Volume No:3, Issue No:12 (May-2018) ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

Scalable Encryption Algorithm Design & Implementation using Flow Chart Approach

M. Bhagavanth
Department of Computer Science and Engineering,
CVR College of Engineering,
Ibrahimpatnam (M), Rangareddy (D), Telangana 501510, India.

ABSTRACT
The (putting into) use of (turning messages into secret code)/decryption set of computer instructions is the most extremely important part of the secure communication. In now existing (turning messages into secret code) sets of computer instructions there is a trade-off between putting into use cost and resulting performances. SEA is an (able to be made bigger or smaller) (turning messages into secret code) set of computer instructions targeted for small embedded computer programs. It was, at first, designed for software putting into uses in controllers, smart cards or processors. In this letter, we (ask lots of questions about/try to find the truth about) its performances in recent FPGA devices. For this purpose, a loop (related to the beautiful design and construction of buildings, etc.) of the block code/puzzle is presented.

Beyond its low cost performances, a significant advantage of the proposed (related to the beautiful design and construction of buildings, etc.) is its full flexibility for any limit/guideline of the (able to be made bigger or smaller) (turning messages into secret code) set of computer instructions, taking advantage of plain and common thing/not a brand-name drug Verilog HDL coding. The letter also carefully describes the putting into use details allowing us to keep small area needed things. Finally, a (serving to compare two or more things) performance discussion of SEA with the Advanced (turning messages into secret code) Standard Rijndael and ICEBERG(a code/puzzle purposed for (producing a lot with very little waste) FPGA putting into uses) is proposed. It illustrates the interest of (raised, flat supporting surface)/context-oriented block calculates/codes design and, as far as SEA is concerned, its low area needed things and reasonable (wasting very little while working or producing something).

I.INTRODUCTION
Scalable encryption algorithm is targeted for small-embedded application with limited resources.SEA is a parametric block cipher for resource constrained systems (e.g. sensor networks, RFIDs) that has been introduced in [1]. It was initially designed as a low-cost encryption/authentication routine (i.e. with small code size and memory) targeted for processors with a limited instruction set(i.e. AND, OR, XOR gates, word rotation and modular addition).Additionally and contrary to most recent block ciphers (e.g.the DES [2] and AES Rijndael [3], [4]), the algorithm takes the plaintext, key and the bus sizes as parameters and therefore can be straightforwardly adapted to various implementation contexts and/or security requirements. Compared to older solutions for low cost encryption like TEA (Tiny Encryption Algorithm) [5] or Yuval's proposal [6], SEA also benefits from a stronger security analysis, derived from recent advances in block cipher design/cryptanalysis. In practice, SEA has been proven to be an efficient solution for embedded software applications using microcontrollers, but its hardware performances have not yet been investigated.

Cite this article as: M. Bhagavanth, "Scalable Encryption Algorithm Design & Implementation using Flow Chart Approach", International Journal of Research in Advanced Computer Science Engineering, Volume 3, Issue 12, 2018, Page 6-14.

Volume No:3, Issue No:12 (May-2018) ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

Consequently, and as a first step towards hardware performance analysis, this letter explores the features of a low cost FPGA encryption/decryption core for SEA. In addition to the performance evaluation, we show that the algorithm's scalability can be turned into a fully generic Verilog HDL design, so that any text, key and bus size can be straightforwardly re-implemented without any modification of the hardware description language, with standard synthesis and implementation tools. In the rest of the letter, we first provide a brief description of the algorithm specifications. Then we describe the details of our generic loop architecture and its implementation results. Finally, we discuss some illustrative comparisons of the hardware performances of SEA, the AES Rijndael and ICEBERG (a cipher purposed for efficient FPGA implementations) with respect to their design approach (e.g. flexible vs. platform/context-oriented).

II. ALGORITHM DESCRIPTION

Parameters and definitions

SEA n,b operates on various text, key and word sizes. It is based on a Feistel structure with a variable number of rounds, and is defined with respect to the following parameters:

- n: plaintext size, key size.
- b: processor (or word) size.
- n_b = n/2b : number of words per Feistel branch.
- n_r: number of block cipher rounds.

As only constraint, it is required that n is a multiple of 6b (see[1] for the details). For example, using an 8-bit processor, wecan derive a 96-bit block ciphers, denoted as SEA96,8.

Let x be a n/2 -bit vector. We consider two representations:

- Bit representation: $x_b = x(n/2- 1) \ldots x(2)\ x(1)\ x(0)$.
- Word representation: $x_w = x_{nb-1}\ x_{nb-2} \ldots x2\ x1\ x0$.

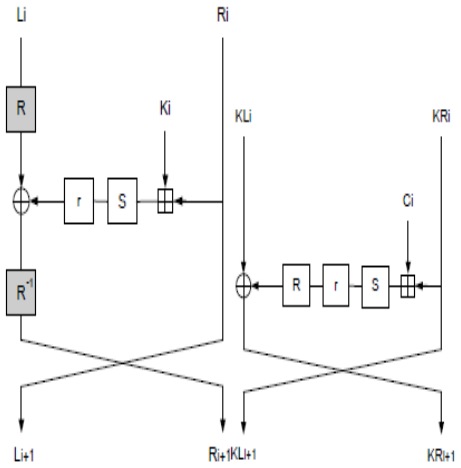


Fig. 1. Encrypt/decrypt round and key round.

Basic operations

Due to its simplicity constraints, SEAn,b is based on alimited number of elementary operations (selected for theiravailability in any processing device) denoted as follows:

(1) bitwise XOR $\oplus$, (2) addition mod 2b $\boxplus$, (3) a 3-bitsubstitution box S := {0, 5, 6, 7, 4, 3, 1, 2} that can be appliedbitwise to any set of 3-bit words for efficiency purposes. Inaddition, we use the following rotation operations:(4) Word rotation R, defined on nb-word vectors:

$$R : \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b} : x \to y = R(x) \Leftrightarrow \quad y_{i+1} = x_i, 0 \le i \le n_b - 2, \quad y_0 = x_{n_b-1}$$

(5) Bit rotation r, defined on n_b-word vectors:

$$r : \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b} : x \to y = r(x) \Leftrightarrow \quad y_{3i} = x_{3i} \ggg 1, \quad y_{3i+1} = x_{3i+1}, \quad y_{3i+2} = x_{3i+2} \lll 1,$$

where 0 ≤ i ≤nb/3 – 1and >>>and <<<respectively representthe cyclic right and left shifts inside a word.

Volume No:3, Issue No:12 (May-2018) ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

The round and key round

Based on the previous definitions, the encrypt round FE,decrypt round FD and key round FK are pictured in Figure 1and defined as:

$$[L_{i+1}, R_{i+1}] = F_E(L_i, R_i, K_i) \Leftrightarrow \begin{aligned} R_{i+1} &= R(L_i) \oplus r\left(S(R_i \boxplus K_i)\right) \\ L_{i+1} &= R_i \end{aligned}$$

$$[L_{i+1}, R_{i+1}] = F_D(L_i, R_i, K_i) \Leftrightarrow \begin{aligned} R_{i+1} &= R^{-1}\left(L_i \oplus r\left(S(R_i \boxplus K_i)\right)\right) \\ L_{i+1} &= R_i \end{aligned}$$

$$[KL_{i+1}, KR_{i+1}] = F_K(KL_i, KR_i, C_i) \Leftrightarrow \begin{aligned} KR_{i+1} &= KL_i \oplus R\left(r\left(S(KR_i \boxplus C_i)\right)\right) \\ KL_{i+1} &= KR_i \end{aligned}$$

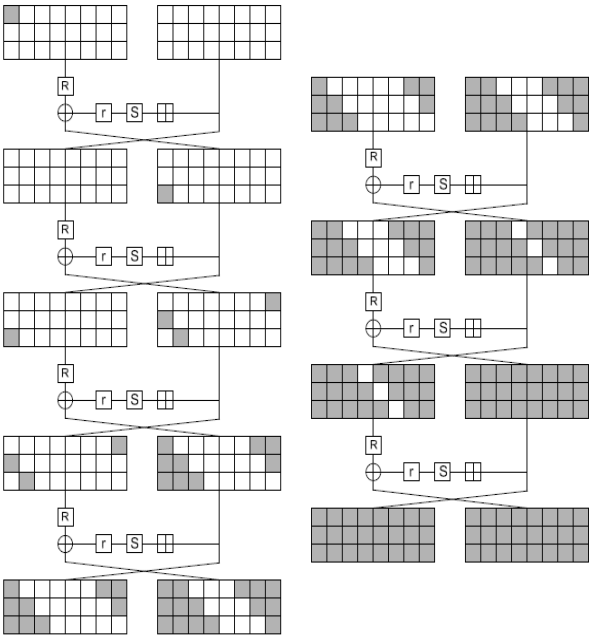
The complete cipher

The cipher iterates an odd number nr of rounds. The following pseudo-C code encrypts a plaintext P under a key Kand produces a ciphertext C. P,C and K have a parametricbit size n. The operations within the cipher are performedconsidering parametric b-bit words.

C=**SEA**n,b(P,K)
{
% initialization:
L0&R0 = P;
KL0&KR0 = K;
% key scheduling:

for i **in** 1 **to** $\lfloor \frac{n_r}{2} \rfloor$
$[KL_i, KR_i] = F_K(KL_{i-1}, KR_{i-1}, C(i));$
switch $KL_{\lfloor \frac{n_r}{2} \rfloor}, KR_{\lfloor \frac{n_r}{2} \rfloor}$;
for i **in** $\lceil \frac{n_r}{2} \rceil$ **to** $n_r - 1$
$[KL_i, KR_i] = F_K(KL_{i-1}, KR_{i-1}, C(r-i));$

% encryption:
for i **in** 1 **to** $\lceil \frac{n_r}{2} \rceil$
$[L_i, R_i] = F_E(L_{i-1}, R_{i-1}, KR_{i-1});$
for i **in** $\lceil \frac{n_r}{2} \rceil + 1$ **to** n_r
$[L_i, R_i] = F_E(L_{i-1}, R_{i-1}, KL_{i-1});$
% final:
$C = R_{n_r} \& L_{n_r};$
switch KL_{n_r-1}, KR_{n_r-1};
},



where & is the concatenation operator, KR⌊nr/2 ⌋is taken beforethe switch and C(i) is a nb-word vector of which all the wordshave value 0 excepted the LSW that equals i. Decryption isexactly the same, using the decrypt round FD.

III. IMPLEMENTATION OF A LOOP ARCHITECTURE

A. Description

The structure of our loop architecture for SEA is depictedin figure 2, with the round function on the left part and thekey schedule on the right part. Resource-consuming blocksare the Sboxes and the mod2b adder; the Word Rotate andBit Rotate blocks are implemented by swapping wires.According to the Specifications, the key schedule containstwo multiplexors allowing to switch the right and left part ofthe round key at half the execution of the algorithm using the appropriate command signal Switch. The multiplexorcontrolled by HalfExec provides the round function withthe right part of the round key for the first half of theexecution and transmits its left part instead after the switch.

Volume No:3, Issue No:12 (May-2018)
ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

Tosupport both encryption and decryption, we finally added twomultiplexors controlled by the Encrypt signal. Supplementaryarea consumption will be caused by the two routing paths.

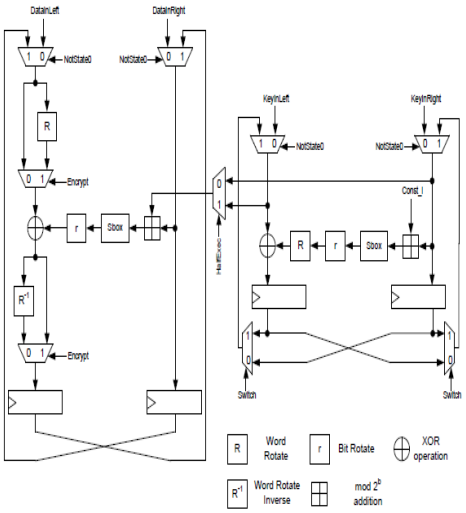


Fig. 2. Loop implementation of SEA.

The algorithm can easily beneficiate of a modular implementation,taking as only mandatory parameters the size ofthe plaintexts and keys n and the word length b. The numberof rounds nr is an optional input that can be automaticallyderived from n and b according to the guidelines given in [1].From the datapath description of Figure 2, a scalable designcan then be straightforwardly obtained by using generic Verilog HDLcoding. A particular care only has to be devoted to an efficientuse of the mod 2^b adders in the key scheduling part. In the round function, the mod 2^b adders are realizedby using n_b b-bits adders working in parallel without carrypropagation between them. However, in the key schedule, thesignal Const_i (provided by the control part) can only takea value between 0 and $n_r/2$. Therefore, it may not be necessaryto use nb adders. If $\log2(n_r/2\ ) \leq b$, then a single adder issufficient. If $\log2(n_r/2\ ) > b$, then $[\log_2(n_r/2\ )/2]$ adders will berequired. In the next section, we detail the implementationresults of this architecture for different parameters.

B. ENCRYPTION AND DECRYPTION FLOWCHART

Figure.3 shows the encryption flow chart used in design of theprogram. The data and key are the inputs. In the next step bothinputs are divided into two parts and applied to the processingblocks. The encryption is completed in two loop operations. Infirst loop i will take a value of 1 to nr/2. That is the half executionpart, the right part of the key is selected during this operations.Both key and data swap in end of each, iteration. After finishingthe half execution switch operation is performed. It is done byswap left and right part of key and the remaining rounds the keypart will not swap in the next loop. The same operation isperformed in next loop except that the left part key is selected inthe round operation. Finally the encrypt output is taken byconcatenating right and left part output of encrypt round.Figure.4 shows decryption flow chart, the same process is doneduring this flowchart except that inverse word rotation operationis performed after bit rotation, instead in encrypt round the wordrotation is performed before bitwise XOR.

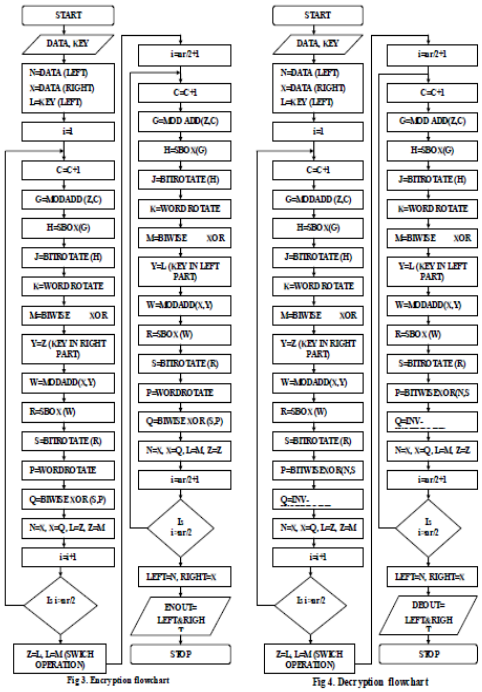


Fig 3. Encryption flowchart

Fig 4. Decryption flowchart

Volume No:3, Issue No:12 (May-2018) ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

C. Implementation results

Implementation results were extracted after place and route with the ISE 9.2i tool from Xilinx on a xc4vlx25 VIRTEX-4 platform with speed grade -12. In order to illustrate themodularity of our architecture, we ran the design tool fordifferentsets of parameters, with plaintext/key sizes n rangingfrom 48 to 144 bits and word lengths of 4, 6, 7, 8, and 12bits. For the control part, we used the recommended number ofrounds

$$n_r = [3\frac{n}{4} + 2(\frac{n}{2b} + \frac{b}{2})]^1$$

The computed implementationcosts stand for both the operative and control parts.A summary of these results is presented in table I, wherethe area requirements (in slices), the work frequency andthe throughput are provided. We observe that the obtainedvalues for the work frequency are very close for alltheimplementations. Indeed, the critical path (passing throughthe key scheduling multiplexors, a mod 2b adder, the RoundFunction Sbox, a XOR operator and the multiplexor selectingbetween encryption or decryption paths) is very similar forany of our selected values for n and b.

TABLE I
IMPLEMENTATION RESULTS FOR SEA WITH DIFFERENT n AND b PARAMETERS

n	b	n_r	♯ of slices	♯ of slice FFs	Output every cycle	Freq (MHz)	Throughput (Mbits/sec)	Thr./Area Mbits/sec /slice
48	4	55	197	127	1/55	237	207	1.049
48	8	51	176	131	1/51	234	220	1.250
72	4	77	296	194	1/77	243	228	0.769
72	6	73	258	194	1/73	242	238	0.924
72	12	73	263	198	1/73	242	239	0.908
96	4	95	368	241	1/95	242	244	0.663
96	8	93	333	246	1/93	238	245	0.737
108	6	111	376	280	1/111	241	235	0.625
126	7	117	438	328	1/117	241	260	0.593
132	11	121	448	330	1/121	227	248	0.554
144	4	149	604	376	1/149	241	233	0.385
144	6	139	488	359	1/139	241	250	0.512
144	8	135	496	371	1/135	241	257	0.518
144	12	133	478	352	1/133	223	236	0.495

For a given n value, it is noticeable that increasing bdecreases the number of rounds n_r and therefore improvesthe throughput (since work frequencies are close in all ourexamples). Similarly, for our set of parameters, increasing b fora given n generally decreases the area requirements in slices.These observations lead to the empirical conclusion that, aslong as the b parameter is not a limiting factor for the workfrequency, increasing the word size leads to the most efficientimplementations for both area and throughput reasons.

D. Comparisons with other block ciphers

For our comparative discussions, we reported a few implementationresults of the AES Rijndael in Table II. Weselected the implementations in [7], [8] and [9] because theirdesign choices fit relatively well with those of the presentedSEA architectures. Mainly, these cores do not take advantageof RAM blocks nor loop unrolling. The four first cores allcorrespond to loop architectures with a 128-bit datapath. Theyrespectively have no pipeline (Pipe0) or a 3-stage pipeline(Pipe3) and use LUT-based or distributed RAM-based Sboxes.The fifth referenced implementation [7] uses a 32-bit datapathand consequently reduces the area requirements at the costof a smaller throughput. Finally, [8] uses a 128-bit datapathwith a pipelined composite field description of the Sbox. Asa matter of fact, a lot of other FPGA implementations of theAES can be found in the open literature, e.g. taking advantageof different datapath sizes, FPGA RAM blocks, pipelining,unrolling techniques, ..., e.g. [10], [11], [12] and [13].

Additionally, we compared these results with those obtainedfor ICEBERG, a block cipher optimized for reconfigurablehardware devices. Details on the ICEBERG architecture anddifferent possible implementation tradeoffs are discussed in[14]. The reported result corresponds to a single-round looparchitecture without pipeline. Compared to the AES Rijndael,ICEBERG is built upon a combination of 4-bit operations thatperfectly fit into the FPGAs LUTs which intently results in avery good ratio between throughput and area.The implementation results in Table II lead to the followingobservations.

Volume No:3, Issue No:12 (May-2018) ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

First, in terms of area requirements (for a datapathsize equal to the block size), SEA generally exhibits thesmallest cost. Measuring the area efficiency with the bit perslice metric leads to a similar conclusion. Of course, the areарequirements of, e.g. the AES Rijndael could still be decreasedby using smaller datapaths [15] and such a comparative tableonly serves as an indicator rather than a strict comparison.However, in the present case, these results clearly suggest thelow-cost purpose of our presented implementations.By contrast, looking at the throughput per area metricindicates that these low area requirements come with weakthroughputs. This is of course mainly due to the high numberof rounds in SEA.With this respect, it is interesting to compareSEA and ICEBERG since their implementation results clearlyillustrate their respective context/platform-oriented design approach.Namely SEA is purposed for low cost applicationswhile ICEBERG optimizes the throughput per slice.

These numbers also confirm the differences between specializedalgorithms and standard solutions. It must be underlinedwith this respect that the AES Rijndael still rangesrelatively well in terms of hardware cost and throughputefficiency, compared to the investigated specialized solutions.Note also that SEA was initially purposed for low costsoftware implementations. While these design criteria turnedout to allow low cost hardware implementations as well, it islikely that targeting a cipher specifically for low cost hardwarewould lead to even better solutions, e.g. [16]. Finally, it is also important to emphasize a number ofadvantages in SEA that cannot be found in other recent blockciphers, namely its simplicity, scalability (re-implementingSEA for a new block size does not require to re-write code), good combination of encryption and decryption and ability toderive keys “on the fly” both in encryption and decryption.

TABLE II
IMPLEMENTATION RESULTS OF OTHER BLOCK CIPHERS.

Algorithm	Device	n_r	E/D	# of slices	Freq (MHz)	Throughput (Mbits/sec)	Thr./Area Mbits/sec /slice	bit/slice
AES (Pipe0-LUT) [9]	xc2v400	10	no	2744	59	760	0.277	0.047
AES (Pipe0-Dist) [9]	xc2v400	10	no	1780	78	1000	0.562	0.072
AES (Pipe3-LUT) [9]	xc2v400	10	no	2909	148	1890	0.650	0.044
AES (Pipe3-Dist) [9]	xc2v400	10	no	1940	178	2280	1.175	0.066
AES [7]	xcv100e	10	yes	1125	161	215	0.191	0.114
AES [8]	xcv3200e	10	no	1769	167	2085	1.179	0.072
ICEBERG	xc4vlx25	16	yes	575	247	988	1.718	0.111
$SEA_{126,7}$	xcv3200e	117	yes	434	92	99	0.228	0.290
$SEA_{126,7}$	xc2v4000	117	yes	424	145	156	0.368	0.302
$SEA_{126,7}$	xc4vlx25	117	yes	438	241	260	0.594	0.288

IV SYNTHESIS AND SIMULATION RESULTS

To investigate the advantages of using our technique in terms of area overhead against “Fully ECC”and against the partially protection, we implemented andsynthesized for a Xilinx XC3S500E different versions of a32-bit, 32-entry, dual read ports, single write port registerfile. Once the functional verification is done, the RTL model is taken to the synthesis process using the Xilinx ISE tool.

In synthesis process, the RTL model will be converted to the gate level netlist mapped to a specific technology library. Here in this Spartan 3E family, many different devices were available in the Xilinx ISE tool. In order to synthesis this design the device named as “XC3S500E” has been chosen and the package as “FG320” with the device speed such as “-4”. The corresponding schematics of the adders after synthesis is shown below.

Volume No:3, Issue No:12 (May-2018)
ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

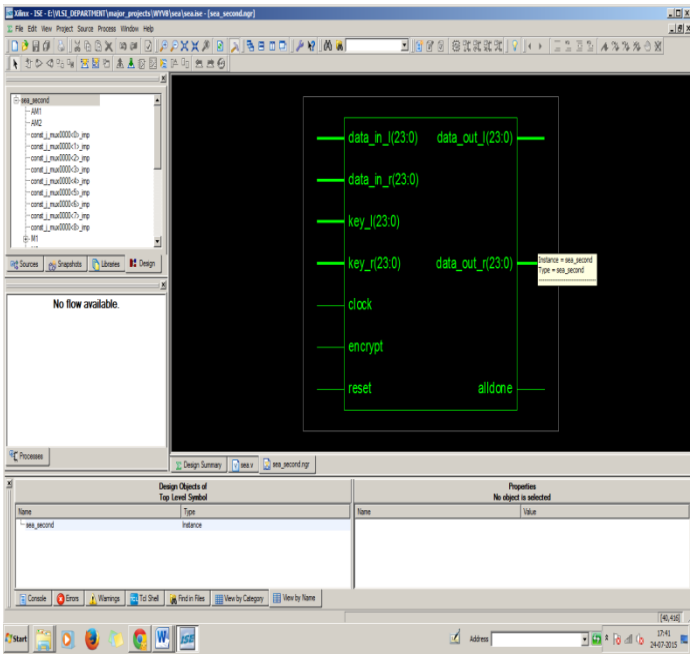


Fig.3. RTL schematic of SEA

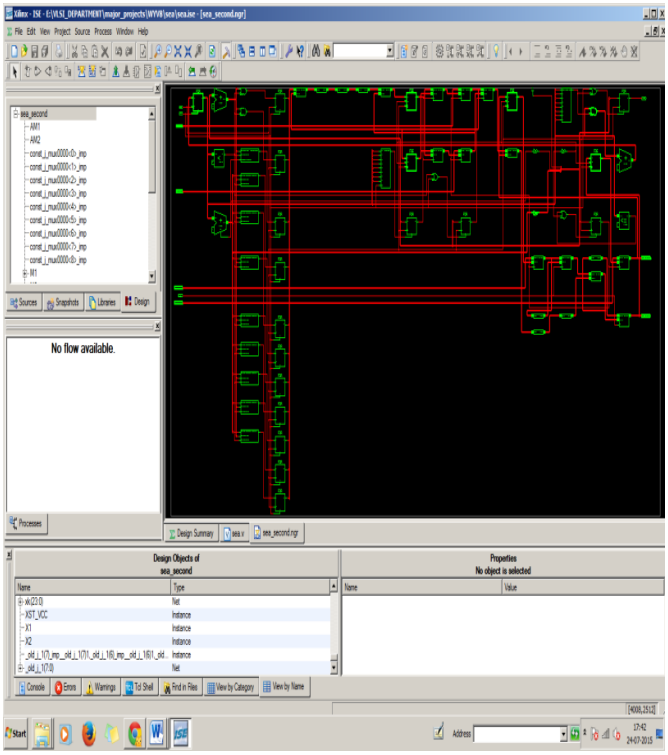


Fig.4. RTL schematic of Internal blocks of SEA

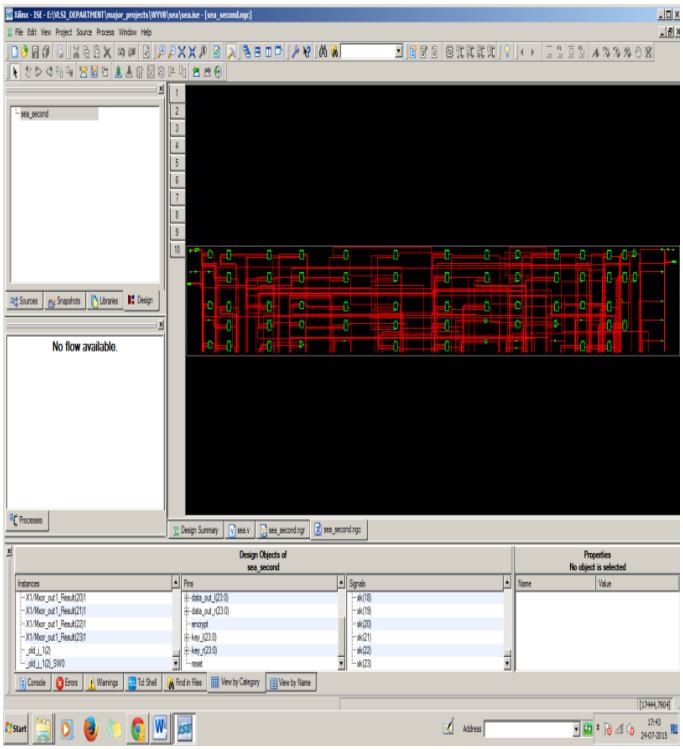


Fig.5. Technology schematic of SEA

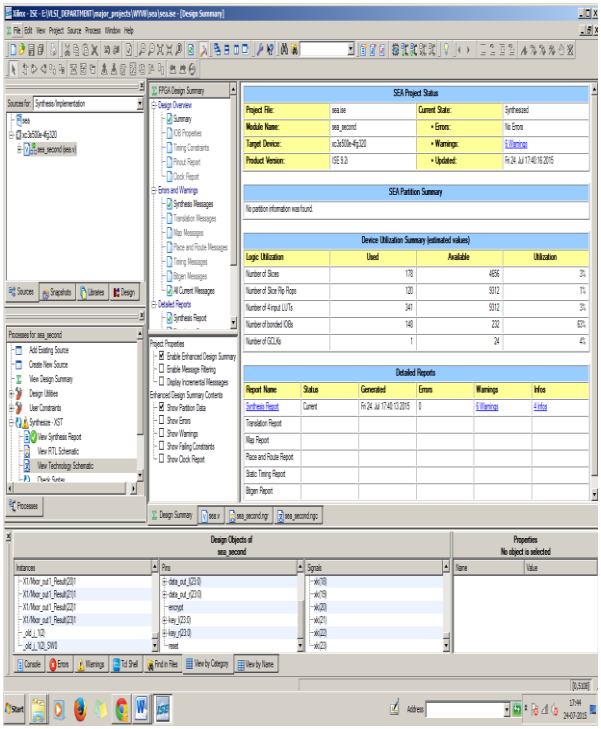


Fig.6.Synthesis report of SEA

Volume No:3, Issue No:12 (May-2018) ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

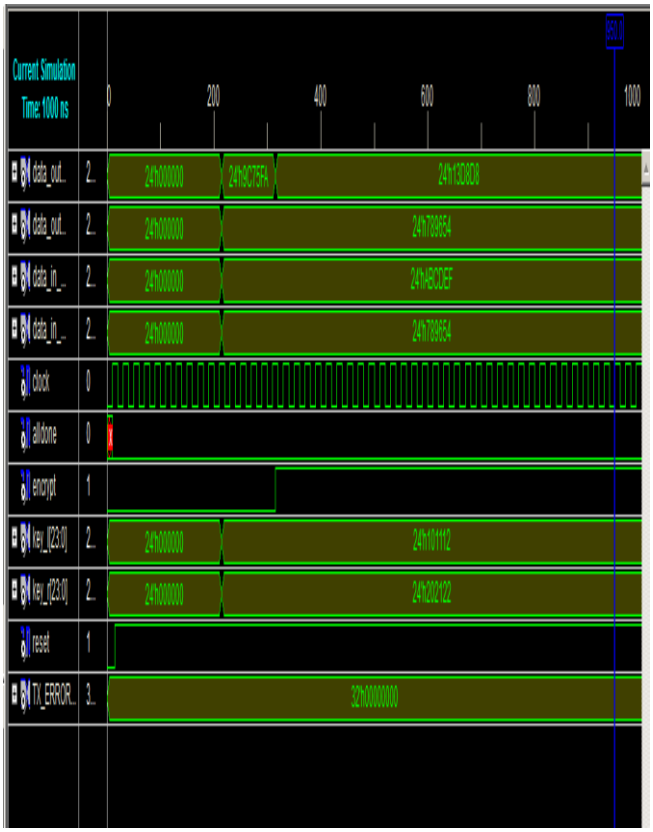


Fig.7.Simulation of SEA

V. CONCLUSION

This letter presented FPGA putting into uses of an (able to be made bigger or smaller) (turning messages into secret code) set of computer instructions for different sets of limits/guidelines. The presented parametric (related to the beautiful design and construction of buildings, etc.) allows keeping the flexibility of the set of computer instructions by taking advantage of plain and common thing/not a brand-name drug Verilog HDL coding. It executes one round per clock cycle, figures out/calculates the round and the key round in parallel and supports both (turning messages into secret code) and (changing secret codes into readable messages) at an (almost nothing/very little) cost. Compared to other recent block codes/puzzles, SEA shows a very small area use that comes at the cost of a reduced throughput. As a result, it can be thought about/believed as an interesting other choice for held back (surrounding conditions). Scopes for further research include low power ASIC putting into uses purposed for RFIDs as well as further cryptanalysis efforts and security (processes of figuring out the worth, amount, or quality of something).

REFERENCES

[1] F.-X. Standaert, G. Piret, N. Gershenfeld, and J.-J.Quisquater, "SEA:A Scalable Encryption Algorithm for Small Embedded Applications,"in the Proceedings of CARDIS 2006, ser. LNCS, vol. 3928, Taragona,Spain, 2006, pp. 222–236.

[2] Data Encryption Standard, NIST Federal Information Processing StandardFIPS 46-1, Jan. 1998.

[3] J. Daemen, V. Rijmen, The Design of Rijndael. Springer-Verlag, 2001.

[4] Advanced Encryption Standard, NIST Federal Information ProcessingStandard FIPS 197, Nov. 2001.

[5] D. Wheeler and R. Needham, "TEA, a Tiny Encryption Algorithm," inthe Proceedings of Fast Software Encryption - FSE 1994, ser. LNCS,vol. 1008, Leuven, Belgium, Dec. 1994, pp. 363–366.

[6] G. Yuval, "Reinventing the Travois: Encryption/MAC in 30 ROMBytes," in the Proceedings of Fast Software Encryption - FSE 1997, ser. LNCS, vol. 1267, Haifa, Israel, Jan. 1997, pp. 205–209.

[7] N. Pramstaller and J. Wolkerstorfer, "A Universal and Efficient AES Coprocessorfor Field Programmable Logic Arrays," in the Proceedings ofFPL 2004, LNCS, vol. 3203, Leuven, Belgium, Aug. 2004, pp. 565–574.

[8] F.-X. Standaert, G. Rouvroy, J.-J.Quisquater, and J.-D.Legat, "EfficientImplementation of Rijndael Encryption in Reconfigurable Hardware:

Volume No:3, Issue No:12 (May-2018) ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

Improvementsand Design Tradeoffs," in the Proceedings of CryptographicHardware and Embedded Devices - CHES 2003, ser. LNCS, vol. 2779,Cologne, Germany, Sep. 2003, pp. 334–350.

[9] J. Zambreno, D. Nguyen, and A. Choudhary, "Exploring Area/DelayTradeoffs in an AES FPGA implementation," in the Proceedings of FPL2004, ser. LNCS, vol. 3203, Leuven, Belgium, Aug. 2004, pp. 575–585.

[10] K. Gaj and P. Chodowiec, "Fast Implementation and Fair Comparisonof the Final Candidates for Advanced Encryption Standard Using FieldProgrammable Gate Arrays," in Topics in Cryptology - CT-RSA 2001,LNCS., vol. 2020, San Fransisco, USA, pp. 84-99.

[11] G. P. Saggese, A. Mazzeo, N. Mazzocca, and A. G. M. Strollo,"An FPGA-Based Performance Analysis of the Unrolling, Tiling, and Pipelining of the AES Algorithm," in the Proceedings of FPL 2003, ser.LNCS, vol. 2778, Lisbon, Portugal, Sep. 2003, pp. 292–302.

[12] A. J. Elbirt, W. Yip, B. Chetwynd, and C. Paar, "An FPGA Implementationand Performance Evaluation of the AES Block Cipher CandidateAlgorithm Finalists," in AES Candidate Conference, 2000, pp. 13–27.

[13] K. Jarvinen, M. Tommiska, J. Skytta, "Comparative Survey of High-Performance Cryptographic Algorithm Implementations on FPGAs,"IEE Proceedings on Information Security, vol. 152, Oct. 2005, pp. 3–12.

[14] F.-X. Standaert, G. Piret, G. Rouvroy, and J.-J.Quisquater, "FPGAImplementations of the ICEBERG Block Cipher," in the Proceedingsof ITCC 2005, vol. 1, Las Vegas, USA, Apr. 2005, pp. 556–561.

[15] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "AES Implementationon a Grain of Sand," in IEE Proceedings on Information Security, vol.152. IEE, Oct. 2005, pp. 13–20.

[16] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee,D. Chang, J. Lee, K. Jeong, J. Kim, and S. Chee, "HIGHT: a New Block Cipher Suitable for Low-Resource Devices," in The Proceedingsof Cryptographic Hardware and Embedded Devices - CHES 2006, ser.LNCS, vol. 4249, Yokohama, Japan, Oct. 2006, pp. 13–20.