CC-004

Shift-and-Add 法を用いた対数変換回路の高精度化
Improving the Accuracy of Logarithmic Conversion Circuits Using the Shift-and-Add Method

比嘉 駿† 田中 勇樹†† 魏 書剛‡
Shun Higa Yuuki Tanaka Shugang Wei

1. はじめに

デジタル情報処理は様々な演算から構成されている．その中で，乗算，除算，べき乗，平方根などの演算は複雑であり，回路面積や遅延時間等のコストがかかりやすいものになっている．そこに対数演算を用いることで乗算を加算に，除算を減算に，累乗は乗算，平方根は除算というように演算を簡略化することができる．近年では，対数変換の手法として入力データの上位数ビットを数回右シフトした値を元の値に足し合わせることによって，回路面積と変換精度のトレードオフを実現した Shift-and-Add 法が提案されている[1]．しかし，この方法は入力の上位数ビットにのみにしか演算を行わないため十分な変換精度が得られていなかった．

本研究では，底を２とする対数を求める変換回路について，回路面積と遅延時間の増大を抑えつつ変換精度の向上を図ることを目的とする．

2. 2 進数における対数変換

2 進数における対数変換では，数Nを$Z \in (0,1)$とするとき z_u~z_0を整数部，z_{-1}~z_{-v}を小数部として以下のように表して考える．

$$N = z_u \ z_{u-1} \cdots z_0 \, . z_{-1} \cdots z_{-v} \tag{1}$$

このとき，Nの最上位非ゼロビットをz_kとすると，Nは以下のように表せる．

$$N = 2^k(1+m) \tag{2}$$

ここで，仮数mは，$0 \le m < 1$である．これにより，底を２とし，真数をNとする対数は，

$$\log_2 N = k + \log_2(1+m) \tag{3}$$

と表すことができ，Nの対数は$\log_2(1+m)$の値を求めることで得られる．また，本論ではnビットの仮数mを$m = \{m_{n-1}\ m_{n-2} \ldots m_1\ m_0\}$とし，その上位下位$t$ビットをそれぞれ$m_{tMSBits}$，$m_{tLSBits}$と表す．

3. 先行研究

3.1 Mitchell の方法[2]

Mitchell は$\log_2(1+m)$の値を**図 1** で示されるような直線で近似し，仮数mの値をそのまま用いた．そして，式(3)に示すようにkにmを加えることで$\log_2 N$の近似値が得られる．変換関数を以下に示す．

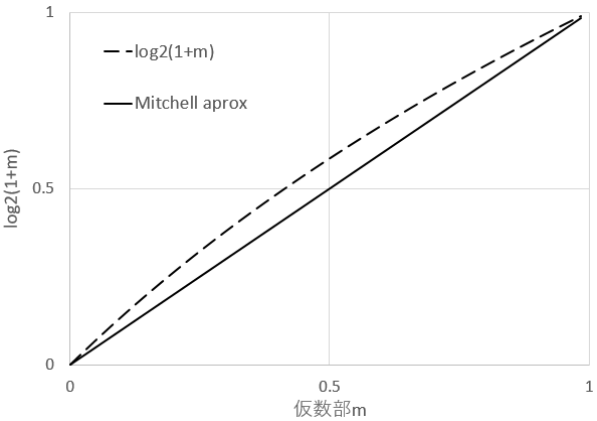

図 1　Mitchell の方法

$$\log_2(1+m) \cong m \tag{4}$$

3.2 Paul の方法[3]

Paul は Mitchell の方法で近似を行った際に生じる誤差を，LUT(Look Up Table)と線形補間を組み合わせることによって変換誤差を低減する方法を提案した．Mitchell の方法で近似した際に生じる誤差E_Mは

$$E_M = \log_2(1+m) - m \tag{5}$$

であり，この式で表される誤差曲線を2^t点でサンプリングし変換ビット数に応じたワード長で LUT の一つ目のエントリとして格納する．このとき，LUT のアドレスは入力される仮数mの上位tビットである．次にテーブルに格納された値の間の値を考えると，対数の近似値$\log_2(1+m)'$は以下のように表される．

$$\log_2(1+m)' = m + a + \frac{(b-a) \times n_1}{2^{k-t}} \tag{6}$$

ここで，mは対数変換前の入力値の仮数部，aはその仮数部の上位tビットで LUT にアクセスされるテーブルに格納されるE_Mの値，bはaに隣接する次のテーブルに格納された値である．また，kは Mitchell の変換による誤差を補間する際に使用される仮数mのビット数，n_1は仮数mのkビットのうち LUT のアドレスとして使用されなかった$k-t$ビットを整数値として扱う値である．

† 群馬大学 大学院理工学府 知能機械創製理工学教育プログラム　Education program of mechanical science and technology, graduate school of science and technology, Gunma University
†† 群馬大学 大学院理工学府 電子情報部門　Division of electronics and informatics, mathematics and physics, graduate school of science and technology, Gunma University
‡ 群馬大学 大学院理工学府 知能機械創製部門　Division of mechanical science and technology, graduate school of science and technology, Gunma University

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

Paul は式(6)の計算を行う際に第 3 項の分子の計算を対数法則を用いており，一つ目の LUT にaと$\log_2(b-a)$の値を格納し，n_1を Mitchell の方法で対数近似を行っており，回路上では加算として実現している．また対数から真数を求める際には Mitchell の方法より

$$2^x = (1+x) \tag{7}$$

と近似し，2 つ目の LUT にその変換時の誤差となる$2^x-(1+x)$を格納し，E_Mと同様にサンプリングし格納して真数を求めている．

以上の方法で式(6)に示されるような線形補間を実現している．

3.3 Sangregory の方法[4]

Sangregory らは，Mitchell の方法より正確な対数変換を行う改良型の変換回路を提案した．Mitchell の近似において$\log_2(1+m)$の誤差が最大になる$m=0.5$付近では，小数部の 4 ビット目(2^{-4})からの誤差が発生するため，それ以降のビットに修正を行う必要がある[4]．そこで，$0 \le m < 0.5$では直線の傾きを大きく，$0.5 \le m < 1$では小さくなるように補正することで誤差を低減している．具体的には，$0 \le m < 0.5$ではmの値を右に 2 回シフトしたものを加算し，$0.5 \le m < 1$ではmの全ビットを反転したものを加算している．提案されている変換関数を式(8)に示す．

$$\log_2(1+m)' = \begin{cases} m + 2^{-2}m_{4MSBits} & 0 \le m < 0.5 \\ m + 2^{-2}\overline{m}_{4MSBits} & 0.5 \le m < 1 \end{cases} \tag{8}$$

3.4 Abed の方法[5]

Abed の 3 領域法では近似領域を 3 つの不等間隔領域に分割し，仮数部mの 4 つの MSB を用いることで，先に紹介した Sangregory らが提案していた 2 領域での近似と比べて誤差の少ない補正を実現している．その変換関数を式(9)に示す．

$$\log_2(1+m)' = \begin{cases} m + 2^{-2}m_{4MSBits} & 0 \le m < 0.25 \\ m + (2^{-4}+2^{-6}) & 0.25 \le m < 0.75 \\ m + 2^{-2}\overline{m}_{4MSBits} & 0.75 \le m < 1 \end{cases} \tag{9}$$

また，Abed らはこの方法の近似領域を 6 領域に拡張した方法[5]も提案しており，より高精度の近似を実現している．

3.5 Juang の方法[1]

Juang が提案した変換法[1]は，各領域の近似直線の傾きをより対数曲線に近づけたものである．これまでに紹介した方法では，近似直線の傾きは 2 のべき乗でのみで構成されていたが，Juang は 2 の指数を複数組み合わせることでより正確に傾きを再現している．その変換関数を式(10)に示す．

$$\log_2(1+m)' = \begin{cases} m + (2^{-3}+2^{-4})m_{4MSBits} & 0 \le m < 0.5 \\ m - (2^{-3}+2^{-5})m_{4MSBits} + (2^{-3}+2^{-5}) & 0.5 \le m < 1 \end{cases} \tag{10}$$

ここまで紹介してきた変換方法では，入力の仮数mの上位数ビットに対してのみシフト加算を行っているため，区間ごとに傾きが 1 の直線の切片のみを変えたような出力になり，変換誤差が拡大しやすいものであった．

4. 提案する変換方法

今回提案する方法では，式(11)で示すように仮数の下位tビットに対して適切なシフト量を用いてシフト加算することによって近似直線の傾きSを再現し，上位ビットを LUT のアドレスとして格納した近似直線の切片Iを加えることで対数変換を行う．

$$\log_2(1+m)' = I + S \times m_{tLSBits} \tag{11}$$

4.1 8 ビットの変換関数

8 ビットの対数変換では，仮数mの上位 3 ビットを用いて 6 つの領域に区分し，下位 4 ビットに対してシフト加算を行い，上位 4 ビットを LUT のアドレスとして各区間内での切片をワード長 8 ビットで格納している．式(12)に変換関数を示す．

$$\log_2(1+m)' = \begin{cases} I + (1+2^{-2}+2^{-3})m_{4LSBits} & 0 \le m < 0.125 \\ I + (1+2^{-2})m_{4LSBits} & 0.125 \le m < 0.25 \\ I + (1+2^{-3})m_{4LSBits} & 0.25 \le m < 0.375 \\ I + m_{4LSBits} & 0.375 \le m < 0.5 \\ I + (1-2^{-3})m_{4LSBits} & 0.5 \le m < 0.75 \\ I + (1-2^{-2})m_{4LSBits} & 0.75 \le m < 1 \end{cases} \tag{12}$$

ここで，Iは仮数mの上位 4 ビットをアドレスとして LUT に 8 ビットで格納された近似直線の切片である．

4.1.1 シフト量の決定

シフト量は，分割した領域ごとに$\log_2(1+m)$の傾きを最小二乗法で求め，2 のべき乗の数の組み合わせで表現できる範囲で求めた数値に近似する．例えば，$\log(1+m)$の第 2 領域$(0.125 \le m < 0.25)$では傾きを最小二乗法で求めると，2 進数では$(1.0011011\cdots)_2$ と表され，これを以下のように近似する．

$$1.0011011_2 \cong 1 + 2^{-2} \tag{13}$$

同様に，第 5 領域$(0.5 \le m < 0.75)$では

$$0.1110001_2 \cong 1 - 2^{-3} \tag{14}$$

としている．

4.1.2 LUT に格納する値の決定

LUT に格納する値は前節で決定した近似直線の切片になる．近似直線の傾きは最小二乗法で求めたが，今回提案した手法では一つ（1 領域）の傾きに対して複数の切片を用意する必要があり，これは LUT のテーブル数に比例して増加するため，後々ビット数を拡張することを考えると一つひとつを最小二乗法で決定するのは現実的では

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

ない．よって，以下の２ステップにより格納する値を決定する．

ステップ１

一時的な格納値として，LUT のアドレスとして用いる仮数mの上位 4 ビットで表される値を真数とする対数の真値を格納する．具体的には，一時的な格納値をI'，仮数mの上位 4 ビットで表される値を$m_{4MSBits}$と置くと式(15)で書き表せる．

$$I' = \log_2(1 + m_{4MSBits}) \tag{15}$$

そして，ここで格納した値で誤差の調査を行う．

ステップ２

ステップ１で生じる誤差を調べると，LUT の１つのテーブルの範囲内での最大誤差はそのテーブルの最後値で発生する．よって，その値の半分の値をステップ１で定めた値から差し引くことで最大誤差を抑えることができる．ただし，その最大誤差の値が最終桁の重みの２倍(2^{-7})未満のとき誤差が拡大する可能性がある．そのため最大誤差の絶対値が$2ulp$以上の場合のみこの処理を行う．

具体的には，最終的に LUT に格納するする値Iは，LUT の１つのテーブルの範囲内での最大誤差をE_{max}とすると以下の式で表される．

$$I = \begin{cases} I' - \dfrac{E_{max}}{2} & E_{max} \geq 2^{-7} \\ I' & E_{max} < 2^{-7} \end{cases} \tag{16}$$

4.2 16 ビットの変換関数

16 ビットの対数変換では，仮数mの上位４ビットを 12 個の領域に区分し，下位 8 ビットに対してシフト加算を行い，上位 8 ビットを LUT のアドレスとして各区間内での切片をワード長 16 ビットで格納している．変換関数を式(17)に示す．

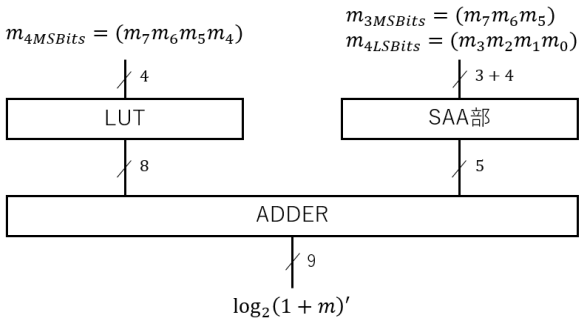

図２　変換ブロック図（8 ビット変換回路）

ここで，Iは仮数mの上位 8 ビットをアドレスとして LUT に 16 ビットで格納された近似直線の切片である．また，シフト量と LUT への格納値は 8 ビットの際と同じ方法で決定する．

5. 回路実装

5.1 8 ビット変換回路

前節で述べた 8 ビットの変換関数を実装する回路を**図 2**，**図 3** に示す．式(12)の第１項は LUT から，第２項 Shift-and-Add(SAA)部から出力され，ADDER によってそれらを加算している．図 3 に示すように，式(12)中の$2^{-2}m_{4LSBits}$，$2^{-3}m_{4LSBits}$の項の有無と，それらの正負は仮数mの上位 3 ビットを用いて AND 素子と XOR 素子によって制御している．ただし，図中では「selector」と表して簡略している．selector の真理値表を**表 1** に示す．また HA，FA はそれぞれ半加算器，全加算器であり，それぞれの入力の一つは右隣の桁上げ出力と接続されている．左端の XOR は仮数mの最上位ビットによって SAA 部から出力される最上位ビットへの桁上げ，桁あふれを制御し近似領域に応じた数値の出力を可能にしている．

$$\log_2(1+m)' = \begin{cases} I + \left(1 + 2^{-2} + 2^{-3} + 2^{-5}\right) m_{8LSBits} & 0 \leq m < 0.0625 \\ I + \left(1 + 2^{-2} - 2^{-4} + 2^{-6}\right) m_{8LSBits} & 0.0625 \leq m < 0.125 \\ I + \left(1 + 2^{-2}\right) m_{8LSBits} & 0.125 \leq m < 0.1875 \\ I + \left(1 + 2^{-3} + 2^{-4}\right) m_{8LSBits} & 0.1875 \leq m < 0.25 \\ I + \left(1 + 2^{-3}\right) m_{8LSBits} & 0.25 \leq m < 0.3125 \\ I + \left(1 + 2^{-4} + 2^{-6}\right) m_{8LSBits} & 0.3125 \leq m < 0.375 \\ I + \left(1 + 2^{-5}\right) m_{8LSBits} & 0.375 \leq m < 0.4375 \\ I + \left(1 - 2^{-5} + 2^{-6}\right) m_{8LSBits} & 0.4375 \leq m < 0.5 \\ I + \left(1 - 2^{-3} + 2^{-5} + 2^{-6}\right) m_{8LSBits} & 0.5 \leq m < 0.625 \\ I + \left(1 - 2^{-2} + 2^{-4} + 2^{-5} + 2^{-6}\right) m_{8LSBits} & 0.625 \leq m < 0.75 \\ I + \left(1 - 2^{-2} + 2^{-5} + 2^{-6}\right) m_{8LSBits} & 0.75 \leq m < 0.875 \\ I + \left(1 - 2^{-2}\right) m_{8LSBits} & 0.875 \leq m < 1 \end{cases} \tag{17}$$

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

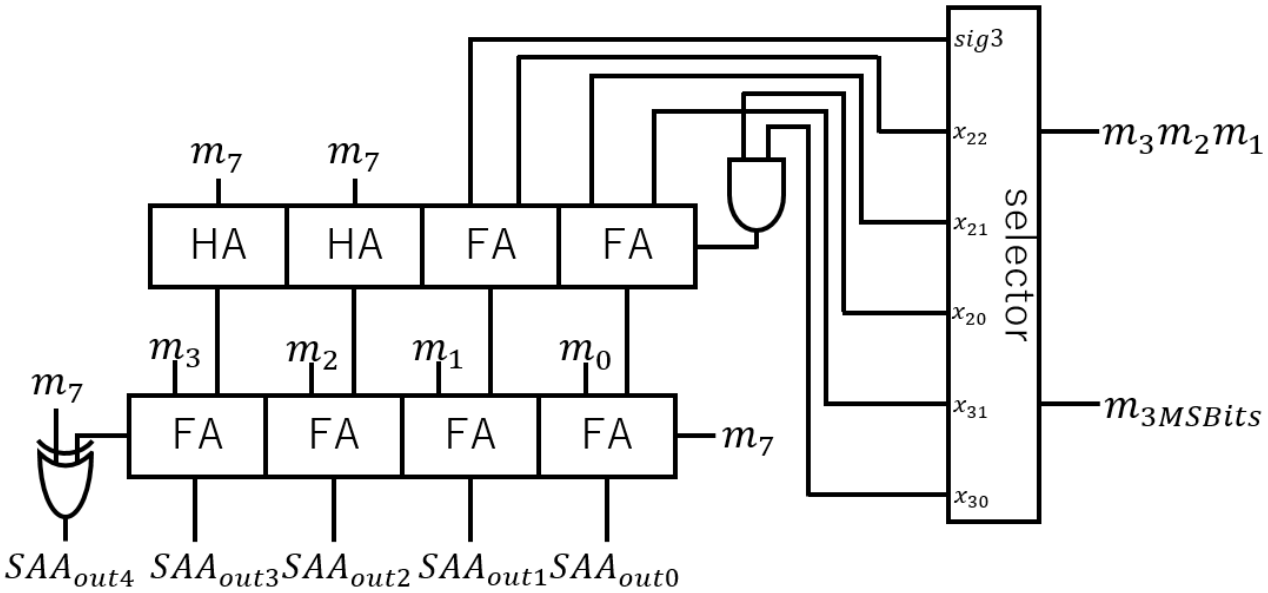

図 3　SAA 部（8 ビット変換回路）

表 1　SAA 部（8 ビット）selector 真理値表

$m_{3MSBits}$	x_{22}	x_{21}	x_{20}	x_{31}	x_{30}	sig3
000	m_3	m_2	m_1	m_3	m_2	0
001	m_3	m_2	m_1	0	0	0
010	0	0	0	m_3	m_2	0
011	0	0	0	0	0	0
10 ∗	0	0	0	$\overline{m_3}$	$\overline{m_2}$	1
11 ∗	$\overline{m_3}$	$\overline{m_2}$	$\overline{m_1}$	0	0	0

5.2 16 ビット変換回路

16 ビットの変換関数を実装する回路を**図 4**，**図 5** に示す．16 ビットの場合は加算の項数が多いため，CSA（桁上げ保存加算器）を用い，入力する仮数mは SAA 部内ではなく ADDER によって加算を行っている．また，回路内で正しく減算が行えるように信号$sig5, sig35, sig235$により，重みが$2^{-2}, 2^{-3}, 2^{-5}$ の項の符号を CSA 内の上位ビットに入力している．8 ビットの変換回路と同様に各項の有無を制御する信号の生成はやや複雑になっているので，ここでは「selector」と表し簡略している．16 ビット変換回路の selector の真理値表を**表 2** に示す．

6. 回路評価

6.1 8 ビット変換回路で生じる誤差

8 ビットの変換回路の入力値に対する変換誤差を**図 6** に示す．最大誤差は$m = 0.04296875$のとき$0.00601 < 2^{-7}$であった．出力の 8 ビット目は、出力外である重みが2^{-9}以下の桁からの繰り上がりの有無によって数値がばらついてしまうが、今回の回路ではそれを考慮していないため得られた精度としては十分だと考えられる。

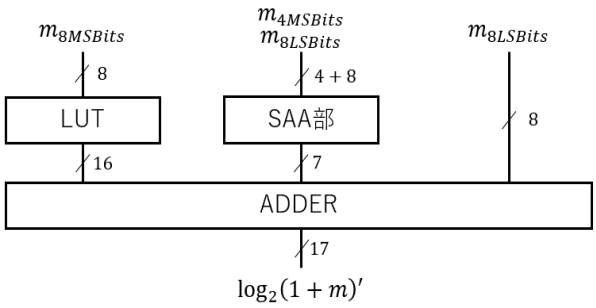

図 4　変換ブロック図（16 ビット変換回路）

6.2 16 ビット変換回路で生じる誤差

16 ビットの変換回路の入力値に対する変換誤差を**図 7** に示す．最大誤差は$m = 0.05867$のとき$0.0000997 < 2^{-13.2}$であった．今回提案した方法でこれ以上の変換精度を得るには，LUT のテーブル数あるいは近似領域数を増加させる必要がある．近似領域を増加させる場合，領域ごとの近似直線の傾きの変換がより細かなものになり，それに伴いより小さな$m_{6MSBits}$の係数が必要になる．今回設計した変換回路での最小の係数は2^{-6}であったが，仮に変換領域を今回の 2 倍の 24 領域とした場合、直線近似を行う際には2^{-7}以下の係数が必要となり SAA 部内での加算回数が増加する．さらに2^{-8}よりも小さな係数を用いると、右シフト後の値の最上位ビットの重みが2^{-17}となり，出力値への影響が小さい桁での加算が必要となるため，増加するハードウェアコストに見合った変換精度向上は得にくくなる考えられる．

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

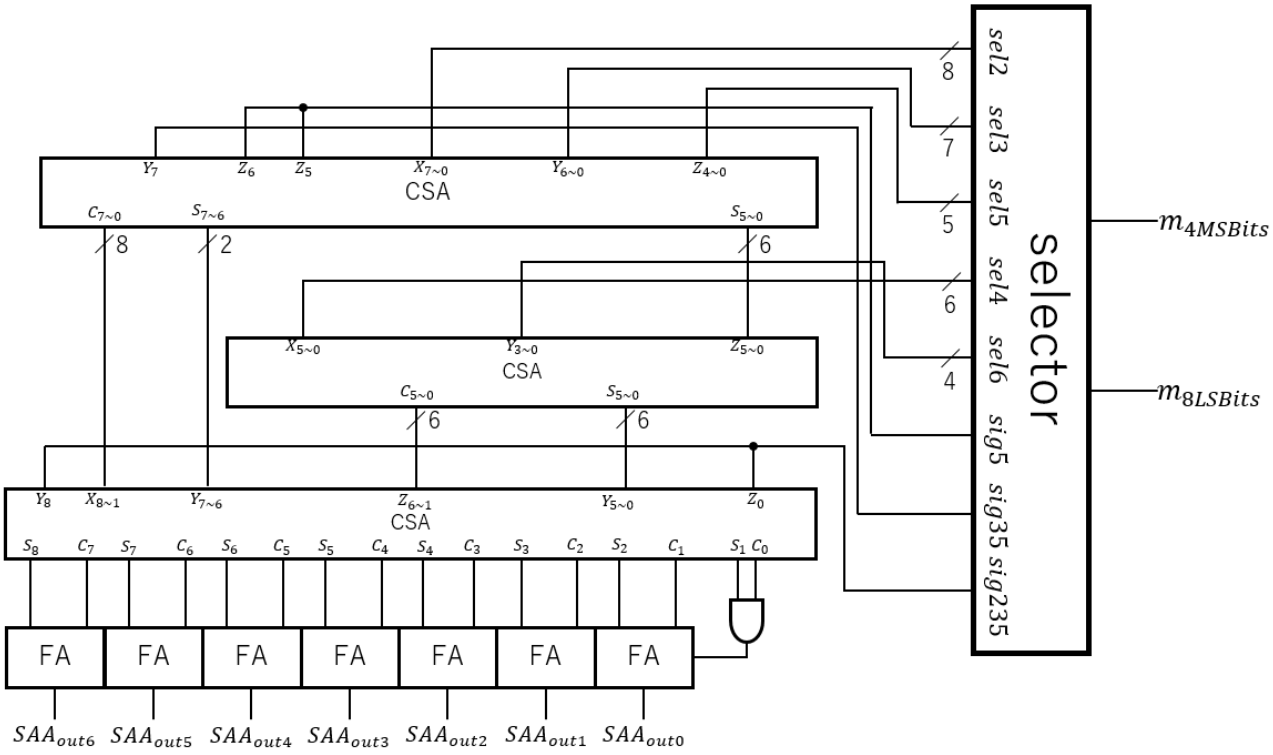

図 5　SAA 部（16 ビット変換回路）

表 2　SAA 部（16 ビット）selector 真理値表

$m_{4MSBits}$	sel2	sel3	sel4	sel5
0000	$m_7m_6m_5m_4m_3m_2m_1m_0$	$m_7m_6m_5m_4m_3m_2m_1$	000000	$m_7m_6m_5m_4m_3$
0001	$m_7m_6m_5m_4m_3m_2m_1m_0$	0000000	$m_7m_6m_5m_4m_3m_2$	00000
0010	$m_7m_6m_5m_4m_3m_2m_1m_0$	0000000	000000	00000
0011	00000000	$m_7m_6m_5m_4m_3m_2m_1$	$m_7m_6m_5m_4m_3m_2$	00000
0100	00000000	$m_7m_6m_5m_4m_3m_2m_1$	000000	00000
0101	00000000	0000000	$m_7m_6m_5m_4m_3m_2$	00000
0110	00000000	0000000	000000	$m_7m_6m_5m_4m_3$
0111	00000000	0000000	000000	$\overline{m_7m_6m_5m_4m_3}$
100 *	00000000	0000000	000000	$m_7m_6m_5m_4m_3$
101 *	$\overline{m_7m_6m_5m_4m_3m_2m_1m_0}$	$\overline{m_7m_6m_5m_4m_3m_2m_1}$	$m_7m_6m_5m_4m_3m_2$	$m_7m_6m_5m_4m_3$
110 *	$\overline{m_7m_6m_5m_4m_3m_2m_1m_0}$	0000000	000000	$m_7m_6m_5m_4m_3$
111 *	$\overline{m_7m_6m_5m_4m_3m_2m_1m_0}$	0000000	000000	00000

$m_{4MSBits}$	sel6	sig5	sig35	sig235
0000	0000	0	0	0
0001	$m_7m_6m_5m_4$	0	0	0
0010	0000	0	0	0
0011	0000	0	0	0
0100	0000	0	0	0
0101	$m_7m_6m_5m_4$	0	0	0
0110	0000	0	0	0
0111	$m_7m_6m_5m_4$	0	0	0
100 *	$m_7m_6m_5m_4$	1	1	1
101 *	$m_7m_6m_5m_4$	0	1	1
110 *	$m_7m_6m_5m_4$	0	0	1
111 *	0000	0	0	1

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

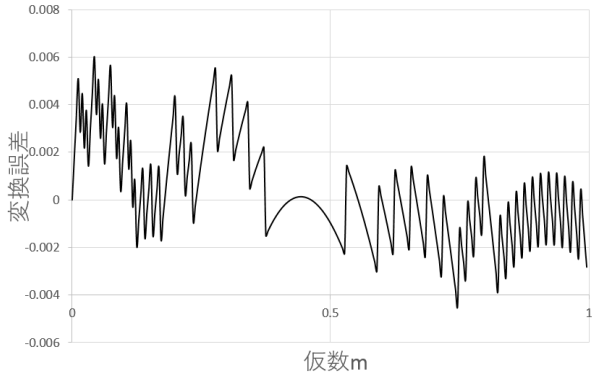

図 6　変換誤差（8 ビット）

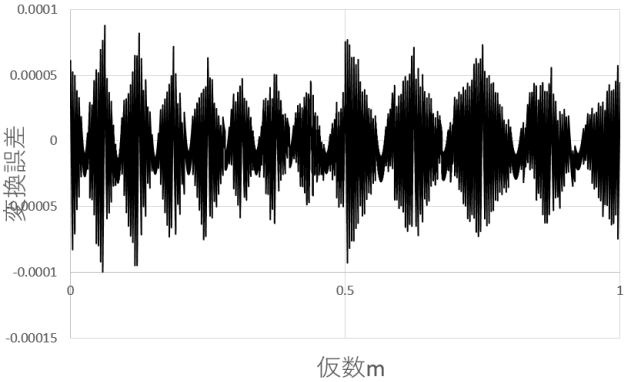

図 7　変換誤差（16 ビット）

7. 先行研究との比較

今回提案した回路と先行研究で示されていた回路の面積，遅延時間，最大誤差を**表 3** に示す．ここで示されている数値は，各対数変換回路を Synopsys 社製 DesignCompiler 上で，0.18μmCMOS ゲートアレイ設計ライブラリにより論理合成を行って得られたものである．

表 3　先行研究との比較

変換方法	面積〔μ㎡〕	遅延時間〔ns〕	最大誤差
Juang(9bit)	2183.73	8.9	0.0365683
Proposed(8bit)	2464.77	8.5	0.0060084
Paul(16bit)	28046.19	46.6	0.0000693
Proposed(16bit)	19187.30	19.5	0.0000997

提案した 8 ビットの変換方法と Juang の方法と比較すると，回路面積は約 15%増加し，遅延時間は約 16%縮小することができた．しかし 16 ビットの変換回路は，Paul の方法と比較すると，面積は 44%，遅延時間は約 58%縮小できているものの，変換誤差が約 44%増大しており，変換方法の見直しが必要になっていると考えられる．

8. 結言

今回は，従来の上位ビットにシフト加算を行う対数変換回路に対し，入力の下位ビットを演算対象とする対数変換回路を作成しそれぞれ比較した．入力が 8 ビットの場合は従来の回路と比べて面積や遅延時間をほとんど変えずに十分な変換精度を得ることができた．ただし仮数mを 16 ビットに拡張した場合では変換精度が不十分であった．今後の課題として，新たな 16 ビット以上への拡張方法，あるいは拡張可能な変換方法の模索が必要である．

謝辞

本研究は，東京大学 VDEC 活動を通して，日本シノプシス合同会社の協力で行われたものである．

参考文献

[1] Tso-Bing Juang, “A Lower Error and ROM-Free Logarithmic Converter for Digital Signal Processing Applications”, IEEE Trans. Circuits and Systems II: Express Briefs, Vol.56, pp.931–935, Dec. 2009.
[2] J. N. Mitchell, “Computer multiplication and division using binary logarithms”, IRE Trans. Electron. Comput., vol.11, no.11, pp.512–517, Aug. 1962.
[3] S. Paul, N. Jayakumar and S. P. Khatri, “A fast hardware approach for approximate efficient logarithm and antilogarithm computations”, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.17, no.2, pp.269–277, Feb. 2009.
[4] S. L. SanGregory, C. Brothers and D, R. E. Siferd. Gallagher, “A fast, lowpower logarithm approximation with CMOS VLSI implementation”, Proc. IEEE MWSCAS, vol.1, pp.388–391, Aug. 1999.
[5] K. H. Abed and R. E. Siferd, “CMOS VLSI implementation of a low-power logarithmic converter”, IEEE Trans. Comput., vol.52, no.11, pp.1421–1433, Nov. 2003.

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.