Harnessing Conventional Video Processing Insights for Emerging 3D Video Generation Models: A Comprehensive Attention-aware Way

Tianlang Zhao*[1], Jun Liu*[1,2], Xingyang Li*[1],
Li Ding[1], Jinhao Li[1,3], Shuaiheng Li[1], Jinbo Hu[1], Guohao Dai[1,2,3†]
[1]Shanghai Jiao Tong University [2]Infinigence-AI [3]SII
† Corresponding author: daiguohao@sjtu.edu.cn

***Abstract*—Video Generation Models based on 3D full attention (3D-VGMs) have significantly enhanced video quality. However, their inference overhead remains substantial, primarily due to the high computational cost of the attention mechanism, which accounts for over 75% of computations. Inspired by the success of conventional video processing, where video compression exploits similarities among patches, we point out that *the attention mechanism can also harness the benefits from similarities among tokens*. Nonetheless, two critical problems arise: (1) How can similarities be efficiently acquired in real-time? (2) How can workload balance be maintained when similar tokens are randomly distributed?**

To address these problems and leverage similarities for 3D-VGMs, we propose SIMPICKER, a comprehensive attention-aware algorithm-hardware co-design for 3D-VGMs. Our core methodology is to fully utilize similarities in attention through both coarse-grained and fine-grained approaches while adopting dynamic adaptive strategies to leverage them. *From the algorithm perspective*, we propose a speculation-based similarity exploitation algorithm, allowing real-time importance speculation on the frame level, which is coarse-grained, and the token level, which is fine-grained. *From the micro-architecture perspective*, we propose a buffered lookup table-based (LUT-based) multiplication architecture for FP-INT multiplication and further eliminate potential bank conflicts to accelerate unimportant attention computation. *From the mapping perspective*, SIMPICKER proposes an adaptive grouping strategy in speculation to tame workload imbalance caused by randomly distributed similar tokens and allow seamless integration of our algorithms. Extensive experiments show that SIMPICKER achieves an average of 5.21×, 1.45× speedup and 17.92×, 1.63× energy efficiency compared to the NVIDIA A100 GPU and the state-of-the-art accelerators.

I. INTRODUCTION

Video Generation Models (VGMs) [1]–[3] have successfully created photorealistic videos using diffusion models. These models show a strong understanding of physical laws [4], marking a milestone towards advanced world models [5]. Diffusion transformer-based models [6]–[8] have shown superior performance in video generation tasks, tackling efficiency and scalability problems that challenge traditional UNet-based structures [9], [10]. The original 2D+1D attention VGMs (2D+1D VGMs) [7], [8], [11] handle spatial and temporal dimensions separately while resulting in temporal inconsistency of videos [12]. 3D-VGMs [12]–[15] are more powerful alternatives which integrate spatial and temporal information, producing higher-quality videos with better consistency.

Despite their superior performance, 3D-VGMs demand significantly more computational resources. Fig. 1 (a) illustrates the computation gap between 2D+1D VGMs and 3D-VGMs, showing that 3D-VGMs have 2×, 4× computation compared to their 2D+1D counterparts in medium-length videos and long videos. Additionally, the primary computational bottleneck shifts to the attention, which consumes over 75% of aggregate computations. The data is evaluated on CogVideoX [12] and Latte1 [8], aligning all the parameters.

Video processing technologies [16]–[18] have achieved remarkable effects due to their ability to leverage spatial and temporal similarities between patches. For instance, Discrete Cosine Transform (DCT) [19] exploits similarities between patches by converting spatial data into the frequency domain with the most energy concentrated in low-frequency components, significantly reducing high-frequency details and minimizing data redundancy. Guided by this, **we can harness the benefits from similarities among tokens for 3D-VGMs**.

However, two challenges hinder their potential for accelerating the inference of 3D-VGMs. First, **how can similarity be efficiently acquired in real time?** Existing works acquire approximate maps of attention importance via low-bit matrix multiplication [20], [21], but this approach results in significant accuracy loss of over 5% while offering nearly no acceleration in 3D-VGMs. Second, **how can workload balance be maintained when similar tokens are randomly distributed?** Although many works [22]–[24] have reused one token's results for other similar tokens based on similarities, they incur heavy workload imbalance due to randomly distributed similar tokens (computing utilization $<$37%).

To tackle the above challenges, we propose SIMPICKER, a comprehensive attention-aware algorithm-hardware co-design. Our core methodology is to fully utilize similarities in attention by both fine-grained and coarse-grained approaches and adopt dynamic and adaptive strategies to tap their full potential. The contributions can be summarized as follows:

- We introduce a **speculation-based similarity exploitation** algorithm to exploit similarities on the frame/token level. We identify the important parts in attention by pre-

*Equal Contribution.

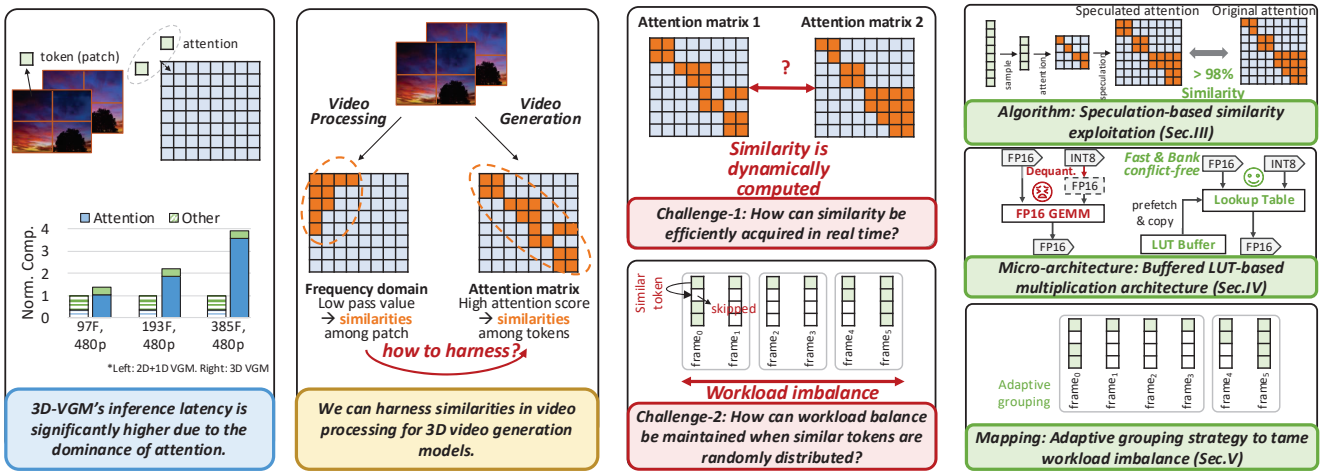

Fig. 1: Overview of SIMPICKER, a comprehensive attention-aware algorithm-hardware co-design in 3D-VGMs.

computing a small frame-level importance map so we can perform mixed-precision multiplication for unimportant parts. Also, we speculate on the input tokens and leverage their similarities to reuse results. By our speculation-based similarity exploitation algorithm, SIMPICKER can reduce over 65% FP-FP multiplications to FP-INT multiplications in attention computation, thus achieving 2.01× speedup compared with the original VGM.

- We propose a **buffered lookup table-based(LUT-based) multiplication architecture**, SIMCORE, to provide support for our algorithm. SIMCORE supports FP-INT multiplication and eliminates the bank conflicts during parallel accesses by a prefetching strategy of LUT Buffer. Our architecture achieves 1.45× speedup compared to naive design without buffered LUT.
- We propose an **adaptive grouping strategy** to tame workload imbalance introduced by randomly distributed similar tokens, allowing seamless integration between our frame-level and token-level speculations, accelerating the model by 1.10× on average.

Extensive experiments on various models and datasets demonstrate that SIMPICKER achieves an average of 5.21×, 1.45× speedup and 17.92×, 1.63× energy efficiency compared to the NVIDIA A100 GPU, and the SOTA accelerators.

II. BACKGROUND AND MOTIVATION

A. Diffusion Transformer and 3D Full Attention

3D-VGMs commonly utilize DiT as their backbone for its scalability and performance. Fig. 2 illustrates 3D-VGMs' structure and the noise prediction process, which takes CogVideoX [12] as an example. At a specific diffusion timestep [25], the input comprises a partially denoised video and prompt, both of which undergo tokenization and embedding to form a continuous token sequence $X^{2\times TN\times C}$. Here, 2 corresponds to the batch size, T denotes the number of video frames, N represents the number of tokens per frame, and C

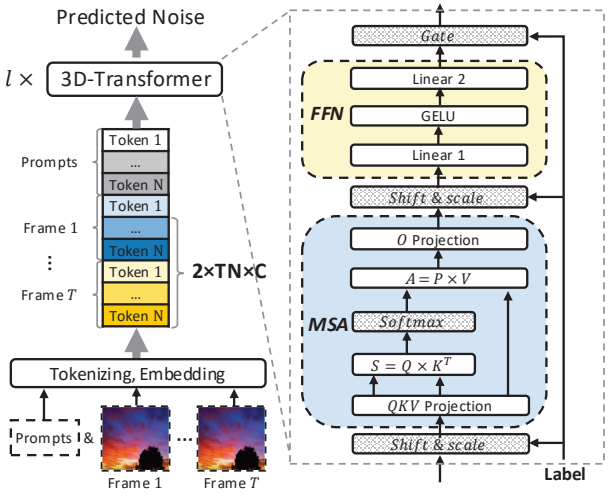

Fig. 2: 3D-VGM's structure and the noise prediction process.

indicates the hidden dimension. The token sequence is then processed through l layers of DiTs, which compute attention across all tokens from different frames.

A DiT includes a multi-head self-attention (MSA), a feed-forward network (FFN), and label embedding operators. The computing flow of MSA can be expressed by $QKV = XW_{QKV}$, $S = QK^T$, $P = Softmax(\frac{S}{\sqrt{d}})$, and $A = PV$.

B. Motivation

As illustrated in Fig. 1 (a), the primary bottleneck lies within the attention computation, which constitutes over 75% of the total computational amount. This finding prompts us to mitigate the overhead associated with the attention mechanism. Drawing inspiration from successful video processing techniques, our approach exploits the similarities and speculates importance at the frame and token levels. Also, we need dynamic and adaptive strategies to tackle the workload imbalance problems caused by similarity exploitation and fully utilize our integrated framework.

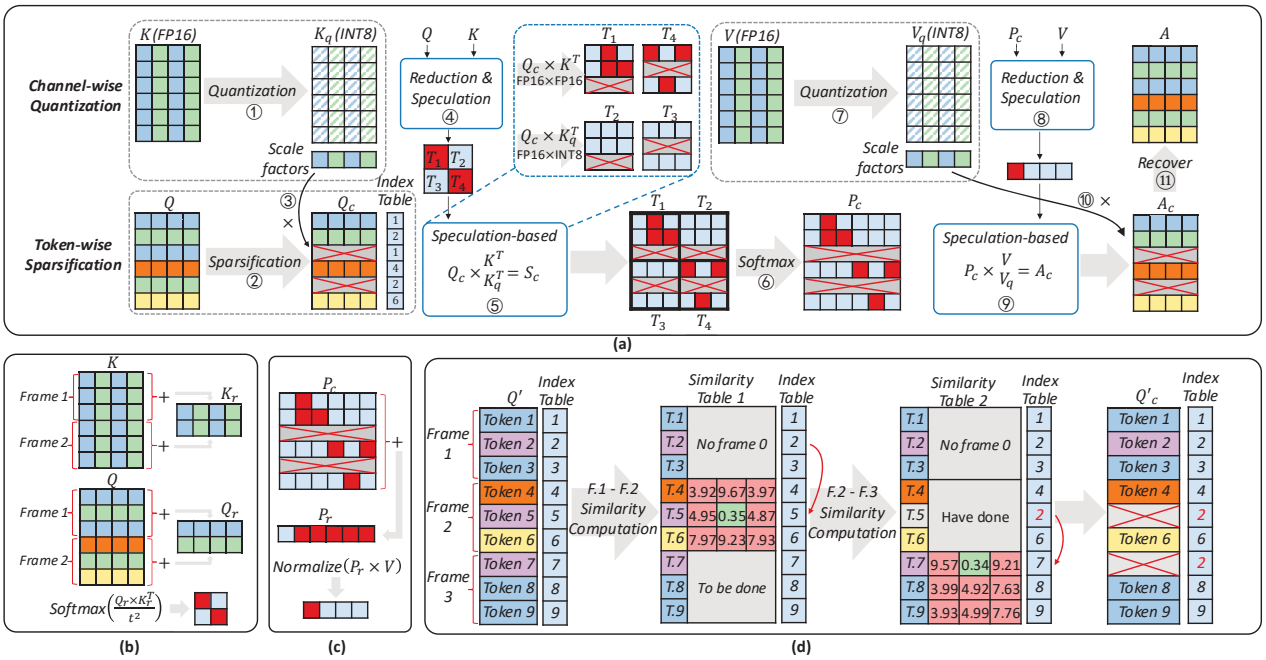

Fig. 3: (a) Overview of the algorithm. We speculate token-level similarities in Q by computing similarities and employ a speculation-based matrix multiplication algorithm for attention. (b) Speculation method for computing QK^T. (c) Speculation method for computing PV. (d) A simple example of temporal token-level similarity exploitation.

III. Speculation-based Similarity Exploitation

A. Algorithm Overview

Fig. 3 demonstrates the overview of our speculation-based similarity exploitation approach. We exploit similarities and speculate important attention parts by pre-computing a small frame-level importance map. Besides, we exploit similarities in the input tokens of the query matrix Q, thereby sparsifying it to accelerate the following computations.

B. Speculation-based Tile-level Similarity Exploitation

Importance Speculation in Attention. Guided by the observation that a small fraction of tiles holds most attention scores, we can perform importance speculation to accelerate attention computation. We aim to speculate $Z = XY$ at the tile granularity of $h \times w$, where Z can be P or A matrix in attention. We first reduce X by computing the sum over every h consecutive rows and Y over every w consecutive columns, generating the reduced matrices X_r and Y_r. We then compute the reduced product $Z_r = X_r Y_r$ and process Z_r row by row, greedily selecting the largest values until their sum exceeds a pre-defined proportion of the aggregated sum. This provides an importance map Z_{pred}, an indicator of importance for the actual computation of $Z = XY$. Fig. 3 (b) and Fig. 3 (c) illustrate the reduction and speculation process for $P = Softmax(\frac{QK^T}{\sqrt{d}})$ and $A = PV$. The ideal granularity for P's computation is $N \times N$. For A, the optimal tile size is $M \times 1$, where M is the total number of tokens.

The overhead of speculation relative to original matrix multiplication is approximately $\frac{M+N^2}{NM}$ for P and $\frac{M+d}{Md}$ for A. For the normal settings of 480p, 96-frame videos for CogVideoX, they are 3.92% and 1.57%, respectively, which are negligible. Despite some slight differences between our speculated importance map for P and P's actual importance map introduced by the $Softmax$ operator, they incur nearly no accuracy loss as most parts of them are the same.

Speculation-based Attention Computation. We quantize the K, V matrices channel-wise with symmetric 8-bit quantization for their channel-wise distribution pattern, obtaining scale factors and the quantized forms K_q and V_q. We do not quantize Q, as our token-level similarity exploitation sparsifies its structure. During the computation of QK^T and PV, we choose to perform FP-INT multiplication for unimportant parts and FP-FP multiplication for the rest, based on the frame-level importance map acquired during the aforementioned speculation. Thus, we significantly reduce over 65% FP-FP multiplications to FP-INT multiplications.

C. Speculation-based Token-level Similarity Exploitation

We exploit Q's token-level similarity to reduce computations in row dimension by speculating that tokens with similar values also have similar outputs after attention. We adopt such speculation in temporal and spatial dimensions in every layer, allowing full utilization of token-level similarity.

At the temporal dimension, the token-level similarity exploitation is performed frame by frame. We set t as a pre-defined window length, and for every token we process, we choose a $t \times t$ window in the previous frame and perform a similarity check with every token inside. We use the L2 norm to represent the similarity between two tokens. Among the $t \times t$ similarity values, we compare the minimum value with a pre-defined threshold τ. The computation of this token is reduced

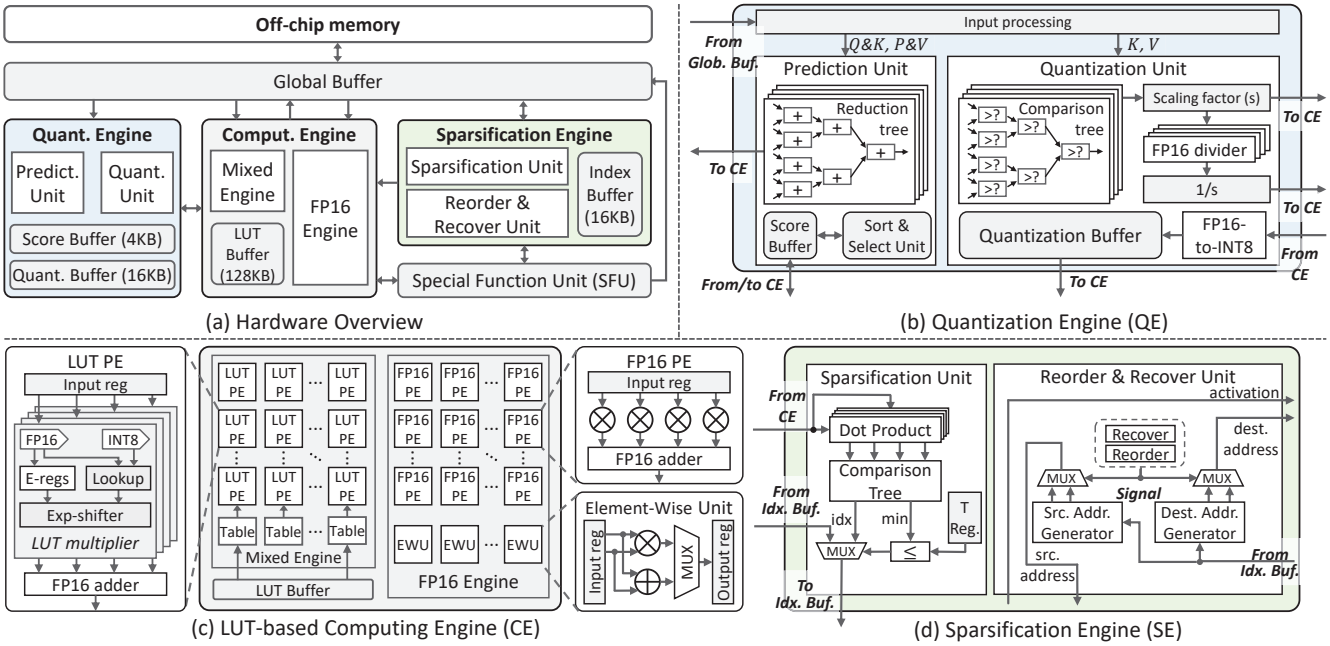

Fig. 4: (a) Our hardware architecture, SIMCORE, features (c) a LUT-based Computing Engine, and units that support our algorithm: (b) Quantization Engine, (d) Sparsification Engine

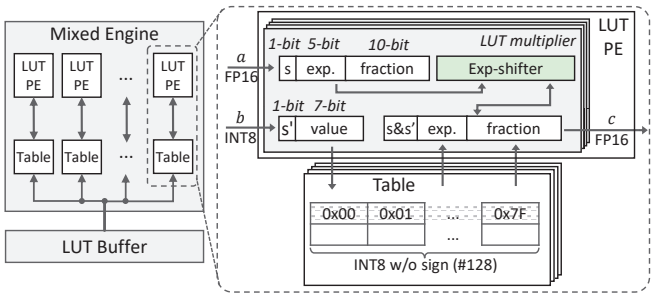

Fig. 5: The LUT-based multiplier. Each LUT multiplier is paired with a local table that prefetches and copies the intermediate results for multiplication, eliminating bank conflicts.

if the minimum value is less than τ. Fig. 3 (d) illustrates our temporal similarity exploitation.

In the spatial domain, we exploit token-level similarity by grouping tokens in each frame. We choose a reference token within each group and compute the similarity between every other token and the reference token in this group. For every non-reference token, we similarly check the threshold and adopt the result of the reference token if it is met.

IV. BUFFERED LUT-BASED MULTIPLICATION ARCHITECTURE

A. Overview of SIMCORE

We present SIMCORE, an end-to-end 3D-VGM accelerator featuring a buffered LUT-based Computing Engine that accelerates the computation of unimportant attention. SIMCORE also includes additional architectural supports for the importance speculation in attention and the token-level similarity exploitation. Fig. 4 shows our hardware overview.

B. LUT-based Computing Engine

The LUT-based Computing Engine (LUT-based CE), as illustrated in Fig. 4 (c), includes an FP16 Engine that manages FP-FP computations and a Mixed Engine that handles FP-INT multiplication with a LUT Buffer. The FP16 Engine deploys Element-Wise Units (EWU) to perform element-wise computations for quantization and sparsification. It also contains FP16 PEs that handle FP-FP matrix multiplication.

The Mixed Engine contains an array of LUT PEs, as depicted in Fig. 5. We employ the observation that the products between the 7-bit INT8 magnitudes and the 10-bit FP16 fractions can be precomputed, and they only occupy 128KB on-chip buffer size. By storing these precomputed products in the LUT Buffer, the FP-INT multiplication can be simplified into one lookup in LUT and one shift in the exponent, thus significantly reducing computations.

However, bank conflicts must be solved since different LUT PEs may require the same intermediate results, which stalls our LUT-based CE and deteriorates efficiency. Our solution utilizes a group of local tables for each PE, whose multipliers share the same FP16 operands. Before each FP-INT MAC, we prefetch intermediate results by copying the table from the LUT Buffer to each PE's local tables. We leverage double-buffer trick so that the cost of our prefetch strategy can be hidden by the computation time of the multiplier.

C. Additional Architectural Support

Shown in Fig. 4 (b), the Quantization Engine is mainly composed of a Prediction Unit (PU) and a Quantization Unit (QU). The PU aggregates input matrices via a Reduction Tree and multiplies the reduced matrices. It sorts the matrix, greedily selects the highest values with its Sort&Select Unit(S&SU), and stores the importance map in the Score Buffer. The

QU quantizes K and V into the 8-bit one's complement representation, where the last 7 bits represent magnitude.

Fig. 4 (d) shows the Sparsification Engine, which is composed of a Sparsification Unit (SU), a Reorder & Recover Unit (R&RU), and an Index Buffer. After the query matrix Q is generated, the SU exploits token-level similarity by computing the similarities and letting the Comparison Tree find the highest similarity, which is compared to the threshold in the T register. If the condition is satisfied, the Index Buffer will be updated. The R&RU reorders Q to sparsified Q_c and recover A from sparsified A_c. For both reordering and recovering, We traverse the index table, generating each index's source and destination addresses. Then, the data will be transferred according to the index and the addresses.

V. Adaptive Grouping Strategy

A. Algorithm-Hardware Mapping Overview

We describe the MSA dataflow on SimCore. Other phases in 3D-VGMs primarily involve FP-FP operations and element-wise operations and do not use the devised engines.

Initially, Q and K are reduced, and the importance map for P is generated in the Prediction Unit with the assistance of the Special Function Unit (SFU) and S&SU. Concurrently, the SU and QU produce sparsified Q_c and quantized K_q, respectively. The CE then computes S and P row-by-row, leveraging both the FP16 Engine and Mixed Engine, guided by the importance map of P. This row-by-row computation maximizes parallelism and enables the fusion of $P = Softmax\left(\frac{S}{\sqrt{d}}\right)$ and $A = PV$. As every t rows of S are computed in parallel, they are forwarded to the SFU to generate P. This row slice of P is sent to the QE to obtain the importance map for A. Here, we employ an adaptive grouping strategy for P (Sec. V-B). Subsequently, V columns are reordered and grouped based on the importance map for optimized resource utilization. Upon completing the full attention process, the Recovery Unit in the SE restores the A_c matrix to its original form.

B. Adaptive Grouping Strategy to Tame Workload Imbalance

Randomly distributed similar tokens lead to workload imbalances in the sparsified Q_c matrix. For instance, during the attention computation phase, the workload of each PE is imbalanced, necessitating the compression of Q_c to utilize computational resources fully. Such imbalance also occurs in the importance map speculation for A, where we need to recover P_c into P for speculation fully, and the distribution of P_c's indices is random and uneven. Moreover, the recovery incurs significant memory overhead due to the separation of similar token pairs in different frames.

We propose an adaptive grouping strategy to tame workload imbalance and integrate our frame-level and token-level speculations seamlessly. Instead of recovering P from P_c, we directly reduce P_c for subsequent speculation, as our observations show that the value distribution in each column of A in even, allowing such a strategy with no accuracy loss.

TABLE I: Accuracy of SimPicker in 3D-VGMs.

Component	Method	CLIPSIM ↑	VBench ↑
Open-Sora-Plan v1.2	Baseline	0.196	0.305
	SimPicker (Ours)	0.196	0.305
CogVideoX-5B	Baseline	0.304	0.311
	SimPicker (Ours)	0.307	0.312

VI. Evaluation

A. Experimental Setup

Hardware Setup. We implement SimCore using Verilog. We set CE comprising 16×8 LUT PEs with 8 LUT multipliers in each, 8×8 FP16 PEs with 8 multipliers in each, and 4×8 EWUs. The configurations of the QE and SE are aligned with CE's parallelism, enabling pipelined operator processing. The on-chip buffer includes a 208KB Global Buffer, 256KB Lookup Table, 128KB Lookup Table buffer, 16KB Index buffer, 16KB Quantization buffer, and 4KB Score buffer.

The area and power of logic units are evaluated at 1 GHz frequency using Synopsys Design Compiler under a 32nm process. Cacti 7 [26] are used for on-chip buffer evaluations.

Models and Datasets. We evaluate SimPicker on two SOTA 3D-VGMs, CogVideoX-5B [12] and Open-Sora-Plan v1.2 [14]. CogVideoX-5B concatenates text prompt tokens and video tokens together without MCA, while Open-Sora-Plan v1.2 still uses cross-attention to capture the correlation between prompt and video. We evaluate the impact of SimPicker on model accuracy on the VBench dataset [27], a carefully designed benchmark containing comprehensive prompts that describe the scene in detail.

Tasks and Metrics. We evaluate the quality of videos generated by SimPicker with two metrics, CLIPSIM [28] and the evaluation model in VBench [27]. CLIPSIM employs CLIP [28] to evaluate the similarity between the text prompt and the generated video. The evaluation metric of VBench gives a comprehensive analysis from multiple perspectives.

Baselines. To compare the performance and energy efficiency of SimPicker with SOTA works, we run open-sourced 3D-VGMs on the NVIDIA A100 GPU [29]. The inference latency and power of A100 are measured by python time library [30] and CUDA event API [31]. We also compare several SOTA ASIC accelerators, including two DiT-based works (InterArch [22] and CMC [32]) and two attention-based works (Sanger [20] and FACT [21])uu. For a fair comparison, we unify the number of FP16 MACs, on-chip buffer capacity, and frequency with those of SimCore. The off-chip bandwidth is set to 128GB/s. We design and implement a cycle-level behavioral simulator to model the operational execution of different accelerators. The CODEC auxiliary module in CMC is excluded from the evaluation.

B. Evaluation of Model Accuracy

We choose the original models with FP16 precision as the baseline. For SimPicker, the token-level similarity threshold is set to 2.5, while the proportion of the aggregated sum of important tiles in speculation is 0.65. Table. I shows that SimPicker has no accuracy loss on both metrics.

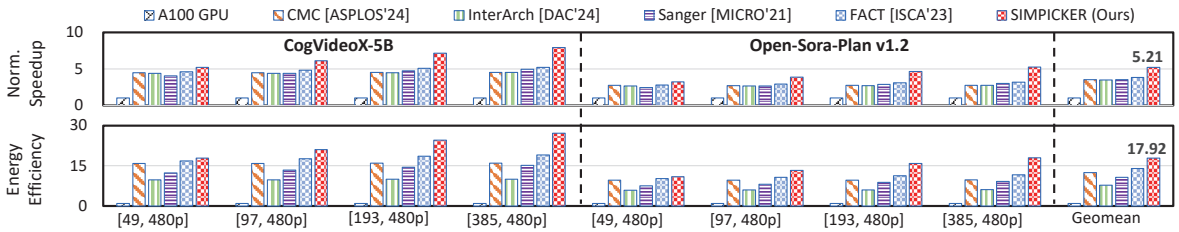

Fig. 6: Overall Performace of SIMPICKER. The horizontal axis represents [frames, resolution] of the generated video.

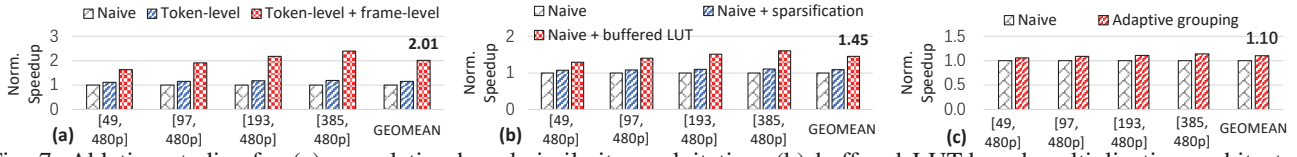

Fig. 7: Ablation studies for (a) speculation-based similarity exploitation, (b) buffered LUT-based multiplication architecture, and (c) adaptive grouping strategy. The horizontal axis represents [frames, resolution].

TABLE II: The area and power of SIMCORE.

Component		Area (mm^2)	Power (mW)
Computing Engine	16×8 LUT PEs	1.23	47.10
	8×8 FP16 PEs	2.71	323.07
	4×8 EWUs	0.04	3.58
Quantization Engine	Prediction Unit	0.02	1.40
	Quantization Unit	0.03	1.89
Spasification Engine	Sparsification Unit	0.14	16.36
	Reorder and Recovery Unit	0.06	2.69
Special Function Unit		0.23	11.02
628KB On-chip buffer		1.47	61.67
Total (32nm)		5.93	468.79

C. Hardware Analysis

The area and power of SIMCORE are shown in Table. II. The total area of SIMCORE is 5.93 mm^2, and the power is 468.79 mW. The Computing Engine is the main module, accounting for the largest proportion of 67%. The additional Quantization Engine and Sparsification Engine are only 0.25 mm^2, accounting for 4.21% of the total area.

D. Overall Performance

For both models, we evaluate the end-to-end latency and the power consumption among NVIDIA A100 GPU [29] and the ASIC accelerators under the same prompt and workload.

Performance with GPUs. For NVIDIA A100 GPU, SIMPICKER achieves 5.21× speedup on average due to our-buffered LUT-based FP-INT multiplication architecture that efficiently supports mixed-precision multiplication. Besides, SIMPICKER also achieves an average of 17.92× energy efficiency since our adaptive grouping strategy fully utilizes the hardware computing resources. The gap between SIMPICKER and GPU further expands when the video length increases, showing that SIMPICKER is also suitable for scalable settings.

Performance with ASIC accelerators For the SOTA ASIC accelerators, our SIMPICKER achieves an average 1.49× speedup and 1.87× energy efficiency compared to DiT-based accelerators CMC and InterArch. For attention-based accelerators, we get an average 1.41× speedup and 1.33× energy efficiency for Sanger and FACT.

E. Ablation Study

Speculation-based similarity exploitation algorithm. We test SIMPICKER's end-to-end latency with a naive algorithm and the algorithm that only exploits token-level similarity in our architecture under the setting of no accuracy loss. As shown in Fig. 7 (a), SIMPICKER increases the speedup by an average of 2.01×, which shows that our algorithm can effectively exploit the similarity in both frame-level and token-level. To be mentioned here, GPU cannot fully support our algorithm since there's an FP-INT multiplication.

Buffered LUT-based multiplication architecture. Experiments are taken to evaluate the end-to-end latency of our model compared to the naive one and a naive one equipped with a sparsification engine. The naive architecture only has an FP16 Engine and a Quantization Engine. As shown in Fig. 7 (b), such design significantly impacts the end-to-end latency of 1.45× on average, which indicates that our architecture can effectively support our algorithm.

Adaptive grouping strategy. As illustrated in Fig. 7 (c), we evaluate SIMPICKER's latency and breakdown with and without the adaptive reduction strategy. Results show the adaptive grouping strategy achieves a 1.10× speedup compared to a workload-imbalanced one. That's because token-level similarities are harder to exploit in 3D-VGMs due to their capabilities of generating more continuous video than their 2D+1D counterparts, leading to a low ratio of similar tokens.

VII. CONCLUSION

In this paper, we propose SIMPICKER, a comprehensive attention-aware algorithm-hardware co-design for 3D-VGMs inspired by video processing methods. SIMPICKER achieves 5.21×, 1.45× speedup and 17.92×, 1.63× energy efficiency compared to NVIDIA A100 GPU and SOTA accelerators. We hope that more works can be inspired and existing methods in video processing can be harnessed to boost video generation.

VIII. ACKNOWLEDGEMENT

This work was sponsored by the National Natural Science Foundation of China (No. U21B2031), Shanghai Rising-Star Program (No. 24QB2706200).

REFERENCES

[1] T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, *et al.*, "Video generation models as world simulators," 2024.

[2] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, *et al.*, "Stable video diffusion: Scaling latent video diffusion models to large datasets," *arXiv preprint arXiv:2311.15127*, 2023.

[3] W. Hong, M. Ding, W. Zheng, X. Liu, and J. Tang, "Cogvideo: Large-scale pretraining for text-to-video generation via transformers," *arXiv preprint arXiv:2205.15868*, 2022.

[4] B. Kang, Y. Yue, R. Lu, Z. Lin, Y. Zhao, K. Wang, G. Huang, and J. Feng, "How far is video generation from world model: A physical law perspective," 2024.

[5] S. Yang, J. Walker, J. Parker-Holder, Y. Du, J. Bruce, A. Barreto, P. Abbeel, and D. Schuurmans, "Video as the new language for real-world decision making," *arXiv preprint arXiv:2402.17139*, 2024.

[6] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182, 2023.

[7] Z. Zheng, X. Peng, T. Yang, C. Shen, S. Li, H. Liu, Y. Zhou, T. Li, and Y. You, "Open-sora: Democratizing efficient video production for all," March 2024.

[8] X. Ma, Y. Wang, G. Jia, X. Chen, Z. Liu, Y.-F. Li, C. Chen, and Y. Qiao, "Latte: Latent diffusion transformer for video generation," *arXiv preprint arXiv:2401.03048*, 2024.

[9] S. Yu, K. Sohn, S. Kim, and J. Shin, "Video probabilistic diffusion models in projected latent space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[10] K. Kim, H. Lee, J. Park, S. Kim, K. Lee, S. Kim, and J. Yoo, "Hybrid video diffusion models with 2d triplane and 3d wavelet representation," *arXiv preprint arXiv:2402.13729*, 2024.

[11] O. Bar-Tal, H. Chefer, O. Tov, C. Herrmann, R. Paiss, S. Zada, A. Ephrat, J. Hur, G. Liu, A. Raj, *et al.*, "Lumiere: A space-time diffusion model for video generation," *arXiv preprint arXiv:2401.12945*, 2024.

[12] Z. Yang, J. Teng, W. Zheng, M. Ding, S. Huang, J. Xu, Y. Yang, W. Hong, X. Zhang, G. Feng, *et al.*, "Cogvideox: Text-to-video diffusion models with an expert transformer," *arXiv preprint arXiv:2408.06072*, 2024.

[13] P. Gao, L. Zhuo, Z. Lin, C. Liu, J. Chen, R. Du, E. Xie, X. Luo, L. Qiu, Y. Zhang, *et al.*, "Lumina-t2x: Transforming text into any modality, resolution, and duration via flow-based large diffusion transformers," *arXiv preprint arXiv:2405.05945*, 2024.

[14] P.-Y. Lab and T. A. etc., "Open-sora-plan," Apr. 2024.

[15] J. Xu, X. Zou, K. Huang, Y. Chen, B. Liu, M. Cheng, X. Shi, and J. Huang, "Easyanimate: A high-performance long video generation method based on transformer architecture," *arXiv preprint arXiv:2405.18991*, 2024.

[16] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[17] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[18] M. M. Alam, T. Nguyen, M. Hagan, and D. Chandler, "A perceptual quantization strategy for hevc based on a convolutional neural network trained on natural images," p. 959918, 09 2015.

[19] Wikipedia contributors, "Discrete cosine transform — Wikipedia, the free encyclopedia," 2024. [Online; accessed 20-November-2024].

[20] L. Lu, Y. Jin, H. Bi, Z. Luo, P. Li, T. Wang, and Y. Liang, "Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, (New York, NY, USA), p. 977–991, Association for Computing Machinery, 2021.

[21] Y. Qin, Y. Wang, D. Deng, Z. Zhao, X. Yang, L. Liu, S. Wei, Y. Hu, and S. Yin, "Fact: Ffn-attention co-optimized transformer architecture with eager correlation prediction," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, (New York, NY, USA), Association for Computing Machinery, 2023.

[22] X. Wang, Z. Song, and X. Liang, "Interarch: Video transformer acceleration via inter-feature deduplication with cube-based dataflow," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, DAC '24, (New York, NY, USA), Association for Computing Machinery, 2024.

[23] X. Zhao, X. Jin, K. Wang, and Y. You, "Real-time video generation with pyramid attention broadcast," *arXiv preprint arXiv:2408.12588*, 2024.

[24] P. Selvaraju, T. Ding, T. Chen, I. Zharkov, and L. Liang, "Fora: Fast-forward caching in diffusion transformer acceleration," *arXiv preprint arXiv:2407.01425*, 2024.

[25] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[26] A. S. Naveen Muralimanohar and V. Srinivas, "Hewlettpackard/cacti." https://github.com/HewlettPackard/cacti Accessed May 22, 2023.

[27] Z. Huang, Y. He, J. Yu, F. Zhang, C. Si, Y. Jiang, Y. Zhang, T. Wu, Q. Jin, N. Chanpaisit, Y. Wang, X. Chen, L. Wang, D. Lin, Y. Qiao, and Z. Liu, "VBench: Comprehensive benchmark suite for video generative models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[28] C. Wu, L. Huang, Q. Zhang, B. Li, L. Ji, F. Yang, G. Sapiro, and N. Duan, "Godiva: Generating open-domain videos from natural descriptions," *arXiv preprint arXiv:2104.14806*, 2021.

[29] N. Corporation, *NVIDIA A100 Tensor Core GPU Architecture*, 2020. Last accessed on November 14, 2024.

[30] Python., "Python time library.." https://docs.python.org/3/library/time.html.

[31] I. NVIDIA, "Cuda event api. https://docs.nvidia.com/cuda/cuda-runtime-api/index.html," 2024.

[32] Z. Song, C. Qi, F. Liu, N. Jing, and X. Liang, "Cmc: Video transformer acceleration via codec assisted matrix condensing," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, (New York, NY, USA), p. 201–215, Association for Computing Machinery, 2024.