Journal of Engineering and Applied Sciences 13 (8): 1974-1979, 2018
ISSN: 1816-949X

An FPGA Implementation of the Serpent Algorithm using Xilinx System Generator

M.F. Al-Gailani
Information Engineering College, Al Nahrain University, Baghdad, Iraq

Abstract: Serpent cipher had been designed as an alternative for the previous encryption standard algorithm, the data encryption standard. It had been chosen within the shortlist by the National Institute of Standard and Technology which eventually selected Rijndael cipher as an Advanced Encryption Standard. There is no doubt on the security of Serpent, however because of the high number of rounds it has slower implementation speed compared to the chosen algorithm. All over, it is important to design an efficient hardware implementation for a well-known algorithm as an option in case the current standard is being attacked. This study presents an FPGA implementation of the Serpent algorithm, the design is focused on the area rather than throughput thus iterative looping architecture is suggested. It is implemented on the target device Xilinx Virtex-6 xc6vlx195t-3ff1156 using ISE design suite 14.7. The design achieved a maximum frequency of 111.757 MHz providing 0.447 Gbit/sec of throughput.

Key words: FPGA, Serpent cipher, SP-network, throughput, TPS, design

INTRODUCTION

Serpent algorithm has been designed by Anderson *et al.* (1998) and Biham *et al.* (1998). Their product was a response to the announcement of the US National Institute of Standards and Technology (NIST) for developing a new standard called Advanced Encryption Standard (AES) to replace the DES (FIPS, 2001). DES was developed by the IBM and adopted in 1977 as a standard for non-classified applications (Standard, 1977). The algorithm was very secure and efficient, however because of the rapid technological development its key length and block size become inadequate to achieve the required level of security. NIST set conditions on applicant's algorithm such that the new algorithm should be at least secure as Triple DES and fast as DES. The block size of the algorithm must be 128 bit as well the key length in addition the key must supports 192 bit and 256 bit length.

Serpent had been designed based on these requirements and its non-linear part which is the heart of any algorithm was designed based on using the S-boxes of DES in a new structure. These substitution boxes were extensively analyzed and it had been proven that they are secure against attacks including both differential (Biham and Shamir, 1991, 1993) and linear (Matsui, 1993) cryptanalysis and their related techniques.

In this study, an efficient hardware architecture of Serpent algorithm is designed and implemented based on FPGA platform, full details are presented in the following sections.

Table 1:Multiple architectures implementation of serpent

Architecture	CLB slices	Throughput (Mbps)
Iterative looping	5511	62
Iterative looping with partial loop unrolling	7964	444
Full loop unrolling	8103	312
Full pipelined	9004	1029

Table 2: Different serpent architectures implementation

Architecture	No. of slices	No. of LUTs	No. of Ffs	Throughput (Mbps)
Combinational	9999	18756	0	854
Cipher-only	11897	22680	4224	19692
Fully pipelined	14750	24331	16768	17534

Literature review: Elbirt *et al.* (2001) and Elbirt and Paar (2000) researchers had considered multiple architectures of Serpent. Their designs were implemented on Xilinx XCV1000 FPGA board. The details of implementation are shown in Table 1.

Sugier proposed three architectures which were implemented in Xilinx Spartan-3E XC3S1600E-5 device. First architecture is called combinational, the second is cipher-only pipelined and the last is fully pipelined architecture. There resource utilization and throughput are illustrated in Table 2.

Lazaro *et al.* (2004), a fully pipelined architecture was proposed and implemented at Xilinx Virtex-II X2C2000-6 FPGA device. It has been shown that the device can run at a throughput of 40 Gb/sec.

Najafi *et al.* (2004), a higher performance in terms of speed and area is achieved using a partial reconfiguration in the implementation. The design is fully pipelined from inside and outside for each round which was implemented

at Virtex XCV1000-6 FPGA device. It achieved a throughput of 22 Gbps with 7504 slices of an occupied area.

Damaj *et al*. proposed a parallel reconfigurable hardware implementations using RC-1000 a static reconfigurable system from Xilinx Virtex-E FPGA and MorphoSys dynamic reconfigurable computer. There suggested design runs at speed of 12.21 Mbps and occupies an area of 19198 slices

Patterson (2000) has proposed a dynamic implementation of Serpent in a Virtex FPGA XCV400-4 using Jbits which provides a run-time modification of the configuration bitstream. A throughput of 10 Gbits/sec is achieved through this design, noting that the subkeys are calculated in software and used as constants in the proposed design.

MATERIALS AND METHODS

Serpent cipher: Serpent cipher is a 128 bit iterated block cipher based on SP-network. The cipher starts by initial permutation followed by 32 rounds and ends up with a final permutation, the flowchart of the cipher is shown in Fig. 1. The final permutation is the inverse of the initial permutation. The usage of these two layers have no security significance, they are used to simplify the implementation. Each round except the final consists of key mixing operation, substitution boxes and a liner transformation. The last round has the same structure, however an additional key mixing operation is used instead of the linear transformation. In the key mixing operation, the intermediate data is XORed with the 128 bit subkey that is generated by the key schedule algorithm. The 128 bit combination is then passed to the S-boxes layer. Each round has a different (4×4) S-box which is duplicated 32 times and implemented in parallel. In the final stage (liner transformation) the 128 bit (four 32 bit words) is linearly mixed up with a number of operations based on rotation (<<<), shift (<<) with different offset and XOR (⊕) as shown in Fig. 2.

As the cipher is based on SP-network, the decryption algorithm is not the same as the encryption algorithm in that an inverse of the S-boxes as well as the inverse linear transformation (Fig. 3) have to be used instead. In addition, the subkeys have to be used in reverse order.

The cipher requires 33 128 bit subkeys, one subkey for each round and an extra one for the last round, these are equivalent to 132 32 bit words. A 132 32 bit prekey words are first generated by applying Eq. 1. Where the eight 32 bit words w-8, ..., w-1 represent the user supplied 256 bit key and Ø is the fractional part of the golden ratio $(\sqrt{5+1}/2)$ or 0×9e3779b9 in hexadecimal:

$$wi = (wi\text{-}8 \oplus wi\text{-}5 \oplus wi\text{-}3 \oplus wi\text{-}1 \oplus \oplus i) <<< 11 \quad (1)$$

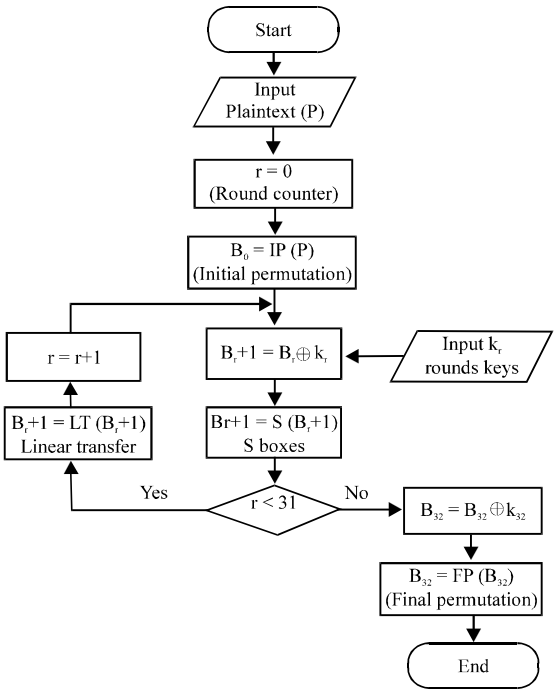

Fig. 1: Cipher flowchart

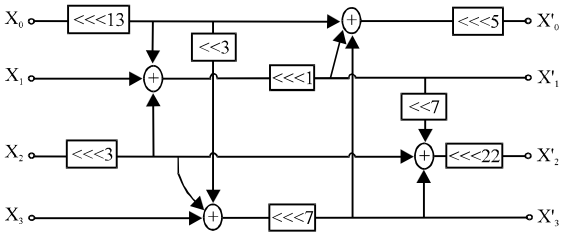

Fig. 2: Forward linear transformation

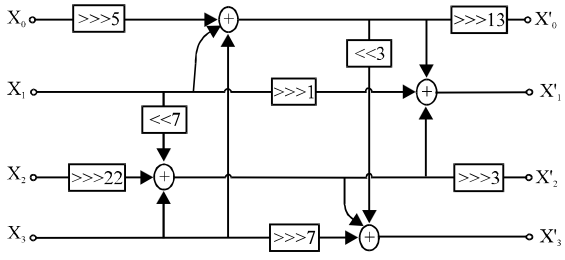

Fig. 3: Backward linear transformation

After that the generated prekey words are transformed into subkey words by passing them through the S-boxes in some predefined order and rearranged the results by applying the initial permutation.

Algorithm implementation: The cipher is designed and implemented based on Fields Programmable Gate Array

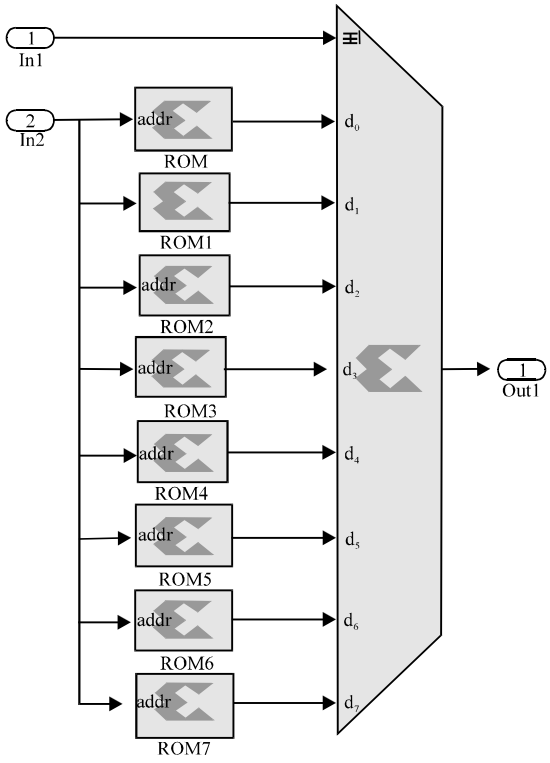

Fig. 4: Conceptual structure of an FPGA device (Chu, 2008)

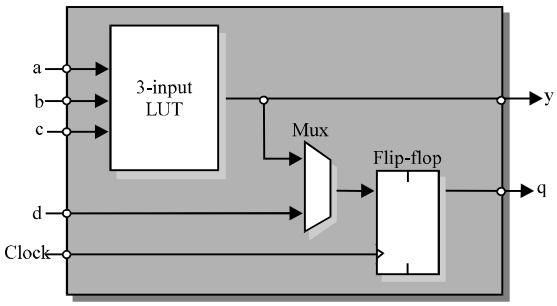

Fig. 5: The key elements forming a simple programmable logic block (Maxfield, 2004)

(FPGA) platform. This technology is chosen because of its high potential and flexibility which delivers the proposed model within the planned schedule time and cost, unlike non-reconfigurable devices such as Application-Specific Integrated Circuit (ASIC) which has less flexibility, more time consuming in the design and extra cost. FPGA is a logic device that contains lots of configurable logic cells and programmable switches. It is arranged in two-dimensional array as shown in Fig. 4 (Chu, 2008).

The logic cells are meanly comprised of Look-Up Tables (LUT) and flip-flops (Fig. 5) (Maxfield, 2004),

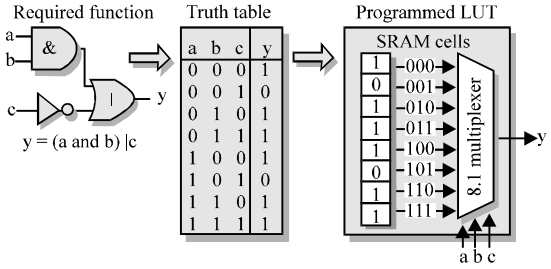

Fig. 6: Configuring a LUT (Maxfield, 2004)

look-up tables are configured as distributed RAM, shift registers or combinational logic (Fig. 6) (Maxfield, 2004). The programmable switches are configured to provide interconnections among the logic elements. In addition, the device also contains block RAMs, high speed input/output interface pins and clock management.

The Xilinx Virtex XC6VLX130T-3FF1156C FPGA board is chosen as a target device. This reconfigurable device has sufficient hardware resources to implement the cipher with optimal performance. It has 128 K logic cells, 480 DSP48E1 slices, 9504 kB block RAM (each store 36 kB), 20 GTX transceiver, 600 user input/output pins and its Configurable Logic Blocks (CLBs) consist of 20 k slices and 1740 kB distributed RAM.

The hardware design of an algorithm can have different architectures depending on the application and availability of resources. It can be categorized as iterative looping, loop unrolling, external pipeline (partial, full), internal pipeline (partial, full), parallel and hybrid architectures.

The focus of the proposed design was on the area rather than throughput thus iterative looping architecture is considered. For 32 round system, only one round is designed (unrolled) and the entire process is attained by iterating the data 32 times. Thus, it is an efficient approach for minimizing the utilized hardware. However, in contrast the number of clock cycles is increased, it will be in optimal situations equal to the number of rounds as the case with the proposed design.

The design can be created using Hardware Description Language (HDL) such as VHDL or Verilog, High Level language such as SystemC, Handel-C, Catapult C, Impulse C or high level graphical design entry such as Xilinx system generator or LabVIEW.

The system is designed schematically using Xilinx system generator. All elements of the round function and key generation are implemented as a combinational logic. The initial and final permutation are implemented by first converting the 16 bit inputs into 128 bit using concat and slice Xilinx blocks, then the outputs are

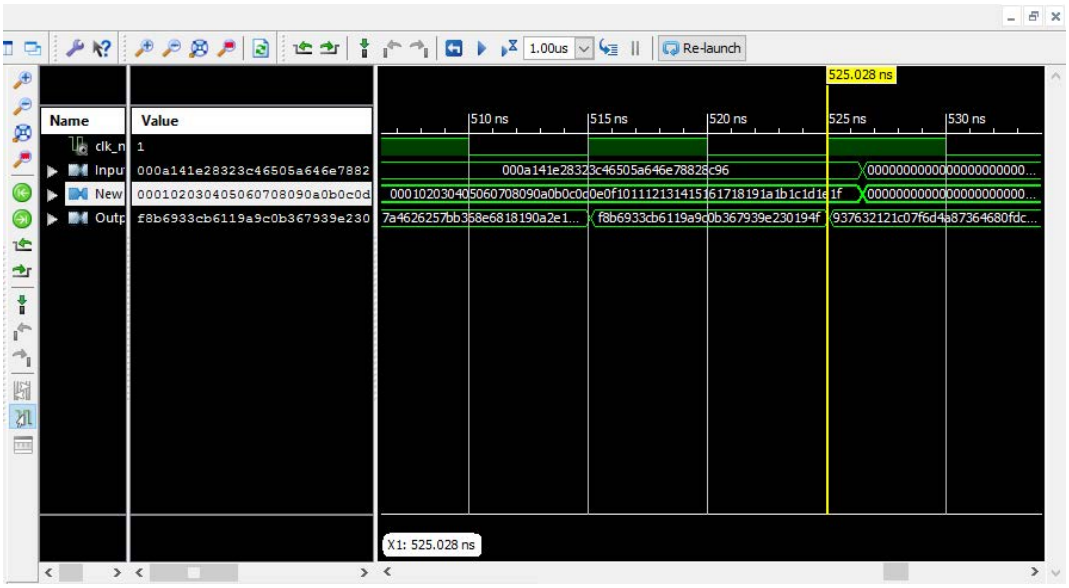

Fig. 7: Simulation result

hardwired according to the specified tables (Biham *et al*., 1998) and finally the results are converted back to 16 bit using the same blocks.

The substitution boxes are implemented using distributed memories. Each ROM is configured by setting up the depth to 16 and initialized the initial value vector with the S-box values. The cipher uses eight different S-boxes. Each S-box is used four times every 8-round which is duplicated 32 times to be implemented in parallel. As shown in Fig. 7, the selection between the different S-boxes (ROMs) is achieved through the multiplexer which is controlled by the select line that is connected to a counter that resets every eight cycles.

In the linear transformation three different operations are used. First, the circular shift which is implemented using Xilinx BitBasher, for example to rotates the byte 3 bit to the left the BitBasher is configured as a = {b[28:0], b[31:29]}. Next, the shift operation which is realized using Xilinx Shift block and the XOR operation like the key mixing operation using Xilinx Bitwise expression.

Performance evaluation: To evaluate the performance of the design the throughput of the implementations is calculated according to Eq. 2:

$$\text{Throughput} = \frac{\text{Number of percentage bit} \times \text{clock frequency}}{\text{Number of clocks cycles per encrypted block}} \quad (2)$$

Where the number of processed bits is equal to the block length which is 128 bit and the number of clock cycles per encryption block is 32 cycles one for each round. The clock frequency which is the maximum operating frequency in addition to the other measurements such as the number of utilized hardware resources and the power consumption of the implementation are obtained from the Xilinx Integrated Synthesis Environment (ISE) report files.

In addition to the throughput, a Throughput Per Slice (TPS) is also considered which is calculated according to Eq. 3. This metric gives an indicator for the hardware resource cost. For optimum implementation the designer seeks to raise the throughput in addition to the TPS:

$$\text{TPS} = \text{Throughput} / \text{Number of occupied slices} \quad (3)$$

RESULTS AND DISCUSSION

Serpent algorithm is designed schematically by the use of the Xilinx system generator which has an extensive library of variety of block sets that is installed in the MATLAB environment. During the system generator, a netlist file for the proposed algorithm is generated which is then synthesized and simulated through the use of the Xilinx ISE design suite 14.7 and ModelSim, respectively. The results of the implementation of the algorithm are summarized in Table 3.

The design is focused on the applications that require limited resources, thus the emphasis was on the amount of resources occupied by the design rather than the throughput. Accordingly, iterative looping architecture is designed and implemented, otherwise

Table 3:Implementation results summary

Target device	xc6vlx130t-3ff1156
Device utilization summary	
Slice logic utilization	Used
No. of slice Registers	392
No. of slice LUTs	1,552
No. of occupied slices	512
No. of bonded IOBs	513
Maximum frequency	111.757 MHz
Throughput	0.447 Gbps
TPS	0.873 Mbps/Slice
Supply power (W)	
Dynamic	0.657
Quiescent	3.089
Total	3.746

otherarchitectures such as loop unrolling can be suggested. For the purpose of obtaining rapid and efficient implementation as well as obtaining satisfactory results all inputs and intermediate data are processed simultaneously in parallel at the same time, achieving an optimal implementation of one clock cycle per round. The design has efficient results compared to a similar architecture regarding logic area, throughput and power consumption. However, for a fair comparison between the obtained results and the others, all designs that compare must have the same architecture and carried out in the same device (family and type) as different devices produce different timing which in turn vary based on the availability of logics and routing resources.

The design is simulated using ModelSim simulator, where the wave window is displaced within the ModelSim Graphical User Interface (GUI) main window and hexadecimal radix is applied to the selected signals which are the inputs including the plaintext and the key and the output the ciphertext. Part of the simulation waveforms result of the design is shown in Fig. 7.

CONCLUSION

In this study, a parallel iterative looping architecture of Serpent algorithm (AES candidate) is successfully designed and implemented on a configurable hardware device Virtex-6 FPGA from Xilinx and ISE 14.7. The encryption runs with a speed of 0.447 Gbit/sec and occupying an area of 512 slices out of 31200 (1%) with 32 execution cycles, one clock cycle per round. The total power consumption of the design was 3.746 W. The performance results of the implementation of the proposed design is considered efficient when evaluating the amount of resource utilization and the obtained throughput compared to other related works.

REFERENCES

Anderson, R., E. Biham and L. Knudsen, 1998. Serpent: A proposal for the advanced encryption standard first Advanced Encryption Standard (AES) Conference. National Institute of Standards and Technology, Ventura, California.

Biham, E. and A. Shamir, 1991. Differential cryptanalysis of DES-like cryptosystems. J. Cryptol., 4: 3-72.

Biham, E. and A. Shamir, 1993. Differential Ryptanalysis of the Full 16-Round DES. In: Differential Cryptanalysis of the Data Encryption Standard, Biham, E. and A. Shamir (Eds.). Springer, New York, USA., ISBN:978-1-4613-9316-0, pp: 79-88.

Biham, E., R. Anderson and L. Knudsen, 1998. Serpent: A New Block Cipher Proposal. In: Fast Software Encryption, Vaudenay, S. (Ed.). Springer, Berlin, Germany, pp: 222-238.

Chu, P.P., 2008. FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version. John Wiley & Sons, Hoboken, New Jersey, USA.

Elbirt, A.J. and C. Paar, 2000. An FPGA implementation and performance evaluation of the serpent block cipher. Proceedings of the 2000 ACM/SIGDA 8th International Symposium on Field Programmable Gate Arrays, February 10-11, 2000, ACM, New York, USA., ISBN:1-58113-193-3, pp: 33-40.

Elbirt, A.J., W. Yip, B. Chetwynd and C. Paar, 2001. An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists. IEEE Trans. Very Large Scale Integr. Syst., 9: 545-557.

FIPS, P., 2001. Federal information processing standards publication: Advanced Encryption Standard (AES). National Institute of Standards and Technology, Gaithersburg, Maryland.

Lazaro, J., A. Astarloa, J. Arias, U. Bidarte and C. Cuadrado, 2004. High throughput serpent encryption implementation. Proceedings of the 14th International Conference on Field Programmable Logic and Application, August 30-September1, 2004, Springer, Berlin, Germany, pp: 996-1000.

Matsui, M., 1993. Linear Cryptanalysis Method for DES Cipher. In: Advances in Eurocrypt 93. Eurocrypt 1993. Lecture Notes in Computer Science, Helleseth, T. (Ed.). Springer, Berlin, Germany, pp: 386-397.

Maxfield, C., 2004. The Design Warrior's Guide to FPGAs: Devices, Tools and Flows. Newnes Publishers, MA. USA., ISBN: 0750676043, pp: 542.

Najafi, B., B. Sadeghian, M.S. Zamani and A. Valizadeh, 2004. High speed implementation of serpent algorithm. Proceedings of the 16th International Conference on Microelectronics, December 6-8, 2004, IEEE, Tehran, Iran, ISBN:0-7803-8656-6, pp: 718-721.

Patterson, C., 2000. A Dynamic FPGA Implementation of the Serpent Block Cipher. In: Cryptographic Hardware and Embedded Systems, Koc, C.K. and C. Paar (Eds.). Springer, Berlin, Germany, pp: 141-155.

Standard, D.E., 1977. Federal information processing standards publication 46. National Bureau of Standards, US Department of Commerce, Washington, USA.