VAQF: Fully Automatic Software-Hardware Co-Design Framework for Low-Bit Vision Transformer

Mengshu Sun\textsuperscript{1}, Haoyu Ma\textsuperscript{2}, Guoliang Kang\textsuperscript{3}, Yifan Jiang\textsuperscript{3}, Tianlong Chen\textsuperscript{3}, Xiaolong Ma\textsuperscript{1}, Zhangyang Wang\textsuperscript{3}, Yanzhi Wang\textsuperscript{1}

\textsuperscript{1}Northeastern University \textsuperscript{2}University of California, Irvine \textsuperscript{3}University of Texas at Austin
\{sun.meng, ma.xiaol, yanz.wang\}@northeastern.edu haoyum3@uci.edu
kgl.prml@gmail.com \{yifanjiang97, tianlong.chen, atlaswang\}@utexas.edu

Abstract—The transformer architectures with attention mechanisms have obtained success in Nature Language Processing (NLP), and Vision Transformers (ViTs) have recently extended the application domains to various vision tasks. While achieving high performance, ViTs suffer from large model size and high computation complexity that hinders the deployment of them on edge devices. To achieve high throughput on hardware and preserve the model accuracy simultaneously, we propose VAQF, a framework that builds inference accelerators on FPGA platforms for quantized ViTs with binary weights and low-precision activations. Given the model structure and the desired frame rate, VAQF will automatically output the required quantization precision for activations as well as the optimized parameter settings of the accelerator that fulfill the hardware requirements. The implementations are developed with Vivado High-Level Synthesis (HLS) on the Xilinx ZCU102 FPGA board, and the evaluation results with the DeiT-base model indicate that a frame rate requirement of 24 frames per second (FPS) is satisfied with 8-bit activation quantization, and a target of 30 FPS is met with 6-bit activation quantization. To the best of our knowledge, this is the first time quantization has been incorporated into ViT acceleration on FPGAs with the help of a fully automatic framework to guide the quantization strategy on the software side and the accelerator implementations on the hardware side given the target frame rate. Very small compilation time cost is incurred compared with quantization training, and the generated accelerators show the capability of achieving real-time execution for state-of-the-art ViT models on FPGAs.

I. INTRODUCTION

The attention mechanisms, especially the transformer architectures \cite{38, 39}, have achieved remarkable progress in Nature Language Processing (NLP) in the past few years \cite{32, 3}. Recently, the Vision Transformer (ViT) structure \cite{11} firstly introduces the transformers into the image classification task and suggests that a convolution-free architecture can achieve state-of-the-art performance. Later on, transformers have been widely used in several vision tasks performance, such as detection \cite{4, 51}, segmentation \cite{50, 43}, and pose estimation \cite{20, 19}. However, the excellent performance improvement requires increasing the model size and computation complexity, and it is difficult to deploy these huge models into real-world applications like augmented reality and autonomous driving.

To address this issue, many efforts have been devoted to compressing the cumbersome transformer architectures into a lightweight counterpart, including knowledge distillation \cite{17, 85, 38, 39}, pruning \cite{28, 8, 7}, and quantization \cite{48, 11, 56}. Among all these compression techniques, quantization is a popular solution as it still preserves the original network architecture. Concretely, quantization aims to replace the 32-bit parameter with a low-bitwidth representation. In NLP, the recent Binary-BERT \cite{1} pushes BERT quantization to the limit by weight binarization, and introduces quantization on activations to bring additional energy savings. However, compared with the one-hot tokens in language, the input image patches in vision contain richer information, and it is unclear whether the binarization is still effective in ViTs.

In this paper, we first introduce the binarization into vision transformers. As fully binarization will lead to significant accuracy loss for ViTs, we adopt binary precision for weights and low precision for activations. Different from Binary-BERT, our method directly applies the methods in 1-bit convolutions \cite{34, 24} to achieve the binary weights. To support the inference of quantized ViT models on FPGAs, we propose a ViT Automatic Quantization Framework, namely VAQF, to generate ViT accelerators according to the real-time frame rate requirement. A compilation step is conducted to automatically determine the quantization precision for activations on the software side and the accelerator parameter settings on the hardware side when the model weights are compressed into binary format. From a specific activation precision, a set of accelerator parameters can be inferred, and the overall resource utilization and inference performance can thus be estimated in advance. If the estimated frame rate meets the target, then the corresponding activation precision is decided to guide the quantization process, and the corresponding accelerator settings are adopted for hardware implementations. Depending on the specific model structure and target frame rate, this compilation step costs several minutes to several hours, which is less than one tenth of the training time for quantization. As for the ViT accelerator, a general compute engine is included to handle both fully-connected (FC) layers with one matrix multiplication and multi-head attention layers performing the matrix multiplication
for multiple times. Because of the binary weights, the quantized computations can be replaced with additions and subtractions, and are thus implemented with lookup tables (LUTs) on FPGAs. The computations along the output channel, input channel, and head dimensions are totally or partially pipelined following the accelerator settings. Besides, the data packing technique is adopted to mitigate the memory burden and improve the overall computation throughput of the implementations.

Specifically, when binarizing the weights, our quantized ViT can achieve 79.5% on the ImageNet-1K validation set, which is just a 2.3% performance drop compared with the full-precision model (81.8%) but brings 32× reduction in model size. When further quantizing the activations to 8-bit and 6-bit, our method can still achieve accuracy of 77.6% and 76.5%, respectively, which outperforms previous state-of-the-art lightweight ViTs. The accelerator implementations are developed with Vivado High-Level Synthesis (HLS) and evaluated on the Xilinx ZCU102 FPGA board with the DeiT-base model. The experimental results indicate that 8-bit activation quantization is necessary for an inference frame rate of 24 frames per second (FPS), and 6-bit activation quantization is needed for 30 FPS.

Our main contributions are summarized as follows:

- We firstly build the quantized vision transformer with binary weights and low precision activations and achieve new state-of-the-art performance compared with other lightweight ViTs.
- We propose VAQF, a fully automatic framework that guides both quantization training and FPGA mapping. Given the target frame rate, VAQF generates the required quantization precision and accelerator description for direct software and hardware implementations.
- We design a layer-specific optimization scheme, and comprehensive FPGA optimization techniques are utilized to fully explore the data efficiency, execution parallelism, and resource utilization that maximize the performance and achieve real-time execution for quantized ViT models.

The rest of the paper is organized as follows. Section II discusses the related work in ViT compression and inference acceleration on hardware. Section III provides an overall flow of VAQF. The ViT quantization methods are presented in Section IV, and the implementation details of ViT acceleration with quantization on FPGAs are introduced in Section V. The experimental results are reported in Section VI. Finally, Section VII concludes the paper.

II. RELATED WORK

A. Vision Transformers

The Vision Transformer (ViT) architecture is firstly proposed in [11], which uses the attention mechanism [40] to solve various vision tasks. Compared to traditional CNN structures that operate on a fixed-sized window with restricted spatial interactions [33], ViT allows all the positions in an image to interact through transformer blocks. Since then, many variants have been proposed [13, 23, 45, 42, 14, 44, 6, 87]. For example, DeiT [39], T2T-ViT [46] and Mixer [9] tackle the data-inefficiency problem and make ViT trainable only with ImageNet-1K [10]. PiT [13] proposes a pooling-based Vision Transformer architecture with a desirable spatial dimension for better model capability and generalization performance. LV-ViT [16] introduces a token labeling approach to improve training. PS-ViT [47] abandons the fixed length tokens with progressive sampled tokens.

B. Transformer Quantization

Since the BERT era [40], quantization has been extensively studied to reduce the memory and computation complexity of the transformer architectures [30, 48, 56, 49, 11]. In detail, [45] finetunes the BERT with 8-bit quantization-aware training, and successfully compresses BERT with minimal accuracy loss. To avoid severe accuracy drop, [36] uses the mixed-precision quantization. The later TernaryBERT [49] proposes to use approximation based and loss-aware ternarization to ternarize the weights in the BERT, and use distillation to further reduce the accuracy drop caused by lower capacity. The BinaryBERT [11] suggests that it is difficult to train a binary BERT directly due to its complex loss landscape, and proposes ternary weight splitting strategy to make the binary BERT inherit the good performance of the ternary one. However, all of them are designed for NLP, not for the computer vision tasks. The most recent work [25] evaluates the post-training quantization on ViT and achieves comparable performance with full-precision ViT. However, they just push the compression ratio to 4× (i.e. 8-bit) and the method is not tailored for acceleration on hardware like FPGAs.

C. Lightweight ViTs

Unlike CNNs, the ViT [11] is typically more cumbersome and computationally extensive. Therefore, recently, lots of studies try to make ViT lightweight and efficient. These works usually revise the architecture of ViT [39] and introduce the image-specific inductive bias of CNNs back into ViT [42, 46, 8] to reduce the number of parameters and complexity. In detail, the DeiT [39] introduces the distillation token and applies knowledge distillation to transfer knowledge from pre-trained networks to lightweight ViTs. The T2T [46] proposes progressive tokenization to model the local structure information. The Cross-ViT [5] uses a dual-branch transformer to combine image patches of different sizes to produce stronger image features, and propose a cross attention module to reduce computation. The MobileViT [27] combines the strengths of CNNs and ViTs by replacing local processing in convolutions with global processing using transformers. All of these methods heavily modify the architecture of the standard ViT, while our quantized ViT still maintains the origin network architectures.

D. Transformer Acceleration on FPGAs

Model compression techniques have been applied and adjusted for transformer acceleration on hardware. The approach in [18] leveraged block-circulant structure for weight representation and converted the matrix-vector multiplications in FC layers to
FFT/IFFT computations on FPGA. Block-based weight pruning was applied to transformers in [31] with block-balanced sparsity and in [29] with column balanced block-wise sparsity. The balance in these methods means that the number of sparse rows or columns in each weight block is the same, or the number of sparse blocks along each column is kept the same. This facilitates full computation resource utilization and high throughput of hardware implementations.

While pruning needs balanced sparsity for high resource usage efficiency, quantization is naturally more friendly to FPGA implementations. The method in [22] employed $8 \times 4$-bit and $8 \times 8$-bit quantization on different parts of BERT. VAQF differs from previous work in the following aspects: 1) the quantization process is guided by the compilation step that determines the required activation precision given the target frame rate; 2) The precision for activation quantization is chosen from a wider range to meet a specific real-time frame rate requirement.

III. VAQF OVERVIEW

Figure I illustrates VAQF that builds an FPGA-based ViT inference accelerator. The ViT structure and desired frame rate (target FPS) are provided as input information. A compilation step is conducted to decide the required precision for activations with the accelerator settings to satisfy the FPS target, when the weights are binary. Specifically for an activation precision, a set of accelerator parameters can be inferred, and the overall resource utilization and inference performance can thus be estimated in advance. If the estimated frame rate meets the target, then the corresponding activation precision and accelerator settings are simultaneously decided. On the software side, this activation precision guides the quantization with the PyTorch library, as explained in Section IV and the quantized ViT parameters are sent to the FPGA platform for model inferences. On the hardware side, the parameter settings are adopted for the accelerator, as described in Section V. This compilation step costs several minutes to several hours depending on factors such as the target frame rate, the number of model layers, and the layer dimensions. Compared with quantization that takes days for training, the compilation time is small.

The accelerator description in C++ format is then generated and synthesized with the Vivado HLS tool. The rules of setting the accelerator parameters include the initial settings to maximize the computation parallelism under the specific precision, and the adjustments of parameters if the Vivado implementation fails because of placement or routing issues. The parameters may be slightly adjusted once or twice, and a successful implementation generates a bitstream file to be deployed on FPGAs. The synthesis and implementation for the accelerator take hours, which is also small compared with quantization training.

The key of this flow is to determine the required activation precision. For a baseline accelerator, 16-bit fixed-point numbers are used to represent the 32-bit floating-point parameters and activations of unquantized models without introducing accuracy loss. The activation precision will therefore be chosen from range 1 to 16 bits for higher throughput than the baseline design. As the frame rate has a reciprocal relationship with the total inference time of all the model layers, maximizing the frame rate is equivalent to minimizing the model inference time. The calculation of inference time will be elaborated in Section V-C3. The theoretical maximum frame rate for a ViT structure, denoted by FR$_{\text{max}}$, can be obtained assuming the activation precision is 1-bit, i.e., both weights and activations are binary. For a given target frame rate FR$_{\text{tgt}}$, the feasibility for accelerator implementation is first assessed by comparing FR$_{\text{tgt}}$ with FR$_{\text{max}}$. FR$_{\text{tgt}}$ ≤ FR$_{\text{max}}$ means the accelerator supporting a frame rate no lower than FR$_{\text{tgt}}$ can be implemented, and the appropriate precision is found through a binary search procedure. With a selection range of 1 to 16 bits, up to four rounds of search are conducted to find the required precision. In addition, if there exist multiple frame rate targets, all the possible precisions can be evaluated.

IV. VI T QUANTIZATION METHOD

A. Preliminary: Vision Transformer

In this section, we revisit the architecture of the visual transformer (ViT). We give details about its each component.

Patch Embedding The ViT firstly processes a 2D image into a sequence of flattened 2D patches [11] to adapt the 1D input sequence of the standard transformer. In detail, we denote the RGB image as $I \in \mathbb{R}^{H \times W \times 3}$, where $(H, W)$ is the resolution of the original image. $I$ is decomposed into $x_p \in \mathbb{R}^{N_p \times (3 \cdot P^2)}$, where $P$ is the resolution of each patch, and $N_p = HW/P^2$ is the total number of patches. To fit the dimension of the hidden vector $M$, ViT applies a linear transformation $W \in \mathbb{R}^{(3 \cdot P^2) \times M}$ to project $x_p$ into $X_1 \in \mathbb{R}^{N_p \times M}$. The ViT also appends the trainable [CLS] token $X_{\text{cls}} \in \mathbb{R}^{1 \times M}$ to maintain positional information. Thus, the input to the transformer encoder can be represented as,

$$X_0 = [X_{\text{cls}}, X_1] + E_{\text{pos}} \in \mathbb{R}^{(N_p+1) \times M} \quad (1)$$

Where $[\ldots]$ is the concatenation operation. For simplicity, we denote $F = N_p + 1$ as the total number of tokens.

Transformer Encoder Layers The transformer encoder includes several layers of multi-headed self-attention (MHSAs) and multi-layer perceptron (MLP) blocks. The Layernorm (LN) is applied prior to each block. Figure 2 shows the architecture of the transformer encoder layer, and operations in one encoder layer is summarized as:

$$X_{l}' = \text{MHSAS} (\text{LN}(X_{l-1})) + X_{l-1}, \quad l = 1 \ldots L \quad (2)$$

$$X_1 = \text{MLP} (\text{LN}(X_{l}')) + X_{l}', \quad l = 1 \ldots L$$

In detail, the self-attention function aims to match a query and a set of key-value pairs to an output [40]. Given input sequence $X_l$, three linear projections $W^q_l, W^k_l, W^v_l \in \mathbb{R}^{M \times M_h}$ are firstly applied to transfer $X_l$ into the query $Q_l \in \mathbb{R}^{F \times M_h}$, the key $K_l \in \mathbb{R}^{F \times M_h}$, and the value $V_l \in \mathbb{R}^{F \times M_h}$. The attention function is calculated by

$$\text{Softmax}(Q_lK_l^T/\sqrt{D})V_l \in \mathbb{R}^{F \times M_h} \quad (3)$$
Furthermore, for multi-head self-attention, \( N_h \) self-attention functions are applied to the input \( X \), and each of them produces an output sequence. The final output is the projection of the concatenated sequences. Typically, \( M_h \) is set to \( M/N_h \) to maintain the consistency between the size of inputs and outputs.

The MLP block consists of two linear layers followed by a GELU activation. The first linear layer (with weights \( W_{l1}^{m1} \in \mathbb{R}^{M \times 4M} \) ) expands the dimension from \( M \) to \( 4M \), and the second layer (with weights \( W_{l2}^{m2} \in \mathbb{R}^{4M \times M} \) ) reduces it back to \( M \).

**Output Head** Different from the pooling layers used in computer vision, ViT directly adds the output head upon \( X_0^i \), which is the representation of the CLS token and serves as the embedding of the entire image. The output head is defined as:

\[
y = \text{LN}(X_0^i)W_{\text{out}},
\]

where \( W_{\text{out}} \in \mathbb{R}^{M \times C} \), and \( C \) is the total number of classes.

### B. ViT Quantization

Similar to [24], [1], we binarize the weights in ViT, and reduce the activations into low-precision, to achieve the trade-off between efficiency and accuracy.

**Binary Weights** Following the definition in [24], [24], given the matrices of real number weights \( W_r \), the matrices of binary weights \( W_b \) are obtained by

\[
w_b = \frac{\|W_r\|_1}{n} \text{Sign}(w_r) = \begin{cases} +\frac{\|W_r\|_1}{n}, & \text{if } w_r > 0 \\ -\frac{\|W_r\|_1}{n}, & \text{if } w_r \leq 0 \end{cases}
\]

where \( w_r \) and \( w_b \) denote one specific element in the matrix \( W_r \) and \( W_b \), respectively. \( \frac{\|W_r\|_1}{n} \) is the scaling factor to minimize the difference between binary and real-valued weights. For ViT, all trainable weights in the transformer encoder layers belong to the linear transformation. Thus, we simply replace the weights from \( W_r \) to \( W_b \) in that linear layer to obtain the activations.

**Progressive Binary Training** As suggested by [1], training a binary BERT from scratch is challenging, as the loss landscape is steep. To alleviate this issue, we propose the progressive binary training strategy. Specifically, during the training process, we randomly select \( p\% \) elements in \( W_r \), and only binarize these selected elements while keep other elements full-precision. \( p\% \) is set to 0\% at the beginning of the training, then it grows linearly with the increase of training epoch, and achieves 100\% when the training process is completed. Therefore, \( W_p \) consists of both binary weights and real-value weights during the training process. Specifically,

\[
W_p = M_p \cdot W_b + (1 - M_p) \cdot W_r,
\]

where \( M_p \) is a mask with the same size as \( W_r \). During training, \( p\% \) elements of the mask are set to 1, while the rest are 0.

**Implementation Details** Typically, both the first layer and the output head are not quantized in previous binary networks [24], as they are associated with the inputs and outputs. Similarly, we only consider quantizing the weights and activations within each transformer encoder, and for the patch embedding and the output head, full-precision weights and activations are utilized. Specifically, we binarize the weights in all attention layers (\( W_i^m \), \( W_i^k \), and \( W_i^v \)) and the weights in MLP layers (\( W_{l1}^{m1} \) and \( W_{l2}^{m2} \)) with Eq. (5). We consider a three-step training process: 1) Train a full-precision ViT from scratch and achieve the real-valued parameters; 2) Finetune the full-precision ViT with progressive binary training to obtain ViT with binary weights and full-precision activations; 3) Finetune the binary-weight model to quantize activations with desired precision.
V. VISION TRANSFORMER ACCELERATION ON HARDWARE

In this section, we first discuss the optimization techniques for different types of computations in ViT layers (Sections V-A and V-B), and then provide a detailed flow of optimizing the accelerator parameters given the model structure and desired frame rate (Section V-C).

A. Compute Engine for Fully-Connected and Attention Layers

The implementation details of ViT acceleration on FPGA are displayed in Figure 3 and the notations of variables and parameters are listed in Table I. As the storage and computation resources are limited on FPGA devices, the loop tiling technique is adopted to split the input, weight, and output data for each ViT layer into tiles, and tiling for an unquantized layer is shown as an example in Figure 3(a).

| Notation | Description |
|----------|-------------|
| \( N \) | Number of output (input) channels |
| \( P \) | Number of token sequences |
| \( T_m \) (\( T_m^0 \)) | Tiling size for unquantized (quantized) data in output channel dimension |
| \( T_n \) (\( T_n^0 \)) | Tiling size for unquantized (quantized) data in input channel dimension |
| \( N_h \) | Total number of heads for computation in parallel |
| \( G \) (\( G^q \)) | Number of unquantized (quantized) data packed as one |
| \( S_{port} \) | Size of each AXI port on FPGA |
| \( p_{in} \), \( p_{out} \), \( p_{wgt} \) | Number of AXI ports used for data transfer of input (output, weight) tile |
| \( J_{in} \) (\( J_{wgt} \), \( J_{out} \), \( J_{compt} \)) | Number of clock cycles for input transfer (weight transfer, output transfer, computation) for a group of tiles |
| \( B_m \) (\( B_{out} \), \( B_{wgt} \)) | Number of BRAMs used by input (output, weight) tile |
| \( S_{bram} \) (\( S_{dsp} \), \( S_{lut} \)) | Available number of BRAMs (DSPs, LUTs) on FPGA |

Table I: NOTATIONS FOR THE ViT ACCELERATOR.

The most computation-intensive layers in ViTs include FC layers that exist in both multi-layer perceptron (MLP) modules and multi-head attention modules, and scaled dot-product attention layers that appear in multi-head attention modules. The primary computations of these two types of layers are both matrix multiplications. While an FC layer performs only one matrix multiplication, multi-head attention repeats the computations multiple times in parallel, so the accelerator is designed such that the computations can be executed in parallel across \( P_h \) attention heads. To make the design also compatible for FC computations, the \( N \) input channels in an FC layer are split into \( N_h \) groups, \( P_h \), of which are processed by the compute engine simultaneously at a time. A control signal is added to indicate whether the current layer is multi-head attention. For multi-head attention, the \( N_h \) heads of results are kept as they are, while for an FC layer, these results from \( N_h \) input channel groups are added together, making the final result for each output channel accumulated from all the input channels.

The main computation flow general for these two types of layers is displayed in Figure 3(b). The loops L2, L3, L4 under L1 are unrolled and pipelined so that the compute engine can manage \( T_m \cdot P_h \cdot T_n \) multiply-accumulate (MAC) operations in parallel. For illustration simplicity of the MAC operation with the input and weight, the head and the input channel dimensions are shown separately. Figure 3(c) provides the processing flow of one ViT layer with or without quantization. The accelerator parameters needed for the compute engine include \( T_m \) (\( T_m^0 \)), \( T_n \) (\( T_n^0 \)), \( G \) (\( G^q \)), and \( P_h \), the settings and adjustments of which are elaborated in Section V-C. Unquantized computations with high precision are managed by the DSP resources on FPGA, whereas quantized computations can be replaced with additions or subtractions that would be managed by LUTs, since the weight value is binarized as either +1 or −1.

B. Processing of Other Layers in Vision Transformer

In addition to matrix multiplications in MLP and multi-head attention modules, ViTs contain convolution, scaling, softmax, activation, normalization, 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 is the same as the patch size \( P \), meaning that the input data are used only once when a weight kernel slides across the input feature map, as shown in Figure 3 where \( C \), \( H \) and \( W \) denote the number of input channels, the height and width for the original input feature. The scaling, softmax, and GELU activation operations are performed on the host CPU of the FPGA, which introduces very small latency overhead for embedded FPGAs compared with matrix multiplications.

1) Processing of Normalization Layers and Skip-Connections: In ViTs, layer normalization is applied at the beginning of each multi-head attention and each MLP module, and there is an identity skip-connection linking the input activations of each normalization layer and the output activations of the subsequent module, which can be seen from Figure 2. In the sequential processing of the layers, the inputs of the normalization layer require to be stored for later additions with the outputs of the following multi-head attention or MLP module. Since normalization layers are not so compute-intensive as FC ones, their parameters and inputs are not quantized but rather kept with higher precision, namely 16-bit on hardware to preserve the model accuracy. As a result, two data transfer ports are needed, one for the inputs of normalization that are stored as unquantized data, and the other for the outputs of normalization, also the inputs for the subsequent FC layer, which are stored as quantized data. It needs to be emphasized that the data transfer port for quantized FC inputs is necessary to minimize the input loading time for the FC layer as the input tiles would be loaded multiple times in the matrix multiplication with loop tiling.
C. Design of FPGA-Based Vision Transformer Accelerator

Before building the ViT accelerator supporting a specific frame rate, a baseline accelerator is realized for unquantized models, whose 32-bit floating-point parameters and activations are represented with 16-bit fixed-point numbers to reduce the computation and storage resource utilization without accuracy loss on hardware. Let us denote the optimized parameters for the baseline design as $T_{\text{base}}^m$, $T_{\text{base}}^n$, and $G_{\text{base}}$. These parameters are treated as the starting point to find the required activation precision and optimize the ViT accelerator parameters.

Provided the target frame rate, VAQF will determine the proper activation precision, which is an inverse procedure, as the activation precision directly affects the accelerator settings and hence the actual frame rate. The following discussion of hardware implementation details is on the basis of a specific activation precision, but it does not mean the proper precision has been decided in the first step. Instead, it is found among several possible precisions after the analysis in terms of resource utilization and inference performance.

1) Data Packing: The data packing technique is employed to reduce the block RAM (BRAM) usage and the latency of data transfer between off-chip memory (DDR) and on-chip memory (BRAM). Although each BRAM on Xilinx FPGAs can accommodate 18k-bit data, the space of the whole BRAM might not be fully utilized, leading to high BRAM usage in the ViT acceleration implementation. This mainly results from loop pipelining and unrolling that requires each related data array to be partitioned into multiple smaller arrays. Data packing can mitigate this issue through concatenating multiple low-precision numbers as one number. With a data packing factor of $G$, the overall BRAM usage can be reduced by up to $G$ times, and the number of clock cycles for input loading and output storage can be reduced by $G$ times as well.

Each AXI port on FPGA with size $S_{\text{port}}$ can accommodate $G$ unquantized data or $G^q$ quantized data. If $S_{\text{port}} = 64$, then $G = 4$ for 16-bit data used in our baseline design, and $G^q = 8$ if the quantization precision is 8-bit. A special case is when $S_{\text{port}}$ cannot be divided exactly by the bit-length of quantized values. Take 6-bit quantization precision as an instance, $G^q = \lceil \frac{64}{6} \rceil = 10$, and only 60 of the 64 bits are exploited. Data packing is performed in the input channel dimension of the weights and input activations, and in the output channel dimension of the output activations. It is not displayed in Section 3(b) for illustration simplicity of MAC operation.

2) Determining Parameters for Best Computation Parallelism: Although most layers of the ViT model are quantized, the first and last layers are not quantized to better maintain the model accuracy, and the accelerator compute engine still needs to handle unquantized data. As a result, we have two groups of accelerator parameters to be determined, i.e., $T_m$, $T_n$, and $G$: 

Fig. 3. Detailed implementation of ViT accelerator. (a) Loop tiling of input, weight, and output for one model layer; (b) Computation flow in compute engine with loop tiling, pipelining and unrolling; (c) Processing flow of one model layer based on whether it is quantized or not.

Fig. 4. Conversion of input and weight tensors for the first convolutional layer.
for unquantized layers, and \( T^q_m \), \( T^q_n \) and \( G^q \) for quantized ones, while the parameter \( P_h \) remains the same in both situations. The ViT layers are handled one by one, so the accelerator will not perform unquantized computations and quantized ones simultaneously. Nonetheless, the same BRAMs for input, weight and output data can be utilized whether the layer is quantized or not. The related accelerator parameters are therefore decided to make the best effort to utilize the BRAMs.

When creating the ViT accelerator with support of quantization, the accelerator parameters for unquantized layers are first set as \( T_m = T^\text{base}_m \) and \( G = G^\text{base} \), and \( T_m \) is initially set to a value that is near to \( T^\text{base}_m \) and can be divided exactly by \( G \) and \( G^q \). As mentioned in Section [V.C], \( G^q \) is directly calculated according to \( S_{p_{\text{rot}}} \) and the required quantization precision. \( T^q_n \) is then calculated by \( T^q_n = \left[ T_m \cdot \frac{G^q}{G} \right] \) for maximum utilization ratio of BRAMs for quantized data. \( P_h \) is usually a value that can divide \( N_h \) exactly. For instance, if \( N_h = 6 \), \( P_h \) is set to 3; if \( N_h = 8 \) or \( N_h = 12 \), then \( P_h \) is 4. In the next step, \( T_m \) is set as equal to \( T_m \), and a hardware design with all these parameters is synthesized and implemented with Vivado HLS as an initial try. If the FPGA board cannot accommodate this design due to placement or routing issues (usually resulting from overutilization of LUTs), then \( T_m \) and \( T^q_m \) are adjusted for the implementation to fit into the board and minimize the overall inference latency as well. For low activation precision, \( T_m \) is reduced and \( T^q_m \) is increased until the FPGA resources are fully exploited. In this procedure, both \( T_m \) and \( T^q_m \) are kept as values that can be divided exactly by \( G \) and \( G^q \) for convenience of output storage.

3) Performance Analysis and Objective Function: The main variables in the equations in this section are explained in Table[1]. For one layer \( i \) in ViT, the numbers of clock cycles needed for input tile loading, weight tile loading, and output tile storage are calculated as

\[
\begin{align*}
J_{in} &= N_h \cdot \left( 1 - \alpha \right) \cdot \left[ \frac{T_m}{G} \right] + \alpha \cdot \left[ \frac{T^q_m}{G^q} \right] \cdot \left( 1 + \beta \right) \cdot \left[ \frac{F}{P_{in}} \right], \\
J_{wgt} &= N_h \cdot \left( 1 - \alpha \right) \cdot \left[ \frac{T_m}{G} \right] + \alpha \cdot \left[ \frac{T^q_m}{G^q} \right] \cdot \left( 1 + \beta \right) \cdot \left[ \frac{F}{P_{wgt}} \right], \\
J_{out} &= (1 + \gamma) \cdot \left( 1 - \beta \right) \cdot \left[ \frac{T_m}{G} \right] + \beta \cdot \left[ \frac{T^q_m}{G^q} \right] \cdot \left( 1 + \beta \right) \cdot \left[ \frac{F}{P_{out}} \right],
\end{align*}
\]

where \( \alpha \) is 1 if the inputs and weights are quantized else 0, \( \beta \) is 1 if the outputs are quantized else 0, and \( \gamma \) 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

\[
J_{\text{cmpt}} = F \cdot \left[ \frac{N_h}{P_h} \right].
\]

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

\[
J_{lc} = \max \{ J_{in}, J_{wgt}, J_{\text{cmpt}} \}.
\]

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

\[
J_s = \max \left\{ J_{in} \cdot \left( 1 - \alpha \right) \cdot \left[ \frac{N}{N_h \cdot T_n} \right] + \alpha \cdot \left[ \frac{N}{N_h \cdot T^q_n} \right], J_{cmpt}, J_{out} \right\}.
\]

The overall clock cycle number for a ViT layer \( i \) is therefore described by

\[
J_i = \left( 1 - \beta \right) \cdot \left[ \frac{M}{T_m} \right] + \beta \cdot \left[ \frac{M}{T^q_m} \right] \cdot J_s + J_{out}.
\]

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

\[
\begin{align*}
B_{in} &= 2 \cdot \left[ \frac{T_m}{G} \right] \cdot \left[ \frac{F \cdot G \cdot 16}{18k} \right], \\
B_{wgt} &= 2 \cdot \left[ \frac{T^q_m}{G^q} \right] \cdot \left[ \frac{T_m \cdot G \cdot 16}{18k} \right], \\
B_{out} &= 2 \cdot \left[ \frac{T^q_m}{G^q} \right] \cdot \left[ \frac{F \cdot G^q \cdot b^q}{18k} \right],
\end{align*}
\]

where \( b^q \) is the activation bit-width in quantization. As for the DSP utilization, since each DSP can manage one unquantized MAC operation with a 16-bit input and a 16-bit weight, the number of used DSPs is calculated by \( T_m \cdot P_h \cdot T_n \) to perform \( T_m \cdot P_h \cdot T_n \) MAC operations in parallel.

Finally, our objective function can be written as

\[
\text{minimize } \sum_i J_i
\]

subject to

\[
B_{in} + B_{wgt} + B_{out} \leq S_{\text{bram}}, \\
T_m \cdot P_h \cdot T_n \leq S_{\text{dsp}} \cdot r_{\text{dsp}}, \\
C_{\text{lut}} \cdot T^q_m \cdot P_h \cdot T^q_n \leq S_{\text{lut}} \cdot r_{\text{lut}},
\]

where \( r_{\text{dsp}} \) and \( r_{\text{lut}} \) are respectively the maximum ratio of DSPs and LUTs to be utilized for MAC operations, and \( C_{\text{lut}} \) is the LUT cost for one MAC with quantized operands. As mentioned in Section [III] minimizing the overall model latency is equivalent to maximizing the frame rate. If the maximum frame rate is no lower than the target frame rate, then the accelerator with the corresponding activation precision and parameter settings is feasible.

VI. Evaluation

A. Experimental Setup

Datasets & Architectures We conduct all experiments on ImageNet-1K[10] and use the top-1 accuracy on validation set as the evaluation metric. All images are resized to 224. We use the DeiT-base without the distillation token as the default ViT architecture[39]. We consider three types of quantization precision: 1) binary weights, and full-precision activations (W1A32); 2) binary weights, and 8-bit activations (W1A8); 3) binary weights, and 6-bit activations (W1A6).

Training We strictly follow the training settings in [39] for all the three training stages. In detail, the network is trained...
for 300 epochs and is optimized by AdamW [26] with weight decay 0.05. The batch size is set to 512. The learning rate is set to \(5 \times 10^{-4}\) initially and is decayed with a cosine annealing schedule. All data augmentations in [39] are also included during the training process.

**Hardware** The ViT accelerators with different precisions are implemented on the Xilinx ZCU102 FPGA platform with 2520 DSPs and 274k LUTs, while VAQF can be generalized to other types of FPGAs. For all the implementations, the operating frequency is set to 150 MHz to maximize the computation efficiency and avoid timing violation. The hardware implementations are preformed with the HLS tool of Xilinx Vivado 2020.1.

**B. Results on Software**

1) **Comparison with Other Lightweight ViTs:** We first compare our quantized DeiT-base with other lightweight ViTs that are trained from scratch on the ImageNet-1K dataset. All the models are trained without additional distillation. We also report the space usage of devices for each model/method, which is calculated as the product of the number of parameters and the precision. All results are shown in Table I. Firstly, all other lightweight ViTs are still full-precision models, while the weights in our method are all binarized. It can be seen that with binarized weight, it is feasible to consume much less storage space of devices without changing the original network architecture. Secondly, compared with the full-precision DeiT-base, our weight-binarized version only drops 2.3% accuracy (from 81.8% to 79.5%). When further quantizing the activations to low precision, the accuracy can still maintain 77.6% for 8-bit and and 76.5% for 6-bit, which is still much better than other lightweight ViTs. Although quantizing the activations does not reduce the model size on devices, it can reduce the inference time on the hardware. More details of the efficiency evaluation will be discussed in Section VI-C.

**TABLE II**

| Method              | Accuracy (%) | Space Usage |
|---------------------|--------------|-------------|
| DeiT-base           | 81.8         | 86M \(\times\) 32 |
| T2T [46]            | 71.7         | 4.7M \(\times\) 32 |
| DeiT [39]           | 72.2         | 5.7M \(\times\) 32 |
| PiT [15]            | 73.0         | 4.9M \(\times\) 32 |
| Cross-ViT [5]       | 73.4         | 6.9M \(\times\) 32 |
| MobileViT [27]      | 74.8         | 2.3M \(\times\) 32 |
| Ours (DeiT-base-W1A32) | 79.5         | 86M \(\times\) 1 |
| Ours (DeiT-base-W1A8)  | 77.6         | 86M \(\times\) 1 |
| Ours (DeiT-base-W1A6)  | 76.5         | 86M \(\times\) 1 |

2) **Ablation Studies:** **Architecture of ViT** Besides DeiT-base, we also evaluate the accuracy of mix-quantization on DeiT-tiny and DeiT-small [39] on ImageNet. In detail, the number of parameters of DeiT-tiny is 5 million, and that of DeiT-small is 22 million, due to the smaller number of embedding dimension and fewer heads. Table III presents the comparison between the full-precision model (W32A32) and the model with binary weights and full-precision activations (W1A32). Different from the DeiT-base, after binarizing their weights, the accuracy of both DeiT-tiny and DeiT-small drop heavily. We hypothesize that these lightweight networks are very fragile as the number of parameters is quite limited. As a consequence, when binarizing their weights, the learning capacity is heavily restricted and thus it is challenging to learn the spatial inductive bias for vision from scratch [11]. Therefore, it is difficult to binarize these lightweight ViTs and maintain their accuracy.

**TABLE III**

| Quantization Precision | DeiT-tiny | DeiT-small |
|------------------------|-----------|------------|
| W32A32                 | 72.2      | 79.9       |
| W1A32                  | 51.5      | 70.4       |

**Training Schedules** In this section, we aim to show the effect of three-stage training steps and progressive binarization strategy. Due to the computing limitation, we randomly sample 100 categories from the full ImageNet datasets (named “ImageNet-100”), and conduct the ablation studies on this subset. Specifically, we take 32-bit pre-training or progressive binary training out of the training procedure to evaluate the effectiveness of each strategy. Table IV shows the ablation results. All the results are based on DeiT-small architecture. Remarkably, without the full-precision pre-training, there is a huge drop of accuracy. When further removing the progressive binarization, there is an additional 0.9% accuracy drop. These results verify the effectiveness of our proposed stage-wise finetuning strategy and progressive binarization.

**TABLE IV**

| Method                  | Accuracy (%) |
|-------------------------|--------------|
| W1A32                   | 84.3         |
| W1A32 (w/o pre-training)| 79.3         |
| W1A32 (w/o progressive  )| 78.4         |

**C. Results on Hardware**

1) **Resource Utilization and Performance of ViT Accelerators:** Table V summarizes the resource utilization and performance of the VAQF-generated ViT accelerators with various frame rates and precisions on FPGAs. The quantization precision is represented by \(W[q_w]A[q_a]\), where \(q_w\) is the bit-length for weights and \(q_a\) the bit-length for activations. For quantized models, the precision on hardware is the same as that on software, while for unquantized models, the parameters and activations in W32A32 precision on software are represented with W16A16 precision on hardware to reduce the computation efficiency and avoid timing violation.
and storage resource utilization without accuracy loss. The utilization of resources on FPGA is shown with the percentage of the used and total available number. The numbers of LUTs and FFs are expressed as thousands, and the number of BRAMs is listed treating the capacity of each BRAM as 36k bits. The performance of ViT implementations are obtained from the inference process of DeiT-base in terms of the frame rate in FPS, the throughput in giga operations per second (GOPS), and computation efficiency including GOPS per DSP and GOPS per thousands of LUTs.

The inference speed of the W1A8 design is 24.8 FPS, and that for W1A6 is 31.6 FPS. Compared with the baseline implementation with W32A32, these two designs respectively achieve 2.48× and 3.16× acceleration. In these low-precision accelerators, the parallelism in unquantized computations is degraded because of the reduction of DSP utilization, while that in quantized computations is enhanced since more LUTs are utilized. This assists the accelerators to attain higher overall throughput and better fit into the available computation resources, avoiding placement and routing issues. Higher throughput and lower DSP utilization results in 2.49× improvement for W1A8 and W1A6 designs, and this ratio for W1A6 can reach 7.37×. With slightly more LUT utilization for the quantized operations and extra logic to select between unquantized or quantized operations, the GOPS/kLUT increases by 2.09× and 2.29× for W1A8 and W1A6 designs, respectively.

These experimental results demonstrate that the accelerators generated by VAQF are able to achieve real-time inferences, i.e., a frame rate requirement of 24 FPS is satisfied with 8-bit quantization for activations, and a target of 30 FPS is met with 6-bit activation quantization. VAQF can be further generalized to other frame rate targets and other types of transformers.

2) Comparison with Other Implementations: Our accelerators are further compared with previous work, CPU and GPU with regard to FPS, power, and energy efficiency. Since no study using quantization has been carried out for ViT acceleration on FPGAs, the accelerators for BERT in [22] with 8 × 4-bit and 8 × 8-bit quantization are used for comparison, and other implementation results are all obtained for DeiT-base. As shown in Table [VI], our W1A8 and W1A6 accelerators both outperform the BERT design on ZCU102 in terms of FPS, power and energy efficiency, and the W1A6 design has the highest FPS/W among all implementations. Compared with the inferences on Intel(R) Core(TM) i7-9800X CPU and NVIDIA TITAN RTX GPU, our W1A6 design has 27.0× and 5.7× improvement in FPS/W.

VII. CONCLUSION

This work proposes a framework called VAQF for automatic construction of inference accelerators for ViTs with binary weights and low-precision activations under the desired frame rate. VAQF first performs compilation to decide the required activation precision with the accelerator settings related to the computation parallelism by analyzing the resource utilization and inference performance. And then for the accelerator implementation, a compute engine general for FC layers and multi-head attention layers is designed, and the quantized computations are replaced with additions and subtractions that are implemented with LUTs. Optimization techniques like data packing and loop pipelining are utilized with different factors for unquantized and quantized computations to enhance the computation parallelism and memory utilization efficiency. The experimental results for the DeiT-base model show that the accelerator with 8-bit activation quantization achieves an inference speed of 24.8 FPS, and that with 6-bit activation quantization attains 31.6 FPS, which can meet various real-time requirements.

REFERENCES

[1] 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. arXiv preprint arXiv:2012.15701, 2020.

[2] H. Bao, L. Dong, and F. Wei. Beit: Bert pre-training of image transformers. arXiv preprint arXiv:2106.08254, 2021.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2020.
