Stochastic Computing に用いる定数を近似する回路の合成手法

Synthetic method of circuit that approximates the constant for the Stochastic Computing

壷阪 幸輝 † 山下 茂 ‡
Koki Tsubosaka Shigeru Yamashita

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，従来回路よりも低面積で設計が可能であり，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，従来回路の微細化に伴い，従来回路以上に回路の並列化が可能であることから，Stochastic Computing が再注目されている [2]．

Stochastic Computing において，必要とする定数を得る手法として，真にランダムなソースから論理回路を用いて，候補回路に回路を付加して設計する手法が考えられている [3] [4]．しかし，これらの手法では，設計できる回路の形状にある種の制限がある．

そこで，本論文では，回路の形状に制限なく，必要とするゲート数の少ない論理式を選択するヒューリスティックを提案する．本手法により設計した回路において必要とするゲート数と精度を検証した．検証の結果，エラー率が十分小さく設計できる場合に，必要とするゲート数を削減できた．

2. Stochastic Computing

本章では，Stochastic Computing の概要を述べる．

2.1 Stochastic Number

Stochastic Computing は，ビット列に存在する 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 の数が 2 個なので，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したビット列を Stochastic Number (以下，SN) と呼ぶ．SN はビット列の 1 の存在確率が等しければ，同じ値を表現する．例えば，101100 と 01010110 は SN において，ともに 1/2 を表現する．また，SN は一般に，SN のビット列の長さが長いほど，数値を精度よく表現できる [5]．

以下では，数値 $x, y, ...$ を表現する SN をそれぞれ $X, Y, ...$ とする．そして，$X, Y, ...$ の中の 1 の存在確率をそれぞれ $P(X = 1), P(Y = 1), ...$ と表す．

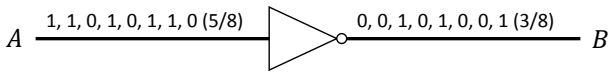

図 1: NOT 回路

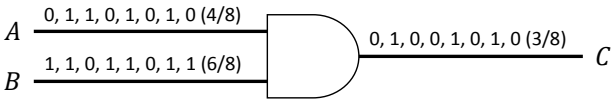

図 2: AND 回路

2.2 Stochastic Computing における各回路の機能

本節では，Stochastic Computing における各回路の機能について述べる．

2.2.1 NOT 回路

Stochastic Computing において，図 1 のような NOT 回路は，

$$\begin{aligned} b &= P(B = 1) = P(A = 0) \\ &= 1 - a \end{aligned} \tag{1}$$

より，入力される SN に対して，式 1 のように SN を出力として得られる．図 1 の例では，$a = 5/8$ で $b = 3/8$ を得ている．

2.2.2 AND 回路

Stochastic Computing において，図 2 のような AND 回路は，

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

より，入力される互いに独立の二つの SN に対して，式 2 のように SN を出力として得られる．図 2 の例では，$a = 4/8$，$b = 6/8$ で $c = 3/8$ を得ている．

† 立命館大学大学院, Graduate School of Information Science and Engineering, Ritsumeikan University
‡ 立命館大学, Ritsumeikan University

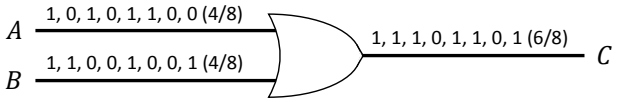

図 3: OR 回路

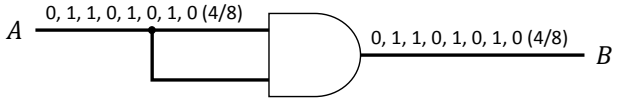

図 4: 相関のある入力

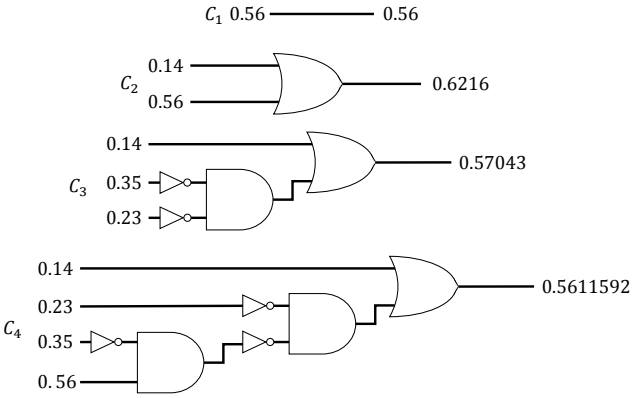

図 5: 候補回路にゲートを加えた設計

2.2.3 OR 回路

Stochastic Computing において，図 3 のような OR 回路は，

$$\begin{aligned} c &= P(C=1) \\ &= P(A=1 \, or \, B=1) \\ &= P(A=0)P(B=1) + P(A=1)P(B=0) \\ &\quad + P(A=1)P(B=1) \\ &= (1-a)b + a(1-b) + a \cdot b \\ &= a + b - a \cdot b \end{aligned} \tag{3}$$

より，入力される互いに独立の二つの SN に対して，式 3 のように SN を出力として得られる．図 3 の例では，$a = 4/8$，$b = 4/8$ で $c = 6/8$ を得ている．

2.3 Stochastic Number の相関

Stochastic Computing はビット列の 1 の存在確率を用いているため，入力値同士に相関がある場合，計算精度が悪くなることが知られている．例えば，a^2 を得るために，AND ゲートに同じ SN の A を入力すると，出力される SN の B は図 4 のように，A がそのまま出力され，$b = a^2$ ではなく，$b = a$ となる．そのため，論理ゲートに入力される SN が，独立に生成されていることを保証する必要がある．

3. 既存の定数に近似する設計

Qian と Wang らは候補回路にゲートを追加して，次の候補回路を設計する手法を提案した [3] [4]．本章では，Wang らの手法について概要を説明する．

3.1 真にランダムなソースの定数近似

SN は，一般的に Stochastic Number Generator (以下，SNG) と呼ばれる擬似乱数を用いて生成されている [2]．しかし，SNG は必要とするハードウェアコストが大きい．そこで，SN を得るために，高エントロピー物理ソースの例えば，熱雑音，放射性崩壊，ブラウン運動等を用いた，真にランダムなソースを用いることが考えられている [4]．真にランダムなソースを用いる場合，制御が困難なため，利用するソースが異なる確率を提供することがある．よって，所望の SN はソースから組み合わせ回路を用いて取得する．例えば，真にランダムなソースから，0.4, 0.5, 0.7 が与えられ，これを入力確率として用いることができるとき，0.2 の SN を生成したいとすると，AND ゲートに 2 つの入力確率として 0.4 と 0.5 を選ぶと，出力確率は目標確率の 0.2 となる．

このとき，それぞれのソースのランダムビットは与えられた入力確率がそれぞれ，高々1 度のみ利用できるとする．以上のことから，本論文の問題設定は次のようになる．

> ソース確率集合 $S = \{p_1, p_2, ..., p_n\}$ と目標確率 q^* が与えられ，S から選ばれた確率のランダム入力を受け取り，確率 q^* の SN を生成する回路を設計する． S の各要素は，最大 1 回まで入力確率として使用することができる．

なお，ソース確率集合の各要素を論理式の各変数とみなしたとき，0-1 線形計画法による最小項の組み合わせ最適化問題を解くことで，構成可能な全ての論理式を計算することにより，最適解を求めることが可能である．しかし，これは NP 完全問題であることと，必要とするゲート数が大きくなるため，ヒューリスティックに解くことが必要とされる [3].

3.2 候補回路へのゲート追加による設計

ヒューリスティックに解く手法として，候補回路にゲートを追加するインクリメンタルな設計手法が考えられている [4].

[4] では，図 5 のように，候補回路の C_1 から次の候補回路の C_2，さらに候補回路 C_2 から次の候補回路 C_3… のように，存在する入力にゲートを追加して，インクリメンタルに次の候補回路を設計し，設計した候補回路で最も目標確率に近い出力をする回路を選択する．図 5 で

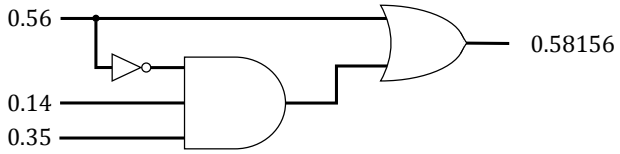

図 6: 形状に制限のない回路

はソース確率集合 $S = \{0.14, 0.23, 0.35, 0.56\}$ から目標確率 0.58 をインクリメンタルに設計し，設計された候補回路のうち，最も目標確率に近い値を出力する候補回路を選択する．図 5 では C_3 の候補回路が選択される．

4. 形状に制限のない回路設計

本章では，形状に制限なく回路を発見することが可能なヒューリスティックを提案する．

4.1 基本方針

3.2 節で述べたように，最適解を求める場合，ソース確率集合の各要素を論理式の各変数とみなしたときの，全ての最小項の組み合わせを求めることになる．しかし，[3] および，3.2 節で示した [4] で提案している設計では，最小項を用いることなく，インクリメンタルに回路と入力を付加して設計している．そのため，図 6 のような，入力から複数のゲートに結線された回路を設計することが不可能であり，発見可能な解空間が狭くなってしまっている．

そこで本論文では，設計する回路のハードウェアコストを考慮しながら，最小項を貪欲に選択することにより，目的確率に近似させる手法を提案する．

4.2 最小項を貪欲に選択するアルゴリズム

最小項を貪欲に選択するだけでは，選択されるの項が離散的になることが考えられるため，必要とするハードウェアコストが増大することが考えられる．そのために，各確率をどの変数に割り当てるかを考える必要がある．また，真理値表において連続に選択すると，ハードウェアコストを削減した回路を設計できると考えられる．よって，以下のように，出力を定数に近似させる設計手法を提案する．

ステップ 1 ソース確率集合にそれぞれ変数を指定する．

ステップ 2 指定された変数順に最小項を計算した表を作成する．

ステップ 3 表の先頭から順に貪欲に最小項を選択し，最も目標確率に近くなると終了する．

ステップ 4 得られた最小項の組み合わせで回路を設計する．

表 1: 最小項の確率値

最小項	確率値
$abcd$	0.0211288
$abc\overline{d}$	0.0063112
$ab\overline{c}d$	0.0392392
$ab\overline{c}\overline{d}$	0.0117208
$a\overline{b}cd$	0.1297912
$a\overline{b}c\overline{d}$	0.0387688
$a\overline{b}\overline{c}d$	0.2410408
$a\overline{b}\overline{c}\overline{d}$	0.0719992
$\overline{a}bcd$	0.0166012
$\overline{a}bc\overline{d}$	0.0049588
$\overline{a}b\overline{c}d$	0.0308308
$\overline{a}b\overline{c}\overline{d}$	0.0092092
$\overline{a}\overline{b}cd$	0.1019788
$\overline{a}\overline{b}c\overline{d}$	0.0304612
$\overline{a}\overline{b}\overline{c}d$	0.1893892
$\overline{a}\overline{b}\overline{c}\overline{d}$	0.0565708

ステップ 5 (1 − 目標確率) についてステップ 1 からステップ 4 を試行する．

ステップ 6 得られた二つの回路のうち目標確率に近い出力確率を得られる回路を選択する．

ステップ 1 は，後述する方法で，肯定および否定の両方の論理を考慮した上で，各変数が $a, b, c, d, ..$ と順に選択される．例えば，目標確率が 0.58 で，ソース確率集合，$S = \{0.14, 0.23, 0.35, 0.56\}$ のとき，後述する方法で $a = 0.56$，$b = 0.14$，$c = 0.35$，$d = 1 - 0.23 = 0.77$ と選択される．

次に，ステップ 2 で表 1 のように最小項が計算される．

そして，ステップ 3 で表 1 の最小項 $abcd$ の確率値 0.211288 から順に，表 1 の下に向かって，目標確率を超えるまで貪欲に確率値を加算する．最小項 $abcd$ から $\overline{a}b\overline{c}d$ まで確率値を加算すると，加算した結果が 0.6123908 となり，目標確率を超える．このとき，加算する前の結果，すなわち，最小項が $abcd$ から $\overline{a}bc\overline{d}$ までの確率値の和が 0.58156 であることがわかっているので，二つの値と目標確率との差の絶対値の小さい方を選択する．この場合，$|0.6123908 - 0.58| = 0.0323908$，$|0.58156 - 0.58| = 0.00156$ となり，絶対値の差が最小項 $abcd$ から $\overline{a}bc\overline{d}$ までの方が小さいので，こちらを選択する．よって，$abcd + abc\overline{d} + ab\overline{c}d + ab\overline{c}\overline{d} + a\overline{b}cd + a\overline{b}c\overline{d} + a\overline{b}\overline{c}d + a\overline{b}\overline{c}\overline{d} + \overline{a}bcd + \overline{a}bc\overline{d}$ が選択され，これを簡単化した $a + \overline{a}bc$ を得る．

さらに，ステップ 4 でステップ 3 で得られた論理式を合成する．このとき，ステップ 1 で否定を選択した変数

d について NOT ゲートを付加した回路を合成する．そして，$a + \bar{a}bc$ を実現する図 6 の回路を得る．

また，ステップ 5 では，回路の出力に NOT 回路を追加して設計できる回路，すなわち，(1 - 目標確率) に近い出力する回路をステップ 1 からステップ 4 にしたがって探索する．以上の例では $1 - 0.58 = 0.42$ から，0.42 をステップ 1 からステップ 4 にしたがって回路を設計する．

最後に，ステップ 5 までで得られた二つの回路から，より目標確率との絶対値の差が小さい回路を選択する．以上の例では，0.58 を目標確率とするとき，0.58156 を得られ，0.42 を目標確率とするとき，0.43043 を得られるので，図 6 のような 0.58 を目標確率とする回路を選択する．

4.3 ソース確率集合から変数を決定するアルゴリズム

4.2 節で提案したアルゴリズムのステップ 1 について述べる．

まず，原則として，目標確率に近い変数から選択する．4.2 節で述べたように，表 1 の最小項 $abcd$ の確率値 0.211288 から順に，表 1 の下に向かって，目標確率を超えるまで貪欲に確率値を加算していく．よって例えば，ソース確率およびその否定のうちで，目標確率に最も近い 0.56 を変数 a と決めると，表 1 の最小項 $abcd$ の確率値 0.211288 から $a\bar{b}\bar{c}\bar{d}$ の確率値 0.0719992 まで全て加算すると，結果は 0.56 となる．したがって，$abcd$ から $a\bar{b}\bar{c}\bar{d}$ を超えた範囲で，$abcd$ から順に加算された結果が 0.58 に最も近づく場合があるはずである．そのため，次の試行では $\bar{a}bcd$ から $\bar{a}\bar{b}\bar{c}\bar{d}$ までの範囲で探索すればよい．また，$\bar{a}bcd$ から $\bar{a}\bar{b}\bar{c}\bar{d}$ までの範囲で探索する場合，全ての最小項には $\bar{a}$ が乗算されている．よって，目標確率 0.58 から 0.56 を減算した 0.02 を新たな目標確率とした上で，目標確率に対し $\bar{a}$，すなわち 0.44 で割った値に最も近い値を探索すると，加算された結果の範囲が，0.56 から 1 を 0 から 1 として，更新された目標確率 0.02 に最も近い値を探索することと，みなすことができる．なお，目標確率が探索した値を上まった場合，例えば，以上の例において目標確率 0.58 ではなく，0.53 を探索するような場合，$abcd$ から $a\bar{b}\bar{c}\bar{d}$ までを探索する．

Algorithm 1 に本節で述べた変数を決定するアルゴリズムを示す．例えば，目標確率が 0.58 で，ソース確率集合 $S = \{0.14, 0.23, 0.35, 0.56\}$ のとき，Algorithm 1 の 1 行目から，$target$ は 0.58 となる．そして，x が前述の探索する範囲の既に決定した変数のみから作成できる確率である．Algorithm 1 の 3 行目で初期状態として x は 1 で初期化されている．Algorithm 1 の 5 行目で，ソース確率集合 S は $S = \{0.14, 0.23, 0.35, 0.44, 0.56, 0.65, 0.77, 0.86\}$ となる．Algorithm 1 の 7 行目以下では次のことを繰り返す．まず，$target$ に最も近い 0.56 を変数 a として選択する．

Algorithm 1 ソース確率集合から変数を決定

```
1: target ⇐ 目標確率
2: var_num ⇐ ソース確率集合 S の要素数
3: x ⇐ 1
4: for i = 1 to var_num do
5:    S に 1 - S[i] の要素を追加
6: end for
7: for i = 1 to var_num do
8:    j ⇐ S の配列要素のうち最も target/x に近い要素の値
9:    chose_var[i] ⇐ j
10:   S から j の要素を除去
11:   S から 1 − j の要素を除去
12:   if target > x ∗ j then
13:      target ⇐ target − x ∗ j
14:      x ⇐ x ∗ (1 − j)
15:   else
16:      x ⇐ x ∗ j
17:   end if
18: end for
```

また，Algorithm 1 の 10 行目と 11 行目により，ソース確率集合は $S = \{0.14, 0.23, 0.35, 0.65, 0.77, 0.86\}$ となる．そして，$target = 0.58 > 1 * 0.56$ なので，Algorithm 1 の 13 行目から，$target$ は $0.58 - 0.56 = 0.02$ となる．また，Algorithm 1 の 14 行目から，x は 0.44 となる．その後，$target/x = 0.02/0.44 = 0.045...$ に最も近い 0.14 を変数 b として選択される．以降同様に変数 c，d として 0.35，0.77 が選択される．

5. 検証結果

本章では，提案した手法の検証と考察を述べる．

5.1 評価方法

[4] においては，ソース確率集合および，目標確率の値がランダムに設定されているが，刻み幅が示されていない．よって，比較が困難なため，最適な回路との比較を行う．最適な回路を得るためには，3.1 節で述べたように，NP 完全問題を解くので時間を要する．そのため，ソース確率集合の要素数は 4 個に限定した．そして，ソース確率集合と目標確率を刻み幅が $\frac{1}{1024}$ のランダムに 100 回試行し，ゲート数を検証した．なお，ゲート数は NOT ゲートは無視し，AND ゲートの個数で評価を行う．

5.2 検証結果

目標確率に対する，提案した手法による回路のエラー率と最適な回路のエラー率の差ごとに，ゲート数の平均を表 2 に示す．表 2 から，半数以上が，最適な回路とエ

表 2: 設計に必要な AND ゲート数

エラー率の差	該当数	最適	提案手法
$< 0.1\%$	19	5.79	2.21
$< 1\%$	41	6.44	2.29
$< 10\%$	39	6.90	2.08
$\geq 10\%$	1	8	3

ラー率が 1%未満しか変わらない精度において，必要とするゲート数を 1/2 以下で設計できることが確認できた．

6. おわりに

本研究では，真にランダムなソースから所望の定数に近似させるヒューリスティックを提案した．既存手法によるヒューリスティックでは，入力から複数のゲートに結線された回路を設計できないため，設計できる回路の形状に制限があるが，提案したヒューリスティックでは，入力から複数のゲートに結線された回路も設計可能である．また，提案した手法では，必要とするゲート数も少なく設計することができることを確認した．

表 2 から，1%以上のエラー率の差のケースが少なくないので，より精度を向上させることが，今後の研究として考えられる．

参考文献

[1] BR Gaines. Stochastic computing systems. In *Advances in information systems science*, pp. 37–172. Springer, 1969.

[2] Armin Alaghi and John P Hayes. Survey of stochastic computing. *ACM Transactions on Embedded computing systems (TECS)*, Vol. 12, No. 2s, p. 92, 2013.

[3] Weikang Qian, Marc D Riedel, Hongchao Zhou, and Jehoshua Bruck. Transforming probabilities with combinational logic. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Vol. 30, No. 9, pp. 1279–1292, 2011.

[4] Chen Wang and Weikang Qian. Optimizing multilevel combinational circuits for generating random bits. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pp. 139–144. IEEE, 2013.

[5] 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.