ENGINEERING Information Technology & Electronic Engineering
https://www.academax.com/EITEE; https://www.engineering.org.cn
https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=11344065
E-mail: eng.itee@zju.edu.cn

Engineering Journals
Available on IEEE Xplore®

Research Article
https://doi.org/10.1631/ENG.ITEE.2025.0180

De-blocking adaptive feedback control design for shared-buffer CIOQ switching architecture

Rui ZHENG[1], Jianliang SHEN[1]✉, Fan ZHANG[2], Ping LV[1], Peijie LI[1], Yu SHAO[1], Zhengbin ZHU[3]

[1]*Information Engineering University, Zhengzhou 450001, China*
[2]*College of Computer Science and Artificial Intelligence, Fudan University, Shanghai 200433, China*
[3]*The Chinese Armed Police Force Command College, Tianjin 300250, China*

Abstract: To address the issues of head-of-line (HOL) blocking at the virtual output queue (VOQ) level, packet loss, and congestion spreading caused by buffer overflow in the shared-buffer-based combined input and output queued (CIOQ) switching architecture, while enhancing its performance and stability, we propose a de-blocking adaptive feedback control (AFC) design in this study. The introduction of the credit timeout detection mechanism (CTDM) enables the CIOQ to achieve theoretical 100% non-blocking state, effectively eliminating the impact of HOL blocking. With the combined effect of the proposed VOQ dynamic regulation algorithm (VDRA) and threshold dynamic adaptive algorithm (TDAA), it can reduce the risk of congestion spreading caused by buffer overflow and consequently improve the overall performance of the system. Both theoretical analysis and experimental results demonstrate that, under typical traffic conditions, the proposed design achieves a maximum throughput of 1499.66 Gb/s and a minimum latency of 83 ns. Additionally, the effective throughput ratio reaches 96.94%, with a data link layer packet (DLLP) loss ratio of merely 0.61% and a packet loss rate as low as 0.6%. In comparison with traditional CIOQ and input queued (IQ) switch architectures, the proposed design demonstrates improvements in throughput by 15.12% and 20.55%, and forwarding latency is reduced by 26.9% and 54.7%, respectively, and the system stability is stronger, which can fully satisfy the demand for data exchange in complex situations.

Key words: Shared-buffer CIOQ switching architecture; Head-of-line (HOL) blocking; Congestion spreading; Adaptive feedback control (AFC); Peripheral component interconnect express (PCIe) interconnect protocol

1 Introduction

As the third-generation high-speed serial expansion bus standard succeeding instruction set architecture (ISA) and peripheral component interconnect (PCI) (Hou et al., 2024), peripheral component interconnect express (PCIe) delivers high bandwidth, low latency, and reliability. These advantages enable its ubiquitous adoption in core infrastructure domains including hyperscale data centers (HDC), hyper-converged infrastructure (HCI), high-performance computing (HPC), and artificial intelligence (AI). In practice, however, there is a significant mismatch between the physical link speed and the forwarding performance of the switching architecture (Bandara et al., 2024). On the one hand, with the iterative evolution of the PCIe technology standard, its physical layer link speed continues to break through, and the bidirectional theoretical throughput can be as high as 512 GB/s in the PCIe 7.0 protocol (Nag, 2023). On the other hand, the existing switching equipment packet processing capability exhibits obvious technical lag, and its actual forwarding rate and throughput can't effectively match the high-speed transmission capacity provided by the underlying physical link (Zhou et al., 2024). This technological development imbalance leads to the transmission pressure of the entire network gradually shifting to the switching architecture of the core node, which handles the packet forwarding performance and load stability issues, becoming a key bottleneck restricting the performance of the entire network.

The main task of the switching architecture is to provide a way to route packets from the input port to the output port quickly and efficiently. As a null-separated non-blocking

✉ Jianliang SHEN, shenjianliang@outlook.com
Rui ZHENG, https://orcid.org/0009-0004-6458-5439
Jianliang SHEN, https://orcid.org/0009-0000-5647-043X
Fan ZHANG, https://orcid.org/0000-0001-7456-8377
CLC number: TP393.02

Received: Dec. 19, 2025; Revision accepted: Jan. 23, 2026;
Crosschecked: Feb. 6, 2026; Published online: Mar. 16, 2026

© The Authors 2026. Published by Zhejiang University Press Co., Ltd. This is an open access article distributed under the terms of the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

switching architecture, the crossbar switching architecture has been widely used in various routers and switches (Zhang et al., 2025). It is functionally equivalent to multiple parallel buses; it can simultaneously complete the matching between multiple input and output ports in parallel, and compared to the bus-based switching architecture, it has stronger switching capability. In the crossbar switching architecture implementation, the combined input and output queued (CIOQ)-type architecture with shared-buffer fully integrates the advantages of traditional input queued (IQ) and output queued (OQ) switching architecture (Chen BW, 2025). By setting up dual buffers at the input and output ends, it not only provides throughput close to IQ and OQ in a diverse input traffic environment, but also effectively balances the traffic and mitigates transient congestion, thereby providing robust quality of service (QoS) support. Additionally, compared to combined input crosspoint queued (CICQ)-type switching architecture, although CIOQ is more prone to suffer from head-of-line (HOL) blocking (Hu B et al., 2018), CIOQ does not need to add buffers at the crosspoints. This not only simplifies the complexity of buffer resource management, but also significantly reduces the hardware implementation cost. Furthermore, CIOQ maintains inherent advantages in switching latency and throughput, providing more than adequate performance in some specific scenarios.

Fig. 1 illustrates the current mainstream shared-buffer CIOQ switching architecture in the industry. Its inputs and outputs employ a dual-buffering system to maintain a shared-buffer, which is used to buffer data streams with different characteristics (Hu JB et al., 2023). The input side implements virtualized storage pools through a shared-memory input module (SIM), while the output side implements physical queue management with a shared-memory output module (SOM). The architecture uses a linked-list-based storage management mechanism, where SIM and SOM maintain PCIe packets stored in the form of a linked-list, and the scheduler extracts packets from the buffer and sends them by parsing the linked-list information, thus it can fully use the fragmented memory (Ouyang et al., 2018). In addition, to mitigate the HOL blocking of input ports, CIOQ uses the virtual output queue (VOQ) mechanism. In this approach, each input port maintains separate buffer queues for different output ports. Based on traffic characteristics, each VOQ adaptively adjusts the depth of its logical first in first out (FIFO) queue (Dagli and Belviranli, 2024). Since all packets in the same VOQ share the same destination output port, port-level HOL blocking is avoided.

However, there are still many bottlenecks in the existing shared-buffer CIOQ switching architecture. First, while the VOQ mechanism confines HOL blocking to individual VOQs within an input port (Mohtavipour et al., 2020), it fails to eliminate its fundamental impact; deterministic network performance thus requires complete blocking removal (Yébenes et al., 2019). Second, all VOQs in the input port share the buffer pool with a dynamic adjustment mechanism instead of allocating a fixed buffer space for each VOQ. Although HOL blocking in one VOQ does not immediately affect others, prolonged blocking consumes substantial shared-buffer space. As accumulated packets fill the buffer, available resources for other VOQs diminish rapidly, triggering cascading packet loss. Moreover, in highly interconnected topologies, complex flow control mechanisms can propagate such issues from a single switch to the entire network (Kim et al., 2024), degrading latency and throughput, disrupting load balancing, and potentially causing network-wide stagnation. Concurrently, bursty traffic is an essential consideration in the CIOQ switching architecture. In typical emerging scenarios—such as multi-physics field iterative computation in HPC fluid simulation (Xu et al., 2023), gradient synchronization in distributed machine learning training, and time-sensitive stream processing in real-time data analysis—the high-speed network must handle large-scale data transmission. Data traffic in these cases arrives in bursts and is highly non-uniform (Kaltenhauser et al., 2024). The transmission rate during these bursts is much higher than average, creating short-term traffic peaks. This unpredictability further raises the risk of CIOQ buffer overflow. Furthermore, there is an increase in time-sensitive scenarios in human–computer interaction (HCI) cloud services (Luo et al., 2024), and such nanosecond-level transmission demands require more robust switching architectures to support such intensive data exchanges (Palnitkar and Kanade, 2024), thus placing higher requirements on the performance and stability of the switching architectures, and further highlighting the impact of VOQ level HOL blocking on CIOQ switching architecture.

The main contributions of this study are as follows:

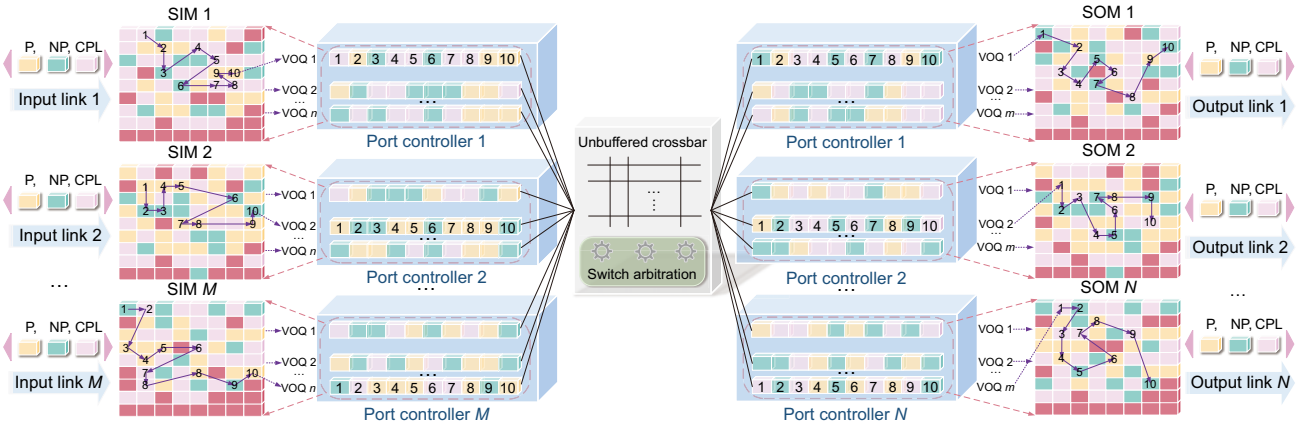

Fig. 1 Schematic diagram of the shared-buffer CIOQ switching architecture

1. Leveraging credit-based flow control, we design the credit timeout detection mechanism (CTDM), which aims to identify and detect potential HOL blocking risks in the input ports of the CIOQ switching architecture at an early stage. This mechanism updates the feedback information rapidly at the hardware level and triggers the corresponding algorithm in time before the HOL blocking occurs, thus removing the HOL blocking at the VOQ level and ultimately achieving the theoretically 100% non-blocking CIOQ switching architecture.

2. To enhance the adaptability of CTDM to different traffic patterns, taking full consideration of the current traffic load, historical port rate, and other factors in the input VOQ, the threshold dynamic adaptive algorithm (TDAA) is proposed for dynamically adjusting the threshold value of the triggering speed regulation algorithm in CTDM. This algorithm not only enhances the flexibility of CTDM's response to multiple traffic scenarios but also effectively avoids frequent triggering of the algorithms, thereby reducing the CIOQ system overhead.

3. Aiming at the transient high load problem caused by bursty traffic, we finely regulate the sending rate of VOQs in the input port in this study. The dynamic deceleration and acceleration algorithms are designed for VOQs with and without credit blocking timeout on the target port, and the effective control of VOQ sending rate is realized at the hardware level. This method not only reduces the risk of congestion spreading due to buffer overflow but also significantly improves the overall performance and operational stability of the CIOQ switching architecture.

2 Related works

In CIOQ switching architecture, the matching from input port to output port is realized by the scheduling algorithm of the switching matrix, and in recent years, scholars have conducted extensive research on the design of CIOQ switching architecture scheduling algorithms. Ran et al. (2024) optimized the iterative simple loop iterative protocol (iSLIP) scheduling algorithm based on the CIOQ architecture and proposed the PMQF_iSLIP algorithm, which prioritizes full queues to improve throughput and is able to use bandwidth resources more efficiently than the traditional iSLIP algorithm. However, the introduction of additional judgment logic leads to a dramatic increase in its hardware complexity, and this algorithm also has the potential risk of increased latency. Firoozshahian et al. (2007) proposed the feedback loop grant scheduler (FLGS) scheduling algorithm. This algorithm is able to fully emulate the performance of the OQ switching architecture with a boost speed of 2. The algorithm possesses a localization feature, so as to reduce the need for global information exchange, but it's hardware implementation is more complex and it needs to be carefully designed to ensure that the scheduling process is fully localized. Ran et al. (2023) proposed the exhaustion of priority service empty queues and mixed weights (EPEMW) algorithm for the CIOQ switching architecture, this method dynamically adjusts the weight values to improve the fairness of VOQ packet forwarding; however, the algorithm performs poorly in the face of bursty traffic, and it is difficult to maintain a stable forwarding efficiency. Hu B et al. (2018) proposed the head-of-line removal framework (HRF) single iteration scheduling algorithm to simplify the scheduling process of the CIOQ switching architecture, but there is still considerable room for improvement in its overall performance.

At the early stage of these designs, such algorithms pay more attention to the improvement of scheduling performance, while ignoring the characteristics and defects of the CIOQ switching architecture, and at the same time incurring higher hardware resource overhead. For example, iSLIP, parallel iterative matching (PIM), round-robin with port allocation (RPA), iterative longest port first (iLPF), and their derived algorithms have a complexity of at least $O(N^2)$, while an algorithm such as maximum urgency-based crossbar scheduling (MUCS) has a complexity of even $O(N^3)$. Although it is possible to improve the algorithmic efficiency to a certain extent by using pipelined scheduling, great matching scheduling, and deterministic offline scheduling, due to the limitations of the low-voltage differential signaling (LVDS) transmission technology and the packaging technology, it is difficult for the current CIOQ switching architectures to be scaled up to more than T-bit switching capacity. Consequently, simply improving the performance of the scheduling algorithm can no longer solve the problems of memory bandwidth bottleneck, throughput latency performance guarantee, and QoS guarantee faced by the current CIOQ switching architecture, and it is urgently needed to seek breakthroughs from the two dimensions of the switching architecture's architectural system and flow control mechanism.

For this reason, Shen et al. (2018) proposed a feedback-driven hierarchical switching architecture based on the reverse transmission mechanism, which realizes the dynamic adjustment of the scheduling strategy by introducing a feedback mechanism. The hierarchical structure effectively reduces port conflict probability and scheduler complexity compared to single-stage switching architecture. Shen et al. (2019) further optimized the two-level switching architecture to enable each level of scheduling to make full use of the feedback information for scheduling decisions, thus improving matching accuracy and flexibility. However, if the feedback information is not updated in a timely manner, additional scheduling latency is triggered. Zyla et al. (2024) proposed the shared dynamic matching grant (SDMG) CIOQ switching architecture, which achieves a theoretical 100% throughput, but the design of the architecture is extremely complex and the hardware cost is rather high, restricting its wide application in practical deployment. Dong et al. (2024) proposed a software-defined multiprotocol switching architecture based on the CIOQ switching architecture, which takes full advantage of the flexibility of the CIOQ architecture to build a switching circuit that can support four high-speed interconnect protocols simultaneously. By inputting the downstream configuration rules from the port controllers, the architecture can realize a fast response to changes in network topology and traffic patterns. However, supporting multiple protocols also requires more complex hardware and software co-design, thus significantly increasing the development and maintenance costs of the system. In summary, although existing studies have optimized the CIOQ switching architecture to varying degrees, they still show clear shortcomings in handling instantaneous resource contention, buffer overflow-induced packet loss, and VOQ queue blocking caused

by unexpected traffic. The system's performance therefore has considerable room for improvement, and more in-depth research and innovation in both architecture design and flow control mechanisms are urgently needed.

3 Adaptive feedback control design

To effectively address the performance bottleneck faced by the current CIOQ switching architecture in practical applications and the difficulties caused by the VOQ level HOL blocking, based on the credit flow control mechanism in the PCIe protocol (Wu et al., 2025), we propose an adaptive feedback control (AFC) mechanism oriented to the ontology of the CIOQ switching architecture in this study. As shown in Fig. 2, the mechanism mainly consists of three parts: the CTDM, TDAA, and VOQ dynamic regulation algorithm (VDRA) composition. A timeout detection module is deployed at each output port in the CIOQ to perform credit timeout detection for the three types of packets in the PCIe. This module is able to sense the risk of HOL blocking at the corresponding input port well in advance of blocking, and triggers the VOQ adjustment algorithm for the corresponding input port in a timely manner. To further enhance the system's adaptability to different traffic patterns, by combining key parameters such as current load conditions and historical sending rates, this study designs TDAA, which is used to dynamically adjust the threshold setting of the triggered speed regulation algorithm in CTDM. In addition, at each input port level, by adopting two different types of traffic shaping techniques for VOQs with credit blocking timeout triggered and remaining untriggered timeout at the destination port, this study designs the timeout-queue deceleration algorithm (TQDA) and non-timeout-queue acceleration algorithm (NTAA), respectively. The above mechanism realizes the fine-grained control of the VOQ sending rate at the hardware level, thus supporting the release of the shared-buffer space in a timely manner and reducing the risk of congestion spreading due to buffer overflow, so that the CIOQ switching architecture can still maintain excellent performance and stability under high load and bursty traffic scenarios. The related symbols used in the algorithms of this study are described in Table 1.

Table 1 Description of the main symbols in the algorithm of this study

Notation	Description
R_{current}	The current sending rate of VOQs
L_{current}	The current link load flow value
$T_{\text{threshold}}$	Real-time trigger threshold value
Δt	Size of the real-time counting window
T_{base}	Baseline value of the threshold
$R_{\max}$	Maximum rate the VOQ is allowed to send
$L_{\max}$	Maximum rate the link is allowed to send
γ	Rate-sensitivity factor
δ	Load-sensitivity factor
$T_{\max}$	A preset maximum threshold number of times
$T_{\min}$	The value of the minimum trigger threshold
size_i	Size of each packet received by the VOQ
size_j	Size of packets received on the entire port
R_{hist}	VOQ historical transmit rate
R_{new}	Updated VOQ sending rate
α, β	Speed reduction adjustment parameters
η	Smoothing factor in exponential moving average (EMA)
$R_{\text{hist\_old}}$	Historical transmission rate
R_{accel}	Target transmission rate after speed increase
α_{accel}	Acceleration factor in NTAA
β_{accel}	Traffic-sensitivity factor in NTAA

3.1 Credit timeout detection mechanism

CTDM is designed to identify and deal with potential HOL blocking problems in the CIOQ switching architecture at an early stage, thus empowering the input ports with the ability to sense credit-deficient states in advance. The schematic of its principle is shown in Fig. 3. The core lies in predicting the risk of HOL blocking at the input port in advance by monitoring the credit volume of the output port in real time. When the number of prolonged credit insufficiencies for a VOQ is detected to exceed a predetermined threshold, it is determined that a credit blocking timeout has occurred for that input VOQ port, and the rate adjustment algorithm for the input port is immediately triggered, thereby realizing active intervention and mitigation of the congested state.

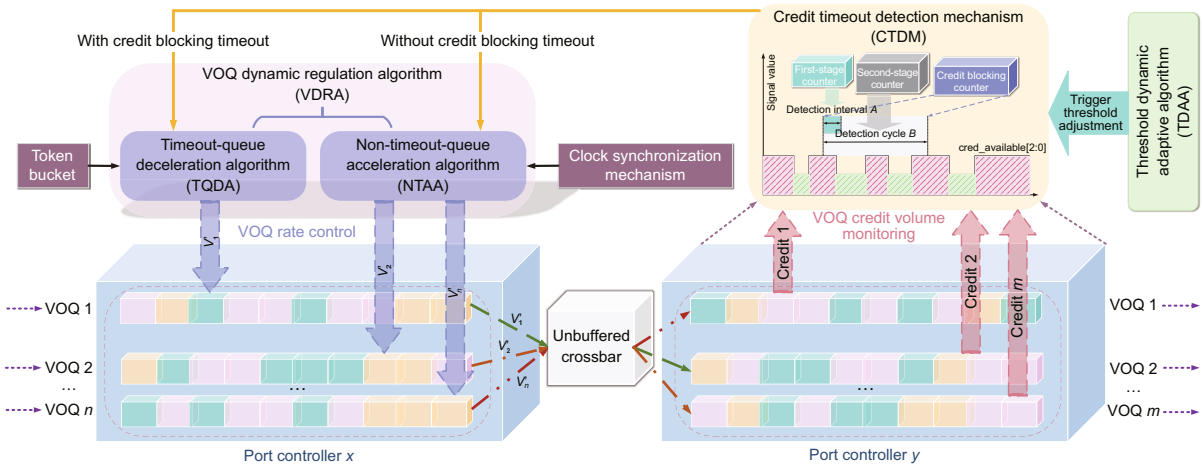

Fig. 2 Schematic diagram of AFC

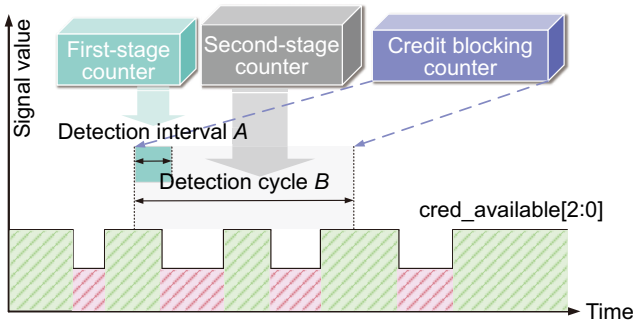

Fig. 3 Schematic diagram of CTDM

In Fig. 3, the cred_available[2:0] signal is an indication signal in the output port in which the current credit amount is not enough to send a maximum load packet, where the bits [0], [1], and [2] denote the posted (P), non-posted (NP), and completion (CPL) packets in the PCIe protocol, respectively. Each transaction type maintains an independent credit counter at the output end, with an initial value determined by the allocated buffer space and maximum packet length. When a packet is enqueued at the input end, the corresponding credit for that type is consumed. Upon successful transmission of the packet from the output end, the credit is released. As a signal monitored in real time within CTDM, the input stops sending packets when this signal is low, and the signal being held low for an extended period is regarded as an abnormality, indicating that there may be a potential blocking problem. Therefore, the CTDM uses the amount of time in which the cred_available[2:0] signal remains low as the basis for detection. In practice, the CTDM uses a periodic monitoring mechanism, which divides each detection cycle B into a number of detection intervals A. During the detection cycle B, it determines whether the cred_available[2:0] signal is persistently low during each detection interval A. If so, the value of the credit blocking counter is increased; otherwise, if the signal is pulled high at least once, it proceeds to the next round of detection. At the end of the detection cycle B, it is judged whether the value of the credit blocking counter exceeds the set threshold value, and once the threshold is exceeded, it is recognized that a credit volume blocking timeout has occurred, and the VDRA process is immediately triggered and initiated to implement the necessary rate adjustment measures.

As the basis of AFC design, by automatically identifying and processing potential HOL blocking at the hardware level, CTDM reduces the dependence on external intervention, improves the autonomous decision-making capability of the system, and avoids the risk of HOL blocking occurring in a single VOQ, which in turn leads to the degradation of the performance of the entire switching port or even the switching architecture. In addition, by triggering the sending rate of the port queue, the generation of HOL blocking at the VOQ level is effectively blocked, and the overall performance of the switching architecture can be improved while enhancing the stability of the system. At the specific hardware implementation level, CTDM adopts a two-stage counter structure to reduce the bit-width requirement of credit blocking counters, thus enhancing the flexibility of threshold configuration while effectively saving hardware resources.

Specifically, the detection interval counter serves as a first-stage counter for maintaining the time length of the detection interval A, clearing zero at the boundary of the detection interval A, and restarting the cycle of counting. Since only a short time cycle needs to be maintained, this counter has a small bit width. Moreover, the detection cycle counter is used to maintain the time of detection cycle B, so its bit width is also relatively small. As opposed to using a single large-bit width counter to achieve the same functionality, the two-stage counter architecture can work in concert with a finite bit width so that the accumulated values of the credit blocking counters are jointly maintained, ultimately significantly reducing the hardware overhead of the CIOQ switching architecture.

3.2 Threshold dynamic adaptive algorithm

Different application scenarios have differences in tolerance and handling strategies for credit blocking (de la Rosa et al., 2025), so the threshold parameter for triggering the VDRA in CTDM needs to be flexibly adjusted according to the actual traffic size and load state. In scenarios with high transmission rates or strong bursty traffic impacts, CIOQ is more likely to have insufficient credits, and at this time, the threshold should be lowered so that the VDRA can be triggered in advance to cope with the impact of bursty traffic. In contrast, when the transmission rate is low or the traffic is more uniform and stable, the threshold can be raised appropriately, thus avoiding too frequent triggering of the VDRA and unnecessary system interventions, and ultimately reducing the overall resource consumption and overhead of the CIOQ switching architecture during operation. Based on the above guidelines, this study designs the TDAA based on the current VOQ transmission rate, traffic load, and other factors, which dynamically calculates and updates the threshold $T_{\text{threshold}}$ by collecting the current sending rate R_{current} and link load flow value L_{current} of each VOQ in real time, where the formulas of R_{current} and L_{current} are shown in Eqs. (1) and (2):

$$R_{\text{current}} = \frac{\sum_{i-1}^{n} \text{size}_i}{\Delta t}, \tag{1}$$

$$L_{\text{current}} = \frac{\sum_{j-1}^{m} \text{size}_j}{\Delta t}. \tag{2}$$

The update formula for the threshold can be expressed as

$$T_{\text{threshold}} = T_{\text{base}} \left(1 - \gamma \frac{R_{\text{current}}}{R_{\max}} - \delta \frac{L_{\text{current}}}{L_{\max}}\right). \tag{3}$$

Bringing Eqs. (1) and (2) into Eq. (3) yields the complete threshold update formula, as shown in Eq. (4).

$$T_{\text{threshold}} = T_{\text{base}} \left(1 - \gamma \frac{\sum_{i=1}^{n} \text{size}_i}{\Delta t R_{\max}} - \delta \frac{\sum_{j=1}^{m} \text{size}_j}{\Delta t L_{\max}}\right), \tag{4}$$

where the T_{base} is a baseline value of the threshold that represents the default threshold when there is no rate or load influence, and it can provide a reference point so that the threshold is dynamically adjusted with respect to the change in T_{base}, ensuring that the threshold is not too small in extreme cases. $R_{\max}$ and $L_{\max}$ represent the maximum rate that the VOQ is allowed to send and the maximum rate that the link

is allowed to send, respectively. They are used in the TDAA to implement a fine-grained control of the packet sending rate normalization, and their values can be derived directly from the specific version of the PCIe protocol, and then stored in registers in advance. Taking $R_{\max}$ as an example, it is used to normalize the current sending rate R_{current} of the current VOQ and convert the maximum rate to a relative proportional value $R_{\text{current}}/R_{\max}$. It can more accurately reflect the proportion of the current sending rate relative to the maximum capacity, thus measuring the impact of the current sending rate on the threshold adjustment and ensuring that the adjustment does not exceed this upper limit and not lead to the risk of overloading. When R_{current} is close to $R_{\max}$, it indicates that this VOQ is close to working at full capacity, at which time the threshold is lowered by the formula to prevent congestion, and on the contrary, the threshold can be raised appropriately to avoid frequent triggering. This strategy enhances the robustness of the algorithm so that it can react more accurately to changes in the network state. Additionally, the rate-sensitivity factor γ and the load-sensitivity factor δ are used to dynamically adjust the threshold $T_{\text{threshold}}$ in the CTDM according to the current rate and traffic load, and they can be used to control the influence of the current sending rate of the VOQ and the ratio of the traffic load of the whole link relative to the maximum processing capacity on the threshold, respectively, and their values will be discussed in Section 4.1. The pseudo-code description of the TDAA is shown in Algorithm 1.

Algorithm 1 TDAA

Input: packet counter, size_j, size_i, Δt, $R_{\max}$, $L_{\max}$, and T_{base}
Output: updated threshold $T_{\text{threshold}}$
loop
  count1 $\leftarrow 0$, count2 $\leftarrow 0$
  sum_packets $\leftarrow 0$, sum_size $\leftarrow 0$
  for each input port i **do**
    sum_packets $\leftarrow$ sum_packets + size_j
    sum_size $\leftarrow$ sum_size + size_i
  end for
  $L_{\text{current}} \leftarrow \dfrac{\text{sum\_size}}{\Delta t}$ // Compute current link load
  $R_{\text{current}} \leftarrow \dfrac{\text{sum\_packets}}{\Delta t}$ // Compute current sending rate
  $T_{\text{threshold}} \leftarrow T_{\text{base}}\left(1-\gamma\dfrac{R_{\text{current}}}{R_{\max}}-\delta\dfrac{L_{\text{current}}}{L_{\max}}\right)$
  // Dynamically update timeout threshold
  if $T_{\text{threshold}} < T_{\min}$ **then**
    $T_{\text{threshold}} \leftarrow T_{\min}$
  end if
  return $T_{\text{threshold}}$
end loop

In the hardware implementation of the TDAA, R_{current} and L_{current} need to be acquired by an additional hardware module. R_{current} as the value that requires real-time calculation, is directly calculated by setting a simple counter in each VOQ, counting the amount of data sent within the most recent time window Δt, and combining the size of Δt to calculate the current transmission rate, thereby achieving regular updates of R_{current}. Moreover, L_{current} is obtained through the traffic monitoring module in the CIOQ, specifically, by configuring a global counter group at the CIOQ input ports, which counts the total traffic of all input ports.

3.3 VDRA

The VDRA consists of two parts: the TQDA and NTAA. It is mainly used to accurately control the sending rate of individual VOQs, so as to cope with the instantaneous high load problem brought by bursty traffic scenarios. Specifically, VOQs that experience credit blocking timeouts are decelerated, thereby slowing down the consumption of credit volume. In contrast, based on the current link load conditions, the remaining VOQs are accelerated to ensure that the CIOQ system performance does not significantly decline. The core idea is to use the token bucket traffic shaper and clock synchronization mechanism to ensure that VOQ packets are sent at a predetermined rate at the hardware level.

3.3.1 TQDA

At the output port, when the number of times that a certain type of packet credit is continuously missing and reaches the set threshold, to provide sufficient time for the recovery of the credit, the relevant VOQs destined for the port are dynamically decelerated so that the VOQs send packets slowly, thus avoiding the generation of HOL blocking at the VOQ level, and at the same time ensuring the continuous release of the shared-buffer. To this end, the TQDA is designed in this study, and the specific regulation strategy is described as follows. First, the historical transmission rate R_{hist} maintains the historical sending rate of this VOQ and is used as the basis for rate reduction. Second, if the threshold in CTDM is set smaller, indicating a more severe credit deficit, then the rate should be further reduced, and the VOQ sending rate should be updated in real time through Eq. (5):

$$R_{\text{new}} = \alpha R_{\text{hist}}\left(1-\beta\frac{T_{\max}}{T_{\text{threshold}}}\right). \tag{5}$$

Here, $T_{\max}$ is a preset maximum threshold number of times, which is used to normalize the effect of $T_{\text{threshold}}$. This can maintain consistency and comparability across different network conditions, allowing the algorithm to provide a more stable response. When $T_{\text{threshold}}$ is small, the value of $(1-\beta T_{\max}/T_{\text{threshold}})$ will decrease, while R_{new} will decrease significantly, resulting in a further reduction in the sending rate. Moreover, α and β are used as adjustment parameters for the TQDA; they are used to modulate rate reduction intensity so that overly aggressive slowdowns do not lead to an overall degradation of CIOQ performance. In addition, R_{hist} is not a pre-determined fixed value, and its value is obtained using the exponential moving average (EMA) method to smooth data fluctuations. EMA calculates a new average by weighting the current sending rate R_{current} and the historical average according to Eq. (6). It should be noted that the fixed window length Δt of the time window based method will lead to response lag or overreaction when facing real-time changes, and compared to the overhead of frequent computation and storage of the time window Δt method, the EMA has more flexibility and lower computational complexity, which can effectively minimize the impact of short-term fluctuations on the results and thus provide more stable rate estimation, especially

for data streams with bursty character.

$$R_{\text{hist}} = \left(1-\eta\right) R_{\text{hist\_old}} + \eta R_{\text{current}}, \tag{6}$$

where η acts as a smoothing factor in EMA, which determines the weights of the old and new data in the calculation. A smaller η implies more reliance on the historical rate $R_{\text{hist\_old}}$, while a larger η focuses more on relying on the current rate R_{current}. The pseudo-code for TQDA is shown in Algorithm 2.

Algorithm 2 TQDA

Input: R_{current}, $R_{\text{hist\_old}}$, α, β, η, $T_{\text{threshold}}$, count3, and cred_available[x]
Output: updated rate R_{new}
$R_{\text{hist}} \leftarrow (1-\eta)R_{\text{hist\_old}} + \eta R_{\text{current}}$
// EMA update of historical rate
if cred_available[x] = 0 **then**
    count3 $\leftarrow$ count3 + 1
else
    count3 $\leftarrow$ 0
end if // Update timeout counter based on credit availability from // CTDM
if count3 > $T_{\text{threshold}}$ **then**
    $T_{\max} \leftarrow$ get_max_wait_time() // Get from CTDM
    $R_{\text{new}} \leftarrow \alpha R_{\text{hist}} \left(1-\beta \dfrac{T_{\max}}{T_{\text{threshold}}}\right)$
else
    $R_{\text{new}} \leftarrow R_{\text{hist}}$ // No deceleration needed
end if
return R_{new}
Update token bucket generation rate to R_{new}

In the hardware implementation of the TQDA, to realize the flexible and precise control of the sending rate of a single VOQ, and to ensure that the sending rate of the VOQ is adjusted independently without affecting the rest of the queue, this study employs the traffic shaping technique based on the token bucket to realize the TQDA. Specifically, based on the updated R_{new} in the TQDA, it is used to update the generation rate of tokens in the token bucket for that VOQ. Based on the size of the PCIe packet, the token generator calculates the number of tokens that need to be generated per unit of time to match the target sending rate. While the transmitter is responsible for the actual sending of the packet, and only when there are enough tokens in the token bucket, the packet is taken out of the corresponding VOQ and forwarded. When the destination output port recovers the credit volume, it indicates that the credit volume blocking timeout state is lifted, and then the timing stops and returns to the original rate for transmission. It is worth noting that while this throttling strategy introduces a slight local throughput penalty, it effectively prevents global performance degradation caused by HOL blocking. Consequently, it achieves higher effective throughput and lower average latency at the system level.

3.3.2 NTAA

Blocking generation is actively avoided in the TQDA by decreasing the sending rate of the potentially HOL-blocking VOQs, but it introduces certain potential drawbacks. The reduction of the VOQ sending rate may adversely affect the overall transmission performance of the CIOQ switching architecture, especially in terms of increased latency and decreased throughput. At the same time, as the sending rate decreases, the release speed of the shared-buffer space slows down. However, upstream links remain unaware of the switch's internal state and continue enqueueing packets into the shared-buffer. This causes the FIFO queues of decelerated VOQs to grow continuously, potentially occupying the entire buffer and generating numerous "victim packets" (i.e., innocent packets) in other VOQs, ultimately leading to overflow and packet loss. For this reason, we design the NTAA for VOQs without credit blocking timeout, and it also takes into full consideration of the current link load L_{current}, historical transmit rate R_{hist}, and other factors, ensuring that it does not cause new congestion on the destination output port, and fully accelerating the sending rate of this type of VOQs. This not only accelerates the buffer release, but also compensates for the possible performance loss of the TQDA. To ensure that the accelerated rate does not exceed the maximum rate that the VOQ is allowed to send $R_{\max}$, this study uses Eq. (7) to calculate the new target transmission rate R_{accel}:

$$R_{\text{accel}} = \min\left\{ R_{\text{hist}} \left(1+\alpha_{\text{accel}} \left(1-\beta_{\text{accel}} \frac{L_{\text{current}}}{L_{\max}}\right)\right),\ R_{\max} \right\}. \tag{7}$$

The core of the formulation is to regulate the acceleration magnitude by linking load information L_{current} and historical rate R_{hist}, thus ensuring appropriate acceleration when the CIOQ load is low and maintaining a more conservative acceleration strategy when the load is close to saturation. Here, α_{accel} is used as an acceleration factor to control the acceleration magnitude, and β_{accel} is used as a traffic-sensitivity factor to adjust the acceleration magnitude according to the flow magnitude. $1+\alpha_{\text{accel}}\left(1-\beta_{\text{accel}} L_{\text{current}}/L_{\max}\right)$, as the acceleration coefficient, fully takes into account the effect of the current link load on the acceleration magnitude. As L_{current} approaches $L_{\max}$, $L_{\text{current}}/L_{\max}$ approaches 1, so the acceleration factor approaches 1; at this point, no significant acceleration takes place. When L_{current} is small, $L_{\text{current}}/L_{\max}$ is small, and therefore, the acceleration factor is large, and then the VOQ is controlled for significant acceleration. In addition, $\min\left(\bullet, R_{\max}\right)$ ensures that the new target sending rate does not exceed $R_{\max}$, thus avoiding the introduction of new output port congestion. The pseudo-code description of the NTAA is shown in Algorithm 3.

In the AFC mechanism, the number of VOQs performing acceleration within the same port is much larger than the number of VOQs performing deceleration. Although the token bucket-based traffic shaping technique has more flexibility and accuracy, its hardware implementation requires maintaining separate token generation and consumption mechanisms for each VOQ, which leads to a significant increase in resource overhead, making it difficult to apply to scenarios of large-scale concurrent acceleration of VOQs. Therefore, unlike the token bucket-based traffic shaping technique used in the TQDA, the NTAA employs a synchronization control mechanism based on high-precision clock signals when implementing VOQ acceleration. This mechanism works by using highly accurate clock signals, thereby ensuring that packets are sent at predetermined intervals. Specifically, a high-precision clock provided by a clock source is used as a reference, and the time interval is tracked by a timer. Each time the timer expires, it is checked

Algorithm 3 NTAA

Input: R_{hist}, α_{accel}, β_{accel}, $L_{\max}$, $R_{\max}$, timeout counter count3, and threshold $T_{\text{threshold}}$
Output: accelerated rate R_{accel}
if count3 $< T_{\text{threshold}}$ **then**
  loop
    $L_{\text{current}} \leftarrow$ get_current_load()
    // Obtain real-time link load from traffic monitor
    $R_{\text{accel}} \leftarrow \min\left\{R_{\text{hist}}\left(1 + \alpha_{\text{accel}}\left(1 - \beta_{\text{accel}} \cdot \frac{L_{\text{current}}}{L_{\max}}\right)\right),\ R_{\max}\right\}$
    // Dynamically increase transmission rate under low load
    break
  end loop
else
  $R_{\text{accel}} \leftarrow R_{\text{hist}}$ // No acceleration under timeout
end if
return R_{accel}
Apply clock-synchronized rate control using R_{accel}

to see if there is a pending packet to be sent. The time interval is adjusted by dynamic calculations to achieve a transmission rate of R_{accel}. This mechanism can effectively achieve a specific VOQ rate increase without changing the physical underlying link rate.

3.3.3 Optimality theory proof of the VDRA

In the VDRA, for the same input port, dynamic balance of overall system resources and performance optimization is achieved by accelerating non-credit-blocking timeout VOQs via the NTAA while simultaneously applying the TQDA to reduce the transmission rate of VOQs that may cause HOL blocking. For the performance impact of the TQDA on the CIOQ, if we only consider that a particular VOQ needs to send the same number of packets before and after the rate reduction, the adoption of TQDA will theoretically lead to an increase in the time required for that VOQ to complete forwarding due to the rate reduction, which may have a negative impact on the overall transmission efficiency of the system. Therefore, to verify the positive performance value of the TQDA in practical applications from the theoretical level, this study further conducts the following theoretical derivation, aiming at proving that, under the premise of taking into account the network congestion and the cost of waiting for blocking, the total time required to complete the forwarding of the same number of packets after adopting the TQDA is shorter than that without this algorithm. The idea of analyzing the proof is realized in two ways: first, the transmission efficiency after speed reduction; second, the effect of blocking timeout on the overall transmission efficiency. After a blocking timeout has occurred for a particular VOQ, we assume scenarios with and without the TQDA, respectively, and use the inverse method to compare the magnitude of the time they take.

Assuming that without the TQDA, due to the occurrence of HOL blocking, the packets in the VOQ need to wait for a period of time T_{wait} to resume transmission. During this period, no packets are sent out; the total transmission time can be expressed as

$$T_{\text{Non\_TQDA}} = T_{\text{wait}} + D/R_{\text{hist}}, \tag{8}$$

where D is the total number of packets transmitted.

Assuming that with the TQDA, the VOQ will send data at rate R_{new}, and that this rate is maintained throughout (ignoring the effects of subtle factors), the total transmission time can be expressed as

$$T_{\text{TQDA}} = D/R_{\text{hist}}. \tag{9}$$

Bringing Eq. (5) into Eq. (9) yields

$$T_{\text{TQDA}} = D \Big/ \left(\alpha R_{\text{hist}}\left(1 - \beta\frac{T_{\max}}{T_{\text{threshold}}}\right)\right). \tag{10}$$

Since when a credit times out to the point of generating a HOL blockage, it means that the VOQ waits close to $T_{\text{threshold}}$, due to the adjustment factor α being less than 1; therefore,

$$\alpha\left(1 - \beta\frac{T_{\max}}{T_{\text{threshold}}}\right) < 1. \tag{11}$$

To simplify the analysis, assume that $T_{\text{wait}} = kT_{\text{threshold}}$, where k is greater than 1. Taking this into Eq. (8) yields

$$T_{\text{Non\_TQDA}} = kT_{\text{threshold}} + D/R_{\text{hist}}. \tag{12}$$

Thus, to prove $T_{\text{Non\_TQDA}} > T_{\text{TQDA}}$, it is only necessary to prove

$$\frac{D}{\alpha R_{\text{hist}}\left(1 - \beta T_{\max}/T_{\text{threshold}}\right)} < kT_{\text{threshold}} + \frac{D}{R_{\text{hist}}}. \tag{13}$$

After multiplying both sides by $\alpha R_{\text{hist}}\left(1 - \beta T_{\text{threshold}}/T_{\max}\right)$, expand and combine as terms to obtain

$$\begin{aligned} D <& kT_{\text{threshold}}\alpha R_{\text{hist}}\left(1 - \beta\frac{T_{\max}}{T_{\text{threshold}}}\right) \\ &+ D\alpha\left(1 - \beta\frac{T_{\max}}{T_{\text{threshold}}}\right). \end{aligned} \tag{14}$$

From inequality (11), it can be seen that the second additive term on the right-hand side of inequality (14), $D\alpha\left(1 - \beta T_{\max}/T_{\text{threshold}}\right)$, is less than D. Therefore, when measuring the impact of the TQDA, it is only necessary for $kT_{\text{threshlod}}\alpha R_{\text{hist}}\left(1 - \beta T_{\max}/T_{\text{threshold}}\right)$ to be small enough. We can ensure that inequality (13) holds true, and $kT_{\text{threshlod}}\alpha R_{\text{hist}}\left(1 - \beta T_{\max}/T_{\text{threshold}}\right)$ actually represents the relationship between the additional transmission time and the HOL blocking time after the introduction of the TQDA; if it is small enough, it proves that the impact of the speed reduction can be ignored. In practice, k is usually greater than but very close to 1. In this case, assuming that the value of $T_{\text{threshold}}$ is also small and $k \approx 1$, there is

$$\begin{aligned} & kT_{\text{threshold}}\alpha R_{\text{hist}}\left(1 - \beta\frac{T_{\max}}{T_{\text{threshold}}}\right) \\ \approx\ & T_{\text{threshold}}\alpha R_{\text{hist}}\left(1 - \beta\frac{T_{\max}}{T_{\text{threshold}}}\right), \end{aligned} \tag{15}$$

whereas the adjustment factors α and β are very small values (usually <0.8 or even smaller); therefore, inequality (15) can

be further simplified as

$$T_{\text{threshold}}\alpha R_{\text{hist}}\left(1-\beta\frac{T_{\max}}{T_{\text{threshold}}}\right)\leq T_{\text{threshold}}\alpha R_{\text{hist}}. \quad (16)$$

Since the right term in inequality (16) is a very small value, it shows that, theoretically, the waiting time T_{wait} caused by HOL blocking is much larger than the additional time consumed by the TQDA, thus proving that, with reasonable parameter configurations, it takes less time to transmit the same number of packets with the TQDA, and the VOQ transmission latency is lower. In addition, the NTAA is used in VDRA to accelerate the processing of a larger number of VOQs, which can further improve the packet forwarding efficiency of the whole port. Therefore, on the basis of the above, to strengthen the theoretical foundation for analyzing the performance impact of the VDRA, we prove the inference of the impact of the TQDA and the NTAA on the latency when they work together in this study, and the proof process is as follows.

Suppose that the total number of VOQs in a certain input port is N, and the number of VOQs that need to be decelerated is M, and the relationship satisfies $M \ll N$, then the VOQs that need to be accelerated are $N-M$. In this case, the total transmission time without using the VDRA can be expressed as

$$\begin{aligned}T_{\text{Non\_VDRA}} &= M\left(kT_{\text{threshold}}+\frac{D}{R_{\text{hist}}}\right)+(N-M)\frac{D}{R_{\text{hist}}}\\ &= N\frac{D}{R_{\text{hist}}}+MkT_{\text{threshold}}.\end{aligned} \quad (17)$$

The total transmission time overhead when using the VDRA is divided into two parts, i.e., $T_{\text{VDRA}}=T_{\text{TQDA}}+T_{\text{fast}}$. For the M VOQs that need to be slowed down, the total time to send them is T_{TQDA}, and for the $N-M$ VOQs that need to be accelerated for processing, the transmission time can be expressed as

$$\begin{aligned}T_{\text{fast}} =&\frac{D}{R_{\text{accel}}}\\ =&D/\min\left\{R_{\text{hist}}\left(1+\alpha_{\text{accel}}\left(1-\beta_{\text{accel}}\frac{L_{\text{current}}}{L_{\max}}\right)\right),\ R_{\max}\right\}.\end{aligned} \quad (18)$$

The differential value ΔT between Eqs. (17) and (18) represents the difference in time consumed by the two strategies, and to ensure that the data transmission time of the whole CIOQ must be shorter than that without VDRA after adopting VDRA, it is necessary to ensure that $\Delta T>0$. After bringing Eqs. (17) and (18) into ΔT, the simplification leads to

$$\begin{aligned}&N-M\frac{1}{\alpha\left(1-\beta\frac{T_{\max}}{T_{\text{threshold}}}\right)}-(N-M)\\ &\cdot\frac{1}{\min\left\{1+\gamma\left(1-\frac{L_{\text{current}}}{L_{\max}}\right),\frac{R_{\max}}{R_{\text{hist}}}\right\}}\\ &>-\frac{MkT_{\text{threshold}}R_{\text{hist}}}{D}.\end{aligned} \quad (19)$$

The VDRA in this study is essentially a "low load then accelerating" algorithm, while at high loads the acceleration needs to be controlled. In inequality (19), a low link load yields a smaller $L_{\text{current}}/L_{\max}$, thereby increasing $L_{\text{current}}/L_{\max}$ and enhancing acceleration. Meanwhile, although $\alpha<1$ implies a larger $1/\alpha$, the term $\beta T_{\text{threshold}}/T_{\max}$ remains small, keeping the overall impact manageable. Therefore, to make inequality (19) hold, the key lies in the choice of the parameters in $1/\left(\alpha\left[1-\beta\left(T_{\max}/T_{\text{threshold}}\right)\right]\right)$ and $1/\min\left\{\left[\left(1+\gamma\left(1-L_{\text{current}}/L_{\max}\right)\right)\right],R_{\max}/R_{\text{hist}}\right\}$. If the link load is low and γ is set slightly larger, the inequality is guaranteed to hold even if α is smaller. When the link load is low, i.e., L_{current} is small and the degree of link idleness is large, it exactly satisfies the condition that the VDRA can be sufficiently accelerated, which in turn can improve the performance of the whole CIOQ switching architecture. Therefore, in summary, through the individual discussion of the TQDA and the comprehensive derivation proof of the VDRA composed of TQDA and NTAA, it is clearly demonstrated that the proposed algorithm is theoretically highly feasible and can significantly improve the latency performance of the CIOQ switching architecture with appropriate parameter settings.

4 Test and simulation analysis

The experimental process of this study primarily encompasses four aspects: algorithm parameter tuning, performance simulation comparison of switching architectures, universal verification methodology (UVM) simulation and hardware encoding testing, and HCI scenario performance comparison. This establishes a complete logical progression from simulation analysis and specific design to scenario-based testing, effectively validating the validity and universality of the design proposed in this study.

4.1 Algorithm parameter determination

This study uses the Opnet 14.5 simulation platform to construct a 4×8 switching model for simulation analysis. To ensure the representativeness and scalability of simulation results in typical data center and HPC network environments, experimental parameters are strictly configured based on the actual characteristics of mainstream hardware platforms. Specifically, the link bandwidth cap ($R_{\max}$=28.8 GB/s) corresponds to the effective throughput after accounting for approximately 90% of PCIe Gen4 protocol overhead, which is widely used in interconnect scenarios for NVIDIA or AI accelerators. Additionally, the time window Δt=50 ns aligns with the scheduling granularity of high-speed switching chips like Broadcom Tomahawk, effectively capturing the micro-dynamics of bursty traffic. Finally, buffer thresholds T_{base}=4 and $T_{\max}$=6 are based on common VOQ queue depths, covering conditions from light load to near-congestion. Additionally, $T_{\min}$=1 ensures that at least one timeout occurs in the environment. The traffic model employs uniform and bursty modes, representing steady-state and burst data flows, respectively, aligning with typical traffic characteristics. Other common parameters are set to η=1.2, $A=25$ ns, $L_{\max}$=28.8 GB/s, and $B=150$ ns.

In the algorithms presented in this study, parameters are mutually independent and do not exhibit a master-dependent relationship. This decoupled design between algorithms achieves exceptional separability during parameter

tuning, avoiding local optima traps caused by parameter coupling. Therefore, to analyze the impact of parameter values in different algorithms on the performance of the CIOQ exchange structure, this study employs a control variable method for simulation design. Within each algorithm, by altering one parameter while fixing all others, we analyze how the modified constant value affects performance and select the optimal parameter setting. This process is repeated across different algorithms, continuing until the combination of constant parameters that yields the best CIOQ performance is identified.

Specifically, in the TDAA, parameters δ and γ are both constant values ranging from 0 to 1. A larger δ value makes the threshold more sensitive to real-time traffic load changes, making it suitable for bursty traffic scenarios. A smaller δ value keeps the threshold relatively stable, reducing unnecessary throttling operations. Furthermore, as shown in Eq. (4), for dynamic threshold updates, a larger γ value can further reduce switching delays. In the VDRA, compared to other parameters, the throttling factors α and β, acceleration factor α_{accel}, and traffic-sensitivity factor β_{accel} exert a more significant influence on the outcome. A larger α ensures a smaller throttling magnitude. The value is chosen if system stability is prioritized. A larger β value indicates stronger dependence of the deceleration mechanism on the current traffic state. Furthermore, larger values for parameters α_{accel} and β_{accel} yield more pronounced acceleration effects. However, excessive acceleration may cause downstream output port congestion, leading to a decline rather than an improvement in CIOQ performance. Based on these considerations, the experimental parameter values for the algorithm defined in this study are shown in Table 2.

Table 2 Parameter settings for the sensitivity analysis of TDAA, TQDA, and NTAA

Algorithm	Key parameter	Preset value	Packet flow model
TDAA	γ	0.1, 0.5, 0.7 ($\delta = 0.4$)	Uniform flow mode & Burst flow mode
	δ	0.1, 0.3, 0.5 ($\gamma = 0.5$)	
TQDA	α	0.1, 0.5, 0.9 ($\beta = 0.4$)	Burst flow mode
	β	0.1, 0.4, 0.8 ($\alpha = 0.5$)	
NTAA	α_{accel}	0.1, 0.7, 0.9 ($\beta_{\text{accel}} = 0.5$)	Burst flow mode
	β_{accel}	0.2, 0.6, 0.8 ($\alpha_{\text{accel}} = 0.5$)	

In the uniform packet arrival mode, the effects of parameters γ and δ on performance are shown in Figs. 4 and 5. In the burst packet arrival mode, the effects of parameters γ and δ on performance are shown in Figs. 6 and 7. Under the burst data flow model, the effects of parameters α and β on performance in the TQDA are shown in Figs. 8 and 9. The effects of parameter variations α_{accel} and β_{accel} on performance in the NTAA are shown in Figs. 10 and 11.

Longitudinal comparisons reveal that under both data flow modes, as parameters γ and δ increase, the overall latency of the CIOQ switching structure decreases while throughput

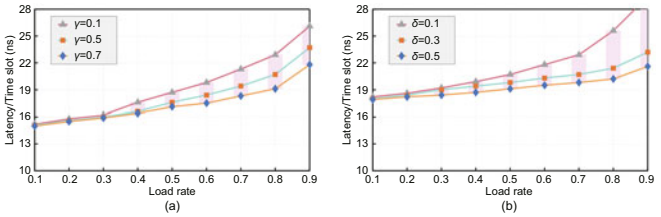

Fig. 4 Latency sensitivity analysis of parameters γ and δ in uniform packet arrival mode: (a) δ is fixed at 0.4 and γ = 0.7, 0.5, and 0.1; (b) γ is fixed at 0.5 and δ = 0.5, 0.3, and 0.1

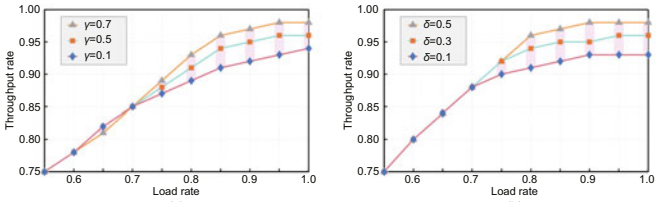

Fig. 5 Throughput sensitivity analysis of parameters γ and δ in uniform packet arrival mode: (a) δ is fixed at 0.4 and γ = 0.7, 0.5, and 0.1; (b) γ is fixed at 0.5 and δ = 0.5, 0.3, and 0.1

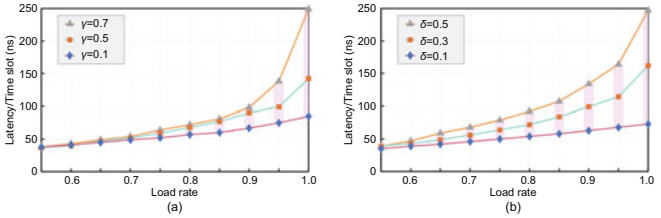

Fig. 6 Latency sensitivity analysis of parameters γ and δ in packet burst arrival mode: (a) δ is fixed at 0.4 and γ = 0.7, 0.5, and 0.1; (b) γ is fixed at 0.5 and δ = 0.5, 0.3, and 0.1

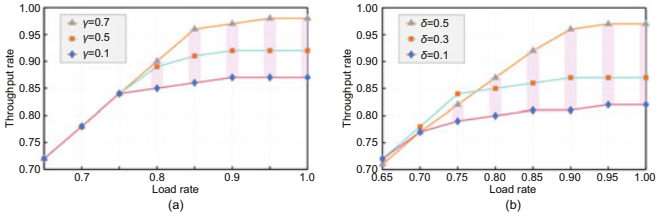

Fig. 7 Latency sensitivity analysis of parameters γ and δ in packet burst arrival mode: (a) δ is fixed at 0.4 and γ = 0.7, 0.5, and 0.1; (b) γ is fixed at 0.5 and δ = 0.5, 0.3, and 0.1

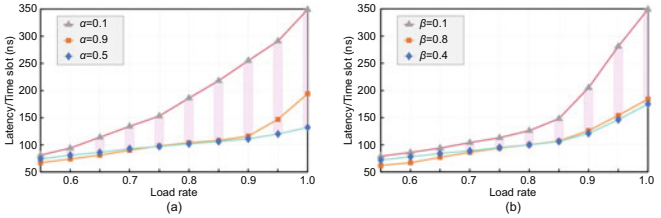

Fig. 8 Impact of TQDA parameters on delay performance in bursty traffic mode: (a) β is fixed at 0.4 and α = 0.9, 0.5, and 0.1; (b) α is fixed at 0.5 and β = 0.8, 0.4, and 0.1

correspondingly increases. However, Figs. 6 and 7 reveal that parameter δ exerts a more pronounced influence on switch performance, indicating heightened sensitivity to parameter δ during threshold updates. Therefore, to ensure high system

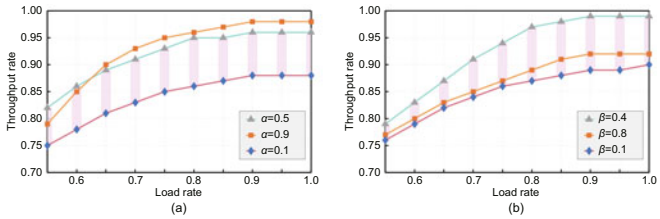

Fig. 9 Impact of TQDA parameters on throughput-rate performance in bursty traffic mode: (a) β is fixed at 0.4 and $\alpha = 0.9$, 0.5, and 0.1; (b) α is fixed at 0.5 and $\beta = 0.8$, 0.4, and 0.1

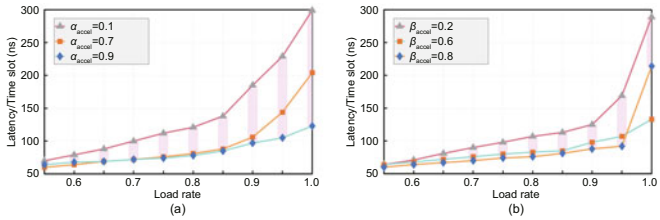

Fig. 10 Impact of NTAA parameters on delay performance in bursty traffic mode: (a) β_{accel} is fixed at 0.5 and $\alpha_{\mathrm{accel}} = 0.9$, 0.7, and 0.1; (b) α_{accel} is fixed at 0.5 and $\beta_{\mathrm{accel}} = 0.8$, 0.6, and 0.2

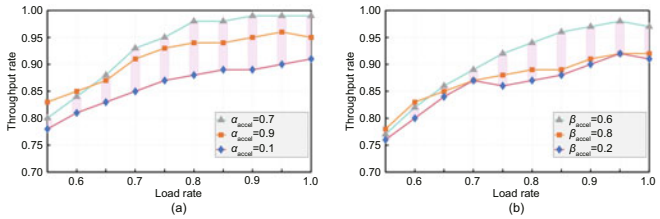

Fig. 11 Impact of NTAA parameters on throughput-rate performance in bursty traffic mode: (a) β_{accel} is fixed at 0.5 and $\alpha_{\mathrm{accel}} = 0.9$, 0.7, and 0.1; (b) α_{accel} is fixed at 0.5 and $\beta_{\mathrm{accel}} = 0.8$, 0.6, and 0.2

responsiveness during bursty traffic scenarios while maintaining operational efficiency under steady-state conditions, the optimal values for both parameters γ and δ are set to 0.5. As shown in Figs. 8 and 9, under bursty traffic mode, when parameters α and β in the TQDA are increased, the CIOQ throughput gradually increases while latency correspondingly decreases. However, excessively large values cause the VOQ to throttle too aggressively, halting further throughput growth and even inducing a delay increase. Therefore, under the experimental conditions of this study, the optimal values for α and β are 0.5 and 0.4, respectively. Similarly, Figs. 10 and 11 reveal analogous behavior for parameters α_{accel} and β_{accel} in the NTAA, yielding optimal values of 0.7 and 0.6. In summary, the optimal parameter combination for the VDRA designed in this study is $\gamma = 0.5$, $\delta = 0.5$, $\alpha = 0.5$, $\beta = 0.4$, $\alpha_{\mathrm{accel}} = 0.7$, and $\beta_{\mathrm{accel}} = 0.6$. This parameter set demonstrates cross-scenario consistency, enabling robust performance across various typical and extreme traffic conditions.

4.2 Performance simulation comparison of switching architectures

Internet data streams have self-similar characteristics that are significantly different from traditional telecommunication network data traffic, and their data usually arrives at the network device end in bursts of clusters. To fully evaluate the practicality and advantages of the design in this study, we have selected several types of crossbar switching architectures that are common in the industry for performance simulation comparison. In addition, to fully simulate the complex network switching scenarios, we choose the uniform traffic model, non-uniform distributed traffic (NUDT) model, and bursty traffic model to simulate various scenarios in real applications, respectively. Meanwhile, we also choose the hot-spot model, which is used to simulate typical scenarios in HCI and data centers where multiple heterogeneous service devices access a single storage device at the same time. By comparing and analyzing the throughput and latency under different load rates, the processing capability of the switching architecture can be explored as it approaches its limits. Fig. 12 illustrates the latency simulation results of different types of switching architectures under different traffic models, and Fig. 13 exhibits the simulation results of throughput rates.

As shown in Fig. 12, the switching architecture design proposed in this study has significant advantages in terms of latency performance. As the load rate gradually increases, the average switching latency in all types of crossbar switching architectures shows an increasing trend. But under four typical

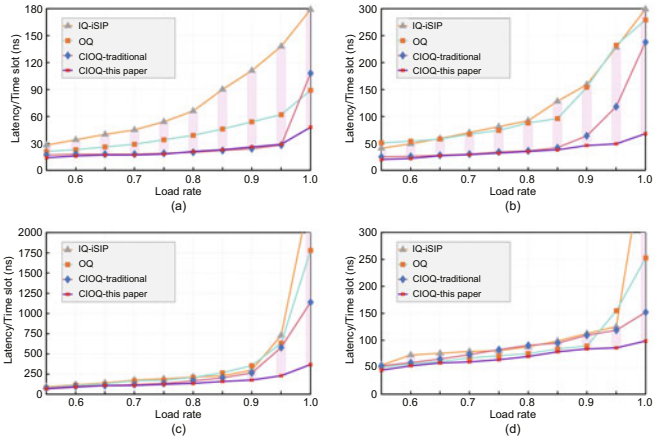

Fig. 12 Delay simulation curves for different switching architectures with different traffic models: (a) uniform traffic mode; (b) NUDT mode; (c) bursty traffic mode; (d) hot-spot traffic mode

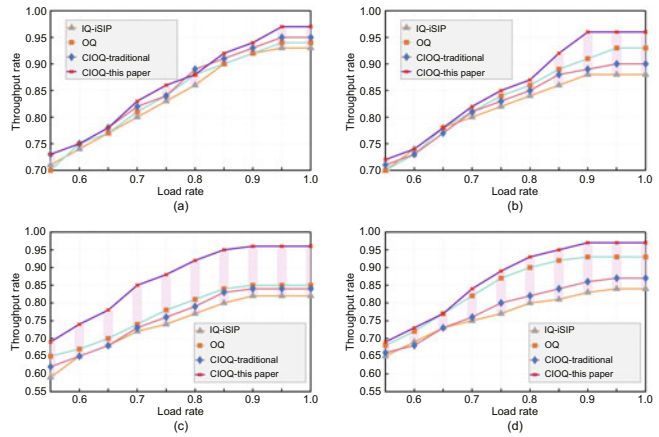

Fig. 13 Throughput-rate simulation curves for different switching architectures under different traffic models: (a) uniform traffic mode; (b) NUDT mode; (c) bursty traffic mode; (d) hot-spot traffic mode

traffic patterns, the design in this study consistently exhibits the lowest latency performance, which demonstrates its robustness under high-load conditions. Moreover, in the uniform traffic mode and NUDT mode, the variability of packet arrivals is mainly reflected in the inter-port rate distribution, and the overall input traffic still maintains a relatively uniform distribution across input ports, so the impact on the switching architecture is relatively smooth. In these traffic environments, the difference in latency performance between different switching architectures is small, and they can all maintain good forwarding efficiency. However, in bursty traffic mode and hot-spot traffic mode, the number of packets at different ports for a short period of time is significantly higher than the average traffic level, and the packets arriving at the ports pile up in clusters. These local or global bursty forwarding demands put forward higher requirements on the processing capability and stability of the switching architecture, and the performance of different types of switching architectures varies greatly in this scenario, especially for the current widely used IQ-type switching architectures, whose latency performance deteriorates dramatically in this scenario and is much higher than the latency level designed in this study. This fully reflects the superiority of the proposed algorithm in coping with bursty traffic. Additionally, the throughput rate simulation results shown in Fig. 13 are generally consistent with the trend of latency performance.

The results show that the design of this study has better throughput than other types of switching architectures under various traffic models, especially in bursty traffic and hotspot traffic scenarios, which show more significant performance advantages. In summary, the simulation results fully verify that the design of this study has a significant improvement over the traditional switching architectures, IQ, OQ, and CIOQ switching architectures, in terms of the two key performance indicators, latency and throughput. It not only enhances the stability of the system under high load conditions but also improves its adaptability under multiple complex network environments.

4.3 UVM simulation and hardware encoding testing

Most of the existing studies on crossbar are based on $N \times N$ symmetric cross switches. In contrast, asymmetric crossbar (ASC) has fewer practical applications due to the high design difficulty and layout and wiring difficulties. However, in an asymmetric architecture, the number of input ports M and the number of output ports N need not be equal. It has been demonstrated that when the ratio of N to M is larger, it rather contributes to the throughput of the switching architecture (Luo et al., 2024). In the development of a self-developed PCIe Gen4 switch chip with a 3×12 port configuration, the design described in this study is coded and implemented. To further verify the performance of the design in this study in terms of performance metrics, we build a UVM simulation and verification platform based on the universal verification methodology to verify the minimum forwarding latency of CIOQ. Meanwhile, after Verilog hardware coding of the design in this study, we launch a real-world comparison test of throughput and latency in the form of field-programmable gate array (FPGA) prototyping. Furthermore, to explore the overhead of different designs at the hardware circuit level, we analyze the consumption of different hardware resources in a statistical comparison.

4.3.1 UVM simulation verification

To investigate the effectiveness of the design in this study in terms of latency performance improvement, we have specifically developed targeted test cases based on the UVM verification methodology, so as to validate the minimum forwarding latency of the CIOQ architecture designed in this study; that is, in the test case, it is guaranteed that the packet arrives at the output port with sufficient credit and is forwarded immediately without waiting and decelerating, thus providing an effective comparison benchmark for the actual latency performance tests subsequently conducted on the FPGA platform. During the testing process, it is compiled and simulated by the Verilog compiler simulator (VCS) simulation tool, and analyzed by Verdi software to generate waveform diagrams. Fig. 14 shows the simulation results under the conditions of PCIe Gen4 mode, 8X Lane, and 800 Hz clock frequency.

At a clock frequency of 800 Hz, the time of each clock cycle is 1.25 ns. According to the simulation waveform analysis results, a single packet occupies a total of 22 clock cycles from entering the switching architecture to completing the forwarding, so its minimum forwarding latency is only 22×1.25=27.5 ns. This result shows that, under ideal conditions, the switching architecture proposed in this study has extremely low forwarding latency.

4.3.2 FPGA performance testing and analysis

In terms of hardware implementation, after coding the Verilog hardware circuit design for the IQ switching

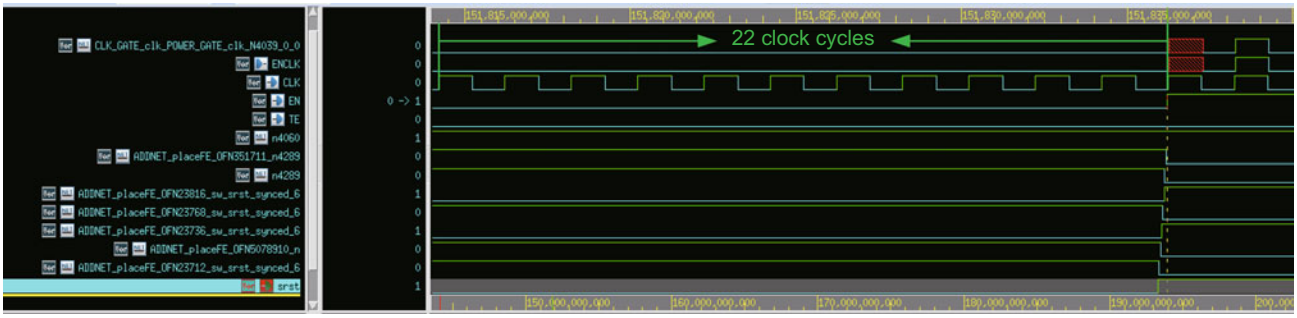

Fig. 14 Simulated waveforms with minimum switching forwarding delay

architecture and the design in this study separately, we introduce the PCIe Gen4 switching chip PEX8748 designed by PLX (now acquired by Broadcom) as a comparative test benchmark for throughput and latency, and the switch architecture type used in the PEX8748 is the traditional CIOQ switch architecture (Zheng R et al., 2026). Under the condition of PCIe Gen4 mode and 8X Lane at both the sender and receiver, we build the corresponding topology environment, and then test the throughput and forwarding latency of each switching architecture when traversing different packet lengths. It is worth mentioning that the frequency of DLLP transmission serves as a key factor affecting the throughput performance. The more frequent the DLLP transmission, the lower the percentage of payload, which in turn leads to a decrease in the throughput performance of the whole system. Therefore, the impact of DLLP is analyzed with a focus on the testing process. Fig. 15 illustrates the test results for throughput.

As can be seen in Fig. 15a, as the packet length increases, the throughput of the three switching architectures stabilizes after a near-linear increase. Among them, under different packet length conditions, the CIOQ switching architecture designed in this study possesses significantly higher throughput than that of the traditional CIOQ and IQ, which can be fully adapted to complex scenarios with higher performance requirements. Moreover, the actual true percentage of the total throughput can be calculated from Eq. (20):

$$\text{Throughput} = \frac{\dfrac{\text{TLP}}{\text{DLLP}_{\text{num}}}\left(\text{TLP}_{\text{lth}} - 8\right)}{\dfrac{\text{TLP}}{\text{DLLP}_{\text{num}}}\text{TLP}_{\text{lth}} + \text{DLLP}_{\text{lth}}}. \tag{20}$$

In Eq. (20), TLP and TLP_{lth} represent the number and length of transaction layer packets, respectively, while DLLP_{num} and DLLP_{lth} represent the number and length of DLLPs, respectively.

The higher the percentage, the higher the effective throughput of the switching architecture. According to Fig. 15b, it can be seen that the DLLP traffic of the switching architecture designed in this study is always lower than that of the other two types of architectures during the testing process. With the gradual increase of the packet length from 8 to 512 DW (DW refers to data word), the DLLP throughput of the switching architecture designed in this study decreases faster, which further enhances the percentage of the payload in the total throughput, and thus significantly optimizing the actual transmission efficiency of the system. Table 3 illustrates the test results of the design in this study for throughput at different packet lengths.

Combining the data in Fig. 15 and Table 3, it can be seen that the CIOQ switching architecture designed in this study significantly outperforms the commercial chip PEX8748 (based on the traditional CIOQ architecture) and the IQ architecture in terms of maximum throughput. The three peak maximum throughputs are 1499.66, 1301.91, and 1243.25 Gb/s, respectively. Compared to the PEX8748 switching architecture and IQ, the maximum throughput improvement achieved by this design is 15.12% and 20.55%, respectively. When the payload is 512 DW, its maximum throughput is as high as 1499.66 Gb/s, the total throughput percentage is as high as 96.94%, and the DLLP throughput percentage is only 0.61%, which does not bring too much flow control overhead due to the introduction of the AFC mechanism. In addition, to compare the magnitude of latency performance improvement, we conduct a comparison test of the total point-to-point latency under different packet lengths, and the results are shown in Fig. 16.

Fig. 16 illustrates that as the packet length increases from 1 to 512 DW, the switching latency for all three types of switching architectures shows an upward trend. But the design in this study has lower forwarding latency than the remaining two switching architectures for different packet lengths. At a packet length of 512 DW, the minimum forwarding delay required for this design is only 83 ns, and the peak switching latency for the design in this study, PEX8748, and IQ is 189, 206, and 225 ns, respectively. The maximum reduction in forwarding latency is about 26.9% and 54.7% (at a packet length of 64 DW) compared to PEX8748 and IQ. Compared to the minimum simulation latency in Section 4.3.1, the latency of the different switching architectures is somewhat higher, which is due to the additional flow control overhead imposed by the corresponding mechanisms. In addition, we test the packet loss rate of the design in this study, and its average packet loss rate is only 0.6%, indicating that the design can still maintain high data transmission integrity under high load conditions. After code logic synthesis, we compare the FPGA resource overhead for different types of switching architectures, as shown in Table 4.

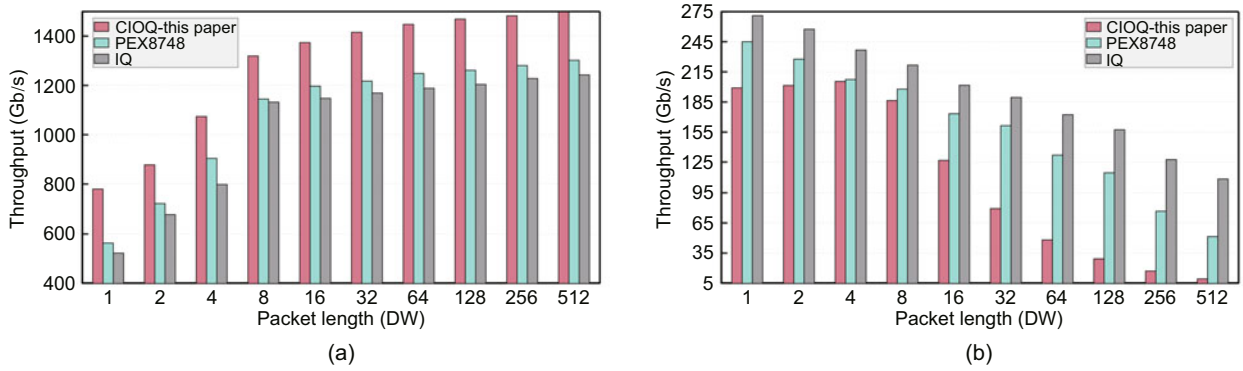

Fig. 15 Throughput comparison when traversing packet length with different switching architectures: (a) total throughput at different packet lengths; (b) DLLP throughput at different packet lengths

Table 3 Throughput test results statistics under the design of this study

Packet length (DW)	TLP maximum throughput (Gb/s)	DLLP maximum throughput (Gb/s)	Total throughput (Gb/s)	Percentage of TLP (%)	Percentage of DLLP (%)	Percentage of total throughput (%)
1	480.13	199.09	679.22	31.26	12.96	44.22
2	576.16	201.45	777.61	37.51	13.12	50.63
4	768.21	205.57	973.78	50.01	13.38	63.39
8	1133.01	186.42	1319.43	73.76	12.14	85.90
16	1247.34	126.77	1374.11	81.21	8.25	89.46
32	1336.89	79.31	1416.20	87.04	5.16	92.20
64	1399.77	48.03	1447.80	91.13	3.13	94.26
128	1439.93	29.18	1469.11	93.75	1.90	95.65
256	1465.90	16.94	1482.84	95.44	1.10	96.54
512	1490.40	9.26	1499.66	96.33	0.61	96.94

Table 4 FPGA resource overhead and percentage

Architecture	LUT (Instance)	FF (Instance)	LUTRAM (Instance)	BRAM (Instance)
This study	1 111 242 (28.43%)	972 213 (7.61%)	11 612 (1.97%)	894.42 (43.01%)
Traditional CIOQ	984 493 (24.09%)	664 789 (7.53%)	10 313 (1.75%)	906.40 (43.57%)
IQ	997 688 (24.42%)	612 454 (6.93%)	19 768 (3.35%)	912.34 (43.87%)

LUT: look-up table; FF: flip-flop; LUTRAM: LUT-based random access memory; BRAM: block random access memory

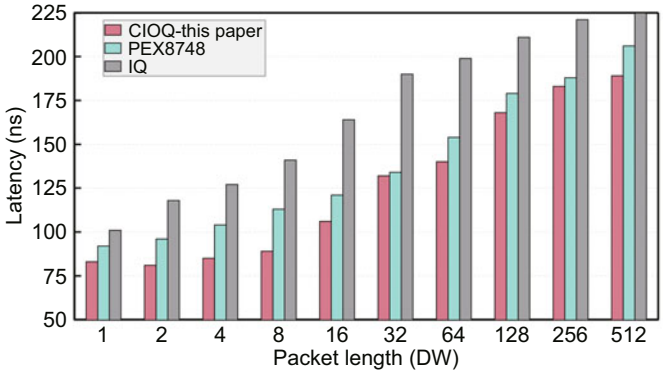

Fig. 16 Comparison of the delay of different switching architectures at different packet lengths

Table 4 illustrates that the design based on this study has a higher logical resource overhead for the same switch size (3×12) and the same port performance. The reason for this is that the design of the AFC in this study mostly relies on hardware implementation, while introducing higher design complexity in shared-buffer management. However, in terms of buffer utilization, due to the adoption of an efficient buffer allocation and release mechanism, the utilization of buffer resources is significantly improved, resulting in a smaller buffer resource overhead, which is actually required under the same buffer capacity requirements. Furthermore, as the core module of the switch chip, the switch architecture plays a crucial role in power consumption estimation prior to the chip entering the manufacturing process. The comprehensive evaluation of the number of instance_count is 2 857 762, the memory area is 1.749 775 mm^2, and the module unit area is 6.605 771 mm^2 under the SMIC 28 nm process, which is about 11.18% of the entire chip area after the design for testability (DFT) and place and route (PR). In terms of power consumption, its power consumption is 11 495 mW, which is about 95.82% of the power consumption of the whole chip. Compared to PEX8748 and traditional CIOQ, it occupies a larger proportion of the chip area and power consumption, which leads to an increase in the difficulty of the back-end layout and wiring, but this cost stems from the performance gain brought by the AFC mechanism introduced to enhance the switching efficiency.

4.4 HCI scenario performance comparison

In practice, network traffic is often not of a single type, but a combination of characteristics. HCI heterogeneous domain cross-domain interconnection, as a typical application scenario of bursty traffic, not only has the basic data characteristics of the bursty traffic pattern, but also has the local bursting characteristics of the hotspot traffic pattern. At the same time, when using the switching architecture for cross-domain interactions, the amount of data generated in a short period of time during the master-standby inversion increases dramatically, creating a small packet-intensive switching scenario. At the same time, when distributed heterogeneous resources in various domains interact, they usually involve large-scale bulk data transfers, resulting in a significant increase in the payload of a single or multiple packets. Moreover, during cross-domain transmission, multiple heterogeneous domains may read and write to the same local solid state drive (SSD) storage resources, thus generating the need for packet switching under hotspot traffic. This type of scenario is extremely common in HCI and data centers, and also puts forward higher requirements for the performance stability of the switching architecture. From the analysis in Section 4.2, it can be seen that in bursty traffic and hotspot traffic modes, the throughput and latency performance of this study is better than that of the rest of the switching architectures. As can be seen from Section 4.3.2, for a certain fixed packet length, the CIOQ under the design of this study performs better than that of the rest of the switching architectures as well. Therefore, on this basis, to fully test the comprehensive performance in real and covering scenarios of bursty traffic, uniform traffic, hotspot traffic, and NUDT patterns, we build a hyper-converged environment as shown in Fig. 17. The environment is based on the X86 architecture of the Intel processor, the ARM architecture of the Phytium processor, and the Longxin processor to build three heterogeneous domains (Zheng CM et al., 2022), constituting an HCI heterogeneous interconnect system containing three host domains. Each host domain has a different model and

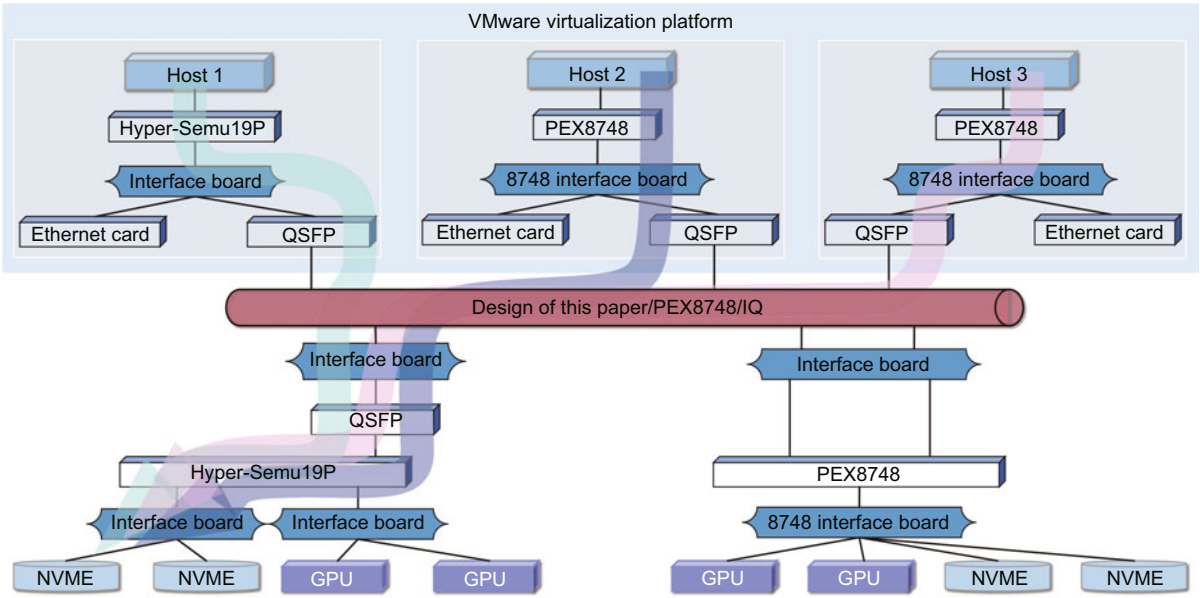

Fig. 17 Deployment diagram of HCI heterogeneous interconnect stress testing scenarios. QSFP: quad small form-factor pluggable; NVME: non-volatile memory express

independent SSD disks, Ethernet cards, and graphics processing unit (GPU) devices.

In the constructed HCI heterogeneous interconnection system, the central switching structure is replaced by the design in this study, PEX8748, and IQ structure to realize the horizontal comparison of the performance of different switching structures (Chen C et al., 2024). Among them, both the design of this study and the IQ-type switching architecture are carried by the Hyper-Semu 19p-FPGA board after encoding. In the case of mixing different traffic loads in multiple host domains, a master-standby inversion operation is carried out every 2 h, and different heterogeneous domains maintain high load communication under the complex communication scenario of taking over each other's equipment, thus simulating the high concurrency and high load communication demand during equipment failure switching or planned maintenance in real applications. To comprehensively evaluate the stability performance of different types of switching architectures under long-time operation, the Perf performance testing tool is used to conduct long-time stress tests on the switching architectures under the system, and the performance stability tests of different types of switching architectures are fully carried out under different load pressures. Figs. 18 and 19 show the results of the latency and throughput stability stress tests, respectively.

In terms of latency, it is shown by Fig. 18 that under high load impact, the latency stability of the PEX8748 as well as the IQ-type switching architecture is much lower than that of the design in this study. Under the long time high traffic impact of payload changing from 256 to 2048 bytes, the minimum switching latency of the design in this study, PEX8748, and IQ is 112, 129, and 165 ns, respectively. Compared to PEX8748 and IQ, the latency of the design in this study is reduced by 15.18% and 47.32%, respectively, demonstrating better response efficiency. Additionally, as the length of the stress testing time increases, the packet traffic is constantly superimposed, the number of pending packets backlogged inside the switching architecture continues to accumulate, resulting in more complex packet forwarding work in the switching architecture, and the switching latency of different switching architectures exhibits an upward trend. However, the PEX8748 and IQ switching architectures have much larger increasing slopes and their performance deteriorates rapidly with running time. At a payload of 2048 bytes, when the stress testing time is extended from 8 to 48 h, the latency differences of the three switching architectures are 60, 88, and 125 ns, respectively. It is well shown that the design in this study has strong stability in terms of latency performance, so that it can cope with all kinds of complex switching environments. In terms of throughput, it can be seen from Fig. 19 that, similar to the latency performance stability, the design in this study presents a more stable throughput performance compared to the remaining two types of switching architecture. Specifically, under different load pressures and stress testing time, the throughput performance of the design in this study is higher than that of the remaining two switching architectures, where the maximum throughput of the design in this study is 1397.22 Gb/s at a payload of 2048 bytes, which is higher than that of PEX8748 (1286.03 Gb/s) and IQ (1234.29 Gb/s), with an enhancement of about 8.6% and 13.2%, respectively. Meanwhile, as the payload and stress time increase, the throughput under the design of this study decreases much less than the remaining two types of switching architectures. With a payload of 2048 bytes, when the stress testing time is extended from 8 to 48 h, the throughput drops only 38.45 Gb/s, indicating that the design in this study can provide stable throughput with strong practicality and feasibility under the impact of complex mixed traffic.

In summary, in the simulations with various typical traffic models, the CIOQ switching architecture under the design of this study shows better performance than that of the IQ, traditional CIOQ, and OQ switching architectures in terms of latency control and throughput capability. Furthermore, FPGA tests after hardware coding implementation show that,

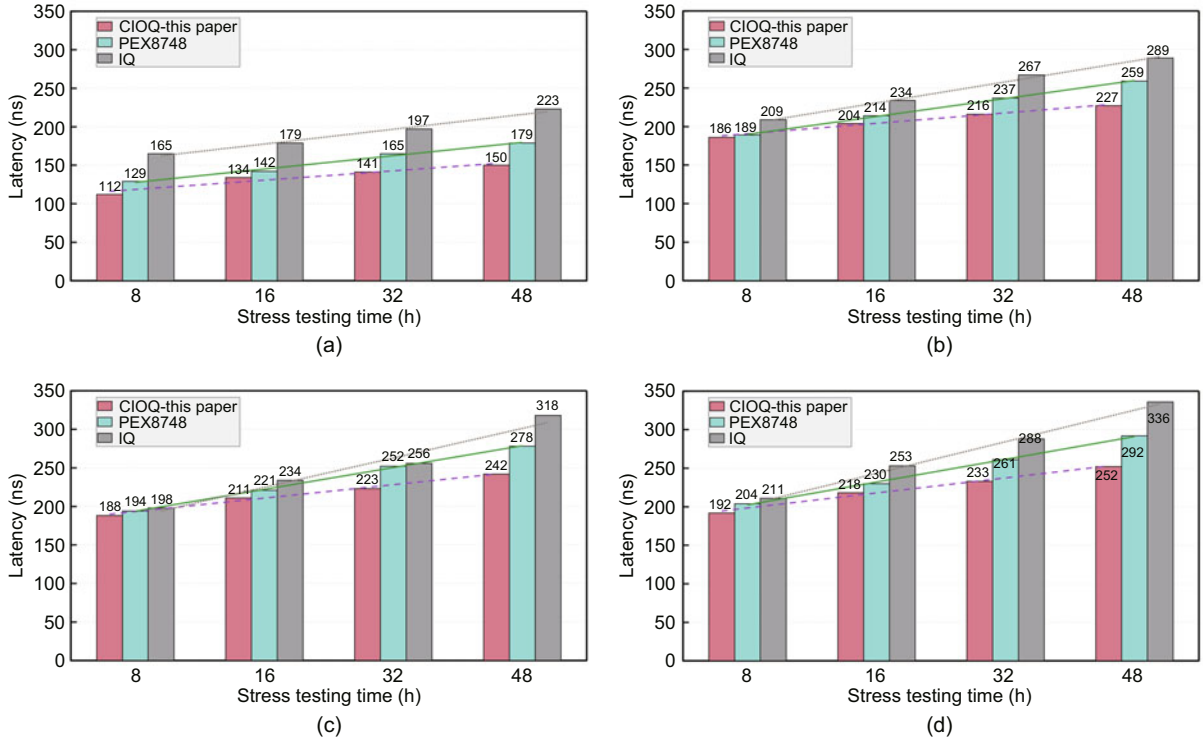

Fig. 18 Stability test results for switching delay performance: (a) max payload size = 256 bytes; (b) max payload size = 512 bytes; (c) max payload size = 1024 bytes; (d) max payload size = 2048 bytes

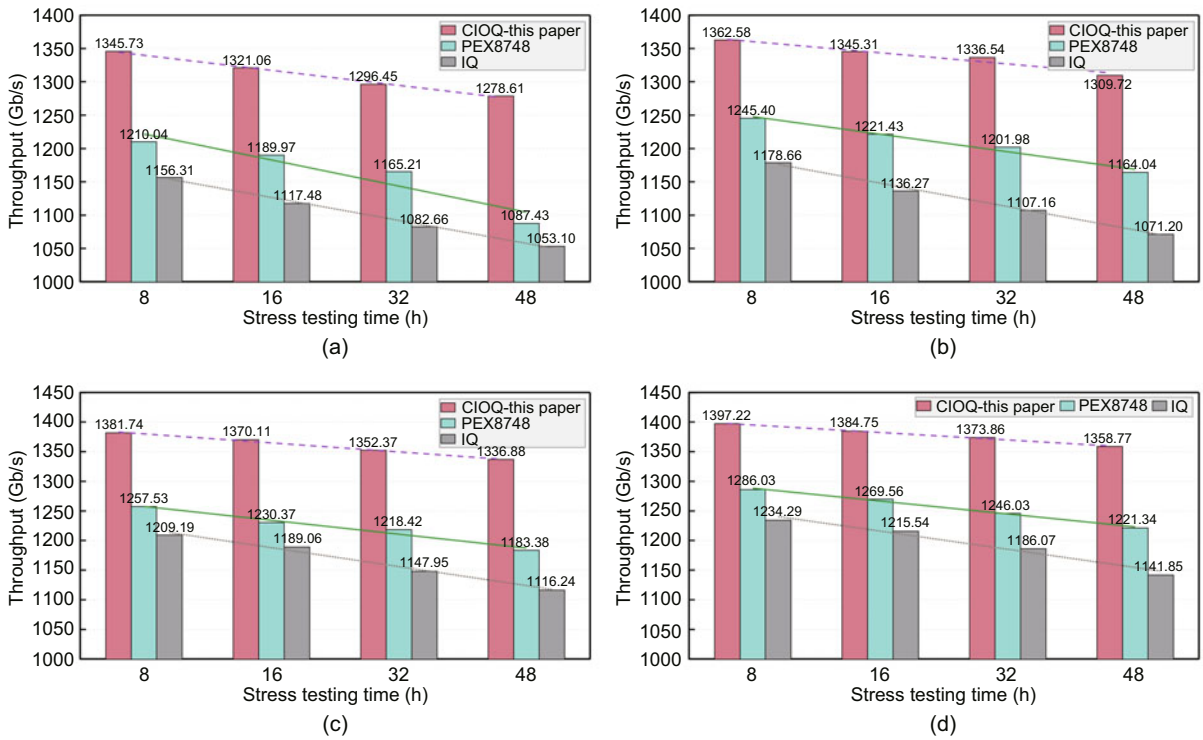

Fig. 19 Throughput performance stability test results: (a) max payload size = 256 bytes; (b) max payload size = 512 bytes; (c) max payload size = 1024 bytes; (d) max payload size = 2048 bytes

consistent with simulation findings, the design in this study provides enhanced throughput and latency performance. In addition, in the test of HCI heterogeneous interconnection scenarios constructed in this study, under complex scenarios such as multi-domain collaboration and master-standby reversal, its performance has stronger stability, and it can satisfy the needs of packet forwarding in all kinds of scenarios, thus possessing stronger scenario adaptability. Specific comparative findings are summarized in Table 5.

Table 5 Conclusions from the comparison of various indicators

Architecture	Maximum throughput (Gb/s)	Percentage of maximum throughput (%)	Min/Max latency (ns)	Maximum throughput in HCI (Gb/s)	Min/Max latency in HCI (ns)	Throughput performance stability	Switching latency performance stability	Hardware resource overhead
This study	1499.66 (512 DW)	96.94 (512 DW)	83/189	1397.22 (2048 bytes)	112/252 (256/2048 bytes)	High	High	High
IQ	1243.25 (512 DW)	78.15 (512 DW)	101/225	1234.29 (2048 bytes)	165/336 (256/2048 bytes)	Low	Low	Low
Traditional CIOQ	1301.91 (512 DW)	81.64 (512 DW)	92/206	1286.03 (2048 bytes)	129/292 (256/2048 bytes)	Middle	Middle	Middle

5 Conclusions

To solve the problems of HOL blocking, buffer overflow, and congestion spreading at the VOQ level in the shared-buffer CIOQ switching architecture, and at the same time, to improve its performance and stability in bursty traffic scenarios, this study proposes a de-blocking AFC design based on the credit flow control mechanism. First, by introducing the CTDM, the input port is given the ability to sense the risk of HOL blocking generation in advance, thus realizing the proactive prevention of HOL blocking. Second, to further enhance the adaptability of CTDM in different traffic scenarios, this study proposes the TDAA, which takes into account the flexibility of the system while effectively controlling the extra overhead. Finally, to comprehensively improve the overall performance and operational stability of the CIOQ switching architecture, this study designs the VDRA, which realizes the fine regulation of the sending rate of each VOQ. Simulation and actual test results show that the design in this study has significant improvement in latency and throughput performance compared with traditional CIOQ, IQ, and OQ switching architecture, and exhibits stronger performance stability in complex traffic scenarios such as HCI, and the experimental results are consistent with the theoretical analysis.

However, the above performance improvement is accompanied by a certain hardware resource overhead. Therefore, the subsequent research will focus on the continuous optimization of the hardware implementation of the design scheme in this study, by decoupling the design of functional modules, and reducing the resource consumption as much as possible under the premise of guaranteeing the performance. Furthermore, the design concept of this study has good portability and can be further extended to other types of switching architectures. In the future, we will explore the application potential and performance of this design concept in different architectures.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2022YFB4500900).

Author contributions

Rui ZHENG was responsible for the experimental design, participated in data analysis and interpretation of results, and drafted the paper. Jianliang SHEN and Fan ZHANG constructed the theoretical model and conducted a comprehensive review of the literature. Peijie LI conducted the experiments and collected the data. Ping LV organized the theoretical knowledge into mathematical formulas. Yu SHAO drafted and revised the paper. Zhengbin ZHU was responsible for collecting and organizing preliminary research materials. Rui ZHENG finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declaration on the use of generative AI tools

During the preparation of this work, the authors used ChatGPT to improve language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

Bandara S, Sanaullah A, Tahir Z, et al., 2024. Performance evaluation of VirtIO device drivers for host-FPGA PCIe communication. IEEE Int Parallel and Distributed Processing Symp Workshops, p.169-176. https://doi.org/10.1109/IPDPSW63119.2024.00043

Chen BW, Zhou WB, 2025. A review of input-queued scheduling algorithms in high-speed switching systems. *Microelectr Comput*, 42(5):1-8 (in Chinese). https://doi.org/10.19304/J.ISSN1000-7180.2024.0301

Chen C, Li H, Zhou L, 2024. A design of radar high-speed data storage module based on VPX. 4th Int Conf on Computer Science Electronic Information Engineering and Intelligent Control Technology, p.234-239. https://doi.org/10.1109/CEI63587.2024.10871594

Dagli I, Belviranli ME, 2024. Shared memory-contention-aware concurrent DNN execution for diversely heterogeneous system-on-chips. Proc 29th ACM SIGPLAN Annual Symp on Principles and Practice of Parallel Programming, p.243-256. https://doi.org/10.1145/3627535.3638502

de la Rosa MS, Gomez-Lopez G, Andújar FJ, et al., 2025. Quality-of-service provision for BXIv3-based interconnection networks. *J Supercomput*, 81(4):601. https://doi.org/10.1007/S11227-025-07069-1

Dong CL, Shen JL, Li PJ, et al., 2024. Multi-protocol switching circuit for software defined interconnection system. *J Commun*, 45(5):44-53 (in Chinese). https://doi.org/10.11959/j.issn.1000-436x.2024076

Firoozshahian A, Manshadi V, Goel A, et al., 2007. Efficient, fully local algorithms for CIOQ switches. 26th IEEE Int Conf on Computer Communications, p.2491-2495. https://doi.org/10.1109/INFCOM.2007.307

Hou WT, Zhang J, Wang ZK, et al., 2024. Understanding routable PCIe performance for composable infrastructures. 21st USENIX Symp on Networked Systems Design and Implementation, p.297-312.

Hu B, Fan FJ, Yeung KL, et al., 2018. Highest rank first: a new class of single-iteration scheduling algorithms for input-queued switches. *IEEE Access*, 6:11046-11062. https://doi.org/10.1109/ACCESS.2018.2800686

Hu JB, Huang JW, Wang JX, et al., 2023. A transmission control mechanism for lossless datacenter network based on direct congestion notification. *Acta Electr Sin*, 51(9):2355-2365 (in Chinese). https://doi.org/10.12263/DZXB.20220491

Kaltenhauser A, Stefanidi E, Schöning J, 2024. Playing with perspectives and unveiling the autoethnographic kaleidoscope in HCI – a literature review of autoethnographies. Proc CHI Conf on Human Factors in Computing Systems, Article 819. https://doi.org/10.1145/3613904.3642355

Kim H, Ryu J, Lee J, 2024. TCCL: discovering better communication paths for PCIe GPU clusters. Proc 29th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems, p.999-1015. https://doi.org/10.1145/3620666.3651362

Luo JF, Yu F, Li WJ, et al., 2024. A novel switch architecture for multi-die optimization with efficient connections. *Electronics*, 13(16):3205. https://doi.org/10.3390/electronics13163205

Mohtavipour SM, Mollajafari M, Naseri A, 2020. A novel packet exchanging strategy for preventing HoL-blocking in fat-trees. *Clust Comput*, 23(2):461-482. https://doi.org/10.1007/s10586-019-02940-2

Nag SN, 2023. Technical analysis of PCIe to PCIe 6: a next-generation interface evolution. *World J Eng Technol*, 11(3):504-525. https://doi.org/10.4236/wjet.2023.113036

Ouyang YM, Yang JF, Xing K, et al., 2018. An improved communication scheme for non-HOL-blocking wireless NoC. *Integration*, 60:240-247. https://doi.org/10.1016/j.vlsi.2017.10.005

Palnitkar SS, Kanade S, 2024. Q-memory task routing to prevent deadlocks in Ethernet control with memory crossbar switching. *Opt Mem Neur Netw*, 33(1):72-85. https://doi.org/10.3103/S1060992X24010077

Ran C, Su H, Sun Y, et al., 2023. Research on fair scheduling algorithm of high-performance input queuing switch. *Foreign Electr Meas Technol*, 42(2):114-119 (in Chinese). https://doi.org/10.19652/j.cnki.femt.2204512

Ran C, Su H, Sun Y, et al., 2024. Research on improved iSLIP scheduling algorithm based on CIOQ architecture. *Comput Simul*, 41(4):325-329, 417 (in Chinese). https://doi.org/10.3969/j.issn.1006-9348.2024.04.061

Shen ZJ, Gao J, Wu RG, 2018. Feedback and reverse transmission mechanism based two-stage switch architecture. *J Electr Inform Technol*, 40(3):697-704 (in Chinese). https://doi.org/10.11999/JEIT170531

Shen ZJ, Tao DH, Gao J, 2019. Multichannel-feedback-based two-stage switch architecture. *J Front Comput Sci Technol*, 13(9):1516-1523 (in Chinese). https://doi.org/10.3778/j.issn.1673-9418.1812003

Wu WX, Zhang T, Li Z, et al., 2025. Dynamic per-flow queues in shared buffer TSN switches. *ACM Trans Des Autom Electr Syst*, 30(3):38. https://doi.org/10.1145/3718087

Xu W, Dainoff MJ, Ge LZ, et al., 2023. Transitioning to human interaction with AI systems: new challenges and opportunities for HCI professionals to enable human-centered AI. *Int J Hum–Comput Int*, 39(3):494-518. https://doi.org/10.1080/10447318.2022.2041900

Yébenes P, Escudero-Sahuquillo J, García PJ, et al., 2019. Head-of-line blocking avoidance in slim fly networks using deadlock-free non-minimal and adaptive routing. *Concurr Comput*, 31(2):e4441. https://doi.org/10.1002/cpe.4441

Zhang YR, Wang SG, Ren FY, 2025. A review of lossless network traffic management. *J Comput Res Dev*, 62(5):1290-1306 (in Chinese). https://doi.org/10.7544/issn1000-1239.202440096

Zheng CM, Yao XX, Zhou F, et al., 2022. Adaption and implementation of server chipsets for the Loongson CPU. *Chin J Eng*, 44(7):1244-1254 (in Chinese). https:/doi.org/10.13374/j.issn2095-9389.2021.10.08.003

Zheng R, Shen JL, Lv P, et al., 2026. Optimized design of non-transparent bridge for heterogeneous interconnects in hyper-converged infrastructure. *J Electr Inform Technol*, 48(2):567-582 (in Chinese). https://doi.org/10.11999/JEIT250272

Zhou WW, Sheng WX, Yan BY, 2024. A single-chip wafer-level packaged SR-crossbar RF MEMS switch matrix. *IEEE Electr Dev Lett*, 45(7):1309-1312. https://doi.org/10.1109/LED.2024.3403550

Zyla K, Liess M, Wild T, et al., 2024. FlexCross: high-speed and flexible packet processing via a crosspoint-queued crossbar. 27th Euromicro Conf on Digital System Design, p.98-105. https://doi.org/10.1109/DSD64264.2024.00022