一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
VLD2023-81,RECONF2023-84(2024-01)

# AFEストカスティック演算器における出力ビット幅増大の抑制手法

瀬戸　大暉†　　藤枝　直輝†

† 愛知工業大学　工学部　〒470–0392 愛知県豊田市八草町八千草 1247
E-mail: †{v20089vv,nfujieda}@aitech.ac.jp

**あらまし**　Stochastic Computing (SC) は，画像処理や機械学習といった分野への応用が期待されている．SC の新たな符号化方式に AFE (Amplitude and Frequency Encoding) があるが，演算回路の多段化に伴い出力ビット幅が増大していく欠点がある．本稿では，F-Corrector と S-Corrector という 2 種類の補正回路を提案し，出力ビット幅の増大を抑制した新しい AFE SC 演算器を実装する．PYNQ-Z1 ボードを用いた実機評価の結果，新しい AFE SC 演算器の演算誤差は，従来の AFE SC 演算器と同等かそれ未満であることが確認された．

**キーワード**　ディジタル回路設計，ストカスティックコンピューティング，AFE (Amplitude and Frequency Encoding)，FPGA (Field-Programmable Gate Array)

# Suppression of output bit width growth in AFE stochastic computing units

Daiki SETO† and Naoki FUJIEDA†

† Faculty of Engineering, Aichi Institute of Technology　1247 Yachigusa, Yakusa-cho, Toyota-shi, 470–0392 Japan
E-mail: †{v20089vv,nfujieda}@aitech.ac.jp

**Abstract**　Stochastic Computing (SC) is expected to be applied to fields such as image processing and machine learning. Amplitude and Frequency Encoding (AFE) is a new encoding method for SC. However, it has a drawback that the output bit width grows when cascading AFE SC operators. This paper proposes two types of correction circuits, F-Corrector and S-Corrector, and implements new AFE SC operators that suppress the growth of output bit-width. The results of the evaluation using the PYNQ-Z1 board showed that the arithmetic error of the proposed AFE SC operators was the same as or less than that of the conventional AFE SC operators.

**Key words**　Digital Circuit Design, Stochastic Computing, AFE (Amplitude and Frequency Encoding), FPGA (Field-Programmable Gate Array)

## 1. はじめに

2012 年に AlexNet [1] が発表されてから，画像処理や機械学習といった分野への注目が高まっている．近年の機械学習は，脳のニューロンを模した計算ユニットからなる層を多数並べた，深層ニューラルネットワークが主流となっている．これを FPGA (Field-Programmable Gate Array) で実装するためのフレームワークとして，DPU (Vitis AI) [2] や FINN [3] が挙げられる．また，深層学習による画像認識の前処理や後処理には，フィルタなどの様々な画像処理が用いられる．これらの問題点として，ハードウェア規模が大きいことが挙げられる．

そこで，Stochastic Computing (SC) [4] の画像処理や機械学習分野への応用が期待される．SC は，ビットストリーム内の “1” の割合を用いて数値を表現する疑似アナログ演算である．SC の利点は，演算回路のハードウェア規模を小さくできることである．例えば，確率 $p_A$ で “1” になる信号 A と確率 $p_B$ で “1” になる信号 B の AND 演算をすると，確率 $p_A p_B$ で “1” になる信号が得られる．つまり，乗算が AND ゲート 1 つで実現できる．実際に，SC により画像処理演算を極めて小さなハードウェア規模で実現できることが示されている [5]．更に，ニューラルネットワークを SC で実装することもできる [6].

SC の欠点を克服するための研究も行われている．SC の欠点の 1 つは，乱数を用いて演算を行うために，演算結果にある程度の誤差が生じることである．他の欠点としては，精度を高めるために十分な長さのビットストリームが必要なため，計算に時間がかかることが挙げられる．Sim ら [7] は，演算時間を従来の方法から 50% 削減する手法を提唱している．また，精度を向上させるために Sobol’ 列を用いる手法 [8] や，疑似乱数の周期性を利用した決定論的手法 [9] が提唱されている．

加えて，近年 AFE (Amplitude and Frequency Encoding) [10] という新たな符号化方式が提案された．通常の SC では，1 つの値を表現するのに 1 ビットの信号を用いる．一方で AFE で

は，1 つの値を表現するのに 2 ビット以上の短いビット幅をもつ固定小数点の信号を用いる．演算は通常の 2 進数の演算をその短いビット幅の信号で行う．AFE は，精度に対するハードウェア規模と演算時間の効率が良いと主張されている [10]．しかし，従来の SC と異なり演算の出力ビット幅が入力ビット幅よりも大きい．このため，演算回路の多段化により信号のビット幅が増大していく．これは，SC の利点である演算回路のハードウェア規模の小ささを損なうことになりうる．

本研究の目的は，誤差が可能な限り小さく，なおかつハードウェア規模の小さい SC 演算回路を実現することである．AFE をベースとして，演算回路の出力ビット幅が入力ビット幅より大きくならないように設計する．具体的には，演算回路を FPGA の Look-Up Table (LUT) に置き換えて，真理値表に基づいて出力を決定する．更に，桁あふれや端数処理に伴う誤差を軽減するため，S-Corrector と F-Corrector という 2 種類の補正回路を提案する．本稿では 2 ビット 4 入力，2 ビット出力の平均演算回路と乗算回路を評価対象とする．提案した回路を FPGA SoC に実装し，Digilent 社の PYNQ-Z1 ボード上で実機評価をする．評価は，本来の計算結果との誤差の絶対値と，演算回路で使用された論理素子の個数によって行う．

## 2. 研 究 背 景

### 2.1 Stochastic Computing

SC では，ビットストリーム内の “1” の割合を用いて数値を表現する．1 つの値を表現するために 1 本のディジタル信号を用いる．その信号の $L$ クロックサイクル間の値を観測すると $L$ ビットのビットストリームが得られる．これを Stochastic Bitstream (SB) と呼ぶ．従来型 SC ではビットの重みは 1 である．また，本研究で対象とするユニポーラ形式で扱う値の範囲は，$[0, 1]$ である．ストリーム長 $L$ のストカスティックストリームの精度は，$1/L$ である．ここで，従来型 SC において，ビットストリーム内の任意の値 $x(i)$ は，$p$ を確率変数として，$P(X = 1) = p, P(X = 0) = 1 - p$ の確率分布をもつ離散確率変数 $X$ で生成される．$X$ の期待値は，

$$E[X] = 1 \cdot p + 0 \cdot (1 - p) = p \tag{1}$$

となる．つまり，$X$ の期待値 $E[X]$ は，SB の実数値を表していると考えることができる．

従来型 SC の演算回路を図 1 に示す．あるクロックサイクルで信号 A, B, S, Q が “1” となる確率をそれぞれ $p_A, p_B, p_S, p_Q$ とする．従来型 SC の場合，乗算回路は AND ゲートで実現する．出力信号 Q は，A と B がともに “1” であるとき，“1” を出力する．よって，Q が “1” となる確率は，

$$p_Q = p_A p_B \tag{2}$$

と表せる．また，2 つの入力の積が和よりも十分に小さいとき，加算回路は OR ゲートで近似的に実現できる．出力信号 Q は，A と B のいずれかが “1” であるとき，“1” を出力する．A と B のいずれかが “1” である確率は，それぞれの入力信号が “1” となる確率の和から，入力信号がともに “1” となる確率を

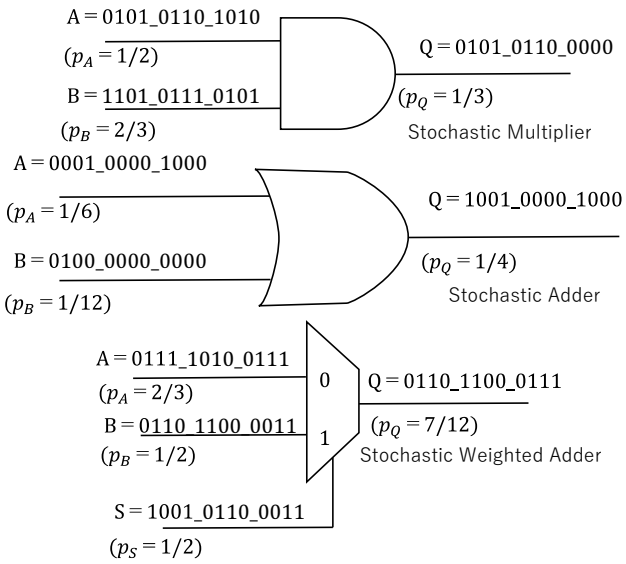


図 1　SC 演算回路の例

Fig. 1　Examples for arithmetic circuits for SC

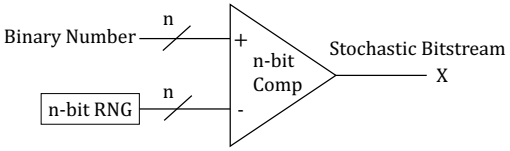


図 2　従来型 SC の SB 生成器

Fig. 2　SB Generator for Conventional SC

引けばよいので，

$$p_Q = p_A + p_B - p_A p_B \approx p_A + p_B \quad (p_A p_B \ll p_A + p_B) \tag{3}$$

と表せる．更に，重み付き加算回路は，マルチプレクサ (MUX) で実現する．この回路で出力信号 Q は，確率 $p_S$ で B を，確率 $1 - p_S$ で A を出力する．出力信号 Q が “1” となる確率は，

$$p_Q = (1 - p_S)p_A + p_S p_B \tag{4}$$

となる．このとき，$p_S = 0.5$ とすることで 2 入力の平均演算を行う．

SB の生成は，図 2 に示す Linear Feedback Shift Register (LFSR) などの乱数生成器 (RNG) と比較器を用いる．入力を小数部 $n$ ビットの 2 進数とする．入力と $n$ ビットの乱数を比較し，乱数の方が小さい場合には X を “1” とする．こうして入力と等確率で “1” を出力することで SB を生成する．また，SB から 2 進数への変換はアップカウンタを用いる．図 3 に SB から 2 進数への変換器を示す．最後にこのカウンタの値をストリーム長 $L$ で割ることで 2 進数への変換を行う．

ここまで，SC の変換器や演算回路について述べた．しかし，SC は確率演算に基づいて行われるため，それぞれの信号が確率的に独立である必要がある．例えば，図 1 の AND ゲートの例において，信号間に正の相関があり，A が “1” であるときは B も必ず “1” になると仮定する．このとき出力 Q は，A が “1” のときに “1” となる．つまり，A の信号がそのまま出力され，乗算器としては機能しない．

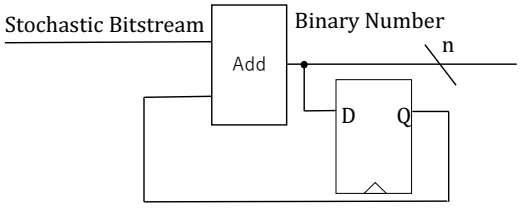


図 3　SB から 2 進数への変換器

Fig. 3　SB to Binary Number Coverter

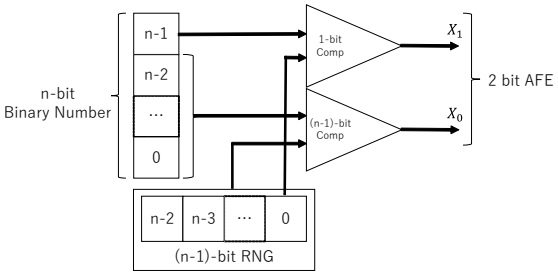


図 4　2 ビット AFE の SB 生成器 (Type-A) [10]

Fig. 4　SB Generator for 2-bit AFE (Type-A) [10]

### 2.2　Amplitude and Frequency Encoding

Amplitude and Frequency Encoding (AFE) [10] は，SC の新しい符号化方式である．2 本以上のディジタル信号を使って 1 つの数値を表現する．ここで，$N$ 本のディジタル信号を使った AFE を $N$ ビット AFE と定義する．また，本研究では $N$ ビット AFE の $k$ ビット目（LSB を 0 ビット目とする）の重みを $2^{-(N-1-k)}$ であるとして扱う．MSB の重みは常に 1 である．例えば 2 ビット AFE の LSB の重みは，0.5 である．式 (1) において従来型 SC では，$X \in \{1, 0\}$ である．一方で，AFE は $X$ が 1 と 0 以外の離散値も取りうるように拡張する．加算および乗算は $X$ と $Y$ が互いに独立であれば期待値の性質により，

$$E[X+Y] = E[X] + E[Y] \tag{5}$$

$$E[X \cdot Y] = E[X] \cdot E[Y] \tag{6}$$

となるため，2 進数演算の加算器・乗算器で実現できる．

AFE のビットストリームの生成方法は Type-A と Type-B の 2 種類がある．Type-A の 2 ビット AFE 生成器を図 4 に示す．入力 2 進数の下位 $(n-1)$ ビットと $(n-1)$ ビット RNG の出力を比較し，AFE の 0 ビット目を得る．入力データの MSB と RNG の任意の 1 ビットを比較し，AFE の 1 ビット目を得る．MSB は入力が 0.5 より小さいとき常に “0” となり，0.5 より大きいときは確率 0.5 で “1” となる．LSB は入力が 0.5 より小さいときは入力の実数値の 2 倍と同じ確率で “1” を出力し，0.5 より大きいときは，入力の実数値の 2 倍から 1 を引いた確率で “1” を出力する．本研究では，Type-A の SB 生成器を用いる．

ここで，AFE の出力ビット幅について考える．AFE の入力を整数部が 1 ビット，小数部が 1 ビットの固定小数点数とする．2 入力の積は最大で $1.1_2(1.5) \times 1.1_2(1.5) = 10.01_2(2.25)$ となる．ここで出力は整数部が 2 ビット，小数部が 2 ビットの計 4 ビットとなる．よって，乗算を行うたびに出力ビット幅が増大していくことがわかる．

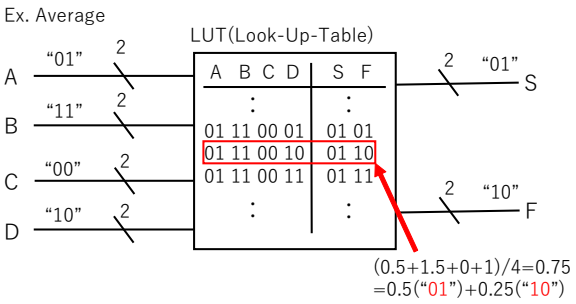


図 5　2 ビット 4 入力平均演算 LUT

Fig. 5　2-bit 4-input averaging LUT

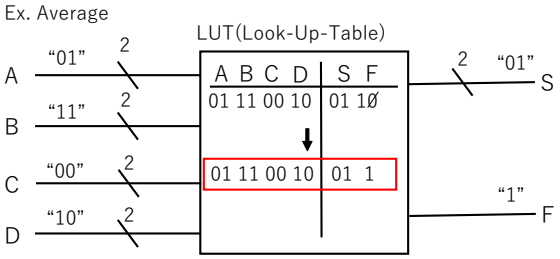


図 6　2 ビット 4 入力平均演算 LUT における F の切り詰め

Fig. 6　Truncation of F on 2-bit 4-input averaging LUT

## 3.　提 案 手 法

本研究では，4 入力の平均演算回路と乗算回路を対象に，演算回路の実装を行う．入力を $[0, 1]$ の実数値 $a, b, c, d$ としたとき，平均演算回路では $(a+b+c+d)/4$ を，乗算回路では $abcd$ を求める．演算回路の入力は，MSB が 1，LSB が 0.5 の重みとなる 2 ビット AFE とする．平均演算の最大値は，$(1.1_2(1.5) + 1.1_2(1.5) + 1.1_2(1.5) + 1.1_2(1.5))/4 = 1.1_2(1.5)$ である．取りうる小数部の値は実数値で，0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875 である．つまり，従来の AFE における平均演算回路では，整数部が 1 ビット，小数部が 3 ビット必要となる．また，乗算の最大値は，$1.1_2(1.5) \times 1.1_2(1.5) \times 1.1_2(1.5) \times 1.1_2(1.5) = 101.0001_2(5.0625)$ である．取りうる小数部の値は実数値で，0, 0.0625，0.125，0.1875，0.25，0.375，0.5，0.5625，0.75，0.6875 である．つまり，従来の AFE における乗算回路では，整数部が 3 ビット，小数部が 4 ビット必要となる．

提案手法では，FPGA の LUT を用いて AFE の演算を行う．LUT は，S (Significant digits) と F (Fraction) の 2 つの出力をもつ．S は整数部が 1 ビット以上，小数部が 1 ビットとする．F は S で表現することができない端数を表す．平均演算回路に対する，従来の AFE と等価な LUT 実装を，図 5 に示す．平均演算回路では整数部 1 ビット，小数部 3 ビットが必要であるため，出力は S，F ともに 2 ビットとなる．F のビット幅については，丸めを行うことでより小さい幅を選択できる．図 6 に，平均演算回路の LUT 実装において F を 1 ビット幅にした場合を示す．F は切り捨てた MSB が “1” ならば切り上げ，“0” ならば切り捨てを行い丸める．これにより，F は $p_F \in \{0, 1, 2\}$ ビットで実装が可能となる．ここで $p_F$ は，端数情報の精度で

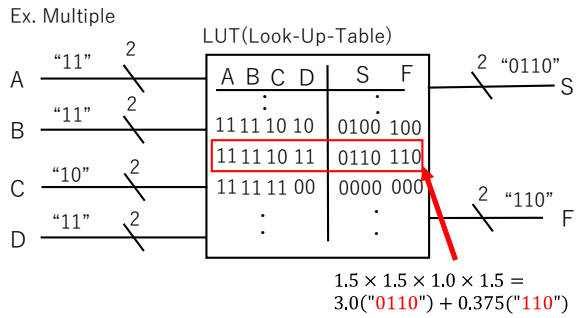


図 7　2 ビット 4 入力乗算 LUT

Fig. 7　2-bit 4-input multiplier LUT

あり，増減することで演算精度とハードウェア規模の制御が可能となる．図 7 に，乗算回路に対する AFE の LUT 実装を示す．乗算回路では整数部 3 ビット，小数部 4 ビットが必要であるため，S は 4 ビット幅であり，F は $p_F \in \{0,1,2,3\}$ ビットで実装可能である．

次に，提案手法では，S と F から 2 ビットの AFE 出力を生成するため，LUT の後段に補正回路を加える．図 8 に，本研究で提案する AFE SC 演算回路を示す．補正回路は，S-Corrector と F-Corrector の 2 種類からなる．F-Corrector は LUT から出力された端数情報 F を積算する．積算値が実数値で 0.5 以上のとき，後述する S-Corrector に “1” を出力し，積算値から実数値で 0.5 を引く．S-Corrector は LUT の出力 S と F-Corrector から渡された端数の桁上げ，前回までの加算の余りを飽和加算する．加算の余りはレジスタに記憶し，次のクロックサイクルで飽和加算の入力として与える．レジスタのビット幅は可変とし，パラメータ $p_S$ とする．$p_S$ は $p_F$ と同様，増減することで演算精度とハードウェア規模の制御が可能となる．S-Corrector から出力する 2 ビット AFE を，演算回路全体の最終的な出力とする．例えば，LUT の出力 S が $01_2$（実数値で 0.5），端数の桁上げが $1_2$ (0.5)，前回までの加算の余りは $10_2$ (1.0) であるとする．この場合，飽和加算 $1_2 + 1_2 + 10_2 = 100_2 = 11_2 + 1_2$ により，AFE 出力として $11_2$ (1.5) を出力し，加算の余りとして $1_2$ (0.5) をレジスタに保存する．

図 8 では，提案する演算回路を 4 つの場合に分けて示している．$p_F = 0$ のとき LUT は S のみを出力し，S-Corrector での飽和加算における端数の桁上げは常に “0” とする．$p_S = 0$ のとき S-Corrector は飽和加算のみを行い，加算の余りの記憶を行わない．$p_F = p_S = 0$ のとき，飽和加算の機能は LUT に内包する．すなわち，F-Corrector，S-Corrector を使用せず，LUT で S のみ上位ビットを切り詰め，2 ビットで出力をする．切り詰めた際に上位ビットに “1” が 1 つ以上含まれていれば，出力は “11” とする．

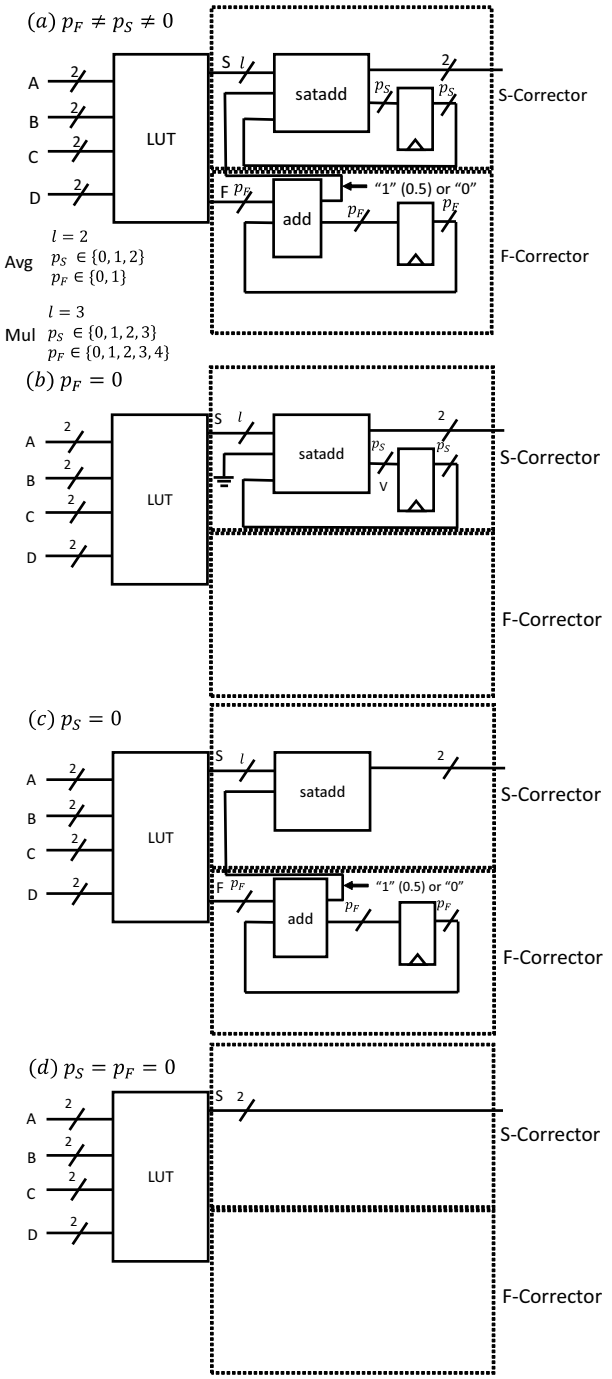


図 8　補正回路付き AFE 演算回路

Fig. 8　Arithmetic circuit on AFE with corrector

## 4. 評　　価

### 4.1 評 価 方 法

評価は，PYNQ-Z1 ボード (FPGA: Zynq XC7Z020) を用いる．PYNQ プラットホーム上の Jupyter Notebook から PL (Programmable Logic) の制御を行う．入力として $[0, 1)$ の乱数を与える．ストリーム長は 10,000 とし，測定回数は 10,000 回とする．また，演算器の入力と出力はともに 2 ビットの AFE である．入力はライブラリ PYNQ-BitPack [11] の改良版にて正規化を行う．対象とする演算は，平均演算と乗算である．平均演算では，$p_F \in \{0,1,2\}$ ビット（精度は実数で，0.5，0.25，0.125），$p_S \in \{0,1\}$ ビットで検証する．乗算では，$p_F \in \{0,1,2,3\}$ ビット（精度は実数で，0.5，0.25，0.125，0.0625），$p_S \in \{0,1,2,3,4\}$ ビットで検証する．誤差の絶対値 (MAE: Mean Absolute Error) で性能を評価する．MAE は以下の式により定義する．

$$MAE = \frac{\sum|(SC \text{ の演算結果}) - (\text{本来の演算結果})|}{(\text{測定回数})} \qquad (7)$$

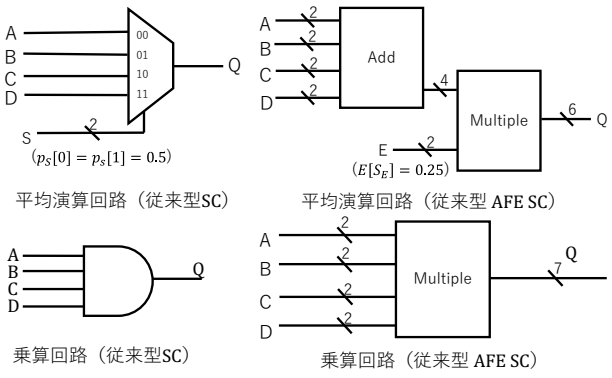


図 9　比較対象とする従来の SC，AFE SC 乗算，平均演算回路

Fig. 9　Conventional SC and AFE SC multiplier and averaging circuits for comparison

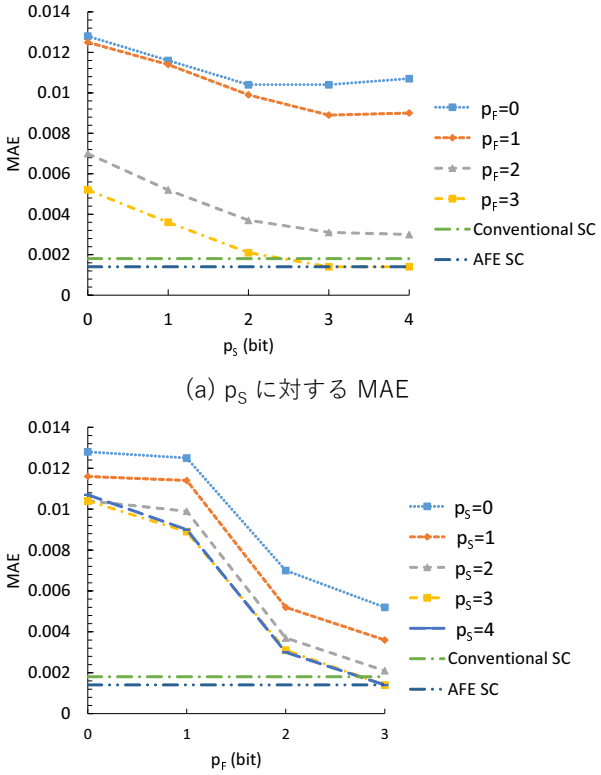


図 10　乗算回路の MAE

Fig. 10　MAE of multiplier

比較対象として，従来型 SC と 従来型 AFE SC の乗算，平均演算回路を使用する．それぞれの回路図を図 9 に示す．また，演算回路を Vivado 2020.2 で論理合成を行ったときの LUT 利用数を記録する．

### 4.2　評 価 結 果

乗算回路における MAE の測定結果を図 10 に示す．いずれも縦軸は MAE であり，横軸はパラメータ $p_S$ または $p_F$ である．また，従来型 SC (Conventional SC) と従来型 AFE SC の結果を，それぞれ緑と青の点線で示している．図 10 (a) より，$p_S$ が 3 ビットのとき MAE が底打ちになっており，従来型 AFE SC と同等になっていることがわかる．また，図 10 (b) より，$p_F$ が 3 ビットのとき MAE が 従来型 AFE SC と同等になっていることがわかる．

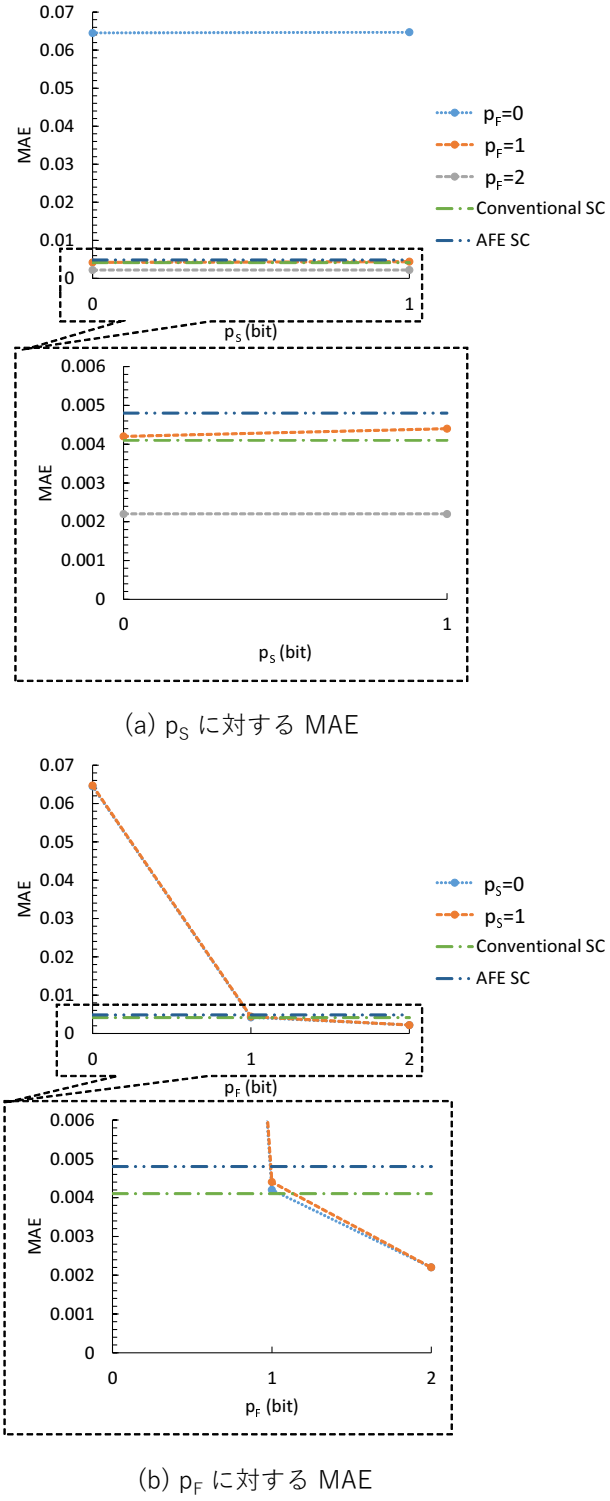


図 11　平均演算回路の MAE

Fig. 11　MAE of averaging circuit

次に，平均演算回路における MAE の測定結果を図 11 に示す．図 11 (a) より，$p_S$ が 0 ビットのとき MAE が底打ちになっており，従来型 AFE SC や従来型 SC より小さくなっていることがわかる．また，図 11 (b) より，$p_F$ が 2 ビットのとき MAE

表 1　乗算回路の LUT 利用数

| | $p_S = 0$ | $p_S = 1$ | $p_S = 2$ | $p_S = 3$ | $p_S = 4$ |
|---|---|---|---|---|---|
| $p_F = 0$ | 8 | 13 | 17 | 23 | 29 |
| $p_F = 1$ | 22 | 19 | 29 | 30 | 27 |
| $p_F = 2$ | 27 | 29 | 32 | 34 | 39 |
| $p_F = 3$ | 28 | 31 | 33 | 36 | 40 |

表 2　平均演算回路の LUT 利用数

| | $p_S = 0$ | $p_S = 1$ |
|---|---|---|
| $p_F = 0$ | 3 | 3 |
| $p_F = 1$ | 8 | 9 |
| $p_F = 2$ | 9 | 12 |

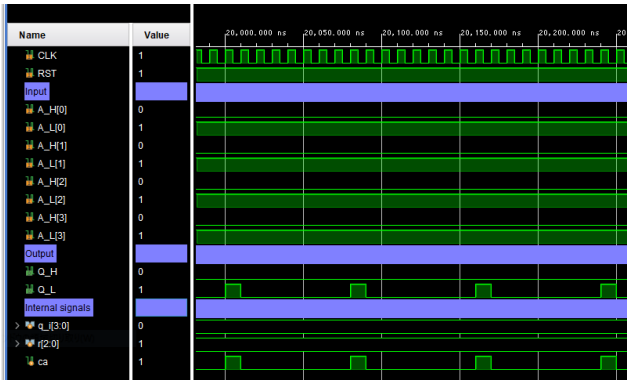


図 12　乗算器の入力すべてに 0.5 を与えたときのシミュレーション波形
Fig. 12　Simulated waveform when all multiplier inputs are given 0.5

が従来型 AFE SC や従来型 SC より小さくなっていることがわかる．これは従来型 AFE SC の場合，割り算を行うために追加で SB を必要とするため，乱数のばらつきの影響が大きくなったことが原因と考えられる．

乗算，平均演算回路における LUT の利用数をそれぞれ表 1，2 に示す．表 1，2 から LUT の利用数は，$p_S$ や $p_F$ の増加に応じて，それぞれおおむね独立して増加していることがわかる．ここで，従来型 SC の乗算，平均演算回路における LUT の利用数はともに 1 個であり，従来型 AFE SC の乗算，平均演算回路ではそれぞれ，13 個，12 個であった．乗算回路では従来型 AFE SC と比べて全体的に LUT の利用数が増加するものの，平均演算回路では LUT の利用数は従来型 AFE SC と同等以下となった．

### 4.3　考　　察

ここでは，信号間に相関が生じる可能性について考える．具体例として，乗算器の入力すべてに 0.5 を与えたときのシミュレーション波形を図 12 に示す．範囲 $[0, 1]$ の入力を 32 ビットに正規化すると，入力が 0x1FFFFFFF（実数値で 0.49999…）となる．このとき Type-A の AFE SB 生成器（2.2 節参照）では，MSB は常に “0” を出力し，LSB は，極めて 1 に近い確率で “1” を出力する．つまり，すべての入力がほとんどのクロックサイクルで同一となる．これを LUT に入力すると，LUT の出力は S が常に “0000”，F は常に “001” (0.0625) になる．補正回路にランダム性はないため，図 12 に示す通り，AFE SC 出力は 8 サイクルに 1 度の周期で “01” となる．

## 5.　お わ り に

本稿では，AFE SC の出力ビット幅増大の問題を解決する，新たな演算器の実装手法を提案した．切り捨てた端数は順序回路を用いて SB に反映させることで，誤差を抑えながら出力ビット幅を制限できる．一方で，従来の手法と比べて LUT の利用数が増加するほか，フリップフロップが追加で必要となる．

明らかになっている課題として相関の解消が挙げられる．Lee ら [12] で述べられている通り，SC における各信号の相関は，演算結果に大きな影響を与える．4.3 節で示した周期的な信号は，演算器を多段化した際に相関の問題を生じさせると考えられる．そのため，補正回路に何らかのランダム性を加えるなどの方法で，出力の周期性を抑え，相関の問題を解消することが今後の課題である．

**謝辞**　本研究の一部は，日東学術振興財団の支援によるものである．

### 文　　献

[1] A. Krizhevsky, I. Sutskever, and G.E. Hinton, “Imagenet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems, vol.2, pp.1097–1105, 2012.
[2] Advanced Micro Devices, Inc., “DPUCZDX8G for Zynq UltraScale+ MPSoCs, product guide PG338 (v4.1),” 2023.
[3] Y. Umuroglu, N.J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, “FINN: A framework for fast, scalable binarized neural network inference,” 2017 ACM/SIGDA international symposium on field-programmable gate arrays, pp.65–74, 2017.
[4] W.J. Gross and V.C. Gaudet, Stochastic Computing: Techniques and Applications, Springer, 2019.
[5] P. Li, D.J. Lilja, W. Qian, K. Bazargan, and M.D. Riedel, “Computation on stochastic bit streams digital image processing case studies,” IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol.22, no.3, pp.449–462, 2013.
[6] A. Ren, Z. Li, C. Ding, Q. Qiu, Y. Wang, J. Li, X. Qian, and B. Yuan, “SC-DCNN: Highly-scalable deep convolutional neural network using stochastic computing,” ACM SIGPLAN Notices, vol.52, no.4, pp.405–418, 2017.
[7] H. Sim and J. Lee, “A new stochastic computing multiplier with application to deep convolutional neural networks,” 54th annual design automation conference 2017, pp.1–6, 2017.
[8] S. Liu and J. Han, “Toward energy-efficient stochastic circuits using parallel Sobol sequences,” IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol.26, no.7, pp.1326–1339, 2018.
[9] D. Jenson and M. Riedel, “A deterministic approach to stochastic computation,” 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp.1–8, 2016.
[10] Y. Chen and H. Li, “Stochastic computing using Amplitude and Frequency Encoding,” IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol.30, no.5, pp.656–660, 2022.
[11] N. Fujieda, “A Python-based evaluation framework for stochastic computing circuits on FPGA SoC,” 2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW), pp.81–86, 2021.
[12] V.T. Lee, A. Alaghi, and L. Ceze, “Correlation manipulating circuits for stochastic computing,” 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.1417–1422, 2018.