Indian Journal of Science and Technology, *Vol 9(29), DOI: 10.17485/ijst/2016/v9i29/90845, August 2016* *ISSN (Print) : 0974-6846* *ISSN (Online) : 0974-5645*

Implementation of High Throughput Extended Tiny Encryption Algorithm Block Cipher in Field Programmable Gate Array

S. Sopna[1*] and R. Muthaiah[2]

[1]School of Computing, SASTRA University, Thirumalaisamudram, Thanjavur – 613401, Tamil Nadu, India; sopnaselvarajan@gmail.com

[2]Information and Communication Technology, School of Computing, SASTRA University, Thirumalaisamudram, Thanjavur – 613401, Tamil Nadu, India; sjamuthaiah@core.sastra.edu

Abstract

The main objective of the paper is to improve the throughput and increase the speed of the XTEA (Extended Tiny Encryption Algorithm) which is one of the cryptographic algorithms. The technique used to improve the throughput is parallel processing. In previous, they have used pipelining which increases the overhead that is not desirable. The error detection scheme employed in this paper is RERO (Recomputing with Rotated Operand). The previous error detection REPO (Recomputing with Permuted Operands) occupies more memory than RERO. The embedded systems have been developing with many sensitive nodes like Nano-sensors, Radio Frequency Identification tags, etc. As these systems have many constraints, the required security is given by the light weight block ciphers like PRESENT, XTEA and SIMON. These block ciphers are more suitable for these embedded systems when compared to Advanced Encryption Standard (AES). Providing security alone does not give assurance for their reliability while these architectures are liable to malicious and natural faults. Due to hardware failures, various types of faults occur while implementing in hardware. To overcome this, there are many schemes for error detection that can be applied to these ciphers. One of the error detection schemes is RERO (Recomputing with Rotated Operands). Through RERO, error coverage is achieved high. Parallel processing is applied to the XTEA algorithm to increase the throughput. The two halves that are used in the XTEA algorithm are executed at the same time through the parallel processing. Finally, this algorithm is implemented in the Altera FPGA (Field Programmable Gate Array). The throughput is increased by 60% when compared to previous systems that use sub pipelining. This type of structure finds its application in low resource embedded systems having sensitive nodes like RFID tags and Nano sensors.

Keywords: Cryptography, Error Detection, FPGA, RERO, Throughput, XTEA

1. Introduction

The deployment of the small computing devices like RFID (Radio Frequency Identification) devices and the wireless devices with Nano sensor nodes is the elucidate trend of the IT landscape of this century. These applications have sensitivity which makes the light weight cryptography to give more confidentiality. A new form of algorithm, XTEA which is a block cipher with light weight was developed. As this algorithm is remarkable for simplicity, it is appropriate for implementing in hardware. It is extensively used to provide the light weight security.

In Very Large Scale Integration (VLSI), the natural faults are usually caused by the hardware failures. In the embedded systems and also in the cryptographic hardware, the unfavourable effects of these faults are magnified taking into account the sensitivity of the architecture and also the fault attacks. The fault attacks are referred to as the attacks of the side channel. For other cryptographic algorithms like AES, the CED (Concurrent Error Detection) structures[1–5] are achieved for the architectures that are reliable[6], through various analysis.

Many analyses have been carried out regarding the light weight block ciphers like SIMON[7]. Among those,

** Author for correspondence*

XTEA is the most efficient. It is the fastest block cipher in existence. It is used in many cryptographic applications which are used in real life. It has very small code size. It uses the simple operations like XOR, shift functions and addition. It provides confidentiality for the nodes that have minimum power and memory.

Using this XTEA algorithm, the reliability of the architecture and the fault resilience are achieved. One of the error detection schemes is RERO. It is applied to this algorithm to achieve high error coverage. By achieving high error coverage, the architecture is reliably used in the embedded systems having the sensitive nodes. It is implemented in FPGA[8].

2. XTEA Algorithm

The Extended Tiny Encryption Algorithm (XTEA) which is one of the light weight block cipher accepts an input data of 64 bit block and the size of the key is 128 bits. The Feistel network accepts the input data block. The data block of 64 bits is separated into two cycles of 32 bit each. These two halves are denoted as X and Y. This network has cycles which are denoted as N. In this case, N = 32.

The algorithm which is given in Figure 1 describes the flow of converting the plain text to the cipher text. All the additions and the subtractions in the XTEA are modulo. “<<4” denotes the logical shift left by 4 bits. “>>5” denotes the logical shift by 5 bits. “^” denotes the XOR operation in this algorithm. In the algorithm, the permutation is the first part and the sub key generation is the second part. The size of the key which is 128 bit is subdivided into four blocks of 32 each. The *[sum]* is the function which is used to choose one of the four sub key blocks. It chooses depending on the 0th and 1st bits i.e. 11th and 12th bits of the 2nd half cycle of the sum. The result of the permutation and the result of the sub key generation are XORed. When encrypting, the result of the XOR function is given to the X and Y through addition whereas when decrypting, the result is applied to the X and Y through subtraction. Each cycle has two cycles. XTEA has 32 rounds and thus has 64 cycles. At the end of the first half cycle, the sum is calculated and the sum increases by delta which is a constant. The value of delta is equal to the integral part of the $\left(\sqrt{5}-1\right)\times 2^{31}$ that is given in the hexadecimal. The simulation result of XTEA is given in Figure 2.

Algorithm: XTEA encryption algorithm

```
Inputs: v[0]-v[1] = 64 bit data, key[0]- key[3] =128 bit
Encryption:
Initialization:
v0 = v[0]
v1 = v[1]
delta = 0x9E3779B9
sum = 0
n – Number of rounds.
for i = 0 to n-1 do
v0 = v0 + ((( v1<<4) ^ (v1 >> 5)) + v1) ^ (sum + key
[sum&3]).
sum = sum + delta
v1 = v1 + ((( v0 << 4 ) ^ (v0 >> 5)) + v0) ^ (sum + key
[sum >> 11&3]).
end for.
```

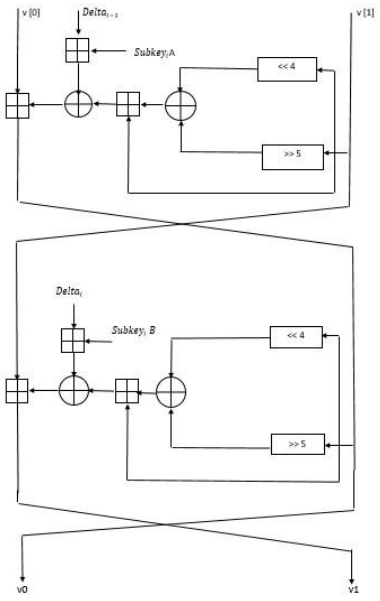
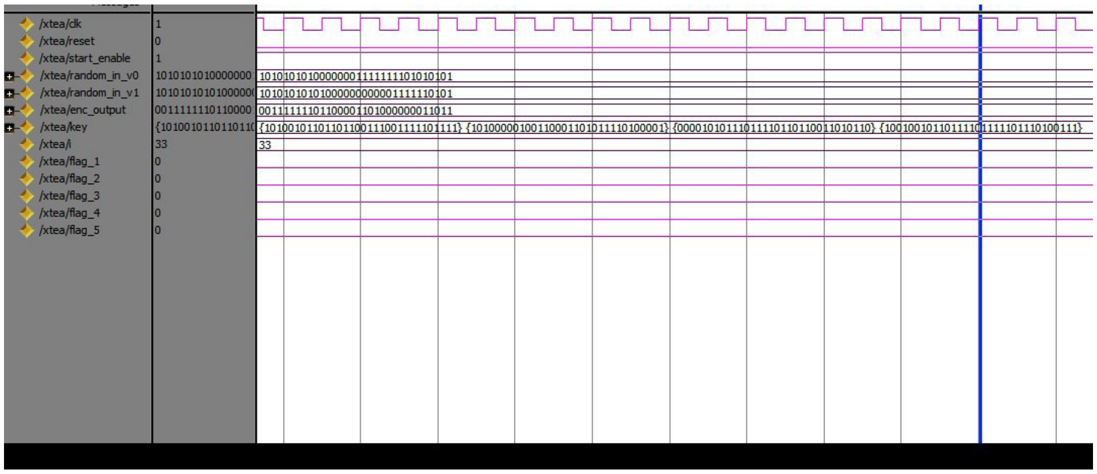


Figure 1. XTEA encryption algorithm..

3. Error Detection Schemes

There are two error detection schemes for XTEA algorithm. They are recomputing with Rotated Operands and signature based diagnosis. These methods can be also applicable to other light weight block ciphers like SIMON and other hash functions but with small modifications. Only the error detection schemes for encryption are proposed in this paper.



```
/xtea/clk             1
/xtea/reset           0
/xtea/start_enable    1
/xtea/random_in_v0    101010101000000001   1010101010000000111111110101 0101
/xtea/random_in_v1    10101010101000000(   1010101010100000000000111111 0101
/xtea/enc_output      001111111101100001   0011111111011000011010000000 11011
/xtea/key             {1010010110110110   {10100101101101100111001111101111} {10100000100110001101011110100001} {00001010111011110110110011010110} {10010010110111101111101110100111}
/xtea/i               33                   33
/xtea/flag_1          0
/xtea/flag_2          0
/xtea/flag_3          0
/xtea/flag_4          0
/xtea/flag_5          0
```

Figure 2. Simulation result of XTEA algorithm.

3.1 Signature based Diagnosis

The parity prediction function is the straight forward in this scheme. $\hat{P}$ is the predicted parities. This is used for the shift and the XOR operations. Let A and B be the two inputs, a_i and b_i be the input bits, P_A and P_B be the actual parities for the inputs A and B, for the XOR operation,

$$\hat{P}_s = \sum_{i=0}^{n-1}(a_i \wedge b_i) = \sum_{i=0}^{n-1} a_i \wedge \sum_{i=0}^{n-1} b_i = P_A \wedge P_B$$

In this diagnosis, the carry generation block duplication is avoided by the following solutions.

- To assure the less hardware cost, the similar carry generation logic which is used in the ripple carry adder is used.
- To assure the high speed, it better uses the input carry from the usual generation of carry logic than using the slice of the check carries of the previous one.

3.2 Recomputing with Rotated Operand (RERO)

Normally, the error detection scheme that is based on the time redundancy has the inability to detect the permanent faults. The performance is degraded in the time redundancy techniques. RERO detects both the transient faults and the permanent faults.

RERO is one of the error detection schemes which represents the concurrent detection. The limitations of the previous error detection which is termed as RESO are overcome by the RERO. The main idea of the RERO is the re-computation of the rotated operands by k-bits. Let it rotates the operand by k-bits. All the processes in the given operation are executed twice. Once they are executed by using the usual operands and for the second time they are processed using the rotated operands. For example, in ALU, RERO detects the (k-1) errors that are arithmetic and (k mod n) logic errors, where n is the length of the ALU. The flow diagram of RERO is given in Figure 3.

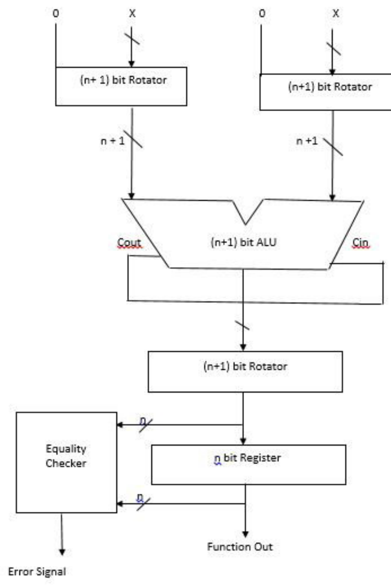


Figure 3. RERO – Conceptual diagram.

There are three rotators in this concurrent detection of errors. In the first step, all the three rotators can't perform the rotating operation as the process is done using the normal operands. In the next step, the two left rotators perform its operation by shifting the data by k bits. Then the right rotator does its part by k bits.

Among the fault attacks, the most usual one is transient faults. The permanent faults outbreak the logical gates in the circuits of the embedded hardware and in the cryptographic hardware. Let φ and $φ^{-1}$ are the cyclic shifts or the n bit rotations of the LSB (Least Significant Bits) and the MSB (Most Significant Bits) of the operand respectively.

g - Arithmetic function
Л - Input to g
g(Л) - Output from g
$g(Л) = φ^{-1}(g(φ(Л)))$

The RERO method is applied such that the storage of the first run i.e. first computation result is done followed by the storage of the second computation result. Both are compared. If both the results are similar, it shows that there is no error whereas when both the results are different, it specifies that there are some errors. This error is indicated by the error indication flag.

It is possible to improve the throughput and the efficiency through the sub-pipelining so that the frequency is also increased. This pipelining uses one register to sub-pipeline the given structure. $Л_1$ and $Л_2$ are the 2 splits of the stages of the pipeline. The first step is to apply the input to the architecture. This is performed in 1st sequence. As soon as the 2nd quasi of this circuit, that is $Л_2$, which executes the 1st input, rotated product of 1st input has been given to the 1st quasi of this circuit that is $Л_1$. Process continues until the last result of the rotated product of the input is obtained. Though the sub-pipelining increases the throughput, the hardware overhead is increased. So we are going for parallel processing.

4. Fault Model

The fault model is used to estimate the error finding ability of the hardware architecture. The single stuck at and multi stuck at faults are measured. The stuck at faults can be of permanent or transient faults. They cover both the natural and malicious faults. The single stuck at faults are the faults that model the natural failures in the structures. The natural failures are the most suitable cases for the attackers. They can be of single event upsets. These injection of single stuck at fault is difficult to obtain information for an attacker because of many technological constraints such as larger bus lines.

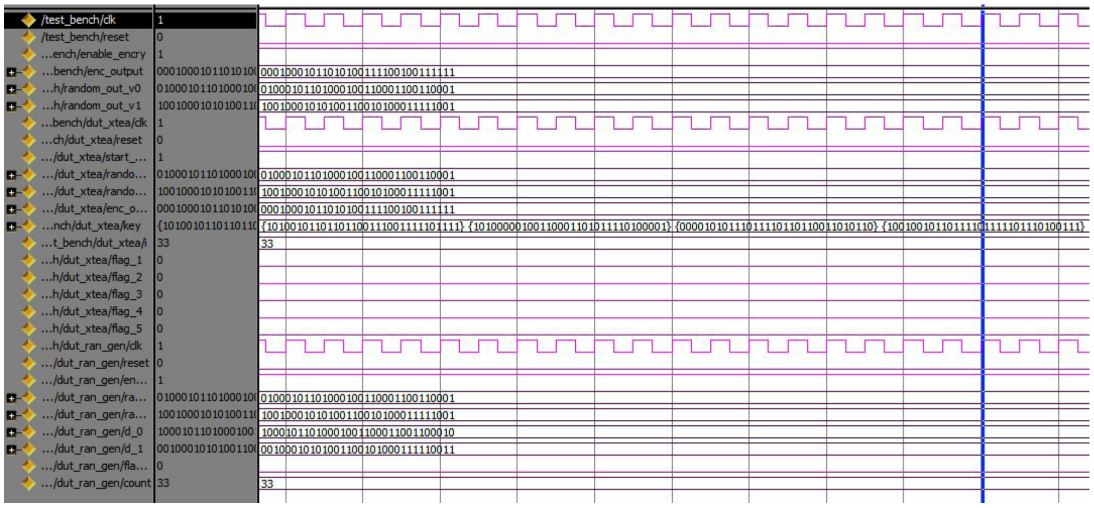
Most of the internal faults are characterized by transient faults. These are localised faults. If the faults affect one parity, then it has the detection rate is 50%. If the faults are randomly distributed, then the detection rate is very high. Each parity detects 50% of detection rate and not more than that. The randomly distributed faults, which can be of transient fault or permanent fault, detect with higher rate if it is a multiple fault model. The transient faults are detected with the high ratio with the RERO. Using the LFSR (Linear Feedback Shift Register), the error coverage is assessed. To inject some multiple errors randomly, the Fibonacci LFSR is used. In this type of implementation, the location of the errors, the type of the errors and the number of errors are all chosen randomly.

The flags are used to indicate the errors and are frequently monitored. So we can easily detect and count such errors. If faults are injected in the parity circuitry by the attacker, the single or multiple stuck-at – fault that is injected are detected in the parity prediction circuit if it is not masked. The parity prediction block becomes ineffective because of the dual injection in the prediction and the original circuits for the multiple stuck at faults. Thus the multiple stuck at faults are not effective as they make the parity block ineffective. So they are not so preferable because they can't indicate main effects on the error analysis finally. The RERO is not prone for those injections.

5. Parallel Processing

The parallel processing is one among the techniques that can be used to perform different tasks at the same time. Every single task is divided into sub-tasks that is done by various functional units. The two main advantages of this technique are that it increases the speed of the sample which in turn increases the throughput and it consumes less power.

To increase the throughput of the structure, the encryption of the XTEA is done using the parallel processing. It is done by the process such that the two halves of the XTEA algorithm are executed parallel instead of sequentially. The XTEA has 64 Fiestel rounds i.e. it has 32 cycles. So each cycle has two rounds. In each cycle, the two rounds are processed simultaneously via parallel processing rather than sequential process. The simulation results of the proposed structure is given in Figure 4.



```
/test_bench/clk                1
/test_bench/reset              0
...ench/enable_encry           1
...bench/enc_output            00010001011010100  00010001011010100111100100111111
...h/random_out_v0             01000101101000100  01000101101000100110001100110001
...h/random_out_v1             10010001010100110  10010001010100110010100011111001
...bench/dut_xtea/clk          1
...ch/dut_xtea/reset           0
.../dut_xtea/start_...         1
.../dut_xtea/rando...          01000101101000100  01000101101000100110001100110001
.../dut_xtea/rando...          10010001010100110  10010001010100110010100011111001
.../dut_xtea/enc_o...          00010001011010100  00010001011010100111100100111111
...nch/dut_xtea/key            {1010010110110110  {10100101101101100111001111101111} {10100000100110001101011110100001} {00001010111011110110110011010110} {10010010110111101111101110100111}
...t_bench/dut_xtea/i          33                 33
...h/dut_xtea/flag_1           0
...h/dut_xtea/flag_2           0
...h/dut_xtea/flag_3           0
...h/dut_xtea/flag_4           0
...h/dut_xtea/flag_5           0
...h/dut_ran_gen/clk           1
.../dut_ran_gen/reset          0
.../dut_ran_gen/en...          1
.../dut_ran_gen/ra...          01000101101000100  01000101101000100110001100110001
.../dut_ran_gen/ra...          10010001010100110  10010001010100110010100011111001
.../dut_ran_gen/d_0            10001011010001001  10001011010001001100011001100010
.../dut_ran_gen/d_1            00100010101001100  00100010101001100101000111110011
.../dut_ran_gen/fla...         0
.../dut_ran_gen/count          33                 33
```

Figure 4. Simulation result of parallel processing.

Although the pipelining increases the throughput, it has also many disadvantages. It increases the latency as the registers are added. It also increases the hardware overhead. So we are going for parallel processing to increase the throughput.

6. Experimental Results

The analysis has been carried out in the Altera FPGA. The VHDL, which is one of the hardware description language is used as the programming language. The operating frequency for the given structure is 85 MHz. The minimum clock period utilized is given by 11.655 ns.

The performance comparison of both the techniques i.e. pipelining and parallel processing are given in the Table 1. The throughput is improved in parallel processing by 60% in comparison with pipelining. It is estimated in Mbps. Therefore it achieves high speed.

Table 1. Performance comparison of pipelining and parallel processing

Technique	Throughput (Mbps)
Pipelining	259 Mbps
Parallel Processing	648 Mbps

The consumption of power is reduced through the parallel processing. It reduces by 10 mW. The power is estimated in mW. It is 91 mW in case of pipelining whereas it is 81 mW when we use parallel processing. The comparison of the power consumption is given in the Table 2.

Table 2. Comparison of power consumption

Technique	Power Consumption (mW)
Pipelining	91 mW
Parallel Processing	81 mW

7. Conclusion

In this paper, the error detection scheme RERO is applied to the XTEA algorithm and the throughput is increased through the parallel processing. The hardware implementation of the structure is done in the Altera FPGA. The hardware overhead is reduced since we are not using pipelining. The power is reduced in the proposed structure. The proposed scheme finds its applications in the embedded systems having resource constraints and sensitive nodes.

8. References

1. Yen CH, Wu BF. Simple error detection methods for hardware implementation of Advanced Encryption Standard. IEEE Transactions Comput. 2006 Jun; 55(6):720–31.
2. Maistri P, Leveugle R. Double-data-rate computation as a countermeasure against fault analysis. IEEE Trans Comput. 2008 Nov; 57(11):1528–39.
3. Mozaffari Kermani M, Reyhani-Masoleh A. Concurrent structure independent fault detection schemes for the Advanced Encryption Standard. IEEE Trans Comput. 2010 May; 59(5):608–22.
4. Mozaffari Kermani M, Reyhani-Masoleh A. A lightweight high performance fault detection scheme for the Advanced

Encryption Standard using composite fields. IEEE Trans Very-Large Scale Integr (VLSI) Sys. 2011 Jan; 19(1):85–91.

5. Xiaofei G, Karri R. Recomputing with permuted operands: A concurrent error detection approach. IEEE Trans Computer-Aided Design. 2013 Oct; 32(10):1595–608.
6. Mozaffari Kermani M, Azarderakhsh R. Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA. IEEE Trans Ind Electron. 2013 Dec; 60(12):5925–32.
7. Aysu A, Gulcan E, Schaumont P. SIMON says: Break area records of block ciphers on FPGAs. IEEE Embedded Systems Letters. 2014 Jan; 6(2):37–40.
8. Salim PTT, Vigneswaran T. FPGA implementation of hiding information using cryptography. Indian Journal of Science and Technology. 2015 Aug; 8(19):1–7.