A PARALLEL STOCHASTIC COMPUTING SYSTEM WITH IMPROVED ACCURACY

Lifeng Miao and Chaitali Chakrabarti

School of Electrical, Computer and Energy Engineering
Arizona State University, Tempe, USA
{lifeng.miao, chaitali}@asu.edu

ABSTRACT

Stochastic computing (SC) is an attractive computing paradigm because of its fault tolerance and hardware efficiency. Unfortunately, existing SC systems suffer from large latency and inaccuracy problems. In this paper, we propose a new parallel stochastic computing (PSC) system which has higher computing accuracy and faster processing speed compared to existing SC systems. It uses the nibble serial data organization to reduce the latency and a combination of memory based weighted binary generator and data shuffling to improve the accuracy. Simulations on finite impulse response (FIR) filters show that the proposed system with 4 nibbles achieves 35% improvement in accuracy with about 3.5 times increase in processing speed compared to traditional SC. Like SC systems, PSC is also more tolerant of soft errors compared to conventional 2's complement implementations. For a 4-tap FIR filter a 10% injected error causes the root-mean-square-error (RMSE) of PSC to be 0.053 compared to RMSE of 0.188 for 2's complement implementation.

Index Terms— Stochastic computing, Nibble serial, Parallel architecture, FIR filter, FPGA

1. INTRODUCTION

Stochastic Computing (SC) is a new computing paradigm that has high tolerance to noise and is thus ideally suited for noisy circuits in scaled technologies [1]. Here binary numbers are represented by random bit-streams and so a bit flip cannot cause computational havoc. Moreover SC computations can be implemented with very simple circuits and thus have low hardware cost. In recent years, there has been a lot of activity in developing SC systems [1, 2, 3]. The basic elements in stochastic logic have been proposed in [2] along with their finite state machine implementations in [4]. The advantage of implementing image processing algorithms on a SC system has been presented in [5]. SC has also been effectively used in low-density parity-check (LDPC) decoding [6].

While SC has advantages of noise tolerance and hardware efficiency, it has some drawbacks. First, representing a binary number by a random bit stream introduces errors which propagate across the computation. Additional errors are introduced because of correlations in the bit streams. Finally, there is an exponential increase in processing time to achieve higher precision. The inaccuracy introduced during the generation of the random bit stream was addressed in [7, 8] by creating special stochastic number generators (SNG). The method in [9] reduced the correlations among bit-streams by regenerating new and independent stochastic numbers during computing. Recently, an SC inner-product unit based on a novel scaling function was proposed in [10]. This unit was used to implement FIR and IIR filters with low computation errors.

In this paper, we address the accuracy problem of SC as well as its large latency. First, we propose the use of nibble serial organization to divide the long bit streams into several segments and process these segments in parallel. We show that processing nibbles reduce the processing time without affecting the accuracy of computation. Next, we propose a memory based weighted binary generator (MWBG) which reduces the error introduced in the initial conversion. We show that MWBG units can be shared if the nibble serial data are shuffled. More importantly, such a scheme reduces the correlations among bit streams and improves the accuracy. We evaluate the performance of the proposed system by using FIR filter as the case study. The proposed system with 4 nibbles achieves 35% improvement in accuracy with about 3.5 times increase in processing speed compared to traditional SC. It is also highly robust to soft errors; for a 4-tap FIR filter with 10% injected noise, the RMSE of the proposed algorithm is 0.053 compared to RMSE of 0.188 for 2's complement and RMSE of 0.071 for traditional SC.

The rest of the paper is organized as follows. In section 2, we provide the background on SC. In section 3, we introduce nibble serial pattern to address the large latency problem. In section 4, we analyze the sources of inaccuracy in SC and propose methods to reduce them. In section 5, we demonstrate the accuracy and time performance of the proposed parallel SC for digital filters. We conclude the paper in section 6.

2. BACKGROUND

Stochastic computing (SC) transforms computations in deterministic domain into computations in probabilistic domain by

This work was supported in part by NSF CSR 0910699.

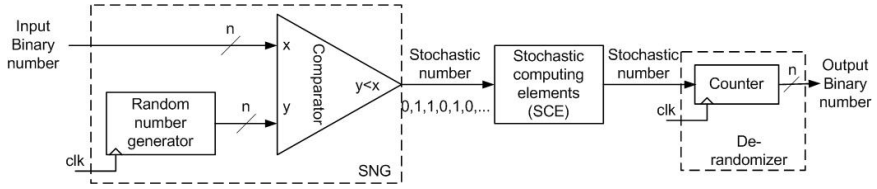

Fig. 1: Overall architecture of a stochastic computing (SC) system.

representing binary numbers as random bit streams [1]. It has high fault tolerance, since a single bit-flip in a long bit stream results in a small change in the value of the corresponding binary number. In contrast, a single bit-flip can cause a huge error in conventional 2's complement computation especially if the bit flip occurs in a high-order bit position. SC systems are typically implemented in bit-serial format, where an n-bit data is represented by a 2^n bit long data stream. Unfortunately, the serial implementation results in an exponential increase in bit stream length which translates to a proportional increase in computation time.

There are two coding formats for SC–unipolar and bipolar [1]. In the unipolar format, a deterministic number $x \in [0, 1]$ is mapped to a bit stream X where $P(X = 1) = x$. For example, the number $x = 0.25$ is mapped into a bit stream with 25% 1s and 75% 0s. In the bipolar format, a deterministic number $x \in [-1, 1]$ is mapped to a bit stream X where $P(X = 1) = (x + 1)/2$.

The overall architecture of an SC system is shown in Fig. 1. It consists of three parts: the stochastic number generator (SNG), the processing elements, and the de-randomizer. The SNG is used to convert the binary number to a random bit stream and is realized by a (pseudo) random number generator and a comparator. The de-randomizer is used to convert the bit stream back to the binary number and it can be implemented by a counter. The stochastic computing elements (SCE) perform computations on the bit streams and include adders, multipliers and comparators. These can be implemented using extremely simple combinational logic as shown in Fig. 2 [2]. Fig. 2(a) shows how a multiplier can be implemented with a single AND gate in Unipolar format and a single XNOR gate in Bipolar format. Adder and subtractor implementations are not as straightforward. In order to keep the output of an adder or subtractor within the same range as the input i.e. between 0 and 1, a scaled adder or subtractor is implemented. Fig. 2(b) shows the gate-level architecture of a scaled adder. The select input S has P(S) = 0.5. Subtractor only needs an extra inverter, as shown in Fig. 2(c), but it can only be implemented in Bipolar format. Fig. 2(d) illustrates the architecture for a comparator, which requires a module for tanh function. The tanh function can be implemented by a finite state machine (FSM) as shown in [11].

The bit serial implementation of an SC system has very low hardware cost. Unfortunately, the computation time for such a system is extremely large. Moreover, SC based systems

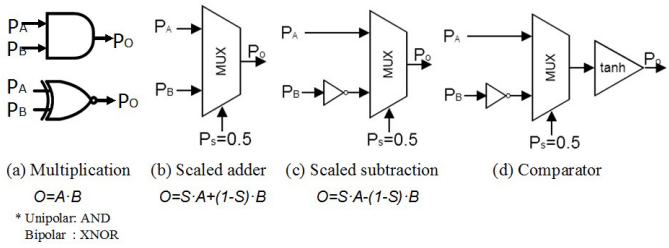

Fig. 2: Basic computational elements in SC.

have additional causes of inaccuracy as will be explained in the following sections.

Table 1: RMSE of initial conversion for SNG and MWBG

	Data range				
	[0,0.2]	[0.2,0.4]	[0.4,0.6]	[0.6,0.8]	[0.8,1]
SNG	$9.1e^{-4}$	0.0014	0.0015	0.0014	$9.1e^{-4}$
MWBG	$5.6e^{-5}$	$5.6e^{-5}$	$5.6e^{-5}$	$5.6e^{-5}$	$5.6e^{-5}$

3. LOW LATENCY WITH NIBBLE SERIAL DATA

Stochastic computing systems can be implemented in bit serial and bit parallel mode [2]. For n-bit data, the bit serial implementation has a latency of at least 2^n. In contrast, the fully parallel implementation has an exponentially large resource utilization. In this section, we propose the use of nibble serial data organization [12, 13] to achieve a balance between fully parallel and bit serial systems. Here the 2^n long bit stream is organized into L nibbles, each of size $2^n/L$. The corresponding architecture is shown in Fig. 3. We use different seeds for different SNGs in Fig. 3 to reduce the correlations among generated nibble segments. The architecture has L times larger hardware but L times lower latency compared to a bit serial implementation. Thus, L can be selected based on the area and timing constraints of the application provided the accuracy is not compromised.

Table 2 shows the accuracy performance in terms of RMSE for basic SC based multiplier and scaled adder for different values of L. 1000 Monte Carlo runs were performed to get the average error. Note that the error is fairly constant across all L and thus increasing L does not affect the RMSE performance. However, for FSM based SCEs such as comparators,

Table 2: RMSE of basic SCE for different number of nibbles when $n = 10$ bits.

L	1	4	8	16	64	256	512
SC Multiplication	0.0083	0.0083	0.0084	0.0085	0.0085	0.0086	0.0088
SC Addition	0.0112	0.0112	0.0113	0.0114	0.0114	0.0115	0.0116

the length of the bit stream in each nibble segment affects the accuracy. In this paper, we focus on digital filters which only includes combinational logic and is thus not affected by the size of the nibble.

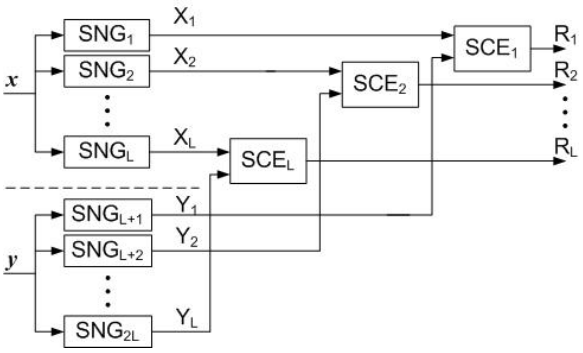

Fig. 3: Architecture of nibble serial implementation for SC.

4. IMPROVING THE ACCURACY OF SC SYSTEMS

4.1. Sources of inaccuracy

There are two main sources of inaccuracies in SC systems: (1) initial conversion error introduced by SNG; and (2) computation error caused by correlations of different bit streams. SNG [2] consists of a random number generator (RNG) and a comparator as shown in Fig 1. In each clock cycle, the RNG generates a random number from a uniform distribution over the interval [0,1] and the comparator produces a 1 if the random number is less than the binary number and a 0 otherwise. The SNG has an inherent inaccuracy due to the fluctuations in random number generation, even when using a high-quality random number source [3]. The inaccuracy introduced by SNG is shown in Table 1 line 1. Here $n = 10$ bits and the uniformly distributed random numbers were generated using Matlab. 1000 Monte Carlo runs were performed to get the RMSE. We see that the average RMSE for SNG over the interval [0,1] is in the order of 10^{-3}, which means that, on average, there is one bit-flip in 1024 bits. In addition, for different ranges, the RMSE values are different; the inaccuracy is highest for numbers close to 0.5.

Now even if the output of SNG is accurate, there will still be an inaccuracy due to correlations among the stochastic numbers being processed. For example, the product of two stochastic numbers $S_1 = (1,1,1,1,0,0,0,0)$ and $S_2 = (0,0,0,0,1,1,1,1)$ is equal to $(0,0,0,0,0,0,0,0) = 0$ which is different from the true result $1/2 \times 1/2 = 1/4$. This is because S_1 and S_2 are correlated. Two n-bit binary sequences $S_1 = (S_1(1), S_1(2), ..., S_1(n))$ and $S_2 = (S_2(1), S_2(2), ..., S_2(n))$ are uncorrelated [7] if and only if

$$\sum_{i=1}^{n} S_1(i)S_2(i) = \frac{\sum_{i=1}^{n} S_1(i) \times \sum_{i=1}^{n} S_2(i)}{n}.$$

Fig. 5 shows the SC multiplication error corresponding to the correlations of two input stochastic numbers. Here the correlations are calculated as

$$\frac{n\sum_{i=1}^{n} S_1(i)S_2(i) - \sum_{i=1}^{n} S_1(i) \times \sum_{i=1}^{n} S_2(i)}{\sigma_1\sigma_2},$$

where $\sigma_1 = \sqrt{n\sum_{i=1}^{n} S_1(i)^2 - \left(\sum_{i=1}^{n} S_1(i)\right)^2}$ and $\sigma_2 = \sqrt{n\sum_{i=1}^{n} S_2(i)^2 - \left(\sum_{i=1}^{n} S_2(i)\right)^2}$. We can see from the figure that the SC error increases linearly with increase in the correlation. Thus it is important that the input bit streams be kept as uncorrelated as possible. Next, several methods are described to reduce the inaccuracy due to correlation.

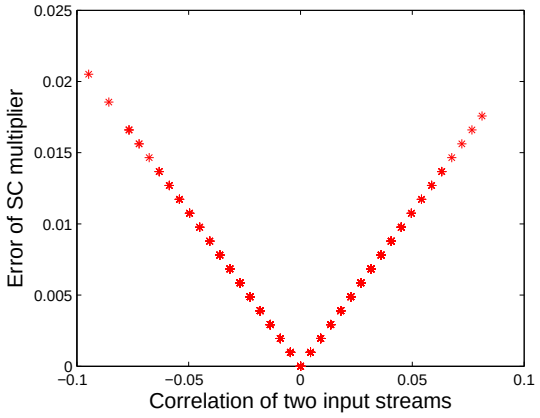

Fig. 5: SC multiplier error as a function of correlation.

4.2. Memory Based Weighted Binary Generator

In order to reduce the inaccuracy introduced by SNG, a new type of SNG called weighted binary generator (WBG) was introduced in [8]. The main idea of WBG is to first generate n stochastic weights $\{w_i\}_{i=0}^{n-1}$ using LFSR, where n is the precision of the binary number and $P(w_i = 1) = 2^{-(n-i)}$. These stochastic weights have non-overlapping 1s and the stochastic number X can be generated by $X = \sum_{i=0}^{n-1} w_i x[i]$, where

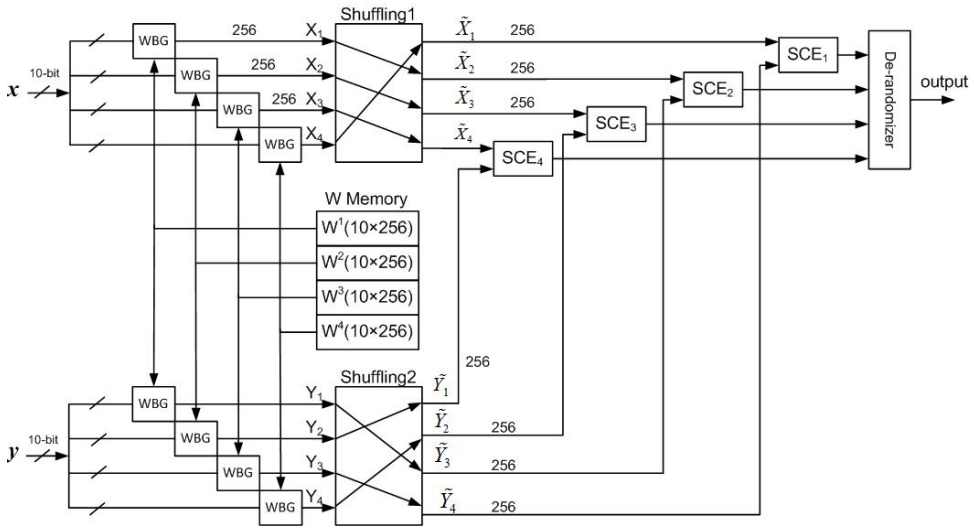

Fig. 4: Architecture for shuffling nibble serial data with MWBG when $n = 10$ and $L = 4$.

$x[i]$ is the ith bit of the binary number. However, the complexity of the WBG circuit increases linearly with bit length n. For example, to convert a 10-bit binary number, we need a 10th-order linear feedback shift register (LFSR) and several gates with very high fan in. In order to reduce the hardware complexity of WBG, we propose a memory based WBG (MWBG). Since the stochastic weights $\{w_i\}_{i=0}^{n-1}$ are constant, we can pre-calculate the weights and store them in a $n \times 2^n$ bits memory, where the ith row of the memory stores w_i. During serial processing, the weights are read from the memory 2^n times and each time they are multiplied with x and ORed to generate the stochastic output bit. The memory based WBG architecture is shown in Fig. 6. The conversion error of proposed MWBG method is compared with traditional SNG method in Table 1. For 1000 Monte Carlo runs, the RMSE of MWBG over the interval [0,1] is $5.6 \times e^{-5}$ which is about 20 times smaller than the error of SNG.

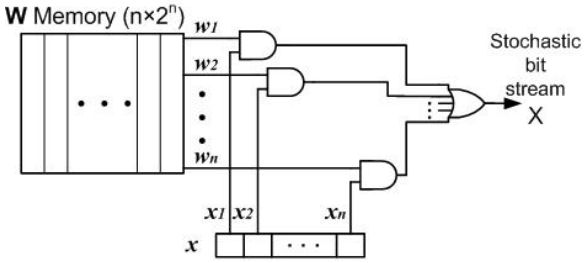

Fig. 6: Memory based weighted binary generator.

4.3. Shuffling Nibble Serial Data

The second source of inaccuracy is due to correlations among different bit streams being processed. The straight forward method to reduce these correlations is to use different random seeds for different SNGs or use uncorrelated weights for different MWBGs. In the latter case, the required memory will increase linearly with the number of inputs. Here we propose to use the same weights memory for different inputs but shuffle the nibble serial data to reduce the correlations.

The architecture for shuffling nibble serial data with a single MWBG is shown in Fig. 4. For n-bit precision, the original $n \times 2^n$ W memory is divided into L sub-memories $W^\ell, \ell = 1, ..., L$, each of size $n \times 2^n/L$, where L is the number of nibbles. Each input is multiplied with $W^\ell, \ell = 1, ..., L$ concurrently to generate L parallel data streams which are then shuffled. Each input has a unique shuffling pattern as shown in Fig 4. In order to keep the shuffling patterns independent for different inputs, one W memory can be shared by L inputs. For example, in Fig. 4, $L = 4$ and so one weight memory is shared by 4 inputs.

Data shuffling is essential to reduce the correlation among the data streams. To illustrate its importance, consider SC multiplication with $n = 10$ and $L = 4$. The RMSE with data shuffling is 0.0037, which is significantly less than the RMSE without data shuffling which is 0.0386. Thus, the proposed shuffling scheme for nibble serial data reduces the memory footprint while keeping the correlations among bit streams low, thereby improving the accuracy.

The overall architecture for the proposed parallel SC is shown in Fig. 4. The 2^n length bit stream is divided into L nibbles (here $n = 10$ and $L = 4$), and all the nibbles are processed simultaneously to reduce the processing time. The MWBG and shuffling modules reduce the inaccuracy problem caused by initial conversion error and correlations among bit streams, resulting in improved performance as will be shown in the next section.

5. SYSTEM EVALUATION RESULTS

In this section, we evaluate the performance of the proposed parallel stochastic computing (PSC) system in Fig. 4 for different FIR filter implementations. We compare the proposed PSC with traditional serial SC [2] in terms of accuracy, pro-

cessing time and resource utilization. We also show the high fault-tolerance of PSC compared to conventional 2's complement implementation.

5.1. Simulation setup

We implement FIR filters using the proposed system and demonstrate its performance. The block diagram of a 4-tap FIR filter implementation is shown in Fig. 7. Here $Y[n] = B_0X[n] + B_1X[n-1] + B_2X[n-2] + B_3X[n-3]$. A tree structure was used to minimize the computation depth. For the simulations, the input signal consists of a mixture of four sinusoidal waves of different frequencies and the 4-tap low pass FIR filter has a cut-off frequency of 0.4π. The length of the stochastic bit stream is 1024, which corresponds to 10-bit precision.

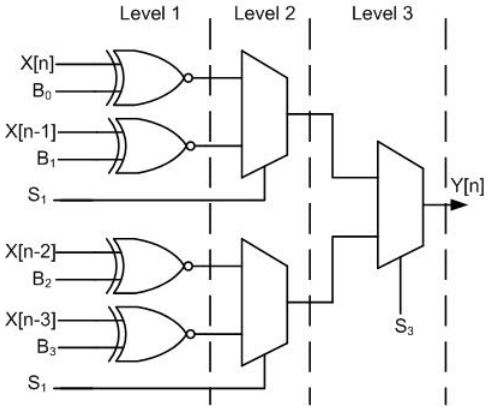

Fig. 7: Block diagram of a 4-tap FIR filter.

5.2. Performance as a function of number of nibbles

To evaluate the area, timing and RMSE performance of nibble serial data organization, a 4-tap low pass FIR filter based on the PSC was implemented on a Xilinx Virtex-5 FPGA platform. Since the 4 tap FIR filter has 8 inputs (see Fig. 7), the architecture has $8/L$ WMBGs, where L is the number of nibbles. Table 3 illustrates the resource utilization, processing cycles and RMSE performance for different values of L for the 4-tap filter. We can see that with the increase in L, the number of slices increases almost linearly. However, the number of BlockRAM units decreases linearly with L since each WMBG unit can be shared by L inputs. Here one WMBG is implemented by two BlockRAMs and the size of each BlockRAM is 36 Kbits. The number of processing cycles decreases almost linearly with L, as expected. In terms of the RMSE performance, there is only a slight increase in RMSE (from .0061 for L=1 to .0097 for L=8). Thus we conclude that for FIR filters implementations L does not adversely effect the performance. L can be chosen based on the area or timing constraint of the application. For high-speed systems, large L is a good choice; if area is a constraint, small L is preferred.

Table 3: Area, speed and accuracy comparison for the 4-tap FIR filter.

Number of nibbles L	1	2	4	8
Slices	32	66	158	323
BlockRAMs	16	8	4	2
Cycles	1049	553	304	187
RMSE	0.0061	0.0078	0.0086	0.0097

5.3. Accuracy performance of MWBG

The memory based weighted binary generator (MWBG) was proposed in Section 4.2 to improve the accuracy of SC. Table 4 compares the accuracy performance of SNG based system [2] with the MWBG based system in terms of RMSE. Here the floating point FIR filter output obtained using Matlab was used as the ground truth. We see that in each level of the FIR filter, RMSE of the MWBG system is much smaller than the SNG system. This is because the MWBG generates stochastic bit streams more accurately. In both systems, the RMSE increases as the number of levels increases.

Table 4: RMSE of SC with SNG and MWBG for low pass 4-tap FIR filter with $L = 1$.

Filter step	Level 1	Level 2	Level 3
SC with SNG	0.0090	0.0105	0.0117
SC with MWBG	0.0036	0.0049	0.0061

5.4. Comparison of PSC with traditional SC

Next, we present the performance of the PSC system in terms of RMSE and latency. We implement 2-tap, 4-tap and 8-tap FIR filters using two methods: serial SC with SNG [2] and proposed PSC. To obtain the RMSE, 1000 Monte Carlo simulations were performed and the errors with respect to floating point results were computed. From Table 5, we can see that compared to traditional SC [2], the PSC with 4 nibbles improved the computing accuracy by about 35% and reduced the processing time by about 3.5 times. Thus, using MWBG and shuffling nibble serial data, the proposed PSC effectively improved the accuracy and reduced the processing time compared to traditional SC. We also implemented the FIR filter with transposed structure for comparison. The RMSE the transposed structure is 0.0074 compared to 0.0061 for the tree structure.

Table 5: Accuracy and processing time comparisons for low pass FIR filters with L=4.

Filter tap	SC in [2]		PSC	
	RMSE	Cycles	RMSE	Cycles
2	0.0096	1041	0.0062	302
4	0.0129	1043	0.0086	304
8	0.0237	1046	0.0147	307

5.5. Fault tolerance analysis

High fault-tolerance is one of the main advantages of SC. We test the fault-tolerance of the proposed PSC by randomly injecting soft errors into the processing units, and measuring the corresponding RMSE at the output. Fig. 8 illustrates the RMSE obtained for a 4-tap FIR filter using conventional 2's complement processing, serial SC [2] and PSC with 4 nibbles for different error rates. Here for 2's complement implementation, we used 10-bit precision and for both SC and PSC, the length of the stochastic bit stream was 1024. 1000 Monte Carlo runs were performed to obtain the RMSE. We can see that the RMSE of PSC is the smallest, e.g., for 10% injecting error rate, the RMSE of PSC is 0.053, while the RMSEs of 2's complement implementation and traditional SC are 0.188 and 0.071, respectively. As the soft error rate increases, the RMSE of conventional computing increases significantly. However, the PSC can still produce fairly good computing results even at error rates as high as 20%.

Additionally, PSC takes fewer hardware resources than 2's complement computing. To prove this point, we synthesized the 4-tap FIR filter using Synopsys DC-compiler with 90nm technology. The area utilization for 10-bit 2's complement implementation was 18,330 units. In contrast, the area utilization for proposed PSC was only 2,241 units. If area-latency product is considered as the hardware performance metric, then the area-latency product of the proposed PSC system is 3.2 times larger than the 2's complement implementation. We believe that this penalty is not as severe considering the high error tolerance of such systems.

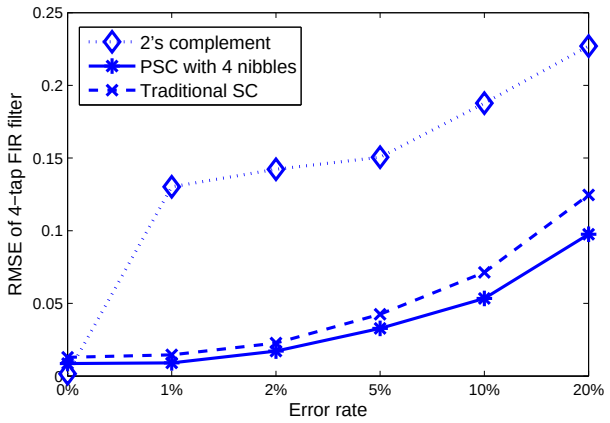

Fig. 8: Fault-tolerance comparison of 2's complement computing, traditional SC [2] and PSC for 4-tap FIR filter.

6. CONCLUSION

In this paper, we presented a new parallel stochastic computing system which has better accuracy performance with less processing time comparing to traditional SC. We proposed the use of nibble serial data organization to speed up the processing of long bit-streams. To improve the accuracy, we integrated the memory based weighted binary generator with data shuffle and showed that this reduces both the initial conversion error and the computing error due to correlations. We demonstrated the performance of our system using low pass FIR filters as the case study. We showed that the proposed PSC with 4 nibbles improved the computing accuracy by 35% and improved the processing time by about 3.5 times compared to the traditional SC. Furthermore, we demonstrated the high fault-tolerance of PSC by comparing it with conventional 2's complement implementation.

7. REFERENCES

[1] B.R. Gaines, "Stochastic computing systems," *Advances in Information Systems Science*, vol. 2, pp. 37–172, 1969.

[2] B.D. Brown and H.C. Card, "Stochastic neural computation. I. computational elements," *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 891–905, 2001.

[3] A. Alaghi and J.P. Hayes, "Survey of stochastic computing," *ACM Trans. Embedded Computing Systems*, 2012.

[4] P. Li, W. Qian, M.D. Riedel, K. Bazargan, and D.J. Lilja, "The synthesis of linear finite state machine-based stochastic computational elements," in *17th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2012, pp. 757–762.

[5] P. Li and D.J. Lilja, "Using stochastic computing to implement digital image processing algorithms," in *IEEE 29th International Conference on Computer Design (ICCD)*, 2011, pp. 154–161.

[6] A Naderi, S. Mannor, M. Sawan, and W.J. Gross, "Delayed stochastic decoding of LDPC codes," *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5617–5626, 2011.

[7] P. Jeavons, D.A. Cohen, and J. Shawe-Taylor, "Generating binary sequences for stochastic computing," *IEEE Transactions on Information Theory*, vol. 40, no. 3, pp. 716–720, 1994.

[8] P.K. Gupta and R. Kumaresan, "Binary multiplication with pn sequences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 4, pp. 603–606, 1988.

[9] S.S. Tehrani, A. Naderi, G.A. Kamendje, S. Hemati, S. Mannor, and W.J. Gross, "Majority-based tracking forecast memories for stochastic ldpc decoding," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4883–4896, 2010.

[10] Y.N. Chang and K. K. Parhi, "Architectures for digital filters using stochastic computing," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013.

[11] W. Qian and M.D. Riedel, "The synthesis of robust polynomial arithmetic with stochastic logic," in *45th ACM/IEEE Design Automation Conference (DAC)*, 2008, pp. 648–653.

[12] R.A. Cottrell, "Nibble-serial: an architecture for VLSI digital signal processors," in *IEEE International Symposium on Circuits and Systems*, 1988, pp. 1779–1782.

[13] K. K. Parhi, *VLSI digital signal processing systems: design and implementation*, John Wiley and Sons, 1999.