Hardware Implementation of High Speed RC4 Algorithm in FPGA

Chandra Mouli.R
Department of ECE G.V.P.College of Engg (A)
Visakhapatnam, AP, India

K.R.K.Sastry
Department of ECE G.V.P.College of Engg (A)
Visakhapatnam, AP, India

ABSTRACT
This paper presents high speed and an area efficient hardware implementation of the RC4 algorithm. The proposed design uses Block RAM (BRAM) implementation to reduce the area and to increase the speed of operation hence throughput. The proposed design uses only one 256 bytes simple dual port RAM for key stream generation and it takes 3 clock cycles per byte. It supports a variable key length of from 1 byte to 256 bytes and achieves 54.8MB/s throughput at 164.6MHz operating frequency. The design is targeted on XC2V250FG256 Xilinx FPGA and met the operating frequency of 164.6MHz. The RC4 algorithm is implemented in Verilog HDL.

Keywords
BRAM, CPLD, FPGA, RC4 Algorithm, Stream Cipher, Simple Dual Port RAM.

1. INTRODUCTION
The RC4 algorithm has been proposed by Ron Rivest in 1987. It is a variable key size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation. In terms of application, RC4 is the most widely used software based stream cipher. For example, it is used in Secure Socket Layers (SSL) to protect internet traffic and in Wired Equivalent Privacy (WEP) to secure wireless network.

In RC4 algorithm byte wise swapping of S-box elements is needed. To perform byte swapping, S-box elements need to be processed through three different steps viz. i) read i^{th} location data from the S-box. ii) Calculate for a new location j, read j^{th} location data from the S-box and move the i^{th} location data into j^{th} location iii) write j^{th} location data into i^{th} location. Each byte of ciphering key is generated after all three steps are finished. The hardware implementation of the RC4 algorithm proposed in [1] takes 8 clock cycles, a software implementation of RC4 using assembly language presented in [2] requires 7 clock cycles, a RAM-based CPLD implementation in [3] require 4 clock cycles, RAM-based hardware implementation of RC4 stream ciphers in [4] and [5] takes 3 clock cycles. In addition the architecture used in [4] uses 3 blocks of 256bytes RAMs for swapping of S-box elements and the throughput is 22MB/sec. In another approach presented in [6], a register based S-box is implemented, its processing time is reduced to 1 clock cycle but the long critical path caused by large MUX and DEMUX gates limits the operating frequency of the core.

In this paper, the proposed implementation is parameterized in order to support variable key length from 1byte to 256bytes. It consists of a 256bytes simple dual port RAM for final key stream generation, a 64bytes Block RAM as FIFO for input data controlling operations and a 256bytes Block RAM to store the input key. The design requires 768 (256*3) clock cycles for initial permutation of the S-box elements and then it takes 3*N clock cycles to produce the N byte of cipher data.

This paper is organized as follows. RC4 algorithm is explained in Section II. The proposed architecture is shown in section III. Hardware device utilization and frequency of operation the implementation are compared with the other works in section IV. The conclusions are specified in section V.

2. RC4 STREAM CIPHER
RC4 Algorithm shown in Fig.1 uses a variable length key of 1Byte to 256Bytes, and the key is used to initialize a 256-Bytes array. The array is used for subsequent generation of pseudo-random Bytes and then generates a pseudorandom key stream, which is XORed with the plaintext/ciphertext to give the ciphertext/plaintext [7]. The RC4 algorithm has three phases: initialization of S-box, initial permutation of the S-box elements and final key stream generation. All the three phases must be performed for every new key.

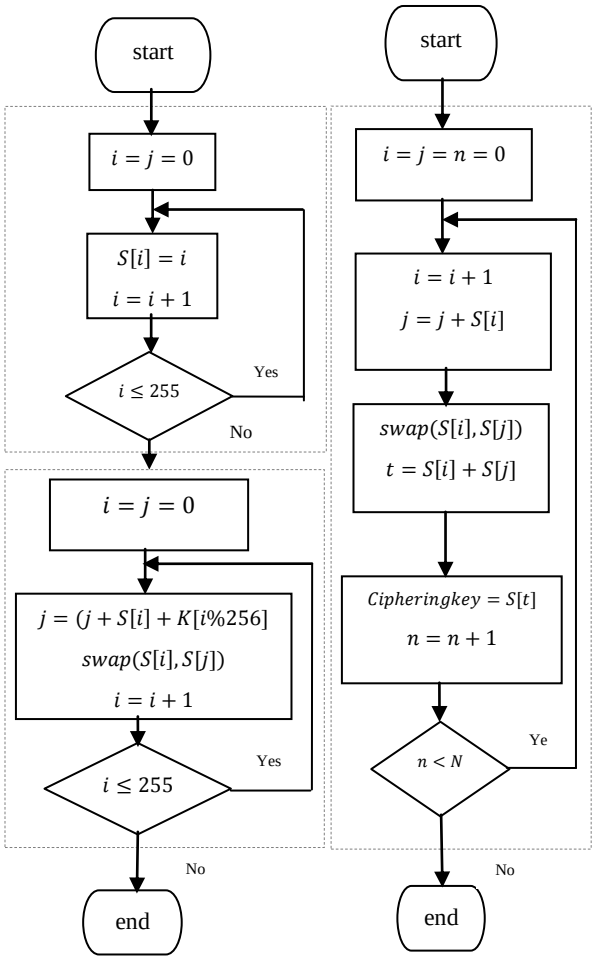

Fig. 1: RC4 Algorithm flow chart

2.1 Initialization of S-box:

The values of S-box are set equal to the values from 0 through 255 in a linear manner, i.e S[0]=0, S[1]=1, S[2]=2 …....., S[255]=255. The operations can be summarized as

for i = 0 to 255

$S[i] = i;$

2.2 Initial permutation of S:

The Initial Permutation of the S box is explained in the following pseudo code.

$j = 0;$

for i = 0 to 255

$j = j + S[i] + K[i\%key\_len])\%256;$

$swap(S[i], S[j]);$

2.3 Key Stream Generation:

Once the S vector is initialized, the input key is no longer used. Key Stream generation is explained the following code

$i = j = 0;$

$while(true)$

$i = (i + 1)\%256;$

$j = (j + S[i])\%256;$

$swap(S[i], S[j]);$

$key = \{S[(S[i] + S[j]\%256)]\};$

Every generated key byte is simply XOR ed with the incoming plaintext/ ciphertext to generate ciphertext/ plaintext.

dataout = key XOR plaintext (Byte wise)

In Fig.1, N is the number of required final key stream bytes for encryption/decryption.

3. PROPOSED ARCHITECTURE

The block diagram of the proposed architecture is shown in Fig.2. It consists of one 64bytes BRAM and two 256bytes BRAMs. The 64bytes BRAM is used as FIFO and one 256bytes BRAM is used as S-box and other 256bytes BRAM is used to store the key.

The BRAM used for the S-box is simple dual port RAM, it has one write port and one read port. The block diagram of the simple dual port RAM is shown in Fig.3. It has one write port and one read port. The write port has the signals addra, dina, wea and clka and the read port has signals addrb, clkb and doutb. If wea signal enabled then data in dina signal is stored in memory location indicated by addra signal, if the read address addrb is given then data in that addrb location is available on the doutb signal.

In the proposed architecture the Initialization process is divided into two steps. In the first step, at the initial state the S-box is filled linearly, such as S[0] = 0, S[1] = 1, ..., S[255] = 255, it is completed when reset state occurs. In the second step, when the key_vld is asserted the key is loaded into a register array, from the FPGA memory where the key of predefined length is stored.

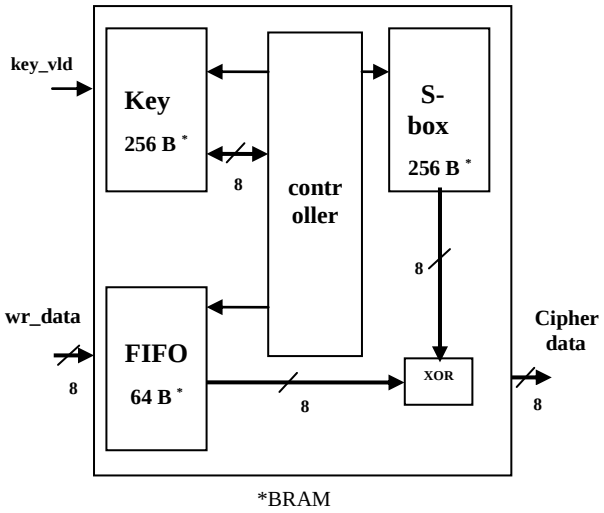

Fig 2: Block diagram of proposed architecture

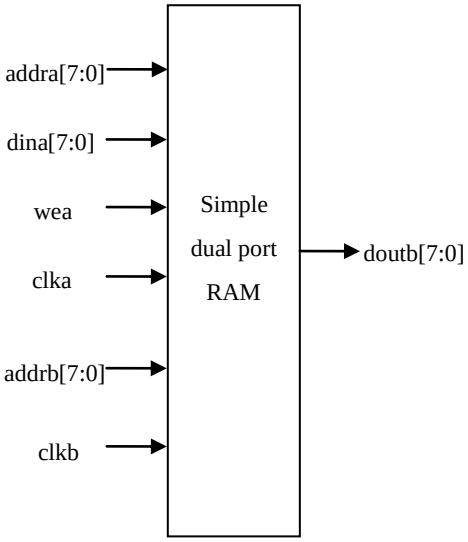

Fig 3: Block diagram of Simple Dual port RAM

The initial swapping of S-box elements stored in BRAM requires 3 clock cycles per iteration and the necessary control logic is implemented in the code. In the first clock cycle the data in the i^{th} location, S[i] is available by the read command given in the previous clock cycle. Also 'j' is calculated using S[i] and read address of the j^{th} location is asserted. In the second clock cycle data stored in the i^{th} location is written in the current j^{th} location, 'i' is incremented and given as read address of S[i], which is used in next iteration. In third clock cycle the contents of j^{th} location, S[j] available by the read command given in the first clock cycle and S[j] is written into the i^{th} location. This process is repeaters for 256 times to complete the initial permutation of S-box elements. It needs total 769 (3*256+1=769) clock cycles to complete the initial swapping.

In the key stream generation phase '*i*' and '*j*' values calculated as per algorithm shown in Fig 2 and swapping of S-box elements is performed as explained in the initial swapping. In this phase in each iteration S[{S[i]+S[j] } mod 256] is performed in the third cycle which gives the final key byte. This phase is repeated as long as the input data is available in the FIFO. Bytewise XOR operation of final key byte with the

incoming plaintext/ciphertext is implemented to generate the ciphertext/plaintext. A 64 bytes FIFO is used for the incoming data to match the delay due to the aforesaid operation

This phase requires 3*n (n is number of bytes in the input data) clock cycles. The total duration of the RC4 algorithm consisting of 769+3*n clock cycles. Any subsequent assertion of key_vld would need all the three phases to be performed as mentioned in section II.

4. RESULTS AND COMPARISON

The architecture is developed in Verilog HDL and simulated in Xilinx ISE. The whole design is targeted on Xilinx 2V250fg256 FPGA and the device implementation results are shown in Table 1.

Table1: Implementation Results

Target device : **Xilinx 2V250fg256**		
	Used	**Available**
Number of Slice Flip Flops	135	3072
Number of 4 input LUTs	290	3072
Number of bonded IOBs	23	172
Frequency	164.6MHz	
Throughput	54.8MB/s	

Table 2: Hardware Resources and Frequency Comparison

	[1]	**[3]**	**Proposed**
FPGA Device	XC400E-4013EPQ208-2	XC2V250fg256	
Number of Slice Flops	255	138	135
Frequency (MHz)	40	64	164.6
Throughput (MB/s)	5	22	54.8

The results in terms of area and frequency of proposed implementation are compared with [1], [3] in Table 2. The results show the improved area and frequency of operation as compared to [1] and [3]. The results show that the proposed system provides higher throughput 54.8 MB/s, at 164.6MHz. It also requires only 135 flops. The proposed system uses only one block of 256bytes. RAM for S-box where as [3] uses three blocks of 256bytes RAMs for S-box.

5. CONCLUSION

In this paper a Block RAM based hardware implementation of RC4 algorithm is shown. The area reduction and the higher operating frequencies are achieved by implementing simple dual port RAM (BRAM) used as S-box. The processing time to produce one byte of ciphertext is 3 clock cycles. The system supports a variable key length of 1 to 256 Bytes. The system provides 54.8MB/s throughput at 164.6MHz clock frequency.

The work may be extended to improve the processing speed and the throughput by implementing different RAM models.

6. REFERENCES

[1] P. Hamalainen, M. Hannikainen, T. Hamalainen and J. Saarinen, "Hardware Implementation of the Improved WEP and RC4 Encryption Algorithms for Wireless Terminals", The European Signal Processing Conference (EUSIPCO'2000), pp. 2289-2292, September 5-8, 2000.

[2] B. Schneier, D. Whiting, "Fast Software Encryption: Designing Encryption Algorithms for Optimal Software Speed on the Intel Pentium Processor", Fast Software Encryption workshop (FSE97), LNCS, Vol. 1267, pp. 242-259, Springer-Verlag, Haifa, Israel, January 20-22, 1997.

[3] P. D. Kundarewich, S. J.E. Wilton, A. J. Hu, "A CPLD-Based RC-4 Cracking System", The 1999 Canadian Conference on Electrical and Computer Engineering, May 1999, vol.1, pp. 397 – 402.

[4] P. Kitsos, G. Kostopoulos, N. Sklavos, and O. Koufopavlou, "Hardware Implementation of the RC4 stream Cipher", IEEE 46th Midwest Symposium on Circuits & Systems, vol.3, pp. 1363-1366, 2003.

[5] K.H Tsoi, K.H Lee and P.H.W Leong, "A Massively Parallel RC4 Key Search Engine", Proc. of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02), September 22 - 24, 2002 Napa, California, pp. 13-21.

[6] S. S. Gupta, K. Sinha, S. Maitra and B. P. Sinha, "One Byte per Clock: A Novel RC4 Hardware", 11th International Conference on Cryptology - Indocrypt 2010 Dec. 2010, India.

[7] William Stallings, "Cryptography and Network Security– Principles and Practice", Fifth Edition, Prentice Hall, 2011.