ストカスティック数を用いた絶対値関数及び不連続関数の実装と評価

石川 遼太[1]　多和田 雅師[1]　戸川 望[1]

概要：ストカスティックコンピューティングでは算術演算や関数を単純な論理回路として実装することができる．単純な論理回路は小面積・低消費電力であるため様々なアプリケーションへの応用が研究されている．ビット列の 1 の割合で値を定めるストカスティック数を用いて単純な論理回路で算術演算・関数を実装するとき，もとの関数が微分可能でなければならない制約が存在する．本稿では以前に提案したストカスティック数によるステップ関数を用いてこの制約を回避し，微分可能ではないストカスティック関数を実現する．微分可能ではない関数として、ストカスティック数による絶対値関数及び不連続関数を実装，評価する．

1. はじめに

計算結果に誤りを許容することで回路の微細化を目論む設計戦略としてストカスティックコンピューティング (SC) が存在する [1]．SC はビットレベルの論理演算を繰り返し，確率的な算術演算を達成する．論理回路の回路面積は小さいため，SC により全体の回路面積の削減・消費電力の削減が期待できる．演算 1 回ごとの精度が求められず，繰り返し演算するアプリケーションとしてニューラルネットワークの活性化関数がある．活性化関数を算術関数として実装する研究が進んでいる [3–5]．一方で，SC による実装が難しい算術関数として急峻な関数や不連続関数が挙げられる．活性化関数のうち Rectified Linear Unit (ReLU) 関数やステップ関数，これらの合成関数は急峻な関数や不連続関数として出現する．SC の実応用に向けて急峻な関数や不連続関数の実装は必要不可欠である．

関数に入力された値を Flip-Flop に保存しておき急峻な出力の境を判定し，SC としてステップ関数を実現する研究が存在する [6]．この SC によるステップ関数の不連続性を適用して任意の急峻な関数や不連続関数を実現できる可能性がある．本稿では SC によるステップ関数を用い，急峻な関数として絶対値関数および，不連続関数として任意の関数の合成による不連続関数を提案する．実装例として三角関数 sin 関数と cos 関数の合成関数を実装する．実装した絶対値関数と三角関数の合成関数に対し計算精度評価する．

[1] 早稲田大学

2. ストカスティック数との算術演算

本章ではストカスティック数 (SN) を紹介し，その算術演算回路について議論する．

2.1 ストカスティック数 (SN)

ストカスティックコンピューティング (SC) では，ストカスティック数 (SN) を用いて計算する．SN は，各ビットが 0 と 1 で構成される任意の長さのビット列である．単極表現 (uni-polar expression) の SN u について，ビット長を $|u|$，i 番目のビットを u_i と表す．SN u のビット列の内の 1 の出現回数を S_u とすると，u の値 V_u は 以下の式 (1) で定義される．

$$V_u = P_u = S_u/|u| \tag{1}$$

ここで，P_u は u のビット列中の 1 の出現頻度であり，$0 \leq P_u \leq 1$ が成り立つ．例えば，$u = 01000100$ のとき，その値は $V_u = 0.25$ となる．このように $V_u = P_u$ を満たす表現方法を単極表現と呼ぶ．単極表現では，$0 \leq V_u \leq 1$ が成り立つ．

単極表現の他にも両極表現 (bi-polar expression) がある．両極表現の SN b について，ビット長を $|b|$，i 番目のビットを b_i と表す．SN b のビット列の内の 1 の出現回数を S_b とすると，b の値 V_b は 以下の式 (2) で定義される．

$$V_b = 2 \times P_b - 1 = 2 \times S_b/|b| - 1 \tag{2}$$

ここで，P_b は b のビット列中の 1 の出現頻度であり，$0 \leq P_b \leq 1$ が成り立つ．例えば，$b = 01000100$ のとき，その値は $V_b = -0.5$ となる．このように $V_b = 2 \times P_b - 1$ を満た

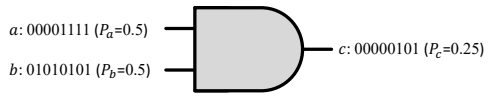

図 1 AND ゲートを用いた単極表現のストカスティック数の乗算の例.

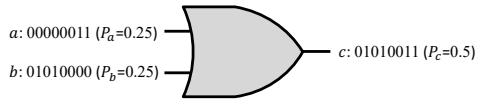

図 2 OR ゲートを用いた単極表現のストカスティック数の加算の例.

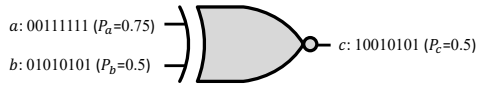

図 3 XNOR ゲートを用いた両極表現のストカスティック数の乗算の例.

す表現方法を両極表現と呼ぶ．両極表現では，$-1 \leq V_b \leq 1$ が成り立つ．

2.2 ストカスティック数を用いた算術演算回路

SC では，論理回路に SN のビット列から 1 ビットずつ順に入力することで算術演算を行う．単極表現では，AND ゲートで乗算を，OR 回路で加算を，NOT ゲートで減算をそれぞれ実装できる [1].

乗算について，入力 SN を a, b，出力 SN を c としたときの V_c は式 (3) のように表される．

$$V_c = P_c = P_a \times P_b = V_a \times V_b \tag{3}$$

図 1 では，$V_a = V_b = 0.5$ である a $(= 00001111)$ と b $(= 01010101)$ を掛けて $V_c = 0.25$ である c $(= 00000101)$ を得ている．

加算について，入力 SN を a, b，出力 SN を c としたときの V_c は式 (3) のように表される．

$$V_c = P_c = P_a + P_b - P_a \times P_b = V_a + V_b - V_a V_b \tag{4}$$

図 2 では，$V_a = V_b = 0.25$ である a $(= 00000011)$ と b $(= 01010000)$ を足して $V_c = 0.5$ である c $(= 01010011)$ を得ている．このように，$V_a \times V_b$ が十分小さくなる場合，正しく加算することができる．

一方，両極表現では，両極表現では XNOR ゲートで乗算を，MUX 回路で加算を，NOT ゲートで減算をそれぞれ実装できる [1]．乗算について，入力 SN を a, b，出力 SN を c としたときの V_c は式 (5) のように表される．

$$\begin{aligned} V_c &= 2 \times P_c - 1 \\ &= 2 \times \{1 - (P_a \times (1 - P_b) \\ &\quad + P_b \times (1 - P_a))\} - 1 \\ &= (2 \times P_a - 1) \times (2 \times P_b - 1) \\ &= V_a \times V_b \end{aligned} \tag{5}$$

図 3 では，$V_a = 0.5$，$V_b = 0$ である a $(= 00111111)$ と b $(= 01010101)$ を掛けて $V_c = 0$ である c $(= 10010101)$ を得ている．

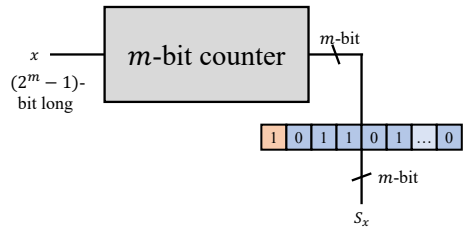

図 4 ストカスティック数を 2 進数への変換する回路.

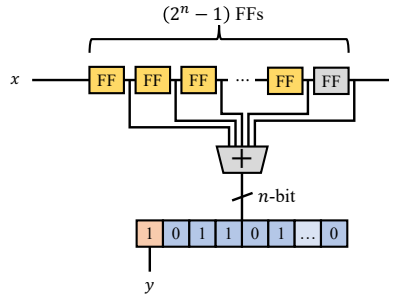

図 5 ストカスティック数を用いたステップ関数の回路 [6].

3. ストカスティック数を用いたステップ関数 [6]

本章では，以前提案した SN を用いたステップ関数を紹介する [6]．ステップ関数はある入力値を境に 0 または 1 を出力する関数である．例えば，0 を閾値にする場合は以下のような関数になる．

$$f(x) = \begin{cases} 1, & (x \geq 0) \\ 0, & (x < 0) \end{cases} \tag{6}$$

SN を入出力として用いたステップ関数を実装する中で正確な出力を得る場合，図 4 の 2 進数の変換の後にその出力を境界値と比較する必要がある．例えば，1 の出現率として 1/2 を境界値にする場合，図 4 のカウンタの MSB (赤色のレジスタ) の値を入力 SN のビット長分出力すれば良い．しかし，このような実装をしてしまうと，SC での利点であるビット毎の演算ができなくなってしまう．

この問題を解決するために SN を用いたステップ関数を提案した [6]．その回路を図 5 に示す．この回路は，図 4 の回路のカウンタ部を FF に置き換えたものである．入力 SN x ビット列のうちの最後に入力された $2^n - 1$ ビットを保持し，その中の 1 の数を数える．そして，その MSB (赤色のビット) を出力 SN y として出力する．

FF に格納されているビットが全て入力 SN x のビットであると仮定した場合出力 SN y のあるビットが 1 になるのは $N(= 2^n - 1)$ 個の FF に格納されているビットの過半数が 1 となる場合である．そのため，入力 SN x のビット列の 1 の出現確率が p であるとすると，出力 SN y の 1 の出現頻度 P_y の期待値 $E(P_y)$ は，

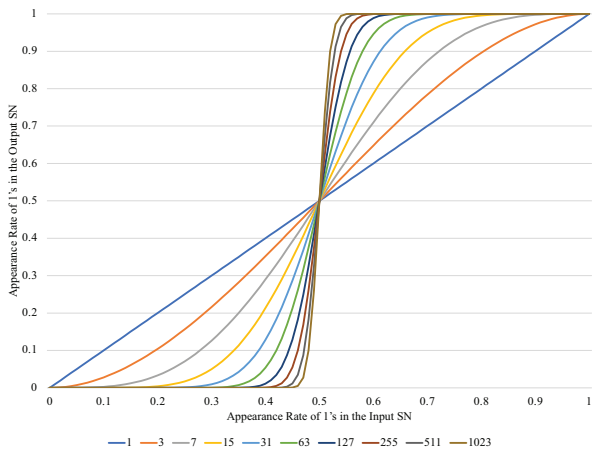

図 6　ステップ関数の回路における入出力ストカスティック数の 1 の出現頻度の期待値の関係.

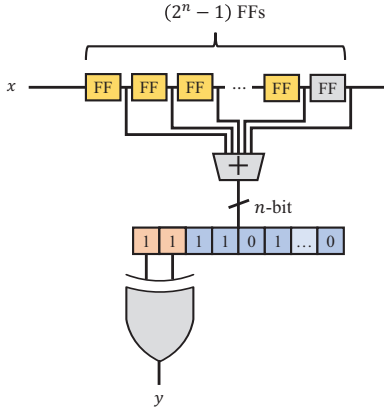

図 7　ストカスティック数を用いた閾値を変更したステップ関数の回路.

$$E(P_y) = \sum_{i=\frac{N+1}{2}}^{N} ({}_N\mathrm{C}_i \times p^i (1-p)^{N-i}) \tag{7}$$

となる．図 6 に入力 SN x のビット列の 1 の出現確率 p を横軸に，出力 SN y の 1 の出現頻度の期待値 $E(P_y)$ を縦軸にしたグラフを示す．このグラフでは，$n = 1,\ 2,\ 3,\ \ldots,\ 10$, つまり $N = 1,\ 3,\ 7,\ 15,\ 31,\ 63,\ 128,$
$255,\ 511,\ 1023$ の場合を示している．このグラフから，n および N が大きくなれば大きくなるほどステップ関数に近づく．その式は以下のように表される．

$$P_y = \begin{cases} 1, & (P_x \geq \frac{1}{2}) \\ 0, & (P_x < \frac{1}{2}) \end{cases} \tag{8}$$

ここで，出力 SN y として出力するビットを変えると，異なる関数が実装できる．例えば，上位 2 ビットを XOR ゲートに入力した出力を出力する場合 (図 7)，その期待値は図 8 となる．このグラフでは，$n = 2,\ 3,\ \ldots,\ 10$，つまり $N = 3,\ 7,\ 15,\ 31,\ 63,\ 127,\ 255,\ 511,\ 1023$ の場合を示している．その式は以下のように表される．

$$P_y = \begin{cases} 1, & (\frac{1}{4} < P_x \leq \frac{3}{4}) \\ 0, & (otherwise) \end{cases} \tag{9}$$

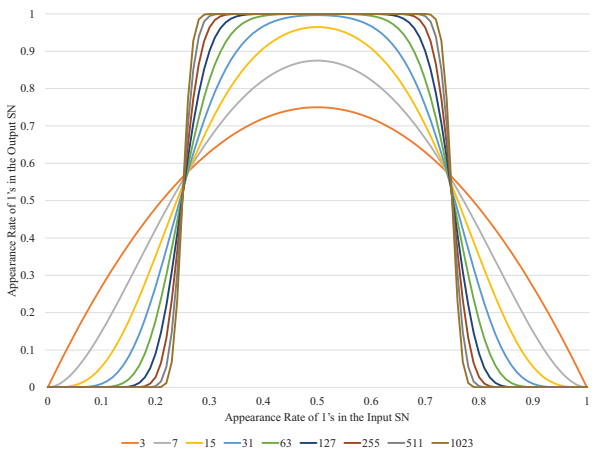

図 8　閾値を変更したステップ関数の回路における入出力ストカスティック数の 1 の出現頻度の期待値の関係.

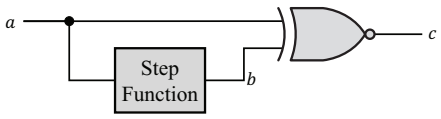

図 9　両極表現で表されたストカスティック数の絶対値を計算する回路.

4. ストカスティック数を用いた絶対値関数

本章では SN を用いたステップ関数の回路を既存の回路と組み合わせた絶対値関数の回路を提案する．

4.1 提案手法

ここでは式 (2) の両極表現で表された SN を用いて絶対値を算出する回路を提案する．SN には 2.1 節に示したように $V_x = 2 \times P_x - 1$ で定義される両極表現があり，SC では両極表現においても論理回路に SN のビット列から 1 ビットずつ順に入力することで算術演算を行える．

図 3 の XNOR ゲートと図 5 のステップ関数の回路によって両極表現で表された SN の絶対値をとることができる．図 9 のように，SN a をステップ関数の回路に入力し (ステップ関数回路の出力 SN を b とする)，a と b を XNOR 回路に入力して出力 SN c を得るとすると，それらの値 V_b, V_c は，以下の式のように表される．

$$\begin{aligned} V_b &= 2P_b - 1 \\ &= \begin{cases} 2 \times 1 - 1, & (P_a \geq \frac{1}{2}) \\ 2 \times 0 - 1, & (P_a < \frac{1}{2}) \end{cases} \\ &= \begin{cases} 1, & (V_a \geq 0) \\ -1, & (V_a < 0) \end{cases} \end{aligned} \tag{10}$$

表 1　ストカスティック数を用いた絶対値関数の回路の回路面積.

N [FFs]	Area [NANDs]
1	13.75
3	75
7	166
15	305

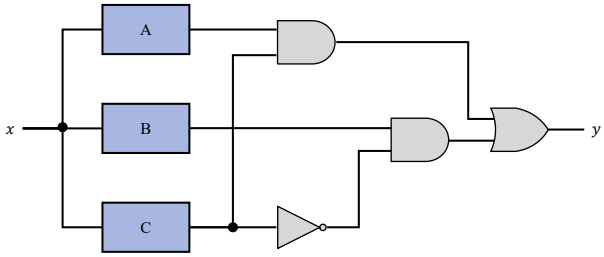

図 10　ストカスティック数の不連続関数を計算する回路.

$$
\begin{aligned}
V_c &= V_a \times V_b \\
&= \begin{cases} V_a \times 1, & (V_a \geq 0) \\ V_a \times -1, & (V_a < 0) \end{cases} \\
&= \begin{cases} V_a, & (V_a \geq 0) \\ -V_a, & (V_a < 0) \end{cases} \\
&= |V_a| \qquad (11)
\end{aligned}
$$

このように，XNOR ゲートとステップ関数の回路を組み合わせることによって両極表現で表された SN の絶対値を算出することができる．

4.2　回路面積

論理合成によって求めた SN を用いた絶対値関数の回路の回路面積を表 1 に示す．論理合成は Design Compiler version D-2010.03-SP5 と STARC 90nm ライブラリを用いて行った．合成できた回路について，ステップ関数の回路に加算器が不要な $N = 1$ の場合を除き，NAND ゲート換算で約 $20N$ 個になった．

5.　ストカスティック数を用いた不連続関数

本章では SN を用いたステップ関数の回路を既存の回路と組み合わせた不連続関数の回路を提案する．

5.1　提案手法

ストカスティック数の不連続関数を計算する回路として図 10 に示す回路を提案する．提案回路の前段には A, B, C のストカスティック回路を配置し，後段にはマルチプレクサと同型の回路を配置する．A, B は任意のストカスティック回路を実装可能である．C は任意の閾値のステップ関数 [6] を実装する．提案した回路は SN x を入力として SN y を出力する．

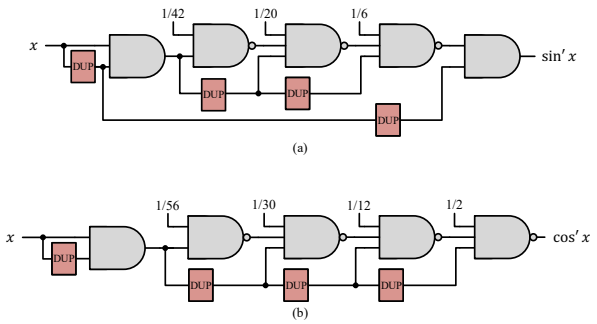

図 11　複製器を用いた関数の実装. (a) sin' 関数　(b) cos' 関数.

5.2　前提となる回路

ここでは，図 7 のステップ関数の回路以外に，[2] で提案した複製器によって実装した回路を前提として用いる．図 11(a) では式 (12) で表される sin の近似関数を実装している．

$$
\begin{aligned}
\sin x &\approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \\
&= x(1 - \frac{x^2}{6}(1 - \frac{x^2}{20}(1 - \frac{x^2}{42}))) \\
&= \sin' x \qquad (12)
\end{aligned}
$$

なお，図 11(a) 中の DUP は図 12 に示す 2^nRRR 複製器である．図 12(a) の In，Out がそれぞれ入力，出力であり，RU_m は図 12(b) に示されるレジスタユニットである．

図 11(b) では式 (13) で表される cos の近似関数を実装している．

$$
\begin{aligned}
\cos x &\approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \\
&= 1 - \frac{x^2}{2}(1 - \frac{x^2}{12}(1 - \frac{x^2}{30}(1 - \frac{x^2}{56}))) \\
&= \cos' x \qquad (13)
\end{aligned}
$$

図 11(b) 中の DUP も同様に図 12 に示す 2^nRRR 複製器である．

5.3　提案手法の実装例

提案手法の実装例として, 図 10 の A に cos 関数, B に sin 関数, C に 1/4, 3/4 を閾値とするステップ関数を実装する．ここでは式 (7) で表されたステップ関数と，式 (8)，(9) の sin や cos の近似関数を用いて不連続関数を算出する回路を提案する．

前項の回路によって不連続関数を実装することができる．図 13 のように，SN x をステップ関数の回路に入力することを考える．ここで，a–f は各回路・ゲートの出力 SN である．それら SN の値は，以下の式のように表される．

$$V_a = \cos'(V_x) \qquad (14)$$

$$V_b = \sin'(V_x) \qquad (15)$$

$$V_c = \begin{cases} 1, & (\frac{1}{4} < V_x \leq \frac{3}{4}) \\ 0, & (otherwise) \end{cases} \qquad (16)$$

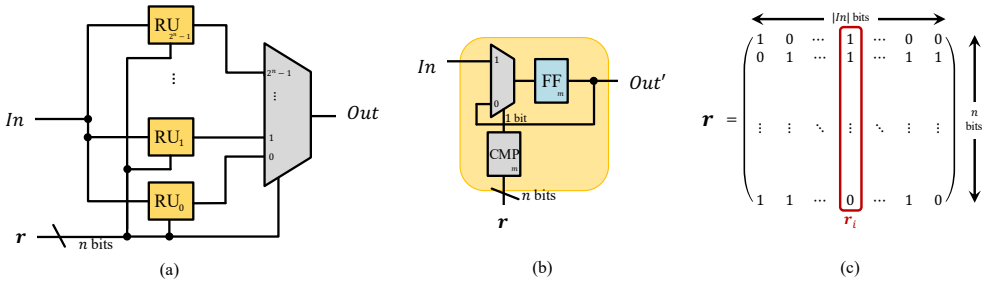

図 12　2^nRRR 複製器. (a) 全体のモデル. (b) m 番目のレジスタユニット RU_m. (c) 幅 n ビット，長さ $|In|$ ビットのランダムビット列 $\boldsymbol{r}$.

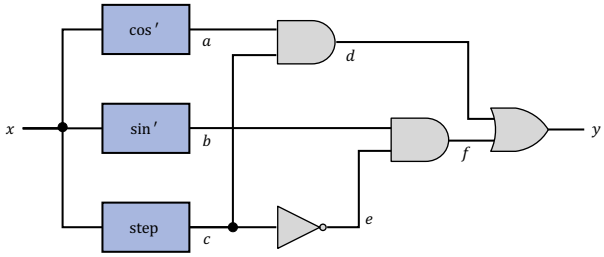

図 13　ストカスティック数の不連続関数を計算する回路 (実装例).

$$V_d = V_a \times V_c = \begin{cases} \cos'(V_x), & (\frac{1}{4} < V_x \leq \frac{3}{4}) \\ 0, & (otherwise) \end{cases} \tag{17}$$

$$V_e = 1 - V_c = \begin{cases} 0, & (\frac{1}{4} < V_x \leq \frac{3}{4}) \\ 1, & (otherwise) \end{cases} \tag{18}$$

$$V_f = V_b \times V_e = \begin{cases} 0, & (\frac{1}{4} < V_x \leq \frac{3}{4}) \\ \sin'(V_x), & (otherwise) \end{cases} \tag{19}$$

$$V_y = V_d + V_f - V_d \times V_f = \begin{cases} \cos'(V_x) + 0 - \cos'(V_x) \times 0, & (\frac{1}{4} < V_x \leq \frac{3}{4}) \\ 0 + \sin'(V_x) - 0 \times \sin'(V_x), & (otherwise) \end{cases} = \begin{cases} \cos'(V_x), & (\frac{1}{4} < V_x \leq \frac{3}{4}) \\ \sin'(V_x), & (otherwise) \end{cases} \tag{20}$$

このように，既存の回路とステップ関数の回路を組み合わせることによって両極表現で表された SN の不連続関数を算出することができる．

5.4　回路面積

論理合成によって求めた SN を用いた不連続関数の回路の回路面積を表 2 に示す．論理合成は 4.2 節と同様に Design Compiler version D-2010.03-SP5 と STARC 90nm ライブラリを用いて行った．本結果から，ステップ関数の FF 一つ毎に約 20NAND ゲート分の面積が増えること，複製器によっては面積が大きく変化することがわかった．

表 2　ストカスティック数を用いた不連続関数の回路面積 (NAND ゲート換算).

N [FFs]	1RRR	2RRR	4RRR	8RRR
3	2082	2188	2956	3920
7	2173	2279	3047	4011
15	2312	2418	3186	4150

6.　評価実験

本章では SN を用いた絶対値関数の回路のシミュレーションを行い，評価する．

6.1　実験方法

SN を用いた絶対値関数の回路のシミュレーションを行い，評価する．$N = 1, 3, 7, 15, 31, 63, 128, 255, 511, 1023$ に対して以下の条件で実験を行う．(ただし，不連続関数は設計上 $N = 1$ の回路は存在しえない)

- 実行環境: Python 3.6.3
- 実装回路: 図 9 の絶対値関数の回路，図 13 の不連続関数の回路
- 目標関数: $V_y = \begin{cases} V_x, & (V_x \geq 0) \\ -V_x, & (V_x < 0) \end{cases}$,
  $V_y = \begin{cases} \cos'(V_x), & (\frac{1}{4} < V_x \leq \frac{3}{4}) \\ \sin'(V_x), & (otherwise) \end{cases}$
- 入力 SN の 1 の出現頻度: 0 から 1 まで，0.01 刻み
- 入力 SN のビット長: 10000 ビット
- 出力 SN のビット長: 10000 ビット
- FF の初期値: 順に 1010...01 と格納
- 評価指標: 出力の目標関数に対する MSE

ここで，MSE (Mean Square Error) の値は以下の式で表される．

$$MSE = \frac{1}{101} \sum_{i=0}^{100} \left(f_{theor}(\frac{i}{100}) - f_{actual}(\frac{i}{100}) \right)^2 \tag{21}$$

$f_{theor}(\frac{i}{100})$, $f_{actual}(\frac{i}{100})$ はそれぞれ目標関数に $i/100$ を代入した場合の出力の値と，実装回路に値が $i/100$ の SN を入力した場合の出力 SN の値である．

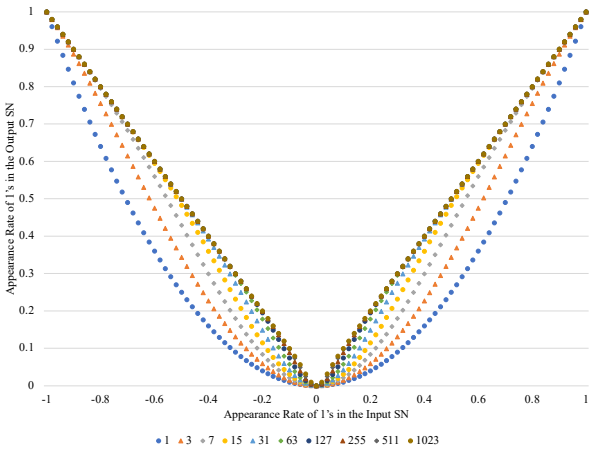

図 14 絶対値関数の回路の入出力ストカスティック数の関係.

表 3 絶対値関数の回路の出力の MSE.

N [FFs]	MSE
1	3.30×10^{-2}
3	1.34×10^{-2}
7	5.05×10^{-3}
15	1.85×10^{-3}
31	6.64×10^{-4}
63	2.37×10^{-4}
128	8.41×10^{-5}
255	2.98×10^{-5}
511	1.05×10^{-5}
1023	3.66×10^{-6}

6.2 実験結果

図 9 の絶対値関数の回路に 10 通りの N で 101 通りの P_x を持つ SN x 入力した際の出力を図 14 に示す．この図から N が大きくなるにつれて絶対値関数に近付くことがわかる．また，絶対値関数の回路の出力の，目標関数に対する MSE を表 3 に示す．この表からも N が大きくなるにつれて絶対値関数に近付くことがわかる．

図 13 の不連続関数の回路に 9 通りの N，4 通りの複製器で 101 通りの P_x を持つ SN x 入力した際の出力の，目標関数に対する MSE を表 4 に示す．N が大きくなるにつれて目標関数に近付くことがわかる．本実験条件下では 2RRR または 4RRR 複製器が最も優れた結果となった．ただ，複製器による誤差は小さく，ステップ関数の精度のほうがより重要であることがわかった．図 15 に，最も MSE の小さかった $N = 1023$ で，複製器は 2RRR の際の出力を示す．この図から入力が 0.25，0.75 の際に誤差が大きくなることがわかる．

7. おわりに

本稿では，SN を用いた絶対値関数及び不連続関数の実装法を提案し，評価した．今後は，ステップ関数の絶対値関数以外の応用先を模索するとともに，ニューラルネットワークなど実用的な回路に実装，評価したい．

表 4 不連続関数の回路の出力の MSE.

N [FFs]	1RRR	2RRR	4RRR	8RRR
3	4.75×10^{-2}	4.32×10^{-2}	4.11×10^{-2}	4.44×10^{-2}
7	2.85×10^{-2}	2.68×10^{-2}	2.58×10^{-2}	2.71×10^{-2}
15	1.77×10^{-2}	1.70×10^{-2}	1.65×10^{-2}	1.71×10^{-2}
31	1.20×10^{-2}	1.12×10^{-2}	1.10×10^{-2}	1.18×10^{-2}
63	7.96×10^{-3}	7.33×10^{-3}	7.26×10^{-3}	7.83×10^{-3}
127	5.15×10^{-3}	4.77×10^{-3}	4.73×10^{-3}	5.19×10^{-3}
255	3.26×10^{-3}	2.96×10^{-3}	2.99×10^{-3}	3.19×10^{-3}
511	1.99×10^{-3}	1.77×10^{-3}	1.81×10^{-3}	1.95×10^{-3}
1023	1.09×10^{-3}	9.90×10^{-4}	1.02×10^{-3}	1.11×10^{-3}

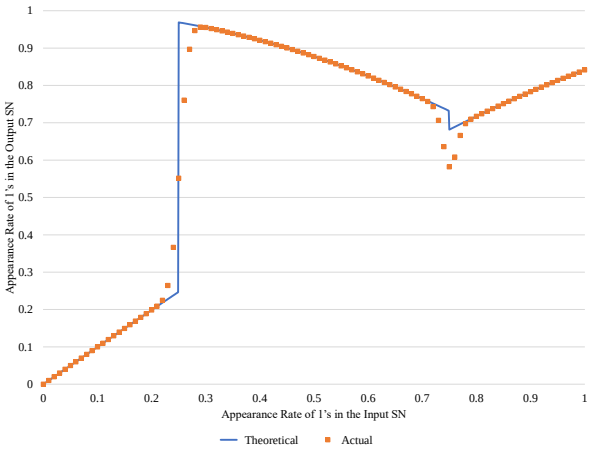

図 15 不連続関数の回路の入出力ストカスティック数の関係.

謝辞

本研究は一部，JSPS 科研費 (19H04080) の助成を受けたものです．

参考文献

[1] Gaines, B. R.: Stochastic Computing, *Proc. Spring Joint Computer Conference*, pp. 149–156 (1967).
[2] Ishikawa, R., Tawada, M., Yanagisawa, M. and Togawa, N.: Scalable Stochastic Number Duplicators for Accuracy-flexible Arithmetic Circuit Design, *IPSJ Transactions on System LSI Design Methodology (TSLDM)*, Vol. 13, pp. 10–20 (2020).
[3] Li, B., Qin, Y., Yuan, B. and Lilja, D. J.: Neural Network Classifiers Using Stochastic Computing with a Hardware-Oriented Approximate Activation Function, *Proc. International Conference on Computer Design (ICCD)*, pp. 97–104 (2017).
[4] Li, J., Yuan, Z., Li, Z., Ding, C., Ren, A., Qiu, Q., Draper, J. and Wang, Y.: Hardware-driven nonlinear activation for stochastic computing based deep convolutional neural networks, *Proc. International Joint Conference on Neural Networks (IJCNN)*, pp. 1230–1236 (2017).
[5] Yeo, I., Gi, S., Lee, B. and Chu, M.: Stochastic implementation of the activation function for artificial neural networks, *Proc. Biomedical Circuits and Systems Conference (BioCAS)*, pp. 440–443 (2016).
[6] 石川遼太, 多和田雅師, 柳澤政生, 戸川望: ストカステイック数を用いたステップ関数の実装と評価, 電子情報通信学会技術研究報告, Vol. 119, No. 282, pp. 69–74 (2019).