What Is Reconfiguration Delay, Really? An End-to-End View of Photonic Switching

Deeksha P Rao
Purdue University

Vamsi Addanki
Purdue University

Abstract

Reconfiguration delay is a defining systems parameter of photonic switching. Yet, it is almost universally modeled as a single constant in network design and circuit scheduling algorithms, namely the optical switching time reported by the device. We argue that this abstraction is no longer sufficient. From an application's perspective, reconfiguration begins when a new circuit is requested and ends only when communication resumes over the new path. The latency observed by the application spans an end-to-end critical path across multiple hardware and software components, not just the optical switch.

This paper asks a simple question: what is reconfiguration delay, really? We argue that it should be understood as an end-to-end systems property rather than a device property. To make this concrete, we dissect the critical path of a circuit transition, spanning host-to-controller communication, controller processing, electrical signaling, and optical switching. Our measurements show that optical switching itself accounts for only a small fraction of the end-to-end latency, even before considering transceiver recovery and receiver-side reactions. This perspective raises a fundamental systems question: if most of the end-to-end delay lies outside the switch fabric, what should future photonic switching systems optimize? We conclude by outlining a research agenda for reducing end-to-end reconfiguration delay across the hardware–software stack.

CCS Concepts

• **Networks** → **Data center networks**; *Network control algorithms*; • **Hardware** → *Networking hardware.*

Keywords

Photonic interconnect, Optical circuit switching

ACM Reference Format:
Deeksha P Rao and Vamsi Addanki. 2026. What Is Reconfiguration Delay, Really? An End-to-End View of Photonic Switching. In *The 25th ACM Workshop on Hot Topics in Networks (HotNets '26),*



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
HotNets '26, November 16–17, 2026, Salt Lake City, UT, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-3029-0/2026/11
https://doi.org/10.1145/3847290.3859969

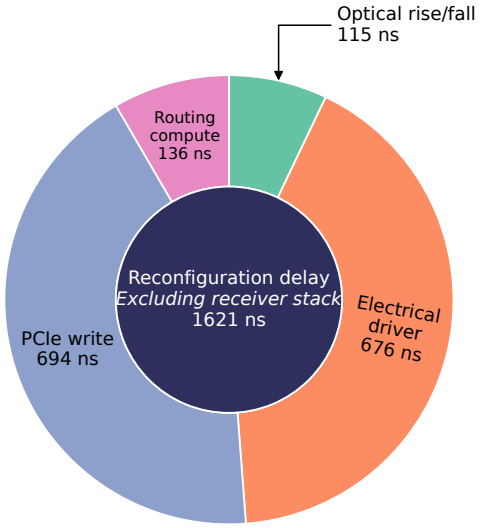


Figure 1: Reconfiguration delay breakdown excluding receiver locking, with optical rise/fall time contributing only 7%.

November 16–17, 2026, Salt Lake City, UT, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3847290.3859969

1 Introduction

The growing demand for network bandwidth, together with the slowdown of Moore's Law, has renewed interest in photonic switching, both in academia [1, 2, 18, 22, 25, 26, 40, 42, 44, 48, 55] and industry [17, 45]. Photonic switching promises high bandwidth and low power consumption, making it an attractive communication substrate for datacenters and high-performance computing [5, 12, 18, 40, 44, 48], particularly for chip-to-chip communication in scale-up systems [1, 21].

The circuit-switched nature of photonic interconnects introduces two fundamental systems questions: *(i)* what circuit topology should be established, and *(ii)* when should the topology be reconfigured? Both questions fundamentally depend on the reconfiguration delay of the interconnect [1, 8, 27, 40, 48]. Existing works typically model reconfiguration delay as a single constant determined by the switching technology, while device vendors and photonic hardware research report the intrinsic optical switching time, ranging from tens of nanoseconds for electro-optic switches [30, 39] to milliseconds for mechanical switches [37]. This paper asks a different question.

What does reconfiguration delay actually comprise in a photonic switching system?

To answer this question, we consider synchronous, source-initiated photonic switching, in which a traffic source (e.g., a GPU) requests a new circuit and a controller programs the optical interconnect [5, 26, 40]. We distinguish two intervals: the control path delay, from circuit request to light arrival, and receiver recovery, from light arrival to traffic resumption. Figure 1 traces the resulting end-to-end critical path in our testbed using nanosecond-scale electro-optic switches. A striking observation is that optical switching accounts for only ≈ 7% of the measured reconfiguration delay, even before considering the well-known problem of transceiver locking at the receiver. It covers the control-path delay (≈[1.6] μs in our prototype); receiver recovery is about six orders of magnitude larger on commodity NICs (≈1 s) and is analyzed separately in §3.6.

The control-path latency arises from stages that are rarely considered together. Before the optical switch can change state, the source must communicate the circuit request to the controller. Using a GPU connected to an FPGA controller over PCIe, this communication alone contributes ≈ 694 ns on average, making it the largest component in our testbed. The controller must then translate the requested circuit into switch configurations. In our FPGA implementation, routing an 8×8 Beneš interconnect requires ≈ 136 ns, increasing to 1.3μs for a 32×32 interconnect, nearly an order of magnitude larger than the optical switching time itself. Finally, the control signal must propagate from the controller to the optical switch before the optical transition occurs, introducing another non-negligible delay. Once the circuit has been established, the destination transceiver must reacquire lock, a well-known challenge that can add tens of microseconds [14]. We discuss this receiver-side latency separately and exclude it from the end-to-end measurements presented in Figure 1. Importantly, each individual latency, including the optical switching time itself, are neither constant or deterministic, and introduce non-negligible variance when modeling nanosoecond-scale systems. We elaborate on each of these components in Section 3 and quantify their contribution to the overall reconfiguration delay.

This perspective shifts the optimization target from the intrinsic switching time of the optical device to the end-to-end latency experienced by distributed applications. The purpose of this paper is not to claim that these individual delays are new. Rather, we place them on a common timeline to expose the critical path of photonic switch reconfiguration. Naturally, from a network design and circuit scheduling perspective, we urge the community to consider variance in reconfiguration delay as a first-class parameter, opening new avenues for research in online algorithms where the cost of reconfiguration

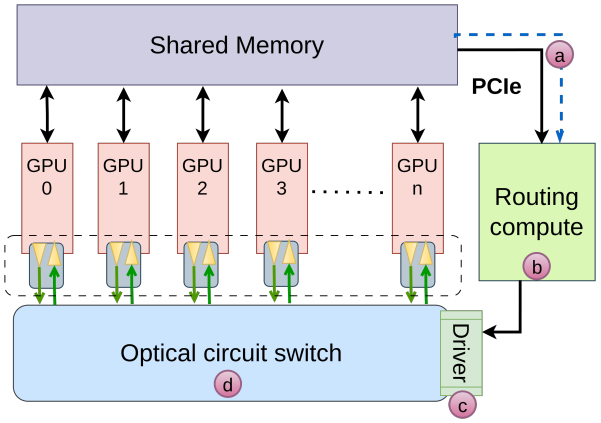


Figure 2: Hardware prototype

is partially known and only revealed after the fact. From a systems and hardware perspective, we believe it is high-time to revisit the classic problem of interconnect routing e.g., Beneš routing, and efficient algorithms that can compute switch states within a small fraction of the switching time. We discuss several future research directions at the end of the paper.

2 Hardware Prototype

As part of our broader effort to bring photonic interconnects closer to GPUs and end hosts in modern scale-up systems, we developed an end-to-end hardware prototype to study reconfiguration delay.

Figure 2 illustrates the architecture. Our prototype consists of eight NVIDIA RTX PRO 6000 GPUs, each connected to a BlueField-3 DPU over PCIe Gen5 with dedicated PCIe bandwidth. Since current GPUs cannot directly interface with optical transceivers, we treat the GPU, DPU, and optical transceiver as a single compute node. The BlueField-3 hosts the optical transceiver and operates in DPU mode, allowing forwarding rules to be programmed directly into hardware through Open vSwitch. This abstraction affects only the data path; the control path, which is the focus of this paper, remains unchanged. Each transceiver connects to the photonic interconnect through single-mode LC fiber.

The photonic interconnect is an 8 × 8 Beneš network constructed from seventeen 2×2 electro-optic switches [46]. Each switch consists of a birefringence crystal whose refractive index changes in response to an applied voltage [9, 41], thereby changing the optical path. Consequently, every optical reconfiguration requires an electrical driver circuit that controls the voltage to transition each switch between its cross and bar states.

An FPGA acts as the reconfiguration controller between the compute nodes and the optical fabric. Connected to the GPUs over PCIe Gen5 and to the optical switches through GPIO, it receives circuit requests, computes the corresponding Beneš switch configuration, and simultaneously drives all seventeen

switches using TTL control signals. This control path naturally decomposes reconfiguration into four stages: communicating the circuit request from the GPU to the FPGA, computing the switch configuration, propagating control signals to the switch drivers, and performing the optical transition.

Our prototype supports arbitrary communication topologies across the eight GPUs while allowing circuits to be reconfigured at runtime. More importantly, it exposes every stage of the end-to-end reconfiguration pipeline, allowing us to isolate and quantify the latency introduced by each stage rather than observing only the optical switching event.

3 Dissecting the Reconfiguration Delay

Figure 2 illustrates the control path responsible for establishing a new optical circuit. Every reconfiguration begins when the communication runtime determines that a new topology is required, for example, at the boundary of a collective communication step. The requested logical topology is communicated from the compute node to the switch controller, translated into individual switch configurations, and finally propagated as electrical control signals that actuate the optical switches. While the optical transition itself lasts only a few nanoseconds, every stage preceding it contributes to the end-to-end reconfiguration delay.

This section examines the control path corresponding to points ⓐ–ⓓ in Figure 2. Our controller is implemented on an AMD VCK190 Versal FPGA connected to the compute nodes through PCIe Gen5 and to the optical switches through GPIO. Upon receiving a circuit request, the FPGA computes the corresponding Beneš switch configuration and simultaneously drives the electrical interfaces of all optical switches. We next quantify the latency introduced by each stage of this pipeline.

3.1 Host-to-Controller Communication

The first stage of reconfiguration is communicating the circuit request from the compute node to the FPGA controller. The VCK190 integrates the Cache Coherent PCIe Module 4 (CPM4), which supports several PCIe communication mechanisms including Programmable I/O (PIO), Bus Master DMA (BMD), XDMA, QDMA, and AXI bridges. Since a circuit request consists of only a few bytes describing the desired topology, throughput is secondary to latency. We therefore employ PIO, which exposes the FPGA BAR space directly to the host and allows the GPU to access FPGA registers with minimal software and hardware overhead. In contrast, DMA-based approaches incur additional setup costs that dominate such small control-plane transactions.

Figure 3 shows the cumulative distribution of PCIe write latency measured between the host and the FPGA. The distribution rises sharply between 330 ns and 1 μs, with an average write latency of approximately 694 ns. More than 99% of writes complete within 1.15 μs, making

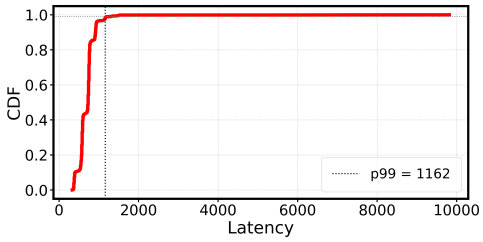


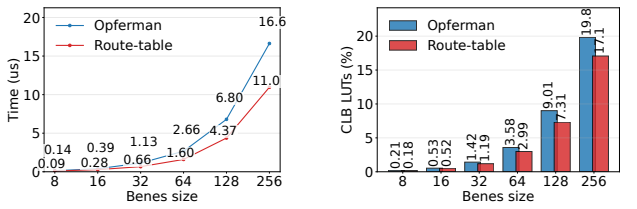
Figure 3: CDF of PCIe write latency between the compute node and the FPGA controller, with 99^{th} percentile latency of 1162 ns and average latency of 694 ns.

PCIe communication the single largest contributor to the end-to-end reconfiguration delay in our prototype.

3.2 Controller Processing

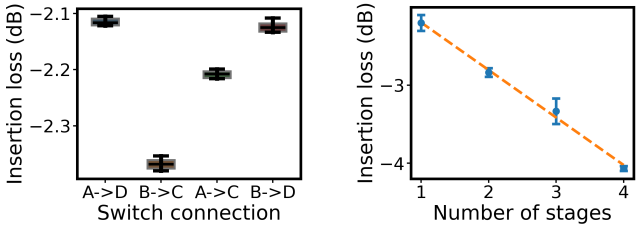
Once the circuit request reaches the FPGA, it must be translated into individual switch configurations. By routing, we refer to the process of mapping a requested logical topology to the state of every 2×2 switch in the underlying interconnect. The complexity of this translation depends on the switching architecture. Optical switching fabric are typically constructed in modular fashion using units of switching elements. Different topologies can be employed to scale an $N \times N$ from a unit 1×2 or a 2×2. Among all popular topologies include PILOSS [20], Double-Layer Network(DLN) [7], Crossbar [28] and Beneš topologies [50]. Every topology has a different number of input, output, and intermediary layers. With the increase in the number of layers, the dependency between paths reduces. And thus, the partial reconfigurability of the switching fabric increases [24]. However, it directly affects the number of switches used and thus the complexity of mapping logical topology to the switch states. While crossbar fabrics can establish connections with little or no routing computation, they require $O(N^2)$ switching elements and become prohibitively expensive at scale [11]. Practical photonic interconnects therefore increasingly adopt multistage Rearrangeably Non-Blocking (RNB) fabrics such as Beneš, which realize arbitrary permutations using only $O(N \log N)$ switching elements at the cost of online routing computation [29, 30, 36, 39]. Other scalable multistage switching architectures, including Clos and related variants, pose similar configuration problems that must be solved before the optical fabric can be reconfigured [11]. Consequently, controller computation itself becomes part of the critical reconfiguration path.

Figure 4a reports the hardware execution time of two FPGA implementations of Beneš routing: a graph-coloring-based implementation following Opferman's algorithm [36] and a route-table-based implementation [34]. Figure 4b shows the corresponding FPGA resource utilization. For an 8×8 interconnect used in our prototype, routing completes in approximately 116–160 ns, already exceeding the intrinsic optical



(a) Hardware execution time **(b) FPGA resource utilization**

Figure 4: Routing computation time and hardware resource utilization for Beneš interconnects shows non-negligible overhead in the critical path.



(a) Insertion loss of a switch **(b) Insertion loss with Scaling**

Figure 5: Insertion loss with different switch states and varied number of stages.

switching time of many electro-optic devices. As the switch fabric scales, however, routing latency grows rapidly [35, 53]. For a 256×256 Beneš network, the graph-coloring implementation requires 16.6μs, while the route-table implementation requires 11.0 μs. The graph-coloring implementation also consumes approximately 20% of the FPGA LUTs, compared to 17% for the route-table implementation.

The key observation is not which routing implementation is faster, but that routing itself quickly becomes a first-order contributor to reconfiguration delay. While optical switching technology has steadily reduced device switching times to only a few nanoseconds, controller computation continues to grow with the size and complexity of the switching fabric. As photonic interconnects scale, the latency of computing switch configurations can dominate the intrinsic optical switching time, making controller processing an increasingly important target for end-to-end optimization.

When is routing on the critical path? Many workloads that motivate runtime reconfiguration necessitate online computation: in data-dependent MoE all-to-all traffic [26], multi-tenant job arrivals, or failures, the next configuration is known only at runtime. This places routing directly on the critical path. However, routing is not always on the critical path: demand-oblivious schedules [4, 33] and collectives whose communication patterns are known at compile time [22, 40] allow switch states to be precomputed. We discuss this tradeoff, and the limits of precomputation at scale, in §4.

3.3 Insertion Loss

Routing algorithms are often formulated purely in terms of connectivity. In practice, however, the optical properties of the interconnect introduce additional constraints. Among these, insertion loss (IL) is particularly important because it determines whether a candidate path satisfies the available optical power budget.

Insertion loss becomes increasingly important as the switch fabric scales, and its growth depends strongly on the topology connecting the underlying switching elements. The topology determines how many elements each optical path traverses and whether this number varies across source–destination pairs. Fortunately, Beneš networks possess an important property: every input-output path traverses the same number of switching stages. As a result, all source-destination pairs experience nearly identical path loss, allowing routing to be performed without explicitly accounting for insertion loss. Figure 5a shows the measured insertion loss of a representative 2×2 switch. Although different switching states exhibit small variations in insertion loss, Figure 5b shows that the end-to-end insertion loss scales approximately linearly with the number of switching stages. Since every path in a Beneš network traverses the same number of stages, the resulting path loss remains nearly uniform across all routes.

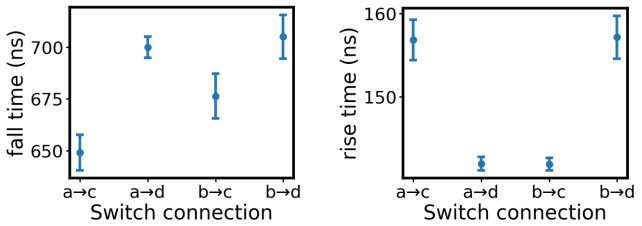
This property does not generally hold for other photonic switching architectures. Grid-based and irregular topologies admit multiple paths with different lengths and different numbers of switching elements, resulting in different insertion loss for different routes [19, 23]. Routing algorithms for such architectures must jointly consider connectivity and optical feasibility , increasing the complexity of controller processing and, consequently, the end-to-end reconfiguration latency.

3.4 Electrical Signaling

The final stage of the control path is actuating the optical switches. This stage bridges the digital control plane and the optical fabric, translating controller outputs into the physical signals required by the switching elements. Every photonic switching technology requires such an interface between the controller and the optical device, making it an unavoidable component of end-to-end reconfiguration delay.

The design of the actuation circuitry depends on the underlying switching technology. MEMS switches require precision actuators and high-resolution DACs to steer micro-mirrors with low settling times [31]. Thermo-optic and magneto-optic switches require drivers capable of generating controlled thermal and magnetic fields, respectively. Electro-optic switches instead require high-speed voltage drivers with fast rise and fall times. Although recent electro-optic devices demonstrate intrinsic switching times of only a few nanoseconds [20, 29, 52], and emerging devices project picosecond-scale operation [15], the latency of the driver circuitry often exceeds that of the optical switching element itself. Faster

switching devices alone do not necessarily reduce the end-to-end reconfiguration delay experienced by applications.



(a) Fall time **(b) Rise time**

Figure 6: Electrical-to-Optical rise/fall time for different switch configurations: $a \rightarrow c$ & $b \rightarrow d$ represent bar connection and $a \rightarrow d$ & $b \rightarrow c$ represent the cross connection.

Our electro-optic switches operate at voltages substantially higher than the FPGA I/O levels. Each FPGA control signal first passes through a dedicated level translator before reaching the switch driver. The level translators contribute an additional 20–30 ns of latency, while the complete electrical signaling stage, including the driver circuitry, accounts for approximately 41% of the measured end-to-end reconfiguration delay in our prototype.

Figure 7 shows the electrical-to-optical delay, defined as the time between issuing the electrical control signal and the onset of the optical transition, across the four possible switching transitions. We observe that the electrical-to-optical delay for optical rise transitions ranges from approximately 140 to 160 ns, whereas the corresponding delay for optical fall transitions ranges from approximately 540 to 600 ns. Figure 6 summarizes the distribution of these delays across all four transitions, showing that the electrical-to-optical delay exhibits non-negligible variation that should be accounted for when modeling nanosecond-scale photonic switching.

The driver circuitry also bounds the maximum reconfiguration frequency of the optical fabric. Even if the optical switching element itself operates in only a few nanoseconds, successive reconfigurations cannot occur faster than the driver can reliably actuate the switch. This observation extends beyond electro-optic devices: regardless of the underlying switching technology, the circuitry responsible for controlling the switching element forms an integral stage of the end-to-end reconfiguration pipeline.

3.5 Optical Switching

The final stage of reconfiguration is the optical transition itself. The latency of this stage is determined by the underlying switching mechanism. Free-space technologies such as 3-D MEMS [51] and LCOS [47] typically exhibit switching times ranging from microseconds to milliseconds, whereas integrated electro-optic devices achieve transitions in only a few nanoseconds [10, 29]. Our prototype employs an electro-optic switch, enabling nanosecond-scale optical transitions.

Figure 7 shows oscilloscope traces for the four possible state transitions of the 2×2 switch. Across all transitions, the optical rise and fall times remain between approximately 5 ns and 151 ns. The slight variation across transition types arises from the switching element's physical characteristics, but remains within the nanosecond regime. These measurements confirm that the optical transition itself contributes only a small fraction of the overall reconfiguration delay in our prototype. The dominant latency is outside the optical signal itself!

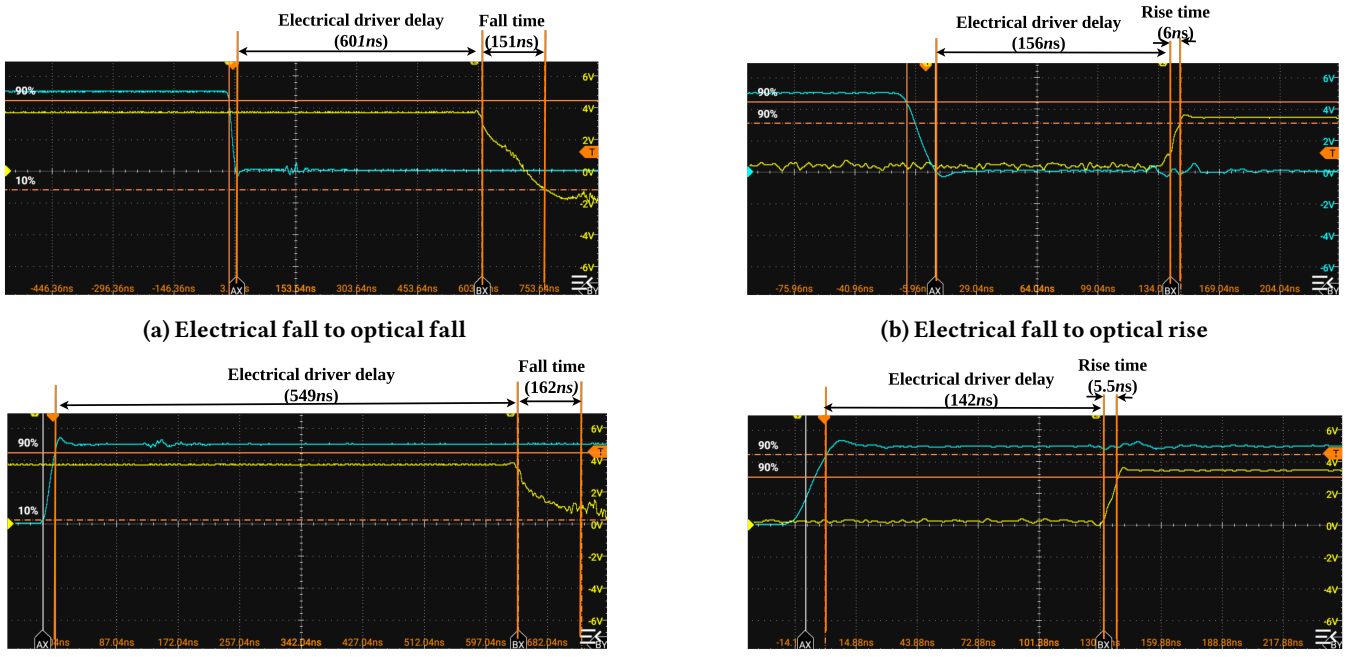
3.6 Receiver Stack: This is the Not the End!

The end-to-end reconfiguration pipeline does not end when the optical circuit is established. Once the optical transition completes, the receiver must reacquire transceiver lock, after which the NIC firmware, device driver, and host kernel react to the resulting link-state changes before communication can safely resume. These receiver-side effects dominate the overall reconfiguration delay and have been the subject of significant prior work [14, 49, 54].

To understand this overhead, we measured the receiver stack latency using NVIDIA BlueField-3 NICs. Our testbed consistently exhibited approximately 1 s before the link became operational following a circuit transition. In contrast, kernel tracing showed that the Linux networking stack spends only 120–150 μs processing the corresponding link-state events. Since the `mlx5_core` driver is open source while the NIC firmware remains proprietary, we cannot fully attribute the remaining latency. We observed similar behavior on ConnectX-6 NICs. Prior work approaches receiver locking differently. NICs used in RotorNet's [32] prototype lacked built-in low-latency loss-of-signal detection, allowing the switch reconfiguration events to be communicated to endpoints via an out-of-band notification channel. At 20 µs slot times, this workaround was sufficient as locking overhead could be absorbed. Sirius [6] eliminates the problem entirely through phase caching and storing the per-sender phase offset so that re-locking is nearly instantaneous, but this is only feasible because its fixed cyclic schedule makes the next sender's identity known in advance. Receiver recovery is intentionally outside the scope of the latency breakdown presented in Figure 1. Our objective is to expose the often-overlooked control path that precedes receiver recovery. Nevertheless, once the control path itself is reduced to the microsecond scale, the receiver stack becomes the dominant component of application-visible reconfiguration delay and warrants renewed systems attention.

4 Future Research Directions

The objective of this paper is not merely to decompose reconfiguration delay, but to expose the systems challenges that emerge when photonic switching is viewed end-to-end.



(a) Electrical fall to optical fall

(b) Electrical fall to optical rise

(c) Electrical rise to optical fall

(d) Electrical rise to optical rise

Figure 7: Measured transition delays for a 2×2 optical switching element: While the intrinsic optical switching latency is approximately 150 ns, the electrical-to-optical response time reaches up to 600 ns, making the driver and actuation path a non-negligible contributor to the overall reconfiguration delay.

Our measurements suggest that reducing application-visible reconfiguration delay requires coordinated advances across the hardware–software stack. We highlight several research directions that naturally arise from this perspective.

Hardware-efficient routing: Routing computation lies directly on the critical reconfiguration path, yet comparatively less attention has been paid to its absolute execution time on hardware than to its algorithmic complexity. Our measurements motivate routing algorithms that jointly optimize hardware execution latency, memory footprint, and resource utilization, alongside asymptotic complexity.

An alternative direction is to shrink the configuration space until precomputation becomes feasible again. Rather than supporting every possible circuit configuration, photonic interconnects could be designed around a carefully chosen subset of configurations that captures communication patterns observed in practice. Such restricted configuration spaces could dramatically simplify controller design and even enable lookup-table-based routing. Characterizing the resulting tradeoff between routing flexibility, controller complexity, and performance impact remains an interesting systems problem.

Oblivious switching [2–4, 32, 33] represents one point in this design space, replacing online routing with a small set of precomputed permutations at the cost of multi-hop forwarding and synchronization. Understanding when demand-aware and oblivious switching provide the best end-to-end performance remains an open question.

Multi-tenancy and interconnect sharing: Most prior work on circuit scheduling assumes either blocking interconnects, where every reconfiguration disrupts the entire fabric [1, 5, 8, 13, 25, 27, 38, 40, 43], or strictly non-blocking interconnects [16], where reconfigurations are independent across ports. Practical photonic interconnects, however, are often implemented as Rearrangeably Non-Blocking (RNB) networks, where reconfiguring one circuit may require modifying switches simultaneously used by other active connections. Consequently, a reconfiguration requested by one tenant can temporarily affect unrelated traffic sharing the same interconnect.

These interactions become increasingly important once reconfiguration is viewed as an end-to-end operation rather than a nanosecond-scale switching event. Circuit schedulers must reason not only about communication dependencies, but also about shared reconfiguration overheads arising from controller computation, electrical signaling, and receiver recovery. Developing scheduling algorithms that jointly optimize workload completion time while accounting for these interactions remains an important open systems problem.

Control interfaces for GPUs: Host-to-controller communication over PCIe accounts for a significant fraction of the end-to-end reconfiguration delay despite transferring only a few bytes. This raises a simple question: is a general-purpose memory transaction the right abstraction for circuit control? Circuit reconfiguration requires only reliable command delivery, ordering, and completion semantics. Identifying the minimal hardware–software protocol that provides these guarantees while minimizing latency presents an interesting systems research direction.

Receiver stack and NIC support: The receiver stack and transceiver recovery remain the dominant contributors to application-visible reconfiguration delay. Significant prior work has reduced transceiver locking to nanosecond-scale using burst-mode transceivers [14] and multicast-based control protocols reporting 3.1 ns clock and data recovery [49]. While this is only one milestone in the reconfiguration process, the application can resume communication only after the receiver stack, NIC, and operating system collectively recognize that the newly established circuit is ready [54]. This exposes a broader systems challenge: what does it mean for a circuit reconfiguration to be *complete*? The controller, optical switch, transceiver, NIC, operating system, and application each observe different completion events. Defining the appropriate completion semantics across these layers, and determining how these events should be propagated to the application, presents an interesting systems research direction.

Generalization of the measurements: The absolute latencies we report are specific to our prototype, but which stages lie on the critical path depends on how reconfiguration is controlled. *PCIe write:* In GPU-controlled reconfiguration, for example, at the boundary of a collective communication step, the circuit request must travel from the GPU to the controller. A GPU-issued PCIe write to a discrete controller (§3.1) is the most direct such path. Passing the request through the CPU only increases this cost, since the GPU must first notify the CPU before the write is issued. Conversely, dedicated control interfaces for GPUs, as discussed above, may provide a more direct, lower-latency channel between GPUs and the controller. In switch-centric reconfiguration, where the switch controller follows a fixed schedule [6, 33], no per-request message is needed, and this stage disappears. *Routing:* Routing is needed only when the configuration must be computed at runtime; fixed configurations can be precomputed, as discussed in §3.2 and under *Hardware-efficient routing* in §4. *Signaling:* Every photonic switch must be actuated electrically, so signaling is present in all designs. However, the latency depends on the switch's actuation mechanism, whether thermo-optic, electro-optic, magneto-optic, or MEMS-based (§3.4).

We believe end-to-end photonic switching has emerged as a rich systems problem. As switching technologies and circuit scheduling algorithms continue to mature, the next performance gains will come from optimizing the end-to-end reconfiguration pipeline.

Acknowledgements

We thank the anonymous reviewers and our shepherd for their valuable feedback. This work was supported by the National Science Foundation (NSF) under Grant IIS-2551372.

References

[1] Vamsi Addanki. When light bends to the collective will: A theory and vision for adaptive photonic scale-up domains. In *Proceedings of the 24th ACM Workshop on Hot Topics in Networks*, HotNets '25, page 326–334, New York, NY, USA, 2025. Association for Computing Machinery.

[2] Vamsi Addanki, Chen Avin, Goran Dario Knabe, Giannis Patronas, Dimitris Syrivelis, Nikos Terzenidis, Paraskevas Bakopoulos, Ilias Marinos, and Stefan Schmid. Vermilion: A traffic-aware reconfigurable optical interconnect with formal throughput guarantees, 2025.

[3] Vamsi Addanki, Chen Avin, and Stefan Schmid. Mars: Near-optimal throughput with shallow buffers in reconfigurable datacenter networks. *Proc. ACM Meas. Anal. Comput. Syst.*, 7(1), mar 2023.

[4] Daniel Amir, Nitika Saran, Tegan Wilson, Robert Kleinberg, Vishal Shrivastav, and Hakim Weatherspoon. Shale: A practical, scalable oblivious reconfigurable network. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 449–464, New York, NY, USA, 2024. Association for Computing Machinery.

[5] Rukshani Athapathu and George Porter. Reconfigurability within collective communication algorithms. In *Proceedings of the 2nd Workshop on Networks for AI Computing*, NAIC '25, page 43–49, New York, NY, USA, 2025. Association for Computing Machinery.

[6] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, and Hugh Williams. Sirius: A flat datacenter network with nanosecond optical switching. In *Proceedings of ACM SIGCOMM*, 2020.

[7] Peng Bao, Chunhui Yao, Richard V. Penty, and Qixiang Cheng. Silicon electro-optic mach–zehnder switch fabric with ultralow-crosstalk. *Journal of Lightwave Technology*, 43(13):6296–6305, 2025.

[8] Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, and Pramod Viswanath. Costly circuits, submodular schedules and approximate carathéodory theorems. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, SIGMETRICS '16, page 75–88, New York, NY, USA, 2016. Association for Computing Machinery.

[9] Volodymyr Borshch, Sergij V Shiyanovskii, and Oleg D Lavrentovich. Nanosecond electro-optic switching of a liquid crystal. *Physical review letters*, 111(10):107802, 2013.

[10] Qixiang Cheng, Sébastien Rumley, Meisam Bahadori, and Keren Bergman. Photonic switching in high performance datacenters [invited]. *Opt. Express*, 26(12):16022–16043, Jun 2018.

[11] Charles Clos. A study of non-blocking switching networks. *The Bell System Technical Journal*, 32(2):406–424, 1953.

[12] Eric Ding, Barry Lyu, Bhaskar Kataria, and Rachee Singh. Opus: Photonic rail-optimized fabric in ml datacenters, 2026.

[13] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, page 339–350, New York, NY, USA, 2010. Association for Computing Machinery.

[14] John Alexander Forencich. *System-level considerations for optical switching in data center networks.* University of California, San Diego, 2020.

[15] Cristiano M. Gallep and Evandro Conforti. Simulations on picosecond non-linear electro-optic switching using an ase-calibrated semiconductor optical amplifier model. *Optics Communications*, 236(1):131–139, 2004.

[16] Xin Sunny Huang, Xiaoye Steven Sun, and T.S. Eugene Ng. Sunflow: Efficient optical circuit scheduling for coflows. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, page 297–311, New York, NY, USA, 2016. Association for Computing Machinery.

[17] iPronics. ipronics unveils world's first silicon photonics optical circuit switch for ai-driven data centers optical network transformation. https://ipronics.com/ipronics-unveils-worlds-first-silicon-photonics-optical-circuit-switch-for-ai-driven-data-centers-optical-network-transformation/, March 2025. Accessed on 2025-10-09.

[18] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Clifford Young, Xiang Zhou, Zongwei Zhou, and David A Patterson. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[19] Akhilesh S. P. Khope, Mitra Saeidi, Raymond Yu, Xinru Wu, Andrew M. Netherton, Yuan Liu, Zeyu Zhang, Yujie Xia, Garey Fleeman, Alexander Spott, Sergio Pinna, Clint Schow, Roger Helkey, Luke Theogarajan, Rod C. Alferness, Adel A. M. Saleh, and John E. Bowers. Multi-wavelength selective crossbar switch. *Opt. Express*, 27(4):5203–5216, Feb 2019.

[20] Ryotaro Konoike, Keijiro Suzuki, and Kazuhiro Ikeda. Path-independent insertion-loss (piloss) 8 × 8 silicon photonics switch with <8 nsec switching time. In *2022 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, 2022.

[21] Abhishek Vijaya Kumar, Arjun Devraj, Darius Bunandar, and Rachee Singh. A case for server-scale photonic connectivity. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, HotNets '24, page 290–299, New York, NY, USA, 2024. Association for Computing Machinery.

[22] Abhishek Vijaya Kumar, Arjun Devraj, and Rachee Singh. Pccl: Photonic circuit-switched collective communication for distributed ml, 2025.

[23] Sébastien Le Beux, Hui Li, Gabriela Nicolescu, Jelena Trajkovic, and Ian O'Connor. Optical crossbars on chip, a comparative study based on worst-case losses. *Concurrency and Computation: Practice and Experience*, 26(15):2492–2503, 2014.

[24] Benjamin G. Lee and Nicolas Dupuis. Silicon photonic switch fabrics: Technology and architecture. *Journal of Lightwave Technology*, 37(1):6–20, 2019.

[25] Kevin Liang, Litao Qiao, Isaac Keslassy, and Bill Lin. Scheduling parallel optical circuit switches for ai training, 2026.

[26] Xudong Liao, Yijun Sun, Han Tian, Xinchen Wan, Yilun Jin, Zilong Wang, Zhenghang Ren, Xinyang Huang, Wenxue Li, Kin Fai Tse, Zhizhen Zhong, Guyue Liu, Ying Zhang, Xiaofeng Ye, Yiming Zhang, and Kai Chen. Mixnet: A runtime reconfigurable optical-electrical fabric for distributed mixture-of-experts training. In *Proceedings of the ACM SIGCOMM 2025 Conference*, SIGCOMM '25, page 554–574, New York, NY, USA, 2025. Association for Computing Machinery.

[27] He Liu, Matthew K. Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M. Voelker, David G. Andersen, Michael Kaminsky, George Porter, and Alex C. Snoeren. Scheduling techniques for hybrid circuit/packet networks. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '15, New York, NY, USA, 2015. Association for Computing Machinery.

[28] Jiaxing Liu, Zhan Li, Michel Inman, Zhaohui Ma, Yunong Tang, and Yu ping Huang. All-optical crossbar switching on chip. *Opt. Lett.*, 50(4):1357–1360, Feb 2025.

[29] Liangjun Lu, Xiaorui Li, Wei Gao, Xin Li, Linjie Zhou, and Jianping Chen. Silicon non-blocking 4 × 4 optical switch chip integrated with both thermal and electro-optic tuners. *IEEE Photonics Journal*, 11(6):1–9, 2019.

[30] Liangjun Lu, Shuoyi Zhao, Linjie Zhou, Dong Li, Zuxiang Li, Minjuan Wang, Xinwan Li, and Jianping Chen. 16 × 16 non-blocking silicon optical switch based on electro-optic mach-zehnder interferometers. *Opt. Express*, 24(9):9295–9307, May 2016.

[31] Yuan Ma, Xiqun Zhu, and R.W. Newcomb. Cost effective high voltage driver for large channel count optical mems switch applications. In *2005 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 5397–5400 Vol. 6, 2005.

[32] William M. Mellette, Alex Forencich, Rukshani Athapathu, Alex C. Snoeren, George Papen, and George Porter. Realizing rotornet: Toward practical microsecond scale optical networking. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 392–414, New York, NY, USA, 2024. Association for Computing Machinery.

[33] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. Rotornet: A scalable, low-complexity, optical datacenter network. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 267–280, New York, NY, USA, 2017. Association for Computing Machinery.

[34] Dimitris Nikolaidis, Panos Groumas, Christos Kouloumentas, and Hercules Avramopoulos. Novel benes network routing algorithm and hardware implementation. *Technologies*, 10(1):16, 2022.

[35] Dimitris Nikolaidis, Panos Groumas, Christos Kouloumentas, and Hercules Avramopoulos. Novel benes network routing algorithm and hardware implementation. *Technologies*, 10(1), 2022.

[36] D. C. Opferman and N. T. Tsao-wu. On a class of rearrangeable switching networks part i: Control algorithm. *The Bell System Technical Journal*, 50(5):1579–1600, 1971.

[37] Polatis. POLATIS® 7000 Optical Circuit Switch. https://www.hubersuhner.com/en/shop/product/other-systems/optical-switches/rack-mount-circuit-switches/85223159/polatis-7000-optical-circuit-switch.

[38] George Porter, Richard Strong, Nathan Farrington, Alex Forencich, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Integrating microsecond circuit switching into the data center. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, page 447–458, New York, NY, USA, 2013. Association for Computing Machinery.

[39] Lei Qiao, Weijie Tang, and Tao Chu. 32× 32 silicon electro-optic switch with built-in monitors and balanced-status units. *Scientific Reports*, 7(1):42306, 2017.

[40] Mahir Rahman, Samuel Joseph, Nihar Kodkani, Behnaz Arzani, and Vamsi Addanki. Harvest: Adaptive photonic switching schedules for collective communication in scale-up domains. In *Proceedings of the ACM SIGCOMM 2026 Conference*, ACM SIGCOMM '26, 2026.

[41] Tian-Zi Shen, Seung-Ho Hong, and Jang-Kun Song. Electro-optical switching of graphene oxide liquid crystals with an extremely large kerr coefficient. *Nature materials*, 13(4):394–399, 2014.

[42] Luis Torrijos-Morán and Daniel Pérez-López. Industry insight: photonics to scale ai data centers. *npj Nanophotonics*, 3(1):8, 2026.

[43] Víctor Valls, George Iosifidis, and Leandros Tassiulas. Birkhoff's decomposition revisited: Sparse scheduling for high-speed circuit switches. *IEEE/ACM Transactions on Networking*, 29(6):2399–2412, 2021.

[44] Abhishek Vijaya Kumar, Eric Ding, Arjun Devraj, Darius Bunandar, and Rachee Singh. Reconfigurable torus fabrics for multi-tenant ml. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '26, page 1547–1565, New York, NY, USA, 2026. Association for Computing Machinery.

[45] Mark Wade, Erik Anderson, Shahab Ardalan, Pavan Bhargava, Sidney Buchbinder, Michael L. Davenport, John Fini, Haiwei Lu, Chen Li, Roy Meade, Chandru Ramamurthy, Michael Rust, Forrest Sedgwick, Vladimir Stojanovic, Derek Van Orden, Chong Zhang, Chen Sun, Sergey Y. Shumarayev, Conor O'Keeffe, Tim T. Hoang, David Kehlet, Ravi V. Mahajan, Matthew T. Guzy, Allen Chan, and Tina Tran. Teraphy: A chiplet technology for low-power, high-bandwidth in-package optical i/o. *IEEE Micro*, 40(2):63–71, 2020.

[46] Abraham Waksman. A permutation network. *J. ACM*, 15(1):159–163, January 1968.

[47] Mi Wang, Liangjia Zong, Lei Mao, Andres Marquez, Yabin Ye, Han Zhao, and Francisco Javier Vaquero Caballero. Lcos slm study and its application in wavelength selective switch. *Photonics*, 4(2), 2017.

[48] Zhenguo Wu, Benjamin Klenk, Larry Dennison, and Keren Bergman. Actina: Adapting circuit-switching techniques for ai networking architectures. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '25, page 1211–1222, New York, NY, USA, 2025. Association for Computing Machinery.

[49] Xuwei Xue and Nicola Calabretta. Nanosecond optical switching and control system for data center networks. *Nature communications*, 13(1):2257, 2022.

[50] Xuwei Xue, Shaojuan Zhang, Bingli Guo, Wei Ji, Rui Yin, Bin Chen, and Shanguo Huang. Optical switching data center networks: Understanding techniques and challenges, 2023.

[51] M. Yano, F. Yamagishi, and T. Tsuda. Optical mems for photonic switching-compact and stable optical crossconnect switches for simple, fast, and flexible wavelength applications in recent photonic networks. *IEEE Journal of Selected Topics in Quantum Electronics*, 11(2):383–394, 2005.

[52] Yuexin Yin, Xinru Xu, Xinyu Lv, Bo Tang, Peng Zhang, and Daming Zhang. 3 × 3 non-blocking soi electro-optic switch. *IEEE Photonics Technology Letters*, 35(24):1295–1298, 2023.

[53] Rami Zecharia and Yuval Shavitt. A parallel algorithm and scalable architecture for routing in beneš networks. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, pages 921–930, 2024.

[54] Johannes Zerwas, Wolfgang Kellerer, and Andreas Blenk. What you need to know about optical circuit reconfigurations in datacenter networks. In *2021 33th International Teletraffic Congress (ITC-33)*, pages 1–9, 2021.

[55] Hanqing Zhu, Shupeng Ning, Hongjian Zhou, Ziang Yin, Ray T. Chen, Jiaqi Gu, and David Z. Pan. Harnessing photonics for machine intelligence, 2026.