Auto-ViT-Acc: An FPGA-Aware Automatic Acceleration Framework for Vision Transformer with Mixed-Scheme Quantization

Zhengang Li†[1], Mengshu Sun†[1], Alec Lu[2], Haoyu Ma[3], Geng Yuan[1], Yanyue Xie[1], Hao Tang[4], Yanyu Li[1], Miriam Leeser[1], Zhangyang Wang[5], Xue Lin[1], Zhenman Fang[2]
[1]Northeastern University, [2]Simon Fraser University, [3]University of California, Irvine,
[4]ETH Zurich, [5]The University of Texas at Austin
E-mail: [1]{li.zhen, sun.meng, yuan.geng, xie.yany, li.yanyu, xue.lin}@northeastern.edu, [1]mel@coe.neu.edu,
[2]{alec_lu, zhenman}@sfu.ca, [3]haoyum3@uci.edu, [4]hao.tang@vision.ee.ethz.ch, [5]atlaswang@utexas.edu

Abstract—**Vision transformers (ViTs) are emerging with significantly improved accuracy in computer vision tasks. However, their complex architecture and enormous computation/storage demand impose urgent needs for new hardware accelerator design methodology. This work proposes an FPGA-aware automatic ViT acceleration framework based on the proposed mixed-scheme quantization. To the best of our knowledge, this is the first FPGA-based ViT acceleration framework exploring model quantization. Compared with state-of-the-art ViT quantization work (algorithmic approach only without hardware acceleration), our quantization achieves 0.47% to 1.36% higher Top-1 accuracy under the same bit-width. Compared with the 32-bit floating-point baseline FPGA accelerator, our accelerator achieves around 5.6× improvement on the frame rate (i.e., 56.8 FPS vs. 10.0 FPS) with 0.71% accuracy drop on ImageNet dataset for DeiT-base.**

I. Introduction

Transformer, an attention-based encoder-decoder architecture [1], has revolutionized the field of natural language processing (NLP) in the past five years. Inspired by NLP successes, researchers began to adopt transformer-like architecture to computer vision tasks i.e., vision transformers (ViTs), achieving better performance compared with state-of-the-art convolutional neural networks (CNNs) [2]–[4]. However, the complex model architecture and enormous computation and storage of ViT make it a challenging task for their deployment into resource constrained edge devices.

Model quantization, as a crucial technique for DNN inference acceleration on edge devices, has been broadly explored for CNNs [5]–[10] with different bit-widths and also different quantization schemes, e.g., fixed-point and power-of-two (PoT). These two types of schemes were mixed in [11] for FPGA-based implementations to fully utilize the hardware computation resources. As for quantization of transformer models, few efforts [12] have been devoted to ViTs, while majority of work [13]–[15] was still on transformers for NLP with purely algorithmic approaches. There are two open problems for ViT quantization: 1. Do existing quantization schemes for CNNs work well for ViTs? 2. How to systematically determine the bit-width and mixing ratio in mixed-scheme quantization for better accuracy and throughput performance for ViTs?

In this paper, we first explore the feasibility of the well-studied CNN quantization schemes—including fixed-point, PoT, and their mix—on ViT and make the following observations. First, fixed quantization possesses superior accuracy performance, and its computation can be efficiently implemented with the DSP resources on FPGA. Second, the PoT scheme offers a highly efficient quantization with still acceptable accuracy, where multiplications can be replaced by simple shift operations, and thus suitable for implementation with LUT resource on FPGA. Finally, combining fixed-point and PoT has the potential to further improve FPGA resource utilization for inference acceleration while maintaining accuracy.

Based on the above, we develop an FPGA-aware automatic ViT acceleration (Auto-ViT-Acc) framework for our mixed-scheme ViT quantization algorithm. It contains an "FPGA Resource Utilization Modeling" module to give performance analysis and estimate the frame rate (FPS) for the FPGA ViT accelerator under a certain setting of model bit-widths, which will be reduced until the target FPS is achieved. In this way, the bit-width and the ratio of fixed-point quantized rows over PoT quantized rows can be optimized and used as inputs to guide the quantization algorithm. This framework also designs a novel FPGA compute engine for ViT multi-head attention with optimizations for accelerators. We automate the entire workflow based on a target FPS, to obtain a quantized model and an FPGA accelerator. The contributions of our work are summarized as follows:

- **An FPGA-aware mixed-scheme ViT quantization algorithm that can fully leverage heterogeneous FPGA resources while maximally retaining accuracy.**
- **An automated ViT acceleration framework with FPGA resource utilization modeling to automatically find the best combination of quantization bit-widths and the scheme mixing ratio for a target FPS.**
- **A novel FPGA computing engine for ViT multi-head attention and related accelerator optimizations.**
- **To the best of our knowledge, Auto-ViT-Acc is the first for ViT acceleration on FPGAs exploring model quantization with significant performance improvements.**

II. Related Work

A. Vision Transformer

The ViT architecture was first proposed in [2], which adopts the self-attention mechanism [1] for image classification tasks.

†These authors contributed equally.

Different from CNNs, ViT interprets an image as a sequence of patches and then inputs to standard transformer encoders as used in NLP. However, it requires pre-training with complex and massive datasets such as ImageNet-21k and JFT-300M. To address this, DeiT [3] and T2T-ViT [4] were proposed to reduce dependency on massive pre-training and achieve better accuracy than ResNets [16] of comparable size on ImageNet.

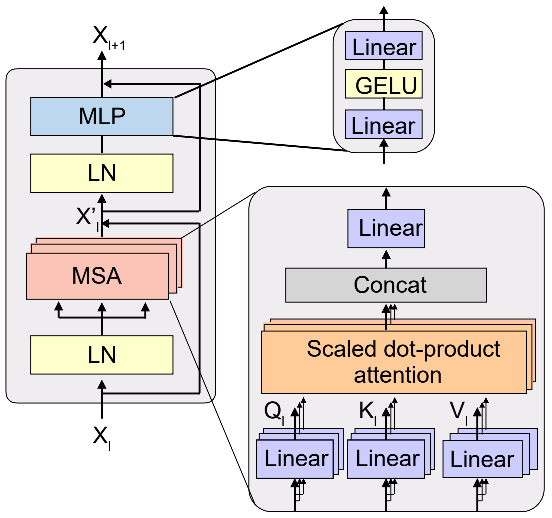

Fig. 1. Transformer encoder block structure.

In ViT, the main model architecture is transformer encoder blocks with multi-headed self-attention (MSA) and multi-layer perceptron (MLP) blocks as shown in Fig. 1. The layernorm (LN) is applied prior to MSA and MLP. The encoder block operations are:

$$\begin{aligned} \mathbf{X'}_l &= \mathrm{MSA}(\mathrm{LN}(\mathbf{X}_l)) + \mathbf{X}_l, \\ \mathbf{X}_{l+1} &= \mathrm{MLP}(\mathrm{LN}(\mathbf{X'}_l)) + \mathbf{X'}_l, \end{aligned} \tag{1}$$

where $\mathbf{X}_l$ denotes the input sequence of the l-th encoder block.

These modules involve large matrix multiplications incurring the most computational cost. Therefore, we quantize all linear layers involved in matrix multiplication, but not layer normalization, due to their low computational cost and potential effects on accuracy.

B. DNN Model Quantization

1) Quantization Schemes: Model quantization has been intensively explored for deep neural networks (DNNs) such as CNNs and recurrent neural networks (RNNs).

There are schemes using uniform quantization intervals including binary [5], [6] ternary [17], and low-bit-width fixed-point [7], [8]. Although binary and ternary quantization significantly reduce operations and simplify hardware implementation, it introduces large accuracy loss. The fixed-point quantization scheme, on the other hand, applies modest and flexible quantization rates to preserve accuracy close to that of 32-bit floating-point models. For instance, 4-bit fixed-point introduces zero or negligible accuracy loss. Fixed-point quantization scheme was implemented with different methods and algorithms, such as DoReFa-Net [7] and PACT [8].

There are also schemes using non-uniform quantization intervals such as power-of-two (PoT) [9] and additive PoT [10], by which multiplications can be replaced with bit shifting operations. Furthermore, PoT presents higher precision around the mean, and therefore better fits the Gaussian distribution of DNN weights [18]. But it exhibits rigid resolution issue that results in moderate accuracy loss, which cannot be mitigated even with higher bit-width. To overcome it, additive PoT was proposed by using a sum of multiple PoT numbers.

2) Transformer Quantization: Quantization has also been applied to transformers, in particular, bidirectional encoder representations from transformers (BERTs) [1]. Specifically, [13] finetuned BERT through 8-bit quantization-aware training. The later TernaryBERT [14] proposed to use an approximation-based and loss-aware ternarization for BERT, and distillation to further reduce accuracy drop caused by lower capacity. BinaryBERT [15] suggested that it is difficult to train a binary BERT directly due to its complex loss landscape and proposed a ternary weight splitting strategy to derive binary BERT with performance as the ternary one. All the aforementioned work targeted BERT in NLP tasks, not covering ViT in computer vision tasks.

A recent work [12] evaluated the post-training quantization on ViT and achieved comparable accuracy as the full-precision version. However, they only used a low quantization rate i.e., 4×, which is equivalent to 8-bit quantization precision. Further, it is a pure algorithmic method and not suitable for acceleration on hardware like FPGAs.

C. Transformer Accelerators on FPGAs

Recently, weight pruning approaches have also been applied for transformer acceleration on FPGAs. The study in [19] leveraged block-circulant matrix-based weight representation and FFT/IFFT-based processing elements for matrix-vector multiplication for fully-connected (FC) layers. Block-based weight pruning was applied in [20] to accelerate transformers on FPGAs. [21] proposed a structural pruning method with memory footprint awareness to compress weights to similar sizes. This method effectively compresses the attention mechanism and achieves efficient deployment of data buffers and computing kernels on FPGAs. Differently, our work explores model quantization for ViT acceleration on FPGA and is orthogonal and complementary to pruning-based prior arts.

III. New Challenges and Novelty

ViTs leverage the attention mechanism [1] to fulfill various computer vision tasks. Compared to CNNs that operate on a fixed-size window with restricted spatial interactions, ViT allows data at all the positions in an image to interact through transformer encoder blocks and thus improving accuracy [22]. As mentioned in [3], ViTs can perform better than representative CNNs like ResNet [16] and ResNeXt [23]. For instance, DeiT-small with a comparable number of parameters and operations as ResNet-50 achieves higher accuracy than

ResNeXt-101, whose size is around $4\times$ as that of ResNet-50. DeiT-base with comparable size as ResNeXt-101 achieves much higher accuracy.

Although with significant accuracy improvement, there exist challenges in hardware acceleration of ViTs, especially on resource-limited edge devices. First, even the light-weight DeiT-small model is already a large model for edge devices. Furthermore, the complexity of multi-head self-attention brings in a new optimization dimension of hardware parallelism. (Detailed discussions are provided in Sec. V-B.) Therefore, model compression techniques including pruning and quantization become essential in ViT hardware acceleration. Unlike most prior arts mentioned in Sec. II-C, this paper focuses on model quantization for ViT hardware acceleration.

Existing work on ViT quantization [12] adopted the fixed-point quantization scheme with 8-bit precision. In this paper, we observe that leveraging PoT quantization, which allows multiplications to be replaced by simple shift operations, can achieve better inference performance on FPGAs with the LUT resources, with negligible accuracy loss. Moroever, when combining the fixed-point quantization that mainly consumes the DSP resources on FPGAs, there is more potential to fully utilize the FPGA resource for even better performance.

Besides various quantization schemes, layer-wise multi-precision quantization has been well investigated in [24]–[26] that assign precisions onto weights and activations of individual layers. However, as pointed out in [11], this type of quantization is incompatible with layer-by-layer inference execution on hardware accelerators since it introduces non-uniformality among layers. In contrast, this paper adopts the *mixed-scheme quantization within each layer*, with a mixture of fixed-point and PoT schemes. Different from [11] that focuses on CNN acceleration, we use PoT in replacement of their Sum-of-PoT for improved computation efficiency while avoiding compromising accuracy. Unlike [24]–[26] which deal with a large search space for precision assignment, we propose a practical mixed-scheme ViT quantization algorithm that closely coordinates with the FPGA-based accelerator design.

For mixed-scheme quantization, we need the co-design of the quantization algorithm and the FPGA accelerator. We propose a set of automated mechanism (Sec. V-A) with FPGA resource utilization modeling to automatically find the best combination of quantization bit-widths and mixed-scheme ratio for a targeted FPS. Furthermore, from the hardware design aspect, we have the following observations: First, to prevent extra hardware overhead on output shifting among two schemes i.e., fixed-point and PoT, we propose to align the outputs from two quantization schemes by deriving the relation between their precisions i.e., bit-widths. Explanations are in Sec. V-A. Second, we propose to use the same ratio of fixed-point to PoT for each head of the MSA module to fully exploit parallelism of FPGA.

IV. FPGA-Aware Mixed-Scheme ViT Quantization Algorithm

A. *Quantization Scheme and Precision*

We propose to use a mixture of fixed-point and PoT within each layer. Note that we apply quantization only to linear layers of ViT, which involve the most computation-intensive matrix multiplications. We do not quantize for softmax and layer normalization, due to their low computational cost. Fixed-point quantization scheme has superior accuracy performance, and its computation can be implemented efficiently with DSP resources on FPGA. PoT is a highly efficient quantization scheme with still acceptable accuracy, where multiplications can be replaced by bit shifting operations, and thus suitable for implementation with LUT resource on FPGA. Combining fixed-point and PoT can increase FPGA resource utilization to speed up inference, at the same time, retain accuracy.

We use $\prod_{(b,\alpha)}^{\text{Fixed}}$ and $\prod_{(b,\alpha)}^{\text{PoT}}$ to represent the fixed-point and PoT quantizers, respectively, where b denotes the bit-width and α denotes scaling factor. Detailed quantizer functions can be found in [11]. In general, a quantizer function maps a floating-point value into a fixed-point or PoT quantized value, equal to multiplication of the scaling factor with a quantization level represented by a b-bit number. For both quantization schemes, b-bit number representation corresponds to $2^b - 1$ quantization levels (with 1-bit for sign). As for the selection of precision or bit-width, to avoid the large search space of scheme and precision assignment and to preserve hardware uniformity among layers, we specify the precision candidates as: b-bit for fixed-point quantized weights, b'-bit for PoT quantized weights, and b-bit for activations.[1]

B. *Proposed ViT Quantization Algorithm*

As shown in Algorithm 1, our proposed FPGA-aware mixed-scheme ViT quantization algorithm performs quantization training with given bit-widths i.e., b and b', and the ratio of PoT quantized rows i.e., k_{pot} in each layer (the rest rows are fixed-point quantized). We use the same ratio k_{pot} among different heads of the MSA module to fully exploit the parallelism of FPGA. b, b', and k_{pot} are determined from our Auto-ViT-Acc framework. The quantization scheme is assigned down to the row level of a weight matrix based on the weight distribution. In general, if a row has a smaller variance, the PoT scheme is assigned; and otherwise, the fixed-point scheme is assigned.

V. Proposed Auto-ViT-Acc Framework

This section first gives an overview of Auto-ViT-Acc, and then discusses the optimization techniques in the ViT computation engine (Sec. V-B and V-C), and finally provides FPGA resource modeling to determine b, b', and k_{pot} for target frame rate (FPS) (Sec. V-D).

[1]Even for PoT scheme, only weights are PoT quantized and corresponding activations are still fixed-point quantized in order to replace multiplication with bit shifting.

Algorithm 1: FPGA-aware mixed-scheme ViT quantization.

```
input : 32-bit floating-point pre-trained ViT model $\mathcal{M}$ with
        weights $\mathbf{W}$; bit-width for fixed-point $b$; bit-width for
        PoT $b'$; ratio of PoT quantized rows in each layer
        $k_{\mathrm{PoT}}$;
output: Quantized model $\hat{\mathcal{M}}$.
1  foreach batch do
       // forward propagation
2      foreach layer $i$ in $\mathcal{M}$ do
           // calculate variance for each row
3          foreach row $\mathbf{W}_{ij}$ in layer $i$ do
4              $\mathrm{var}_{ij} \leftarrow variance(\mathbf{W}_{ij})$ ;
           // assign weight quantization scheme
              for rows
5          foreach row $\mathbf{W}_{ij}$ in layer $i$ do
6              if $var_{ij}$ belongs to the bottom $k_{\mathrm{PoT}}$ group then
7                  $\hat{\mathbf{W}}_{ij} \leftarrow \prod_{b'}^{\mathrm{PoT}}(\mathbf{W}_{ij})$;
8              else
9                  $\hat{\mathbf{W}}_{ij} \leftarrow \prod_{b}^{\mathrm{Fixed}}(\mathbf{W}_{ij})$;
10         $\mathbf{A}_i \leftarrow \hat{\mathbf{W}}_i \cdot \hat{\mathbf{A}}_{i-1}$;
           // quantize activations
11         $\hat{\mathbf{A}}_i \leftarrow \prod_{b}^{\mathrm{Fixed}}(\mathbf{A}_i)$;
       // backward propagation
12     foreach layer $i$ (reverse order) do
13         $\frac{\partial Loss}{\partial \mathbf{W}_i} \leftarrow \frac{\partial Loss}{\hat{\mathbf{W}}_i}$;
14         $\frac{\partial Loss}{\partial input_i} \leftarrow \frac{\partial Loss}{\hat{\mathbf{A}}_{i-1}}$;
15 Return $\hat{\mathcal{M}} \leftarrow \mathcal{M}\{\hat{\mathbf{W}}\}$.
```

A. Overview and Design Space Exploration

Fig. 2 provides the workflow of our Auto-ViT-Acc framework for automatic generations of ViT accelerators. We start from "FPGA Resource Utilization Modeling" module to give performance analysis and estimate the frame rate (FPS) of FPGA ViT accelerator with given bit-widths for the Fixed and PoT schemes i.e., b and b'. We reduce the bit-widths until fulfilling the target FPS. The details of resource modeling and performance analysis are discussed in Section V-D, which also derive the desired ratio for PoT quantized rows k_{pot}. Then our proposed mixed-scheme ViT quantization algorithm uses b, b', and k_{pot} to derive quantized ViT model, which will be implemented on FPGA by going through "C++ Description for Accelerator", "Xilinx Vitis High-Level Synthesis (HLS)", and "Accelerator Bitstream".

About bit-widths, for each layer, we quantize some of the rows into Fixed with b-bit for weights and b-bit for the corresponding activations i.e., Fixed W[b]A[b], and quantize the rest rows into PoT W[b']A[b] i.e., b'-bit for weights and b-bit for corresponding activations. To prevent extra hardware overhead on output shifting among two schemes, we propose to align the outputs from two schemes by setting $2^{(b'-1)} \leq b$, i.e., if b-bit is used for Fixed, then $b' = \lfloor \log_2 b \rfloor + 1$ is used for PoT. For example, the 4-bit Fixed scheme matches the 3-bit PoT scheme, and 8-bit Fixed scheme matches the 4-bit PoT scheme. This is because the product of the Fixed W[b]A[b] multiplication has a bit-width of $2 \cdot b$, and the same output bit-width is required in the PoT W[b']A[b] multiplication realized by left shifting the input activation by b' bits.

B. Compute Engine for Multi-Head Attention

The notations used in ViT accelerators are listed in Table I. The accelerator designs are based on loop tiling shown in Fig. 3, where the input, weight, and output data for each ViT layer are split into tiles for FPGA resource-saving. With pipelining and unrolling of loops, the compute engine can manage $(T_m^{\mathrm{Fix}} + T_m^{\mathrm{PoT}}) \cdot P_h \cdot T_n$ multiply-accumulate (MAC) operations in parallel.

TABLE I. Notations for ViT Accelerator

Notation	Description
k_{PoT}	The ratio of PoT quantized rows for weights
M (N)	Number of output (input) channels
F	Number of token sequences
T_n	Tiling size for data in input channel dimension in each head
T_m^{Fix} (T_m^{PoT})	Tiling size for Fixed (PoT) data in output channel dimension
N_h	Total number of heads
P_h	Number of heads for computation in parallel
D (D')	Number of data packed as one for activations and Fixed weights (PoT weights)
A_{in} (A_{out}, A_{wgt})	Number of AXI ports used for data transfer of input (output, weight) tile
L_{in} (L_{wgt}, L_{out}, L_{cmpt})	Number of clock cycles for input transfer (weight transfer, output transfer, computation) for a group of tiles
B_{in} (B_{out}, B_{wgt})	Number of BRAMs used by input (output, weight) tile
$C_{\mathrm{dsp}}^{\mathrm{Fix}}$	DSP cost for each MAC operation with Fixed weight
$C_{\mathrm{lut}}^{\mathrm{Fix}}$ ($C_{\mathrm{lut}}^{\mathrm{PoT}}$)	LUT cost for each MAC operation with Fixed (PoT) weight

ViT computations mainly comprise matrix multiplications in multi-layer perceptron (MLP) modules and multi-head self-attention (MSA) modules. Each MSA can be seen as multiple parallel matrix multiplications, and therefore the accelerator is designed to process P_h attention heads in parallel, by splitting the N input channels into N_h groups. This input channel splitting is also done for fully connected (FC) layers, each containing only one matrix multiplication for compatibility, and the results need to be accumulated from all the input channels in all the heads.

C. Optimizations in ViT Accelerator

1) Processing of Other Computations: In addition to matrix multiplications, ViTs contain convolution, scaling, softmax, activation, normalization (LN), and skip-connection addition operations. The first layer of a ViT is a convolutional layer that can be converted to an FC layer because its kernel size and stride are the same as the patch size, meaning that the input data are used only once when a weight kernel slides across the input feature map. The scaling, softmax, and GELU activation operations are performed on the host CPU of the FPGA, which introduces a small latency overhead for embedded FPGAs compared with matrix multiplications.

As illustrated in Fig. 1, LN is applied at the beginning of each MSA or MLP module. The LN inputs require to be stored for later additions due to the identity skip-connection linking

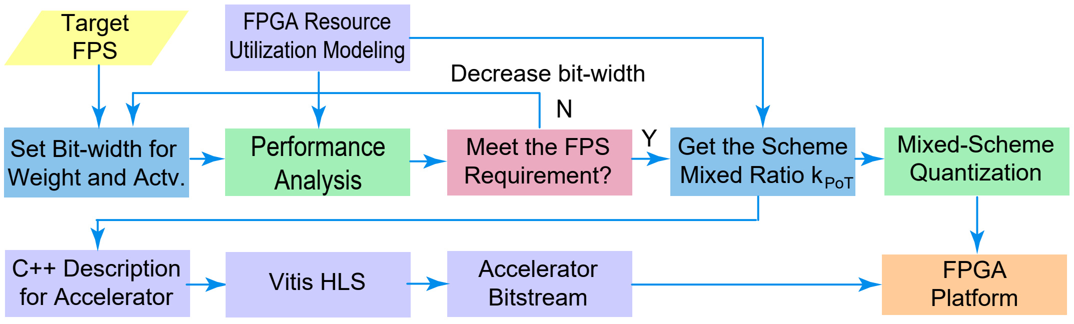
Target FPS
FPGA Resource Utilization Modeling
Decrease bit-width
N
Y
Set Bit-width for Weight and Actv.
Performance Analysis
Meet the FPS Requirement?
Get the Scheme Mixed Ratio k_{PoT}
Mixed-Scheme Quantization
C++ Description for Accelerator
Vitis HLS
Accelerator Bitstream
FPGA Platform

Fig. 2. Overview of Auto-ViT-Acc framework.

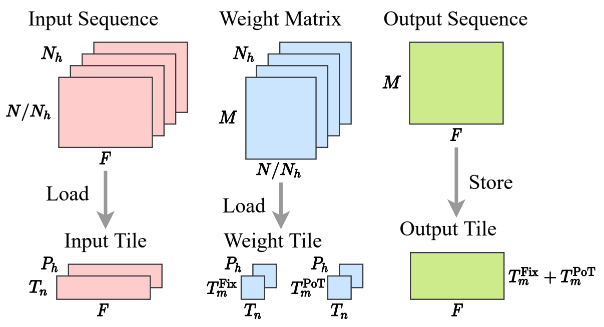

Fig. 3. Tiling in ViT computations.

the input activations of each LN and the output activations of the subsequent module. Considering that keeping LN operations unquantized will not incur much computation overhead but help maintain the model accuracy, the LN parameters and inputs are represented with 16-bit precision on hardware. Two data transfer ports are needed respectively for unquantized LN input and quantized LN outputs (which are also inputs of the next FC layer) to minimize the input loading time for subsequent FC computations.

2) DSP Packing: To fully exploit the potential of DSP resources on FPGAs, we pack multiple low-bit multiplications within each DSP following [27], [28]. Each DSP (DSP48E2) on the ZCU102 board could support the computation of $P = (A + D) \times B$, where both A and D are 27-bit operands, B is an 18-bit operand, and P is the 45-bit output. One DSP can accommodate two 8×8-bit multiplications by holding one weight in A and two input activation values in B, or four 4×4-bit multiplications by holding one weight in A, another weight in D, and two inputs in B. It is worth noting that the number of 4×4-bit multiplications handled by each DSP in this design is higher than that in [11], resulting in higher resource utilization efficiency and throughput.

D. ViT Accelerator Design with Resource Modeling and Performance Analysis

An FPGA board contains primarily two types of computation resources, namely DSPs and LUTs. Multiplications with fixed-point weights are computed with DSPs, and those with PoT weights can be replaced by shifting operations that are computed with LUTs. The DSP and LUT cost requires to be precisely estimated to find the best ratio between the numbers of fixed-point and PoT weights and thus maximizing the throughput on FPGAs.

The parameters to be determined for the accelerator include T_m^{Fix} (T_m^{PoT}), T_n, D (D'), and P_h. On a specific FPGA board, the maximum achievable FPS, denoted by $\mathrm{FPS}_{\mathrm{max}}$, can be estimated according to our analysis of FPGA resource utilization and performance. Given the target FPS, denoted by $\mathrm{FPS}_{\mathrm{tgt}}$, we first find the precision and scheme combination satisfying $\mathrm{FPS}_{\mathrm{max}} \geq \mathrm{FPS}_{\mathrm{tgt}}$. Under this precision, we fix P_h, T_n, D (D'), and T_m^{Fix}, and adjust T_m^{PoT} to meet the target FPS and obtain the best model accuracy. In detail, P_h is set to a value that can divide N_h exactly for full exploitation of computation resources, i.e., $P_h = 3$ for $N_h = 6$, and $P_h = 4$ for $N_h = 8$ or $N_h = 12$. D is decided based on the FPGA AXI port size and the quantization bit-width of Fixed weights, and is the same for activations in both Fixed and PoT computations as well as weights in Fixed computations. The bit-width of PoT weights is lower, corresponding to D'. T_n is set to the same value as D. The computation parallelism along the output channel dimension is decided by the sum $T_m^{\mathrm{Fix}} + T_m^{\mathrm{PoT}}$, and the model accuracy in quantization is affected by the ratio $k_{\mathrm{PoT}} = \frac{T_m^{\mathrm{PoT}}}{T_m^{\mathrm{PoT}} + T_m^{\mathrm{Fix}}}$, i.e., lower k_{PoT} will result in higher model accuracy. We therefore reduce T_m^{PoT} to make the actual FPS equal to $\mathrm{FPS}_{\mathrm{tgt}}$ if $\mathrm{FPS}_{\mathrm{max}} > \mathrm{FPS}_{\mathrm{tgt}}$ under this precision, and the actual k_{PoT} ratio will guide the quantization process and the hardware implementations with all these parameters.

1) FPGA Resource Utilization Modeling: In contrast to DSP usage, LUT consumption for shifting operations and also for logic is difficult to estimate, and therefore we build a resource utilization model through several simple experiments to model the LUT cost as a linear function of computation parallelism (the number of parallel operations in each clock cycle). For Fixed W[b]A[b] + PoT W[b']A[b] quantization, the LUT cost is analyzed for both W[b]A[b] Fixed multiplications executed on DSPs (denoted by $C_{\mathrm{lut}}^{\mathrm{Fix}}$), and W[$b'$]A[$b$] PoT multiplications executed on LUTs (denoted by $C_{\mathrm{lut}}^{\mathrm{PoT}}$). The LUT cost can then be obtained from the slopes of the fitted lines. It is worth noting that employing DSPs for multiplications consumes LUTs as well, resulting from data packing and accumulation operations, etc.

2) Inference Latency Analysis: The actual FPS is the reciprocal of the inference latency, which is analyzed below, with main variables explained in Table I. For one layer i in ViTs, the numbers of clock cycles needed for input tile loading, weight tile loading, and output tile storage are calculated as

$$\begin{aligned} L_{\text{in}} &= P_h \cdot \left\lceil \frac{T_n}{D} \right\rceil \cdot \left\lceil \frac{F}{A_{\text{in}}} \right\rceil, \\ L_{\text{wgt}} &= P_h \cdot \left(\left\lceil \frac{T_n}{D} \right\rceil \cdot \left\lceil \frac{T_m^{\text{Fix}}}{A_{\text{wgt}}} \right\rceil + \left\lceil \frac{T_n}{D'} \right\rceil \cdot \left\lceil \frac{T_m^{\text{PoT}}}{A_{\text{wgt}}} \right\rceil \right), \\ L_{\text{out}} &= (1+\gamma) \cdot \left\lceil \frac{T_m^{\text{Fix}} + T_m^{\text{PoT}}}{D} \right\rceil \cdot \left\lceil \frac{F}{A_{\text{out}}} \right\rceil, \end{aligned} \tag{2}$$

where γ is $N_h - 1$ if the current layer is a multi-head attention layer else 0. Additionally, the clock cycle number of computations for one group of tiles is

$$L_{\text{cmpt}} = \left\lceil \frac{F}{2} \right\rceil \cdot \left\lceil \frac{N_h}{P_h} \right\rceil, \tag{3}$$

as two input values are fetched in each clock cycle for DSP packing. The data loading and computation for the tiles are conducted simultaneously with the double buffering technique to overlap the data transfer with computations. The clock cycle number of this process is $L_1 = \max\{L_{\text{in}}, L_{\text{wgt}}, L_{\text{cmpt}}\}$. And to obtain the accumulation of output results, this process is performed multiple times. The clock cycle number for calculating the whole output tile is $L_2 = \max\left\{L_1 \cdot \left\lceil \frac{N}{P_h \cdot T_n} \right\rceil + L_{\text{cmpt}}, L_{\text{out}}\right\}$. The total number of clock cycles for a ViT layer i is therefore described by

$$L_{\text{tot}}^i = \left\lceil \frac{M}{T_m^{\text{Fix}} + T_m^{\text{PoT}}} \right\rceil \cdot L_2 + L_{\text{out}}. \tag{4}$$

Under a working frequency f, the FPS is calculated as $\frac{f}{\sum_i L_{\text{tot}}^i}$.

With double buffering, the 18k-bit BRAM usage of the input, weight, and output tiles are given by

$$\begin{aligned} B_{\text{in}} &= 2 \cdot P_h \cdot \left\lceil \frac{T_n}{D} \right\rceil \cdot \left\lceil \frac{b \cdot F \cdot D}{18\text{k}} \right\rceil, \\ B_{\text{wgt}} &= 2 \cdot P_h \cdot \left(\left\lceil \frac{T_n}{D} \right\rceil \cdot \left\lceil \frac{b \cdot T_m^{\text{Fix}} \cdot D}{18\text{k}} \right\rceil + \left\lceil \frac{T_n}{D'} \right\rceil \cdot \left\lceil \frac{b' \cdot T_m^{\text{PoT}} \cdot D'}{18\text{k}} \right\rceil \right), \\ B_{\text{out}} &= 2 \cdot N_h \cdot \left\lceil \frac{T_m^{\text{Fix}} + T_m^{\text{PoT}}}{D} \right\rceil \cdot \left\lceil \frac{b \cdot F \cdot D}{18\text{k}} \right\rceil. \end{aligned} \tag{5}$$

The DSP and LUT consumption is proportional to the total MAC computation parallelism. Specifically, $C_{\text{dsp}}^{\text{Fix}} = 0.25$ for each multiplication with W4A4 or smaller precision, and $C_{\text{dsp}}^{\text{Fix}} = 0.5$ for each multiplication with W5A5 to W8A8 precision. In summary, the FPS and k_{PoT} for the ViT are decided satisfying

$$\begin{aligned} B_{\text{in}} + B_{\text{wgt}} + B_{\text{out}} &\le S_{\text{bram}}, \\ C_{\text{dsp}}^{\text{Fix}} \cdot T_m^{\text{Fix}} \cdot P_h \cdot T_n &\le S_{\text{dsp}} \cdot r_{\text{dsp}}, \\ \left(C_{\text{lut}}^{\text{Fix}} \cdot T_m^{\text{Fix}} + C_{\text{lut}}^{\text{PoT}} \cdot T_m^{\text{PoT}}\right) \cdot P_h \cdot T_n &\le S_{\text{lut}} \cdot r_{\text{lut}}, \end{aligned} \tag{6}$$

where S_{bram}, S_{dsp}, S_{lut} are the available number of BRAMs, DSPs, and LUTs on FPGA, and r_{dsp} and r_{lut} are the maximum ratio of DSPs and LUTs to be utilized for MAC operations.

VI. Experiments

A. Experimental Setups

Our experiments include model quantization and hardware implementations for ViTs of different sizes, namely DeiT-small and DeiT-base, without the distillation tokens [3]. Our quantization training process takes 100 epochs with a batch size of 512, on top of the pre-training process with 300 epochs. The learning rate is set to 5×10^{-4} initially and decayed with a cosine annealing schedule. The AdamW [29] optimizer is used with the weight decay of 0.05. Training tricks to improve the accuracy include warmup training of 3 epochs and label smoothing with a factor of 0.1. The quantization adopts the same hyper-parameters for all models and is conducted on 4 NVIDIA Ampere A100 GPUs with CUDA 11.0 and PyTorch 1.7 frameworks on the Ubuntu operating system. The quantized models are then evaluated on the Xilinx ZCU102 FPGA platform consisting of 2520 DSPs and 274.1k LUTs. To maximize the computation efficiency without timing violation, the working frequency is set to 150 MHz for all the designs implemented through Xilinx Vitis and Vitis HLS 2020.2. We use the official DeiT model (W32A32) as our baseline. And the W32A32 data in baseline unquantized models are represented in 16-bit format when implemented on FPGA. This conversion incurs negligible accuracy degradation, which is common for FPGA implementations.

B. Experimental Results

Comparison of Different Quantization Schemes. The comparison results of different quantization schemes in terms of accuracy after quantization and performance with resource utilization are listed in Table II. All the activation are quantized with Fixed schemes, and the weights are quantized with the schemes as shown in the first column. It can be seen that the PoT quantization on ViTs obtains noticeable throughput improvement compared with the Fixed-point quantization at the same bit-width level with manageable accuracy loss, and our mixed-scheme quantization further achieves higher throughput and better model accuracy than the PoT quantization. With various FPS targets, we investigate the effectiveness of our mixed-scheme quantization by adjusting the bit-widths and scheme mixing ratio for different models. Specifically, we set the target FPS as 150 and 100 for DeiT-small, and 50 and 30 for DeiT-base.

For DeiT-small, it can be seen that a target FPS of 150 can be met using W4A4+W3A4 quantization precision with PoT ratio $k_{\text{PoT}} = 43\%$ and the Top-1 accuracy reaches 77.94%, outperforming the W8A8 model of PTQ [12] by 0.47% even with a lower bit-width. For the desired FPS of 100, the implementation using W8A8+W4A8 precision with PoT ratio of $k_{\text{PoT}} = 43\%$ can fulfill the requirement with 78.74% accuracy, which is 1.27% higher than that of PTQ. As for DeiT-base, the accuracy loss incurred by quantization is less than 1%, while 55 FPS with 81.14% accuracy can be achieved using W4A4+W3A4 precision with $k_{\text{PoT}} = 40\%$, and 33 FPS with 81.84% accuracy can be reached using W8A8+W4A8 precision with $k_{\text{PoT}} = 45\%$.

TABLE II. Accuracy and Hardware Results under Different Quantization Schemes for DeiT-small and DeiT-base on ImageNet Dataset

Quantization Weight Scheme	Bit-Width (Weight/Actv.)	Model Accuracy (%) Top-1	Top-5	Resource Utilization DSP	kLUT	Power (W)	Thrpt. (GOPS)	Frame Rate (FPS)	Energy Eff. (FPS/W)
DeiT-small									
Baseline	W32A32	79.85	94.97	1745 (69%)	130 (47%)	8.38	354.5	38.9	4.64
PTQ [12] (Fixed)	W8A8	77.47 (−2.38)	-	-	-	-	-	-	-
Fixed	W4A4	78.50 (−1.35)	94.41 (−0.56)	1933 (77%)	137 (50%)	10.44	1186.6	130.3	12.48
PoT	W3A4	77.24 (−2.61)	93.89 (−1.08)	13 (1%)	176 (64%)	6.55	1374.1	150.9	23.04
Mixed ($\text{FPS}_{\text{tgt}} = 150$)	W4A4+W3A4 ($k_{\text{PoT}} = 43\%$)	77.94 (−1.91)	94.07 (−0.90)	1549 (61%)	193 (70%)	10.34	1418.4	155.8	15.06
Fixed	W8A8	79.69 (−0.16)	94.89 (−0.08)	1936 (77%)	122 (44%)	8.46	711.2	78.1	9.23
PoT	W4A8	77.97 (−1.88)	94.06 (−0.91)	16 (1%)	175 (64%)	8.58	837.0	91.9	10.71
Mixed ($\text{FPS}_{\text{tgt}} = 100$)	W8A8+W4A8 ($k_{\text{PoT}} = 43\%$)	78.74 (−1.11)	94.50 (−0.47)	1552 (62%)	185 (67%)	9.63	907.8	99.7	10.35
DeiT-base									
Baseline	W32A32	81.85	95.59	1564 (62%)	120 (44%)	9.91	345.8	10.0	1.01
PTQ [12] (Fixed)	W8A8	80.48 (−1.37)	-	-	-	-	-	-	-
Fixed	W4A4	81.33 (−0.52)	95.63 (+0.06)	2064 (82%)	139 (51%)	11.27	1648.1	47.5	4.21
PoT	W3A4	80.87 (−0.98)	95.57 (−0.02)	19 (1%)	191 (70%)	8.11	1958.4	56.4	6.95
Mixed ($\text{FPS}_{\text{tgt}} = 50$)	W4A4+W3A4 ($k_{\text{PoT}} = 40\%$)	81.14 (−0.71)	95.60 (+0.01)	1555 (62%)	179 (65%)	11.03	1970.3	56.8	5.15
Fixed	W8A8	81.93 (+0.08)	95.90 (+0.31)	2066 (82%)	128 (47%)	9.40	899.6	25.9	2.76
PoT	W4A8	81.51 (−0.34)	95.73 (+0.14)	20 (1%)	192 (70%)	7.24	1080.5	31.1	4.30
Mixed ($\text{FPS}_{\text{tgt}} = 30$)	W8A8+W4A8 ($k_{\text{PoT}} = 45\%$)	81.84 (−0.01)	95.85 (+0.26)	1556 (62%)	186 (68%)	9.31	1181.5	34.0	3.66

Comparison with Baseline and Other Framework. Under the similar quantization bit-width, the Top-1 accuracy of our Fixed W8A8 + PoT W4A8 model is 1.36% higher than that in PTQ. Under a lower bit-width, our Fixed W4A4 + PoT W3A4 model still outperforms the W8A8 model of PTQ by 0.66%. Compared with the 32-bit baseline model, our quantized model achieves around 5.6× improvement on frame rate (i.e., 56.8 FPS vs. 10.0 FPS) with only 0.71% Top-1 accuracy drop.

TABLE III. Performance comparison between TX2 and ZCU102 (FPGA) on full-precision models.

Model	Hardware	Power (W)	Latency (ms)	FPS
DeiT-small	TX2	11.87	54	18.52
	ZCU102	8.38	26	38.90
DeiT-base	TX2	12.28	127	7.87
	ZCU102	9.91	100	10.00

Comparison with Embeded CPU/GPU. We also test DeiT-base and DeiT-small on Jetson TX2 with 4-core ARM CPU and NVIDIA Pascal GPU, and compared them with our FPGA (ZCU102) implementation. Since TX2 GPU does not support low-bit computation, we only present the performance of the full precision model as shown in Table III. Overall, compared to TX2 GPU, our FPGA implementation achieves about 2x and 1.3x speedup on DeiT-small and DeiT-base, respectively, with 2.37 W 3.49 W lower power consumption. Even without quantization, our FPGA implementation is still more efficient compared with TX2 with the similar compute capability level.

VII. Conclusion

In this paper, we propose an FPGA-aware automatic ViT acceleration (Auto-ViT-Acc) framework for our mixed-scheme ViT quantization algorithm. The bit-width and the ratio of fixed-point quantized rows over PoT quantized rows can be optimized and used as inputs to guide the quantization algorithm. This framework also designs a novel FPGA compute engine for ViT multi-head attention with optimizations for accelerators. We automate the entire workflow based on a target FPS, to obtain a quantized model and an FPGA accelerator. Compared with the 32-bit floating-point baseline FPGA accelerator, our accelerator achieves around 5.6× improvement on the frame rate with 0.71% accuracy drop on ImageNet dataset for DeiT-base. To the best of our knowledge, this is the first work for quantization-based ViT acceleration on FPGAs.

Acknowledgment

This work is partly supported by NSF CCF-1901378; NSERC Discovery Grant RGPIN-2019-04613, DGECR-2019-00120, Alliance Grant ALLRP-552042-2020; CFI John R. Evans Leaders Fund.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[3] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*, 2021, pp. 10 347–10 357.

[4] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 558–567.

[5] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[6] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, 2016, pp. 525–542.

[7] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv:1606.06160*, 2016.

[8] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv:1805.06085*, 2018.

[9] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with admm," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] Y. Li, X. Dong, and W. Wang, “Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks,” in *International Conference on Learning Representations*, 2020.

[11] S.-E. Chang, Y. Li, M. Sun, R. Shi, H. K.-H. So, X. Qian, Y. Wang, and X. Lin, “Mix and match: A novel fpga-centric deep neural network quantization framework,” in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 208–220.

[12] Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, and W. Gao, “Post-training quantization for vision transformer,” in *Advances in Neural Information Processing Systems*, 2021.

[13] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, “Q8bert: Quantized 8bit bert,” in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, 2019, pp. 36–39.

[14] W. Zhang, L. Hou, Y. Yin, L. Shang, X. Chen, X. Jiang, and Q. Liu, “Ternarybert: Distillation-aware ultra-low bit bert,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[15] H. Bai, W. Zhang, L. Hou, L. Shang, J. Jin, X. Jiang, Q. Liu, M. Lyu, and I. King, “Binarybert: Pushing the limit of bert quantization,” in *ACL/IJCNLP (1)*, 2021.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[17] Z. He and D. Fan, “Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 438–11 446.

[18] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, “Weight uncertainty in neural network,” in *International Conference on Machine Learning*, 2015, pp. 1613–1622.

[19] B. Li, S. Pandey, H. Fang, Y. Lyv, J. Li, J. Chen, M. Xie, L. Wan, H. Liu, and C. Ding, “Ftrans: energy-efficient acceleration of transformers using fpga,” in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2020, pp. 175–180.

[20] P. Qi, Y. Song, H. Peng, S. Huang, Q. Zhuge, and E. H.-M. Sha, “Accommodating transformer onto fpga: Coupling the balanced model compression and fpga-implementation optimization,” in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, 2021, pp. 163–168.

[21] X. Zhang, Y. Wu, P. Zhou, X. Tang, and J. Hu, “Algorithm-hardware co-design of attention mechanism on fpga devices,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–24, 2021.

[22] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, “Do vision transformers see like convolutional neural networks?” in *Advances in Neural Information Processing Systems*, 2021.

[23] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500.

[24] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, “Haq: Hardware-aware automated quantization with mixed precision,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8612–8620.

[25] S. Uhlich, L. Mauch, F. Cardinaux, K. Yoshiyama, J. A. Garcia, S. Tiedemann, T. Kemp, and A. Nakamura, “Mixed precision dnns: All you need is a good parametrization,” in *International Conference on Learning Representations*, 2020.

[26] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, “Hawq: Hessian aware quantization of neural networks with mixed-precision,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[27] Xilinx, “Deep learning with int8 optimization on xilinx devices,” 2017, last accessed Mar. 28, 2022. [Online]. Available: https://docs.xilinx.com/v/u/en-US/wp486-deep-learning-int8

[28] ——, “Convolutional neural network with int4 optimization on xilinx devices,” 2020, last accessed Mar. 28, 2022. [Online]. Available: https://docs.xilinx.com/v/u/en-US/wp521-4bit-optimization

[29] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *International Conference on Learning Representations*, 2019.