FILM-QNN: Efficient FPGA Acceleration of Deep Neural Networks with Intra-Layer, Mixed-Precision Quantization

Rajasekhar Pittala , Ramesh Babu Pittala

Assistant Professor, Lakireddy Bali Reddy College of Engineering

Mylavaram

Professor,HOD,CSE KLR College of Engineering and Technology

Paloncha

rajasekhar.pittala@gmail.com, prameshbabu526@gmail.com

ABSTRACT

Quantization techniques have been extensively employed to decrease on-chip storage and increase computation speed in response to the growing need to deploy Deep Neural Network (DNN) inference models on edge devices with constrained resources. Existing DNN quantization- tion work that uses quantization below 8 bits may, however, see either a noticeable drop in accuracy or a sizable chasm between the theoretical gain in computing throughput and the practical increase in inference speed. To quantize and accelerate numerous DNN models across various embedded FPGA devices, we present a generic framework here dubbed FILM-QNN. To begin, we suggest a new approach for intra-layer mixed-precision quantization, in which the filters of various layers have varying degrees of accuracy. Based on our empirical research, we have identified the candidate precision levels and assignment granularity that will maintain accuracy while maximizing hardware parallelism. Second, in order to maximize throughput with the given resources, we use a number of optimization strategies for the FPGA accelerator design in favor of quantized computations. These strategies include DSP packing, weight reordering, and data packing. To further expedite the calculations with mixed precisions inside each layer, a complete resource model is built to balance the allocation of FPGA compute resources (LUTs and DSPs) as well as data transport and on-chip storage resources (BRAMs). Finally, we use Vivado High-Level Synthesis (HLS) on Xilinx PYNQ-Z2 and ZCU102 FPGA boards to make FILM-QNN more portable. Our experimental results on ResNet-18, ResNet-50, and MobileNet-V2 show that mixed-precision intra-layer implementations (95% 4-bit weights, 5% 8-bit weights, and all 5-bit activations) can achieve the same level of accuracy as the 8-bit (and 32-bit) versions, and the same level of throughput as the 4-bit designs (214.8 FPS, 109.1 FPS, and 537.9 FPS on ZCU102).

KEYWORDS

Accelerating hardware, FPGAs, mixed-precision quantization, compressed models, and deep learning

ACM Mengshu Sun, Zhengang Li, Alec Lu, Yanyu Li, Sung-En Chang, Xiaolong Ma, Xue Lin, and Zhenman Fang are some examples of a reference list. 2022. Using Intra-Layer, Mixed-Precision Quantization to Effectively Accelerate Deep Neural Networks on FPGAs. U.S.A., Virtual Event, February 27-March 1, 2022, Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '22). Twelve pages, published by ACM in New York.

In light of DNNs' widespread success, the technology is being used more and more for inference on edge devices such embedded FPGAs and ASICs. Model compression is a crucial process for DNN acceleration on these edge devices due to the increasing number of parameters and calculations required by complex DNN designs. Model quantization is a key model compression technique that has been extensively studied at both the algorithm-level [2-4, 6-10, 12, 17, 18, 22-26, 29, 34, 36, 39, 40, 44, 45, 49, 51, 54-56] and the hardware-level [5, 14-16, 20, 28, 30-32, 35, 42, 43, 48, 52]. Unfortunately, not many research have focused on improving hardware computation performance while still maintaining model correctness. However, several earlier works [6, 7, 18, 24, 25, 36, 56] have shown that low-precision (below 8 bits) quantization, such as binary, ternary, and 4-bit fixed-point quantization, may achieve high throughput. However, the drop in accuracy of their models becomes significant. However, a number of studies [7, 20, 24, 36, 56] inter-layer mixed-precision quantization, in which various layers are assigned different precisions, is an effort to reduce the accuracy loss. However, this results in poor hardware efficiency due to the fact that several inference layers now call for unique sets of resources and run in order. On an FPGA platform, for instance, the higher-precision first and final DNN layers would mostly use DSPs, whereas the lower-precision intermediate DNN levels would primarily use. Using look-up tables (LUTs) might result in less precise results. This means that either DSPs or LUTs are sitting idly by while a single DNN layer is being executed. We develop a generic and resource-efficient FPGA acceleration framework for Intra-Layer, Mixed-

Precision Quantized Deep Neural Networks (DNNs) called FILM-QNN, which allows us to simultaneously increase the inference throughput and the model's accuracy. Based on the observation that computations with different precisions within a layer run simultaneously, mixed-precision quantization (i.e., mixing higher preci- sion and lower precision within each layer) has the potential to not only preserve the model accuracy but also better utilize all types of FPGA resources for concurrent computations.

First, we propose the FILM-QNN, an intra-layer mixed-precision quantization algorithm that uses a combination of low-precision (e.g., 95% of 4-bit) weights, high-precision (e.g., 5% of 8-bit) weights, and moderate-precision (e.g., 5-bit) activations in each layer. During the DNN training process, we give weights with varying degrees of accuracy at the filter granularity in order to offer sufficient hardware par- allelism, with a focus on assigning high precision to weights that might cause large quantization errors. We also use different optimization strategies for quantized calculations in order to increase the computation throughput, and we pipeline the accelerator with parallelization along the input channel and output filter dimensions in the FPGA accelerator architecture. By efficiently packing multiple low-precision operations into a single DSP, 2) weight reordering arranges weights with the same precision together in each tile to eliminate the indexing overhead, and 3) data packing efficiently packs multiple low-precision weights (or activations) to widen their data width, saving on-chip storage and enhancing data access parallelism. Finally, we construct a thorough model to investigate the FPGA's resource allocation, which includes computing resources like DSPs and LUTs, and on-chip memory like Block RAMs (BRAMs), to speed up the calculations with varying degrees of accuracy in parallel for maximum throughput.

We have built FILM-QNN using Xilinx Vivado HLS and tested it with ResNet-18, ResNet-50, and MobileNet-V2 models on two embedded FPGA boards (Xilinx PYNQ-Z2 and ZCU102) to show how portable it is. Our optimized implementations with 95% 4-bit and 5% 8-bit intra-layer, mixed-precision quantization achieve accuracy of 70.47 percent, 77.25%, and 65.67 percent for the three models, respectively, and throughput of 27.8 frames per second on the PYNQ-Z2, 13.3 frames per second on the ZCU102, and 537.9 frames per second on the ZCU102, respectively, on par with the pure 4-bit precision designs.

1 RELATED WORK

{− + }

1.1 DNN Model Quantization

Accelerating DNN inference on edge devices, including embedded FPGAs and ASICs, requires the use of model quantization. The 8-bit quantization that can produce almost the same inference accuracy as 32-bit floating-point based DNN models has already been extensively utilized by industry. In this article, we focus on quantization methods with less than 8 bits of accuracy. {− + }

Uniform Extremely Low-Precision Quantization, Version 1.1.1. Binary and ternary quantization, where each network value only needs one or two bits, are the most common forms of uniform very low-precision quantization.

In binary quantization, a single bit represents a value of 1. Binaryconnect [6], Bina- rized Neural Network (BNN) [7], XNOR-net [36], and ABC-Net [25] are other illustrative works. In contrast, zero is used as an additional quantization level in ternary quantization in TWN [24], TTQ [56], and [18], where two bits represent values of 1, 0, 1. The problem with these uniform low-precision quantization approaches is that they lose a lot of accuracy at > 5% (in the case of binary quantization) and 2%-3% (in the case of ternary quantization). Mixed-Precision Quantization Across Layers. To preserve model fidelity in prior work using low-precision quantization [6, 17, 18, 37], it is typical to leave the first and final layer unquantized or to quantize them with no less than 8 bits. To further enhance the quantization accuracy, further research [8, 9, 39, 40, 44, 45, 49] investigated inter-layer, mixed-precision quantization techniques. Layer-wise precision assignment of weights and activations generates a large search space that has been explored using a variety of approaches and algorithms, including as HAQ [44], DNAS [45], and Mixed Precision DNNs [40]. More bits are allocated to layers that are sensitive to quantization errors via the HAWQ [9], HAWQ-V2 [8], PyHessian [49], and Q-BERT [39] algorithms, which solve Hessian matrices to determine the optimal bit- width assignment of each layer. As discussed in Section 1 and analyzed in Section 4, poor hardware efficiency is a potential drawback of inter-layer, mixed-precision quantization. *Mixed-Precision Quantization Within a Single Layer. Intra-layer mixed-precision quantization, which allows for varied precisions or schemes inside each layer, has been the subject of numerous studies [26, 34] that further elaborate on the notion of mixed-precision quantization. In order to improve model accuracy, RVQuant [34] suggests value-aware quantization, which uses low-precision quantization solely for sparse data in weights and activations. When compared to uniform low-precision quantization, it is not possible to achieve significantly faster inference times. AutoQ [26] uses reinforcement learning, which is computationally costly, to calculate the quantization accuracy within the kernel level.*

In this study, we investigate the possibilities of intra-layer, mixed-precision quantization to boost hardware efficiency. Unlike previous research, we aim to maintain model correctness while simultaneously increasing inference throughput. Quantized DNNs Boosted with FPGAs

Bit-level flexibility is increased in ASIC designs thanks to dynamic quantization with bit fusion [38] and striped bit-serial units [21], both of which are used to accommodate varying bit-widths in deep neural network layers. Model quant- tization is another technique that has found widespread use in DNN FPGA implementations [14]. The majority of the literature is dedicated to binary quantization, with calculations of binarized weights and activations being carried out using XNOR gates [16, 30, 31, 41, 53]. While the first convolutional layer keeps its 8-bit input, the work in [20] uses a two-stage arithmetic unit implemented by LUTs for 2-bit quantization. Using LUTs for shift-accumulation operations, [28] explores the use of power-of-two (PoT) quantized models. Various methods, such as the mixed-precision quantization presented in [43] for inter-layer 1-bit, 2-bit, 4-bit, and 8-bit weights, software-hardware co-design for 1-bit weights and 4-bit activations [48], a greedy approach [15] to

find the radix location of each layer, and automated production of RTL DNN components [52] for higher precision (8 and 16 bits). The bit-width of DNNs is not a restriction for the OpenCL-based deep learning accelerator described in [5].

To boost inference throughput without sacrificing model correctness, our work is unique in its emphasis on FPGA acceleration of DNNs using intra-layer, mixed-precision quantization. This is crucial since mapping the whole model (i.e. all layers) of big DNNs for on-chip processing concurrently is sometimes not achievable. While we utilize mixed-precision (high and low bit-width) quantization inside each layer, the closest work to ours is in [2], which uses mixed quantization methods (fixed-point and power-of-two quantizations). Additionally, in this work, we use a number of optimization strategies for low-precision computations on FPGAs and provide a thorough model to equilibrium the FPGA resource allocation. In Section 4.4, we'll show you how we stacked up numerically against [2].

High computing performance in FPGA implementations requires not just model quantization but also well-balanced use of available resources. For instance, to optimize performance for DNN designs with 16-bit fixed-point accuracy, a design space exploration tool is described in [35] to explore architectural options with different resource allocation of DSPs, on-chip memory, and off-chip bandwidth. The calculation for quantization bit-widths below 11, however, is always done in LUTs due to a lack of study for the LUT utilization. As a consequence, DSPs on FPGAs would be underutilized if used with quantization models of low precision (i.e., fewer than 8 bits). We simulate the calculation using LUTs and DSPs (and on-chip memory and off-chip bandwidth) with the intention of equitably distributing resources among various levels of accuracy while making full use of all available resources. In addition, we use the DSP packing approach described in [32, 42, 46, 47] to get the most out of digital signal processors. low-precision calculation efficiency.PROPOSED FILM-QNN FRAMEWORK

1.2 Accurate and Hardware-Friendly Intra-Layer, Mixed-Precision Quantization

quantization on a scale proportional to the bit width. Due to its greater accuracy performance, 8-bit quantization may be used as the high bit-width one in this case. This is why, even with uniform low bit-width quantiza- tion techniques [3, 4, 10, 12, 17, 22, 55], the first and final layers are always quantized to at least 8 bits, if not left unquantized altogether. We base our innovative intra-layer, mixed-precision quantization on the aforementioned findings. In order to maintain accuracy and decrease execution cost throughout the layer-by-layer inference computations, we suggest combining low and high bit-width quantization within layers and at an appropriate granularity. It's important to keep in mind that it's usually not feasible to transfer the whole model of big DNNs onto an FPGA and run all layers concurrently. To guarantee hardware parallelism, we fine-tuned the granularity of intra-layer quantization assignment to the filter level. To further ensure accuracy, we utilize a modest bit-width quantization for all activations throughout the whole DNN model.

In conclusion, our empirical analysis (with findings shown in Section 4.2) shows that, for our intra-layer, mixed-precision quantization technique, we use a total of three precision levels: To maintain accuracy and reduce execution overhead for inference acceleration, we use (I) low bit-width (4-bit) for the majority of the filters in a layer, (II) high bit-width (8-bit) for R = 5% of the filters in a layer, and (III) a moderate bit-width (5-bit) for all the activations throughout the model. In Section 4.2, we'll look at how changing the R value (the fraction of high-bit-width filters) affects the precision. To meet the aforementioned bit-width restrictions (I), (II), and (III), we quantize the model using a unique intra-layer, mixed-precision quan- tization strategy, as shown in Algorithm 1. We propose computing the quantization error E as a means of assigning accuracy to the filters (rows) of the weight tensor (matrix) of each convolutional (fully-connected) layer. Algorithm 1 combines the suggested precision assignment procedure with the quantization-aware Straight Through Estimator (STE) [1, 2]. The following are the specifics of our methodology.

For example, think of a deep neural network (DNN) of L layers. In this context, Wl stands for the weight tensor (matrix) of the l th layer. Al stands for the l th layer being active. In the weight tensor (matrix) Wl, the k-th filter (row) is denoted by Wk. The quantized weight and activation values in m-bit fixed-point quantization come from 1 2 To maintain the precision of quantized models and speed up inference, researchers have looked at mixed-precision quantization. However, the layer-by-layer inference execution on hardware platforms is incompatible with the current inter-layer techniques [8, 9, 39, 40, 44, 45, 49] because they assign various precisions to the DNN's layers. However, the current intra-layer methods [26, 34] use In this equation, V(m) is given by (1)where is a scaling factor: V(m) = 0, 2m1 1, 2m1 1,. Activation quantization uses a fixed value connected to the dataset, whereas weight quantization uses a number determined as the greatest absolute value of all weights in each layer. When converting from a floating-point to a fixed-point representation, the quantizer function is written as

DNN inference requires a more granular level of accuracy for assignment inside layers, which increases the execution cost. As a result, there is a significant chasm between the (theoretical) decrease in computation quantity and the (actual) increase in inference speed.

The following findings inspired us to develop our method of intra-layer mixed-precision quantization: 1) It's well knowledge that 8-bit quantized inference models can reliably maintain the same level of precision as full-precision models [13]. 2) When combined with a high-bit-width mixer, the accuracy loss brought on by low-bit-width quantization may be reduced.

1 **Convolution**(□, □ , □) {

```
2     for (□□ = 1 : □□, □□ = 1 : □□) // kernel
3       L1: for (□□ = 1 : □□, □□ = 1 : □□) { // feature map
4 #PIPELINE II=1
5           L2: for (□□ = 1 : □□) { // filter
6 #UNROLL
7             L3: for (□□ = 1 : □□) { // input channel
8 #UNROLL
9               □ [□] [□□] [□□] + =
                // assign quantization precision for eachfilter in the layer
6               foreach filter W□ in W□ do
7                 Ŵ□ ← Q_Hb(W□) if quan_error□ belongs to the
10              □ [□] [□□] [□□] [□□] × [□□] [□□ × □□ + □□] [□□ × □□ + □□];
11                                                                    }}}}
                top □ (5%) group;
8                 Ŵ ← Q_Lb(W□) otherwise;
```

□ shortcut connection). For FILM-QNN, we introduce the extra opti-mizations for quantized computations in this subsection, including

```
9               A ← Ŵ□ · Â□−1;
                // activations are quantized in Moderate
bit-width
10              Â ← Q_Mb(A□);
// backward propagation
11            foreach layer □ (reverse order) do
12              □□□□□ ← □□□□;
```

DSP packing, weight reordering, and data packing.

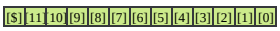
Base FPGA Accelerator. Without loss of generality, we use the most computation-intensive convolutional layer to demonstrate our FPGA design details. Given the limited FPGA resources, we apply the commonly used tiling technique in the accelerator implementation, where the accelerator only computes for a single tile of input feature maps, weight kernels, and output feature maps from

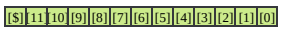
```
13              □W□
                □□□□□ ← ;
                □□□□□□
                ^       ^
□□□□□
Â□−1
```

a layer at a time. Algorithm 2 provides the pseudo code of the tiledconvolution accelerator design. Details are explained as follows.

```
14 Return M ← M{W}.
```

We use □□ (□), □□ (□□), □ □□ (□ □□), □□ (□□) and □□ (□□) to denote





We propose the quantization error of a filter W as:

$$E\ W_{\square}, \hat{W}^{\square}_{\square} = \|W^{\square}_{\square} \cdot \hat{A}_{\square-1} - \hat{W}^{\square}_{\square} \cdot \hat{A}_{\square-1}\|_2 \quad (4)$$

where $\|\cdot\|$ convolution procedure (Algorithm 2), which computes the tile of the Tiled convolution technique dimensions and explanations are shown in Table 1. During processing, the tiled input feature maps and weights are loaded in a burst fashion from off-chip memory into on-chip Block RAM (BRAM) buffers.

The number 2 denotes the standard for Language Level 2. The quantization error is the ratio of the full-precision filter's output feature maps to the quantized filter's output feature maps. The quantization errors of each filter in a layer are calculated as though they were quantized with a low bit-width (lines#4&5) in order to determine their respective levels of accuracy. High bit-width quantization is used for filters whose quantization errors fall within the top R (e.g., 5%) group; low bit-width quantization is used for all other filters (lines #6 and #8). Lines 4 and 8 of Algorithm 1 show how our accuracy assignment process is embedded into the STE technique, which employs rounded weights/activations in forward propagation and addresses the unavailable gradient problem in backward propagation by making the derivative of the rounding function equal to 1.

FPGA Accelerator Design and Optimization Our base FPGA accelerator design is similar to [50] with the sup- port of various types of DNN layers including convolutional layer, fully-connected layer, batch normalization layer, activation (ReLU) layer, and pooling layer, as well as skip connection (namely identity corresponding output feature maps. Lastly, the output tile data on BRAM is written back to the off-chip memory in a burst mode.

Inside the tiled convolution procedure (Algorithm 2), Loop L1 is pipelined with initial interval (II) of 1 to achieve computation parallelism of □□ □□ in Loops L2 (filter, or output channel di- mension) and L3 (input channel dimension). The buffered tiles (in BRAMs) are partitioned in the corresponding dimensions to sup- port the parallelism: the input tile □ is completely partitioned along the first dimension (input channel dimension), the output tile □ is completely partitioned along the first dimension (filter dimension), and the weight tile □ is completely partitioned along the first and second dimensions (both filter and input channel dimensions). In addition, double buffering is applied to pipelining the following stages: loading input and weight tiles, computing the convolution, and storing the output tile.

DSP Packing for Multiple Quantized Multiplications. In order to make the most of the DSP resources available on FPGAs, we stack many

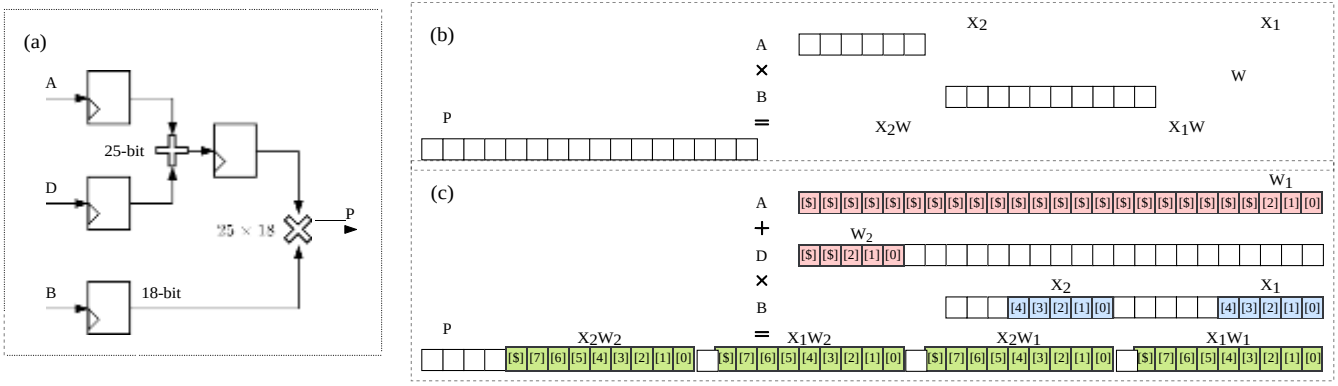


Figure 1: Multiple quantized multiplications supported on a single DSP48E1. (a) The 25 × 18-bit multiplication supported on DSP48E1; (b) Two 8 × 5-bit multiplications packed on DSP48E1; and (c) Four 4 × 5-bit multiplications packed on DSP48E1.

Table 1: Notations used in FILM-Q Opera

Notation	Description
□□ (□□)	Tiling size in output (input) channel

nd □, occupying the bits 4 : 0 and 18 : 14 , and one 8-bitsigned weight □ is presented in the 18-bit operand □ occupyingits bits 7 : 0 , where [$] denotes the sign of the weight. The two 13-bit multiplication results can be extracted from the bits 12 : 0 and 26 : 14 , respectively, in the 43-bit product □ of the DSP. Thebit 13 in the product □ is reserved for overflow. To reduce theLUT consumption for additions, only lower bits of these operandsand output are occupied.

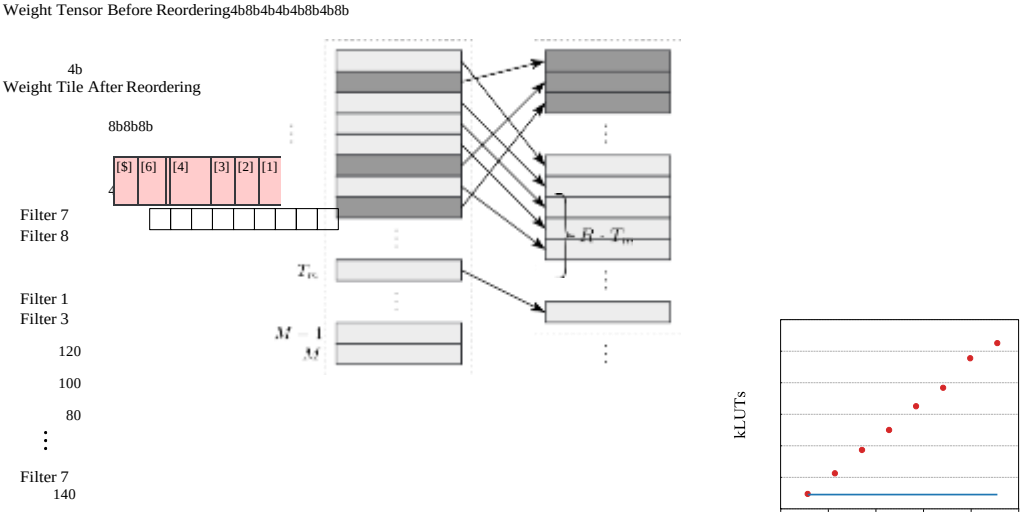
Two low-accuracy multiplications inside each DSP are given using 4-bit signed weights W1 and W2 in line with [46, 47] so that four 45 multiplications may be packed into a DSP48E1, as illustrated in Fig. 1(c). One 25 18-bit multiplication (and one 25-bit addition) is possible on the DSPs on the PYNQ-Z2 board, but on the ZCU102 board, one 27 18-bit multiplication is possible on the DSPs. We'll utilize the DSP48E1 (25 18-bit) to demonstrate our DSP packing technique. In our FILM-QNN, we employ weight precisions of 4, 8, and 5, and activation precisions of 5. For this, support for 4-bit and 8-bit multiplications is required. The computation P = A D B, where A and D are 25-bit operands, B is an 18-bit operand, and P is the 43-bit result, may be performed by a DSP48E1, as shown in Fig. 1(a). As shown in Figure 1(b) and Figure 1(c), a DSP48E1 can do two 8-bit 5-bit multiplications or four 4-bit 5-bit multiplications. Figure 1(b) shows how a DSP48E1 can fit the equivalent of two 8-bit 5-bit multiplications. The activations of X1 and X2 are shown as 25-bit representations.

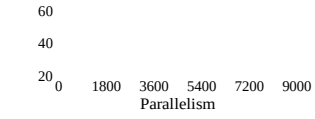
The weights' sign bits are appropriately extended and carried by bits 4:0 of operand A and bits 23:20 of operand D, while bits 4:0 and 14:10 of operand B provide two 5-bit activations X1 and X2 to the weights. The product P of the 43 bits used by the DSP is 9 bits for bits 8:0, 18:10, 28:20, and 38:30. The same way, one extra bit is saved between the four final outputs in case of overflow.

While we demonstrate DSP packaging on Xilinx FPGAs, similar approaches might be applied with Intel FPGAs, which have either 18 or 27 fixed-point DSPs with a bit depth of 18 or 27. [19].

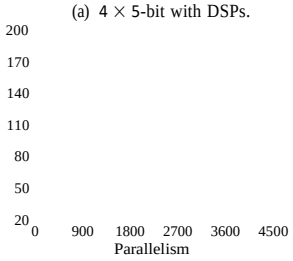
□□ (□□)	Tiling size in output feature height (width) dime
□□□ (□□□)	Tiling size in input feature height (width) dimen
□□ (□□)	Kernel size in height (width) dimension
□□ (□□)	Stride in height (width) dimension
□ 4×5,□□□ □□□ (□ 8×5,□□□ □□□)	Number of LUT used by a 4 × 5-bit (or 8 × 5-bit) multiplication executed on DSPs
□ 4×5 □□□ (□ 8×5 □□□)	Number of DSP (LUT) used by multiplications executed on DSPs (LUTs)
□	Number of low-bit data packed together
□ 4×5 □□□ (□ 8×5 □□□)	Number 4 × 5-bit (8 × 5-bit) multiplications executed on DSPs (LUTs)

Weight Reordering to Reduce Indexing Overhead. To decide which (output) filters in a layer should be assigned 8-bit quantization, we use an intra-layer, mixed-precision quantization technique. The R = 5% of High-bit (8-bit) filters, however, may be unevenly spread throughout the whole layer, as seen in Fig. 2's left side. Since distinct compute engines handle the 4-bit and 8-bit calculations independently (and simultaneously), the (random) indexing overhead to access the 8-bit and 4-bit filters in these compute engines may arise. We suggest the weight reordering method seen in Fig. 2 to lessen this indexing burden. Let's suppose a layer contains M filters in total, and those filters are spread out among Tm weight tiles. We rearrange the filter ordering for each weight tile such that the first R Tm filters are kept for the 8-bit quantized ones and the remainder of the filters in the tile are for the 4-bit quantized ones. The filters in this layer are rearranged to match the

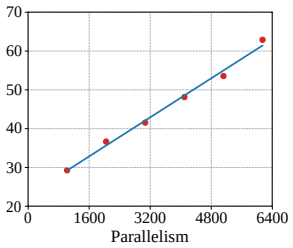


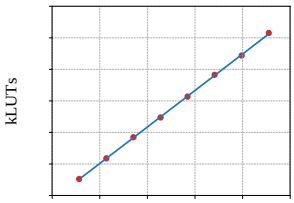


(a) 4 × 5-bit with DSPs.



(b) 4 × 5-bit with LUTs.





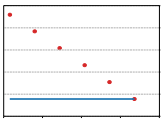
Filter 4b
Filter 4b
4b Filter T,,

220
140

Figure 2: Weight filter reordering after quantization. Eachfilter (block in the figure) has three dimensions with total

180

100



size of □□ × □□ × □□ .

3.3 The indexing cost for sequentially linked layers is removed by using this weight reordering strategy at compilation time before to conducting the DNN inference on hardware. DSPs provide for an 8-by-5-bit resolution. 20 0 800 1600 2400 3200

3.4 Simplicity (b)8 x 5 bits with look-up tables.

3.5 \ However, the filters in la and lb may be modified otherwise in the case of skip-connected layers. Loading activation outputs from layer la and layer lb for addition may cause layer lb's filter order to be out of sync with layer la's, but this may be avoided by preserving layer la's filter order relative to that of the reordered layer lb and then reloading the layers with the appropriate filter order. The indexing overhead for ResNet-18 was reduced from 67.97 kbit to 27.75 kbit (2.45), for ResNet-50 it was reduced from 323.0 kbit to 198.8 kbit (1.62), and for MobileNet-V2 it was reduced from 211.6 kbit to 9.28 kbit (22.8) by modifying the filter weights in each layer.In order to achieve data compression and parallelization, information packing is used. The low-precision data access in FILM-QNN (i.e., 4-bit and 8-bit for weights, and 5-bit for activations) may lead to under-utilization of BRAM capacity and over-use of the BRAM banks (for parallel port access) due to the much greater bit-width of a BRAM bank (18k bits). In order to decrease the amount of on-chip BRAM required and boost the parallelism of the hardware's data access, we use the data packing technique to merge several low-precision data items into a single big data item. The ideal data packing size, G, is determined by the processing speed and storage capacity of a certain FPGA board. Up to the limit of BRAM's relatively tiny bit width, this increases the on-chip bandwidth by a factor of G. The effective off-chip memory band-width is enhanced by a factor of G [27], thanks to the 64-bit width of the PYNQ-Z2 board and the 128-bit width of the ZCU102 board's off-chip memory access ports. Don't forget that Intel FPGAs might benefit from this data-packing method as well.

3.6 Input and output tiles' G channels' 5-bit activation values are combined into a single value. Tile I's values have been transformed from their original 5-bit Tn T in T in format into 5-bit Tn G T in T in information. Similar information to Tm G Tr Tc may be found in Tile O of the 5 G-bit output. For the W input channel, a lot of data is jammed in there. By combining each pair of 4-bit weights together, the buffer can store 8 bits of information when T wgt = Tm 2 1 R. It is easy to get the subset of packed data that is required for computations by using shifting and AND operations.

3.7 Figure 3: LUTs using ZCU102 parallelism.

1.2.1In (a) and (b), digital signal processors multiply 8 bits by 5 bits, while in (c) and (d), lookup tables do the same.Additional Layer Processing. It is possible that the hardware implementation of DNNs will allow the convolutional and fully-connected layers to share the processing kernel for matrix multiplication operations. Compared to matrix multiplication, the DNN operations of batch normalization, activation (ReLU), pooling, and skip connection (addition processes) are much more computationally efficient. These steps are executed before to saving the output buffers' results, therefore their impact on latency is negligible. The floating-point accuracy of a model's parameters is maintained during quantization via a batch normalization layer. The hardware implementation of batch normalization combines the weights and biases with the running mean and running variance parameters. The weights and biases of the relevant batch normalization layer (8-bit or 16-bit precision) are combined with the scaling factor in each convolutional layer (given in equation (1)) to preserve model integrity. In the presence of a skip connection, the results of this batch normalization layer and a preceding ReLU layer would be mixed. An optional pooling layer may be included between each comparison-performing ReLU layer to quantize the activation values. This means that the accuracy hit from representing the activation values using 5-bit integers is rather small.

3.8 FPGA Resource Allocation Modeling

Our current baseline FPGA accelerator design for quantized DNN models might benefit from further optimization of computation parallelism making use of available computational resources (i.e. DSPs and LUTs), on-chip memory (i.e. BRAM), and memory bandwidth. Even with specified computing resources (for example, DSPs or LUTs), we have trouble estimating how much the LUT will cost for a certain computation type (for example, 4 5-bit or 8 5-bit multiplication). What this entails

First, we'll go through how LUTs are used in each of the four examples presented in Fig. 3.

*dsp*As an example, consider (a) 4-by-5-bit multiplications using DSPs, (b) 4-by-5-bit multiplications using LUTs, and (c) 4-by-5-bit multiplications using both DSPs and LUTs. Since the slopes of the fitted lines equal 4 5 (C45 C45,dsp)Sdspdsp + C45 Slut lut (10), we can use these values to get the

LUT cost for

a 4 × 5-bit multiplication executed on DSPs or LUTs, denoted by . Similarly, we also derive , , and .

× ×

consumes
$qC^{4\times5}C^{4\times5} + C^{8\times5}C^{4\times5}$like accumulation. Next, we use a convolutional layer to show the details of our FPGA resource modeling and optimization, including the analysis on computing resources as well as off-chip data transfez and on-chip memory

Computation Resource Analysis. From the computation re- *Off-chip Data Transfer and On-chip Memory Analysis.* During the inference execution of the FPGA accelerator, weight need

to be transferred through burst access from the off-chip DRAM memory to the on-chip BRAM buffers. The transferred tiles include input tile with size , output tile with size , where . The solution is then expressed as off-chip data transfer or on-chip memory resources.

(8) off-chip data transfer or on-chip memory resources.

(14)

The above constraints (6) and (7) ensure the DSP and LUT utilization is under the allowable threshold, i.e., and with the total resource amounts denoted by and . And the con- straint (8) makes sure we have no less than = 5% for 8 5-bit multiplications. Depending on the characteristics of the target FPGA device, the final solution may converge to one of the four boundaryconditions, which is determined by the output combination of two factors: 1) whether the 8 5-bit multiplications can be more effi- ciently processed on DSPs or LUTs on FPGAs; and 2) whether the available DSP (or LUT) resources are sufficient to handle those 8 5-bit multiplications. For each of these four boundary conditions, the situation is described when one of the four parameters (,

where the BRAM block size of 18Kb is used and the BRAM usage foreach buffer rounds up to the nearest whole number ( ). Also notethat the double buffering technique is used to overlap computationswith off-chip memory accesses.

Constraint (13) is on off-chip data transfer, where , and are the number of computation cycles, weight transfer cycles, and input transfer cycles for one group of tiles, respectively.The output buffer transfer cycles are negligible and thus not in- cluded. , and are obtained as

, , and) is zero. The solution process for these four boundary conditions is similar. Due to space limitation, here we

(15)

only present the solution for the case where the 8 × 5-bit multiplication is more efficiently processed on LUTs than on DSPs, and the available LUT resources are sufficient to handle those 8 5-bitmultiplications, i.e.,

Where $\mathcal{S}_{port}$ denotes the bit-width of one AXI port.

3.3.1 Summary. Ideally, we need to jointly solve both problems.

4 (5) and (11) for optimized FPGA resource allocation. In fact, we

find that we only need to solve problem (5) and then check with

Provides a higher parallelism result for most of FPGAs. That is, the achievable parallelism degree in our quantized DNNs is generallybounded by the available computation resources on FPGAs, not the

$C^{4\times5}$ (5) and $C^{4\times5} - C^{4\times5,dsp}$ (11) for optimized FPGA resource allocation. In fact, we

$\frac{lut}{C^{8\times5}} \quad \frac{lut}{- C^{8\times5,dsp}}$

find that we only need to solve problem (5) and then check with

$\frac{S_{dsp} \cdot \gamma_{dsp}}{}$

$\cdot q \cdot S_{lut} \cdot \gamma_{lut}$

$\frac{4 \quad 5}{dsp}$

$\leq$

$q \cdot C^{4\times5,sp} + C^{8\times5}$

Provides a higher parallelism result for most of FPGAs. That is, the achievable parallelism degree in our quantized DNNs is generally bounded by the available computation resources on FPGAs, not the

EVALUATION

1.3 Experimental Setup

1.4 Different sized DNN models, including ResNet-18, ResNet-50, and MobileNet-V2, were used in our studies, and both model quantization and hardware implementations were tested for each. We improve accuracy by retraining the baseline models with FP32 precision, which we obtained by downloading them from the TorchVi- sion library [11]. Then, we apply intra-layer mixed-precision quantization to the weights, experimenting with a range of weight-to-activation bit depths (from 3 to 6 bits) and weight-to-activation bit-ratios (R = 0%, 3%, 5%, 10%, 20%, 100%). In the first 100 epochs, we find the 5% 8-bit filters, and in the next 50 epochs, we fine-tune the model parameters while keeping the precision constant.

1.5 ResNet-50 has an initial learning rate of 1.024, whereas ResNet-18 and MobileNet-V2 both start at 0.512. ResNet-50 has a batch size of 1024, whereas the other two models each have a batch size of 512. The models are trained using the SGD optimizer with the parameters momentum = 0.875 and weight decay = 1 32768. At the outset of the quanti- zation, 8 epochs of warmup training are executed, and label smoothing is applied at a factor of 0.1 in order to increase accuracy. According to NVIDIA [33], these are the optimal settings for learning how to classify images. Model quantization is executed on 8 GeForce RTX 2080 Ti GPUs running Ubuntu 18.04 with the CUDA 10.2 and PyTorch 1.5 frameworks.

1.6 The quantized models are then tested on two distinct embedded FPGA platforms (Xilinx PYNQ-Z2; i.e., XC7Z020) and ZCU102 to show that our framework is applicable to a wide variety of FPGA boards. In comparison to the ZCU102 board's 2520 DSPs and 274.1k LUTs, the PYNQ-Z2 board's total of 220 DSPs and 53.2k LUTs seems somewhat little. Working frequencies of 100 MHz for PYNQ-Z2 and 150 MHz for ZCU102 were chosen for all DNN models to provide maximum computation efficiency without time violation. On PYNQ-Z2, we use a data packing size of G = 6, whereas on ZCU102 we use G = 8. Xilinx Vivado 2020.1 is used for the high-level synthesis of the hardware designs.

1.7 Accuracy Results

We undertake verification tests using ResNet-18, as shown in Tables 2 and 3, to back up the empirical conclusions we reported in Section 3.1. In the first row of the table, we see the precision of the unquantized baseline model from the TorchVision collection [11]. Our model, which we've fine-tuned to reach 70.78% and 89.94% in top-1 and top-5 accuracy, respectively, is included in the final row for comparison purposes given that the official supplied model may utilize different training settings.

1.7.1 Accuracy Findings when R is a Mixed Ratio. Table 2 shows a comparison of the model's accuracy when the mixed ratio R is varied while the activation bit-width remains fixed at 5 bits. First, when R = 0% (all weights are 4-bit), the top-1 model accuracy is 69.63% and the top-5 model accuracy is 89.28%. In other words, the top-1 accuracy drops by more than 1% when using 4-bit quantization. Second, the top-1/top-5 accuracy is increased to 70.60%/89.83% when R equals 100%, i.e., when all weights utilize 8-bit, which is very close to the same as the fine-tuned full precision model. Third, the top-1 accuracy of the model reaches 70.01%, 70.47%, 70.45%, and 70.56% for R = 3%, 5%, 10%, and 20%, respectively.**Table 2: Comparisons of ResNet-18 model accuracy under different percentages of 8-bit weights on ImageNet dataset**

Quantization Method	Weight Bit-width	Activation Bit-width	Top-1 Accuracy	Top-5 Accuracy
Baseline	32b	32b	69.57%	89.24%
Retrain	32b	32b	70.78%	89.94%
W4A5	4b	5b	69.63%	89.28%
Mixed	97% 4b + 3% 8b	5b	70.01%	89.33%
Mixed	95% 4b + 5% 8b	5b	70.47%	89.63%
Mixed	90% 4b + 10% 8b	5b	70.45%	89.55%
Mixed	80% 4b + 20% 8b	5b	70.56%	89.59%
Mixed (Inter)	4b + 8b	5b	69.92%	89.37%
W8A5	8b	5b	70.60%	89.83%

Table 3: Comparisons of ResNet-18 model accuracy under different bit-widths of activations on ImageNet dataset

Quantization Method	Weight Bit-width	Activation Bit-width	Top-1 Accuracy	Top-5 Accuracy
Baseline	32b	32b	69.57%	89.24%
Retrain	32b	32b	70.78%	89.94%
Mixed	95% 4b + 5% 8b	3b	68.91%	89.10%
Mixed	95% 4b + 5% 8b	4b	69.98%	89.38%
Mixed	95% 4b + 5% 8b	5b	70.47%	89.63%
Mixed	95% 4b + 5% 8b	6b	70.51%	89.64%

1.8 respectively, demonstrating that R = 5% may make effective use of the advantages offered by both greater precision and fewer bits' worth of weight. When R is increased by more than around 5 percentage points, there is no discernible gain in precision. To simplify the design of a single hardware accelerator that can be utilized by all layers, we use the same R = 5% value throughout. This already yields a very high degree of precision, as shown in Table 2. A little improved accuracy may be achieved by tuning a distinct R value for each layer, but this would result in inefficient hardware implementation that would be difficult to reuse.

1.9 Comparison of Accuracy at Various Activation Bit-Widths (1.7.1). Table 3 displays our findings as we investigate the effect of activation bit-width. We evaluate the robustness of the model over a range of activation bit-widths, from 3 bits to 6 bits, while maintaining a mixed-weight precision of R = 5% throughout. The results demonstrate that the model's accuracy improves with increasing activation bit-width, albeit the rate of improvement slows with increasing activation bit-width. Top-1 / top-5 accuracy reaches 70.47 / 89.63 percent when the bit-width of activation is 5 bits, which is quite similar to the result obtained with 6-bit activation. Because of the hardware efficiency gains, we choose for 5-bit activations in our FPGA studies rather than 6-bit ones.

1.10 To show that "5- bit activations and R = 5% 8-bit weights mixed with 4-bit weight" is a powerful combination, we just rely on the verification tests. one does not imply that one model is superior to others. It's important to note that our system works with any bit-widths for weights and activations. The framework's accessibility is unaffected by a shift in the bit-width combination.

Overall Throughput and Resource Results To better understand the performance of FILM-QNN, in Table 4, wecompare its peak throughput and overall resource utilization on

Table 4: Resource utilization and optimized computation throughput under different weight quantization precisions

Quantization Precision	Computation Resource	Implementation on ZCU102 (150 MHz)							Implementation on PYNQ-Z2 (100 MHz)						
		Utilization		W4A5 Op[a]		W8A5 Op		Peak Thrpt. (GOPS)	Utilization		W4A5 Op		W8A5 Op		Peak Thrpt. (GOPS)
		LUT	DSP	LUT	DSP	LUT	DSP		LUT	DSP	LUT	DSP	LUT	DSP	
100% W4	LUT	78%	1%	10240	0	-	-	1536	81%	13%	1728	0	-	-	172.8
	DSP														
	LUT + DSP	51%	82%	0	16384	-	-	2458	56%	95%	0	1440	-	-	144
		68%	78%	2048	15360	-	-	2611	77%	95%	576	1440	-	-	201.6
100% W8	LUT	76%	1%	-	-	6656	0	998.4	75%	13%	-	-	1152	0	115.2
	DSP														
	LUT + DSP	23%	82%	-	-	0	8192	1229	31%	95%	-	-	0	720	72
		65%	83%	-	-	3072	8192	1690	75%	95%	-	-	576	720	129.6
95% W4 + 5% W8	LUT	76%	2%	8192	0	1024	0	1382	81%	13%	1584	0	144	0	172.8
	DSP	54%	83%	0	14336	0	1024	2304	49%	95%	0	1152	0	144	129.6
	LUT + DSP	66%	83%	0	16384	1024	0	2611	78%	95%	720	1152	0	144	201.6

[a]The number of operations per cycle for 4-bit weights (W4) and 5-bit activations (A5). W8 means 8-bit weights.

Table 5: Comparisons of DNN implementations between previous studies, inter-layer quantization, and intra-layer quantization for ImageNet dataset on PYNQ-Z2 (XC7Z020) FPGA

Implementation	VGG [15]	ResNet-18 [2]	MobileNet-V2 [2]	ResNet-18	ResNet-50	MobileNet-V2	ResNet-18	ResNet-50	MobileNet-V2
				(Inter-Layer)			**(Intra-Layer)**		
Bit-Width	W8A8	W4A4		Middle W4A5, First & Last W8A5			95% W4A5 + 5% W8A5		
Top-1 Accuracy	67.62%	70.27%	65.64%	69.92%	77.08%	64.38%	70.47%	77.25%	65.67%
Frequency (MHz)	214	100		100			100		
kLUT	29.9 (56%)	28.3 (53%)		39.1 (74%)			41.3 (78%)		
DSP	190 (86%)	220 (100%)		214 (97%)			208 (95%)		
BRAM36	85.5 (61%)	56 (40%)		126.5 (90%)			123 (88%)		
Power (W)	3.5	-	-	3.0			3.5		
Frame Rate (FPS)	2.72	21.3	120.7	12.9	7.8	49.2	27.8	13.3	132.3
Throughput (GOPS)	84.3 (CONV)	77.0	71.8	46.8	63.6	29.3	100.6	108.6	78.7
GOPS/kLUT	2.825	2.725	2.538	1.197	1.627	0.749	2.436	2.629	1.907
GOPS/DSP	0.444	0.350	0.326	0.219	0.297	0.137	0.484	0.522	0.379
Energy Efficiency (GOPS/W)	24.1	-	-	15.6	21.2	9.8	28.7	31.0	22.5

Different quantization accuracies and resource allocation algorithms were applied to PYNQ-Z2 and ZCU102 FPGA boards. It is important to remember that the data packing optimization provided in Section 3.2.3 allows FILM-QNN to scale indefinitely, as predicted by our modeling and proven by the resource use in Table 5. Since the multiply-accumulate (MAC) operations are comparable to two separate operations, we focus on them since they need the greatest processing power in DNNs. We investigate the computing capabilities when 1) only using LUTs for MAC operations, 2) only using DSPs for MAC operations, and 3) utilizing both LUTs and DSPs for each quantization precision (4-bit only, 8-bit only, and 95% 4-bit and 5% 8-bit for the weights).

Operations with 4-bit weights (W4A5 Op columns) and 8-bit weights (W8A5 Op columns) list the attainable parallelism degree on each kind of FPGA computation resources, i.e. the number of operations handled by LUTs or DSPs each cycle. For each implementation, we derive its peak throughput in Giga Op- erations Per Second (GOPS) based on its degree of complete parallelism and its operating frequency. Using 95% 4-bit weight quantization and 5% 8-bit weight quantization in optimal implementations, peak performance was 2611 GOPS on ZCU102 and 201.6 GOPS on PYNQ-Z2. Using both LUTs and DSPs on ZCU102 results in 1.89x and 1.13x greater throughput than using LUTs solely or DSPs exclusively, respectively, within the 95% 4-bit and 5% 8-bit weight quantization, respectively.

These enhancements on PYNQ-Z2 are 1.17 and 1.56, respectively. Similarly, when working with weight quantization precisions of just 4 bits or 8 bits, employing both LUTs and DSPs gives higher throughput than using either one alone. Reason being, in mixed-precision implementations, LUT and DSP usage ratios are greater and more balanced than in systems using just LUTs or only DSPs for calculations.

In general, we find that it is preferable to execute as many MAC operations as possible using the FPGA's DSP capabilities before resorting to using the LUTs. Using DSPs for MAC operations also results in the consumption of LUTs owing to data packing, with the LUT consumption per DSP being around 2 times greater for W4A5 operations than for W8A5 operations due to the fact that a single DSP may do four W4A5 multiplications or two W8A5 multiplications, respectively. Our 95% 4-bit and 5% 8-bit weight quantization provides the best peak throughput, matching the 4-bit alone weight quantization on PYNQ-Z2 and ZCU102, among varied quantization precisions employing LUTs and DSPs. Throughput on PYNQ-Z2 is increased by 1.56, while throughput on ZCU102 is increased by 1.54, as compared to the 8-bit alone weight quantization. The throughput of our mixed-precision weight quantization and the 4-bit only weight quantization are same.

Table 6: Comparisons of DNN implementations between inter-layer and intra-layer quantization for ImageNet dataset on ZCU102 FPGA

Implementation	ResNet-18	ResNet-50	MobileNet-V2	ResNet-18	ResNet-50	MobileNet-V2
	(Inter-Layer)			(**Intra-Layer**)		
Bit-Width	Middle W4A5, First & Last W8A5			95% W4A5 + 5% W8A5		
Top-1 Accuracy	69.92%	77.08%	64.38%	70.47%	77.25%	65.67%
Frequency (MHz)	150			150		
kLUT	174.5 (64%)			180.1 (66%)		
DSP	2096 (83%)			2092 (83%)		
BRAM36	439 (48%)			440.5 (48%)		
Power (W)	13.4			12.9		
Frame Rate (FPS)	72.8	47.4	190.1	214.8	109.1	537.9
Thrpt. (GOPS)	263.7	387.8	113.3	778.9	891.4	320.1
GOPS/kLUT	1.511	2.222	0.649	4.324	4.948	1.777
GOPS/DSP	0.126	0.185	0.054	0.372	0.426	0.153
Energy Efficiency (GOPS/W)	19.7	28.9	8.5	60.4	69.1	24.8

quantization since the FPGA board could not accommodate more computations due to the routing issue.

1.11 Comparison with Prior Studies andInter-Layer Quantization

Different quantization accuracies and resource allocation algorithms were applied to PYNQ-Z2 and ZCU102 FPGA boards. It is important to remember that the data packing optimization provided in Section 3.2.3 allows FILM-QNN to scale indefinitely, as predicted by our modeling and proven by the resource use in Table 5. Since the multiply-accumulate (MAC) operations are comparable to two separate operations, we focus on them since they need the greatest processing power in DNNs. We investigate the computing capabilities when 1) only using LUTs for MAC operations, 2) only using DSPs for MAC operations, and 3) utilizing both LUTs and DSPs for each quantization precision (4-bit only, 8-bit only, and 95% 4-bit and 5% 8-bit for the weights).

Operations with 4-bit weights (W4A5 Op columns) and 8-bit weights (W8A5 Op columns) list the attainable parallelism degree on each kind of FPGA computation resources, i.e. the number of operations handled by LUTs or DSPs each cycle. For each implementation, we derive its peak throughput in Giga Op- erations Per Second (GOPS) based on its degree of complete parallelism and its operating frequency. Using 95% 4-bit weight quantization and 5% 8-bit weight quantization in optimal implementations, peak performance was 2611 GOPS on ZCU102 and 201.6 GOPS on PYNQ-Z2. Using both LUTs and DSPs on ZCU102 results in 1.89x and 1.13x greater throughput than using LUTs solely or DSPs exclusively, respectively, within the 95% 4-bit and 5% 8-bit weight quantization, respectively.

These enhancements on PYNQ-Z2 are 1.17 and 1.56, respectively. Similarly, when working with weight quantization precisions of just 4 bits or 8 bits, employing both LUTs and DSPs gives higher throughput than using either one alone. Reason being, in mixed-precision implementations, LUT and DSP usage ratios are greater and more balanced than in systems using just LUTs or only DSPs for calculations.

In general, we find that it is preferable to execute as many MAC operations as possible using the FPGA's DSP capabilities before resorting to using the LUTs. Using DSPs for MAC operations also results in the consumption of LUTs owing to data packing, with the LUT consumption per DSP being around 2 times greater for W4A5 operations than for W8A5 operations due to the fact that a single DSP may do four W4A5 multiplications or two W8A5 multiplications, respectively. Our 95% 4-bit and 5% 8-bit weight quantization provides the best peak throughput, matching the 4-bit alone weight quantization on PYNQ-Z2 and ZCU102, among varied quantization precisions employing LUTs and DSPs. Throughput on PYNQ-Z2 is increased by 1.56, while throughput on ZCU102 is increased by 1.54, as compared to the 8-bit alone weight quantization. The throughput of our mixed-precision weight quantization and the 4-bit only weight quantization are same.

2 CONCLUSIONS

In this research, we provide FILM-QNN, a generic framework for quantizing and speeding up DNNs on FPGAs. We suggest quantizing each layer of a DNN with the majority of weight filters as 4-bit and the minority of weight filters as 8-bit to maintain accuracy and hardware parallelism. We use DSP packing, weight reordering, and data packing, all of which are optimization approaches for low-precision calculations, in the design of our FPGA accelerator. To further facilitate the simultaneous calculations at varying degrees of accuracy, we provide a comprehensive model to direct the fair allocation of FPGA resources. We have also used Vivado HLS to evaluate FILM-QNN on three other DNN models running on two different Xilinx FPGA platforms (PYNQ-Z2 and ZCU102): ResNet-18 and ResNet-50, and MobileNet-V2. The accuracy of our optimized mixed-precision implementations (70.47%, 77.25%, and 65.67% for the three models, respectively) is equivalent to that of the 8-bit precision designs, while the throughput (214.8 FPS, 109.1 FPS, and 537.9 FPS on ZCU102) is comparable to that of the 4-bit precision implementations.

ACKNOWLEDGMENTS

NSF CCF-1901378, NSERC Discovery Grant RGPIN-2019-04613, NSERC Discovery Grant DGECR-2019-00120, Alliance Grant ALLRP-552042-2020, and CFI John R. Evans Leaders Fund have all contributed to this study.

REFERENCES

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).
[2] Sung-En Chang, Yanyu Li, Mengshu Sun, Runbin Shi, Hayden K-H So, XuehaiQian, Yanzhi Wang, and Xue Lin. 2021. Mix and Match: A novel FPGA-centric deepneural network quantization framework. In *2021 IEEE International Symposiumon High-Performance Computer Architecture (HPCA)*. IEEE, 208–220.
[3] Gong Cheng, Lu Ye, Li Tao, Zhang Xiaofan, Hao Cong, Chen Deming, and Chen Yao. 2019. □L2Q: An Ultra-Low Loss Quantization Method for DNN. *The 2019 International Joint Conference on Neural Networks (IJCNN)* (2019), 1–8.
[4] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085* (2018).
[5] Philip Colangelo, Nasibeh Nasiri, Eriko Nurvitadhi, Asit Mishra, Martin Margala,and Kevin Nealis. 2018. Exploration of low numeric precision deep learning inference using intel® FPGAs. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 73–80.
[6] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binarycon-nect: Training deep neural networks with binary weights during propagations.In *Advances in neural information processing systems (NeurIPS)*. 3123–3131.
[7] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*(2016).
[8] Zhen Dong, Zhewei Yao, Yaohui Cai, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2019. HAWQ-V2: Hessian Aware trace-Weighted Quantization of neural networks. *arXiv preprint arXiv:1911.03852* (2019).
[9] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2019. Hawq: Hessian aware quantization of neural networks with mixed- precision. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 293–302.
[10] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. 2019. Learned step size quantization. *International Conference on Learning Representations (ICLR)* (2019).
[11] Facebook. 2021. Torchvision. https://pytorch.org/vision/stable/models.html Last accessed Sept 12, 2021.
[12] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. 2019. Differentiable soft quantization: Bridging full- precision and low-bit neural networks. In *Proceedings of the IEEE InternationalConference on Computer Vision (ICCV)*. 4852–4861.
[13] Google. 2021. TensorFlow. https://www.tensorflow.org/lite Last accessed May27, 2021.
[14] K. Guo, W. Li, K. Zhong, Z. Zhu, S. Zeng, S. Han, Y. Xie, P. Debacker, M. Verhelst, and Y. Wang. 2021. Neural Network Accelerator Comparison. https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator/.
[15] Kaiyuan Guo, Lingzhi Sui, Jiantao Qiu, Jincheng Yu, Junbin Wang, Song Yao, SongHan, Yu Wang, and Huazhong Yang. 2017. Angel-Eye: A complete design flowfor mapping CNN onto embedded FPGA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 37, 1 (2017), 35–47.
[16] Peng Guo, Hong Ma, Ruizhi Chen, Pin Li, Shaolin Xie, and Donglin Wang. 2018.Fbna: A fully binarized neural network accelerator. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 51–513.
[17] Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)* (2016).
[18] Zhezhi He and Deliang Fan. 2019. Simultaneously optimizing weight and quan- tizer of ternary neural network using truncated gaussian approximation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 11438–11446.
[19] Intel. 2017. Intel Arria 10 Native Fixed Point DSP IP Core User Guide. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_nfp_dsp.pdf Last accessed Sept 11, 2021.
[20] Li Jiao, Cheng Luo, Wei Cao, Xuegong Zhou, and Lingli Wang. 2017. Accelerating low bit-width convolutional neural networks with embedded FPGA. In *2017 27thInternational Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 1–4.
[21] Patrick Judd, Jorge Albericio, Tayler Hetherington, Tor M. Aamodt, and Andreas Moshovos. 2016. Stripes: Bit-serial deep neural network computing. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE,1–12.
[22] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. 2019. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4350–4359.

[23] Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. 2018. Extremely low bit neural network: Squeeze the last bit out with admm. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*.
[24] Fengfu Li, Bo Zhang, and Bin Liu. 2016. Ternary weight networks. *arXiv preprint arXiv:1605.04711* (2016).
[25] Xiaofan Lin, Cong Zhao, and Wei Pan. 2017. Towards accurate binary convo- lutional neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*. 345–353.
[26] Qian Lou, Feng Guo, Minje Kim, Lantao Liu, and Lei Jiang. 2019. AutoQ: Auto- mated Kernel-Wise Neural Network Quantization. In *International Conference on Learning Representations (ICLR)*.
[27] Alec Lu, Zhenman Fang, Weihua Liu, and Lesley Shannon. 2021. Demystifying the memory system of modern datacenter FPGAs for software programmers through microbenchmarking. In *The 2021 ACM/SIGDA International Symposiumon Field-Programmable Gate Arrays (FPGA)*. 105–115.
[28] Cheng Luo, Wei Cao, Lingli Wang, and Philip HW Leong. 2019. Rna: An accurate residual network accelerator for quantized and reconstructed deep neural net- works. *IEICE Transactions on Information and Systems* 102, 5 (2019), 1037–1045.
[29] Daisuke Miyashita, Edward H Lee, and Boris Murmann. 2016. Convolu- tional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025* (2016).
[30] Hiroki Nakahara, Tomoya Fujii, and Shimpei Sato. 2017. A fully connected layer elimination for a binarized convolutional neural network on an FPGA. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*.IEEE, 1–4.
[31] Hiroki Nakahara, Haruyoshi Yonekawa, Tsutomu Sasao, Hisashi Iwamoto, and Masato Motomura. 2016. A memory-based realization of a binarized deep convo-lutional neural network. In *2016 International Conference on Field-Programmable Technology (FPT)*. IEEE, 277–280.
[32] Dong Nguyen, Daewoo Kim, and Jongeun Lee. 2017. Double MAC: Doubling the performance of convolutional neural networks on modern FPGAs. In *Design,Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 890–893.
[33] Nvidia. 2021. Nvidia Deep Learning Examples. https://github.com/NVIDIA/ DeepLearningExamples Last accessed Sept 12, 2021.
[34] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. 2018. Value-aware quantizationfor training and inference of neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 580–595.
[35] Atul Rahman, Sangyun Oh, Jongeun Lee, and Kiyoung Choi. 2017. Design spaceexploration of FPGA accelerators for convolutional neural networks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 1147– 1152.
[36] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016.Xnor-net: Imagenet classification using binary convolutional neural networks.In *European conference on computer vision (ECCV)*. Springer, 525–542.
[37] Ao Ren, Tianyun Zhang, Shaokai Ye, Jiayu Li, Wenyao Xu, Xuehai Qian, Xue Lin, and Yanzhi Wang. 2019. Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 925–938.
[38] Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Vikas Chandra, and Hadi Esmaeilzadeh. 2018. Bit fusion: Bit-level dynamically compos- able architecture for accelerating deep neural networks. In *Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE Press,764–775.
[39] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT.. In *Thirty-Second AAAI Conference on Arti- ficial Intelligence (AAAI)*. 8815–8821.
[40] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. 2020. Mixed Precision DNNs: All you need is a good parametrization. *International Conference on Learning Representations (ICLR)* (2020).
[41] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. 2017. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA Inter- national Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 65–74.
[42] Mário Véstias, Rui Policarpo Duarte, José T de Sousa, and Horácio Neto. 2017. Parallel dot-products for deep learning on FPGA. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 1–4.
[43] Junsong Wang, Qiuwen Lou, Xiaofan Zhang, Chao Zhu, Yonghua Lin, and Dem-ing Chen. 2018. 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)*. IEEE, 163–1636.
[44] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. 2019. HAQ: Hardware- Aware Automated Quantization with Mixed Precision. *International Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 8604–8612.
[45] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. 2018. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090* (2018).
[46] Xilinx. 2017. Deep Learning with INT8 Optimization on Xilinx De- vices. https://www.xilinx.com/support/documentation/white_papers/wp486- deep-learning-int8.pdf Last accessed Sept 12, 2021.
[47] Xilinx. 2020. Convolutional Neural Network with INT4 Optimization on Xilinx Devices. https://www.xilinx.com/support/documentation/white_papers/wp521- 4bit-optimization.pdf Last accessed Sept 12, 2021.
[48] Yifan Yang, Qijing Huang, Bichen Wu, Tianjun Zhang, Liang Ma, Giulio Gam-bardella, Michaela Blott, Luciano Lavagno, Kees Vissers, John Wawrzynek, and Kurt Keutzer. 2019. Synetgy: Algorithm-hardware co-design for convnet acceler- ators on embedded fpgas. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 23–32.
[49] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael Mahoney. 2019. PyHessian:Neural networks through the lens of the Hessian. *arXiv preprint arXiv:1912.07145* (2019).
[50] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. 2015. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays (FPGA)*. 161–170.
[51] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. 2018. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*. 365–382.
[52] Xiaofan Zhang, Junsong Wang, Chao Zhu, Yonghua Lin, Jinjun Xiong, Wen-mei Hwu, and Deming Chen. 2018. DNNBuilder: an automated tool for building high-performance DNN hardware accelerators for FPGAs. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.
[53] Ritchie Zhao, Weinan Song, Wentao Zhang, Tianwei Xing, Jeng-Hau Lin, Mani Srivastava, Rajesh Gupta, and Zhiru Zhang. 2017. Accelerating binarized con- volutional neural networks with software-programmable fpgas. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 15–24.
[54] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. 2017. Incre- mental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044* (2017).
[55] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160* (2016).
[56] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. 2017. Trained ternary quantization. In *International Conference on Learning Representations (ICLR)*.