Discriminate Weight Approximation for Efficient DSP Packing in LLM Accelerators

Jun Li, Yangyijian Liu, Chang-Wei Shi, Mingyang Li and Wu-Jun Li*
National Key Laboratory for Novel Software Technology, School of Computer Science, Nanjing University
{lijun, yyj.liu, shicw, limingyang}@smail.nju.edu.cn; liwujun@nju.edu.cn

***Abstract*—Digital signal processor (DSP) array has been widely used in deep learning accelerators. DSP packing is always used to reduce the chip area of DSP arrays. Weight approximation is typically adopted in DSP packing to further reduce chip area. However, existing methods use indiscriminate weight approximation (IWA), which will lead to significant accuracy decrease for large language models (LLMs) and high lookup table (LUT) consumption for LLM accelerators. In this paper, we propose a new method, called discriminate weight approximation (DWA), for efficient DSP packing in LLM accelerators. DWA can reduce LUT consumption while guaranteeing accuracy. Experimental results show that compared with methods without weight approximation, DWA achieves a 1.5 times reduction of chip area and an imperceptible accuracy decrease. Compared with existing IWA methods, DWA achieves the same chip area reduction performance but reduces the LUT consumption by 2.6 to 4.6 times, which results in a five times reduction of deployment cost. Moreover, the accuracy of DWA is far better than that of existing IWA methods.**

***Index Terms*—LLM Accelerator, FPGA, Quantization, DSP packing**

I. Introduction

Deep learning models, especially large language models (LLMs) [1]–[4], have garnered substantial attention from academia and industry due to their significant success. To boost the inference efficiency of deep learning models, hardware accelerators have been more and more widely utilized. In practice, digital signal processor (DSP) array, which contains several DSP slices such as DSP48E2, occupies most of the accelerator's chip area and handles all computation workload [5], [6]. Hence, efficient utilization of DSP array is crucial for reducing the deployment cost of accelerators. DSP packing [7]–[19] can reduce the chip area and subsequently improve DSP array's utilization by integrating multiple multiplications into a single DSP slice.

Weight approximation [10], [13], which substitutes original weight of the model with the one requiring reduced bit width, is typically adopted in DSP packing to further reduce chip area. For example, NPA [13] is a representative DSP packing method with weight approximation. NPA can pack three multiplications with 8-bit precision and six multiplications with 4-bit precision into a single DSP slice by adopting weight approximation. NPA achieves a 1.5 times reduction of chip area, compared to methods without weight approximation adopted in current LLM accelerators such as ALLO [18] and Flightllm [19]. Although some more recent methods [9], [15], [16] without weight approximation can achieve the same chip area reduction performance as NPA in some cases, they focus on either convolution operators or the multiplication with specific precisions, thus have limited applicability. For example, Uint-Packing [16] introduces a method tailored for convolution operators, which cannot be easily adapted for LLM accelerators since LLMs rarely use convolutions. The method in [9] only targets at the multiplication with 4-bit precision, which makes it hard to integrate with other precisions.

*Wu-Jun Li is the corresponding author. This work is supported by NSFC Project (No.12326615), the Key Major Project of the Pengcheng Laboratory (No.PCL2024A06).

Although weight approximation can further reduce chip area in DSP packing, existing methods like NPA [13] use indiscriminate weight approximation (IWA) which will lead to significant accuracy decrease for LLMs and high lookup table (LUT) consumption for LLM accelerators. More specifically, LLMs have significantly more weights than small-scale deep learning models which are the focus of existing IWA methods like NPA. Since IWA dispatches all weight scalars of the model to the approximation process indiscriminately in a non-selective manner, it typically approximates an excessive number of weights. This severely interferes with LLMs' original forward computation, which will significantly decrease the accuracy of LLMs. Moreover, LLMs usually have huge computation workload, making LLM accelerators prefer large DSP arrays to provide high computation capability. However, with the enlargement of DSP arrays in LLM accelerators, IWA consumes too many LUTs because IWA's non-selective manner requires the hardware to process each weight scalar indiscriminately. For example, NPA [13] is evaluated on a DSP array with a small size of 12×12. The LUT consumption scales up with the enlargement of DSP arrays. If the DSP array is enlarged to 128×128, 1139K LUTs are required for the IWA method. While the DSP slice consumption is acceptable for many mid-range FPGAs like AMD Kintex UltraScale [20], these FPGAs cannot accommodate the required LUT consumption, leading to increased deployment cost.

In this paper, we propose a new method, called discriminate weight approximation (DWA), for efficient DSP packing in LLM accelerators. The contributions of this paper are outlined as follows:

- To the best of our knowledge, this is the first work to study DSP packing with weight approximation in LLM accelerators.
- Unlike IWA which performs weight approximation indiscriminately, DWA performs weight approximation discriminately from two levels, including intra-DSP approximation and inter-DSP approximation.
- DWA can reduce LUT consumption while guaranteeing accuracy, compared with existing IWA methods.
- We comprehensively evaluate DWA across multiple DSP packing settings and quantization schemes. Experimental results show that compared with methods without weight approximation, DWA achieves a 1.5 times reduction of chip area and an imperceptible accuracy decrease. Compared with existing IWA methods, DWA achieves the same chip area reduction performance but reduces the LUT consumption by 2.6 to 4.6 times, which results in a five times reduction of deployment cost. Moreover, the accuracy of DWA is far better than that of existing IWA methods.

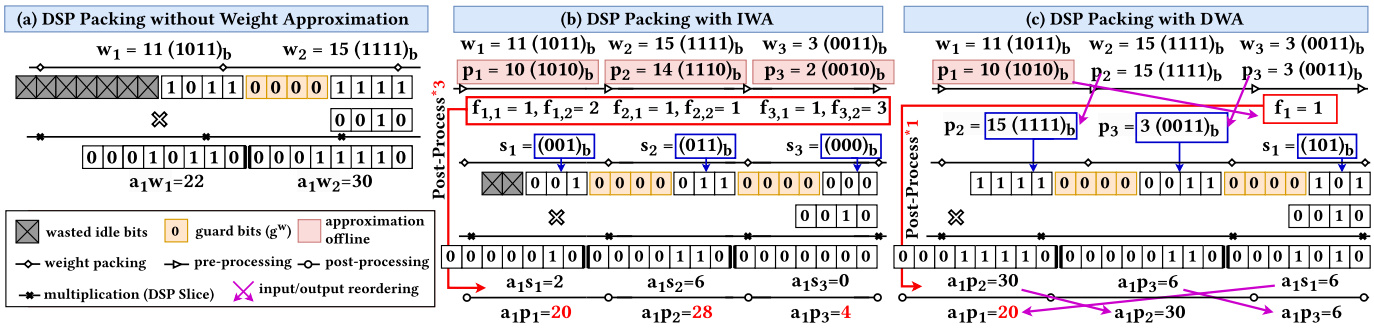

Fig. 1. Computation examples of (a) DSP packing without weight approximation, (b) DSP packing with IWA and (c) DSP packing with DWA (only intra-DSP approximation is used for easy illustration). For all examples, $D^a = 4$, $D^w = 19$, $b^a = 4$, $b^w = 4$. The weight and activation snippet are $\{w_1, w_2, w_3\} = \{11, 15, 3\}$ and $\{a_1\} = \{2\}$. The original individual scalar products without approximation are $\{a_1w_1, a_1w_2, a_1w_3\} = \{22, 30, 6\}$. With IWA, (b) packs three multiplications into a single DSP slice, while (a) can only pack two. However, the computation results of (b) differ from the original results for every individual scalar product. For computation results of (c), only a_1w_1 differs from its corresponding original result, which better preserves models' accuracy and consumes fewer LUTs.

TABLE I
DESCRIPTION OF TERMINOLOGIES IN THIS PAPER.

Object	Terminology	Description
hardware	DSP slice	the vendor-provided hardcore such as DSP48E2
	DSP unit	the wrapper module containing a single DSP slice and submodules required for DSP packing
	DSP array	the 2D array organized with multiple DSP units
activation/ weight data	matrix	input of linear operators in models
	tile	the 2D submatrix partitioned from matrix, serves as a DSP array's input
	snippet	the vector partitioned from each row of a tile, serves as a DSP unit's input

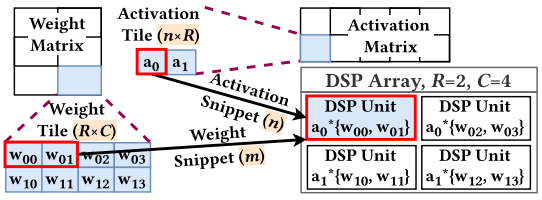

Fig. 2. Illustration of an example DSP array for Fig. 1 (a), where $R = 2$, $C = 4$, $m = 2$ and $n = 1$. The size of this DSP array is denoted as $R \times C = 2 \times 4$. Each DSP row has $C/m = 2$ DSP units.

II. PRELIMINARY KNOWLEDGE

A. Transformer-based LLMs

Transformer-based LLMs usually consist of multiple serially organized decoder layers. Each decoder layer consists of six linear operators, including the Q/K/V/Output linear operators in the attention block and two linear operators in the feed-forward network (FFN). These linear operators occupy the majority of LLMs' whole computation budget, reaching up to 97% for LLMs such as LLaMA-7B [1]. Hence, boosting the inference efficiency of linear operators is crucial for LLM accelerators, which is the focus of this paper.

In practice, LLMs are quantized into reduced bit widths for both activations and weights. For example, A8W4 (activations and weights are respectively quantized into 8-bit and 4-bit) [21] and A4W4 (activations and weights are both quantized into 4-bit) [22] quantization schemes are widely used in LLMs' deployment. With the reduced bit widths, a single DSP slice can compute multiple multiplications parallelly with the DSP packing.

B. DSP Packing

Table I lists the description of some key terminologies in this paper. In the accelerator design, the activation and weight matrix are uniformly partitioned into tiles, which serve as the input of the DSP array. Each row in the tile is further uniformly partitioned into snippets corresponding to a single DSP unit. To keep consistent with existing works [13], we denote the size of a DSP array as the shape of its input weight tile ($R \times C$). For example, for the DSP packing in Fig. 1 (a), Fig. 2 shows an example DSP array. This DSP array's size is denoted as 2×4, since its weight tile has the shape of $R \times C = 2 \times 4$.

Given the weight snippet ($\mathbf{w} = \{w_1, ..., w_m\} \in \mathbb{N}^m$) and the activation snippet ($\mathbf{a} = \{a_1, ..., a_n\} \in \mathbb{N}^n$), the DSP packing without weight approximation [7], [8], [23] can be formulated as follows:

$$\begin{aligned} &\langle a_1, g^a, a_2, g^a, ..., a_n\rangle \odot \langle w_1, g^w, w_2, g^w, ..., w_m\rangle \\ &= \langle a_1w_1, a_1w_2, ..., a_1w_m, a_2w_1, ..., a_nw_m\rangle, \end{aligned} \tag{1}$$

where "$\langle\cdots\rangle$" is the bit concatenation used to pack multiple integers into a binary code. "$\odot$" denotes integer multiplication performed by DSP slices. Let b^a and b^w denote the quantized bit widths for activation and weight scalar, respectively. To avoid bits overlapping in the outputs, guard bits g^w (typically b^a binary zeros) and g^a (typically $(mb^w + (m-1)b^a)$ binary zeros) are respectively inserted into weight and activation snippets during the packing. Moreover, each DSP slice has a pair of vendor-specified maximum bit widths, D^a for activation and D^w for weight. The packed activation snippet $\langle a_1, g^a, a_2, g^a, ..., a_n\rangle$ has the bit width of $(nb^a + (n-1)(mb^w + (m-1)b^a))$ that must not exceed D^a. The packed weight snippet $\langle w_1, g^w, w_2, g^w, ..., w_m\rangle$ requires the bit width of $(mb^w + (m-1)b^a)$ which should be smaller than D^w. Finally, individual scalar products such as a_1w_1 can be derived from the DSP slice's output.

Fig. 1 (a) gives an example of (1), where only two multiplications ($n = 1, m = 2$) can be packed into one DSP slice due to the restriction of D^w. Specifically, the weight snippet $\{w_1, w_2\}$ is packed into the binary code $(1011, 0000, 1111)_b$ with guard bits $(0000)_b$. Activation packing degrades to $\langle a_1\rangle$ when $n = 1$, which is skipped in the example. The 16-bit output of the DSP slice is divided into the two 8-bit individual scalar products a_1w_1 and a_1w_2. With this method, the 2×4 sized DSP array in Fig. 2 actually has $C/m = 2$ DSP units in each row and $R \times C/m = 4$ DSP units in total.

To further improve DSP packing, IWA [10], [13] has been employed to substitute the original weight of the model with the

approximated weight requiring reduced bit width. Specifically, IWA transforms the original $\mathbf{w}$ to the approximated weight snippet ($\mathbf{p} = \{p_1, ..., p_m\} \in \mathbb{N}^m$) with the following equation:

$$p_i = \begin{cases} w_i, & B(w_i) \leq t, \\ v_i, & B(w_i) > t, \end{cases} \tag{2}$$

for all $i \in \{1, ..., m\}$. t is the preset threshold smaller than b^w. $B()$ identifies the required bit width of w_i. When the scalar-level violation (i.e., $B(w_i) > t$) is triggered, w_i is approximated to v_i which requires smaller bit width than w_i. v_i is determined by the following equation:

$$v_i = \underset{\substack{u \in \{0, ..., 2^{b^w}-1\} \\ B(u) \leq t}}{\arg\min} D(u, w_i), \tag{3}$$

for all $i \in \{1, ..., m\}$. $D()$ calculates the distance between two binary codes. Since models' weights do not change during the deployment, (2) and (3) are usually performed offline. In this way, DSP units receive $\mathbf{p}$ rather than $\mathbf{w}$ during the on-the-fly computation.

The approximation precess in (2) and (4) not only affects models' accuracy but also determines hardware design of DSP array. First, IWA frequently triggers scalar-level violation in (2), forcing an excessive number of weight scalars to be approximated. This severely interferes with LLMs' original forward computation, thus significantly decreasing accuracy. Second, the definition of $B()$ directly determines DSP units' hardware design. For example, NPA [13] defines $B(w_i)$ as $(b^w - f_{i,1} - f_{i,2})$, where $f_{i,1}$ and $f_{i,2}$ are determined greedily in the following equation:

$$s_i = ((w_i / 2^{f_{i,1}}) - 1) / 2^{f_{i,2}}. \tag{4}$$

s_i is the binary code transferred to the DSP slice. For example, for $w_i = 3(0011)_b$, $f_{i,1}$ can only be set to 0 to ensure w_i can be divided by $2^{f_{i,1}}$. Then, $f_{i,2}$ is set to 1 since $(w_i/2^{f_{i,1}}) - 1 = 2(0010)_b$ and $s_i = (001)_b$. This definition requires three extra operations compared to (1). The first is to compute auxiliary factors s_i, $f_{i,1}$ and $f_{i,2}$ for each p_i as the pre-processing. The second is to replace w_i in (1) with the s_i during weight packing. The third is the post-processing, which follows the reverse form of the first operation, generating individual scalar products with $f_{i,1}$, $f_{i,2}$ and the DSP slice's output. The second operation does not consume resources in the implementation, while the first and third operations require extra LUTs. The non-selective manner of IWA requires each DSP unit in the DSP array to include m pairs of pre-processing and post-processing modules. The corresponding LUT consumption is unaffordable for large DSP arrays which LLMs prefer.

Fig. 1 (b) gives an example of the DSP packing with IWA, where three multiplications are packed into one DSP slice due to the effectiveness of weight approximation. With t in (2) set to 2, all three weight scalars trigger the scalar-level violation. They are approximated to $p_1 = 10(1010)_b$, $p_2 = 14(1110)_b$ and $p_3 = 2(0010)_b$, respectively. During the on-the-fly computation, for each p_i, the pre-processing firstly computes s_i, $f_{i,1}$ and $f_{i,2}$. After packing $\{s_1, s_2, s_3\}$, the 21-bit output of the DSP slice can be divided into three 7-bit integers $\{a_1 s_1, a_1 s_2, a_1 s_3\}$. With corresponding $f_{i,1}$ and $f_{i,2}$, the final individual scalar products $\{a_1 p_1, a_1 p_2, a_1 p_3\}$ are produced through the reverse form of (4). Each scalar product differs from the original results $\{a_1 w_1, a_1 w_2, a_1 w_3\}$, resulting in decrease of models' accuracy.

III. Discriminate Weight Approximation

As stated in Section II, IWA methods have two issues: significant accuracy decrease and high LUT consumption. Our proposed DWA addresses these issues from two levels: intra-DSP approximation and inter-DSP approximation. Intra-DSP approximation focuses on optimization within a single DSP unit, while inter-DSP approximation focuses on optimization among different DSP units. Specifically, with a snippet-level violation, intra-DSP approximation discriminately selects a subset of scalars from the weight snippet of a DSP unit for approximation. This reduces the number of approximated weight scalars and the required number of pre-processing and post-processing modules. To guarantee the model's accuracy after approximation, inter-DSP approximation discriminately selects a subset of DSP units for approximation and allows the unselected ones to perform computation without approximation.

Algorithm 1 Intra-DSP Approximation

1: **Input**: The quantized weight matrices of LLM $\{W_1, ..., W_L\}$.
2: **Output**: Approximated weight matrices $\{P_1, ..., P_L\}$.
3: **for** $l \in \{1, ..., L\}$ **do**
4: Reshape $W_l \in \mathbb{N}^{M \times K}$ to $\widetilde{W}_l \in \mathbb{N}^{(\frac{M}{R} \cdot \frac{K}{m}) \times R \times m}$;
5: **for** $t \in \{1, ..., (\frac{M}{R} \cdot \frac{K}{m})\}$ **do**
6: **for** $r \in \{1, ..., R\}$ **do**
7: $\{w_1, ..., w_m\} = \widetilde{W}_l[t, r, :]$; // weight snippet.
8: $B^w = \sum_{i=1}^{m} B^*(w_i) + (m-1)b^a$;
9: **if** $B^w > D^w$ **then** // snippet-level violation.
10: $G = (B^w - D^w)/(b^w - t)$; $idx = -1$;
11: **for** $g \in \{1, ..., G\}$ **do**
12: $idx = \min(i)$, s.t. $(B^*(w_i) > t)$ & $(i > idx)$;
13: Approximate w_{idx} with (3);
14: **end for**
15: **end for**
16: **end for**
17: Reversely reshape $\widetilde{W}_l$ to the shape of W_l;
18: $P_l = \widetilde{W}_l$;
19: **end for**

A. Intra-DSP Approximation

Given an LLM, which is represented as a list of weight matrices $\{W_1, ..., W_L\}$, the algorithm of intra-DSP approximation is listed in Algorithm 1. The algorithm is roughly similar to IWA but differs in two aspects. First, the trigger condition for the approximation process is altered from the scalar-level violation in (2) to the snippet-level violation (line 8 in Algorithm 1). Specifically, the algorithm calculates the required bit width B^w for each weight snippet. The computation of B^w follows the bit width calculation of packed weight snippet in (1), in which the function $B^*()$ identifies the required bit width of w^i. Here, we define $B^*(w^i) = b^w - f_i$, where f_i is determined greedily in $\hat{s}_i = w_i / 2^{f_i}$. Second, the method discriminately selects a subset of scalars from the snippet for approximation. Specifically, if the snippet-level violation is triggered (line 9), G scalars are selected for approximation (line 10 $\sim$ line 14). The scalar selection is based on the scalar-level violation in (2), where t is defined as $(b^w - 1)$. An ascending order determines the indexes of these selected scalars. We use Bray-Curtis distance as the distance calculation $D()$ in (3), similar to the existing work [13].

Intra-DSP approximation with snippet-level violation preserves model accuracy better than IWA. Specifically, vector-based weight quantization [21], [24]–[26] is commonly employed in LLMs' quantization, which decomposes weight matrices into small vectors (e.g., 128 scalars) for individual uniform quantization to maintain accuracy. These small vectors enhance the uniform distribution of various quantized integer values. Since different integer values require different bit widths, their uniform distribution can effectively suppress snippet-level violations, drastically reducing the number of approximated

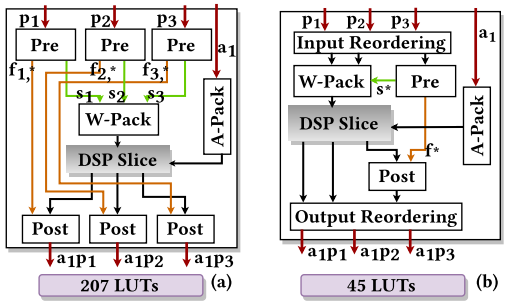

Fig. 3. DSP unit design illustration for A8W4 DSP packing of IWA (a) and intra-DSP approximation (b). Module 'Pre' and 'Post' refer to pre-processing and post-processing modules. Module 'W-Pack' and 'A-Pack' refer to weight and activation packing modules. Each pre-processing module outputs one s_i (green wire) and $f_{i,*}$ (orange wire). The green wires are aggregated in the weight packing module, and orange wires are transferred to post-processing modules. The signals do not interfere with each other at the intersection point. Since the approximation process is performed offline, DSP units receive $\{p_1, p_2, p_3\}$ rather than $\{w_1, w_2, w_3\}$ during the on-the-fly computation.

weights. This enables the intra-DSP approximation to interfere less with the LLMs' original forward computation, contributing to superior accuracy preservation.

Fig. 1 (c) gives an example of DSP packing with intra-DSP approximation. The snippet $\{w_1, w_2, w_3\}$ triggers the snippet-level violation. Unlike IWA, which indiscriminately approximates all weight scalars, our method discriminately selects w_1 for approximation and routes it to the pre-processing. Thus, among the three individual scalar products, only $a_1 w_1$ differs from its corresponding original result, which better preserves accuracy.

Intra-DSP approximation with snippet-level violation requires fewer pre-processing and post-processing modules than IWA, thereby reducing the LUT consumption. For intra-DSP approximation, the required number of pre-processing and post-processing modules within a specific DSP unit is the maximum value of G across all weight snippets mapped to this DSP unit. The consumed LUTs will be reduced if $G < m$, which is fulfilled in practice. Given m and set t to $(b^w - 1)$, $max(G) = (mb^w + (m-1)b^a - D^w)$. Take A8W4 quantization ($b^a = 8$, $b^w = 4$) as an example. Existing work [13] can enlarge the m to 3 for DSP48E2 slice [27], where $n = 1$ and $D^w = 27$. In intra-DSP approximation, each DSP unit requires $max(G) = 1$ pair of pre-processing and post-processing modules, only 33% of that in IWA.

Fig. 3 illustrates the hardware design of a DSP unit for (a) IWA and (b) intra-DSP approximation. As observed, our DSP unit only requires one pair of pre-processing and post-processing modules, while IWA requires three pairs. Moreover, our design requires the input reordering before the pre-processing, and the output reordering behind the post-processing. The reordering module's LUT consumption is determined by m, a relatively small number in DSP packing, restricting the reordering modules' LUT consumption. In practice, for A8W4 quantization, IWA consumes 207 LUTs for each DSP unit, while intra-DSP approximation of DWA only consumes 45 LUTs.

B. Inter-DSP Approximation

Inter-DSP approximation consists of three steps: inter-DSP remapping, accuracy-guaranteed search and design of DSP units without approximation.

1) ***Inter-DSP Remapping****:* All existing DSP packing methods [7], [8], [10], [12]–[14] map weight snippets to DSP units with a rigid mapping scheme. The scheme forces all weight snippets mapped to a specific DSP unit to adopt the same approximation choice despite their different approximation requirements. Inter-DSP remapping addresses this problem by mapping weight snippets to DSP units based on their individualized approximation requirements.

Algorithm 2 Offline Process of Inter-DSP Remapping

1: **Input**: The quantized weight matrices of LLM $\{W_1, ..., W_L\}$.
2: **Output**: Remapped weight matrices $\{E_1, ..., E_L\}$; Routing signals ($Rout$).
3: $Rout = \varnothing$;
4: **for** $l \in \{1, ..., L\}$ **do**
5: Reshape $W_l \in \mathbb{N}^{M \times K}$ to $\widetilde{W}_l \in \mathbb{N}^{(\frac{M}{R} \cdot \frac{K}{C}) \times R \times C}$;
6: **for** $r \in \{1, ..., (\frac{M}{R} \cdot \frac{K}{C})\}$ **do**
7: $Tile = \widetilde{W}_l[r, :, :]$; // weight tile.
8: **cnt** = **SnippetViolationsPerRow**($Tile$);
9: SortRowIdx = **AscendSort**($Tile$, key=**cnt**);
10: $\widetilde{W}_l$[r, :, :] = Tile[SortRowIdx, :];
11: $Rout$ = $Rout \cup$ { **RoutSig**(SortRowIdx) };
12: **end for**
13: Reversely reshape $\widetilde{W}_l$ to the shape of W_l;
14: $E_l = \widetilde{W}_l$;
15: **end for**

However, the naive mapping of weight snippets based on individualized approximation requirements introduces unaffordable routing complexity for activation snippets. In accelerator design, DSP arrays typically compute the dot product of an activation tile and a weight tile per cycle. For linear operators, a single activation tile interacts with multiple weight tiles due to column traversal of the weight matrix. Weight snippets within these tiles exhibit varying distributions of approximation requirements, necessitating distinct mapping schemes for each weight tile. While the mapping between weight snippets and DSP units can be preconfigured through offline reorganization, activation snippets must dynamically align with their corresponding DSP unit positions during each cycle, adhering to the unique mapping scheme of the current weight tile. Since every weight snippet is likely to be mapped to any position of DSP units within the array, the routing complexity of the activation snippets is unaffordable $\mathcal{O}((RC/m)!)$. Hence, we perform the inter-DSP remapping from the DSP row granularity. This granularity reduces the routing complexity to $\mathcal{O}(R!)$ with the naive routing network, and $\mathcal{O}(Rlog_2R)$ with the Benes [28] which is a well-known rearrangeable non-blocking routing network with high efficiency.

Based on the DSP row granularity, we first reorganize the weight snippets within individual weight tiles offline with Algorithm 2. The algorithm traversals each weight tile partitioned from each weight matrix in the model. For a specific weight tile, the function **SnippetViolationsPerRow**() is used to count the number (**cnt**) of snippets that trigger the snippet-level violation (formulated in line 8 of Algorithm 1) for each weight row. The **cnt** provides the reference for the approximation requirement. The algorithm then sorts weight rows in the tile according to the counting results. Finally, weight rows are reorganized according to the sorting row index.

Second, each snippet of the activation tile is routed to corresponding DSP units on-the-fly with a Benes network. The routing signals are determined by Algorithm 2 before the computation. Specifically, they are calculated in **RoutSig**(), which can be implemented with many existing methods [29]. According to the definition of Benes networks, these routing signals require extra $(Rlog_2R - R/2)$ bits of memory for each weight tile. The relative extra memory overhead for routing signals is $X = (Rlog_2R - R/2)/(RCb^w)$. X is imperceptible for users. For example, for a 128×128 DSP array with the 4-bit

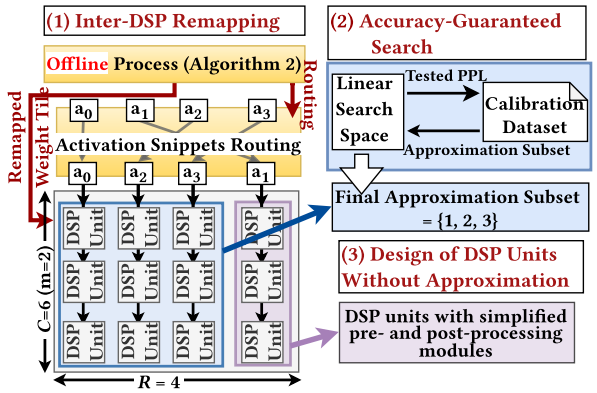

Fig. 4. Overview of our inter-DSP approximation. The DSP array has the size of 4×6 ($R = 4, C = 6, n = 1, m = 2$), including $RC/m = 12$ DSP units. Different colors represent different steps. The DSP array is transposed for illustration, which means that a vertical column of DSP units in the figure represents a DSP row in the text.

weight quantization, X equals 0.88%, which can almost be ignored. X has a complexity of $\mathcal{O}((\log_2 R)/C)$, which scales logarithmically with the DSP array size. Thus the proposed inter-DSP remapping is friendly with large DSP arrays. Fig. 4 (1) illustrates an example, where the activation tile has four snippets and each snippet has one scalar ($n = 1$, $R = 4$). The four activation snippets $\{a_0, a_1, a_2, a_3\}$ is reordered to $\{a_0, a_2, a_3, a_1\}$, under the guidance of routing signals.

*2) **Accuracy-Guaranteed Search**:* Based on the intra-DSP approximation and inter-DSP remapping, the accuracy-guaranteed search is to select a subset of DSP units[1] in DSP array to approximate while guaranteeing the model's accuracy after approximation.

However, we are still faced with two issues. One issue is the absence of a verifier for the model's accuracy with a specific approximation subset. This can be addressed by using the calibration-based verifier, frequently employed in current LLM quantization schemes [21], [22], [24]. Specifically, the accuracy with a specific approximation subset is tested on a preset calibration dataset. It provides the reference for the search process to determine the optimal approximation subset. In practice, we use the same calibration dataset as the work in [24]. The other issue is the large search space. Following the DSP row granularity in inter-DSP remapping, the overall complexity of search space is unaffordable (2^R). To this end, we propose a search algorithm in linear complexity.

Algorithm 3 lists the search process. The first step of the algorithm is to profile the impact of every DSP row on accuracy. During profiling, DSP rows are progressively appended to the approximation subset, with the accuracy of each iteration being recorded (lines 4 to 10 in Algorithm 3). The impact of approximation on the accuracy of a DSP row is defined as the accuracy difference between the row's first approximated iteration and its previous iteration (line 9). The function **IntraApx**() is used to approximate weights with the given approximation subset ($\{1, ..., i\}$). Its process is similar to Algorithm 1, which replaces line 6 in Algorithm 1 "**for** $r \in \{1, ..., R\}$" with "**for** $r \in \{1, ..., i\}$". The function **TestAccuracy**() is used to test accuracy upon calibration dataset with the given list of weight matrices. The second step of the algorithm is to progressively set the DSP row to approximation mode until the accuracy reaches a preset accuracy threshold ($(1+\theta)ppl_0^1$). In practice, θ is set to 1%.

[1]This selected subset of DSP units is called the *approximation subset*.

Algorithm 3 Accuracy-Guaranteed Search

1: **Input**: The remapped weight matrices $\{E_1, ..., E_L\}$.
2: **Output**: Approximation subset (Z).
3: // STEP 1: Accuracy Impact Profile.
4: $F = 0^R$;
5: ppl_0^1 = **TestAccuracy**($\{E_1, ..., E_L\}$); // the original accuracy.
6: **for** $i \in \{1, ..., R\}$ **do**
7: $\{E'_1, ..., E'_L\}$ = **IntraApx**($\{1, ..., i\}, \{E_1, ..., E_L\}$);
8: ppl_i^1 = **TestAccuracy**($\{E'_1, ..., E'_L\}$);
9: $F_i = ppl_i^1 - ppl_{i-1}^1$;
10: **end for**
11: // STEP 2.
12: Idx = **DecendSort**(F);
13: **for** $i \in \{1, ..., R\}$ **do**
14: $\{E'_1, ..., E'_L\}$ = **IntraApx**(Idx[i:], $\{E_1, ..., E_L\}$);
15: ppl_i^2 = **TestAccuracy**($\{E'_1, ..., E'_L\}$);
16: **if** $ppl_i^2 \leq (1+\theta)ppl_0^1$ **then**
17: Z = Idx[i:]; **Return**;
18: **end for**

Algorithm 3 needs a maximum of $(2R)$ accuracy tests upon calibration datasets. It exhibits a linear complexity relative to R and thus is friendly for large DSP arrays. Fig. 4 (2) illustrates an example of this step. As observed, the algorithm exports the {1,2,3} as the final decision on the approximation subset, indicating that the first, second and third DSP rows (marked in blue) are configured with intra-DSP approximation. The fourth DSP row is configured to compute without weight approximation.

*3) **Design of DSP Units without Approximation**:* For those DSP units configured to compute without weight approximation, DWA simplifies the pre-processing in (4). The pre-processing module of (4) computes auxiliary factors s_i, $f_{i,1}$ and $f_{i,2}$ for each w_i. The corresponding multiplication between an activation scalar a_i and an original weight scalar w_i can be formulated as follows:

$$\begin{aligned} a_i w_i &= a_i(2^{f_{i,1}}(1 + 2^{f_{i,2}} s_i)) = 2^{f_{i,1}}(a_i + (a_i 2^{f_{i,2}} s_i)), \\ &= 2^{f_{i,1}}(a_i[f_{i,2} - 1 : 0] + 2^{f_{i,2}}(a_i[b^a - 1 : f_{i,2}] + a_i s_i)), \\ &= \underbrace{(a_i[f_{i,2} - 1 : 0]}_{Post-Process} + \underbrace{(a_i[b^a - 1 : f_{i,2}] + a_i s_i)}_{DSP\ Slice} \underbrace{>> f_{i,2}) >> f_{i,1}}_{Post-Process}. \end{aligned} \tag{5}$$

">>" is the right shift operator. "[ ]" is the bits selection operator. The equation shows two right shift operations and one integer addition for the post-processing. However, we observe that this method is unfriendly for practical implementation. Specifically, both $a_i[f_{i,2} - 1 : 0]$ and $a[b^a - 1 : f_{i,2}]$ have inconstant bit widths, since $f_{i,2}$ changes with the value of w_i. They must be allocated with their maximum bit widths, which are $(b^w - 1)$ and b^a, respectively.

Based on this observation, we directly remove the factor $f_{i,2}$ in (4) and (5), since $f_{i,2}$ is the major factor that complicates the implementation. The pre-processing is simplified to the following:

$$s'_i = ((w_i / 2^{f_{i,1}}) - 1)/2, \tag{6}$$

where $f_{i,1} \in [0, b^w]$. The corresponding multiplication process is formulated as follows:

$$\begin{aligned} a_i w_i &= a_i(2^{f_{i,1}}(1 + s'_i)), \\ &= \underbrace{(a_i[0]}_{Post-Process} + \underbrace{(a_i[b^a - 1 : 1] + a_i s'_i)}_{DSP\ Slice} \underbrace{>> 1) >> f_{i,1}}_{Post-Process}, \end{aligned} \tag{7}$$

where each number has a constant bit width, friendly for the implementation. Moreover, the bit widths of all inputs ($a_i[b^a - 1 : 1]$, a_i and s'_i) of the DSP slice are the same as those of (4), so that

TABLE II
THE LUT CONSUMPTION AND PERPLEXITY COMPARISON BETWEEN OUR DWA AND NPA.

LLM	Packing	LUTs (K)		Perplexity							
				WikiText2				C4			
		NPA	DWA	Original FP16	Original Quantized	NPA	DWA	Original FP16	Original Quantized	NPA	DWA
OPT-6.7B	WOP-A8W4	1139	294	10.86	11.05	17193	11.12	11.74	12.67	7847	13.01
	WOP-A4W4	602	212		14.18	26634	14.36		16.21	12641	15.81
	WAP-A4W4	1572	610				13.89				15.95
LLaMA-7B	WOP-A8W4	1139	248	5.68	5.79	282492	5.80	7.09	7.24	289274	7.25
	WOP-A4W4	602	201		6.47	298431	6.47		8.17	319890	8.17
	WAP-A4W4	1572	565				6.53				8.25
LLaMA-2-7B	WOP-A8W4	1139	248	6.94	7.16	164357	7.17	8.98	9.29	122348	9.30
	WOP-A4W4	602	201		8.31	176944	8.28		10.92	129988	10.96
	WAP-A4W4	1572	607				8.45				11.02
LLaMA-3-8B	WOP-A8W4	1139	248	7.21	7.79	1744868	7.81	10.35	11.46	2090264	11.50
	WOP-A4W4	602	201		11.83	1991260	11.85		17.41	2363273	17.42
	WAP-A4W4	1572	587				11.97				17.57

the DSP slice's utilization is not affected. This DSP unit's design is similar to that in Fig. 3 (a) but has simplified pre-processing and post-processing modules, which consumes less LUTs.

IV. EVALUATION

This section validates the effectiveness of our DWA in reducing LUT consumption and preserving models' accuracy.

A. Experimental Setup

Hardware modules within DWA are implemented with Verilog HDL, and code synthesis is performed by the Vivado 2022.1 toolkit. All synthesis strategies are set to their default parameters. Our major evaluation platform is KU115 [20] with 660K LUTs and 5520 DSP48E2 slices. KU115 is a representative FPGA of the AMD Kintex UltraScale series, the major product line of mid-range FPGAs.

Four mainstream LLMs (OPT-6.7B [2], LLaMA-7B [1], LLaMA-2-7B [3] and LLaMA-3-8B [4]) are selected as the benchmark models. All accuracy tests in this section are based on the state-of-the-art (SOTA) LLM quantization scheme in [21]. The accuracy is reported in perplexity, where a lower value is better than a higher value, and tested on two commonly used datasets for LLMs (WikiText2 [30] and C4 [31]). We combine DWA with A8W4 and A4W4, the most commonly used quantization settings in the LLM's deployment [21], [24]. The specific quantization scheme is not the focus of this paper. We directly use the off-the-shelf vector-based quantization to quantize activation and weight. Both A8W4 and A4W4 use a weight quantization vector size of 128, while their activation quantization vector sizes are the embedding dimension (A8W4) and one-quarter of it (A4W4). It should be noted that DWA is orthogonal to the specific quantization scheme since we do not constrain the quantization scheme in each step of DWA.

According to the difference of activation packing, the DSP packing schemes can be categorized into weight-only packing (WOP) [8] where $n = 1$ and $m > 1$, and weight-activation packing (WAP) [7] where $n > 1$ and $m > 1$. We comprehensively evaluate our DWA across all the two DSP packing schemes. Combined with the quantization settings, WOP can be used for both A8W4 and A4W4 (denoted as WOP-A8W4 and WOP-A4W4), while WAP can only be used for A4W4 (denoted as WAP-A4W4).

B. Comparison with SOTA Methods

To the best of our knowledge, all existing LLM accelerators, such as [18], [19], adopt the DSP packing methods without weight approximation. Specifically, ALLO [18] and FlightLLM [19] can only pack two multiplications ($n = 1, m = 2$) for A8W4 quantization [8]. Compared with these methods, our DWA can pack three multiplications ($n = 1, m = 3$) into a single DSP slice, providing a 1.5 times reduction in chip area without speed decrease.

TABLE III
LUT CONSUMPTION COMPARISON OF A SINGLE DSP UNIT.

Setting	Design	Pre	Post	Total
WOP-A8W4	NPA	39	168	207
	DSP-o	12	57	69
	DSP-w	11	34	45
WOP-A4W4	NAP	36	111	147
	DSP-o	16	44	60
	DSP-w	19	30	49
WAP-A4W4	NPA	36	156	192
	DSP-o	8	66	74
	DSP-w	8	61	69

[1] DSP-o is the DSP unit without approximation (Section III-B3). DSP-w is the DSP unit with intra-DSP approximation (Section III-A).

Furthermore, we compare DWA with NPA [13], the SOTA method to improve DSP packing with weight approximation but tailored for small-scale deep learning models. With a 128×128 DSP array, Table II compares the LUT consumption between the two methods for four LLMs across three packing settings. Specifically, DWA's performance in DSP packing is equivalent to that of NPA, where a single DSP slice can pack 3, 4 and 6 multiplications for WOP-A8W4 ($n = 1, m = 3$), WOP-A4W4 ($n = 1, m = 4$) and WAP-A4W4 ($n = 3, m = 2$), respectively. This indicates that DWA's performance in chip area reduction is equivalent to that of NPA. With the same performance in chip area reduction, DWA significantly reduces LUT consumption by 2.6 to 4.6 times compared to NPA. In practice, the decreased LUT consumption directly contributes to the reduction of the deployment cost. For example, for OPT-6.7B with WOP-A8W4, our DWA requires ($R\lceil C/m \rceil = 128 \times \lceil 128/3 \rceil = 5504$) DSP slices and 294K LUTs, which is less than KU115's 5520 DSP slices and 660K LUTs. NPA consumes the same number (5504) of DSP slices as DWA, but it requires 1139K LUTs, exceeding KU115's 660K LUTs. To avoid scaling down the DSP array and sacrificing speed, NPA must use the higher-end FPGA with more LUTs, such as VU9P [32] with 1200K LUTs, the cost of which is roughly 5 times the commercial cost of KU115. This means that DWA achieves a five times reduction of deployment cost, compared with NPA.

Table III lists the LUT consumption of a single DSP unit designed with NPA, DSP-w (DSP unit with intra-DSP approximation in DWA)

TABLE IV
THE LUT CONSUMPTION (K) OF MODULES IN THE DSP ARRAY.

LLM	Settings	Activation Routing	DSP-w	DSP-o
OPT-6.7B	WOP-A8W4	6	170	119
	WOP-A4W4	2	157	54
	WAP-A4W4	7	35	568
LLaMA-7B	WOP-A8W4	0	248	0
	WOP-A4W4	0	201	0
	WAP-A4W4	0	565	0
LLaMA-2-7B	WOP-A8W4	0	248	0
	WOP-A4W4	0	201	0
	WAP-A4W4	7	71	530
LLaMA-3-8B	WOP-A8W4	0	248	0
	WOP-A4W4	0	201	0
	WAP-A4W4	7	353	227

and DSP-o (DSP unit without approximation in DWA). Input and output reordering modules are respectively fused into pre-processing and post-processing modules during DSP-w's implementation. Due to reduced pre-processing and post-processing modules, DSP-w always consumes the fewest LUTs across three packing settings. DSP-o has the same number of pre-processing and post-processing modules as NPA. However, it consumes less LUTs than NPA due to simplified pre-processing and post-processing modules. The reduced LUT consumption of DSP-w and DSP-o contributes to the overall reduced LUT consumption of DWA listed in Table II.

Table IV reports modules' LUT consumption (activation snippets routing, DSP-w and DSP-o) inside the DSP array with DWA. We can find that activation snippets routing modules consume only a small ratio of total LUTs, ranging from 0 to 1.87%. This is consistent with our discussion in Section III-B1, which states that the activation snippets routing module is friendly with large DSP arrays. Moreover, for different LLMs and packing settings, the varying sensitivity to weight approximation results in the varying distribution of LUT consumption across modules. For example, LLaMA-7B across three packing settings demonstrates low sensitivity to weight approximation. This enables DWA to configure all DSP units to compute with intra-DSP approximation, which mostly reduces the LUT consumption. In this case, the inter-DSP approximation is omitted. Thus, the activation snippets routing module and DSP-o are not required. Conversely, OPT-6.7B is more sensitive to weight approximation, forcing DWA to configure parts of DSP units to compute without approximation to preserve accuracy.

Table II also lists the perplexity of DWA and the IWA method. Please note that the perplexity gap between original FP16 models and their quantized counterparts (denoted as "original quantized") is not the focus of this paper. DWA is orthogonal to specific quantization schemes and mainly focuses on minimizing the perplexity gap between models approximated with DWA and "original quantized" models. Compared with the original quantized perplexity, DWA achieves an imperceptible fluctuation (relatively -2.5% to 2.7%) in perplexity. The negative fluctuation means that DWA's perplexity is lower (better) than the original quantized perplexity. Conversely, the perplexity of models approximated with IWA is significantly higher (worse) than the original quantized perplexity, increasing by several orders of magnitude, which is unacceptable. The accuracy gap between the two methods can be explained by the difference in approximated weight numbers, as illustrated in Fig. 5. Compared to IWA, DWA significantly reduces the approximated weight numbers by $9 \sim 372$ times. This reduced number enables DWA to deviate less from the LLMs' original forward computation and achieves 56999 times better accuracy than IWA on average.

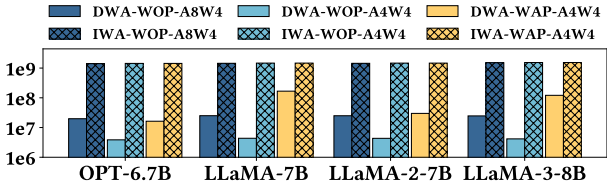

Fig. 5. The approximated weight numbers of DWA and IWA.

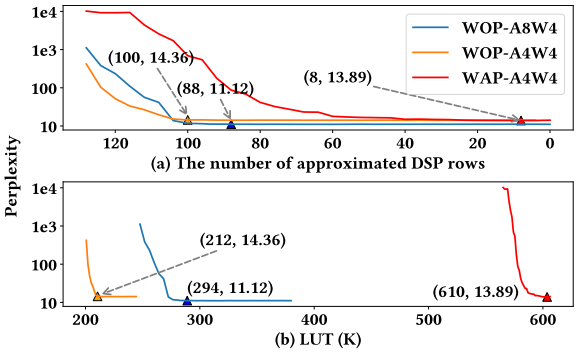

Fig. 6. The illustration of the perplexity (WikiText2) decrease process during our accuracy-guaranteed search for OPT-6.7B. The x-axis of (a) is the number of approximated DSP rows (i.e., the length of Idx[i:] in Algorithm 3), and the x-axis of (b) is the corresponding LUT consumption. The point marked in triangle is the result listed in Table II.

C. Interact between LUTs and Accuracy

Fig. 6 illustrates the perplexity decrease process during our accuracy-guaranteed search for OPT-6.7B. As Fig. 6 (a) shows, the perplexity decreases with the decrease of the number of approximated DSP rows which means fewer approximated weights. Since the DSP unit with intra-DSP approximation (DSP-w) consumes fewer LUTs than the one without approximation (DSP-o), as listed in Table III, the perplexity decreases with the increase of LUTs, as illustrated in Fig. 6 (b). Fig. 6 verifies the effectiveness of the accuracy impact profile step in Algorithm 3. The perplexity of all packing settings decreases drastically at the beginning but fluctuates in a rather limited range at the later stage. This indicates that the DSP units at the beginning stage are more likely to affect the model's perplexity. With the accuracy impact profile step, the accuracy-guaranteed search prioritizes setting these beginning stage DSP units to compute without approximation. This minimizes the number of DSP units without approximation, thus minimizing the LUT consumption.

D. Scalability

For the scalability over model size, we evaluate DWA across three LLMs with similar architecture but different sizes (OPT-125M, OPT-1.3B and OPT-6.7B). As listed in Table V, DWA provides stable LUT consumption for the three LLMs across all packing settings. For example, for WOP-A8W4 packing, DWA consumes 264K, 272K and 294K LUTs for OPT-125M, OPT-1.3B and OPT-6.7B, respectively. Compared with consumed LUTs of NPA (1139K), DWA achieves similar performance in reducing LUTs for LLMs with different sizes. Table V also reports the three LLMs' perplexity with DWA. The perplexity fluctuations of these three LLMs are all reduced to a rather small value (relatively -2.5% to 2.9%), compared with the original quantized perplexity. According to related literature [21], [24], [26], [33], this degree of perplexity fluctuation is imperceptible to users.

TABLE V
THE LUT CONSUMPTION AND PERPLEXITY OF DWA WITH DIFFERENT SIZED LLMS AND DSP ARRAYS.

LLM	Packing	LUTs (K)		Perplexity							
				WikiText2				C4			
		DWA 128×128	DWA 256×256	Original FP16	Original Quantized	DWA 128×128	DWA 256×256	Original FP16	Original Quantized	DWA 128×128	DWA 256×256
OPT-125M	WOP-A8W4	264	1058	27.66	29.74	29.88	29.95	24.61	26.07	26.16	26.16
	WOP-A4W4	205	820		32.95	32.97	32.94		28.29	28.30	28.30
	WAP-A4W4	602	2404			33.21	33.25			28.39	28.44
OPT-1.3B	WOP-A8W4	272	1106	14.62	15.86	15.88	15.92	14.72	15.69	16.15	15.89
	WOP-A4W4	212	838		30.53	30.31	30.42		26.64	26.56	26.63
	WAP-A4W4	603	241			30.71	30.78			26.77	26.77
OPT-6.7B	WOP-A8W4	294	1139	10.86	11.05	11.12	11.12	11.74	12.67	13.01	12.88
	WOP-A4W4	212	842		14.18	14.36	14.36		16.21	15.81	15.75
	WAP-A4W4	610	2376			13.89	14.21			15.95	16.35

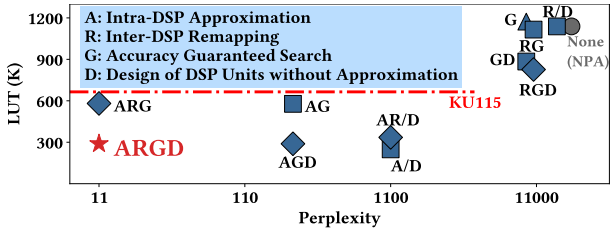

Fig. 7. The results of different component combinations operated upon OPT-6.7B with the WOP-A8W4 packing.

To validate scalability over DSP array size, we further integrate DWA with a larger DSP array (256×256) and compare the LUT consumption and perplexity with those of the former 128×128 DSP array. It should be noted that, to the best of our knowledge, the 256×256 DSP array can not be put into any single off-the-shelf FPGA. This DSP array mainly targets at the deployment scenario with multiple FPGAs, which is also a frequently discussed research topic [18], [34] in LLM accelerators. The specific technique of multiple FPGA deployment is not the focus of this paper, and DWA can be combined with those methods. As shown in Table V, the LUT consumption of the 256×256 DSP array is roughly 4 times that of the 128×128 DSP array, which is a reasonable factor for scaling the DSP array. Specifically, all consumed LUTs of the 256×256 DSP array with our design can be accommodated by 4 KU115 chips, including 2640K LUTs. In Table V, the 256×256 DSP array exhibits a similar perplexity fluctuation as the 128×128 DSP array, where the relative perplexity fluctuation ranges from -2.8% to 1.7% across three LLMs.

E. Ablation Study

Our DWA consists of four components: intra-DSP approximation, inter-DSP remapping, accuracy-guaranteed search and the design of DSP units without approximation. We conduct an ablation study for the four components by performing different component combinations for OPT-6.7B with WOP-A8W4. Fig. 7 shows the scatter diagram of the results, with the x-axis and y-axis indicating perplexity on WikiText2 and LUT consumption, respectively. The closer the point coordinates are to the origin, the smaller the model's perplexity and the less LUT consumption will be.

From Fig. 7, we can find that DWA with all four components (ARGD) achieves the best performance since it lies the closest point to the origin. Specifically, ARGD consumes 294K LUTs, and the perplexity is 11.12, as reported in Table II. The point "None (NPA)", located at the top right corner of the figure, refers to NPA's performance. The points including "/D" represent the overlapped points with or without component D, which have similar performance. For example, the point AR/D represents overlapped points AR and ARD. This is because component D (design of DSP units without approximation) becomes effective only when component G (accuracy-guaranteed search) is adopted. Although some points consume fewer LUTs than ARGD, they exhibit a significantly worse accuracy. For example, the point A/D consumes 248K LUTs since it configures all DSP units to intra-DSP approximation. However, it provides an unacceptable 1112 perplexity due to the absence of other components.

The effectiveness of each component is also verified in Fig. 7. Firstly, the points with component A (AG, AGD, AR/D, A/D, ARGD and ARG) are all located in the bottom left of the points without A (G, GD, RG, RGD and R/D). That is to say, the intra-DSP approximation can always reduce the LUTs and preserve the models' accuracy. Secondly, the effectiveness of component R (inter-DSP remapping) can be verified by comparing ARGD and AGD. With the same approximation subset as ARGD, AGD consumes 288K LUTs, slightly less than ARGD due to the absence of the activation snippets routing module in inter-DSP remapping. However, AGD has the perplexity of 237, still introducing unacceptable accuracy decrease. If AGD does not adopt ARGD's approximation subset, all DSP units will be configured to compute without approximation after the accuracy-guaranteed search, resulting in a 31% increase in LUT consumption compared to ARGD. Thirdly, by comparing (AGD, A/D) or (ARGD, AR/D), we can find that the points without component G always present worse accuracy. With the calibration dataset, our accuracy-guaranteed search algorithm preserves models' accuracy by adaptively finding the optimal trade-off between LUTs and perplexity. Fourthly, the effectiveness of D is also verified by comparing ARGD and ARG. ARGD and ARG have the similar perplexity, but ARG consumes 582K LUTs. This is because DSP units without approximation occupy 31% of the total DSP units in the DSP array for OPT-6.7B with WOP-A8W4. That is, for the 128×128 DSP array, there are 40 rows of DSP units configured to compute without approximation. LUTs consumed by these DSP units occupy 40% of the total LUTs, and component D reduces LUTs by threefold for each DSP unit compared to ARG.

V. CONCLUSION

To address the issues of existing indiscriminant weight approximation (IWA) methods regarding significant accuracy decrease and high LUT consumption, in this paper we propose discriminant weight approximation (DWA) for efficient DSP packing in LLM accelerators. Experimental results show that compared with existing IWA methods, DWA achieves the same chip area reduction performance but reduces the LUT consumption by 2.6 to 4.6 times. Moreover, the accuracy of DWA is far better than that of existing IWA methods.

REFERENCES

[1] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[2] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[3] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[4] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[5] J. Li, W. Wang, and W.-J. Li, "Hardware computation graph for dnn accelerator design automation without inter-pu templates," in *International Conference on Computer-Aided Design (ICCAD)*, 2022.

[6] ——, "Hardware computation graph for dnn accelerator design automation without inter-pu templates," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, pp. 1–1, 2025.

[7] AMD, "Convolutional neural network with int4 optimization on xilinx devices," https://www.xilinx.com/support/documentation/whitepapers/wp521-4bit-optimization.pdf, 2020.

[8] ——, "Deep learning with int8 optimization on xilinx devices," https://www.xilinx.com/support/documentation/whitepapers/wp486-deep-learning-int8.pdf, 2017.

[9] J. Sommer, M. A. Özkan, O. Keszocze, and J. Teich, "Dsp-packing: Squeezing low-precision arithmetic into fpga dsp blocks," in *International Conference on Field-Programmable Logic and Applications (FPL)*, 2022.

[10] A. C. Mert, H. Azgin, E. Kalali, and I. Hamzaoglu, "Efficient multiple constant multiplication using dsp blocks in fpga," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2018.

[11] H.-S. Suh, J. Meng, T. Nguyen, V. Kumar, Y. Cao, and J.-S. Seo, "Algorithm-hardware co-optimization for energy-efficient drone detection on resource-constrained fpga," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 16, no. 2, pp. 1–25, 2023.

[12] S. Lee, D. Kim, D. Nguyen, and J. Lee, "Double mac on a dsp: Boosting the performance of convolutional neural networks on fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 38, no. 5, pp. 888–897, 2019.

[13] E. Kalali and L. R. van, "Near-precise parameter approximation for multiple multiplications on a single dsp block," *IEEE Transactions on Computers (TC)*, vol. 71, no. 9, pp. 2036–2047, 2022.

[14] S. Rasoulinezhad, H. Zhou, L. Wang, and P. H. Leong, "Pir-dsp: An fpga dsp block architecture for multi-precision deep neural networks," in *International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2019.

[15] X. Liu, Y. Chen, P. Ganesh, J. Pan, J. Xiong, and D. Chen, "Hikonv: High throughput quantized convolution with novel bit-wise management and computation," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022.

[16] J. Zhang, M. Zhang, X. Cao, and G. Li, "Uint-packing: Multiply your dnn accelerator performance via unsigned integer dsp packing," in *Design Automation Conference (DAC)*, 2023.

[17] E. Luo, H. Huang, C. Liu, G. Li, B. Yang, Y. Wang, H. Li, and X. Li, "Deepburning-mixq: An open source mixed-precision neural network accelerator design framework for fpgas," in *International Conference on Computer-Aided Design (ICCAD)*, 2023.

[18] H. Chen, J. Zhang, Y. Du, S. Xiang, Z. Yue, N. Zhang, Y. Cai, and Z. Zhang, "Understanding the potential of fpga-based spatial acceleration for large language model inference," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 18, no. 1, pp. 1–29, 2024.

[19] S. Zeng, J. Liu, G. Dai, X. Yang, T. Fu, H. Wang, W. Ma, H. Sun, S. Li, Z. Huang *et al.*, "Flightllm: Efficient large language model inference with a complete mapping flow on fpgas," in *International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2024.

[20] AMD, "Amd kintex ultrascale fpga," https://www.xilinx.com/products/boards-and-kits/dk-u1-kcu1500-g.html, 2023.

[21] Y. Lin, H. Tang, S. Yang, Z. Zhang, G. Xiao, C. Gan, and S. Han, "Qserve: W4a8kv4 quantization and system co-design for efficient llm serving," *arXiv preprint arXiv:2405.04532*, 2024.

[22] W. Shao, M. Chen, Z. Zhang, P. Xu, L. Zhao, Z. Li, K. Zhang, P. Gao, Y. Qiao, and P. Luo, "Omniquant: Omnidirectionally calibrated quantization for large language models," in *International Conference on Learning Representations (ICLR)*, 2024.

[23] AMD, "Ultrascale architecture dsp slice," https://docs.amd.com/v/u/en-US/ug579-ultrascale-dsp, 2021.

[24] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "Awq: Activation-aware weight quantization for llm compression and acceleration," in *Conference on Machine Learning and Systems (MLSys)*, 2024.

[25] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning (ICML)*, 2023.

[26] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," in *International Conference on Learning Representations (ICLR)*, 2023.

[27] AMD, "Dsp48e2," https://docs.amd.com/r/en-US/ug974-vivado-ultrascale-libraries/DSP48E2, 2024.

[28] S. Arora, T. Leighton, and B. Maggs, "On-line algorithms for path selection in a nonblocking network," in *ACM Symposium on Theory of Computing (STOC)*, 1990.

[29] T. T. Lee and S.-Y. Liew, "Parallel routing algorithms in benes-clos networks," in *International Conference on Computer Communications (InfoCom)*, 1996.

[30] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.

[31] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research (JMLR)*, vol. 21, no. 140, pp. 1–67, 2020.

[32] AMD, "Amd virtex ultrascale+ fpga," https://www.xilinx.com/products/boards-and-kits/1-1sn9uo4.html, 2023.

[33] Y. Zhao, C.-Y. Lin, K. Zhu, Z. Ye, L. Chen, S. Zheng, L. Ceze, A. Krishnamurthy, T. Chen, and B. Kasikci, "Atom: Low-bit quantization for efficient and accurate llm serving," in *Conference on Machine Learning and Systems (MLSys)*, 2024.

[34] S. Hong, S. Moon, J. Kim, S. Lee, M. Kim, D. Lee, and J.-Y. Kim, "Dfx: A low-latency multi-fpga appliance for accelerating transformer-based text generation," in *International Symposium on Microarchitecture (MICRO)*, 2023.