計測自動制御学会東北支部 第181回研究集会 (1999.5.21)
資料番号 181-1

直線位相特性を用いた高次FIRフィルタの高性能VLSIアーキテクチャ

High-Performance Architecture of Higher-Order FIR Filters using Linear Phase Characteristic

○田村惣一*, 恒川佳隆*, 吉田等明*, 三浦 守*

○Souichi Tamura*, Yoshitaka Tsunekawa*, Hitoaki Yoshida*, Mamoru Miura*

*岩手大学 工学部

*Faculty of Engineering, Iwate University

キーワード： FIRフィルタ (FIR filter), 直線位相特性 (linear phase characteristic), 分散演算 (distributed arithmetic), 高性能VLSIアーキテクチャ (high-performance VLSI architecture), VLSI評価 (VLSI evaluation)

連絡先： 〒020-8551 盛岡市上田4-3-5 岩手大学 工学部 情報工学科
田村惣一, Tel.: (019)621-6468, Fax.: (019)623-5491, E-mail: souichi@cis.iwate-u.ac.jp

1. はじめに

ディジタルフィルタは計測制御, 通信伝送, オーディオ, 家電製品などの応用分野で広く利用されている. ディジタルフィルタは, 有限長のインパルス応答を有するFIR(Finite Impulse Response)フィルタと無限長のインパルス応答を有するIIR(Infinite Impulse Response)フィルタに分類される. FIRフィルタは直線位相特性を実現できるため, 波形伝送上の問題となる位相ひずみが生じない特長がある. しかし, IIRフィルタでは急峻な遮断特性を低い次数で得られるのに対し, FIRフィルタではそれを得るために次数を相当高くする必要がある. そのため, 現在, 情報通信の分野などにおいて, 極めて高次なFIRフィルタが要求されている. 高次のフィルタを実現する場合, 一般的な乗算器を用いた構成法では, ハードウェア量・消費電力等が非常に増大するため大きな問題となる. そこで内積演算を効率的に行うためにROMを用いた分散演算によるFIRフィルタの構成法が提案されてきた[1].

この構成法に対して, FIRフィルタの直線位相特性を利用したSFA(Serial Full Adder)を用いた手法が提案されているが[2], これは低次でかつROMを分割しない構成法に対する議論しか行われてこなかった. そのため, 高次でこの手法を適用すると消費電力が非常に増大する.

そこで, 本研究では, この手法を高次に適用し, ROMを分割化することによって, 効率的なFIRフィルタの低消費電力化を図る. またROMの代わりに, 我々が提案したROMと同機能でしかも消費電力が小さい最適関数回路を用いた構成法を適用することにより[1], さらなる低消費電力化を

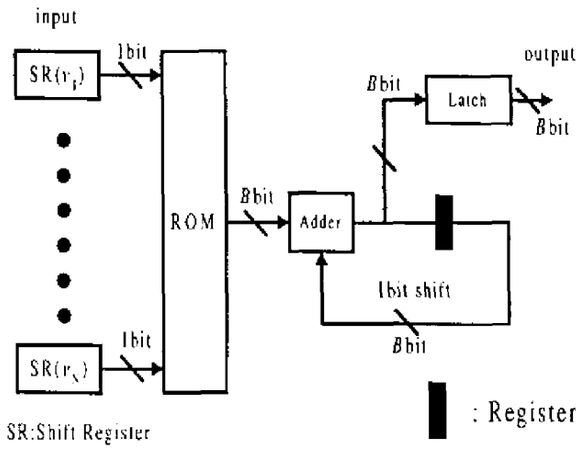

Fig. 1 Basic structure of distributed arithmetic.

図る.

さらにROMなどの関数テーブルを分割することにより，関数テーブルを参照して得られた値を加算する必要があるがこれをCLA(Carry Look Ahead)のbinary treeではなく，CSA(Carry Save Adder)のWallace treeとすることによりさらなる消費電力，滞在時間の減少を図る.

2. 分散演算を適用したVLSIアーキテクチャ

2.1 分散演算型アーキテクチャ

分散演算[3)~5)]とは定係数の内積演算をテーブル・ルックアップにより実現する計算手法である.いま，項数Nの係数ベクトル$\boldsymbol{a}=(a_1,\ldots,a_N)$と変数ベクトル$\boldsymbol{v}=(v_1,\ldots,v_N)$との内積

$$y=\boldsymbol{a}\boldsymbol{v}=\sum_{i=1}^{N}a_iv_i \qquad (1)$$

を考える.ただし，$-1\le v_i<1$で，v_iはBビットの固定小数点形の2の補数表示である.つまり，

$$v_i=-v_i^0+\sum_{k=1}^{B-1}2^{-k}v_i^k \qquad (2)$$

と表される.ここで，v_i^kはv_iのkビット目の値で0または1である.式(2)を式(1)に代入すれば，内

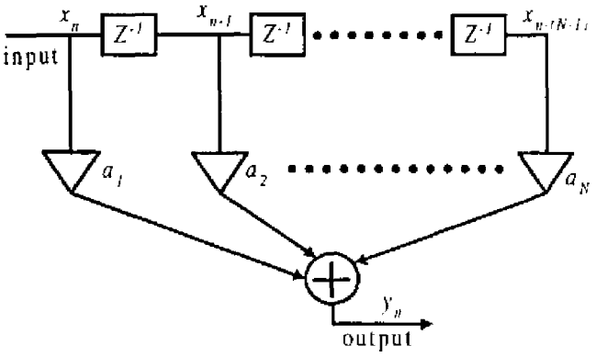

Fig. 2 Direct form structure of FIR filter with n-taps using multipliers.

積演算$\boldsymbol{av}$は次式で示される.

$$y=-\Phi(v_1^0,\ldots,v_N^0)+\sum_{k=1}^{B-1}2^{-k}\Phi(v_1^k,\ldots,v_N^k) \qquad (3)$$

ただし，Φは

$$\Phi(v_1^k,\ldots,v_N^k)=\sum_{i=1}^{N}a_iv_i^k \qquad (4)$$

である.

ここで，分散演算の基本的な構成法をFig. 1に示す.この構成では，$(v_1^k,\ldots,v_N^k)$をアドレスとするROMに，入力変数の各ビットと係数との内積演算の結果Φをテーブルとして書き込んでおき，計算時にはテーブルを参照して得られた値を順次1ビットシフトしながら，加え合わせる操作を行う.つまり，分散演算では内積演算が乗算器を用いずに，語長B回分のテーブルルックアップとシフト加算によって実現できる.

2.2 分散演算に基づくFIRフィルタの構成

Fig. 2に示される一般的なNタップのFIRフィルタは次式で表される.

$$y_n=\sum_{i=1}^{N}a_ix_{n-(i-1)} \qquad (5)$$

ここで，a_iはフィルタ係数，$x_{n-(i-1)}$は入力変数，y_nは出力変数である.式(1)と式(5)より，$(v_1,v_2,\ldots,v_N)$が$(x_n,x_{n-1},\ldots,x_{n-(N-1)})$に対応している

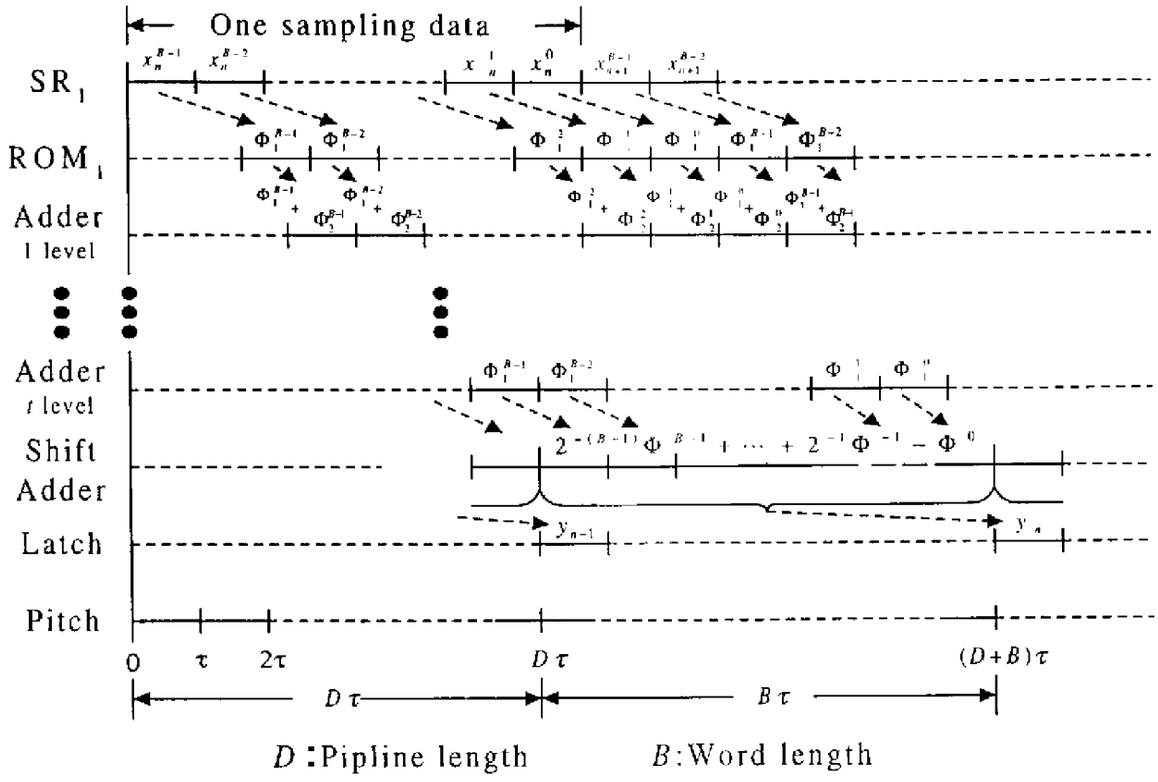

Fig. 5 Timing chart for FIR filter applying distributed arithmetic.

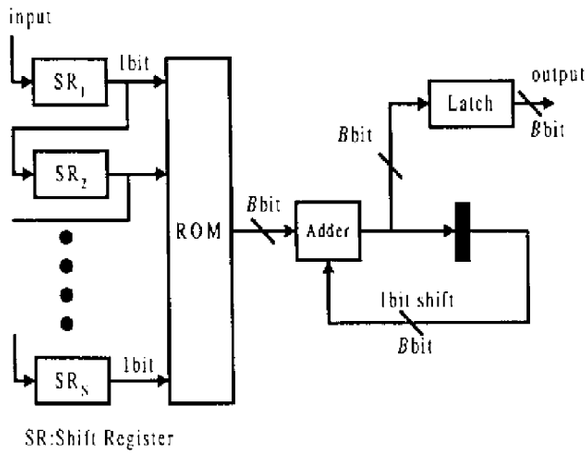

Fig. 3 Basic structure of FIR filter applying distributed arithmetic.

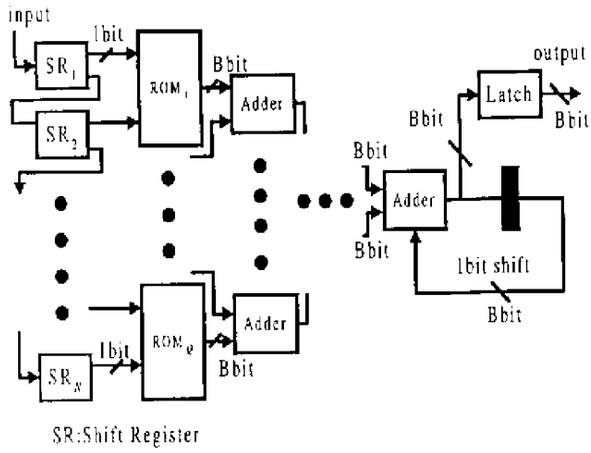

Fig. 4 Structure of FIR filter which divides one function Φ into some function abd their additions.

ことがわかる．また，Fig. 2 からFIRフィルタは，1サンプリング周期ごとに入力データが各遅延器を伝搬して，内積演算を実行していることがわかる．しかし，FIRフィルタに分散演算の適用を考慮した場合，Fig. 1 で示した構成では入力データが伝搬できないため，入力部の構成法を新たに考える必要がある．そこで，入力部のシフトレジスタを Fig. 3 のような構成にする．この構成法によって，入力データを1クロックごとに1ビットずつ伝搬させることができる．これより，分散演算に基づくFIRフィルタの実現が可能となることがわかる．

しかし，Fig. 3 で示した構成をハードウェア実現する場合2^Nワードのメモリ容量が必要となる．そのため，高次のFIRフィルタに対しては膨大なROM容量が必要となる．そこで，メモリ容量を大幅に削減するために関数Φの分割化法を用いる．関数Φの分割化法とは，分割数をQとして以下の処理を施すことである．

$$\begin{aligned}\Phi(v_1^k,\ldots,v_N^k) &= \sum_{i=1}^{N} a_i v_i^k \\ &= \sum_{i=1}^{N/Q} a_i v_i^k + ,\ldots, + \sum_{i=Q'}^{N} a_i v_i^k \\ &= \Phi(v_1^k,\ldots,v_{N/Q}^k) + ,\ldots, + \Phi(v_{Q'}^k,\ldots,v_N^k)\end{aligned} \tag{6}$$

ここで，

$$Q' = \frac{N(Q-1)}{Q} + 1 \tag{7}$$

この手法によって容量の大きな関数用メモリをいくつかの小容量のメモリの加算として実現できる．このように分割化によって大幅にハードウェア量を減少できるのが分散演算の大きな特長となっている．この分割化を適用した分散演算によるFIRフィルタの構成を Fig. 4 に示す．ただし，ここでは関数用メモリにROMを用いており，以下この構成法を従来法と呼ぶ．

次に，分散演算に基づくFIRフィルタにパイプライン処理を施す．このとき，このフィルタは，シフトレジスタで構成された入力部，アドレス生成部，ROMのアクセス部，関数Φの加算部，シフト加算部のステージから構成される．なお，分割数がQの場合，関数Φの加算部では，$\lfloor \log_2 Q \rfloor$の段数分の加算ステージを設けてパイプライン処理を実行している．ここで用いられている記号$\lfloor s \rfloor$は，sを越える最小の整数を表す．この結果，パイプライン処理のピッチτは，ほぼ加算器の演算時間まで抑えられる．

ここで，パイプライン化したときのタイミングチャートを Fig. 5 に示す．同図から入力データが各ステージにおいてピッチτで連続処理できることがわかる．この結果，1つの入力サンプル値が処理時間$B\tau$という次数に依存しない一定の時間で処理される．つまり，このことは，サンプリング周期が$B\tau$であり，次数に依存しない値になることを意味する．

また，パイプラインの段数をDとした場合，あるサンプルが入力されてから，そのサンプルに対応した出力が得られるまでの時間，すなわち滞在時間は Fig. 5 から$(D+B)\tau$となることがわかる．ここで，パイプラインの段数Dは語長Bに対して基本的に小さいことから，滞在時間は次数の増加に対して1から2サンプリング周期の値となる．

3. SFA を用いた構成法

式(1)の$\boldsymbol{v}$を$\boldsymbol{x}$に置き換えて展開すると，

$$\begin{aligned} y_n \;=\; & -\left[a_1 x_1^0 + a_2 x_2^0 + \cdots + a_N a_N^0\right] \\ & + \left[a_1 x_1^1 + a_2 x_2^1 + \cdots + a_N a_N^1\right] \cdot 2^{-1} \\ & \qquad \vdots \\ & + \left[a_1 x_1^{B-1} + a_2 x_2^{B-1} + \cdots \right. \\ & \qquad \left. + a_N a_N^{B-1}\right] \cdot 2^{-(B-1)} \end{aligned} \tag{8}$$

となる．ここでFIRフィルタが直線位相特性を満たし，またタップ数Nが偶数である場合，

$$a_1 = a_N, a_2 - a_{N-1}, \cdots, a_{N/2} = a_{(N/2+1)} \tag{9}$$

となる．これを利用して，式(8)は，

$$
\begin{aligned}
y_n \;=\; & -\left[(x_0^0+x_N^0)a_1+\cdots\right. \\
& \qquad \left.+(x_{N/2}^0+x_{(N/2+1)}^0)a_{N/2}\right] \\
& +\left[(x_0^1+x_N^1)a_1+\cdots\right. \\
& \qquad \left.+(x_{N/2}^1+x_{(N/2+1)}^1)a_{N/2}\right]\cdot 2^{-1} \\
& \qquad \vdots \\
& +\left[(x_0^{B-1}+x_N^{B-1})a_1+\cdots\right. \\
& \qquad \left.+(x_{N/2}^{B-1}+x_{(N/2+1)}^{B-1})a_{N/2}\right]\cdot 2^{-(B-1)}
\end{aligned}
\tag{10}
$$

と表せる．式(10)の(　)中の入力変数x_i^jがそれぞれ1ビットであるために，ここではまずこれを1ビット全加算器(FA)を用いて実現する．1ビット全加算器により桁上げが生じるが，これは同じ係数a_iの1つ上位のビットへの入力とする．例えば語長がBである場合，FAによって$B-1$ビット目で生じた桁上げは$B-2$ビット目へ入力される．SFAを用いた構成法ではこれをタップ数がNの場合，式(10)より$\lfloor N/2 \rfloor$個のSFAを用いて並列に行う．$B-2$ビット目でのFAによる桁上げは$B-3$ビット目へと入力する．これを次々に符号ビットまで行う．しかし，1ビット目から符号ビットへの桁上げがあるためにオーバーフローが生じる可能性がある．そこで符号ビットを1ビット拡張することによりこの問題を解決することができる．関数テーブルには入力変数の各ビットとフィルタ係数との内積演算の結果を書き込んでおくために，式(8)から式(10)のように内積演算の回数が半分に減少すると関数テーブルのメモリ容量は半分に減少する．また，メモリ容量が半分の関数テーブルに分割化法を用いると分割数は半分になる．

例としてタップ数12，関数テーブルの分割数6，アドレス線数2の場合のSFAを用いていない構成法(従来法)をFig. 6に，SFAを用いた構成法をFig. 7にポす．Fig. 6とFig. 7よりSFAを用いた構成法は関数テーブルの個数をP，多入力加算部の加算器数をAとすると，それぞれ$\lfloor P/2 \rfloor$，$\lfloor A/2 \rfloor$とすることができる．

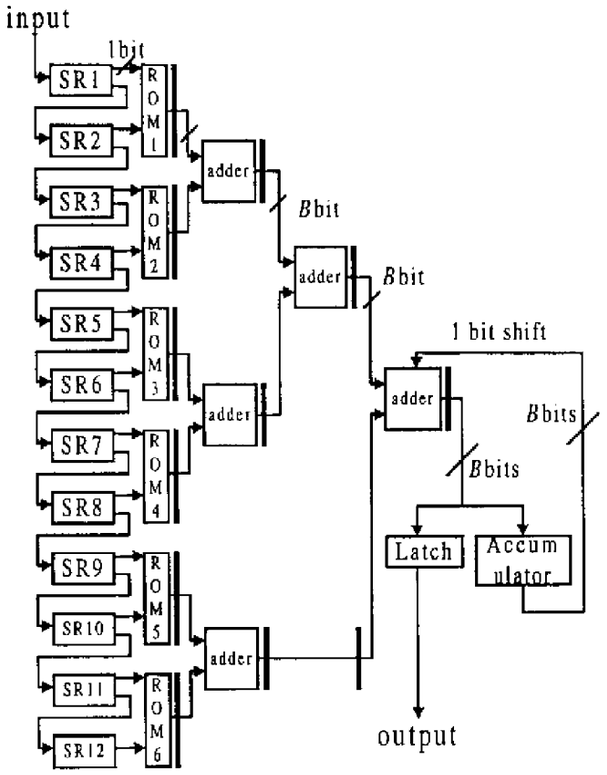

Fig. 6　Structure of FIR filter with no linear phase characteristic.

4.　最適関数回路

分散演算に基づくFIRフィルタは，関数Φの生成にROMを用いた場合，非常に大きな消費電力を必要とする．これは，ROMの消費電力が非常に大きいためである．ゆえに，関数Φの生成にROMではなく，それと同機能で低消費電力となる回路を用いた構成法が要求される．そこで，関数ΦをROMで実現する代わりに，論理ゲートを用いてそれと同機能となる回路を実現させる構成法を提案する．この構成法を関数Φの最適化と呼ぶことにする．また，このとき生成される回路を最適関数回路，ROMのアドレス線に相当するものを最適関数選択線と呼ぶことにする．

ROMに書き込まれている関数Φのテーブルを Fig. 8 (a)のように表す．ここで，$v_1^k \cdots v_m^k$はmビッ

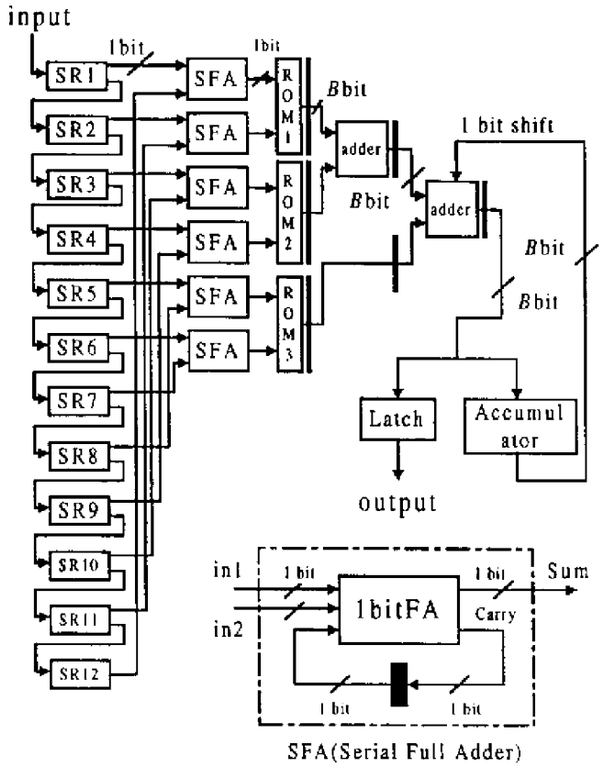

Fig. 7 Structure using SFA of FIR filter with linear phase characteristic.

ト長の入力変数であり，$w_i^{B-1}\cdots w_i^0$はi番地に格納されているBビットのデータを示す．同図(a)の関数$\Phi(v_1^k,\cdots,v_m^k)$を$2^m\times B$を行列W_1とみなすと，

$$W_1=\begin{bmatrix} w_0^{B-1} & \cdots & w_0^0 \\ \vdots & \cdots & \vdots \\ w_{2^m-1}^{B-1} & \cdots & w_{2^m-1}^0 \end{bmatrix} \tag{11}$$

と表せる．ここで，関数Φの最適化で行われている冗長性の削除を大域的削除と局所的削除に大きく分けて考える．大域的削除とは，Fig. 8 (a)(b)で示すように式(11)の行ベクトルに対して共有可能となる行ベクトルを括り出すことである．その結果，生成される行列W_2は，

$$W_2=\begin{bmatrix} w_{hc_1}^{B-1} & \cdots & w_{hc_1}^0 \\ \vdots & \cdots & \vdots \\ w_{hc_i}^{B-1} & \cdots & w_{hc_i}^0 \end{bmatrix} \tag{12}$$

となる．さらに，Fig. 8 (b)(c)で示すように式(12)の列ベクトルに対しても同様な手法を施す．その結果，生成される行列W_3は，

$$W_3=\begin{bmatrix} w_{hc_1}^{vc_1} & \cdots & w_{hc_1}^{vc_j} \\ \vdots & \cdots & \vdots \\ w_{hc_i}^{vc_1} & \cdots & w_{hc_i}^{vc_j} \end{bmatrix} \tag{13}$$

となる．ここで，式(13)で表されるW_3の相違なる各行ベクトルを横のカテゴリと呼ぶことにする．同様に，W_3の相違なる各列ベクトルを縦のカテゴリと呼ぶことにする．この冗長性の削除を基にして，最適関数回路はFig. 9のように横のカテゴリに対する部分機能回路と縦のカテゴリに対する部分機能回路を縦続接続した回路構成になる．一方，局所的削除とは，各部分機能回路に対して二段論理簡単化後に多段論理最適化[6)7)]を施すことである．これらの冗長性の削除によって，最適関数回路のゲート数が減少する．

5. 多入力加算部の高性能化

Fig. 4に示すように分散演算を用いたFIRフィルタの構成法では分割した関数テーブルの出力の総和を求めるために多入力加算が必要となる．これまでは，Fig. 10のようにこの多入力加算部にCLA(Carry Look Ahead)加算器をbinary treeにして用いてきた．しかし，分割数を大きくした場合，この構成法では消費電力が大きくなり問題となる．そのため，この多入力加算の部分を検討することにより，さらなる低消費電力化を図る．

5.1 桁上げ保存加算器

新たな多入力加算部の構成として桁上げ保存加算器[8)](CSA, Carry Save Adder)を利用することを考える．多数項の加算は，Fig. 11に示すように2項の加算を項数回分だけ繰り返すことによって求めることができる．この2項加算の繰り返しにおいて，加算ごとに生じる各ビットの桁上げを保存し，それを次の加算ステップで加えあわせる方法を用いた多数項加算器がCSAである．Fig. 12にCSA

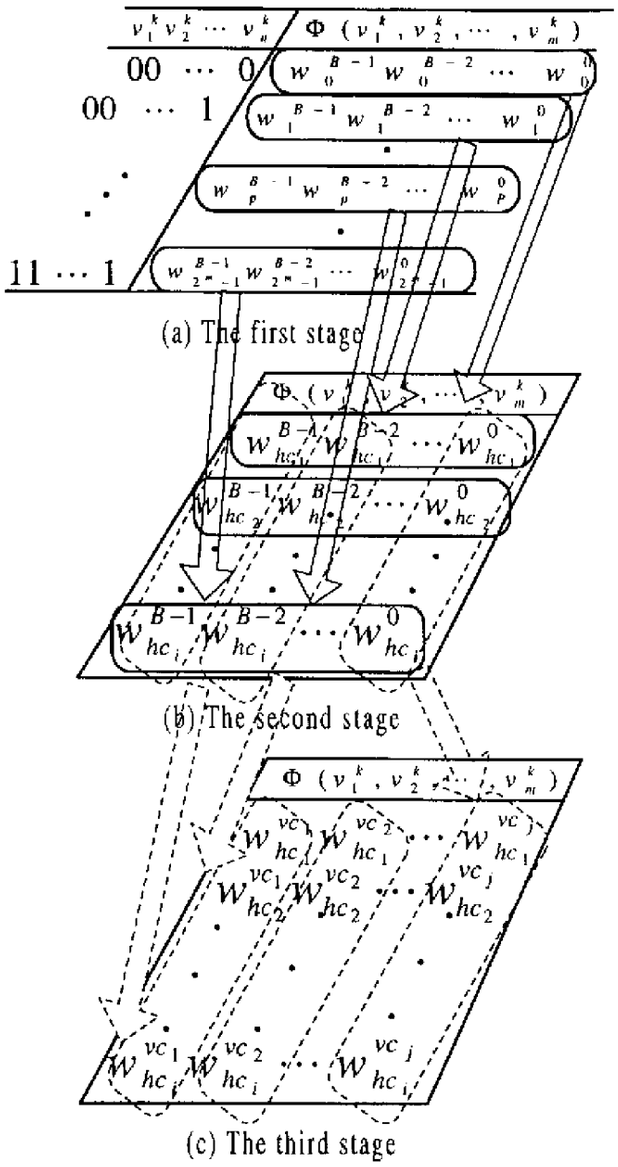

Fig. 8 Elimination of redundancy for function Φ.

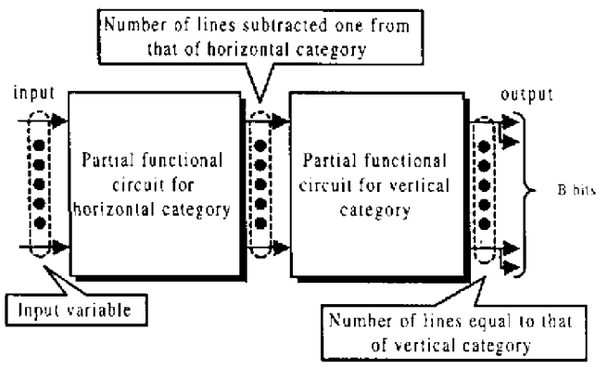

Fig. 9 Structure of optimal functional circuit.

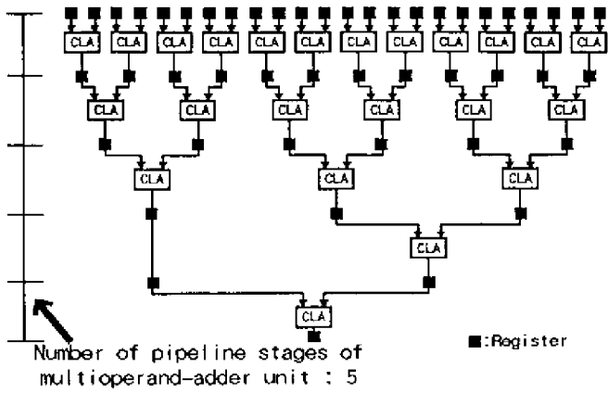

Fig. 10 Structure of binary tree using CLA in multioperand adder unit.

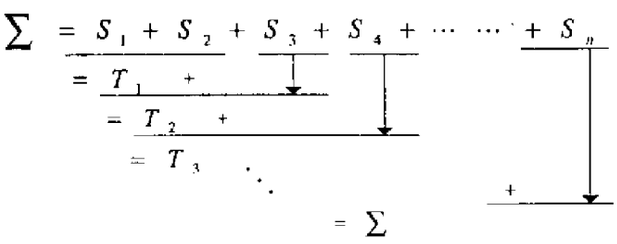

Fig. 11 n-terms addition using iteration of two-terms addition.

の基本構成図を示す．ここでは簡単な例として項数4，ビット数4の場合について示す．Fig. 12 に示すようにCSAは3入力2出力の1ビット全加算器をビット長×項数分だけ並べることによって構成できる．各1ビット全加算器によって生成された和(Sum)と桁上げ(Carry)はそれぞれ次段の1ビット全加算器に入力されるが，その際，和は同じ重みの桁の全加算器に，桁上げは1ビット位の高い桁の全加算器に入力する必要がある．最終段のデータはビット数分の和と桁上げになるため，CLAを用いて計算する必要がある．

5.2 Wallace Tree

Fig. 12 で示した構成法では項数に比例した加算段数が必要となるため，その処理時間が項数に依存してしまう．したがって，項数が増加した場合には処理時間が増大するという問題が生じる．

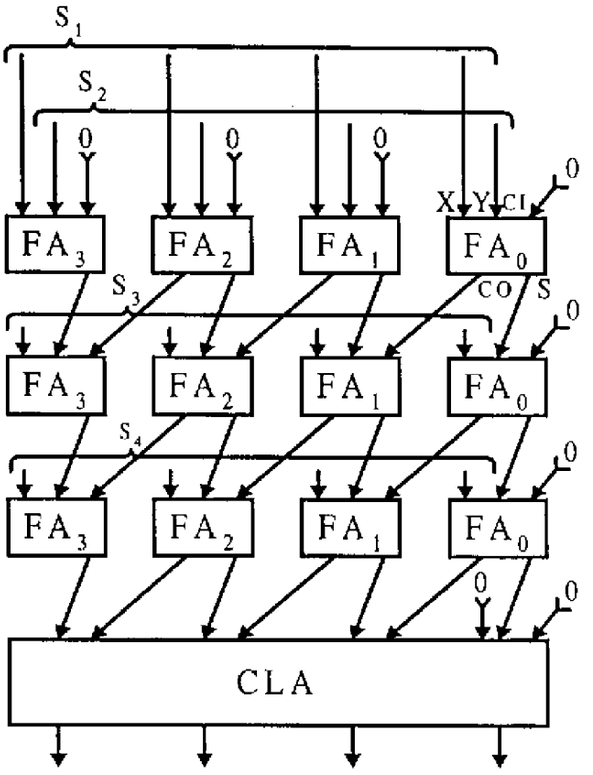

Fig. 12 Basic Structure of CSA.

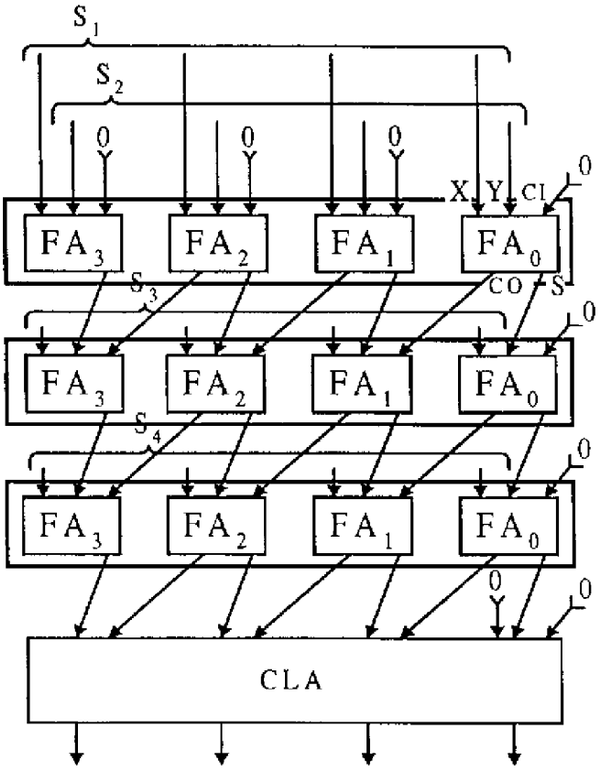

Fig. 13 Modularization for basic structure of CSA.

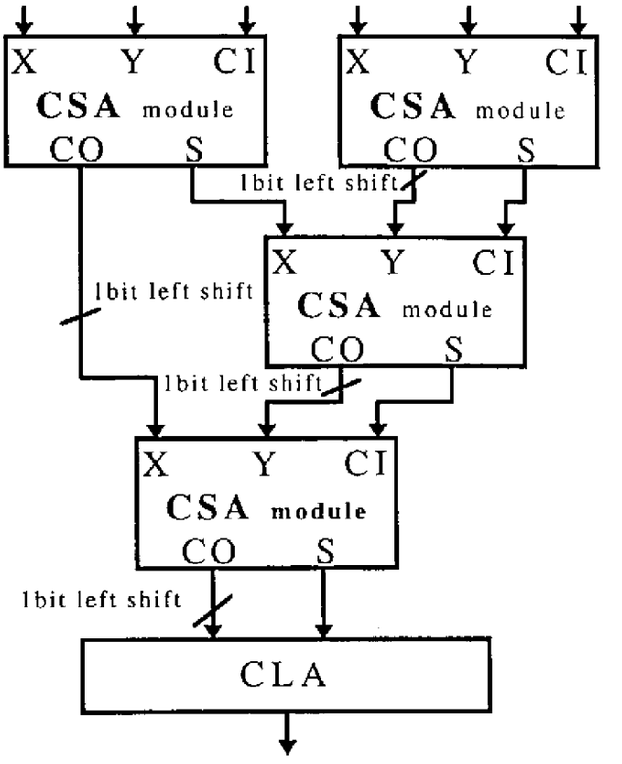

Fig. 14 Modular structure using Wallace tree.

そこでこの加算段数を効率的に減少させる手法としてWallace treeの使用がある．

まず，このWallace treeについて説明する前に新たなCSAの定義を行う．Fig. 12 で示した構成法においてFig. 13 に示すように太線で囲んだ部分を1つのモジュールとして捉えると(これをCSA Moduleと呼ぶ)，このモジュールは3入力2出力のnビットCSA(Fig. 13 では4ビットCSA)として扱うことができる．Wallace treeの構成法ではこの3入力2出力のnビットCSAをtree状に並べて部分積の総和を計算している．Fig. 14 にWallace treeの構成法の簡単な例を示す．各nビットCSAからの出力される桁上げは，次段のnビットCSAに入力する際にすべて1ビット左シフトする必要がある．また，最終段の計算に関してはFig. 13 の構成法と同様にCLAを用いて計算を行う．

このようにWallace treeの構成法では，1段の処理時間が1ビット全加算器の処理時間であるの

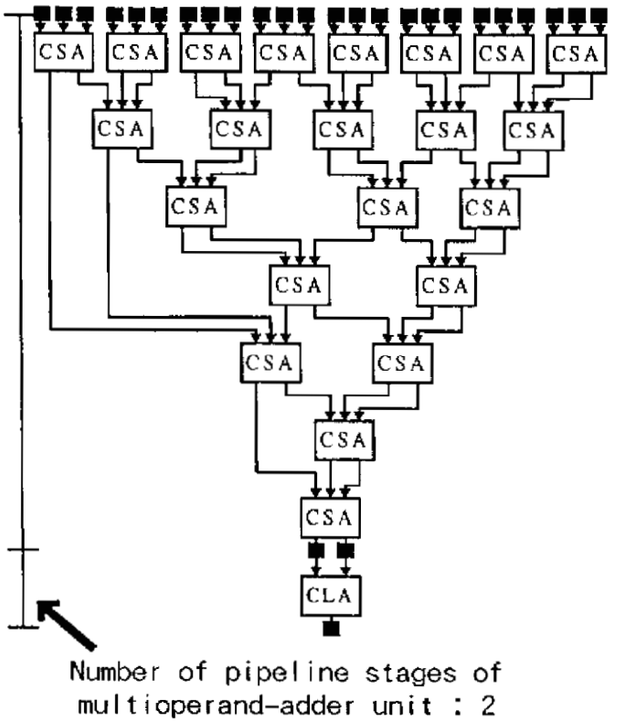

Fig. 15 Structure of Wallace tree using CSA in multioperand-adder unit.

に加え，CSAをtree状に配置することによってその加算段数を大幅に削減することができるため高速に多入力加算の総和を求めることが可能となる．Fig. 15にCSAによるWallace treeの構成法を示す．なお，Fig. 15では，パイプライン処理のピッチTの処理時間内に処理できるCSAの段数の最大の数を1つのステージとしている．Fig. 10とFig. 15よりCSAによるWallace treeの構成法では加算段数を減少できる．

6. VLSI評価

これまで述べてきた従来の構成法(ROM+CLAのbinary tree)，SFAを用いた構成法(ROM+CLAのbinary tree)，および本提案法(SFA+最適関数回路+CSAのWallace tree)に対して，VLSI設計システムPARTHENON[9]による設計および評価を行う．なお，実部品として用いたセル・ライブラリーの設計ルールは，0.8μmCMOSスタンダードセル(VLSIテクノロジ社)であり，電源電圧は5.0Vである．

また，プロセッサの設計にあたり，フィルタの仕様を与える必要がある．ここでは，Remezアルゴリズムを用いて完全な直線位相特性を実現する．この場合の仕様は，タップ数120，規格化通過域端周波数0.10，規格化阻止域端周波数0.12となる低域通過形フィルタとする．また，入出力値のデータ形式を2の補数表現による語長14ビットの固定小数点とする．ただし，プロセッサ内部の演算語長は，計算時のオーバーフローを考慮して整数部を2ビット設けた16ビットとする．

6.1 最適分割数

VLSI評価にあたりROM(最適関数回路)の分割数を決める必要がある．ここで高速性を維持しつつ最も低消費電力となる分割数を最適分割数とすると，我々は多くの経験から，ROMの最適分割数は8，最適関数回路の最適分割数は5であることを明らかにしている[1]．

6.2 評価結果

Table 1に評価結果を示す．Table 1より，従来法と比較してSFAを用いた構成法では消費電力を約38%，面積を約24%削減できた．これはSFAを用いた構成法ではROMと多入力加算部の加算器を半分にすることによって低消費電力化が可能となったためである．ただし，この構成法では入力変数の語長を1ビット拡張したためにサンプリングレートがわずかに減少し，滞在時間がわずかに増加した．さらに，この手法に対して最適関数回路を適用し，また，多入力加算の部分にCLAによるbinary treeではなくCSAによるWallace treeを用いた本提案法では，従来法と比較して，消費電力で約76%，面積で約45%の削減となった．この低消費電力化は，ROMの冗長性の削減を行い，ハードウェア量の最小化を図ることによってはじ

Table 1　評価結果(タップ数120)

	従来法	SFAを用いた構成法		乗算器を用いた直接的な構成法
	ROM	ROM	本提案法(最適関数回路)	
消費電力 [W]	5.14	3.17	1.23	15.4
面積 [mm^2]	10.1	7.64	5.60	92.9
ゲート数	50094	37769	27335	446181
サンプリングレート [MHz]	2.84	2.47	2.47	2.27
滞在時間 [ns]	525	567	513	1127

めて実現可能となる．さらにCSAによるWallace treeを用いて多入力加算部の加算段数を減少させることにより滞在時間がCLAによるbinary treeを用いたときと比べて10％減少可能となった．また，ピッチτを越えないで処理できるCSAの段数は7であった．

また，式(5)を乗算器で直接的に構成した一般的な手法では，消費電力が非常に膨大なものとなる．これは，本提案法に対して実に約13倍の消費電力となっており，このことからも本提案法が非常に効率的な実現法であることがわかる．また，本提案法はより次数の増加に対してもサンプリングレート，滞在時間をほぼ一定の値に保つことが可能となる．

7. まとめ

本報告では120タップという高次のFIRフィルタにおいて，低消費電力化を実現するために，従来のROMを用いた分散演算の手法に対し，さらなる低消費電力化のためにいくつかの手法を適用した．

SFAを用いた構成法ではFIRフィルタの直線位相特性を用いることによりハードウェア量を削減することができる．本報告ではこれを高次において適用した．また，ROMと同機能でかつROMの冗長性を削減することにより低消費電力となる最適関数回路を関数テーブルに用いた．さらに，関数テーブルを分割することにより生じる多入力加算部に対し，CLAによるbinary treeではなく，CSAによるWallace treeを用いた．

これらの手法を組み合わせることにより，従来のROMを用いた分散演算の手法に対し，本提案法では消費電力で約76％，面積で約45％の削減となった．さらに乗算器で直接的に構成した一般的な手法との比較により，本提案法の有効性が明らかになった．

参考文献

1) 恒川佳隆，野崎剛，三浦守: 滞在時間を考慮した高次FIRフィルタの高速・低消費電力形VLSIアーキテクチャ，電気学会論文誌C, vol. **118**-C, No. 7/8, 1098/1107 (1998)

2) L. Mintzer: FIR Filter with Field-programmable Gate Arrays, Journal of VLSI Signal Processing, **6**, 119/127 (1993)

3) C. F. Chen:Implementing FIR Filters with Distributed Arithmetic, IEEE Trans., Acoust. Speech & Signal Process., **ASSP-33**-4, 1318/1321 (1985)

4) C. S. Burrus:Digital Filter Structures Described by Distributed Arithmetic, IEEE Trans., Circuit & Syst., **CAS-24**, 674/680 (1977)

5) A. Peled and B. Liu:A New Hardware Realization of Digital Filters, IEEE Trans., Acoust. Speech & Signal Process., **ASSP-22**-12, 456/462 (1974)

6) 名古屋彰，中村行宏，小栗清，野村亮:高位記述からの大規模論理合成における多段論理最適化: 電子情報通信学会論文誌A, **J74-A**-2 206/217 (1991)

7) R. K. Brayton and C. McClellan: The decomposition and factorization of Boolean expressions, **ISCAS-82** 49/54 (1982)

8) 柴山潔: コンピュータアーキテクチャの基礎, 153/182, 近代科学社 (1993)

9) NTTデータ通信株式会社:PARTHNON User's Manual (1990)