Research Journal of Recent Sciences _____ ISSN 2277-2502
Vol. **3(5),** 65-70, May **(2014)** *Res.J.Recent Sci.*

SOC Implementation of Hybrid Cryptography Techniques using Hight and RC4 Algorithm

Blesslin Sheeba T.[1] and Rangarajan P.[2]
[1]Sathyabama University, Chennai, INDIA
[2]Department of EEE, RMD Engineering College, Chennai, INDIA

Available online at: www.isca.in, www.isca.me
Received 11th February 2014, revised 10th April 2014, accepted 26th April 2014

Abstract

In this paper we present a hybrid cryptographic technique implemented on System on chip, hybrid in the sense that we are going to implement both stream cipher and block cipher in a SOC. The encryption of data will be performed by different cipher base on the application.. Here we are integrating HIGHT and RC4 cryptographic algorithm for block cipher and stream respectively in ALTERA Cyclone IV E, so that we can achieve both encryption of data for transmitting through the communication link (i.e. stream cipher) and encryption data like files and images (i.e. Block cipher) can be implemented in single SOC. Finally we are computing the efficiency for both ultra lightweight cryptography (HIGHT) and lightweight cryptography (RC4) implemented on single SOC.

Keywords: Hight, RC4, System on chip (SOC), Cyclone IV E, Block cipher, Stream Cipher.

Introduction

In this day, with the explosion in the Internet based Electronic Commerce; there is an increasing need to secure the data and commercial transactions. The intent to transmit messages securely is not new. For centuries, people want to keep their communications secret. Hence, a mechanism is required to guarantee the security and privacy of information that is transmitted over the electronic communications media and in the modern era of inexpensive internet connections, data computing and global data communications there is high demand for energy efficiency, computational speedup and data security. There are many effort has been taken in response to ever-increasing need for data security, to protect the data from unwanted action, unauthorized user and destructive force the cryptography plays an important role. Clearly there are two ways to implement any algorithm, i.e. either hardware or software, the choice of platform depends on algorithm performances, cost and flexibility. However in software level implementation of this algorithm leads to high energy consumption of the system, increase in complex algorithm, computational overhead and long execution time. Successful studies have been made for implement of cryptography algorithm to reduce the before said cons. The computational speed can be increased and power consumption can be reduced by increasing the number of core in single chip. High impact computing problem on more capable hardware is emergence realization for many-core system[1]. To get high speed networks the hardware always appears to be the ultimate choice because hardware implementation of cryptographic algorithms is intrinsically more physically secure and run faster than software. In hardware implementations, the flexibility and high speed capability of SOC make them a suitable platform for cryptographic applications. Their reconfigurability means that they can perform the more computationally intensive operations of a range of ciphers depending on security and application requirements.

The two basic scheme of encryption are one-key and two-key ciphers. If the encryption of the plain text and decryption of the corresponding cipher text are performed using same key then it is one-key cipher in contrast if it is performed with different key then it is called is two-key cipher. Further one-key cipher is divided in to block cipher and stream cipher based on their process of computing the cipher text. The process of dividing the plain text in to fixed length block and then performing the encryption on each block to produce cipher text using same key is called as Block cipher and in relation to this if short string of key is used to generate the long sequence of bits and it is added bitwise modulo 2 to the plain text to produce the cipher text means then it will become Stream cipher.

Methodology

Block Cipher: Let M be a plain text message. A block cipher breaks M in to successive blocks M1, M2,…, and key K is used to encipher for each M1 that is

EK(M) = EK(M1) EK(M2)…

Advantages and Disadvantages: i. N character is executed each time so faster than stream cipher. ii. The error one cipher block does not affect the other during transmission. iii. It is used to reduce the time consumption and easy for software implementation. iv. Deletions and Insertions of block is easy. v. Modifying the block is possible. vi. Block encryption is more susceptible.

Stream Cipher: It will break the message M in to bits or successive character M1, M2, …, and K_i a key stream (i.e. K = K1, K2,…)is used to encipher each M_i that is

EK(M) = EK1(M1) EK2(M2)…

Advantages and Disadvantages: i. Encryption and Decryption of one bit at a time will reduce the complexity of hardware implementation. ii. It is resist to deletion and insertion of block. iii. Simple mathematical analysis will reduce the computational complexity. iv. Non dependability of key stream over the message stream will be more secure. v. It will not suitable in software. vi. One error bit will cause a whole transmission error. vii. If the key is short length then repeating of block is possible.

It would be efficient if we achieve the pros of both the cipher we are implementing both the cipher in a single processor (i.e. SOC). The below FIG shows the overall architecture of the propose SOC implementation of both the cipher. In our paper we are considering two algorithm namely. i. HIGHT - Ultra lightweight cryptography for Block cipher. ii. RC4 – Lightweight Cryptography for stream cipher.

From *P.Yalla et.al.,* and *Poschmann et.al.,* the cryptography application like RFID, images, files router use both the lightweight and ultra lightweight cryptography algorithm.

Encryption: From the architecture shown in figure 1 it is clear that in NIOS II processor the input is decided by the decision block, if it is images and files the encryption is done using Hight algorithm (i.e. block cipher) which is ultra lightweight cryptography algorithm, and if file which is transferable through network is detected then encryption is performed using stream cipher (i.e. RC4 algorithm) which is lightweight cryptography algorithm, finally we have the cipher text as encrypted output.

Decryption: The encrypted output or cipher text is given to decrypted block for the reverse process of encryption as shown in figure 2. and the same algorithm for ciphering the text will be used to deciphering the text and finally the we get the plain text. Both the cryptography algorithm were written in verilog HDL language and integrated in NIOS II processor. Both lightweight and ultra lightweight cryptographic algorithm will be elaborate in next section.

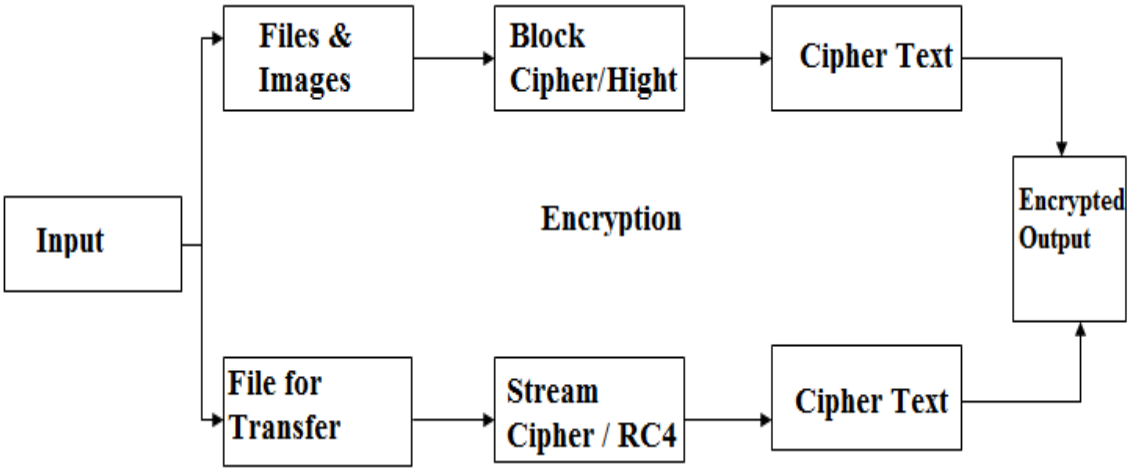

Figure-1
Hybrid Encryption Technique

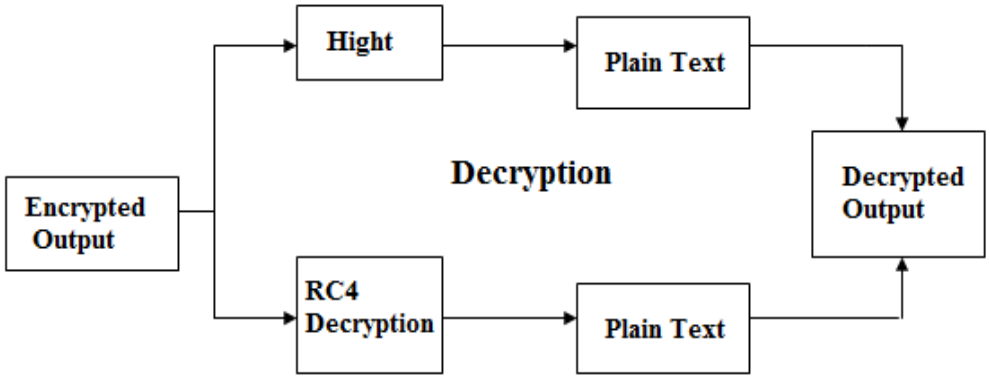

Figure-2
Hybrid Decryption Technique

Cryptographic Algorithm: Lightweight RC4 Algorithm: RC4 is a, variable key size and stream cipher with byte-oriented operations[4,5]. Algorithm is based on the use of a random permutation. The analysis shows that the period of the cipher is overwhelmingly likely to be greater than 10100 [ROBS95]. Eight to sixteen machine operations are required per output byte and cipher can be expected to run very quickly in software[6]. RC4 was kept as a trade secret by RSA Security. The RC4 algorithm is remarkably simple and quite easy to explain, shown in figure 3. A variable-length key from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S[0], S[1], …, S[255]. At all times, S contains a permutation of all 8-bit numbers from 0 through 255. For encryption and decryption byte k (figure 1) is generated from S by selecting one of the 255 entries in a systematic fashion. Each value of k is generated the entries in S are once again permuted.

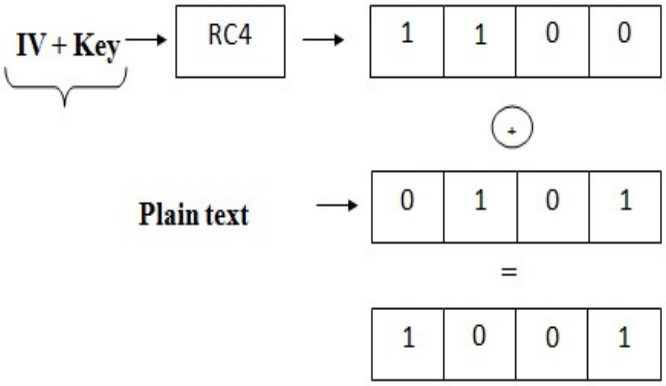

Figure-3
RC4 Architecture

Initialization of S: To begin, the entries of S are set equal to the values from 0 to 255 in ascending order; S[0] = 0, S[1] = 1, …, S[255] = 255. The temporary vector T also created. The length of the key K is 256 bytes, then K is transferred to T[3]. Otherwise, a key length of keylen bytes, the first keylen elements of T copied from K and then K is repeated as many times as necessary to fill out T.

Because the only operation of S is a swap and the only effect is a permutation. S is still contains all the numbers from 0 to 255.

Stream Generation: The input key is no longer used when the S vector is initialized. Stream generation involves starting with S[0] and going through to S[255] and each S[i], swapping S[i] with another byte in S. according to the scheme dictated by the current configuration of S. After the S[255] is reached then the process continues starting over again at S[0]:

To encrypt, XOR the value k with the next byte of plaintext. To decrypt, XOR the value k with the next byte of cipher text[7].

Strength of RC4: A number of papers have been published analyzing methods of attacking RC4 [e.g., [KNUD98], [MIST98], [FLUH00], [MANT01]). None of this approach is practical against RC4 with a reasonable key length as 128 bits. A more serious problem is reported in [FLUH01]. The authors demonstrate that the WEP protocol, intended to provide confidentiality 802.11 wireless LAN networks are vulnerable to a particular attach approach. In essence, the problem is not with RC4 itself but the way in which keys are generated for use as input to RC4. The particular problem does not appear to be applicable to other applications using RC4 and can be remedied in WEP and by changing the way in which keys are generated [8,9]. This problem points out the difficulty in designing a secure system that involves both cryptographic functions and protocols that make use of them.

Ultra Lightweight Hight Algorithm: The HIGHT algorithm consists of 32-rounds with initial and final transformations before the first and after the last rounds respectively. The plaintext P and cipher text C are split into eight 8-bit blocks P_7, … , P_0 and C_7, … , C_0 and the original key K into sixteen 8-bit blocks K_{15} , … , K_0.

The initial transformation uses the four whitening key bytes WK_0, WK_1 , WK_2 and WK_3 to transform a plain text P into the input of first round function $X_0 = X_{0,7} \| \ldots \| X_{0,0}$. In the final transformation, the data is shifted towards right and transforms $X_{32} = X_{32,7} \| \ldots \| X_{32,0}$ into cipher text C by with the four whitening keys WK_0 , WK_1 , WK_2 and WK_3 to transform a plain text P into the input of first round function $X_0 = X_{0,7} \| \ldots \| X_{0,0}$. In the final transformation, the data is shifted towards right and transforms $X_{32} = X_{32,7} \| \ldots \| X_{32,0}$ into cipher text C by with the four whitening keys WK_0 , WK_4 , WK_5, WK_6 and WK_7 . Both transformations perform a XOR or modular addition. The eight 8- bit whitening keys for initial and final transformation $WK_7 \ldots WK_0$ are generated using equation (1).

For $0 \le i, m \le 7$

$$WK_i \leftarrow K_{(i+1) \bmod 8}, WK_{i+4} \leftarrow K_i \quad (1)$$

$$SK_{16i+m} \leftarrow K_{(m-i) \bmod 8} \boxplus \delta_{16i+m} \quad (2)$$

$$SK_{16i+m} \leftarrow K_{(m-i) \bmod 8 + 8} \boxplus \delta_{16i+m+8} \quad (3)$$

The round function uses two auxiliary functions F_0 and F_1 described in equation and respectively, along with XOR and modular addition operations. The two functions F0 and F1 provide bitwise diffusion which is similar to linear transformation from $GF(2)^8$ to $GF(2)^8$. The round function transforms the input, $X_i = X_{i,7} \| \ldots \| X_{i,0}$ into $X_{i+1} = X_{i+1,7} \| \ldots \| X_{i+1,0}$ for $i = 0, \ldots, 31$, which is shown in figure.

The i^{th} round key RK_i Consist of four 8-bit sub keys $SK_{4i}, SK_{4i+1}, SK_{4i+2}$ and SK_{4i+3} which is generation through Eq. and . The 7-bit constants $\delta_0, \ldots, \delta_{127}$ are generated by 7-bit LFSR h. The Characteristics polynomial of h is $x^7 + x^3 + 1$ in $\mathbb{Z}_2$ with a period of $2^7 - 1 = 127$. The initial value of h is set to 1011010_2.

$$F_0(x) = x^{\lll 1} \oplus x^{\lll 2} \oplus x^{\lll 7} \quad (4)$$

$$F_1(x) = x^{\lll 1} \oplus x^{\lll 4} \oplus x^{\lll 6} \quad (5)$$

Ultra Lightweight Architecture of Hight: The scaling of 64 bit algorithm to 8 bit algorithm is reduced the area consumption of HIGHT algorithm.

Data Storage: The most efficient solution for data storage is shift register which is used in round function, but the generalized the HIGHT structure lead to misalignment of data when shift register is used. Additionally clock cycle is expected for realignment reducing the throughput. The complex logic will increase the area consumption. Therefore to overcome it DRAM is used for efficient in term of latency and area. One 8-bit block of data is computed using two 8-bit blocks of data using dual-port DRAM as round function and the shifting is accomplished by addressing. Three 3-bit multiplexer, 12-bit shift register, 3-bit counter are used to generate the address need for all operations as shown in figure 4. 5-bit counter will generate a control signal for all the operation.

Round Function, Initial and Final Transformation: The shift register and 3-bit adder generate the address for round function, the 2 LSB bit of C3 are taken as initial value for computing the first round. This way we can reduce the complexity of control logic and clock cycles. The two extra multiplexer are used to perform the datapath round function for initial and final transformation. The data is loaded in to the shift register during initial transformation to save clock cycles.

Key Storage and Scheduling: The single-port DRAM is made of 128-bit. The two 3-bit counters and 2x1 multiplexer will generate the subkeys and whitening keys. The one 4-bit counter is replaced by two 3-bit counters for addressing.

SOC Implemwntation of Proposed Hybrid Architecture: The added advantage of SOC implementation is that applicability on high volume production circuit, modification and reprogramming of the design can be done at lower unit cost and power consumption is low[10,11].

Figure-5 shows the architecture for SOC implementation of Hybrid Cryptographic technique in which Hight ultra lightweight cryptography and RC4 lightweight cryptography is implemented. The design is implemented in Altera IV E. The data are transmitted and received through Avalon bus, the processing of data is done in Nios processor. We are integrating the SRAM with the architectures so that the algorithm for Hight and RC4 can be dumped on it. The UART interface is used to programming, the control logic or decision block is implemented in processor internal memory so that the decision architecture will choose the data for encryption based on the application. The data to the processor will be sent from the two blocks of input and output buffer through the DMA controller; it will control the flow of data and allocate the Avalon bus to different input based on the current status of the processor. Further to this, processor interface with the SRAM for particular algorithm with the help of SRAM controller.

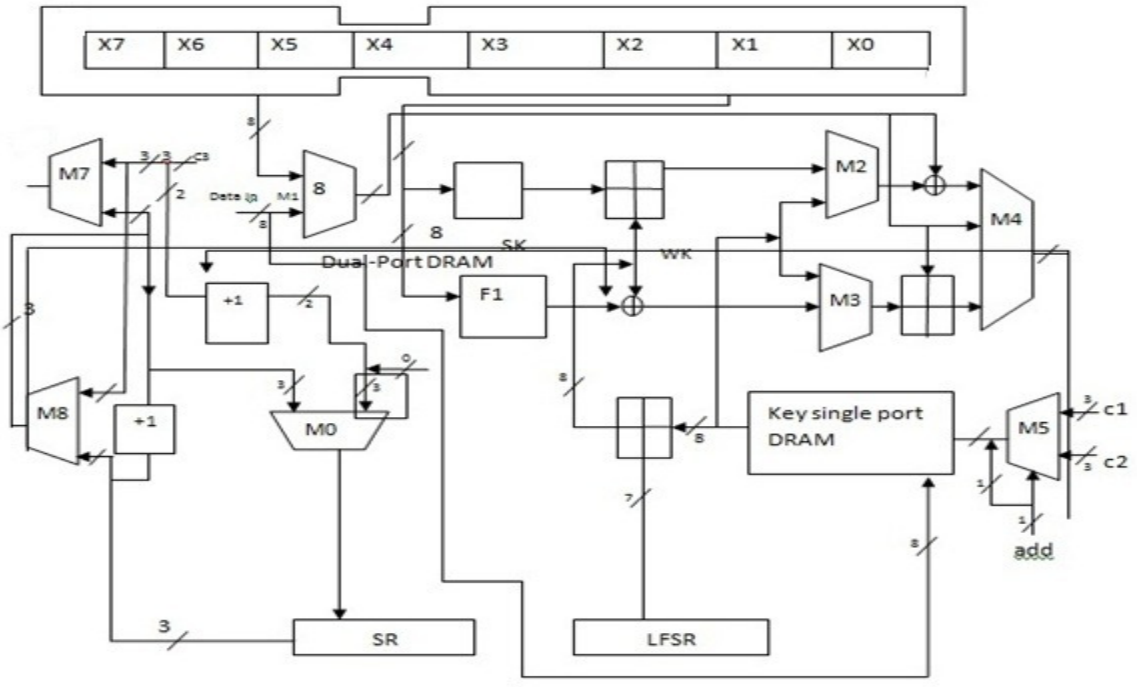

Figure-4
HIGHT Architecture

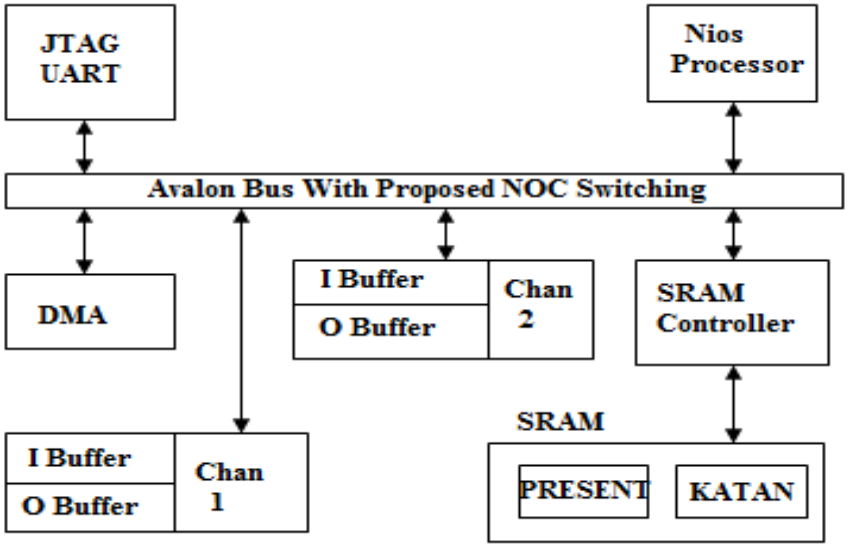

Figure-5
SOC Architecture of Hybrid Cryptography

Results and Discussion

The various results of the proposed Hybrid cryptographic techniques is discussed below. The operation time of the both lightweight cryptography and ultra lightweight cryptography were found to be reduced when comparing with software implementation which is shown in table 1.

And it will be more efficient in a high end application. Analyzing the speed of the proposed system we realize that both software implementation and SOC implementation of cryptographic algorithm produce the result with SOC implementation slightly higher than the Software. But if we consider the key, which means if the length of the key is increased means the speed also increase abruptly in SOC implementation than software which is shown in figure 6.

Table-1
Operation Time

Types	Algorithm	Operation Time (SEC)
Software	HIGHT	7.413
	RC4	7.264
SOC	HIGHT	6.114
	RC4	6.007

And if we depend on the CPU cycle for particular application then the table shows an improved efficiency for the same. It is clear from the result that CPU cycle for the SOC implementation is well suited than the Software implementation, and the efficiency is shown in table 2. Finally we calculate efficiency in terms of Memory, usually the gate elements for ultra lightweight cryptography is less than that of lightweight cryptography and if we implement it in the SOC means the gate count is further reduced and ultra lightweight cryptography gate count is reduced more than that of lightweight cryptography as shown in figure 7. And it will be efficient in application with lesser gate count.

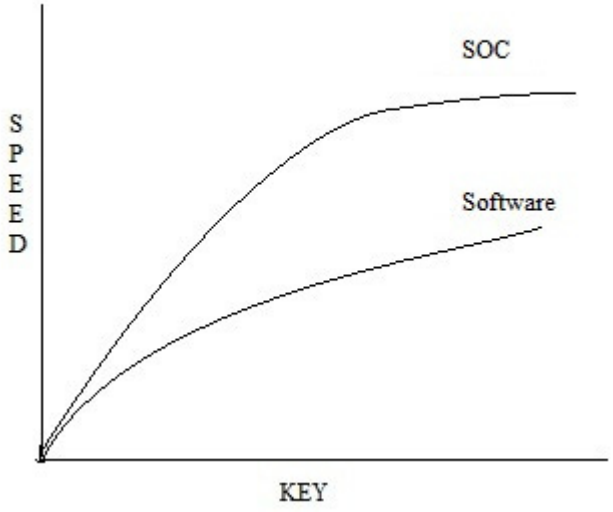

Figure-6
Length of bit Vs Speed

Table-2
CPU Cycle

Types	Algorithm	CPU Cycle
Software	HIGHT	64355
	RC4	70700
SOC	HIGHT	40236
	RC4	56119

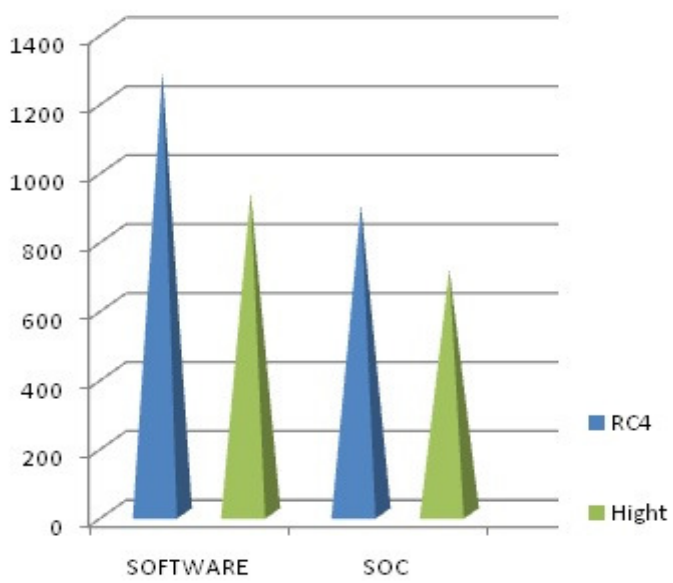

Figure-7
GE count in SOC Implementation

Conclusion

SOC implementation of Hybrid cryptographic techniques is presented in this paper. Proposed implementation achieves the efficiency in terms of operation time and CPU cycle as tabulated above. The speed of the Proposed system is considerable since it has is slightly more than the Software implementation for key with small bits, and finally the speed is increased if the number of bits is increased. Finally the gate count is found to be more less for Ultra light weight cryptography than the other in SOC implementation.

References

1. Diamos, Gregory and Sudhakar Yalamanchili, An Execution Model and Runtime For Heterogeneous Many-Core Systems, 197-200 **(2008)**

2. Yalla Panasayya and Kaps J., Lightweight cryptography for FPGAs, *International Conference on Reconfigurable Computing and FPGAs*, 225-230 *IEEE*, **(2009)**

3. Poschmann Axel, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling, Side-channel resistant crypto for less than 2,300 GE, *Journal of Cryptology,* **24(2)**, 322-345 **(2011)**

4. Paul Rourab, et al., A simple 1-byte 1-clock RC4 design and its efficient implementation in FPGA coprocessor for secured ethernet communication, *arXiv preprint arXiv:*1205.1737 **(2012)**

5. Sasidharan, Sapna, and Deepu Sleeba Philip. "A fast partial image encryption scheme with wavelet transform and RC4, *Intl Journal of Advances in Engineering & Technology,* **1(4),** 322-331 **(2011)**

6. William Stallings, Cryptography – RC4 Algorithm, October **(2011)**

7. Mousa, Allam, and Ahmad Hamad, Evaluation of the RC4 Algorithm for Data Encryption, *IJCSA,* **3(2),** 44-56 **(2006)**

8. Fluhrer, Scott, Itsik Mantin, and Adi Shamir, Weaknesses in the key scheduling algorithm of RC4, *Selected areas in cryptography,* Springer Berlin Heidelberg, 1-24 **(2001)**

9. Saarinen M-JO, The BlueJay Ultra-Lightweight Hybrid Cryptosystem, *IEEE Symposium on Security and Privacy Workshops (SPW),* 27-32 **(2012)**

10. Yuan X-C., et al., Hybrid encryption and decryption technique using microfabricated diffractive optical elements, *Optical Engineering,* **43(11),** 2493-2494 **(2004)**

11. Torkaman Najaf, et al., Innovative Approach to Improve Hybrid Cryptography by Using DNA Steganography, *International Journal of New Computer Architectures & their Applications,* **2(1) (2012)**