DOI: 10.1109/ISVLSI65124.2025.11130195

Ultra-Low-Latency Data Link Layer TX Buffer Architecture for Wireless Communications

***Abstract*—With the upcoming 6G standard, cellular networks will see an expanded feature set, new use cases, and massive increases in key performance indicators (KPIs) for ultra-reliable low-latency communication (uRLLC) and enhanced mobile broadband (eMBB). Achieving Tbps-scale data rates with latencies as low as 100 µs may require rethinking the underlying architecture. Hence, hardware acceleration at all network layers, including the data link layer (DLL), is a promising approach for achieving both high throughput and low latency. We introduce two hardware transmission (TX) buffer designs targeting the DLL of cellular networks, further referred to as *SRAM-based* design and *register array*-based (*RA-based*) design. We discuss their architectures and investigate read, write, and delete latencies, as well as average packet sojourn times under varying ingress rates for designs synthesized with queue counts ranging from 1 to 512, assuming round-robin read access. Our results show mean packet sojourn times as low as 15.554 ns for specific configurations of the RA-based design at 30% of the respective configuration's maximum throughput, peaking at 42.669 µs when operating at 107.54 Gbps—the maximum throughput of the RA-based design with 512 queues. These findings underscore the potential of DLL-level hardware acceleration in meeting 6G's demanding KPIs.**

***Index Terms*—Medium Access Control, 6G, Transmission Buffer, sub-THz, MIMO**

I. Introduction

In 6G, sub-THz frequencies combined with ultra-massive MIMO emerge as a promising solution to achieve both massive data rates and ultra-low latency, which are key for applications such as remote surgery, wireless X-haul, cooperative robotics, and holographic communication [1]–[3]. Rapid progress in the D-Band has already yielded data rates up to 200 Gb/s [4]–[6].

However, making these data rates available within a complex network poses significant challenges and requires support across the entire network stack, including the Physical Layer (PHY) and the Data Link Layer (DLL). The DLL comprises the Medium Access Control (MAC) and Radio Link Control (RLC) layers. The MAC layer is critical for handling HARQ, random access procedures, resource allocation, and shared medium access. A substantial body of research focuses on enhancing MAC functionalities, such as random access procedures and scheduling, primarily from an algorithmic perspective, with AI receiving significant attention [7]–[9].

In contrast, research specifically addressing hardware acceleration of the MAC layer remains comparatively limited, despite its recognized necessity for 6G [10], [11]. Nevertheless, high-performance implementations have been presented by Ehrig and Petri [12] and by Lopacinski et al. [13]. Ehrig and Petri describe a MAC system for broadband 60 GHz communication and elaborate on throughput maximization techniques, while Lopacinski et al. propose a DLL processor in CMOS for terahertz wireless communication, achieving up to 165 Gbit/sec. However, both designs rely on simple FIFO TX buffers and assume point-to-point setups–a significant limitation given that 3GPP-based systems such as 5G and 6G require scalable and efficient point-to-multipoint communication in the RAN.

Linked-list-based shared TX buffers are common in packet switches [14]–[18] and offer the necessary scalability and flexibility needed for large networks, yet most hardware buffer architectures are designed for wired systems and offer limited applicability to wireless communications. Hardware TX buffer architectures for wireless remain underexplored.

Therefore, in this work, we propose two ultra-low-latency shared transmission buffer architectures based on singly linked lists designed for wireless communications. Both designs store packet data in SRAM, with control information placed either in SRAM or dedicated register arrays. We refer to these designs as *SRAM-based* and *RA-based*. The main contributions are:

1) **Shared Buffer Architectures with Parallel R/W:** Two ultra-low-latency shared buffer designs that enable parallel reads and writes with high throughput.
2) **(H)ARQ Support:** Clear separation of read and delete operations facilitates efficient (H)ARQ alongside constant-time, address-based random access – a major difference to existing designs, which makes it suitable for wireless communications.
3) **Latency Analysis:** A thorough evaluation of both proposed designs' read, write, and delete latencies.
4) **Mean Sojourn Time:** An analysis of the mean sojourn time for random uniform writes and round-robin reads, capturing how memory entries flow through the system.

The remainder of the paper is organized as follows. Chapter II reviews related work and its limitations. Chapter III describes the assumed system environment with a focus on the DLL architecture in which our TX buffers operate. Chapter IV details the proposed buffer designs and their key latency properties. Chapters V and VI present analyses of resource utilization, clock frequency, throughput, and latency. Finally, Chapter VII concludes the paper.

II. Related Work

Linked list-based shared buffers have been applied primarily in wired communications, such as Ethernet [14]–[19]. In [14], the authors present a shared packet buffer for IP-based traffic using RLDRAM II memory. Their design employs a configurable time-division multiplexing (TDM) pattern to switch between read and write states but lacks a dedicated

delete operation. Instead, once a packet is read, its memory is returned to a free list that tracks empty cells.

Similar approaches are found in [15], [17], [18] with Dicorato [17] providing an in detail description of their buffer implementation.

Notably, Iwamoto et al. [18] store packet heads in SRAM for low-latency access and offload remaining packet parts to DRAM when packets exceed a certain size. SRAM and DRAM are accessed in parallel, allowing packet tails to become available as soon as the preceding SRAM part is processed. This allows for low-latency, high-throughput buffers built from commercial off-the-shelf (COTS) DRAM. Moreover, by storing packet parts larger than 320 bytes in DRAM, the authors make use of its relative abundance and lower cost, which helps mitigate the effects of internal fragmentation, which is a typical issue with fixed-size allocations.

In [15], the fragmentation problem is addressed by adding a separate tail buffer for packet tails up to 128 bits alongside a 512-bit-wide main buffer.

However, none of these designs include a dedicated delete operation–essential for wireless networks like 5G and future 6G, where (H)ARQ mechanisms are deployed to reduce block error rates (BLER). Under (H)ARQ, packets must remain in the transmission buffer until an acknowledgment from the receiver is received. The same limitation applies to the design by Santos et al. [16], though their work exploits dual-port SRAM to enable true parallel read and write access, without the use of arbitration or duplexing.

Xu et al. [19] present a hardware-based doubly linked list architecture developed for digital image processing. While their design supports a dedicated delete operation that could, in principle, be adapted for (H)ARQ, it only allows a single list. Thus, supporting multiple queues would require instantiating the entire design multiple times, leading to static partitioning of memory resources. This limits scalability, especially when many devices must be served concurrently. Table I compares the discussed buffer designs.

III. System Architecture

We assume a control/data plane split, where user data processing is handled by hardware accelerators based on ASIC or FPGA, while control plane functions mainly utilize commodity components such as CPUs.

User data is forwarded directly to the data plane, which is composed of dedicated hardware components such as TX and RX buffers, along with a scheduler backend that handles frame creation, aggregation, and forwarding to the physical layer based on timing determined by the scheduler frontend running as software in the control unit. Notably, user data bypasses the comparatively slow control plane and is processed end-to-end in hardware.

The scheduler backend also switches between transmit and receive modes when no separate RX and TX paths are available. Furthermore, we assume that the (H)ARQ process is handled entirely within the data plane to enable low-latency retransmissions. No intermediate packet buffers are used for

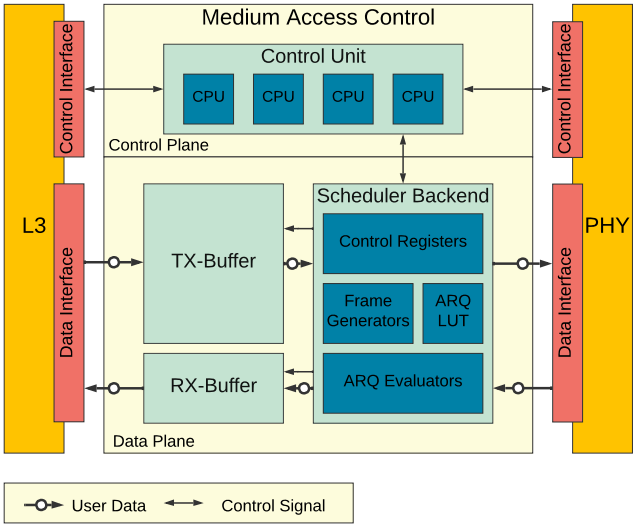


Fig. 1: High-level view of the envisioned MAC architecture, illustrating a control and data plane split where user data is processed by dedicated hardware, and the control unit manages and configures the data plane.

ARQ; instead, retransmissions access packets directly from the TX buffer utilizing a lookup table populated during the initial transmission. By avoiding intermediate buffering, this approach potentially enables a more scalable design by saving on additional ARQ buffer resources while also reducing latency.

IV. Buffer Architecture

We adopt a shared buffer architecture to optimize memory utilization. In contrast to dedicated, fixed-size per-node queues, which result in inefficient memory utilization, shared memory architectures allow all queues to dynamically grow and shrink by accessing a common memory pool. This mitigates memory fragmentation and improves overall system efficiency, especially under varying traffic conditions, by ensuring that available resources can be utilized freely.

We present two designs that utilize the same basic architecture but use either Register Arrays (RA) or dual-port SRAM to hold control information, resulting in different performance behaviors. We refer to the first design as RA-based and the latter as SRAM-based. Both designs use dual-port SRAM to buffer incoming packets and run in a single clock domain.

The logical structure of our architecture resembles an array of singly linked lists (Fig. 2). Each list maintains the frames corresponding to a specific transmission destination. An additional list is used to allocate and deallocate blocks to/from the frame holding lists, also referred to as the *memory pool*. Consequently, frames for any destination can be accessed via an address with worst time complexity $O(1)$, preventing buffer traversal for finding the next applicable frame. The architecture consists of five memory regions, one of which buffers incoming packets. Four additional memory regions

TABLE I: Comparison of transmission buffer hardware designs for various network technologies.

References	Target Application	Connection Type	Delete Operation	Parallel R/W	Memory Technology	Random Access	Allocation Type
[17]	IP Traffic	Wired	No	No	SRAM/DRAM	No	Shared
[15]	Ethernet	Wired	No	No	SRAM/DRAM	No	Shared
[14]	IP Traffic	Wired	No	No	RLDRAM II	No	Shared
[18]	Ethernet	Wired	No	No	SRAM/DRAM	No	Shared
[19]	Image Processing	-	Yes	No	SRAM	Yes	Dedicated
[16]	Ethernet	Wired	No	Yes	SRAM	No	Shared
This Work	5G/6G	Wireless	Yes	Yes	SRAM	Yes	Shared

contain control information and, depending on the design, are saved in either Register Arrays (RA) or dual-port SRAM.

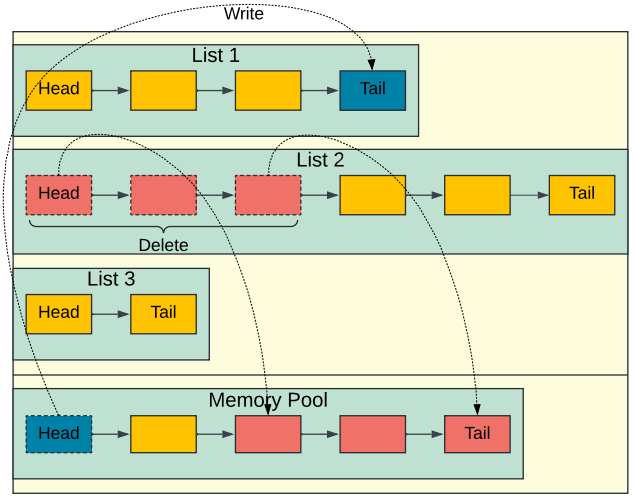


Fig. 2: The transmission buffer is organized as an array of lists. The dark blue block is allocated to List 1 and removed from the head of the memory pool, while the red blocks are deallocated from List 2 and returned to the pool.

a) Memory Controller: The controller consists of two state machines for handling access and keeping the buffer consistent. The access handler manages memory access and data manipulation via the following operations: *read*, *delete*, *size*, and *write*, all of which operate based on a given list index. The collision handler monitors the access handler's state and the delete/write input signals. If a collision is detected, the collision handler ensures control information consistency on the fly without introducing delays, as would be the case with pre-coordinating access via TDM or arbitration (Fig. 3). This is achieved by a finite state machine (FSM) that accounts for all possible interleavings of the delete/write FSMs in the access handler, resolving each potential write-write collision case individually. While exhaustive corner case handling is generally infeasible due to the combinatorial explosion of state interleavings, this approach remains practicable in this case as most interleavings can be trivially excluded from collision detection, as they do not lead to conflicts. Another particularity of our design is that it maintains the length of each list, which may provide useful information to the MAC, for instance, to assess queue occupancy to aid in scheduling decisions. The buffer operations can be summarized as follows:

- **Read:** Retrieves consecutive blocks from head onwards or via passing an address
- **Delete:** Removes a range of blocks for a given list index and appends them to the free list.
- **Size:** Retrieves a given list's length.
- **Write:** Removes a free block from the front of the free list, writes packet data inside, and appends it to the given list.

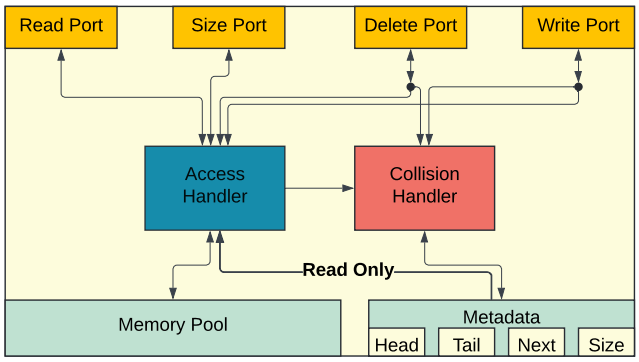


Fig. 3: High-level structure of the singly linked list array.

b) Operation Latency: The *delete*, *read*, and *size* operations are mutually exclusive in the sense that they can't execute in parallel and must either be statically multiplexed in time or an arbitration mechanism must be deployed to resolve conflicts. Write can be executed parallel to any previously mentioned operations without arbitration or multiplexing. An important implication is that the buffer throughput is independent of the interleaving of write and delete/read/size operations. All operations execute in constant time except for the *read* operation of the SRAM-based design, for which the latency to retrieve the first block is 6 clock cycles (cc) while being 3 cc for all subsequent blocks for which the enable signal is not deasserted (Table II).

TABLE II: Latencies of buffer operations in clock cycles (cc).

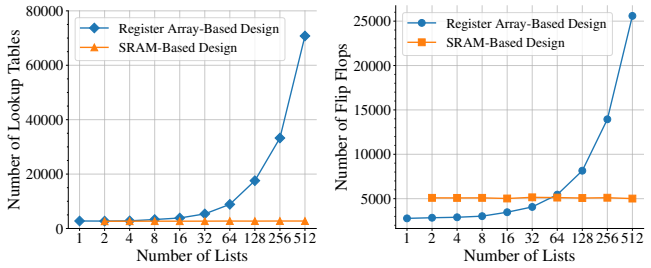
SRAM-Based Design				
Operation	Read	Write	Delete	Size
Latency	3-6 cc	6 cc	6 cc	3 cc
Register Array-Based Design				
Operation	Read	Write	Delete	Size
Latency	3 cc	4 cc	4 cc	1 cc

V. Design Synthesis

The target platform for synthesis is an AMD/Xilinx Zynq UltraScale+ MPSoC ZCU102 evaluation board. We use Vivado 2023.1 as synthesis tool.

1) Design Parameters: We synthesize the SRAM- and RA-based designs with support for $2^n, n \in \{1, 2, \ldots, 9\}$ and $2^n, n \in \{0, 2, \ldots, 9\}$ lists, respectively. The SRAM-based design was not tested for one list support as the block ram configurator of Vivado requires a memory depth of at least two. Additionally, a block size of 2048 bits was selected to assess the performance of the designs under very high bit widths, which are essential for achieving high data rates. The buffer size is 20 Mib (10240 entries). We synthesize every design and configuration for frequencies between 200 and 450 MHz in 10 MHz steps and choose the maximum frequency leading to no timing violations in the post-layout design.

2) Resource Utilization and Clock Frequency: The RA-based design shows less favorable clock frequency degradation properties than the SRAM-based design. While the clock frequency around the maximum of 450 MHz for both designs for queue counts of up to 16, the achievable clock frequency starts to degrade rapidly afterward down to 210 MHz for the RA-based design (Tab. III and Fig. 5). This is explained by the fact that the array of registers has to be constructed from LUTs and FFs, using up increasing resources of the FPGA that are proportional to the queue count. This eventually leads to the congestion of the design implementation (Fig. 4). In contrast, the SRAM-based design behaves virtually static concerning resource utilization of FFs and LUTs as it uses SRAM instead. The additional amount of SRAM needed to accommodate an increased queue count is negligible and, for all tested queue counts, falls below the smallest SRAM primitive available on the ZCU 102, which provides 18 Kbit of memory.



(a) LUT utilization for both RA- and SRAM-based designs.

(b) FF utilization for both RA- and SRAM-based designs.

Fig. 4: Utilization of LUTs and FFs in relation to the number of lists. The x-axes have a logarithmic scale.

VI. Performance Analysis

To confirm the functional correctness of both designs for given synthesis settings and target platform, we run both pre- and post-synthesis simulations. Thereafter, we explore the designs via behavioral simulation for average throughput and latency by applying the respective achievable clock frequencies for configurations described in Chapter V. Every configuration is run 100 times for 50 ms per simulation run. We determine the throughput for every simulation run and sample latency for every packet read and then calculate the averages over the throughput and latency samples. In this context, we define *latency* as the time between the beginning of the write operation and the end of the read operation of a written memory cell.

A. Throughput

We evaluate throughput by writing data to the buffer at an ingress rate equal to the maximum write speed (as determined by the write latency) and reading via a round-robin (RR) schedule. In each write cycle, the queue is chosen uniformly at random. Reading proceeds in four steps:

1) **Choose queue:** Determine the next queue to read from by following an RR schedule.
2) **Determine queue size:** Obtain the number of data blocks inside the chosen queue.
3) **Read data:** Read as many data blocks as determined in the previous step up to a maximum of 128.
4) **Delete data:** Delete all data blocks read and continue with step 1.

All steps except for data reading execute in constant time. The duration of the read step depends on the number of blocks processed. The asymptotic read throughput is the reciprocal of its latency. (For the SRAM-based design, use the lower latency value from Table II. See also Table IV.) As the ingress rate is limited by the write speed, we expect the throughput R to be approximately R_{max}.

$$R_{max} = \frac{W_{\text{data}} f_{\text{clk}}}{\max(N_{\text{clk\_write}}, N_{\text{clk\_read}})} \tag{1}$$

With W_{data} denoting the data width, f_{clk} the operating frequency of the design, $N_{\text{clk\_write}}$ the write latency in clock cycles and $N_{\text{clk\_read}}$ analogous to $N_{\text{clk\_write}}$.

TABLE III: Minimal achieved clock periods for the RA-based (T_{RA}) and the SRAM-based (T_{SRAM}) design.

No. Lists	1	2	4	8	16	32	64	128	256	512
$\mathbf{T_{SRAM}}$	-	2.222 ns	2.222 ns	2.222 ns	2.222 ns	2.222 ns	2.222 ns	2.222 ns	2.222 ns	2.222 ns
$\mathbf{T_{RA}}$	2.222 ns	2.222 ns	2.272 ns	2.222 ns	2.222 ns	2.500 ns	2.564 ns	3.125 ns	3.703 ns	4.761 ns

TABLE IV: Latencies in clock cycles (cc) for the execution steps of RR-based buffer read access; n denotes the number of data blocks read.

SRAM-Based Design			
Choose Queue	**Det. Queue Size**	**Read Data**	**Delete Data**
2 cc	4 cc	(3n + 4) cc	6 cc
Register Array-Based Design			
Choose Queue	**Det. Queue Size**	**Read Data**	**Delete Data**
2 cc	2 cc	(3n + 1) cc	4 cc

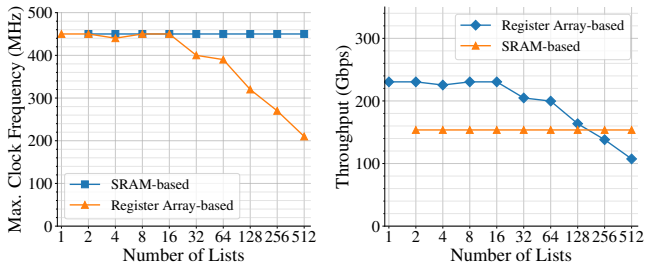


Fig. 5: Achievable clock frequencies and data throughput in relation to the number of lists. The x-axes have a logarithmic scale.

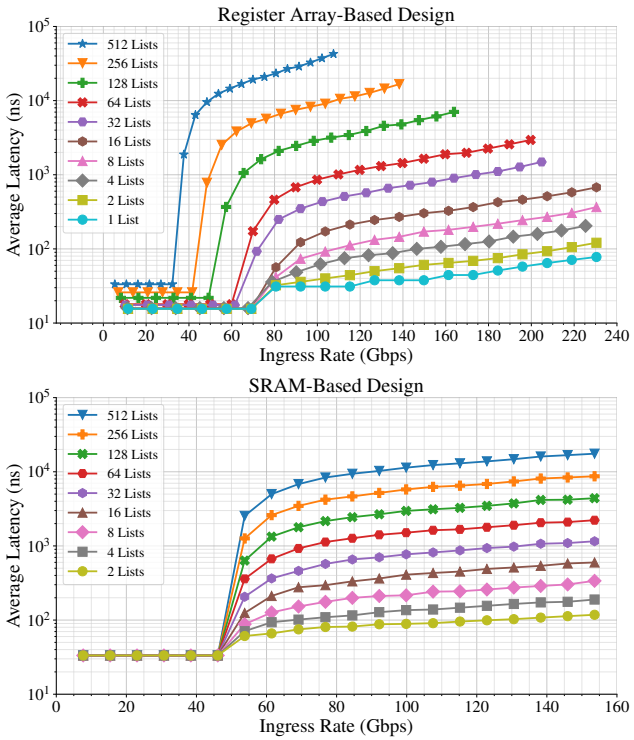


Fig. 6: Latency vs. ingress rate for varying numbers of lists and a block size of 2048 bit. The y-axis has a logarithmic scale.

Simulated throughput closely matches eq. 1; no buffer overflow is observed, and the average number of packets remains well below capacity. The RA-based design's throughput decreases with increasing queue count–reflecting a drop in clock frequency–whereas the SRAM-based design maintains nearly constant throughput. At a queue count of 128, both designs yield similar performance; beyond that, the SRAM-based design outperforms the RA-based design. For instance, the SRAM-based design maintains an average throughput of 153.603 Gbps ($R_{max} = 153.615$ Gbps) across all queue counts, while the RA-based design ranges from 230.423 Gbps for a single queue, to 163.607 Gbps at 128 queues, and settles at 107.358 Gbps for 512 queues.

B. Latency

We analyze latency by varying the ingress rate r from 100% to 5% of R_{max} (eq. 1) in 5% increments. Constant latency is achieved when packets are processed immediately upon entering the RR queue, i. e. when the minimum, average, and maximum latencies are equal. This condition holds when exactly one memory cell is occupied in the buffer at any time and can be expressed as

$$\begin{aligned} L_{\min} = E(L) = L_{\max} \iff \\ R \leq \frac{\max(N_{\text{clk\_write}}, N_{\text{clk\_read}})}{N_{\text{clk\_RR\_min}}} R_{\max} \iff \\ R \leq \frac{W_{\text{data}} f_{\text{clk}}}{N_{\text{clk\_RR\_min}}}, \end{aligned} \tag{2}$$

where $N_{\text{clk\_RR\_min}}$ denotes the number of clock cycles of a single RR cycle when exactly one memory cell is processed within the accessed queue. For the SRAM-based design, this condition is satisfied as long as the write rate does not exceed $(19)^{-1}$ packets per clock cycle ($L = 15$ cc), and for the RA-based design, it holds for rates up to $(12)^{-1}$ packets per clock cycle ($L = 7$ cc). These values represent the reciprocals of the total latency of a single round-robin cycle when exactly one memory cell is processed, as shown in Table IV. In simulations, constant latency is maintained for all ingress rates not exceeding 30% of R_{max}. Beyond this threshold, the latency increases abruptly, coherent with eq. 2. The calculated cut-off points for constant latency (see eq. 2) are 33.33% of R_{max} for the RA-based design and 31.58% of R_{max} for the SRAM-based

design, as can be verified by Tables II and IV.

When also considering clock frequency, this translates to a best-case latency of 15.554 ns for the RA-based design with 1, 2, 8 or 16 lists, assuming the ingress rate does not exceed 30% of $R_{\max}$, and 33.33 ns for all SRAM-based designs under the same rate conditions.

The worst average latencies are observed for 512 lists in both the RA- and SRAM-based designs, reaching 42.669 µs at an ingress rate of 107.540 Gbps and 17.535 µs at 153.615 Gbps, respectively, both corresponding to 100% of $R_{\max}$.

VII. Conclusion

Hardware acceleration of the data link layer offers the potential to combine high performance with low energy consumption in the data link layer. We propose two hardware transmission buffer designs that provide low access latency, high scalability, and the ability to support hundreds of devices simultaneously. In particular, the SRAM-based design demonstrates strong scalability, as detailed in Chapter VI.

We achieve constant buffer access latencies of 7 clock cycles for the register-array-based (RA) design and 15 clock cycles for the SRAM-based design for data rates up to 30% of $R_{\max}$, assuming round-robin read access. When accounting for clock frequency, this translates to a best-case latency of 15.554 ns for the RA-based design (with 1, 2, 8, or 16 lists) and 33.33 ns for the SRAM-based design, both well below the 100 µs latency target envisioned for 6G. Even under worst-case conditions with 512 lists at maximum throughput (100% $R_{\max}$), average latencies remain at 42.669 µs (RA) and 17.535 µs (SRAM), thus meeting the ultra-low latency demands of 6G. Additionally, both designs sustain high clock frequencies even for large buffer sizes, enabling aggregate throughputs in the hundreds of Gbps, representing an important step toward the Tbps-range peak data rates targeted for 6G.

References

[1] ETSI, "Thz technology: a roadmap towards thz communications and networking," ETSI, Tech. Rep. ETSI GR THz 001 V1.1.1, Oct. 2022, online. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/THz/001_099/001/01.01.01_60/gr_THz001v010101p.pdf

[2] M. A. Uusitalo *et al.*, "6G Vision, Value, Use Cases and Technologies From European 6G Flagship Project Hexa-X," *IEEE Access*, vol. 9, pp. 160 004–160 020, 2021, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9625032

[3] S. A. Abdel Hakeem, H. H. Hussein, and H. Kim, "Vision and research directions of 6G technologies and applications," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, Part A, pp. 2419–2442, Jun. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157822001033

[4] A. Karakuzulu, W. A. Ahmad, D. Kissinger, and A. Malignaggi, "A Four-Channel Bidirectional D-Band Phased-Array Transceiver for 200 Gb/s 6G Wireless Communications in a 130-nm BiCMOS Technology," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 5, pp. 1310–1322, May 2023, conference Name: IEEE Journal of Solid-State Circuits. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10005812

[5] W. Deng *et al.*, "A D-Band Joint Radar-Communication CMOS Transceiver," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 2, pp. 411–427, Feb. 2023, conference Name: IEEE Journal of Solid-State Circuits. [Online]. Available: https://ieeexplore.ieee.org/document/9810171/?arnumber=9810171

[6] D. Kissinger, G. Kahmen, and R. Weigel, "Millimeter-Wave and Terahertz Transceivers in SiGe BiCMOS Technologies," *IEEE Transactions on Microwave Theory and Techniques*, vol. 69, no. 10, pp. 4541–4560, Oct. 2021, conference Name: IEEE Transactions on Microwave Theory and Techniques. [Online]. Available: https://ieeexplore.ieee.org/document/9499518/?arnumber=9499518

[7] A. Valcarce, P. Kela, S. Mandelli, and H. Viswanathan, "The Role of AI in 6G MAC," in *2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, Jun. 2024, pp. 723–728, iSSN: 2575-4912. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10597082

[8] J. Du, C. Jiang, J. Wang, Y. Ren, and M. Debbah, "Machine Learning for 6G Wireless Networks: Carrying Forward Enhanced Bandwidth, Massive Access, and Ultrareliable/Low-Latency Service," *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 122–134, Dec. 2020, conference Name: IEEE Vehicular Technology Magazine. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9206115

[9] A. Nauman, T. N. Nguyen, Y. A. Qadri, Z. Nain, K. Cengiz, and S. W. Kim, "Artificial Intelligence in Beyond 5G and 6G Reliable Communications," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 73–78, Mar. 2022, conference Name: IEEE Internet of Things Magazine. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9773090

[10] V. Ziegler, H. Viswanathan, H. Flinck, M. Hoffmann, V. Räisänen, and K. Hätönen, "6G Architecture to Connect the Worlds," *IEEE Access*, vol. 8, pp. 173 508–173 520, 2020, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/9200631/?arnumber=9200631

[11] A. Shahraki, M. Abbasi, M. J. Piran, and A. Taherkordi, "A Comprehensive Survey on 6G Networks:Applications, Core Services, Enabling Technologies, and Future Challenges," Jun. 2021, arXiv:2101.12475 [cs]. [Online]. Available: http://arxiv.org/abs/2101.12475

[12] M. Ehrig and M. Petri, "60 GHz broadband MAC system design for cable replacement in machine vision applications," *AEU - International Journal of Electronics and Communications*, vol. 67, no. 12, pp. 1118–1128, Dec. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1434841113002185

[13] L. Lopacinski *et al.*, "Data Link Layer Processor for 100 Gbps Terahertz Wireless Communications in 28 nm CMOS Technology," *IEEE Access*, vol. 7, pp. 44 489–44 502, 2019, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/8685101/?arnumber=8685101

[14] C. Toal, D. Burns, K. McLaughlin, S. Sezer, and S. O'Kane, "An RLDRAM II Implementation of a 10Gbps Shared Packet Buffer for Network Processing," in *Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*, Aug. 2007, pp. 613–618. [Online]. Available: https://ieeexplore.ieee.org/document/4291975

[15] M. Lau *et al.*, "Gigabit Ethernet switches using a shared buffer architecture," *IEEE Communications Magazine*, vol. 41, no. 12, pp. 76–84, Dec. 2003, conference Name: IEEE Communications Magazine. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1252802

[16] R. Santos, R. Marau, A. Oliveira, P. Pedreiras, and L. Almeida, "Designing a costumized Ethernet switch for safe hard real-time communication," in *2008 IEEE International Workshop on Factory Communication Systems*, May 2008, pp. 169–177. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4638737

[17] P. G. Dicorato, "Hardware linked-list using FPGAs and optical central stage impairments in a sectored packet switch with an optical core (demonstrator)," Ph.D. dissertation, University of Ottawa (Canada), 2007. [Online]. Available: http://hdl.handle.net/10393/27830

[18] H. Iwamoto, Y. Yano, Y. Kuroda, K. Yamamoto, S. Ata, and K. Inoue, "Deterministic High Density Packet-Buffer System for Low Cost Network Systems," in *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, Mar. 2012, pp. 951–956, iSSN: 2332-5658. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6184971

[19] J. Xu, Y. Dou, J. Song, Y. Zhang, and F. Xia, "Design and Synthesis of a High-Speed Hardware Linked-List for Digital Image Processing," in *2008 Congress on Image and Signal Processing*, vol. 4, May 2008, pp. 171–175. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4566638