Journal of Theoretical and Applied Information Technology
10th February 2014. Vol. 60 No.1
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

HIGH THROUGHPUT HARDWARE IMPLEMENTATION FOR RC4 STREAM CIPHER

[1] *M*.RAMKUMAR RAJA, [2] DR. K. THANUSHKODI, [3] S.ARUL JOTHI

[1]Assistant Professor, Department of ECE, CIET, Tamilnadu, India
[2] Director, Akshaya College of Engineering & Technology, Tamilnadu, India
[3]Assistant Professor, Department of ECE, SREC, Tamilnadu, India

E-mail: [1]ramkumarrajavlsi@gmail.com , [2] thanush12@gmail.com , [3]arulbe2005@gmail.com

ABSTRACT

This RC4 is the most popular stream cipher in the domain of cryptology. In this paper, we present a systematic study of the hardware implementation of RC4, and propose the fastest known architecture for the cipher. We combine the ideas of hardware pipeline and loop unrolling to design an architecture that produces two RC4 key stream bytes per clock cycle. We have optimized and implemented our proposed design using Verilog description, synthesized with 45nm technology. The proposed design has a total area of 138459um^2 and shows a power consumption of 382.0935mW.The proposed circuit has a higher operating frequency of 1.387GHz compared to 1.22GHz which is 8.37% higher than the conventional RC4 circuit. The throughput of the proposed RC4 circuit is found to be 22.192Gbps.

Keywords: *High Throughput, Cipher, Rc4 Stream, 45nm, 1.387GHZ*

1. INTRODUCTION

Stream ciphers are broadly classified into two parts depending on the platform most suited to the implementation; namely software stream ciphers and hardware stream ciphers [1]. RC4 is one of the widely used stream ciphers that is mostly implemented in software. Though several other efficient and secure stream ciphers have been discovered after RC4, it is still the most popular stream cipher algorithm due to its simplicity, ease of implementation[2,3], and speed. The RC4 stream cipher was designed by Ron Rivest for RSA Data Security in 1987. In this paper we study several aspects of the hardware implementation of RC4, with respect to its efficient implementation, and present two new hardware designs which allow fast generation of RC4 key stream.

It uses *S*-box *S*, an array of length *N*, where each location of *S* stores one byte(typically, N = 256). A secret key k of size l bytes isused to scramble this permutation (typically, $5 \leq l \leq 16$).Array K of length N holds the main key, with secret key k repeated as $K[y] = k[y \bmod l]$, for $0 \leq y \leq N - 1$.RC4 has two components, namely the Key Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA). The KSA uses the key K to generate a pseudo-random permutation S of $\{0, 1, \ldots, N-1\}$ and PRGA uses this pseudo-random permutation to generate arbitrary number of pseudo-random key stream bytes [4].

2. RC4 ALGORITHM

RC4 Algorithm

The algorithm uses a series data dependent rotations heavily to We consider the generation of two consecutive values of Z together, for the two consecutive plaintext bytes to be encrypted. Assume that the initial values of the variables i, j and S are $i0$, $j0$ and $S0$, respectively. After the first execution of the PRGA loop, these values will be $i1$, $j1$ and $S1$, respectively and the output byte is $Z1$ [5]. Similarly, after the second execution of the PRGA loop, these will be $i2$, $j2$, $S2$ and $Z2$, respectively. Thus, for the first two loops of execution to complete, we have to perform the operations shown in Table 1.

Table 1: First And Second Iterations Of Prga

First Loop	Second Loop
$i1 = i0 + 1$	$i2 = i1 + 1 = i0 + 2$
$j1 = j0 + S0[i1]$	$j2 = j1 + S1[i2] = j0 + S0[i1] + S1[i2$
Swap $S0[i1] \leftrightarrow S0[j1]$	Swap $S1[i2] \leftrightarrow S1[j2]$
$Z1 = S1[S0[i1] + S0[j1]]$	$Z2 = S2[S1[i2] + S1[j2]]$

Journal of Theoretical and Applied Information Technology
10th February 2014. Vol. 60 No.1
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

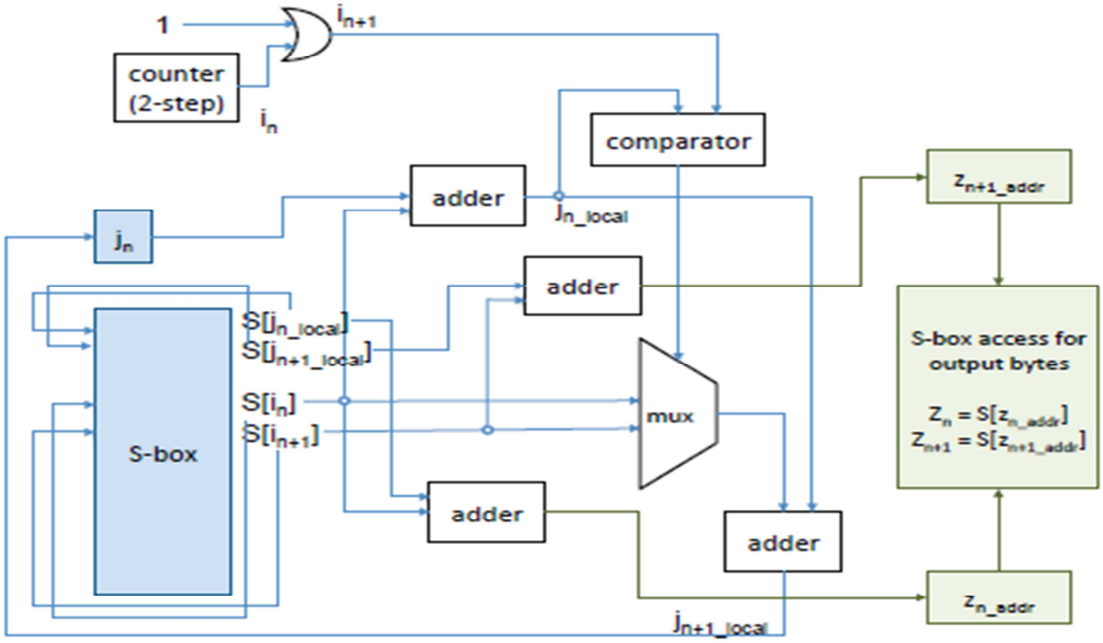

Fig.1 PRGA Circuit Structure For Proposed Architecture

3. PROPOSED RC4 CIRCUIT

3.1 Architecture of PRGA and KSA stage

The schematic diagrams for PRGA and KSA circuits in the proposed design are shown in Fig.1 and Fig.2 respectively. The PRGA circuit operates as per the 2-stage pipeline structure, where the increments of indices take place in the first stage, and so does the double-swap operation for the *S*-box. In the same stage, the addresses for the two consecutive output bytes *Zn* and *Zn*+1 are calculated as the swap does not change the outcomes of the additions *S*[*in*] + *S*[*jn*] or *S*[*in*+1] + *S*[*jn*+1]. In the second stage of the pipeline, the output addresses *zn_addr* and *zn*+1*_addr* are used to read the appropriate keystream bytes from the updated *S*-box [6].The circuit for KSA operates similarly, but has no pipeline feature as the operation happens in a single stage. Here, the increment of indices and swap are done for two consecutive rounds of KSA in a single clock cycle, thereby producing a speed of 2-rounds-per-cycle [7]. Based on this schematic diagram for the circuits, and the port sharing logic, we now attempt the hardware implementation of our new design.Combining our KSA and PRGA architectures, we can obtain 2*N* output streambytes in 2*N* + 259 clock cycles, counting the initial delay of 1 cycle for KSA and 2 cycles for PRGA[8]. The hardware implementation of RC4 described in [9] and [10] provides an output of *N* bytes in 3*N* +768 clock cycles. A formal comparison of the timings is shown in Table 4. One can easily observe that for large *N*, the throughput of our RC4 architecture is 3 times compared to the existing design.

Journal of Theoretical and Applied Information Technology
10th February 2014. Vol. 60 No.1
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

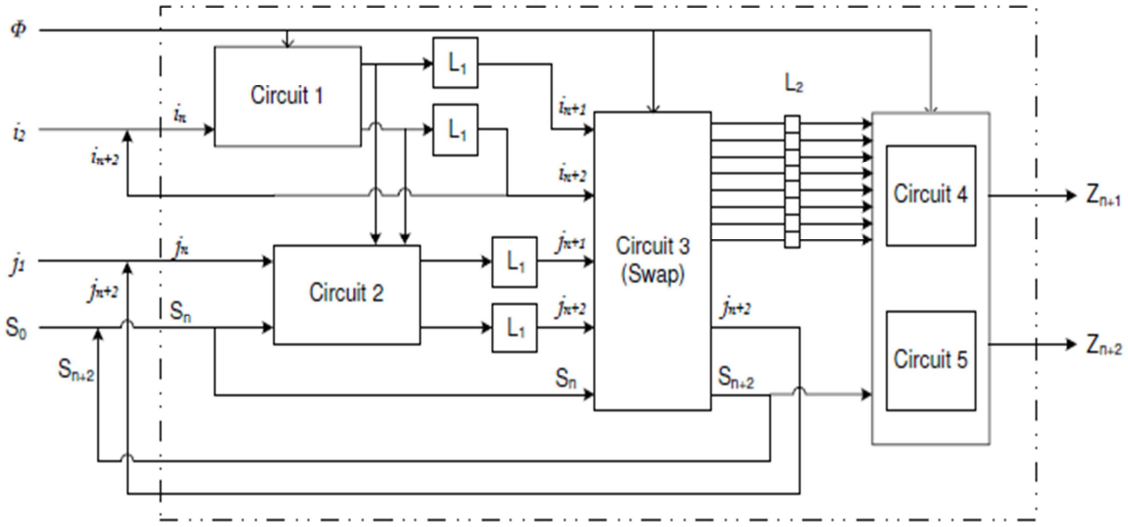

Fig.2 Circuit For KSA Stage Of The Proposed RC4 Architecture

4. IMPLEMENTATION

We have implemented the proposed structure for RC4 stream cipher, using synthesizable VERILOG description. The *S*-register box and *K*-register box are implemented as array of master-slave flip-flops, and are synthesized as standard-cell memory architecture (register-based implementation). The entire Verilog code consists of approximately 900 lines .A major area impact of the circuit originates from the large number of accesses to the *S*-box and the *K*-box from the KSA and PRGA circuit. Since the PRGA and KSA will not run in parallel, we shared the read and write ports of *S*-box and *K*-box between PRGA and KSA. From KSA, 1 read access to *K*-box, 2 read accesses to *S*-box and 2 write accesses to *S*-box are needed. From PRGA, 6 read accesses to *S*-box and 4 write accesses to *S*-box are needed. The 2 read accesses correspond to simultaneous generation of two *Z* values at the last step of PRGA. The 4 read and write accesses correspond to the *double swap* operation. While sharing the mutually exclusive accesses, all the accesses from KSA can be merged amongst the PRGA accesses. Therefore, the total number of read ports to *K*-box is 1, the total number of read ports to *S*-box is 6 and the total number of write ports to *S*-box is 4.

4.1 ISSUES WITH KSA: Note that the general KSA routine runs for 256 iterations to produce the initial permutation of the *S*-box. Moreover, the steps of KSA are quite similar to the steps of PRGA, apart from the following:

• Calculation of *j* involves key *K* along with *S* and *i*.

• Computing *Z*1, *Z*2 is neither required nor advised.

We propose the use of our *loop-unrolled* PRGA architecture for the KSA as well, with some minor modifications, as follows:

1) ***K-register bank:*** Introduce a new register bank for key *K*. It will contain *l* number of 8-bit registers, where $8 \leq l \leq 15$ in practice.

2) ***K-register MUX****:* To read key values *K*[*i*1 mod *l*] and *K*[*i*2 mod *l*] from the *K*-registers, we introduce two 16 to 1 multiplexer unit. The first *l* input lines of this MUX will be fed data from registers *K*[0] to *K*[*l* − 1], and the rest (16 − *l*) inputs can be left floating (recall that $8 \leq l \leq 15$). The control lines of these MUX units will be *i*1 mod *l* and *i*2 mod *l* respectively, and hence the floating inputs will never be selected.

3) ***Modular Counters****:* To obtain modular indices *i*1 mod *l* and *i*2 mod *l*, we incorporate two modular counters (modulo *l*) for the indices. These are synchronous counters and

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the one for *i*2 will have no clock input for the LSB position.

4) ***Extra 2-input Parallel Adders****:* Two 2-input parallel adders are appended to Fig. 2 for adding *K*[*i*1 mod and *K*[*i*2 mod *l*] to *j*1 and *j*2 respectively

5) ***No Outputs:*** Circuits of Fig. 1 and Fig. 2 are removed from the overall structure, so that no output byte is generated during KSA. If any such byte is generated, the key *K* may be compromised. Using this modified hardware configuration, one can implement two rounds of KSA in 2 clock cycles that is "one round per clock", after an initial lag of 1 cycle. Total time required for KSA is 256 + 1 = 257 clock cycles.

*Table **2**: Area report*

Instance	Number of cells	Cell Area	Net Area
Proposed RC4	1171	138352	107

*Table **3:** Power Report*

Instance	Number of cells	Power(nW)	Leakage power(nW)	Total Power(nW)
Proposed RC4	1171	243.147	3820691.969	3820935.115

5. EXPERIMENTAL RESULTS AND ANALYSIS OF THE STRUCTURE

The Verilog code for the proposed RC4 cipher having a throughput of 1 byte per cycle and 2 byte per cycle is written and is synthesized in Cadence 45nm technology and the functional simulation is checked using SIMVISION. Timing analysis was performed for the above proposed designs .Power, area is found out using RTL compiler using 45 nm design technology and it is being tabulated in table 2 and 3. Back end process for the above proposed design was performed using cadence ENCOUNTER. The synthesis results provide us the best throughput for these three designs, obtained by using strict clock period constraints during the implementation

We have optimized and implemented our proposed design using Verilog description, synthesized with 45nm technology. The proposed design has a total area of 138459um^2 and shows a power consumption of 382.0935mW. The proposed circuit has a higher operating frequency of 1.387GHz compared to 1.22GHz which is 8.37% higher than the conventional RC4 circuit. The throughput of the proposed RC4 circuit is found to be 22.192Gbps.

Efficiency of PRGA

The hardware proposed for the PRGA stage of RC4 in proposed design as shown in Fig. 1, produces "one byte per clock" after an initial delay of two clock cycles. Let us call the stage of the PRGA circuit the *nth* stage. This actually denotes the *nth* iteration of our model, which produces the output bytes *Zn*+1 and *Zn*+2. The first block in fig .2 operates at the trailing edge of *φn*, and increments *in* to *in*+1, *in*+2.

During cycle *φn*+1, the combinational part of Circuit operates to produce *jn*+1, *jn*+2. The trailing edge of *φn*+1 releases the latches of type *L*1, and activates the swap circuit. The combinational logic of the swap circuit functions during cycle *φn*+2 and the actual swap operation takes place at the trailing edge of *φn*+2 to produce *Sn*+2 from *Sn*. The combinational logic of these two circuits operate during *φn*+3, and we get the outputs *Zn*+1 and *Zn*+2 at the trailing edge of *φn*+3. This complete block of architecture performs in a cascaded pipeline fashion, as the indices *i*2, *j*2 and the state

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Sn+2 are fed back into the system at the end of $\varphi n+2$ (actually, *in*+2 is fed back at the end of $\varphi n+1$ to allow for the increments at the trailing edge of $\varphi n+2$). The operational gap between two iterations (e.g., *nth* and(*n*+2)*th*) of the system is thus two clock cycles (e.g., φn to $\varphi n+2$), and we obtain two output bytes per iteration. Hence, the PRGA architecture produces 2*N* bytes of output stream in *N* iterations, over 2*N* clock cycles. Note that the initial clock pulse $\varphi 0$ is an extra one, and the production of the output bytes lag the feedback cycle by one clock pulse in every iteration (e.g., $\varphi n+3$ in case of *nth* iteration).Therefore, our model practically produces 2*N* output bytes in 2*N* clock cycles, that is "one byte per clock", after an initial lag of two clock cycles. The performance comparison is made in Table.4 and the complete chip layout obtained after backend at Cadence Encounter is displayed in Fig.3

Table 4: Performance Analysis

Technology (nm)	Design	Max clock frequency(GHz)	KSA (Cycles)	PRGA (bytes/cycle	THROUHPUT (Gbps)
45	Conventional RC4 cipher	1.22	256	1	9.76
45	Design 1	1.37	256	2	21.92
45	Proposed RC4 design	1.387	256	2	22.192

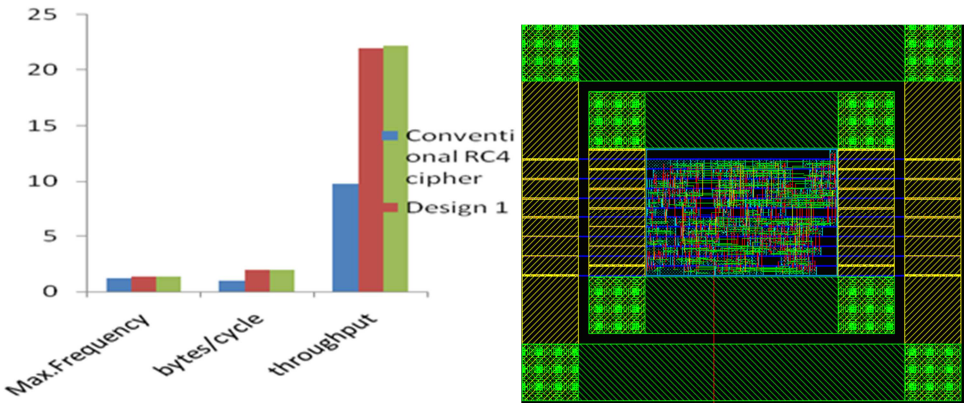

Fig. 3 Performance comparison of RC4 design with complete chip layout.

6. CONCLUSION

In this paper, we present a systematic study of the hardware implementation of RC4, and propose the fastest known architecture for the cipher. We combine the ideas of hardware pipeline and loop unrolling to design an architecture that produces two RC4 key stream bytes per clock cycle. We have optimized and implemented our proposed design using Verilog description, synthesized with 45nm technology. The proposed design has a total area of 138459um^2 and shows a power consumption of 382.0935mW. The proposed circuit has a higher operating frequency of 1.387GHz compared to 1.22GHz which is 8.37% higher than the conventional RC4 circuit. The throughput of the proposed RC4 circuit is found to be 22.192Gbps.

REFERENCES

[1] Software performance results from the e STREAM Project. E STREAM, the ECRYPT Stream Cipher Project.Available at http://www.ecrypt.eu.org/stream/perf/ #results.

[2] The current e STREAM Portfolio. eSTREAM, the ECRYPT StreamCipher Project.(http://www.ecrypt.eu.org/stream/index.html)

[3] S. R. Fluhrer and D. A McGrew. Statistical Analysis of the Alleged RC4 Keystream Generator. FSE 2000, LNCS, Springer-Verlag, Vol.,1978, pp. 19–30, 2000.

[4] S. R. Fluhrer, I. Mantin and A. Shamir. Weaknesses in the Key Scheduling

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Algorithm of RC4. Selected Areas in Cryptography 2001, LNCS, Springer-Verlag, Vol. 2259, pp. 1–24, 2001.

[5] M. D. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos and C. E. Goutis. Comparison of the Hardware Implementation of Stream Ciphers. Int. Arab J. Inf. Tech., Vol. 2, No. 4, pp. 267–274,2005.

[6] J. Golic. Linear statistical weakness of alleged RC4 keystream generator. EUROCRYPT 1997, LNCS, Springer-Verlag, Vol. 1233, pp. 226–238, 1997.

[7] T. Good and M. Benaissa. Hardware Results for Selected Stream Cipher Candidates. eSTREAM, ECRYPT Stream Cipher Project,SASC, Report 2007/023, 2007.

[8] F. K. Gurkaynak, P. Luethi, N. Bernold, R. Blattmann, V. Goode,M. Marghitola, H. Kaeslin, N. Felber and W. Fichtner. Hardware Evaluation of eSTREAM Candidates: Achterbahn, Grain, MICKEY, MOSQUITO, SFINKS, Trivium, VEST, ZK-Crypt. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/015, 2006.

[9] P. Hamalainen, M. Hannikainen, T. Hamalainen and J. Saarinen. Hardware implementation of the improved WEP and RC4 encryptionalgorithms for wireless terminals. In Proc. of Eur. Signal Processing Conf., pp. 2289–2292, 2000.

[10] P. Kitsos, G. Kostopoulos, N. Sklavos and O. Koufopavlou. Hardware Implementation of the RC4 stream Cipher. In Proc. of 46th IEEE Midwest Symposium on Circuits & Systems '03, Cairo, Egypt, 2003.