計測自動制御学会東北支部 第201回研究集会 (2002.5.22)
資料番号 201-5

分散演算を用いた高性能FFTプロセッサのVLSIアーキテクチャ

VLSI Architecture of High-Performance FFT Processors Using Distributed Arithmetic

野崎　剛*，○齊藤優樹*，恒川佳隆*，田山典男*

Takeshi Nozaki*, ○Yuki Saito*, Yoshitaka Tsunekawa*, and Norio Tayama*

*岩手大学大学院 工学研究科

*Faculty of Engineering, Iwate University

キーワード： 高速フーリエ変換(FFT : Fast Fourier Transform), 分散演算 (distributed arithmetic), 低消費電力 (low power dissipation), 小面積(small area), 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
齊藤優樹，Tel.: (019)621-6468，Fax.: (019)621-6468，E-mail: t2302012@iwate-u.ac.jp

1. まえがき

近年，ディジタル信号処理の応用分野の拡大に加えて，マルチメディア関連産業の出現に伴い，我々に身近な音声・画像・映像 (動画像) などの情報表現を媒介にするアプリケーションが重要になってきている[1]．スペクトル解析の主流であるフーリエスペクトル分析，そして高速フーリエ変換(FFT : Fast Fourier Transform)は音声・画像認識アルゴリズムの基本操作であり，次世代ディジタル信号処理技術の中でますますその重要性が高まるものと考えられる．

FFTの一般的な構成法として，乗算器を用いたシグナルフローに基づく構成がある．この構成法は，滞在時間を小さくできるが，ハードウェア量が非常に大きい乗算器を多く用いるため，FFTの入力点数が多い場合，VLSIへの実現が困難となる．

本研究では，分散演算を用いた高性能FFTプロセッサのVLSIアーキテクチャを提案する．小面積化を実現するために，内積演算をハードウェア量の大きい乗算器を用いずに，処理時間が語長のみに依存する分散演算に着目する．分散演算に基づく従来型の構成法として，ROMを用いた構成が提案されている[2]．この構成法は内積演算に乗算器を用いずに分散演算を用いることでハードウェア量が小さくなる．しかし，消費電力が大きいROMを多数用いるため，システム全体に膨大な消費電力が必要となる．

大幅な低消費電力を実現するために，我々が提案してきた最適関数回路の特長[3]に着目し，小面積化と低消費電力化を実現するFFTプロセッサのVLSIアーキテクチャを提案する．この構成法は入力変数を1ビットずつ入力させることから1ビットDA(Distributed Arithmetic)FFTプロセッサと呼ぶことにする．

FFTは入力点数が多くなるにつれて段数が増える．段数に比例して滞在時間が大きくなるため，滞在時間の減少も重要である．従来の分散演算による構成の滞在時間は各入力変数を1ビットずつ伝播するため語長に比例する．そこで，我々は滞在時間を減少するために，入力変数を複数ビット入力させることに着目して，滞在時間の減少が可能なFFTプロセッサのVLSIアーキテクチャを新たに提案する．この構成法は入力変数を多ビットずつ入力させることから多ビットDAFFTプロセッサと

呼ぶことにする．最後に，2つの提案法をVLSI設計，評価した結果，1ビットDAFFTプロセッサは低消費電力化と小面積化をあわせ持った特長を有することを示し，多ビットDAFFTプロセッサは分散演算による構成法において滞在時間の短縮が可能になることを示す．そしてこれらの有効性を明らかにする．

2. FFT (Fast Fourier Transform)

FFTは離散フーリエ変換 (DFT: Discrete Fourier Transform) の演算回数を大幅に減少させる手法である[4]．DFTは次式のようにあらわされる．

$$A_k = \frac{1}{N}\sum_{n=0}^{N-1} x(n)W^{nk} \quad (k=0,1,2,\ldots,N-1) \tag{1}$$

$$W = e^{-j\frac{2\pi}{N}} \tag{2}$$

なお，$N, x(n)$とW^{nk}はそれぞれ入力点数，サンプリング値と回転因子である．FFTは，式(1)の回転因子W^{nk}の周期性と対称性を利用したアルゴリズムである．

2.1 周波数間引き型FFT

サンプリング値$x(n)$を前半と後半の2分割にする．前半の$N/2$のデータから構成されるサンプリング値を$e(n)$，後半のデータから構成されるサンプリング値を$h(n)$とする．前半のデータ$e(n)$を，

$$e(n) = x(n) \quad (0 \leqq n \leqq N/2-1) \tag{3}$$

とし，後半のデータ$h(n)$を，

$$h(n) = x(n+N/2) \quad (0 \leqq n \leqq N/2-1) \tag{4}$$

としたとき，$x(n)$の離散フーリエ変換A_kは，

$$\begin{aligned} A_k &= \frac{1}{N}\sum_{n=0}^{N-1} x(n)W^{nk} \\ &= \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1}\{e(n)W^{nk} + h(n)W^{(n+\frac{N}{2})k}\} \\ &= \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1}\{e(n)+h(n)W^{\frac{N}{2}k}\}W^{nk} \end{aligned} \tag{5}$$

となる．式(5)において$h(n)$に掛かる回転因子$W^{\frac{N}{2}k}$は，式(2)よりkが偶数のときは1となり，奇数のときは-1となる．これよりA_kの偶数番目のスペクトルをA_{2k}，奇数番目をA_{2k+1}とすると，

$$A_{2k} = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1}\{e(n)+h(n)\}W^{2nk} \tag{6}$$

$$\begin{aligned} A_{2k+1} &= \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1}\{e(n)-h(n)\}W^{n(2k+1)} \\ &= \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1}[\{e(n)-h(n)\}W^{n}]W^{2nk} \end{aligned} \tag{7}$$

$$(k=0,1,2,\ldots,N/2-1)$$

となる．

$N/2$がさらに2で割り切れるときは，4つの$N/4$のデータを持つ離散フーリエ変換に置き換えることができ，このアルゴリズムを繰り返すことにより，より高速化が実現できる．$N=8$の場合の高速フーリエ変換の処理過程を図1に示す．

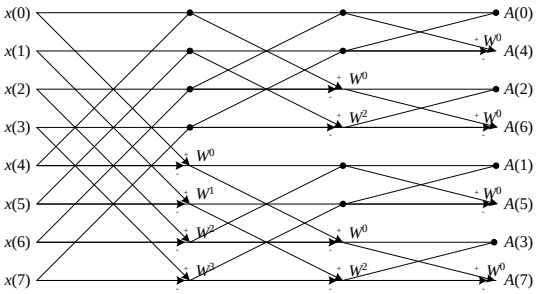


Fig. 1　周波数間引き型FFTの様子

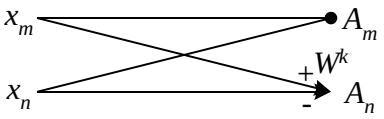


Fig. 2　周波数間引き型のバタフライ演算

2.1.1 周波数間引き型の基本演算

図1より，各段における演算は2つのデータ対の演算から構成されている．この演算をバタフライ演算という．バタフライ演算は，図2のようになっており，次式で表される．

$$A_m = x_m + x_n = x_{Rm} + x_{Rn} + j(x_{Im} + x_{In}) \tag{8}$$

$$A_n = x_m W^k - x_n W^k = (x_m - x_n)W^k \tag{9}$$

$$N = 2^k \tag{10}$$

ここで，W^kは，

$$W^k = e^{-j\frac{2\pi k}{N}} = \cos\theta_k - j\sin\theta_k \quad (11)$$

$$\theta_k = \frac{2\pi k}{N} \quad (12)$$

であるから，式(11)を式(9)に代入すると，

$$\begin{aligned} A_n = {} & (x_{Rm} - x_{Rn})\cos\theta_k \\ & +(x_{Im} - x_{In})\sin\theta_k \\ & +j\{(x_{Rm} - x_{Rn})(-\sin\theta_k) \\ & +(x_{Im} - x_{In})\cos\theta_k\} \end{aligned} \quad (13)$$

となり実部と虚部に分けられる．

乗算器を用いたFFTプロセッサにおいて，ハードウェア量が大きい乗算器を$(2N)\log_2 N$個も必要とする．このことから，入力点数が増加するに従い，システム全体のハードウェア量が膨大になってしまうために，VLSIへの実現が困難であるといえる．

そこで，次章ではハードウェア量の減少を図るため，乗算器を用いずに構成できる分散演算に着目する．

3. 分散演算アーキテクチャ

分散演算は，定係数の内積演算をテーブルルックアップによって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

$$y = \boldsymbol{a}\boldsymbol{v} = \sum_{i=1}^{N} a_i v_i \quad (14)$$

を考える．ただし，$-1 \leqq v_i < 1$ 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，

$$v_i = -v_i^0 + \sum_{k=1}^{B-1} 2^{-k} v_i^k \quad (15)$$

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(15)を式(14)に代入すると，内積演算$\boldsymbol{a}\boldsymbol{v}$は次式で示される．

$$y = -\Phi(v_1^0, \ldots, v_N^0) + \sum_{k=1}^{B-1} 2^{-k}\Phi(v_1^k, \ldots, v_N^k) \quad (16)$$

ただし，Φは

$$\Phi(v_1^k, \ldots, v_N^k) = \sum_{i=1}^{N} a_i v_i^k \quad (17)$$

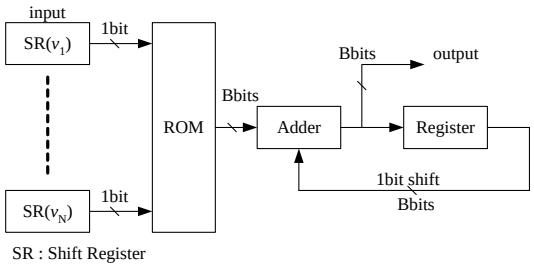


Fig. 3　分散演算の基本構造

である．ここで，分散演算の基本的な構成法を図3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ビット右シフトしながら，加え合わせる操作を行なう．このように，分散演算は乗算器を用いずに，語長B回分のシフトと累積で内積演算を行う．

3.1 分散演算を適用したFFTアルゴリズム

FFTを乗算器を用いて行う場合，FFTプロセッサのハードウェア量が大きくなってしまう．そこで，内積演算が乗算器を用いずに語長B回分のシフトと累積で計算できる分散演算をFFTに適用する．

FFTはバタフライ演算を繰り返すことで計算を行っている．バタフライ演算を表す式(8),(13)において，式(13)の実部と虚部は内積演算とみなすことができる．このことに着目して，式(13)のバタフライ演算に分散演算を適用する．$x_a = x_{Rm} - x_{Rn}, x_b = x_{Im} - x_{In}$とすると，式(13)は，

$$\begin{aligned} A_n = {} & x_a\cos\theta_k + x_b\sin\theta_k \\ & +j\{x_a(-\sin\theta_k) + x_b\cos\theta_k\} \end{aligned} \quad (18)$$

となる．x_a, x_bをBビットの固定小数点形の2の補数表示とすると，

$$x_a = -x_a^0 + \sum_{l=1}^{B-1} 2^{-l} x_a^l \quad (19)$$

$$x_b = -x_b^0 + \sum_{l=1}^{B-1} 2^{-l} x_b^l \quad (20)$$

と表される．ここで，x_a^l, x_b^lはx_a, x_bのlビット目の値で，0または1である．式(19),(20)を式(18)に代入すると，

$$A_n = -\Phi_r(x_a^0, x_b^0) + \sum_{l=1}^{B-1} 2^{-l}\Phi_r(x_a^l, x_b^l)$$

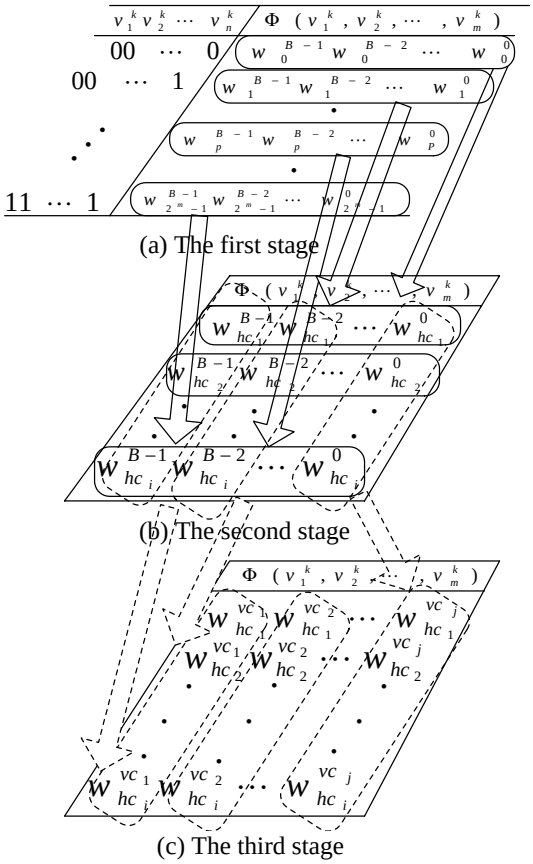


Fig. 4　関数Φに対する冗長性の削除

$$+j\{-\Phi_i(x_a^0, x_b^0) + \sum_{l=1}^{B-1} 2^{-l}\Phi_i(x_a^l, x_b^l)\} \tag{21}$$

ただし，Φ_r, Φ_iは，

$$\Phi_r(x_a^l, x_b^l) = x_a^l \cos\theta_k + x_b^l \sin\theta_k \tag{22}$$

$$\Phi_i(x_a^l, x_b^l) = x_a^l(-\sin\theta_k) + x_b^l \cos\theta_k \tag{23}$$

である．式(21),(22),(23)から，バタフライ演算に分散演算を適用できる．分散演算がFFTに用いられることにより，小面積化が実現できる．

この分散演算をFFTに適用した構成はすでに提案されている[2]．この構成では比較的大きな消費電力を要するROMが多数必要となるため，システム全体の消費電力が膨大になってしまう．

4.　本提案法

本提案法では分散演算の特長だけでなく，我々が提案している最適関数回路の特長[3]にも着目して，小面積化と低消費電力化が実現できる構成法を提案する．

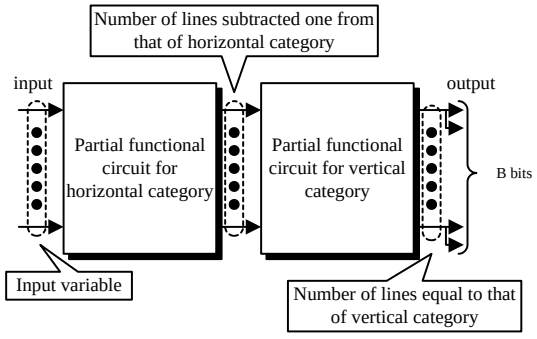


Fig. 5　最適関数回路の構成

また，最適関数回路に基づく分散演算による構成法において滞在時間の減少が可能な構成法も併せて提案する．

4.1　最適関数回路

分散演算において，関数Φの生成にROMを用いた場合，システムに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，我々が提案している最適関数回路に着目して，大幅な低消費電力化を実現できる構成を提案する．

ROMに書き込まれている関数Φのテーブルを図4(a)のように表す．ここで，$v_k^1 \cdots v_k^m$はmビッ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$\Phi(v_k^1, \cdots, v_k^m)$を$2^m \times B$の行列$W_1$とみなすと，

$$W_1 = \begin{bmatrix} w_0^{B-1} & \cdots & w_0^0 \\ \vdots & \cdots & \vdots \\ w_{2^m-1}^{B-1} & \cdots & w_{2^m-1}^0 \end{bmatrix} \tag{24}$$

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，図4(a)(b)で示すように式(24)の行ベクトルに対して共有可能となる行ベクトルを作り出すことである．その結果，生成される行列W_2は，

$$W_2 = \begin{bmatrix} w_{hc_1}^{B-1} & \cdots & w_{hc_1}^0 \\ \vdots & \cdots & \vdots \\ w_{hc_i}^{B-1} & \cdots & w_{hc_i}^0 \end{bmatrix} \tag{25}$$

となる．さらに，図4(b)(c)で示すように式(25)の列ベクトルに対しても同様な手法を施す．その結

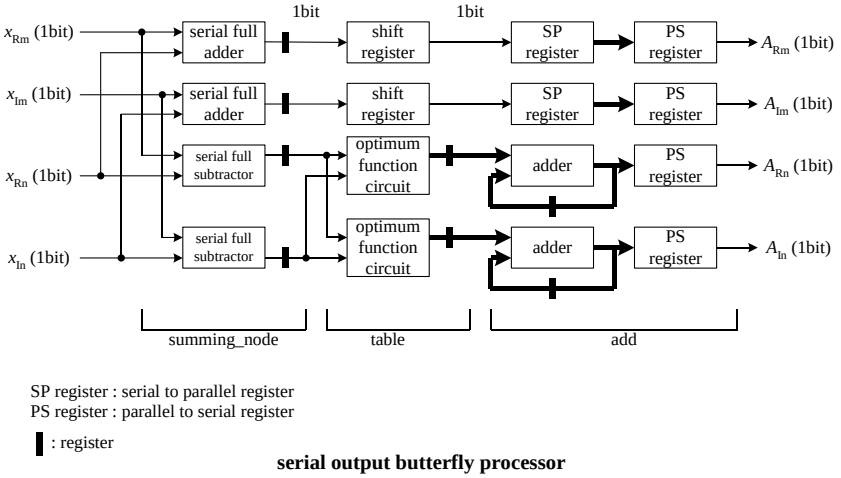


Fig. 6　1ビットDAFFTプロセッサにおけるシリアル出力バタフライ演算プロセッサの構成

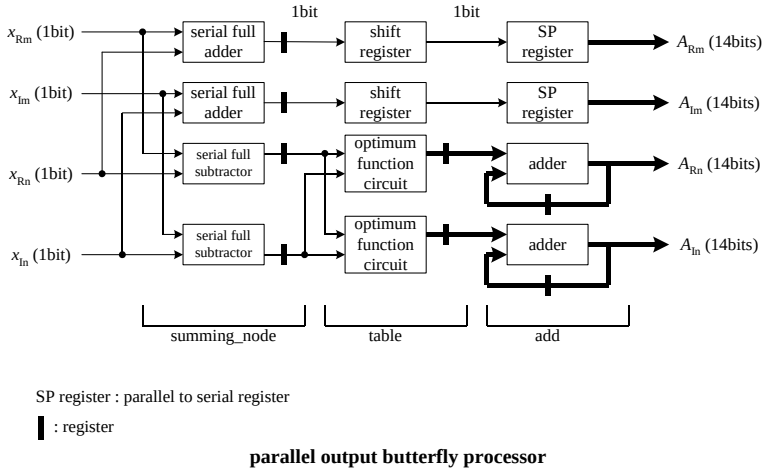


Fig. 7　1ビットDAFFTプロセッサにおけるパラレル出力バタフライ演算プロセッサの構成

果，生成される行列W_3は，

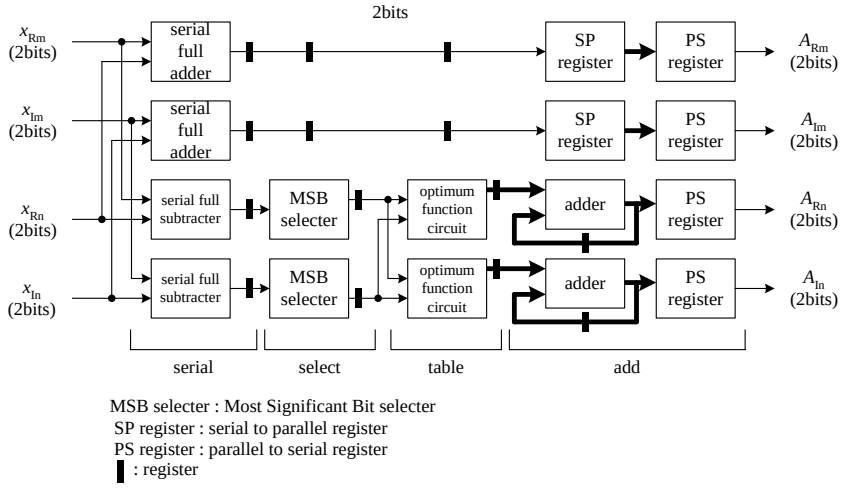
$$W_3 = \begin{bmatrix} w_{hc_1}^{vc_1} & \cdots & w_{hc_1}^{vc_j} \\ \vdots & \cdots & \vdots \\ w_{hc_i}^{vc_1} & \cdots & w_{hc_i}^{vc_j} \end{bmatrix} \tag{26}$$

となる．ここで，式(26)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，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，最適関数回路は図5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．これらの冗長性の削除によって，ゲート数を削減でき，その結果大幅に消費電力を減少できる．この最適関数回路はROMと同機能であるからFFTプロセッサに適用できる．これにより，消費電力が大幅に減少される．

4.2　最適関数回路に基づく分散演算を用いた本構成法

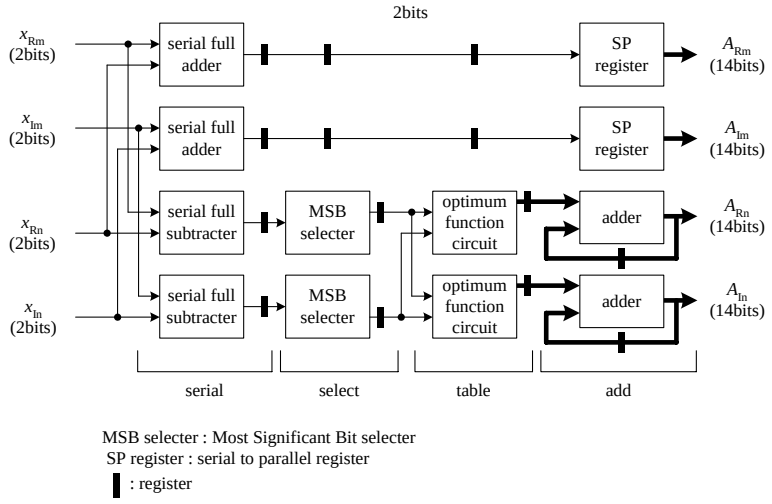
4.2.1　1ビットDAFFTプロセッサ

低消費電力化と小面積化を実現するために，最適関数回路に基づく分散演算に着目した構成法を提案する．本構成法は入力変数を1ビットずつ入力していくことから，1ビットDAFFTプロセッサと呼ぶことにする．1ビットDAFFTプロセッサのバタフライ演算プロセッサは図6,7のような構成で表される．図6の構成は，1クロックごとに1ビットずつ入出力を行う．図7の構成は，入力値が1クロックごとに1ビットずつ入力され，出力値がBビット並列に出力される．図7はFFTの計算結果が出力されるプロセッサとなり，それ以外は全て図6のプロセッサとなる．これらを図10のように構成したものが本提案法のFFTプロセッサである．このFFT



serial output butterfly processor

Fig. 8　多ビットDAFFTプロセッサにおけるシリアル出力バタフライ演算プロセッサの構成



parallel output butterfly processor

Fig. 9　多ビットDAFFTプロセッサにおけるパラレル出力バタフライ演算プロセッサの構成

プロセッサはカウンタからの信号によって制御される．

式(13)の計算は，はじめに入力変数x_mとx_nを実部と虚部においてそれぞれ減算を行い，その結果を最適関数回路に入力させる．最適関数回路には関数Φ_r, Φ_iが格納されており，図6,7において，上段の最適関数回路が関数Φ_rになり，下段の最適関数回路がΦ_iになっている．これら2つの最適関数回路からの出力を順次1ビット右シフトしながら，加算することで式(13)は計算される．この加算は同図のaddに当たる．式(8)については入力変数x_mとx_nの実部と虚部においてそれぞれ加算を行い，その結果をシフトレジスタで順次伝播することで計算している．

式(13)の内積演算に最適関数回路に基づく分散演算を用いることにより大幅な低消費電力化と小面積化が実現できる．

4.2.2　多ビットDAFFTプロセッサ

4.2.1で示した構成は，低消費電力化と小面積化が実現できるが，1ビットずつ処理を行うために，FFTプロセッサの滞在時間が大きくなるという問題が生じる．そこで，入力変数を複数ビット入力させることで滞在時間の減少が可能となることに着目した構成法を提案する．ここでは，入力変数を2ビットずつ入力して，滞在時間の減少を図る．多ビットDAFFTプロセッサのバタフライ演算プロセッサは図8,9のような構成で表される．図8の構

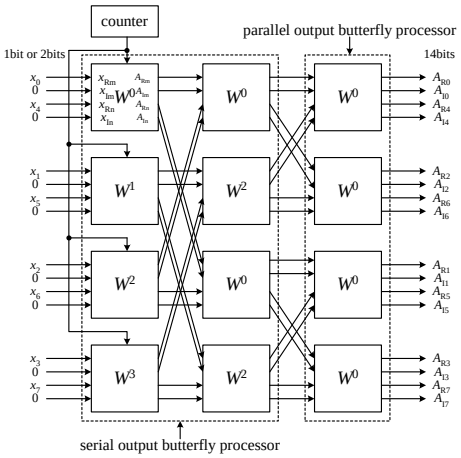


Fig. 10　FFTプロセッサの構成

成は，1クロックごとに2ビットずつ入出力を行う．図9の構成は，入力値がクロックごとに2ビットずつ入力され，出力値がBビット並列に出力される．図9はFFTの計算結果が出力されるプロセッサとなり，それ以外は全て図8のプロセッサとなる．FFTプロセッサの構成は，4.2.1で示した構成と同様に，図10のようになる．また，最適関数回路に基づく分散演算も同様に用いられている．

分散演算は式(16)に示すように，最上位ビットの入力変数と係数の内積演算の結果Φの減算を行う．そこで，入力変数を複数ビット入力させる場合，最上位ビットを判別し，最上位ビットと係数の内積演算を求める必要がある．そこで我々は，最上位ビットを判別する回路を構成した．この回路は図8,9の MSB selecter に当たる．MSB selecter はカウンタからの信号で制御される．最上位ビットを含んだ複数ビットが MSB selecter に入力すると，MSB selecter は最上位ビットとそれ以外のビットに分ける．そして，MSB selecter は最上位ビットに0を結合した複数ビットを入力変数の最後の出力にする．この最上位ビットに0を結合させた複数ビットと係数との内積演算の結果Φが最適関数回路から出力される．その値は加算器に入力する際にビット反転を行う．このビット反転を行った値を加減算器で処理することにより分散演算が可能となる．

このような処理を行うことで，分散演算に基づく構成において入力変数を複数ビットにすることが可能となり，滞在時間の短縮が実現される．

5.　VLSI設計，評価

これまでに提案した2つの構成法の性能を明らかにするためにVLSI設計システムPARTHENONを用いてVLSI設計，評価を行う．なお，用いた設計ルールは0.6 μmCMOS スタンダードセルであり，電源電圧は5.0 [V] である．

本構成のFFTプロセッサは入出力値のデータ形式を2の補数表示による語長14ビットの固定小数点とする．ただし，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．なお，FFTのデータ点数は8点とした．

本研究で提案した2つの構成，関数Φ_r, Φ_iの生成にROMを用いた従来型分散演算に基づく構成と乗算器を用いた一般的な構成の評価結果を表1に示す．なお，使用した乗算器は，部分積の生成にBoothのアルゴリズムと部分積の加算にWallaceの方式とCLA加算器を用いたもので，16×16ビット乗算器である．

5.1　1ビットDAFFTプロセッサの評価

表1より，1ビットDAFFTプロセッサの消費電力はROMを用いた従来型分散演算に基づく構成の約75%，乗算器を用いた構成の約88%削減された．これは最適関数回路の消費電力がROMや乗算器の消費電力に比べて非常に小さいためである．この評価結果から最適関数回路を用いた分散演算に基づく構成は大幅な低消費電力化が可能であるといえる．

面積を比較すると，ROMを用いた構成の約53%，乗算器を用いた構成の約87%削減された．これは最適関数回路のゲート数はROMや乗算器のゲート数よりも少ないからである．この評価結果から最適関数回路を用いた分散演算に基づく構成は小面積化が可能であるといえる．

これらの結果から，最適関数回路を用いた分散演算に基づく1ビットDAFFTプロセッサは低消費電力化および小面積化に対する効果が大きいという特長を示した．

5.2　多ビットDAFFTプロセッサの評価

2ビットDAFFTプロセッサの滞在時間は，ROMを用いた構成の約30%，1ビットDAFFTプロセッサの構成の約26%削減された．これは入力変数を複数ビット入力させることで滞在時間の短縮を図っ

Table 1　FFTプロセッサの評価

	本提案法(最適関数回路)		ROM	乗算器
	1ビットDAFFT	2ビットDAFFT		
消費電力[mW]	803.5	843.1	3158.1	6458.9
面積[mm^2]	3.45162	4.06191	7.30927	27.03559
ゲート数	29044	33947	42949	243439
マシンサイクル[ns]	29	32	29	46
サンプリングレート[MHz]	2.5	3.9	2.5	21.7
滞在時間[ns]	1334	992	1421	332

たためである．この評価結果より，入力変数を複数ビット入力させた分散演算に基づく構成は滞在時間の短縮が可能であるといえる．滞在時間の短縮により2ビットDAFFTプロセッサのサンプリングレートはROMを用いた構成と1ビットDAFFTプロセッサの構成の約1.6倍となり，多ビットDAFFTプロセッサの有効性が明らかとなった．

また，2ビットDAFFTプロセッサの消費電力はROMを用いた従来型分散演算に基づく構成の約73%，乗算器を用いた構成の約87%削減された．面積を比較すると，ROMを用いた構成の約44%，乗算器を用いた構成の約85%削減された．この結果からも最適関数回路を用いた分散演算に基づく構成は低消費電力化および小面積化に対する効果が大きいといえる．

6. むすび

本研究では分散演算を用いた高性能FFTプロセッサのVLSIアーキテクチャを提案した．本提案法では，内積演算を乗算器を用いずに行うことができる分散演算に最適関数回路を用いることにより，小面積化と低消費電力化を図った．また最適関数回路を用いた分散演算において，入力変数を複数ビット入力させることで滞在時間の短縮を図った．小面積化と低消費電力化を図った提案法を1ビットDAFFTプロセッサと，滞在時間の短縮を図った提案法を多ビットDAFFTプロセッサとをそれぞれ設計し，その評価を行った．その結果，1ビットDAFFTプロセッサはROMを用いた従来型分散演算に基づく構成法と乗算器を用いた構成法に対して大幅な低消費電力化と小面積化が可能となった．また，多ビットDAFFTプロセッサは，分散演算を用いた構成のどれよりも滞在時間が減少できた．以上のことから，我々が提案した構成法の有効性が明らかとなった．

今後の課題として，多ビットDAの入力変数を2ビット以外で検討する必要がある．

参考文献

1) 青木孝文，天田博章，樋口龍雄: "冗長複素数系に基づく実数/複素数再構成型算術演算回路の構成", 電子情報通信学会論文誌D-I, Vol.J80-D-I, No.8, pp.674-682, 1997.8

2) Les Mintzer: "The Xilinx FPGA as an FFT processor", Electronic Engineering Vol.69 , No.845, pp.81-84, 1997

3) 野崎剛，恒川佳隆，三浦守: "滞在時間を考慮した高次FIRフィルタの高速・低消費電力形アーキテクチャ", 電気学会論文誌C, vol.118, 7/8, 1998

4) A.V.Oppenheim and R.W.Schafer: "Disital Signal Processing", Prentice-Hall

5) "PARTHENON User's Manual", NTT データ通信株式会社, 1990