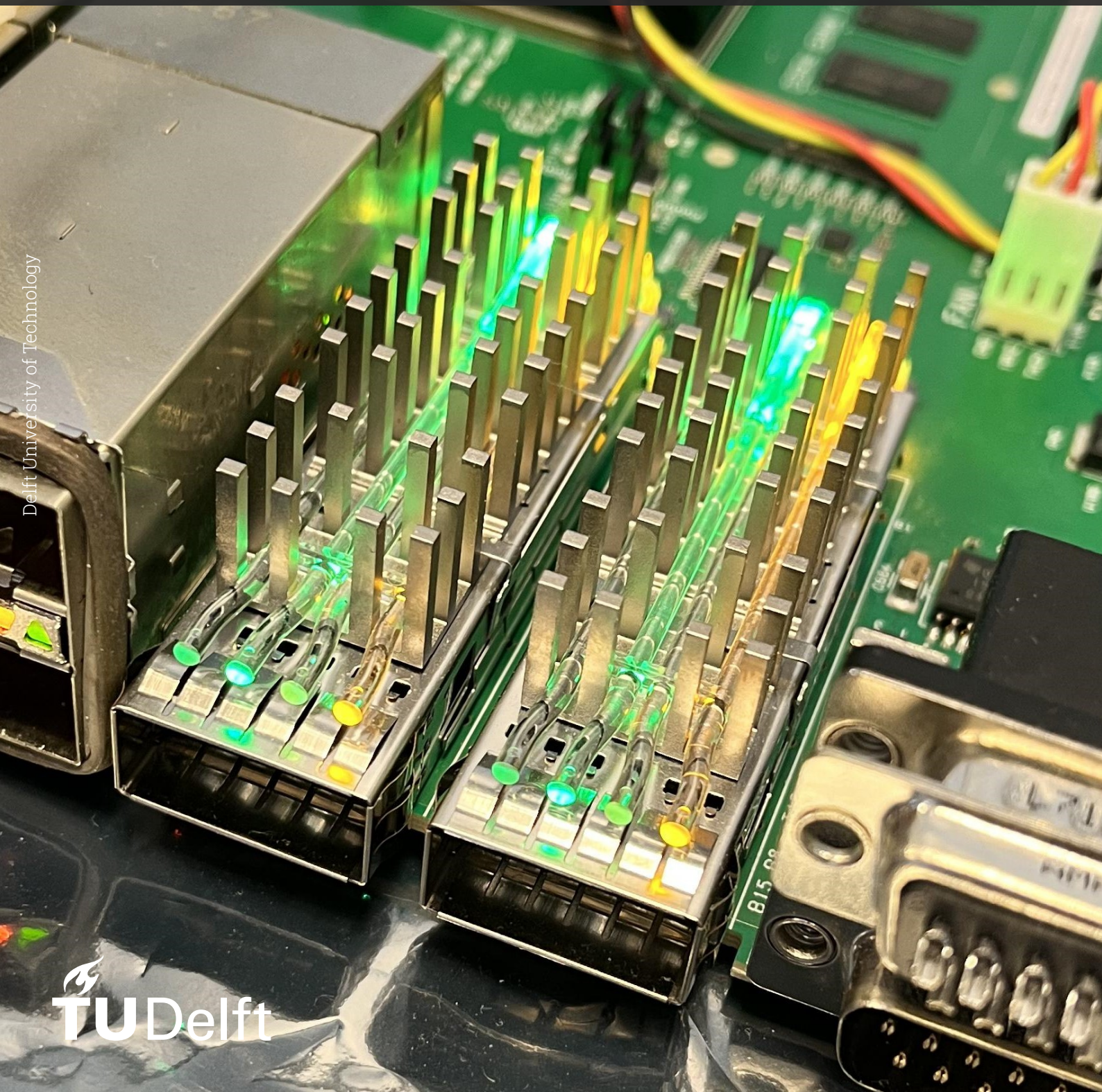
PCIe Over Fiber Optics in FPGA-Based Systems

ES5000: Thesis Project

Lluís Merlos Pieri



Delft University of Technology

TU Delft

PCIe Over Fiber Optics in FPGA-Based Systems

by

Lluís Merlos Pieri

to obtain the degree of Master of Science
in Embedded Systems at the Delft University of Technology

Student number:	5737621	
Project Duration:	November, 2023 - August, 2024	
Thesis committee:	G. Gaydadjiev	TU Delft, Supervisor
	K. Atasu	TU Delft
Company:	A. Walefack	Philips B.V., Daily Supervisor

Cover:	Photograph of a FPGA board by Lluís Merlos Pieri. 12 June 2024
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Preface

This thesis marks the culmination of my work on exploring PCIe technology over fiber as a solution for high-speed data transfer. With the growing demand for faster and more efficient data communication in fields like telecommunications and high-performance computing, I was inspired to investigate how a modular and adaptable design could meet these challenges.

During my time working at Philips, I was exposed to the real-world applications and demands of advanced data systems, which greatly influenced the direction of this research. The project involved extensive research, hands-on experimentation, and overcoming various technical challenges to create a functional system that meets the needs of modern data transfer.

I am grateful to my advisors, colleagues at Philips, and everyone who provided support and feedback throughout this process. Your insights and encouragement have been invaluable in bringing this work to completion.

I hope this thesis contributes to further advancements in high-speed data transfer technologies and sparks future research in this area.

Lluís Merlos Pieri
Delft, August 2024

Abstract

This thesis addresses the growing demand for faster and more reliable data transfer solutions in data-intensive applications, with a strong emphasis on scalability and flexibility. The research centers on the design and implementation of a high-speed data transfer system that utilises PCIe over fiber optics. The main objective is to explore how PCIe technology can be effectively extended over long distances using fiber optics, while maintaining high throughput and low latency, essential for supporting the demands of modern data-intensive environments.

The design and implementation of the system were approached methodically, with careful selection of components and a focus on optimising the system architecture for performance. Detailed experimentation was conducted to evaluate the system's capabilities, resulting in successful data transfer rates of up to 58.02 Gbps using a Gen3 x8 PCIe configuration over a 10-meter fiber link. These findings confirm the viability of PCIe over fiber as an effective solution for high-speed data transfer, while also identifying specific areas where further improvements could be made.

Future work could explore the adoption of newer PCIe generations and the optimisation of critical parameters, such as Maximum Payload Size (MPS). Additionally, there is potential to enhance the system's performance by investigating optical connectors capable of supporting sideband signals. These advancements would further improve the system's adaptability and efficiency across a wide range of application scenarios.

Contents

List of Figures

List of Tables

Nomenclature

ACK	Acknowledgement
C2H	Card to Host
DMA	Direct Memory Access
FPGA	Field Programmable Gate Array
H2C	Host to Card
ISA	International Standard Atmosphere
MPS	Maximum Payload Size
NACK	Not Acknowledgement
OSD	Optical Switch Dispatcher
OSI	Open Systems Interconnect
PCIe	Peripheral Component Interconnect Express
RCB	Read Completion Boundary
TCP	Transmission Control Protocol
XDMA	DMA for PCI Express Subsystem

1 Introduction

1.1. Thesis Motivation

As data-intensive applications continue to push the boundaries of modern technology, the need for faster, more reliable data transfer solutions has become increasingly urgent. This project focuses on the implementation of PCIe over fiber in FPGA-based systems, a cutting-edge solution designed to address these challenges. By combining the high-speed capabilities of PCIe with the long-distance, high-bandwidth potential of fiber optics, this approach aims to create a scalable and efficient interconnect system. The goal is to develop a solution that not only meets the current demands of advanced applications like medical applications that require huge amounts of data but also sets the stage for future innovations in data communication. This research is motivated by the need to enhance the performance and scalability of data transfer systems, ensuring they are equipped to handle the growing complexities of tomorrow's technological landscape.

Medical devices must adhere to stringent certification standards to ensure their safety and efficacy in critical healthcare environments. Because of these rigorous regulatory requirements, the ability to modify or upgrade such devices after they have been certified is severely limited. Therefore, implementing a future-proof, scalable design is crucial. By adopting a design that can easily accommodate upgrades to higher bandwidths or speeds, the impact of these limitations can be significantly reduced. This foresight is essential in maintaining the device's relevance and effectiveness in an ever-advancing medical and technological landscape, where the need for higher performance and greater data throughput is continually increasing.

In light of these challenges, decoupling the device generating data and the processing device allows modular upgrades by integrating more advanced processing technologies as they become available.

Moreover, a decoupled design enables the shared use of processing units across multiple devices, enhancing system efficiency and resource allocation. It also allows for the physical separation of processing units, which can be centralised in dedicated server rooms, thereby optimising space and reducing operational noise in medical settings. Therefore it is important to provide a capable interface that can support large bandwidths and speeds across multiple rooms (e.g., 20m) as well as be scalable to keep up with future needs without major changes. In modern technological settings, the amount of data generated by digital imaging systems is immense and continues to grow exponentially. Data in medical applications such as, high-definition images and precise sensor measurements, require substantial bandwidth to be transmitted in real time without loss of quality or delay, which could compromise patient care. By offering a cost-efficient method to enhance the

processing power of these devices, it becomes possible to analyse data and make decisions more quickly, thereby improving the efficiency and quality of the services provided.

To address their high-speed data processing requirements, Philips is actively researching various cutting-edge technologies within their High-Speed portfolio. The system they are developing is centred around a data generation component, such as an sensor, that produces substantial data traffic. This data is then transmitted via fiber optics to a processing unit, where the necessary computations are performed. The data generation component employs a Field Programmable Gate Array (FPGA) as its platform. FPGAs offers a high degree of flexibility for varied applications and can implement a wide range of communication protocols and technologies suitable for high-speed data transfers.

This thesis explores the implementation of PCI Express (PCIe) over fiber optics as a solution for the architectural needs of medical devices, specifically targeting high-bandwidth and low-latency data transmission. PCIe over fiber involves merging PCIe, a high-speed interface traditionally used for internal component connections within a computer, with fiber optic technology, which facilitates high-bandwidth data transmission over long distances. By integrating these technologies, the goal is to extend PCIe's high performance capabilities beyond the limits of a single device, enabling the swift transfer of data between physically separated components within a medical system.

However, implementing PCIe over fiber presents several technical challenges. One of the primary obstacles is the need to ensure synchronisation and signal integrity over long distances, as fiber optic transmission introduces different latency compared to traditional copper-based PCIe connections. Additionally, the conversion between electrical PCIe signals and optical signals requires specialised hardware, which can introduce complexity and potential bottlenecks. This thesis presents a thorough analysis of the design and implementation process, including the rationale behind choosing PCIe over fiber optics, the technical challenges encountered, and the design decisions made to overcome them.

1.2. Aims and Objectives

This project seeks to explore and develop the use of PCIe technology over fiber as a practical solution for high-speed (50 Gbps), medium-distance (10m) data transfer. Philips is researching the use of multiple technologies in their High-speed portfolio, with a focus on building knowledge around high-speed data transfer from sensors to PCs. The overarching goal of this project is to achieve high-speed, reliable, and scalable data transmission between devices (e.g., sensors) and hosts (e.g., PCs) using well-defined interfaces that can be re-used across a wide range of applications. The Philips's project centres around a common framework with multiple interchangeable technologies, such as Ethernet or PCIe. This thesis specifically studies the viability of PCIe over fiber in this context. Features such as reliability and scalability are considered for such end.

1.2.1. Aims

The specific aims of this thesis are as follows:

1. **To Investigate High-Bandwidth and Low-Latency Connectivity Solutions**: The thesis aims to explore solutions that can achieve high-bandwidth and low-latency data transmission between data sources and processing units. This is critical in settings where the timely and accurate transfer of large volumes of data is essential for functionality;
2. **To Build an Understanding of PCIe Technology**: An essential aim of the thesis is to deepen the understanding of PCI Express (PCIe) technology, focusing on its application, benefits, and limitations within the proposed architecture. This includes examining PCIe's suitability for high-speed data transmission and its integration with fiber optics to meet the demands of modern applications;

3. **To Propose a Decoupled Design**: By decoupling data generation from processing units, the thesis aims to enable modular upgrades, improve system efficiency, and optimize resource allocation without necessitating major changes.

1.2.2. Objectives

The specific objectives of this thesis are as follows:

1. **To Research and Identify Suitable Hardware Platforms for PCIe Over Fiber**: Conduct comprehensive research to identify the state of the art. Then research the most appropriate hardware platforms for high-speed data transmission. This includes evaluating processing units and optical interfaces that can meet the demands of high-volume, high-speed data transfer, while offering the best balance of performance, cost-efficiency, and compatibility with the proposed system design in Chapter 3;
2. **To Evaluate the Viability of PCIe Over Fiber for High-Speed Data Transfer**: Assess the feasibility and performance of implementing PCIe over fiber optics in scenarios requiring high-speed, medium-distance data transfer. This objective involves analyzing the potential advantages and challenges associated with this approach, including scalability, cost, and integration into existing infrastructure.

By focusing on these objectives, the thesis will provide a comprehensive examination of technical solutions for enhancing data transmission. This includes a detailed exploration of hardware platforms and communication technologies, specifically evaluating the suitability of PCIe over fiber optics compared to Ethernet, in achieving the project's goals.

1.3. Contributions

This thesis makes contributions to the field of high-speed data transfer by successfully implementing a system capable of PCIe over fiber optics, developing a scalable system architecture, and providing comprehensive analysis and validation of the proposed solution. These efforts address both the immediate technical challenges and future needs of advanced data communication systems.

The implementation of a system capable of PCIe over fiber to extend the high-speed capabilities of PCIe beyond its traditional boundaries, enabling reliable data transfer across medium distances. This allows for rapid communication between dispersed components, crucial for applications requiring both speed and distance.

The development of a scalable architecture to ensure the system can easily adapt to future demands for higher bandwidths and speeds. Our forward-looking design minimises the need for extensive redesigns, particularly in regulated industries like healthcare.

Finally, the thesis provides **a thorough analysis and experimental validation of the system's performance**, confirming its feasibility and effectiveness. The results demonstrate the impact that the system's parameter have on throughput, latency, reliability, and power consumption. Thus, offering a robust foundation for future advancements in high-speed data transfer systems.

1.4. Structure of the Thesis

The thesis begins with Chapter 2 Background. This chapter contains an assessment of existing high-speed data transfer systems and provides an in-depth explanation of fundamental concepts such as PCI Express and fiber optics, ensuring a comprehensive understanding of the project.

Subsequently, the thesis presents the design of the system in Chapter 3 System Design, detailing the requirements and constraints while outlining the primary methodology employed throughout the research. Further more the proposed design is described in Section 3.2. In each subsection each component that conforms the system is described and specified in detail.

Following the proposed design, Chapter 4 delves into the Implementation phase, where the realisation of the system is thoroughly discussed. This chapter details the strategies and methodologies employed to develop a robust and effective system, ensuring that the design goals are met within the practical constraints of the hardware and software environments. The discussion includes the specific choices made during the implementation, such as component selection, integration techniques, and optimisation approaches that were critical in achieving the desired performance. The challenges encountered during this stage are also described in the last section of the chapter.

Next, the system is analysed in Chapter 5 Experimental Results. This chapter contains comprehensive analysis of the system's performance, based on four core experiments: throughput, link reliability, latency, and power consumption. Each of these experiments is designed to rigorously test the system's capabilities and limitations. The throughput experiment evaluates the maximum data transfer rates achievable by the system under various conditions, while the link reliability experiment assesses the physical layer performance of the data connection. The latency experiment focuses on measuring the delay introduced by the system during data transmission, and the power consumption experiment examines the efficiency of the system in terms of energy use. Together, these analyses provide a well-rounded understanding of the system's overall performance, highlighting its strengths and identifying areas for potential improvement.

Finally, the thesis concludes in Chapter 6 with a concise discussion of the overall findings and their implications. This chapter reflects on the project's success in meeting its objectives, highlighting key insights and evaluating the system's effectiveness. It also considers potential improvements, suggesting areas where further optimisation or expansion could enhance performance. The chapter closes with a final assessment of the project's contributions, emphasising its significance in advancing PCIe over fiber technology and its potential impact on future developments in high-speed data transfer systems.

2

Background

This chapter explores the foundational concepts essential for understanding high-speed data transfer systems. It begins with an introduction to the principles of high-speed data transfer in Section 2.1, highlighting the key considerations necessary for building an effective and efficient system. Next, the chapter provides an overview of fiber optic technology in Section 2.2, which is crucial for enabling fast and reliable data communication. The discussion then shifts to PCIe technology in Section 2.3, detailing its protocol, cabling, and implementation on FPGA platforms. Finally, the chapter offers a comparative analysis of PCIe and TCP/Ethernet in Section 2.4, focusing on their architectural differences, speed capabilities, and cost considerations, providing a comprehensive understanding of the strengths and limitations of each technology in high-speed data transfer applications.

2.1. Introduction to High-Speed Data Transfers Systems

High-speed data transfer systems are designed to move large volumes of data rapidly between devices, systems, or networks. As technology continues to advance at a rapid pace, the demand for faster and more efficient data transfer solutions has become essential. The challenge of swiftly moving vast amounts of data is prevalent across a wide range of industries and is critical in numerous scenarios, including telecommunications, multimedia, computing, and data storage [1].

To address this growing need, high-speed data transfer systems are deployed for both short and long-distance communication. These systems face several key challenges, such as maintaining signal integrity, minimising latency, and ensuring compatibility across various systems and platforms. Achieving reliable high-speed communication requires the implementation of sophisticated error detection and correction mechanisms, precise timing and synchronisation, and effective thermal management to handle the increased power consumption and heat generated by high-speed components.

As industries continue to generate and rely on ever-increasing volumes of data, the ability to transfer this data quickly and reliably has become a foundational aspect of modern technology. High-speed data transfer not only meets current demands but also drives innovation, opening up new possibilities and enabling advancements across various fields.

These systems are comprised of several components or layers depicted in Figure 2.1. This stack of layers is the OSI (Open Systems Interconnection)[2] model. It is a framework that standardises networking functions by dividing communication into different layers, each with specific roles.

1. The Physical Layer handles the physical aspects of the network, including hardware and signal transmission;

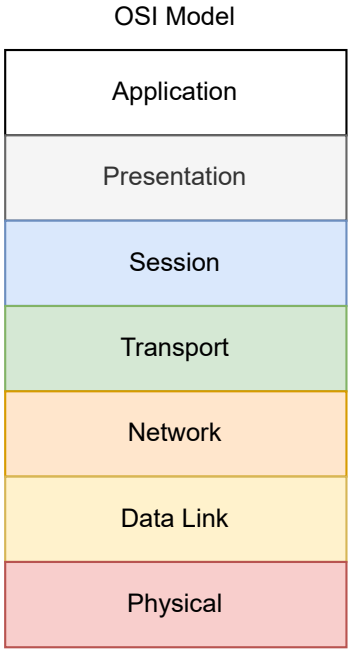


Figure 2.1: Components and layers in a high-speed transfer system.

2. The Link Layer deals with error detection, data framing, and media access control, asserting accurate data transfer across the network;
3. The Network Layer controls the logical addressing and routing, determining the travel path for path;
4. The Transport Layer ensures reliable data transfer, handling error correction, and flow control;
5. The Session Layer manages and controls the connections between computers, maintaining, establishing, and terminating communication sessions;
6. The Presentation Layer translates the data between the application layer and the network, handling encryption, compression, and data formatting;
7. The Application Layer is the topmost layer that interacts directly with the end-user through software applications, providing network services such as file transfer, email, and web browsing.

Each layer communicates with those immediately above and below it, creating a structured system with bounded responsibilities and tasks. It is important to note that not all technologies and protocols implement all the layers, and they may be skipped if required.

The manner in which users are inter-connected with each other and the rest of the system is called the network architecture. It has a critical influence in determining the overall performance, scalability, and flexibility of the network. These connections directly

affect the design considerations and technical decisions required to build a performant system. Depending on the application's specific needs, trade-offs may be necessary to balance simplicity, performance, and flexibility. For instance, a one-to-one connection is straightforward and offers high performance with minimal complexity. However, this approach lacks flexibility and may not be suitable for larger networks, where scalability and dynamic user interactions are essential.

Network components can be connected through various topologies, such as point-to-point, star, bus, or mesh configurations. Each topology has its own set of strengths and weaknesses [3] when it comes to performance and ease of management.

Point-to-point, Figure 2.2, connects two Users directly through a link. The connections are simple and reliable, ideal for direct communication between two users but limited in terms of scalability.

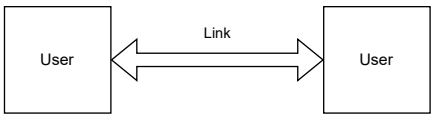


Figure 2.2: High-speed data transfer system using Point-to-point topology

Star topologies, Figure 2.3, centralise connections between User devices on a central Hub, offering ease of management and control. This differs from the point-to-point in the path data takes between User devices. In Point-to-point networks each device directly connects to another. Oppositely, in Star topologies all devices are connected to a central Hub, which is responsible for re-directing the traffic on to the correspondent User device. This centralisation of traffic requires a Hub capable of sustaining the loads coming from each possible path. To summarise, star topologies are more efficient requiring less links while maintaining access to all devices, on the other hand Point-to-point does not require additional and complex components at the expense of harder to scale networks.

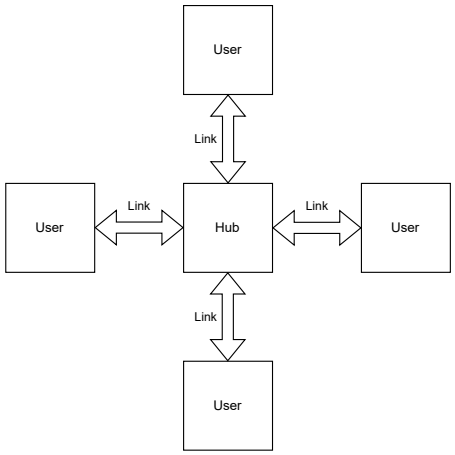


Figure 2.3: High-speed data transfer system using Star topology

Bus topologies, Figure 2.4, connect all Users using a shared communication Link called Bus. They enable shared communication channels but can suffer from congestion and scalability issues if the Link is not properly designed.

Mesh topologies, Figure 2.5, connect Users to other Users directly, forming paths. These paths allow communication between Users using other Users as intermediary nodes.

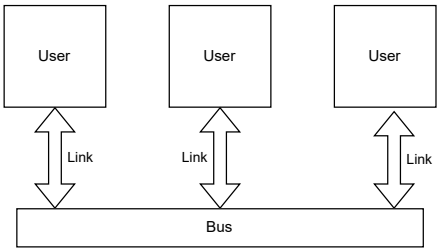


Figure 2.4: High-speed data transfer system using Bus topology

This topology, while complex, provide high redundancy and resilience, making them suitable for systems requiring robust fault tolerance at the expense of increased costs an underutilised resources.

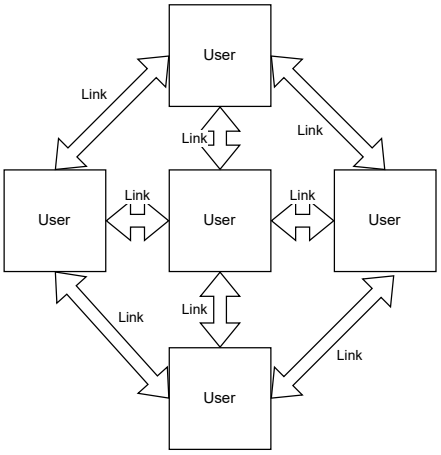


Figure 2.5: High-speed data transfer system using Mesh topology

Selecting the appropriate topology involves careful consideration of these trade-offs, ensuring the network can effectively support the intended activities while meeting performance and reliability requirements. This choice is fundamental for optimising the network's ability to handle the specific demands of the application. Whether it involves a small, simple setup or a large, intricate system with multiple users and connections.

2.1.1. Key Considerations for an Effective High-Speed System

When analysing high-speed systems intended for use across various diverse scenarios, several key factors must be considered to ensure the effectiveness and adaptability of the system under study.

The most important one being throughput, described as the measure of the amount of data successfully transmitted over a network or communication channel within a given period of time [4], which determines the system's ability to handle large volumes of data at a speed. For solutions intended to have longevity and evolution, equally important is scalability, which assesses the system's capacity to adapt and grow to increasing demands without requiring significant re-design and implementation effort. This scalability is crucial in scenarios where future expansion or enhancements are necessary, as it allows for performance improvements without the need for a complete redesign.

Ease of use is also an important factor, particularly in the case of a system involved in a variety of products. A solution that is intuitive and straightforward to implement, that adheres to common standards reduces the complexity and time required for integration, making it more attractive for diverse applications. A standardised interface minimises the effort required to integrate and adopt the system to new products, promoting interoperability and re usability of various components. This allows to make it easier to incorporate the high-speed system into various environments and use cases.

2.2. Fiber Optic Overview

Fiber optics is a technology that utilizes thin strands of glass or plastic fibers to transmit data in the form of light signals. Unlike traditional copper wiring, which relies on electrical signals, fiber optic cables use light to transfer information, allowing for rapid data transmission over long distances. Each fiber is made up of a core, where the light travels, and a surrounding cladding that reflects the light back into the core, minimizing signal loss during the journey. This structure enables fiber optics to carry vast amounts of data with minimal attenuation and is immune to electromagnetic interference (EMI), making it a highly efficient medium for data communication [5].

The advantages of fiber optics in high-speed data transfers are substantial, primarily due to its ability to handle much higher bandwidths than copper cables. This makes fiber optic networks capable of transmitting large volumes of data quickly, which is crucial for applications like internet infrastructure, data centers, and high-performance computing systems. Additionally, fiber optics can maintain the quality of the signal over longer distances without the need for repeaters, making it ideal for connecting devices and systems spread across large geographical areas. Its resistance to EMI also ensures consistent and reliable data transmission, making it indispensable in environments where high-speed, reliable, and secure data transfers are essential [5].

2.3. PCIe Technology Overview

Peripheral Component Interconnect Express (PCIe) [6] is a high-speed interface standard used to connect hardware components within a computer. It is the evolution of the older PCI and AGP standards, but designed to be more flexible and much faster. It's the main way that internal components like graphics cards, SSDs, network cards, and others communicate with the CPU.

It is based around the concept of a lane: two pairs of differential signals for transmission and reception. The combination of multiple lanes make a link of a certain width, usually x1, x2, x4, x8 or x16.

The main strengths of PCIe is its scalability and flexibility. You can use a device that requires fewer lanes in a slot that provides more, such as plugging a PCIe x1 Wi-Fi card into a PCIe x16 slot. This flexibility ensures compatibility across a wide range of hardware and future-proofs your system to some extent. Furthermore, PCIe standards are designed to be both forward and backward compatible. This means you can use a newer device in an older slot and vice versa, though with some limitations. As the transfer rate is doubled after each version of PCIe, a mismatch of versions within device will resort to the lowest common denominator.

The PCIe interface is a replacement for the older PCI/PCI-X. An important difference is that although its often called a bus it is not the case. The PCIe architecture is actually based in point-to-point topology, having separate links connecting the endpoints (devices) and root complex (host). These difference allows for the link to not be arbitrated so there is no limitation on parallel access from multiple endpoints.

2.3.1. PCIe Protocol

The PCIe [6] protocol is structured into several layers that work together to facilitate high-speed data transfer between devices and the motherboard. The key layers include: Physical, Data Link and Transaction layers.

PCIe employs packets to communicate between the different components. This packets are crafted in the Data Link and Transaction layers.

Each layer adds additional information that allow for encapsulation in the communication as seen in Figure 2.6. That is to say each layer is only responsible for its responsibilities and ignores and delegates the rest of the information to the appropriate layer. The blue sections correspond to the physical layer and contain the extra bits resulting from the encoding. Green sections relate to the Data Link Layer and are used for ordering the packages and detecting errors. Finally the orange blocks correspond to the Transaction Layer and contain information such as the type of package and additional information (header, the payload (data) and a secondary error detection mechanism (ERCR).

Figure 2.6: Data encapsulation in PCIe: Blue is Physical Layer, Green is Data Link Layer and Orange is Transaction Layer

The Physical Layer, is the foundation of the PCIe protocol, responsible for the actual transmission and reception of data across PCIe lanes. It manages the electrical and mechanical aspects of the connection, including physical signalling and clock synchronisation.

Beyond simple data transfer, the Physical Layer also handles initialisation, maintenance control, and status tracking of the PCIe link. It plays a crucial role in framing and encoding data, using encoding schemes such as 8b/10b in Generations 1 and 2 or 128b/130b in fruther Generationsd. These encoding schemes are designed to ensure sufficient transitions between 0s and 1s, which are necessary to prevent DC offset and to facilitate accurate clock recovery from the data signal.

Unlike some other communication protocols, PCIe does not require the explicit sharing of a clock signal between components, often relying instead on a sideband signal for synchronisation in traditional copper-based connections. However, when PCIe is extended over fiber, this sideband signal is typically absent. In such cases, the clock recovery mechanism becomes essential for maintaining synchronisation between devices, ensuring reliable and accurate data transfer.

The Data Link Layer is critical for ensuring reliable data transfer between two directly connected PCIe devices. It is responsible for error detection and correction, employing mechanisms such as Cyclic Redundancy Check (CRC) to identify any issues in the transmitted data.

If an error is detected during transmission, the Data Link Layer prevents the corrupted packet from being passed on to the higher Transaction Layer. Instead, it initiates a re transmission process to correct the error, ensuring that only accurate and verified data reaches the Transaction Layer.

To facilitate this process, a retry buffer is employed, which temporarily stores packets until they are successfully acknowledged by the receiving device. If an error is detected, the data in the retry buffer is re-transmitted. The acknowledgement, or lack of thereof, as well as the retry messages are a responsibility of this layer. This approach guarantees that the data received by the Transaction Layer is accurate and free of transmission errors, maintaining the integrity of the communication.

The Transaction Layer Packet (TLP) layer in PCIe is responsible for managing high-level data transactions between devices. It packs data into packets, known as Transaction Layer Packets (TLPs), which are then transmitted across the link.

The packets of this layer handle various types of operations, such as memory reads and writes, I/O requests, and configuration accesses. The Transaction Layer uses headers to provide the necessary information such as the type of transaction based on the command and address fields. Once the packet is formed, it is passed down to the Data Link Layer for error checking and reliable delivery.

The TLP layer also includes mechanisms for flow control, ensuring that data is transmitted at a rate that the receiving device can handle. This layer is crucial for coordinating complex data exchanges between PCIe devices, ensuring that transactions are correctly processed and delivered to their intended destinations [6].

Two main parameters of PCIe, the Maximum Payload Size (MPS) and the Read Completion Boundary (RCB) have a direct effect on the size of TLP packages [6].

MPS defines the largest amount of data, in bytes, that can be sent in a Transaction Layer Packet (TLP). It has a direct effect on the efficiency of data transfers as with larger MPS values, more data can be sent in each packet, reducing overhead. However, the MPS must be supported by both communicating devices, reducing to the lowest denominator in the case of a mismatch.

On the other hand, RCB specifies the boundary at which read completions are split. It determines how data is divided when responding to a read request, influencing how data is fragmented and reassembled. The RCB value is typically either 64 bytes or 128 bytes, and it helps optimise the handling of read requests, particularly in systems with different memory configurations or performance characteristics.

2.3.2. Cabling

PCI-SIG, the organisation launched its Cabling Work Group in 2003[7] to define requirements for external cable assemblies and connectors to support scalable PCIe links at 2.5 GT/s. This group continued to develop these specifications alongside the main PCI Express Base Specification.

In 2015, the OCuLink Work Group [7] was formed to standardise PCIe platform architectures on a common form factor, supporting both internal and external PCIe cabled connections at speeds of 2.5, 5.0, and 8.0 GT/s. Despite its initial progress, the OCuLink Work Group faced challenges in keeping up with the evolving PCIe Base Specification and closed in 2021.

PCI-SIG is actively addressing the industry's cabling needs through three work groups: the Electrical Work Group (EWG), Cabling Work Group (CWG), and Optical Work Group (OWG). In 2021, a sub-work group within the EWG developed the CopprLink Internal and External Cable specifications for PCIe 5.0 and 6.0, offering support for signalling at 32.0 and 64.0 GT/s.

These specifications use industry-standard connectors, facilitating faster adoption and reducing costs. CopprLink cables are designed to evolve with future bandwidth generations, benefiting next-generation, latency-sensitive applications such as data centre servers, storage, networking, and accelerators.

The specification for an Optical interface has not been completed and as such custom solutions have appeared.

One of these solutions is the Avago's MiniPOD. These are high-performance fiber optic modules designed for short-range, parallel multi-lane data communication and interconnect applications. The compact optical modules are engineered to function over multi mode fiber systems, utilising a nominal wavelength of 850 nm. They each have capacity

for 12 lanes which can be used for an PCI Express Generation 3 with 4 or even 8 lanes if multiple MiniPODs are aggregated, as showcased in [8].

Another comparable product is Samtec's FireFly[9], which offers the added benefit of integrated PCIe solutions, such as the PCOA[10]. This product combines a PCIe switch with FireFly connectors, providing a more convenient and user-friendly interface for extending PCIe off-board.

2.3.3. Implementation Platform: FPGA

PCIe requires special hardware to be used. This hardware implements the specification and provides the functionality. When exploring and deviating from the standard it becomes a necessity the use of a platform that can be easily changed and iterated over.

Field Programmable Gate Array (FPGA), provide the perfect solution for this need. They are a type of integrated circuit (IC) that can be programmed or reconfigured by the user after manufacturing. Unlike traditional Application Specific Integrated Circuits (ASIC), where the functionality is fixed during production, FPGAs offer a flexible and customisable platform that can be tailored to specific tasks or applications [11].

This flexibility allows to implement complex system, such as the PCIe Specification in this case. This is achieved using what is commonly referred to as IP, which stands for Intellectual Property. These are pre-designed, reusable blocks or modules of logic that can be integrated into FPGA designs. Such IP cores are typically developed to perform specific functions, allowing designers to incorporate complex functionalities into their FPGA projects without having to design everything from scratch.

2.4. Comparison Between PCIe and TCP/Ethernet

In modern high-performance computing (HPC) and data processing systems, the acquisition of external data is usually handled with Network Interface Cards (NICs). These specialised hardware components are designed to interface with external networks, capturing data that is transmitted over Ethernet. Whether it's streaming data from remote sensors, servers, or other networked devices, NICs play a crucial role in bringing this external data into the host system for further analysis and processing.

This section explores the key differences between this solution and a PCI Express-only approach, studied in more detail in the full scope of the thesis.

2.4.1. Architecture

To transmit data it must go through a stack of layers. In Figure 2.7 a simplified version of the OSI Model is used. On the left side there is the stack used with a NIC-based Ethernet/TCP communicating system and on the right side there is the stack necessary for a PCIe only solution.

Once the Ethernet traffic is captured by these NICs in the lower layers, it must be transmitted to the host system, where the data can be processed and utilised. This connection is established through the PCIe bus directly to the central processing unit (CPU) and memory of the system. The PCIe bus provides the necessary bandwidth and low latency required for such tasks, enabling rapid data transfer from the NIC to the host.

However, this process presents a significant challenge: the overhead caused by protocol translation. Ethernet data is typically transmitted across networks using the TCP/IP (Transmission Control Protocol/Internet Protocol) stack, a reliable and versatile protocol suite designed to handle various network configurations. When this Ethernet traffic reaches the NIC and is transmitted over the PCIe bus to the host, it must undergo a transformation. The data, originally encapsulated within TCP/IP packets, needs to be

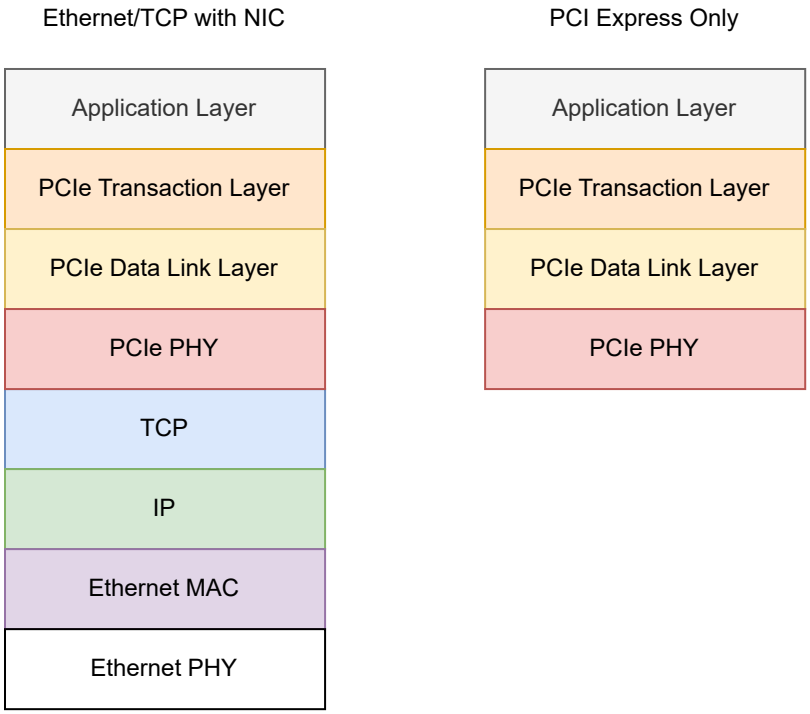


Figure 2.7: Communication stack comparison between a NIC-based system using Ethernet TCP/IP and PCI Express only

unpacked, the headers processed, and then reassembled into a format that the host system's applications can utilize. This translation is not a trivial task; it introduces latency and consumes valuable computational resources, thereby reducing the overall efficiency of data acquisition.

2.4.2. Speed Capabilities

When comparing the speeds attainable with PCIe and Ethernet (over TCP/IP), it's clear that PCIe offers significantly higher data transfer rates. PCIe is designed for high-speed, low-latency communication within a single system, with the ability to achieve speeds ranging from 2.5 GT/s (GigaTransfers per second) per lane in PCIe Gen 1 to up to 128 GT/s per lane in PCIe Gen 7. With configurations that allow for multiple lanes (x4, x8, x16), PCIe can deliver aggregate data rates of up to several tens of gigabytes per second, making it ideal for applications requiring rapid data transfer between internal components like GPUs, SSDs, and NICs.

In contrast, Ethernet, particularly when operating over TCP/IP, typically achieves lower speeds due to the additional overhead and the nature of network communication. Common Ethernet standards like 1 Gigabit Ethernet (1 Gbps), 10 Gigabit Ethernet (10 Gbps), and even 100 Gigabit Ethernet (100 Gbps) offer high-speed networking but still fall short of the raw throughput provided by PCIe. Moreover, Ethernet is subject to higher latency due to network routing, packet handling, and protocol overhead. While Ethernet excels in connecting devices across local and wide-area networks, PCIe is unmatched for high-speed data transfers within a single system, where maximum bandwidth and minimal latency are crucial.

2.4.3. Cost Considerations

A NIC-based solution that incorporates TCP directly in hardware using FPGA technology, especially to achieve speeds like 80 Gbps, tends to be much more expensive to license than an equivalent PCIe core due to the greater complexity and specialized features required. TCP/IP is a robust protocol suite responsible for numerous networking functions, including error handling, flow control, packet sequencing, and congestion management. Implementing these functions in hardware requires a highly intricate design, extensive development, and thorough testing to ensure reliable performance at such high speeds. This significantly increases the cost of the IP, as it must handle the challenges of networking in less predictable environments. In contrast, a PCIe core, while also supporting high speeds, is designed for point-to-point communication in a more controlled setting, making it less complex and thus less expensive to license.

In this chapter, we examined the essential concepts and technologies related to high-speed data transfer systems. We started with an introduction to the principles of high-speed data transfer, emphasising the key factors necessary for creating an effective and efficient system. Following this, we explored fiber optic technology, highlighting its importance in facilitating fast and reliable data communication. Our focus then shifted to PCIe technology, where we discussed its protocol, cabling, and implementation on FPGA platforms. Lastly, we performed a comparative analysis of PCIe and TCP/Ethernet, looking at their architectural differences, speed capabilities, and cost considerations, to provide a well-rounded understanding of the strengths and limitations of each technology in high-speed data transfer scenarios.

3

System Design

In this chapter, the design modelling choices are described in detail. This project is conducted in collaboration with a company as part of a larger research initiative exploring multiple technologies for long-distance data transfers, which imposes certain constraints on the available options for different components. In Section 3.1 there is a detailed explanation of the system requirements, while Section 3.2 presents a comprehensive overview of the proposed System Architecture, including a thorough description of each component and sub-component.

3.1. Requirements

The system requirements outline the specific criteria that the design must meet to achieve the project's objectives. These requirements are derived from the need to ensure high performance, reliability, and compatibility with existing infrastructure. They also have to address the challenges of using fiber optics for medium distance high-speed transfers. The requirements are categorised into functional, performance, and technical constraints.

These requirements provide a clear and detailed framework for the design and implementation of the PCIe over fiber system, ensuring that the final system will meet the necessary standards for high-speed, reliable, and scalable data transmission over medium distances.

3.1.1. Functional Requirements

The system is required to utilise PCIe technology for data transfer between a device and a host, ensuring high-speed communication that meets the demanding performance standards of modern applications. This choice of protocol is motivated by Philips who wants to study the viability of PCIe as a potential solution in their High-speed portfolio.

Additionally, the system must employ optical fiber as the medium for data transmission. Optical fiber is chosen to overcome the limitations of copper cables, particularly in maintaining signal integrity and performance over longer distances. This ensures that the system can support high-speed data transfer across substantial distances without degradation.

Finally, the design must be scalable, allowing for future upgrades and multiple use cases without the need for significant modifications. This scalability ensures that the system can adapt to evolving technological requirements, providing a flexible and future-proof solution for a wide range of applications.

To summarise:

- PCIe protocol: The system must employ the PCIe technology to transfer data between a device and a host;
- Optical: The data must be transferred using optical fiber to overcome the inherent limitations of copper over sizeable distances;
- Scalability: The design must allow for scalability, supporting future upgrades and multiple use cases, without requiring significant modifications.

3.1.2. Performance Requirements

The system is required to support a data transfer rate of at least 50 Gbps, ensuring it remains competitive with other high-speed solutions within the Philips's High-speed portfolio. Additionally, the design must be scalable, with the capability to eventually achieve transfer rates of up to 100 Gbps as future needs arise.

Moreover, the system should be capable of supporting fiber runs of at least 10 meters, ensuring effective data transmission over medium distances without compromising performance. Ensuring data correctness is also paramount; the system must either guarantee the integrity of transmitted data or implement robust error-detection mechanisms to identify and manage any transmission errors.

In aggregate:

- Throughput: The system must support a data transfer rate of at least 50 Gbps, to be comparable with the other solutions in the High-speed Lab, with scalability in mind to eventually reach 100 Gbps;
- Distance capability: The system should support fiber runs of at least 10m;
- Data correctness: The system must guarantee data correctness of the transmission, or at least provide error detecting measures.

3.1.3. Technical Constraints

The system must be based on an FPGA platform, ensuring seamless integration with the existing infrastructure of the Philips' High-speed portfolio. This choice allows for the necessary flexibility and performance required for high-speed data processing tasks.

Furthermore, the system must provide an AXI interface, which is essential for compatibility and integration with other components within the same portfolio. The use of this standardised interface ensures smooth communication between various modules of the system.

Lastly, the host system for data acquisition must be a Linux-based PC. This requirement aligns with the existing setup in the portfolio, facilitating easier integration and consistent operation across all interconnected systems

To encapsulate:

- FPGA based: The device must be based around an FPGA;
- AXI interface: the system must provide an AXI interface;
- Acquisition PC (Host): the host must be a Linux PC.

3.2. System Architecture

After having stipulated the objectives and requirements it is necessary the begin the design process. Based on the aforementioned conditions a top level system is devised in Figure 3.1. Three main components are identified: the device, the link and the host.

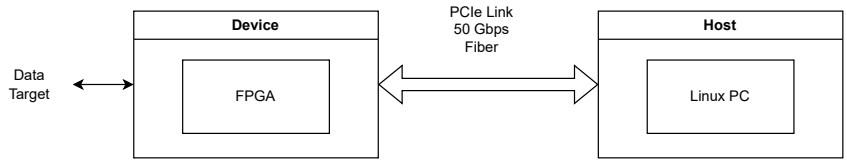


Figure 3.1: Top-level Visualisation of the System's Architecture

In this system, the device acts as the source of data, generating the data that needs to be transmitted, while the host acts as the sink, receiving and processing it. The two components are interconnected through a high-speed communication link, which is crucial for ensuring efficient and reliable data transfer between them.

Building on the requirements outlined in the previous section, we can now define essential aspects of the system's architecture, with a particular focus on the platforms chosen for both the device and the host. The device, which is responsible for originating the data, will use an FPGA as its base platform. The requirement of using an FPGA is warranted due to its flexibility, high performance, and ability to handle complex data processing tasks with low latency, making it ideal for the demands of high-speed communication systems like PCIe over fiber.

On the opposite end, the host, which is responsible for receiving and processing the incoming data, will use a Linux-based PC. A Linux PC is selected as the host platform due to its robust support for PCIe interfaces, extensive software ecosystem, and powerful processing capabilities. Additionally, Linux offers a highly flexible environment that can be fitted to meet the specific needs of the project, alongside great driver support and software ecosystem [12].

This selection of platforms is critical, as it sets the foundation for the entire system, making sure that both components can fulfil the requirements identified earlier. The FPGA provides the necessary hardware acceleration and parallel processing capabilities to manage the data generation and transmission efficiently, while the Linux PC offers the versatility and processing power needed to handle complex data reception and subse processing tasks.

Furthermore, this configuration supports future scalability and flexibility, as both FPGAs and Linux-based systems are highly modular and can be upgraded or reconfigured as project requirements evolve. This ensures that the system not only meets the current project objectives but can also handle future use-cases which may be more demanding.

3.2.1. Device Sub-System

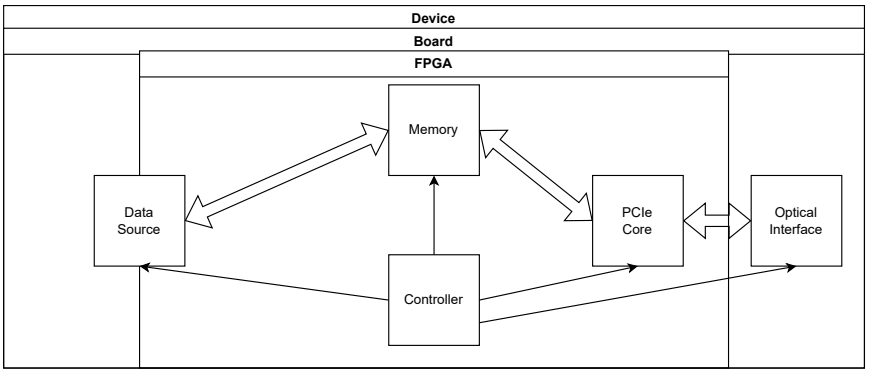


Figure 3.2: Block design of the device

The device architecture is composed of several key functional components, each playing a critical role in ensuring high-speed data transfer within the system. These components work together to manage the flow of data from its origin to its destination, while maintaining the performance and reliability standards required by the system design.

As illustrated in Figure 3.2, the first component in the data flow, starting from the right, is the optical interface. This interface serves as the bridge between the electrical signals within the FPGA and the optical signals used for long-distance communication. Typically, this optical interface is not embedded directly within the FPGA itself but is located on the FPGA's carrier board as a peripheral device. This arrangement allows for greater flexibility and modularity, enabling the system to support various types of optical transceivers depending on the specific requirements of the application, such as distance, data rate, and environmental conditions.

The optical interface is directly connected to the PCIe core, which is responsible for managing all PCIe transactions between the device and the host. The PCIe core is a critical component that handles the high-speed communication protocols, ensuring that data requests and responses are efficiently processed. This core is responsible for providing all the functionality described in the PCIe specification.

To facilitate rapid data access and transfer, the PCIe core is coupled with the device's memory. This memory serves as a temporary storage area where data is either fetched from or written to, depending on the direction of the transfer. The connection between the PCIe core and the memory should be capable of handling the required throughput.

The memory is also linked to the data target, which could be any component or module within the device responsible for generating or collecting the data that needs to be transmitted to the host. This connection allows the data target to efficiently store its output in the memory, where it can be accessed by the PCIe core for transmission.

Overseeing the entire operation is a centralised control unit, which is responsible for configuring, activating, and monitoring all the components within the device. This control unit plays a crucial role in coordinating the activities of the optical interface, PCIe core, memory, and data source, ensuring that they work together seamlessly to achieve the desired performance. It manages the initialisation of the system and can control the other components if the need for it arises.

This centralised approach not only simplifies the management of the device but also enhances its reliability, as the control unit can quickly respond to any issues and maintain the smooth operation of the system.

The width of the arrows symbolizes the communication bandwidth between the modules. The wide white arrows represent the primary data flow, carrying the largest volume of data. It is crucial that the interfaces connected by these arrows are capable of handling the high bandwidth demands to ensure efficient data transfer.

In contrast, the thinner arrows represent control signals, which require much less bandwidth and are not as critical to system performance.

Based on the desired functionality and system requirements, the key characteristics of each component of the device can now be defined.

3.2.1.1. Optical Interface

The optical interface component is responsible for converting data between electrical and optical signals, and as such, it must handle the required bandwidth in both directions effectively. Respecting the shape of the physical layer of PCIe, which consists of a pair of Transmission (TX) and Reception (RX) channels. In the electrical side of things the channels must be formed by differential pairs, but no such restriction exists when changing the medium to optical.

While a straightforward approach might suggest setting a minimum speed requirement of 50 Gbps, a deeper understanding of PCIe allows for a more precise determination. Each PCIe generation defines a base speed for a single lane, with performance scaling as multiple lanes are combined into a single link. These links can only be formed with powers of 2 lanes (1, 2, 4, 8, and 16), which limits the configurations capable of exceeding 50 Gbps. Table 3.1 summaries the speeds defined by each PCIe generation across various link widths. Each generation doubles the previous' one speed, and by aggregating lanes the performance is also increased. The unit used is Giga Transfers per second (GT/s) and it includes the encoding bits. This is why Generations 1 and 2 have a odd base speeds as they use the 8b/10b encoding which result in 2 and 4 Giga bits per second (Gbps). To meet the 50 Gbps requirement, the first viable configuration delivers 64 GT/s. Although Giga Transfers per second (GT/s) and Giga bits per second (Gbps) are not directly equivalent due to encoding overhead, this rate is still sufficient.

From PCIe Generation 3 onwards, a 128b/130b encoding scheme is used, where 128 bits of data are transmitted using 130 transactions (T). This relationship allows us to calculate the actual data rate in Gbps by multiplying by 128 and dividing by 130, yielding 63.02 Gbps, which comfortably meets the 50 Gbps requirement.

Gen	x1	x2	x4	x8	x16
1	2.5	5	10	20	40
2	5	10	20	40	80
3	8	16	32	64	128
4	16	32	64	128	256
5	32	64	128	256	512

Table 3.1: Summary of PCIe speeds across different Generations and Link Widths, measured in GT/s

Thus the requirements for the optical interface can be summarized as:

- **Speed**: Must be at least 64 Gbps;
- **Interface 1:** Must provide an optical pair of channels, Transmission (TX) and Reception (RX) per lane;
- **Interface 2:** Must provide an electrical differential pair of channels, Transmission (TX) and Reception (RX) per lane.

3.2.1.2. PCIe Core

The PCIe Core component is responsible for managing data transfers via PCIe and must be fully compliant with PCIe specifications. As established in the previous section, it must support a minimum throughput of 64 Gbps.

This level of throughput can be achieved through various configurations, so the component must support at least one of these configurations. For instance, in PCIe Generation 2, achieving 64 Gbps requires 16 lanes, with the lane requirement being reduced by half for each subsequent generation.

To ensure seamless integration within the FPGA ecosystem, it is recommended to use standard interfaces between components. A commonly used interface in FPGA designs is the Advanced eXtensible Interface (AXI), which is widely supported by FPGA IP cores.

The key characteristics of AXI relevant to this discussion are its data width and operating clock frequency, both of which must be sufficient to accommodate the required data transfer speed.

Hence the requirements for the PCIe Core are:

- **Speed**: Must be at least 64 Gbps;

- **Interface 1:** Must provide an electrical differential pair of channels, Transmission (TX) and Reception (RX) per lane;
- **Interface 2:** Must provide the data using the AXI interface with adequate data width and clocking;
- **PCIe Generation** support must be one of the following:
  - Gen2 at 16 lanes;
  - Gen3 at 8 lanes;
  - Gen4 at 4 lanes;
  - Gen5 at 2 lanes;
  - Gen6+ at 1 lanes.

3.2.1.3. Memory

The Memory component must be capable of accepting data from both the PCIe core and the data source. Consequently, it must operate at double the data rate to handle both read and write operations from each side effectively.

Furthermore, it is crucial to utilise the previously mentioned generic interface, AXI, to ensure interoperability across the system. This adherence to a standard interface facilitates seamless integration with other components within the FPGA ecosystem.

The memory size will vary depending on the specific application, as the traffic pattern and communication needs significantly influence the necessary buffer size.

Accordingly, the requirements for the memory component are:

- **Speed**: Must be at least twice (2x) 64 Gbps to accommodate for reads and writes coming from both adjacent components;
- **Interface**: Must provide two interfaces, one per component it connects with, for the data using the AXI interface with adequate width and clocking.

3.2.1.4. Data Source Interface

This component is tasked with ingesting or consuming data into memory, making it a crucial link in the data processing chain. To fulfil this role effectively, it must operate at the same high rate as the other system components, supporting a minimum throughput of 64 Gbps. This capacity is necessary to accommodate the maximum bandwidth of the PCIe interface, ensuring that the data can be efficiently stored without creating bottlenecks. The component's performance must be aligned with the overall system speed, guaranteeing that it can handle the high data rates required for seamless operation and maintain the integrity and flow of data through the entire system.

Consequently, the requirements for the data target are:

- **Speed**: At least 64 Gbps;
- **Interface:** Must provide the data using the AXI interface with adequate data width and clocking.

3.2.2. Host Sub-System

As depicted in Figure 3.3, the architecture includes the optical interface, along with the necessary drivers and applications that facilitate communication between the Host PC and the connected device.

The optical interface is crucial in this setup, as it is responsible for converting the incoming optical signals into electrical signals. Once converted, these electrical signals are relayed

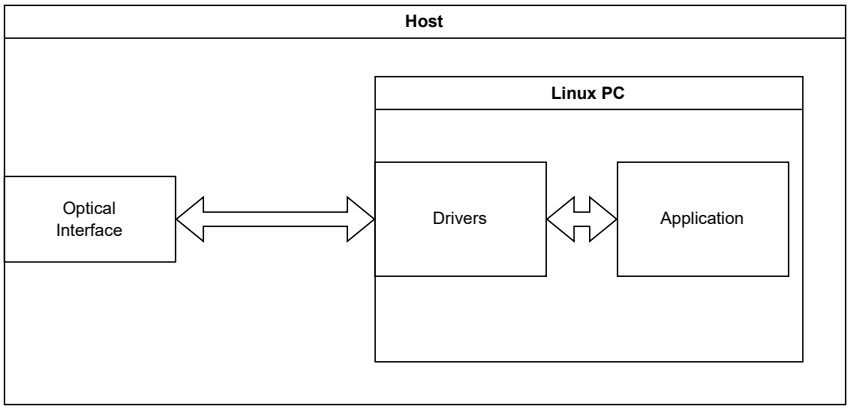


Figure 3.3: Host component block diagram.

to the Host PC via the standard PCIe interface. This conversion process is essential for maintaining the integrity of the high-speed data transfer over long distances, allowing the Host PC to seamlessly interact with the device as if it were directly connected via a conventional PCIe link.

From a software perspective, two primary components are identified: the drivers and the application layer. The drivers are responsible for managing the interaction with the device's PCIe endpoint, utilising the PCIe protocol correctly. They serve as the link between the hardware and software layers, facilitating the smooth operation and management of data transfers.

The application, on the other hand, plays the controlling role on the overall operation of the system. It is responsible for initiating and managing data transfers in both directions, overseeing the flow of information, and ensuring that the system performs according to the desired specifications. The application not only triggers these transfers but also monitors their progress, making adjustments as necessary to optimise performance.

These components combined create a complete host architecture capable of performing the task at hand.

3.2.2.1. Optical Interface

This component has the same required functionality than the optical interface defined in Section 3.2.1.1. As it is tasked with converting data between electrical and optical signals and must effectively manage the required bandwidth in both directions. Following with the structure of the PCIe physical layer, which comprises separate Transmission (TX) and Reception (RX) channels, the electrical connections rely on differential pairs to maintain signal integrity. However, when transitioning to an optical medium, these concept of a differential pair is no longer applicable, offering greater flexibility in signal transmissionis.

The PCIe lane configuration restrictions remain in place, offering several options that can achieve 50 Gbps, all aiming for a target of 64 GT/s.

Additionally, the electrical component is required to interface through the traditional PCB edge connector, ensuring optimal compatibility with standard desktop PCs.

Therefore, the requirements for the host's optical interface are as follows:

- **Speed**: At least 64 Gbps;
- **Interface 1:** Must provide an optical pair of channels, Transmission (TX) and Reception (RX) per lane;

- **Interface 2:** Must provide an electrical differential pair of channels, Transmission (TX) and Reception (RX) per lane in a standard PCB Edge format.

3.2.2.2. Linux PC

This component is responsible for managing data transmissions and ensuring the seamless operation of the entire system. It must be compatible with the specific version of PCIe being utilised, and it must provide an interface with a sufficient number of lanes to support the required data throughput.

Thus, the requirements for the host Linux PC:

- **PCIe Generation** support must be on of the following:
  - Gen2 at 16 lanes;
  - Gen3 at 8 lanes;
  - Gen4 at 4 lanes;
  - Gen5 at 2 lanes;
  - Gen6+ at 1 lanes.

In addition to handling the hardware interface, this component serves as the platform for the software that orchestrates the entire operation. The software architecture is divided into two distinct parts: the drivers and the application.

Drivers

The drivers are crucial for interfacing with the PCIe endpoint across the link, managing the low-level communication and ensuring that data is transmitted and received efficiently. They act as the essential bridge between the hardware and the higher-level operations.

Application

The application, operating at a higher layer, is responsible for controlling the overall data flow. It triggers the transmission processes, produces the data to be sent, and handles the data that is received. This division of responsibilities ensures that the system runs smoothly, with the drivers maintaining stable communication and the application managing the logic and operations that drive the data transmissions. This creates a robust framework that supports the complex requirements of high-speed PCIe communication.

3.2.3. Link: Sub-System Interconnect

The link component plays a critical role in bridging the connection between the Device and the Host, serving as the physical link that enables communication between the two.

As previously outlined, in Section 3.2.1.1, the component must utilise fiber optic cables with a sufficient number of cores to support both the transmit (TX) and receive (RX) pairs. These fiber cores must be capable of sustaining the high-speed data transfer rates required for PCIe communication, with a minimum throughput of 64 Gbps. This ensures that the link can handle the demands of high-bandwidth applications, providing reliable and efficient data transmission between the Device and the Host.

The selection of fiber with the appropriate specifications is essential to maintain the integrity of the PCIe link over potentially long distances, ensuring that the system meets its performance objectives.

Hence the requirements for the data target are:

- **Speed**: At least 64 Gbps;

- **Interface:** Must provide sufficient fiber cores to be utilised for each channel of each lane required:
  - Gen2 at 16 lanes, 32 fiber cores;
  - Gen3 at 8 lanes, 16 fiber cores;
  - Gen4 at 4 lanes, 8 fiber cores;
  - Gen5 at 2 lanes, 4 fiber cores;
  - Gen6+ at 1 lanes, 2 fiber cores.

In this chapter, we have outlined the design of a PCIe over fiber communication system, focusing on the essential components required to achieve high-speed, reliable data transfer. By defining the specific requirements for the Optical Interface, PCIe Core, Memory, Data Source Interface, and Host components, we have established a solid foundation for the system's architecture. These components, each with clearly defined performance and interface standards, ensure seamless integration and optimal functionality within the overall system. Additionally, the chapter addressed the crucial role of the physical link, highlighting the importance of selecting appropriate fiber optic cables to support the high data rates necessary for effective PCIe communication. Collectively, these design considerations aim to meet the project's objectives of achieving a scalable, high-performance system capable of handling the rigorous demands of modern data transfer applications.

4

Implementation

In this chapter, an implementation of the system is described. First, in Section 4.1, the overall strategy employed for developing the system is described. Then in Sections 4.2 and 4.3 the choices made for each component are outlined. Finally in Section 4.4, the challenges encountered during development are disclosed and explained

4.1. Strategy

For implementing the system it is important to follow an structured approach to ensure that the selected components not only meet current requirement but also provide a solid foundation for future upgrades and enhancements.

The first step in choosing components is to have clearly defined the system's requirements. This involves understanding the core functions that the system must perform and the conditions under which it will operate. Having defined the requirement in the previous chapter it is important to lay out which criteria is used to select the components.

The goal with this implementation is to provide a proof of concept that the system is functional and feasible. As well as lay down the work needed to get there.

The next step is to prioritise the key criteria that will influence the selection of components. Criteria used for selecting them include:

- Compatibility;
- Availability;
- Scalability;
- Ease of use.

It is mandatory to ensure that all components are compatible with one another. This includes verifying hardware interfaces (such as PCIe compatibility, connectors and bus speeds) and software support (driver availability and operating system support). Choosing components that are not compatible with each other will lead to system failure or sub optimal performance.

Equally important is selecting components that can be obtained with a reasonable effort. This includes using available FPGA platforms and reusing components from other projects at no cost.

Additionally, given the primary goal of scalability of the design, components should also be chosen keeping in mind further upgrades. For example compatibility with newer PCIe Generations is important even if the rest of the design is constraint on that regard.

Finally, component choice options that are easy to use can be prioritised if the design does not degrade because of it. Ready to use components make the implementation easier and facilitate working towards the goal of the project.

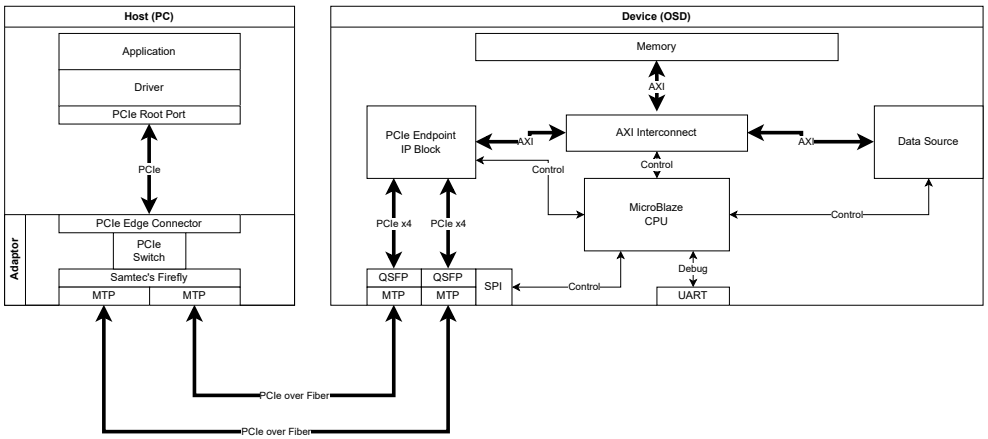


Figure 4.1: Implementation using Optical Switch Dispatcher (OSD) FPGA by Philips

Figure 4.1 provides a diagram summarises the choices made for each components, as well as additional components, such as the UART, added during the implementation. A more detailed explanation will be provided in the following subsections.

The Host is based on a Linux PC and contains uses a custom application with the XDMA drivers[13] and has the PCOA adaptor card by Samtec inserted in the PCIe Root port.

The Device is based on the Optical Switch Dispatcher (OSD) FPGA board. This board was made internally by Philips for multiple projects around CT machines. It uses the PCIe XDMA IP core for PCIe, QSFP28 transceivers configured using an SPI interface controlled by a MicroBlaze CPU[14]. This CPU also controls the PCIe component, the data source and the memory interconnect. Additionally it contains a UART port for debug information. Finally, the memory is implemented using Block RAM and accessed through an AXI Interconnect. Finally, the Link is implemented using two MTP fiber runs.

4.2. Component Choice: Device

As seen in the previous chapter the device is to be based on an FPGA platform. The most useful way for developing FPGA projects is by using development boards. These boards contain all the necessary components to run the FPGA as well as multitude of input and output (I/O) options for the connectivity.

The rest of the components are then constrained to ensure compatibility with the board in mind. This include IPs and IO peripherals availability.

4.2.1. FPGA Platform

For the project several boards are available, including both mass market options by Xilinx and in-house designed boards by Philips.

The OSD board from Philips is selected as the platform due to multiple reasons. It is the board used in the other parallel experiments in Philips' High-speed Lab thus ensuring compatibility and maintain comparability with the other designs. It is equipped with two QSFP28 optical transceiver cages that support up to 28 Gbps per lane and can fit up to 8 PCIe Lanes working at a maximum of PCI Express Generation 4 (16 GT/s).

The FPGA IC is an UltraScale+ XCKU15P-FFVE1517-1-i that is equipped with PCIe Blocks capable of running at Gen 3 with 16 lanes.

Thus, the highest possible PCIe configuration available on this board is Gen 3 with 8 lanes, which satisfies the requirement of throughput (64 Gbps).

4.2.2. PCIe Core

For this platform, two PCIe-related IP cores are available: the UltraScale+ Integrated Block for PCI Express and the DMA/Bridge Subsystem for PCI Express, commonly referred to as XDMA. The primary distinction between the two is that the XDMA IP not only incorporates the UltraScale+ Integrated Block for PCI Express but also integrates a Direct Memory Access (DMA) engine. Additionally, XDMA provides ready-to-use Linux drivers, facilitating easier interaction with the system.

Given these advantages, the XDMA IP is selected as it simplifies the implementation process by offering a comprehensive, pre-configured solution specifically designed for efficient data transmission. This reduces development time and effort, allowing for quicker deployment and more reliable performance.

This IP, provides an full AXI interface which is scaled according the PCIe Generation and Lane configuration. For example, in this implementation it is set to work with a Data Width of 256 bits with a clock at 250 MHz resulting in exactly 64 Gbps. This flexibility allows the component to be easily scaled in the future with higher parameters by increasing the data width or clock frequency according to the future design requirements. The IP also includes test benches to verify functionality and ensure that the design is performing as expected during the simulation stage of the FPGA development process.

Other important features is the inclusion of debug interfaces such as JTAG and IBERT. Which can be used to diagnose issues occurring in the IP.

4.2.3. Memory

Memory is essential for storing data, and the selected board offers two options: Block RAM (BRAM) and DDR4 memory. BRAM is integrated directly into the FPGA's fabric, providing extremely fast access compared to off-chip memory like DDR4. However, BRAM is limited by the internal resources of the FPGA, meaning its capacity cannot be expanded if future needs grow.

On the other hand, DDR4 memory is external, and its capacity can be increased by using larger modules or designing a new board. Both types of memory can be used simultaneously. without penalty. This is done by assigning different memory addresses to each memory component, offering flexibility in design.

In this case, the decision is guided by the ease of implementation. BRAM is chosen because it doesn't require additional circuitry, simplifying the design process. Furthermore, its lower latency compared to DDR4 minimizes the risk of bottlenecks, ensuring that the performance of the optical link is not compromised.

Both components offer a full AXI interface that can be tuned to the appropriate values. In order to provide enough bandwidth for simultaneously read and write, the AXI Data Width is doubled to 512 bits.

An additional IP called AXI Interconnect is used for accessing the memory. As the memory only has one port a way to arbitrate access to it is needed. This IP is used to connect multiple AXI interfaces together. It does so by internally transforming the interface data width and clock when it is miss-matched with the other end of the communication link. This process is done automatically and it also takes care of arbitraging the connections between components to avoid.

4.2.4. Controller

It is not practical to control and command all the other components using exclusively hardware logic. For this end a processor programmable with software is desirable.

MicroBlaze[14] is a soft processor core designed by Xilinx for implementation on their FPGA devices. Unlike a hard processor, which is physically embedded in the silicon of the FPGA, a soft processor is defined in programmable logic, allowing it to be customised and configured according to specific application needs.

This processor can be programmed in C and allows to control components using code. To that end, when choosing its parameters an AXI interface is added. This AXI interface is not performance critical as it is not tasked with transferring data, only control signals. Because of this, the width and clock can be kept low to reduce power consumption.

This choice of component fulfil all the requirements as well as providing easy and flexible upgrade paths for the future.

4.2.5. Optical Interface

The board has equipped multiple options for connecting optical modules: SFP+ and QSFP28. They are both compact network interfaces that allow plugging optical transceivers in a standardised manner. The main difference between the is the speed and amount of lanes they support. The SFP+ is capable of running one lane at 10 Gbps, while the QSFP28 supports 4 lanes up to 28 Gbps. This means that each SFP+ support up to PCIe Gen3 x1 while the QSFP28 does Gen4 x4.

Considering the fact that the board has 10 SFP+ and 2 QSFP28, the maximum achievable PCIe configuration is Gen3 x8 and Gen4 x8 respectively, satisfying the requirements. It is important to note that this configuration with SFP+ would require 8 fiber cables, while only 2 when using QSFP28 at one fiber cable per cage.

Hence, given better scalability (Gen 4 support) and ease of use (two fibers) the QSFP28 is the interfaced used.

The optical transceiver used in the cages is chosen to be compatible with the optical interface of the other side of the link, described in Section 4.3.1, which is MTP-12 [15]. The Cisco QSFP-100G-SR4-S[16] is used for this purpose and the choice is motivated by Samtec, the manufacturer of the PCIe Adaptor Card, described in Section 4.3.1, which recommends this model. This module is designed for use with 100GBASE Ethernet up to 100m and has a max data rate of 4x 25.78Gbps, which is sufficient for PCIe Gen 4 x4, that requires 4x 16 Gbps.

4.3. Component Choice: Host

As seen in the previous chapter, the base platform of the Host is a Linux PC. This is a requirement motivated by the ease of developing with Linux and its trajectory of use within Philips. Linux is more flexible and customisable and these qualities are invaluable during development stages.

The development machine available for the project is equipped with an Intel Xeon W-1250P @ 4.8 GHz with CPU and 32 GB of RAM. This CPU implements the PCIe Gen 3 Specification and is limited to a maximum of 16 lanes[17] which is sufficient to fulfil the requirements. As an interface, motherboard offers a PCIe slots with 16 and 4 lanes to be able to insert PCIe devices.

The operating system is Ubuntu 20.04.6 LTS, commonly used within Philips and as such provides stability and compatibility with a wide array of tools and programs which may

be helpful during development. As well as being compatible with the drivers described in Section 4.3.2.

4.3.1. Optical Interface

The optical interface component within the host system is responsible for converting optical signals into electrical signals and routing them to the standard PCIe slot typically found in most PCs. This task presents significant challenges, particularly due to the absence of an established standard by the PCI-SIG (PCI Special Interest Group) for such optical-to-electrical PCIe connections. As noted in the literature, PCI-SIG has yet to publish a formal standard that governs the integration of optical interfaces with PCIe[7]. However, the lack of a formal standard does not preclude the availability of off-the-shelf solutions that can be utilised for this purpose.

One product that addresses this need is the PCOA product line developed by Samtec [10]. This line includes the PCIe-Over-Fiber FireFly Adapter Card, which uses Samtec's FireFly optical transceivers to facilitate the conversion between optical and electrical signals. The PCOA adapter cards are designed to support PCIe Gen 4 x16 connections and can interface externally through one or two MTP12 or MTP24 connectors . Additionally, the PCOA cards can function as a PCIe switch, enabling the division of PCIe lanes into separate links, thus offering greater flexibility.

However, it is important to note that the PCOA product line is specifically tested and validated for use with FireFly transceivers at both ends of the connection. In the context of the current system design, where the device employs QSFP28 connectors, there is no guarantee that the PCOA adapter functions as expected. This potential incompatibility posed a risk to the project, which necessitated further investigation.

4.3.2. Drivers

In the implementation of the design showed in Figure 4.1, the selection of drivers plays a crucial role in ensuring seamless interaction between the hardware components and the software layers. Given the complexity and specificity of the design, the choice of drivers was influenced heavily by the integration of a particular IP core within the system. This IP core, namely the DMA/Bridge Subsystem for PCI Express (XDMA), provides a comprehensive solution that includes ready-to-use drivers designed to interface directly with the hardware.

The availability of these pre-packaged drivers simplifies the implementation process significantly. Since the XDMA IP core is already integrated into the design, utilising the provided drivers is the most logical and efficient choice. These drivers are tailored to work seamlessly with the XDMA architecture, ensuring optimal performance and reducing the potential for compatibility issues. Moreover, the drivers are well-supported and maintained, which ensures that any future updates or bug fixes can be easily incorporated into the system, enhancing both reliability and longevity. They are only supported on certain Linux distributions, on of which being Ubuntu 20.04.6 LTS. This compatibility detail also defines the host OS.

Given that the XDMA drivers are specifically designed to work with the hardware configuration of this project, and considering the lack of alternative drivers that could offer similar levels of integration and support, the decision to use these drivers is straightforward. This choice not only streamlines the implementation process but also ensures that the system's software and hardware components are fully aligned, resulting in a more robust and efficient overall design.

4.3.3. Application

In the implementation of the application, the drivers provided by the XDMA IP core offer an interface for interacting with the device and managing data transfers in both directions, Card-to-Host (C2H) and Host-to-Card (H2C). These drivers abstract much of the complexity involved in configuring and controlling the DMA engine, allowing for efficient data movement with minimal effort from the application layer.

Two approaches were employed in the application to handle data transfers: a simple approach and an optimised approach. In the simple approach, each transfer is initiated individually, with the Host PC directly managing the setup and completion of each DMA transaction. This first method is easy to implement and is effective for smaller or less time-sensitive transfers, where the overhead of setting up each transfer is not a significant enough to become a concern.

However, to enhance performance and reduce the Host PC's involvement, an optimised approach was also employed. In this method, the DMA descriptors, which define the source, destination, and size of each data transfer, are linked together in a chain. This chaining allows the DMA engine to process multiple transfers consecutively without requiring additional intervention from the Host PC for each individual transaction. By pre-configuring a series of transfers in this way, the Host PC can initiate the chain and then offload the bulk of the work to the DMA engine, significantly reducing the CPU overhead and increasing the overall throughput of the system

4.4. Challenges Encountered

In any complex engineering project, challenges are an inevitable part of the implementation process. These challenges often arise from the need to integrate multiple components, adapt to evolving requirements, and ensure that all elements of the design function together harmoniously.

In the context of this project, several key challenges were encountered during the implementation phase. These challenges provided valuable insights into areas where improvements and optimisations could be made as well as pitfalls to avoid. This section explores the various obstacles faced, the solutions employed to overcome them, and the lessons learned that can inform future projects.

4.4.1. PCIe IP Compatibility

In the initial steps of the project, the VCU108 development board was considered. It is built around the AMD Virtex UltraScale platform, offering robust support for PCIe Gen 3 with up to 8 lanes. The board is also equipped with a single QSFP28 optical transceiver cage, capable of handling 4 channels at 28 Gbps each, and features a PCIe edge connector directly integrated into the PCB. The initial plan was to use the VCU108 for both validating a copper-based version of the design and creating a versatile platform that could accommodate optical transmissions as well.

However, an unexpected compatibility issue arose with the specific version of the PCIe IP available for the platform. This version did not support the use of GTY transceivers for PCIe communication. GTY transceivers are high-performance, high-speed serial transceivers designed for applications requiring data rates of up to 32.75 Gbps[**<empty citation>**]. They are directly connected to the QSFP28 cage and are the only option to use it. The inability to utilize these transceivers for PCIe significantly limited the intended versatility of the VCU108 platform, necessitating a reevaluation of the design approach or the exploration of alternative solutions.

4.4.1.1. Link Establishment

Receiver Detect

During the integration of components an issue was encountered where the PCIe Link was not getting established. The debug capabilities of the IP were used to monitor the Link Training and Status State Machine (LTSSM). The LTSSM is a logic black in the Physical Layer of PCIe that configures and establishes the PCIe Link by negotiation the link width, speed and other settings to the link partner. The main mechanism for exchanging information during the link training phase is exchanging Ordered Sets. This sets are pre-determined fixed length packets that are easy to identify and are transmitted in all lanes in parallel. The LTSSM must go through a series of states until the L0 state is reached. This state is the default state once the link has been established and all transactions require the link to be in it.

The first state to happen after a reset is the Detect stage in which the electrical connection is monitored looking for a termination that indicates the presence of a link partner.

This mechanism does not work using optical interfaces and must be bypassed. This is achieved by modifying the generated Verilog files. It is required to bypass the detection in all lanes as failure to do so can put the LTSSM in a corrupted state.

Once this step is bypassed, the LTSSM will be stuck on the Polling stage sending Ordered Sets, failing to receive an answer and restarting the LTSSM.

Although it is not compliant with the PCIe specification, this change does not effect the majority of use cases because once a link partner is activated the LTSSM continues its progression normally.

Traditionally, PCIe devices include side band signals to also relay this information. This is not possible with the current implementation using QSFP28 interfaces as all available channels are used for data transfer, although a different implementation with the FireFly connector does allow up to 4 side band signals as each connector contains 12 channels.

Clocking

Another issue encountered was with the reference clock, which caused the LTSSM to fail during the stages following Polling. The reference clock is crucial for maintaining synchronisation between link partners, as it aids in recovering the clock embedded in the serial data transmission.

The problem was traced to a mismatch in the clock frequencies between the two link partners: the PCOA card was operating with a 100 MHz clock, while the FPGA was using a 125 MHz clock. The solution was straightforward and consisted on replacing the onboard crystal oscillator with one running at the same 100 MHz frequency as the PCOA, ensuring proper synchronisation and resolving the issue.

4.4.2. Adaptor Card Overheating

An additional challenge emerged after the link was successfully established. After several minutes of operation, the connection would unexpectedly drop, even in the absence of active data transfers.

The root cause of this issue was traced to the PCOA card, which was overheating due to insufficient cooling. To address this, an external fan was introduced to maintain an appropriate operating temperature for the card. This also required the removal of the side panel from the PC case. Once adequate cooling was provided, the system operated smoothly, and no further functional issues were encountered.

However, this solution is impractical for real production systems as it compromises the physical integrity of the setup and increases the risk of dust accumulation and hardware damage. Consequently, an alternative solution has to be devised before deployment of the product.

In this chapter, we detailed the implementation process of the PCIe over fiber communication system, covering the strategy employed, the selection of components for both the device and host, and the challenges encountered during development. The structured approach ensured that the chosen components met the system's current requirements while providing a foundation for future upgrades. Key decisions, such as the selection of the FPGA platform, PCIe core, memory, and optical interfaces, were guided by criteria including compatibility, scalability, and ease of use. The chapter also highlighted the practical challenges faced, such as PCIe IP compatibility and overheating, and the solutions implemented to overcome them. These insights contribute to a deeper understanding of the system's development and provide valuable lessons for future projects.

5

Experimental Results

This chapter explains the performance obtained out of the system design described in the previous chapter. The experimental setup is described in Section 5.1. This setup contains the implement design, as well as hardware and software tools to complete the system and analyse it. Section 5.2 presents the throughput results obtained after analysing the impact of descriptor allocation, amount of Direct Memory Access (DMA) channels, PCIe link width and generation as well as the impact that fiber length has with each configuration. Section 5.3 and Section 5.4 explores the effect that fiber length has on bit error rate and latency respectively. The last experiment in Section 5.5 provides the power consumption contribution of the fiber length, amount of DMA channels, PCIe link width and generation. To conclude the chapter, Section 5.6 contains a discussion about the results obtained.

5.1. Experimental Setup

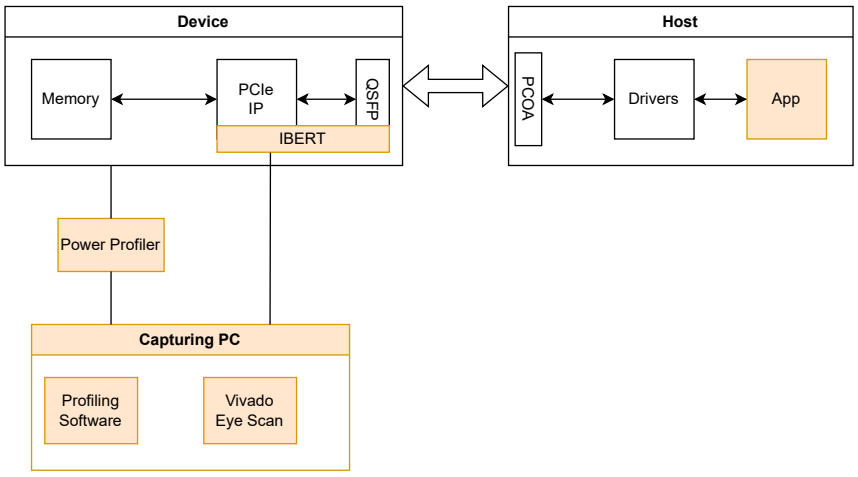


Figure 5.1: Component overview of the profiled system

The experimental setup used is shown in Figure 5.1. This setup consists of the high-speed data transfer system consisting of a device and a host, alongside the necessary tools to profile the system. These tools are marked with the colour orange on the diagram.

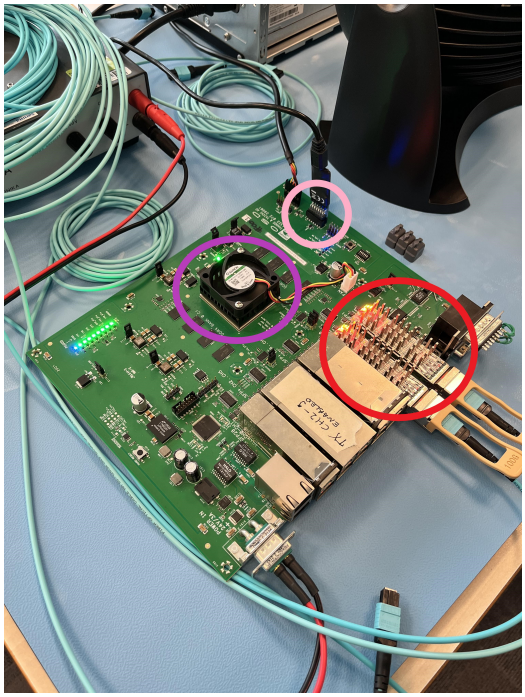


Figure 5.2: Optical Switch Dispatcher (OSD) board by Philips

5.1.1. Device Configuration

The device component is an FPGA board called Optical Switch Dispatcher (OSD) and was designed internally Philips pictured in Figure 5.2.

This board is centred around the UltraScale+ XCKU15P-FFVE1517-1-i [18] FPGA by Xilinx, circled in purple. It has two QSFP28 cages equipped with QSFP-100G-SR4 [16] optical transceivers by Cisco, circled in red. Each QSFP28 contains 4 differential transmission/reception lanes for a total of 8 lanes suitable for PCIe. The maximum speed they support is 4x25 Gbps which is sufficient for PCIe Gen3 that works at 8 GT/s per lane.

The firmware running on the FPGA has been described in the with more detail in the previous chapters. It is based around the XDMA IP[19] by Xilinx that is a wrapper around their base PCIe IP coupled with their DMA solutions which simplifies setting up efficient DMA transfers. On the particular FPGA chip used by this board the XDMA IP supports up to Gen3 x16.

This IP also allows the integration of a solution called In-system Bit Error Rate Tester that provides a convenient interface for performing eye-scans that can be used to identify problems on the physical level accessible through JTAG debug port, circled in pink. The XDMA IP is controlled using and embedded MicroBlaze [14] processor that controls the system. To simplify the experiments, no data generation components is on the device and the data is generated on the Host. This results in a loop back configuration that mainly focuses on the PCIe over fiber aspect of the design.

5.1.2. Host Configuration

The Host of the experimental setup, consists of multiple components, both software and hardware, used with a PC as a base platform.

The PC acting as a host in these experiments is equipped with an Intel Intel Xeon W-1250P CPU at 4.8GHz and 32 GB of RAM. Its motherboard supports up to PCIe Gen3 with 16

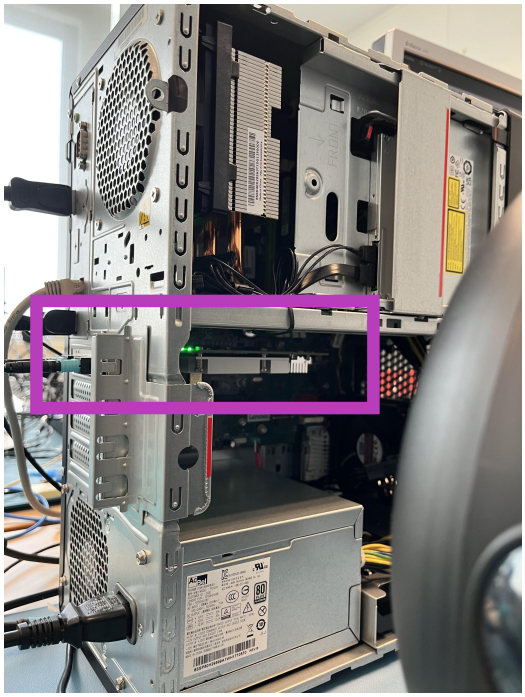


Figure 5.3: Host PC used for the experiments, with the PCOA card by Samtec into a PCIe slot.

lanes, with values of 256 and 128 bytes for the Maximum Payload Size (MPS) and Read Completion Boundary (RCB) respectively.

Inserted in the main PCIe slot has a PCOA-G4-D4-01 [10] by Samtec shown in Figure 5.3 inside the purple rectangle. This adaptor card, covered in more detail in an earlier chapter, converts the optical PCIe signals back into electric and forwards it transparently to the root port of the host using a PCB Edge connector. This particular model can work up to PCIe Gen4 x8. The interface with the device is a pair of MTP-12 connectors, each connecting to a QSFP28 transceiver on the device through a twelve core optical fiber.

From a software point of view two main components are present as well: the drivers and an application. These components are running on top of the operating system of the host, which is Linux Ubuntu 20.04.6 LTS. This OS is chosen because it has support for the XDMA Linux drivers [13] provided by Xilinx for the XDMA IP.

The application is different depending the experiment. For the throughput experiment, in Section 5.2, the application performs and times the transfers of data packages. Two approaches are used that are described in more detail in Sections 4.3.3. This same application is used with the reliability and power experiments, Section 5.3 and Section 5.5, for triggering the transfers but the measurements are taken with different tools explained in the next section.

For the latency experiment, Section 5.4, the software pcie-lat [20] by Andre Richter is used as the application. It performs a 32 bit read, the minimum, possible and measures the round trip time it takes to complete. It provides an easy interface for performing the test a statistically significant amount of times.

5.1.3. Fiber Optic Setup

The fiber used for this experiments, pictured in Figure 5.4 are FOPC-01-01-X-12-01 optical MTP patch cables, circled in purple. Their lengths are chosen to be 3m, 10m and 100m as these represent a typical range of distances for the final solution. For cost reasons there is only one 100m cable allowing for a maximum of an x4 link with it. For 3m and

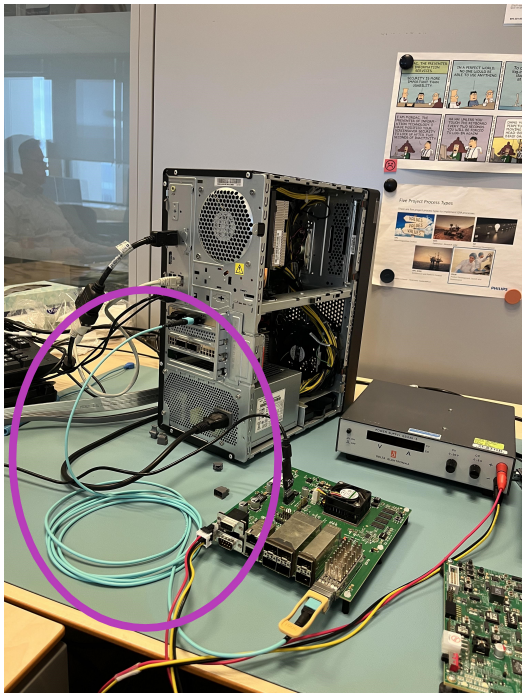


Figure 5.4: Experiment setup with a single 3m fiber.

10m it is possibly to reach a width of x8. The manufacturer suggests using OM4 fiber for commercial solutions but states that there should not be any problems using OM3 for prototyping stages.

5.1.4. Profiler Tools

Some of the experiments require external tools, outside the host and device, to capture the desired data. Such is the case for the power and reliability experiments.

For the reliability experiment, Section 5.3, an external PC is used for connecting to the JTAG interface of the FPGA and control the In-system Bit Error Rate component, which allows for easy eye-scan capture using Vivado [21] Hardware manager.

To the power experiment, Section 5.5, the device is powered using a Otii Arc Pro by Qoitech[22]. This device can source voltage and current while simultaneously measuring them. It is limited to 5V/5A which would be insufficient for the 24V/3A needed by the FPGA. This limitation is overcome by connecting a 19V power supply in series, which allows to get the power consumption by measuring the current.

5.2. Study of the system: Throughput

The first experiment will test the system's throughput. This experiment's goal is to understand the factors that effect the throughput in a PCIe over fiber system. This is done to be able to design and develop future systems with a deep understanding of the limitation factors that may appear. In this case, various parameters are systematically adjusted and compared to assess their impact on performance. These parameters, shown in Figure 5.5, can be divided into multiple categories depending on where they are most relevant. In the drivers it is the descriptor allocation strategy, in the IP it is the amount of Direct Memory Access (DMA) channels, the PCIe Link itself can be impacted by link width and generation as well as the impact that fiber length has with each configuration and, finally the Traffic direction can also impact the results. After each individual test, the

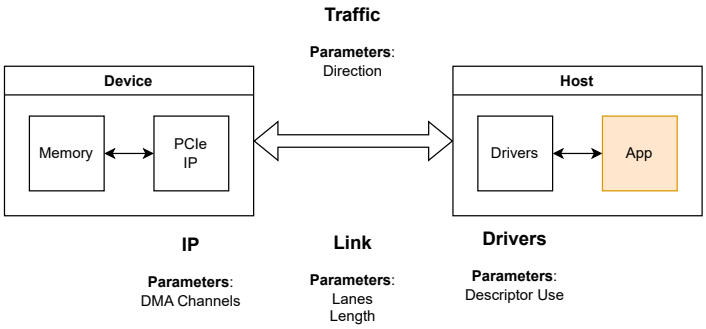


Figure 5.5: Experiment setup for the throughput test.

results are analysed and then, with that information, further tests are performed.

5.2.1. Implementation Details

To conduct this experiment, the setup described in Section 5.1 is utilised. On the Host PC, the XDMA drivers are loaded, followed by a startup check to ensure proper linkage and basic functionality.

The application used for this experiment is a custom C program designed to interface with the driver, facilitating the transmission and reception of data. The program also measures the time taken for these transfers, allowing the calculation of throughput. To reduce variability, transfers are repeated multiple times. The experiment includes both Card-to-Host (C2H) and Host-to-Card (H2C) directions, with varying packet sizes, each a power of two bytes. This approach simplifies the process and provides comprehensive data across a wide range of sizes. The contents of each packet are randomly generated for every transfer. Given the nature of the PCIe protocol, which includes robust error detection and correction mechanisms, data received above the Transaction Layer Protocol is guaranteed to be accurate. Detected errors are either corrected automatically or, if uncorrectable, trigger a retransmission request, ensuring data integrity.

The tests are conducted using different fiber lengths (3m, 10m, and 100m) except where not feasible due to resource limitations, such as the case with an 8-lane link over 100m, which requires a second fiber. To simplify the experiment and maintain focus on the PCIe optical link, a loop back configuration is employed, where data is always generated on the Host side.

5.2.2. Simple Transfers Throughput

The first experiment uses read and write calls to the driver to perform the transfers. It is immediately apparent in Figure 5.6 that the performance is not reaching the expected throughput associated with the 3rd generation of PCIe, most likely due to OS overhead.

The experiment does reveal interesting patterns in throughput. For both the 3m and 10m fibers, the throughput remains quite similar, particularly in Host to Card (H2C) transmissions compared to Card to Host (C2H). This consistency suggests that the relatively short fiber lengths do not significantly impact performance, and the observed differences in throughput are likely due to inherent characteristics of the data flow rather than the fiber length itself. However, when using a 100m fiber, the bandwidth drops significantly. H2C throughput is shown to be higher than C2H and is caused by the nature of PCIe, which requires more overhead for read than write operations because on is a Non-Posted command and the other is Posted. Performance in both cases only begins to

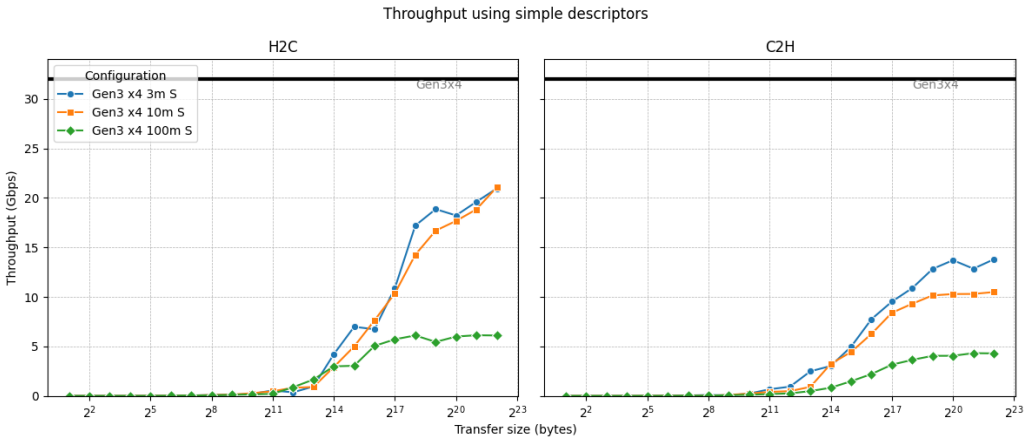


Figure 5.6: Throughput obtained using simple descriptors

improve noticeably around a packet size of 2^{12}, with a more pronounced increase in H2C transmissions, highlighting its relative efficiency even under conditions of greater latency.

These results are significantly lower than the theoretical maximum supported by an Gen3 x4 which is defined in the specification as

$$8\ GT/s * 4\ lanes = 32\ GT/s$$

a total of 32GT/s. Also the non-linearity and apparent randomness in the noise of the results suggests that other components in the transmission are affecting the performance, in this case the OS. In order to solve this, a more optimised transfer system is used in the next experiment.

5.2.3. Queued Descriptors Throughput

With the results from the previous test a more sophisticated software approach is used. The idea being that reusing the OS descriptors on a circular basis so the allocation time is only paid once, as per the recommendations found in Xilinx Answer documentation[23].

This strategy consists on reusing the same descriptors allocated once for the OS in a circular manner, this way it is not needed to ask, and thus waste time allocating OS resources.

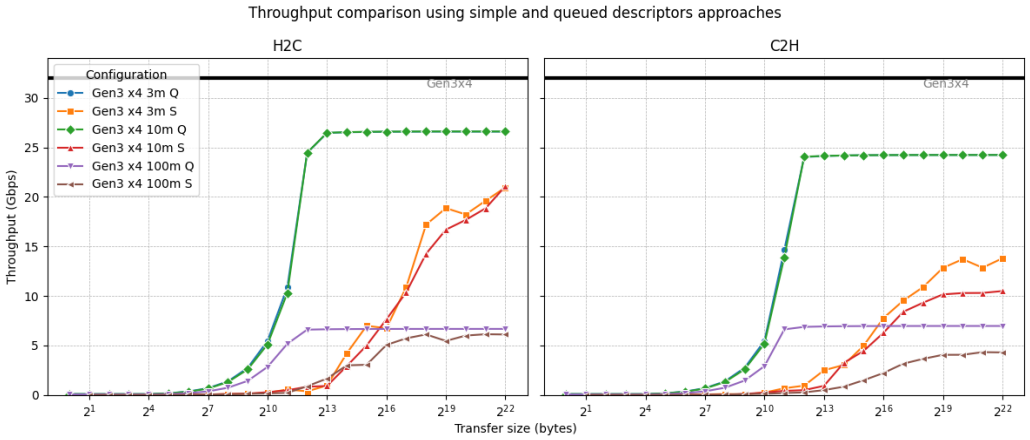


Figure 5.7: Throughput comparison between simple and queued descriptors

Following this new strategy, the results are significantly different as shown in Figure 5.7.

The phenomenon where H2C has better performance compared to C2H continues to happen. It is also relevant to note that at 100m the throughput is much lower and is closer in both directions compared to 3m and 10m.

To fully grasp the performance at 3m and 10m, particularly the inefficiencies in the protocol, it is crucial to understand the constraints imposed by the system settings. The performance, especially in Host to Card (H2C) transfers, is limited by the Maximum Payload Size (MPS) setting, which is configured at 256 bytes, even though the card can handle up to 1024-byte payloads because the Host system does not support higher values.

This setting causes payloads to be fragmented, effectively capping the maximum achievable throughput. A more accurate theoretical maximum can be calculated by determining the packet efficiency using the following formula

$$Efficency = \frac{MPS}{MPS + OVERHEAD}$$

With an overhead of 22 bytes in a Gen3 32-bit addressing mode, the packet efficiency is 92.09%. When factoring in the 128b/130b encoding rates, this efficiency further decreases to 90.67%. For a 32 GT/s system, this translates to a maximum theoretical bandwidth of 29.01 Gbps. However, the observed throughput in the experiment was 26.56 Gbps, indicating additional factors impacting performance. Among these factors are the bandwidth used for ACK and NACK signals necessary for flow control, and special frames like Skip Ordered Sets introduced by the PCIe PHY for link management, which further reduce the available bandwidth.

In the case of Card to Host (C2H) transfers, the communication is less efficient because the maximum payload size in Read operations is governed not by the MPS setting but by the Read Completion Boundary (RCB). The RCB is used by the Root Port of a system to allocate resources across multiple links without overwhelming the host. In this experiment, the RCB was set to 128 bytes, which is lower than the MPS and requires more packets and acknowledgements. To alleviate the acknowledgement burden, PCIe allows multiple packets to be acknowledged with a single ACK. The efficiency calculation is modified as

$$Efficency = \frac{RCB}{RCB + OVERHEAD}$$

In this case, the result for C2H transfers is 85.53%, which, when combined with encoding overhead, reduces to 84.02%. This corresponds to an effective bandwidth of 26.88 Gbps, which, although still slightly higher, is much closer to the observed experimental results. This discrepancy is likely due to the calculation not accounting for all overhead factors.

And additional detail to note is that the performance only reaches the maximum with frames higher than 2^8 (256) bytes at 2^{13} (8,192) and 2^{12} (4,096) for H2C and C2H respectively. It is hypothesised to be due to OS overhead, even if the PCIe protocol splits the payload into multiple packets, a system call is needed to start the transfer at the OS level. It is the case that at lower transfer sizes these calls happen more frequently and may affect the system's performance.

It can also be observed that the throughput is much lower with the 100m fiber compared to 3m and 10m. Plausible causes for this phenomenon are the reliability issues and increased latency found with the longer fiber. Both of these hypothesis are tested in Experiments 5.3.2 and 5.3.3 respectively.

5.2.4. Throughput by DMA Channels

By utilising more DMA channels, the system is able to mitigate latency more effectively, resulting in a notable performance increase when using a 100m fiber as seen in Figure 5.8. This is because the IP starts servicing the transfer the moment any completion data arrives, and thus reducing the processing latency. The additional channels allow for parallel data

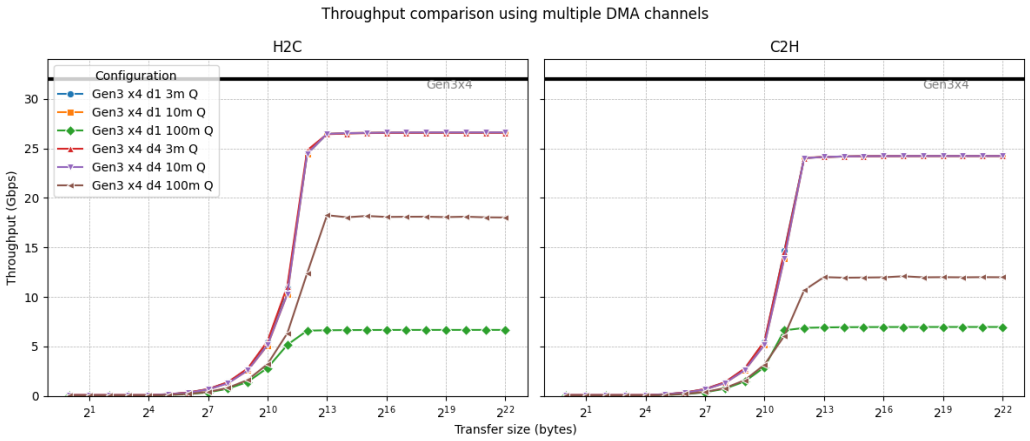


Figure 5.8: System's throughput using multiple DMA channels

processing, which helps to overcome the latency introduced by the longer fiber length, thereby improving overall throughput. However, with shorter fibers, such as 3m and 10m, the latency is already minimal, so the performance remains relatively unchanged regardless of the number of DMA channels used. In these cases, the additional channels do not provide a significant advantage, as the primary bottleneck with longer runs, latency, has no impact.

5.2.5. Throughput by PCIe Link Width

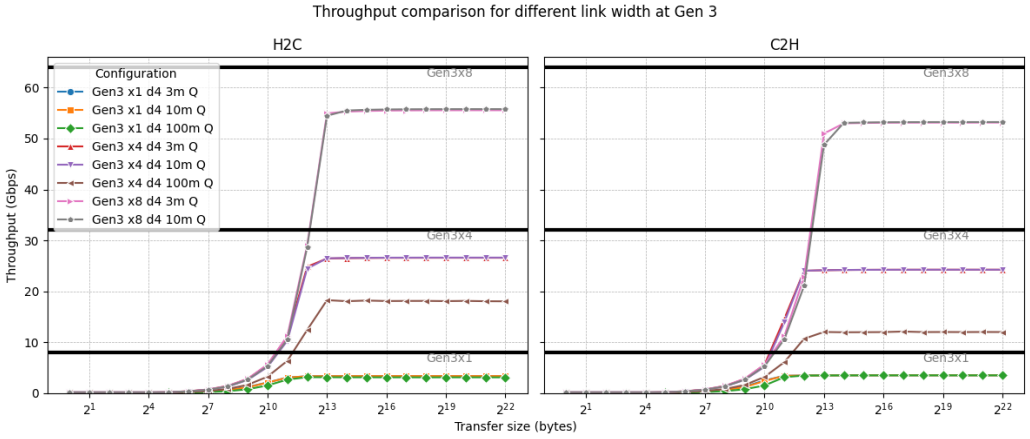


Figure 5.9: Throughput using different PCIe Link widths

This experiment aims to explore the impact of adding more lanes on the system's throughput. The setup mirrors the previous experiment, utilising a Gen3 link with four DMA channels and the queued descriptors approach. Throughput is measured across three fiber lengths—3m, 10m, and 100m—using 1, 4, and 8 lanes. The only exception is the 8-lane configuration at 100m, which was not feasible due to the high cost of the fiber, limiting the experiment to a maximum of 4 lanes for this length.

As with previous experiments, the differences between 3m and 10m fibers are negligible, with both achieving similar performance under all conditions. The results, Figure 5.9, for both directions in the x4 and x8 configurations closely align with the efficiency values calculated earlier: 90.07% for H2C and 84.02% for C2H. These efficiency rates correspond to throughput's of 29.01 Gbps and 26.89 Gbps for Gen3x4 (32 GT/s), and 58.02 Gbps and 53.77 Gbps for Gen3x8. As shown in Figure 5.9, the obtained throughput is very close to

these theoretical values, though slightly lower due to the exclusion of Link Management traffic in the efficiency calculations, which also consumes bandwidth.

An interesting anomaly occurs with the single-lane configuration, where the throughput is approximately 50% of the expected value. This pattern is consistent across all fiber lengths, including the 100m fiber, which previously showed degraded performance in other scenarios. This suggests the presence of a different bottleneck when only one lane is used. It is hypothesized that the issue arises from Link Management, as its packets compete with TLP packets for bandwidth. Without additional lanes to help mitigate the impact, this competition likely causes increased downtime.

5.3. Study of the System: Link Reliability

The second experiment will test the system's link reliability. This experiment aims to analyse the reliability of the PCIe over fiber link. It is already known that data correctness is guaranteed once a package is received above the Transaction Layer, so this experiment is focused on the effect that fiber length has on errors on the physical layer.

5.3.1. Implementation Details

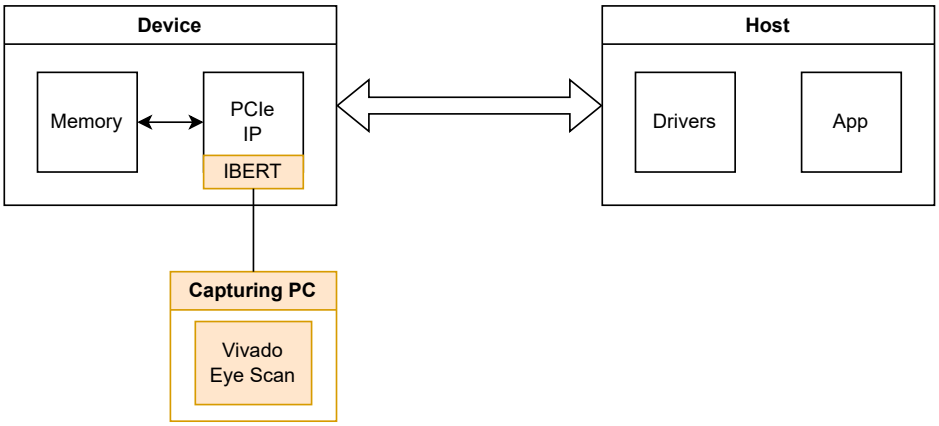


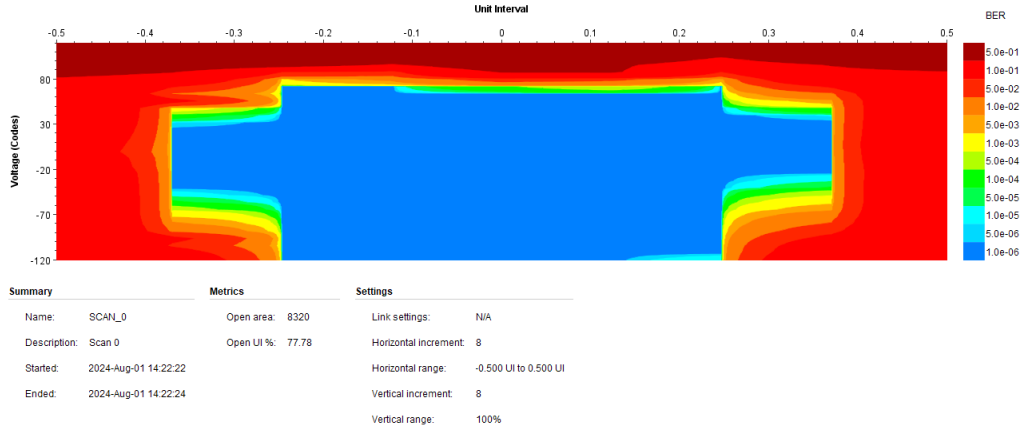
Figure 5.10: Experiment setup for the link reliability test

In order to perform this experiment, the setup described in Section 5.1 is used. On the Host PC, the XDMA drivers are loaded up and start up check is performed to ensure correct linkage. An extra PC, highlited here in Figure 5.10, is used to access the JTAG port of the board and thus control the IBERT [24] (Integrated Bit Error Tester) component. The IBERT tool embedded in the XDMA IP by Xilinx, is used to measure the Open Area of the eye as one of the indicators of reliability. It is a powerful tool used for testing and validating high-speed serial I/O, such as PCIe links. IBERT allows users to configure, monitor, and analyse the performance of transceivers on Xilinx FPGAs. It shows real-time metrics, including bit error rate, eye diagrams, and other signal integrity parameters.

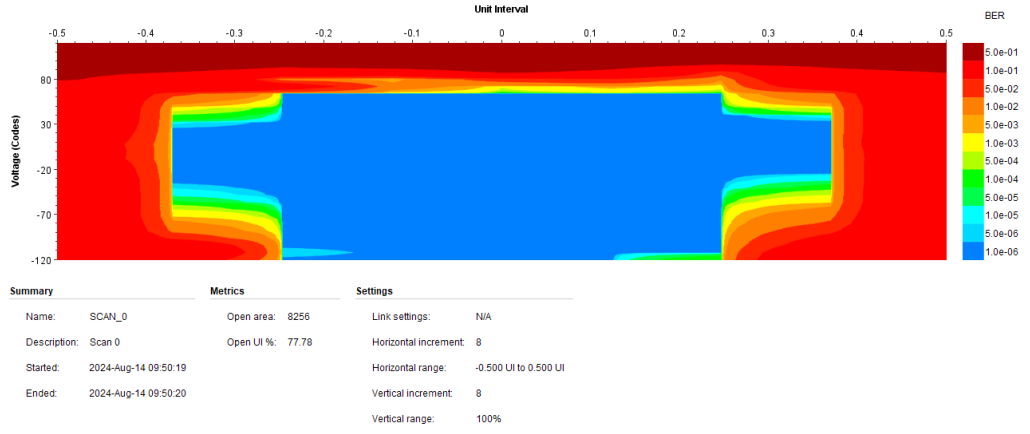
The eye scan is performed while a continuous transfer using queued descriptors, four lanes, four DMA channels and transfer size of 8,192 is being carried on both directions.

5.3.2. Results and Analysis

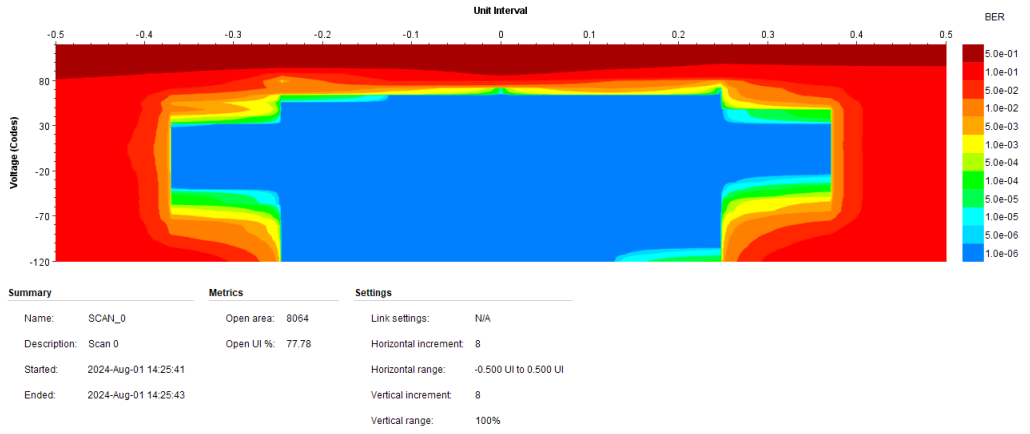
The open eye area measurement is a unit, the count of all bins, defined by the unit interval and voltage, that has had smaller amount of measurements than the dwell parameter, in this case 1×10^{-6}.



(a) 3m Optical Fiber



(b) 10m Optical Fiber



(c) 100m Optical Fiber

Figure 5.11: Example eye scan performed using IBERT with 3m (a), 10m (b), and 100m (c) optical fiber lengths

Open Area	Avg	Std
3m	8320.0 units	79.82 units
10m	8211.2 units	70.4 units
100m	8147.2 units	85.6 units

Table 5.1: Average and deviation for the Eye scan performed using IBERT at different fiber lengths

Table 5.1 provides the results obtained after running the experiment 10 times per fiber length. The difference in area observed between the different lengths is less than 2.1% in all cases. The measurements are even more similar if the standard deviation is also taken into consideration.

As an illustrative example Figures 5.11a, 5.11b and 5.11c showcase eyes results obtained for all three lengths of fiber. The darkest hue of blue represents the open are of the eye and doesn't present significant difference between them.

Such a small difference in area indicates that error rate is not the cause for reduced performance on the 100m fiber.

5.4. Study of the System: Latency

The next experiment will test the system's latency. This experiment aims to analyse the latency resulting from using different fiber lengths and the effects it may pose on the resulting throughput.

5.4.1. Implementation Details

In order to perform this experiment, the setup described in Section 5.1 is used. On the Host PC, the XDMA drivers are loaded up and start up check is performed to ensure correct linkage. Latency is measured using an open source tool, `pcie-lat` [20]. The tool writes a 32 bit word from the host to the FPGA and reads it back, measuring the time taken for the round trip.

5.4.2. Results and Analysis

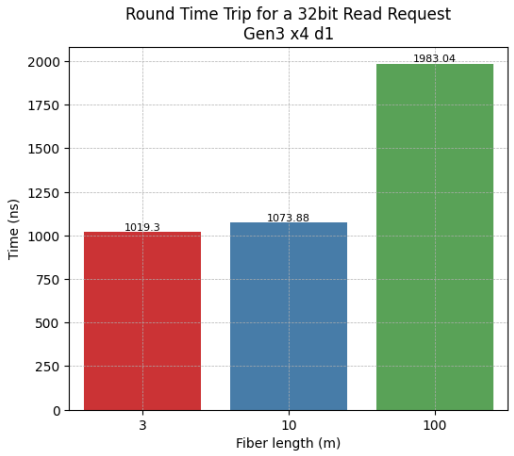


Figure 5.12: Latency comparison for different cable lengths

The results of the latency test, in Figure 5.12, show that the latency is quite similar for the 3m and 10m fibers, but it doubles when using the 100m fiber. All measurements include the time for processing the request and should be constant across all different lengths.

Considering the fact that the speed of light in a vacuum is approximately 3×10^8 m/s, it is necessary to compensate for the change of medium. The speed of light in optical fiber was measured to be approximately 2×10^8 m/s in an experiment by the Kansas State University[25]. To calculate the one way travel time the following formula is used:

$$t_{travel} = \frac{d}{v}$$

where d is the fiber length and v is the speed of light in optical fiber. This gives a total time travel in fiber per direction to be 15ns, 50ns and 500ns with 3m, 10m and 100m. The total round time trip also includes the processing time which should be the similar across all fiber lengths. Given that:

$$RTT = 2 * t_{travel} + t_{proc}$$

then:

$$t_{proc} = RTT - 2 * t_{travel}$$

This formula produces total processing times of 989.30 ns, 973.88 ns, and 983.04 ns, respectively. The difference between them can be attributed to the OS overhead as 15 ns are a 72 clock cycles of the CPU running at 4.8 GHz. It is also important to note that the processing time includes both Host and Device processing times.

These calculations reinforce the consistency of the experimental results, confirming that latency is a key factor in the similar performance observed with 3m and 10m fibers, while also explaining the significant reduction in performance with the 100m fiber.

In the PCIe Specification [6] there is a concept defined as Ack Latency Limit that defines the maximum latency in Symbol times to not overwhelm the retry buffer and limit the performance. In Table 3.10 of the specification, here in this document as Figure 5.2, the upper limit of the ACK Latency for maximum throughput is shown. The unit is Symbol times defined as the time it takes to transmit a byte. This means that in Gen 1 and Gen 2 the Symbol Time is the time it takes to transmit 10 units (T), due to the 8b/10b encoding, while in Gen 3 and onwards its approximated to only 8 units (T), due to the usage of 128b/130b encoding.

Thus, the Symbol Time for Gen 3 is 1 ns

$$SymbolTime = \frac{8\ units/symbol}{8 \times 10^9\ units/second} = 1ns$$

Table 3-10: Ack Transmission Latency Limit and AckFactor for 8.0 GT/s Mode Operation by Link Width and Max Payload (Symbol Times)

		Link Operating Width						
		x1	x2	x4	x8	x12	x16	x32
Max_Payload_Size (bytes)	128	333 AF = 1.4	224 AF = 1.4	169 AF = 1.4	163 AF = 2.5	154 AF = 3.0	144 AF = 3.0	129 AF = 3.0
	256	512 AF = 1.4	313 AF = 1.4	214 AF = 1.4	203 AF = 2.5	186 AF = 3.0	168 AF = 3.0	141 AF = 3.0
	512	655 AF = 1.0	385 AF = 1.0	250 AF = 1.0	182 AF = 1.0	205 AF = 2.0	182 AF = 2.0	148 AF = 2.0
	1024	1167 AF = 1.0	641 AF = 1.0	378 AF = 1.0	246 AF = 1.0	290 AF = 2.0	246 AF = 2.0	180 AF = 2.0
	2048	2191 AF = 1.0	1153 AF = 1.0	634 AF = 1.0	374 AF = 1.0	461 AF = 2.0	374 AF = 2.0	244 AF = 2.0
	4096	4239 AF = 1.0	2177 AF = 1.0	1146 AF = 1.0	630 AF = 1.0	802 AF = 2.0	630 AF = 2.0	372 AF = 2.0

Figure 5.13: Ack Transmission Latency Limit and AckFactor for 8.0 GT/s Mode Operation by Link Width and Max Payload (Symbol Times). Table extracted from the PCIe Specification 4.0[6]

Fiber length	Fiber latency (2 * d / v)	Gen3x4 MPS=128 (C2H) <169 ns	Gen3x4 MPS=256 (H2C) <214 ns
3 m	30 ns	Yes	Yes
10 m	100 ns	Yes	Yes
100 m	1000 ns	No	No

Table 5.2: Comparison between fiber latency times and Ack Transmission Latency limit for experiment setup

When compared to the fiber latency calculated for each length it becomes clear that this value is exceeded with 100m fiber.

The experiment confirms that fiber length imposes strict limits on throughput due to the added delay. For the PC used in this experiment, with a Maximum Payload Size (MPS) of 256 bytes and a Gen3 x4 link, the upper limit for fiber length before throughput begins to degrade is approximately 21.4 meters. This calculation assumes a processing time of zero for simplicity to provide an upper bound limit:

$$Maximum\ Fiber\ length = \frac{214\ ns}{2} * 2 \times 10^8 = 21.4m$$

These results suggest that it might be possible to maintain maximum throughput over a 100m fiber with a host capable of supporting an MPS of 4096 bytes. However, this feature is not commonly supported in most systems. It is also not possible to achieve full throughput going beyond 4 lanes at that fiber length as all configurations have Ack Latency limits lower than 1000 ns, latency of the 100m fiber.

5.5. Study of the System: Power

The power experiment is meant to analyse the factors that affect the power consumption of the device. This factors include fiber length, link width, link speed and number of DMA channels. It is also important to characterise the expected consumption on different scenarios such as: idle, card to host (C2H) and host to card (H2C) transactions. By gathering this information it is possible to make estimations of the consumption based on predicted data transfer patterns.

5.5.1. Implementation Details

This experiment requires the use of power profiler connected to the device as shown in Figure 5.14. The profiler used is an Otii Arc Pro by Qoitech[22] and is controlled using an external PC. This tool can source voltage and current while simultaneously measuring them. It is limited to 5V/5A which would be insufficient for the 24V/3A needed by the FPGA. This limitation is overcome by connecting a 19V power supply in series, which allows to get the power consumption by measuring the current.

For each configuration of the system, the power consumption is sampled at frequency of 4,000 Hz. For each of the scenarios (idle, C2H and H2C) it is done over a period of one minute. The final measurement is then obtained from the average during that time.

To ensure the maximum number of transitions between logic levels 0 and 1, and thus forcing the system to the most possible work the payload used in Gen 1 and Gen 2 consists of the byte: "101 10101" or 0xAA. This byte is encoded using the 8b/10b scheme as as the Data symbol D21.5 which results in becoming "10101 01010" in both running disparity (RD+ and RD-) cases. Based on the throughput results, a frame size of 4,194,304 bytes (2^{22}) is used for maximum system utilisation.

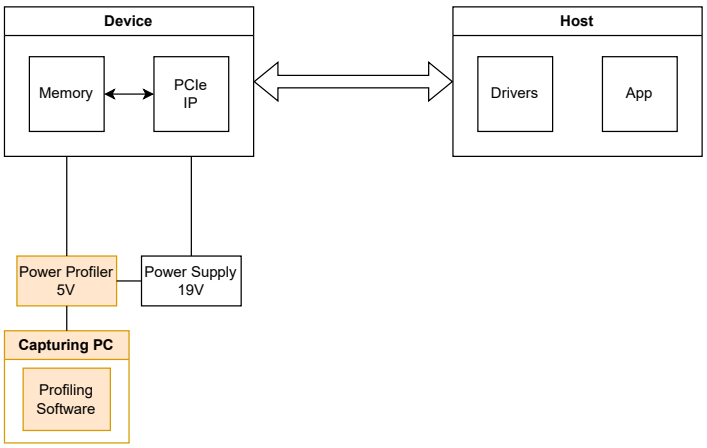


Figure 5.14: Average power consumption of a Gen3 x4 link at different fiber lengths

For Gen3, the encoding scheme was changed to 128b/130b which does not guarantee DC equilibrium. For that reason an extra step is applied that consists of scrambling the data with a LFSR (Linear-Feedback Shift Register) which provides the equilibrium statistically instead. For this reason it is not feasible to provide a worst-case payload, hence the payload employed with Gen 1 and 2 is used with Gen 3 as well.

The experiment is performed in the three scenarios: idle, C2H and H2C and is repeated methodically changing one parameter at a time. First by using different fiber lengths (3m, 10m and 100m), second by changing the width of the link (x1, x4 and x8), then by chaning the amount of DMA channels (1 and 4) and finally by chaning the PCIe Generation (Gen1, Gen2 and Gen3).

The following parameters are analysed on this test:

- Fiber length at 3 and 100m;
- Link width at 1, 2 and 4 lanes;
- Number of DMA channels at 1 and 4 channels;
- Link speed at Gen1, Gen2 and Gen3 speeds.

5.5.2. Fiber Length Impact in Power Consumption

The first experiment looks at understanding the effect of different fiber lengths introduce in the power consumption. The configuration for this experiment is PCIe Gen3 with 4 lanes and 4 DMA channels. The only parameter changed is the fiber length. The frame size chosen is 2^{22} bytes for maximum throughput and the payload contents are bytes 0xAA but the actual content of the payload is largely irrelevant due to the scramble feature of PCIe 3.0.

Figure 5.15 presents the results of the experiment. It can be observed that power consumption during idle remains consistent across both fiber lengths tested. The first significant difference arises during data transmissions, where power consumption with a 100m fiber is lower in both directions compared to the 3m fiber. This is attributed to the reduced throughput, as noted in Experiment 5.4, where the increased latency with the 100m fiber results in lower throughput, consequently leading to reduced power consumption since less data is being transmitted. Another key observation is that Host to Card (H2C) transmissions consume more power than Card to Host (C2H) transmissions. This occurs with both fiber lengths and is also linked to the reduced throughput during C2H transmissions as less power is used for data movement.

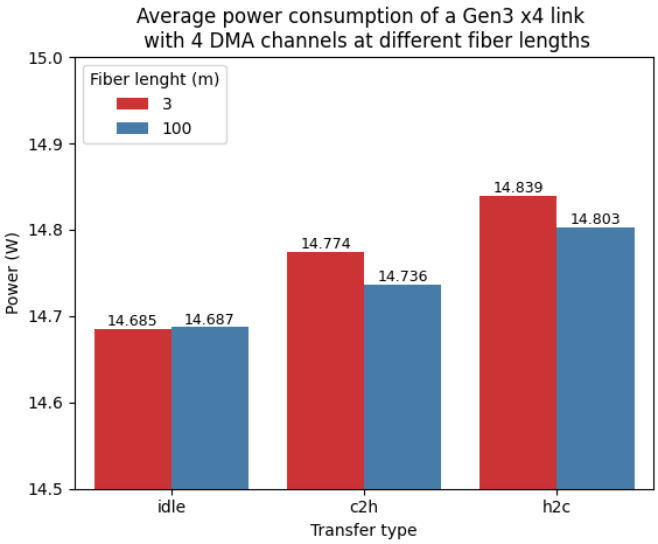


Figure 5.15: Average power consumption of a Gen3 x4 link with 4 DMA channels at different fiber lengths

5.5.3. Power Usage by PCIe Link Width

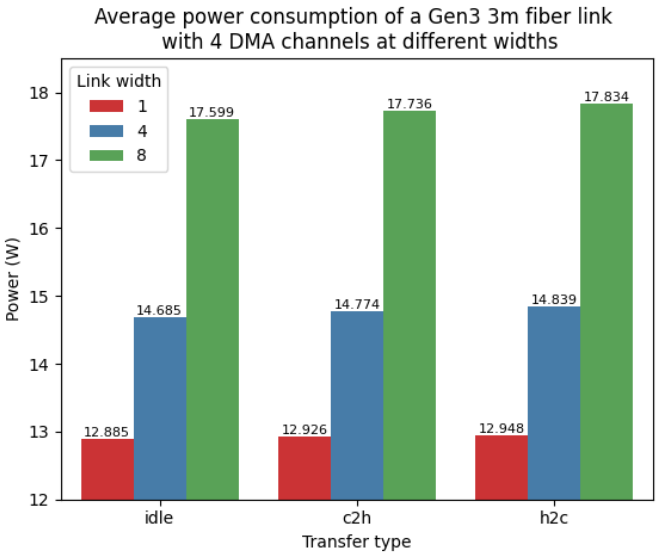


Figure 5.16: Average power consumption of a Gen3 3m fiber link with 4 DMA channels at different widths

The second experiment is meant for understanding how much a lane contributes to the power consumption. The configuration for this experiment is PCIe Gen3 with a 3m fiber and 4 DMA channels. The only parameter changed is the link length. The frame size chosen is 2^{22} bytes for maximum throughput and the payload contents are bytes 0xAA but the actual content of the payload is largely irrelevant due to the scramble feature of PCIe 3.0.

Figure 5.16 contains the results of the experiment. It can be observed that all types of scenarios tend to scale similarly with the number of lanes. These findings are consistent with the earlier throughput results, where performance also increased proportionally with the number of lanes. It is important to highlight that idle consumption scales as well. While this may seem counter-intuitive, there is a straightforward explanation. When no transactions occur in PCIe, the link enters a state known as Logical Idle, during which it

continuously transmits and receives a special Logic Idle data symbol to maintain clock synchronisation between the link partners. Although the PCIe specifications describe low power modes, optical PCIe research recommends not enabling them, as the Electrical Idle is known to cause issues by saturating optical transceivers.

5.5.4. Power Consumption by DMA Channels

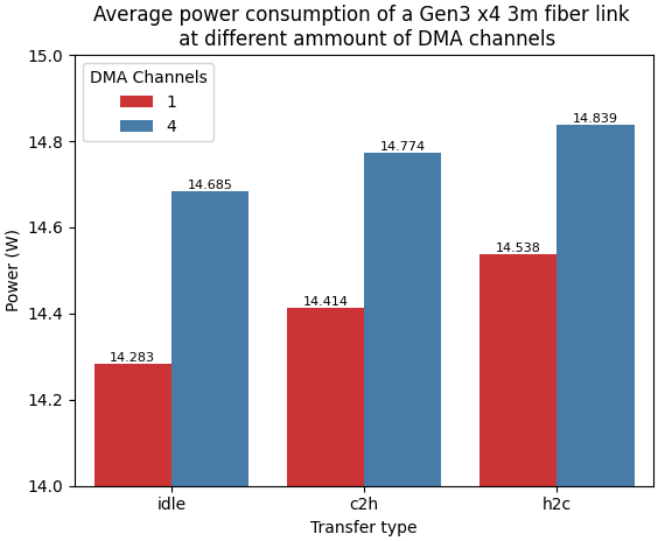


Figure 5.17: Average power consumption of a Gen3 x4 3m fiber link at different at different amounts of DMA channels

The third experiment focuses on the impact having multiple DMA channels on the FPGA design causes on the overall power consumption. The configuration for this experiment is PCIe Gen3 x4 link with a 3m fiber. The only parameter changed is the number of DMA Channels. Figure 5.17 contains the results of the experiments.

The key takeaway from the results is that, despite no difference in throughput, as shown in Figure 5.8, overall power consumption is significantly higher with four channels compared to just one. Specifically, using four channels results in a 2.81% increase in power consumption during idle, a 2.50% increase during Card-to-Host transmissions, and a 2.07% increase during Host-to-Card transmissions.

It's important to note that performance remains comparable only when short fiber lengths are used, as illustrated in Figure 5.8. However, with longer fibers, such as 100 meters, throughput is significantly higher with multiple DMA channels. The observed throughput increases of 92% in Card-to-Host transmissions and 175% in Host-to-Card transmissions suggest that the efficiency gain is more pronounced in Host-to-Card transmissions.

5.5.5. Power Performance Across PCIe Generations

The last experiment aims to showcase the effect the PCIe Gen has on the power consumption. The configuration for this experiment is link with 4 lanes with a 3m fiber and 4 DMA channels. The only parameter changed is the s. The frame size chosen is 2^{22} bytes for maximum throughput and the payload contents are bytes 0xAA for but the actual content of the payload is largely irrelevant for Gen3 due to the scramble feature of PCIe 3.0 but not for Gen 1 and Gen 2.

The results obtained by adjusting the generation speed reveal an interesting contradiction. While the power consumption increases by 4.10% when moving from Gen 1 to Gen 2, and by 4.95% from Gen 2 to Gen 3, these increases are disproportionate to the changes in speed.

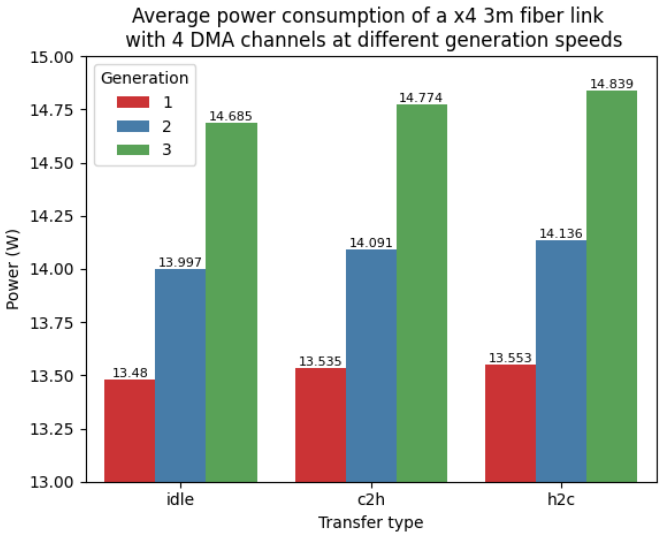


Figure 5.18: Average power consumption of a Gen3 x4 link at different fiber gens

Specifically, the speed doubles (100% increase) from Gen 1 to Gen 2, yet the power increase is relatively modest. Conversely, the speed only increases by 60% from Gen 2 to Gen 3, but the power consumption increase is slightly higher. This discrepancy can be attributed to the fact that Gen 3 utilises a much more efficient encoding system, which allows it to achieve higher data rates with only a marginal increase in power consumption. The efficiency of the Gen 3 encoding system offsets the expected power increase, explaining why the power scaling does not directly correspond to the speed scaling.

5.6. Discussion

The experimental results obtained from the series of tests conducted on the PCIe over fiber system using FPGA technology offer several important insights into the performance, reliability, and power efficiency of the system. This discussion synthesises the findings from each experiment, assesses their implications, and identifies areas for improvement.

The throughput experiments revealed several key factors influencing system performance, including descriptor allocation, the number of DMA channels, PCIe link width, and fiber length. The use of simple descriptors led to sub optimal performance, primarily due to operating system overhead. However, by employing queued descriptors, a significant improvement in throughput was observed, aligning more closely with the theoretical maximums. This underscores the importance of optimising software strategies alongside hardware configurations to achieve maximum system performance.

The analysis also highlighted the impact of fiber length on throughput, particularly with longer fibers such as the 100m cable. The reduced throughput observed with longer fibers is likely due to increased latency and potential reliability issues, as suggested by the follow-up experiments. Interestingly, the use of multiple DMA channels effectively mitigated the latency introduced by longer fibers, improving throughput. This finding suggests that parallel processing and efficient resource management are crucial for maintaining high performance over extended distances.

The results related to PCIe link width further confirmed that increasing the number of lanes directly improves throughput, as expected. However, the performance anomaly observed with single-lane configurations warrants further investigation. Understanding the root cause of this discrepancy could lead to more efficient designs, particularly for systems constrained to fewer lanes.

The reliability experiment, which measured the open area of the eye diagram across different fiber lengths, indicated that error rates remained low and consistent across all tested lengths. This suggests that the physical integrity of the optical link is well-maintained, even at longer distances, and that the reduced throughput at 100m is not due to increased errors at the physical layer.

The latency experiment, however, confirmed that fiber length significantly affects system latency, with the 100m fiber doubling the latency compared to the shorter lengths. The increased latency observed with the 100m fiber corresponds with a reduction in throughput, as the system's ability to process and acknowledge transactions in a timely manner is compromised. The comparison with the PCIe specification's Ack Latency Limit further supports the finding that latency becomes a critical bottleneck at longer distances, impacting the system's ability to maintain maximum throughput.

The power experiments provided valuable insights into the factors contributing to the system's power consumption. As expected, power consumption scaled with both the number of active lanes and the number of DMA channels, although the increase in power was relatively modest compared to the gains in throughput achieved through these configurations. This suggests that the system's design is relatively power-efficient, particularly when scaling up the number of lanes or channels.

Interestingly, the power consumption was lower with the 100m fiber compared to the 3m fiber during active data transfers. This counter intuitive result can be explained by the reduced throughput observed with the longer fiber, which leads to less overall data being processed and, consequently, lower power usage. However, this reduction in power comes at the cost of performance, highlighting the trade-offs that must be considered when designing systems for extended distances.

The analysis of power consumption across different PCIe generations revealed that while higher generations consume more power, the increase is not directly proportional to the increase in speed. The more efficient encoding used in Gen3 partially offsets the expected rise in power consumption, suggesting that future designs could benefit from adopting newer PCIe generations to balance performance and power efficiency.

The experimental results demonstrate that the PCIe over fiber system is capable of achieving high-performance data transfer rates, but also highlight the importance of careful system design and optimisation. The findings underscore the need for:

- **Optimised Software Strategies:** Proper descriptor management and the use of queued descriptors can significantly enhance throughput, particularly in systems where OS overhead is a limiting factor;
- **Latency Management:** While the physical layer remains robust across distances, managing latency becomes crucial as fiber length increases. Implementing strategies to mitigate latency, such as using multiple DMA channels, is essential for maintaining performance over longer distances;
- **Power Efficiency:** While the system shows good power efficiency relative to its performance, further improvements could be achieved by exploring more efficient power management techniques, particularly when scaling the number of lanes or DMA channels;
- **Future Directions:** The discrepancies observed in single-lane configurations and the potential for further power optimisation present opportunities for future research. Additionally, investigating the use of platforms with higher MPS capabilities or more advanced PCIe generations could further enhance the system's capabilities, particularly in long-distance applications.

In conclusion, the experiments conducted provide a comprehensive understanding of the factors affecting the performance and efficiency of PCIe over fiber systems. These insights not only validate the system's design but also pave the way for future enhancements and

optimisations, ensuring that the system can meet the demands of increasingly complex and data-intensive applications.

6

Conclusion

This chapter summarises the key accomplishments of this dissertation and suggests potential areas for future research. Section 6.1 outlines the most significant findings and insights gained from this work. To conclude, Section 6.2 explores possible directions for continued research.

6.1. Summary

In this thesis, we embarked on an investigation into the potential of PCIe technology over fiber connection as a solution for high-speed data transfer, with a focus on achieving a decoupled and modular design. Throughout this research, we delved into a range of relevant technologies, including fiber optics, communication protocols, and the hardware platforms necessary for their implementation, to thoroughly explore the viability and performance of a PCIe Over Fiber solution using FPGAs.

The project continued with a comprehensive analysis of the performance, functional, and technical requirements needed to meet the desired standards. This analysis was instrumental in guiding the selection of components and shaping the overall system architecture, ensuring that each element was aligned with the goals of scalability, reliability, and high performance. Consequently, the careful consideration of these factors allowed us to design a system architecture that could adapt to evolving needs while maintaining robust performance.

During the implementation phase, we focused on integrating FPGA platforms, configuring PCIe cores, and implementing optical interfaces, all of which were critical to achieving the system's objectives. This phase presented several challenges, including issues with the receiver detection mechanism, clock misalignment, and component incompatibilities discovered after selection. Each of these challenges was addressed through detailed analysis and the development of targeted solutions, allowing us to refine the system and ensure its reliability.

The final system was rigorously validated through a series of experiments, demonstrating that it successfully meets the desired performance criteria. Additionally, we also identified the relevant parameters and elements that constrain the available performance. The results confirmed that the system is capable of achieving a throughput of 58.02 Gbps using a Gen3 x8 configuration over a distance of 10 meters, with the potential to reach up to 20 meters without significant degradation in performance. These findings underscore the effectiveness of the PCIe over fiber approach in high-speed data transfer applications.

In conclusion, this thesis has demonstrated the significant potential of PCIe technology over fiber as a viable solution for high-speed data transfer in a modular and decoupled

design. By thoroughly exploring and addressing the challenges associated with integrating advanced hardware and communication protocols, we have laid a solid foundation for further advancements in this field. The results achieved not only validate the approach but also highlight areas for future improvement, offering a clear path forward for enhancing system performance and scalability. As technology continues to evolve, the insights gained from this research will contribute to the ongoing development of more efficient and reliable data transfer systems. Thus, ensuring that they meet the growing demands of modern applications.

6.2. Future Work

Looking ahead, future research could expand upon these findings by upgrading to newer PCIe generations and utilising more advanced hardware. Particular emphasis should be placed on exploring higher Maximum Payload Sizes (MPS) and Read Completion Boundaries (RCB), which have been identified as potential bottlenecks in this study. By addressing these areas, future developments could further enhance the capabilities and performance of PCIe over fiber systems, paving the way for even more robust and scalable high-speed data transfer solutions.

Additionally, implementing Samtec's Firefly connectors on both sides of the link could be a promising direction for future research. Using Firefly connectors would allow access to sideband signals, which are often critical for maintaining system synchronisation and managing out-of-band signalling. This approach could improve the overall reliability and functionality of the PCIe link over fiber, providing additional control and monitoring capabilities that are typically not accessible with standard optical transceivers. Exploring the integration of these connectors could lead to more versatile and robust designs, especially in applications requiring precise control over sideband communication.

Another intriguing area for future exploration is experimenting with different connection schemes, such as integrating two PCIe cores within the FPGA to connect to two different PCs. This dual-core setup could enable more complex data distribution and parallel processing scenarios, potentially increasing the overall system throughput and redundancy. By leveraging multiple PCIe links, researchers could investigate new ways to optimise data transfer efficiency and resilience, particularly in environments where high availability and fault tolerance are critical. This line of research could open up new possibilities for multi-system communication and distributed computing over fiber, further advancing the state of the art in high-speed data transfer technologies.

References

[1] William Stallings. *High Speed Networks and Internets: Performance and Quality of Service*. 2nd. USA: Prentice Hall PTR, 2001. ISBN: 0130322210.

[2] Organization for. *ISO/IEC 7498-1:1994*. 2024. URL: https://www.iso.org/standard/20269.html.

[3] B. Forouzan. *Data Communications and Networking: Fifth Edition*. McGraw-Hill Education, 2012. ISBN: 9780077445393. URL: https://books.google.nl/books?id=yMSbngEACAAJ.

[4] J.F. Kurose and K.W. Ross. *Computer Networking: A Top-down Approach*. Pearson, 2017. ISBN: 9780133594140. URL: https://books.google.nl/books?id=OljpOAAACAAJ.

[5] R. Ramaswami, K. Sivarajan, and G. Sasaki. *Optical Networks: A Practical Perspective*. Morgan Kaufmann series in networking. Elsevier Science, 2009. ISBN: 9780080920726. URL: https://books.google.nl/books?id=WpByp4Ip0z8C.

[6] PCI-SIG. *PCI Express Base Specification Revision 4.0, Version 1.0*. https://pcisig.com/specifications/pciexpress/specifications/PCI_Express_Base_4.0_Rev_1.0.pdf. 2017.

[7] *PCIe Cabling*. 2024. URL: https://pcisig.com/blog/pcie%C2%AE-cabling-%E2%80%93-journey-copprlink%E2%84%A2.

[8] *Avago MiniPOD*. 2024. URL: https://www.broadcom.com/video/6328986830112.

[9] *Samtec FireFly*. 2024. URL: https://www.samtec.com/optics/systems/firefly/.

[10] Samtec. *PCI Express®-Over-Fiber FireFly™ Adaptor Card*. 2024. URL: https://www.samtec.com/products/pcoa.

[11] C. Maxfield. *The Design Warrior's Guide to FPGAs: Devices, Tools and Flows*. Elsevier Science, 2004. ISBN: 9780080477138. URL: https://books.google.com.cy/books?id=dnuwr2xOFpUC.

[12] A. Rubini and J. Corbet. *Linux Device Drivers*. Nutshell handbooks. O'Reilly & Associates, 2001. ISBN: 9780596000080. URL: https://books.google.nl/books?id=53-wm0d-BFwC.

[13] Xilinx. *Xilinx DMA IP Reference drivers*. 2024. URL: https://github.com/Xilinx/dma_ip_drivers.

[14] *AMD MicroBlaze™ xn–Processor-o69d*. [Online; accessed 16. Aug. 2024]. Aug. 2024. URL: https://www.xilinx.com/products/design-tools/microblaze.html.

[15] *FOPC*. [Online; accessed 16. Aug. 2024]. Aug. 2024. URL: https://www.samtec.com/products/fopc.

[16] *Cisco Transceiver Modules*. [Online; accessed 16. Aug. 2024]. Apr. 2024. URL: https://www.cisco.com/c/en/us/products/collateral/interfaces-modules/transceiver-modules/datasheet-c78-736282.html.

[17] *Intel® Xeon® W-1250P Processor (12M Cache, 4.10 GHz)*. [Online; accessed 16. Aug. 2024]. Aug. 2024. URL: https://www.intel.com/content/www/us/en/products/sku/199340/intel-xeon-w1250p-processor-12m-cache-4-10-ghz/specifications.html.

[18] *FFVA1156 (XCKU15P) and FFRA1156 (XQKU15P) • UltraScale+ Device Packaging and Pinouts Product Specification User Guide (UG575) • Reader • AMD Technical Information Portal*. [Online; accessed 16. Aug. 2024]. Aug. 2024. URL: https://docs.amd.com/r/en-US/ug575-ultrascale-pkg-pinout/FFVA1156-XCKU15P-and-FFRA1156-XQKU15P.

[19] Xilinx. *DMA for PCI Express (PCIe) Subsystem*. 2024. URL: https://www.xilinx.com/products/intellectual-property/pcie-dma.html.

[20] Andre Richter. *pcie-lat*. 2020. URL: https://github.com/andre-richter/pcie-lat.

[21] *Vivado Overview*. [Online; accessed 16. Aug. 2024]. Aug. 2024. URL: https://www.xilinx.com/products/design-tools/vivado.html.

[22] Qoitech. *Otii Arc Pro*. 2024. URL: https://www.qoitech.com/otii-arc-pro/.

[23] Xilinx. *DMA Subsystem for PCI Express (Vivado 2016.3) - Performance Numbers*. 2023. URL: https://support.xilinx.com/s/article/68049?language=en_US.

[24] Xilinx. *In-System IBERT IP*. 2024. URL: https://www.xilinx.com/products/intellectual-property/in_system_ibert.html.

[25] Nasser M. Juma et al. "Integrating Experimentation and Instrumentation". In: *Measuring the Speed of Light in an Optical Fiber*. 2009. URL: https://web.phys.ksu.edu/posters/2009/juma-Adv-Lab-S09.pdf.