MDPI

Article

Design and Implementation of a UART Module on FPGA Using RTL for Cryptographic Encryption and Decryption Techniques

Christian-Antonio Colin-Cejudo[1,†,‡], Gonzalo-Issac Duchén-Sánchez [2,‡] and Firstname Lastname [2,*]

[1] Affiliation 1; christian.upiita@gmai.com
[2] Affiliation 2; gduchen@ieee.org
* Correspondence: e-mail@e-mail.com; Tel.: (optional; include country code; if there are multiple corresponding authors, add author initials) +xx-xxxx-xxx-xxxx (F.L.)
† Current address: Affiliation.
‡ These authors contributed equally to this work.

Abstract: The increasing demand for robust and efficient information security has led to the growing adoption of specialized hardware for cryptographic operations. In response to the rise in cyber threats and the need to process large volumes of data in real time, hardware-based cryptographic solutions offer significant advantages in terms of performance, resistance to attacks, and secure storage of cryptographic keys. This thesis presents the implementation of a secure communication system using the UART (Universal Asynchronous Receiver-Transmitter) protocol as the foundation for a Register Transfer Level (RTL) design on an FPGA platform. The base protocol was modified to introduce an additional hardware-level security layer. Furthermore, cryptographic techniques—specifically encryption and decryption—were integrated into the design to enhance data protection and integrity during transmission. The results demonstrate the feasibility of embedding cryptographic mechanisms directly into communication hardware, providing a scalable and efficient solution for secure embedded systems.

Keywords: Cryptographic hardware; Information security; Encryption, Decryption; Cryptographic keys; Digital signatures; Authentication; Cyber threats; Data processing; Communication protocols; UART (Universal Asynchronous Receiver-Transmitter); FPGA (Field-Programmable Gate Array); RTL design (Register Transfer Level); Hardware security layer; Cryptographic techniques; Critical infrastructure.

1. Introduction

Currently, information security has become a fundamental pillar for the development of reliable digital systems. The increasing sophistication of cyber threats, combined with the exponential growth in data generation and transmission, demands increasingly robust and efficient solutions. In this context, the use of specialized hardware for cryptographic operations has emerged as an effective alternative to the limitations of software-only cryptographic processing.

Cryptographic hardware offers significant advantages, such as higher performance, lower latency, reduced energy consumption, and greater resistance to both physical and logical attacks. These characteristics make it an ideal solution for embedded systems, IoT devices, industrial applications, and environments where security and operational efficiency are top priorities.

Field Programmable Gate Arrays (FPGAs) have become one of the most versatile technologies for electronic system design. These devices provide a cost-effective solution,

Received:
Revised:
Accepted:
Published:

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Journal Not Specified* **2025**, *1*, 0. https://doi.org/

especially for low-volume production, since the initial cost of prototyping is considerably lower compared to Application-Specific Integrated Circuits (ASICs). Moreover, a key advantage is their reconfigurability during operation, which allows a single device to perform multiple predefined functions, thereby optimizing space and reducing costs.

Today, electronic systems are present in almost every aspect of daily life, from consumer products to industrial control systems, automotive applications, security, and beyond. The current trend in electronic design is characterized by the increasing complexity of components, which demands solutions that are user-friendly, versatile, low-power, and quick to market. Modern integrated circuit technology makes it possible to integrate these complex systems into highly compact dimensions, referred to as embedded systems or Systems on Chip (SoC). These systems, designed to fulfill specific functions, combine hardware and software tailored for each task. In this context, FPGAs, thanks to their reconfigurability, are valuable tools for developing prototypes or small-scale series at affordable costs.

Among the most widely used communication protocols in embedded systems is the Universal Asynchronous Receiver-Transmitter (UART), due to its simplicity, low implementation cost, and broad compatibility. However, this protocol lacks native security mechanisms, making it vulnerable to interception, manipulation, and unauthorized access.

This thesis proposes the design and implementation of a cryptographic security layer on top of the UART protocol, using a Register Transfer Level (RTL) approach on an FPGA platform. The implementation leverages a Hardware Description Language (HDL) to model and design digital circuits. The modification of the protocol not only ensures secure data transmission but also integrates encryption and decryption techniques directly into hardware. This solution aims to demonstrate the feasibility of incorporating efficient cryptographic security into embedded communication systems without compromising performance or scalability.

2. Materials and Methods

To properly contextualize the present thesis, it is necessary to address the origins of the standard cryptographic algorithm Caesar, as well as its evolution and applications over time. Likewise, the conditions under which information security services and mechanisms emerged will be explored, along with the ways in which they have been implemented across different environments. This analysis allows for the delineation of the problem that this research seeks to address through the proposed hardware-based implementation. Consequently, a review of the state of the art in the field is carried out, with particular emphasis on embedded systems that incorporate security mechanisms.

This review seeks to establish the foundations underlying the design of cryptographic solutions on reconfigurable platforms such as FPGAs, highlighting their relevance in scenarios where information protection is critical. This theoretical framework provides the basis for the methodological and experimental development presented in the following chapters.

2.1. Classical Cryptography Fundamentals

Cryptography is the science and art of protecting information through techniques that transform readable data (plaintext) into unintelligible data (ciphertext), so that only authorized individuals can access its content (Stallings, 2017). Throughout history, cryptography has evolved from simple substitution and transposition methods to complex algorithms based on mathematical theory, modern algebra, and number theory.

Although classical methods are now considered insecure against modern attacks, they provide a foundation for understanding fundamental concepts such as key usage,

symmetric encryption, and cryptanalysis. One of the most representative algorithms of classical cryptography is the Caesar cipher, also known as the shift cipher.

2.2. The Caesar Cipher: Historical Background

The Caesar cipher is named after Julius Caesar, who used it to encrypt military communications. Historical records indicate that Caesar applied a shift of three positions to encode his messages (Kahn, 1996). While its simplicity renders it trivial by modern standards, at the time it provided a basic level of protection against untrained or illiterate readers.

This cipher is classified as a monoalphabetic substitution cipher, in which each letter of the alphabet is systematically replaced by another according to a fixed shift key.

2.3. Operation of the Algorithm

From a mathematical perspective, the Caesar cipher can be represented by the following function: $$C(x) = (x + k) mod n$$

Where:

- $C(x)$ is the encrypted character.
- x is the index of the character in the alphabet $A = 0, B = 1, \ldots, Z = 25$.
- k is the shift key.
- n is the total number of characters in the alphabet, usually $n = 26$ for the Latin alphabet.

Decryption consists of applying the inverse operation:

$$P(x) = (x - k) \bmod n$$

For example, if the word ***"HOLA"*** is encrypted with a shift of $k = 3$, the result is ***"KROD"***. To decrypt, the inverse shift is applied.

2.4. Security and Vulnerabilities

The Caesar cipher is vulnerable to several types of cryptographic attacks:

- **Brute-force attack:** Since there are only 25 possible keys (excluding the null shift), an attacker can test all combinations in a short time.
- **Frequency analysis:** Each language has a characteristic letter distribution. In Spanish, the letters E", A", and "O" are the most common. If a sufficiently long text is encrypted, these frequencies are preserved, allowing the shift used to be identified.

These weaknesses make it unsuitable for any modern application requiring real confidentiality, but it remains useful as a development and educational tool.

2.5. Usefulness in Cryptographic System Design

Despite its limitations, the Caesar cipher is frequently used in the design of hardware/software cryptographic systems for educational and experimental purposes. Its simple structure makes it an ideal choice for:

- Introducing sequential data processing techniques.
- Implementing finite state machines for encoding and decoding.
- Evaluating logical resource usage and propagation delays on platforms such as FPGAs.
- Comparing the behavior of more complex ciphers with respect to basic algorithms in low-level environments (VHDL, Verilog).

In digital design projects with RTL (Register Transfer Level) architecture, the Caesar cipher is used as a case study to optimize modular shift operations, ASCII character encoding, and key generation.

2.6. Background on SystemVerilog

2.7. Evolution of Digital Design and the Emergence of SystemVerilog

The development of digital design has undergone significant transformations since its inception, beginning with physical prototyping using breadboards and evolving to sophisticated computer-aided simulation and modeling platforms. CAD (Computer-Aided Design) tools allowed for more precise and efficient simulation of electronic circuits, although their initial use required overcoming certain technical complexity barriers.

These tools facilitated the implementation of electrical schematics and enabled the definition of input signals consistent with system logic, producing outputs that were easily interpretable. Simulators such as SPICE became benchmarks in both academic and industrial settings by providing accurate modeling of electrical behavior. Once results were validated, the physical circuit design was carried out using specialized printed circuit board (PCB) tools, such as the OrCAD environment, particularly its PCB Layout module.

During this stage, the Adaptable Computing Laboratory began implementing electronic neural circuit models using these platforms (Padrón, 1997). Later, as the complexity of neural network models increased, tools such as MATLAB and its graphical environment SIMULINK were incorporated, allowing a direct mathematical representation of dynamic systems. This evolution enabled more effective interpretation of results, both quantitatively and qualitatively (Padrón et al., 2000, pp. 338–349).

In parallel, the reconfigurable hardware industry, particularly through FPGA boards manufactured by Xilinx, allowed addressing specific problems via programmable interfaces. Each board is designed to offer flexibility to the user, who can program its components in hardware description languages such as Verilog or VHDL and integrate various interfaces depending on the physical or logical availability of the system. This versatility made development kits fundamental tools for designing custom solutions, especially in rapid prototyping and functional validation contexts.

As system-level design requirements grew and verification became a critical stage, SystemVerilog emerged as an extension and evolution of the Verilog language. SystemVerilog not only incorporated improvements in structural and behavioral hardware description but also integrated capabilities for functional verification, object-oriented programming, and complex system modeling. This evolution addressed the need to unify the design and verification flow under a single language, facilitating the implementation of advanced testbenches, reusable interfaces, and verification environments compatible with modern methodologies such as UVM (Universal Verification Methodology).

In this regard, the use of FPGA boards programmed with SystemVerilog allows combining the precision of RTL design with advanced verification capabilities, which is essential in development environments aiming to ensure functionality, performance, and reliability from the early stages of a project. The history and evolution of this language reflect a trend toward consolidating tools that integrate design, simulation, verification, and synthesis within a single development environment.

Relevant chronology of SystemVerilog development:

- 1984: The Verilog language is introduced as a hardware description tool by Gateway Design Automation, later acquired by Cadence (IEEE, 2008).
- 1990s: Verilog becomes one of the main standards for digital design, widely used in the electronics industry.
- 1999: Development of SystemVerilog begins as an extension of Verilog to address limitations in verification and modeling (Accellera Systems Initiative, 2002).
- 2002: SystemVerilog is formally adopted by Accellera as a standard for design and verification.

- 2005: SystemVerilog is standardized by IEEE as IEEE 1800-2005, consolidating its industrial use (IEEE, 2005).
- 2012: The IEEE 1800-2012 revision is published, incorporating improvements in synthesis and verification.
- 2017: The IEEE 1800-2017 revision is published, expanding features for complex system design and verification.

2.8. Hardware Description Languages (HDL) and Their Application in Digital Design

Hardware Description Languages (HDL) emerged in response to the need for digital designers to have formal tools that allow the specification, modeling, and verification of digital systems in a structured manner and at different levels of abstraction. These languages not only facilitate communication among designers but also enable smooth interaction between computer-aided design (CAD) tools and the digital models themselves (Terés et al., 1998).

HDL-based development environments integrate compilation, simulation, and synthesis tools, allowing the functional behavior of a design to be validated before its physical implementation. Today, the most widely used HDL languages are VHDL and Verilog, both standardized by the IEEE (Institute of Electrical and Electronics Engineers) and broadly adopted in the digital design industry.

VHDL: Description, Application, and Structure

VHDL (VHSIC Hardware Description Language) was originally developed to document and verify high-speed integrated circuits within the VHSIC program of the U.S. Department of Defense. Its syntax and semantics are based on the ADA programming language, providing a robust structure suitable for critical and real-time systems.

VHDL allows modeling a digital system at different levels of description: behavioral, register-transfer level (RTL), and structural logic level. This enables coverage of nearly the entire digital development cycle, from functional specifications to prototype generation, excluding only the physical layout.

The correct choice of abstraction level in the description depends on the design objective. For instance, if the goal is to generate a physical implementation using synthesis tools, the RTL level is preferred. In contrast, to validate the functionality of complex algorithms through simulation, the algorithmic level is more appropriate (Baena, 2010).

2.9. Comparison between Hardware Description Languages and Software Development

Since their inception, various computer-aided design (CAD) tools and environments have employed specific languages and formats to represent and manage the diverse elements involved in electronic design. Some of these languages consisted of explicit notations used by the designer to describe inputs to particular tools, while others corresponded to internal intermediate formats, optimized for automated processing within the CAD environment and generally hidden from the designer.

However, these descriptions were limited to the particular ecosystem of each tool, offering neither portability nor standardization. The emergence of Hardware Description Languages (HDL) represented a significant advance by introducing a higher degree of standardization and incorporating principles from software engineering into the specification and modeling of digital hardware.

From a syntactic perspective, HDLs share similarities with high-level programming languages (HLL), which facilitates their learning for engineers with prior software development experience. For example, Verilog shares many features with the C language, whereas VHDL inherits its structure and semantics from the ADA language. While these similarities can accelerate HDL adoption, they can also lead to conceptual errors if software-oriented

strategies are applied in contexts where an accurate representation of the hardware's physical behavior is required.

It is essential that, when an HDL model is intended for synthesis and physical implementation (e.g., on FPGAs or ASICs), the designer adopts a hardware-oriented mindset, describing the logic using a structural or register-transfer level (RTL) approach rather than merely an algorithmic one. Conversely, employing software-like structures can be appropriate when the goal is solely functional simulation of the system, optimizing performance during verification and timing analysis stages.

In summary, HDLs are high-level formal languages specifically designed to represent electronic circuits at various levels of abstraction—from basic logic gates to complete digital systems—and allow flexible, structured, and precise hardware modeling, leveraging concepts from both electronic design and software development, as shown in Figure 1.

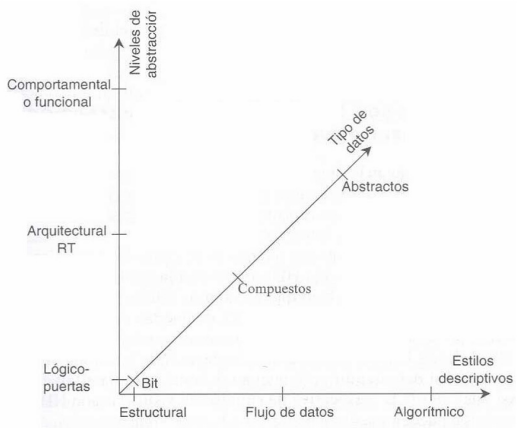


Figure 1. Abstraction/precision levels and VHDL modeling styles.

HDLs were initially developed to model the functional behavior of electronic components, enabling simulation prior to physical implementation. Nevertheless, they are also widely used for the structural description of digital circuits, facilitating subsequent synthesis and verification through validated simulation processes (Figure 2).

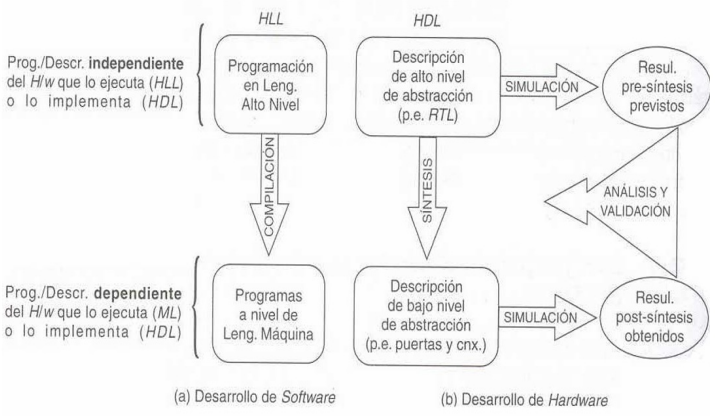


Figure 2. Software versus hardware development: (a) abstraction levels and high-level languages, and (b) basic top-down design scheme using HDL.

Building on the initial descriptions, top-down design approaches allow progressive synthesis processes that gradually lead to more detailed implementation levels, ultimately achieving a specific, technology-dependent physical description. Validation at each stage of the design is performed through functional simulations and analyses, enabling suc-

cessive verification and correction iterations until the behavior conforms to the specified requirements (Figure 2(b)).

3. Description of the Altera DE2-115 Development Board

The Altera DE2-115 board, developed by Terasic Technologies, is a prototyping platform based on the Altera (now Intel) Cyclone IV EP4CE115F29C7N FPGA, designed for educational environments, research, and advanced digital system development. This board offers a versatile architecture that allows implementation ranging from basic logic designs to complex digital systems, including embedded systems, custom controllers, digital signal processing (DSP), and System-on-Chip (SoC) prototypes.

Key features include 114,480 logic elements (LEs), 3,888 Kbits of embedded RAM, and 528 general-purpose input/output (GPIO) pins, providing significant flexibility for interfacing with multiple peripheral devices. The board also includes external memory such as 128 MB SDRAM, 2 MB SRAM, and 2 MB NOR Flash, in addition to SD card support for external storage.

The DE2-115 incorporates essential interfaces for developing interactive systems, including USB ports, Gigabit Ethernet, VGA output, audio input/output, and a set of integrated user I/O devices such as switches, buttons, LEDs, and seven-segment displays. Additionally, it features a 50 MHz crystal oscillator and expansion connectors compatible with additional modules.

This platform is widely used in academic laboratories and applied research environments due to its compatibility with development tools such as Quartus Prime, facilitating the implementation of designs in HDL languages like VHDL and SystemVerilog. Its capability to perform simulation, verification, and synthesis of designs within a single tool makes it ideal for higher education projects in electronic engineering, mechatronics, and digital systems Figure **??**.

Highlighted Technical Specifications:

FPGA: Altera Cyclone IV EP4CE115F29C7N with 114,480 logic elements (LEs)

- **Memory:**
  - 2 MB SRAM
  - 128 MB SDRAM
  - 2 MB NOR Flash
  - SD card for external storage
- **User Interfaces:**
  - 18 switches and 18 push-buttons
  - 18 red LEDs and 9 green LEDs
  - 8 seven-segment displays
- **Connectivity:**
  - USB Type-A and Type-B ports
  - 10/100/1000 Mbps Ethernet port (Gigabit)
  - VGA input/output
  - 40-pin GPIO expansion connectors
- **Audio and Video:**
  - Audio input/output (3.5 mm jack)
  - VGA signal input (with ADC) and VGA output
- **Clock:** 50 MHz crystal oscillator
- **Programming and Debugging:** JTAG support, compatible with Quartus II

This is an example of a quote.

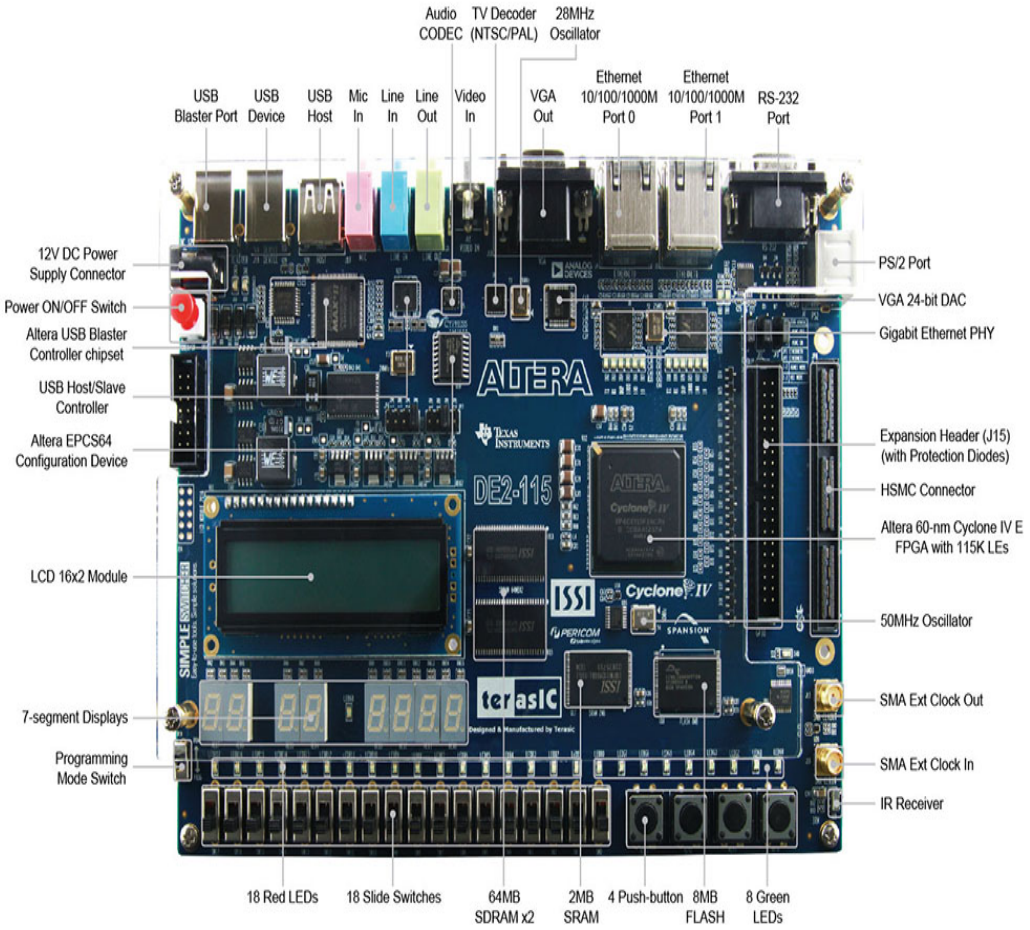


Figure 3. Altera DE2-115 Development Board.

4. Results

This section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation as well as the experimental conclusions that can be drawn.

4.1. Subsection

4.1.1. Subsubsection

Bulleted lists look like this:

- First bullet;
- Second bullet;
- Third bullet.

Numbered lists can be added as follows:

1. First item;
2. Second item;
3. Third item.

The text continues here.

4.2. Figures, Tables and Schemes

All figures and tables should be cited in the main text as Figure 3, Table 1, etc.

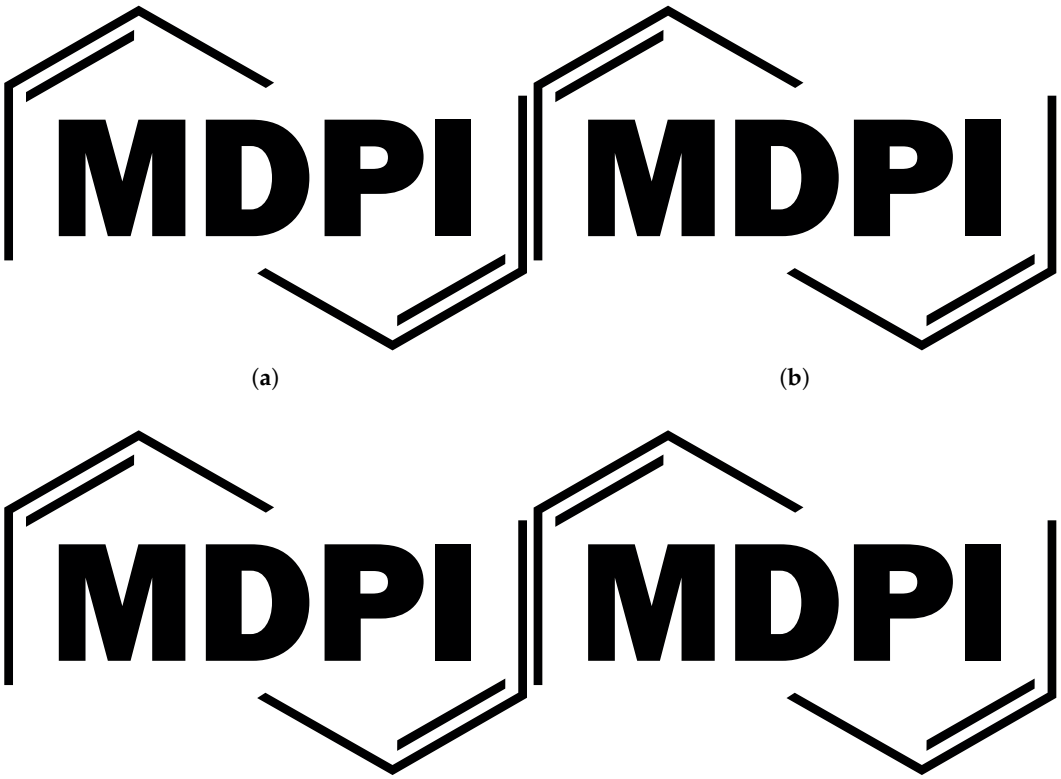
Figure 4. This is a figure. Schemes follow the same formatting.

Table 1. This is a table caption. Tables should be placed in the main text near to the first time they are cited.

Title 1	Title 2	Title 3
Entry 1	Data	Data
Entry 2	Data	Data [1]

[1] Tables may have a footer.

The text continues here (Figure 4 and Table 2).



(**a**) (**b**)

(**c**) (**d**)

Figure 5. This is a wide figure. Schemes follow the same formatting. If there are multiple panels, they should be listed as: (**a**) Description of what is contained in the first panel. (**b**) Description of what is contained in the second panel. (**c**) Description of what is contained in the third panel. (**d**) Description of what is contained in the fourth panel. Figures should be placed in the main text near to the first time they are cited. A caption on a single line should be centered.

Table 2. This is a wide table.

Title 1	Title 2	Title 3	Title 4
Entry 1 *	Data Data Data	Data Data Data	Data Data Data
Entry 2	Data Data Data	Data Data Data	Data Data Data

* Tables may have a footer.

Text.

Text.

4.3. Formatting of Mathematical Components

This is the example 1 of equation:

$$a = 1, \tag{1}$$

the text following an equation need not be a new paragraph. Please punctuate equations as regular text.

This is the example 2 of equation:

$$a = b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t + u + v + w + x + y + z \tag{2}$$

Please punctuate equations as regular text. Theorem-type environments (including propositions, lemmas, corollaries etc.) can be formatted as follows:

Theorem 1. *Example text of a theorem.*

The text continues here. Proofs must be formatted as follows:

Proof of Theorem 1. Text of the proof. Note that the phrase "of Theorem 1" is optional if it is clear which theorem is being referred to. □

The text continues here.

5. Discussion

Authors should discuss the results and how they can be interpreted from the perspective of previous studies and of the working hypotheses. The findings and their implications should be discussed in the broadest context possible. Future research directions may also be highlighted.

6. Conclusions

This section is not mandatory, but can be added to the manuscript if the discussion is unusually long or complex.

7. Patents

This section is not mandatory, but may be added if there are patents resulting from the work reported in this manuscript.

Author Contributions: For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualiza-

tion, X.X. and Y.Y.; methodology, X.X.; software, X.X.; validation, X.X., Y.Y. and Z.Z.; formal analysis, X.X.; investigation, X.X.; resources, X.X.; data curation, X.X.; writing—original draft preparation, X.X.; writing—review and editing, X.X.; visualization, X.X.; supervision, X.X.; project administration, X.X.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.", please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Funding: Please add: "This research received no external funding" or "This research was funded by NAME OF FUNDER grant number XXX." and and "The APC was funded by XXX". Check carefully that the details given are accurate and use the standard spelling of funding agency names at https://search.crossref.org/funding, any errors may affect your future funding.

Institutional Review Board Statement: In this section, you should add the Institutional Review Board Statement and approval number, if relevant to your study. You might choose to exclude this statement if the study did not require ethical approval. Please note that the Editorial Office might ask you for further information. Please add "The study was conducted in accordance with the Declaration of Helsinki, and approved by the Institutional Review Board (or Ethics Committee) of NAME OF INSTITUTE (protocol code XXX and date of approval)." for studies involving humans. OR "The animal study protocol was approved by the Institutional Review Board (or Ethics Committee) of NAME OF INSTITUTE (protocol code XXX and date of approval)." for studies involving animals. OR "Ethical review and approval were waived for this study due to REASON (please provide a detailed justification)." OR "Not applicable" for studies not involving humans or animals.

Informed Consent Statement: Any research article describing a study involving humans should contain this statement. Please add "Informed consent was obtained from all subjects involved in the study." OR "Patient consent was waived due to REASON (please provide a detailed justification)." OR "Not applicable" for studies not involving humans. You might also choose to exclude this statement if the study did not involve humans.

Written informed consent for publication must be obtained from participating patients who can be identified (including by the patients themselves). Please state "Written informed consent has been obtained from the patient(s) to publish this paper" if applicable.

Data Availability Statement: We encourage all authors of articles published in MDPI journals to share their research data. In this section, please provide details regarding where data supporting reported results can be found, including links to publicly archived datasets analyzed or generated during the study. Where no new data were created, or where data is unavailable due to privacy or ethical restrictions, a statement is still required. Suggested Data Availability Statements are available in section "MDPI Research Data Policies" at https://www.mdpi.com/ethics.

Acknowledgments: In this section you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments). Where GenAI has been used for purposes such as generating text, data, or graphics, or for study design, data collection, analysis, or interpretation of data, please add "During the preparation of this manuscript/study, the author(s) used [tool name, version information] for the purposes of [description of use]. The authors have reviewed and edited the output and take full responsibility for the content of this publication."

Conflicts of Interest: Declare conflicts of interest or state "The authors declare no conflicts of interest." Authors must identify and declare any personal circumstances or interest that may be perceived as inappropriately influencing the representation or interpretation of reported research results. Any role of the funders in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; or in the decision to publish the results must be declared in this section. If there is no role, please state "The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results".

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of open access journals
TLA	Three letter acronym
LD	Linear dichroism

Appendix A

Appendix A.1

The appendix is an optional section that can contain details and data supplemental to the main text—for example, explanations of experimental details that would disrupt the flow of the main text but nonetheless remain crucial to understanding and reproducing the research shown; figures of replicates for experiments of which representative data are shown in the main text can be added here if brief, or as Supplementary Data. Mathematical proofs of results not central to the paper can be added as an appendix.

Table A1. This is a table caption.

Title 1	Title 2	Title 3
Entry 1	Data	Data
Entry 2	Data	Data

Appendix B

All appendix sections must be cited in the main text. In the appendices, Figures, Tables, etc. should be labeled, starting with "A"—e.g., Figure A1, Figure A2, etc.

References

1. Author 1, T. The title of the cited article. *Journal Abbreviation* **2008**, *10*, 142–149.
2. Author 2, L. The title of the cited contribution. In *The Book Title*; Editor 1, F., Editor 2, A., Eds.; Publishing House: City, Country, 2007; pp. 32–58.
3. Author 1, A.; Author 2, B. *Book Title*, 3rd ed.; Publisher: Publisher Location, Country, 2008; pp. 154–196.
4. Author 1, A.B.; Author 2, C. Title of Unpublished Work. *Abbreviated Journal Name* year, *phrase indicating stage of publication (submitted; accepted; in press)*.
5. Title of Site. Available online: URL (accessed on Day Month Year).
6. Author 1, A.B.; Author 2, C.D.; Author 3, E.F. Title of presentation. In Proceedings of the Name of the Conference, Location of Conference, Country, Date of Conference (Day Month Year); Abstract Number (optional), Pagination (optional).
7. Author 1, A.B. Title of Thesis. Level of Thesis, Degree-Granting University, Location of University, Date of Completion.