IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 10, 2020, accepted December 28, 2020, date of publication January 5, 2021, date of current version January 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3049299

CNN Inference Using a Preprocessing Precision Controller and Approximate Multipliers With Various Precisions

ISSAM HAMMAD[1], (Graduate Student Member, IEEE), LING LI[1], KAMAL EL-SANKARY[1], (Member, IEEE), AND W. MARTIN SNELGROVE[2], (Member, IEEE)

[1]Microelectronics and VLSI Research Laboratory, Department of Electrical and Computer Engineering, Dalhousie University, Halifax, NS B3J 2X4, Canada
[2]Untether AI, Toronto, ON M5V 2H2, Canada

Corresponding author: Issam Hammad (issam.hammad@dal.ca)

This work was supported in part by the Killam Trust Scholarship, and in part by the Natural Sciences and Engineering Research Council of Canada.

ABSTRACT This article proposes boosting the multiplication performance for convolutional neural network (CNN) inference using a precision prediction preprocessor which controls various precision approximate multipliers. Previously, utilizing approximate multipliers for CNN inference was proposed to enhance the power, speed, and area at a cost of a tolerable drop in the accuracy. Low precision approximate multipliers can achieve massive performance gains; however, utilizing them is not feasible due to the large accuracy loss they cause. To maximize the multiplication performance gains while minimizing the accuracy loss, this article proposes using a tiny two-class precision controller to utilize low and high precision approximate multipliers hybridly. The performance benefits for the proposed concept are presented for multi-core multi-precision architectures and single-core reconfigurable architectures. Additionally, a design for a merged reconfigurable approximate multiplier with two precisions is proposed for utilization in single-core architectures. For performance comparison, several segments-based approximate multipliers with different precisions were synthesized using CMOS 15nm technology. For accuracy evaluation, the concept was simulated on VGG19, Xception, and DenseNet201 using the ImageNetV2 dataset. This article will demonstrate that the proposed concept can achieve significant performance gains with a minimal accuracy loss when compared to designs that utilize exact multipliers or single-precision approximate multipliers.

INDEX TERMS Approximate computing, approximate multiplier, CNN accelerator, deep learning, reconfigurable approximate multiplier, precision prediction.

I. INTRODUCTION

Approximate computing is emerging as a viable way to achieve significant performance enhancement in terms of power, speed, and area for computationally heavy digital system on chip (SoC) designs [1]–[5]. Even though a significant performance enhancement can be achieved using approximate computing, these techniques have the obvious cost of certain levels of inaccuracy in the output. However, for large systems, what matters is the impact of the approximate computing on the accuracy of the entire system and not on each sub-module that resides within the system. Approximate multipliers are one of the most common operators for approximate computing. These multipliers produce an approximated output for the multiplication which contains a certain inaccuracy, However, they can achieve significant performance gains in terms of power, speed, and area compared to exact multipliers when utilized in SoC design. Improving the performance by increasing the speed and lowering the power allows for reducing the energy consumption per operation. Several approximate multiplier designs have been proposed in the literature such as [6]–[12].

Image recognition using deep learning [13] has been booming in the last few years. Using fixed-point arithmetic to improve the performance of convolutional neural network (CNN) accelerators was proposed by the industry as can be seen in the articles published by Qualcomm in [14] and IBM in [15]. Additionally, several application-specific integrated

The associate editor coordinating the review of this manuscript and approving it for publication was Tallha Akram.

circuit (ASIC) designs for fixed-point CNN accelerators have been proposed in the literature such as [16]–[19]. Using a 16-bit base for the design of ASIC CNN accelerators is common as can be seen in [16]–[18]. Additionally, several field-programmable gate array (FPGA) designs for 16-bit based fixed-point CNN accelerators have been proposed such as [20]–[22].

Based on the architectures in [16]–[22], CNN accelerators are designed using arrays of processing elements (PEs), at the core of each PE, a multiply and accumulate (MAC) unit exists. Hence, the multiplier is a primary component in the design of CNN accelerators, and any improvement in the performance of the multiplier will scale up to improve the performance of the entire accelerator. The focus of CNN accelerators has been on improving the inference performance. This is because CNN training is usually done once using powerful graphics processing units (GPUs), following that inference is performed thousands or even millions of times before a model update or retraining is required.

The utilization of approximate multipliers in the hardware design of convolutional neural networks (CNNs) has been proposed previously to enhance the performance in terms of power, speed, and area [23]–[28]. Moreover, using a reconfigurable approximate multiplier based on calculating the error variance was proposed in [29]. Lower precision approximate multipliers can achieve higher performance gains as can be seen in [23]–[27]. However, this performance enhancement has a cost of a drop in the CNN accuracy which is inversely proportional to the precision. This creates a trade-off in terms of how much performance gains can be achieved vs. how much accuracy can be sacrificed. Hence, utilizing low precision approximate multipliers might not be feasible when the accuracy loss is large. Based on the challenge that this trade-off presents, this article proposes the concept of predicting and dynamically configuring the precision of approximate multipliers for CNN inference. This article will demonstrate that the impact of approximate multipliers' precision on the inference accuracy varies widely between the different image classes. An image class contains a group of images that belong to the same category. (e.g: hen, bee, zebra . . . etc). For certain image classes, the CNN achieves a lower accuracy when lowering the approximate multiplier precision, for other image classes the CNN accuracy stays constant, and interestingly, in other smaller percentages of image classes, the CNN achieves higher accuracy with lower precision approximate multipliers. Reaching this finding was accomplished experimentally, were utilizing lower precision approximate multiplier results in a more optimal CNN solution for certain image classes. Accordingly, the image classes in a dataset can be divided into two precision categories, a low precision category which contains image classes that can be predicted with the same CNN accuracy or better using lower precision approximate multipliers, and a high precision category which contains the rest of the image classes.

To predict the adequate processing precision for each image as low or high, this article proposes using a tiny

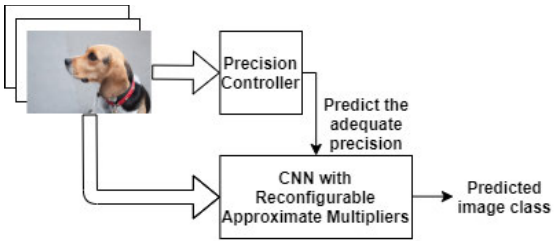


FIGURE 1. High-level demonstration of the proposed concept.

two-class CNN preprocessing precision controller. The controller can be utilized in a system that contains multiple approximate-multiplier based CNN inference accelerators with different precisions, or in a single CNN inference accelerator built with precision reconfigurable approximate multipliers. The controller's objective is to maximize the overall performance gains by maximizing the usage of lower precision approximate multipliers whenever this does not cause an additional accuracy loss. This article proposes a methodology which augments the performance of CNN inference using the concept of the precision controller to enable the utilization of existing low precision and high precision approximate multiplier hybridly. It also proposes a new precision reconfigurable approximate multiplier to utilize the precision controller concept in single-core designs. Fig. 1 illustrates a high-level design for the proposed concept which includes a preprocessing precision controller which controllers a CNN network complied on inference accelerators with reconfigurable approximate multipliers.

To demonstrate the performance gains of the proposed concept in terms of power, speed, and area, several approximate multiplier designs using the static segment method (SSM) and the dynamic segment method (DSM) with different precisions were synthesized using CMOS 15nm technology. For CNN accuracy analysis and comparison, Keras [30] was used to simulate the proposed design on VGG19 [31], Xception [32], and DenseNet201 [33] using ImageNetV2 TopImages dataset [34]. Two architectures for utilizing the proposed concept are presented in this article. The first is a multi-core architecture that uses approximate-multiplier based CNN inference accelerators with various precisions, and the second is using a single-core accelerator built with precision reconfigurable approximate multipliers. To enable the utilization of the proposed concept in single-core designs, a new merged reconfigurable approximate multiplier with two precisions is proposed. This is an additional research contribution that the article presents. Using both architectures, this article will demonstrate that the proposed concept can achieve significant performance enhancements with minimal accuracy loss compared architectures with 16-bit exact signed multipliers or a single-precision approximate multiplier. All simulations were based on a 16-bit representation as it is common in CNN accelerators as can be seen in [16]–[18] and [20]–[22].

This article is structured as follows: In section II the architecture of the segment based approximate multipliers

using the static segment method (SSM) and the dynamic segment method (DSM) is presented. Section III presents the concept of training the precision prediction preprocessing controller. Section IV presents the baseline performance and accuracy simulation SSM and DSM approximate multipliers with various segment sizes using VGG19, Xception, and DenseNet201. Section V illustrates how the proposed concept of using a precision controller with reconfigurable approximate multipliers can be utilized in both multi-core and single-core architectures. Section VI concludes the research findings of this article.

II. SEGMENT BASED APPROXIMATE MULTIPLIERS

Several approximate multipliers techniques are proposed in the literature such Segmentation, High Radix, Rounding, and Perforation [35]. The proposed concept of using a pre-processing precision controller can be applied to any approximate multiplier technique with controllable precision. Nevertheless, segment-based approximate multipliers using both SSM and DSM were selected for the simulation to present the precision controller concept in this article. This provides a comparison between a high precession multiplier (the DSM) and lower power and area one (the SSM). The DSM based multiplier using DRUM's design [7] can provide notably high accuracy, although it has a larger area and energy consumption compared to other multipliers such as [6], [8], and [9]. On the other hand, the SSM based multiplier [6] has a more efficient circuit compared to other approximate multipliers such as [7]–[9]. Moreover, the segment-based technique allows of an efficient implementation for the merged reconfigurable approximate multiplier which is presented in this article. Segment based approximate multiplication is one of the efficient and simple techniques used to design approximate multipliers. Using this technique, only a segment of the multiplicand is passed to the multiplier to approximate the multiplication. This allows for the multiplication of $n \times n$ number using an $m \times m$ multiplier, where $m < n$. As an example, a 16×16 bit multiplication can be approximated using an 8×8 bit multiplier with a byte segment or even 4×4 bit multiplier with a nibble segment. Controlling the precision of these multipliers is performed by adjusting segment size *(m)*. Fig. 2 demonstrates the general design of a segment based approximate multiplier for an $n \times n$ number.

Several techniques for the segment selection have been proposed in the literature, this includes the segment static segment method (SSM) which was adopted in the approximate multiplier design Narayanamoorthy *et al.* in [6] and the dynamic segment method (DSM) which was also proposed by [6] and was implemented in the approximate multiplier design DRUM in [7]. Based on [6], [7], an inversely proportional relationship exists between the segment size and the achieved performance gains.

Using an SSM segmenter, an *n-bit* integer number is divided into a static *i (m-bit)* segments with a (k) offset between the starting bits of two consecutive segments. The *m-bit* segment with the leading one is selected to approximate the multiplication. Fig 3. demonstrates the circuit implementation of the SSM segmenter. As can be seen from the Fig 3., the SSM segmenter consists of a multiplexer (MUX) to select between the (i) segment. On the selection lines, the *m-bits* of each segment are passed through an OR gate.

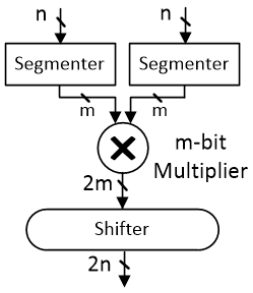


FIGURE 2. Segment based approximate multiplier.

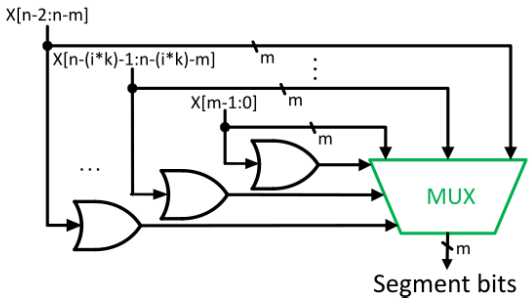


FIGURE 3. The SSM segmenter.

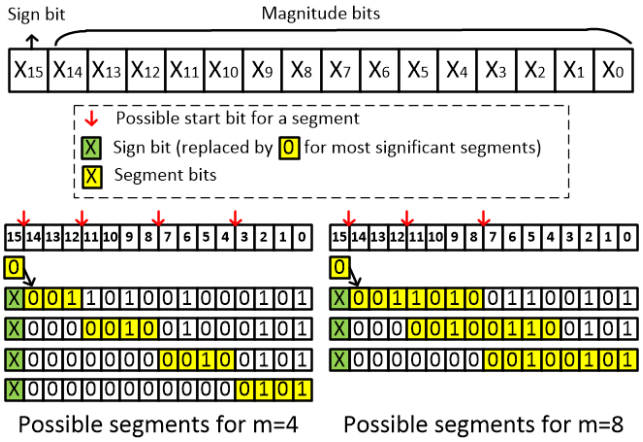


FIGURE 4. SSM segmentation example using k = 4.

Fig 4. illustrates an example of the possible segments for SSM using $k = 4$ for with segment sizes of $m = 8$ and $m = 4$. The example shows the segmentation based on a signed 16-bit integer number. This slightly differs compared to the presented unsigned format in [6].

Fig 5. shows an example for an SSM based approximate multiplication using $m = 8$. As can be seen in Fig 5., the 16×16 bit number can be estimated using two SSM segmenters, an 8×8 bit multiplier, a shifter, and an XOR for the sign bit. For the most significant segment, the sign bit is replaced by zero to create equal size segments.

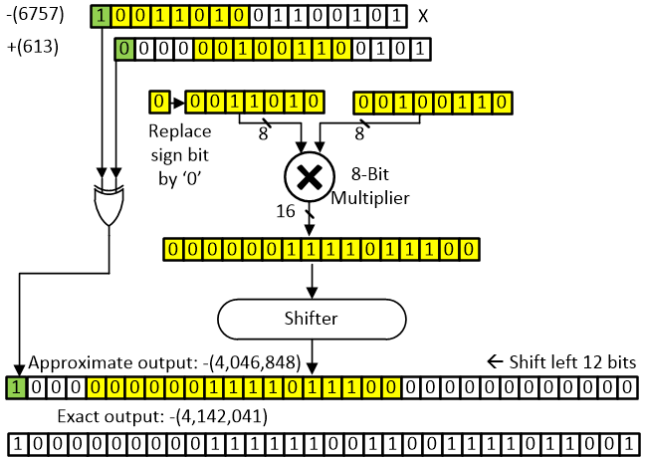


FIGURE 5. SSM approximate multiplication example using m = 8.

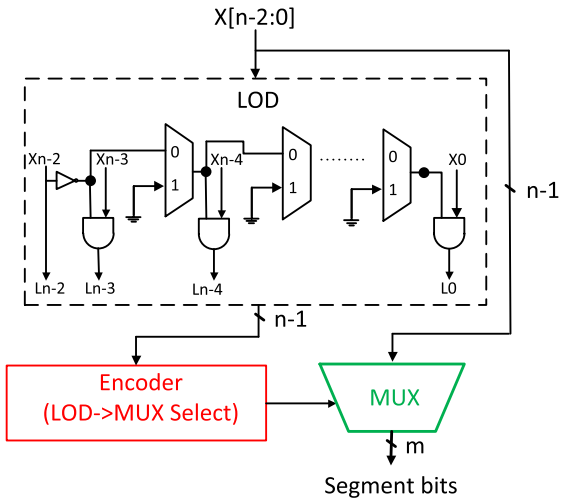


FIGURE 6. The DSM segmenter.

The DSM segmenter is more complicated and costly compared to the SSM segmenter. The DSM segmenter detects the leading one in an *n-bit* number then extracts the following *m-1* bits to enable the utilization of an $m \times m$ *bit* multipliers to approximate $n \times n$ *bit* multiplication. The DSM method as originally proposed by [6] but was further improved in the approximate multiplier DRUM [7]. DRUM's approximate multiplier works by detecting the leading one in the number then extracting the following *m-2* bits, any remaining truncated portion is estimated by setting the segment's least significant bit (LSB) to '1'. Fig 6. shows the circuit implementation for the DSM segmenter. The segmenter consists of a leading one detector (LOD) circuit in addition to an encoder and a MUX.

Fig.7. shows an example of the DSM segmentation based on DRUM for both $m = 8$ *and* $m = 4$. As can be seen in Fig. 7, the segment starts with the leading one, followed by the next *m-2* bits. The segment's LSB is set to '1' to approximate the remaining truncated bits assuming a uniform distribution for the operands [7].

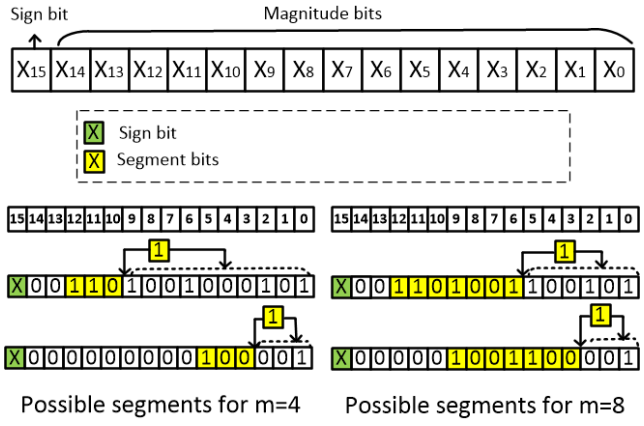


FIGURE 7. DSM segmentation example based on DRUM.

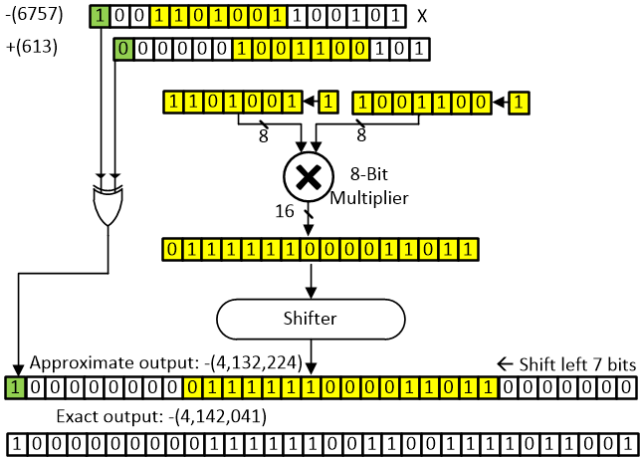


FIGURE 8. DSM approximate multiplication example for m = 8.

Fig. 8 shows an example of the DSM multiplication using $m = 8$. As can be seen in the figure, both 16×16 input numbers are segmented by extracting the leading one followed by the next *m-2* bits while the segments LSB bit is set to '1'. Following the segmentation, an 8×8 bit multiplier is used then the multiplier's output is shifted. For the sign bit, an XOR is used.

III. BUILDING A PRECISION PREPROCESSING CONTROLLER

As previously discussed, utilizing a low precision approximate multiplier in CNN inference will not be feasible if it leads to large accuracy loss despite the massive performance gains it can achieve. As an example, an SSM approximate multiplier with $m = 4$ can achieve a 153% speed increase, an 88% power reduction, and an 82% area reduction compared to an exact multiplier. However, using it in VGG19 inference based on ImageNetV2 causes a 15.3% accuracy loss compared to an exact multiplier as will be seen in the next section. This dilemma has created a motivation for finding a solution that allows for partial utilization of these multipliers only when the accuracy loss is minimal. A solution was found after studying the impact of the approximate

multiplier precision on each image class individually, it was found that the CNN's inference accuracy with low precision approximate multipliers varies widely between image classes. Surprisingly, certain image classes can be classified with higher accuracy using low precision approximate multipliers such as $m = 4$ compared to higher precision approximate multipliers such as $m = 8$. For another set of image classes, there was no difference in CNN classification accuracy between low and high precision approximate multipliers. The remaining larger set of image classes achieves better classification using a higher precision approximate multiplier.

To utilize this important finding in improving the overall performance by maximizing the usage of low precision approximate multipliers while minimizing the cost in terms of accuracy loss, a preprocessing precision controller was developed to predict the adequate precision category for the input image. This allows for the development of a system that contains approximate multipliers with different precisions or precision reconfigurable approximate multipliers. Such a system can maximize performance by utilizing low precision approximate multipliers only when achieving a better or the same CNN classification accuracy is predicted. This preprocessing controller is a two-class tiny CNN network that can predict the adequate approximate multiplier precision as either low or high. The exact segment sizes for what is considered low and what is considered high will depend on how the training was performed.

To train a preprocessing precision controller a labeled dataset with the adequate approximate multiplier precision for each image class must be created for each CNN network. To do that, the CNN predictions for each image using different approximate multiplier precisions should be obtained. Following that, if it was determined that on average an image class can be classified with the same or better accuracy using the lower precision mode, the entire images in this class will be labeled '0'. If using higher precision mode achieves better classification accuracy for that image class, the entire images in that class will be labeled as '1'. Fig. 9 illustrates using a flowchart the process for developing a precision preprocessing controller for the CNN's approximate multipliers. Using the illustrated process in Fig. 9, a precision controller with different precisions (m) was trained for VGG19, Xception, and DenseNet201 using a subset of ImageNet [36]. The training set consisted of 50 images from each image class with a total of 50000 images, while the cross-validation test set consisted of 10 images from each image class with a total of 10000 images. For the final accuracy evaluation, the controller along with the image classification CNN networks were tested using ImageNetV2 Top-Images dataset which consists of 10000 images using both SSM and DSM.

Fig. 10 demonstrates the architecture of the preprocessing precision controller. The controller is a tiny CNN that has two output nodes with Softmax activation and contains three 2D convolution layers. These layers have shapes of (224,224,3), (56,56,6), and (14,14,12) consecutively. Following each convolutional layer, a (4×4) max-pooling layer exists. ReLU [37] activation was used in all the layers. The controller was designed to have a minimal overhead in terms of the number of parameters and the multiply and accumulate (MAC) operations. Therefore, the smallest possible design with the least parameters but with an acceptable accuracy was selected. Table 1 details the controller overhead compared to VGG19, Xception, and DenseNet201.

As can be seen in Table 1, the controller parameters overhead is as low as 0.008% in the case of VGGNet19 and

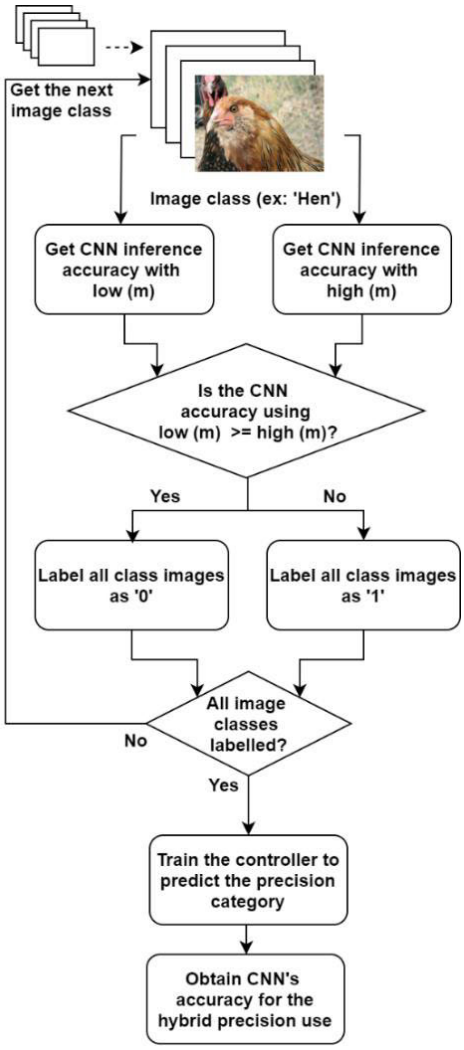


FIGURE 9. Developing a precision controller flowchart.

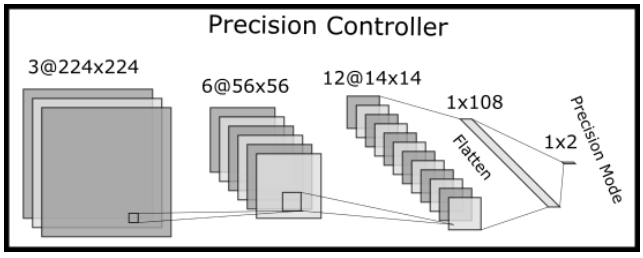


FIGURE 10. A proposed tiny CNN precision controller.

TABLE 1. Precision controller overhead comparison.

Network	Total Param.	Total MACs	Controller Param. Overhead	Controller MACs Overhead
Controller	1130	4.7M	N/A	N/A
VGG19[31]	143.7M	19.64G	0.0008%	0.0239%
Xception[32]	22.9M	8.404G	0.0049%	0.0559%
DenseNet201[33]	20.2M	3.347G	0.0056%	0.1404%

as high as 0.0056% in the case of DenseNet201. In terms of the MACs, overhead is as low as 0.0239% in the case of VGGNet19 and as high as 0.1404% in the case of DenseNet201. Based on these numbers, the preprocessing precision controller overhead can be considered negligible.

The next section will present a performance comparison between the SSM and the DSM approximate multipliers using $m = 4$, $m = 6$, and $m = 8$ against the exact multiplier. Also, it will present the inference accuracy simulation for using these approximate multipliers using VGG19, Xception, and DenseNet201. Section V will build on section III and section IV to demonstrate how the precision controller can be utilized in multi-core and single-core architectures. Section V will also present the achieved controller accuracy for each network and accordingly, the achieved image classification accuracy with hybrid precision approximate multipliers.

IV. BASELINE PERFORMANCE AND ACCURACY

To demonstrate and compare the performance gains that the SSM and DSM approximate multipliers can achieve, they were implemented using Verilog hardware description language (HDL) using Nangate 15nm FinFET standard cell library [38]. The Verilog implementation was synthesized using Synopsys Design Compiler. The analysis is applied to check the setup violations of the multipliers. The delay-annotated netlists of the multipliers are simulated using Modelism SE to verify their operations. In this implementation, the delay, power, and area using $m = 4$, $m = 6$, and $m = 8$ precisions were obtained. Table 2 details the SSM performance and performance gains compared to an exact multiplier for each precision. Table 3 details the DSM performance and performance gains compared to an exact multiplier for each precision.

As can be seen from Table 2 and Table 3 , both the SSM and DSM can achieve significant performance gains compared to the exact multiplier. These gains are inversely proportional to the segment size *(m)*. The SSM multipliers can achieve higher performance gains compared to the DSM multipliers due to having a simpler segmenter as was presented in Section II.

To evaluate the impact of the SSM and DSM approximate multiplication techniques on the CNN's inference accuracy, a simulation for these techniques were performed using three popular CNN networks, the VGG19, Xception, and DenseNet201. Pretrained models using ImageNet for these CNN networks are available as part of the deep learning platform Keras [30]. For the inference simulation, ImagenetV2 Top-Images [34] dataset was used. ImageNetV2 is a new test set built based on the ImageNet benchmark and was released primarily for inference accuracy evaluation. ImageNetV2 contains 10000 images based on 1000 different classes which are identical to the classes in the ImageNet dataset.

To simulate the impact of the approximate multiplication on the inference accuracy, Keras's custom layer functionality was utilized. This functionality allows for the addition of custom mathematical or logical operations anywhere inside the CNN. By utilizing this functionality, the SSM and DSM segmenters were simulated by creating segmentation layers. These segmentation layers were injected before all convolution layers and fully connected layers. In these layers, the previous layer's outputs were scaled and cast to Int16 numbers, then using bitwise operations, the segmenter was simulated by masking the numbers using a mask which was created based on the segment size (m). For the model weight, all weights were scaled, cast, and masked to simulate the segmentation before starting the inference.

Table 4 lists the achieved CNN inference accuracy for VGG19, Xception, and DenseNet201 using the SSM technique with $m = 4$, $m = 6$, and $m = 8$ precisions. Table 5 lists similar information but by utilizing the DSM technique. In both tables, the approximate multipliers' accuracies are compared against the accuracy of an exact signed Int16 multiplier. In both tables, the accuracy loss compared to an exact multiplier is listed between brackets.

As can be seen in Table 4 and Table 5 , the inference accuracy loss for all simulated networks increases when reducing the precision (m). Nevertheless, the inference accuracy loss using approximate multipliers with $m = 8$ was very minimal. In the case of DSM, the accuracy loss was 0.21% for VGG19, 0.23% for Xception, and 0.59% for DenseNet201. For SSM, the difference was a bit higher with 0.47% for VGG19, 0.62% for Xception, and 1.49% for DenseNet201. Using SSM with a precision of $m = 4$, the accuracy losses were significant where the losses were 15.31%, 18.42%, and 50.45% for VGG19, Xception, and DenseNet201, respectively. The losses were also significant using DSM as can be seen in Table 5.

For both the SSM and the DSM, DenseNet201 had a significant drop in accuracy when the precision was dropped from $m = 6$ to $m = 4$. In the case of SSM, the accuracy loss increased from 4.7% to 50.45%, while for DSM, the loss increased from 3.5% to 46.61%. This eliminated the possibility of utilizing the case of $m = 4$ for DenseNet201 in any hybrid precision design as the loss is very large.

V. USING THE PRECISION PREPROCESSOR TO CONTROL VARIOUS PRECISIONS APPROXIMATE MULTIPLIERS

A. MULTI-CORE ARCHITECTURE

As CNN networks vary in terms of input shape, depth, and the number of filters, ASIC CNN accelerators are designed using a large-scale integration of PEs. This allows the accelerators to be reconfigurable which allows for the compilation of the CNN network that the user wants to deploy. The core component in the PE is the multiplier. This design can be seen in the

TABLE 2. The SSM [6] approximate multiplier performance compared to an exact multiplier.

Design	Delay (ps)	Power (µW)	Area (µm^2)	Speed Increase	Power Reduction	Area Reduction
Exact	330.8	230.1	337.03	N/A	N/A	N/A
m=8	206.06	69.22	108.28	60.54%	69.92%	67.87%
m=6	182.75	52.09	83.705	81.01%	77.36%	75.16%
m=4	130.91	27.66	60.65	152.69%	87.98%	82.00%

TABLE 3. The DSM (Based on DRUM [7]) approximate multiplier performance compared to an exact multiplier.

Design	Delay (ps)	Power (uW)	Area (µm^2)	Speed Increase	Power Reduction	Area Reduction
Exact	330.8	230.1	337.03	N/A	N/A	N/A
m=8	305.26	181.9	204.16	8.37%	20.95%	39.42%
m=6	261.61	133.3	170.289	26.45%	42.07%	49.47%
m=4	199.98	65.63	125.154	65.42%	71.48%	62.87%

TABLE 4. CNN inference accuracy using the SSM [6] approximate multiplication.

Design	VGG-19[31]		Xception[32]		DenseNet-201[33]	
	Top-1 Accuracy	Top-5 Accuracy	Top-1 Accuracy	Top-5 Accuracy	Top-1 Accuracy	Top-5 Accuracy
Exact	73.59%	92.28%	80.78%	95.92%	79.9%	95.59%
m=8	73.12% (-0.47%)	92.06% (-0.62%)	79.88% (-0.90%)	95.11% (-0.81%)	78.41% (-1.49%)	94.38% (-1.21%)
m=6	64.94% (-8.65%)	88.23% (-4.45%)	71.51% (-9.27%)	89.56% (-6.36%)	75.20% (-4.70%)	92.39% (-3.20%)
m=4	58.28% (-15.31%)	85.78% (-6.90%)	62.36% (-18.42%)	85.22% (-10.70%)	29.45% (-50.45%)	62.58% (-33.01%)

TABLE 5. CNN inference accuracy using the DSM (Based on DRUM [7]) approximate multiplication.

Design	VGG-19 [31]		Xception [32]		DenseNet-201 [33]	
	Top-1 Accuracy	Top-5 Accuracy	Top-1 Accuracy	Top-5 Accuracy	Top-1 Accuracy	Top-5 Accuracy
Exact	73.59%	92.28%	80.78%	95.92%	79.9%	95.59%
m=8	73.38% (-0.21%)	92.15% (-0.13%)	80.55% (-0.23%)	95.61% (-0.31%)	79.31% (-0.59%)	95.38% (-0.21%)
m=6	66.86% (-6.73%)	89.39% (-2.89%)	72.13% (-8.65%)	91.06% (-4.86%)	76.40% (-3.50%)	93.54% (-2.05%)
m=4	62.20% (-11.39%)	88.31% (-3.97%)	63.36% (-17.42%)	86.65% (-9.27%)	33.29% (-46.61%)	65.20% (-30.39%)

fixed-point CNN accelerators such as [16]–[22]. In addition to the floating-point accelerators such as [39], [40].

The proposed concept of using a preprocessing precision controller with approximate multiplier-based CNN accelerators can be utilized in a large system such as a cluster of inference accelerators. Such a cluster can contain hundreds or thousands of approximate multiplier-based CNN inference accelerators with various precisions and can be deployed in a data center or in a cloud backbone. Once a CNN model is compiled on the cluster for inference, a small overhead of PEs can be allocated for the deployment of the preprocessing precision controller as well. As was presented in Table 1, the controller overhead is negligible and can as low as 0.008% in the case of VGGNet19 and as high as 0.0.056% in the case of DenseNet201. While the overhead in terms of the MACs is as low as 0.0239% in the case of VGGNet19 and as high as 0.1404% in the case of DenseNet201. If needed, the controller can be mimicked to (i) number of controllers to allow for parallel inference of the compiled model. The controller training can occur as part of the model's compilation on the cluster or prior to that where the controller's parameters and shape can be provided alongside with the model's parameter and shape.

For each compiled network, (n) low precision accelerators (p) high precision accelerators can be allocated. The (n/p) ratio can be determined based on the expected usage of the low vs. high precision for that model. The allocated accelerators can be shared among different complied CNN networks. The cluster resource manager and scheduler will handle the sharing of the various accelerators among the various complied CNN networks. The design of hardware accelerators in data centers is discussed in [41]. Fig. 11 provides a high-level illustration of how preprocessing precision controllers can be used with multiple approximate-multiplier based CNN accelerators with various precisions in a cluster.

To demonstrate the performance benefits of the hybrid use of approximate multipliers with various precisions based on precision prediction, preprocessing precision controllers were trained using a subset of ImageNet for VGG19, Xception, and DenseNet201 by following the process in

TABLE 6. Performance and accuracy for the hybrid use of approximate multipliers with various precisions compared to an exact multiplier.

Network	Seg. Type	Hybrid Precision Modes	Multiplication Delay (ps)	Multiplication Power (μW)	Speed Increase vs. Exact	Power Reduction vs. Exact	CNN Top-1 Accuracy	Accuracy Loss vs. Exact	Precision Controller Accuracy	Low Precision Utilization
VGG19 [31]	SSM	m=4 & 8	174.422	51.723	89.66%	77.52%	72.13%	1.46%	79.4%	42.1%
VGG19 [31]	DSM	m=4 & 8	260.305	132.253	27.08%	42.52%	72.61%	0.98%	80.2%	42.7%
Xception[32]	SSM	m=4 & 8	180.734	55.214	83.03%	76.00%	78.20%	2.58%	78.1%	33.7%
Xception[32]	DSM	m=4 & 8	269.254	142.136	22.86%	38.23%	79.37%	1.41%	78.6%	34.2%
DenseNet201[33]	SSM	m=6 & 8	191.957	58.856	72.33%	74.42%	78.76%	1.14%	77.9%	60.5%
DenseNet201[33]	DSM	m=6 & 8	278.153	151.719	18.93%	34.06%	79.52%	0.38%	78.5%	62.1%

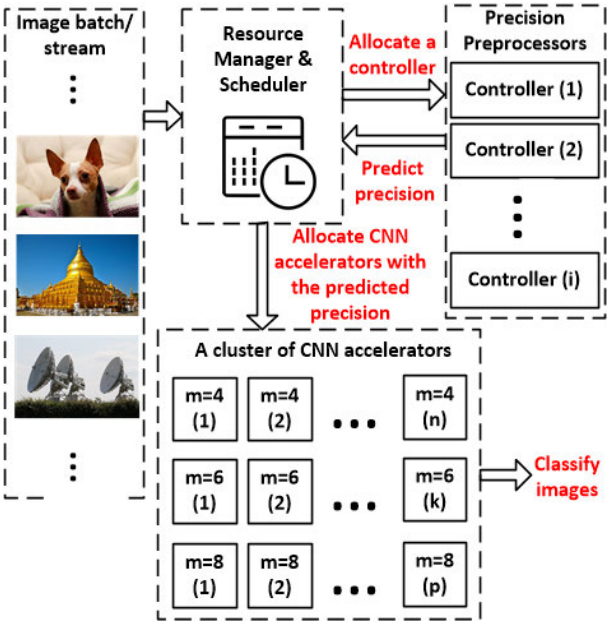


FIGURE 11. Using preprocessing precision controllers with multiple approximate-multiplier based CNN accelerators with various precisions in a cluster.

Fig 9. For each network both, the SSM and the DSM techniques were simulated. Table 6 details the performance gains in terms of power and speed that the hybrid use of approximate multiplier with various precisions can achieve compared to an exact multiplier. For VGG19 and Xception the controller was trained to select either a precision of $m = 4$ or precision of $m = 8$. In the case of the DenseNet201, the lower precision was selected as $m = 6$ because $m = 4$ had extremely low accuracy. As can be seen in Table 6 , using a hybrid of approximate multipliers with various precisions can achieve large performance gains with minimal accuracy loss compared to exact multipliers. Overall, the SSM achieves significantly higher speed and lower power compared to the DSM in all the cases, however, the cost in terms of the network's accuracy loss is higher.

For each simulation in Table 6 , the accuracy of the precision controller is listed as well. As can be seen from the table, the controller's accuracy ranges between 77.9% and 80.2% depending on the network, the segmentation type, and the precision modes. Therefore, the approximate multiplier precision prediction is not ideal. Nevertheless, the accuracy loss and performance gains calculations in Table 6 includes simulating this imperfection in the controller. Table 6 includes the low mode utilization percentage which represents the actual usage of the low precision mode including the false positives. If an ideal controller with 100% accuracy existed, the hybrid precision accuracy will surpass the accuracy of an exact multiplication. That is because a subset of image classes can be classified with better accuracy using lower precision approximate multipliers compared to higher precision approximate multipliers or even exact multipliers.

Using hybrid approximate multipliers with predicted precision can also achieve a better performance-accuracy trade-off compared to using a single-precision approximate multiplier. This is illustrated in Fig. 12 where the performance gains and the accuracy loss compared to the exact multiplier are plotted for both designs. Fig. 12 contains a sub-plot for each scenario listed in Table 6. An example can be seen in Fig. 12 (a), where the hybrid SSM multiplier with $m = (4\&8)$ achieved an 89.66% speed increase and a 77.52% power reduction which is better compared to the performance gains for the case of $m = 6$ with an 81.01% speed increase and a 77.36% power reduction. Nevertheless, the network's accuracy loss was only 1.5% in the case of $m = (4\&8)$ compared to 8.7% in the case of $m = 6$. From the energy perspective, the SSM multiplier with $m = (4\&8)$ can achieve an 88.15% energy reduction compared to 87.49% in the case of $m = 6$. The energy of each multiplier was obtained by calculating the product of the power and the delay.

Another example can be seen in the case of DenseNet201 in Fig. (12) (c) and (f), wherein both the SSM and the DSM, using hybrid approximate multiplier precisions with $m = (6\&8)$ achieved better performance gains with lower accuracy loss compared to the case of $m = 8$. Having a lower accuracy loss in the hybrid configuration might be surprising. However, this is due to having a subset of image classes which can be classified with better accuracy using $m = 6$ compared to $m = 8$ as explained previously.

On the cluster level, an additional performance advantage can be also achieved in terms of the total area. The area is proportional to the approximate multiplier precision as per Table 2 and Table 3. Therefore, building a cluster using (n) accelerators with high precision approximate multipliers

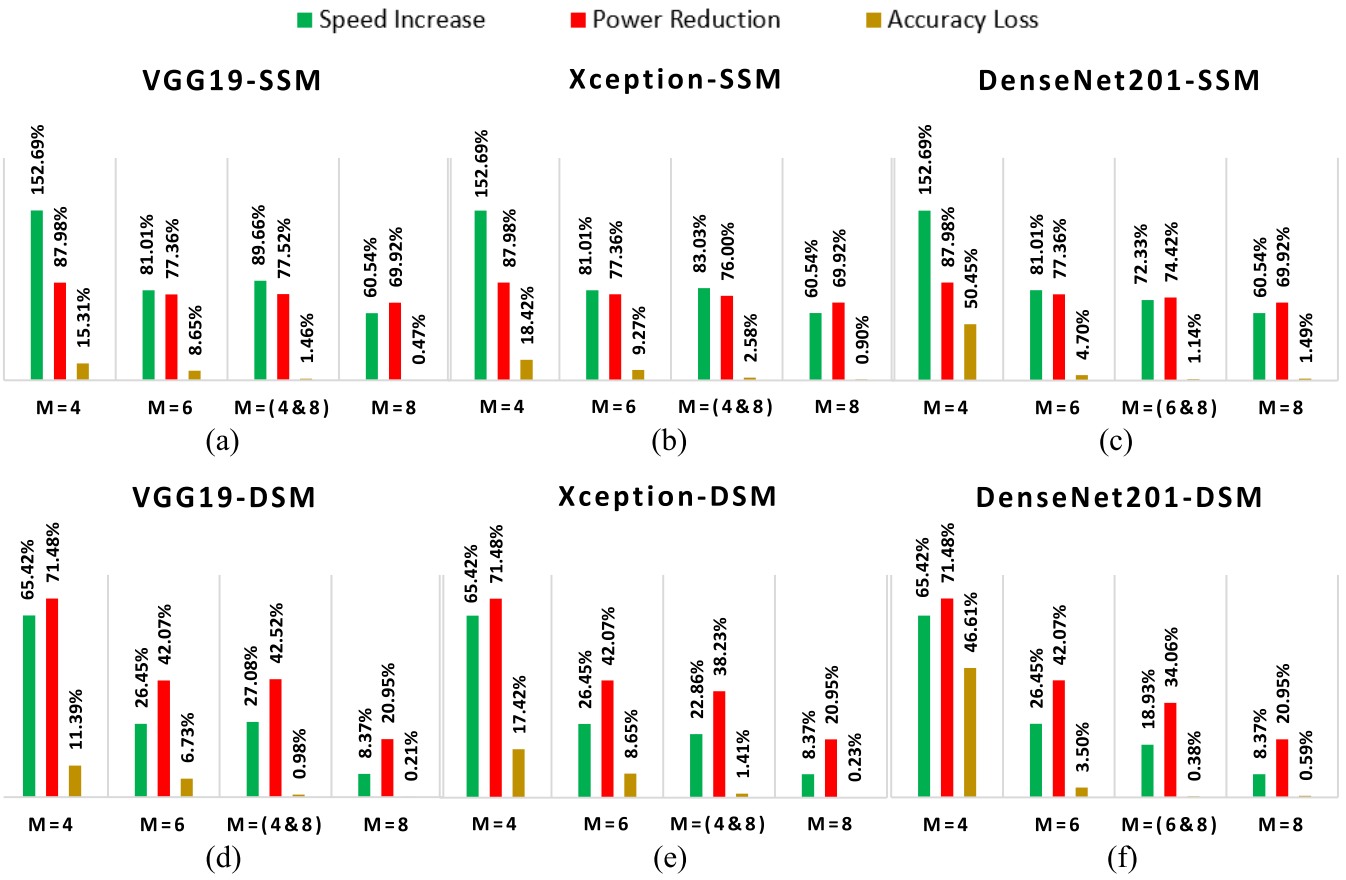


FIGURE 12. Performance gains and accuracy loss using hybrid and single precision approximate multipliers compared to an exact multiplier.

only will require more area than a cluster which contains (n) accelerators built using various approximate multipliers precisions.

B. SINGLE-CORE RECONFIGURABLE ARCHITECTURE

Utilizing the proposed concept of predicting and configuring the precision of approximate-multiplier based CNN accelerators is not limited to the multi-core architecture. This section will present how a reconfigurable approximate multiplier with two precisions can be implemented to allow for the utilization of the concept in single-core systems. This will expand the proposed concept to include the design of approximate multiplier-based CNN accelerators for embedded systems and low power applications such as [42], [43].

Fig 13. demonstrates how a reconfigurable approximate multiplier with two precisions can be implemented inside each MAC unit. Using this design either the high or the low precision approximate multiplier will be power up or down based on the precision controller signal "CN". Tri-state buffers are used to control the multiplier's output flow.

As can be seen from Fig. 13, this architecture requires implementing two approximate multipliers with high and low precisions inside the MAC. This will result in area overhead compared to a single-precision approximate multiplier design. Nevertheless, this hybrid reconfigurable precision design can achieve better performance in terms of speed and power compared to the single-precision design. Also, its total area is still less than an exact multiplier design. Optimizing the design in Fig. 13 can be achieved by designing a merged reconfigurable approximate multiplier that supports two precisions.

Fig. 14 shows a proposed design for a merged reconfigurable approximate multiplier that supports precisions of $m = (4\&8)$. As can be seen from the figure, the merged reconfigurable multiplier can be designed using four (4×4) bit multipliers, segmenters, a shifter, tri-state buffers for flow control, and an XOR for the sign bit. This multiplier will be partially turned off when $m = 4$ is activated using a *VDDxCN* control signal allowing for higher speed and lower power execution. In terms of component sharing, one 4×4 multiplier is shared between the two precision, in addition to the shifter, the merged segmenter, and the sign bit XOR. Fig. 14 illustrates a generic merged reconfigurable multiplier for both the SSM and DSM, the difference between these two techniques is contained in the design of the merged segmenter. Table 7 lists the performance comparison between the merged and the separated reconfigurable design using VGG19 and based on the low precision utilization rate as per Table 6.

TABLE 7. Performance comparison for single-core the separated and merged reconfigurable designs using vgg19.

Hybrid Design Type for m=(4&8)	Seg. Type	Delay (ps)	Power (µW)	Area (μm^2)	Speed Increase vs. Exact	Power Reduction vs. Exact	Area Reduction vs. Exact	Accuracy Loss
Separated	SSM	175.240	52.641	171.100	88.77%	77.12%	49.23%	1.46%
Merged	SSM	181.239	55.223	114.930	82.52%	76.00%	66.20%	1.46%
Separated	DSM	262.312	137.203	332.114	26.11%	40.37%	1.46%	0.98%
Merged	DSM	273.432	146.410	214.980	20.98%	36.37%	36.21%	0.98%

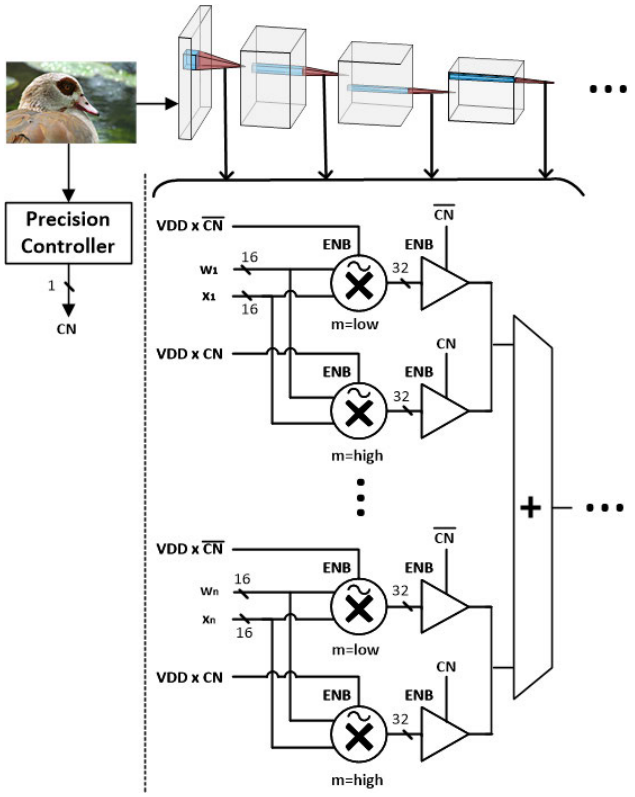


FIGURE 13. Reconfigurable design with two separated approximate multipliers.

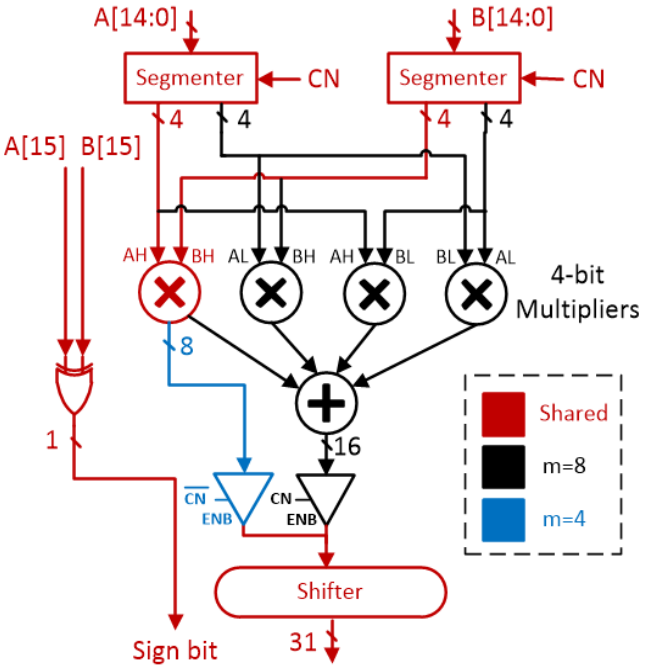


FIGURE 14. Reconfigurable design with a merged approximate multiplier for m = (4&8).

Fig 15. Illustrates the performance-accuracy trade-off for various approximate multiplier precisions compared to an exact multiplier based on VGG19. As can be seen in the figure, the reconfigurable approximate multiplier achieves a better performance-accuracy trade-off compared to single-precision approximate multipliers. The SSM $m = (4\&8)$ merged design achieves an 82.52% speed increase, a 76% power reduction, and a 65.9% area reduction with a 1.46% accuracy loss compared to an exact multiplier. On the other hand, the DSM $m = (4\&8)$ merged design achieves a 20.98% speed increase, a 36.37% power reduction, and a 36.21% area reduction with a 0.98% accuracy loss compared to an exact multiplier. The merged $m = (4\&8)$ reconfigurable designs have a slight area overhead and a slightly higher accuracy loss compared to a single-precision design with $m = 8$. Nevertheless, the performance gains in terms of speed and power are large.

Fig. 16 illustrates the speed and power performance gains of the hybrid $m = (4\&8)$ designs compared to a single-precision design with $m = 8$. As can be seen in the figure, the SSM $m = (4\&8)$ merged design can achieve a 13.7% speed increase and 25.35% power reduction compared to a single-precision design with $m = 8$. For DSM, the gains are an 11.64% increase in speed and a 24.24% power reduction. This is an additional energy savings of 29.83% in the case of merged SSM and 27.9% in the case of merged DSM. The cost in terms of the area overhead is 5.3% for DSM merged design and 6.1% for SSM merged design. While the accuracy loss difference is 0.77% for DSM merged design and 0.99% for SSM merged design.

For cases where reducing the area is less important, the hybrid separated design can be utilized to achieve higher speed and power lower and therefore lower energy consumption. The SSM separated $m = (4\&8)$ design can achieve a higher speed gain and a comparable power reduction compared to the case of $m = 6$ with far less accuracy loss. The SSM separated $m = (4\&8)$ design achieved a speed increase of 88.77% and a power reduction of 77.12% and an accuracy loss of 1.46% which is better than $m = 6$ with a speed increase of 81.01% and 77.36%. and an accuracy loss of 8.65%.

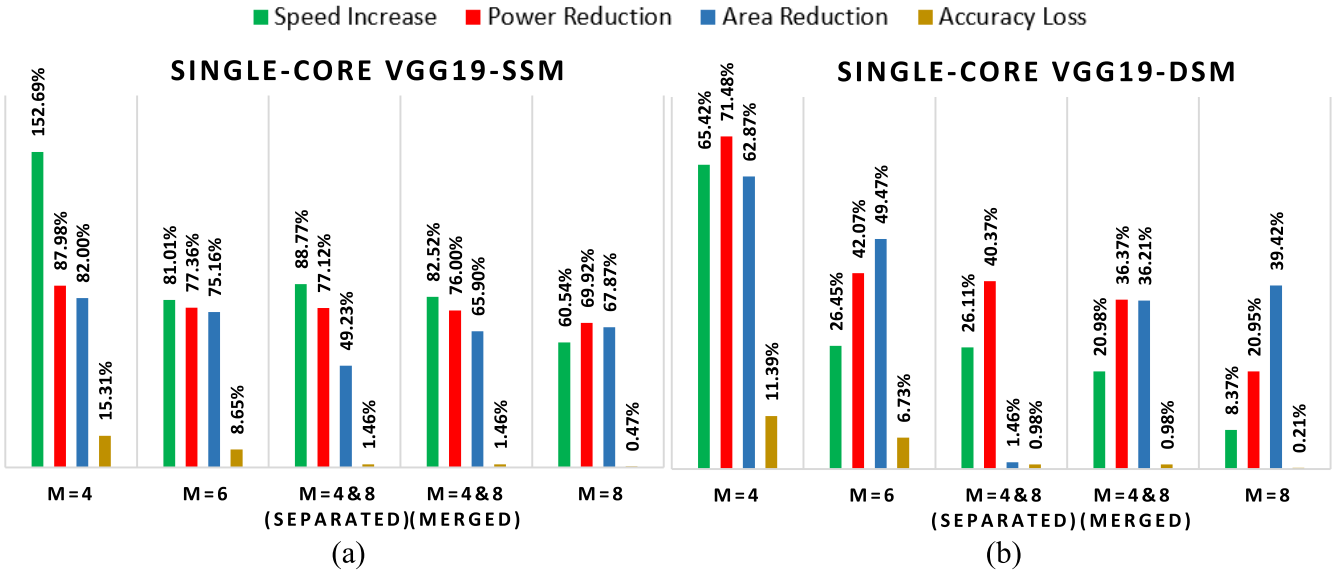


FIGURE 15. The single-core performance-accuracy trade-off for various approximate multiplier precisions compared to an exact multiplier based on VGG19.

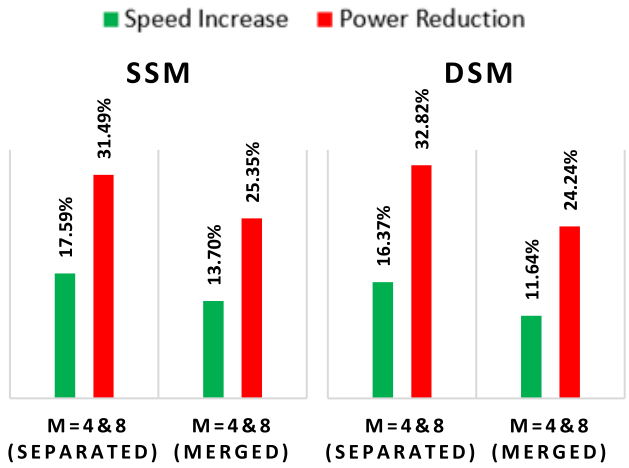


FIGURE 16. Hybrid m = (4&8) performance gains compared to a single-precision with m = 8 based on VGG19.

Overall, the merged hybrid design achieves more balanced power-speed-area gains compared to the separated hybrid design. Nevertheless, the merged design achieves lower gains in terms of the speed increase and the power reduction and therefore in energy saving. Hence, for designs that prioritize increasing the speed or reducing the energy consumption over reducing the area, the separated design might be a better option.

VI. CONCLUSION

This article proposed a new architecture for improving the multiplication performance for CNN inference. The proposed architecture consists of a preprocessing precision controller at the system level and approximate multipliers with various precisions at the PE level. The proposed preprocessing controller determines the adequate precision for the network's approximate multipliers to classify the input image. The precision controller concept was inspired after discovering that a subset of image classes can be classified with the same or better accuracy using lower precision approximate multipliers. Based on this finding, the controller was trained to determine if the input image belongs to that subset or not. The controller is built using a tiny two-class CNN which has a negligible overhead in terms of parameters and MACs compared to the controlled image classification CNN. A performance analysis was presented based on the SSM and DSM methods using CMOS 15nm technology. Additionally, an accuracy analysis using VGG19, Xception, and DenseNet201 was presented. Overall, utilizing the SSM achieves a better performance-accuracy trade-off compared to using the DSM. As an example, for VGG-19 using a multi-core hybrid design with m = (4&8), the SSM achieves 89.66% speed increase and 77.52% power reduction with an accuracy loss of 1.46% compared to a speed increase of 27.08% and power reduction of 42.52% and an accuracy loss of 0.98% in the case of the DSM. Additionally, the total area of two DSM multipliers with m = (4&8) is 94.9% larger than two SSM multipliers with m = (4&8). Using the DSM is an option if minimizing the accuracy loss is critical for the designer. Maximizing the benefits of the proposed design can be achieved in a multi-core architecture with a large number of CNN inference accelerators, where the accelerators are built using different approximate multiplier precisions. Using the multi-core architecture, the proposed concept can achieve significant performance gains compared to designs with exact multipliers. Additionally, it can achieve a significantly better performance-accuracy trade-off compared to designs that uses single-precision approximate multipliers. This article

has also presented the performance benefits of utilizing the proposed concept in single-core designs. To optimize this utilization, a new merged approximate multiplier with two configurable precisions was proposed. The proposed concept of using a tiny preprocessing controller to determine the adequate processing precision is not limited to the DSM and SSM architectures and can be expanded to other approximate multiplier designs, also it is expandable beyond approximate multipliers. The concept can be investigated in floating-point designs to control cores with variable precisions. It is also not limited to image classification and can be explored to improve the hardware performance of other problems such as video and audio classification.

REFERENCES

[1] W. Liu, F. Lombardi, and M. Shulte, "A retrospective and prospective view of approximate computing," *Proc. IEEE*, vol. 108, no. 3, pp. 394–399, Mar. 2020.

[2] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. 18th IEEE Eur. Test Symp. (ETS)*, May 2013, pp. 1–6.

[3] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Des. Test. Comput.*, vol. 33, no. 1, pp. 8–22, Feb. 2016.

[4] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. 50th Annu. Design Autom. Conf. (DAC)*, 2013, pp. 1–9.

[5] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surveys*, vol. 48, no. 4, pp. 1–33, May 2016.

[6] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[7] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 418–425.

[8] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, "RoBA multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 393–401, Feb. 2017.

[9] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "TOSAM: An energy-efficient Truncation- and rounding-based scalable approximate multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 5, pp. 1161–1173, May 2019.

[10] S. Venkatachalam and S.-B. Ko, "Design of power and area efficient approximate multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 5, pp. 1782–1786, May 2017.

[11] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 10, pp. 3105–3117, Oct. 2016.

[12] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate Radix-4 booth multipliers for error-tolerant computing," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1435–1441, Aug. 2017.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[14] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2849–2858.

[15] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1737–1746.

[16] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[17] L. Cavigelli, D. Gschwend, C. Mayer, S. Willi, B. Muheim, and L. Benini, "Origami: A convolutional network accelerator," in *Proc. 25th Ed. Great Lakes Symp. VLSI*, 2015, pp. 199–204.

[18] S. Yin and J.-S. Seo, "A 2.6 TOPS/W 16-bit fixed-point convolutional neural network learning processor in 65-nm CMOS," *IEEE Solid-State Circuits Lett.*, vol. 3, pp. 13–16, 2020.

[19] Z. Yuan, Y. Liu, J. Yue, Y. Yang, J. Wang, X. Feng, J. Zhao, X. Li, and H. Yang, "STICKER: An energy-efficient multi-sparsity compatible accelerator for convolutional neural networks in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 55, no. 2, pp. 465–477, Feb. 2020.

[20] X. Wei, C. H. Yu, P. Zhang, Y. Chen, Y. Wang, H. Hu, Y. Liang, and J. Cong, "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs," in *Proc. 54th Annu. Design Autom. Conf.*, Jun. 2017, pp. 1–6.

[21] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, "A high performance FPGA-based accelerator for large-scale convolutional neural networks," in *Proc. 26th Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2016, pp. 1–9.

[22] F. Liang, Y. Yang, G. Zhang, X. Zhang, and B. Wu, "Design of 16-bit fixed-point CNN coprocessor based on FPGA," in *Proc. IEEE 23rd Int. Conf. Digit. Signal Process. (DSP)*, Nov. 2018, pp. 1–5.

[23] I. Hammad and K. El-Sankary, "Impact of approximate multipliers on VGG deep learning network," *IEEE Access*, vol. 6, pp. 60438–60444, 2018.

[24] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *Proc. 35th Int. Conf. Comput.-Aided Design*, Austin, TX, USA, Nov. 2016, pp. 1–7.

[25] Z. Liu, A. Yazdanbakhsh, T. Park, H. Esmaeilzadeh, and N. S. Kim, "SiMul: An algorithm-driven approximate multiplier design for machine learning," *IEEE Micro*, vol. 38, no. 4, pp. 50–59, Jul. 2018.

[26] I. Hammad, K. El-Sankary, and J. Gu, "Deep learning training with simulated approximate multipliers," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dali, China, Dec. 2019, pp. 47–51.

[27] M. S. Ansari, V. Mrazek, B. F. Cockburn, L. Sekanina, Z. Vasicek, and J. Han, "Improving the accuracy and hardware efficiency of neural networks using approximate multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 317–328, Feb. 2020.

[28] B. Moons, D. Bankman, and M. Verhelst, *Circuit Techniques for Approximate Computing*, in *Embedded Deep Learning*. Cham, Switzerland: Springer, 2019.

[29] Z. G. Tasoulas, G. Zervakis, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Weight-oriented approximation for energy-efficient neural network inference accelerators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 12, pp. 4670–4683, Dec. 2020.

[30] F. Chollet. (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[32] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[33] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[34] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do ImageNet classifiers generalize to ImageNet?" 2019, *arXiv:1902.10811*. [Online]. Available: http://arxiv.org/abs/1902.10811

[35] V. Leon, K. Asimakopoulos, S. Xydis, D. Soudris, and K. Pekmestzi, "Cooperative arithmetic-aware approximation techniques for energy-efficient multipliers," in *Proc. 56th Annu. Design Automat. Conf.*, 2019, pp. 1–6.

[36] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[37] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," 2018, *arXiv:1803.08375*. [Online]. Available: http://arxiv.org/abs/1803.08375

[38] (2020). *NanGate FreePDK15 Generic Open Cell Library*. Silicon Integration Initiative. (Si2). [Online]. Available: https://si2.org/open-cell-library/

[39] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, N. Boden, A. Borchers, and R. Boyle, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, 2017, pp. 1–12.

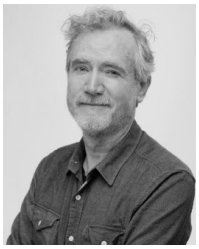
[40] X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou, and X. Ji, "High-performance FPGA-based CNN accelerator with block-floating-point arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 8, pp. 1874–1885, Aug. 2019.

[41] C. Kachris, B. Falsafi, and D. Soudris, Eds. *Hardware Accelerators in Data Centers*, vol. 1, no. 1. Springer, 2019.
[42] H. Mao, S. Yao, T. Tang, B. Li, J. Yao, and Y. Wang, "Towards real-time object detection on embedded systems," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 3, pp. 417–431, Sep. 2018.
[43] B. Sun, L. Yang, P. Dong, W. Zhang, J. Dong, and C. Young, "Ultra power-efficient cnn domain specific accelerator with 9.3 tops/watt for mobile and embedded applications," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1677–1685.

ISSAM HAMMAD (Graduate Student Member, IEEE) received the B.Sc. degree in computer engineering from the Princess Sumaya University for Technology, Amman, Jordan, in 2008, and the M.A.Sc. degree in electrical and computer engineering from Dalhousie University, NS, Canada, in 2010, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. He is currently a Senior Data Scientist with the Department of Energy, Alithya Digital Technology Corporation, Toronto, ON, Canada. He has more than ten years of industrial experience in machine learning, video compression, cryptography, cloud and web technologies, and nuclear software systems. His current research interests include deep learning, embedded/edge AI, and approximate computing.

LING LI received the B.Sc. degree in electrical engineering from Dalhousie University, Halifax, NS, Canada, in 2019, where she is currently pursuing the M.A.Sc. degree in electrical and computer engineering. Her research interests include approximate computing systems and low-power circuits and systems.

KAMAL EL-SANKARY (Member, IEEE) received the B.Eng. degree from Lebanese University, Tripoli, Lebanon, in 1997, the M.A.Sc. degree in electrical engineering from the University of Quebec, Montreal, QC, Canada, in 2002, and the Ph.D. degree in electrical engineering from École Polytechnique, University of Montreal, Montreal, in 2006. He joined the Department of Electrical and Computer Engineering, Dalhousie University, Halifax, NS, Canada, in 2006, where he is currently a Professor. His current research interests include mixed-signal, analog, digital, and RF integrated circuits design and embedded systems. He is the Chair of the Atlantic Canada IEEE Circuits and Systems and the Solid State Circuits Joint Chapter.



W. MARTIN SNELGROVE (Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Toronto, ON, Canada, in 1982. He taught at the University of Toronto from 1982 to 1992, then moved to Carleton University, ON, Canada, as a Professor and held the OCRI/NSERC Industrial Research Chair in high-speed integrated circuits from 1992 to 1998. He led a series of high-performance technical teams for a range of companies, including a Chief Scientist with Philsar Electronics (acquired by Conexant Systems), the CTO with Soma Networks, and the CEO of Dissonance, Inc., and Kapik Integration. In 2018, he co-founded Untether AI, a semiconductors company focusing on the development of ultra-efficient, high-performance AI chips to enable new frontiers in AI applications, and served as the CEO from 2018 to 2019, where he has been the CTO since 2019. He has authored 36 patents and over 100 refereed publications. He received the 1986 Circuits and Systems Society GuilleminCauer Award for an article coauthored with A. S. Sedra in 1986.