# 4-to-1 セレクタ論理及び 2-to-1 セレクタ論理を利用したバイリニア補間演算器の設計と評価

塩 雅史 † 柳澤 政生 †† 戸川 望 †

† 早稲田大学大学院基幹理工学研究科情報理工学専攻
†† 早稲田大学大学院基幹理工学研究科電子光システム学専攻

画像の補間処理は，ディジタル画像処理における基本技術の 1 つである．その応用技術としては，画像の解像度変換，フレームレート変換などがあげられる．これらの技術は近年の携帯電話やタブレット PC などの表示デバイスの小型により効率化が求められている．バイリニア補間は補間演算の 1 つであり，周囲 4 つの値から線形的に値を補間する．画像の拡大・縮小など実用的に用いられることも多い．本稿では，バイリニア補間演算をビットレベル式変形し 2-to-1 セレクタ及び 4-to-1 セレクタに帰着させることで，桁上げ伝搬遅延を削減し高速化したセレクタ論理帰着型バイリニア補間演算器を提案する．セレクタ論理帰着型バイリニア補間演算器において，2-to-1 セレクタのみに帰着する方法，4-to-1 セレクタへの帰着を併用する方法など，複数の方法で実装し評価・比較した．

# A Bi-Linear Interpolation Circuit Using 4-to-1 and 2-to-1 Selector Logics and Its Evaluations

Masashi SHIO† Masao YANAGISAWA†† Nozomu TOGAWA†

† Dept. of Computer Science and Engineering, Waseda University.
†† Dept. of Electronic and Photonic Systems, Waseda University.

Image interpolation is one of the basic techniques in digital image processing, and often used in a mobile phone and a tablet PC in recent years. Bi-linear interpolation is one of the interpolation operations and interpolates a value linearly from the values of the four circumferences. In this paper, we propose a high-speed bi-linear interpolation circuit reducing carry propagation delay by using 2-to-1 and 4-to-1 selector logics. Our bi-linear interpolation circuit is designed by using (1) only 2-to-1 selector logics and (2) a combination of 2-to-1 and 4-to-1 selector logics. We have implemented our bi-linear circuits and evaluated them. Experiments demonstrate their effectiveness and efficiency.

## 1 はじめに

近年，アプリケーションの規模の増大に伴い，システム LSI はより効率的で高速な演算が期待されている．それに伴い，特定の演算に特化した高速かつ効率的な演算器の設計が求められている．補間演算はこうした効率化・高速化が要求される演算の一つである．

画像の補間処理は，ディジタル画像処理における基本技術の 1 つである．その応用技術としては，画像の解像度変換 [1]，インターレス画像からプログレッシブ画像を作成する IP 変換，フレームレート変換，画像の回転などがあげられる．特に画像の解像度を変換する補間処理については，近年の高解像度ディスプレイや携帯電話などの表示デバイスの多様化により，非常に多くの場面で利用されるようになってきている．しかし，補間処理は計算量が大きく，解像度変換等の実行速度にも多大な影響を与える．したがって補間処理に用いられる補間演算に対して計算量の削減が必要であると考えられる．

補間演算は，得られたデータ列から範囲内の値を推定する演算であり，画像の拡大・縮小や，歪みの補正などに用いられる．中でもバイリニア補間 [3] は周囲 4 つの点から線形的に値を補間する演算であり，2 つの値を直線的に結ぶことで補間を実行する線形補間の組み合わせとして表すこともできる．したがって，線形補間演算を効率化することでバイリニア補間の効率化が実現される．

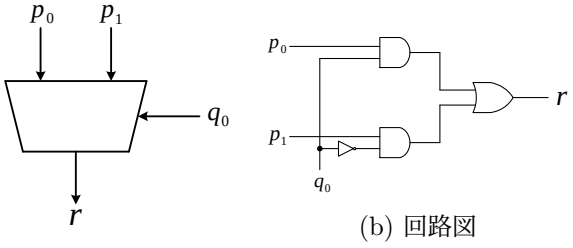
演算を高速化する手法としてビットレベル式変形がある．ビットレベル式変形は演算をビットレベルで表現し整理することによって回路設計を最適化する手法である [2].

いま，線形補間演算に対しビットレベル式変形を実行すると，その部分積が $p_0q_0 + p_1\bar{q}_0$ ($p_0$, $p_1$, $q_0$は 1 ビット変数) という形式により記述できることに気付く [4]．$p_0q_0 + p_1\bar{q}_0$ を評価すると，これは $q_0 = 1$ のとき $p_0$ を，$q_0 = 0$ のとき $p_1$ を出力する，表 1 及び図 1 に示すような 2-to-1 セレクタとなっている．2-to-1 セレクタ $p_0q_0 + p_1\bar{q}_0$ は値域が $0 \leq p_0q_0 + p_1\bar{q}_0 \leq 1$ でありけた上げがなく，この部分積を前処理としてまとめて事前演算すれば部分積を 1/2 程度に削減することができる．すなわち，線形補間演算をビットレベル式変形し 2-to-1 セレクタに帰着することで，部分積のけた上げをすることなく部分積を 1/2 程度に削減し高速な演算が期待できる．

同様にバイリニア補間に対しビットレベル変形を実行すると，上記で述べた 2-to-1 セレクタに加え

表 1: セレクタ論理の真理値表.

| 入力 | | | 出力 |
|---|---|---|---|
| $p_0$ | $p_1$ | $q_0$ | $r$ |
| $P_0$ | $P_1$ | 0 | $P_1$ |
| $P_0$ | $P_1$ | 1 | $P_0$ |




(a) ブロック図　(b) 回路図

図 1: 2-to-1 セレクタ.

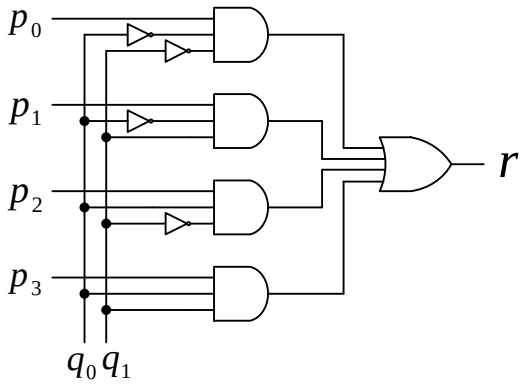



図 2: 4-to-1 セレクタ回路図.

$p_0\overline{q_0}\overline{q_1} + p_1\overline{q_0}q_1 + p_2q_0\overline{q_1} + p_3q_0q_1$ ($p_0$, $p_1$, $p_2$, $p_3$, $q_0$, $q_1$は 1 ビット変数) という形式により記述できることに気付く．これは図 2 に示すような 4-to-1 セレクタによって実現される．したがって線形補間と同様に帰着部分をまとめて事前演算することにより部分積を削減し高速な演算が期待できる．

本稿ではバイリニア補間演算をビットレベル式変形し 2-to-1 セレクタ及び 4-to-1 セレクタに帰着させることで，桁上げ伝搬遅延を削減し高速化したセレクタ論理帰着型バイリニア補間演算器を提案する．セレクタ論理帰着型バイリニア補間演算器において，2-to-1 セレクタのみに帰着する方法，4-to-1 セレクタへの帰着を併用する方法など，複数の方法で実装し評価・比較した．

## 2 セレクタ論理を利用した線形補間演算器の設計

本章では線形補間を取り上げ，線形補間演算をビットレベル式変形することでセレクタ論理に帰着した線形補間演算器を設計する [4].

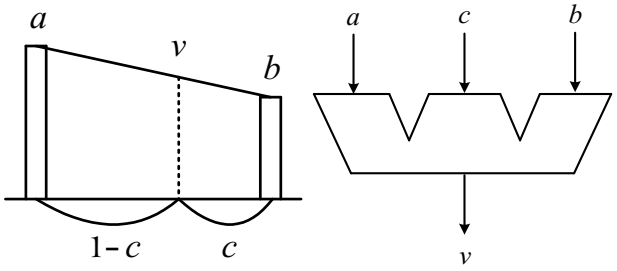
### 2.1 線形補間

補間とは既知のデータの範囲内で値を推定することである．線形補間は 2 つの端点を直線で結んで補間する演算であり，補間演算の中でも実用的に用いられることが多い [1].

2 つの端点を示す入力信号を $a$, $b$ とし，その間の値を $(1-c) : c$ の割合で線形補間したときの出力を $v$ とする．ただし $a$, $b$ は 0 以上の整数で，$c$ は $0<c<1$ の固定小数点数とする．このとき線形補間は下式で表現できる．

$$v = ac + b(1-c) \tag{1}$$

線形補間の概要を図 3(a) に，ブロック図を図 3(b) に示す.




(a) 線形補間の概要　(b) 線形補間のブロック図

図 3: 線形補間.

### 2.2 セレクタ論理帰着型線形補間演算器

さて，ここで線形補間演算をビットレベル式変形しよう．入力信号 $a$, $b$ は $n$ ビットの 0 以上の整数であり，これをビットレベルで表現すると下式となる.

$$\begin{aligned} a &= [a_{n-1}a_{n-2}...a_1a_0] \\ &= a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + ... + a_02^0 \\ &= \sum_{j=0}^{n-1} a_j 2^j \end{aligned} \tag{2}$$

$$\begin{aligned} b &= [b_{n-1}b_{n-2}...b_1b_0] \\ &= b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + ... + b_02^0 \\ &= \sum_{j=0}^{n-1} b_j 2^j \end{aligned} \tag{3}$$

また，補間の割合を示す $c$ は $0<c<1$ の $n$ ビットの ko 小数によって表すと，そのビットレベル式表現は下式となる.

$$\begin{aligned} c &= [0.c_{n-1}c_{n-2}...c_1c_0] \\ &= c_{n-1}2^{-1} + c_{n-2}2^{-2} + ... + c_02^{-n} \\ &= \sum_{i=0}^{n-1} c_i 2^{-(n-i)} \end{aligned} \tag{4}$$

式 (1) に式 (2), (3), (4) を代入すると，線形補間演算のビットレベル式表現は下式のように表せる.

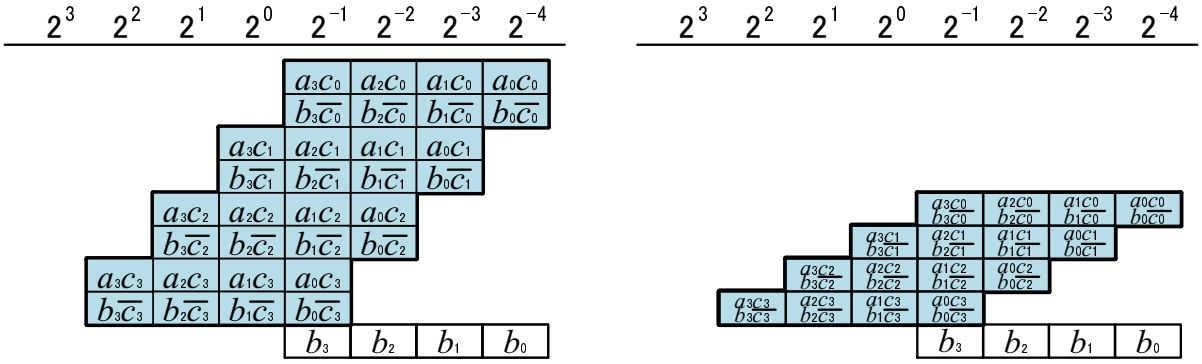
$$\begin{aligned} \text{式 (1)} = &\left(\sum_{j=0}^{n-1} a_j 2^j\right)\left(\sum_{i=0}^{n-1} c_i 2^{-(n-i)}\right) \\ &+ \left(\sum_{j=0}^{n-1} b_j 2^j\right)\left(\underline{1 - \sum_{i=0}^{n-1} c_i 2^{-(n-i)}}\right) \end{aligned} \tag{5}$$

ここで，2 の補数の性質から式 (5) の下線部は下式となる.

$$1 - \sum_{i=0}^{n-1} c_i 2^{-(n-i)} \quad = \sum_{i=0}^{n-1} \overline{c_i} 2^{-(n-i)} + 2^{-n} \tag{6}$$

式 (6) を式 (5) に代入し整理すると下式となる.

$$\begin{aligned} \text{式 (5)} = &\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} (\underline{a_jc_i + b_j\overline{c_i}}) 2^{-(n-i-j)} \\ &+ \sum_{j=0}^{n-1} b_j 2^{-(n-j)} \end{aligned} \tag{7}$$




(a) セレクタ論理に帰着させない場合　　(b) セレクタ論理に帰着させた場合

図 4: $n = 4$ の場合の線形補間演算の部分積.

ここで式 (7) を見ると下線部は 2-to-1 セレクタ $p_0q_0 + p_1\bar{q}_0$ と同形式となっており，セレクタ論理に帰着可能であることに気付く．セレクタ論理に帰着させない場合の式 (7) の部分積は合計で $2n^2 + n$ 個である．一方，式 (7) の下線部を 2-to-1 セレクタに帰着した場合，セレクタ論理部分を事前に演算することにすれば，帰着した部分の部分積数は 1/2 程度となり部分積の個数は $n^2 + n$ 個となる．したがって 2-to-1 セレクタに帰着させることで $n^2$ 個の部分積を削減でき，高速化が期待される．

$n = 4$ の場合の線形補間演算の部分積を図 4 に示す．実線で囲んだ各部が部分積 1 つ 1 つを表し，太線で囲んだ色つき部分が 2-to-1 セレクタに帰着可能な部分である．図 4 より，$n = 4$ の場合の線形補間演算では 2-to-1 セレクタに帰着することによって，部分積が 36 個から 20 個に削減できることが分かる．

## 3 セレクタ論理を利用したバイリニア補間演算器の設計

本章ではバイリニア補間を取り上げ，4-to-1 及び 2-to-1 セレクタに帰着したバイリニア補間演算器を提案する．

### 3.1 バイリニア補間

バイリニア補間 [3] は，画素の補間位置を包括する正方形領域を構成する 4 画素の画素値から線形的に値を補間することで，補間位置における画素値を算出する手法である．縦横に対してそれぞれ線形補間が行われるためバイリニア法と呼ばれる．図 5 にバイリニア法による画素値の補間の様子を示す．

ここで補間位置の周囲の画素の画素値を図 5 のように $a$，$b$，$c$，$d$ とし，横方向の補間の割合を $x$，縦方向の補間の割合 $y$ とする．ただし $a$，$b$，$c$，$d$ は 0 以上の整数，$x$，$y$ は $0<x, y<1$ を満たす固定小数点数とする．求める補間位置の画素値を $u$ とすると，$u$ は下式で表せる．

$$u = \{\underline{ax + b(1-x)}\}y + \{\underline{cx + d(1-x)}\}(1-y) \tag{8}$$

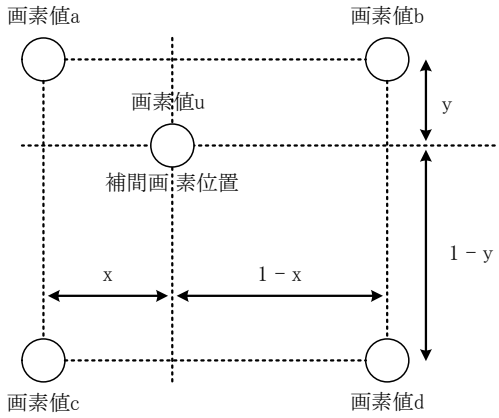



図 5: バイリニア補間.

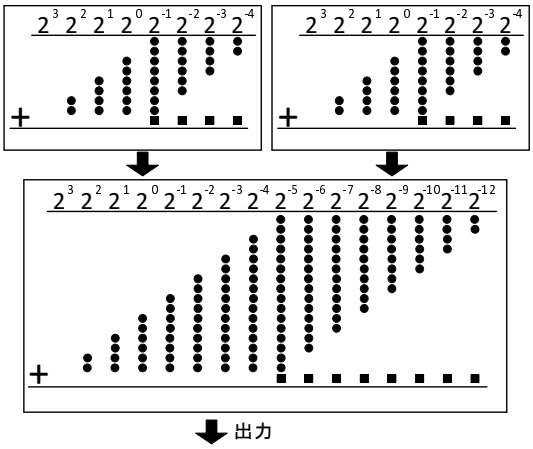
### 3.2 セレクタ論理帰着型バイリニア補間演算器

本節ではセレクタ論理帰着型バイリニア補間演算器について，2.2 節のセレクタ論理帰着型線形補間演算器を組み合わせて実現する方法，バイリニア補間演算に直接ビットレベル変形を適用し，4-to-1 及び 2-to-1 セレクタに帰着させる方法の 2 種類の設計方法を提案する．

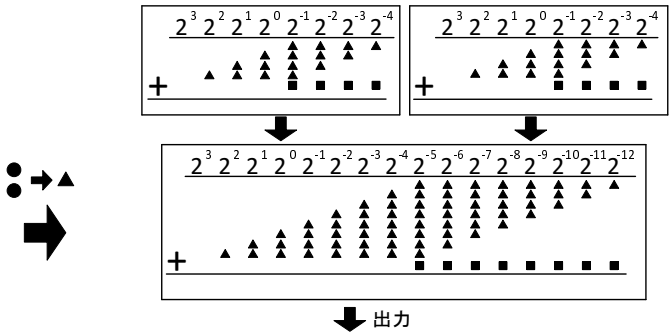
#### 3.2.1 セレクタ論理帰着型線形補間演算器を組み合わせる方法

式 (8) は下線部に着目するとそれぞれ入力が $\{a, b, x\}$，$\{c, d, x\}$ の線形補間の形式となっており、また式全体としても $\{ax + b(1-x), cx + d(1-x), y\}$ の 3 項を入力とした線形補間の形となっている．したがってバイリニア補間は図 6 に示すように線形補間演算器 3 つによって実現できる．2 章で説明したように線形補間はセレクタ論理に帰着可能であるので，図 6 からセレクタ論理帰着型線形補間演算器を 3 つ用いることでセレクタ論理帰着型のバイリニア補間演算器を設計することが可能であることが分かる．この設計方法で $n$ ビット入力の場合，セレクタへ帰着する前の部分積数が $12n^2 + 4n$ セレクタへ帰着した後の部分積数が $6n^2 + 4n$ であり，セレクタへの帰着に $6n^2$ 個の部分積を削減できる．

$n = 4$ の場合の部分積を図 7 に示す．図 7(a)，(b) の丸，三角及び四角 1 つ 1 つが部分積 1 ビットを表




(a) セレクタ論理に帰着させない場合




(b) セレクタ論理に帰着させた場合

図 7: $n = 4$ の場合のバイリニア補間演算の部分積.

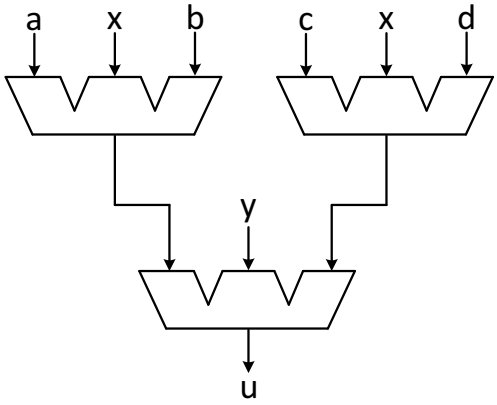



図 6: バイリニア補間のブロック図.

し，黒丸の部分は 2-to-1 セレクタに帰着できる部分，黒三角の部分は 2-to-1 セレクタへ帰着した部分である．2-to-1 セレクタにより黒丸 2 つが黒三角 1 つに帰着される．図 7 からセレクタへの帰着によって部分積数が 208 個から 112 個へ 96 個削減されていることが分かる．

### 3.2.2 4-to-1 及び 2-to-1 セレクタに帰着させる方法

本項ではバイリニア補間演算に直接ビットレベル変形を適用し，4-to-1 及び 2-to-1 セレクタに帰着させたバイリニア補間演算器を提案する．

バイリニア補間をセレクタ論理に帰着するために式 (1) に対してビットレベル式変形を行う．式 (8) 中の $a$, $b$, $c$, $d$, $x$, $y$ はそれぞれ以下のようにビットレベル式で表現できる．

$$a = \sum_{j=0}^{n-1} a_j 2^j \tag{9}$$

$$b = \sum_{j=0}^{n-1} b_j 2^j \tag{10}$$

$$c = \sum_{j=0}^{n-1} c_j 2^j \tag{11}$$

$$d = \sum_{j=0}^{n-1} d_j 2^j \tag{12}$$

$$x = \sum_{i=0}^{n-1} x_i 2^{-(n-i)} \tag{13}$$

$$y = \sum_{h=0}^{n-1} y_h 2^{-(n-h)} \tag{14}$$

式 (9)〜(14) により式 (8) は以下のようになる．

$$\begin{aligned}
&\text{式 (8)} = \\
&(\sum_{j=0}^{n-1} a_j 2^j)(\sum_{i=0}^{n-1} x_i 2^{-(n-i)})(\sum_{h=0}^{n-1} y_h 2^{-(n-h)}) \\
&+(\sum_{j=0}^{n-1} b_j 2^j)(1-\underline{\sum_{i=0}^{n-1} x_i 2^{-(n-i)}})(\sum_{h=0}^{n-1} y_h 2^{-(n-h)}) \\
&+(\sum_{j=0}^{n-1} c_j 2^j)(\sum_{i=0}^{n-1} x_i 2^{-(n-i)})(1-\underline{\sum_{h=0}^{n-1} y_h 2^{-(n-h)}}) \\
&+(\sum_{j=0}^{n-1} d_j 2^j)(1-\underline{\sum_{i=0}^{n-1} x_i 2^{-(n-i)}})(1-\underline{\sum_{h=0}^{n-1} y_h 2^{-(n-h)}})
\end{aligned} \tag{15}$$

ここで式 (15) の下線部は 2 の補数の性質からそれぞれ以下のように変形できる．

$$-\sum_{i=0}^{n-1} x_i 2^{-(n-i)} = -1 + \sum_{i=0}^{n-1} \overline{x_i} 2^{-(n-i)} + 2^{-n} \tag{16}$$

$$-\sum_{h=0}^{n-1} y_h 2^{-(n-h)} = -1 + \sum_{h=0}^{n-1} \overline{y_h} 2^{-(n-h)} + 2^{-n} \tag{17}$$

式 (15) に式 (16)，(17) を代入し，展開，整理すると以

下のようになる．

$$
\begin{aligned}
&\text{式 (15)} = \\
&(\sum_{j=0}^{n-1} a_j 2^j)(\sum_{i=0}^{n-1} x_i 2^{-(n-i)})(\sum_{h=0}^{n-1} y_h 2^{-(n-h)}) \\
&+(\sum_{j=0}^{n-1} b_j 2^j)(\sum_{i=0}^{n-1} \overline{x}_i 2^{-(n-i)} + 2^{-n})(\sum_{h=0}^{n-1} y_h 2^{-(n-h)}) \\
&+(\sum_{j=0}^{n-1} c_j 2^j)(\sum_{i=0}^{n-1} x_i 2^{-(n-i)})(\sum_{h=0}^{n-1} \overline{y}_h 2^{-(n-h)} + 2^{-n}) \\
&+(\sum_{j=0}^{n-1} d_j 2^j)(\sum_{i=0}^{n-1} \overline{x}_i 2^{-(n-i)} + 2^{-n})(\sum_{h=0}^{n-1} \overline{y}_h 2^{-(n-h)} + 2^{-n}) \\
=&\sum_{h=0}^{n-1}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(\underline{\underline{a_j x_i y_h + b_j \overline{x_i} y_h + c_j x_i \overline{y_h} + d_j \overline{x_i y_h}}})2^{-(2n-h-i-j)} \\
&+\sum_{h=0}^{n-1}\sum_{j=0}^{n-1}(\underline{b_j y_h + d_j \overline{y_h}})2^{-(2n-h-j)} \\
&+\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(\underline{c_j x_i + d_j \overline{x_i}})2^{-(2n-i-j)} \\
&+\sum_{j=0}^{n-1} d_j 2^{-(2n-j)} \qquad (18)
\end{aligned}
$$

式 (18) の二重下線部及び下線部に着目すると，二重下線部は図 2 に示した 4-to-1 セレクタ，下線部は図 1 に示した 2-to-1 セレクタの形式になっていることが分かる．したがってこれらの部分をセレクタ論理を用いて実装することで演算の効率化が期待できる．

$n = 4$ の場合の部分積を図 8 に示す．図 8(a), (b) の丸, 三角及び四角 1 つ 1 つが部分積 1 ビットを表し，白丸の部分は 4-to-1 セレクタ，黒丸の部分は 2-to-1 セレクタに帰着できる部分であり，白三角の部分は 4-to-1 セレクタ，黒三角の部分は 2-to-1 セレクタへ帰着した部分である．セレクタへを帰着することによって全体の部分積数は 324 個から 100 個に削減されている．図 7(b) と図 8(b) の部分積数を比較すると，図 7(b) が 112 個，図 8(b) が 100 個となり単純な部分積数では図 8(b) の方が少ない．

$n$ ビット入力においてはセレクタ論理帰着型線形補間演算器を組み合わせる場合 (図 7(b)) で $6n^2 + 4n$ 個の部分積，4-to-1 及び 2-to-1 セレクタに帰着させる場合 (図 8(b)) で $n^3 + 2n^2 + n$ 個の部分積であるので $n = 5$ 以上では 4-to-1 セレクタへの帰着を利用する図 8(b) の場合の方が部分積数は多くなってしまう．ただし遅延はクリティカルパスでの加算段数によって変わるため，4-to-1 セレクタを利用した場合の方が遅延が小さくなる可能性もあると考えられる．

## 4 評価実験

本章では，2 章で提案したセレクタ論理帰着型バイリニア補間演算器をいくつかの方法で実装し，論理合成ツールによって評価・比較する．なお，セレクタ論理帰着型バイリニア補間演算器の設計にはハードウェア記述言語 VHDL を用いた．

### 4.1 実験方法

論理合成ツールによる評価では，Synopsys 社の Design Compiler D-2010.03-SP5 のトポグラフィカルモードを用いた．また，解析の際，セルライブラリには STARC [*1](CMOS 90 nm) の設計ルールを用いた．

4bit 入力 12bit 出力，8bit 入力 24bit 出力の 2 種類のセレクタ論理帰着型バイリニア補間演算器をそれぞれ以下の 4 つの方法で実装した．

1. HDL の算術演算子のみで演算する [Op]
2. 算術演算子による線形補間の組み合わせによって実現する [comb Op]
3. 2-to-1 セレクタ帰着型線形補間の組み合わせによって実現する [comb 2Sel]
4. 4-to-1 及び 2-to-1 セレクタにより部分積を生成し HDL の算術演算子によって加算する [4Sel]

[comb Op] 及び [comb 2Sel] では図 6 に示したように線形補間の組み合わせとして設計する．ただし，[comb Op] の線形補間では，算術演算子のみで演算を行うものとし，[comb 2Sel] の線形補間では，2-to-1 セレクタによって部分積を生成し，部分積加算には算術演算子を利用する．

表 2: バイリニア補間演算器の論理合成結果 (遅延最小化).

| bit | 手法 | 遅延時間 [ns] | 面積 [$\mu$m$^2$] |
|---|---|---|---|
| 4 | Op | 1.17 (100%) | 5483 (100%) |
| | comb Op | 1.22 (104%) | 5111 ( 93%) |
| | comb 2Sel | 1.03 ( 88%) | **3254 ( 59%)** |
| | 4Sel+Op | **0.90 ( 77%)** | 4351 ( 79%) |
| 8 | Op | 1.70 (100%) | 18372 (100%) |
| | comb Op | 1.72 (101%) | 20048 (109%) |
| | comb 2Sel | 1.46 ( 86%) | **10945 ( 60%)** |
| | 4Sel+Op | **1.36 ( 80%)** | 33124 (180%) |

### 4.2 論理合成結果

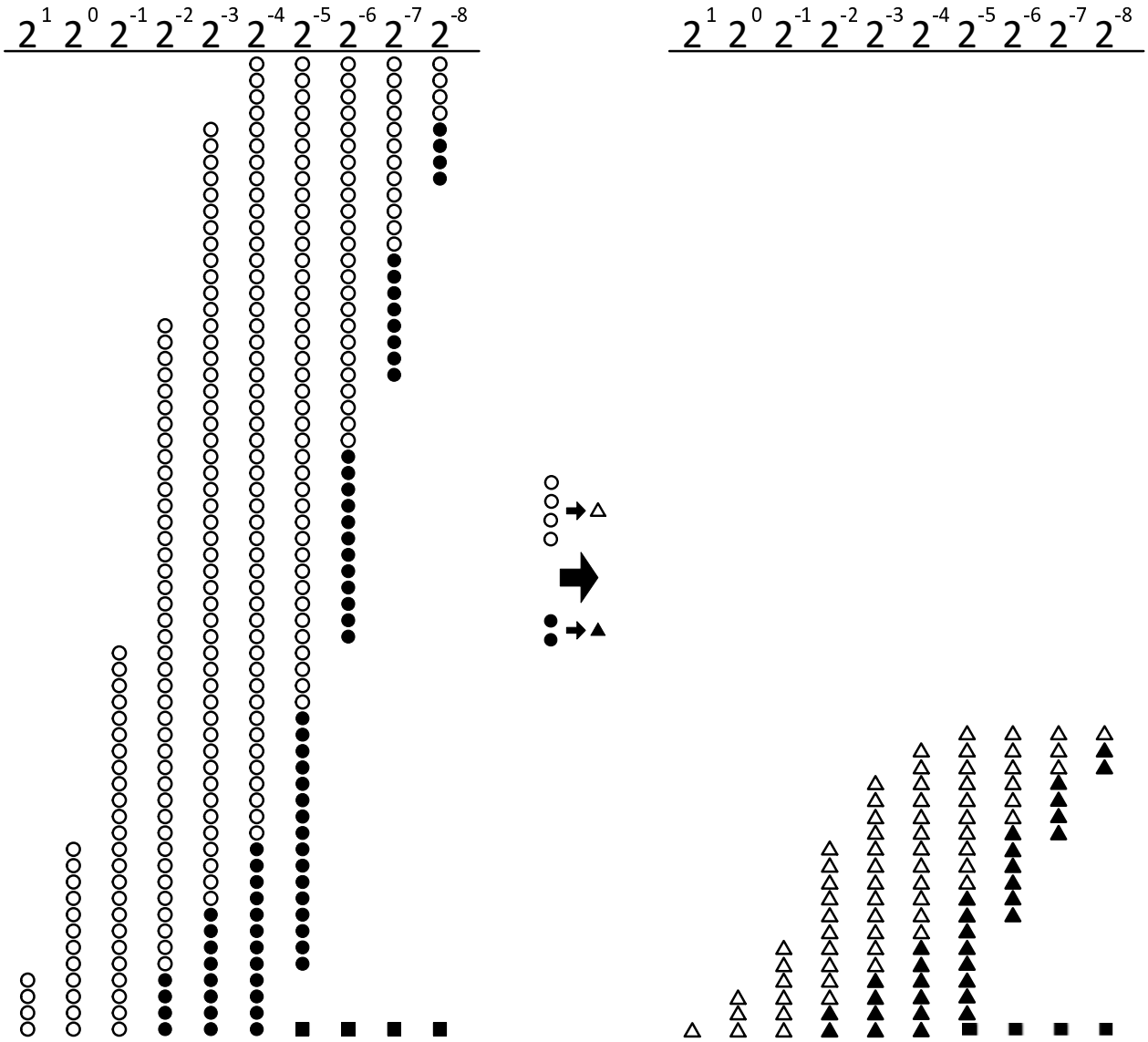
セレクタ論理帰着型バイリニア補間演算器の論理合成結果として，遅延最小化の制約で合成した場合の遅延と面積の関係を表 2 に示す．

表 2 において [Op] とその他の手法を比較すると，遅延は今回提案した [4Sel] が最も削減率が高く，4bit では 23%，8bit では 20% 削減された．面積は [comb 2Sel] が最も削減率が高く，4bit では 41%，8bit では 40% 削減された．

## 5 おわりに

本稿では 4-to-1 セレクタ及び 2-to-1 セレクタに帰着したバイリニア補間演算の設計，評価を行った．[4Sel] では，ビットレベル式変形における展開によって線形補間の組み合わせによる実装よりもセレクタへの帰着前の部分積数が増加してしまうが，一方で最も遅延の削減率が高く，最大で 23% の遅延削減を達成した．こ

[*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センターを通じ，株式会社半導体理工学研究センター (STARC) と株式会社先端 SoC 基盤技術開発 (ASPLA) の協力で開発されたものである．




(a) セレクタ論理に帰着させない場合　　(b) セレクタ論理に帰着させた場合

図 8: $n = 4$ の場合のバイリニア補間演算の部分積.

れは遅延に関わる部分での加算段数について効果的に削減がされているためだと考えられる.

今後の課題として，画像処理等の実アプリケーションでの遅延・面積削減効果の実証が挙げられる.

**謝辞** 本研究の一部は，科研費 (課題番号 25540021) による.

## 参考文献

[1] A. Belahmidi and F. Guichard, “A partial differential equation approach to image zoom,” in *Proc. ICIP*, pp. 649–652, 2004.

[2] M. J. Schulte, L. Marquette, S. Krithivasan, E. G. Walters III, and J. Glossner, “Combined Multiplication and sum-of-squares units,” in *Proc. IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, pp. 204–214, 2003.

[3] 伊藤 貴之, CG とビジュアルコンピューティング入門，森北出版, 2006.

[4] M. Shio, M. Yanagisawa, and N. Togawa, ”Linear and bi-linear interpolation circuits Using selector Logics and their evaluations,” in *Proc. IEEE International Symposium on Circuits and Systems*, pp. 1436-1439, 2014.