Vol.2012-SLDM-155 No.7
Vol.2012-EMB-24 No.7
2012/3/2

セレクタ論理を利用した高速補間演算器設計

岩田愛実[†1]　吉原弘峰[†1]
柳澤政生[†2]　戸川望[†1]

補間演算は既知のデータ列を基にして各区間の範囲内を埋める数値または関数を求める演算で，画像の拡大，縮小や魚眼画像の補正といった処理に利用される．キュービックスプライン補間は周囲 4 点から 3 次関数を用いることで補間を行うため精度が高く，より滑らかな補間ができるため実用的に用いられる．しかし，キュービックスプライン補間では扱う既知データが多く，計算が複雑なために処理に時間がかかる．そのため，補間演算処理をリアルタイムに行うには演算の高速化が必要である．本稿では，補間演算器にセレクタ論理を組み込むことで桁上げ伝搬遅延を削減し，演算器を高速化する手法を提案する．周囲 2 点を基に補間を行う線形補間では，算術演算子を用いて設計した従来の線形補間演算器に比べ，遅延時間は最大 15 %削減された．キュービックスプライン補間演算では，従来のキュービックスプライン補間演算器に比べ，遅延時間は最大 25 %削減された．

A Fast Interpolation Unit Using Selector Logics

Manami Iwata,[†1] Hiromine Yoshihara,[†1]
Masao Yangisawa[†2] and Nozomu Togawa[†1]

Interpolation is a technique that fills the gaps between existing data, which is often applied to image scaling and superresolution. Cubic spline interpolation, one of the interpolation techniques, obtains a cubic function based on the four existing points and fills their gaps very smoothly and precisely. However, it takes a lot of time because it requires many data and complex calculation. Speeding-up cubic spline interpolation is the key to realize a practical image scaling system. In this paper, we firstly focus on linear interpolation and propose a high-speed linear interpolation circuit based on "selector logics." Secondly, we propose a high-speed cubic spline interpolation circuit composed of our proposed linear interpolation circuits. Experimental results demonstrate that our linear interpolation circuit improves the performance by 15 % and that our cubic interpolation circuit improves the performance by 25 %, compared to a conventional interpolation design.

1. はじめに

情報通信やマルチメディア用途の LSI は要求仕様実現のためにアプリケーション専用演算器を必要とする．こうした専用演算器は，アプリケーション規模の増大に伴い，より高速な演算を期待されている．特定の演算に特化した効率的な回路を設計することで専用演算器の性能を向上させることが必要である．本稿では専用演算器として補間演算器を取り上げ，線形補間演算器とキュービックスプライン補間演算器を設計し，高速化を図る．

補間演算とは，既知のデータ列を基にして各区間内を埋める数値を求める演算であり，画像の拡大，縮小や歪みの補正などに用いられる．補間演算のうち最もよく使われるのが線形補間である[2]．線形補間では 2 つの端点を直線で結んで未知データを求める．よって，演算自体は単純であるが，実際の応用場面では精度が不十分となる．例えば，画像を拡大するときに倍率が大きくなると線形補間では特に画像の輪郭部分が不鮮鋭になる問題[6] をあげることができる．

線形補間よりも精度の高い補間演算としてキュービックスプライン補間がある．キュービックスプライン補間では周囲 4 点を利用して未知データの 3 次関数を求めることで補間を行うため，精度が高く滑らかな補間ができる．一方で，扱う既知データの増加により計算が複雑となるため処理に時間がかかる．高い精度を維持してリアルタイム補間を行うには，キュービックスプライン補間の高速化は必須である．

これに対し著者らの知る限り，補間演算のアルゴリズムを改善し高速化を図る研究は多くあるが[1),3)]，線形補間演算器やキュービックスプライン補間演算器をアーキテクチャレベルで最適化する既存研究はほとんどない．

本稿では，補間演算器にセレクタ論理を組み込むことで桁上げ伝搬遅延を削減し，演算器を高速化する手法を提案する．周囲 2 点を基に補間を行う線形補間では，算術演算子を用いて設計した従来の線形補間演算器に比べ，遅延時間は最大 15 %削減された．キュービックスプライン補間演算では，従来のキュービックスプライン補間演算器に比べ，遅延時間は最大 25 %削減された．

†1 早稲田大学基幹理工学部情報理工学科
Dept. of Computer Science and Engineering, Waseda University
†2 早稲田大学基幹理工学部電子光システム学科
Dept. of Electronic and Photonic Systems, Waseda University

Vol.2012-SLDM-155 No.7
Vol.2012-EMB-24 No.7
2012/3/2

本稿は次のように構成される．第 2 章では，セレクタ論理の演算式と出力の特性を説明する．第 3 章では，まず線形補間演算をビットレベル式変形してセレクタ論理に帰着させることで高速な線形補間演算器の設計を提案する．次に，線形補間演算器を利用したキュービックスプライン補間演算器の設計を提案する．第 4 章では，セレクタ論理帰着型線形補間演算器とキュービックスプライン補間演算器を実装し，従来手法と比較・検討する．第 5 章ではまとめを述べる．

2. セレクタ論理

1 ビットの変数 x，y，z と w を考える．このときセレクタ論理は以下の式で表現される．

$$w = xz + y\bar{z}. \tag{1}$$

式 (1) は MOS スイッチを使うことで，遅延時間と面積が AND 回路，OR 回路といった基本ゲート 1 つ分と同等になるセレクタ回路を作ることができる．一般に，1 ビットの入力が 2 つまたは 3 つの演算では出力の最大値と最小値の差が 1 より大きく，桁上げが発生する．ところが，セレクタ論理の出力の最大値と最小値の差は，

$$\max \mid xz + y\bar{z} \mid - \min \mid xz + y\bar{z} \mid \leq 1 \tag{2}$$

となる．すなわち，セレクタ論理では出力は 0 または 1 となり，桁上げが発生しない．よって，セレクタ論理を用いることで桁上げ伝搬遅延を削減でき，高速化が図れる．

3. 補間演算器の設計

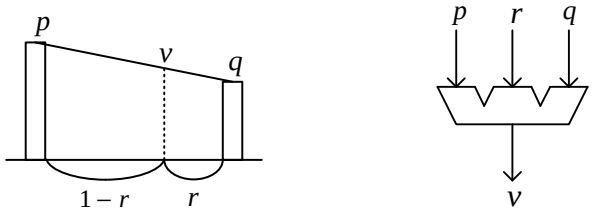
線形補間は，補間法の中では計算コストが低いためよく用いられる．補間の精度を上げるには補間に利用する周囲の点の数を増やせばよいが，その分計算コストが増える．補間演算器を設計するには，補間の精度と計算コストのトレードオフの関係を考慮する必要がある．

キュービックスプライン補間は，線形補間より精度を向上させた補間である．キュービックスプライン補間では扱う既知データが多く，計算が複雑なため処理に時間がかかる．よって，補間演算処理をリアルタイムに行うには演算の高速化が必要である．本章では線形補間とキュービックスプライン補間をセレクタ論理に帰着させ，高速化する手法を提案する．

3.1 線形補間演算器

本節では，セレクタ論理に帰着させた線形補間演算器の設計方法を提案する．線形補間とは，2 つの端点を直線で結んで求める補間演算であり，演算量が多い補間法の中でも比較的演算量が少ないため実用的に用いられることが多い．

2 つの端点を示す入力信号を p, q とし，その間の値を $(1-r) : r$ の割合で線形補間したときの出力を v とすると，線形補間の演算式は式 (3) で表される．



(a) 線形補間の概要.　(b) 線形補間演算器.

図 1 線形補間.

$$v = pr + q(1-r). \tag{3}$$

なお，p, q は整数で，r は $0 < r < 1$ とする．図 1 に線形補間を示す．図 1(a) は線形補間の概要，(b) は設計する線形補間演算器で，p, q, r が入力で v が出力である．

ビットレベル式変形を用いて，線形補間演算器をセレクタ論理に帰着させよう．入力信号 p, q を n ビットの符号付き整数とすると，p, q のビットレベル表現は，

$$p = [p_{n-1} \cdots p_1 p_0]_{\mathrm{b}} = -p_{n-1}2^{n-1} + \sum_{j=0}^{n-2} p_j 2^j \tag{4}$$

$$q = [q_{n-1} \cdots q_1 q_0]_{\mathrm{b}} = -q_{n-1}2^{n-1} + \sum_{j=0}^{n-2} q_j 2^j \tag{5}$$

となる．補間の割合を示す信号 r を $0 < r < 1$ を満たす n ビットの小数とすると，r のビットレベル表現は，

$$r = [0.r_{n-1} \cdots r_1 r_0]_{\mathrm{b}} = \sum_{i=0}^{n-1} r_i 2^{-(n-i)} \tag{6}$$

となる．式 (4)，(5)，(6) を用いて式 (3) を置き換えると，

$$\begin{aligned} \text{Eq. (3)} = &\left(-p_{n-1}2^{n-1} + \sum_{j=0}^{n-2} p_j 2^j\right)\left(\sum_{i=0}^{n-1} r_i 2^{-(n-i)}\right) \\ &+ \left(-q_{n-1}2^{n-1} + \sum_{j=0}^{n-2} q_j 2^j\right)\left(1 - \sum_{i=0}^{n-1} r_i 2^{-(n-i)}\right) \end{aligned} \tag{7}$$

Vol.2012-SLDM-155 No.7
Vol.2012-EMB-24 No.7
2012/3/2

となる．

式 (7) を展開すると，

$$\text{Eq. (7)} = p_{n-1}2^{n-1}\left(\underline{-\sum_{i=0}^{n-1} r_i 2^{-(n-i)}}\right) + \sum_{i=0}^{n-1}\sum_{j=0}^{n-2} p_j r_i 2^{-(n-i-j)} - q_{n-1}2^{n-1} + q_{n-1}\sum_{i=0}^{n-1} r_i 2^{i-1} + \sum_{j=0}^{n-2} q_j 2^j + \sum_{j=0}^{n-2} q_j 2^j \left(\underline{-\sum_{i=0}^{n-1} r_i 2^{-(n-i)}}\right) \tag{8}$$

となる．ここで，2 の補数を用いて $-r$ を表現すると，

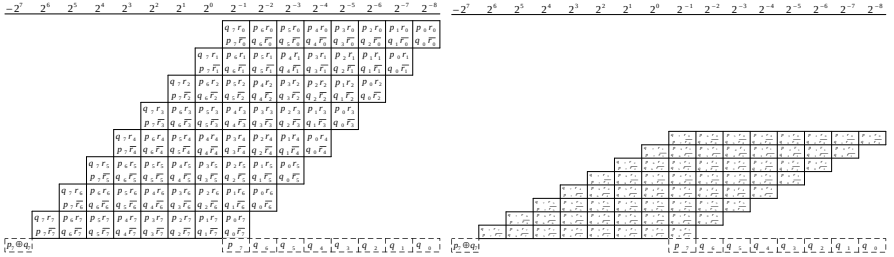
$$-r = -\left(-0\cdot 2^0 + \sum_{i=0}^{n-1} r_i 2^{-(n-i)}\right) = -1 + \sum_{i=0}^{n-1} \overline{r}_i 2^{-(n-i)} + 2^{-n} \tag{9}$$

と表せる．式 (8) の下線部の負項を式 (9) で置き換えて整理すると，

$$\begin{aligned}\text{Eq. (8)} &= p_{n-1}2^{n-1}\left(-1 + \sum_{i=0}^{n-1} \overline{r}_i 2^{-(n-i)} + 2^{-n}\right) + \sum_{i=0}^{n-1}\sum_{j=0}^{n-2} p_j r_i 2^{-(n-i-j)} - q_{n-1}2^{n-1} \\ &\quad + q_{n-1}\sum_{i=0}^{n-1} r_i 2^{i-1} + \sum_{j=0}^{n-2} q_j 2^j + \sum_{j=0}^{n-2} q_j 2^j \left(-1 + \sum_{i=0}^{n-1} \overline{r}_i 2^{-(n-i)} + 2^{-n}\right) \\ &= -\left(p_{n-1} + q_{n-1}\right)2^{n-1} + p_{n-1}2^{-1} + \sum_{i=0}^{n-1}\left(q_{n-1}r_i + p_{n-1}\overline{r_i}\right)2^{i-1} \\ &\quad + \sum_{i=0}^{n-1}\sum_{j=0}^{n-2}\left(p_j r_i + q_j \overline{r_i}\right)2^{-(n-i-j)} + \sum_{j=0}^{n-2} q_j 2^{-(n-j)}\end{aligned} \tag{10}$$

となる．式 (10) の符号部 $(-(p_{n-1}+q_{n-1})2^{n-1})$ は，$p_{n-1} = q_{n-1} = 1$ のときは式の形より下位の桁から必ず桁上げ入力があることがわかるため，$(-(p_{n-1} \oplus q_{n-1})2^{n-1})$ と変形できる．式 (10) を整理すると線形補間の演算式は，

$$\text{Eq. (10)} = -\left(p_{n-1} \oplus q_{n-1}\right)2^{n-1} + p_{n-1}2^{-1} + \sum_{i=0}^{n-1}\left(\underbrace{q_{n-1}r_i + p_{n-1}\overline{r_i}}\right)2^{i-1} + \sum_{i=0}^{n-1}\sum_{j=0}^{n-2}\left(\underbrace{p_j r_i + q_j \overline{r_i}}\right)2^{-(n-i-j)} + \sum_{j=0}^{n-2} q_j 2^{-(n-j)} \tag{11}$$



(a) セレクタ論理を用いない場合．　(b) セレクタ論理に帰着させた場合．

図 2　$n = 8$ のときの線形補間演算の部分積．

と表現できる．

さてここで式 (11) の波線部に着目しよう．すると式 (11) はちょうど式 (1) と同じ形式を持っていることに気付く．よって，この部分をセレクタ論理に帰着させることができる．セレクタ論理をあらかじめ計算しておくと部分積数が半分になる．線形補間演算器をセレクタ論理に帰着させることで部分積加算での桁上げ伝搬遅延が削減でき，演算が高速となる．

$n = 8$ のときの線形補間演算の部分積を図 2 に示す．図 2(a) では，実線で囲んだ部分に 2 個ずつ部分積が含まれている．この 2 個ずつの部分積を，図 2(b) ではセレクタ論理に帰着させることで 1 個ずつにまとめている．図 2 より，セレクタ論理に帰着させることで，$n = 8$ のときの線形補間演算の部分積は 137 個から 73 個に削減できるとわかる．

3.2　キュービックスプライン補間演算器

本節では，3.1 節で設計したセレクタ論理帰着型線形補間演算器を利用して，高速なキュービックスプライン補間演算器を設計する．キュービックスプライン補間では周囲 4 点の信号 f_i（$i = -1, 0, 1, 2$）を入力とする[5]．ただし，f_i は n ビットの符号なし整数とする．補間するのは f_0 と f_1 の間の区間であり，補間する未知データ $f(x)(0 < x < 1)$ は 3 次関数によって

$$f(x) = \underline{\left[\underline{\{ax + b(1-x)\}}x + c(1-x)\right]}x + d(1-x) \tag{12}$$

と表現できる．a, b, c, d は後述する式 (19) によって表される．式 (12) の 3 つの下線部はそれぞれ線形補間の形となる．よって，キュービックスプライン補間演算器は線形補間演算器を 3 段重ねにすることで設計でき，キュービックスプライン補間演算器をセレクタ論理に帰着させるためには，セレクタ論理帰着型線形補間演算器を用いて設計すればよい．

Vol.2012-SLDM-155 No.7
Vol.2012-EMB-24 No.7
2012/3/2

図 3 にキュービックスプライン補間演算器の構成を示す．キュービックスプライン補間演算器は，図 1(b) で示した線形補間演算器を 3 つ利用して構成している．図 3 では，後述する式 (19) により，4 つの入力信号から b と c を計算して 1 段目と 2 段目の線形補間演算器にそれぞれ入力している．

1 段目から 3 段目までの線形補間演算器の出力をそれぞれ s, t, u とおくと，線形補間の演算式を表す式 (3) より，

$$s = ax + b(1-x) \tag{13}$$
$$t = sx + c(1-x) \tag{14}$$
$$u = tx + d(1-x) \tag{15}$$

となる．1 段目の線形補間演算器の出力である式 (13) は，式 (11) より，

$$\begin{aligned}\text{Eq. (13)} = &-(a_{n-1} \oplus b_{n-1})\, 2^{n-1} + a_{n-1}2^{-1} + \sum_{i=0}^{n-1} \left(\underbrace{b_{n-1}x_i + a_{n-1}\bar{x_i}} \right) 2^{i-1} \\ &+ \sum_{i=0}^{n-1}\sum_{j=0}^{n-2} \left(\underbrace{a_j x_i + b_j \bar{x_i}} \right) 2^{-(n-i-j)} + \sum_{j=0}^{n-2} b_j 2^{-(n-j)}\end{aligned} \tag{16}$$

となり，式 (16) の波線部がセレクタ論理に帰着できる．

2 段目の線形補間演算器の出力である式 (14) は，式 (11) より，

$$\begin{aligned}\text{Eq. (14)} = &-(s_{n-1} \oplus c_{n-1})\, 2^{n-1} + s_{n-1}2^{-1} + \sum_{i=0}^{n-1} \left(\underbrace{c_{n-1}x_i + s_{n-1}\bar{x_i}} \right) 2^{i-1} \\ &+ \sum_{i=0}^{n-1}\sum_{j=0}^{n-2} \left(\underbrace{s_j x_i + c_j \bar{x_i}} \right) 2^{-(n-i-j)} + \sum_{j=0}^{n-2} c_j 2^{-(n-j)}\end{aligned} \tag{17}$$

となり，式 (17) の波線部がセレクタ論理に帰着できる．

3 段目の線形補間演算器の出力である式 (15) は，式 (11) より，

$$\begin{aligned}\text{Eq. (15)} = &-(t_{n-1} \oplus d_{n-1})\, 2^{n-1} + t_{n-1}2^{-1} + \sum_{i=0}^{n-1} \left(\underbrace{d_{n-1}x_i + t_{n-1}\bar{x_i}} \right) 2^{i-1} \\ &+ \sum_{i=0}^{n-1}\sum_{j=0}^{n-2} \left(\underbrace{t_j x_i + d_j \bar{x_i}} \right) 2^{-(n-i-j)} + \sum_{j=0}^{n-2} d_j 2^{-(n-j)}\end{aligned} \tag{18}$$

となり，式 (18) の波線部がセレクタ論理に帰着できる．

以上のように，キュービックスプライン補間演算器は，内部で利用する 3 つの線形補間演算器それぞれに対して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，演算を高速化できる．

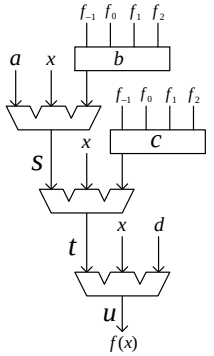


図 3 キュービックスプライン補間演算器.

なお，式 (12) でのキュービックスプライン補間の係数 a，b，c，d は，

$$\begin{aligned} a &= f_1 \\ b &= \frac{1}{5}(3f_{-1} - 7f_0 + 12f_1 - 3f_2) \\ c &= \frac{1}{5}(-4f_{-1} + 6f_0 + 4f_1 - f_2) \\ d &= f_0 \end{aligned} \tag{19}$$

と求めることができる[5]．

n ビットの符号なし整数を入力すると，補間係数 b, c の範囲は

$$\begin{aligned} -2(2^n - 1) \le b \le 3(2^n - 1) \\ -(2^n - 1) \le c \le 2(2^n - 1) \end{aligned} \tag{20}$$

となる．式 (20) を満たす b, c を表現するには $n+3$ ビット必要である．よって，n ビットのキュービックスプライン補間演算器を設計するには式 (16)〜(18) を $n+3$ ビットとした線形補間演算器を用いる．

4. 評 価 実 験

本章では，線形補間演算器とキュービックスプライン補間演算器を実装した結果を示す．各演算器の設計には Verilog HDL を使用し，Synopsys 社の Design Compiler のトポグラ

Vol.2012-SLDM-155 No.7
Vol.2012-EMB-24 No.7
2012/3/2

フィカルモードを用いて論理合成を行った．また，面積の算出についても同ソフトウェアを使用した．解析には STARC ⋆1(CMOS 90 nm) のセルライブラリを用いた．

以下の 2 種類の手法と提案手法について比較した．

手法 1（算術演算子を用いる場合）： 加算（+），減算 (-)，乗算（*）といった Verilog HDL の算術演算子を用い，Design Compiler で論理合成する．従来手法として，提案手法との比較対象とする．

手法 2（セレクタ論理を用いない場合）： セレクタ論理を用いずに部分積を生成し，Design Compiler で加算する．提案手法での純粋なセレクタ論理の効果を図るための比較対象として使用する．

提案手法： セレクタ論理を用いて部分積を生成し，Design Compiler で加算する．

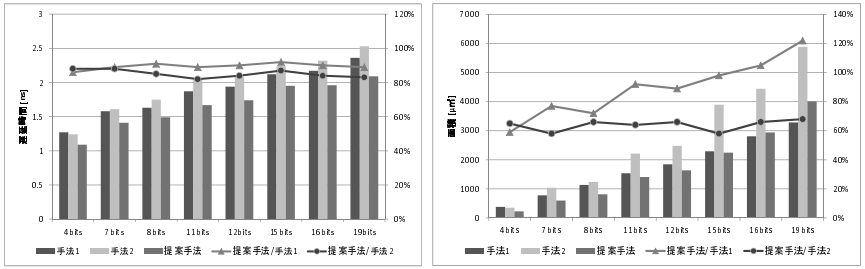
4.1 線形補間演算器

本節では，線形補間演算器の評価実験結果を示す．設計の際のビット長は 4 ビットから 16 ビットまでの 4 ビット刻みとする．さらに，これらのビット長に 3 を足したビット長の演算器も実装する．これは，n ビットのキュービックスプライン補間演算器を実装する際に，$n+3$ ビットの線形補間演算器を使用するためである．

図 4 に線形補間演算器の論理合成結果を示す．図 4(a) には遅延時間のグラフを，(b) には面積のグラフを示す．図 4 の折れ線グラフは，手法 1 に対する提案手法の割合と手法 2 に対する提案手法の割合を表す．

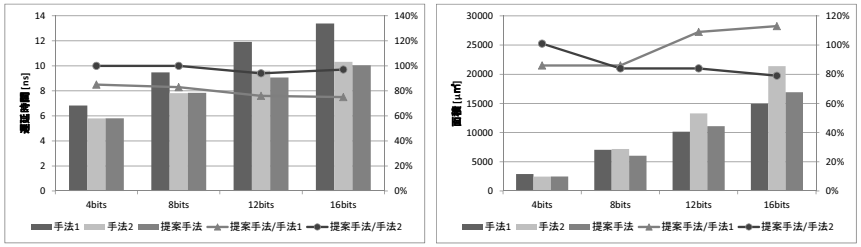
図 4(a) より，提案手法の遅延時間はいずれのビット長でも最小となることがわかる．提案手法では，手法 1 と比較して遅延時間が平均 10 %，最大で 14 %削減できた．また，手法 2 と比較して遅延時間が平均 15 %，最大で 18 %削減できた．図 4(b) より，提案手法は手法 1 と比較した場合，15 ビット以下では面積が削減されたが，それ以上のビット長では手法 1 の方が面積が小さくなった．これは，回路規模が大きくなるほど，Design Compiler による面積最適化が作用しやすくなるためであると考えられる．図 4(b) より，手法 2 と比較した場合には，いずれのビット長でも提案手法の方が面積が小さく，平均 36 %，最大で 42 %，提案手法の面積が削減できた．以上より，線形補間演算器をセレクタ論理に帰着させて設計することは遅延時間の削減に有効であり，ビット長が短いときには面積の削減にも有効であるといえる．

⋆1 STARC[90 nm] 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し，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) 線形補間演算器の遅延時間．　(b) 線形補間演算器の面積．

図 4 線形補間演算器の論理合成結果．



(a) キュービックスプライン補間演算器の遅延時間．　(b) キュービックスプライン補間演算器の面積．

図 5 キュービックスプライン補間演算器の論理合成結果．

4.2 キュービックスプライン補間演算器

本節では，線形補間演算器の評価実験結果を示す．設計の際のビット長は 4 ビットから 16 ビットまでの 4 ビット刻みとする．

面積制約を与えない場合

図 5 にキュービックスプライン補間演算器の論理合成結果を示す．図 5(a) には遅延時間のグラフを，(b) には面積のグラフを示す．図 5 での折れ線グラフは，手法 1 に対する提案手法の割合と手法 2 に対する提案手法の割合を示す．

図 5(a) より，手法 1 と比較すると，提案手法では平均 20 %，最大 25 %遅延時間が削減できた．図 5(b) より，4 ビットと 8 ビットの場合は手法 1 に対して提案手法の面積が 14 %削

Vol.2012-SLDM-155 No.7
Vol.2012-EMB-24 No.7
2012/3/2

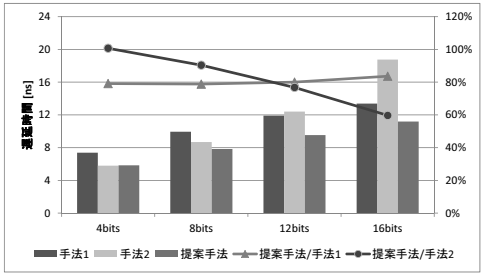


図 6 面積制約を与えたキュービックスプライン補間演算器の遅延時間.

減されたが，12 ビットと 16 ビットの場合は手法 1 の方が面積が小さくなる．これは，回路規模が大きくなるほど，Design Compiler による面積最適化が作用しやすくなるためであると考えられる．

図 5(a) より，手法 2 と提案手法を比較すると，4 ビットと 8 ビットの場合は両者の遅延時間がほぼ等しく，12 ビットと 16 ビットの場合はそれぞれ 6 %，3 %の遅延時間の削減にとどまった．図 5(b) より，8 ビット以上での提案手法の面積は手法 2 と比較して約 20 %削減された．よって，手法 2 では面積を大きくすることで遅延時間を削減していると考えられる．そこで，次に面積制約を与えた場合の遅延時間を比較する．

面積制約を与えた場合

図 6 に面積制約を与えた上でのキュービックスプライン補間演算器の論理合成結果を示す．なお，図 5 の 3 つの手法の中で各ビット長ごとに最小の面積を基準とし，基準面積以下となるよう面積制約を与えた．

図 6 より，手法 1 に対して提案手法では平均 19 %，最大 21 %遅延時間が削減できた．図 6 より，手法 2 と比較した提案手法の遅延時間は，4 ビットのときはほぼ同じだが，8 ビット以上では提案手法では平均 24 %，最大 40 %遅延時間が削減できた．以上より，キュービックスプライン補間演算器をセレクタ論理に帰着させた設計は，遅延時間の削減に有効であるといえる．

5. おわりに

本稿では，セレクタ論理を利用して高速な補間演算器を設計する手法を提案した．線形補間演算器は，演算式をビットレベル式変形するとセレクタ論理の演算式を含む形になるため，セレクタ論理を組み込むことができる．キュービックスプライン補間演算器をセレクタ論理に帰着させるには，セレクタ論理帰着型線形補間演算器を 3 段に重ねればよい．

線形補間では，算術演算子を用いて設計した従来の線形補間演算器と比較して，遅延時間を最大 15 %削減できた．キュービックスプライン補間演算では，面積制約を与えない場合，従来のキュービックスプライン補間演算器と比較して，遅延時間を最大 25 %削減できた．一方，面積制約を与えて純粋なセレクタ論理の効果を検討した場合，8 ビット以上のキュービックスプライン補間演算器ではセレクタ論理に帰着させることで平均 24 %遅延時間を削減できた．

今後の課題として，セレクタ論理に帰着させた補間演算器を用いて画像処理を行うシステム全体を構築し，検証を行うことがあげられる．

謝　辞

本研究の一部は，科学研究費補助金 (課題番号 22300019) ならびに放送文化基金助成による．

参 考 文 献

1) A. Asharafi, R. Adhami and A. Milenković, “A direct digital frequency synthesizer based on the quasi-linear interpolation method,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, pp. 863–872, 2010.
2) A. Belahmidi and F. Guichard, “A partial differential equation approach to image zoom,” in *Proc. ICIP*, pp.649–652, 2004.
3) D. Barrera, A. Guessab, M. J. Ibáñez and O. Nouisser, “Optimal bivariate C1 cubic quasi-interpolation on a type-2 triangulation,” *J. Comput. and Appl.Math.*, vol. 234, pp. 1188–1199, 2010.
4) 外村元伸, 実設計に応用できる演算回路スキルを身につけよう, *Design Wave Magazine*, pp. 39–48, 2006.
5) 外村元伸, キュービック・スプライン補間演算器の設計, *Design Wave Magazine*, pp. 120–126, 2006.
6) 沼田宗敏, 野村俊, 神谷和秀, 輿水大和, 田代発造, “フーリエ変換を用いた B スプライン曲線補間による CT 画像の鮮鋭化,” 情報処理学会論文誌, vol. 46, no. 10, pp. 2546–2555, 2005.
7) 森隆寛, 外村元伸, 大住勇治, 後藤敏, 池永剛, “キュービック補間に基づく魚眼画像の高画質補正アルゴリズムおよび専用ハードウェアエンジンの提案,” 画像電子学会誌, vol. 36, no. 5, pp. 680–687, 2007.