

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL

ENHANCING THE PCIE DATA LINK LAYER INTEGRITY
USING ENERGY EFFICIENT LIGHTWEIGHT
CRYPTOGRAPHIC ALGORITHMS

MUHAMMED MUSTAFA ÖNEY

A THESIS OF MASTER OF SCIENCE
DEPARTMENT OF ELECTRONICS ENGINEERING

ADVISOR: ASSIST. PROF. İHSAN ÇİÇEK

OCTOBER 2024

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL

ENHANCING THE PCIE DATA LINK LAYER
INTEGRITY USING ENERGY EFFICIENT
LIGHTWEIGHT CRYPTOGRAPHIC ALGORITHMS

MUHAMMED MUSTAFA ÖNEY

A THESIS OF MASTER OF SCIENCE
DEPARTMENT OF ELECTRONICS ENGINEERING

ADVISOR: ASSIST. PROF. İHSAN ÇİÇEK

OCTOBER 2024

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ENERJİ VERİMLİ HAFİF ŞİFRELEME
ALGORİTMALARI KULLANARAK PCIE VERİ
BAĞLANTISI KATMANI BÜTÜNLÜĞÜNÜN
GELİŞTİRİLMESİ

MUHAMMED MUSTAFA ÖNEY

YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN: DR. ÖĞR. ÜYESİ İHSAN ÇİÇEK

EKİM 2024



MASTER of SCIENCE JURY APPROVAL FORM

A thesis submitted by Muhammed Mustafa Öney defended on 02/10/2024 before the jury formed with the 30/09/2024 date and 2024/55 numbered decision of the GTU Graduate Administration Board, has been accepted as a MASTER of SCIENCE thesis in the Department of Electronics Engineering.

JURY

MEMBER

(THESIS ADVISOR) : Assist. Prof. Dr. İhsan ÇİÇEK

MEMBER

: Assoc. Prof. Dr. Serdar Süer ERDEM

MEMBER

: Assist. Prof. Dr. Salih BAYAR

APPROVAL

Gebze Technical University Graduate Administration Board/....../..... date and
...../.... numbered decision.

SIGNATURE/SEAL

On behalf of my beloved daughter and wife...

ABSTRACT

PCIe, a high-speed serial communication protocol, uses the Link Cyclic Redundancy Check (LCRC) for data integrity, though traditional LCRC methods are power-intensive and not suited for energy-efficient applications. This thesis explores enhancing data integrity in PCIe's data link layer using lightweight cryptographic (LWC) algorithms to improve energy efficiency.

The research identifies a gap in studies focusing on reducing power consumption in PCIe-based data integrity mechanisms. This study addresses this gap by proposing energy-efficient alternatives to the standard LCRC, including LWC algorithms like ASCON, GRAIN, and Trivium, and parallel LFSR structures. These methods are compared to the LCRC in terms of power consumption, data integrity, and speed, achieving significant power reductions and improved efficiency.

Key improvements include:

- An energy-efficient method that maintains the LCRC function from the PCI Express specification.
- Enhanced LCRC method, reducing dynamic power consumption by 38% to 85%.
- Lightweight cryptographic algorithms and Reed-Solomon code as viable alternatives to LCRC.
- Enhanced data integrity with low power consumption and high operating frequencies.
- Increased reliability for battery-powered applications with new LCRC methods.

This research advances PCIe data integrity and supports energy-efficient, power-constrained systems, with implications for high-performance computing and data-intensive applications.

Keywords: PCIe, Data Integrity, LCRC, LWC, Reed-Solomon Code, Data Link Layer

ÖZET

Yüksek hızlı seri iletişim protokolü olan PCIe, veri bütünlüğü için Bağlantı Döngüsel Yedeklilik Denetimi'ni (LCRC) kullanır, ancak standart LCRC yöntemi güç yoğun ve enerji açısından verimli uygulamalar için uygun değildir. Bu tez, enerji verimliliğini artırmak için hafif kriptografik (LWC) algoritmaları kullanarak PCIe'nin veri bağlantı katmanında veri bütünlüğünü geliştirmeyi araştırmaktadır.

Araştırmada, PCIe tabanlı veri bütünlüğü mekanizmalarında güç tüketimini azaltmaya odaklanan çalışmalarda bir boşluk olduğu tespit edilmiştir. Bu çalışma, ASCON, GRAIN ve Trivium gibi LWC algoritmaları ve paralel LFSR yapıları dahil olmak üzere standart LCRC'ye enerji açısından verimli alternatifler önererek bu boşluğu gidermiştir. Bu yöntemler, güç tüketimi, veri bütünlüğü ve hız açısından LCRC ile karşılaştırılarak ve önemli güç tüketimi kazanımları elde edilmiş ve verimlilik artırılmıştır.

Bu çalışma ile elde edilen önemli iyileştirmeler şunlardır:

- PCI Express spesifikasyonunda verilen standart LCRC mekanizmasının işlevi korunarak, enerji açısından daha verimli bir yöntem sunulmaktadır.
- Geliştirilmiş LCRC adı verilen yöntem sayesinde, LCRC yapısının dinamik güç tüketiminde %38'den %85'e kadar önemli bir azalma sağlanır.
- Geliştirilmiş LCRC'nin yanı sıra çeşitli hafif şifreleme algoritmaları ve Reed Solomon kodu üzerinde de bir çalışma yapılmış ve bunların standart LCRC yöntemine iyi bir alternatif olabileceği gösterilmiştir.
- Batarya ile beslenen uygulamalar için daha fazla veri bütünlüğü ve güvenilirliği sağlayan yeni LCRC yöntemleri sunulmuştur.

Bu araştırma, PCIe veri bütünlüğünü bir adım daha ileri taşıyarak, yüksek performanslı veri işlemleri ve yoğun hesaplama uygulamaları için verimli bir ortam sunulmasına katkıda bulunmaktadır.

Keywords: PCIe, Veri Bütünlüğü, LCRC, LWC, Reed-Solomon Kodu, Veri Bağlantısı Katmanı

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Dr. İhsan Çiçek for his constant and determined guidance, and patience throughout my project. His insightful comments and unwavering support were the most influential factors in shaping this work. It was an honor and a privilege to collaborate with him.

In addition to my advisor, the support of my workplace and managers is undeniable. I would also like to express my sincere gratitude to TÜBİTAK BİLGEM, especially the Integrated Circuit Design and Training Laboratory and the Scientific and Technological Research Council of Turkey. Their generous provision of resources and equipment was crucial to the successful completion of this study. Thanks to the technical and moral support of my colleagues, I was able to overcome many difficulties. Their cooperation and encouragement contribute greatly to the progress of this project.

Finally, I am grateful to my beloved wife and family who have always supported me along many years. Their unwavering love, endless patience and constant encouragement have been the supporting point of my academic journey. Their efforts have a visible impact on shaping my life.

I am proud of being their son, husband and father.

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	vi
ÖZET	vii
ACKNOWLEDGEMENT	viii
TABLE of CONTENTS	ix
LIST of SYMBOLS and ABBREVIATIONS	xi
LIST of FIGURES	xiii
LIST of TABLES	xiv
1. INTRODUCTION	1
1.1. Background	2
1.2. Motivation of the Thesis	5
1.3. Literature Overview	5
1.4. Contributions of the Thesis	6
2. PCIe STANDARD OVERVIEW	9
2.1. The Operation of PCIe	12
2.1.1. Transaction Layer	13
2.1.2. Data Link Layer	17
2.1.3. Physical Layer	20
3. PCIE IP CORE DESIGN	24
3.1. Control Blocks	24
3.2. Tx Path of the Data Link Layer	26
3.3. Rx Path of the Data Link Layer	30
4. ALTERNATIVE APPROACHES FOR DATALINK INTEGRITY	35
4.1. Standard LCRC Mechanism in PCIe Specification	36
4.2. Enhanced LCRC Mechanism	38
4.3. Lightweight Cryptographic Algorithms as LCRCs	40
4.3.1. ASCON LWC	40
4.3.2. GRAIN LWC	41
4.3.3. Trivium LWC	43
4.3.4. Reed Solomon	46
5. PERFORMANCE EVALUATION	49
REFERENCES	53

BIOGRAPHY	59
PUBLICATIONS WITHIN the SCOPE of THESIS STUDY	60

LIST OF SYMBOLS AND ABBREVIATIONS

Ack	: Acknowledgement
AEAD	: Authenticated Encryption with Associated Data
AGP	: Accelerated Graphics Port
AI	: Artificial Intelligence
ASIC	: Application Specific Integrated Circuit
BUFG	: Buffer Global
CDC	: Cyclic Delay Compensation
CDR	: Clock Data Recovery
CPU	: Central Processing Unit
CRC	: Cyclic Redundancy Check
CXL	: Compute Express Link
DC-NXT	: Design compiler NXT
DLCMSM	: Data Link Control and Management Status State Machine
DLL	: Data Link Layer
DLLP	: Data Link Layer Packet
DMA	: Direct Memory Access
DRAM	: Dynamic Random Access Memory
ECG	: Electrocardiogram
ECRC	: End-to-End Cyclic Redundancy Check
FEC	: Forward Error Correction
FF	: Flip Flop
FIFO	: First in first Out
FC	: Flow Control
FPGA	: Field Programmable Gate Array
GF	: Galois Field
GPU	: Graphics Processing Unit
HDL	: Hardware Description Language
HPC	: High-Performance Computing
I/O	: Input/Output
IoT	: Internet of Things
IP	: Intellectual Property
LCRC	: Link Cyclic Redundancy Check
LFSR	: Linear Feedback Shift Register
LTSSM	: Link Training and Status State Machine
LUT	: Look Up Table
LUTRAM	: Look Up Table Random Access Memory
LWC	: Lightweight Cryptographic Algorithm
MAC	: Media Access Control
MATLAB	: Matrix Laboratory
Nak	: Negative Acknowledgement
NFSR	: Non-Linear Feedback Shift Register
NIST	: National Institute of Standards and Technology
NRS	: Next Receive Sequence

NTS	: Next Transmit Sequence
NVMe	: Non-Volatile Memory Express
OS	: Ordered Set
OTPs	: One Time Password
PCI	: Peripheral Component Interconnect
PCIe	: Peripheral Component Interconnect Express
PCI-X	: Peripheral Component Interconnect Extended
PAM	: Pulse Amplitude Modulation
PC	: Personal Computer
PCI-SIG	: PCI Special Interest Group
PHY	: Physical Layer
PIPE	: Physical Interface for PCI Express
PLL	: Phase Locked Loop
PM	: Power Management
QoS	: Quality of Service
RAID	: Redundant Array of Independent Disks
RFID	: Radio Frequency Identification
RS	: Reed Solomon
Rx	: Receiver
SerDes	: Serializer-Deserializer
SHA	: Secure Hash Algorithm
SSD	: Solid State Drive
TLP	: Transaction Layer Packet
TSMC	: Taiwan Semiconductor Manufacturing Company
Tx	: Transmitter
UCIe	: Universal Chiplet Interconnect Express
WNS	: Worst Negative Slack
$c(x)$	: Codeword Polynomial
$g(x)$	: The Generator Polynomial
$m(x)$	: The Message
$r(x)$	: Received Polynomial
$S(x)$	: Syndrome Polynomial
MB/s	: Megabytes per Second
MHz	: Megahertz
ns	: Nanosecond
GHz	: Gigahertz
GT/s	: Giga Transfers per Second
W	: Watt

LIST OF FIGURES

	<u>Page</u>
Figure 1.1: PCIe usage in computer architecture.	1
Figure 1.2: HPC cluster overview.	2
Figure 1.3: PCIe $\times 16$ lane example.	4
 Figure 2.1: PCIe lanes create links.	 10
Figure 2.2: PCIe lanes include two differential pairs.	10
Figure 2.3: A typical PCIe based system.	12
Figure 2.4: PCIe protocol stack.	13
Figure 2.5: Sections of a transaction layer packet.	13
Figure 2.6: Extensions of a TLP.	14
Figure 2.7: Transaction layer mechanisms : Flow control, Transaction Ordering and QoS.	16
Figure 2.8: Data link layer.	17
Figure 2.9: PCIe physical layer architecture.	20
 Figure 3.1: DLCMSM States.	 24
Figure 3.2: Data link layer micro architecture.	26
Figure 3.3: Transaction layer packet generator.	27
Figure 3.4: PCIe Data Link Layer packet formats.	28
Figure 3.5: Overview of the Ack/Nak protocol.	30
Figure 3.6: Synthesis schematic of our PCIe DLL in Synopsys DC-NXT.	31
 Figure 4.1: LCRC mechanism in PCIe Data Link Layer.	 35
Figure 4.2: Calculation of LCRC.	36
Figure 4.3: Overview of the building blocks in Grain.	42
Figure 4.4: Overview of the building blocks in Trivium.	44
 Figure 5.1: Simulation snapshot of incoming TLP.	 49
Figure 5.2: LCRC check of incoming TLP on simulation environment.	50

LIST OF TABLES

	<u>Page</u>
Table 2.1: Link width and bandwidth for different PCIe generations.	11
Table 3.1: The cell count of our DLL.	32
Table 3.2: The area of our DLL.	32
Table 3.3: The power consumption details of our DLL.	34
Table 4.1: Standart serial LCRC wrapper implementation results.	37
Table 4.2: Enhanced LCRC implementation results.	39
Table 4.3: ASCON as LCRC implementation results.	41
Table 4.4: GRAIN as LCRC implementation results.	43
Table 4.5: Trivium as LCRC implementation results.	45
Table 4.6: Reed Solomon as LCRC implementation results.	48
Table 5.1: Comparison of various LCRC implementations on Xilinx KCU105 board.	51

1. INTRODUCTION

Peripheral Component Interconnect Express (PCIe) is a serial communication protocol that succeeded PCI, PCI-X, and AGP, which previously facilitated high-speed parallel communication. Developed beginning in 2001, PCIe is utilized to connect high-performance hardware. Its widespread application spans both personal computers and servers, making it a critical component in various domains. PCIe provides high bandwidth data transmission between the Central Processing Unit (CPU) and Graphics Processing Unit (GPU) as it can be seen in Fig. 1.2. This connection is especially important in gaming PCs and professional applications that require graphics processing. GPUs have much more processing power than the CPU for computationally intensive operations, and thanks to PCIe, fast and efficient data exchange is provided between these two processing units. This is critical in areas such as graphics and video processing, as well as scientific computing and machine learning [1]. It is also widely used in Solid State Drives (SSDs). SSDs, which have much faster data reading and writing speeds compared to traditional hard disk drives, can maximize their performance thanks to the PCIe connection. PCIe-based Non-Volatile Memory Express (NVMe) SSDs stand out with their high-speed data transmission and low latency. This provides a great advantage, especially for data-intensive applications, games and other scenarios that require high-speed data processing.

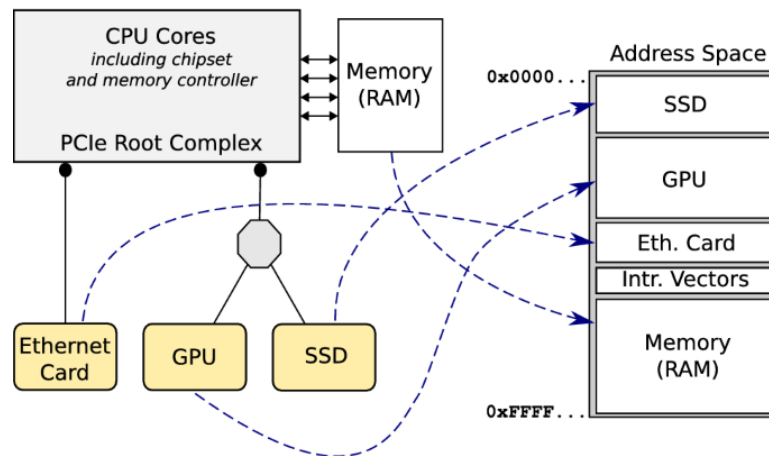


Figure 1.1: PCIe usage in computer architecture [2].

Artificial intelligence and machine learning applications need powerful hardware to quickly process large amounts of data. PCIe allows GPUs and other accelerators to communicate effectively with CPUs, providing an ideal solution for such applications. During training of artificial intelligence models, the high bandwidth and low latency of PCIe play an important role when large data sets need to be processed and analyzed. This enables a wide range of AI applications to run efficiently, from research laboratories to large data centers [3].

The use of PCIe in avionic systems, that is, in the electronic systems of aircraft, offers significant advantages. Thanks to features such as high bandwidth, low weight, resistance to electromagnetic interference and robustness, PCIe-based optical cables provide many advantages over copper cables. This improves reliability and performance in in-flight data transmission and communication systems. PCIe connections provide fast and reliable data transmission between various sensors, control systems and communication devices found on aircraft [4].

1.1. Background

Originally, it was designed solely as a local bus, providing data transmission between the CPU, GPU and other I/O devices. But over time, PCIe's capabilities have been expanded and expanded into broader application areas, such as data centers and high-computation environments.

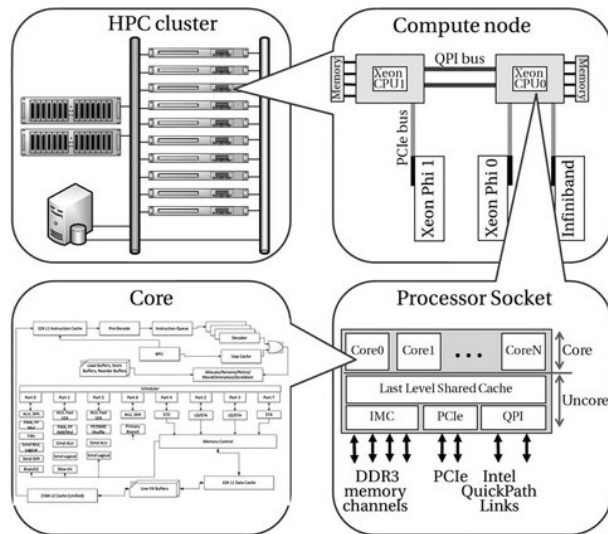


Figure 1.2: HPC cluster overview [5].

PCIe has begun to play a critical role in modern data centers and high-performance computing (HPC) environments by supporting point-to-point links and network connections. With this protocol becoming a standard, it has become an indispensable technology for large-scale data processing and storage solutions. Especially in artificial intelligence and machine learning applications, the advantages offered by PCIe are of great importance for the rapid processing and analysis of large data sets [6].

PCIe has multiple generations and configurations that offer different speeds. Each generation makes the protocol even more capable, offering faster speeds and advanced features compared to the previous generation. Each generation of PCIe following new technologies and improvements that increase data transmission speeds and enable more efficient data transfer.

- PCIe 1.0: The first PCIe generation offers speeds of 2.5 Gigatransfers per second (GT/s) and can reach data transfer rates of 250 MB/s in $\times 1$ lane configuration.
- PCIe 2.0: Second generation PCIe increases speeds to 5 GT/s and enables data transfer rates of 500 MB/s in $\times 1$ lane configuration.
- PCIe 3.0: Third generation PCIe increases speeds to 8 GT/s and enables data transfer rates of 1 GB/s in $\times 1$ lane configuration. This generation provides higher efficiency in data transmission with the 128b/130b coding system and creates less overhead compared to the previous 8b/10b coding system.
- PCIe 4.0: Fourth generation PCIe delivers 2 GB/s data transfer rates in $\times 1$ lane configuration, increasing speeds to 16 GT/s. It stands out with its physical layer improvements and the ability to operate at higher frequencies. This means higher data transfer speeds for data centers and high-performance computing applications.
- PCIe 5.0: Fifth generation increases speeds to 32 GT/s and enables data transfer rates of 4 GB/s in $\times 1$ lane configuration. This generation doubles data transmission speeds while maintaining backward compatibility with previous generations. This ensures that existing PCIe devices can run on next-generation PCIe slots.
- PCIe 6.0: The latest generation, PCIe 6.0, offers speeds of 64 GT/s and can reach data transfer rates of 8 GB/s in $\times 1$ lane configuration. The newest generation,

PCIe 6.0, enables high data transfer speeds by offering features such as PAM-4 (Pulse Amplitude Modulation) signaling and advanced error correction mechanisms (FEC). Additionally, it provides low latency and high reliability [7].

PCIe can operate in different configurations depending on the number of lanes used in data transmission. A PCIe link can have different lane configurations such as $\times 1$, $\times 4$, $\times 8$, $\times 16$. These lanes directly affect data transmission speed and bandwidth. For example, PCIe 3.0 in $\times 16$ configuration can deliver data transfer speeds of up to 16 GB/s. This flexibility allows PCIe to meet both low and high bandwidth requirements.

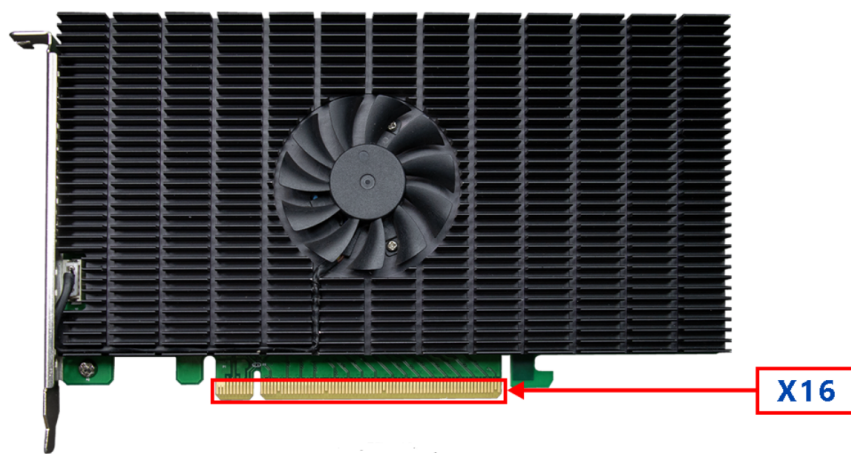


Figure 1.3: PCIe $\times 16$ lane example.

With each generation, the PCIe protocol is equipped with new features that allow communication at higher speeds. These developments have increased the performance of PCIe and enabled its use in wider application areas. Thanks to developing silicon technology, new protocols based on PCIe have begun to emerge. One of the most notable among these protocols is the Compute Express Link (CXL) protocol. CXL provides data transmission with higher efficiency and lower latency by using the physical layer of PCIe. CXL is an open industry standard interconnect protocol that provides cache coherency and memory semantics between different devices such as processors, accelerators, and memory devices. CXL is designed specifically for data centers and high-performance computing environments. One of the main advantages of CXL is that it offers lower latencies and higher bandwidth, using the same physical layer as PCIe [8].

CXL works using the physical layer of PCIe and supports alternative protocols on the PCIe physical layer (PHY). This allows CXL to deliver high performance with low latency and power efficiency. For example, PCIe 6.0 offers speeds of 64.0 GT/s with PAM-4 signaling, while the CXL protocol optimizes data transmission by taking advantage of these higher speeds [9]. CXL also supports important use cases such as memory expansion, resource pooling, and multi-server resource sharing [10]. This provides more efficient and scalable solutions in data centers and high-performance computing environments.

The fact that computation-intensive hardware is closely related to PCIe is also valid for battery-powered mobile systems, whose use is increasing today. These mobile systems, equipped with high computing capabilities, must work intertwined with PCIe. In other words, edge devices now also contain PCIe. For example, license plate recognition systems, military applications, artificial intelligence applications that perform edge calculations, and various IoT devices that perform edge calculations can be classified in this category.

1.2. Motivation of the Thesis

With the rise of mobility, the need for low power also increases. Mobile devices and edge devices need energy efficiency to extend battery life. PCIe's energy efficiency increases the performance of these devices while minimizing energy consumption. For example, PCIe's energy-efficient modes and low power consumption enable mobile devices and edge devices to run longer [9]. But beyond these low power consumption modes, there are some other ways to reduce power consumption applications can also be made. In addition, it is possible to increase data integrity while aiming to reduce power consumption, which is the main focus of this study.

1.3. Literature Overview

Although PCIe-based power consumption reduction studies are rare in the literature, there are the following studies on increasing data integrity. Some of these bring unique reliability solutions. Some have also offered solutions with merging topics such as blockchain and heavy-weighted crypto algorithms.

Methods to detect and correct silent data corruption in high-performance computing have been investigated by [11], with significant improvements in system reliability achieved through robust error detection and automatic correction mechanisms. An IoT-based framework for ECG monitoring with secure data transmission has been developed by Guangyu [12], which has been found to be crucial for healthcare applications, thereby improving data integrity and security in healthcare monitoring. A blockchain-based mechanism for data integrity protection in cloud environments has been presented by PengCheng et al. [13], with data trust and reliability enhanced through decentralized verification. A method for privacy-preserving data integrity verification in healthcare has been presented by Jian et al. [14], with a balance between data integrity and privacy achieved using lightweight structures critical for sensitive data. Various encryption techniques to enhance cloud data security have been detailed by Bijeta et al. [15], with significant improvements in data confidentiality and reliability noted. The role of blockchain technology in enhancing electronic health record interoperability and integrity has been examined by Rateb et al. [16], contributing to secure and reliable health data exchange. Strategies to minimize risks and increase data transfer reliability by securing data transfers against silent data corruption have been presented by Batyr et al. [17]. Methods to prevent data loss and corruption in file transfers have been examined by Ahmed and Engin [18], with increased data reliability achieved through fast and effective integrity verification techniques. Finally, a proactive approach to protecting cloud data integrity by integrating blockchain technology for instant detection of data corruption in cloud computing has been presented by E. Angelin et al. [19]. These studies offer innovative solutions to ensure the integrity and reliability of data across different platforms and applications. However, all these studies were done with high power consuming PCIe based applications.

1.4. Contributions of the Thesis

In this study, unlike the listed studies, optimizations and innovative approaches on special CRC(Cyclic Redundancy Code) structures in the PCIe protocol stack are discussed. This thesis improves the integrity mechanisms at a slightly lower level in terms of power consumption and achieves this within the PCI Express interface. Our thesis

also contributes to end-to-end integrity by providing a more sustainable environment for power-limited environments by using lightweight cryptographic algorithms. The following gains were obtained:

- We eliminated the CDC structure required for serial LFSRs in the PCIe protocol to be used in practice, thus eliminating the need for clock domain switching and CRC mechanisms can be operated at high speeds within a single clock period, allowing for convenient timing and increased overall performance.
- We proposed new LCRC methods by maintaining the low power consumption and high operating frequencies we achieved by parallelizing the LFSRs within the LCRC mechanism. And we further increased the reliability with the suggested methods.
- The proposed LFSR parallelization method has resulted in a reduction of the dynamic power consumption of the LCRC structure used in the PCIe protocol by 38% to 85%. This method can also be applied to other similar CRC calculation algorithms, such as ECRC and DLLP CRC.
- The processed data can be sampled at the Nyquist frequency, which increases data reliability, when the data transfer rate for the proposed LCRC algorithms is selected to support higher speeds than the trained link speed for link communication.
- The amount of data the integrity block can instantly handle with the same amount of energy has been improved.
- Although we sacrificed backward compatibility with light-weight encryption algorithms, we increased maintainability by proposing methods that provide higher integrity than the integrity method we have already developed. And we have created a more reliable environment for battery powered applications.
- We not only improved integrity, but also created an environment for data security studies.

In the light of the above information, this thesis is organized as follows: Following the introduction, the general and technical details of the PCIe Standard are discussed in the second section and the reason for focusing on LCRC in this study is examined. Chapter

3 shows the location of the newly developed integrity mechanisms within the PCIe IP core hardware and the LCRC process, as well as explaining the designed DLL layer in detail. Chapter 4 details the work done to improve the LCRC algorithm presented in the specification in terms of integrity and operating frequency. In Chapter 5, performance results are presented and discussed, and the future research directions discussed and the thesis is concluded.

2. PCIe STANDARD OVERVIEW

PCI Express is a high-speed interface standard that connects peripheral devices to a computer's motherboard. It is often used to connect graphics cards, sound cards, network adapters, and other peripherals to a processor. PCI Express is similar to the older PCI and AGP standards, offering higher speeds and greater bandwidth than its predecessors. It was designed to replace the older PCI standard, which was first introduced in the 1990s. The main difference between PCI and PCI Express is how they transfer data. PCI uses a parallel bus architecture, where data is transmitted over multiple wires at the same time. This allows for higher bandwidth, but it also increases the number of pins needed to connect components. PCI Express, on the other hand, uses a serial bus architecture, where data is transmitted over a single pair of wires. This reduces the number of pins needed, but it also limits the amount of data that can be transferred at one time. However, PCI Express uses a technique called data encoding to increase the amount of data that can be transmitted over a single pair of wires, allowing it to achieve higher bandwidth than PCI. The development of the PCIe standard began in 2003 with the release of version 1.0. Since then, the standard has undergone several updates, including versions 1.1, 2.0, 3.0, 3.1, 4.0, 5.0, and 6.0, each providing higher bandwidth and lower latency than the previous version, and work is still ongoing on the latest version, PCIe 7.0. The first official draft of PCIe 7.0 v0.5 was released in 2024 [20]. PCIe provides faster data transfer between a processor and peripheral devices such as graphics cards, solid state drives, and network cards; meeting the needs of high-performance applications such as cryptocurrency mining, artificial intelligence acceleration, and data centers. In terms of hardware, PCIe uses a point-to-point architecture that provides faster data transfer and scalability. PCIe standards are maintained by the PCI-SIG, an industry group that includes major companies such as Intel, AMD, and NVIDIA.

PCIe uses a point-to-point architecture that provides fast data transfer and scalability, and is a high-speed serial bus interface. Like many high-speed serial transmissions, PCIe uses a bidirectional connection and is capable of sending and receiving informa-

tion simultaneously. The model used is called a dual-simplex connection because each interface has a unidirectional transmit path and a unidirectional receive path. Since PCIe is a point-to-point architecture, a dual-simplex connection must be between two devices. This path between devices is called a "Link", and Links consist of one or more transmitter and receiver pairs as it can be seen in Fig. 2.1. Such a transmitter-receiver pair is called a "Lane" see Fig. 2.2, and the specification allows a Link to consist of 1, 2, 4, 8, 12, 16, or 32 Lanes.

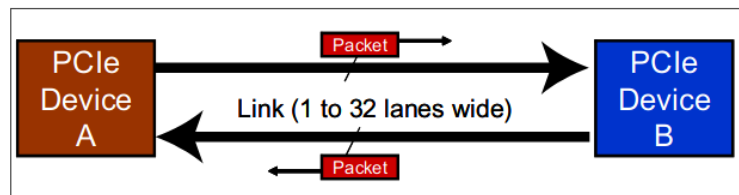


Figure 2.1: PCIe lanes create links [21].

The number of lanes is called Link Width and is represented as $\times 1$, $\times 4$, $\times 8$, $\times 16$ and $\times 32$. The more lanes used in the design, the more bandwidth will be provided, but increasing the number of lanes also increases power consumption and area requirements.

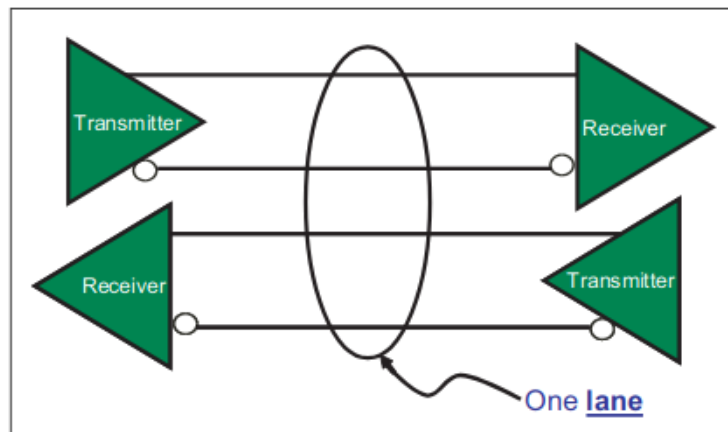


Figure 2.2: PCIe lanes include two differential pairs [21].

One of the most important design goals for PCIe is backward compatibility with PCI software. This motivates the use of software that is already installed and running on existing systems. In order to achieve this, all address spaces used in PCI have been moved

to the PCIe structure, and since PCIe requires more space, the address space has simply been expanded compared to the PCI address space. In this way, old software can also work in harmony with new PCIe hardware. This also applies to the new generations of PCIe; software defined for PCIe 2.0 can also work in harmony with PCIe 5.0 hardware. One of the biggest motivations for PCI Express is to provide high-speed data transfer by reducing the number of pins. When data is to be transferred through serial channels, it is very clear that much higher transfer speeds must be provided compared to parallel channels. Nevertheless, for the first generations of PCIe, versions 1.0 and 2.0, the transceiver circuits can operate at speeds of 2.5 GT/s and 5.0 GT/s. In addition, there is a speed of 8.0 GT/s for the Gen3 version. The symbol received as 10 bits in the receiver/transmitter circuit is sampled at a speed of up to 8.0 GT/s and transferred differentially out of the pin. The bandwidth calculation for PCIe devices can be seen in Table 2.1.

Table 2.1: Link width and bandwidth for different PCIe generations.

Link Width	x1	x2	x4	x8	x12	x16	x32
Gen1 Bandwidth (GB/s)	0.5	1	2	4	6	8	16
Gen2 Bandwidth (GB/s)	1	2	4	8	12	16	32
Gen3 Bandwidth (GB/s)	2	4	8	16	24	32	64
Gen4 Bandwidth (GB/s)	4	8	16	24	32	64	128
Gen5 Bandwidth (GB/s)	8	16	24	32	64	128	256
Gen6 Bandwidth (GB/s)	16	24	32	64	128	256	512

There is no clock signal transferred in PCIe links. Instead, a symbol is encrypted before being transmitted to the receiver and clk information is embedded in the data. The serial data arriving at the receiver is examined and the clock information in the data is recovered via PLL and used in the PHY layer.

When the PCIe structure is examined at the system level, it can only be used as a simple tree structure which is shown in Fig. 2.3. The connection between the CPU and PCIe buses can include different components such as processors and DRAM interfaces. This structure is called the entire "Root Complex". The root complex is the structure responsible for transmitting the data it receives from the processor, main memory or peripherals to the appropriate devices. Switches allow multiple devices to be connected

to a single PCIe port.

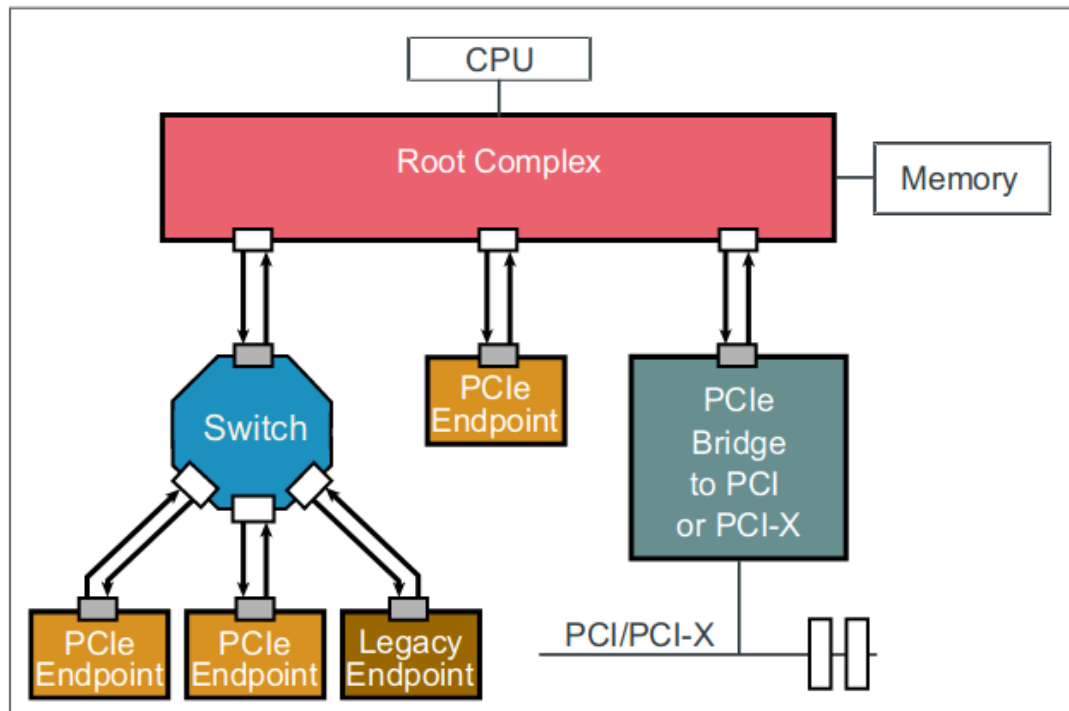


Figure 2.3: A typical PCIe based system [21].

In theory, they work as packet routers by examining incoming data packets and addresses. Bridges provide the connection between PCIe and other bus structures. Endpoints are the most basic PCIe devices that can work as terminators or initiators in a data transfer, and they have a simpler structure compared to the Root Complex.

2.1. The Operation of PCIe

PCIe can be classified as a packet-defined protocol. Data received from the software in accordance with the PCIe format is divided into different traffic channels. It is classified according to its priorities and sent to the receiving device using insurance mechanisms against the corruption/failure of the sent data. The PCIe interface data path is divided into stages and a different function is implemented in each stage. These stages can be examined under 3 different headings:

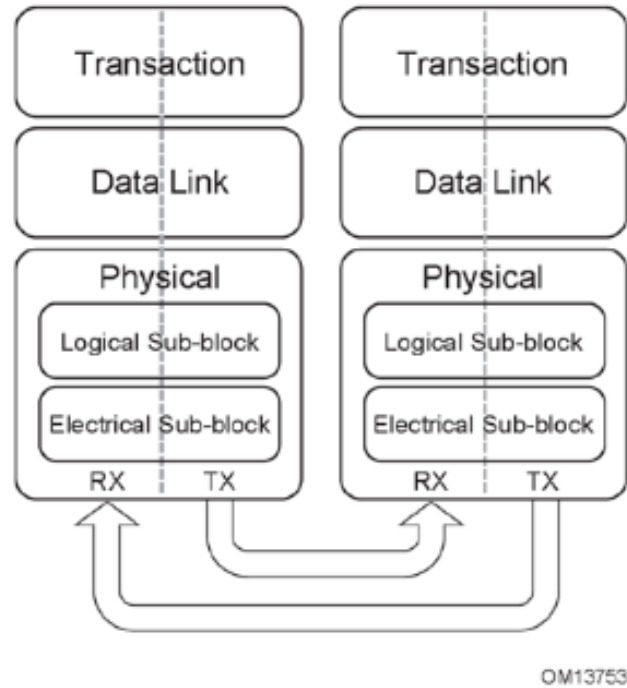


Figure 2.4: PCIe protocol stack [22].

2.1.1. Transaction Layer

It is responsible for the preparation and decoding of TLP packets, quality of service, flow control, and transaction ordering functions. Data packets arriving at the PCIe hardware via DMA are first received at this layer. A transaction is the collective name given to a packet that transmits a command to a target device and any completion packets that the target sends back in response. PCIe packets can be grouped into 4 main groups: memory transactions, IO transactions, configuration transactions and messages.

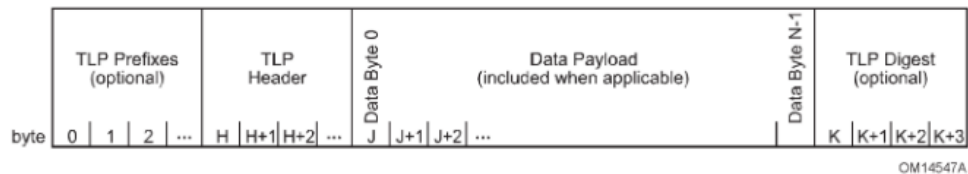


Figure 2.5: Sections of a transaction layer packet [22].

A transaction generated by software is reformatted as a transaction layer packet (TLP) in the transaction layer, and the same TLP packet ends up in the receiver's transaction layer. Memory, I/O, configuration TLPs, and messages pass through the data link layer

(DLL) and physical layer (PHY), respectively, to reach the neighboring device. Once generated at the transaction layer, TLPs add fragments to the packet as it moves between the Data Link Layer (DLL) and Physical Layer (PHY), and then verify at the receiver that these fragments were correctly transmitted across the Link.

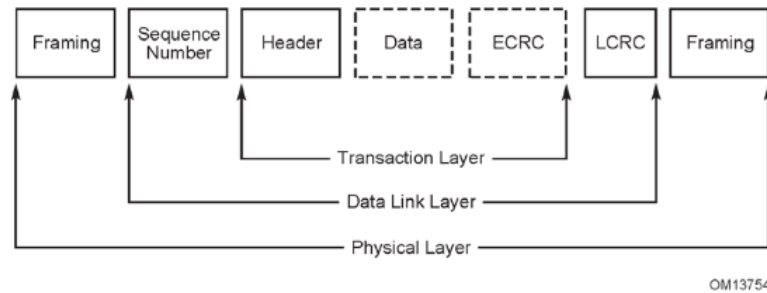


Figure 2.6: Extensions of a TLP.

In addition to the production and transfer of Transaction Layer Packets (TLPs), the data transfer operations at the protocol level encompass several other critical tasks, such as Quality of Service (QoS), Transaction Ordering, and Flow Control. These tasks collectively enhance the efficiency, reliability, and performance of data transactions within the PCIe architecture.

Quality of Service (QoS) in PCIe is designed to manage the prioritization of data traffic, ensuring that critical data packets receive the necessary bandwidth and low latency for optimal performance. QoS mechanisms achieve this by categorizing traffic into 4 different classes and applying specific policies to each class to guarantee a certain level of performance. This is particularly important in scenarios where multiple types of traffic coexist, such as real-time video streaming, bulk data transfers, and high-priority control signals.

The implementation of QoS involves the use of various techniques, such as traffic shaping, admission control, and packet scheduling. For instance, traffic shaping regulates the flow of packets to prevent congestion and ensure smooth data transmission. Admission control mechanisms decide whether new traffic flows can be accommodated without degrading the performance of existing flows. Packet scheduling ensures that high-priority packets are transmitted promptly, while lower-priority packets may be

delayed during periods of congestion [23]. The contribution of QoS to the PCIe protocol is significant as it allows for the efficient and reliable handling of diverse data streams, ensuring that performance-critical applications can function without interruption. QoS mechanisms help maintain the overall quality and reliability of the network by minimizing packet loss, jitter, and latency.

In addition to QoS protocol, PCIe also increases system performance by performing Transaction Ordering. Transaction Ordering in PCIe refers to the rules and mechanisms that ensure the correct sequence of data packets during transmission and reception. This is crucial for maintaining data consistency and integrity, particularly in complex systems where multiple transactions may be occurring simultaneously. Transaction Ordering ensures that dependent transactions are processed in the correct order, preventing data corruption and ensuring that operations are completed accurately. PCIe employs various ordering rules, such as the strict ordering rule, relaxed ordering rule, and ID-based ordering. These rules dictate how packets should be handled concerning other packets in the system. For example, strict ordering requires that packets are processed in the exact order they are issued, while relaxed ordering allows for some reordering to improve performance without compromising data integrity [24]. The contribution of Transaction Ordering to the protocol is primarily in enhancing the speed and reliability of data transactions. By ensuring that packets are processed in the correct order, the system can avoid data conflicts and ensure that all operations are completed correctly and efficiently. This improves the overall performance and stability of the PCIe interface [7].

Another reason why the PCI Express protocol is a very fast interface is that it uses the Flow Control theory. Flow Control is a mechanism that manages the rate of data transmission between devices to prevent buffer overflow and data loss. This is achieved by regulating the flow of data packets based on the available buffer space and network conditions. Flow Control ensures that the sender does not overwhelm the receiver with more data than it can handle, thereby maintaining a smooth and efficient data transfer process.

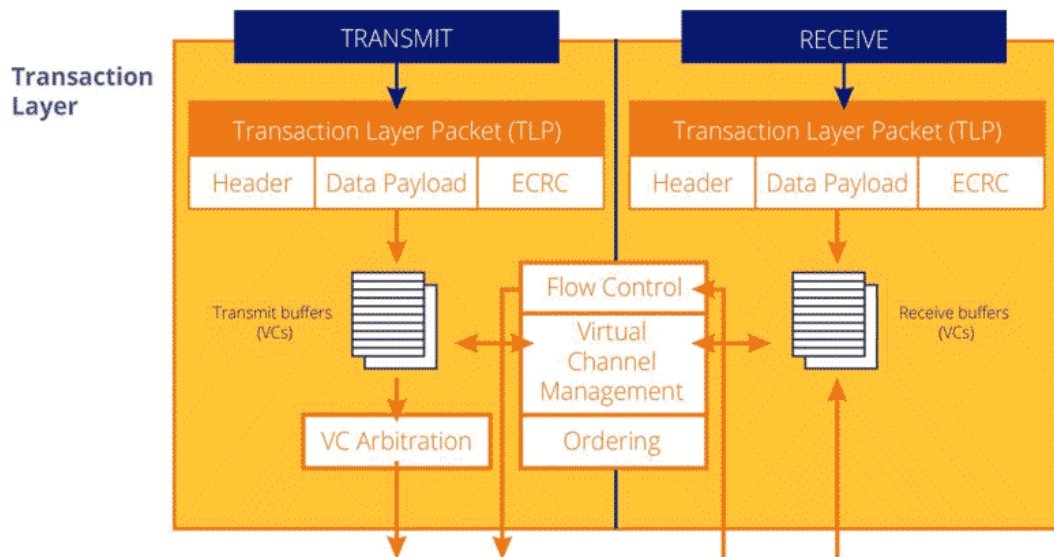


Figure 2.7: Transaction layer mechanisms : Flow control, Transaction Ordering and QoS [25].

PCIe employs two primary types of flow control: credit-based flow control and token-based flow control. In credit-based flow control, the receiver informs the sender of the available buffer space by issuing credits, allowing the sender to transmit data only when it has sufficient credits. Token-based flow control uses tokens to manage the data transmission rate, with each token representing permission to send a certain amount of data [26]. The absence of flow control mechanisms would lead to severe issues such as buffer overflows, increased packet loss, and overall network instability. Without flow control, the sender might transmit data at a rate faster than the receiver can process, leading to data corruption and loss. Flow control ensures that data is transmitted at a manageable rate, preserving the integrity and performance of the network. The three mechanisms of Transaction Layer can be seen on Fig. 2.7.

Flow control works harmoniously with the Data Link Layer (DLL) in PCIe. The DLL manages transmission of data link layer packets including flow control credits. FC counters is located in Transaction Layer and Data Link Layer is responsible for sending the initialization type Flow Control DLLPs and update type DLLPs. Together, they ensure that data is transmitted efficiently and accurately, maintaining the overall health and performance of the PCIe network.

2.1.2. Data Link Layer

Data Link Layer is generally responsible for link management and performs three main functions: TLP error correction, flow control and link power management. DLLPs are used to perform the three main functions mentioned. DLLPs are transferred between the Data Link Layers of two neighboring devices on a link and travel only between neighboring devices and are not directed anywhere else. In other words, a DLLP packet is generated at the DLL layer and consumed at the DLL layer of the opposite device. The DLL architecture is comprised of various components, primarily the Transmit (Tx) and Receive (Rx) channels, along with control blocks. These components work together to ensure the reliable delivery of data packets.

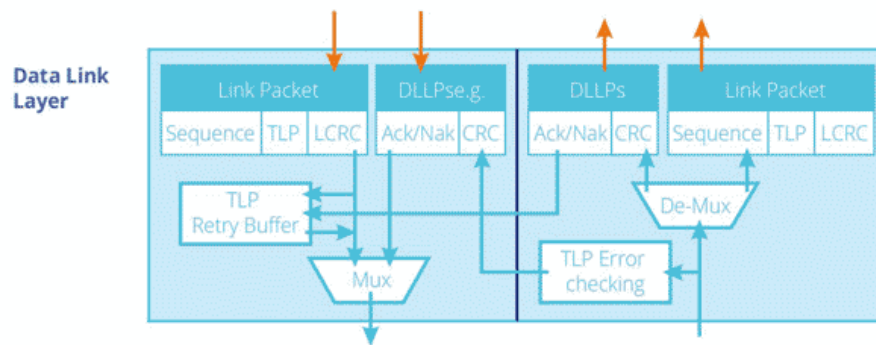


Figure 2.8: Data link layer [25].

Transmit Channel: Handles outgoing data by taking TLPs from the Transaction Layer, adding necessary headers, and calculating CRC. The calculated CRC is appended to the TLP, forming a DLLP, which is then transmitted to the receiver.

Receive Channel: Manages incoming data, verifying the CRC to ensure data integrity. If errors are detected, corrective measures are taken, including requesting retransmission if necessary.

Control Blocks: These blocks manage the state and operations of the DLL, including error handling, flow control, and communication with the Physical Layer.

A critical feature in the Tx channel is the retry buffer, which temporarily stores TLPs until an acknowledgment is received from the receiver. If a packet is lost or an error is

detected, the retry buffer allows the Tx channel to retransmit the affected packet without reprocessing it from the Transaction Layer, thus improving efficiency and reliability [7].

DLLPs are specialized packets generated by the Tx channel and sent to the receiver's DLL. These packets handle various control functions within the DLL. There are several types of DLLPs, including:

- Acknowledgment DLLPs: Used to confirm the receipt of data packets.
- Flow Control DLLPs: Manage the flow of data by indicating the available buffer space.
- Power Management DLLPs: Handle power management features, such as putting the link into a low-power state.

Flow Control DLLPs are critical for maintaining efficient data transfer and avoiding congestion. Init FC DLLPs provide information on the current buffer status of the receiver. Update FC DLLPs update the sender on the amount of buffer space freed up at the receiver. These DLLPs work closely with the state machine to dynamically adjust data flow based on real-time conditions, ensuring smooth and efficient communication [27].

Like TLPs, DLLPs are protected against corruption by CRC. Every DLLP and TLP is appended with a CRC value to detect errors during transmission. If the CRC check fails, corrective actions such as packet retransmission are initiated. The use of CRC significantly enhances data reliability, though it does introduce additional computational overhead and power consumption. Despite this, its benefits in ensuring error-free communication make it indispensable in PCIe architecture [28].

When data packets (TLPs and DLLPs) arrive at the receiver, the first step is to perform integrity checks. These checks involve verifying the CRC values appended to the packets during transmission. If a packet passes the integrity check, it is considered error-free and is processed accordingly. For DLLPs, this means routing them to be consumed in the DLL to further processing. For TLPs, it involves stripping off any additional headers added by the DLL before passing the data up to the Transaction Layer [7]. If a TLP fails the integrity check, a negative acknowledgment (Nak) DLLP must

be sent back to the transmitter. This is accomplished by invoking the logic in the Tx channel responsible for generating DLLPs. The Nak DLLP informs the sender that the specific TLP was received with errors, prompting the sender to retransmit the faulty TLP. This mechanism ensures that data corruption does not go uncorrected, maintaining the integrity of the communication channel. On the other hand, if a DLLP fails the integrity check, it is simply discarded. This is because DLLPs typically handle control functions and are not as critical as TLPs in terms of data content. Discarding erroneous DLLPs helps in preventing the propagation of control errors throughout the system.

When a Nak DLLP is received by the receiver, it triggers the retransmission of the TLP identified by the Nak. The Tx channel relies on the retry buffer to manage this process. The retry buffer stores TLPs until an acknowledgment (Ack) DLLP is received, confirming successful reception by the receiver. Upon receiving a Nak, the transmitter retrieves the specified TLP from the retry buffer and retransmits it. If an Ack DLLP is received, it indicates that the corresponding TLPs have been successfully received and acknowledged. The Ack DLLP references the TLPs by their sequence numbers. Consequently, the retry buffer can then purge these acknowledged TLPs, freeing up space for new data. This acknowledgment process is vital for ensuring reliable data transfer and efficient use of buffer space [29].

The Data Link Control and Management Status State Machine (DLCMSM) plays a central role in managing the operations of the Rx channel. It monitors the status of incoming TLPs and DLLPs, handling error conditions and coordinating the actions required to maintain data integrity. For instance, when a Nak DLLP is received, the DLCMSM directs the retry buffer to retransmit the specified TLPs. Similarly, it updates the state machine upon receiving an Ack DLLP, ensuring that the system is aware of the successfully transmitted and acknowledged packets. The DLCMSM also manages the flow control, ensuring that data is transmitted and received at a rate that prevents buffer overflow and maintains efficient data flow. By keeping track of the status of each packet, the DLCMSM ensures that the data link layer operates smoothly and reliably.

2.1.3. Physical Layer

The Physical (PHY) layer is the lowest layer in the hierarchy and is responsible for generating and decoding Ordered Sets (OS). It processes TLP and DLLP received from upper layers along with OS, and transmits them to the receiving device. The PHY layer incorporates clock and data recovery (CDR) circuits for differential data reception, including circuits for clock and data recovery, and manages the Link Training and Status State Machine (LTSSM) responsible for link initialization and settings. This layer uniquely combines both analog and digital blocks, playing a critical role in ensuring the integrity and proper transmission of data across the network. It manages low-level signal transmission and reception tasks, utilizing both analog and digital circuits to handle the various demands of data communication. By processing ordered sets along with TLP and DLLP packets, it ensures synchronized and error-free data transfer.

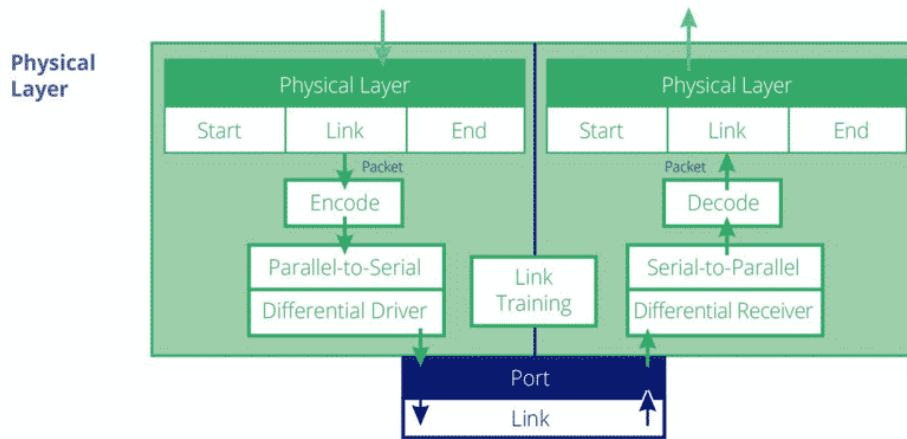


Figure 2.9: PCIe physical layer architecture [25].

In the PHY layer, various packet types are employed, including TLP and DLLP, which are crucial for transaction and data link layer communications respectively. The ordered sets (OS) are also processed here to maintain the proper sequencing of data and control information, ensuring coherent communication. The LTSSM within the PHY layer is responsible for managing the states involved in initializing and maintaining the link. It controls the transition between different states required for establishing a stable and efficient connection, ensuring that the link parameters are correctly negotiated and set.

The Link Training and Status State Machine (LTSSM) comprises several states, each playing a specific role in managing the link. The Detect state involves the initial detection of a potential link partner and validation of the presence of a connection. The Polling state focuses on configuring the link and includes sub-states like Polling.Active and Polling.Config, where the link is tested and parameters are exchanged to ensure compatibility. The Configuration state finalizes the link parameters and prepares for data transmission. This includes setting the appropriate link width and speed.

The L0 (L0s and L0p) states represent the normal operational state where data transmission occurs. L0s is a low-power state that reduces power consumption while maintaining a ready state for quick resumption of full operation. The Recovery state is used to re-establish synchronization if any errors are detected or if there's a need to change link parameters. The L1 and L2 states are low-power states with L1 being a standby mode and L2 representing a deeper sleep mode with significant power savings.

The L3 state is the lowest power state, effectively shutting down the link, requiring a full reinitialization to return to operational state. The Hot Reset state allows for a quick reset of the link, useful in scenarios where the link needs to be reinitialized without a complete power cycle. The Loopback state is used for testing the integrity of the data path by routing transmitted data back to the transmitter for verification.

Lanes are individual data transmission pathways within the PHY layer, allowing parallel transmission of data. Lane to lane de-skew is a technique used to align data that arrives at slightly different times across multiple lanes, ensuring that the data is properly synchronized and can be accurately processed. Scramblers are used in the PHY layer to randomize the data being transmitted, reducing the likelihood of signal interference and improving signal integrity. They are critical for maintaining robust communication across different generations, from Gen1 to Gen6, ensuring backward compatibility and forward efficiency.

The PHY Interface for the PCI Express (PIPE) standardizes the connection between the PHY and the Media Access Control (MAC) layer, facilitating smooth communication between these layers. It defines the physical link characteristics and the protocol for data exchange. Encoders and decoders in the PHY layer convert data between its digital format and the encoded format suitable for transmission. These components ensure that

data integrity is maintained across different generations of the PCI Express standard, from Gen1 to Gen6.

Encoders and decoders are essential in the PHY layer to translate data into a form that can be effectively transmitted and received over the physical medium. For Gen1 and Gen2, the 8b/10b encoding scheme is employed. This scheme translates 8-bit data into 10-bit encoded data, ensuring sufficient signal transitions for clock recovery and providing error detection capabilities [30] Gen3 onwards uses a more efficient 128b/130b encoding scheme, which significantly reduces the overhead to just 1.5%, allowing higher data rates with better bandwidth utilization. This scheme translates 128 bits of data into 130 bits, incorporating alignment markers to help maintain data integrity over high-speed links [31].

Gen5 and Gen6 continue to build on this foundation with even more advanced encoding schemes. Gen5 employs 128b/130b encoding with additional optimizations to support data rates up to 32 GT/s. This generation also includes enhanced equalization techniques to maintain signal integrity at these higher speeds [32]. Gen6, on the other hand, adopts PAM-4 (Pulse Amplitude Modulation 4-level) signaling to achieve data rates up to 64 GT/s. PAM-4 encoding increases the number of bits per symbol, effectively doubling the data rate without increasing the bandwidth. This method, however, introduces new challenges such as increased susceptibility to noise and signal degradation, which are addressed through advanced equalization and error correction techniques [33].

Elastic buffers in the PHY layer absorb timing variations between the transmitting and receiving clocks. They help in smoothing out any discrepancies, ensuring that the data is delivered consistently and without error. PLLs are crucial in the PHY layer for generating stable and synchronized clock signals from a noisy input. They ensure that the timing of the data signals is precise, which is vital for high-speed data communication. CDRs recover clock and data signals from the incoming differential signals. They play a vital role in maintaining the integrity of the received data by accurately extracting the timing information.

Equalizers in the PHY layer compensate for signal degradation over long distances or through various mediums. They adjust the amplitude and phase of the received signals to restore the original signal quality. PAM-4 is a modulation technique used in high-

speed communication to increase the data rate by encoding two bits per symbol. This technique allows for more efficient data transmission by effectively doubling the data rate without doubling the bandwidth. SerDes components convert parallel data into serial data for transmission and then back into parallel data for reception. They are essential for high-speed data communication, enabling efficient and reliable transmission over various distances.

As can be inferred from a forementioned material, PCIe is a very heavy-weight and complicated communication protocol. Although it consists of fundamentally three main layers, it is composed of dozens of sub-blocks. Being designed as a high-speed communication protocol, it is inherently power-hungry. To manage power consumption, numerous methods are applied within this protocol. Any power improvement in a specific part of this protocol can enable significant power savings in a PC system.

The main focus of this work is power/performance optimization of CRC mechanisms located in DLL. While continuing on these efforts, a more performance-efficient implementation of the CRC mechanisms already has been presented in the specification of Gen7 [20]. Additionally, emerging topics were explored, and some CRC implementations were proposed using Lightweight Cryptographic Algorithms (LWCs).

3. PCIE IP CORE DESIGN

The integrity mechanisms planned to be developed in this study are generally located in the Data Link Layer. For this reason, the Data Link Layer of the PCI Express Controller block has been implemented.

In the implementation of integrity mechanisms within PCI Express Controllers, the Data Link Layer plays a crucial role. The DLL is primarily responsible for managing the link between devices and ensuring the reliable transmission of data. It achieves this through three main functions: Transaction Layer Packet error correction, flow control, and link power management. To accomplish these tasks, the Data Link Layer utilizes Data Link Layer Packets. DLLPs facilitate the communication of flow control information and power management signals between devices, ensuring that the data transmission remains efficient and reliable. Additionally, the DLL acts as an intermediary, carrying TLPs between the Transaction Layer and the Media Access Control (MAC) layer. Data Link Layer architecture will be examined under 3 main groups. Control blocks, Rx data path and Tx data path.

3.1. Control Blocks

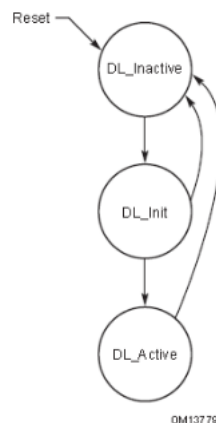


Figure 3.1: DLCMSM States [21].

The Data Link Control and Management Status State Machine (DLCMSM) is respon-

sible for managing the transmission of Transaction Layer Packets and Data Link Layer Packets, as well as the overall management of the link in PCI Express. This state machine ensures that data transmission is reliable and that flow control is properly maintained. As it can be seen on Fig. 3.1 the DLCMSM operates in four states: DL_inactive, DL_init1, DL_init2, DL_active.

DL_init state has 2 substates named as init1 and init2. Each with specific roles and transitions based on the link status and signals received.

DL_inactive: In this state, no data traffic occurs within the DLL. The link is inactive, and neither TLPs nor DLLPs are being transmitted or received.

DL_init1: When the MAC layer raises the phy_linkup signal, indicating that the physical layer is ready, the DLCMSM transitions from DL_inactive to DL_init1. In this state, the DLL begins receiving flow control (FC) credit values from the Transaction Layer. These FC credits are crucial for managing the data flow and are transmitted to the corresponding device using DLLPs.

DL_init2: After transitioning from DL_init1, the DLCMSM enters the DL_init2 state. In this state, the exchange of FC credits continues using Init2_FC DLLPs. This exchange ensures that both devices are synchronized and ready for data transfer.

DL_active: Once both types of initialization FC DLLPs have been sent and received, the DLL raises the DLup signal, indicating that the link is fully established and active. The DLCMSM then transitions to the DL_active state. In this state, active data transmission and reception occur. The DLL utilizes two data channels: the Rx data channel for receiving data and the Tx data channel for transmitting data.

During the DL_init and DL_active states, the DLCMSM ensures the seamless flow of data by managing the transmission and reception processes. This includes handling the FC credits to maintain a balance between the amount of data being sent and the receiving device's capacity to process it. The Data Link Control and Management Status State Machine is the fundamental block responsible for managing traffic within the Data Link Layer of PCI Express. In addition to the DLCMSM, the DLL includes Transmit and Receive data channels.

3.2. Tx Path of the Data Link Layer

As shown in Fig. 3.2, the transmitter channel includes packet generator modules, a retry mechanism, and packet transmission hardware. The module that first receives the Transaction Layer Packets from the Transaction Layer is the packet generator module.

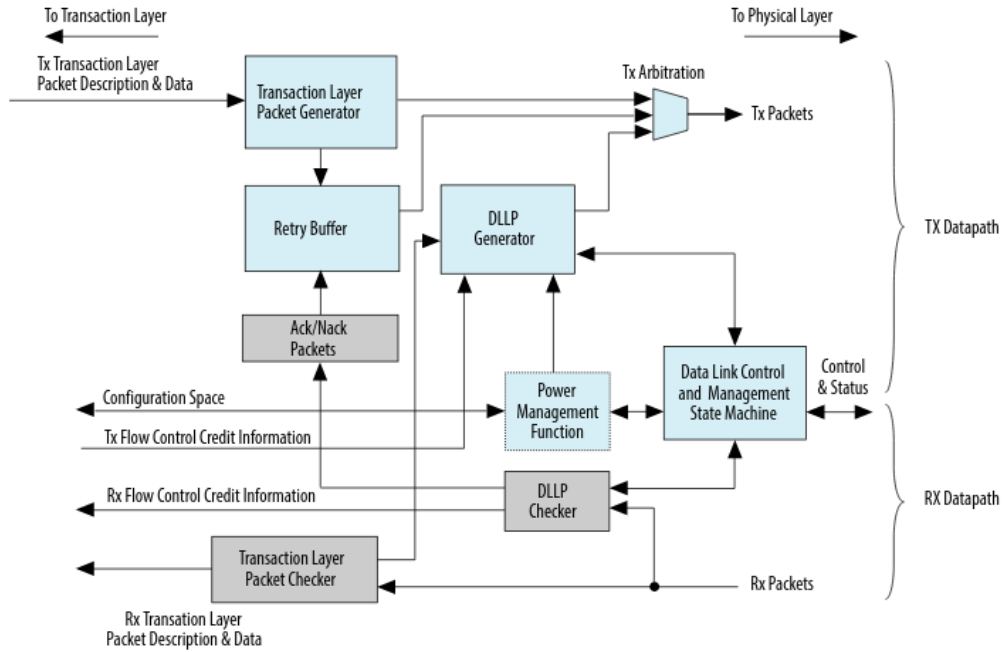


Figure 3.2: Data link layer micro architecture [21].

The Transaction Layer Packet generator module shown in (Fig. 3.3) firstly generates a tag number called the Next Transmit Sequence (NTS) for each received TLP. This tag number is assigned to each transmitted TLP. To facilitate this, the module maintains an NTS counter. Once the NTS is appended, the TLP enters the enhanced Link Cyclic Redundancy Check module, where a 32-bit LCRC computation is performed. The calculated LCRC value is then appended to the tail of the TLP, completing the TLP packet generation process. Both the NTS tagging and the CRC calculation processes are accomplished within a single clock pulse. The TLPs, after LCRC computation, are issued to the Media Access Control layer. Meanwhile, each transmitted TLP is also stored in the retry buffer. The retry buffer operates similarly to a FIFO (First In, First Out) buffer, but its read and write pointers are configured differently. It is designed as a FIFO-like structure capable of retransmitting every TLP sent to the MAC layer.

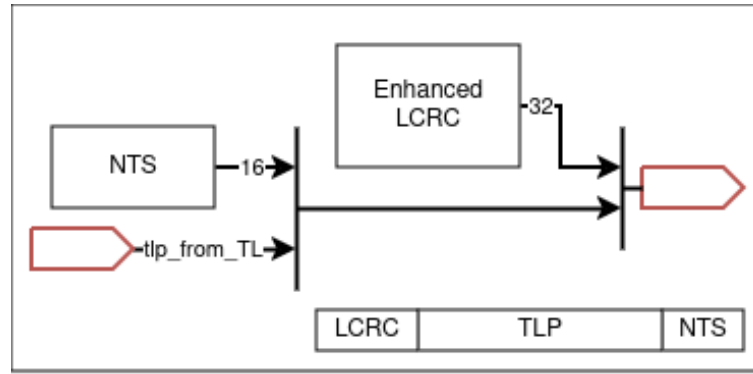


Figure 3.3: Transaction layer packet generator.

The retry buffer contains registers that hold the addresses of the read minimum and read maximum pointers. If an Acknowledge message is received for each transmitted TLP, the read minimum and read maximum pointers are equal, indicating that the memory is empty. If a TLP has been sent to the MAC but an Acknowledge message has not been received, the read minimum pointer records the index indicating the Start of Frame for the stored TLP. Similarly, the index holding the End of Frame for the same TLP is recorded at the read maximum pointer address. During any retry TLP operation, the read pointer issues read requests to all indexes between the read minimum and read maximum addresses, resending the stored TLPs to the MAC layer.

By incorporating these mechanisms, the system ensures reliable transmission of data packets, minimizing the chances of data loss and enhancing the overall efficiency of the data link layer. The packet generator, LCRC computation, and retry buffer work together seamlessly to handle TLPs efficiently, ensuring data integrity and successful communication across the layers.

The transmission path in the Data Link Layer not only handles the packaging of Transaction Layer Packets but also takes on the responsibility of Data Link Layer Packet generation. The DLLP generator within the Tx path produces various types of DLLPs, including Acknowledge/Negative Acknowledge (Ack/Nak) DLLPs, Flow Control DLLPs, and Power Management (PM) DLLPs. Before the link becomes fully operational, it is crucial to send FC initialization DLLPs. To achieve this, the FC buffers in the Transaction Layer provide credit limit information, which is then used to generate FC init1 and FC init2 DLLPs.

Once the link is operational, there is an ongoing need to generate FC DLLPs. Because the available space in the TL's FC buffers can vary during runtime, it is necessary to periodically send FC update DLLPs to the connected device. These update FC DLLPs are also generated based on the FC credits obtained from the TL.

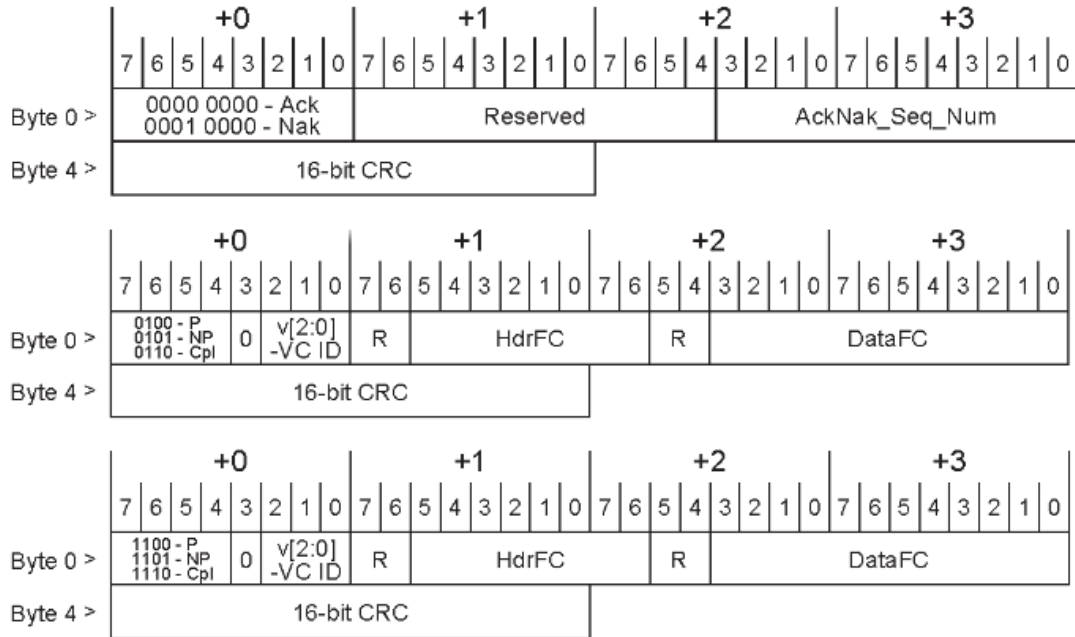


Figure 3.4: PCIe Data Link Layer packet formats.

Additionally, the DLLP generator module produces power management DLLPs, which are responsible for the power management of the link. It also generates Nak DLLPs to report any erroneously received TLPs, and Ack DLLPs to confirm the receipt and processing of TLPs without any errors. The encoding details for some of these DLLPs are provided in Fig. 3.4. The process of generating these different types of DLLPs is integral to maintaining efficient and reliable communication over the PCI Express interface. By ensuring proper flow control and power management, the system can operate smoothly and respond appropriately to different states and conditions of the data link. According to the decoding shown in Fig. 3.4, the generated DLLPs undergo a CRC calculation process. Each DLLP is subjected to a 16-bit CRC computation to ensure data integrity. Once the CRC is calculated, these DLLPs are transmitted to the MAC layer.

To manage the timing and priority of data packets being sent to the MAC layer, the Tx arbiter module is implemented. This module plays a critical role in determining which data packet should be sent to the MAC layer at any given time. The Tx arbiter makes this decision based on feedback from various components within the Data Link Layer. By evaluating these inputs, the arbiter prioritizes data packets and ensures their orderly transmission to the MAC layer. The decision-making process of the Tx arbiter considers three primary factors:

- **Timeout Mechanisms for DLLP Packets:** The Data Link Layer includes timeout mechanisms for DLLP packets. If a timeout occurs, generating an interrupt, the affected DLLP packet must be prioritized and sent to the MAC layer immediately. This ensures that critical DLLP packets are transmitted without delay, maintaining the reliability and efficiency of the communication link.
- **Completion of Ongoing Packet Transmission:** The Tx arbiter ensures that once the transmission of a TLP or DLLP packet has begun, it must be completed before initiating the transmission of another packet. This rule prevents the interruption of ongoing transmissions, thereby avoiding data corruption and ensuring that each packet is fully and correctly transmitted before moving on to the next one.
- **Frequent Transmission of Ack, Nak, and Update DLLPs:** To prevent operational stalls, the Tx arbiter prioritizes the frequent transmission of Ack, Nak, and Update DLLPs. These packets are crucial for maintaining the flow of data and communication integrity. Ack and Nak DLLPs provide acknowledgment of received packets and notifications of errors, respectively, while Update DLLPs convey important flow control information. Regular transmission of these packets ensures smooth and continuous operation of the data link.

By adhering to these considerations, the Tx arbiter effectively manages the flow of data packets within the Data Link Layer, ensuring efficient and orderly communication with the MAC layer.

3.3. Rx Path of the Data Link Layer

In the receiver channel, when the Data Link Layer receives DLLPs it must verify the integrity of these packets to ensure error-free communication. This involves performing the 16 bit CRC check. If the LCRC check reveals an error, the DLLP is discarded, and an error flag is set. This prevents corrupted control data from affecting the communication link's reliability. Upon successful integrity validation, the DLLPs are decoded based on their specific types. Each type of DLLP serves a different function within the PCIe protocol.

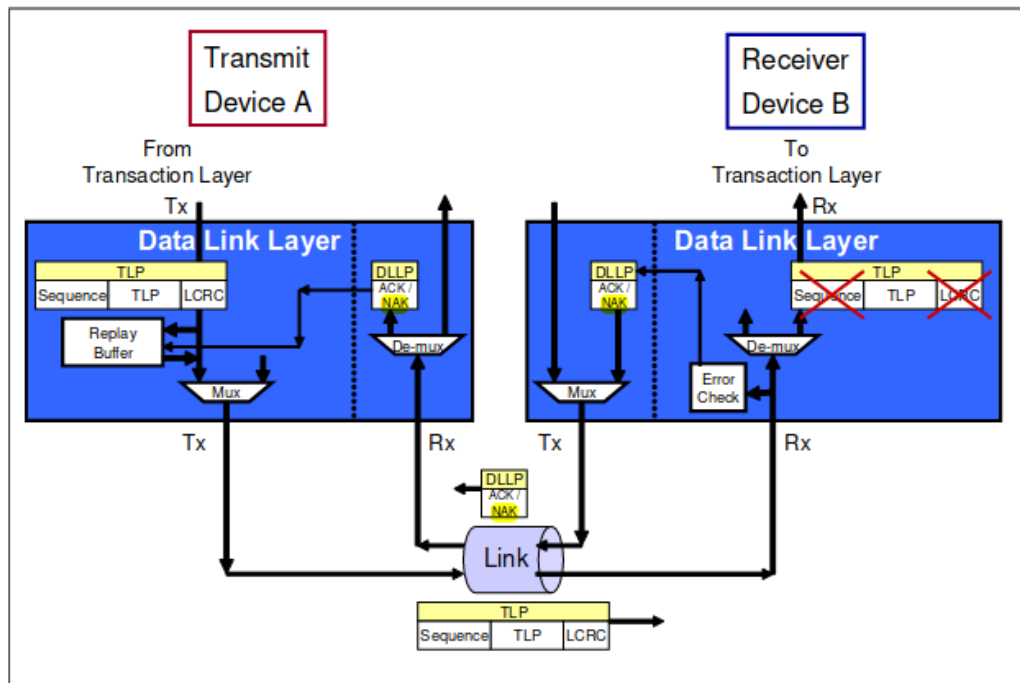


Figure 3.5: Overview of the Ack/Nak protocol [21].

FC DLLPs manage the flow control between devices. These packets contain header and data credit values, which are extracted and sent to the transaction layer to regulate the flow of TLPs, ensuring that the sender does not overwhelm the receiver. On receiving an FC DLLP, the extracted header and data credit values are passed to the transaction layer. This ensures that the data flow remains controlled and efficient, preventing buffer overflows and underflows.

Ack DLLPs indicate successful receipt of TLPs. The sequence number in the Ack

DLLP is used to update the retry buffer (see Fig. 3.5), confirming which TLPs have been successfully received and can be removed from the buffer. For Nak DLLPs, the sequence number triggers the retransmission of the specified TLPs from the retry buffer, ensuring data integrity and reliability. PM DLLPs are used for managing the power states of the link. These packets instruct the receiver to transition the link to a different power state, optimizing power consumption without compromising the link's performance. This involves transitioning to a lower power state when the link is idle or to a higher power state when data transmission resumes.

Unlike Data Link Layer Packets, which are fully processed by the DLL, TLPs are treated as a "black box." This means that the contents of TLPs are not interpreted or modified by the DLL. Instead, the DLL focuses on ensuring the integrity and proper alignment of TLPs before passing them on to the Transaction Layer. The DLL performs a final check to ensure that the TLP has not been corrupted during transmission. This check ensures that only valid and intact TLPs are forwarded to the Transaction Layer. The aligned and verified TLP is then sent to the Transaction Layer for further processing. The Transaction Layer interprets and processes the TLP, performing the necessary actions based on the data contained within the packet.

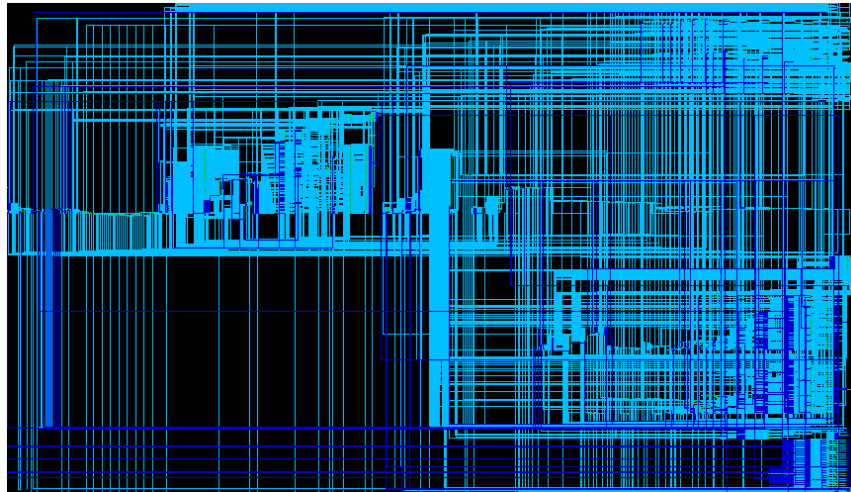


Figure 3.6: Synthesis schematic of our PCIe DLL in Synopsys DC-NXT.

All of the Rx, control and Tx blocks described above was modeled using the System Verilog HDL language and synthesized using the Synopsys DC-NXT tool using TSMC

16 nm technology. The enhanced LCRC hardware, which is the subject of this publication, was used in the synthesized hardware. The obtained results are as follows:

Table 3.1: The cell count of our DLL.

Cell Count	
Hierarchical Cell Count	24
Hierarchical Port Count	3120
Leaf Cell Count	357063
Buf/Inv Cell Count	33977
Buf Cell Count	22332
Inv Cell Count	11645
CT Buf/Inv Cell Count	0
Combinational Cell Count	247422
Sequential Cell Count	109641
Macro Count	0

Table 3.2: The area of our DLL.

Area	
Combinational Area	69267.519305
Noncombinational Area	125057.057980
Buf/Inv Area	6136.041705
Total Buffer Area	4765.75
Total Inverter Area	1370.29
Macro/Black Box Area	0.000000
Net Area	0.000000
Net XLength	3451694.75
Net YLength	3259680.50
Cell Area	194324.577286
Design Area	194324.577286
Net Length	6711375.00

For the 2.5 ns constraint, there is no path that violates the WNS in the setup and hold paths. The delay of the path with the worst timing is reported as 2.42 ns.

The schematic visual of our custom Data Link Layer is depicted in Fig. 3.6. As illustrated by the schematic and corroborated by various reports, although only a single layer is implemented, it occupies a significant area. The considerable number of logic cells required for just one layer underscores the power-hungry nature of the PCI Express interface. Therefore, optimizing every hardware block within this structure is imperative. In light of this, a comprehensive study was conducted to explore whether more efficient hardware could replace the existing improved LCRC hardware utilized in the synthesized system. This study aimed to identify and implement alternative hardware solutions that could potentially enhance performance and reduce power consumption. The goal was to streamline the Data Link Layer, ensuring it operates more efficiently while maintaining or even improving its functionality. The findings indicated that several hardware options could provide better efficiency compared to the current LCRC hardware. These alternatives were evaluated based on various parameters such as power consumption, processing speed, and overall footprint within the system. The research suggested that implementing these new hardware solutions could significantly improve the energy efficiency of the PCI Express interface. Moreover, it is crucial to continually refine the hardware blocks to keep up with the evolving demands of technology and to ensure that the interface remains robust yet efficient. Future work will focus on further optimizing these hardware components and exploring innovative solutions to reduce the power consumption of the PCI Express interface without compromising its performance. By adopting these improvements, the system can achieve a more balanced trade-off between power consumption and performance, ultimately leading to a more sustainable and efficient operation. Such advancements are essential for the continued development and optimization of high-performance computing systems that rely on the PCI Express interface.

Table 3.3: The power consumption details of our DLL.

Power Group	Internal Power	Switching Power	Leakage Power	Total Power (%)
io_pad	0.0000	0.0000	0.0000	0.0000 (0.00%)
memory	0.0000	0.0000	0.0000	0.0000 (0.00%)
black_box	0.0000	0.0000	0.0000	0.0000 (0.00%)
clock_network	133.8396	0.0000	0.0000	133.8396 (99.68%)
register	1.3474e-02	9.9582e-03	6.9365e+03	3.0374e-02 (0.02%)
sequential	13.0684	0.0018617	6934.4	13.0772 (75.95%)
combinational	0.1575	0.2341	9.0293e+03	0.4007 (0.30%)
Total	134.01 mW	0.24 mW	15.96 uW	134.27 mW

4. ALTERNATIVE APPROACHES FOR DATALINK INTEGRITY

Link Cyclic Redundancy Check is a mechanism used for error checking and detection during data transmission. It ensures the integrity of data frames by incorporating a Cyclic Redundancy Check code. The primary function of LCRC is to detect any corruption in the received data and ensure the data sent remains intact.

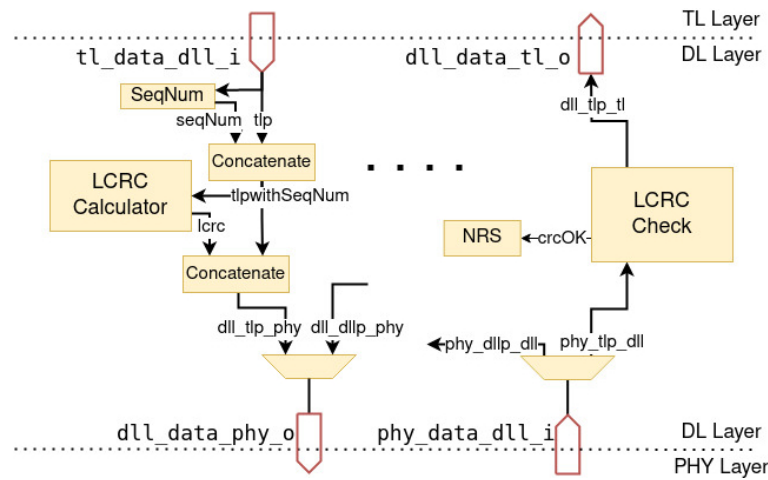


Figure 4.1: LCRC mechanism in PCIe Data Link Layer.

As it can be seen in Fig. 4.1, when Transaction Layer Packets come from the PHY layer, they are processed through the LCRC calculation mechanism for an integrity check. If the CRC code in the incoming TLP matches the calculated LCRC, the TLP is considered intact. The Next Receive Sequence is then incremented by one, and the incoming TLP is directed to the Transaction Layer. If a mismatch is detected between the calculated LCRC and the CRC in the TLP, the error is reported, and the incoming TLP is discarded without being forwarded to the TL.

Initially, a 12-bit Sequence Number is added to the TLPs received from the TL. This TLP, concatenated with the Sequence Number, is sent to the LCRC module. The calculated CRC value is queued with this data and transferred to the PHY Layer. The LCRC mechanism relies on an Linear Feedback Shift Register (LFSR) circuit with the

0x04C11DB7 polynomial. It processes 1-bit of input at a time and produces a 32-bit CRC code upon completion. Before starting, all flip-flops in the LFSR are set to binary 1. The LCRC mechanism operates in a slow clock domain, receiving 1 byte of data at a time. It maintains the CRC calculation by feeding 1-bit of the byte to the LFSR in an eight times faster clock domain, starting from the least significant bit. After completing the CRC calculation, all flip-flops of the LFSR are reset to binary 1.

4.1. Standard LCRC Mechanism in PCIe Specification

The standard LCRC mechanism in PCIe is essentially a serial Linear Feedback Shift Register circuit. This mechanism utilizes the polynomial 0x04C11DB7 for its computations. During each clock cycle, the LFSR processes the TLP data serially to generate a CRC code. Starting with an initial seed value of 0xFFFFFFFF, the LFSR operates by taking the input TLP data bit by bit, beginning with bit 0 of byte 0 (bit 8 of the TLP sequence number), and continues through each successive byte from bit 0 to bit 7 which can be seen in Fig. 4.2 This method ensures that the CRC value is computed sequentially, incorporating each bit of the data into the calculation. The resulting 32-bit CRC value is then appended to the end of the TLP, providing a robust means of verifying data integrity during transmission between Data Link Layers. This CRC code helps in detecting and correcting errors, thus ensuring the reliability of data transfer across the PCIe interface.

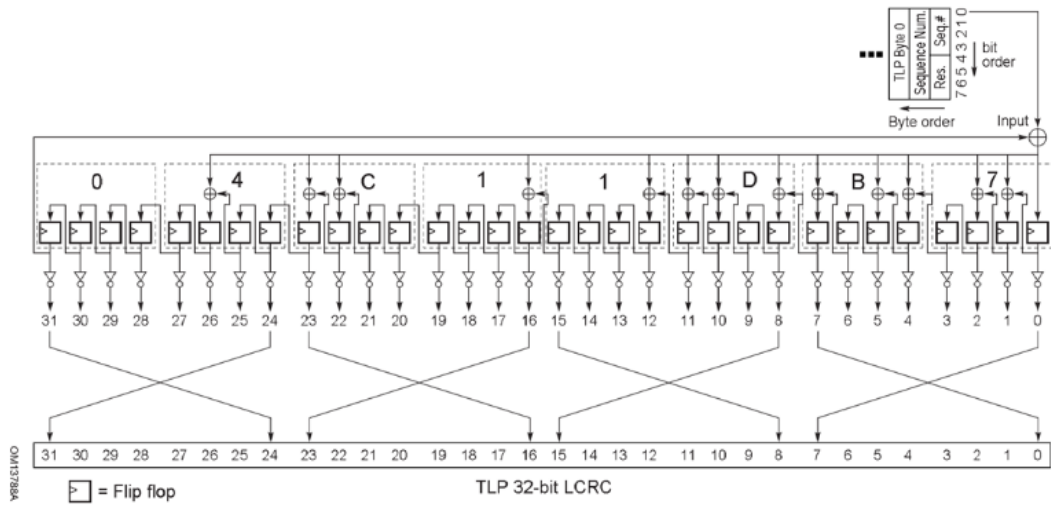


Figure 4.2: Calculation of LCRC [22].

However, to achieve high-speed and high-bandwidth data transfer, data paths must be configured in parallel. Consequently, if the standard LCRC implementation is used, the LCRC hardware would need to operate at a clock speed that is eight times higher than that of the Data Link Layer. This requirement can introduce synchronization problems, as the LCRC circuit must keep pace with the rapid data transfer rates. Such high-speed operation is particularly challenging in FPGA-based emulation environments, where the verification of designs becomes problematic. For instance, even at a data transfer rate of 2.5 GT/s, the LCRC hardware would need to function at a minimum clock speed of 2 GHz [34], assuming the controller is operating with a single lane. This clock speed is often beyond the capabilities of FPGA emulation, making it impractical to implement and verify the LCRC in such environments. Instead, while direct hardware implementation in emulation environments may not be feasible, the design can still be simulated to verify its functionality. For actual deployment, the design can then be synthesized and implemented using ASIC backend processes, which are capable of handling the required high-speed operations.

When the LCRC mechanism, as specified in the PCI Express specification [22], is designed in Verilog and implemented on the Xilinx KCU105 Board, the results obtained are shown in Table 4.1.

Table 4.1: Standart serial LCRC wrapper implementation results.

Parameter	Value
LUT	71
LUTRAM	0
FF	38
IO	108
BUFG	1
WNS (ns)	-3.546
Freq (MHz)	73.82
Power (W)	0.118
Junction Temperature (°C)	25.5

When the prototype LCRC circuit was implemented for the Xilinx Kintex Ultrascale XCKU040-2FFVA1156E FPGA device, it was found to operate at a maximum clock rate of 73.82 MHz, which is insufficient to meet PCIe throughput requirements for em-

ulation environment. To address this issue, parallel LFSR circuits could be employed, or the system could be designed for ASIC implementation, potentially achieving the desired frequencies. However, both solutions have their drawbacks. Higher frequencies result in increased dynamic power consumption, while using multiple LFSRs raises both area and static power consumption. Given the power-intensive nature of the PCIe interface, it is crucial to optimize each subsystem for power efficiency as well as performance. Therefore, various approaches are explored in this section.

4.2. Enhanced LCRC Mechanism

Implementing CRC using a Serial LFSR is often inefficient for certain architectures as it processes only one data bit per clock cycle due to serial data input [35, 36]. For an N -bit data bus, each clock pulse is required to process each bit serially in the CRC module [36]. Research into parallel CRC generation has shown that parallelizing the CRC circuit can enhance efficiency [37]. Furthermore, Perez's work on byte-wise CRC calculations provides practical insights into implementing parallel CRC circuits [38]. This paper's method of reducing power consumption relies on parallelizing the serial CRC circuit. The CRC32 polynomial used for data integrity in the PCIe Data Link Layer is given by [22]:

$$G(x) = 1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32} \quad (4.1)$$

The CRC output depends on the current state of the CRC and the data input. This polynomial can be represented as a matrix based on the input width and the CRC width for parallelization. To derive these matrices, the behavior of our parallel LFSR circuit must be modeled. MATLAB routines were created to advance the LFSR N times for any input width. Thus, the next CRC (M_{next}) state can be determined based on an N -bit input. Two matrices are generated: $H1$, defining the M_{next} matrix when $M_{current} = 0$, and $H2$, defining the M_{next} states when $N_{in} = 0$. For PCIe LCRC, $H1$ is an 8×32 matrix, and $H2$ is a 32×32 matrix.

After obtaining $H1$ and $H2$, the LCRC value is calculated using XOR combinations

of $LFSR_{current}$ and the parallel input. The parallel LCRC mechanism for PCIe LCRC can be expressed as:

$$LFSR_{next} = \text{transpose}(H1) \oplus \text{transpose}(H2) \quad (4.2)$$

New LCRC modules, optimized with different parallel input widths, were produced. The traditional LCRC structure is referred to as non-optimized. Results for the target Xilinx Kintex Ultrascale XCKU040-2FFVA1156E FPGA device show that power and energy consumption per bit decreases exponentially with increased input parallelism, stabilizing around 145 mW.

The proposed technique significantly reduces power and energy consumption per bit in the PCIe interface. Increased parallelism results in higher area costs but lowers power consumption per processed bit. For this implementation parallel input width of the enhanced LCRC is 64 bits and implementation results can be seen in Table 4.2.

Table 4.2: Enhanced LCRC implementation results.

Parameter	Value
LUT	245
LUTRAM	0
FF	33
IO	108
BUFG	1
WNS (ns)	-2.827
Freq (MHz)	77.96
Power (W)	0.141
Junction Temperature (°C)	25.6

In conclusion, our proposed LCRC technique enhances performance compared to the non-optimized version, which requires multiple clock domains and increases area and power consumption. Eliminating the need for CDC improves timing flexibility, boosting throughput and overall performance. Ensuring data throughput above PCIe specifications can further enhance reliability by reducing the sampling frequency for LCRC hardware usage.

4.3. Lightweight Cryptographic Algorithms as LCRCs

In addition, if we want to take the functionality of the LCRC hardware a little further, we can use other hardware developed for cryptological applications instead of the standard LCRC hardware. For example, hash algorithms such as SHA-2 and SHA-3 can be used. However, since these algorithms are heavy-weight algorithms, they come with disadvantages for high-throughput applications, which are the subject of this study. They also cause high power consumption. If another algorithm is to be used instead of LCRC, it should be able to reach higher speeds than LCRC or have lower power consumption. This also coincides with the need to develop hardware that provides low power consumption due to the increase in mobilization we offer.

In cryptographic applications, ensuring the security and efficiency of hardware implementations is critical, especially in resource-constrained environments such as IoT devices, RFID tags, and sensor networks. Lightweight cryptographic algorithms, designed to meet these stringent requirements, offer a promising alternative to traditional methods like Cyclic Redundancy Check. For this reason, in this study, the use and implementation of some lightweight crypto algorithms as CRC were compared. The first of these is ASCON LWC [39].

4.3.1. ASCON LWC

ASCON stands out for its robust security features and efficient performance. This paper provides an in-depth analysis of ASCON as a superior alternative to CRC, particularly in high-throughput and low-power applications. ASCON was developed during the CAESAR competition [40], aimed at identifying efficient and secure authenticated encryption schemes. ASCON's design features a 320-bit permutation, divided into five 64-bit register words, ensuring 128-bit security. This structure makes ASCON suitable for both encryption and hashing tasks. Its lightweight design is particularly advantageous for hardware implementations in resource-constrained environments, where efficiency and security are paramount. ASCON may offer several key advantages according to the literature. Abdulgadir et al. states that ASCON has been shown to consume significantly less power compared to other cryptographic algorithms. For instance, FPGA implementations reveal that ASCON has the lowest dynamic power

change per frequency increase among several NIST Lightweight Cryptography candidates. This characteristic makes ASCON highly suitable for applications where energy efficiency is crucial [41]. Kaur et al. claims ASCON’s design allows for high throughput operations, essential for real-time applications. Implementations of ASCON on various hardware platforms have demonstrated its capability to achieve high data processing rates while maintaining low power consumption. This makes ASCON an ideal choice for high-throughput applications [42] And Eryilmaz shows an implementation case of the ASCON core on an open source RISC-V processor [43].

Based on this information, it was also implemented an open source ASCON core [39] as LCRC mechanism. The implementation results with Xilinx Vivado IDE are as in Table 4.3.

Table 4.3: ASCON as LCRC implementation results.

Parameter	Value
LUT	734
LUTRAM	0
FF	373
IO	108
BUFG	1
WNS (ns)	0.352
Freq (MHz)	103.65
Power (W)	0.114
Junction Temperature (°C)	25.5

To be compared with ASCON128a, other LWC algorithms also implemented and the implementation results were evaluated in the following paragraphs.

4.3.2. GRAIN LWC

Grain-128AEAD is a stream cipher designed for use in resource-constrained environments such as IoT devices. It belongs to the Grain family of ciphers and supports Authenticated Encryption with Associated Data (AEAD), ensuring both the confidentiality and integrity of data. The cipher consists of two main components: pre-output generator and an authentication block. The pre-output generator includes a Linear Feedback

platforms such as ESP32 for generating One Time Passwords (OTPs) by Nicolas et al [46]. This method enhances security in IoT applications by utilizing the high-security features of Grain to produce unpredictable sequences of random numbers. Berbain et al. have developed key recovery attacks against Grain that can compromise its security with significant computational effort [47].

In this study, GRAIN was used as a simple hash function rather than security functions and it was evaluated whether it could be used instead of enhanced LCRC hardware. An open source Grain core [48] was used in the implementation and the GRAIN implementation results are given in the Table 4.4.

Table 4.4: GRAIN as LCRC implementation results.

Parameter	Value
LUT	2605
LUTRAM	0
FF	782
IO	108
BUFG	1
WNS (ns)	0.392
Freq (MHz)	104.08
Power (W)	0.28
Junction Temperature (°C)	26.3

To be compared with ASCON128a and GRAIN, another Trivium algorithm was also implemented and the implementation results were evaluated.

4.3.3. Trivium LWC

Trivium is a lightweight, hardware-oriented stream cipher designed by Christophe De Canniere and Bart Preneel for the European eSTREAM project [49]. It is known for its simplicity and efficiency in hardware implementations. Trivium has been subjected to extensive cryptographic analysis since its inception, making it a popular subject in the cryptographic community. This section includes the literature surrounding Trivium, its applications, and a detailed explanation of its working mechanism.

Maximov and Biryukov conducted studies on TRIVIUM-like designs and presented

state recovering and linear distinguishing attacks, suggesting that Trivium has a thin safety margin and proposing an improved design for enhanced security [50]. Hojsik and Rudolf applied differential fault analysis on Trivium, demonstrating how injecting faults can reveal the cipher's inner state and private key [51]. Jiao et al. improved the guess-and-determine attack on Trivium by enhancing attack efficiency through several techniques, resulting in reduced complexity [52]. Tian et al. discussed new design principles for stream ciphers based on Trivium, introducing k-order primitive polynomials for new designs [53]. Teo et al. applied algebraic attack approaches to Trivium-like ciphers, presenting new attacks on simplified versions like Bivium-A [54]. These studies highlight Trivium's robustness despite various proposed attacks and underscore the importance of ongoing cryptographic analysis to adapt and strengthen its security. That's why Trivium is also implemented as an LCRC mechanism of PCIe and investigated if it is a good alternative of the enhanced LCRC.

Trivium is designed as a bit-oriented stream cipher that operates on an 80-bit secret key and an 80-bit initialization vector (IV). The cipher initializes its 288-bit internal state using the key and IV and then updates this state to produce a keystream. The initialization phase involves setting up the internal state followed by 4 cycles of state updates. The keystream generation phase then proceeds by iteratively updating the state and producing output bits.

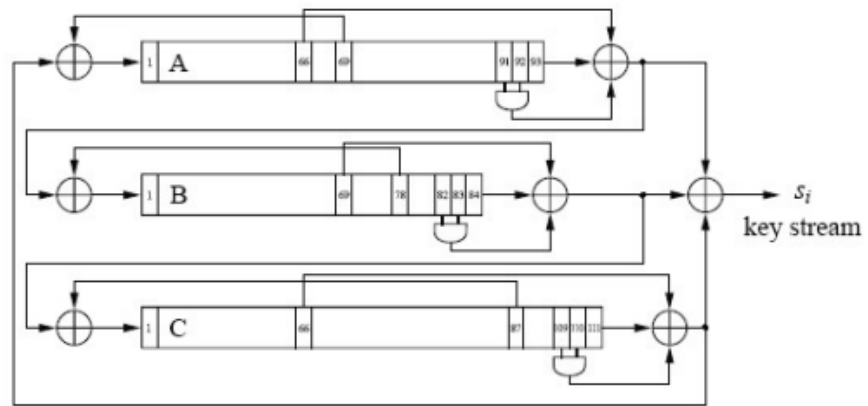


Figure 4.4: Overview of the building blocks in Trivium [49].

In initialization phase:

- The 80-bit key is loaded into the first 80 bits of the internal state.
- The 80-bit IV is loaded into the next 80 bits of the internal state.
- The remaining 128 bits of the internal state are set to zero, except for the last three bits, which are set to 1.
- The internal state is then updated for 1152 cycles (4 times 288) without producing output bits.

Keystream Generation Phase:

- The internal state is updated in each cycle using nonlinear feedback functions.
- In each cycle, three state bits are updated, and one keystream bit is produced.
- The update functions involve XOR and AND operations between specific bits of the internal state.

An open source Trivium core [55] was used in the implemented Trivium algorithm to investigate whether it could be used instead of LCRC. Trivium implementation results are shown in Table 4.5.

Table 4.5: Trivium as LCRC implementation results.

Parameter	Value
LUT	141
LUTRAM	24
FF	373
IO	108
BUFG	1
Worst Negative Slack (WNS) (ns)	0.200
Freq (MHz)	102.04
Power (W)	0.095
Junction Temperature (°C)	25.4

Finally, the Reed Solomon algorithm was also implemented to compare with ASCON128a, GRAIN and Trivium, and the implementation results were evaluated.

4.3.4. Reed Solomon

Reed-Solomon (RS) codes are a class of error-correcting codes that are widely used in digital communications and storage systems. Developed by Irving S. Reed and Gustave Solomon in 1960, these codes are highly effective in correcting multiple symbol errors, making them integral to various applications, from compact discs to deep-space communication. Reed-Solomon codes differ from lightweight cryptography algorithms, which are designed for environments with severe constraints on processing power, memory, and energy consumption, such as RFID tags and sensor networks. Lightweight cryptography methods, focus on minimal resource usage and simplicity. In contrast, RS codes, though more complex, offer robust error correction and are used in applications requiring high data integrity. Lightweight cryptography algorithms, while efficient, generally do not provide the same level of error correction and are better suited for environments where resource constraints are a primary concern [56]. One of the reasons why Reed Solomon codes are discussed in this study is to investigate what the differences between RS and LWCs will result in implementation.

The literature on Reed-Solomon codes is extensive, reflecting their broad applicability and robustness. Berlekamp introduced bit-serial Reed-Solomon encoders, highlighting their efficiency compared to linear cyclic binary codes [57]. Wicker and Bhargava provided a comprehensive historical overview and discussed various applications, from space exploration to compact discs [58]. Plank simplified RS coding implementation for system programmers in RAID-like systems [59].

Lee et al. proposed a high-speed Reed-Solomon decoder suitable for digital audio/video equipment, emphasizing the code's ability to correct burst errors efficiently [60]. Guruswami and Wootters explored the exact repair problem in distributed storage, demonstrating RS codes as optimal regenerating codes in specific parameter regimes [61]. Schmidt et al. examined collaborative decoding of interleaved RS codes, showing their ability to correct error patterns beyond half the minimum distance [62].

Sudan introduced a randomized algorithm for decoding RS codes beyond the conventional error-correction bound, marking a significant advancement in decoding techniques [63]. Cardarilli et al. designed a self-checking RS encoder leveraging Galois

field properties for enhanced fault tolerance in transmission and storage systems [64].

In this study, it was planned to use Reed Solomon codes only as a parallelized CRC circuit without using the error correction features and compared with the enhanced LCRC mechanism in terms of performance.

Reed-Solomon codes operate over finite fields (Galois fields). Each codeword comprises symbols represented by polynomials. The encoding process generates a polynomial from the message and adds redundant symbols to form the codeword, ensuring robust error detection and correction.

In encoding phase:

- The message is represented as a polynomial $m(x)$.
- The generator polynomial $g(x)$ is used to produce the codeword polynomial $c(x)$, which includes the message and redundant symbols.
- The codeword polynomial is derived as

$$c(x) = m(x) \cdot g(x). \quad (4.3)$$

In decoding phase:

- The received polynomial $r(x)$ may contain errors.
- The syndrome polynomial $S(x)$ is calculated to detect errors.
- The error locator polynomial and error magnitude polynomial are used to identify and correct errors.

Reed-Solomon codes utilize Galois fields, specifically $GF(2^m)$, where arithmetic operations are performed modulo a primitive polynomial. This enables efficient error detection and correction, crucial for handling burst errors.

Reed-Solomon codes can be implemented in both hardware and software. Hardware implementations often involve dedicated circuits for encoding and decoding, leveraging parallel processing for high-speed applications. Software implementations use efficient algorithms to perform Galois field arithmetic and error correction. In this study, an open source Reed Solomon core [65] implemented which can be seen in Table 4.6.

Table 4.6: Reed Solomon as LCRC implementation results.

Parameter	Value
LUT	242
LUTRAM	0
FF	326
IO	116
BUFG	1
Worst Negative Slack (WNS) (ns)	-0.183
Freq (MHz)	98.20
Power (W)	0.135
Junction Temperature (°C)	25.6

5. PERFORMANCE EVALUATION

In this research, we propose a new design methodology for the energy-efficient implementation of the PCI Express-compliant CRC code generating mechanism, which has found widespread application in mobile computing systems. We outline the basic ideas underlying our methodology and offer specifications for a hardware implementation that maximizes power efficiency. The outcomes of the performance evaluation show that both output throughput and energy efficiency can be significantly increased. Power consumption can be improved by 38% to 85%, according to the our publication "Power Optimization of LCRC for A Custom PCI Express 6.0 IP Core," which is given as an enhanced LCRC in Chapter 3. The suggested approach is specially modified to the CRC structure on other protocols in addition to being made to adhere to the PCIe standard.

Enhanced LCRC mechanism has been used in simulation environments within a holistic custom PCI Express Interface containing PHY, DLL and TL and has proven itself in the field. It can be used operationally on both FPGA and ASIC backend. The obtained simulation snapshot can be seen in Fig. 5.1.

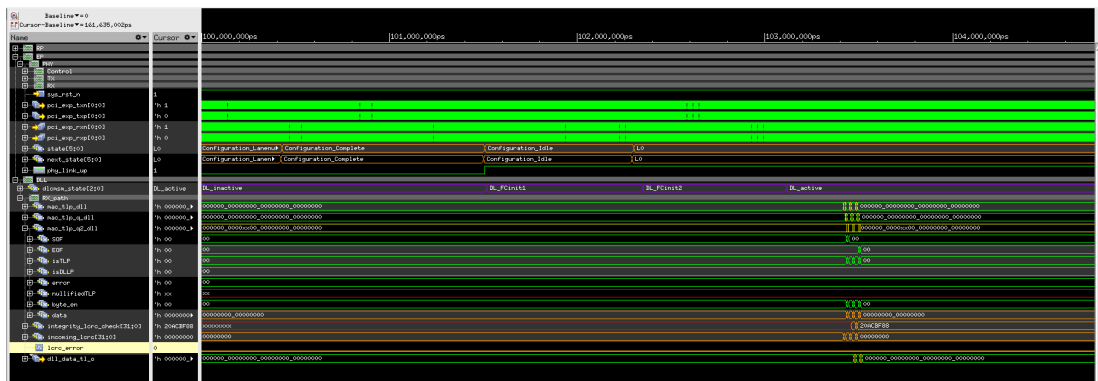


Figure 5.1: Simulation snapshot of incoming TLP.

In Fig. 5.1, firstly LTSSM in PHY layer is trained to L0 state and then it asserts phy_link_up signal. After that, DLCMSM state machine in DLL is trained to DL_active state. Now TLP exchange is made on Rx and TX of DLL. In the simulation, it is

Table 5.1: Comparison of various LCRC implementations on Xilinx KCU105 board.

Parameter	ASCON128	TRIVIUM	GRAIN	REED SOLOMON	Enhanced LCRC	Serial LCRC
LUT	734	141	2605	242	245	71
LUTRAM	0	24	0	0	0	0
FF	373	373	782	326	33	38
IO	108	108	108	116	108	108
BUFG	1	1	1	1	1	1
WNS (ns)	0.352	0.200	0.392	-0.183	-2.827	-3.546
Freq (MHz)	103.65	102.04	104.08	98.20	77.96	73.82
Power (W)	0.114	0.095	0.28	0.135	0.141	0.118
Junction Temperature (°C)	25.5	25.4	26.3	25.6	25.6	25.5

consumption to enhanced LCRC hardware, but it is more advantageous than enhanced LCRC hardware in terms of operating frequency. Therefore, if backward compatibility can be sacrificed, it is appropriate to use Reed Solomon cores instead of LCRC hardware. Based on the findings presented in this study, it has been demonstrated that the LCRC hardware within the PCI Express core offers substantial potential for further development. This hardware not only exhibits a significant opportunity for optimization but also indicates that enhancements in power efficiency and performance can lead to considerable benefits for the entire PCI Express interface. The study highlights that through targeted improvements in the CRC calculation unit or other components within the PCI Express architecture, there is potential to achieve noteworthy advancements in the interface's overall efficiency.

In conclusion, this thesis has demonstrated significant advancements in the enhancement of data integrity within the PCIe Data Link Layer by employing energy-efficient LWC algorithms. The research addressed the limitations of traditional Link Cyclic Redundancy Check (LCRC) mechanisms, which are typically power-intensive and less efficient for high-speed data transmission. By integrating LWCs such as ASCON, GRAIN, and Trivium, the study achieved notable reductions in power consumption while maintaining or improving data integrity, making the solutions highly relevant for modern computing environments. The proposed LCRC mechanisms leveraged parallel processing to enhance performance and reduce power usage, thus filling a critical gap in the literature that often overlooks the energy efficiency of data integrity measures within PCIe systems.

The findings of this thesis offer a comprehensive solution that balances the need for high data integrity with the requirements for low power consumption, thereby contributing to the fields of high-performance and energy-efficient computing. Future research should focus on further optimizing these LCRC mechanisms, integrating them with emerging technologies such as CXL and UCIe, and validating their performance in real-world PCIe environments. Additionally, exploring the scalability and adaptability of these solutions to other high-performance applications and enhancing their security features will be crucial for advancing the state of the art in energy-efficient data integrity technologies for high-speed computing interfaces.

REFERENCES

- [1] Verma A., Dahiya P. K., (2017), “Pcie bus: A state-of-the-art-review”, IOSR journal of VLSI and Signal Processing, 07, 24–28.
- [2] Markussen J., Kristiansen L., Seifert F., Riegler M., Griwodz C., Halvorsen P., Borgli R., Stensland H., (2020), “Flexible device compositions and dynamic resource sharing in pcie interconnected clusters using device lending”, Cluster Computing, 22, 1–24.
- [3] Zainol M. A., Nunez-Yanez J. L., (2016), “Cpcie: A compression-enabled pcie core for energy and performance optimization”, 2016 IEEE Nordic Circuits and Systems Conference (NORCAS), 1–6, Copenhagen, Denmark, 1–2 November.
- [4] Verdiell J.-M., Zbinden E., Lee R., Troxell B., (2011), “Pcie optical interconnects”, 2011 IEEE Avionics, Fiber- Optics and Photonics Technology Conference, 21–22, San Diego, CA, USA, 4–6 October.
- [5] Supalov A., Semin A., Klemm M., Dahnken C., (2014), “Optimizing HPC Applications with Intel® Cluster Tools”, 1st Edition, Apress.
- [6] Neugebauer R., Antichi G., Zazo J. F., Audzevich Y., López-Buedo S., Moore A., (2018), “Understanding pcie performance for end host networking”, Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, 327–341, Budapest, Hungary, 20–25 August.
- [7] Sharma D. D., (2021), “Pci express 6.0 specification: A low-latency, high-bandwidth, high-reliability, and cost-effective interconnect with 64.0 gt/s pam-4 signaling”, IEEE Micro, 41, 23–29.
- [8] Sharma D. D., Choudhary S., (2023), “Pipelined and partitionable forward error correction and cyclic redundancy check circuitry implementation for pci express® 6.0”, 2023 IEEE Symposium on High-Performance Interconnects (HOTI), 1–8, CA, USA, 23–25 August.
- [9] Ryu S., Kim S., Jun J., Moon D., Lee K., Choi J., Kim S., Kim H., Kim L., Choi W. H., Nam M., Hwang D., Roh H., Joo Y.-P., (2023), “System optimization of data analytics platforms using compute express link (cxl) memory”, 2023 IEEE International Conference on Big Data and Smart Computing (BigComp), 9–12, Jeju, South Korea, 13–16 February.
- [10] Gouk D., Kwon M., Bae H., Lee S., Jung M., (2023), “Memory pooling with cxl”,

IEEE Micro, 43, 48–57.

- [11] Fiala D., Mueller F., Engelmann C., Riesen R., Ferreira K., Brightwell R., (2012), “Detection and correction of silent data corruption for large-scale high-performance computing”, SC ’12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 1–12, Salt Lake City, UT, 10–16 November.
- [12] Xu G., (2020), “Iot-assisted ecg monitoring framework with secure data transmission for health care applications”, IEEE Access, 8, 74586–74594.
- [13] Wei P., Wang D., Zhao Y., Tyagi S. K. S., Kumar N., (2020), “Blockchain data-based cloud data integrity protection mechanism”, Future Generation Computer Systems, 102, 902–911.
- [14] Xu J., Wei L., Wu W., Wang A., Zhang Y., Zhou F., (2020), “Privacy-preserving data integrity verification by using lightweight streaming authenticated data structures for healthcare cyber–physical system”, Future Generation Computer Systems, 108, 1287–1296.
- [15] Seth B., Dalal S., Jaglan V., Le D.-N., Mohan S., Srivastava G., (2022), “Integrating encryption techniques for secure data storage in the cloud”, Transactions on Emerging Telecommunications Technologies, 33(4), e4108.
- [16] Jabbar R., Fetais N., Krichen M., Barkaoui K., (2020), “Blockchain technology for healthcare: Enhancing shared electronic health record interoperability and integrity”, 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), 310–317, Doha, Qatar, 2–5 February.
- [17] Charyyev B., Alhussen A., Sapkota H., Pouyoul E., Gunes M. H., Arslan E., (2019), “Towards securing data transfers against silent data corruption”, 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 262–271, Larnaca, Cyprus, 14–17 May.
- [18] Alhussen A., Arslan E., (2021), “Avoiding data loss and corruption for file transfers with fast integrity verification”, Journal of Parallel and Distributed Computing, 152, 33–44.
- [19] Kanimozhi E. A., Suguna M., Mercy Shalini S., (2019), “Immediate detection of data corruption by integrating blockchain in cloud computing”, 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), 1–4, Vellore, India, 30–31 March.
- [20] Web 1, (2024), <http://pcisig.com/blog/pcie%C2%AE-70-specification->

version-05-now-available-full-draft-available-members, Accessed: 09.06.2024.

- [21] M. Jackson R. Budruk J. W., Anderson D., (2012), “PCI Express Technology 3.0”, 1st Edition, MindShare Press.
- [22] PCI-SIG (2022), Pci express base specification revision 6.0 version 1.0, Technical Report No : 6.0, PCI-SIG, Beaverton, OR, USA.
- [23] Chao H. J., Guo X., (2001), “Quality of Service Control in High-Speed Networks”, 1st Edition, Wiley.
- [24] Nakamura H., Takayama H., Yamaguchi Y., Boku T., (2017), “Thorough analysis of pcie gen3 communication”, 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), 1–6, Cancun, Mexico, 4–6 December.
- [25] Rambus (2024), “Pci express glossary”, <https://www.rambus.com/interface-ip/pci-express-glossary/>, Accessed: 09/06/2024.
- [26] Khafidin A., Andrasto T., Suryono S., (2019), “Implementation flow control to improve quality of service on computer networks”, Indonesian Journal of Electrical Engineering and Computer Science, 16, 1474.
- [27] Richter A., Herber C., Wild T., Herkersdorf A., (2016), “Resolving performance interference in sr-iov setups with pcie quality-of-service extensions”, 2016 Euromicro Conference on Digital System Design (DSD), 454–462, Limassol, Cyprus, 31 August–2 September.
- [28] Liu H., Qiu Z., Pan W., Li J., Zheng L., Gao Y., (2020), “Low-cost and programmable crc implementation based on fpga”, IEEE Transactions on Circuits and Systems II: Express Briefs, 68, 211–215.
- [29] Liu W., Pang Y., Li X., Zhang Y., (2022), “Design and implementation of a radiation-hardened on-board fpga maintenance control chip”, 2022 International Conference on Frontiers of Communications, Information System and Data Science (CISDS), 100–105, Guangzhou, China, 25–27 November.
- [30] Qihao L., Huihui W., Feng Z., Jianzhong Z., Jun-sheng L., You L., (2012), “An efficient physical coding sublayer for pci express in 65nm cmos”, 2012 International Symposium on Intelligent Signal Processing and Communications Systems, 625–629, Tamsui, New Taipei City, Taiwan, 4–7 November.
- [31] Bae W., Cho S.-Y., Jeong D.-K., (2021), “A 1.93-pj/bit pci express gen4 phy transmitter with on-chip supply regulators in 28 nm cmos”, Electronics, 10(1), 68.

- [32] Rohilla G., Mathur D., Ghanekar U., (2020), “Functional verification of mac-phy layer of pci express gen5.0 with pipe interface using uvm”, 2020 International Conference for Emerging Technology (INCET), 1–5, Belgaum, India, 5–7 June.
- [33] Sharma D. D., (2022), “A low-latency and low-power approach for coherency and memory protocols on pci express 6.0 phy at 64.0 gt/s with pam-4 signaling”, IEEE Micro, 42, 37–43.
- [34] Corporation I., (2018), Phy interface for the pci express, sata, usb 3.1, displayport, and converged io architectures (version 5.1), Technical Report No : 5.1, Intel, Piscataway, NJ, USA.
- [35] Stavinov E., (2010), “A practical parallel crc generation method”, Circuit Cellar Magazine for Computer Applications, 31(234), 38.
- [36] Sheng-Ju S., (2015), “Implementation of cyclic redundancy check in data communication”, 2015 International Conference on Computational Intelligence and Communication Networks (CICN), 529–531, Jabalpur, India, 12–14 December.
- [37] Campobello G., Patanè S., Russo M., (2003), “Parallel crc realization”, IEEE Transactions on Computers, 52, 1312–1319.
- [38] Mahajan R., Devi K. P., Bagai D., (2020), “Area efficient parallel lfsr for cyclic redundancy check”, International Journal of Electrical and Computer Engineering (IJECE), 10, 1755.
- [39] Herrera E., (2024), “Ascon128”, <https://github.com/edgarherrera03/ASCON128.git>, Accessed: 09/06/2024.
- [40] Web 1, (2024), “Ascon - lightweight and robust cryptographic algorithm”, <https://ascon.iaik.tugraz.at/>, Accessed: 09/06/2024.
- [41] Abdulgadir A., Diehl W., Kaps J., (2019), “An open-source platform for evaluation of hardware implementations of lightweight authenticated ciphers”, 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig), 1–5, Cancun, Mexico, 9–11 December.
- [42] Kaur J., Kermani M. M., Azarderakhsh R., (2022), “Hardware constructions for error detection in lightweight authenticated cipher ascon benchmarked on fpga”, IEEE Transactions on Circuits and Systems II: Express Briefs, 69, 2276–2280.
- [43] ERYILMAZ Y. E., (2022), Extending the instruction set of risc-v processor for ascon algorithm, Master’s thesis, Istanbul Technical University.

- [44] Sönnerup J., Hell M., Sönnerup M., Khattar R., (2019), “Efficient hardware implementations of grain-128aead”, *Lecture Notes in Computer Science*, 11898, 495–513.
- [45] Li B., Zhang H., Lin D., (2023), “Efficient (masked) hardware implementation of grain-128aeadv2”, *Security and Communication Networks*, 2023, 1–16.
- [46] Teffandi N., Feryputri N. A.-Z., Hasanuddin M. O., Syafalni I., Sutisna N., (2023), “Grain algorithm implementation for lightweight hardware-based otp authentication”, *2023 International Conference on Electrical Engineering and Informatics (ICEEI)*, 1–6, Bandung, Indonesia, 10–11 October.
- [47] Berbain C., Gilbert H., Maximov A., (2006), “Cryptanalysis of grain”, *Proceedings of the 13th international conference on Fast Software Encryption*, 15–29, Graz, Austria, 15–17 March.
- [48] Mattias Sönnerup R. K., Sönnerup J., (2019), “Grain-128aead-vhdl”, <https://github.com/Grain-128AEAD/Grain-128AEAD-VHDL/tree/master>, Accessed: 09/06/2024.
- [49] De Cannière C., Preneel B., (2008), “Trivium”, *Springer Berlin Heidelberg, Berlin, Heidelberg*, 244–266.
- [50] Maximov A., Biryukov A., (2007), “Two trivial attacks on trivium”, *IACR Cryptol. ePrint Arch.*, 2007, 21.
- [51] Hojsík M., Rudolf B., (2008), Differential fault analysis of trivium, in K. Nyberg, ed., “Fast Software Encryption”, *Springer Berlin Heidelberg, Berlin, Heidelberg*, 158–172.
- [52] Jiao L., Hao Y., Li Y., (2019), “Improved guess-and-determine attack on trivium”, *IET Inf. Secur.*, 13, 411–419.
- [53] Tian Y., Chen G., Li J., (2009), “On the design of trivium”, *IACR Cryptol. ePrint Arch.*, 2009, 431.
- [54] Teo S.-G., Wong K. K.-H., Bartlett H., Simpson L., Dawson E., (2013), “Algebraic analysis of trivium-like ciphers”, *IACR Cryptol. ePrint Arch.*, 2013, 240.
- [55] Web 1, (2016), “Trivium”, <https://github.com/FuzzyLogic/Trivium>, Accessed: 09/06/2024.
- [56] Buchanan W., Li S., Asif R., (2017), “Lightweight cryptography methods”, *Journal of Cyber Security Technology*, 1, 187 – 201.

- [57] Berlekamp E., (1982), “Bit-serial reed - solomon encoders”, IEEE Transactions on Information Theory, 28, 869–874.
- [58] Wicker S. B., Bhargava V. K., (1999), “Reed-Solomon Codes and Their Applications”, 1st Edition, John Wiley & Sons, Inc.
- [59] Plank J. S., (1997), “A tutorial on reed–solomon coding for fault-tolerance in raid-like systems”, Software: Practice and Experience, 27(9), 995–1012.
- [60] Lee M. H., Choi S. B., Chang J. S., (1995), “A high speed reed-solomon decoder”, IEEE Transactions on Consumer Electronics, 41(4), 1142–1149.
- [61] Guruswami V., Wootters M., (2015), “Repairing reed-solomon codes”, IEEE Transactions on Information Theory, 63(9), 5684–5698.
- [62] Schmidt G., Sidorenko V. R., Bossert M., (2009), “Collaborative decoding of interleaved reed–solomon codes and concatenated code designs”, IEEE Transactions on Information Theory, 55(7), 2991–3012.
- [63] Sudan M., (1997), “Decoding of reed solomon codes beyond the error-correction bound”, Journal of Complexity, 13(1), 180–193.
- [64] Cardarilli G., Pontarelli S., Re M., Salsano A., (2005), “A self checking reed solomon encoder: design and analysis”, 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT’05), 111–119, Monterey, CA, 3–5 October.
- [65] Web 1, (2020), “rs-codec”, <https://github.com/RedFlag2017/rs-codec>, Accessed: 09/06/2024.

BIOGRAPHY

Muhammed Mustafa ÖNEY completed his bachelor's degree at Istanbul Technical University in 2019. He is currently pursuing his master's degree at Gebze Technical University. He works as a Digital ASIC Design Engineer in TÛTEL laboratory at TÛBİTAK BİLGEM. His major interests are high-speed communication protocols, accelerator hardwares, and RISC-V architectures.

PUBLICATIONS AND PRESENTATIONS FROM THE THESIS

”Power Optimization of LCRC for Custom PCI Express 6.0 IP Core”

The paper is accepted by ICEEE-2024 conference, and it is presented in April 2024.
The publication is going to be archived in IEEE Xplore.

