This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2021.3051848, IEEE Design and Test

High Performance Deterministic Stochastic Computing Using Residue Number System

Kamyar Givaki[1], Reza Hojabr[1], M. H. Gholamrezaei[2], Ahmad Khonsari[1,2], Saeid Gorgin[3], Dara Rahmati[4], and M. Hassan Najafi[5]

[1] School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
[2] School of Computer Sciences, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
[3] Iranian Research Organization for Science and Technology (IROST), Tehran, Iran
[4] Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran
[5] School of Computing and Informatics, University of Louisiana, Lafayette, LA, USA
{givakik, r.hojabr}@ut.ac.ir, gholamrezaei@ipm.ir, a_khonsari@ut.ac.ir, gorgin@irost.ir, d_rahmati@sbu.ac.ir, najafi@louisiana.edu

***Abstract*—Stochastic Computing (SC) is a re-emerging computing paradigm with a great potential to surpass conventional binary implementations in term of hardware cost. The inaccuracy of computations is an important challenge with conventional SC designs. Recently, some deterministic approaches to SC were proposed. These methods are able to produce completely accurate results. However, they take a long processing time to produce exact results which directly translates to very high energy consumption. This work proposes a design methodology based on the Residue Number System (RNS) to mitigate the long processing time of the deterministic methods of SC. Leveraging RNS, the length of bit-streams decreases *exponentially* as high bit-width operands are replaced with low bit-width residues. Compared to the state-of-the-art deterministic methods, the proposed approach delivers more than 2400× and 3200× reduction in processing time and energy consumption, respectively, for 8-bit multiplication operation. Synthesis results further show that for a processing element (PE) in DNN accelerators, the proposed design occupies 84%, 52% and 30% less area compared to the conventional binary, RNS, and deterministic designs of SC, respectively.**

***Index Terms*—Stochastic computing, deterministic bit-stream processing, residue number system, accelerator, energy efficiency.**

I. Introduction

Stochastic computing [1], [2], an unconventional computing paradigm processing bit-streams, has been recently gained considerable attention in the hardware design and computer architecture communities. Higher tolerance to noise and lower hardware cost compared to conventional binary-radix-based designs are the most appealing advantages of SC. Inaccuracy of computation and long processing time, however, are the major weaknesses of SC. The inaccuracy is mainly due to random fluctuations in generating bit-streams and to correlation (independence) between bit-streams [1]. The common method to improve the accuracy is to increase the length of the bit-streams. However, longer bit-stream means longer latency and higher energy consumption.

Recently, some *deterministic* approaches for SC were proposed [3], [4]. These methods are able to perform completely accurate computation with SC circuits. In these methods, the required independence between bit-streams is provided by three approaches: rotation of bit-streams [3], clock dividing bit-streams [3], or using bit-streams with relatively prime lengths [4]. These methods guarantee exact computation by processing the bit-streams for the product of the length of the input bit-streams. For example, to multiply two n-bit precision data, each represented using a bit-stream of length 2^n, the deterministic methods take 2^{2n} cycles to produce the exact result which is a bit-stream of 2^{2n} bit length. Although the deterministic methods outperform the conventional SC in term of computation accuracy, the long processing time and consequently high energy consumption (power × time) limit their application.

In this work, we propose a novel approach to mitigate the long processing time of the current deterministic methods of SC. We integrate the deterministic methods with the Residue Number Systems (RNS) [5] to reduce the length of bit-streams *exponentially*. RNS is a non-weighted number representation that offers a high degree of parallelism and modularity. In RNS, each number is decomposed into a set of residues that are smaller than the primary number. The arithmetic operations are performed on each residue independently and in parallel with the operations on other residues. This makes RNS a computation technique free of carry propagation. The parallelism offered by RNS is a unique property that distinguishes it from other number systems like Logarithmic Number System (LNS). The proposed design outperforms the prior designs developed based on the deterministic approaches in terms of processing time and energy consumption. Synthesis results on implementation of multiplication operation show that the proposed approach reduces the hardware area and power consumption more than 88% and 56%, respectively, compared to those of the conventional 32-bit binary implementation. In summary, the main contributions of this work are as follows:

i. We integrate the deterministic methods of SC with the RNS representation. The proposed design provides a lower hardware area and power consumption compared to the conventional binary and RNS implementations.
ii. The proposed design methodology results in a significant reduction in the number of processing cycles (ratio of $2^{2*(n-\lceil \log_2 n \rceil)}$ where n is the data bit-width) compared

to the state-of-the-art deterministic methods of SC.

iii. We propose an FSM-based reverse converter to directly convert the RNS-based bit-streams to binary numbers.

iv. Evaluating our approach on a state-of-the-art processing element (PE) for convolutional neural networks [6] shows 84%, 52% and 30% area reduction, respectively, compared to the baseline binary, RNS, and clock division deterministic implementation.

The rest of the paper is structured as follows. Section II presents the necessary background on SC, deterministic methods of SC, and also RNS. Section III describes the proposed methodology and suggests an intuitive implementation for the micro-architecture of the proposed processing system. Section IV provides the results and compares our method with state-of-the-art works. Finally, Section V concludes the paper.

II. Preliminaries

A. Stochastic Computing

In SC, computation is performed on random or unary bit-streams [4]. The ratio of the number of 1s to the length of a bit-stream determines the value of the bit-stream. Complex arithmetic operations are implemented using simple logic gates in this paradigm. For example, an AND gate can be used to multiply two numbers in stochastic domain [2]. With stochastic representation, a stream of 2^n bits can precisely represent any n-bit precision number. A number in binary format is conventionally converted to a stochastic bit-stream by comparing a pseudo-random or an increasing/decreasing number to the target number [4]. The output of comparison generates one bit of the bit-stream at any cycle. A stochastic bit-stream can be converted back to the binary format by simply counting the number of ones in the bit-stream using a binary counter [1].

A requirement for performing operations such as multiplication in the stochastic domain is to convert the data into *independent* bit-streams. In conventional SC, this independence is provided by using different sources of random numbers in generating bit-streams. However, due to random fluctuations in generating random bit-streams and also correlation between bit-streams, the computations in conventional SC are not accurate [4].

B. Deterministic methods of SC

Three deterministic methods for SC were proposed recently [3], [4]. By properly structuring bit-streams, accurate and deterministic computations are carried out using SC logic. The deterministic approaches guarantee the independence by using bit-streams with relatively prime lengths, clock dividing bit-streams, and rotation of bit-streams. These methods guarantee that each bit of one operand (bit-stream) sees every bit of the other operands (bit-streams) exactly once, resulting in a deterministic and completely accurate computation. Fig. 1 exemplifies the three deterministic methods. The first method simply uses two bit-streams with relatively prime lengths. It iterates them until they have a length equal to the product of the length of the bit-streams. In the second method (i.e., clock division method), the second bit-stream is clock divided by the length of the first bit-stream. Finally, in the rotation method, the second bit-stream is stalled for one cycle after each time generating the first bit-stream.

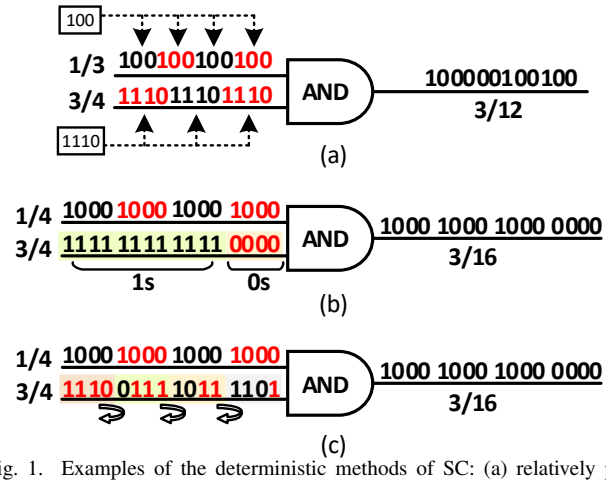

Fig. 1. Examples of the deterministic methods of SC: (a) relatively prime bit-stream lengths, (b) clock division, (c) rotation method.

All these deterministic methods guarantee accurate computation if generating and processing bit-streams for the *product of the length of the input bit-streams*. This means that processing i n-bit precision data, each represented by a 2^n-bit bit-stream, takes $2^{i\times n}$ cycles to calculate an accurate result. This long processing time translates to high energy consumption making the current deterministic methods inefficient for many applications. In this work, we exploit the RNS number representation to mitigate the long processing time of the deterministic methods of SC.

C. Residue Number System

RNS is a number representation system in which a number is represented by its residues when divided by the members of a set of relatively prime integers (called *moduli set*). Formal definition of RNS can be written as follows:

- Considering a set of L relatively prime numbers $< m_1, m_2, ..., m_L >$, an integer value x is represented by $< x_1, x_2, ..., x_L >$ where $x_i = x \ mod \ m_i$.

An important advantage of using RNS is that complex and costly operations are split into several independent and simple operations running in parallel [5]. For example, a 2-input integer multiplication ($x \times y$) is performed in three steps:

- Convert x and y to their RNS representation, i.e., (x_i, y_i) where $i = 1,...,L$ and L is cardinality of the moduli set.
- Multiply each pair of (x_i, y_i)
- Convert the multiplication result back to conventional binary representation.

This computation flow can be extended to other arithmetic operations such as addition and subtraction. Considering the fact that the residues are significantly smaller than the original number, a set of significantly shorter bit-streams can be used to precisely represent each number.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2021.3051848, IEEE Design and Test

An RNS system with moduli set $< m_1, m_2, ..., m_L >$ represents $DR = m_1 \times m_2 \times ... \times m_L$ different numbers. DR is called the *dynamic range* of the system. To represent an n-bit binary number, the dynamic range of the selected moduli set should be greater than 2^n to guarantee that all the numbers in the range of [0, $2^n - 1$] are covered. Fig. 2(a) illustrates how to calculate $x \times y$ in RNS. As shown, the main operation is decomposed into three smaller operations running in parallel. If the result of an operation is greater than its corresponding residue, the result should be divided by the residue to achieve the remainder. *Chinese Reminder Theorem* (CRT) [7] is a method to convert a number in the RNS format to its corresponding binary representation (reverse conversion).

There have been some prior works employing RNS in acceleration of neural networks [8], [9]. Our work is essentially different from those works in the sense that we exploit RNS to improve the performance of the deterministic methods of SC. This is realized by the bit-width reduction offered by RNS which results in exponential reduction in the length of bit-streams and so in the the number of processing cycles.

III. Proposed Design

In this section, we elaborate the proposed approach. In the deterministic methods, the number of clock cycles required to perform an operation follows an exponential function of the binary bit-width of the operands. Multiplying two n-bit precision data takes 2^{2n} clock cycles as 2^{2n}-bit bit-streams must be processed [4]. Using high-bit-width binary numbers as the operands of the operations leads to very long processing time and hence very high energy consumption. Exploiting the inherent parallel nature of the RNS, we split the high-bit-width operands of the computations into operands (residues) with low bit-widths. While the idea is presented based on the multiplication operation, the proposed method is also applicable to the deterministic scaled addition/subtraction [4].

A. Precise SC in RNS Domain

Dividing the computation into several parallel lanes, each lane processing low bit-width operands, decreases the complexity of the computation. Decreasing the bit-width of the operands means exponentially shorter bit-streams to represent each data. Fig. 2 illustrates the idea of combining the deterministic methods of SC and RNS through a simple example. As shown in Fig. 2(b), multiplying two 8-bit precision numbers with the deterministic methods takes $2^{2\times8}$ clock cycles. Fig. 2(c) shows how the proposed method reduces the number of processing cycles. With RNS, the precision of the operands decreases to three bits. Hence, it takes only $2^{2\times3}$ clock cycles to perform all sub-computations, which means $2^{10}\times$ fewer cycles compared to the non-RNS design. Note that the proposed RNS-based design has some hardware cost overheads to convert the outputs back from RNS to the binary format. In many applications, this conversion can be postponed to the final stage of computation. We will show that, even with the conversion overhead, the proposed design provides a higher energy efficiency compared to the current deterministic designs of SC due to an exponential decrease in the processing time.

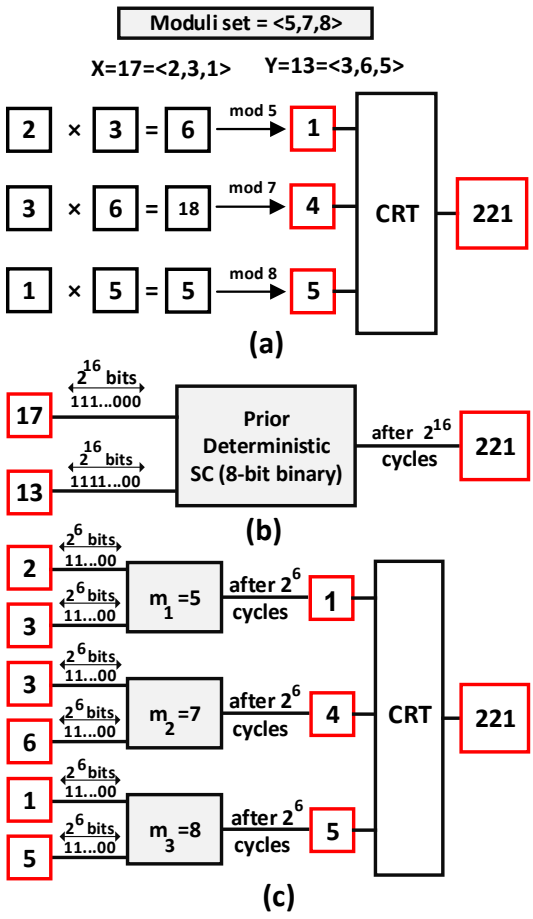

Fig. 2. (a) RNS multiplication (b) previously proposed deterministic SC [4] (c) our proposed RNS-based deterministic method.

B. Moduli Set Selection

Table I compares the processing time of the proposed design and the deterministic designs for different data precisions and moduli sets. As it can be seen, the performance improvement offered by the proposed design changes by changing the moduli set (both the number of moduli and the value of each modulus). So, choosing a proper moduli set is an important step to achieve the best performance using the proposed technique.

The symmetricity of the design is another important design aspect that should be considered. In conventional RNS, circuits for each operation in different moduli have different timing characteristics. For instance, the critical path (CP) delay of a modular multiplier in mode 2^n is much lower than the CP delay for a modular multiplier in mode $2^n - 1$. Typically, there are more than two moduli in a moduli set. The difference between the CP of the fastest and the slowest moduli is considerable. The slowest modulus determines the overall system's speed. In our proposed method, the CP delay depends on the binary precision (bitwidth) of the moduli; the modulus

TABLE I
PROCESSING TIME COMPARISON OF THE PROPOSED METHOD AND THE DETERMINISTIC METHODS

Binary Precision	Moduli Set	Prior Deterministic [3], [4] (cycles)	Proposed Method (cycles)	Clock Cycles Reduction Ratio
8-bits	8,7,5	2^{16}	2^{6}	2^{10}
16-bits	16,15,13,11,7	2^{32}	2^{8}	2^{22}
16-bits	64,63,61	2^{32}	2^{12}	2^{20}
24-bits	32,31,29,27,25	2^{48}	2^{10}	2^{38}
24-bits	64,63,61,59,5	2^{48}	2^{12}	2^{36}
24-bits	512,511,509	2^{48}	2^{18}	2^{30}
32-bits	2048,2047,2045	2^{64}	2^{22}	2^{42}

with the same bit-width have the same CP delay and the modulus with different bit-widths have different CP delays. For example, in the moduli set $(64, 63, 61, 59, 5)$, circuits for moduli 64, 63, 61, and 59 have the same CP delay which is different to the CP delay of the circuit for modulus 5. Another concern is the number of required clock cycles to complete the computations. In the moduli set $(64, 63, 61, 59, 5)$, the result of modulus 5 is ready after $2^{2\times3}$ cycles, while the results of other moduli are ready after $2^{2\times6}$ cycles. Thus, some parts of the circuit are underutilized for a large fraction of processing time. To avoid this type of inefficiency, we pick up a moduli set where all the modulus have the same bit-width and so take the same number of processing cycles.

In this work, we use the following moduli set with three members: $(2^m-3,\ 2^m-1,\ 2^m)$. These numbers are the three greatest relatively prime numbers represented by m bits. The residues of dividing any integer number by these three moduli can be represented by the same number of bits (i.e., m bits). It is easy to prove that the inequality in Eq. (1) is true for any $m > 2$. The inequality shows that these three numbers can represent any $n = 2^{3m-1}$-bit precision binary number. For example, with $m = 3$, the resulted RNS can represent any 8-bit precision binary number.

$$(2^m-3)\times(2^m-1)\times(2^m) \geq 2^{3m-1} \tag{1}$$

Proof. consider $2^m = X$. Eq. (1) can be rewritten as follows:

$$(X-3)\times(X-1) \geq \frac{X^2}{2} \tag{2}$$

Eq. (2) holds when $X \geq 8$. So, for $m \geq 3$, Eq. (1) is always true.

Example: Assume A is a 9-bit binary number. With the moduli set $(2^m-3,\ 2^m-1,\ 2^m)$, the smallest bit-width for RNS representation of A is given by $3m-1 \geq 9$. The smallest integer number that satisfies this inequality is 4.

Table II compares the number of processing cycles for the case of multiplying two binary numbers with the proposed design when using $(2^m-3,\ 2^m-1,\ 2^m)$ as the moduli set, and with the current deterministic methods of SC. Multiplying two n-bit precision data with the deterministic methods takes 2^{2n} cycles. The reduction ratio (the number of cycles with the deterministic methods divided by the number of cycles with the proposed method) is formulated in Eq. 3.

TABLE II
REQUIRED NUMBER OF CLOCK CYCLES WHEN MULTIPLYING TWO VALUES WITH THE PROPOSED METHOD BASED ON $(2^m-3,\ 2^m-1,\ 2^m)$ MODULI SET VS. WITH THE DETERMINISTIC SC.

Binary Bit-width	Modulus Bit-width	Prior Deterministic [3], [4] (cycles)	Proposed Method (cycles)	Clock Cycles Reduction Ratio
8	3	2^{16}	2^{6}	2^{10}
9	4	2^{18}	2^{8}	2^{10}
10	4	2^{20}	2^{8}	2^{12}
11	4	2^{22}	2^{8}	2^{14}
12	5	2^{24}	2^{10}	2^{14}
13	5	2^{26}	2^{10}	2^{16}
14	5	2^{28}	2^{10}	2^{18}
15	6	2^{30}	2^{12}	2^{18}
16	6	2^{32}	2^{12}	2^{20}
17	6	2^{34}	2^{12}	2^{22}
18	7	2^{36}	2^{14}	2^{22}
19	7	2^{38}	2^{14}	2^{24}
20	7	2^{40}	2^{14}	2^{26}

$$reduction\_ratio = 2^{2*(n-\lceil \log_2 n \rceil)} \tag{3}$$

As can be seen in Table II, the reduction ratio increases by increasing the data bit-width which means higher savings in the processing time and energy consumption. We note that the selected moduli set is not efficient for processing of low bit-width operands, e.g., for $n \leq 8$. For processing of such operands other modules sets such as $(2^m-1,\ 2^m)$ is recommended.

C. Hardware Architecture

Fig. 3 demonstrates a high level architecture of the proposed design for n-bit precision multiplication. The moduli set is $(2^m-3,\ 2^m-1,\ 2^m)$. An input buffer is used to store two distinct numbers, A and B, in the RNS format. These two numbers are provided as the inputs of the circuit, each using three residues, i.e., A_{2^m}, A_{2^m-1}, and A_{2^m-3}. We use the low-cost FSM-based bit-stream generator proposed in [10] to convert the residues to bit-stream representation. Converting each residue requires a separate multiplexer (MUX) unit. Two FSMs are reused and connected to the select input of the MUX units to choose one of the binary bits of the residue at any cycle. To guarantee the required independence between the two inputs of each multiplication operation, each one of the two FSMs is configured to generate a different low-discrepancy pattern [10]. Each FSM is connected to three MUX units, each MUX for converting one of the three residues. In cases that the architecture is extended to perform several multiplications in parallel, the two FSMs will be shared between more multipliers and their effective overhead will be lower.

As shown in Fig. 3(a), three parallel `AND` gates are employed to perform the multiplication operations on the residues. Each `AND` gate is followed by a counter to convert the produced output bit-stream to binary representation. To calculate the residues of the outputs, the *reset* signal of each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2021.3051848, IEEE Design and Test

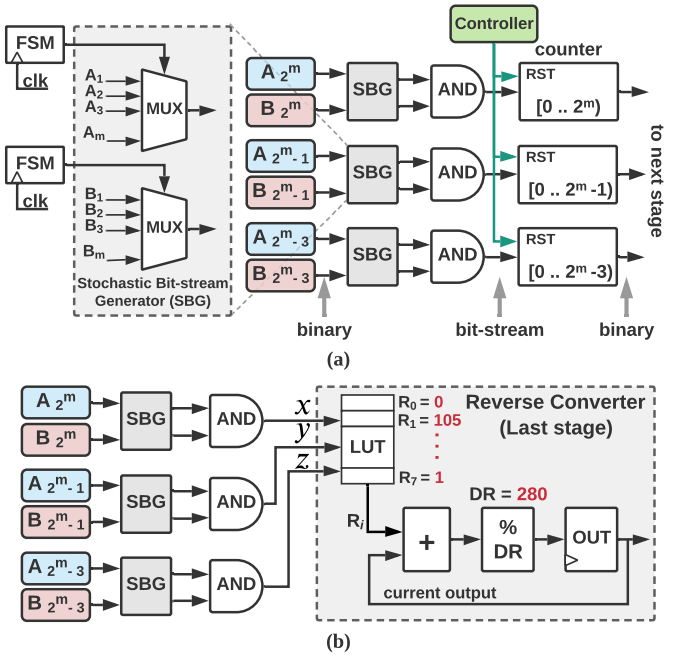

Fig. 3. Microarchitecture of the proposed method for multiplication operation (a) intermediate stages without reverse conversion, (b) last stage with Reverse Converter (RC).

counter is set to the corresponding modulus. The state of these counters also plays a register role to store the final results. The controller manages other aspects of the system for correct operation. We note that in contrast to the conventional SC designs in which streaming inputs can be processed partially, the proposed design needs to process the entire bit-stream to accurately perform RNS computations.

D. Reverse Conversion

In applications where we have several consecutive stages of operations in a pipelined fashion, there is no need to convert the RNS numbers back to binary format in the intermediate stages. In fact, all the intermediate computations can be performed in the RNS domain and the RNS-to-binary conversion will be confined to the last stage of the computations. A naive method to perform this reverse conversion is by using a CRT module [11]. The CRT module receives the residue numbers from the counters and converts them to the equivalent representation in the binary domain. However, CRT requires a high-power logic circuit which reduces the effectiveness of the RNS method. Here, we propose an efficient FSM-based Reverse Converter (RC) to directly convert the residue bit-streams to binary representation. Since the inputs of the proposed RC are RNS-based numbers in the bit-stream format, the counters can be removed from the last stage.

Table III depicts the transitions of the proposed RC. The FSM has DR states, entitled 0 to $DR-1$. The converted result (output) is equal to the final state's number after iterating over all input bit-streams. In Table III, S_c represents the current state, $\%$ shows the Mod operation, and R_i shows the reversed

TABLE III
THE STATE TRANSITION TABLE OF THE PROPOSED FSM-BASED RC. S_0 IS THE INITIAL STATE. x, y, AND z SHOW THE OUTPUT BIT-STREAMS OF EACH COMPUTING LANE AT ANY CYCLE. c IS THE CURRENT OUTPUT OF THE REVERSE CONVERTER.

#	x (2^m-3)	y (2^m-1)	z (2^m)	Next State (Output)	Example $(m=3, DR=280)$
0	0	0	0	$S_{(c+R_0)\%DR}$	S_c
1	0	0	1	$S_{(c+R_1)\%DR}$	$S_{(c+105)\%280}$
2	0	1	0	$S_{(c+R_2)\%DR}$	$S_{(c+120)\%280}$
3	0	1	1	$S_{(c+R_3)\%DR}$	$S_{(c+225)\%280}$
4	1	0	0	$S_{(c+R_4)\%DR}$	$S_{(c+56)\%280}$
5	1	0	1	$S_{(c+R_5)\%DR}$	$S_{(c+161)\%280}$
6	1	1	0	$S_{(c+R_6)\%DR}$	$S_{(c+176)\%280}$
7	1	1	1	$S_{(c+R_7)\%DR}$	$S_{(c+1)\%280}$

binary equivalent of the converter's input vector, where i is the integer value of the input vector. Since our moduli set has three modulus, the array R is of size $2^3 = 8$ which is independent of m. For instance, $R_1 = 105$ is the only number in the dynamic range whose remainders of division by 2^3-3, 2^3-1, and 2^3 are equal to 0, 0, and 1, respectively. Similarly, given $m = 3$, $DR = (2^3-3) \times (2^3-1) \times (2^3) = 280$, and $R = \{0, 105, 120, 225, 56, 161, 176, 1\}$ can be calculated and stored in a look-up table at design time. Fig. 3(b) shows the equivalent hardware needed for the reverse conversion. The output register is initialized to zero, hence the initial state in the transition table is S_0. In each cycle, the RC unit chooses one of the stored values in the LUT based on its input values, and calculates the next state (output). Note that the forward conversion (binary-to-RNS conversion) can be done in the proposed design in an offline manner at no additional hardware cost.

IV. Experimental Results

To evaluate the performance of the proposed design we implement a two-input multiplier with various bit-widths by 1) our proposed design, 2) conventional binary design, 3) conventional RNS, and 4) clock division deterministic design [4]. We also evaluate the hardware efficiency of the multiplier when using the proposed design with the FSM-based RC compared to an RNS-based design that uses the RC of [12]. We do not compare with the conventional nondeterministic SC due to its inability to produce accurate results and its significantly higher latency and energy consumption compared to the deterministic methods of SC [4]. The conventional RNS design is implemented based on $(2^m-1, 2^m, 2^m+1)$ moduli set as it provides the minimum hardware cost compared to other moduli sets including $(2^m-3, 2^m-1, 2^m)$. We further compare the cost of implementing a PE proposed in a state-of-the-art convolutional neural network accelerator [6] based on binary, RNS, and proposed design methodology. To evaluate the hardware efficiency of the implemented designs we develop a cycle-accurate micro-architectural simulator in RTL VHDL. We use the Synopsys Design Compiler to synthesize the implemented designs with the 45nm FreePDK gate library.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2021.3051848, IEEE Design and Test

TABLE IV
THE SYNTHESIS RESULTS FOR TWO-INPUT MULTIPLICATIONS IN 45NM TECHNOLOGY. THE RESULTS ARE SHOWN FOR DIFFERENT BIT-WIDTHS OF OPERANDS.

Bit-Width	8-bit							12-bit						
Design Approach	CP (ns)	Area (μm^2)	Power @ 100MHz (mW)	Power @ Max Freq. (mW)	EPC (pJ)	# of Cycles	Energy @ Max Freq. (pJ)	CP (ns)	Area (μm^2)	Power @ 100MHz (mW)	Power@ Max Freq. (mW)	EPC (pJ)	# of Cycles	Energy @ Max Freq. (pJ)
Deterministic SC	1.06	957.38	0.07	0.60	0.64	2^{16}	4.16E+4	1.21	1426.20	0.10	0.72	0.88	2^{24}	1.46E+7
Our proposed	**0.45**	**328.51**	**0.02**	**0.45**	**0.20**	2^{6}	**1.29E+1**	**0.63**	**563.63**	**0.04**	**0.50**	**0.32**	2^{10}	**3.22E+2**
RNS	0.75	431.76	0.03	0.34	0.26	1	2.56E-1	0.95	1016.03	0.08	0.77	0.73	1	7.31E-1
Binary	1.54	1068.13	0.08	0.48	0.74	1	7.39E-1	2.40	2277.99	0.18	0.68	1.63	1	1.63E+0
Our proposed + RC	**1.09**	**489.48**	**0.04**	**0.28**	**0.30**	2^{6}	**1.95E+1**	**1.70**	**817.99**	**0.06**	**0.34**	**0.58**	2^{10}	**5.91E+2**
RNS + RC	1.68	669.69	0.06	0.31	0.52	1	5.21E-1	2.01	1460.93	0.15	0.71	1.43	1	1.43E+0
Bit-Width	16-bit							32-bit						
Design Approach	CP (ns)	Area (μm^2)	Power @ 100MHz (mW)	Power @ Max Freq. (mW)	EPC (pJ)	# of Cycles	Energy @ Max Freq. (pJ)	CP (ns)	Area (μm^2)	Power @ 100MHz (mW)	Power @ Max Freq. (mW)	EPC (pJ)	# of Cycles	Energy @ Max Freq. (pJ)
Deterministic SC	1.26	1922.26	0.13	0.89	1.12	2^{32}	4.81E+9	1.93	3693.86	0.23	1.07	2.07	2^{64}	2.21E+19
Our proposed	**0.70**	**685.18**	**0.04**	**0.53**	**0.37**	2^{12}	**1.52E+3**	**0.94**	**1322.02**	**0.08**	**0.70**	**0.66**	2^{22}	**2.76E+6**
RNS	1.23	1582.01	0.13	0.91	1.11	1	1.11E+0	1.99	4576.16	0.42	1.89	3.76	1	3.76E+0
Binary	4.06	4474.82	0.39	0.85	3.45	1	3.45E+0	7.59	11591.53	1.35	1.58	11.99	1	11.99E+0
Our proposed + RC	**1.84**	**983.18**	**0.07**	**0.36**	**0.66**	2^{12}	**2.72E+3**	**2.88**	**1796.48**	**0.16**	**0.57**	**1.65**	2^{22}	**6.88E+6**
RNS + RC	2.42	2131.10	0.22	0.87	2.11	1	2.22E+0	4.47	5612.83	0.62	1.72	7.69	1	7.69E+0

The proposed design delivers the same computation accuracy and the same result as the baseline binary and RNS design.

A. Hardware Cost for Multiplication Operation

For each design approach, we implement 8-, 12-, 16-, and 32-bit multipliers. Table IV shows the CP delay, hardware area, power consumption at 100 MHz and at the maximum working frequency, energy per cycle (EPC = CP $\times$ power consumption at the maximum working frequency), number of clock cycles to complete the computation, and total energy consumption (EPC $\times$ number of clock cycles) of the implemented designs. The proposed design offers, on average, 79% and 49% saving in the hardware area cost and 84% and 57% reduction in the EPC compared to the binary and non-stochastic RNS counterpart, respectively. Considering the fact that the total energy of the SC designs depends on the number of processing cycles, the binary and RNS designs consume lower energy to produce accurate results.

Compared to the clock division deterministic design of [4], the proposed RNS-based design requires less hardware resources to generate bit-streams (counters and comparators are replaced with FSMs and MUXs), and to convert the output bit-streams back to the binary format. For example, for the 16-bit multiplication, the proposed design requires three 6-bit counters in its output while the clock division design needs a 32-bit counter. On average, the proposed design saves the area and power consumption by 63.5% and 32.5%, respectively, compared to the clock division design. The hardware cost of the proposed design with and without RC is reported Table IV. The rows with "+ RC" are the ones that include the RC circuit. As it can be seen, even with RC the primary saving is still observed in the processing time and energy consumption. The proposed design delivers more than 2400$\times$ speedup and 3200$\times$ energy reduction for 8-bit precision multiplication. For 32-bit precision multiplication, the latency and energy consumption reduce by a factor of 9.03×10^{12} and 1.38×10^{13}, respectively.

TABLE V
COMPARISON OF CRITICAL PATH DELAY, AREA, POWER, AND ENERGY CONSUMPTION OF A PE.

Design Approach	CP (ns)	Area (μm^2)	Power @ 100MHz (mW)	Power @ Max Freq. (mW)	EPC (pJ)	# of Cycles	Energy @ Max Freq. (pJ)
Deterministic SC	2.06	1922	0.15	0.62	1.28	2^{32}	5.48E+9
Our proposed	**0.83**	**1339**	**0.14**	**0.18**	**0.15**	2^{12}	**6.12E+2**
RNS	1.82	2802	0.17	0.85	1.54	1	1.54
Binary	5.43	8613	0.49	0.86	4.65	1	4.65

B. Hardware Cost for Neural Network Computations

Due to the increase in the use of neural networks (NNs) in all fields of science, designing NN accelerators is an important task for the researchers in the EDA community. To further evaluate the proposed design we implement the *PE* of *Eyeriss* [6] with different design approaches. We only implement the computational and the logical parts of the PE and do not take scratchpads into account as they are the same in the proposed and binary design. We evaluate the design for 16-bit precision data-width. Table V shows the synthesis results. As it can be seen, the proposed design occupies 84% lower area compared to the binary implementation. It also needs 52% lower area compared to the non-stochastic RNS implementation. The proposed design achieves 2.60×10^{6} and 8.94×10^{6} times reduction in terms of processing time and energy consumption, respectively, compared to the conventional deterministic design of SC.

The product of power consumption, CP delay, and occupied area can be used as a measure of hardware efficiency. Based on this metric, the proposed design is 201$\times$ and 21.5$\times$ more efficient than the conventional binary and RNS design, respectively. It should be noted that for processing time and also total

energy consumption, the conventional binary implementation still outperforms SC-based designs when performing a completely accurate computation. The proposed design is a good fit for embedded applications with severe limitations on the area and power consumption but not very efficient for high-performance and energy-limited applications.

V. Conclusion

In this work, we proposed a novel method to mitigate the long processing time and the high energy consumption of the current deterministic methods of SC. The proposed method significantly reduces the number of processing cycles by employing the RNS representation. Hardware implementation of a two-input multiplier and also a processing element of neural network computation show that the proposed design approach delivers a lower hardware area and power cost compared to the baseline binary, RNS, and clock division design of SC. The proposed RNS-based design is significantly faster and consumes considerably lower energy than the current deterministic designs of SC. These properties make the proposed design an attractive alternative to the current deterministic designs of SC for resource-limited applications.

VI. Acknowledgment

This work was supported in part by the Louisiana Board of Regents Support Fund no. LEQSF(2020-23)-RD-A-26 and National Science Foundation grant no. 2019511.

References

[1] A. Alaghi, W. Qian, and J. P. Hayes, "The promise and challenge of stochastic computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1515–1531, 2018.

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

[3] D. Jenson and M. Riedel, "A deterministic approach to stochastic computation," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2016, pp. 1–8.

[4] M. H. Najafi, D. Jenson, D. J. Lilja, and M. D. Riedel, "Performing stochastic computation deterministically," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2925–2938, 2019.

[5] H. L. Garner, "The residue number system," in *Papers presented at the western joint computer conference*. ACM, 1959, pp. 146–153.

[6] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, 2016.

[7] C.-H. Chang, A. S. Molahosseini, A. A. E. Zarandi, and T. F. Tay, "Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications," *IEEE circuits and systems magazine*, vol. 15, no. 4, pp. 26–44, 2015.

[8] S. Salamat, M. Imani, S. Gupta, and T. Rosing, "Rnsnet: In-memory neural network acceleration using residue number system," in *2018 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 2018, pp. 1–12.

[9] N. Samimi, M. Kamal, A. Afzalli-Kusha, and M. Pedram, "Res-DNN: A Residue Number System-Based DNN Accelerator Unit," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2019.

[10] S. Asadi and M. H. Najafi, "Late Breaking Results: LDFSM: A Low-Cost Bit-Stream Generator for Low-Discrepancy Stochastic Computing," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020.

[11] Y. Wang, "Residue-to-binary converters based on new chinese remainder theorems," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 3, pp. 197–205, 2000.

[12] H. Pettenghi, R. Chaves, and L. Sousa, "Method to design general rns reverse converters for extended moduli sets," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 12, 2013.

Kamyar Givaki (givakik@ut.ac.ir) is a last-year Ph.D. candidate in Computer Engineering at the University of Tehran, Iran. He received his MSc and BSc from the Isfahan University of Technology and K.N.Toosi University of Technology, respectively. His research is focused on application-specific accelerators, computer arithmetic, and approximate computing.

Reza Hojabr (r.hojabr@ut.ac.ir) is a last-year Ph.D. candidate in Computer Engineering at the University of Tehran, Iran. He received his MSc and BSc from the University of Tehran and K.N.Toosi University of Technology, respectively. His research interests include domain-specific accelerators, approximate computing, machine learning, and agile hardware design.

M. H. Gholamrezaei (gholamrezaei@ipm.ir) received his B.S. degree in computer architecture engineering from Shahid Beheshti University, Tehran, Iran, in 2020. He is currently a Research Assistant at the School of Computer Science at the Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. His research interests include computer arithmetic, embedded systems, and design automation.

Ahmad Khonsari (a_khonsari@ut.ac.ir) received his Ph.D. degree in Computer Science from the University of Glasgow, UK, in 2003. He is currently an Associate Professor with the Department of ECE, University of Tehran, Iran, and a researcher at the Institute for Research in Fundamental Sciences (IPM), Iran. His research interests include simulation, data analysis and performance modeling/evaluation of networked and distributed systems.

Saeid Gorgin (gorgin@irost.ir) received his Ph.D. degree in computer system architecture from Shahid Beheshti University, Tehran, Iran in 2010. He is currently an Assistant Professor of Computer Engineering with Iranian Research Organization for Science and Technology (IROST), Tehran. His current research interests include computing systems, computer arithmetic, and VLSI design.

Dara Rahmati (d_rahmati@sbu.ac.ir) received the B.Sc. and M.Sc. degrees in computer engineering from the University of Tehran, Tehran, Iran and the Ph.D. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2012. He is currently an Assistant Professor with the Computer Science and Engineering Department, Shahid Beheshti University, Tehran, Iran. Dr. Rahmati is a member of IEEE.

M. Hassan Najafi (najafi@louisiana.edu) received his Ph.D. degree in Electrical Engineering from the University of Minnesota, Twin Cities, USA, in 2018. He is currently an Assistant Professor with the School of Computing and Informatics, University of Louisiana at Lafayette, LA, USA. His research interests include stochastic and approximate computing, unary processing, in-memory computing, and machine learning.