

An Efficient Accelerator Based on Lightweight Deformable 3D-CNN for Video Super-Resolution

Siyu Zhang^{ID}, Wendong Mao^{ID}, and Zhongfeng Wang^{ID}, *Fellow, IEEE*

Abstract—Deformable convolutional networks (DCNs) have shown outstanding potential in video super-resolution with their powerful inter-frame feature alignment. However, deploying DCNs on resource-limited devices is challenging, due to their high computational complexity and irregular memory accesses. In this work, an algorithm-hardware co-optimization framework is proposed to accelerate the DCNs on field-programmable gate array (FPGA). Firstly, at the algorithm level, an anchor-based lightweight deformable network (ALDNet) is proposed to extract spatio-temporal information from the aligned features, boosting the visual effects with low model complexity. Secondly, to reduce intensive multiplications, an innovative shift-based deformable 3D convolution is developed using low-cost bit shifts and additions, maintaining comparable reconstruction quality. Thirdly, at the hardware level, a dedicated critical processing core, together with a block-level interleaving storage scheme, is presented to avoid dynamic and irregular memory accesses caused by the deformable convolutions. Finally, an overall architecture is designed to accelerate the ALDNet and implemented on an Intel Stratix 10GX platform. Experimental results demonstrate that the proposed design can provide significantly better visual perception than other FPGA-based super-resolution implementations. Meanwhile, compared with the prior hardware accelerators, our design can achieve 2.75 \times and 1.63 \times improvements in terms of throughput and energy efficiency, respectively.

Index Terms—Video super-resolution, deformable convolution, field-programmable gate array (FPGA), hardware accelerator.

I. INTRODUCTION

VIDEO super-resolution (VSR) has been widely applied in various applications, such as video surveillance [1], high-definition devices [2], and satellite imagery [3]. The basic idea is to reconstruct high-resolution (HR) video streams from their low-resolution (LR) counterparts by gathering adjacent frames [4]. Over the past decade, encouraged by the unprecedented attainments of deep neural networks (DNNs), many learning-based VSR methods with alignment mechanisms have achieved remarkable progress [5]. Although surpassing the traditional interpolation-based methods [6], [7] significantly, they introduce expensive computation and memory overhead

when being deployed on edge devices. Therefore, designing a lightweight and hardware-friendly DNN model for VSR is still an open problem.

Previous learning-based methods [8] have promoted the visual perception of VSR to some extent with powerful high-end graphics processing unit (GPU) clusters. Nevertheless, sophisticated networks make them difficult to be performed in real-time on resource-limited mobile devices. Hence, high-speed and low-power edge deployments become essential to improving user experiences. Some advanced solutions to accelerate VSR models are being gradually exploited. Reference [9] developed a super-resolution (SR) method based on a hardware-friendly convolutional neural network (CNN) and designed a pipelined hardware architecture, which employed line memories to process each frame. Chang et al. [10] concentrated on reconstructing HR frames by a deconvolutional network and designing an energy-efficient accelerator to execute the designed model. However, the works mentioned above only perform single image super-resolution (SISR) for each frame individually and ignore the critical temporal consistency, causing non-negligible performance degradation for VSR. To incorporate the spatio-temporal information, ERVSR [11] relieved the problem of temporal inconsistency with a lightweight recurrent neural network (RNN). Unfortunately, it still needs to overcome the problem of aligning the reference frames with supporting frames by combining motion information.

Nowadays, the DCN [12], as one of the novel variants of standard CNNs, has attracted widespread attention in many fields [13], [14]. Numerous researchers have employed it to enhance the alignment abilities and attained promising VSR scores [15], [16]. Tian et al. [15] first explored the application of DCNs in VSR. They constructed a temporally-deformable alignment network (TDAN) to align frames in the embedding space and obtained a remarkable breakthrough with a small-sized model. After that, many well-designed variants have aroused great interest in many studies, such as the pyramid, cascading and deformable (PCD) alignment module [8], non-local structure [16], deformable 3D convolution (D3D) [17], flow-guided deformable alignment [18], and other technologies [19]. The intrinsic reason is that offsets extracted by the DCNs have extremely similar patterns to the intrinsic optical flow in videos. Moreover, the DCNs efficiently boost the offset diversity and capture better-aligned representations [20]. These achievements also bring huge requirements for accelerating the DCN-based VSR methods in resource-limited devices.

Manuscript received 1 December 2022; revised 19 February 2023; accepted 11 March 2023. Date of publication 23 March 2023; date of current version 30 May 2023. This work was supported in part by the National Key R&D Program of China under Grant 2022YFB4400604. This article was recommended by Associate Editor M. Martina. (Corresponding authors: Wendong Mao; Zhongfeng Wang.)

The authors are with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210008, China (e-mail: syzhang@smail.nju.edu.cn; wdmao@smail.nju.edu.cn; zfwang@nju.edu.cn).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/TCSI.2023.3258446>.

Digital Object Identifier 10.1109/TCSI.2023.3258446

1549-8328 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See <https://www.ieee.org/publications/rights/index.html> for more information.

There are noticeable differences between the DCNs and other DNNs, such as standard CNNs, deconvolutional neural networks (DeCNNs), and RNNs. Therefore, directly employing existing accelerators [10], [21], [22] to the DCNs will seriously degrade their computational efficiency or even make them fail to work. Specifically, the challenges are reflected in three aspects. Firstly, dynamically produced offsets enable the DCNs to sample arbitrary locations instead of regular sliding windows in the feature map, causing irregular memory access (IMA) patterns. Not only that, IMA incurs serious access conflicts, endangers parallel designs, and sacrifices input reuse. Secondly, flexible sampling positions force the processor to access partial off-chip pixels, causing cache misses. Thirdly, the DCNs introduce unavoidable extra computation overhead, which is mainly caused by the offset generations and neighboring feature interpolations.

These challenges drive many attempts to study dedicated accelerators for the DCNs [23], [24]. For example, several simplifications have been investigated to restrict the flexible and irregular receptive field, such as bounding the shape [25], [26], range [27], [28], and diversity [24] of offsets. Although these optimizations alleviate the IMA and overhead of loading offsets, oversimplifications will yield inaccurate motion estimation and thus seriously influence the visual consistency of VSR. On the contrary, to extremely preserve the flexibility of sampling grids, [29] developed a tile dependency table (TDT) and corresponding scheduler with complicated control logic. Besides, in [30], a memory-efficient DCN accelerator is proposed to retrieve input activations regularly by a specialized register array. However, the fly in the ointment is that [30] needs to store all pixels within a randomly deformable field on chip at the same time, which will aggravate unacceptable memory overhead if dealing with pixel-processing tasks with large data volumes.

Based on the above analysis, we propose an algorithm-hardware co-optimization framework to efficiently accelerate the VSR tasks with a lightweight deformable 3D-CNN. The main contributions of this paper are listed as follows:

- At the algorithm level, a hardware-inspired ALDNet is proposed to achieve outstanding visual quality of VSR with low model complexity. It aligns neighboring frames and combines spatio-temporal information to reconstruct high-quality HR video sequences.
- An innovative shift-based deformable 3D convolution (SDfConv3D) is developed to reduce expensive multiplications using hardware-friendly bit shifts and additions, maintaining fine reconstruction results. In addition, a tile decoupling computing strategy (TDS) is introduced to avoid accessing pixels of many tiles simultaneously and promote partial fusion dataflow for the hardware design.
- At the hardware level, to accelerate the SDfConv3D, a dedicated critical processing core (CPC), together with a block-level interleaving storage scheme, is presented to avoid dynamic and irregular memory accesses caused by the deformable convolutions. Besides, for the remaining convolutions, a reconfigurable auxiliary processing core (APC) is designed to cooperate with the CPC efficiently.

- Based on the algorithm-hardware co-optimization, an efficient accelerator is developed and implemented on an Intel Stratix 10GX platform. Experimental results show that our design performs superior visual perception in VSR and surpasses prior hardware accelerators significantly in terms of throughput and energy efficiency.

The rest of this paper is organized as follows. Section II gives a detailed introduction of previous works. Section III describes the specifics of the ALDNet. The overall architecture and processing cores are detailed in Section IV. Section V exhibits some evaluations regarding algorithms and hardware efficiency. Finally, the conclusion is drawn in Section VI.

II. RELATED WORKS

A. Deformable Convolutional Networks

Standard CNNs sample features by regular-shaped sliding windows and compute weighted summations with learnable parameters. Thus, a properly configured receptive field is essential for extracting reasonable visual representations. To enhance the flexibility and adaptability, Dai et al. [12] first proposed a DCN for object detection and semantic segmentation tasks, wherein a convolutional layer dynamically generated various offsets to enhance the sampling range. With the flexible offsets, the DCN can adaptively adjust the receptive field at different positions based on the actual object geometry.

Based on the vanilla DCNs, previous studies have designed many variants and achieved excellent accuracy in various computer vision tasks, such as action detection [31], image denoising [32], dehazing [33], and SR [15]. Novel researches employ deeper networks and more complex computing units to accurately model the offsets, but these techniques aggravate the training cost and memory resource overhead. Thus, it is challenging to construct lightweight DCNs for resource-constrained devices while maintaining advanced accuracy.

B. Deformable Convolutional Networks for VSR

Random motion of cameras and objects usually causes the adjacent frames to be misaligned. To explore inter-frame correlations, well-established VSR systems [18] employed explicit or implicit alignment mechanisms to mine complementary sub-pixel information. However, inaccurate motion estimation and compensation (MEMC) methods deteriorate the computational efficiency, and incur ambiguous perception.

Compared with the explicit alignment in traditional optical flow estimations, DCNs can implicitly align adjacent frames at the feature level, and model motion information more accurately. Tian et al. [15] first proposed a TDAN model based on the DCN. By dynamically predicting the offsets, they avoided inaccurate optical flow and warping strategies. To capture global correspondence and enhance fine details, EDVR [8] handled large motions with a powerful PCD module and won in NTIRE 2019. D3D [17] combined spatio-temporal and motion-aware information, achieving state-of-the-art quantitative scores. In NTIRE 2021, Basic VSR++ [18] used a flow-guided deformable alignment strategy to solve the

occlusion problem, and overcame the VSR track challenges. Thanks to the dynamical offsets, the DCN-based methods can maintain temporal alignment sufficiently, and tackle luminance changes or object movements adaptively. Therefore, in this paper, a novel DCN-based VSR model is developed to ensure superior visual quality and processing speed.

C. Hardware Implementations for DCNs

Recently, researchers have shown an increased interest in the DCN accelerator. To simplify the offsets, [25] and [26] deployed a depthwise deformable convolution on FPGA. They rounded the offsets to avoid bilinear interpolations, and designed a square respective field with varying dilation factors. Reference [27] added a regularization term on the training loss to restrict the randomly expanded sampling ranges. However, it transfers the interpolated results to external memory, causing colossal power consumption and memory bandwidth requirements. In [28], a differential round function was introduced to bound the offsets and quantize their fractional parts to one bit. For pixel processing tasks, a joint denoising and demosaicing method (DeformJDD) [24] was first proposed to share a group of offsets at different positions. Nevertheless, random motions of different targets lead to diverse optical flow vectors for different pixels in practical video streams. The oversimplifications in these works inevitably hinder the DCNs from capturing diverse offsets and accurate optical flow in the video processing tasks.

To optimize dynamic and irregular accesses, [29] and [30] developed a sophisticated scheduling algorithm and a dimension conversion register array, respectively. Unfortunately, these works mainly support object detection tasks with limited input resolution. Therefore, the large data volume in the video streams will leave them with unbearable resource overhead.

D. Hardware Implementations for Super-Resolution

Super-resolution (SR) is mainly divided into two categories: SISR and VSR. For SISR [2], [7], the basic idea is to utilize spatial information to upsample the LR video streams frame by frame. Kim et al. [2] trained a linear mapping kernel to upscale LR input patches, and designed a hardware architecture for the upscale step. In [34], horizontal and vertical flips were applied to avoid pre-enlarging inputs, and a hardware accelerator was also designed for a CNN-based SR model. To reduce computational complexity and fully utilize the limited memory resource, [9] processed the LR frames line-by-line with a lightweight CNN model and achieved real-time processing. Reference [10] developed an SR system to support DeCNNs and CNNs simultaneously. Considering that the CNN-based SR methods required massive computation overhead and communication bandwidth, SRNPU [35] proposed a tile-based selective strategy and reconfigurable cyclic ring architecture. However, due to the negligence of critical temporal information and the limited capacity of models, these approaches affect the consistency of upsampled video streams and thus degrade the visual effect.

In contrast, VSR methods upscale the target frame by combining multiple adjacent LR frames and fusing spatial and

temporal features. For example, ERVSR [11] was developed to explore the underlying inter-frame temporal correlations. Nevertheless, RNN-based VSR approaches require considerable memory resources for hidden states. Moreover, the lack of the temporal alignment also makes it challenging to adapt to fast-varying motions. To date, few studies have attempted to incorporate the inter-frame alignment in the VSR accelerators.

III. HARDWARE-INSPIRED VSR METHOD

A. Overview of ALDNet Architecture

Considering the limited hardware resources of edge devices, we propose a hardware-inspired deformable 3D-CNN, called ALDNet, which is depicted in Fig. 1(a). The basic principle is to restore an HR frame $I_z^{HR} \in \mathbb{R}^{sNix \times sNiy \times Nif}$ from its corresponding LR reference frame $I_z^{LR} \in \mathbb{R}^{Nix \times Niy \times Nif}$ and $Niz - 1$ supporting frames $\{I_{z-(Niz-1)/2}^{LR}, \dots, I_{z-1}^{LR}, I_{z+1}^{LR}, \dots, I_{z+(Niz-1)/2}^{LR}\}$, where $Nix \times Niy$ is the size of the LR frame, s is the upsampling factor, and Nif denotes the number of input channels. The main goal is to make the prediction I_z^{SR} as close to the ground truth I_z^{HR} as possible. Generally, the ALDNet can be divided into four parts: shallow feature extraction, implicit feature alignment, deep feature fusion, and reconstruction.

The first stage extracts the spatio-temporal information with a depthwise separable 3D convolutional layer, including a depthwise 3D convolutional layer (DwConv3D) and a pointwise 3D convolutional layer (PwConv3D).

In the second stage, intermediate feature sequences are computed at tile granularity based on the TDS. As shown in Fig. 1(b), ResD3D blocks stacked by several SdfConv3Ds perform the spatial deformations and temporal alignments at the same time. Specifically, given a tile $F \in \mathbb{R}^{Tix \times Tiy \times Niz \times Nif}$ as an input of the ResD3D blocks $H_{ResD3D}(\bullet)$:

$$F^A = H_{ResD3D}(F), \quad (1)$$

where $F^A \in \mathbb{R}^{Tox \times Toy \times Noz \times Nof}$ is the aligned feature frames. The height and width of each tile are $Tox = Tix$ and $Toy = Tiy$. Noz and $Nof = Nif$ are the number of frames and channels. Besides, residual learning is utilized to improve the learning convergence and prevent gradient vanishing.

In the third stage, the obtained Niz frames F^A are further aggregated across the temporal dimension by a temporal average pooling (TAP) function:

$$\tilde{F}_{x,y,m} = \frac{1}{Niz} \sum_{z=0}^{Niz-1} F_{x,y,z,m}^A, \quad (2)$$

where $\tilde{F} \in \mathbb{R}^{Nox \times Noy \times Nof}$ denotes the fused feature maps by combining all the output tiles. Nox and Noy are the sizes of spatial dimension, and $Nix = Nox$, $Niy = Noy$. Then, ResC2D blocks, which are constructed by depthwise separable 2D convolutional layers in Fig. 1(c), are responsible for capturing deep spatial representations $\tilde{F}' \in \mathbb{R}^{Nox \times Noy \times Nof}$.

In the fourth stage, \tilde{F}' is propagated into an upscale block to reconstruct I_z^{SR} in Fig. 1(d). Considering the high structural similarity between input and output sequences, an anchor-based residual learning strategy [36] is introduced to replace

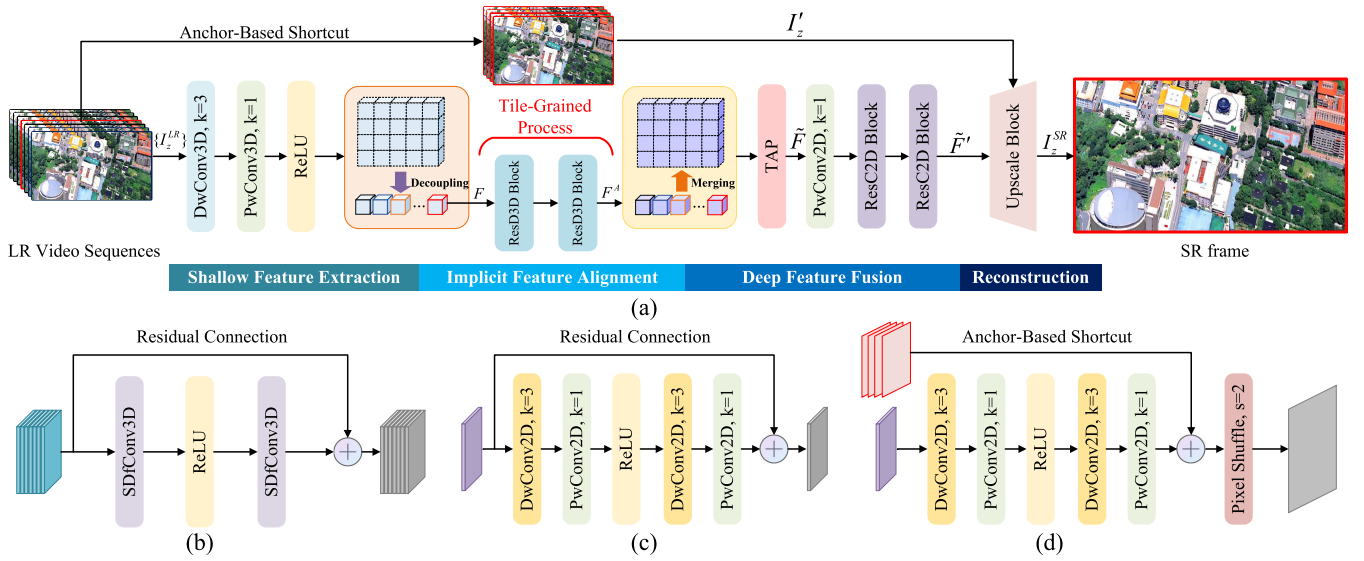


Fig. 1. Block diagrams of the proposed hardware-inspired VSR method. The stride of all the convolutions is 1. Except that the number of the input channel in the first convolutional layer and the output channel in the last convolution layer is 1, the channel number of the remaining convolutions equals 36. $Nkx = Nky = Nkz = k$ indicates the kernel size and s expresses the upsampling factor. (a) Anchor-based lightweight deformable network (ALDNet). (b) ResD3D block. (c) ResC2D block. (d) Upscale block.

pixel-wise interpolations in the shortcut. The LR reference frame is repeated by s^2 times and concatenated into a sequence $I_z^{LR} \in \mathbb{R}^{Nix \times Niy \times s^2}$ in the anchor-based shortcut. Afterward, I_z^{LR} is upsampled to yield an anchor map for the final SR frame. Compared with the mainstream VSR methods [17], [18] which pre-enlarge the LR reference frames with multiplication-based interpolations, the main benefits are two-fold: the anchor-based residual learning avoids high-cost pixel-wise interpolations, and the post-upsampling process prevents performing extensive convolutions in the HR feature frames, economizing computing resource requirements in the hardware design.

B. Shift-Based Deformable 3D Convolution

Intensive multiplications empower robust model fitting abilities to the DNNs, but constrain their deployments on resource-limited devices. To handle this problem, existing studies [37], [38] replaced the heavy multiplications with low-cost bit shifts and additions, and achieved a positive trade-off between efficiency and accuracy. These works promote us to design a hardware-friendly SDfConv3D with a shift pointwise 3D convolutional layer (SPwConv3D) and shift bilinear interpolation layer (SBilinear) to reduce the resource burden of the multiplications. In Fig. 2, SDfConv3D consists of three stages: offset generation, spatial deformation, and feature extraction.

1) *Offset Generation*: At the i -th loop, a tile $F^i \in \mathbb{R}^{Tix \times Tiy \times Niz \times Nif}$ is employed to predict the corresponding offsets $\Psi^i \in \mathbb{R}^{Tix \times Tiy \times Niz \times Noff}$. For example, for a pixel $F^i(x_0^i, y_0^i, z_0^i)$, the offset generation process H_Ψ becomes:

$$\begin{aligned} \Psi^i(x_0^i, y_0^i, z_0^i) &= H_\Psi(F^i(x_0^i, y_0^i, z_0^i)) \\ &= \left\{ (\Delta x_n^i, \Delta y_n^i) | n = 1, \dots, |\mathcal{R}| \right\}, \end{aligned} \quad (3)$$

where \mathcal{R} denotes a 3D sampling grid of size $Nkx \times Nky \times Nkz$. If the kernel size is $Nkx = Nky = Nkz = 3$, we can get

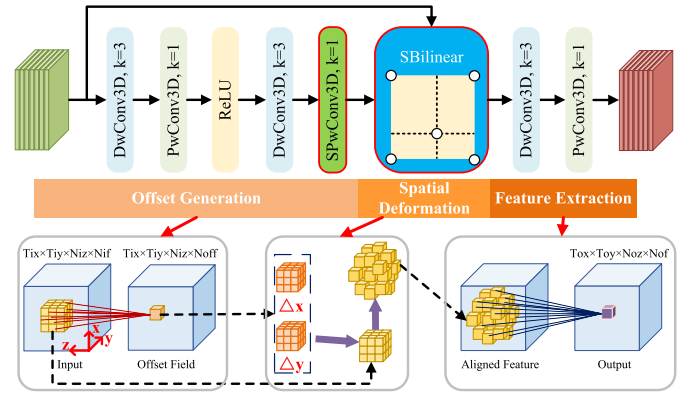


Fig. 2. Shift-based deformable 3D convolution (SDfConv3D).

$\mathcal{R} = \{(-1, -1, -1), (-1, -1, 0), \dots, (1, 1, 0), (1, 1, 1)\}$ and the offset channel numbers $Noff = 2|\mathcal{R}| = 54$. Besides, a series of cascaded separable 3D convolutional layers and a SPwConv3D are employed to build H_Ψ . For the SPwConv3D layer, the weight tensor $\omega^{SPwConv3D} \in \mathbb{R}^{1 \times 1 \times 1 \times Nif \times Nof}$ is parameterized by integer power of 2:

$$s = \text{sgn}(\omega^{SPwConv3D}), p = \lfloor \log_2 |\omega^{SPwConv3D}| \rfloor, \quad (4)$$

where s and p indicate low-cost sign flips and bitwise shifts, respectively. Thus, the computing process of the SPwConv3D can be expressed by logical bit shifts and additions:

$$\begin{aligned} O_j^{SPwConv3D} &= \sum_{m=0}^{Nif-1} F_m^{SPwConv3D} * \omega_{j,m}^{SPwConv3D} \\ &= \sum_{m=0}^{Nif-1} F_m^{SPwConv3D} * (s_{j,m} \cdot 2^{p_{j,m}}), \end{aligned} \quad (5)$$

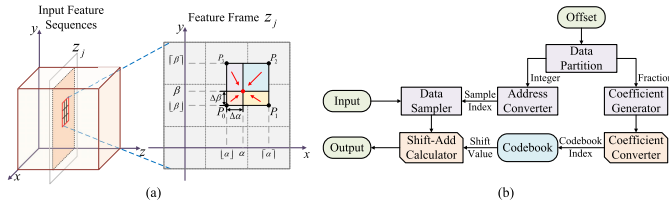


Fig. 3. Description of the proposed shift bilinear interpolation layer (SBilinear). (a) Traditional bilinear interpolation. (b) Computing flow of SBilinear.

where $F_m^{SPwConv3D}$ and $O_j^{SPwConv3D}$ denote the m -th input and j -th output feature map, respectively. $*$ is a convolution.

2) *Spatial Deformation*: To further save computational cost, the SDfConv3D executes spatial deformations and keeps the original temporal structure. Specifically, since the spatially-variant offsets Ψ^i may be fractional, the coordinates of the deformed respective filed $\{(\alpha_n^i, \beta_n^i, \gamma_n^i)\}$ can be expressed as:

$$\begin{aligned}\alpha_n^i &= x_0^i + x_n^i + \Delta x_n^i = \lfloor \alpha_n^i \rfloor + \Delta \alpha_n^i, \\ \beta_n^i &= y_0^i + y_n^i + \Delta y_n^i = \lfloor \beta_n^i \rfloor + \Delta \beta_n^i, \\ \gamma_n^i &= z_0^i + z_n^i,\end{aligned}\quad (6)$$

where $(x_n^i, y_n^i, z_n^i) \in \mathcal{R}$ represents coordinates in the regular sampling grid. $\Delta \alpha_n^i$ and $\Delta \beta_n^i$ are the fractional parts of α_n^i and β_n^i , respectively. To synthesize exact features based on the non-integer coordinates, a bilinear interpolation $G(\bullet)$ is introduced and shown in Fig. 3(a). It can be formulated as:

$$G(F^i, \Delta \alpha_n^i, \Delta \beta_n^i) = \mathbf{U}^T \mathbf{V}. \quad (7)$$

Here, the interpolation coefficient vector \mathbf{U} and the sampled pixel vector \mathbf{V} for each feature map can be specified to:

$$\mathbf{U} = \begin{bmatrix} 1 - \Delta \alpha_n^i - c_1 \\ c_1 \\ \Delta \alpha_n^i - c_0 \\ c_0 \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} F^i(\lfloor \alpha_n^i \rfloor, \lfloor \beta_n^i \rfloor, \gamma_n^i) \\ F^i(\lfloor \alpha_n^i \rfloor, \lfloor \beta_n^i \rfloor, \gamma_n^i) \\ F^i(\lfloor \alpha_n^i \rfloor, \lfloor \beta_n^i \rfloor, \gamma_n^i) \\ F^i(\lfloor \alpha_n^i \rfloor, \lfloor \beta_n^i \rfloor, \gamma_n^i) \end{bmatrix}, \quad (8)$$

where $c_0 = \Delta \alpha_n^i \Delta \beta_n^i$ and $c_1 = \Delta \beta_n^i - c_0$ are the shared parameters to reduce repeated calculations.

Considering that a 3D convolution needs $3 \times 3 \times 3 = 27$ multiplications to generate an output pixel, if extra bilinear interpolations are inserted, the multiplications will be increased by five times. A simple way is to transform \mathbf{U} into the power of 2 formats. However, unlike the random trainable weights, the coefficients have a fixed range $[0, 1]$. Computing power of 2 coefficients will bring larger approximation errors in $[0.5, 1]$ than other segments. To address these issues, [38] accumulated shift values, which were retrieved from different pre-defined codebooks, effectively controlling the approximation errors.

On the basis of this, a shift bilinear interpolation (SBilinear) $\tilde{G}(\bullet)$ is proposed in this work to simplify the vector-based dot production of the traditional bilinear interpolations. As described in Fig. 3(b), each offset is partitioned into an integer and a fractional part. On the one side, an address converter converts the integer part into available indices to sample exact inputs for V . On the other side, a coefficient

generator is designed to transform the fractional part into the SBilinear coefficients. Then, each element u_d of the coefficient vector U can be approximated to the accumulations of L power of 2 values by a coefficient converter, which can be expressed as:

$$u_d \approx \sum_{l=1}^L C_l [\Omega_d(l)],$$

$$C_l = \{0, \pm 2^{-l+1}, \pm 2^{-l}, \pm 2^{-l-1}, \dots, \pm 2^{-l-\lfloor M/2 \rfloor + 2}\}, \quad (9)$$

where C indicates the codebook set. Each codebook has M candidate elements. Ω_d is the index to search power of 2 values for u_d . Finally, the intensive vector-based dot productions of $G(\bullet)$ are simplified to several low-cost bit shifts and additions. The computing process of the SBilinear $\tilde{G}(\bullet)$ is outlined as Algorithm 1. In this paper, we choose $N = 2$ and $M = 15$ to simplify the following hardware designs in Section IV.

Algorithm 1 Computing Process of SBilinear

```

1 Require: SBilinear coefficient  $u_d \in \mathbf{U}$ , sampled feature  $v_d \in \mathbf{V}$ , the number of codebook elements  $M$ , the number of codebooks  $L$  and the number of SBilinear coefficients  $D$ .
2 Initialize:  $o_d = 0, D = 4$ ;
3 for  $d = 0; d < D; d++$  do
4    $q_d = 0$ ;
5   for  $l = 0; l < L; l++$  do
6      $\varepsilon = \text{sgn}(u_d)$ ;
7      $\delta = |u_d|$ ;
8      $\eta = \lfloor \log_2 \delta \rfloor$ ;
9     if  $\delta > 1.5 \times 2^\eta$  then
10       $\eta = \eta + 1$ ;
11      $\rho = \varepsilon \times 2^\eta$ ;
12      $\Omega_d(l) = \varepsilon \times (-l - \eta + 1)$ ;
13     if  $2 \times |\Omega_d(l)| > M - 1$  then
14       $\rho = 0$ ;
15       $\Omega_d(l) = 0$ ;
16      $u_d = u_d - \rho$ ;
17      $s_l \leftarrow C_l[\Omega_d(l)]$  // Deduce the shift value;
18      $q_d = q_d + v_d \gg s_l$ ;
19    $o_d = o_d + q_d$ ;
20 Return: Accumulated SBilinear results  $o_d$ ;
```

3) *Feature Extraction*: At last, with F^i and $\tilde{G}(\bullet)$, the SDfConv3D can be formulated as:

$$O^i(x_0^i, y_0^i, z_0^i) = \sum_{n=0}^{|\mathcal{R}|-1} \omega_n^{DW} \omega_n^{PW} \tilde{G}(F^i, \Delta \alpha_n^i, \Delta \beta_n^i), \quad (10)$$

where $\omega_n^{DW} \in \mathbb{R}^{Nkx \times Nky \times Nkz \times Nif}$ and $\omega_n^{PW} \in \mathbb{R}^{1 \times 1 \times 1 \times Nif \times Nof}$ indicates the convolution weights of the DwConv3D and PwConv3D layers, respectively. O^i denotes the output feature map of the SDfConv3D.

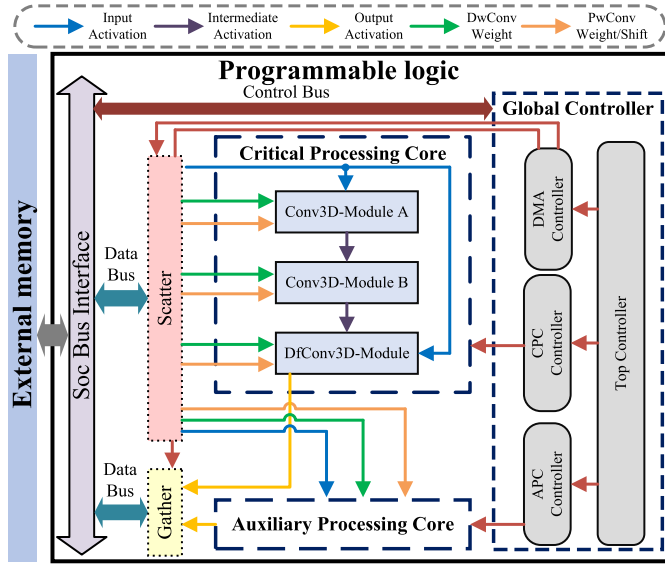


Fig. 4. Overall hardware architecture for ALDNet.

C. Tile Decoupling Computing Strategy

It is worth noting that the inter-frame alignment consumes many hardware resources. Inspired by [35], we propose a TDS to implement subregion-wise processes beyond the memory limit. It decouples the input sequences into multiple overlapping tiles in the spatial dimension. As shown in Fig. 1(a), the feature alignment stage is executed at tile granularity, and merely one tile is processed for each loop. The advantages of the TDS are two-fold. On the one hand, the subregion-wise process effectively prevents accessing partial pixels which exceed the scope of current tiles caused by the deformable convolution. Meanwhile, the overlap sufficiently maintains the visual coherence of the global shape and the fidelity of the details. On the other hand, in terms of hardware design, TDS offers a huge opportunity for designing partial fusion dataflow to integrate the computation-intensive ResD3D blocks into a single processing core. The partial fusion dataflow can effectively reduce the energy consumption and off-chip memory bandwidth of data movements caused by large input resolution.

IV. EFFICIENT HARDWARE ARCHITECTURE FOR ALDNET

A. Overall Hardware Architecture and Efficient Dataflow

In this section, we develop a hardware architecture for the inference of the proposed ALDNet. Fig. 4 depicts the top-level block diagram of the overall architecture, which mainly involves a **critical processing core (CPC)**, an **auxiliary processing core (APC)**, and a **global controller**. For clarity, we leverage arrows of different colors to highlight different data types, such as activations, weights, shift values, and control signals.

1) *Critical Processing Core*: CPC specifically performs the SDFConv3D in a fully pipelined manner between layers. To avoid additional memory overhead for intermediate partial sum (psum), our design methodology is to fuse the dataflow of depthwise and pointwise convolutions, and develop some

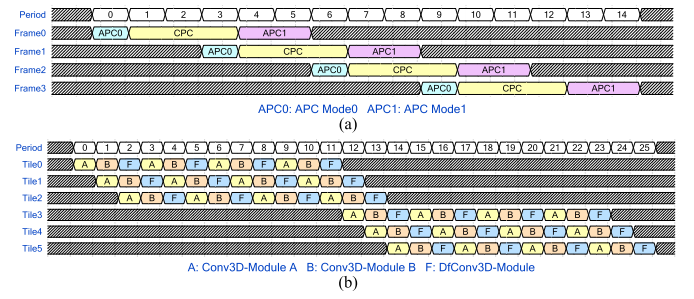


Fig. 5. Timing diagrams of different modules. (a) Processing schedule of CPC and APC. (b) Processing schedule of the elementary modules of CPC.

dedicated modules to accelerate different stages of the SDFConv3D in Fig. 2. Specifically, the offset generation process is modularized into an **A-type standard 3D convolution module (Conv3D-Module A)** and a **B-type standard 3D convolution module (Conv3D-Module B)**. Moreover, a **deformable 3D convolution module (DfConv3D-Module)** is responsible for performing the spatial deformation and 3D feature extraction.

2) *Auxiliary Processing Core*: To cooperate with CPC, APC supports the remaining convolutions of ALDNet, including a separable 3D convolution with one input channel and some separable 2D convolutions, using a few hardware resources. It is reconfigured into two modes by setting a switch flag and working in a time-division multiplexing manner.

3) *Controller*: In the global controller, the **CPC controller** and **APC controller** orchestrate the two cores to conduct different computations. The **direct memory access (DMA) controller** generates descriptors to manage data interactions between on-chip buffers with **external memory**. Besides, the **scatter** module takes charge of allocating the data to specified on-chip buffers. The **gather** module rearranges the outputs in a uniform layout for later use.

4) *Pipeline Schedule*: Fig. 5 demonstrates timing diagrams of the global and local dataflow. For clarity, in Fig. 5(a), an output frame is yielded by successively executing APC Mode0, CPC, and APC Mode1. By reasonably customizing computational parallelism and data reuse, APC alternatively switches the two working modes and executes with CPC cooperatively based on the data dependency of ALDNet. In Fig. 5(b), three convolution modules are abbreviated as A, B, and F, respectively. Since there are two ResD3D blocks in the implicit feature alignment stage, CPC needs to be repeated four times to produce an output tile. Meanwhile, each module is triggered by the arrival of three tiles in each period, ensuring high computing resource utilization and processing speed.

B. Deformable 3D Convolution Module

Generally, the deformable 3D convolution has two characteristics. For one thing, since the sampling positions are random, the utilization probability of each input is not uniform. For another, the offsets entirely depend on the inputs and vary on the fly. Thus, a block-level interleaving storage scheme is incorporated into the DfConv3D-Module to hinder the irregular and dynamic memory accesses, as shown in Fig. 6.

1) *Memory Management*: The input tiles are 4-D tensors with the size of $T_{ix} \times T_{iy} \times N_{iz} \times N_{if}$. To improve the

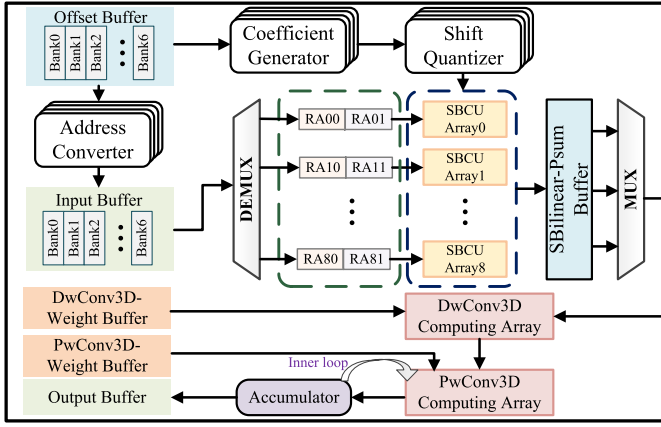


Fig. 6. Deformable 3D convolution module (DfConv3D-Module).

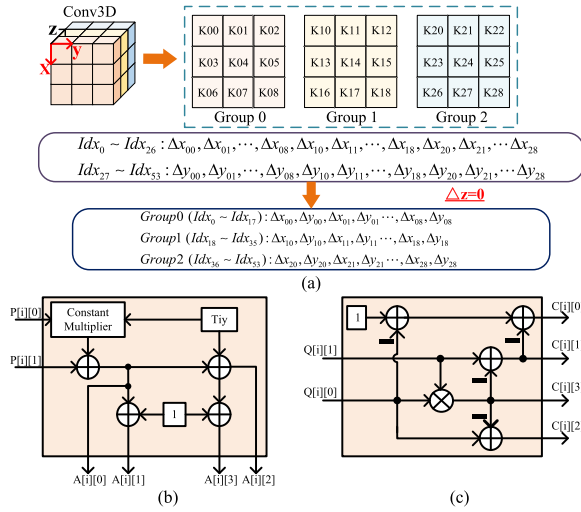


Fig. 7. Description of the process of offsets, where Idx denotes an input channel index. (a) Offset rearrangement rules. (b) Address Converter. (c) Coefficient Generator.

throughput, we sufficiently exploit the degree of parallelism in the temporal dimension. In this way, an Input Buffer including $N_{iz} = P_{iz} = 7$ input banks is adopted to save video frames separately. The Offset Buffer also contains seven banks, similarly. Besides, weights are streamed into the DwConv3D-Weight Buffer and the PwConv3D-Weight Buffer through the scatter module. Finally, results are sent to the Output Buffer, which has the same pattern as the Input Buffer, and wait to be fetched later. To perform the residue learning, identity mapping inputs are usually prefetched by the Output Buffer when the second SDfConv3D is executing. This ensures that the residual values generated by the DfConv3D-Module can be accurately fused with the corresponding identity mapping values.

2) *Offsets Decomposition*: Based on the above layout, $N_{off} \times N_{iz} = 54 \times 7$ offsets are accessed in each cycle. Considering that the SDfConv3D performs spatial deformations, we seek to unroll the space dimension of the 3D kernel, so the degree of parallelism in a 3D kernel is $P_{ks} = N_{kx} \times N_{ky} = 9$. Specifically, as shown in Fig. 7(a), since the original order of offsets does not meet the design requirements, we first rearrange and split the offsets into $N_{kz} = 3$ groups. Each group

contains $P_{ks} \times P_{iz} = 9 \times 7$ pairs of coordinate offsets. By this means, a complete 3D convolution operation can be performed in N_{kz} loops. Then, according to the scales determined by the fixed-point quantization, the offsets are partitioned into a set of integer parts $P[i]$ and fractional parts $Q[i]$, $i = 0, 1, \dots, 62$. Next, as shown in Fig. 7(b) and (c), $P[i]$ and $Q[i]$ are assigned to address converters and coefficient generators, respectively. For example, if the integer parts of Δx_{00} and Δy_{00} are assigned to $P[i][0]$ and $P[i][1]$, and their fractional parts are assigned to $Q[i][0]$ and $Q[i][1]$, the corresponding sampling indices $A[i]$ and SBilinear coefficients $C[i]$ can be yielded by the address converter and coefficient generator, respectively. At last, since massive multiplication-based interpolations bring expensive computation overhead, we draw support from the shift quantizer, following Eq. (9). Thus, $C[i]$ can be transformed into codebook indices to reduce the usage of multipliers by low-cost bit shifts and additions.

3) *Block-Level Interleaving Storage Scheme*: To handle the IMA patterns and boost input reuse of deformable convolutions, the block-level interleaving storage scheme is designed as an additional hierarchy of memory to cache sampling values and perform parallel shift-based interpolations with low memory resource overhead, as shown in Fig. 8. There are two groups of register arrays (RAs), named RA0 and RA1, which are controlled by two signals. To obtain high throughput, each RA accommodates $N_b \times \lceil N_{if}/P_{if} \rceil \times P_{ks} = 4 \times 4 \times 9$ data blocks and works in a ping-pong manner. “0” means that old data can be overwritten from top to bottom by new data, called write status. “1” indicates that full RA sends data to following computing modules from left to right, called read status. Since the SBilinear employs $N_b = 4$ neighboring features to compute the deformed features in $N_b = 4$ loops, this affects the computational efficiency and resource overhead seriously. To that end, to perform SBilinear with N_b neighbors simultaneously, the running directions of the two states are configured to be orthogonal. Generally, the computing flow for the block-level interleaving storage scheme is as follows:

- (1) *Initial Stage*: P_{ks} input groups of size $N_{if} \times P_{iz} = 36 \times 7$ are retrieved in each cycle according to the sampling indices. Then, to unroll the input channel dimension, each group is split into four data blocks of size $P_{if} \times P_{iz} = 9 \times 7$ to fill up RA1 by row. After $N_b = 4$ cycles, the status signal of RA1 is set as “1”.
- (2) *Stage 0 (Status 01)*: In each cycle, new $P_{ks} \times \lceil N_{if}/P_{if} \rceil$ blocks are loaded on RA0 and overwrite the old data by row. Simultaneously, $P_{ks} \times N_b$ blocks are sent from RA1 to the computational logic by column.
- (3) *Stage 1 (Status 10)*: After $N_b = 4$ loops, the statuses are exchanged. Likewise, the old data of RA1 is overwritten by row in each cycle. RA0 pops up $P_{ks} \times N_b$ blocks by column at the same time. Then, the whole process will return to stage 0 again after $N_b = 4$ loops.
- (4) *Final Stage*: Once all the preloaded weights accomplish transferring and calculating, the treatment of these inputs will stop. Afterward, a new round of weights and input blocks is transferred on chip and repeat the above routes.

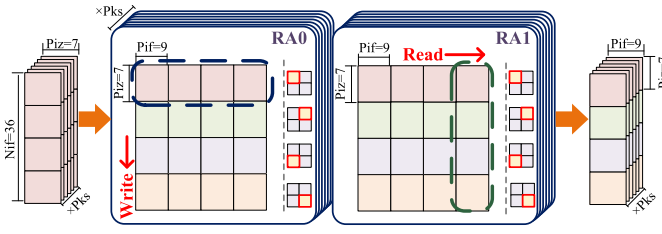


Fig. 8. Illustration of the block-level interleaving storage scheme.

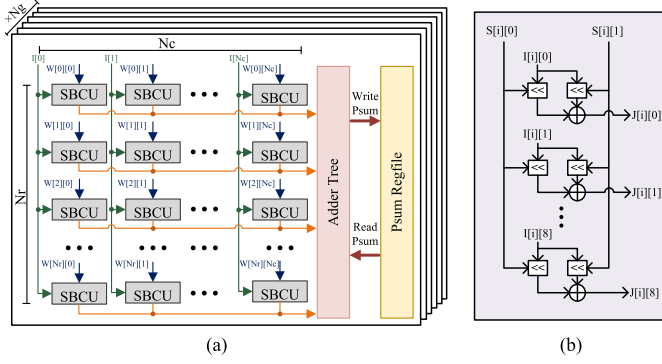


Fig. 9. Illustration of the computing array. (a) Structure of the computing array. (b) Microarchitecture of SBilinear computing units (SBCU).

4) *Computing Array of Different Layers*: The computational logic of DfConv3D-Module mainly consists of some computing arrays for the SBilinear, DwConv3D and PwConv3D operations. To streamline the design process, these arrays are designed with similar high-level connection types, but with different microarchitectures of the computing units, as shown in Fig. 9(a). Different layers can be supported by direct instantiations of this generic structure and simply setting design parameters. Specifically, to enhance data reuse and reduce resource overhead, inputs are shared between computing unit columns, and weights are preloaded on different computing units, respectively. Following the weight stationary dataflow [21], after finishing all calculations, results exported from computing units are temporally reduced by adder trees and transported to psum regfiles. To be clear, we leverage the number of rows, columns, and clusters, expressed as (Nr, Nc, Ng) , to define this computing array.

When it comes to the SBilinear, the structure parameters are $(Nr, Nc, Ng) = (Piz, Nb, Pks) = (7, 4, 9)$. Each SBilinear (SBCU) unit in Fig. 9(b) includes 18 shift registers and 9 adders, allowing it to execute $Pif = 9$ input channels in parallel. In this situation, the sampling pixels and SBilinear coefficients play the roles of inputs and weights, respectively. For DwConv3D, we set $(Nr, Nc, Ng) = (Pif, Pks, 1) = (9, 9, 1)$ to parallel process channel and spatial dimensions of 3D kernels. The parameters for PwConv3D are set as $(Nr, Nc, Ng) = (Pof, Pif, 1) = (12, 9, 1)$ to unroll the input and output channel dimensions. They have the same microarchitecture of computing units, including $Piz = 7$ multipliers to parallel execute Piz element-wise multiplications.

C. Standard 3D Convolution Module

Conv3D-Module executes the separable convolutions in the offset generation stage of SDfConv3D in Fig. 2. It mainly

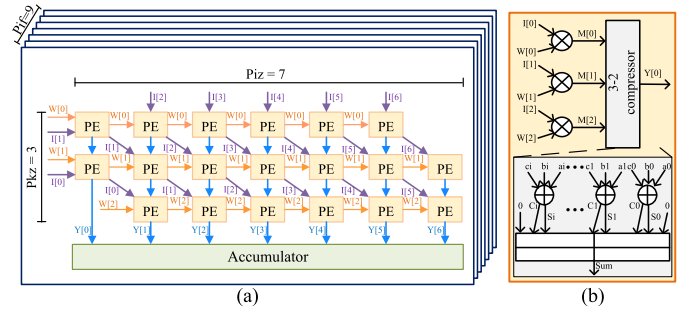


Fig. 10. Computational logic for DwConv3D in the Conv3D-Module. (a) Structure of the PE array. (b) Microarchitecture of the PE for DwConv3D.

has two types, including Conv3D-Module A and Conv3D-Module B, to compute a standard separable 3D convolution and a special version with the SPwConv3D layer, respectively. Specifically, a fast computing array for DwConv3D is designed to reduce the data movements and avoid padding abundant zero elements in the temporal dimension, which is shown in Fig. 10(a). Each PE array composes $Pif = 9$ processing element (PE) clusters, which are designed to process $Pif = 9$ input channels of DwConv3D in parallel. A PE cluster contains 19 PEs, and each PE takes charge of performing $Pky = 3$ multiply-accumulate (MAC) operations like Fig. 10(b). Besides, the weights are reused horizontally in each row, and $Piz = 7$ inputs in the temporal dimension are transferred to each cluster simultaneously and reused diagonally. Afterward, $Piz \times Pif = 7 \times 9$ psums are accumulated vertically across each column and directly streamed to the cascade PwConv3D layers. For PwConv3D and SPwConv3D, the computational logic is consistent with Fig. 9(a). Their computing units consist of $Piz = 7$ multipliers or shift registers to parallel perform element-wise multiplications in the temporal dimension.

D. Auxiliary Processing Core

APC is developed to perform the various remaining operations except for SDfConv3D in ALDNet, including a separable 3D convolution with one input channel and some separable 2D convolutions. In order to save hardware resources and match the speed of CPC, it executes in a time-division multiplexing manner to alternately achieve these two modes.

In terms of the memory system, an Input Buffer is reused to save the 3D or 4D input tensors. For 3D input tensors, each address naturally contains $Nif = 36$ channels for each pixel. For 4D input tensors, to unify the storage format of different convolutions, the process in the temporal dimension of a 3D kernel is changed to the input channel dimension. Fig. 11(a) provides a 1-D convolution as an example to demonstrate a 3D kernel with only one channel sliding along the temporal dimension. Since the stride of the 3D kernels equals one, the number of features in each pixel is flattened to $3 \times 7 = 21$. For the anchor-based shortcut, anchor pixels are preloaded to the corresponding addresses of Output Buffer during the execution of the last layer. In this way, anchor-based residual learning is performed by combining the anchor values and convolution

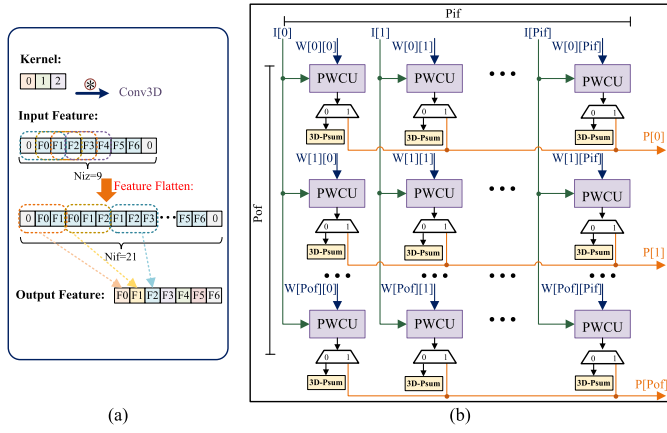


Fig. 11. Illustration of the APC. (a) Feature flatten. (b) Architecture of the reconfigurable computing array for PwConv3D and PwConv2D.

results. At last, the VSR results can be directly transferred to the same address without any memory rearrangement.

In terms of the computing system, we employ a row-stationary dataflow [39] to implement channel-wise processing for DwConv2D and process $P_{ox} = 4$ rows in parallel. To perform the PwConv3D and PwConv2D, a reconfigurable computing array is developed and illustrated in Fig. 11(b). It mainly has two computation modes configured by a switch flag. One mode is the execution of PwConv2D, whose disposal is to unroll the input and output channel dimensions. In another mode, since PwConv3D does not calculate input channel-wise accumulations, intermediate 3D psums at different positions are directly sent to 3D-Psum regfiles. Besides, to economize on the hardware resources, each computing unit of the pointwise convolutions (PwCU) only includes one multiplication.

V. EXPERIMENTAL RESULTS

A. Experimental Setup

We train the ALDNet on the public Vimeo-90k dataset [40]. When testing the model, multiple VSR datasets are employed for algorithm evaluations, such as Vid4 [41], SPMCS30 [42], and UDM10 [43]. Considering that SISR belongs to the extreme case of the VSR (the sequence length equals one frame), we also conduct experiments on publicly available SISR datasets, including Set5 [44], Set14 [45], BSDS100 [46], Urban100 [47], and Manga109 [48], and compare with the existing FPGA-based VSR methods.

The model is trained on NVIDIA Tesla P100 GPUs offline. Seven input frames are fed into the ALDNet to incorporate neighboring temporal information. The number of our model parameters is 39.7K. For the training phase, we optimize the model with the Adam method [49] under the PyTorch framework, and Charbonnier loss [50] is employed to deal with outliers. To convert values from the floating-point (FP) to fixed-point (FXP) format, weights and activations are quantized into 8 and 12 bits, respectively, based on a quantization-aware training (QAT) strategy [51]. All the experiments are performed with the upsampling scale factor of 2 and the overlapping size of 2. For quantitative analysis, we utilize peak signal-to-noise ratio (PSNR) and structural

TABLE I
RESOURCE USAGE OF OUR ACCELERATOR

Platform	Frequency	DSP	Logic Element	BRAM
Intel Stratix 10GX 2500	200MHz	1305 (26%)	311k(38%) in ALMs	2465 (24%)

similarity index (SSIM) [52] in the luminance channel of YCbCr color space as metrics to evaluate the difference between reconstructed SR and HR video sequences.

For the hardware acceleration, the overall design is described in Verilog HDL, and implemented on an Intel Stratix 10GX 2500 development board, which is composed of 821.1K adaptive logic modules (ALMs), 5,011 DSPs, and 10,202 M20K BRAM blocks. Each DSP block can perform a 32-bit FP multiplication or two 18-bit \times 19-bit FXP multiplications. We leverage Quartus Prime Pro 18.1 to synthesize, place, route our design, and estimate the total power by incorporating the CSV file into the power estimation tool. Specifically, the resource usage of our design is listed in Table I.

B. Storage and Computation Analysis

To explore the design space, the data interactions and computations are analyzed to optimize the design parameters (P^* and T^*) of loop unrolling and tiling. Generally, the inputs and weights are split into multiple tiles to meet the on-chip memory capacity. We employ the weight-stationary dataflow because of limited trainable parameters. According to the TDS, the channel dimension is not split by loop tilling. Thus, the latency (ms) of transferring one input tile from DRAM to the on-chip buffer can be expressed as:

$$T_{tile} = \frac{T_{ix} \times T_{iy} \times N_{iz} \times N_{if} \times \#Px_bits/8}{BW \times 10^{-6}}, \quad (11)$$

where BW is the bandwidth (GByte/s), $\#Px_bits$ denotes bit number per pixel. The tile number is $N_{tile} = \lceil N_{ix}/T_{ix} \rceil \times \lceil N_{iy}/T_{iy} \rceil$. To avoid the waste of hardware resources and time, the data transmission latency needs to be shorter than the computation latency, which means $T_{prepare}^{total} < T_{comp}^{total}$.

Since our design has multiple convolution modules, it is important to estimate the computation latency for each module individually in order to jointly optimize the P^* and T^* . For CPC, the computing cycle number per tile is formulated as:

$$\mathcal{N}_{cycle}^{CPC} = T_{ox} \times T_{oy} \times \lceil \frac{N_{if}}{P_{if}} \rceil \times \max\{N_{kx}, \lceil \frac{N_{of}}{P_{of}} \rceil\}, \quad (12)$$

where $\mathcal{N}_{cycle}^{CPC}$ also denotes the cycle number per period in Fig. 5(b). Since the CPC processes three tiles simultaneously in each period, and each tile needs to pass four SDFConv3Ds, the period number can be calculated based on \mathcal{N}_{tile}^I :

$$\mathcal{N}_{period} = \begin{cases} (\mathcal{N}_{tile} \times 4 + 2), & \mathcal{N}_{tile} = 3k \\ (\lceil \mathcal{N}_{tile}/3 \rceil + 1) \times 12, & \mathcal{N}_{tile} = 3k + 1 \\ (\lceil \mathcal{N}_{tile}/3 \rceil \times 12 + 13), & \mathcal{N}_{tile} = 3k + 2. \end{cases} \quad (13)$$

For APC, the computing cycles per tile for two modes become:

$$\mathcal{N}_{cycle}^{APC-0} = Toy \times \max\{Noz \times Nkz \times \lceil \frac{Tox}{Pox} \rceil, Tox \times \lceil \frac{Noz \times Nkz}{Pif} \rceil \times \lceil \frac{Nof}{Pof} \rceil\}, \quad (14)$$

$$\mathcal{N}_{cycle}^{APC-1} = Toy \times \max\{Nif \times \lceil \frac{Tox}{Pox} \rceil, Tox \times \lceil \frac{Nif}{Pif} \rceil \times \lceil \frac{Nof}{Pof} \rceil\}. \quad (15)$$

Finally, according to the above computing cycles per tile, tile number, and the data dependence in Fig. 5, the total computation latency to reconstruct one frame becomes:

$$\mathcal{T}_{comp}^{total} = \max\{\mathcal{T}_{comp}^{CPC}, \mathcal{T}_{comp}^{APC-0} + \mathcal{T}_{comp}^{APC-1}\}, \quad (16)$$

where \mathcal{T}_{comp}^{CPC} is the computation latency of CPC. $\mathcal{T}_{comp}^{APC-0}$ and $\mathcal{T}_{comp}^{APC-1}$ is the computation latency of APC in two modes.

Based on the analysis of storage and computations, a set of design parameters is determined in this work to ensure high computing resource utilization and processing speed. Such multidimensional constrained global optimization problems usually have multiple optimal solutions, reflecting the scalability of our design. Thus, one can also flexibly explore different P^* and T^* to satisfy the specified constraints of available hardware resources on different platforms.

C. Algorithm Comparison

In this section, to accurately evaluate the algorithm capacity of our design, some recently published FPGA-based SR methods are chosen for comparison, such as interpolation-based methods [6], [7], traditional machine learning-based method [2], and some deep learning-based methods, including [9], [10], and ERVSR [11].

1) *Quantitative Quality Comparison*: We conduct the experiments on VSR and SISR datasets. The quantitative metrics are summarized in Table II, from where we can see that our design achieves the best PSNR and SSIM scores on all benchmark datasets for both data formats. Since the traditional methods, i.e., [2], [6], and [7], are restricted by poor model fitting abilities, they suffer from worse PSNR and SSIM scores than the proposed method by a large margin. Compared with the previous deep learning-based methods, the quantization bitwidth of the proposed method is the lowest. Although the FXP representations reduce the scores slightly, the aligned spatio-temporal information helps to maintain the model's representation capacity. On the SISR datasets, although missing the critical temporal information, our design can still adaptively capture the geometric transformations of various objects, and dynamically adjust the spatial receptive field. In the same way, it can be observed that our design consistently surpasses all the competitors and achieves the best PSNR and SSIM scores, especially on the Manga109 dataset.

2) *Visual Effect Assessment*: Fig. 12 exhibits the visual effects of reconstructed HR video sequences and their cropped portions, using the bicubic interpolation and four other FPGA-based SR methods. From the zoom-in regions, it can be observed that our resulting VSR video sequences present sharp

edges and few artifacts, making them perceptually plausible. In particular, the majority of other works produce blurry, distorted, or deceptive artifacts. Nevertheless, our design can successfully recreate the building structures, the alphabet patterns, and the vertical pillars with outstanding visual quality. Thus, the proposed method preserves finer texture details and suppresses the noise, making the visual effect most similar to the reference video sequences in Fig. 12(f).

Assessing temporal consistency for VSR systems is essential, since it influences the visual perception of the video sequences. To demonstrate the effectiveness of keeping the temporal consistency, we analyze the temporal profile [53] of several video streams in Fig. 13. Flickering artifacts are presented as jitters and jagged lines in the profiles, while strong consistency indicates sharp and continuous profiles. More specifically, a temporal profile is produced from the same horizontal row of a selected region in each frame, and all the retrieved rows are stacked vertically along the temporal dimension. As shown in Fig. 13, since other methods either reconstruct each frame independently or lack the MEMC stage, they are unsuitable for establishing long-term temporal dependence and thus indicate flickering artifacts. In contrast, the temporal alignment introduced in our design provides pleasant visual perception with sharper structures and alphabets of the car. Thus, it intrinsically boots the smoothness of the temporal transition in the reconstructed video frames.

3) *Ablation Study of the ALDNet*: The number of frames, channels, and epochs are essential hyperparameters that affect the training efficiency and model capacity. Fig. 14(a) shows that increasing the frame number is beneficial to merging much temporal information and improving reconstruction quality. Fig. 14(b) demonstrates the PSNR score for different channels. Although the VSR quality promotes as the channel number rises, many parameters and computations are also introduced. Thus, to pursue the trade-off between quality and computations, we set the numbers of frames and channels are 7 and 36. Besides, the PSNR score and loss of each epoch are recorded in Fig. 15, which can ensure the convergence of our method.

D. Hardware Comparison

Finally, we compare the proposed accelerator with previous FPGA-based works and a widely-used commercial GPU product (RTX 2080Ti). A popular VSR dataset, Vid4, is employed to evaluate the hardware performance comprehensively. Moreover, some essential metrics are mainly considered, such as total power, throughput (GOPS), and energy efficiency (GOP/J). To the best of our knowledge, most existing SR accelerators are specially designed based on CNNs or DeCNNs, which ignore inter-frame alignment and incur ambiguous visual perception and flickering artifacts. This is **the first work** to develop an efficient **deformable 3D-CNN accelerator for high-quality VSR**. As a result, quantitative and qualitative comparisons with some SR accelerators based on CNNs [9] and DeCNNs [10], some dedicated DCN accelerators [23], [24], [26], [28], [30], and our design are carried out and summarized in Table III and Table IV.

As shown in Table III, the proposed design provides $11.67\times$ better throughput and $256.46\times$ better energy

TABLE II
QUANTITATIVE COMPARISON METRICS (PSNR/SSIM) FOR OUR MODELS AND 6 TESTED METHODS. FOR EACH COLUMN,
WE SHOW THE BEST RESULTS IN BOLD AND THE SECOND UNDERLINED

		VSR Datasets				SISR Datasets					
		Vid4	SPMCS30	UMD10	Average	Set5	Set14	BSDS100	Urban100	Manga109	Average
Method	(W,A)	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSMR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSMR/SSIM	PSNR/SSIM	PSMR/SSIM
Bicubic [6]	FP	28.42/0.8702	32.31/0.9198	38.42/0.9709	33.05/0.9203	33.65/0.9333	29.92/0.8666	29.47/0.8477	26.53/0.8417	28.27/0.9122	29.57/0.8803
Yang [7]	FP	-	-	-	-	34.00/-	29.97/-	-	-	-	31.99/-
	FXP	-	-	-	-	33.83/-	29.77/-	-	-	-	31.80/-
Kim [2]	FXP	-	-	-	-	34.78/0.9460	31.63/0.9083	30.48/0.8776	-	-	32.30/0.9106
Kim [9]	FP	30.62/0.9169	35.16/0.9527	42.08/0.9840	35.95/0.9512	36.67/0.9552	32.51/0.9076	<u>31.33/0.8887</u>	29.30/0.8954	35.37/0.9676	33.04/0.9229
	(10,14)	-	-	-	-	36.64/0.9543	32.47/0.9070	31.31/0.8877	<u>29.32/0.8939</u>	-	32.44/0.9107
Chang [10]	FP	30.63/0.9159	34.98/0.9501	41.67/0.9823	35.76/0.9494	36.47/0.9532	32.38/0.9059	31.23/0.8873	29.05/0.8891	35.03/0.9639	32.83/0.9199
	(13,13)	-	-	-	-	36.40/0.9527	32.21/0.9047	31.15/0.8858	-	-	<u>33.25/0.9144</u>
ERVSR [11]	FP	<u>30.99/0.9222</u>	<u>35.30/0.9535</u>	<u>42.40/0.9843</u>	<u>36.23/0.9533</u>	<u>36.80/0.9556</u>	<u>32.53/0.9079</u>	<u>31.32/0.8891</u>	<u>29.32/0.8957</u>	<u>35.83/0.9684</u>	33.16/0.9233
	(12,16)	30.94/0.9209	35.26/0.9529	42.32/0.9840	36.17/0.9526	36.76/0.9553	32.51/0.9076	31.31/0.8887	29.30/0.8952	35.78/0.9682	33.13/0.9230
ALDNet	FP	31.66/0.9300	35.75/0.9578	42.65/0.9861	36.69/0.9580	37.08/0.9591	32.81/0.9125	31.42/0.8939	29.39/0.9064	36.42/0.9715	33.42/0.9287
	(8,12)	31.63/0.9300	35.74/0.9578	42.58/0.9859	36.65/0.9579	37.06/0.9590	32.80/0.9125	31.41/0.8938	29.38/0.9063	36.41/0.9714	33.41/0.9286



Fig. 12. $2\times$ scale VSR result images of the proposed method and other SR methods which are deployed on FPGA. (a) Result of bicubic interpolation [6]. (b) Result of Kim et al. [9]. (c) Result of Chang et al. [10]. (d) Result of ERVSR [11]. (e) Result of the proposed method. (f) Ground truth.

efficiency than the GPU. Besides, our design achieves $2.75\times$ throughput and $1.63\times$ energy efficiency improvement than other FPGA-based hardware accelerators. Besides, our design attains better throughput, but lower power consumption than CNN-based or DeCNN-based SR accelerators. It should be noted that, compared with these prior works, although our design uses comparable or more hardware resources, the acceleration of sophisticated deformable 3D

convolutions and high-quality video processing tasks is also implemented.

As shown in Fig. 16(a), compared with a traditional multiplication-based method, 73.35% of the multiplications are cut down by the proposed SDfConv3D, which helps the limited DSP resources be allocated reasonably to some computing modules with high complexity. By this means, DfConv3D-Module occupies 40% DSPs to perform the

TABLE III
COMPARISONS WITH EXISTING FPGA-BASED ACCELERATION SOLUTIONS

	[26]	[30]	[28]	[24]	GPU	[9]	[10]	[23]	Ours
Year	2021	2021	2022	2022	-	2019	2020	2022	2022
Task	Object Detection			Denoising	Image/Video Super-Resolution				
Model	CodeNet	CenterNet	CenterNet	DeformJDD	ALDNet	CNN	FSRCNN	D3Dnet	ALDNet
Platform	ZYNQ UltraScale+	Virtex-7	Intel Arria 10	ZYNQ UltraScale+	RTX 2080Ti	Kintex UltraScale	Kintex-7	Virtex UltraScale+	Intel Stratix 10GX 2500
Technology(nm)	20	28	20	16	12	20	28	20	14
Methodology	HLS	Verilog HDL	Verilog HDL	Verilog HDL	PyTorch	Verilog HDL	Verilog HDL	Verilog HDL	Verilog HDL
Frequency(MHz)	250	200	150	250	1350	150	130	200	200
Precision(W-A bit)	FXP 4-8	FXP 8-8	FXP 8-16	FXP 8-8	FP 32	FXP 10-14	FXP 13-13	FXP 8-10	FXP 8-12
DSP Usage	360	297	1518	2304	-	1920	1512	1539	1305
Logic Utilization	34.1k(LUTs) 41.8k(FFs)	16.7k(LUTs) 27.7k(FFs)	274.0k (in ALMs)	205.0k(LUTs) 293.5k(FFs)	-	151.0k(LUTs) 121.0k(FFs)	167.0k(LUTs) 158.0k(FFs)	158.0k(LUTs) 319.5k(FFs)	311.2k (in ALMs)
BRAM Usage(Mbyte)	0.972(100.00%)	0.520(6.14%)	5.888(87.00%)	2.448(59.65%)	-	0.194(7.19%)	0.945(26.42%)	8.120(83.54%)	6.018(24.16%)
Power(W)	5.60	2.51	-	-	210.54	5.69	5.40	-	9.60
Throughput (GOPS)	72.00	126.32	822.31	529.92	194.04 (94.55 [†])	639.57 [‡]	780.00	295.29	2265.07 (1105.60 [‡])
Energy Efficiency (GOP/J)	12.86	50.33	-	-	0.92 (0.45 [†])	112.48	144.90	-	235.94 (115.17 [†])

[†]The shift-and-add operations which replace the corresponding multiplications of SDFConv3D are not included when computing these metrics.

[‡]The throughput is computed based on MAC operations and average computation time provided by [9].

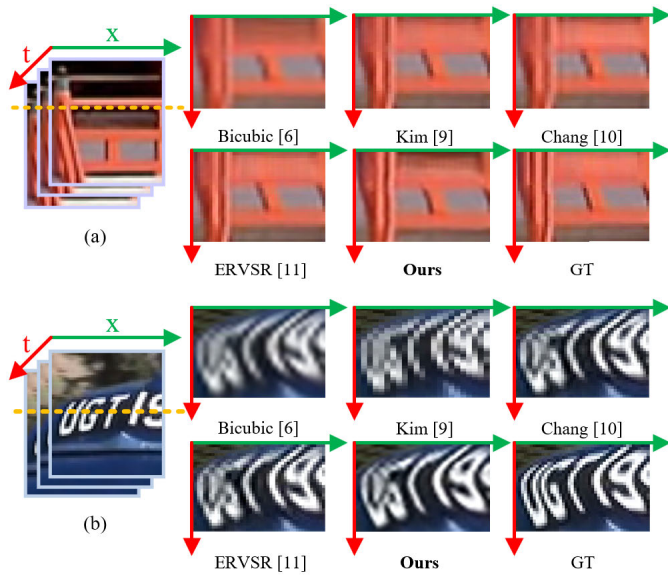


Fig. 13. Comparisons of temporal profile. (a) SPMCS30 - car05-001. (b) SPMCS30 - jvc-009-001.

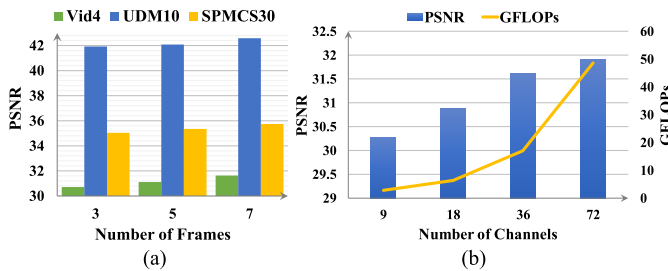


Fig. 14. Analysis of the proposed ALDNet under different hyperparameters. (a) Number of frames. (b) Number of channels.

deformable convolution in our design. 95% DSPs that we use are allocated to CPC to mainly accelerate the implicit feature alignment stage of ALDNet, which is shown in Fig. 16(b).

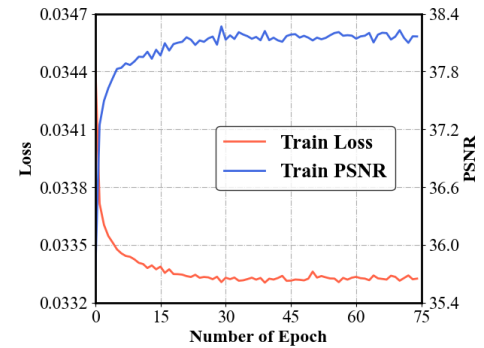


Fig. 15. Learning curve of the proposed model.

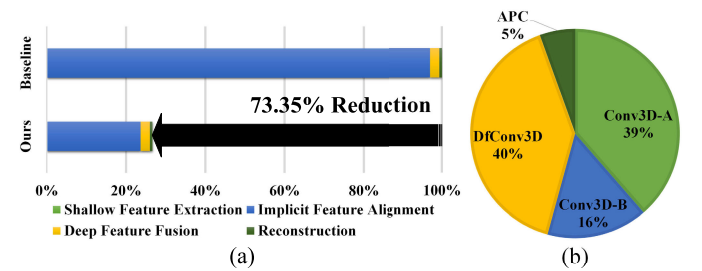


Fig. 16. Analysis of computing resources. (a) The reduction of performed operations. (b) DSP ratio of different modules.

At last, the frame rate can reach 32 FPS, which guarantees our design to satisfy the requirement of real-time VSR.

These improvements mainly originate from two aspects. At the algorithm level, we introduce a SPwConv3D and SBilinear layer in the SDfConv3D and maintain the offset diversity, significantly relieving the computing resource overhead caused by intensive interpolations and boosting energy efficiency. Meanwhile, the TDS alleviates excessive memory requirements resulted from the deformed receptive field and large input resolution. At the hardware level, the block-level

TABLE IV
QUALITATIVE COMPARISONS OF VARIOUS DEFORMABLE
CONVOLUTION ACCELERATION STRATEGIES

Metrics	Deformable Shape	Flexible Offset	Separable Convolution	Pixel Processing Task
[25], [26]	Square	N	Y	N
[27], [28]	Arbitrary	N	N	N
[29], [30]	Arbitrary	Y	N	N
[24]	Arbitrary	N	Y	Y
Ours	Arbitrary	Y	Y	Y

interleaving storage scheme prevents memory access conflicts and sophisticated control logic, improving resource utilization. Besides, the constructed critical processing core and auxiliary processing core with a high degree of parallelism and flexible scheduling pattern contribute to high hardware efficiency.

Table IV presents some qualitative comparisons with prior DCN accelerators [24], [25], [26], [27], [28], [29], [30]. For example, due to the constrained square respective fields [25] and [26], all potential activations can be stored with various line buffers, but this memory arrangement is inappropriate for arbitrary-shape sampling patterns. Besides, although bounding the diversity and range of the offsets like [24], [27], and [28] can prevent memory access conflicts, these simplifications will block the extraction of diverse optical flow and accurate estimation of motion vectors. References [29] and [30] can maintain the offset flexibility and retrieve the sampled inputs regularly, but they can not support separable deformable 3D convolutions and pixel processing tasks with large data volumes. However, these problems can be completely resolved by the proposed accelerator, since our design can not only guarantee high-speed executions of the separable deformable 3D convolutions with flexible and complete offsets, but also be applicable for pixel processing tasks with large feature resolutions like VSR.

VI. CONCLUSION

In this work, we have proposed, developed, and validated an algorithm-hardware co-optimization framework to accelerate the deformable 3D-CNN for the VSR tasks. At the algorithm level, a hardware-inspired model, called ALDNet, together with an innovative SDFConv3D, is proposed to capture spatio-temporal information from the aligned features. At the hardware level, a critical processing core incorporating a block-level interleaving storage scheme is designed to avoid memory access conflicts caused by the deformable convolutions. Finally, an overall hardware architecture is implemented on an Intel Stratix 10GX platform, providing superior visual perception in the VSR tasks. Meanwhile, compared with prior accelerators, our design demonstrates much better results in terms of throughput and energy efficiency.

For future research, this work can be further investigated and generalized to more interesting real-time video understanding tasks, such as video compression, action recognition, and video restoration.

REFERENCES

- [1] H. Seibel, S. Goldenstein, and A. Rocha, "Eyes on the target: Super-resolution and license-plate recognition in low-quality surveillance videos," *IEEE Access*, vol. 5, pp. 20020–20035, 2017.
- [2] Y. Kim, J.-S. Choi, and M. Kim, "2X super-resolution hardware using edge-orientation-based linear mapping for real-time 4K UHD 60 fps video applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 9, pp. 1274–1278, Sep. 2018.
- [3] Y. Luo, L. Zhou, S. Wang, and Z. Wang, "Video satellite imagery super resolution via convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 12, pp. 2398–2402, Dec. 2017.
- [4] W. Wen, W. Ren, Y. Shi, Y. Nie, J. Zhang, and X. Cao, "Video super-resolution via a spatio-temporal alignment network," *IEEE Trans. Image Process.*, vol. 31, pp. 1761–1773, 2022.
- [5] H. Liu et al., "Video super resolution based on deep learning: A comprehensive survey," *Artif. Intell. Rev.*, vol. 55, no. 8, pp. 5981–6035, Apr. 2022.
- [6] H. S. Hou and H. Andrews, "Cubic splines for image interpolation and digital filtering," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-26, no. 6, pp. 508–517, Dec. 1978.
- [7] M.-C. Yang, K.-L. Liu, and S.-Y. Chien, "A real-time FHD learning-based super-resolution system without a frame buffer," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 12, pp. 1407–1411, Dec. 2017.
- [8] X. Wang, K. C. K. Chan, K. Yu, C. Dong, and C. C. Loy, "EDVR: Video restoration with enhanced deformable convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1954–1963.
- [9] Y. Kim, J.-S. Choi, and M. Kim, "A real-time convolutional neural network for super-resolution on FPGA with applications to 4K UHD 60 fps video services," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 8, pp. 2521–2534, Aug. 2019.
- [10] J.-W. Chang, K.-W. Kang, and S.-J. Kang, "An energy-efficient FPGA-based deconvolutional neural networks accelerator for single image super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 1, pp. 281–295, Jan. 2020.
- [11] K. Sun, M. Koch, Z. Wang, S. Jovanovic, H. Rabah, and S. Simon, "An FPGA-based residual recurrent neural network for real-time video super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 4, pp. 1739–1750, Apr. 2022.
- [12] J. Dai et al., "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 764–773.
- [13] G. Bertasius, L. Torresani, and J. Shi, "Object detection in video with spatiotemporal sampling networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 342–357.
- [14] Y. Zhao, Y. Xiong, and D. Lin, "Trajectory convolution for action recognition," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2018, pp. 2208–2219.
- [15] Y. Tian, Y. Zhang, Y. Fu, and C. Xu, "TDAN: Temporally-deformable alignment network for video super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3357–3366.
- [16] H. Wang, D. Su, C. Liu, L. Jin, X. Sun, and X. Peng, "Deformable non-local network for video super-resolution," *IEEE Access*, vol. 7, pp. 177734–177744, 2019.
- [17] X. Ying, L. Wang, Y. Wang, W. Sheng, W. An, and Y. Guo, "Deformable 3D convolution for video super-resolution," *IEEE Signal Process. Lett.*, vol. 27, pp. 1500–1504, 2020.
- [18] K. C. K. Chan, S. Zhou, X. Xu, and C. C. Loy, "BasicVSR++: Improving video super-resolution with enhanced propagation and alignment," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5962–5971.
- [19] J. Chen, X. Tan, C. Shan, S. Liu, and Z. Chen, "VESR-Net: The winning solution to Youku video enhancement and super-resolution challenge," 2020, *arXiv:2003.02115*.
- [20] K. C. Chan, X. Wang, K. Yu, C. Dong, and C. C. Loy, "Understanding deformable alignment in video super-resolution," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2021, pp. 973–981.
- [21] Y. Ma, Y. Cao, S. Vrudhula, and J.-S. Seo, "Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, Feb. 2017, pp. 45–54.
- [22] Z. Wang, J. Lin, and Z. Wang, "Accelerating recurrent neural networks: A memory-efficient approach," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2763–2775, Oct. 2017.
- [23] S. Zhang, W. Mao, and Z. Wang, "An efficient accelerator of deformable 3D convolutional network for video super-resolution," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2022, pp. 110–115.
- [24] J. Guan et al., "Memory-efficient deformable convolution based joint denoising and demosaicing for UHD images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 11, pp. 7346–7358, Nov. 2022.

- [25] Q. Huang et al., "Algorithm-hardware co-design for deformable convolution," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2019, pp. 48–51.
- [26] Q. Huang et al., "CoDeNet: Efficient deployment of input-adaptive object detection on embedded FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2021, pp. 206–216.
- [27] S. Ahn, J.-W. Chang, and S.-J. Kang, "An efficient accelerator design methodology for deformable convolutional networks," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 3075–3079.
- [28] S. Li, S. Cao, L. Hui, Z. Jiang, Y. Sun, and S. Xu, "A computational-efficient deformable convolution network accelerator via hardware and algorithm co-optimization," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Nov. 2022, pp. 1–6.
- [29] D. Xu et al., "Energy-efficient accelerator design for deformable convolution networks," 2021, *arXiv:2107.02547*.
- [30] Y. Yu, J. Luo, W. Mao, and Z. Wang, "A memory-efficient hardware architecture for deformable convolutional networks," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Oct. 2021, pp. 140–145.
- [31] K.-N. Mac, D. Joshi, R. Yeh, J. Xiong, R. Feris, and M. Do, "Learning motion in feature space: Locally-consistent deformable convolution networks for fine-grained action detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6281–6290.
- [32] M. Chang, Q. Li, H. Feng, and Z. Xu, "Spatial-adaptive network for single image denoising," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Aug. 2020, pp. 171–187.
- [33] H. Wu et al., "Contrastive learning for compact single image dehazing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10546–10555.
- [34] T. Manabe, Y. Shibata, and K. Oguri, "FPGA implementation of a real-time super-resolution system using a convolutional neural network," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2016, pp. 249–252.
- [35] J. Lee, J. Lee, and H.-J. Yoo, "SRNPU: An energy-efficient CNN-based super-resolution processor with tile-based selective super-resolution in mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 10, no. 3, pp. 320–334, Sep. 2020.
- [36] Z. Du, J. Liu, J. Tang, and G. Wu, "Anchor-based plain net for mobile image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 2494–2502.
- [37] M. Elhoushi, Z. Chen, F. Shafiq, Y. H. Tian, and J. Y. Li, "DeepShift: Towards multiplication-less neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 2359–2368.
- [38] D. A. Gudovskiy and L. Rigazio, "ShiftCNN: Generalized low-precision architecture for inference of convolutional neural networks," 2017, *arXiv:1706.02393*.
- [39] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.
- [40] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *Int. J. Comput. Vis.*, vol. 127, no. 8, pp. 1106–1125, Feb. 2019.
- [41] C. Liu and D. Sun, "On Bayesian adaptive video super resolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 346–360, Feb. 2014.
- [42] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia, "Detail-revealing deep video super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4482–4490.
- [43] P. Yi, Z. Y. Wang, K. Jiang, J. J. Jiang, and J. Y. Ma, "Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3106–3115.
- [44] M. Bevilacqua, A. Roumy, C. Guillemot, and M. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, Sep. 2012, pp. 1–10.
- [45] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. 7th Int. Conf. Curves Surf.*, Jun. 2010, pp. 711–730.
- [46] D. R. Martin, C. C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Jul. 2001, pp. 416–425.
- [47] Y. Huang, W. Wang, and L. Wang, "Bidirectional recurrent convolutional networks for multi-frame super-resolution," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2015, pp. 235–243.
- [48] K. Aizawa et al., "Building a Manga dataset 'Manga109' with annotations for multimedia applications," *IEEE Multimedia Mag.*, vol. 27, no. 2, pp. 8–18, Apr. 2020.
- [49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2015, pp. 1–15.
- [50] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Proc. IEEE Int. Conf. Inf. Process. (ICIP)*, vol. 2, Nov. 1994, pp. 168–172.
- [51] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 2704–2713.
- [52] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
- [53] J. Caballero et al., "Real-time video super-resolution with spatio-temporal networks and motion compensation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2848–2857.



Siyu Zhang received the M.S. degree in information and communication engineering from the Harbin Institute of Technology, Harbin, China, in 2021. He is currently pursuing the Ph.D. degree in information and communication engineering with Nanjing University, Nanjing, China. His current research interests include deep learning model acceleration and algorithm-hardware co-design for image/video processing. He received the IEEE ISVLSI 2022 Best Paper Award.



Wendong Mao received the B.S. degree in information engineering from Jilin University, Changchun, China, in 2018. She is currently pursuing the Ph.D. degree in information and communication engineering with Nanjing University, China. She was a Visiting Student with the Wangxuan Institute of Computer Technology, Peking University, Beijing, in 2019. Her current research interests include image processing algorithm and efficient hardware design for deep learning. She received the IEEE ISVLSI 2022 Best Paper Award.



Zhongfeng Wang (Fellow, IEEE) received the B.E. and M.S. degrees from the Department of Automation, Tsinghua University, Beijing, China, in 1988 and 1990, respectively, and the Ph.D. degree from the University of Minnesota, Minneapolis, in 2000.

He was a leading VLSI Architect with Broadcom Corporation, San Jose, CA, USA, from 2007 to 2016. Before that, he worked with Oregon State University and National Semiconductor Corporation.

He is a world-recognized expert on low-power high-speed VLSI design for signal processing systems. He has been a Distinguished Professor with Nanjing University, China, since 2016. He has published more than 200 technical papers with multiple best paper awards received from the IEEE technical societies, among which is the VLSI Transactions Best Paper Award in 2007. He has edited one book VLSI and held more than 20 U.S. and Chinese patents. In the current record, he has had many papers ranking among top 25 most (annually) downloaded manuscripts in IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS. Moreover, he has contributed significantly to the industrial standards. So far, his technical proposals have been adopted by more than 15 international networking standards. His current research interests include optimized VLSI design for digital communications and deep learning. He was a TPC member and various chairs for tens of international conferences. In 2015, he was elevated to the Fellow of IEEE for contributions to VLSI design and implementation of FEC coding. He has served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, and IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS for many terms.