIMD 拡張命令セット、また、SPARC プロセッサには、VIS (Visual Instruction Set) と呼ばれるメディア演算命令セットが定義されており、いずれも差分の絶対値総和を直接求めることができる。一方、富士通製 VLIW 型プロセッサ FR-V には、このような命令は定義されていない。そこで、FR-V では、飽和付き加減算命令を組み合わせることにより高速化を図った。すなわち、例えば $|a-b|$ を求めるとき、まず、$a-b$ と $b-a$ の各飽和付き減算結果を求める。飽和減算では減算結果が負の数になると 0 に飽和するため、それらを加算すると差の絶対値 $|a-b|$ が求まる。8 並列実行型 VLIW である FR550 は、整数演算命令およびメディア演算命令をそれぞれ 4 命令並列実行できるものの、ロード命令や一部のメディア演算命令は 2 個までしか同時発行できないため、4 並列実行型 VLIW である FR500 のコードをそのまま 8 並列に拡張することはできない。そこで新たな工夫として、加算にメディアクワッド積和演算命令を用いた。飽和減算命令により求めた結果を、飽和加算命令と積和演算命令により同時に累積する。また、VLIW を効率よく利用できる命令スケジューリングも行い、最大 6 命令を同時実行するようにコーディングを行った。

以上の高速化手法を用いて、距離画像生成処理における各プロセッサの性能および消費電力を評価した。この結果、どのプロセッサにおいても、メディア演算命令を用いることによって距離画像生成処理能力が飛躍的に向上することがわかった。また、性能と消費電力の両方を考慮する評価尺度 ($fps/Watt$) を用いて比較した結果、予想通り、FR550 が最も優れていることがわかった。さらに、FR550 に対して絶対差の総和を求める命令を追加することにより、ステップ数比で約 32%、命令並列度の改善をほどこすことにより、ステップ数比で約 52% 性能を向上できることを示した。

Evaluation of Media Processors with Stereo Depth Extraction

Yuho SHIMIZU

Abstract

Recently, the performance of computers has been increasing rapidly, but it is not enough for image processing which needs a huge amount of calculation. Therefore, the large-scale systems using a multiprocessor or propriety hardware are proposed to achieve real time image processing. On the other hand, large-scale propriety systems spend not only development time and cost, but also having a problem on consuming high power consumption. Especially, when we want to apply these applications to power-aware products, such as cellular phones, digital cameras, or surveillance cameras, the miniaturization of power supply equipment and the passive air cooling system which does not require a fan are indispensable. So reducing power consumption is an important subject.

Considering these backgrounds, I thought that the general-purpose processor with powerful media processing features becomes important in the future, because it can process images with low power consumption and at high performance. To establish my prediction, I researched techniques which accelerate a stereo depth extraction, using Fujitsu VLIW processor FR-V and general-purpose microprocessors which have media instructions. A VLIW architecture can be expected low power consumption, becase in this case, a compiler performs scheduling of operations so, a circuit scale will be smaller and it uses less power than a superscalar architecture which performs scheduling by hardware. The reason why I picked up the stereo depth extraction as the target application, is that this appliocation is needed for an automatic surveillance system and such system is expected to use widely in real world.

The stereo depth extraction is based on detecting the distance to an object in the stereo images by measuring the disparity of the corresponding points. The SSD (Sum of Squared Difference) algorithm is used in order to find the corresponding points from images on either side. This algorithm uses a property that the difference of surrounding pixel value is close to zero at the corresponding points. However the SSD requires huge amount of calculation, so, there is

another algorithm called the SAD (Sum of Absolute Difference) which uses a sum of the absolute difference of pixel value to reduce the amount of calculation. In this paper, using the SAD algorithm, I accelerate a comparison process which occupies most of a stereo depth extraction with media instructions. Although comparison processing using media instructions varies from a kind of instruction sets carried in each processor, it can be realized almost as follows. First, pixel data is stored as an arrangement on the memory. The data of one pixel is loaded to the register from small windows on either side, next absolute values of pixel difference is calculated and accumulated. The SSE and SSE2 (streaming SIMD extended instruction sets) on Intel IA-32 processors, the VIS (Visual Instruction Set) on SPARC processors, can calculate sum of absolute difference directly. On the other hand, on FR-V processors, there are no such instructions are not defined, so, I attempted to process SAD quickly by combining the addition/subtraction operation with saturation. The FR550, the highest version of FR-V, can execute 8-parallel in one VLIW instruction, but there is restriction that addition/subtraction operation with saturation can be issued only four of eight slots. The idea is that multiply and accumulate operation is also used for addition operation. Moreover, operational scheduling which can use VLIW efficiently is also performed, and coding was performed so that six operations might be executed at the maximum.

Using above techniques for high speed processing, I evaluated the performance and power consumption of each processors in a stereo depth extraction. As a result, the performance of stereo depth extraction improves dramatically by using media instructions in any processors. Moreover, as a result of comparing using the evaluation measure ($fps/Watt$) in consideration of both the performance and power consumption, FR550 is the most excellent as expected. Also, by adding an operation that directly calculates a sum of absolute difference to FR550, the performance can be improved about 32% by the number of steps. And by improving the number of parallel operations, the performance can be improved about 52% by the number of steps.

距離画像生成処理における
メディアプロセッサの評価

目次

第 1 章　はじめに　1

1.1　研究の背景 1

1.2　関連研究 1

第 2 章　距離画像生成アルゴリズム　3

2.1　視差と距離の関係 3

2.2　対応点探索アルゴリズム 3

2.3　処理の流れ 4

第 3 章　メディア演算命令を用いた高速化手法　6

3.1　一般的な高速化手法 6

3.2　FR-V における高速化 7

3.2.1　FR500 7

3.2.2　FR550 9

3.3　Intel IA-32 における高速化 13

3.3.1　MMX, SSE 13

3.3.2　SSE2 15

3.4　SPARC における高速化 17

第 4 章　評価　18

4.1　評価システム 18

4.2　測定結果および考察 19

第 5 章　おわりに　23

謝辞　23

参考文献　24

第1章　はじめに

1.1　研究の背景

近年、計算機の性能が急速に向上しているものの、膨大な計算量が必要な画像処理にはまだまだ十分とは言えない。例えば、自動監視システムでは人の動きをリアルタイムに把握できなければならない。しかし、現在の汎用計算機を用いてこのようなシステムを実現することは困難であり、マルチプロセッサや専用ハードウェアを用いた専用並列処理システムが提案されている。

一方、専用ハードウェアを用いたシステムの開発には多くの時間とコストを必要とするため、ソフトウェアによる処理が望まれている。また、携帯電話やデジタルカメラ等のモバイル製品を小型化するためには、電源装置の小型化やファンに頼らない自然空冷が不可欠であり、プロセッサの低消費電力化が重要な課題となっている。

以上の背景から、本論文では、画像処理を低消費電力かつ高速に行う汎用プロセッサが今後重要になると考え、画像処理における汎用プロセッサの性能について定量的評価を行った。アプリケーションプログラムには、自動監視システムに必要となるステレオ距離画像生成処理を選択した。この処理は、ステレオ画像から、距離画像、すなわち物体までの距離が近い部分ほど白く、遠い部分ほど黒く表示した画像を生成するものである。単に距離を測るためだけであればレーザー等による能動的手法により実現できるものの、多くの電力を必要し、また、レーザーを受ける対象に影響を及ぼす可能性があることから、使用できる場合が限られる。一方、ステレオ画像を用いる手法では、このような問題が生じない。

1.2　関連研究

実際の映像とコンピュータグラフィックスを合成する技術に、Chroma keying がある。これは、背景を青色にして人を撮影し、青色の部分をコンピュータグラフィックスで置き換えるというものである。Chroma keying では、現実映像は常に仮想映像の前面にあることが前提である。一方、Carnegie Mellon 大学では、1994 年頃から CMU Video-Rate Stereo Machine と呼ばれるシステムの研究 [1] を行っており、ステレオアルゴリズムを応用して、15 fps (frames per second) の画像を生成できる Z Keying のシステムが開発されている。Z Keying は現実

映像と仮想映像をより柔軟に融合させる技術である。色度 (chromaticity) の代わりに距離情報を利用し、現実映像と仮想映像を比較して、近いものを表示することにより、Chroma keying を用いたシステムよりも複雑な映像を作成することができる。

SONY が開発した Entertainment Vision Sensor は、カラー動画像および 3 次元距離情報を、15 fps または 30 fps で取得可能な高機能 CMOS イメージセンサーである [2]。動画像と距離情報を同時に利用することにより、画像認識や動き検出など、これまで大規模なシステムにより実現していた機能を小規模なシステムにより実現することができる。このシステムは、距離情報取得のために光切断法を使用している。光切断法とは、カメラの光軸と角度を持たせて照射したスリット状のレーザー光がどの角度に見えるかに基づいて距離を測定するものである。レーザー光とカメラによる三角測量を基本とするため、カメラ 1 台のみによる測定が可能である。

CANESTA が開発したセンサーチップは、レーダーと同様の原理に基づいており、目には見えない光の点滅を用いて、対象からの光の到達時間を測定することにより、距離を求めている [3]。処理速度は 50 fps である。

TYZX が開発したシステムは、距離測定のための視差測定専用プロセッサである [4, 5]。特に、DeepSea は 7×7 ピクセルの範囲を比較することにより、左右画像の対応点を調べている。カメラの個体差や製造ばらつきがあるため、画素の RGB 値を直接比較するのではなく、基準点 (中心点) に対して他の画素が明るいか暗いかの 2 値ベクトルを用いた比較を行っている。

以上のように、従来の研究では専用プロセッサを用いて距離計算を行っている。しかし、これらのシステムは将来、高速な汎用プロセッサによって、より安価に実現できると考えられる。そこで本論文では、富士通製 VLIW 型プロセッサ FR-V やメディア演算命令を搭載した汎用マイクロプロセッサを用いて距離画像生成を高速化する研究を行った。特に VLIW アーキテクチャは命令のスケジューリングをコンパイラが行うため、スケジューリングをハードウェアで行うスーパースカラアーキテクチャより回路規模を小さくでき、消費電力も少ないことが予想されることから、モバイルシステムに適していると考えている。

本稿では、FR-V や汎用マイクロプロセッサのメディア演算命令を用いた場合の距離画像生成について、処理速度や消費電力の観点から比較を行い、考察を加える。

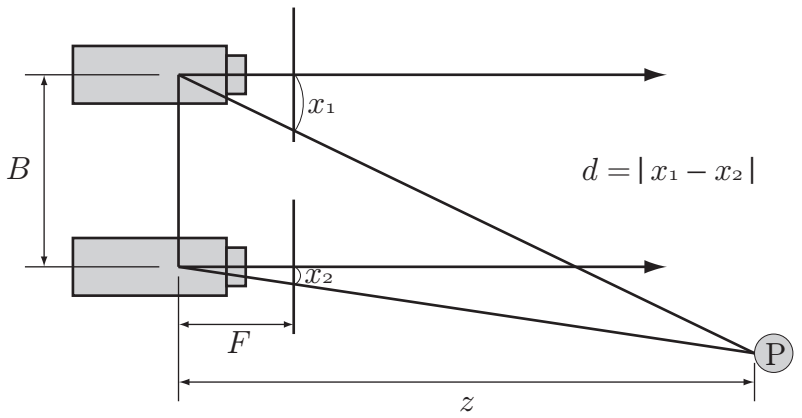

図 1: ステレオカメラにおける視差と距離の関係

第 2 章　距離画像生成アルゴリズム

2.1　視差と距離の関係

ステレオ画像において、物体の視差、すなわち左右の画像における対応点の位置の違い (disparity) を測ることによって、その物体までの距離を求めることができる。図 1 に、ステレオカメラにおける視差と距離の関係を示す。平行に置いた 2 台のカメラにより、ある物体 P を撮影したとき、左右の画像における P の x 座標をそれぞれ $x_1,\ x_2$ とすると、視差 d は、

$$d = |x_1 - x_2| \tag{1}$$

である。また、B をベースライン距離 (カメラ間の距離)、F をカメラの焦点距離とすると、d とカメラから P までの距離 (distance) z には次の関係が成り立つ [6]。

$$d = BF\frac{1}{z} \tag{2}$$

すなわち、ステレオ画像における視差から、物体までの距離を求めることができる。

2.2　対応点探索アルゴリズム

視差を求めるためには、左右の画像から対応点を見つけなければならない。よく用いられるアルゴリズムに SSD (Sum of Squared Difference) がある。これは、対応点では、周囲の画素値の差が 0 に近いという性質を用いたアルゴリズムである。図 2 に、ステレオ画像と視差の関係を示す。基準となる画像の点 A

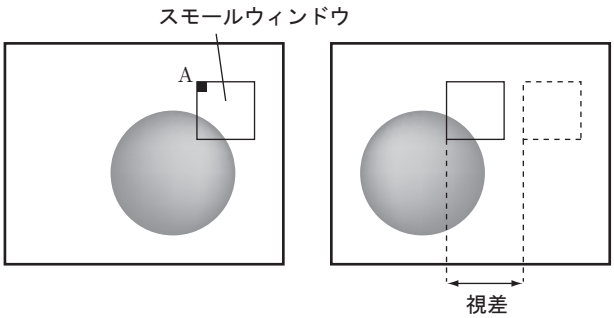

図 2: ステレオ画像と視差

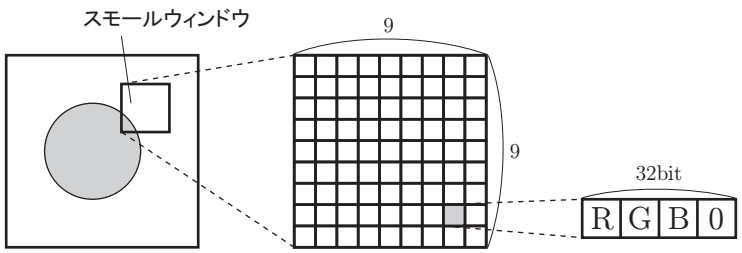

図 3: スモールウィンドウ

の周囲にスモールウィンドウを設け、他方のスモールウィンドウとの間の各画素値の差の自乗和を計算する。スモールウィンドウを動かしながらこの計算を行い、画素差の自乗和が最小となる点を探索する。対応点の x 座標の差が視差となる。ただし、スモールウィンドウごとに画素差の自乗和を計算するため、膨大な計算量が必要となる。

より計算量を減らすためのアルゴリズムとして、画素差の絶対値の総和を用いる SAD (Sum of Absolute Difference) がある。なお、いずれのアルゴリズムにおいても、スモールウィンドウが大きいほど正確な距離画像が得られるものの、計算量は二次関数的に増加する。

2.3 処理の流れ

本論文では、スモールウィンドウのサイズを 9×9 ピクセルとする SAD を用いた。また、各画素値は上位 24 ビットを RGB 各 8 ビット、下位 8 ビットを 0 とする 32 ビットとした (図 3)。

以下に、距離画像生成処理の手順を示す。

(1) ぼかし

ノイズがあると、(2) の輪郭検出処理においてノイズを輪郭とみなすこと

があるため、2 台のカメラからのステレオ画像をぼかす。

(2) 輪郭検出

SAD では、輪郭を含まない、同じような色からなるスモールウィンドウが連続する場合、対応点を検出することができない。これは、正しい対応点以外のスモールウィンドウに対しても SAD の値が最小となりうるためである。誤認識を防ぐために、まず、左右の画像において輪郭を検出し、輪郭部分における視差を求めることにした。

(3) ノイズ除去

左右の輪郭画像からノイズを除去する。

(4) 距離計算

距離画像生成処理の大部分を占める。基準となる一方の画像において 9×9 ピクセルのスモールウィンドウを仮定し、他方の画像についても同じ座標にスモールウィンドウを仮定する。左画像のスモールウィンドウは右に、右画像のスモールウィンドウは左に動かしながら、スモールウィンドウ中の 81 ピクセルについて各画素差の絶対値の総和を求める。総和が最小、かつ、その点が輪郭を含む場合、対応点とみなし、視差 (x 座標の差) を記録する。そうでなければ最近輪郭とみなした点から視差を補完する。これを全ての点について繰り返す。また、この操作を左右の画像に対して行い、2 枚の距離画像を生成する。

(5) 距離画像の合成

2 枚の距離画像を比較し、各画素について、距離の遠い方を最終的な距離として採用する。

(6) ノイズ除去

距離画像からノイズを除去する。

(7) 補完

輪郭が不明瞭なために生じる、くしの歯状の誤差を補正するために、上下の距離画像を元にして補完する。

図 4 からわかるように、距離画像生成では、スモールウィンドウの比較処理が大部分 (約 97%) を占める。

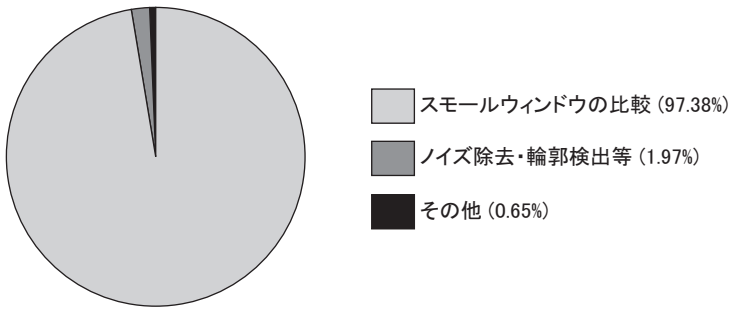

図 4: 距離画像生成処理におけるスモールウィンドウの比較の割合

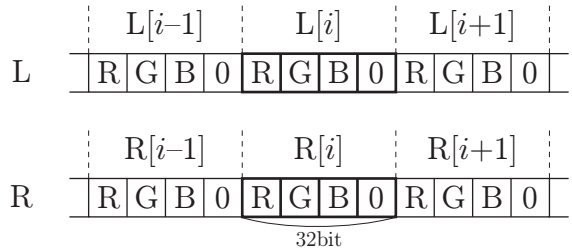

図 5: メモリ上の画像データ配列

第 3 章　メディア演算命令を用いた高速化手法

3.1　一般的な高速化手法

距離画像生成処理の大部分を占める比較処理をメディア演算命令を用いて高速化することにした。

メディア演算命令を用いた比較処理は、各プロセッサに搭載されている命令セットの種類によって異なるものの、概ね以下のように実現することができる。

図 5 に示すように、画素データはメモリ上で配列として保持されている。図 6 に、一般的なカーネルコードを示す。左右のスモールウィンドウから 1 ピクセル分のデータ $\mathrm{L}[i]$, $\mathrm{R}[i]$ を LOAD 命令でレジスタ L_i, R_i にロードし、DIFF 命令で RGB 成分ごとに画素差の絶対値を計算し、その和 D_i を求め、D_i を ADD 命令でレジスタ S に累積する。ここで、DIFF 命令は、左画像における画素の R 成分を L_R 等と書くと、

$$D_i = |L_R - R_R| + |L_G - R_G| + |L_B - R_B| \tag{3}$$

を直接求める命令である。この命令が搭載されていないプロセッサの場合は、他の命令を使用して画素差の絶対値を求める必要がある。

```
LOAD  L[i] → Li
LOAD  R[i] → Ri
DIFF  Li, Ri → Di
ADD   S + Di → S
```

図 6: 一般的なカーネルコード

3.2 FR-V における高速化

FR-V には、整数演算命令セットとともに、画像処理や MPEG ビデオに適したメディア演算命令セット、オーディオ信号の圧縮・伸張や座標計算に適した浮動小数点演算命令セット、信号処理用の DSP 演算命令セット、カスタマイズ可能なユーザ定義命令セット等が搭載されており、適切な命令セットを選択し、VLIW の枠組みを用いて並列実行させることができる。

評価に用いたのは、FR500 (4 並列実行型 VLIW) および FR550 (8 並列実行型 VLIW) である。

3.2.1 FR500

FR500 は整数演算命令と、浮動小数点演算命令またはメディア命令をそれぞれ最大 2 個同時実行することができる [7]。

FR-V のメディア演算命令では、画素値の各 8 ビットの差の絶対値を直接求めることはできない。しかし、メディアクワッド 飽和付き加算/減算命令 MQADDHUS/MQSUBHUS (Quad Add/Subtract Unsigned Halfword with Saturation) を工夫して使うことにより、差の絶対値を求めることができる。例えば、$|a-b|$ を求めるとき、$a-b$ と $b-a$ の各飽和付き減算結果を求める。飽和減算では減算結果が負の数になると 0 に飽和するため、それらを加算すると差の絶対値 $|a-b|$ が求まる。MQSUBHUS は 1 命令で同時に 4 個の減算ができ、2 命令が並列実行できるので、1 ステップで 8 個の減算ができる。9×9 のスモールウィンドウの 1 行に対する比較処理を、メディア演算命令を使用してハンドコーディングした。

図 7 に、1 ピクセルどうしの比較処理の流れを示す。左右のスモールウィンドウから 1 ピクセル分の 32 ビットデータ $\mathrm{L}[i]$, $\mathrm{R}[i]$ を LDFI 命令によりレジスタ L_i, R_i にロードし、MBTOH 命令により各 8 ビットの RGB 値を 16 ビットに拡張し、左右それぞれの画素データをレジスタ F_i, F_{i+1} と G_i, G_{i+1} に格納する。

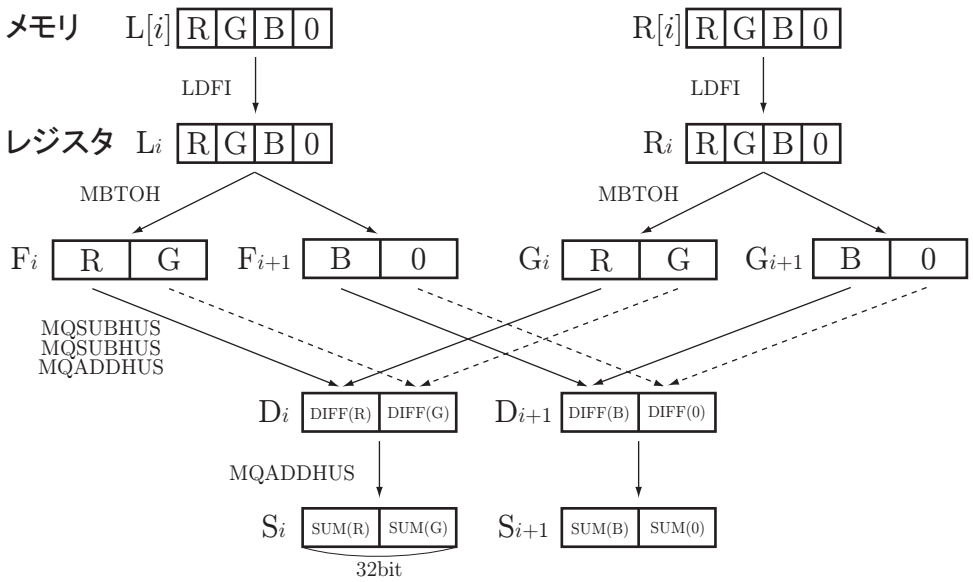

図 7: FR500 を用いた場合の比較処理の流れ

DIFF(R) は、R 成分について MQSUBHUS と MQADDHUS により飽和演算を行い、

$$\mathrm{DIFF(R)} = |\mathrm{F}_{i\mathrm{R}} - \mathrm{G}_{i\mathrm{R}}| \quad (4)$$

$$= (\mathrm{F}_{i\mathrm{R}} - \mathrm{G}_{i\mathrm{R}}) + (\mathrm{G}_{i\mathrm{R}} - \mathrm{F}_{i\mathrm{R}}) \quad (5)$$

を求めることを意味する。G, B 成分についても同様に求め、結果をアキュムレート用レジスタ S_i, S_{i+1} に累積する。

各命令の動作は次の通りである [8]。

LDFI Load FR register (Immediate). メモリ $\mathrm{L}[i]$ から FR レジスタ L_i へ 4 バイトデータを転送する。FR レジスタは、浮動小数点演算やメディア演算で用いる 32 ビットレジスタである。

MBTOH Byte To Halfword. FR レジスタ L_i 上の 32 ビット値を各 8 ビットずつ区切り、16 ビット値 4 つに変換し、2 個の FR レジスタ F_i, F_{i+1} に格納する。

MQSUBHUS Quad Subtract Unsigned Halfword with Saturation. 2 個の FR レジスタを用い、F_i, F_{i+1} と G_i, G_{i+1} 間において 4 つの 16 ビット整数の飽和付き SIMD 減算を行う。

MQADDHUS Quad Add Unsigned Halfword with Saturation. 2 個の FR レジスタを用い、F_i, F_{i+1} と G_i, G_{i+1} 間において 4 つの 16 ビット

LDFI L[i]→Li	LDFI L[i+1]→Li+1	MBTOH Li+2→Fi,Fi+1	MBTOH Li+3→Fi+2,Fi+3
LDFI R[i]→Ri	LDFI R[i+1]→Ri+1	MBTOH Ri+2→Gi,Gi+1	MBTOH Ri+3→Gi+2,Gi+3
		MQSUBHUS Fi - Gi→X	MQSUBHUS Fi+2 - Gi+2→Z
		MQSUBHUS Gi - Fi→Y	MQSUBHUS Gi+2 - Fi+2→U
		MQADDHUS X + Y→Di	MQADDHUS Z + U→Di+2
		MQADDHUS Di + Si→Si	MQADDHUS Di+2 + Si+2→Si+2
MBTOH Li→Fi,Fi+1	MBTOH Li+1→Fi+2,Fi+3	LDFI L[i+2]→Li+2	LDFI L[i+3]→Li+3
MBTOH Ri→Gi,Gi+1	MBTOH Ri+1→Gi+2,Gi+3	LDFI R[i+2]→Ri+2	LDFI R[i+3]→Ri+3
MQSUBHUS Fi - Gi→X	MQSUBHUS Fi+2 - Gi+2→Z		
MQSUBHUS Gi - Fi→Y	MQSUBHUS Gi+2 - Fi+2→U		
MQADDHUS X + Y→Di	MQADDHUS Z + U→Di+2		
MQADDHUS Di + Si→Si	MQADDHUS Di+2 + Si+2→Si+2		
LDFI L[i+4]→Li+4	LDFI L[i+5]→Li+5	MBTOH Li+2→Fi,Fi+1	MBTOH Li+3→Fi+2,Fi+3
LDFI R[i+4]→Ri+4	LDFI R[i+5]→Ri+5	MBTOH Ri+2→Gi,Gi+1	MBTOH Ri+3→Gi+2,Gi+3
		MQSUBHUS Fi - Gi→X	MQSUBHUS Fi+2 - Gi+2→Z
		MQSUBHUS Gi - Fi→Y	MQSUBHUS Gi+2 - Fi+2→U
		MQADDHUS X + Y→Di	MQADDHUS Z + U→Di+2
		MQADDHUS Di + Si→Si	MQADDHUS Di+2 + Si+2→Si+2
MBTOH Li+4→Fi,Fi+1	MBTOH Li+5→Fi+2,Fi+3	LDFI L[i+6]→Li+6	LDFI L[i+7]→Li+7
MBTOH Ri+4→Gi,Gi+1	MBTOH Ri+5→Gi+2,Gi+3	LDFI R[i+6]→Ri+6	LDFI R[i+7]→Ri+7
MQSUBHUS Fi - Gi→X	MQSUBHUS Fi+2 - Gi+2→Z		
MQSUBHUS Gi - Fi→Y	MQSUBHUS Gi+2 - Fi+2→U		
MQADDHUS X + Y→Di	MQADDHUS Z + U→Di+2		
MQADDHUS Di + Si→Si	MQADDHUS Di+2 + Si+2→Si+2		
LDFI L[i+8]→Li+8	LDFI L[i+9]→Li+9	MBTOH Li+6→Fi,Fi+1	MBTOH Li+7→Fi+2,Fi+3
LDFI R[i+8]→Ri+8	LDFI R[i+9]→Ri+9	MBTOH Ri+6→Gi,Gi+1	MBTOH Ri+7→Gi+2,Gi+3
		MQSUBHUS Fi - Gi→X	MQSUBHUS Fi+2 - Gi+2→Z

図 8: FR500 でメディア演算命令を用いたカーネルコード

整数の飽和付き SIMD 加算を行う。

図 8 に、FR500 におけるカーネルコードを示す。太線部分が、メディア演算命令を使用して差の絶対値を求める基本ブロックであり、1 行が 1 つの VLIW 命令に対応する。1 つの基本ブロックが 2 ピクセルずつを並列実行している。さらに、ソフトウェアパイプライニングによってこの基本ブロックを 2 つずつオーバーラップさせることにより、最大 4 命令 (図 8 の網掛け部分) を同時実行するようにコーディングを行った。

3.2.2 FR550

FR550 は整数演算命令と、浮動小数点演算命令またはメディア命令をそれぞれ最大 4 個同時発行できる [7]。

FR550 は、整数演算命令およびメディア演算命令をそれぞれ 4 命令並列実行できるものの、ロード命令や一部のメディア演算命令は 2 個までしか同時発行できない。そのため、FR500 での 4 並列プログラムをそのまま 8 並列に拡張す

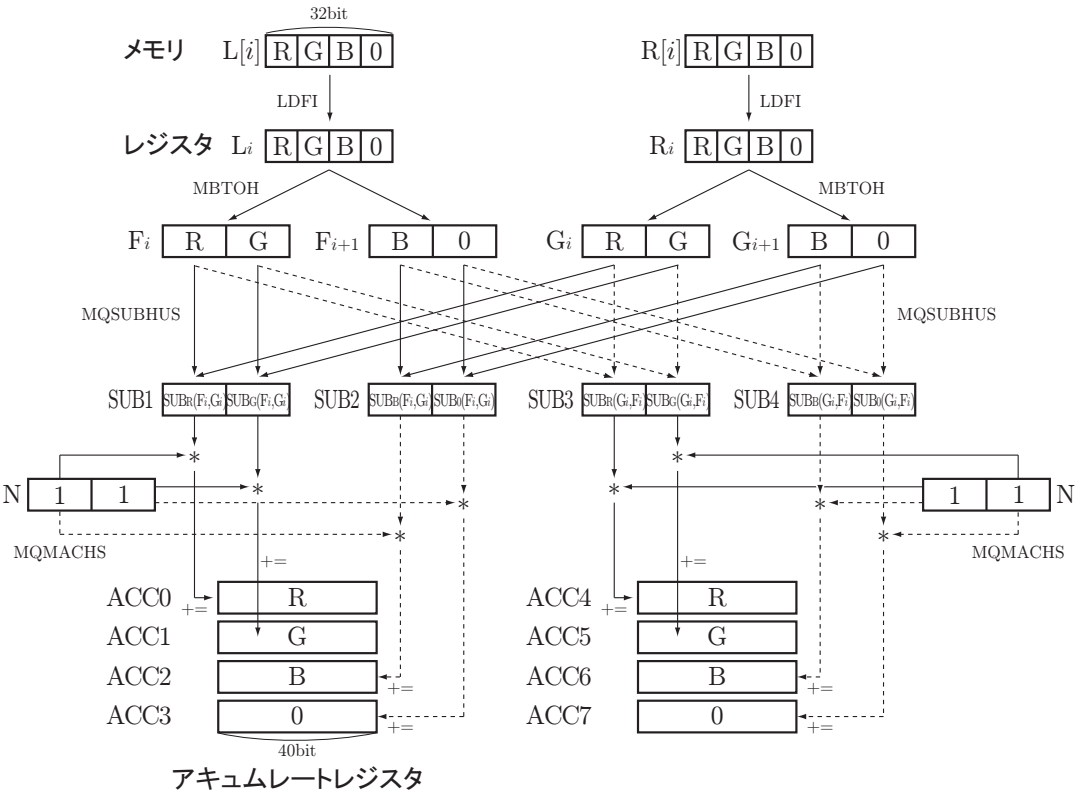

図 9: FR550 において積和演算命令を用いた場合の比較処理の流れ

ることはできない。そこで、新たな工夫が必要となる。飽和加減算命令は 2 個までしか同時発行できないため、メディア演算命令の残りの 2 スロットには、他のメディア演算命令と並列実行可能なメディアクワッド積和演算命令 MQMACHS (Quad Multiply and Accumulate Signed Halfword) を使用した。飽和減算により求めた値を本命令を用いて累積することによって、差の絶対値の総和を求めることができる。

図 9 に、メディアクワッド積和演算命令を用いた場合の比較処理の流れを示す。左右のスモールウィンドウから 1 ピクセル分の 32 ビットデータ $\mathrm{L}[i]$, $\mathrm{R}[i]$ を LDFI 命令によりレジスタ L_i, R_i にロードし、MBTOH 命令により各 8 ビットの RGB 値を 16 ビットに拡張し、左右それぞれレジスタ F_i, F_{i+1} と G_i, G_{i+1} に格納する。次に MQSUBHUS 命令により、各 RGB 成分について飽和減算を行う。例えば R 成分について考えると、まず、MQSUBHUS 命令で左右のピクセルの R 成分の差

$$\mathrm{SUB_R}(\mathrm{F}_i, \mathrm{G}_i) \quad = \quad \mathrm{F}_i - \mathrm{G}_i \tag{6}$$

$$\mathrm{SUB_R}(\mathrm{G}_i, \mathrm{F}_i) \;=\; \mathrm{G}_i - \mathrm{F}_i \tag{7}$$

を求め、それぞれレジスタ SUB1 と SUB3 の上位 16 ビットに格納する。さらにこれらの和を求めなければならないものの、飽和減算命令によってメディア演算命令スロットが埋まっているため、飽和加算命令 MQADDHUS は使えない。そこで、積和演算命令の代わりに、メディアクワッド積和演算命令を用いる。この命令は本来、行列の内積等を計算する命令であるものの、乗数を 1 にすることにより、加算命令として使うことができる。すなわち、上下 16 ビットに 1 を格納したレジスタ N を用意し、SUB1 と N との積和計算により、SUB1 の上下 16 ビット値をそのままアキュムレートレジスタに累積できる。以上の方法により、R 成分の差を累積する。G, B 成分についても同様である。最後に各アキュムレータの値を足し合わせることにより、スモールウィンドウ中の画素差の総和を求めることができる。

各命令の動作は次の通りである [8]。

ADD GR レジスタどうしの和を求める。GR レジスタは、32 ビットの汎用レジスタである。ここでは、スモールウィンドウの、ある行の先頭を指すポインタを移動させるために用いる。

MQMACHS Quad Multiply and Accumulate Signed Halfword. 2 個の FR レジスタを用いて、4 つの符号付き 16 ビット数のメディアクワッド積和演算を行い、結果を 40 ビットのアキュムレートレジスタに足しこむ。レジスタ SUB1 と N 間の積和演算とは、SUB1 の上位 16 ビットを $\mathrm{SUB1}_u$、下位 16 ビットを $\mathrm{SUB1}_l$ 等とすると、

$$\mathrm{SUB1}_u \cdot \mathrm{N}_u + \mathrm{SUB1}_l \cdot \mathrm{N}_l \tag{8}$$

を求める演算である。

図 10 に、FR550 におけるカーネルコードを示す。太線部分が、メディア演算命令を使用してスモールウィンドウの 1 行分の差の絶対値を求める基本ブロックであり、1 行が 1 つの VLIW 命令に対応する。図 10 において、`ADD L` は、メモリ配列 $\mathrm{L}[i]$ の先頭をスモールウィンドウのある行に移動することを意味する。`LDFI L`i は、メモリ $\mathrm{L}[i]$ からレジスタ L_i へ 32 ビットデータをロードすることを意味する。`MBTOH L`i は、L_i の各 8 ビットの RGB 値を 16 ビットに拡張し、レ

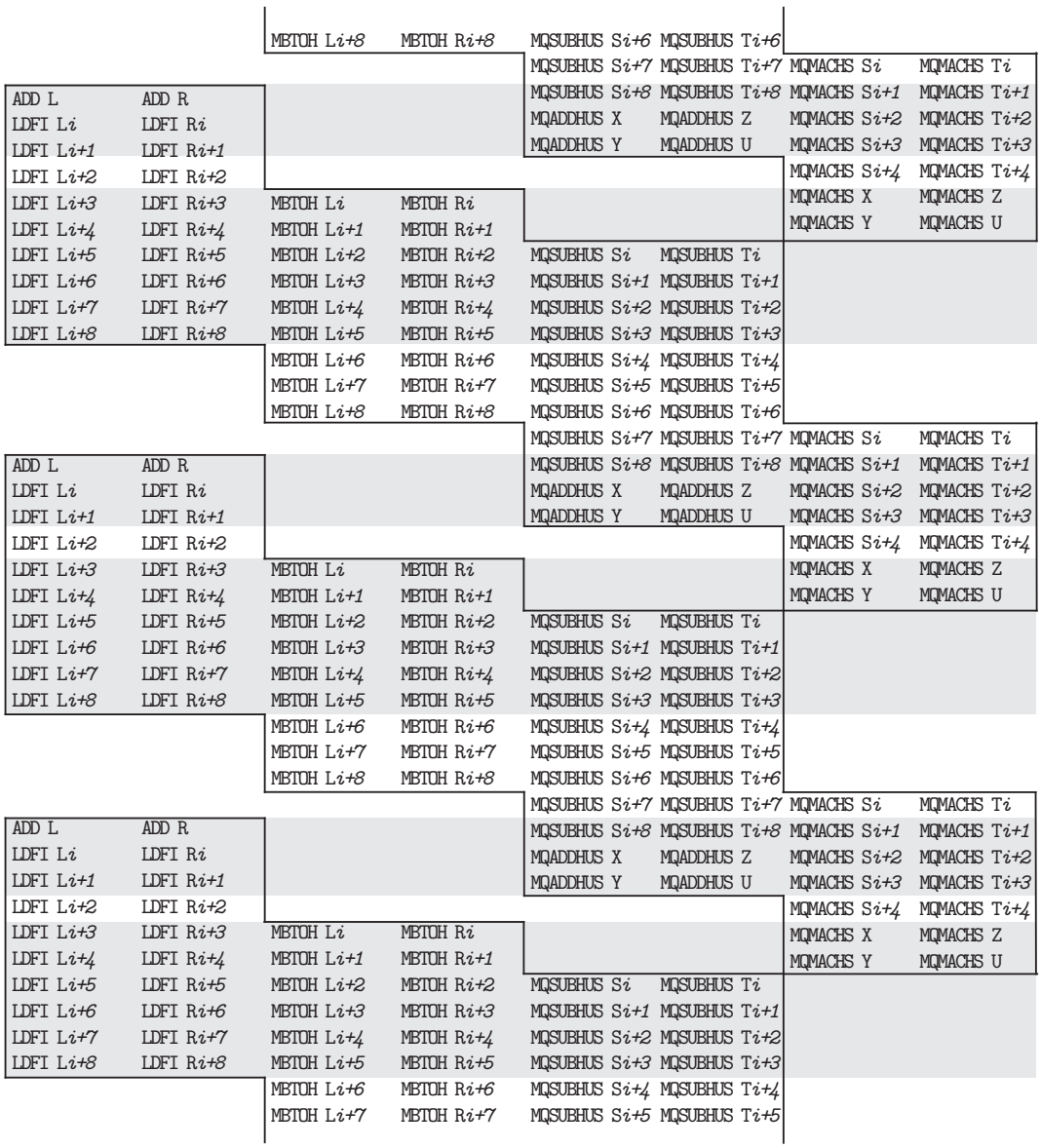

図 10: FR550 でメディア演算命令を用いたカーネルコード

ジスタ F_i, F_{i+1} に格納する。`MQSUBHUS S`i および`MQSUBHUS T`i は、それぞれ、

$$S_i = \mathrm{SUB}(\mathrm{F}_i, \mathrm{G}_i) = \mathrm{F}_i - \mathrm{G}_i \tag{9}$$

$$T_i = \mathrm{SUB}(\mathrm{G}_i, \mathrm{F}_i) = \mathrm{G}_i - \mathrm{F}_i \tag{10}$$

を計算する。`MQADDHUS X` 等は、

$$X = S_{i+5} + S_{i+6} \tag{11}$$

$$Y = S_{i+7} + S_{i+8} \tag{12}$$

$$Z = T_{i+5} + T_{i+6} \quad (13)$$

$$U = T_{i+7} + T_{i+8} \quad (14)$$

を計算し、MQMACHS Si 等は、S_i の値をアキュムレートレジスタへ累積することに対応する。さらに、ソフトウェアパイプライニングによって基本ブロックを 2 つずつオーバーラップさせることにより、最大 6 命令 (図の網掛け部分) を同時実行するようにコーディングを行った。

3.3 Intel IA-32 における高速化

3.3.1 MMX, SSE

MMX 命令は、MMX テクノロジ対応の Pentium プロセッサと Pentium II プロセッサから、IA-32 アーキテクチャに導入された SIMD 命令である [11]。MMX 命令は、MMX テクノロジレジスタ (以下、MMX レジスタという) または汎用レジスタを用いてパックドバイト、パックドワード、パックドダブルワード、またはクワッドワード整数オペランドを操作することができる。

また、ストリーミング SIMD 拡張命令 (SSE) は、Pentium III プロセッサファミリから IA-32 アーキテクチャに導入された命令である。これらの拡張命令は、高度な 2D および 3D 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、テレフォニ、およびビデオ会議などのアプリケーション用に、IA-32 プロセッサのパフォーマンスを強化するために開発されたものである。SSE では、64 ビット MMX レジスタ、64 ビットパックド整数データ型、およびパックド整数に対して SIMD 演算を実行する命令が用意されている。また、SSE は MMX の SIMD 実行モデルを拡張したものであり、128 ビットレジスタ内のパックドデータおよびスカラ単精度浮動小数点値を処理するための機能が追加されている。

さて、MMX には、差の絶対値を直接求める命令は搭載されていない。このため、MMX のみを用いた場合、FR-V と同様に飽和加減算命令を使用する必要がある。一方、SSE にはバイト整数の絶対差を求める命令 PSADBW が搭載されているため、IA-32 プロセッサの評価には SSE を用いることにした。ただし、SSE の PSADBW 命令は、128 ビットの XMM レジスタではなく、64 ビットの MMX レジスタを使用するため、ロード等には MMX 命令も用いる必要がある。1 つの PSADBW 命令により 8 つのバイト整数の絶対差を計算することができるので、2 ピクセル分の比較計算を一度に行うことが可能である。

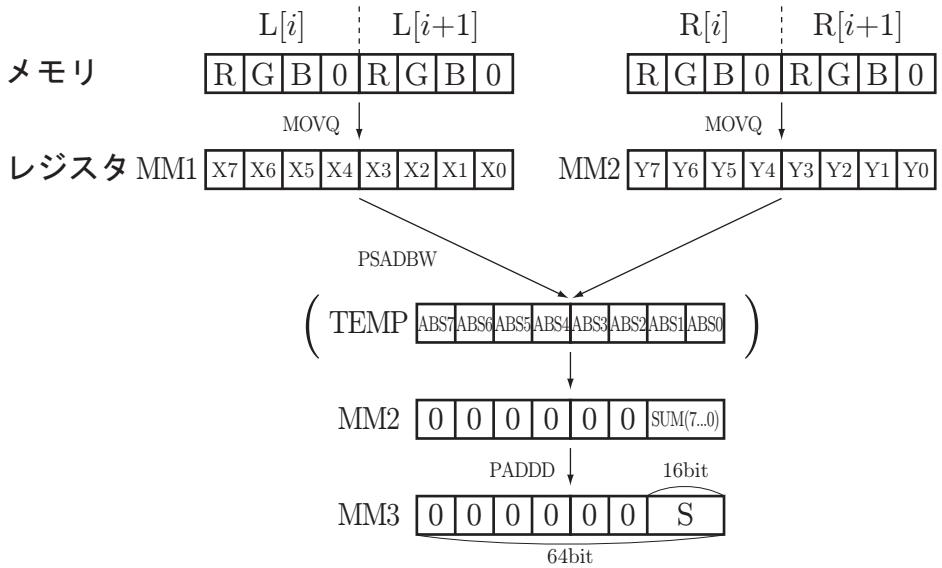

図 11: MMX, SSE を用いたときのデータの流れ

データの流れは図 11 のようになっている。左右のスモールウィンドウの 2 ピクセル分のデータを、MOVQ 命令によりメモリ L$[i]$, L$[i+1]$ から MMX レジスタ MM1、R$[i]$, R$[i+1]$ から MM2 にそれぞれロードし、PSADBW 命令により 64 ビット中の各 8 ビットずつについて絶対差を求め、これらを加算した結果を MM2 に格納する。その後、PADDD 命令によりアキュムレート用レジスタ MM3 に累積する。

各命令の動作は次の通りである。

MOVQ　Move Quadword. 64 ビットのパックドデータを、メモリから MMX レジスタ (またはその反対方向) に移動するか、MMX レジスタ間を移動させる。

PADDD　Add Packed Integers. ラップアラウンドモードを使用して、ソースオペランドとデスティネーションオペランドの対応する符号付きまたは符号なしのデータを加算する。データ型は、パックドダブルワードである。

PSADBW　Compute Sum of Absolute Differences. 2 つのソースオペランドにおいて対応する符号なしバイト整数の絶対差を計算して、これらの差を加算し、得られた和をデスティネーションオペランドの最下

```
movq   L[i], L[i+1] → mm1
movq   R[i], R[i+1] → mm2
psadbw mm1, mm2 → mm2
paddd  mm2, mm3 → mm3
```

図 12: MMX と SSE を用いたカーネルコード

位ワードに格納する。すなわち、図 11 において、

$$\begin{aligned} \mathrm{SUM}(7\ldots0) &= \sum_{i=0}^{7} \mathrm{ABS}i \\ &= \sum_{i=0}^{7} |\mathrm{X}i - \mathrm{Y}i| \end{aligned} \tag{15}$$

を求める演算である。MMX レジスタを使用する。

MMX と SSE を用いたカーネルコードを図 12 に示す。本カーネルコードを、スモールウィンドウ 81 ピクセルのうち 80 ピクセル分 (40 回) を行い、残りの 1 ピクセルは単独で計算を行った。なお、Intel IA-32 プロセッサはスーパースカラであり、VLIW のような命令スケジューリングは行っていない。

3.3.2 SSE2

ストリーミング SIMD 拡張命令 2 (SSE2) は、Pentium 4 プロセッサファミリから IA-32 アーキテクチャに導入された命令である。これらの拡張命令は、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、および工学計算などのアプリケーション向けに、IA-32 プロセッサのパフォーマンスを強化するために開発されたものである。

SSE2 は、MMX や SSE と同様に、SIMD 実行モデルを使用する。SSE2 では、SSE の SIMD 実行モデルが拡張され、パックド倍精度浮動小数点値と 128 ビットパックド整数の処理がサポートされた。

SSE2 を用いた場合でも、考え方は SSE のときと全く同じである。SSE2 の PSADBW 命令は SSE のものを 128 ビットに拡張したものなので、16 個のバイト整数の絶対差が計算でき、4 ピクセル分の比較計算を行うことができる。

データの流れは図 13 のようになっている。左右のスモールウィンドウの 4 ピクセル分のデータを、MOVDQU 命令によりメモリ L[i], L[$i+1$], L[$i+2$], L[$i+3$]

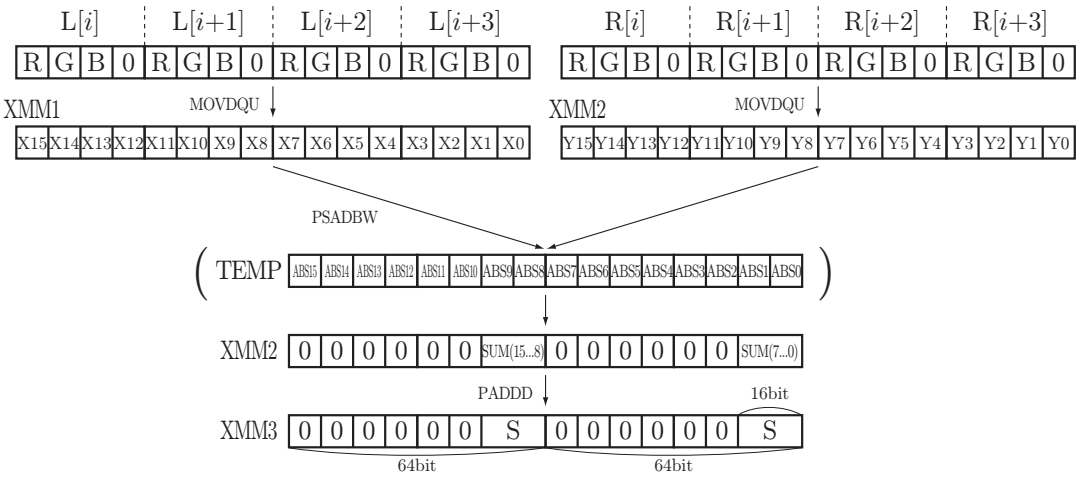

図 13: SSE2 を用いたときのデータの流れ

から XMM レジスタ XMM1 に、また、R[i], R[$i+1$], R[$i+2$], R[$i+3$] から XMM2 にそれぞれロードし、PSADBW 命令で 128 ビットデータ中の各 8 ビットずつの絶対差を求め、これらを加算した結果を XMM2 に格納する。その後、PADDD 命令によりアキュムレート用レジスタ XMM3 に累積する。

各命令の動作は次の通りである。

MOVDQU　Move Unaligned Double Quadword. メモリから XMM レジスタ、XMM レジスタからメモリ、または XMM レジスタ間について、ダブルクワッドワードオペランドを転送する。

PADDD　MMX の PADDD を 128 ビットに拡張したものである。SSE レジスタを使用する。

PSADBW　SSE の PSADBW を 128 ビットに拡張したものである。すなわち、図 13 において、

$$\mathrm{SUM}(7\ldots0) = \sum_{i=0}^{7} \mathrm{ABS}i \tag{16}$$

$$\mathrm{SUM}(15\ldots8) = \sum_{i=8}^{15} \mathrm{ABS}i \tag{17}$$

の両方を求める演算である。SSE レジスタを使用する。

SSE2 を用いたカーネルコードを図 14 に示す。本カーネルコードを、スモールウィンドウ 81 ピクセルのうち 80 ピクセル分 (20 回) を行い、残りの 1 ピクセルは SSE の PSADBW 命令を用いて単独で SAD を行った。SSE と同様に、命

```
movdqu L[i], L[i+1], L[i+2], L[i+3] → xmm1
movdqu R[i], R[i+1], R[i+2], R[i+3] → xmm2
psadbw xmm1, xmm2 → xmm2
paddd  xmm2, xmm3 → xmm3
```

図 14: SSE2 を用いたカーネルコード

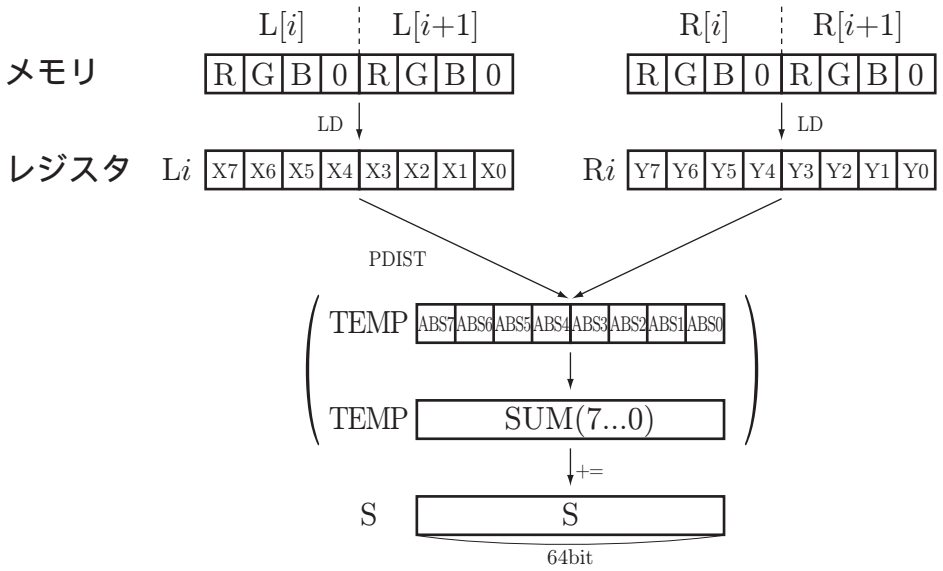

図 15: VIS を用いたときのデータの流れ

令スケジューリングは行っていない。

3.4 SPARC における高速化

SPARC プロセッサには、VIS (Visual Instruction Set) が搭載されている [12]。Intel の SSE 等と同様に、SIMD 型の並列演算、データのパック/アンパック、ビット操作等の命令がある。そこで、SPARC プロセッサでも同様のメディア演算命令を用いた距離画像生成プログラムを作成した。

SSE の PSADBW 命令と同様に、UltraSPARC の VIS にも PDIST 命令が用意されている。これは、64 ビットレジスタ中の各 8 ビットずつ絶対差を計算し、総和を求める命令である。SSE の PSADBW 命令と異なるのは、計算結果を第 3 オペランドであるデスティネーションレジスタに足しこむ点である。SAD 計算の後に加算命令を必要としないため、全体のステップ数を減らせる利点がある。

図 16 に、VIS を用いたときのデータの流れを示す。左右のスモールウィンド

```
ld    L[i], L[i+1] → Li
ld    R[i], R[i+1] → Ri
pdist Li, Ri, S → S
```

図 16: VIS を用いたカーネルコード

ウの 2 ピクセル分のデータを、LD 命令によりメモリ $\mathrm{L}[i]$, $\mathrm{L}[i+1]$ をからレジスタ L_i に、また、$\mathrm{R}[i]$, $\mathrm{R}[i+1]$ から R_i にそれぞれロードし、PDIST 命令により 64 ビット中の各 8 ビットずつ SIMD の絶対差を求め、これらを加算した結果をアキュムレート用レジスタ S に累積する。すなわち、PDIST 命令は、

$$\mathrm{S} \quad \leftarrow \quad \mathrm{S} + \mathrm{SUM}(7 \ldots 0) \tag{18}$$

を求める命令である。

図 16 に、VIS を用いたカーネルコードを示す。本カーネルコードを、スモールウィンドウ 81 ピクセルのうち 80 ピクセル分 (40 回) を行い、残りの 1 ピクセルは単独で計算を行った。SSE, SSE2 と同様に、命令スケジューリングは行っていない。

第 4 章　評価

4.1　評価システム

距離画像生成システムは、2 台のカメラで撮影したステレオ画像をいったん PC (Pentium III 1GHz、FreeBSD) に取り込み、画像データを LAN 経由で各種プロセッサシステムへ転送し、各プロセッサ上で画像処理を行った後、処理結果を PC へ戻し、ディスプレイに表示する構成としている。FR-V 上で動作するプログラムは Windows 2000 上の富士通製統合開発環境 SOFTUNE により作成し、RS-232C 経由で FR-V へ転送する。システム構成を図 17 に示す。

図 18 に、測定に用いたステレオ画像、また、図 19 に、生成した距離画像を示す。以下では、カーネル部分であるスモールウィンドウの比較処理の実行時間を測定し、処理に必要な消費電力量を求めた。

表 1 に、測定環境を示す。FR-V の評価には、C プログラムを SOFTUNE で最適化したものと、メディア演算命令によるものを用いた。また、汎用スーパースカラプロセッサの評価には、C プログラムを gcc -O2 で最適化したものと、メディア演算命令によるものを用いた。なお、OS は、Intel IA-32 は FreeBSD、

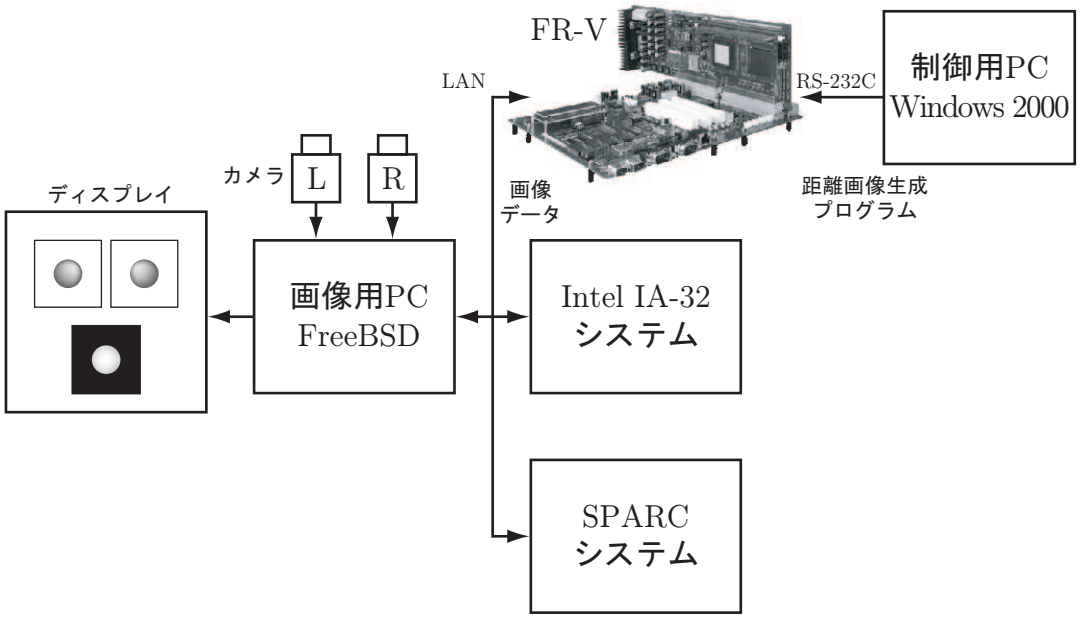

図 17: システム構成

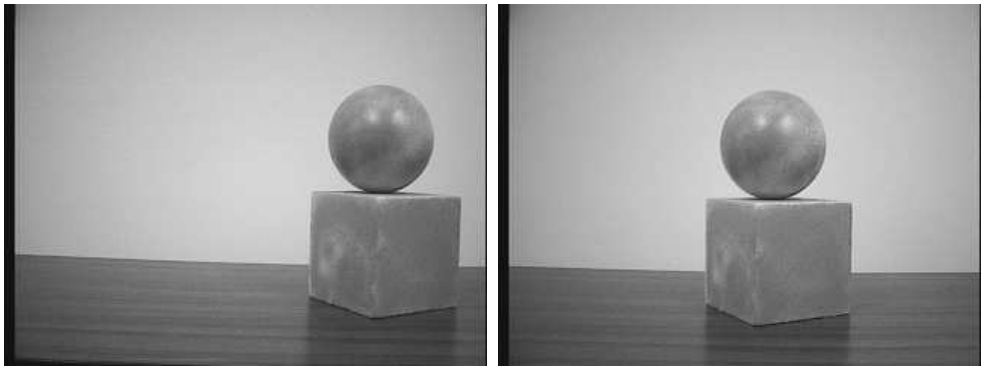

図 18: 測定に用いたステレオ画像

SPARC は Solaris である。

4.2 測定結果および考察

実行時間を図 20、消費電力量を図 21 に示す。まず、図 20 から、どのプロセッサにおいても、メディア演算命令を用いることによって距離画像生成処理能力が飛躍的に向上していることがわかる。特に、Intel IA-32 プロセッサでは 13 ~ 14 倍の性能向上が見られた。

また、図 21 より、消費電力量は FR550 が最も小さいことがわかる。

ところで、消費電力の評価尺度には、命令実行速度をどの程度重視するかに

表 1: 測定環境 [13, 14, 15, 16]

略称	周波数 [MHz]	消費電力 [W]	ロードモジュールの特徴
FR500 + 一般命令	240	1.8	C プログラムを SOFTUNE により最適化
FR500 + Media 命令			メディア演算命令を用いたハンドコーディング
FR550 + 一般命令	466	2.19	C プログラムを SOFTUNE により最適化
FR500 + Media 命令			メディア演算命令を用いたハンドコーディング
Pentium III + 一般命令	1000	26.1	C プログラムを gcc -O2 により最適化
Pentium III + MMX,SSE			MMX, SSE を用いたハンドコーディング
Xeon + 一般命令	2800	74.0	C プログラムを gcc -O2 により最適化
Xeon + SSE2			SSE2 を用いたハンドコーディング
UltraSPARC II + 一般命令	400	19	C プログラムを gcc -O2 により最適化
UltraSPARC II + VIS			VIS を用いたハンドコーディング
UltraSPARC III + 一般命令	900	65	C プログラムを gcc -O2 により最適化
UltraSPARC III + VIS			VIS を用いたハンドコーディング
UltraSPARC IIe + 一般命令	502	13	C プログラムを gcc -O2 により最適化
UltraSPARC IIe + VIS			VIS を用いたハンドコーディング
SPARC64 GP + 一般命令	400	70	C プログラムを gcc -O2 により最適化
SPARC64 GP + VIS			VIS を用いたハンドコーディング

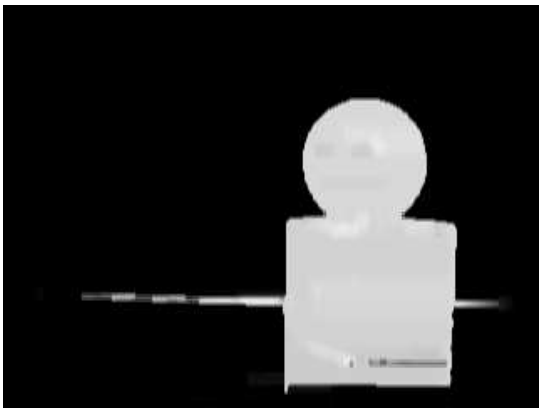

図 19: 生成される距離画像

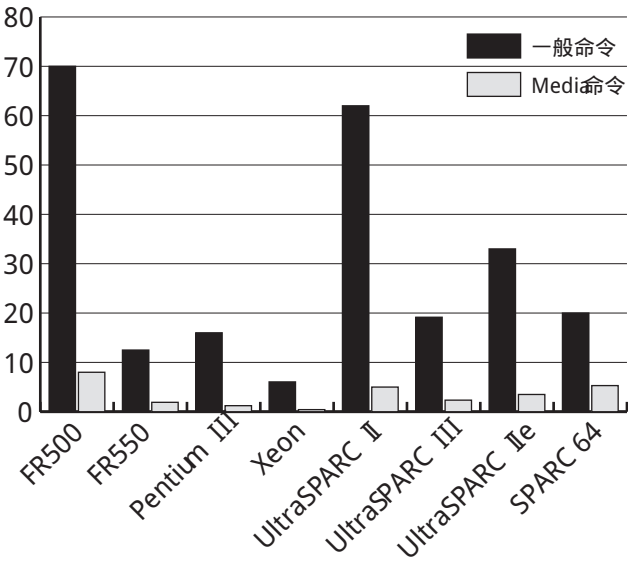

図 20: 実行時間 [s]

よっていくつかが考えられる [17]。例えば $Mips/Watt$ を用いた場合では

$$\frac{\text{Instruction/Delay}}{\text{Power/Delay}} \tag{19}$$

が評価尺度となる。本論文では、評価を行うプロセッサのアーキテクチャが異なり、総命令数が一定でないため、$Mips/Watt$ による評価の代わりに、$fps/Watt$ を用いた。これは、

$$\frac{\text{frames per second}}{\text{Power}} \tag{20}$$

と同値である。本評価尺度により作成したグラフを図 22 に示す。この結果から、性能と消費電力の両方を考慮した比較においても FR550 が最も優れているとい

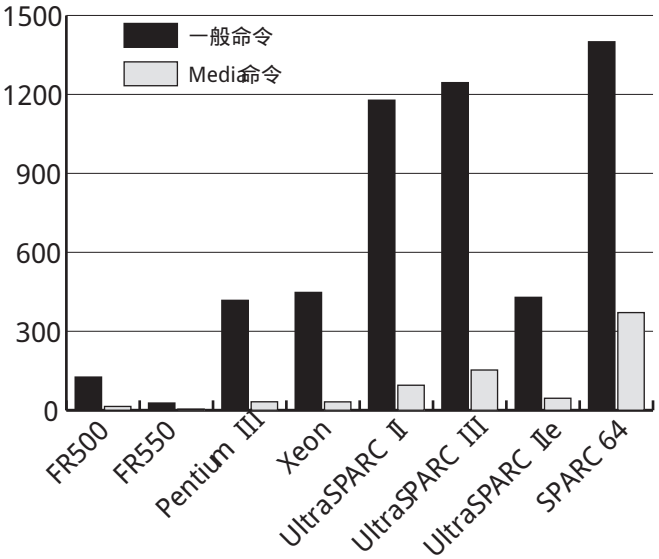

図 21: 消費電力量 [Ws]

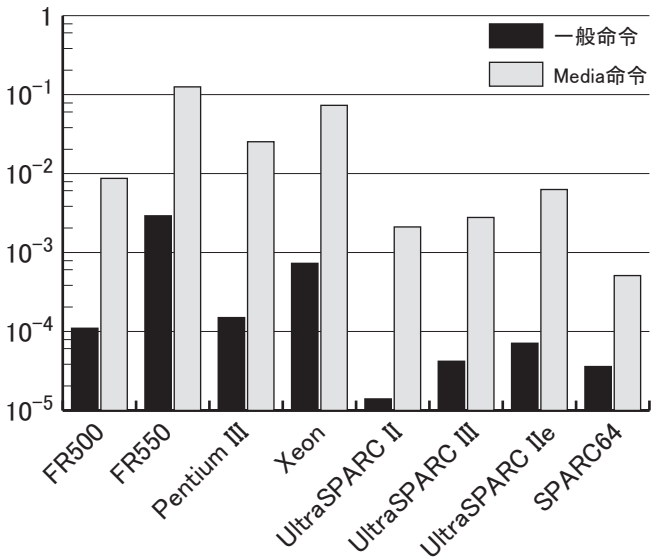

図 22: *fps/ Watt*

える。

さて、FR550 において、スモールウィンドウ全体の 81 ピクセルに対する差の絶対値の総和計算は、机上計算で 141 ステップである。この部分が 5M 回実行されるので 705M ステップを要する。466MHz で 1.91 秒かかるため 890M サイクル動いたことになる。よって CPI (Cycle per Instruction ただし VLIW 命令を 1 命令と数えている) は 1.26 となった．

ところで、FR-V には VIS の PDIST 命令のような差の絶対値を直接求める命令がないため、その計算と結果の累積に 4 命令を要したものの、PDIST 命令に相当する専用命令がある場合には 1 命令により実行できる。この場合、FR550 では、差の絶対値計算に関わる命令を 45 ステップ削減できるため、ステップ数比で約 32%の性能向上が期待できる。

さらに、ロードや PDIST 命令に相当する命令が 4 命令並列実行できるとすると、全体のステップ数は 68 ステップとなり、ステップ数比で約 52%の性能向上が期待できる。

第 5 章　おわりに

本稿では、VLIW 型メディアプロセッサ FR-V や汎用スーパースカラプロセッサを用いた距離画像生成について、処理速度や消費電力の観点から比較を行った。

その結果、いずれのプロセッサにおいてもメディア演算命令を用いることによって処理速度が飛躍的に向上することを示した。また、性能および消費電力に関する評価尺度 *fps/ Watt* についても FR550 が最も優れていることを示した。以上のことから、低消費電力を保ちつつ高い処理能力が要求されるシステムへの組込用プロセッサには FR-V が適しているといえる。

さらに、FR550 については、VIS の PDIST 命令に相当する専用命令を追加することにより、ステップ数比で約 32%の性能向上が期待でき、また、ロードや PDIST 命令に相当する命令を 4 命令並列実行することが可能であれば、ステップ数比で約 52%の性能向上が期待できることを示した。

さて、自動監視システムを実現するためには、距離画像を生成するだけではなく、監視する物体までの実際の距離も求めなければならない。距離画像から正確な距離を求めることについては、今後の課題である。

謝辞

本研究の機会を与えてくださった富田眞治教授に深甚なる謝意を表します。また、本研究に関して適切なご指導を賜った北村俊明・広島市立大学教授、中島康彦助教授、森眞一郎助教授、五島正裕助手、津邑公暁助手に心から感謝いたします。また、本研究でお世話になった富士通研究所システム LSI 開発研究所第二開発プロジェクト部長の高橋宏政氏に心から感謝いたします。さらに、日

頃からご助力頂いた京都大学大学院情報学研究科通信情報システム専攻富田研究室の諸兄に心から感謝いたします。

参考文献

[1] Kanade T.: A Stereo Machine for Video-Rate Dense Depth Mapping and Its New Applications, *Proceedings of 15th Computer Vision and Pattern Recognition Conference (CVPR)* (1996).

[2] SONY 木原研究所: Entertainment Vision Sensor, http://www.sony.co.jp/SonyInfo/News/Press/200202/02-0207/ (2002).

[3] CANESTA: http://www.canesta.com/ (2002).

[4] TYZX: Real-time Stereo Vision for Real-world Object Tracking (2002).

[5] United States Patent No.US 6,215,898 B1 (2001).

[6] Okutomi M.: A Multiple-Baseline Stereo, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol.15, No.4 (1993).

[7] 富士通: FR-V Family FR500シリーズ MB93501 LSI仕様書 (2001).

[8] 富士通: FR-V Family FR500 Series Instruction Set Manual (2001).

[9] 富士通: FR-V Family FR550シリーズ MB93551 LSI仕様書 (2002).

[10] 富士通: FR-V Family FR550 Series Instruction Set Manual (2002).

[11] Intel: IA-32 Intel Architecture Software Developer's Manual (2001).

[12] Sun Microsystems: The VIS Instruction Set, ver.1.0 (Jun. 2002)

[13] 富士通: FR-V Series, http://www.fme.fujitsu.com/products/micro/fr/ (2002).

[14] Intel: Intel Architecture Processors, http://developer.intel.com/design/processor/ (2002).

[15] Sun: UltraSPARC Processors, http://www.sun.com/processors/ (2002).

[16] 引地徹, 加藤哲, 大田秀信, 嘉喜村靖: 64ビット RISCプロセッサ:SPARC64 GP, 雑誌富士通 (Jul. 2000).

[17] Brook, D.: Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors, *IEEE MICRO*, Nov/Dec, pp.26-44 (2000).