Jumping Shift

A Logarithmic Quantization Method for Low-Power CNN Acceleration

Jiang, Longxing; Aledo , David; van Leuken, Rene

DOI
10.23919/DATE56975.2023.10137169

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)

Jumping Shift: A Logarithmic Quantization Method for Low-Power CNN Acceleration

Longxing Jiang, David Aledo, Rene van Leuken
Circuits and Systems Group
Delft University of Technology

***Abstract*—Logarithmic quantization for Convolutional Neural Networks (CNN): a) fits well typical weights and activation distributions, and b) allows the replacement of the multiplication operation by a shift operation that can be implemented with fewer hardware resources. We propose a new quantization method named Jumping Log Quantization (JLQ). The key idea of JLQ is to extend the quantization range, by adding a coefficient parameter "s" in the power of two exponents (2^{sx+i}). This quantization strategy skips some values from the standard logarithmic quantization. In addition, we also develop a small hardware-friendly optimization called weight de-zero. Zero-valued weights that cannot be performed by a single shift operation are all replaced with logarithmic weights to reduce hardware resources with almost no accuracy loss. To implement the Multiply-And-Accumulate (MAC) operation (needed to compute convolutions) when the weights are JLQ-ed and de-zeroed, a new Processing Element (PE) have been developed. This new PE uses a modified barrel shifter that can efficiently avoid the skipped values. Resource utilization, area, and power consumption of the new PE standing alone are reported. We have found that JLQ performs better than other state-of-the-art logarithmic quantization methods when the bit width of the operands becomes very small.**

***Index Terms*—Convolutional Neural Network, Low-power hardware acceleration, Logarithmic Quantization, FPGA**

I. Introduction

In recent years, there has been a lot of interest in deep learning. Convolutional Neural Network (CNN), one of the most mature deep learning models, has attracted attention in various fields such as medical research [1] [2], language processing [3], and visual imagery [4] [5] [6]. Despite its popularity, due to its huge data volume, intensive computation, and frequent memory access, deploying a CNN on low-power hardware systems is still challenging. To make deep neural networks generally easier to deploy on hardware devices like FPGA and ASIC, various quantization algorithms [7] [8] [9] have been devised to reduce memory requirements. Among all quantization methods, logarithmic quantization is very suitable for low-power inference because as logarithmic weights are represented by powers of two. The memory only needs to store the integer power index instead of the floating point weight. Besides, logarithmic quantization fits better the common CNN Gaussian-like distributions of weights and activations, than the more uniform integer quantization. Furthermore, logarithmic quantization allows the replacement of the multiplication operation (massively involved in CNN computation) by a shift operation, which can be implemented with fewer hardware resources. Because of all the previous reasons, logarithmic quantization effectively reduces the CNN memory footprint and reduces the area and power consumption of the Processing Elements (PEs) involved in CNN computation.

In this paper, we introduce a logarithmic quantization technique named Jumping Logarithmic Quantization (JLQ) that can achieve higher accuracy than traditional logarithmic quantization at low-bit quantization by extending the quantization range. JLQ replaces traditional logarithmic weights ($\pm 2^{x+i}$) with jumping logarithmic weights ($\pm 2^{sx+i}$, "s" is the jumping step parameter) so that the quantization range is extended when the bit width is the same. We performed tests on CIFAR10 [10], CIFAR100 [10], and TinyImageNet [11] to evaluate the accuracy based on our quantization technique. Compared with the state-of-the-art logarithmic quantization algorithm, our method has obvious advantages in the low-bit (2 or 3-bit) quantization cases. Besides, we develop a hardware-friendly weight de-zero optimization. Hardware resources are reduced by replacing zero-valued weights with logarithmic weights that can be performed by shift operations.

In addition, we design a processing element (PE) based on JLQ and weight de-zero optimization, aiming to perform MAC operations of CNN models with fewer resources, lower area, and power consumption. The processing element (PE) realizes the quantization range extension of JLQ without additional hardware resources by optimizing the traditional barrel shifter. For ASIC simulation, we synthesized our PE design, a multiplier-based PE design, and a traditional shifter-based PE design using TSMC 28nm technology (in the 3-bit and 2-bit quantization case). Compared with other competitors, our PE design has less area and power consumption. In addition, we also conduct experiments regarding the resource consumption of different PEs on Xilinx ARTIX7 XC7A100T FPGA. The results show that compared with traditional shifter-based PE, our PE achieves 35% and 28.6% LUT reduction in 2-bit and 3-bit quantization cases respectively.

The rest of this paper is organized as follows. The related work on logarithmic quantification is described in Section II. Then, we introduce our proposed quantization method, JLQ, in Section III. Section IV presents the accuracy results of JLQ on various datasets and backbone networks. In Section V, we present our PE design and the corresponding hardware implementation results. Finally, Section VI and Section VII are the conclusion and future work, respectively.

This work was supported by the NewControl Project, funded by Electronic Components and Systems for European Leadership Joint Undertaking (ECSEL JU) in collaboration with the European Union's H2020 Framework Programme and National Authorities, under grant agreement no. 826653-2.

978-3-9819263-7-8/DATE23/© 2023 EDAA

II. Related Work

LogNN [12] was the first to propose logarithmic quantization for CNNs. They propose two methods: in the first method, the weights remain fix-point and the activations are log-quantized. While in the second method, both weights and activations are log-quantized.

ShiftCNN [13] replaces each multiplication with a set of 2 or 3 shifts, therefore their PE consumes more hardware resources and has higher power consumption than the PE with only a single shifter. Similar to ShiftCNN, in [14] and [15], to improve the accuracy, they also replace each multiplication with the sum of two shift operations.

In DeepShift [8], the author proposes two methods: DeepShift-Q and DeepShift-PS. DeepShift-Q can be regarded as the standard logarithmic quantization. In DeepShift-PS, since the power-of-two function is differentiable, they implement a deeper derivation of the backward propagation pass, which is the main innovation in this paper. Besides, both DeepShift-Q and DeepShift-PS support training from scratch.

The main feature of INQ [9] is implementing incremental retraining from a pre-trained model to increase the quantization accuracy. The INQ algorithm consists of three steps: the first step is to sort the weights by absolute value and divide them into two groups according to a certain proportion; the second step is to quantize the group of weights with larger absolute value; the third step is to retrain the group of weights with relatively smaller absolute value. After iterating these three steps, the overall quantization can be completed. In INQ, the proportion of each quantization is a hyperparameter.

In [16], the author also uses incremental retraining and makes it more adaptive to sparse models through their centralized quantization method.

Furthermore, logarithmic quantization is also applied in approximate computing. In [17], the author proposes an approximate shifter-based PE. And in [18], an approximate logarithmic data representation is proposed for CNN training. Compared with these methods, in our JLQ method, once the quantization has been performed, the computations are exact. Besides, our JLQ method gets higher accuracy in extreme low-bit quantization cases: 2-bit and 3-bit quantization with 8-bit activations.

III. Jumping Logarithmic Quantization

Many studies indicated that weights in most mainstream non-sparse CNN models generally follow a Gaussian-like distribution [19]. That means, the majority of the CNN weights have small values, and only a few outliers have relatively large values. Based on this information, some quantization techniques [7] intentionally use the quantization strategy of the Lloyd–Max quantizer: fewer sampling points are used to quantize weights with large absolute values and more sampling points are used to quantize weights with small absolute values. However, some previous studies show that weights with larger absolute values are not inessential. In fact, they are more critical in feature extraction than those with smaller values [20]. Therefore, based on these three facts, we developed the new logarithmic quantization technique JLQ.

To achieve our goal, this quantization technique extends the quantization range by introducing one jumping step parameter "s" so that both big-value weights and small-value weights can be taken into consideration in the extreme low-bit quantization case. In JLQ, the logarithmic quantization values can be represented as follows, where "i" can be regarded as a scaling parameter (or pre-shift):

- **Quantization Weights** = $(\pm 2^{sx+i})$

A. Quantization Error estimation model

To theoretically determine the optimal hyperparameter "s" (jumping step parameter) of the JLQ, a quantization error estimation model is proposed. In this model, the main factor taken into account is the signal-to-noise ratio (SNR). In our model, weights W_i are assumed to be in the range of [-1, 1], and they follow a Gaussian distribution N(μ, δ^2), where μ is very close to 0. In addition, since the slope of the long tail part of the Gaussian distribution is extremely small, for simplicity, weights located in the [-1, -3δ) and (3δ, 1] intervals are regarded to have a uniform distribution instead of a Gaussian distribution in our model. For most typical non-sparse CNN models, the value of δ is between 0.01 and 0.09 [19]. To facilitate the comparison of quantization errors engendered by different jumping steps and scaling parameters, in this error estimation model, δ is assumed to be the average of 0.01 and 0.09, which is 0.05.

Consider a general N-level quantizer that is specified by N + 1 decision borders b_i, where i = 0 to N; and N reconstruction points x_i, where i = 1 to N. The quantization operator $Q(x)$ is given by $Q(x) = x_i$ if $b_{i-1} < x \leq b_i$ (In our model, we use the same strategy of Lloyd–Max quantizer to determine b_{i-1} and b_i). Given a random model with probability density function $f_x(x)$, the distortion D can be represented as (1):

$$D = \sum_{i=1}^{N} \int_{b_{i-1}}^{b_i} (x - x_i)^2 f_x(x) dx \tag{1}$$

The signal-to-noise ratio (SNR) can be calculated as (2), where D is calculated using (1):

$$SNR = 10 \cdot \log_{10} \cdot \frac{\delta_x^2}{D} \tag{2}$$

When weights obey a uniform distribution, (1) is easy to calculate, since the probability density function $f_x(x)$ and the variance δ_x^2 can be regarded as constant. However, when weights obey a Gaussian distribution, the probability density function $f_x(x)$ turns to be complex. In this case, (1) is no longer possible to be calculated with the Newton-Leibniz formula. To calculate it numerically, the alternative method used in this paper is the Simpson's Rule formula.

B. Weight De-zero Optimization

Since zero weights cannot be applied to the shift operation, additional hardware resources (multiplexers) will be needed to perform them. In order to reduce hardware resources, we propose a weight de-zero optimization, which can be achieved by replacing the ternary sign operator $\{-1, 0, +1\}$ with the

TABLE I
ESTIMATED SIGNAL TO NOISE RATIO

s	i	W	SNR(dB)
1	0	2	0^{T}
1	-2	2	-5.13
1	-3	2	2.93
1	-4	2	7.83
1	-5	2	4.69
2	-2	2	2.49
2	-3	2	7.90
2	**-4**	**2**	**8.76**
2	-5	2	4.72
3	-1	2	2.46
3	-2	2	4.23
3	-3	2	5.49
3	-4	2	8.14
1	0	3	0.18^{T}
2	0	3	8.88
2	-1	3	9.01
2	**-2**	**3**	**9.12**
2	-3	3	8.37

$^{\mathrm{T}}$Ternary sign.

binary sign operator $\{-1, +1\}$ An example that can show the difference between implementing the ternary sign operator and the binary sign operator is as follows:

- **Quantization weights with ternary operator (3-bit)**: ($\pm 2^0$,$\pm 2^{-1}$,$\pm 2^{-2}$,0)
- **Quantization weights with binary operator (3-bit)**: ($\pm 2^0$,$\pm 2^{-1}$,$\pm 2^{-2}$,$\pm 2^{-3}$)

C. Parameter Selection

In order to obtain the theoretical optimal parameters, it is necessary to calculate and compare the estimated SNR of different parameter sets. Summarizing these results yields Table I.

It can be seen that for low-bit quantization cases, extending the quantization range can effectively increase SNR. And among all the jumping parameter sets, s=2 is optimal in theory.

IV. BENCHMARK RESULTS

We have tested the training results on 3 datasets: CIFAR10 [10], CIFAR100 [10], and Tinyimagenet [11]. For the CIFAR10 dataset, the original DeepShift-PS and DeepShift-Q are set as the baseline, and two DeepShift modes integrated with JLQ are set as the comparison group to get a preliminary conclusion. And CIFAR100 and Tiny ImageNet datasets are used to verify the scalability of JLQ. For a fair comparison, we set all the training parameters consistent with those of DeepShift [8]. Different from the display strategy of DeepShift, which only shows the best accuracy results, the accuracy we show is the median after three experiments.

A. Benchmark Statement

In order to reduce the burden of memory, researchers often want a quantization method that can minimize bit-width for both, weights and activation data. In order to demonstrate the advantages of our proposed method in low-bit quantization, most of our experiments are based on 8 activation bits. And since the training of DeepShift-PS at 8 activation bits fluctuates greatly and its loss curve is difficult to converge in this case according to our experiments, we select the highest accuracy during the entire training process as the final accuracy value of a single experiment when doing experiments in relation to DeepShift-PS.

TABLE II
CIFAR10 BENCHMARK FROM PRE-TRAINED, ACC@1

W	A	Sign	Base-PS	JLQ-PS	Base-Q	JLQ-Q
2	8	T	81.75%	90.04%	90.37%	**93.03%**
2	8	B	81.11%	89.89%	89.46%	**93.15%**
3	8	T	90.53%	90.07%	92.67%	**93.27%**
3	8	B	90.37%	89.81%	92.01%	**93.31%**
4	8	T	93.75%	90.18%	**93.95%**	93.29%
4	8	B	93.61%	90.06%	**93.88%**	93.34%
4	32	T	**94.06%**	90.24%	94.05%	93.46%
4	32	B	93.91%	90.17%	**93.95%**	93.37%

In our test strategy, we will first test the jumping logarithmic quantization with the parameter set "s=2, i=-3" compared with the DeepShift baseline. At the same time, in order to test the effect of weight de-zero optimization on the accuracy, experiments using the ternary sign operator and binary sign operator will also be performed based on the CIFAR10 dataset. After that, we will test the accuracy of jumping logarithmic quantization with other parameter sets to determine the optimal scaling factor (i) while verifying the prediction of our error estimation model.

In our tables, "W" refers to the number of bits to represent weights, "A" refers to the number of bits to represent activation bits, "T" refers to the ternary sign operator, "B" refers to the binary sign operator, and "Acc@N" accuracy means that the correct class gets to be in the Top-N probabilities for it to count as "correct". We also highlight the useful data in our table to make the data comparison and conclusion extraction more convenient.

B. CIFAR10 Dataset

The accuracy results shown in Table II are based on the results reproduced by us instead of the results from the original DeepShift paper.

We can see that in 2-bit quantization, DeepShift-PS integrated with jumping logarithmic quantization has better performance than the corresponding baseline. Meanwhile, DeepShift-Q integrating with jumping logarithmic quantization achieves higher accuracy than any baseline in 2-bit and 3-bit quantization. Another conclusion is that quantization methods using the ternary sign operator only have slight accuracy advantages compared to those using the binary sign operator. That means, using the binary sign operator that is more hardware-friendly is feasible.

We also find some drawbacks of JLQ. For 4-bit or larger bit-width quantization, it brings little to no accuracy improvement than 3-bit quantization. This can be explained by the following two reasons. The first reason is that JLQ adjusts the step size and overdraws the accuracy improvement by expanding the quantization range in advance. Another reason is that for most non-sparse CNN models when the logarithmic weight

TABLE III
CIFAR10 RESNET18 BASED ON OTHER PARAMETER SETTINGS OF JLQ (METHOD=JLQ-Q, A=8, SIGN=B)

Parameters	W	From pre-trained
s = 1, i = -3	2	89.56%
s = 1, i = -4	2	89.98%
s = 2, i = -1	2	87.81%
s = 2, i = -2	2	90.60%
s = 2, i = -3	2	**93.15%**
s = 2, i = -4	2	91.71%
s = 1, i = -3	3	93.10%
s = 1, i = -4	3	91.90%
s = 2, i = -1	3	**93.33%**
s = 2, i = -2	3	93.20%
s = 2, i = -3	3	93.31%

is less than a certain small value, the mutual substitution between those adjacent small logarithmic weights will become very obvious. For example, adding or removing weight value $\pm 2^{-8}$,$\pm 2^{-9}$ or $\pm 2^{-10}$ by adjusting the quantization range will have almost no impact when there exists 2^{-7} in the original quantization case. As for the performance of other parameter settings, to fully demonstrate the advantage of extending the quantization range, we also intentionally filter parameter sets such as "s=1, i=-1", "s=1, i=-2" and so on, which only achieve a small range extension. And the results are shown in Table III (JLQ-Q results of s = 2, i= -3 are mentioned again for the convenience of conclusion extraction).

For CIFAR10 Dataset, in 2-bit quantization, the parameter set that achieves the best accuracy is "s=2, i=-3", and in 3-bit quantization, the optimal parameter set is "s=2, i=-1". It can be seen that the optimal parameter sets in CIFAR10 benchmark results are a little bit different from those in our error estimation model. This is because weights with larger absolute values are essential in feature extraction. Namely, it is infeasible to pay full attention to weights with smaller absolute values when implementing quantization, even though they may bring more benefits in SNR improvement.

C. CIFAR100 Dataset

As for the CIFAR100 dataset, we simplified our experiments based on the results of CIFAR10. We intentionally removed experiments in relation to Baseline PS and JLQ-PS, considering Baseline Q shows more benefits after integrating with JLQ. We conduct experiments with two backbone networks, ResNet18 and GoogleNet, to verify the scalability of JLQ. For the convenience of hardware implementation, we keep the weight de-zero optimization and apply it to all experiments of CIFAR100. Additionally, we add a control group named "original" that does not implement any quantization to better demonstrate the accuracy comparison. The accuracy results based on ResNet18 are shown in Table IV. The accuracy results based on GoogleNet are shown in Table V.

Since the CIFAR100 dataset has more types of images than the CIFAR10 dataset, it becomes more difficult to identify the CIFAR100 dataset. In this case, the advantages of JLQ are more obvious. In terms of accuracy, the conclusions in GoogleNet are consistent with those in ResNet. In the 2-bit and 3-bit quantization cases, whether the model is trained from scratch or from a pre-trained model, the accuracy is dramatically improved by implementing JLQ. However, in the 4-bit quantization case, the JLQ shows no advantages compared to the baseline as a trade-off.

TABLE IV
RESNET18 CIFAR100 BENCHMARK (SIGN=B)

Method	W	A	From scratch		From pre-trained	
			Acc@1	Acc@5	Acc@1	Acc@5
Original	32	32	74.75%	92.97%	-	-
Base-Q	2	8	28.05%	58.29%	64.97%	86.69%
JLQ-Q	2	8	**71.02%**	**90.97%**	**72.33%**	**91.43%**
Base-Q	3	8	61.79%	87.68%	66.84%	88.45%
JLQ-Q	3	8	**72.73%**	**91.40%**	**73.45%**	**92.17%**
Base-Q	4	8	**74.10%**	**92.46%**	**73.99%**	**92.60%**
JLQ-Q	4	8	72.75%	91.51%	73.33%	92.13%
Base-Q	4	32	**74.33%**	**92.56%**	**74.14%**	**92.71%**
JLQ-Q	4	32	73.15%	91.77%	73.57%	92.36%

TABLE V
GOOGLENET CIFAR100 BENCHMARK (SIGN=B)

Method	W	A	From scratch		From pre-trained	
			Acc@1	Acc@5	Acc@1	Acc@5
Original	32	32	78.17%	94.58%	-	-
Base-Q	2	8	44.97%	76.30%	62.90%	87.07%
JLQ-Q	2	8	**76.51%**	**93.82%**	**76.36%**	**93.65%**
Base-Q	3	8	67.89%	90.44%	67.73%	89.80%
JLQ-Q	3	8	**77.11%**	**94.19%**	**77.27%**	**94.06%**
Base-Q	4	8	**77.52%**	**94.37%**	**78.06%**	**94.55%**
JLQ-Q	4	8	77.15%	94.14%	77.38%	94.19%
Base-Q	4	32	**77.80%**	**94.56%**	**78.09%**	**94.47%**
JLQ-Q	4	32	77.35%	94.27%	77.63%	94.31%

D. Tiny ImageNet Dataset

In order to test whether JLQ can maintain the accuracy advantage in the case of extremely low-bit quantization (2-bit and 3-bit) on larger datasets, we conducted experiments based on the Tiny ImageNet dataset. The Tiny ImageNet dataset [11] is a modified subset of the original ImageNet dataset [21] with 200 different classes, 100,000 training examples and 10,000 validation examples. The resolution of the images is only 64x64 pixels, which makes it more challenging to extract information from it than the original ImageNet dataset. To get pre-trained models of Tiny ImageNet Dataset, we train the networks from scratch in 90 epochs based on Imagenet pre-trained weights. After that, we train the networks from those pre-trained models for 15 epochs (Other training parameters are consistent with those in ImageNet experiments mentioned in [8]). In addition, to increase the accuracy of the Resnet18 baseline, we use a fine-tuning method by removing the max pooling layer to reduce information loss of the image in the early stage of CNN. The accuracy results based on ResNet18, and Inception-v3 are shown in Table VI, and VII correspondingly.

It can be clearly seen that although the dataset becomes larger, compared to the baseline, JLQ still maintains the advantage of accuracy in a very low-bit quantization case.

TABLE VI
RESNET18 TINY IMAGENET BENCHMARK (SIGN=B)

Method	W	A	From scratch	
			Acc@1	Acc@5
Original	32	32	60.30%	82.10%
Base-Q	2	8	37.40%	62.54%
JLQ-Q(s=2,i=-3)	2	8	**57.32%**	**80.02%**
Base-Q	3	8	48.60%	73.51%
JLQ-Q(s=2,i=-1)	3	8	**58.36%**	**81.03%**

TABLE VII
INCEPTION-V3 TINY IMAGENET BENCHMARK (SIGN=B)

Method	W	A	From scratch	
			Acc@1	Acc@5
Original	32	32	68.09%	86.89%
Base-Q	2	8	10.08%	28.81%
JLQ-Q(s=2,i=-3)	2	8	65.98%	85.28%
Base-Q	3	8	59.83%	81.57%
JLQ-Q(s=2,i=-1)	3	8	66.37%	85.43%

V. HARDWARE DESIGN

A. Design Of Processing Element

The processing element serves as the core of neural network acceleration. Therefore, the optimization of a PE will directly affect the efficiency of the overall hardware design. A general weight-stationary shifter-based PE design is illustrated in Fig. 1. The shift operation of the feature and the weight is first conducted, then followed by a negative or positive value selection and zero value selection.

Since our quantization method does not have the weight of zero value, our PE removes a multiplexer in relation to zero bit and can save one multiplexer compared to traditional PE.

B. Design Of Shifter

Although JLQ extends the quantization range, it does not mean that it requires a larger shifter to achieve the corresponding quantization. By modifying the barrel shifter, our PE achieves the purpose of range extension without adding any hardware burden. Taking the 3-bit jumping log quantization with step = 2 i = -1 as an example, the logarithmic quantization weights, in this case, are as follows:($\pm 2^{-1}$,$\pm 2^{-3}$,$\pm 2^{-5}$,$\pm 2^{-7}$). These weight values will correspond to the shift operations of $\gg 1$, $\gg 3$, $\gg 5$, and $\gg 7$ respectively. These operations only entail the 2-bit barrel shifter shown in Fig. 2 instead of a traditional 3-bit barrel shifter to be computed. In this example shifter, the layer of multiplexers that operate right shift one bit is removed. To save hardware resources, some multiplexers are substituted by AND gates or removed (when become unnecessary). The first layer of AND gates and multiplexers represents the $\gg 4$ operation, and the second layer represents the $\gg 2$ operation. Besides, the input is pre-shifted 1 bit in advance. Consequently, $\gg 1$, $\gg 3$, $\gg 5$, $\gg 7$ operations can be achieved using this shifter.

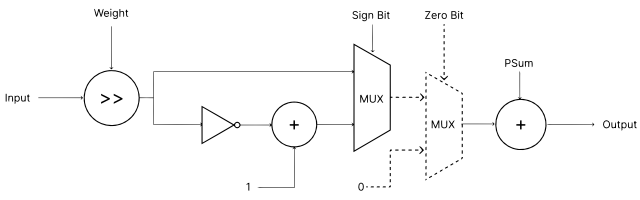

Fig. 1. Shifter-Based PE (Our PE is the solid part, and the dotted part is what we remove from traditional shifter-based PE in our optimization)

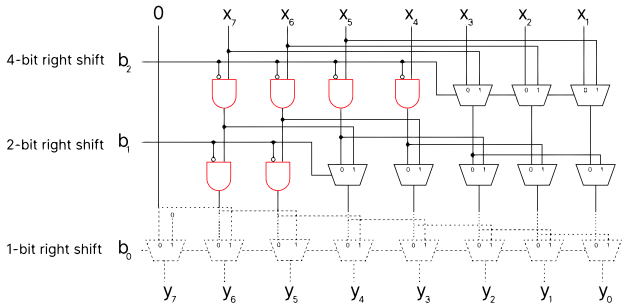

Fig. 2. Shifter Design (Our shifter is the solid part and the red part. The dotted part is what we remove from the traditional 3-bit barrel shifter in our optimization, and the red part indicates the replacement of multiplexers with AND gates. The first column of multiplexers can be directly removed due to the pre-shift operation).

TABLE VIII
RESOURCES CONSUMPTION IN SINGLE PE

PE type	W	LUT[a]	A(μm^2)[b]	P(mw)[b]
multiplier-based PE	3	55	221.186	1.3200
shifter-based PE	3	46	209.720	1.1928
our PE	3	33	203.840	1.1879
multiplier-based PE	2	43	200.900	1.2047
shifter-based PE	2	43	202.664	1.1567
our PE	2	28	192.178	1.1516

[a]For Xilinx ARTIX7 XC7A100T FPGA
[b]For TSMC 28 nm technology (ASIC)

C. Implementation Results

In order to calculate area and power consumption, different types of PE were synthesized using the Synopsys Design Compiler for TSMC 28 nm technology. The resource utilization is tested on Xilinx ARTIX7 XC7A100T. The basic resulting resources, area consumption, and power consumption for different PEs are given in Table VIII, and the normalized results of resource utilization and area are shown in Fig. 3 and Fig. 4 respectively.

It can be seen that our PE has advantages in hardware utilization, area, and power consumption over both traditional shifter-based PE and multiplier-based PE. Specifically, as shown in Fig. 3, our PE reduce the 35% and 28.6% LUT compared with traditional shifter-based PE in 2-bit and 3-bit quantization case respectively. As for area and power consumption, although the

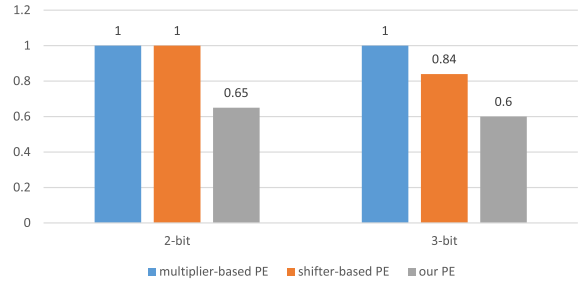

Fig. 3. Normalized LUT Resources

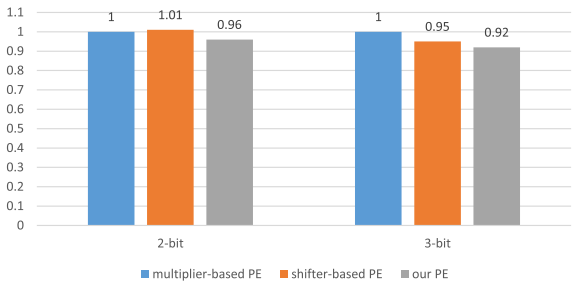

Fig. 4. Normalized Area

differences are not as prominent as those in resource utilization, our PE still performs better than traditional shifter-based PE.

VI. Conclusion

In this paper, we propose a quantization method called Jumping Logarithmic Quantization (JLQ), a weight de-zero optimization, and a cost-efficient PE design. In the experiments of CIFAR10, CIFAR100, and Tiny ImageNet, after integrating JLQ and weight de-zero optimization, the accuracy of the baseline has been greatly improved in both 2-bit quantization and 3-bit quantization. Implementation results show that our PE can maximally reduce the area and power consumption up to 19.7% and 17.2% compared with traditional multiplier-based PE under similar accuracy conditions.

VII. Future work

We have proven that jumping logarithmic quantization and weight de-zero optimization perform well on small datasets in the extreme low-bit quantization case. But the verification based on big datasets such as ImageNet [21] is not yet provided in this paper. It is expected to reach similar conclusions. In addition, in the aspect of hardware, putting our PE into a CNN accelerator to evaluate the overall throughput, resource utilization, and power consumption is yet to be done in our work. We expect a strong reduction in resource utilization and power consumption since JLQ is able to dramatically reduce the bit-width of the memory from the common-used bit down to 3-bit or 2-bit. Finally, the integration of three theoretically non-conflicting optimizations in logarithmic quantization, INQ [9], JLQ, and DeepShift [8], is also a very interesting future topic.

References

[1] M. C. M. C. S. D. W. A. N. e. a. Varun Gulshan, Lily Peng, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *JAMA*, vol. 316, no. 22, pp. 2402–2410, 12 2016.

[2] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[3] M. C. Chen, R. L. Ball, L. Yang, N. Moradzadeh, B. E. Chapman, D. B. Larson, C. P. Langlotz, T. J. Amrhein, and M. P. Lungren, "Deep learning to classify radiology free-text reports," *Radiology*, vol. 286, no. 3, pp. 845–852, 2018.

[4] I. S. A. Krizhevsky and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097 – 1105.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[7] E. Kalali and R. van Leuken, "A power-efficient parameter quantization technique for CNN accelerators," in *2021 24th Euromicro Conference on Digital System Design (DSD)*, 2021, pp. 18–23.

[8] M. Elhoushi, Z. Chen, F. Shafiq, Y. H. Tian, and J. Y. Li, "DeepShift: Towards multiplication-less neural networks," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 2359–2368.

[9] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[10] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[11] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets," *arXiv preprint arXiv:1707.08819*, 2017.

[12] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," *arXiv preprint arXiv:1603.01025*, 2016.

[13] D. A. Gudovskiy and L. Rigazio, "ShiftCNN: Generalized Low-Precision Architecture for Inference of Convolutional Neural Networks," *arXiv preprint arXiv:1706.02393*, 2017.

[14] C. Yang, B. Li, and Y. Wang, "A fully quantitative scheme with fine-grained tuning method for lightweight CNN acceleration," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2019, pp. 125–126.

[15] J. Xu, Y. Huan, L.-R. Zheng, and Z. Zou, "A low-power arithmetic element for multi-base logarithmic computation on deep neural networks," in *2018 31st IEEE International System-on-Chip Conference (SOCC)*, 2018, pp. 43–48.

[16] Y. Zhao, X. Gao, D. Bates, R. Mullins, and C.-Z. Xu, "Focused quantization for sparse CNNs," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] C. F. B. Fong, J. Mu, and W. Zhang, "A cost-effective cnn accelerator design with configurable pu on fpga," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 31–36.

[18] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," *arXiv preprint arXiv:1603.01025*, 2016.

[19] R. S. Jie Li, "Jizhong dianxing juanji shenjing wangluo de quanzhong fenxi yu yanjiu[Weight Analysis and Research of Several Typical Convolutional Neural Networks]," in *JOURNAL OF QINGDAO UNIVERSITY (Natural Science Edition)*, 2019.

[20] Y. Liu, X. Liu, and L. Liang, "Optimize FPGA-Based Neural Network Accelerator with Bit-Shift Quantization," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.