Military Technical College
Kobry El-Kobba
Cairo, Egypt

12-th International Conference
on
Aerospace Sciences &
Aviation Technology

A NOVEL APPROACH FOR IMPLEMENTING RC4 STREAM CIPHER ALGORITHM

Mohamed Nabil * and Alaa Eldin Rohiem **

ABSTRACT

The implementation of cryptographic algorithms on reconfigurable hardware devices based on Field Programmable Gate Arrays (FPGA) devices is highly attractive. They can run faster than software implementations while preserving the physical security of hardware solutions. Meanwhile, they maintain the flexibility obtained by using those software solutions [1]. In this work, a proposed hardware implementation of RC4 algorithm on field programmable gate arrays (FPGAs) is introduced. The design was described using the hardware description language VHDL (Very high speed integrated circuit hardware description language). The design was implemented on devices from XILINX and we achieved speeds of up to **280 Mbits/s**.

KEYWORDS

Stream ciphers, RC4 algorithm, FPGA, Secure Sockets Layer (SSL), Wired Equivalent Privacy (WEP), VHDL.

* Egyptian Armed Forces.

I. INTRODUCTION

Secret key cryptographic systems can be categorized into either *block* or *stream* ciphers. Block ciphers are memory less algorithms that permute *N*-bit blocks of plaintext data under the influence of the secret key and generate *N*-bit blocks of encrypted data [1].
Stream ciphers contain internal states and typically operate serially by generating a stream of pseudorandom key bits, the key stream (stream ciphers are also called key stream generators)[2]. The key stream is then bitwise XORed with the data to encrypt/decrypt. Stream ciphers do not suffer from the error propagation, as in the block ones, because each bit is independently encrypted/decrypted from any other.They are generally much faster than block ciphers and they have greater software efficiency [2]. One of the most widely used stream ciphers is the RC4 stream cipher. It was designed by R. Rivest in 1987[3]. Due to its remarkable simplicity, RC4 cipher is used in software applications such as Secure Sockets Layer (SSL) (to protect Internet traffic) and Wired Equivalent Privacy (WEP) (to secure wireless networks)[3]. RC4 is extremely fast and simple, which makes it ideal for real time data transmission.
The paper is organized as follows:

1) Description of RC4 algorithm.
2) FPGA design cycle.
3) RC4 design methodology.
4) Results and performance evaluation.
5) Conclusions**.**
6) References.

II. DESCRIPTION OF RC4 ALGORITHM:

RC4 is a variable-length key of length from 1 to 256 bytes which 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, a byte *k* is generated from S by selecting one of the 255 entries in a systematic fashion as shown in Fig. 1 As byte *k* is generated, the entries in *S* are once again permuted [2] to generate the next value *k* and so on. The algorithm works in two phases which are initialization of s-box and stream generation.

II. 1 S_BOX Initialization

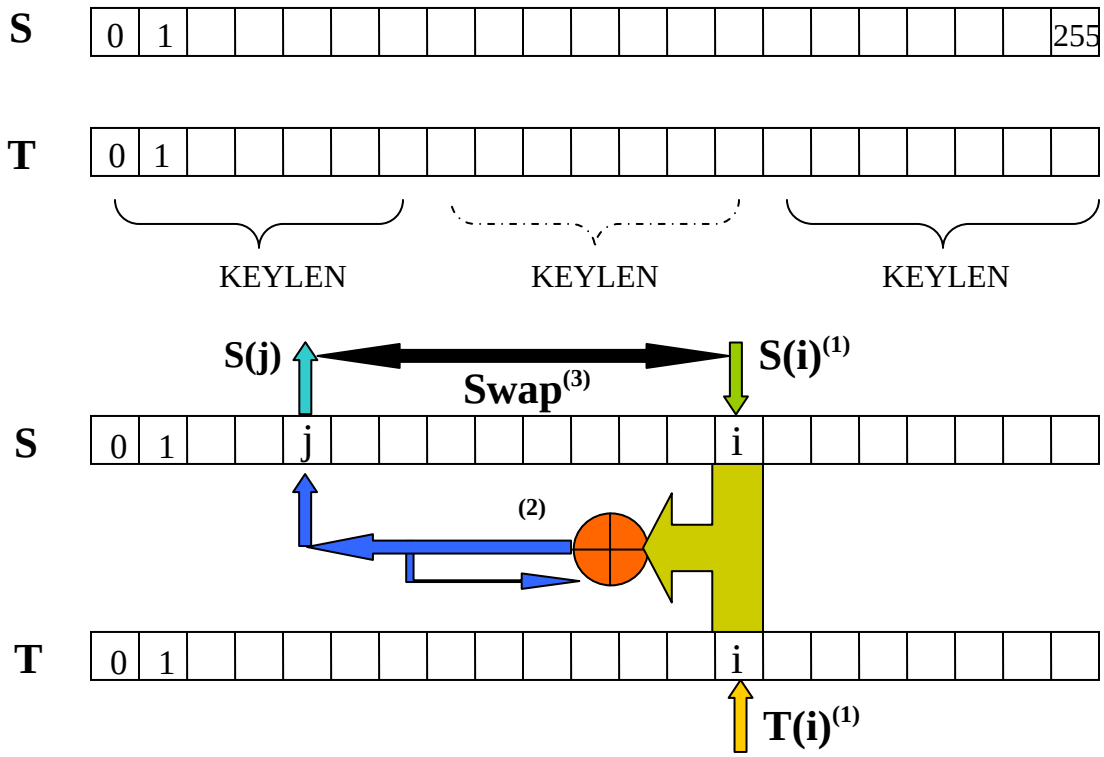
At the beginning, the entries of **S_box** are set equal to the values from 0 through 255 in ascending order; that is; S[0] = 0, S[1] = 1,..., S[255] = 255. A temporary vector T is also created. If the length of the key K is 256 bytes, then *K* is transferred to T. Otherwise, for a key of length *(key len)* bytes, the first *(keylen)* elements of T are copied from *K* and then *K* is repeated as many times as necessary to fill out T. These preliminary operations can be summarized as follows [2]:

```
/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod [keylen];
```

Next, we use T to produce the initial permutation of *S*. This involves starting with S[0] and going through to S[255], and, for each S[i], swapping S[i] with another byte in S according to a scheme dictated by T[i]:

```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
j = (j + S[i] + T[i]) mod 256;
Swap (S[i], S[j]);
```

(1) Initialization Phase



(2) Stream Generation Phase

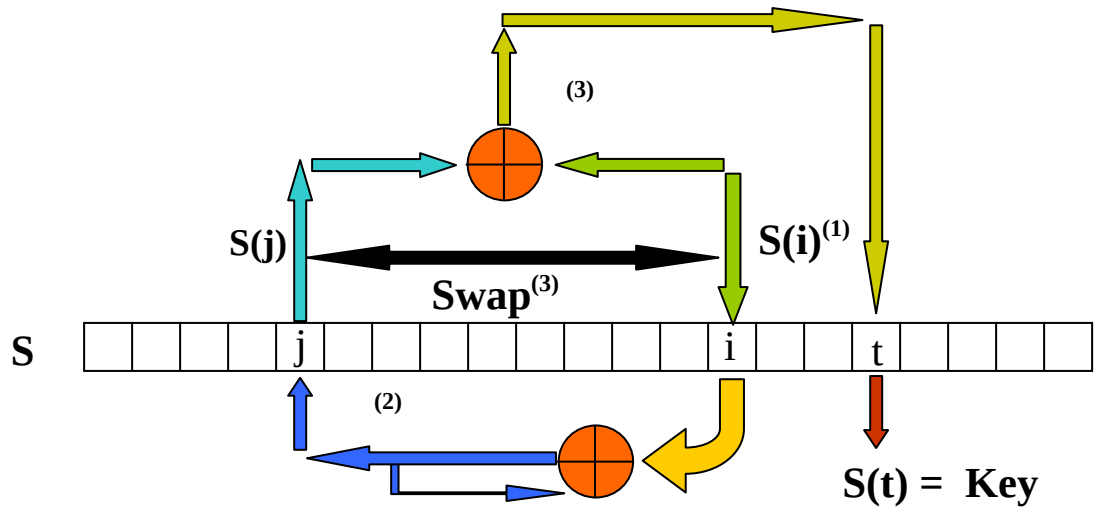


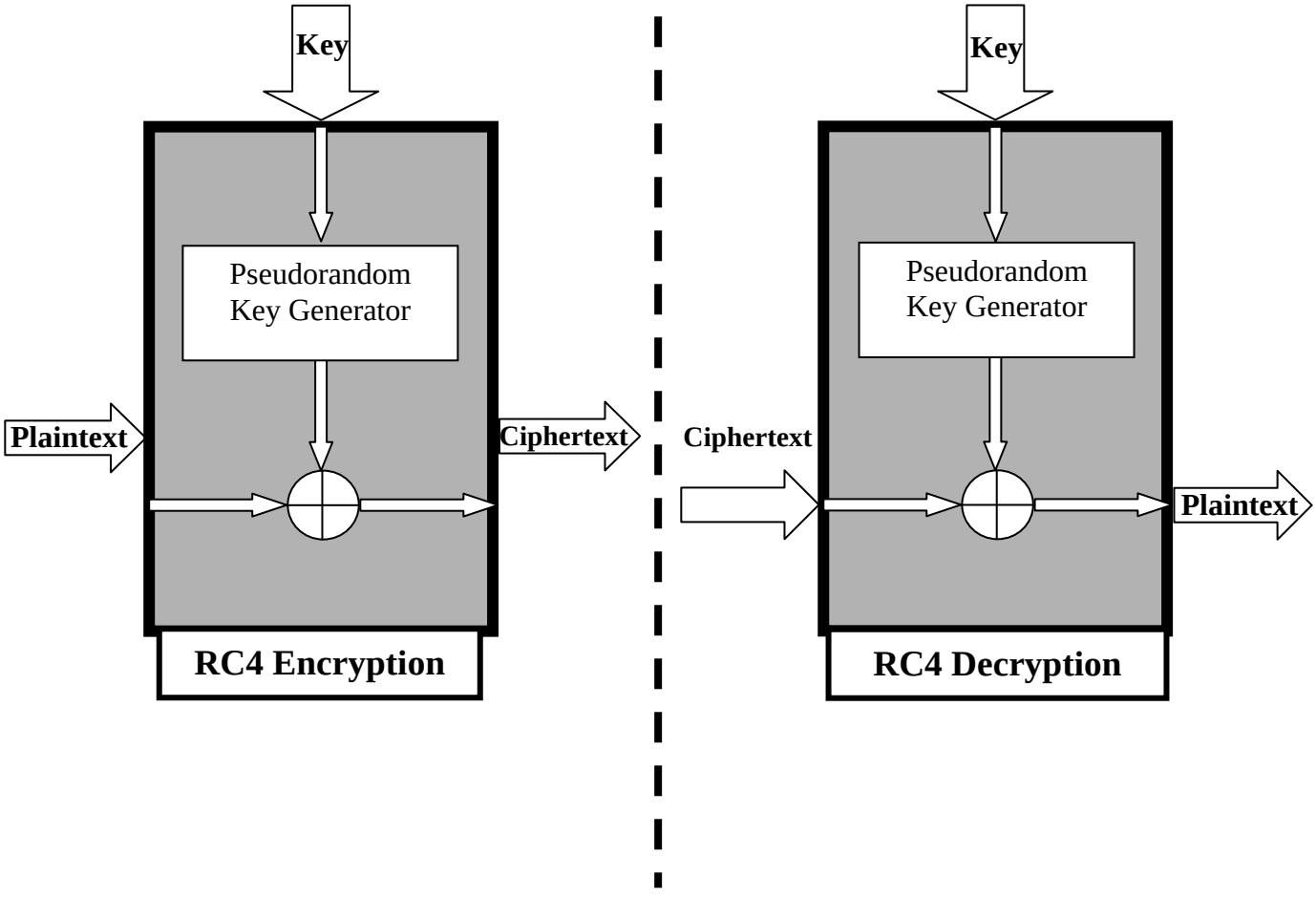
Fig. 1. RC4 Initialization and stream generation

II.2 Stream Generation :

Once the S vector is initialized, the input key is no longer used. Stream generation involves cycling through all the elements of S[i], and, for each S[i], swapping S[i] with another byte in S according to a scheme dictated by the current configuration of S [2]. After S[255] is reached, the process continues, starting over again at S[0]:

```
/* Stream Generation */
i, j = 0;
while (true)
i = (i + 1) mod 256;
j = (j + S[i]) mod 256;
Swap (S[i], S[j]);
t = (S[i] + S[j]) mod 256;
k = S[t];
```

To encrypt, XOR the value *k* with the next byte of plaintext. To decrypt, XOR the value *k* with the next byte of ciphertext. Fig. 2 illustrates this operation.



II. 2 Stream Generation

Once the S vector is initialized, the input key is no longer used. Stream generation involves cycling through all the elements of S[i], and for each S[i], swapping S[i] with another byte in S according to a scheme dictated by the current configuration of S [2]. After S[255] is reached, the process continues, starting over again at S[0] These operations can be summarized as follows [2]:

```
/* Stream Generation */
i, j = 0;
while (true)
i = (i + 1) mod 256;
j = (j + S[i]) mod 256;
Swap (S[i], S[j]);
t = (S[i] + S[j]) mod 256;
k = S[t];
```

The plaintext is encrypted by XOR the value k with the next byte of plaintext. The ciphertext is decrypted by XOR the value k with the next byte of ciphertext as shown in Fig. (2).

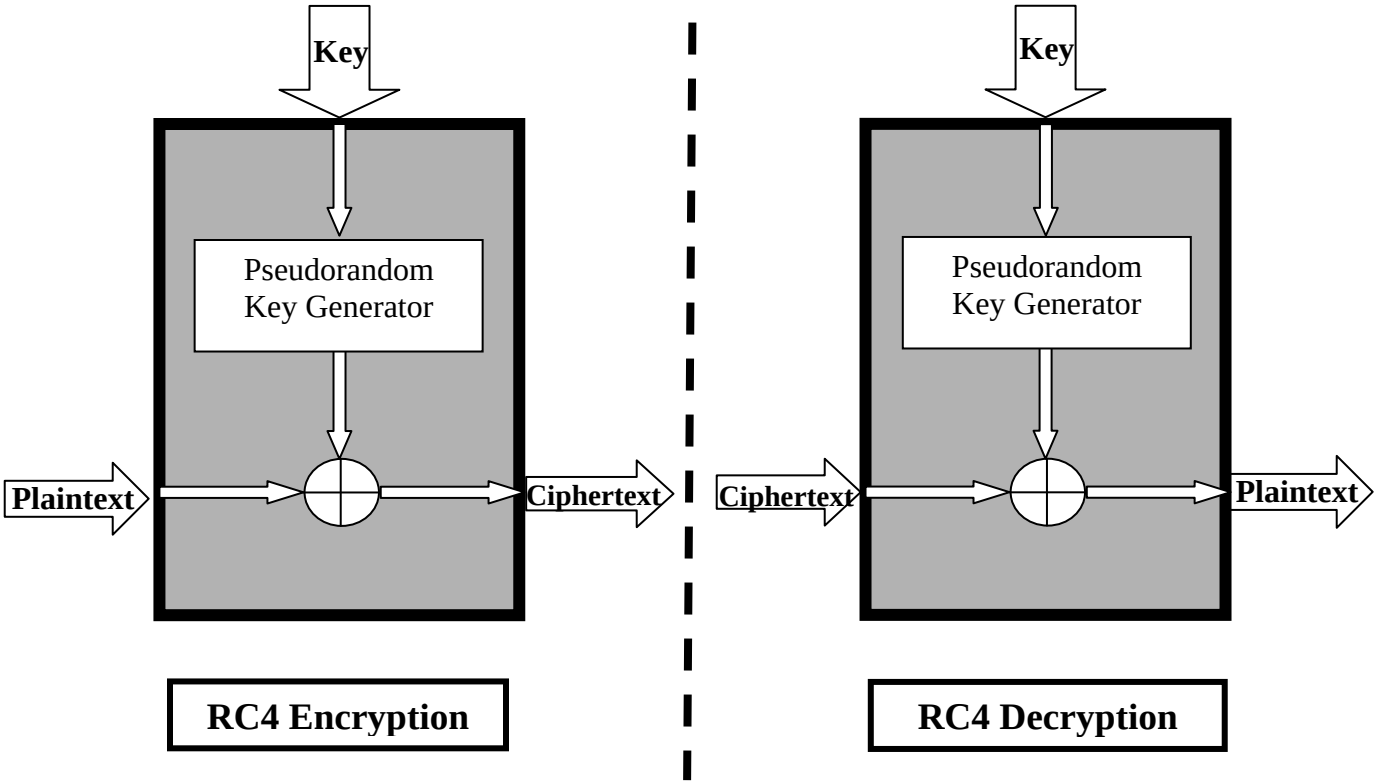


Fig. 2. RC4 Ciphertext and plaintext generation

III. RC4 DESIGN METHODOLOGY

III. 1 RC4 Design:

RC4 algorithm is broken down the into small elementary computational units. A ROM (with 256 byte length and 8 bit wide) is used to hold the key bytes (K_box), which must be equally distributed to fill out the whole ROM. Three Block RAMs of the parameterized modules of Xilinx memory chips were constructed to hold the transaction and swapping of data imposed by the algorithm. A 8 bit counter is used for the addressing purposes for the key ROM and the other 3 block rams during initialization and stream generation phases. Multiplexers, they are used to switch the data and the address buses during the aforementioned phases and were implemented using the combinational logic.
Adders, two adders were constructed using the VHDL code to achieve the addition operation required by the algorithm. Registers: they are implemented using the parameterized registers available on the FPGA resources; to be used is during the swapping operation. Control logic is to control the timing, (appropriate clocks) and control signals for RC4 algorithm during the two phases of the algorithm. The control logic unit is designed by mapping the timing for the RC4 algorithm into two ROMs, one ROM for each phase.

III. 2 Logic Operation

III.2.1 Initialization phase :

The first phase needs three clock cycles per iteration. The S-box consists of three 256 bytes RAM blocks as shown in Fig.3. The S-boxs is linearly filled, such as S0 = 0, S1 = 1, S2 = 2,...,S255 = 255. Each RAM block has five inputs clock signal, read signal, write signal, address and data 8-bit buses, one 8-bit output which *Si* and *Sj* for the first two RAM blocks and the key is output of the last one.
The logic construction for the RC4 algorithm is described as shown in Fig..4.
At the first clock cycle, the value of 8-bit counter is used to address the first RAM block. The value of *Si* (stored in the *Si_register*) is used for the computation of the new value of *j* as it is shown in Fig.4. The j-adder is used for the computation of the new value of *j*. The j-adder accepts as input the values of *Ki* and *Si*. At the second clock cycle, the new produced value of j is used as an address for the second RAM block. The stored value in this address is temporarily stored in the *Sj_register*. At the third cycle, the contents of the *Si_register* and *Sj_register* are written at the *j* and *i* addresses, respectively. With this procedure, the swapping is achieved.
The first phase needs three clock cycles per iteration. Therefore, the total time that is required in the key set-up phase is ***256*3 = 768*** clock cycles.

III.2.2 Stream generation phase

The same hardware is being re-used. The difference in this phase is that the values of the K-box are not used. After the completion of the first phase, the multiplexer in Fig..4 selects the zero value input. In meantime, the *j_register and i_register* are initialized to zero so that to be ready for the second phase. After the two aforementioned actions, the procedure of key stream generation can begin. The operations at the first three steps are similar to those of the key set-up phase except that the S-box is already initialized. At the *first* step, the value of *i* is used as address in the first RAM block and the value of *Si* is stored in the *Si_register*. In addition, the new value of *j* is computed. At the *second* step, the new value of *j* is used as address of the second RAM block and the value of *Sj* is stored in the *Sj_register*. In this step, the values of *Si* and *Sj* are being added in the *t-adder* and the result of the addition is stored in the *St_register*. At the *third* step, the contents of the *Si_register* and *Sj_register* are written at the *j* and *i* address, respectively, and the value of the *t_register* is being used as address for the third RAM block. Therefore, the value of *St* is also produced in the third step. This value of *St* is the generated key stream byte.
After the completion of this phase, each byte in the key stream can be generated and used for encryption/decryption. The bitwise XORing of the key stream with the plaintext/ciphertext achieves the encryption/decryption.

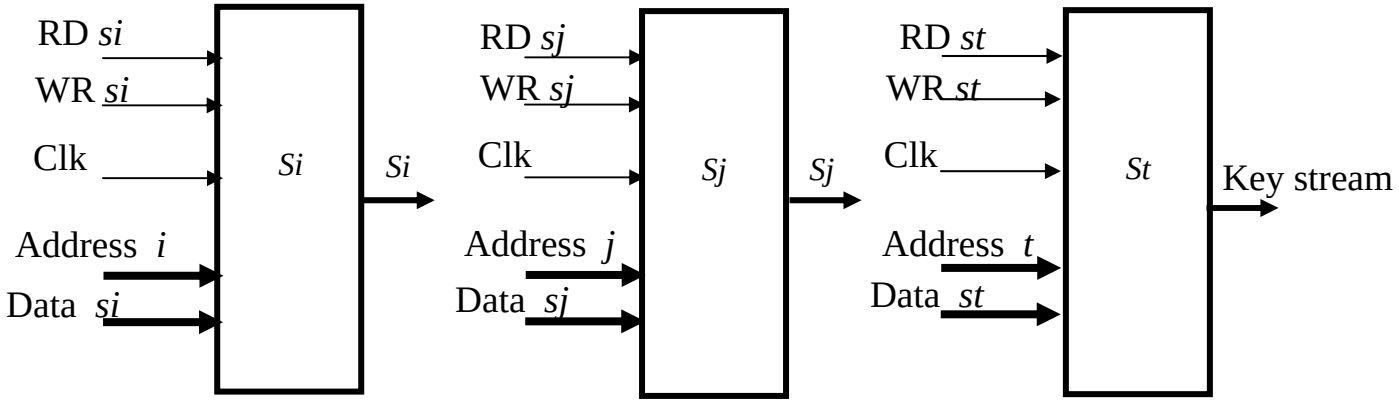


Fig. 3. S_BOX

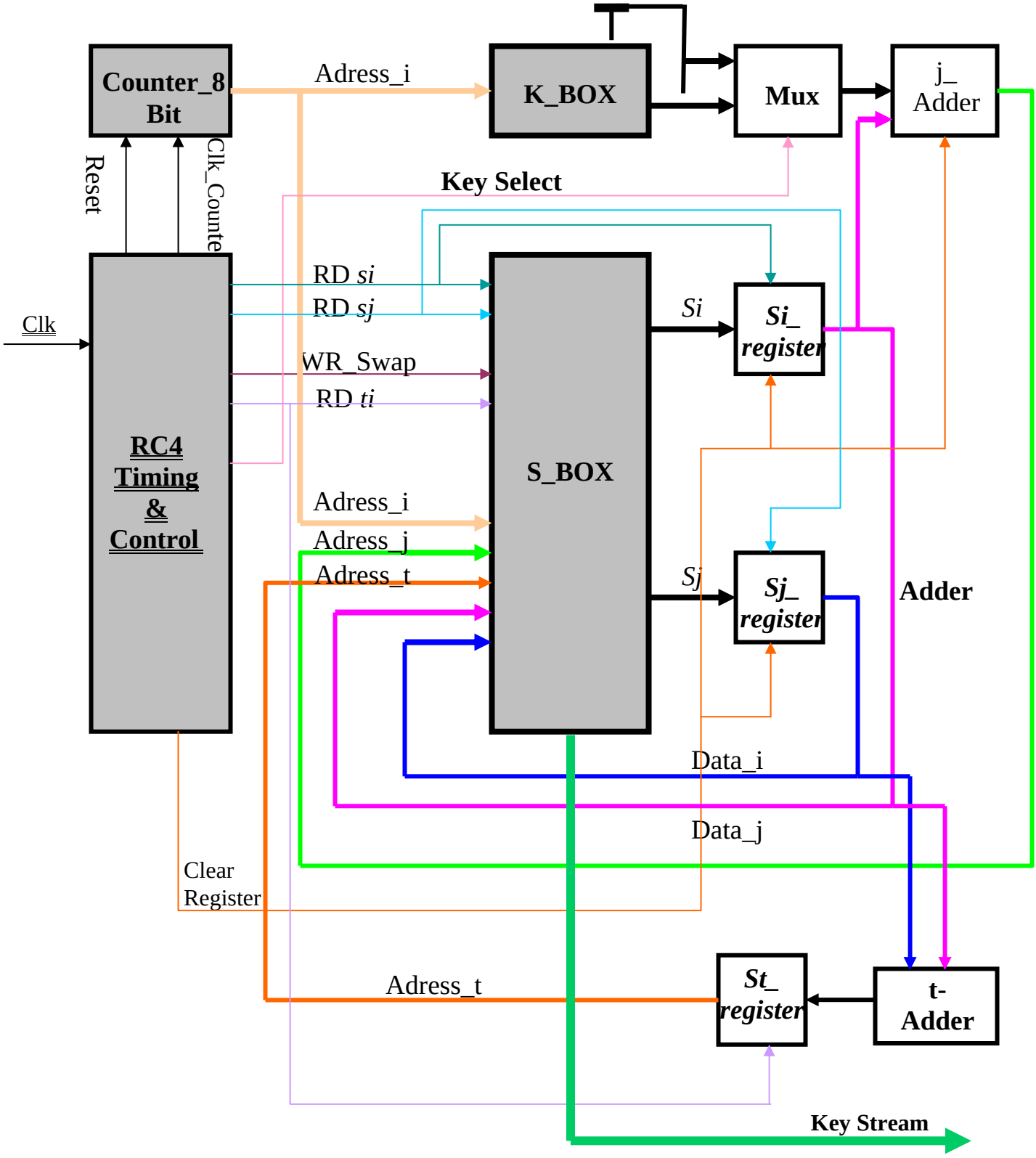

Counter_8 Bit
Adress_i
K_BOX
Mux
j_ Adder
Reset
Clk_Counte
Key Select
RD si
RD sj
Si
Si_ register
Clk
WR_Swap
RD ti
RC4 Timing & Control
S_BOX
Adress_i
Adress_j
Adress_t
Sj
Sj_ register
Adder
Data_i
Data_j
Clear Register
Adress_t
St_ register
t- Adder
Key Stream

Fig. 4. RC4 logic.

IV. SIMULATION AND HARDWARE IMPLEMENTATION RESULTS

Fig. 5 shows the simulation results for the proposed RC4 design. The clock signal used in the proposed algorithm is 104.921 MHZ. The signal names are identical to those shown in Fig. [3, 4].The signal (*rd_s_i*) is responsible for the reading the value of *Si* which is stored in the *Si_register*. The signal (*rd_s_i*) is responsible for the reading the value of *Sj* which is stored in the *Sj_register*. Signal wr_swap is responsible for swapping the content of both *i* and *j* RAMs. These signals are generated during the initialization phase (which continues 255*3=765 cycle) and they continue during the Stream Generation Phase. After the 765 cycle, a clr_latch signal is generated to clear the *i* and *j* registers. In the stream cipher phase the same three signals are still generated in addition to (*rd_t_i*) signal which is responsible for generation of the key. All these signals are generated by mapping these timing for the aforementioned four signals into EPROM, which gives optimized hardware for generation for the timing of the RC4 algorithm.

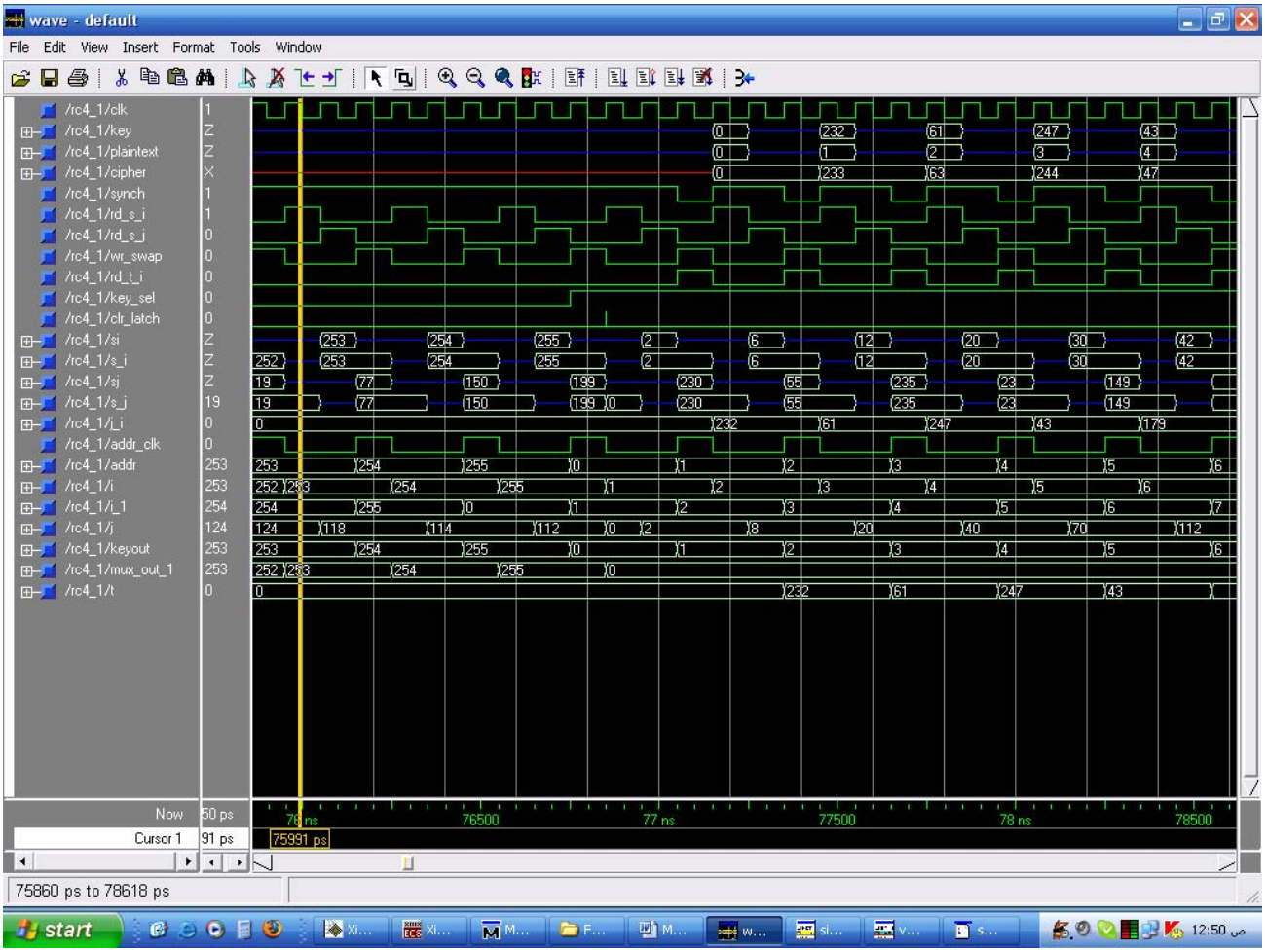


Fig. 5. RC4 simulation results

The algorithm is implemented on a XILINX SPARTAN II (2s200pq208-5) . ISE5.1 synthesis tool (XST) for synthesis and placing and routing the design.
The performance (in terms of throughput) and of consumed area (in terms of FPGA CLB slices). The throughput attained is ***104.921* (8/3) = 279.79*** Mbit / sec.
For the implemented stream ciphers, are presented in Table 1. All the designs were synthesized on a Xilinx SPARTAN II. From table 1,it is clear that implementing the S_BOXes on the RAM blocks elements reduces greatly the amount of gate count on the device. Further, it enhances the performance of the RC4 algorithm. Since lock RAM has the advantage of being SRAM memories structure and does not have a combinational paths problem, which means high performance for getting out the S_BOXes values when they are called.

Table1. Device Utilization Summery After Placing And Routing

Attributes	Used	Percentage
Number of SLICEs	431 out of 2352	18%
Number of 4_input LUTs	995 out of 4704	21%
Number of Flip Flops	159 out of 4704	3%
Number of BLOCK_RAMs	3 out of 14	21%

The results for the proposed RC4 algorithm design are compared with the results given in [5, 6]. The results give that the proposed design is faster then the results introduced in [5,6]. Our proposed design is implemented on XILINX SPARTAN II chips where they are low cost comparing with Xilinx Virtex-IITM 2V250FG256 FPGA used in [5].

V. CONCLUSION

This paper presents a hardware implementation for one of the most powerful stream ciphers, that is known as RC4. In this work, we exploited the advantages afforded by the FGPAs (*e.g*. flexibility, logic resources, routing resources and low cost) to implement this design, using the available block rams resources on those FGPAs to implement the S_ BOXs of the RC4 algorithm to accelerate the calculations of cipher. The achieved results gave throughput better then [5, 6] and the other known results. Besides, the usage of low cost chips is favorable to implement such powerful algorithms.

REFERENCES

[1] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. “Handbook of Applied Cryptography”. CRC Press, 1996.

[2] Prentice Hall. Stallings W. Cryptography and Network Security - Principles and Practices (4th Ed.

[3]

[4] Bruce Schneier, "Applied Cryptography", 2nd Edition, Wiley, 1996.

[5] Spartan-II2.5V FPGA Family Introduction and Ordering Information. An application note ds001_1 introduced by Xilinx. M.Galanis, P.Kitsos, G.Kostopoulos, Nicolas Sklavos, and Costas Goutis Electrical and Computer Engineering Department, University of Patras. Greece. The International Arab Journal of Information Technology, Vol. 2, No. 4, October 2005.

[6] Hamalainen P., Hännikäinen M., Hamalainen T., and Saar J., “Hardware Implementation of the Improved WEP and RC4 Encryption Algorithms for Wireless Terminals,” *in Proceedings European Signal Processing Conference (EUSIPCO)*, Tampere, Finland, pp. 2289-2292, September 2000.

[7]

[8] R. Stinson, Cryptography: Theory and Practice, CRC Press, Boca Raton, 1995. Kundarewich P. D., Wilton S. J. E., and Hu A. J, “A CPLD-Based RC4 Cracking System,” *in Proceedings of the Canadian Conference on Electrical and Computer Engineering*, May 1999.