Efficient Modular Adders for Scalable Encryption Algorithm

K.J. Jegadish Kumar
Assistant Professor
SSN College of Engineering
Kalavakkam-603110 Chennai,

K.Chenna Kesava Reddy
Principal
Jyothismathi College of
Engineering and Technology
Shamirpet-500078, India

S. Salivahanan
Principal
SSN College of Engineering
Kalavakkam-603110
Chennai, India

ABSTRACT

SEA – Scalable Encryption Algorithm is a block cipher based symmetric encryption scheme, specially designed for resource constrained devices. SEA proposes low computational encryption routines (i.e. less code size, memory and power) for processors with a restricted instruction set. SEA is parametric with plain-text, key and microprocessor size, and meant for efficient combination of encryption/decryption and key derivation. The performance of modified SEA using efficient architectures of 2^b and 2^b-1 modular adders in a Field programmable gate array (FPGA) device is investigated. In this paper, an iterative based loop design of the block cipher is first implemented on FPGA. The proposed modular adders in SEA achieve lower area and power consumption on the target platform VIRTEX-4, xc4vl25-10ff668. Beyond its low cost performances, the proposed architecture is fully flexible with any parameters and takes advantage of generic VHDL coding.

Keywords: Block ciphers, constrained applications, Modular adders, FPGA implementation.

1. INTRODUCTION

Symmetric encryption schemes designed for resource constrained devices do not have a long history. Remarkable examples of such encryption schemes are the Tiny Encryption Algorithm (TEA) or Yuval's proposal [1]. However, both of them are old and could not afford commendable security against attacks such as linear and differential cryptanalysis [2]. Recent ciphers fairly concentrate on finding a good trade-off between cost, security and performance. Consequently, there arises a requirement for a new cryptosystem that endows with apt solution for resource constrained systems. Embedded applications that are basic building infrastructures represent a noteworthy opportunity and challenge for new cryptosystem like Scalable Encryption Algorithm (SEA) [1, 2,].

1.1 SEA: Delineation

The primary purpose for SEA is to implement in limited processing resources (*e.g.* a small processor); the proposed architecture is parametric with respect to plain-text, cipher-text, key and the processor size. Since the architecture is parametric in nature, there is flexibility of implementing in all platforms with minimum code change. Most algorithms perform differently on different platforms. But SEA is an exception. This is due to the fact that for a given processor, SEA uses a smaller sized ciphering routine and the security is achieved based on the key size. Since we operate on a limited resource processor, only some basic operations like XOR, AND, OR, efficient mod 2^b addition are done [3].

1.2 Related Work

There are many cryptographic algorithms that require high or moderate processing power and area. They are Advanced Encryption Standard (AES) [4-7], Data Encryption Standard (DES) [8], Tiny Encryption Algorithm (TEA) [9,10], and Extended TEA (XTEA) [11]. These encryption algorithms are not suited to be implemented in a resource constrained system due to various complexities involved like i.e. Non-scalability, Processor Intensive, and Security Level [2].

AES (Rijndael) [4-7] comprises three block ciphers, all the block cipher vary depending on the number of bits. AES is a fixed block cipher of 128 bits with a key size of 128,192,256. AES requires four 256 entry, 32 bit tables, so totally 4096 of memory which equals 1kB for each table. AES is more processor intensive and is non-scalable, so it cannot be implemented on constrained systems. Though there are efficient implementations of AES, there are still non- scalable for need of any processing platform.

DES [8] is based on symmetric key algorithms of bit size 56. DES is the classic symmetric key encryption algorithm that receives a finite length of plain-text bits and alters through the series of complex operations into a different bit sequences known as cipher-text by using the similar key. Though DES is not highly secured, it is widely used in a mode of operation as suggested by Federal Information Processing Standard (FIPS-81). DES is more processor intensive, non-scalable and breakable by Linear Cryptanalysis.

TEA [9] or Yuval's proposal [10] is notable for its simplicity and implementable on various platforms (scalability). It works on 64 bit blocks and makes use of 128 bit key. When crypt is analyzed with equivalent key, each key gives three other keys. So in terms of security TEA is insecure. XTEA[11] was an advanced version of TEA, mainly aimed at improving all the security glitches. XTEA has complex key scheduling and rearrangement of Shift XOR and addition operations. XTEA is vulnerable to related key differential attack. Like SEA, HIGHT [12] is also another Block Cipher for resource constrained

systems, but it is non-scalable and consumes more number of gates and the throughput and operating frequency are much less when compared to SEA. So, implementing SEA for constrained systems is a better option.

This paper is organized as follows: The introduction and literature survey are provided in Section 1. Section 2 describes parameters, definitions and basic operations for implementation of SEA. Section 3 illustrates hardware implementation of efficient Modular adders. In Section 4, implementation results of different modular adder architectures are presented and compared. Finally, conclusion is given in Section5.

2. SEA IMPLEMENTATION

Recent symmetric ciphers observe tradeoffs between the cost and implementation of hardware/software, which has influence in their performance. They are especially designed for efficient implementations on wide range of applications. SEA on other hand defines to be better choice as they are friendly enough to restrict processing resources and achieve high throughput. Generally, design objective of SEA is to be a cost effective cipher and certification schemes for processors having restricted instruction set. Similar to AES [5-8] and DES, Ciphers SEA also combines plain-text and key and are parameterized by bus sizes. In contrast, solutions that are old for low cost ciphers like TEA or Yuval's proposal, SEA additionally promotes a resistance to cryptanalysis. When put into practice, SEA was demonstrated to be convenient for embedded applications using microcontrollers and low cost hardware implementations. However, SEA's efficient hardware implementation and performances are described by Standaert et.al [1, 2] are yet to be investigated to meet efficiency in hardware costs. In [3], we proposed a modified SEA with a simple modular adder and successfully achieved efficiency in terms of area, power and Computation speed. This paper therefore further proposes to investigate this algorithm, modified for area and power in constrained applications, with different variants of modular adders. The investigation begins with an exploration of the quality of a cost effective FPGA implementation of SEA [1, 2] and our aim is to modify SEA using efficient modular adders as in Beuchat [13] to reduce the hardware complexities.

2.1 Explanation of Algorithm

2.1.1 Fundamental Operations

Due to its simplicity constraints, $SEA_{n,b}$ is based on a restricted number of uncomplicated operations (Chosen for their ease of use in any processing device) denoted as bitwise XOR , substitution box S(S-Box), word (left) rotation R and inverse word rotation R^{-1}, bit rotation r, addition mod 2^b. The thorough explanation of algorithm has been presented [1]. Studies were done on each operator to understand finally the Substitution Box and Modulo 2^b adders costs for more hardware complexity.

The fundamental addition mod 2^b $\boxplus$ is narrated [1] as follows:

$$\boxplus : \mathbb{Z}_{2^b}^{n_b} \times \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b} : x, y \to z = x \boxplus y \Leftrightarrow$$
$$z_i = x_i \boxplus y_i, \quad 0 \le i \le n_b - 1$$

In this Section, we give a complete description of the algorithm, starting with the important parameters, and then emphasizing its basic operation. Afterwards follows the round and key round description of SEA is presented [1].

2.1.2 Encryption/Decryption and Key Generation

The encryption round F_E, decryption round F_D and key scheduling round F_K are defined as:

Encryption Round $\mathbf{F_E}$:

$$[L_{i+1}, R_{i+1}] = F_E(L_i, R_i, K_i) \Leftrightarrow R_{i+1} = R(L_i) \oplus r\big(S(R_i \boxplus K_i)\big), \quad L_{i+1} = R_i$$

Decryption Round $\mathbf{F_D}$:

$$[L_{i+1}, R_{i+1}] = F_D(L_i, R_i, K_i) \Leftrightarrow R_{i+1} = R^{-1}\Big(L_i \oplus r\big(S(R_i \boxplus K_i)\big)\Big), \quad L_{i+1} = R_i$$

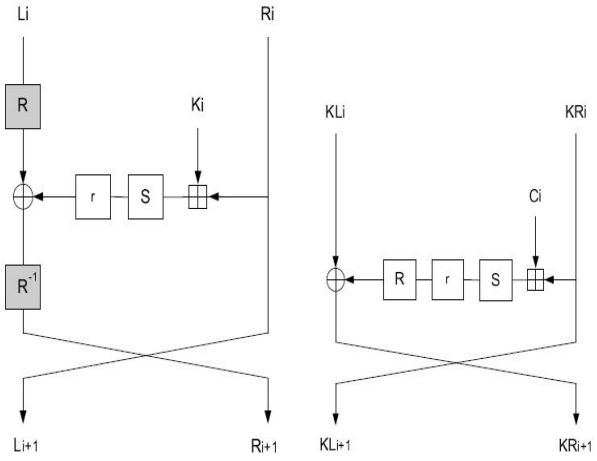

Figure 2.1. Encrypt /decrypt round and key round [1]

Key Scheduling Round $\mathbf{F_K}$:

$$[KL_{i+1}, KR_{i+1}] = F_K(KL_i, KR_i, C_i) \Leftrightarrow KR_{i+1} = KL_i \oplus R\Big(r\big(S(KR_i \boxplus C_i)\big)\Big), \quad KL_{i+1} = KR_i$$

2.1.3 Cipher Process

Enciphering is done iteratively by a number of rounds denoted as 'n_r'. This procedure is illustrated by the pseudo code [1]. P is the plain-text representing the original data, K is the key and C is the cipher-text. A parametric bit size known as 'n' is used in

P, C and K. The operations within the cipher are performed considering parametric words of b-bit. In this pseudo code, the '&' relates to the concatenation operator, KR $n_r/2$ is taken before the switch and C(i) is a n_b -word vector of which all the words have value 0 excepted the LSB that equals i. Decryption is exactly the same, using the decrypt round F_D. Since n_r is odd, for key scheduling and encryption the value of n_r must rounded up or down [2].

2.1.4 Advised Number of rounds

Recommended Number of rounds n_r is calculated which gives high resistant to various known attacks like linear and differential attacks. The value of n_r must be always odd, if not 1 must be added to make it odd [1,2].

3. HARDWARE REALIZATION

3.1 Explanation

The first analysis step to the hardware implementation of SEA proposes to take a look at a simple and direct implementation of the algorithm on an FPGA platform, achieving one round per clock cycle and represents as the loop implementation. Components of the cipher that consume resources are the S-boxes and the mod 2^b adder; the Word Rotate and Bit Rotate operations are simply implemented by swapping wires [1]. Based on the specifications, the key scheduling routines uses two multiplexers permitting to switch the right and left part of the round key at half the way of executing the algorithm, using the suitable command signal Swap. The multiplexer is managed by Switch that offers the round function with the right part of the round key for the first half of the execution and transmits its left part instead after the switch. The Generic Loop Architecture is simple and only changes in the location of the R and R^{-1} Block. The number of rounds n_r is an elective input that can be involuntarily derived from n and b. In this paper, we mainly focus on modified SEA with different architectures of light weight Modular adders [13] in consideration of efficient area and low power optimization at the synthesizable VHDL design level. Each architecture of modular adders is implemented individually in VHDL and then combined with other components to build a whole SEA [1, 2].

3.2 Realization of efficient Modular adders

Our proposed modification of SEA based on various Modular adder designs [13] are constructed using basic components like carry propagate adders, 2×1 multiplexers, OR gate.
Let x and y are the two numbers and assume that they belong to a set {0, 1, 2, … m-1}. Then the modulo m addition is defined as:

$$(x + y) \bmod m = x + y \quad \text{if } x + y < m$$
$$= x + y - m \quad \text{if } x + y \geq m \qquad (3.1)$$

Though it appears that the above mentioned equation can be realized using simple arithmetic operators, it involves with the complexity in the implementation procedures. It requires a suitable algorithm to reduce the cost of implementation and area. Powerful hardware operators are essential for this purpose.

Let $b = [\log_2 m]+1$ be the number of bits. It is used in encoding modulo m arithmetic operators by considering the inputs and outputs. This can be done by using any of the three operating methods based on table, adder or hybrid.

3.2.1 Adder-Based Operators

Figure 3.1 illustrates the implementation aspects of the operation shown in Equation (3.1). The proof of the evaluation of these procedures is given [13, 14, 15]. This architecture is very much suitable for FPGAs as it uses only multiplexer and two carry propagate adders.

The architecture of implemented algorithm[13] is portrayed in Figure 3.1. This modulo addition algorithm is used to modify SEA [1] which reduces the overall complexity.

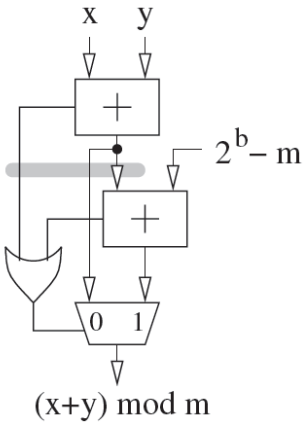

Figure 3.1

3.2.1 Modulo ($2^b \pm 1$) Addition

A few improvising of the adder-based operator illustrated earlier are given for specific values of m i.e., 2^b. For example, one's complement addition or modulo ($2^b - 1$) addition [12] is defined by

$$(x + y) \bmod (2^b - 1)$$
$$= (x + y + 1) \bmod 2^b \quad , \text{if } x + y + 1 \geq 2^b$$
$$= x + y , \quad \text{if } x + y + 1 < 2^b \qquad (3.2)$$

Figure 3.2 [13] shows the architecture of the equivalent hardware operator. Because of the condition $x + y + 1 \geq 2^b$, we execute two additions in parallel and select the better result with a multiplexer by considering that zero has a double representation in one's complement, namely "0 . . . 0" and "1 . . . 1" (i.e. 0 is congruent to $2^b - 1$ (modulo $2^b - 1$)). In case of second encoding of zero accommodated by the path of computation, Equation (3.2) can be rewritten as in (3.3) [13].

$$(x + y) \bmod (2^b - 1)$$
$$= (x + y + 1) \bmod 2^b \quad \text{if } x + y \geq 2^b$$
$$= x + y \quad \text{if } x + y < 2^b \qquad (3.3)$$

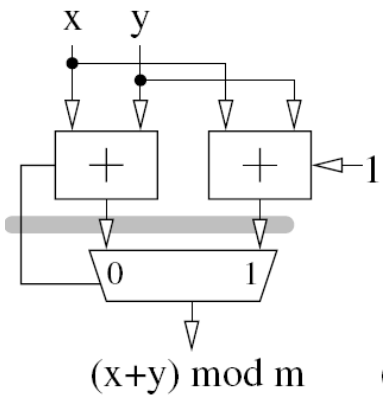

Figure 3.2

The carry-out (c_{out}) of the addition of x + y specifies whether the increment is to be done. Figure 3.2 depicts the advantage of evaluating the addition of x + y and x + y = 1 in parallel based on the result of C_{out}. Figure 3.3 [13] illustrates an another architecture in which the sum x + y is added with C_{out}

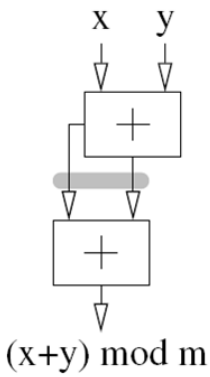

Figure 3.3

4. IMPLEMENTATION RESULTS

We have used synthesizable VHDL code for designing and implementing each circuit as illustrated in Figure 3.1, 3.2 and 3.3. Implementation results were extracted with the Xilinx ISE 9.2i tool on a device XC4VLX25, VIRTEX-4 platform with speed grade-12 and XPower Analyzer tool was used to analyze the power consumption of the implementation. We have then performed a sequence of trials with the tool in order to estimate the memory constraints in terms of slices/area and power consumption both static and dynamic power of each modular adder according to m. Our main objective was to weigh against three architectures of a modular adders Mod_adder1 (Figure 3.1), Mod_adder2 (Figure 3.2) and Mod_adder3 (Figure 3.3).

The operators depicted in Figure 3.2 do not appreciably progress as in adder-based operator in Figure 3.1. The modulo ($2^b - 1$) adder described in [13] is shown in Figure 3.3 is compact as it does not use any multiplexer for its operation. The area and power consumption of the three architectures are compared with each other. This points out that a determination of the double representation of zero can direct the hardware implementation of an arithmetic operator in superior way.

The implementation was done for variants bit data (n) and a processor word size (b). We achieved reduction in number of slices in Figure 4.1, area in terms of Gate Count in Figure 4.2, Dynamic power consumption in Figure 4.3, Static power consumption in Figure 4.4 and Total power consumption in Figure 4.5. Consequently, our implementation of SEA exhibited a extremely small area consumption that comes at the cost of increased throughput and decreased power consumption. Therefore, it can be considered as the attractive substitute for constrained devices.

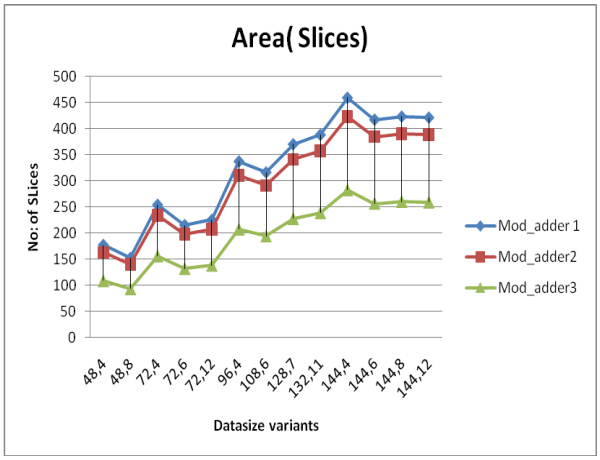

Figure 4.1

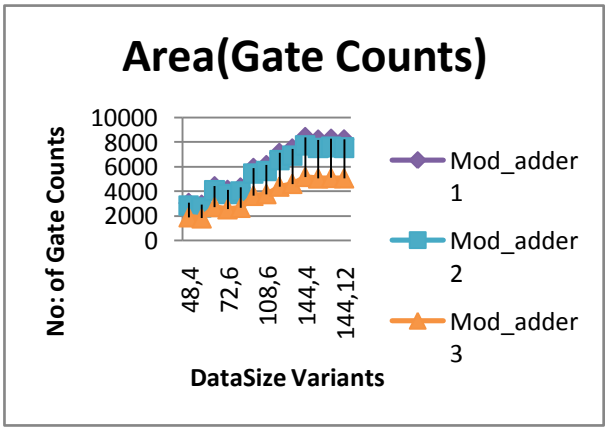

Figure 4.2

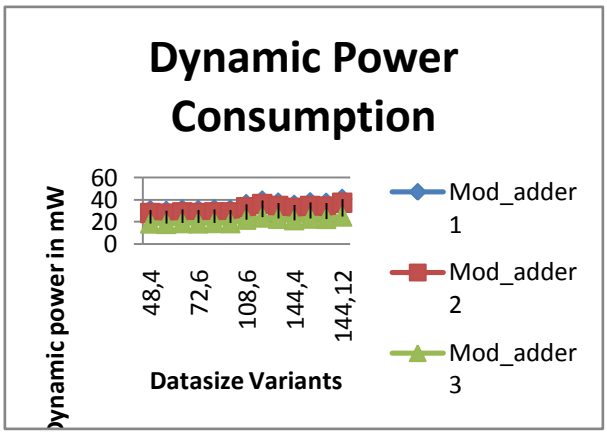

Figure 4.3

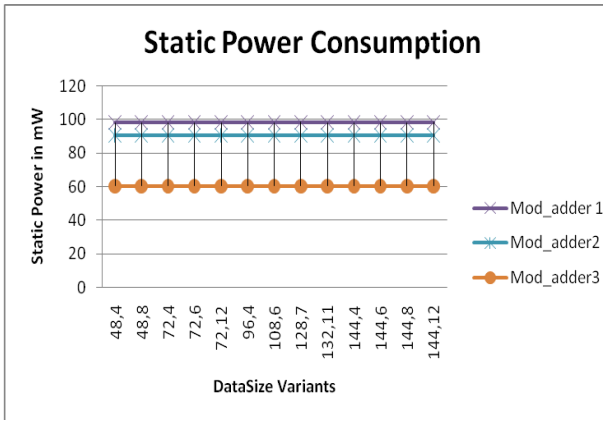

Figure 4.4

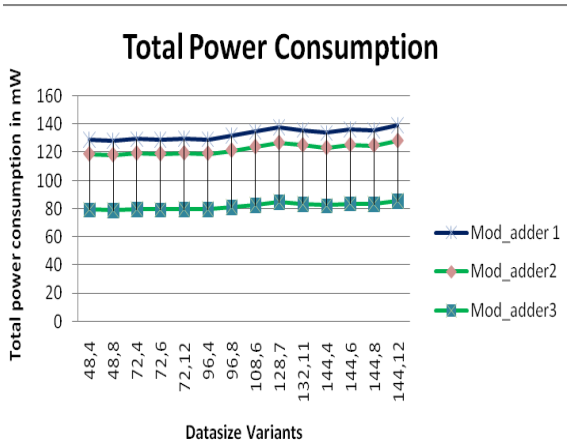

Figure 4.5

5. CONCLUSION

The flexible cipher SEA is primarily designed for efficient implementation in software. Later the design proved its significance in hardware realization as this implementation lead to better solutions compared to software realization. Thorough investigation of SEA and efficent modular adders in various papers guided us to efficiently implement the modified version of SEA and to be more specific, the optimized modulo 2^b addition in hardware. It also gives better performance in terms of area, power and delay. It has promising merits like simplicity, scalability and combination of good encrypting and decrypting routines. Hence, Our modified SEA shows better fitness performance in resource constrained devices.

6. REFERENCES

[1] F.Mace, F.X Standaert, J J Quisquater "FPGA implementation(s) of a Scalable Encryption algorithm" IEEE Transactions on VLSI Systems, Vol.16, 2008, pp.212-216.

[2] Francois-Xavier Standaert, Gilles Piret, Neil Gershenfeld, Jean-Jacques Quisquater "SEA a Scalable Encryption Algorithm for Small Embedded Applications" in Proc.CARDIS, 2006, pp 222-236.

[3] K.J.Jegadish Kumar, S.Salivahanan, K.Chenna Kesava Reddy, "Implementation of Low power Scalable Encryption algorithm",International Journal of Computer applications, Volume-11, Dec 2010, pp.14-18.

[4] Advanced Encryption Standard, FIPS PUB 197, Nov. 2001.

[5] N. Pramstaller and J. Wolkerstorfer, "A universal and efficient AES co-processor for field programmable logic arrays," in Proc. FPL, 2004, pp. 565–574.

[6] Francisco Rodriguez-Henriquez, N.A. Saqib,A. Diaz-Perez,Cetin Kaya K09, "Cryptographic Algorithms on Reconfigurable Hardware", Springer Series on Signals and Communication Technology, 2006.

[7] J. Daemen and V. Rijmen, The Design of Rijndael. New York: Springer-Verlag, 2001.

[8] Data Encryption Standard, FIPS PUB 46-3, Oct. 1999.

[9] D.J. Wheeler, R. Needham, TEA, a Tiny Encryption Algorithm, proceedings of FSE 1994, Lecture Notes in Computer Science, vol 1008, Springer-Verlag, Leuven, Belgium, December 1994, pp 363-366.

[10] G. Yuval, "Reinventing the travois: Encryption/MAC in 30 ROM bytes," in Proc. Fast Softw. Encryption (FSE), 1997, pp. 205–209.

[11] J.P. Kaps, "Chai-Tea, Cryptographic Hardware Implementations of XTEA, 9th International Conference on Cryptology in India – INDOCRYPT 2008, LNCS 5356, pp. 363-375, 2008.

[12] D. Hong et al., "HIGHT: A New Block Cipher Suitable for Low-Resource Device," Proceedings of CHES 2006, Lecture Notes in Computer Science, Vol.4249, pp. 46-59, Yokohama, Japan, October 2006.

[13] Beuchat,J.-L.; Lab. De l'Infonnatique du Parallelisme, " Some Modular adders and multipliers for Field programmable Gate arrays", in Proc. Parallel and Distributed processing symposium 2003.

[14] J.-L. Beuchat. "Modular Multiplication for FPGA Implementation of the IDEA Block Cipher", Technical Report 2002-32, Laboratoire de l'Informatique du Parall´elisme, Ecole NormaleSup´erieure de Lyon, 46 All´ee d'Italie, 69364 LyonCedex 07, Sept. 2002.

[15] J.-L. Beuchat and A. Tisserand. "Small Multiplier-based Multiplication and Division Operators for Virtex-II Devices", number 2438 in Lecture Notes in Computer Science, Springer, 2002, pp 513–522.