セレクタ論理を適用したFFTプロセッサのFPGA実装評価

平井 勇也[1,a)]　川村 一志[2]　柳澤 政生[1]　戸川 望[1]

概要：高速フーリエ変換（FFT）は信号処理に代表される様々なアプリケーションで利用されており，高速な FFT プロセッサを設計することが求められる．本稿では，FFT 中の差積演算（減算後に乗算を行う演算）に着目した高速な FFT プロセッサの設計手法を提案する．差積演算にビットレベル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搬遅延を削減し，FFT プロセッサの処理速度向上を図る．本設計手法では，遅延時間の大きな差積演算のみをセレクタ論理に帰着させることで必要なハードウェア資源を抑えた設計を可能にする．差積演算の一部をセレクタ論理に帰着させた FFT プロセッサを FPGA に実装した結果，通常の FFT プロセッサに比べ，処理速度を最大 21%向上するとともに，LUT 数を最大 33%削減できることを確認した．

キーワード：セレクタ論理，高速フーリエ変換，FPGA

1. はじめに

高速フーリエ変換（FFT: fast Fourier transform）は信号処理に代表される様々なアプリケーションで利用されており，アクセラレーションのための高速な FFT プロセッサを設計することが求められる [1]．FFT に含まれる基本演算（加算，減算，乗算）のうち，乗算が処理時間のボトルネックとなることから，乗算処理の高速化が FFT 処理の高速化につながると期待される．

セレクタ論理は 1 ビットの入力信号 2 つを 1 ビットの制御信号を用いて選択する演算を表す．セレクタ論理の出力値域は必ず 1 以下であるため，セレクタ論理に帰着させることで算術演算の桁上げ処理時間を削減できる可能性がある．セレクタ論理に帰着させることで高速化できる演算のひとつに差積演算 $(a-b)\times c$ がある．差積演算にビットレベル式変形を施しセレクタ論理に帰着させることで，桁上げ伝播遅延の削減が可能である [2]．

これまでに，セレクタ論理を適用した重み付き加算器 [3] やバイリニア補間演算器 [4] が提案され，セレクタ論理による種々の演算器の高速化が確認されてきた．さらに，FFT 処理に用いるバタフライ演算を対象としたセレクタ論理による高速な演算器の設計手法が提案されており，ASIC [2] 及び FPGA [5] で高速化が実証されてきた．しかし，我々の知る限り，セレクタ論理を適用した FFT プロセッサの設計手法は提案されておらず，設計手法の提案及び実装評価が求められる．

以上の議論のもと，本稿ではセレクタ論理を適用した FFT プロセッサの設計手法を提案する．FFT [6] には差積演算が多用されるため，それらの差積演算をセレクタ論理に帰着させ，FFT プロセッサの処理速度向上を図る．本設計手法では，遅延時間の大きな差積演算のみをセレクタ論理に帰着させることで必要なハードウェア資源を抑えた設計を可能にする．本稿では，セレクタ論理適用型 FFT プロセッサを Xilinx 社の FPGA に実装し，評価する．その結果，我々が提案するセレクタ論理適用型 FFT プロセッサは通常の FFT プロセッサに比べ，処理速度を最大 21%向上するとともに，LUT 数を最大 33%削減できることを確認した．

本稿は以下のように構成される．2 章では，セレクタ論理を紹介し，差積演算をセレクタ論理に帰着させる手法を示す．3 章では，高速フーリエ変換（FFT）を紹介し，FFT にセレクタ論理を適用する方法を示す．4 章では，セレクタ論理を適用した FFT プロセッサの設計手法を提案する．5 章では，セレクタ論理適用型 FFT プロセッサを FPGA に実装し，性能評価する．6 章では，結論を述べる．

2. セレクタ論理に帰着した差積演算

本章では，セレクタ論理の紹介に続いて，差積演算をセレクタ論理に帰着させる手法を紹介する [2]．

[1] 早稲田大学大学院基幹理工研究科情報理工・情報通信専攻
169-8555 東京都新宿区大久保 3-4-1
[2] 早稲田大学理工学術院総合研究所
[a)] yuya.hirai@togawa.cs.waseda.ac.jp

表 1　セレクタ論理（$w = x \cdot z + y \cdot \overline{z}$）の真理値表.

Inputs			Output
x	y	z	w
X	Y	0	X
X	Y	1	Y

2.1　セレクタ論理

セレクタ論理は 1 ビットの入力信号 2 つを 1 ビットの制御信号を用いて選択する．1 ビットの入力信号 x, y 及び 1 ビットの制御信号 z に対し，セレクタ論理の出力信号 w は以下の式で表される．

$$w = x \cdot z + y \cdot \overline{z} \tag{1}$$

表 1 にセレクタ論理の真理値表を示す．表 1 からセレクタ論理の出力値域は 1 以下となり，2 進数で桁上げが発生しない．すなわち，算術演算式にビットレベル式変形を施しセレクタ論理に帰着させることで，桁上げ処理時間を削減し，演算器を高速化できる可能性がある．さらに，桁上げ回数が削減されることから加算器の個数が削減され，回路面積の削減も期待できる．

式 (1) において，$\cdot$ は論理積，$+$ は論理和を表すが，セレクタ論理演算には桁上げが存在しないため，セレクタ論理に帰着させる演算の算術和部分を論理和に置き換えても式の意味が変わらない．そのため，本稿ではビットレベル式のセレクタ論理帰着部分の $+$ は論理和と算術和を混合して表現する場合がある．また，論理積 $\cdot$ は省略する．

2.2　差積演算のビットレベル式変形

差積演算 $(a-b)\times c$ は FFT で多用される演算のひとつである．n ビット変数 a，b，c の i（$0 \le i \le n-1$）ビット目を a_i，b_i，c_i と表記すると，n ビット差積演算はビットレベルで以下のように変形できる．

$$\begin{aligned}
(a-b)c = \;& ac + b(-c) \\
=\;& \sum_{i=0}^{n-2}\sum_{j=0}^{n-2}(a_i c_j + b_i \overline{c_j})2^{i+j} \\
& + a_{n-1}2^{n-1}\left(-\sum_{i=0}^{n-2} c_i 2^i\right) \\
& + c_{n-1}2^{n-1}\left(-\sum_{i=0}^{n-2} a_i 2^i\right) \\
& + b_{n-1}2^{n-1}\left(-\sum_{i=0}^{n-2} \overline{c_i} 2^i\right) \\
& + \overline{c_{n-1}}2^{n-1}\left(-\sum_{i=0}^{n-2} b_i 2^i\right) \\
& - b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i \\
& + (a_{n-1}c_{n-1} + b_{n-1}\overline{c_{n-1}})2^{2n-2}
\end{aligned} \tag{2}$$

負の部分積項は符号ビットの処理が必要となり，正の部分積項に比べ余分な回路を必要とする．そこで，式 (2) 中に下線部で示された負の項を削減するため，2 の補数表現で成り立つ以下の式を用いる．

$$\begin{aligned}
-\sum_{i=0}^{n-2} a_i 2^i &= -\left(-0\cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i\right) \\
&= -1\cdot 2^{n-1} + \sum_{i=0}^{n-2} \overline{a_i} 2^i + 1
\end{aligned} \tag{3}$$

$$\begin{aligned}
-\sum_{i=0}^{n-2} b_i 2^i &= -\left(-0\cdot 2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i\right) \\
&= -1\cdot 2^{n-1} + \sum_{i=0}^{n-2} \overline{b_i} 2^i + 1
\end{aligned} \tag{4}$$

$$\begin{aligned}
-\sum_{i=0}^{n-2} c_i 2^i &= -\left(-0\cdot 2^{n-1} + \sum_{i=0}^{n-2} c_i 2^i\right) \\
&= -1\cdot 2^{n-1} + \sum_{i=0}^{n-2} \overline{c_i} 2^i + 1
\end{aligned} \tag{5}$$

$$\begin{aligned}
-\sum_{i=0}^{n-2} \overline{c_i} 2^i &= -\left(-0\cdot 2^{n-1} + \sum_{i=0}^{n-2} c_i 2^i\right) \\
&= -1\cdot 2^{n-1} + \sum_{i=0}^{n-2} \overline{c_i} 2^i + 1
\end{aligned} \tag{6}$$

式 (3)〜式 (6) を用いて式 (2) 中の負の項を削減し，以下の式を得る．

$$\begin{aligned}
(a-b)c = \;& \sum_{i=0}^{n-2}\sum_{j=0}^{n-2}(a_i c_j + b_i \overline{c_j})2^{i+j} + \sum_{i=0}^{n-2} b_i 2^i \\
& + \sum_{i=0}^{n-2}\{(b_{n-1}c_i + a_{n-1}\overline{c_i}) \\
& + (\overline{a_i}c_{n-1} + \overline{b_i}\,\overline{c_{n-1}})\}2^{i+n-1} \\
& + (a_{n-1}c_{n-1} + b_{n-1}\overline{c_{n-1}} \\
& \quad - a_{n-1} - c_{n-1} - b_{n-1} - \overline{c_{n-1}})2^{2n-2} \\
& + a_{n-1}2^{n-1} + (c_{n-1} + \overline{c_{n-1}})2^{n-1}
\end{aligned} \tag{7}$$

式 (7) の 2^{2n-2} の項に着目する．負の項を削減するとともに，セレクタ論理が適用可能となるよう，以下のように変形する．

$$\begin{aligned}
& (a_{n-1}c_{n-1} + b_{n-1}\overline{c_{n-1}} \\
& \quad - a_{n-1} - c_{n-1} - b_{n-1} - \overline{c_{n-1}})2^{2n-2} \\
&= \{(a_{n-1}c_{n-1} - c_{n-1} - a_{n-1} + 1) - 1 \\
& \quad + (b_{n-1}\overline{c_{n-1}} - b_{n-1} - \overline{c_{n-1}} + 1) - 1\}2^{2n-2} \\
&= \{(\overline{a_{n-1}}\,\overline{c_{n-1}}) + (\overline{b_{n-1}}c_{n-1}) - 2\}2^{2n-2}
\end{aligned} \tag{8}$$

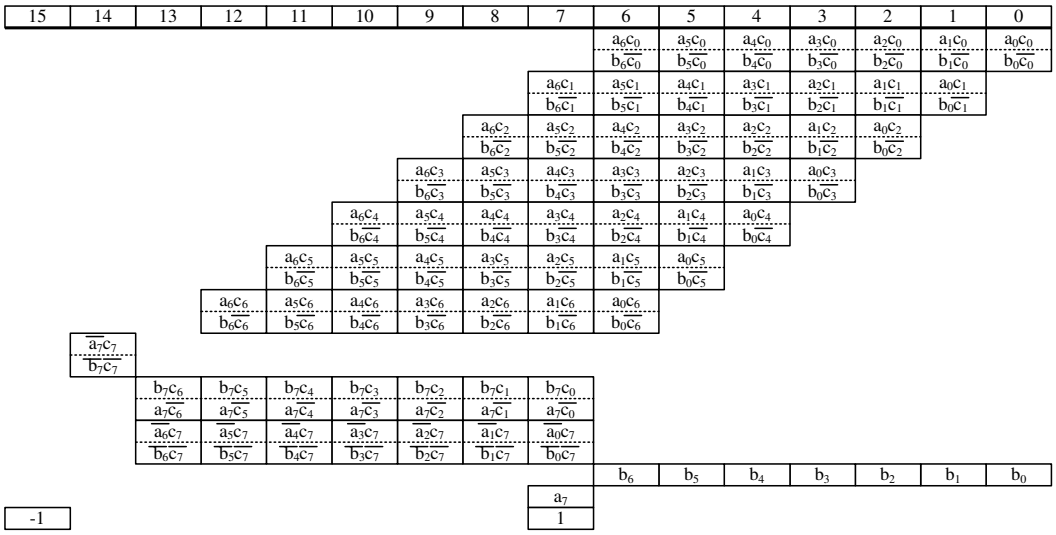

図 1 8 ビット差積演算の部分積.

式 (8) を式 (7) に代入し，次式を得る．

$$
\begin{aligned}
(a-b)c = & \sum_{i=0}^{n-2}\sum_{j=0}^{n-2}\underline{(a_i c_j + b_i \overline{c_j})}2^{i+j} \\
& + \sum_{i=0}^{n-2} b_i 2^i \\
& + \sum_{i=0}^{n-2}\{\underline{(b_{n-1}c_i + a_{n-1}\overline{c_i})} \\
& \qquad + \underline{(\overline{a_i}c_{n-1} + \overline{b_i}\,\overline{c_{n-1}})}\}2^{i+n-1} \\
& + \{\underline{(\overline{a_{n-1}}\,\overline{c_{n-1}}) + (\overline{b_{n-1}}c_{n-1})}\}2^{2n-2} \\
& + a_{n-1}2^{n-1} + 2^{n-1} - 2^{2n-1} \qquad (9)
\end{aligned}
$$

式 (9) から，部分積項の数は合計 $2n^2+n+2$ となる．式 (9) の下線部はセレクタ論理を適用可能な項であり，セレクタ論理の適用により部分積項の数を n^2+n+2 に削減可能である．このようにしてセレクタ論理に帰着させることで n^2 個の部分積項を削減でき，差積演算を効率化できる．上記のビットレベル式変形により 8 ビット差積演算の場合に生成される部分積を図 1 に示す．図 1 の 1 行目は桁数を表し，それ以降の行は部分積を表す．8 ビット差積演算の場合，セレクタ論理の適用により部分積項の数を 138 から 72 に削減可能である．

3. セレクタ論理を適用した高速フーリエ変換

本章では，高速フーリエ変換 [6] を紹介し，高速フーリエ変換にセレクタ論理を適用する方法を示す．

3.1 高速フーリエ変換

高速フーリエ変換（FFT）は離散フーリエ変換（DFT: discrete Fourier transform）の計算方法に工夫を加え高速に計算できるようにしたものである．

標本数が N 点の DFT（N 点 DFT）は次式で表される．

$$X(k) = \sum_{s=0}^{N-1} x(s) W_N^{ks} \qquad (10)$$

ここで，$k = 0, 1, \cdots, N-1$ であり，$W_N = e^{-j\frac{2\pi}{N}}$ である．$X(0)$ から $X(N-1)$ まで計算すると，複素乗算が N^2 回必要となる．式 (10) は，前半部分と後半部分に分けることで以下のように変形できる．

$$
\begin{aligned}
X(k) &= \sum_{s=0}^{\frac{N}{2}-1} x(s) W_N^{ks} + \sum_{s=\frac{N}{2}}^{N-1} x(s) W_N^{ks} \\
&= \sum_{s=0}^{\frac{N}{2}-1} \left\{ x(s) + W_N^{k\frac{N}{2}} x\left(s+\frac{N}{2}\right)\right\} W_N^{ks} \qquad (11)
\end{aligned}
$$

ここで，N が偶数の場合に $X(k)$ の添え字 k を偶数（$2m$）と奇数（$2m+1$）に分けると以下のように変形できる $\left(0 \le m \le \frac{N}{2}-1\right)$.

$$X(2m) = \sum_{s=0}^{\frac{N}{2}-1} \left\{ x(s) + x\left(s+\frac{N}{2}\right)\right\} W_{N/2}^{ms} \qquad (12)$$

$$X(2m+1) = \sum_{s=0}^{\frac{N}{2}-1} \left\{ x(s) - x\left(s+\frac{N}{2}\right)\right\} W_N^{s} W_{N/2}^{ms} \qquad (13)$$

上式において，$x_{even}(s)$ と $x_{odd}(s)$ を次式で定める．

$$x_{even}(s) = x(s) + x\left(s+\frac{N}{2}\right) \qquad (14)$$

$$x_{odd}(s) = \left\{ x(s) - x\left(s+\frac{N}{2}\right)\right\} W_N^{s} \qquad (15)$$

このとき，式 (12) と式 (13) はそれぞれ，$x_{even}(s)$ と $x_{odd}(s)$ を入力とする $N/2$ 点 DFT を表す．つまり，N 点 DFT を 2 つの $N/2$ 点 DFT に分割することができる．N が 2 のべき乗で表されるとき，同様の分割を繰り返すことで最終的

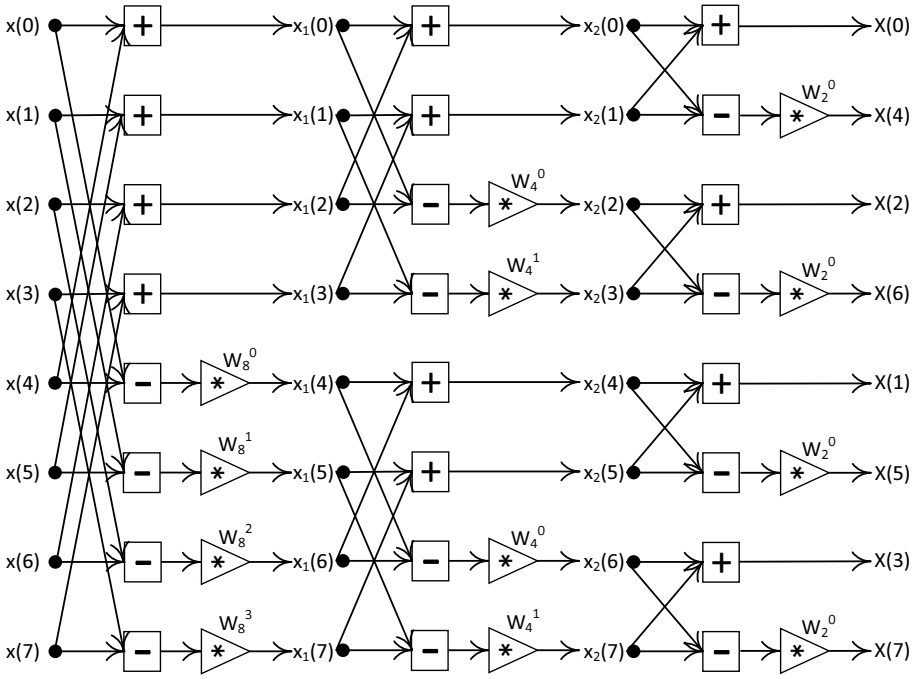

図 2 8 点 FFT のシグナルフローグラフ.

に 2 点 DFT にまで分解することができる．このようにして FFT では W_N の周期性を利用して式 (10) を再帰的に分割し，高速に演算する．結果として，N 点 FFT では複素乗算が $\frac{N}{2}\log_2 N$ 回必要となる．

3.2 高速フーリエ変換へのセレクタ論理の適用

本稿では，セレクタ論理を用いて 8 点 FFT プロセッサを設計し，実装評価する．本節にて 8 点 FFT に含まれる差積演算をセレクタ論理に帰着させる方法を示す．

8 点 FFT をすべて書き下すと，以下のようになる．

$$
\begin{cases}
X(0) = \{\big(x(0)+x(4)\big)+\big(x(2)+x(6)\big)\} \\
\qquad +\{\big(x(1)+x(5)\big)+\big(x(3)+x(7)\big)\} \\
X(1) = \{\big(x(0)-x(4)\big)W_8^0+\big(x(2)-x(6)\big)W_8^2\} \\
\qquad +\{\big(x(1)-x(5)\big)W_8^1+\big(x(3)-x(7)\big)W_8^3\} \\
X(2) = \{\big(x(0)+x(4)\big)-\big(x(2)+x(6)\big)\}W_4^0 \\
\qquad +\{\big(x(1)+x(5)\big)-\big(x(3)+x(7)\big)W_4^1\} \\
X(3) = \{\big(x(0)-x(4)\big)W_8^0-\big(x(2)-x(6)\big)W_8^2\}W_4^0 \\
\qquad +\{\big(x(1)-x(5)\big)W_8^1+\big(x(3)-x(7)\big)W_8^3\}W_4^1 \\
X(4) = \big\{\{\big(x(0)+x(4)\big)+\big(x(2)+x(6)\big)\} \\
\qquad -\{\big(x(1)+x(5)\big)+\big(x(3)+x(7)\big)\}\big\}W_2^0 \\
X(5) = \big\{\{\big(x(0)-x(4)\big)W_8^0+\big(x(2)-x(6)\big)W_8^2\} \\
\qquad -\{\big(x(1)-x(5)\big)W_8^1+\big(x(3)-x(7)\big)W_8^3\}\big\}W_2^0 \\
X(6) = \big\{\{\big(x(0)+x(4)\big)-\big(x(2)+x(6)\big)\}W_4^0 \\
\qquad +\{\big(x(1)+x(5)\big)-\big(x(3)+x(7)\big)W_4^1\}\big\}W_2^0 \\
X(7) = \big\{\{\big(x(0)-x(4)\big)W_8^0-\big(x(2)-x(6)\big)W_8^2\}W_4^0 \\
\qquad -\{\big(x(1)-x(5)\big)W_8^1+\big(x(3)-x(7)\big)W_8^3\}W_4^1\big\}W_2^0
\end{cases}
\tag{16}
$$

式 (16) には $\{x(1)-x(5)\}W_8^1$ のような差積演算が複数含まれる．ここでは，$\{x(1)-x(5)\}W_8^1$ をセレクタ論理に帰着させる方法を示す．$\{x(1)-x(5)\}W_8^1 = A+Bj$，$x(1)=a+bj$，$x(5)=c+dj$，$W_8^1=e+fj$ と表すと，出力の実部 A について以下のように変形できる．

$$
\begin{aligned}
A = \;& (a-c)e+(d-b)f \\
=\;& \sum_{i=0}^{n-2}\sum_{j=0}^{n-2}\{\underline{(a_ie_j+c_i\overline{e_j})}+\underline{(d_if_j+b_i\overline{f_j})}\}2^{i+j} \\
& +\sum_{i=0}^{n-2}(c_i+b_i)2^i \\
& +\sum_{i=0}^{n-2}\{\underline{(c_{n-1}e_i+a_{n-1}\overline{e_i})}+\underline{(b_{n-1}f_i+d_{n-1}\overline{f_i})} \\
& \quad +\underline{(\overline{a_i}e_{n-1}+\overline{c_ie_{n-1}})}+\underline{(\overline{d_i}f_{n-1}+\overline{b_if_{n-1}})}\}2^{i+n-1} \\
& +\{\underline{(\overline{a_{n-1}c_{n-1}}+\overline{b_{n-1}}c_{n-1})} \\
& \quad +\underline{(\overline{b_{n-1}f_{n-1}}+\overline{d_{n-1}}f_{n-1})}\}2^{2n-2} \\
& -2^{2n}+(a_{n-1}+d_{n-1})2^{n-1}+2^n
\end{aligned}
\tag{17}
$$

式 (17) の表現により，下線部にセレクタ論理を適用可能である．出力の虚部 $B=(a-c)f+(b-d)e$ も同様の方法でセレクタ論理に帰着させることができる．

以上により，$\{x(1)-x(5)\}W_8^1$ をセレクタ論理に帰着可能である．他の差積演算についても同様の方法でセレクタ論理に帰着させることができる．

4. セレクタ論理を適用した FFT プロセッサの設計

本章では，セレクタ論理を適用した FFT プロセッサの設計手法を提案する．本稿で想定する FFT プロセッサの

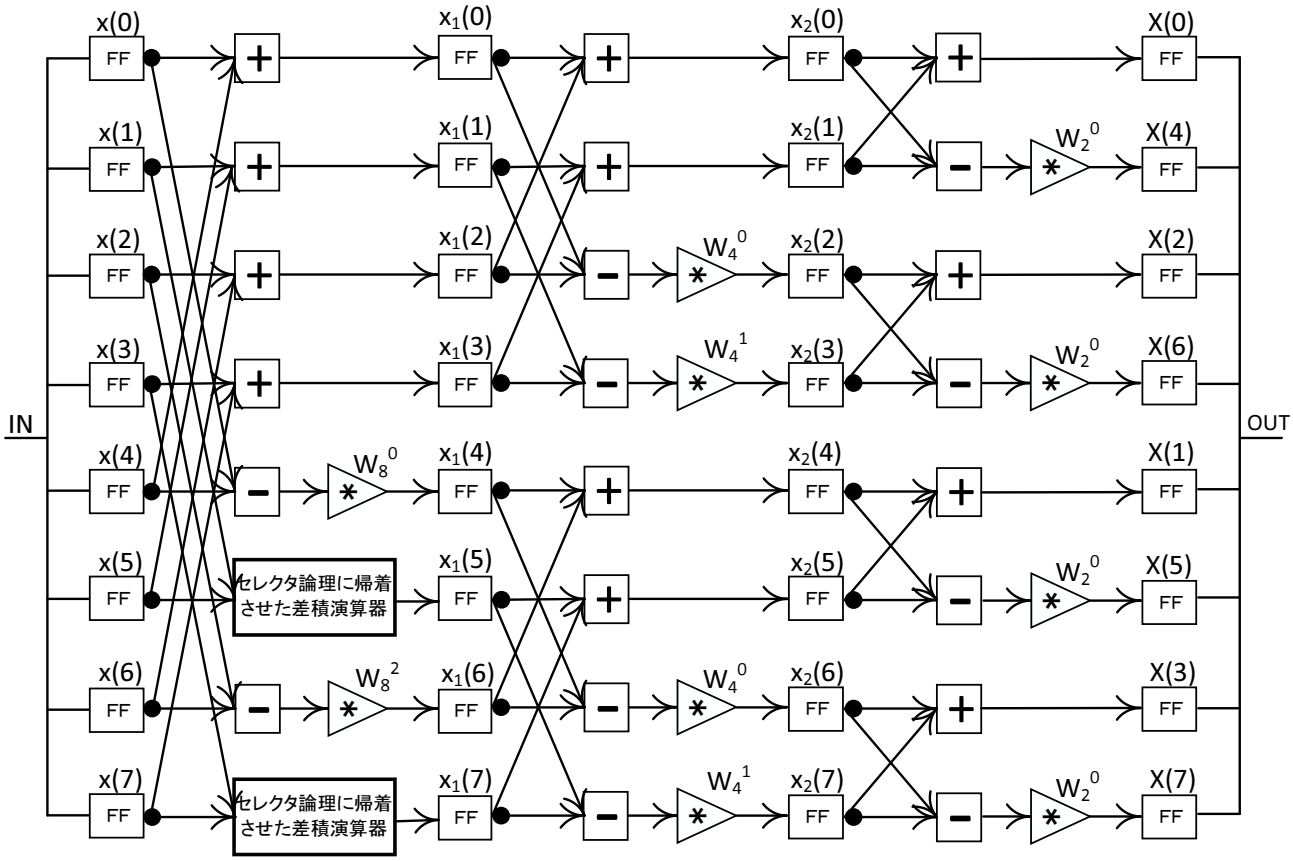

図 3 セレクタ論理を適用した 8 点 FFT プロセッサ.

アーキテクチャを 4.1 節に示し，4.2 節にセレクタ論理適用型 FFT プロセッサの設計手法を示す.

4.1 FFT プロセッサアーキテクチャ

式 (16) から 8 点 FFT のシグナルフローグラフは図 2 となる．図 2 にもとづき，基本となる 8 点 FFT プロセッサを設計する．本プロセッサのアーキテクチャと動作を以下に示す.

- 入力 $x(s)$ $(0 \leq s \leq 7)$ を格納する FF を用意する．1 サイクルに 1 データを本 FF に読み込み，読み込みが完了した次のサイクルから FFT 処理を開始する.
- 中間データ $x_1(t)$，$x_2(u)$ $(0 \leq t, u \leq 7)$ を格納する FF を用意する．FFT 処理を開始した後，最初のサイクルで 8 個の計算結果を $x_1(t)$ を表す FF に格納し，次のサイクルで 8 個の計算結果を $x_2(u)$ を表す FF に格納する.
- 出力 $X(k)$ $(0 \leq k \leq 7)$ を格納する FF を用意する．本 FF には FFT 処理を開始してから 3 サイクル後に計算結果が格納される．その次のサイクルより，1 サイクルに 1 個のデータを出力する.

上記 FFT プロセッサを論理合成した結果，2 箇所の差積演算（$\{x(1)-x(5)\}W_8^1$，$\{x(3)-x(7)\}W_8^3$）の遅延時間が大きく，ボトルネックとなっていることを確認した．一方，$W_8^0 = W_4^0 = W_2^0 = 1$ 及び $W_8^2 = W_4^1 = -j$ であることから，残り 10 箇所の差積演算は計算内容が単純であり，遅延時間が小さくなっていると考えられる.

4.2 セレクタ論理適用型 FFT プロセッサの設計

4.1 節での議論にもとづき，8 点 FFT に含まれる 12 箇所の差積演算のうち，遅延時間のボトルネックとなる 2 箇所の差積演算（$\{x(1)-x(5)\}W_8^1$，$\{x(3)-x(7)\}W_8^3$）のみを 3.2 節に示した方法でセレクタ論理に帰着させる．本稿にて提案するセレクタ論理適用型 8 点 FFT プロセッサを図 3 に示す.

図 3 中のセレクタ論理に帰着させた差積演算器では，式 (17) により生成した部分積項をセレクタ論理により削減する．このとき，最終的な計算結果は部分積項を加算し取りまとめることで得ることができる．以降，部分積項のとりまとめについて考察する.

部分積項のとりまとめにおいて，Wallace tree [7] や Dadda tree [8] を利用することが考えられる．これらは各桁の部分積の項数を 2 項に圧縮する手法であり，圧縮された部分積項を 2 項加算することで最終的な計算結果を得る．一方，部分積項を圧縮せず算術演算子を用いて部分積項の加算を記述し，論理合成ツールの最適化を適用することも考えられる．先行研究 [5] では，FFT に用いられるバタフライ演算器を対象に部分積項のとりまとめ方法を比較・検討しており，Wallace tree や Dadda tree を用いるよりも

表 2　8 点 FFT プロセッサの FPGA 実装結果.

Word size	設計手法	最小クロック周期 [ns]	FF	LUT	I/O
8 bit	通常	4.3	534	1140	35
	セレクタ論理適用型	4.0	534	886	35
16 bit	通常	6.3	1061	3498	67
	セレクタ論理適用型	5.0	1061	2647	67
24 bit	通常	7.6	1590	6706	99
	セレクタ論理適用型	6.6	1590	4494	99

算術演算子を用いた構成の方が総遅延時間が小さくなることを示している．以上の議論のもと，本稿では部分積項のとりまとめに算術演算子を用いて論理合成ツールの最適化を適用する方法を採用する．

5. FPGA 実装評価

本章では，前章で提案した設計手法のもと，セレクタ論理を適用した 8 点 FFT プロセッサを Verilog HDL で記述し，FPGA に実装する．

5.1 実験環境

ワードサイズを 8 ビット，16 ビット，24 ビットと変化させ，8 点 FFT プロセッサを FPGA に実装する．対象 FPGA ボードは Xilinx 社 Virtex-7 の xc7vx690tffg1926-2 とし，論理合成・レイアウト設計ツールとして Xilinx 社 Vivado2015.2 を用いる．本実験では，（セレクタ論理非適用型の）通常 FFT プロセッサとセレクタ論理適用型 FFT プロセッサの 2 通りを実装し，性能を評価する．

5.2 実装結果

実装結果を表 2 に示す．表 2 の 3 列目は最小クロック周期を表し，ここに示された値が小さいほど FFT 処理を高速に実行可能である．表 2 の 4〜6 列目は必要なハードウェア資源数（フリップフロップ数，LUT 数，I/O 数）を表す．本実験で実装した 2 通りの FFT プロセッサは基本となるアーキテクチャが同じであるため，同じワードサイズであれば FF 数と I/O 数に違いが見られない．一方，差積演算の一部をセレクタ論理に帰着させることにより最小クロック周期及び LUT 数が変化している．

表 2 の結果から，セレクタ論理適用型 8 点 FFT プロセッサは（セレクタ論理非適用型）8 点 FFT プロセッサと比較して処理速度を向上させるとともに，LUT 数を削減している．ワードサイズが 8bit の場合，処理速度を 7%向上させ，LUT 数を 22%削減した．ワードサイズが 16bit の場合，処理速度を 21%向上させ，LUT 数を 24%削減した．ワードサイズが 24bit の場合，処理速度を 13%向上させ，LUT 数を 33%削減した．従って，我々が提案するセレクタ論理適用型 FFT プロセッサは通常の FFT プロセッサに比べ，処理速度を平均 14%，最大 21%向上するとともに，LUT 数を平均 27%，最大 33%削減できる．FFT プロセッサのポイント数を 16 点以上とした場合でも，遅延時間のボトルネックとなる箇所は同様であり，処理速度の向上率について同傾向の結果を得ることができると予測される．

6. おわりに

本稿ではセレクタ論理を適用した FFT プロセッサの設計手法を提案した．本設計手法では，遅延時間の大きな差積演算のみをセレクタ論理に帰着させることで処理性能を向上させるとともに，必要なハードウェア資源を抑えた設計を可能にする．セレクタ論理適用型 FFT プロセッサを Xilinx 社の FPGA に実装した結果，我々が提案するセレクタ論理適用型 FFT プロセッサは通常の FFT プロセッサに比べ，処理速度を最大 21%向上するとともに，LUT 数を最大 33%削減できることを確認した．

謝辞　本研究は JST CREST JPMJCR1503 の支援を受けたものである．

参考文献

[1] W. Wang and X. Huang, “FPGA implementation of a large-number multiplier for fully homomorphic encryption,” in *Proc. of International Symposium on Circuits and Systems (ISCAS)*, pp. 2589–2592, 2013.

[2] Y. Tsukamoto, M. Yanagisawa, T. Ohtsuki, and N. Togawa, “A fast selector-based subtract-multiplication unit and its application to butterfly unit,” *IPSJ Transactions on System LSI Design Methodology*, no. 2, vol. 4, pp. 60–69, 2011.

[3] H. Yoshihara, M. Yanagisawa, and N. Togawa, “A fast weighted adder by reducing partial product for reconstruction in super-resolution,” *IPSJ Transactions on System LSI Design Methodology*, vol. 5, pp. 96–105, 2012.

[4] M. Shio, M. Yanagisawa, and N. Togawa, “Linear and bi-linear interpolation circuits using selector logics and their evaluations,” in *Proc. of International Symposium on Circuits and Systems (ISCAS)*, pp. 1436–1439, 2014.

[5] 伊東光希, 柳澤政生, 戸川望, “セレクタ論理に帰着させたバタフライ演算器の FPGA 実装評価,” 信学技報, vol. 116, no. 330, pp. 67–72, 2016.

[6] A. Saidi, “Generalized FFT algorithm,” in *Proc. of ICC*, vol. 1, pp. 227–231, 1993.

[7] C. S. Wallace, “A suggestion for a fast multiplier,” *IEEE Trans. on Electronic Computers*, vol. 13, pp. 14–17, 1964.

[8] L. Dadda, “Some schemes for parallel multiplier,” *Alta Frequenza*, vol. 34, pp. 349–356, 1965.