ViViD: A Versatile CGRA-Based Framework for Visual Rendering Acceleration on VR Devices

Yanzhou Tang*
New York University
New York, NY, USA
ytang614@gatech.edu

Tianhua Xia
New York University
New York, NY, USA
tx856@nyu.edu

Cheng Tan
Google/ASU
Mountain View, CA, USA
chengtan@asu.edu

Sai Qian Zhang
New York University
New York, NY, USA
sai.zhang@nyu.edu

Abstract

Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS) have emerged as two of the most promising techniques for high-fidelity scene reconstruction and novel view synthesis. These methods embody a well recognized trade-off: NeRF excels at producing photorealistic images with fine view-dependent details but incurs prohibitive computational costs, whereas 3DGS achieves faster performance at reduced cost, often at the expense of visual fidelity. This trade-off underscores that supporting only a single rendering paradigm is inadequate for comprehensive edge applications such as virtual reality (VR), where user experience requirements may vary across rendering settings. Despite advances in efficient implementations of individual methods, a versatile solution capable of accommodating both NeRF and 3DGS remains an open challenge.

To address this challenge, we introduce *ViViD*, a full-stack solution that supports both rendering paradigms with superior efficiency. At the algorithmic level, ViViD employs an lookahead culling strategy that eliminates redundant elements, substantially reducing computational overhead. At the hardware system level, ViViD leverages a Coarse-Grained Reconfigurable Architecture (CGRA) to efficiently execute critical kernels from both NeRF and 3DGS. Further gains are achieved through a customized quantization unit for bottleneck rendering functions, maximizing throughput while preserving image quality. Together, these optimizations deliver up to an 18.2× performance speedup and 161.0× energy efficiency speedup over optimized baselines on edge platforms NVIDIA Jetson Orin NX, enabling real-time, high fidelity rendering for edge applications.

1 Introduction

Achieving real-time, photorealistic rendering of real-world scenes is crucial for immersive Virtual Reality (VR) applications, yet it remains a persistent challenge for traditional graphics pipelines [3]. In recent years, numerous neural rendering methods have been proposed to address this gap; among them, *Neural Radiance Fields* (NeRF) [2, 23, 24, 27] and *3D Gaussian Splatting* (3DGS) [15] have emerged as two of the most promising approaches. NeRF employs an implicit representation that can achieve exceptional photorealism but often incurs high computational costs, while 3DGS leverages an explicit point-based formulation to enable real-time rendering, yet sometimes with reduced reconstruction fidelity. These complementary strengths and weaknesses highlight a fundamental trade-off between visual quality and efficiency in VR scenarios.

However, both paradigms still fall short of satisfying the stringent requirements of modern VR systems, especially on standalone edge VR devices, where high visual fidelity must be achieved together with low latency and high frame rates (FPS). To illustrate this, we profile four representative rendering methods, including three NeRF variants (MLP-based [27], Hash Grid [24], Low Rank Decomposition [2]) and 3DGS [15], on an NVIDIA Jetson Orin NX edge GPU [4] using the NeRF Synthetic dataset [23]. This experimental setup aligns with prior studies that employ edge GPUs to emulate the computational environment of standalone edge VR devices [8, 37]. As shown in Figure 1(a), these results reveal a gap between current methods and real-time VR needs, with frame rates still below the 15–20 FPS required for smooth interaction [1].

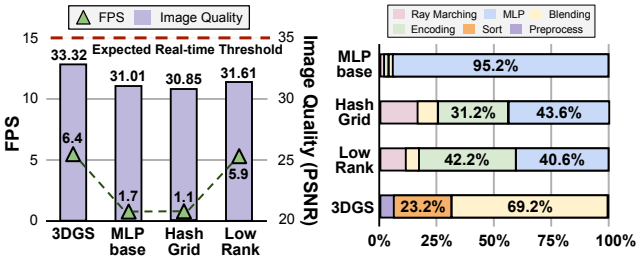

Figure 1: (a) Frames-per-second (FPS) of NeRF/3DGS on the NeRF Synthetic dataset [23], compared against the real-time threshold (15 FPS). (b) Runtime breakdown of volumetric and point-based neural rendering on the NeRF Synthetic dataset [23].

These limitations can be mitigated by adding a dedicated hardware module for rendering in modern VR devices. Because rendering is always running during immersive applications, executing it entirely on the GPU not only monopolizes GPU resources and prevents them from supporting other important workloads, but also increases latency due to GPU's under-optimized performance[36]. Offloading rendering to a specialized hardware can enhance responsiveness and accelerating rendering algorithms to sustain real time frame rates of 15 to 20 FPS required for immersive experiences.

Figure 1(b) presents the latency breakdown for NeRF and 3DGS, revealing that these two paradigms possess fundamentally different computational bottlenecks. While NeRF are primarily limited by grid encoding and multi layer perceptron (MLP) execution, 3DGS spends the majority of its cycles on sorting and alpha blending operations.

Because these performance bottlenecks vary so significantly depending on the rendering method, existing hardware designs have naturally trended toward extreme specialization. Consequently, while prior studies have introduced a variety of accelerator designs targeting either 3DGS or NeRF individually [14, 17, 18, 33], these

*Also with Georgia Institute of Technology, work done during internship at New York University.

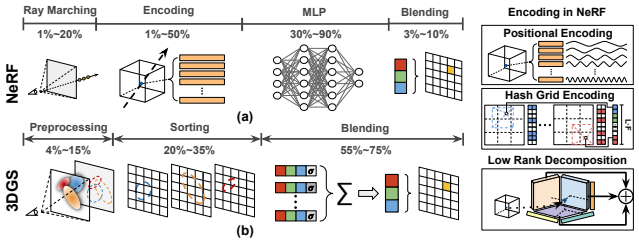

Figure 2: Overview of (a) NeRF pipelines, including MLP-based NeRF, Hash Grid NeRF, and Low-Rank Decomposition NeRF with different encoding strategies, and (b) the 3DGS pipeline.

solutions remain primarily optimized for a single paradigm. As both NeRF and 3DGS continue to advance and demonstrate strong potential for future VR systems, a practical plug-in must therefore offer sufficient flexibility to support both approaches. This highlights the need for a **versatile hardware module** that can be reconfigured to efficiently support both rendering methods.

To this end, we propose *ViViD*: A Versatile CGRA-Based Framework for Visual Rendering Acceleration on VR Devices. Our main contributions are:

- We propose a CGRA-based versatile framework. This flexible design can be dynamically configured with optimized dataflows, enabling efficient support for the distinct computational patterns of both 3DGS and NeRF.
- We present *Lookahead Culling*, a novel strategy that enhances the early termination decision, thereby reducing redundant computations across both NeRF and 3DGS.
- We augment the CGRA's processing elements (PEs) with a configurable quantization mechanism that supports various data types. This capability enables aggressive quantization of bottleneck functions, thereby maximizing hardware throughput while incurring minimal loss in visual fidelity.

2 Background and Related Work

2.1 NeRF and 3DGS Computation

Recent advances in neural rendering have made NeRF and 3DGS the leading methods for novel view synthesis. NeRF [23] employs a multi layer perceptron (MLP) to implicitly encode a scene as a continuous function that maps spatial coordinates and viewing directions to density σ and color $\mathbf{c}$. By contrast, 3DGS [15] takes an explicit approach, representing the scene as a collection of 3D Gaussian primitives, each parameterized by its position, covariance, color, and opacity. Pixel rendering is carried out by accumulating color and opacity contributions along a camera ray through alpha compositing.

As illustrated in Figure 2 (a), the NeRF pipeline consists of four stages: *ray marching*, *encoding*, *MLP inference*, and *blending*. Figure 1(b) presents the latency breakdown of the NeRF pipeline running on the NeRF Synthetic dataset[23]. In MLP-based NeRF [27], positional encoding maps input coordinates into a higher dimensional space [23], but this requires a large MLP, which consumes over 90% of the execution time. To mitigate this overhead, hash grid NeRF[24] and low rank decomposition NeRF[2] introduced alternative encodings that reduce MLP size but add non-trivial encoding costs, with encoding latency ranging from about 1% in MLP-based models to over 40%. 3DGS adopts a Gaussian-centric pipeline consisting of *preprocessing* and *sorting*, with *sorting* alone accounting for 20–35% of the runtime. The complete pipeline is illustrated in Figure 2 (b), while the runtime distribution is detailed in Fig. 1(b), which shows that pixel-centric *blending* stage is the most demanding, responsible for 55–75% of the overall latency.

Table 1: Architecture comparison. CGRA uniquely maintains a well-balanced, high capability across all essential metrics for emerging rendering tasks.

Metric	CPU	GPU	FPGA	**CGRA**	ASIC
Energy Eff.	Low	Moderate	Moderate	**High**	Very High
Performance	Low	High	Moderate	**High**	Very High
Flexibility	Very High	High	Moderate	**Moderate**	Low

2.2 Accelerators for NeRF and 3DGS

Despite the remarkable advances in rendering speed and quality achieved by NeRF and 3DGS, further performance improvements remain an active area of research. To this end, a wide range of acceleration strategies have been explored, including software-level algorithmic optimizations, the design of specialized hardware accelerators, and integrated hardware–software co-design approaches [7, 14, 17–19, 26–28, 33, 35]. For 3D Gaussian Splatting, GSCore [18] introduces a hardware accelerator that leverages shape-aware testing, hierarchical sorting, and rasterization skipping to reduce redundancy, while PS-GS [14] improves pipeline parallelism using a low-complexity accelerator with group-wise rendering. On the NeRF side, RT-NeRF [20] delivers real-time inference by decoupling spatial sampling from neural evaluation through a pipelined, two-level scheduling design. From a broader system perspective, NeuRex [17] employs a hybrid hardware–software framework to address data sparsity and workload imbalance via data-aware scheduling and hierarchical caching.

2.3 Why CGRA?

As indicated in Table 1, CGRAs provide a practical balance between the adaptability of general-purpose processors and the execution efficiency of dedicated hardware. This dual advantage of programmability and power reduction positions them as a strong candidate for the shifting rendering workloads of VR edge platforms [22, 25]. Standalone VR devices face stringent thermal and battery limitations, making them highly sensitive to power dissipation. Since rendering processes require substantial computation and energy, the target hardware must achieve high efficiency while operating within a constrained absolute power budget. The intrinsic energy benefits of CGRAs have led to their widespread application in IoT and edge computing, where specific implementations operate at sub-milliwatt levels [9, 10]. Consequently, adopting a CGRA-based accelerator offers a practical hardware solution for VR rendering.

Beyond strict power considerations, the programmability of the accelerator is equally essential. The domain of VR rendering currently lacks a single standardized algorithm and continues to advance with novel representations like NeRF and 3DGS. Hardware accelerators must therefore retain sufficient flexibility to accommodate these diverse computational paradigms. By providing reconfigurability at the PE level, CGRAs effectively address this requirement. This architectural versatility ensures that a variety of rendering algorithms can be mapped and executed efficiently without the need for repeated hardware redesigns.

3 ViViD Algorithm

To generate a pixel, both 3DGS and NeRF accumulate contributions from a sequence of samples or Gaussians that determine its final color. This process is mathematically formulated as ***blending***, where the final color C is computed as:

$$C = \sum_{i=1}^{N} T_i \alpha_i c_i, \quad \text{with} \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{1}$$

Here, c_i and α_i are the color and opacity of the i-th sample, and T_i is the accumulated transmittance. The equation describes an iterative process where the color contribution of each sample is sequentially blended to form the final pixel value. To accelerate this rendering process, both NeRF and 3DGS adopt early termination mechanisms in two ways: intra-iteration *Alpha-based early termination* and inter-iteration *Transmittance-based early termination*.

- *Alpha-based early termination (intra-iteration)*: During an iteration, if the opacity α of a sample or Gaussian falls below $1/255$, which is the minimum resolution of an 8-bit color channel, it is treated as having no contribution to the final pixel, and the current iteration is skipped.
- *Transmittance-based early termination (inter-iteration)*: once the accumulated transmittance T drops below a predefined threshold $T_{\text{threshold}}$, subsequent samples or Gaussians are regarded as having negligible impact and all remaining iterations for that pixel are terminated.

Based on the properties of these early termination mechanisms, we introduce **Lookahead Culling**, a strategy that reduces the computational workload of NeRF and 3DGS by advancing either the intra iteration or inter iteration early termination checks. Specifically, in NeRF, we reposition the inter iteration early termination prior to the color MLP, which significantly reduces the massive MLP computations. In 3DGS, we preemptively trigger the intra iteration early termination through an estimation approach, thereby reducing the computational overhead of the blending stage. Since MLP execution and blending are the primary performance bottlenecks for NeRF and 3DGS respectively, Lookahead Culling effectively targets and eliminates the redundant operations in both pipelines.

Lookahead Culling I: Postponing Color MLP in NeRF. As illustrated in Figure 3 (a), NeRF rendering process begins by casting rays from the camera's viewpoint. Points are sampled along these rays, and their color and density are accumulated sequentially according to the blending operation described by Equation 1. In NeRF, the opacity α_i for each sample i is derived from its volume density σ_i and the ray step size δ_i:

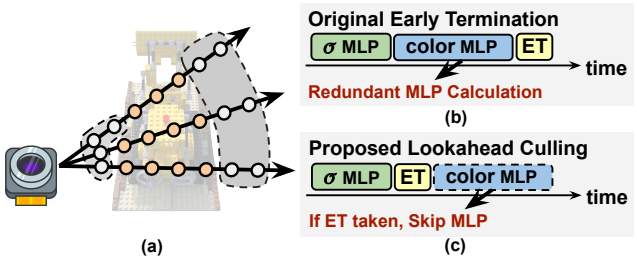

Figure 3: (a) Illustration of ray marching in NeRF, where samples in shaded regions are pruned by early termination (ET). (b) The baseline NeRF dataflow, where early termination is performed after color MLP, leading to redundant computations. (c) Our proposed Lookahead Culling I, dataflow repositions the ET check before the color MLP, eliminating unnecessary computations.

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i). \tag{2}$$

The standard NeRF pipeline first evaluates both the σ (density) MLP and the color MLP before reaching the final blending stage. However, as shown in Equation 2, the opacity α_i, which determines both early termination conditions, depends only on the output of the σ MLP. This leads to a major inefficiency: when a sample is eliminated by early termination (ET), its prior color MLP computation becomes wasted [13]. As illustrated in Figure 3 (a), samples in the grayed-out regions are discarded once the ET condition is triggered. The timeline in Figure 3(b) further shows that, for these samples, the expensive color MLP (blue) is executed in vain, resulting in redundant computation.

To address this inefficiency, our *Lookahead Culling I* strategy reorders the pipeline by performing early termination checks immediately after the σ MLP and before the color MLP, as shown in Figure 3 (c). Since this transformation is mathematically equivalent to the original formulation, it has no negative impact on the final image quality. In this way, only the surviving samples are forwarded to the color computation, eliminating redundant operations along the color branch.

Lookahead Culling II: Preemptive Approximate Alpha Evaluation in 3DGS. As shown in Figure 1, the blending stage is the most computationally dominant kernel in 3DGS, with the exponential operation alone contributing over 50% of the total computational cost. However, this high cost is primarily due to the opacity calculation for each Gaussian:

$$\alpha_i = \min\big(0.99,\ \alpha_{\text{base}} \cdot \exp(-\text{power}_i)\big), \tag{3}$$

where α_{base} is the learned base opacity and power_i is a function of the Gaussian's distance to the pixel center. As shown in Equation 3, the final opacity α_i is directly determined by the exponential input power_i, which establishes a one-to-one mapping between the input and output. Exploiting this property, we pre-compute an input threshold for power_i that enables alpha-based early termination to be applied prior to exponentiation. Since α_i depends jointly on α_{base} and the exponential term, this threshold must be

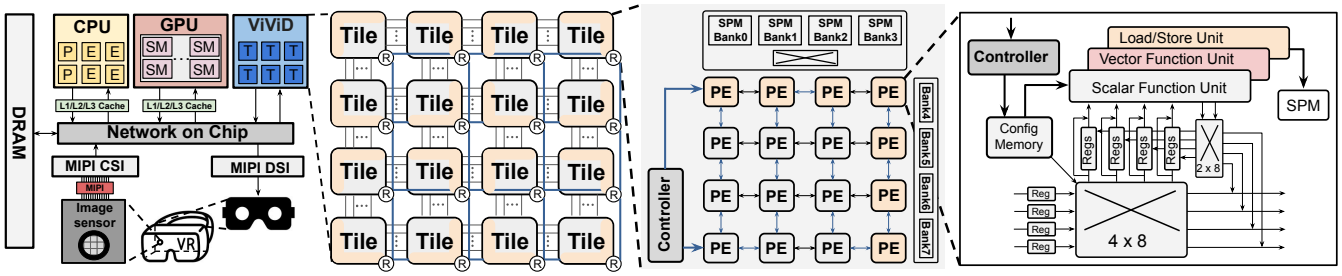

Figure 4: The Overall VR System and Internal Structure of ViVid Architecture

chosen conservatively, typically close to zero, making the method approximate and requiring a careful balance to preserve final image quality. This optimization, realized as a concrete instance of our *Lookahead Culling II* strategy, eliminates redundant exponential evaluations that would otherwise be discarded, further enhancing the blending efficiency.

It is important to note that while *Lookahead Culling II* focuses on optimizing the intra iteration alpha based early termination, our 3DGS pipeline also retains the standard inter iteration transmittance based early termination. Since the transmittance based mechanism is inherently utilized in the original 3DGS implementation and has been effectively adopted in prior hardware accelerators [18], our proposed preemptive alpha based optimization serves as an orthogonal enhancement. We do not introduce further algorithmic modifications to the transmittance based early termination itself; rather, we maintain its standard execution to work in tandem with our lookahead strategy, ensuring maximum computational savings across both intra iteration and inter iteration levels.

4 ViViD Architecture

In this section, we introduce the ViViD architecture, which is designed to efficiently support both NeRF and 3DGS rendering methods. As illustrated in Figure 4, ViViD is implemented as a unified accelerator within the system on chip (SoC) that powers VR devices. The SoC integrates conventional CPU and GPU cores alongside the ViViD accelerator, all connected to the main DRAM via a network on chip (NoC). ViViD is dedicated to offloading the computationally intensive workloads of neural rendering, thereby freeing the GPU to handle other demanding tasks such as machine learning.

The internal architecture of ViViD is a homogeneous CGRA consisting of a 4×4 array of tiles, each of which contains a 4×4 array of processing elements (PEs). This design is engineered to efficiently support the diverse computational patterns found in rendering pipelines with reduced architectural complexity and enhanced communication locality. Each PE functions as the smallest reconfigurable compute unit, capable of executing both scalar and vector operations. Specifically, each tile is equipped with a dedicated local scratchpad memory (SPM), which is partitioned into multiple banks to support parallel memory access.

To enable efficient data exchange between the PEs and the SPM, the PEs located on two specific edges of the PE array within each tile are equipped with dedicated load store units (highlighted in orange in Figure 4). Crucially, the exact position of these load store enabled edges varies depending on the spatial location of the tile within the global architecture. We logically divide the global 4×4 tile array into four 2×2 tile clusters. Load store capabilities are exclusively provisioned to the PEs situated along the four outer boundaries of each 2×2 cluster. Consequently, the orientation of the two load store enabled edges within a specific tile is strictly determined by its position within its parent 2×2 cluster. These units manage memory access requests to the tile local SPM, striking an optimal balance between memory bandwidth provision and hardware overhead.

A key strength of the ViViD CGRA architecture is its modal configurability. The array supports spatial and temporal reconfiguration, with each PE reprogrammable every cycle via dedicated configuration SRAM. This allows it to function either as a purely spatial accelerator or as a spatial–temporal fabric. At runtime, the array can adopt a TPU-like systolic configuration for matrix multiplications in MLP inference, or switch to a GPU-like SIMD configuration, where frequent temporal reprogramming improves PE utilization for tasks such as preprocessing, encoding, and pixel blending.

4.1 TPU-like Dataflow

In the TPU like configuration, ViViD adopts an output stationary dataflow to accelerate general matrix multiplication (GEMM), a core operation in MLP inference. This mode offers two distinct sub configurations to adapt to the dynamically changing matrix dimensions across different network layers, as illustrated in Figure 5(a). Specifically, the architecture leverages the specialized spatial arrangement of the load store PEs to switch between a Unified Mode and a Partitioned Mode.

The Unified Mode configures the entire 4×4 tile array into a single, massive 16×16 PE systolic array. This configuration is highly efficient for the early layers of the multi layer perceptron, where the input and hidden layer dimensions are large, allowing the architecture to maximize data reuse and spatial parallelism.

However, the final layers of rendering networks typically project features down to a very small dimension, such as a 3 channel RGB output. Mapping such a narrow matrix onto a large 16×16 array would result in severe spatial underutilization, leaving the majority of the PEs idle. To overcome this, the architecture switches to the Partitioned Mode, which logically divides the CGRA into four independent 2×2 tile clusters, yielding four 8×8 PE systolic arrays. This mode perfectly exploits the outer boundary load store PEs of each 2×2 cluster described previously, enabling them to fetch and feed data independently. By operating as four distinct arrays, the Partitioned Mode allows the system to process multiple independent batches of rays simultaneously.

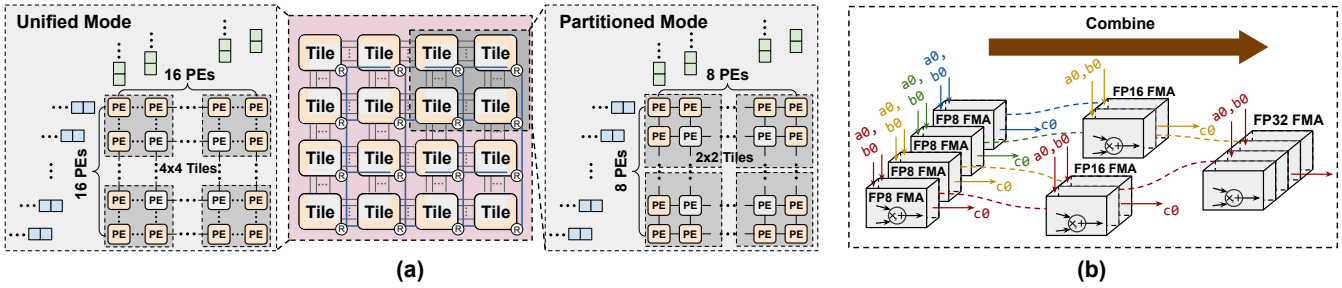

Figure 5: Configurable TPU-like Mode in CGRA and Vector FU. (a) TPU-like dataflow with a 4×4 tile array that can be configured as a unified 16×16 PE array or as 2×2 supertiles. (b) Composable vector FU architecture supporting FP8/FP16/FP32 FMA.

4.2 GPU-like Dataflow

For workloads with high data parallelism, such as Gaussian-centric preprocessing in 3DGS, the encoding stage in NeRF, and the pixel-centric blending stage in both algorithms, the CGRA can be configured into a GPU-like dataflow mode. In this configuration, each tile operates as a Single Instruction, Multiple Data (SIMD) engine. Leveraging the vector capabilities of the PEs, each tile can process four threads concurrently. Consequently, the full 4×4 tile array functions as a 64-thread parallel processor, where each tile independently executes the same instruction stream on different data. This mode is highly effective for tasks that can be broken down into numerous independent work items, providing the high degree of parallelism required for graphics-oriented rendering computations.

4.3 Rendering with Multi-precision Support

As shown in prior work [6, 30, 38], both 3DGS and NeRF run effectively in low precision. To leverage this, the CGRA integrates multi-precision functional units (FUs), enabling accuracy–throughput tradeoffs when configuring. Each vector FU supports three execution modes: single-lane FP32 fused multiply–add (FMA), dual-lane FP16 FMAs, and quad-lane FP8 FMAs. A configuration controller dynamically selects the appropriate mode for each dataflow region.

As illustrated in Figure 5(b), a FU is organized into slices. In FP8 mode, four slices operate in parallel; in FP16 mode, pairs of slices are combined; and in FP32 mode, all slices are merged into a single datapath. This design enables linear throughput scaling with reduced precision, where FP16 and FP8 achieve 2× and 4× the throughput of FP32, respectively. With unchanged pipeline depth and forwarding, precision scaling impacts only parallelism and energy efficiency, leaving kernel code intact. Operands are packed and unpacked at the CGRA boundary, while the SM register interface remains consistent.

4.4 Programmability and Compiler Support

To ensure that the ViViD architecture remains adaptable to rapidly evolving rendering algorithms, we provide a comprehensive end to end compilation stack. A historical challenge with CGRA is the extreme complexity of hardware mapping, which often requires significant manual effort. Our framework eliminates this barrier by building seamlessly upon the robust LLVM compiler infrastructure.

The compilation process automatically lowers high level algorithmic implementations into LLVM intermediate representations, performs domain specific optimizations, and generates the final dataflow graphs. A dedicated mapping engine, inspired by automated spatial mappers [31], then automatically assigns these computation graphs onto the PE array, while orchestrating the necessary routing and cycle accurate scheduling. By abstracting away the underlying hardware complexity, this automated toolchain allows developers to deploy emerging rendering techniques onto the accelerator with zero manual mapping overhead. This ensures that the hardware remains highly future proof and accessible to software researchers.

4.5 Configuration Memory and Latency Hiding

Coarser kernel and mode switches can be preloaded or streamed. The ViViD architecture supports per cycle instruction switching utilizing on tile configuration memories, allowing configuration delivery and execution to be streamed continuously. Furthermore, we employ double buffered configuration memories, enabling the system to load the next kernel configuration while the current kernel is executing. This mechanism effectively hides the majority of the reconfiguration latency during steady state execution.

To establish a conservative upper bound on the potential execution bubble at a kernel boundary, we assume a worst case modulo schedule with an initiation interval (II) of 10. This represents a strict worst case scenario, as typical kernels exhibit smaller II values owing to regular control flow and higher data reuse, which consequently reduces the volume of configuration data that must be refreshed during a switch. Under this assumption, if each processing element requires II configuration entries for the schedule, a complete reload across all 256 processing elements necessitates a total configuration volume calculated as:

$$N_{\text{entries}} = 256 \times II = 256 \times 10 = 2560 \tag{4}$$

Assuming a conservative configuration bandwidth of one entry written per cycle, the maximum potential execution bubble is bounded at 2560 cycles. At an operating frequency of 1 GHz, this absolute worst case delay translates to approximately 2.56 μs, formulated as:

$$t_{\text{bubble}} = \frac{2560 \text{ cycles}}{1 \times 10^9 \text{ Hz}} = 2.56\,\mu\text{s} \tag{5}$$

Table 2: Rendering Quality Comparison, entries marked with an asterisk indicate our reproduced results.

Methods	Implementation	Avg.
MLP-based NeRF [27]	Software	31.01
Instant-NGP [24]	Software	30.85*
TensorRF [2]	Software	31.61*
3DGS [15]	Software	34.47
ICARUS [26]	ASIC MLP based NeRF	30.21
NeuGPU [28]	ASIC Instant NGP	30.69
RT-NeRF [20]	ASIC TensoRF	31.79
GSCore [18]	ASIC 3DGS	34.40
ViViD	**CGRA MLP based NeRF**	**30.81**
ViViD	**CGRA Instant NGP**	**30.72**
ViViD	**CGRA TensoRF**	**31.23**
ViViD	**CGRA 3DGS**	**33.98**

In practice, this minimal overhead is heavily amortized by the overall kernel duration and effectively mitigated by the aforementioned double buffered architecture.

5 Experimental Evaluation

5.1 Experimental Setup

Our evaluation uses PyTorch and CUDA implementations of four representative real time rendering methods. To capture distinct architectural paradigms within neural rendering, we include three NeRF variants: KiloNeRF [27] as the MLP based method, Instant NGP [24] representing the hash grid approach, and TensoRF [2] based on low rank decomposition. We also evaluate the 3DGS technique using its official implementation [15]. To establish a performance reference and characterize the workloads, we profile rendering kernels on two NVIDIA edge platforms, Jetson Orin NX [4] and Jetson Nano [5]. These devices have also been employed in earlier work to emulate the computational conditions of standalone edge VR systems [8, 37].

The ViViD CGRA is evaluated using a dedicated simulation framework built on a custom LLVM based toolchain. Compiler passes in this toolchain map rendering kernels onto the CGRA and provide detailed cycle level latency analysis. Synthesizable RTL is generated with VecPAC [32] and synthesized using Synopsys Design Compiler targeting a 45nm technology library, which provides accurate estimates of area, power, and maximum frequency. On chip SRAM area and energy are characterized using CACTI 7.0 [21]. To support fair comparisons across different process nodes, power results are further normalized with DeepScaleTool [29].

5.2 Algorithm Evaluation

This section evaluates the impact of our proposed algorithmic enhancements on final rendering quality. Specifically, we embed the Lookahead Culling techniques introduced in Section 3 into both the 3DGS and NeRF rendering pipelines. Furthermore, we employ FP8 quantization for the exponential functions during the 3DGS blending stage and for the MLP within the NeRF architecture. To quantitatively measure the effect of these pipeline modifications,

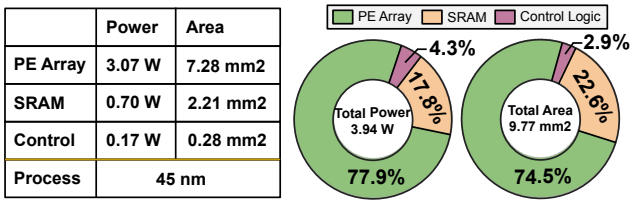

Figure 6: Power and area breakdown of ViViD accelerator

we utilize the NeRF Synthetic dataset [23] and adopt the Peak Signal to Noise Ratio (PSNR) as our primary evaluation metric. We compare the rendering fidelity of the ViViD algorithm against both the original unoptimized baselines and state of the art specialized acceleration frameworks. As presented in Table 2, the results indicate that the PSNR degradation is strictly contained to less than 1 dB across all evaluated methods and scenes relative to the baselines. Because a visual difference of under 1 dB is generally imperceptible and resides comfortably below established human perceptual thresholds [34], these findings confirm that the ViViD algorithm successfully preserves image fidelity with negligible consequences.

5.3 Hardware Evaluation

Power and Area Breakdown. We synthesized the ViViD accelerator using Synopsys Design Compiler with a 45nm TSMC technology library, targeting a frequency of 1 GHz to obtain detailed power and area estimates. On-chip SRAM area and energy were characterized separately using CACTI-7.0 [21]. As shown in Figure 6, the accelerator consumes a total of 3.94 W. The breakdown indicates that the PE array is the dominant contributor, accounting for 77.9% of the total power, followed by SRAM at 17.8% and the control logic at 4.3%. The total chip area is 9.77 mm^2, with the PE array occupying 74.5%, SRAM 22.6%, and the control logic 2.9%.

To evaluate the performance of our software-hardware co-design, we implement MLP based NeRF, Hash Grid NeRF, Low Rank Decomposition NeRF, and 3DGS on the ViViD accelerator using scenes from the NeRF Synthetic dataset [23]. To explicitly clarify our experimental boundaries and address the scope of our comparisons, the evaluations presented in this section measure the collective benefits of our entire co-design framework. Accordingly, the ViViD accelerator executes these workloads equipped with both the Lookahead Culling optimizations proposed in Section 3 and the FP8 quantization strategies detailed in Section 5.2. As shown in Figure 7(c), all four rendering methods achieve performance exceeding 15 FPS, thereby satisfying the real-time requirements for interactive applications as outlined in prior work [1].

Conversely, all evaluated baseline platforms, including commercial edge GPUs [4, 5], unified accelerator Uni-Render [19] and specialized accelerators [11, 18, 20, 28], execute the standard, unoptimized implementations of these rendering pipelines utilizing their original precision formats. This comparative methodology is adopted because these algorithmic enhancements are primary contributions of our holistic co-design, specifically formulated to synergize with the ViViD architecture. Therefore, comparing our optimized system against the standard baseline implementations

provides the most accurate assessment of the total performance and efficiency uplift delivered by our comprehensive solution.

Comparison with Edge GPUs. When evaluated against commercial edge GPU platforms [4, 5], the ViViD architecture demonstrates substantial advantages in both computational throughput and power utilization. As shown in Figure 7(a) (b), our accelerator achieves a significant performance speedup ranging from 2.8× to 18.2×, alongside an energy efficiency improvement of 24.5× to 161.0× over the Jetson Orin NX edge GPU. The Jetson Nano, despite its lower performance roofline, keeps pace with the Orin NX on most tasks, only exhibiting a notable performance drop on the compute intensive MLP-based NeRF, where it is 4× slower and 2.38× less energy efficient. Compared to the Jetson Nano, our accelerator delivers a performance speedup of 3.6× to 58.8× and an energy efficiency improvement of 19.1× to 309×.

Comparison with Unified Accelerators. We also evaluate our architecture against Uni-Render [19], a unified accelerator designed to support various NeRF models and 3DGS. As shown in Figure 7(a) (b), for MLP-based NeRF, Low Rank Decomposition NeRF, and 3DGS tasks, ViViD achieves a performance speedup ranging from 1.07× to 2.15× and an energy efficiency improvement ranging from 2.5× to 4.19× over Uni-Render. It is noteworthy that Uni-Render maintains an advantage on the Hash Grid NeRF workload. This is primarily because Uni-Render employs meticulously designed, micro-operator dataflows on its CGRA, which highly optimizes the irregular memory accesses inherent in hash encoding. Conversely, as described in Section 4.4, all ViViD kernels are automatically mapped from high-level algorithmic representations directly via our compiler. While this automated compilation stack excels at mapping compute-intensive tasks, it is currently less efficient at handling highly irregular memory access patterns compared to manual tuning. This architectural trade-off is further corroborated by ViViD's strong performance rebound on MLP-based NeRF, Low Rank Decomposition NeRF, and 3DGS; because these workloads contain a higher proportion of compute-intensive kernels, our compiler maps them efficiently, which, coupled with our software-hardware co-design, allows ViViD to surpass Uni-Render.

Comparison with Specialized Accelerators. Finally, we benchmark ViViD against dedicated specialized accelerators tailored for each specific rendering method [11, 18, 20, 28]. As expected, ViViD exhibits a performance gap ranging from 1.75× to 5.29× and an energy efficiency gap ranging from 1.47× to 11.9× compared to these highly specialized baselines in Figure 7(a) (b). This outcome aligns consistently with the architectural characteristics presented in Table 1. To maintain broad generality and support diverse rendering pipelines on a single unified fabric, our CGRA inherently sacrifices a certain degree of performance and power efficiency compared to ASICs that are rigidly hardwired for a single algorithm. Crucially, despite this architectural trade-off, ViViD consistently sustains the frame rates necessary for meeting established real-time performance thresholds [1]. This confirms that ViViD strikes a highly practical balance, trading off extreme, single-task specialization for the programmability and versatility needed to support rapidly evolving algorithms without compromising the user experience.

Contribution Analysis. To identify the source of the performance gains, we separate the overall speedup into contributions from algorithmic optimizations (which encompass our proposed lookahead culling and FP8 quantization) and hardware acceleration (derived from execution on our CGRA). For each pipeline, the individual speedups are normalized so that their combined contribution sums to 100%. As shown in Figure 7(c), hardware acceleration dominates the overall improvement for most representations, accounting for 65.3%, 66.5%, and 84.3% of the total speedup in Hash Grid NeRF, Low Rank Decomposition NeRF, and 3DGS, respectively. However, MLP-based NeRF presents a notable exception; algorithmic optimizations contribute a dominant 74.2% to the performance gain, primarily because the heavy computational bottleneck inherent in deep MLPs is significantly alleviated by our quantization strategies.

Summary and Future Adaptability. As shown in Figure 7(c), driven by this algorithm-hardware co-design, ViViD achieves optimized absolute frame rates ranging from 16.5 to 62.1 FPS, successfully satisfying real-time interactive requirements. As demonstrated throughout our evaluations, this holistic framework delivers substantial efficiency gains over edge GPUs and surpasses unified rendering accelerators in the majority of evaluated scenarios. While a performance gap naturally remains when compared to dedicated specialized accelerators, our CGRA offsets this by trading single-task efficiency for generality. Because all dataflows are fully automated by our compiler rather than relying on manual hardware mapping, ViViD can instantly adapt to and accelerate newly emerging rendering methods. This end-to-end programmability ensures that ViViD provides a highly practical, future-proof solution for the rapidly evolving field of neural rendering.

5.4 Adaptability Across Datasets

To demonstrate that ViViD can accelerate various rendering scenarios, we expand our evaluation beyond synthetic scenes [23] by incorporating real-world outdoor [16] and indoor [12] environments across varying resolutions. As shown in Table 3, ViViD achieves performance speedups over the Jetson Orin NX [4] baseline across all evaluated scenes. Furthermore, the acceleration ratio exhibits a positive correlation with input resolution. For example, the 3DGS speedup drops to 8.0× on the lower-resolution *Truck* scene (979 × 546), whereas higher-resolution scenes like *Drjohnson* (1332×876) yield increased speedups across all workloads, reaching 11.6× for 3DGS and 22.1× for Hash Grid NeRF. This occurs because Lookahead Culling bypasses redundant computations based on scene complexity. As total pixel count increases, the absolute number of early-terminated rays or culled Gaussians grows, leading to a larger workload reduction.

Processing larger resolutions introduces higher hardware overhead for memory-intensive operations. Specifically, algorithms like sorting in 3DGS or hash encoding in Hash Grid NeRF require more memory bandwidth to handle the expanded data. However, the data reduction from Lookahead Culling is more dominant, masking the disadvantages caused by bandwidth constraints and allowing the overall speedup to continue improving on higher resolution scenes.

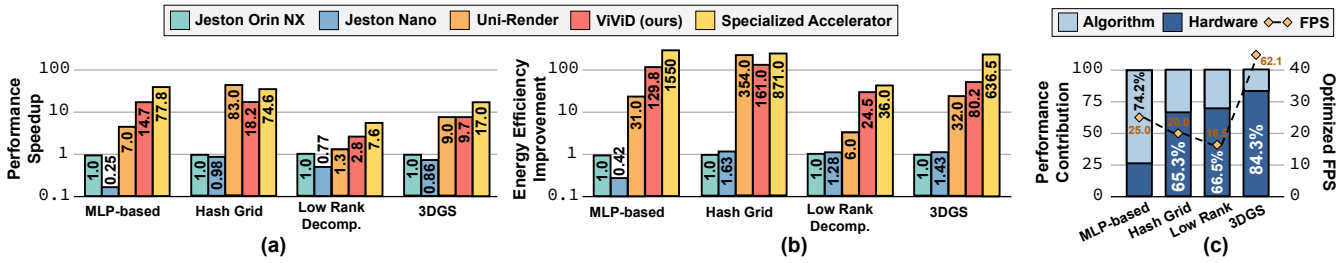

Figure 7: (a) Performance speedup and (b) Energy efficiency improvement of the proposed ViViD accelerator compared against edge GPUs [4, 5], a unified accelerator [19], and specialized accelerators. Within the specialized accelerator category, the baselines correspond to MetaVRain [11] for MLP-based NeRF, NeuGPU [28] for Hash Grid NeRF, RT-NeRF [20] for Low Rank Decomposition NeRF, and GSCore [18] for 3DGS. (c) Breakdown of overall speedup into algorithmic and hardware contributions, overlaid with the achieved optimized FPS for each rendering method.

Table 3: Performance speedup comparison of the proposed ViViD architecture across diverse rendering scenarios and resolutions. The evaluated workloads encompass a comprehensive range of dataset types, including synthetic object-centric scenes [23], real-world outdoor environments [16], and high-resolution real-world indoor spaces [12].

Dataset	Scene	MLP-based	Hash Grid	Low Rank	3DGS
Synthetic NeRF [23]	Average (800×800)	14.7×	18.2×	2.8×	9.7×
Tanks&Temples [16]	Train (980×545)	13.4×	16.6×	2.6×	8.9×
	Truck (979×546)	12.6×	15.3×	2.1×	8.0×
Deep Blending [12]	Playroom (1264×832)	16.5×	20.6×	3.4×	10.8×
	Drjohnson (1332×876)	17.9×	22.1×	3.6×	11.6×

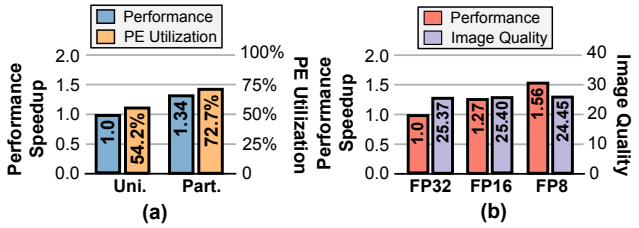

Figure 8: Performance and metrics. (a) TPU-like MLP, unified vs. partitioned: performance + PE utilization. (b) Blending kernel, FP16/FP8 vs. FP32: performance + image quality.

5.5 Ablation Study

In this section, we conduct a series of ablation studies to analyze the impact of key design choices. We specifically investigate two aspects: the configuration in TPU-like mode and the co-design between the multi-precision MUL/ADD units within the PEs and the algorithmic quantization.

Configuration in TPU-like mode. As proposed in Section 4.1, ViViD accelerator can configure its array into two distinct modes: a single, large array for large matrix multiplications, or multiple, fine-grained smaller arrays for small matrix multiplications. This flexibility is especially useful in the color MLP's final layer, which typically narrows the output dimension to 3 for RGB values. Performing this operation on a large 16×16 array would leave many PEs underutilized, leading to wasted computation. Therefore, we conduct a targeted experiment on the color MLP to compare the performance of these two modes. The results in Figure 8(a) show that when configured as a single 16×16 array, the PE utilization is only 54.2%. In contrast, by reconfiguring the array into four 8×8 arrays, we achieve a 1.34× performance speedup, and the PE utilization increases to 72.7%.

Quantization Optimization and Trade-off. As described in Section 4.3, the floating point multiply-add units in our PEs are composed of lower-bit building blocks, allowing them to be dynamically reconfigured to support FP8, FP16, or FP32 operations. This design enables flexible trade-offs between performance and precision, where lower bitwidths can substantially increase throughput but may also risk degrading the fidelity of the rendered output. To systematically evaluate this balance, we conduct quantization experiments using the *truck* scene from the Tanks&Temples dataset [16], specifically focusing on the exponential function within the Blending kernel of 3DGS. We applied FP8 and FP16 quantization to these components and assessed their effects on both system performance and the rendered images fidelity. As shown in Figure 8(b), FP16 quantization has a negligible impact on image quality, while FP8 quantization results in a PSNR drop of approximately 1 dB, which remains within an acceptable visual range. In terms of performance, FP16 quantization brings a 1.27× speedup to the blending kernel, and FP8 quantization achieves an even higher speedup of 1.56×.

6 Conclusion

In this work, we propose *ViViD*, a full-stack solution designed to support both NeRF and 3DGS with high efficiency. ViViD demonstrates that real-time, high fidelity rendering on resource constrained VR devices is achievable by co-designing algorithmic optimizations

with a CGRA-based hardware architecture. Through flexible execution modes, adaptive precision scaling, and efficient kernel scheduling, ViViD maximizes throughput while minimizing energy consumption. It achieves up to 18.2× speedup and 161.0× energy efficiency over edge GPU baselines, bridging the gap between efficiency and visual quality for next-generation immersive applications and demonstrating the immense potential of reconfigurable hardware in accelerating emerging rendering workloads.

References

[1] Rachel Albert, Anjul Patney, David Luebke, and Joohwan Kim. 2017. Latency requirements for foveated rendering in virtual reality. *ACM Transactions on Applied Perception (TAP)* 14, 4 (2017), 1–13.

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. TensoRF: Tensorial Radiance Fields. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII* (Tel Aviv, Israel). Springer-Verlag, Berlin, Heidelberg, 333–350. doi:10.1007/978-3-031-19824-3_20

[3] Guikun Chen and Wenguan Wang. 2025. A Survey on 3D Gaussian Splatting. arXiv:2401.03890 [cs.CV] https://arxiv.org/abs/2401.03890

[4] NVIDIA Corporation. [n. d.]. Jetson AGX Orin. Online. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/ Accessed: 2024-11-07.

[5] NVIDIA Corporation. [n. d.]. Jetson AGX Orin. Online. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/ Accessed: 2025-09-14.

[6] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. 2024. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. arXiv:2311.17245 [cs.CV] https://arxiv.org/abs/2311.17245

[7] Yu Feng, Zihan Liu, Jingwen Leng, Minyi Guo, and Yuhao Zhu. 2024. Cicero: Addressing Algorithmic and Architectural Bottlenecks in Neural Rendering by Radiance Warping and Memory Optimizations. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. 1293–1308. doi:10.1109/ISCA59077.2024.00096

[8] Antonin Gilles, Pierre Le Gargasson, Grégory Hocquet, and Patrick Gioia. 2023. Holographic near-eye display with real-time embedded rendering. In *SIGGRAPH Asia 2023 Conference Papers*. 1–10.

[9] Graham Gobieski, Ahmet Oguz Atli, Kenneth Mai, Brandon Lucia, and Nathan Beckmann. 2021. Snafu: An Ultra-Low-Power, Energy-Minimal CGRA-Generation Framework and Architecture. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 1027–1040. doi:10.1109/ISCA52012.2021.00084

[10] Graham Gobieski, Souradip Ghosh, Marijn Heule, Todd Mowry, Tony Nowatzki, Nathan Beckmann, and Brandon Lucia. 2022. RipTide: A Programmable, Energy-Minimal Dataflow Compiler and Architecture. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 546–564. doi:10.1109/MICRO56248.2022.00046

[11] Donghyeon Han, Junha Ryu, Sangyeob Kim, Sangjin Kim, Jongjun Park, and Hoi-Jun Yoo. 2024. MetaVRain: A Mobile Neural 3-D Rendering Processor With Bundle-Frame-Familiarity-Based NeRF Acceleration and Hybrid DNN Computing. *IEEE Journal of Solid-State Circuits* 59, 1 (2024), 65–78. doi:10.1109/JSSC.2023.3291871

[12] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.* 37, 6, Article 257 (Dec. 2018), 15 pages. doi:10.1145/3272127.3275084

[13] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. 2021. Baking Neural Radiance Fields for Real-Time View Synthesis. arXiv:2103.14645 [cs.CV] https://arxiv.org/abs/2103.14645

[14] Joongho Jo and Jongsun Park. 2025. PS-GS: Group-Wise Parallel Rendering with Stage-Wise Complexity Reductions for Real-Time 3D Gaussian Splatting. In *2025 Design, Automation Test in Europe Conference (DATE)*. 1–7. doi:10.23919/DATE64628.2025.10992921

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4, Article 139 (July 2023), 14 pages. doi:10.1145/3592433

[16] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.* 36, 4, Article 78 (July 2017), 13 pages. doi:10.1145/3072959.3073599

[17] Junseo Lee, Kwanseok Choi, Jungi Lee, Seokwon Lee, Joonho Whangbo, and Jaewoong Sim. 2023. NeuRex: A Case for Neural Rendering Acceleration. In *Proceedings of the 50th Annual International Symposium on Computer Architecture* (Orlando, FL, USA) *(ISCA '23)*. Association for Computing Machinery, New York, NY, USA, Article 21, 13 pages. doi:10.1145/3579371.3589056

[18] Junseo Lee, Seokwon Lee, Jungi Lee, Junyong Park, and Jaewoong Sim. 2024. GSCore: Efficient Radiance Field Rendering via Architectural Support for 3D Gaussian Splatting. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (La Jolla, CA, USA) *(ASPLOS '24)*. Association for Computing Machinery, New York, NY, USA, 497–511. doi:10.1145/3620666.3651385

[19] Chaojian Li, Sixu Li, Linrui Jiang, Jingqun Zhang, and Yingyan Celine Lin. 2025. Uni-Render: A Unified Accelerator for Real-Time Rendering Across Diverse Neural Renderers. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 246–260. doi:10.1109/HPCA61900.2025.00029

[20] Chaojian Li, Sixu Li, Yang Zhao, Wenbo Zhu, and Yingyan Lin. 2022. RT-NeRF: Real-Time On-Device Neural Radiance Fields Towards Immersive AR/VR Rendering. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design* (San Diego, California) *(ICCAD '22)*. Association for Computing Machinery, New York, NY, USA, Article 132, 9 pages. doi:10.1145/3508352.3549380

[21] Sheng Li, Ke Chen, Jung Ho Ahn, Jay B. Brockman, and Norman P. Jouppi. 2011. CACTI-P: Architecture-level modeling for SRAM-based structures with advanced leakage reduction techniques. In *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 694–701. doi:10.1109/ICCAD.2011.6105405

[22] Leibo Liu, Jianfeng Zhu, Zhaoshi Li, Yanan Lu, Yangdong Deng, Jie Han, Shouyi Yin, and Shaojun Wei. 2019. A Survey of Coarse-Grained Reconfigurable Architecture and Design: Taxonomy, Challenges, and Applications. *ACM Comput. Surv.* 52, 6, Article 118 (Oct. 2019), 39 pages. doi:10.1145/3357375

[23] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (Dec. 2021), 99–106. doi:10.1145/3503250

[24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages. doi:10.1145/3528223.3530127

[25] Artur Podobas, Kentaro Sano, and Satoshi Matsuoka. 2020. A Survey on Coarse-Grained Reconfigurable Architectures From a Performance Perspective. *IEEE Access* 8 (2020), 146719–146743. doi:10.1109/ACCESS.2020.3012084

[26] Chaolin Rao, Huangjie Yu, Haochuan Wan, Jindong Zhou, Yueyang Zheng, Minye Wu, Yu Ma, Anpei Chen, Binzhe Yuan, Pingqiang Zhou, Xin Lou, and Jingyi Yu. 2022. ICARUS: A Specialized Architecture for Neural Radiance Fields Rendering. *ACM Trans. Graph.* 41, 6, Article 234 (Nov. 2022), 14 pages. doi:10.1145/3550454.3555505

[27] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 14315–14325. doi:10.1109/ICCV48922.2021.01407

[28] Junha Ryu, Hankyul Kwon, Wonhoon Park, Zhiyong Li, Beomseok Kwon, Donghyeon Han, Dongseok Im, Sangyeob Kim, Hyungnam Joo, Minsung Kim, and Hoi-Jun Yoo. 2025. NeuGPU: An Energy-Efficient Neural Graphics Processing Unit for Instant Modeling and Real-Time Rendering on Mobile Devices. *IEEE Journal of Solid-State Circuits* 60, 1 (2025), 99–111. doi:10.1109/JSSC.2024.3447701

[29] Satyabrata Sarangi and Bevan Baas. 2021. DeepScaleTool: A Tool for the Accurate Estimation of Technology Scaling in the Deep-Submicron Era. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. doi:10.1109/ISCAS51556.2021.9401196

[30] Jinglei Shi and Christine Guillemot. 2023. Distilled Low Rank Neural Radiance Field with Quantization for Light Field Compression. arXiv:2208.00164 [cs.CV] https://arxiv.org/abs/2208.00164

[31] Cheng Tan, Nicolas Bohm Agostini, Jeff Zhang, Marco Minutoli, Vito Giovanni Castellana, Chenhao Xie, Tong Geng, Ang Li, Kevin Barker, and Antonino Tumeo. 2021. OpenCGRA: Democratizing Coarse-Grained Reconfigurable Arrays. In *2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. 149–155. doi:10.1109/ASAP52443.2021.00029

[32] Cheng Tan, Deepak Patil, Antonino Tumeo, Gabriel Weisz, Steve Reinhardt, and Jeff Zhang. 2023. VecPAC: A Vectorizable and Precision-Aware CGRA. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–9. doi:10.1109/ICCAD57390.2023.10323910

[33] Yuanfang Wang, Yu Li, Jianli Chen, Jun Yu, and Kun Wang. 2025. FAMERS: An FPGA Accelerator for Memory-Efficient Edge-Rendered 3D Gaussian Splatting. In *2025 Design, Automation Test in Europe Conference (DATE)*. 1–7. doi:10.23919/DATE64628.2025.10993091

[34] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[35] Lizhou Wu, Haozhe Zhu, Jiapei Zheng, Mengjie Li, Yinuo Cheng, Qi Liu, Xiaoyang Zeng, and Chixiao Chen. 2024. Hi-NeRF: A Multicore NeRF Accelerator With Hierarchical Empty Space Skipping for Edge 3-D Rendering. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 32, 12 (2024), 2315–2326. doi:10.1109/TVLSI.2024.3458032

[36] Chenhao Xie, Xie Li, Yang Hu, Huwan Peng, Michael Taylor, and Shuaiwen Leon Song. 2021. Q-VR: system-level design for future mobile collaborative virtual reality. In *Proceedings of the 26th ACM International Conference on Architectural*

Support for Programming Languages and Operating Systems (Virtual, USA) *(ASPLOS '21)*. Association for Computing Machinery, New York, NY, USA, 587–599. doi:10.1145/3445814.3446715

[37] Ziliang Zhang, Zexin Li, Hyoseung Kim, and Cong Liu. 2024. BOXR: Body and head motion Optimization framework for eXtended Reality. *arXiv preprint arXiv:2410.13084* (2024).

[38] Hongliang Zhong, Jingbo Zhang, and Jing Liao. 2024. VQ-NeRF: Neural Reflectance Decomposition and Editing With Vector Quantization. *IEEE Transactions on Visualization and Computer Graphics* 30, 9 (2024), 6247–6260. doi:10.1109/TVCG.2023.3330518