Hindawi
International Journal of Reconfigurable Computing
Volume 2023, Article ID 9222318, 36 pages
https://doi.org/10.1155/2023/9222318

Research Article

Open-Source Ethernet MAC IP Cores for FPGAs: Overview and Evaluation

Christian Fibich, Patrick Schmitt, Roland Höller, and Peter Rössler

University of Applied Sciences Technikum Wien, Department of Electronic Engineering, Vienna, Austria

Correspondence should be addressed to Christian Fibich; fibich@technikum-wien.at

Received 31 October 2022; Revised 27 April 2023; Accepted 3 May 2023; Published 23 May 2023

Academic Editor: Volodymyr Kindratenko

Field-Programmable Gate Arrays (FPGAs) can be found in an increasing number of application domains, such as the telecom industry, the automotive electronics sector, or automation technology as well as in the area of reconfigurable computing. In recent years, it can be observed that the open-source idea which is known from the software domain for a long time also became popular in the world of hardware and FPGA design. In the era of the Internet of Things, many of today's electronic devices implement some kind of network interface with Ethernet being known as one of the most widely used network standards. Thus, there is consequently a high demand on available Ethernet implementations for FPGA platforms. The goal of this work is to survey available open-source Ethernet MAC IP cores, evaluate existing designs in terms of performance, resource utilization, code quality, or maturity, and to present and summarize the evaluation results herein. Furthermore, advantages of commercial solutions and related publication work are discussed. To the authors' best knowledge, this is the first publication that evaluates and compares existing open-source Ethernet MAC IP cores on a large scale. This work should help designers to select an appropriate open-source Ethernet MAC for an FPGA design and shows possible pitfalls and things to pay attention when using an open-source IP core in general. Finally, the authors would like to show that the open-source community can be also very helpful in the world of hardware in terms of design reuse or time to market.

1. Introduction

Today, Ethernet is by far one of the most important, if not the most important computer network technology [1, 2]. It was developed in 1973 at Xerox Palo Alto Research Center (PARC) and has been approved as the IEEE 802.3 standard in 1983. Since then the original Ethernet technology was further developed to a great extent, and today the IEEE 802.3 standard includes numerous supplementary sections resulting in thousands of pages of documentation. During the last decades, Ethernet has become the dominant LAN technology to interconnect computers in, e.g., homes, office buildings, or at university campuses. Over the years and apart from its intended purpose, Ethernet is increasingly used in more and more other application fields such as telecommunications [3], the automotive sector [4], industrial automation [5], and even avionics [6]. The efforts to make use of Ethernet in industrial applications by replacing traditional fieldbus technologies are commonly summarized under the term "Industrial Ethernet" [7]. However, this term does not refer to a single standard but rather to various approaches aiming to introduce determinism and real-time behavior, rugged connectors, or networking infrastructure with extended temperature range in order to work in harsh environments. Examples for Industrial Ethernet protocols are PROFINET, EtherCAT [7], and EtherNet/IP (where "IP" refers to "Industrial Protocol," see [8]).

Ethernet basically covers Layer 1 and Layer 2 of the Open Systems Interconnection (OSI) model including specifications of the communication media (see Figure 1). The classic Ethernet implementation made use of a coaxial cable while for recent variants, twisted pair and fiber optic links are the most common types of media. Typical transmission speeds in today's Ethernet implementations are 10, 100, and 1000 Mbit/

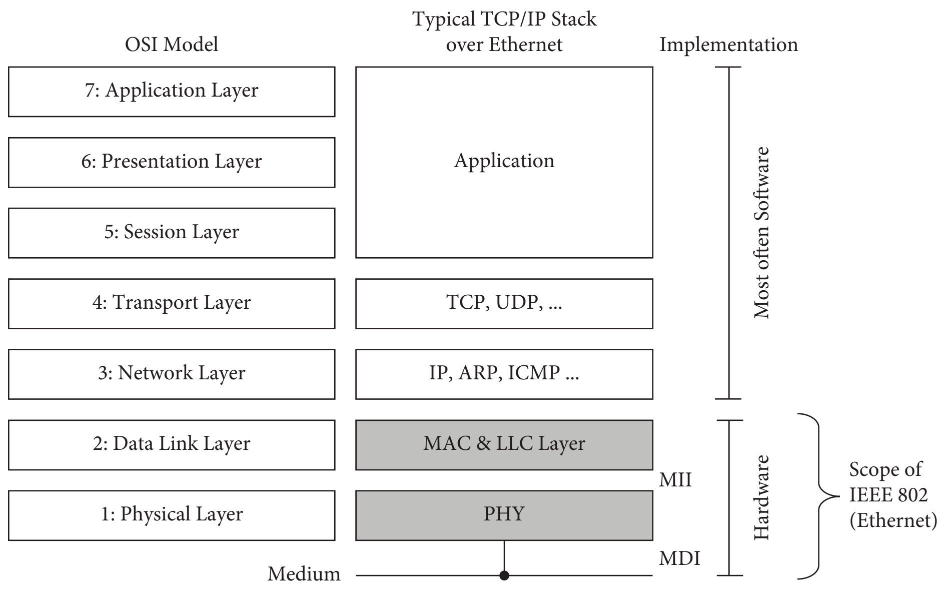

Figure 1: Typical implementation of TCP/IP stack over Ethernet.

s. However, bitrates of 10, 40, and 100 Gbit/s are also standardized since some years and the IEEE 802.3bs physical layer specification (adopted in 2017) supports a transmission speed of up to 400 Gbit/s. The physical layer of Ethernet is often abbreviated as Physical Layer (PHY) while the data link layer, which includes both the Medium Access Control (MAC) and the Logical Link Control (LLC) sublayer, is commonly referred to as Medium Access Control (MAC). Both PHY and MAC are implemented in a hardware device while upper protocol layers sitting on top of Ethernet (e.g., TCP, UDP, IP, ARP, and ICMP) are most often implemented as a software stack running on a CPU or microcontroller (however, hardware implementations for the previously mentioned upper protocol layers are also available, e.g., for applications with demands on high throughput and/or low latency [9]). The Ethernet PHY, the MAC, and the CPU (processing the upper protocol layers) can be either separate devices or a single-chip solution (e.g., a microcontroller that integrates both the PHY and the MAC on-chip). If the PHY and the MAC are individual devices, they are interconnected over the so called Media Independent Interface (MII) whereby different variants of this interface exist, such as Reduced Media Independent Interface (RMII) or Gigabit Media Independent Interface (GMII). The physical media is connected to the PHY over the Media Dependent Interface (MDI).

Due to the widespread use of Ethernet, there is consequently a high demand on available Ethernet implementations. When an Ethernet interface is required for a Field-Programmable Gate Array (FPGA) or Application-Specific Integrated Circuit (ASIC) design, a common method is to integrate the MAC and the CPU (that processes, e.g., a TCP/IP stack) on-chip. For this reason, Ethernet MACs are available from FPGA vendors, Intellectual Property (IP) providers, and other companies in form of an IP core. However, existing commercial solutions come with some limitations:

(i) IP cores from FPGA vendors are often technology-dependent which makes it difficult to port the design to devices from other FPGA vendors.

(ii) IP cores that are provided as a netlist cannot be modified by the user, making it impossible to change the existing design (in order to adjust the receive/transmit buffer sizes, replace the on-chip bus interface, etc.), add additional features (e.g., to move functionality to dedicated offload engines), or to fix bugs.

(iii) Finally, license fees have to be paid for most commercial IP cores.

Fortunately, the open-source idea that is known from the software domain for a long time also became popular in the world of hardware design since some years [10]. Thus, the usage of an open-source Ethernet MAC IP core can be a solution to overcome the limitations of commercial IP cores mentioned previously which finally was the motivation for the authors of this work. The goal of this paper is to survey available open-source Ethernet MAC IP cores, evaluate existing designs in terms of performance, resource utilization, code quality, or maturity, and to present and summarize the evaluation results herein. The following sections are structured as follows. Section 2 provides an overview of existing work related to the scope of this publication. Section 3 presents results of our survey on open-source Ethernet MAC IP cores including information such as Internet source and license model for each IP core, design language, supported bitrates, PHY and application interfaces, available documentation and testbenches, existing reference implementations, and features like support for DMA transfers, VLAN tagging, or Precision Time Protocol (PTP). Furthermore, the consumed FPGA resources are compared in this section in terms of logic resources, memory, and other technology-specific building blocks

(Phase-Locked Loops (PLLs), delay elements, etc.). Here, the synthesis and linting reports are discussed on a large basis since they reflect the code quality of the IP cores. Eight projects have been selected for a closer evaluation whereby details of the selection process are outlined in Section 4. In Section 5, the evaluation and measurement setup for the selected projects (FPGA platforms, used wrappers and hardware environment to embed the IP cores, Ethernet tools, etc.) is presented. Finally, the results of the various measurements (network throughput, latency, and packet loss at different frame lengths) are provided in Section 6. The paper is concluded by a discussion of the evaluation results and provides a brief outlook to our future research work in Section 7.

2. Related Work

Existing publications that are related to this work cover, e.g., Ethernet MAC designs for FPGAs, hardware implementations of network protocols such as IP, UDP, or ARP, and papers where Ethernet MACs only act as a use case to investigate other research topics. For example, Qian et al. introduced a Verilog implementation of a 10/100 Mbit/s Ethernet controller in a short paper [11]. Unfortunately, numbers concerning network performance or resource consumption are missing. Yi et al. presented an implementation of a Ten Gigabit Ethernet MAC in [12]. While less details on the actual implementation of the MAC are given, the focus of the paper is, however, on a new CRC calculation method. Another 10 Gbit/s Ethernet MAC was described in [13] by Xiao et al. that was implemented on a Xilinx Virtex-6 FPGA device. In the bachelor thesis [14] from the University Ilmenau/Germany, the author Kerling provided a lot of implementation details of an Ethernet MAC coded in VHDL and targeted at FPGAs, supporting link speeds of 10, 100, and 1000 Mbit/s. The design is publicly available under an open-source license. A few Ethernet MACs are listed in the thesis under related work, but they are all commercial IP cores.

A number of existing publications propose hardware implementations of higher level protocols such as IP or UDP built on top of Ethernet. In [15], the Universidad Autónoma de Madrid and the ETH Zürich introduced "Limago," an FPGA-based open-source implementation of a TCP/IP stack operating at 100 Gbit/s which, according to the authors, is the first complete description of an FPGA-based TCP/IP stack at this bitrate. The design is based on Vivado-HLS and makes use of a commercial Ethernet MAC. The paper includes results from performance measurements of the network stack as well as the resource consumption for different configurations of the framework. A similar TCP/IP implementation for FPGAs called SiTCP was presented by the University of Tokyo in [16]. In order to be vendor-independent, here the author argues against using a hard-macro for the Ethernet MAC and therefore makes use of a custom built MAC. The Technical University of Munich presented an UDP/IP core for FPGAs based on a hard-wired Ethernet MAC from Xilinx in [17]. Measurements have been performed by the authors concerning network throughput and packet loss. Moreover, a comparison of the consumed FPGA resources with a UDP/IP stack from Löfgren et al. [18] was done. In [19], Sütő and Oniga presented a custom built Ethernet MAC with low resource usage that includes hardware implementations of ARP and DHCP. Here the intended use case is communication of sensor values from an embedded sensor node and shall contribute to the "Internet of Things."

The "Corundum" project by the University of California is an open-source FPGA-based prototyping platform for network interface development at up to 100 Gbit/s and beyond [20]. The platform has an even broader focus than the previously mentioned work since it includes 10 G/25 G/100 Gbit/s Ethernet MACs, PCI Express Gen 3, a custom PCIe DMA engine, and high-precision IEEE 1588 PTP timestamping. It makes use of the Xilinx Ethernet CMAC hard core for 100 G Ethernet and an own FPGA-based implementation for <100 Gbit/s (which is not described in detail by the authors). The conference paper [21] from Santos et al. has yet another scope and describes the FPGA-based architecture of a modified Ethernet switch providing real-time communication based on the Flexible Time-Triggered paradigm. It utilizes the Xilinx Tri-Mode Ethernet MAC soft core which can operate at 10/100/1000 Mbit/s. Other publications such as [22, 23] implement an Ethernet MAC only as a use case while the focus of research is verification.

In summary and to the best of our knowledge, no publication could be found that compares and evaluates available open-source Ethernet MACs on a large scale. That, as well as providing general insights into the potential usefulness of open-source IP cores, was our primary motivation to write this survey and evaluation paper.

3. Overview of Open-Source Ethernet MAC IP Cores

The first step of the overview and evaluation of open-source Ethernet MAC IP cores described in this paper is to gain a comprehensive overview of open-source projects available in this context. Three major sources for finding these projects can be mentioned:

(1) A traditional source for open-source IP is the web platform *opencores.org*. Founded in 1999, it provides a directory and source code versioning and hosting of various open-source IP cores targeting FPGAs and ASICs alike. In recent years, a subset of cores has been mirrored to other sites. For example, *freecores.github*.io is a project to mirror the source code from *opencores.org* to *github.com*.

(2) Due to the popularity of git-based source code hosting platforms such as *github.com*, a number of analyzed IP cores have also been identified that are hosted on these platforms. These projects are harder to find, having to rely on the site's search function or Internet search engines or third-party directories such as the former platform *librecores.org* that was closed in October 2022 (an archived version can be

found here: https://web.archive.org/web/20220626133000/https://www.librecores.org/project/list (accessed: May 5, 2023)).

(3) Some projects are only available from the website or private source code hosting instance of the respective author or core vendor. Identification of these projects wholly relies on the indexing of an Internet search engine or mention of the location in a publication (e.g., [14]).

3.1. Identified Projects. The projects that have been identified at the time of writing identified using the sources mentioned above are listed in Table 1 in alphabetical order. As some of the identified projects are quite popular, often not only the original repository of the core appears in search results, but also other projects that include these cores in their designs. These projects are not shown in Table 1, which aims at showing the original set of Ethernet MAC cores found.

In addition to the identified projects, Table 1 also provides the version of the IP core (git commit, SVN revision, or version number), as well as the release date of the version evaluated in this work. Furthermore, basic information about the core and its features is provided.

The context in which an IP core can be used in a digital design is related to the license under which the source code is released. While most of the identified cores are released under a traditional open-source software license such as GNU General Public License (GPL) or Berkeley System Distribution (BSD), in recent years licenses especially suitable for open-source hardware such as the CERN Open Hardware License (OHL), the Solderpad License, or the NetFPGA Hardware-Software license have become available. The license, for example, impacts if the core can be used commercially at all and which parts of the source code (if any) need to be published if it is used in a commercial product. A discussion of licenses for open-source hardware can be found in [42].

The language which is used to describe the hardware of a core impacts the ease of integration into the context of a larger project. Most identified cores are described in a "traditional" hardware description language such as VHDL or Verilog. The two exceptions are the two projects *An Ethernet Controller* and *Litex Liteeth*. The former is described in *Chisel*, a hardware description language based on the *Scala* programming language. Chisel is, for example, used in the Rocket Chip Generator (https://github.com/chipsalliance/rocket-chip, accessed: May 5, 2023) to describe the RISC-V Rocket CPU core. Using the Scala Build Tool (sbt), a synthesizable Verilog representation of the design is generated. In contrast, Litex Liteeth is described in *Migen*, a hardware description system and core library written in Python, that also generates synthesizable Verilog. Apart from the identified Ethernet IP core, the Litex project (https://github.com/enjoy-digital/litex, accessed: May 5, 2023) provides a System-on-Chip (SoC) build system and core library (e.g., DRAM, PCIe, and SATA cores) written in Migen.

Furthermore, the supplementary material provided by the core's repository is detailed in the columns "Testbench", "Documentation", and "Reference Implementation" of Table 1.

Providing a testbench with a core allows the potential user of a core to quickly bring up and confirm the functionality of the core in simulation. Furthermore, it may serve as an indicator that some thought has been given to the verification of the core by the author(s).

If a project includes reference implementations for one or more FPGA development boards, this can serve as an indicator that the project is indeed synthesizable and was at some point tested in actual hardware by the author(s). Furthermore, important implementation details such as how to integrate the core into a functional system and which parts of the project need to be ported for a specific FPGA technology can be learned from such an implementation.

The documentation a core provides has been classified into three categories:

(i) Code comments (CC) document the source code itself inline.

(ii) Readme (*R*) files are often short text files describing the most important aspects of a core (e.g., intended application, FPGA family, and build system). In projects hosted on *github.com* or a similar system, these files are rendered as a project's "landing page."

(iii) Some projects also provide long-form (LF) documentation, either in the form of a user manual, specification document(s), or both.

Availability of high-quality documentation significantly reduces the time needed until a project can be used productively. Otherwise, this information needs to be extracted from example implementations, testbenches, or the source code of the core itself.

The principal set of features along which the identified IP cores are classified is shown in Table 2.

Concerning the supported communication speed, three classes were introduced: (1) 10/100 Mbit/s, the "traditional" Ethernet speed, (2) 1 Gbit/s, a standard speed nowadays, and (3) >1 Gbit/s, e.g., 10, 25, or more Gbit/s as a *fast* communication speed. While some cores only support one speed class, others support multiple standards. The supported speed is closely related to the supported MII variant to interface to an Ethernet PHY. For example, the original MII (4 bit parallel data, 25 MHz clock frequency) was introduced in the 100 Mbit/s Fast Ethernet standard. RMII falls into the same Ethernet speed class but doubles the frequency to 50 MHz in order to halve the number of required data signals. Gigabit Ethernet requires a different MII variant, such as GMII (8 bit parallel data, 125 MHz clock frequency), the double-data-rate Reduced Gigabit Media Independent Interface (RGMII) (4 bit parallel data, 125 MHz clock frequency, one nibble transferred per clock edge), or the 625 MHz double-data-rate Serial Gigabit Media Independent Interface (SGMII). Even faster Ethernet speeds require even more complex interfaces such as the 156.25 MHz double-data-rate 32-bit parallel Ten Gigabit

Table 1: Overview of identified open-source Ethernet MAC IP cores.

Project	Analyzed version	Version date	License	Language	Documentation[a]	Testbench	Ref. impl.
An Ethernet Controller [24]	b2a334d (git)	2019-05-24	BSD 2-clause	Chisel	CC, *R*	Yes	—
Ariane-Ethernet [25]	ff9710f (git)	2019-02-06	MIT	System Verilog	—	—	—
Gaisler GRETH [26]	2020.4-b4261	2020-12-15	GPL	VHDL	CC, LF	—	Yes
LeWiz LMAC1 [27]	ac5c2ef (git)	2019-05-17	LGPL	Verilog	CC, LF, *R*	Yes	Yes
LeWiz LMAC2 [28]	07725d4 (git)	2019-01-25	LGPL	Verilog	CC, LF, *R*	Yes	Yes
LeWiz LMAC3 [29]	852c99b (git)	2019-07-31	LGPL	Verilog	CC, LF, *R*	Yes	Yes
Litex Liteeth [30]	435c67d (git)	2021-05-27	BSD 2-clause	Migen	CC, *R*	Yes	Yes
NFMAC10G [31]	c21bfea (git)	2016-02-25	NetFPGA	Verilog	CC, *R*	Yes	—
Opencores Ethernet Tri Mode [32]	33 (svn)	2009-03-09	LGPL	Verilog	CC, LF	Yes	—
Opencores Ethmac [33]	368 (svn)	2012-02-14	LGPL	Verilog	CC, LF, *R*	Yes	—
Opencores Gbiteth [34]	3 (svn)	2013-08-23	LGPL	VHDL	CC	—	—
Opencores Minimac [35]	3 (svn)	2010-08-24	GPL	Verilog	CC, LF	Yes	—
Opencores XGE_LL_MAC [36]	2 (svn)	2012-12-01	LGPL	Verilog	CC	—	—
Opencores XGE_MAC [37]	31 (svn)	2017-03-15	LGPL	Verilog	CC, LF, *R*	Yes	—
P. Kerling Ethernet MAC [38]	b4cf145 (git)	2015-09-08	BSD-derived	VHDL	CC, LF, *R*	Yes	—
Verilog-Ethernet [39]	b09e01b (git)	2021-06-03	MIT	Verilog	CC, *R*	Yes	Yes
WGE 100 [40]	4c5ec19 (git)	2012-01-07	BSD 3-clause	Verilog	CC, LF	Yes	—
WhiteRabbit [41]	69cc4cc3 (git)	2017-12-18	LGPL	VHDL	CC, LF	Yes	Yes

[a]CC = code comments, *R* = readme file, and LF = long-form documentation.

Table 2: Features of identified open-source Ethernet MAC IP cores.

Project	Speed	PHY interface	Control interface	Data interface	Primitives
An Ethernet Controller [24]	10/100	MII	OCP	OCP-master	—
Ariane-Ethernet [25]	10/100, 1 G	RGMII	Other address/data bus	Other address/data bus	Xilinx
Gaisler GRETH [26]	10/100	MII, RMII	APB	AHB-master	—
LeWiz LMAC1 [27]	10/100, 1 G	XGMII	RTL	AXI-Stream, RTL + int. FIFO IF	—
LeWiz LMAC2 [28]	>1 G	XGMII	RTL	AXI-Stream, RTL + int. FIFO IF	—
LeWiz LMAC3 [29]	>1 G	CGMII	RTL	AXI-Stream, RTL + int. FIFO IF	—
Litex Liteeth [30]	10/100, 1 G	MII, {G, RG} MII	Wishbone	Wishbone-slave	Xilinx
NFMAC10G [31]	>1 G	XGMII	RTL	AXI-Stream	—
Opencores Ethernet Tri Mode [32]	10/100, 1 G	MII, GMII	Other address/data bus	Int. FIFO IF	Intel, Xilinx
Opencores Ethmac [33]	10/100	MII	Wishbone	Wishbone-master	—
Opencores Gbiteth [34]	10/100, 1 G	RGMII	Wishbone	Wishbone-master	Intel
Opencores Minimac [35]	10/100	MII	Other address/data bus	Wishbone-master	—
Opencores XGE_LL_MAC [36]	>1 G	XGMII	RTL	Ext. FIFO IF	—
Opencores XGE_MAC [37]	>1 G	XGMII	Wishbone	Int. FIFO IF	—
P. Kerling Ethernet MAC [38]	10/100, 1 G	MII, GMII	RTL	Ext. FIFO IF, int. FIFO IF	Xilinx
Verilog-Ethernet [39]	10/100, 1 G, >1 G	MII, {G, RG, XG}MII	RTL	AXI-Stream	Intel, Xilinx
WGE 100 [40]	10/100, 1 G	MII, GMII	RTL	Ext. FIFO IF	Xilinx
WhiteRabbit [41]	1 G	PCS	Wishbone	Wishbone	—

Media Independent Interface (XGMII). An exception to the support of one or more MII variants is the WhiteRabbit project that only supports direct connection to a Physical Coding Sublayer (PCS) core.

Furthermore, the interface that a core provides to access its features is an important detail to consider when integrating the core into a planned or existing system. Two kinds of tasks for the core's interface have been considered in this work: the *control* part of the interface that is used to configure the core or alter its behavior (e.g., setting MAC address filtering), and the *data* part of the interface that is used to transfer data to be sent into the core and data received out of the core. In some cases, both of these parts are actually integrated into one single interface. This is, for example, the case for a core that provides a single memory mapped address space which allows access to configuration registers and data buffers via an on-chip bus system. In other cases, a separate interface exists for the two tasks. In the "Interface" columns of Table 2, apart from standard on-chip bus interfaces such as Wishbone, Advanced Microcontroller Bus Architecture (AMBA) Advanced High-Performance Bus (AHB), AMBA Advanced Peripheral Bus (APB), AMBA Advanced eXtensible Interface Bus (AXI), and Open Core Protocol (OCP), also non-standard interfaces had to be considered. In this context, Register Transfer Level (RTL) stands for discrete ports that must be driven by external logic but do not follow a well-defined interface. A core is classified as having an "External FIFO Interface" if it exposes a read/write request signal along with a read/write data signal for connection of an external FIFO, while it is classified as having an "Internal FIFO Interface" if it exposes these signals for external access to a FIFO contained in the core. Additionally, a core is classified as having an "other address/data bus" if it does not use a standard on-chip bus but exposes a generic address and data bus as well as control signals for external access. Finally, some cores are able to retrieve frames to be sent, and store received frames, in external memory acting as a bus master, i.e., provide Direct Memory Access (DMA) capabilities. In these cases, the control interface is used to establish DMA descriptors that the core then uses to access the correct memory locations. These cores are described as having a "master" interface.

While a standardized bus interface may be preferred when integrating a core into a complex or CPU-centered SoC, other interfaces such as a plain FIFO or streaming interface may be easier to interface from arbitrary logic that would else potentially be required to generate sequences of bus transactions just to bring up the core for transmission or reception.

The (technology) Primitives column of Table 2 shows for which FPGA vendor—if any—a core instantiates technology primitives. If such primitives are instantiated in the RTL description of a core, these need to be replaced when porting the core to another FPGA family or vendor. For some technology-dependent components (e.g., BRAM), current synthesis tools are able to infer these components from an RTL description. However, when using more complex components such as Double Data Rate (DDR) components or transceivers necessary for high-speed Ethernet standards, direct instantiation of these primitives becomes necessary.

Table 3 provides further insight into the technical features the cores provide. It can be seen that almost all cores provide checking and insertion of the Ethernet Frame Checksum (FCS) as well as First-In-First-Out (FIFO) buffers for decoupling the MAC from the network and/or from the user logic.

While often required, an implementation for Management Data Input Output (MDIO) for configuration and monitoring of the PHY is only provided by a subset of the identified cores. If MDIO is not present, the user must either supply an external implementation or use the PHY's power-up defaults, which may not be possible in every case.

Several cores provide DMA support to offload the task of reading frames to be transmitted and writing received frames into memory from the CPU. In these cases, the cores often require the setup of elaborate *DMA descriptor* systems that point to locations in memory where the core should place received frames or fetch frames to be transmitted from. Further offloading is provided by those cores that allow filtering incoming frames for specific MAC addresses or allow insertion of the host's MAC address into transmitted frames.

As an alternative to DMA, some cores include internal memory-mapped RAM buffers themselves that can be read from or written to from a bus interface.

A small subset of the cores provides special features such as support for VLAN tagging or the PTP for clock synchronization.

While Ethernet standards for transmission speeds lower than 10 Gbit/s include half-duplex operations, only three cores support this mode of operation. This includes handling the carrier sense (CRS) and collision (COL) signals generated by the PHY, as well as implementing Carrier Sense Multiple Access/Collision Detection (CSMACD). However, as nowadays the vast majority of Ethernet networks employ switches and full-duplex operations, these features were not evaluated in hardware.

Finally, the (default) width of the data path of the analyzed cores is provided in Table 3. The majority of the analyzed cores use a data path width geared towards modern on-chip bus systems and CPU interfaces of 32 or 64 bit. As an exception, P. Kerling's MAC and some variants of the Verilog-Ethernet project's MAC provide byte-wise access to the transmitted or received data. The default network-side interface supported by the WhiteRabbit wr-endpoint core is a 16-bit variant of PCS. Finally, the LMAC3 project—geared towards data rates of up to 100 Gbit/s—provides a 256-bit wide interface.

3.2. Out-of-Context Synthesis. In order to confirm completeness and basic synthesizability of the identified IP core projects, the cores were synthesized using Xilinx's Vivado 2019.1 FPGA implementation tool. For cores that allow significantly different variants (e.g., different configurations, speeds, and interfaces), multiple synthesis runs were performed. These synthesis runs were done in Out-of-Context

Table 3: Extended features of identified open-source Ethernet MAC IP cores.

Project	MDIO	Default datapath width (bits)	DMA	RX FIFO	TX FIFO	RX RAM buffer	TX RAM buffer	FCS	VLAN	PTP support	MAC address filtering	MAC address insertion	Half duplex
An Ethernet Controller [24]	—	32	Yes	—	—	—	—	—	—	—	—	—	—
Ariane-Ethernet [25]	Yes	64	—	Yes	Yes	Yes	Yes	Yes	—	—	Yes	—	—
Gaisler GRETH [26]	Yes	32	Yes	Yes	Yes	—	—	Yes	—	—	Yes	—	Yes
LeWiz LMAC1 [27]	—	64	—	Yes	Yes	—	—	Yes	Yes	—	Yes	Yes	—
LeWiz LMAC2 [28]	—	64	—	Yes	Yes	—	—	Yes	Yes	—	Yes	Yes	—
LeWiz LMAC3 [29]	—	256	—	Yes	Yes	—	—	Yes	Yes	—	Yes	Yes	—
Litex Liteeth [30]	Yes	32	—	Yes	Yes	Yes	Yes	Yes	—	—	—	—	—
NFMAC10G [31]	—	64	—	Yes	Yes	—	—	Yes	—	—	—	—	—
Opencores Ethernet Tri Mode [32]	Yes	32	—	Yes	Yes	—	—	Yes	—	—	Yes	Yes	Yes
Opencores Ethmac [33]	Yes	32	Yes	Yes	Yes	—	—	Yes	—	—	Yes	Yes	Yes
Opencores Gbiteth [34]	Yes	32	Yes	Yes	—	Yes	Yes	Yes	Yes	—	Yes	—	—
Opencores Minimac [35]	Yes	32	Yes	Yes	Yes	—	—	—	—	—	—	—	—
Opencores XGE_LL_MAC [36]	—	64	—	—	—	—	—	Yes	—	—	—	—	—
Opencores XGE_MAC [37]	—	64	—	Yes	Yes	—	—	Yes	—	—	—	—	—
P. Kerling Ethernet MAC [38]	Yes	8	—	Yes	Yes	—	—	Yes	—	—	Yes	Yes	—
Verilog-Ethernet [39]	—	64, 8	—	Yes	Yes	—	—	Yes	—	Yes	—	—	—
WGE 100 [40]	Yes	32	—	Yes	Yes	—	—	Yes	—	—	Yes	—	—
WhiteRabbit [41]	—	16	—	Yes	Yes	—	—	Yes	Yes	Yes	—	—	—

(OOC) mode, which is Vivado's term for performing a synthesis run only, with no required physical (e.g., pin) constraints and without insertion of IO buffers. The result of this synthesis run is a technology-dependent gate-level netlist that can—in an actual design flow—be instantiated in a hierarchical design.

The OOC synthesis runs allow to judge if a core is synthesizable as-is, or, alternatively which components of the core need to be ported to the specific FPGA technology in order to be synthesizable. Furthermore, basic resource usage after synthesis allows comparison between the cores and identification of possible bugs (e.g., unintended instantiation of latches). Finally, it provides access to Vivado's linting capabilities. For example, its (critical) synthesis warnings can be analyzed and used as an indicator of basic code quality. In the context of Ethernet, also the results of Vivado's Clock Domain Crossing (CDC) analysis tool are relevant because most cores incorporate CDCs from the interface clock domain to the MII clock domain and vice versa.

As a target technology for the preliminary OOC synthesis, the widely used Artix 7 device family by Xilinx was chosen. This technology provides 6-input fracturable Look-Up Tables (LUTs), 18 Kbit and 36 Kbit memory blocks, and 25×18 DSP blocks.

3.2.1. Synthesized Cores and Variants. In order to perform OOC synthesis for each of the cores listed in Table 1, Hardware Description Language (HDL) wrappers that instantiate the respective core and set top-level parameters were implemented. The parameter values were chosen to reflect default values set either in the core's top-level module or mentioned in the documentation. In those cases where a core was available in different variants (e.g., different interface types or Ethernet speeds), multiple variants were synthesized. This concerns the following cores:

(i) LeWiz's LMAC cores provide a native interface to internal receive and transmit FIFOs and an AXI-Stream interface. The LMAC1 core was thus synthesized in the native variant, referred to as LMAC_CORE_TOP after the top-level module that was synthesized. Additionally, the LMAC1_CORE_AXIS variant contains the AXI-Stream interface mentioned above.

Additionally, LeWiz's LMAC cores contain a FIFO implementation inferred from HDL. However, it was seen during synthesis that this description is not completely understood as intended by Xilinx Vivado, which implements the FIFO's logic in flip-flops and LUTs instead of more suitable memory resources. Thus, each of the variants of the LMAC1 core mentioned above was synthesized in two variants: one with the original inferred FIFO implementation and one where this implementation has been replaced by a macro provided by Xilinx (using the Xilinx Parameterized Macro (XPM)-FIFO core).

The remaining cores provided by LeWiz (LMAC2 and LMAC3) were only synthesized in variants that use the native interface and the XPM-based FIFO implementation.

(ii) Litex's Liteeth provides several different interfaces to PHYs (and thus also different Ethernet speeds) and different application interfaces as well. Both variants provide internal, memory-mapped data buffers for received and transmitted Ethernet frames that can be read and written via Wishbone.

(iii) The 10 Gbit/s project NFMAC10G provides a bare variant that places tight constraints on the interface to external logic and a more comfortable user interface. According to NFMAC10G's documentation, this interface allows for flow control on the receive side and more flexible interfaces on the transmit side. Additionally, this interface also filters out received frames with invalid FCSs. The bare variant is referred to as *nfmac10g*, and the variant with the more convenient interface is referred to as *nfmac10g_user_intf* (see Table 4).

(iv) In a similar way, P. Kerling's Ethernet MAC provides a bare variant that requires external FIFOs and one where these FIFOs are implemented internally, and the read and write port to them is exposed on the interface.

(v) The Verilog-Ethernet project provides a large variety of MACs supporting different PHY interfaces. Five different variants, instantiating a subset of these interfaces, were synthesized. All of the variants provide AXI-Stream access to internal receive and transmit FIFOs.

(vi) WhiteRabbit is a complete system for highly accurate clock synchronization for data transfer and control at CERN. The WhiteRabbit code repository provides a variety of different cores to implement this system, among others a complete SoC implementing Network Interface Controller (NIC), containing elaborate filtering and even a Lattice Mico32 CPU core. Only the MAC implementation of this project was synthesized for this work (*wr_endpoint*).

In addition, synthesis constraint files (.xdc files) for Xilinx Vivado were implemented that define clocks for the top-level clock inputs in order to enable Vivado to perform CDC analysis. To that end, except if noted otherwise, each clock input was considered to drive a clock that is asynchronous to all other clock inputs. Additionally, this constraint file allows to constrain clock multiplexers. Several of the analyzed cores (cores capable of 10/100 Mbit/s and 1 Gbit/s) use clock multiplexers to select between the 25 MHz MII transmission clock generated by the PHY in 10/100 Mbit/s mode and the 125 MHz GMII gigabit transmission clock generated by the MAC in 1 Gbit/s mode. Vivado requires manual constraining of clock multiplexers in order to perform correct case analysis of the propagated clocks [43].

Table 4: OOC synthesis resource results.

Project	Variant	LUTs	FFs	Latches	CARRY4	SRL	LUTRAM	BRAM 18K	BRAM 36K	Clock buffers/MUXes
An Ethernet Controller [24]	—	450	704	0	16	0	0	0	0	0
Ariane-Ethernet [25]	—	943	782	0	34	2	16	0	12	1
Gaisler GRETH [26]	—	1137	866	0	54	0	96	0	0	0
LeWiz LMAC1 [27]	LMAC_CORE1_AXIS	242144	406797	383	720	1	0	0	0	0
LeWiz LMAC1 [27]	LMAC_CORE1_AXIS_XPM_FIFO	5721	6269	1	569	1	240	4	12	0
LeWiz LMAC1 [27]	LMAC_CORE_TOP	261693	421734	318	619	1	0	0	0	0
LeWiz LMAC1 [27]	LMAC_CORE_TOP_XPM_FIFO	4957	5297	1	501	1	96	5	10	0
LeWiz LMAC2 [28]	LMAC_CORE_TOP_XPM_FIFO	5954	6325	1	547	1	96	5	10	0
LeWiz LMAC3 [29]	LMAC_CORE_TOP_XPM_FIFO	47300	21068	230	838	259	0	9	51	0
Litex Liteeth [30]	liteeth	896	670	0	51	0	80	16	2	0
Litex Liteeth [30]	liteeth-rgmii	867	648	0	51	0	80	16	2	3
NFMAC10G [31]	nfmac10g	3010	1203	0	18	0	0	0	0	0
NFMAC10G [31]	nfmac10g_with_user_intf	3583	2257	0	36	0	0	0	2	0
Opencores Ethernet Tri Mode [32]	ethernet_tri_mode	1443	1399	0	61	0	0	0	6	3
Opencores Ethernet Tri Mode [32]	ethernet_tri_mode_single_clk	1443	1399	0	61	0	0	0	6	3
Opencores Ethmac [33]	—	2179	2343	0	76	0	0	4	0	0
Opencores Gbiteth [34]	—	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Opencores Minimac [35]	—	636	637	0	16	0	0	2	0	0
Opencores XGE_LL_MAC [36]	—	2494	887	0	47	0	0	0	0	0
Opencores XGE_MAC [37]	—	2286	1524	0	53	0	48	0	4	0
P. Kerling Ethernet MAC [38]	pkerling_ethernet_mac	435	256	0	6	0	0	0	0	2
P. Kerling Ethernet MAC [38]	pkerling_ethernet_mac_with_fifos	853	617	0	62	2	0	1	1	2
Verilog-Ethernet [39]	verilog-ethernet-eth_mac_10g_fifo	2953	1263	0	32	0	0	2	2	0
Verilog-Ethernet [39]	verilog-ethernet-eth_mac_1g_fifo	520	522	0	43	0	0	2	2	0
Verilog-Ethernet [39]	verilog-ethernet-eth_mac_1g_gmii_fifo	535	576	0	43	0	0	2	2	2
Verilog-Ethernet [39]	verilog-ethernet-eth_mac_1g_rgmii_fifo	559	565	0	43	0	0	2	2	1
Verilog-Ethernet [39]	verilog-ethernet-eth_mac_mii_fifo	498	538	0	43	0	0	2	2	2
WGE 100 [40]	—	615	838	0	66	0	28	2	0	2
WhiteRabbit [41]	wr-endpoint	1833	1630	0	45	1	0	4	0	0

Furthermore, some cores need manually ported components (e.g., IO components such as DDR inputs and outputs or FIFO buffers), which were added in a preliminarily form to allow synthesis. While these implementations were not verified in simulation or hardware, they should nevertheless provide information about the approximate relative resource usage of the cores.

3.2.2. Resource Results. The primary output of the OOC synthesis process is a technology-dependent netlist of the synthesized cores and thus information about the amount of FPGA resources needed to implement them. Basic resource usage data for the synthesized variants can be found in alphabetical order in Table 4, and usage of additional, more specialized resources can be found in Table 5. These results were obtained by running Vivado's OOC synthesis (synth_design -mode out_of_context) with default options and one pass of optimization (opt_design) afterwards. The latter command performs basic optimizations such as propagating constants and removing nets and cells with no fan-out [44].

Due to missing VHDL package files, containing, e.g., type definitions, the project *Opencores Gbiteth* was not synthesizable and thus could not be analyzed in this and the following steps.

On first glance, the dramatic resource usage of the LMAC1 variants that use inferred FIFOs—obviously in a way not correctly recognized by Vivado—can be spotted in Table 4. If RAM-based FIFOs are instantiated in these cases, more sensible resource results are produced.

Furthermore, LeWiz's LMAC3 core consumes a large number of LUTs even in the XPM-FIFO variant compared to the other analyzed MACs. Analyzing the hierarchical resource results reveals that the majority of the additional resource usage (more than 30000 LUTs and 7000 flip-flops) compared to LMAC2 stems from the implementation of receive and transmit CRC blocks. As LMAC3 uses a 256 bit wide data path and supports data rates of up to 100 Gbit/s, a high-performing but less resource-efficient CRC implementation may have been chosen. An increase in resource usage when increasing the transmission speed above 1 Gbit/s can also be seen when comparing the Verilog-Ethernet 1 Gbit/s variants to the 10 Gbit/s variant.

Apart from the outliers mentioned above, it can be seen that the analyzed cores span a wide range of sizes. While some implementations (e.g., the 100 Mbit/s and 1 Gbit/s variants of Verilog-Ethernet) use close to 500 LUTs and FFs, apparently more complex implementations such as the LMAC variants and to a limited extent also GRETH, Opencores Ethernet Tri Mode, and Opencores Ethmac consume 1000 s of each. When comparing the resource usage of the different cores, the vastly different amount of functionality provided by the individual cores must be taken into account. For example, the MACs provided by the Verilog-Ethernet project are relatively bare-bone—these cores allow to place frames to be sent into a FIFO and read received ones from another FIFO, with little additional functionality apart from checking FCS validity on incoming frames and inserting the FCS in outgoing ones. On the other hand, for MACs such as Gaisler's GRETH or Ethmac, the focus seems to be on CPU-based systems, with these cores providing a large number of configuration registers, interrupt circuitry, and DMA functionality. Furthermore, the width of the data path may play a role in the used LUTs and FFs.

Different sizes of provided FIFOs and buffers explain the difference in LUT-based RAM (LUTRAM) and Block RAM (BRAM) seen among the different cores. For example, both Litex variants and Ariane-Ethernet provide internal memory-mapped receive and transmit RAM buffers in contrast to Gaisler GRETH and Opencores Ethmac that use DMA to write to external memory.

Another important insight provided by the resource counts is the number of instantiated latches (in contrast to the number of instantiated flip-flops). Latches inferred by the synthesis tool instead of flip-flops are often the result of incorrect descriptions of combinational or sequential logic in an HDL (sometimes called "unintended latches"), as they may, among other problems, complicate correct static timing analysis [45]. If latches are instantiated, this prompts analysis into the responsible sections of the hardware description in order to verify if the latch was actually intended. In Table 4, the only cores that instantiate latches are LeWiz's LMAC cores. Part of the latches instantiated by Vivado may be caused by the incompatible description of the FIFO inferred from HDL. However, also the variants that use an instantiated XPM FIFO contain at least one latch. Analysis of the source code of LMAC1 and LMAC2 revealed that the latch in both cases is instantiated in the design unit tcore_rx_xgmii. Here, the signal pre_pkt_we_wire is assigned in a way that requires implementation as a latch, and it thus must be counted as an intended latch. Listing 1 shows an excerpt of the responsible Verilog code in LMAC2. LMAC3 applies similar patterns in its design units tcore_rx_cgmii and eth_crc32_gen. The larger number in the LMAC3 case stems from the CRC generator because the affected signal in this case is 32 bits wide, and the design unit is instantiated once in the receive and once in the transmit path. Furthermore, as these latches fan out to a large number of cells, the synthesis tool replicates each latch two or three times.

Most cores themselves do not require FPGA resources beyond LUTs, flip-flops, and block memory to implement their functionality. Only the subset of cores listed in Table 5 requires additional, more specialized resources. Among the required resources are the following IO- and clock-related FPGA resources:

(i) OBUF and IBUF elements that are ordinary buffers for input and output signals in Xilinx FPGAs that have been explicitly instantiated in the RTL description.

(ii) ODDR and IDDR elements that implement drivers or receivers of double-data-rate signals. These elements either receive a DDR data signal and clock, and generate two single data rate signals from it, or vice versa [46]. This is needed for DDR PHY interfaces such as RGMII.

Table 5: Additional OOC synthesis resource results.

Project	Variant	Additional FPGA resources
Ariane-Ethernet [25]	—	ODDR: 6, IDELAYE2: 5, IDDR: 5, IDELAYCTRL: 1, BUFIO: 1
Litex Liteeth [30]	liteeth-rgmii	ODDR: 6, IDELAYE2: 5, IDDR: 5, OBUF: 6, IBUF: 6, PLLE2_ADV: 1
P. Kerling Ethernet MAC [38]	pkerling_ethernet_mac	ODDR: 1, IDELAYE2: 10, IBUF: 1
P. Kerling Ethernet MAC [38]	pkerling_ethernet_mac_with_fifos	ODDR: 1, IDELAYE2: 10, IBUF: 1
Verilog-Ethernet [39]	verilog-ethernet-eth_mac_1g_gmii_fifo	ODDR: 1, BUFIO: 1
Verilog-Ethernet [39]	verilog-ethernet-eth_mac_1g_rgmii_fifo	ODDR: 6, IDDR: 5, BUFIO: 1
Verilog-Ethernet [39]	verilog-ethernet-eth_mac_mii_fifo	BUFIO: 1
WGE 100 [40]	—	ODDR: 1

```
always @ ( * ) begin
    pre_pkt_we_wire =
        !rst_ ? 1'b0:
        <...cond1...> ? 1'b0:
        <...cond2...> ? 1'b0:
        <...cond3...> ? 1'b1:
        pre_pkt_we_wire
        ;
end
```

Listing 1: Latch description in LMAC2.

(iii) IDELAYE2 elements that allow to delay signals (either coming from a pin or the FPGA fabric) by a configurable duration [46]. These elements are used in some cores to, for example, delay incoming data and control signals relative to their clock to improve signal stability when sampled. In order to function properly, either the core itself (as in the case of Ariane-Ethernet) or the instantiating logic must instantiate an IDELAYCTRL element that calibrates the IDELAYE2's delay taps to a reference clock.

(iv) BUFIO elements that implement clock buffers that can drive global clock nets from an input pin [47].

(v) PLLE2_ADV elements that instantiate one of the FPGA's PLLs for clock synthesis, skew compensation, and phase shifting [47].

3.2.3. Linting Results. Important side effects of the OOC synthesis experiments are the automated linting checks that are performed during the synthesis process. Generally speaking, Xilinx Vivado's synthesis process is rather sensitive when generating warnings, warning both about minor imperfections in the input HDL as well as about potential bugs. Two kinds of warnings are generated: *Warnings,* for situations that may lead to suboptimal results and where user action thus may be taken, and *Critical Warnings* for constructs Vivado deems "*outside the best practices for an FPGA family*" and thus recommends user action [44].

Table 6 shows a summary of the number of warnings generated during OOC synthesis of the analyzed cores. The generated warnings have been summarized into the following categories:

(i) Warnings concerning *constraints*, for example, clocks.

(ii) Warnings concerning the generation of (potentially unintended) *latches*.

(iii) *Linting* warnings that contain information about benign imperfections of the HDL code.

(iv) Warnings that describe which parts of the design are trimmed or *optimized* away.

(v) *Simulation mismatch* warnings that refer to constructs that may lead to different behavior in hardware and in logic simulation.

(vi) Warnings concerning the *structure* of the design.

(vii) As their own category due to their observed number, warnings that inform about *internally unconnected ports*. Vivado reacts rather sensitive to this condition, generating warnings of this kind even if not all bits of a vector are used in a module that is driven by this vector.

Vivado limits reporting of each individual warning message to 100 occurrences. In these cases, a warning count of "100+" is shown in Table 6. The actual warning IDs generated by Vivado that have been subsumed into the categories described above are listed in Table 7.

Of these categories, we consider warnings that fall into the constraints, latches, simulation mismatch, and structural categories to be especially serious.

Most of the analyzed cores and their variants generate relatively few serious warnings. The variants of LMAC1 that have been synthesized using the original FIFO inferred from HDL are the candidates that produce the most of these warnings. Once the problematic FIFO descriptions are replaced with vendor-defined instantiated ones, most of these warnings disappear. However, in all LMAC variants, as also seen in Section 3.2.2 some latches are generated. In addition, Vivado warns about a latch being generated in the WhiteRabbit core that appears to be later optimized away as the final resource count in Table 4 shows no latches instantiated for this core. Listing 2 shows the VHDL code that causes Vivado to infer a latch. The conditional assignment statement is missing an else case, requiring consistency_match to hold its value when the condition is not met. As the responsible VHDL description never lets the signal return to zero and the signal is not initialized at declaration, this may constitute a bug resulting in an unintended latch description.

Few cores, namely, Gaisler GRETH, Opencores Ethernet Tri Mode, Opencores Ethmac, and Opencores XGE LL MAC, cause warnings in the *simulation mismatch* category. As seen in Table 7, warnings in this category concern (1) the sensitivity lists of processes and (2) the description of the reset behavior of flip-flops. In the case of Gaisler GRETH, one signal that is read in a combinational process is not part of the process sensitivity list. This may cause a logic simulator not to reevaluate the process when only this signal of the process's inputs changes. The same issue is present for one signal in the Opencores Ethernet Tri Mode project, one

Table 6: Vivado synthesis warnings.

Project	Variant	Constraints	Latches	Linting	Optimization	Simulation mismatch	Structural	Unconnected (internal) port
An Ethernet Controller	—	—	—	6	12	—	—	100+
Ariane-Ethernet	—	—	—	2	4	—	—	21
Gaisler GRETH	—	—	—	100+	73	1	18	100+
LeWiz LMAC1	LMAC_CORE1_AXIS	2	41	1	100+	100+	—	100+
LeWiz LMAC1	LMAC_CORE1_AXIS_XPM_FIFO	—	1	1	88	—	—	100+
LeWiz LMAC1	LMAC_CORE_TOP	5	33	2	94	96	—	100+
LeWiz LMAC1	LMAC_CORE_TOP_XPM_FIFO	—	1	—	80	—	—	100+
LeWiz LMAC2	LMAC_CORE2_TOP_XPM_FIFO	—	1	—	82	—	—	100+
LeWiz LMAC3	LMAC_CORE3_TOP_XPM_FIFO	—	2	2	100+	—	—	100+
Litex Liteeth	liteeth	—	—	—	56	—	—	16
Litex Liteeth	liteeth-rgmii	—	—	6	66	—	—	12
NFMAC10G	nfmac10g	—	—	61	2	—	2	100+
NFMAC10G	nfmac10g_with_user_intf	—	—	61	2	—	2	100+
Opencores Ethernet Tri Mode	clk_reg_is_clk_user	—	—	35	6	2	12	100+
Opencores Ethernet Tri Mode	clk_reg_is_not_clk_user	—	—	35	6	2	12	100+
Opencores Ethmac	—	—	—	1	1	1	1	17
Opencores Gbiteth	—	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Opencores Minimac	—	—	—	12	2	—	3	80
Opencores XGE_LL_MAC	xge_ll_mac	—	—	12	3	1	—	1
Opencores XGE_MAC	xge_mac	—	—	—	1	—	—	67
Opencores XGE_MAC	xge_mac_ramstyle	—	—	—	1	—	—	67
P. Kerling Ethernet MAC	pkerling_ethernet_mac	—	—	1	3	—	2	6
P. Kerling Ethernet MAC	pkerling_ethernet_mac_with_fifos	—	—	—	9	—	2	6
Verilog-Ethernet	verilog-ethernet-eth_mac_10g_fifo	—	—	100+	13	—	4	100+
Verilog-Ethernet	verilog-ethernet-eth_mac_1g_fifo	—	—	26	10	—	—	100+
Verilog-Ethernet	verilog-ethernet-eth_mac_1g_gmii_fifo	—	—	26	10	—	—	100+
Verilog-Ethernet	verilog-ethernet-eth_mac_1g_rgmii_fifo	—	—	27	11	—	—	100+
Verilog-Ethernet	verilog-ethernet-eth_mac_mii_fifo	—	—	28	10	—	—	100+
WGE 100	—	—	—	22	—	—	1	20
WhiteRabbit	wr-endpoint	—	1	—	22	—	27	100+

Table 7: Vivado warning categories.

Category	Warning ID	Warning text
Constraints	Timing 38–493	Port name has one or several leaf clock pins in its transitive fan-out without any clock buffer on the path and no clock reaching the clock pin(s). Vivado cannot infer the clock source when no clock buffer is found on the path to a leaf clock pin.
Latches	Synth 8–327	Inferring latch for variable name
Linting	Synth 8–151	Case item value is unreachable
Linting	Synth 8–2490	Overwriting previous definition of module name
Linting	Synth 8–2507	Parameter declaration becomes local in name with formal parameter declaration list
Linting	Synth 8–2644	Root scope declaration is not allowed in Verilog 95/2K mode
Linting	Synth 8–3917	Design name has port name driven by constant value
Linting	Synth 8–4747	Shared variables must be of a protected type
Linting	Synth 8–639	System function call name not supported
Linting	Synth 8–689	Width (N) of port connection name does not match port width (M) of module name
Linting	Synth 8–7023	Instance name of module name has N connections declared, but only M given
Optimization	Synth 8–3332	Sequential element name is unused and will be removed from module name
Optimization	Synth 8–3936	Found unconnected internal register name and it is trimmed from N to M bits
Optimization	Synth 8–4446	All outputs are unconnected for this instance and logic may be removed
Optimization	Synth 8–6014	Unused sequential element name was removed
Simulation mismatch	Synth 8–567	Referenced signal name should be on the sensitivity list
Simulation mismatch	Synth 8–5788	Register name in module name has both set and reset with same priority. This may cause simulation mismatches. Consider rewriting code
Simulation mismatch	Synth 8–614	Signal name is read in the process but is not in the sensitivity list
Simulation mismatch	Synth 8–6426	Mix of sync and async assignments to register name in module name in the same process may cause logic issues
Structural	Synth 8–3848	Net name in module/entity name does not have driver
Structural	Synth 8–6104	Input port name has an internal driver
Unconnected (internal) port	Synth 8–3331	Design name has unconnected port name

```
consistency_match <= '1' when
    (<...cond1...> and
    <...cond2...> and
    <...cond3...>);
```

Listing 2: Latch description in WhiteRabbit wr-endpoint.

signal in the Opencores Ethmac project, and one signal in the Opencores XGE LL MAC project.

Furthermore, in the Opencores Ethernet Tri Mode project, Vivado complains about the register description shown in Listing 3 with a Synth 8–5788 warning (see Table 7). If the input RegInit is tied to a constant in an instantiation, Vivado infers flip-flops with asynchronous reset for the 0-bits and flip-flops with asynchronous set for the 1-bits. In the RTL description, all RegInit inputs are tied to constants at instantiations of RegCPUData. If they, however, were not tied to constants—which is apparently assumed by Vivado during OOC synthesis at first—the value that is asynchronously loaded into each of the flip-flops making up RegOut would depend on a non-constant value. As this is not supported by the flip-flops provided by Xilinx's 7 Series FPGAs, this would need to be implemented using additional logic, which Synth 8–5788 warns about.

Finally, some cores leave (parts of) signals unassigned, i.e., with no driver, resulting in Synth 8–3848 warnings, which are classified as *Structural*. This is (relatively) benign when the respective signals are not used but can evolve into a bug if they drive logic in the future. If they are used, the synthesis tool assumes a value for the concerned signal (e.g., assumes constant zero), which may or may not behave as intended.

In addition to Ethernet MAC functionality, Gaisler GRETH optionally implements a UDP-to-AHB bridge referred to as Ethernet Debug Communication Link (EDCL). The top level of GRETH provides a secondary AHB master interface for EDCL that operates in parallel to the primary AHB master, the bus interface of GRETH's DMA engine. If EDCL is not used, the outputs of the signals of this secondary AHB master are left unconnected. No immediate effects on the design due to these unconnected signals are to be expected as long as the secondary AHB interface is also left unconnected at instantiation externally.

In the case of nfmac10g, the two concerned undriven signals are outputs of a module that are not used at instantiation. Thus, no immediate effects on the functionality of the core are to be expected in this case. In the case of Opencores Ethmac, this concerns a single "debug" signal that can be read from the Wishbone-mapped "Debug" register. Opencores Triple Speed Ethernet, however, does not drive several nets that feed into the MIIM (i.e., MDIO)

```
module RegCPUData (..., RegInit, RegOut, ...);
...
input [15:0] RegInit;
output [15:0] RegOut;
...
always @ (posedge Reset or posedge Clk)
  if (Reset)
    RegOut <= RegInit;
  else if (<conditions>)
    RegOut <= CD_in_reg;
endmodule
```

Listing 3: Register Description in Opencores Ethernet Tri Mode.

block from the configuration register module. This has the effect that no transactions on the MDIO interface can be issued from the control interface, and thus most of the MDIO logic is optimized away.

As mentioned before, in addition to the "normal" warnings discussed above, Vivado also generates *Critical Warnings* for constructs deemed especially dangerous. Only three kinds of these critical warnings were observed in the cores that were subjected to the OOC synthesis process, as shown in Table 8:

(i) Warnings about multi-driven nets (Vivado Warnings Synth 8–6858 and Synth 8–6859). This is only the case in LeWiz's LMAC1 core. Here, in two design units, two Verilog coding errors are responsible: (1) a reg that is driven from an always block also has a continuous assign statement driving it and (2) a reg that is reset in one always block is driven in another.

(ii) Warnings about incorrect BRAM instantiation (Vivado Warning Netlist 29–368), which are generated if not all input ports necessary for the instantiated RAM width are connected. Vivado warns that this might cause incorrect BRAM behavior. This only concerns the Opencores Ethernet Tri Mode MAC core.

(iii) Cores that use tri-state cells generate warnings of type Synth 8–5799 with the message "Converted tricell instance <name> to logic." This happens because in OOC synthesis, (1) tri-state ports are not supported and (2) internal tri-state nets are not supported in general. In the case of both Litex Liteeth variants, both variants of P. Kerling's Ethernet MAC, and WGE 100, these critical warnings can be traced to MDIO implementations, where the data signal is bidirectional and implemented as an inout port in these cases. However, in Opencores Ethmac, a description of an inferred memory block (eth_spram_256×32) triggers the warning as it contains the assignment shown in Listing 4.

3.2.4. Clock Domain Crossings. A further important indicator of the quality of a complex IP core is the handling of CDCs. In most analyzed cores, CDCs are present. For example, any core that allows the chip-side interface to run with a clock that is not related to the MII clock needs CDCs.

CDCs are considered a critical part of any digital design because improper handling may lead to timing (setup/hold) violations at runtime, leading to unwanted behavior due to metastability. Thus, there are some "best-practice" accepted design patterns for dealing with CDCs of different types, such as

(i) Synchronizer chains of multiple flip-flops for single-bit signals

(ii) Synchronizer chains for control logic that controls consistent sampling of multi-bit signals

(iii) Encoding multi-bit signals using Gray code

(iv) Use of dual-clocked technology elements such as dual-clock FIFOs and dual-port BRAMs

CDC analysis is a state-of-the art verification technique provided by tools such as Mentor Graphics Questa CDC (https://eda.sw.siemens.com/en-US/ic/questa/design-solutions/clock-domain-crossing/, accessed: May 5, 2023) and Synopsys Spyglass CDC (https://www.synopsys.com/verification/static-and-formal-verification/spyglass/spyglass-cdc.html, accessed: May 5, 2023). Xilinx Vivado also provides some support for structural CDC analysis in the form of the report_cdc command. Vivado's CDC analysis identifies paths crossing from one clock domain to another. It then tries to identify "best-practice" CDC structures according to vendor-defined guidelines [48]. If such "safe" structures cannot be determined or unsafe structures are detected, Vivado generates CDC warnings. As these warnings might be overly sensitive, they have to be reviewed by a designer. The result of this review can then either be to introduce a fix of the identified error or to document why the particular CDC is safe in a way that is not understood by the analysis tool.

There are some cores where a CDC analysis is not applicable as they do not include clock domain crossings:

Table 8: Vivado critical synthesis warnings.

Project	Variant	BRAM instantiation	Multi-driven nets	Tri-cell conversion
LeWiz LMAC1	LMAC_CORE1_AXIS	—	12	—
LeWiz LMAC1	LMAC_CORE1_AXIS_XPM_FIFO	—	12	—
LeWiz LMAC1	LMAC_CORE_TOP	—	12	—
LeWiz LMAC1	LMAC_CORE_TOP_XPM_FIFO	—	12	—
Litex Liteeth	liteeth	—	—	1
Litex Liteeth	liteeth-rgmii	—	—	1
Opencores Ethernet Tri Mode	clk_reg_is_clk_user	12	—	—
Opencores Ethernet Tri Mode	clk_reg_is_not_clk_user	12	—	—
Opencores Ethmac	—	—	—	32
Opencores Minimac	—	—	—	1
P. Kerling Ethernet MAC	pkerling_ethernet_mac	—	—	1
P. Kerling Ethernet MAC	pkerling_ethernet_mac_with_fifos	—	—	1
WGE 100	—	—	—	1

```
//Data output drivers
assign dato = (oe & ce) ? q: {32{1'bz}};
```

Listing 4: Internal Tri-State in Opencores Ethmac.

(i) The entire logic of An Ethernet Controller is clocked at the interface clock. The 25 MHz MII RX and TX clocks are sampled and used as control signals for the respective RX and TX logic. This logic thus relies on being clocked significantly faster than the MII clock. The RX and TX clocks are synchronized to the interface clock with a 2-stage synchronizer. In the synchronized clock, rising edges are detected, and this rising-edge signal (synchronous to the interface clock) is then used as a control signal to load the MII RX signals into flip-flops.

(ii) In its basic version (nfmac10g variant), the NFMAC10G project does not provide clock domain crossings. The RX and TX parts of the MAC are expected to run with the XGMII RX and TX clocks, respectively. However, the nfmac10g_with_user_-intf variant provides FIFOs for decoupling the user logic and the XGMII interfaces.

(iii) In the same way, Opencores XGE LL MAC requires that the core is clocked with the XGMII clock. It does not support different RX and TX clocks.

(iv) While the various LeWiz LMAC cores in their basic variant (LMAC_COREx_TOP) provide multiple clock inputs that indicate that support for different MII and interface clocks *should* be possible, it is evident from the provided FPGA example projects that the authors' intention is to input the same clock signal into each of these clock ports. Decoupling between user logic and the MII clock happens in the AXI-Stream Interface module that was evaluated for the LMAC1 variant (LMAC_CORE1_AXIS and LMAC_CORE1_AXIS_XPM_FIFO).

The number of clocks, as well as asynchronous clock pairs with actual paths between the source and destination clock, and the results of CDC analysis performed using Vivado after OOC synthesis can be found in Table 9. The table shows the total count of warnings in the CDC report, the number of *Critical* CDC warnings, and the warnings triggered by each core (CDC-1, CDC-2, etc.). *Critical* warnings have been marked with an asterisk. These are the warnings that Vivado classified as especially critical, requiring user intervention. A description of these warning IDs is shown in Table 10.

Of the warnings shown in Tables 9 and 10, at least CDC-15 can be considered informational only. This warning is generated by Vivado when a clock-enable controlled CDC structure is detected. In this CDC structure, a control signal (e.g., a "valid" signal) is synchronized to the destination clock using a synchronizer chain and often converted to a pulse. This signal is then used as clock-enable signal of flip-flops that sample a multi-bit signal into the destination clock domain without any other synchronizing logic. The correct operation of this structure depends on surrounding logic to ensure that the multi-bit signal does not change during sampling (e.g., by employing a handshake pattern). Thus, the CDC-15 warning suggests the considered CDC for review. It does, however, not indicate the detection of a potentially dangerous design pattern. In the same way, CDC-2 could be considered as (relatively) benign. Xilinx suggests to inform the implementation tool of a register used in a synchronizer by setting the ASYNC_REG property of the corresponding RTL signal. This prevents, e.g., absorbing these flip-flops into non-CDC capable FPGA resources such as LUT-based shift registers (SRL16 and SRL32). A missing ASYNC_REG property indicates that this might happen in future implementation runs, even if it did not happen in the current run and the CDC was correctly detected.

Considered more serious—although non-critical—warnings are CDC-5 and CDC-6. These

Table 9: CDC warning results.

Project	Variant	Clocks	Clock pairs	Total	Critical	CDC-1*	CDC-2	CDC-4*	CDC-5	CDC-6	CDC-7*	CDC-10*	CDC-11*	CDC-12*	CDC-13*	CDC-14*	CDC-15	CDC-26
An Ethernet Controller	—	1	N/A	N/A	N/A	—	—	—	—	—	—	—	—	—	—	—	—	—
Ariane-Ethernet	—	2	2	7	4	—	1	—	2	—	1	2	—	—	1	—	—	—
Gaisler GRETH	—	3	4	148	81	44	1	1	—	—	2	2	—	—	32	—	66	—
LeWiz LMAC1	LMAC_CORE1_AXIS	3	4	33140	4	—	—	—	—	—	—	—	—	—	—	4	33136	—
LeWiz LMAC1	LMAC_CORE1_AXIS_XPM_FIFO	3	2	48	0	—	—	—	—	—	—	—	—	—	—	—	48	—
LeWiz LMAC1	LMAC_CORE_TOP	1	N/A	N/A	N/A	—	—	—	—	—	—	—	—	—	—	—	—	—
LeWiz LMAC1	LMAC_CORE_TOP_XPM_FIFO	1	N/A	N/A	N/A	—	—	—	—	—	—	—	—	—	—	—	—	—
LeWiz LMAC2	LMAC_CORE2_TOP_XPM_FIFO	1	N/A	N/A	N/A	—	—	—	—	—	—	—	—	—	—	—	—	—
LeWiz LMAC3	LMAC_CORE3_TOP_XPM_FIFO	1	N/A	N/A	N/A	—	—	—	—	—	—	—	—	—	—	—	—	—
Litex Liteeth	liteeth	3	4	6	2	—	—	—	—	4	—	2	—	—	—	—	—	—
Litex Liteeth	liteeth-rgmii	3	4	4	0	—	—	—	—	4	—	—	—	—	—	—	—	—
NFMAC10G	nfmac10g	1	N/A	N/A	N/A	—	—	—	—	—	—	—	—	—	—	—	—	—
NFMAC10G	nfmac10g_with_user_intf	4	4	8	8	4	—	4	—	—	—	—	—	—	—	—	—	—
Opencores Ethernet Tri Mode	clk_reg_is_clk_user	12	23	649	445	363	2	28	—	—	—	—	—	4	46	4	202	—
Opencores Ethernet Tri Mode	clk_reg_is_not_clk_user	13	32	659	455	369	2	32	—	—	—	—	—	4	46	4	202	—
Opencores Ethmac	—	3	6	1132	373	365	20	2	—	—	—	6	—	—	—	—	739	—
Opencores Minimac	—	3	4	5	0	—	5	—	—	—	—	—	—	—	—	—	—	—
Opencores XGE LL MAC	—	1	N/A	N/A	N/A	—	—	—	—	—	—	—	—	—	—	—	—	—
Opencores XGE_MAC	xge_mac	4	10	48	0	—	12	—	8	—	—	—	—	—	—	—	—	28
Opencores XGE_MAC	xge_mac_ramstyle	4	10	20	0	—	12	—	8	—	—	—	—	—	—	—	—	—
P. Kerling Ethernet MAC	pkerling_ethernet_mac	6	5	186	126	122	—	—	—	—	—	—	—	—	4	—	60	—
P. Kerling Ethernet MAC	pkerling_ethernet_mac_with_fifos	7	6	264	204	200	—	—	—	—	—	—	—	—	4	—	60	—
Verilog-Ethernet	verilog-ethernet-eth_mac_10g_fifo	4	7	31	8	—	—	—	—	3	4	4	—	—	—	—	20	—
Verilog-Ethernet	verilog-ethernet-eth_mac_1g_fifo	4	7	37	8	—	—	—	—	3	4	4	—	—	—	—	26	—
Verilog-Ethernet	verilog-ethernet-eth_mac_1g_gmii_fifo	7	12	62	15	—	—	—	1	4	8	4	—	2	1	—	39	3
Verilog-Ethernet	verilog-ethernet-eth_mac_1g_rgmii_fifo	4	7	42	10	—	—	—	1	3	6	4	—	—	—	—	26	2
Verilog-Ethernet	verilog-ethernet-eth_mac_mii_fifo	4	7	41	10	—	—	—	—	3	6	4	—	—	—	—	26	2
WGE 100	—	9	13	313	170	157	68	2	2	—	—	2	—	—	9	—	73	—
WhiteRabbit	wr-endpoint	4	7	188	93	78	4	—	—	4	5	6	2	—	2	—	87	—

Table 10: CDC warning description.

ID	Severity	Description
CDC-1	Critical	A single-bit CDC path is not synchronized or has unknown CDC circuitry
CDC-2	Warning	A single-bit CDC path is synchronized with a 2+ stage synchronizer but the ASYNC_REG property is missing on all or some of the synchronizer flip-flops
CDC-4	Critical	A multi-bit bus CDC path is not synchronized or has unknown CDC circuitry
CDC-5	Warning	A multi-bit bus CDC path is synchronized with a 2+ stage synchronizer but the ASYNC_REG property is missing on all or some of the synchronizer flip-flops
CDC-6	Warning	A multi-bit bus CDC path is synchronized with a 2+ stage synchronizer and the ASYNC_REG property is present
CDC-7	Critical	An asynchronous signal (clear or preset) is not synchronized or has unknown CDC circuitry
CDC-10	Critical	Combinatorial logic has been detected in the fan-in of a synchronization circuit
CDC-11	Critical	A fan-out has been detected before a synchronization circuit
CDC-12	Critical	Data from multiple clocks are found in the fan-in of a synchronization circuit
CDC-13	Critical	1-bit CDC detected on a non-FD primitive
CDC-14	Critical	Multi-bit CDC detected on a non-FD primitive
CDC-15	Warning	Clock-enable controlled CDC
CDC-26	Warning	RAM-to-FD CDC: LUTRAM read/write potential collision

warnings are generated if a bus is synchronized using a synchronizer for each bit. If the bits of the bus are not required to be consistent in each clock cycle after synchronization, this may be acceptable. The warnings indicate that the respective RTL should be reviewed to ensure that this is the case. The difference between CDC-5 and CDC-6 is that CDC-5 additionally warns about missing ASYNC_REG properties.

Finally, CDC-26 is a rather technology-specific warning. It indicates that the write port of a LUTRAM and the flip-flop latching its output are clocked with two different clocks, effectively forming a CDC. As reading of the LUTRAM in 7 Series FPGAs happens asynchronously [49], there exists a path that crosses directly from the write to the read clock domain in the case that the read and write addresses are set to the same value. If the user logic ensures that this never happens, a LUTRAM can be used safely for a CDC [48]. Warning CDC-26 thus informs the user to review that this is always the case.

If the analysis tool cannot match the logic on a path between different clock domains to a known "safe" CDC pattern or if it recognizes an "unsafe" pattern, *Critical* CDC warnings are generated. These are marked in Table 9 with an asterisk following the warning ID.

If the logic on a CDC path cannot be matched to a single or multiple-bit synchronizer and also does not match to an enable-, multiplexer-, or BRAM-based design pattern, the *Critical* warnings CDC-1 (for single-bit signals) and CDC-4 (for multi-bit signals) are generated. If the asynchronous reset or set signal of a flip-flop is concerned instead of its data (D) or clock enable (CE) input, the CDC-7*Critical* warning is generated.

Critical warnings CDC-10, CDC-11, and CDC-12 inform about synchronizers that have been recognized but violate some "best practices" such that the input of a synchronizer should directly originate from a flip-flop in the source clock domain, not from combinational logic as this may introduce glitches into the synchronizer, reducing the Mean Time Between Failures (MTBF).

Finally, CDC-13 and CDC-14 are technology-specific critical warnings, informing the user that a CDC is present between a flip-flop of the source clock domain and another technology element that is clocked by the destination clock. This may, for example, happen if a synchronizer chain (with no reset of the flip-flops) is not constrained with the ASYNC_REG attribute. In this case, the chain of flip-flops may be interpreted as a shift register by the implementation tool and mapped to a LUT-based shift register (SRL16 or SRL32 [49]), especially if more than two are used. This is also the case if a flip-flop in the source clock domain directly feeds into a port of a memory block clocked by the destination clock.

The warnings generated by Vivado's CDC analysis indicate potential problems in the circuit; however, not all warnings correspond to actually critical circuitry. Thus, besides the actual count of warnings in the respective category, in-depth analysis of the reported paths is necessary to judge the quality of the implemented CDCs. In the following, for each analyzed core or core variant, a brief description of the sources of the CDC warnings is provided.

(i) **Ariane Ethernet** causes relatively few CDC warnings to be generated during OOC synthesis. The CDC-7 and CDC-10 warnings concern 3-stage reset synchronizers, which contain combinational logic to OR multiple reset sources either before the first stage or between the first and the second stage. The CDC-13 warning refers to an intended synchronizer that is however not constrained as such. This register chain is mapped by Vivado to a LUT-based shift register, which is not recommended by Xilinx for implementing synchronizers. Finally, the tool warns about multi-bit signals that are synchronized from one clock domain into another, which concerns Gray-coded pointers in the RX and TX FIFO here, which is a commonly used design pattern.

(ii) **Gaisler's GRETH** employs clock-enable controlled structures for its many clock domain crossings. Here, a "valid" signal from the RX or TX clock domain is synchronized into the core clock domain. This signal gates the transfer of other signals from the source clock domain into the destination clock domain (or vice versa). However, these paths are often not fully constrained for Xilinx tools (via the ASYNC_REG property), even when using the constraint files for Xilinx Artix 7 supplied by the authors of the core. This leads to the synthesis tool often not implementing the intended pattern by combining the synchronized control signal with other signals from the destination clock domain to form the data input signal to the destination flip-flops. This explains the CDC-1 and CDC-4 warnings. Furthermore, two reset synchronizers receive the output of combinational logic ORing the external reset with an internal one, leading to two CDC-10 warnings. Finally, the CDC-13 warning refers to a clock domain crossing within an inferred FIFO, where the "write enable" port of a dual-ported LUT-based memory is driven from one clock domain, the data however from another clock domain.

(iii) From the implementation examples provided by **LeWiz LMAC1**, it can be seen that all clock inputs with the exception of the AXI-Stream read and write clocks are intended to be driven from the same clock source. Thus, only the AXI-variants have paths that cross between clock domains. For these variants, Vivado reports very few (original variant with inferred FIFO) to no CDC warnings at all (modified variant with XPM instantiated FIFO). The CDC-4 warnings in the original variant concern the use of latches in the FIFOs, which are considered unsafe by Vivado for implementing clock domain crossings. The remaining LeWiz's LMAC cores (LMAC 2 and LMAC 3) were only synthesized in the non-AXI variant, and thus no CDCs are reported here.

(iv) Both **Litex** variants (MII and RGMII) generate very few CDC warnings. The critical CDC-10 warnings concern reset synchronizers that are driven in the MII variant by combinational functions that generate reset signals for the RX and TX clock domains. The non-critical CDC-6 warnings inform that some multi-bit signals are synchronized. As this concerns Gray-coded pointers in the FIFOs, these are considered to be benign.

(v) The **NFMAC10G** project provides two variants: a bare-bone variant without any clock domain crossings into a user clock domain (the interface runs with the PHY's receive and transmit clocks, respectively) and a variant with a more comfortable interface. Only the latter variant is considered for CDC analysis. For this variant, Vivado reported four paths where a single signal (a control signal) passes between two clock domains. This is intended to be the signal that controls transfer of wider buses in a "CE-controlled" fashion. However, the synchronizer is coded on RTL in a way that Vivado does not infer a clean chain of flip-flops. Rather, it instantiates flip-flops with combinational logic in between that are capable of setting the individual stages to a particular reset value. This leads to the intended synchronizer not acting as such and prompts the reported CDC-1 warnings. Furthermore, as the control signal is now not synchronized properly, the "CE-controlled" pattern is also not recognized as such, and the respective paths are reported as critical CDC-4 warnings. Overall, there are nevertheless very few paths criticized by Vivado's CDC analysis.

(vi) There are several clocking issues with the **Opencores Ethernet Tri Mode** project. Firstly, the design is constructed in a way that both the MII clock and clock with half its frequency are needed. The provided design unit for dividing this clock does this using a flip-flop. It is noted in a comment that this unit is intended "for simulation only" and needs to be "replaced according to technology." However, the supplied example implementation for Virtex 5 also uses this exact design unit. Due to the clock skew incurred by this clock divider, the two clocks are considered as asynchronous to each other. As these clocks are mutually exclusive, however, this leads to no additional clock domain crossing warnings.

Another potential problem is the use of Xilinx's BUFGMUX primitive as a clock multiplexer. On 7 Series FPGA, this uses the BUFGCTRL resource with the clock enable pins (CE0 and CE1) used to select between the two source clocks. However, using the CE pins on BUFGCTRL may cause glitches on the output clock if the signal driving the CE inputs violates the setup/hold time of either source clock [47]. This leads to clock domain crossings between the clock driving the select inputs and both input clocks to be selected using the clock multiplexer. These are reported as critical CDC-13 warnings. As an alternative, the BUFGMUX_CTRL macro switches the clocks glitch-free using the $S0$ and $S1$ inputs [47]. However, this alternative can only be used if both input clocks are free-running, which may not be applicable in this situation (when switching over from MII to GMII, the MII TX clock may already be disabled).

As there are multiple clock multiplexers in the design—that have been constrained for this work properly—and a destination flip-flop of a CDC is clocked by the output of this clock multiplexer, Vivado naturally detects two clock domain crossings for any given path to such flip-flops, if both clocks are considered asynchronous to the source clock. Thus, the high count of CDC warnings with this project can partly be explained by duplicate warnings due to the use of clock multiplexers.

However, the main problem that leads to massive numbers of critical warnings in the CDC report is that registers from the register bank are controlled via Wishbone and thus clocked by the Wishbone clock. They, however, fan out to the RX and TX clock domains without synchronization. It may be the case that the authors assumed that the configuration registers are not changed very often, and thus the probability of a metastability event during the entire time of operation was considered low. However, these paths are detected by Vivado and reported as critical CDC-1 and CDC-4 warnings.

Additionally, there are some paths between the different clock domains that are insufficiently synchronized: A single flip-flop is used to transfer the signal from the source clock domain into the destination clock domain, with an additional flip-flop often used for edge detection. This also leads to a substantial amount of CDC-1 warnings. This is also stated in an open issue in the issue tracker on the project's page on Opencores (https://opencores.org/projects/ethernet_tri_mode/issues/12, open since 2008, accessed: May 5, 2023).

Finally, the design provides two different interface clock inputs: the user clock and the register clock. The user clock is used to clock the data interface to the RX and TX MAC components, while the register clock drives the (control/status) register bank and the MDIO module. The documentation treats these as two unrelated clocks, which is also done in the variant *clk_reg_is_not_clk_user*. However, there are no synchronizers on the CDCs between these two clock domains, and the provided implementation example connects the same clock signal to both inputs. This leads to a slightly reduced number of CDC warnings as shown in the results for variant *clk_reg_is_clk_user*.

(vii) The **Opencores Ethmac** project partially suffers from the same problem as Opencores Ethernet Tri Mode: The register bank that contains configuration and status registers is clocked by the Wishbone clock, while its outputs are often used in combinational signals that feed into various flip-flops in the RX and TX clock domains. These lead to the majority of critical CDC-1 warnings issued by Vivado.

Furthermore, for handling PAUSE frames, but also for the purpose of enabling internal loopback, there are some paths that cross directly from the RX clock domain to the TX clock domain. These paths lack synchronization to the target domain, also contributing to the critical CDC-1 warnings. This would only be acceptable if it is always guaranteed that the RX and TX clock signals are in phase. However, the MII standard specifies that the TX clock is a free-running clock generated by the PHY, while the RX clock is recovered from incoming data.

Furthermore, the core also includes flip-flop chains as synchronizers on some paths. Constraints for Xilinx tools (ASYNC_REG) are missing for all of these synchronizers, leading to some CDC-2 warnings. Additionally, some of these synchronizers are not correctly implemented, using the output of the first flip-flop to drive combinational logic. This causes Vivado to not accept them as valid synchronizers, contributing additional CDC-1 warnings. With some synchronizers, the only detected problem is however that they are fed from combinational logic in the source clock domain, leading to a few CDC-10 warnings.

(viii) On the other hand, the **Opencores Minimac** project contains very few, and correctly implemented, synchronizers. These synchronizers transport reset signals and a TX control signal from the interface clock domain to the RX and TX clock domains and FIFO status signals (full and empty) from the RX and TX clock domains back to the interface clock domain. However, no constraints for Xilinx tools are provided, leading to warnings about these missing constraints (CDC-2).

(ix) The **Opencores XGE_MAC** contains few clock domain crossings, which generally follow accepted design patterns (clock-enable controlled CDCs where a control signal is synchronized and subsequently used to transfer data into the destination domain and Gray-coded pointers in the FIFOs). However, the project does not provide constraints for Xilinx Vivado, leading to some non-critical CDC-2 (single bits) and CDC-5 (Gray-coded pointers) warnings.

Furthermore, as there is no constraint used to direct the synthesis tool on how to implement the memory buffers for the inferred FIFOs, Vivado implements them as LUT-based memory due to their size. This leads to some CDC-26 warnings that warn about using LUTRAM in a CDC. If these memories are constrained to be implemented in block memory resources (as done with variant xge_mac_ramstyle) by setting the constraint SYN_RAMSTYLE = block on the memory, these warnings disappear.

(x) P. Kerling's Ethernet MAC is a triple-speed implementation, supporting MII and GMII. Thus, it also requires the use of clock multiplexers to select between the different clocking patterns employed by these two interfaces. The use of clock multiplexers once again has the potential to double the number of CDCs reported by Xilinx Vivado because one CDC is reported for each destination clock that may reach a flip-flop through the clock multiplexer.

The majority of CDC-1 warnings are caused by clock domain crossings between the clock for the MDIO interface and the other clocks in the design (125 MHz GMII clock, as well as RX and TX clocks). This is the case because the MDIO interface generates a speed [2 : 0] signal that selects between 10 Mbit/s, 100 Mbit/s, and 1 Gbit/s operation. This signal is not synchronized to any of the clock domains mentioned above. It may be the case that this signal is not expected to change value very often; however, this should then be documented.

Furthermore, also in this case, the asynchronously switching BUFGMUX instead of the synchronous BUFGMUX_CTRL is used, leading to some CDC-13 warnings. This is justified in a code comment for the instantiation of BUFGMUX with the need to perform switchover while one of the clocks is inactive.

(xi) The various evaluated variants of the **Verilog-Ethernet** project, with various PHY interfaces, generally lead to relatively few CDC warnings. Furthermore, constraint files for Xilinx Vivado and Intel Quartus are provided. In the Xilinx case, these constrain, for example, the synchronizers using ASYNC_REG properties.

One general pattern that can be found in all of the variants is that some reset synchronizers either are driven by combinational logic that ORs multiple reset sources or use combinational logic between the first and second stage of the usually three-stage synchronizers, explaining the CDC-7 and CDC-10 warnings.

The multi-bit CDCs reported (as CDC-5 and CDC-6 warnings) for these variants are generally either Gray-coded pointers in asynchronous FIFOs or two status signals (error_bad_fcs and error_-bad_frame). In the latter case, these are directly connected to top-level outputs, shifting the responsibility to deal with potential inconsistencies between the two signals during one interface clock cycle to the user.

The CDC-12 warning in the GMII variant of Verilog-Ethernet is caused by a reset synchronizer that receives ORed reset signals from the TX and user clock domains, while the CDC-13 warning is once again the result of using the asynchronously switching BUFGMUX clock multiplexer.

Finally, the CDC-26 warnings in the MII, GMII, and RGMII variants are generated for paths from reset input pins to the reset inputs of flip-flops of the respective reset synchronizers. These warnings are only generated by Vivado when "false-path" constraints are set for the reset inputs of these flip-flops, which is done by the constraint files for the respective PHY interfaces supplied by the Verilog-Ethernet project. When the set_false_path constraints are removed, the CDC-26 warnings are no longer generated.

As described earlier, the CDC-26 warning informs the designer to review a certain path that uses LUTRAM as a synchronization element, which is only valid if surrounding logic prohibits that the read and write addresses of the LUT carry the same value. As no LUTRAM is involved in the paths for which the CDC-26 warnings are generated (precisely, no LUTRAM is used in any of the three designs, as can be seen in Table 4) and the warnings vanish when the "false-path" constraint for the reset inputs of the concerned flip-flops are removed, it is assumed that Vivado erroneously generates CDC-26 warnings for these paths. However, it might be the case that Vivado intends to generate a different warning for these paths but mischaracterizes it as belonging to the CDC-26 category.

(xii) The **WGE 100** Ethernet MAC is triple-speed capable, once again requiring clock multiplexers. However, as the input delay is not constrained, no CDC-13 warnings concerning the clock multiplexers are generated. The present critical CDC-13 warnings are generated on paths that feed into BRAM cells, with the destination port being clocked by a different clock than the source of the path.

As a further consequence of clock multiplexers being used, some warnings appear to be duplicates as discussed in the context of other cores above.

The design furthermore uses a flip-flop based clock divider to divide the RX clock provided by the PHY by two. Thus, the divided clock is considered as asynchronous to the input clock. A large number of CDC-1 warnings originate from the (G) MII input signals. These signals (data and data valid) are initially sampled directly by the RX clock provided by the PHY. They are then resampled using the divided clock if MII (as opposed to GMII) is selected. Under these circumstances, the source clock and destination clock are considered asynchronous to each other, leading to a large number of CDC-1 warnings. Furthermore, a second flip-flop-based clock divider is used to divide the TX clock. This leads to further CDC-1 warnings and a CDC-4 warning that is caused by TX data signals being resampled by the TX clock directly feeding into the PHY.

One reset synchronizer is driven by combinational logic, causing a critical CDC-10 warning.

Finally, less critically, a number of correctly implemented two-stage synchronizers are part of the design. These are, however, not constrained for Xilinx Vivado (ASYNC_REG), leading to non-critical CDC-2 warnings.

(xiii) Finally, **WhiteRabbit wr-core** contains a number of paths that fan out from one clock domain to another without synchronizers in between, causing the critical CDC-1 warnings reported by Vivado. From the HDL code, "CE-controlled" patterns may be intended here, but not implemented by the synthesis tool as such. Constraining the synchronizers for the control signals for Xilinx tools (ASYNC_REG) may cause these patterns being implemented correctly. These missing constraints are reported on some correctly detected synchronizers as non-critical CDC-2 warnings. Some of these synchronizers, however, are driven by combinational logic instead of by a flip-flop, causing the CDC-10 warnings. This also concerns a small number of reset synchronizers, reported as CDC-7 warnings.

In a single instance, a signal (the PTP timestamp trigger signal) is synchronized with two parallel synchronizers into the same destination clock domain. This is done once on the rising and once on the falling clock edge. This violates Vivado's corresponding rule CDC-11 that forbids the same signal being synchronized twice into the same clock domain.

Finally, multi-bit CDCs (concerning pointers in FIFOs) are correctly implemented as Gray code pointers, reported by Vivado as non-critical CDC-6 warnings.

In summary, three groups of cores can be identified based on the analysis of their clock domain crossings: (1) Cores that seem to correctly handle the CDCs present in their respective designs, leading to very few to no problems during CDC analysis. If there are (critical) warnings at all, they concern (relatively) benign patterns like missing ASYNC_REG constraints (that are solvable without changing the source code by the use of a constraint file) or warnings concerning combinational logic feeding into reset synchronizers (e.g., ORing multiple reset inputs). Ariane Ethernet, LeWiz LMAC1 (AXI Variant), Litex Liteeth, NFMAC10G, Opencores Minimac, Opencores XGE_MAC, and all variants of the Verilog-Ethernet project fall into this category. (2) Projects that contain best-practice clock domain crossing logic that is however not correctly implemented by the synthesis tool due to missing constraints or potentially due to coding style. These problems may be fixed by additional constraints via constraint files but may also require restructuring HDL code to make the intended CDC pattern clear to the synthesis tool. Gaisler's GRETH falls into this category. (3) Finally, some projects contain both a large number of paths between clock domains, and at least some of these paths have been shown to not include (correct) synchronization logic. Some of these paths may have been considered "safe" by the authors, as they—for example—"change only seldomly" (code comment in P. Kerling's Ethernet MAC) or "[are] available long time before its actual use" (in the "Ethernet IP Core Design Document" for Opencores Ethmac). This should then, however, be documented clearly and on a per-path basis—either in the form of constraints (analysis tools often allow to "waive" CDC warnings on specific paths) or in long-form documentation. In some cases, clearer documentation of which clock inputs are considered to be in phase to each other would also be desirable. It might even be necessary, if a core in this category is to be used, to review the identified potentially unsafe CDC in detail and patch the core's HDL code with safer CDC patterns. Opencores Ethernet Tri Mode and Ethmac as well as P. Kerling's Ethernet MAC, WGE-100, and White-Rabbit fall into this category.

Finally, while the analysis presented above may be used to guide selection of a MAC IP core, it is certainly beneficial for a user of any of these cores to perform a CDC analysis of the entire design that also helps to identify problems that may arise in the interface from user logic to the selected core.

4. Evaluation Scope

In addition to the analysis of the OOC synthesis results (i.e., resource, linting, and CDC analysis results) discussed in Section 3, a subset of the identified MAC cores was also subjected to evaluation in physical hardware. This allows testing the interoperability with known-working network hardware (i.e., PC Ethernet interfaces) and measuring the performance of the evaluated cores. In terms of performance measurements, the receive and transmit latency of the cores as well as their supported throughput was measured.

This evaluation was carried out on a subset of the cores shown in Table 4. This subset was selected for evaluation due to the following criteria:

(i) C1 (**Ethernet speed**): Due to their ubiquity in today's network infrastructure, especially when considering the context of embedded systems, the core to be evaluated shall support 10/100 Mbit/s or 1 Gbit/s Ethernet speeds. This precludes MACs from evaluation that only support Ethernet speeds above 1 Gbit/s, e.g., 2.5 Gbit/s, 10 Gbit/s, and higher, such as LMAC2 and LMAC3, as well as NFMAC10G, XGE_(LL)_MAC, and the 10 Gbit/s variant provided by Verilog-Ethernet.

(ii) **C2** (**documentation**): The evaluated MACs shall provide sufficient documentation and/or example code that provides guidance in how to port the core to a new hardware platform. Both Ariane-Ethernet and An Ethernet Controller provide little to no documentation or implementation examples at all, and thus these cores were not evaluated. Furthermore, a detailed analysis of the source code of Ariane-Ethernet revealed that its Ethernet component heavily draws on an older version of Verilog-Ethernet's source code and basically only adapts its AXI-Stream interface to a memory-mapped control/status register and frame buffer interface.

(iii) **C3** (**porting effort**): The porting process to a "standard" 10/100 Mbit/s or 1 Gbit/s Ethernet platform shall involve no extensive implementation work other than (1) porting memory-based components—e.g., FIFOs—as well as clock and I/O components to the target FPGA and (2) the development of an interface module that allows the MAC to loop back received data with slight changes (e.g., inverting every received byte). This precludes three cores or variants from evaluation—the variants of LeWiz's LMAC that use an inferred FIFO lead to massive usage of FPGA resources, and thus only the variants where these FIFOs were replaced by an instantiated XPM implementation could be evaluated. Opencores Minimac does not provide internal FCS checking and calculation. Evaluating this core in the same way as the other cores would entail implementing a mechanism for calculating and updating a FCS for each outgoing frame. WhiteRabbit wr-endpoint does not provide an MII/RMII/GMII/RGMII interface. Instead, it provides a PCS/PMA interface intended for direct connection to—for example—Xilinx's GTP or GTH hard macros, or Intel's equivalent transceivers.

(iv) **C4** (**variant selection**): Finally, some cores provide different variants, for example, in the form of different interfaces to the PHY. Here, a selection was made for which variant(s) to evaluate. This decision was based on which variant fits the available target platforms for 100 Mbit/s or 1 Gbit/s best.

The selection process can be seen in Table 11.

Thus, we arrive at eight projects to be evaluated in total (in ten variants—two for Litex Liteeth and Verilog-Ethernet each). Of these eight projects, a subset of seven projects was evaluated with a 100 Mbit/s Ethernet capable platform. Another subset of six projects was evaluated with a 1 Gbit/s capable platform, with an overlap of five projects (Litex Liteeth, Opencores Ethernet Tri-Mode, P. Kerling's Ethernet MAC, Verilog-Ethernet, and WGE 100) that were evaluated on both platforms.

5. Evaluation Setup

In order to perform the function, throughput, and latency tests in a hardware implementation of the cores selected in Section 3, an evaluation platform is needed. The main task of this platform is to instantiate the respective core under test, supply it with the required clock signals, configure the core via its control interface, and finally to operate its data interface. Furthermore, the platform needs to adapt the individual MAC's PHY interfaces to the PHY available on the board to be used. In this section, Section 5.1 describes the general setup of the evaluation platform, and Section 5.2 details the MAC-specific hardware needed to operate each core, as well as adaptations made in order to be operable with the test platform.

5.1. Approach and Evaluation Platforms. Evaluations were carried out on two different Xilinx-based platforms: Cores capable of 100 Mbit/s Ethernet speed were evaluated on a Digilent BASYS3 development board, which is based on an Artix 7 FPGA. As this board does not provide Ethernet connectivity by itself, an external RMII PHY—a Microchip LAN8720 [50]—was externally connected. Cores that are capable of 1 Gbit/s Ethernet speed were additionally evaluated on a Kintex Ultrascale-based AVNET KU040 development board. This board provides two Texas Instruments DP83867 Gigabit-capable RGMII PHYs [51].

The main functionality of the harness implemented on both platforms is to implement a loopback of received frames on the user-side interface, allowing to send back frames received from an external Ethernet device. As each MAC provides a different user interface—some provide AXI-Stream or FIFO interfaces, others access to memory-mapped buffers, and others require DMA buffers—this module needs to be implemented for each MAC individually. In order to allow to differentiate between frames sent by the external device and those looped back by the MAC under test in a packet capture file, this interface block negates each data word in the loopback process. This furthermore has the effect that the MAC has to calculate a new FCS, and thereby this functionality is also tested by the performed evaluations.

Additionally, our platform supports the intended latency measurements by taking timestamps at certain points in the receive and transmit paths of the respective cores, as well as measurement of the received Inter-Frame Gap (IFG). Measuring the received IFG allows to roughly judge the throughput achieved by the Control PC and is done by counting clock cycles between deassertion and re-assertion of the respective RMII or RGMII *valid* signal. If the PC achieves the maximum bandwidth specified by the respective Ethernet standards, the expected IFG for 100 Mbit/s Ethernet is 960 ns and 96 ns for 1 Gbit/s Ethernet; if the achieved bandwidth is lower due to factors influenced by the packet generator software, operating system, or network interface hardware, a longer received IFG is expected. For latency measurements, timestamps are taken in the receive path at the reception of special Ethernet frames on the PHY interface and at the first indication of the core's user-side interface that a new frame is available. In the transmit path, timestamps are taken when a frame to be transmitted is placed into the responsibility of the MAC and at the time the frame is visible on the PHY interface. The exact point where the timestamp for received and transmitted frames is taken differs between the RMII (100 Mbit/s) and RGMII (1 Gbit/s) version. In the latter case, the timestamp is taken after conversion of RGMII to GMII. This is done because RGMII requires special DDR IO resources that are directly connected to an FPGA pin and thus cannot fan out to a second receiver. Furthermore, the PHY employed in the RGMII variant requires some configuration that is sent via MDIO. This is not necessary with the used RMII PHY.

A final task of our hardware platform implemented on the target FPGA is to adapt the PHY interface provided by each core to the PHY interface provided by the board (RMII for 100 Mbit/s and RGMII for 1 Gbit/s) in case the respective PHY does not support the interface supplied by the board. A block diagram of our platform can be found in Figure 2.

The function, throughput, and latency tests are carried out by a host PC (3.2 GHz Intel i5-3470U CPU with 4 cores, 8 GB RAM, Intel 82571EB/GB PCIe Gigabit Ethernet NIC) running Debian GNU/Linux 11 (Kernel 5.10.0-14). In order to disable automatic IFG adaptation and to provide an IFG above the one specified by the relevant Ethernet standards, a patched e1000e Linux driver was used for the NIC.

Function as well as throughput is tested by transmitting 100000 Ethernet frames of various sizes (48 (padded to 64 bytes by PC's network stack according to the Ethernet standard), 64, 128, 256, 512, 1024, 1400, and 1536(maximum standard Ethernet frame length) bytes) from the PC to the MAC under test. This is done with each of the packet

Table 11: Core selection for hardware evaluation.

Project	Variant	C1	C2	C3	C4	Scope
An Ethernet Controller	–	yes	no			
Ariane-Ethernet	–	yes	no			
Gaisler GRETH	–	yes	yes	yes	–	100M
LeWiz LMAC1	LMAC_CORE1_AXIS	yes	yes	no (FF-FIFO)		
LeWiz LMAC1	LMAC_CORE1_AXIS_XPM_FIFO	yes	yes	yes	no	
LeWiz LMAC1	LMAC_CORE_TOP	yes	yes	no (FF-FIFO)		
LeWiz LMAC1	LMAC_CORE_TOP_XPM_FIFO	yes	yes	yes	yes	1G
Litex Liteeth	liteeth	yes	yes	yes	yes (100M)	100M
Litex Liteeth	liteeth-rgmii	yes	yes	yes	yes (1G)	1G
NFMAC10G	nfmac10g	no				
NFMAC10G	nfmac10g_with_user_intf	no				
Opencores Ethernet Tri Mode	–	yes	yes	yes	–	100M, 1G
Opencores Ethmac	–	yes	yes	yes	–	100M
Opencores Minimac	–	yes	yes	no (FCS)		
Opencores XGE_LL_MAC	–	no				
Opencores XGE_MAC	–	no				
P. Kerling Ethernet MAC	pkerling_ethernet_mac	yes	yes	yes	no	
P. Kerling Ethernet MAC	pkerling_ethernet_mac_with_fifos	yes	yes	yes	yes	100M, 1G
Verilog-Ethernet	verilog-ethernet-eth_mac_10g_fifo	no				
Verilog-Ethernet	verilog-ethernet-eth_mac_1g_fifo	yes	yes	yes	no	
Verilog-Ethernet	verilog-ethernet-eth_mac_1g_gmii_fifo	yes	yes	yes	no	
Verilog-Ethernet	verilog-ethernet-eth_mac_1g_rgmii_fifo	yes	yes	yes	yes (1G)	1G
Verilog-Ethernet	verilog-ethernet-eth_mac_mii_fifo	yes	yes	yes	yes (100M)	100M
WGE 100	–	yes	yes	yes	–	100M, 1G
WhiteRabbit	wr-endpoint	yes	yes	no (PCS/PMA only)		

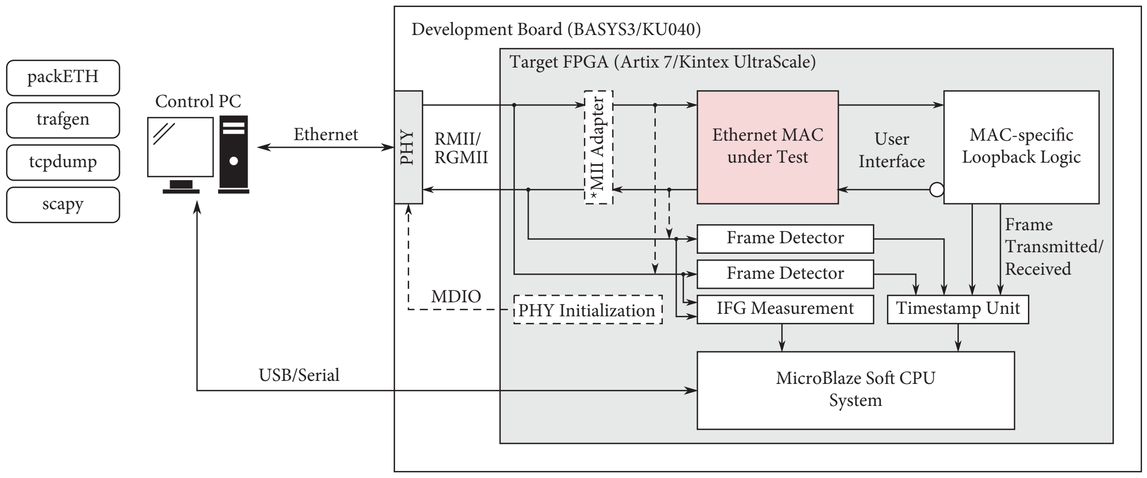

Figure 2: Block diagram of evaluation platform.

generator programs *packETH* (command-line version) (https://github.com/jemcek/packETH, accessed: May 5, 2023) and *trafgen* (part of the netsniff-ng toolkit, see https://netsniff-ng.org/, accessed: May 5, 2023) at the maximum rate the host PC can achieve. Concerning throughput, it must be noted that not only the throughput of the MAC but also of the MAC-specific interface hardware modules, implemented for this work, is tested. Thus, if frame loss does occur, the evaluated MAC might not be the single culprit.

For transmission with packETH, first a PCAP file containing the frame to be sent is generated by a Python script using the *Scapy* module. This frame contains the Ethernet header to be sent and a payload to bring the entire frame size up to the size to be tested (see above). The payload is initialized with pseudo-random data. Subsequently, this frame is handed to packeth, which replaces first bytes of the payload with a sequence number incremented at each sent frame and a short ASCII text containing the name of the MAC.

Testing with *trafgen* is done to corroborate the results obtained with *packETH*. Here, the same frame sizes are sent, however, with a payload consisting of a repetition of the same byte (0x4C), and without counters. This would, in theory, allow higher transmission rates. As trafgen per default uses multiple CPU threads to generate frames, this would impact the CPU time available for the packet capture tool. In order to minimize frames dropped due to high CPU utilization, *trafgen* was restricted to using only one thread.

The MACs, with the help of the purpose-built interface hardware, loop back each received frame to the PC. Each data byte is inverted in the interface hardware in order to allow differentiation between frames sent by the PC and those sent by the MAC. In parallel, the IFG between incoming frames is sampled and can be queried by the PC via the serial interface. This is done so that the actually achieved average throughput during testing (as achieved by the PC) is known.

The PC, in turn, records every returning frame using the packet capture program *tcpdump* into a PCAP file. This file can then subsequently be analyzed if (1) the appropriate number of frames has been sent back (or frame loss occurred, for example, due to limited throughput in the MAC and erroneous FCS calculation), (2) each frame has the expected frame length, and (3) each frame contains the expected data. One difference between the results for tests with *packETH* and *trafgen* is that only with the former incoming as well as frames can be captured using tcpdump. While both tools use a raw socket in the PF_PACKET protocol family, *packETH* uses the sendto (2) system call to send data, and trafgen places frames directly into a buffer shared with the kernel. Thus, only incoming frames show up in *tcpdump's* PCAP files when transmitting with the latter packet generator.

For the latency tests, 128 individual frames for each frame size are sent using *Scapy*. The RX frame detector monitors for the first 24 bytes of a special Ethernet frame and takes a timestamp in case it is detected. Subsequently, the MAC interface block takes timestamps when a frame becomes available on the MAC's RX interface as well as when the frame is placed into the TX interface. Finally, another TX frame detector monitors for the start of the looped-back special frame and takes a timestamp when it is observed. These timestamps are continuously transmitted by the MicroBlaze CPU to the host PC, which monitors them on a serial interface.

A sequence of events for reception of a frame can be seen in Figure 3, and that for transmission of a frame can be seen in Figure 4. The actually reported reception latency for each MAC is calculated as the difference of the MII timestamp and interface timestamps, minus the transmission duration that remains after the frame has been detected by the RX frame detector. In a similar way, the transmission latency is calculated from the individual timestamps.

5.2. Evaluated Cores. In the following, the cores selected for evaluation as shown in Table 11 are described in further detail. In addition, the MAC interface module developed for each core and the necessary adaptions to port the core to the target platforms are described. Finally, information about the core's receive and transmit sequence is provided along with information about the points where timestamps for latency measurements are taken.

The following MACs were evaluated on the 100 Mbit/s Ethernet platform (BASYS3, RMII, and LAN8720). All MACs were instantiated in their MII variant (even if others were available) and interfaced to the RMII LAN8720 PHY using Xilinx's MII to RMII converter core.

(i) **Gaisler's GRETH MAC** is part of Gaisler's IP core library GRIP. It provides an AMBA AHB/APB memory-mapped interface on the user side and multiple MII variants on the PHY side. An overview of the wrapper can be seen in the block diagram in Figure 5.

This MAC handles receiving and transmitting frames via DMA. It requires the user to set up DMA descriptors for received and transmitted frames containing memory locations where to write received frames to and where to read transmitted frames from. Thus, in addition to this setup, which is done via an APB slave interface, the core also requires access to system memory via its DMA port implemented as an AHB master. This memory contains both the stored frame data as well as the DMA descriptors.

Furthermore, a mechanism is required that monitors the status of the RX and TX DMA descriptors in order to perform the intended loopback functionality in hardware. This mechanism must transfer the addresses and frame lengths of RX DMA descriptors once they hold a received frame to TX DMA descriptors and re-enable the RX and TX descriptors when they are no longer used.

Configuration of the MAC's APB registers is done via a state machine. For holding the DMA descriptors as well as the frame data, a custom AHB module is employed. This module infers memory from VHDL for storage of the data elements discussed above. In addition, however, the AHB is monitored. When an RX descriptor is written back with the frame length and status value, the corresponding address is written into a FIFO additionally. Furthermore, the next available RX buffer (of 8 available buffers) is enabled. The FIFO of available frames to be transmitted back is queried every time the core requests a new TX descriptor.

Timestamps are taken in hardware when (1) the DMA engine writes back an RX descriptor with status and length, indicating that a new RX frame is available in system memory, and (2) when a new TX descriptor is enabled, indicating that a frame to be transmitted is placed in the responsibility of the DMA engine. Furthermore, an informational timestamp is taken when the TX descriptor is written back, indicating that transmission is done.

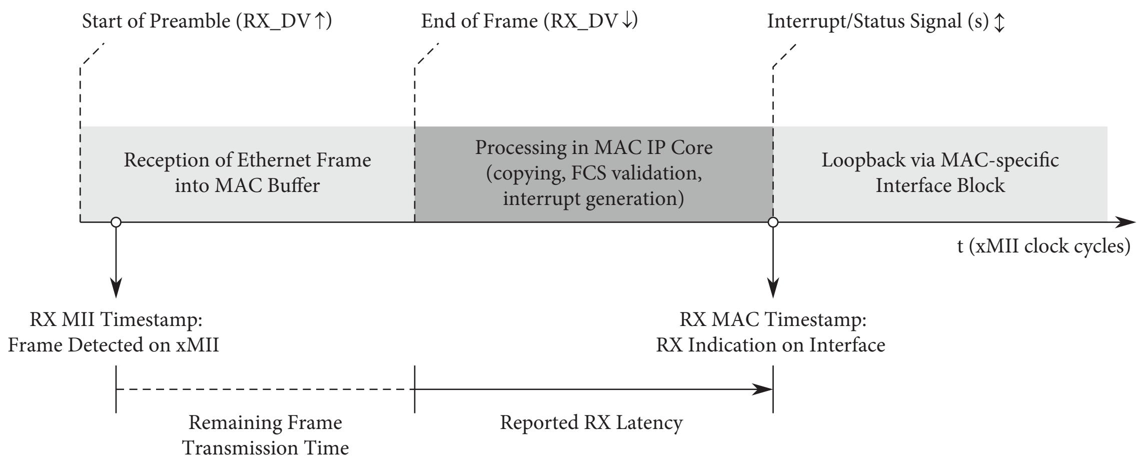
Start of Preamble (RX_DV↑)
End of Frame (RX_DV↓)
Interrupt/Status Signal (s) ↕
Reception of Ethernet Frame into MAC Buffer
Processing in MAC IP Core (copying, FCS validation, interrupt generation)
Loopback via MAC-specific Interface Block
t (xMII clock cycles)
RX MII Timestamp: Frame Detected on xMII
RX MAC Timestamp: RX Indication on Interface
Remaining Frame Transmission Time
Reported RX Latency

Figure 3: Sequence for RX latency measurement.

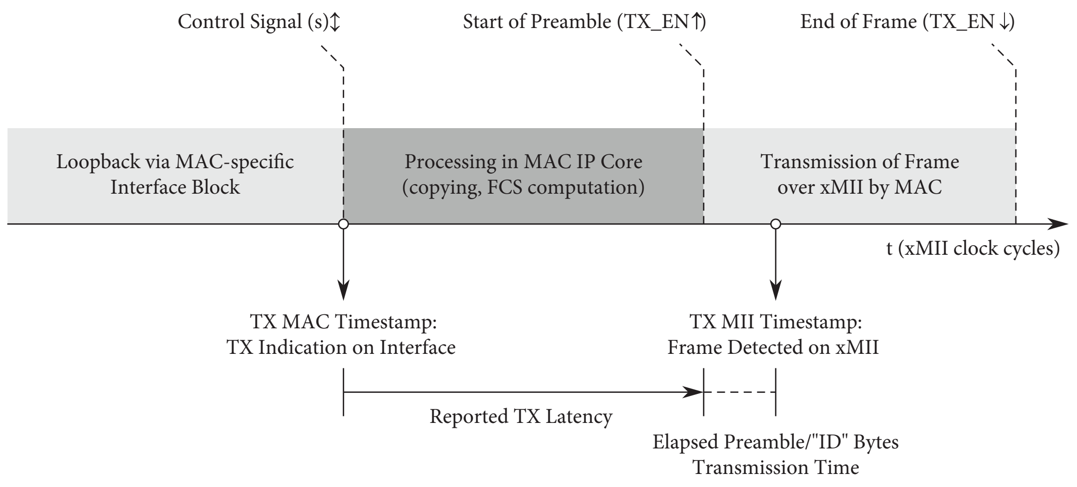
Control Signal (s)↕
Start of Preamble (TX_EN↑)
End of Frame (TX_EN↓)
Loopback via MAC-specific Interface Block
Processing in MAC IP Core (copying, FCS computation)
Transmission of Frame over xMII by MAC
t (xMII clock cycles)
TX MAC Timestamp: TX Indication on Interface
TX MII Timestamp: Frame Detected on xMII
Reported TX Latency
Elapsed Preamble/"ID" Bytes Transmission Time

Figure 4: Sequence for TX latency measurement.

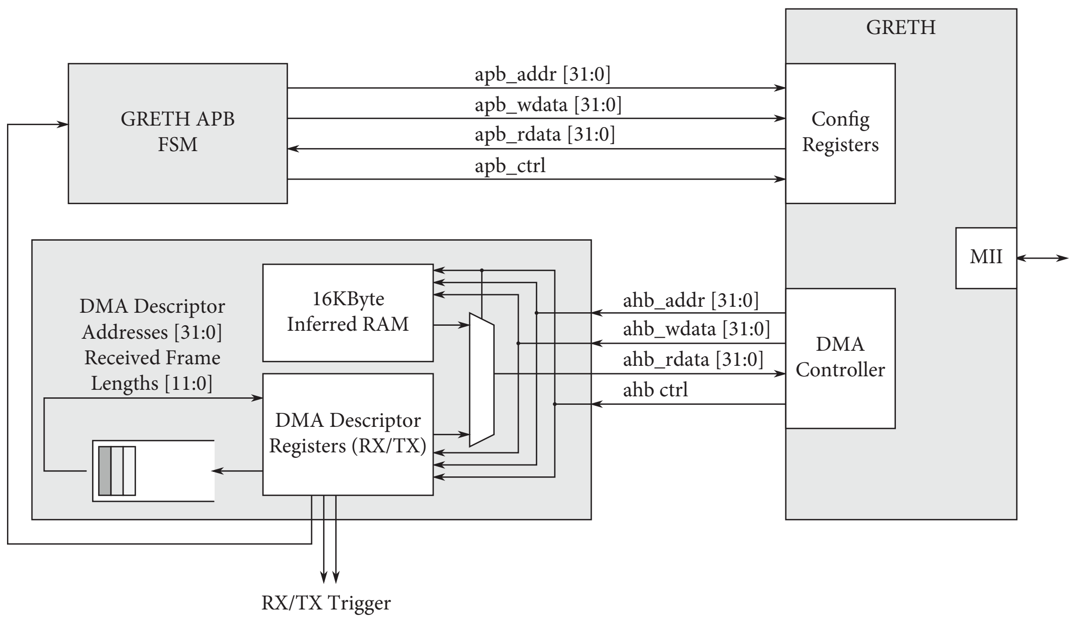

Figure 5: Wrapper for Gaisler GRETH.

(ii) **Litex Liteeth** is a MAC written in Migen, a Python-based environment for describing hardware, which can be exported to VHDL and Verilog. Different user interfaces and capabilities are provided by the project, among others a Wishbone memory-mapped interface that provides access to the received and transmitted Ethernet frames, as well as an Etherbone [52] compatible interface that provides a UDP-to-Wishbone bridge.

For this evaluation, the Wishbone-based bare-metal Ethernet variant was used. This interface provides access to a set of control and status registers as well as internal RAM buffers for two received frames and two frames to be transmitted. This means that for looping back a received frame, it has to be copied from the RX to the TX buffer.

Handling the control interface, as well as copying frames to be sent back, is done by a Finite State Machine (FSM). It first sets up the control registers (configuring the current slot to receive and enabling reception) and then waits for an RX interrupt to occur. Subsequently, the state machine reads the frame length, copies the data to be sent back, and finally writes the TX frame length and enables transmission. The presence of two buffers means that while one frame is being transmitted, another one can be received at the same time. An overview of the wrapper can be seen in Figure 6.

The interface RX timestamp is taken when an RX interrupt is received by the core, indicating that a new frame is available in the corresponding buffer. The TX timestamp is taken when enabling the "SRAM Reader," i.e., the component that reads the frame to be transmitted from the internal transmit buffer.

(iii) The control interface provided by **Opencores Ethernet Tri Mode MAC** is a memory-mapped parallel interface to control/status registers, while the received and transmitted frames are exchanged over a FIFO interface, i.e., an interface consisting of a read/write signal, a 32-bit data bus, a "byte-enable" signal that indicates how many bytes in this bus are valid, and a set of three status signals (start-of-frame, end-of-frame, and data available/free).

Concerning the configuration, the high and low watermark values for the receive and transmit FIFOs need to be set via the register interface, as well as the Ethernet speed. This is done by a state machine before operation, after which frames are simply looped back by connecting the TX FIFO interface back to back to the RX FIFO interface.

The RX timestamp is taken on the rising edge of the RX FIFO's "data available" signal, indicating that a received frame is present on the FIFO interface. In turn, the TX timestamp is taken on the rising edge of the end-of-frame signal that feeds into the TX FIFO interface.

(iv) **Opencores Ethmac** is controlled by a Wishbone slave interface that allows to access a bank of configuration registers. Furthermore, this slave interface includes a memory-mapped buffer area to write DMA descriptors into. Received frames are stored in external memory using a Wishbone master, which also fetches frames to be sent from external buffers.

A principle similar to Gaisler's GRETH is followed for looping back frames received via this core: A state machine configures the MAC initially (i.e., sets the configuration registers and configures an initial RX DMA descriptor). Subsequently, the interrupt line is monitored until a new frame is received. The state machine then reacts by (1) setting the pointer in the RX DMA descriptor to the next buffer segment, and re-enables it so that the next frame can be received, and (2) storing the buffer address and frame length of the currently received frame into a FIFO. Once the RX DMA descriptor has been enabled, the state machine checks if the MAC's transmitter is ready to send a new frame. If this is the case, the next buffer address and frame length are retrieved from the FIFO and configured as a TX DMA descriptor. Once this descriptor is updated and enabled, the MAC transmits the stored frame back to the PC.

For this purpose, in the wrapper module, an XPM FIFO has been instantiated, and a Wishbone-compatible byte-enabled RAM buffer (for storing received frames) has been inferred. An overview of the wrapper can be seen in Figure 7.

The receive timestamp is taken when the state machine handles an RX interrupt, while the transmit timestamp is taken when handing off a frame to the transmitter is finished (by loading the next TX DMA descriptor).

(v) **P. Kerling's Ethernet MAC** needs minimal configuration (e.g., Ethernet speed if not using auto negotiation via MDIO) on some RTL ports. In the *MAC-with-FIFOs* variant, the data interface is a relatively straight-forward FIFO interface consisting of conventional read/write, 8-bit wide data, and empty/full signals. However, as the frame length is stored in the same FIFO, there is a protocol to be followed when reading a received frame or writing a frame to be sent. The first two bytes contain the frame length; subsequently, the frame content is stored. This protocol is implemented in a state machine that communicates to the core via these FIFO ports.

Several components needed to be ported to the Artix 7 based target platform in order to evaluate this core:

(1) For triple-speed operation, the core uses clock multiplexers to select between the MII and GMII clock signals. As it is used in single-speed

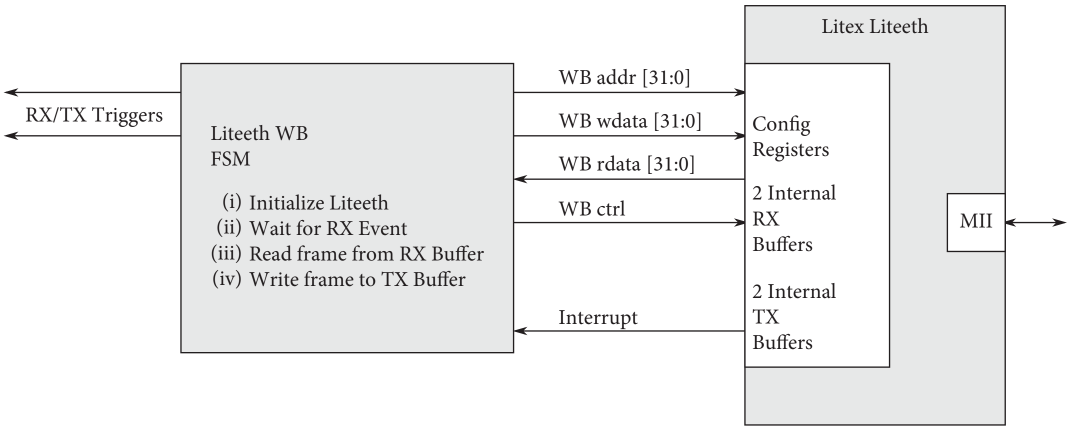

Figure 6: Wrapper for Litex Liteeth.

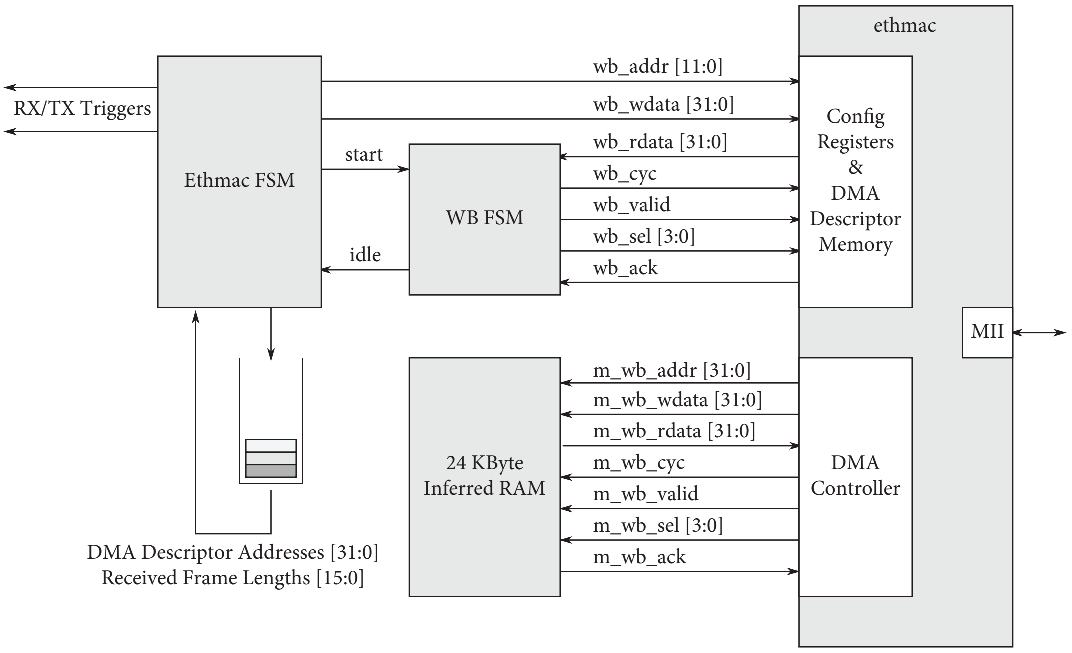

Figure 7: Wrapper for Opencores Ethmac.

mode for our evaluations, this construct was removed.

(2) The core provides FIFO components as Xilinx ISE IP core files. These needed to be upgraded to be usable with Vivado for Artix 7.

(3) Some I/O components needed to be removed/ deactivated. The original core is intended to directly interface to a GMII PHY. It directly instantiates I/O buffers in an attempt to guarantee a certain clock-to-input/output delay and to make sure that some registers are directly packed into the I/O resources. However, as the core is instantiated in a wrapper and communicates to the PHY via an MII-to-RMII converter, these direct instantiations of I/O resources are no longer needed and in fact not accepted by the implementation tool.

The RX timestamp is taken when the state machine waits for the RX FIFO to indicate that it is no longer empty, while the TX timestamp is taken when writing the last byte to be sent to the TX FIFO.

(vi) All MACs provided by the **Verilog-Ethernet** project allow only minimal configuration (the IFG to be transmitted via an RTL port). They provide two AMBA AXI-Stream interfaces for received frames and frames to be transmitted. These interfaces include the standard AXI-StreamTLAST signal to indicate the end of a frame, while the beginning is assumed once data become first available after a reset and after the end of the previous frame has been indicated. On the RX interface, data become only available if the frame checksum has been checked. Thus, once the AXI-StreamTREADY signal is asserted by the core on this interface, it can be assumed that it only is deasserted when the frame has been read out in its entirety.

This interface allows for an extremely simple loopback implementation—just connecting the two AXI-Stream interfaces back to back.

The RX timestamp is taken when the TREADY signal is asserted, while the TX timestamp is taken when placing the last frame to be transmitted on the TX AXI-Stream interface (i.e., when the TX TLAST signal is asserted).

(viii) Finally, **WGE 100** employs a similar interface to P. Kerling's MAC: Both RX and TX data are accessed via a FIFO interface with data, start-of-frame, read/write and full/empty signals. Before the actual frame data, the frame length is stored in either FIFO. However, in contrast to the interface provided by P. Kerling's MAC, this core provides a 32-bit interface. In this case, the upper 16 bits of the first word in the RX FIFO contain the frame length, requiring that this length is stored in the TX FIFO unaltered, while the remainder must be inverted (as inverted data shall be sent back as stated above). This is done by some multiplexing in the wrapper; otherwise, the loopback logic consists of connecting the RX and TX FIFOs back to back.

This MAC, similar to P. Kerling's MAC, instantiates some specialized I/O resources to drive the GMII interface. As this interface is connected to an internal MII-to-RMII converter, this is not needed and was replaced by a direct connection.

The MAC provides FIFO instantiations for Xilinx Spartan 3E devices in the form of NGC netlists generated by Xilinx ISE. These netlists are compatible with Artix 7 and thus could be used directly without re-generating the FIFO IP cores.

The receive timestamp for this core is taken when the start-of-frame signal of the RX FIFO interface is activated, while the transmit timestamp is taken on the falling edge of the write signal to the TX FIFO—this is possible because once a frame is indicated to be available in the RX FIFO, it can be read out (and thus written to the TX FIFO) without ever deasserting the read/write signals.

The MACs Litex Liteeth, Opencores Ethernet Tri Mode, P. Kerling Ethernet MAC, Verilog-Ethernet, and WGE 100 were evaluated on the 100 Mbit/s platform as well as on the 1 Gbit/s platform. On the interface side, no significant changes were needed in most cases in order to support 1 Gbit/s Ethernet speed.

Both Litex Liteeth and Verilog-Ethernet provide support for RGMII, and thus no adapter was needed in between MAC and PHY for these cores. However, the other MACs only support GMII. These cores were adapted to the available RGMII PHY by using the applicable converter provided by the Verilog-Ethernet project.

Furthermore, for the operation of the 1 Gbit/s PHY, implementing an MDIO interface was required in order to configure the PHY to 1 Gbit/s Ethernet speed. This was done for all evaluated MACs using code provided in an example from the Verilog-Ethernet project.

The following significant adaptations were needed for performing the evaluations on the 1 Gbit/s Ethernet platform (KU040, RGMII):

(i) **LeWiz's LMAC1** was only evaluated on the 1 Gbit/s Ethernet platform. Thus, an additional interface module tailored to the core's interface had to be implemented. The configuration for the core (i.e., Ethernet speed, local MAC address, and support for broadcast frames and promiscuous mode) is done via RTL ports.

As a data interface on the user side, this MAC provides a FIFO interface. However, it does not only provide RX and TX FIFOs but also a FIFO that contains the packet lengths of received frames. For reception, the "frame length" FIFO must first be read out, and then the appropriate number of words from the data FIFO. For transmission, first the frame length needs to be written to the TX FIFO followed by the frame content. Handling the two receive FIFOs and the transmit FIFO was done using a state machine. This state machine also detects when the frame length and data FIFOs are inconsistent (when more data *should* be available in the data FIFO according to the frame length read earlier, but the data FIFO indicates emptiness). In this case, a reset of the core is triggered.

Furthermore, while it is claimed in the core's documentation that GMII is supported, the core actually implements an XGMII-like interface. In contrast to standard XGMII, which is 32 bits wide, this interface is only 8 bits wide. It however requires a start-of-frame and end-of-frame symbol indicated using a single "control" signal (standard XGMII has four control signals). This interface is adapted to GMII using a purpose-built interface module that inserts control symbols in received GMII transfers and removes them from outgoing transfers.

The RX timestamp is taken when the RX data FIFO's "empty" signal is deactivated, while the TX timestamp is taken by the state machine when all data have been written to the TX FIFO.

An overview of the wrapper can be seen in Figure 8.

In addition, the inferred FIFOs provided by the core had to be replaced by instantiated XPM FIFOs as done in the OOC synthesis experiments.

(ii) **Litex Liteeth:** As the RGMII interface provided by this core uses an IDELAY element, the instantiation of a DELAYCTRL primitive is required anywhere in the design. The DELAYCTRL primitive tunes the delay taps of all the FPGA's delay elements to a reference clock. As this primitive is not

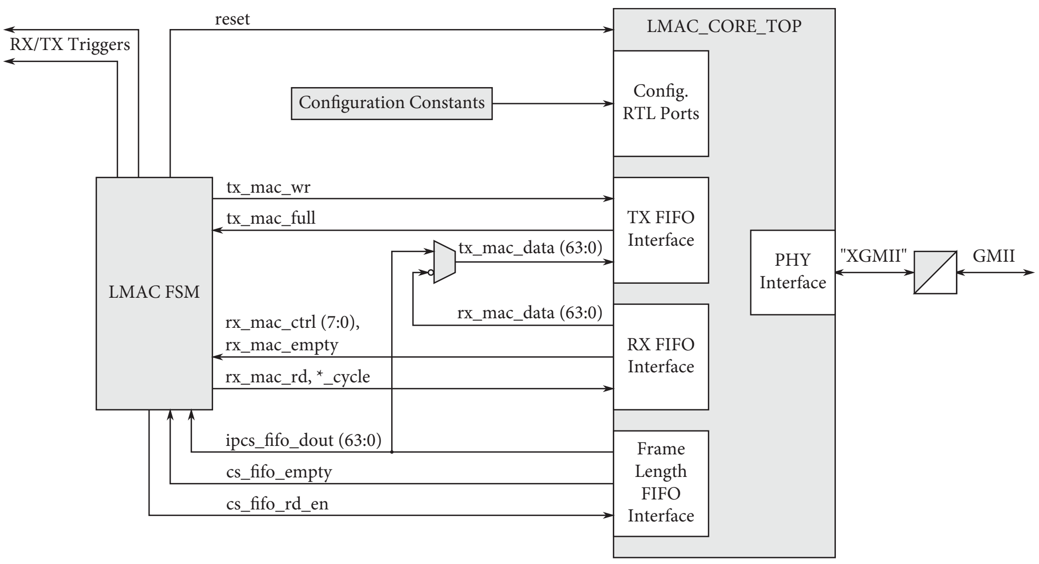

Figure 8: Wrapper for LeWiz LMAC1.

instantiated by the core, the instantiation is however enforced by Vivado's design rule check, and the interface module does this instantiation.

(iii) **WGE 100:** The NGC netlists for the FIFOs required by the core are not compatible with the Ultrascale design flow as reported by Xilinx Vivado. In order to continue using the netlists—and thereby avoiding the need to re-generate these IP cores—the NGC netlists were converted to Verilog netlists using Xilinx ISE 14.7.

6. Evaluation Results

As mentioned in Section 5.1, measuring the received IFG may serve as a rough quantification of the throughput achieved by the PC. These values were measured in hardware by counting the number of RMII or RGMII clock cycles between the deassertion and subsequent assertion of the respective "data valid" signal.

Table 12 shows the average Inter-Frame Gaps (IFGs) and their standard deviations generated by the PC in the 100 Mbit/s experiments when using each frame generator program. It can be seen that (1) the generated IFGs are longer than the minimum transmitted IFG of 960 ns defined in IEEE 802.3-2018 and (2) that there seems to be no significant difference with frame length and between both frame generators in the average output data rate. Table 13 shows similar data for the 1 Gbit/s experiments. The IFGs here once more are longer than the minimum transmitted IFG of 96 ns defined in IEEE 802.3-2018. Furthermore, when the PC sends short frames (especially with packETH), the generated IFGs seem to be significantly longer than when it sends frames in the range of 128 to 1400 bytes. Additionally, the variance is increased when sending especially short frames (48 and 64 bytes) when compared to longer ones. At the maximum frame length (1518 bytes), longer average delays between frames have been measured. It can be seen that the maximum theoretical throughput as allowed by the relevant Ethernet standards was not achieved during these tests. However, the throughput experiments can nevertheless yield information about the ability of the evaluated cores to replicate this "realistic" throughput in addition to an overall function test with reference "known-good" Ethernet equipment.

Table 12: Average PC $\longrightarrow$ FPGA IFGs (100 MBit/s).

Frame length (bytes)	Average IFG (ns)	
	packETH	trafgen
48	1127.95 ± 11.18	1129.20 ± 11.18
64	1128.04 ± 10.36	1128.39 ± 10.76
128	1127.68 ± 10.80	1127.95 ± 11.18
256	1128.30 ± 10.75	1128.66 ± 10.80
512	1127.59 ± 10.94	1128.30 ± 11.55
1024	1127.86 ± 12.38	1129.46 ± 11.19
1280	1127.68 ± 11.60	1125.71 ± 11.50
1400	1127.32 ± 12.56	1127.41 ± 12.15
1518	1128.48 ± 11.42	1128.39 ± 12.17

Table 13: Average PC $\longrightarrow$ FPGA IFGs (1 GBit/s).

Frame length (bytes)	Average IFG (ns)	
	packETH	trafgen
48	424.50 ± 179.28	255.17 ± 513.90
64	422.71 ± 105.67	135.92 ± 0.81
128	133.33 ± 5.98	136.00 ± 0.00
256	133.33 ± 5.98	133.29 ± 5.99
512	133.33 ± 5.98	136.00 ± 0.00
1024	133.33 ± 5.98	136.00 ± 0.00
1280	133.33 ± 5.98	136.00 ± 0.00
1400	133.33 ± 5.98	136.00 ± 0.00
1518	146.67 ± 11.96	152.00 ± 0.00

For these throughput tests, as discussed in Section 5.1, 100000 Ethernet frames of varying sizes were sent to the MAC, which looped them back to the PC where they have been recorded for checking. When testing with *PackETH*, the frames include an incrementing sequence number well as pseudo-random data in the remaining payload. As *PackETH* allows recording of outgoing and incoming frames, the generated PCAP file could then be analyzed to verify that all sent frames were correctly looped back by the MAC. This was done by verifying the sequence number and the remaining payload against the sent data.

When transmitting data using *trafgen*, the outgoing data cannot be recorded in the same way using *tcpdump*. With *trafgen*, only the content of the received frames and their absolute count could be verified (100000 sent frames should result in 100000 frames looped back by the MAC).

Testing the selection of MACs for 100 Mbit/s (Opencores Ethernet Tri Mode, Opencores Ethmac, Gaisler GRETH, Litex Liteeth, P. Kerling, WGE 100, and Verilog-Ethernet) yielded the following results:

(i) With packETH as frame generator, all of the checks described above for all of the MACs tested at that speed and frame sizes were successful. All 100000 pairs of frames transmitted by the PC were correctly looped back by the MACs under test, and thus 200000 frames could be captured in total.

(ii) With trafgen, only received frames coming from the MAC could be captured. However, summing up the frames that could be captured successfully by tcpdump, it was verified that all MACs transmitted back 100000 frames.

Considering the selection of MACs tested at 1 Gbit/s (Opencores Ethernet Tri Mode, LeWiz LMAC1, Litex Liteeth, P. Kerling, WGE 100, and Verilog-Ethernet), all MACs correctly looped back the frames transmitted to them with the exception of LeWiz LMAC1. When testing LeWiz LMAC1 with *packETH* and when testing it with *trafgen*, some frame loss could be observed. This can be seen in (a) and (b) in Table 4. Here, on the one hand, the expected number of captured frames (200000 for *packETH* and 100000 for *trafgen*) can be seen. On the other hand, these tables show the frames actually captured by *tcpdump*, revealing a considerable discrepancy. The packet capture software *tcpdump* did not report any dropped frames—neither by the Kernel ("Dropped—K") nor the interface ("Dropped—IF"). As this effect occurred at every frame size and did occur only with this MAC, it is assumed that the frames are either dropped due to the MAC-specific loopback logic implemented for LMAC1 or due to problems internal to the core ("Dropped—Core").

Latency results are plotted for the 100 Mbit/s experiments in Figure 9. While there exists some variation between the exact latency of the cores, the cores evaluated for 100 Mbit/s exhibit both receive and transmit latencies in the same order of magnitude. The latency variations between the cores may also be impacted by the signals in the design where the receive and transmit interface timestamps have been taken. Thus, no actually "fastest" core can be selected on the basis of these data. The results, however, may allow the conclusion that all cores seem to be suitable for efficient 100 Mbit/s operation.

Figure 10 shows latency results for the cores evaluated with 1 Gbit/s Ethernet speed. Concerning transmit latency, all evaluated cores are fairly consistent in the time it takes to begin to send a frame—give or take a few clock cycles. The only major difference between the cores can be seen with the receive latency: while five of the six cores tested at 1 Gbit/s forward a received frame to the interface within ≤ 20 clock cycles, it takes more than 200 for LeWiz LMAC1.

7. Discussion and Future Work

In the era of the Internet of Things, many of today's electronic devices implement some kind of network interface with Ethernet being known as one of the most widely used network standards. There is consequently a high demand on available Ethernet implementations for FPGA platforms. Existing commercial solutions for Ethernet MAC IP cores may come with some limitations such as technology dependencies, license fees, and the inability to perform design changes or to add special features. Here, the usage of an open-source IP core can be a solution to overcome these drawbacks. Since to the best of our knowledge no publication could be found that compares available open-source Ethernet MACs on a large basis (see Section 2), we wanted to provide an overview of existing open-source IP cores including an evaluation in terms of performance, resource utilization, or code quality herein.

During our survey, 18 open-source projects could be found at Internet sources like *opencores.org* or *github.com* that have been listed in Table 1. Concerning code quality, the availability of a reference implementation may be a first indicator of the maturity of an IP core (see Table 1 and discussions in Section 3.1). Next, the reports of a logic synthesis tool typically provide important information in that context. One of the 18 projects shown in Table 1 (Opencores Gbiteth) was not synthesizable at all and thus was excluded from further evaluation. The synthesis warnings for the remaining 17 cores are summarized in Table 6. Here, especially warnings in the categories "constraints," "latches," "simulation mismatch," and "structural" have been considered to be serious (see discussion in Section 3.2.3). However, except for the variants of the LeWiz LMAC1 core, most of the analyzed IP cores show relatively few of those serious warnings. Furthermore, in every digital design, special attention should be paid to signals that cross clock domains since improper handling may lead to metastability effects and faults that can be extremely hard to debug. Based on our analysis, the cores Gaisler GRETH, Opencores Ethernet Tri Mode, Opencores Ethmac, P. Kerling's Ethernet MAC, WGE 100, and WhiteRabbit may be problematic in that context, and therefore signal paths mentioned in Section 3.2.4 should be closely examined and improved before one of those IP cores is used in a project.

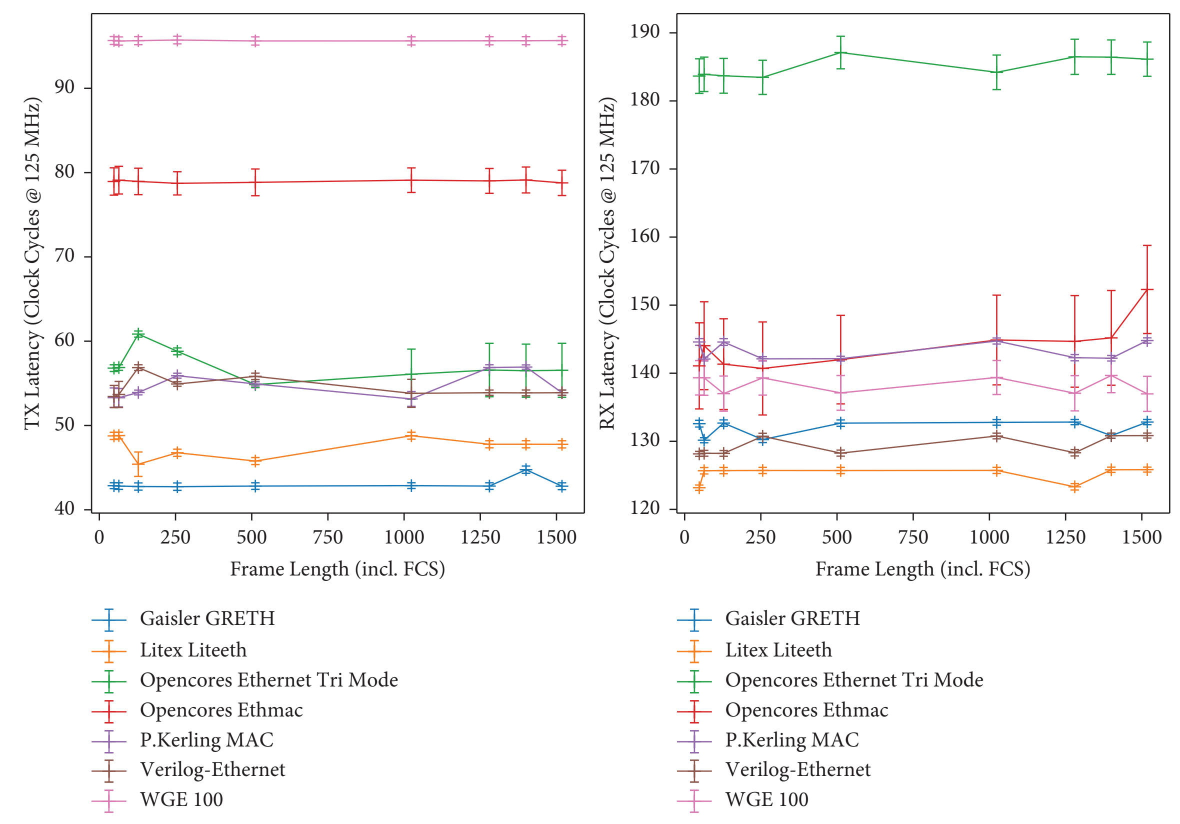

FIGURE 9: Latency measurements for 100 Mbit/s MACs.

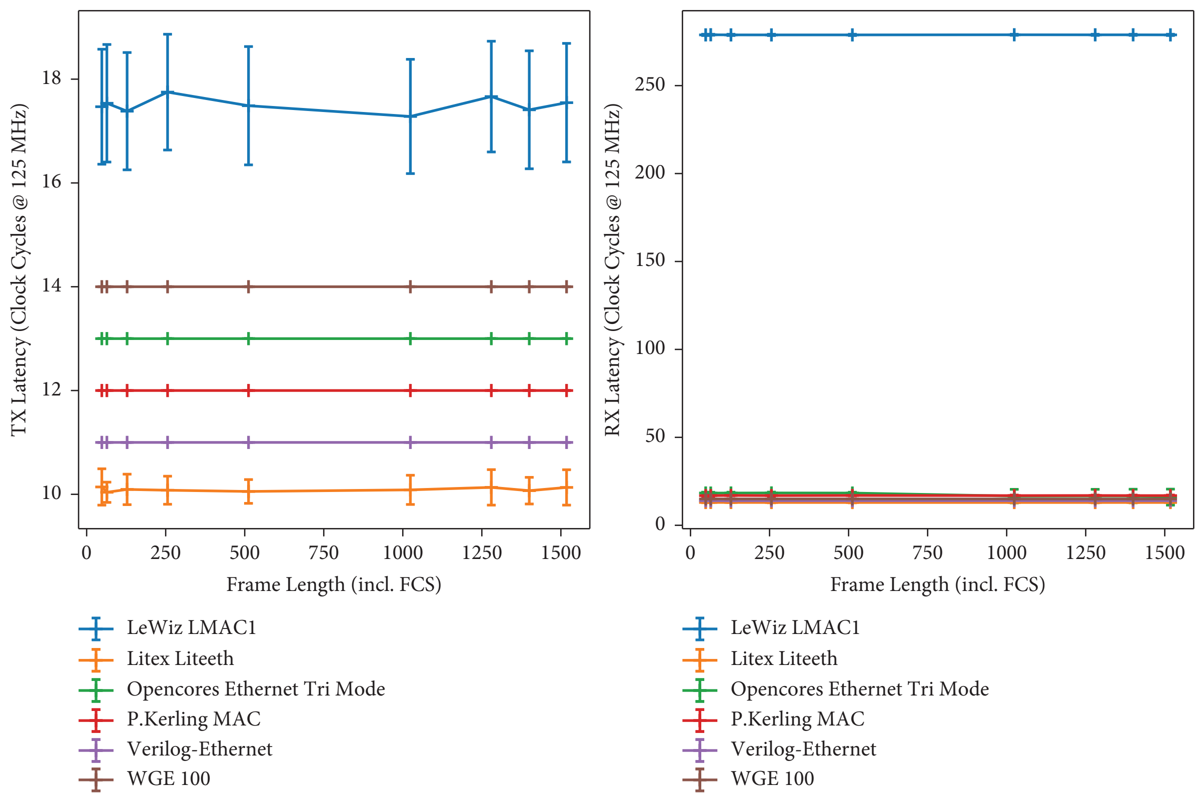

FIGURE 10: Latency measurements for 1 Gbit/s MACs.

Some of the 17 synthesizable cores shown in Table 1 have not been evaluated further due to insufficient documentation or because the porting effort to a specific FPGA technology has been considered as being too high (for details, see Section 4). Moreover, only Ethernet MACs with support for network speeds of either 10/100 Mbit/s or 1 Gbit/s have been selected for a prototype evaluation since these bitrates are most widely used today. This results in a number of eight projects (see Table 11) that have been closely examined on an FPGA-based prototype platform (described in Section 5). As the measurement results in Section 6 show, all of the eight Ethernet MACs could be successfully operated on a real-world hardware platform. However, packet loss could be observed for the LeWiz LMAC1 core (see (a) and (b) in Table 14). As shown in Figure 8, the receive and transmit latencies for the cores evaluated at 100 Mbit/s are in the same order of magnitude while for the cores that have been operated at 1 Gbit/s, the receive latency of the LeWiz LMAC1 MAC is much higher than that of other cores (see Figure 9).

In summary, 16 out the 18 Ethernet MAC IP cores shown in Table 1 can be principally recommended to be used without larger re-design work (Opencores Gbiteth is not synthesizable and for the LeWiz LMAC1, a number of serious synthesis warnings and packet loss during prototyping could be observed). However, the integration and porting efforts of the IP cores An Ethernet Controller and Ariane-Ethernet may be higher than those of other cores while the clock domain crossings of the cores Gaisler GRETH, Opencores Ethernet Tri Mode, Opencores Ethmac, P. Kerling's Ethernet MAC, WGE 100, and WhiteRabbit are strongly suggested to be improved before these IP cores are used in a project.

Concerning the remaining Ethernet MACs and based on our evaluations, there is no "best" IP core. Instead, parameters such as the supported network bitrate (10/100 Mbit/s, 1 Gbit/s, >1 Gbit/s, etc.), the PHY interface (MII, GMII, RGMII, etc.), or the application interface (AXI, Wishbone, propriety interface, etc.) may heavily influence whether an IP core is suitable for a specific application or not (see Table 2).

The same applies to special features such as support for DMA transfers, VLAN tagging, or PTP (see Table 3). This is also reflected by the FPGA resources consumed by an IP core which largely depends on the implemented features (see Section 3.2.2 and Tables 4 and 5). Besides, the license model may impact whether the core can be used commercially at all and which parts of the source code need to be disclosed if being used in a commercial product. On the other hand, the design language (shown in Table 1) should not be an issue since all surveyed cores are available in VHDL, Verilog, or System Verilog, and modern FPGA tools can typically handle all of these languages. However, for two cores (An Ethernet Controller and Litex Liteeth), a special "build tool" must be used in order to generate synthesizable HDL code.

As already mentioned, the selection of an Ethernet MAC highly depends on the intended use case. Let us focus, for example, on applications with requirements on high bandwidth and/or low latencies. Such requirements exist, e.g., in the telecommunications sector where demand for bandwidth is ever increasing due to developments such as High Definition (HD) and Ultra High Definition (UHD) video streaming. Also, in the automotive area where Ethernet is used for quite some time as an in-vehicle network, bandwidth requirements are constantly growing. With the advent of Advanced Driver-Assistance Systems (ADASs) and automated driving, substantially more data need to be exchanged with sensor data coming from different places and have to be distributed to various locations [4]. For data centers which are used for large-scale computation or to host services such as Internet search machines (e.g., Google, Bing, etc.) and video streaming platforms (YouTube, Netflix, etc.), the bandwidth requirements on the network infrastructure are enormous [53]. When such high-performance computing clusters are used to process, e.g., artificial intelligence applications, latencies are important as well since due to the huge network traffic, the network latency heavily influences the time needed for the distributed calculations [54]. In industrial communication systems, low latencies are also often a must to minimize response times [7]. The same applies to control networks in avionics [6].

Based on our measurements described in Section 6, a ranking of the evaluated 1 Gbit/s Ethernet MACs in terms of network speed and latency can be derived as shown in Table 15. This table shows the cores ranked by their average measured transmit latency (in clock cycles at 125 MHz) as well as the average latency measurements and standard deviations for both received and transmitted frames. This ranking may help designers to select an open-source Ethernet core when a high-speed/low-latency MAC is needed for a particular application. Of course, the ranking can be completely different if other requirements than speed or latency are in the focus of an application.

TABLE 14: 1 Gbit/s throughput results for LeWiz LMAC1.

Size	Expected	Dropped			Captured
		K	IF	Core	
(a) Using packETH as traffic generator					
48	200000	0	0	53	199947
64	200000	0	0	58	199942
128	200000	0	0	10	199990
256	200000	0	0	2	199998
512	200000	0	0	14	199986
1024	200000	0	0	4	199996
1280	200000	0	0	19	199981
1400	200000	0	0	9	199991
1518	200000	0	0	198	199802
(b) Using trafgen as traffic generator					
48	100000	0	0	59	99941
64	100000	0	0	66	99934
128	100000	0	0	124	99876
256	100000	0	0	219	99781
512	100000	0	0	158	99842
1024	100000	0	0	871	99129
1280	100000	0	0	1095	98905
1400	100000	0	0	430	99570
1518	100000	0	0	1302	98698

Table 15: Ranking of 1 Gbit/s open-source Ethernet MACs in terms of high speed/low latency.

Rank	Project	Speed	Average TX latency (clock cycles)	Average RX latency (clock cycles)
1	Litex Liteeth	1 Gbit/s	10.09 ± 0.29	12.97 ± 0.16
2	Verilog-Ethernet	1 Gbit/s	11.00 ± 0.00	13.97 ± 0.17
3	P. Kerling MAC	1 Gbit/s	12.00 ± 0.00	16.91 ± 0.29
4	Opencores Ethernet Tri Mode	1 Gbit/s	13.00 ± 0.00	17.52 ± 2.85
5	WGE 100	1 Gbit/s	14.00 ± 0.00	14.99 ± 0.14
6	LeWiz LMAC1	1 Gbit/s	17.50 ± 1.12	278.86 ± 0.35

To the authors' best knowledge, this is the first publication that evaluates and compares existing open-source Ethernet MAC IP cores on a large scale. This work should help designers to select an appropriate open-source Ethernet MAC for an FPGA design and shows possible pitfalls and things to pay attention when using an open-source IP core in general. Finally, the authors would like to show that the open-source community can be also very helpful in the world of hardware in terms of design reuse or time to market.

As part of future work, we hope to evaluate open-source hardware implementations of higher-level protocols such as IP or UDP built on top of Ethernet that have already been mentioned in Section 2. Moreover, a comparison of open-source and freely available TCP/IP network stacks is in our research focus as well.

Data Availability

The source code of all IP cores that have been evaluated in this work is publicly available over the Internet. The Internet links are provided in References.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Austrian Federal Ministry for Digital and Economic Affairs (BM:DW) and the National Foundation for Research, Technology and Development related to the Josef Ressel Center "Innovative Platforms for Electronic-Based Systems" (INES), managed by the Christian Doppler Research Association.

References

[1] C. E. Spurgeon and J. Zimmerman, *Ethernet – the Definitive Guide*, O'Reilly Media, Sebastopol, CA, USA, 2nd edition, 2014.

[2] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, Pearson, London, UK, 5th edition, 2010.

[3] J. Aweya, "Implementing synchronous ethernet in telecommunication systems," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1080–1113, 2014.

[4] K. Matheus and T. Königseder, *Automotive Ethernet*, Cambridge University Press, Cambridge, UK, 2nd edition, 2017.

[5] J. C. Warren, "Industrial networking," *IEEE Industry Applications Magazine*, vol. 17, no. 2, pp. 20–24, 2011.

[6] A. Mifdaoui and A. Amari, "Real-time ethernet solutions supporting ring topology from an avionics perspective: a short survey," in *Proceedings of the 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8, Cyprus, Nicosia, September 2017.

[7] J. Sommer, S. Gunreben, F. Feller et al., "Ethernet—a survey on its fields of application," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 263–284, 2010.

[8] J. S. Rinaldi and J. Wendorf, *EtherNet/IP: The Everyman's Guide to the Most Widely Used Manufacturing Protocol*, Independently Published, Traverse City, MI, USA, 2018.

[9] L. Ding, P. Kang, W. Yin, and L. Wang, "Hardware TCP Offload Engine based on 10-Gbps Ethernet for low-latency network communication," in *Proceedings of the 2016 International Conference on Field-Programmable Technology (FPT)*, pp. 269–272, Xi'an, China, December 2016.

[10] G. Gupta, T. Nowatzki, V. Gangadhar, and K. Sankaralingam, "Kickstarting semiconductor innovation with open source hardware," *Computer*, vol. 50, no. 6, pp. 50–59, 2017.

[11] M. Qian, J. Zhu, Y. Cao, and C. Yang, "Design and FPGA verification of dual-speed adaptive ethernet controller," in *Proceedings of the 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–3, Wuhan, China, September 2011.

[12] Q. Yi, M. Shi, and G. Zhong, "Design and fpga implementation of ten gigabit ethernet mac controller," in *Proceedings of the 2017 IEEE 2nd Advanced Information Technology Electronic and Automation Control Conference (IAEAC)*, pp. 1395–1398, Chongqing, China, March 2017.

[13] T. Xiao, S. Sun, and F. Meng, "Design of an enhanced 10Gb/s ethernet MAC controller for DCB offloading on FPGA," in *Proceedings of the 2015 International Conference on Network and Information Systems for Computers*, pp. 385–388, Taipei, Taiwan, September 2015.

[14] P. Kerling, "Design, implementation, and test of a tri-mode Ethernet MAC on an FPGA," 2015, https://www.db-thueringen.de/receive/dbt_mods_00038245.

[15] M. Ruiz, S. David, G. Sutter, G. Alonso, and S. López-Buedo, "Limago: an FPGA-based open-source 100 GbE TCP/IP stack," in *Proceedings of the 2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 286–292, Barcelona, Spain, September 2019.

[16] T. Uchida, "Hardware-based TCP processor for gigabit ethernet," *IEEE Transactions on Nuclear Science*, vol. 55, no. 3, pp. 1631–1637, 2008.

[17] N. Alachiotis, S. A. Berger, and A. Stamatakis, "Efficient PC-FPGA communication over gigabit ethernet," in *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology*, pp. 1727–1734, Washington, DC, USA, June 2010.

[18] A. Lofgren, L. Lodesten, S. Sjoholm, and H. Hansson, "An analysis of FPGA-based UDP/IP stack parallelism for

embedded Ethernet connectivity," in *Proceedings of the 2005 NORCHIP*, pp. 94–97, Vilnius, Lithuania, November 2005.

[19] J. Sütő and S. Oniga, "FPGA implemented reduced Ethernet MAC," in *Proceedings of the 2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 29–32, Budapest, Hungary, December 2013.

[20] A. Forencich and C. Alex, "Snoeren, George Porter and George Papen. "Corundum: an open-source 100-gbps NIC"," in *Proceedings of the 2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 38–46, Fayetteville, AR, USA, May 2020.

[21] R. Santos, R. Marau, and A. Vieira, "A synthesizable ethernet switch with enhanced real-time features," in *Proceedings of the 2009 35th Annual Conference of IEEE Industrial Electronics*, pp. 2817–2824, Porto, Portuga, November 2009.

[22] M. H. Assaf, S. R. Das, W. Hermas, and M. Emil, "Petriu, and S. Biswas. "Verification of ethernet IP core MAC design using deterministic test methodology," in *Proceedings of the 2008 IEEE Instrumentation and Measurement Technology Conference*, pp. 1669–1674, Victoria, Canada, May 2008.

[23] M. Gayathri, R. Sebastian, S. R. Mary, and A. Thomas, "A SV-UVM framework for Verification of SGMII IP core with reusable AXI to WB Bridge UVC," in *Proceedings of the 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, pp. 1–4, Coimbatore, India, January 2016.

[24] Github, "An Ethernet Controller IP Core," 2023, https://github.com/egk696/an-ethernet-controller.

[25] Github, "Ariane Ethernet IP Core," 2023, https://github.com/lowRISC/ariane-ethernet.

[26] Github, "Gaisler GRETH IP Core," 2023, https://www.gaisler.com/index.php/products/ipcores/ethernet/greth.

[27] Github, "LeWiz LMAC1 IP Core," 2023, https://github.com/lewiz-support/LMAC_CORE1.

[28] Github, "LeWiz LMAC2 IP Core," 2023, https://github.com/lewiz-support/LMAC_CORE2.

[29] Github, "LeWiz LMAC3 IP Core," 2023, https://github.com/lewiz-support/LMAC_CORE3.

[30] Github, "Litex Liteeth IP core," 2023, https://github.com/enjoy-digital/liteeth.

[31] Github, "NFMAC10G IP core," 2023, https://github.com/forconesi/nfmac10g.

[32] Opencores, "Ethernet Tri Mode IP Core," 2023, https://opencores.org/projects/ethernet_tri_mode.

[33] Opencores, "Ethmac IP Core," 2023, https://opencores.org/projects/ethmac.

[34] Opencores, "Gbiteth IP Core," 2023, https://opencores.org/projects/gbiteth.

[35] Opencores, "Minimac IP Core," 2023, https://opencores.org/projects/minimac.

[36] Opencores, "XGE_LL_MAC IP core," 2023, https://opencores.org/projects/xge_ll_mac.

[37] Opencores, "XGE_MAC IP core," 2023, https://opencores.org/projects/xge_mac.

[38] P. Kerling, "Ethernet MAC IP core," 2023, https://github.com/yol/ethernet_mac.

[39] Github, "Verilog-Ethernet IP Core," 2023, https://github.com/alexforencich/verilog-ethernet.

[40] Github, "WGE 100 IP Core," 2023, https://github.com/ros-drivers/wge100_driver.

[41] Open Hardware Repository, "WhiteRabbit IP core," 2023, https://ohwr.org/project/wr-cores/tree/master/modules/wr_endpoint.

[42] M. Montón and X. Salazar, "On licenses for [open] hardware," in *Proceedings of the 2020 35th Conference on Design of Circuits and Integrated Systems (DCIS)*, pp. 1–6, Segovia, Spain, November 2020.

[43] Xilinx, Inc "UG949 (v2019.1) UltraFast Design Methodology Guide for the Vivado Design Suite", https://docs.xilinx.com/v/u/2019.1-English/ug949-vivado-design-methodology.

[44] Xilinx, Inc. "UG835 (v2019.1) Vivado Design Suite Tcl Command Reference Guide", https://docs.xilinx.com/v/u/2019.1-English/ug835-vivado-tcl-commands.

[45] M. Arora, *The Art of Hardware Architecture: Design Methods and Techniques for Digital Circuits*, Springer New York, New York, NY, USA, 2012.

[46] Xilinx, Inc. https://docs.xilinx.com/v/u/en-US/ug471_7Series_SelectIO.

[47] Xilinx, Inc. https://docs.xilinx.com/v/u/en-US/ug472_7Series_Clocking.

[48] Xilinx, Inc. https://docs.xilinx.com/v/u/2019.1-English/ug906-vivado-design-analysis.

[49] Xilinx, Inc. https://docs.xilinx.com/v/u/en-US/ug474_7Series_CLB.

[50] Microchip Technology Inc, "DS00002165B: LAN8720A/LAN8720AI Small Footprint RMII 10/100 Ethernet Transceiver with HP Auto-MDIX Support Datasheet," 2016, https://ww1.microchip.com/downloads/en/devicedoc/00002165b.pdf.

[51] Texas Instruments Inc, "SNLS484G: DP83867IR/CR Robust, High Immunity 10/100/1000 Ethernet Physical Layer Transceiver Datasheet," 2022, https://www.ti.com/lit/ds/snls484g/snls484g.pdf.

[52] M. Kreider, W. Terpstra, J. Lewis, T. Włostowski, and J. Serrano, "EtherBone - A network layer for the Wishbone SoC bus," in *Proceedings of the ICALEPCS 2011*, vol. 111010, Grenoble, France, October 2011.

[53] Y. Liu, K. Jogesh, M. Veeraraghavan, L. Dong, and M. Hamdi, *Data Center Networks: Topologies, Architectures and Fault-Tolerance Characteristics*, Springer Science & Business Media, Berlin, Germany, 2013.

[54] D. Yuan, H. Kan, and S. Wang, "Ultra low-latency MAC/PCS IP for high-speed ethernet," in *Proceedings of the 2020 International Conference on Space-Air-Ground Computing (SAGC)*, pp. 73–75, Beijing, China, December 2020.