HDL設計における乗算器の性能比較

野尻 誠*, 久津輪 敏郎**, 小堀 研一**

Performance Comparison of Various Multipliers Based on HDL Design

Makoto NOJIRI*, Toshiro KUTSUWA** and Ken-ichi KOBORI**

*大阪工業大学大学院電気工学専攻（〒535 大阪府大阪市旭区大宮5-16-1）

**大阪工業大学電子工学科（〒535 大阪府大阪市旭区大宮5-16-1）

*Master Course Electrical Engineering, Osaka Institute of Technology (5-16-1 Omiya, Asahi-ku, Osaka-shi, Osaka 535)

**Department of Electronic Engineering, Osaka Institute of Technology (5-16-1 Omiya, Asahi-ku, Osaka-shi, Osaka 535)

Abstract

Demands of minimization of digital signal processor, and reduction of power consumption have been risen in recent years. From these demands, high performance multiplier has to be developed for designs of digital filter and video transmission LSI. In this paper, we compare various multipliers by the hardware description language, and design FIR filter based on the above result.

Key Words : *Hardware Description Language, Top-down Design, Multiplier, Filter, Digital Signal Processor*

1. 緒 論

デジタルコンピュータにおける演算アルゴリズムは、機械のなかのメモリやレジスタへの数値の格納のされかたに大きく依存している。内部の数表現が異なってくれば、そのための演算ハードウェアの方式も当然変わってくる。したがって数表現の選択は、設計者の立場からコンピュータのアーキテクチャに、利用者の立場からは数値解析手法に大きな影響を与えることになる。

通常のコンピュータは、基数 $r \geq 2$, デジット集合 $\{0, 1, \dots, r-1\}$ からなる固定基数の演算システムを用いている。ある数のすべてのデジットは正の値をもち、各々の数は一意的に表現される。例えば固定基数の演算システムを用い、一般的に使われているWallce乗算器では、被演算数の各桁同士の積を部分積として求め、最後に生成した部分積の総和を加算して積を求める。しかしこれでは、ビット数が増えればそれに比例して部分積の数が増え、たくさんのハードウェアを必要とする。また、乗数を2ビットごとに区切り、0～3倍の乗算として部分積を算出するL-Booth乗算器は、乗数を2ビットに区切るにより、部分積の数を半分に抑えることができるが、あらかじめ被乗数の0倍から3倍の値を生成するハードウェアが必要なことと、それを選び出すマルチプレクサなどのハードウェアを必要とするといった問題がある。

そこで、デジット集合が正の数だけでなく負の数も許す固定基数表示法を拡張した正準サインデジット表現

(Canonical Signed-Digit:CSD)を用いることで、乗算中の零による平均シフト長を大きくでき、乗算に必要な加算演算を減少させることができるCSD乗算器に注目してみた。しかしこのCSD乗算器のアルゴリズムはすでに提案されたものであり¹⁾, CSD乗算器を使った回路設計の研究は行われている²⁾。

今回は、最近注目され、設計現場に普及しはじめているハードウェア記述言語（HDL; Hardware Description Language）を用いて、論理的ではなく、実際に設計を行うことで実用的な乗算器かを問う性能比較を行ってみた³⁾。また回路設計実例として、CSD乗算器を使ったFIRフィルタを設計することによって、CSD乗算器の信頼度についても調べてみた⁴⁾。

2. 数表現

コンピュータで実用されているのは、有限精度の演算方式だけであるから、可能な数表現は長さの有限なものに限定される。したがって、演算システムと内部の数表現の選択のしかたによって、機械命令の設計の効率や近似計算の精度が大きく左右されることがある。そこで、高速の演算器を設計する際に有効であることが知られている符号付きデジット（Signed-Digit:SD）表現、および正準サインデジット（Canonical Signed-Digit:CSD）表現について述べる⁵⁾。

2.1 符号付きデジット集合

ある与えられた基数 r に対して、SD数の各デジットは、 α を整数として、つぎの $2\alpha+1$ 個の値をとることができる。

$$\Sigma_r = \{-\alpha, \dots, -1, 0, 1, \dots, \alpha\}$$

ここで、最大デジット値 α は次の領域になければならない。

$$\left\lceil \frac{r-1}{2} \right\rceil \leq \alpha \leq r-1$$

また、各整数は $\alpha \geq 1$ 、 $r \geq 2$ となっていることが前提である。平衡デジット集合 Σ_r において最小冗長度を実現するためには、最大値として次の値を選べばよい。

$$\alpha = \left\lceil \frac{r}{2} \right\rceil$$

ここで $[x]$ は実数 x より小さいか、等しい最大の整数を表す。そしてSD数は、次式で表記することができる。

$$Y = (y_{n-1} \dots y_0 y_{-1} \dots y_{-k})_r$$

ただし、 n および k は語長を表している。

このときSD数 Y の代数値、 Y_u は次式で与えられる。

$$Y_u = \sum_{i=-k}^{n-1} y_i \times r^i$$

ここで、例として $Y_u = -3$ の場合を考える。語長は $n=4$ 、 $k=0$ とし、デジット集合は $\{1, \bar{0}, 1\}$ とする。1は $\bar{1}$ のデジット-1を表している。このとき-3は以下の5つのデジットで表現できる。

$$Y = (00\bar{1}\bar{1})_2 = -1 \times 2^1 - 1 \times 2^0$$

$$Y = (0\bar{1}01)_2 = -1 \times 2^2 + 1 \times 2^0$$

$$Y = (\bar{1}101)_2 = -1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0$$

$$Y = (0\bar{1}\bar{1}1)_2 = -1 \times 2^2 + 1 \times 2^1 - 1 \times 2^0$$

$$Y = (\bar{1}11\bar{1})_2 = -1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 - 1 \times 2^0$$

2.2 正準サインデジット表現

ある数においてSD表現ではいくつかの表現が存在するが、SD表現が、

$$D = D_{n-1}D_{n-2} \dots D_1D_0$$

であるとすると、非零のデジットが連続して表れないものを正準サインデジット表現とよび、次式のように定義される。

$$D_i \times D_{i-1} = 0 \quad 1 \leq i \leq n-1$$

ただし、 D の最左端の2つのデジットの積が1にならない条件、

$$D_{n-1} \times D_{n-2} \neq 1$$

を満たす必要がある。例えば先に述べたように $Y_u = -3$ の場合、SD表現は5種類存在するが、条件を満たした $(0\bar{1}01)_2$ がCSD表現であり、SD表現から得られるCSD表現はただ1つとなる。

2.3 2進数からCSD表現への変換

デジタル信号処理では、情報を2進数で扱うためCSD表現を用いた乗算を行う場合、データを2進数からCSD表現へ変換 (decode) する必要がある。そして2進数表現は、SD表現の特殊な形と考えて2進数の左端に0を付加して変換操作を行う。その変換アルゴリズムを図1に示す¹⁾。

3. CSD乗算器の設計

CSD乗算器の設計において、乗数だけをCSDにデコードし、それによって生成される部分積を半分に減らす方法をとる。例えば2の補数表現を用いる乗算器の場合は、被乗数、乗数ともに2の補数器を準備しておき、負の数を補数化して乗算を実行し、結果が負ならば2の補数に直す補正が必要である。これに対してCSD表現を用いる場合には、乗数について2の補数をとるの必要がなく、CSDにデコードすることにより生成される部分積を半分に減らすことができ、得られる演算結果も補正する必要がないため、その分のハードウェアを削減できる。

今回設計したCSD乗算器は、図2に示すようにCSDデコード制御信号生成部と、部分積生成部および部分積加算部から成る乗算器本体部の2つのブロックから構成されている。それでは、各機能ブロックの機能について説明する。

3.1 制御信号生成部

CSDデコード部の設計であるが、まず節2.3で述べた変換アルゴリズムを用いて2進数表現をCSD表現へと変換操

図1. 変換アルゴリズム

図2. 4×4CSD乗算器構成図

作を行う。次にどのビットに非零ビットが立っているかを検出しなければならない。その対策として、CSDデコードの際にその情報を乗算器本体に知らせる方法をとる。それにはフラグ ($f_0 \sim f_7$) を用いて、各フラグにおいてどのビットが立っているかどうかという情報を持たせる。フラグ番号が奇数 (f_{2i+1}) のものと偶数 (f_{2i}) のものとが1対でCSD表現 (D_i) を表し、フラグ番号が奇数のフラグが立つと乗算器本体部内の2の補数器によって2の補数演算を実行し、フラグ番号が偶数のフラグが立つと論理積演算を実行し、フラグ番号奇数、偶数のフラグ両方ともが立たない場合はゼロ演算を実行するようにした。このフラグ表現で論理積演算実行の検出と、CSDデコードの検出を同時に行うことができる。図1で示した変換アルゴリズムでCSD表現へコード変換を行い、各ビットの状態をフラグで表したものを表1に示す。

3.2 乗算器本体部

乗算器本体の設計であるが、制御信号生成部より送られたフラグ信号をもとに部分積を生成し、それを加算していく。加算器には3入力2出力の桁上げ保存加算器を使い、

表1. CSD表現へのコード変換

B_{i+1}	B_i	C_i	C_{i+1}	D_i	f_{2i+1}	f_{2i}
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	1	$\bar{1}$	1	0
1	1	0	1	$\bar{1}$	1	0
1	1	1	1	0	0	0

B_{i+1}, B_i : 乗数 D_i : CSD表現 $(i=0, 1, \dots, n-1)$
 C_{i+1}, C_i : 桁上げ信号 f_{2i+1}, f_{2i} : フラグ

ただし、 f_{2i+1}, f_{2i} は D_i の状態を示す2本のフラグである
 $D_i=1$ なら $f_{2i}=1$, $D_i=\bar{1}$ なら $f_{2i+1}=1$
 f_{2i} のフラグが立っているとき、論理積演算を実行
 f_{2i+1} のフラグが立っているとき、2の補数演算を実行
 f_{2i+1}, f_{2i} フラグともに立っていないとき、ゼロ演算を実行

加算演算を順次実行していき、最後に2入力1出力の桁上げ先見加算器で乗算結果を出力する。例えば、Wallce 乗算器で4×4の乗算を実行する場合、部分積の加算演算を4回行わなくてはならないが、CSD表現を用いた乗算器では乗数を2ビットずつで区切ってみると、どちらか1ビットが必ずゼロとなり、その分の部分積算出の演算を省いてシフトさせることができ、4回必要な加算演算を2回に減らすことができる。その演算例を図3に示す。同様な考え方で、8×8の乗算を実行する場合においても4回の加算演算で結果を得ることができるといったように、ビット幅が増えてもWallce 乗算器と比べて、CSD表現を用いた乗算器では、加算演算を半分に減らすことができる。

3.3 乗算器の性能比較

今回設計したCSD乗算器と、人間が手計算で行う演算をそのまま実現したWallce 乗算器、乗数を2ビットごとに区切り、0倍～3倍の乗算として部分積を算出する L-Booth乗算器を加えた3つの乗算器を、NTTが開発したハードウェア記述言語SFLおよび論理合成システム PARTHENONを使い、まず乗算器のソースをコーディングし、シミュレーションで動作確認をしたあと論理合成により最適化を行い、PARTHENONの中にあるセルライブラリをもとにテクノロジマッピングを行うことにより、各乗算器に対してソースの行数、論理合成にかかった時間、消費電力、面積、ゲート数、遅延時間を求めた。それを表2に示す。なお今回設計に使用したマシンは、Sun の SPARCstation10である。ただし表においての乗算演算は、ビット幅8×8ビットの演算とする。

表2の結果より、CSD乗算器が優れていることがわかった。これはCSD乗算器が乗数を2ビットずつで区切って調べると、どちらか1ビットが必ずゼロとなり、その分の部分積算出の演算を省いてシフトさせることができ、必要な加算演算を半分に減らすことができ、その結果、面積、ゲート数を減らし、ひいては消費電力、遅延時間も小さくなったと考えられる。

図3. 4×4のWallce乗算演算とCSD乗算演算

表2. 乗算器の性能比較

	Wallce乗算器	L-Booth乗算器	CSD乗算器
消費電力 (μ W/MHz)	2729.3	2426.5	2139.9
面積 ($k\mu m^2$)	566.23	506.14	467.80
ゲート数 (ゲート)	714	622	576
遅延時間 (ns)	52.95	65.42	49.20
ソースの行数	184	218	232
論理合成にかかった時間(秒)	213	257	249

図4. ビット幅拡張におけるゲート数比較

またビット幅を拡張した場合におけるゲート数の比較を図4に示す。

4. FIRフィルタの設計

信号処理ASICでは、特にFIR（有限長インパルス応答）フィルタがよく使われている。そこで今回、優位性が確認されたCSD乗算器を用い、実用的な回路であるコサインロールオフフィルタの設計を行った^{4),5)}。

4.1 コサインロールオフフィルタ

このフィルタは、QAM（直交振幅変調）をはじめデジタル変調系に多用される帯域制限フィルタで、その通過特性は直線とコサインカーブの組み合わせになっていて、公称カットオフ周波数を中心に奇対称形をしている。また比較的精度が高いフィルタが簡単に作れることでも知られている。

4.2 インパルス応答の算出

FIRフィルタにおいてインパルス応答は、そのシステムに関する多くの情報をもっている。そこでコサインロールオフフィルタのインパルス応答を次式より求める。

$$h(t) = \frac{\sin \pi t / T}{\pi t / T} \times \frac{\cos \alpha t / T}{1 - (2\alpha t / T)^2}$$

ただし、Tはナイキスト間隔、 α はロールオフ率とする。

4.3 効率的なデータ形式変換

上式より求めたインパルス応答をまず、固定小数点データ形式に変換し、必要な乗算器の数を計算すると、このフィルタにおいて19個必要なことがわかった。さらにこれをCSD表現に変換し、必要乗算器の個数を計算すると、11個となった。これは、CSD表現を用いると乗算器を合成する

表3. フィルタの性能比較

	従来フィルタ	本設計フィルタ
消費電力 (μ W/MHz)	37672.2	8920.9
面積 ($k\mu m^2$)	9356.43	2215.35
ゲート数 (ゲート)	11032	2942
遅延時間 (ns)	55.52	20.19

上で非ゼロビットが連続して現れないため、部分積の数が半分以下になるためである。さらにこれについて、共通項をくり出し論理圧縮を行うことで、最終的に乗算器7個でこのフィルタを構成することが可能となる。

4.4 設計評価

フィルタの設計評価として、Wallce乗算器を用い、改良を施していないフィルタを従来フィルタとし、CSD表現、およびCSD乗算器を用い、さらに論理圧縮を行ったフィルタを本設計フィルタとして、消費電力、面積、ゲート数、遅延時間で性能評価比較を行った。それを表3に示す。今回改良した本設計フィルタは、すべての面で従来フィルタを上回る性能を示すことがわかった。

5. 結論

本論文ではハードウェア記述言語SFLを用い、正準サインデジット表現を採用したCSD乗算器を設計し、従来の乗算器と性能比較することによりその優位性を確認した。このSFLおよび論理合成システムPARTHENONを用いた設計環境により、従来の設計手法では手間と日数がかかっていた設計を、乗算器のソース記述を変えるだけで設計の変更ができ、動作シミュレーションを効率的に行うことができた。また設計応用例として、CSD乗算器を使ったFIRフィルタの設計を行い、実回路においての優位性も確認できた。

(1996.4.1-受理 1996.7.4-再受理)

文 献

- 1) Kei Hang, 監訳・堀越：“コンピュータ高速演算方式”，近代科学社（1980）
- 2) R. Hartley：“Optimization of canonic signed digit multipliers for filter design”，proceeding IEEE International Symposium on Circuits and Systems, pp.1992-1995, (1991.6)
- 3) 野尻 誠，久津輪敏郎，小堀研一，江端克彦：“HDLによる乗算器のトップダウン設計”，平成7年電気関係学会関西支部連合大会 G 360（1995.11）
- 4) 野尻 誠，久津輪敏郎，小堀研一：“HDLによるデジタル符号化LSIのトップダウン設計”，第10回回路実装学術講演大会 13A-16（1996.3）
- 5) 尾知 博：“デジタル・フィルタ設計入門”，CQ出版社（1990.5）