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

2008－SLDM－135（5）
2008／5／8

# セレクタ論理を用いたバタフライ演算器の設計

名村 健† 戸川 望† 柳澤 政生† 大附 辰夫† 外村 元伸††

† 早稲田大学大学院基幹理工学研究科情報理工学専攻
〒169–8555 東京都新宿区大久保 3–4–1
†† 大日本印刷株式会社電子モジュール開発センター
〒808–0135 福岡県北九州市若松区ひびきの 2–5
E-mail: †namura@ohtsuki.comm.waseda.ac.jp

**あらまし** 算術演算回路の処理を高速化する手法として，セレクタ論理を利用した算術演算回路が提案されている．本稿では，FFT におけるバタフライ演算式を式変形し，セレクタ論理に帰着させることで桁上げ伝播処理を削減することによって可変点数に対応した新しいバタフライ演算回路構成を提案する．評価実験をした結果，提案したバタフライ演算器は，算術演算子を用いて設計した従来のバタフライ演算構造に比べ，速度優先設計で 21.8% 高速化することができることを確認した．

**キーワード** セレクタ論理，バタフライ演算，FFT，OFDM，

# Radix-2 Butterfly Circuit Architecture Using Selector Logic

Takeshi NAMURA†, Nozomu TOGAWA†, Masao YANAGISAWA†, Tatsuo OHTSUKI†, and Motonobu TONOMURA††

† Dept. of Computer Science and Engineering, Waseda University
3–4–1 Okubo, Shinjuku, Tokyo, 169–8555 Japan
†† Electronic Module Development Center, Dai Nippon Printing Corporation
2–5 Hibikino, Wakamatsu, Kitakyushu, Fukuoka, 808–0135 Japan
E-mail: †namura@ohtsuki.comm.waseda.ac.jp

**Abstract** An arithmetic circuit using selector logic has been proposed, as a high computational approach for processing. In this paper, we propose a radix-2 butterfly circuit architecture using selector logic. Our butterfly circuit reduces carry propagations, compared to conventional butterfly circuits. Experimental results show that our proposed butterfly circuit improves the performance by 21.8% , compared to conventional butterfly circuits.

**Key words** selector logic, butterfly circuit, FFT, OFDM

## 1. まえがき

今日，携帯電話やディジタル・カメラ，DVD レコーダといったシステム LSI の多くで，画像処理や音声処理，エラー訂正処理や暗号処理など特定の処理に特化した専用回路が組み込まれている．これらのシステム LSI には，高性能，小面積，低消費電力であることが要求される．この要求に対して組み込まれる専用回路を効率的に設計することは効果的なアプローチの 1 つである．

セレクタ論理は算術演算回路の部分積生成回路における出力数を削減し，桁上げ伝播処理を削減する手法であり，特別な演算式構成を持っている場合 (2.2 節参照) に適応できる [10]．部分積生成回路の出力数が削減することにより，2 進数表現をもとに設計した演算回路と比較して桁上げ伝播処理が削減し高速化することができる．

FFT は離散フーリエ変換における計算を三角関数の周期性を利用して高速に行う手法であり，さまざまな分野において用いられる演算である．これまで，アルゴリズムを工夫することによって複素乗算回数を削減する手法 [4] やパイプライン構成を用いた手法 [1]，さまざまなフーリエ変換点数に対応した可変構造の FFT が提案されてきた．可変構造の FFT における演算回路の構成として，バタフライ演算回路を複数個用いてパイプライン演算をする構成 [2] や一つのバタフライ演算回路を繰り返し使用する構成が提案されてきた [7]．これらの構成にお

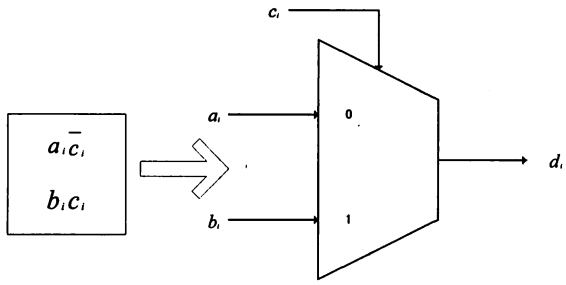


図 1　部分積項におけるセレクタ論理

いて FFT は，Radix-2 と Radix-4 のバタフライ演算器を組み合わせて実現しているが，それぞれのバタフライ演算器はすでに確立されているためアルゴリズムを工夫することは難しい．可変構造の FFT 演算回路は OFDM 変調器等に用いられている [11].

本稿では，FFT における Radix-2 バタフライ演算回路をセレクタ論理に帰着させ，桁上げ伝播処理を削減し高速化することによって可変点数にも対応した新しいバタフライ演算回路構成を提案する．バタフライ演算回路に対し式変形を行った 2 進数表現出力値をもとに，部分積生成回路をセレクタ回路，部分積加算回路は Wallace Tree を用いる．従来の Cooley-Tukey の Radix-2 バタフライ演算回路と比較を行った結果，遅延時間が 21.8% 削減された．

## 2. セレクタ論理帰着型演算回路

演算回路のビットレベル処理手法において，これまで 2 進数表現を用いた回路 [9]，冗長 2 進数表現を用いた回路 [6]，DA (Distributed Arithmetic) 法を用いた演算回路 [8] 等がある．本章では，ビットレベル処理手法の一つとしてセレクタ論理を取り上げる．

### 2.1 セレクタ論理

入力信号を $a$，$b$，制御信号を $c$ とした時のセレクタ論理は $\bar{c}$ を $c$ の論理反転値とすると，$a \wedge \bar{c} \vee b \wedge c$ と表すことができる．演算において $a_i \wedge \bar{c_j} \vee b_i \wedge c_j$ のような部分積項が存在すると，セレクタ論理に帰着することができ，2 つの部分積は桁上げをしない．そのため桁上げ伝播処理を削減することができ，高速化することができる．部分積項におけるセレクタ論理を図 1 に示す．部分積項をセレクタ論理に帰着した回路構成は，2 進数表現をもとに設計した回路構成と比較し部分積生成回路において出力数が削減できる．また，冗長 2 進数表現のような符号化，復号化処理を必要としない．

### 2.2 2 の補数表現

2.1 節で取り上げたセレクタ論理に帰着するには論理反転値を求めるための式変形が必要である．論理反転値を求めるため，符号なし数では 1 から固定小数点数を引くことによって，2 の補数符号付き数では負の数を 2 の補数表現を用いて表す．$D$ を固定小数点数とすると，論理反転値を求める式変形は以下のように行うことができる．尚，$\bar{D}$ は $D$ の論理反転値とする．

符号なし数 ($0 \leqq D \leqq 1$)：

$$(1-D) = \bar{D} + [0.0\ldots01] \quad (1)$$

2 の補数符号付き数 ($-1 \leqq D < 1$)：

$$-D = \bar{D} + [0.0\ldots01] \quad (2)$$

このことから，算術演算式が以下のような式構成を持っている場合，演算式は 2.1 節で説明したセレクタ論理に帰着できるため桁上げ伝播処理を削減でき高速化することが期待される．

**符号なし数**

符号なし数の場合 $n$ ビットの入力 $A$，$B$ と $m$ ビットの入力 $D$ の演算において，$A$，$B$，$D$ はそれぞれ

$$A = [a_0.a_{-1}\ldots a_{n-2}a_{n-1}] = 2^{-n}\sum_{i=0}^{n-1} a_i 2^i \quad (a_i \in \{0,1\})$$

$$B = [b_0.b_{-1}\ldots b_{n-2}b_{n-1}] = 2^{-n}\sum_{i=0}^{n-1} b_i 2^i \quad (b_i \in \{0,1\})$$

$$D = [d_0.d_{-1}\ldots d_{m-2}d_{m-1}] = 2^{-m}\sum_{i=0}^{m-1} d_i 2^i \quad (d_i \in \{0,1\})$$

と表すことができる．符号なし数で $(1-D)$ は論理反転値 $\bar{D}$ を用いて式 (1) のように変形できることから，$(1-D)A$，$DB$ の項が同時に含まれている演算式を 2.1 節で説明したセレクタ論理に帰着させることができる．$(1-D)A + DB$ は以下のように式変形できる．

$$\begin{aligned}
&(1-D)A + DB \\
&= 2^{-m}\left(1 + \sum_{i=0}^{m-1}\bar{d}_i 2^i\right)\left(2^{-n}\sum_{j=0}^{n-1} a_j 2^j\right) \\
&\quad + \left(2^{-m}\sum_{i=0}^{m-1} d_i 2^i\right)\left(2^{-n}\sum_{j=0}^{n-1} b_j 2^j\right) \\
&= 2^{-(m+n)}\left\{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} a_j\bar{d}_i 2^{i+j} + \sum_{j=0}^{n-1} a_j 2^j\right\} \\
&\quad + 2^{-(m+n)}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} b_j d_i 2^{i+j} \\
&= 2^{-(m+n)}\left\{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(a_j\bar{d}_i + b_j d_j)2^{i+j}\right\} \\
&\quad + 2^{-(m+n)}\sum_{j=0}^{n-1} a_j 2^j \\
&= \bar{D}A + DB + A2^{-m} \qquad (3)
\end{aligned}$$

もし，演算式 $(1-D)A + DB$ を図 2(a) のようにそのまま実現すると減算器，乗算器と加算器を縦続に接続することになる．この場合減算，乗算の最終加算，加算それぞれに対して桁上げ (桁下げ) 伝播処理が必要となる．しかし，式 (3) のように 2 進数表現ベースで式変形を行いセレクタ論理に帰着させると図 2(b) のように桁上げ伝播処理が削減し，乗算器 1 個とほぼ同程度の計算時間で演算回路を設計できる．

**2 の補数符号付き数**

2 の補数符号付き数の場合 $n$ ビットの入力 $A$，$B$ と $m$ ビットの入力 $C$ の演算において，$A$，$B$，$C$ はそれぞれ

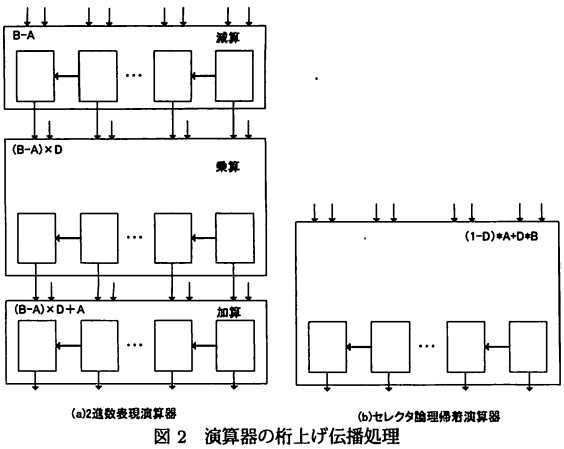


図 2　演算器の桁上げ伝播処理

$$
\begin{aligned}
A &= [a_0.a_{-1}\ldots a_{n-2}a_{n-1}] \\
&= 2^{-n}\left(-a_{n-1}2^{n-1}+\sum_{i=0}^{n-2}a_i2^i\right) \quad (a_i\in\{0,1\}) \\
B &= [b_0.b_{-1}\ldots b_{n-2}b_{n-1}] \\
&= 2^{-n}\left(-b_{n-1}2^{n-1}+\sum_{i=0}^{n-2}b_i2^i\right) \quad (b_i\in\{0,1\}) \\
C &= [c_0.c_{-1}\ldots c_{m-2}c_{m-1}] \\
&= 2^{-m}\left(-c_{m-1}2^{m-1}+\sum_{i=0}^{m-1}c_i2^i\right) \quad (c_i\in\{0,1\})
\end{aligned}
$$

と表すことができる．2 の補数符号付き数 $-C$ は論理反転値 $\bar{C}$ を用いて式 (2) のように変形できることから，演算式において $CA$，$-CB$ の項が同時に含まれていればセレクタ論理に帰着させることができる．$(A-B)C$ の式変形は以下のように行う．

$$
\begin{aligned}
&(A-B)C = AC+B(-C) \\
&= 2^{-(n+m)}\left(-a_{n-1}2^{n-1}+\sum_{i=0}^{n-2}a_i2^i\right)\left(-c_{m-1}2^{m-1}\right. \\
&\left.+\sum_{i=0}^{m-2}c_i2^i\right)+2^{-(n+m)}\left(-b_{n-1}2^{n-1}+\sum_{i=0}^{n-2}b_i2^i\right) \\
&\left(-\bar{c}_{m-1}2^{m-1}+\sum_{i=0}^{m-2}\bar{c}_i2^i+1\right) \\
&= \Bigg\{-(a_{n-1}\vee c_{m-1}+b_{n-1}\vee\bar{c}_{n-1})2^{m+n-2} \\
&\quad+\sum_{i=0}^{n-2}\left(c_{m-1}\bar{a}_i+\bar{c}_{m-1}\bar{b}_i\right)2^{m+i-1} \\
&\quad+\sum_{i=0}^{m-2}\left(a_{n-1}\bar{c}_i+b_{n-1}c_i\right)2^{n+i-1} \\
&\quad+2^{m-1}+a_{n-1}2^{n-1}+\sum_{i=0}^{n-2}b_i2^i \\
&\quad+\sum_{i=0}^{n-2}\sum_{j=0}^{m-2}(a_ic_j+b_i\bar{c}_j)2^{i+j}\Bigg\}2^{-(n+m)}
\end{aligned}
$$

符号付き数の場合 $n$ ビットの入力 $A$，$B$ と $m$ ビット入力の演算で演算式 $(A-B)C$ をそのまま設計すると，減算を行った結果を用いて乗算を行うため減算，乗算において桁上げ伝播処理を行う．一方，式変形を施しセレクタ論理に帰着させることで桁上げ伝播処理が削減でき高速化することが期待される．

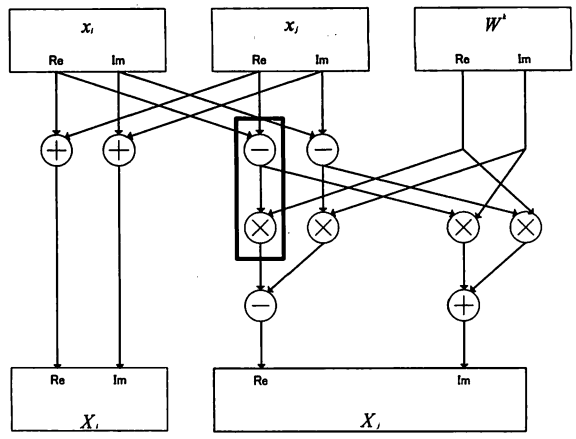


図 3　Radix-2 バタフライ演算の構成図

## 3. セレクタ論理を用いたバタフライ演算器の設計

バタフライ演算は複素乗算や FFT 演算などで頻繁に用いられるアルゴリズムであり，FFT 演算回路では主に Radix-4 または Radix-2 のバタフライ演算回路が用いられている．以下に Radix-2 バタフライ演算回路を示しセレクタ論理の適応について検討したのち，セレクタ論理にもとづくバタフライ演算回路構成を提案する．

### 3.1 Radix-2 バタフライ演算

Radix-2 バタフライ演算は複素数 2 入力に対して複素数 2 出力の計算が行われ．演算式は，$N$ 点の解析信号 $x(n)$ を $N$ 点の $X(k)$ に変換する DFT の式を変換して式 (4) のように表すことができる [3]．

$$
X(k)=\sum_{n=0}^{\frac{N}{2}-1}\{x(n)+(-1)^kx(n+\tfrac{N}{2})\}{W_N}^{nk} \qquad (4)
$$

Radix-2 バタフライ演算はデータ点数が $2^M$ ($M$：整数) である時に適応できる．

Radix-2 バタフライ演算の構成図として，Cooley-Tukey 型演算構成を図 3 で示す [5]．Cooley-Tukey 型演算は複素数 2 入力に対して，6 回の加減算と 4 回の乗算が行われている．ここで，$x_i$ は偶数番目の入力信号，$x_j$ は奇数番目の入力信号を表す．また $W^k$ は回転因子を表し，計算によって求めると多くのハードウェアコストが必要となるため，あらかじめ計算して LUT(Look-Up Table) に書き込んでおく．

ここで，図 3 を見ると，囲われている部分において減算，乗算を縦列に行う式構成を持っていることが分かる．すなわち，Cooley-Tukey 型演算回路は 2.2 節における $(A-B)C$ 演算と同様に，2 の補数表現を用いて式変形を行うことでセレクタ論理に帰着できると考えられる．

### 3.2 提案する Radix-2 バタフライ演算回路

Cooley-Tukey 型の Radix-2 バタフライ演算に対して以下のような式変形を行う．まず，3.1 節の図 3 において，Re $x_i$，Im $x_i$，Re $x_j$，Im $x_j$，Re $W^k$，Im $W^k$ は $n$ ビットの符号付き固定小数点数であるとする．つまり，これらは式 (5) のように表すことができる．

$$\left.\begin{aligned}
\text{Re } x_i &= -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i \\
\text{Im } x_i &= -b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i \\
\text{Re } x_j &= -c_{n-1}2^{n-1} + \sum_{i=0}^{n-2} c_i 2^i \\
\text{Im } x_j &= -d_{n-1}2^{n-1} + \sum_{i=0}^{n-2} d_i 2^i \\
\text{Re } W^k &= -e_{n-1}2^{n-1} + \sum_{i=0}^{n-2} e_i 2^i \\
\text{Im } W^k &= -f_{n-1}2^{n-1} + \sum_{i=0}^{n-2} f_i 2^i
\end{aligned}\right\} \quad (5)$$

式 (5) を 1 ビット符号拡張した数を式 (6) で表す．

$$\left.\begin{aligned}
\text{Re } x_i &= -a_{n-1}2^{n} + \sum_{i=0}^{n-1} a_i 2^i \\
\text{Im } x_i &= -b_{n-1}2^{n} + \sum_{i=0}^{n-1} b_i 2^i \\
\text{Re } x_j &= -c_{n-1}2^{n} + \sum_{i=0}^{n-1} c_i 2^i \\
\text{Im } x_j &= -d_{n-1}2^{n} + \sum_{i=0}^{n-1} d_i 2^i \\
\text{Re } W^k &= -e_{n-1}2^{n} + \sum_{i=0}^{n-1} e_i 2^i \\
\text{Im } W^k &= -f_{n-1}2^{n} + \sum_{i=0}^{n-1} f_i 2^i
\end{aligned}\right\} \quad (6)$$

式 (6) から，出力値 Re $X_j$ は式 (7) のように式変形することができる．

$$\begin{aligned}
\text{Re } X_j &= (\text{Re } x_i - \text{Re } x_j)\text{Re } W^k \\
&\quad -(\text{Im } x_i - \text{Im } x_j)\text{Im } W^k \\
&= \left(-a_{n-1}2^n + \sum_{i=0}^{n-1} a_i 2^i\right)\left(-e_{n-1}2^n \right. \\
&\quad \left. + \sum_{i=0}^{n-1} e_i 2^i\right) + \left(-c_{n-1}2^n + \sum_{i=0}^{n-1} c_i 2^i\right) \\
&\quad \left(-\bar{e}_{n-1}2^n + \sum_{i=0}^{n-1} \bar{e}_i 2^i + 1\right) \\
&\quad + \left(-d_{n-1}2^n + \sum_{i=0}^{n-1} d_i 2^i\right)\left(-f_{n-1}2^n \right. \\
&\quad \left. + \sum_{i=0}^{n-1} f_i 2^i\right) + \left(-b_{n-1}2^n + \sum_{i=0}^{n-1} b_i 2^i\right) \\
&\quad \left(-\bar{f}_{n-1}2^n + \sum_{i=0}^{n-1} \bar{f}_i 2^i + 1\right)
\end{aligned}$$

$$\begin{aligned}
\text{Re } X_j &= (a_{n-1}e_{n-1} + c_{n-1}\bar{e}_{n-1} + d_{n-1}f_{n-1} \\
&\quad + b_{n-1}\bar{f}_{n-1})2^{2n} \underline{- a_{n-1}2^n \sum_{i=0}^{n-1} e_i 2^i} \\
&\quad \underline{- e_{n-1}2^n \sum_{i=0}^{n-1} a_i 2^i - \bar{e}_{n-1}2^n \sum_{i=0}^{n-1} c_i 2^i} \\
&\quad \underline{- c_{n-1}2^n \sum_{i=0}^{n-1} \bar{e}_i 2^i - d_{n-1}2^n \sum_{i=0}^{n-1} f_i 2^i} \\
&\quad \underline{- f_{n-1}2^n \sum_{i=0}^{n-1} d_i 2^i - \bar{f}_{n-1}2^n \sum_{i=0}^{n-1} b_i 2^i} \\
&\quad \underline{- b_{n-1}2^n \sum_{i=0}^{n-1} \bar{f}_i 2^i} + \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(a_i e_j + \\
&\quad c_i \bar{e}_j + d_i f_j + b_i \bar{f}_j)2^{i+j} - c_{n-1}2^n \\
&\quad - b_{n-1}2^n + \sum_{i=0}^{n-1}(b_i + c_i)2^i \qquad (7)
\end{aligned}$$

式 (7) の下線部分において，負項から正項に変換するため式 (8) を用いる．

$$\begin{aligned}
-\sum_{i=0}^{n-1} s_i 2^i &= \sum_{i=0}^{n-1}(1 - s_i)2^i - \sum_{i=0}^{n-1} 2^i \\
&= \sum_{i=0}^{n-1} \bar{s}_i 2^i + 1 - 2^n \qquad (8)
\end{aligned}$$

式 (8) から出力値 Re $X_j$ は以下の式で与えられる．

$$\begin{aligned}
\text{Re } X_j &= (a_{n-1}e_{n-1} + c_{n-1}\bar{e}_{n-1} + d_{n-1}f_{n-1} \\
&\quad + b_{n-1}\bar{f}_{n-1} - a_{n-1} - e_{n-1} - c_{n-1} \\
&\quad - \bar{e}_{n-1} - d_{n-1} - f_{n-1} - b_{n-1} - \bar{f}_{n-1})2^{2n} \\
&\quad + \sum_{i=0}^{n-1}(e_{n-1}\bar{a}_i + \bar{e}_{n-1}\bar{c}_i + a_{n-1}\bar{e}_i + c_{n-1}e_i \\
&\quad + d_{n-1}\bar{f}_i + b_{n-1}f_i + f_{n-1}\bar{d}_i + \bar{f}_{n-1}\bar{b}_i)2^{n+i} \\
&\quad + 2(e_{n-1} + \bar{e}_{n-1})2^n + (d_{n-1} + a_{n-1})2^n \\
&\quad + \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(a_i e_j + c_i \bar{e}_j + d_i f_j + b_i \bar{f}_j)2^{i+j} \\
&\quad + \sum_{i=0}^{n-2}(b_i + c_i)2^i \\
&= -(a_{n-1} \vee e_{n-1} + c_{n-1} \vee \bar{e}_{n-1} \\
&\quad + d_{n-1} \vee f_{n-1} + b_{n-1} \vee \bar{f}_{n-1})2^{2n} \\
&\quad + \sum_{i=0}^{n-1}(e_{n-1}\bar{a}_i + \bar{e}_{n-1}\bar{c}_i + a_{n-1}\bar{e}_i + c_{n-1}e_i \\
&\quad + d_{n-1}\bar{f}_i + b_{n-1}f_i + f_{n-1}\bar{d}_i + \bar{f}_{n-1}\bar{b}_i)2^{n+i} \\
&\quad + 2^{n+1} + (d_{n-1} + a_{n-1})2^n + \sum_{i=0}^{n-1}(b_i + c_i)2^i \\
&\quad + \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(a_i e_j + c_i \bar{e}_j + d_i f_j + b_i \bar{f}_j)2^{i+j} \qquad (9)
\end{aligned}$$

このように式変形することで，バタフライ演算は以下のようにセレクタ論理に帰着した演算器構成を取ることができる．

**セレクタ論理を用いた Radix-2 バタフライ演算構成**

セレクタ論理を用いた Radix-2 バタフライ演算構成は，式 (9) で示された計算結果を 0 から $(2n-1)$ 桁までそれぞれの桁ごと，式 (9) で示された計算結果をもとに加算する．このとき桁上げ信号を上位桁に伝播する．$2n$ 桁 (最上位桁) は最上位桁に

ある項と下位桁からの桁上げ信号全ての XOR を取る回路構成である．

[例 1] 以下に $n=4$ の例を示す．図 4 は式 (9) において $n=4$ を代入し，各桁ごとに部分積をまとめたもの表す．

（1） $i$ =0($2^0$ の桁) のとき $c_0\bar{e_0}$，$a_0e_0$，$b_0\bar{f_0}$，$d_0f_0$，$b_0$，$c_0$ の 6 項の加算を行い $2^1$ 桁に桁上げ信号を伝播する．$c_0\bar{e_0}$，$a_0e_0$ 項と，$b_0\bar{f_0}$，$d_0f_0$ 項においてセレクタ論理を用いることができる．

（2） $i=1$($2^1$ の桁) のとき 10 項及び $2^0$ 桁からの桁上げ信号の加算を行い，$2^2$ 桁に桁上げ信号を伝播する．8 項においてセレクタ論理を用いることができる．

（3） $i=2$($2^2$ の桁) のとき 14 項及び $2^1$ 桁からの桁上げ信号の加算を行い，$2^3$ 桁に桁上げ信号を伝播する．12 項においてセレクタ論理を用いることができる

（4） $i=3$($2^3$ の桁) のとき 18 項及び $2^2$ 桁からの桁上げ信号の加算を行い，$2^4$ 桁に桁上げ信号を伝播する．16 項においてセレクタ論理を用いることができる

（5） $i=4$($2^4$ の桁) のとき 22 項及び $2^3$ 桁からの桁上げ信号の加算を行い，$2^5$ 桁に桁上げ信号を伝播する．20 項においてセレクタ論理を用いることができる

（6） $i=5$($2^5$ の桁) のとき 17 項及び $2^4$ 桁からの桁上げ信号の加算を行い，$2^6$ 桁に桁上げ信号を伝播する．16 項においてセレクタ論理を用いることができる

（7） $i=6$($2^6$ の桁) のとき 12 項及び $2^5$ 桁からの桁上げ信号の加算を行い，$2^7$ 桁に桁上げ信号を伝播する．12 項においてセレクタ論理を用いることができる

（8） $i=7$($2^7$ の桁) のとき 8 項及び $2^6$ 桁からの桁上げ信号の加算を行い，$2^8$ 桁に桁上げ信号を伝播する．8 項においてセレクタ論理を用いることができる

（9） $i=8$($2^8$ の桁) のとき 4 項及び $2^7$ 桁からの桁上げ信号全ての XOR を取る．

図 4 のように積み重ねた部分積の加算には Wallace Tree を用いた．セレクタ論理に帰着した新しいバタフライ演算構成を図 5 に示す．

同様に Im $x_j$ においても式変形することでセレクタ論理に帰着した演算器構成を取ることができる．ここで，セレクタ論理に帰着することにより部分積は桁上げ伝播処理を削減し，高速化が期待できる．

[定理 1] セレクタ論理を用いた Radix-2 バタフライ演算構成は，式 (9) で示された計算を算出する回路を与える．

**証明.** $i$ 桁 ($0 \leqq i \leqq 2n-1$) に対し，セレクタ論理を用いた Radix-2 バタフライ演算構成が式 (9) を計算していることは自明である．従って $i=2n$ 桁の時を考える．Radix-2 バタフライ演算は $n$ ビット符号付き数に対して減算，乗算，加算（減算）を順に行う．$n$ ビット符号付き数 $X$ は $-2^{n-1} \leqq X \leqq 2^{n-1}-1$ の値の範囲を表すことが出来るため，$n$ ビット符号付き数の減算 $(X-Y)$ の取り得る範囲は $-2^n+1 \leqq (X-Y) \leqq 2^n-1$ となる．減算した数 $(X-Y)$ と $n$ ビット符号付き数 $Z$ の乗算結果の範囲は $-2^{n-1}(2^n-1) \leqq (X-Y)Z \leqq -2^{n-1}(-2^n+1)$ となるため $2^{2n}$ ビットの演算精度が必要である．Radix-2 バタフライ演算の演算精度は減算，乗算を縦列に演算した数の加算または減算するため減算，乗算を縦列に行うのに必要なビット数より 1 ビット多い $2^{2n+1}$ ビットとなる．そのため，符号拡張し演算を行った結果は必ず $(2n+1)$ ビットで表すことができる．この結果，$2^{2n}$ の係数は 0 か (-1) である．一方，式 (9) において $2^{2n}$ の係数は，最上位にある項と下位桁からの桁上がりを全て加算した値である．ここで，全ての値が 0，1，(-1) のいずれか値を取る加算の結果が必ず 0 か (-1) である時は全ての値の絶対値 $|0|$, $|1|$, $|(-1)|$ の XOR を取った値に (-1) をかけた値と等しくなる．それゆえ，$2^n$ の係数は単に最上位にある項と下位桁からの XOR を取ればよい． □

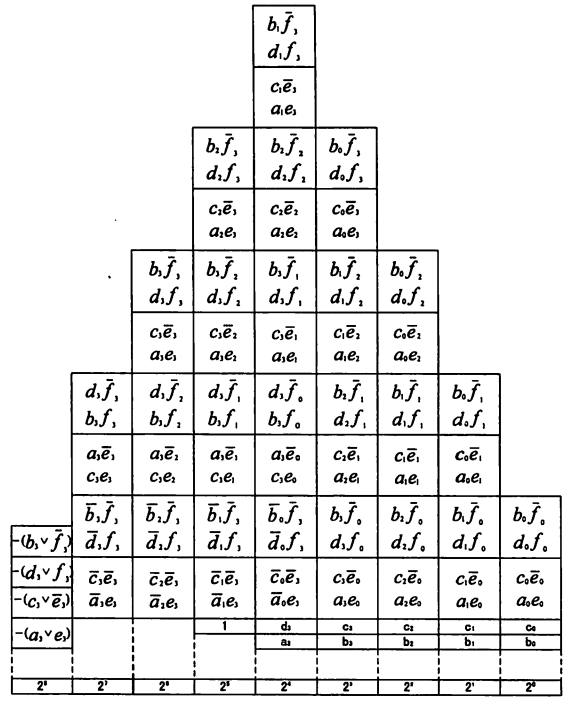


図 4 提案するバタフライ演算の部分積項

## 4. 評価実験

本節では，16 ビットの入力に対して，提案した Radix-2 バタフライ演算器を算術演算子を用いて設計した従来の算術演算子を用いた Radix-2 バタフライ演算器と比較して性能を評価した結果を示す．ハードウェア記述言語 VHDL を用いて設計し，Synopsys 社の DesignCompiler を用いて論理合成を行った．評価には論理合成結果から得られた回路面積と回路の性能を決定する遅延時間を用いて行った．また，論理合成時には速度を優先する制約を与えた．セルライブラリには STARC(注1)(CMOS90[nm]) の設計ルールを用いた．

設計した 2 つのバタフライ演算器の演算資源及びクリティカ

(注1)：STARC[90nm] ライブラリは東京大学大規模集積システム設計教育研究センターを通し，株式会社半導体理工研究センター (STARC) との協同で開発されたものである．

表 1　Radix-2 バタフライ演算器の性能比較

| | 提案した Radix-2 バタフライ演算器 | | 算術演算子を用いた Radix-2 バタフライ演算器 |
|---|---|---|---|
| 部分積生成回路 | セレクタ | 1152(16 × 16 × 2 × 2+16 × 4 × 2) | 17bit 加算器× 2 |
| 部分積加算回路 | Full Adder | 1086(527 × 2+32) | 17bit 減算器× 2 |
| | Half Adder | 30 | 17bit 乗算器× 4 |
| 最終加算回路 | 33bit 加算器× 2<br>17bit 加算器× 2(Re $X_i$,Im $X_i$) | | 34bit 減算器<br>34bit 加算器 |
| ゲート数 | 15312 (速度優先) | | 11861 (速度優先) |
| 段数 | not+セレクタ+Full Adder8 段+33 ビット加算器 | | 17bit 減算器+17bit 乗算器+34bit 減算器 |
| 遅延時間 | 0.61 ns (速度優先) | | 0.78 ns (速度優先) |
| 遅延時間比較 | 0.782 | | 1 |
| 面積比較 | 1.291 | | 1 |

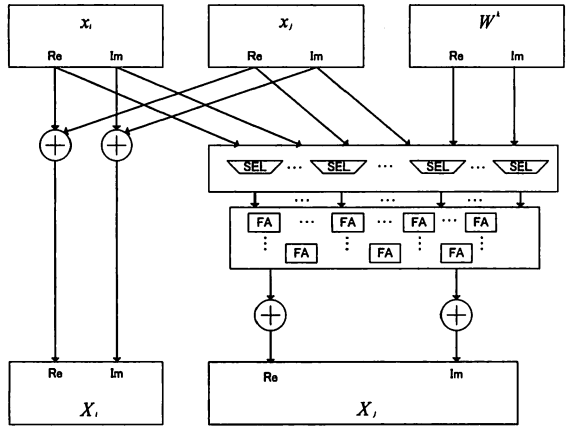


図 5　提案するバタフライ演算構成

ルパス，回路の構成面積の指標としてゲート数 (2NAND 回路換算)，回路の性能を決定する遅延時間を表 1 で示す．提案した Radix-2 バタフライ演算器は従来のバタフライ演算器と比較して 29.1% 面積が増加したが，速度優先設計において 21.8% 高速化することができた．FFT はさまざまな分野において頻繁に用いられる演算であり高性能化が求められているが，FFT において頻繁に用いられるバタフライ演算器のアルゴリズムは既に確立しているためアルゴリズムの工夫は難しい．しかし，演算式に着目し，式変形を行ったセレクタ論理を用いた Radix-2 バタフライ演算は高性能化の要求を満たす構成を取ることができる．

## 5. むすびに

本稿では，高性能な可変点数 FFT 演算回路の設計に対するアプローチとして，Radix-2 バタフライ演算回路においてビットレベル処理を行い，セレクタ論理に帰着させた新しいバタフライ演算構成を提案し，評価実験から従来のバタフライ演算回路と比較して 21.8% 高速化することができた．本稿で設計した Radix-2 バタフライ演算器は，部分積生成回路で 16 ビット入力に対して 1152 個のセレクタ回路を用いている．回路規模が大きくなるにつれ，部分積生成回路を AND 回路で用いて設計した回路と比べ，セレクタ回路で実現する回路の構成面積は全体の面積の中で大きな影響を及ぼす．この問題に対して，セレクタの回路をトランスミッションゲートではなく，pMOS または nMOS の片方だけを利用したパストランジスタを用いた構成をとる．パストランジスタ回路を用いることにより AND 回路と同等のトランジスタ回路でセレクタ論理を実現できるため，バタフライ演算器の回路規模削減が期待できる．これが今後の課題である．

謝　　辞

本研究をまとめるにあたり，御支援を賜った大日本印刷株式会社の方々に深く感謝致します．

## 文　　献

[1] Chang-Yeh Wang, "Hybrid word-length optimization methods of piplined FFT processors, " *IEEE Transaction on Computers*, vol. 56, No. 8, pp. 1105–1118, 2007.

[2] Fredrik Kristensen, Peter Nilsson, Anders Olsson, "A flexible FFT processor, " in *Proc. of 20th NORCHIP Conference*, pp. 121–126, 2002.

[3] 樋口龍雄, ディジタル信号処理の基礎, 昭晃堂.

[4] Jae H. Beak, Byung S. Son, Byung G. Jo, Myung H. Sunwoo, and Seung K. Oh, "A continuous flow mixed-radix FFT architecture with an in-place algorithm, " *IEEE Transaction on Circuits and Systems.II*, vol. 2, pp. 133–136, 2003.

[5] JW. Cooley and JW. Tukey, "An algorithm for the machine calculation of complex fourier series, " *Mathematics of Computation*, vol. 19, pp. 297–301, 1965.

[6] 小川由真, 陳土爽清, 魏書剛, "SD 数演算を用いた剰余数系一重み数系変換回路, " 信学技法, Vol. 104, No. 590, pp. 79–84, 2005.

[7] 白川昌哉, 富家修, 大竹俊也, 樋口裕二, "OFDM 用 FFT LSI の開発, " 映像情報メディア学会年次大会予稿集, No. 36, pp. 36, 1998.

[8] Shousheng He and Mats Tarkelson, "A complex multiplier using distributed arithmetic, " in *Proc. of Custom Intergrated Circuits Conference*, pp. 71–74, 1996.

[9] 高木直史, 算術演算の *VLSI* アルゴリズム, コロナ社.

[10] 外村元伸, 実設計に応用できる演算回路のスキルを身につけよう, Design wave magazine, pp. 39–48, May 2006.

[11] Wen-Chang Yeh and Chein-Wei Jen, "High-speed and low-power split-radix FFT, " *IEEE Transaction on Signal Processing*, vol. 51, No. 3, pp. 864-874, 2003.