electronics

MDPI

Article

Hardware Support to Minimize the End-to-End Delay in Ethernet-Based Ring Networks

Tomás P. Corrêa [1,*] and Luis Almeida [2]

[1] Electrical Engineering Graduate Program, Federal University of Minas Gerais, Belo Horizonte 31270-901, Brazil
[2] CISTER, Instituto de Telecomunicações, FEUP-University of Porto, 4200-465 Porto, Portugal; lda@fe.up.pt
* Correspondence: tc.ufmg@gmail.com

Received: 29 August 2019; Accepted: 25 September 2019; Published: 28 September 2019

Abstract: Ethernet is a popular networking technology in factory automation and industrial embedded systems, frequently using a ring topology for improved fault-tolerance. As many applications demand ever shorter cycle times and a higher number of nodes, the popular ring endure to remain as a valid topology. In this work, we discuss the factors that determine the ring network delay and show how they affect the network cycle time. Since increasing the link capacity has limited reach, we explore a time-triggered protocol that brings the nodes forwarding delay near to the physical layer delay. Additionally, we propose hardware accelerators based on FPGA technology that minimise the packet reception delay from physical reception to delivery to an application handler, preserving Ethernet layers and being compatible with its standard. This paper explains the accelerators concept and implementation, presents measurements using standard Media Access Control implementations, and shows the solution effectiveness with experimental results. We achieved a delay, from physical reception to the triggering of a user-level handler, of 1.1 µs independent of the packet length.

Keywords: end-to-end delay; ethernet; hardware accelerator

1. Introduction

In last years, the number of applications requiring high-speed communications, such as motion control [1], Modular Multilevel Converters (MMC) [2,3], power train or chassis control [4], has increased. Those applications demand short update cycles of less than 200 µs and represent a challenge to the communication network design.

The network topology, one of the designer's degrees of freedom, has a strong impact on the number and length of the communication links, on the network delay, and the tolerance to failures. Among the topologies that can be considered (e.g., star, tree, ring, bus and hybrid, see Figure 1), the ring topology is often the preferred one, because it has clear benefits in network cabling, reducing the number of links and their length, thus simplifying deployment and maintenance. Moreover, the ring topology is the simplest one that offers two disjoint paths between any two nodes, so it is tolerant to single node or link failures. However, the ring structure implies longer delays than other alternatives, such as a tree topology, and fails to reach the required performance as the number of nodes increases.

In this work, we discuss the factors that determine the ring network delay and show how they affect the network cycle time. As shown below, increasing the link capacity has limited benefits, thus, when performance is at a premium, a co-design between the control and the communication might be necessary. We also explore a time-triggered protocol and the synchronisation of the nodes physical layer (PHY), such that the forwarding delay of a packet passing through an active node is reduced to

the PHY delay and a single clock cycle of the interface between PHY and the Media Access Control (MAC) layer.

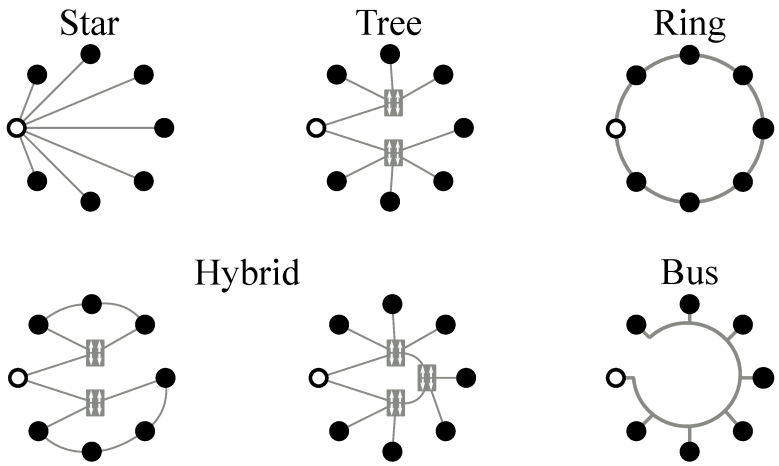


Figure 1. Network topologies. The small white dots represent the master node and the black dots the slaves.

Another contribution of this work is the use of hardware accelerators to eliminate the node delay dependency on the packet size, a point often overlooked. This reduction is relevant, because it decreases the end-to-end delay that affects the application control performance [5], which is the sum of cycle time plus the node internal delays in transferring data to and from the network. Those accelerators can be easily implemented in Field Programmable Gate Array (FPGA) or Application Specific Integrated Circuit (ASIC) technologies.

This paper is organised as follows. Section 2 discuss related work. Section 3 introduces the concept of Minimum Cycle Time (MCT), compares different network topologies, and explains how link capacity and protocol influence the MCT. Section 4 shows reception delay measurements of three MAC implementations, making evident that the MAC choice is relevant when the application demands a reduced end-to-end delay. Section 5 describes a technique to reduce the active node forwarding delay to the latency of the Physical Layer (PHY) latency plus a single clock cycle of the MAC interface. Section 6 details two proposed hardware accelerators and explains how they enable removing the dependency of the MAC reception delay on the payload size. Section 7 states the conclusions of this work.

2. Related Work on Reducing the Cycle-Time

Curiously, when discussing the cycle time, node internal delays are often ignored or underestimated, but recently authors have recognised their relevance. Briscoe et al. [6] classified the sources of Internet latency into five categories, among them the *Intra-end-host* delays. Ramaswamy et al. [7] found that functionalities allocated inside network middle hops (routers), such as encryption, introduce processing delays that may represent up to 50% of the total network delay. Bertocco et al. [8] pointed to the relevant delays that access points introduce when employed to interface wired and wireless networks. Those works, as well as most that discuss network latency, are mostly concerned with large networks and non-real-time or soft real-time applications.

Orfanus et al. [9] and Cottet et al. [10] touched the subject of reducing the node internal delays when they implemented an EtherCAT Master stack capable of reaching cycle times of 20 µs, but they did not consider the influence of the MAC type or the delay dependence on payload. Corrêa et al. [11] found the transmission and reception internal delays in a Zynq System-on-Chip (SoC) to be as high as 30 µs for large packets using Ethernet technology. The MAC architecture and the number of times data

are copied are key aspects. In general, we did not find a work that minimises the delays incurred by a ring topology, which is what we propose here.

3. Networks and Their Minimum Cycle Time

As referred above, the ring topology presents several advantages in cabling reduction and management as well as in fault-tolerance. However, applications such as high-speed servoing and MMC control demand update cycles of less than 200 µs. Such short cycles represent a challenge to this topology, as the number of nodes influences the Minimum Cycle Time (MCT), i.e., the delay in updating all slave nodes in the network with new data, considering master–slave logical interactions as typical in industry. In the ring topology, the MCT is expressed by Equation (1), where κ is the number of slaves, P is the payload in bytes, P_{overhead} is the number of bytes necessary for frame packing, P_{max} is the maximum payload that fits in one frame, T_{byte} is the time to transfer one byte, T_{fw} is the forwarding delay, and $\lceil x \rceil$ is the ceiling function, returning the nearest integer not smaller than x.

$$MCT_{\text{Ring}} = P \cdot T_{\text{byte}} + \left\lceil \frac{P}{P_{max}} \right\rceil \cdot P_{\text{overhead}} \cdot T_{\text{byte}} + \kappa \cdot T_{\text{fw}}, \tag{1}$$

To put the ring *MCT* in perspective, consider Equation (2) that corresponds to the equivalent cycle time in a tree topology using switches. The tree MCT depends on the packet length and the number of hops (Equation (3)), where *ports* is how many downwards (i.e., leaves side) ports the switches have, and T_{hop} is the hop forwarding delay. Cut-through switches take forwarding decision immediately after the destination address field has arrived, so the forwarding delay is in the order of microseconds and can be less than 3 µs with Fast Ethernet and less than 1 µs with Gigabit Ethernet [12,13].

$$MCT_{\text{Tree}} = P \cdot T_{\text{byte}} + \left\lceil \frac{P}{1500} \right\rceil \cdot 50 \cdot T_{\text{byte}} + \text{hops} \cdot T_{\text{hop}} \tag{2}$$

$$\text{hops} = \lceil log_{\text{ports}} \kappa \rceil + 1 \tag{3}$$

In Figure 2, we show the MCT of concrete representative technologies and a payload of 4 Bytes/node. On the one hand, we have EtherCAT ring ($P_{max} = 1488$ Bytes, $P_{\text{overhead}} = 50$ Bytes, $T_{\text{byte}} = 80$ ns, and $T_{fw} = 0.8$ µs) and, on the other hand, we use Fast Ethernet switches with four and eight downward ports per switch. As the number of nodes increase, the performance difference is remarkable and, more important, it shows how ring topology performance may become inadequate for larger networks.

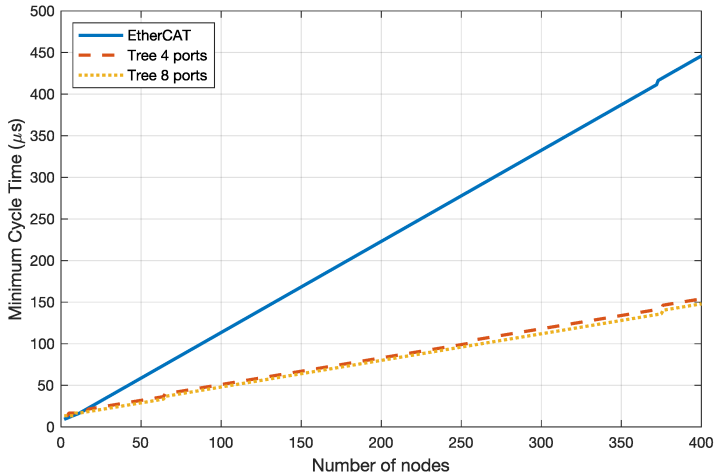


Figure 2. Minimum Cycle Time for the ring (EtherCAT) and tree (Fast Ethernet) network topologies, when broadcasting information from the master controller to cells, only.

Looking carefully at Equation (1), we can see that different aspects influence the MCT. When the goal is to reduce it, the designer needs to consider several dimensions—the link technology (relates to T_{byte}, T_{fw}, P_{max}), the protocol (relates to P_{max}, $P_{overhead}$, and T_{fw}), and the application (relates to P and κ)—to reach the least MCT possible, as we explain next.

3.1. Link Technology

Link technology, and more specifically speed, is likely the first digital communication aspect that comes to mind when the designer wants to reduce communication delay and increase throughput. The link technology has direct influence on the time to transmit one byte (T_{byte}) and may limit the maximum amount of data a node can transmit at once (even when the link can continuously transmit data, the protocol might still impose a finite P_{max}). The third variable that the link technology influences is the forwarding delay (T_{fw}), but the technical literature has little on how the link capacity affects this parameter. Before we investigate this relation, let us first discuss how link speed and forwarding delay limit the reduction of the MCT.

Consider the ring MCT expression in Equation (1). Note that its first two terms depend on the time spent in transmitting an entire frame (the "frame" delay), which is related to the link transmission rate. The last term in Equation (1) depends on the forwarding delay incurred when packets cross active elements in the network (the "propagation" delay). Both frame and forwarding delay depend directly on the link technology.

To better understand their impact, we represent the MCT in a bi-dimensional space of latency versus bandwidth. There, we can identify two regions: the bandwidth dominated region, in which the frame delay represents most of the MCT; and the latency dominated region, in which the propagation delay represents most of the MCT [14]. The boundary between those two regions corresponds to a situation in which both effects have equal influence on the MCT and it is defined by Equation (4).

Often, applications have a payload proportional to the number of nodes ($P = \kappa \cdot P_{Bytes/node}$) or such dependency is a good approximation. If we neglect the protocol overhead (in an efficient protocol it should represent a small fraction of the payload) then the critical forwarding delay ($T_{\text{fw}}^{\text{critical}}$), defined as the forwarding delay that equals the frame delay, becomes proportional to the communication bandwidth and payload per node Equation (5). Note that the number of nodes in the network becomes irrelevant.

$$\left(P + \overline{P}_{\text{overhead}}\right) \cdot T_{\text{byte}} = \kappa \cdot T_{\text{fw}} \tag{4}$$

$$T_{\text{fw}}^{\text{critical}} = T_{\text{byte}} \cdot P_{\text{Bytes/node}} \tag{5}$$

Figure 3 shows a plot of Equation (5) in logarithmic scale for 4, 8, and 16 Bytes/node. The bandwidth and the latency dominated regions are, respectively, below and above the critical forwarding delay line. As the bandwidth increases, the critical forwarding delay falls accordingly. For example, when increasing the bandwidth from 100 Mbit/s to 1 Gbit/s, the critical forwarding delay falls from around 300 ns to near 30 ns.

In practice, the forwarding delay is lower bounded by the PHY transceivers that impose minimum latencies larger than the corresponding critical forwarding delays, particularly for higher link bandwidths. This is due to physical limitations of their internal analog and digital circuitry, since they become more complex to support the higher speeds. To observe this fact, we plot in Figure 3 the typical forwarding delays of Ethernet PHY transceivers (combined Tx + Rx latency) for three bandwidths (10, 100 and 1000 Mbit/s). For completeness, we also plot the latencies of the Serialiser/Deserialiser (SerDes) transceivers provided by Xilinx for high-speed technologies such as Serial RapidIO, Interlaken, PCI Express 2.0 and SATA, namely GTP, GTX and GTH. As it can be seen, for bandwidths higher than Fast Ethernet, all transceivers fall in the latency dominated MCT region.

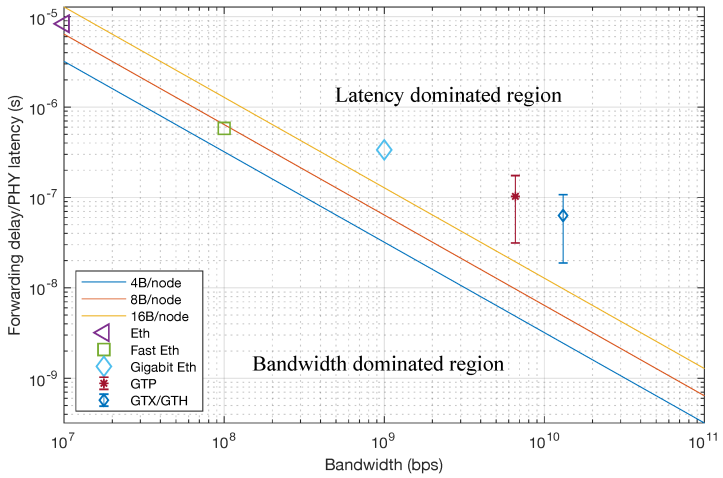


Figure 3. Critical Forwarding Delay and the forwarding delay of Ethernet and SerDes high-speed transceivers (with minimum and maximum values depending on configuration).

3.2. Protocol and Application

A data communication protocol sets the rules for the transmission of data between nodes [15], which includes how data are packed, which nodes can transmit data, when and how the nodes are addressed, how to avoid or deal with collisions, if a node is responsible for managing the network and how it will do that, etc. Therefore, the protocol is a central aspect in defining the network functionality and performance indexes, among them the MCT.

As an example, Equations (6)–(9) correspond to the MCT of PROFINET IRT and EtherCAT (two established real-time Ethernet protocols), of VABS [16] (a recently proposed protocol), and of TTRing [17] (a protocol proposed specifically for the control of MMCs that uses the time triggered paradigm to reduce the forwarding delay). As all protocols use Ethernet, the minimum payload is 46 Bytes ($P_2 = P_3 = max(46, P_{\text{Bytes/node}} \cdot \kappa)$, where $max(x_1, x_2)$ returns the parameter with the largest value). Additionally, PROFINET IRT uses the dynamic packing strategy that requires a minimum payload per node of 38 Bytes [18], i.e., $P_1 = max(46, 38 \cdot \kappa, P_{\text{Bytes/node}} \cdot \kappa)$. In TTRing, P_{master} and P_{slave} are master and slave payloads and GW is the duration of the guard window between phases. The distributed nature of the control in TTring makes both payloads independent of the number of nodes, although they must still be at least 46 Bytes long, $P_{master} \geq 46$ and $P_{slave} \geq 46$. As each protocol has a different HW implementation, the forwarding delays are also different. For calculating the MCTs presented in Figure 4, we adopted the following values: $T^2_{\text{fw}} = 775$ ns, $T^3_{\text{fw}} = 550$ ns, and $T^4_{\text{fw}} = 490$ ns.

$$MCT_1 = P_1 \cdot T_{\text{byte}} + \left\lceil \frac{P_1}{1492} \right\rceil \cdot 36 \cdot T_{\text{byte}} \tag{6}$$

$$MCT_2 = P_2 \cdot T_{\text{byte}} + \left\lceil \frac{P_2}{1488} \right\rceil \cdot 50 \cdot T_{\text{byte}} + \kappa \cdot T^2_{\text{fw}} \tag{7}$$

$$MCT_3 = P_3 \cdot T_{\text{byte}} + \left\lceil \frac{P_3}{1024} \right\rceil \cdot 10 \cdot T_{\text{byte}} + \kappa \cdot T^3_{\text{fw}} \tag{8}$$

$$MCT_4 = (P_{master} + P_{slave}) \cdot T_{\text{byte}} + \kappa \cdot T^4_{\text{fw}} + 2 \cdot GW + \left(\left\lceil \frac{P_{master}}{1500} \right\rceil + \left\lceil \frac{P_{slave}}{1500} \right\rceil \right) \cdot 50 \cdot T_{\text{byte}} \tag{9}$$

Figure 4 shows a comparison of the protocols MCT for different number of nodes and a payload of 4 Bytes/node. In this specific example, PROFINET IRT is the worst performing protocol. Concerning TTRing, the protocol sends two packets per cycle, which has a strong performance penalty for small networks and lower link bandwidth. Thus, at 100 Mbit/s TTRing yields a worst result than

EtherCAT and VABS. For a network larger than 22 nodes, the TTRing protocol outperforms the others. When compared to EtherCAT, the MCT of TTRing 100 Mbit/s becomes less than half when the network is larger than 228 nodes and equal to 209 µs against 446 µs of EtherCAT with 400 nodes. The TTRing Gigabit implementation is faster than the other protocols for any network size.

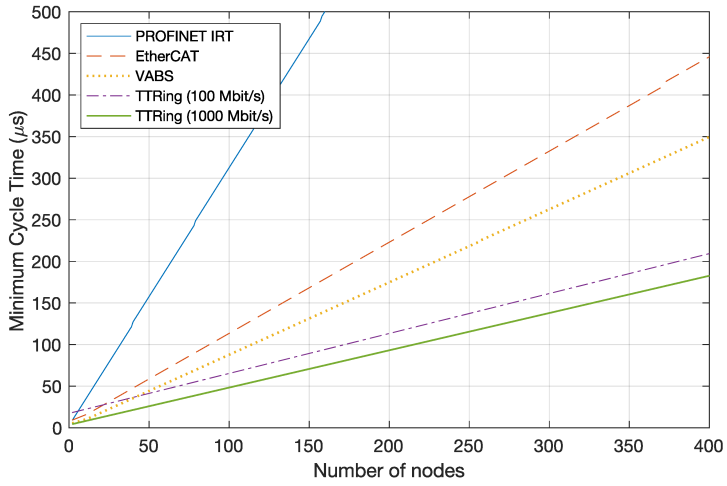


Figure 4. Minimum Cycle Time for several protocols depending on the network size.

When we consider Equation (1) as a general equation for ring topologies, it is clear that the protocol directly defines the packet maximum payload (P_{max}) and overhead ($P_{overhead}$). More subtle, though, is the influence that the protocol has in the forwarding delay (T_{fw}). It manifests itself through the rules that a node needs to obey for forwarding the packet. If, for instance, the node needs to check that the frame is valid (similar to store-and-forward switches) or to read the destination address (similar to cut-through switches) before forwarding, then the protocol imposes a higher minimum forwarding delay and the MCT increases.

Another dependence between forwarding delay and protocol, not a direct one but anyway important, is the performance of the node implementation at system deployment time. For example, the forwarding delay of the VABS implementation is 500 µs [16] but it could be possible to reduce this delay by using more recent components. A similar situation happens with EtherCAT, but it is more difficult to put a number on its forwarding delay: *Prytz* [19] mentioned a delay of 500 ns; Vitturi et al. [1] measured an average delay of 1 µs; and Orfanus et al. [9] wrote that it is lower than 1µs. This divergence is mainly due to different implementations, PHYs and configurations adopted. The latest ASIC from Beckhoff (EtherCAT inventor) can arguably be considered one of the fastest alternatives. According to the ET1100 Hardware datasheet [20], the maximum forwarding delay from the Media Independent Interface (MII) in the receiver to the MII in the transmitter is 335 ns plus the PHY delay, when the Rx Buffer is set to the default size of seven [21]. Again, if EtherCAT is the protocol of choice, the best the designer can do is to pick the quickest slave implementation.

If using a higher link capacity, choosing a fast protocol, and adopting the fastest implementation is insufficient for achieving the necessary performance, the designer can lastly resort to modifying the application such that the amount of data flowing in the network is reduced or even the number of nodes minimised. In this case, we can conceive a complete control and communication co-design, in which details of the control strategy influences the design of the network and vice versa. Since this is not the focus of this work, a discussion of the co-design is not included here.

4. MAC and Reception Delay

As mentioned in the Introduction, the node internal delays are often ignored, but they can represent a considerable fraction of the end-to-end delay, especially in the high performance

applications considered here. The internal delay takes place as the node moves data internally and the protocol stack processes the packet.

In Ethernet, the next layer above the physical one is the Media Access Control. It performs, together with the Logical Link Control, the functions of the Data Link Layer, as described by the Open Systems Interconnection (OSI) model. Its main functions are [22]:

1. Data encapsulation (transmit and receive):
   (a) Framing (frame boundary delimitation, frame synchronisation);
   (b) Addressing (handling of source and destination addresses); and
   (c) Error detection (detection of physical medium transmission errors).
2. Media Access Management:
   (a) Medium allocation (collision avoidance); and
   (b) Contention resolution (collision handling).

Therefore, it is the MAC sublayer that defines the Ethernet packet format, including its fields and size, the addressing possibilities (unicast, multicast, and broadcast), the order of bit transmission, and the mode of operation (half- or full-duplex). It is also responsible for generating and verifying the Frame Check Sequence (FCS), for supporting nodes segregation in a same physical network with VLAN tags, and for enforcing the minimum and maximum frame length, the interframe gap, and the media access rule—Carrier Sense Multiple Access with Collision Detection (CSMA/CD, ignored in full-duplex mode).

MAC Implementations and Delay Measurements

To verify the impact of the MAC in the communication delay, let us use the concrete case of the Zync SoC that combines an ARM processor with an FPGA fabric and offers the designer three MAC implementations, namely the Gigabit Ethernet MAC (GEM), the Ethernet Lite, and the Tri-Speed MAC (TEMAC). A previous work [11] found Ethernet Lite delays to be lower than GEM, but it had no information on the TEMAC, which we include here. These three different MAC implementations are representative of different classes that can be found in a variety of devices. For example, Texas Instrument's Keystone architecture, employed in the C667x and C665x multicore processor families, uses a MAC peripheral that has a local First In First Out (FIFO) buffer and transfers the information from/to the main memory using a Direct Memory Access (DMA) engine [23], as with the Zynq GEM.

The test setup employed was a Zynq SoC board (Trenz TE0729) and carrier board (Trenz TEB0729) running a modified version of the Echo Server demo application. It was connected to a host PC that sends packets with the desired characteristics. We designed a capture unit in VHDL to count the number of clock cycles between two different events, such that we could measure the packets timing with an adequate resolution (10 ns). When measuring the incoming delays, the trailing edge of the signal that the PHY sends to the MAC when receiving a packet (RX_DV) causes the capture unit to start counting and a software-controlled output stops it. When measuring outgoing delays, the software-controlled output triggers the start and the rising edge of the signal that the MAC sends to the PHY to start transmission (TX_EN) stops it. We verified the capture unit measurement by connecting the start and stop signals to an oscilloscope. For every test run, the program logged 512 measurements and sent them to the host PC for processing.

Figure 5 shows the mean value and deviation for each measurement point (64, 256 and 1024 Bytes packet payload) and for each MAC alternative, where "EL,DM" stands for Ethernet Lite with Device Memory and "EL,SO" with Strong-Ordered memory models [11]. For the incoming packets, the measurements were taken from the moment the MAC received a packet to the moment the processor: (i) finished copying the respective data to the final memory destination; and (ii) when entering an UDP receive callback function, assuming it is an UDP packet (as commonly used in real-time applications) and that the receiver suffers no interference from higher priority tasks.

The results show Ethernet Lite outperforming GEM and TEMAC in terms of reception delay independently of the packet size. It is surprising, as GEM is a dedicated hard peripheral, integrated to the processor, and TEMAC employs the higher performance AXI-stream interface.

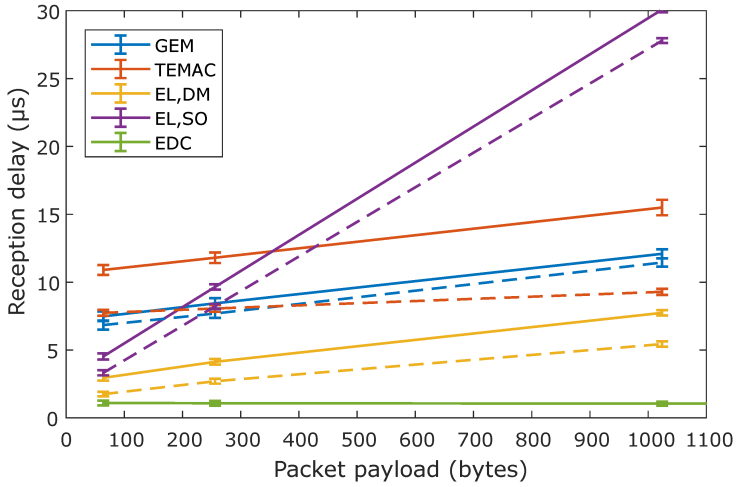


Figure 5. Incoming packet delay (essentially the MAC delay): (i) time from the reception of a packet to transferring the respective data to its final memory destination (dashed lines); and (ii) time to enter a UDP callback function (full lines). The EDC line (lower line) corresponds to the results achieved with the EDC hardware accelerator described in Section 6.

Moreover, the Ethernet Lite MAC generates an interrupt upon packet reception, because it saves the data internally in the MAC and henceforth further data movement needs the participation of the processor. Therefore, Ethernet Lite presents the lowest interrupt service latency (680 ns) at the expense of higher CPU utilisation (though it is still possible to configure a DMA to transfer data to the main memory). In contrast, both GEM and TEMAC interrupt service latencies (7.3 µs and 5.3 µs, respectively, with packet size of 1024 Bytes) have a dependence on the packet size, because the associated DMA engine transfers the packet to the main memory before flagging the interrupt.

The measurements reported in Figure 5 show that the MAC and protocol stack delays are relevant. They are of the same order of magnitude of the MCT in small networks and they contribute to the end-to-end communication delay. As discussed in [24], the end-to-end latency adds to the loop delay and has negative influence in the closed loop system response. Moreover, in certain protocols (e.g., POWERLINK), the MAC may also be involved in the packet forwarding, causing the respective delay to impact the MCT, too.

5. Crunching the Forwarding Delay

Many applications of fast ring networks require broadcasting packets as well as collecting information from the nodes. These two types of interactions can be carried out in alternate phases [17]. Then, in the broadcast phase, the forwarding of each bit can be done almost immediately. Taking advantage of this phase separation it is possible to squeeze the forwarding time in data broadcasts to levels that have not been reported in the literature before. In this section, we present a node interface that identifies these phases and does a direct connection from the receiver to the transmitter PHY while supplying the data to the node MAC block. This node interface can be readily implemented in FPGA technology.

5.1. An Enhanced Node Interface

We show in Figure 6 details of the implementation of the slave nodes network interface, using FPGA technology and assuming a typical master–slave functional architecture. Both receiver

and transmitter PHY interfaces are RGMII (Reduced Gigabit Medium-Independent Interfaces). This requires data rate conversion blocks in both sides that convert the dual data rate to single data rate (IIDR) and vice versa (ODDR). The output of a multiplexer (MUX) drives the transmitter. It has two inputs: the received data and the output of a FIFO buffer. The FIFO stores the data coming from the MAC layer, because it is in a different clock domain. A scheduler controls the multiplexer through the *Select* signal, which determines when the broadcast phase is on. In that phase, the input data are immediately supplied to the output block. Else, the node transmits the data coming from the MAC and stored in the FIFO. In any case, the incoming data are also supplied to the MAC layer.

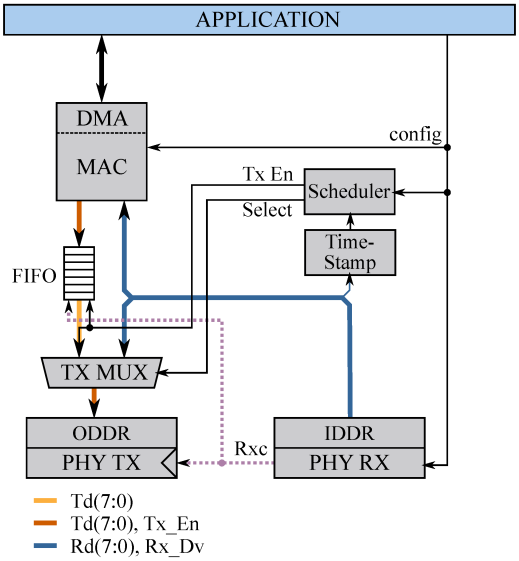


Figure 6. Details of the slave node interface implementation for a dual-phase ring protocol (broadcast and data collection phases). During the broadcast phase, the logic directly connects the incoming data to the transceiver of the node second port.

This implementation allows a single clock delay in the internal logic during the broadcast phase, i.e., 40 ns with Fast Ethernet (4-bit symbols at 10 ns/bit at the IDDR output) and 8 ns with Gigabit Ethernet (8-bit symbols at 1 ns/bit at the IDDR output). The remaining components of the forwarding delay are the PHY transmitter and receiver latencies. Table 1 lists the latency values of common Ethernet PHYs taken from the respective datasheets. Thus, the forwarding delay of a Faster Ethernet node can be as low as 282 ns (40 + 72 + 170 ns) when using the currently quickest PHY (the KSZ8091MLX from Microchip).

Since this implementation causes a single clock delay in the node internal logic, the additional delay and uncertainty necessary for clock domain crossing becomes relevant, as we explain next.

Table 1. Latency of PHYs operating at 100 Mbit/s.

Device	Manufacturer	Max. Latency
DP83867 [25]	Texas Instrument	90 ns (Tx) + 288 ns (Rx)
88E1510P/Q [26]	Marvell	362 ns (Tx + Rx)
88E1510 [a]	Marvell	1.2 µs (Tx + Rx)
KSZ8091MLX [27]	Microchip	72 ns (Tx) + 170 ns (Rx)
VSC8601/VSC8641 [28]	Microsemi	200 ns (Tx) + 380 ns (Rx)

[a] Measured.

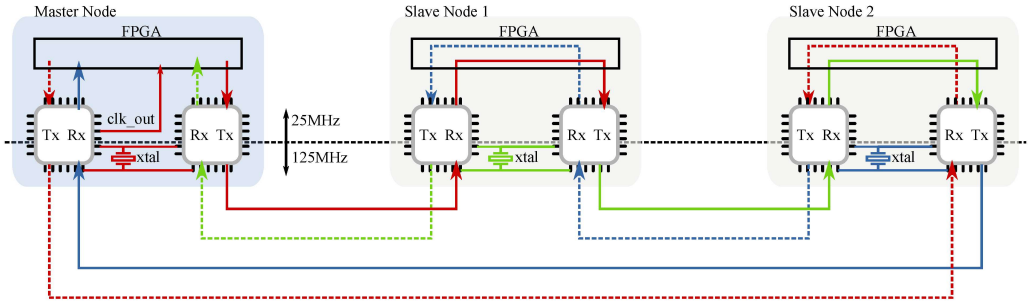
5.2. Physical Layer Synchronisation

High-speed links (a relative concept, but here we consider data rates equal to or higher than 100 Mbit/s), such as Ethernet, rely on synchronous modulation, in which the transmitter embeds the

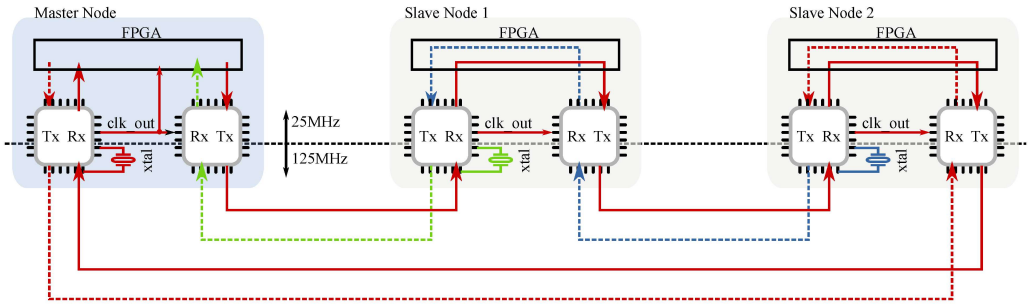
clock signal together with the data so the receiver can recover the clock and make it available for its internal logic. In multi-port nodes, as in ring networks, the PHY of the incoming port supplies the data, directly (broadcast) or indirectly (via the MAC), to the outgoing port(s) for transmission. Since the received clock of the incoming port may be different from the one employed for transmission in an outgoing port, each node has potentially different clock domains that must be crossed appropriately to guarantee reliable operation of the network.

A typical strategy for Clock Domains Crossing (CDC) is double flopping, which causes a variable delay from one to two clock cycles depending on the phase shift between the clocks [29]. The consequence of this phenomenon is that each node has a variable delay to forward the incoming packet if the PHY transmitter uses a clock that is not phase-locked to the recovered clock of the receiver. To make this clear, consider the ring network with three nodes depicted in Figure 7. The PHY integrated circuits have a clock input pin that serves as the reference for the internal and the transmitter logics. When a node has two ports, as in a ring, a common design practice is to use the same crystal as a reference for both PHYs, as in Figure 7a. In such case, the transmitters on a node are in the same clock domain, but, as the receiver locks to the clock of the neighbouring node, CDC is necessary. Certain PHY implementations (e.g., PHYs with RGMII interface) handle the CDC internally, while in others (e.g., PHYs with MII) the node logic must take care of proper clock domain crossing.

A different possibility uses PHYs that can output the recovered clock upon reception to a dedicated pin. Then, if this signal is forwarded to the PHY of the other port and employed as the reference for the transmitter, the network will have a single clock domain (Figure 7b). With all the PHYs operating in sync, we avoid CDC as well as the variable delay associated with it. Note that the implementation shown in Figure 7b can only synchronise the PHYs in one direction (clockwise in this case) because the first PHY in each node, i.e., the one at the left side of the node, has a local clock reference that it will use for transmitting data in the other direction (counter-clockwise in this case).



(**a**) Not synchronised.



(**b**) Synchronised.

Figure 7. Physical Layers synchronisation. The continuous lines represent one direction of the ring network and the dashed lines the other one.

Rigorously, a variable error will still be present in Fast Ethernet mode, because of the different rates between the media (125 Mbit/s, due to the 4B5B encoding) and the MII (25 Msymbols/s, 4-bit symbols), but it remains constant while the link is established [30].

6. Reducing the Reception Latency

To reduce the reception delay, we proposed two specific components that provide hardware support to the reception process and which can be easily instantiated in common FPGAs or employed in ASICs. The first such component is a *Packet Identifier* (PI) that will read all incoming packets and extract certain fields, such as EtherType, IP addresses, UDP port numbers, etc. The PI then writes those fields on a specific set of registers that are made available to the packet reception handler. The handler applies a set of filters with desired values and, in the case of match, it extracts the packet and provides it directly to the user application, thus bypassing the protocol stack processing.

The PI implementation is a long state machine that navigates through the packet fields, extracting the relevant information, and, when it is the case, comparing them with predefined values. For example, the state machine waits the field EtherType. If the frame is of Internet Protocol (IP) type (0x0800), the bit *IsIP* on the interface registers is set to one and the state machine follows the branch that corresponds to an IP frame.

Though the PI bypasses the protocol stacks for identified (e.g., real-time) packets, the main responsible for the reception latency, i.e., the data copying, is left untouched. The reception process copies the packet data at least twice, first from the MAC to the main memory, and then from this temporary position to its final location. Equally important, the data copying starts only after the MAC has completely received the frame and confirmed that the Frame Check Sequence is valid.

Under those circumstances, we propose a second hardware accelerator, the *Ethernet Direct Copy* (EDC), to further reduce the reception latency. It listens to data coming from the PHY and saves them directly to the main memory, to a position predetermined by the node user application layer.

The diagram of Figure 8 represents the EDC architecture. On the PHY side, the accelerator receives the clock, data valid and data signals. Once the data are valid, it waits for the start of frame delimiter to fill an asynchronous FIFO buffer, collecting the data into 32-bit words. The FIFO is responsible for the clock domain crossing of the data and, once it reaches a certain threshold signalled by the Almost Full signal, the accelerator starts writing the packet to the predefined location, either at the On Chip Memory or the external Dual Data Rate (DDR) Random Access Memory (RAM).

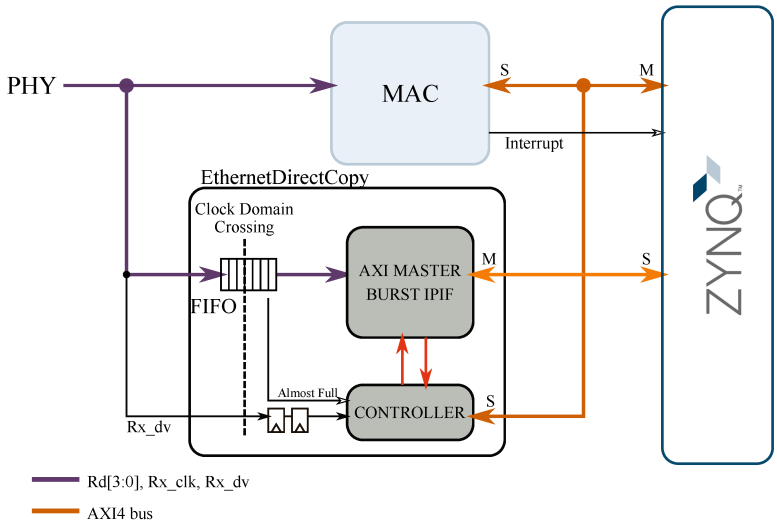


Figure 8. Ethernet Direct Copy hardware accelerator block diagram.

The main signals involved in the EDC operation are shown in Figure 9, captured with Vivado Integrated Logic Analyser. The rising edge of *mii_rx_dv* marks the start of packet reception ($t = 0$). The accelerator monitors the incoming data and waits for the start of frame delimiter to collect them into a 32-bit word and write it to the FIFO ($t = 92$), which happens every time the signal *fifo_WrEn* is high. Once the FIFO reaches the almost full level (*fifo_AlmostFull*, $t = 356$), the controller triggers a write burst (16 Bytes, in this case) using the AXI Master Burst IPIF. This happens when *ip2bus_mstwr_req* goes high and, after some handshake, data are read from the FIFO (*fifo_RdEn*). Note that the AXI master has a unique address phase (the master assigns an address when *axi_awvalid* is high) followed by a write phase in which several words are written to the memory in a burst (when *axi_wvalid* and *axi_wready* are high).

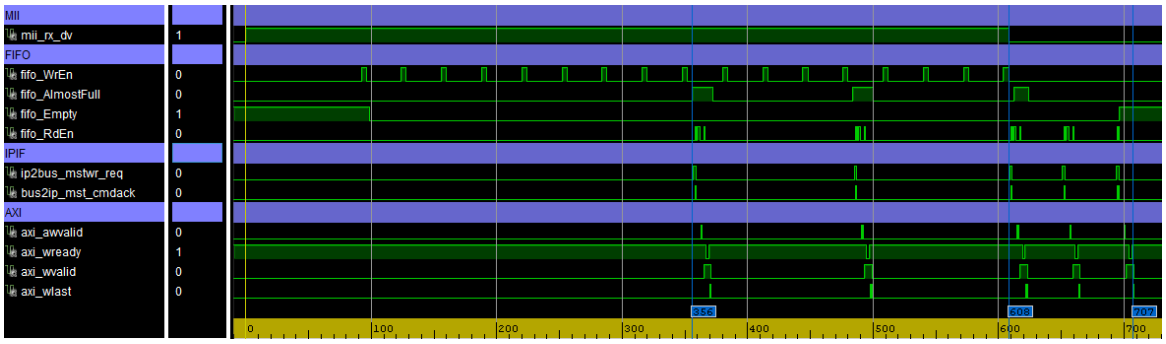


Figure 9. Capture showing EDC internal signals while receiving a packet with 64 Bytes. *X*-axis in samples, sampling period equal to 10 ns.

As the IPIF reads data from the FIFO faster than the PHY writes, the FIFO comes out of the almost full condition. While data are arriving, the FIFO gets filled again and triggers write transactions a number of times. Once the reception of the packet completes (*mii_dv* falling edge, $t = 608$), the controller performs the last bursts to empty the FIFO ($t = 707$).

Simultaneous to the EDC operation, the MAC receives the packet and stores it internally. If the MAC has identified a valid packet with a correct Frame Check Sequence (FCS), it flags an interrupt and the processor switches context to service it. At the entrance of the Interrupt Service Routine, the processor checks the fields extracted by the PI. Then, in the case of a high-priority packet, the processor jumps directly to the user callback function. Note that this sequence keeps the frame checking and the protocol stack processing, but the system does them on-the-fly together with moving data to the final destination. As a consequence, the time to complete the transfer and to enter the user callback is independent of the packet length. This is visible in the results shown in Figure 10 and, comparing with the other MACs, in Figure 5 (EDC).

Since the EDC writes data to a given memory region when packets arrive and we are unable to control when they will come, the system must take care of data consistency. One solution is to use a pool of memory regions where the EDC writes incoming data, and employ buffers descriptors (similar to the ones implemented in a DMA engine) to control the state of each region and hold a pointer to them. A simpler solution is to prevent the EDC from writing new data to the memory while the processor is using it or, in other words, to pause EDC while a packet is being processed. In this case, it is necessary to have a minimum interval between packets to ensure that, while the EDC is paused, no high priority packet will arrive and pass through the long latency path, i.e., MAC $\rightarrow$ protocol stack $\rightarrow$ user callback.

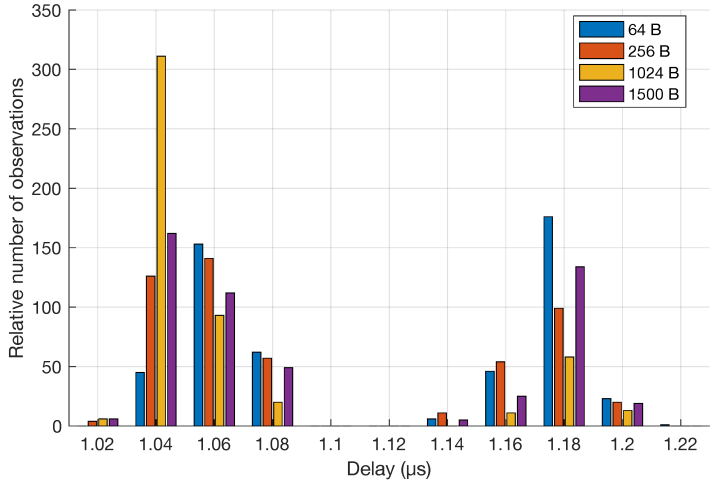


Figure 10. Delay to enter a UDP callback function after receiving packet using Ethernet Direct Copy accelerator.

7. Conclusions

As industrial applications demand higher update rates and the network size grows, the node-to-node data flow of the ring topology starts to impose important restrictions on the Minimum Cycle Time.

In this work, we discuss the most important aspects that influence the MCT of ring networks and explain why increasing the link capacity alone has limited benefits. With this in mind, we explore a time-triggered protocol that resorts to a broadcast and a local phase, such that the forwarding delay of the nodes in the broadcast phase can be reduced to the PHY delay plus a single clock cycle of the PHY/MAC interface.

Besides the reduction of the MCT, we present the concept, implementation details, and experimental results of hardware accelerators that minimise the reception delay of Ethernet packets. The main innovation of the proposed solution is to transfer the packet data directly to the main memory of the processor while it is being received, thus removing the delay dependence on packet length. The experimental results show the proposed solution to be 6.4 µs and 11 µs faster than Gigabit MAC (likely the most popular choice) when the packet size is 64 Bytes and 1024 Bytes, respectively.

The paradigm of transferring incoming data directly to the main memory, to a location defined by application software, is not constrained to implementations in FPGAs and can be incorporated into the design of high-performance MACs, where industrial embedded applications that demand minimal latency and MCTs in the order of tenths of microseconds can benefit from it.

Beyond enabling faster cycle times and lower end-to-end latencies, the proposed solution preserves Ethernet layer structure and keeps the implementation of higher layers in software. As real-time Ethernet protocols move to higher data rates, the minimisation of the internal delays becomes more important, thus the higher performance brought using the accelerator gains relevance.

Author Contributions: Conceptualisation, T.P.C. and L.A.; methodology, T.P.C. and L.A.; investigation, T.P.C. and L.A.; writing—original draft preparation, T.P.C.; writing—review and editing, T.P.C. and L.A.; and supervision, L.A.;

Funding: This research was funded by the Conselho Nacional de Desenvolvimento Científico e Tecnológico, Brazil, under the grant 233411/2014-3.

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

References

1. Vitturi, S.; Peretti, L.; Seno, L.; Zigliotto, M.; Zunino, C. Real-time Ethernet networks for motion control. *Comput. Stand. Interfaces* **2011**, *33*, 465–476. [CrossRef]
2. Mathe, L.; Burlacu, P.D.; Teodorescu, R. Control of a Modular Multilevel Converter With Reduced Internal Data Exchange. *IEEE Trans. Ind. Inform.* **2017**, *13*, 248–257. [CrossRef]
3. Yang, S.; Tang, Y.; Wang, P. Distributed Control for a Modular Multilevel Converter. *IEEE Trans. Power Electron.* **2018**, *33*, 5578–5591. [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. Corrêa, T.P.; Bueno, E.J.; Rodriguez, F.J. Communication Network Latency Compensation in a Modular Multilevel Converter. In Proceedings of the IEEE Energy Conversion Congress and Exposition (ECCE), Cincinnati, OH, USA, 1–5 October 2017. [CrossRef]
6. Briscoe, B.; Brunstrom, A.; Petlund, A.; Hayes, D.; Ros, D.; Tsang, I.; Gjessing, S.; Fairhurst, G.; Griwodz, C.; Welzl, M. Reducing Internet Latency: A Survey of Techniques and Their Merits. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2149–2196. [CrossRef]
7. Ramaswamy, R.; Ning, W.; Wolf, T. Characterizing network processing delay. In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), Dallas, TX, USA, 29 November–3 December 2004; Volume 3; pp. 1629–1634. [CrossRef]
8. Bertocco, M.; Narduzzi, C.; Tramarin, F. Estimation of the delay of network devices in hybrid wired/wireless real-time industrial communication systems. In Proceedings of the 2012 IEEE International Instrumentation and Measurement Technology Conference, Graz, Austria, 13–16 May 2012; pp. 2016–2021. [CrossRef]
9. Orfanus, D.; Indergaard, R.; Prytz, G.; Wien, T. EtherCAT-based platform for distributed control in high-performance industrial applications. In Proceedings of the 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), Cagliari, Italy, 10–13 September 2013; pp. 1–8. [CrossRef]
10. Cottet, D.; van der Merwe, W.; Agostini, F.; Riedel, G.; Oikonomou, N.; Rueetschi, A.; Geyer, T.; Gradinger, T.; Velthuis, R.; Wunsch, B.; et al. Integration technologies for a fully modular and hot-swappable MV multi-level concept converter. In Proceedings of the PCIM Europe 2015. International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management, Nuremberg, Germany, 19–20 May 2015; pp. 1–8.
11. Corrêa, T.P.; Almeida, L.; Peña, E.B. Hardware/Software Implementation Factors Influencing Ethernet Latency. In Proceedings of the 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), Porto, Portugal, 18–20 July 2018; pp. 323–328. [CrossRef]
12. Flatt, H.; Jasperneite, J.; Schewe, F. An FPGA based cut-through switch optimized for one-step PTP and real-time Ethernet. In Proceedings of the 2013 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS), Lemgo, Germany, 22–27 September 2013; pp. 7–12. [CrossRef]
13. Woods, J. Cut-Through Considerations and Impacts to Industrial Networks. In Proceedings of the IEEE 802.1 WG Meeting, Stuttgart, Germany, 15–18 May 2017.
14. Kleinrock, L. The latency/bandwidth tradeoff in gigabit networks. *IEEE Commun. Mag.* **1992**, *30*, 36–40. [CrossRef]
15. Insam, E. *TCP/IP Embedded Internet Applications*; Newnes: Oxford, UK, 2003.
16. Schlesinger, R.; Springer, A.; Sauter, T. New approach for improvements and comparison of high performance real-time Ethernet networks. In Proceedings of the IEEE Emerging Technology and Factory Automation (ETFA), Barcelona, Spain, 16–19 September 2014; pp. 1–4. [CrossRef]
17. Corrêa, T.P.; Almeida, L. Ultra Short Cycle Protocol for Partly Decentralized Control Applications. In Proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12–15 September 2017. [CrossRef]
18. Schlesinger, R.; Springer, A. VABS-A new approach for Real Time Ethernet. In Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society, Vienna, Austria, 10–13 November 2013; pp. 4506–4511.

19. Prytz, G. A performance analysis of EtherCAT and PROFINET IRT. In Proceedings of the IEEE International Conference Emerging Technologies & Factory Automation (ETFA), Hamburg, Germany, 15–18 September 2008; pp. 408–415.
20. Beckhoff Automation GmbH & Co. HW Datasheet ET1100—Section 3. Available online: https://download.beckhoff.com/download/document/io/ethercat-development-products/ethercat_et1100_datasheet_v2i0.pdf (accessed on 22 June 2017).
21. Beckhoff Automation GmbH & Co. EtherCAT Slave Controller—Section II. Available online: https://download.beckhoff.com/download/document/io/ethercat-development-products/ethercat_esc_datasheet_sec2_registers_2i9.pdf (accessed on 22 June 2017).
22. IEEE Standard for Ethernet. *IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012)*; IEEE: Piscataway, NJ, USA, 2016; pp. 1–4017. [CrossRef]
23. Texas Instrument. *Ethernet Media Access Controller (EMAC)/ Management Data Input/Output (MDIO) User Guide*; Texas Instrument: Dallas, TX, USA, 2012.
24. Corrêa, T.P.; Francisco, J.; Rodríguez, F.; Bueno, E.J. Model-Based Latency Compensation for Network Controlled Modular Multilevel Converters. *Electronics* **2018**, *8*, 22. [CrossRef]
25. Texas Instrument. DP83867E/IS/CS Robust, High Immunity, Small FormFactor 10/100/1000 Ethernet Physical Layer Transceiver. Available online: http://www.ti.com/lit/ds/symlink/dp83867is.pdf (accessed on 2 February 2017).
26. Mittra, K. Marvell PHYs for Low-Latency Industrial Ethernet. Available online: http://blogs.marvell.com/2016/10/marvell-phys-for-low-latency-industrial-ethernet/ (accessed on 2 February 2017).
27. Microchip. Datasheet KSZ8091MLX—10BASE-T/100BASE-TX Physical Layer Transceiver. Available online: http://ww1.microchip.com/downloads/en/DeviceDoc/00002297A.pdf (accessed on 21 March 2017).
28. VITESSE. Gigabit Ethernet PHY Device Latency. Available online: https://ethernet.microsemi.com/products/download.php?fid=4307&number\=VSC8224 (accessed on 2 February 2017).
29. Kilts, S. *Advanced FPGA Design: Architecture, Implementation, and Optimization*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2007.
30. Blattner, J.; Weibel, H. Study on propagation delay variation of 100BASE-Tx Ethernet PHY chips. In Proceedings of the Conference on IEEE-1588. Gaithersburg, MD, USA, 27–29 September 2004.