Demo: Programmable Packet Scheduling on FPGA-based Switches

Mufaddal Enayath Hussain
SMILE
Paris, France
mufaddal.enayathhussain@smile.fr

Mario Patetta
CEDRIC, *Cnam*
Paris, France
mario.patetta@cnam.fr

Jonathan Rivalan
SMILE
Paris, France
jonathan.rivalan@smile.fr

***Abstract*—We present an FPGA prototype of programmable traffic manager capable of dynamic adjusting its scheduling behaviour, enabling tailored enforcing of Quality-of-Service (QoS) guarantees. The proposed design is integrated within the Corundum NIC framework and deployed on an AMD Alveo U50 FPGA board that operates with 4x10 Gbps ports. The demo shows the runtime configurability of both packet classification and scheduling policies—including weighted bandwidth sharing, strict priority and traffic shaping.**

***Index Terms*—Traffic management, programmable packet scheduler, FPGA, programmable data plane, quality of service.**

I. Introduction

The ongoing research on 6G envisions high-performance and adaptive networks, capable of supporting services with stringent throughput, latency and reliability requirements. Moreover, recent disruptive phenomena such as the proliferation of Generative AI applications underscore an increasing degree of variability in the expected traffic demands of future networks. These trends motivate the adoption of programmable packet processing hardware, enabling network operators to flexibly redefine data plane functionality bypassing the long design cycles of novel silicon chips.

Among data plane operations, packet scheduling is critical to enforce Quality-of-Service (QoS) guaranties. Despite this, both conventional and programmable data plane devices—such as the Portable Switch Architecture [1]—employ fixed-function scheduling disciplines, mostly based on strict priority, Deficit Round Robin (DRR) [2] and traffic shaping. Though capable of modifying their parameters, operators cannot introduce new ones—most notably, delay-sensitive and dynamic priority algorithms (e.g., pFabric [3], PIAS [4]). Programmable schedulers have been proposed to overcome these limitations, allowing scheduling policies to be flexibly expressed even in fixed-function hardware [5], [6], [7], [8].

Most of the related work focuses on design and hardware implementation of the proposed architectures, but fall short in terms of realistic evaluation. Some validate their design using software simulations [8], while others use synthetic traffic generated internally in the FPGA [6].

This paper presents a packet switch featuring per-egress-port programmable schedulers. We implement it on AMD Alveo U50 FPGA, using the Corundum NIC framework [9], allowing to test the scheduler in a realistic environment with live traffic. The scheduler consists of two main components:

- Packet Dispatcher (PD): sends incoming packets to First-In First-Out (FIFO) queues based on dedicated lookup tables, implemented using Content Addressable Memories (CAMs).
- Traffic Manager (TM): orchestrates transmissions from multiple FIFO queues, leveraging on the Push-In-Extract-Out (PIEO) programmable scheduler architecture [6].

Using the board's PCIe interface, a controller running in the host machine can modify rules and scheduling parameters at run-time.

This paper focuses on the practical demonstration and evaluation of the scheduler using an FPGA prototype.

The remainder of this paper is organized as follows. Section II describes the architecture of the proposed programmable scheduler. Section III outlines the experimental setup and demonstration workflow. Finally, Section IV concludes the paper and presents future perspectives.

II. Scheduler Architecture

Figure 1 illustrates the block diagram of the programmable scheduler, characterized by n ingress and m egress ports. The design is logically divided into three subsystems: the input stage, the buffer stage and the output stage.

In the input stage, PDs extracts header fields—such as packet length and Class-of-Service (CoS) identifiers—from incoming packets, and assigns them to the appropriate FIFO queues in the buffer stage. Each dispatcher manages an independent set of FIFOs, which are further organized into groups corresponding to the egress ports. Within each group, different queues represent distinct CoS. Finally, at the output stage, the TMs determine the order of queues to schedule for transmission toward each egress port, as well as the amount of data to send before moving to the next queue.

In the remaining of this section, we describe in detail the building blocks of the input and output stages, i.e., the PD and TM.

A. Packet Dispatcher

PDs are responsible for classifying incoming packets and assigning them to appropriate FIFO queues, based on their egress port and CoS. Figure 2 shows the functional blocks of the PD module. When a packet arrives, a parser module extracts metadata from predefined bit locations. Some metadata

979-8-3315-5141-4/25/$31.00 ©2025 IEEE

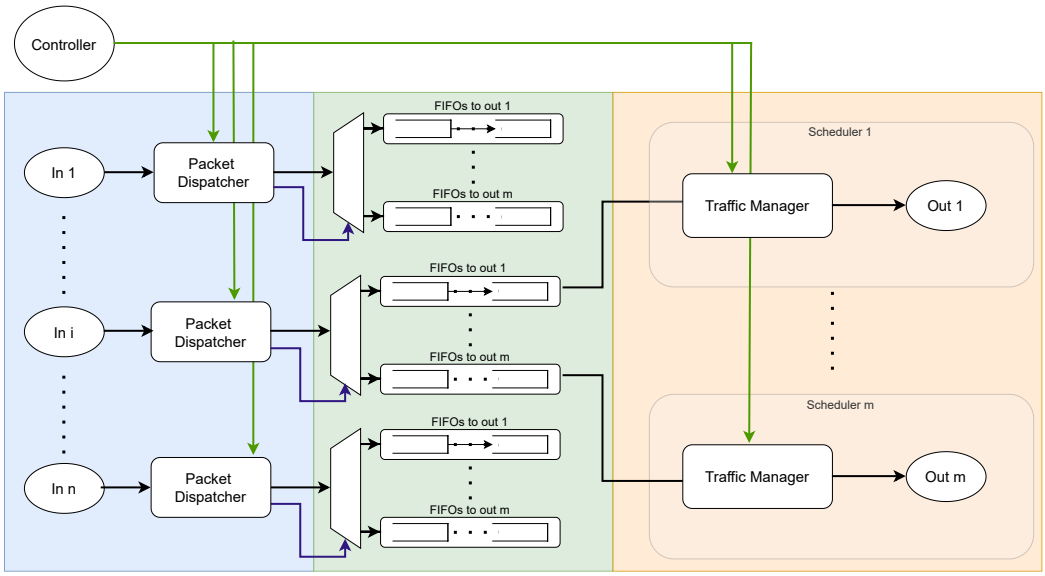

Fig. 1. Programmable scheduler blocker diagram

is used to query a CAM unit, while the rest is appended to the packet for later use by the schedulers at the output stage. The CAM output identifies the FIFO queue to forward the packet to, based on the provided metadata, thus determining its egress port and CoS.

Finally, the CAM manager block offers a control interface to configure the content of the CAM at runtime, ensuring that updates occur only during idle periods to avoid disrupting data plane operations.

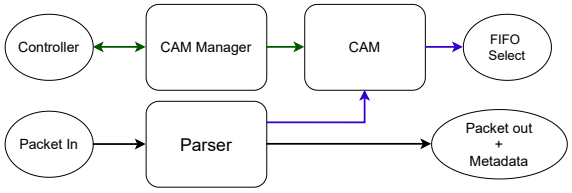

Fig. 2. Packet dispatcher block diagram.

B. Traffic Manager

In the output stage, packet transmission toward each egress port is orchestrated by a TM. When the corresponding port is ready and packets are waiting in eligible buffers, the TM determines their transmission order and rate according to the configured scheduling parameters. Figure 3 shows that the TM is composed of two subsystems: the parameter store and the FIFO scheduler.

The parameter store consists of a collection of registers that hold per-FIFO scheduling parameters and auxiliary packet-drop counters. The scheduler, which represents the core component of the TM, is structured following the PIEO framework [6]. It comprises three key elements:

1) PIEO: maintains references to FIFO queues and sorts them according to their rank;
2) Pre-enqueue function: determines whether a non-empty FIFO is eligible for insertion into the PIEO queue based on shaping parameters and current transmission rates. Upon insertion, it assigns ranks, based on the priorities set in the parameter store.
3) Post-dequeue function: once a FIFO reference is extracted from the PIEO, it determines how many packets or bytes to transmit.

In the proposed design, supported scheduling disciplines include strict priority scheduling, DRR and traffic shaping. The related parameters maintained in the parameter store can be updated at run-time, as the CAM contents in the PD.

III. Technical Demonstration

The proposed architecture is deployed on an AMD Alveo U50 FPGA board, using the corundum NIC framework. Consequently, the FPGA operates as a programmable switch capable of executing dynamic scheduling. The prototype features: (i) four ingress and four egress interfaces operating at 10 Gbps each; (ii) 64 FIFOs, divided into groups of 16 FIFOs per egress port, further divided based on ingress port and CoS; (iii) VLAN-based service classification.

This section describes the proposed technical demonstration. First, we cover the software environment, which establishes the tools and monitoring framework used in the experiment. Then, we discuss the tests conducted to validate the functionality of the scheduler.

A. Software Environment

In the experimental setup, traffic is generated using host-based tools such as iperf and pktgen, which create VLAN-tagged UDP flows at a constant bitrate. A custom driver configures the FPGA via the PCIe interface, updating CAM rules and scheduling parameters. Monitoring is performed using PMACCT, an open-source network monitoring tool that collects and aggregates packet statistics by CoS (VLAN) for each output port. Finally, collected metrics are visualized through Grafana dashboards.

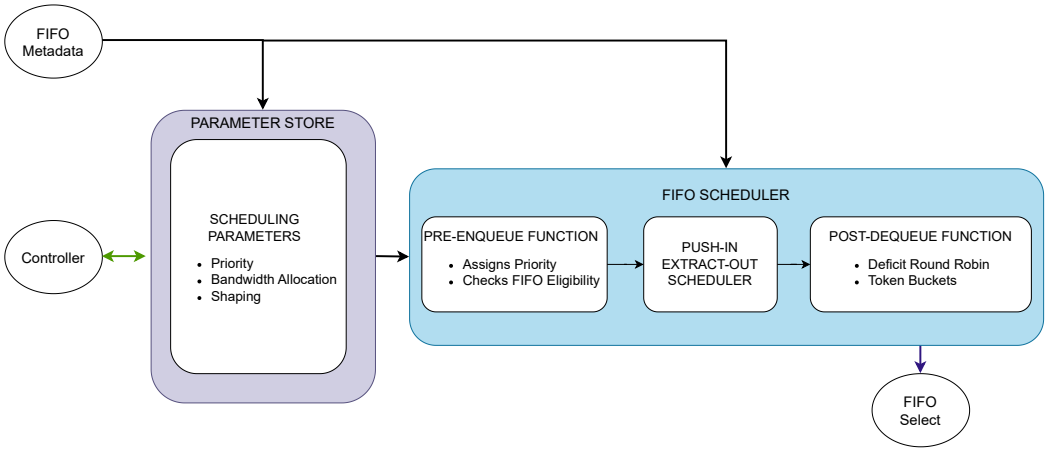

Fig. 3. Traffic manager block diagram

B. Demonstration Procedure

Initially, the PD and TM are configured to accept all incoming CoS and perform fair bandwidth sharing among them—i.e., no shaping, same priority and same DRR quantum assigned to all FIFOs. Accordingly, traffic streams with different VLAN identifiers are activated at a constant bitrate—making sure to saturate the 10 Gbps throughput of a single FPGA egress port—and collected metrics are displayed on the dashboard, showing that the bandwidth is effectively shared fairly.

Next, we use the driver to modify CAM rules at runtime, such that some of the flows are no longer matched—and are therefore dropped by the PD—while others are forwarded to a different egress port.

Finally, scheduling parameters are adjusted to showcase the supported scheduling disciplines. In particular, DRR, static priority, and traffic shaping are alternated at runtime, along with their combinations.

IV. Conclusion

The evolution toward 6G networks, characterised by stringent performance requirements and rapidly changing traffic patterns, demands greater flexibility in the data plane. In this context, programmable scheduling architectures constitute a promising technology for supporting diverse, evolving services without incurring the cost and delay of new hardware development.

This paper presents a practical demonstration of a programmable packet scheduler built on the Corundum FPGA NIC framework, integrating a CAM-based packet dispatcher and a PIEO-based traffic manager capable of expressing scheduling policies commonly employed in state-of-the-art switches. Both modules support run-time reconfiguration via a custom driver.

By deploying the proposed programmable packet scheduler as an FPGA prototype, this demo showcases the feasibility of programmable scheduling at line rate.

Future work can be envisioned to enhance the three main subsystems. The PD can be generalised to use any combination of packet header fields—possibly leveraging on hash functions to compress the CAM input space. The buffer stage can be enhanced by implementing shared memory queues, capable of dynamically adapting their size based on traffic conditions. Finally, the TM can be further refined, adding support for more advanced disciplines, including hierarchical scheduling, asynchronous priority escalation and starvation prevention mechanisms.

Acknowledgements

This work was funded by the ANR HEIDIS (https://heidis.roc.cnam.fr; contract nb: ANR-21-CE25-0019), the IPCEI ME/CT Orange (contract nb: DOS0239248/00), and the France 2030 IE6 (Internet of Edges for 6G; contract nb. DOS0223931) projects.

References

[1] The P4.org Architecture Working Group, "$P4_{16}$ Portable Switch Architecture (PSA)," 2021. Accessed: Aug. 29, 2025.

[2] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375–385, 1996.

[3] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 435–446, 2013.

[4] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "{Information-Agnostic} flow scheduling for commodity data centers," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pp. 455–468, 2015.

[5] A. Sivaraman, S. Subramanian, M. Alizadeh, S. Chole, S.-T. Chuang, A. Agrawal, H. Balakrishnan, T. Edsall, S. Katti, and N. McKeown, "Programmable packet scheduling at line rate," in *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, (New York, NY, USA), p. 4457, Association for Computing Machinery, 2016.

[6] V. Shrivastav, "Fast, scalable, and programmable packet scheduler in hardware," in *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, (New York, NY, USA), p. 367379, Association for Computing Machinery, 2019.

[7] N. K. Sharma, C. Zhao, M. Liu, P. G. Kannan, C. Kim, A. Krishnamurthy, and A. Sivaraman, "Programmable calendar queues for high-speed packet scheduling," in *Proceedings of the 17th Usenix Conference on Networked Systems Design and Implementation*, NSDI'20, (USA), p. 685700, USENIX Association, 2020.

[8] M. Elbediwy, B. Pontikakis, A. Ghaffari, J.-P. David, and Y. Savaria, "Dr-pifo: A dynamic ranking packet scheduler using a push-in-first-out queue," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 355–371, 2024.

[9] A. Forencich, A. C. Snoeren, G. Porter, and G. Papen, "Corundum: An open-source 100-gbps nic," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 38–46, 2020.