Penerbit Akademia Baru

Journal of Advanced Research in Computing and Applications

Journal homepage:
https://akademiabaru.com/submit/index.php/arca/index
ISSN: 2462-1927

Compact Hardware Implementation of The CLEFIA Block Cipher

Jia Jian Tew[1], Chia Yee Ooi[1,*], Chong Yeam Tan[2]

[1] Embedded System iKohza, Electronic System Electronic Department, Malaysia-Japan International Institute of Technology, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia
[2] SkyeChip Sdn Bhd, Bayan Lepas, Penang, Malaysia

ARTICLE INFO

Article history:
Received 10 January 2023
Received in revised form 18 February 2023
Accepted 5 March 2023
Available online 20 March 2023

Keywords:
CLEFIA block cipher; hardware implementation; algorithm

ABSTRACT

This study presents the implementation of the CLEFIA block cipher, a lightweight symmetric encryption algorithm, with a focus on its application in secure communication and data protection. In recent years, several lightweight block ciphers for hardware implementation have been proposed. Block ciphers are used to protect data in cryptographic applications. CLEFIA is known for its strong security properties and efficient performance, making it suitable for resource-constrained environments. The objective of this report is to implement CLEFIA algorithm in hardware description language and to develop a hardware implementation of the CLEFIA algorithm with less memory space requirement. The report provides an overview of the CLEFIA algorithm, including its round structure, key expansion, and encryption/decryption processes. The implementation process utilizes Verilog, a hardware description language, to design the 128-bit key length of CLEFIA hardware modules. The VCS simulation tool is employed for functional verification, ensuring the correctness of the implementation. Additionally, Design Compiler, a synthesis tool, is utilized for optimizing the design and generating efficient gate-level representations. By incorporating modern tools like VCS, and Design Compiler, along with a modular design technique, the report presents a practical and efficient approach to implementing 128-bit key length of CLEFIA. The use of concurrency and optimized circuitry to carry out high-speed encryption operations is highlighted in the discussion of 128-bit key length of CLEFIA's implementation in hardware. This report outcome has successfully achieved the implementation of 128-bit key length of CLEFIA and successfully reduce the Gate Equivalence (GE) by 76.07% after replacing (96x32)-bit Memory block with Constant Generator and Round Key Generator. This report also examines the possible advantages of CLEFIA implementation in hardware, including improved performance, resource efficiency, and effortless integration with current systems and protocols.

1. Introduction

Several lightweight block ciphers for hardware implementation have been proposed in recent years. Block ciphers are used in cryptography applications to protect data. These cryptographic fundamentals have been a key focus of study in the field [1]. The advanced encryption standard (AES), one of the most significant and useful block ciphers, has been replaced by a number of lightweight block ciphers that consume less hardware. The cost of implementing the AES in hardware might be

* *Corresponding author.*
E-mail address: ooichiayee@utm.my (Chia Yee Ooi)

prohibitive [2]. The block cipher CLEFIA [3] is suited for hardware implementation, such as embedded crypto-processors used in cryptographic application systems. It has a 128-bit data block size and 128, 192, and 256-bit key length. The SONY cooperation created this block cipher to offer versatility and security in cryptographic applications [3].

Through the use of methods including diffusion switch mechanisms, several diffusion matrices, and two different non-linear S-boxes, this block cipher increases the security of the encryption process [4]. The International Organisation for Standardization (ISO) and the International Electrotechnical Commission (IEC) (ISO/IEC 29192-2) standardized the CLEFIA block cipher [5]. Additionally, the CLEFIA is suggested to the Internet Engineering Task Force and Application for Cryptographic Techniques towards the Revision of the e-Government Recommended Ciphers List for use on the Internet and the e-Government. The objective of this research is to implement CLEFIA algorithm in hardware description language and to develop a hardware implementation of the CLEFIA algorithm with less memory space requirement.

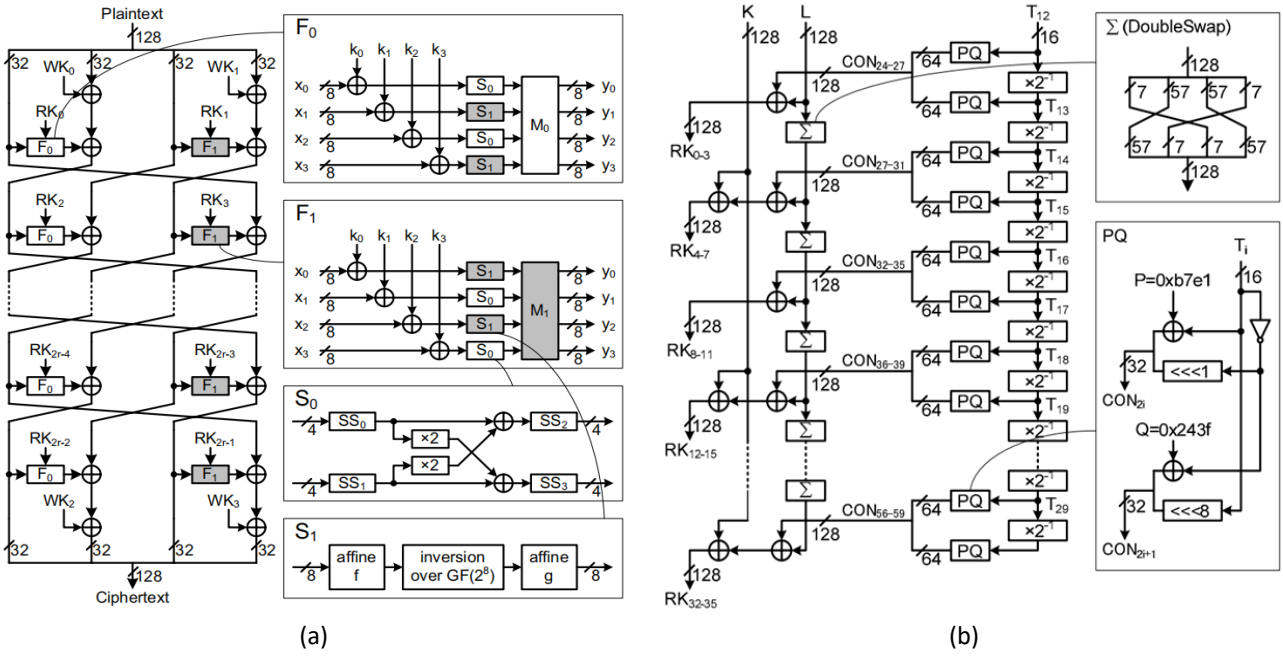

Fig. 1. Structures for (a) Data Processing block, (b) Key Scheduling block in CLEFIA algorithm

The Generalise Feistel Network (GFN) structure, an n-branch enlargement of the Feistel Network, was used in Figure 1a to represent a data processing block of the CLEFIA algorithm. A 128-bit block is divided into four 32-bit subblocks using a four-branch GFN in CLEFIA. For data randomization, F0 and F1 32-bit input/output F-function types are employed. The 8-bit input/output S-boxes, XOR operations, and multiplications by the diffusion matrices M0 and M1 make up the F-functions.

A multiplication inverse over the Galois field GF(28) and an affine transformation like AES and Camellia are used to define the S-box S1. The four 4-bit input/output S-boxes (SS0~SS3) and constant multiplications over GF(24) are combined to form the S-box S0.

Figure 1(b) displays the crucial CLEFIA scheduler block. A secret key K, an intermediate key L, and constant values CONi are XORed to produce round keys RKi. Utilizing the GFN, the intermediate key L is created by combining K and CONi. A DoubleSwapfunction Σ, which is described as a bit-wise permutation, modifies the intermediate key L. Constant multiplications over GF(216) are used to define the constant values CONi, which are made up of the smaller constants T, P, and Q [3].

2. Proposed CLEFIA Block

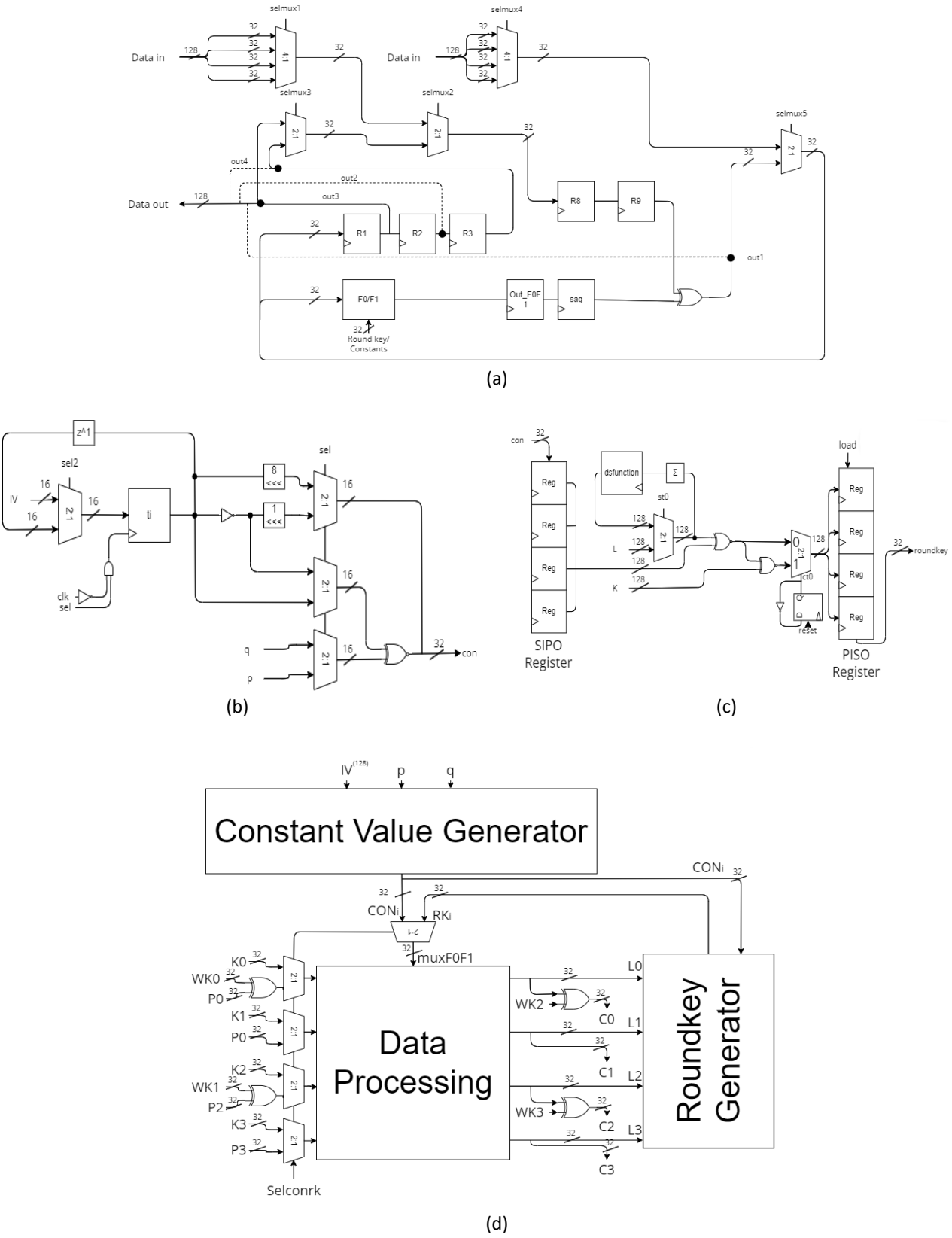

Fig. 2. Implementation of hardware structures for (a) data processing block, (b) constant generator, (c) round key generator, (d) 128bit key length CLEFIA block architecture

Figure 2(a) shows the structure for the generation of the constant values. In this structure, one constant value is generated at each Clock Cycle (CC). The initial value IV $^{(128)}$ is loaded into register ti and $CON_0^{(128)}$ are generated in the first CC, while the sel2 signal is equal to '1'. In the 3rd CCs, the sel2 signal is equal to '0' and the next constant values, $CON_1^{(128)}$ are computed. The 60 constant values for 128-bit keys are generated by the structure in 60 CCs. For implementation, multiplication by $(0x0002)^{-1}$ or (z^{-1}) ie $B=z^{-1}A$ which the B are shown in Eq. (1).

$$B = [a_0, (a_{15} \oplus a_0), a_{14}, (a_{13} \oplus a_0), a_{12}, (a_{11} \oplus a_0), a_{10}, a_9, a_8, a_7, a_6, (a_5 \oplus a_0), (a_4 \oplus a_0), a_3, a_3, a_1] \quad (1)$$

The structures for the generation of round keys for cases of 128-bit main keys are shown in Figure 2(b). Multiplexer with control signal Ct0 and D flip-flop is used for implementation of two conditions, namely odd i and even i. If i is an even number, data input 0 of the multiplexer is selected and for the odd number of I, data input 1 of the multiplexer is selected.

Therefore, in these structures, for loops with even number and odd number the output of the D flip-flop is equal to '0' and '1', respectively. The Serial in Parallel Out (SIPO) register was used to receive 32-bit output from Constant Generator block to ensure that the con input with the 128-bit of the constant. The Parallel In Serial Out (PISO) register was used to ensure the output of the Round Key Generator will output 32-bit of round key in 1 CC. The 36 constant values for 128-bit keys are generated by the structure in 36 CCs.

Figure 2(c) illustrates the Data Processing architecture, indicating the width of the data path. This architecture can process a round of the GFN encryption function in just 2 clock cycles (CC). With a 128-bit key, it requires 36 CCs to complete 18 rounds of the 4-branch GFN encryption process. Each round involves fetching the respective round key, RKi, into the F-Functions F0/F1 block within 1 CC. The F-Function block applies a substitution process to the input block by dividing it into four 32-bit words and replacing each word using predefined substitution tables (S0 and S1). Following the substitution process, the F-Function block applies a diffusion process involving linear transformation and bitwise operations to mix the block's bits (M0 and M1).

After the F-Function block, the data updates the Output_F0F1 register and prepares for the next round of XOR operation. This iterative process repeats for 36 rounds to accommodate a 128-bit key. The whitening keys (WK) are derived from the original key, with the 64 most significant bits serving as the whitening keys for the first round of encryption, and the remaining bits acting as the whitening keys for the last round. The combination of out4, out3, out2, and out1 produces the 128-bit cipher text output after the encryption process.

The selmux1 and selmux4 selectors determine the desired data input into the data path. Once the desired input is fetched into the data path, selmux2 and selmux5 control the data within the path during the iterative process. Lastly, selmux3 determines the correct data to XOR with the F-Functioned data.

In the 128-bit CLEFIA block cipher, Data Processing block is used for generating the 128-bit of L key with the combination of 32-bit of L0, L1, L2, and L3 at (the first part) and encryption processing, 128-bit of cipher text, C with the combination of 32-bit of C0, C1, C2, and C3 (the second part) in two separate times. The proposed structure of CLEFIA block cipher for the 128-bit key is shown in Figure 2(d).

With the control of the selection signal Selconrk, the circuit is able to switch between the two inputs which are 128-bit of Plain text, P with the combination of 32-bit of P0, P1, P2, and P3 and 128-bit of Key, K with the combination of 32-bit of K0, K1, K2, and K3 for 2 different computation of L key or Cipher text, C. With the change of signal Selconrk, the input to the F-Function block also will switch between con and round key.

3. Results

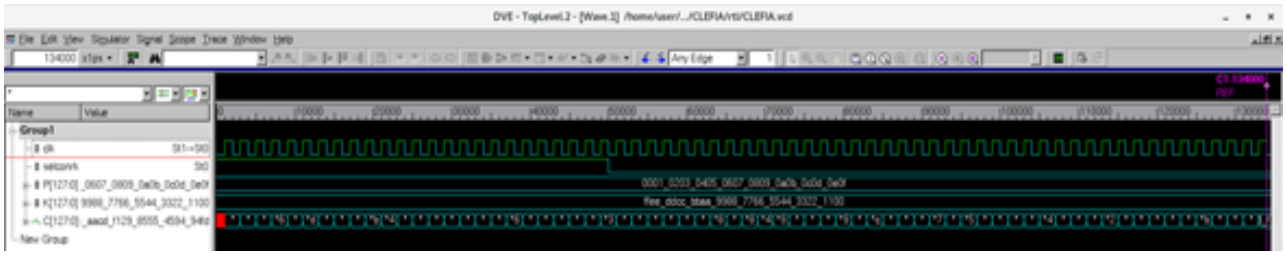

Fig. 3. Simulation result for Timing Diagram of proposed 128bit CLEFIA architecture in DVE

Figure 3 shows the timing diagram of the proposed CLEFIA block computation of the L key and cipher text, C. The simulation result shows that proposed CLEFIA block needs 68 CCs to finish the computation of L key and cipher text, C. The 128-bit CLEFIA block takes 26 CCs to compute L key and 42 CCs to compute cipher text, C.

During the computation of L key, the first CC in Data Processing block is the buffer time for the Constant Generator to generate value. Data Processing block also needs 1 CC waiting time to process the correct output, making the computation of L key taking 2 extra more CCs than the expected timing.

Table 1
Comparison of Gate Equivalence (GE) between the components

Component	Gate Equivalence (GE)	Reduction of GE (%)
Constant Generator and Round Key Generator	5230.90	76.07
(96x32)-bit Memory Block	21858.35	

Performance of the hardware implementation of the CLEFIA algorithm with less memory space requirement in term of Gate Equivalence (GE) is discussed. To reduce the memory space requirement, modules Constant Generator and Round Key Generator were implemented. Without these modules, (96x32)-bit of memory block is needed to store the 60 constant values and 36 round keys value for 128-bit CLEFIA. Bigger memory is needed for 196-bit and 256-bit of CLEFIA.

The area of modules Constant Generator and Round Key Generator was compared with the (96x32)-bit of memory block in term of Gate Equivalent (GE). GE stands for a unit of measure which allows specifying manufacturing-technology-independent complexity of digital electronic circuits. Before the comparison, Verilog code for modules Constant Generator, Round Key Generator, and (96x32)-bit of memory block were compiled by using DC compiler with the library cb13fs120_tsmc_max.

Table 1 shows the comparison result between the Constant Generator and Round Key Generator modules and (96x32)-bit Memory Block. The area in term of GE was reduced for 76.07%.

4. Conclusions

In conclusion, the objectives of this project were to implement the CLEFIA algorithm using a hardware description language (Verilog) and to develop a hardware implementation of the algorithm that requires less memory space by reducing the GE needed by 76.07%. By achieving these objectives, several benefits can be obtained. CLEFIA offers a strong level of security against various cryptographic attacks. Implementation of CLEFIA in hardware can provide additional protection against side-channel attacks, such as timing analysis and power analysis, by carefully designing the hardware circuitry to minimise leakage of sensitive information. Integrating CLEFIA in hardware allows for seamless integration into existing systems and protocols. Hardware implementations can be easily incorporated into various applications, including secure communication channels, storage devices, and cryptographic processors, enabling efficient and secure data protection. Furthermore, by focusing on reducing the memory space requirement in the hardware implementation, the project aims to optimize resource utilization. This optimization can lead to more efficient utilization of the available hardware resources, making the implementation suitable for resource-constrained environments such as embedded systems. Successfully achieving these objectives contributes to the advancement of secure and efficient cryptographic systems. The implementation of the CLEFIA algorithm in Verilog and the reduction in memory space requirement bring the benefits of improved performance, reduced hardware resource usage, and enhanced compatibility with various applications and platforms.

Acknowledgement

The authors would like to thank Malaysia-Japan International Institute of Technology, Universiti Teknologi Malaysia (UTM) for their support in this research.

References

[1] Hatzivasilis, George, Konstantinos Fysarakis, Ioannis Papaefstathiou, and Charalampos Manifavas. "A review of lightweight block ciphers." Journal of cryptographic Engineering 8 (2018): 141-184. https://doi.org/10.1007/s13389-017-0160-y

[1] Farashahi, Reza Rezaeian, Bahram Rashidi, and Sayed Masoud Sayedi. "FPGA based fast and high-throughput 2-slow retiming 128-bit AES encryption algorithm." Microelectronics journal 45, no. 8 (2014): 1014-1025. https://doi.org/10.1016/j.mejo.2014.05.004

[2] The 128 bit block cipher CLEFIA: algorithm specification, Sony Corporation, Version 1, 2010.

[3] Shirai, Taizo, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. "The 128-bit blockcipher CLEFIA." In *Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers 14*, pp. 181-195. Springer Berlin Heidelberg, 2007. https://doi.org/10.1007/978-3-540-74619-5_12

[4] CLEFIA standardization in ISO/IEC 29192-2 (Online).