Area-Efficient Scaling-Free DFT/FFT Design Using Stochastic Computing

Bo Yuan, Yanzhi Wang, and Zhongfeng Wang, *Fellow, IEEE*

***Abstract*—Discrete Fourier transform (DFT) is an important transformation technique in signal processing tasks. Due to its ultrahigh computing complexity as $O(N^2)$, N-point DFT is usually implemented in the format of fast Fourier transformation (FFT) with the complexity of $O(N \log N)$. Despite this significant reduction in complexity, the hardware cost of the multiplication-intensive N-point FFT is still very prohibitive, particularly for many large-scale applications that require large N. This brief, *for the first time*, proposes high-accuracy low-complexity scaling-free stochastic DFT/FFT designs. With the use of the stochastic computing technique, the hardware complexity of the DFT/FFT designs is significantly reduced. More importantly, this brief presents the scaling-free stochastic adder and the random number generator sharing scheme, which enable a significant reduction in accuracy loss and hardware cost. Analysis results show that the proposed stochastic DFT/FFT designs achieve much better hardware performance and accuracy performance than state-of-the-art stochastic design.**

***Index Terms*—Discrete Fourier transform (DFT)/fast Fourier transformation (FFT), random number generator (RNG) sharing, scaling-free, stochastic computing (SC).**

I. Introduction

AMONG various discrete transforms, discrete Fourier transformation (DFT) [1] is the most important technique that performs Fourier analysis in various practical applications, such as digital signal processing and wireless communications, to name a few. Due to its ultrahigh computing complexity as $O(N^2)$, in practice the N-point DFT is usually performed in the form of the fast Fourier transformation (FFT) [1] with complexity as $O(N \log N)$. Despite this significant reduction in computing complexity, the hardware cost of the multiplication-intensive N-point FFT can still be prohibitive, particularly for many large-scale applications that require large N.

To address this problem, many optimization techniques, such as folding and resource sharing [2], had been applied to develop area-efficient FFT designs [3]–[5]. In addition, for a fixed-size FFT, many multipliers can be customized as constant multipliers to reduce the area. However, the limitations of these prior efforts impede their application in general cases and further optimization on the area. For instance, the use of the folding or resource-sharing technique is not suited for the cases when the time-domain data is continuously input to the FFT design since no tradeoff between the area and time can be made in this scenario. In addition, the strategy of largely using N-dependent constant multipliers cannot be adopted when highly flexible/reconfigurable FFT designs are required. To sum up, the exploration of general methodology to reduce the hardware complexity of DFT/FFT is nontrivial and very challenging.

Recently, *stochastic computing* (SC) [6], [7], as an emerging computing technique, has shown its potential application for area-efficient DFT/FFT design [8]. Deviated from the conventional FFT designs that are based on the 2's complement computing (referred as *binary computing*), the design in [8] is based on the SC that has very simple arithmetic units. In particular, the original high-complexity multiplier in binary computing can now be implemented with a simple AND gate in the scenario of SC, thereby making SC very attractive as the solution for area-efficient FFT design.

However, the stochastic design in [8] is not as area efficient as expected due to two reasons. First, the design in [8] is targeted for DFT with $O(N^2)$, instead of FFT with $O(N \log N)$. This is because the precision loss incurred by the downscaling effect of the stochastic adder is much more severe in the butterfly-based FFT architecture than that in the inner product-based DFT architecture. Second, the required N copies of the binary-to-stochastic (B-to-S) conversion units consume a large amount of hardware resource. Hence, to date, the state-of-the-art stochastic DFT/FFT design cannot achieve high accuracy and small footprint for large N cases simultaneously.

This brief, *for the first time*, proposes high-accuracy low-complexity scaling-free stochastic DFT/FFT designs. Joint optimizations on both fundamental stochastic arithmetic units and interface blocks are performed to improve the accuracy and the hardware performance of the proposed designs. The contribution of this brief is twofold. First, we propose a general methodology of designing scaling-free stochastic arithmetic unit, thereby enabling the long addition operation in DFT/FFT to retain high computing accuracy. Second, we propose a novel random number generator (RNG) sharing scheme that significantly reduces the hardware cost of the interface between the binary computing and SC domains. Based on the proposed approaches, example 256-point stochastic DFT/FFT designs are developed. Performance analysis shows that the proposed DFT designs achieve much better computation accuracy than the state-of-the-art stochastic DFT design. More importantly, to the best of our knowledge, the proposed FFT design is the first stochastic FFT design that has very high computation accuracy and very low hardware cost, thereby unlocking the potentiality of its widespread applications in the practical systems.

The rest of this brief is organized as follows. A brief review of DFT/FFT and SC is given in Section II. Section III first discusses the challenges of state-of-the-art stochastic DFT/FFT

Manuscript received May 14, 2016; revised August 15, 2016; accepted August 19, 2016. Date of publication August 26, 2016; date of current version November 22, 2016. This brief was recommended by Associate Editor C. W. Wu.

B. Yuan is with the Department of Electrical Engineering, City University of New York, New York, NY 10031 USA (e-mail: byuan@ccny.cuny.edu).

Y. Wang is with the Department of Electrical Engineering Computer Science, Syracuse University, Syracuse, NY 13244 USA (e-mail: ywang393@syr.edu).

Z. Wang is with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210093, China (e-mail: zfwang@nju.edu.cn).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2016.2603465

1549-7747 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: Zhejiang University. Downloaded on January 18,2021 at 07:10:55 UTC from IEEE Xplore. Restrictions apply.

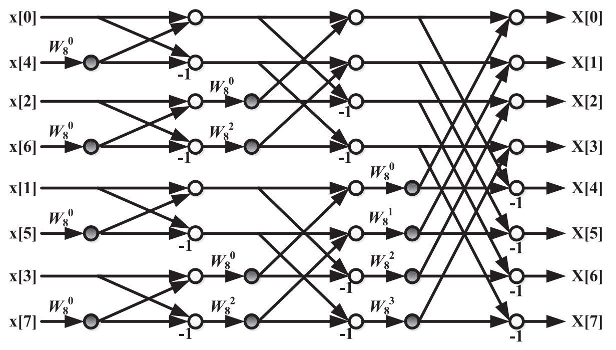


Fig. 1. Architecture of the 8-point FFT.

designs, and then proposes high-accuracy stochastic adder design and RNG sharing scheme. Hardware architectures of the proposed stochastic DFT/FFT are also presented in this section. The accuracy performance and hardware performance of the proposed stochastic designs are analyzed in Section IV. Section V draws the conclusion.

II. Background

A. DFT and FFT

As indicated in Section I, DFT is a very powerful signal processing technique to perform time-to-frequency domain transformation. In general, N-point DFT for a discrete signal $x[n]$ can be performed as

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn} \tag{1}$$

where $W_N = e^{-j(2\pi/N)}$, and $X[k]$ is the discrete output signal.

As shown in (1), the direct computation from the definition of DFT is very inefficient since N^2 times of computation are required to calculate all the $X[k]$, where $0 \le k \le N-1$. To address this problem, FFT was proposed to speed up the DFT computation. As shown in Fig. 1, by utilizing the recursive property of DFT, the butterfly-based FFT enables significant reduction in complexity from $O(N^2)$ to $O(N \log N)$. For the detail of derivation of FFT, the reader is referred to [1].

B. SC

SC is an emerging computing technique that has unique advantages in low hardware cost and high error resilience. Different from conventional binary computing, SC utilizes the ratio of bit "1" in a bitstream to represent the number. More specifically, in the *unipolar* format SC [6], [7], x within range [0, 1] can be represented by any length-L bitstream with X bits, where $x = X/L$ [see Fig. 2(a)]. Based on this representation, the fundamental arithmetic units can now be implemented with very simple logic gates. For instance, the original high-complexity multiplier can be replaced with a simple AND gate in the scenario of unipolar SC system [see Fig. 2(b)], thereby leading to significant savings in hardware complexity.

Considering the range of the unipolar SC is limited with a positive number, *bipolar* SC [6], [7] is a more practical SC representation since it can represent both positive and negative numbers: In the bipolar SC system, a number x within range $[-1, 1]$ can be represented by a length-L bit stream with X bits of "1," where $x = 2(X/L) - 1$ [see Fig. 2(d)]. Similar to the

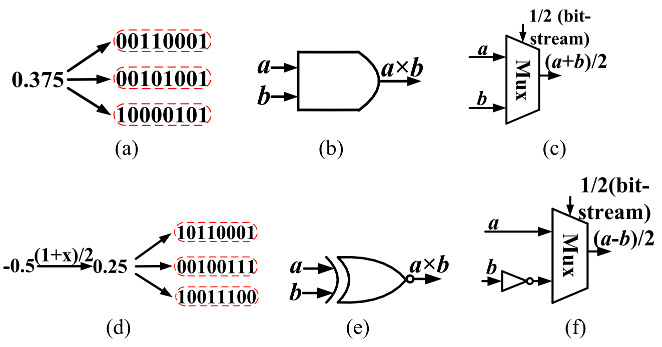


Fig. 2. (a) Sample unipolar representation. (b) Unipolar multiplication. (c) Scaled addition. (d) Sample bipolar representation. (e) Bipolar multiplication. (f) Scaled subtraction. Cited from [8].

case in unipolar SC, the stochastic circuits for multiplication in the bipolar SC is also very simple [see Fig. 2(e)]. Notice that the unipolar SC and the bipolar SC have the same stochastic circuits for scaled addition and subtraction [see Fig. 2(c) and (f)].

III. Proposed Stochastic DFT/FFT

A. Challenge of State-of-the-Art Stochastic DFT/FFT Design

Inspired by the inherent low-cost advantage of stochastic arithmetic units, the hardware design of stochastic FFT seems a very promising solution for area-efficient implementation of FFT. However, as indicated in [8], the N-size stochastic FFT suffers from severe computation accuracy loss incurred by the downscaled operation of stochastic adder. As shown in Fig. 2(c), the output of stochastic adder in either the unipolar or bipolar SC representation is the downscaled sum with scaling factor as 1/2. Considering the computation of each $X[k]$ requires N times of stochastic addition, the actual kth output of the stochastic FFT, referred to as $X_{\text{act}}[k]$, is $X_{\text{act}}[k] = X[k]/2^N$. Such largely downscaled output $X_{\text{act}}[k]$ is too small that it cannot be represented in any SC system with a moderate length of bitstream, thereby causing a very severe computation accuracy loss. For instance, the simulation results in [8] showed that in the bipolar SC system with length-1024 bitstream, the accuracy loss of an 8-point stochastic FFT is already very severe [signal-to-noise ratio (SNR) (SNR) < -10 dB]. Considering in practical systems N is usually much larger than 8, the imprecision of the final output of stochastic FFT in large N case is even higher, thereby impeding any potential applications of stochastic FFT. Notice that here the accuracy loss incurred by the downscaled addition cannot be compensated by postprocessing $X_{\text{act}}[k]$, such as upscaling $X_{\text{act}}[k]$ by 2^N. This is because during the downscaled procedure, the information loss caused by the insufficient representation precision is permanent. Increasing the length of bitstream may be a potential solution to avoid the aforementioned precision loss; however, this approach causes an exponential increase in latency; hence, it is not suited for efficient hardware design. Similarly, the immediate stagewise compensation for each addition is also inefficient because the stochastic linear gain units for compensation introduce large overhead and additional inaccuracy source.

Impeded by the aforementioned series of problems for stochastic FFT, [8] proposed an uneven-multiplexor (MUX)-based stochastic DFT design. Although the use of an uneven MUX can alleviate the downscaling effect to some extent, the stochastic DFT in [8] still suffers very severe computation accuracy loss in large N cases. More importantly, due to the inherent

Authorized licensed use limited to: Zhejiang University. Downloaded on January 18,2021 at 07:10:55 UTC from IEEE Xplore. Restrictions apply.

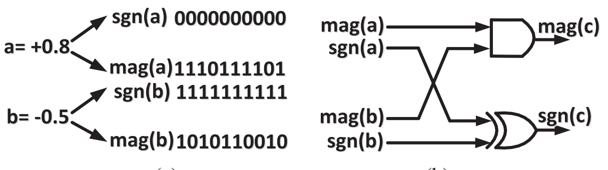


Fig. 3. (a) Sample two-line SC representation. (b) Two-line SC multiplier.

high computation complexity of DFT, the stochastic DFT is not a good candidate for area-efficient hardware design.

B. High-Accuracy Scaling-Free Stochastic Adder

The aforementioned analysis of the existing challenges of state-of-the-art stochastic DFT/FFT shows that an inherently scaling-free stochastic adder is the key to design efficient high-accuracy stochastic DFT/FFT. Motivated by this demand, this section presents the general methodology of a high-accuracy scaling-free stochastic adder. Next, we first introduce the underlying SC representation for this new stochastic adder and then present the design methodology.

Two-Line SC Representation: Different from the conventional downscaled stochastic adder that is based on either the unipolar or bipolar SC representation, the proposed scaling-free stochastic adder is based on the two-line SC representation [9]. As illustrated in Fig. 3(a), two bit-streams, instead of one, are used to interpret a number x. More specifically, the magnitude and sign information of x, are carried by a *magnitude stream* and a *sign stream*, respectively. If we denote $M(X_i)$ and $S(X_i)$ be the i-th bit of the magnitude stream and sign stream with length L, respectively, then x that is within $[-1,\ 1]$ can be represented as

$$x = \left(\frac{1}{L}\right) \sum_{i=0}^{L-1} \left(1 - 2S(X_i)M(X_i)\right). \tag{2}$$

Due to the difference in the underlying SC representation, the stochastic arithmetic circuits in this two-line SC scenario are different from the ones in Fig. 2. For instance, Fig. 3(b) shows the architecture of the two-line-based stochastic multiplier. Although it is seen that the use of this two-line SC representation introduces additional hardware for arithmetic units, this representation has a unique advantage for designing scaling-free stochastic adder, which is described next.

Design Methodology for Scaling-Free Stochastic Adder: Now considering the two numbers a and b and their sum c, the corresponding ith bits in the magnitude/sign bitstreams of a and b and c in this two-line SC representation are $M(A_i)/S(A_i)$, $M(B_i)/S(B_i)$ and $M(C_i)/S(C_i)$, respectively. Recall that in (2), each pair of $M(X_i)$ and $S(X_i)$ jointly contribute $X_i = (1 - 2S(X_i))M(X_i)$ to the final result; therefore, the contribution from two ith bits of the two bitstreams for c is $C_i = (1 - 2S(C_i))M(C_i)$. More specifically, C_i depends on the sum of $A_i = (1 - 2S(A_i))M(A_i)$ and $B_i = (1 - 2S(B_i))M(B_i)$. However, since C_i can be only the element within set $\{-1,\ 0,\ 1\}$, it is not the simple sum of A_i and B_i since we need to consider the effect of carry. For instance, when both A_i and B_i are 1 or -1, C_i can be only 1 or -1 and a carry "1" or "-1" should be propagated to the calculation of C_{i+1} to guarantee the valid calculation of sum. Aside from that, in order to retain the accuracy of stochastic addition, the probability of the occurrence of multiple carry bits should be considered since hundreds of bitwise additions are performed. In that scenario, many complicated cases, including the accumulation of the previous carry bits and the cancellation of "1" and "-1" carry bits should be also taken into account during the design of stochastic adder. Based on all the aforementioned design specifications, a general two-line-based stochastic addition scheme is developed as follows.

Scheme A: High-Accuracy Scaling-Free Stochastic Addition

```
1: Initialize $Counter = 0$, threshold value $\lambda$
2: Input: $M(A_i), S(A_i), M(B_i), S(B_i)$
3: $A_i = (1 - 2S(A_i))M(A_i), B_i = (1 - 2S(B_i))M(B_i)$
4: If $(A_i + B_i == 2)$ then $\{C_i = 1;\ Counter{+}{+}$ when
                                        $Counter < \lambda\}$
5: Else If $(A_i + B_i == 1)$ then $C_i = 1$
6:     Else If $(A_i + B_i == -1)$ then $C_i = -1$
7:         Else If $(A_i + B_i == -2)$ then
           $\{C_i = -1;\ Counter{-}{-}$ when $Counter > -\lambda\}$
8:             Else If $(Counter > 0)$ then
                                        $\{C_i = 1; Counter{-}{-}\}$
9:                 Else If $(Counter < 0)$ then
                                        $\{C_i = -1; Counter{+}{+}\}$
10:                    Else $C_i = 0$
11: If $(C_i == 1)$ $\{M(C_i) = 1;\ S(C_i) = 0\}$
12: Else If $(C_i == -1)$ $\{M(C_i) = 1;\ S(C_i) = 1\}$
13:     Else $\{M(C_i) = 0;\ S(C_i) = 0\}$
14: Output: $M(C_i), S(C_i)$
```

In the aforementioned scheme, a preset threshold value λ is used to define the maximum number of carry bits allowed in the proposed stochastic adder. Based on the current sum of A_i and B_i, C_i is generated as one element within $\{-1,\ 0.\ 1\}$. Meanwhile, the value of a small counter, which stores the number of the existing unused carry bits, also changes according to the value of C_i and the current value of the counter. To sum up, such small counter releases or holds the carry bits dynamically to ensure that the carry effect occurred during the bitwise addition can be taken care as much as it can, thereby retaining the high computation accuracy of the entire addition. Notice that here, since λ bounds the allocating capability of the counter (see line 4 and line 7 in Scheme A), the accuracy of the adder can be adjusted by setting different values of λ. Our simulation results show that $\lambda = 1$ or 2 is sufficient to guarantee the accuracy of the stochastic adder in large-scale applications. For some applications that need ultrahigh computation accuracy, a larger λ can be selected to offer the required accuracy.

Aside from the property of dynamic accuracy configuration, the most attractive advantage of the proposed two-line stochastic adder is its scaling-free behavior. This is because the bitwise addition for the component bitstreams of the adders' inputs is nonscaled and similar to 1-bit full adder, thereby enabling that the entire addition is completely scaling-free. In Section III-D, this scaling-free adder is used to build the scaling-free stochastic DFT/FFT designs.

It should be noticed that the proposed scaling-free stochastic adder (see Fig. 4) has higher hardware complexity than the downscaled version in Fig. 2(c). However, this overhead is worthwhile due to two reasons. First, the severe accuracy loss incurred by the downscaled adder is the bottleneck of existing

Authorized licensed use limited to: Zhejiang University. Downloaded on January 18,2021 at 07:10:55 UTC from IEEE Xplore. Restrictions apply.

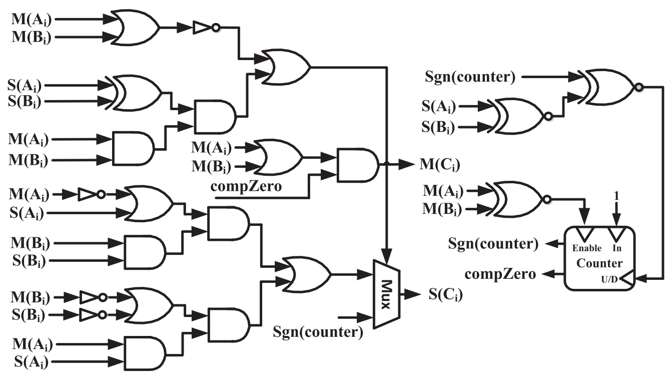


Fig. 4. Architecture of the scaling-free stochastic adder.

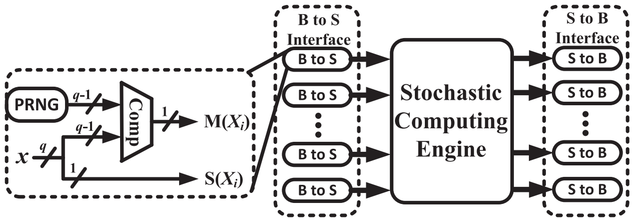


Fig. 5. Complete SC system with the computing engine and interface.

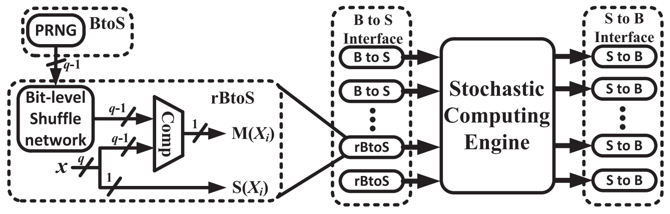


Fig. 6. Proposed RNG sharing scheme in the stochastic DFT/FFT.

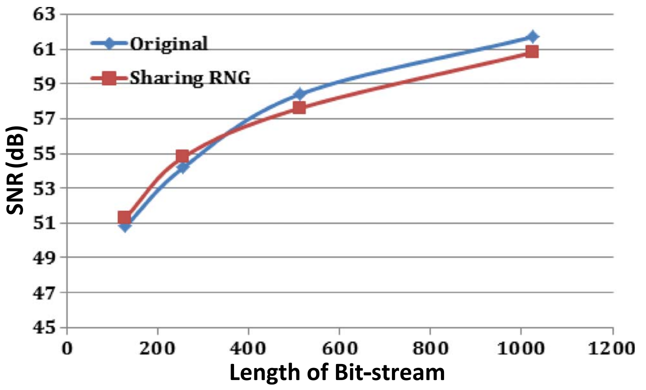


Fig. 7. SNR performance with the use of the RNG sharing scheme.

stochastic DFT/FFT designs. Second, this two-line SC system has sufficient area budget to make tradeoff for higher accuracy.

C. RNG Sharing Scheme

In general, a complete SC system contains SC engines that perform main computing tasks and interface blocks that perform conversion between the SC and binary computing domains. As indicated in [10], in some applications, those interface blocks may occupy a large amount of silicon area and thus eliminate the benefit of low hardware cost offered by the SC engine. Unfortunately, the DFT/FFT is just one of such applications. For an N-point DFT/FFT, a straightforward design requires N copies of B-to-S conversion block that contains pseudo RNG (PRNG) (see Fig. 5). As a result, N copies of high-complexity PRNGs are required to implement a stochastic DFT/FFT, thereby causing significant increase in area.

To address this challenge, this section proposes an efficient RNG sharing scheme. The key idea is to use the randomness offered by very few RNGs to generate the necessary uncorrelated randomness for all input bitstreams. More specifically, as shown in Fig. 6, only two RNGs are equipped in the entire stochastic DFT/FFT design to offer the randomness. In each cycle, aside from the pseudorandom number that is generated by its own architecture, the two RNGs can also generate other pseudorandom number by shuffling their output bits. In general, a $(q-1)$-data-width linear feedback shift register-based PRNG can generate $2^{q-1}-1$ pseudorandom number beyond its own output with the use of this shuffling strategy. As a result, the number of the required RNGs is reduced from N to 2, thereby leading to significant reduction in hardware cost of interface blocks.

It should be noted that the proposed RNG sharing scheme does not introduce severe correlation, which is a main factor that degrades the computation accuracy of general SC systems. This is because the proposed bit-level shuffling is a nonlinear transformation (“interleaving”) instead of simple addition or subtraction to the original random number, thereby breaking the correlation between the pre- and postshuffled random numbers. Simulation results also verify this point. Fig. 7 shows the use of the proposed RNG sharing scheme in the stochastic inner product, which is the kernel operation in the stochastic DFT/FFT (a butterfly unit in FFT can be viewed as inner product). It is seen that such reduced use of RNGs retains the similar computation accuracy in terms of SNR. In addition, if the system has very stringent requirement on correlation, the low-cost bitstream splitting and shuffling technique in [11] can be applied and combined with this RNG sharing scheme to achieve even lower correlation.

In general, the proposed RNG sharing scheme is not only suited for the stochastic DFT/FFT design but can be also widely applied in many other multi-input SC systems, including image processing, machine learning, etc. In addition, the proposed sharing scheme is not only limited in the scenario of two-line SC representation but is also compatible for the conventional unipolar and bipolar SC representations.

D. Overall Architecture of Stochastic DFT/FFT

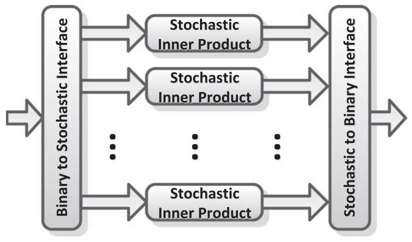
Based on the two-line-based stochastic adder and multiplier, the overall architectures of the stochastic DFT and FFT designs are developed, as shown in Fig. 8(a) and (b), respectively. Here, the stochastic inner product and butterfly unit, as defined in their mathematic format, consist of the fundamental stochastic adder, subtractor, and multiplier. Notice that the design of the two-line stochastic subtractor is very similar to that of stochastic adder via preinverting $S(B_i)$ when $M(B_i)$ is 1.

IV. Performance Analysis

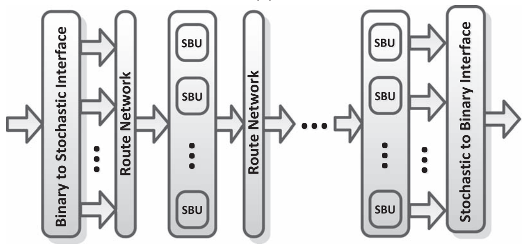
A. Computation Accuracy

As indicated in Section I, the insufficient computation accuracy is the bottleneck of the applications of the stochastic

Authorized licensed use limited to: Zhejiang University. Downloaded on January 18,2021 at 07:10:55 UTC from IEEE Xplore. Restrictions apply.



(a)


Binary to Stochastic Interface
Route Network
SBU
Route Network
SBU
Stochastic to Binary Interface

(b)

Fig. 8. Architectures for (a) stochastic DFT and (b) stochastic FFT.

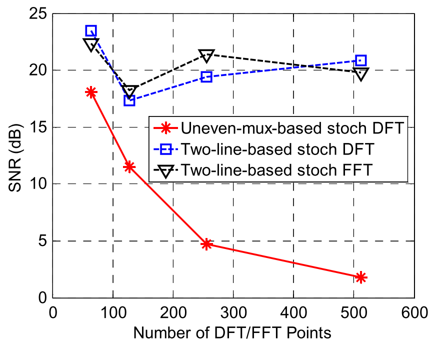


Fig. 9. SNR performance of the stochastic DFT/FFT with a length-1024 bitstream.

DFT/FFT in the practical systems. Fig. 9 shows the SNR performance of different stochastic DFT/FFT designs. It is seen that, as compared with the prior uneven-MUX-based stochastic DFT design in [8], the proposed two-line-based stochastic DFT and FFT have much higher computation accuracy in terms of SNR due to the use of scaling-free stochastic adder. In particular, it should be noted that the computation accuracy of the proposed two designs does not show the degrading behavior as N increases, thereby implying that the proposed designs are well suited for practical large-scale applications.

B. Hardware Performance

Table I summarizes the hardware performance of the different DFT/FFT implementations. Here, the size of N is set as 256 for all the designs. Different stages of pipelining are applied to the different designs to achieve high clock rate. Notice that here all the designs are implemented in the same systolic-based [2] architecture without use of any folding for fair comparison. In addition, all the implemented multipliers are set as general multipliers to enable the maximum generality of the listed FFT/DFT designs for different configurations.

From this table, it can be observed that, although the proposed stochastic DFT design has higher hardware complexity than the DFT design in [8], it is still the first stochastic DFT design that is suited for real systems with high computation accuracy. In addition, the proposed two-line FFT design has significantly lower hardware cost and accuracy loss than the stochastic DFT design in [8]. More importantly, as compared with the conventional DFT/FFT designs with the same systolic-based architecture, the proposed stochastic DFT/FFT designs enable a significant reduction in hardware cost, thereby making themselves very attractive candidates for area-efficient DFT/FFT architectures in the practical large-scale signal processing systems.

TABLE I
HARDWARE PERFORMANCE OF THE STOCHASTIC DFT/FFT WITH $\lambda = 1$

Design	Proposed Two-line FFT	Proposed Two-line DFT	Uneven MUX-based DFT [8]	Conv. FFT	Conv. DFT
SC Scheme	Two-line	Two-line	One-line	N/A	N/A
Pipeline Stage	2	3	1	3	3
Tech.(nm)	CMOS 90nm				
Clock Frequency	600MHz				
Gate Counts	172K	2540K	1050K	720K	38740K

V. CONCLUSION

This brief has presented the area-efficient scaling-free stochastic DFT/FFT designs. With the use of the proposed scaling-free stochastic adder and the RNG sharing scheme, the computation accuracy loss and overall hardware complexity are significantly reduced, thereby paving the way for the widespread applications of stochastic DFT/FFT in practical systems.

REFERENCES

[1] A. Y. Oppenheim *et al.*, *Discrete-Time Signal Processing*, vol. 2. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.
[2] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York, NY, USA: Wiley, 1999.
[3] M. Ayinala, M. J. Brown, and K. K. Parhi, “Pipelined parallel FFT architectures via folding transformation,” *IEEE Trans. VLSI Syst.*, vol. 20, no. 6, pp. 1068–1081, Jun. 2012.
[4] M. Ayinala, Y. Lao, and K. K. Parhi, “An in-place FFT architecture for real-valued signals,” *IEEE Trans. Circuits and Systems-II: Transactions Briefs*, vol. 60, no. 10, pp. 652–656, Oct. 2013.
[5] M. Ayinala and K. K. Parhi, “FFT architectures for real-valued signals based on radix-2^3 and radix-2^4 algorithms,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 9, pp. 2422–2430, Sep. 2013.
[6] B. Gaines, “Stochastic computing systems,” *Adv. Inf. Syst. Sci.*, vol. 2, no. 2, pp. 37–172, 1969.
[7] B. Brown and H. Card, “Stochastic neural computation I: Computational elements,” *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891–905, Sep. 2001.
[8] B. Yuan, Y. Wang, and Z. Wang, “Area-efficient error-resilient discrete Fourier transformation design using stochastic computing,” in *Proc. ACM 26th GLSVLSI*, 2016, pp. 33–38.
[9] S. L. Toral, J. M. Quero, and L. G. Franquelo, “Stochastic pulse coded arithmetic,” in *Proc. IEEE ISCAS*, May 2000, pp. 599–602.
[10] P. Knag, W. Lu, and Z. Zhang, “A native stochastic computing architecture enabled by memristors,” *IEEE Trans. Nanotechnol.*, vol. 13, no. 2, pp. 283–293, Mar. 2014.
[11] B. Yuan, C. Zhang, and Z. Wang, “Design space exploration for hardware-efficient stochastic computing: A case study on discrete cosine transformation,” in *Proc. IEEE ICASSP*, 2016, pp. 6555–6559.

Authorized licensed use limited to: Zhejiang University. Downloaded on January 18,2021 at 07:10:55 UTC from IEEE Xplore. Restrictions apply.