AdderNet 2.0: Optimal FPGA Acceleration of AdderNet with Activation-Oriented Quantization and Fused Bias Removal based Memory Optimization

Yunxiang Zhang, Omar Al Kailani, and Wenfeng Zhao
Binghamton University, Binghamton, NY, USA
{yzhan392,oalkail1,wzhao}@binghamton.edu

ABSTRACT

Convolutional neural networks (CNNs) are computationally demanding due to expensive Multiply-ACcumulate (MAC) operations. Emerging neural network models, such as AdderNet, exploit efficient arithmetic alternatives like sum-of-absolute-difference (SAD) operations to replace the costly MAC operations, while still achieving competitive model accuracy as the CNN counterparts. Nevertheless, existing AdderNet accelerators still face critical implementation challenges to achieve maximal hardware and energy efficiency. This paper presents AdderNet 2.0, an algorithm-hardware co-design framework featuring a novel Activation-Oriented Quantization (AOQ) strategy, a Fused Bias Removal (FBR) scheme for on-chip feature memory bitwidth reduction, and optimal PE designs to improve the overall resource utilization towards optimal AdderNet accelerator designs. Multiple AdderNet 2.0 accelerator design variants were implemented on Xilinx KV-260 FPGA. Experimental results show that the INT6 AdderNet 2.0 accelerators achieve significant hardware resource and energy savings when compared to prior CNN and AdderNet designs.

KEYWORDS

AdderNet, activation-oriented quantization, fused bias removal, FPGA acceleration.

ACM Reference Format:
Yunxiang Zhang, Omar Al Kailani, and Wenfeng Zhao. 2024. AdderNet 2.0: Optimal FPGA Acceleration of AdderNet with Activation-Oriented Quantization and Fused Bias Removal based Memory Optimization. In *Proceedings of (Design Automation Conference (DAC) '24).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Since the inception of AlexNet [1], deep learning algorithms such as convolutional neural networks (CNNs), have proliferated [2, 3] and achieved great success in many computer vision and machine intelligence tasks with deeper and wider architecture. The underlying multiply-accumulate (MAC) operations in CNNs are known to be computational demanding, memory-bounded, and power-hungry. Thereby, CNN accelerators are designed with massively-parallel processing elements (PEs) to speed up the workloads. It becomes more challenging when deploying deeper models to resource-constrained edge devices, where the growing number of model parameters and computations renders the CNN workloads overwhelming on embedded computing platforms and ASIC hardware.

Hardware-efficient CNN models, such as bottleneck architecture and point/depth-wise convolution [4, 5], have been devised to amortize the computational cost. Model pruning and quantization have been widely explored [6–9] to reduce the overall MAC operations and the arithmetic/memory bitwidth, such as INT8 [10] and INT4 [11]. Nevertheless, the MAC hardware is still inevitable for the aforementioned techniques. Several multiplication-less approaches have also been investigated, such as binarized neural networks [12], Deepshift [13] and ShiftAddNet [14], where costly multiplications are replaced with Boolean operations, bitwise shift and accumulation, sign flipping, at the cost of either accuracy degradation, or hardware overheads from high bitwidth and programmable shifters.

Recently, alternative MLDL models beyond multiplication have been proposed. One such example is AdderNet [15], which offers a promising hardware-efficient, multiplication-less, and highly accurate alternative to CNNs. AdderNet employs ℓ_1-norm kernel feature extraction, which corresponds to efficient sum-of-absolute-difference (SAD) hardware alternative to the MAC counterparts. Several AdderNet-based accelerators have been demonstrated on FPGA [16–18] and ASIC [19]. However, these designs are mostly limited to high model quantization bitwidth (INT16), and still sub-optimal in resource utilization. Recent work [20] introduced a weight-standardization-based quantization (WSQ) strategy to achieve INT8 SAD kernel and DSP-LUT co-packing of parallel SAD operations. Nevertheless, WSQ leads to model accuracy drop, leaving AdderNet quantization an outstanding challenge. Moreover, existing AdderNet accelerators still require high bitwidth on-chip memory for storing the SAD features of the adder layer.

This paper presents AdderNet 2.0, an algorithm-hardware co-design framework featuring activation-oriented quantization (AOQ) and fused-bias-removal (FBR) for addressing the aforementioned challenges in AdderNet. The main contributions are as follows:

- First, we propose a hardware-friendly activation-oriented quantization (AOQ) scheme to preserve the AdderNet model accuracy while still achieving low quantization bitwidth down to INT6;
- Second, we present a fused bias removal (FBR) technique and the corresponding PE design to minimize the on-chip memory bitwidth in AdderNet accelerator.

Design Automation Conference (DAC) '24, June 23–27, 2024, San Francisco, CA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

To demonstrate the efficacy of the proposed techniques of AdderNet 2.0, we validated model quantization performances using ResNet-20 and ResNet-50, developed and evaluated tile-based neural network accelerators targeting these models, all deployed on Xilinx Kria KV-260 FPGA. AdderNet 2.0 ensures improved inference accuracy down to INT6 as compared to existing INT10 and INT8 AdderNet designs, while it further leads to comprehensive hardware and energy savings with iso-throughput constraint. Compared with SOTA CNN designs, AdderNet 2.0 achieves significantly improved hardware and energy efficiencies (throughput per LUT/DSP/BRAM/energy) while exhibiting comparable inference accuracy.

The rest of this paper is organized as follows. Section 2 introduces the background and motivation. Section 2.2 presents the proposed AOQ scheme. Section 3 details the proposed FBR scheme. Section 4 covers AdderNet 2.0 accelerator implementation. Section 6 presents the experimental results and Section 7 concludes this paper.

2 BACKGROUND AND MOTIVATION

2.1 Preliminaries on AdderNet

Let $W \in \mathbb{R}^{k\times k\times c_i\times c_o}$ denotes the 4-D weight tensor in a convolutional layer of a CNN model, where k, c_i, and c_o are the filter kernel, input and output channel sizes, respectively. For a given input feature map $X \in \mathbb{R}^{h\times w\times c_i}$, where h and w are height and width of the input. The cross-correlation between the input and the 4-D weight tensor will yield an output Y:

$$Y(m,n,c_o) = \sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{l=1}^{c_i} X(m+i,n+j,l)\times W(i,j,l,c_i), \quad (1)$$

where m and n are the height and width of the output, and Y can be interpreted as similarity measures between input and convolutional filters.

AdderNet [15] offers an alternative solution to the convolution kernels by replacing multiplication with absolute difference operation, which can be regarded as the ℓ_1-norm distance metric between the input feature map and the filter kernels. Specifically, the output feature Y can be expressed as:

$$Y(m,n,c_o) = -\sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{l=1}^{c_i} |X(m+i,n+j,k) - W(i,j,l,c_i)|. \quad (2)$$

Hence, the expensive MAC operations in CNNs can be substituted by efficient SAD operations in AdderNet.

2.2 AdderNet Accelerator Design Challenges

2.2.1 Model Quantization. AdderNet suffers from unique quantization challenges as the well-established weight quantization methods for CNNs do not work well for AdderNet. Fig. 1(a) shows the top-1 inference accuracy versus different weight quantization bitwidth of CNN, AdderNet [15], and WSQ-AdderNet [20] based on ResNet-20 backbone and CIFAR-10 dataset. Unlike CNN in which weight and ReLU activation quantization are uncorrelated, ReLU activation and subsequent layer weights in AdderNet should use the same quantization step size (Δ) to ensure correct SAD operation. Moreover, AdderNet weights follow a Laplace distribution [15] with a zero-mean but significantly larger variance compared to the ReLU activation. In this way, a large Δ value facilitates lower weight quantization bitwidth, but this inevitably degrades model accuracy

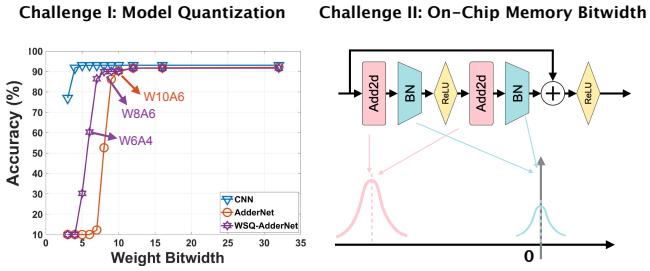

Figure 1: Design Challenges for AdderNet Accelerators.

caused by diminishing ReLU activations during quantization in AdderNet. Hence, baseline AdderNet weights can only be quantized down to 10-bit, which significantly offsets the hardware benefits of employing the SAD kernels. Alternatively, WSQ-AdderNet [20] achieves INT8 weight, in which weight standardization (WS) technique leads to approximately 4× reduction in weight dynamic range. Nevertheless, WS incurs model accuracy drop (around 1%) as compared to the baseline AdderNet. So far, weight-oriented quantization approaches are less scalable for AdderNet. Alternative quantization strategies are highly desired. In Section 3, we will introduce an innovative activation-oriented quantization (AOQ) scheme for AdderNet.

2.2.2 Feature Memory Optimization. The peak feature map memory bitwidth is another design challenge and has not been well optimized in previous AdderNet accelerators. Due to the cost function defined in Eq. (2), the intermediate features of the SAD kernel outputs continuously accumulate towards larger negative values, as illustrated in Fig. 1(b). This suggests that high memory bitwidth is required to store the negative values before batch normalization (BN) layer. This will also hamper the accelerator throughput in large neural network models (e.g., ResNet-50), as the features will be frequently moved between the off-chip DRAM and on-chip memories. Inspired by the fact that the negative SAD features in AdderNet will ultimately be normalized by the BN layer and eventually yield a zero-mean and smaller variance outputs, we will investigate a novel dataflow optimization strategy for feature memory bitwidth reduction in AdderNet. In particular, we propose a fused bias removal (FBR) technique to achieve this goal, as will be detailed in Section 4. The main idea is to get rid of the negative mean of the SAD features on-the-fly to achieve memory bitwidth reduction, which can be realized efficiently through dataflow optimization and PE hardware modification.

3 ACTIVATION-ORIENTED QUANTIZATION FOR ADDERNET

In this section, we detail our proposed activation-oriented quantization scheme for AdderNet quantization.

First, for a given layer, the **BN** outputs will follow a standard scalar quantization procedure, i.e., for a q-bit quantization, the quantized $\mathbf{BN}(\cdot)$ outputs are,

$$\mathbf{BN}(\cdot) \in [-2^{q-1}, 2^{q-1}-1], \quad (3)$$

and this will lead to a $(q$-1)-bit non-negative **ReLU** output,

$$\mathbf{ReLU}(\cdot) \in [0, 2^{q-1}-1], \quad (4)$$

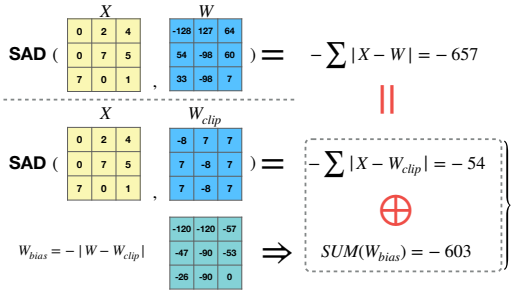

Figure 2: A conceptual illustration of Activation-Oriented Quantization with weight clipping.

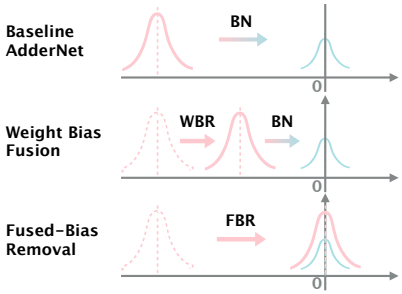

Figure 3: A conceptual illustration of the Fused Bias Removal (FBR) scheme.

for the subsequent Adder layer, quantized weights W is derived based on the original floating-point weights and Δ, which yields a bitwidth much higher than q. Under the AOQ scheme, the quantized signed integer weights will undergo a weight clipping process,

$$\mathbf{W}_{clip}(w) = \begin{cases} -2^{q-1}, & w < -2^{q-1} \\ w, & w \in [-2^{q-1}, 2^{q-1}-1] \\ 2^{q-1}-1, & w > 2^{q-1}-1 \end{cases} \quad (5)$$

In this way, the new clipped weights $\mathbf{W}_{clip}$ are quantized to q-bit. The difference between the original W and the clipped W_{clip} is defined as the weight bias,

$$\mathbf{W}_{bias} = \mathbf{W} - \mathbf{W}_{clip}, \quad (6)$$

Thanks to the SAD operation, below equations are mathematically equivalent:

$$-\sum|\mathbf{X}-\mathbf{W}| \equiv -\sum|\mathbf{X}-\mathbf{W}_{clip}| + \sum\mathbf{W}_{bias}, \quad (7)$$

in which the newly derived term $(\sum|\mathbf{X}-\mathbf{W}_{clip}|)$ is a q-bit clipped SAD operation. $\sum\mathbf{W}_{bias}$ is a constant for a given q when model training is done, which can be pre-computed without any hardware overheads.

An example of INT4 AOQ with a filter kernel size of 3×3 is illustrated in Fig. 2. First, a 4-bit (−8~7) scalar quantization is applied on the BN output, this will lead to a 3-bit (0~7) non-negative ReLU output. This will allow for the calculation of Δ, which will then be used for weight quantization. Afterward, the SAD operation can be performed between $\mathbf{W}$ and $\mathbf{X}$. As mentioned earlier, the quantized weights still have a larger bitwidth (≥4) than the activation due to the large dynamic range. Fortunately, such large weights are not compulsory for SAD computation, as illustrated in Fig. 2. Through weight clipping (WC) scheme, only the new clipped 4-bit weights $\mathbf{W}_{clip}$ is needed for the SAD operation, while $\sum\mathbf{W}_{bias}$ can be pre-computed (-603) and added back to the clipped SAD results (-54) to obtain the original SAD value (-657). It becomes clear that the proposed AOQ scheme ensures correct SAD operation with clipped weights and pre-computed weight bias, which paves the way towards practical and efficient AdderNet hardware implementation.

4 FUSED BIAS REMOVAL

Inspired by the BN layer functionality in AdderNet, we propose a fused bias removal (FBR) scheme to optimize the dataflow and reduce the feature memory bitwidth. The fused bias is contributed by the weight bias from AOQ and the residual BN bias, which will be fused and removed during the SAD accumulation process.

As discussed earlier, the intermediate SAD features in baseline AdderNet continuously accumulate towards a large negative value. In the subsequent BN layer, the negative SAD features will be normalized via learnable parameters α and β,

$$\mathbf{Y} = \alpha\frac{\mathbf{X}-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta = \frac{\alpha}{\sqrt{\sigma^2+\epsilon}}\mathbf{X} + (\beta - \alpha\frac{\mu}{\sqrt{\sigma^2+\epsilon}}), \quad (8)$$

where $\mathbf{X}$ and $\mathbf{Y}$ are the input and output of BN, μ and σ are the mean and variance of X, and ϵ is a hyper-parameter for model stabilization.

BN is pivotal for AdderNet as it will normalize the large and negative SAD features back to zero-mean and a standard deviation of 1. Hence, after model training, the scaling coefficient $\frac{\alpha}{\sqrt{\sigma^2+\epsilon}}$ is very small, while the BN bias $(\beta - \alpha\frac{\mu}{\sqrt{\sigma^2+\epsilon}})$ is a positive value to neutralize the large DC offset of the SAD features. However, existing AdderNet PE designs account for high number of SAD operations before the BN layer, thereby leading to large memory bitwidth for storing the SAD results.

The proposed FBR scheme involves two-stage optimization, as illustrated in Fig. 3. The first optimization refers to bias fusion, where the weight bias $\sum\mathbf{W}_{bias}$ derived from the AOQ scheme is fused into the BN bias. By default, SAD results are restored by adding $\sum\mathbf{W}_{bias}$ back to the clipped SAD results. In fact, this constant weight bias $\sum\mathbf{W}_{bias}$ is a part of the freezed running mean μ of the negative SAD features, which will eventually be removed in the BN layer. In FBR, we propose to move and fuse the weight bias $\sum\mathbf{W}_{bias}$ into BN bias $(\beta - \alpha\frac{\mu}{\sqrt{\sigma^2+\epsilon}})$, resulting in a fused residual BN bias $(\beta - \alpha\frac{\mu-\sum\mathbf{W}_{bias}}{\sqrt{\sigma^2+\epsilon}})$, which are mathematically equivalent.

Nevertheless, the clipped SAD operations still incur a relatively large negative values and requires BN to normalize the clipped SAD results. Hence, the second stage of FBR is to dynamically "remove" the fused bias $\mathbf{FB} = (\mu - \sum\mathbf{W}_{bias})$ by subtracting it on-the-fly during the SAD process. This requires modifying the existing SAD PEs with additional subtractors to support the bias removal. Hardware support for FBR will be detailed in the following section.

5 ADDERNET 2.0 FPGA ACCELERATOR

5.1 Accelerator Architecture

To demonstrate the effectiveness of the proposed techniques, we validate both AOQ and FBR using ResNet-20 and ResNet-50 as backbone networks. For both designs, we resorted to tile-based FPGA accelerator architecture, as shown in Fig. 4. For ResNet-20, we

Algorithm 1: HLS Pseudo Code for AdderNet 2.0 PE supporting Octo SAD and FBR

Input: **IF with size of** $T_{in} \times T_R \times T_C$
W with size of $T_{out} \times T_{in} \times K_R \times K_C$
FB $= (\mu - \sum \mathbf{W}_{bias})$ **with size of** T_{out}
Output: **OF with size of** $T_{out} \times T_R \times T_C$
FBR-Add2D (IF, W, OF)
for $(k_R = 1 : K_R, k_C = 1 : K_C)$ **do**
  for $t_R = 1 : T_R, t_C = 1 : T_C$ **do**
    # pipeline II=1
    for $t_{out} = 1 : T_{out}$ **do**
      # unroll
      for $t_{in} = 1 : T_{in}$ **do**
        # unroll
        R0 $= |\mathbf{IF}[t_{in}] - \mathbf{W}[t_{in}]|$
        R1 $= |\mathbf{IF}[t_{in}+1] - \mathbf{W}[t_{in}+1]|$
        R2 $= |\mathbf{IF}[t_{in}+2] - \mathbf{W}[t_{in}+2]|$
        R3 $= |\mathbf{IF}[t_{in}+3] - \mathbf{W}[t_{in}+3]|$
        R4 $= |\mathbf{IF}[t_{in}+4] - \mathbf{W}[t_{in}+4]|$
        R5 $= |\mathbf{IF}[t_{in}+5] - \mathbf{W}[t_{in}+5]|$
        R6 $= |\mathbf{IF}[t_{in}+6] - \mathbf{W}[t_{in}+6]|$
        R7 $= |\mathbf{IF}[t_{in}+7] - \mathbf{W}[t_{in}+7]|$
        R8 = R0 + R1, R9 = R2 + R3
        R10 = R4 + R5, R11 = R6 + R7
        R12 = R8 + R9, R13 = R10 + R11
        TEMP = R12 + R13
      end
      OF$[t_{out}] =$ **TEMP** $- \frac{\mathbf{FB}}{K_R \times K_C}$
    end
  end
end

choose to implement heterogeneous layer-wise PE designs tailored to support different layers including SAD/CONV, BN, ReLU, Pooling, and identity shortcut connection to maximize the hardware utilization efficiency. While for ResNet50, we choose to implement a homogeneous one-PE-for-all-layer architecture. These design choices are similar to those reported in previous literature [21–23]. Dedicated BN layers with 1×1 convolution will be designed for AdderNet with freezed running mean and variance after model training. In addition, double buffer technique is applied all accelerators to achieve maximal throughput, while additional pipeline will be inserted when appropriate to retain a high operating frequency.

5.2 FBR-SAD PE Design and DSP-LUT Co-packing

The proposed AOQ scheme is mostly agnostic to the underlying hardware, while the FBR scheme requires the hardware support to achieve mean removal. In particular, FBR can be achieved at different hardware granularities. For instance, it is intuitive to perform FBR for each absolute-difference operations, which is the most efficient way to dynamically eliminate the DC offset of SAD features. Nevertheless, this will lead to substantial PE size caused by the additional subtractions for each absolute difference computation.

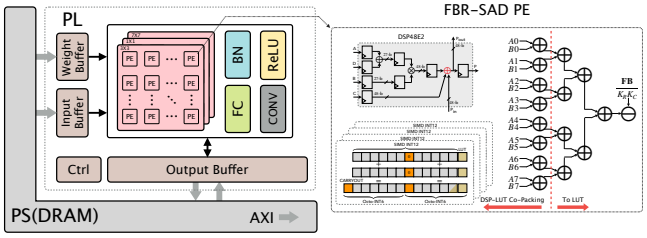

Figure 4: Tile-based PE architecture and FBR-PE with DSP-LUT co-packing.

Algorithm 1 shows the HLS pseudo code for our FBR-SAD PE design. Basically, our PE designs follows channel-wise unrolling to define the dimension of parallelism. With Octo-SAD PE and the subsequent adder-tree designs, the FBR implementation in this work will subtract $\frac{1}{K_R K_C}$ of the fused bias on-the-fly with only one additional subtractor. This will be achieved outside the output channel loop and ultimately mapped LUT slices. Note that **TEMP** and **OF** will be mapped to FF and BRAM slices during implementation. Hence, **TEMP** can still use a higher bitwidth during SAD computation, while **OF** will only use a lower bitwidth due to FBR. The corresponding assignment involves an explicit bitshift that can be automatically handled by HLS to ensure the correctness.

Similar to standard DSP packing strategies [20] to support SIMD SAD operations, we will leverage the DSP slices to achieve Octo-INT8 and Octo-INT6 SAD operations with one DSP48E2 slice and additional LUT slices, as shown in Fig. 4. Taking INT6 as an example, DSP48E2 slice will first be configured in Quad-INT12 mode. Then the INT12 adder will be further decomposed into one 6-bit adder and one 5-bit adder, with one guard bit used to ensure the correct addition of both adders [26]. The higher 6-bit adder will reuse the CARRYOUT bit of the INT12 adder, while the lower 5-bit adder will be augmented with LUT slices to form a 6-bit adder. Similar methodology will also be applied to INT8 packing, but with more LUT slices dedicated INT8 SAD PE compared with INT6 design.

6 EXPERIMENTAL RESULTS

6.1 Experimental Setup

All ResNet20 models were trained with FP32 precision on GeForce RTX 3090 GPU with CUDA 10.2 and PyTorch 2.0 on Ubuntu 20.04 TLS. The initial learning rate was 0.05, and the batch size was set to 256. The SGD optimizer was used with momentum as 0.9 and a weight decaying coefficient of $5e^{-4}$ for model training. An adaptive learning rule with cosine learning rate decay [27] was used. For ResNet-50, pre-trained model [15] was adopted. The proposed AOQ and FBR techniques were applied to both ResNet-20 and ResNet-50, with both INT8 and INT6 designs implemented for the proposed AdderNet 2.0 accelerators. For all accelerators, the working frequency of KV260 was set to 200 MHz. All accelerator designs were implemented in HLS C using Xilinx Vitis HLS and Vivado 2022.1. DSP-LUT co-packing was achieved by manually instantiating a DSP slice in the HLS generated Verilog RTL design of FBR-SAD PEs. The final design will be packaged as a Vivado IP to generate the bitstream. The chose hardware platform is Xilinx Kria KV260, which has 1,248 DSP, 117-kLUT, and 144 BRAM36 slices.

Table 1: Comparison of the proposed AdderNet 2.0 accelerator designs with the state-of-the-art CNN and AdderNet accelerators.

	ResNet-20 CNN [24]	ResNet-20 WSQ[20]	ResNet-20 This work	ResNet-50 BNN[25]	ResNet-50 CNN[8]	ResNet-50 (S) This work	ResNet-50 (L) This work
Bit-Width	W8A8	W8A6	W6A6	W1A1.4	W4A8	W6A6	W6A6
Top-1 Acc (%)	91.3	89.92	90.52	71.8	72.1	73.2	73.2
Device	KV260	KV260	KV260	ZU3EG	VU9P	KV260	KV260
Freq.(MHz)	274	200	200	250	200	200	200
kLUT	81.2	58.65	35.524	50.6	1,888	52.3	85.67
DSP	626	161	144	224	2527	176	400
BRAM36	73.5	68[a]	38	201	8,068	33.5	35
Latency (ms)	0.32	1.232	1.113	20.8	3.49	22.75	16.13
Thrpt. (GOPS)	616	214.6	214.6	-	2,873	262.4	620.08
Power (W)	3.61	1.695	0.763	6.1	47.7	0.923	1.092
GOPS/kLUT	7.58	3.65	6.4	-	1.52	5.01	7.24
GOPS/DSP	0.98	1.33	1.49	-	1.13	1.49	1.55
Energy Effi. (GOPS/W)	170.6	126.6	281.26	-	60.2	284.3	567.83

[a] The original reported BRAM does not consider overflow in different layers.

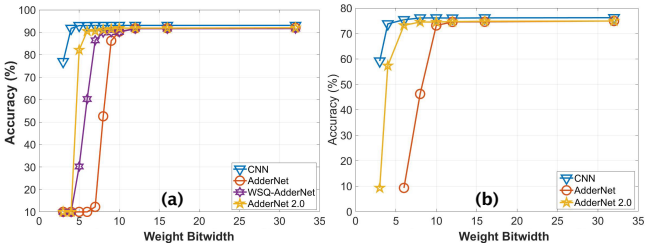

Figure 5: Top-1 inference accuracy of quantized CNN and AdderNets using ResNet-20 and ResNet-50 backbone.

6.2 Main Results

6.2.1 AdderNet 2.0 Quantization. Fig. 5(a) show the top-1 inference accuracy versus varying quantization bitwidth of CNN and AdderNets based on ResNet-20 backbone and CIFAR-10 dataset. As can be observed, CNN is quantization friendly and can preserve high inference accuracy for INT8 (93.1%) and INT4 (91.75%), respectively. Baseline AdderNet and WSQ-AdderNet can only retain acceptable accuracy for INT10 (90.3%) and INT8 (89.9%), respectively, while AdderNet 2.0 achieves improved inference accuracy for INT8 (91.1%) and INT6 (90.5%). Fig. 5(b) show the top-1 inference accuracy versus varying quantization bitwidth of CNN and AdderNets based on ResNet-50 backbone and ImageNet. Similarly, CNN achieves high inference accuracy for INT8 (76.1%) and INT4 (73.8%). Baseline AdderNet can only retain acceptable accuracy down to INT10 (73.15%). AdderNet 2.0 achieves improved inference accuracy on INT8 (74.5%) and INT6 (73.2%). In sum, AdderNet 2.0 leads to 4-bit bitwidth reduction and achieves improved inference accuracy as compared to baseline AdderNet. These results demonstrate the proposed AOQ scheme for AdderNet 2.0 outperforms existing weight-oriented quantization scheme.

6.2.2 AOQ and FBR effects on AdderNet Quantization. Fig. 6 shows an example of the statistics of adder layer weights/output and BN

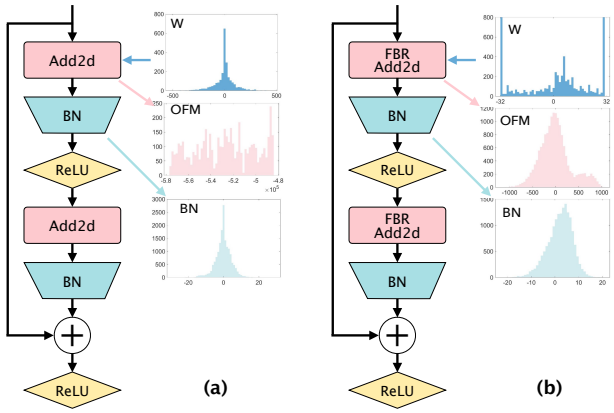

Figure 6: The statistics of weight, SAD feature, and BN output in (a) baseline AdderNet (INT10), and (b) AdderNet 2.0 (INT6). Both uses ResNet-20 as the backbone network.

layer output for baseline AdderNet and AdderNet 2.0, where the former adopts INT10 quantization and the latter adopts INT6 quantization. As can be observed, AOQ scheme yields clipped weights that have significantly smaller dynamic range as compared to the weights in baseline AdderNet. When fueled with the FBR scheme, the output of the FBR-Adder layer yields zero-mean features with significantly smaller dynamic range. As can be observed, the on-chip memory bitwidth for storing the SAD features is reduced from 19-bit in baseline AdderNet to only 11-bit in AdderNet 2.0, which is a significant amount of bitwidth reduction and will ultimately reflected via BRAM usage.

6.2.3 Accelerator Throughput and Resource. Table 1 summarizes major design parameters of AdderNet 2.0 accelerator design variants for ResNet-20 and ResNet-50, and those of the state-of-the-art (SOTA) CNN and AdderNet accelerators. For ResNet-20, our INT6 AdderNet 2.0 accelerator achieves 1.65× LUT, 1.12× DSP, and 1.79×

BRAM reduction as compared to the INT8 WSQ-AdderNet [20], while the model accuracy has improved by 0.6%. These advantages further translate into 2.2× energy efficiency improvements. Compared with a SOTA INT8 CNN accelerator [24], our INT6 AdderNet design shows 2.87× lower throughput. Nevertheless, the CNN design uses 2.28× LUT, 4.27× DSP, and 1.93× BRAM as compared to our design. Finally, our design shows 1.65× energy efficiency improvement. For ResNet-50, we implemented two INT6 AdderNet 2.0 accelerator designs with different numbers of PEs (32-256-1024 as Small (S), 32-1024-2048 as Large (L)). The small AdderNet 2.0 accelerator yields comparable resource utilization as that of a SOTA BNN (Binarized neural network with 1-bit weight and 1.4-bit activation) accelerator [25]. Still, the small accelerator achieves 1.27× DSP, 3× BRAM reduction, and 1.4% accuracy improvement. The large AdderNet 2.0 accelerator, which uses 22× less LUT, 6.3× less DSP, 230× less BRAM, still delivers merely 4.6× lower throughput, as compared to a SOTA W4A8 CNN [8]. Moreover, the energy efficiency is improved by 9.4×. These results clearly demonstrate the proposed approaches, including AOQ and FBR, lead to highly efficient AdderNet 2.0 accelerator designs in terms of hardware resource and energy savings, while still retaining high inference accuracy.

7 CONCLUSION

In this paper, we present AdderNet 2.0, an algorithm-hardware co-design framework to address model quantization and hardware utilization challenges. In particular, an optimal activation-oriented quantization technique and a fused bias removal based scheme for memory footprint reduction are proposed. The proposed AOQ technique ensures quantization down to INT6, while still retains high inference accuracy as compared to SOTA AdderNet quantization techniques. The FBR scheme further optimizes the feature memory bitwidth via optimized dataflow and hardware support for fused bias removal. When both techniques are employed and deployed for ResNet-20 and ResNet-50 FPGA accelerator designs, the optimized AdderNet 2.0 accelerator designs achieve substantial hardware resource savings (LUT/DSP/BRAM) when compared with SOTA CNN and AdderNet based designs. Hence, AdderNet 2.0 makes it possible to deploy large network models on extremely resource-constrained devices. In conclusion, AdderNet 2.0 base neural network accelerators can be a promising alternative to the widely adopted CNN-based accelerators for resource-limited platforms, yielding highly efficient designs.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[5] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[6] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[7] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[8] Wenqi Lou, Lei Gong, Chao Wang, Zidong Du, and Xuehai Zhou. Octcnn: A high throughput fpga accelerator for cnns using octave convolution algorithm. *IEEE Transactions on Computers*, 71(8):1847–1859, 2021.

[9] Junsong Wang, Qiuwen Lou, Xiaofan Zhang, Chao Zhu, Yonghua Lin, and Deming Chen. Design flow of accelerating hybrid extremely low bit-width neural network in embedded fpga. In *2018 28th international conference on field programmable logic and applications (FPL)*, pages 163–1636. IEEE, 2018.

[10] Xilinx. Wp486: Deep learning with int8 optimization on xilinx devices. *White Paper*, 2017.

[11] AmirAli Abdolrashidi, Lisa Wang, Shivani Agrawal, Jonathan Malmaud, Oleg Rybakov, Chas Leichner, and Lukasz Lew. Pareto-optimal quantized resnet is mostly 4-bit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3091–3099, 2021.

[12] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[13] Mostafa Elhoushi, Zihao Chen, Farhan Shafiq, Ye Henry Tian, and Joey Yiwei Li. Deepshift: Towards multiplication-less neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2359–2368, 2021.

[14] Haoran You, Xiaohan Chen, Yongan Zhang, Chaojian Li, Sicheng Li, Zihao Liu, Zhangyang Wang, and Yingyan Lin. Shiftaddnet: A hardware-inspired deep network. *Advances in Neural Information Processing Systems*, 33:2771–2783, 2020.

[15] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1468–1477, 2020.

[16] Yunhe Wang, Mingqiang Huang, Kai Han, Hanting Chen, Wei Zhang, Chunjing Xu, and Dacheng Tao. Addernet and its minimalist hardware design for energy-efficient artificial intelligence. *arXiv preprint arXiv:2101.10015*, 2021.

[17] Mingyong Zhuang, Xinhui Liao, Huhong Wu, Jianyang Zhou, and Zichao Guo. Vlsi architecture design for adder convolution neural network accelerator. In *2021 IEEE 15th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, pages 5–9. IEEE, 2021.

[18] Ke Ma, Lei Mao, and Shinji Kimura. A novel fpga-based convolution accelerator for addernet.

[19] Soyeon Um, Sangyeob Kim, Sangjin Kim, and Hoi-Jun Yoo. A 43.1 tops/w energy-efficient absolute-difference-accumulation operation computing-in-memory with computation reuse. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(5):1605–1609, 2021.

[20] Yunxiang Zhang, Biao Sun, Weixiong Jiang, Yajun Ha, Miao Hu, and Wenfeng Zhao. Wsq-addernet: Efficient weight standardization based quantized addernet fpga accelerator design with high-density int8 dsp-lut co-packing optimization. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

[21] Huimin Li, Xitian Fan, Li Jiao, Wei Cao, Xuegong Zhou, and Lingli Wang. A high performance fpga-based accelerator for large-scale convolutional neural networks. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–9. IEEE, 2016.

[22] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 161–170, 2015.

[23] Xuechao Wei, Yun Liang, Xiuhong Li, Cody Hao Yu, Peng Zhang, and Jason Cong. Tgpa: tile-grained pipeline architecture for low latency cnn inference. In *Proceedings of the International Conference on Computer-Aided Design*, pages 1–8, 2018.

[24] Filippo Minnella, Teodoro Urso, Mihai T Lazarescu, and Luciano Lavagno. Design and optimization of residual neural network accelerators for low-power fpgas using high-level synthesis. *arXiv preprint arXiv:2309.15631*, 2023.

[25] Yichi Zhang, Junhao Pan, Xinheng Liu, Hongzheng Chen, Deming Chen, and Zhiru Zhang. Fracbnn: Accurate and fpga-efficient binary neural networks with fractional activations. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 171–182, 2021.

[26] Jan Sommer, M Akif Özkan, Oliver Keszocze, and Jürgen Teich. Dsp-packing: Squeezing low-precision arithmetic into fpga dsp blocks. *arXiv preprint arXiv:2203.11028*, 2022.

[27] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.