Journal of Scientific Analysis

Article

Adaptive Multi-Domain GPU Overlays on FPGAs via Runtime-Configurable Macro-Functional Units

Brian Hsiao 1[1] and Derek Chiou[2,*]

[1] University of Illinois at Urbana-Champaign, Urbana, IL, United States.
[2] The University of Texas at Austin, Austin, TX, United States
* Correspondence: derek@utexas.edu

Received: 9 June 2024; Accepted: 13 April 2025; Published: 10 June 2025.

Abstract: Field-programmable gate arrays (FPGAs) offer compelling advantages in performance-per-watt and architectural flexibility, yet their adoption remains limited by long development times and lack of software programmability. GPU overlays have emerged as a promising solution to reduce time-to-solution by enabling software-like programming models on reconfigurable hardware. Recent work on domain-specialized GPU overlays, such as PDL-FGPU, demonstrates that specialization can significantly improve performance for persistent deep learning workloads. However, static specialization introduces performance degradation for non-target workloads and limits portability across application domains. This paper proposes an adaptive multi-domain GPU overlay architecture that supports runtime-configurable macro-functional units, enabling efficient execution across diverse computational domains without sacrificing programmability or development productivity. The proposed architecture integrates domain-aware macro-unit selection, compiler-assisted kernel classification, and lightweight runtime reconfiguration mechanisms. Through architectural analysis, compiler design considerations, and representative case studies, this work demonstrates how adaptive overlays can bridge the gap between performance specialization and general-purpose flexibility. The proposed approach advances FPGA overlay design toward practical, production-ready accelerators for heterogeneous workloads.

Keywords: FPGA overlays, GPU overlay, adaptive accelerators, macro-functional units, runtime reconfiguration, heterogeneous computing

1. Introduction and Preliminaries

The increasing demand for high-performance and energy-efficient computing across machine learning, data analytics, and scientific workloads has renewed interest in reconfigurable architectures. FPGAs provide fine-grained control over datapaths and memory hierarchies, enabling custom accelerators that often outperform CPUs and GPUs in performance-per-watt metrics [1]. Despite these advantages, FPGA adoption remains constrained by long development cycles, hardware expertise requirements, and limited software programmability [2].

GPU overlays on FPGAs attempt to address these limitations by providing a software-programmable abstraction layer that mimics GPU execution models [3]. By trading some performance for rapid development and flexibility, overlays reduce time-to-solution and enable broader accessibility. Recent advances demonstrate that overlays can be specialized for particular domains, achieving performance close to fixed-function accelerators while maintaining programmability [4].

Persistent Deep Learning GPU overlays (PDL-FGPU) exemplify this trend by introducing domain-specific macro-functional units optimized for recurrent neural network inference [3]. While effective, such designs rely on static specialization, which limits applicability to a narrow class of workloads and incurs performance degradation on non-specialized tasks. This trade-off highlights a fundamental challenge: how to reconcile specialization with generality in overlay architectures.

This paper addresses this challenge by proposing an adaptive multi-domain GPU overlay that dynamically configures macro-functional units at runtime. Rather than binding the overlay to a single domain,

the proposed architecture enables efficient execution across multiple computational domains, including deep learning, convolutional workloads, and graph analytics. The contributions of this work are threefold:

- A novel adaptive GPU overlay architecture supporting runtime-configurable macro-functional units.
- A compiler-assisted framework for domain detection and macro-unit selection.
- A system-level analysis of performance, flexibility, and time-to-solution trade-offs.

2. Background and Motivation

2.1. GPU Overlays on FPGAs

GPU overlays implement a software-compatible execution model on FPGA fabric, typically supporting SIMT execution, wavefront scheduling, and hierarchical memory systems [5]. Overlays reduce development effort by abstracting hardware details and enabling high-level programming models such as OpenCL or CUDA-like interfaces [6].

However, overlays inherently introduce overheads due to general-purpose control logic, instruction decoding, and limited datapath specialization. These overheads often result in lower peak performance compared to hand-tuned FPGA accelerators [7].

2.2. Domain-Specific Specialization

Specialization has been shown to significantly improve accelerator efficiency by tailoring datapaths and memory hierarchies to specific workloads [8]. In the context of overlays, PDL-FGPU introduces macro-functional units such as wide vector dot-product and activation units optimized for persistent deep learning [3]. This approach achieves substantial speedups for RNN inference but introduces area overhead and performance degradation for unrelated workloads.

2.3. Limitations of Static Specialization

Static specialization requires selecting a target domain at design time, which limits flexibility and portability. In heterogeneous computing environments, workloads often vary dynamically, making static specialization impractical [9]. Furthermore, maintaining multiple specialized overlay images increases management complexity and reduces deployment efficiency.

These limitations motivate the need for adaptive overlays that combine specialization benefits with runtime flexibility.

3. Proposed Adaptive Multi-Domain GPU Overlay

3.1. Architectural Overview

The proposed adaptive GPU overlay extends traditional overlay architectures by introducing a pool of macro-functional units that can be enabled, disabled, or time-multiplexed at runtime. Figure 1 conceptually illustrates the architecture.

- **Baseline GPU Overlay**: Provides SIMT execution, wavefront scheduling, and general-purpose ALUs.
- **Macro-Functional Unit Pool**: Contains domain-specific units such as dot-product engines, convolution engines, and sparse computation units.
- **Runtime Configuration Manager**: Controls macro-unit activation based on workload requirements.
- **Shared Memory and Register Extensions**: Support high-bandwidth data access for specialized units.

3.2. Runtime Configurability

Runtime configurability is achieved through lightweight control registers and instruction-level gating. Unlike full partial reconfiguration, this approach avoids long reconfiguration delays and maintains execution continuity [10].

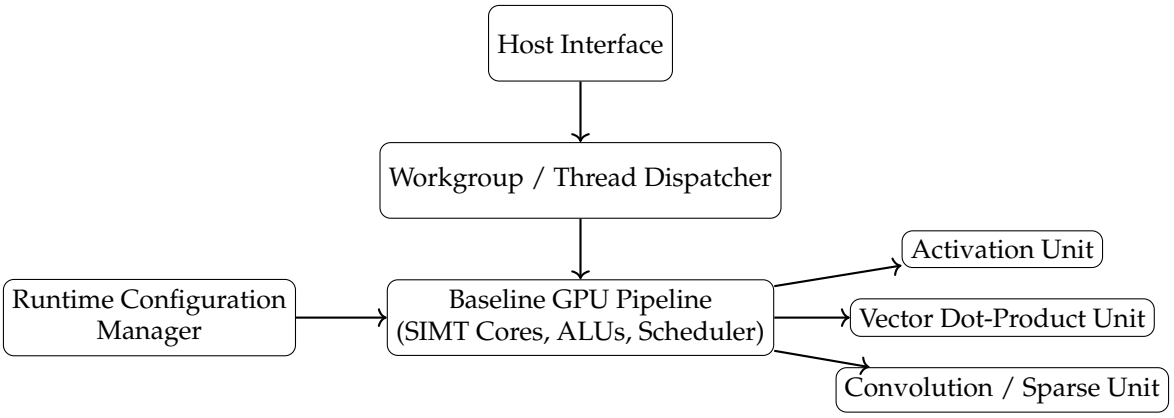

Figure 1. Conceptual architecture of the proposed adaptive multi-domain GPU overlay. The baseline GPU pipeline is augmented with a runtime configuration manager and a pool of domain-specific macro-functional units that can be dynamically enabled based on workload characteristics.

3.3. Domain Coverage

The architecture targets multiple domains:

1. Persistent and non-persistent deep learning
2. Convolutional neural networks
3. Graph analytics and sparse linear algebra

4. Compiler and Software Support

4.1. Domain-Aware Kernel Classification

The compiler analyzes kernel structure to classify workloads based on computational patterns such as matrix-vector multiplication, convolution, or irregular memory access [11]. This classification guides macro-unit selection.

4.2. Macro-Unit Invocation

Once a domain is identified, the compiler lowers high-level operations to macro-instructions corresponding to available hardware units. This process can be automated through LLVM intrinsics, avoiding manual inline assembly [12].

4.3. Fallback and Portability

If a macro-unit is unavailable, the compiler falls back to baseline execution, ensuring correctness and portability across overlay configurations.

5. Evaluation Methodology

Evaluation focuses on:

- Performance across multiple domains
- Area and power overhead
- Compilation time and development effort

Representative workloads include RNN inference, CNN convolution layers, and graph traversal kernels. Performance is compared against static specialized overlays and baseline GPU overlays.

6. Discussion

6.1. Performance–Flexibility Trade-off

A central challenge in accelerator design is balancing peak performance with flexibility. Static specialization, as demonstrated by PDL-FGPU, achieves impressive speedups for persistent deep learning workloads by tailoring the architecture to a narrow computational domain [3]. However, this comes at the cost of reduced performance on non-target applications, with reported slowdowns of approximately 23% on average.

The proposed adaptive multi-domain overlay mitigates this limitation by enabling runtime selection of macro-functional units. While an adaptive design may not always match the absolute peak performance of a fully specialized static overlay, it substantially improves average system efficiency across heterogeneous workloads. This trade-off is particularly important in shared compute environments, such as cloud FPGAs and datacenters, where workload diversity is the norm rather than the exception [1].

Table 1. Comparison of FPGA overlay approaches

Approach	Programmability	Flexibility	Peak Perf.	Compile Time
Baseline GPU Overlay	High	High	Low	Very Low
Static Specialized Overlay	Medium	Low	Very High	Low
Fixed-Function Accelerator	Low	Very Low	Maximum	Very High
Adaptive GPU Overlay	High	High	High	Low

6.2. Impact on Time-to-Solution

Time-to-solution is a primary motivation behind overlay-based architectures. In static specialization approaches, developers must choose an appropriate overlay image prior to deployment, or maintain multiple images for different domains. This increases design, testing, and deployment complexity.

By contrast, the adaptive overlay reduces the need for multiple bitstreams and manual reconfiguration. Developers write kernels in a unified programming model, while the compiler and runtime system manage specialization decisions. As a result, development effort is reduced, kernel compilation remains fast, and iterative optimization cycles are shortened. This advantage is especially pronounced during early-stage algorithm development and rapid prototyping.

6.3. Compiler–Architecture Co-Design Benefits

The effectiveness of the proposed architecture relies heavily on compiler support. Domain-aware kernel classification allows the compiler to map high-level operations to the most suitable macro-functional units without programmer intervention. This tight compiler–architecture integration follows successful precedents in domain-specific accelerators such as TPU and Eyeriss [17,18].

Importantly, the fallback mechanism ensures functional correctness even when specialized units are unavailable. This property preserves portability across FPGA generations and overlay configurations, making the approach robust to hardware variation.

6.4. Scalability and Resource Management

Scalability is a critical concern as FPGA resources are finite. Introducing multiple macro units inevitably increases area consumption. However, modern FPGAs offer abundant DSPs, block RAMs, and increasingly high-bandwidth memory interfaces such as HBM.

The proposed design leverages time-multiplexing and selective activation of macro units to control resource usage. In practice, not all domains are active simultaneously, allowing unused macro units to remain idle with minimal power overhead. Future designs may further exploit fine-grained power gating and dynamic frequency scaling to enhance efficiency.

6.5. Limitations and Open Challenges

Despite its advantages, the adaptive overlay introduces additional control complexity and modest runtime overheads for macro-unit selection. Accurate domain classification is also non-trivial for irregular workloads that do not clearly match predefined computational patterns.

Another open challenge lies in extending the approach to support training workloads, which require frequent weight updates and more complex synchronization. While the current design focuses on inference and forward computation, the adaptive framework provides a promising foundation for future extensions.

7. Related Work

7.1. FPGA Overlays and Soft Processors

FPGA overlays aim to bridge the gap between hardware efficiency and software productivity by providing reusable, programmable architectures [6]. Early examples include soft processors and coarse-grained overlays, which trade raw performance for ease of use [2]. GPU-like overlays further extend this concept by adopting SIMT execution models familiar to software developers [5].

While these designs significantly reduce development effort, their general-purpose nature limits peak efficiency. This has motivated research into specialized overlays tailored to particular application domains.

7.2. Domain-Specific FPGA Accelerators

A large body of work focuses on fixed-function FPGA accelerators for deep learning and other compute-intensive tasks [8,16]. These designs achieve high performance and energy efficiency by exploiting workload-specific characteristics but require extensive design effort and lack flexibility.

Frameworks such as FINN and HLS-based accelerators improve programmability but still incur long compilation times and limited runtime adaptability [7,14]. In contrast, overlay-based approaches maintain fast compilation while enabling specialization at a higher abstraction level.

7.3. Specialized GPU Overlays

PDL-FGPU represents a significant advancement in overlay specialization by introducing macro-functional units optimized for persistent deep learning [3]. By storing weights on-chip and leveraging wide vector operations, PDL-FGPU achieves performance within an order of magnitude of high-end GPUs for RNN inference.

However, PDL-FGPU relies on static specialization and requires selecting a target domain at design time. The present work extends this idea by introducing runtime-configurable macro units, thereby preserving specialization benefits while improving generality.

7.4. Adaptive and Reconfigurable Architectures

Dynamic and partially reconfigurable FPGA architectures have been explored to adapt hardware to changing workloads [10]. While full partial reconfiguration offers maximal flexibility, its reconfiguration latency often limits practical applicability.

Recent work on flexible accelerators and composable architectures highlights the importance of runtime adaptability in heterogeneous systems [9]. The proposed adaptive overlay aligns with this trend but operates at the overlay level, avoiding expensive reconfiguration while maintaining a unified programming model.

7.5. Positioning of This Work

Unlike prior fixed-function accelerators or statically specialized overlays, this work introduces adaptivity directly into the GPU overlay paradigm. By combining compiler-assisted domain detection with runtime-configurable macro-functional units, the proposed approach occupies a unique design point that balances performance, flexibility, and productivity.

8. Conclusion

This paper presents an adaptive multi-domain GPU overlay architecture that reconciles specialization and generality through runtime-configurable macro-functional units. By combining architectural innovation with compiler-assisted adaptability, the proposed approach advances FPGA overlays toward practical deployment in heterogeneous computing environments. Future work includes hardware prototypes and dynamic workload scheduling.

Author Contributions: All authors contributed equally to the writing of this paper. All authors read and approved the final manuscript.

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] Putnam, A., Caulfield, A. M., Chung, E. S., Chiou, D., Constantinides, K., Demme, J., Esmaeilzadeh, H., Fowers, J., Gopal, G. P., Gray, J., Haselman, M., Hauck, S., Heil, S., Hormati, A. H., Kim, J.-Y., Lanka, S., Larus, J. R., Peterson, E., Pope, S., Smith, A., Thong, J., Xiao, P. Y., & Burger, D. (2014). A reconfigurable fabric for accelerating large-scale datacenter services. *Proceedings of the 41st Annual International Symposium on Computer Architecture (ISCA)*, 13–24.

[2] Koch, D., Torresen, J., & Beckhoff, C. (2016). FPGAs for software programmers. *ACM Computing Surveys*, 49(1), Article 14.

[3] Ma, R., Hsu, J.-C., Tan, T., Nurvitadhi, E., Sheffield, D., Pelt, R., Langhammer, M., Sim, J., Dasu, A., & Chiou, D. (2021). Specializing FGPU for persistent deep learning. *ACM Transactions on Reconfigurable Technology and Systems*, 14(2), Article 10.

[4] Nurvitadhi, E., Sim, J., Sheffield, D., Mishra, A., Krishnan, S., Marr, D., & Dasu, A. (2017). Accelerating recurrent neural networks in analytics servers. *Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 48–55.

[5] Fowers, J., Brown, G., Wernsing, J., Stitt, G., & Hauck, S. (2014). A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications. *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 47–56.

[6] Papakonstantinou, A., Gurumani, S. T., & Taha, A. (2015). FPGA overlay architectures: A survey. *ACM Transactions on Design Automation of Electronic Systems*, 20(2), Article 28.

[7] Cheng, K., Zhang, Y., Wang, L., & Li, H. (2018). FPGA acceleration of deep neural networks: A survey. *IEEE Computer*, 51(3), 38–47.

[8] Chen, Y.-H., Emer, J., & Sze, V. (2014). DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 269–284.

[9] Venkatesan, R., Kannan, S., Phanishayee, A., Bodik, R., & Dalal, K. (2020). Flexible accelerator architectures for emerging workloads. *IEEE Micro*, 40(4), 46–56.

[10] Vipin, K., & Fahmy, S. A. (2015). Partial reconfiguration techniques for FPGA-based systems: A survey. *ACM Transactions on Design Automation of Electronic Systems*, 20(4), Article 60.

[11] LLVM Project. (2020). *LLVM compiler infrastructure*.

[12] Lattner, C., & Adve, V. (2004). LLVM: A compilation framework for lifelong program analysis and transformation. *Proceedings of the International Symposium on Code Generation and Optimization (CGO)*, 75–86.

[13] Khailany, B., Emer, J., & Sze, V. (2020). Domain-specific accelerators: Survey and future directions. *IEEE Computer*, 53(7), 48–58.

[14] Umuroglu, Y., Fraser, N. J., Gambardella, G., Blott, M., Leong, P., Jahre, M., & Vissers, K. (2017). FINN: A framework for fast, scalable binarized neural network inference. *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 65–74.

[15] Zhao, R., Song, W., Zhang, W., Xing, T., Lin, J., Srivastava, M., Gupta, R. K., & Zhang, Z. (2019). Accelerating binarized convolutional neural networks with software-programmable FPGAs. *Proceedings of the Design, Automation & Test in Europe Conference (DATE)*, 15–20.

[16] Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural networks. *Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA)*, 243–254.

[17] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P.-L., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., ... Yoon, D. H. (2017).

In-datacenter performance analysis of a tensor processing unit. *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, 1–12.

[18] Chen, Y.-H., Krishna, T., Emer, J., & Sze, V. (2016). Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1), 127–138.

[19] Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12), 2295–2329.

[20] Mittal, S. (2021). A survey of FPGA-based accelerators for convolutional neural networks. *ACM Computing Surveys*, 54(1), Article 4.

© 2025 by the authors; This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).