# Efficient Implementations of CRAFT Cipher For Internet of Things

Jiahao Xiang[a,b], Lang Li[a,b,*]

[a]College of Computer Science and Technology, Hengyang Normal University, Hengyang, 421002, , China
[b]Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang Normal University, Hengyang, 421002, , China

**Abstract**

The advent of the Internet of Things (IoT) has revolutionized device and data interactions, enabling unprecedented connectivity and automation. However, achieving specific targets of low area usage and high throughput on these devices presents a significant challenge. This paper introduces two innovative designs for the CRAFT lightweight block cipher, each targeting a different aspect: low area usage and high throughput. The design for low area usage employs a serial architecture, reducing the datapath from 64-bit to 4-bit, which significantly decreases the area. Additionally, Boolean satisfiability (SAT) solvers are utilized to identify a lower-cost area implementation of the cipher's confusion component. Conversely, the design for high throughput leverages an unrolled architecture, minimizing latency from 32 to 16 and enhancing throughput. The proposed designs were evaluated on three distinct FPGA platforms: Artix-7, Kintex-7, and Spartan-7. The results indicate that the low area design reduces area usage by 15.82% compared to the PRESENT cipher. Conversely, the unrolled design doubles the throughput rate at 100MHz. Moreover, the unrolled design significantly reduces energy consumption per bit by 47.89% compared to the PRESENT cipher. These findings suggest that the introduced designs have the potential to enhance both the security and efficiency of IoT devices.

*Keywords:* Internet of Things, Lightweight block cipher, Field-programmable gate array(FPGA), Low-area, High-throughput

## 1. Introduction

The Internet of Things (IoT) is making rapid strides into various sectors such as personal health care, environmental monitoring, home automation, smart mobility, and Industry 4.0. As security concerns continue to surface, they are thoroughly discussed in [1]. The most effective way to ensure data protection is by employing the cryptography techniques outlined in [2].

However, the resource constraints of many IoT devices pose challenges for the implementation of robust security measures. These devices often have limited memory, processing power, and energy. Therefore, the security measures need to be lightweight to ensure they do not overburden the resources. Lightweight cryptography, a subset of cryptography, provides solutions specifically designed for these resource-limited devices, as discussed in [3].

Lightweight cryptography has been extensively studied in recent years. Examples of this include PRESENT [4], LED [5], Midori [6], QTL [7], GIFT [8], CRAFT [9], Shadow [10], DULBC [11], IVLBC [12], BipBip [13], and LELBC [14]. More ciphers can be found in [15]. The implementation of lightweight ciphers for various applications has also been widely studied.

Efficient implementation allows lightweight ciphers to be used in various settings. A hardware implementation can enhance the performance of these ciphers in resource-limited environments. For instance, a 16-bit datapath architecture for the PRESENT cipher, which reduces both the area and power consumption, is presented in the work of Lara-Nino et al. [16]. Additionally, an optimized key schedule of PRESENT, resulting in a smaller area, is proposed by Pandey et al. [17]. Shahbazi et al. [18] propose an 8-bit serial architecture for AES, which also reduces the area and power consumption. Li et al. [19] present unrolled architectures and a low-cost architecture for PRINCE, separately optimizing the throughput and area. Bharathi et al. [20] enhance the performance of the PRESENT cipher by expanding the key length. Finally, Yang et al. [21] share components in the cipher process for LILLIPUT, resulting in a smaller area.

This work presents the first implementation of CRAFT on FPGA platforms. Two architectures for CRAFT, Serial and Unrolled, are proposed. The Serial architecture reduces the datapath from 64-bit to 4-bit, meaning it only uses one S-Box, which significantly reduces the area usage. The Unrolled architecture reduces the latency of the encryption process, thereby improving the throughput rate. A SAT solver is used to find the optimal implementation of the S-Box, which further reduces the area of the S-Box. The experiments are conducted on three different FPGA platforms: Artix-7, Kintex-7, and Spartan-7. The contributions of this article can be summarized as follows.

- Two architectures for CRAFT, Serial and Unrolled, are proposed. These are optimized for area and throughput, respectively. The Serial architecture reduces the area usage by 15.72% compared to the work of Bharathi et al. [20]. The Unrolled architecture doubles the throughput rate compared to the same work.

*Corresponding author
*Email address:* lilang911@126.com (Lang Li)

- A SAT solver is used to find the optimal implementation of the S-Box, which further reduces the area of the S-Box. The proposed S-Box reduces the area by 28.9% compared to the work of Bao et al. [22].
- All architectures are implemented on three different FPGA platforms: Artix-7, Kintex-7, and Spartan-7. The results suggest that the proposed designs could potentially enhance the security and efficiency of IoT devices.

The remainder of this article is structured as follows: Section 2 provides the specifications of CRAFT. The two proposed architectures for CRAFT are discussed in Section 3. Section 4 outlines the experimental evaluation metrics and environment. Section 5 presents the results of all the architectures. Finally, Section 6 concludes the work.

## 2. Specification of CRAFT

The key notations used in this paper are outlined in Table 1. CRAFT, a lightweight tweakable block cipher, is built from involutory building blocks. It operates on a 64-bit block and utilizes a 128-bit key, along with a 64-bit tweak. In this cipher, a 64-bit input plaintext is transformed into a 64-bit output ciphertext using a 128-bit key . The key $K$ is split into $K_0$ and $K_1$. A 64-bit tweak is also used in the process. The structure of CRAFT is illustrated in Figure 1. The CRAFT encryption process is described in Algorithm 1. The decryption process is the same as the encryption process, except that the round keys are used in reverse order.

Table 1: Key Notations Used in This Paper

| Notation | Description |
|---|---|
| $TK_i$ | 64-bit tweakeys used in the $i^{th}$ round |
| $RC_i$ | 64-bit round constant for the $i^{th}$ round |
| $R_i$ | Function for the $i^{th}$ round |
| $SB$ | S-Box operation |
| $MC$ | Mix-Columns operation |
| $PN$ | PermuteNibbles operation |
| $Q$ | Permutation used in key schedule |
| $\oplus$ | XOR operation |
| $\|\|$ | Concatenation operation |

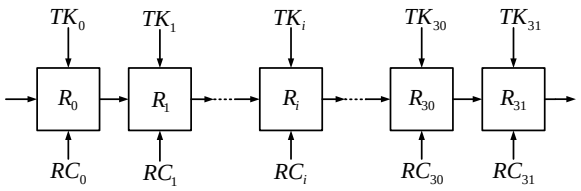


Figure 1: Structure of CRAFT

The round function, denoted as $R$, consists of three operations: Mix-Columns, PermuteNibbles, and S-Box. The Mix-Columns operation is a linear transformation that multiplies the input column by a constant matrix, $M$, to generate the output column. Notably, $M$ is an involutory matrix, as shown in Equation 1.

**Algorithm 1** CRAFT Encryption Process

**Input:** Plaintext $X$, Key $K_0||K_1$, Tweak $T$
**Output:** Ciphertext $Y$

$TK_0 \leftarrow K_0 \oplus T$
$TK_1 \leftarrow K_1 \oplus T$
$TK_2 \leftarrow K_0 \oplus Q(T)$
$TK_3 \leftarrow K_1 \oplus Q(T)$
$Y \leftarrow X$
**for** $i \leftarrow 0$ to $31$ **do**
    $Y \leftarrow MC(Y)$
    $Y_{4,5} \leftarrow Y_{4,5} \oplus RC_i$
    $Y \leftarrow Y \oplus TK_{i \bmod 4}$
    **if** $i \neq 31$ **then**
        $Y \leftarrow PN(Y)$
        $Y \leftarrow SB(Y)$
    **end if**
**end for**

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

The PermuteNibbles operation, an involutory permutation, operates on 4-bit nibbles. This operation triggers additional S-Boxes, thereby bolstering the cipher's security. The illustration of the PermuteNibbles operation is provided in Equation 2. The permutation $Q$ is utilized in the key schedule, as depicted in Equation 3.

$$PN = [15, 12, 13, 14, 10, 9, 8, 11, 6, 5, 4, 7, 1, 2, 3, 0] \quad (2)$$

$$Q = [12, 10, 15, 5, 14, 8, 9, 2, 11, 3, 7, 4, 6, 0, 1, 13] \quad (3)$$

The S-Box operation, a nonlinear transformation, introduces confusion into the cipher. This operation is carried out using a 4-bit S-Box, depicted in Table 2, with values represented in hexadecimal notation.

Table 2: S-Box of CRAFT

| Input | Output | Input | Output |
|---|---|---|---|
| 0x0 | 0xC | 0x8 | 0x8 |
| 0x1 | 0xA | 0x9 | 0x9 |
| 0x2 | 0xD | 0xA | 0x1 |
| 0x3 | 0x3 | 0xB | 0x5 |
| 0x4 | 0xE | 0xC | 0x0 |
| 0x5 | 0xB | 0xD | 0x2 |
| 0x6 | 0xF | 0xE | 0x4 |
| 0x7 | 0x7 | 0xF | 0x6 |

The round constants, denoted as $RC$, are defined as $RC = a_3||a_2||a_1||a_0||0||b_2||b_1||b_0$. The initial round constant, $RC_0$, is set to 0x11. The process for updating the round constants is depicted in Equations 4.

$$\begin{aligned}(a_3, a_2, a_1, a_0) &\leftarrow (a_1 \oplus a_0, a_3, a_2, a_1)\\ (b_2, b_1, b_0) &\leftarrow (b_1 \oplus b_0, b_2, b_1)\end{aligned} \tag{4}$$

## 3. Implementations

For the first time, the components of CRAFT have been optimized to achieve efficient area and throughput, resulting in two proposed implementation architectures: Serial and Unrolled.

### *3.1. Serial Architecture (A1)*

Compared to round-based architectures, serial architectures can significantly reduce area usage by reusing components. For instance, the number of S-Boxes is reduced from 16 to 1. The clock gating technique is also employed to enable each component and minimize the energy consumption of encryption. The proposed architecture is presented in Figure 2.

The design includes one S-Box, one 4-bit Mix-columns, and two register banks for storing keys (referred to as Key-Register) and plaintext (referred to as State-Register). These also act as temporary registers for storing intermediate results. To store intermediate results into the State-Register bank, the design includes one feedback path. The PermuteNibbles is incorporated into the State-Register bank.

It's worth noting that the execution of permute requires 64-bit. To reuse the State-Register block, the order of execution of S-Box and Permute is altered. Additionally, the first round of the encryption process avoids the Permute operation through the control signal, ensuring the correctness of the encryption algorithm.

#### *3.1.1. S-Box Optimization*

S-Box provides a confusing characteristic for the entire encryption algorithm, however it requires a large amount of area. There are different methods of implementation of S-Box. The most popular implementation is using a lookup table (LUT), such as [16]. However it uses a lot of flip-flop, which will bring a lot of area consumption. Using S-Box's equivalent logical expression for this will reduce area consumption, such as [22], [23].

Boolean satisfiability (SAT) solvers can be used to find S-Box that satisfy certain implement, such as being resistant to software or hardware implement. In more detail, the S-Box implement can be encoded as Boolean constraints by representing the S-Box as a truth table and then using Boolean variables to represent the input and output bits of the S-Box. The constraints can then be formulated based on the desired implement of the S-Box. Once the S-Box implement are encoded as Boolean constraints, a SAT solver can be used to find a satisfying assignment to these constraints, which corresponds to an S-Box that satisfies the desired implement.

The gate equivalent complexity(GEC) of a SAT instance is the number of logical gates required to implement the Boolean formula that represents the instance. GEC can be calculated by converting the Boolean formula into a circuit of logical gates, such as AND, OR, and NOT gates. The number of gates in the circuit corresponds to the GEC of the instance.

In our design, we optimize and use GEC encoding scheme of [23] to implement the S-Box. Our encoding scheme as follows in Equations 5:

$$\begin{aligned}&\forall i \in \{0, 1, \ldots, K-1\}:\\ &T_i = F_{if}(BB_i[0], \sim (Q_{4i} \cdot Q_{4i+1}) \cdot \sim Q_{4i+2} \cdot Q_{4i+3})\\ &\quad + F_{if}(BB_i[1], Q_{4i+2} \cdot (Q_{4i} + Q_{4i+1}))\\ &\quad + F_{if}(BB_i[2], Q_{4i} \cdot Q_{4i+1} \cdot Q_{4i+2})\\ &\quad + F_{if}(BB_i[3], Q_{4i+2}) + F_{if}(BB_i[4], Q_{4i})\\ &\quad + F_{if}(BB_i[5], Q_{4i} \cdot Q_{4i+1})\\ &\quad + F_{if}(BB_i[6], Q_{4i} + Q_{4i+1}) + F_{if}(BB_i[7], max)\end{aligned} \tag{5}$$

where $K$ is numbers of the logical gates, $Q_{4i} - Q_{4i+3}$ is the input of the $i^{th}$ logical gate, $T_i$ is the output of the $i^{th}$ logical gate, and $F_{if}$ is a function that returns the value of the second argument if the first argument is true and returns the value of zero otherwise. The value of $max$ is all one's in the binary expression, which is represented logically as an inverse. $BB_i$ represents the type of the $i^{th}$ logical gate, which is a 8-bit binary number. The different types of logical gate used in this encoding scheme are listed in Table 3.

The optimized scheme of S-Box is shown in Equations 6, where $X_0 - X_3$ is the input of the S-Box and $Y_0 - Y_3$ is the output of the S-Box. The proposed scheme of S-Box is implemented by four MOAI1 gates, three MAOI1 gates, and one AND3 gate. This module of the proposed S-Box reduced the area by 28.9% with [22] (based on gate equivalent estimation on UMC 180nm library).

$$\begin{aligned}T_0 &= \text{MAOI1}(X_0, X_1, X_0, X_1)\\ T_1 &= \text{AND3}(X_3, X_2, X_3)\\ T_2 &= \text{MAOI1}(X_1, X_2, X_0, X_3)\\ T_3 &= \text{MOAI1}(X_1, X_0, X_2, X_2)\\ T_4 &= \text{MOAI1}(X_3, T_0, T_3, T_3)\\ T_5 &= \text{MOAI1}(T_3, T_0, X_0, T_1)\\ T_6 &= \text{MAOI1}(X_0, T_0, X_3, T_0)\\ T_7 &= \text{MOAI1}(X_0, T_1, T_2, T_2)\\ Y_0 = T_5 \quad Y_1 &= T_7 \quad Y_2 = T_6 \quad Y_3 = T_4\end{aligned} \tag{6}$$

#### *3.1.2. Mix-Columns Optimization*

The Mix-Columns component is a linear transformation of the input column. The input column is multiplied by a constant matrix $M$ to produce the output column. $M$ is a involutory matrix, which means $M^2 = E$, where $E$ is the identity matrix. It is easy to decrypt the ciphertext by multiplying the ciphertext

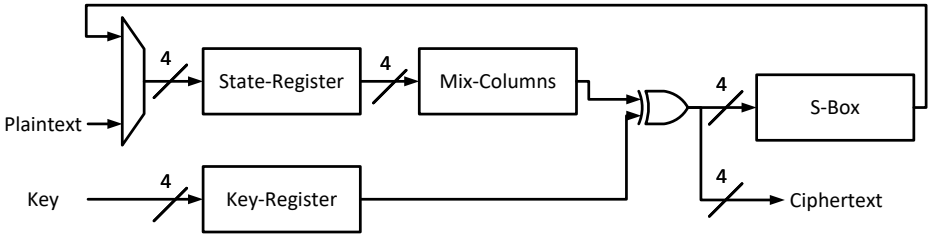


Figure 2: Serial architecture of CRAFT

Table 3: Encoding of different types of logical gate

| logical expression | $BB_i[0{:}7]$ | gate type |
|---|---|---|
| $Q_0 \oplus Q_1$ | 0 0 0 0 0 0 1 0 | XOR |
| $\sim (Q_0 \oplus Q_1)$ | 0 0 0 0 0 0 1 1 | XNOR |
| $Q_0 \wedge Q_1$ | 0 0 0 0 0 1 0 0 | AND |
| $\sim (Q_0 \wedge Q_1)$ | 0 0 0 0 0 1 0 1 | NAND |
| $Q_0 \vee Q_1$ | 0 0 0 0 0 1 1 0 | OR |
| $\sim (Q_0 \vee Q_1)$ | 0 0 0 0 0 1 1 1 | NOR |
| $\sim Q_0$ | 0 0 0 0 1 0 0 1 | NOT |
| $\sim Q_1$ | 0 0 0 0 1 0 1 1 | NOT |
| $\sim Q_2$ | 0 0 0 1 0 0 0 1 | NOT |
| $Q_0 \oplus Q_1 \oplus Q_2$ | 0 0 0 1 0 0 1 0 | XOR3 |
| $\sim (Q_0 \oplus Q_1 \oplus Q_2)$ | 0 0 0 1 0 0 1 1 | XNOR3 |
| $Q_0 \wedge Q_1 \wedge Q_2$ | 0 0 1 0 0 0 0 0 | AND3 |
| $\sim (Q_0 \wedge Q_1 \wedge Q_2)$ | 0 0 1 0 0 0 0 1 | NAND3 |
| $Q_0 \vee Q_1 \vee Q_2$ | 0 1 1 1 0 1 1 0 | OR3 |
| $\sim (Q_0 \vee Q_1 \vee Q_2)$ | 0 1 1 1 0 1 1 1 | NOR3 |
| $\sim ((Q_0 \wedge Q_1) \vee (\sim (Q_2 \vee Q_3)))$ | 1 0 1 1 0 0 0 0 | MAOI1 |
| $\sim (\sim (Q_0 \wedge Q_1) \wedge ((Q_2 \vee Q_3)))$ | 1 0 1 1 0 0 0 1 | MOAI1 |

with $M$ again. The Mix-columns component is shown in Equation 7. where $I_{0,j}$, $I_{1,j}$, $I_{2,j}$, and $I_{3,j}$ are the input column, $I'_{0,j}$, $I'_{1,j}$, $I'_{2,j}$, and $I'_{3,j}$ are the output column, and $j$ is the column index, $j \in \{0, \ldots, 3\}$.

$$\begin{bmatrix} I'_{0,j} \\ I'_{1,j} \\ I'_{2,j} \\ I'_{3,j} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{0,j} \\ I_{1,j} \\ I_{2,j} \\ I_{3,j} \end{bmatrix} \tag{7}$$

In order to reduce the area of this component, the serial architecture of Mix-Columns is utilized, as shown in Figure 3. The serial architecture of Mix-Columns requires four 4-bit registers, two multiplexers, and three XOR gates. The operation of Mix-Columns involves three distinct stages: freeze, shift, and add. During the freeze stage, the register values are kept unchanged by setting both $CM_0$ and $CM_1$ to 0. In the shift stage, a shift in the register values from $RM_0$ to $RM_4$ is induced by setting both $CM_0$ and $CM_1$ to 1. Finally, in the add stage, an addition operation on the column values is executed according to Equation 7. This is achieved by setting $CM_0$ and $CM_1$ to 0 and 1, respectively.

The timing diagram for the serial architecture of Mix-Columns is depicted in Figure 4. It requires five clock cycles to compute the next columns from the previous ones, and an additional four clock cycles to transfer data from the internal register of Mix-Columns to the State-Register. Therefore, a complete state round requires a total of 36 clock cycles.

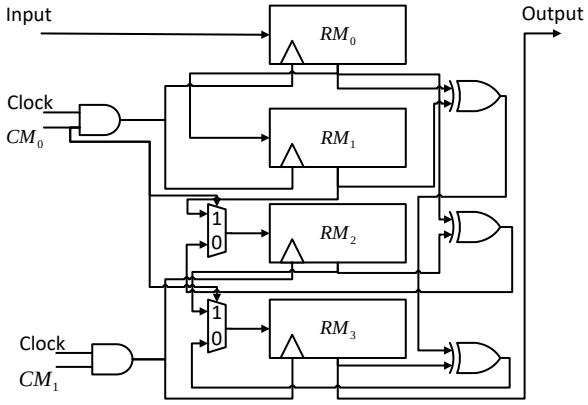


Figure 3: Serial Architecture of Mix-Columns with clock gating

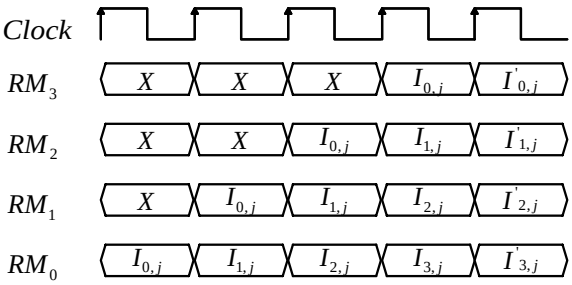


Figure 4: Timing Diagram for the Serial Architecture of Mix-Columns

### *3.1.3. Control Units*

The finite-state machine (FSM) of the serial architecture, as shown in Figure 5, begins its encryption process by storing the initial key and plaintext in the Key-Register and State-Register, respectively. The key undergoes expansion in the Key-Schedule phase, during which the clocks of the Mix-Columns and State-Register are disabled. The Mix-Columns phase follows, storing

one column of the State-Register in the Mix-Columns registers and taking five clock cycles to execute Mix-Columns on one column. Upon completion of this phase, the clocks of the State-Register and Key-Register are disabled. The S-Box phase then takes over, sending the data stored in the Mix-Columns registers back to the State-Register and XORing it with the keys, a process that takes another four clock cycles. This cycle between the Mix-Columns and S-Box phases is repeated four times for the four columns of the State-Register. The Permute operation is then executed in one clock cycle inside the State-Register. The encryption process concludes when the Round counter reaches 31, at which point the ciphertext is stored in the State-Register.

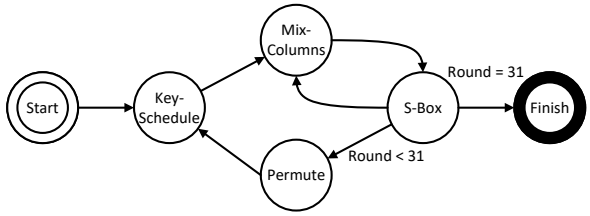


Figure 5: Finite-state machine for Serial Architecture

Clock gating, a technique discussed in [18], can help reduce the dynamic power consumption of the encryption. This technique is applied separately to the State-Register, Key-Register, and Mix-Columns. For example, during the Key-Schedule phase, the clock of the State-Register and Mix-Columns is turned off because these two blocks are not needed. This helps save a significant amount of power. Figure 6 shows the timing diagram of a design that uses the clock gating technique.

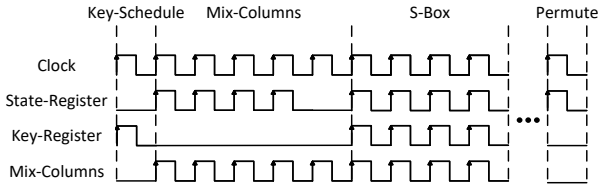


Figure 6: Timing diagram for Serial Architecture

### *3.2. Unrolled Architecture (A2)*

The architecture shown in Figure 7 includes two S-Boxes, two Mix-Columns, a Key, a State-Register, two PermuteNibbles, and one feedback path. It's designed to perform a 30-round encryption process in just 15 cycles. Only the Mix-Columns and Add-Key operations are carried out in the final cycle, finishing the encryption process in a total of 16 cycles.

The unrolled architecture, based on the iterative architecture from [9], completes the encryption process in only 16 cycles, compared to the 32 cycles needed by the iterative architecture. Here, a cycle includes two round functions of CRAFT. While this approach might use more area, it provides higher throughput at the same frequency.

### *3.3. Iterative Architecture (A3)*

The architecture from [20] operates on a round-based architecture. It employs a single round function to encrypt a block, incorporating a S-Box, a Permute, and an Add-Key operation. This round function is executed 32 times to encrypt a single block. Concurrently, the Key Schedule runs alongside the round function. This architecture is depicted in Figure 8. For comparison, different architectures are listed in Table 4.

Table 4: Different architectures description

| Architecture | Cipher | Description | Reference |
|---|---|---|---|
| A1 | CRAFT | Serial | This work |
| A2 | CRAFT | Unrolled | This work |
| A3 | PRESENT | Iterative | [20] |

## 4. Experimental Evaluation

In ASIC implementations, the gate equivalent (GE) is often used to evaluate the area consumption of a design. A single GE is equivalent to a two-input NAND gate. To calculate the area in GE, we divide the total area (in $\mu m^2$) by the area of a two-input NAND gate (also in $\mu m^2$). However, the number of GEs can vary based on the specific technology used, as discussed in [24]. For instance, the number of GEs for the same design will differ between UMC 180nm technology and TSMC 180nm technology. Therefore, GE is not suitable for comparing the area consumption of different designs on different technologies. In order to ensure a fair comparison, we evaluate the area consumption of the proposed designs using FPGA implementations, a method also utilized in [3].

### *4.1. Platform*

The proposed architectures were implemented on a Xilinx FPGA board using Vivado v2023.2. To test in various environments, three different FPGA platforms were used for benchmarking: Artix-7(xc7a100tcsg324-1), Kintex-7(xc7k70tfbg484-1), and Spartan-7(xc7s100fgga484-1). Artix-7 offers high performance in resource-limited situations. Spartan-7 is designed for high-restriction environments. Kintex-7 is ideal for applications like 3G and 4G wireless, flat panel displays, and video over IP solutions.

### *4.2. Area*

The Area metric, which includes components like Flip-Flops, LUTs, and Slices, is used to measure the area used by the proposed designs. To make a fair comparison, the FPGA's embedded memory blocks were not used. This was done by turning off the related settings in the VHDL, as suggested in [25]. Also, all designs were synthesized and implemented using the same settings, specifically, the default settings of Vivado Synthesis and Implementation.

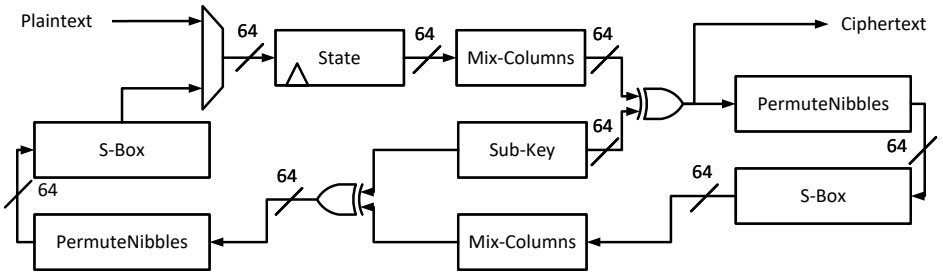


Figure 7: Unrolled architecture of CRAFT

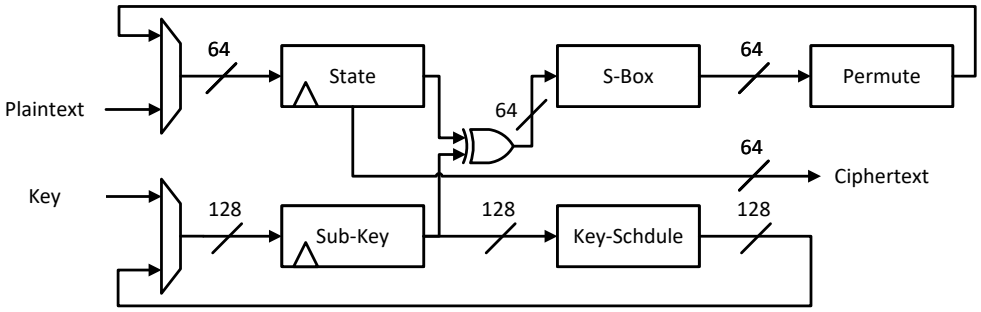


Figure 8: Iterative architecture of PRESENT

### *4.3. Throughput*

The performance of the proposed designs is evaluated using the Throughput metric. This metric uses three parameters: the maximum throughput rate, the throughput rate at 100MHz, and the throughput rate per slice. The maximum throughput rate is the highest rate that our designs can achieve, calculated using Equation 8. The throughput rate at 100MHz shows the rate achievable when the clock frequency is set to 100MHz, calculated using Equation 9. The throughput rate per slice is a measure of efficiency, calculated by dividing the throughput rate by the number of Slices (Equation 10). In these calculations, the Plaintext Size is 64-bit, Latency refers to the number of clock cycles required to encrypt a single block, and Slices refers to the number of Slices used by the design.

$$MaxThroughput(Thr) = \frac{MaxFrequency \times BlockSize}{Latency} \quad (8)$$

$$Throughput_{@100MHz}(Thr^*) = \frac{100MHz \times BlockSize}{Latency} \quad (9)$$

$$ThroughputPerSlice = \frac{Thr}{Slices} \quad (10)$$

### *4.4. Power and Energy*

The Power metric, which includes both dynamic and static power consumption, is used to evaluate the power consumption of the proposed designs, as defined in Equation 11. On the other hand, the Energy metric measures the energy consumption of the designs. It's calculated by multiplying the power consumption by the time needed to encrypt a single block. This time is determined by dividing the latency by the frequency, as explained in Equation 12.

$$\begin{aligned} \text{Total Power}(TP) = \text{Dynamic Power}(DP) \\ + \text{Static Power}(SP) \end{aligned} \quad (11)$$

$$Energy(E) = \frac{TP \times Latency}{Frequency} \quad (12)$$

## 5. Results

This section presents the results of the proposed designs, divided into three categories: area, throughput, and power and energy. The results are demonstrated across three different FPGA platforms: Artix-7, Kintex-7, and Spartan-7. The area consumption of the designs is displayed in Table 5, the throughput results are illustrated in Table 6, and the details of power and energy consumption are provided in Table 7.

The area consumption of the proposed designs is evaluated based on three factors: Flip-Flops (FF), Look-Up Tables (LUT), and Slices. These designs are compared with the iterative architecture of PRESENT (A3) , as described in [20]. The results indicate that the proposed designs consume 15.72% less area than the iterative architecture of PRESENT.

Table 5: Area used for the three Architectures

| Platform | Design | $State(bit)$ | $Key(bit)$ | $FF$ | $LUT$ | $Slices$ |
|---|---|---|---|---|---|---|
| Artix-7 | A1 | 64 | 128 | **144** | **177** | **59** |
| | A2 | 64 | 128 | 157 | 378 | 111 |
| | A3 | 64 | 128 | 201 | 243 | 70 |
| Kintex-7 | A1 | 64 | 128 | **144** | **178** | **58** |
| | A2 | 64 | 128 | 157 | 377 | 115 |
| | A3 | 64 | 128 | 201 | 244 | 68 |
| Spartan-7 | A1 | 64 | 128 | **144** | **177** | **57** |
| | A2 | 64 | 128 | 157 | 381 | 118 |
| | A3 | 64 | 128 | 201 | 244 | 70 |

Table 6: Throughput results for the three Architectures

| Platform | Design | $Latency$ | $MaxF(MHz)$ | $Thr(Mbps)$ | $Thr^*(Mbps)$[a] | $Thr/Slices(Kbps/Slices)$ |
|---|---|---|---|---|---|---|
| Artix-7 | A1 | 1215 | **557.41** | 29.36 | 5.27 | 497.65 |
| | A2 | **16** | 142.38 | 569.52 | **400.00** | 5130.81 |
| | A3 | 32 | 274.04 | 548.08 | 200.00 | 7829.71 |
| Kintex-7 | A1 | 1215 | **853.97** | 44.98 | 5.27 | 775.57 |
| | A2 | **16** | 175.25 | 701.00 | **400.00** | 6095.65 |
| | A3 | 32 | 357.78 | 715.56 | 200.00 | 10522.94 |
| Spartan-7 | A1 | 1215 | **525.76** | 27.69 | 5.27 | 485.87 |
| | A2 | **16** | 138.86 | 555.44 | **400.00** | 4707.12 |
| | A3 | 32 | 296.29 | 592.58 | 200.00 | 8465.43 |

[a] Throughput rate at 100MHz

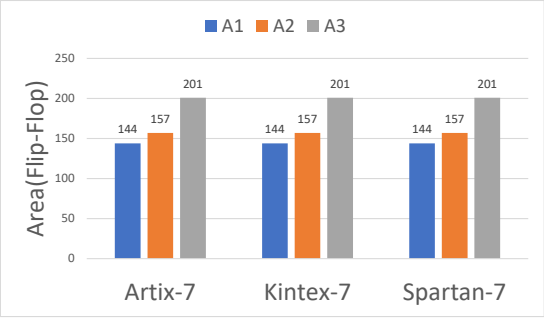


Figure 9: Comparison of Flip-Flop in three Architectures

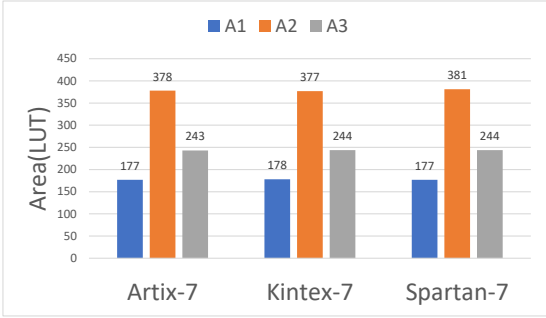


Figure 10: Comparison of Look-Up Tables in three Architectures

Regarding the Flip-Flops (FF), the key schedule of the CRAFT cipher is implemented using multiplexers. This eliminates the need for FF to store the sub-key, resulting in a lower FF count compared to other ciphers. This is a significant factor contributing to the CRAFT cipher's requirement of less than 1000 GE, which is the lowest known requirement on the IBM 130 nm ASIC library, as shown in [9]. A comparison of FF counts is provided in Figure 9.

As illustrated in Figure 10, when it comes to Look-Up Tables (LUT), the proposed designs (A1) require fewer LUTs than the iterative architecture of PRESENT (A3). This is attributed to the fact that the proposed designs utilize a single S-Box, in contrast to the 16 S-Boxes used by the iterative architecture of PRESENT. Furthermore, the proposed designs also require fewer LUTs than the unrolled architecture of CRAFT, which uses 32 S-Boxes, compared to just one in the proposed designs.

The serial architecture (A1) has a lower Slices cost compared to the iterative architecture of PRESENT (A3), thanks to the reduction in FF and LUT usage. Among the platforms, Spartan-7 is the most efficient in terms of Slices, followed by Artix-7 and Kintex-7. These results are illustrated in Figure 11. However, the lower Max Frequency of Spartan-7 will be considered in the

Table 7: Power and Energy consumption for the three Architectures

| Platform | Design | $DP(mW)$ | $SP(mW)$ | $TP(mW)$ | $E(uJ)$ | $E/bit(nJ/bit)$ |
|---|---|---|---|---|---|---|
| Artix-7 | A1 | 2.00 | 139.00 | 141.00 | 1.71 | 26.77 |
| | A2 | 7.00 | 139.00 | 146.00 | **0.02** | **0.37** |
| | A3 | 2.00 | 139.00 | 141.00 | 0.05 | 0.71 |
| Kintex-7 | A1 | 2.00 | 145.00 | 147.00 | 1.79 | 27.91 |
| | A2 | 8.00 | 145.00 | 153.00 | **0.02** | **0.38** |
| | A3 | 3.00 | 145.00 | 148.00 | 0.05 | 0.74 |
| Spartan-7 | A1 | 2.00 | 140.00 | 142.00 | 1.73 | 26.96 |
| | A2 | 7.00 | 140.00 | 147.00 | **0.02** | **0.37** |
| | A3 | 3.00 | 140.00 | 143.00 | 0.05 | 0.72 |

DP: Dynamic Power SP: Static Power TP: Total Power E: Energy

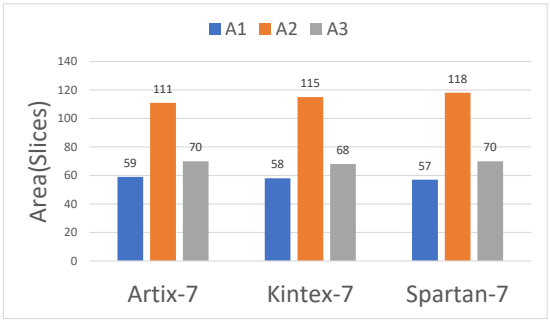


Figure 11: Comparison of Slices in three Architectures

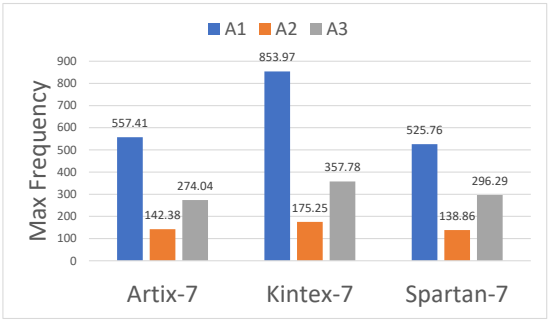


Figure 12: Comparison of Max Frequency in three Architectures

Throughput comparison.

Figure 12 illustrates that the proposed designs (A1) have a higher Max Frequency than the iterative architecture of PRESENT (A3). This improvement is due to two key factors. First, the S-Box is optimized with the GEC encoding scheme, reducing its delay. Second, the serial architecture of the design further reduces the overall delay of the encryption process. However, among all platforms, Spartan-7 has the lowest Max Frequency, primarily because it has the fewest LUTs, as shown in Figure 10. The unrolled architecture (A2) reduces the latency to 16, which doubles the Throughput rate at 100MHz compared to the iterative architecture of PRESENT (A3). This data is presented in Table 6.

The serial architecture (A1) has the highest energy per bit due to its higher latency, which results in the smallest area. Conversely, the unrolled architecture (A2) has the lowest energy per bit because it has the lowest latency. The energy per bit of the iterative architecture of PRESENT (A3) falls between that of the serial architecture (A1) and the unrolled architecture (A2). Compared to the iterative architecture of PRESENT (A3), the unrolled architecture (A2) reduces energy per bit by 47.89%. Figure 13 illustrates the energy per bit for the three architectures.

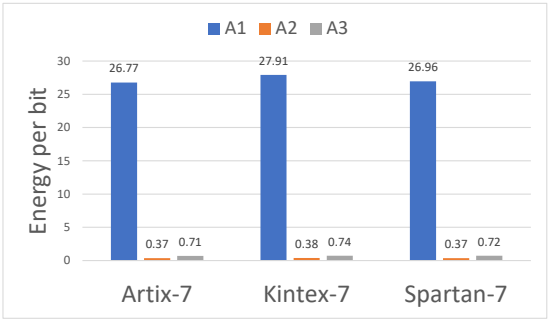


Figure 13: Comparison of Energy per bit in three Architectures

## 6. Conclusion

The Internet of Things (IoT) has brought about a revolution in the interaction with devices and data, enabling unprecedented levels of connectivity and automation. However, the resource-constrained environments in which IoT devices often operate pose challenges for the implementation of robust security measures. Lightweight cryptographic algorithms, such as the CRAFT cipher, are crucial for ensuring these devices' security without overtaxing their limited resources.

This paper presents two architectures for the CRAFT cipher: Serial and Unrolled. The Serial architecture reduces the area consumption by 15.72% compared to the iterative architecture of PRESENT. The Unrolled architecture, on the other hand, reduces the latency to 16, effectively doubling the throughput rate at 100MHz compared to the iterative architecture of PRESENT. Additionally, the Unrolled architecture reduces energy per bit by 47.89% compared to the iterative architecture of PRESENT. These proposed architectures are well-suited for resource-constrained environments, such as those found in IoT devices.

Future work could involve investigating the application of these proposed architectures in real-world IoT devices and measuring their performance. Additionally, implementing these architectures on ASICs could further reduce area consumption.

## References


[1] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, A. Zanella, Iot: Internet of threats? A survey of practical security vulnerabilities in real iot devices, IEEE Internet Things J. 6 (5) (2019) 8182–8201. doi:10.1109/JIOT.2019.2935189.

[2] D. Swessi, H. Idoudi, A survey on internet-of-things security: Threats and emerging countermeasures, Wirel. Pers. Commun. 124 (2) (2022) 1557–1592. doi:10.1007/S11277-021-09420-0.

[3] K. Mohajerani, R. Haeussler, R. Nagpal, F. Farahmand, A. Abdulgadir, J. Kaps, K. Gaj, FPGA benchmarking of round 2 candidates in the NIST lightweight cryptography standardization process: Methodology, metrics, tools, and results, IACR Cryptol. ePrint Arch. (2020) 1207.
URL https://eprint.iacr.org/2020/1207

[4] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, C. Vikkelsoe, PRESENT: an ultra-lightweight block cipher, in: P. Paillier, I. Verbauwhede (Eds.), Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings, Vol. 4727 of Lecture Notes in Computer Science, Springer, 2007, pp. 450–466. doi:10.1007/978-3-540-74735-2_31.
URL https://doi.org/10.1007/978-3-540-74735-2_31

[5] J. Guo, T. Peyrin, A. Poschmann, M. J. B. Robshaw, The LED block cipher, in: B. Preneel, T. Takagi (Eds.), Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings, Vol. 6917 of Lecture Notes in Computer Science, Springer, 2011, pp. 326–341. doi:10.1007/978-3-642-23951-9_22.
URL https://doi.org/10.1007/978-3-642-23951-9_22

[6] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, F. Regazzoni, Midori: A block cipher for low energy, in: T. Iwata, J. H. Cheon (Eds.), Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II, Vol. 9453 of Lecture Notes in Computer Science, Springer, 2015, pp. 411–436. doi:10.1007/978-3-662-48800-3_17.
URL https://doi.org/10.1007/978-3-662-48800-3_17

[7] L. Li, B. Liu, H. Wang, Qtl: A new ultra-lightweight block cipher, Microprocessors and Microsystems 45 (2016) 45–55. doi:10.1016/j.micpro.2016.03.011.

[8] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, Y. Todo, GIFT: A small present - towards reaching the limit of lightweight encryption, in: W. Fischer, N. Homma (Eds.), Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings, Vol. 10529 of Lecture Notes in Computer Science, Springer, 2017, pp. 321–345. doi:10.1007/978-3-319-66787-4_16.
URL https://doi.org/10.1007/978-3-319-66787-4_16

[9] C. Beierle, G. Leander, A. Moradi, S. Rasoolzadeh, CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks, IACR Trans. Symmetric Cryptol. 2019 (1) (2019) 5–45. doi:10.13154/TOSC.V2019.I1.5-45.

[10] Y. Guo, L. Li, B. Liu, Shadow: A lightweight block cipher for iot nodes, IEEE Internet Things J. 8 (16) (2021) 13014–13023. doi:10.1109/JIOT.2021.3064203.

[11] J. Yang, L. Li, Y. Guo, X. Huang, DULBC: A dynamic ultra-lightweight block cipher with high-throughput, Integr. 87 (2022) 221–230. doi:10.1016/J.VLSI.2022.07.011.

[12] X. Huang, L. Li, J. Yang, IVLBC: an involutive lightweight block cipher for internet of things, IEEE Syst. J. 17 (2) (2023) 3192–3203. doi:10.1109/JSYST.2022.3227951.

[13] Y. Belkheyar, J. Daemen, C. Dobraunig, S. Ghosh, S. Rasoolzadeh, Bipbip: A low-latency tweakable block cipher with small dimensions, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2023 (1) (2023) 326–368. doi:10.46586/TCHES.V2023.I1.326-368.

[14] Q. Song, L. Li, X. Huang, Lelbc: A low energy lightweight block cipher for smart agriculture, Internet of Things 25 (2024) 101022. doi:10.1016/j.iot.2023.101022.

[15] A. A. Zakaria, A. H. Azni, F. Ridzuan, N. H. Zakaria, M. Daud, Systematic literature review: Trend analysis on the design of lightweight block cipher, J. King Saud Univ. Comput. Inf. Sci. 35 (5) (2023) 101550. doi:10.1016/J.JKSUCI.2023.04.003.

[16] C. A. Lara-Nino, A. Diaz-Perez, M. Morales-Sandoval, Lightweight hardware architectures for the present cipher in FPGA, IEEE Trans. Circuits Syst. I Regul. Pap. 64-I (9) (2017) 2544–2555. doi:10.1109/TCSI.2017.2686783.

[17] J. G. Pandey, T. Goel, A. Karmakar, Hardware architectures for PRESENT block cipher and their FPGA implementations, IET Circuits Devices Syst. 13 (7) (2019) 958–969. doi:10.1049/IET-CDS.2018.5273.

[18] K. Shahbazi, S. Ko, Area-efficient nano-aes implementation for internet-of-things devices, IEEE Trans. Very Large Scale Integr. Syst. 29 (1) (2021) 136–148. doi:10.1109/TVLSI.2020.3033928.

[19] L. Li, J. Feng, B. Liu, Y. Guo, Q. Li, Implementation of PRINCE with resource-efficient structures based on fpgas, Frontiers Inf. Technol. Electron. Eng. 22 (11) (2021) 1505–1516. doi:10.1631/FITEE.2000688.

[20] R. Bharathi, N. Parvatham, Light-weight present block cipher model for iot security on fpga., Intelligent Automation & Soft Computing 33 (1) (2022) 35–49. doi:10.32604/iasc.2022.020681.

[21] J. Yang, L. Li, X. Huang, Low area and high throughput hardware implementations for the lilliput cipher, International Journal of Circuit Theory and Applications (2023). doi:10.1002/cta.3892.

[22] Z. Bao, J. Guo, S. Ling, Y. Sasaki, Peigen–a platform for evaluation, implementation, and generation of s-boxes, IACR Transactions on Symmetric Cryptology (2019) 330–394doi:10.46586/tosc.v2019.i1.330-394.

[23] J. Feng, Y. Wei, F. Zhang, E. Pasalic, Y. Zhou, Novel optimized implementations of lightweight cryptographic s-boxes via sat solvers, IEEE Transactions on Circuits and Systems I: Regular Papers (2023). doi:10.1109/tcsi.2023.3325559.

[24] M. S. Turan, K. McKay, D. Chang, L. E. Bassham, J. Kang, N. D. Waller, J. M. Kelsey, D. Hong, Status report on the final round of the nist lightweight cryptography standardization process (2023). doi:10.6028/nist.ir.8454.

[25] A. Xilinx, Ultrafast design methodology guide for xilinx fpgas and socs: Super logic region (2022).