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

2004－ＳＬＤＭ－113　（３）
2004／1／22

DCT向けプロセッサの構造最適化に関する研究

関根 敦司† 後藤 源助‡ 多田 十兵衛‡

†山形大学工学部 〒992-8510 山形県米沢市城南 4-3-16

E-mail: †ash@eigoto12.yz.yamagata-u.ac.jp

あらまし 現在市場に出ている動画像を扱うシステムLSIには、画像や音声の伸張技術としてDCTが扱われている。一般的に、DCTのように処理が複雑で汎用のプロセッサだけで扱うと演算時間が非常にかかってしまう処理には、DSPのような専用のプロセッサを用意する。しかしチップにDSPを乗せるとハードウェアコストが非常に増えてしまう。そこでハードウェアコストの増加をできるだけ抑えしかも高速であるようにプロセッサの構造を最適化する必要性が出てくる。本研究ではDCTでもっとも演算時間のかかるΣの演算を専用に扱うハードウェアとして積和演算、Σ演算器を取り上げ、それぞれのプロセッサのみの場合とDCTブロックをプロセッサに組み込んだ場合での演算時間とハードウェアコストを比較した。結果として演算時間は積和演算器、Σ演算器ともに約 40%短縮し、ハードウェアコストは積和演算器で約48%、Σ演算器で約46%削減できた。

キーワード DCT、プロセッサ、演算時間、積和演算

Research on optimization of the processor structure for DCT

Atsushi Sekine† Gensuke Goto‡ and Jube Tada‡

†Yamagata University faculty of technology 4-3-16, Jonan, Yonezawa-shi, Yamagata-ken, 99-8510 Japan

Abstract DCT is used for a picture or sound extension technology to realize by the system LSI dealing with video signals . Generally, the dedicated processors are needed to process DCT or any other video/audio signals, such as a DSP. However, if a DSP is adopted as a processing element, hardware cost will increase very much. Then, suppressing the increase in hardware cost is necessary as much as possible, and moreover optimizing the structure of a processor is required for high-speed processing. In this research, performance of multiple-accumulate operation and Σ-operation were raised by implementing a major part of DCT as hardware. Speed and hardware cost is compared with respected to conventional multiple-accumulate hardware and Σ-operation machine with a normal processor and a processor-added DCT block. It was shown that the operation speed is 40% faster than normal processor. The cost is 48% lower for the multiple-accumulate hardware and 46% lower for the Σ-operation than processor added the DCT block.

• **Keyword** DCT, processor, operation time, multiple-accumulate operation

1. 初めに

1.1 背景

現在市場に出ている携帯電話、PDA、カーナビゲーションシステム、ハイビジョンTV、情報家電など、動画像処理を扱うシステムLSIは、高速かつ製造コストの削減からチップ面積の削減が求められている。[1]

また現在のDCT技術は、画像・音声伸張技術としていろいろな場面で応用されている。一般的に上述のシステムをLSIに実装する際、1チップ内に汎用のプロセッサとDCTなどの処理を行うDSP(デジタルシグナルプロセッサ）との二つのプロセッサを乗せている。このため、二つのプロセッサにチップ面積の大部分が取られてしまう。

DSPを搭載せずにDCTを実現しようとした場合、汎用プロセッサにプログラムとして与えることになり、チップ面積は大幅に削減することができる。しかしDCTのように処理が複雑なプログラムを実行すると演算時間が増大してしまい、結果として上述の高速であるシステムという要求をみたさない。以上のことから、チップ内にDSPを持たずに汎用プロセッサの改善

だけで高速化が行えることが理想である。

1.2 目的

本研究では、ハードウェアコストを削減しかつ演算時間を損なわないように、プロセッサの構造をアプリケーションに応じて最適化することを目的とする。

そのため、アプリケーションでもっとも演算時間のかかる部分をプロセッサに専用命令として組み込み、演算時間とハードウェアコストの削減をする。今回はDCT においてもっとも演算時間のかかるΣ演算部分をハードウェアとしてプロセッサに組み込み、演算時間の短縮、ハードウェアコストの低下を試みた。

2. 実験方法

本研究では、DCT を汎用プロセッサにプログラムで実現した場合、DCT のもっとも演算時間のかかるΣの演算を従来の手法である積和演算器としてプロセッサに組み込んで実現した場合、新たな手法としてΣ演算器をプロセッサに組み込んで実装した場合、すべてをハードウェアで実現した場合についてそれぞれハードウェアのコストと演算時間を比較、検討した。

2.1 プロセッサ

今回使用するプロセッサは、32-bit でオープンソースであるプロセッサを使用した。このプロセッサは、最小限のハードウェアで実現しているため内部の改変が容易である。図 1 に今回使用したプロセッサの命令例(ADD、SUB 命令)を示す。このように、MOV 命令などで第一引数に特定のレジスタ(ALU_IN、ALU_ADD、ALU_SUBなど)にデータを入れることで、演算を行い、出力用レジスタ(ALUOUT)からデータを取り出すといった流れで演算が行われていく。データの流れは、命令メモリから入ったアドレスがデータバスを通りコアモジュールで WE 信号（ニーモニックの第一引数)、OE 信号（ニーモニックの第二引数）を生成し、レジスタモジュールから必要なレジスタを引き出し目的の演算モジュールで演算を行う。

```
■Add
    ■ MOV ALU_IN R1
    ■  MOV ALU_ADD R2
    ■  MOV R3  ALUOUT
■SUB
    ■ MOV ALU_IN R1
    ■MOV ALU_SUB R2
    ■MOV R2 ALUOUT
```

図 1　プロセッサの命令例
（ADD、SUB 命令）

2.2 DCT

今回使用する DCT の演算範囲は 32-bit で 8×8 個の値に対して、S_{00} から S_{77} の値を演算した。式（1）に DCT 演算の式を示す。

$$S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} S_{yx} \cos\left(\frac{(2x+1)\times u \times \pi}{16}\right) \cos\left(\frac{(2y+1)\times v \times \pi}{16}\right) \quad (1)$$

ここで

$$C_u \cdot C_v = \begin{cases} \frac{1}{\sqrt{2}} & \text{u,v=0の時} \\ 1 & \text{それ以外} \end{cases}$$

である。S_{yx} の値は画素値を示している。

2.3 コサインテーブル

コサインの演算は非常に時間がかかるためプログラムで値を出さず、あらかじめコサインテーブルとしてすべての場合のプロセッサに組み込んだ。式(2)にコサインテーブルで計算されるコサインの式を示す。この演算結果をテーブルとして用いた。

$$\cos\left(\frac{(2x+1)\times u \times \pi}{16}\right) \times \cos\left(\frac{(2y+1)\times v \times \pi}{16}\right) \quad (2)$$

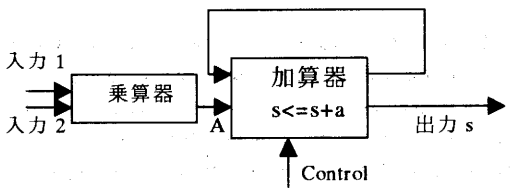

図 2　積和演算器のブロック図

2.4 積和演算器

積和演算器に用いる乗算器は、Wallace Tree[2] を用いて高速化し、$\Sigma a_i b_i$ の値を求めた。図 2 に積和演算器のブロック図を示す。図のように、乗算器の出力 a の値が出るたび加算器の信号 s と、Control 信号から出力を意味する信号が送られてくるまで加算を行う。

2.5 Σ演算器

図 3 にΣ演算器のブロック図を示す。こちらも乗算器は Wallace Tree を用いた。また乗算器の出力をレジスタに格納し、8 個のデータがそろった時点で加算を行う。その際、加算は高速化のために Wallace Tree を改良して用いた。また、Control 信号によりレジスタ

に直接値を入れることも可能にした。またΣ演算に利用した Wallace Tree の例を図 4 に示す。この図は 8-bit での例だが、実際には 32-bit で演算している。図の枠は FA、HA を用いる範囲、黒丸は FA の演算結果、△は FA の桁上がり、四角は演算結果である。このように 8-bit の 4 段の加算が最終的に 8-bit の 2 段の加算に圧縮されるのが Wallace Tree である。

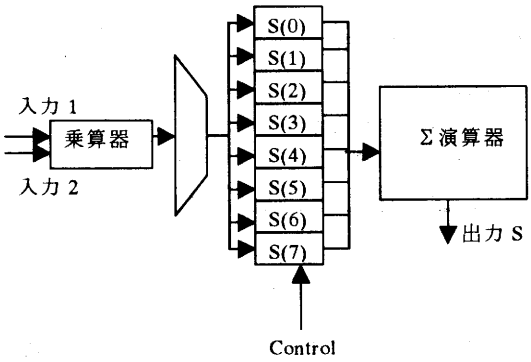

図 3　Σ演算器のブロック図

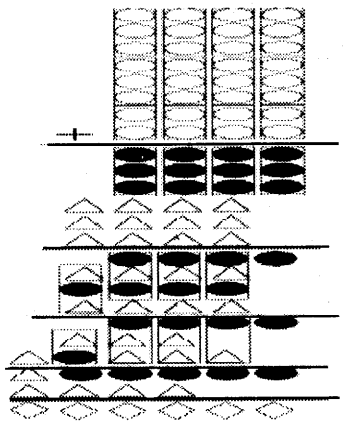

図 4　Wallace Tree を用いたΣ演算の例（4-bit、8 個の加算）

3. 実験結果

前章であげた 4 つのアプローチでの演算時間とゲート数を調べた。

3.1　プロセッサのみ

表 3 のプロセッサはプロセッサに DCT 演算をさせた場合の DCT の演算時間とゲート数である。この表からわかるように、プロセッサしかないのでハードウェアコストはとても小さいが、演算時間がとても多い。

3.2　積和演算器

表 3 の積和演算は積和演算器を組み込んだプロセッサを用いた場合の DCT の演算時間とゲート数である。この結果から演算時間はプロセッサの場合よりも約 40%抑えることができたが、ゲート数は約 9.79%増加した。また、図 5 に積和演算器を追加した時に使用したコードの例を示す。サイクル数は 194 サイクルであった。

```
IMM REGA 24
IMM COS_IN 0
MOV  REGB COS_OUT
IMM REGA 15
IMM COS_IN 16
MOV REGB COS_OUT
IMM REGA 62
IMM COS_IN 32
MOV REGB COS_OUT
        .
        .
        .
MOV REGB COS_OUT
IMM REGA 95
IMM COS_IN 992
MOV REGB COS_OUT
IMM REGA 80
IMM COS_IN 1008
MOV REGB COS_OUT
MOV R0 SG_OUT
```

図 5　積和演算器を追加したプロセッサにあたえたコード例

3.3　Σ演算器

表 3 のΣ演算はΣ演算器を組み込んだプロセッサを用いた場合の DCT の演算時間とゲート数である。この結果から演算時間はプロセッサの場合よりも約 40%抑えることができ、ゲート数は約 14.14%増加した。ゲート数の増加が積和演算器よりも多いのは内部で使用したレジスタが原因と考えられる。図 6 に積和演算器を追加した時に使用したコードの例を示す。サイクル数は 209 サイクルであった。

3.4　DCT ブロック

表 3 のハードウェアは DCT とプロセッサを用いた場合の DCT の演算時間とゲート数である。この表からわかるように演算時間はプロセッサのみの場合と比較して約 51%短縮したが、汎用プロセッサと DCT がハードウェアとして存在するためハードウェアコストが約 110.17%増加した。

表 3　すべての処理時間とゲート数

	演算時間(ns)	ゲート数
プロセッサのみ	413824	12282
プロセッサ+積和演算	250884	13485
プロセッサ+Σ演算器	252160	14019
プロセッサ+DCT	199040	25813

```
IMM REGA 24
IMM COS_IN 0
MOV REGB COS_OUT
IMM REGA 15
IMM COS_IN 16
MOV REGB COS_OUT
IMM REGA 62
IMM COS_IN 32
MOV REGB COS_OUT
        .
        .
        .
MOV R7 SG_OUT
MOV S0 R0
MOV S1 R1
MOV S2 R2
MOV S3 R3
MOV S4 R4
MOV S5 R5
MOV S6 R6
MOV S7 R7
```

図 6　Σ演算器を追加したプロセッサに与えたコード例

4. 考察

4.1　ハードウェアコストの比較

表 4.1 に、すべてのハードウェアコストとハードウェアを基本とした時のコスト削減率の比較を示す。この表から、ゲート数はプロセッサがもっとも少なく次いで積和演算器、Σ演算器となり、もっとも多くなったのはハードウェアであった。積和演算器とΣ演算器は、プロセッサよりも多少大きいがほぼ同じゲート数で実現できたと言える。Σ演算器のほうが、600 ゲートほどゲート数が多いのは、Σ演算器で使用したレジスタと加算部分のレジスタの分で増えたと考えられる。ハードウェアのゲート数がもっとも多くなったのは、プロセッサと DCT 両方をチップに乗せようとしたからである。

4.2　演算時間での比較

表 4.2 にすべての演算時間と、プロセッサを基本とした時の演算時間短縮割合の比較を示す。

この表から、演算時間はプロセッサがもっとも長く、次いでΣ演算器、積和演算器となり、もっとも演算時間が短かったのは DCT をハードウェアのみであらわした時であった。積和演算器とΣ演算器はほぼ同じ演算時間であったが、わずかにΣ演算器の方が演算時間が短かったのはΣ演算器の仕様で 8 回しか演算を行えず、8 回の演算を行った値を一度汎用レジスタに保存し、最後に 8 個の汎用レジスタからデータをロードし、加算を行う必要がある。このため積和演算器では 64 クロックで加算処理を行えるが、Σ演算器では 64+8 クロック必要になったためであると考えられる。

表 4.1 すべてのハードウェアコストとハードウェアを基本とした時のコスト削減率

	ゲート数	コスト削減率(%)
プロセッサのみ	12282	52.42
プロセッサ+積和演算	13485	47.76
プロセッサ+Σ演算器	14019	45.69
プロセッサ+DCT	25813	0

表 4.2 すべての演算時間と、プロセッサを基本とした時の演算時間短縮割合

	処理時間(ns)	短縮割合(%)
プロセッサのみ	413824	100
プロセッサ+積和演算	250884	39.37
プロセッサ+Σ演算器	252160	39.07
プロセッサ+DCT	199040	51.90

5. まとめ

実験結果から、以下のことが言える。ハードウェアコストの面で見ると積和演算器、Σ演算器ともにプロセッサとほぼ変わらないものを作成することができた。演算時間の面からは、積和演算器、Σ演算器ともに約40%の演算時間短縮ができ、これは DCT をハードウェアとしてプロセッサの外部に置いたときとほぼ変わらない値である。ハードウェアコスト、演算時間の両面から見るとアプリケーションが DCT の場合に最適な構造であるといえるのは積和演算器を追加したプロセッサである。

また今後の課題としては、同じ手法を用いて異なるアプリケーションに対して構造最適化を行い、構造最適化の指標をとっていきたい。

文　献

[1] 持田侑宏：" DSP の現状と動向",情報処理,30,11,pp1291-1299(1989-11)

[2] "A 4.1-ns Compact 54×54-b Multiplier Utilizing Sign-Select Booth Encoders", G. Goto, A. Inoue, R. Ohe, S. Kashiwakura, S. Mitarai, T. Tsuru, and T. Izawa, IEEE Journal of Solid-State Circuits, Vol. 32, No. 11,pp 1676-1682, 1997/11.