GREEN: An Approximate SIMD/MIMD CGRA for Energy-Efficient ProcessiNg at the Edge

Zahra Ebrahimi, Akash Kumar, *Senior Member, IEEE*

***Abstract*—The rapid evolution of compute-intensive programs from bio-signal to image-, and video-processing has motivated moving toward *Coarse Grained Reconfigurable Architectures* (CGRAs), having high parallelism capability with post-fabrication datapath versatility. To enhance energy-efficiency of such error-resilient applications, State-of-the-Art (SoA) CGRAs exploit approximation techniques, while maintaining an acceptable accuracy for the final Quality of Result (QoR). However, such CGRAs suffer from overheads of utilizing separate Add/Mul/Div units. We propose *GREEN* as an energy-efficient CGRA, which enables synergistic effects of a chain of approximation and optimization techniques in various levels of abstraction, from application-, to architecture-, to circuit-level, in a cross-layer hierarchy. Enabling this, GREEN offers different levels of energy-accuracy trade-offs through the flexibility of its small Processing Elements (PEs), each of which can support various functionalities and precision-adaptability in a Single Instruction, Multiple Data (SIMD) or Multiple Instruction, Multiple Data (MIMD) manner.**

Experimental results obtained with Synopsys Design Compiler and Cadence Innovus at 45 nm CMOS technology node demonstrate the efficiency of the proposed SISD/SIMD/MIMD CGRA over the accurate and SoA counterparts. In particular, the MIMD mode of GREEN enables up to 6.6× higher throughput while dissipating 21% less energy than the accurate counterpart. Moreover, the end-to-end evaluation of GREEN variants on eight single- and multi-kernel applications from classification, bio-signal (ECG/EEG), and image/video processing domains demonstrates significant performance improvement, compared to the accurate CGRA. In particular, GREEN-MIMD not only speeds-ups the ECG QRS detection by 49% and consumes 43% less area and 66% less energy than the accurate CGRA, but also maintains the heartbeat detection accuracy at 100%. GREEN implementations is available at *https://cfaed.tu-dresden.de/pd-downloads*.

***Index Terms*—SIMD/MIMD, Approximate Computing, CGRA, Bio-signal, ECG/EEG, Classification, Unmanned Air Vehicles, High Throughput, Energy-Efficiency, Edge Computing.**

I. Introduction

Coarse-Grained Reconfigurable Arrays (CGRAs) strike a balance between flexibility and energy-efficiency (see Fig. 1) through word-level reconfigurability, making them viable acceleration platforms for executing a broad range of tasks at the edge. CGRAs are already commercialized in Samsung Galaxy [1], Intel datacenter processors [2, 3], IBM's RaPiD which is an approximate CGRA for the acceleration of Artificial Intelligence (AI) tasks [4], Wave DPU (a data-flow processing unit for deep neural networks) [5] or persuaded the major Field Programmable Gate Array (FPGA) vendors to coarsen the granularity of their chips [6], [7]. Endowed to their post fabrication data-path versatility, CGRAs have also recently gained momentum for bio-signal processing (commercialized in e.g., Samsung Galaxy smartphones/smartwatches [1, 8]), where adaptations with patient's physiology is of high priority

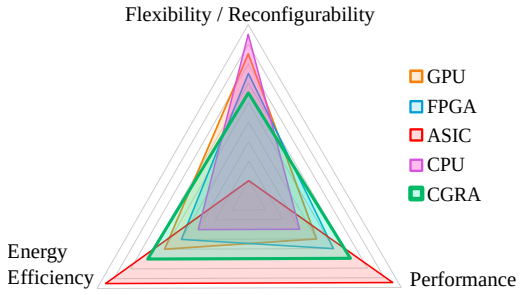

Fig. 1: Flexibility, performance, and energy-efficiency comparison between CGRA and other architectures [19]

[9, 10]. Proliferating use-cases of such Internet of Things (IoT) nodes are 24/7 wearable health monitoring gadgets, as 47% of cardiac diseases – the major cause of death globally – occurs outside the hospitals [11]. In fact, wearable devices are deemed as one of the fastest-growing industries and their instances are projected to surpass one billion [12] and their market value is expected to grow triple and worth over $54 billion by 2023 [13]. Moreover, the real time video streaming and image processing are ubiquitous in various IoT services. In fact, Cisco has reported that the share of video processing traffic in the global internet traffic has already passed 82% and is projected to increase in upcoming years [14, 15] and continues to grow in the upcoming years [16, 17]. Therefore, to push the real-time processing of such compute-intensive workloads to severely resource-constrained 24/7 portable gadgets, higher throughput should be achieved in stringent power-budget [18].

Beside benefiting from the architecture-level merits of CGRAs, coping with the severe energy constraints of edge nodes also entails application-level techniques that can reduce the load of computations while satisfying an acceptable Quality of Service (QoS). In this context, SoAs have pronounced the error-resiliency potential of a wide range of applications, from image/video processing to classification and bio-signal processing. In particular, cutting-edge studies have demonstrated that 1) *processing* stage dissipates up to 70% of total energy in wearable nodes [20]. 2) Bio-signals are permeated with noise and redundancy, and their processing algorithms exhibit high parallelism and approximation-amenability [21]. Hence, such computations can benefit greatly from an approximate CGRA, that is adjustable with patient's changing condition/activity/physiology or application upgradability (which usually outpace hardware updates). Nevertheless, few works [21, 22] have shown notable gains by approximating kernels of an ECG program. In fact, the implementation of these works are either fully-customized, through an ASIC approach

or fully-reconfigurable, at the bit-level granularity of FPGA.

To exploit the inherent error-resiliency of image, and video-processing applications, several approximation techniques have emerged which are further deployed on CGRAs [23, 24]. However, these techniques suffer from two main shortcomings: 1) *CGRA-specific*: inevitable routing-power and die-area overhead, which stem from collecting accuracy-configurable or separate adder/multiplier/divider units, with an inter- or intra-PE heterogeneity [23, 25, 24]. In fact, studies like [26, 27] and our analysis (will be be detailed later on, in Fig. 2), demonstrate the longer latency and higher energy of division operation compared to the multiplication, which can confine the operating frequency of PEs. 2) *General*: ignoring the fact that application-level optimizations will be poorly reflected or even throttled in an non-optimized hardware (e.g., under-utilization of FPGA DSPs in precision scaling to low bit-width). Particularly, the latter has raised the quest for *Cross-Layer Approximation*, the goal of which is to unlock the potential for approximation efficiently, across layers of abstraction [28]. Although valuable studies ([29] and its references) have attempted to enable this cohesive, cross-layer hierarchy by leveraging circuit-level imprecise adder and/or multiplier in tandem with application-level precision scaling, their evaluations have been mainly limited to neural networks. Moreover, such approaches have left architecture-level opportunities untouched.

The concept of SIMD has been recently exploited by e.g., Xilinx [30], Intel [31, 32], and IBM [4, 33], to 1) provide support for precision-variability *in a single unit*, 2) harness the data-level parallelism capability of applications and 3) reduce the share of routing/control over logic. Such efforts, however, are hitherto restricted to addition/multiplication with a common operand, or multiplier/divider [27, 22], customized for an FPGA implementation. This highlights the demand for an approximate SIMD ALU, that can also be utilized for a variety of applications. Moreover, while addition and multiplication are frequent functions in the above-mentioned error-resilient workloads, prior works [23, 27, 26] and our evaluations (detailed in Fig. 2) show that long latency and high energy of division (2-8$\times$ of the same bit-width multiplication), not only can confine the application speed, but also consume considerable portion of ALU area/energy. These hurdles have hitherto prevented offloading of division included kernels to most of CGRA accelerators and relieving host processor from context-switching on such occasional interrupts (which also exacerbates processor/accelerator communication traffic).

To surmount the foregoing challenges, *BioCare* [34] has been designed, as an area-, performance-, and energy-efficient CGRA, evaluated on biomedical workloads. BioCare supports multiple-functionality and precision-adaptability through the flexibility of its PEs; each can perform addition, multiplication, or division, on two data precisions (8- and 16-bit). Particularly, featuring SIMD at intra- rather than conventional inter-PE granularity (e.g., adopted in Morphosys architecture [35]) not only yields higher throughput per area by mitigating the share of global routing, but also provides accuracy-configurability by switching precision at runtime. Such a light-weight SIMD architecture can perfectly fit for multiplication-exhaustive applications, while allowing better utilization of Processing Elements (PEs) when division is also required.

In this article, we propose *GREEN*, which expands the function-versatility and precision-adaptability of BioCare. In fact, GREEN enables cross-layer approximation through the flexibility of its PEs; each can concurrently perform addition, multiplication, *and* division, on different data parts (4, 8-, or 16-bit data precision). This conjoins instruction- and data-level parallelism (ILP/DLP) into a light-weight realization of MIMD [36]. It should be noted that all CGRAs belong to MIMD model, according to Flynn's taxonomy [19]. GREEN has even taken a step further & enabled multi-functionality with varied precision within *each* PE. In fact, while SIMD mode can perfectly fit for acceleration of similar operations with varied precision e.g., convolutions, MIMD allows better utilization of PEs when simultaneous computations of varied operations is of interest [36]. Prevalent examples that can benefit from this light-weight SIMD/MIMD architecture are Neural Networks (NN), image, video, and bio-signal processing applications, as all have division along with addition and multiplication operations, in different precision. The need of such architecture has also been cited in recent industry-academia collaborations for emerging frontiers of AI acceleration [37]. In short, we make the following novel technical research contributions:

- Expanding the function-versatility and precision-adaptability of SIMD PEs in BioCare (16/8 bit) to also support MIMD mode in GREEN for 16/8/4-bit precision.
- Expanding the architecture by adding the global memory and also measuring the reconfiguration time and energy for the end-to-end evaluation of multi-kernel applications on the proposed CGRA.
- Applying an application-level sensitivity analysis to gauge the error-resiliency of kernels to inexact multiplication/division and precision scaling. This is followed by applying a greedy heuristic to tune the precision of kernels in multi-kernel applications.
- Expanding the evaluated applications to eight single- and multi-kernel case studies from different domains: classification, bio-signal (ECG/EEG), image-, and video-processing.

We open-source the implementations for GREEN SISD/SIMD/MIMD CGRA at *https://cfaed.tu-dresden.de/pd-downloads*, to springboard future research for reconfigurable and approximate computing communities.

II. Related Work

Herein, we present a brief survey of accurate/approximate SIMD/MIMD CGRAs and pinpoint SoAs in Table I.

State-of-the-art energy-efficient CGRAs: many domain-specific CGRAs have targeted energy-efficiency, which are designed by specialization of their architecture for a range of applications. RAP [41] and BrainWave [42, 43] are customized for the acceleration of bio-signal processing kernels by extending/modifying the architecture of Blocks SIMD CGRA [44]. While Blocks has a homogeneous structure, RAP includes heterogeneous PEs, each supporting 24-bit fixed-point accurate multiplication, division, and CORDIC operations. ASAP [40] enables precision-variability for 16- and 32-bit fixed- and floating-point (FP) numbers, only by discarding/zero-padding

TABLE I: Summary of SoA studies in the literature from the perspectives of approximation and parallelization

SIMD/[1] MIMD	Approximate Add/Mul/Div	Optimization/ Approx. Layer	Description of Work	Platform	Performance Improvement	Accuracy
✗/ ✗	✓/ ✓/ ✓	Circuit	Power-gate configurable Add/Mul [25, 24, 38] & Div [23] in PEs	CGRA	{Delay, Energy} +	PSNR 26, SSIM 0.9
✗/ ✗	✓/ ✓/ ✗	Circ. / Appl.	Quantization with inexact Muls in NN [29] and its references)	ASIC	Energy ++	Classific. loss $<10\%$
✗/ ✗	✓/ ✓/ ✗	Circ. / Appl.	Inexact Add/Mul in ECG analysis (fixed precision kernels) [21]	ASIC	{Area, Energy} ++	PSNR 11, QRS 100
✗/ ✗	✗/ ✓/ ✗	Circuit	Converting Mul to serial bit-wise AND operations via SC [39]	CGRA	{Area, Energy} +	QoR loss $<10\%$
✗/ ✗	✓/ ✓/ ✗	Circ. / Appl.	Prec. variability in fixed/FP by zero-padding mantissa bits [40]	CGRA	$\{\frac{\text{Perf.}}{\text{Watt}}, \frac{\text{Perf.}}{\text{Area}}\}$++	QoR loss $<5\%$
✗/ ✓	Accurate	✗	Process bio-signals by multi-datapath PEs (ALU+Reg+Mux) [9, 10]	CGRA	{Energy, Tput} +	-
✓/ ✗	✗/ ✓/ ✓	Circ. / Arch.	First hybrid Mul/Div (LUT-based, customized for FPGAs) [27]	FPGA	{Energy, Tput} +	ARE 0.8%, PSNR 45
✓/ ✗	Accurate	✗	Combine 2 Muls with a common operand in FPGA DSPs [31, 30]	ASIC	{Delay, Energy} +	-
✓/✓	✗/ ✓/ ✓	Circ. / Arch. /Application	A SISD/SIMD/MIMD CGRA architecture that supports a chain of approximations (precision scaling on top of inexact multiplication and division operations), in a cross-layer hierarchy	CGRA	{Area, Delay}+ {Energy, Tput} ++	PSNR 28, QRS 100, Vector-Detect. 90%, Root Mean Square Error (RMSE) 10%

[1] Ability to support sub-word parallelism: same instruction on different data (SIMD) and different instructions on different data (MIMD)

certain mantissa bits. Beside customized PEs, other domain-specific customization in these CGRAs is applied by e.g., refining the network topology (adding diagonal links) and deciding the size of the data memory.

SIMD/MIMD CGRAs: Targeting a high accuracy, computations can be carried out in 32- or 16-bit integer, in both bio-signal and image processing applications [45]. Nevertheless, bounding the precision to 16- or 8-bit is reported to be satisfactory in SoA studies [46, 26]. HEAL-WEAR [10], i-DSPs [9], Blocks [44], and PRECISION [47] have been employed for *accurate* computations while TRANSPIRE [48] supports precision conversion among FP numbers. RaPiD is an approximate CGRA, commercialized by IBM for the acceleration of AI tasks [4, 33], in which 8-way PEs and Special Function Units (SFUs) are capable of performing multiply-accumulate (MAC) operations on sub-INT and FP numbers, respectively. Overall, to enable ILP and DLP execution models, these SIMD/MIMD architectures are characterized by their high area-footprint [49, 50], stemming from separate datapaths in each of their constituent PEs.

Approximate CGRAs: literature studies in this cutting-edge track are limited to power-gating heterogeneous PEs (each PE has separate and/or different inexact types of adder/multiplier/divider that are *adopted from literature*) or replacing accurate multiplier/divider with approximate ones. X-CGRA [24, 25], PX-CGRA [38], and GP-CGRA [23] integrate approximate adders, multipliers, and dividers into their heterogeneous PEs to trade energy consumption with the accuracy of the final output. The heterogeneous architecture of *GP-CGRA* consists of a 2D array of approximate and accurate tiles. The work has adopted a set of approximate adders and multipliers from the literature. The accurate divider is also adopted from OpenCores [51] and further approximated based on the application requirements at *design* time. Based on the given quality constraints, various *feasible* CGRA configurations are generated through an iterative-process by approximating an operation in the application's Data flow Graph (DFG) and verifying whether the accuracy constraint is met. In the final configuration which has the lowest power among the feasible ones, each CGRA tile is assigned with a specific functional unit, which only supports one precision mode (selected at design time). Switching to accurate configuration is enabled at run-time via power-gating the approximate tiles. *X-CGRA* is composed of Quality Scalable PEs, each of which consists of accuracy-configurable approximate Carry-Look-Ahead (CLA) adder and Dadda multiplier. Each QSPE supports four accuracy modes for the pair of adder and multiplier, i.e., each can be accurate or inexact. As only one approximate mode is supported for all the adders (and one for all the multipliers) of the whole CGRA, such *required* approximation level should be determined at design time. In X-CGRA work, assigning the quality level of each adder and multiplier node in the DFG representation of the application is solved via the Integer Linear Programming (ILP) approach which is rather a time-consuming heuristic. In another trend, the recent work of [39] exploits Stochastic Computing (SC) to convert multiplication to simple bit-wise AND operation. However, such serial-based processing of the bit-streams results in significant latency-overhead. To amortize the area- and/or the latency-overhead of the aforementioned functional units, authors in [27, 52] have narrowed their focus toward enabling *approximate* SIMD multiplier/divider. However, designing approximate SIMD/MIMD PEs for harnessing the DLP/ILP capabilities of error-resilient programs is still unexplored.

Precision-tuning strategies for multi-kernel applications: as the pioneer of *bio-signal processing* approximation, XBioSip has presented an approximation strategy for multi-kernel applications and evaluated it on Pan-Tompkins algorithm for heart-beat detection. In XBioSip, the precision of all kernels is uniformly tuned to 16-bit, but each kernel uses a different level of approximation (via truncation of LSBs) for its addition and multiplication operations. These levels are determined through an aggressive approach, i.e., kernels are approximated as much as possible, in their appearance order. Therefore, this approach did not consider the significance of kernels in terms of their contribution into the *gained performance* versus *quality loss*, when approximation is applied. Other works in this field have mostly focused on layer-wise quantization of Neural Networks (NNs) [53, 54, 55, 29]. In their adopted strategies, ranking the layers for quantization is mainly determined by their robustness to precision scaling. In other words, their proposed saliency metric is fluctuations in the final Quality of Result (ΔQoR). However, considering merely ΔQoR, neglects the significance order of layers on the end-to-end gained performance, or in general, the *relation* between *performance gain* and *QoR loss*. Tackling this short-

coming, Plasticine methodology [22] has proposed $\frac{\Delta\text{Performance}}{\Delta\text{QoR}}$ as the saliency metric and then proposed a greedy heuristic for precision-tuning of multi-kernel applications, which has shown better results over SoAs and provides near-optimal solutions. Considering the lower complexity and usually better run-time of greedy heuristics over multi objective strategies such as Genetic Algorithms, we exploit the sensitivity analysis and heuristic proposed in Plasticine [22] for precision-tuning of applications in this article.

III. Preliminaries and Background

Mitchell's Multiplication and Division Algorithms: as shown in Eq. 1 and Eq. 2, Mitchell's algorithms perform imprecise multiplication and division in the logarithmic representation of numbers. Consider the binary representation for N-bit unsigned input A, which can be written as Eq. 3, where k (the exponent) indicates the position of the leading one. The rest of the bits (starting from position $k-1$ to 0) are considered as the fractional part and fall in the range of $0 \leq x < 1$ [56].

$$P = A \times B \xrightarrow[\text{Log}]{\text{Approx.}} \widetilde{Log_P} = \widetilde{Log_A} + \widetilde{Log_B} \xrightarrow[\text{Anti-Log}]{\text{Approx.}} \tilde{P} = 2^{\widetilde{Log_P}} \quad (1)$$

$$D = A \div B \xrightarrow[\text{Log}]{\text{Approx.}} \widetilde{Log_D} = \widetilde{Log_A} - \widetilde{Log_B} \xrightarrow[\text{Anti-Log}]{\text{Approx.}} \tilde{D} = 2^{\widetilde{Log_D}} \quad (2)$$

$$A = 2^k + \sum_{i=0}^{k-1} 2^i b_i = 2^k(1+x) \xrightarrow{e.g.} 58 = 2^5(1+0.11010)_2, 18 = 2^4(1+0.001)_2 \quad (3)$$

In the linear approximation of *log* function, $log_2(1+x)$ is approximated to x when $0 \leq x < 1$ [57]. Therefore, the approximate log of input A is obtained by concatenation of integer part (the exponent k) and fractional part (the rest of the bits, starting from position *k-1* to *0*), as shown in Eq. 4:

$$\log_2(A) \simeq k + x \rightarrow \log_2(58) \simeq (101.11010)_2, \log_2(18) \simeq (100.001)_2 \quad (4)$$

After applying the same step on the second input to get its approximate log, the summation (subtraction) of two parts is obtained in Eq. 5 (Eq. 6).

$$\widetilde{Log_2}(\tilde{P}) = (k_1 + k_2) + (x_1 + x_2) \rightarrow K_s = (1001)_2, X_s = (0.1111)_2 \quad (5)$$

$$\widetilde{Log_2}(\tilde{D}) = (k_1 - k_2) + (x_1 - x_2) \rightarrow K_s = (1)_2, X_s = (0.1011)_2 \quad (6)$$

Finally, by applying the anti-log (which mathematically is a shift operation), binary representation of the approximate product (quotient) are derived by Eq. 7 (Eq. 8):

$$\widetilde{P} = \begin{cases} 2^{k_1+k_2}(1 + x_1 + x_2), & x_1 + x_2 < 1 \\ 2^{k_1+k_2+1}(x_1 + x_2), & x_1 + x_2 \geq 1 \end{cases} \rightarrow \widetilde{P} = 992, P_{acc} = 1044 \quad (7)$$

$$\widetilde{D} = \begin{cases} 2^{k_1-k_2-1}(2 + x_1 - x_2), & x_1 - x_2 < 0 \\ 2^{k_1-k_2}(1 + x_1 - x_2), & x_1 - x_2 \geq 0 \end{cases} \rightarrow \widetilde{D} = (11)_2 = D_{acc} = 3 \quad (8)$$

Performing Log and anti-log is obtained by using variable bit-width barrel-shifters. The amount of bits to be shifted – for extracting the fractional parts – depends on the design of the multiplier or divider. For example, when operation precision is 16-bit, maximum bits to be shifted for the input and output barrel-shifters are 15 and 31, respectively, in multiplication mode, as the summation of error-correction terms sometimes results in an overflow: $integer\_part1 + integer\_part2 + overflow = 15 + 15 + 1 = 31$. During the shifting, extra zeros are inserted to the right-side of the fractional parts for proper alignment and keeping the bit-width constant (i.e. 4-, 8-, or 16-bit depending on the operating-precision mode). Further implementation details can be found in [46, 26].

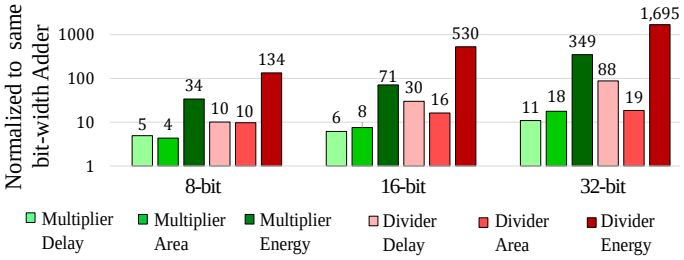

Fig. 2: Comparing area, delay, and energy of 8-, 16-, and 32-bit addition, multiplication, and division operations, when implemented in an ASIC approach.

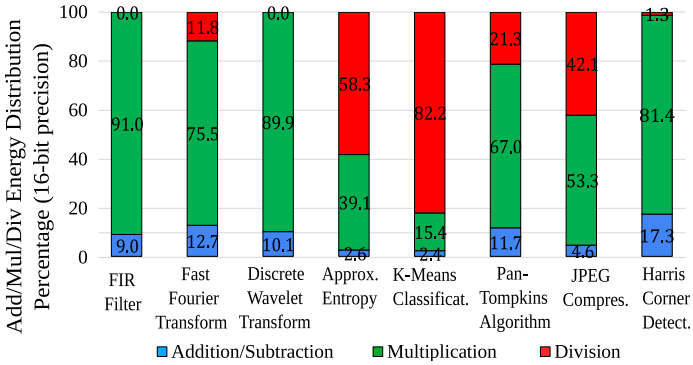

Fig. 3: The cumulative energy of ALU operations in bio-signal and image processing applications (16-bit Precision)

IV. GREEN: A SIMD/MIMD CGRA for Energy-Efficient Processing at the Edge

To design the GREEN CGRA which enables runtime accuracy-energy trade-off, we have targeted approximation and optimization techniques at three layers of abstraction: inexact multiplication and division (circuit-level) [27], utilizing SIMD/MIMD opportunities (architecture-level), and precision scaling (application-level) [22]. The reason behind adopting precision scaling is three-fold: first, not only it reduces the load of computation, but also the end-to-end latency of the application will be shortened, due to the reduction in the propagation delay of individual operations (in cases that a lower precision can be employed for all kernels of an application). Second, precision scaling perfectly aligns with the structure of SIMD/MIMD modes (when needed) and can further reduce the memory footprint and its associated data movement energy. Third, as shown by multiple studies on the same applications, neglecting the LSBs due to the precision-scaling will not significantly affect the output QoR [21, 26, 22].

A. Performance Metrics of ALU Operations and ALU Energy Breakdown in Single- and Multi-kernel Applications

Fig. 2 shows area, delay, and energy metrics for different operations (addition, multiplication, division) of different operand sizes (8-, 16-, and 32-bit). As demonstrated in this figure, all the performance metrics in multiplier and divider are exponentially larger than of an adder having the same size

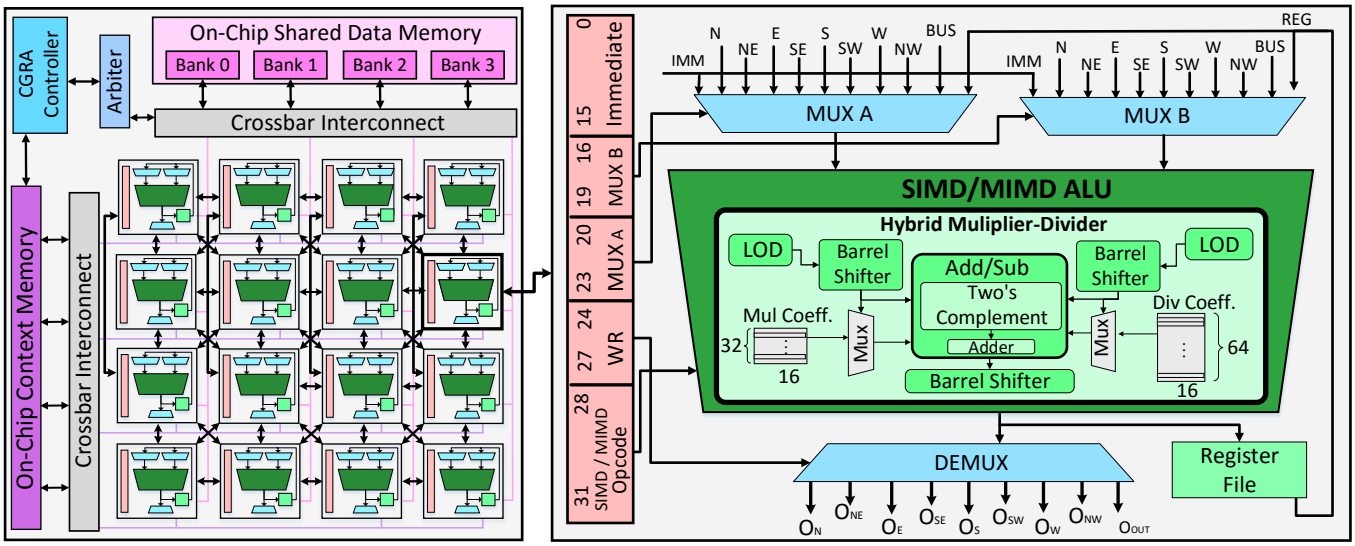

Fig. 4: (a) GREEN CGRA architecture, (b) The proposed structure of SISD/SIMD/MIMD Processing Element (PE)

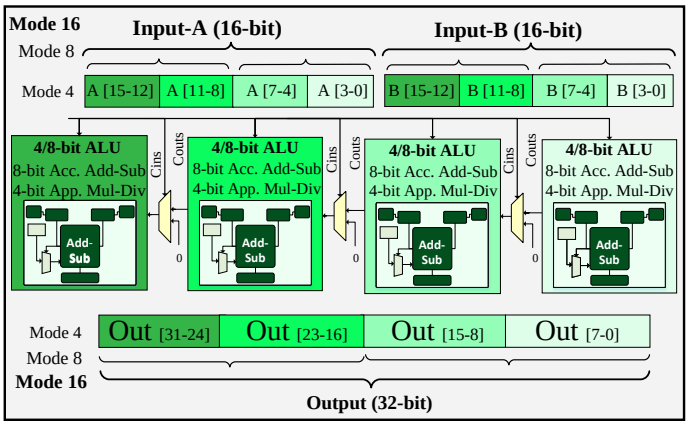

Fig. 5: The SISD/SIMD/MIMD structure of GREEN ALU (based on ASIC-modified multiplier-divider of SIMDive [27])

(similar observations have been partly concluded in studies like [26, 57]). This motivates the use of approximate multiplier and divider at circuit-level.

In addition, we have also measured the energy consumption of different ALU operations (based on their distribution), in eight single- and multi-kernel applications and the results are illustrated in Fig. 3. The case-study applications cover tasks from pre-processing (band-pass FIR filter for noise removal), to feature extraction (Fast Fourier Transform, Discrete Wavelet Transform, and approximate entropy), and classification (K-Means). These multi-kernel applications are ubiquitous in classification, bio-signal (ECG/EEG), image-, and video-processing domains. For example, Pan-Tompkins algorithm not only is the atomic task in heart diseases diagnosis or brain epilepsy and sleep apnea, but also employed in biometric authentication. The results shown in Fig. 3 exhibit that although division is less frequent, it still consumes substantial portion of ALU energy in approximatable kernels of the applications.

The aforementioned observations and different resiliency of application kernels to precision scaling and inexact multiplication and division operations (see Sec. VI-B) demonstrates the need for an architecture that supports different accuracy levels at runtime (application-level optimization through precision-scaling). This is achievable by the proposed GREEN SIMD/MIMD CGRA, discussed in the following. Enabling such accuracy-energy and accuracy-performance trade-offs on-the-fly also contributes to extending the battery lifetime and the real-time processing in IoT edge devices.

B. The Structure of Proposed SIMD/MIMD CGRA

GREEN CGRA, depicted in Fig. 4, features a systolic array structure containing homogeneous PEs, each of which encompasses an ALU that can be configured to SISD, SIMD, or MIMD mode.

Structure of PEs: each ALU is mainly assembled by a shifter, Leading-One Detector (LOD) to determine the position of the leading one in the binary representation of the input, adder, and two's complement unit to handle signed numbers or subtraction operation which is used also in the division operation. We kept the structure of addition/subtraction unit accurate, for two reasons. First, addition-subtraction unit not only requires smaller area, but also consumes less energy versus multiplication and division (see figures 2 and 3). Second, according to the experiments conducted in recent studies [58, 22] and our sensitivity analysis (see Sec. VI. B), on different applications from matrix multiplication based, image, and bio-signal processing domains, the approximation of addition operation can result in higher QoR fluctuations than when multiplication is approximated.

For multiplication and division (implemented using the above-mentioned ALU components), we have designed a customized SIMD multiplier-divider based on SIMDive [27], the rationale behind which is three-fold: ① the error-reduction approach proposed in SIMDive is adjustable and independent from operand-size. Plus, the errors are nearly-unbiased (centered nearly symmetrical around zero), therefore, can nullify each others in successive kernels having mainly Add/Mul operations (unbiased-error has shown to play a pivotal role in

approximation of consecutive kernels having an aggregation-based structure [59, 60, 54]). Such a low-error feature also enables exploiting a chain of approximation techniques, i.e., precision scaling on top of approximate multiplication and division. ② Our ASIC-customized multiplier-divider implementation, even in SIMD mode is significantly better than their accurate counterparts, in terms of area, latency, and energy. Such a resource-efficient unit bodes well for the design of a SIMD ALU, especially as the latency of approximate division is reduced, even less than of a same-size multiplier (see Table III). ③ The flexibility provided by the multiplication-division hybrid mode enables on-the-fly switching between two functionalities without the need for reconfiguration while facilitating a resource-efficient implementation for an ALU. Our additional modification over SIMDive lies in reducing the error-coefficients for multiplication mode (from 64 to 32) and specializing its LUT-customized structure for an ASIC implementation, to be utilized in a SIMD/MIMD CGRA.

In the proposed GREEN ALU (shown in Fig. 5), the integration of multiplication and division together with addition and shift operation has enabled three advantages: ①, this integration has prevented the overhead of collecting individual units separately, and hence, reduced the area footprint and energy of the ALU. ② As mentioned previously, division acts as the speed-bottleneck operation in ALU. Therefore, integrating the multiplier and divider units not only has substantially circumvented the long latency of division to the latency of the same-size multiplier, but also significantly reduced the critical path delay of ALU (especially for 16-bit precision). Such an integration is also highly desirable for e.g., high-performance or RISC-V processors employing 32/64-bit integer and single/double-precision floating-point operations. ③ The proposed ALU supports an approximate *SIMD* division by decomposing a larger one into smaller instances (which is not mathematically practical in *accurate* mode). Enabling sub-word level parallelism via modifying the SISD, to also provide support for SIMD/MIMD modes, is achieved by configuring the carry chain (for Add/Sub units), LOD units, multiplexers associated with small ROMs, and output barrel-shifters to provide support from 16-bit SISD for parallable 4/8-bit modes.

Table II presents the supported ALU opcodes in GREEN that supports on-the-fly alteration of operations' precision and functionalities in SISD/SIMD/MIMD modes. The light, medium, and dark green present the functions that will be used in SISD, SIMD, and MIMD mode, respectively. The presented 16 opcodes are selected, based on the most frequent SISD/SIMD/MIMD functionalities that are required in the eight case study applications. In fact, the selection has been done considering both the results of the error-resiliency sensitivity analysis (to obtain the lowest bit-width precision for each of the kernels' operations) and profiling the sequence of operations in the computational kernels. Nonetheless, opcodes can be customized/expanded depending on the requirements of the target application, especially considering that 10-bit immediate field (rather than 16-bit in the current version) suffices accommodation of the largest filter-coefficients in the case study applications. PEs also include small ROMs for the constant error-reduction coefficients for multiplication and division (the constant coefficients are directly obtained from our previous study [27] and are applicable to different sizes of multiplication/division). To minimize the size of this ROM and its associated multiplexer that selects the coefficient, we stored half of the multiplication coefficients, as swapping the operands does not affect SIMDive multiplication error. Selection of the appropriate coefficient is carried through the multiplexers illustrated in Fig. 4 (32-to-1 for multiplication and 64-to-1 for division), based on the three MSBs of the fractional parts of the given input operands [27]. In case of 4 or 8-bit SIMD/MIMD modes (see Fig. 5), the appropriate coefficients are selected via separate multiplexers and added to each of the 4/8-bit ALUs. Finally, the intermediate result of a PE is stored in its local registers and can be accessed for the next operation.

TABLE II: Supported SISD, SIMD, and MIMD opcodes in GREEN CGRA

Function	Opcode	Function	Opcode
ADD32	0000	DIV8_DIV8	1000
MUL16	0001	DIV4_DIV4_DIV4_DIV4	1001
DIV16	0010	ADD8_MUL4_MUL4_MUL4	1010
ADD16_ADD16	0011	ADD8_ADD8_DIV8	1011
ADD16_ADD8_ADD8	0100	ADD8_MUL8_DIV4	1100
ADD8_ADD8_ADD8_ADD8	0101	ADD8_MUL4_DIV8	1101
MUL8_MUL8	0110	MUL8_DIV4_DIV4	1110
MUL4_MUL4_MUL4_MUL4	0111	ADD16_MUL8	1111

Modifications to enable sub-word parallelism at run-time: enabling the 16-bit SISD ALU to also provide support for 8-/4-bit SIMD/MIMD modes (shown in Fig. 5) is achieved by modifying each sub-modules, separately: the LODs and barrel-shifters are implemented with a modular approach based on 4-bit LODs. Together with extra circuitry (including a multiplexer which is controlled by the SISD/SIMD/MIMD signal), they can operate at 8- and 16-bit at run-time as well. Extra cascading logic is also used to concatenate the fractional parts when precision is 8- or 16-bit. On the other hand, the adders became configurable to 4-, 8-, and 16-bit by inserting multiplexers at proper locations along the carry chain (see Fig. 5). As mentioned previously, the error-coefficients are applicable to different sizes of multiplication/division, therefore, marginal modification were required for the ROMs, so that they can be accessed by multiple ports (in case of sub-word parallelism).

Local and global memories: In the GREEN architecture two memory types are used. The local memories are dual ported register files and the global memory is constructed out of dual ported SRAM units. The size of global memory is 12 kByte and it is sufficient for e.g., 30-second batch of 16-bit ECG samples, acquired at a sampling frequency of 200 Hz (30 sec $\times$ 200 Hz $\times$ $\frac{16}{8}$ Byte = 12 KiB). Each bank in the global data memory, can be individually powered off, on, or placed into retention mode until the next processing interval. Multi-banking feature also allows performing simultaneous operations with various functionalities and precision, needed for SIMD/MIMD modes. Currently, the GREEN CGRA loads data from global memory and stores it in the local register files, and the design of a Direct Memory Access (DMA) Controller is envisioned for future versions of GREEN.

Network structure: The interconnect pattern in GREEN CGRA is a 2D mesh. As CGRAs have a variation

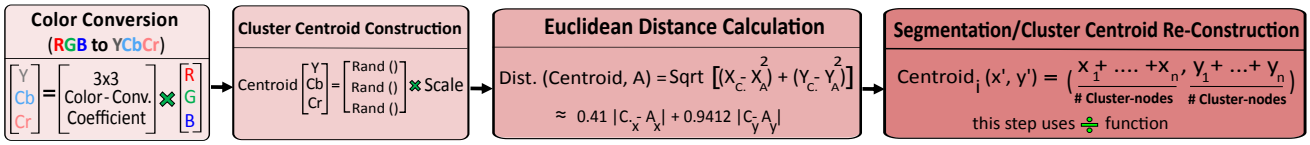

Fig. 6: Kernel structure of K-Means application

of interconnect structures depending on application-domain use-case, we have augmented the network with diagonal and 2-hop links (between first and third rows) which enable energy-efficient processing of bio-signals [61]. Although the addition of these links has increased the interconnection area/power, ultimately the Instruction Per Cycle (IPC) and execution time/energy of the total application has improved: prolonged schedules are avoided when PEs are not used as routing nodes. Similar observations on the benefit of using diagonal links are also drawn in [62, 63, 64].

V. Application-Level Sensitivity Analysis

The recent studies of [22, 54] on multi-kernel applications have shown that the significance order of kernels might differ w.r.t the error-resiliency and the gains in each of the performance metrics, when approximated by the same techniques (or in general, the relation between end-to-end ΔPerformance-gain and ΔQoR).

Therefore, to efficiently adjust the approximation knobs in the eight case study applications, herein we apply the sensitivity analysis proposed in [22]. In fact, this is a prerequisite to tune the precision of different kernels that will be mapped on the SIMD/MIMD GREEN CGRA. Via this early-analysis, we explore the effect of inexact multiplication-division and precision scaling on the end-to-end accuracy and performance metrics of the applications. Moreover, we also estimate the minimum required number of PEs for individual kernels. Minimizing the number of PEs can also positively contribute to reducing the overhead of reconfiguration/data movement between consecutive kernels by e.g., concurrently executing two kernels on the same CGRA.

We have modified the sensitivity analysis of [22] from three perspectives: 1) for this manuscript, the synthesis flow and performance measurements are changed from the LUT- to the ASIC-based implementation. 2) we have also targeted a resource-efficient mapping, customized for an ASIC implementation (e.g., efficient mapping of FFT on CGRA [65]). 3) we have further expanded the depth of the sensitivity analysis and approximation heuristic by also considering the precision of 4-bit in our Design Space Exploration (DSE).

Application partitioning and mapping: the source-code of JPEG and K-Means applications are adopted from AxBench [66], Pan-Tompkins algorithm from XBioSip [21], and Harris Corner Detection (HCD) is developed from scratch. Before mapping, the basic implementation of JPEG and K-Means applications are further optimized for a resource-efficient implementation by e.g., transforming 2D-DCT computations to the butterfly-based 1D-DCT approach [66, 67]. It should be noted that the non-critical portions of each application (e.g., not related to memory or loop index calculation) are extracted for the approximations. Afterwards, the applications are partitioned into various *computational* kernels (excluding instructions that are not implementable by the CGRA). Then, through an in-house C++ script, as shown in Fig. 8, each of the constituent kernels are converted into a Data-Flow Graph (DFG) to be mapped on the CGRA. In the DFG generating phase, we did not apply any optimization technique except loop-transformation (loop-unrolling and flattening the nested loops) and partial prediction (in which both operations of a conditional statement are assigned to different nodes/PEs). Afterwards, though profiling the output, common simple operations and/or SIMD/MIMD possible operations without data-dependency were grouped together considering both the architecture of the given SIMD/MIMD ALU and the sensitivity-analysis results that reveals whether the kernel can be approximated to a lower bit-width precision (this is discussed in the following). Finally, the DFGs are mapped onto the CGRA based on the widely-used list scheduling algorithm (also adopted by HLS-based mapping approaches in the literature [68]), which traverses the kernels' DFG nodes with a resource-aware approach [69]. Dijkstra's algorithm [70] has been employed to find the shortest path for the mapping (binding) of operations with data-dependency that cannot be directly mapped to neighbouring nodes.

Observations from sensitivity analysis: results are demonstrated in Fig. 7 (the experimental setup are detailed in Section VI). Among the eight case study applications, four are multi-kernel (K-Means classification, Pan-Tompkins heartbeat QRS detection, JPEG Compression, and Harris Corner Detection). For the sake of brevity, the structure of K-Means is illustrated in Fig. 6 and readers can refer to [18] for more details on the structure of other multi-kernel applications. It should be noted that for the Euclidean distance kernel, used in the K-means application, we have employed the widely-used approximation method of Baptista [71] which is more straightforward and resource-efficient than general strategies for the approximation of square root function [72, 73, 74]. A summary of the sensitivity analysis results are discussed herein:

- *Substituting accurate multiplier-divider with SIMDive*: deploying SIMDive multiplier-divider in the kernels has marginally affected the QoR in most of the applications. For example, as shown in Fig. 7, the PSNR of 16-bit SIMDive based Pan-Tompkins applications is 42.3 and the QRS detection is maintained at 100%. Such a good PSNR is endowed to the near-zero biased error of SIMDive units (which can cancel out each other) when exploited in the aggregation-based structure (i.e., mostly addition/multiplication operations) of the kernels [22]. Based on this observation, we have been able to exploit a homogeneous structure of PEs, all utilizing the SIMDive units (rather than collecting separate versions of accurate multiplier and divider, which is adopted by other approximate CGRA works in the literature [24, 25, 38, 23]). In contrast, the approximation of

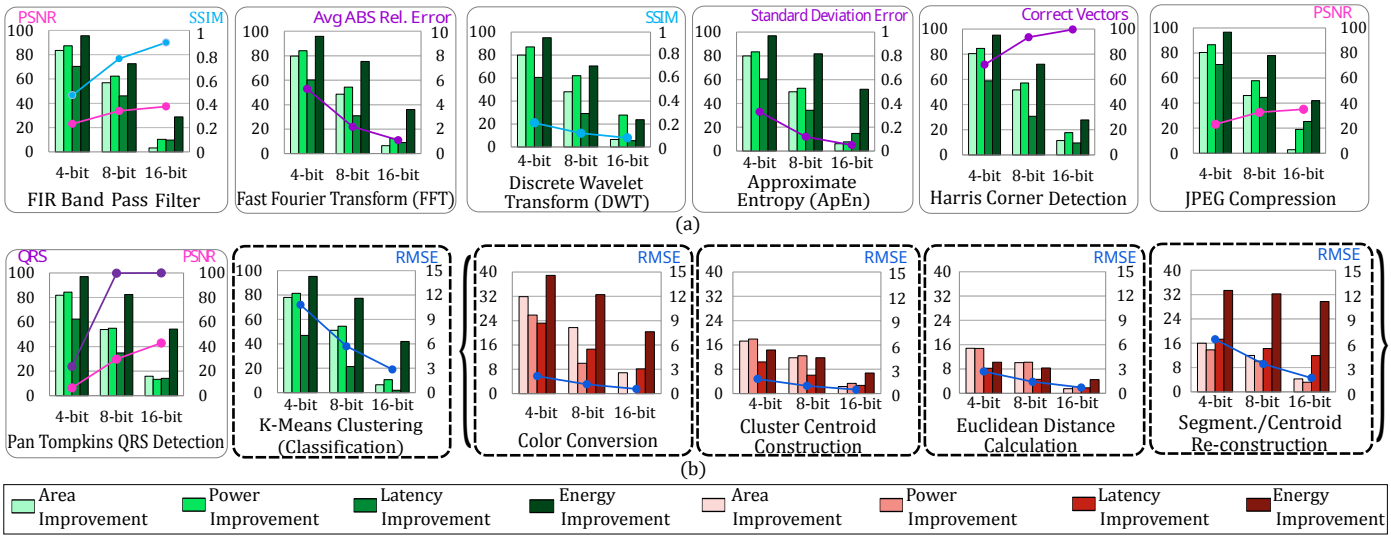

Fig. 7: (a) Sensitivity analysis of eight single- and multi-kernel applications to SIMDive multiplication & division and precision scaling (Ref: 16-bit kernels with accurate multiplication-division): figures show the trade-off between end-to-end Δperformance & ΔQoR, after approximation. (b) A detailed example on the sensitivity of *individual* kernels (K-Means application).

the addition functions can adversely affect the accuracy, especially when applied in kernels such as *differentiator* or *moving average window* in Pan-Tompkins algorithm or *corner response calculation* in Harris Corner Detection. This observation justifies our previous statement in Section IV. B that considering the small contribution of adder in the total area/energy of PE (see Fig. 2 and Fig. 3), it is better to preserve the addition function accurate and instead focused on an adaptable precision scaling strategy.

- *Down-scaling precision from 16- to 8-, and 4-bit*: analyzing real-world ECG and EEG signals from MIT-BIH and CHB-MIT databases [75] shows that the samples are unevenly distributed in the range of 16-bit: 94.8% of ECG and 92.2% of EEG input samples can be trimmed to be fitted into 8-bit, respectively, without affecting final QoR significantly, when passed through the calculations. In fact, even the uniform 8-bit precision with accurate operations provides nearly 100% heartbeat QRS detection. Moreover, the fluctuations in the image PSNR and the number of detected corners/correct vectors in Harris Corner Detection, after down-scaling the precision to 8-bit, were also less than 1% and 7%, respectively. However, setting the uniform precision of 4-bit for *all* kernels, have resulted in noticeable accuracy drop for some applications, especially the ones included more than one division operation. For example, as shown in Fig. 7, the QRS detection ratio is drastically dropped to $\sim$24% in Pan-Tompkins algorithm or RMSE in K-Means application goes beyond 16%. Such observations demonstrate that a mixed-precision strategy is better suited for approximation of multi-kernel applications.
- *Kernels/operations contribute differently to the QoR changes and the gains in different performance metrics*: as can be observed in Fig. 7, applying approximation may result in different orders for Δ*performance gain* and Δ*QoR*. For example, as can be seen in K-means case study, when area improvement is the goal, the approximation of 'color conversion' is more beneficial, while 'cluster centroid construction' is more error-resilient to approximation. In fact, the importance order of kernels may differ, even for different performance metrics. Such observations are also corroborated for other applications, when approximated by different approximation techniques e.g., LSB truncation of addition and multiplication functions [21, 22].

TABLE III: Circuit-level metrics of 16-bit adder and multiplier and 8-bit divider (normalized to the accurate versions)

		ARE[1]	PRE[2]	EB[3]	Area	Power	Delay	Energy
Accurate	Acc_Add (16-bit)	-	-	-	1	1	1	1
	Acc_Mul (16-bit)	-	-	-	1	1	1	1
	Acc_Div (8-bit)[4]	-	-	-	0.6	1.13	1.28	1.44
GREEN	Acc_Add (16-bit)	-	-	-	1	1	1	1
	SIMDive Mul/Div 64-coeff. (16-bit) [27]	0.8/0.7	6.9/5.2	-0.04/-0.01	0.51	0.48	0.71	0.35
XBioSip [21]	AppAdd5 (2-16)	0.01-30	100	-0.01-6.9	0.52	0.48	0.7	0.35
	AppMultV1[5] (2-16)	0.01-61	100	0.03-60	0.54	0.53	0.63	0.37
X-CGRA [24]	RAP-CLA W6	0.2	100	0.2	1.16	1.18	0.87	1.03
	Dadda DQ42C4	8.1	51	8.1	0.66	0.62	0.69	0.43
GP-CGRA [23]	Add_Design 4	0.3	100	0.01	0.57	0.45	0.66	0.35
	Mul_Version Lit	3.4	22	3.4	0.8	0.9	0.88	0.79

[1]Average of Absolute Relative Error, [2]Peak Relative Error, [3]Error Bias (all error metrics are presented in percentage), [4]Normalized to 16-bit multiplier. [5]AppMultV1 adder, used in XBioSip approach, is aggressively approximated.

VI. Results and Discussion

A. Experimental Setup, Circuit- & Architecture-Level Results

Architectural parameters: We have assessed the collective performance metrics for different array sizes of GREEN, in all SISD, SIMD, and MIMD modes with 16-bit (subword of 8- and 4-bits) inputs. GREEN is evaluated against approximate 16-bit X-CGRA [24] and GP-CGRA [23]. The divider size has been adjusted for 8-bit for these CGRAs. This decision is adopted based on the reason that division is not only the most area- and energy-consuming ALU operation (recalling Fig. 2), but also it is the most sensitive operation w.r.t the changes in applications' QoR (see Fig. 7). In fact, our analysis delineates that some division-included kernels/applications

undergo noticeable QoR drop when the precision of division is set to 4-bit (see Fig. 7). After setting the precision of division to 8-bit, we have analysed approximate adder and multiplier candidates in each X-CGRA and GP-CGRA works and selected those structures with lowest *resource*×*error bias* (note, in X-CGRA and GP-CGRA studies, inexact adders and multipliers with different approximation approaches have been examined). It should be mentioned that the approximations of selected adders and multipliers are also set to the level that the 100% QRS detection is maintained for heartbeat detection application. This is achieved by focusing more on approximating the multiplier units, as they are less QoR-sensitive and more energy-hungry than adders. In addition, we have evaluated XBioSip approximation approach [21] by implementing it in CGRA with a minimal-overhead for making 2-16 LSBs in adder/multiplier configurable (in order to support different LSB-truncation variations for different kernels). We kept the same routing structure in all CGRAs and refrained from utilizing specialized scheduling/mapping optimizations.

Experimental Setup: Fig. 8 illustrates the design- and approximation methodology in GREEN CGRA. The architectures are coded from scratch in HDL Verilog, synthesised with Nangate 45-nm technology library using Synopsis Design Compiler. All the memories are also implemented by commercial 40-nm technology that can be taped out [44, 76]. RTL simulations are performed on the post-synthesis net-lists to obtain the activity files and afterwards, the net-lists combined with the activity files are input to the Cadence Innovus for placement/routing. Critical path delay and power estimations are then obtained by simulating the post placed-and-routed net-list at the typical corners. In parallel, the accuracy of approximation approaches has been assessed in both circuit-level (peak and average of absolute relative error and error-bias) and application-level (PSNR, SSIM, RMSE, QRS detection percentage, number of correct vectors, and standard deviation error) through Python and MATLAB simulations.

Circuit-Level Results (individual adders, multipliers, and dividers): Table III summarizes circuit-level characteristics of inexact adders, multipliers, and dividers and provides insights for the selection of SIMDive [27] which bodes well for design of an ALU to be utilized in the GREEN CGRA. Following inferences are highlighted in results:

- Table III shows that the SIMDive multiplier/divider achieves the lowest error-bias while yielding higher performance improvement. Specifically, the latency of divider is reduced to less than of an accurate multiplier of the same size. This justifies that SIMDive hybrid multiplier/divider suits for an approximate ALU design. It is also worth underlining that although our application-level analysis exhibits that even 4-coefficient version of SIMDive provides 100% QRS detection accuracy, but we have used the more accurate, 64-coefficient version, to create opportunity for employing the adaptable greedy precision-scaling strategy of [22] for multi-kernel applications.
- The circuit-level results also demonstrate that the Mitchell-based designs are more suited than accurate or modular-based competitors for an SIMD ALU design as: 1) an SIMD divider is only achievable in an *approximate* mode. 2) Considering the hierarchical-based structure of accurate multipliers (2D array of half-adders/full-adders), the resource footprint grows quadratic (x^2) when the operand size is doubled. This factor would be smaller ($\sim$2.8) in the SIMDive-based ALU: for instance, a double sized Add/Sub is achieved by connecting two smaller instances. Moreover, thanks to the 2D to 1D conversion through Mitchell's logarithmic multiplication, the Add/Sub unit, itself, is re-used in the SIMDive multiplication-division unit.

TABLE IV: Architecture-level metrics of GREEN and baseline CGRAs at post-synthesis phase

		Area[1] (μm^2 $\times 10^3$)	Power[2] (mW)	Chain Latency (ns)	Peak Throughput (GOPS)	Energy (pJ)	Area (μm^2 $\times 10^3$)	Power (mW)	Chain Latency (ns)	Peak Throughput (GOPS)	Energy (pJ)
		PE					2×2				
Accurate		4.7	0.23	9.8	0.1	2.35	17.1	0.83	19.3	0.39	8.4
GREEN	SISD	3.4	0.2	4.9	0.2	1.16	11.5	0.72	10.8	0.74	4.4
GREEN	SIMD	4.8	0.25	5.7	0.18 – 0.72[3]	1.43	17.4	0.9	11.3	0.7 – 2.8	5.7
GREEN	MIMD	5.2	0.27	5.9	0.17 – 0.68	1.6	18.9	0.97	12.3	0.65 – 2.6	6.7
XBioSip		6.7	0.31	15.8	0.07	4.91	25.4	1.12	32.8	0.24	18.7
X-CGRA		4.2	0.21	7.9	0.12	1.83	14.7	0.77	17.8	0.47	6.9
GP-CGRA		4.6	0.22	8.5	0.11	2.07	16.6	0.8	18.3	0.44	7.6
		4×4					8×8				
Accurate		71	3.5	39.4	1.59	33.7	279	14.2	79.2	6.27	141
GREEN	SISD	49	3.1	21.9	2.94	16.5	206	12.2	44.8	11.4	68
GREEN	SIMD	73	3.7	23.6	2.7 – 10.8	22.3	302	15.3	47.2	10.8 – 43.3	91
GREEN	MIMD	78	4	25.7	2.4 – 9.9	27	320	16.4	55	9.3 – 37	113
XBioSip		104	4.6	67.4	0.95	78.9	423	18.9	154	3.5	364
X-CGRA		61	3.2	36.1	1.72	28.7	245	13	72.5	7.2	118
GP-CGRA		70	3.4	37.7	1.67	30.2	267	13.8	75.8	6.9	129

[1]Reports are for the logic, excluding the share for global memories and routing.
[2]Synthesis of all CGRAs are performed with area-opt goal at 300 MHz frequency.
[3]Peak throughput of SIMD/MIMD can be up to 4× (when operands are 4-bit).

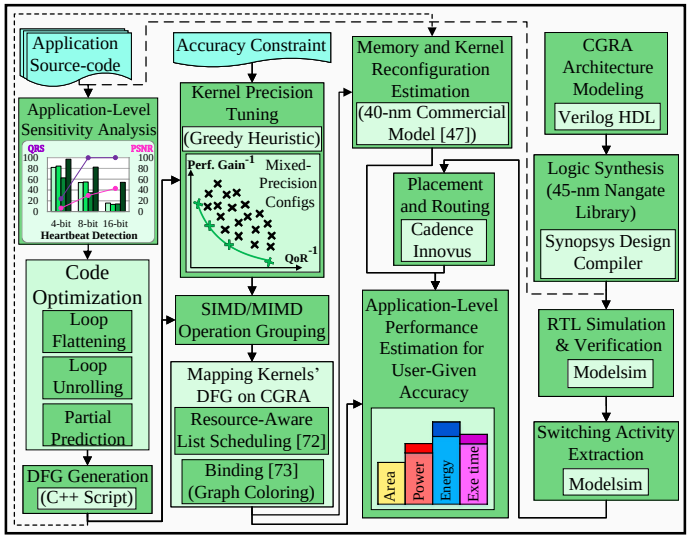

Fig. 8: Overall design and approximation methodology in the GREEN CGRA.

Architecture-Level Results (the whole CGRA): Table IV details the architecture-level metrics of different CGRAs at post-synthesis phase. Please note, recalling Table II, the SISD, SIMD, and MIMD versions of GREEN CGRA support 3, 10, and 16 customized ALU opcodes, respectively, and each variant renders different performance metrics. Afterwards, Fig. 9 demonstrates the area-breakdown and performance-efficiency of all CGRAs, after placement and routing phase on the Cadence Innovus. Finally, Fig. 10 illustrates the quantitative relation among the values (that are presented in Table IV) and compares the performance-efficiency of different CGRAs at

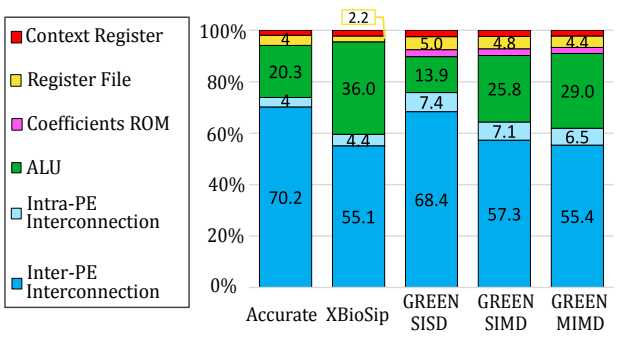

Fig. 9: Area breakdown of CGRAs after place-and-route (excluding the global memory)

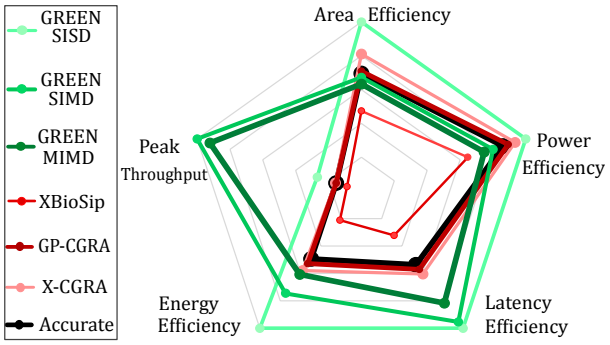

Fig. 10: Performance efficiency of CGRAs at architecture-level (for relative comparison, efficiency is translated as the inverse of resource consumption)

architecture-level. The following inferences are highlighted in the architecture-level results, the results are presented excluding the external memories.:

- *GREEN outperforms other CGRAs at architecture-level*: as indicated in Table IV and Fig. 10, the SISD version of GREEN is up to 32% smaller, 55% more energy-efficient, and 49% faster than the accurate CGRA. Although augmenting SISD to SIMD and MIMD structures has increased the complexity and therefore, resource consumption (e.g., 46.4% and 55.3% area-overhead for SIMD, and MIMD modes, respectively, in the 8×8 CGRA), the maximum throughput is also significantly increased through SIMD and MIMD modes (up to 3.8× and 3.25× for SIMD and MIMD, respectively). Furthermore, the GREEN CGRA can enable up to 6.7× higher throughput compared to the accurate counterpart (when the PEs are configured to SIMD or MIMD mode and operate at 4-bit precision).
- Table IV indicates that the XBioSip approximation approach [21] for multi-kernel applications (enabling accuracy-configurability of LSBs rather than precision-scaling of kernels) is beneficial, only in a fully-customized ASIC implementation. In contrast, the overhead for supporting such LSB configurability in the PE-based template of a CGRA is significant, as all FAs and 2×2 multipliers operating on LSBs of up to 16-bit should support both accurate and approximate modes. As such, this approach will be counter-productive when realized in a homogeneous CGRA structure (please note, it is out of the scope of this work to go into design and evaluating heterogeneous CGRA structures).
- Although the results presented in Table IV and Fig. 10 demonstrate that GREEN SIMD has slightly better performance than the MIMD version at *architecture-level* (as having less complex PE due to supporting less opcodes, see Table II), it will be shown that the MIMD mode renders better performance at *application-level* (see Fig. 11). The reason behind is that the MIMD allows concurrent execution of more functions in the same PE and therefore, kernel operations require less number of PEs to be implemented.

B. Application-Level Results

Precision tuning of kernels for mapping on SIMD/MIMD CGRA: the sensitivity analysis has revealed that the kernels/operations contribute differently in Δperformance gains and ΔQoR. Therefore, to efficiently set the precision of kernels/operations, the overall greedy strategy of [22] has been adopted (the pseudo-code of which is presented in Algorithm 1), the goal of which is to maximize performance gains while maintaining a user-defined accuracy threshold. The inputs of heuristic are the user QoR constraint and the information obtained from the sensitivity analysis, i.e., four lists $L_1 - L_4$ that are for approximating multiplication, division, or down-scaling the precision of the kernel to either 8- or 4-bit. Each list contains the end-to-end performance-gain and QoR-loss of the entire application, when only that kernel is approximated with the target technique and the rest are accurate. It should be noted that the greedy heuristic is customized for this article, by using the $\frac{\Delta\text{Throughput}}{\Delta\text{QoR}}$ as the deciding metric that reflects the gained performance over a possible accuracy loss. In fact, a higher throughput (gained by down-scaling the precision) better reflects the performance gain in a pre-fabricated SIMD/MIMD CGRA that is consisted of *fixed-area* PEs. The methodology is as follows: first, all application kernels, having accurate operations, are uniformly set to 16-bit precision. Afterwards, the *Saliency List* is generated by calculating the $\frac{\Delta\text{Throughput}}{\Delta\text{QoR}}$ for each pair of {kernel, technique}. The lists are then merged and sorted in a descending order to demonstrate the saliency order of techniques that resulted in higher performance gain with less QoR loss (when considered individually). Finally, the greedy heuristic is applied with an iterative approach: in each iteration, the pair of {kernel, technique} is selected that exists on top of the saliency list. Recalling the sensitivity analysis, the primary-applied techniques are replacing accurate multiplication and division with SIMDive versions of the same bit-width precision, due to their low error and high performance gain potentials (especially for division operation). The generated configuration is then evaluated on diverse samples to verify whether the end-to-end user-defined QoR threshold is maintained. Whenever the accuracy of the generated configuration crosses the threshold (with up to 5% difference with the user-defined threshold), the heuristic backtracks to the previous accuracy-satisfied configuration and follows the search by appraising the next candidate on top of the saliency-list.

For the QoR threshold of applications, in this manuscript we have adhered to 100% QRS detection ratio for Pan-Tompkins (and PSNR of 30 dB), 10% RMSE for K-Means, PSNR of 28 dB for JPEG compression, and 90% correct vector

Algorithm 1: Greedy Approximation Heuristic for Multi-Kernel Applications (customized from [22])

Input: L1: {E2E Tput Gain$_{PS}$, QoR-Loss} $\forall$ 8-bit Prec. Kernel
Input: L2: {E2E Tput Gain$_{PS}$, QoR-Loss} $\forall$ 4-bit Prec. Kernel
Input: User-QoR-Const, Kernel-List
Output: Kernels [Precision]

```
Saliency-List = Array [];
// Calc. throughput-gain of prec. scaling, on each kernel, individually
for i in Kernel-List do
    // This kernel is scaled to 8-bit, others are 16-bit
    Saliency-List ← L1[i]→ (ΔTput/ΔQoR)(8);
    // This kernel is scaled to 4-bit, others are 16-bit
    Saliency-List ← L2[i]→ (ΔTput/ΔQoR)(4);
end
Descending Sort (Saliency-List);
while (!timeout) do
    for i in Saliency-List do
        // Approximate in descending order of ΔTput/ΔQoR
        Config_Approx = Kernels [Saliency-List_i];
        Output-QoR = Evaluate (Config_Approx);
        // Also explore temporary configs
        if Output-QoR ≥ 0.95 × User-QoR-Const then
            if Output-QoR ≥ User-QoR-Const then
                // Update candidate configuration
                Config_temp ← Config_Approx;
            end
            i ← i + 1
            go to 10;
        else
            Break;
        end
    end
end
```

detection for Harris Corner Detection (that is reported to be an acceptable confidence-level for object/movement tracking [77]). It should be noted that, although the study of XBioSip [21] has set the PSNR threshold of 19 dB (which also satisfies QRS ratio of 100%), herein we adhere to a higher PSNR value, i. e., 30 dB, to ensure an acceptable signal strength in the output (in cases that the output signal is further used for other processing, e.g., detecting the of type of the anomaly). For the above-mentioned quality thresholds, the following precision-configurations are found by the heuristic: [4-8-4-16-16] for the Harris Corner Detection having five-kernels, [4-4-8-4-16] for the five-kernel ECG Pan-Tompkins QRS detection, [4-4-8-16] for the four-kernel K-Means, and [4-8-16] for the three-kernel JPEG Compression. Accordingly, the adopted SISD/SIMD/MIMD structure of kernels for the proposed kernel configurations is detailed in Table V. It should be noted that for packing the SIMD/MIMD operations inside the precision reduced kernels, the dependency of multiplication and/or division operations (referring to Fig. 6) has been taken into account.

We have adopted the one-kernel-at-a-time strategy of [10], suited for the processing of bio-signals, as these applications have brief period of sampling followed by a relatively large

TABLE V: Kernels' SISD/SIMD/MIMD structure in three application, w.r.t the final approximate configuration

Harris Corner Detection		**Pan-Tompkins (ECG)**	
Kernel Name	**Configuration**	**Kernel Name**	**Configuration**
RGB to Grayscale	SIMD 4-bit	Low-Pass Filter	MIMD 4-bit
Gaussian Smoothing	MIMD 8-bit	High-Pass Filter	SIMD 4-bit
Derivative Sobel	MIMD 4-bit	Differentiator	MIMD 8-bit
Tensor & Score Response	SISD 16-bit	Squarer	SIMD 4-bit
Normalization	SISD 16-bit	Moving Avg Filter	SISD 16-bit
K-Means Clustering		**JPEG Compression**	
Kernel Name	**Configuration**	**Kernel Name**	**Configuration**
RGB to YCbCr	MIMD 4-bit	RGB to YCbCr	MIMD 4-bit
Cluster Cent. Const.	SIMD 4-bit	2D-DCT	MIMD 8-bit
Euclidean Distance Calc.	SIMD 8-bit	Quantization	SISD 16-bit
Segmentation/Cluster Cent. Re-Construction	SISD 16-bit		

idle interval. In this strategy, the output of intermediate kernels are stored in the on-chip memory – which are directly accessible by PEs – and fetched when PEs are configured to implement the next kernel (an 8×8 array suffices mapping the biggest sub-kernels of the case study applications).

Execution Time and Energy Measurements: to calculate the end-to-end execution time of the applications, the widely-used list scheduling algorithm has been adopted for traversing the kernels' DFG nodes with a resource-aware approach [69]. For energy measurements, we have followed the energy-model and estimation approach of [76]. In this model, formulated in Eq. 9, the energy of computation, memory, and kernel configuration (pre-stored in the memory) are calibrated separately:

$$E_{memory} = (N_{load} \times E_{load}) + (N_{store} \times E_{store}) + (N_{not-idle} \times E_{static}) + (N_{idle} \times E_{idle}) \quad (9)$$

Energy of computation: the static power of each CGRA architecture is extracted from the Innovus reports. For the dynamic power measurements, the RTL simulations were performed on a random sequence of data on the post-synthesis net-lists of the CGRA architectures to obtain the CGRA-specific activity files. *Energy of memory and kernel configuration:* for this part, three terms are summed up (see Eq. 9), as described in the following: ① *Access energy*: energy of memory load and store, which are measured separately. ② *Static (memory-enabled)*: the static energy of memory in one clock cycle is multiplied with the number of clock cycles that the kernel is executing. ③ *Static (memory-disabled)*: the idle energy of memory in one clock cycle is multiplied with the number of clock cycles that it is idle. It should be noted that each of these numbers (for one clock cycle) are looked up directly from the commercial manufacturer datasheets for the 40-nm technology node [76].

Application-Level Comparison of CGRAs: Fig. 11 compares the performance metrics of different CGRAs at application-level, through executing multi-kernel applications. Recalling Table V, the SISD/SIMD/MIMD configuration of PEs for the GREEN CGRA, are adjusted via the modified greedy approximation strategy of [22] for multi-kernel applications. The upper part of each bar (power, energy, or reconfiguration time) in Fig. 11 exhibits the share for reconfiguration, while the lower part represents the share of computation. The amount for each of the performance metrics is normalized to its baseline

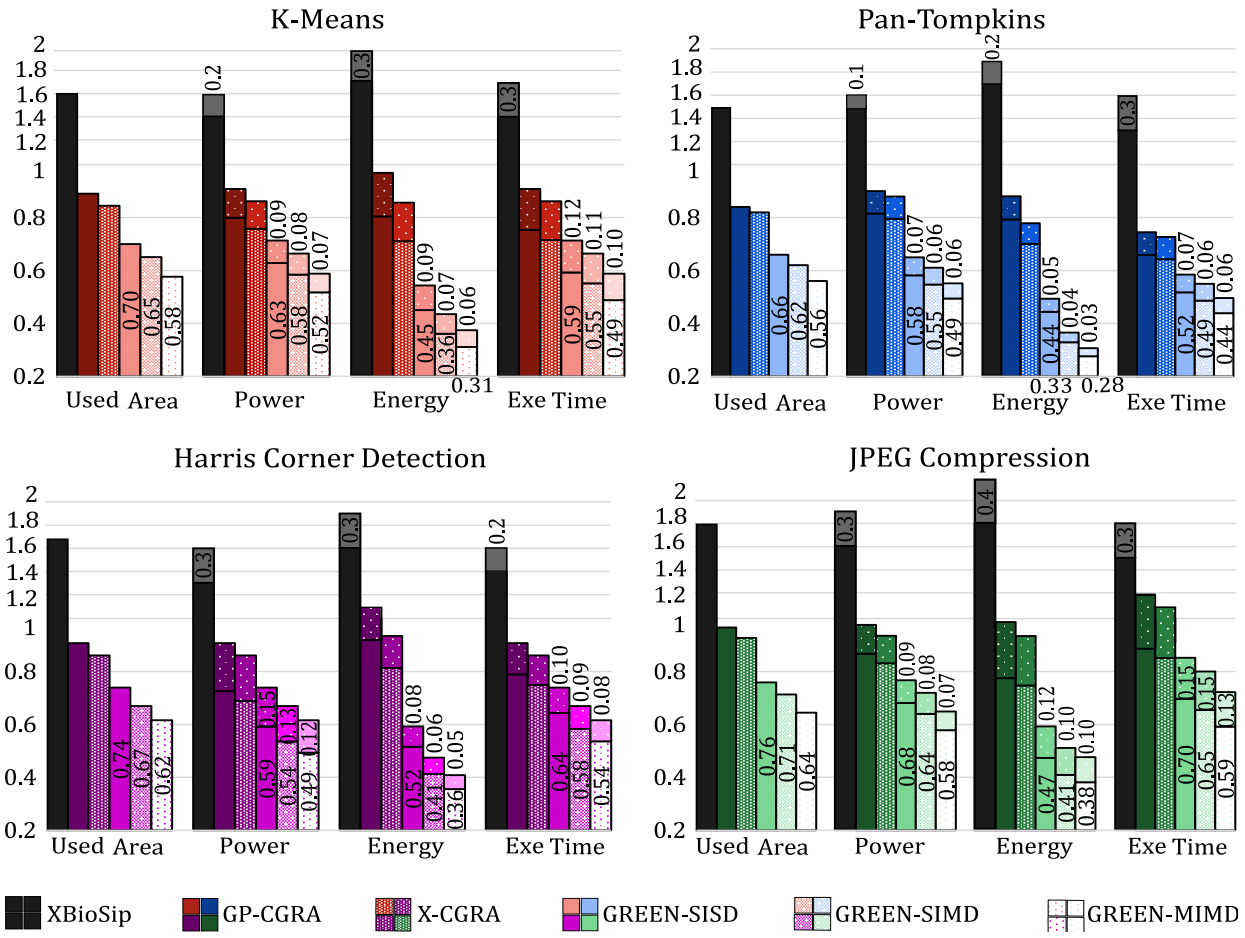

Fig. 11: Application-level comparison of CGRAs, by running multi-kernel applications

counterpart, i.e., of the accurate CGRA. For example, the pair of power values (1.4, 0.2) in the K-Means application in Fig. 11 demonstrates that the total power of XbioSip-based CGRA is 1.6× of accurate CGRA, in which the shares of computation and reconfiguration are 1.4× and 0.2×, respectively. Please note that for the sake of visibility, the maximum value in the y-axis of sub-figures are set to 1, as only the XbioSip-based CGRA has lower performance than the accurate CGRA. The following deductions are notable:

- *GREEN offers substantial application-level savings, while maintaining an acceptable QoR*: although the architecture-level gains of GP-CGRA and X-CGRA (from Table IV) are also reflected at the application-level, ultimately the SIMD/MIMD modes of GREEN enable the highest performance gains by implementing multiple functions of reduced-precision in a smaller number of PEs.
- As can be seen in the figures, the overhead of reconfiguration time and energy not only depends on the application structure, but also on the CGRA architecture. In fact, for each of the applications, the SIMD/MIMD version of GREEN CGRA requires lower reconfiguration time and energy, as kernels are precision-reduced and less (groups of) data needs to be communicated to/from the memory. For example, such overheads are the lowest for the Pan-Tompkins application, in which the average precision of kernels are more reduced (kernel structure of [4-4-8-4-16]) and highest for the JPEG Compression, in which kernels are less precision-reduced ([4-8-16]).
- *Benefits of GREEN CGRA for edge computing*: processing at a lower precision has two advantages. First, it contributes to the savings in the memory energy, both for reconfiguration and data access. Second, the end-to-end latency of the application is reduced, due to the shortened propagation delay of precision-reduced individual operations (in application cases that all PEs are configured to operate at 4- or 8-bit precision). This can improve the clock frequency of the CGRA and hence, the execution time of the application. In fact, improving the application speed and energy is highly desirable for enabling real-time processing at the edge (e.g., heart anomaly detection in wearable 24/7 gadgets). Enabling such a shorter response time and reduced energy consumption through processing and storing data in the proximity of data is referred to as *Multi-access Edge Computing* (MEC) [78]. MEC can provide many advantages for battery-operated devices. From energy and real-time processing perspectives, it enables pre-processing and filtering or extraction of necessary features at the edge (rather than transferring the whole sampled data to the cloud). From data privacy perspective, private information of users can be preserved at the edge, e.g., in smart-health gadgets.

It is worth highlighting that the unutilized PEs of

GREEN CGRA (when running the program in SIMD/MIMD modes) can be power-gated, or leveraged for analysis of data at a higher sampling rate or accommodation of more tasks.

VII. Conclusions and Future Work

In this article, we proposed GREEN, which serves as a stepping stone for the energy-efficient processing of multi-kernel applications at the edge. The proposed SIMD/MIMD CGRA template benefits from high-throughput and energy-efficiency of instruction- and data-level parallelism enabled by the light-weight PEs, each of which supports different functionalities at various data-width precision. In fact, each of PEs implements a chain of approximations in a cross-layer hierarchy (precision scaling on top of the inexact multiplication-division in a hybrid functionality). The evaluation of GREEN MIMD on applications from classification, bio-signal (ECG/EEG), and image/video processing domains have demonstrated that not only it enables up to 6.6× higher throughput than the accurate CGRA, but also achieves up to 49% and 66% reduction in the end-to-end application latency and energy, respectively (while maintaining an acceptable QoR).

Acknowledgement

We thank Dr. Mark Wijtvliet for his guidance for the presented work. This research is co-funded by the German research foundation (DFG) on project X-ReAp (380524764) and the European Social Fund (ESF) on Project Re-learning (100382146).

References

[1] Changmoo Kim et al. 2014. ULP-SRP: Ultra Low-Power Samsung Reconfigurable Processor for Biomedical Applications. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 7, 3.

[2] Jinjie Tang et al. 2019. Processors, Methods, And Systems With A Configurable Spatial Accelerator Having A Sequencer Data-flow Operator. US Patent 10,380,063.

[3] Brendan Barry et al. 2015. Always-on Vision Processing Unit for Mobile Applications. *IEEE Micro*, 35, 2.

[4] Swagath Venkataramani et al. 2021. RaPiD: AI Accelerator for Ultra-low Precision Training and Inference. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*.

[5] Chris Nicol. 2017. A Coarse Grain Reconfigurable Array for Statically Scheduled Data Flow Computing. *Wave Computing White Paper*.

[6] Brian Gaide et al. 2019. Xilinx Adaptive Compute Acceleration Platform: VersalTM Architecture. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*.

[7] Kermin E Fleming et al. 2019. Apparatus, Methods, And Systems With A Configurable Spatial Accelerator. US Patent 10,445,250.

[8] AnandTech. 2015. Samsung Exynos 7420 Deep Dive - Inside A Modern 14nm SoC. (2015).

[9] Loris Duch et al. 2019. i-DPs CGRA: An Interleaved-Datapaths Reconfigurable Accelerator for Embedded Bio-Signal Processing. *IEEE Embedded Systems Letters (ESL)*, 11, 2.

[10] Loris Duch et al. 2017. HEAL-WEAR: An Ultra-Low Power Heterogeneous System for Bio-Signal Analysis. *IEEE Transactions on Circuits and Systems I: Regular Papers (TCAS-I)*, 64, 9.

[11] World Health Organisation. 2018. Cardiovascular diseases. (2018).

[12] Statista Global Data Platform. 2020. Number of Wearable Devices Worldwide by 2022. (2020).

[13] Global Data. 2019. Wearable Technology Value by 2023. (2019).

[14] Cisco. 2020. Cisco Annual Internet Report - VNI Complete Forecast Highlights. (2020).

[15] Cisco. 2018. Cisco Annual Internet Report - VNI Complete Forecast Highlights. (2018).

[16] Sandvine. 2021. The mobile internet phenomena report 2021, (2021).

[17] Sandvine. 2019. (2019).

[18] Zahra Ebrahimi et al. 2022. RAPID: AppRoximAte Pipelined Soft MultIpliers and Dividers for High-Throughput and Energy-Efficiency. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*.

[19] Leibo Liu et al. 2019. A Survey of Coarse-Grained Reconfigurable Architecture and Design: Taxonomy, Challenges, and Applications. *ACM Computing Surveys (CSUR)*, 52, 6.

[20] Rubén Braojos et al. 2017. A Synchronization-Based Hybrid-Memory Multi-Core Architecture for Energy-Efficient Biomedical Signal Processing. *IEEE Transactions on Computers (TC)*, 66, 4.

[21] Bharath Srinivas Prabakaran et al. 2019. XBioSiP: A Methodology for Approximate Bio-Signal Processing at the Edge. In *ACM/IEEE Design Automation Conference (DAC)*.

[22] Zahra Ebrahimi et al. 2021. Plasticine: A Cross-Layer Approximation Methodology for Multi-Kernel Applications through Minimally Biased, High-Throughput, and Energy-Efficient SIMD Soft Multiplier-Divider. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 27, 2.

[23] Marcelo Brandalero et al. 2018. Approximate On-the-Fly Coarse-Grained Reconfigurable Acceleration for General-Purpose Applications. In *ACM/IEEE Design Automation Conference (DAC)*.

[24] Omid Akbari et al. 2020. X-CGRA: An Energy-Efficient Approximate Coarse-Grained Reconfigurable Architecture. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 39, 10.

[25] Omid Akbari et al. 2018. Toward Approximate Computing for Coarse-Grained Reconfigurable Architectures. *IEEE Micro*, 38, 6.

[26] Hassaan Saadat et al. 2019. Approximate Integer and Floating-Point Dividers with Near-Zero Error Bias. In *ACM/IEEE Design Automation Conference (DAC)*.

[27] Zahra Ebrahimi et al. 2020. SIMDive: Approximate SIMD Soft Multiplier-Divider for FPGAs with Tunable Accuracy. In *ACM Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*.

[28] Muhammad Shafique et al. 2016. Invited - Cross-Layer Approximate Computing: From Logic to Architectures. In *ACM/IEEE Design Automation Conference (DAC)*.

[29] Vojtech Mrazek et al. 2019. ALWANN: Automatic Layer-Wise Approximation of Deep Neural Network Accelerators without Retraining. In *International Conference on Computer-Aided Design (ICCAD)*.

[30] Yao Fu et al. 2017. Deep Learning with INT8 Optimization on Xilinx Devices. *White Paper*.

[31] Martin Langhammer et al. 2020. High Density Pipelined 8bit Multiplier Systolic Arrays for FPGA. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*.

[32] M. Langhammer et al. 2019. Extracting INT8 Multipliers from INT18 Multipliers. In *International Conference on Field Programmable Logic and Applications (FPL)*.

[33] Swagath Venkataramani et al. 2020. Efficient ai system design with cross-layer approximate computing. *Proceedings of the IEEE*, 108, 12.

[34] Zahra Ebrahimi and Akash Kumar. 2021. BioCare: An Energy-Efficient CGRA for Bio-Signal Processing at the Edge. In *IEEE International Symposium on Circuits and Systems (ISCAS)*.

[35] Hartej Singh et al. 2000. MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications. *IEEE Transactions on Computers (TC)*, 49, 5.

[36] Yoonjin Kim et al. 2009. Low Power Reconfiguration Technique for Coarse-Grained Reconfigurable Architecture. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, 17, 5.

[37] Amir Yazdanbakhsh et al. 2018. GANAX: A Unified MIMD-SIMD Acceleration for Generative Adversarial Networks. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*.

[38] Omid Akbari et al. 2018. PX-CGRA: Polymorphic approximate coarse-grained reconfigurable architecture. In *Design, Automation & Test in Europe Conference Exhibition (DATE)*.

[39] Bo Wang et al. 2022. Towards Energy-Efficient CGRAs via Stochastic Computing. In *Design, Automation & Test in Europe Conference Exhibition (DATE)*.

[40] Cheng Tan et al. 2022. ASAP: Automatic Synthesis of Area-Efficient and Precision-Aware CGRAs. In *ACM International Conference on Supercomputing (ICS)*.

[41] Wooseok Byun et al. 2022. An Energy-Efficient Domain-Specific Reconfigurable Array Processor With Heterogeneous PEs for Wearable Brain–Computer Interface SoCs. *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*.

[42] Barry de Bruin et al. 2020. Brainwave: an energy-efficient eeg monitoring system - evaluation and trade-offs. In *ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*.

[43] Barry de Bruin et al. 2021. Multi-Level Optimization of an Ultra-Low Power BrainWave System for Non-Convulsive Seizure Detection. *IEEE Transactions on Biomedical Circuits and Systems (TBioCAS)*, 15, 5.
[44] Mark Wijtvliet et al. 2019. Blocks: Redesigning Coarse Grained Reconfigurable Architectures for Energy Efficiency. In *International Conference on Field Programmable Logic and Applications (FPL)*.
[45] SmartCardia. 2018. ARM Cortex-M3 inside SmartCardia. (2018).
[46] Hassaan Saadat et al. 2020. Realm: reduced-error approximate log-based integer multiplier. In *Design, Automation & Test in Europe Conference Exhibition (DATE)*.
[47] Alejandro Nieto et al. 2016. PRECISION: A Reconfigurable SIMD/MIMD Coprocessor for Computer Vision Systems-on-Chip. *IEEE Transactions on Computers (TC)*, 65, 8.
[48] Rohit Prasad et al. 2020. TRANSPIRE: An energy-efficient TRANSprecision floating-point Programmable archItectuRE. In *Design, Automation Test in Europe Conference & Exhibition (DATE)*.
[49] Intel. 2020. Intel Stratix 10 GX/SX Device Overview. (2020).
[50] Xilinx. 2018. Zynq-7000 SoC Technical Reference Manual. (2018).
[51] OpenCores. 2011. Hardware Division Unit. https://opencores.org/projects/divider. (2011).
[52] Roberto R. Osorio and Gabriel Rodríguez. 2019. Truncated SIMD Multiplier Architecture for Approximate Computing in Low-Power Programmable Processors. *IEEE Access*, 7.
[53] S. Lee et al. 2017. High-level synthesis of approximate hardware under joint precision and voltage scaling. In *Design, Automation, & Test in Europe (DATE)*.
[54] Z. G. Tasoulas et al. 2020. Weight-Oriented Approximation for Energy-Efficient Neural Network Inference Accelerators. *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, 67, 12.
[55] P. Judd et al. 2015. Reduced-precision strategies for bounded memory in deep neural nets. *arXiv preprint*.
[56] Zahra Ebrahimi et al. 2020. LeAp: Leading-one Detection-based Softcore Approximate Multipliers with Tunable Accuracy. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*.
[57] H. Saadat et al. 2018. Minimally Biased Multipliers for Approximate Integer and Floating-Point Multiplication. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 37, 11.
[58] H. Jiang et al. 2020. Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications. *Proceedings of the IEEE*, 108, 12.
[59] G. Zervakis et al. 2021. Approximate Computing for ML: State-of-the-Art, Challenges and Visions. In *Asia & South Pacific Design Automation Conference (ASP-DAC)*.
[60] Mohammad Saeed Ansari et al. 2020. Improving the Accuracy and Hardware Efficiency of Neural Networks Using Approximate Multipliers. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, 28, 2.
[61] João Lopes et al. 2017. Evaluation of CGRA architecture for real-time processing of biological signals on wearable devices. In *International Conference on ReConFigurable Computing and FPGAs (ReConfig)*.
[62] Frank Bouwens et al. 2007. Architectural Exploration of the ADRES Coarse-Grained Reconfigurable Array. In *Reconfigurable Computing: Architectures, Tools and Applications*.
[63] Jonghee W. Yoon et al. 2013. Architecture Customization of On-Chip Reconfigurable Accelerators. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 18, 4.
[64] S. Alexander Chin et al. 2017. CGRA-ME: A Unified Framework for CGRA Modelling and Exploration. In *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*.
[65] Li Yu et al. 2020. Efficient FFT Implementation on a CGRA. In *International Conference on Mathematics and Artificial Intelligence (ICMAI)*.
[66] A. Yazdanbakhsh et al. 2017. AxBench: A Multiplatform Benchmark Suite for Approximate Computing. *Design & Test of Computers*, 34, 2.
[67] M. Jridi et al. 2013. Low complexity DCT engine for image and video compression. In *Real-Time Image and Video Processing (RTIVP)*. Volume 8656.
[68] Jong Kyung Paek et al. 2011. Binary acceleration using coarse-grained reconfigurable architecture. *SIGARCH Comput. Archit. News*, 38, 4.
[69] Giovanni De Micheli. 1994. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education.
[70] E. W. Dijkstra. 1959. *A Note on Two Problems in Connexion with Graphs*. Numerical Mathematics.
[71] Wikipedia. 2017. Approximation of Euclidean distance. (2017).
[72] Honglan Jiang et al. 2019. Low-Power Unsigned Divider and Square Root Circuit Designs Using Adaptive Approximation. *IEEE Transactions on Computers (TC)*, 68, 11.
[73] E.M. Schwarz and M.J. Flynn. 1993. Hardware starting approximation for the square root operation. In *IEEE International Symposium on Computer Arithmetic (ARITH)*.
[74] Duhwan Kim and Sunggu Lee. 2022. Approximate Square Root Circuits with Low Latency and Power Dissipation. *Electronics*, 11, 1.
[75] Ary L Goldberger et al. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation*, 101, 23.
[76] Mark Wijtvliet et al. 2021. CGRA-EAM—Rapid Energy and Area Estimation for Coarse-Grained Reconfigurable Architectures. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 14, 4.
[77] T. Nomani et al. 2020. xUAVs: Towards Efficient Approximate Computing for UAVs—Low Power Approximate Adders With Single LUT Delay for FPGA-Based Aerial Imaging Optimization. *IEEE Access*, 8.
[78] Alaa Awad Abdellatif et al. 2019. Edge Computing for Smart Health: Context-Aware Approaches, Opportunities, and Challenges. *IEEE Network*, 33, 3.

Zahra Ebrahimi received her B.Sc. and M.Sc. degrees in computer engineering at Sharif University of Technology (SUT), Iran, in 2014 and 2016, respectively. Meanwhile, she was also a Research Assistant at the Data Storage, Networks, and Processing Laboratory, at SUT. She started her Ph.D. at the Center for Advancing Electronics Dresden (cfaed), Technische Universität Dresden, Germany, in 2018. Since April 2024, she is also a Research Associate at Ruhr-Universität Bochum. Her research interests include approximate computing, reconfigurable accelerator design, in-network and edge computing.

Akash Kumar (SM'13) received the joint Ph.D. degree in electrical engineering and embedded systems from the Eindhoven University of Technology, Eindhoven, The Netherlands, and the National University of Singapore (NUS), Singapore, in 2009. From 2009 to 2015, he was with NUS. From October 2015 until March 2024, he was a Professor with Technische Universität Dresden, Dresden, Germany, where he was directing the Chair for Processor Design. Since April 2024, he is directing the chair of Embedded Systems at Ruhr University Bochum, Germany. His research interests include the design and analysis of low-power embedded multiprocessor systems, and designing secure systems with emerging nano-technologies.