electronics

MDPI

Article

FPGA-Based Time-Triggered Ethernet with a Decoupled Control–Data Architecture and a Deadline-Aware Weighted Load-Balancing Traffic Scheduling Method

Yanlong Wang [1], Xiaofu Zou [2,*] and Luning Guan [1]

[1] School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China; by2103053@buaa.edu.cn (Y.W.); 19373468@buaa.edu.cn (L.G.)
[2] School of Artificial Intelligence, Beihang University, Beijing 100191, China
* Correspondence: windyfuzi5311@buaa.edu.cn

Abstract

Time-Triggered Ethernet (TTE, SAE AS6802) ensures deterministic communication for safety-critical distributed systems, but hardware implementation requires tight coordination between clock synchronization and schedule execution. This paper presents an FPGA-based TTE interface featuring a decoupled control–data architecture and an offline deadline-aware weighted load-balancing method. The control plane implements hardware-level AS6802 synchronization functions, while the data plane executes schedule-triggered UDP frame transmissions using a common 125 MHz logical time base (8 ns resolution). To balance channel utilization while guaranteeing timeliness, Earliest Deadline First (EDF) instance ordering is combined with transmission-duration-weighted load balancing over candidate transmission positions. The system was validated on two Xilinx Artix-7 FPGAs interconnected via a direct 1 Gb/s link. Experimental results demonstrate that the PCF timestamp-interval residual remains within ±8.00 ns (mean absolute error of 2.88 ns), while the measured compensated physical clock skew between independent boards is 0.71 ± 3.58 ns with a maximum of 9.60 ns. For the evaluated five-flow TT workload (13 instances per 32 ms matrix cycle), the interface achieves a mean end-to-end hardware latency of 2.45 ms (maximum 3.78 ms), zero packet loss, zero deadline violations, and a 96.97% load-balance rate across the configured logical channels.

Keywords: Time-Triggered Ethernet; FPGA; fault-tolerant clock synchronization; control–data decoupling; deadline-aware load balancing

Check for updates

Academic Editor: Stefano Ricci

Received: 22 September 2026
Revised: 21 October 2026
Accepted: 23 October 2026
Published: 26 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Industrial control, distributed sensing, intelligent transportation, and autonomous systems increasingly share unified Ethernet infrastructures to transmit real-time measurements, safety-critical control commands, synchronization frames, and background traffic [1–3]. These heterogeneous traffic classes exhibit fundamentally distinct temporal requirements. Although best-effort (BE) traffic primarily seeks high throughput, control and safety-critical traffic must be delivered strictly before explicit deadlines. Therefore, average latency alone is insufficient to characterize deterministic communication; frame release times, queue forwarding orders, resource occupation, and end-to-end latency variations must be strictly governed [4,5].

Time-Triggered Ethernet (TTE), standardized by SAE AS6802, enforces deterministic communication by establishing a synchronized network-wide clock base where nodes trans-

mit critical time-triggered (TT) frames at predefined slots, coexisting with rate-constrained (RC) and best-effort traffic [4,5]. However, achieving guaranteed determinism in practical implementations faces two coupled challenges.

First, on the hardware interface level, a global time base alone does not guarantee schedule adherence. If protocol control frame (PCF) processing, local clock correction, and transmit buffering introduce unmodeled processing latencies, a mathematically feasible schedule cannot be reliably reproduced by the hardware data path. Existing designs frequently separate synchronization state management from scheduled data transmission, leading to timing variations at slot boundaries. Second, on the scheduling level, Earliest Deadline First (EDF) ordering prioritizes deadline urgency, but its classic optimality does not directly apply to non-preemptive, multi-channel Ethernet environments [6,7]. Pure EDF ordering often concentrates traffic on specific routes, while conventional load-balancing heuristics typically balance raw frame counts without accounting for disparate frame transmission durations. This can cause channel congestion and unexpected deadline violations.

Although significant efforts have addressed static TTE/TSN scheduling [8–10], high-precision clock synchronization [11–13], and hardware-assisted communication controllers [14–16], two key aspects remain insufficiently addressed in an integrated manner: the hardware coordination between synchronization control and scheduled transmission within an integrated FPGA interface, and a network scheduling strategy that balances channel occupation while strictly preserving deadline urgency.

To address these challenges, this paper presents an FPGA-based Time-Triggered Ethernet interface architecture alongside an offline deadline-aware weighted load-balancing scheduling method. The primary contributions of this work are twofold.

First, this paper develops a control–data plane decoupled FPGA architecture for TTE synchronization and scheduled transmission. Protocol control frame processing, synchronization state control, local clock correction, and IP/UDP time-triggered transmission are implemented as coordinated hardware functions sharing a common local time representation.

Second, this paper proposes a weighted load-balancing method combined with Earliest Deadline First ordering. Earliest Deadline First preserves deadline priority, while accumulated transmission duration is used to evaluate channel occupation. Candidate transmission positions are constrained by deadlines, release times, protocol windows, guard intervals, and resource exclusivity before schedule execution.

The remainder of this paper is organized as follows: Section 2 analyzes related works in deterministic Ethernet scheduling and synchronization implementations. Section 3 details the proposed FPGA-based TTE architecture. Section 4 presents the deadline-aware weighted load-balancing scheduling method. Section 5 discusses the experimental setup, hardware results, comparative evaluations, and scalability analyses. Section 6 concludes the paper.

2. Analysis of Related Works

2.1. Deterministic Ethernet and Time-Triggered Communication

Conventional Ethernet provides flexible connectivity and efficient statistical multiplexing, but its best-effort forwarding mechanism does not inherently bound queueing delay or guarantee the completion time of critical traffic. Deterministic Networking addresses this problem by constraining resource allocation, forwarding behavior, and service protection across a network domain [1]. TSN extends Ethernet with time synchronization, scheduled transmission, traffic shaping, frame preemption, and reliability mechanisms [2–4]. These functions allow critical and non-critical traffic to share a physical network while receiving different temporal guarantees.

IEEE 802.1AS establishes a common time base for time-sensitive applications, whereas IEEE 802.1Qbv controls the transmission opportunity of queues through a Gate Control List (GCL) [2,3]. A GCL specifies the opening and closing state of traffic queues at predefined instants. Its effectiveness depends on clock synchronization accuracy, frame transmission duration, guard intervals, and the treatment of frames already in transmission when a protected interval begins. Related work has also examined flexible slot boundaries and mixed-triggered communication to improve resource utilization while preserving deterministic behavior [10].

TTE follows a closely related time-triggered principle. Under SAE AS6802, critical frames are transmitted at predefined points according to a globally consistent time base [5]. TTE supports different traffic classes and provides synchronization services based on protocol control frames. Its static transmission segment can isolate critical traffic from RC and BE traffic, but the resulting determinism depends on both the correctness of the schedule and the implementation of synchronization and frame processing functions.

TTE has been studied from architectural, analytical, and implementation perspectives. Kopetz et al. introduced the design principles of TTEthernet as a distributed real-time communication system [14]. The TTEthernet dataflow concept further described the relation between synchronized communication and data transmission [15]. Hardware implementations of TTE switches and interfaces have investigated the realization of synchronization services, traffic classification, and deterministic forwarding in Ethernet devices [16]. These works establish the architectural basis for TTE, but they do not prescribe a single FPGA organization for combining synchronization logic, conventional Ethernet protocols, and application-level TT traffic.

The coexistence of different traffic classes introduces additional scheduling constraints. Rate-constrained traffic can be analyzed using the extended trajectory approach, while TT traffic is transmitted according to a static schedule and best-effort traffic receives no strict timing guarantee [17]. A deterministic interface must therefore reserve sufficient time for synchronization and TT traffic while allowing remaining bandwidth to serve lower priority traffic. The design used in this paper adopts separated PCF, TT, and RC/BE windows to simplify hardware execution and experimental verification.

2.2. Clock Synchronization and Hardware Timing

Clock synchronization is a prerequisite for a distributed schedule. IEEE 1588 provides a general precision clock synchronization protocol for networked measurement and control systems [18]. IEEE 802.1AS adapts synchronization procedures to TSN environments and provides the time reference required by scheduled transmission [2]. The achievable accuracy depends on timestamp position, oscillator stability, link asymmetry, residence time, delay estimation, and hardware support.

The influence of clock synchronization on TTE real-time performance has been investigated through node and switch models. Synchronization error affects the relationship between a message transmission point and its reception window; when the accumulated timing error exceeds the available margin, packet loss or deadline violation may occur [19]. This observation indicates that synchronization accuracy must be evaluated together with schedule window width and transmission delay rather than as an isolated metric.

Dedicated hardware timing logic can reduce the uncertainty caused by software interrupts, operating system scheduling, and variable buffer processing. High-precision synchronization chips for industrial sensor and actuator networks demonstrate the importance of hardware timestamping and deterministic timing paths [11]. Studies of wireless fieldbus and industrial sensor networks have also emphasized the relationship between synchronization precision, communication delay, and application-level control perfor-

mance [12]. In virtualized distributed real-time systems, the organization of the clock source and resource sharing mechanism similarly affects the synchronization bound [13].

TTE coordinates synchronization through Synchronization Master (SM), Compression Master (CM), and Synchronization Client (SC) roles [5,19]. Figure 1 shows the Ethernet encapsulation and field layout of the PCF, schematically redrawn by the authors based on the SAE AS6802 standard specification [5]. The Ethernet EtherType 0x891D identifies PCF traffic, whereas the internal Type field distinguishes PCF categories. Integration cycle, membership, priority, domain, and transparent clock fields support synchronization state management and timing correction. Their hardware processing connects received synchronization information to local clock maintenance.

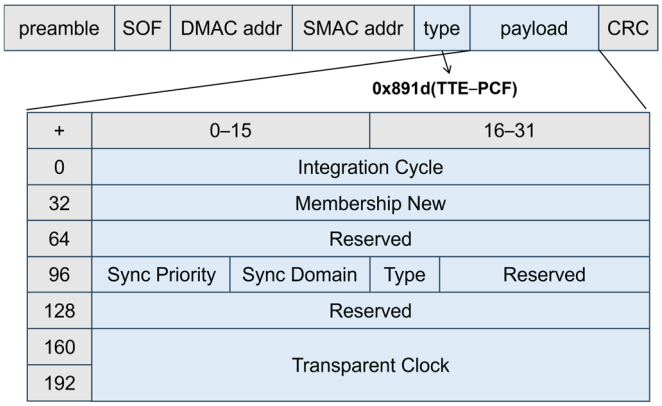


Figure 1. Ethernet encapsulation and field layout of the protocol control frame.

Figure 2 illustrates the SM–CM frame exchanges during synchronization startup and maintenance, schematically redrawn by the authors according to the AS6802 protocol state definitions [5,19]. Cold Start (CS) and Cold-Start Acknowledgment (CA) frames participate in startup, followed by periodic Integration (IN) frames during synchronized operation. The diagram summarizes these exchanges; state transitions additionally depend on membership and timeout conditions. Hardware evaluation must distinguish roles sharing one oscillator from nodes using independent clocks, since only the latter configuration evaluates synchronization between independent clock sources.

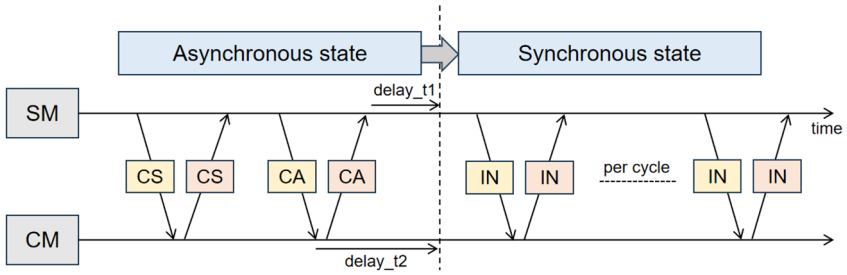


Figure 2. SM–CM protocol control frame exchanges during synchronization startup and maintenance.

2.3. Scheduling and Slot Allocation

The generation of a time-triggered schedule is a constrained assignment problem involving flows, nodes, links, transmission durations, release times, periods, deadlines, and resource conflicts. Each periodic flow must be assigned to a feasible transmission position, and incompatible transmissions must not overlap on the same physical resource. Multi-hop communication additionally requires precedence constraints between consecutive links.

Exact approaches, including integer programming and constraint programming, can express timing, routing, and resource-exclusivity requirements in a unified model.

Constraint-programming methods have been used to optimize routing and scheduling for deterministic hybrid traffic while considering both TT and best-effort services [8]. These methods can provide high-quality or optimal solutions for moderate problem sizes, but their execution time and memory demand may limit direct use in rapidly changing or hardware-constrained environments.

Heuristic and metaheuristic methods reduce the search burden. Static TTE scheduling has been addressed using fuzzy particle swarm optimization and related optimization strategies [7]. Other studies have examined TTE scheduling planning and the construction of time-triggered transmission tables [20]. Heuristic scheduling for TSN has also been proposed to generate GCL entries with lower computational complexity [21]. These methods improve scalability, but their performance depends strongly on the priority rule, search order, and objective function.

EDF is widely used to prioritize tasks with earlier deadlines. Its classical result guarantees optimality for a preemptive uniprocessor model under specified assumptions [6]. Network scheduling violates several of these assumptions because Ethernet frames are generally non-preemptive, flows may traverse multiple links, and the same forwarding node may serve several competing channels. EDF is therefore useful for ranking traffic urgency but cannot alone guarantee balanced use of network resources across a multi-hop topology.

Load balancing provides a complementary criterion. A flow count metric is simple but does not reflect the actual occupation of a channel when flow durations or transmission frequencies differ. A more representative load is the accumulated transmission duration of all instances assigned to a channel. This metric can be incorporated after EDF ordering so that urgent flows are considered first and subsequent assignments are directed toward feasible resources with lower accumulated occupation. The resulting method preserves a clear priority rule while limiting the concentration of traffic on one node or channel.

To clarify the positioning of this work relative to existing deterministic scheduling and implementation studies, Table 1 summarizes representative approaches based on their optimization objectives, key timing constraints, load metrics, hardware validation status, and evaluated scales.

Table 1. Comparison of representative deterministic scheduling and implementation approaches.

Reference/ Approach	Optimization Objective	Key Timing Constraints	Channel Load Metric	Hardware Validation Platform	Evaluated Flow Scale
Chen et al. [7]	Latency minimization via Fuzzy PSO	Deadlines, periods, jitter, transmission windows	None (feasibility focused)	Simulation only	10–50 flows (synthetic network)
Akram et al. [8]	Joint routing & scheduling (CP)	End-to-end deadlines, frame isolation, multi-hop	Path count heuristic	Simulation only	Synthetic hybrid topologies
Yang et al. [21]	Fast heuristic GCL generation	Queue limits, flow periods, port exclusivity	Unweighted frame count	Simulation only	10–100 flows (TSN configuration)
Steinhammer et al. [16]	Hardware TTE switch prototype	Standard AS6802 slot isolation, TT forwarding	None (static routing)	Custom FPGA/ASIC hardware	Standard protocol test streams

Figure 3 illustrates the temporal hierarchy between the matrix cycle and the basic cycle, where each basic cycle is partitioned into time-triggered and event-triggered intervals. This cyclic structure defines the periodic slot boundaries used by the scheduling model in Section 4.

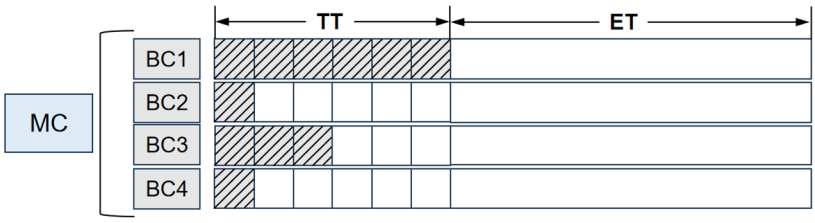


Figure 3. Matrix-cycle and basic-cycle structure with time-triggered and event-triggered intervals.

2.4. Reconfiguration, Mixed Traffic, and Research Gap

Static schedule generation is effective when traffic and topology are known in advance, but industrial networks may experience changes in routes, flow sets, or node availability. Reconfiguration mechanisms for deterministic networks attempt to reduce lost frames and the duration of schedule updates [9]. Cross-domain deterministic transmission has also been studied through software-defined control and coordinated scheduling [22]. These approaches improve adaptability, but online reconfiguration introduces additional consistency and deployment problems that are outside the scope of the present implementation.

Mixed-criticality scheduling must also protect critical traffic from congestion caused by lower-priority flows. Deadline-constrained packet classification and scheduling have been studied for input-queued switches [23]. Congestion control methods and preemption mechanisms can further reduce the impact of non-critical traffic [24]. In industrial wireless and hybrid networks, predictable_Scheduling has been investigated to coordinate time-sensitive and real-time services under variable communication conditions [25,26]. These studies confirm that temporal isolation, resource allocation, and traffic priority should be considered jointly.

Despite these advances, real-time industrial communication requires addressing the tight coupling between synchronization accuracy, schedule validity, traffic window isolation, and hardware-level transmission latency. In particular, there is a clear necessity to resolve the clock synchronization problem within the hardware fabric and validate the resulting transmission schedules directly on an FPGA platform, rather than relying exclusively on software simulations. Furthermore, in non-preemptive networks with heterogeneous frame lengths, a coordinated scheduling methodology is needed to reconcile deadline urgency with duration-weighted channel balancing, ensuring that urgent traffic is safeguarded while communication channels are utilized uniformly. Meeting these requirements calls for an integrated deterministic interface architecture that decouples synchronization state control from packet transmission pipelines, alongside an offline scheduling procedure that guarantees strict constraint satisfaction and hardware-level reproducibility. These engineering needs motivate the FPGA interface architecture and scheduling method presented in the subsequent sections.

3. Proposed FPGA-Based Time-Triggered Ethernet Architecture

This section presents the FPGA architecture used to implement the selected AS6802-based synchronization mechanisms and scheduled TT traffic transmission. It distinguishes protocol-level synchronization functions from FPGA-specific implementation choices, including clock-domain organization, protocol arbitration, and the custom IP/UDP TT data path. FPGA resource utilization, implementation timing, synchronization measurements, and traffic-transmission results are reported in Section 5.

3.1. System Architecture and Physical–Logical Mapping

The experimental platform comprises two FPGA development boards, an external Ethernet switch, and a host computer. FPGA1 implements the logical nodes SM1, SC, SM2,

SW1, and SW2, while FPGA2 implements the CM. SM1 and SM2 operate as synchronization masters, SC operates as a synchronization client, and CM performs the compression and synchronization reference generation functions. SW1 and SW2 denote logical aggregation channels within the topology and scheduling model rather than separate physical switching devices. Each FPGA board has an independent local oscillator. Accordingly, the local clocks on FPGA1 and FPGA2 are initially independent in frequency and phase and are aligned through PCF-based synchronization. The two boards do not share a common physical clock source. Within each board, the Ethernet interface, TTE synchronization core, local clock controller, schedule table controller, and packet processing logic are organized around the internal clock domains described in Section 3.4. The physical platform and the mapping between physical devices and logical roles are shown in Figure 4.

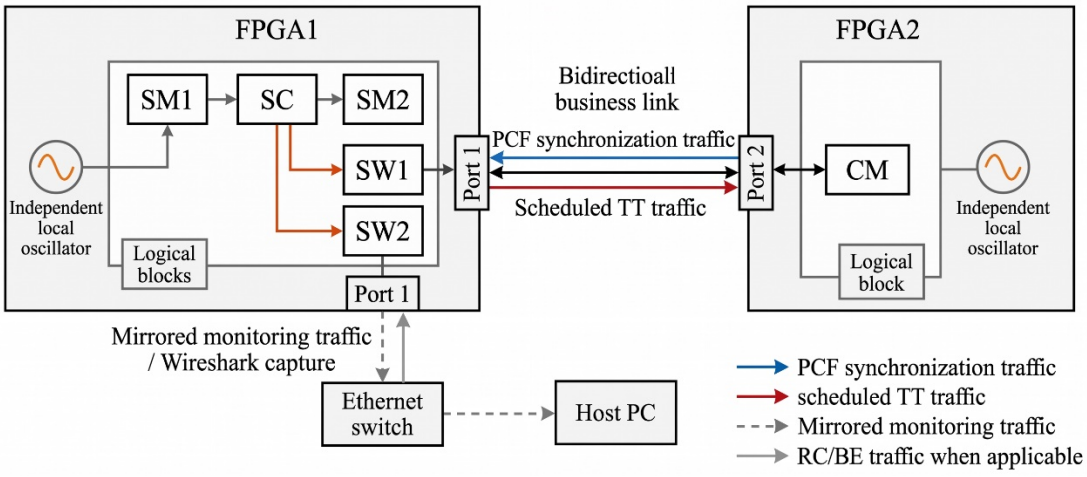


Figure 4. Physical platform and physical–logical mapping of the FPGA-based TTE system.

The two Ethernet ports on each FPGA serve different purposes. Port 2 is the direct board-to-board business link and carries the actual PCF synchronization traffic and scheduled TT traffic between FPGA1 and FPGA2. Port 1 is configured as an observation interface. Traffic associated with the direct business link is mirrored to Port 1 and forwarded through the external Ethernet switch to the host computer for packet capture and analysis. Therefore, the external switch is used primarily as a monitoring path rather than as the forwarding infrastructure of the critical traffic path. The reported synchronization and deterministic transmission results are obtained from the direct FPGA-to-FPGA link, not from a multi-switch forwarding network.

Three classes of traffic coexist in the architecture. PCF frames are processed by the TTE synchronization path. Scheduled TT traffic is transported through a custom IP/UDP data path and is released according to the synchronized local time and the loaded schedule table. Ordinary rate-constrained or best-effort traffic is transmitted only during the residual transmission interval. In the experimental configuration, an 8 ms basic cycle is divided into a synchronization phase from 0 to 1 ms, a TT transmission phase from 1 to 3 ms, and an RC/BE transmission phase from 3 to 8 ms. This partition is an implementation-level traffic isolation strategy and is not intended to represent all possible configurations permitted by AS6802.

On reception, the Ethernet MAC control module examines the EtherType after a valid Ethernet header has been received. Frames identified as PCFs by the implemented EtherType value of 0x891D are delivered to the TTE synchronization core. IPv4 frames with EtherType 0x0800 are delivered to the ARP and UDP processing path. On transmission, the Ethernet control module selects the PCF, TT, or ordinary data path according to the

synchronization state, the local time, the active schedule entry, and the availability of payload data.

FPGA1 implements SM1, SC, SM2, SW1, and SW2; FPGA2 implements CM. Port 2 forms the direct business link for PCF and TT traffic, whereas Port 1 provides a mirrored monitoring path to the switch and host computer.

3.2. Hardware Clock Synchronization Module

The synchronization module implements selected synchronization procedures derived from SAE AS6802, including cold start, PCF exchange, transparent clock processing, permanence processing, compression, and local clock correction. The implementation scope is limited to the mechanisms described in Table 2 and does not constitute complete AS6802 conformance or certification. Figure 5 summarizes the implemented PCF interaction sequence and the associated clock-synchronization processing flow.

Table 2. Implemented functions, verification methods, and unverified scope of the FPGA-based TTE interface.

Function	Implementation Method	Verification Method	Not Yet Verified
PCF processing	Hardware parsing and generation of the implemented $0 \times 891D$ PCF format, including validation, buffering, transparent clock handling, and routing to the synchronization core	Packet capture, frame parsing, and synchronization state transition tests	Full conformance testing for all PCF fields, boundary conditions, and independent implementations
Synchronization roles	FPGA1 implements SM1, SM2, and SC; FPGA2 implements CM; SW1 and SW2 are logical aggregation channels	Cold start, acknowledgment, integration frame, and membership-related functional tests	Independent multi-vendor physical node behavior and all role-specific fault conditions
Clock correction	Permanence, compression, local clock correction, and cycle control logic implemented in FPGA hardware	Multi-node synchronization experiments reported in Section 5	Oscillator failure, PCF loss, inconsistent membership, and fault recovery behavior
TT application traffic	Custom IP/UDP traffic path triggered by synchronized local time and a loaded schedule table	Slot-level traffic capture, deadline analysis, and receive side packet inspection	Third-party interoperability and equivalence to a complete standardized TT application profile
Protocol arbitration	PCF priority, schedule-ordered TT selection, and phase-based RC/BE admission control	Mixed-traffic packet capture and phase boundary verification	Formal worst-case analysis for all queueing and backpressure conditions
Reserved window protection	Guarded admission of ordinary frames; no verified mid-frame interruption or resumption	Variable-length frame tests at phase boundaries	IEEE Ethernet frame preemption behavior
Fault tolerance	Timeout, membership, and implemented synchronization state handling	Functional state and timeout tests	Comprehensive fault containment, redundancy, failure injection, and recovery validation

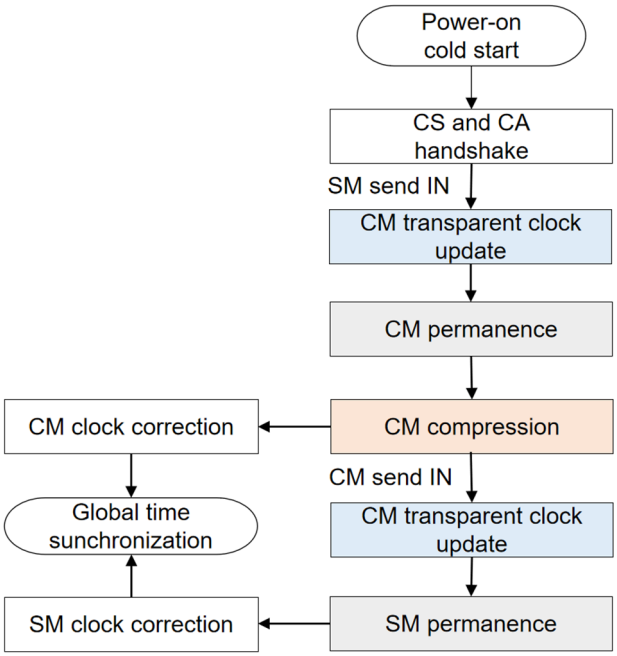


Figure 5. Algorithm flow of PCF interaction and clock synchronization.

After power-on, the synchronization masters and compression master enter the cold-start procedure. CS and CA frames are exchanged until the relevant timeout, membership, and state transition conditions are satisfied. The synchronization masters then transmit IN frames. At the CM, received INs are processed by the transparent clock, permanence, and compression functions. The CM subsequently transmits updated IN frames, which are processed by the permanence and clock correction functions at the SM and SC nodes. This closed loop aligns the local logical clocks of the participating nodes.

The transparent clock mechanism records the configured and observed delay contributions associated with PCF transmission and reception. For a transmitting node, the transparent clock value is initialized as

$$TC_0 = D^{\mathrm{dyn}}_{\mathrm{send},0} + D^{\mathrm{sta}}_{\mathrm{send},0}. \tag{1}$$

For an intermediate relay node, the accumulated transparent clock is expressed as

$$TC_i = TC_{i-1} + D^{\mathrm{dyn}}_{\mathrm{relay},i} + D^{\mathrm{sta}}_{\mathrm{relay},i} + D_{\mathrm{wire},i}. \tag{2}$$

At a receiving node, the value becomes

$$TC_n = TC_{n-1} + D^{\mathrm{dyn}}_{\mathrm{receive},n-1} + D^{\mathrm{sta}}_{\mathrm{receive},n-1} + D_{\mathrm{wire},n}. \tag{3}$$

where the superscripts dyn and sta denote the dynamic queueing/processing delay and static hardware delay at the transmitting, relay, or receiving node, respectively, and $D_{\mathrm{wire},i}$ denotes the propagation delay on link segment i. The maximum configured end-to-end transmission delay is represented by

$$D_{\max} = \max(TC_n). \tag{4}$$

In this design, the timestamp used by the synchronization logic is generated at an internal MAC-side event point. On reception, the timestamp corresponds to the point at which the received PCF has been accepted by the MAC-side receive control logic. On transmission, it corresponds to the point at which the transmit controller commits the

PCF to the MAC-side transmission path. The implementation does not claim independent timestamp generation at the PHY pins. Therefore, fixed MAC, PHY, serialization, and cable delay contributions are represented through the configured delay terms rather than directly measured at the physical interface.

The PCFReceive module parses the Ethernet and PCF fields, checks the implemented frame type and synchronization domain information, and discards invalid frames. Its output is a valid internal PCF record containing the PCF type, integration cycle, membership information, synchronization priority, synchronization domain, transparent clock value, and reception timestamp. The PermanenceProc module then computes a permanence delay and a permanence point according to

$$D_{\text{perm}} = D_{\max} - TC_n, \tag{5}$$

$$P_{\text{perm}} = P_{\text{receive}} + D_{\text{perm}}. \tag{6}$$

Here, P_{receive} is the local reception timestamp, and the permanence point P_{perm} provides a corrected reference time for PCFs received through paths with different transmission delays. The module also performs the ordering and validity handling required by the implemented synchronization sequence.

At the CM, the CompressProc module processes the permanence information from the participating synchronization masters within the configured observation interval. The compression delay and compression point are represented as

$$D_{\text{comp}} = W_{\text{obs}} + D_{\text{calc}} + C_{\text{comp}}, \tag{7}$$

$$P_{\text{comp}} = P_{\text{perm}} + D_{\text{comp}}. \tag{8}$$

W_{obs} is the maximum observation window. D_{calc} is the calculation delay. C_{comp} is the configured compression correction. The compression result is used to generate the CM synchronization reference and to construct the outgoing IN frame.

The TTE core separates PCF data processing from state control. The data path transfers PCF records sequentially through frame reception, permanence processing, compression processing where applicable, and PCF transmission. The control path coordinates state transitions and processing commands using the outputs of the timeout, membership, local clock, local cycle, and synchronization computation modules. StateCtrl determines the local synchronization state. TimeOut evaluates state-dependent time limits. CliqueDetect evaluates the implemented membership condition. LocalCycleCtrl maintains the integration cycle information, while LocalClkCtrl maintains the local logical clock. SyncCompute converts the permanence and compression results into a clock correction event or correction value. PCFTransmit encapsulates the processed synchronization information and requests frame transmission.

This separation provides a modular hardware processing structure that enables the control state machine to deterministically coordinate the synchronization pipeline alongside the quantitative timing and resource evaluations presented in Section 5. Figure 6 shows the hardware organization of the synchronization core, and Figure 7 illustrates the timing relationships of the principal synchronization signals during cold-start and synchronized operation.

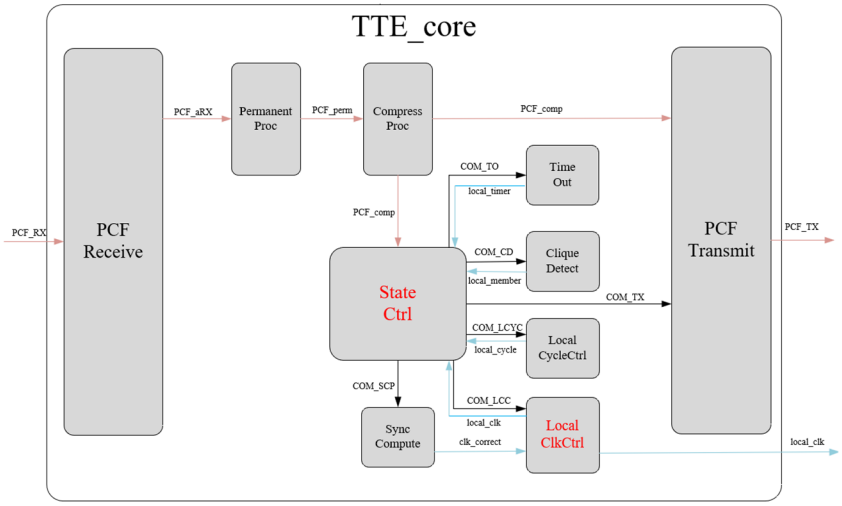


Figure 6. Hardware organization of the TTE time-synchronization core.

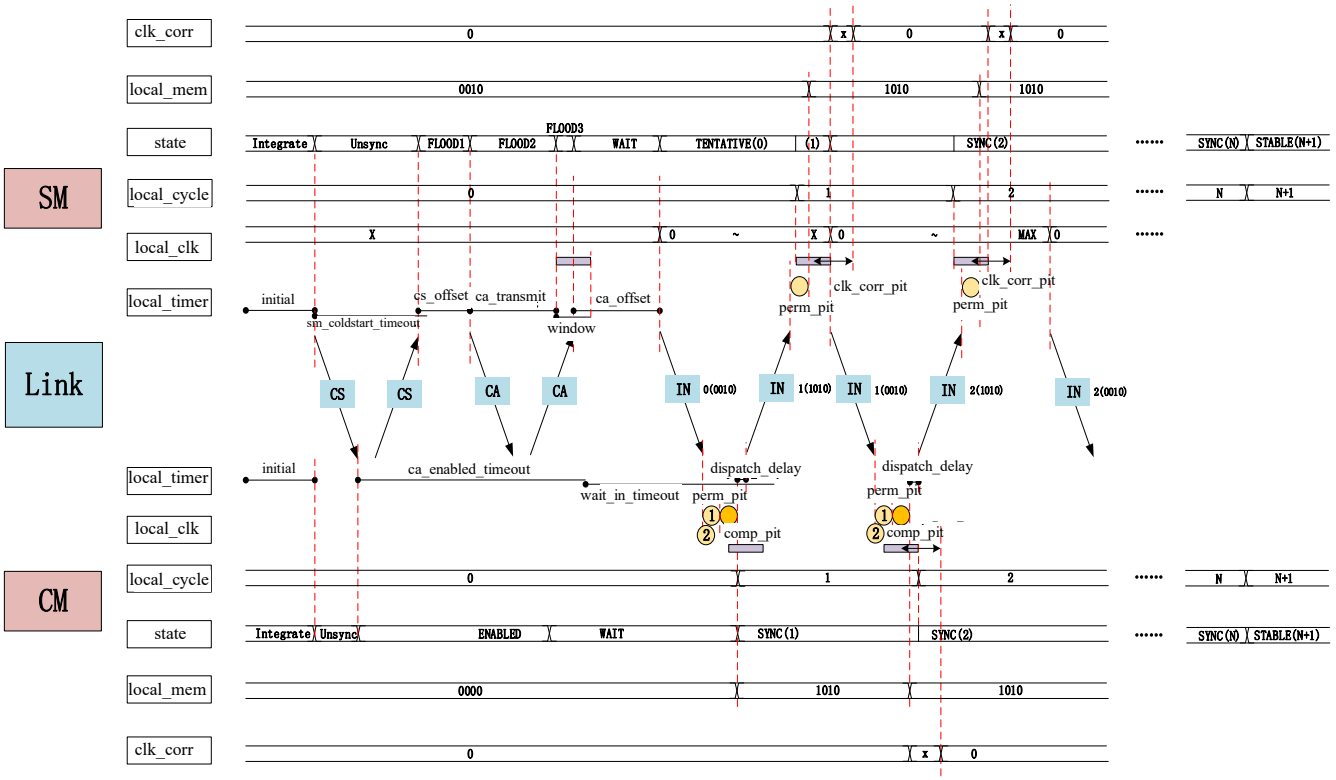


Figure 7. Timing relationships of the core synchronization signals during cold start and synchronized operation.

3.3. Scheduled Traffic Transmission and Protocol Arbitration

The schedule table connects the offline scheduling algorithm to the FPGA transmission hardware. Each valid schedule entry specifies at least the basic cycle position, transmission time or slot identifier, TT flow identifier, logical channel assignment, source and destination addressing information, payload length, and validity or repetition control. The table is loaded by the schedule-table loading module and stored in a dedicated 256-entry on-chip schedule memory. During operation, the controller compares the synchronized local time and local cycle value with the next valid schedule entry. A TT transmission request is issued only when the time condition is satisfied and the associated payload is available in the transmit buffer.

The scheduled time is interpreted as an internal frame release point rather than as the first bit transmission instant at the PHY pins. After a schedule match, the UDP control module constructs or selects the corresponding IP/UDP packet, and the Ethernet MAC control module serializes the Ethernet header, IP/UDP header, payload, padding where necessary, and frame check sequence. Consequently, the physical transmission time includes the internal request, arbitration, frame construction, MAC, and PHY delays discussed in Section 3.4.

The basic cycle is partitioned to isolate the three traffic classes. PCF synchronization traffic occupies the interval from 0 to 1 ms, scheduled TT traffic occupies the interval from 1 to 3 ms, and RC/BE traffic is admitted from 3 to 8 ms. When PCF and TT requests are simultaneously pending, the PCF request has priority because continuous synchronization is required for schedule validity. Within the TT phase, the schedule table order determines the transmission order of eligible TT flows.

If a TT entry becomes active while its payload has not been prepared, the controller does not transmit an incomplete or invalid frame. The entry may be held within its configured eligibility interval if a later release still satisfies the relevant timing policy. Otherwise, the controller records the entry as missed and advances to the next valid request. The treatment of missed TT entries is an implementation-level scheduling policy and is evaluated through the deadline and packet-loss experiments in Section 5.

RC/BE traffic is admitted only if the current local time is within the RC/BE phase and the candidate frame can complete transmission before the next reserved PCF or TT phase begins. The admission decision is based on the maximum configured frame length and the corresponding link occupation time. When the remaining time is insufficient, the ordinary frame is deferred to a subsequent RC/BE interval. This mechanism prevents ordinary frames from crossing into reserved traffic windows.

The present implementation does not claim Ethernet frame preemption. In particular, an RC/BE frame that has already entered MAC serialization is not interrupted and resumed when a PCF or TT request arrives. Critical traffic is protected through time window partitioning, admission control, and guard time calculation rather than mid-frame preemption. Accordingly, ordinary traffic must be prevented from starting when its complete transmission would overlap with a reserved synchronization or TT interval. Figure 8 shows the integrated FPGA architecture, including the synchronization core, Ethernet processing path, protocol arbitration logic, and schedule-table controller.

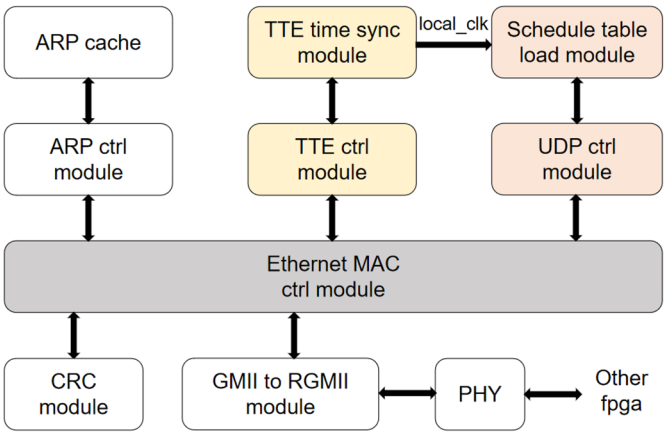


Figure 8. Integrated FPGA TTE system architecture.

3.4. Clock Domains and Implementation Details

The architecture uses a 125 MHz Ethernet-related clock and a 200 MHz clock generated by the PLL. The 125 MHz clock, corresponding to an 8 ns period, is used as the

reference time base for the local logical clock, local cycle counter, synchronization timing, schedule table comparison, and primary MAC-side control. In the experimental configuration, the 0.4 ms IN interval corresponds to 50,000 cycles of the 125 MHz clock. Unless another scaling factor is configured, the local time counter therefore advances with an 8 ns nominal resolution.

The 200 MHz PLL-generated clock is used by the GMII-to-RGMII conversion logic and the PHY-facing interface. RGMII transfers data using double data rate signaling, whereas the principal FPGA processing modules use single-edge synchronous logic. The GMII-to-RGMII module performs the required conversion between these representations. The synchronization algorithm and schedule controller remain referenced to the 125 MHz logical time domain; the 200 MHz domain is used for Ethernet interface timing and data conversion.

Clock domain crossings are handled through synchronized control handshakes and asynchronous FIFO structures. Single-bit control signals are synchronized before being consumed in the destination domain. Multi-bit metadata, timestamps, and packet data are transferred as stable valid ready transactions or as complete FIFO entries, rather than by independent sampling of individual bits. This arrangement prevents incoherent transfer of timing values and frame metadata between the 125 MHz and 200 MHz domains.

Receive-side buffers decouple Ethernet reception from PCF processing and UDP payload storage, while transmit-side buffers hold TT payloads until their scheduled release points. In the implemented RTL configuration, the schedule-table memory supports up to 256 entries, and the TT payload buffer provides 16 kB of storage realized using four 36 Kb block RAMs (BRAM36). These buffers absorb bounded processing and clock-domain-crossing delays; they do not eliminate the need to verify schedule feasibility or deadline compliance.

For an Ethernet frame of length L bytes on a 1 Gb/s link, the nominal serialization time is

$$T_{\text{ser}} = \frac{8L}{10^9}. \tag{9}$$

The complete wire occupancy time additionally includes the 8-byte preamble/start-of-frame delimiter (SFD) and the 12-byte interframe gap (IFG), yielding $8(L + 20)/10^9$ s. This maximum wire occupancy time is used by the RC/BE admission controller to define the required guard interval before each reserved PCF and TT window.

3.5. Implemented Functions and Standard-Related Boundaries

The design combines selected AS6802-related synchronization mechanisms with a custom schedule-driven UDP traffic path. Specifically, the PCF processing and clock synchronization functions follow the SAE AS6802 synchronization model, whereas the IP/UDP transport path serves as an engineering extension to carry scheduled application data through the FPGA Ethernet stack.

Accordingly, the term AS6802-based in this study refers to the implementation of selected synchronization mechanisms and PCF processing procedures derived from the standard. It does not constitute a declaration of complete AS6802 certification or universal interoperability. The remaining boundaries include PHY-level timestamp validation, independent interoperability testing, complete fault handling, redundancy behavior, and exhaustive standard conformance verification.

4. Deadline-Aware Load-Balanced Traffic Scheduling

The scheduling method generates an offline transmission table for periodic TT flows while jointly considering deadlines, link availability, reserved transmission windows, and load distribution. In contrast to a procedure that first selects the least loaded node and sub-

sequently attempts to place the flow, the proposed method constructs only timing-feasible candidates and then applies load balancing among those candidates. Schedule generation is performed by host-side software because topology traversal, candidate enumeration, and conflict checking are not included in the FPGA data path.

4.1. Scheduling Inputs and Instance Generation

The scheduling input consists of the TT flow set $\mathcal{F}$, the network graph $G = (\mathcal{V}, \mathcal{E})$, and a set of candidate paths. A flow f_i is represented as

$$f_i = (id_i, \text{type}_i, s_i, d_i, T_i, D_i, L_i, \phi_i, \mathcal{P}_i), \tag{10}$$

where id_i and type_i denote the flow identifier and traffic type, respectively; s_i and d_i are the source and destination nodes; T_i is the period; D_i is the relative end-to-end deadline; L_i is the frame length; ϕ_i is the release phase; and $\mathcal{P}_i$ is the set of admissible paths or logical aggregation channels. A zero phase is used when no independent phase is specified. Each link $e \in \mathcal{E}$ is characterized by its transmission rate R_e, propagation delay δ_e, and guard time g_e. The link occupation time of flow f_i on link e is

$$\tau_{i,e} = \left\lceil \frac{8(L_i + H_i)}{R_e \Delta} \right\rceil \Delta + g_e, \tag{11}$$

where H_i accounts for the Ethernet, IP/UDP, and physical-layer overhead included in the scheduling model, and Δ is the scheduling time quantum (set to Δ = 8 ns to match the 125 MHz FPGA clock period). This formulation ensures that the resource reservation covers complete frame transmission rather than payload transmission alone.

The matrix cycle is the least common multiple of all flow periods, whereas the basic cycle is their greatest common divisor:

$$MC = \text{lcm}_{f_i \in \mathcal{F}}(T_i), BC = \text{gcd}_{f_i \in \mathcal{F}}(T_i). \tag{12}$$

Because the experimental flow periods are integer multiples of 8 ms, their greatest common divisor directly yields a basic cycle of 8 ms. At 125 MHz, one basic cycle corresponds to 10^6 clock periods. As illustrated in Figure 9, the interval from 0 to 1 ms is reserved for PCF traffic, 1 to 3 ms for TT traffic, and 3 to 8 ms for RC/BE traffic. The corresponding TT transmission window in basic cycle b is

$$W_b^{TT} = [\, b \cdot BC + 1 \text{ ms}, \;\; b \cdot BC + 3 \text{ ms}). \tag{13}$$

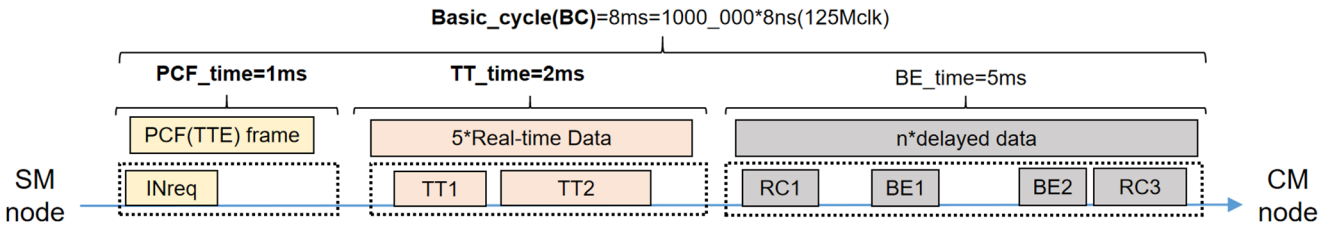


Figure 9. Protocol partitioning within an 8 ms basic cycle.

Each periodic flow is expanded into individual instances over one matrix cycle. The number of instances of f_i is

$$n_i = \frac{MC}{T_i}, \tag{14}$$

and instance $J_{i,k}$, $0 \leq k < n_i$, has the release time and absolute deadline

$$r_{i,k} = \phi_i + kT_i,\ d_{i,k} = r_{i,k} + D_i. \tag{15}$$

The complete scheduling set is $\mathcal{J} = \{J_{i,k}\}$, with total size $N_J = \sum_i n_i$. Because each evaluated flow has a period no shorter than the basic cycle and a fixed release phase, each flow generates at most one transmission instance within any single basic cycle.

4.2. Deadline-Aware Candidate Selection

Scheduling is performed at the instance level. Unassigned instances are ordered by nondecreasing absolute deadline. For instances with the same deadline, smaller slack is given priority. The initial slack of $J_{i,k}$ is defined as

$$\sigma_{i,k} = d_{i,k} - r_{i,k} - \min_{p \in \mathcal{P}_i} \sum_{e \in p} (\tau_{i,e} + \delta_e). \tag{16}$$

This tie-breaking rule prioritizes instances with less temporal flexibility when deadlines are identical, while preserving the strict feasibility checks applied to every candidate.

For the selected instance, the scheduler enumerates its admissible paths and possible transmission positions within the TT windows intersecting $[r_{i,k}, d_{i,k}]$. A candidate is retained only if it is released before transmission, completes no later than its absolute deadline, remains entirely within valid TT windows, preserves the required guard interval, and does not overlap an existing reservation on any shared physical resource. For a multi-hop path, transmission on hop $h + 1$ may begin only after completion on hop h, including the modeled forwarding and propagation delay:

$$s_{i,k,h+1} \geq s_{i,k,h} + \tau_{i,e_h} + \delta_{e_h}. \tag{17}$$

The final completion time must satisfy $C_{i,k} \leq d_{i,k}$. Logical channels mapped to the same physical interface are checked against a shared physical resource calendar to prevent overlapping transmissions on the egress link. Candidates crossing the PCF or RC/BE windows are rejected. Reservations near the end of the matrix cycle are additionally checked against the first cycle reservations of the next repeated matrix cycle.

4.3. Load-Aware Assignment and Failure Handling

Load balancing is applied only after candidate feasibility has been established. Let B_e denote the TT transmission capacity available on channel resource e during one matrix cycle, where e represents either an independent link or a configured logical aggregation channel such as SW1 and SW2. Its normalized load after assigning candidate c is

$$U_e(c) = \frac{A_e + \sum_{q \in c: e \in q} \tau_{q,e}}{B_e}, \tag{18}$$

where A_e is the accumulated transmission duration already assigned to resource e. Temporal conflict checks are enforced on the shared physical interface calendar, whereas channel occupation $U_e(c)$ is tracked per channel to distribute traffic across admissible routes. For each feasible candidate, the scheduler evaluates the resulting load spread:

$$\Psi(c) = \max_{e \in \mathcal{E}} U_e(c) - \min_{e \in \mathcal{E}} U_e(c). \tag{19}$$

The candidate with the smallest $\Psi(c)$ is selected. Ties are resolved by the earliest end-to-end completion time and then by path identifier to ensure deterministic and reproducible output. This lexicographic rule first enforces all timing and resource constraints, then

improves load distribution, and finally favors earlier completion. For reporting purposes, the load-balance rate is

$$LBR = \frac{\min\limits_{e:U_e>0} U_e}{\max\limits_{e:U_e>0} U_e}, \tag{20}$$

where channel resources that cannot carry the considered flows are excluded. A value close to one indicates a more uniform distribution across the configured channels, but LBR is not used as evidence of deadline feasibility. If no feasible candidate exists for an instance, the implemented scheduler terminates and reports that no feasible schedule was found under the current configuration. The host-side procedure is summarized in Algorithm 1.

Algorithm 1. Deadline-aware load-balanced scheduling

Input: Periodic TT flow set F; topology graph G; admissible path or logical-channel set P for each flow; TT transmission windows; resource calendars.
Output: Validated schedule table_S, or NO_SCHEDULE_FOUND.
Determine the matrix cycle and basic cycle from the input flow periods.
Expand every periodic TT flow into its transmission instances within one matrix cycle.
For each instance, determine its release time, absolute deadline, required transmission duration, admissible paths or channels, and applicable TT windows.
Sort all instances by increasing absolute deadline.
For instances with the same deadline, give priority to the instance with smaller available slack.
Initialize an empty schedule table_S.
Initialize the calendar of every physical resource and set all resource loads to zero.
for each instance J in the sorted instance set do
    Initialize an empty feasible-candidate set C.
    for each admissible path or logical channel p of J do
        Enumerate candidate transmission positions within the valid TT windows.
        for each candidate position c do
            Check that c is not earlier than the release time of J.
            Check that c completes before the absolute deadline of J.
            Check that c remains completely inside the applicable TT window.
            Check the required guard interval before and after the reservation.
            Check that c does not overlap an existing reservation on any shared physical resource.
            If p contains multiple hops, check the forwarding order and delay between consecutive hops.
            Check that c does not create a conflict at the boundary between consecutive matrix cycles.
            if all checks are satisfied then
                Estimate the resulting normalized load of the resources used by c.
                Add c to C.
            end if
        end for
    end for
    if C is empty then
        return NO_SCHEDULE_FOUND.
    end if
    Select the candidate with the smallest resulting load spread.
    If candidates have the same load spread, select the one with the earliest completion time.
    If a tie remains, select the candidate with the lowest path or channel identifier.

Algorithm 1. *Cont.*

32: Reserve the selected candidate in all relevant resource calendars.
33: Update the corresponding resource loads.
34: Append the selected candidate to S.
35: end for
36: Independently validate S for instance completeness, deadlines, conflicts, guards, and cycle boundaries.
37: if validation fails then
38: return NO_SCHEDULE_FOUND.
39: end if
40: return S.

4.4. Schedule Validation and Timing Analysis

The generated table is checked by an independent validator that does not reuse the candidate selection decisions of the scheduler. The validator reconstructs all expected instances over one matrix cycle and confirms that each instance appears exactly once. It verifies release times, absolute deadlines, physical resource conflicts, multi-hop precedence, TT window containment, guard intervals, and conflicts between the final reservations of one matrix cycle and the initial reservations of the next. A schedule is accepted only when all checks succeed. This establishes that the output satisfies the modeled constraints and assumptions; it does not establish that the heuristic can discover every feasible schedule.

For an accepted instance, the modeled response time is

$$R_{i,k}^{model} = C_{i,k}^{model} - r_{i,k}. \tag{21}$$

This value differs from hardware-measured latency, which includes the actual FPGA release, arbitration, buffering, MAC/PHY processing, propagation, and receive-side timestamping delays. The maximum observed hardware latency is the largest value in the finite measurement set and should not be interpreted as a formal worst-case bound. A deterministic upper bound can be claimed only when the release phases, processing delay bounds, clock error, frame lengths, and all interference assumptions are explicitly included in the model.

The present model assumes periodic releases with fixed phases. If traffic may arrive at arbitrary times, the waiting time until the next eligible TT window must be added. For the implemented 8 ms cycle, such waiting may approach one basic cycle before transmission and therefore cannot be inferred from the reported average latency. Deadline satisfaction is consequently stated only for the validated periodic flow set under the specified release and timing assumptions.

4.5. Computational Complexity

Let N_J be the total number of instances in one matrix cycle, $P_{\max}$ the maximum number of admissible paths per instance, Q the maximum number of candidate transmission positions examined per path, and $H_{\max}$ the maximum number of resources on a path. Instance sorting requires $O(N_J \log N_J)$. Candidate generation and validation require at most

$$O(N_J P_{\max} Q H_{\max}), \tag{22}$$

when resource conflicts are checked through indexed calendars with bounded time interval queries. If a linear scan of existing reservations is used, the conflict checking term must additionally include the number of reservations already stored on each resource. The independent validator has the same order as checking all accepted reservations and

resource calendars. Because the present scheduler operates as a single-pass greedy heuristic without recursive backtracking, its worst-case time complexity remains polynomial. These computations are executed by host-side software, while the FPGA stores and executes the validated schedule table.

5. Experimental Setup and Evaluation Methodology

This section evaluates the FPGA implementation, PCF-based synchronization behavior, schedule table validity, and scheduled TT transmission performance. The scheduling table was generated offline by host-side software and subsequently loaded into the FPGA implementation. The FPGA executed PCF synchronization, TT frame release, protocol arbitration, and Ethernet transmission according to the configuration described in Sections 3 and 4.

5.1. Experimental Platform and Configurations

The experimental platform consisted of two Xilinx Artix-7 FPGA development boards (San Jose, CA, USA) (both equipped with XC7A100T-1CSG324C devices), a five-port Gigabit Ethernet switch, and a host computer. FPGA1 implemented the logical roles SM1, SC, SM2, SW1, and SW2, whereas FPGA2 implemented the CM role. The two boards were driven by independent onboard crystal oscillators. Because SM1, SC, and SM2 resided within FPGA1 and shared its internal clocking infrastructure, their measurements characterize intra-FPGA role consistency, whereas the FPGA1–FPGA2 exchange evaluates synchronization across two independent board-level clock sources.

Port 2 served as the sole direct board-to-board physical business link carrying PCF and TT frames, whereas Port 1 mirrored transmitted and received traffic through the external switch to the host computer exclusively for packet inspection and was excluded from latency calculations. In the evaluated topology, SW1 and SW2 were internal logical aggregation channels rather than separate physical switch ports; both mapped to the shared Port 2 MAC/PHY egress interface on FPGA1. Consequently, physical conflict checking was enforced on the shared Port 2 resource calendar, and the LBR reported in this study characterizes logical-channel load distribution rather than load balancing across independent physical egress links.

The system operated at 1 Gb/s with a 125 MHz local clock and a 200 MHz PLL clock for the RGMII interface. The IN frame interval was set to 0.4 ms, and the TT schedule used an 8 ms basic cycle and a 32 ms matrix cycle, reserving 0–1 ms, 1–3 ms, and 3–8 ms within each basic cycle for PCF, TT, and RC/BE traffic, respectively. The principal FPGA hardware validation used five periodic TT flows generating 13 instances per matrix cycle, whereas the 20–200-flow scalability evaluations in Section 5.6 were executed solely by the host-side scheduler without FPGA re-synthesis or hardware transmission. The physical-to-logical route mapping and timing parameters of the five evaluated TT flows are summarized in Table 3, while the platform, synthesis, and scheduling configurations are listed in Table 4. Figure 10 illustrates the board-level module partitioning and physical-to-logical role mapping between FPGA1 and FPGA2; Figure 11 depicts the physical testbed wiring, distinguishing the direct Port 2 business link from the mirrored Port 1 monitoring path; and Figure 12 summarizes the seven-step experimental workflow from bitstream loading and synchronization convergence to schedule loading, packet capture, and timing evaluation.

Table 3. Physical–logical mapping and TT-flow parameters (*MC* = 32 ms, *BC* = 8 ms).

Flow	Logical Route	Physical Link	T_i (ms)	n_i	ϕ_i (ms)	D_i (ms)	L_i/L_w (B)	τ_i (µs)
TT1	SM1 → SW1 → CM	Shared Port 2	16	2	0.0	5.0	434/500	200–240
TT2	SM1 → SW1 → CM	Shared Port 2	8	4	0.0	4.0	309/375	160
TT3	SM2 → SW2 → CM	Shared Port 2	8	4	0.0	4.0	559/625	200
TT4	SC → SW1 → CM	Shared Port 2	16	2	0.0	5.0	684/750	280
TT5	SM2 → SW2 → CM	Shared Port 2	32	1	0.0	6.0	309/375	160

Table 4. Experimental platform configuration.

Component	Reference Configuration
FPGA device (FPGA1 and FPGA2)	Xilinx Artix-7 XC7A100T-1CSG324C (speed grade -1)
FPGA boards and PHY	Dual-port Artix-7 development boards with Realtek RTL8211E-VL Gigabit RGMII PHYs
Board clock sources	Independent 125 MHz onboard crystal oscillators (±25 ppm) on FPGA1 and FPGA2
Internal clock constraints	125 MHz time-base and MAC clock (8 ns); 200 MHz PLL clock for GMII-to-RGMII conversion (5 ns)
Physical Ethernet links	Port 2: 1 Gb/s direct Cat-6 business link (1 m); Port 1: mirrored monitoring link via five-port unmanaged GbE switch
FPGA tool and strategies	Xilinx Vivado 2020.2; synthesis: Vivado Synthesis Defaults; implementation: Performance_ExplorePostRoutePhysOpt; post-implementation timing analysis
Modules included in utilization	PCF synchronization core, LocalClkCtrl, schedule controller (256-entry BRAM; 13 active), 16 kB TT payload buffer, UDP/IP/MAC data path, RGMII interface, and Port 1 mirroring logic
Scheduling parameters	Time quantum Δ = 8 ns (1 clock tick); protocol/PHY overhead H_i = 66 B; guard band g_e = 0.96 µs (120 clock ticks)
Host scheduler and capture PC	Intel Core i7-10700 CPU @ 2.90 GHz, 16 GB DDR4 RAM, Ubuntu 20.04 LTS, Python 3.8/C++17 scheduler, Wireshark 4.0.3

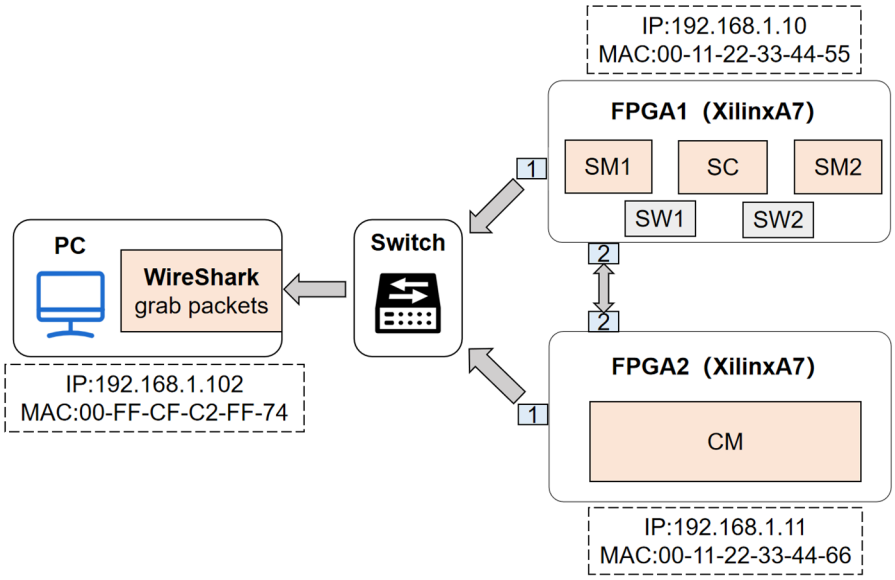


Figure 10. Hardware platform architecture and role mapping.

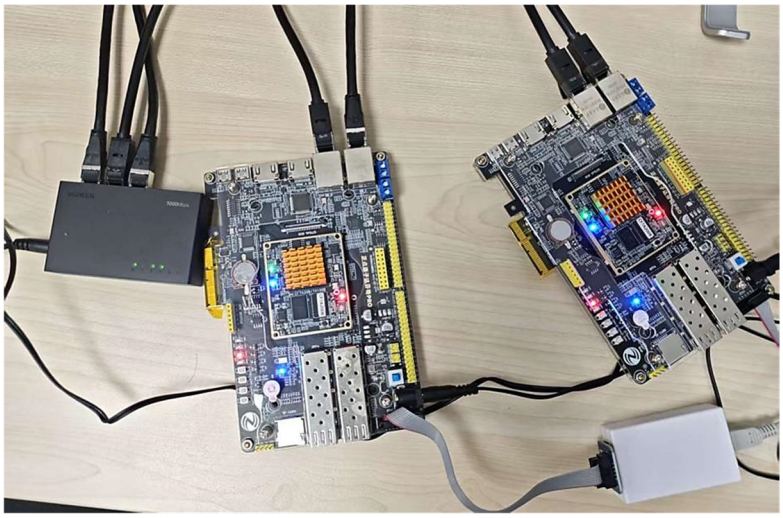


Figure 11. Physical experimental testbed and link connections.

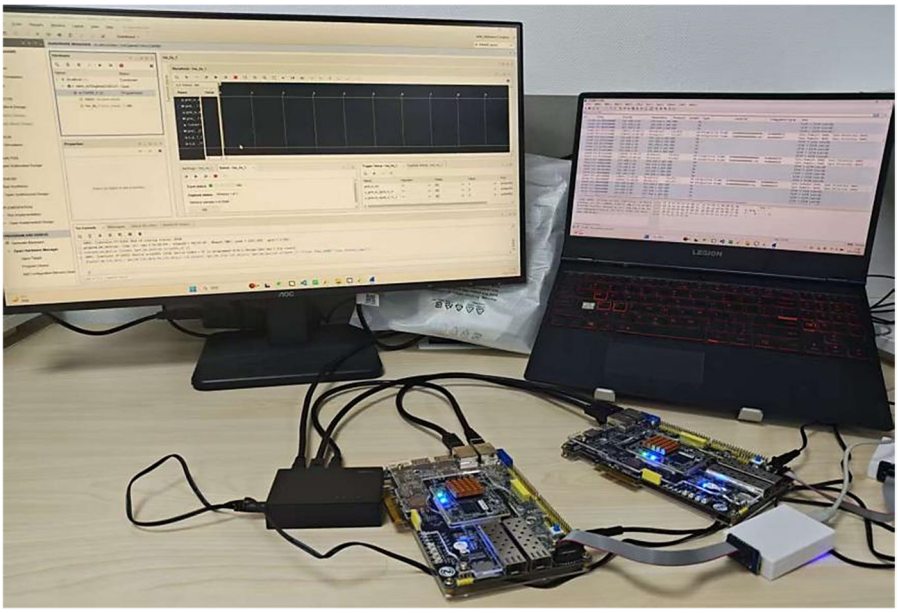


Figure 12. Experimental workflow from configuration to timing analysis.

5.2. Measurement Definitions and Experimental Protocol

The synchronization evaluation distinguished internal timestamp-interval stability from external inter-board phase alignment. First, the PCF timestamp-interval residual evaluated the deviation of consecutive PCF-associated Local_Clk intervals from the configured 0.4 ms integration period after the synchronization state became stable. Let t_j and t_{j+1} denote two adjacent MAC-side Local_Clk timestamps with an 8 ns resolution associated with consecutive IN frames. The timestamp-interval residual was calculated as

$$e_j = (t_{j+1} - t_j) - 0.4\ \text{ms}. \tag{23}$$

This metric characterizes interval stability rather than the absolute phase offset between two independent board clocks; therefore, a bounded residual within ±8 ns does not represent direct inter-board phase error. The signed mean, mean absolute error, maximum absolute residual, and standard deviation were calculated as

$$\bar{e} = \frac{1}{N}\sum_{j=1}^{N} e_j, MAE = \frac{1}{N}\sum_{j=1}^{N} |e_j|, \tag{24}$$

$$E_{\max} = \max_j \mid e_j \mid, \sigma_e = \sqrt{\frac{1}{N-1}\sum_{j=1}^{N}(e_j - \bar{e})^2}. \quad (25)$$

Second, to evaluate physical inter-board clock alignment, each FPGA routed a synchronized cycle-strobe signal from LocalClkCtrl to a dedicated test pin. The external clock-output skew was measured on rising edges using a 1 GHz digital oscilloscope as $\Delta\tau_{\text{comp},m} = \tau_{\text{FPGA1},m} - \tau_{\text{FPGA2},m} - \Delta\tau_{\text{cal}}$, where $\Delta\tau_{\text{cal}}$ is the probe and channel delay difference calibrated by connecting both matched probes to the same FPGA1 reference pin prior to testing, as summarized in Table 5. Five independent experimental runs were conducted. Before each run, both boards were power-cycled, reconfigured with their bitstreams, synchronized to the stable_State, and loaded with the validated schedule table. Each run lasted 30 min after synchronization convergence, yielding 4.5×10^6 valid IN-interval samples per run and 2.25×10^7 samples in total, alongside 1.0×10^5 decimated rising-edge pairs per run and 5.0×10^5 edge pairs in total. The one-hour drift was defined as the difference between the signed mean residuals obtained from two 10-min observation windows separated by one hour.

Table 5. Synchronization, skew, and latency measurement protocol configurations.

Parameter	Reference Configuration
Oscilloscope and sampling	Tektronix MSO54, 1 GHz analog bandwidth, 6.25 GS/s real-time sampling rate
Time-interval accuracy	160 ps sample period, 10 ps interpolated resolution, ±150 ps dual-channel uncertainty
Probes and test signal	Two matched TPP1000 1 GHz passive probes, 1.3 m length, connected to LocalClkCtrl strobe pins
Reference edge and rate	Rising edge at 1.65 V threshold for 3.3 V LVCMOS pins, 18 ms strobe period
Channel delay calibration	Common-source split-signal calibration on FPGA1 pin; $\Delta\tau_{\text{cal}}$ = 0.18 ns; uncompensated mean and max: 0.89 ns and 9.78 ns; compensated: 0.71 ns and 9.60 ns
Timestamp latch points	t_j: MAC-side IN timestamp; t_q^{tx}: schedule release to UDP/MAC; t_q^{rx}: destination MAC frame valid
Experimental run protocol	5 independent 30-min runs with full power-cycle, re-synchronization, and schedule reload before each run
Statistical metrics	Mean, standard deviation, P50, P95, P99, and maximum observed value without outlier removal

Third, traffic latency was measured from the source-side FPGA release timestamp to the destination-side FPGA receive timestamp. For packet q, the latency was defined as

$$L_q = t_q^{\text{rx}} - t_q^{\text{tx}}, \quad (26)$$

where t_q^{tx} is latched on FPGA1 upon schedule release to the UDP/MAC path and t_q^{rx} is latched on FPGA2 upon frame validation by the receive MAC. Thus, L_q spans source buffering, network-stack traversal, 1 m cable propagation, and MAC reception under the shared synchronized time base, whose sub-10 ns skew is negligible compared to millisecond-scale TT latency. Mirrored Port 1 Wireshark timestamps were used solely to verify traffic order and slot isolation rather than latency. Across five 30-min runs spanning 281,250 matrix cycles, the five flows generated 3,656,250 TT packets. All distributional metrics were evaluated directly from raw samples without outlier rejection.

5.3. Hardware Implementation Results

The complete TTE interface was synthesized, placed, and routed for the XC7A100T-1CSG324C device on FPGA1. The post-implementation hierarchy included the PCF synchronization core, LocalClkCtrl, schedule controller, TT payload buffer, UDP/IP/MAC data path, RGMII interface, and Port 1 mirroring logic. Static timing analysis confirmed timing closure across both clock domains, achieving a worst negative slack of +0.74 ns for the 125 MHz system clock and +0.42 ns for the 200 MHz RGMII clock, with zero failing endpoints. Under the evaluated five-flow configuration, the design consumed 14.11% of LUTs, 7.72% of flip-flops, 10.37% of BRAM36 blocks, and no DSP slices. Of the 14 utilized BRAM36 blocks, 4 blocks implemented the 16 kB TT payload buffer, 1 block stored the 256-entry schedule table, 6 blocks supported the dual-port MAC and clock-domain-crossing FIFOs, and 3 blocks buffered PCF and UDP/ARP records. These results indicate unused on-chip resources under the evaluated configuration, although they do not by themselves establish hardware scalability for substantially larger traffic workloads. The post-implementation resource utilization, timing summary, and module latencies are listed in Table 6.

Table 6. Post-implementation FPGA resource, timing, and module latency results.

Item	Used	Available	Result	Source
Slice LUTs	8946	63,400	14.11%	Post-route report
Slice flip-flops	9782	126,800	7.72%	Post-route report
Block RAM, BRAM36	14	135	10.37%	Post-route report
DSP48E1 slices	0	240	0.00%	Post-route report
125 MHz main clock	8.00 ns	–	WNS = +0.74 ns	Static timing report
200 MHz RGMII clock	5.00 ns	–	WNS = +0.42 ns	Static timing report
PCF parsing latency	8 cycles	–	64 ns at 125 MHz	RTL sim and ILA
Permanence latency	6 cycles	–	48 ns at 125 MHz	RTL sim and ILA
Compression latency	14 cycles	–	112 ns at 125 MHz	RTL sim and ILA
PCF TX prep latency	10 cycles	–	80 ns at 125 MHz	RTL sim and ILA
Schedule table memory	256 entries	1 BRAM36	13 entries active	RTL configuration
TT payload buffer	16 kB	4 BRAM36	5 flows buffered	RTL configuration
Included top modules	All active	–	PCF, TT, UDP/MAC, RGMII, Mirror	Top-level netlist

The PCF parsing, permanence, compression, and transmit-preparation modules were implemented as deterministic synchronous pipelines in the 125 MHz domain, while the control path maintained the synchronization state, local cycle counter, timeout checks, and clock correction commands. The module latencies in Table 6 denote internal module-boundary processing durations derived from fixed RTL pipeline stages and verified through post-implementation timing simulation and on-chip logic analysis, rather than end-to-end wire-to-wire packet latency. Regarding hardware scaling, the pre-allocated 256-entry BRAM table and sequential pointer comparator allow active schedule entries to increase from 13 to 256 without additional table memory or lookup cycles. However, accommodating more concurrent TT flows in hardware requires expanding the payload buffers, queue descriptors, and arbitration logic. A synthesis check expanding the payload buffer to 64 kB for 64 flows increased BRAM36 utilization to 19.26% and LUT utilization to 18.64% with a +0.58 ns slack at 125 MHz. Nevertheless, live hardware transmission in Section 5.5 was validated only for the 13-entry schedule, whereas the 20–200-flow evaluations in Section 5.6 characterize host-side schedule generation only.

5.4. Clock Synchronization Results

This section reports two distinct synchronization measurements: the internal PCF timestamp-interval residual, which evaluates integration-period stability relative to the

0.4 ms nominal interval, and the compensated external clock-output skew, which evaluates physical edge alignment between the two independently clocked FPGA boards. Figure 13 illustrates the cold-start sequence captured via the mirrored monitoring path before and after synchronization convergence. Before FPGA2 was enabled, SM1 on FPGA1 transmitted Cold-Start frames with EtherType 0x891D and PCF Type 0x2 without receiving acknowledgments. Once CM on FPGA2 entered operation, Cold-Start Acknowledgment frames with PCF Type 0x4 were exchanged, and both boards transitioned into synchronized operation with periodic Integration frames of PCF Type 0x8 every 0.4 ms. Figure 14 shows the decoded fields of representative Integration frames during steady operation, where the incrementing integration_cycle field, stable membership bitmap, and 8 ns-resolution LocalClk field confirm continuous closed-loop clock correction.

No.	Time	Source	Destination	Protocol	Length	Type	Local Clk	Integration Cycle	Info
4505	35.857324000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	coldstart frame	0x000000000000d980	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4506	35.865292000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	coldstart frame	0x000000000000d98a	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4507	35.873291000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	coldstart frame	0x000000000000d994	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4508	35.881292000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	coldstart frame	0x000000000000d99e	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4509	35.888723000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	0x1	0x0000000000000011	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4510	35.889240000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	coldstart frame	0x000000000000d9a8	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4511	35.889644000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	coldstart frame	0x000000000000001b	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4512	35.897669000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	coldstart frame	0x0000000000000025	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4513	35.897669000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	coldstart ack frame	0x000000000000d9b7	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4514	35.898082000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	coldstart ack frame	0x0000000000000034	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4515	35.906501000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	coldstart ack frame	0x0000000000000043	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4516	35.906501000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4517	35.906501000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34d	0x00000000	Sync Domain: 0x01 Sync Priority: 0x01
4518	35.914046000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00000001	Sync Domain: 0x01 Sync Priority: 0x01
4519	35.914447000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34a	0x00000001	Sync Domain: 0x01 Sync Priority: 0x01
4520	35.922045000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00000002	Sync Domain: 0x01 Sync Priority: 0x01
4521	35.922443000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34d	0x00000002	Sync Domain: 0x01 Sync Priority: 0x01
4522	35.930043000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00000003	Sync Domain: 0x01 Sync Priority: 0x01
4523	35.930440000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c350	0x00000003	Sync Domain: 0x01 Sync Priority: 0x01
4524	35.938047000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00000004	Sync Domain: 0x01 Sync Priority: 0x01
4525	35.938485000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34e	0x00000004	Sync Domain: 0x01 Sync Priority: 0x01
4526	35.946042000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00000005	Sync Domain: 0x01 Sync Priority: 0x01
4527	35.946436000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34c	0x00000005	Sync Domain: 0x01 Sync Priority: 0x01

Figure 13. PCF cold-start sequence showing CS, CA, and IN frame exchanges.

No.	Time	Source	Destination	Protocol	Length	Type	Local Clk	Integration Cycle	Info
16969	85.664956000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x0000184d	Sync Domain: 0x01 Sync Priority: 0x01
16970	85.665251000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34d	0x0000184d	Sync Domain: 0x01 Sync Priority: 0x01
16971	85.672821000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x0000184e	Sync Domain: 0x01 Sync Priority: 0x01
16972	85.673475000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34d	0x0000184e	Sync Domain: 0x01 Sync Priority: 0x01
16973	85.680990000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x0000184f	Sync Domain: 0x01 Sync Priority: 0x01
16974	85.681249000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34e	0x0000184f	Sync Domain: 0x01 Sync Priority: 0x01
16975	85.689250000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00001850	Sync Domain: 0x01 Sync Priority: 0x01
16976	85.689250000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34a	0x00001850	Sync Domain: 0x01 Sync Priority: 0x01
16977	85.696809000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00001851	Sync Domain: 0x01 Sync Priority: 0x01
16978	85.697253000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34b	0x00001851	Sync Domain: 0x01 Sync Priority: 0x01
16979	85.705070000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00001852	Sync Domain: 0x01 Sync Priority: 0x01
16980	85.705220000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34f	0x00001852	Sync Domain: 0x01 Sync Priority: 0x01
16981	85.713078000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00001853	Sync Domain: 0x01 Sync Priority: 0x01
16982	85.713178000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34e	0x00001853	Sync Domain: 0x01 Sync Priority: 0x01
16983	85.721238000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00001854	Sync Domain: 0x01 Sync Priority: 0x01
16984	85.721238000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34e	0x00001854	Sync Domain: 0x01 Sync Priority: 0x01
16985	85.729044000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00001855	Sync Domain: 0x01 Sync Priority: 0x01
16986	85.729177000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34d	0x00001855	Sync Domain: 0x01 Sync Priority: 0x01
16987	85.736811000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00001856	Sync Domain: 0x01 Sync Priority: 0x01
16988	85.737626000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34e	0x00001856	Sync Domain: 0x01 Sync Priority: 0x01
16989	85.744798000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00001857	Sync Domain: 0x01 Sync Priority: 0x01
16990	85.745185000	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame	0x000000000000c34c	0x00001857	Sync Domain: 0x01 Sync Priority: 0x01
16991	85.752815000	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame	0x0000000000000000	0x00001858	Sync Domain: 0x01 Sync Priority: 0x01

Figure 14. Decoded integration-frame fields during steady synchronization.

Figure 15 shows the Local_Clk interval tracking profile of SM1 across observation windows near 1 s, 5 s, and 20 s after synchronization convergence, where the horizontal axis denotes the sequential sampling index and the vertical axis represents the scaled interval benchmark centered at 800 ns with deviations bounded within ±8 ns. This curve reflects interval stability across consecutive integration cycles rather than absolute inter-board phase offset. Figure 16 compares the synchronous tracking profiles of SM1, SC, and SM2 within the same observation cycle. Because SM1, SC, and SM2 were implemented inside FPGA1 and shared the same 125 MHz crystal oscillator, their close agreement verifies the internal consistency of the role-specific permanence and clock-correction logic within one

chip, whereas only the FPGA1–FPGA2 exchange and external oscilloscope test evaluate synchronization across independent board-level clock sources.

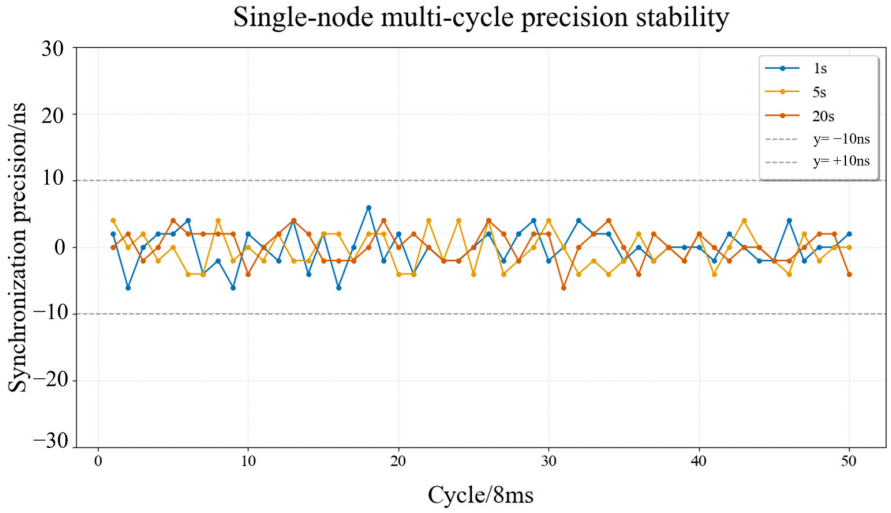


Figure 15. Timestamp-interval stability of SM1 across multiple observation windows.

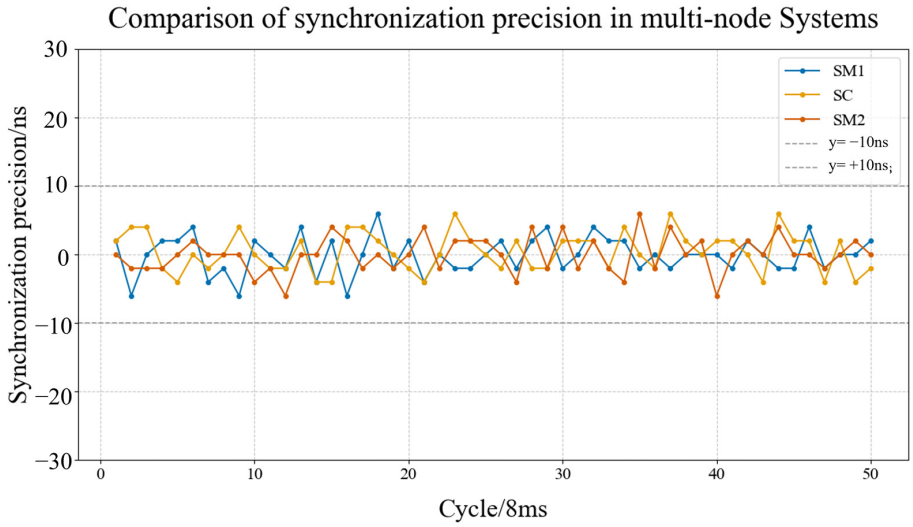


Figure 16. Intra-FPGA synchronization consistency among SM1, SC, and SM2 within the same cycle.

Table 7 summarizes the distributional statistics for both the internal PCF timestamp-interval residuals across 2.25×10^7 samples and the compensated external clock-output skew across 5.0×10^5 oscilloscope edge pairs over the five 30-min runs. For the inter-board FPGA1–FPGA2 exchange, the PCF timestamp-interval residual was centered near zero with a signed mean of −0.36 ns, a median absolute residual of 2.40 ns, a 95th percentile of 6.40 ns, a 99th percentile of 8.00 ns, a maximum absolute residual of 8.00 ns matching one 125 MHz clock tick, and a one-hour drift of 1.12 ns. The standard deviation of the signed mean across the five independent runs remained within 0.09 ns. Before probe-and-channel delay compensation, the raw external clock-output skew between FPGA1 and FPGA2 exhibited a mean of 0.89 ns and a maximum of 9.78 ns. After subtracting the calibrated 0.18 ns channel delay offset, the compensated skew achieved a signed mean of 0.71 ns, a standard deviation of 3.58 ns, a 95th percentile of 6.84 ns, a 99th percentile of 8.42 ns, and a maximum observed absolute skew of 9.60 ns. These results demonstrate stable_Synchronization for the evaluated two-board platform and test duration, although the observed maxima represent empirical bounds over finite sample sets rather than formal worst-case guarantees under arbitrary topologies or fault conditions.

Table 7. PCF timestamp-interval residual and compensated external clock-skew statistics.

Metric	FPGA1–FPGA2	SM1	SC	SM2	Ext. Skew
Valid samples	22,500,000	22,500,000	22,500,000	22,500,000	500,000
Signed mean/ns	−0.36	0.08	−0.04	0.12	0.71
Inter-run mean SD/ns	0.09	0.05	0.04	0.05	0.11
Mean abs. error/ns	2.88	2.24	2.16	2.32	2.84
Standard deviation/ns	3.46	2.97	2.86	3.02	3.58
Median abs. P50/ns	2.40	1.60	1.60	1.60	2.35
95th percentile P95/ns	6.40	5.60	5.60	5.60	6.84
99th percentile P99/ns	8.00	7.20	7.20	7.20	8.42
Max. abs. value/ns	8.00	8.00	8.00	8.00	9.60
One-hour drift/ns	1.12	0.64	0.56	0.72	1.25

Note: Columns 2–5 report internal PCF timestamp-interval deviations from the 0.4 ms period rather than absolute inter-board phase offsets. Ext. Skew reports the compensated physical inter-board clock skew measured by oscilloscope.

5.5. Scheduling and Traffic Transmission Results

The five periodic TT flows defined in Table 3 were expanded into 13 instances across the 32 ms matrix cycle, comprising four 8 ms basic cycles labeled Cycle 0 to Cycle 3. Specifically, TT1 with a 16 ms period generated 2 instances, TT2 with an 8 ms period generated 4 instances, TT3 with an 8 ms period generated 4 instances, TT4 with a 16 ms period generated 2 instances, and TT5 with a 32 ms period generated 1 instance. Figure 17 illustrates the logical scheduling topology, where SM1, SC, and SM2 on FPGA1 transmit critical traffic to CM on FPGA2 through two logical aggregation channels SW1 and SW2 sharing the physical Port 2 link. Figure 18 and Table 8 present the generated 13-instance schedule table. In the hardware register encoding shown in Figure 18, logical channels SW1 and SW2 are denoted by internal channel identifiers CH4 and CH5, respectively, and each scheduled slot occupies 200 µs within the 1–3 ms TT window. As shown in Table 8, SW1 carries 6 instances with an accumulated duration of 1280 µs, and SW2 carries 7 instances with an accumulated duration of 1320 µs, yielding a logical-channel load-balance rate of 96.97% and a combined physical Port 2 TT-window utilization of 32.50%. Table 9 summarizes the independent validation results, confirming zero missing or duplicate instances and zero release, deadline, conflict, guard, or boundary violations, while all six fault-injected invalid test tables were rejected.

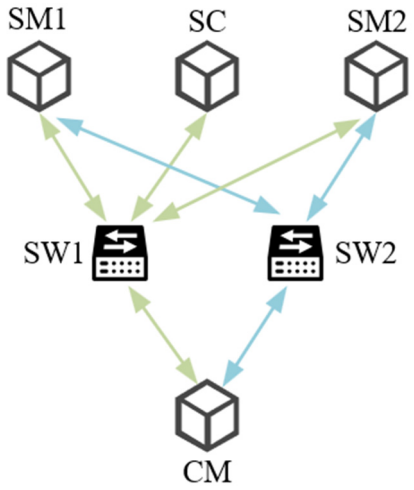


Figure 17. Logical scheduling topology with two aggregation channels.

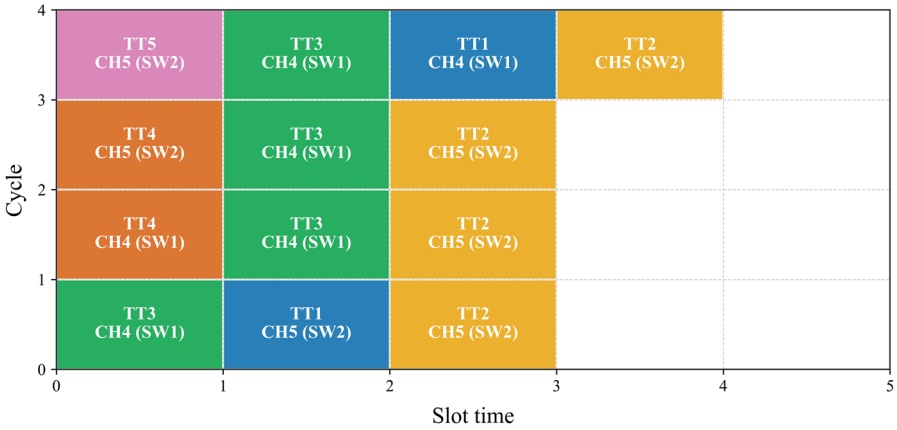


Figure 18. TT flow and logical-channel assignments across four basic cycles.

Table 8. Generated 13-instance schedule table and logical-channel occupation over one 32 ms matrix cycle.

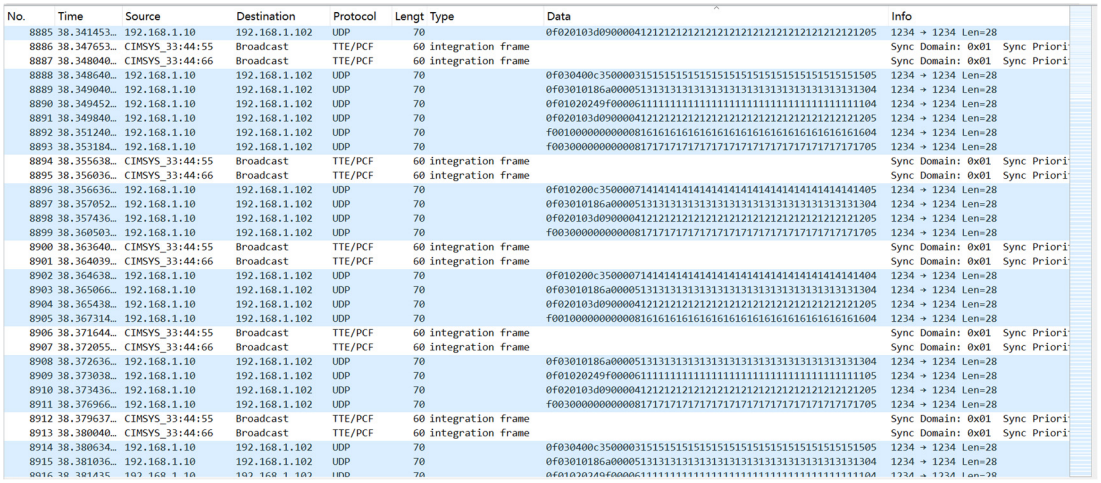
Cycle	Slot Index	Instance	Flow	Channel ID	Release/ms	Window/ms	Duration/µs	Deadline/ms
0	0	$J_{3,0}$	TT3	CH4, SW1	0.00	1.00–1.20	200	4.00
0	1	$J_{1,0}$	TT1	CH5, SW2	0.00	1.40–1.64	240	5.00
0	2	$J_{2,0}$	TT2	CH5, SW2	0.00	1.80–1.96	160	4.00
1	0	$J_{4,0}$	TT4	CH4, SW1	8.00	9.04–9.32	280	13.00
1	1	$J_{3,1}$	TT3	CH4, SW1	8.00	9.40–9.60	200	12.00
1	2	$J_{2,1}$	TT2	CH5, SW2	8.00	9.80–9.96	160	12.00
2	0	$J_{4,1}$	TT4	CH5, SW2	16.00	17.00–17.28	280	21.00
2	1	$J_{3,2}$	TT3	CH4, SW1	16.00	17.40–17.60	200	20.00
2	2	$J_{2,2}$	TT2	CH5, SW2	16.00	17.80–17.96	160	20.00
3	0	$J_{5,0}$	TT5	CH5, SW2	24.00	25.00–25.16	160	30.00
3	1	$J_{3,3}$	TT3	CH4, SW1	24.00	25.40–25.60	200	28.00
3	2	$J_{1,1}$	TT1	CH4, SW1	24.00	25.80–26.00	200	29.00
3	3	$J_{2,3}$	TT2	CH5, SW2	24.00	26.20–26.36	160	28.00
Summary	–	13 total	TT1–TT5	SW1: 6, SW2: 7	–	Shared Port 2	SW1: 1280, SW2: 1320	LBR = 96.97%

Table 9. Independent schedule validation and fault-rejection results.

Validation Item	Validated Schedule	Fault-Injection Test
Matrix and basic cycle	32 ms and 8 ms	Verified against flow LCM and GCD
Expected and allocated instances	13 and 13	12-instance table rejected
Missing or duplicate instances	0 and 0	Duplicate $J_{2,0}$ table rejected
Release-time violations	0	Premature release table rejected
Deadline violations	0	Late completion table rejected
Physical Port 2 conflicts	0	Overlapping SW1/SW2 slot rejected
Guard-interval violations	0	Unprotected boundary table rejected
Matrix-cycle boundary conflicts	0	Cross-cycle wrap conflict rejected

The validated schedule table was loaded into FPGA1 for hardware execution. Figure 19 shows a representative Wireshark capture from the mirrored Port 1 interface during mixed-traffic transmission, where broadcast TTE/PCF integration frames of 60 bytes alternate with scheduled UDP frames of 70 bytes carrying 28-byte application payloads from 192.168.1.10 to 192.168.1.102. The leading hexadecimal payload bytes encode the traffic class, flow identifier, and assigned logical channel. Figure 20 reconstructs the actual transmission slots across four consecutive 8 ms basic cycles on Port 2. In each basic cycle, two TTE/PCF frames are transmitted near 0.02 ms and 0.42 ms within the 0–1 ms synchronization window, scheduled UDP-TT frames occupy the 1–3 ms window, and non-critical UDP-BE frames

are admitted only within the 3–8 ms window without crossing the 8 ms guard boundary. Inside each colored block of Figure 20, the upper digit identifies the flow index, where 1 to 5 denote TT1 to TT5 and 6 to 7 denote background UDP-BE flows, while the lower digit identifies the logical channel, where 4 and 5 denote SW1 and SW2, matching Figure 18 and Table 8.



No.	Time	Source	Destination	Protocol	Lengt	Type	Data	Info
8885	38.341453…	192.168.1.10	192.168.1.102	UDP	70		0f020103d0900004121212121212121212121212121212121212121205	1234 → 1234 Len=28
8886	38.347653…	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame		Sync Domain: 0x01 Sync Priori
8887	38.348040…	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame		Sync Domain: 0x01 Sync Priori
8888	38.348640…	192.168.1.10	192.168.1.102	UDP	70		0f030400c3500003151515151515151515151515151515151515151505	1234 → 1234 Len=28
8889	38.349040…	192.168.1.10	192.168.1.102	UDP	70		0f03010186a0000513131313131313131313131313131313131313304	1234 → 1234 Len=28
8890	38.349452…	192.168.1.10	192.168.1.102	UDP	70		0f01020249f0000611111111111111111111111111111111111111104	1234 → 1234 Len=28
8891	38.349840…	192.168.1.10	192.168.1.102	UDP	70		0f020103d0900004121212121212121212121212121212121212121205	1234 → 1234 Len=28
8892	38.351240…	192.168.1.10	192.168.1.102	UDP	70		f0010000000000081616161616161616161616161616161616161604	1234 → 1234 Len=28
8893	38.353184…	192.168.1.10	192.168.1.102	UDP	70		f0030000000000081717171717171717171717171717171717171705	1234 → 1234 Len=28
8894	38.355638…	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame		Sync Domain: 0x01 Sync Priori
8895	38.356036…	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame		Sync Domain: 0x01 Sync Priori
8896	38.356636…	192.168.1.10	192.168.1.102	UDP	70		0f010200c3500007141414141414141414141414141414141414141405	1234 → 1234 Len=28
8897	38.357052…	192.168.1.10	192.168.1.102	UDP	70		0f03010186a0000513131313131313131313131313131313131313304	1234 → 1234 Len=28
8898	38.357436…	192.168.1.10	192.168.1.102	UDP	70		0f020103d0900004121212121212121212121212121212121212121205	1234 → 1234 Len=28
8899	38.360503…	192.168.1.10	192.168.1.102	UDP	70		f0030000000000081717171717171717171717171717171717171705	1234 → 1234 Len=28
8900	38.363640…	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame		Sync Domain: 0x01 Sync Priori
8901	38.364039…	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame		Sync Domain: 0x01 Sync Priori
8902	38.364638…	192.168.1.10	192.168.1.102	UDP	70		0f010200c3500007141414141414141414141414141414141414141404	1234 → 1234 Len=28
8903	38.365066…	192.168.1.10	192.168.1.102	UDP	70		0f03010186a0000513131313131313131313131313131313131313304	1234 → 1234 Len=28
8904	38.365438…	192.168.1.10	192.168.1.102	UDP	70		0f020103d0900004121212121212121212121212121212121212121205	1234 → 1234 Len=28
8905	38.367314…	192.168.1.10	192.168.1.102	UDP	70		f0010000000000081616161616161616161616161616161616161604	1234 → 1234 Len=28
8906	38.371644…	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame		Sync Domain: 0x01 Sync Priori
8907	38.372055…	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame		Sync Domain: 0x01 Sync Priori
8908	38.372636…	192.168.1.10	192.168.1.102	UDP	70		0f03010186a0000513131313131313131313131313131313131313304	1234 → 1234 Len=28
8909	38.373038…	192.168.1.10	192.168.1.102	UDP	70		0f01020249f0000611111111111111111111111111111111111111105	1234 → 1234 Len=28
8910	38.373436…	192.168.1.10	192.168.1.102	UDP	70		0f020103d0900004121212121212121212121212121212121212121205	1234 → 1234 Len=28
8911	38.376966…	192.168.1.10	192.168.1.102	UDP	70		f0030000000000081717171717171717171717171717171717171705	1234 → 1234 Len=28
8912	38.379637…	CIMSYS_33:44:55	Broadcast	TTE/PCF	60	integration frame		Sync Domain: 0x01 Sync Priori
8913	38.380040…	CIMSYS_33:44:66	Broadcast	TTE/PCF	60	integration frame		Sync Domain: 0x01 Sync Priori
8914	38.380634…	192.168.1.10	192.168.1.102	UDP	70		0f030400c3500003151515151515151515151515151515151515151505	1234 → 1234 Len=28
8915	38.381036…	192.168.1.10	192.168.1.102	UDP	70		0f03010186a0000513131313131313131313131313131313131313304	1234 → 1234 Len=28

Figure 19. Wireshark packet capture—data of different protocols.

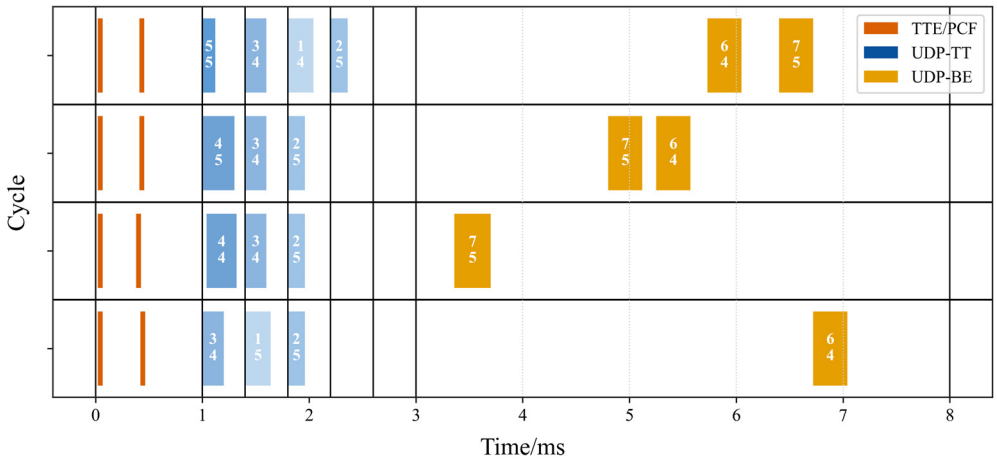


Figure 20. Reconstructed transmission slots for PCF, TT, and BE traffic.

Table 10 summarizes the flow-level hardware transmission statistics pooled across the five independent 30-min runs. Because each 30-min run spans 1800 s/0.032 s = 56,250 matrix cycles, the five runs cover 281,250 matrix cycles in total. Multiplying 281,250 matrix cycles by the instance counts $n_i \in \{2, 4, 4, 2, 1\}$ yields the exact packet counts in Table 10, totaling 3,656,250 transmitted TT packets. A packet q of flow f_i was counted as a deadline violation if $L_q > D_i$, where the relative deadlines D_i are 5.0, 4.0, 4.0, 5.0, and 6.0 ms for TT1 through TT5, respectively. Across all 3,656,250 packets, the aggregate mean latency was 2.45 ms with an inter-run mean standard deviation of 0.03 ms across the five runs, an aggregate standard deviation of 0.42 ms, and a maximum observed latency of 3.78 ms. Every flow maintained a positive safety margin of at least 0.42 ms below its deadline, and zero packet losses, duplicates, or deadline violations were observed. These results confirm deterministic execution for the evaluated five-flow configuration on the shared Port 2 link, representing empirical hardware verification rather than a formal worst-case bound for multi-hop networks.

Table 10. Flow-level hardware transmission results.

Flow	Sent Packets	Received Packets	Mean Latency/ms	Std. Dev./ms	P99 Latency/ms	Maximum Observed Latency/ms	Deadline Violations	Packet Loss	Duplicates
TT1	562,500	562,500	2.20	0.24	2.74	3.12	0	0	0
TT2	1,125,000	1,125,000	2.80	0.31	3.41	3.58	0	0	0
TT3	1,125,000	1,125,000	2.36	0.29	3.01	3.35	0	0	0
TT4	562,500	562,500	1.88	0.21	2.39	2.66	0	0	0
TT5	281,250	281,250	2.96	0.34	3.57	3.78	0	0	0

Figures 21–23 illustrate the multi-cycle packet statistics over a 50-basic-cycle observation window. Figure 21 shows the per-cycle packet quantity stacked by traffic class, where TTE/PCF remains fixed at 2 frames per basic cycle, UDP-TT alternates deterministically in a 3-3-4-3 pattern summing to 13 frames every four basic cycles, and UDP-BE fluctuates between 0 and 2 frames depending on residual window arrivals. Figure 22 shows the corresponding packet-count proportions over the 50 basic cycles, where UDP-TT accounts for 50.8% with 160 frames, TTE/PCF accounts for 31.3% with 99 frames, and UDP-BE accounts for 17.9% with 56 frames. Figure 23 plots the cycle-wise average TT latency across the 50 basic cycles, which fluctuates between 1.36 ms and 3.35 ms around the 2.45 ms aggregate mean according to the scheduled release offsets within each basic cycle.

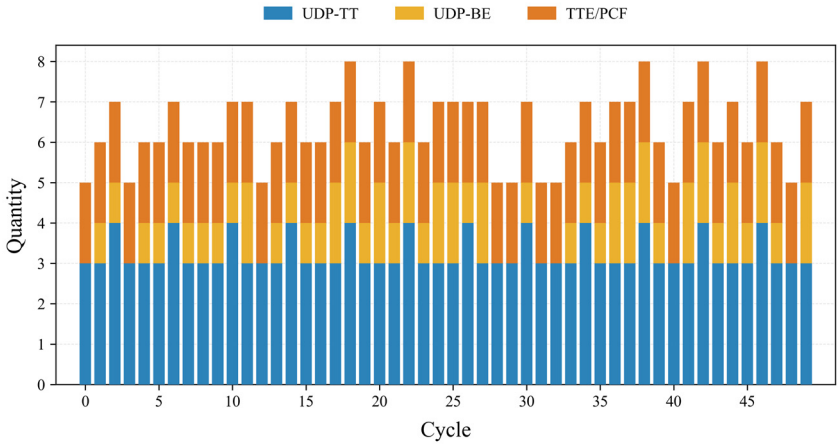


Figure 21. Per-cycle packet counts of TT, BE, and PCF traffic.

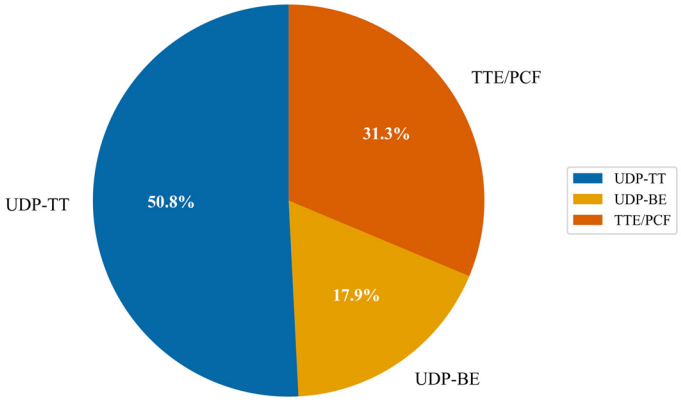


Figure 22. Packet-count proportions of TT, BE, and PCF traffic.

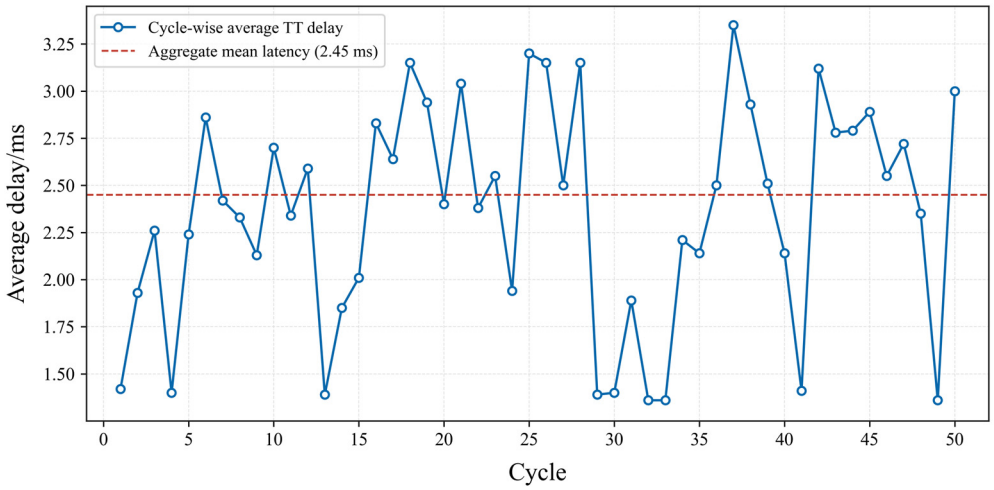


Figure 23. Cycle-wise average latency of scheduled TT traffic.

5.6. Comparative Evaluation, Scalability, and Discussion

The EDF, Fast Maximum Matching (FMM), LB, and EDF+LB methods were compared with the None transmission reference using the same five TT flows and physical platform. The release phases, frame lengths, admissible logical channels, 8 ms basic cycle, 32 ms matrix cycle, protocol windows, guard times, and shared Port 2 resource calendar were held constant.

None released eligible TT frames in FIFO order at the earliest available opportunity within the TT window, subject to PCF priority and physical-resource exclusivity. Logical channels followed a fixed admissible mapping. It generated no offline schedule. EDF used the instance ordering of Section 4 and selected the earliest timing-feasible candidate, with channel identifier resolving ties. LB processed instances by release time and flow identifier and selected the candidate with the smallest resulting duration-weighted load spread, followed by earliest completion and channel identifier. EDF+LB combined EDF instance ordering with the same load-aware candidate selection. All three offline methods applied identical release, deadline, window, guard, and physical-resource checks.

FMM was an FMM-inspired assignment heuristic implemented for comparison. In each basic cycle, the TT window was divided into five non-overlapping 0.4 ms cells. The bipartite graph connected unscheduled instances to cells in which their complete reservations satisfied release times, deadlines, channel reachability, and guard requirements. SW1 and SW2 mapped to the same cell vertex because they shared Port 2. Maximum-cardinality matching selected conflict-free assignments; traversal followed flow identifier, cell time, and channel identifier. Unmatched instances were reconsidered in later eligible cycles. This baseline maximized assignment count without an explicit deadline-priority or load-balancing objective. It was evaluated only on the direct-link topology and did not test multi-hop precedence.

Each method underwent five independent 30-min hardware runs using the timestamp definition in Section 5.2, with 3,656,250 TT packets evaluated per method. Mean latency, SD, P99, and maximum latency were calculated from pooled packet samples. The deadline violation rate was the number of packets satisfying $L_q > D_i$ divided by 3,656,250. Logical-channel LBR was calculated from accumulated reserved duration over complete matrix cycles. Offline generation time included schedule construction and independent validation and was averaged over 100 repetitions. None's online time was measured per basic-cycle selection batch and was reported separately. Host-side timing used a single process on an Intel Core i7-10700 CPU at a nominal 2.90 GHz with 16 GB DDR4 RAM, Ubuntu 20.04 LTS, and Python 3.8.

Table 11 and Figure 24 distinguish the effects of deadline ordering and logical-channel assignment. EDF produced the lowest mean latency, while LB and EDF+LB achieved the highest LBR. EDF+LB increased mean latency by 0.34 ms relative to EDF while increasing LBR from 35.42% to 96.97%. The results support a configuration-specific trade-off between latency and logical-channel balance. Because SW1 and SW2 share Port 2, the higher LBR does not indicate additional physical-link capacity.

Table 11. Comparative TT latency, logical-channel balance, and processing time.

Method	Mean/ms	SD/ms	P99/ms	Max/ms	Logical LBR	Offline/ms	Online/ms
None	2.64	0.46	3.66	3.96	35.42%	N/A	0.12
EDF	2.11	0.38	3.18	3.55	35.42%	0.84	N/A
FMM	2.38	0.41	3.47	3.78	75.68%	1.06	N/A
LB	2.61	0.45	3.72	3.98	96.97%	0.91	N/A
EDF+LB	2.45	0.42	3.46	3.78	96.97%	1.28	N/A

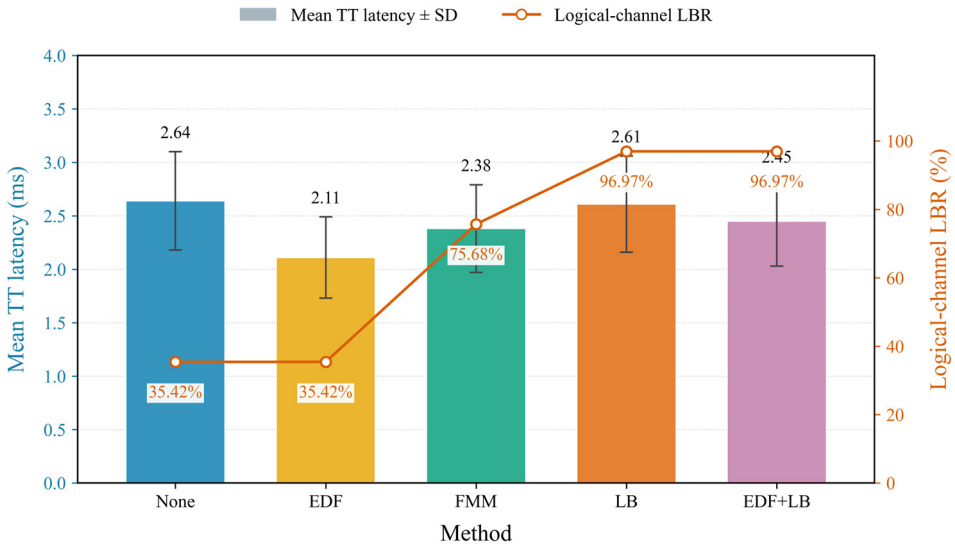


Figure 24. Comparison of average delay and load-balancing rate.

Host-side scalability was evaluated with 20, 50, 100, and 200 flows, using 1000 traffic sets per setting and random seeds 2026001–2027000. Each set contained 40% of flows with an 8 ms period, 40% with a 16 ms period, and 20% with a 32 ms period, giving 2.6 instances per flow per matrix cycle. Frame lengths of 64, 256, 512, and 1518 bytes were sampled with equal probability; release phases were selected from 0, 0.4, and 0.8 ms, and relative deadlines from 2–6 ms in 1 ms increments. Both logical channels were admissible.

The larger sets targeted TT-window utilizations of 46.8%, 61.5%, 73.6%, and 84.2%, with individual sets sampled within one percentage point of each target. Reservation budgets were scaled to the target load while covering serialization, Ethernet overhead, and a 0.8 µs guard interval. These budgets represented configured resource reservations rather than Ethernet serialization times alone. Candidate positions were generated from window openings, release times, and reservation boundaries and rounded to the 8 ns time quantum.

As summarized in Table 12, schedule-generation time increased with instance count, while the feasible-schedule success rate decreased at higher offered utilization. A failed attempt indicated that the heuristic did not find a complete validated schedule and did not prove infeasibility. The 20–200-flow schedules were not loaded into the FPGA, whose implemented table capacity was 256 entries. Accordingly, these tests did not evaluate larger-table lookup latency, BRAM use, FIFO occupancy, MAC throughput, or timing closure.

Table 12. Host-side schedule-generation scalability with increasing TT-flow counts.

TT Flows	Instances/MC	Traffic Sets	Failed	Mean TT Use	Mean Time/ms	Success	Mean Logical LBR
5	13	Fixed set	0	32.5%	1.28	N/A	96.97%
20	52	1000	0	46.8%	7.64	100.0%	94.21%
50	130	1000	13	61.5%	28.37	98.7%	92.84%
100	260	1000	54	73.6%	96.82	94.6%	90.16%
200	520	1000	121	84.2%	318.45	87.9%	87.03%

6. Conclusions

This paper presented an FPGA-based Time-Triggered Ethernet interface that integrates SAE AS6802 synchronization mechanisms with schedule-driven IP/UDP packet transmission. By decoupling synchronization control from packet data processing while referencing a shared 125 MHz local time base, the hardware architecture coordinates PCF parsing, permanence and compression calculations, clock correction, and time-triggered frame release within a unified pipeline. On the scheduling level, Earliest Deadline First instance ordering is combined with transmission-duration-weighted load balancing across timing-feasible candidate slots, preserving deadline urgency while equalizing channel occupation.

The system was validated on two independently clocked Xilinx Artix-7 FPGA boards connected by a direct 1 Gb/s link. Post-implementation results confirmed timing closure at 125 MHz and 200 MHz with moderate resource utilization. Over 22.5 million PCF samples, the timestamp-interval residual remained within ± 8.00 ns with a mean absolute error of 2.88 ns, while oscilloscope measurements across 500,000 rising-edge pairs showed a compensated inter-board clock skew of 0.71 ± 3.58 ns and a maximum of 9.60 ns. For the five-flow, 13-instance workload over a 32 ms matrix cycle, hardware tests across 3,656,250 TT packets achieved a mean latency of 2.45 ms, a maximum latency of 3.78 ms, and zero packet losses or deadline violations. Comparative evaluations showed that the proposed scheduler improved the logical-channel load-balance rate from 35.42% under pure EDF to 96.97%, while host-side scalability tests evaluated schedule generation up to 200 flows.

The present study has clear boundaries. It implements selected AS6802 synchronization procedures and a custom UDP-based TT data path rather than full standard certification, PHY-pin timestamping, frame preemption, or fault-recovery mechanisms. Moreover, because the two logical aggregation channels share a single physical egress port on the current testbed, the reported load-balance rate reflects logical-channel distribution rather than multi-link physical balancing, and workloads of 20 to 200 flows were evaluated only in host software. Future work will extend the platform to multi-hop switched topologies with independent physical links, hardware fault injection, online schedule reconfiguration, and formal worst-case latency analysis.

Author Contributions: Conceptualization, Y.W. and X.Z.; methodology, Y.W.; software, Y.W.; validation, Y.W.; formal analysis, Y.W.; investigation, Y.W.; resources, X.Z. and L.G.; data curation, Y.W. and X.Z.; writing—original draft preparation, Y.W. and L.G.; writing—review and editing, Y.W. and X.Z.; visualization, Y.W. and X.Z.; supervision, Y.W.; project administration, Y.W.; funding acquisition, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (Grant No. 52575556), the Beijing Outstanding Young Scientist Program, the Fundamental Research Funds for the Central Universities, and the Beijing Natural Science Foundation (Grant No. 24JL002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ARP	Address Resolution Protocol
BC	Basic Cycle
BE	Best-Effort
BRAM	Block Random-Access Memory
CA	Cold-Start Acknowledgment
CM	Compression Master
CP	Constraint Programming
CS	Cold Start
EDF	Earliest Deadline First
FIFO	First-In First-Out
FMM	Fast Maximum Matching
FPGA	Field-Programmable Gate Array
GCL	Gate Control List
GMII	Gigabit Media-Independent Interface
IFG	Interframe Gap
IN	Integration Frame
LBR	Load-Balance Rate
LUT	Look-Up Table
MAC	Media Access Control
MAE	Mean Absolute Error
MC	Matrix Cycle
PCF	Protocol Control Frame
PHY	Physical Layer
PLL	Phase-Locked Loop
RC	Rate-Constrained
RGMII	Reduced Gigabit Media-Independent Interface
RTL	Register-Transfer Level
SC	Synchronization Client
SFD	Start-of-Frame Delimiter
SM	Synchronization Master
TSN	Time-Sensitive Networking
TT	Time-Triggered
TTE	Time-Triggered Ethernet
UDP	User Datagram Protocol
WNS	Worst Negative Slack

References

1. Finn, N.; Thubert, P.; Varga, B.; Farkas, J. *Deterministic Networking Architecture*; RFC 8655; RFC Editor: Fremont, CA, USA, 2019. [CrossRef]
2. *IEEE Std 802.1AS-2020*; IEEE Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications. IEEE: New York, NY, USA, 2020. [CrossRef]
3. *IEEE Std 802.1Qbv-2015*; IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic. IEEE: New York, NY, USA, 2016. [CrossRef]
4. Nasrallah, A.; Thyagaturu, A.S.; Alharbi, Z.; Wang, C.; Shao, X.; Reisslein, M.; ElBakoury, H. Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 88–145. [CrossRef]
5. *SAE Standard AS6802*; Time-Triggered Ethernet. SAE International: Warrendale, PA, USA, 2011. [CrossRef]

6. Liu, C.L.; Layland, J.W. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM* **1973**, *20*, 46–61. [CrossRef]
7. Akram, B.O.; Noordin, N.K.; Hashim, F.; Rasid, M.F.A.; Salman, M.I.; Abdulghani, A.M. Joint scheduling and routing optimization for deterministic hybrid traffic in time-sensitive networks using constraint programming. *IEEE Access* **2023**, *11*, 142764–142779. [CrossRef]
8. Chen, H.; Wang, L.; Shen, P.; Di, J. Static schedule generation for time-triggered Ethernet based on fuzzy particle swarm optimization. *Chin. J. Electron.* **2019**, *28*, 1250–1258. [CrossRef]
9. Li, Z.; Wan, H.; Pang, Z.; Chen, Q.; Deng, Y.; Zhao, X.; Gao, Y.; Song, X.; Gu, M. An enhanced reconfiguration for deterministic transmission in time-triggered networks. *IEEE/ACM Trans. Netw.* **2019**, *27*, 1124–1137. [CrossRef]
10. Einspieler, S.; Steinwender, B.; Elmenreich, W. Mixed-triggered communication with limited elastic slot boundaries. *Microprocess. Microsyst.* **2021**, *86*, 104323. [CrossRef]
11. Chen, C.-H.; Lin, M.-Y.; Shih, Y.-C.; Chen, C.-C. High-precision time synchronization chip design for industrial sensor and actuator network. *Microprocess. Microsyst.* **2022**, *91*, 104507. [CrossRef]
12. Chen, C.-H.; Lin, M.-Y.; Tew, W.-P. Wireless fieldbus networking with precision time synchronization for a low-power WSAN. *Microprocess. Microsyst.* **2022**, *90*, 104509. [CrossRef]
13. Ruh, J.; Steiner, W.; Fohler, G. Clock synchronization in virtualized distributed real-time systems using IEEE 802.1AS and ACRN. *IEEE Access* **2021**, *9*, 126075–126094. [CrossRef]
14. Kopetz, H.; Ademaj, A.; Grillinger, P.; Steinhammer, K. The time-triggered Ethernet (TTE) design. In *Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05), Seattle, WA, USA, 18–20 May 2005*; IEEE: Piscataway, NJ, USA, 2005; pp. 22–33. [CrossRef]
15. Steiner, W.; Bauer, G.; Hall, B.; Paulitsch, M.; Varadarajan, S. TTEthernet dataflow concept. In *Proceedings of the Eighth IEEE International Symposium on Network Computing and Applications, Cambridge, MA, USA, 9–11 July 2009*; IEEE: Piscataway, NJ, USA, 2009; pp. 319–322. [CrossRef]
16. Steinhammer, K.; Grillinger, P.; Ademaj, A.; Kopetz, H. A time-triggered Ethernet switch. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, Munich, Germany, 6–10 March 2006*; IEEE: Piscataway, NJ, USA, 2006; pp. 794–799. [CrossRef]
17. Yan, L.; Luo, Z.; Tang, X.; Kong, Y. Timing analysis of rate-constrained traffic in TTEthernet using extended trajectory approach. In *Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 1039–1042. [CrossRef]
18. *IEEE Std 1588-2019*; IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. IEEE: New York, NY, USA, 2020. [CrossRef]
19. Liu, S.; Hamdulla, A. The issues on time synchronization technology in time triggered Ethernet. *J. Phys. Conf. Ser.* **2020**, *1673*, 012015. [CrossRef]
20. Wei, A.; Zhang, G.; Zhang, T. Research on time triggered Ethernet scheduling planning method. *J. Phys. Conf. Ser.* **2020**, *1518*, 012007. [CrossRef]
21. Yang, Y.; Zhai, Q.; Zhou, Y.; Mao, Y.; Hou, H. A heuristic scheduling algorithm for time-trigger flows in time-sensitive networking. In *Proceedings of the 2023 42nd Chinese Control Conference (CCC), Tianjin, China, 24–26 July 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 1927–1932. [CrossRef]
22. Guo, M.; Shou, G.; Liu, Y.; Hu, Y. Software-defined time-sensitive networking for cross-domain deterministic transmission. *Electronics* **2024**, *13*, 1246. [CrossRef]
23. Zheng, L.; Wei, G.; Zhang, K.; Chu, H. Traffic classification and packet scheduling strategy with deadline constraints for input-queued switches in time-sensitive networking. *Electronics* **2024**, *13*, 629. [CrossRef]
24. Gao, Q.; Li, J.; Liu, Y.; Xing, Y. Optimal control method for congestion control and delay reduction in deterministic networks. *IEEE Access* **2023**, *11*, 82912–82923. [CrossRef]
25. Wang, H.; Tan, S.; Zhu, Y.; Li, M. Deterministic scheduling with optimization of average transmission delays in industrial wireless sensor networks. *IEEE Access* **2020**, *8*, 18852–18862. [CrossRef]
26. Zhou, Y.; Feng, L.; Jiang, X.; Li, W.; Zhou, F. Predictable wireless networked scheduling for bridging hybrid time-sensitive and real-time services. *IEEE Trans. Commun.* **2024**, *72*, 3664–3680. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.