社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

FIR フィルタの算術分解を用いた LUT カスケードによる実現

笹尾　勤†　井口　幸洋††　鈴木　隆広††

† 九州工業大学 情報工学部 電子情報工学科　〒 820–8502 福岡県飯塚市大字川津 680–4
†† 明治大学 理工学部 情報科学科　〒 214–8571 川崎市多摩区東三田 1–1–1

あらまし　FIR フィルタの分散演算の組合せ論理回路の数学モデルとして, n 入力 q 出力 WS 関数を定義する. 次に, WS 関数を k 入力 q 出力のセルを用いた LUT カスケードで実現する方法を示す. 更に, 次の 3 点を明らかにする：(1) 単一の ROM で実現する場合に比べ, LUT カスケードを用いた実現は, 必要メモリ量を大幅に削減できる. (2) 算術分解を用いて WS 関数を実現する新しい方法を提案する. (3)FIR フィルタを組込みメモリを用いて FPGA 上に実現した結果を示す.

キーワード　ディジタルフィルタ, 分散演算, LUT カスケード, 関数分解, 二分決定グラフ, FPGA.

On LUT Cascade Realizations of FIR Filters Using Arithmetic Decomposition

Tsutomu SASAO†, Yukihiro IGUCHI††, and Takahiro SUZUKI††

† Department of Computer Science and Electronics, Kyushu Institute of Technology
†† Department of Computer Science, Meiji University

Abstract　This paper first defines the n-input q-output WS function, as a mathematical model of the combinational part of the distributed arithmetic of a finite impulse response (FIR) filter. Then, it shows a method to realize the WS function by an LUT cascade with k-input q-output cells. Furthermore, it 1) shows that LUT cascade realizations require much smaller memory than the single ROM realizations; 2) presents new design method for a WS function by arithmetic decomposition, and 3) shows design results of FIR filters using FPGAs with embedded memories.

Key words　Digital filter, Distributed arithmetic, LUT cascade, Functional decomposition, Binary decision diagram, FPGA.

1. はじめに

ディジタルフィルタは, 信号処理における基本要素である [9]. 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) フィルタとに分類できる. FIR フィルタは, 非再帰形で構成でき, 常に安定な動作を行うという特徴がある. また, 線形位相特性をもたせられるので, 波形伝送にも利用できる.

FPGA 上に FIR フィルタを実現する際, 積和演算を分散演算 (Distributed Arithmetic: DA) という手法で, テーブル参照に置き換えて実行する方法が知られている [7], [15], [16]. テーブル参照には, FPGA 内の組込みメモリを利用できる.

本論文では, FIR フィルタの組合せ回路部を図 3.1 に示す LUT カスケードと呼ぶ, メモリを直列に接続した回路で実現する方法を提案する. この方法では, 分散演算を率直に 1 つのメモリで実現するよりも, はるかに少ないメモリ量で実現できる. 本手法は, FPGA の組込みメモリや, 専用 LUT カスケードチップ [8] を用いて FIR フィルタを実現する際に応用可能である.

本論文の構成を以下に示す. 第 2 章では, FIR フィルタについて, 第 3 章では, LUT カスケードと関数分解とについて概説する. 第 4 章では, WS 関数を定義し, LUT カスケードを用いた実現法を示す. 第 5 章では, WS 関数の算術分解法を提案する. 第 6 章では, 実験結果を示す. 最後に第 7 章でまとめを行う.

本論文は, 文献 [13] を日本語に翻訳したものである.

2. FIR フィルタ

［定義 2.1］ **FIR フィルタ**とは,

$$\mathcal{Y}(n) = \sum_{i=0}^{N-1} h_i \cdot \mathcal{X}(n-i) \tag{2.1}$$

を計算する回路である. ここで, $\mathcal{X}(i)$ は, 時刻 i における入力

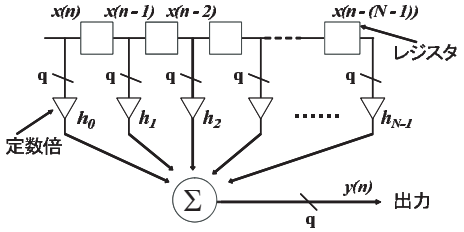

図 2.1　FIR フィルタ (並列実現).

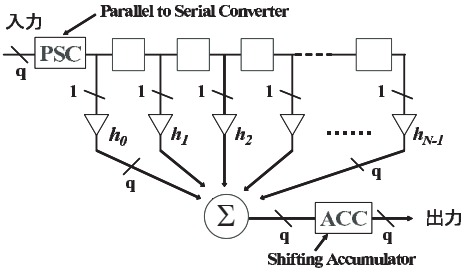

図 2.2　FIR フィルタ (直列実現).

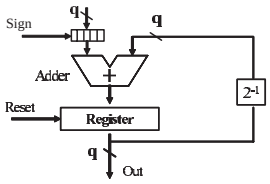

図 2.3　ACC の実現例.

$\mathcal{X}$ の値, $\mathcal{Y}(i)$ は, 時刻 i における出力 $\mathcal{Y}$ の値を表す(注1). また, h_i は q ビットの固定小数点で表現された数で**フィルタ係数**を表し, N は, フィルタの**タップ数**を示す(注2).

図 2.1 は, 式 (2.1) を直接実現する回路である. $(N-1)$ 段の q ビット・シフトレジスタの他に, q ビットの定数乗算器 N 個と, q ビットの数を N 個加算する回路から構成されており, ハードウエア量は多い. 図 2.1 の回路のハードウエア量を削減するために, q ビットの数をビット直列に計算するように変更した回路が, 図 2.2 である. ここで, PSC は並列直列変換器 (Parallel to Serial Converter) を表している. また, この場合 $h_0, h_1, \ldots, h_{N-1}$ への入力は, 0 または 1 の数となるので, 定数乗算器は, AND ゲートで置換できる. 図 2.2 の ACC はシフト累算器 (Shifting Accumulator) を表す.　これは, シフト演算をしながら加算を繰り返す回路であり, 例えば, 図 2.3 の回路で実現可能である.

この方式では, ハードウエア量は約 q 分の 1 に削減可能であるが, 処理時間は約 q 倍となる. 図 2.2 の組合せ回路の部分は N 入力 q 出力となるが, この部分は後で定義する WS 関数を実現する. 線形位相特性を満たす FIR フィルタでは, その係数が関係 $h_i = h_{N-i-1}$ を満たす. このようなフィルタを**対称フィルタ**という. このとき, 図 2.2 の回路は図 2.4 の回路に変更でき, ハードウエア量を削減できる. この場合, q ビットの数を $(N+1)/2$ 個加算する回路を用いればよい. また, 図 2.4

(注1)：本論文では, $\mathcal{X}$ や $\mathcal{Y}$ 等の変数は, フィルタで用いる信号値を表し, x_i は論理変数, $\vec{X}_1$, $\vec{X}_2$ は論理変数のベクトルを表す.

(注2)：一般に入力変数 $\mathcal{X}$ のビット数, フィルタ係数 h_i のビット数, 出力 $\mathcal{Y}$ のビット数は異なっても良い. しかし, 本論文では簡単のため全て q ビットで表現されているものとする.

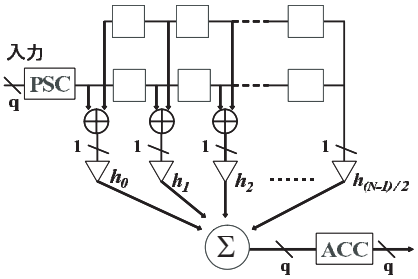

図 2.4　FIR フィルタ (直列対称実現).

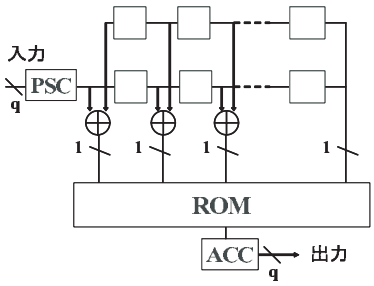

図 2.5　FIR フィルタ (直列対称 ROM 実現).

表 2.1　WS 関数の例.

Address	Data
000	0
001	h_0
010	h_1
011	$h_0 + h_1$
100	h_2
101	$h_0 + h_2$
110	$h_1 + h_2$
111	$h_0 + h_1 + h_2$

の $\oplus$ の記号は直列加算器を示す(注3). 図 2.5 は, 組合せ回路の部分を ROM で実現したものである. 例えば入力数が 3 の場合, ROM には表 2.1 に示す値を予め計算して貯えておく. これは, 後で定義する WS 関数の例となっている. これは, **分散演算** (Distributed Arithmetic) とも呼ばれ, 多数の乗算器と多入力の加算器を 1 個のメモリで置き換えることができるので, 畳み込み演算を実現する際に多用されている [1], [3], [6], [7], [15], [16]. 但し, この方法を利用できるのは, 一方の入力が定数のときのみである.　FIR フィルタでは, フィルタの係数が定数なので分散演算を使用できる. 一般に, ROM の入力数を n, 出力数を q とするとき, ROM に必要なメモリ量は, $q2^n$ ビットとなる.

3.　LUT カスケードと関数分解

本章では, LUT カスケードと関数分解との関係を述べる.

[定義 3.1]　**LUT カスケード**は, 与えられた論理関数を図 3.1 の回路構造で計算する. 各ブロックを**セル**といい, 各セルは任意の論理関数を実現する. また, 隣接するセル間を接続する信号線を**レイル**と呼ぶ.

[定義 3.2]　多出力論理関数 $\vec{F}(\vec{X})$ を $\vec{F} = g(h(\vec{X}_1), \vec{X}_2)$ の形で表現することを**関数分解**という. この時, $\vec{X}_1$ の変数を**束縛変数**, $\vec{X}_2$ の変数を**自由変数**という. ここで h は多値の関数であ

(注3)：直列加算の直前には Reset 信号を加えて, レジスタやフリップ・フロップの値を 0 にする必要がある.

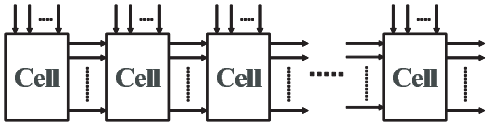

図 3.1　LUT カスケード.

表 3.1　分解表の例.

$\vec{X}_1 = (x_0, x_1, x_2)$

$\vec{X}_2 = (x_3, x_4)$	000	001	010	011	100	101	110	111
00	001	010	011	100	001	100	100	100
01	111	110	000	011	111	010	010	011
10	010	110	010	001	010	010	010	001
11	010	011	101	010	010	011	011	010

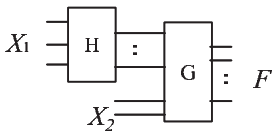

図 3.2　関 数 分 解.

る(注4).

[定義 3.3]　$\vec{X}_1$ の変数を列に, $\vec{X}_2$ の変数を行に対応させ, $\vec{F}(\vec{X}_1, \vec{X}_2)$ の値が, 対応する要素となる表を $\vec{F}(\vec{X}_1, \vec{X}_2)$ の**関数分解表**という. q 出力関数の場合, 各要素は, q ビットのベクトルとなる. 異なる列パターンの数を**列複雑度**という.

[例 3.1]　表 3.1 に 5 入力 3 出力論理関数の分解表の例を示す. ここで, $\vec{X}_1 = (x_0, x_1, x_2)$, $\vec{X}_2 = (x_3, x_4)$ である. また, 列複雑度は 5 である.　(例終り)

[定理 3.1]　関数分解 $\vec{F}(\vec{X}) = g(h(\vec{X}_1), \vec{X}_2)$ において, 分解表の列複雑度が μ のとき, 関数 $\vec{F}(\vec{X})$ は, 図 3.2 の回路構造で実現可能である. また, 二つのブロック H と G の間の信号線は, $\lceil \log_2 \mu \rceil$ 本あれば十分である.

[定理 3.2]　関数 $\vec{F}$ の変数を $\vec{X} = (x_0, x_1, \ldots, x_{n-1})$ とする. $\vec{X}_1 = (x_0, x_1, \ldots, x_i), \vec{X}_2 = (x_{i+1}, \ldots, x_{n-1})$ とするとき, 全ての i $(q < i < n-1)$ に関して, $\vec{F}(\vec{X}_1, \vec{X}_2)$ の分解表の列複雑度が 2^q 以下ならば, $\vec{F}$ は, k 入力 q 出力のセルを用いた LUT カスケードで実現可能である. ここで, $k > q$ である.

(証　　明)

$\vec{X}_1 = (x_0, x_1, \ldots, x_{k-1}), \vec{X}_2 = (x_k, \ldots, x_{n-1})$ として, $\vec{F}(\vec{X})$ を関数分解すると, 図 3.2 の回路が得られる. 二つのブロック間の信号線数を q とすると, G は $n-k+q$ 入力回路となる. G に対して, 再度関数分解を適用する. この際, 束縛変数がすべての中間変数 (即ち, H の出力信号線) を含むようにする. このとき, 分解表の列複雑度はやはり 2^q 以下となる.

以上のように関数分解を繰り返し適用することにより, 図 3.1 のカスケード回路を得る.　(証明終)

4.　WS 関数とその LUT カスケード実現

[定義 4.1]　n 入力 q 出力 **WS 関数** [11] $\vec{F}(x_0, x_1, \ldots, x_{n-1})$ とは,

$$\sum_{i=0}^{n-1} h_i \cdot x_i \qquad (4.1)$$

の値を計算し, その結果を q ビットの 2 進数で表現する n 入力 q 出力論理関数である(注5). ここで, h_i は q ビットの 2 進数で表現されている係数である. 数の表現には, 固定小数点表現を用い, 負数は 2 の補数表現を用い, 量子化ビット数には, 符号のための 1 ビットを含むものとする. h_i の 2 進表現を $(\vec{h}_i)_2$ とすると, $\vec{F}$ は,

$$\begin{aligned}
\vec{F}(1, 0, \ldots, 0) &= (\vec{h}_0)_2 \\
\vec{F}(0, 1, \ldots, 0) &= (\vec{h}_1)_2 \\
&\vdots \\
\vec{F}(0, 0, \ldots, 1) &= (\vec{h}_{n-1})_2 \quad \text{を満たすものとする.}
\end{aligned}$$

WS 関数は, 係数を q ビットに量子化し, その後加算した関数である.

次の補題は, 任意の q 出力 WS 関数の分解表の列複雑度は高々2^q であることを示す.

[補題 4.1]　n 入力 q 出力 WS 関数を $\vec{F}(x_0, x_1, \ldots, x_{n-1})$ とする. いま, $(\vec{X}_1, \vec{X}_2)$ を $\vec{X} = (x_0, x_1, \ldots, x_{n-1})$ の分割, ただし, $\vec{X}_1 = (x_0, x_1, \ldots, x_{i-1}), \vec{X}_2 = (x_i, x_{i+1}, \ldots, x_{n-1})$ とする. $\vec{X}_1$ を束縛変数, $\vec{X}_2$ を自由変数とする $\vec{F}$ の分解表を考える. このとき, この分解表の列複雑度は高々2^q である.

(証　　明)

$i \leqq q$ のとき, 列複雑度は高々 2^q であるため, $i > q$ の場合を考える. 分解表の第 1 行目, 即ち, $\vec{X}_2 = (0, 0, \ldots, 0)$ の行を考える. このとき, 分解表の各要素は q ビットのベクトルであるので, 異なる要素の数は高々 2^q 個である. 従って, $\vec{a}$, $\vec{b} \in \{0,1\}^i, \vec{a} \neq \vec{b}$ としたとき, $\vec{F}(\vec{a}, \vec{0}) = \vec{F}(\vec{b}, \vec{0})$ となる $\vec{X}_1$ への二つの割り当て $\vec{a}$, $\vec{b}$ に対応する二つの列が存在する. 次に, これらの二つの列の第 j 行目 $(j > 0)$ を考える. このとき, $\vec{X}_2 = (x_i, x_{i+1}, \ldots, x_{n-1})$ の値を $\vec{c}$ とすれば, 定義 4.1 より, $\vec{F}$ は関係

$$\begin{aligned}
\vec{F}(\vec{a}, \vec{c}) &= \vec{F}(\vec{a}, \vec{0}) + \vec{F}(\vec{0}, \vec{c}) \\
\vec{F}(\vec{b}, \vec{c}) &= \vec{F}(\vec{b}, \vec{0}) + \vec{F}(\vec{0}, \vec{c})
\end{aligned}$$

を満たす. ここで, 記号 + は桁上がりを許す 2 進数の加算を示す. 従って, $\vec{F}(\vec{a}, \vec{c}) = \vec{F}(\vec{b}, \vec{c})$ が成立する. この関係は, すべての $j > 0$ で成立するので, $\vec{a}$, $\vec{b}$ に対応する二つの列のパターンが等しいことがわかる.

以上のことより, 分解表の列複雑度は高々2^q であることが証明できた.　(証明終)

[例 4.1]　表 4.1 に $n = 5$, $\vec{X}_1 = (x_0, x_1, x_2)$, $\vec{X}_2 = (x_3, x_4)$ の分解表の例を示す. 今, 分解表の各要素は 2 ビットのベクトルと仮定する. この時, 分解表には, 高々 4 つの異なったベクトルしか現れない. 分解表の第 1 行目, $(x_3, x_4) = (0, 0)$ の行を考える. 今, $(x_0, x_1, x_2) = (0, 1, 1)$ の列の要素と, $(x_0, x_1, x_2) = (1, 0, 0)$ の列の要素が等しいと仮定する. 即ち, $h_1 + h_2 = h_0$ とする. 次に, 分解表の第 2 行目, $(x_3, x_4) = (0, 1)$ の行を考え

(注4) : 文献 [4] では h が 2 値の場合のみ考案している. 文献 [5] でも $\vec{F}$ は単一出力関数の場合のみ考案している.

(注5) : 非対称フィルタの場合は, $n = N$ であり, 対称フィルタの場合は, $n = (N+1)/2$ である.

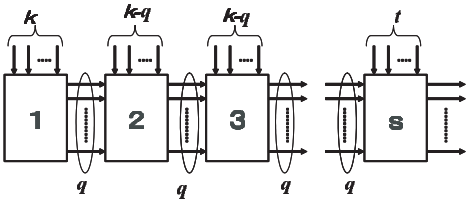

図 4.1　WS 関数のカスケード実現.

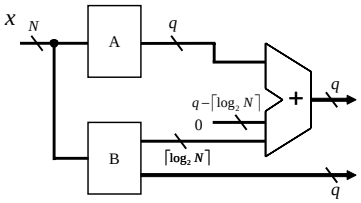

図 5.1　$2q$ 出力の WS 関数の算術分解.

る．WS 関数の性質より, 第 2 行目は, 第 1 行目に h_4 を加えたものである．従って, $(x_0, x_1, x_2) = (0, 1, 1)$ の列の要素 $h_1 + h_2 + h_4$ と, $(x_0, x_1, x_2) = (1, 0, 0)$ の列の要素 $h_0 + h_4$ が等しい．即ち, $h_1 + h_2 + h_4 = h_0 + h_4$ が成立する．同様なことが, 分解表の第 3 行目と第 4 行目に対しても成立する．即ち, $(x_0, x_1, x_2) = (0, 1, 1)$ の列の要素と, $(x_0, x_1, x_2) = (1, 0, 0)$ の列の要素が等しい．従って, $(x_0, x_1, x_2) = (0, 1, 1)$ の列のパターンと, $(x_0, x_1, x_2) = (1, 0, 0)$ の列のパターンが等しい．つまり, 第一行目で二つの列が等しければ, 全ての行でその二つの列は等しい．以上のことから, 分解表の列複雑度は高々4 であることが証明できる．　(例終り)

［定理 4.1］　q 出力 WS 関数を実現する LUT カスケードのレール数は q 本あれば十分である．また, 各セルの入力数は $q + 1$, 出力数は q あれば十分である．

(証　　明)

補題 4.1 と定理 3.2 より, 2 つのブロック間の信号線は，q 本である．少なくとも 1 本の外部入力を各セルに必要とする．　(証明終)

［定理 4.2］　任意の n 入力 q 出力 WS 関数を実現する LUT カスケードを k 入力 q 出力のセルで実現する場合, セル数は, $\lceil \frac{n-k-1}{k-q} \rceil + 2$ 個あれば十分である．

(証　　明)

定理 3.2 の方法で, WS 関数をカスケード実現すると, 図 4.1 を得る．このカスケードのレイル数を q, セル数を s とする．また, 最後のセルの入力数を t とすると, 関係 $k + (s - 2)(k - q) + t = n, 1 \leqq t \leqq k - q$ が成立する．これより, $s \leqq \lceil \frac{n-k-1}{k-q} \rceil + 2$ を得る．　(証明終)

k 入力 q 出力のセルのビット数は $q2^k$ であることより, 次の系を得る．

［系 4.1］　n 入力 q 出力 WS 関数を LUT カスケードで実現する際, k 入力 q 出力のセルを用いた場合, メモリのビット数は $(\lceil \frac{n-k-1}{k-q} \rceil + 2)2^k q$ あれば十分である．

［補題 4.2］　n 入力 q 出力 WS 関数を直接実現する ROM のビット数は $q2^n$ である．

［系 4.2］　n 入力 q 出力 WS 関数を ROM で直接実現する場合のメモリ量と k 入力 q 出力のセルを用いて LUT カスケードを実現する場合のメモリ量の比は, およそ $(\lceil \frac{n-k-1}{k-q} \rceil + 2)2^{k-n}$ である．

系 4.2 で $k = q + 1$ とおくと, 上の比は $(n - q)2^{q-n+1}$ となる．これより, $n - q$ の値が大きいとき, LUT カスケードを用いた場合のメモリの削減効果が高いことが分かる．

5.　WS 関数の算術分解

フィルタを実現する際，q は量子化ビットのビット数を表している．実験結果より q 出力の WS 関数を LUT カスケードで実現する場合，$(q + 1)$ 入力 q 出力のセルを必要とすることが分かる．従って q の値が大きい場合，FPGA 内に大容量の組込みメモリが必要になる．このような多出力の WS 関数を小規模 FPGA で実現するには WS 関数を複数のより小さい関数に分解して実現できる．

$2q$ 出力の WS 関数は，以下のような 2 つの関数に分解できる．h_i を $2q$ 出力の WS 関数の係数とすると，h_i は以下のように表せる．

$$h_i = 2^q h_{Ai} + h_{Bi},$$

ここで，h_{Ai} は h_i の最上位の q ビットを表し，h_{Bi} は最下位の q ビットを表している．この場合，$2q$ 出力の WS 関数を図 5.1 に示すように，2 つの WS 関数と 1 個の加算器とで実現できる．加算器は，$2q$ 入力，q 出力であることに注意されたい．

［定理 5.1］　$2q$ 出力の WS 関数

$$F(x_0, x_1, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} h_i x_i$$

は，以下の 2 つの WS 関数に分解できる．

$$2^q F_A(x_0, x_1, \ldots, x_{n-1}) + F_B(x_0, x_1, \ldots, x_{n-1})$$

ここで,

$$F_A(x_0, x_1, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} h_{Ai} x_i$$

は，q 出力の WS 関数,

$$F_B(x_0, x_1, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} h_{Bi} x_i$$

は，$q + \lceil \log_2 N \rceil$ 出力の WS 関数である．また

$$h_i = 2^q h_{Ai} + h_{Bi}$$

である．

これを **WS 関数の算術分解**と呼ぶ.

同様に，$4q$ 出力の WS 関数は，以下の 4 つの WS 関数に分解できる．h_i は，$4q$ 出力の WS 関数の係数である．よって h_i は以下のように表すことができる．

$$h_i = 2^{3q} h_{Ai} + 2^{2q} h_{Bi} + 2^q h_{Ci} + h_{Di}$$

ここで，h_{Ai}，h_{Bi}，h_{Ci}，h_{Di} は，q ビットの数を表す．図 5.2 に示すように，$4q$ 出力の WS 関数は 4 つの q 出力の WS 関数と 3 つの加算器とを用いて実現できる．ブロック A は q 出力の WS 関数，B,C,D は，$q + \lceil \log_2 N \rceil$ 出力の WS 関数である．出力につながる加算器は，$4q$ 入力，$2q$ 出力である．

表 4.1　WS 関数の分解表の例.

$\vec{X}_2=(x_3,x_4)$ \ $\vec{X}_1=(x_0,x_1,x_2)$	000	001	010	011	100	101	110	111
00	0	h_2	h_1	h_1+h_2	h_0	h_0+h_2	h_0+h_1	$h_0+h_1+h_2$
01	h_4	h_2+h_4	h_1+h_4	$h_1+h_2+h_4$	h_0+h_4	$h_0+h_2+h_4$	$h_0+h_1+h_4$	$h_0+h_1+h_2+h_4$
10	h_3	h_2+h_3	h_1+h_3	$h_1+h_2+h_3$	h_0+h_3	$h_0+h_2+h_3$	$h_0+h_1+h_3$	$h_0+h_1+h_2h_3$
11	h_3+h_4	$h_2+h_3+h_4$	$h_1+h_3+h_4$	$h_1+h_2+h_3+h_4$	$h_0+h_3+h_4$	$h_0+h_2+h_3+h_4$	$h_0+h_1+h_3+h_4$	$h_0+h_1+h_2+h_3+h_4$

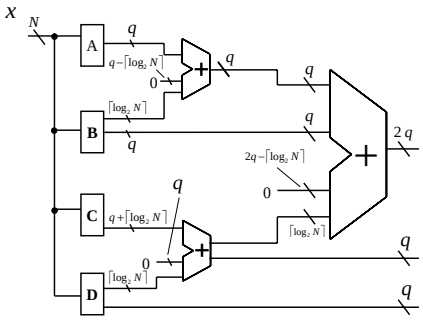

図 5.2　$4q$ 出力の WS 関数の算術分解.

6. 実験結果

6.1 実験方法

前章で得られた理論的結果を確認するために, 多数の FIR フィルタの WS 関数を LUT カスケードを用いて設計した. LUT カスケードの設計に分解表を用いるのは能率が悪いので, 二分決定グラフ (binary decision diagram :BDD) を用いた [10], [12]. この場合，BDD の幅が分解表の列複雑度に対応する．設計パラメータは，以下の通り：

- フィルタの種類:(HPF, LPF, BPF, BEF)
- タップ数 N: (9,17,33)
- 量子化ビット数 q: (8,10,12,14,16)
- 窓関数: (カイザー窓関数, ハニング窓関数)

である. 従って, 全部で 100 通り以上の WS 関数を実現した. フィルタ係数の丸めの方法は，$(q+1)$ 桁目が 0 の時切り捨て，1 の時 q 桁目に 1 を加算である．

6.2 FPGA への実装

FIR フィルタを実装する FPGA として，Altera 社の CycloneII FPGA [2] を用いた．これは，LUT 型のロジックエレメント (LE) と M4K と呼ばれる組込みメモリを持っている．M4K は 4096 ビットと 512 ビットのパリティビットを持っており，入出力のビット数を変更できる．LUT カスケードをこの M4K を用いて実現した．FPGA への実装を行った時の環境を表 6.1 に示す.

6.3 実験結果の考察

表 6.2 は ROM と WS 関数を LUT カスケードで実現した場合の比較である．紙面の都合により，($N=33$，$q=16$，カイザー窓関数）の場合のみを表に示す．対称フィルタで実現したため，$n=(N+1)/2$ となっている．

量子化ビットが 16 より小さい場合，いくつものフィルタ係数 $h_0, h_1, \ldots, h_{n-1}$ が 0 に丸められた．この場合，WS 関数は一部の入力変数にのみ依存し，フィルタとして正常に機能しなかった．よって，量子化ビット数を増やさなければならないが，q 出力の WS 関数を実現するためには，$q+1$ 入力，q 出力のセルが必要となる．従って FPGA で実現する場合，大容量の組込みメモリを必要となる．そこで，メモリ量を削減するために，図 5.2 で示した算術分解 ($q=4$) を用いる．

表 6.1　FPGA の実験環境

FPGA device: Cyclone II	
Device type:	EP2C70F896C7 [2]
Number of LEs:	68416
I/O pins:	622 (out of 896 total pins)
Memory bits:	1152000
Number of M4Ks:	250
Number of DSP blocks (9-bit):	300
Tool for Logic Synthesis and Fitting	
Altera, Quartus II V4.1 [2]	

算術分解を用いた WS 関数の出力数は，多くとも $q+\lceil \log_2 N \rceil$ ビットである．よって，$q=4, N=17$ とすると WS 関数を LUT カスケードで実現した場合，セルへの入力数は，定理 4.1 で示したように，多くとも $q+\lceil \log_2 N \rceil + 1 = 10$ 入力である．表 6.2 では，図 5.2 で示された量子化ビット=16 のフィルタを実現している．この表には，各カスケードのサイズ，BDD の最大幅，各カスケードのメモリ量，M4K の使用数，LE の使用数，動作周波数を示す．すべての場合において，カスケード実現によりメモリ量を削減できた．単一のメモリの場合 $2^{17}\times 16 = 2M$ ビット必要であるのに対し，LUT カスケードの場合は，29〜40k ビットのみ必要であった．これらを FPGA 上に実装すると，組込みメモリは全体の 4〜5%，LE は全体の 1%未満の使用量で実現できる．使用した CycloneII の FPGPA の M4K の個数は 250 個であり，全てのメモリ量は 1M ビットになる．よって本稿で提案した LUT カスケードと算術分解を組み合わせた実現は可能だが，単一のメモリでの実現では，この FPGA に収めることができない.

動作周波数は，101〜109MHz であった.

図 2.4 の h_0 に対応する変数 x_0 を BDD の根側に置き，$h_{(N-1)/2}$ に対応する変数 $x_{(N-1)/2}$ を BDD の葉側に置くことにする．この BDD の初期変数順序を用いるとサイズの小さな BDD を作成でき，また小さな LUT カスケードを作成できる．WS 関数の q や N が大きくなるに従って，列複雑度は大きくなる.

7. 結　　論

本論文では, FIR フィルタの組合せ回路の部分を表現する関数として WS 関数を定義した．さらに, WS 関数を LUT カス

表 6.2　WS 関数の ROM による実現と LUT カスケードによる実現の比較.

Filter Type (Cut Off Frequency)		HPF (5kHz)				LPF (5kHz)				BPF (5k-10kHz)				BEF (5k-10kHz)			
ROM	#IN	17				17				17				17			
	#OUT	16				16				16				16			
#Cascades		4				4				4				4			
Cascade		A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
#Cells		1	2	2	3	1	2	3	3	1	2	3	2	1	2	3	3
Cell0	#IN	10	10	10	10	7	8	9	10	7	10	9	10	9	10	10	10
	#OUT	4	6	7	7	4	5	5	6	4	7	6	6	4	7	7	7
Cell1	#IN		8	9	10		9	10	10		6	10	10		7	10	9
	#OUT		6	7	6		8	7	7		5	6	6		6	7	6
Cell2	#IN				7			8	9			5				6	6
	#OUT				5			6	7			4				5	5
Max Width BDD		11	56	66	90	8	130	91	111	8	88	63	58	10	50	71	78
Cascade Memory Bits [bits]		4096	7680	14336	13952	512	5376	11264	16896	512	7488	9344	12288	2048	7936	14656	10560
Total Memory Bits [bits]		40064				34048				29632				35200			
#M4K's(Out of 250)		13 (5 %)				12 (4 %)				12 (4 %)				13 (5 %)			
#LEs(Out of 68,416)		586 (< 1 %)				585 (< 1 %)				564 (< 1 %)				568 (< 1 %)			
Frequency[MHz]		105				104				109				101			

ケードと加算器とを用いて実現する方法を提案した．主要な結果を以下に示す：

（1） 図 5.1，図 5.2 で示したように，WS 関数を算術分解を用いて実現した.

（2） LUT カスケードでの実現は, 単一の ROM に比べ必要メモリ量の大幅な削減が可能.

（3） FIR フィルタを実現する際，算術分解は有効である.

量子化ビット数の値が大きい時は，出力の部分をいくつかに算術分解して実現しなければならない.

また，タップ数が大きい時，入力部分をいくつかに分割し，各部分で WS 関数を作成し，最終的に加算器を用いて実現できる．これにより劇的にメモリ量を削減できる．WS 関数及び加算器は LUT カスケードを用いて実現できる.

本論文では, ディジタルフィルタの分散演算のモデルとて WS 関数を定義したが, WS 関数は，このほか, DCT(Discrete Cosine Transform), DFT(Discrete Fourier Transform), 畳み込み演算にも利用可能である.

謝辞

本研究は, 一部, 文部科学省・科学研究費補助金, および, 文部科学省・北九州地域知的クラスター創成事業の補助金による. ご討論頂いた Jon.T.Butler 教授，永山忍博士，ヤマハ (株) の臼井章氏，LUT カスケード設計ツールで協力頂いた松浦宗寛氏，中原啓貴氏に感謝する.

文　献

[1] Actel Corporation, "CoreFIR: Finite implulse response (FIR) filter generator," http://www.actel.com/ipdocs/CoreFIR_IP_DS. pdf

[2] Altera: http://www.altera.com/

[3] R. J. Andraka, "FIR filter fits in an FPGA using a bit serial approach," *Proceedings of the EE-Times 3rd Annual PLD design Conference and Exhibit*, March 1993.

[4] R. L. Ashenhurst, "The decomposition of switching functions," *In Proceedings of an International Symposium on the Theory of Switching*, pp. 74-116, April 1957.

[5] H. A. Curtis, *A New Approach to The Design of Switching Circuits*, D. Van Nostrand Co., Princeton, NJ, 1962.

[6] T-T. Do, H. Kropp, C. Reuter, and P. Pirsch, "A flexible implementation of high-performance FIR filters on Xilinx FPGAs," *FPL 1998*, Estonia, Aug. 31 - Sept. 3, 1998, pp. 441-445.

[7] L. Mintzer, "FIR filters with field-programmable gate arrays," *Journal of VLSI Signal Processing*, Aug. 1993, pp. 120-127.

[8] K. Nakamura, T. Sasao, M. Matsuura, K. Tanaka, K. Yoshizumi, H. Qin, and Y. Iguchi, "Programmable logic device with an 8-stage cascade of 64K-bit asynchronous SRAMs," *Cool Chips VIII, IEEE Symposium on Low-Power and High-Speed Chips*, April 20-22, 2005, Yokohama, Japan.

[9] K. K. Parhi, *VLSI Digital Signal Processing Systems Design and Implementation*, John Wiley, New York, 1999.

[10] T. Sasao, M. Matsuura, and Y. Iguchi, "A cascade realization of multiple-output function for reconfigurable hardware," *International Workshop on Logic and Synthesis(IWLS01)*, Lake Tahoe, CA, June 12-15, 2001, pp. 225-230.

[11] T. Sasao, "Analysis and synthesis of weighted-sum functions," *International Workshop on Logic and Synthesis*, Lake Arrowhead, CA, USA, June 8-10, 2005, pp.455-462.

[12] T. Sasao and M. Matsuura, "A method to decompose multiple-output logic functions," *41st Design Automation Conference*, San Diego, CA, USA, June 2-6, 2004, pp. 428-433.

[13] T. Sasao, Y. Iguchi, T. Suzuki, "On LUT cascade realizations of FIR filters," *DSD2005, 8th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, Porto, Portugal, Aug. 30 - Sept. 3, 2005 (to be published).

[14] 笹尾勤, 井口幸洋, 鈴木隆広, "FIR フィルタの LUT カスケードによる実現について," 電子情報通信学会 VLSI 設計技術研究会, 2005 年 03 月 10-11 日, VLD2004-126.

[15] S. Yu and E. E. Swartzlander, "DCT implementation with distributed arithmetic," *IEEE Trans. on Computers*, Vol. 50, No. 9, Sept. 2001,pp. 985-991.

[16] S. A. White, " Applications of distributed arithmetic to digital signal processing: a tutorial review," *IEEE ASSP Magazine*, Vol. 6, No. 3, July 1989, pp. 4-19.