INTEGRATION, the VLSI journal 101 (2025) 102337

Contents lists available at ScienceDirect

Integration, the VLSI Journal

journal homepage: www.elsevier.com/locate/vlsi

SEAM: A synergetic energy-efficient approximate multiplier for application demanding substantial computational resources☆

Youngwoo Jeong, Joungmin Park, Raehyeong Kim, Seung Eun Lee *

Department of Electronic Engineering, Seoul National University of Science and Technology, Seoul, 01811, Republic of Korea

ARTICLE INFO

Keywords:
Approximate multiplier
Energy-efficient computing
Low power
Convolutional neural network
Fuzzy logic

ABSTRACT

Approximate computing constitutes a paradigm in which accuracy is exchanged for enhanced energy efficiency when contrasted with conventional computing methodologies. This approach has been devised to address the escalating demand stemming from the rapid expansion of application systems. This paper proposes an approximate multiplier for systems with heavy computational load. By amalgamating the attributes of a Dynamic range unbiased multiplier (DRUM) with an Approximate wallace tree multiplier (AWTM), we have devised a Synergetic energy-efficient approximate multiplier (SEAM) aimed at mitigating the occurrence of worst-case errors inherent in AWTM. The SEAM was analyzed for circuit area and power consumption using Design Compiler with Synopsys GPDK 32 nm. Experimental results demonstrated that SEAM achieved up to 80.46% reduction in circuit area and 82.6% reduction in power consumption compared to a precise multiplier. Furthermore, compared to DRUM, SEAM showed a 15.55% reduction in circuit area and 45.73% reduction in power consumption. In order to validate the feasibility of the proposed approximate multiplier, the circuit was implemented on a Field-programmable gate array (FPGA) and applied to a fuzzy logic-based pathfinding algorithm and a Convolutional neural network (CNN) accelerator. For the pathfinding algorithm, most error metrics of the SEAM showed similar values to the DRUM. Moreover, when applied to the CNN accelerator and experimented with the CIFAR-10 dataset and MNIST dataset, the proposed multiplier exhibited identical precision, recall, and F1 score values. Despite applying SEAM, we achieved a maximum 3.1% increase in classification metrics for a specific case. These results indicate the significant potential of the SEAM in reducing the area of overall system while minimizing errors.

1. Introduction

The advancements of the semiconductor industry have led to rapid progress in Artificial intelligence (AI), including machine learning and Convolutional neural network (CNN), with widespread applications across various systems [1,2]. As algorithms of AI become increasingly sophisticated, they show a significant improvement in overall accuracy [3,4]. However, the growing complexity of algorithms and the escalating demand for throughput have led to a need for additional hardware resources [5,6]. This surge in demand has driven the continuous evolution of computing capabilities, accelerating the progress of high-performance processors, dedicated circuits, and application system architectures [7,8]. These demands increasingly require more computing power, thereby emphasizing the importance of low-power and energy efficiency in circuits and systems.

The imperative for energy efficiency has become prominent due to the substantial environmental and economic impact of power consumption in computing [9,10]. Traditional computing methodologies, especially in arithmetic operations like multiplication and division, often impose a significant energy burden as they aim for absolute accuracy. The standardized nature of these operations limits the extent to which optimization can significantly reduce circuit size and power consumption.

In response to this trend, edge computing has emerged, involving the offloading of certain computations from the cloud to edge devices [11]. However, edge devices face greater hardware resource constraints compared to the cloud devices. Therefore, strategies such as adopting hardware-friendly neural networks and algorithms such as Binary neural network (BNN) or K-nearest neighbor (k-NN), which

☆ This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-RS-2022-00156295) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

* Corresponding author.
E-mail address: seung.lee@seoultech.ac.kr (S.E. Lee).

https://doi.org/10.1016/j.vlsi.2024.102337
Received 13 May 2024; Received in revised form 3 September 2024; Accepted 15 December 2024
Available online 30 December 2024

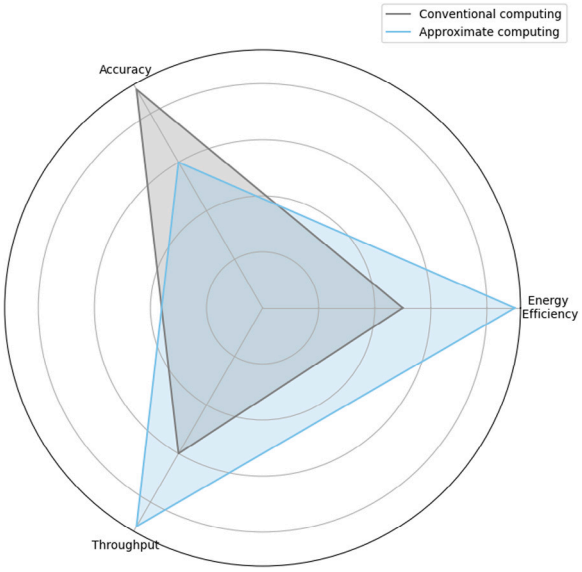

Fig. 1. Difference between conventional computing and approximate computing.

simplify computations, have been introduced to reduce resource usage [12,13]. Despite these efforts, the demand for high-performance CNN remains significant.

To address the need for energy-efficient neural networks, approximate computing has emerged as a promising solution. This innovative paradigm originates from the recognition that not all applications require precise computations. Fig. 1 illustrates the distinction between conventional and approximate computing, emphasizing intentional imprecision in computational tasks. The focus is on strategically balancing the trade-off between accuracy and resource consumption [14,15]. By allowing controlled levels of error in calculations, approximate computing aims to optimize energy efficiency without compromising the essential functionality of diverse applications.

Approximate computing has found applications that tolerate errors or require high performance, such as image processing, artificial intelligence, mobile devices, IoT devices, and real-time processing systems [16–20]. These systems prioritize decreasing power consumption and increasing and energy efficiency by allowing for errors and noise that are imperceptible to humans. The benefits of approximate computing lie in its ability to cater to error-tolerant applications without significantly compromising functionality, making it a valuable approach in various domains.

In this paper, we propose a Synergetic energy-efficient approximate multiplier (SEAM) with a focus on analyzing the feasibility of the proposed circuit when implemented in multiplication and applied to specific applications. The contributions of this paper are outlined as follows:

- We introduce a multiplier that synergistically combines the characteristics of two existing approximate multipliers: the Dynamic ranged unbiased multiplier (DRUM) and the Approximate wallace tree multiplier (AWTM). This unique integration leverages the error resilience of DRUM and the structural efficiency of AWTM, resulting in a hybrid approach that significantly reduces the circuit area and power consumption while maintaining accuracy. This integration is not a straightforward combination but a carefully designed methodology that addresses the limitations of each individual multiplier while enhancing overall performance.

- The proposed SEAM is compared with other state-of-the-art approximate multipliers. A detailed analysis of circuit area and power consumption is performed using Design Compiler, demonstrating that SEAM achieves a 80.46% reduction in circuit area and a 82.6% reduction in power consumption compared to traditional multipliers. These improvements are achieved without compromising accuracy, making SEAM highly suitable for energy-constrained environments.

- The proposed approximate multiplier is applied to two specific application systems: CNN-based image processing and a pathfinding algorithm based on fuzzy logic. Error rates for these application systems are analyzed to assess the viability of the proposed multiplier. The results demonstrate significant improvements in performance and energy efficiency compared to an accurate multiplier, highlighting the practical effectiveness of SEAM in real-world applications.

2. Related work

2.1. Evaluation metrics of approximate computing

The most fundamental performance evaluation metrics for approximate computing are Error rate (ER) and Error distance (ED) [21]. ER represents the probability of errors occurring, while ED indicates the difference between the accurate value and the approximate value. When expressing the accurate value as A and the approximate value as A', ED is calculated as the sum of absolute differences. Relative error distance (RED) expresses the relative difference between the accurate and approximate values, calculated as the sum of absolute differences divided by the accurate value. Typically, Relative error (RE), expressed as a percentage, is favored over RED, calculated as $RE = RED \times 100$. When comparing results, if ED is equal, RE becomes a more crucial performance metric in approximate computing due to its tendency to show larger RE for smaller A values [22]. In experiments involving multiple scenarios, metrics utilizing ED and RED, such as Mean error distance (MED), Mean relative error distance (MRED), Mean squared error (MSE), and Root mean squared error (RMSE), are employed [23]. Eqs. (1), (2), (3), and (4) represent these calculations, with $P(ED_i)$ and $P(RED_i)$ denoting the probabilities of ED_i and RED_i occurrences.

$$MED = \sum_{i=1}^{n} ED_i \times P(ED_i) \tag{1}$$

$$MRED = \sum_{i=1}^{n} RED_i \times P(RED_i) \tag{2}$$

$$MSE = \sum_{i=1}^{n} ED_i^2 \times P(ED_i) \tag{3}$$

$$RMSE = \sqrt{MSE} \tag{4}$$

Absolute value-based metrics are also utilized. Mean absolute error (MAE) and Mean absolute percentage error (MAPE) are metrics that consider the absolute values of errors and their percentage, respectively [24]. These metrics, by not ignoring the magnitude of errors, serve as robust measures of prediction accuracy. Mean percentage error (MPE) is derived from MAPE but without absolute values. MPE can be positive or negative, indicating whether the prediction is larger or smaller than the actual value. While MPE can be useful for evaluating relative accuracy, it is recommended to use it in conjunction with other metrics in cases where the direction of the error is crucial. Eqs. (5), (6), and (7) depict the calculations for these three metrics. In the context of MAE, MAPE, and MPE, 'n' represents the number of samples. These equations utilize N to denote the total number of data points or samples used to compute the respective metrics.

$$MAE = \frac{\sum_{i=1}^{n} |A_i - A_i'|}{n} \tag{5}$$

$$MAPE = \frac{100}{n} \times \left| \frac{\sum_{i=1}^{n} A_i - A_i'}{A_i} \right| \quad (6)$$

$$MPE = \frac{100}{n} \times \frac{\sum_{i=1}^{n} A_i - A_i'}{A_i} \quad (7)$$

Furthermore, Worst-case error (WC-Error) and Worst-case relative error (WCR-Error) are considered, particularly in systems sensitive to large errors [25]. Eqs. (8) and (9) presents the calculations for these metrics.

$$WC - Error = max_{i=1}^{n}(ED_i) \quad (8)$$

$$WCR - Error = max_{i=1}^{n}(RED_i) \quad (9)$$

Additionally, circuit implementation using Electronics design automation (EDA) tools and analysis of area and power consumption are performed [26,27]. For instance, circuits are synthesized using 28 nm CMOS, 32 nm GPDK, 45 nm open-source NanGate, and the results are compared and analyzed to evaluate the performance of approximate computing circuits. As the process and settings significantly impact all experimental outcomes, conducting experiments with the same configuration is crucial for proper comparisons.

The evaluation of approximate arithmetic circuits is challenging with a single metric. Therefore, a comprehensive evaluation using multiple metrics is essential to thoroughly assess the performance of circuit from various perspectives. This multifaceted evaluation allows for a more accurate understanding of the performance of approximate arithmetic circuits and enhances their utility in real-world applications.

2.2. Approximate multiplier

S. Abed proposed the Approximate wallace tree multiplier (AWTM) [28]. The conventional Wallace tree multiplier (WTM) encounters critical path delays in columns where all 8 bits need to be added during an 8×8 multiplication. To address this issue, the paper introduces a separate carry-in preconsumption circuit. Additionally, to reduce circuit size, the proposed multiplier accurately computes only the lower two bits while fixing the intermediate bits to one. The upper bits are computed as accurately as possible using imprecise carry values. Moreover, the operation divides one multiplier into seven multipliers, efficiently reducing circuit size relative to error rates. Simulations for 4×4, 8×8, and 16×16 multipliers yielded an average accuracy ranging from 99.85 to 99.965, with circuit area and power reduced by up to 41.96.

S. Venkatachalam presented an approximate multiplier based on the radix-4 booth multiplier [29]. They reconstructed the K-map by flipping four values in the conventional booth encoding-based K-map. The proposed multiplier further reduced circuit size by using only OR gates, instead of adders, for the lower eight bits in the reconstructed partial product matrix. Experimentation on the 8×8 multiplier demonstrated a 41% reduction in circuit area and 49% decrease in power consumption. Finally, the MRED for the proposed multiplier was measured at 6.75×10^{-2}.

On the other hand, S. Hashemi proposed the DRUM for approximate application systems [30]. The DRUM employs an accurate multiplier but reduces the input bit count to decrease the overall circuit area. For instance, when performing 8 × 8 multiplication, the proposed method involves performing the approximation operation using only a 4 × 4 multiplier. Initially, the Leading one detector (LOD) is used to extract the bits based on the leftmost one as the reference, as numbers have more significant values towards the Most significant bit (MSB). Subsequently, accurate multiplication is performed using the extracted two numbers, and the final approximate result is obtained by left-shifting the result by the truncated bit count. The proposed multiplier can control the error rate by adjusting the number of extracted bits. Experimental results for extraction bit counts from four bits to nine bits showed a maximum average absolute error rate of six.

Additionally, several other approximate multipliers have been studied, including under-designed multiplier, broken-array multiplier, error-tolerant multiplier, inaccurate multiplier, and approximate compressor-based multiplier. Each of these introduces unique strategies to achieve approximate multiplication in various scenarios [31].

2.3. Application

Approximate computing has been primarily applied to systems that are not highly sensitive to errors due to its characteristic trade-off between accuracy and circuit complexity. Numerous studies have explored the application of approximate computing in image processing. In [32], the preapproximation method, which has a uniform distribution and noise with an average of zero, was proposed. The study applied the preapproximation method to edge detection and Gaussian filtering in image processing. The experiments on edge detection verified the efficient operation for most images when using a specific bit length, assessed through structural similarity and peak signal-to-noise ratio metrics. Although there could be a difference of one or two bits in extreme cases, it was considered acceptable since the model was generated based on the probability distribution. Similar experiments were conducted for Gaussian filtering, visually confirming minimal differences between the two cases, validating the feasibility of proposed multiplier.

Research has also been conducted on applying approximate computing to JPEG compression, where C. K. Jha [33] proposed Floating point approximate dividers (FPADs) for IEEE 754 floating-point operations. Since the mantissa division in IEEE 754 dividers constitutes 92% of the area and 95% of the delay, and introducing approximations to the exponent or sign bits would lead to significant errors, the mantissa division was approximated using shift and addition operations. Applying FPADs to the quantization step of the JPEG compression algorithm, the results were compared with existing approximate dividers, achieving 2.84× power-delay product gain while maintaining the same error rate for 512 × 512 grayscale image.

In [34], a probabilistic full-adder, which modified the conventional full-adder, was proposed, and a probabilistic multiplier applying this adder to an array multiplier was introduced. Results from applying probabilistic and conventional multipliers to image sharpening showed visually indistinguishable images. Furthermore, the proposed multiplier required only 50% of the energy compared to the conventional multiplier, demonstrating 2× energy saving with similar performance. Experimental results on landscape images showed more noise compared to other result images. This was attributed to landscape images being dominated by low-frequency components, and it was expected that using more than 50% energy would reduce the noise.

Moreover, research has been conducted on applying approximate computing to deep learning [35,36]. In the context of deep learning, the focus has been on reducing the operations required by approximating multiplication and division, rather than replacing them with approximate multipliers and dividers [37,38]. In [35], an end-to-end trainable Neural Network named AXNet was proposed by merging an approximator and a predictor for the first hidden layer. However, coordinating these two networks is challenging, and separately training them with different optimization goals takes a considerable amount of time. AXNet achieved a 50.7% reduction in training time compared to existing approximate computing frameworks. Research has also explored applying approximate computing circuits to Neural Network accelerators [39].

The power line modulation technique, intentionally altering the output for specific input vectors, was applied to multiplication and addition units. The Neural Network accelerator achieved 1.78−2.67× reduction in energy consumption for the TSMC 65 nm process. Hardware characteristics of the exact Neural functional unit (NFU) and inexact

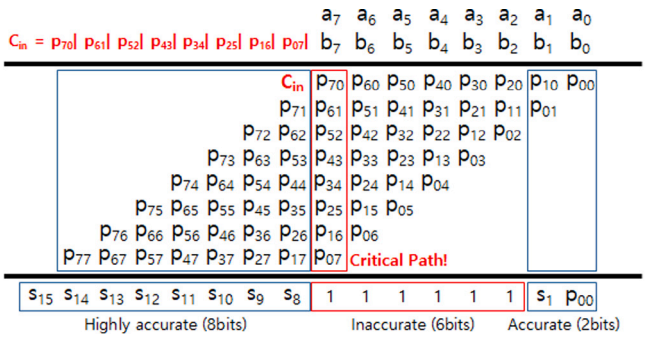

Fig. 2. Approximate wallace tree multiplication.

NFU were analyzed, showing area and power energy measurements of 0.66 mm^2 and 0.28 mm^2, 66.37 mW and 32.86 mW, and 56.91nJ and 28.18nJ, respectively. Finally, experimental results demonstrated that even with approximately 42% less area and 50% less energy, the inexact NFU still achieved 97.1% accuracy compared to the exact NFU.

3. Architecture

A new approximate multiplier is proposed by combining the features of two existing approximate multipliers, AWTM and DRUM. First, Fig. 2 illustrates the multiplication method of AWTM. When the inputs are two 8-bit numbers, the least significant two bits are accurately computed using a half adder, while the middle six bits are fixed to one without computation. The upper 8-bits are approximated by performing an OR operation on the lower bits, and these values are precisely calculated using a half adder and a full adder. This approach ensures accuracy in the lower 2-bits, inaccuracy in the middle 6-bits, and a certain level of accuracy in the upper 8-bits. Since the higher bits are more important than the lower bits in numbers, this method efficiently reduces circuit area and power compared to traditional WTM by eliminating the full adders and half adders required for the computation of the middle bits.

On the other hand, DRUM is an approximate multiplier that extracts an arbitrary k-bit from the MSB to reduce multiplication and later corrects the result through shift operations. By adjusting the value of k according to the application, it is possible to efficiently balance the trade-off between error distance and circuit area in the circuit. However, since an accurate multiplier must be used, increasing k results in a larger area for the approximate multiplier.

SEAM is a mutually complementary approximate multiplier that combines LOD of DRUM with AWTM. The LOD is a circuit that identifies the position of the largest one in the input of the multiplication. Passing through the LOD ensures that the extracted number always has MSB set to one. Using this number as the input for AWTM increases the likelihood that the MSBs of the two input numbers in AWTM are fixed to one. This significantly reduces the occurrence of Worst-Case Error in the computation results of AWTM. The proposed multiplier, like DRUM, also has the potential for development by adjusting the k-bit to regulate the trade-off relationship between Error Rate and Area.

Fig. 3 illustrates the architecture of the proposed SEAM. Similar to the existing DRUM, the circuit consists of steering logic and arithmetic logic for flexibly extracting specific bits. The steering logic includes a type converter, LOD, truncator, and barrel shifter. Initially, the input signed values are converted to unsigned values through the type converter since the final multiplication is performed as an unsigned operation. LOD detects the one closest to the MSB for both input values, and if no one is detected, the final result is fixed to zero. The truncator generates a new number by cutting the desired number of bits based on this value. Then, by replacing the LSB of the two values with one, they are fed into the AWTM. The multiplication result is generated by

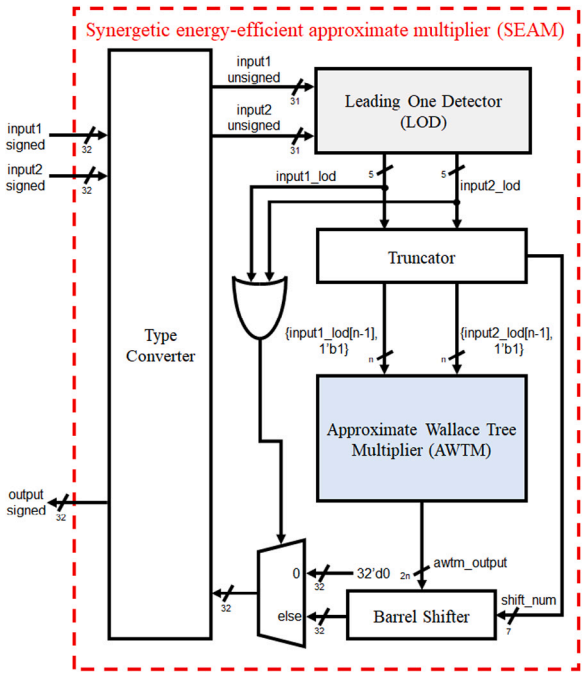

Fig. 3. Architecture of the synergetic energy-efficient approximate multiplier.

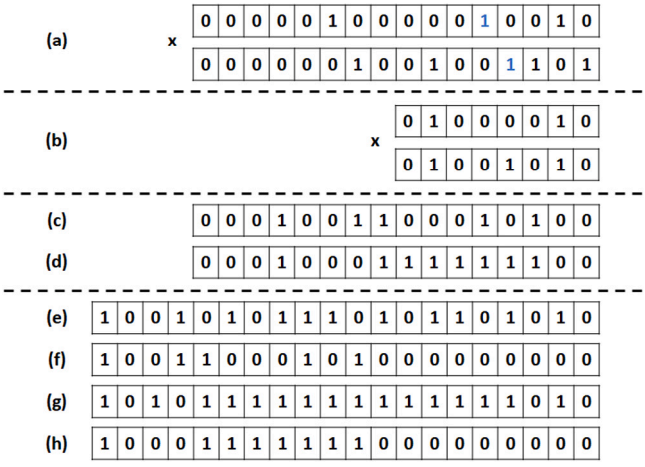

Fig. 4. Flow of the synergetic energy-efficient approximate multiplier. (a) Input value (b) Approximated input (c) Pre-shift operation result of DRUM (d) Pre-shift operation result of SEAM (e) Exact result ($613{,}738_{10}$) (f) Result of DRUM ($616{,}224_{10}$) (g) Result of AWTM ($720{,}890_{10}$) (h) Result of SEAM ($622{,}496_{10}$).

left-shifting the output of AWTM by the previously truncated bits, and the final result is extracted in signed form through the type converter.

Fig. 4 depicts the operational sequence of SEAM. Initially, specific bits are extracted for both input numbers to generate the inputs Fig. 4(b) for DRUM and SEAM, and the LSB is set to one. This ensures that the MSBs of the inputs for AWTM in SEAM are always set to one, minimizing the ER of the multiplier. While there is still a risk of WC-Error if the two numbers in input Fig. 4(a) are less than eight bits, the probability of WC-Error occurring significantly decreases in 32-bit operations, especially in cases where the numbers are in Fixed-point unit (FXU). Fig. 4(c) represents the accurate result of the operation for Fig. 4(b), and (d) illustrates the result of operating Fig. 4(b) with AWTM, showing that all intermediate bits are fixed to one. Finally, Fig. 4(e) to (h) present the final results of each multiplier.

Table 1
Synthesis results of the 32-bit floating point unit and the SEAM-based 32-bit floating point unit.

Multiplier	Area [μm^2]	Power [mW]
Floating-point unit	5417	378.29
SEAM	1929	135.41

To investigate the inherent error rate of the proposed SEAM, we analyzed the MRED and WCR-Error for 65,536 cases in 16-bit multiplication operations using a Python simulator. The measured values were found to be 0.99 and 1.00. In contrast, the MRED of other approximate multipliers analyzed in Fig. 22 of [40] is generally measured between 0 and 0.2. The reason for the higher MRED of SEAM is due to its specific characteristics. This multiplier has a clear WC-Error. When both input numbers are one, the result is fixed at 254. In other words, regardless of the input, the result is always calculated as 254 or higher. This can lead to significant errors in some operations, as the operation for the identity element one does not work accurately. This issue not only occurs for 8-bit operations but also becomes more critical as the input bits increase. Consequently, the measured MRED and WCR-Error for all operations are inevitably higher compared to other approximate multipliers. However, if this error does not have a significant impact at the application level, then the value of SEAM is justified.

4. Experiment

When evaluating the performance of approximate circuits, it is crucial to thoroughly analyze the trade-off between energy-efficiency and accuracy. Therefore, in this paper, we not only implemented the proposed multiplier with Verilog HDL, but also applied SEAM to a CNN accelerator and a fuzzy logic-based pathfinding algorithm by replacing all of the multipliers to SEAM. Finally, we compared area, power, energy, and accuracy with an accurate multipliers and other approximate multipliers in the same environment.

4.1. Energy-efficiency

To assess the energy-efficiency of the proposed multiplier, we synthesized the proposed multiplier using Design Compiler with the Synopsys 32 nm GPDK. First of all, we conducted experiment for comparing the energy-efficiency when applying SEAM to a floating-point unit. We aimed to examine the extent of area and power reduction compared to the floating-point unit. A comparison was made between a 32-bit floating-point unit without SEAM and a unit with SEAM applied. Table 1 presents the synthesis results. As a result, the circuit area decreased by 64.39%, and the total power consumption was decreased by 64.2%.

Additionally, we compared SEAM with other approximate multipliers. Table 2 presents the analysis results of the proposed multiplier, DRUM with various k values, and WTM. All multipliers were based on 32-bit FXU consisting of a 22-bit integer part and a 10-bit fractional part. By analyzing not only energy but also Energy delay product (EDP) and the Product of power, delay, and area (PDA), we aimed to comprehensively evaluate the performance of circuits considering the balance. Lower values in all metrics generally indicate smaller circuits and higher energy efficiency. The circuit area of SEAM was 80.46% smaller than the WTM and at most 55.18% smaller than the DRUM, showing similar values to DRUM6. The power consumption of SEAM was also measured to be 82.6% smaller than the WTM and at most 87.1% smaller than the DRUMs. In terms of critical path delay, the WTM exhibited the lowest value, while SEAM showed 68% higher than the WTM. This was speculated to be due to the use of mux, which is not typically used in general multipliers. Regarding EDP and PDA, SEAM achieved the lowest values. Overall, SEAM performed slightly better than DRUM6 and showed 47.62% and 94.96% performance gains compared to the WTM. The proposed 32-bit FXU based SEAM was applied to embedded fuzzy logic controller to analyze its accuracy.

4.2. Accuracy

4.2.1. Convolutional Neural Network (CNN)

The SEAM was applied to the CNN accelerator to analyze its accuracy. In this experiment, the CNN accelerator proposed by J. Park was adopted, and experiments were conducted on the CIFAR-10 and MNIST datasets [41,42]. The CIFAR-10 dataset consists of images categorized into 10 diverse classes. Each image has a size of 32×32 pixels, with each pixel composed of RGB values. The dataset comprises a total of 60,000 images, with 6000 images per category, divided into 50,000 training data and 10,000 test data. Additionally, the CNN we used has three convolutional layers followed by ReLU and Maxpooling, and two fully connected (FC) layers. All convolution operations use a fixed 3×3 filter with a stride of 1. The input–output channel relationships for the convolution layers are: $(1, 16) \rightarrow (16, 16) \rightarrow (16, 8)$, and for the FC layers: $(72, 64) \rightarrow (64, 10)$. The performance of CNN varies depending on the dataset it is experimented on. For instance, experimenting on entirely different categories such as dogs and airplanes may yield high performance, whereas experimenting on similar categories such as dogs and cats may result in lower performance. Therefore, the aim of this experiment was to measure performance across various categories.

There were seven scenarios as depicted in Table 3. Case A to Case F are categories in the CIFAR-10 dataset that group data with similar features, such as machinery and animals. Meanwhile, Case D and Case E involve adding entirely different categories to two existing similar categories. Finally, Case G focuses solely on experimenting with the MNIST dataset.

In this experiment, input images were represented in a 32-bit floating-point format with values ranging between −1 and 1 to facilitate smooth inference by the CNN accelerator. For input images with three RGB channels, models corresponding to each classification scenario were first trained, and then the parameters were applied to the CNN accelerator for inference. The CNN accelerator embeds a convolution accelerator based on systolic array architecture, enabling more efficient convolution operations [41]. The systolic array structure is crucially utilized in the accelerator as it is well-suited for accelerating convolution operations. Additionally, the accelerator includes functionalities such as zero-padding, max-pooling, as well as activation functions like sigmoid and ReLU. It performs computations based on the IEEE-754 floating-point format. The CNN accelerator consists of processing elements that perform multiplication and addition operations, organized in a systolic array architecture, with many floating-point multiplication units included. We conducted comparative experiments

Table 2
Synthesis results of the various approximate multipliers with Design Compiler.

Multiplier	Area		Power		Delay		Energy		EDP		PDA	
	[μm^2]	Ratio	[mW]	Ratio	[ns]	Ratio	[nJ]	Ratio	[ns × μJ]	Ratio	[$\mu J \times \mu m^2$]	Ratio
Wallace	10,563	5.12	383.14	5.75	24.34	**1.00**	9.33	3.42	0.23	2.03	3.57	19.65
DRUM6	2098	1.02	76.50	1.15	38.98	1.60	2.98	1.09	0.12	1.04	0.23	1.25
DRUM8	2444	1.18	122.82	1.84	39.95	1.64	4.91	1.80	0.20	1.76	0.60	3.31
DRUM16	4605	2.23	516.55	7.75	60.7	2.49	31.35	11.49	1.90	17.05	16.20	89.07
SEAM	2064	**1.00**	66.66	**1.00**	40.92	1.68	2.73	**1.00**	0.11	**1.00**	0.18	**1.00**

Table 3
Scenarios of the experiment.

Case	Category
Case A	Airplane, Ship
Case B	Cat, Dog
Case C	Airplane, Automobile
Case D	Airplane, Automobile, Cat
Case E	Airplane, Automobile, Frog
Case F	Airplane, Automobile, Ship, Truck
Case G	MNIST dataset

by introducing or not introducing SEAM into these multipliers. The experimental environment includes the transmission of input image data, layer information, and commands to the CNN accelerator, along with a data transceiver and CNN accelerator for receiving results. The CNN accelerator is implemented on the Cyclone IV 4CE115C8G model, a field-programmable gate array.

For the evaluation between the CNN accelerator and the SEAM-based CNN accelerator, accuracy, precision, recall, and F1 score were analyzed. To understand these metrics, it is essential to explain True positive (TP), True negative (TN), False positive (FP), and False negative (FN). TP represents instances where the model correctly predicts positive outcomes, accurately identifying actual positive instances. TN refers to instances where the model correctly predicts negative outcomes, accurately identifying actual negative instances. FP occurs when the model incorrectly predicts positive outcomes, falsely identifying an instance as positive when it is actually negative. FN describes instances where the model incorrectly predicts negative outcomes, falsely identifying an instance as negative when it is actually positive. Using these four values, accuracy, precision, recall, and F1 score are computed. Accuracy measures the proportion of correctly classified instances among all instances. Precision represents the ratio of correctly predicted positive samples among the predicted positive classes. Recall indicates how accurately the model detects positives among the actual positive classes. F1 score is the harmonic mean of precision and recall, summarizing the performance of model in a single number. Eqs. (10), (11), (12), and (13) represent the calculation formulas for these four metrics.

In the analysis of the performance metrics presented in Table 4, applying SEAM to the CNN shows that the results are similar to those obtained with exact multiplier and DRUM across various cases. For case A, case C, and case G, the accuracy, precision, recall, and F1 score for SEAM are nearly identical to those of the exact multiplier. In case B and case E, while there are slight variations in accuracy, precision, recall, and F1 scores between exact multiplier and SEAM. Case D shows that SEAM slightly improves recall compared to the exact multiplier, but the accuracy, precision, and F1 score remain similar to those of exact multiplier and DRUM. For case F, the metrics for SEAM are close to those of exact multiplier and DRUM. The slight variations observed are not substantial enough to indicate a significant impact on performance. Overall, SEAM demonstrates performance metrics that are nearly indistinguishable from those of the exact multiplier and DRUM. The differences are minimal, showing that SEAM maintains high levels of accuracy, precision, recall, and F1 score while potentially offering other benefits like energy efficiency and hardware simplification.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{10}$$

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$F1\ score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{13}$$

To further compare with other approximate multipliers, we analyzed delay, power, and area using the results from Table 2 in relation to the findings from Fig. 23 of [40]. For this comparison, the synthesis result of DRUM with UMC 40 nm was employed as the baseline, and the results of SEAM were normalized with Synopsys 32 nm GPDK accordingly. The estimated reduction ratios for SEAM in the experimental environment of the referenced paper are 2% for delay, 45% for power, and 70% for area. These ratios correspond to the 11th, 3rd, and 4th positions among the 16 approximate multipliers tested in that study. Despite the relatively high reduction ratios, the error metrics of SEAM, compared to those with higher reduction ratios, show that metrics of SEAM are reasonable given the context of the higher MRED observed in other approximate multipliers.

4.2.2. Fuzzy logic based pathfinding algorithm

To validate the feasibility of SEAM for fuzzy logic, a pathfinding algorithm was adopted [44,45]. This algorithm was proposed for four-wheel-drive autonomous mobile robots and encompasses all components of fuzzy logic. In fuzzy logic, typically all arithmetic operations are included. Therefore, to compare error-related metrics, not only approximate multipliers but also approximate dividers were applied, and the results were comprehensively analyzed. The approximate dividers used include a 32-bit FXU, [43] with k values of 4 and 8, and an approximate divider applying the goldschmidt method with Look-up table (LUT). We developed a Python simulator replicating the functionality of the pathfinding algorithm with approximate multiplier and approximate dividers. A total of 46,656 input data points were utilized to analyze and Table 5 shows the values of error-related metrics included MAE, MSE, RMSE, MRED, MAPE, MPE, WC-Error, and WCR-Error.

Table 5 showed significant variations in metrics based on the type of divider used. This indicates that errors in the fuzzification at the beginning of fuzzy logic have a substantial impact on the entire system. The SEAM consistently demonstrated lower error metric values compared to DRUM6. When used with FXU divider 32 × 32, it showed lower error-related metric values for all metrics except MPE. Moreover, when compared to DRUM8, SEAM along with some approximate divider, exhibited lower values for some metrics. When error-related metrics were compared overall, the comprehensive metric of DRUM16 was found to be the most superior. However, when considering the energy efficiency of the circuit and conducting a comprehensive evaluation, SEAM exhibited the most superior error-related metrics among circuits with similar energy efficiency. Therefore, the feasibility of SEAM has been validated.

Table 4
Classification metrics of CNN, DRUM-CNN, SEAM-CNN on the CIFAR-10 dataset.

Cases	Accuracy [%]			Precision [%]			Recall [%]			F1 score [%]		
	Exact	DRUM	SEAM	Exact	DRUM	SEAM	Exact	DRUM	SEAM	Exact	DRUM	SEAM
Case A	80.1	80.0	80. 1	80.1	80.0	80.1	80.1	80.0	80.1	80.1	80.0	80.1
Case B	70.4	70.8	70.2	73.4	73.7	73.4	70.3	70.7	70.1	71.8	72.2	71.7
Case C	94.3	94.2	94.0	94.3	94.2	94.0	94.3	94.2	94.0	94.3	94.2	94.0
Case D	85.3	84.8	85.0	85.6	85.2	85.5	85.5	88.8	88.6	85.6	85.1	85.4
Case E	89.3	88.9	88.7	89.6	89.2	89.1	89.2	88.8	88.6	89.4	89.0	88.9
Case F	71.5	72.1	71.7	72.4	72.9	72.5	71.4	72.0	71.6	71.9	72.4	72.0
Case G	97.4	97.4	97.3	97.4	97.4	97.3	97.3	97.3	97.2	97.4	97.4	97.3

Table 5
Error-related metrics when applying various approximate multipliers and approximate dividers to the fuzzy logic-based pathfinding algorithm.

Divider	Multiplier	MAE		RMSE		MAPE		MPE		WCR-Error	
		%	Ratio	%	Ratio	%	Ratio	%	Ratio	%	Ratio
FXU 32 × 32	DRUM6	1.09	4.74	1.65	4.56	10.02	1.89	−58.11	8.16	3.92	1.55
	DRUM8	0.4	1.74	0.62	1.70	6.82	1.29	−7.12	**1.00**	2.54	1.01
	DRUM16	0.23	**1.00**	0.36	**1.00**	5.29	**1.00**	9.81	−1.38	2.52	**1.00**
	SEAM	0.78	3.39	1.06	2.94	9.13	1.73	−67.32	9.46	3.52	1.40
[43] (k = 4)	DRUM6	4.52	1.02	6.31	1.04	20.78	1.24	279.36	**1.00**	6.27	**1.00**
	DRUM8	4.86	1.10	6.71	1.10	19.71	1.18	367.10	1.31	8.04	1.28
	DRUM16	4.95	1.12	6.90	1.13	19.49	1.16	−397.50	−1.42	8.16	1.30
	SEAM	4.42	**1.00**	6.08	**1.00**	16.73	**1.00**	298.66	1.07	8.02	1.28
[43] (k = 8)	DRUM6	1.77	1.79	2.81	1.81	18.59	1.80	−146.83	−6.07	10.20	1.47
	DRUM8	0.99	**1.00**	1.55	**1.00**	10.73	1.04	−48.77	−2.01	7.65	1.10
	DRUM16	1.05	1.06	1.56	1.01	10.32	**1.00**	24.21	**1.00**	6.93	**1.00**
	SEAM	1.39	1.40	2.18	1.41	16.12	1.56	118.14	4.88	10.33	1.49
Gold. + LUT	DRUM6	1.13	4.19	1.63	4.05	10.40	1.94	57.33	14.16	3.94	1.57
	DRUM8	0.43	1.59	0.63	1.58	6.30	1.18	9.87	2.44	2.53	1.01
	DRUM16	0.27	**1.00**	0.40	**1.00**	5.36	**1.00**	4.05	**1.00**	2.51	**1.00**
	SEAM	0.82	3.04	1.13	2.80	8.34	1.56	63.11	15.59	3.41	1.36

5. Conclusion

In this paper, we propose SEAM, a combination of two approximate multipliers, DRUM, and AWTM. SEAM effectively reduces the worst-case error factor of AWTM, minimizing the circuit size while maintaining the values of error-related metrics. The proposed approximate multiplier was designed in Verilog HDL, and the feasibility of the circuit was validated through simulation and FPGA implementation.

When synthesized using Synopsys 32 nm GPDK, the proposed approximate multiplier exhibits circuit areas of 2,064 μm^2. These figures represent a reduction of 80.46% compared to accurate multipliers. Additionally, a power consumption decrease of 82.6% was achieved.

The applicability of the approximate multiplier varies depending on the application systems due to the different WC-Error of the multiplier. Therefore, we verified the feasibility of the proposed approximate arithmetic multiplier for two application systems. First, we applied SEAM to the multiplication operation in the CNN accelerator and analyzed the results. The experiments on various scenarios of CIFAR-10 and MNIST, SEAM even showed higher values for precision, recall, and F1 score in some cases. Next, when applying the proposed circuits to the fuzzy logic based pathfinding algorithm, SEAM was demonstrated superior performance in most error-related metrics compared to DRUM6. This is a meaningful result, considering the similar circuit areas of DRUM6 and SEAM. Furthermore, when compared to DRUM8, SEAM showed similar results in some metrics, indicating the minimal impact of worst case occurrence despite the reduced circuit area due to the application of SEAM.

In conclusion, the proposed approximate arithmetic multiplier is not only applicable to embedded fuzzy logic controller but also to other application system. It is expected to be particularly efficient in systems where some level of error tolerance is acceptable.

CRediT authorship contribution statement

Youngwoo Jeong: Conceived the study, Designed the experiments, Writing – original draft, Writing – review & editing. **Joungmin Park:** Conducted the experiments, Data collection and analysis, Execution of the experiments, Writing – original draft, Writing – review & editing. **Raehyeong Kim:** Writing – original draft, Writing – review & editing. **Seung Eun Lee:** Supervised the project, Provided funding, Approved the final manuscript, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

References

[1] D. Amuru, A. Zahra, H.V. Vudumula, P.K. Cherupally, S.R. Gurram, A. Ahmad, Z. Abbas, AI/ML algorithms and applications in VLSI design and technology, Integration 93 (2023) 102048.

[2] Y. Ma, Z. Wang, H. Yang, L. Yang, Artificial intelligence applications in the development of autonomous vehicles: a survey, IEEE/CAA J. Autom. Sin. 7 (2) (2020) 315–329.

[3] P. Jiang, D. Ergu, F. Liu, Y. Cai, B. Ma, A review of yolo algorithm developments, Procedia Comput. Sci. 199 (2022) 1066–1073.

[4] E. Talpes, D.D. Sarma, G. Venkataramanan, P. Bannon, B. McGee, B. Floering, A. Jalote, C. Hsiong, S. Arora, A. Gorti, et al., Compute solution for tesla's full self-driving computer, IEEE Micro 40 (2) (2020) 25–35.

[5] M.A. Talib, S. Majzoub, Q. Nasir, D. Jamal, A systematic literature review on hardware implementation of artificial intelligence algorithms, J. Supercomput. 77 (2) (2021) 1897–1938.

[6] N. Gupta, Chapter one - introduction to hardware accelerator systems for artificial intelligence and machine learning, in: S. Kim, G.C. Deka (Eds.), Hardware Accelerator Systems for Artificial Intelligence and Machine Learning, in: Advances in Computers, vol. 122, Elsevier, 2021, pp. 1–21.

[7] S. Venkataramani, V. Srinivasan, W. Wang, S. Sen, J. Zhang, A. Agrawal, M. Kar, S. Jain, A. Mannari, H. Tran, et al., RaPiD: AI accelerator for ultra-low precision training and inference, in: 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture, ISCA, IEEE, 2021, pp. 153–166.

[8] A. Skillman, T. Edso, A technical overview of cortex-m55 and ethos-u55: Arm's most capable processors for endpoint ai, in: 2020 IEEE Hot Chips 32 Symposium, HCS, IEEE Computer Society, 2020, pp. 1–20.

[9] T. Khurshid, V. Singh, Energy efficient design of unbalanced ternary logic gates and arithmetic circuits using CNTFET, AEU-Int. J. Electron. Commun. 163 (2023) 154601.

[10] S. Choi, J. Yang, G. Wang, Emerging memristive artificial synapses and neurons for energy-efficient neuromorphic computing, Adv. Mater. 32 (51) (2020) 2004659.

[11] Y. Jeong, H.W. Oh, S. Kim, S.E. Lee, An edge AI device based intelligent transportation system, 2022.

[12] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, N. Sebe, Binary neural networks: A survey, Pattern Recognit. 105 (2020) 107281.

[13] Y.H. Yoon, D.H. Hwang, J.H. Yang, S.E. Lee, Intellino: Processor for embedded artificial intelligence, Electronics 9 (7) (2020) 1169.

[14] H. Baba, T. Yang, M. Inoue, K. Tajima, T. Ukezono, T. Sato, A low-power and small-area multiplier for accuracy-scalable approximate computing, in: 2018 IEEE Computer Society Annual Symposium on VLSI, ISVLSI, IEEE, 2018, pp. 569–574.

[15] G. Zervakis, S. Xydis, K. Tsoumanis, D. Soudris, K. Pekmestzi, Hybrid approximate multiplier architectures for improved power-accuracy trade-offs, in: 2015 IEEE/ACM International Symposium on Low Power Electronics and Design, ISLPED, IEEE, 2015, pp. 79–84.

[16] J. Kim, W.S. Jeong, Y. Jeong, S.E. Lee, Parallel stochastic computing architecture for computationally intensive applications, Electronics 12 (7) (2023) 1749.

[17] A. Ghosh, A. Raha, A. Mukherjee, Energy-efficient IoT-health monitoring system using approximate computing, Internet Things 9 (2020) 100166.

[18] K. Roy, Approximate computing for energy-efficient error-resilient multimedia systems, in: 2013 IEEE 16th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, DDECS, IEEE Computer Society, 2013, pp. 5–6.
[19] K. Roy, A. Raghunathan, Approximate computing: An energy-efficient computing technique for error resilient applications, in: 2015 IEEE Computer Society Annual Symposium on VLSI, IEEE, 2015, pp. 473–475.
[20] V. Pejović, Towards approximate mobile computing, GetMobile: Mob. Comput. Commun. 22 (4) (2019) 9–12.
[21] S. Froehlich, D. Große, R. Drechsler, One method-all error-metrics: a three-stage approach for error-metric evaluation in approximate computing, in: 2019 Design, Automation & Test in Europe Conference & Exhibition, DATE, IEEE, 2019, pp. 284–287.
[22] C. Chen, J. Twycross, J.M. Garibaldi, A new accuracy measure based on bounded relative error for time series forecasting, PLoS One 12 (3) (2017) e0174202.
[23] C. Xu, X. Wu, W. Yin, Q. Xu, N. Jing, X. Liang, L. Jiang, On quality trade-off control for approximate computing using iterative training, in: Proceedings of the 54th Annual Design Automation Conference 2017, 2017, pp. 1–6.
[24] Y. Mannepalli, V.B. Korede, M. Rao, Novel approximate multiplier designs for edge detection application, in: Proceedings of the 2021 on Great Lakes Symposium on VLSI, 2021, pp. 371–377.
[25] S. Froehlich, D. Große, R. Drechsler, Approximate hardware generation using symbolic computer algebra employing grobner basis, in: 2018 Design, Automation & Test in Europe Conference & Exhibition, DATE, IEEE, 2018, pp. 889–892.
[26] Y. Wang, J. Deng, Y. Fang, H. Li, X. Li, Resilience-aware frequency tuning for neural-network-based approximate computing chips, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 25 (10) (2017) 2736–2748.
[27] Q. Zhang, Q. Xu, Approxit: A quality management framework of approximate computing for iterative methods, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (5) (2017) 991–1002.
[28] S. Abed, Y. Khalil, M. Modhaffar, I. Ahmad, High-performance low-power approximate wallace tree multiplier, Int. J. Circuit Theory Appl. 46 (12) (2018) 2334–2348.
[29] S. Venkatachalam, H.J. Lee, S.-B. Ko, Power efficient approximate booth multiplier, in: 2018 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2018, pp. 1–4.
[30] S. Hashemi, R.I. Bahar, S. Reda, DRUM: A dynamic range unbiased multiplier for approximate applications, in: 2015 IEEE/ACM International Conference on Computer-Aided Design, ICCAD, IEEE, 2015, pp. 418–425.
[31] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, J. Han, A comparative evaluation of approximate multipliers, in: 2016 IEEE/ACM International Symposium on Nanoscale Architectures, NANOARCH, IEEE, 2016, pp. 191–196.
[32] Y. Xiang, L. Li, S. Yuan, W. Zhou, B. Guo, Metrics, noise propagation models, and design framework for floating-point approximate computing, IEEE Access 9 (2021) 71039–71052.
[33] C.K. Jha, K. Prasad, V.K. Srivastava, J. Mekie, FPAD: a multistage approximation methodology for designing floating point approximate dividers, in: 2020 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2020, pp. 1–5.
[34] M.S. Lau, K.-V. Ling, Y.-C. Chu, Energy-aware probabilistic multiplier: design and analysis, in: Proceedings of the 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, 2009, pp. 281–290.
[35] Q. Zhang, T. Wang, Y. Tian, F. Yuan, Q. Xu, Approxann: An approximate computing framework for artificial neural network, in: 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE, IEEE, 2015, pp. 701–706.
[36] Z. Peng, X. Chen, C. Xu, N. Jing, X. Liang, C. Lu, L. Jiang, AXNet: ApproXimate computing using an end-to-end trainable neural network, in: Proceedings of the International Conference on Computer-Aided Design, 2018, pp. 1–8.
[37] S. Sen, S. Venkataramani, A. Raghunathan, Approximate computing for spiking neural networks, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, IEEE, 2017, pp. 193–198.
[38] S. Venkataramani, X. Sun, N. Wang, C.-Y. Chen, J. Choi, M. Kang, A. Agarwal, J. Oh, S. Jain, T. Babinsky, et al., Efficient AI system design with cross-layer approximate computing, Proc. IEEE 108 (12) (2020) 2232–2250.
[39] Z. Du, A. Lingamneni, Y. Chen, K.V. Palem, O. Temam, C. Wu, Leveraging the error resilience of neural networks for designing highly energy efficient accelerators, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 34 (8) (2015) 1223–1235.
[40] Y. Wu, C. Chen, W. Xiao, X. Wang, C. Wen, J. Han, X. Yin, W. Qian, C. Zhuo, A survey on approximate multiplier designs for energy efficiency: From algorithms to circuits, ACM Trans. Des. Autom. Electron. Syst. 29 (1) (2024) http://dx.doi.org/10.1145/3610291.
[41] J. Park, S. An, J. Kim, S.E. Lee, Continuous convolution accelerator with data reuse based on systolic architecture, in: 2023 20th International SoC Design Conference, ISOCC, 2023, pp. 319–320.
[42] J. Park, J. Shin, R. Kim, S. An, S. Lee, J. Kim, J. Oh, Y. Jeong, S. Kim, Y.R. Jeong, S.E. Lee, Accelerating strawberry ripeness classification using a convolution-based feature extractor along with an edge AI processor, Electronics 13 (2) (2024).
[43] S. Hashemi, R.I. Bahar, S. Reda, A low-power dynamic divider for approximate applications, in: Proceedings of the 53rd Annual Design Automation Conference, DAC '16, Association for Computing Machinery, New York, NY, USA, 2016.
[44] Y. Jeong, W.S. Jeong, J.Y. Shin, S.E. Lee, The design of embedded fuzzy logic controller for autonomous mobile robots, in: 2023 20th International SoC Design Conference, ISOCC, 2023, pp. 145–146.
[45] Y.W. Jeong, K.H. Go, S.E. Lee, Robot-on-chip: Computing on a single chip for an autonomous robot, in: 2022 IEEE International Conference on Consumer Electronics, ICCE, 2022, pp. 1–3.