School of
Electronics and Communication Engineering

Mini Project Report

on

Integrated Secure Data Encryption using AXI-Stream and PRESENT Algorithm

By:

1. **Shreesha Hegde** USN: 01FE21BEC226
2. **Prajwal Gouda J** USN: 01FE21BEC187
3. **Shrinidhi K T** USN: 01FE21BEC136
4. **Nandish Marali** USN: 01FE21BEC303

Semester: V, 2023-2024

Under the Guidance of

Dr. Nalini C I
Prof. Shraddha H

K.L.E SOCIETY'S
KLE Technological University,
HUBBALLI-580031
2023-2024

SCHOOL OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that project entitled **" Integrated Secure Data Encryption using AXI-Stream and PRESENT Algorithm "** is a bonafide work carried out by the student team of **"Shreesha Hegde (01FE21BEC226), Prajwal Gouda J (01FE21BEC187), Shrinidhi K T (01FE21BEC136), Nandish Marali (01FE21BEC303)"**. The project report has been approved as it satisfies the requirements with respect to the mini project work prescribed by the university curriculum for BE (V Semester) in School of Electronics and Communication Engineering of KLE Technological University for the academic year 2023-2024.

Prof. Shraddha Hiremath	**Dr. Suneeta V Budihal**	**Dr. B. S. Anami**
Guide	**Head of School**	**Registrar**

External Viva:

Name of Examiners **Signature with date**

1.

2.

ACKNOWLEDGMENT

We would like to extend our sincere thanks and gratitude to Dr. Suneetha V Budhihal, Head of School, for granting us the opportunity to acquire knowledge and collaborate on this project. We extend our sincere gratitude to our mini-project mentor, Prof. Shraddha H. We are very appreciative of her extensive technical expertise and assistance pertaining to the necessary technical components for the successful completion of this project. Furthermore, her encouragement has served as a catalyst for our increased efforts. Lastly, I would like to express my gratitude to all members of the department, starting from the lowest level, for their steadfast dedication, expertise, and assistance, which were indispensable in the successful completion of this project.

-The project team

ABSTRACT

In the dynamic landscape of cyber-security, safeguarding electronic circuits against potential threats has become paramount. This abstract introduces a pioneering endeavor aimed at enhancing the security posture of digital systems by developing an AXI-compatible PRESENT encryption IP core tailored for Nexys boards. Unlike traditional approaches, this initiative focuses solely on the encryption aspect, ensuring that sensitive data remains protected during transmission within the system.

The proposed core integrates seamlessly with the Advanced eXtensible Interface (AXI), facilitating secure data transmission between digital modules while mitigating the risk of unauthorized access and data extraction attempts. Leveraging the renowned security features of the PRESENT encryption algorithm, the core encrypts data in real-time, thereby bolstering the confidentiality and integrity of information within the system.

Designed with resource efficiency in mind, the core boasts a compact footprint, making it well-suited for integration into Nexys boards without compromising system performance. Preliminary testing on the Nexys platform has demonstrated promising results, affirming the core's effectiveness in enhancing cyber-security measures within digital systems.

Furthermore, with its exceptional performance capabilities and compatibility with Nexys boards, the core offers a versatile solution for securing data transmission in various applications where confidentiality is paramount.

In conclusion, the development of this AXI-compatible PRESENT encryption IP core represents a significant advancement in cyber-security, offering a robust solution to safeguard sensitive data during transmission within digital systems. By focusing on encryption and leveraging the inherent security features of the PRESENT algorithm, the core addresses the pressing need for enhanced data protection in today's interconnected world.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.0.1 The Rise of ECUs and the Networked Vehicle: A Modern Transformation

The early 1970s witnessed a pivotal shift in automotive design with the introduction of Electronic Control Units (ECUs). Enabled by advancements in integrated circuits and microprocessors, these miniaturized computers revolutionized vehicle functionality. Initially implemented for simple tasks like wiper activation, the number of ECUs has steadily risen, reaching up to 100 per vehicle in contemporary models. These ubiquitous units now govern nearly every aspect of the car's operation, from basic functionalities like lighting and climate control to critical safety systems like anti-lock braking and brake-by-wire technology. Furthermore, the pursuit of autonomous driving necessitates the integration of increasingly complex ECUs, such as Advanced Driver Assistance Systems (ADAS) coupled with their dedicated sensor control units (lidar, radar). This intricate network relies on wired and wireless communication protocols like CAN bus, MOST bus, FlexRay, and radio frequency (RF) to enable seamless data exchange between ECUs. Notably, sensor data plays a crucial role in ECU operation, influencing vehicle behavior in real-time. For instance, cruise control modifies speed based on sensor input received from steering wheel buttons.
The advent of affordable wireless communication technologies like Bluetooth, LTE, Wi-Fi, and RFID has spurred yet another transformation. Automotive manufacturers and OEMs leverage these technologies in ECUs to enhance both driver and passenger experience. Safety features like General Motors' OnStar system, telematics units, smartphone-vehicle speaker integration via Bluetooth, and platforms like Android Auto and Apple CarPlay exemplify this trend.
In essence, modern vehicles have evolved into sophisticated networks of mini-computers, meticulously collaborating to govern diverse functionalities. This shift necessitates a holistic understanding of ECU interactions, communication protocols, and the integration of emerging technologies for a comprehensive analysis of the contemporary networked vehicle.

1.0.2 Revealing Vulnerabilities: Unraveling the First Automotive Cybersecurity Breach

In 2015, Miller and Valasek showcased a notable automotive system hack by remotely compromising a 2014 Jeep Cherokee. They revealed how they exploited a vulnerability in the head unit to gain control over the physical functions of the driving subsystem, such as steering and braking.

The vehicle was completely remote-controlled, with no physical contact required. Miller said that he could theoretically manage any of the roughly 1.4 million automobiles in the United States, regardless of location or distance. All that was necessary was for someone to activate the car, which granted access to the system.[0.1]

1.0.3 Essential Components of Automotive Security

1. Access Control Systems:

Regulate who can access vehicle functions. Example: Role-based access control (RBAC) prevents unauthorized users from altering critical systems remotely, as demonstrated by the Miller experiment.

2. Encryption and Secure Communication Protocols:

Encrypt data to prevent unauthorized interception. Example: Secure communication protocols like TLS ensure that commands sent to the vehicle, such as those exploited in the Miller experiment, are protected from tampering.

3. Firmware and Software Security:

Ensure the integrity of vehicle software. Example: Secure boot mechanisms verify the authenticity of software updates, preventing unauthorized modifications that could enable remote manipulation, as demonstrated in the Miller experiment.

4. Physical Security Measures:

Protect vehicle components from physical tampering. Example: Tamper-resistant hardware prevents attackers from physically accessing vehicle systems to facilitate remote manipulation, as seen in the Miller experiment.

5. Regulatory Compliance and Standards:

Adhere to industry standards for cybersecurity. Example: Compliance with standards like ISO/SAE 21434 ensures that vehicles are developed with cybersecurity best practices, reducing the risk of incidents like the Miller experiment.

1.0.4 The PRESENT algorithm

Before diving into PRESENT-80 algorithm[1] it is worthy to mention some of the related works with their resource utilization here. In the landscape of block ciphers, DES stands out for its strategic design aimed at hardware efficiency, particularly tailored to the technological constraints prevalent in the early 1970s semiconductor circuits. Notably, DES exhibits commendable implementation properties, with estimates approximating around 3000 GE for a conventional implementation [1], and roughly 2300 GE for a serialized counterpart [1]. Despite its prowess, DES's utility is constrained by its key length, prompting the exploration of alternatives like DESXL (2168 GE) to mitigate this limitation .
In the contemporary cryptographic discourse, a seminal work by offers an exhaustive exploration of a low-cost AES implementation. However, the implementation requirements for

AES hover around 3600 GE, owing to Rijndael's design ethos, which prioritizes software efficiency for 8- and 32-bit processors.
While the exact hardware specifications for the Tiny Encryption Algorithm (TEA) and XTEA remain undisclosed, a rudimentary estimate suggests TEA necessitates at least 2100 GE, whereas XTEA approximates around 2000 GE.
Various proposals have surfaced for cost-effective implementations, including mCrypton , HIGHT , SEA , and CGEN (though the latter isn't primarily focused on block ciphers). Notably, mCrypton has undergone meticulous hardware assessment, revealing a requirement of 2949 GE, while HIGHT demands roughly 3000 GE. Similarly, SEA, boasting parameters akin to PRESENT, mandates approximately 2280 GE.
PRESENT is a type of block cipher known as an SP-network. It is comprised of 31 rounds and operates on 64-bit blocks. Two key lengths are supported: 80 bits and 128 bits. Despite the availability of both key lengths, for applications like tag-based deployments where security demands are relatively low, the version with 80-bit keys is recommended. This level of key length provides more than adequate security for such applications, striking a balance between security and resource efficiency. Each round of the PRESENT cipher follows this process:

1. Round Key Addition (XOR Operation):

- In each of the 31 rounds, a round key Ki (where 1 ¡ = i ¡ = 32) is XORed with the block. The key K32 is specifically used for post-whitening.

2. Linear Bitwise Permutation:

- After the XOR operation with the round key, a linear bitwise permutation is applied to the block.

3. Non-linear Substitution Layer:

Throughout the explanation, bit numbering is done from zero, with bit zero positioned on the right of a block or word. The detailed algorithm is discussed in section 3.1 .

1.0.5 AXI4-Stream Protocol

The Advanced eXtensible Interface (AXI) is a key on-chip communication protocol within the Advanced Microcontroller Bus Architecture (AMBA) specification[0.3]. Initially introduced in 2003 with AMBA3, AXI evolved with the 2010 AMBA4 revision, introducing AXI4, AXI4-Lite, and AXI4-Stream protocols.

Key Advantages:

1. Versatility: Supports multiple data streams over shared wires, enhancing system flexibility.
2. Compatibility: Accommodates a wide range of stream types, ensuring compatibility with diverse data transmission requirements.
3. Reliability: Defines clear associations between transfers and packets, facilitating seamless and reliable data exchange.
4. Simplicity: Simplifies system integration by providing a standardized framework for connecting various master and slave components.

5. Scalability: Enhances scalability by enabling the connection of multiple masters and slaves within the system.

1.0.6 IP-Cores

An IP core, or Intellectual Property core, is a pre-designed and pre-verified block of digital logic or functionality that can be integrated into larger electronic designs or systems. It encapsulates specific intellectual property, such as a processor, memory controller, or communication interface, into a reusable and configurable module. Typically described in a hardware description language (HDL) like Verilog or VHDL, an IP core consists of RTL code detailing its behavior, inputs, outputs, internal logic, and timing requirements. These cores are technology-independent and can be synthesized to target various FPGA or ASIC technologies, offering flexibility and scalability in system design. ex:
Provided by third-party vendors or developed in-house, IP cores streamline system development by enabling rapid integration of complex functionalities. Designers can customize and configure IP cores to meet specific requirements through parameters or interface options provided by the IP vendor. This approach significantly reduces development time and risk, allowing designers to focus on higher-level system integration and innovation while leveraging proven IP cores for critical functionalities. There are number of tools for easy development of these soft IP-Cores. ex: Vivado-Vitis

1.1 Motivation

The integration of AXI4 Stream with the PRESENT-80 encryption algorithm presents a significant opportunity to streamline system connectivity and enhance versatility in embedded systems. By incorporating the AXI4 Stream interface, our project facilitates seamless integration with various peripherals and components, simplifying system architecture and reducing development complexity. This integration not only simplifies connectivity but also ensures compatibility with a diverse range of system configurations, enabling effortless interoperability within larger systems. Moreover, the utilization of AXI4 Stream enables efficient handling of smaller width but stream data, optimizing resource utilization and maximizing data throughput. Overall, this integration represents a strategic approach to system design, offering improved connectivity, scalability, and efficiency while advancing the reliability and resilience of embedded systems technology.

1.2 Objectives

1] Ensure a understanding of the PRESENT-80 algorithm, including its fundamental components and encryption/decryption methods.
2] Adapting the PRESENT-80 algorithm to suit the Nexys 7 FPGA architecture effectively while optimising for hardware implementation.
3] Optimise resource allocation on the FPGA board to save space while maximising performance, taking into account logic parts, memory blocks, and routing resources.
4] Consider the FPGA board's power consumption and temperature restrictions throughout the implementation phase to guarantee that the design functions safely and reliably over long periods of time.

1.3 Literature survey

- **PRESENT: An Ultra-Lightweight Block Cipher[1]** :- An overview of the body of available literature is given in this study. Other situations have given similar designs some thought. Students might utilize the current study's design as a tutorial. The report provides an overview of the state of the field's research at the moment. It talks about several strategies and tactics that have been employed in the literature. The study emphasizes how crucial it is to take various contexts into account while creating systems that are similar. The design's teaching element qualifies it for educational use, especially for pupils who are enthusiastic about the subject.

- **Enhancing PRESENT-80 and Substitution-Permutation Network Cipher Security with Dynamic "Keyed" Permutation Networks[2]**:- A chosen-plaintext attack called DPA compares variations in output to variations in input. SCADPA uses the PRESENT-80 cipher's permutation networks' non-random nature to its advantage. When a microcontroller is operating, its electromagnetic emissions are measured by SCADPA. To record the round output differential for analysis, SCADPA employs selected plaintext values. Non-random behavior is evident in the SBox and PBox data of PRESENT-80.

- **Present-80 Encryption Algorithm Implementation on GPRS Arduino Mega-2560 Cyber Physical Tracking System[3]**:-SatGen3 Simulator is used to mimic VTS devices and Arduino Uno is used for tracking. For communication purposes, VTS is limited to 1.6 km. Firebase and the Android platform work well with VTS applications. With 31 rounds, the encryption technique known as PRESENT-80 is a simple block cipher. Using encryption techniques to increase security during device communication. A simple cryptography technique is required for embedded systems.

- **Light-Weight Present Block Cipher Model for IoT Security on FPGA[4]**:-Explains many cipher algorithms in detail as well as the most modern PRESENT lightweight algorithms. Explains how ciphers are implemented on hardware and software platforms. Evaluates the effectiveness of hardware-based ciphers in relation to energy, area, power, and throughput. Evaluates the PRESENT algorithm's performance on the FPGA platform. Evaluates and contrasts several lightweight block ciphers' performance metrics. Introduces a novel lightweight hybrid cryptosystem designed for Internet of Things infrastructure. Contrasts the AES and PRESENT block ciphers' performance for Internet of Things applications. Explains the application of lightweight ciphers in industrial WSN and provides mathematical evidence. Examines the PRESENT and CLEFIA lightweight block ciphers' performance measures. Explains how to use ASIC platforms to construct lightweight ciphers with improved limitations.

- **Implementation of Lightweight Cryptography Core PRESENT and DM-PRESENT on FPGA[5]**:-The study looks into building the PRESENT and DM-PRESENT lightweight cryptography cores on Intel FPGAs and shows how resource-efficient they are in comparison to other methods. The PRESENT core requires 2,945 logic elements, 1,824 registers, and 273,408 memory bits for a 64-bit block and 80-bit key, while the DM-PRESENT core uses 2,336 logic elements, 1,380 registers, and 273,408 memory bits, according to the comprehensive analysis in Section III. It covers the broader field of lightweight cryptography as well, classifying it into four main groups. The study concludes by highlighting the significance of these low-cost cryptographic techniques in scenarios when resources are limited in Section IV.

- **Performance Evaluation of the Present Cryptographic Algorithm over FPGA[6]**:-The research examines the performance of the Present cryptographic algorithm on FPGA, highlighting its flexibility to reconfigurable systems. Its authors, Edwar Gomez, Cesar Hernández, and Fredy Martinez, provide a complete assessment of the encryption technique, including research, design, implementation, and testing. Notably, the Present algorithm is a unique block-based encryption approach that has yet to be breached. The study emphasises its potential as a strong cryptographic solution appropriate for FPGA implementation. The authors' comprehensive study gives information on the algorithm's effectiveness and suitability for practical applications. This paper makes a substantial addition to the continuing investigation of lightweight encryption algorithms on FPGA systems.

- **FPGA IMPLEMENTATION OF PRESENT ALGORITHM WITH IMPROVED SECURITY[7]**:-The research focuses on improving the security and efficiency of the PRESENT method, which is suited for resource-constrained devices such as IoT and RFID tags, using FPGA implementation. It presents a revolutionary 4-bit iterative architecture that is optimised for resource conservation and is especially well-suited to such devices. Additionally, a redesigned key-dependent architecture is presented to improve security against possible data breaches in IoT contexts. The design has a loop-based paradigm with 16 S-boxes for increased security and achieves a latency of 51 or 63 cycles while including important components such as flip-flops, XOR gates, and specialised modules for shifting without extra hardware. This study aims to improve the PRESENT algorithm's usability in lightweight cryptographic applications by addressing resource restrictions in various device deployments.

- **Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT[8]**:-The authors provide new attacks on PRESENT, including 128-bit keyed version, which uses a 17-round related-key rectangle technique. Furthermore, they investigate HIGHT's resistance to impossible differential attacks, showing novel 26-round and 31-round related-key impossible differential assaults. The difficulty of related-key attacks on PRESENT is around 2104 memory accesses, whereas refined attacks on HIGHT need 2119.53 reduced round HIGHT evaluations.

- **On the primitivity of PRESENT and other lightweight ciphers[9]**:-The study investigates the features of the PRESENT cypher, with a particular emphasis on its mixing layer and S-Box. Despite not achieving all of the conditions given in Theorem 2.4, the PRESENT mixing layer is recognised for being highly proper, with its S-Box demonstrating weak 4-uniformity and strong 1-anti-invariance. Despite these departures from the hypothesis, the group created by the PRESENT round functions is considered basic, demonstrating its usefulness in cryptographic applications. Furthermore, the research investigates instances in which the group G is a wreath product, giving information on the order of its subgroups Ti. Overall, the investigation illuminates the cryptographic strength and features of the PRESENT 80 algorithm.

- **A low cost and portable mini motor car system with a BNN accelerator on FPGA[10]**:-Field Programmable Gate Arrays (FPGAs) are particularly well-suited for building Binarized Neural Networks (BNN), according to research initiatives. Frameworks like as FINN and GUINNESS make it easier to create FPGA-based DNN accelerators, such as BNN, with fewer parameters and no need for off-chip memory, lowering latency and power consumption. Furthermore, plans for self-driving vehicles with FPGA implementations seek to expand research and educational prospects, however commercial choices like as the ad-refket kit may be prohibitively costly.

- **A Power Efficient Neural Network Implementation on Heterogeneous FPGA and GPUDevices[11]**:-The research provides a literature assessment of FPGA-centric processing for embedded multimedia applications, highlighting its power efficiency and quicker inference than GPUs. It covers FPGA programming issues and focuses on effective implementation in heterogeneous contexts, arguing for model parallelism and automated device placement techniques. The proposed framework uses FPGAs for fully connected layers and GPUs for floating-point operations to optimise computation speed and power efficiency, with an architecture inspired by LeNet-5 and device selection based on power consumption and modularity benefits, using Artix-7 FPGAs and Nvidia Jetson TX2 GPUs.

- **Flexible structures of lightweight block ciphers PRESENT, SIMON and LED[12]**:-The research examines the flexible and high-throughput hardware architectures of lightweight block cyphers such as PRESENT, SIMON, and LED designed for IoT applications. These architectures enable changing key sizes, which improves flexibility to security needs. Implementation findings show improved area, throughput, and throughput/area metrics compared to previous efforts, with a special emphasis on the often ignored issue of structural flexibility. It also discusses the essential issues of security and privacy in IoT systems, highlighting the acceptable security levels provided by PRESENT, SIMON, and LED cyphers in cryptographic applications.

- **FPGA Random Number Generator[13]**:-The article discusses the importance of random number generation in different applications and presents a Verilog-based FPGA hardware architecture for synchronous 32-bit random number generation. Its system architecture consists of ADC-based voltage measurement, a random processing unit, a 7-segment display, and a UART transmit module. The FPGA-based generator uses linear feedback algorithms to generate continuous streams of random numbers, which are then analysed for statistical randomness using Python analysis, while also admitting the possibility of using true-random numbers from physical sources such as radioactive decay or electrical noise.

- **Linear Cryptanalysis of Reduced-Round PRESENT[14]**:-The research undertakes a literature review on the PRESENT block cypher, applying multidimensional linear cryptanalysis to extract its 80-bit secret key with 25 rounds out of 31. The data complexity is roughly 262.4. It proves the effectiveness of this attack, outperforming key exhaustive search even for the 26-round version of PRESENT, and asserts supremacy over earlier attack techniques. Furthermore, the article applies the multidimensional linear assault to reduced PRESENT versions, demonstrating its superior performance over traditional linear attacks and offering insights into the PRESENT structure and attack framework.

1.4 Problem statement

Integrated Secure Data Encryption using AXI4-Stream and PRESENT Algorithm

1.5 Application in Societal Context

Implementing the PRESENT-80 algorithm on the Nexys 7 FPGA device achieves many goals with important social implications. The major goal is to efficiently implement the algorithm, optimise resource utilisation, and evaluate performance. This approach improves data security, especially in environments where mobility and intelligent technologies are present. By

safeguarding communication routes and data transfers, it helps to make systems safer and more dependable, lowering exposure to cyber attacks. Furthermore, FPGA-based encryption protects sensitive information and fosters user confidence. Overall, its implementation improves the security and dependability of systems in the mobility and intelligent infrastructure sectors, which benefits societal well-being.

1.6 Project Planning and bill of materials

Project planning: After reviewing relevant research papers, a comprehensive understanding of the project's scope and requirements has been attained. Planning a cryptography project involves several essential steps to ensure its success. Firstly, clear project objectives must be specified, outlining the intended outcomes and deliverables. Establishing the project's scope delineates the boundaries and defines the extent of work to be undertaken. Allocation of resources, including human resources, time, and budget, is crucial for effective project execution.
Moreover, it is imperative to thoroughly review the documentation provided for AXI4 Stream and the Vivado design tool. These resources offer invaluable insights and guidance for integrating the AXI4 Stream interface and navigating the development process within the Vivado environment. By familiarizing oneself with these documents, project stakeholders can gain a deeper understanding of the technical specifications and best practices, thereby facilitating smoother implementation and mitigating potential challenges. Identifying potential hazards and risks is also crucial to anticipate and mitigate any challenges that may arise during development. Maintaining regular oversight of project development ensures progress aligns with established timelines and objectives. Lastly, meticulous documentation of project results is essential for accountability and future reference. By meticulously considering these aspects, the project planning process lays the groundwork for achieving the desired goals and objectives while ensuring the successful execution of the cryptography project.

Bill of Materials (BOM):
Nexys 7 FPGA board.
Xilinx Vivado 2023.2 (software tool)

1.7 Organization of the report

Name of chapter 2 and brief description about it
Name of Chapter 3 and brief description about it and so on

Chapter 2

System design

2.1 Functional block diagram

The data enters the system via the INPUT AXIS4 port, is buffered in the FIFO, and then processed by the SIPO block. The processed data is then placed in the OUTPUT AXI4-Stream FIFO before being transferred from the system.

1. **INPUT AXIS4:** This is the system's input port, which accepts data via an AXI4-Stream interface. AXI4-Stream is a high-performance, point-to-point interface that is widely used in embedded systems to transport data between components.
2. **A First-In, First-Out buffer (FIFO):** holds data before it is processed. FIFOs are often used to smooth the flow of data between components with varying speeds or data consumption rates.
3. **SIPO:** This block translates serial data to parallel data. Serial data is transferred one bit at a time, while parallel data is sent many bits at once.
4. **PRESENT-80:** The data is then sent via the Present 80 algorithm.
5. **OUTPUT AXI4-Stream FIFO:** This is the system's output port, which stores processed data before it is transferred.

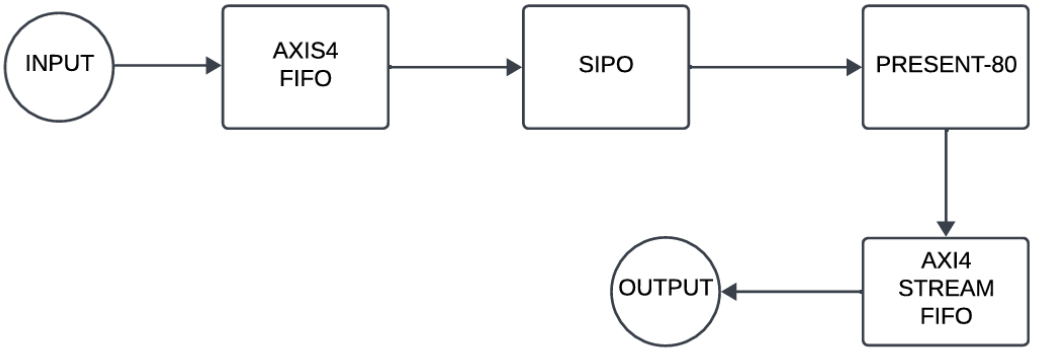

Figure 2.1: Functional Block Diagram

2.2 Final design

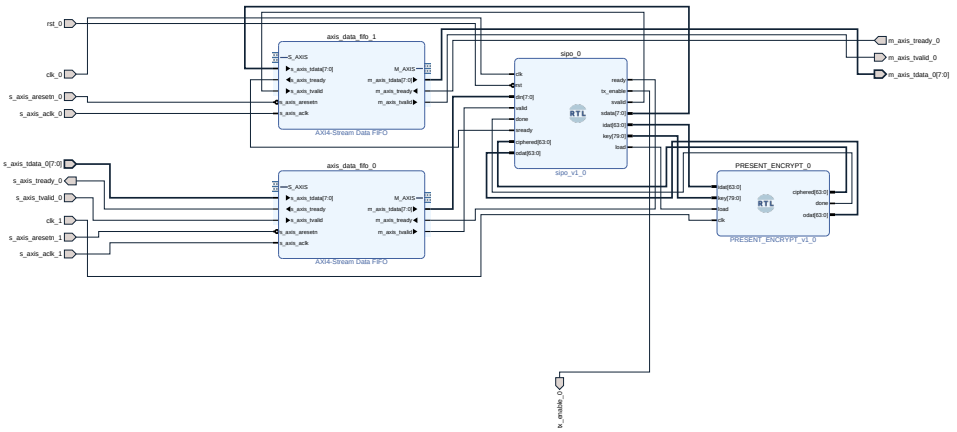

Figure 2.2: Final Design for Implementation

Chapter 3

Implementation details

¡return¿ In this chapter we discuss the function flow,architecture, algorithms, incorporating systems and subsystems , related accessories in detail.

3.1 Specifications and final system of architecture

3.1.1 Subsystems

Here we are discussing the subsystems required to build our final systems. Our systems mainly consists of :
1. PRESENT-80 encryption engine
2. AXI4 Stream data FIFO
Also it consists of a Serial-In-Parallel-Out register which acts as a buffer and data width converter (from 8-bit to 64-bit) at input end and a Parallel-In-Serial-Out at the output end for the same purpose. we'll discuss them in detail in following subsubsections.

PRESENT-80

The PRESENT block cipher algorithm, which consists of the Substitution-Permutation network (henceforth referred to as the "SP network"), executes 31 rounds of operations to produce the ciphertext. Key lengths of 80 or 128 bits are allowed, however 64 bits of input data is the minimum required. Experts advise 80-bit key length for effective performance since real-world applications have limited resources [present-original-paper]. An XOR operation introduces a round key Ki for 1 i 32 in each of the 31 rounds, and K32 is used for post-whitening. A non-linear substitution layer and a linear bitwise permutation come next. One 4-bit S-box S is used by the non-linear layer, and it is applied 16 times in parallel throughout each round. Following a standard numbering scheme, bits are numbered starting at zero, with bit zero located on the the word or block's right.

RoundKeyaddition: To add the round key Ki = i 63... i 0 for 1 i ¡ 32 and the current state b63... b0, do the operation bj → bj i j. sBoxlayer. The present S-box is a 4-bit to 4-bit S-box (F 4 2 → F 4 2). The operation of this box in hex notation is indicated by the table below.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Figure 3.1: Action of S-box in hexadecimal notation

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Figure 3.2: The bit permutation used in PRESENT

In the sBoxLayer, the current state b63 . . . b0 is divided into sixteen 4-bit words w15 . . . w0. Each word wi is formed by concatenating the bits b4i+3, b4i+2, b4i+1, and b4i, where i ranges from 0 to 15. The updated state values are obtained by using the output nibble S[wi] in a straightforward manner. Player. The bit permutation used in the current scenario is shown in the following table. The bit at location i in the state is shifted to position P(i) The key schedule accepts keys with a length of either 80 or 128 bits. Our primary emphasis is on the version that utilizes 80-bit keys. The key provided by the user is kept in a key register called K and is represented as k79k78 . . . k0. During round i, the 64-bit round key Ki is formed by taking the 64 leftmost bits of the current contents of register K, represented as 6362 . . . 0. Therefore, at round i, we may express Ki as the concatenation of 6362 . . . 0, which is equivalent to k79k78 . . . k16. After the round key Ki is extracted, the key register K = k79k78 . . . k0 is modified.
1. [k79k78 . . . k1k0] = [k18k17 . . . k20k19]
2. [k79k78k77k76] = S[k79k78k77k76]
3. [k19k18k17k16k15] = [k19k18k17k16k15] round_counter

AXI4-Stream data FIFO

This is a ready made IPcore which is avilable in Vivado, which employs the AXI4 stream protocol for stream data. Here we are using it in 8-bit data width mode. But there are multiple data width options are available . Also the developer provided more user friendly options like packet mode, r/w count, user bits etc. But we are not using any of them. Following are the pins and their functions:

$\text{axis}_t data : -dataport axis_t ready : -readytoreceivedata$

axis-tvalid:- valid data available Here for every pin the prefix "s" refers that it is slave and "m" refers that it is master. In AXI, a master always sends the data and slave always receives the data. The configurations we are using are:

tdata width=8-bits

Packet Mode= No Thus, this AXI4-Stream data ip core reduced the data pins from 64 pin to 10 pins which will helps the system to use lesser no. of IO resources and hence the power consumption.

SIPO and PISO

These two registers plays vital role in converting the streamlined data into 64-bit wide data as this datawidth is required for the encryption algorithm. At the input side, after the axi core recieves data, SIPO will convert the data to 64 bits and at the output side PISO will convert

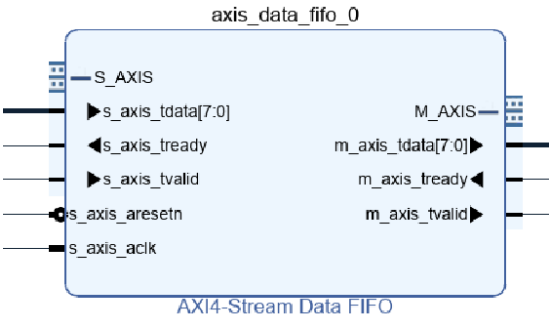

Figure 3.3: AXI4 Stream Data FIFO

64 bit wide ciphertext into 8 bit streamline data and feed it to axi master. This can be seen clear in flowchart.

3.2 Flowchart

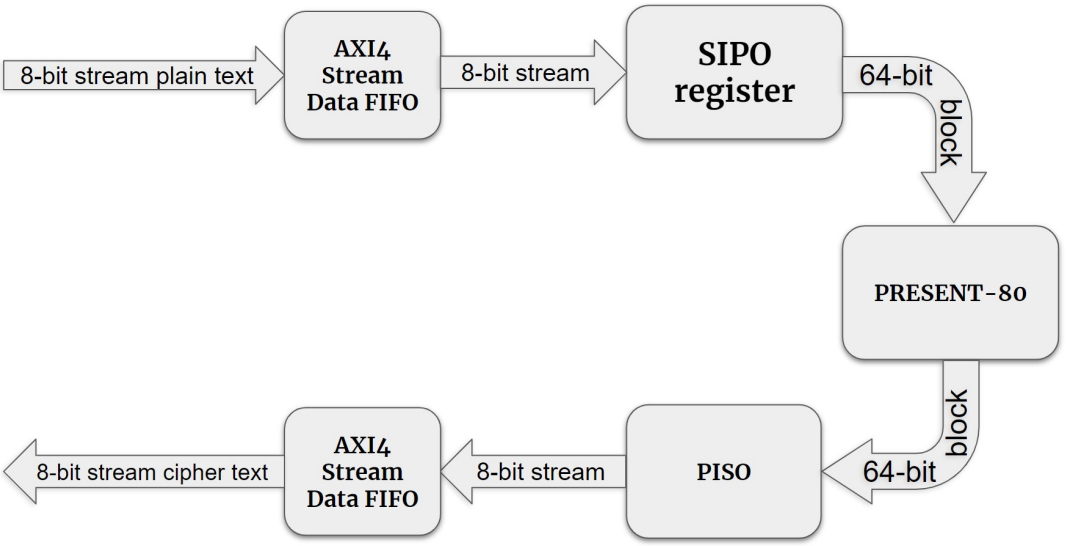

Figure 3.4: Flow chart of system architecture

This logic flow has been implemented using Vivado 2023.2 software tool in verilog language. Here we are considering the constraint that key is fixed.

Chapter 4

Optimization

In this chapter we discuss on various optimization technics and the one we used.

4.1 Introduction to optimization

Efficient implementation of cryptographic algorithms on FPGA platforms demands meticulous optimization to balance performance and resource constraints. In this section, we delve into strategies aimed at enhancing the implementation of the PRESENT-80 algorithm on the Nexys 7 FPGA board. Through a combination of architectural refinements and algorithmic optimizations, we endeavor to maximize throughput, minimize resource utilization, and optimize power consumption. By elucidating the rationale behind each optimization approach and evaluating their impact on performance metrics, this section provides valuable insights into the optimization methodologies employed, paving the way for advancements in lightweight cryptography and hardware security on FPGA platforms.

4.2 Types of Optimization

Pipeline Design: Implementing a pipelined architecture can significantly improve throughput by breaking down the encryption process into smaller stages and allowing multiple encryption operations to be processed simultaneously. Careful consideration should be given to balancing pipeline stages to minimize latency and maximize throughput.
Resource Sharing: Identify opportunities for resource sharing to minimize the utilization of FPGA resources. For instance, reusing hardware modules for common operations, such as bitwise operations or data permutation, across different stages of the algorithm can reduce the overall resource footprint.
Bit-Level Optimization: Leveraging the inherent parallelism in FPGA architectures, consider optimizing operations at the bit level to maximize resource utilization and improve performance. Techniques such as bit-slicing can be employed to process multiple data bits in parallel, enhancing throughput.
Memory Optimization: Efficient management of on-chip memory resources, such as Block RAM (BRAM), can contribute to overall optimization. Utilize memory hierarchy effectively, minimizing access latencies and maximizing data throughput.
Clock Frequency Optimization: Fine-tune the clock frequency of critical paths within the design to achieve optimal performance while meeting timing constraints. Utilize FPGA synthesis tools to perform timing analysis and identify areas for clock frequency optimization
Power Optimization: Implement power-aware design techniques, such as clock gating and dynamic voltage and frequency scaling (DVFS), to minimize power consumption while

maintaining desired performance levels. Utilize FPGA power analysis tools to identify power-hungry components and optimize accordingly.

4.3 Selection and justification of optimization method

Here we are using AXI4 Stream data FIFO to provide input and drawing the output obtained in streamlined manner, below is the brief explaination about this approach:
A FIFO is a hardware structure used to temporarily store data in a first-in, first-out order. The AXI FIFO acts as a buffer, allowing a smoother exchange of data between components by accommodating temporary differences in data arrival or processing rates. Thus the AXI4 Stream data FIFO ip core takes input plaintext in serial manner, and stores it in its fifo buffer. Thus it minimises the requirement of 64 pins for taking data input. **When we analysed the power consumption of PRESENT core alone, it was reaching about 72 W and exceeding the junction temperature for safe operation. And it was mainly due to over usage of IO resources which require heavy power. When we integrated the AXI4 stream ip core at the input end, it was reduced by almost 3 times and after integrating at output side another 3 times reduced .** Thus power requirement reached the safe zone. This integration not only helps to power reduction, it also helps to make the system compatible with other systems. Thus, following additional advantages are obtained:

1. Streamlined Data Transfer: AXI Stream concentrates on the continuous flow of data, eliminating the overhead associated with address channels in other AXI protocols. This corresponds nicely with the sequential structure of encryption procedures, maximizing data transfer.

2. Smooth Flow:- The protocol includes signals, such as tvalid and tready, to provide flawless coordination between the data producer (AXI Stream Data FIFO) and the consumer (SIPO AXI 0). This avoids data loss or overruns.

3. Burst Capability:- AXI Stream allows burst transfers, allowing the transfer of several data words in fast order. This functionality supports to optimizing throughput and lowering delay in the encryption process.

   The efficiency and flexibility of the AXI Stream protocol contribute greatly to the overall performance and reliability of this PRESENT 80 FPGA implementation.

Chapter 5

Results and discussions

In this chapter we discuss on results obtained from the project and its related analysis.

5.1 Result Analysis

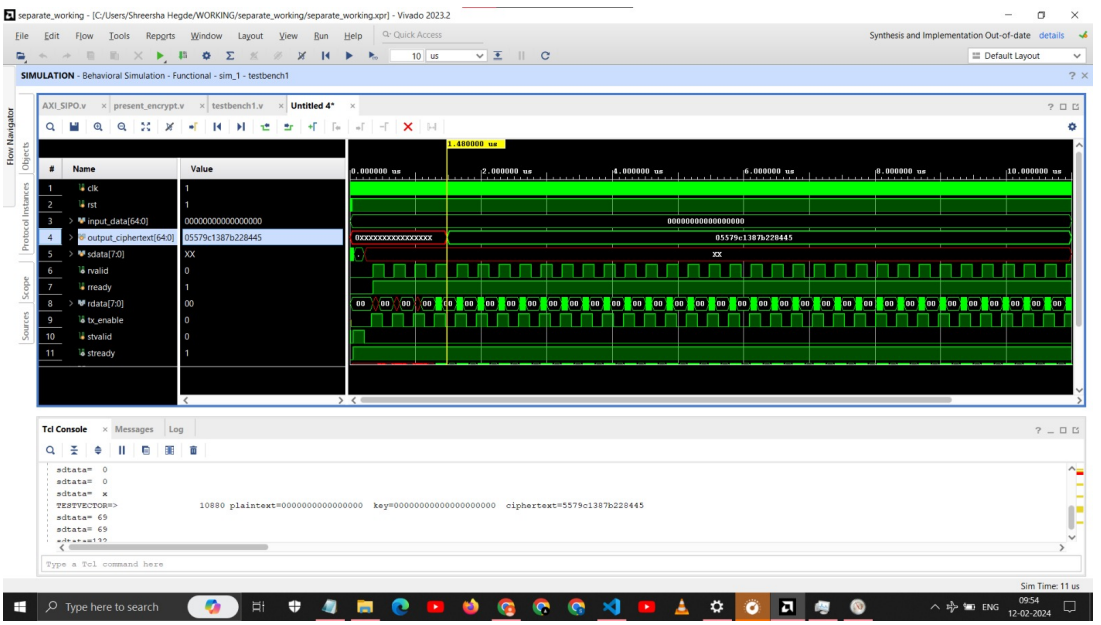

Figure 5.1: The simulation testing of AXI Stream Interconnect (SIPO) design

Above figure shows the simulation results for

plaintext = 64'h00000000000000;

key = 80'h00000000000000000000; As the fig shows, from initializing the circuit to complete the encryption and getting out of AXI peripheral, it just takes 1.4 us where clock frequency=100 Mhz. Thus we can achive cipher decipher process in very minimum time which is helpful in time critical applications like access authorization, real time data transfer etc. The system has been tested for many more inputs and the resulting ciphertext has been verified with ground truth.

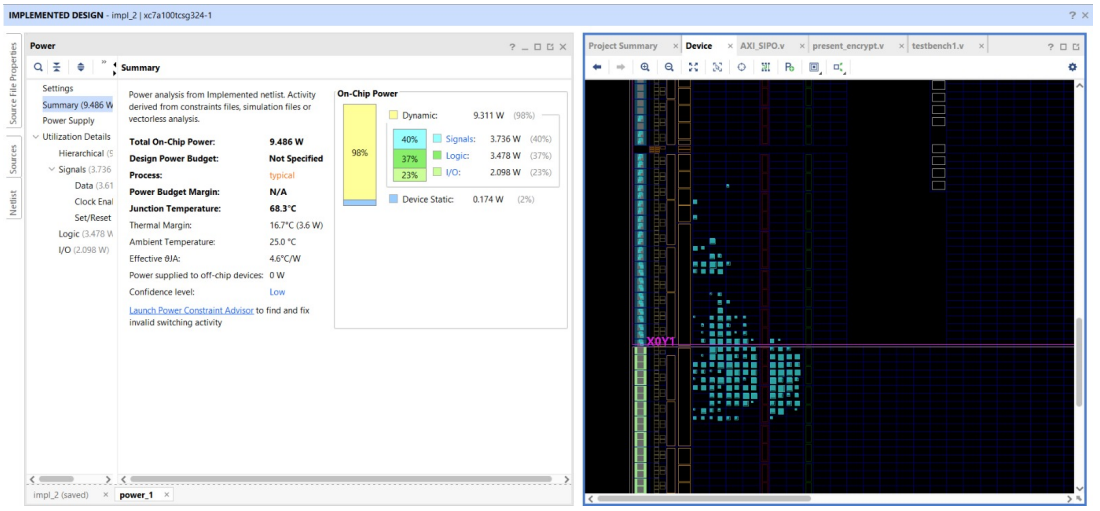

Figure 5.2: Power consumption of the implemented design

Total ON-Chip Power	9.486 W
Design Power Budget	Not Specified
Process	typical
Power Budget Margin	N/A
Junction Temperature	68.3 degree Celsius
Thermal Margin	16.7 degree Celsius (3.6 W)
Ambient Temperature	25.0 degree Celsius
Effective 0JA	4,6 degree Celsius/W
Power supplied to off-chip devices	0 W
Confidence level	Low

Table 5.1: Power Analysis from Implemented Netlist

The report shows the power consumption of the implemented design. The total on-chip power is 9.486 W,. The largest contributor to the on-chip power is the logic (47%), followed by the signals (40%) and the I/O (23%). The junction temperature of the device is 68.3°C, which is 16.7°C below the thermal margin. The ambient temperature is 25°C. Also, the following table shows the utilized resources on chip.

Resources	Utilization	Availability	Utilization Percentage
LUT	326	63400	0.51
LUTRAM	16	19000	0.08
FF	545	126800	0.43
IO	28	210	13.33

Table 5.2: Resource utilization Table

- Resource: This column shows the type of resource being used. The four resources listed in the table are:
- LUT: Look-up tables, which are the basic logic blocks of an FPGA.
- LUTRAM: Block RAM, which is a type of memory that can be used to store data on an FPGA.

- FF: Flip-flops, which are used to store data that changes over time.
- BRAM: Block RAM, which is another type of memory that can be used to store data on an FPGA.
- Utilization: This column shows the number of units of each resource that are being used by the design. For example, the table shows that the design is using 326 LUTs, 16 LUTRAMs, 545 FFs, and 28 BRAMs.
- Available: This column shows the total number of units of each resource that are available on the FPGA. For example, the table shows that there are 63400 LUTs, 19000 LUTRAMs, 126800 FFs, and 210 BRAMs available on the FPGA.
- Utilization : This column shows the percentage of each resource that is being used by the design. For example, the table shows that 0.51 of the LUTs, 0.08 of the LUTRAMs, 0.43 of the FFs, and 13.33 of the BRAMs are being used by the design.

Finally the system has been packed with Vivado IP integrator tool and been uploaded to github as an opensource repository.

Chapter 6

Conclusions and future scope

In this chapter we discuss on the conclusion we arrived at and future scope for the developed system.

6.1 Conclusion

The PRESENT-80 encryption algorithm is renowned for its potent combination of strength and efficiency, making it ideal for time-sensitive and resource-constrained applications. By integrating with the AXI4-Stream protocol, PRESENT-80 achieves enhanced energy and resource efficiency, streamlining its integration with a diverse array of in-chip peripherals. This synergy not only amplifies the algorithm's performance but also facilitates seamless collaboration with other components within the chip, ultimately bolstering the overall efficacy of the encryption process.

6.2 Future scope

Stepping ahead, the project implementing the PRESENT-80 algorithm on the Nexys 7 FPGA board shows great potential for future improvement and extension. One route of investigation is to include more powerful encryption standards such as Advanced Encryption Standards (AES), which cater to circumstances that need greater degrees of protection. Furthermore, optimising the implementation for low-power devices by decreasing input data width in AXI may increase its usefulness in energy-efficient computing contexts such as IoT and mobile devices. Scaling the approach to bigger FPGA platforms or using parallel processing might improve hardware acceleration for high-throughput applications like data centres and cloud computing. Also developing it for taking 80/128 bit key as second input can make it more robust by assigning the key dynamically. Furthermore, integrating with future technologies such as blockchain or edge computing allows you to solve new security concerns and abuse creative designs. Continuous vulnerability research and countermeasure development will be critical to ensuring the system's resilience against changing cyber attacks. Seeking standardisation and certification for the solution may encourage wider acceptance across industries, while investigating cross-domain applications outside of established sectors may reveal new use cases and consumers. By following these options, the project may maintain its position at the guard of encryption technology, responding to new demands and enhancing the security environment.

References

1. Bogdanov, Andrey, et al. "PRESENT: An ultra-lightweight block cipher." Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9. Springer Berlin Heidelberg, 2007.

2. Lewandowski, Matthew, and Srinivas Katkoori. "Enhancing PRESENT-80 and Substitution-Permutation Network Cipher Security with Dynamic" Keyed" Permutation Networks." 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2021.

3. Novazrianto, Dwi, Rini Wisnu Wardhani, and Naufal Hafiz Syahidan. "Present-80 Encryption Algorithm Implementation on GPRS Arduino Mega-2560 Cyber Physical Tracking System." 2021 9th International Conference on Information and Communication Technology (ICoICT). IEEE, 2021.

4. Bharathi, R., and N. Parvatham. "Light-Weight Present Block Cipher Model for IoT Security on FPGA."Intelligent Automation and Soft Computing 33.1 (2022).

5. Lam, To-Nguyen, and Duc-Hung Le. "Implementation of Lightweight Cryptography Core PRESENT and DM-PRESENT on FPGA." 2022 International Conference on Advanced Technologies for Communications (ATC). IEEE, 2022.

6. Gomez, Edwar, Cesar Hernández, and Fredy Martinez. "Performance evaluation of the present cryptographic algorithm over FPGA." Contemp. Eng. Sci. 10 (2017): 555-567.

7. Reddy, Vallish Kumar, et al. "FPGA implementation of present algorithm with improved security." 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2019.

8. Onur Ozen1, Kerem Varıcı, Cihangir Tezcan, and Celebi Kocair. "Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT" Conference Paper · January 2009

9. Riccardo Aragona, Marco Calderin, Antonio Tortora and Maria Tota. " On the primitivity of PRESENT and other lightweight ciphers"

10. Fumio Hamanaka, Takuto Kanamori, and Kenji Kise " A low cost and portable mini motor car system with a BNN accelerator on FPGA" 2021 IEEE 14th International Symposium on Embedded Multicore.

11. Yuexuan Tu, Saad Sadiq, Yudong Tao, Mei-Ling Shyu and Shu-Ching Chen " A Power Efficient Neural Network Implementation on Heterogeneous FPGA and GPUDevices" 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Sciences(IRI).

12. Y Bahram Rashidi " Flexible structures of lightweight block ciphers PRESENT, SIMON and LED"

13. Jacob Hammond " FPGA Random Number Generator" May 4th, 2022.

14. Joo Yeon Cho "Linear Cryptanalysis of Reduced-Round PRESENT"