Vol.2017-ARC-225 No.20
Vol.2017-SLDM-179 No.20
Vol.2017-EMB-44 No.20
2017/3/9

Stochastic Computingにおける マルチプレクサの制御入力として 複雑な式を用いる回路設計

壷阪 幸輝[1,a)]　山下 茂[2,b)]

概要：Stochastic Computing において，任意の関数を実現する手法として，近似関数を利用した設計手法が存在する．そのため，近似誤差が生じ，計算精度の低下に繋がる．そこで，本稿では，近似関数を利用することなく，マルチプレクサを用いた任意の関数を実現する手法を提案する．本稿ではマルチプレクサの制御入力として変数を入力する手法と，式を入力する手法を試行し，式を入力した場合の設計の方が回路規模を低減できることを検証した．

1. はじめに

Stochastic Computing は，1960 年代に提案されたコンピューティングパラダイムである [1]．Stochastic Computing の利点として，通常の回路よりも低面積で設計が可能であること，ソフトエラー耐性に優れていることが挙げられる．そのため，近年の通常の回路の微細化にともない，より並列化が可能となり，Stochastic Computing の問題点として指摘されていた，計算精度及び，計算速度の問題が緩和されてきたため，Stochastic Computing が再注目されている [2] [3].

Stochastic Computing において，任意の関数を実現する回路の設計手法として，近似を用いて実現する手法が主に考えられてきた [4] [5]．そのため，任意の関数を実現するためには，近似関数を生成する必要がある．

そこで，本論文ではマルチプレクサを用いて，必ずしも近似関数を用いることなく，任意の関数を実現する手法を提案する．マルチプレクサ用いて設計する手法として，マルチプレクサの制御入力に変数を与える手法と，式を与える手法を提案する．また，それぞれ検証した結果，マルチプレクサの制御入力として式を入力した場合の方が，変数を入力する場合に比べ，最大 72%削減できていることが確認できた．

以下，第 2 章で Stochastic Computing の概要について述べる．続いて，第 3 章でマルチプレクサの制御入力に変数として入力する手法について述べる．そして，第 4 章でマルチプレクサの制御入力に式として入力する手法について述べる．第 5 章で提案手法の評価方法とその結果について述べる．最後に，第 6 章でまとめと今後の課題を述べる．

[1] 立命館大学大学院情報理工学研究科
[2] 立命館大学情報理工学部
[a)] neko@ngc.is.ritsumei.ac.jp
[b)] ger@cs.ritsumei.ac.jp

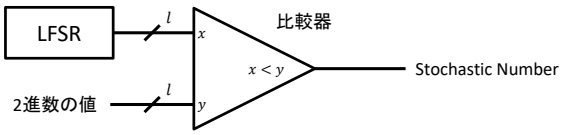

図 1　Stochastic Number の生成

2. Stochastic Computing

2.1 Stochastic Number

Stochastic Computing では，ビット列における 1 の存在確率によって数値を表現する．例えば，Stochastic Computing でビット列 00100100 は，8 ビットのうち 1 が 2 個存在するので，1 の存在確率は 1/4 であり，10 進数の 0.25 を表現する．このように数値を表現するビット列を Stochastic Number (以下，SN) と呼ぶ．ビット列の 1 の存在確率が等しい SN は，同じ値を表現する．例えば，101100 と 01010110 は Stochastic Computing において，ともに 1/2 を表現する．また，SN は一般に，ビット列の長さが長いほど，数値を精度よく表現できる [6].

本研究において，SN は，図 1 に示すように，Linear Feedback Shift Register (LFSR) を用いた乱数生成回路と比較器を用いて，2 進数の値に応じて 1 ビットずつ生成される．以下では，$X, Y, ...$ のように大文字で SN を表し，その表現する値をそれぞれ，$x, y, ...$ のように小文字で表す．そして，$X, Y, ...$ の 1 の存在確率をそれぞれ $P(X = 1), P(Y = 1), ...$

Vol.2017-ARC-225 No.20
Vol.2017-SLDM-179 No.20
Vol.2017-EMB-44 No.20
2017/3/9

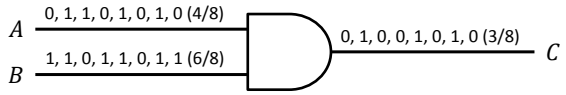

図 2 Stochastic Computing における乗算

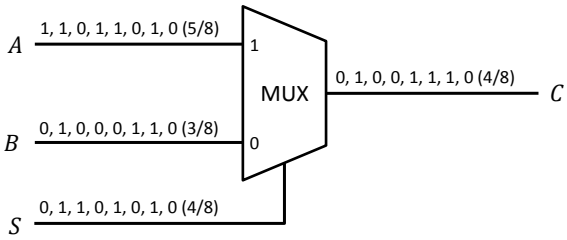

図 3 Stochastic Computing における加算

と表す．このとき，$P(X=1)=x, P(Y=1)=y, ...$ が成り立つ．また，説明する上で数値 x を表現する独立した SN を複数必要とする場合，SN はそれぞれ $X_1, X_2, ...$ と表し，$X_1, X_2, ...$ の 1 の存在確率を $P(X_1=1), P(X_2=1), ...$ と表す．このとき，$P(X_1=1)=x, P(X_2=1)=x, ...$ である．なお，$X_0, X_1, ...$ や $x_0, x_1, ...$ のような添字付きの文字を表現する SN の記号と，SN の表す値を表す記号は必要に応じて明示する．

2.2 Stochastic Computing の計算原理

Stochastic Computing の演算操作は簡単な論理ゲートで実現できる．本節では Stochastic Computing における，主な計算原理について説明する．

2.2.1 乗算

Stochastic Computing の乗算は図 2 に示されるように AND ゲートで実現できる [2]．2 個の SN の A, B に相関がないと仮定すると，

$$\begin{aligned} c &= P(C=1) \\ &= P(A=1\,and\,B=1) = P(A=1)P(B=1) \\ &= a \cdot b \end{aligned} \tag{1}$$

となる．よって，Stochastic Computing では，AND ゲートで乗算の結果を得られる．図 2 の例では，$a=4/8$，$b=6/8$ を入力したとき，$c=3/8$ となるため，正しく計算できていることがわかる．

2.2.2 加算

Stochastic Computing の加算は，適切にスケールすることにより，図 3 に示されるようなマルチプレクサで実現できる [2]．3 個の SN の A, B, C に相関がないと仮定すると，

$$\begin{aligned} c &= P(C=1) \\ &= P(S=1\,and\,A=1) + P(S=0\,and\,B=1) \\ &= P(S=1)P(A=1) + P(S=0)P(B=1) \\ &= s \cdot a + (1-s) \cdot b \end{aligned} \tag{2}$$

となる．また，式 2 は $s=1/2$ の時，

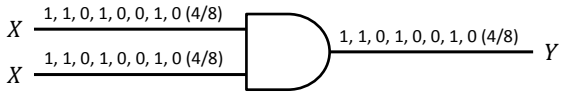

図 4 相関のある乗算

$$\begin{aligned} c &= s \cdot a + (1-s) \cdot b = \frac{1}{2}a + \left(1 - \frac{1}{2}\right) b \\ &= \frac{1}{2}(a+b) \end{aligned} \tag{3}$$

となる．よって，Stochastic Computing では，マルチプレクサで 1/2 倍にスケールされた加算の結果を得られる．図 3 の例では，$a=5/8$，$b=3/8$，$s=4/8$ を入力したとき，$c=4/8$ となるため，正しく計算されていることがわかる．

2.3 Stochastic Computing の制約

Stochastic Computing には，いくつかの制約があることが知られている．まず，Stochastic Computing はビット列の 1 の存在確率を用いて計算するため，入力の SN 同士に相関がある場合，計算の精度が悪くなることが知られている．例えば，x^2 を計算するために，AND ゲートに同じ SN の X を入力すると，出力される SN の Y は図 4 のように，X がそのまま出力される．このとき，$y=x^2$ ではなく，$y=x$ となる．したがって，計算の精度を低下させないために，論理ゲートに入力される SN が独立していることを保証する必要がある．

また，本論文で扱う Stochastic Computing の SN で表現できる数値の範囲は，$[0,1]$ である．そのため，Stochastic Computing における演算操作の入力値と出力値は，$[0,1]$ の範囲内であることを保証する必要がある．

3. 変数を制御入力に持つマルチプレクサによる設計手法

Stochastic Computing において，任意の関数を実現する手法として，近似関数を用いる設計手法が提案されている [4] [5]．これらの設計手法では，設計する関数を近似する必要があるため，近似に際して誤差が生じることが知られている [7]．そこで，本章では，マルチプレクサの制御入力として変数を用いて，近似を用いることなく，回路を設計する手法について述べる．

3.1 マルチプレクサを用いた一次関数の実現

2.2.2 節で説明したように，Stochastic Computing において，マルチプレクサは式 2 を実現する回路である．ここで，変数 x 及び定数 m, n からなる，一般的な一次関数の式を $y=mx+n$ とすると，

$$\begin{aligned} y &= mx+n = (m+n-n)x+n \\ &= x(m+n) + (1-x)n \end{aligned} \tag{4}$$

Vol.2017-ARC-225 No.20
Vol.2017-SLDM-179 No.20
Vol.2017-EMB-44 No.20
2017/3/9

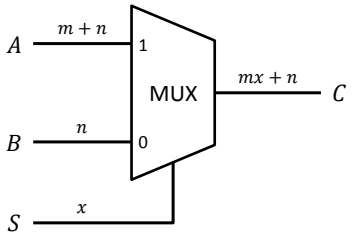

図 5 マルチプレクサの制御入力に変数を与えて一次関数を実現した回路

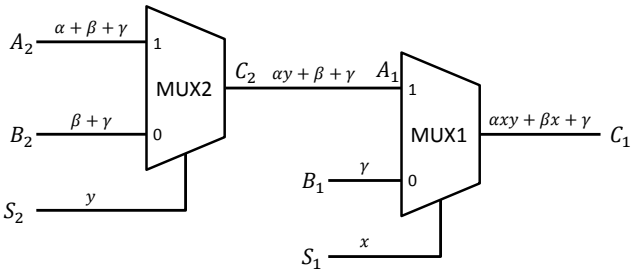

図 6 $\alpha xy + \beta x + \gamma$ における設計

のように式変形できる．このとき，式 4 は，式 2 に $(a, b, c, s) = (m + n, n, y, x)$ として代入した結果に一致する．よって，一次関数 $y = mx + n$ は，図 5 のようにマルチプレクサを用いることにより，実現できる．図 5 では，マルチプレクサの 1 側の入力として，$P(A = 1) = m + n$ となる SN の A，0 側の入力として，$P(B = 1) = n$ となる SN の B，制御入力として，$P(S = 1) = x$ となる SN の S を与えている．このように，一次関数の式をマルチプレクサで実現するとき，マルチプレクサの制御入力にのみ，変数が入力される．

3.2 マルチプレクサを用いた多変数関数の実現

任意の多変数関数は，ある変数に着目したときの一次関数とみなせる．このことを利用し，任意の多変数関数を実現することを考える．例えば，$f(x, y) = \alpha xy + \beta x + \gamma$ は，変数 x に着目して式変形すると，

$$f(x, y) = \alpha xy + \beta x + \gamma = (\alpha y + \beta)x + \gamma \tag{5}$$

となる．したがって，式 4 と同様の式変形ができることから，式 5 は，

$$\begin{aligned} f(x, y) &= (\alpha y + \beta)x + \gamma \\ &= (\alpha y + \beta + \gamma - \gamma)x + \gamma \\ &= x(\alpha y + \beta + \gamma) + (1 - x)\gamma \end{aligned} \tag{6}$$

のように式変形できる．よって，式 6 は，図 6 の MUX1 のようなマルチプレクサで実現できる．このとき，図 6 の MUX1 の 1 側に，$P(A_1 = 1) = \alpha y + \beta + \gamma$ を表す SN が入力されると，$f(x, y)$ を実現する回路が設計できる．このとき，$P(A_1 = 1) = \alpha y + \beta + \gamma$ は，y についての一次関数として同様に式変形すると，

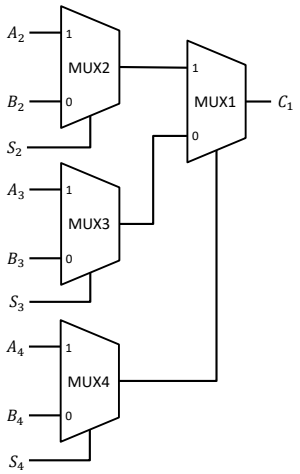

図 7 他のマルチプレクサの出力が制御入力に接続されたマルチプレクサ

$$\begin{aligned} \alpha y + \beta + \gamma &= \{\alpha + \beta + \gamma - (\beta + \gamma)\}y + \beta + \gamma \\ &= y(\alpha + \beta + \gamma) + (1 - y)(\beta + \gamma) \end{aligned} \tag{7}$$

となるため，図 6 の MUX2 のように実現できる．したがって，図 6 の MUX1 と MUX2 により，多変数関数 $f(x, y)$ を実現できる．関数を実現する回路を設計する際に，制御入力に入力する変数は，マルチプレクサの 1 側と 0 側に入力される式の項数の和が少なくなるように決定して入力する．

4. 式を制御入力に持つマルチプレクサによる設計手法

3 章で説明した手法は，マルチプレクサの制御入力として，常に変数を入力する回路が設計される．そのため，マルチプレクサの制御入力として変数を与える手法では，図 7 のような，マルチプレクサの制御入力に，他のマルチプレクサの出力を接続した回路，すなわち，式で与えられる回路が設計できないという制限が存在する．そのため，3 章で述べた手法では設計できない回路において，より効率的に設計できる可能性がある．また，マルチプレクサの制御入力に，独立して生成された SN をそれぞれ入力する必要があるため，図 1 のような，SN を生成する回路が増加する．本章では，マルチプレクサの制御入力として式を与えることにより，必要とするマルチプレクサの個数を削減する手法を提案する．

4.1 マルチプレクサの 1 側と 0 側に入力される式の導出方法

制御入力として式を与えることで，関数 F を実現するマルチプレクサを図 8 に示す．このとき，マルチプレクサの 1 側の入力を関数 f，0 側の入力を関数 g，制御入力を関数 h とする．これは，式 2 から，

$$F = hf + (1 - h)g$$

Vol.2017-ARC-225 No.20
Vol.2017-SLDM-179 No.20
Vol.2017-EMB-44 No.20
2017/3/9

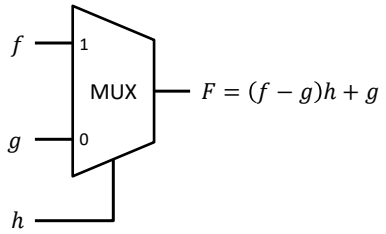

図 8　制御入力として式を与えるマルチプレクサ

$$= hf - hg + g$$
$$= (f-g)h + g \tag{8}$$

となり，F を h で除算したとき，剰余として g が得られることがわかる．また，商の式 $f-g$ に，導出した g を加算すると，f も得られる．したがって，マルチプレクサの制御入力に入力する式を決定すると，マルチプレクサの 1 側および 0 側に入力する式も決定することができる．

4.2　除算する式の決定方法

本節では，設計する回路の増加を抑制するように，マルチプレクサの制御入力に与える式を決定する手法を提案する．

4.2.1　除算する式の決定方法

マルチプレクサは，$s=0$ または $s=1$ でないとき，式 4 において，式の項数は 2 個になっている．また，式 4 と項数が式 4 より 1 個多い式 5 を比較すると，それぞれ実現した図 5 と図 6 から，必要となるマルチプレクサが 1 個増加していることがわかる．これらの点から，マルチプレクサを回路に用いると，式の項数が増加すると考えられる．よって，設計される回路規模を削減するためには，入力として与える式の項数を減らせばよい．したがって，除算する式と，除算した際に得られる，商の式，商と剰余の和の式の 3 つの式における項数の和が少なくなるように，除算する式を決定する．そこで，不定値を用いて除算する式を決定することを考える．Algorithm 1 で除算する式を決定する手順を示す．以下では，多変数関数

$$f(x,y) = -\frac{1}{18}xy + \frac{2}{9}x + \frac{1}{24}y^2 + \frac{9}{112}y + \frac{23}{168} \tag{9}$$

を実現する場合を例として示す．まず，Algorithm 1 の 1 から 4 行目で実現する関数の変数と項数を取得する．次に，Algorithm 1 の 5 行目で実現する関数の項数がより多いことを確認する．実現する関数 $f(x,y)$ は変数が 2 種類，項数が 5 個の式なので，条件を満たす．よって，Algorithm 1 の 6 から 23 行目を行う．そして，Algorithm 1 の 6 から 15 行目で，変数 x,y と不定値 a,b,c からなる式

$$ax + by + c \tag{10}$$

を除算する式として仮決めする．なお，実現する関数の変数が x,y,z のような 3 種類以上の場合も同様に，仮決めする式は，不定値 a,b,c,d を用いて，$ax+by+cz+d$ のよう

Algorithm 1 除算する式を決定するアルゴリズム

```
1: f ← 実現する関数
2: finalDivider ← 1                          ▷ 最終的に除算する式
3: fTerms ← 実現する関数の項数
4: fVars ← 実現する関数の変数の種類の数
5: while fVars <= fTerms + 2 do
6:     fTermList[fVars] ← 実現する関数の変数,x, y, z, ...
7:     unDef[fVars + 1] ← 不定値, a, b, c, ...
8:     divideFactor ← 0
9:     for i = 0 to fVars do                 ▷ 仮決めした式の作成
10:        if i ≠ fVars then                 ▷ 不定値 * 変数の項の追加
11:            divideFactor += fTermList[i] * unDef[i]
12:        else                              ▷ 不定値の項の追加
13:            divideFactor += unDef[i]
14:        end if
15:    end for
16:    quotient = f / divideFactor           ▷ 商の計算
17:    h = f % divideFactor      ▷ マルチプレクサの 0 側の入力
18:    g = quotient + sideZero   ▷ マルチプレクサの 1 側の入力
19:    solve unDef               ▷ g と h の式の項をできるだけ消去
20:    finalFactor = finalFactor * divideFactor
21:    f = quotient      ▷ 商の式を新たな実現する関数として設定
22:    fTerms ← f の項数
23:    fVars ← f の項数
24: end while
25: if finalDivider == 1 then      ▷ 除算する式が未発見の場合
26:    finalFactor ←3 章で決定される変数
27: end if
```

Vol.2017-ARC-225 No.20
Vol.2017-SLDM-179 No.20
Vol.2017-EMB-44 No.20
2017/3/9

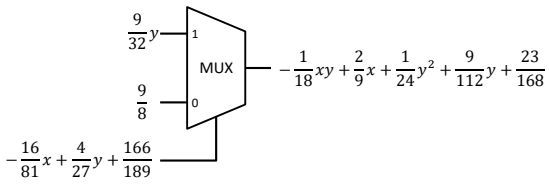

図 9 提案手法による $f(x,y)$ の設計

に設定する．Algorithm 1 の 16 行目と 18 行目では，式 9 を式 10 で除算し，

$$\begin{aligned} f(x,y) &= -\frac{1}{18}xy+\frac{2}{9}x+\frac{1}{24}y^2+\frac{9}{112}y+\frac{23}{168} \\ &= (ax+by+c)\left(-\frac{1}{18a}y+\frac{2}{9a}\right) \\ &\quad +\left(\frac{b}{18a}+\frac{1}{24}\right)y^2+\left(\frac{c-4b}{18a}+\frac{9}{112}\right)y \\ &\quad -\frac{2c}{9a}+\frac{23}{168} \end{aligned} \tag{11}$$

のように，商の式と剰余の式から，マルチプレクサの 1 側と 0 側に入力される式を得る．$f(x,y)$ の場合，マルチプレクサの 1 側と 0 側に入力される式 $g(y)$ および $h(y)$ は，それぞれ以下のように導出される．

$$\begin{aligned} g(y) &= \left(\frac{b}{18a}+\frac{1}{24}\right)y^2+\left(\frac{c-4b-1}{18a}+\frac{9}{112}\right)y \\ &\quad -\frac{2c-2}{9a}+\frac{23}{168} \end{aligned} \tag{12}$$

$$\begin{aligned} h(y) &= \left(\frac{b}{17a}+\frac{1}{24}\right)y^2+\left(\frac{c-4b}{18a}+\frac{9}{112}\right)y \\ &\quad -\frac{2c}{9a}+\frac{23}{168} \end{aligned} \tag{13}$$

そして，Algorithm 1 の 19 行目では不定値 a,b,c を決定する．不定値の決定方法については，3.2.2 節にて詳細を述べる．このとき，$(a,b,c)=(-\frac{16}{81},\frac{4}{27},\frac{166}{189})$ を得るとすると，式 10 と式 12 および式 13 はそれぞれ，

$$ax+by+c = -\frac{16}{81}x+\frac{4}{27}y+\frac{166}{189} \tag{14}$$

$$g(y) = \frac{9}{32}y \tag{15}$$

$$h = \frac{9}{8} \tag{16}$$

となる．Algorithm 1 の 20 行目では，最終的に除算する式を暫定的に更新する．$f(x,y)$ の場合，最終的に除算する式が 1 だったため，最終的に除算する式は式 14 に更新される．そして，Algorithm 1 の 21 から 23 行目で，得た商の式を新たな実現する関数とする．その後，Algorithm 1 の 5 行目から繰り返し実行される．関数 $f(x,y)$ の場合，上述したように各入力に与える式を求めたのち，Algorithm 1 の 21 行目で式 $\frac{9}{32}y-\frac{9}{8}$ を新たな関数として，Algorithm 1 の 5 行目に戻ると，変数が 1 種類，項数が 2 個となっているので，繰り返さることはない．よって，式 14 が最終的な除算する式となる．結果として，$f(x,y)$ は，図 9 のようなマルチプレクサで実現される．

上記の例では，$f(x,y)$ を x についての次数の降順で各項が並べられている．本手法では，特定の変数についての式として除算を行うため，選択する変数によって，商の式と剰余の式が異なる．例えば，式 9 に対して除算する式を仮決めする際，y について行うと，

$$\begin{aligned} f(x,y) &= \frac{1}{24}y^2+\frac{9}{112}y-\frac{1}{18}yx+\frac{2}{9}x+\frac{23}{168} \\ &= (by+ax+c)\left(\frac{1}{24b}y-\frac{1}{18b}x+\frac{9}{112b}\right) \\ &\quad +\frac{a}{18b}x^2-\frac{a}{24b}yx+\left(\frac{c}{18b}-\frac{9a}{112b}+\frac{2}{9}\right)x \\ &\quad -\frac{c}{24b}y-\frac{9c}{112b}+\frac{23}{168} \end{aligned} \tag{17}$$

のようになる．したがって，マルチプレクサの入力も異なる．そのため，実現する関数のすべての変数についても Algorithm 1 の操作を試行し，除算する式を求める．$f(x,y)$ の例では，変数 y についても Algorithm 1 の操作を行い，除算する式として，$\frac{1}{4}y$ を得る．その後，マルチプレクサの各入力に与えられる式の項数の和が，最も少ないものを選択する．$f(x,y)$ の例では，変数 y の場合のマルチプレクサの 1 側の入力の式が $\frac{1}{24}y^2+\frac{9}{112}y+\frac{23}{168}$ と，0 側の入力の式が $\frac{2}{9}x+\frac{1}{24}y^2+\frac{9}{112}y+\frac{23}{168}$ となる．よって，マルチプレクサに入力される式の項数の総和は，変数 x については 5 個，変数 y については 8 個となるので，変数 x について得られた除算する式 14 が選択され，図 9 のように設計される．

4.2.2 不定値の係数の決定方法

本手法では，4.2.1 節で説明したように，不定値を用いた除算を行っている．これにより，除算した後の式の各項の係数の値は可変である．このとき，係数の値が 0 になれば，その項を消去することができ，式の項数を削減できる．よって，除算した後の商の式と，商の式と剰余の式の和の式における，いずれかの項の係数の値が 0 になるように，不定値を決定する．

不定値を決定する際は，消去する項が，できるだけ次数の高い項となるようにする．これは，変数の次数が大きい項は，その項を表す SN を生成するために必要な回路が，次数の大きさに応じて必要となるため，回路規模の増大に繋がるためである．消去する項は，貪欲法にて決定する．

以下で，式 12 と式 13 から，不定値を決定する手順を例として示す．まず，式 12 と式 13 のうち，次数の高い項は，$\left(\frac{b}{18a}+\frac{1}{24}\right)y^2$ であるので，この項の係数 $\left(\frac{b}{18a}+\frac{1}{24}\right)y^2$ が 0 となるように不定値を求める．このとき，

$$\begin{aligned} \frac{b}{18a}+\frac{1}{24} &= 0 \\ a &= -\frac{4}{3}b \end{aligned} \tag{18}$$

が得られる．次に，式 13 の $\left(\frac{c-4b}{18a}+\frac{9}{112}\right)y$ の係数を 0 にすることを考える．また，式 18 と合わせて，

$$\frac{c-4b}{18a}+\frac{9}{112} = 0$$

$$b = \frac{14}{83}c \tag{19}$$

$$a = -\frac{56}{249}c \tag{20}$$

が得られる．続いて，式 13 の $\left(\frac{c-4b}{18a} + \frac{9}{112}\right)y$ の項を 0 にするとき，式 19 と式 20 の両方を満たす解は存在しないため，次に次数の高い項の消去を試行する．このとき，式 12 の定数項 $-\frac{2c-2}{9a} + \frac{23}{168}$ から，

$$-\frac{2c-2}{9a} + \frac{23}{168} = 0$$

$$c = \frac{207}{336}a + 1 \tag{21}$$

が得られる．そして，式 18，式 19 および式 21 と合わせて，

$$b = \frac{14}{83}\left(\frac{207}{336}a + 1\right) = \frac{69}{664}a + \frac{14}{83} = \frac{4}{27} \tag{22}$$

$$a = -\frac{4}{3}b = -\frac{4}{3}\left(\frac{4}{27}\right) = -\frac{16}{81} \tag{23}$$

$$c = \frac{207}{336}a + 1 = \frac{207}{336}\left(-\frac{16}{81} + 1\right) = \frac{166}{189} \tag{24}$$

となり，$(a, b, c) = (-\frac{16}{81}, \frac{4}{27}, \frac{166}{189})$ とするとき，消去することができる．

このように不定値を決定するにあたり，項を消去できない可能性が存在する．その場合，暫定的な最終的な除算する式から，除算する式を得る．また，Algorithm 1 の 20 行目で除算する式を得られなかった場合，Algorithm 1 の 25 から 27 行目で，2 章で述べた手法により，マルチプレクサの制御入力に与える項を決定する．

5. 検証結果と考察

5.1 評価方法

本論文で提案した，近似を用いない任意の関数を実現する手法として，マルチプレクサの制御入力に変数を入力する手法と，式を入力する手法において，与えられた式を実現する回路を設計し，マルチプレクサの個数を比較する．以下に与えた式を示す．

式 1: $\frac{1}{24}x^2 + \frac{9}{112}x - \frac{1}{18}xy + \frac{2}{9}y + \frac{23}{168}$
式 2: $\frac{1}{24}x^2 + \frac{9}{112}x + \frac{23}{168}$
式 3: $\frac{2}{5}x^2y^2 + \frac{1}{14}x^2y + \frac{3}{25}xy^2 - \frac{4}{7}xy + \frac{1}{24}y + \frac{4}{41}$
式 4: $\frac{1}{18}x^2 + \frac{13}{63}xy + \frac{77}{468}x + \frac{4}{21}y^2 + \frac{79}{273}y + \frac{5}{78}$
式 5: $\frac{5}{56}x^2 - \frac{235}{448}xy - \frac{5}{112}x - \frac{2}{7}y + \frac{2}{9}$
式 6: $\frac{5}{56}x^3 + \frac{3}{64}x^2y^2 - \frac{3}{38}x^2y - \frac{5}{112}x^2 + \frac{15}{112}xy - \frac{30}{119}x + \frac{9}{128}y^3 - \frac{261}{760}y^2 - \frac{2077}{4256}y + \frac{55}{126}$

5.2 検証結果と考察

検証結果を表 1 に示す．表 1 から，式 3 以外の式を実現する回路において，必要なマルチプレクサの個数は，2 章で述べたマルチプレクサの制御入力に変数を入力手法に比べ，少数，もしくは同数であることが確認された．式 6 を実現する場合，マルチプレクサの個数が最大 72%削減できていることが確認された．一方，式 3 を実現する場合，必要となるマルチプレクサの個数がマルチプレクサの制御入力に変数を入力する場合に比べ，増加していることが確認された．これは，除算した際に，剰余の式の項数が増大に対して，不定値の決定時に，消去できる項が少なかったことが考えられる．除算した際に生じる剰余の式の項数の増加に対して，不定値の決定時に消去することができた項数が少なかったからであると考えられる．

表 1 設計するために必要なマルチプレクサの個数

	変数を入力する手法	式を入力する手法
式 1	5	4
式 2	2	2
式 3	5	8
式 4	13	6
式 5	5	5
式 6	59	16

6. おわりに

本研究では，Stochastic Computing において，任意の関数を近似を用いずに実現する手法として，マルチプレクサの制御入力に変数を入力する手法と式を入力する手法を提案した．検証の結果，マルチプレクサの制御入力として式を与えることにより，マルチプレクサを削減できることが確認された．今後の研究として，5.1 節の式 3 の例に対しても，回路を削減できる式を生成する手法を考案する必要があると考えられる．

参考文献

[1] Gaines, B.: Stochastic computing systems, *Advances in information systems science*, Springer, pp. 37–172 (1969).
[2] Li, P.: Analysis, design, and logic synthesis of finite-state machine-based Stochastic computing, PhD Thesis, The University of Minnesota (2013).
[3] Alaghi, A. and Hayes, J. P.: Survey of stochastic computing, *ACM Transactions on Embedded computing systems (TECS)*, Vol. 12, No. 2s, p. 92 (2013).
[4] Li, X., Qian, W., Riedel, M. D., Bazargan, K. and Lilja, D. J.: A reconfigurable stochastic architecture for highly reliable computing, *Proceedings of the 19th ACM Great Lakes symposium on VLSI*, ACM, pp. 315–320 (2009).
[5] Zhao, Z. and Qian, W.: A general design of stochastic circuit and its synthesis, *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, IEEE, pp. 1467–1472 (2015).
[6] 石井章太，砂盛大貴，市原英行，　岩垣剛，井上智生：相関を持つストカスティック数の演算精度に与える影響に関する考察 (応用設計, システムオンシリコンを支える設計技術)，電子情報通信学会技術研究報告. VLD, VLSI 設計技術， Vol. 113, No. 454, pp. 79–84 (2014).
[7] Wu, Y., Wang, C. and Qian, W.: Minimizing error of stochastic computation through linear transformation, *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, ACM, pp. 349–354 (2015).