Co-Via: A Video Frame Interpolation Accelerator Exploiting Codec Information Reuse

Haishuang Fan[1,2], Qichu Sun[1,2], Jingya Wu[1,*], Wenyan Lu[1,3], Xiaowei Li[1], Guihai Yan[1,3,*]
{fanhaishuang20z, sunqichu22z, wujingya, luwenyan, lxw, yan}@ict.ac.cn
SKLP, Institute of Computing Technology, Chinese Academy of Sciences[1]
University of Chinese Academy of Sciences[2], *YUSUR Technology Co., Ltd.*[3], *corresponding author*[*]

ABSTRACT

Video Frame Interpolation (VFI) aims to generate intermediate frames between consecutive frames. Recent DNN-based VFI offers superior quality but suffers from performance issues. However, very few studies have focused on VFI hardware acceleration and existing work overlooks temporal information from compressed video bitstreams. In this paper, we propose a novel compressed VFI workflow and an accelerator, Co-Via. Co-Via exploits codec information reuse to reduce complex DNN computations and alleviate hardware pressure. FPGA-based Co-Via outperforms an RTX 4090 GPU 10.31×, offering a 43.08× energy efficiency boost. Its ASIC version achieves 2.4× higher throughput and 3.6× energy efficiency than the state-of-the-art solution.

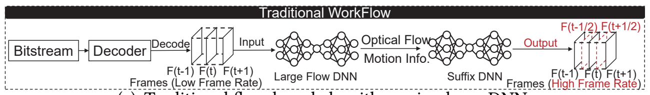
KEYWORDS

Video Frame Interpolation, Codec, Accelerator, FPGA

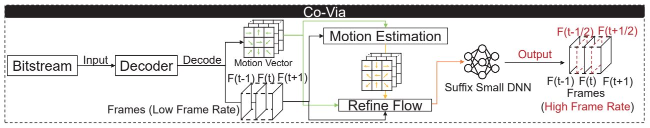
1 INTRODUCTION

Video frame interpolation (VFI) is a fundamental computer vision technique to enhance the temporal resolution of an input video by generating non-existent intermediate frames between consecutive frames. It finds applications in various areas such as frame rate upconversion [1] and text-to-video generation [12]. In recent years, deep learning has emerged as a highly successful approach for VFI [7, 8], thanks to its ability to produce superior quality compared to traditional interpolation methods [4]. However, the increasing complexity of deep neural network (DNN)-based VFI models presents challenges, such as high computational demands and memory bandwidth requirements. Therefore, hardware acceleration becomes crucial in achieving high-throughput VFI systems.

Most prior DNN accelerator studies [2, 5, 14] predominantly focus on object detection, video recognition, and super-resolution tasks, with little attention to VFI acceleration. VISTA [9] and FIACCEL [6] are the only two works implementing CNN accelerators for VFI. However, they primarily accelerate kernel-based VFI and adopt a per-frame inference, disregarding the temporal information between video frames, resulting in performance and quality constraints. Recent developments have seen the emergence of numerous flow-based VFI methods [7, 8], which utilize motion information and have achieved prominence in referenced research. The flow-based VFI workflow is depicted in Fig.1(a). In real-life applications such as live video streaming, videos are often compressed into



(a) Traditional flow-based algorithm using large DNN



(b) Proposed codec information reuse algorithm using small DNN

Figure 1: Deep leaning workflow for VFI

compact bitstreams for efficient storage and transmission, incorporating abundant temporal-spatial information. However, prior VFI algorithms and accelerators only consider uncompressed frames as input and lack codec engines for processing the compressed data. Consequently, the traditional workflow requires decoding the condensed bitstreams to reconstruct raw frames ($F(t-1)$, $F(t)$, $F(t+1)$ in Fig.1(a)) and then employing a large DNN to obtain optical flow. This motion information is subsequently transmitted to a suffix DNN to synthesize the interpolated results (red dashed frames $F(t \pm 1/2)$ in Fig.1(a)).

The above workflow is a de facto standard process but overlooks the valuable temporal information that can be extracted from compact bitstreams. This temporal information, especially motion vectors (MVs), reflects motion information and shares semantic similarity with optical flows. Moreover, our analysis of a typical VFI DNN, RRIN [7], reveals that the optical-flow DNN consumes more than 60% time. Therefore, obtaining temporal information from the bitstreams and utilizing it as a replacement for optical flow can save substantial DNN computation. However, this crucial observation has long been ignored in state-of-the-art VFI solutions. On the one hand, traditional VFI flows drop the temporal information after decoding, and conventional VFI DNN models [7, 8] are solely designed for processing raw frames. On the other hand, traditional DNN accelerators, such as VISTA [9], lack an engine to decode compressed bitstreams and require additional software codecs. This not only introduces extra latency but also incurs increased energy costs. Therefore, a direct VFI accelerator for compressed videos becomes imperative to shorten the critical path in VFI.

Based on the above observations, we propose a novel compressed video VFI workflow and an accelerator for the workflow called Co-Via, as shown in Fig 1(b). Co-Via seamlessly integrates the decoder and DNN engines. Unlike traditional methods, Co-Via can directly perform VFI on compressed videos, skip significant DNN computation by reusing codec information, and alleviate hardware design constraints. Our contributions can be summarized as follows:

1) To complete the VFI conversion of a compressed video bitstream, we propose a novel VFI workflow that skips extensive DNN computation. Inspired by the fact that MVs are similar to optical

DAC '24, June 23–27, 2024, San Francisco, CA, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3655669

flows, we reuse MVs from the codec information to quickly reconstruct interpolated frames with a lightweight suffix DNN. However, directly reusing codec information results in apparent quality loss. To address this issue, we utilize another lightweight DNN to refine MVs for capturing high-frequency information and exploit bidirectional MVs for more accurate motion estimation.

2) We design an accelerator for direct compressed VFI called Co-Via. It integrates a decoder, a motion estimation (ME) module, and a DNN engine. The decoder extracts decoding information and raw frames, while ME computes backward MVs by reusing forward MVs from the decoder to reduce computation and hardware complexity. Based on U-Net modules, the DNN engine performs optical flow refinement and frame generation. Given the varying computation and memory access characteristics across different U-Net layers, we divide the U-Net networks into three groups and employ heterogeneous cores for each to improve hardware utilization. Moreover, we observe that Deconvolution operations in U-Net entail redundant memory access and computation, leading us to reformulate them into Convolution to eliminate this redundancy.

3) We leverage tile-level parallel pipeline scheduling to enhance system throughput. Although Co-Via directly integrates Decoder, ME, and DNN engines, running them in a purely serial mode introduces numerous pipeline bubbles and degrades performance. To overcome this, we parallelize the Decoder and DNN engine and synchronize the computation flow across different modules. To further enhance throughput, we employ a tile-based approach to dividing a frame into multiple tiles and allowing subsequent computation to start immediately after decoding.

Experiment results demonstrate that FPGA-based Co-Via achieves $8.65 \sim 11.69\times$ performance improvement over an RTX 4090 GPU and $38.11 \sim 46.79\times$ energy efficiency. Its ASIC version achieves $2.4\times$ throughput (4K@120fps) and $3.6\times$ energy efficiency improvement than SOTA [9]. To our knowledge, this is the first accelerator for flow-based VFI and the first unified decoder and DNN accelerator for VFI acceleration. By reusing decoding information, Co-Via accelerates the crucial optical flow generation process and saves a substantial amount of DNN computation.

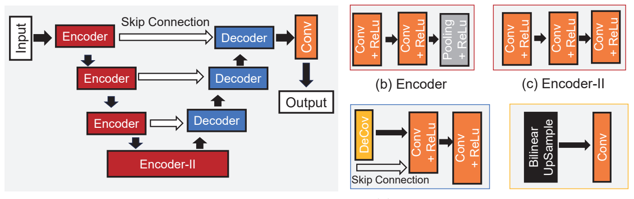
2 BACKGROUND

2.1 Video Codec

Video codec exploits redundancy within and across frames to compress video into fewer bytes. The video encoder selects a reference macroblock (MB) from the reference frame for MBs in the current frame, based on the minimum sum of absolute error (SAE), and computes their pixel differences called residuals, which is referred to as motion estimation (ME). The position bias between the reference and current MB is the motion vector (MV), a crucial temporal parameter. After predicting MVs, the residuals and MVs undergo a series of compression steps to produce a compact video bitstream, including transformation, quantization, and entropy coding. A quantization parameter (QP) is configured to achieve a trade-off between video quality and bit rate. The video decoder can recover raw frames and MVs from these bitstreams. This paper takes H.264/AVC [16], the most-applied codec standard, as an example.

2.2 Video Frame Interpolation Algorithm

Video frame interpolation (VFI) aims to synthesize non-existent intermediate frames between consecutive frames. It is widely used in various applications, including frame rate-up conversion and text-to-video generation. Recent works have achieved the start-of-art accuracy with advances in DNN [3, 7, 8]. DNN-based methods can be classified into kernel-based methods [3] and flow-based methods [7, 8]. The quality and computation overhead of kernel-based methods are severely constrained by the kernel size. Flow-based methods have recently been predominant in referenced research and adopt a two-step solution: 1) generating the optical flow based on input frames and 2) warping input frames to synthesize new frames based on motion information. However, most DNN models suffer from numerous parameters and cannot meet the throughput and energy efficiency requirements in real-life VFI systems.



(a) The Unet Architecture (b) Encoder (c) Encoder-II (d) Decoder (e) Deconvolutional Layer

Figure 2: The network architecture of U-Net

RRIN. To accelerate VFI, RRIN [7] adopts a modified U-Net [11] with reduced depths and parameters to implement DNN sub-modules. These sub-modules encompass optical-flow generation, frame combination, and frame generation. RRIN first estimates the bi-directional optical flow. Subsequently, the flow is refined through residue learning. RRIN uses this refined flow to warp two input frames. The linear combination of two warped frames uses a weight map learned by a U-Net. Once the coarse frame is obtained, RRIN continues using residue learning for further refinement and outputs the final frame. This study uses RRIN as the baseline flow-based algorithm.

U-Net. U-Net [11] is an important DNN structure in VFI and RRIN employs four U-Nets. In recent years, U-Net has been widely applied in multiple areas of image processing, like image segmentation. A U-Net module comprises three primary components: encoder, skip connection, and decoder, as shown in Fig.2(a)~(d). In the past, several FPGA accelerators [17] have been developed for U-Net to enhance image segmentation. However, it still lacks accelerators for VFI tasks. Unlike image segmentation, which uses a single U-Net, VFI demands multiple U-Net modules with different depths and channels. Therefore, FPGA accelerators for VFI need higher hardware complexity.

3 MOTIVATION AND ANALYSIS

We conduct RRIN for VFI tasks on an NVIDIA RTX 4090 GPU. Details of the system configuration and datasets used in this evaluation are given in Section 6.1. Fig.3(a) demonstrates that its performance on 4K datasets is less than 3FPS (frame per second) and falls short of the real-time 30FPS requirement for live streaming services. The inefficiency of RRIN and challenges in designing a VFI accelerator can be attributed to several factors:

1). Inefficient Optical Flow Generation: The time breakdown of RRIN, depicted in Fig.3(a), reveals that the optical-flow generation is the performance bottleneck, taking up more than 60% of time. This is reasonable because it has significantly more parameters, around 5 to 6 times more than the other two modules. After a thorough analysis of the optical-flow results, we discover that they reflect the motion relationships between video frames, and

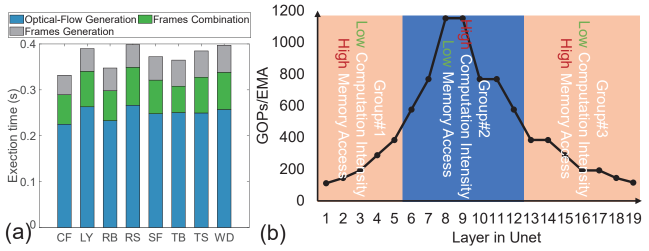


Figure 3: Performance and characterize analysis of RRIN. (a) The execution time breakdown of one 4K frame. (b) The computation and memory access of different layers in U-Net.

this motion information has a semantic similarity with motion vectors (MV) in video codecs. The key insight is that MV information already exists in the video bitstreams. Since our goal is to build a high-throughput compressed video VFI system, reusing the temporal information in the compressed domain is friendly to our system and provides a vital optimization to boost the performance. Moreover, the inherent continuity of motion relationships between video frames ensures output quality. To explore the new design space of the VFI accelerator based on parameter-reusing, we analyze the different reusing algorithms to get a better trade-off between performance and accuracy in Section 4.1.

2). **Redundant Deconvolution Operations:** Fig.2 shows that U-Net consists of Deconvolution (DeConv) and Convolution (Conv) layers. These two kinds of layers have different computation patterns, introducing the hardware overhead and design complexity. The U-Net in RRIN implements Deconv using the bilinear interpolation-based upsampling and a Conv layer, shown in Fig.2(e), to recover the fine-grained details. However, during the upsampled step, the interpolated values and their positions in output can be derived from the input feature map (FM). Therefore, the bilinear-based upsampling can be discarded and combined into Conv in hardware. This observation reveals that Deconv in U-Net has redundant computations and memory access. We discuss removing this redundancy and converting Deconvolution into a unified Convolution operation in Section 4.2.

3). **Variances in Computational and Memory Access Characteristics Across U-Net Layers:** One U-Net within RRIN consists of 19 layers, forming a sophisticated network structure. We analyzed the computational complexity (measured in GOP) and total external memory access (EMA) across different layers and calculated their ratios. The results, shown in Fig.3(b), emphasize the various characteristics in computational intensity and memory access across their layers, stemming from the differences in feature map size and channel counts. However, traditional AI engines often employ a uniform PE array with fixed dataflow and memory access patterns, leading to low hardware utilization when processing U-Net. To address this issue, we propose a practical solution involving specialized heterogeneous cores capable of efficiently handling layers with varying characteristics. To facilitate this, we divided U-Net layers into three groups, as shown in Fig.3(b). Group#1 and Group#3, characterized by large feature maps and fewer channels, have high memory demands but low computation intensity. In contrast, Group#2 has an inverse profile, with small feature maps and more channels. We design heterogeneous cores tailored to these three groups with further details in Section 5.5.

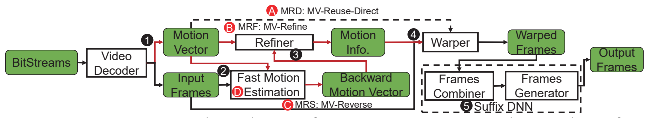


Figure 4: Proposed codec information reuse algorithm for video frame interpolation

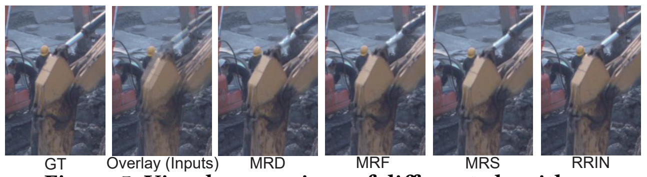


Figure 5: Visual comparison of different algorithms

4 OPTIMIZATION

4.1 Proposed Codec Information Reusing Algorithm for VFI

As shown in Fig.4, the core idea of the algorithm is to reuse decoding information and skip the optical flow generation to accelerate VFI. We present a baseline algorithm, MRD, followed by two improved algorithms, MRF and MRS.

Ⓐ **MV-Reuse-Direct (MRD):** Initially, we obtain MVs from the decoder and use them as the optical flow in VFI. However, MRD introduces a significant 2.06dB quality loss compared to RRIN on the 4K CF video sequence, as depicted in Fig.6(a).

Ⓑ **MV-Refine (MRF):** MRD neglects high-frequency motion details between video frames and degrades the quality. To address this, we utilize a lightweight U-Net to refine MVs. The model parameters for refinement are only 1/6 of Optical-Flow U-Net. Fig.6(a) shows that MRF improves the quality by 0.78 dB compared to MRD.

Ⓒ **MV-Reverse (MRS)**: Despite the improvements from MRF, it still has a 1.28dB quality loss compared to RRIN. We discovered that MVs in codec are unidirectional, representing motion from the previous frame to the next, which can be called forward MVs. In the prior two algorithms, we simply inverted forward MVs to represent MVs from the next frame to the previous frame (backward MVs). However, due to the inherent motion asymmetry, these two MVs exhibit differences. To acquire bidirectional motion information, we use motion estimation (ME) with the next frame as a reference to compute backward MVs for the previous frame. With bidirectional motion vectors, MRS has a 1.47 dB improvement in video quality, resulting in only a 0.21 dB loss compared to RRIN shown in Fig.6(a). Also, as seen in Fig 5, the texture details of MRS are very close to RRIN, so this quality loss is negligible.

Ⓓ **Fast-Motion-Estimation (Fast-ME):** ME searches in the reference frame to find the reference MB for a current MB based on a large search window shown in Fig.7(a)-I. This method incurs significant computational and memory overhead. Given that we already have forward MVs and that forward and backward MVs exhibit similarity, we introduce a fast-ME technique based on reusing forward MVs, as depicted in Fig.7(a)-II. We use the forward MVs (MV_{fw}) as the midpoint for the search window and reduce the window size from 128x128 to 16x16. This significantly reduces the computational cost of motion estimation without accuracy loss.

Algorithm Overview: After two stages of algorithm optimization, we present the proposed codec information reusing algorithm

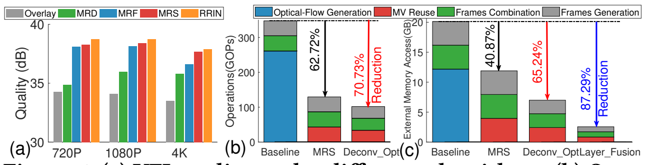


Figure 6: (a) VFI quality under different algorithms. (b) Operations (GOPs) and (c) External Memory Access (EMA, GB) of one 4K frame under different optimizations

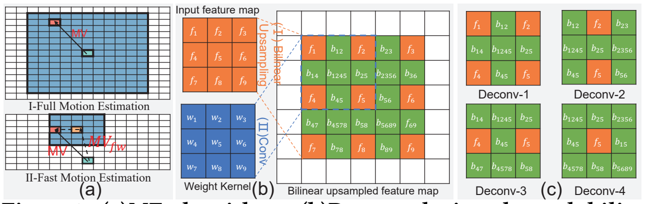


Figure 7: (a)ME algorithms. (b)Deconvolution through bilinear interpolation and convolution. (c)Four types of Deconv.

for VFI in Fig.4. ❶ We first decode the video bitstreams to obtain input frames and (forward) MVs. ❷ We apply a Fast-ME on input frames, reusing forward MVs to get backward MVs. ❸ The Refiner fine-tunes the forward and backward MVs to obtain accurate motion information. ❹ The Warper uses the motion information to warp input frames and get two warped frames. ❺ These warped frames are then fed to the Suffix DNN to generate the output frames. The suffix DNN in our algorithm includes a frames combiner and a frames generator. **In summary**, by reusing MVs, the U-Net computation for optical flow is bypassed, resulting in a 62.72% reduction in computational complexity and a 40.87% decrease in memory access, as shown in Fig. 6(b) and Fig.6(c).

4.2 Deconv Reformulation

In this subsection, we reformulate Deconv to eliminate its redundancy. Its original form is shown in Fig 7(b). It takes a 3 × 3 kernel and an input FM. Deconv first does a bilinear interpolation on input FMs to get upsampled FMs (b_{ij} is the interpolated value):

$$b_{ij} = (f_i + f_j)/2 \tag{1}$$

Then, it applies the kernel to upsampled FMs to get output FMs:

$$output = w_1 \times f_1 + w_2 \times b_{12} + w_3 \times f_2 + w_4 \times b_{14} + w_5 \times b_{1245} + w_6 \times b_{25} + w_7 \times f_4 + w_8 \times b_{45} + w_9 \times f_5 \tag{2}$$

Equation (2) has nine multiplications and eight additions. Equation (1) indicates that b_{ij} has a fixed relation with f_i and f_j. Therefore, equation (1) and equation (4.2) can be combined as:

$$output = w_1^{'} \times f_1 + w_2^{'} \times f_2 + w_3^{'} \times f_3 + w_4^{'} \times f_4 \tag{3}$$

$$w_1^{'} = w_1 + w_2/2 + w_4/2 + w_5/4, w_2^{'} = w_3 + w_2/2 + w_6/2 + w_5/4$$
$$w_3^{'} = w_7 + w_4/2 + w_8/2 + w_5/4, w_4^{'} = w_9 + w_6/2 + w_8/2 + w_5/4$$

Equation (3) only has four multiplications and three additions. Deconv can be divided into four types as shown in Fig.7(c): Deconv-1~4. As we rewrite, Deconv-1, Deconv-2, and DeConv-3 can also be rewritten with only six multiplications and five additions. After removing bilinear upsampling, Deconv shares the same expression with Conv, allowing us to design a unified hardware engine in Section 5. After optimizing Deconvolution ($Deconv_{Opt}$), the computation complexity is reduced by 70.73% as shown in Fig 6(b). Furthermore, by avoiding the upsampling operation and using smaller FMs instead of larger ones, we reduce the memory access by 65.24%.

5 THE ARCHITECTURE OF CO-VIA

5.1 Architecture Overview

The proposed compressed video VFI accelerator, Co-Via, is illustrated in Fig.8. It comprises seven components: (a) Decoder, (b) Decoding Information Buffer, (c) Motion Estimation, (d) Refine Flow, (e) Warp, (f) Frame Combine, and (g) Frame Generate Units.

The algorithm given in Fig.4 can work in a pure software implementation. However, this serial workflow, called Co-Via-Serial, operates on a frame-by-frame basis and may hinder the throughput. There are two issues arise from this approach: 1) Operating the Decoder and DNN engine in a serial mode causes pipeline bubbles and reduces hardware efficiency. 2) VFI introduces multiple U-Net modules to complete diverse functions. Sharing a single U-Net engine among these modules can lead to significant latency due to their sequential execution. Moreover, the varying characteristics across U-Net layers lower the hardware utilization.

This paper presents a Co-Via-Tile-Parallel architecture to tackle these issues through micro-architecture design and pipeline scheduling. Co-Via achieves high throughput and energy efficiency with three key features: 1) A decoder that extracts decoding information, reconstructs frames and maximizes system throughput by enabling fine-grained scheduling with the DNN engine. 2) A lightweight and flexible ME engine that computes bidirectional MVs and ensures efficient reuse of decoding information within Co-Via. 3) A U-Net engine consisting of three diverse custom cores that enhance the execution efficiency of different layers. Three heterogeneous U-Net engines are employed to execute various modules in parallel.

5.2 Co-Via Pipeline Scheme

The pipeline of Co-Via-Non-Reuse is depicted in Fig.10(a). The Optical Flow Generation is a significant bottleneck. To alleviate this, Co-Via-Serial reuses decoding information, as shown in Fig.10(b), and reduces the latency. However, since different units follow the serial frame order, each unit waits for the results of the preceding units, and some units will go into an ideal state, leading to an inefficient pipeline. Co-Via-Parallel employs three diverse U-Net Engines to tackle this, enabling the parallel execution of Decoder, Refine Flow, Frames Combine, and Frames Generate Units, as shown in Fig.10(c). This strategy greatly enhances system throughput. But there is one performance issue: each whole frame requires (e.g.,

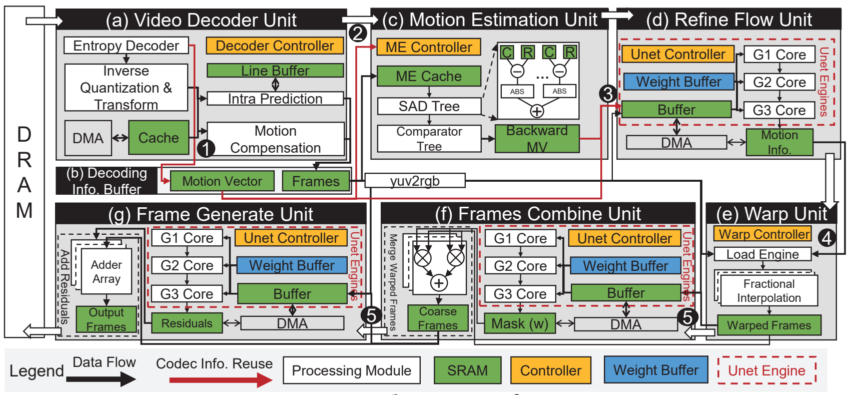


Figure 8: Architecture of Co-Via

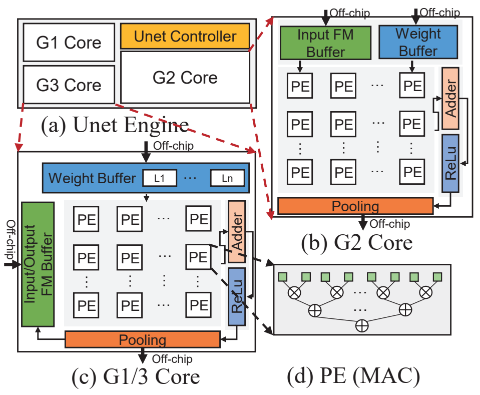


Figure 9: Architecture of U-Net Engines

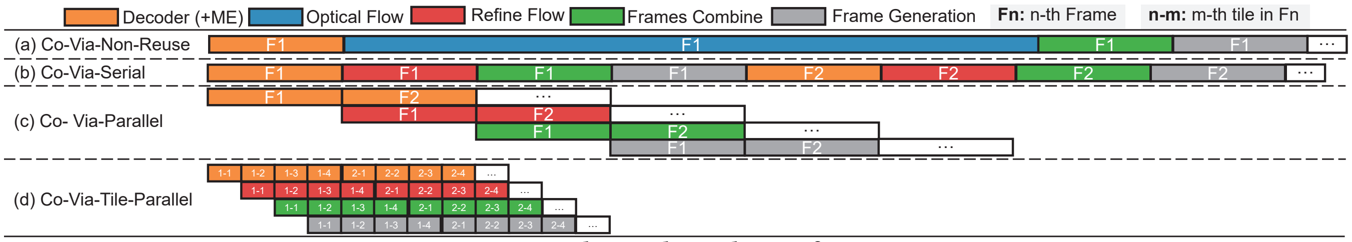


Figure 10: The pipeline scheme of Co-Via

4K) sequentially going across all modules, resulting in processing latency equivalent to the sum of these modules. This long time is not friendly to latency-sensitive applications. To resolve this, a Co-Via-Tile-Parallel scheme is proposed, as shown in Fig.10(d). The key idea is to split a frame into multiple tiles. Co-Via processes these tiles one by one, and different tiles are in parallel. Consequently, the latency of processing a single frame is sufficiently reduced.

5.3 Decoder

Unlike traditional decoder, our decoder shown in Fig.8(a) can extract MVs from video bitstreams. Employing a custom hardware decoder instead of a software one offers several advantages. First, it can reduce the decoding latency and energy cost. Additionally, it allows DNN engines to process compressed videos and enables fine-grained pipeline scheduling to improve system throughput.

5.4 Motion Estimation Unit

The lightweight ME consists of a ME cache, the sum of absolute difference (SAD) trees, and a comparator tree shown in Fig.8(c). Multiple SAD trees usually search MVs in parallel on a large search window to improve throughput. However, fast-ME reduces the window size by 1/64 for lower computation complexity. Therefore, we only use one SAD tree in ME. The ME cache is also reduced since the pixels cached by the on-chip buffer decrease.

5.5 U-Net Engine

The proposed U-Net Engine architecture, shown in Fig.9(a), consists of a U-Net controller and three heterogeneous cores (G1, G2, and G3), corresponding to the three types of layers in U-Net. Our workload analysis in Section 3 has highlighted that processors with a unified core may suffer from inefficient resource utilization due to variations in computation and memory access characteristics across different layers. To address this, we categorize layers into three groups: Group#1 (layer1∼layer5), Group#2 (layer6∼layer12), and Group#3 (layer13∼layer19), and custom heterogeneous cores.

The G1&3 core is responsible for Group#1&#3. Because Group#1 & #3 have high memory requirements and frequently access FMs, we adopt a layer-fusion strategy to alleviate this bottleneck. G1&3 generate a small portion of results before forwarding them to the next layer for processing. This allows FMs to flow through on-chip buffers and be reused by multiple layers, significantly reducing off-chip memory access. With this layer-fusion approach, we reduce the external memory access (EMA) by 87.29% as shown in Fig.6(c). The architecture of G1&3 cores, shown in Fig.9(c), supports this operation. The input and output buffers are implemented as ping-pong buffers, which are shared by multiple layers. Additionally, the weight data from different layers are traversed circularly and pre-fetched into the weight buffer. In contrast, considering the high computation intensity and low memory access in Group#2, the G2 core does not support the fusion layer and only supports layer-by-layer processing. Its architecture is shown in Fig. 9(b). It stores weights of one layer in the weight buffer, and FMs are streamed from off-chip memory, and the results are directly written back to the off-chip memory. Moreover, after converting the Deconv into Conv in Section 4, G2 and G3 perform uniformly Conv operations and support the reformulated Conv dataflow. And G1 only supports the regular Conv because Group#1 does not have Deconv layers.

This U-Net engine is employed to support the three units in Co-Via: Refine Flow Unit (Fig.8(d)), Frames Combine Unit (Fig.8(f)), and Frame Generate Unit (Fig.8(g)). Their computation resources are proportional to computation complexity to balance the pipeline.

5.6 Warp Unit

The warp unit, shown in Fig.8(e), uses refined MVs to warp input frames. These MVs have fractional values that capture high-frequency motion information. The warp unit employs integer MVs to load input frames and performs fractional interpolation with fractional MVs to generate the warped frames.

6 EXPERIMENT AND EVALUATION

6.1 Experimental Setup

Hardware Setup. The VFI system consists of a workstation with two 10-core Intel Xeon Silver 4114 @ 2.20GHz, 128G of DDR4-RAM, and FPGA accelerators connected to the host via PCIe. We implemented Co-Via with Verilog-HDL and synthesized, placed, and routed it to a Xilinx Alveo U200 FPGA with 64GB DDR4 using the Vivado 2020.02 tool at 200MHz frequency. We used JM.18.0 [15] as the H.264/AVC [16] codec reference software and developed a cycle-accurate simulator to validate the hardware platform.

Datasets. We used eight 4K video sequences [13] from SJTU Media Lab for evaluation: Construction Field (CF), Library (LY), Residential Building (RB), Runners (RS), Scarf (SF), Tall Buildings (TB), Tree Shade (TS), and Wood (WD). 720P and 1080P videos were obtained by bilinear down-sampling to evaluate different resolutions. We treat the raw video frames as the ground truth (GT). We used the AVC reference software JM 18.0 at QP22 to encode videos and generate the compressed video bitstreams. The VFI algorithms were evaluated by PSNR, which measures video quality by calculating the difference between interpolated and GT frames.

Baseline. We compare Co-Via with an NVIDIA RTX 4090 GPU. To compare Co-Via with the state-of-the-art (SOTA) VFI accelerator, VISTA [9], we synthesized Co-Via by Design Compiler with TSMC 28nm technology under the timing constraint of 500MHz.

6.2 Performance

Fig.11 shows the VFI performance improvement of different video sequences under different architectures, normalized to the GPU. We see that direct hardware acceleration, Co-Via-Non-Reuse, only has a 0.84 ∼ 1.53× speedup. By reusing the codec information to skip the optical flow generation, Via-Reuse-Serial achieves a 1.94 ∼ 3.37× speedup. Furthermore, Via-Reuse-Parallel employs three U-Net engines to parallel VFI units and improve speedup to 4.53 ∼ 6.68×. Finally, using the fine-grained tile-level parallelism,

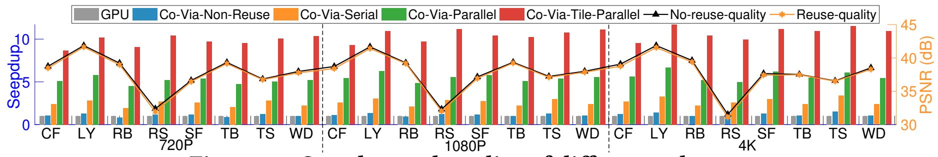


Figure 11: Speedup and quality of different schemes

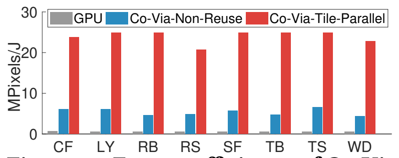


Figure 12: Energy efficiency of Co-Via

Co-Reuse-Tile-Parallel achieves a 8.65 ∼ 11.69× speedup compared with GPU. This significant speedup comes from skipping extensive DNN computation and fine-grained micro-architecture co-design of codec and DNN engines.

6.3 Quality/Accuracy Analysis

The lines in Fig.11 show the quality of compressed video VFI under the non-reuse and reuse methods. Compared with the non-reuse process, the reuse method has an average of 0.18dB of quality loss. But the texture details have no significant differences talked about in Section 4.1. Additionally, VISTA achieves a VFI quality of only 30.5 dB on a 720P Xiph-Shields video [10]. Co-Via outperforms VISTA with an impressive 32.91 dB on the same video, demonstrating its superior quality to SOTA accelerators.

6.4 Energy Efficiency

Fig.12 shows the energy efficiency of different sequences under different architectures. We use the MPixels/J (Million Pixels per Joule) as a standard metric. The average energy efficiency of GPU, Co-Via-Non-Reuse, and Co-Via-Tile-Parallel are 0.56, 5.37, and 23.97 MPixels/J respectively. Our proposed Co-Via-Tile-Parallel has a 38.11 ∼ 46.79× higher energy efficiency than GPU. This is because of the computation and EMA reduction due to the codec information reuse, the DeConv optimization, and the costumed U-Net engine.

6.5 Codec Compression Ratio Effect

Fig.13 compares quality and compression ratios across various codec configurations. We employ four codec configurations, each with distinct quantization parameters (QPs): Raw (no compression), Low (QP=5), Median (QP=22), and High (QP=38). The average compression ratios (compared to Raw) are as follows: Low (5.83), Median (15.91), and High (67.61). Regarding VFI quality, Raw video exhibits the highest quality, and quality gradually deteriorates with increased compression ratios. While a high compression ratio reduces the pressure on video transmission, it also hurts quality. Median offers a good compromise, providing more than a 15× compression ratio with slight quality loss.

6.6 Implementation Results and Comparison

Table1 shows an overall comparison of Co-Via with prior accelerators. Co-Via is the only one with a decoder to support compressed video VFI and codec information reuse. Uni-OPU[17] is a U-Net FPGA accelerator for image segmentation but cannot support VFI and FIACCEL[6] is a SOTA FPGA accelerator for VFI. Compared with them, the FPGA version of Co-Via achieves a 1.3× higher throughput and 4× energy efficiency. What's more, the ASIC version achieves a 996Mpixels/s (4K@120fps) throughput, which is 2.4× than SOTA VISTA [9], and a 14.2 TOPS/W energy efficiency, 3.6× higher than SOTA.

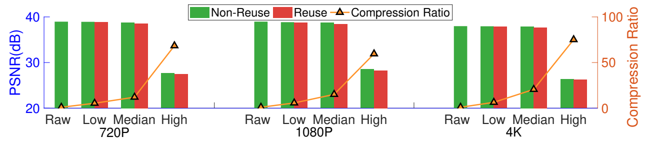


Figure 13: The impact on VFI of video compression ratios

Table 1: Overall comparisons with prior works

	FPGA			ASIC	
	Uni-OPU[17]	FIACCEL [6]	Co-Via	VISTA [9]	Co-Via
Technology	XC7Z100	XCU250	U200	40nm	28nm
Frequency (MHz)	200	300	200	200	500
DSP Usage	1987	2644	3783	-	-
LUTS Usage (K)	117	78	287	-	-
FFs Usage (K)	247	98	352	-	-
BRAM (KB)	494	622	939	994	1200
Peak TOPS	2.35	1.5	4.8	2.8	9.9
Throughput (Mpixles/s)	-	187	249	415	996(4K@120fps)
Energy Efficiency (TOPS/W)	-	0.15	0.6	4	14.2
Codec Reuse	no	no	yes	no	yes

7 CONCLUSION

This paper proposes a novel video VFI flow and a specific accelerator called Co-Via, which performs VFI directly on compressed video bitstreams. Co-Via integrates decoder and heterogeneous U-Net engines and schedules tasks at the tile level. Inspired by the similarity between optical flow and MVs, Co-Via uses a codec information reuse algorithm in VFI and implements it in hardware to improve the throughput. The FPGA and ASIC implementations show that Co-Via provides superior throughput and energy efficiency.

ACKNOWLEDGMENT

This work is supported in part by National Natural Science Foundation of China (NSFC) under grant No.(62002340, 62090020), Youth Innovation Promotion Association CAS under grant No.Y201923, and the Strategic Priority Research Program of CAS under grant No.XDB44030100. Part of this work is supported by the internship program of YUSUR Technology Co., Ltd. The corresponding authors are Guihai Yan (yan@ict.ac.cn) and Jingya Wu (jingyawu@ict.ac.cn).

REFERENCES

[1] Wenbo Bao et al. 2018. High-order model and dynamic filtering for frame rate up-conversion. *IEEE Transactions on Image Processing* 27, 8 (2018), 3813–3826.
[2] Liang Chang et al. 2023. HDSuper: Algorithm-Hardware Co-design for Light-weight High-quality Super-Resolution Accelerator. In *DAC*. IEEE, 1–6.
[3] Xianhang Cheng et al. 2021. Multiple video frame interpolation via enhanced deformable separable convolution. *TPAMI* 44, 10 (2021), 7029–7045.
[4] Byeong-Doo Choi et al. 2006. Frame rate up-conversion using perspective transform. *IEEE Transactions on Consumer Electronics* 52, 3 (2006), 975–982.
[5] Haishuang Fan et al. 2023. Co-ViSu: a Video Super-Resolution Accelerator Exploiting Codec Information Reuse. In *FPL*. IEEE, 93–100.
[6] Min Wu Jeong et al. 2023. FIACCEL: Memory Efficient Frame Interpolation Accelerator for Full-HD Video. *TCAS-II* (2023).
[7] Haopeng Li et al. 2020. Video frame interpolation via residue refinement. In *ICASSP 2020*. IEEE, 2613–2617.
[8] Zhen Li et al. 2023. AMT: All-Pairs Multi-Field Transforms for Efficient Frame Interpolation. In *CVPR*. 9801–9810.
[9] Kai-Ping Lin et al. 2023. VISTA: A 704mW 4K-UHD CNN Processor for Video and Image Spatial/Temporal Interpolation Acceleration. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 48–50.
[10] Christopher Montgomery and H Lars. 1994. Xiph. org video test media (derf's collection). *Online, https://media.xiph.org/video/derf* 6 (1994).
[11] Olaf Ronneberger et al. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI 2015*. Springer, 234–241.
[12] Uriel Singer et al. 2022. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792* (2022).
[13] Li Song et al. 2013. The SJTU 4K video sequence dataset. In *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 34–35.
[14] Zhuoran Song et al. 2020. Vr-dann: Real-time video recognition via decoder-assisted neural network acceleration. In *MICRO*. IEEE, 698–710.
[15] K Sühring et al. 2011. JVT reference software model, version JM18. 0.
[16] Thomas Wiegand et al. 2003. Overview of the H. 264/AVC video coding standard. *TCSVT* 13, 7 (2003), 560–576.
[17] Yunxuan Yu et al. 2020. Uni-OPU: An FPGA-based uniform accelerator for convolutional and transposed convolutional networks. *IEEE transactions on very large scale integration (VLSI) systems* 28, 7 (2020), 1545–1556.