FPGAs Are the Hero In-Network Computing Needs

Suhaib A. Fahmy
KAUST
Thuwal, Saudi Arabia
suhaib.fahmy@kaust.edu.sa

Ziyi Yang
KAUST
Thuwal, Saudi Arabia
ziyi.yang@kaust.edu.sa

Yixi Chen
KAUST
Thuwal, Saudi Arabia
yixi.chen@kaust.edu.sa

Gustavo Alonso
ETH Zürich
Zürich, Switzerland

Zsolt István
Technical University of Darmstadt
Darmstadt, Germany

Marco Canini
KAUST
Thuwal, Saudi Arabia

Abstract

In-network computing has gained some traction recently with the advent of programmable switches in the datacenter, however, for widespread adoption as a general computing paradigm, more flexible deployment platforms are required. We argue that FPGAs, suitably virtualized, can serve such a role, covering the continuum from edge to cloud, addressing the key barrier of hardware deployability and flexibility for in-network computing. FPGAs offer orders of magnitude more efficient execution than general purpose CPUs and low overhead packet ingestion—and with the right hardware virtualization techniques, they can also be flexible. We propose to bring these capabilities together in a Stream-Oriented Hardware Offload paradigm supported on lightweight hostless FPGA platforms to enable In-Network Acceleration of present and emerging distributed applications.

CCS Concepts

• **Networks** → **In-network processing**; • **Hardware** → **Reconfigurable logic applications**; • **Software and its engineering** → *Distributed systems organizing principles.*

Keywords

In-network computing, field programmable gate arrays, hardware acceleration

ACM Reference Format:
Suhaib A. Fahmy, Ziyi Yang, Yixi Chen, Gustavo Alonso, Zsolt István, and Marco Canini. 2025. FPGAs Are the Hero In-Network Computing Needs. In *16th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys '25), October 12–13, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3725783.3764402

APSys '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-1572-3/25/10
https://doi.org/10.1145/3725783.3764402

1 Introduction

In-network computing (INC) has been discussed for close to three decades, with *Active Networks* [58] proposing the idea of replacing passive packets with active *capsules* that contain programs to be executed at each traversed router. However, adoption as a general compute paradigm has not fully materialized, in part due to a lack of suitable abstractions, a lack of hardware that can simultaneously support generalized computing and high performance networking, and limited use-cases.

In recent years, there has been renewed interest in INC, driven by the availability of programmable switches and SmartNICs, with modeling and generalized execution explored [14, 18]. P4 programming and Protocol Independent Switch Architecture (PISA), have emerged as a promising platform for demonstrating such ideas [51]. Management and processing operations that would traditionally run on host CPUs can be offloaded to programmable networking nodes along the data transmission path, freeing up valuable CPU resources. This enhances the performance of distributed systems, due to reduced traversal of hierarchies in the processing of packet data. Examples of traffic-processing primitives suitable for offloading include load balancing [2], congestion control [49], and intrusion detection/prevention [70]. These applications share a common assumption: they operate on a per-packet basis with payloads small enough to fit within a single packet, or processing that can be performed in the context of a single packet.

More complex Layer 7 (L7) applications, composed of function graphs, also present INC-friendly semantics. Various prototypes have demonstrated benefits such as in-network acceleration for key-value stores [34, 38, 39], transaction processing and databases [31, 32, 53], consensus [16, 17], and gradient aggregation in distributed machine learning [1, 37, 52]. These proofs of concept hint at what could be achieved with a suitable general abstraction.

Given the promising benefits of INC at the application level, the question of platform suitability arises. Existing networking devices have significant limitations. SmartNICs

with lightweight Arm cores do not offer sufficient compute capability, while offloads to specialized ASICs such as the Data Processing Accelerator (DPA) in the BlueField-3 limit flexibility. Programmable switches offer very high throughput but impose rigid programming constraints that do not suit many L7 applications. Most INC demonstrations using programmable switches rely on workarounds to offload only partial segments of application logic, e.g., ATP [37] casts floating-point to integer arithmetic as supported on Tofino switches.

We argue that the core reason that INC has not yet seen more widespread adoption as a general compute paradigm lies in the absence of a unified abstraction framework and hardware platform capable of addressing its various challenges holistically. While prior works have attempted to solve specific problems within the INC stack, these solutions are often narrow in scope and tightly coupled to their respective implementations, limiting generalization and broader adoption. To enable a mature and widely deployable INC framework, we outline four key features it must incorporate:

Low-overhead compute invocation. Deploying an application graph of functions to INC resources necessitates those functions be computed efficiently, with low latency, and for network ingestion to have minimal overhead to reap the expected benefits. That is, reducing abstraction overhead is critical—it represents a fixed cost per INC invocation and directly impacts the feasible set of offloaded functions and scalability to larger function chains. This means we require line-rate packet ingestion and processing and fully self-contained hardware acceleration of a range of functions without reliance on a host system.

Application-oriented invocation abstraction. Packet semantics are insufficient for general L7 offloads as many functions operate on data that exceeds the size of a single packet. In a packet-oriented system, this necessitates statefulness between packet processing invocations. A general INC abstraction should allocate resources and process requests at a granularity that matches application semantics, rather than at the packet level. This allows more efficient allocation and invocation of resources, and should be flexibly defined for generality.

Lightweight virtualization. Maximizing utilization and efficiency necessitates hardware resources be flexibly exploited. It is essential to enable multiple independent applications and users to leverage hardware resources in an efficient, isolated manner, and maximize hardware utilization with evolving workloads. To ensure scalability and responsiveness, INC resources should be allocated on a per-request basis, avoiding static pre-allocation. Accelerator pipelines should be invoked only when the required data is available to minimize delays due to network packet ingestion.

Feasible deployment strategy. As an emerging computing paradigm, it is commercially impractical to advocate for complete replacement of existing networking infrastructure to enable new INC capabilities. A more viable approach is for the framework to support augmenting current infrastructure in a way that scales with demand. This not only provides a practical deployment path–both within and beyond datacenter environments–but also lays the groundwork for transitioning to more efficient, purpose-built solutions as the technology matures. INC devices can be introduced into a live testbed without interrupting ongoing services, and workloads can be gradually offloaded from the original service stack to the accelerators.

Besides existing demonstration of INC applications in datacenters, a variety of broader emerging applications present significant opportunities for exploiting INC. These include smart and cognitive cities [22], augmented reality [47], self-driving [44], smart grids [42], coupling of digital twins [54], and video surveillance [10]. These applications all rely on data from multiple distributed sources, that can include complex sensors, such as high definition cameras, Lidar, environmental sensors, and end user terminals, which generate high volumes of images, video frames, dense sensor captures, or word embeddings. These streams of data are processed through complex compute pipelines, comprising multiple composed functions, such as the layers of a deep neural network [61], event detection and object segmentation in video frames, or key exchange and signature verification with advanced encryption schemes. These applications can also have challenging latency constraints due to safety criticality or interaction time requirements. INC would allow these applications to be absorbed into the network, reducing latency and data movement.

Hence, we seek a hardware platform capable of supporting meaningful application logic while satisfying the aforementioned requirements. We propose a new abstraction to address the aforementioned requirements, which we call *In-Network Acceleration*. We argue that *virtualized, hostless* FPGAs represent an ideal platform for deploying such L7 INC applications. The functions that comprise an application are fully executed within hardware accelerators which are invoked at application-defined granularity, in terms of the data size and processing complexity (e.g., image processing filters, neural network model segments, stream processing operations). These accelerators are composed across the network to implement full applications avoiding the data movement and energy overheads of typical host-managed deployments. Leveraging FPGAs' reconfigurability enables efficient hardware sharing and evolution of supported primitives over time to adapt to changing workloads. FPGAs' flexibility also allows low-level network transport abstractions

to evolve over time to encompass additional INC capabilities. Hence, a fixed hardware deployment transforms into a flexible, general-purpose computing fabric capable of supporting diverse applications with minimal network-to-compute overhead.

2 Hardware Offload Today

While FPGAs have already been widely adopted in networking research and deployed for networked applications, we find that existing abstractions fall short of meeting the objectives outlined in §1. Here, we review existing approaches that utilize FPGAs and hardware accelerators in networked applications, and highlight the key limitations that prevent them from serving as effective INC hardware platforms.

Host Oriented Hardware Offload (HOHO) is the dominant approach for deploying FPGAs and other accelerators in cloud computing, where these resources are exploited by software running on a host. This may be user facing and virtualized, or exploited behind the scenes by the operator. Applications are typically long-running, deal with large amounts of stored data shared between the host and FPGAs, which typically execute a complex portion of the whole application to achieve performance improvements. The accelerator is fully "owned" by the host application and used for offload as needed. Examples include Microsoft Catapult [48], which augmented cloud servers with FPGAs to initially accelerate the ranking engine of Bing and other service tasks [9], and Brainwave [11]. AWS F1 [4] and Alibaba F3 [3] offer instances that include FPGAs as a resource similar to GPUs, where an accelerator is integrated and managed through a host software application.

ASIC accelerators have also found use in the datacenter. Google developed the Tensor Processing Unit [35], targeting the requirements of machine learning workloads using a systolic array architecture. Similar efforts include Facebook's video processing ASICs [63] and Microsoft's Corsica compression ASIC [15]. Custom hardware that is general enough to accelerate dominant datacenter workloads is more efficient than implementing accelerators on FPGAs. However, there is a tension between the application flexibility afforded by FPGAs and the raw economic efficiency of custom ASICs. For a datacenter hyperscaler, stable application demand that requires a specific computational pattern (e.g. matrix/tensor operations) can justify the cost and effort of custom ASIC design. However, FPGAs have the flexibility to explore a wide range of applications on the same hardware deployment, extending the life of these platforms, while still reaping significant efficiency benefits over CPU baselines. FPGAs are also capable of combining accelerated processing with network ingestion logic in the same hardware, enabling the virtualization itself to be "hardware accelerated".

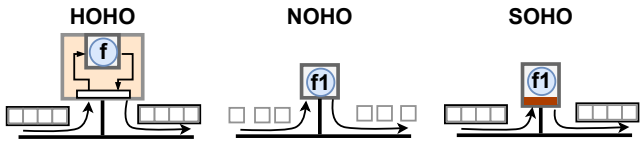

Figure 1: Comparison of hardware offload approaches. Application-level requests (black boxes) are comprised of multiple packet fragments (grey boxes). HOHO uses a host to manage application semantics and customized invocation of accelerators. NOHO can only operate on individual packets. SOHO adds a hardware abstraction to ingest and process application-level data without the involvement of a host, enabling full L7 offloads.

HOHO is not suitable for INC due to reliance on a host server, resulting in significant overhead due to the variability of network ingestion and offload over PCIe [13, 72], also resulting in wasted CPU cycles on data movement. HOHO makes most sense for stored data computations, where the result of a computation is required at the host, or where data is already present in the host.

In **Network Oriented Hardware Offload (NOHO)**, hardware is used to offload network functions, usually invisibly to applications. Hardware operates on individual packets, processing only headers or also payloads through small pipelines. The aforementioned programmable switch based INC examples fall within this paradigm. Examples of FPGA-based platforms include NetFPGA [65] and FPGA-based NICs like Corundum [23], PANIC [40], and SuperNIC [41]. Applications demonstrated using this approach include intrusion detection [66], network monitoring [62], firewalls [7], collective operations [43], etc. Some INC demonstrations have built atop these FPGA platforms, such as for key-value store in Caribou [28] and LaKe [59], demonstrating significant latency, throughput, and power improvements compared to software baselines. Other examples include State Machine Replication [29] and the Paxos consensus protocol implemented using a P4-to-FPGA compiler [16], demonstrating similar benefits. Similarly AxleDB [50] offers orders of magnitude improved efficiency for SQL queries. However, where the granularity of operations exceeds a single packet, each accelerator is designed as part of a fully customized application-specific communication stack to ingest and and process data from the network, with accelerator logic tightly coupled with network data processing. This leads to a complex monolithic hardware design lacking an abstraction to support flexible or evolving workloads. Hence, while these examples show the potential of FPGA-based INC, they do not offer a feasible abstraction for general deployment.

3 In-Network Acceleration

We propose *In-Network Acceleration* through a paradigm we call **Stream Oriented Hardware Offload (SOHO)**. In this

Table 1: Hardware offload approaches and their properties.

Property	Host-Oriented (HOHO)	Network-Oriented (NOHO)	Stream-Oriented (SOHO)
Data granularity	Large stored data	Individual packets	Application request-level
Application complexity	Rich multi-function	Fixed infrastructure	Self-contained functions
Control of resources	User dedicated	Operator	Per-request
Statefulness	Fully stateful	Stateless	Request-level statefulness
Data Movement Overhead	SW kernel network stack/PCIe	Tightly coupled packet payload	Abstracted request reassembly
Resource allocation	Host-coupled	Fixed	Per-request
Deployment approach	Incremental	Replacement	Incremental
Example for video streams	Complete analytics application	Pointwise operations	Frame-based processing

approach, applications are defined as graphs of composed functions, which are defined atomically at a granularity of request invocation: e.g., a CNN applied to an image patch, a face detector applied to a frame of video, or outlier detection applied to a time window capture of sensor data. Functions are implemented as fully self-contained accelerators on hostless FPGAs, enabling composed functions to communicate directly across the network with minimal latency and energy overhead from data ingestion. A single virtualized FPGA can host multiple accelerators, which can be reconfigured at runtime via partial reconfiguration. Accelerators are invoked at at the granularity of an application request, allowing fine-grained time-multiplexed sharing and resource allocation between multiple client requests. This approach combines the low-overhead processing benefits of NOHO, with the meaningful application-level semantics of HOHO.

In Table 1, we summarize the characteristics of the three hardware offload paradigms we have outlined. Fig. 1 compares the three approaches. We now illustrate how SOHO satisfies the requirements laid out in §1 and why FPGAs are a well-suited platform for implementation.

Low-overhead compute invocation: The ability to ingest and operate on network data at line rate on FPGAs has been widely demonstrated [25, 26]. The framework we propose is lightweight and fully implemented in hardware, enabling ingestion of network data and direct invocation of accelerators on request data without the involvement of a host. Crucially, hardware accelerators do not generally support preemption and context switching mid-execution due to large distributed internal state. Hence, adopting a run-to-completion model for acclerator invocation at the request level significantly simplifies control logic. Multiple accelerators can be hosted on a single FPGA with steering and queuing logic consuming minimal area and latency.

Application-oriented invocation abstraction: Assembling packets into application-level requests can be performed in-flight and adds minimal overhead on FPGAs due to their low latency on-chip memory capabilities. On-chip buffers can ensure requests are serviced with minimal waiting time, while accommodating request sizes in the hundreds of KBs with ease. By standardizing the data ingestion interface, SOHO allows a wide variety of accelerators to be integrated with minimal effort, without tight coupling to the underlying platform. The key requirement is that all functions comprising the application can be offloaded into hardware accelerators. While we focus on a stateless abstraction, FPGA platforms include ample off-chip memory, allowing alternative levels of statefulness to be explored through evolution of the framework.

On accelerator design: Designing accelerators at the RTL level with languages like SystemVerilog is traditionally complex and time consuming. However High Level Synthesis (HLS) tools have significantly improved, and been successfully applied in building both packet processing systems [55] and general purpose accelerators [12]. Our proposal is focused on the integration of accelerators in a networked context. We do not claim that all possible applications can be mapped or that users can have complete flexibility to design custom accelerators. However, learning from the stateless function abstraction in serverless computing, we do believe a suitably rich library of pre-designed functions can be composed into a wide range of meaningful applications. Ample examples of accelerator designs across application domains have been demonstrated in the literature. A set of functions can be compiled into a function library that is loaded on-demand at runtime through partial reconfiguration. Alternative programmable hardware structures like coarse grained overlays [30, 57] can also be explored, allowing a more software-oriented programming interface with rapid compilation.

Lightweight virtualization: FPGAs offer true spatial partitioning and isolation [27, 60]. Separate partitions of hardware resources are dedicated to distinct functions in a way that they do not interfere with each other. Fair sharing of input bandwidth and memory access is achievable through standard resource virtualization approaches. Additionally, partial reconfiguration enables accelerator swapping at runtime [64], and a suitable abstraction can enable function swapping at the millisecond timescale [21, 36]. This enables

a virtualization granularity of function execution on a complete input request, maximizing hardware utilization.

Feasible deployment strategy: SOHO integrates seamlessly with existing network infrastructure and supports flexible form factors to accommodate diverse deployment constraints. The FPGA uniquely encapsulates the whole SOHO abstraction and the compute functions within a single self-contained device that is not reliant on a standard host server. We propose these FPGA devices be network-attached to switches in the network. Being reprogrammable, the SOHO abstraction can be updated in-place without replacing the underlying hardware. The FPGAs act as standard network endpoints while retaining the ability to be reconfigured, enabling adoption of emerging paradigms such as message transport [56] and application-defined networks [71]. Their wide range of sizes and power profiles allows SOHO to be deployed on either off-the-shelf accelerator cards or custom low-power platforms, depending on performance and energy requirements. Recent FPGA platforms increasingly include embedded on-chip processors which can be used to enhance control-plane capabilities, which our framework integrates without interference with data ingestion and accelerator pipelines. Lastly, compared to programmable switching ASICs, FPGAs offer greater generality and longer-term flexibility. Unlike Intel's discontinued Tofino/Intelligent Fabric Processor (IFP) [45], FPGAs continue to provide a viable, sustainable platform. Meanwhile, proprietary in-network computing devices (e.g., IPU [24], Azure Boost DPU [5], FBOSS [46]) are typically for internal use and inaccessible to the broader research community.

4 Achieving SOHO with FPGA Accelerators

In-network acceleration augments existing networking infrastructure with independent FPGAs directly attached to networks switches. These FPGAs have off-chip DRAM and high bandwidth network interfaces. The FPGA presents virtual network endpoint interfaces which can terminate distinct traffic flows, implementing suitable network stacks, and hosts a number of reconfigurable accelerator slots that can load functions from a pre-compiled function library. A lightweight abstraction manages request assembly and accelerator invocation.

Clients establish sessions and submit requests spanning multiple packets and containing metadata in a custom request header format. These are reassembled in virtual hardware queues allocated dynamically on a per-request basis to avoid the overhead and resource under-utilization that would result from static allocation policies. Reassembled requests are scheduled to the appropriate accelerator, which runs to completion per request, returning results to the network interface, where they can pass to the next node in the function chain or back to the requester. Request-level processing means accelerators can interleave independent requests without storing state. Dynamic allocation of accelerators to slots is supported through partial reconfiguration of the FPGA, enabling adaptation to dynamic workloads. Live request statistics are collected to help determine when slots should be reconfigured.

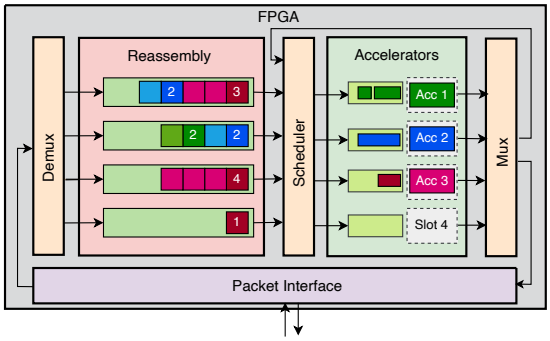

Figure 2: Proposed SOHO framework. Packet payloads are passed to Demux as fragments which are then reassembled in buffers based on information in the request headers. Complete requests queue at the corresponding accelerator. Numbers indicate request size in fragments.

Application performance is improved due to (1) direct ingestion of request packets into hardware without involving a host or software network stack, (2) processing of complete requests using optimized hardware accelerators, (3) high utilization of hardware resources to service distinct requests.

Initial Experiments

We show here that *hostless* FPGAs are capable of applying the SOHO abstraction with minimal overhead and that hardware accelerators offer a significant performance improvement.

Testbed. We prototype our system using an AMD Alveo U280 accelerator card. The FPGA functions as the receiving endpoint and performs all request processing independently of any host. Fig. 2 illustrates the internal hardware interface. Received packets are first processed by a TCP stack based on [55], with payloads forwarded to our framework. Clients are implemented using DPDK-based Libtpa [8], generating high throughput streams of application-level requests, each of which may span multiple packets over a 100 Gbps network directly connected to the FPGA.

Implementation. Fig. 2 illustrates how our framework reassembles packets on-chip into complete requests before invoking accelerator functions. Metadata from both network and application-layer headers is passed and buffers are allocated at the granularity of requests. A new request cannot join a buffer in which there is an incomplete started request but can join buffers containing complete requests from other

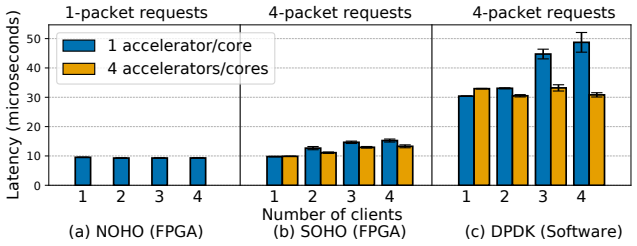

Figure 3: Latency comparison of different abstractions.

clients. Once a request is reassembled, it is queued at the relevant accelerator, and results are passed to the output network interface. Once an accelerator has completed a request, it can process any complete request in its queue. Hardware queues can have sophisticated signaling and control with minimal overhead, enabling a variety of extensions to queuing logic.

Evaluation. We measure client-observed latency of a Top-K function implemented in fixed-function hardware (NOHO), and compare it to our flexible FPGA-based SOHO framework. Clients issue back-to-back requests for 10 seconds with a 1024B packet size. NOHO is restricted to single-packet (1024B) requests, while SOHO and software accept larger 4096B requests (fragmented across packets). Fig. 3 shows the FPGA SOHO abstraction (b) incurs a minimal overhead of 0.3–3 μs additional latency when hosting 4 accelerators (i.e., equal processing load) compared to fixed function NOHO (a). Software running on 1 or 4 AMD EPYC 7763 CPU cores (c) shows significantly higher latency and variability with more clients. SOHO reduces latency and improves scalability compared to the software baseline, achieving a 3× speedup when using four accelerators in SOHO versus four CPU cores in the software implementation. This highlights the speedup benefits combined with minimal overhead incurred by the SOHO abstraction on FPGA. This also results in a significant efficiency saving since the FPGA can be deployed without a host, dramatically reducing power consumption compared to a hosted accelerator. A more thorough investigation of this design can be referenced in [68].

5 Research Directions

We have demonstrated it is possible to implement in hardware on an FPGA a lightweight accelerator invocation abstraction that presents a request-level interface, moving data into accelerator pipelines with minimal overhead. There remain a number of areas to develop to make In-Network Acceleration a reality.

Service discovery protocol. In a real deployment, discovery and allocation of in-network accelerators needs to be addressed. Beaconing, as in SCION [69], or pathlet discovery as in MTP [33] could serve the purpose of resource discovery. FPGAs are well suited to offering real time telemetry to enable dynamic discovery and allocation to suitably share resources across multiple concurrent applications and users.

Function chaining. In-network computing is most effective when supporting directed acyclic graphs (DAGs) of composed functions allocated across distributed resources, due to the significant layer traversal overhead savings. This view of applications is amenable as has been demonstrated in serverless computing (including exploratory work combining FPGAs and the serverless paradigm [6, 20]. New algorithms are required to address dynamic mapping of DAGs to distributed accelerator resources based on dynamic conditions, similar to the scenario explored in MTP [33].

Quality of Service. In-network telemetry can address quality of service guarantees [67], including dealing with mixed criticality workloads. Capturing this telemetry has minimal impact on the data plane due to the hardware pipeline isolation capabilities of FPGAs, thereby offering a level of dynamic responsiveness that would not normally be possible with traditional computing devices. It would additionally be necessary to consider robustness in the case of hardware failures or resource capacity limits.

Extending our prototype. Supporting partial reconfiguration, enabling alternative queuing strategies, providing real-time availability metrics, and supporting task DAGs remains to be done. Modern FPGAs also include more complex subsystems in their silicon, including processors, PCIe interfaces, and, more recently, networking functions [19]. Exploiting embedded processors to create a more capable and flexible control plane would enable many of the above capabilities without impacting data plane performance. It is also feasible for new FPGAs to be designed that integrate switching functionality and request assembly pipelines, resulting in a single-chip solution coupling ASIC-level switching with programmable computing for further efficiency and performance gains.

6 Conclusions

Building on the success of INC in a constrained datacenter environment, we have proposed *In-Network Acceleration* through a stream-oriented request-level abstraction for general L7 in-network computing on FPGAs. This paradigm would enable INC to emerge more widely, with potential transformational benefits for a range of emerging distributed applications in cognitive cities and connected infrastructure. FPGAs offer the flexibility to evolve deployed infrastructure with emerging applications and novel networking approaches through augmentation, thereby de-risking hardware deployment. We have demonstrated a prototype and discussed some key challenges to be explored by researchers in enabling this computing paradigm.

Acknowledgments

This publication is based upon work supported by King Abdullah University of Science and Technology (KAUST) under Award No. ORFS-CRG11-2022-5017.

References

[1] [n. d.]. NVIDIA Scalable Hierarchical Aggregation and Reduction Protocol (SHARP). https://docs.nvidia.com/networking/display/sharpv300

[2] Ashkan Aghdai, Michael I-C Wang, Yang Xu, Charles H-P Wen, and H Jonathan Chao. 2019. In-network congestion-aware load balancing at transport layer. In *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*.

[3] Alibaba Cloud. [n. d.]. Retired ECS instance types: f3, FPGA-accelerated compute-optimized instance family. https://www.alibabacloud.com/help/en/ecs/user-guide/retired-instance-types?spm=a2c63.p38356.0.0.6f872f09BQQOot#F3

[4] Amazon Web Services. [n. d.]. Amazon EC2 F1 Instances. https://aws.amazon.com/ec2/instance-types/f1/

[5] Azure Infrastructure Blog. 2024. *Enhancing Infrastructure Efficiency with Azure Boost DPU*. https://techcommunity.microsoft.com/blog/azureinfrastructureblog/enhancing-infrastructure-efficiency-with-azure-boost-dpu/4298901 Accessed: 2025-05-29.

[6] Marco Bacis, Rolando Brondolin, and Marco D Santambrogio. 2020. BlastFunction: an FPGA-as-a-service system for accelerated serverless computing. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*. 852–857.

[7] Marco Spaziani Brunella, Giacomo Belocchi, Marco Bonola, Salvatore Pontarelli, Giuseppe Siracusano, Giuseppe Bianchi, Aniello Cammarano, Alessandro Palumbo, Luca Petrucci, and Roberto Bifulco. 2022. hXDP: Efficient software packet processing on FPGA NICs. *Commun. ACM* 65, 8 (2022), 92–100.

[8] ByteDance. 2024. Libtpa: A DPDK-based userspace TCP stack. https://github.com/bytedance/libtpa Accessed: Mar. 7, 2025.

[9] Adrian M Caulfield, Eric S Chung, Andrew Putnam, Hari Angepat, Jeremy Fowers, Michael Haselman, Stephen Heil, Matt Humphrey, Puneet Kaur, Joo-Young Kim, Daniel Lo, Todd Massengill, Kalin Ovtcharov, Lisa Papamichael, Michael andWoods, Sitaram Lanka, Derek Chiou, and Doug Burger. 2016. A cloud-scale acceleration architecture. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–13.

[10] Jianguo Chen, Kenli Li, Qingying Deng, Keqin Li, and S Yu Philip. 2019. Distributed deep learning model for intelligent video surveillance systems with edge computing. *IEEE Transactions on Industrial Informatics* (2019).

[11] Eric Chung, Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Adrian Caulfield, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Maleen Abeydeera, Logan Adams, Hari Angepat, Christian Boehn, Derek Chiou, Oren Firestein, Alessandro Forin, Kang Su Gatlin, Mahdi Ghandi, Stephen Heil, Kyle Holohan, Ahmad El Husseini, Tamas Juhasz, Kara Kagi, Ratna K. Kovvuri, Sitaram Lanka, Friedel van Megen, Dima Mukhortov, Prerak Patel, Brandon Perez, Amanda Grace Rapsang, Steven K. Reinhardt, Bita Darvish Rouhani, Adam Sapek, Raja Seera, Sangeetha Shekar, Balaji Sridharan, Gabriel Weisz, Lisa Woods, Phillip Yi Xiao, Dan Zhang, Ritchie Zhao, and Doug Burger. 2018. Serving DNNs in real time at datacenter scale with project brainwave. *IEEE Micro* 38, 2 (2018), 8–20.

[12] Jason Cong, Jason Lau, Gai Liu, Stephen Neuendorffer, Peichen Pan, Kees Vissers, and Zhiru Zhang. 2022. FPGA HLS today: successes, challenges, and opportunities. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 15, 4 (2022), 1–42.

[13] Ryan A Cooke and Suhaib A Fahmy. 2020. Characterizing latency overheads in the deployment of FPGA accelerators. In *Proceedings of the International Conference on Field-Programmable Logic and Applications (FPL)*. 347–352.

[14] Ryan A Cooke and Suhaib A Fahmy. 2020. A model for distributed in-network and near-edge computing with heterogeneous hardware. *Future Generation Computer Systems* 105 (2020), 395–409.

[15] Microsoft Corporation and Broadcom Corporation. 2019. Project Zipline Top Micro Architecture Specification. https://github.com/opencomputeproject/Project-Zipline/blob/master/specs/Project_Zipline_Top_Micro_Architecture_Specification.docx

[16] Huynh Tu Dang, Pietro Bressana, Han Wang, Ki Suh Lee, Noa Zilberman, Hakim Weatherspoon, Marco Canini, Fernando Pedone, and Robert Soulé. 2020. P4xos: Consensus as a network service. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1726–1738.

[17] Huynh Tu Dang, Marco Canini, Fernando Pedone, and Robert Soulé. 2016. Paxos made switch-y. *ACM SIGCOMM Computer Communication Review* 46, 2 (2016), 18–24.

[18] Rajdeep Das and Alex C Snoeren. 2020. Enabling active networking on RMT hardware. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*. 175–181.

[19] Jaideep Dastidar, David Riddoch, Jason Moore, Steven Pope, and Jim Wesselkamper. 2023. AMD 400G Adaptive SmartNIC SoC–Technology Preview. *IEEE Micro* (2023).

[20] Dong Du, Qingyuan Liu, Xueqiang Jiang, Yubin Xia, Binyu Zang, and Haibo Chen. 2022. Serverless computing on heterogeneous computers. In *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Pperating Systems (ASPLOS)*. 797–813.

[21] Suhaib A Fahmy, Kizheppatt Vipin, and Shanker Shreejith. 2015. Virtualized FPGA accelerators for efficient cloud computing. In *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 430–435.

[22] Matthias Finger and Edy Portmann. 2016. *Towards Cognitive Cities: Advances in Cognitive Computing and its Application to the Governance of Large Urban Systems*. Springer, Switzerland.

[23] Alex Forencich, Alex C Snoeren, George Porter, and George Papen. 2020. Corundum: An open-source 100-Gbps NIC. In *Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 38–46.

[24] Google Cloud. 2023. Titanium underpins Google's workload-optimized infrastructure. https://cloud.google.com/blog/products/compute/titanium-underpins-googles-workload-optimized-infrastructure Accessed: 2025-05-29.

[25] Stephen Ibanez, Gordon Brebner, Nick McKeown, and Noa Zilberman. 2019. The P4->NetFPGA workflow for line-rate packet processing. In *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. 1–9.

[26] Zsolt István, Gustavo Alonso, Michaela Blott, and Kees Vissers. 2015. A hash table for line-rate data processing. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 8, 2 (2015), 1–15.

[27] Zsolt István, Gustavo Alonso, and Ankit Singla. 2018. Providing Multi-tenant Services with FPGAs: Case Study on a Key-Value Store. In *28th International Conference on Field Programmable Logic and Applications, FPL 2018, Dublin, Ireland, August 27-31, 2018*. IEEE Computer Society, 119–124.

[28] Zsolt István, David Sidler, and Gustavo Alonso. 2017. Caribou: Intelligent distributed storage. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1202–1213.

[29] Zsolt István, David Sidler, Gustavo Alonso, and Marko Vukolic. 2016. Consensus in a box: Inexpensive coordination in hardware. In *Proceedings of the USENIX Symposium on Networked Systems Design and*

Implementation (NSDI). 425–438.

[30] Abhishek Kumar Jain, Douglas L Maskell, and Suhaib A Fahmy. 2021. Coarse Grained FPGA Overlay for Rapid Just-In-Time Accelerator Compilation. *IEEE Transactions on Parallel and Distributed Systems* 33, 6 (2021), 1478–1490.

[31] Matthias Jasny, Lasse Thostrup, Tobias Ziegler, and Carsten Binnig. 2022. P4db-the case for in-network oltp. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. 1375–1389.

[32] Theo Jepsen, Alberto Lerner, Fernando Pedone, Robert Soulé, and Philippe Cudré-Mauroux. 2021. In-network support for transaction triaging. *Proceedings of the VLDB Endowment* 14, 9 (2021), 1626–1639.

[33] Tao Ji, Rohan Vardekar, Balajee Vamanan, Brent E Stephens, and Aditya Akella. 2025. {MTP}: Transport for {In-Network} Computing. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 959–977.

[34] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. 2017. Netcache: Balancing key-value stores with fast in-network caching. In *Proceedings of the Symposium on operating systems principles (SOSP)*. 121–136.

[35] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the International Symposium on Computer Architecture*. 1–12.

[36] Dario Korolija, Timothy Roscoe, and Gustavo Alonso. 2020. Do OS abstractions make sense on FPGAs?. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 991–1010.

[37] ChonLam Lao, Yanfang Le, Kshiteej Mahajan, Yixi Chen, Wenfei Wu, Aditya Akella, and Michael Swift. 2021. ATP: In-network Aggregation for Multi-tenant Learning. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

[38] Bojie Li, Zhenyuan Ruan, Wencong Xiao, Yuanwei Lu, Yongqiang Xiong, Andrew Putnam, Enhong Chen, and Lintao Zhang. 2017. Kv-direct: High-performance in-memory key-value store with programmable nic. In *Proceedings of the Symposium on Operating Systems Principles (SOSP)*. 137–152.

[39] Xiaozhou Li, Raghav Sethi, Michael Kaminsky, David G Andersen, and Michael J Freedman. 2016. Be fast, cheap and in control with {SwitchKV}. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 31–44.

[40] Jiaxin Lin, Kiran Patel, Brent E Stephens, Anirudh Sivaraman, and Aditya Akella. 2020. PANIC: A High-Performance programmable NIC for multi-tenant networks. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 243–259.

[41] Will Lin, Yizhou Shan, Ryan Kosta, Arvind Krishnamurthy, and Yiying Zhang. 2024. SuperNIC: An FPGA-Based, Cloud-Oriented SmartNIC. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*. 130–141.

[42] Dongqi Liu, Haolan Liang, Xiangjun Zeng, Qiong Zhang, Zidong Zhang, and Minhong Li. 2022. Edge computing application, architecture, and challenges in ubiquitous power internet of things. *Frontiers in Energy Research* 10 (2022), 850252.

[43] Rui Ma, Evangelos Georganas, Alexander Heinecke, Sergey Gribok, Andrew Boutros, and Eriko Nurvitadhi. 2022. FPGA-based AI smart NICs for scalable distributed AI training systems. *IEEE Computer Architecture Letters* 21, 2 (2022), 49–52.

[44] Tianle Mai, Sahil Garg, Haipeng Yao, Jiangtian Nie, Georges Kaddoum, and Zehui Xiong. 2021. In-network intelligence control: Toward a self-driving networking architecture. *IEEE Network* 35, 2 (2021), 53–59.

[45] Nick McKeown. 2023. Intel's Tofino Update to the P4 Community. https://groups.google.com/a/lists.p4.org/g/p4-announce/c/frXi_jjmawE. Message to the P4 Announce mailing list.

[46] Meta Engineering. 2021. *OCP Summit 2021: How we're building open and efficient data center networks*. https://engineering.fb.com/2021/11/09/data-center-engineering/ocp-summit-2021/ Accessed: 2025-05-29.

[47] Nuno Pereira, Anthony Rowe, Michael W Farb, Ivan Liang, Edward Lu, and Eric Riebling. 2021. ARENA: The augmented reality edge networking architecture. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 479–488.

[48] Andrew Putnam, Adrian M Caulfield, Eric S Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck, Stephen Heil, Amir Hormati, Joo-Young Kim, Sitaram Lanka, James Larus, Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, and Doug Burger. 2014. A reconfigurable fabric for accelerating large-scale datacenter services. *ACM SIGARCH Computer Architecture News* 42, 3 (2014), 13–24.

[49] Sergi Rene, Onur Ascigil, Ioannis Psaras, and George Pavlou. 2021. A congestion control framework based on in-network resource pooling. *IEEE/ACM Transactions on Networking* 30, 2 (2021), 683–697.

[50] Behzad Salami, Gorker Alp Malazgirt, Oriol Arcas-Abella, Arda Yurdakul, and Nehir Sonmez. 2017. AxleDB: A novel programmable query processing platform on FPGA. *Microprocessors and Microsystems* 51 (2017), 142–164.

[51] Amedeo Sapio, Ibrahim Abdelaziz, Abdulla Aldilaijan, Marco Canini, and Panos Kalnis. 2017. In-network computation is a dumb idea whose time has come. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*. 150–156.

[52] Amedeo Sapio, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan Ports, and Peter Richtarik. 2021. Scaling Distributed Machine Learning with In-Network Aggregation. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

[53] Henry N Schuh, Weihao Liang, Ming Liu, Jacob Nelson, and Arvind Krishnamurthy. 2021. Xenic: SmartNIC-accelerated distributed transactions. In *Proceedings of the ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*. 740–755.

[54] Summit Shrestha, Zhengquan Li, Khairul Mottakin, Zheng Song, and Qiang Zhu. 2023. From Tight Coupling to Flexibility: A Digital Twin Middleware Layer for the ShakeAlert System. In *Proceedings of the IEEE/ACM Symposium on Edge Computing (SEC)*. 313–318.

[55] David Sidler, Zsolt István, and Gustavo Alonso. 2016. Low-latency TCP/IP stack for data center applications. In *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*.

[56] Brent E Stephens, Darius Grassi, Hamidreza Almasi, Tao Ji, Balajee Vamanan, and Aditya Akella. 2021. TCP is harmful to in-network computing: designing a message transport protocol (MTP). In *Proceedings*

of the ACM Workshop on Hot Topics in Networks. 61–68.

[57] Ian Taras and Jason H Anderson. 2019. Impact of FPGA architecture on area and performance of CGRA overlays. In *Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 87–95.

[58] David L Tennenhouse and David J Wetherall. 1996. Towards an active network architecture. *ACM SIGCOMM Computer Communication Review* 26, 2 (1996), 5–17.

[59] Yuta Tokusashi, Huynh Tu Dang, Fernando Pedone, Robert Soulé, and Noa Zilberman. 2019. The case for in-network computing on demand. In *Proceedings of the European Conference on Computer Systems (EuroSys)*. 1–16.

[60] Stephen M Trimberger and Jason J Moore. 2014. FPGA security: Motivations, features, and applications. *Proc. IEEE* 102, 8 (2014), 1248–1265.

[61] Naveen Vedula, Reza Hojabr, Ahmad Khonsari, and Arrvindh Shriraman. 2021. X-Layer: Building Composable Pipelined Dataflows for Low-Rank Convolutions. In *International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE, 103–115.

[62] Petr Velan and Viktor Puš. 2015. High-density network flow monitoring. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 996–1001.

[63] Prahlad Venkatapuram, Zhao Wang, and Chandra Mallipedi. 2020. Custom silicon at Facebook: A datacenter infrastructure perspective on video transcoding and machine learning. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*. 199–202.

[64] Kizheppatt Vipin and Suhaib A Fahmy. 2018. FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications. *Comput. Surveys* 51, 4 (2018), 1–39.

[65] Greg Watson, Nick McKeown, and Martin Casado. 2006. NetFPGA: A tool for network research and education. In *Workshop on Architectural Research using FPGA Platforms (WARFP)*, Vol. 3.

[66] Nicholas Weaver, Vern Paxson, and Jose M Gonzalez. 2007. The Shunt: an FPGA-based accelerator for network intrusion prevention. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*. 199–206.

[67] Tong Wu, Haipeng Yao, Wenji He, Zunliang Wang, Tianle Mai, Zehui Xiong, and Song Guo. 2023. Low-cost network measurement through intelligent in-band network telemetry orchestration. In *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 4326–4331.

[68] Ziyi Yang, Krishnan B. Iyer, Yixi Chen, Ran Shu, Zsolt István, Marco Canini, and Suhaib A. Fahmy. 2025. OffRAC: Offloading Through Remote Accelerator Calls. arXiv:2504.04404 [cs.NI] https://arxiv.org/abs/2504.04404

[69] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G Andersen. 2011. SCION: Scalability, control, and isolation on next-generation networks. In *Proceedings of the IEEE Symposium on Security and Privacy*. 212–227.

[70] Zhipeng Zhao, Hugo Sadok, Nirav Atre, James C Hoe, Vyas Sekar, and Justine Sherry. 2020. Achieving 100gbps intrusion prevention on a single server. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 1083–1100.

[71] Xiangfeng Zhu, Weixin Deng, Banruo Liu, Jingrong Chen, Yongji Wu, Thomas Anderson, Arvind Krishnamurthy, Ratul Mahajan, and Danyang Zhuo. 2023. Application Defined Networks. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*. 87–94.

[72] Noa Zilberman, Matthew Grosvenor, Diana Andreea Popescu, Neelakandan Manihatty-Bojan, Gianni Antichi, Marcin Wójcik, and Andrew W Moore. 2017. Where has my time gone?. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)*. 201–214.