Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2017.DOI*

# Hardware design and Evaluation of an FPGA-based Network Switch Supporting Asynchronous Traffic Shaping for Time Sensitive Networking

**AKRAM BEN AHMED, TAKAHIRO HIROFUCHI, AND TAKAAKI FUKAI, (Member, IEEE)**
Digital Architecture Research Center, National Institute of Advanced Industrial Sciences and Technology, 2-3-26 Aomi, Koto-ku, Tokyo, Japan

Corresponding author: Akram BEN AHMED (e-mail: akram.benahmed@aist.go.jp).

This paper is based on results obtained from the project, "Research and Development Project of the Enhanced infrastructures for Post-5G Information and Communication Systems" (JPNP20017), commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

## ABSTRACT

Time Sensitive Networking (TSN) has been widely adopted to respond to the growing need for reliable and latency-bound communication in 5G/Post5G applications. It introduces a set of new open standards to conventional IEEE 802.3 Ethernet networks that aim to provide deterministic, reliable, high-bandwidth, and low-latency communication. Although many TSN-compliant switches have been proposed in the industry, only a few academic research works have proposed a comprehensive, reconfigurable, and open-source hardware switch architecture supporting deterministic transmission in data networks. In this paper, we present a reconfigurable hardware architecture of an FPGA-based network switch supporting TSN. The proposed switch leverages Asynchronous Traffic Shaping (ATS) as its traffic shaping algorithm to forward frames in a per-flow interleaved transmission. We present the key architectural components and implementation aspects of the proposed switch and discuss the evaluation results in a fair amount of detail to validate our proposal.

**INDEX TERMS** Asynchronous Traffic Shaping, FPGA, Switch Architecture, Time Sensitive Networks

## I. INTRODUCTION

In the past years, technology has witnessed a rapid and unparalleled increase in the number of connected components via networks. This growth has become more intensive with the emergence of new paradigms such as the Internet-of-Things, Edge Computing, and 5G communication. Consequently, the volume of data to be shared and exchanged has been experiencing an exponential increase. As new functionalities are continually introduced, the demand for communication capabilities also escalates. This trajectory is anticipated to persist, particularly as reliance on critical applications like autonomous driving, cyber-physical systems, and industrial automation intensifies.

Within such applications, the necessity for time-critical traffic management with ultra-low latency (ULL) on the scale of milliseconds or less has become paramount [1] [2]. Latency exceeding acceptable thresholds could precipitate compromised performance, potential hazards, or even catastrophic outcomes in terms of material damage or human safety. Therefore, the need for robust communication protocols capable of guaranteeing bounded low latency, minimal delay variation, and negligible data loss for time-sensitive and mission-critical traffic has never been more pronounced.

Addressing these evolving demands has led to the widespread adoption of Time Sensitive Networking (TSN) [3], an extension of conventional IEEE 802.3 Ethernet networks. TSN, defined by the IEEE 802.1 Time-Sensitive Networking Task Group, introduces additional standards and sub-protocols that offer deterministic guarantees, enhanced robustness, and integrated diagnostics and management services [4]. Certain TSN standards are gradually being integrated into the primary Ethernet Bridge standard (IEEE 802.1Q-2022) as they become essential requirements, even for standard Ethernet communication.

Traffic shaping and prioritization are key functions of a TSN switch that ensure adherence to the Quality of Service (QoS). They are responsible for controlling the timing of packet transmissions and giving transmission priority to critical packets over best-effort ones. In this fashion, a TSN flow does not violate predefined latency thresholds even in the presence of other flows in the network. TSN networks support various traffic shaping algorithms, also known as scheduling or queuing mechanisms. Each algorithm exhibits unique characteristics tailored to different QoS demands based on the deployed application. Thus, traffic shaping is a crucial design choice that has a significant impact on the configuration complexity and applications' QoS.

In this paper, we focus on Asynchronous Traffic Shaping (ATS) which offers deterministic latency as it operates asynchronously without the need for global clock synchronization in the network. Therefore, it provides deterministic latency with low implementation complexity and can utilize the bandwidth efficiently, even in the presence of high link utilization.

There has been a lot of work that tried to highlight the characteristics of ATS and other traffic shaping algorithms, from theoretical or simulation perspectives, and evaluate/compare their corresponding latency under different types of traffic. On the other hand, the hardware implementation of ATS on FPGA is almost nonexistent in the literature where ATS functionalities are implemented, evaluated, and the results are cross-validated against the theoretical results stipulated by TSN standards.

Starting from the facts stated above, we present in this paper a hardware implementation of an FPGA-based network switch supporting ATS. We offer the scientific community a chance to use and customize the proposed open-source switch [5] by detailing its key hardware components and algorithms in a fair amount of detail. Through our experiments, we evaluate the capabilities of the proposed switch in terms of traffic shaping, prioritization, and upper-bound latency conformity with the theoretical values stipulated by the TSN standard.

The rest of this paper is organized as follows. Section II gives a brief overview of the main characteristics of ATS. Section III discusses some of the related works to TSN evaluation and implementation. In Section IV, we introduce our FPGA-based switch supporting ATS. Section V is dedicated to the evaluation while in Section VI we validate the latency boundaries of the proposed switch with the TSN theoretical models. Finally, we end this paper with the conclusion in Section VII.

## II. ASYNCHRONOUS TRAFFIC SHAPING OVERVIEW

The Asynchronous Traffic Shaper (ATS) is a traffic shaping algorithm which is one of the fundamental features of TSN as it facilitates the transportation of both critical and non-critical traffic across a bridged network with shared resources, accommodating various QoS requirements to ensure latency bounds [3]. Before explaining how ATS works, it is important to first understand the key components of a given TSN architecture.

### A. TYPICAL TSN COMPOSITION

In a typical TSN network, the infrastructure comprises multiple bridges and end-stations, where an end-station can function as a talker (sender), a listener (receiver), or both. Data flows originating from a talker are termed TSN streams, traversing the network via bridges (switches). End-stations and their associated streams may have distinct Quality-of-Service (QoS) requirements, which are conveyed to bridges for resource reservation via Centralized User Configuration (CUC) and Centralized Network Configuration (CNC) (previously referred to as IEEE 802.1Qcc), or optionally through the Stream Reservation Protocol (previously referred to as IEEE 802.1Qat). Furthermore, each egress port in a bridge can accommodate up to eight output queues (referred to as Traffic Classes), each assigned to a specific priority level.

In addition to traffic shaping, the IEEE 802.1Qav standard, later merged to primary the Ethernet Bridge standard (IEEE 802.1Q-2022) [3], mandates the inclusion of an arbiter responsible for ensuring the proper forwarding of packets based on their traffic class priority. As previously discussed, traffic is divided into eight classes, with each class assigned to one of the eight egress queues depending on the priority indicated in the Priority Code Point (PCP) field within the VLAN tag of the frame. Consequently, queue priorities are strictly adhered to ensure that frames are selected from higher priority queues and forwarded over lower priority or best-effort traffic streams.

As previously mentioned, TSN networks support various traffic shaping algorithms. These algorithms include Strict Priority (SP), Credit Based Shaper (CBS), and Asynchronous Traffic Shaper (ATS), among others [3]. Each algorithm exhibits unique characteristics tailored to different QoS demands based on the deployed application. By using these traffic shaping algorithms, not only the latency of time-sensitive traffic can be reduced and guaranteed; but, also the stream arrival can be smoothed. As a result, the network congestion can be reduced. As previously indicated, we opted in this paper for ATS, and we proceed to outline its primary features and principles.

### B. ATS TRAFFIC SHAPING

ATS is a scheduling algorithm that prioritizes and schedules traffic through per-class queuing and per-stream reshaping. ATS is considered one of the most flexible scheduling algorithms capable of handling mixed traffic, including both periodic and sporadic traffic. In contrast to Credit Based Shaping, ATS can host and shape multiple flows in a traffic class. Consequently, it ably manages the combination of diverse traffic patterns, encompassing arbitrary periodic and sporadic traffic flows, with efficiency.

ATS functions by assessing the data rate of an incoming traffic stream and computing the transmission EligibilityTime for each frame. Upon reaching this EligibilityTime, the

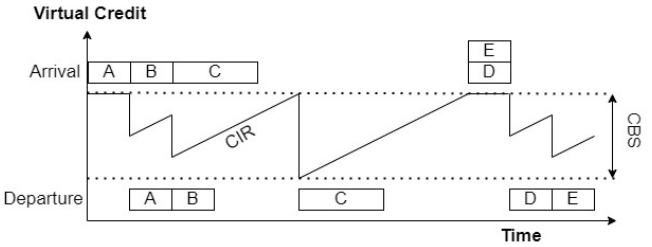



FIGURE 1: ATS Operation example. “CIR" and “CBS" stand for CommittedInformationRate and CommittedBurstSize, respectively.

frame gains permission for transmission; thereby, generating shaped output traffic. The calculation of EligibilityTime can be conducted autonomously for multiple streams. Nevertheless, given that frames from these various streams may utilize the same queue, their individual transmission times can be influenced by one another.

ATS incorporates two configurable parameters for each flow: the CommittedInformationRate (CIR) and the CommittedBurstSize (CBS). The CIR bears resemblance to Credit Based Shaper algorithm’s idleSlope parameter [3], as it dictates the average outgoing data rate to which traffic is confined. Conversely, the CBS permits back-to-back frame transmission within this size. Additionally, users can specify a Maximum Residence Time value. The shaper makes sure that packets spend less than this specified time in the queue by discarding packets that would exceed it.

In the IEEE standard terminology, an ATS Scheduler calculates the EligibilityTimes of frames of a single flow, which is a part of the state machine (i.e., an ATS Scheduler Group) that maintains the credits of the flows from the same input port and input priority. The ATS Transmission Selection Mechanism assigned for a traffic class sorts frames by their EligibilityTimes and dequeues frames upon their EligibilityTimes.

Figure 1 shows an example of how ATS works. Here, the assumed frame sizes are selected only for the sake of simplicity and easy understanding. (1) Initially, Frame A of $CBS/2$ frame size arrives to the input queue where it is set as eligible for transmission since the virtual credit $\geq$ frame’s size. Thus, it is delivered without further delay. (2) Later, the credit is increased by CIR while Frame B of similar $CBS/2$ frame size arrives. Similarly, it is forwarded as there is enough credit. (3) A larger Frame C (of $CBS$ frame size) arrives and since there is not enough credit, its delivery is delayed until the virtual credit reaches CBS. (4) The virtual credit is first reset to zero, it keeps increasing by CIR until reaching again the CBS, and it is maintained at this level due to the absence of new incoming frames. (5) Frames D and E arrive at the same time. Since there is enough credit for Frame D, it is forwarded first followed by Frame E.

## III. RELATED WORK

In this section, we highlight some of the works conducted so far that tried to tackle some of the challenges in TSN. As we demonstrate hereafter, only a few academic works have proposed the hardware design of TSN functionalities that makes the need for openly available reconfigurable FPGA-based designs important to improve TSN research.

### Surveys

Numerous works have undertaken surveys to explore TSN technologies and their applications in recent years. For example, *Seol et al.* [6] conducted a comprehensive general survey, categorizing and discussing various TSN technologies and methodologies. Additionally, other surveys have focused on the application of TSN in automotive and intelligent driving [7], as well as in smart factories [8]. Among these surveys, [6] includes a short passage where some of the simulation-based works focusing on ATS, or comparison with other scheduling algorithms, are introduced. In fact, there are only few works that mainly target the simulation or Implementation of ATS, but not enough works to summarize it in a survey article.

### Simulation-based

Apart from surveys, most of the Simulation-based studies proposed in the literature have focused on simulating different behaviors of TSN networks and the traffic they convey, rather than giving special attention to ATS. For instance, *Nasrallah et al.* [4] examined the comparison between TAS and ATS by performing several experiments on OMNET++. They observed that TAS can achieve ultra-short latencies under specific settings while ATS performs generally well compared to TAS for sporadic traffic. Similarly, using OMNET++, *Debnath et al.* [9] tried to evaluate various models of mixed TSN shaper architectures including CBS, TAS, and ATS. They showed each model’s strong and weak points, and they highlighted the potential benefits of combining several shapers together. Similarly, works by *Jiang et al.* [10] and *Falk et al.* [11] proposed simulation models for TSN using *OMNET++* to explore different TSN features.

Another portion of simulation-based research has focused on ATS and how to improve its performance. For example, *Zhou et al.* [12] tried to analyze ATS under different scheduling approaches by assessing the end-to-end delay, buffer usage, and frame loss rate using their in-house simulator. *Guo et al.* [13] proposed another variation of ATS, called Time Sensitive Queuing (TSQ), to ease the deployment of TSN networks and to reduce the buffer consumption. They showed the potential of their proposal through Network Calculus and the NS-3 simulator.

### Hardware-based

In the realm of hardware-based solutions, numerous industrial TSN switches and solutions have been developed. However, detailed implementation specifics are not openly provided. Conversely, academia has seen relatively few studies focusing on TSN hardware methodologies.

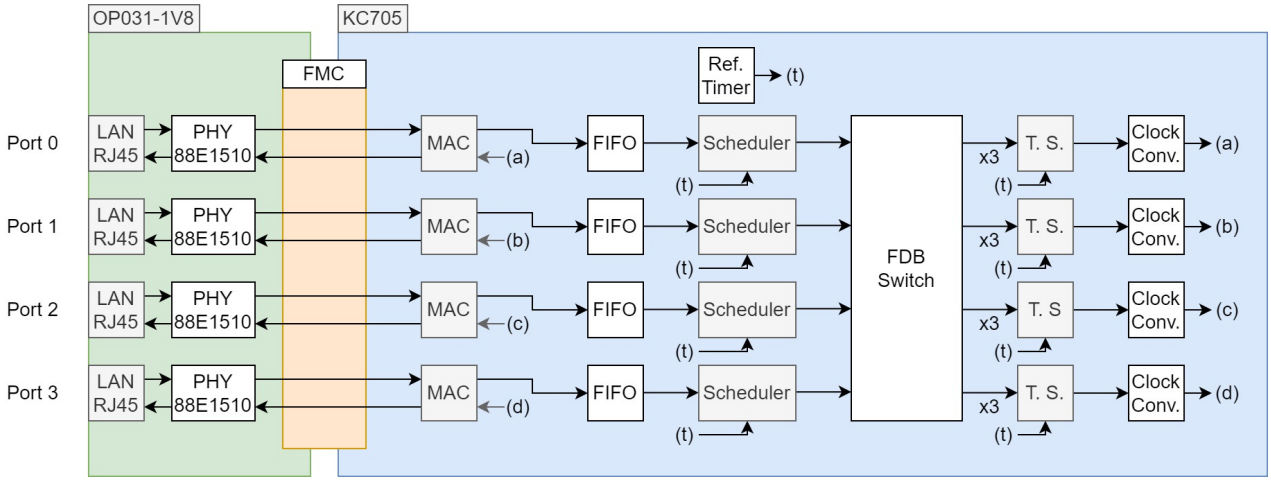



FIGURE 2: Proposed FPGA-based ATS-switch block diagram

Most of the works proposed so far have addressed the development of hardware-based testbeds for TSN deployment without a particular focus on ATS. For instance, *TSN-FlexTest* [14] builds a testbed using generic Commercial off-the-shelf (COTS) hardware and open-source software components to enable flexible TSN measurements and monitoring. It can provide TSN researchers with valuable insights regarding the behavior of different network stream patterns in a single TSN switch, without a comprehensive understanding of the entire network. *Rezabek et al.* [15] have proposed *En-GINE*, a flexible infrastructure that allows researchers to evaluate and monitor the different features and design parameters of Intra-Vehicular TSN Networks through reproducible and replicable experiments. To achieve their goal, they relied on TSN-capable Intel Network Interface Cards and virtual *Open vSwitch* [16]. In a similar COTS-based approach, *Do et al.* [17] gave special attention to ATS and focused on improving it by analyzing the performance potential of several shapers and their combinations. They first conducted a numerical analysis and then validated their calculations using Infineon's TC275 commercial switch and Vector's VN5620 as their End Systems. They showed that combining ATS with Length Ratio Quotient (LRQ) within a predefined threshold frame size has the most stable performance among the proposed approaches.

The above works may offer TSN researchers a certain degree of flexibility, reproducibility, and ease of use to illustrate different key insights into TSN performance. However, users cannot change the hardware architecture of the used COTS or freely modify some of their settings. TSN applications have a wide range of usage and require different needs and specifications for the used hardware, where each application may require a different set of TSN standards. Thus, a minimum degree of reconfigurability to tune the used hardware switch is required for a comprehensive TSN performance analysis.

When it comes to configurable switch architectures, very few academic research works have been proposed in the literature. *Zhou et al.* [18] proposed a partial and isolated implementation of an ATS scheduling entity without developing the entire switch. Thus, it is difficult to have a comprehensive idea about ATS behavior in real-case implementation without a full and reconfigurable switch design. *Quan et al.* [19] is probably the only work that also aspires to provide an open-source framework for TSN development. However, the proposed switch does not support ATS. In addition, the GitHub repository that they are providing does not specify any license. This makes reusing, modifying, and sharing their project problematic. Moreover, most of the provided materials are not written in English. These factors make the authors' open-source claims, unfortunately, obsolete.

## IV. PROPOSED FPGA-BASED ATS SWITCH HARDWARE IMPLEMENTATION

The proposed FPGA-based switch supports ATS capabilities for TSN and 1000BASE-T with a maximum designed speed of 1 Gbps. It supports two Ethernet frame lists: Ethernet II and IEEE 802.3ac (VLAN tag support). In addition, we set the Maximum Transmission Unit (MTU) to be 1500 bytes.

As illustrated in Figure 2, the main components of our switch are implemented on an AMD Kintex 7 FPGA KC705 Evaluation Kit [20]. Since KC705 has only a single Ethernet port, we attached to it an Opsero OP031-1V8 Ethernet FMC [21] via the "FMC HPC" connector, as can be seen in Figure 2. This card is an add-on/expansion board that hosts four Marvell 88E151x Gigabit Ethernet PHYs to provide four ports of gigabit Ethernet connectivity to the KC705 board. It is important to mention that our proposed design is not limited to the current implementation on KC705 and that it can be easily customized to other implementation on other FPGA boards.

### A. MAC BLOCK

Incoming streams from a specific Ethernet port are initially directed to the MAC block, which facilitates data bridging between the PHY (utilizing the RGMII interface) and the

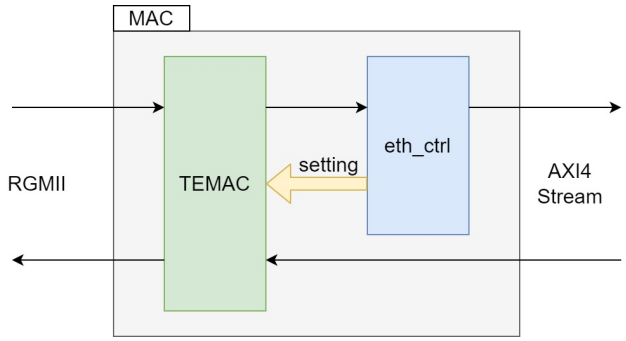



FIGURE 3: MAC block diagram

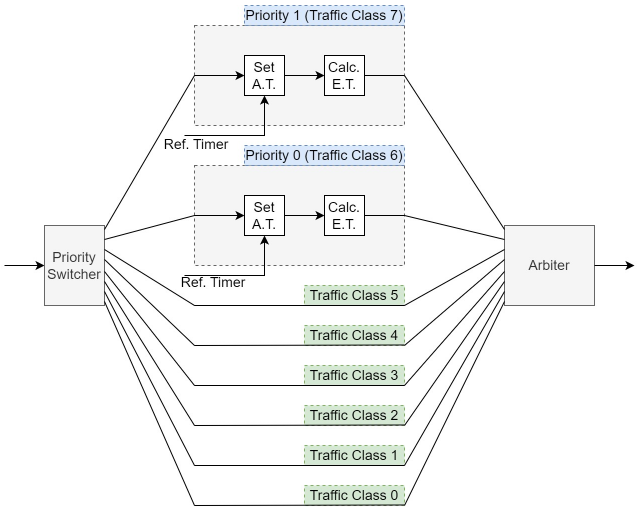



FIGURE 4: Scheduler block diagram

FPGA programmable logic section via an AXI4-Stream bus. Within the MAC block, incoming streams undergo processing where the Preamble, Start Frame Delimiter (SFD), and Frame Check Sequence (FCS) fields are removed from the input Ethernet frames. Conversely, these fields are added for outgoing frames. As depicted in Figure 3, the MAC block comprises the Tri-Mode Ethernet Media Access Controller (TEMAC) module, which is an official soft IP from AMD/Xilinx for the Ethernet MAC layer [22] (not embedded in the PS section), alongside the “eth_ctrl", a modified sample code of the TEMAC IP designed to support four-channel functionality. It is worth mentioning that as of writing, AMD provides 1G TEMAC IP free of charge for 120 days, and its project license without that restriction can be purchased, if necessary.

### B. PRE-SWITCH FIFO BLOCK

Frames are later stored in the input FIFO blocks, as shown in Figure 2 before going to the “Scheduler" block. These FIFO blocks eventually serve for storing purposes, as well as for clock conversion between the MAC side and FPGA core side. Both domains work at 125MHz clock frequency. However, they have different clock sources; therefore, synchronization is needed. The FIFO blocks are configured to the “Packet Mode". When setting a giving FIFO to this mode, it delays the start of frame (burst) forwarding to the next module until the end (LAST beat) of the frame is received. Therefore, latency increases depending on the length of the stream (it is the same as the frame length) to gain more transmission reliability. However, this setting can be removed if latency reduction is preferred.

TABLE 1: PCP vs Priority assignment.

| PCP in VLAN tag | Priority (Traffic Class) |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 2 | 6 |
| 3 | 7 |
| 4 | 2 |
| 5 | 3 |
| 6 | 4 |
| 7 | 5 |
| No VLAN tag | 5 |

### C. SCHEDULER

As represented in Figure 4, this block starts by dividing input frames according to their priority and switching between them in accordance to their traffic class. As mentioned in Section II-A, the traffic is divided into eight classes, with each class assigned to one of the eight egress queues. The priority is dictated by the PCP field in accordance with the TSN standard, as described in Section II-A. The adopted priority assignment is illustrated in Table 1. In this table, the higher the Traffic Class index is, the higher the priority.

The proposed switch prototype assumes that ATS is used for Traffic Classes 6 and 7. Therefore, frames within these two classes are forwarded to the “Set Arrival Time" and “Calculate Eligibility Time" blocks, explained hereafter, while frames below Traffic Class 5 are skipped to the “Arbiter" where Ethernet frames with higher priority are output first.

#### 1) Set Arrival Time block

This block, depicted by “Set A.T." in Figure 4, is responsible for appending the Arrival Time field to the tail of the Ethernet frame internally. As input, “Set A.T." has the reference time (t) from the “Ref. Timer" block which generates a reference timer count for each module in the proposed switch. In the “Set A.T." block, when the first beat of the Ethernet frame passes, the input (t) value is set to a 9 bytes Arrival Time field and attached to the end of the Ethernet frame, as shown in Figure 5. This means it is set when the ‘tlast’ signal of the AXI4-Stream input is asserted.

#### 2) Calculate Eligibility Time block

Represented by “Calc. E.T." in Figure 4, this block fetches the Arrival Time field appended in the received frame from the previous “Set A.T." block to calculate the EligibilityTime.

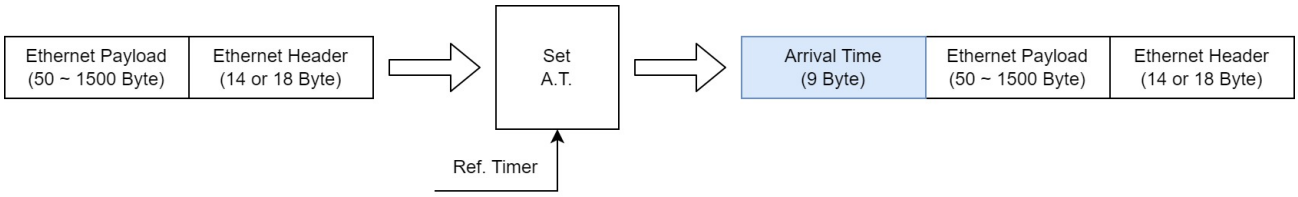



FIGURE 5: Set Arrival Time block behavior


FIGURE 6: Calculate EligibilityTime block behavior

The calculation is based on the `ProcessFrame(frame)` procedure stipulated by the IEEE 802.1Q-2022 standard, representing the state machine explained in Algorithm 1 [3] where:

- `Length(frame)` is the size of the frame, including all media-dependent overhead.
- `BucketEmptyTime` is the state variable of each ATS scheduler which preserves what corresponds to the virtual credit of each flow. In the IEEE standard, the virtual credit is maintained in the time domain.
- `ArrivalTime(frame)` refers to the instant of time at which the arrival of the entire frame is detected, as explained in Figure 5.
- `GroupEligibilityTime` represents a variable that contains the most recent value of the EligibilityTime variable from the previous frame, as processed by any ATS scheduler instance in the same ATS scheduler group.
- `AssignAndProceed(frame, EligibilityTime)` calculates an assigned eligibility time parameter (assignedEligibilityTime) that is used by the ATS transmission selection algorithm.
- `Discard(frame)` discards the frame and increases the corresponding counter in the associated reception port.

As demonstrated in Figure 6, the "Calc. E.T." block calculates the EligibilityTime then appends it to the frame tail as a 9 bytes field to replace the no-longer needed Arrival Time. In addition, the shaper makes sure that frames do not exceed their residency in the queue over a specified time using the Maximum Residence Time value. As shown in Algorithm 1, if $EligibilityTime > ArrivalTime + MaxResidenceTime$, the relevant Ethernet frame is discarded in this block and no longer propagated to subsequent blocks.

It is worth noting that the flow identification has not been formally detailed in the IEEE802.1Q-2022 standard. In our prototype, it is done using IPv4 Src and Dst addresses and port of the Input Ethernet frame embedded in the IP Header, partially based on the IEEE802.1CB-2017 standard [23].

**Algorithm 1** ProcessFrame (frame)

**LengthRecoveryDuration** =
Length(frame)/ CommittedInformationRate;
**EmptyToFullDuration** =
CommittedBurstSize/ CommittedInformationRate;
**SchedulerEligibilityTime** =
BucketEmptyTime + LengthRecoveryDuration;
**BucketFullTime** =
BucketEmptyTime + EmptyToFullDuration;
**EligibilityTime** = MAX(ArrivalTime(frame),
GroupEligibilityTime, SchedulerEligibilityTime);

**if** (EligibilityTime $\leq$ ArrivalTime(frame) + MaxResidenceTime/1.0e9)
**then** `//Frame is valid`
  GroupEligibilityTime = EligibilityTime;
  **if** (EligibilityTime < BucketFullTime) **then**
    BucketEmptyTime = SchedulerEligibilityTime;
  **else**
    BucketEmptyTime = SchedulerEligibilityTime + EligibilityTime - BucketFullTime;
  **end if**
  AssignAndProceed(frame, EligibilityTime);
**else**`//Frame is invalid`
  Discard(frame);
**end if**

### D. FDB SWITCH BLOCK

The "FDB Switch" block functions as an L2 Switch with Filtering Database (FDB) capabilities. As depicted in Figure 7, the "FDB" submodule is located at the entrance of the block and is responsible for receiving frames from the four input ports, storing the MAC address information for each port, and directing Ethernet frames only to specific ports based on this stored information. It registers the Source MAC Address of the received Ethernet frame to the Database by linking it with the received port and then queries the Destination MAC Address of the received Ethernet frame to the Database. If it is registered in the Database, the destination port is decided

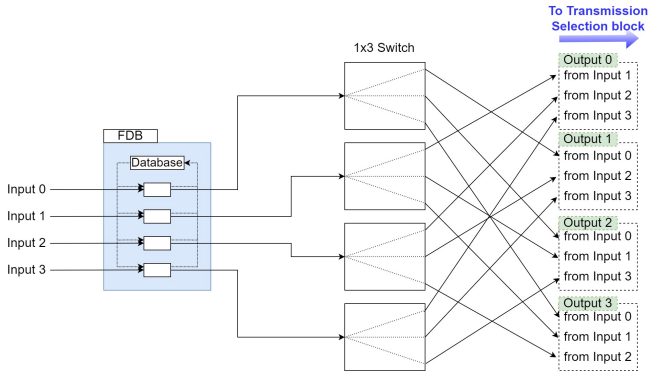



FIGURE 7: FDB Switch block diagram

to the linked port. Otherwise, it excludes the port from where the frame was received, copies the Ethernet frame, and sends it to the three remaining output ports. It is important to mention that this block just decides the destination port and does not switch the Ethernet frames.

The "1x3 Switch" block switches Ethernet frames based on the destination port information which is given in the "FDB" submodule. It is worth noting that the priority is the same for each input port in this switch. However, once it starts forwarding the Ethernet frame from a certain input, it will continue doing so until the end of the Ethernet frame.

For outputs, the "FDB Switch" block has a total of 12 outputs, with each set of 3 outputs (excluding the port from where the frame was received) is connected to one of the four "Transmission Selection" blocks representing the four physical Ethernet output ports, as explained hereafter.

### *E. TRANSMISSION SELECTION BLOCK*

The Transmission Selection block incorporates the ATS Transmission Selection Mechanism. As previously mentioned in Section II-B, the ATS Transmission Selection Mechanism assigned for a traffic class sorts frames by their EligibilityTimes and dequeues frames upon reaching their EligibilityTimes.

As illustrated in Figure 8, frames at each of the three inputs, coming from the "FDB Switch" block, are first switched to one of the eight Traffic Classes based on the Priority Code Point (PCP). Later, frames of Traffic Class 7 and 6 are stored in the "ATS Queue" which is a "Packet Mode" FIFO. Here, the output control mechanism and queues are independent for each input port and priority. On the other hand, frames of Traffic Class 5 and below are stored in a regular FIFO.

The "ATS E. T. Gate" manages the output of the Ethernet frame based on the comparison between the EligibilityTime field embedded in the frame and the (t) received from the "Reference Timer" block according to the following two cases:

- If $(t) < EligibilityTime$, the Ethernet frame is stored in a FIFO and its transmission is delayed.
- When $(t) \geq EligibilityTime$, the Ethernet frame is directly forwarded.

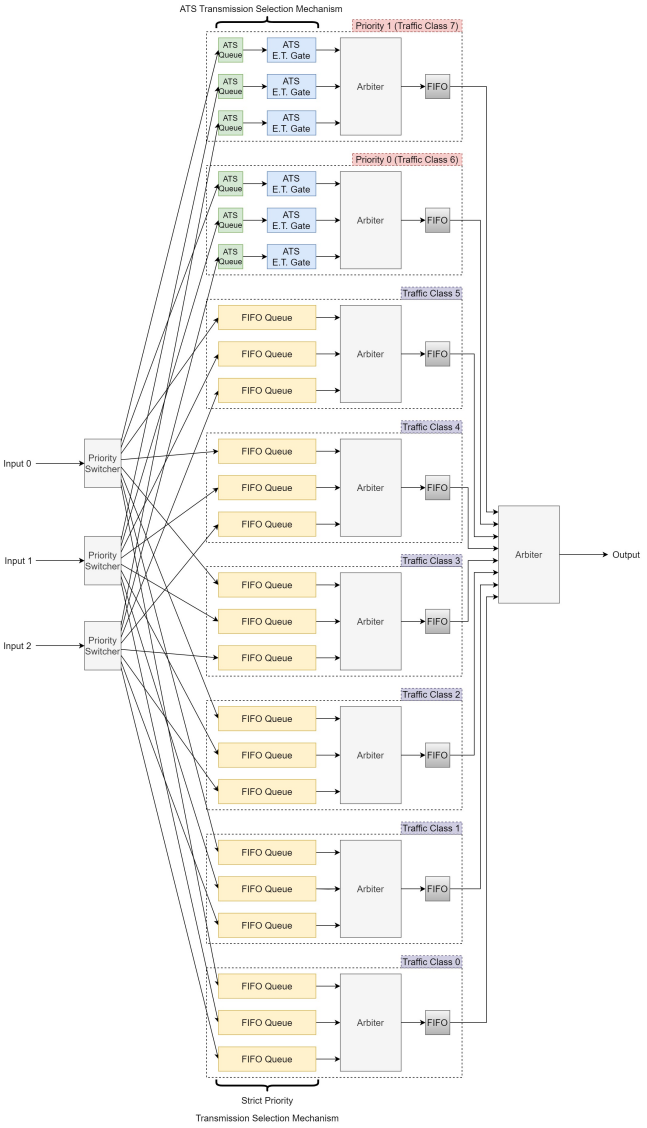



FIGURE 8: Transmission Selection block diagram

At the end of this comparison, the EligibilityTime field is removed from the output frame. Then, a first round of arbitration is performed between the output Ethernet frames of the same Traffic Class before being temporarily stored in another simple FIFO. Later, the second round of arbitration allows the Ethernet frame with higher priority to be output first. Finally, when leaving the "T. S." block, streams are forwarded for clock conversion before being forwarded back to the MAC blocks, as shown in Figure 2.

## V. EVALUATION

### *A. EXPERIMENT SETUP*

Figure 9 shows the overview of the setup of experiments. Host A has two GbE NICs (Intel i210) and Host B has a single one. Their ports are connected to our developed network switch, respectively. The link speeds of the connections between the ports were all 1 Gbps. To measure latency and

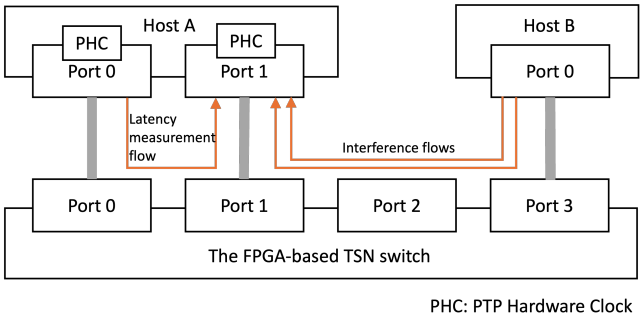



FIGURE 9: Experiment setup overview

frame intervals, we used the hardware timestamp mechanism (i.e., PTP hardware clock) of Intel i210 [24]. It records the transmit and receive timings of a frame. It has an 8-ns resolution. Due to synchronization accuracy between the PTP hardware clocks of Port 0 and Port 1 of Host A, the measured latency can be discussed in a 1-us precision in this setup.

A latency measurement program, `plget`, was used with our minor customization. We used the transmission rate of 100 frames per second for the flow of latency measurement and used the 1500-byte Ethernet payload. The bit rate of the flow was 1.2Mbps. It utilized 0.12% of the GbE link. Because Intel i210 NIC can record its transmission time only for one frame existing in its hardware queue, with a higher frame rate than it, the NIC fails to record the correct transmission times. This is a limitation in our latency measurement mechanism. It is important to mention that this is not a limitation in the proposed switch. We consider that the obtained results support that the latency bounding of the proposed switch correctly also works for the higher frame rate of a flow. As shown in Section V-B, the latency of a frame in each component of the switch is deterministic for the flow without contention. It depends on the size of a frame. As shown in Section VI, with contention, the observed latency matched to the theoretical model. These facts do not depend on the frame rate of the flow.

To obtain arrival frame intervals, `iperf` was used to generate flows. `tcpdump` with the hardware timestamp option enabled was used to capture the arrival times. Host A and B are equipped with an Intel i7-13700 CPU. The frequency of its CPU core was fixed at its maximum speed. Its temporal sleep mechanism (i.e., Intel C-state) was disabled.

When the payload size of all the Ethernet frames was set to 1500 bytes, the size of Ethernet frames at the physical layer was 1542 bytes including Inter-Frame Gap (12 bytes), Preamble (8 bytes), the Ethernet header of the VLAN support (18 bytes), and Frame Check Sequence (4 bytes).

### B. BASIC LATENCY OF AN ATS TRAFFIC CLASS

We used Integrated Logic Analyzer (ILA) to measure how many cycles are consumed in each component of the TSN switch. This information is necessary to determine the appropriate value for the ProcessingMaxDelay (PMD) parameter.

#### 1) ProcessingMaxDelay parameter Background

An ATS scheduler calculates the EligibilityTime of a frame according to the CommittedInformationRate (CIR), CommittedBurstSize (CBS), and other parameters. Then, the ATS Transmission Selection Mechanism transmits the frame upon reaching the EligibilityTime. A network switch supporting ATS has its implementation-specific delay between the time when the EligibilityTime is calculated and the time when the frame is queued to the ATS transmission selection. This delay is called the Processing Delay in the IEEE standard.

The IEEE standard stipulates that the maximum value of this delay needs to be added to the EligibilityTime. It is specified by the ProcessingMaxDelay parameter. If ATS needs to delay a frame to shape the flow (i.e., in the case that, since the previous frame was transmitted a few moments ago, the transmission of the current frame needs to wait so as to meet a CIR and a CBS), at the moment when the frame is queued at the ATS transmission selection, its EligibilityTime needs to be later than the current clock. Otherwise, the Transmission Selection mechanism sends the frame instantly and cannot insert the necessary delay to achieve the shaping rate. Thus, the assigned EligibilityTime must take into account how many cycles are necessary to reach the queue of the Transmission Selection in the worst case.

#### 2) Component latency measured by ILA

ATS was assigned to Traffic Class 7. The CIR of the used ATS scheduler was set to 100 Mbps and its CBS was set to 1542 Bytes. For comparison, Strict Priority was assigned to Traffic Class 5. For ILA analysis, a frame was sent from one NIC to the other NIC via the network switch. Since the frame was the first and only one, the ATS transmission selection did not intentionally delay the frame. The used cycles obtained by ILA are considered an inevitable base overhead.

Table 2 shows the used cycles to process one frame of different payload sizes for ATS. The EligibilityTime adjustment by PMD was disabled. The gray-shaded components are specific to ATS, and the others are common for any transmission selections. The clock of the FPGA board runs at 125 MHz; thus, one cycle corresponds to 8 ns. Table 3 shows used cycles for Strict Priority. The blue-shaded component is specific to Traffic Classes 5 and below in the current implementation. Despite repeating the experiments several times, we did not observe any deviations in the consumed cycles. There were no FPGA components whose consumed cycles fluctuated. The consumed cycles depended on the size of a frame.

Overall, the total latency of an ATS traffic class is higher than that of the Strict Priority. In the case of the 1500-byte payload, a frame via ATS consumes 97.8 us, while a frame via Strict Priority consumes 49.2 us. The mechanism of ATS is more complex than the simple FIFO queue of Strict Priority, which results in a longer latency. The “Calc. E.T." block and the “ATS Queue" module consumed 6080 cycles (i.e., 49 us). The reason of the priority switcher module in Strict Priority was slightly larger than that of ATS is presumably

TABLE 2: Latency of each component for a frame of an ATS traffic class with PMD set to 0 us, obtained by ILA

| Payload Size (Bytes) | Eth_Ctrl (Cycles) | Pre-Sw FIFO (Cycles) | Set A.T. (Cycles) | Calc. E.T. (Cycles) | FDB (Cycles) | ATS Queue + E.T. Gate (Cycles) | FIFO (Cycles) | Total (ns) |
|---|---|---|---|---|---|---|---|---|
| 64 | 156 | 73 | 1 | 174 | 99 | 162 | 3 | 5906 |
| 100 | 228 | 109 | 1 | 246 | 135 | 234 | 3 | 8210 |
| 300 | 628 | 309 | 1 | 646 | 335 | 634 | 3 | 21010 |
| 500 | 1028 | 509 | 1 | 1046 | 535 | 1034 | 3 | 33810 |
| 700 | 1428 | 709 | 1 | 1446 | 735 | 1434 | 3 | 46610 |
| 900 | 1828 | 909 | 1 | 1846 | 935 | 1834 | 3 | 59410 |
| 1100 | 2228 | 1109 | 1 | 2246 | 1135 | 2234 | 3 | 72210 |
| 1300 | 2628 | 1309 | 1 | 2646 | 1335 | 2634 | 3 | 85010 |
| 1500 | 3028 | 1509 | 1 | 3046 | 1535 | 3034 | 3 | 97810 |

TABLE 3: Latency of each component for a frame of a Strict Priority class, obtained by ILA

| Payload Size (Bytes) | Eth_Ctrl (Cycles) | Pre-Sw FIFO (Cycles) | Set A.T. (Cycles) | Calc. E.T. (Cycles) | FDB (Cycles) | FIFO Queue (Cycles) | FIFO (Cycles) | Total (ns) |
|---|---|---|---|---|---|---|---|---|
| 64 | 156 | 73 | 0 | 1 | 99 | 5 | 3 | 3258 |
| 100 | 228 | 109 | 0 | 1 | 135 | 5 | 3 | 4410 |
| 300 | 628 | 309 | 0 | 1 | 335 | 5 | 3 | 10810 |
| 500 | 1028 | 509 | 0 | 1 | 535 | 5 | 3 | 17210 |
| 700 | 1428 | 709 | 0 | 1 | 735 | 5 | 3 | 23610 |
| 900 | 1828 | 909 | 0 | 1 | 935 | 5 | 3 | 30010 |
| 1100 | 2228 | 1109 | 0 | 1 | 1135 | 5 | 3 | 36410 |
| 1300 | 2628 | 1309 | 0 | 1 | 1335 | 5 | 3 | 42810 |
| 1500 | 3028 | 1509 | 0 | 1 | 1535 | 5 | 3 | 49210 |

TABLE 4: Latency of each component for a frame of an ATS traffic class with PMD set to 50 us, obtained by ILA

| Payload Size (Bytes) | Eth_Ctrl (Cycles) | Pre-Sw FIFO (Cycles) | Set A.T. (Cycles) | Calc. E.T. (Cycles) | FDB (Cycles) | ATS Queue + E.T. Gate (Cycles) | FIFO (Cycles) | Total (ns) |
|---|---|---|---|---|---|---|---|---|
| 64 | 156 | 73 | 1 | 174 | 99 | 6047 | 3 | 52986 |
| 100 | 228 | 109 | 1 | 246 | 135 | 5975 | 3 | 54138 |
| 300 | 628 | 309 | 1 | 646 | 335 | 5575 | 3 | 60538 |
| 500 | 1028 | 509 | 1 | 1046 | 535 | 5175 | 3 | 66938 |
| 700 | 1428 | 709 | 1 | 1446 | 735 | 4775 | 3 | 73338 |
| 900 | 1828 | 909 | 1 | 1846 | 935 | 4375 | 3 | 79738 |
| 1100 | 2228 | 1109 | 1 | 2246 | 1135 | 3975 | 3 | 86138 |
| 1300 | 2628 | 1309 | 1 | 2646 | 1335 | 3575 | 3 | 92538 |
| 1500 | 3028 | 1509 | 1 | 3046 | 1535 | 3175 | 3 | 98938 |

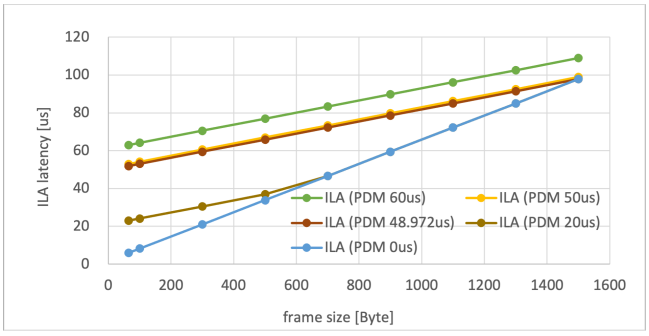



FIGURE 10: Total latency of the network switch for a frame of an ATS traffic class, obtained by ILA

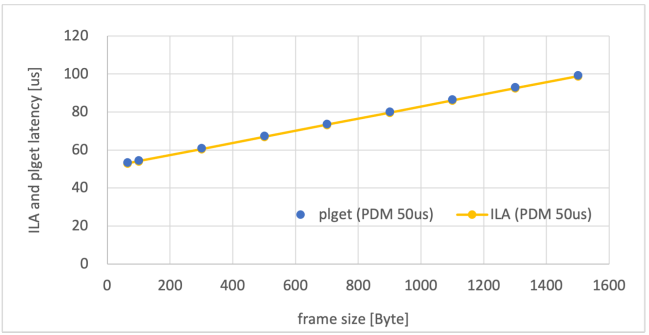



FIGURE 11: Average latency of the network switch for frames of an ATS traffic class, measured by `plget`

related to the fact that the priority of Traffic Class 5 is lower than that of Traffic Class 7. In several modules, the used cycles proportionally increased as the frame size increased. These modules internally use one (or more) packet-mode FIFO(s). It temporally buffers the entire bits of a frame.

The processing delay was caused in part by the “Calc. E.T." block and in another part by the “ATS Queue" module. Technically speaking, it is possible to directly measure the processing delay by ILA. However, it requires adding ILA probing points inside these modules, which involves addi-

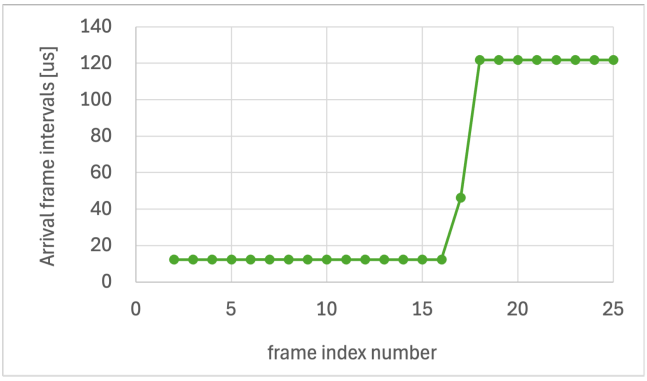



FIGURE 12: Arrival frame intervals with CommittedBurstSize set to $1542 * 16$ bytes

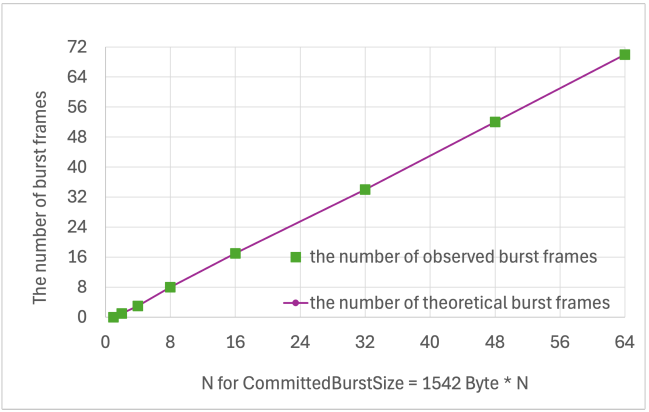



FIGURE 13: Number of observed burst frames with CommittedBurstSize set to $1542 * N$ bytes

tional complexities. Instead, we estimated its maximum value from the observed latency of a frame with different PMD configurations.

Figure 10 shows the total latency of the network switch for an ATS traffic class. For example, when PMD was set to 20 us, the frame of a 64-byte payload arrived at the transmission queue before its adjusted EligibilityTime, and then waited there until its EligibilityTime. Thus, the latency increased from the case where PMD was 0 us. The frame of a 500-byte or larger payload arrived after its adjusted EligibilityTime, and did not wait there; the latency was the same as the case where PMD was 0 us.

This way, we estimated the smallest PMD as 48.972 us, by which, regardless of the frame size, the adjusted EligibilityTime of a frame does not become past at the entrance of the transmission selection queue. [1] We chose 50 us for the default value of PMD in the current prototype. See Table 4 for the used cycles with this parameter value. PMD has an impact only on the used cycles in the “ATS Queue" module.

To validate the results obtained by ILA, we also measured the latency of the network switch from one NIC to another NIC of Host A by using `plget.plget` sent 100,000 frames at 100 frames per second for 100 seconds (i.e., 1.2 Mbps) As shown in Figure 11, the average of the measured latency was the same as that measured by ILA. The largest standard deviation was 218 ns with the payload size with 300 bytes. Its confidence interval of the 95% confidence level was 4.3 ns, which was sufficiently small to discuss latencies in a 1-us resolution. With the CIR set to 10 Mbps, we obtained the same measured latency. The CIR does not affect the latency when the rate of the flow is smaller than it. ATS does not delay the frames of the flow.

### C. BURST SIZE CONTROL

We tested the capability of ATS to control the burst size of a flow. In contrast to Credit Based Shaping, ATS allows burst transfer within CBS. The CIR of an ATS scheduler was set to 100 Mbps. For ease of discussion, its CBS was set to $1542*N$ bytes, where N is a positive integer. We sent 100 UDP frames at the maximum rate of 1GbE by `iperf` from one NIC (Port 0) to the other NIC (Port 1) of Host A via the ATS switch and captured the frames' arrival times at Port 1 of Host A by `tcpdump` with the hardware timestamp option enabled. For example, with the CBS of 1542 bytes, the last 100-th frame arrived at the destination approximately 12 ms after the first frame was transmitted at the source. It is expected that, up to the allowed burst size, ATS forwards frames without inserting a delay. After that, ATS delays frames to make frame intervals conforming to CIR.

Figure 12 shows the arrival frame intervals with CBS set to $1542 * 16$ bytes. We omit the first interval that was exceptionally high (approximately 1 ms) with any $N$ number. The transmission timings of the first two frames at the sender side are presumably unstable due to the behavior of the operating system. The next 16 intervals were exactly 12.336 us, which means that the network switch forwarded frames in a back-to-back manner without inserting any delays. The 18th and later intervals were 121.752 or 121.760 us. Since there was no remaining burstiness, the network switch started postponing frames to shape the flow to the CIR. As shown in Figure 13, for other CBS values, the observed burst frames matched the expected theoretical values.

### D. RATE CONTROL FOR A SINGLE FLOW

We confirm that our ATS implementation correctly shapes the rate of a single flow to the CIR of the ATS scheduler assigned for the flow. In a manner similar to the previous experiment, we sent a flow of UDP frames at the maximum rate of 1GbE by `iperf` for 10 seconds and captured the arrival times of frames; the total number of sent frames was approximately 800,000. The CBS was set to 1542 bytes and CIR was gradually changed. It is expected that ATS drops excessive frames and rectifies frame intervals; thereby, achieving the specified rate. Figure 14 shows that the av-

[1] In this case, the frame is the first frame and its EligibilityTime is calculated as its ArrivalTime according to the algorithm of the ATS scheduler state machine. This is the earliest possible EligibilityTime. 48.972 us is determined by the most severe case.

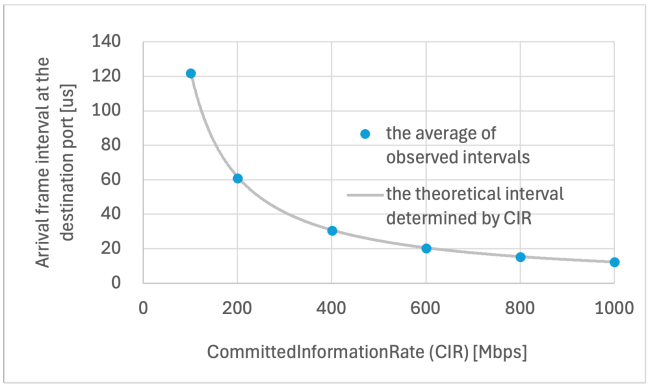



FIGURE 14: Arrival frame intervals of a flow shaped by ATS

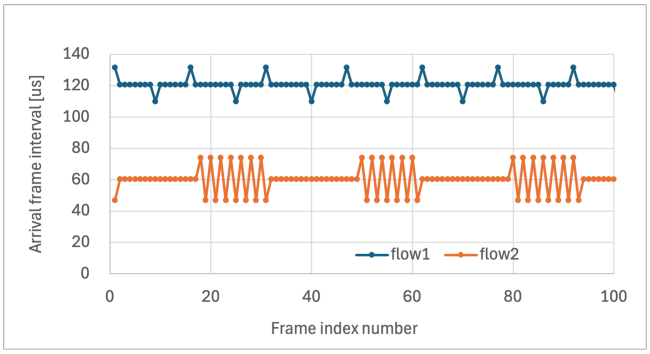



FIGURE 15: Arrival frame intervals of two flows shaped by an ATS Scheduler Group

erage of the observed arrival intervals exactly matched the theoretical ones determined by CIR. The standard deviation of the observed intervals was 3.1 ns to 5.6 ns, which was caused by the 8-ns resolution of the hardware timestamp. The confidence interval of the 95% confidence level was under 1 ns, which was sufficiently small for the discussion.

### *E. RATE CONTROL FOR MULTIPLE FLOWS*

ATS can shape multiple flows individually without implementing per-flow queues in the switch. This is an advantage over Credit Based Shapers, which need to implement per-class shaping. Even though a TSN switch uses all the traffic classes (i.e., eight traffic classes) for Credit Based Shapers, it can shape only eight distinct flows.

In the default adopted configuration, our ATS-capable TSN switch supports 16 flows per input port per input priority. In fact, one of the advantages of our implementation is that the necessary resource to implement one flow is only a few registers. Specifically, the ones to preserve its CBS and CIR parameters, and the one to maintain its `BucketEmptyTime` state value, previously explained in Algorithm 1. Consequently, it is possible to increase the supported flows without major concerns about the limit of available FPGA resources.

The state machine of an ATS Scheduler Group calculates the EligibilityTimes for the frames of the flows that arrive at the same input port and with the same input priority. As shown in Algorithm 1, the state machine determines to which flow an arriving frame belongs to, calculates its EligibilityTime with reference to the state and configuration values, and then repeats the same steps for the next frame. Even though multiple flows share the state machine (and indeed share the state variable of `GroupBucketEmptyTime` of the ATS Scheduler Group), this mechanism assures different CIRs and CBSs for individual flows. This mechanism is called the Interleaved Regulator. Although frames of multiple flows are interleaved in the state machine (i.e., the EligibilityTime of a frame in a flow likely affects that of the next frame that might be of another flow), the worst-case latency of each flow is bounded and not different from the case where per-flow queues are implemented. This will be discussed in detail later in Section VI.

To confirm the capability of the Interleaved Regulator, we generated two flows of UDP frames at 500 Mbps each. Both flows were sent from Port 0 of Host A simultaneously for 10 seconds; the total number of sent frames was approximately 400,000 for each flow. The input priority of the flows was the same in the switch. For each flow, the destination UDP port of frames was differently set. Accordingly, the matching conditions to detect flows were set in the configuration registers of the switch.

The CIR of one flow was set to 100 Mbps, and that of the other one was to 200 Mbps. Thus, the transmit rates of both flows exceeded the designated CIRs. The switch was expected to drop excessive frames and adjust frame intervals to meet their CIR. Their CBS were set to 1542 Bytes. The MaxResidenceTime of the ATS Scheduler Group was set to 134 us, which is the maximum allowed period of time between the arrival time of a frame and its EligibilityTime. If that period of a frame exceeds the MaxResidenceTime, the state machine drops the frame. Since the CIR of 100 Mbps needs a 123 us interval between two frames, the MaxResidenceTime of the state machine was set to a larger value than it. Otherwise, we observed that the switch dropped frames excessively, resulting in a smaller forwarding rate than the CIR.

Figure 15 shows the observed arrival frame intervals of the flows at the destination port via the network switch. The expected frame interval of the CIR of 100 Mbps is 123.0 us and that of 200 Mbps is 61.5 us. The averages of the observed frame intervals (i.e., 120.5 us and 60.3 us) were nearly the same as the theoretical values, which correspond to 102.4 Mbps and 204.6 Mbps, respectively.

The observed repeated fluctuations of frame intervals in both flows are $\pm$ 10 us in the flow of the CIR of 100 Mbps, and $\pm$ 13 us in the flow of 200 Mbps. This is an expected behavior of the Interleaved Regulator. For a flow whose transmit rate exceeds its CIR, ATS adjusts the frame intervals so as to meet the CIR on average, and only assures that its latency via the switch is under $MaxResidenceTime + PMD$. To

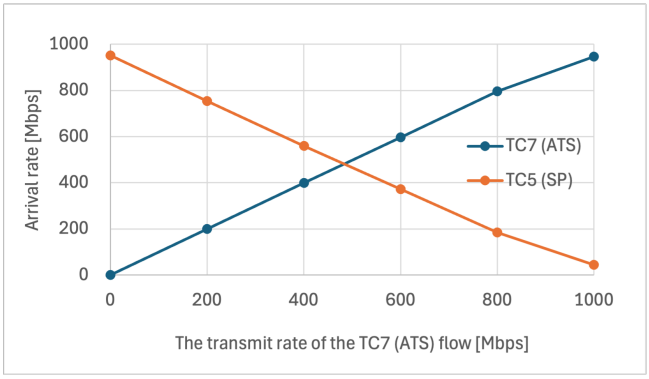



FIGURE 16: Arrival rates of the competing flows via the network switch

bound the latency of a flow, the transmit rate at its sender node needs to be equal or under the CIR agreed upon with all the switches on the path of the flow.

We confirmed that the ATS implementation shaped multiple flows arriving at one input port of the switch, respectively. Although not described here, we also confirmed that the Interleaved Regulator correctly worked for the simpler cases where only the transmit rate of one flow exceeded its CIR or those of the flows did not. The Interleaved Regulator of our switch works for more flows in the same manner, thanks to the deterministic behavior of FPGA.

### F. FIXED PRIORITY ARBITRATION

The IEEE standard states that a TSN switch needs to implement up to eight traffic classes, and a higher index of traffic class has strict precedence over a lower one. To confirm that this mechanism correctly works in the proposed ATS-capable switch, we generated two competing flows for 10 seconds. One was a flow of frames transmitted from Port 0 of Host A at different rates of ATS, and the other one was from Port 0 of Host B at 1 Gbps. With the frame rate of 1Gbps, approximately 800,000 frames were sent in 10 seconds. The switch forwarded both flows to Port 1 of Host A. At the egress port of the switch, the ATS Transmission Selection Mechanism of Traffic Class 7 was assigned to ATS and that of Traffic Class 5 was assigned to Strict Priority. The flow from Host A was treated as Traffic Class 7, and that of Host B was treated as Traffic Class 5. The CIR of the flow was set to 1000 Mbps.

Figure 16 shows the arrival rates of both flows at Port 1 of Host A. As the transmit rate of the ATS flow increased, its arrival rate at the destination increased accordingly, without any frame drops. The frames of the lower priority flow were forwarded only with the remaining bandwidth. The aggregate throughput of both flows was continuously 1 Gbps.

The TSN switch successfully prioritized the higher priority Traffic Class. Only when no frame of Traffic Class 7 was ready for transmission, it forwarded frames from Traffic Class 5. We also conducted additional experiments and confirmed that no matter what transmission selection mechanism (e.g., ATS, Credit-based shaping, Strict Priority) was assigned to a traffic class, this fixed priority arbitration mechanism correctly worked. This behavior is mandatory for the theoretical model to bound the latency of a priority flow.

### G. HARDWARE RESOURCE UTILIZATION

In this evaluation, we present the hardware complexity of the proposed FPGA-based switch. As previously stated, our switch is implemented on an AMD Xilinx Kintex 7 FPGA KC705 Evaluation Kit [20]. Therefore, the resource utilization results were obtained using the AMD Xilinx Vivado 2022.1 tool [25]. Table 5 showcases the resource utilization of each component of our switch as well as the percentage of the available resources.

As depicted in this table, a significant portion of the Logic Lookup Tables (LUTs) is utilized by the “FDB Switch" as it manages the necessary multiplexing. Similarly, due to the presence of two arbiters, the “Transmission Selection" block consumed a smaller portion of the LUTs. Conversely, Memory Lookup Tables (LUTRAMs) and BRAM blocks are primarily consumed by the “Pre-switch FIFOs" and “Transmission Selection" blocks. This distribution is natural since these blocks include buffers for storing processed frames before forwarding them downstream.

It can be seen from this table that the entire proposed switch consumed a reasonable portion of the LUTs, FFs, and LUTRAMs. However, in terms of BRAM blocks, the design is almost reaching its limit. Nonetheless, it is worth noting that the current implementation supports 16 flows per input port per input priority, and that increasing this number will not have a major impact on the BRAM utilization. Therefore, we believe that the obtained benefits with the current configuration within the utilized resources are very practical. Furthermore, the used KC705 evaluation board is among the most affordable and low-power FPGA development boards, featuring relatively limited resources compared to other mid- or top-tier commercially available counterparts. We assert that the current architecture enables the implementation of additional TSN features in larger boards while maintaining FPGA costs at a minimum.

## VI. LATENCY BOUNDS VALIDATION AND DISCUSSION

### A. CONSIDERATIONS

The Annex V of the IEEE standard [3] presents a framework to model the end-to-end delay along the path of an ATS flow. It should be noted that the Annex is marked as informative, which means that its description is not intended to be standard clauses that implementations need to obey. The end-to-end delay is the sum of per-hop delays, and a per-hop delay is the delay between two network entities (e.g., sender/receiver host machine, or switch) connected via a physical medium. We consider that the model of a per-hop delay in the Annex is unclear in part and is not consistent with at least our ATS implementation.

The Annex mentions the buffering delay caused by store and forward operations. However, there are no clear defi-

TABLE 5: Proposed switch hardware complexity

| Component | MAC | FIFO | Sched. | FDB Sw. | Trans. Selec. | Clk. Conv. | Others | Total | Utiliz. (%) |
|---|---|---|---|---|---|---|---|---|---|
| LUTs | 9832 | 5477 | 65879 | 17539 | 51736 | 267 | 5097 | 155827 | 76.46 |
| FFs | 15379 | 3664 | 50456 | 15015 | 41296 | 448 | 7396 | 133627 | 32.78 |
| LUTRAMs | 1501 | 4096 | 112 | 32 | 25776 | 32 | 212 | 31761 | 49.63 |
| BRAMs | 8 | 0 | 16 | 2 | 384 | 0 | 3 | 413 | 92.81 |

nitions of what store and forward operations actually stand for. We presume that, in their model, this delay accounts for the period of time to receive all the bits of a frame, i.e., $l/R$ where $l$ is the frame size and $R$ is the link speed. Although not clearly written in the paper [26], the original ATS model seems to assume that when the bits of a frame arrive at an Ethernet port, the switch waits for the last bit of the frame by buffering received bits, and after receiving the last bit, moves to the next step such as FDB lookup. This buffering happens only once in the switch.

We consider that this behavior is not a general assumption; but, rather is implementation-specific. In the proposed switch we opted for the Store-and-Forward packet switching rather than the Cut-through one. This choice allows a simple forwarding of TSN frames while avoiding any additional potential complications that can be caused by processing a frame without storing the entire frame in a buffer. Thus, as shown in Section V-B, our TSN switch employs packet-mode FIFOs in several components, and their used cycles are proportional to the size of a frame. However, this mode can be eliminated by changing the concerned buffers' settings to regular FIFOs. In other words, Cut-through switching can be easily adopted instead if users wish to further reduce the latency caused by waiting for the entire frame forwarding.

To enable the estimation of the end-to-end delay, we modified the delay analysis model of the IEEE standard. We reorganize a per-hop delay into three portions; 1) the delays of the FPGA components shown in Section V-B including those caused by packet-mode FIFOs and PMD, 2) the sum of the buffering delay caused by the contention of concurrent flows, and 3) the media-dependant delay between the PHYs of both endpoints. We already discussed the first portion. The contribution of the third one is a few microseconds at most. Thus, in the following experiments, we focus on the second portion.

It should be noted that, as proved in [26], the delay caused by the Interleaved Regulator (i.e., the period of time to wait for the EligibilityTime of a frame in the queue of the ATS transmission mechanism) does not introduce any additional delay to the per-hop delay, in several cases (i.e., the case that ATS flows belong to the highest traffic class and the case that the minimum frame sizes of ATS flows are the same). Even in other cases, it will have little impact on the per-hop delay because the buffering delay caused by contention will be more dominant.

The buffering delay model caused by contention is general for any TSN implementations supporting ATS. This delay happens in the queue of the Transmission Selection mechanism of a traffic class. In this paper, we denote it by $d_{CO}$. The Token Bucket Emulation case of Equation (9) in [26] gives this delay of the original ATS algorithm. Since [27] proved that the original ATS algorithm and the standard algorithm are equivalent, we apply the equation for our ATS implementation.

The maximum contention delay for the flow $f$ in the queue of the Transmission Selection mechanism at the $k$-th hop switch is formulated with the defined parameter variables in the Annex:

$$d_{CO,max}(k,f) = \frac{\sum_{g \in F_H(k,f) \cup F_S(k,f)} b_{max}(k,g) - l_{min}(f) + l_{LP,max}(k,f)}{R(k) - \sum_{g \in F_H(k,f)} r_{max}(k,g)} \quad (1)$$

where:

- $\sum_{g \in F_H(k,f) \cup F_S(k,f)} b_{max}(k,g)$: The sum of the maximum burst sizes of all the flows in higher traffic classes $F_H(k,f)$ and those in the same traffic class $F_S(k,f)$
- $l_{min}(f)$: The minimum frame size of the flow $f$
- $l_{LP,max}(k,f)$: The maximum frame size of any lower traffic classes
- $R(k)$: The link speed of the $k$-th hop
- $\sum_{g \in F_H(k,f)} r_{max}(k,g)$: The sum of the maximum transmit rates of all the flows in higher traffic classes $F_H(k,f)$

The setup of the experiments was a 2-hop topology. The contention delay in the switch was of the second hop. That of the first hop is in the network interface of Host A, which is not necessary to discuss.

### B. LATENCY OF AN ATS FLOW COMPETING WITH A BEST-EFFORT FLOW

We measured the latency of frames of an ATS flow through Traffic Class 7 assigned to the ATS Transmission Selection mechanism. This flow with a best-effort flow through Traffic Class 5 assigned to Strict Priority. The frames of the ATS flow were transmitted from Port 0 of Host A by `plget`, its transmission rate was 100 frame per second (i.e., 1.2 Mbps) and 10,000 frames were sent for 100 seconds. The frames of the best-effort flow were transmitted from Port 0 of Host B at different rates by `iperf`. The destination of both flows was Port 1 of Host A. The CIR of the ATS flow was set to 100 Mbps and its CBS was set to 1542 Bytes. As noted in Section V-A, the link rates of the connections between the ports were 1 Gbps.

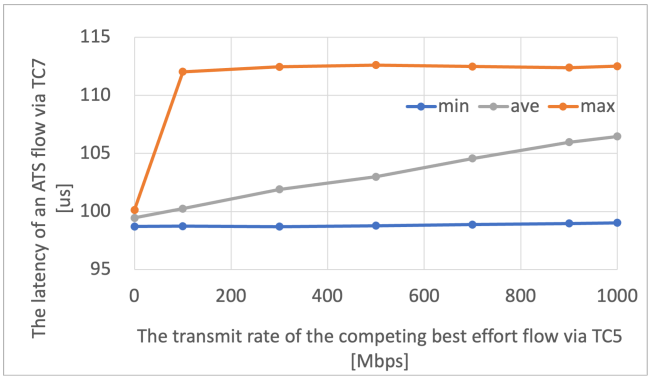



FIGURE 17: Latency of an ATS flow via Traffic Class 7 competing with a best-effort flow via Traffic Class 5

Figure 17 shows the minimum/average/maximum latency of the frames of the ATS flows. As the transmit rate of the competing best-effort flow increased, the possibility of contention in the egress port of the switch increased, which resulted in a higher average latency. However, with any transmit rate of the best-effort flow, the observed worst latency of the ATS flow was approximately 112 us, which was 13 us increase from the minimum latency. According to Equation 3, the worst-case latency caused by contention is calculated as 12.3 us.

$$d_{CO,max}(2,f) = \frac{b_{max}(2,f) - l_{min}(f) + l_{LP,max}(2,f)}{R(2)} = \frac{1542B - 1542B + 1542B}{1Gbps} = 12.3us \quad (2)$$

We consider that the observed latency increase of 13 us is consistent with the theoretical value, taking into account that the precision of the latency measurement mechanism was 1 us, as discussed above. Note that as for Figure 17, the largest standard deviation was 4.26 us at 700 Mbps. Its confidence interval with the 95% confidence level was 83.6 ns. This value is sufficiently small to discuss the population mean by the plotted sample means.

Taking a closer look, Figure 18 shows the time-series data of the observed latency of each frame in the ATS flow, in the case of a competing 500-Mbps best-effort flow. A dot in the graph corresponds to each frame. The first approximately 1500 frames of the ATS flow conflicted with the frames of the best-effort flow with high probability. After that, conflicts were not observed for approximately 2000 frames. Frames of both flows were transmitted periodically in their own transmit intervals, and the difference in intervals caused the pattern of possible conflicts.

### C. LATENCY OF AN ATS FLOW COMPETING WITH ANOTHER ATS FLOW IN THE SAME TRAFFIC CLASS

In this subsection, we measure the latency of an ATS flow competing with another ATS flow in the same traffic class. Instead of the best-effort flow, the other ATS flow was transmitted from Port 0 of Host B at 1 Gbps by `iperf`. Its CIR

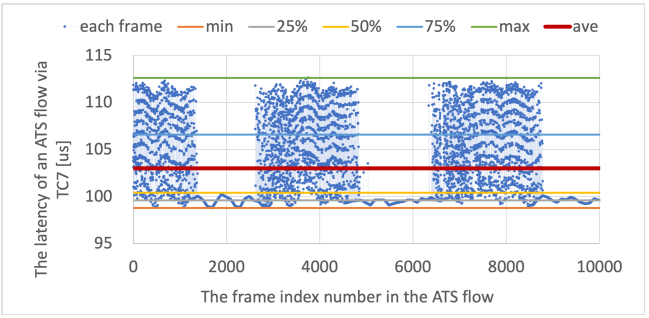



FIGURE 18: Latency of each frame of the ATS flow competing with the best-effort flow of 500 Mbps

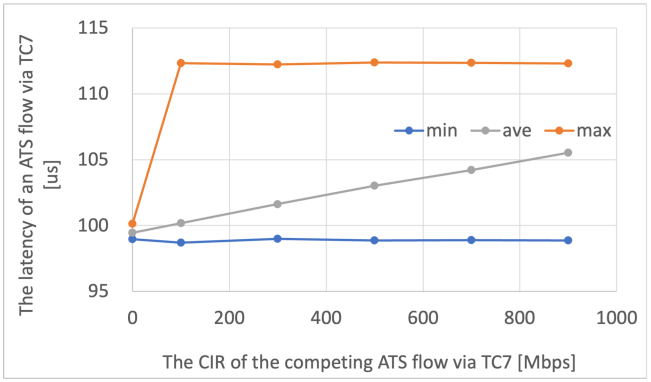



FIGURE 19: Latency of an ATS flow via Traffic Class 7 competing with another ATS flow in the same traffic class

was set to different rates at the switch. Since the ATS flows arrived at different ports of the switch, respectively, their ATS scheduler groups were different. The setting of the latency measurement flow by plget was the same as Section VI-B.

As shown in Figure 19, the latency of the ATS flow was impacted by the other ATS flow in the same traffic class. As the set CIR of the competing ATS flow increased, the possibility of frame contention of the two ATS flows at the egress port of the switch increased. However, the worst-case latency increase was limited to approximately 13 us in any case, which is close to the theoretical value calculated by the below equation:

$$d_{CO,max}(2,f_1) = \frac{\sum_{g\in f_1,f_2} b_{max}(2,g) - l_{min}(f_1)}{R(2)} = \frac{(1542B + 1542B) - 1542B}{1Gbps} = 12.3us \quad (3)$$

Note that as for Figure 19, the largest standard deviation was 4.17 us at 500 Mbps. Its confidence interval with the 95% confidence level was 81.7 ns.

### D. LATENCY OF AN ATS FLOW COMPETING WITH AN ATS FLOW IN ANOTHER TRAFFIC CLASS AND A BEST-EFFORT FLOW

Hereafter, we discuss the latency of an ATS flow that simultaneously competes with another ATS flow in another traffic

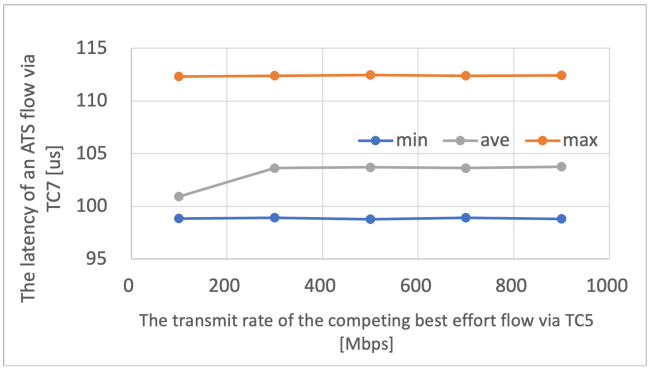



FIGURE 20: Latency of an ATS flow via Traffic Class 7 competing an ATS flow via Traffic Class 6 and a best-effort flow via Traffic Class 5

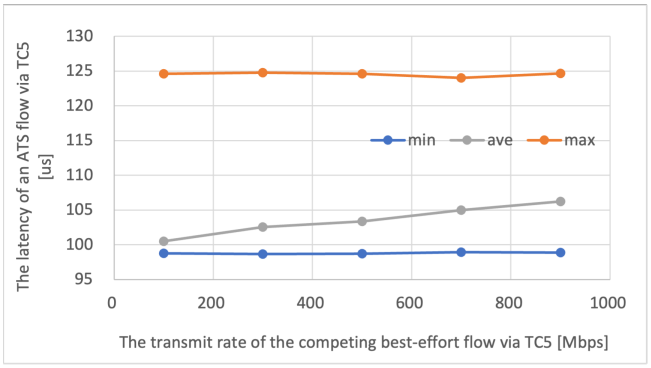



FIGURE 21: Latency of an ATS flow via Traffic Class 6 competing an ATS flow via Traffic Class 7 and a best-effort flow via Traffic Class 5

class as well as with a best-effort flow. First, we measured the latency of an ATS flow in the highest traffic class (i.e., Traffic Class 7). This ATS flow was transmitted from Port 0 of Host A by `plget` in the same manner as Section VI-B. The other ATS flow of the second highest traffic class (i.e., Traffic Class 6) was transmitted from Port 0 of Host B at 100 Mbps by `iperf`. The CIRs of both ATS flows were set to 100 Mbps and their CBSs were set to 1542 Bytes. The best-effort flow was transmitted from Port 0 of Host B at different rates by `iperf`.

As for the ATS flow of the highest traffic class in this experiment, the theoretical worst-case latency caused by contention is 12.3 us, which is estimated by the same Equation 2. The obtained result from our experiment (shown in Figure 20), demonstrates an approximate 13 us latency increase, which was consistent with the theoretical value mentioned above. Note that as for Figure 20, the largest standard deviation was 4.31 us at 900 Mbps. Its confidence interval with the 95% confidence level was 84.5 ns.

Second, we measured the latency of an ATS flow in the second highest traffic class (i.e., Traffic Class 6). It was transmitted from Port 0 of Host A, and an ATS flow in a higher Traffic Class 7 was transmitted from Port 0 of Host B. The other configurations were the same as the previous experiment. In this case, the theoretical worst-case latency caused by contention was calculated as 27.4 us using the following equation:

$$d_{CO,max}(2,f_2) = \frac{\sum_{g\in f_1\cup f_2} b_{max}(2,g) - l_{min}(f_2) + l_{LP,max}(2,f_2)}{R(2) - \sum_{g\in f_1} r_{max}(2,g)} = \frac{(1542B+1542B) - 1542B + 1542B}{1Gbps - 100Mbps} = 27.4us \quad (4)$$

As presented in Figure 21, the observed latency increase was approximately 26 us, which can be supported by the theoretical value. Figure 22 shows the latency of each frame of the ATS flow via Traffic Class 6, in the case of a 500-Mbps best-effort flow. Because the transmit intervals of all

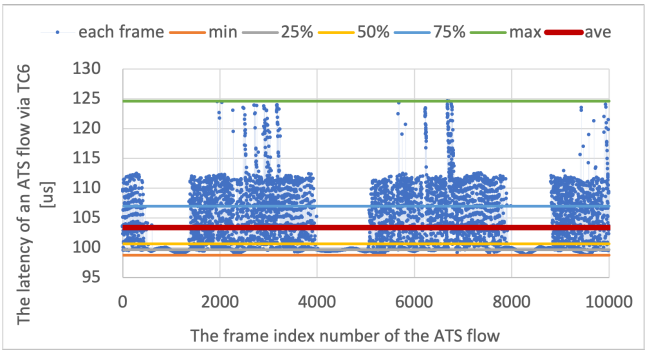



FIGURE 22: The latency of each frame of the ATS flow via Traffic Class 6 competing an ATS flow via Traffic Class 7 and a 500-Mbps best-effort flow via Traffic Class 5

the flows were different, the possibility of conflicts changed into a pattern. In the worst case where the frames of the three flows were simultaneously ready for transmission, the ATS flow of the second highest traffic class experienced a 26-us delay. Note that as for Figure 21, the largest standard deviation was 4.93 us at 300 Mbps. Its confidence interval with the 95% confidence level was 96.6 ns.

## VII. CONCLUSION

In this paper, we proposed a hardware implementation of Asynchronous Traffic Shaping functionalities on an FPGA-based switch for Time Sensitive Networks. We gave a detailed description of the proposed switch design and implementation on an affordable FPGA board and we validated its functionalities through several evaluations. To the best of our knowledge, this is the first open-source switch architecture that supports ATS.

Our evaluations show that the proposed switch properly shapes the traffic by eliminating irregular bursts and efficiently forwarding streams depending on their priority, according to the TSN standards rules.

We also demonstrate that our latency evaluation results conform with the ATS theoretical model. Thanks to the de-

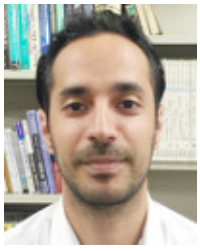
terministic nature of FPGAs, the proposed switch has demonstrated a stable behavior without observing any discrepancy in the obtained results despite reproducing the experiments several times under several scenarios. Furthermore, to eliminate the potential reliability risks related to process variation, the proposed switch was implemented and tested on more than one FPGA board. All the conducted tests did not show any instability nor any type of faults.

In this paper, we aim to provide a transparent, accessible, and open-source design framework with software and hardware implementation details, and simple demos to demonstrate the implemented TSN capabilities. We aspire to allow researchers to reuse, modify, and optimize the current implementation by adding other TSN features.

As future priorities, we aim to port the current 1G switch to 10G and integrate it into a real practical Multi-access Edge Computing (MEC) ecosystem. This ecosystem will be tested and validated to be used in Smart Factories and Autonomous driving.

## REFERENCES

[1] M. A. Al-Maqri, M. A. Alrshah, and M. Othman, "Review on qos provisioning approaches for supporting video traffic in ieee802.11e: Challenges and issues," IEEE Access, vol. 6, p. 55202–55219, 2018.
[2] M. Raza, H. Le-Minh, N. Aslam, S. Hussain, M. A. Imran, R. Tafazolli, and H. X. Nguyen, "Dynamic priority based reliable real-time communications for infrastructure-less networks," IEEE Access, vol. 6, pp. 67 338–67 359, 2018.
[3] IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks, Institute of Electrical and Electronics Engineers, 2022.
[4] A. Nasrallah, A. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of ieee 802.1 tsn time aware shaper (tas) and asynchronous traffic shaper (ats)," IEEE Access, vol. 7, pp. 44 165–44 181, 2019.
[5] The AIST-TSN project repository, https://github.com/CCIRT/aist-tsn/, 2024.
[6] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely survey of time-sensitive networking: Past and future directions," IEEE Access, vol. 9, pp. 142 506–142 527, 2021.
[7] Y. Xu and J. Huang, "A survey on time-sensitive networking standards and applications for intelligent driving," Processes, vol. 11, no. 7, 2023.
[8] T. Fedullo, A. Morato, F. Tramarin, L. Rovati, and S. Vitturi, "A comprehensive review on time sensitive networks with a special focus on its applicability to industrial smart and distributed measurement systems," Sensors, vol. 22, no. 4, 2022.
[9] R. Debnath, P. Hortig, L. Zhao, and S. Steinhorst, "Advanced modeling and analysis of individual and combined tsn shapers in omnet++," in 2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2023, pp. 176–185.
[10] J. Jiang, Y. Li, S. H. Hong, A. Xu, and K. Wang, "A time-sensitive networking (tsn) simulation model based on omnet++," in 2018 IEEE International Conference on Mechatronics and Automation (ICMA), 2018, pp. 643–648.
[11] J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel, "Nesting: Simulating ieee time-sensitive networking (tsn) in omnet++," in 2019 International Conference on Networked Systems (NetSys), 2019, pp. 1–8.
[12] Z. Zhou, Y. Yan, M. Berger, and S. Ruepp, "Analysis and modeling of asynchronous traffic shaping in time sensitive networks," in 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), 2018, pp. 1–4.
[13] H. Guo, W. Li, J. Lin, J. Zhou, Q. Xun, S. Zhan, Y. Huang, Y. Wang, and Q. Cheng, "A scalable asynchronous traffic shaping mechanism for tsn with time slot and polling," in NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, 2023, pp. 1–5.
[14] M. Ulbricht, S. Senk, H. K. Nazari, H.-H. Liu, M. Reisslein, G. T. Nguyen, and F. H. P. Fitzek, "Tsn-flextest: Flexible tsn measurement testbed," IEEE Transactions on Network and Service Management, vol. 21, no. 2, pp. 1387–1402, 2024.
[15] F. Rezabek, M. Bosk, T. Paul, K. Holzinger, S. Gallenmüller, A. Gonzalez, A. Kane, F. Fons, Z. Haigang, G. Carle, and J. Ott, "Engine: Developing a flexible research infrastructure for reliable and scalable intra-vehicular tsn networks," in 2021 17th International Conference on Network and Service Management (CNSM), 2021, pp. 530–536.
[16] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of open vSwitch," in 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15). USENIX Association, May 2015, pp. 117–130.
[17] Y. S. Do, S. B. Oh, S. J. Lim, and J. W. Jeon, "Approach to improving asynchronous traffic shaping performance using a combination of shaper," in 2024 IEEE 33rd International Symposium on Industrial Electronics (ISIE), 2024, pp. 1–6.
[18] Z. Zhou, M. S. Berger, and Y. Yan, "Mapping tsn traffic scheduling and shaping to fpga-based architecture," IEEE Access, vol. 8, pp. 221 503–221 512, 2020.
[19] J. Y. Wei Quan, Wenwen Fu and Z. Sun, "Opentsn: an open-source project for time-sensitive networking system development," CCF Transactions on Networking, vol. 3, pp. 51–65, 2020.
[20] AMD Xilinx Cooperation, KC705 Evaluation Board for the Kintex-7 FPGA, UG810 (v1.9), 2019.
[21] O. E. D. Inc., https://ethernetfmc.com/, 2022.
[22] Advanced Micro Devices, Inc., Tri-Mode Ethernet MAC LogiCORE IP Product Guide (PG051), v9.0, 2023.
[23] IEEE Standard for Local and Metropolitan Area Networks–Frame Replication and Elimination for Reliability, Institute of Electrical and Electronics Engineers, 2017.
[24] Intel Cooperation, Intel Ethernet Controller I210 Datasheet, Revision Number: 3.7, 2021.
[25] AMD Xilinx Cooperation, Vivado Design Suite User Guide: Getting Started (UG910), 2022.
[26] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in 2016 28th Euromicro Conference on Real-Time Systems (ECRTS), 2016, pp. 75–85.
[27] M. Boyer, "Equivalence between the Urgency Based Shaper and Asynchronous Traffic Shaping in Time Sensitive Networking," Leibniz Transactions on Embedded Systems, vol. 9, no. 1, pp. 1:1–1:27, 2024.



AKRAM BEN AHMED received the Ph.D. degree in computer science and engineering from the University of Aizu, Japan, in 2015. He is currently a Research Scientist with the National Institute of Advanced Industrial Science and Technology, Tokyo, Japan. His research interests include on-chip interconnection networks, reliable and fault-tolerant systems, and ultralow-power embedded systems.

TAKAHIRO HIROFUCHI is a Senior Research Scientist and a team leader in the National Institute of Advanced Industrial Science and Technology (AIST) in Japan. He is working on the co-design of software and hardware for edge and cloud computing. He obtained a Ph.D. of engineering in March 2007 at the Graduate School of Information Science of Nara Institute of Science and Technology (NAIST). He is an expert of operating system, virtual machine, and network technologies.

TAKAAKI FUKAI is a Research Scientist at National Institute of Advanced Industrial Science and Technology (AIST) in Japan. He received his B.S. degree in Engineering from Okayama University, Japan, in 2013. He received his M.S. and his Ph.D. degree in Engineering from University of Tsukuba, Japan, in 2015 and 2018, respectively. His research interests include operating systems, virtualization, and cloud computing.