

HI-APP: Hardware-friendly Fully-Integer Approximation of Nonlinear Functions in Quantized CLIP-ViT

Beom Jin Kang and Hyun Kim

Department of Electrical and Information Engineering and Research Center for Electrical and Information Technology
Seoul National University of Science and Technology, Seoul 01811, Korea
beomjin@seoultech.ac.kr, hyunkim@seoultech.ac.kr

Abstract—Recently, vision–language models (VLMs) have delivered state-of-the-art multi-modal accuracy, yet deploying them on FPGA/ASIC accelerators remains costly: after quantizing general matrix multiply (GEMM) to low-bit integer math, non-linear functions (NLFs) dominate resources and power. Prior NLF approximations either adopt high-precision piecewise-linear (PWL) methods that still consume substantial logic/DSP budgets, or low-precision integer surrogates that necessitate fine-tuning to recover accuracy. In this paper, we present hardware-efficient, training-free approximations for two representative NLFs, namely GELU and LayerNorm. First, for GELU, we propose a power-of-two (PoT) PWL scheme: we analytically study the LUT-entry/accuracy trade-off under input clipping, introduce an automatic clipping-point selection to meet a target error, and convert segment slopes to PoT to replace multipliers with shifts. Second, for LayerNorm, we eliminate floating-point operations in quantized pipelines via a PoT-based mean estimator and a log-based shift-LUT approximation of the reciprocal square root for variance normalization. Both designs compile a common shift-add datapath and co-optimize naturally with quantized GEMM. On quantized CLIP-ViTs, our approach is plug-and-play (no additional training) and incurs at most a 0.93% Top-1 drop on ImageNet. A prototype on Xilinx FPGA reduces DSP usage by up to 100%, LUTs by 69.8%, and FFs by 96.0%, delivering substantial gains in resource efficiency and deployability. These results indicate that simple, PoT-driven approximations can cap NLF overheads and enable practical, resource-aware VLM acceleration on reconfigurable and custom silicon. We provide the HI-APP implementation at <https://github.com/IDSL-SeoulTech/HI-APP>.

Index Terms—Approximate computing, Vision–language model, Non-linear function, Quantization, FPGA Accelerator

I. INTRODUCTION

Recently, visual language models (VLMs) have shown strong performance in multi-modal tasks such as image captioning and visual question answering [1], [2]. However, the high computational cost of VLMs poses major challenges for on-device AI in applications such as physical AI and autonomous driving [3]. While GPUs offer high performance, they fundamentally fail to meet the power and resource

constraints of on-device environments [4]. As a result, VLM-specific FPGA accelerators have gained interest, yet deploying VLMs on FPGA requires extensive compute units for large-scale general matrix multiply (GEMM) and non-linear function (NLF) operations, driving substantial resource and power consumption [5], [6]. One major source of these challenges is the visual encoder (VE), which dominates VLM computation [7], [8]; in particular, CLIP-ViT [9], a widely used VE, processes many image tokens, significantly increasing FPGA resource demands [10]. Therefore, hardware-efficient VE design remains a critical challenge, and we focus on optimizing CLIP-ViT as a practical solution.

Meanwhile, as noted earlier, CLIP-ViT computations consist of GEMM and NLF, and most optimization studies (*e.g.*, quantization [11]–[13], low-rank approximation [14]–[16], pruning [17]–[19]) have focused mainly on GEMM. Among them, quantization reduces model size by converting high-precision activations and weights (*e.g.*, floating-point (FP) 16-bit) into low bit-precision integer (INT) data type (*e.g.*, 4/8-bit), while casting FP GEMM operations into INT GEMM for efficient computation [20], [21]. In addition, INT GEMM units are more hardware-efficient than FP GEMM units and significantly reduce power consumption, driving their widespread adoption in accelerator studies [22]–[24]. However, despite these efforts, NLF optimization remains an open challenge. NLF consumes substantial FPGA resources and forms a major bottleneck, limiting efficiency and scalability in the inference pipeline [25]–[28]. Thus, deploying CLIP-ViT on FPGA accelerators requires not only quantization-based GEMM optimization but also hardware-friendly NLF design.

Against this background, recent studies have explored various NLF optimization methods. PEANO-ViT [28] optimized key nonlinear functions using piece-wise linear (PWL)-based LUT approximation, while GQA-LUT [25] and RI-LUT [29] proposed similar LUT-based methods. QUNF [30] improved accuracy over prior approaches with quadratic polynomial approximation. However, prior studies show clear limitations in integrating with quantized networks. For example, some prior works [28]–[30] rely on high-precision fixed-point multiplications, causing significant FPGA resource usage, especially DSPs, the most constrained hardware units [31], [32]. In contrast, other approaches [25], [26] proposed approximation methods that support low-precision inference by considering quantized networks. However, they failed to preserve inference

This work was supported by the Technology Innovation Program(or Industrial Strategic Technology Development Program)(RS-2025-02307330, Development and Validation of On-Device AI Semiconductor for Manufacturing Automation Robots through sVLM-Based Situational Awareness) funded by the Ministry of Trade Industry & Energy(MOTIE, Korea) and K-CHIPS (Korea Collaborative & High-tech Initiative for Prospective Semiconductor Research) (RS-2025-02305531) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea). (Corresponding author: Hyun Kim)

accuracy due to reduced approximation precision and required additional training resources for accuracy recovery. Ultimately, an approximation method is needed that avoids the high resource cost of high-precision approaches and the accuracy loss of low-precision ones, while ensuring efficient and stable accuracy without extra fine-tuning. Moreover, although Softmax [33], GELU [34], and LayerNorm [35] are all core NLFs in the inference pipeline, prior research has focused almost exclusively on Softmax approximation [36], [37].

Motivated by the aforementioned challenges, this paper proposes *HI-APP*, a hardware-friendly NLF approximation method tailored to the integer computation flow of quantized CLIP-ViT. In particular, we focus on approximating GELU and LayerNorm while analyzing compatibility with existing Softmax methods. Specifically, for GELU, we derive the optimal input clipping range in the PWL approximation and represent slopes as combinations of power-of-two (PoT) terms, replacing multiplications with simple shifts. This significantly reduces computational complexity and improves hardware efficiency. In LayerNorm, we introduce a PoT-based grouped-mean calculation to eliminate the high-precision division in mean computation and apply a log-based approximation for reciprocal square root (RSQRT). These methods enable the efficient deployment of CLIP-ViT on FPGAs without additional training. Our contributions can be summarized as follows:

- **Fully-integer activation function approximation:** For GELU, we analyze the accuracy–LUT trade-off with respect to the input clipping range in PWL approximation and derive an optimal range analytically. Building on this, we approximate the PWL slope using PoT-term combinations, replacing multiplications with shifts to minimize DSP usage while reducing information loss. We further employ LUT-based linear interpolation to the fractional part, achieving high accuracy without extra training.
- **Efficient LayerNorm integerization:** To reduce the burden of division in LayerNorm’s mean computation, we propose a PoT-based grouped-mean method. For RSQRT, we implement a hardware-friendly log-based approximation: the integer part handled with shifts and the fractional part with LUTs, minimizing extra multiplications.
- **Hardware-aware integration for CLIP-ViTs:** By integrating the proposed GELU and LayerNorm approximations into quantized CLIP-ViTs, we ensure full compatibility with quantized GEMM. On ImageNet, *HI-APP* shows a 0.93% accuracy drop from the INT8 baseline. Our FPGA results demonstrate complete DSP elimination for GELU and substantial LUT and FF reductions—69.8% and 96.0% for LayerNorm, respectively.

II. RELATED WORKS

A. Activation Function Approximation

CLIP-ViT employs GELU [34] during training to enhance representational capacity through non-linearity, with its sigmoid-based approximation expressed as follows:

$$\text{GELU}(x) \approx x \cdot \sigma(1.702x) = x \frac{1}{1 + e^{-1.702x}} \quad (1)$$

Implementing the exponential and division operations in Eq. (1) requires extensive hardware resources, posing a major challenge for deployment on resource-constrained devices. To mitigate this, various approximation methods have been proposed. PEANO-ViT [28] restricts the GELU input range to improve PWL accuracy while reducing hardware costs. GQA-LUT [25] introduces a quantization-aware PWL scheme using integer breakpoints optimized via a genetic algorithm. QUNF [30] employs a high-precision second-order Taylor expansion and shift-add approximations for quadratic terms. However, existing methods still face limitations: (i) PWL-based approaches [25], [28], [38] overlook the trade-off between LUT entry count and accuracy, (ii) LUT-based approximations continue to rely on high-precision multiplications, resulting in heavy DSP usage, the most constrained resource on FPGAs, and (iii) low-bit integer approximations [25], [26] incur notable accuracy loss and require additional retraining, reducing their practical applicability.

To overcome these limitations, we first derive the optimal input clipping range to improve the accuracy-LUT trade-off. Then, we convert the PWL slope into PoT form, enabling shift-based operations and substantially reducing DSP usage. In addition, the precision loss in the fractional part from shift operations is compensated with LUT-based linear interpolation, preserving accuracy. Consequently, we propose an approximation method that achieves both hardware efficiency and accuracy without requiring additional training.

B. LayerNorm Approximation

LayerNorm is a widely used normalization method that improves training stability and convergence speed, defined as:

$$\text{LayerNorm}(x_t) = \frac{x_t - \mu_{mean}}{\sqrt{\sigma_{var}}} \cdot \gamma + \delta \quad (2)$$

where μ_{mean} and σ_{var} are dynamically computed for each input vector x_t , while γ and δ are learnable parameters. RSQRT, the core operation of LayerNorm, is computationally expensive and a major source of hardware resource consumption. Therefore, efficient implementations of RSQRT have been actively explored. For example, PEANO-ViT [28] proposed a division-free RSQRT approximation by combining a log-based method with LUT correction, whereas I-ViT [26] introduced an iterative SQRT approximation. In addition, QUNF [30] and GQA-LUT [25] applied a second-order polynomial approximation and a quantization-aware LUT approximation, respectively, similar to the activation function methods described earlier. However, when performing LayerNorm in quantized networks, the mean computation, as well as RSQRT, remains a major bottleneck. Specifically, mean calculation involves high-precision division, which consumes substantial hardware resources. To address this, we introduce PoT-based grouped-mean computation to eliminate high-precision division. In addition, we enhance hardware efficiency through log-based RSQRT approximation and preserve fractional precision using LUT-based approximation.

III. PROPOSED METHODS: HI-APP

A. Activation Function Approximation

1) Clipping Optimization for PWL-based Approximation:

Since GELU converges to 0 for small negative inputs and to $y = x$ for large positive inputs, input clipping is used to improve the precision of PWL approximation [25], [28], [38]. In particular, let the clipping bounds be ϕ_- and ϕ_+ . The GELU function is then expressed as follows:

$$\text{GELU}(x) \approx \begin{cases} 0 & \text{if } x < \phi_- \\ x \cdot \sigma(1.702x) & \text{if } \phi_- \leq x < \phi_+ \\ x & \text{if } \phi_+ < x \end{cases} \quad (3)$$

In this case, the interval $\phi_- \leq x < \phi_+$ is approximated with PWL, during which two types of errors occur. In particular, as shown in Fig. 1, clipping error (CE) occurs during the process of clipping the approximation region, while within the region, PWL approximation error (PAE) arises from the PWL approximation itself. CE and PAE behave oppositely with respect to the clipping range. As the range widens, CE decreases since more of the convergence region is covered, but PAE increases due to limited LUT entries. Conversely, a narrower range reduces PAE but raises CE as part of the convergence region is missed. Ultimately, the variation of CE and PAE with the clipping range creates a trade-off between LUT entries and accuracy. To maximize LUT efficiency in PWL approximation, it is thus crucial to search for the optimal clipping range that balances CE and PAE.

Accordingly, we propose a method to determine the optimal clipping range. The problem of finding ϕ_- and ϕ_+ that minimize CE is defined as follows:

$$\phi_- = \arg \max_{x < 0} x \quad \text{s.t.} \quad |\nabla g(x)| = 0, \quad |\nabla^2 g(x)| = 0 \quad (4)$$

$$\phi_+ = \arg \min_{x \geq 0} x \quad \text{s.t.} \quad |\nabla g(x)| = 1, \quad |\nabla^2 g(x)| = 0 \quad (5)$$

where $g(x)$ denotes the GELU function, which shows different convergence behavior in the negative and positive regions, requiring distinct conditions. In the negative region, GELU converges to 0 as $x \rightarrow -\infty$, so the CE-minimizing point is where both slope and curvature vanish. In the positive region, since $\text{GELU}(x) \rightarrow x$ as $x \rightarrow \infty$, the CE-minimizing boundary is where the slope approaches 1 and the curvature vanishes. However, no point satisfies both derivative conditions exactly due to GELU's shape. Thus, in practice, we relax these ideal conditions by introducing an error tolerance ω to approximate the clipping boundary.

To solve this optimization problem, we apply a gradient descent-based numerical optimization method. For a candidate clipping range \tilde{x} , we define the following objective function:

$$\mathcal{L}_{\phi_-}(\tilde{x}) = \nabla g(\tilde{x})^2 + \nabla^2 g(\tilde{x})^2 \quad (6)$$

$$\mathcal{L}_{\phi_+}(\tilde{x}) = (\nabla g(\tilde{x}) - 1)^2 + \nabla^2 g(\tilde{x})^2 \quad (7)$$

To obtain the solution, we initialize $\tilde{x} = 2.0$ for the positive region and $\tilde{x} = -2.0$ for the negative region, and apply gradient descent with a learning rate of 0.1. The value of \tilde{x} is

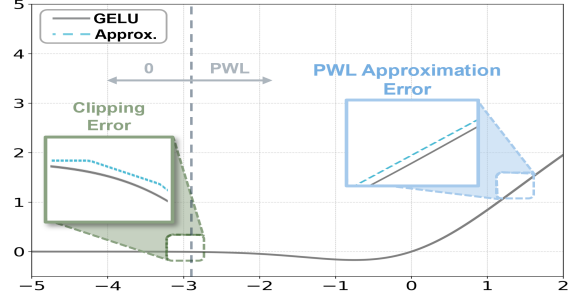


Fig. 1: Example of CE and PAE occurring with input clipping in PWL approximation

iteratively updated over 100 steps to derive a set of candidate solutions. Ultimately, under the condition of $\epsilon = 10^{-3}$, the optimal clipping range was derived as $[\phi_-, \phi_+] = [-3.3, 3.3]$. This range satisfies the condition $\{\mathcal{L}_{\phi_-}(x) < \omega, \mathcal{L}_{\phi_+}(x) < \omega\}$, while also serving as the region for minimizing PAE, indicating a practical boundary that jointly optimizes PAE and CE within the limited number of LUT entries.

2) *Hardware-Friendly GELU Approximation:* We approximate the sigmoid function directly to remove high-cost division operations. Using logarithmic properties, the sigmoid applied to dequantized inputs can be reformulated as follows:

$$\frac{1}{1 + e^{-s'x_q}} = 2^{-\log_2(1 + e^{-s'x_q})} \quad (8)$$

Here, x_q is the quantized input and $s' = 1.702s$, where s is the scale factor. We approximate $\log_2(1 + e^{-s'x_q})$ using PWL. The input range of $s'x_q$ is limited to $[-3.3, 3.3]$ and divided into N segments, each represented by a linear function. Slopes and biases are obtained by linear regression, and the linear functions for the N segments are given as follows:

$$y = \begin{cases} 0 & \text{if } x < \phi_- \\ \alpha_i s' x_q + \beta_i & \text{if } \phi_{i-1} \leq x < \phi_i, \quad i = 1, 2, \dots, N \\ s' x_q & \text{if } x \geq \phi_+ \end{cases} \quad (9)$$

where ϕ_- and ϕ_+ are the lower and upper bounds of the approximation range, set to -3.3 and 3.3 . The interval is uniformly divided into $N = 6$ segments, with α_i and β_i denoting the slope and bias of each segment. The approximated GELU function is then expressed as follows:

$$\text{GELU}(sx_q) \approx sx_q \cdot 2^{-(\alpha s' x_q + \beta)} = (2^{-\beta} s) \cdot x_q 2^{-\alpha s' x_q} \quad (10)$$

Ultimately, inference requires computing only $2^{-\alpha s' x_q}$ using hardware-friendly logic. At this point, implementing $\alpha s' x_q$ with fixed-point multiplication incurs heavy DSP usage, making it less efficient than LUT/FF-based logic [31], [32], [39]. Accordingly, to minimize DSP usage, we convert $\alpha s'$ into PoT terms and implement it with shift-based logic. However, converting $\alpha s'$ into a single PoT term causes severe accuracy loss. To address this, we apply multi-PoT decomposition (MPD), which splits $\alpha s'$ into multiple PoT terms to improve approximation accuracy [40]. Specifically, MPD with q PoT terms can be expressed as follows:

$$\alpha s' x_q \approx (2^{\epsilon_1} \cdot x_q + 2^{\epsilon_2} \cdot x_q + \dots + 2^{\epsilon_q} \cdot x_q) \quad (11)$$

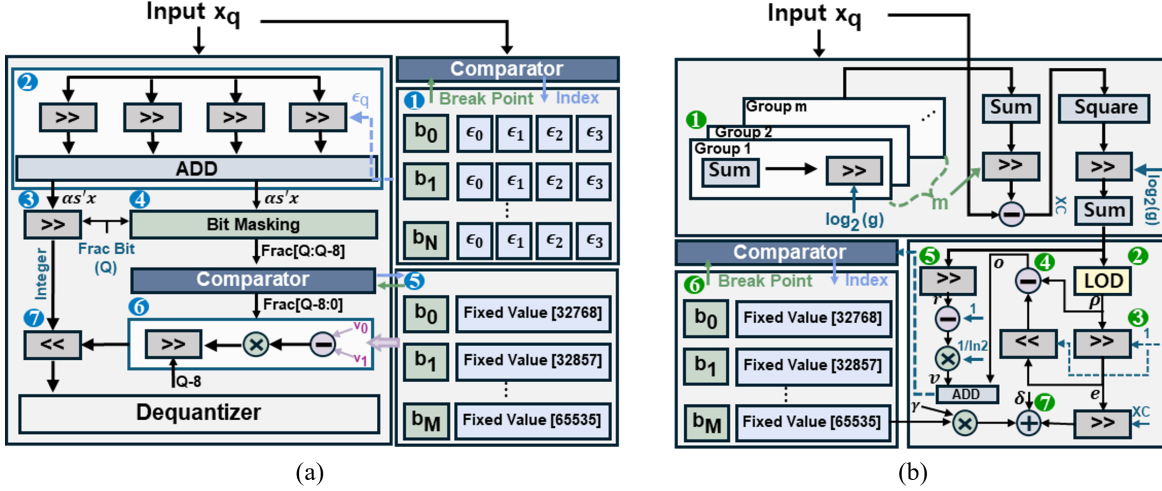


Fig. 2: Architecture of the proposed *HI-APP* approximation modules. (a) shows the architecture of the GELU approximation module, and (b) illustrates the architecture of the LayerNorm approximation module.

Here, 2^{ϵ_q} denotes an individual PoT term used in the polynomial expansion. Ultimately, MPD compensates for the error introduced during the PoT conversion process.

Lastly, we compute 2^{-b} (with $b = \alpha s' x_q$) by decomposing b into integer and fractional parts, handling the integer part with shifts and approximating the fractional part via a LUT. Directly storing $2^{b_{\text{frac}}}$ for all fractional bits $b_{\text{frac}} \in [0, 1)$ causes exponential LUT growth and excessive hardware cost. To overcome this limitation, we store a fixed number of LUT entries by uniformly partitioning the fractional range and approximating the remaining bits with linear interpolation. Specifically, for the fractional part b_{frac} , $2^{b_{\text{frac}}}$ is uniformly partitioned and stored in a 256-entry LUT. The upper 8 bits of b_{frac} are mapped to the segment index i to obtain the approximated value v_i . To further enhance the precision of approximation, the adjacent entry v_{i+1} is retrieved to compute the difference $\Delta v = v_{i+1} - v_i$. The residual bits z , which are not used in the LUT indexing process, are employed as interpolation weights to apply linear interpolation as follows:

$$2^{b_{\text{frac}}} \approx v_i + \frac{(v_{i+1} - v_i) \cdot z}{z_{\text{scale}}}. \quad (12)$$

Here, z is an integer from the lower z_{bit} of b_{frac} unused in LUT lookup, and $z_{\text{scale}} = 2^{z_{\text{bit}}}$ is the normalization factor, handled by a right shift. Ultimately, the proposed method attains high approximation accuracy while minimizing LUT size through simple residual-based linear interpolation.

B. LayerNorm Approximation

1) *Grouped-mean Calculation:* LayerNorm in Eq. (2) begins by computing the mean of the input vector x_t through summing all d elements and dividing by d , a high-precision operation that adds significant hardware overhead. To address this limitation, we propose a grouped-mean approach that reduces the denominator d , thereby lowering the required bit-precision. Furthermore, $\frac{1}{d}$ is transformed into a reciprocal PoT

form and implemented using shift-based operations. First, the d -dimensional data is divided into groups of g elements to compute the grouped mean. When g is set to a PoT value, the grouped-mean operation can be performed using shift operations as follows:

$$\mu_{\text{mean}} = \frac{1}{m} \left\{ \sum_{j=1}^m \left(\sum_{i=1}^g x_{j,i} \right) \gg \log_2(g) \right\} \quad (13)$$

We set $g = 64$, and compute $m = \lfloor \frac{d}{g} \rfloor$. For example, when $d = 1024$, m becomes 16, and $\frac{1}{m}$ can be implemented simply with a 4-bit shift. In contrast, when $d = 768$, m becomes 12, which cannot be implemented using shifts. Thus, when $d = 768$, $\frac{1}{m}$ is decomposed into four PoT terms \mathbf{m}_k to preserve precision while minimizing multiplication operations as:

$$\tilde{\mu}_{\text{mean}} \approx \sum_{k=1}^4 \left\{ \sum_{j=1}^m \left(\sum_{i=1}^g x_{j,i} \right) \gg \log_2(g) \right\} \gg \mathbf{m}_k \quad (14)$$

Overall, the proposed method enables the mean computation to be handled solely by shifts within the inference flow of a quantized network, while eliminating high-precision division in FPGA-based LayerNorm implementation, thereby significantly improving resource efficiency.

2) *Hardware-Friendly Reciprocal Square Root Approximation:* The RSQRT term of LayerNorm can be expressed through the logarithmic identity [28] as follows:

$$\frac{1}{\sqrt{x}} = x^{-1/2} = 2^{-\frac{1}{2} \log_2 x}. \quad (15)$$

where $\log_2 x$ can be decomposed into integer and fractional parts. Let $\log_2 x = \rho + v$, where $\rho = \lfloor \log_2 x \rfloor$ represents the integer part, and $v \in [0, 1)$ represents the fractional part. The integer part ρ is implemented using leading-one detector (LOD) logic [41]. Therefore, the value to be computed is $\frac{1}{2}(\rho + v)$. At this point, if the integer operation $\frac{\rho}{2}$ is implemented simply by a shift operation, the fractional bits are lost when ρ

is odd. Accordingly, we decompose ρ into the form $\rho = 2e + o$. That is, by expressing ρ as the even quotient $e = \lfloor \rho/2 \rfloor$ and the residual bit $o \in [0, 1)$, e can be easily obtained by the operation $\rho \gg 1$, while o preserves the remaining bit when ρ is odd. Subsequently, o is processed together with the fractional part v of $\log_2 x$ to preserve fractional precision. Consequently, RSQRT can be formulated as follows:

$$2^{-\frac{1}{2} \log_2 x} \approx 2^{-\frac{1}{2}(\rho+v)} = 2^{-e} \cdot 2^{-\frac{o+v}{2}}. \quad (16)$$

At this point, 2^{-e} can be handled by a simple shift, enabling hardware-friendly implementation. However, the residual term $2^{-\frac{o+v}{2}}$ must be computed. While o is easily obtained from $\rho - (e \ll 1)$, v is the fractional part of $\log_2(x)$ and thus cannot be directly computed, so it is obtained through linear approximation as follows:

$$v \approx \frac{r-1}{\ln 2}, \quad \text{where } r = x \gg \rho \quad (17)$$

where $\frac{1}{\ln 2}$ can be stored in fixed-point form as a reciprocal, enabling implementation through simple multiplication. Consequently, the LayerNorm approximation through this process can finally be expressed as follows:

$$\text{LayerNorm}(x) \approx \{(x - \mu_{mean}) \gg e\} \cdot 2^{-\frac{1}{2}(o+v)} \cdot \gamma + \delta \quad (18)$$

Although $o + v$ can be computed using fixed-point operations, $2^{-\frac{o+v}{2}}$ cannot be performed via shifts. Thus, we employ a linear interpolation-based LUT approximation, similar to that used for GELU. This approach preserves the fractional precision of fixed-point LayerNorm data with minimal logic, maintaining high approximation accuracy.

C. Hardware Implementation

We implemented the proposed approximation methods as hardware modules on FPGAs. Fig. 2 shows the overall *HI-APP* architecture. First, in the GELU approximation module, PWL slopes are decomposed with MPD and stored in a LUT (1). The breakpoints for each LUT segment are quantized into integers and stored together with the MPD values. For input, the comparator determines the LUT index from the breakpoints, and the corresponding MPD values are loaded to perform operations in the shift-add module (2). Next, the integer and fractional parts are separated with shift and bit-masking operations (3, 4). The upper 8-bit fraction is then compared with LUT breakpoints (5) to determine the index, and values v_0 and v_1 are loaded into the interpolation module (6) for fractional linear interpolation. Finally, the integer part is processed with shifts (7), and the output is generated by the dequantizer. In the LayerNorm approximation module, the mean is first computed by the grouped-mean unit (1). The variance is then derived in the same way, and the integer part ρ is obtained with the LOD module (2). Next, e and o are obtained through (3–4), and the fractional part r is computed using (5). The value v is derived by linear approximation, and the comparator determines the index by comparing LUT (6) breakpoints. Finally, the γ and δ operations are applied to generate the final output (7).

TABLE I: Comparison of approximation errors across different NLFs using various approximation methods.

Method	GELU		Swish		LayerNorm	
	MSE	MAE	MSE	MAE	MSE	MAE
QUNF [30]	1.68e-4	8.62e-3	1.65e-4	8.62e-3	5.96e-3	4.19e-2
PEANO-ViT [28]	2.78e-4	1.38e-2	-	-	6.95e-3	4.07e-2
I-ViT [26]	1.57e-3	1.91e-2	5.74e-3	3.47e-2	2.08e-2	5.16e-2
GQA-LUT [25]	1.85e-4	1.12e-2	-	-	2.46e-3	2.49e-2
HI-APP	5.46e-5	6.33e-3	8.58e-5	6.33e-3	1.54e-3	2.11e-2

In summary, *HI-APP* offers key advantages over existing NLF approximations. First, most multiplications are replaced with PoT-based shift-add structures, greatly reducing DSP usage. Second, unlike conventional fully-integer designs that lose accuracy by truncating or omitting the fractional part, our approach uses lightweight LUT-based interpolation to handle fractional computations efficiently while preserving accuracy. Consequently, *HI-APP* mitigates both resource overhead and accuracy loss, achieving a balanced trade-off between hardware efficiency and model performance.

IV. EXPERIMENTAL RESULTS

A. Experimental Environments

We apply *HI-APP* to CLIP-ViTs and evaluated its classification accuracy on the ImageNet dataset [42]. The proposed GELU and LayerNorm approximations were combined with an existing Softmax approximation. For this, we adopt GQA-LUT [25] to ensure integer compatibility and accuracy in our evaluation. For hardware evaluation, we synthesized the approximation modules on a Xilinx Zynq Ultrascale+ XCZU9EG board with high-level synthesis (HLS) and compared resource usage against state-of-the-art (SOTA) approximation methods.

B. Evaluation

1) *Software Level Evaluation*: We first evaluate the approximation error of each NLF using mean squared error (MSE) and mean absolute error (MAE). Sample data were generated over the input range $[-4, 4]$ with an interval of 2^{-10} , and GQA-LUT [25], I-ViT [26], and *HI-APP* were evaluated in the integer domain after quantization. PEANO-ViT [28] and QUNF [30] were evaluated by converting inputs to fixed-point after dequantization, while LayerNorm experiments used a subset of ImageNet. For the Swish function, some methods were excluded due to unavailable LUT information, while *HI-APP* was implemented by adjusting the slope parameter. As shown in Table I, *HI-APP* achieved the lowest MSE and MAE across all NLFs, reducing errors for GELU and Swish by up to $30\times$ compared to prior methods, while maintaining low approximation error for LayerNorm. These results demonstrate that *HI-APP* provides superior approximation accuracy.

Next, in Table II, we evaluate ImageNet classification accuracy by applying the approximation methods to CLIP ViT-B/32 and ViT-L/14, using 8-bit symmetric quantization for both weights and activations. Subsequently, the NLF approximation methods are applied to measure accuracy. To ensure a fair comparison, we do not perform any additional training. On CLIP ViT-B/32 and ViT-L/14, our method incurred only 0.93% and 0.54% accuracy drops relative to the INT8 baseline,

TABLE II: Comparison of ImageNet Top-1 accuracy of CLIP-ViTs with different approximation methods.

Model	Method	Approximations	Accuracy(%)
CLIP ViT-B/32	Baseline	-	62.35
	QUNF [30]	-	55.93
	PEANO-ViT [28]	ALL	44.31
	GQA-LUT [25]	-	43.86
	HI-APP	GELU + LayerNorm	61.54
		ALL	61.42
CLIP ViT-L/14	Baseline	-	74.57
	QUNF [30]	-	69.92
	PEANO-ViT [28]	ALL	66.06
	GQA-LUT [25]	-	57.43
	HI-APP	GELU + LayerNorm	74.16
		ALL	74.03

TABLE III: Comparison of FPGA resource utilization with different approximation methods across NLFs

Functions	Method	DSP (Diff.)	LUT (Diff.)	FF (Diff.)
GELU	Baseline	247	17914	22368
	QUNF [30]	12 (-95.1%)	6569 (-63.3%)	1924 (-91.4%)
	PEANO-ViT [28]	16 (-93.5%)	2940 (-83.5%)	2951 (-86.1%)
	GQA-LUT [25]	15 (-93.9%)	4290 (-76.1%)	1313 (-94.1%)
	HI-APP	0 (-100%)	8722 (-51.3%)	2659 (-88.1%)
LayerNorm	Baseline	51	24609	29831
	QUNF [30]	46 (-9.8%)	7661 (-68.9%)	5720 (-80.8%)
	PEANO-ViT [28]	52 (+1.9%)	8157 (-66.8%)	8621 (-71.1%)
	GQA-LUT [25]	46 (-9.8%)	13985 (-43.2%)	5371 (-81.9%)
	HI-APP	42 (-17.6%)	7440 (-69.8%)	1193 (-96.0%)

whereas existing methods suffered losses ranging from 4.56% to 19.16%. These results show that *HI-APP* minimizes accuracy degradation and ensures stable performance.

2) *Hardware Level Evaluation*: Table III shows hardware resource utilization when synthesizing the approximation modules, with the level of parallelism fixed at 16. For GELU, the proposed method replaces all multiplications with shifts, eliminating the need for DSPs. Considering that DSP blocks are far more limited than LUTs and FFs, and their overuse severely hinders parallel scalability and flexibility on FPGAs [39], *HI-APP* markedly improves resource utilization efficiency over SOTA methods, despite *HI-APP* incurring a modest increase in LUT usage. It is also noteworthy that *HI-APP* offers superior accuracy retention over SOTA methods and avoids retraining overhead, making it a practical design choice. Meanwhile, the LayerNorm module also achieves the highest resource efficiency by computing the mean in groups to lower bit-precision requirements and processing most operations with PoT-based shifts. *HI-APP* reduces DSP usage by up to 17.6% and LUT and FF usage by 69.8% and 96.0%, respectively, representing the largest reductions among prior studies and improving the accuracy–resource trade-off.

C. Ablation Studies

1) *Clipping Range Analysis*: Table IV summarizes how PAE, CE, and overall GELU approximation error vary with the clipping range. Wider ranges enlarge PWL intervals and significantly increase PAE, while tighter ranges (e.g., [-3.3, 3.3]) reduce PAE; however, excessively narrow ranges cause a sharp increase in CE. For [-4, 4], CE remains small, but PAE is relatively elevated, resulting in a larger overall error. Consequently, we identify [-3.3, 3.3] as the optimal range under limited LUT entries, as it minimizes both PAE and CE.

TABLE IV: Comparison of PAE, CE, and overall GELU approximation error with respect to the input clipping range

Range	LUT Entry	MSE			LUT Entry	MSE		
		PAE	CE	GELU		PAE	CE	GELU
[-6;6]		9.05e-4	3.51e-17	9.31e-4		4.42e-4	3.51e-17	2.49e-4
[-5;5]		4.76e-4	3.82e-14	3.72e-4		2.84e-4	3.82e-14	9.15e-5
[-4;4]	6	2.70e-4	1.61e-8	1.06e-4	8	1.96e-4	1.61e-8	3.15e-5
[-3.3;3.3]		2.27e-4	1.02e-6	5.46e-5		1.80e-4	1.02e-6	2.23e-5
[-2;2]		7.32e-5	4.67e-4	4.05e-4		1.14e-4	4.67e-4	4.08e-4

TABLE V: ImageNet classification accuracy of CLIP ViT-L/14 with each NLF approximation method

Model	Method	GELU	LN	Accuracy(%)
CLIP ViT-L/14	Baseline	-	-	74.57
	QUNF [30]	✓	×	70.20
		×	✓	70.05
		✓	✓	69.97
	PEANO-ViT [28]	✓	×	69.34
		×	✓	72.98
		✓	✓	65.98
	GQA-LUT [25]	✓	×	59.81
		×	✓	74.02
		✓	✓	58.79
		✓	×	74.62
		×	✓	73.81
	✓	✓	74.16	

TABLE VI: Compatibility comparison between *HI-APP* and various Softmax approximation methods

Model	Softmax		
	QUNF [30]	GQA-LUT [25]	PEANO-ViT [28]
CLIP ViT-B/32	61.52	61.42	61.37
CLIP ViT-L/14	74.05	74.03	73.90

2) *Impact of NLF Approximations*: Table V shows that PEANO-ViT and GQA-LUT incur larger accuracy losses for GELU than LayerNorm, indicating GELU’s greater impact on overall performance. In contrast, *HI-APP* achieves a 0.05% gain over the INT8 baseline, demonstrating its ability to preserve accuracy while remaining hardware-efficient.

3) *Compatibility Analysis with Softmax Approximation*: We analyze the compatibility of the proposed GELU and LayerNorm approximations with prior Softmax methods. Specifically, we integrate QUNF, GQA-LUT, and Peano-ViT Softmax approximations into CLIP-ViT alongside *HI-APP* and compare ImageNet accuracy. As shown in Table VI, the proposed GELU and LayerNorm demonstrate strong compatibility with existing Softmax approaches. While QUNF yields the highest accuracy, we adopt GQA-LUT to ensure quantized network compatibility and enable a fully-integer inference pipeline.

V. CONCLUSION

We propose *HI-APP*, a hardware-friendly approximation framework for nonlinear functions in quantized CLIP-ViTs. For GELU, we derive an optimal clipping range and replace multiplications with PoT-based shifts to minimize DSP usage while preserving accuracy. For LayerNorm, we redesign mean computation and RSQRT using PoT-based integer approximations, removing high-precision division and lowering DSP reliance. Integrated with quantized GEMM, *HI-APP* yields only a 0.93% ImageNet accuracy drop. On FPGAs, it eliminates DSPs for GELU and cuts LUT and FF usage for LayerNorm by 69.8% and 96.0%, respectively. Overall, *HI-APP* enables efficient, retraining-free nonlinear approximation for CLIP-ViTs with strong accuracy and deployability.

REFERENCES

- [1] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *Advances in neural information processing systems*, vol. 36, pp. 34 892–34 916, 2023.
- [2] J.-B. Alayrac *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in neural information processing systems*, vol. 35, pp. 23 716–23 736, 2022.
- [3] M. Xiong, Z. Wen, Z. Gu, X. Liu, R. Zhang, H. Kang, J. Yang, J. Zhang, W. Li, C. He *et al.*, "Prune2drive: A plug-and-play framework for accelerating vision-language models in autonomous driving," *arXiv preprint arXiv:2508.13305*, 2025.
- [4] B. J. Kang, H. I. Lee, S. K. Yoon, Y. C. Kim, S. B. Jeong, H. Kim *et al.*, "A survey of fpga and asic designs for transformer inference acceleration and optimization," *Journal of Systems Architecture*, vol. 155, p. 103247, 2024.
- [5] R. Machupalli, M. Hossain, and M. Mandal, "Review of asic accelerators for deep neural network," *Microprocessors and Microsystems*, vol. 89, p. 104441, 2022.
- [6] M. Cho and Y. Kim, "Fpga-based convolutional neural network accelerator with resource-optimized approximate multiply-accumulate unit," *Electronics*, vol. 10, no. 22, p. 2859, 2021.
- [7] K. Li, S. Goyal, J. D. Semedo, and J. Z. Kolter, "Inference optimal vlms need fewer visual tokens and more parameters," in *The Thirteenth International Conference on Learning Representations*.
- [8] Y. Wen, Q. Cao, Q. Fu, S. Mehta, and M. Najibi, "Efficient vision-language models by summarizing visual tokens into compact registers," *arXiv preprint arXiv:2410.14072*, 2024.
- [9] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. Pmlr, 2021, pp. 8748–8763.
- [10] K. Li, S. Goyal, J. D. Semedo, and J. Z. Kolter, "Inference optimal vlms need fewer visual tokens and more parameters," in *The Thirteenth International Conference on Learning Representations*.
- [11] N. J. Kim, J. Lee, and H. Kim, "Hyq: Hardware-friendly post-training quantization for cnn-transformer hybrid networks," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24. International Joint Conferences on Artificial Intelligence Organization*, vol. 8, 2024, pp. 4291–4299.
- [12] C. Wang, Z. Wang, X. Xu, Y. Tang, J. Zhou, and J. Lu, "Q-vlm: Post-training quantization for large vision-language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 114 553–114 573, 2024.
- [13] D. Choi and H. Kim, "Gradq-vit: Robust and efficient gradient quantization for vision transformers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 15, 2025, pp. 16 019–16 027.
- [14] B. J. Kang, N. Kim, and H. Kim, "Lra-qvit: Integrating low-rank approximation and quantization for robust and efficient vision transformers," in *Forty-second International Conference on Machine Learning*.
- [15] X. Huang, Y.-L. Huang, and Z. Wen, "Sola: Leveraging soft activation sparsity and low-rank decomposition for large language model compression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 16, 2025, pp. 17 494–17 502.
- [16] H. Yu and J. Wu, "Compressing transformers: features are low-rank, but weights are not!" in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, 2023, pp. 11 007–11 015.
- [17] N. J. Kim and H. Kim, "Trunk pruning: Highly compatible channel pruning for convolutional neural networks without fine-tuning," *IEEE Transactions on Multimedia*, vol. 26, pp. 5588–5599, 2023.
- [18] K. Koo and H. Kim, "V-skip: Vectorized kernel-based structured kernel pruning for accelerating deep convolutional neural networks," *IEEE Access*, vol. 11, pp. 118 547–118 557, 2023.
- [19] N. J. Kim and H. Kim, "Repsgd: Channel pruning using reparameterization for accelerating convolutional neural networks," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2023, pp. 1–5.
- [20] Z. Yao, Z. Dong, Z. Zheng, A. Gholami, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. Mahoney *et al.*, "Hawq-v3: Dyadic neural network quantization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 875–11 886.
- [21] Y. Lin, T. Zhang, P. Sun, Z. Li, and S. Zhou, "Fq-vit: Post-training quantization for fully quantized vision transformer," *arXiv preprint arXiv:2111.13824*, 2021.
- [22] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-power computer vision*. Chapman and Hall/CRC, 2022, pp. 291–326.
- [23] Q. Zhang, J. Cao, Y. Zhang, S. Zhang, Q. Zhang, and D. Yu, "Fpga implementation of quantized convolutional neural networks," in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*. IEEE, 2019, pp. 1605–1610.
- [24] M. Tasci, A. Istanbulu, V. Tumen, and S. Kosunalp, "Fpga-qnn: quantized neural network hardware acceleration on fpgas," *Applied Sciences*, vol. 15, no. 2, p. 688, 2025.
- [25] P. Dong, Y. Tan, D. Zhang, T. Ni, X. Liu, Y. Liu, P. Luo, L. Liang, S.-Y. Liu, X. Huang *et al.*, "Genetic quantization-aware approximation for non-linear operations in transformers," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.
- [26] Z. Li and Q. Gu, "I-vit: Integer-only quantization for efficient vision transformer inference," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 065–17 075.
- [27] H. Nam, S. B. Seo, V. S. Mailthody, N. Michael, and L. Li, "I-bert: Inductive generalization of transformer to arbitrary context lengths," *arXiv preprint arXiv:2006.10220*, 2020.
- [28] M. E. Sadeghi, A. Fayyazi, S. Azizi, and M. Pedram, "Peano-vit: Power-efficient approximations of non-linearities in vision transformers," in *Proceedings of the 29th ACM/IEEE International Symposium on Low Power Electronics and Design*, 2024, pp. 1–6.
- [29] J. Kim, J. Lee, J. Choi, J. Han, and S. Lee, "Range-invariant approximation of non-linear operations for efficient bert fine-tuning," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.
- [30] H. Du, C. Wen, Z. Chen, L. Zhang, Q. Sun, Z. Yan, and C. Zhuo, "Algorithm-hardware co-design of a unified accelerator for non-linear functions in transformers," in *2025 Design, Automation & Test in Europe Conference (DATE)*. IEEE, 2025, pp. 1–7.
- [31] A. Böttcher and M. Kumm, "Multiplier design addressing area-delay trade-offs by using dsp and logic resources on fpgas," in *2024 IEEE 35th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2024, pp. 217–225.
- [32] 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)*. IEEE, 2021, pp. 208–220.
- [33] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," *Advances in neural information processing systems*, vol. 2, 1989.
- [34] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.
- [35] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
- [36] G. Islamoglu, M. Scherer, G. Paulin, T. Fischer, V. J. Jung, A. Garofalo, and L. Benini, "Ita: An energy-efficient attention and softmax accelerator for quantized transformers," in *2023 IEEE/ACM International Symposium on Low Power Electronics and Design*. IEEE, 2023, pp. 1–6.
- [37] N. A. Koca, A. T. Do, and C.-H. Chang, "Hardware-efficient softmax approximation for self-attention networks," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.
- [38] J. Yu, J. Park, S. Park, M. Kim, S. Lee, D. H. Lee, and J. Choi, "Nn-lut: Neural approximation of non-linear operations for efficient transformer inference," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 577–582.
- [39] Q. Guo, J. Wan, S. Xu, M. Li, and Y. Wang, "Hg-pipe: Vision transformer acceleration with hybrid-grained pipeline," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024, pp. 1–9.
- [40] Y. Li, X. Dong, and W. Wang, "Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks," *arXiv preprint arXiv:1909.13144*, 2019.
- [41] K. Kurnaraj and R. Seshasayanan, "Leading one detectors and leading one position detectors—an evolutionary design methodology," *Canadian journal of electrical and computer engineering*, vol. 36, no. 3, pp. 103–110, 2014.
- [42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.