Towards Energy-Efficient CGRAs via Stochastic Computing

Bo Wang, Rong Zhu, Jiaxing Shang, Dajiang Liu*
College of Computer Science, Chongqing University, Chongqing 400044, China
{wangbocs,zhur,shangjx,liudj}@cqu.edu.cn
*Corresponding author

***Abstract*—Stochastic computing (SC) is a promising computing paradigm for low-power and low-cost applications with the added benefit of high error tolerance. Meanwhile, Coarse-Grained Reconfigurable Architecture (CGRA) is also a promising platform for domain-specific applications for its combination of energy efficiency and flexibility. Intuitively, introducing SC to CGRA would synergistically reinforce the strengths of both paradigms. Accordingly, this paper proposes an SC-based CGRA by replacing the exact multiplication in traditional CGRA with an SC-based multiplication, where the problem of accuracy and latency are both improved using parallel stochastic sequence generators and leading zero shifters. In addition, with the flexible connections among PEs, the high-accuracy operation can be easily achieved by combing neighbor PEs without switching costs like power-gating. Compared to the state-of-the-art approximate computing design of CGRA, our proposed CGRA has 16% more energy reduction and 34% energy efficiency improvement while keeping high configuration flexibility.**

***Index Terms*—Approximate Computing, Stochastic Computing, CGRA, Energy Efficiency, Configuration Flexibility.**

I. Introduction

Coarse-grained reconfigurable architecture (CGRA) is an attractive platform that combines high-energy efficiency and flexible versatility. The architecture view and internal connections of a typical CGRA are shown in Fig.1. It is mainly composed of a host processor, a context memory, a data memory, and a processing element (PE) array interconnected in a 2D mesh pattern. CGRA can promise simultaneous high-performance and high power efficiency. However, to face the dark silicon era, we still need to explore a more efficient CGRA.

To further optimize CGRA, some works attempt to improve the energy efficiency of the PE in CGRA based on dynamic control of voltage and frequency (DVFS). For example, [6] intends to apply the DVFS to PE to improve energy efficiency by selecting optimal frequency/voltage to match the performance requirements. As DFVS introduces negligible switching costs, the flexibility of the CGRA is greatly weakened in this work. Simultaneously, dual-V_{DD} is also a well-known means of reducing power consumption through dynamic conversion between high and low voltages. In [10], dual-V_{DD} is introduced in CGRAs, which assigns high V_{DD} to PE operations with high latency (e.g., multiplications), and low V_{DD} to low latency operations (e.g., additions). This method still introduces the switching overhead of the voltage conversion and also reduces the flexibility of CGRA (more clock cycles to re-configure PE).

This work was supported by the National Natural Science Foundation of China under Grant 61804017.

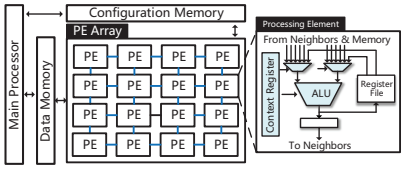

Fig. 1: A typcial CGRA.

Recently, X-CGRA was proposed in [2], where the approximate computing paradigm was introduced into the architecture exploration of CGRA. Since approximate computing can trade energy and computing time with the accuracy of output, the energy efficiency of CGRA could be further improved. To support different operating modes (exact or approximate mode), X-CGRA introduces a quality-scalable arithmetic logic unit (ALU), including approximate and supplementary parts, where power-gating is used to turn on or turn off the supplementary part to achieve mode switching.

Modulo scheduling [4] is a commonly used technique to accelerate loop execution on CGRA, where the initiation interval (II) between adjacent iterations is the key metric. When the data-flow graph (DFG) of a loop is mapped on X-CGRA and runs in pipelined fashion using modulo scheduling, operations with different accuracy requirements may be placed on the same PE at different times step, leading to frequent approximate mode switching of the scalable ALU. Moreover, because a mode switching based on power-gating has additional delay overhead [9], it will inevitably increase the II, leading to performance loss. Therefore, power-gating-based approximate computing is not friendly to modulo scheduling on CGRA.

To tackle the challenge mentioned above, this paper tries to bring stochastic computing (SC) to CGRA. SC is a reemerging computing paradigm [3], which could dramatically reduce the complexity of hardware by converting multiplication (MUL) to simple bit-wise AND operation. However, as typical SC encodes a binary number to a much longer bit-stream by stochastic number generator (SNG), it demands long bit-width operands, which are usually processed serially. Although serial processing of the bit-stream could save many resources, it also brings unacceptable latency for most scenery. So we introduce

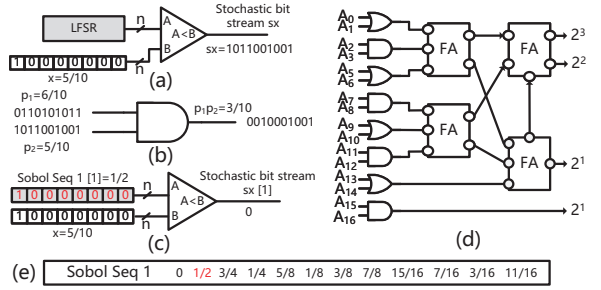

Fig. 2: The basic compositions of SC: (a) stochastic MUL; (b) the SNG based on LFSR; (c) the SNG based on Sobol sequence; (d) approximate parallel counter (APC); (e) two Sobol sequence with length of 12.

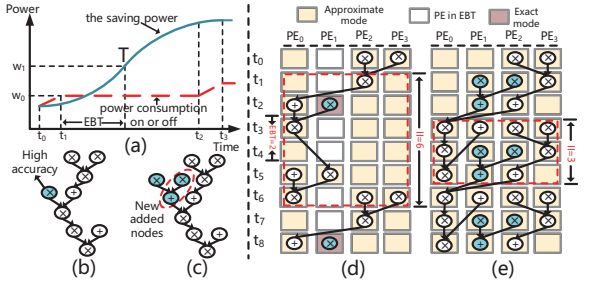

Fig. 3: Limitations of power-gating-based approximate computing: (a) energy consumption curve using power-gating; (b) a DFG example with a high accuracy operator; (c) add some operators to increase accuracy; (d) the mapping of the DFG in (b); (e) the mapping of the DFG in (c).

the parallel processing of the bit-stream in SC to CGRA, which could perfectly match the characteristics of CGRA. The main reason is that CGRA has inherent coarse-grained operands and operations, such that the parallel bit-wise operation could be easily achieved. More importantly, by combining bit-streams from neighbor PEs using the flexible connections among PEs, the accuracy of SC could be flexibly scaled and the control of progressive precision could be easily achieved.

In this paper, we propose an SC-based CGRA (SC-CGRA) by replacing the typical exact MULs with SC-based MULs. SC-CGRA tailors and adapts the SC paradigm to CGRA. It converts exact MUL into simple AND on parallel bit-stream, greatly reducing area and power. In return, CGRA offers a flexible length extension of bit-stream to address the problem of performance loss caused by mode switching. The main contributions of this paper are summarized as follows:

1) We demonstrate an energy-efficient SC-based MUL with deterministic strategy and leading zero shifting such that a single PE could achieve high energy efficiency without much accuracy loss.
2) We propose an accuracy scaling strategy by combing the bit-streams of neighbor PEs to form a longer bitstream such that the output accuracy could be flexibly scaled.
3) We put forward a scalable quality mapping algorithm, which evaluates the error sensitivity of the operator to meet the final accuracy requirements while trying to maintain the pipelining performance.

II. Background And Motivation

A. Stochastic Computing

SC is a reemerging computing paradigm, which can implement complex operations with low power and low-cost circuit [3]. In the SC domain, a mantissa P between [0,1] is serialized into a bit-stream, and the probability of 1 appearing in the bit-stream is just equal to the value of P. For example, as shown in Fig. 2(a), linear feedback shift register (LFSR) will generate a pseudo-random number in each clock cycle in SNG. By comparing x (0.5) with the pseudo-random number, 0 or 1 can be obtained. After 10 clock cycles, a stochastic bit-stream will be generated (1011001001). Because SC uses stochastic bit-stream to encode real values, it can replace MUL with an AND gate. Accordingly, SC can lead to cost-efficient computing circuits. For example, $p_1 \times p_2 = 0.3$ can be easily obtained through AND gate, as shown in Fig. 2(b). After that, we can use a low-cost hardware component called accumulative parallel counter (APC) to perform addition and stochastic-to-binary conversion simultaneously. As shown in Fig. 2(d), if we sent 0010001001 to APC, we will get 3 as a result.

B. LD Sequence Based Stochastic Computing

Some works [7] show that the inherent random fluctuation error of SC leads to the degradation of accuracy and excessive hardware usage. After removing randomness and using the deterministic approach, the accuracy of calculation results can be ensured. Low discrepancy (LD) sequences (e.g., Sobol and Halton) have been recently used in improving the speed of computation on stochastic bit-streams [7]. In LD sequences, 1s and 0s in the bit-streams are uniformly spaced so that the streams do not suffer from random fluctuations. Therefore, generating a bit-stream with LD sequence could achieve comparable accuracy for SC with much less bit-width than that based on LFSR [7]. As one type of LD sequence, Sobol sequence-based SC has the advantage of better energy efficiency [7]. Fig. 2(e) and (c) show one Sobol sequence with a length of 12 and the process of comparing the operand x to the second cell of the seq_1, respectively. Usually, the Sobol sequence generator consumes more hardware resources [7]. However, with a fixed LD sequence pattern [8], the LD sequence could be previously generated and got fixed in hardware implementation. Therefore, using the fixed Sobol sequence to generate stochastic bit-stream would be a good choice in SC implementation in scenarios with high energy efficiency and high accuracy requirements.

C. Motivation

Considering the timing of power-gating, as shown in Fig. 3(a), the solid line is the saving power's curve that debilitates the circuit, and the dotted line indicates the energy consumption of the power gate when it is turned off at t_0 and turned on at t_2 ($t_3 - t_2$ is the additional wake-up latency). Specifically, the power gating could be meaningful when $w_1 - w_0 \geq w_0$ at time T as shown in Fig. 3(a). This time point is defined as energy break-even time (EBT) [9], which has been introduced to quantify the trade-off between energy saving and penalty (EBT can reach about 3.7ns in 45nm technology [9]).

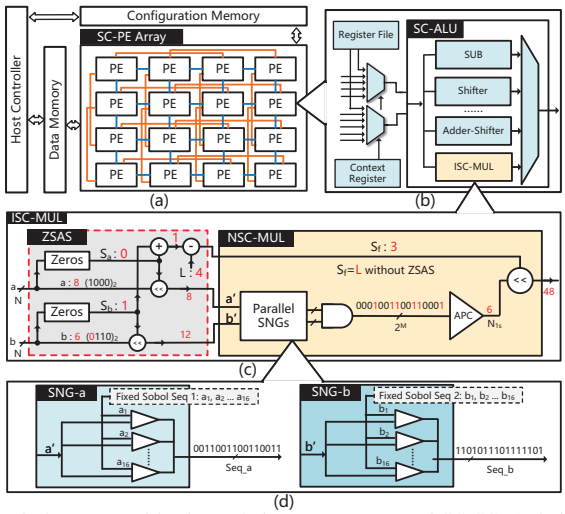

Fig. 4: Our proposed hardware design: (a) our proposed SC-CGRA design; (b) our SC-ALU design; (c) our SC-based MUL design; (d) our parallel SNG based on Sobol sequence.

In order to show the impact of power-gating technology on software pipelining performance clearly, we construct a DFG mapping example in Fig. 3 (assuming the clock period is 2ns). In Fig. 3(b), as the blue operator in the DFG is crucial to the output quality of the whole DFG, it should be in exact mode while other operators are in approximate mode. When mapping this DFG onto X-CGRA, the operators with different accuracy requirements will be mapped on the same PE. As shown in Fig. 3(d), PE_1 needs to do mode switch continuously. Because of the EBT (at least 2 clock cycles under 45nm technology [9]), the additional latency will make the II=6, which will reduce the pipelining performance. If SC is applied to each PE, the accuracy of the critical operation could be easily scaled by adding new operations (i.e., increasing the length of the stochastic bitstream). When mapping the transformed DFG onto SC-CGRA, we could finally achieve II of 3 (Fig. 3(e)) as there is no power-gating-based mode switching involved.

III. Hardware Design

Fig. 4(a) shows our overview of a 4×4 SC-CGRA, which is mainly composed of a host controller, a data memory, a configuration memory, and a dynamic reconfigurable PE array that has scalable quality. SC-CGRA supports both spatial and temporal mapping and uses 2D mesh plus [4] interconnection mode. As shown in Fig. 4(b), the ALU in SC-CGRA is distinguished from that in traditional CGRA from two aspects: 1) the exact MUL is replaced with our improved SC-based MUL (ISC-MUL). 2) the adder is replaced with an adder-shifter such that the sum of adder could be shifted to the right by 0 (for normal adder) or 1 bit (for accuracy scaling), determined by configuration contexts.

A. Naive SC-Based MUL Design

As the yellow region shown in Fig. 4(c), the naive SC-based MUL (NSC-MUL) consists of two parallels SNGs, a multi-bit AND, an APC, and a left shifter. To achieve parallel SNG, we introduce the Sobol sequence to implement the SNG. To further reduce the resource consumption, we also use a fixed LD sequence pattern [8], such that the Sobol sequence in each PE is fixed in hardware implementation. Therefore, only with wires connected to power or ground, no extra storage resource is needed for the Sobol Sequence. For further quality scaling, each PE holds an independent Sobol sequence with different cell values. Besides, we used MRED in [2] as our error metric to measure the accuracy of 32-bit ISC-MUL, as shown in Table I. It is easy to see that Sobol sequences produced by different seeds have little effect on the accuracy. Consequently, the fixed Sobol sequence in MUL implementation is acceptable.

TABLE I: The 32-bit ISC-MUL's accuracy with different Sobol sequences.

Seed	(12.1, 13.6)	(15.6, 17.4)	(23.6, 19.8)	(36.8, 21.4)
Error(%)	4.3	5.8	4.9	5.2
Seed	(45.96, 23.6)	(21.16, 34.7)	(30.78, 34.8)	(35.96, 23.9)
Error(%)	5.1	4.7	4.8	5.3

As shown in Fig. 4(d), in SNG-a, operand a' is compared to $2^M = 16$ cells of Sobol sequence_1 in parallel and a bit-stream with the length of 16 is quickly generated. Similarly, SNG-b takes operand b' and generates another bit-stream simultaneously. Taking the two 16-bit streams as input, the AND gate followed by an APC could finally count the number 1s (N_{1s}) in the output bit-stream. Therefore, the output of the MUL is $N_{1s}/2^M$. As shown in Fig. 4(c), there are six 1s in the output bit-stream, and the output result is 6/16.

It is worth noting that we directly send N-bit operands in integer format to SNGs while SNGs are supposed to receive mantissa. As the integer value of an N-bit binary number is 2^N times the mantissa value represented by the same binary number, it is equivalent to dividing the integer operand by 2^N if it is directly sent to an SNG. Therefore, the output result ($N_{1s}/2^M$) is actually equivalent to $a'/2^N \times b'/2^N$. As a result, the MUL result of a' and b' could be represented as follows:

$$a' \times b' = N_{1s} \times 2^{2N-M} \quad (1)$$

Therefore, an extra left shifter is added to the APC to get the right MUL value of operand a' and b', and the number of bits that need to be shifted is named L, equaling to $2N - M$. As shown in Fig. 4(c), as N and M are both 4, we need to shift the APC result by 4 bits. In addition, NSC-MUL in Fig. 4(c) omits the processing of the sign bits. To support singed MUL, an XOR gate, taking the sign bit of the two operands as input to generate the output sign bit.

B. Improved SC-based MUL using Leading Zero Shifter

When SC is applied to CGRA, it commonly suffers from low accuracy. Specifically, for SNG based on the Sobol sequence, when the input of SNG is a small mantissa, the input will be ignored after the comparison operation. For example, when a small mantissa such as $n = (0010)_2$ is compared to a Sobol sequence ($\frac{1}{8}(0010)_2, \frac{5}{8}(1010)_2, \frac{3}{8}(0110)_2, \frac{7}{8}(1110)_2$), it will generate an all-zero bit-stream and further limit the application of SC in CGRA. In order to improve the accuracy of a MUL, we design a leading zero shifting module (the gray region shown in Fig. 4(c)). This module includes two leading zeros counters, two left shifters, a 4-bit adder, and a 4-bit subtractor (4 bits to count the number of 0s between

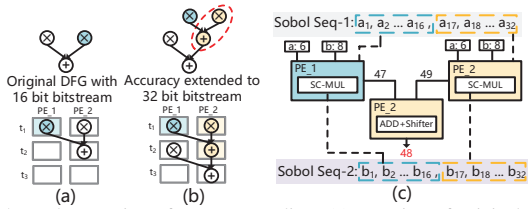

Fig. 5: Implementation of accuracy scaling: (a) mapping of original DFG; (b) mapping of the new DFG with added nodes to increase the accuracy; (c) implementation of accuracy scaling by combing neighbor PEs.

$[0, 15]$ for 16-bit operands). We call this module ZSAS (Zeros & Shifter & Adder & Subtractor), and it works as follows:

At first, the number of leading zeros of the input operands (a, b) is obtained by leading zeros counters (S_a for a and S_b for b), and then the operands are shifted by S_a and S_b bits to the left respectively. In this way, some small operands can be temporarily amplified, so that they will not be ignored when passing through the comparators of SNG. Afterward, S_a and S_b are sent into the adder to get the total number of bits shifted to the left in advance. To get the right MUL result, the output of MUL is supposed to be shifted to the right by $S_a + S_b$ bits. We also note that there is a left shifter in the NSC-MUL that shifts the result to the left by L bits. Consequently, a subtractor is used in ZSAS to get the actual number of bits that need to be shifted to the left, $S_f = L - (S_a + S_b)$.

Fig.4 (c) also shows the overall process of our ISC-MUL: firstly, the 4-bit input operands ($a = 8, b = 6$) will be shifted to the left through ZSAS. Since $a = 8(1000)_2$ and $b = 6(0110)_2$ have 0 and 1 leading zeros ($S_a = 0$ and $S_b = 1$), and then a and b are amplified to 8 and 12, respectively. Meanwhile, $S_f = L-(S_a+S_b) = 3$ can be easily calculated in ZSAS. And secondly, we use SNGs to quickly convert a' and b' into two 16-bit stochastic bit-streams (M=4). After the two bit-streams are processed by AND and APC, the number of 1s is obtained ($N_{1s} = 6$). Finally, by shifting N_{1s} to the left by $S_f = 4 - 1$ (3), the output (48) of MUL is obtained.

As shown in Table II, 65536 16-bit random inputs are used to evaluate the accuracy of NSC-MUL and ISC-MUL. It is easy to see that the NSC-MUL has a large error and is difficult to be directly applied to CGRA, and ISC-MUL performs well and can meet the accuracy of general applications. As the bit-stream length of 128 would reduce the error rate to 2% while longer bit-streams have few accuracy gains, we set the longest bit-stream in SC-CGRA to 128 bits, including concatenated bit-stream from multiple PEs.

C. Quality Scaling By Combing Neighbor PEs

Although the accuracy of ISC-MUL is greatly improved, it is still limited by the length of the parallel fixed Sobol sequence in each PE, which would not be too long in real implementation. To further improve the accuracy of the MUL operation, we could concatenate the fixed Sobol sequences in different PEs to form a longer Sobol sequence. Since SC-MUL results are determined by counting the number of 1s in the output of the AND gate, we just need to add SC-MULs output results with several adders and shift the sum of each adder to the right by 1 bit. As shown in Fig. 5, in the original DFG, to increase the accuracy of the blue MUL operator, an extra MUL operator, and an adding-shifting operator are added (shown in Fig. 5(b)), where the two MUL operators take the same operands ($a = 6, b = 8$) as inputs. As shown in Fig. 4(c), by executing the two MUL operators on PE_1 and PE_2, 47 and 49 are obtained as results. Then, taking the two results as inputs, the Adder-Shifter gets a high-accuracy output (48).

TABLE II: The accuracy performance of the NSC-MUL and ISC-MUL.

Bit-stream length	8	16	32	64	128	256
Error(NSC-MUL)	62%	60%	56%	56%	40%	35%
Error(ISC-MUL)	24%	12%	5%	3.3%	2%	1.5%

Besides, we also note that as each added MUL operator takes the same operands, the out-degree of the operands is increased. Therefore, mesh-plus [4] connection topology is adopted to support these high-degree operators.

IV. Mapping Algorithm

As the quality scaling with PE combination would cost more resources, we need to carefully determine the accuracy level of each operator in a DFG to satisfy the output quality requirement. Therefore, we formulate the accuracy level assignment using integer linear programming (ILP), which has formal expression and is helpful to find the optimal solution.

A. Establishment of Constraints

Decision variable definition: Since different operators have different effects on the output quality of the overall DFG [2]. We will define K binary decision variables $x_i^k \in \{0, 1\}$ for each MUL operator i in the DFG, where $k \in \{1, K\}$ represents the k^{th} accuracy level. K is an empirical parameter, which specifically represents how many basic stochastic bit-streams would be concatenated to form a longer bit-stream. The larger the K is, the higher accuracy the operator will achieve.

C1: The accuracy level of each MUL is unique: Each MUL operator can only be of one accuracy level. Therefore, the uniqueness constraint on the decision variable of the MUL operator is constructed as follows:

$$\mathbf{C1}: \sum_{k=1}^{K} x_i^k = 1, \forall i \in [1, \mathrm{G}] \tag{2}$$

where G is the total number of MUL operators in the DFG.

C2: The accuracy constraint of the whole DFG: For a given accuracy requirement, the output quality degradation should not be greater than a given upper bound ($Q_{D_{UB}}$). As the work in [2], we also formulate the output quality degradation according to the error distance (ED) and error sensitivity (ES). ED is represented as $|O - O^k|$, where O and O^k indicate the exact output and the approximate output in level k, respectively. Based on ED, ES is defined as follows:

$$ES_i^k = \frac{ED_{i,o}^k}{ED_i^k} \tag{3}$$

where $ED_{i,o}^k$ and ED_i^k respectively indicate the output ED of the whole DFG and the output ED of the operator i, when the operator i is an MUL operator with the accuracy level of k while the other operators are in exact mode. As mentioned

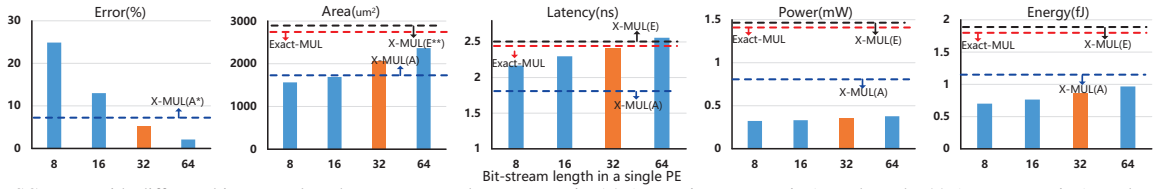

Fig. 6: ISC-MUL with different bit-stream lengths are compared to X-MUL in A* (approximate operating) mode and E** (exact operating) mode [1], and exact MUL in accuracy, area, latency, power, and energy.

in [2], the error variance of the DFG (v_o) can be calculated as:

$$v_o = \sum_{i=1}^{G}\sum_{k=1}^{K}(ES_i^k)^2 \cdot v_i \cdot x_i \tag{4}$$

where v_i is the output error variance of the i^{th} node.

Finally, as the work in [2], the error variance should satisfy the following constraint:

$$\mathbf{C2}: v_o \leq \frac{Q^2_{D_{\mathrm{UB}}}}{N-1}\sum_{n=1}^{N} O_n^2 \tag{5}$$

where N and O_n indicate the total number of samples and the exact output of the n^{th} DFG sample, respectively.

B. Establishment of Objective Function

Because our accuracy scaling strategy needs to expand the original DFG, which may increase the resource minimal II, and reduce the performance of software pipelining. Therefore, we need to minimize the total number of added operators while satisfying constraints C1 and C2.

Given a MUL operator i, if we want to increase its accuracy level to $k_i \in \{1, K\}$, we need to add $k_i - 1$ extra MUL operators for the operator i to increase accuracy, and add $k_i - 1$ adding-shifting operators to get the final result. To sum up, the objective function of our ILP model will represent the minimum of numbers of newly added operators in DFG:

$$min\sum_{i=1}^{G} 2(k_i - 1) = min\sum_{i=1}^{G} 2\left(\sum_{j=1}^{K} j \cdot x_i^j - 1\right) \tag{6}$$

From the formulation above, we note that as MUL operators are only a part of the whole DFG and the maximal accuracy level K is rare small ($K = 4$ will achieve 98% accuracy if a single PE includes 32-bit stochastic streams), the number of decision variables would not be too large, and therefore the ILP problem is of low complexity in most cases.

C. Overall Implementation Flow

The overall implementation flow of our mapping algorithm can be summarized as follows: firstly, we will define binary decision variables, accuracy constraints, and objective function to establish the optimization problem. And then the sophisticated ILP solver [5] is used to solve the problem. At last, according to the operator accuracy allocation scheme, a new DFG is generated by adding extra operators to the original DFG.

After the expanded DFG is obtained, it will be mapped on CGRA as a normal DFG using modulo scheduling techniques [4]. At first, we schedule the DFG using the list scheduling method [2]. Then, we use the max-clique-based method [2] to perform place-and-route and further obtain a valid mapping result. As there is no EBT overhead from power-grating, the expanded DFG could be fully pipelined in a temporal-mapping fashion.

V. Result And Discussion

In this section, we will compare our design with some baselines in terms of hardware and accuracy performance. The process of getting hardware evaluation is as follows: firstly, we will use Verilog HDL to design our hardware; secondly, based on 45-nm CMOS process library, we use Synopsys Design Compiler to synthesize the verified Verilog code. Finally, Prime Time of Synopsys is used to get energy consumption.

A. Explore the Length of Bit-stream in SC-CGRA

The bit-stream length in each ISC-MUL will affect both resource consumption and accuracy performance. To fully explore the performance of different bit-stream lengths, the accuracy, area, latency, power, and energy of bit-stream with 4 types of length-based ISC-MULs are measured in Fig. 6 (65536 random numbers as inputs to measure accuracy using MRED in [2]). Meanwhile, we also compare with the exact MUL (Exact-MUL) and the Dadda multiplier [1] (X-MUL) in A (approximate operating) mode and E (exact operating) mode [1], which are marked with red, blue, and black dotted lines in the Fig. 6 , respectively (the error rate of the Exact-MUL is 0 and the operands of ISC-MUL and all baselines are 16 bits).

It is easy to see from Fig. 6 that the ISC-MULs with 4 types of length have lower power and energy than Exact-MUL and X-MUL. But ISC-MULs with 8-bit and 16-bit have lower accuracy, which is even lower than X-MUL in A mode. Although the 64-bit ISC-MUL has high accuracy, its latency is higher than Exact-MUL. And the circuit area of 64-bit ISC-MUL is much larger than X-MUL in A mode. In summary, the 32-bit ISC-MUL is a compromise choice in error rate, area, latency, and power. It is worth noting that X-MUL in A mode has the smallest latency. But in E mode, it has a higher latency than the Exact-MUL. And in practical applications, to meet higher accuracy requirements, there will be X-MULs working in A mode and E mode simultaneously, which leads to the critical path depending on the latency in E mode.

B. SC-CGRA Versus Typical Exact CGRA And X-CGRA

As shown in Table III, the measured parameters for architecture evaluation include delay (critical path latency), circuit area, power, and energy. It is easy to see that X-CGRA has maximum circuit area because of the extra power-gating circuit. On the contrary, our SC-PEs greatly simplify the circuit

TABLE III: Hardware performance of SC-CGRA and X-CGRA.

*QSPEs : Qulity-scalable PEs in [2], **E : Exact mode, ***A : Approximate mode.

Design Parameters	Arithmetic Units State		Chain Latency(ps)	Area (um^2)	Power (mW)	Energy (fJ)
PE Types			Target Module			
	Add	Mul.	4×4 CGRA			
Typ. PE			12895	249790.63	71.17	528274.21
QSPEs	E	E	13354	280783.67	72.61	564474.06
	A	E	10413		63.81	405025.89
	E	A	8744		58.12	293230.26
	A	A	7134		54.82	226786.36
SC-PEs	-		12783	193471.06	43.04	318659.81

so that our design has the smallest hardware area. Specifically, compared with the typical exact CGRA and X-CGRA, the area of SC-CGRA can decrease by 23% and 38%, respectively.

Besides, SC-CGRA can reduce power consumption by about 46% compared with traditional exact CGRA. Moreover, compared with the four modes of X-CGRA, the power of SC-CGRA is reduced by 67%, 50%, 32%, and 14%, respectively. In terms of energy consumption, SC-CGRA is 65% lower than traditional CGRA and performs better than X-CGRA in exact ADD and exact MUL mode (E, E) and approximate ADD and exact MUL (A, E) mode, reducing 77% and 27% respectively. X-CGRA is better than SC-CGRA in (E, A) and (A, A) modes, mainly due to the reduction of delay in these two modes. However, once one operation needs to be in (E, E) mode in X-CGRA, the delay of all operations will be the longest one when calculating the energy. Consequently, there will be less energy benefit for X-CGRA in actual cases.

C. The Efficacy of SC-CGRA

To demonstrate the efficiency of SC-CGRA in different applications, we use benchmarks similar to [2], including 64^{th} polynomial evaluation (POE), and 8- and 64-Tap filters, 2×2 and 4×4 matrix-matrix multiplication (MMM). Further, to measure the performance of software pipelining, we also use MMM of 10×10 and 12×12 (the II will be 2 to meet the 4×4 resource constraints). To get accuracy evaluation, we use an accuracy evaluator with 65536 random inputs. And the error rate is calculated using the output quality. Besides, we use the cost function (CF) proposed in [2] to measure the energy efficiency:

$$\mathrm{CF} = \frac{\text{Energy} \times \text{Delay} \times \text{Area}}{1 - Q_{\text{loss}}} \tag{7}$$

where Q_{loss} and delay indicate the output quality and chain latency, respectively. From Equation (7), we note that the lower the CF value is, the higher the energy efficiency of the architecture is.

Fig. 7 shows the energy reduction and CF reduction with different output quality requirements for the 6 different kernels. The x-axis indicates the upper bound of output quality degradation. The left y-axis and right y-axis indicate the relative reduction (energy and CF) and the actual output quality degradation. As the curves shown in Fig. 7, under the same upper bound constraint, SC-CGRA can obtain a smaller actual output quality degradation. In addition, the ISC-MUL can implement a quality scalable MUL operation with variable-length stochastic bit-stream, including 32, 64, 96, and 128 by combing neighbor PEs. As a result, SC-CGRA has more fine-grained control on accuracy compared to X-CGRA.

At the same time, as can be seen from the bar chart in Fig. 7, because SC-CGRA uses ISC-MUL to greatly reduce

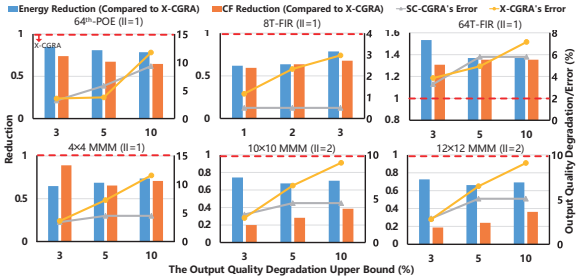

Fig. 7: The SC-CGRA's energy reduction and CF reduction compared to X-CGRA and the output quality of the applications implemented on SC-CGRA and X-CGRA under different output quality degradation upper bounds.

the circuit complexity, the energy efficiency of SC-CGRA is much better than that of X-CGRA. Especially in the cases of II>1, because SC-CGRA can do low-cost accuracy mode switching in the temporal-mapping fashion (only adds some extra operators to the original DFG). But X-CGRA needs to use power-gating technology to switch the accuracy mode. Due to the existence of EBT, every power-gating step will have some additional delay, which leads to its poor performance. On average, SC-CGRA has 16% more energy reduction and 34% CF improvement than those of X-CGRA, respectively.

VI. Summary

In this paper, we propose an energy-efficient CGRA based on SC (SC-CGRA). In addition to having low energy consumption and low hardware overhead, SC-CGRA also supports high-flexible temporal mapping. With the flexible connections among PEs, the high-accuracy operation could be easily implemented by combing neighbor PEs without switching costs like power-gating. Compared to the state-of-art approximate computing design of CGRA, SC-CGRA has 16% more energy reduction and 34% energy efficiency improvement.

References

[1] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram. Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(4):1352–1361, 2017.

[2] O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, and M. Shafique. X-cgra: An energy-efficient approximate coarse-grained reconfigurable architecture. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2558–2571, 2020.

[3] M. Alawad and M. Lin. Survey of stochastic-based computation paradigms. *IEEE Transactions on Emerging Topics in Computing*, 7(1):98–114, 2019.

[4] M. Bingfeng, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins. Exploiting loop-level parallelism on coarse-grained reconfigurable architectures using modulo scheduling. In *2003 Design, Automation and Test in Europe Conference and Exhibition*, pages 296–301, 2003.

[5] P. Feautrier. Parametric integer programming. *RAIRO Recherche Opérationnelle*, 22(3):243–268, 1988.

[6] S. Jafri, O. Bag, A. Hemani, N. Farahini, K. Paul, J. Plosila, and H. Tenhunen. Energy-aware coarse-grained reconfigurable architectures using dynamically reconfigurable isolation cells. In *International Symposium on Quality Electronic Design (ISQED)*, pages 104–111, 2013.

[7] S. Liu and J. Han. Toward energy-efficient stochastic circuits using parallel sobol sequences. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(7):1326–1339, 2018.

[8] H. Sim and J. Lee. A new stochastic computing multiplier with application to deep convolutional neural networks. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2017.

[9] K. Usami and N. Ohkubo. A design approach for fine-grained run-time power gating using locally extracted sleep signals. In *2006 International Conference on Computer Design*, pages 155–161, 2006.

[10] S. Yin, J. Gu, D. Liu, L. Liu, and S. Wei. Joint modulo scheduling and v_{dd} assignment for loop mapping on dual- v_{dd} cgras. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(9):1475–1488, 2016.