An Improved Deterministic Stochastic MAC (SC-MAC) for High Power Efficiency Design

Ming Ming Wong(✉), Lu Chen, and Anh Tuan Do

Institute of Microelectronics (IME) A*STAR, Singapore, Singapore
{wong_ming_ming,doat}@ime.a-star.edu.sg

Abstract. Convolutional Neural Network (CNN) is widely acknowledged as an effective machine learning model for various detection and recognition tasks. However, CNN often requires a significant amount of hardware resources and is high in its power consumption. This hinders the widespread deployment of CNN model in embedded systems and wearable devices. Therefore, stochastic computing (SC) which leverages the power-accuracy trade-off, began to gain popularity in various neural network (NN) implementations. This paper presents an improved SC multiply-and-accumulate (MAC) unit that can be utilized as convolution engines in CNN. The proposed SC-MAC is operated using deterministic sequence and the design achieves latency and power reductions through parallelism and split mechanism optimizations. Furthermore, we also introduce decoder-based Stochastic Number Generator (SNG) that is capable of generating uncorrelated and segmented stochastic number (SN) without using random sources. The proposed deterministic and split SC-MAC is synthesized using typical libraries of UMC 40 nm technology for detailed hardware evaluation. The functionality of the presented SC-MAC is also verified in CNN using the MNIST dataset. Overall, our SC-MAC is proven to achieve higher power efficiency (GMACS/mW) and lower in energy consumption (pJ/MAC) as compared to the related works.

Keywords: Stochastic Computing (SC) · Stochastic Number Generator (SNG) · multiply-and-accumulate (MAC) · Shared segmented/split design · Convolution engine

1 Introduction

In recent years, Convolutional Neural Network (CNN) has emerged as one of the most promising artificial neural networks and has been deployed in a wide range of machine learning applications such as image/video classification

This research is supported by Programmatic grant no. A1687b0033 from the Singapore government's Research, Innovation and Enterprise 2020 plan (Advanced Manufacturing and Engineering domain).

© IFIP International Federation for Information Processing 2022
Published by Springer Nature Switzerland AG 2022
V. Grimblatt et al. (Eds.): VLSI-SoC 2021, IFIP AICT 661, pp. 245–266, 2022.
https://doi.org/10.1007/978-3-031-16818-5_12

[4,14], speech recognition [25] and natural language processing [8]. Software-based CNN/DCNN usually requires high-performance computer (with accelerators such as GPUs) to process excessive and intensive computations at a very high processing speed.

The core operation in CNN/DNN computational layer is the convolution function that involves multiplication and accumulation of the receptive field and a set of filters/kernels. Due to the excessive amount of multiply-accumulate (MAC) functions, CNN has rather high implementation cost. With that, MAC unit is also known as system's bottleneck as its design's characteristic fundamentally determines the system's overall area, power and performance [31]. Hence, researchers have placed great emphasis on MAC design optimization techniques, so as to enable neural network (NN) deployment in resource-constrained embedded systems. Instead of exploring new optimization techniques in binary arithmetic computing, this study focuses on the non-conventional computing domain, which is Stochastic Computing (SC) for CNN implementation.

In the recent decade, SC has appeared to be a popular solution for hardware implementation of NN. SC is a form of approximation computing that substitutes complex mathematical operations with simple logic gates. The biggest advantage of employing SC is that the resultant circuitry has significantly smaller hardware footage compared to their binary fixed point counterparts [6,18]. Besides, SC is also proven capable to outperform conventional computing in terms of fault tolerance [23]. It is further reported that DSP and NN in their nature are able to work relatively well using SC, provided its internal computations are able to attain a certain level of accuracy [23,31]. However, the typical approach for binary-to-stochastic domain conversion is rather costly as it requires random number generators (RNGs). Not only that, in order to minimize the conversion error due to the random fluctuations, the required length of the bit-streams is increased exponentially with respect to its binary resolution n. As a result, stochastic bit-stream is larger than 2^{2n} bits [13].

This paper is an extended version of an earlier publication of ours [29], where we provide further descriptions and analysis of our SC-MAC design. To this end, we highlight the following contribution aspects of this work:

– We present a decoder-based stochastic number generator (SNG) that produces deterministic and highly uncorrelated stochastic number (SN). Precision progression of the SNs that are generated from both the positive and negative binary numbers are analyzed. Results proved that our SNG achieves lower representation error and requires smaller resolution bit.
– We address the main challenges in the conventional non-deterministic SC in achieving low latency and high accuracy multiplication. We further demonstrate that SC multiplication using our SNG is free from random fluctuation and accuracy loss due to data correlation.
– We incorporate parallelism and split mechanism in our SC-MAC unit in order to reduce the computational latency. The implementation of both of the optimization techniques is discussed in detail and we further elaborate the integration of approximate parallel counter (APC) in our SC-MAC design.

– Our design's implementation cost and its performance are evaluated using 40 nm process while the computational accuracy is validated in CNN using MNIST dataset.

The rest of the paper is organized as follows. Section 2 briefly introduces the background concepts of stochastic computing (SC) that are used in the rest of the chapter. Section 3 highlights the challenges and the problems in the conventional non-deterministic SNG design. Section 4 describes our new decoder-based SNG design and its analysis results are presented as well. Section 5 elaborates our new SC-MAC solution that is optimized with parallelism and split mechanism. Section 6 summarizes the overall performance analysis and the benchmarks with related works. Finally, the conclusions are drawn in Sect. 7.

2 Theory of Stochastic Computation

Stochastic computing (SC) is a form of non-conventional computation where the computational data is represented as a result of continuous time stochastic process [9]. This section describes the preliminaries of SC that will be used throughout this chapter.

2.1 Architecture of Stochastic Computing (SC)

The general architecture of Stochastic Computing (SC) is illustrated in Fig. 1. The architecture is comprised of stochastic number generator (SNG) that converts (or randomized) binary values into stochastic bit-streams. Meanwhile, the arithmetic functions (i.e. multiplication, addition/subtraction and many more) are implemented as stochastic computational elements (SCE) by using simple logic gates. The final outputs of SCE are converted (or de-randomized) back to the binary representation. This conversion is performed through counting the total number of non-zero bits in the stochastic bit-stream. Further descriptions of the SC components are elaborated in the following subsections.

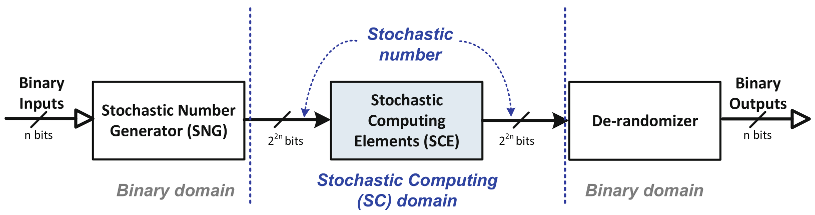

Fig. 1. Stochastic computing architecture that comprises of stochastic number generator (SNG), stochastic computing elements (SCE) and de-randomizer. The SCE is performed in SC domain where its inputs and outputs are represented in stochastic numbers (SN).

2.2 Stochastic Number (SN)

The computational data (i.e. **Stochastic Number (SN)**) is encoded in the form of digitized probability which is defined by the number of non-zero bits in the bit-stream. In other words, the SN value is associated with the ratio of total bit-1s to the total bit number [23]. SN represents computational data in two different formats: *unipolar* and *bipolar* representations [9]. Using unipolar representation, the values are bound within the internal $0 \leq s \leq 1$, while using bipolar representation, the values are extended to $-1 \leq s \leq 1$.

With $P(S = 1)$ is the probability of non-zero bits in bit-stream S, both the unipolar (UR) and bipolar (BR) representations are derived using Eq. 1 [9,28]. For example, in Fig. 2, bit-stream S of $2^4 = 16$ bits has 13 bit-1. In unipolar representation, bit-stream S is equivalent to $UR = P(S = 1) = 0.625$. Meanwhile, in bipolar representation, the same bit-stream will be interpreted as $BR = 0.8125$.

$$\begin{aligned} UR &= P(S=1) \\ BR &= 2\left(P(S=1) - \frac{1}{2}\right) \end{aligned} \tag{1}$$

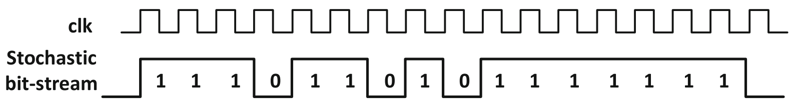

Fig. 2. Stochastic Number (SN) generated in serial, $S = 13/16$ representing 0.625 in unipolar representation and 0.8125 in bipolar representation.

Note that stochastic representation is not defined based on the position of any particular bit in the bit-stream S [28]. Instead, it is based on the probability of total bit-1s at arbitrary position. As SC utilizes non-positional number representation, it is less susceptible towards errors that are caused by bit-flip. Meanwhile, in the conventional 2's complement computation, single bit-flip on the higher-order bit will lead to significant error. As all the bits in the SN bit-stream carry equal significance, single bit-flip in a long bit-stream will only cause a minor deviation in its binary representation.

2.3 Binary-Stochastic Data Conversion

There are two types of converters (refer Fig. 3) that are required in SC architecture, which is the *randomizer* or generally known as the stochastic number generator (SNG), and the *de-randomizer* or simply known as counter.

SNG performs binary to stochastic conversion and it is typically designed using random source and comparator. The conversion is based on the comparison between the binary data and the values from the random source (refer

Fig. 3 (i)). If the binary input is larger than the random value, output bit-1 will be generated, otherwise output bit-0 will be generated. Linear Feedback Shift Register (LFSR) is one of the common choices for (pseudo) random number generator [12]. A k-bit LFSR is able to generate a total of $2^k - 1$ unique k-bit outputs. Therefore, comparing the binary input X (of k-bit) with the LFSR output sequence will generate SN bit-stream of $2^k - 1$ bits. The generated bit-stream contains a total of $X - 1$ bit-1s which the SN of binary input X is represented as $\frac{X}{2^k}$ [12].

On the other hand, converting the SN back to the binary data is simply calculating the total bit-1s in the bit-stream. This can be easily implemented using counter such as shown in Fig. 3 (ii). Following the nature of this stochastic representation, the subsequent SC arithmetic can be implemented using simple logic circuits [2,23] which will be explained next.

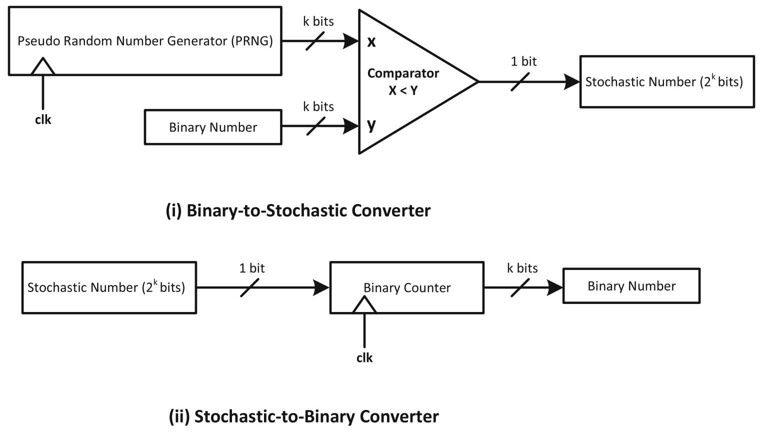
Pseudo Random Number Generator (PRNG)
clk
k bits
X
Comparator
X < Y
Y
k bits
Binary Number
1 bit
Stochastic Number (2^k bits)
(i) Binary-to-Stochastic Converter
Stochastic Number (2^k bits)
1 bit
Binary Counter
clk
k bits
Binary Number
(ii) Stochastic-to-Binary Converter

Fig. 3. Computational data conversion. (i) Randomizer/SNG: From binary data to stochastic representation (ii) De-randomizer/counter: From stochastic representation back to binary data.

2.4 Stochastic Computing Elements (SCE)

Multiplication in SC can be effectively implemented using single logical gate. Assuming the input to the multiplication, X_1 and X_2 are uncorrelated, the derivation of its output Y, is given in Eq. 2. With that, as depicted in Fig. 4, the logical AND gate and logical XNOR gate are used as a SC multiplier in unipolar and bipolar representation respectively.

$$\begin{aligned} y &= P(Y) \\ &= P(X_1) \cdot P(X_2) + (1 - P(X_1)) \cdot (1 - P(X_2)) \end{aligned} \tag{2}$$

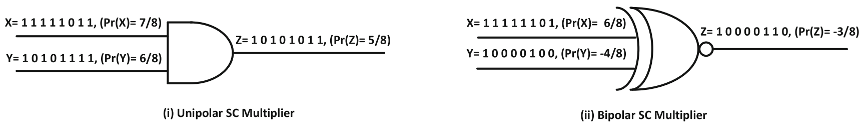
X= 1 1 1 1 1 0 1 1, (Pr(X)= 7/8)
Y= 1 0 1 0 1 1 1 1, (Pr(Y)= 6/8)
Z= 1 0 1 0 1 0 1 1, (Pr(Z)= 5/8)
(i) Unipolar SC Multiplier
X= 1 1 1 1 1 1 0 1, (Pr(X)= 6/8)
Y= 1 0 0 0 0 1 0 0, (Pr(Y)= -4/8)
Z= 1 0 0 0 0 1 1 0, (Pr(Z)= -3/8)
(ii) Bipolar SC Multiplier

Fig. 4. Stochastic Multiplier for (i) unipolar and (ii) bipolar representations.

Addition in SC is performed in a scaled manner such that the output is probability value within the range $[0, 1]$. To be exact, SC addition requires a constant scale, S, such that the sum (Y) of two inputs X_1 and X_2, is defined as Eq. 3.

$$\begin{aligned} y &= P(Y) \\ &= P(S)P(X_1) + (1 - P(S))(P(X_2)) \\ &= SX_1 + (1 - S)X_2 \end{aligned} \tag{3}$$

Thus, multiplexer with conditional select line S, set as $P(S) = \frac{1}{2}$ can be used to realize the scaled addition of two stochastic bit-streams in digital circuit.

Subtraction in SC is essentially the same as the SC addition but with one of the input is inverted (using logical NOT gate). Both the stochastic scaled adder and scaled subtractor are shown in Fig. 5.

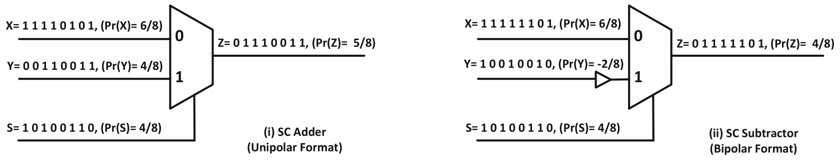
X= 1 1 1 1 0 1 0 1, (Pr(X)= 6/8)
Y= 0 0 1 1 0 0 1 1, (Pr(Y)= 4/8)
0
1
Z= 0 1 1 1 0 0 1 1, (Pr(Z)= 5/8)
S= 1 0 1 0 0 1 1 0, (Pr(S)= 4/8)
(i) SC Adder
(Unipolar Format)
X= 1 1 1 1 1 1 0 1, (Pr(X)= 6/8)
Y= 1 0 0 1 0 0 1 0, (Pr(Y)= -2/8)
0
1
Z= 0 1 1 1 1 1 0 1, (Pr(Z)= 4/8)
S= 1 0 1 0 0 1 1 0, (Pr(S)= 4/8)
(ii) SC Subtractor
(Bipolar Format)

Fig. 5. Stochastic scaled adder/substractor for (i) unipolar and (ii) bipolar representations.

3 Challenges in Conventional (Non-deterministic) Stochastic Computing (SC)

Though SC is a favourable alternative to binary computing, we have identified two main drawbacks in the computation, which also serve as the main motivations of this study. First, the **stochastic number generator (SNG) incurs the major overhead** in the entire SC system. To be exact, conventional SNG that utilizes random sources consumed up to 80% of the overall computational cost [3]. For instance, the LFSR-based SNG that is commonly used for binary-to-stochastic conversion (refer to Sect. 2), has high power dissipation per area as compared to the SC element such as the logical AND or XNOR gates.

Following this, several works either designed compact random source (such as RNG) for SNG [11] or proposed random source sharing between the SNGs in order to reduce the overall hardware area cost [16,20]. However, it is worth a note that the latter approach causes correlation between the input SN bit-streams [20]. This leads to the second challenge in SC circuit design, which the **correlation between the SN bit-streams will degrade the overall computational accuracy**. In other words, correlation in the input data tends to alter the expected outcome of the stochastic logic and this is evident in stochastic multiplication. One of the potential scenarios is to multiply stochastic inputs that are directly inverse of each other and this produces the output Z as zero instead of the product $P_x P_y$.

The technical implication of data correlation in SC context is reported in [1, 22]. The works proved that SC multiplication which involves logical AND/XNOR gates will suffer from accuracy degradation when the inputs are correlated. On the other hand, the study further reported that such output discrepancy does not happen when correlated or uncorrelated input data are used in the multiplexer. Therefore, stochastic scaled addition/subtraction naturally is not affected. This analysis is also summarized using the examples given in Fig. 6.

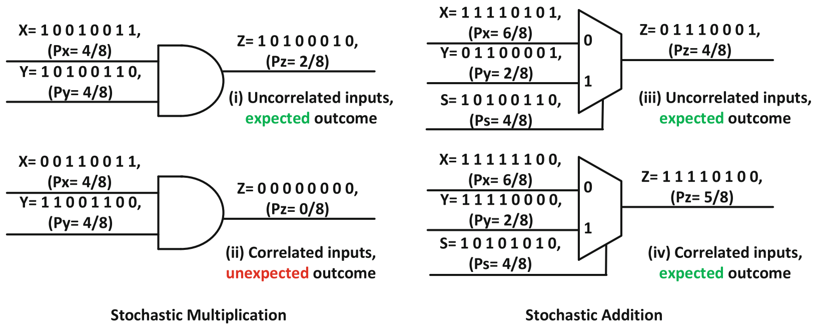

Fig. 6. Analysis of SCE with correlation data. (Left) Stochastic multiplication (i) using uncorrelated inputs and produces accurate/expected result and (ii) using correlated inputs and produces inaccurate/unexpected results. (Right) Stochastic addition with accurate/expected result using (iii) uncorrelated inputs and (iv) correlated inputs.

Based on the discussion above, it is evident that uncorrelated SN generation is essential to obtain high accuracy computation in SC multiplication, as well as SC-MAC. This also implies that random source is not necessarily needed for SNG. Therefore, this study focuses on non-conventional SC approach where the computations are performed on deterministic sequences.

4 New Lightweight and Deterministic SNG Design

In this work, we presented a new SNG that overcomes the drawbacks in the conventional design (refer to Sect. 3). The new SNG is lightweight, and produces deterministic and uncorrelated SN without the need of random sources (PRNG/RNG). Detailed description of our design and its analysis results are elaborated in the following subsections.

4.1 Decoder-Based and Deterministic SNG

The main concept of the proposed SNG is using binary data to produce a primary k-bit pattern which can be repeated p times to generate the deterministic SN bit-stream. An example of the generated SN bit-stream is shown in Fig. 7. Furthermore, both the parameters p and k can be configured in a way that the SN fulfills the required precision and correlation levels in the SC system.

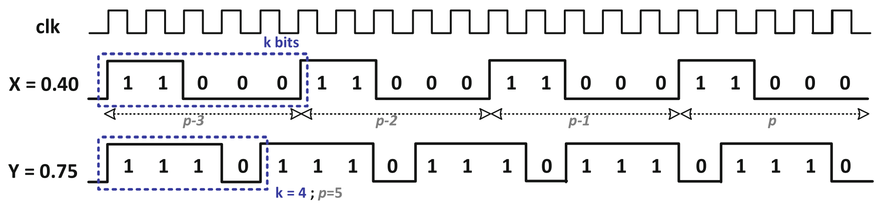

Fig. 7. Deterministic SN examples (i) For X = 0.40, 2/5 is repeated 4 times to generate bit-stream that comprises of 40% bit-1 and 60% bit-0. (ii) For Y = 0.75, 3/4 is repeated 5 times to generate bit-stream that comprises of 75% bit-1 and 25% bit-0.

The core design of this SNG is the k-bit pattern generator, which we implemented using n-to-2^n decoder. With that, the decoder's output is concatenated p times to produce SN bit-stream of $k \times p$ bit length. The design is low in hardware cost and high in efficiency because the SN bit-stream can be generated instantaneously within a clock cycle. The proposed decoder-based SNG and its comparison with the conventional SNG and the existing deterministic SNGs (using source/wave generator) are illustrated in Fig. 8.

In addition to the design complexity, we also analyzed the precision progression of the generated SN in representing the positive and the negative binary numbers. As our deterministic SNG does not utilize random source, the generated SN is free from random fluctuations. Besides, with our SNG, highly accurate SN can be attained with smaller resolution bit. As presented in Fig. 9, conventional SNG (using random source) requires at least 2^{16} bits to accurately represent bipolar real number in the intervals of $[0, 1]$ and $[-1, 0]$. On the other hand, our proposed SNG requires only 2^6 bits to sufficiently represent the same bipolar numbers (refer Fig. 10).

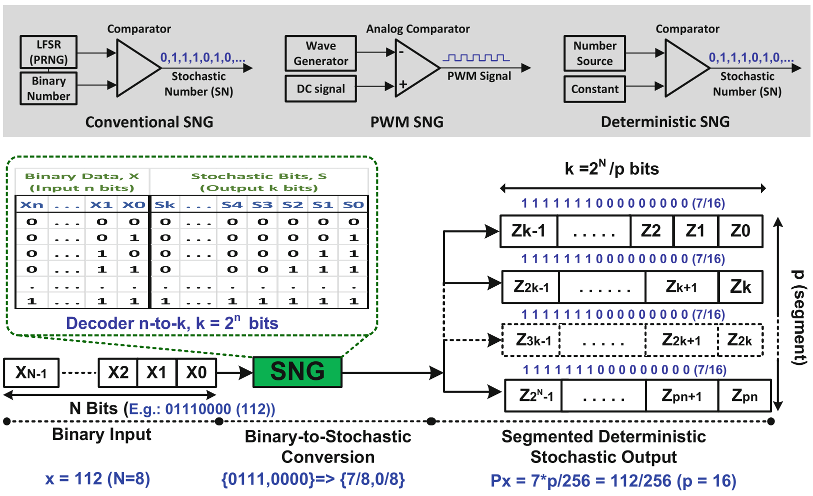

Fig. 8. (Top) SNG designs from prior works. The deterministic SNG design is reported [13] .(Bottom) The proposed decoder-based SNG: SNG conversion for 8-bits input, X, to deterministic SN output of 2^8-bits with ($p = 16$) segments and each segment is ($k = 16$) bit-length. Note that the parameters p and k can be configured according to the application's requirements.

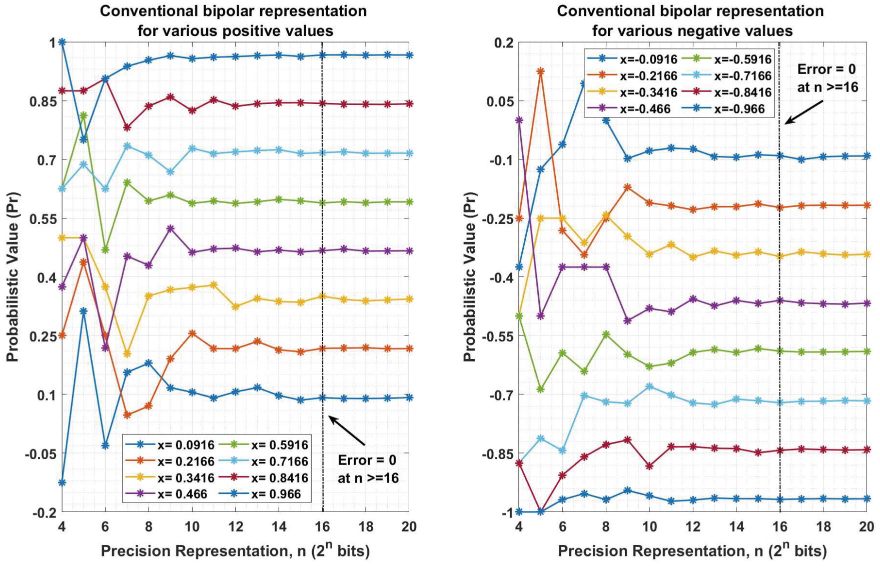

Fig. 9. Stochastic values derived using conventional (non-deterministic) SNG across a range of 2^n precision bits. It is shown that at least 2^{16} bits is required to represent both (i) positive and (ii) negative values in SN without error.

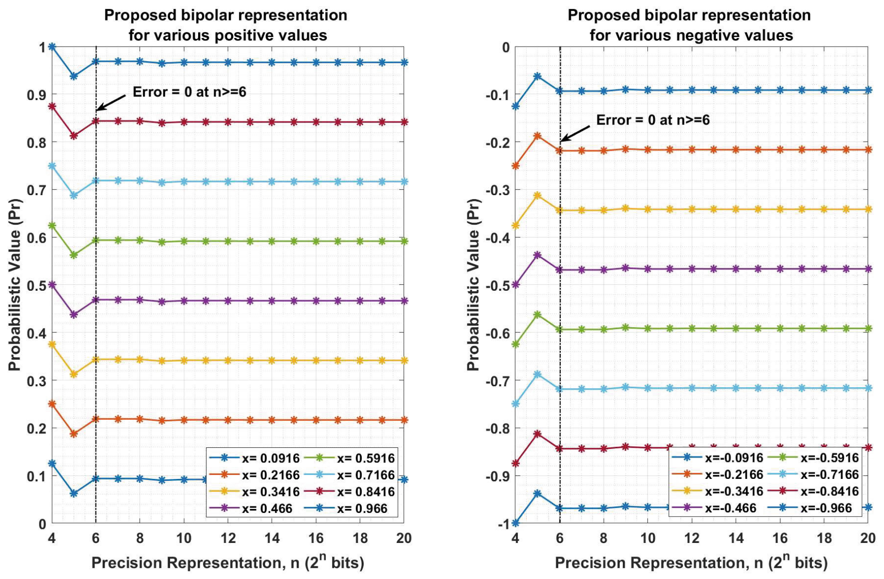

Fig. 10. Stochastic values derived using proposed (deterministic) SNG across a range of 2^n precision bits. It is shown that 2^6 bits is sufficient to represent both (i) positive and (ii) negative values in SN without error.

4.2 Stochastic Multiplication Using Decoder-Based SNG

Since the proposed SNG produces deterministic bit-stream which is comprised of p repetitions of k-bit pattern, we configure these parameters to generate uncorrelated SNs for stochastic multiplications. Given u/x is represented in x-bits with u number of bit-1 and the remaining $x - u$ bits are zeros. Similarly, given v/y is represented in y-bits with v number of bit-1 and the remaining $y - v$ bits are zeros. Assuming both bit-streams are repeated to S-bit length, the stochastic multiplication of $(u/x) \times (v/y)$ can be computed correctly if S is the least common multiple (LCM) number of x and y and that x and y are relatively prime [21].

As an example, the generated SN using the proposed SNG for input X and Y are shown in Fig. 11 and are contrasted with the conventional random SNs. In Fig. 11 (ii), the input $x = 2/5$ is repeated 3 times while the input $y = 2/3$ is repeated 5 times to produce bit-streams of 15-bit (i.e. $LCM(5, 3)$). As a result, multiplying (AND) both the bit-streams produces the same result as the conventional stochastic multiplication in Fig. 11 (i). This example demonstrates the generated deterministic SNs are uncorrelated and are feasible for SC multiplication.

We further analyzed the distribution of the output obtained from SC multiplication across a range of different j-bit resolution. In this analysis, SC multiplication is performed using SNs of the same value 0.6 (i.e. 0.6×0.6) that is represented using bit-streams of 2^j length with j varies from 5 to 10 bits. The

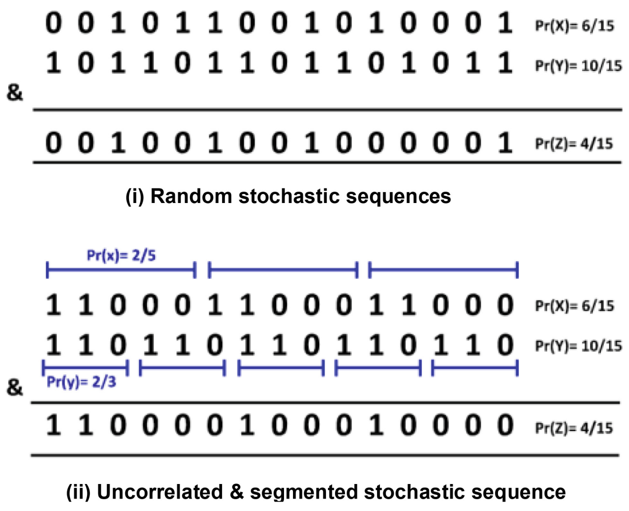
0 0 1 0 1 1 0 0 1 0 1 0 0 0 1 Pr(X)= 6/15
1 0 1 1 0 1 1 0 1 1 0 1 0 1 1 Pr(Y)= 10/15
&
0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 Pr(Z)= 4/15
(i) Random stochastic sequences
Pr(x)= 2/5
1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 Pr(X)= 6/15
1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 Pr(Y)= 10/15
Pr(y)= 2/3
&
1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 Pr(Z)= 4/15
(ii) Uncorrelated & segmented stochastic sequence

Fig. 11. SC unipolar multiplication (logical AND operation) of two input sequences, X and Y, which are generated in two different probabilistic representations; (i) random stochastic sequence (ii) uncorrelated and segmented stochastic sequence. The example shows that sequences in (ii) that are generated by the proposed SNG can be used to perform multiplication in the same way as the conventional random sequence in (i).

computation is experimented using SNs generated from (i) Conventional SNG, (ii) SNG [16] and (iii) Proposed SNG. The outcome of the SC multiplication using various precision bits is shown in Fig. 12.

In this figure, the accuracy of SC multiplication is reflected in terms of the mean, max and min values where the expected value is $0.6 \times 0.6 = 0.36$. From the observation in Fig. 12 (iii), multiplication using deterministic and uncorrelated SNs produces output that is free from random fluctuation resultant from PRNG/RNG. With that, the output is always consistent and hence, the max and min values are the same as the median value for all the precision bit. Therefore, our SNG in (iii) has outperformed (i) and (ii) in terms of the quality of the generated SN bit-streams. In addition to that, we further extended the experiment to using random inputs to perform SC multiplication. The average errors with respect to the range of precision bits are summarized in Fig. 12 (iv). For $j = 10$ bits, the observed error from the multiplication is less than 3%. This analysis has proven that our proposed SNG ensures both the SC representation accuracy and multiplication accuracy.

4.3 Near Zero Bipolar Representation Analysis

We further evaluate the accuracy of the proposed SNG in converting the near zero binary value to SC bipolar representation. For SC bipolar encoding, it is known that near-zero values tend to generate large random errors and this will affect the accuracy in SC multiplication [15]. In CNN/DNN architecture, the

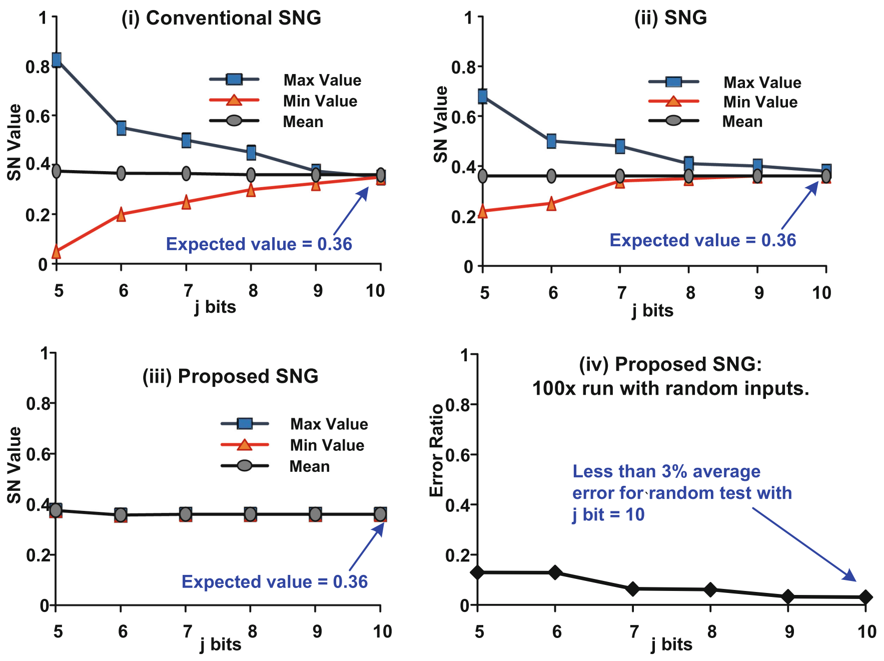

Fig. 12. Accuracy of SC unipolar multiplication (logical AND operation) using inputs 0.6 derived from (i) Conventional SNG (ii) SNG reported in [16] and (iii) Proposed SNG. The expected value is 0.36. (iv) Average error ratio for 100 runs with random inputs.

synaptic weights are often initialized to normally distributed random numbers. At the same time, the weights value are aggregated towards zero (due to L1-, L2-regularization) so as to give penalties to non-zero parameters as a means to prevent over-fitting [10]. Thus, deploying SC for CNN/DNN applications can be challenging where the majority kernel weights are near-zero values. In this case, using deterministic SN which is highly consistent (without random fluctuation) will be a better alternative.

In this analysis, the representation accuracy for zero and near-zero values using deterministic SN (from our SNG) as compared to the conventional random SN is shown in Fig. 13. First, the accuracy obtained from zero values encoding in SC bipolar representation using a range of precision bit is shown in Fig. 13 (i). The result shows that it requires more than 2^{16} bits to achieve error free SC encoding for zero binary value. On the other hand, our deterministic approach only requires 2^4 bits to accurately represent zero value. Next, we analyzed the approximation error obtained for near zero values ($x \in [-0.02, 0.02]$) in SC bipolar representation. Based on the result in Fig. 13 (ii), conventional bipolar representation has significant higher approximation error for near zero values. The error is observed to be higher at the value is closer towards zero.

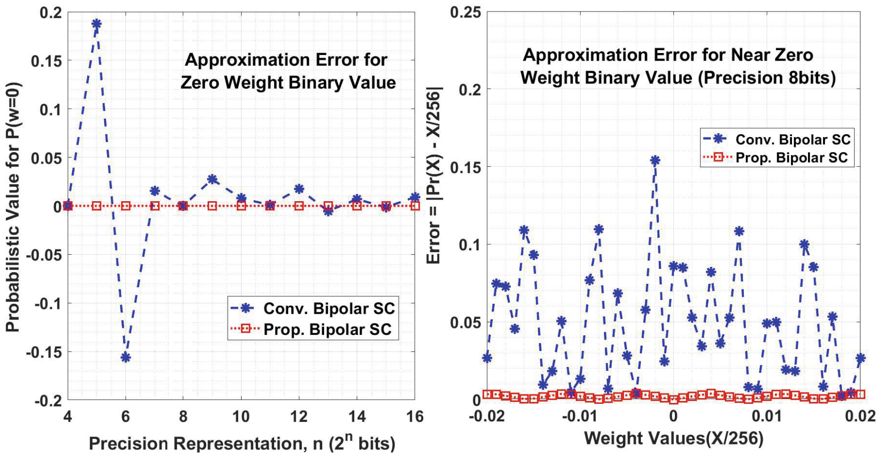

Fig. 13. Error analysis in SC bipolar representation for (i) zero value (ii) near zero values. The analysis results show that the proposed SNG is able to produce SN with lower error.

4.4 Hardware Area and Power Analysis

In this subsection, we present the analysis of the hardware cost (area resource and power consumption) of the proposed SNG. The synthesis result of the existing designs and our SNG are summarized in Table 1. The results show that our SNG has the lowest power consumption and is the most lightweight (area) compared to both the conventional and the other deterministic-based SNGs.

5 New SC-MAC Solution for Convolution Engine

In the previous section, we presented a new SNG design that generates uncorrelated and deterministic SN which (i) is free from random fluctuation errors and (ii) achieves high bipolar representation accuracy. Following this, we propose a new SC-MAC design for convolution engine that computes under deterministic stochastic logic. We further incorporated *parallelism* and *split SC* mechanism in order to achieve energy reduction and power efficiency improvement. Detailed description of the proposed SC-MAC design will be explained in the following subsections.

Table 1. Hardware area and power for various SNG designs for 2 parallel inputs of 8-bits resolution, synthesized using 40 nm process node at nominal voltage.

SNG	Area	Power	Remarks
Work [23]	357.14	366.62	LFSR and comparator
Work [3]	263.90	470.91	Analog-stochastic Converter
Work [30]	211.75	210.17	Error cancellation (ECPCC)
Work [21]	237.04	395.06	PWM (deterministic)
	327.46	NA	Relative prime (deterministic)
Work [13]	436.61	NA	Rotation (deterministic)
	327.46	NA	Clock divider (deterministic)
This work	173.71	4.2	Split SC and decoder (deterministic)

5.1 Parallel and Split SC Computation

Conventional SC often suffers from long computational latency due to SNG that is operated in a serial manner. On the other hand, the decoder-based SNG presented in this work enables the SN bit-stream to be generated instantaneously (refer to Sect. 4). With that, the overall SC performance is no longer constrained by the stochastic input generation rate. Subsequently, speed improvement techniques are feasible to be incorporated in our SC-MAC design effectively.

First, **parallelism** technique can be employed in order to reduce the total execution cycles. Using this approach, bit-parallel processing is incorporated in the SC-MAC operation such that L-bit sequence is partitioned into L/r sequences of r bits. This way, all of the L/r sequences can be processed in parallel as shown in the example in Fig. 14. This figure shows that the 16-bit input is partitioned into four of 4-bit sequences which can be processed simultaneously. Therefore, for our SC-MAC implementation, we have chosen $r = 32$ such that the inputs to the SC-MAC with $L = 256$ can be completed in 8 cycles. In each cycle, the r-XNOR can be processed simultaneously and followed by accumulation, which will be discussed in the next subsection.

Second, **split SC** mechanism can be incorporated to effectively reduce the SN bit-length. We introduced split SC-MAC architecture where the N bits fixed-point binary data is split into k parts prior to the computation. Therefore, the resulting SN is represented as k times of $2^{N/k}$ bit-streams (instead of a single 2^N bit-stream). This results in computation speedup by a factor of k. Similar approach was presented in [7] but the work reviewed that their design incurred higher area and power consumption as compared to the original SC architecture. Furthermore, the design also required additional SNGs to generate parallel bit-streams and this leads to hardware cost overheads.

In this study, we implemented split stochastic processing with $k = 2$ such that the binary inputs are divided into 2 equal segments (refer Fig. 16). In the context of convolution, given two fixed point binary input to the SC-MAC are input feature X, and kernel weights W, these inputs are divided as $X = \{X_H, X_L\}$ and

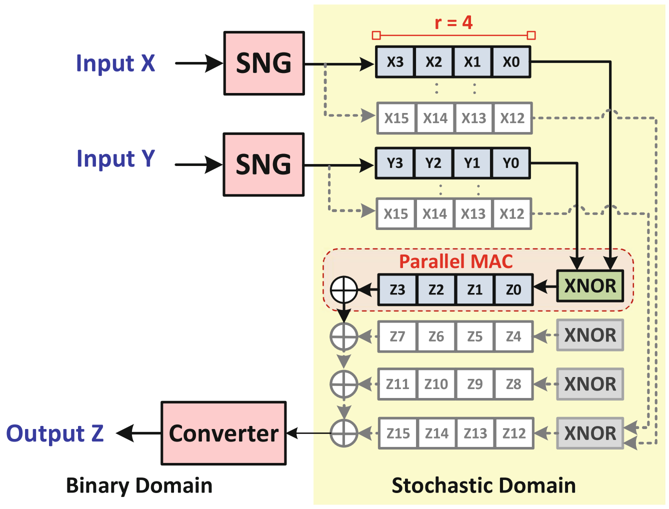

Fig. 14. Bit-parallelism of $r = 4$ for SC-MAC function. This example shows the input sequence of 16-bits that are partitioned into shorter sequences ($r = 4$).

$W = \{W_H, W_L\}$. With that the SNG converts the segments X_H, X_L, W_H, W_L to SN bit-streams prior to MAC operations.

5.2 Optimized Bipolar SC Multiplication and Addition

The overall architecture of our split SC-MAC architecture as the core computation for convolution engine is depicted in Fig. 15. An example is provided in the figure to explain the computation. The bipolar SC multiplication and SC addition for the MAC operation in our design are explained in the following.

SC Multiplication: While the conventional bipolar SC multiplication is described in Sect. 2, the split bipolar SC multiplication used in this work is described in Eq. 4 (also refer to Fig. 16). Note that in Eq. 4, the term $P_r(X_{lo}) \cdot P_r(W_{hi})$ is intentionally excluded from the original dot-product terms (i.e. $P_r(X_{lo}) \cdot P_r(W_{lo})$, $P_r(X_{lo}) \cdot P_r(W_{hi})$, $P_r(X_{hi}) \cdot P_r(W_{lo})$, $P_r(X_{hi}) \cdot P_r(W_{hi})$). For the input feature X, its MSB carries larger significance over its LSB and meanwhile, the kernel weights values are often very small and hence its LSB is more significant compared than its MSB. Therefore, for split SC-MAC, this dot-product term can be omitted in order to reduce the computation complexity.

The split bipolar SC multiplication requires XNOR for dot-product $(\cdot)$ and the term $\{P_r(X_{hi}) + P_r(X_{lo})\}$ is simply a bit concatenation. Furthermore, the dot-product is performed using 32 XNORs executed in parallel.

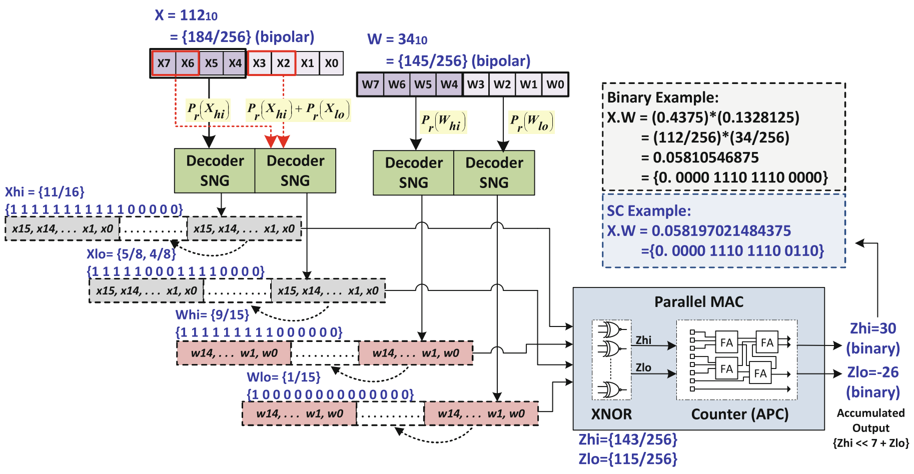

Fig. 15. Architecture of the proposed split SC-MAC. SNG is performed on the LSB and MSB of both the 8-bits input feature (X) and the kernel weight (W). SNGs generate deterministic SN of 256-bits using segments of 16-bits ($\times 15$) and 15-bits ($\times 16$) respectively. The parallel SC-MAC can be referred to Fig. 14 with $r = 32$. The output is converted back via bit-shifting and addition.

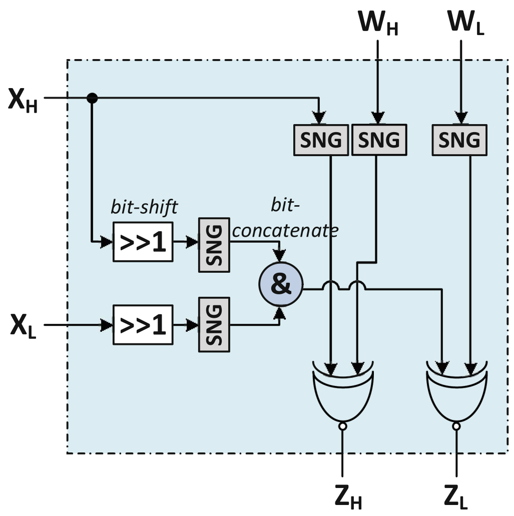

Fig. 16. Split SC and deterministic bipolar multiplication (using XNOR) for input $X = \{X_H, X_L\}$ and $W = \{W_H, W_L\}$. The bipolar multiplication is as derived in Eq. 4.

$$
\begin{aligned}
P_r(X) &= \{P_r(X_{hi}), P_r(X_{lo})\} \\
P_r(W) &= \{P_r(W_{hi}), P_r(W_{lo})\}
\end{aligned}
$$

$$
\begin{aligned}
P_r(Z) &= P_r(X) \cdot P_r(W) \\
&= \{P_r(X_{hi}) \cdot P_r(W_{hi}), P_r(W_{lo}) \cdot [P_r(X_{hi}) + P_r(X_{lo})]\} \qquad (4)
\end{aligned}
$$

SC Addition: Stochastic representation values are kept within the probability interval of either $[0, 1]$ or $[-1, 1]$ for unipolar and bipolar format respectively. Therefore, a typical SC addition/subtraction is implemented using multiplexer (with fixed select-and-scale) in order to keep the output value within the interval (see Fig. 17). Given an example where SC-MAC is used as a convolution engine for grayscale images with a filter of $N \times N$ kernel size, there are two potential problems that can be identified. First, there will be an inevitable precision lost as the output can only be scaled up to the closest factor of $\lceil log_2(N \times N)\rceil$. Furthermore, the accuracy will be affected due to the loss of $n - 1$ information [19]. Second, if there are several zero value operands throughout the MAC computation, using a multiplexer with constant selector as the SC scaled-adder will end up over scaling in the accumulated output.

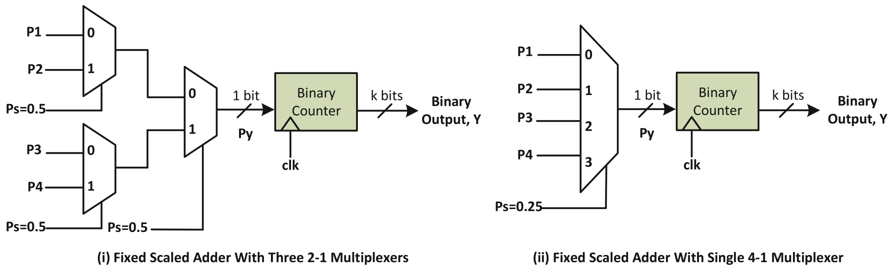

Fig. 17. Conventional SC fixed scaled adders for 4-operands.

With that, using a parallel counter (see Fig. 18 (i)) will guarantee accurate accumulation but it is consisted of array of full adders (FA). FA uses binary adder logic circuit and hence it is relatively high in hardware cost [17,19]. Therefore, in this study, we utilize Approximate Parallel Counter (APC) [17] which has reduced number of FA components (see Fig. 18 (ii)). The computation fulfils the same counting function but using less area and power consumption compared to the parallel counter. However, there is a slight trade-off in the accuracy, which is acceptable for approximation computing such as SC. Unlike multiplexers, APC enables us to have the flexibility to scale the accumulated values (i.e. count-and-scale) which the factor can be fine-tuned to suit different applications. Not only that, since the output of the APC is already in binary domain, the stochastic-to-binary conversion/de-randomizer is no longer needed.

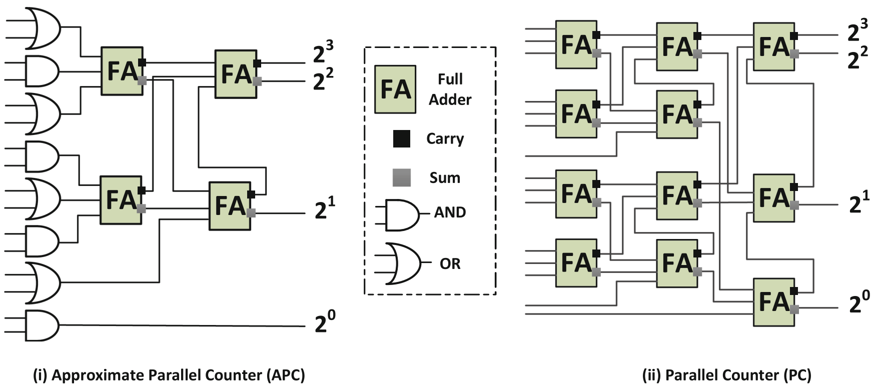
FA
2^3
2^2
2^1
2^0
FA Full Adder
Carry
Sum
AND
OR
(i) Approximate Parallel Counter (APC)
(ii) Parallel Counter (PC)

Fig. 18. Converter/De-randomizer to convert SN bit-stream to binary number using counter [17]. The example shown is to convert 16-bit SN to 4-bit binary using (i) Approximate Parallel Counter (ii) Parallel counter.

6 Experimental Result and Analysis

In this section, we discuss the evaluation result of our proposed SC-MAC unit in terms of its functionality as convolution engine, as well as its performance in hardware implementation. In doing so, our SC-MAC is integrated in CNN and the network's accuracy performance in MNIST classification is analyzed. Therefore, a total of $70,000$ MNIST data samples with each is 28×28 of handwritten digit images in grayscale are used for classification. The CNN topology used in our test case is as the following.

The first hidden layer is a Convolution layer that has 32 filters and all of the filters have 3×3 sliding window with a stride of 1. This is followed by a Max pooling layer with pool size of 2×2. Regularization layer (Dropout) is configured to randomly exclude 25% of the neurons in the layer to avoid over-fitting. This is followed by a Flatten layer that converts 2D convolution matrix data to 1D data, which will then be connected to Fully Connected layer. Dropout layer of 30% is used after that and finally, the output layer has 10 neuron for 10 classes. Using the proposed SC-MAC in the Convolution layer, the accuracy obtained from MNIST classification is 98.2%.

Furthermore, we benchmark our SC-MAC design with the existing works in terms of power efficiency (TOPS/W) and energy per operation (pJ/MAC). For comprehensive hardware analysis, our design is synthesized using typical libraries of UMC 40 nm technology. In Table 2, the works presented in [26] and [27] are the only work besides ours that implemented SC-MAC without using random sources.

The work in [26] presented a new SC multiplication algorithm which is also known as vectorized multiplication (BISC-MVM). In this design, the SNG utilized a finite state machine (FSM) and a multiplexer to generate deterministic SN with bit shuffling pattern. The work is further extended to the implementation in Fully Connected layer and has successfully achieved performance improvement

Table 2. Hardware evaluation and comparison for different SC-CNN/DNN accelerator designs. For benchmarking purposes, the measurement metrics are normalized to 40 nm process node.

Design	Accuracy (%)	Frequency (MHz)	Delay (ns)	Power (mW)	Energy (pJ)	Power Eff. (GMACS/mW)	Energy/MAC (pJ/MAC)	Contribution Descriptions
Work [5] (2017)	97.7	400	650	2.80	1.82×10^6	0.18	3.446	Integral SC, shared LFSR, quasi-synchronous
Work [24] (no.6) (2017)	98.3	200	1,280	3,138	4.0×10^6	0.57	1.744	APC-based inner product, max pooling
Work [24] (no.11) (2017)	96.6	200	1,280	1,360	1.78×10^6	1.32	0.775	APC/MUX-based inner product, average pooling
Work [31] (2017)	99.1	1,667	0.60	3,704	2,222	0.03	33.069	SNG sharing, unipolar SC, SC-based ReLu & Max
Work [15] (2016)	97.6	1,000	5.6	3.20	16.0	11.16	0.085	Shared LFSR, FSM activation, EDT, weight scaling
Work [26] (2017)*	98.5	1,000	1.5	22.28	33.3	7.89	0.126	BISC-MVM, bit shuffling FSM-SNG, CIFAR-10
Work [27] (2019)*	98.2	1,000	1.84	11.74	21.61	11.85	0.084	Extension from [26] to FC layer
This Work*	98.2	1,000	8.0	2.88	23.04	**12.50**#	**0.080**	Decoder-SNG, deterministic & split SC-MAC

*SC-MAC implementation without random source (RNG/PRNG)
#1 MAC is equivalent to 2 OPs and hence our power eff. is equal to 25TOPS/W

[27]. The study performed quantitative analysis that is inclusive of the end-to-end SC computation only. In addition, the work in [15] proposed LFSR-based SNG and the SNGs are effectively shared in the parallel computation. Despite the fact that the proposed SNG sharing approach does not lead to data correlation, the generated SNs may be susceptible towards random fluctuation error. In addition to that, the reported accuracy is lower compared to our work.

With this study, we have further proven the feasibility of utilizing deterministic sequence for SC. The presented bipolar decoder-based SNG is able to generate deterministic and uncorrelated SN which attains high precision progression with shorter bit-length compared to the conventional SNG. The generated SN bit-stream is free from random fluctuations error for both the zero and near-zero bipolar representation. Furthermore, our SNG is the most compact in size and has the lowest power consumption compared to the existing deterministic SNG in digital domain [13,21]. As the SN is generated instantaneously in our design, this enable further latency reduction through parallelism and split mechanism in our SC-MAC design. Overall, our proposed SC-MAC design attained the highest power efficiency (12.5 GMACS/mW or 25TOPS/W) and the lowest energy per MAC operation (80 fJ/MAC) as compared to the prior arts.

7 Conclusion

In summary, we presented a power-efficient deterministic SC-MAC unit, that is suitable to be deployed as a lightweight convolution engine. This SC-MAC utilizes decoder-based SNG that is capable of generating deterministic and uncorrelated SN bit-streams without using random source. Furthermore, the new SNG requires significantly shorter bit-length to accurately encode bipolar SN without random fluctuation. As the SNs are generated instantaneously, the subsequent computation latency can be reduced effectively and this leads to energy savings in the proposed SC-MAC. Our work incorporated parallelism and split mechanism in the presented SC-MAC unit in order to improve the power efficiency of the design without incurring excessive cost. We further demonstrated the proposed SC-MAC as convolution engine in CNN and its functionality is tested in MNIST classification. The experimental results proved that our deterministic SC-MAC surpasses the existing designs in terms of GMACS/W and fJ/MAC metrics.

References

1. Alaghi, A., Hayes, J.P.: Exploiting correlation in stochastic circuit design. In: 2013 IEEE 31st International Conference on Computer Design (ICCD), pp. 39–46 (2013). https://doi.org/10.1109/ICCD.2013.6657023
2. Alaghi, A., Hayes, J.P.: Survey of stochastic computing. ACM Trans. Embed. Comput. Syst. **12**(2), 92:1-92:19 (2013). https://doi.org/10.1145/2465787.2465794
3. Alaghi, A., Li, C., Hayes, J.P.: Stochastic circuits for real-time image-processing applications. In: Proceedings of the 50th Annual Design Automation Conference. DAC 2013, Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2463209.2488901

4. Andrej, K., George, T., Sanketh, S., Thomas, L., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR (2014)
5. Ardakani, A., Leduc-Primeau, F., Onizawa, N., Hanyu, T., Gross, W.J.: VLSI implementation of deep neural network using integral stochastic computing. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **25**(10), 2688–2699 (2017). https://doi.org/10.1109/TVLSI.2017.2654298
6. Brown, B., Card, H.: Stochastic neural computation. ii. soft competitive learning. IEEE Trans. Comput. **50**(9), 906–920 (2001). https://doi.org/10.1109/12.954506
7. Chippa, V.K., Venkataramani, S., Roy, K., Raghunathan, A.: Storm: a stochastic recognition and mining processor. In: 2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp. 39–44 (2014)
8. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning, pp. 160–167. ICML 2008, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1390156.1390177
9. Gaines, B.R.: Stochastic computing. In: Proceedings of the 18–20 April 1967, Spring Joint Computer Conference, pp. 149–156. AFIPS 1967 (Spring), ACM, New York, NY, USA (1967). https://doi.org/10.1145/1465482.1465505
10. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. CoRR abs/1506.02626 (2015). http://arxiv.org/abs/1506.02626
11. Ichihara, H., Ishii, S., Sunamori, D., Iwagaki, T., Inoue, T.: Compact and accurate stochastic circuits with shared random number sources. In: 2014 IEEE 32nd International Conference on Computer Design (ICCD), pp. 361–366 (2014). https://doi.org/10.1109/ICCD.2014.6974706
12. Ichikawa, K., Yamashita, S.: A multiply accumulator for stochastic numbers without scaling errors. In: 2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID), pp. 88–93 (2021). https://doi.org/10.1109/VLSID51830.2021.00020
13. Jenson, D., Riedel, M.: A deterministic approach to stochastic computation. In: 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1–8 (2016)
14. Karen, S., Andrew, Z.: Very deep convolutional networks for large-scale image recognition (2015)
15. Kim, K., Kim, J., Yu, J., Seo, J., Lee, J., Choi, K.: Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks. In: 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6 (2016). https://doi.org/10.1145/2897937.2898011
16. Kim, K., Lee, J., Choi, K.: An energy-efficient random number generator for stochastic circuits. In: 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 256–261 (2016). https://doi.org/10.1109/ASPDAC.2016.7428020
17. Kim, K., Lee, J., Choi, K.: Approximate de-randomizer for stochastic circuits. In: 2015 International SoC Design Conference (ISOCC), pp. 123–124 (2015). https://doi.org/10.1109/ISOCC.2015.7401667
18. Li, J., Ren, A., Li, Z., Ding, C., Yuan, B., Qiu, Q., Wang, Y.: Towards acceleration of deep convolutional neural networks using stochastic computing. In: 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 115–120 (2017). https://doi.org/10.1109/ASPDAC.2017.7858306

19. Li, J., et al.: Normalization and dropout for stochastic computing-based deep convolutional neural networks. Integration **65**, 395–403 (2019). https://doi.org/10.1016/j.vlsi.2017.11.002, https://www.sciencedirect.com/science/article/pii/S0167926017302328
20. Liu, Y., Wang, Y., Lombardi, F., Han, J.: An energy-efficient online-learning stochastic computational deep belief network. IEEE J. Emerg. Sel. Top. Circ. Syst. **8**(3), 454–465 (2018). https://doi.org/10.1109/JETCAS.2018.2852705
21. Najafi, M.H., Jamali-Zavareh, S., Lilja, D.J., Riedel, M.D., Bazargan, K., Harjani, R.: Time-encoded values for highly efficient stochastic circuits. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **25**(5), 1644–1657 (2017). https://doi.org/10.1109/TVLSI.2016.2645902
22. Parhi, M., Riedel, M.D., Parhi, K.K.: Effect of bit-level correlation in stochastic computing. In: 2015 IEEE International Conference on Digital Signal Processing (DSP), pp. 463–467 (2015). https://doi.org/10.1109/ICDSP.2015.7251915
23. Qian, W., Li, X., Riedel, M.D., Bazargan, K., Lilja, D.J.: An architecture for fault-tolerant computation with stochastic logic. IEEE Trans. Comput. **60**(1), 93–105 (2011). https://doi.org/10.1109/TC.2010.202
24. Ren, A., et al.: SC-DCNN: highly-scalable deep convolutional neural network using stochastic computing. SIGOPS Oper. Syst. Rev. **51**(2), 405–418 (2017). https://doi.org/10.1145/3093315.3037746
25. Sainath, T.N., Mohamed, A.R., Kingsbury, B., Ramabhadran, B.: Deep convolutional neural networks for LVCSR. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8614–8618 (2013). https://doi.org/10.1109/ICASSP.2013.6639347
26. Sim, H., Lee, J.: A new stochastic computing multiplier with application to deep convolutional neural networks. In: 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6 (2017). https://doi.org/10.1145/3061639.3062290
27. Sim, H., Lee, J.: Cost-effective stochastic MAC circuits for deep neural networks. Neural Netw. **117**, 152–162 (2019). https://doi.org/10.1016/j.neunet.2019.04.017, http://www.sciencedirect.com/science/article/pii/S0893608019301236
28. Sousa, L.: Nonconventional computer arithmetic circuits, systems and applications. IEEE Circ. Syst. Mag. **21**(1), 6–40 (2021). https://doi.org/10.1109/MCAS.2020.3027425
29. Wong, M.M., Chen, L., Do, A.T.: A 25 TOPS/W high power efficiency deterministic and split stochastic MAC (SC-MAC) design. In: 2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC), pp. 1–6 (2021). https://doi.org/10.1109/VLSI-SoC53125.2021.9606972
30. Yang, M., Hayes, J.P., Fan, D., Qian, W.: Design of accurate stochastic number generators with noisy emerging devices for stochastic computing. In: 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 638–644 (2017). https://doi.org/10.1109/ICCAD.2017.8203837
31. Yu, J., Kim, K., Lee, J., Choi, K.: Accurate and efficient stochastic computing hardware for convolutional neural networks. In: 2017 IEEE International Conference on Computer Design (ICCD), pp. 105–112 (2017). https://doi.org/10.1109/ICCD.2017.24