A Reconfigurable Approximate Multiplier for Quantized CNN Applications

Chuliang Guo[1] , Li Zhang[1] , Xian Zhou[1] , Weikang Qian[2] , Cheng Zhuo[1]
[1] College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China
[2] University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, Shanghai, China
Email: czhuo@zju.edu.cn

Abstract – Quantized CNNs, featured with different bit-widths at different layers, have been widely deployed in mobile and embedded applications. The implementation of a quantized CNN may have multiple multipliers at different precisions with limited resource reuse or one multiplier at higher precision than needed causing area overhead. It is then highly desired to design a multiplier by accounting for the characteristics of quantized CNNs to ensure both flexibility and energy efficiency. In this work, we present a reconfigurable approximate multiplier to support multiplications at various precisions, *i.e.*, bit-widths. Moreover, unlike prior works assuming uniform distribution with bit-wise independence, a quantized CNN may have centralized weight distribution and hence follow a Gaussian-like distribution with correlated adjacent bits. Thus, a new block-based approximate adder is also proposed as part of the multiplier to ensure energy efficient operation with awareness of bit-wise correlation. Our experimental results show that the proposed adder significantly reduces the error rate by 76-98% over a state-of-the-art approximate adder for such scenarios. Moreover, with the deployment of the proposed multiplier, which is 17% faster and 22% more power saving than a Xilinx multiplier IP at the same precision, a quantized CNN implemented in FPGA achieves 17% latency reduction and 15% power saving compared with a full precision case.

I. Introduction

Deep learning has achieved enormous success in the past few years due to its accuracy, robustness, and efficiency in various tasks. Deep learning typically employs a convolutional neural network (CNN) architecture that may conduct millions to billions multiply-and-accumulate (MAC) operations per second [1, 2]. Compared with the conventional machine learning techniques, deep learning is more computationally intensive. Thus, energy efficiency, *i.e.,* energy consumption per operation, has become a concern for deep learning implementation and deployment [3, 4]. This is especially true for mobile and embedded devices that are desired to satisfy the tight power constraints [5].

The energy breakdown of a CNN indicates that with a huge number of concurrent convolution operations from time to time, the MAC operation appears to be the bottleneck to energy efficiency [1, 2]. Although multiplier is a common arithmetic component that has been widely studied for decades [6, 7], the past focus is mainly placed upon accuracy and performance. In order to improve its energy efficiency, parameter quantization has been investigated to reduce the precision, *i.e.*, bit-width, of operands and achieve faster speed while maintaining similar accuracy. Such a strategy can be application specific and hence demand significant training and tuning overhead for different scenarios.

On the other hand, due to its deep and multi-channel structure, CNN has been found with intrinsic error tolerance. This allows designers to step further from quantization and use approximate arithmetic to improve energy efficiency. The approximate arithmetic may consume less energy to compute an approximate solution, which, however, has a minimal impact on the ultimate accuracy of a CNN [8, 9]. Thus, approximate computing is considered as a promising alternative to explore trade-off between accuracy and efficiency in addition to parameter quantization [10-12]. As the most basic arithmetic operations, various approximate adders and multipliers have been studied under different assumptions. One common assumption is that the inputs are independent and uniformly distributed, thereby reducing the occurrence probability of a long carry chain [13, 14]. Researchers also come with various proposals on approximations in partial products [15] and reduction trees [16-18]. However, *most designs have a fixed bit-width and can hardly be adapted to different scenarios without additional design efforts.*

Recently, the concept of dynamic accuracy scaling (DAS) has been proposed for the multiplier to adapt to various scenarios without redesign [19]. The DAS multiplier employs an array architecture as shown in Fig. 1(a) to support up to 4-bit multiplication. When computing a multiplication with a shorter bit-width, *e.g.*, 2-bit, as shown in Fig. 1(b), only part of the architecture is used to reduce its critical path length. However, due to its diagonal critical path, when under such a scenario, the majority of the multiplier is actually inactive, thereby causing resource waste and actually hurting energy efficiency. The inefficiency for such a reconfigurable array DAS multiplier is even worse for a quantized CNN, whose weights often have a smaller bit-width than the inputs, *e.g.*, 8-bit weights for 16- or 32-bit inputs. Such *asymmetricity* in the bit-width for operands makes a DAS multiplier unsuitable for quantized CNN applications.

Thus, for the implementation of a quantized CNN, it is highly desired to account for its unique characteristics to design a multiplier with both energy efficiency and flexibility. Apparently, this is not a trivial task. In this work, to address the aforementioned issues, we propose a reconfigurable approximate multiplier architecture to support multiplications at different bit-widths. In particular, our main contributions include:

1) **Reconfigurable multiplier:** Instead of partial use of array multipliers for shorter bit-width operation, we here propose to design a multiplier that can either partition a long bit-width multiplication to short ones and merge the results later with a merging module, or conduct two short bit-width multiplications in parallel. Fig. 1(c) illustrates an example of the proposed design. Unlike the ones in Fig. 1(a) or 1(b), the proposed design supports multiple short bit-width multiplications in parallel without resource waste (plotted in red and blue in Fig. 1(c) for two parallel multiplications).
2) **Sign extension**: An efficient sign extension scheme is key to long bit-width multiplication partitioning and merging in

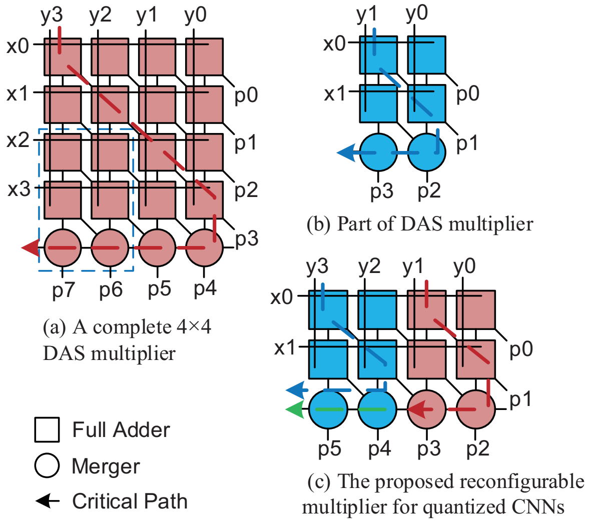
978-1-7281-4123-7/20/$31.00 ©2020 IEEE

a signed multiplication. Such a reconfigurable multiplier is then particularly beneficial to a quantized CNN that may carry different bit-widths at different layers and conduct multiplications with unequal operands.

3) **Energy efficient approximate adder:** The ultimate result of long bit-width multiplication is merged by shifting and adding the shorter products. Due to the heavier usage of such adders in our multiplier, a block-based approximate adder is proposed as the merger to achieve energy efficiency, including a sign error correction scheme for quantized CNNs in signed digits.

4) **Theoretical analysis of correlated adjacent bits for inputs with Gaussian distribution:** For a quantized CNN, the inputs to the approximate adder in the proposed multiplier are not necessarily uniformly distributed as assumed in many prior works. Instead, it is more like a Gaussian distribution, *i.e.*, with correlation from bit to bit (as detailed in section II-A). We provide an in-depth analysis of the relationship between the correlation of adjacent bits and the underlying distribution. The findings then function as the guideline for our approximate adder design.

Our experimental results show that the proposed approximate adder can significantly reduce the errors for the inputs following Gaussian distributions, with a 76-98% error reduction in comparison with a state-of-the-art approximate adder. Based on that, the proposed reconfigurable approximate multiplier can achieve 19% speed up with 22% power saving over the speed-optimized Xilinx IP. Finally, we demonstrate the efficiency of the proposed design when deployed in a quantized VGG16 for image classification task on ImageNet [20]. The proposed design can obtain 17% latency reduction and 15% power saving with merely 3% accuracy loss.



(a) A complete 4×4 DAS multiplier

(b) Part of DAS multiplier

(c) The proposed reconfigurable multiplier for quantized CNNs

Fig. 1 DAS multiplier examples for: (a) 4×4-bit multiplication; (b) 2×2-bit multiplication, where *x*'s and *y*'s are inputs and *p*'s are partial products; and (c) the proposed multiplier.

II. Preliminaries and Background

A. Multiplication in Quantized CNN

Quantization has been proved as an effective method to achieve energy efficiency [21-23]. Various strategies have been proposed to conduct different levels of quantization at different CNN layers. For example, [24] employs 16-bit feature but different weight bit precision from 1 bit to 16 bit to achieve 50.6TOPs/W energy efficiency. Similarly, [25] employs more aggressive quantization to achieve a model at KB level. Thus, it has been a common practice in both algorithm and hardware implementations to deploy variable quantization precisions, *e.g.,* 8- and 4-bit, at different layers [26]. However, since the input to a CNN typically maintains at 32- or 16-bit, a multiplier in a quantized CNN needs to repeatedly compute multiplications with asymmetric operands, *e.g.,* 16-bit *vs.* 8-bit, and with different bit-widths at different layers, *e.g.,* 32- by 8-bit for one layer but 32- by 4-bit for another. *A multiplier with a fixed bit-width is then either incapable or inefficient for such scenarios.*

Moreover, even if the original weights before quantization is uniformly distributed, the quantized weights tend to be more centralized after training due to the sparsity driven nature and the fact that many weights are small, *i.e.,* close to 0. For example, by employing the quantization methods as in [27], the weights in a VGG16 network can be quantized to 8-bit and form a Gaussian-like distribution, as shown in Fig. 2. Similar findings and more theoretical analysis have been discussed in prior works [28, 29] that *the distributions of quantized weights are roughly Gaussian rather than uniform distributions.*

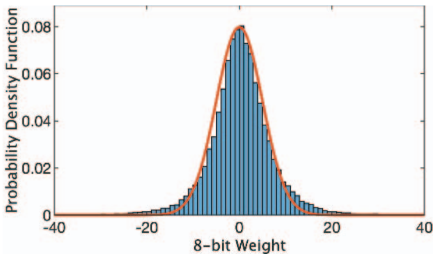


Fig. 2 Distribution of quantized weights in a VGG that follows a Gaussian distribution.

B. Error Metrics for Approximate Arithmetic

There are four common error metrics used to evaluate the accuracy of an approximate arithmetic design: error rate (ER), mean error distance (MED), mean relative error distance (MRED), and mean square error (MSE). Error distance (ED) is defined as the absolute value of difference between accurate and approximate results S and S^*:

$$ED = |S - S^*| \tag{1}$$

ER is defined as the percentage of inputs when their EDs are non-zero:

$$ER = P(ED > 0) \tag{2}$$

MED, MRED, and MSE are computed from ED as:

$$MED = E[ED] = \sum_{ED_i \in \Omega} ED_i P(ED_i) \tag{3}$$

$$MSE = E[ED^2] = \sum_{ED_i \in \Omega} ED_i^2 P(ED_i) \tag{4}$$

$$MRED = E\left[\frac{ED}{S}\right] = \sum_{ED_i \in \Omega} \frac{ED_i}{S_i} P(ED_i) \tag{5}$$

where Ω is the set of all EDs. ER is commonly used for error recovery in approximate computing. MED and MRED are criteria to measure the average closeness between the accurate and approximate results, while MSE is more associated with peak signal-to-noise ratio (PSNR) [30].

C. Multiplier

The hardware implementation of a multiplier for two digital inputs typically involves three steps: 1) Generate partial products; 2) Reduce partial product array until 2 operands are left; 3) Propagate the carry of the addition of the remaining operands for the final result. There are two challenges in such a multiplier design. One is to reduce the size of a partial product reduction tree [31]. The other is to reduce the length of the carry propagation path [32]. In many prior approximate multiplier works, they either approximate the partial products or use approximate counters and compressors in the partial product reduction tree, thereby improving the critical path delay.

III. Proposed Reconfigurable Approximate Multiplier

A. Overview of the Design

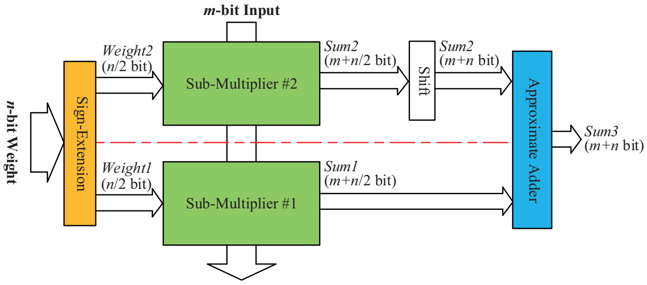


Fig. 3 Overview of the proposed multiplier with three key modules: sign-extension for partitioning, sub-multiplier and approximate adder-based merger.

Fig. 3 presents an overview of the proposed reconfigurable approximate multiplier. The multiplier contains three key parts: (1) **Sign extension module (detailed in section III-B)** for operand partition; (2) **Sub-multiplier module** using Modified Booth Wallace multiplier; (3) **Merger module (detailed in section III-C)** using an approximate adder. The multiplier can function in two modes:

- **Long bit-width multiplication mode:** This mode supports multiplication with two operands with unequal bit-widths, *e.g.*, *m* bits and *n* bits. This can happen when one is for the input to a layer of CNN and the other is for the weight, *i.e.*, $m > n$. The sign extension module partitions a signed multiplication into two shorter ones, which will be calculated in two sub-multipliers for the least significant part (LSP) and the most significant part (MSP). The two sub-multipliers conduct two accurate $m \times n/2$ signed multiplications in parallel. A logic shifter shifts the output of MSP sub-multiplier and send to the merger module for merging.
- **Short bit-width multiplication mode**: Under this mode, the multiplier conducts two multiplications in parallel. Instead of one *n*-bit weight, two shorter weights of $n/2$ bits are prefetched and sent to the two sub-multipliers. Two *m*-bit operands are then simultaneously sent to the two sub-multipliers for multiplications. The results are separately calculated with logic shifter and merger module both bypassed under this mode.

Apparently, compared with the DAS multipliers in Fig. 1, the proposed structure doubles the available multipliers when dealing with short bit-width multiplications, *e.g.*, 16- by 4-bit multiplications. Or it can be used to conduct long bit-width multiplication without introducing a new multiplier, *e.g.*, 16- by 8-bit. Though the presented structure supports *m*- by *n*-bit and *m*- by $n/2$-bit multiplications, the proposed idea is general and can be extended to more finer partitioning, *e.g.*, $n/4$ or support a hybrid combination of different bit-widths.

In the implementation, the sub-multiplier employs a modified Booth Wallace multiplier to provide better performance. They recode sign-extended $n/2$-bit weight to generate fewer partial products, and multiply the $n/2$-bit weight by the m-bit input to obtain an $(m + n/2)$-bit output. Since $n/2$-bit can be small, *i.e.*, 4-bit, the computation is fast and hence employs full precision adders for the carry-propagation addition in the sub-multipliers. Such a sub-multiplier design also helps avoid complex wiring commonly met in partial products reduction trees. For example, for 16- by 4-bit signed sub-multiplier, with radix-4 modified Booth encoding, 3 rows of partial products are generated, and only one carry save adder is needed, thereby preventing complex wiring to further improve performance.

B. Sign-Extension Module

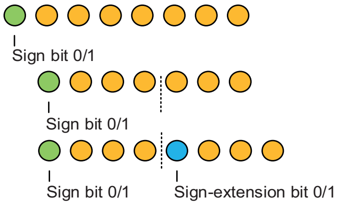


Fig. 4 Sign-extension for an n-bit operand when $n = 8$.

The sign-extension module partitions a long bit-width operand to shorter ones to conduct two $m \times n/2$-bit signed multiplications instead of a direct $m \times n$-bit multiplication. For a signed number, its most significant bit (MSB) is a sign bit (either 0 or 1). Since the two sub-multipliers both conduct signed multiplications, there is an accuracy and efficiency trade-off in the proposed sign-extension scheme. Our proposed design works under the assumption that the *n*-bit operand ranges from -2^{n-2} to $2^{n-2} - 1$.

When the n-bit operand is non-negative, *i.e.*, MSB is 0, since the non-negative operand ranges from 0 to $2^{n-2} - 1$, then the first two bits are all 0. We then have the last $n/2 - 1$ bits from the original n bits as the LSP, and add a 0 at the beginning to avoid an input overflow for the LSP sub-multiplier. On the other hand, the MSP contains $n/2$ bits with the bits from the position $n/2$ to $n - 2$ for the MSP sub-multiplier. When $n = 8$, the achieved LSP and MSP are:

00XX_XXXX = 0XXX_XXX → 0XXX_**0**XXX

Similarly, when the number is negative, if the number is smaller than $-(2^{n-3} + 1)$, we can have the last $n/2 - 1$ bits plus "0" as LSP and bits in the range of $[n/2, n - 2]$ as MSP. An example for $n = 8$ is as below:

11XX_XXXX = 1XXX_XXX
= 1XXX_000 + 0000_XXX → 1XXX_**0**XXX

Otherwise, the first $n/2 + 1$ bits are all sign bits. We then have the last $n/2 - 1$ bits plus "1" at the beginning as LSP and all zero as MSP. A case of $n = 8$ is as below:

1111_1XXX = 1111_XXX = 1XXX → 0000_**1**XXX

The concept of the proposed sign extension is briefly demonstrated in Fig. 4.

C. *Approximate Adder-Based Merger*

Block-based approximate adders have been widely studied [33, 34]. One state-of-the-art approximate adder is Generic Accuracy Configurable Adders (GeAr) [35], it has q sub-adders, p bits for carry prediction and r bits for sum. The error occurs when any sub-adder cannot obtain the *generated* carry-in signal from its current precedent. Then the error probability for a sub-adder (excluding the first one) is:

$$P_{error} = \sum_{i=0}^{r-1} P(Gi) \prod_{j=i+1}^{r+p-1} P(P_j) = \frac{1}{2^{p+1}} - \frac{1}{2^{p+r+1}} \qquad (6)$$

Since the first sub-adder is always correct, the ER of GeAr can be approximately computed as:

$$P_{Error} \approx 1 - (1 - P_{error})^{q-1}$$
$$= 1 - (1 - \tfrac{1}{2^{p+1}} + \tfrac{1}{2^{p+r+1}})^{q-1} \qquad (7)$$

A key assumption for the aforementioned analysis is independence of bits from the uniformly distributed inputs, thereby resulting in little probability of a long carry chain.

However, for a quantized CNN, we have observed that the inputs for multipliers roughly follow a Gaussian distribution instead of a uniform distribution. **This simply indicates that the input bits are correlated and may result in a long carry chain propagating errors**. If we still follow a similar design strategy as GeAr, it may cause error increase from such an approximate addition. In order to resolve that, we need to answer the following question: *How can we redesign the approximate adder for such a scenario?*

Without loss of generality, we assume to have an n-bit signed number $a[n-1:0]$ that follows a Gaussian distribution $N(\mu, \sigma)$. We would like to focus on the correlations of the adjacent bits to understand the chance of propagating errors through a long carry chain and its dependence on the underlying distribution.

We denote the correlation coefficient between the two adjacent bits x and y as ρ_{xy}, where x and y can either be 0 or 1. With X and Y representing the random variables for x and y, the correlation coefficient is:

$$\rho_{xy} = \frac{E(XY) - E(X) \times E(Y)}{\sqrt{D(X)}\sqrt{D(Y)}} \qquad (8)$$

where $E(x)$ is the probability of x being 1 in this case and $D(x) = E(x^2) - E(x)^2$. Due to the sparsity driven nature of CNN training, we can assume $\mu = 0$ for the weight distribution, as observed in Fig. 2. Then the probability for a bit being 1 is 0.5, *i.e.*, $E(X) = E(Y) = 0.5$.

For a binary sequence of 0/1 where $E(X^2) = E(X)$,

$$D(X) = E(X) - E(X)^2 = 0.25 \qquad (9)$$

Thus, we can rewrite

$$\rho_{xy} = \frac{E(XY) - 0.25}{0.25} \qquad (10)$$

The value of $E(XY)$ depends on the underlying Gaussian distribution, or σ in this case. With analysis of the following three scenarios, we can reach the findings as below:

- When $\sigma \approx 0$, it indicates that most numbers are very close to 0. For signed numbers, this simply indicates that 0 is the only non-negative number and -1 is the only negative number. Thus, there are equal probabilities for $XY = 0$ or 1, *i.e.*, $E(XY) = 0.5$.
- When $\sigma \to \infty$, this is basically a uniform distribution. In uniform distribution, the adjacent bits can be considered independent, *i.e.*, $E(XY) = E(X) \times E(Y) = 0.25$.
- Now we consider the case that the standard deviation σ can be approximately expressed by powers of two. For an n-bit signed digit $a[n-1:0]$ following a Gaussian distribution, most of its possible values reside in the range of $[-2\sigma, 2\sigma]$ and hence can be effectively expressed using the lower $log_2\sigma + 1$ bits. Meanwhile the higher bits are all 1 or 0 for signed numbers showing high correlation. Thus, the adjacent bits within the index range $[0, log_2\sigma]$ can be considered weakly correlated or uncorrelated as in the uniform distribution, while the higher bits within the index range of $[log_2\sigma + 1, n-1]$ are strongly correlated.

The relationship between the correlation coefficient ρ_{xy} (for two adjacent bits) and σ is plotted in Fig. 5 for the bit width $n = 8$, where we explore ρ_{xy} for the adjacent bits at different locations and different σ. As can be seen from the figure, the bits within the index range of $[0, log_2\sigma]$ for inputs with Gaussian distribution are weakly correlated, similar to uniform distributed ones, while the bits within the range of $[log_2\sigma + 1, n-1]$ are more strongly correlated. This simply indicates that, for Gaussian distributed inputs, there may exist a longer carry chain for the bits at higher positions (which are highly correlated and can result in the sign error for carry propagation). In other words, for prior block-based approximate adders that assume bit-wise independence and hence a shorter carry chain, such a scenario may induce an increased ER.

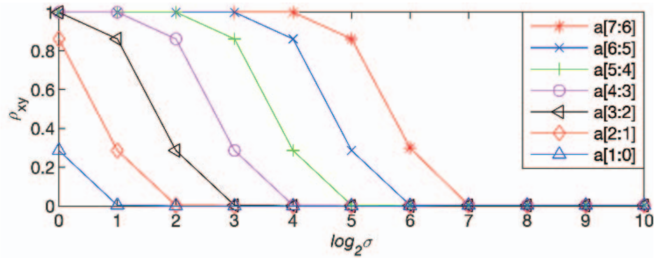


Fig. 5 Correlation coefficient ρ_{xy} for the adjacent bits at different locations and different σ.

In a block-based approximate adder, ER depends on the carry prediction bit-width p rather than the total bit-width n [14]. Then, for a block-based approximate adder with fixed bit-width, the required area for the adder linearly correlates to p, *i.e.*, a larger p will significantly increase the area overhead. Based on the findings above, for inputs following a Gaussian distribution, the bits at lower positions are close to a uniform distribution while the bits at higher positions are more correlated. Then, unlike GeAr that employ blocks with equal size, we propose to have blocks with unequal sizes for the proposed approximate adder and hence keep a small p to reduce ER. Fig. 6 presents an example for the proposed approximate adder when $n = 24$. We have three sub-adders with size of 8, 8, and 16. Sub-adder #1 overlaps with the other two sub-adders, resulting in a small $p = 4$. After each sub-adder operation, r_i bits are added to the result, reducing carry chain length but inducing errors. However, as is discussed for bits at lower positions that are almost uncorrelated, the probability that a carry needs to be propagated to the next stage is very limited. On the other hand, with sub-adder of larger size for bits at higher positions, which are correlated, the increased size actually decreases the computational error. Aware of the correlation of the bits for Gaussian distributed inputs, we design such an unequally sized block structure to trade-off between accuracy and circuit delay.

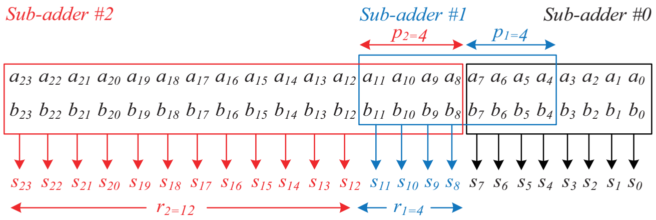

Sub-adder #2
Sub-adder #1
Sub-adder #0
$p_2=4$
$p_1=4$
$a_{23}\ a_{22}\ a_{21}\ a_{20}\ a_{19}\ a_{18}\ a_{17}\ a_{16}\ a_{15}\ a_{14}\ a_{13}\ a_{12}\ a_{11}\ a_{10}\ a_9\ a_8\ a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0$
$b_{23}\ b_{22}\ b_{21}\ b_{20}\ b_{19}\ b_{18}\ b_{17}\ b_{16}\ b_{15}\ b_{14}\ b_{13}\ b_{12}\ b_{11}\ b_{10}\ b_9\ b_8\ b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$
$s_{23}\ s_{22}\ s_{21}\ s_{20}\ s_{19}\ s_{18}\ s_{17}\ s_{16}\ s_{15}\ s_{14}\ s_{13}\ s_{12}\ s_{11}\ s_{10}\ s_9\ s_8\ s_7\ s_6\ s_5\ s_4\ s_3\ s_2\ s_1\ s_0$
$r_2=12$
$r_1=4$

Fig. 6 Proposed approximate adder with a long MSP sub-adder.

Moreover, unlike prior works that treat the sign bits equally important as the other bits, we consider the impact of sign bit error more significant for a quantized CNN and hence provide a simple yet effective sign error correct (SEC) scheme to correct the sign bit error [36]. Take the approximate adder in Fig. 6 for example. The carry-out signal of a sub-adder is erroneous only when the carry-in is 1 and $a_i \oplus b_i = 1$ for any bit pair a_i and b_i of the underlying sub-adder. Apparently, the sub-adder #0 always generates correct result. In order the correct the sign bit error for the next few sub-adders, we can define two flag signals as shown in Fig. 7, which conducts "AND" operation of the all the partial sum s_i When only $flag_i$ is 1, it indicates that the erroneous signal can be propagated to the sign bit. When $flag_2 = 1$ and $flag_1 = 0$, the accurate carry-in signal for sub-adder #2 is the carry-out of sub-adder #1, $C_{out\,\#1}$. If $C_{out\,\#1} = 1$, due to the approximation principle for the approximate adder, it will not propagate to the next sub-adder and therefore introduce and error to the sub-adder #2. Thus, we need to simply inverse all bits of $s[23:12]$ for correction. Similarly, when both $flag_1$ and $flag_2$ are 1, if the carry-out of the sub-adder #0 $C_{out\,\#0} = 1$, we need to inverse all the bits of $s[23:8]$.

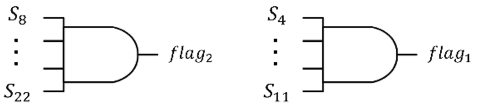


Fig. 7 Circuits to obtain flag signals for sign error correction.

IV. Evaluation

To evaluate the presented architectures in the previous section, we implement the proposed multiplier and adder in Verilog and then conduct the verification on a Xilinx ZCU102 FPGA board.

A. Proposed Block-based Approximate Adder

We first study the behavior of the proposed approximate adder (for 24-bit) when the inputs follow a Gaussian distribution, and then compare the results with the state-of-the-art GeAr design. We investigate the scenarios of the proposed block-based approximate adder with and without the sign error correct (SEC) scheme for various σ and study the metrics for approximate adder, including ER, MED, MRED and MSE.

Fig. 8 shows that, when σ is small, the proposed block-based approximate adder without SEC has a similar ER as GeAr (when $n = 24$, $r = 4$, $p = 4$). However, as σ grows, the performance is significantly improved to achieve up to 76% error reduction. This is because GeAr adder assumes the independence of bits and hence a shorter carry chain. As σ increases, the correlation also grows and results in larger error for GeAr. On the other hand, when with SEC, the proposed adder shows very consistent improvement over GeAr with up to 98% error reduction. As for smaller σ, the sign bit error is dominant for the correlated input bits. Similar observations can be found when calculating the other metrics of MED, MRED and MSE.

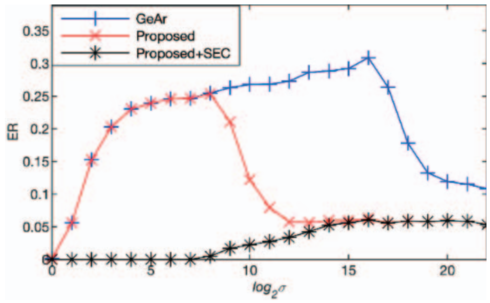


Fig. 8 ER comparison of the proposed approximate adders (with and without SEC) vs. GeAr.

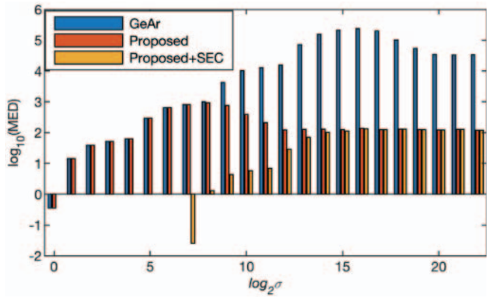


Fig. 9 MED comparison of the proposed approximate adders (with and without SEC) vs. GeAr.

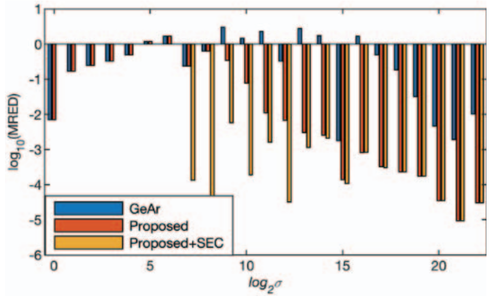


Fig. 10 MRED comparison of the proposed approximate adders (with and without SEC) vs. GeAr.

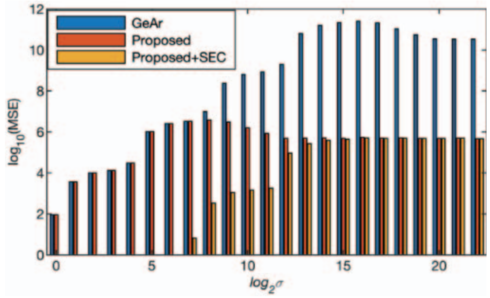


Fig. 11 MSE comparison of the proposed approximate adders (with and without SEC) vs. GeAr.

B. Proposed Reconfigurable Approximate Multiplier

We verify functionality of the proposed reconfigurable approximate multiplier with FPGA implementation and then investigate its performance in comparison with the multiplier IP that Xilinx provides. The case of $m = 16$ and $n = 8$ is studied to support 16- by 8-bit multiplication and 16- by 4-bit multiplication. Under the same timing constraints of 200MHz frequency, Table I compares the logic delays, LUT and power consumption. It is found that the proposed design can achieve 19% delay reduction over a speed-optimized Xilinx IP multiplier, while achieving a lower LUT overhead, with 22% power saving.

C. Deployment in a Quantized CNN

Finally, we deploy the proposed design in a quantized CNN. The CNN employs an Eyeriss-like [37] structure that has 168 processing elements and can be configured to different neural networks. Each processing element contains one proposed multiplier to support one 16- by 8-bit multiplication or two 16- by 4-bit multiplications. A quantized VGG16 network is implemented with different precisions at differ layers, *i.e.,* 8-bit and 4-bit, to conduct the image classification task using ImageNet. Compared with the same network employing full-precision of 16-bit, the quantized network is able to achieve 17% latency reduction and 15% power saving at the cost of 3% accuracy loss for top-1 classification task.

Table I Comparison of logic delay, LUT overhead and power consumption of the proposed multiplier with a speed-optimized Xilinx multiplier IP.

	Logic Delay (ns)	#LUTs	Power (mW)
Proposed	0.715	129	0.21
Xilinx IP	0.886	131	0.27

V. CONCLUSION

This paper proposes a reconfigurable approximate multiplier for quantized CNN applications. With awareness of different precisions at different layers in a quantized CNN, a reconfigurable multiplier is proposed to enable resource reuse for multiplications at different precisions. Then, a block-based approximate adder with sign error correction is designed to facilitate the scenario of Gaussian distributed inputs with correlations between the bits. The proposed design is 19% faster and 22% more power saving than a Xilinx IP at the same precision. When deployed in VGG16 network on FPGA for image classification, the proposal can achieve 17% latency reduction and 15% power saving with merely 3% accuracy loss.

ACKNOWLEGEMENT

This work was partially supported by NSFC with Grant No. 61974133, 61601406, and Guangdong Province with Grant No. 2018B030338001.

REFERENCE

[1] H. Lu, et al. "Tetris: Re-architecting Convolutional Neural Network Computation for Machine Learning Accelerators." Proc. ICCAD, 2018.

[2] Z. Liu, et al. "A Multi-Level Optimization Framework for FPGA-Based Cellular Neural Network Implementation." ACM JETCS, 14(4):1-47, 2018.

[3] C. Zhuo, et al. "From Layout to System: Early Stage Power Delivery and Architecture Co-Exploration." IEEE TCAD, 38(7):1291-1304, 2019.

[4] C. Zhuo, et al. "Noise-Aware DVFS for Efficient Transitions on Battery-Powered IoT Devices." IEEE TCAD, DoI:10.1109/TCAD.2019.2917844, 2019.

[5] S. Luo, et al. "Noise-Aware DVFS Transition Sequence Optimization for Battery-Powered IoT Devices." Proc. DAC, 2018.

[6] H. D. Tiwari, et al. "Multiplier Design Based on Ancient Indian Vedic Mathematics. " Proc. SoCDC, 2008.

[7] M. Bansal, et al. "High performance pipelined signed 64x64-bit multiplier using radix-32 modified Booth algorithm and Wallace structure." Proc. CICN, 2011.

[8] S. Hashemi, et al. "Understanding the Impact of Precision Quantization on the Accuracy and Energy of Neural Networks." Proc. DATE, 2017.

[9] J. H. Ko, et al. "Adaptive Weight Compression for Memory-Efficient Neural Networks." Proc. DATE, 2017.

[10] J. Han and M. Orshansky. "Approximate Computing: An Emerging Paradigm for Energy-Efficient Design." Proc. ETS, 2013.

[11] V. Gupta, et al. "IMPACT: Imprecise Adders for Low-Power Approximate Computing." Proc. ISLPED, 2011.

[12] V. K. Chippa, et al. "Analysis and Characterization of Inherent Application Resilience for Approximate Computing." Proc. DAC, 2013.

[13] A. K. Verma, et al. "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design." Proc. DATE, 2008.

[14] S. Mazahir, et al. "Probabilistic Error Modeling for Approximate Adders. " IEEE TC, 66(3):515-530, 2017.

[15] P. Kulkarni, et al. "Trading Accuracy for Power with an Underdesigned Multiplier Architecture. " Proc. VLSID, 2011.

[16] H. R. Mahdiani, et al. "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications." IEEE TCAS-I, 57(4):850-862, 2010.

[17] K. Y. Kyaw, et al. "Low-power high-speed multiplier for error-tolerant application." Proc. EDSSC, 2010.

[18] K. Bhardwaj, et al. "Power- and area-efficient Approximate Wallace Tree Multiplier for error-resilient systems." Proc. ISQED, 2014.

[19] B. Moons and M. Verhelst. "DVAS: Dynamic Voltage Accuracy Scaling for increased energy-efficiency in approximate computing." Proc. ISLPED, 2015.

[20] J. Deng, et al. "ImageNet: A Large-Scale Hierarchical Image Database." Proc. CVPR, 2009.

[21] Z. Liu, et al. "An Efficient Segmentation Method Using Quantized and Non-linear CeNN for Breast Tumor Classification." IET EL, 54(12):737-738, 2018.

[22] D. Kim, et al. "Convolutional Neural Network Quantization using Generalized Gamma Distribution." arXiv preprint arXiv:1810.13329, 2018.

[23] J. Cheng, et al. "Quantized CNN: A Unified Approach to Accelerate and Compress Convolutional Networks." IEEE TNNLS, DoI: 10.1109/TNNLS.2017.2774288, 2017.

[24] J. Lee, et al. "UNPU: An Energy-Efficient Deep Neural Network Accelerator with Fully Variable Weight Bit Precision." Proc. JSSC, 2019.

[25] I. Chakraborty, et al. "Efficient Hybrid Network Architectures for Extremely Quantized Neural Networks Enabling Intelligence at the Edge." Proc. ACM, 2019.

[26] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper." arXiv preprint arXiv:1806.08324, 2018.

[27] R. Banner, et al. "Scalable Methods for 8-bit Training of Neural Networks." Proc. NIPS, 2018.

[28] D. D. Lin, et al. "Fixed Point Quantization of Deep Convolutional Networks." Proc. ICML, 2016.

[29] Z. Cai, et al. "Deep Learning with Low Precision by Half-Wave Gaussian Quantization." Proc. CVPR, 2014.

[30] L. Li, and H. Zhou. "On error modeling and analysis of approximate adders." Proc. ICCAD, 2014.

[31] E. Antelo, et al. "Improved 64-bit Radix-16 Booth Multiplier Based on Partial Product Array Height Reduction." IEEE TCAS-I, 64(2): 409-418, 2017.

[32] W. Liu, et al. "Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing." IEEE TC, 66(8): 1435-1441, 2017.

[33] N. Zhu, et al. "An enhanced low-power high-speed Adder for Error-Tolerant application." Proc. Proceedings of ISIC, 2009.

[34] A. B. Kahng, et al. "Accuracy-configurable adder for approximate arithmetic designs." Proc. DAC, 2012.

[35] M. Shafique, et al. "A low latency generic accuracy configurable adder." Proc. DAC, 2015.

[36] R. Zhou, et al. "A general sign bit error correction scheme for approximate adders." Proc. GLSVLSI, 2016.

[37] Y. H. Chen, et al. "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks." Proc. ISCA, 2016.