Stochastic Computing-based Baseband Processing for Resource Constraint IoT Devices

[1]Kazi J. Ahmed, [2]Yang G. Kim, [3]Bo Yuan,[4]Myung J. Lee, and [5]Kazuya Tsukamoto
[1]kahmed06@citymail.cuny.edu, [2]ifxcom@hotmail.com, [3]bo.yuan@soe.rutgers.edu,
[4]mlee@ccny.cuny.edu, [5]tsukamoto@cse.kyutech.ac.jp

***Abstract*—With the recent deployment of 5G network, the ever increasing IoT has got a tremendous boost in its expansion and already has penetrated well into the government, commercial and private sectors. With the countless IoT devices and myriad of applications, many of them are resource constrained and have limited energy budget. These IoT devices demand low-energy technique for their computing and communication tasks to stay active for longer period. The two main baseband processes that dissipate bulk of CPU power from the IoT device are synchronization and Finite Impulse Response (FIR) filtering. In this circumstance, hardware-based baseband processing can take these tasks off of the CPU and may significantly reduce energy consumption. While conventional Binary Radix Computing (BC)-based hardware modules can improve power dissipation, Stochastic Computing (SC)-based hardware will certainly cut down much more both the power as well as silicon space in comparison. With this motivation, we propose novel SC-based hardware designs in regards to synchronization and Finite Impulse Response (FIR) filter for resource constraint IoT devices. Comparative analysis shows that our proposed SC-based design can reduce significantly more power and silicon area compared to the BC as well as other proposed SC designs.**

***Index Terms*—Binary Radix Computing (BC), Stochastic Computing (SC), 5G, resource constraint IoT devices, low power baseband processes.**

I. Introduction

TheInternet of Things (IoT) is one of the major emerging technologies accelerated by the 5G network. Within the plethora of IoT devices, many of them are resource constrained and battery operated [21]. These battery-operated IoT devices require to save energy in order to stay active for a long period of operation. However, these devices cannot eschew the computationally intensive baseband processing tasks entailed by radio transceiver. The crux of these tasks are the synchronization and Finite Impulse Response (FIR) filtering [1]. One way to reduce the burden of these computationally intensive tasks is to use a modified SIMD processor-based design as proposed by the authors in [1]However, any processor-based design shall draw more power compared to its hardware-based design counterpart. As a result, hardware based design is always a better option when considering tight power budget, especially for resource constraint IoT devices. Yet, conventional two's complementary-based Binary Radix Computing (BC) hardware-design may require excessive silicon area and hence higher power consumption that may discourage to employ it altogether. The best option among all would be to design using Stochastic Computing (SC)-based hardware accelerator that will substantially reduce energy consumption as well as silicon area.

The advantage of SC-based hardware design is that it occupies considerably less silicon area and thus, is deemed a potential solution for resource constraint IoT devices. However, since SC is generated probabilistically from random number generator (RNG) and it is hard to design absolute-independent RNGs, SC-based design innately incurs error. The errors will get exacerbated when a large number of series SC processing blocks are in use, especially in FIR filtering. Lowering the number of series blocks by employing 2-by-2 fast FIR Algorithm (FFA), our initial work on SC based low order (12) FIR filter design shown to be better in terms of silicon area and performance [2-4]. In this paper, we propose higher order (36) FIR filters appropriate for IoT devices by employing 3-by-3 or 6-by-6 FFA.

While many SC-based FIR filter designs have been proposed, SC-based baseband Synchronization design is yet to be seen in the literature. The basic process of getting the synchronization is to find the correlation of the known and the incoming received signal. The proposed SC-based design exploits the fact that the correlation is simply the inner-product of two signals. Making use of the SC-based inner-product module of our previous work [2], the proposed SC-based Synchronization shows good accuracy. To further evaluate the effectiveness, the proposed SC-based Synchronization design is implemented in C language and the outputs are compared with those of the software-based one. Both numerical and simulation results presented later show that the proposed approach can significantly increase the accuracy of Synchronization as well as higher-order SC FIR filter, thereby paving the way of promoting stochastic DSP modules in practical applications.

II. Background on SC, FIR Filter and Synchronization

In this section, we provide brief backgrounds on SC, FIR filter and Synchronization before putting forth our proposed work.

A. Stochastic Computing (SC)

Derived probabilistically from the BC, SC is represented by a bit stream of '1's and '0's. The BC to SC conversion process takes a number generated from a Random Number Generator (RNG) and compared it with the given BC value by a comparator, resulting in 1 or 0. This task is conducted iteratively to create a *N*-bit stream of 1's and 0's that reflects the BC equivalent of SC value. Since BC is converted to SC

probabilistically, before converting, the BC value has to be made within [0,1] for unipolar and [-1, 1] for bipolar. Any unipolar value $0 \leq x \leq 1$ in BC is converted to a length of *N*-bit stream containing *X* bits of '1's so that x = P(X) = X/N(Fig.1-*a*) [5-7]. In case of bipolar, any value $-1 \leq x \leq 1$ in BC is converted to a bit stream of *N*-bit, out of which *X* bits are '1's such that x = 2(X/N)-1 (Fig.1-*c*) [5-7]. Because of this conversion process, if two SC values P(A) and P(B) are independent, their product P(A)P(B) will be equal to P(C = AxB). In other words, if two SC values are generated using independent RNGs, their product can be computed by an AND gate in SC domain (Fig.1-*b*). Likewise, it can be shown that a simple multiplexer (MUX) can be used to perform SC Addition. Fig. 1 shows BC to SC (B2C) conversions and some basic operational modules. Moreover, SC is not limited to just Multiplication and Addition, for some other arithmetic operations, readers are referred to [5-9].

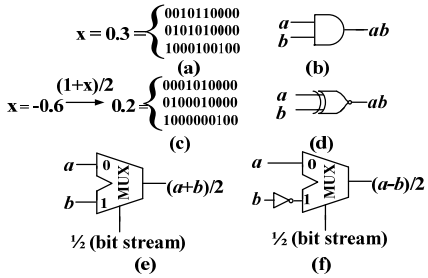

Fig. 1. Some basic SC presentations and operations:
(a) Unipolar B2C conversion, (c) Bipolar B2C conversion (b) Unipolar multiplication, (d) Bipolar multiplication, (c) SC adder, (f) SC subtractor

B. Finite Impulse Response (FIR) filter

FIR filter is one of the major DSP baseband modules used for wireless communication system including 5G technology. The transmitter pulse shaping filter and receiver matched filter used in CDMA systems are two examples of FIR filters [10]. Filter Banks Multicarrier (FBMC) and Universal Filtered Multicarrier (UFMC) systems used in 5G technology, are also FIR filters [11-12].

The output sequence of an FIR filter of order m can be expressed in time domain as

$$y(n) == \sum_{i=0}^{m-1} x(n-i)h_i \quad (1)$$

where $x(n)$ and $y(n)$ are the input and output of the filter respectively at discrete time interval, and h_i are the filter coefficients. To make a hardware module for this FIR filter, m high-complex multipliers are required in BC-based design while the SC-based design entails only m AND gates. This certainly makes SC-based design a better option for IoT device. However, SC-based FIR filter is prone to error. This error accumulates more and more as the number of series blocks in FIR filter increases due to higher number of taps to achieve higher order filter.

C. Synchronization

The main function of the baseband synchronization is to find and locate the beginning of a packet in an incoming message. This is achieved by sending a reference signal attached in the header of each transmitted packet and correlates the incoming reference signal with the receiver's known signal. The outcome of this correlation indicates the location of the start of a packet.

In this proposed work, we demonstrate our work for synchronization of IEEE 802.11 protocol which is achieved by sending pre-defined preamble as the reference signal. However, our design can be applied to any other protocol with little to no modification. The structure of the preamble of this protocol is shown in Fig. 2 [13-14]. Each preamble consists of 10 Short Training Sequences (STSs) and 2 Long Training Sequence (LTSs)

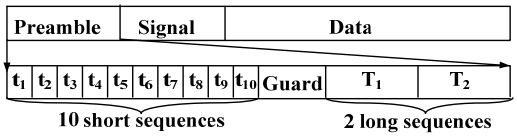

Fig. 2. Preamble structure of IEEE 802.11 (from [13])

IEEE 802.11 uses Orthogonal Frequency Division Multiplexing (OFDM) for high speed and high spectral communications. The expression of the OFDM symbol is presented as follows:

$$S_m(n) = \sum_{k=0}^{N-1} d_{m,k} e^{\frac{j2\pi kn}{N}} \quad (2)$$

where $S_m(n)$is the n-th sample of the m-th symbol, $d_{m,k}$ is the Quadrature Amplitude Modulation (QAM) or Phase Shift Keying (PSK) modulated complex data vector and N is the number of subcarrier[13,15]. Each STS OFDM symbol m has 16 (n = 1, 2, ... 16) samples and consists of 52 sub-carriers ($e^{\frac{j2\pi kn}{N}}$, $N = 52, k = 0, 1, \ldots 51$) (IEEE 802.11n/ac) modulated by the QAM/PSK complex vector$d_{m,k=0-51}$={0,0, -1-1j, 0, 0, 0, -1-1j, 0, 0, 0, 1+1j, 0, 0, 0, 1+1j, 0, 0, 0, 1+1j, 0, 0, 0, 1+1j, 0, 0, 0, 0,0, 0, 0, 1+1j, 0, 0, 0, -1-1j, 0, 0, 0, 1+1j, 0, 0, 0, -1-1j, 0, 0, 0, -1-1j, 0, 0, 0, 1+1j, 0, 0}. The time domain waveform of the 10 symbols are shown in Fig. 3.

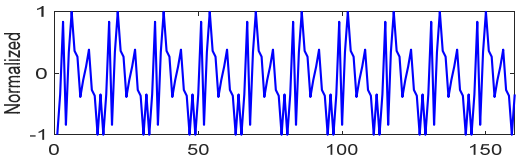

Fig. 3. STS signal in IEEE 802.11 protocol

If the received reference signal at time t is $r(t)$ for a known signal p, the correlation $P(t)$ can be calculated using the following equation [1,13]

$$P(t) = \sum_{m=0}^{L-1} r(t+m)p^*(m) \quad (3)$$

where p^* is the complex conjugate of p and L = 16. The synchronization is achieved by finding the maximum value of the correlation $P(t)$. The start of the incoming packet at receiver can be detected by comparing the magnitude of correlation result with a certain threshold. It is advised to have a dynamic threshold based on incoming signal power.

III. Related Work

The common mathematical block used for both FIR filter and synchronization is the inner-product. In other words, basic hardware designs of FIR filter and synchronization require Multiplication and Adder blocks. The SC-based multiplication can easily be implemented using AND gate. However, basic SC-based Adder block incurs scale down effect and hence becomes the major source of SC error [5-7]. As a result, the main focus of the previous works regarding SC FIR filter design was to improve SC-based Adder block.

Since the basic SC-based Adder (SA) produces scaled down effect, the high order FIR filter based on SA suffers huge accuracy loss. To reduce the scaled down effect of SA, a novel inner-product-based FIR design was proposed in [6]. A 2-input inner-product from [6] is shown in Fig. 4. Here, *x*'s are filter input, *a*'s are filter coefficients and *s*(*a*)'s are the sign of *a*'s.

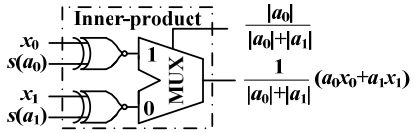

Fig. 4. Inner-product module proposed in [6]

Even though the output of this unit still has the scaled down effect by the term ($|a_0|+|a_1|$), the improved accuracy comes from the assumption that the denominator of the output ($|a_0|+|a_1|$) is less than or close to 1. Moreover, this design also assumes that the values of *h* parameters are pre-known. As a result, the MUX selection input is pre-calculated and converted to SC value before it is placed. However, for practical applications (dynamic environment), the values of *h* parameters are required to be adjusted time to time.

The SC-based design efforts continued and the authors from [7] proposed another new design to mitigate the scaled down effect of SA. In this design, the SC value of *x* is presented using two lines of *N*-bit streams. Separating the magnitude M(X_i) and sign S(X_i), any SC value is represented as $x = \frac{1}{N}\sum_{i=0}^{N-1}[1 - 2S(X_i)M(X_i)]$. In this design,exclusive OR (XOR) and AND gates are used for two-lines SC Multiplier (TSM) and a complex logic circuit is used for two-line SC Adder (TSA) as shown in Fig 5.

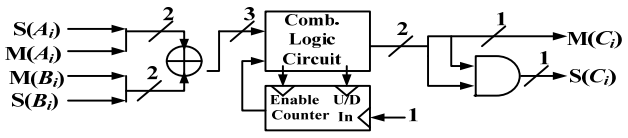

Fig. 5. Adder circuit for TSA from [16]

The accuracy of the TSA is claimed to be independent of the number of stages (taps); however, each stage requires significant amount of SC processing. Moreover, the area and power consumption for each stage are also high. Consequently, it limits the number of stages of FIR filter and makes it inefficient for high order filter.

Yet another stochastic FIR filter was proposed in [17] focusing on the reduction of the scaled down effect of SA by making use of Accumulative Parallel Counter (APC). However, in this design each of these APC adders requires numerous Half Adders (HAs) and Full Adders (FAs) circuits. Consequently, this design requires much more complicated hardware than the others.

The above discussion of the state-of-the-art works shows that the practical FIR filter implementation (high order) using SC-based hardware is still very challenging. Moreover, the prior efforts focused only on the higher accuracy of SC unit, not considering the reduction of the overall error. As for the Synchronization, to the best of our knowledge, this is the first attempt for any SC-based Synchronization design.

IV. Proposed SC-based designs

In this work, we are proposing SC-based designs for both FIR filter and Synchronization. Since both modules require inner-product operation, their basic SC hardware structures are the same. However, proposed SC-based high order FIR filter design breaks the long inner-product chain into short parallel sub-chains by employing FFA. In the following subsections, we introduce FFA with multiple dimensions that are exploited in the proposed FIR filter design.

A. 2-by-2 Fast FIR Algorithm (FFA2)

The major limitation of high order SC-based FIR filter designs is that it entails significant error due to a large number of serial inner-product blocks. To address this inherent difficulty, we introduce FFA in our SC-based FIR filter design. In general, a *q*-by-*q* FFA FIR filter design can virtually reduce an *m* order filter to *m/q* order parallel sub-filters. In this subsection, we first provide the architecture of 2-by-2 SC FFA FIR filter (FFA2). Later, the architecture of other dimensional FFAs will be explained.

The relationship of input *x*(*n*) and output *y*(*n*) of a FIR filter of order *m* in *z*-domain can be expressed as

$$\sum_{n=0}^{\infty} y(n)z^{-n} = Y(z) = H(z)\cdot X(z) = \sum_{n=0}^{m-1} h(n)z^{-n}\cdot\sum_{n=0}^{\infty} x(n)z^{-n}, \qquad (4)$$

where *H*(*z*) is the *z*-transform of the filter coefficient h_i. The *z*-transform of the input *x*(*n*) can be expressed as

$$\begin{aligned} X(z) &= x(0) + x(1)z^{-1} + x(2)z^{-2} + x(3)z^{-3} + \cdots \\ &= [x(0) + x(2)z^{-2} + \cdots] + z^{-1}[x(1) + x(3)z^{-2} + \cdots] \\ &= X_E(z^2) + z^{-1}X_O(z^2), \end{aligned}$$

where $X_E(z^2)$ is the z-transform of the even time series {*x*(2*k*)} and $X_O(z^2)$ is the z-transform of the odd time series {*x*(2*k*+1)} of the input *x*(*n*). Similarly, the z-transform of filter coefficients and the output can be expressed as

$$\begin{aligned} H(z) &= H_E(z^2) + z^{-1}H_O(z^2) \\ Y(z) &= Y_E(z^2) + z^{-1}Y_O(z^2) \qquad (5) \end{aligned}$$

Equating (4) and (5) will provide $Y_E(z^2)$ and $Y_O(z^2)$ in terms of $X_E(z^2)$, $X_O(z^2)$, $H_E(z^2)$ and $H_O(z^2)$ as

$$\begin{aligned} Y_E(z^2) &= X_O(z^2)H_O(z^2) + (z^2)^{-1}[X_E(z^2)H_E(z^2)] \\ Y_O(z^2) &= X_E(z^2)H_O(z^2) + [X_O(z^2)H_E(z^2)]. \end{aligned}$$

Consequently, the output $Y(z)$ can be produced from $X_E(z^2)$, $X_O(z^2)$, $H_E(z^2)$ and $H_O(z^2)$. The schematic diagram for this filter, termed FFA2 is shown in Fig. 6. As seen from Fig. 6, the even $X_E(z^2)$ and odd $X_O(z^2)$ time-series inputs are needed at the same time to generate the even $Y_E(z^2)$ and the odd $Y_O(z^2)$ output series and hence the final output $Y(z)$. Thus, output of m order FIR filter can be generated by $m/2$ order sub-filters. However, the even or odd input series need to be directed to proper h coefficient block to yield the right output. The z-transform of the even and the odd h coefficients of each $m/2$ sub-filters can be expressed as

$$H_E(z^2) = h_0 + h_2 z^{-2} + h_4 z^{-4} + h_6 z^{-6} + \cdots$$
$$H_O(z^2) = h_1 + h_3 z^{-2} + h_5 z^{-4} + h_7 z^{-6} + \cdots$$

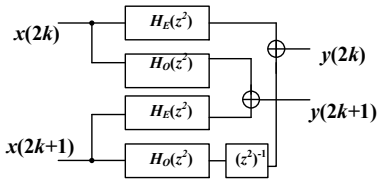

Fig. 6. Schematic diagram of 2-by-2 FFA FIR filter

The outputs of the four H blocks in time domain can be expressed as:

$$\begin{aligned} X_E H_E &= x(0)h_0 + x(2)h_2 + x(4)h_4 + \cdots \\ X_E H_O &= x(0)h_1 + x(2)h_3 + x(4)h_5 + \cdots \\ X_O H_E &= x(1)h_0 + x(3)h_2 + x(5)h_4 + \cdots \\ X_O H_O &= x(1)h_1 + x(3)h_3 + x(5)h_5 + \cdots \end{aligned} \quad (6)$$

The above expressions clearly state that the output of each block is the inner-product of some input x and coefficient h.

B. 3-by-3 FFA FIR Filter Design (FFA3)

Similar to FFA2, the input signal stream $X(z)$in FFA3 will be divided into three groups: 1) $X_0(z^3)$, the z-transform of the first group of time series $\{x(3k)\}$, 2) $X_1(z^3)$, the second group of time series $\{x(3k+1)\}$, and 3) $X_2(z^3)$, the third group of time series $\{x(3k+2)\}$. With these, the z-transform of filter coefficients can be expressed as

$$H(z) = H_0(z^3) + z^{-1}H_1(z^3) + z^{-2}H_2(z^3)$$

As a result, an m order FIR filter would be composed of nine FFA3 sub-filters of order $m/3$. In other words, the number of series inner-product blocks reduced by three. The numerical analysis presented later demonstrates that this reduction significantly lower the error of SC-based FIR filter.

C. Higher Dimensional FFA FIR Filter Design

The number of series inner-product blocks can be reduced further by employing FFA2s and/or FFA3s in layers. For example, an m order FIR filter can be reduced to some $m/4$ order parallel FIR sub-filter by applying layer ofFFA2s inside FFA2s. Same way, an m order FIR filter can be reduced to some $m/6$ order parallel FIR sub-filter by applying layer of FFA2s inside FFA3.Thus, an m order FIR filter can be reduced to any m/q sub-filters applying appropriate layers of FFAs. The general structure of high dimensional FFA (FFAq) is shown in Fig. 7

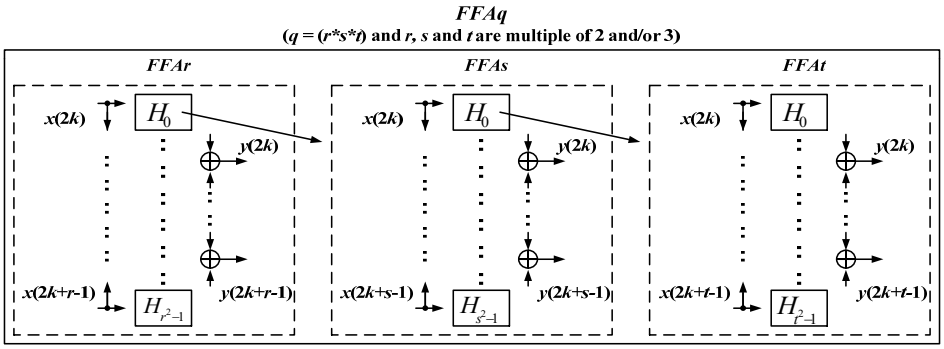

Fig. 7 General structure of SC FFAq

D. Propose SC-based FIR and Synchronization Designs

Our proposed designs are built from 2-input SC inner product. A 2-input SC inner product is designed using JK flip-flops and exclusive NOR (XNOR). JK flip-flop can be used as a 2-input approximate divider in SC domain where inputs are a_0 and a_1 and output is $\frac{|a_0|}{|a_0|+|a_1|}$[7]. Whereas an XNOR gate is used as a 2-input bipolar multiplier [5-9]. Thus the proposed 2-input SC inner product output as shown in Fig. 8 can be derived as

$$x_0 s(a_0) \times \frac{|a_0|}{|a_0|+|a_1|} + x_1 s(a_1) \times \left(1 - \frac{|a_0|}{|a_0|+|a_1|}\right)$$
$$= \frac{x_0 a_0}{|a_0|+|a_1|} + \frac{x_1 a_1}{|a_0|+|a_1|} = \frac{x_0 a_0 + x_1 a_1}{|a_0|+|a_1|}.$$

This 2-input inner SC product can be extended to 4-input by a 4-input approximate divider and an additional MUX. A 4-input approximate divider on the other hand can be produced from its 2-input counterpart. Noted that an AND gate is used to make 2-input unipolar SC multiplier, thus the output of a 4-input approximate divider shown in Fig. 10 can be derived as

$$\frac{\left(|a_0| \times \frac{|a_2|}{|a_2|+|a_3|}\right)}{\left(|a_0| \times \frac{|a_2|}{|a_2|+|a_3|}\right) + \left(|a_2| \times \frac{|a_0|}{|a_0|+|a_1|}\right)} = \frac{\left(\frac{1}{|a_2|+|a_3|}\right)}{\left(\frac{1}{|a_2|+|a_3|}\right) + \left(\frac{1}{|a_0|+|a_1|}\right)} = \frac{|a_0|+|a_1|}{|a_0|+|a_1|+|a_2|+|a_3|}.$$

The 4-input inner product as shown in Fig. 9 can be derived as

$$\frac{x_0 a_0 + x_1 a_1}{|a_0|+|a_1|} \times \frac{|a_0|+|a_1|}{|a_0|+|a_1|+|a_2|+|a_3|} + \frac{x_2 a_2 + x_3 a_3}{|a_2|+|a_3|} \times \left(1 - \frac{|a_0|+|a_1|}{|a_0|+|a_1|+|a_2|+|a_3|}\right)$$
$$= \frac{x_0 a_0 + x_1 a_1 + x_2 a_2 + x_3 a_3}{|a_0|+|a_1|+|a_2|+|a_3|}.$$

Similarly, the a 4-input inner product module can be extended to any number of input inner product as required by the FIR or Synchronization block.

The proposed inner-product design is better than the one proposed in [6]. In [6], considering fixed values of filter

coefficients *a*'s, SC model requires to pre-calculate($|a_0|$ /[$|a_0|+|a_1|$]) before converted to SC. We refer it as [6] Fixed in this paper. On the other hand, considering dynamic environment, where the values of *a*'s will be changed over time, an additional Divider module is required to calculate ($|a_0|$ /[$|a_0|+|a_1|$]) from *a*'s in BC before converting to SC. This is referred to as [6] Dynamic. Thus our proposed design has two major advantages over [6]: 1) it does not require additional BC to SC conversion of $|a_0|$ /($|a_0|+|a_1|$), as a result, does not necessitate extra RNGs and Comparators, 2) it does not require hardware Divider module along with Adder in case of changing (dynamic) values of *a*'s. Especially for FIR filter, coefficient values (*a*'s) need to be changed time to time to accommodate with the varying channel condition. For example in order to design 36 order FIR filter, [6] requires additional 34 Dividers, 34 RNGs, 34 Comparators, 18 2-input Adders, 9 4-input Adders, 4 8-input Adders, 2 16-input Adders and 1 18-input Adders. All of these modules are of 10-bit size. These additional number will increase in square term if this is implemented in FFA form.

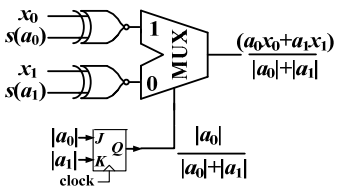

Fig. 8 Proposed 2-input SC inner-product block

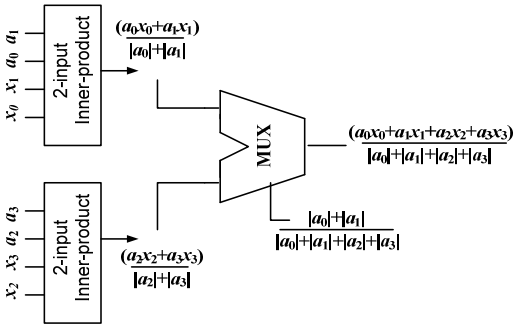

Fig. 9 Proposed 4-input SC inner-product block

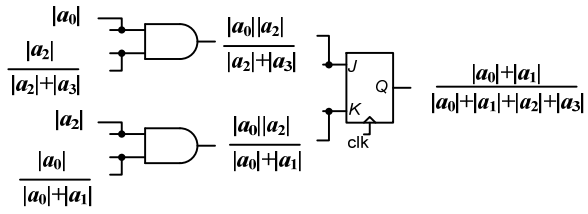

Fig. 10 4-input approximate divider

In case of FIR filter, only the H blocks are implemented using SC modules which compose of inner-products of input *x* and *h* parameter (Equation 6). The delay and the adder modules are created in the BC domain (Fig. 6). Proposed SC FIR filter is unique as it builds on FFA. Thus a 36 order FFA2 requires 18-input inner-products, while FFA3 needs 12-input and FFA6 entails 6-input inner products. On the other hand, Synchronization is designed using 16-input inner products following equation (3).

Apart from re-designing inner-product module, we also implement shuffling network to reduce the number of RNGs. In order to reduce the complexity of BC to SC interface, we introduce a novel RNG sharing scheme. This approach is motivated by the fact that the RNG consumes the largest portion of area and power/energy for each individual BC to SC unit. More specifically, for an *N*-input stochastic computing system that requires *N* copies of RNGs in its BC to SC interface, the area and power/energy consumed by RNGs are very significant. Therefore, the key idea of the proposed approach is to use the randomness offered by a few RNGs to generate the necessary uncorrelated randomness for all input bit-streams. Fig. 11 shows the shuffling network in this proposed SC module.

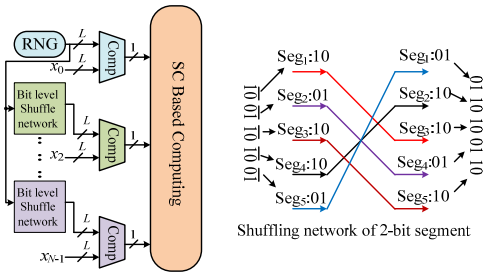

Fig. 11. Shuffling network for random number generations

V. Performance Evaluation

To evaluate the performance of the proposed SC designs, we provide comparative numerical analysis, simulation results, number of gate counts and energy consumptions.

A. Comparative Numerical Analysis

To the best of our knowledge, proposed SC-based Synchronization is first of its kind. Thus, the comparative numerical analysis is generated only for SC FIR filter designs. Nonetheless, the simulation for Synchronization will be presented in the later subsection. The major focus of this numerical analysis is to show the advantage of FFA algorithm-based SC FIR filter. The main merit of the FFA is to reduce the number of filter taps, thus less number of series SC blocks and less SC errors. Recall that the SC multiplication can be done by AND gate when the two SC inputs P(*X*)s are independent to each other. Since RNGs used to generate SC values are not absolutely independent to each other, each input SC processing stage (AND, MUX operations) may incur error. This error propagates through series blocks, getting accumulated towards the output stage. As a result, SC FIR filter design with high number of series blocks produce high error. A comparative error curves are generated for different SC FIR filter designs assuming a 1% error in SC process as shown in Fig.12.For example, considering an SC multiplication/addition operational block produce 1% error, 10 such blocks placed in series (FIR filter is composed of addition and multiplication blocks) will produce an error of (1.01*10-1.0) = 10.4 % error. In this figure, the error generated from SC FIR design proposed in [6] is named as 'inner-product' while the error of proposed SC designs are termed as 'FFA2', 'FFA3' and 'FFA6' (Sec. III)

As seen from the Fig. 12., SC FIR design of [6] produce maximum error because of its higher number of SC series blocks compare to the others. On the other hand, proposed FFA*q* design reduces the *m* order FIR filter into *m*/*q* order FIR sub-filters, thus lowers the number of series blocks proportionately. However, the number of *H* blocks increases by square-term, i.e. more number of outputs from *H* blocks needed to be combined to obtain the filter output (Fig. 6).

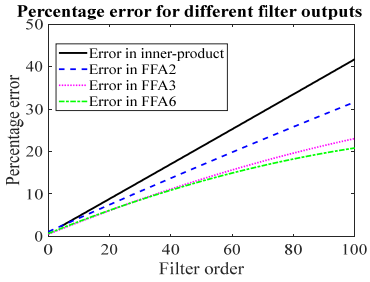

Fig. 12. Errors in different filter outputs

As a result, higher parallelism i.e. high value of *q* will reduce the number of series SC blocks in each *H* block (Fig. 4), while increase the total number of *H* blocks in the overall filter design. Hence, each *H* block may exhibit less error, however, increased number of *H* blocks may increase the overall filter error. In Fig. 12, it is seen that for a filter of order<10, the error performance of FFA2 is similar to that of FFA3 and FFA6, while FFA2 provides less hardware complexity (4 parallel sub-filters in Fig. 6). For a filter of 10<order<40, FFA3 performs better than FFA2 but similar to FFA6, however with lower hardware complexity than FFA6 (9 blocks vs 36 blocks). FFA6 will only be considered better with its higher hardware complexity (36 parallel sub-filters) when the filter order is greater than 45. Therefore, depending on the order of FIR filter, we may choose the value of *q* for FFA*q*that will provide the best SC FIR filter performance.

B. Simulation

1) SC FIR filter design

In addition to the numerical analysis, we also run the simulation to evaluate the output performance. The SC-based FIR filter of order 36 is designed in C programming language emulating different SC hardware blocks from our design. On the other hand, the Impulse Response and the corresponding *h* parameters of the filter are generated using MATLAB R2017 for a band-pass filter of band-8 downlink LTE channel (925-960 MHz) [20]. The input signal is generated from super-positioning of sinusoidal waves of different frequencies as shown in Fig. 13.

The seven green spikes in Fig. 13 (a) are the frequency spectrum of the input signal, generating from the superposition of seven sinusoidal waves whereas the dotted (black) shape is the filter Impulse Response (IR). The blue line corresponding to the output of SC-based FFA FIR filter. In this evaluation, we skip simulation of [6] since it is already outperformed by our SC-based FFA2 in our previous work [2]. Therefore, the output of different FFAs are compared for a 36 order FIR filter. In terms of the performance, FFA3 (Fig. c) and FFA6 (Fig. d) look similar whereas FFA2 (Fig. b) is little worse. The side-band spikes of FFA2 exceed over the side lobes of IR whereas all the side-band spikes of FFA3 and FFA6 are within the side lobes. As a result, the band pass and band rejection of FFA3 and FFA6 are better compared to FFA2. Out of all three designs, FFA3 is considered the best for a 36-order FIR filter since it shows similar performance of FFA6 with less hardware complexity.

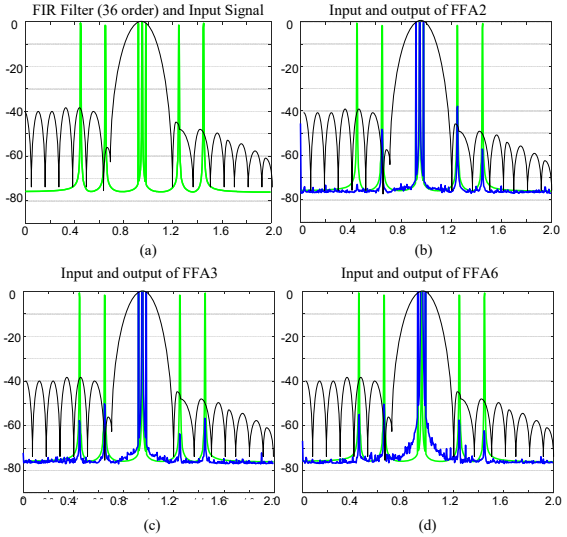

Fig. 13. Comparative output spectrum of FFA2, FFA3 and FFA6

2) SC Synchronization design

To evaluate the performance of our SC-based synchronization, the hardware circuit is again implemented in C-language and the corresponding correlation output is generated from the reference and known signals (Fig. 14).

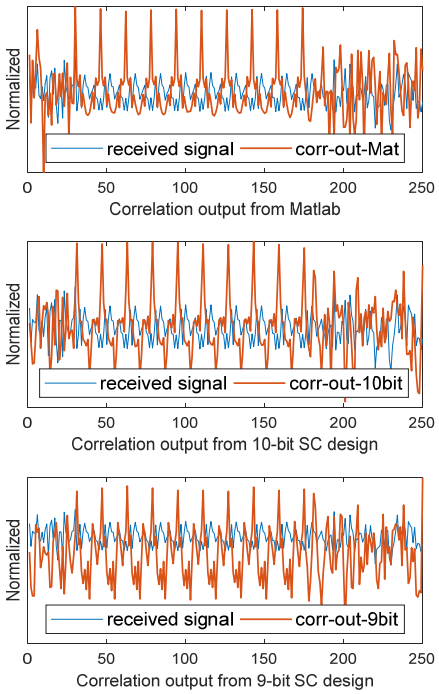

Fig. 14 Synchronization Outputs from Different methods

Recall that each preamble in IEEE 802.11 is generated by concatenating a set of 10 short training sequences (STS)

discussed earlier. Rayleigh fading model is considered for the wireless medium to incorporate the wireless environment. To make a fair comparison, the output obtained from equation (3) using MATLAB is juxtaposed with the output of SC-based design. Moreover, SC-based Synchronization outputs are generated for bit precisions of 10-bit and 9-bit. Since one-bit reduction in SC-design can shorten the total delay by half, lower bit-sized SC designs are always preferable. It is observed from Fig. 14 that outputs of both 10-bit and 9-bit SC designs can detect the location of 10 STS accurately as well as results by MATLAB. Since reference signal has the same period as that of the correlated outputs, i.e. 10 STS, the detection of the beginning of a packet of 9-bit is still possible. However, the performance of 10-bit is better than that of the 9-bit.

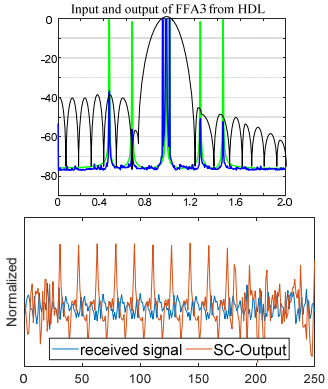

Fig. 15 Input and outputs from HDL

3) *Hardware Implementation of SC Modules*

After having positive results from C programming implementation, we tested the efficacy of our modules by designing in Hardware Description Language (HDL). We considered FFA3 for FIR filter and 10-bit design for Synchronization. The Xilinx ISE design tool is used to implement these modules in verilog HDL. The output from the HDL is shown in Fig. 15, matching the output from C implementation.

C. *Gate counts and Energy Consumptions*

Gate counts is another way to evaluate the occupation of silicon space. It is also related to the energy consumption since high number of gates contribute to high energy dissipation[21]. Because NAND gate is a universal gate, we build all the basic SC block functions with NAND gates. Thus the gates counts corresponding to a design is the number of NAND gates required to create the complete SC module. Recall that for fixed a's, pre-calculated $|a_0|$ /$[|a_0|+|a_1|]$ is used for [6] before converting to its SC value and is termed as [6] Fixed. On the other hand, for dynamic values of a's, an additional Divider module is required for [6] and this is referred to as [6] Dynamic.

Table 1 shows the gate counts of different 36-order FIR filter designs. Here, BC-H/W is the gate counts of BC-based FIR filter. From the table it is absolutely clear that the proposed SC-based FIR filter designs require less number of gates than that of the other designs. Moreover, all SC-based designs occupy way less silicon than that of the BC-based design. On the other hand, [6] Fixed design has very similar gate counts to FFA6 design. However, for any practical implementation, only [6] Dynamic can be used. From the performance of simulation and gate counts, FFA3 obviously stood the best among all SC-based design for 36-order FIR filter. In other words, proposed FFA3 SC-design occupies silicon area lower than 60% of the [6] Dynamic and 25% of the BC-H/W.

TABLE1
GATE COUNTS FOR 36-ORDER FIR FILTER

FFA2	*FFA3*	*FFA6*	[6] Fixed	[6] Dynamic	*BC-H/W*
8,274	9,890	12,240	13,280	16,520	42,540

TABLE2
GATE COUNTS FOR SYNCHRONIZATION

Propose	[6] Fixed	*BC-H/W*
3,356	6,140	20,540

Table 2 shows gate counts of different SC-based designs for Synchronization. Since the reference signal is fixed, the [6] Fixed is used here for comparison. It is seen from Table 2 that the proposed SC-based Synchronization requires less gate counts compare to [6] Fixed and BC-H/W based ones. Moreover, for an IoT device where both of these modules are installed, the actual gate counts would be the addition of these two. That is, the proposed SC-based FIR and Synchronization necessitates a total of 13,246 gates, [6] requires a total of 22,660 and BC-H/W entails a total of 63,080. Clearly, the propose SC-based designs significantly reduces the number of gate counts compare to that of both [6] and BC-H/W.

Energy consumption is a critical issue for resource constraint (battery operated) IoT devices. According to the existing design, both the FIR filter and the Synchronization utilize processing power of CPU. Our proposed SC-based hardware design can significantly reduce this power consumption. For the purpose of evaluation, energy consumption of hardware is calculated from the equivalent circuit composed of resistance and capacitance. The equivalent circuit for the designs are derived from the circuit model of NAND gate [19, 22, 23]. On the other hand, the energy consumption of a software based one is calculated from the CPU process of Raspberry Pi (mostly used in IoT) drawing an active current of 50mA with 700 MHz clock speed and 5V.

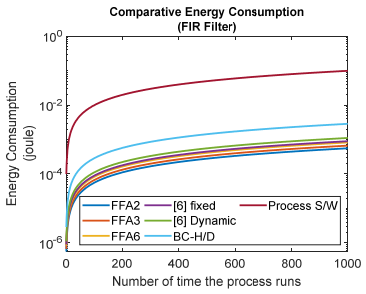

Fig. 16. Energy Consumptions from different FIR filters

Fig. 16 shows energy consumption from different hardware FIR filter designs and software based FIR filter process (Process S/W). It is seen that the software based process dissipate energy of the order of 4 more than the hardware based ones. Fig. 17 shows the energy consumption of different hardware-based FIR filter designs. It is seen from Fig. 17 that the proposed one (FFA3) consumes below50% of the energy

consume by [6] (Dynamic) and 20% of BC-H/W.

Energy consumption of Synchronization shows similar pattern in Fig. 18. The software-based one obviously incurs huge energy dissipation. The proposed one consume below 70% of the one based on [6] and 15% of that of the BC-H/W. In view of the total energy consumption by the two modules (FIR and Synchronization) running in the IoT device, the proposed one consume 50% of the one in [6] and 15% of that of BC-H/W.

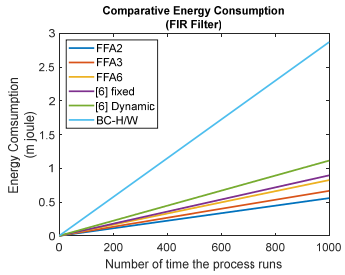

Fig. 17. Energy Consumptions from H/W-based FIR filters

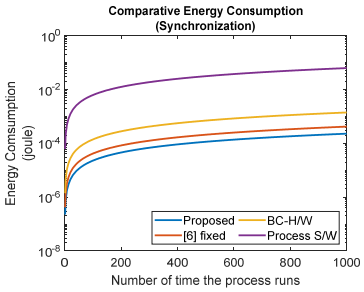

Fig. 18. Energy Consumptions from different Synchronization

VI. Conclusion

Resource constraint IoT devices are becoming a big part of 5G internet. In order to seamlessly co-exist with the other devices, they have to keep all the necessary processes with as little energy and silicon area consumption as possible. Among the base-band processes, FIR filter and Synchronization is considered the most energy consumption ones and the center of our proposed work. Our novel hardware-based designs inherently reduce the energy consumption and SC-based accelerator reduce the silicon area. From our comparative analysis, it is obvious that the software based processes entail higher energy expenditure that can reduce the life time of these devices significantly. On the other hand, SC-based hardware not only occupy less silicon space (less than half), but also consume lower energy (of the order of 4) compared to BC-based ones. Moreover, it is clearly shown that the proposed designs produce significantly less energy (less than 70%), occupy less silicon area (less than 60%) and perform better compared to other SC-based designs.

Acknowledgement

This research is supported by NSF IRNC (#2029295) and NSF JUNO2 (#1818884).

References

[1] Y. Chen, S. Lu, H. Kim, D. Blaauw, R. G. Dreslinski and T. Mudge, "A low power software-defined-radio baseband processor for the Internet of Things," *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Barcelona, 2016, pp. 40-51.

[2] K. J. Ahmed, Bo Yuan, Myung J. Lee, "High-accuracy Stochastic Computing-based FIR Filter Design," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018.

[3] K.K. Parhi, *VLSI Digital Signal Processing Systems, Design and Implementation*, Wiley, Inter-Science, 1999.

[4] D. A. Parker, K. K. Parhi, " Low-Area/Power Parallel FIR Digital Filter Implementations,"*Journal of VLSI Signal Processing systems for signal, image and video technology*,Sept 1997, pp. 75-92.

[5] Y. Oppenheim, R. W. Schafer, J. R. Buck, et. al., Discrete-time signal processing, vol. 2. Prentice-hall Englewood Cliffs, 1989.

[6] Y. N. Chang and K. K. Parhi, "Architectures for digital filters using stochastic computing," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, 2013, pp. 2697-2701

[7] B. Gaines, "Stochastic computing systems," Advances in Information Systems Science, vol. 2. no. 2. pp. 37-172, 1969

[8] P. Li, D. J. Lilja, W. Qian, K. Bazargan and M. D. Riedel, "Computation on Stochastic Bit Streams Digital Image Processing Case Studies," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 449-462, March 2014.

[9] W. Qian, X. Li, M. D. Riedel, K. Bazargan and D. J. Lilja, "An Architecture for Fault-Tolerant Computation with Stochastic Logic," in *IEEE Transactions on Computers*, vol. 60, no. 1, pp. 93-105, Jan. 2011.

[10] Fan Wang, J. J. Chen and G. Agami, "Optimal receiver filter design with applications in IS-2000 CDMA systems," *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, New Orleans, LA, USA, 2003, pp. 496-501 vol.1.

[11] X. Wang, T. Wild and F. Schaich, "Filter Optimization for Carrier-Frequency- and Timing-Offset in Universal Filtered Multi-Carrier Systems," *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, Glasgow, 2015, pp. 1-6.

[12] J. Wen, J. Hua, S. Li, K. Zhou and D. Wang, "Interference-driven designs of nonlinear-phase FIR filter with application in FBMC system," in *China Communications*, vol. 13, no. 12, pp. 15-24, December 2016.

[13] S. Nandula and K. Giridhar, "Robust timing synchronization for OFDM based wireless LAN system," *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region*, Bangalore, India, 2003, pp. 1558-1561 Vol.4.

[14] B. Kim, S. Lee, Seung Sik Lee and S. Choi, "Preamble generation method to improve timing estimation for OFDM system using training sequence," *2015 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, 2015, pp. 142-143.

[15] S. K. Manusani, R. S. Kshetrimayum and R. Bhattacharjee, "Robust Time and Frequency Synchronization in OFDM based 802.11a WLAN systems," *2006 Annual IEEE India Conference*, New Delhi, 2006, pp. 1-4.

[16] B. Yuan and Y. Wang, "High-Accuracy FIR Filter Design Using Stochastic Computing," *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Pittsburgh, PA, 2016, pp. 128-133.

[17] A. Ardakani, F. Leduc-Primeau and W. J. Gross, "Hardware implementation of FIR/IIR digital filters using integral stochastic computation," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, 2016, pp. 6540-6544.

[18] *https://en.wikipedia.org/wiki/LTE_frequency_bands,* Last visited on Oct 18th, 2017.

[19] A. Wiltgen, K. A. Escobar, A. I. Reis and R. P. Ribas, "Power consumption analysis in static CMOS gates," *2013 26th Symposium on Integrated Circuits and Systems Design (SBCCI)*, Curitiba, 2013, pp. 1-6.

[20] 3GPP, TS 36.331 V14.2.2, Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification.

[21] R. Piyare, A. L. Murphy, C. Kiraly, P. Tosato, and D. Brunelli, "Ultra Low Power Wake-Up Radios: A Hardware and Networking Survey,"*IEEE Communications Surveys & Tutorials,* Vol. 19, No. 4, Fourth Quarter 2017.

[22] Kanika Kaur and Arti Noor, " Power Estimation Analysis for CMOS Cell Structures," *International Journal of Advances in Engineering & Technology*, May 2012

[23] J. M. Rabaey, A. Chandrakasan, B. Nikolic, Digital Integrated Circuits A design Perspective January, Prentice Hall Electronics and VLSI Series, 2002