Petabit Router-in-a-Package: Rethinking Internet Routers in the Age of In-Packaged Optics and Heterogeneous Integration

Isaac Keslassy
Technion, UC Berkeley

Bill Lin
UC San Diego

Abstract

The goal of this paper is to apply two groundbreaking scaling transformations from the computing packaging industry to internet routers: heterogeneous integration of High-Bandwidth Memories (HBMs) and chiplets, as well as in-package optics. We propose a novel internet router architecture that leverages these technologies to realize a petabit/sec router in a single integrated package. We first introduce a new Split-Parallel Switch (SPS) architecture that spatially splits (without processing) the incoming fibers and distributes them across smaller independent switches. Then, we design these smaller switches as novel shared-memory HBM switches. We show how a Parallel Frame Interleaving (PFI) algorithm packs traffic into frames and accesses the HBM banks in a cyclical staggered interleaved way to reach HBM peak data rates. We further explain why these new technologies represent a paradigm shift in the design of future internet routers. Finally, we highlight that power consumption may constitute the main scaling bottleneck.

CCS Concepts

• **Networks** → **Routers**.

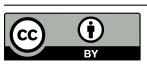
Keywords

Router, HBM, chiplet, in-packaged optics

ACM Reference Format:
Isaac Keslassy and Bill Lin. 2025. Petabit Router-in-a-Package: Rethinking Internet Routers in the Age of In-Packaged Optics and Heterogeneous Integration. In *The 24th ACM Workshop on Hot Topics in Networks (HotNets '25), November 17–18, 2025, College Park, MD, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3772356.3772398

1 Introduction

The goal of this paper is to apply two major scaling transformations from the computing packaging industry to internet routers, by introducing a novel router architecture that can harness these emerging technologies.



HotNets '25, College Park, MD, USA

ACM ISBN 979-8-4007-2280-6/2025/11
https://doi.org/10.1145/3772356.3772398

The computing industry has recently needed to deal with the slowing pace of doubling transistor density below Moore's law, and the increasing power density with the breakdown of Dennard scaling. It has also faced sharply increased demands due to the explosive rise in computing workloads from generative AI applications. To address these challenges, it is undergoing two radical changes.

First, it relies on *heterogeneous integration* within packages, which enables the co-packaging of potentially many 3D stacks of *High-Bandwidth Memories (HBMs)* [58] and many processing dies called *chiplets* in the same heterogeneously integrated package. HBM is a critical enabling technology that provides enormous memory capacity and bandwidth in a single 3D stack with a small footprint. Current HBM stacks only have a footprint of 11 mm × 11 mm [21], significantly less than typical package sizes of 200 mm × 200 mm, and even less than the 500 mm × 500 mm that have been demonstrated with panel-scale glass substrates [28]. HBM4 technology [34, 39] provides up to 64 GB of memory capacity and 20.48 Tb/s of memory bandwidth per stack. The integration of many HBM stacks on a panel-scale substrate could provide *many terabytes of memory capacity* and *petabits/second of memory bandwidth* in a single package. Similarly, panel-scale substrates also enable the incorporation of many *chiplets* to handle the packet processing.

Another critical emerging technology to continued scaling is *in-package optics and photonics interposers* [22, 42, 43]. With the enormous increase in compute and memory capacities made possible by panel-scale integration of HBM stacks and processing chiplets, the industry has turned to *optical I/O* to get huge amounts of data in and out of a package as well as optical communications within the package [22, 42, 43]. Today, photonics packaging with 16 fiber-ribbon arrays, 16 fibers per array, and 8 wavelengths per fiber can already deliver 114 Tb/s of optical I/O [22, 42, 43]. As this technology continues to advance, fiber ribbon arrays of up to 64 fibers [22] and 32 wavelengths per fiber [12, 56] are expected, delivering another order of magnitude of data rates to achieve *petabit-scale optical I/Os*.

These emerging technologies in combination have prompted us to *rethink internet router architectures in the age of in-package optics and heterogeneous integration*. We propose a novel internet router architecture that leverages them to realize a *petabit/sec router in a single integrated package*. Our design is built upon two key architectural contributions:

(1) First, we propose a *Split-Parallel Switch (SPS)* architecture that *spatially splits* (without processing) the incoming fibers across H smaller independent switches. This differs from load-balanced router architectures and other demand-oblivious designs [38, 47, 48] that require electronic packet-by-packet load-balancing to the smaller switches, from parallel packet switch architectures [31] that must coordinate among the smaller switches, and from mesh-based architectures that waste significant I/O capacity and power on interconnecting the smaller switches [10, 61]. SPS exploits the fact that data arrives to our router-in-a-package as optical signals. It enables the Optical-to-Electrical (O/E) and Electrical-to-Optical (E/O) conversions to occur only once, within the smaller switches.

(2) Our second major contribution is the design of the smaller switches as novel shared-memory *HBM switches*. With a small speedup, they can mimic an ideal output-queued shared-memory architecture, the holy grail of router architectures that can handle arbitrary admissible traffic at 100% throughput with work conservation, and for which there is no known algorithm that works at these speeds. These HBM switches differ from prior shared-memory switch architectures [7, 30, 54, 55] in that they take advantage of the capabilities of modern HBMs to provide impressive switching and buffering capabilities. To leverage them, the new *Parallel Frame Interleaving (PFI)* algorithm (i) packs traffic into frames large enough to leverage the enormous bandwidth made possible by HBMs via their ultra-wide interfaces and parallel memory banks; (ii) accesses memory in a cyclical staggered bank interleaved way that can reach peak data rates; and (iii) relies on cyclical crossbars to avoid any scheduling.

Combining these contributions leads to a potential increase in the router capacity per area by 1-2 orders of magnitude vs. current router implementations. We further explain why these new technologies and our suggested architecture form a paradigm shift in how we should conceive and design internet routers. In particular, they impact router buffer sizing and management. Going forward, we argue that *the main scaling bottleneck may be power consumption*, and encourage the community to examine ways to reduce it.

2 Top-Level Split-Parallel Switch

2.1 Design process

We start by describing the thought process that led to our *Split-Parallel Switch (SPS)* design. We mention the leading alternatives in the literature and explain why they do not fit our goals of providing a high throughput with a reasonable power consumption.

Design 1. The simplest design for an $N \times N$ router consists of *a single centralized switch fabric*.

Challenge 1. A single centralized switch cannot keep up with our needed high rates, as it would need prohibitive switching rates as well as memory access rates.

Idea 1. In an $N \times N$ router, we want to introduce H smaller switches to decompose the switching work of our large router into H smaller conceptual routers.

Design 2. A standard approach with a large number H of smaller switches is to organize them in a $\sqrt{H} \times \sqrt{H}$ two-dimensional *mesh* configuration, i.e., each smaller switch is connected to its adjacent neighbors in the same row or column. Incoming packets enter some smaller switch, and pass through a number of intermediate smaller switches before exiting.

Challenge 2. Passing through the many intermediate hops wastes a significant portion of the link and switch capacity, and power. For instance, in a 10×10 mesh, the guaranteed capacity is at most 20% of the total capacity for an arbitrary admissible traffic pattern, wasting 80% of the capacity and power [61]. A recent work [10] proposes to add instead extra I/Os to the switch chiplets to implement the pass-throughs. Still, the number of required extra I/Os would be substantial.

Idea 2. We want to make sure that each packet goes through a number of hops that does not grow with H.

Design 3. We can adopt a *three-stage* network configuration, such as a *Clos* network with three stages of $H/3$. Using this design, we could schedule packets using a load-balanced router approach or another demand-oblivious schedule [38, 47, 48], or a parallel packet switch architecture [31].

Challenge 3. These approaches assume that each input can adopt a packet-by-packet load balancing, and each output can implement a reordering buffer. These are near-impossible to implement in optics, so in practice, all three stages will need electronics. In addition, within the package, we cannot organize all the smaller switches next to each other, so we will also need additional optical waveguide-based links between consecutive logical switches. Thus, a three-stage architecture would need three electronic stages separated by optics, i.e., three OEO conversion stages that would increase the power consumption, both because of the conversions themselves and because the three stages split the processing and memory functions into more chiplets.

Idea 3. Organize the H switches such that a single OEO conversion is needed, i.e., make sure that all the electronic processing for a packet is confined within a single switch.

Design 4. We propose to organize our H smaller switches as H *parallel switches*, such that any incoming packet only accesses *a single* smaller switch and goes through a single OEO conversion. We cannot use electronics to load-balance packets among the H stages, because it would add an OEO conversion. Thus, we rely on *a poor man's solution* of evenly splitting the incoming fibers at each input. We assign $1/H$-th of the fibers to each switch. That is, given F fibers per input, we split the fibers in a straightforward manner by connecting the first F/H fibers that enter each input to the first switch, etc. This solution is non-optimal and does not guarantee a perfect load-balancing as in electronic per-packet load-balancing. In fact, the uneven distribution across smaller switches operating at a reduced capacity may potentially lead to packet losses. This is the cost of having a single OEO stage.

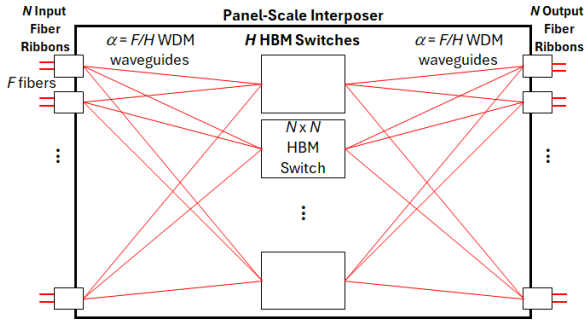


Figure 1: The SPS architecture.

Challenge 4. Our architecture satisfies our power goals, but the straightforward fiber splitting has a couple of issues: (1) Practically, the first fiber of each input is typically connected first, and therefore has a higher load. Thus, the above splitting pattern typically causes a higher load in the first switch, yielding a poorer load-balancing. (2) In addition, an adversarial attacker could exploit the known internal splitting pattern of the fibers even without knowing the specific internal aspects of the router.

Idea 4. We use a pseudo-random pattern to pick the F/H fibers that connect each input to each middle switch.

2.2 Architecture

In this section, we describe the SPS architecture. We parameterize the dimensions so that it can be scaled to different capacities, and illustrate them with a set of implementation parameters that achieve a petabit total I/O in a single package.

Modules. Fig. 1 illustrates the SPS architecture that fits on a single package. The package includes $N = 16$ fiber-ribbon arrays. Each such array comprises $F = 64$ fibers. Thus, there are $N \cdot F = 1,024$ fibers per package.

Each fiber carries $W = 16$ *input* wavelength-division-multiplexing (WDM) channels, of bandwidth $R = 40$ Gb/s each. Likewise, for better packaging, each of the *same* $N = 16$ fiber ribbons also serves as the *egress* of the package, with each fiber also carrying a separate set of $W = 16$ wavelengths as *output* WDM channels. Overall, a package offers $N \cdot F \cdot W \cdot R = 655$ Tb/s of I/O in each direction, i.e., *a total I/O of* 1.31 Pb/s.

Within the package, there are $H = 16$ parallel $N \times N$ HBM switches. Each such switch must support $1/H = 1/16^{\text{th}}$ of the total I/O, i.e., $2(N \cdot F \cdot W \cdot R)/H = 81.92$ Tb/s of memory I/O. The HBM switches include several HBM stacks (§ 3.2).

At each fiber ribbon, wavelengths coming through the F optical fibers are passively coupled to the corresponding wavelengths in the F internal WDM waveguides via optical couplers for continuing the optical signal propagation within the package. From each ribbon, a pseudo-random set of $\alpha = F/H = 4$ WDM waveguides is connected to each of the $H = 16$ smaller HBM switches. Likewise, at the egress, $\alpha = F/H = 4$ WDM waveguides are received from each HBM switch at each output fiber ribbon.

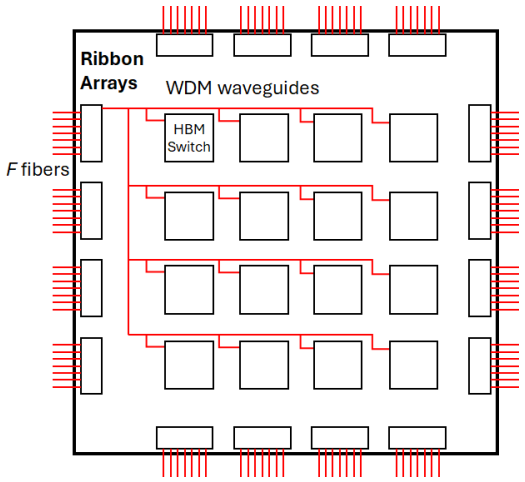


Figure 2: Packaging view of SPS on a 2.5D photonics interposer.

Operation. All traffic incoming at some fiber of some input fiber ribbon is propagated through its corresponding waveguide, and arrives at one of the HBM switches. There, as detailed in § 3.2, the HBM switches convert the optical signal into an electronic signal, extract the packets, and switch them to the HBM switch egress corresponding to their output fiber ribbon. They are then reconverted into an optical signal, sent in one of the waveguide channels for this output ribbon, and forwarded through the corresponding egress fiber.

Packaging. Fig. 2 shows a possible layout within a 2.5D photonics interposer. The $N = 16$ fiber ribbons could be organized as 4 arrays on each side, while the $H = 16$ HBM switches could be arranged as a 4×4 matrix in the middle. For clarity, only the WDM waveguides connected to the first fiber ribbon are outlined.

Modularity. The SPS architecture enables a modular approach, from a single dense 1.31 Pb/s I/O package with 16 HBM switches, to 16 parallel packages of $1/16^{\text{th}}$ the capacity.

3 HBM Switch

3.1 Design process

We now describe the design of the smaller HBM switches.

Challenge 5. Although the SPS architecture introduced in § 2 splits incoming traffic across H smaller switches operating in parallel, each $N \times N$ smaller switch must still handle a significant load of $1/H = 1/16^{\text{th}}$ of the total traffic. This translates into a total incoming and outgoing traffic of $2(N \cdot F \cdot W \cdot R)/H = 81.92$ Tb/s.

Idea 5. To support this large combined data rate, we propose to use modern HBMs.

Design 5. Our reference design, called the *HBM switch*, uses the HBM4 technology [34, 39] that provides a 2,048-bit ultra-wide interface organized as 32 channels of 64 bits each. Recently announced HBM4 commercial offerings [3, 19, 27] have exceeded the HBM4 standard by achieving over 10 Gb/s

per bit for a total bandwidth of $2,048 \cdot 10$ Gb/s = 20.48 Tb/s. By grouping $B = 4$ HBM stacks, we obtain $4 \cdot 20.48$ Tb/s = 81.92 Tb/s of total bandwidth through $T = 4 \cdot 32 = 128$ parallel channels.

Challenge 6. To use HBMs at peak data rate, memory accesses need to satisfy complex timing rules for accessing banks, rows, columns, etc. It is hard to reconcile (1) the arbitrary and non-uniform traffic patterns that the HBM switch needs to service with (2) these complex HBM access rules, while (3) providing strong and predictable performance guarantees. There are two main approaches in the literature. One is to provide a very strong theoretical guarantee and fully implement an ideal output-queued (OQ) shared-memory switch [7, 30, 54, 55]. However, given current large HBMs, tracking packet locations ("bookkeeping") would require prohibitive SRAM sizes of several GBs, and simulating a shadow ideal switch would incur prohibitive compute requirements. Another approach is to randomly spread packets across memory modules [59], then use a large reordering buffer at the output [57, 62, 66]. This is akin to packet spraying in datacenters [14, 26, 45, 68].

Unfortunately, both approaches are oblivious to the specific HBM memory rules and assume *worst-case* random access times, leading to a large throughput loss. Specifically, this worst-case random access needs about 30 ns [34] just to activate and close (precharge) banks. Giving these approaches the benefit of the doubt and assuming that they can leverage the HBM parallel channels, which did not exist 20 years ago, they would still suffer from throughput reduction factors ranging from 2.6× for 1,500-byte packets to 39× for worst-case 64-byte ones. If they don't leverage parallel channels, the reduction can reach 1,250×.

Idea 6. We aim to aggressively design our switch to operate at the *best-case* memory access times rather than the worst-case. To achieve peak data rates, we need to write at once a large quantity of data to harness the many parallel channels available via the HBM's ultra-wide interface. We also want this data to have the same output so it can be read and depart together. To do so, we propose to aggregate incoming packets with a shared output destination into *frames* that can be efficiently striped across all HBM channels in parallel. These frames are similar to a shuttle that waits for passengers to a given destination and departs as soon as it is packed. Frames can contain packets from different inputs as long as they share the same output destination, offering flexible aggregation. These frames are inspired by past works on memory management in single linecards [29] and the uniform frame spreading (UFS) algorithm in load-balanced routers [37].

Design 6. We devise a *Parallel Frame Interleaving (PFI)* algorithm. We outline it next before detailing it in § 3.2.

(1) Frame aggregation. PFI aggregates traffic in two stages. First, at each input, variable-size packets arrive at per-output queues, where they are cut and assembled into fixed-size batches of $k = 4$ KB. Then, at an intermediate buffering stage, batches enter per-output queues, where they are assembled into $K = 512$ KB frames of $K/k = 128$ batches.

(2) Slicing. To prepare for the highly parallel HBM memory accesses, PFI implements the intermediate buffering stage by slicing each batch into N equal chunks that are placed in N parallel intermediate memory modules. Inspired by prior load-balanced switch architectures [37, 38, 44, 67] that do not require switch-fabric scheduling, PFI uses an $N \times N$ cyclical crossbar between the N inputs and the N memory modules. The crossbar staggers the forwarding of the N batch slices from each input to the N intermediate memory modules. Thus, each frame of 128 batches is also sliced into N parts.

(3) Bank interleaving. PFI uses *predictable* and *staggered* bank accesses to achieve peak data rates, which is challenging because frame arrivals are *unpredictable*. To do so, we introduce the idea of partitioning the set of banks into disjoint *bank interleaving groups*, defined to be groups of γ *consecutive* banks ℓ to $(\ell + \gamma - 1)$. For example, with $\gamma = 4$, $L = 64$ banks are partitioned into $L/\gamma = 16$ consecutive groups of $\gamma = 4$ banks. The first group comprises banks 1 to 4, the second banks 5 to 8, etc. Then, given T parallel HBM channels and γ banks, PFI first writes $1/\gamma^{\text{th}}$ of the frame into bank ℓ across all T channels, then $1/\gamma^{\text{th}}$ into bank $\ell + 1$, and so on into all banks in the group in a perfectly *staggered* bank interleaving manner. γ is defined in a way that makes all these bank accesses independent of each other without breaking HBM memory access rules.

(4) No bookkeeping. To avoid complex bookkeeping and ensure frame ordering, PFI deterministically writes consecutive frames for the same output to consecutive bank groups, regardless of frame arrival patterns, i.e., the n-th frame for a given output is written into bank interleaving group $h = (n \mod (L/\gamma))$. On the output side, frames are read in the same sequence, thus keeping the frame order.

(5) Cyclical output reads. PFI reads the HBM cyclically for each of the N outputs. It then proceeds at the output side symmetrically to the input side, reading frames from the HBM as slices across N memory modules, cutting them into sliced batches, sending all batch slices to their output using an $N \times N$ cyclical crossbar, and finally unpacking batches into variable-size packets.

(6) Performance guarantees. We design PFI to guarantee *100% throughput*. In addition, with a small speedup, an HBM switch with PFI can *mimic* an ideal OQ shared-memory switch, i.e., given the same input sequence to the HBM switch and to an ideal switch, any packet departs the HBM switch within a finite delay after its departure from the ideal one [6].

3.2 Architecture and PFI Algorithm

In this section, we detail the internal architecture of the HBM switch (Fig. 3) and outline how PFI handles traffic (Fig. 4).

① **Input port SRAMs.** At each input port, incoming optical data from the waveguides is first converted into electronic signals via O/E conversion. Then, a processing chiplet

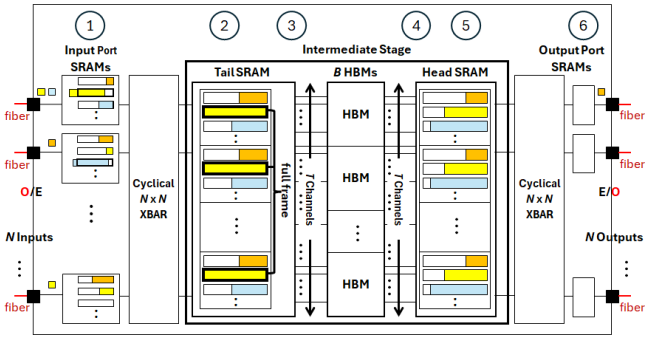


Figure 3: The architecture of each HBM switch.

determines the HBM switch output for incoming variable-length packets, queues them in SRAM by output destination using $N = 16$ per-output queues, then packs them into batches of $k = 4$ KB. Packets may straddle two batches. Once formed, batches for all outputs are (logically) stored in a later FIFO queue. The head-of-line batch is sliced into slices of $k/N = 256$ bytes. Then a cyclical crossbar sends each slice to a different SRAM module in the tail SRAM, always starting from the first SRAM module.

Example. In the first input port of Fig. 3, yellow packets for output 2 are queued in the second queue. They have already formed a batch that is being gradually sent. Additional packets are arriving and starting to form a second batch.

Batch size. Each input port SRAM must support the writing and reading of packets at the data rate of $P = \alpha \cdot W \cdot R = 4 \cdot 16 \cdot 40\,\text{Gb/s} = 2.56\,\text{Tb/s}$, for a total of $2P = 5.12\,\text{Tb/s}$. Assuming a 2.5 GHz clock, each input port SRAM can deliver 2.5 Gb/s per bit of SRAM interface. Thus, the input port SRAM should provide a $5120/2.5 = 2,048$-bit wide interface. The batch size is exactly N times this interface width, $k = 16 \cdot 2,048$ bits $= 4$ KB so that the batch slices can be uniformly spread across the N SRAM modules at the tail SRAM.

② **Tail SRAM.** The tail SRAM stores the batches, aggregates them into frames, then supports the concurrent writing of the T segments in a frame over the $T = 128$ channels available across the ultra-wide interface of the $B = 4$ HBMs.

(i) Cyclical crossbar. An $N \times N$ cyclical crossbar on the input side connects the N input ports to the N SRAM modules in the tail SRAM, using a cyclic rotation (no scheduling). The crossbar facilitates the input ports in striping frames across the N tail SRAM modules. Each crossbar port operates at an input data rate of $P = 2.56$ Tb/s via a $2,048$-bit interface. As the crossbar implements a cyclic rotation, it can just be implemented by simple 1-dimensional multiplexors at the crossbar outputs with cyclic selections. Alternatively, the crossbar could be replaced with an $N \times N$ *mesh* using spatial-division-multiplexing. i.e., the $2,048$ bits available at each input can be split into N sets of $2,048/N = 128$ wires, each set going to one of N outputs for parallel transfers rather than cyclic connections, which may simplify the overall logic.

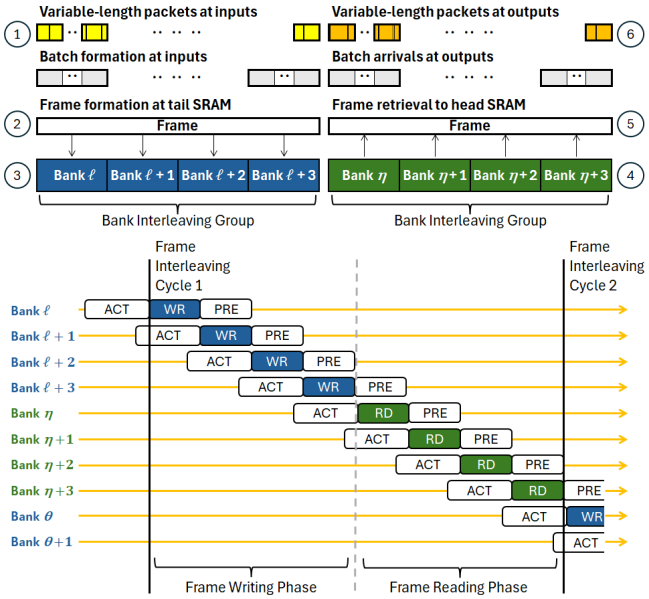


Figure 4: Intuition for the HBM switch operation.

(ii) SRAM modules. The tail SRAM is organized into N separate SRAM modules. Each module is divided into N logical areas corresponding to N per-output queues. Each batch reaches all modules in a staggered way, such that each batch slice is stored in a per-output queue in its module. When the queue size of a module reaches $K/k = 128$ batch slices, it forms a new frame slice, all modules doing so for the same frame in a staggered way. Frame slices for all outputs are then stored in a shared logical FIFO queue.

(iii) Memory width. Each tail SRAM module is $2,048$-bit wide, operating at 2.5 GHz, as for each input port SRAM module. Each HBM channel is 64-bit wide, for a total of 512 bits per group of $T/N = 128/16 = 8$ channels. As HBM4 operates at 10 Gb/s per bit, i.e., 4× the data rate per bit in comparison with the 2.5 GHz SRAM clock, the $2,048$-bit SRAM interface can be readily serialized 4-to-1 to the 512-bit HBM interface via the HBM controller. The group of 4 HBMs provides $T = 4 \cdot 32 = 128$ channels across an ultra-wide interface, so $4 \cdot 2,048 = 8,192$ bits $= 1,024$ bytes could be written into or read from the HBM group in parallel.

③ **Writing into the HBMs.** The blue elements in Fig. 4 illustrate how PFI writes a frame into the HBMs. To exploit the ultra-wide HBM interface with T channels, PFI first writes a segment (blue WR) of size S at bank ℓ into each of the T channels in parallel. Using staggered bank interleaving, it does so $\gamma = 4$ times for each frame, into consecutive banks $\ell, \ell+1, \ell+2, \ell+3$. Thus, the frame size is $K = \gamma \cdot T \cdot S$ bytes. For each bank write, PFI in parallel closes the previous bank (precharge *PRE*) and opens the next one (activation *ACT*).

Our reference design uses $S = 1$ KB, which is the smallest integer multiple of the HBM4 burst-length that satisfies the HBM four-activation window constraint [34] with our bank interleaving schedule, i.e., such that at most four banks are activated at once, while also being a unit fraction of a row length.

We also choose γ so that (i) the precharge (closing) of the first bank in one group completes before the activation of the first bank in the next, to ensure that we can continue the perfect staggered bank interleaving seamlessly from one group to the next, i.e., each group is independent from the previous one; and (ii) γ should also satisfy a four-activation window limit that allows at most four concurrent bank activations, to prevent the memory from drawing too much instantaneous current. $\gamma = 4$ satisfies both conditions. As a result, the frame size is $K = \gamma \cdot T \cdot S = 4 \cdot 128 \cdot 1$ KB = 512 KB.

HBM memory organization. The HBM memory is divided into per-output regions. This region allocation could be static, or dynamic with large per-output pages. Each region is then subdivided hierarchically: (1) first into rows; (2) then into segment-size sub-rows; and (3) finally into banks. We write and read data in this exact FIFO order. For example, in Fig. 4, after writing a blue segment into some bank ℓ, we write the next one in the next bank $\ell + 1$, across all T channels. With a static region allocation, the head, tail, and number of entries of the FIFO can simply be tracked with counters. With dynamic allocation using large per-output pages, a small extra amount of SRAM would suffice to track pointers to these large pages.

④ **Reading from the HBMs.** PFI cyclically reads a frame for each output. Fig. 4 shows how it uses staggered bank interleaving, as for writes. It accesses the first bank of the next output, reads a segment (green RD) across the T channels, then does the same for all $\gamma = 4$ consecutive banks, collecting a full frame for this output.

⑤ **Head SRAM.** Like the tail SRAM, the head SRAM is organized into N SRAM modules. Each head SRAM module obtains a full frame slice and cuts it into batch slices that are first stored in a per-output queue, then sent to their output.

⑥ **Output ports.** At each output port, the received batches are cut back into variable-length packets, which are then converted back into optical signals via E/O conversion, before going out a waveguide. As the waveguides will be going out on an output fiber ribbon, which will go to another router, we have the freedom to carry the signals on any of the $\alpha = 4$ fibers and any of the $W = 16$ output WDM channels. As in ECMP or dynamic link aggregation (LAG) [5, 15], we hash the packets across the available waveguides and wavelengths using their flow 5-tuples.

4 Design Analysis

We now analyze the design tradeoffs using various metrics.

Router buffer sizing. The number of HBM memories should be sized to provide enough (i) *bandwidth* and (ii) *buffering*. We found above that to provide the necessary memory bandwidth, we need to group $B = 4$ HBM4 stacks together per HBM switch. Using 64 GB per HBM4 stack [65], this translates into a large total amount of $H{\cdot}B{\cdot}64 = 16{\cdot}4{\cdot}64 = 4.096$ TB of buffering. To put it into proportion, it means that the router can store up to $(H \cdot B \cdot 64) \cdot 8/(N \cdot F \cdot W \cdot R) = 4.096 \cdot 8/655.36 \approx 51.2$ ms of buffering. This is in line with the old Van Jacobson rule of thumb that a buffer should hold one bandwidth-delay product of packets [32], and much more than the Stanford model of buffer sizing [4, 46]. It is also much more than a "core router buffering in the range of 5-10 msec," as recommended in a Cisco white paper [64]. Cisco's 400-Gbps linecards offer up to 18 ms (Q100 linecard) or 13 ms of buffering (Q200 linecard), e.g., 5 ms for Cisco's 8201-32FH [13, 63].

SRAM sizing. We find that the total needed SRAM size is 14.5 MB, which can be easily implemented today in SRAM. It is a small cost we pay for assembling the large frames that are needed to fully exploit the large HBM parallelism deterministically while accessing memory at peak data rates. Note that if we were to adopt an alternative statistical approach, akin to packet spraying in datacenters, we would not need to pay this cost of forming frames, but would need to pay an alternative memory cost for the packet reordering buffer, which seems to be an order of magnitude higher depending on the acceptable reordering rate [57, 62, 66].

Power estimate. Each of our $H = 16$ HBM switches handles 41 Tb/s of incoming traffic. This is less than the Broadcom Tomahawk 5 BCM878900 switch chip, which can handle 51.2 Tb/s incoming traffic [8] with a power dissipation of 500 W [9]. The Broadcom chip can handle all the packet processing and SRAM buffering. Thus, as a first approximation, the packet processing and SRAM buffering for each HBM switch should consume at most $500 \cdot (41/51.2) = 400$ W. In addition, each HBM4 stack should consume about 75 W [52], so $B = 4$ stacks consume about 300 W. Finally, commercially available silicon photonics can perform OEO conversion at about 1.15 pJ/bit [16–18, 20, 25, 49]. At 81.92 Tb/s of I/O per HBM switch, the power required for OEO conversion for each HBM switch is about 94 W. Overall, each HBM switch should consume about $400 + 300 + 94 = 794$ W, yielding about 12.7 kW for all $H = 16$ HBM switches. This can be compared to the Cerebras WSE-3 wafer-scale processor [36], which is commercially deployed and consumes 23 kW of power. It uses a state-of-the-art liquid and air cooling system [36, 41, 51] to dissipate the heat. The same system could be used to cool our internet router, which consumes just above half the power.

Area estimate. The Broadcom die size is estimated at 800 mm^2 [8]. Assuming conservatively 800 mm^2 for the processing chiplet and $4 \cdot (11 \text{ mm} \cdot 11 \text{ mm}) = 484$ mm^2 for the HBM stacks, we get about $1,284$ mm^2 per HBM switch. For the 16 HBM switches, we obtain $16 \cdot 1284$ mm^2 = $20,544$ mm^2, which is under 10% of the $500 \text{ mm} \cdot 500 \text{ mm} = 250,000$ mm^2 of the surface area of a panel-scale substrate. Thus, the area should not be a bottleneck at this stage.

Traffic matrix at HBM switches. SPS spatially splits the incoming fibers into the HBM switches. This may lead to uneven traffic matrices (TMs) between HBM switches, and possibly traffic loss. In practice, most incoming links would use either ECMP or LAG [5, 15], thus traffic would typically be load-balanced across fibers using hashing [35, 53], leading to even TMs.

Latency and bypass. Waiting to fill frames for scheduling can increase latency. Instead, when there are no full frames, we can use frame padding [33, 37] to decrease latency. A *bypass* mechanism can further reduce latency: If the tail SRAM sees that there is nothing stored in the HBM for output k, then when it is time for the head SRAM to cyclically get a frame for output k, the tail SRAM can bypass the HBM and send the head-of-line (potentially padded) frame directly to the head SRAM.

Frame interleaving cycle. The transitions between the frame write/read phases total about 2% of the cycle duration [34]. Since reads and writes are mandatory in a switch, we consider these transition delays as part of our baseline 100% throughput. Also, HBM4 [34] provides single-bank refresh operations that can be hidden without affecting the cycle time.

5 Impact on networking future

Capacity increase. We have shown a potential increase in router capacity per given area by 1-2 orders of magnitude vs. current routers, given the emerging in-package photonics, heterogeneous integration, and HBM4 technologies; e.g., a Cisco 8201-32FH of 1RU (rack unit) height and HBM can accept 32 lines of 400 Gb/s for an aggregate 12.8 Tb/s, over 50× less than the input bandwidth of our router, while occupying about the same space.

Router evolution. Although our reference design is based on the latest HBM4 memories [3, 19, 27], industry roadmaps show that future HBMs are expected to increase by a factor of 4× in memory capacity and bandwidth [52]. Emerging memory technologies using monolithic 3D stackable DRAM (multi-DRAM layers per die) with direct stacking of memories and processing chiplets in the same 3D stack [23, 24] are expected to increase the memory capacity and memory bandwidth of an HBM stack by a factor of 10× compared to HBM4. These expected improvements will enable us to realize our reference design with far fewer HBM stacks, translating into smaller footprints and power. They can also enable us to realize yet higher capacity routers in a single package.

Buffer sizing. Previous research assumed off-chip memory (DRAM) bandwidth is a key limiting factor in router performance, which prompted much research into minimizing router buffer sizes so that faster routers could be designed with only on-chip memory without the need to go off-chip [46]. However, with the integration of many HBM stacks and packet processing chiplets within the same package, neither memory capacity nor memory bandwidth appears to be a barrier to faster routers, even when scaling to handle massive incoming traffic made possible by in-package photonics I/Os. This *memory glut* means that router buffer sizing research should be revisited. This does not mean that smaller buffers should be automatically discarded. Smaller buffers may still provide better QoS [60] and reduce power consumption, but their impact on router capacity is smaller than previously assumed.

Buffer management. The assumption that "buffer size is not keeping up with the increase in switch capacity" [1] may no longer hold. Thus, the memory glut may also impact buffer management and buffer-sharing algorithms [1, 2], reducing the need for complex algorithms to address memory scarcity and ensure fair allocation.

Designing datacenter switches. Could our switch-fabric designs fit datacenter switches? Datacenter switches rely on the same switch-fabric principles, so it looks like a natural extension. In fact, since they use less buffering than internet routers, they may rely on the SPS architecture with even less power consumption, since they would use HBM switches with much less HBM capacity. However, (i) *latency* is more critical in datacenter networks. Thus, the HBM switch may need to be modified to rely on smaller frames. In addition, (2) *switch radix* may be a more significant concern in datacenters, with a finer output granularity than in core routers. Alternatively, a solution to both problems could be to implement *an SPS architecture that relies on smaller commercial switches as chiplets*. It is interesting to note that Broadcom has started shipping the Tomahawk 6 switch chip [11] and the Jericho 4 switch/router chip [50], both relying on a chiplet architecture.

Wasted internal traffic. The ability to scale routers by 1-2 orders of magnitude can save a significant fraction of the current WAN capacity that is devoted to internal traffic needed to interconnect smaller routers.

Conclusion: the road ahead. What is the bottleneck to further scaling our router designs? § 4 shows that the footprint of our reference design is only a small fraction of the surface area of a panel-scale substrate, so *it is not area-bound*. Also, *it is not yet I/O-bound* since we have assumed 40 Gb/s data rates per wavelength, whereas 112 Gb/s data rates per wavelength are already possible with PAM4 modulation [42], and current panel substrates with 500 mm panel edges can support the attachment of many more fiber ribbons. In fact, the main limiting factor in our router-in-a-package design is *power and heat dissipation*. As noted in § 4, the estimated power of our reference design is just above half that of a commercial wafer-scale processor for which existing state-of-the-art power delivery and cooling mechanisms are available [36, 41, 51].

Going forward, future HBMs [23, 24, 52] should require less power per bit, which is significant since HBM accounts for 40% of our overall power. However, *the processing chiplets, with 50% of power, could become the next significant bottleneck in scaling routers.* Could operators reduce their processing needs if this increases their router capacity? Recent suggestions, such as source routing in SD-WANs [40], may lead the way to future simpler processing.

Acknowledgments

We thank the anonymous reviewers, as well as Sylvia Ratnasamy, Scott Shenker, Alex Krentsel, Sarah McClure, Jose Yallouz, Ilay Yavlovich, Tzu-Chien Hsueh, Yeshaiahu Fainman, Shai Cohen and Yoav Levi for their insightful discussions and comments. This work was partly supported by the Louis and Miriam Benjamin Chair in Computer-Communication Networks and NSF Award No. 2410053.

References

[1] Vamsi Addanki, Maria Apostolaki, Manya Ghobadi, Stefan Schmid, and Laurent Vanbever. 2022. ABM: Active buffer management in datacenters. In *ACM SIGCOMM*. 36–52.

[2] Vamsi Addanki, Wei Bai, Stefan Schmid, and Maria Apostolaki. 2024. Reverie: Low Pass Filter-Based Switch Buffer Sharing for Datacenters with RDMA and TCP Traffic. In *Usenix NSDI*. Santa Clara, CA, 651–668.

[3] Majeed Ahmad. 2025. SK hynix maintains memory leadership with first HBM4. https://www.eetimes.com/sk-hynix-maintains-memory-leadership-with-first-hbm4/.

[4] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. 2004. Sizing router buffers. In *ACM SIGCOMM* (Portland, Oregon, USA). 281–292.

[5] IEEE Standards Association. 2020. 802.1AX-2020 - IEEE standard for local and metropolitan area networks–link aggregation. https://1.ieee802.org/tsn/802-1ax-rev/.

[6] Hagit Attiya, David Hay, and Isaac Keslassy. 2010. Packet-mode emulation of output-queued switches. *IEEE Trans. Comput.* 59, 10 (2010), 1378–1391.

[7] Adnan Aziz, Amit Prakash, and Vijaya Ramachandran. 2003. A near optimal scheduler for switch-memory-switch routers. In *ACM SPAA*. 343–352.

[8] Broadcom. 2025. Tomahawk 5 / BCM78900 Series. In *[Online]. Available: https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm78900-series*.

[9] Joseph Byrne. 2025. Tomahawk 5 switches at 51.2Tbps. In *[Online]. Available: https://www.techinsights.com/blog/tomahawk-5-switches-512tbps*.

[10] S. Chen, S. Pal, and R. Kumar. 2024. Waferscale network switches. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*. 215–229.

[11] Mark Cherney. 2025. Broadcom ships latest networking chip to speed AI. https://www.reuters.com/world/asia-pacific/broadcom-ships-latest-networking-chip-speed-ai-2025-06-03/.

[12] Qingzhong Deng, Ahmed H. El-Saeed, Alaa Elshazly, Guy Lepage, Chiara Marchese, Hakim Kobbi, Rafal Magdziak, Jeroen De Coster, Neha Singh, Marko Ersek Filipcic, Kristof Croes, Dimitrios Velenis, Maumita Chakrabarti, Peter De Heyn, Peter Verheyen, Philippe Absil, Filippo Ferraro, Yoojin Ban, and Joris Van Campenhout. 2024. 32x100 GHz WDM filter based on ultra-compact silicon rings with a high thermal tuning efficiency of 5.85 mW/pi. In *Optical Fiber Communications Conference and Exhibition (OFC)*.

[13] Micha Dery, Orr Krupnik, and Isaac Keslassy. 2023. QueuePilot: Reviving small buffers with a learned AQM policy. In *IEEE Infocom*. 1–10.

[14] Advait Dixit, Pawan Prakash, Y Charlie Hu, and Ramana Rao Kompella. 2013. On the impact of packet spraying in data center networks. In *IEEE Infocom*. 2130–2138.

[15] Orhan Ergun. 2019. LAG vs. ECMP discussion on real network deployments. https://orhanergun.net/lag-vs-ecmp-discussion-on-real-network-deployments.

[16] C. Levy et al. 2023. A 3D-integrated $8\lambda \times 32$ Gbps/λ Silicon Photonic Microring-based DWDM Transmitter. In *2023 IEEE Custom Integrated Circuits Conference (CICC)*. San Antonio, TX, USA, 1–2. https://doi.org/10.1109/CICC56848.2023.10159293

[17] M. Raj et al. 2023. A 0.96pJ/b 7×50Gb/s-per-fiber WDM receiver with stacked 7nm CMOS and 45nm silicon photonic dies. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. San Francisco, CA, USA, 11–13. https://doi.org/10.1109/ISSCC42614.2023.10067306

[18] Y. Li et al. 2024. A -11.6-dBm OMA sensitivity 0.55-pJ/bit 40-Gb/s optical receiver designed using a 2-port-parameter-based design methodology. *IEEE Open Journal of the Solid-State Circuits Society* 4 (Dec. 2024), 328–339. https://doi.org/10.1109/OJSSCS.2024.3376436

[19] Kim Eun-jin. 2025. SK Hynix, Samsung, Micron Compete for HBM4 Market Dominance. https://www.businesskorea.co.kr/news/articleView.html?idxno=253754.

[20] GlobalFoundries. 2025. Next-gen GF Fotonix: Redefining flexibility, bandwidth upgrades & full turnkey support. In *[Online]. Available: https://gf.com/blog/next-gen-gf-fotonix-redefining-flexibility-bandwidth-upgrades-full-turnkey-support/*.

[21] Viral Gosalia. 2024. HBM3E: powering the future of AI with high-bandwidth memory. In *[Online]. Available: https://www.micron.com/about/blog/applications/ai/microns-hbm3e-powering-the-future-of-ai-with-high-bandwidth-memory*.

[22] Nicholas Harris. 2023. Keynote – Ultra-high density photonic interconnect and circuit switching up to the wafer-level with Passage. In *IEEE Hot Intecronnects*.

[23] Po-Kai Hsu, Sungwon Cho, Janak Sharda, Hyeonwoo Park, Suman Datta, and Shimeng Yu. 2025. Monolithic 3D Stackable DRAM Design with BEOL-Compatible Oxide Channel Access Transistor. In *IEEE International Memory Workshop (IMW)*. 1–4.

[24] Po-Kai Hsu, Janak Sharda, Xiangjin Wu, H-S Philip Wong, and Shimeng Yu. 2025. Monolithic 3D Stackable DRAM. *IEEE Nanotechnology Magazine* 19, 2 (2025), 7–16.

[25] Tzu-Chien Hsueh, Bill Lin, Zijun Chen, and Yeshaiahu Fainman. 2025. Panel-Scale Reconfigurable Photonic Interconnects for Scalable AI Computation. *arXiv preprint arXiv:2508.06079* (2025). https://arxiv.org/abs/2508.06079

[26] Jinbin Hu, Jiawei Huang, Wenjun Lv, Yutao Zhou, Jianxin Wang, and Tian He. 2019. CAPS: Coding-based adaptive packet spraying to reduce flow completion time in data center. *IEEE/ACM Transactions on Networking* 27, 6 (2019), 2338–2353.

[27] SK hynix. 2025. SK hynix completes world's first HBM4 development and readies mass production. https://news.skhynix.com/sk-hynix-completes-worlds-first-hbm4-development-and-readies-mass-production/.

[28] Applied Materials Inc. 2021. Applied Materials introduces new technologies and capabilities for accelerating the semiconductor industry's heterogeneous integration roadmap. In *[Online]. Available: https://www.globenewswire.com/news-release/2021/09/08/2293290/0/en/Applied-Materials-Introduces-New-Technologies-and-Capabilities-for-Accelerating-the-Semiconductor-Industry-s-Heterogeneous-Integration-Roadmap.html*.

[29] Sundar Iyer, Ramana Rao Kompella, and Nick McKeown. 2008. Designing packet buffers for router linecards. *IEEE/ACM Transactions on Networking* 16, 3 (2008), 705–717.

[30] S. Iyer and N. McKeown. 2001. Techniques for fast shared memory switches. In *Computer Systems Laboratory, Stanford Univ., Stanford, CA, USA, Tech. Rep. TR01-HPNG-081, 501*.

[31] S. Iyer and N. McKeown. 2003. Analysis of the parallel packet switch architecture. *IEEE/ACM Transactions on Networking* 11, 2 (2003), 314–324.

[32] Van Jacobson. 1990. Modified TCP congestion avoidance algorithm. *Email to the end2end-interest mailing list* (1990). https://www.cs.rice.edu/~eugeneng/teaching/f06/comp529/papers/vanj.90apr30.txt

[33] J.J. Jaramillo, F. Milan, and R. Srikant. 2008. Padded frames: A novel algorithm for stable scheduling in load-balanced switches. *IEEE/ACM Transactions on Networking* 16, 5 (2008), 1212–1225.

[34] JEDEC. 2025. High Bandwidth Memory (HBM4) DRAM. https://www.jedec.org/standards-documents/docs/jesd270-4.

[35] Juniper. 2025. Hashing algorithms for LAG and ECMP. https://www.juniper.net/documentation/us/en/software/junos/interfaces-ethernet-switches/topics/topic-map/hashing-algoriths-lag-bundle-ecmp.html.

[36] P. Kennedy. 2024. Cerebras WSE-3 AI chip launched 56x larger than NVIDIA H100. In *Servers, ServeTheHome Forums*. https://www.servethehome.com/cerebras-wse-3-ai-chip-launched-56x-larger-than-nvidia-h100-vertiv-supermicro-hpe-qualcomm/

[37] Isaac Keslassy. 2004. *The load-balanced router*. Stanford University.

[38] I. Keslassy, S.T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown. 2003. Scaling internet routers using optics. In *ACM SIGCOMM*. 189–200.

[39] Kunal Khullar. 2025. JEDEC finalizes HBM4 memory standard with major bandwidth and efficiency upgrades. https://www.tomshardware.com/pc-components/ram/jedec-finalizes-hbm4-memory-standard-with-major-bandwidth-and-efficiency-upgrades.

[40] Alexander Krentsel, Nitika Saran, Bikash Koley, Subhasree Mandal, Ashok Narayanan, Sylvia Ratnasamy, Ali Al-Shabibi, Anees Shaikh, Rob Shakir, Ankit Singla, and Hakim Weatherspoon. 2024. A decentralized SDN architecture for the WAN. In *Proceedings of the ACM SIGCOMM 2024 Conference*. ACM, Sydney, Australia. https://doi.org/10.1145/3651890.3672257

[41] Y. Kundu, M. Kaur, T. Wig, K. Kumar, P. Kumari, and V. Puri. 2025. A Comparison of the Cerebras Wafer-Scale Integration Technology with Nvidia GPU-based Systems for Artificial Intelligence. In *arXiv preprint, arXiv:2503.11698v1*.

[42] Lightmatter. 2025. Passage L200 3D co-packaged optics. In *[Online]. Available: https://lightmatter.co/products/l200/*.

[43] Lightmatter. 2025. The Passage M1000 3D Photonic Superchip reference platform. In *[Online]. Available: https://lightmatter.co/products/m1000/*.

[44] Bill Lin and Isaac Keslassy. 2010. The concurrent matching switch architecture. *IEEE/ACM Transactions on Networking* 18, 4 (2010), 1330–1343. https://doi.org/10.1109/TNET.2010.2044328

[45] Jie Lu, Jiaqi Gao, Fei Feng, Zhiqiang He, Menglei Zheng, Kun Liu, Jun He, Binbin Liao, Suwei Xu, Ke Sun, et al. 2025. Alibaba Stellar: A New Generation RDMA Network for Cloud AI. In *ACM SIGCOMM*. 453–466.

[46] Nick McKeown, Guido Appenzeller, and Isaac Keslassy. 2019. Sizing router buffers (redux). *ACM SIGCOMM CCR* (2019).

[47] William M Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C Snoeren, and George Porter. 2020. Expanding across time to deliver bandwidth efficiency and low latency. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 1–18.

[48] William M Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C Snoeren, and George Porter. 2017. RotorNet: A scalable, low-complexity, optical datacenter network. In *ACM SIGCOMM*. 267–280.

[49] S. Moazeni et al. 2017. A 40-Gb/s PAM-4 transmitter based on a ring-resonator optical DAC in 45-nm SOI CMOS. *IEEE Journal of Solid-State Circuits* 52, 12 (Dec. 2017), 3503–3516. https://doi.org/10.1109/JSSC.2017.2759467

[50] Timothy Prickett Morgan. 2025. Broadcom stretches switching across datacenters with Jericho 4. https://www.nextplatform.com/2025/08/13/broadcom-stretches-switching-across-datacenters-with-jericho-4/.

[51] H. Mujtaba. 2021. Cerebras unveils its 7nm wafer scale engine 2: The largest AI chip ever built with 2.6 trillion transistors & close to a million cores. In *Announcement, WCCFTECH*. https://wccftech.com/cerebras-unveils-7nm-wafe-scale-engine-2-largest-ai-chip-ever-built/

[52] Hassan Mujtaba. 2025. Next-gen HBM architecture detailed including HBM4, HBM5, HBM6, HBM7 & HBM8: up to 64 TB/s bandwidth, 240 GB capacity per 24-hi stack, embedded cooling. In *[Online]. Available: https://wccftech.com/next-gen-hbm-architecture-detailed-hbm4-hbm5-hbm6-hbm7-hbm8-up-to-64-tbps-bandwidth-240-gb-capacity-per-24-hi-stack-embedded-cooling/*.

[53] Nokia. 2022. LAG and ECMP hashing. https://infocenter.nokia.com/public/7210SAS229R1A/index.jsp?topic=%2Fcom.nokia.TSR_Interface_Guide%2Flag_and_ecmp_ha-ai9j6k6swn.html.

[54] Amit Prakash, Adnan Aziz, and Vijaya Ramachandran. 2004. Randomized parallel schedulers for switch-memory-switch routers: Analysis and numerical studies. In *IEEE Infocom*, Vol. 3. IEEE, 2026–2037.

[55] Amit Prakash, Sadia Sharif, and Adnan Aziz. 2002. An O (log/sup 2/N) parallel algorithm for output queuing. In *IEEE Infocom*, Vol. 3. IEEE, 1623–1629.

[56] A. Rizzo, S. Daudlin, A. Novick, A. James, V. Gopal, V. Murthy, Q. Cheng, B.Y. Kim, X. Ji, Y. Okawachi, and M. van Niekerk. 2022. Petabit-scale silicon photonic interconnects with integrated Kerr frequency combs. *IEEE Journal of Selected Topics in Quantum Electronics* 29, 1 (2022), 1–20.

[57] Ori Rottenstreich, Pu Li, Inbal Horev, Isaac Keslassy, and Shivkumar Kalyanaraman. 2012. The switch reordering contagion: Preventing a few late packets from ruining the whole party. *IEEE Trans. Comput.* 63, 5 (2012), 1262–1276.

[58] Anton Shilov. 2024. SK Hynix says new high bandwidth memory for GPUs on track for 2024 - HBM4 with 2048-bit interface and 1.5TB/s per stack is on the way. In *[Online]. Available: https://www.tomshardware.com/pc-components/gpus/sk-hynix-says-new-high-bandwidth-memory-for-gpus-on-track-for-2024-hbm4-with-2048-bit-interface-and-15tbs-per-stack-is-on-the-way*.

[59] Gireesh Shrimali and Nick McKeown. 2005. Building packet buffers using interleaved memories. In *Proceedings of the 2005 Workshop on High Performance Switching and Routing (HPSR)*. IEEE, 1–5.

[60] Bruce Spang, Brooks Walsh, Te-Yuan Huang, Travis Rusnock, Joe Lawrence, and Nick McKeown. 2019. Buffer Sizing and Video QoE Measurements at Netflix. In *Proceedings of the 2019 Workshop on Buffer Sizing*. ACM, Stanford, CA, 1–7.

[61] G. Sun, C.W. Chang, B. Lin, and L. Zeng. 2012. Oblivious routing design for mesh networks to achieve a new worst-case throughput bound. In *IEEE International Conference on Computer Design (ICCD)*. 427–432.

[62] Ufuk Usubütün, Fraida Fund, and Shivendra Panwar. 2023. Do switches still need to deliver packets in sequence?. In *IEEE HPSR*. IEEE, 89–95.

[63] Lane Wigley. 2022. 8100 & 8200 Deployment Note. *Cisco XRDocs* (March 2022). https://xrdocs.io/8000/blogs/8100-8200-deployment-note/

[64] Lane Wigley. 2022. An update on router buffering. *Cisco White Paper* (March 2022). https://xrdocs.io/8000/Buffering-WP_March_2022.pdf

[65] Wikipedia. 2025. High Bandwidth Memory. https://en.wikipedia.org/wiki/High_Bandwidth_Memory.

[66] Sen Yang, Bill Lin, Paul Tune, and Jun Jim Xu. 2017. A simple resequencing load-balanced switch based on analytical packet reordering bounds. In *IEEE Infocom*. IEEE, 1–9.

[67] Sen Yang, Bill Lin, and Jun Xu. 2017. Safe randomized load-balanced switching by diffusing extra loads. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 1–37. https://doi.org/10.1145/3154498

[68] Jie Zhang, Dafang Zhang, and Kun Huang. 2015. Improving datacenter throughput and robustness with Lazy TCP over packet spraying. *Computer Communications* 62 (2015), 23–33.