DESIGN SPACE EXPLORATION FOR HARDWARE-EFFICIENT STOCHASTIC COMPUTING: A CASE STUDY ON DISCRETE COSINE TRANSFORMATION

Bo Yuan[1], Chuan Zhang[2] and Zhongfeng Wang[3]

[1] Department of Electrical Engineering, City University of New York, City College
[2] National Mobile Communication Research Laboratory, Southeast University
[3] Broadcom Corporation

ABSTRACT

In recent years stochastic computing (SC) is re-gaining increasing attention for its unique advantages on low hardware cost and strong error resilience that are the key metrics for nanoscale CMOS era. However, the potential deployment of SC in practical applications is impeded by the long latency of sequential bit-stream and large complexity of pseudo random number generator (PRNG). Aiming to mitigate these challenges, this paper exploits the design space for hardware-efficient stochastic computing with a case study on 4-point discrete cosine transformation (DCT). First, an efficient compensation mechanism is proposed to solve the scaling problem of SC system. Then, two approaches, namely Splitting-Shuffling (SS) and PRNG sharing techniques are proposed to reduce the overall area and processing latency, respectively. Analysis results show that, sustaining the same computing accuracy, the joint use of the proposed approaches leads to 44% reduction in area and 49% reduction on latency than conventional SC design, respectively.

Index Terms— Stochastic Computing, DCT, Splitting and Shuffling (SS)

1. INTRODUCTION

Modern digital computing systems are built on number representation. Although a number can be represented in various ways, to date all the digital computing processors adopt weighted binary representation for interpreting and processing information. Generally in this binary computing (BC) scenario a number that is represented by n bits has the precision as $1/2^n$.

Different from the above conventional binary computing, stochastic computing (SC) [1-2] represents the number with the use of bit-stream. Here the value of number is interpreted by the portion of bits "1" over the entire stream. Based on this new representation scheme, the corresponding stochastic arithmetic units that function approximately can be developed with very simple logic gates. For instance, the stochastic multiplication can be easily implemented with a simple logic gate (AND or XNOR). This ultra-low hardware cost may lead to significant reduction in both computation and time complexity. In addition, the inherent redundancy in number representation of SC also translates to stronger fault tolerant capability that is a key metric in the current nanoscale CMOS and emerging post-CMOS eras.

This work is partially supported by NSFC under grant 61501116, International Science & Technology Cooperation Program of China under grant 2014DFA11640, Huawei HIRP Flagship under grant YB201504, Jiangsu Provincial NSF under grant BK20140636, and ICRI-MNC.

Despite the above potential advantages of SC, two severe challenges pose obstacles to its practical applications. First, in order to achieve the same $1/2^n$ precision, BC only needs n-bit-width data, while SC needs length-2^n bit-stream, thereby causing extreme long processing latency. Second, the pseudo random number generator (PRNG) that provides the randomness of the bit-stream has very large area, and hence overtakes the original simplicity of SC system.

Aiming to mitigate these challenges, this paper exploits the design space for hardware-efficient SC system and proposes a set of approaches. First, an efficient compensation mechanism is proposed to tackle the scaling problem of SC system. Then, two approaches, namely *S*plitting-Shuffling (SS) and PRNG sharing techniques are presented to reduce the overall area and processing latency of SC system, respectively. The proposed approaches are applied to 4-point discrete cosine transformation (DCT) as case study. Analysis results show that the joint use of the proposed approaches lead to 44% reduction in area and 49% reduction on latency than conventional SC design, respectively without accuracy loss.

The rest of this paper is organized as follows. Section 2 gives brief review of SC and DCT. The proposed approaches are presented and applied to DCT in Section 3. Section 4 analyzes the computation accuracy and hardware performance of stochastic DCT. The conclusions are drawn in Section 5.

2. REVIEW OF SC AND DCT

2.1. Stochastic Computing

As indicated in Section 1, SC system utilizes a stream of bits to represent number. Fig. 1(a) illustrates such representation of 0.625 with a length-8 bit-stream. Notice that in this example because each bit weights equally, a number can be represented by different streams.

Based on this redundant representation, various basic arithmetic functions can now be implemented via stochastic circuits. Fig. 1 (b)(c)(e)(g) shows the corresponding stochastic implementations for different functions. In

addition, the conversion blocks between binary representation and stochastic representation are shown in Fig. 1(d) and Fig. 1(f), respectively. Notice that here bipolar form [1-2] is adopted for SC system since the studied DCT in this paper involves with negative number. For the details of these basic SC circuits, the reader is referred to [1-2].

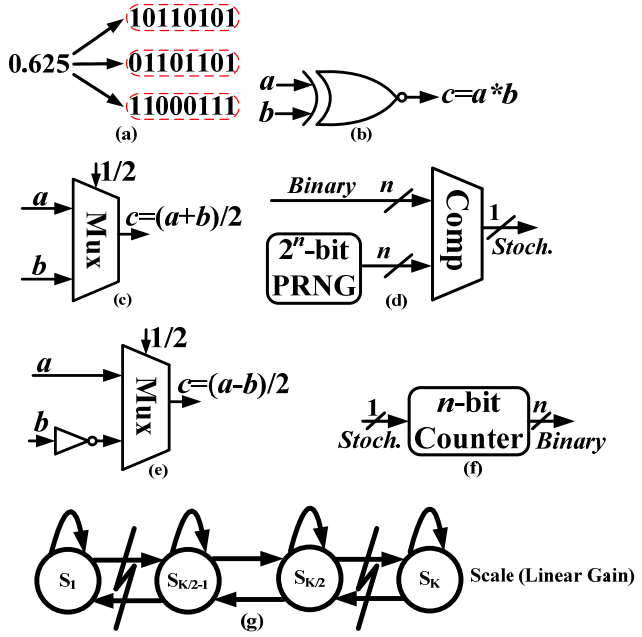

Fig. 1. SC system: (a) Representation (b) Multiplication (c) Scaled addition (d) Binary to stochastic (B2S) conversion (e) Scaled subtraction (f) Stochastic to binary (S2B) conversion (f) Scale (linear gain).

2.2. Discrete Cosine Transformation

Proposed in 1974, discrete cosine transformation (DCT) [3] has become a powerful tool in information processing applications because of its unique energy compaction property. To date DCT is widely used in many audio and image-related scenarios, such as JPEG, MPEG, MP3 and so on.

According to its different definitions, DCT has various types. In this paper we choose the Type-II DCT (DCT-II) as the studied case. Generally, for an N-point DCT-II, the transformation function is shows in (1):

$$X(k) = a(k)\sum_{n=0}^{N-1} x(n)\cos[\frac{(2n+1)\pi k}{2N}],\ k=0,\ 1,\ \ldots,\ N\text{-}1 \qquad (1)$$

where $a(k)=1/\sqrt{2}$ if $k=0$; otherwise $a(k)=1$. For the detail of DCT, the reader is referred to [3].

3. STOCHASTIC DISCRETE COSINE TRANSFORMATION

In this section, with a case study on 4-point DCT, a set of approaches that enable the reduction on latency and area of SC system are proposed. First, we present a straightforward design of stochastic DCT. Then, various hardware-level optimizing techniques are developed to improve the hardware performance of the SC system.

3.1. A straightforward design of stochastic DCT

For the example 4-point DCT, the computation of (1) can be re-described via matrix transformation as shown in (2):

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \\ \cos\frac{\pi}{8} & \cos\frac{3\pi}{8} & -\cos\frac{3\pi}{8} & -\cos\frac{\pi}{8} \\ \cos\frac{\pi}{4} & -\cos\frac{\pi}{4} & -\cos\frac{\pi}{4} & \cos\frac{\pi}{4} \\ \cos\frac{3\pi}{8} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{8} & -\cos\frac{3\pi}{8} \end{bmatrix} \times \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad (2)$$

Notice that in (2) each row of the 4-by-4 transformation matrix is either systematic or anti-systematic. As a result, according to this matrix form, the 4-point stochastic DCT can be developed with a butterfly architecture [4] using the basic stochastic blocks in Fig. 1. Here since each multiplier (**Mul**) in Fig. 1(b) has one constant input as the entry of matrix in (2), the signal line for that fixed input is not depicted in this figure. Notice that these constant bit-streams can be generated by simple logic instead of complex PRNGs.

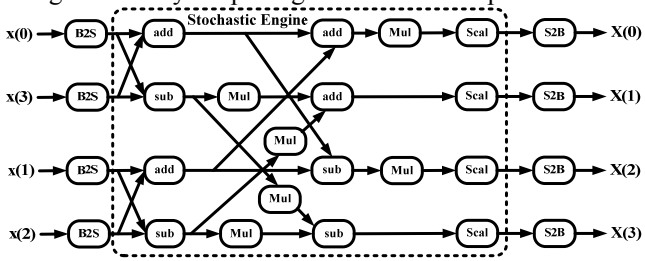

Fig. 2. Straightforward architecture of SC DCT.

3.2. Scaling problem

As indicated in Fig. 2, a stochastic scale block (see Fig. 1(g)) is needed for each output of DCT since the stochastic addition or subtraction (see Fig. 1(c) and (d)) is the scaled version. However, the use of such stochastic scale block causes two problems: 1) This finite state machine (FSM)-based block needs a number of states to guarantee the accuracy, thereby increasing the hardware cost of the entire system. 2) With the sequential logic, the stochastic scale block has escalating accuracy loss as the linear gain increases. Since the linear gain that is needed in N-point DCT is $2^{\wedge}(\log_2 N)=N$, that means for large N cases the accuracy loss caused by this stochastic scale unit is very severe.

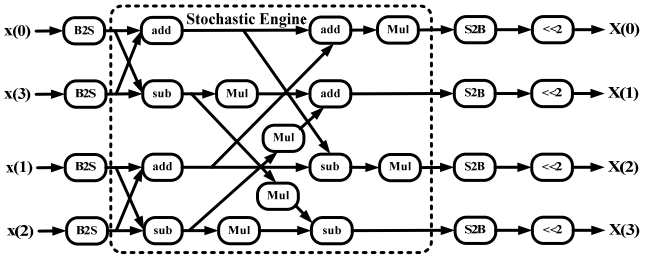

Fig. 3. SC DCT with new compensation scheme.

In order to tackle this challenge, we propose a new scaling scheme for stochastic DCT. Fig, 3 shows the architecture of the new design for 4-piont DCT. The key idea of this approach is to perform the stochastic-to-binary

(S2B) conversion earlier than the straightforward design, and then use the shift operation to scale the outputs. As a result, the original complex FSM-based scale blocks are replaced by the wire-based shifting circuits with very low hardware cost. In addition, the original accuracy loss caused by the scale unit is now avoid at the same time.

3.3. Splitting-Shuffling (SS)

3.3.1. Splitting and shuffling bit-stream

Existing research results [5-9] have shown that one key factor that affects the computation accuracy of SC system is its inherent correlation. In general, two types of correlation, as inter-stream and intra-stream correlations, occur when the bit-streams propagate over the SC system. In order to eliminate these correlations and avoid the accuracy loss, re-randomizing the bit-stream is a common and critical strategy for devising high-performance SC system. A straightforward re-randomizing approach is to utilize a pair of S2B and binary-to-stochastic (B2S) blocks for each target stream, and then makes the streams re-gain the randomness. However, this method requires a large amount of additional S2B and B2S blocks, thereby overtaking the simplicity of stochastic engine. Even worse, this approach also leads to significant increase in latency since extra $m=2^n$ cycles are needed for the additional S2B and B2S blocks, respectively. Consider SC system already suffers from long latency problem; this extra timing cost further worsens this drawback.

In this subsection we propose a novel *splitting and shuffling* (SS) technique to make bit-stream re-gain the randomness with very small cost. The key idea of the SS approach is to split the original bit-stream into multiple segments, and shuffle those segments when necessary. Fig. 4(a) shows an example SS scheme for length-8 bit-stream that is split to 4 segments. It can be seen that, with the use of the proposed approach, the new stream outperforms the original one in terms of switching activity that is an important metric to measure the randomness of bit-stream. More importantly, this SS technique offers substantial benefits on hardware performance: 1) Since this shuffling network only needs extra wire, the hardware cost of the SS approach is extremely low. 2) Segmenting the bits-stream leads to immediate linear reduction in latency, thereby mitigating this long-standing problem for SC systems.

3.3.2. Selection of shuffling scheme

Although SS technique provides a low-cost solution for re-randomizing the bit-stream, without of use of PRNG makes its capability on randomness unguaranteed in theory. Therefore, the proper choice of shuffling scheme is of significant importance to the performance of this approach.

In general, two factors determine the re-randomizing capability of the SS technique. The first factor is the number of the split segments, referred as *s*. In most cases the randomness of the new bit-stream increases as *s* increases. This phenomenon can be interpreted in an intuitive way. When *s* becomes large, it means each segment contains just a few bits. In that case, shuffling these segments significantly breaks the correlations among the original consecutive bits. The second factor is the shuffling scheme. For instance, in Fig. 4(a) the use of the "regular" shuffling scheme renders the switching activity increase from 1 to 2, while an "irregular" shuffling scheme in Fig. 4(b) enables the output bit-stream with switching activity as 4. Therefore, the generated stream in Fig. 4(b) has stronger randomness than that in Fig. 4(a).

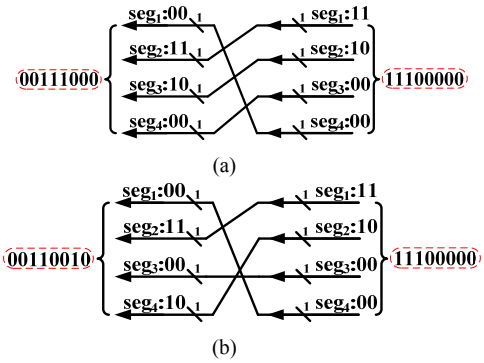

Fig. 4. 4-segment SS with (a) regular shuffling (b) inrregular shuffling.

Due to the importance of shuffling scheme, it is necessary to explore its design methodology. A general principle is to *shuffle the original neighbored segments as far as possible*. This is because such strategy can further reduce the correlation among the successive bits in the original stream. This strategy also explains the phenomenon why Fig. 4(a) and Fig. 4(b) have different randomness. In Fig. 4(a), most segments in the new stream are still neighbored by their original adjacent segments. On the other hand, in Fig. 4(b) the original neighbored segments are now distanced with at least one segment. As a result, the correlation in the original bit-stream is significantly reduced.

3.3.3. PRNG sharing

With the use of the SS technique, the entire data path of the SC system needs to be re-designed. In general, all the basic stochastic arithmetic units are duplicated to *s* copies to be compatible with *s*-segment strategy. Correspondingly, the B2S units at the input end should also be reformulated in this scenario. Fig. 5(a) shows a straightforward design for this reformulation. Here with *s*-segment SS scheme, the original B2S unit that receives one input becomes a stacked B2S unit. As seen in Fig. 5(b), this stacked B2S unit utilizes *s* PRNGs to generate *s* segments of bit-stream. Therefore, for an *N*-input SC system, such as the example *N*-point DCT, the total number of required PRNGs is $N \times s$. Consider PRNG has the dominated area than any other stochastic arithmetic units; the entire *s*-segment SC system incurs significant increase in hardware complexity.

To address this problem, we propose a PRNG sharing scheme and devise an *s*-level B2S unit for *s*-segment

scenario. Fig. 5(c) shows the interplay between the s-level B2S and the stochastic engine. The inner architecture of s-level B2S unit is illustrated in Fig. 5(d). It can be seen that each s-level B2S unit does not contain individual PRNGs but shares s PRNGs that generates s random numbers (RN). Specifically, RN_j, as the random number that is output by $PRNG_j$, is sent to all the s-level B2S units to generate the j-th segment of the corresponding bit-streams, where $1 \le j \le s$. Therefore, the total number of PRNGs is reduced by N times. Notice that because the N s-level B2S units use the same random numbers in each cycle, different shuffle networks are configured in different s-level B2S units to break the correlation. As a result, the segments of bit-stream for different input are still pseudo-independent.

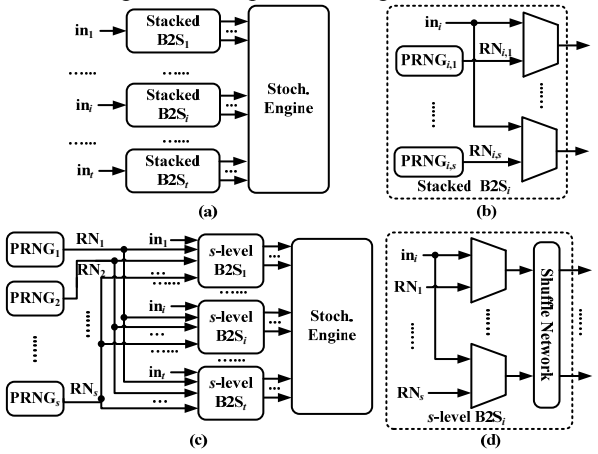

Fig. 5. (a) SC system with stacked B2S. (b) Stacked B2S. (c) SC system with s-level B2S. (d) s-level B2S.

4. RESULTS AND DISCUSSION

4.1. Results

With the use of the set of optimizing approaches in Section 3, the hardware-efficient stochastic DCT can now be developed. Here for the example 4-point DCT, s=2 is set for the SS and PRNG sharing techniques. Fig. 6 shows the root mean square error (RMSE) and signal-noise-ratio (SNR) for different stream-lengths. It can be seen that compared with the conventional stochastic DCT, the proposed optimized design does not have computation accuracy loss.

More importantly, the proposed design shows great advantages with respect to hardware performance. Table 1 summarizes the estimated hardware complexity and latency of the stochastic 4-point DCT designs. It can be seen that the joint use of optimizing approaches lead to 44% reduction and 49% in area and latency, respectively.

4.2. Discussion

4.2.1. Area-Latency tradeoff

Table 1 shows that the proposed design achieves the simultaneous reduction in latency and gate count. At the first glance this phenomenon seems to violate the general principle of VLSI DSP transformation [4]; however the use of PRNG sharing technique can give a proper explanation. From Table 1 it is seen that with nearly 50% reduction in latency (2 segments for each bit-stream), the complexity of stochastic engine is indeed doubled. However, because PRNG has much larger area than any other stochastic arithmetic units, the reduced use of PRNGs (from N to s) brought by PRNG sharing technique naturally enables the propose design achieve significantly lower complexity. Notice that this advantage on low cost will even be enhanced as N increases when entire SC system scales up.

4.2.2. Scope of application of the SS technique

It should be noted that the scope of the proposed SS technique is limited to the combinational-logic-only SC system. In such application, the consecutive bits in each stream is independently generated, hence splitting the stream does not affect the computation accuracy. However, in the sequential-logic-based SC system data dependency between the consecutive bits is required for functional validity, therefore splitting the bit-stream and processing them in parallel will eliminate the data dependency between the last bit of the i-th segment and the first bit of the $(i+1)$-th segment, thereby causing accuracy loss. In general, this accuracy loss will become worse with the increase of s.

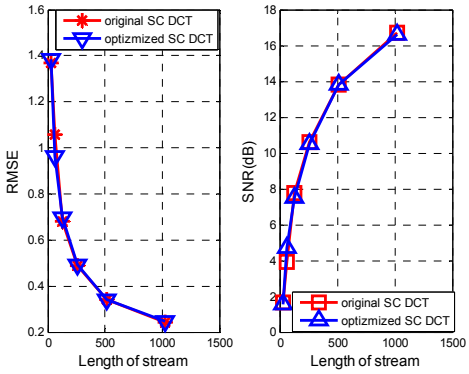

Fig. 6. RMSE and SNR of 4-point SC DCT.

Table 1. Performance of 4-point SC DCT (m=1024)

4-point DCT		Original Design	Optimized Design
# of 1024-length PRNG		4	2
Stochastic Engine	# of XNOR	6	12
	# of 1-bit MUX	8	16
	# of 10-bit-Comp	4	8
# of 16-state FSM		4	0
# of 1-input 10-bit Counter		4	0
# of 2-input 10-bit Counter		0	4
# of 1-bit Register for pipeline		4	8
Total gate counts		4318	2404
Normalized Critical Path		1	1
Latency (cycles)		1025	514

5. CONCLUSION

This paper exploits a set of optimizing techniques for hardware-efficient stochastic computing system. A case study on 4-point DCT is performed. Results show that the proposed approaches enable significant reduction in both latency and complexity.

6. REFERENCES

[1] B. Gaines, “Stochastic computing systems,” Advances in Information Systems Science, vol. 2, no. 2, pp. 37–172, 1969.

[2] B. Brown and H. Card, “Stochastic neural computation I: computational elements,” *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891-905, Sept. 2001.

[3] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete Cosine Transform", *IEEE Trans. on Comput*. vol. C-23, no.1, pp. 90–93, Jan. 1974.

[4] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, New York, NY: John Wiley & Sons Inc., 1999.

[5] A. Alaghi, Cheng Li and John P. Hayes. “Stochastic circuits for real-time image processing applications,” in *Proc. of Design Automation Conference (DAC)*, pp. 1-6, 2013.

[6] A. Alaghi and J. P. Hayes, “Fast and accurate computation using stochastic circuits,” in *Proc. of Design, Automation and Test Conference in Europe (DATE)*, pp. 1-4, 2014.

[7] Peng Li, David J. Lilja, Weikang Qian, and Kia Bazargan, “Computation on stochastic bit streams: Digital image processing case studies,” *IEEE Trans. on Very Large Scale Integrated (VLSI) Systems*, vol. 22, no. 3, pp. 449-462, April 2013.

[8] S. Sharifi Tehrani, W. Gross and S. Mannor, “Stochastic decoding of LDPC codes,” *IEEE Commun. Lett.*, vol. 10, no. 10, pp. 716–718, Oct. 2006.

[9] B. Yuan and K. K. Parhi, “Successive cancellation decoding of polar codes using stochastic computing,” accepted by *Intl. Symp. on Circuit and Systems (ISCAS 2015)*.