cryptography

Article

High Throughput PRESENT Cipher Hardware Architecture for the Medical IoT Applications

Jamunarani Damodharan [1,*], Emalda Roslin Susai Michael [2] and Nasir Shaikh-Husin [3]

1 Faculty of Electronics Engineering, Sathyabama Institute of Science and Technology, Chennai 600119, India
2 Department of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Chennai 600119, India
3 School of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia
* Correspondence: jamunarani.scholar@gmail.com

Abstract: The Internet of Things (IoT) is an intelligent technology applied to various fields like agriculture, healthcare, automation, and defence. Modern medical electronics is also one such field that relies on IoT. Execution time, data security, power, and hardware utilization are the four significant problems that should be addressed in the data communication system between intelligent devices. Due to the risks in the implementation algorithm complexity, certain ciphers are unsuitable for IoT applications. In addition, IoT applications are also implemented on an embedded platform wherein computing resources and memory are limited in number. Here in the research work, a reliable lightweight encryption algorithm with PRESENT has been implemented as a hardware accelerator and optimized for medical IoT-embedded applications. The PRESENT cipher is a reliable, lightweight encryption algorithm in many applications. This paper presents a low latency 32-bit data path of PRESENT cipher architecture that provides high throughput. The proposed hardware architecture has been implemented and tested with XILINX XC7Z030FBG676-2 ZYNQ FPGA board 7000. This work shows an improvement of about 85.54% in throughput with a reasonable trade-off over hardware utilization.

Keywords: field programmable gate array; lightweight cryptography; PRESENT block cipher; symmetric-key encryption; throughput

check for updates

Citation: Damodharan, J.; Susai Michael, E.R.; Shaikh-Husin, N. High Throughput PRESENT Cipher Hardware Architecture for the Medical IoT Applications. *Cryptography* **2023**, *7*, 6. https://doi.org/10.3390/cryptography7010006

Academic Editors: Ki-Hyun Jung and Luis Javier García Villalba

Received: 26 December 2022
Revised: 16 January 2023
Accepted: 17 January 2023
Published: 6 February 2023

1. Introduction

In 2030, more than 50 billion intelligent medical devices are expected to communicate through the internet between various continents [1]. Hence there is an urgency that a lightweight security algorithm must be adopted to transfer sensitive information from one medical device to another. The existing security algorithms, namely Advanced Encryption Standard (AES) [2], Elliptic Curve Cryptography (ECC) [3], Data Encryption Standard (DES) [4], and Blowfish [5], are not applicable for certain medical IoT constraints. Moreover, the lightweight technique of cryptography algorithms plays a predominant role in the data security of IoT [6,7].

To provide fast operation, the security algorithm can be implemented using FPGA devices [8,9]. The FPGA architecture allows security algorithms to be processed in parallel, improving the system's overall throughput. However, even with parallel processing capability, the encryption algorithms must be lightweight for IoT devices. The following are examples of lightweight encryption algorithms that are suitable for FPGA implementation: PRESENT cipher [10], PRINCE cipher [11], SIMON cipher [12], GIFT cipher [13,14], SKINNY cipher [15,16], PHOTON cipher [17], TWINE cipher [18], and SPECK cipher [19]. This paper focuses on improving the PRESENT cipher hardware architecture for IoT applications. Many improvements have been made to the PRESENT cipher algorithm based on previous studies [20–27].

Medical-related IoT services handle personalized data irrespective of location. The biological parameters like blood pressure, pulse, glucose, and temperature should reach the health care service in less than 3 s, and the data rate needed is 80–800 bps [28]. The medical emergency data like respiration rate, Spo2 level, and ECG variations should reach a stipulated location of less than 300 ms, and the data rate should be 50–120 bps [28]. The ECG time limit for three lead data rate is 2.4 kbps, the 5-lead data rate is 10 kbps, and the 12-lead data rate is 12 kbps [28]. The proposed architecture throughput is 692.846 Mbps which renders a data rate more than the required level. The delay for the proposed work is 92.46 ns, which is significantly less than the biomedical data available [28]. The paper augments the hardware architecture through PRESENT cipher for IoT applications.

The proposed architecture is suitable for certain dedicated applications in medical IoT. The optimization algorithm has been designed in such a way that the infusion should not affect the human body, and it should be non-invasive. Thus, the proposed low latency 32-bit data path architecture focuses on developing an optimized hardware architecture for encrypting the input message. In today's world of IoT, medical monitoring data has been transferred from medical implant devices through wireless technology. The data should be securely transmitted, and the implant devices are very much miniature with high speed. Hence to target this requirement with the help of Xilinx Vivado. It has a very high throughput required for medical data transfer by compromising the power concerning resources (number of look-up tables).

The paper is structured as follows: a literature review of the existing PRESENT cipher algorithm conversed in related works under Section 2. Section 3 highlights the overview of the symmetric-key block cipher-PRESENT cipher of the SP-Box Algorithm. Section 4 focused on implementing the existing 16-bit data path and key schedule architecture. The proposed low latency 32-bit data path and key schedule architecture implementation are dealt with in Section 5, and Section 6 compares the architecture in Results and Discussion. The conclusion and direction for subsequent work are furnished in Section 7.

2. Related Works

Bogdanov et al. [10] proposed a PRESENT cipher SP-box (substitution–permutation) algorithm for the 80-bit key in lightweight applications. The cipher has 31 rounds for a 64-bit data path architecture. The post-whitening technique is adopted to increase security against the Brute force attack. The substitution box (S-box) contains the four-bit hexadecimal number, followed by the bitwise permutation (P-box). The S-boxes were accessed parallel in a manner for each round of operation. The 64-bit data path architecture utilizes 32-bit XOR, 32-bit adder, and 192 registers which comprise 80, 148, and 1344 gates, respectively. Bogdanov et al. [10] implemented the PRESENT cipher SP-box algorithm using the Mentor Graphics and synopsis design compiler software, resulting in 1570 gates. However, despite being targeted for lightweight applications, the PRESENT cipher algorithm results in a throughput of 200 kbps at 100 kHz. The implementation of Bogdanov et al. [10] PRESENT SP-box algorithm required more resources and less throughput.

Rolfes et al. [20] implemented the PRESENT cipher algorithm on three different architectures: round-based, parallel, and serial. The round-based architecture processes 80-bit key and 64-bit input messages through the multiplexer for the key schedule and data path unit or architecture. It requires 61 shift registers, 1 S-box, a 5-bit counter, and 2 XOR. The implementation requires 1561 gates with a throughput of 20.6 Mbps at 10 MHz using Mentor Graphics software. The parallel architecture is chosen for its higher throughput with tolerable increased hardware utilization. The total count of S-box, XOR, and P-box is 496, 32, and 31, respectively, which leads to higher-end hardware utilization. The number of gates used gets reduced with the trade-off in latency of 563 clock cycles. The blocks are reused to minimize the S-box and P-box count to one. It utilizes the 64 counts of XOR gates. The serial architecture (SA) uses less than 1000 gates. The efficient implementation of the resource is achieved by sharing the S-box between the data path and the enhanced key schedule architecture. Rolfes et al. [20] proposed architectures depending on the application requirement, which can be deployed. Rolfes et al. [20] architectures were devised during the time frame of the application and provide low throughput, which can be improved upon using the alternate methodology.

Sbeit et al. [21] proposed a Boolean optimization of S-box for the PRESENT cipher algorithm and implementation done with Spartan 3 FPGA. It utilizes 253 LUT and 152 flip-flops with a delay of 32 clock cycles. The cipher text was obtained at the throughput of 516 Mbps for 258 MHz. The main disadvantage of Sbeit et al. [21] architecture is increased hardware requirements that sometimes violate the lightweight scenario.

Yalla et al. [22] proposed a lightweight PRESENT cipher architecture with a 128-bit key as an input. The implementation was done with Spartan 3 FPGA. The proposed lightweight cipher architecture [22] reduced the data path for the S-box and utilized a shift register for P-box implementation. However, the entire block needs 256 clock cycles and 117 slices to compute the PRESENT cipher, which results in a delay of 8.78 ns and a throughput of 0.24 Mbps at 129.77 MHz. Yalla et al. [22] targeted lightweight implementation; however, it has low throughput and high latency.

Kavan et al. [23] implemented two S-box designs (slice and RAM) on Xilinx Spartan XC3S50 FPGA. The slice model PRESENT cipher algorithm S-box design utilizes 83 logic slices and has a latency of 1062 clock cycles, whereas the RAM model uses 85 slices with 1248 clock cycles latency. The throughput for the S-box's slice-based design was 6.03 kbps at 100 kHz, in contrast to the throughput of 5.13 kbps at 250.89 MHz for the RAM-based design. In general, the slice S-box is faster than RAM S-box. However, the architecture proposed by Kavan et al. [23] was stuck by long latency.

Hanley et al. [24] compared iterative architecture (IA) and serial architecture (SA) of the PRESENT cipher with Virtex-5 C5VLX50 FPGA. Both IA and SA approaches have an 8-bit data path that uses an almost equal number of LUTs, showing greater variation in latency with a slightly increased number of LUTs in SA architecture. The IA approach reduces latency compared to SA (47: 295 cycles). The IA and SA utilize 285 and 237 LUTs, respectively. The throughput of IA is 341.64 Mbps at 250.89 MHz, whereas the throughput of SA is 53.32 Mbps at 245.76 MHz.

Tay et al. [25] proposed Karnaugh mapping (K-map) S-box implementation (model 1) for the PRESENT cipher algorithm on FPGA. It utilizes two counters to implement the control logic. The K-map S-box implementation utilized 62 slices on the Virtex5 XC5VLX0 device and can achieve up to 51.32 Mbps throughput at 236.574 MHz. The factorization method (model 2) is also implemented to reduce memory utilization. The resource utilized is 201 flip-flop and 222 LUTs. The latency is 295 cycles and the throughput is 51.32 Mbps for 236.574 MHz [25]. The drawback of Tay et al. [25] model 2 is the long latency in the data path architecture and the large amount of combinational logic required in model 1.

Lara Nino et al. [26] proposed an architecture to optimize the PRESENT cipher algorithm by reducing the number of S-box and the flip-flop utilization. P-box is shrunk by decomposition technique up to 16-bit. The hardware utilized for S-box is four LUT. The reduction S-box method required 145 flip-flops and 524 LUT, 250 clock cycles, and can achieve a 361.16 Mbps throughput at 141.26 MHz [27]. The data path reduction method required 98 flip-flops and 478 LUTs and achieved 64.09 Mbps throughput at 132.19 MHz frequency.

Based on prior works on PRESENT cipher implementation on FPGA [10,21–27], it is evident that it either requires substantial hardware resources with very high throughput or a resource-lean architecture with long latency and low throughput. Lara Nino et al. [27] proposed a lightweight architecture for the PRESENT cipher algorithm with a 16-bit data path and key generation carried out with parallel processing. To overcome the problem mentioned earlier, due to the reduction of the data path, hardware resource utilization has decreased while the throughput remains high. The Lara Nino et al. [27] PRESENT cipher implementation utilizes 160 LUT and achieves 692.846 Mbps throughput with the frequency of 776 MHz on XC7Z030FBG676-2 ZYNQ FPGA [29]. Still, this architecture results in high latency, four clock cycles needed to process the input for one round. To overcome this, the low latency 32-bit data architecture design is proposed in this paper. The proposed architecture maintains a low latency with an overall increase in throughput.

3. Overview of Symmetric-Key Block Cipher-PRESENT Cipher SP-Box Algorithm

The PRESENT cipher algorithm is a combination of SP-box structure. The functioning of the 64-bit design can be summarized in four steps, add-round key (XOR), 2 × 16 substitution box (S-box), 1 × 64 Permutation box, and the key register updating for each round. The input 64-bit message is provided to the data processing engine as 64-bit in one clock cycle. The data processing engine (mux) has one path which consists of a multiplexer and a data pipeline register. The multiplexer is used to select the data between input data and round data. The 80-bit key is generated for 31 rounds, including the initial round, and is stored in the key register. In each round, 64-bit is available. The 64-bit output from each round is XORed with the key data already generated. The actual key length is 80-bit, excluding the bits from 0 to 15. Therefore, a 64-bit input message is easily XORed with a 64-bit key is shown in Figure 1. The fourth process, key updating, was carried out.

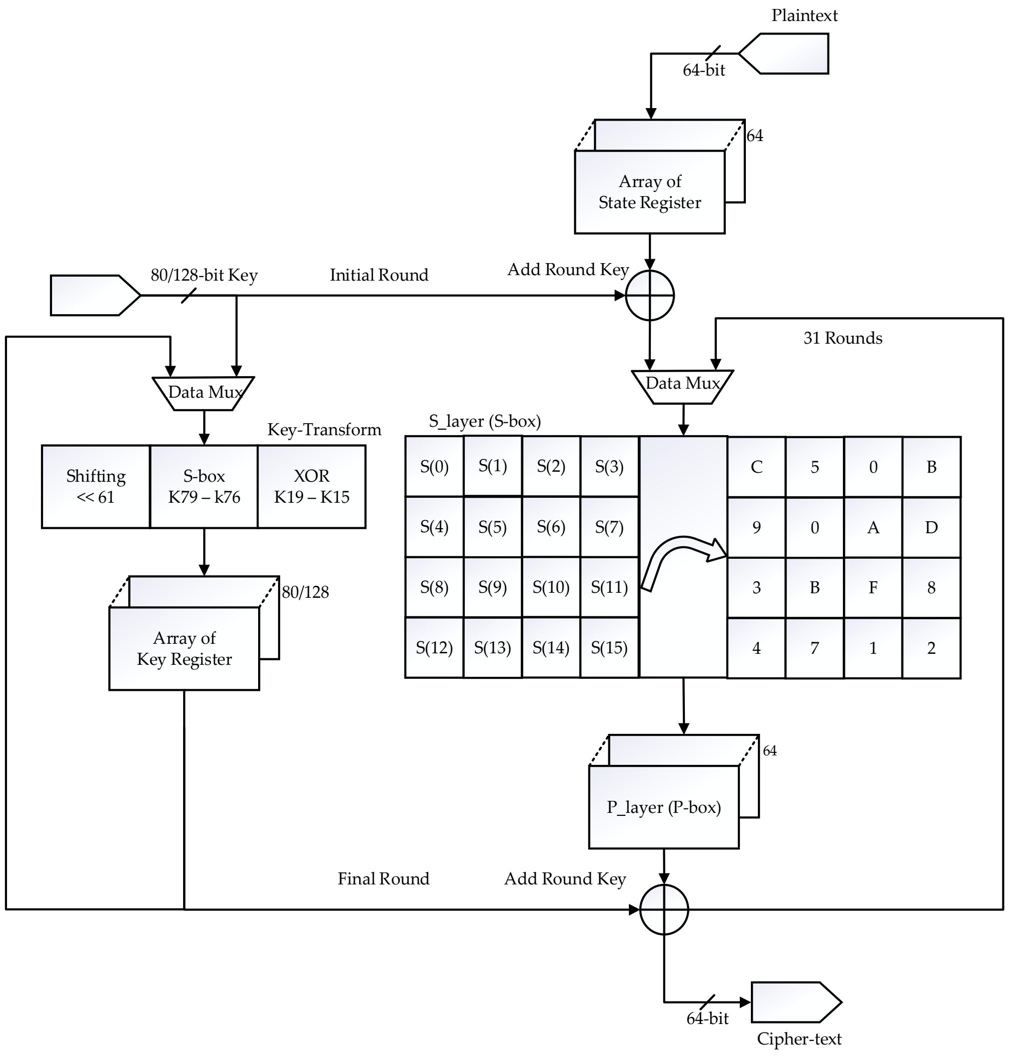


Figure 1. Top view of PRESENT Cipher algorithm.

The key updating involves the following three steps: shifting, S-box substitution, and XOR operation. The 1 × 1 array of keys, bit position 62 to 80, is shifted to the first position, and then bit position 1 to 61 is moved to the last position. In addition, the first four bits referring to the S-box (S1) are replaced according to the table referring to hex decimal numbers [10]. The bits 62 to 65 are XORed with the counter value based on the current iteration number. Two

S-box is used in data path architecture. The 64-bit output of the S-box is carried out to P-box. Where re-indexing, the bit position from the 0th bit to the 63rd bit is processed [10]. In the initial round, the bit 0 output of the LSB S-box is permutated to bit 0. In contrast, bit 3 output is permutated to bit 48 [10]. However, in the second round, bit 0 results are permutated to become bit 1, while bit 3 output is permutated to become bit 49.

In the last (fourth) round, bits 0 and 3 results are permutated to bits 3 and 51, respectively [10]. The output of the P-box is given as a loop back to the multiplexer for the next round process. After the first round, cipher text 1 is given as the input for the second round. Similarly, all four steps are repeated for 31 rounds.

4. Existing 16-Bit Data Path and Key Schedule Architecture

In Lara Nino et al.'s [27] work, the 64-bit plain text (input message) is provided to the data processing engine as 16-bit in four clock cycles. The data processing engine has four paths and each path consists of multiplexers and pipelined data registers, as shown in Figure 2. The multiplexer selects from either the plain text input or round data at each cycle. Each pipelined data register has four paths and consists of 4-bit registers. The 16-bit becomes available in each cycle. In the Add Round Key step, the 16-bit output from each round is XOR with the key data obtained from the key schedule engine shown in step 1. The result is then provided to a one-dimensional S-box (S-layer) given in step 2. Each 4-bit represented in S1(first hex data), S2, S3, S4 and so on. The S-box is a 4-bit substitution box, and such boxes produce 16-bit data for each round. Next, the output of the S-box is forwarded to the P-box (Player). Each P-box processes 16-bit data. Finally, the output of the P-box is looped back to the multiplexer for the next iteration. The bit position is replaced with the 16-bit position as shown in step 3.

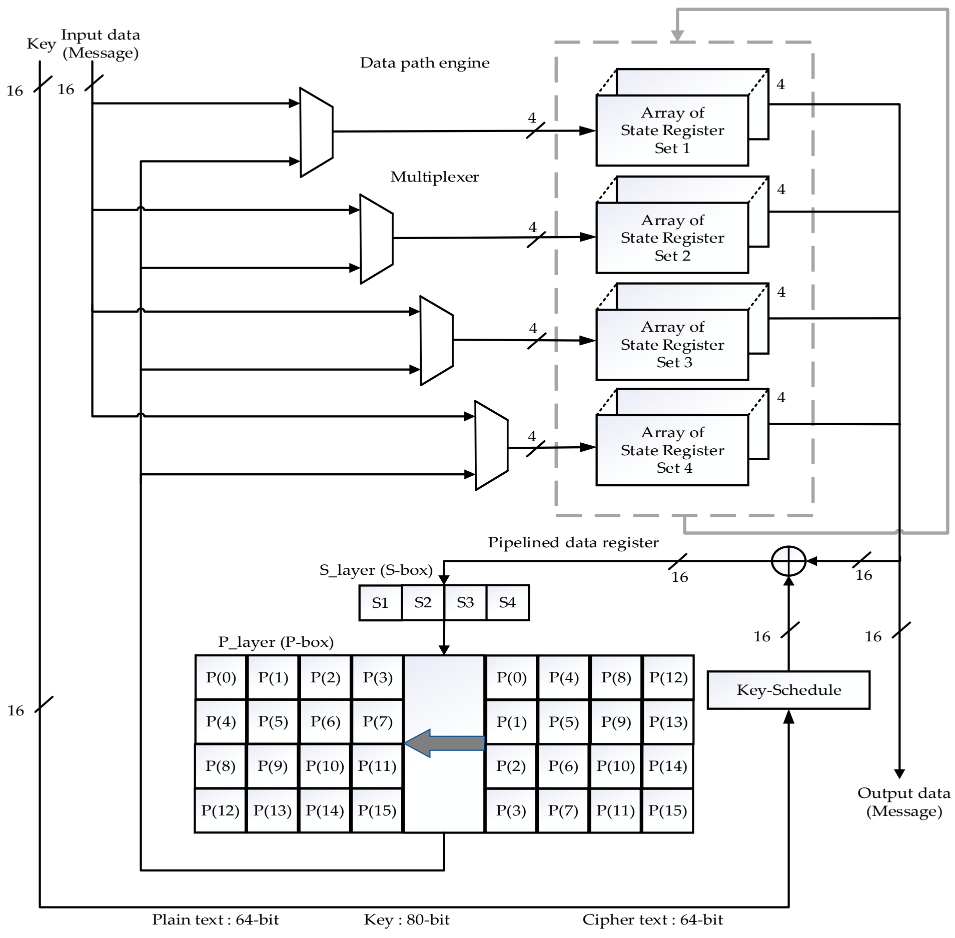


Figure 2. Sixteen-bit architecture for the data path.

Step 1: Add Round Key

$$a_j \to a_j \oplus {k_j}^i$$

a_j is the current state (input plain text message)
k_j is the round key
i is the number of rounds
a_j is directly assigned to S(in)

Step 2: S-layer (S-box)

The output from Add Round Key is referring to the S-box.
It is a non-linear substitution. The input and output specified in hexadecimal.
S(in) $\to$ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
S(out) $\to$ {c, 5, 6, B, 9, 0, A, D, 3, E, F, 8, 4, 7, 1, 2}
S(in) is the output from a_j (Add Round Key) and input to S-box
S(in) referring the S-box to generate the S(out) as follows

S(0) ⇨ C, S(1) ⇨ 5, S(2) ⇨ 6, S(3) ⇨ B,
S(4) ⇨ 9, S(5) ⇨ 0, S(6) ⇨ A, S(7) ⇨ D,
S(8) ⇨ 3, S(9) ⇨ E, S(A) ⇨ F, S(B) ⇨ 8,
S(C) ⇨ 4, S(D) ⇨ 7, S(E) ⇨ 1, S(F) ⇨ 2

The number 0 nibble is replaced by C, similarly, the number 1 nibble is replaced by 5 and so on.

Every four bits are replaced with a 4-bit hex decimal number [10].

The S (out) is the output from S-box is directly assigned to P(j).

Step 3: P-layer (P-box)

P(j) $\to$ (j*4) mod 15 j = 0 to 14;
P(in) $\to$ {0,1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,14,15}
P(out) $\to$ {0, 4, 8,12,1, 5, 9,13, 2, 6,10,14, 3, 7, 11,15}
P(15) $\to$ P (15); no change (position directly assigned)
P(in) is referring the position of a number from 0 to 15
P(j) is derived from the formula based on modular arithmetic
P(out) is changing the position from 0 to 14 based on P(j), as follows:

P(0) ⇦ P(0), P (1) ⇦ P(4), P(2) ⇦ P(8), P(3) ⇦ P(12),
P(4) ⇦ P(1), P (5) ⇦ P(5), P(6) ⇦ P(9), P(7) ⇦ P(13),
P(8) ⇦ P(2), P (9) ⇦ P(6), P(10) ⇦ P(10), P (11) ⇦ P(14),
P(12) ⇦ P(3), P(13) ⇦ P(7), P(14) ⇦ P(11), P(15) ⇦ P(15).

The first position P(1) is replaced with the fourth position, and the second position P(2) is replaced with the eighth position. Similarly, 0 to 14 positions are replaced based on the permutation formula. However, the zeroth and fifteenth position remains the same. The above three steps are referred from [10].

The key schedule engine's entire 80-bit is processed in four cycles. In every cycle, 16 bits are available as round keys. Five 4-bit registers were arranged in this particular structure to form the key register. The key scheduling engine has four paths, each with one 2×1 multiplexer and pipelined key register. The key schedule architecture is similar to the data path architecture, where multiplexers switch the data between the round input keys in each cycle. The 80-bit output from key registers undergoes three processes: shifting, XOR with current counter value, followed by S-box and P-box shown in Figure 3. The

64-bit cipher text (output message) is obtained at the end of the process. The throughput is calculated as referred to in [27]. It is expressed in Equation (1).

$$\text{Throughput} = \frac{\text{Maximum frequency} \times \text{Block size}}{\text{Latency(cycles)}} \tag{1}$$

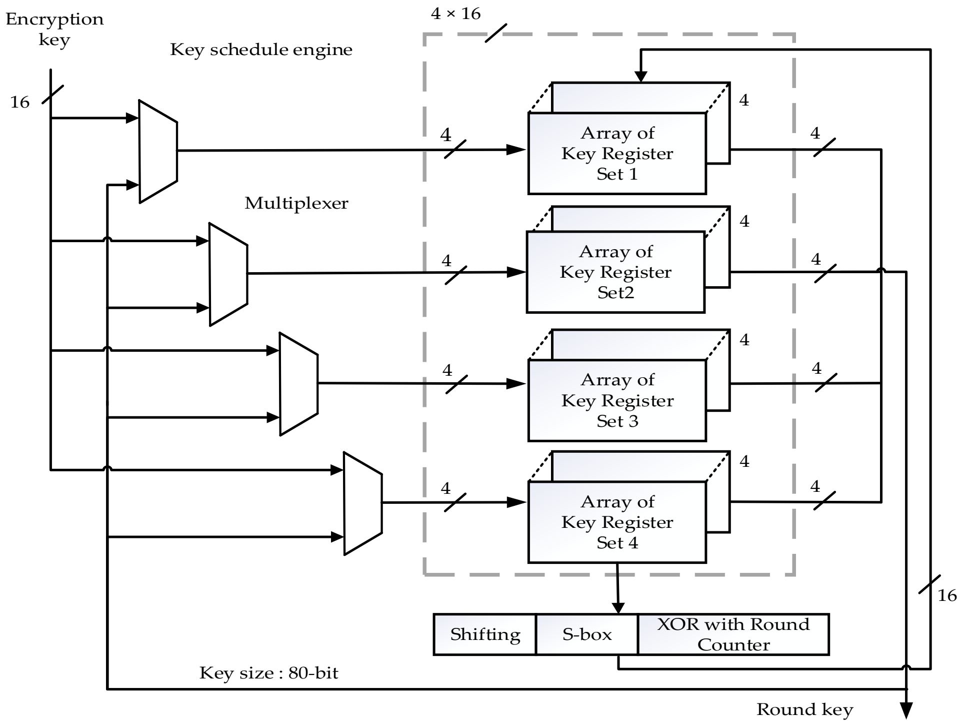


Figure 3. Sixteen-bit architecture for the key schedule.

The implementation was carried out on ZYNQ board 7000 for the devices XC7Z030FBG676-2 [30]. The available resource of LUT, flip-flop, and I/O are 78,600, 157,200, and 250, respectively. In Lara Nino et al. [27], the utilization of LUT for 16-bit architecture is only 0.20%. The flip-flop utilization is 0.10%. The I/O utilization is 21.20%. The total power is 0.157 W, dynamic power is 0.034 W, and static power is 0.123 W. The complete architecture is implemented in 133 clock cycles, which leads to a maximum latency of 171.392 ns. The minimum clock period is 1.29 ns. The maximum operating frequency results are validated through the optimized hardware tool Minerva [31]. In the application of RFID, the lowest frequency chosen for very low resource utilization is 13.56 MHz. The throughput* is calculated for low-frequency RFID, as in [27,30]. It is given by Equation (2).

$$\text{Throughput}{*}= \frac{13.56\ \text{MHZ} \times \text{Block size}}{\text{Latency(cycles)}} \tag{2}$$

5. Proposed Low Latency 32-Bit Data Path and Key Schedule Architecture

This paper proposed an improvement from Lara Nino et al. [27] work with a 32-bit data architecture for the PRESENT cipher algorithm. The 64-bit input is provided to the data engine as 32-bit in each clock cycle. It requires two clock cycles to get the input. The data engine has four paths, in a multiplexer and in a data pipeline register. The multiplexer is used to select the data between input data and round data. The first option in the multiplexer selects data from the primary input, while the second option selects the round data. The data pipeline register consists of four registers, each size of 8-bit, shown in Figure 4. Hence four paths with two registers compute 32-bit data. In each cycle, a 32-bit output becomes available for the following process.

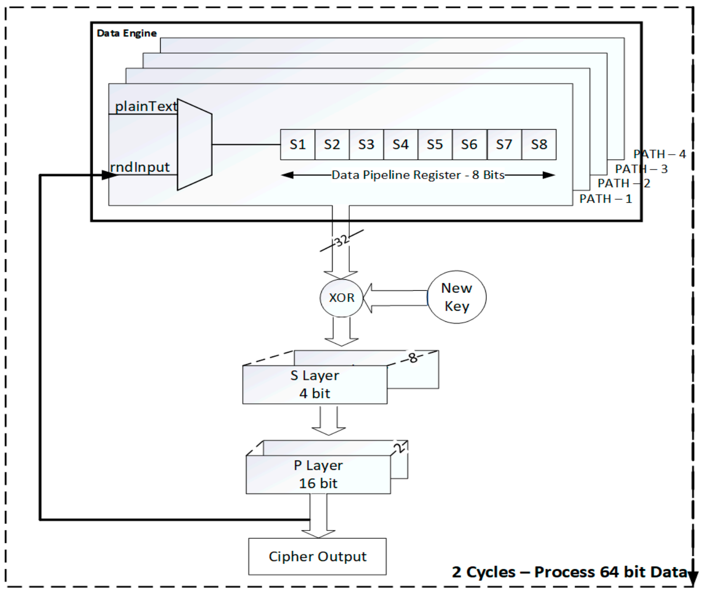


Figure 4. Proposed Architecture for the 32-bit data path unit.

The 32-bit from the data pipeline register is XOR with the key data. The key data is obtained from the proposed key engine (key schedule unit). The output is then provided to S-box is shown in Figure 4. Eight S-boxes are utilized to process the 32-bit data. Inside the S-box, data replacement for every four bit data is obtained after the XOR operation. The output of the S-box is then provided to the P-box. In this, 32-data is repositioned with the 32-bit position or 32-bit data is repositioned with two 16-bit positions. The output of the P-box is looped back to the multiplexer for the next processing round.

The key engine has four paths, in a multiplexer and in a key pipelined register. The 80-bit input key is provided in three clock cycles as 32-bit per clock cycle. The first option in the multiplexer selects the main input key. The second option selects the round data. The key engine is similar to the data engine shown in Figure 5. Three registers of 8-bit size are available in the key pipelined register. The last 16-bit data is excluded during the process. The proposed 32-bit architecture is implemented in the same XC7Z030FBG676-2 on the ZYNQ 7000 [30].

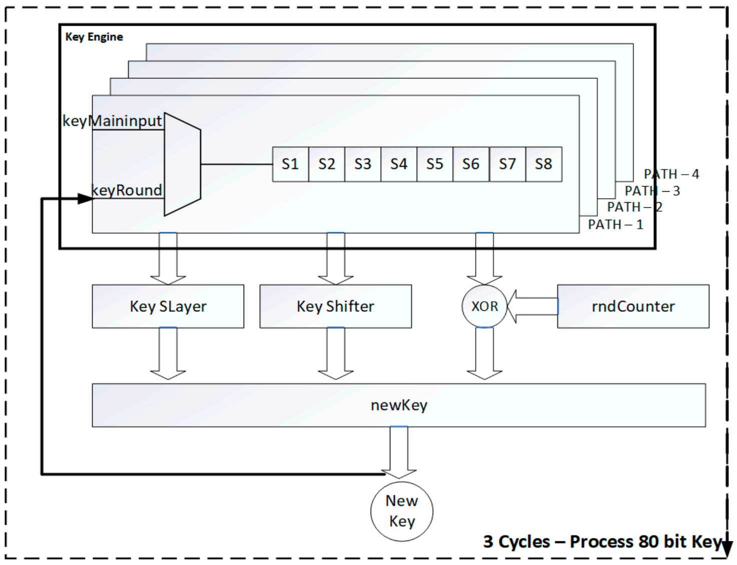


Figure 5. Proposed architecture for the 32-bit key schedule.

The LUT utilization is 185/8600, which is 0.24%. The flip-flop is 169/57200, which is 0.11%. The IO utilization is 40.40%, which is 101/250 availability. The dynamic power utilization is 0.134 W and the static power is 2.62 W. The total on-chip power is 2.754 W. The latency is 92.46 ns with 65 clock cycles. The minimum clock period is 1.49 ns. The Minerva hardware optimization tool is used to validate the results in the proposed work [31].

6. Result and Discussion

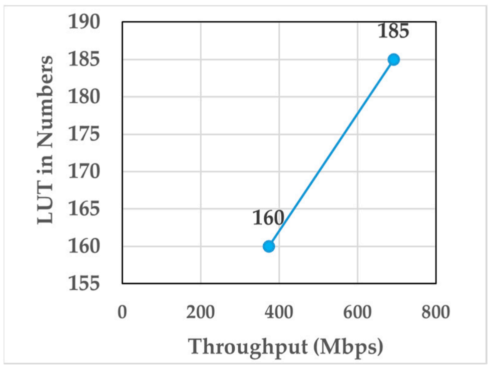
Table 1 compares Lara Nino et al.'s [27] 16-bit data path PRESENT cipher algorithm architecture versus the proposed low latency 32-bit data path architecture. Both implementations are done on the ZYNQ board 7000 [30]. The throughput of the proposed low latency 32-bit architecture increases by 85.54% compared to the 16-bit architecture. The 16-bit architecture requires five S-boxes, two P-boxes, and 133 clock cycles to complete the 31-round process. The proposed architecture requires two P-box, nine S-boxes, and 65 clock cycles to complete the 31-round process. In particular, two cycles are used to process the 32-bit input data in each round for the proposed architecture, resulting in a low latency, which makes the computing speed twice that of Lara Nino et al.'s [27] architecture. The proposed 32-bit architecture increase in throughput would allow IoT with time-constraint applications to perform better. The throughput* is calculated for the frequency of 13.56 MHz. The trade-off between the speed, latency, and resource utilization with the proposed low latency 32-bit architecture versus the Lara Nino et al. [27] architecture is shown in Figure 6. The output and test vector are depicted in Tables 1 and 2. Power utilization is the limitation of the proposed architecture. The main focus of the proposed architecture is increasing the throughput by reducing the latency. Therefore, resource and power utilization are getting compromised and the same optimized architecture can be extended for the power analysis in future work [32,33].

Table 1. Comparison of Resource Utilization.

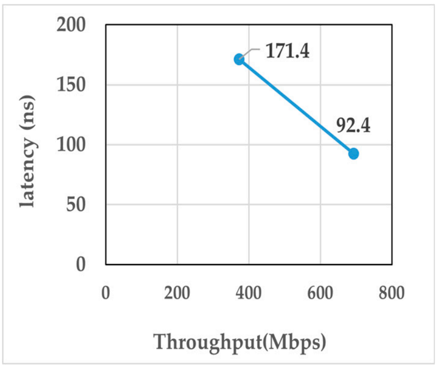
Parameter	Existing Architecture [27]	Proposed Architecture
Block size	64	64
Key size	80	80
Number of slices	48	59
Number of LUT	160	185
Number of flip-flops	153	169
Latency (ns)	171.392	92.46
Latency (Cycles)	133	65
Max. frequency (MHz)	776	703
Throughput (Mbps)	373.413	692.846
Throughput/Slice (kbps)	7.779	11.743
Throughput* (kbps)	6.578	13.459
Total power (W)	0.157	2.754
Static power (W)	0.123	2.62
Dynamic power (W)	0.034	0.134

Table 2. Test vectors.

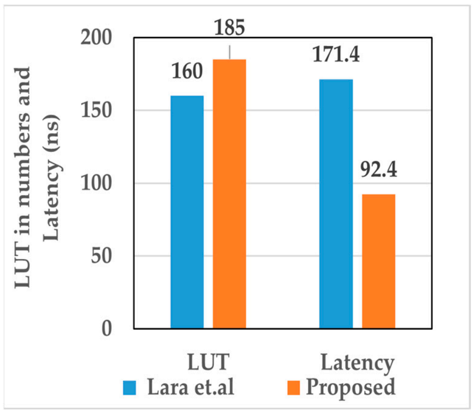
Plain Text Message (Input Data) (64-Bit)	Key Data (80-Bit)	Cipher Text Message (Output Data) (64-Bit)
00000001F708E9B8	0000000008FB8F50f7E0	661B90DFD32CB83C
00001DE63A028FEB	00000000000291056CF3	EBA17AB44B0CA503
018EB8895EED0E10	D005A30380E380000000	F269C4A6405880B3
7CB547399FFD1400	95100D1BF3D0C8000000	5102C10A4646A2A0



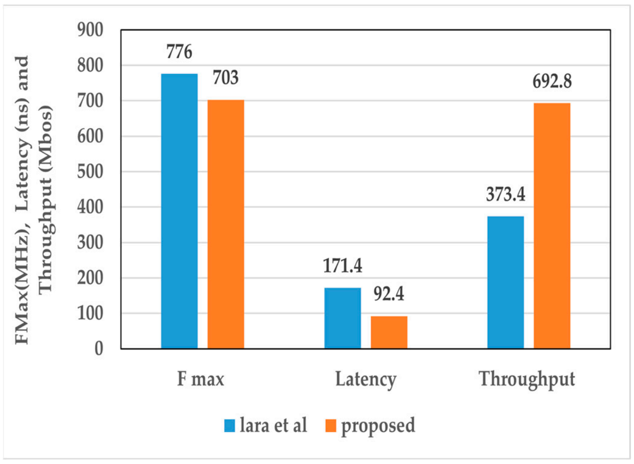
(**a**)



(**b**)



(**c**)



(**d**)

Figure 6. (**a**) The trade-off between throughput and the number of LUT; (**b**) the trade-off between throughput and latency; (**c**) architecture comparison with LUT and latency; (**d**) architecture comparison with throughput, latency, and maximum frequency.

The set of 64-bit plain text messages is listed in column 1 of Table 2, 80-bit key data is represented in column 2 and the corresponding 64-bit cipher text (output data) generated using the Vivado simulator is shown in column 3. The samples are simulated both in MATLAB and XILINX VIVADO. The sample 1 data of plain text message "00000001F708E9B8" is encrypted using the PRESENT cipher algorithm. The encrypted 64-bit cipher text message is "661B90DFD32CB83C". The 80-bit key "0000000008FB8F50f7E0" is used to encrypt the plain text message. Similarly, three more samples are displayed in Table 2.

7. Conclusions

PRESENT cipher algorithm is suitable for high throughput lightweight IoT applications. This paper proposed a low latency 32-bit data path PRESENT cipher architecture on XILINX XC7Z030FBG676-2 ZYNQ board 7000 FPGA. The proposed design improved the overall throughput of Lara et al. [27] architecture by 85.54%, with only an increase in hardware. Overall, the proposed low latency 32-bit data path architecture can compute the PRESENT cipher algorithm more efficiently than the existing architectures. The scientific contribution of the proposed architecture is flexible FPGA architecture; the user can choose to increase the throughput and decrease the latency for PRESENT cipher in the field of cryptography. Future enhancements in the PRESENT algorithm can see the implementation of the reconfigurable architecture, which improves hardware optimization and throughput performance. The proposed architecture may further be investigated for low power analysis, fault tolerance, and Microblaze soft core integration with ZYNQ board as future work.

Author Contributions: J.D.: Conceptualization, methodology, software, validation, formal analysis, Investigation, writing—original draft preparation; E.R.S.M.: validation, writing—review and editing, visualization, supervision; N.S.-H.: validation, writing—review and editing, visualization, supervision. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ullah, I.; Mahmoud, Q.H. Design and Development of Deep Learning-Based Model for Anomaly Detection in IoT Networks. *IEEE Access* **2021**, *9*, 103906–103926. [CrossRef]
2. Arul Murugan, C.; Karthigai Kumar, P.; Sathya Priya, S. FPGA implementation of hardware architecture with AES encryptor using sub-pipelined S-box techniques for compact applications. *Automatika* **2020**, *61*, 682–693. [CrossRef]
3. Marzouqi, H.; Al-Qutayri, M.; Salah, K.; Schinianakis, D.; Stouraitis, T. A high-speed FPGA implementation of an RSD-based ECC processor. *IEEE Trans. Very-Large-Scale Integr. (VLSI) Syst.* **2015**, *24*, 151–164. [CrossRef]
4. Bani-Hani, R.; Harb, S.; Mhaidat, K.; Taqieddin, E. High-throughput and area-efficient FPGA implementations of data encryption standard (DES). *Circuits Syst.* **2014**, *5*, 45–46. [CrossRef]
5. Ahmad, R.; Kho, D.; Ismail, W. Parallel-Pipelined-Memory-Based Blowfish Design with Reduced FPGA Utilization for Secure Zig Bee Real-Time Transmission. *Wirel. Pers. Commun.* **2019**, *104*, 471–489. [CrossRef]
6. Tarus, M.S.; McKay, K.A.; Calik, C.; Chang, D.; Bassham, L. *Status Report on the First Round of the NIST Lightweight Cryptography Standardization Process*; National Institute of Standards and Technology, NIST Interagency/Internal Rep. (NISTIR): Gaithersburg, MD, USA, 2019.
7. Thakor, V.A.; Razzaque, M.A.; Khandaker, M.R.A. Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison, and research opportunities. *IEEE Access* **2021**, *9*, 28177–28193. [CrossRef]
8. Tao, H.; Bhuiyan, M.Z.A.; Abdullah, A.N.; Hassan, M.M.; Zain, J.M.; Hayajneh, T. Secured Data Collection with Hardware-Based Ciphers for IoT-Based Healthcare. *IEEE Internet Things J.* **2019**, *6*, 410–420. [CrossRef]
9. Jamuna Rani, D.; Emalda Roslin, S. Lightweight cryptographic algorithms for medical internet of things (IoT)-a review. In Proceedings of the Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, India, 19 November 2016.
10. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.; Seurin, Y.; Vikkelsoe, C. RESENT: An ultra-lightweight block cipher. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Vienna, Austria, 10–13 September 2007; Springer: Berlin/Heidelberg, Germany, 2007; Volume LNCS 4127, pp. 450–466.
11. Abbas, Y.A.; Jidin, R.; Jamil, N.; Zaba, M.R.; Rusli, M.E.; Tariq, B. Implementation of PRINCE Algorithm in FPGA. In Proceedings of the International Conference on Information Technology and Multimedia (ICIMU), Putrajaya, Malaysia, 18–20 November 2014.
12. Kolbl, S.; Leander, G.; Tiessen, T. Observations on the SIMON block cipher family. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 16–18 August 2015; Springer: Berlin/Heidelberg, Germany, 2015; Volume LNCS 9215, pp. 161–185.
13. Banik, S.; Pandey, S.K.; Peyrin, T.; Sasaki, Y.; Sim, S.M.; Todo, Y. GIFT: A small present. In Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, Taipei, Taiwan, 25–28 September 2017; Springer: Cham, Switzerland, 2017; pp. 321–345.
14. Jamuna Rani, D.; Emalda Roslin, S. Optimized Implementation of Gift Cipher. *Wirel. Pers. Commun.* **2021**, *119*, 2185–2195. [CrossRef]
15. Beierle, C.; Jean, J.; Stefan, K.; Leander, G.; Moradi, A.; Peyrin, T.; Sasaki, Y.; Sasdrich, P.; Sim, S.M. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 123–153.
16. Nallathambi, B.; Palanivel, K. Fault diagnosis architecture for SKINNY family of block ciphers. *Microprocess. Microsyst.* **2020**, *77*, 103202. [CrossRef]
17. Guo, J.; Peyrin, T.; Poschmann, A. The PHOTON family of lightweight hash functions. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 222–239.
18. Suzaki, T.; Minematsu, K.; Morioa, S.; Kobayashi, E. A Lightweight Block Cipher for Multiple Platforms. In Proceedings of the International Conference on Selected Areas in Cryptography, Windsor, ON, Canada, 15–16 August 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 339–354.
19. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK lightweight block ciphers. In Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 8–12 June 2015; pp. 1–6.
20. Rolfes, C.; Poschmann, A.; Leander, G.; Paar, C. Ultra-lightweight implementations for smart devices—Security for 1000 gate equivalents. In Proceedings of the International Conference on Smart Card Research and Advanced Applications, London, UK, 8–11 September 2008; Volume LNCS 5189, pp. 89–103.

21. Sbeiti, M.; Silbermann, M.; Poschmann, A.; Paar, C. Design space exploration of PRESENT implementations for FPGAs. In Proceedings of the 2009 5th Southern Conference on Programmable Logic (SPL), IEEE, Sao Carlos, Brazil, 1–3 April 2009; pp. 141–145.
22. Yalla, P.; Kaps, J.P. Lightweight cryptography for FPGA. In Proceedings of the International Conference on Reconfigurable Computing and FPGAs, IEEE, Cancun, Mexico, 9–11 December 2009; pp. 225–230.
23. Kavun, E.B.; Yalcin, T. RAM-based ultra-lightweight FPGA implementation of PRESENT. In Proceedings of the International Conference on Reconfigurable Computing and FPGAs, IEEE, Cancun, Mexico, 30 November–2 December 2011; pp. 280–285.
24. Hanley, N.; Neill, O.M. RAM-based ultra-lightweight FPGA implementation of PRESENT. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI, Amherst, MA, USA, 19–21 August 2012; pp. 57–62.
25. Tay, J.J.; Wong, M.D.; Wong, M.M.; Zhang, C.; Hijazin, I. Compact FPGA Implementation of PRESENT with Boolean S-Box. In Proceedings of the 6th Asia Symposium on Quality Electronic Design (ASQED), IEEE, Kula Lumpur, Malaysia, 4–5 August 2015; pp. 144–148.
26. Lara-Nino, C.A.; Morales-Sandoval, M.; Diaz-Perez, A. Novel FPGA-based low-cost hardware architecture for the PRESENT block cipher. In Proceedings of the Euro micro-Conference on Digital System Design (DSD), Limassol, Cyprus, 31 August–2 September 2016.
27. Lara-Nino, C.A.; Diaz-Perez, A.; Morales-Sandoval, M. Lightweight hardware architectures for the present cipher in FPGA. *Trans. Circuits Syst.* **2017**, *64*, 2544–2555. [CrossRef]
28. Moosavi, S.R.; Rahmani, A.M.; Westerlund, T.; Yang, G.; Liljeberg, P.; Hannu, T. Pervasive health monitoring based on internet of things: Two case studies. In Proceedings of the 4th International Conference on Wireless Mobile Communication and Healthcare-Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH), IEEE, Athens, Greece, 3–5 November 2014; pp. 275–278.
29. Xilinx, X. Zynq-7000 All Programmable SoC Overview, DS190. *Prod. Specif.* **2018**, 1–25.
30. Finkenzeller, K. *Identification Cards—Contactless Integrated Circuit Cards—Proximity Cards—Part 2: Radio Frequency Power and Signal Interface*, Document ISO/IEC 14, 3rd ed.; John Wiley and Sons: German, UK, 2010; pp. 443–452.
31. Farahmand, F.; Ferozpuri, A.; Diehl, W.; Gaj, K. Minerva: Automated hardware optimization tool. In Proceedings of the International Conference on Reconfigurable Computing and FPGAs (Re ConFig), Cancun, Mexico, 4–6 December 2017; pp. 1–8.
32. Pandey, J.G.; Goel, T.; Karmakar, A. A high-performance and area-efficient VLSI architecture for the PRESENT lightweight cipher. In Proceedings of the 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), IEEE, Pune, India, 6–10 January 2018; pp. 392–397.
33. Maro, E. Modelling of power consumption for Advanced Encryption Standard and PRESENT ciphers. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1155*, 012060. [CrossRef]