Architectures and FPGA Implementations of the 64-bit MISTY1 Block Cipher

P. Kitsos*, M. D. Galanis, O. Koufopavlou

VLSI Design Laboratory
Electrical and Computer Engineering Department
University of Patras, Patras 26 500, Greece

*Corresponding author: Paris Kitsos, VLSI Design Laboratory, Electrical and Computer Engineering Department, University of Patras, 26500 Rio, Patras, Greece.
Phone: +30 2610 997 323, Fax: +30 2610 994 798, e-mail: pkitsos@ee.upatras.gr

Abstract: In this paper, we present two alternative architectures and FPGA implementations of the 64-bit NESSIE proposal, MISTY1 block cipher. The first architecture is suitable for applications with high-performance requirements. A throughput of up to 12.6 Gbps can be achieved at a clock frequency of 168 MHz. The main characteristic of this architecture is that uses RAM blocks embedded in modern FPGA devices in order to implement the S-boxes defined in the block cipher algorithm. The second architecture can be used in implementing applications on area-constrained systems. It utilizes feedback logic and inner pipeline with negative edge-triggered register. This technique shortens the critical path, without increasing the latency of the MISTY1 algorithm execution. Compared with an implementation without inner pipeline, performance improvement of 97% is achieved. The measured throughput of the second architecture implementation is 561 Mbps at 79 MHz.

Keywords: MISTY1, block cipher, cryptography, NESSIE, inner pipeline, negative edge-triggered register, FPGA.

1 Preliminaries

Due to the rapid development of wireless standards, security in mobile communications has gained more importance. However, it is far more difficult to develop new highly qualitative cryptography methods for wireless standards. Some security features have been added and some existing ones have been improved compared with previous mobile systems, in order to achieve more efficient and secure offered services.

Many attempts have taken place in order to establish qualitative cryptography methods. The New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project [1] had as a goal to establish a portfolio of strong cryptographic primitives of various types. For block ciphers a third security level, *normal-legacy*, has been specified, which means a block size of 64 bits instead of 128 (the AES [2] does not specify smaller block size than 128-bit). This was suggested by the project industry board, because the market will still need this block size for compatibility with present applications (e.g., payments with 8-byte personal identification numbers). It is interested in 64-bit block ciphers which are more secure and efficient than the ones presently used. In February 2003, it was announced that the 64-bit block cipher included in the NESSIE portfolio is MISTY1 [3, 4]. This cipher is designed in order to provide high-level security against differential and linear cryptanalysis.

In the third NESSIE workshop [1], a paper for some NESSIE proposal algorithms was presented [5]. The main purpose of that work was the evaluation of these algorithms in terms of hardware implementation performance. In that evaluation, only the encryption

mode of operation was implemented and not the decryption one. In addition, only the unrolled architecture of the algorithm was considered.

Besides from our work, some other architectures and implementations of the MISTY1 cipher have been proposed [6, 7]. The work in [6] is exactly the same one as the MISTY1 implementation proposed in [5]. As previously mentioned, these implementations do not support the decryption mode of the cipher. In [7], two MISTY1 software implementations on a Digital Alpha processor were proposed. Nevertheless, it is well known that the software implementations of security algorithms are much slower than the hardware ones.

In this paper, two architectures and efficient FPGA implementations of the 64-bit NESSIE proposal MISTY1 block cipher are proposed. These two architectures realize both encryption and decryption modes in the same hardware unit. The first architecture is suitable for applications with high-performance requirements. The main feature of this implementation is the unrolling of the cipher rounds in a 75-stage pipeline. Due to the increased critical path delay of the logical expressions, for the S-boxes implementation, the RAM blocks embedded in the considered FPGA devices are utilized.

The second architecture can be used in applications where an area-constrained design is desirable. It uses feedback logic and inner pipeline with negative edge-triggered register [8, 9]. This method shortens the critical path, without increasing the cipher's latency, thus the cipher's throughput is increased.

The rest of the paper is organized as follows: In section 2, the MISTY1 block cipher is described. In section 3, the two proposed hardware architectures are presented and explained in detail. Performance analysis of the FPGA implementations of the two

architectures and comparison results with existing works are given in section 4. Finally, section 5 concludes the paper.

2 Description of the MISTY1 cipher

The MISTY1 [3, 4] block cipher operates with 64-bit block size plaintext and 128-bit secret key. The respective 64-bit ciphertext is produced after a number of *n* rounds, where *n* is a multiple of four. In [10] a number *n*=8 is recommended for usage in real applications. Two are the main parts of the MISTY1 block cipher, the *data randomizing* and the *key scheduling*, which are described in the following.

2.1 Data randomizing part

The MISTY1 data randomizing part for *n*=8 is shown in Fig.1. It consists of 8 identical stages (rounds) with an additional substage (subround).

Fig. 1. Data Randomizing Part

In the encryption mode operation, the 64-bit plaintext is transformed into the 64-bit ciphertext by applying bitwise XOR operations and the sub-functions FO_i $(1 \le i \le 8)$ and FL_i $(1 \le i \le 10)$. In the beginning, the 64-bit plaintext is divided into two 32-bit strings, the left and the right one. The subfunction FO_i uses a 48-bit sub-key KI_i and a 64-bit sub-key

KO_i. The subfuntion FL_i uses a 32-bit sub-key KL_i. The output of each round (stage) is produced by the following equations.

For the odd rounds (i = 1, 3,…, 7) :

Right string: $R_i = FL(L_{i-1}, KL_i)$ and

Left string: $L_i = FL(R_{i-1}, KL_{i+1}) \oplus FO(L_i, KO_i, KI_i)$

For the even rounds (i = 2, 4,…, 8) :

Right string: $R_i = L_{i-1}$ and

Left string: $L_i = R_{i-1} \oplus FO(L_i, KO_i, KI_i)$.

For the last round (i = 9) :

Left string: $R_9=FL(L_8, KL_9)$ and

Right string: $L_9=FL(R_8, KL_{10})$.

The final 64-bit ciphertext is produced from the concatenation of L_9 and R_9.

The decryption mode operation of MISTY1 is similar to the encryption mode. The only differences are the reverse order of the sub-keys and the replacement of the function FL by the function FL^{-1}. Similarly to the encryption mode, in the decryption one the 64-bit ciphertext is divided into the left and right 32-bit strings, which are transformed into the 64-bit plaintext by applying bitwise XOR operations ($\oplus$) and the sub-functions FO_i ($8 \geq i \geq 1$) and FL_i ($10 \geq i \geq 1$). The output of each round is described with the same equations as in

encryption, if the *FL* function is replaced by the FL^{-1}. The resulting plaintext is produced by the concatenation of the final left and right 32-bit strings that are produced by the last subround.

The structure of the *FL* function is shown in Fig. 1. The 32-bit data is split into two 16-bit halves. KL_L is the left and KL_R is the right part of the *KL* 32-bit sub-key, respectively. After AND, OR, and XOR operations between the data and the sub-key a 32-bit string is produced. In the decryption mode, the FL^{-1} function is used instead of the *FL* one.

The 32-bit input data of function *FO* is split into two 16-bit strings (Fig. 1). Then, these strings are correlated with KO_j $(1 \le j \le 4)$ and KI_j $(1 \le j \le 3)$ by using bitwise XOR operations and the sub-functions *FI*. KO_j and KI_j are the left *j*-th 16 bits of *KO* and *KI*, respectively.

The 16-bit input data of the function *FI* is split into two 9-bit and 7-bit strings (Fig. 1). After transformations, bitwise XOR operations and the usage of the substitution tables (S-boxes) S7 and S9, the output string is produced. At the beginning and at the end of FI_j function, the 7-bit string is zero-extended (through the *ZE* module). The *ZE* adds two zero bits in front of the 7-bit string, and in the middle part, the 9-bit string is truncated (through the *TR* module) to 7 bits. The *TR* module truncates the two most significant bits of the 9-bit string. KI_L and KI_R are the left 7 bits and the right 9 bits of *KI*, respectively.

The two S-boxes (S7 and S9) have been designed so that they can be easily implemented in combinational logic as well as by a look-up-tables. Three criteria have been considered in the design of the MISTY1 S-boxes.

- Their average differential/linear probability must be minimal,

- Their delay time in hardware is as short as possible,
- Their algebraic degree is high, if possible.

2.2 Key scheduling part

MISTY1 has a 128-bit key K, which is sub-divided into eight 16-bit sub-keys K_1, K_2, …, K_8 where $K=K_1||K_2||K_3||K_4||K_5||K_6||K_7||K_8$ (|| symbolizes concatenation). From these sub-keys a second set of sub-keys, K_i' $(1 \leq i \leq 8)$ it is produced as is shown in Fig. 2.

Fig. 2. Second Set of Sub-keys

In Table I, the sub-keys that are used in each round are shown.

Table I. Round Sub-keys Mapping Table

3 Proposed hardware architectures

In this section, two alternative hardware architectures of the MISTY1 block cipher are proposed. These architectures are based on the following Eight and One Round structures. The first architecture can be used in high-performance designs, while the second one for designs where a low-area consumption is sought.

3.1 The Eight Rounds architecture

As previously mentioned, the two main parts of the MISTY1 block cipher are the *data randomizing* and the *key scheduling*. In Fig. 3, the data randomizing part of the Eight Rounds architecture is illustrated. Both encryption and decryption procedures can be performed with this architecture. This is opposed to the architectures in [5, 6], where only the encryption process is considered and implemented.

Fig. 3. Eight Rounds Data Randomizing Part Architecture

Compared with previous designs, an alternative architecture for the S-boxes implementation is introduced that exploits the embedded RAM blocks of modern FPGA devices [12]. For the implementation of the S-boxes (S7 and S9), RAM blocks are used instead of logical expressions. Thus a, significant reduction of the critical path delay is achieved. In order to synchronize the S-boxes operations (RAM-based operations), three 1-stage pipeline registers have been inserted in the right branch of the architecture, one for each corresponding RAM (Fig. 4a).

The structure of all the odd and even rounds are identical. For synchronization reasons, pipeline registers are inserted between the functional units. So, after the first input register, a pipeline register with 3-stage delay is added. These pipeline registers are inserted in order to synchronize the key generation part with the data randomizing part. The explanation is given in the following of this paragraph. The insertion of the pipeline registers (9-stage

delay) in the odd and even rounds architectures results in an architecture with a 75 pipeline stages.

Fig. 4b indicates the insertion places of the added pipeline registers in the *FO* function architecture. Due to the *FI* function architecture (Fig. 4a) that uses three 1-stage pipeline registers, for the synchronization of the *FI* functions in *FO* function architecture, 3-stage pipeline registers are needed.

Fig. 4. Architecture of the (a) FI, (b) FO, and (c) FL Functions

The structure of the MISTY1 decryption procedure is similar to the encryption one. In order the proposed architecture to be also suitable for decryption operation, the processes of reversing the sub-keys order and replacing the function *FL* with FL^{-1} are needed. The first one is performed in the proposed MISTY1 key scheduling part and it is described in the following. The latter one is achieved by designing a unit, which implements both *FL* and FL^{-1} functions. The value of the control signal (*enc_dec)* in conjunction with the inserted multiplexer (Fig. 4c) selects the proper output.

The MISTY1 key scheduling architecture of the Eight Rounds architecture is shown in Fig. 5. The proposed scheme allows on-the-fly computation of the sub-keys. The sub-key input in each *FI* function is delayed with the usage of the 2-stage pipeline registers. This is necessary because inside the *FI* unit the KI_L and KI_R sub-keys are entered with a 2-stage delay after the latch of the key value. So, the second array of the sub-keys, K_i', is generated with a 3-stage delay. In order to synchronize the key generation part with the data

randomizing part, an additional 3-stage pipeline register is inserted after the input register and before the start of the data randomizing part (Fig. 3).

Fig. 5. Eight Rounds Key Scheduling Part Architecture

In order the same architecture to be used for both encryption and decryption operations, multiplexers (controlled by the *enc_dec* signal) are added. For example, during the encryption operation the value of the sub-key KO_{31} has the same value as K_3, while during the decryption operation the same value as K_6. So, with the usage of a 2-input 16-bit multiplexer the proper value is selected. Moreover, a Sub-keys Delay Unit (Fig. 5b) is necessary in order to add additional delays. In this unit, 16-bit shift registers are used, so as in each round to add the sub-keys with the proper delay.

3.2 The One Round Architecture

In Fig. 6 the data randomizing part of the One Round architecture is shown. The full MISTY1 block cipher execution requires nine loops of this single round. The output of each round is used as input (through the multiplexers) of the next round. The output of the left branch is used as input in the next right branch, and the output of the right branch is used as input in the next left branch. Both encryption and decryption operations are supported.

Fig. 6. One Round Data Randomizing Part Architecture

The MISTY1 single round, consists of two multiplexers (MUX A) in order the appropriate value between the Plaintext / Ciphertext or the output of the previous round is selected. The input registers are necessary in order to store the input data during the operation of the FL / FL^{-1}. The FL / FL^{-1} unit is structured as described in the eight rounds architecture (Fig. 4c). The second layer of multiplexers (MUX B) select either the output of the FL / FL^{-1} unit or the output of the input registers, when an odd or an even round is executed, respectively.

The architectures of the *FO* and *FI* functions are shown in Fig. 7. In order to reduce the overall system hardware resources for the S-boxes (S7 and S9) implementation, logical expressions are used.

Fig. 7. FO and FI Functions Architecture

For the *FO* function implementation an inner pipeline with negative edge-triggered register is used (Fig. 6, 7). The usage of this technique results in a significant reduction of the round's critical path delays. The negative edge-triggered register is inserted in the *FO* function (Fig. 7), which is roughly in the middle of the round data path (Fig. 6, bold line). The execution time of each round is one system clock cycle. In order to synchronize the processing data paths, similar registers are inserted in the left and right branches of each round (Fig. 6). The result of this insertion is the reduction, roughly in half, of the clock period while the throughput is roughly doubled. A small area penalty is paid from the usage of these pipeline registers. We have to note that the implementation of the negative edge-

triggered register (flip-flop) is easily implemented in FPGA with an insertion of an inverter to the clock line.

The usage of positive (rising) and negative edge-triggered pipeline registers (that capture data on both clock edges) demands a duty cycle of 50%. Deviation from a 50% duty cycle may lead to timing failures in the critical paths [8]. The assumption of a perfect clock with 50% duty cycle is optimistic, giving signals half the clock cycle to propagate from one register to the next. In the low level design, the duty cycle, may not be perfect, and the actual time available for signals to propagate can be smaller. In order to avoid this problem, the worst-case duty cycle of the clock must be accurately modeled in synthesis and timing analysis [11].

For the execution of the whole MISTY1 block cipher, nine system clock cycles are required. The proposed architecture is suitable for area restricted devices because the required hardware resources are reduced relative to the Eight Rounds architecture.

The One Round MISTY1 key scheduling part architecture is illustrated in Fig.8.

Fig. 8. One Round Key Scheduling Part Architecture

The addition of extra delays is achieved, in the Sub-keys Delay Unit (Fig. 8b), by using counters. In this unit, a 2-input 16-bit multiplexer is used in order for the same hardware part to be suitable for both encryption and decryption operations. The multiplexer is controlled by the *enc_dec* signal.

4 Experiments

4.1 Set-up

The proposed MISTY1 architectures were implemented in structural VHDL. The encryption and decryption operation were verified by using the test vectors provided by the NESSIE submission package [1]. The VHDL codes of the two designs were synthesized in Xilinx FPGA devices [12] using the LeonardoSpectrum^{TM} synthesis tool [13]. The correct functionality of the two hardware implementations was verified with simulations using the ModelSim tool from Mentor Graphics [14].

For the hardware implementation of the proposed MISTY1 Eight Round architecture, two Xilinx Virtex devices (XCV1000BG560-6 and XCVII3000BF957-6) were selected. The proposed implementation did not fit in the Virtex-II device, which is considered in [6], because the proposed implementation uses more embedded RAM blocks than the available ones in this device. The XCV1000 device has 128 Kbits of embedded RAM (BlockSelectRAM+), divided in 32 RAM blocks that are separated from the main body of the FPGA. The XCVII3000 device has 1728 Kbits of embedded RAM (BlockSelectRAM+), divided in 96 RAM blocks. For the proposed design 79 Kbits of embedded RAM are used in order to map the necessary for the block cipher S-boxes.

In the Eight Rounds Architecture, for the addition of the sub-keys delays (necessary in the key scheduling part) a 16-bit shift register is used. The Virtex architecture is well suited to easily implement, fast, and efficient shift register by the usage of the SRL16 feature [13]. With this feature, shift registers are implemented without using flip-flop resources. The SRL16 is used to implement a progressive delay line, thereby saving logic resources and

producing the highest performance [13]. So, each 16-bit shift register is implemented by one Look-Up-Table (LUT). The One Round architecture was optimized with the covered area constraint. For this architecture implementation, the Xilinx Virtex XCV400EBG432-8 device was selected.

The proposed implementations of both architectures support encryption and decryption in the same dedicated FPGA device. In order to incorporate this feature, a large number of 2-input 16-bit multiplexers are used. In addition, extra FL^{-1} functions and 2-input 32-bit multiplexers are necessary. So, in the performance and area comparisons with other architectures, (shown in the following section) that support only encryption [5, 6], this feature must be considered. Of course, the penalty is the minor increase of the allocated Control Logic Blocks (CLBs).

4.2 Results

Table II presents the synthesis results of the two proposed MISTY1 architectures. The FPGA device used for synthesizing the implementations is shown. This Table reports the consumed area in terms of CLBs and the clock frequencies. Additionally, synthesis results from existing hardware designs [5], [6] are given for straightforward comparisons. Also, the clock frequency of the software implementation of [7] is given. We mention that the architectures proposed in [5] and [6] are identical, but the authors have reported better implementation performance results in [6] than in [5]. Most probably this difference is due to better VHDL code synthesis of [6]. Finally, in those implementations only the unrolling architecture has been considered.

Table II. Area and frequency results

Table II shows that the implementation of the Eight Rounds architecture on the XCV1000 device achieves an operation frequency of 96 MHz and a throughput of 7.2 Gbps, while the implementation in the XCVII3000 device achieves a frequency of 168 MHz and a throughput of 12.6 Gbps. The critical path delay is determined by the S-box S9 (RAM block) delay. The achieved throughput for the One Round architecture is 561 Mbps, at an operation frequency of 79 MHz, for both encryption and decryption procedures. For one block encryption or decryption operation nine system clock cycles are needed. In order to evaluate the influence of the inner pipeline (negative edge-triggered register), the operation frequency was measured without these pipeline registers. An operation frequency of 40 MHz and a throughput of 284 Mbps were measured. Thus, a major improvement in terms of throughput approximately 97% is achieved.

From Table II it is deduced that the Eight Rounds and the One Round architectures occupy less FPGA area than the implementations of [5] and [6]. Furthermore, the One Round architecture achieves area consumption of approximately 4.5 times smaller than the design in [5]. Also, the One Round structure consumes 2.5 times less number of CLBs when compared with the implementation of proposed Eight Rounds architecture on the XCV1000 device. Thus, the low area consumption of the One Round architecture is justified by the results of Table II.

Fig. 9 illustrates the throughput values of the two proposed architectures and of existing works [5], [6], [7]. Additionally, Fig. 10 shows the throughput-to-area ratio for the considered designs, excluding the software design since this ratio is meaningless in this

case. The throughput-to-area ratio is a measure of the efficiency of a specific hardware implementation. From the results of those figures it is inferred that the proposed MISTY1 architectures are efficient and suitable for FPGA implementation. The Eight Rounds architecture implementation on the XCVII3000 (8-Round (2)) has only smaller throughput from the implementation of [6] on XCVII2000 device. However, our implementation achieves the largest throughput-to-area ratio (3.12 Mbps/slice); thus it is the most efficient hardware implementation among existing MISTY1 FPGA implementations.

Fig. 9. Throughput results

Fig. 10. Throughput-to-area results

5 Conclusions

Two different hardware architectures and FPGA implementations of the MISTY1 block cipher were presented. These architectures support encryption and decryption modes of operations which is not the case in existing works. The Eight Rounds architecture, which is suitable for high-performance designs, achieves a maximum throughput of 12.6 Gbps at an clock frequency of 168 MHz. Additionally, this architecture is more hardware efficient compared with previous hardware architectures. The One Round architecture is suitable for constrained area designs, as shown in the experimental results, and achieves a throughput of 561 Mbps at 79 MHz.

References

[1] "NESSIE. New European Schemes for Signatures, Integrity, and Encryption", https://www.cosic.esat.kuleuven.ac.be/nessie/, 2004.

[2] "Advanced Encryption Standard Development Effort," http://www.nist.gov/aes.

[3] Mitsuru M. "Specification of MISTY1 – a 64-bit Block Cipher". *New European Scheme for Signatures, Integrity, and Encryption (NESSIE) Project*. On line available at https://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions/misty1.zip.

[4] Mitsuru M. "New block encryption algorithm MISTY". *In Fast Software Encryption 1997*, vol. 1267 of LNCS, pages 54-68. Springer-Verlag, 1997.

[5] Standaert F-X, Rouvroy G, Quisquater J-J, Legat J-D. "Efficient FPGA Implementations of Block Ciphers KHAZAD and MISTY1". *In Proceedings of the Third NESSIE Workshop*, November 6-7 2002, Munich, Germany.

[6] Rouvroy G, Standaert F-X, Quisquater J-J, Legat J-D. "Efficient FPGA Implementation of Block Cipher MISTY1". *In Proceedings of the 10th Reconfigurable Architectures Workshop (RAW 2003)*, April 22, Nice, France.

[7] Nakajima J and Matsui M. "Fast Software Implementations of MISTY1 on Alpha Processors". In *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E82-A, No. 1, pp. 107-116, January 1999.

[8] Chung W, Lo T, and Sachdev M. "A Comparative Analysis of Low-Power Low-Voltage Dual-Edge-Triggered Flip-Flop". In *IEEE Transaction on Very Large Scale Integration* (VLSI) *Systems*, Vol. 10, No. 6, pp. 913-918, December 2002.

[9] Strollo A G M, Napoli E, and Cimino C. “Analysis of Power Dissipation in Double Edge-Triggered Flip-Flops”. In *IEEE Transaction on Very Large Scale Integration* (VLSI) *Systems*, Vol. 8, No. 5, pp. 624-629, October 2000.

[10] Matsui M. “Supporting Document of MISTY1”. version 1.1. On line available at https://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions/misty1.zip.

[11] “*Reuse Methodology Manual for System-on-Chip Designs*”, by Michael Keating, and Pierre Bricaud, Kluwer Academic Publishers, 101 Philip Drive, Assinipi Park, Norwell, Massachusetts. 1999.

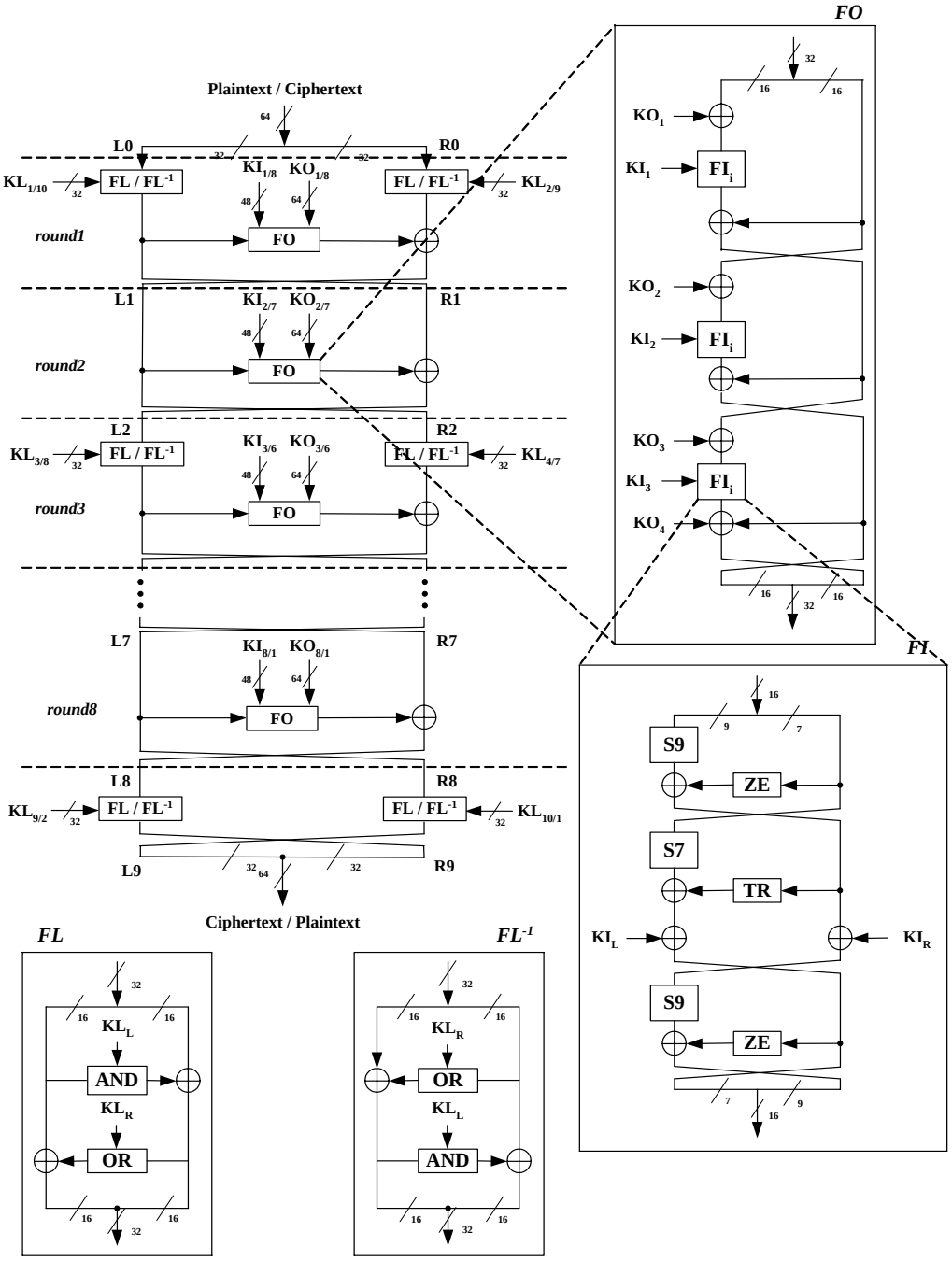
[12] Xilinx Inc., “Virtex, 2.5 V Field Programmable Gate Arrays,” 2001, www.xilinx.com, 2005.

[13] Mentor Graphics Inc., LeonardoSpectrum™ synthesis tool, http://www.mentor.com/leonardospectrum, 2005.

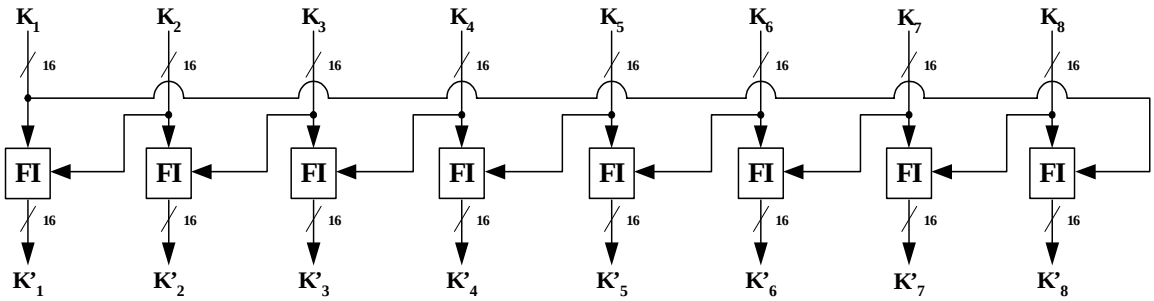
[14] ModelSim™, http://www.model.com, 2005.

List of Figures:

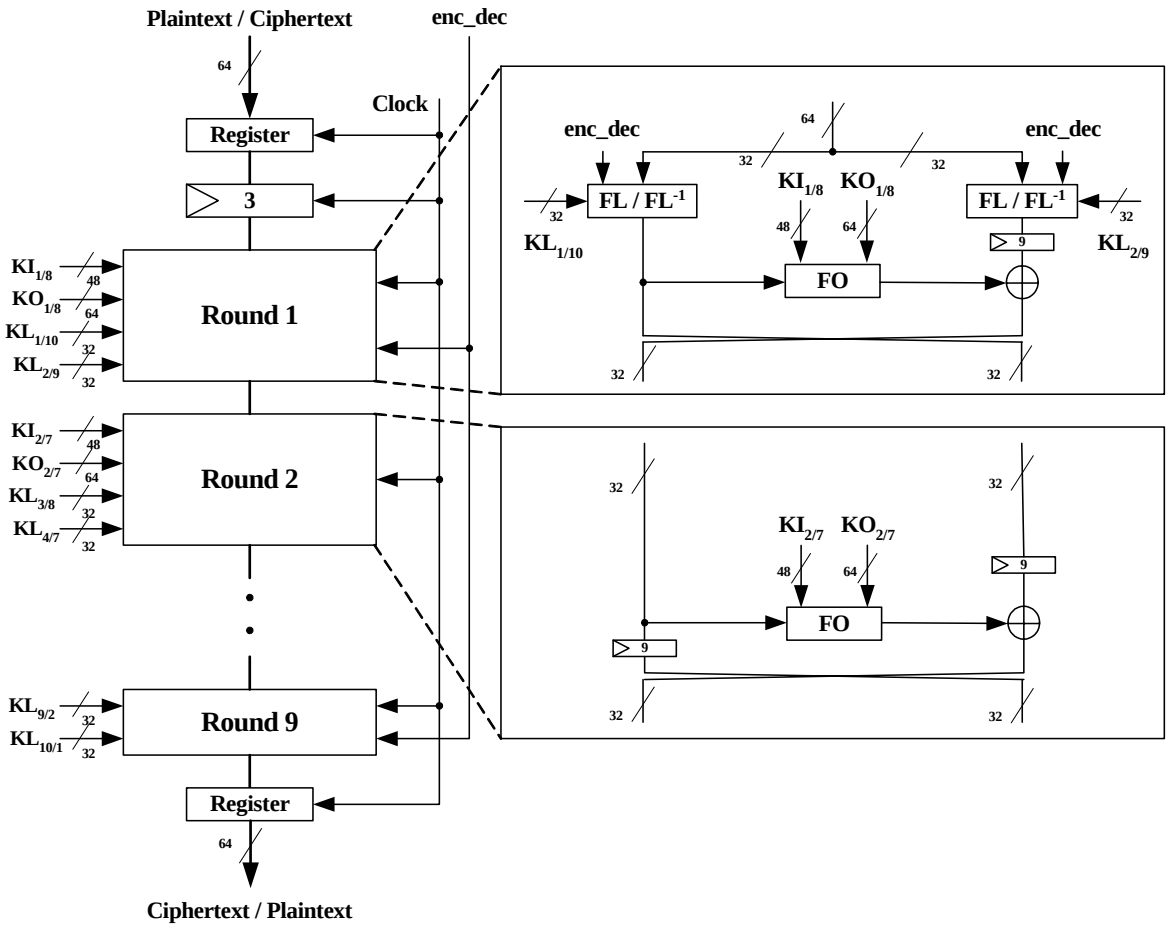
P. Kitsos Fig. 1. Data Randomizing Part



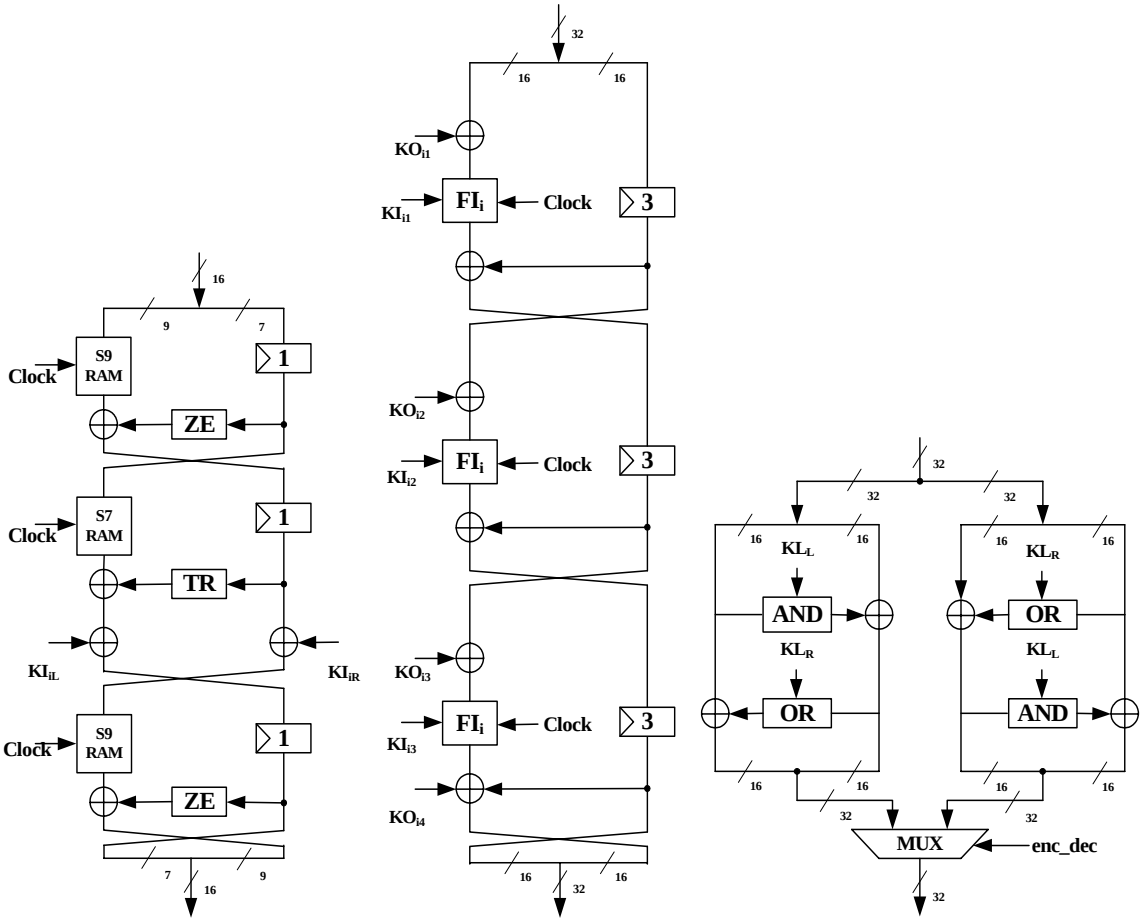
P. Kitsos Fig. 2. Second Set of Sub-keys



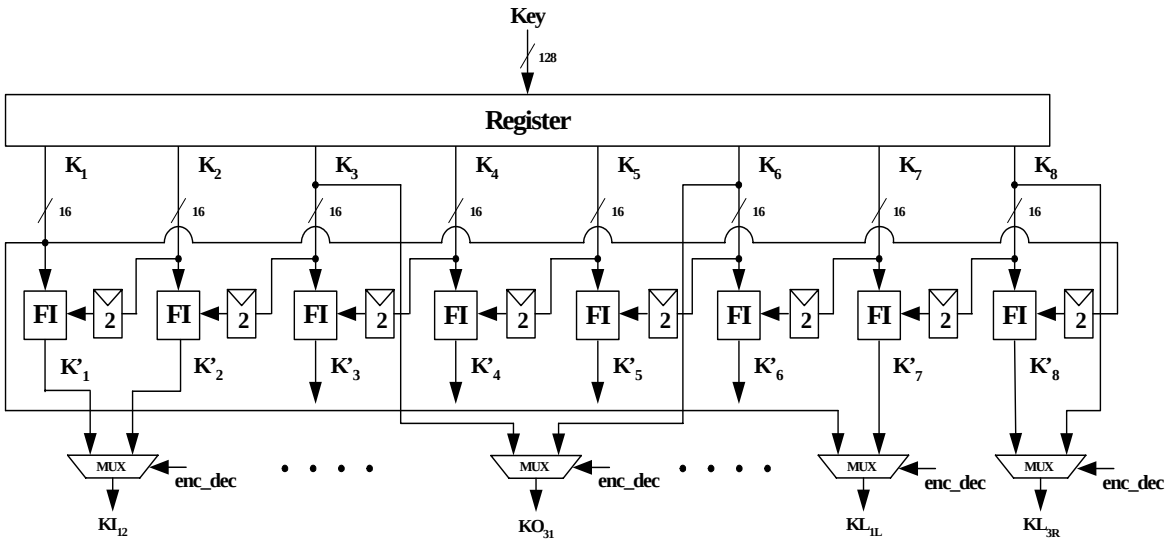
P. Kitsos Fig. 3. Eight Rounds Data Randomizing Part Architecture



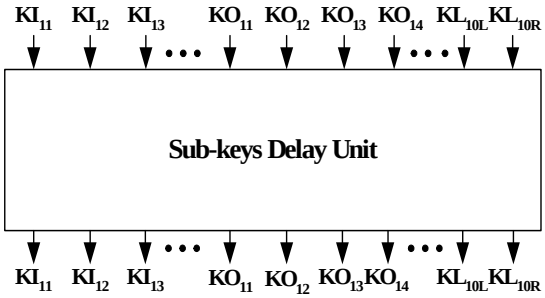
P. Kitsos Fig. 4. Architecture of the (a) FI, (b) FO, and (c) FL Functions



P. Kitsos Fig. 5. Eight Rounds Key Scheduling Part Architecture

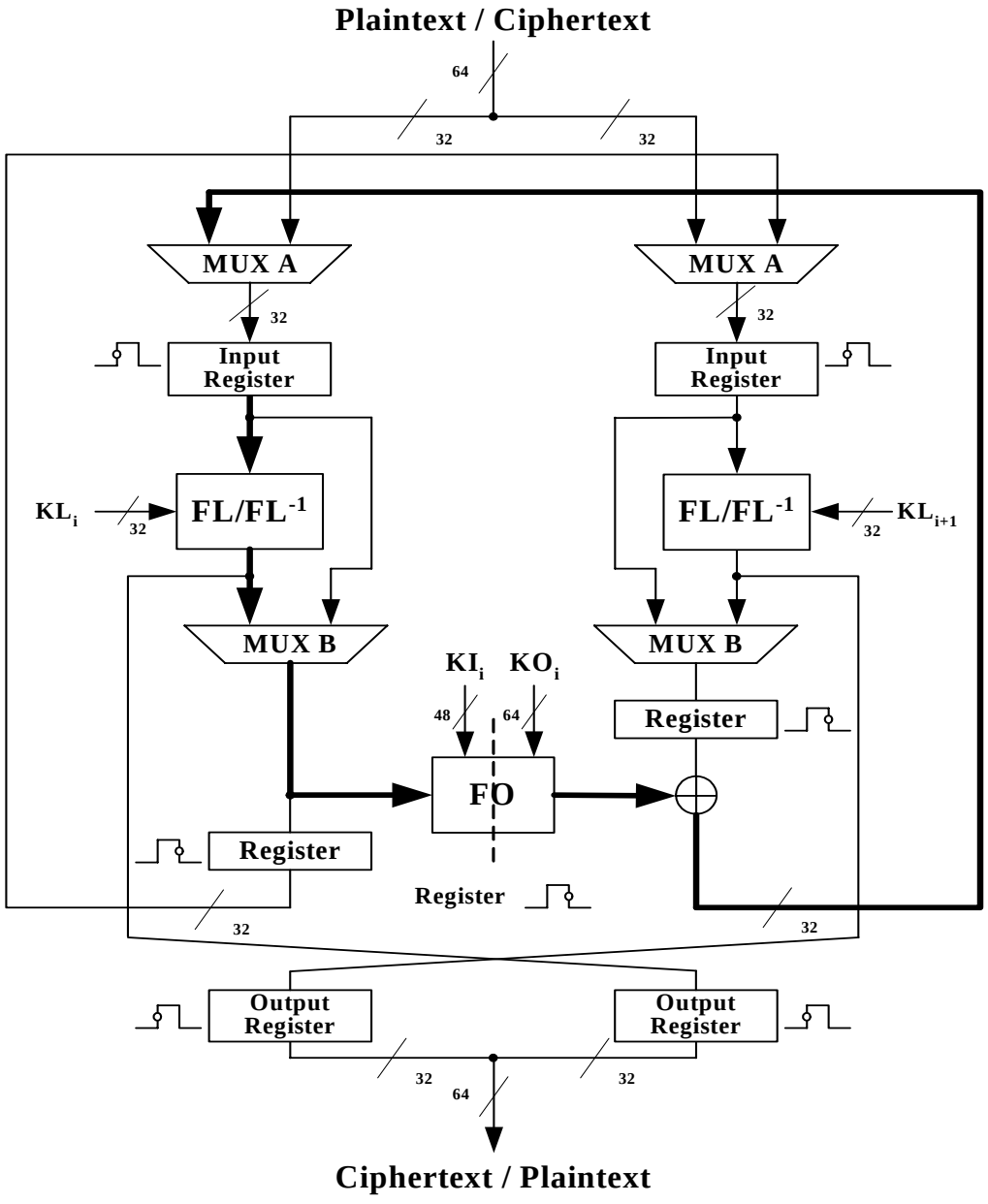


(a)

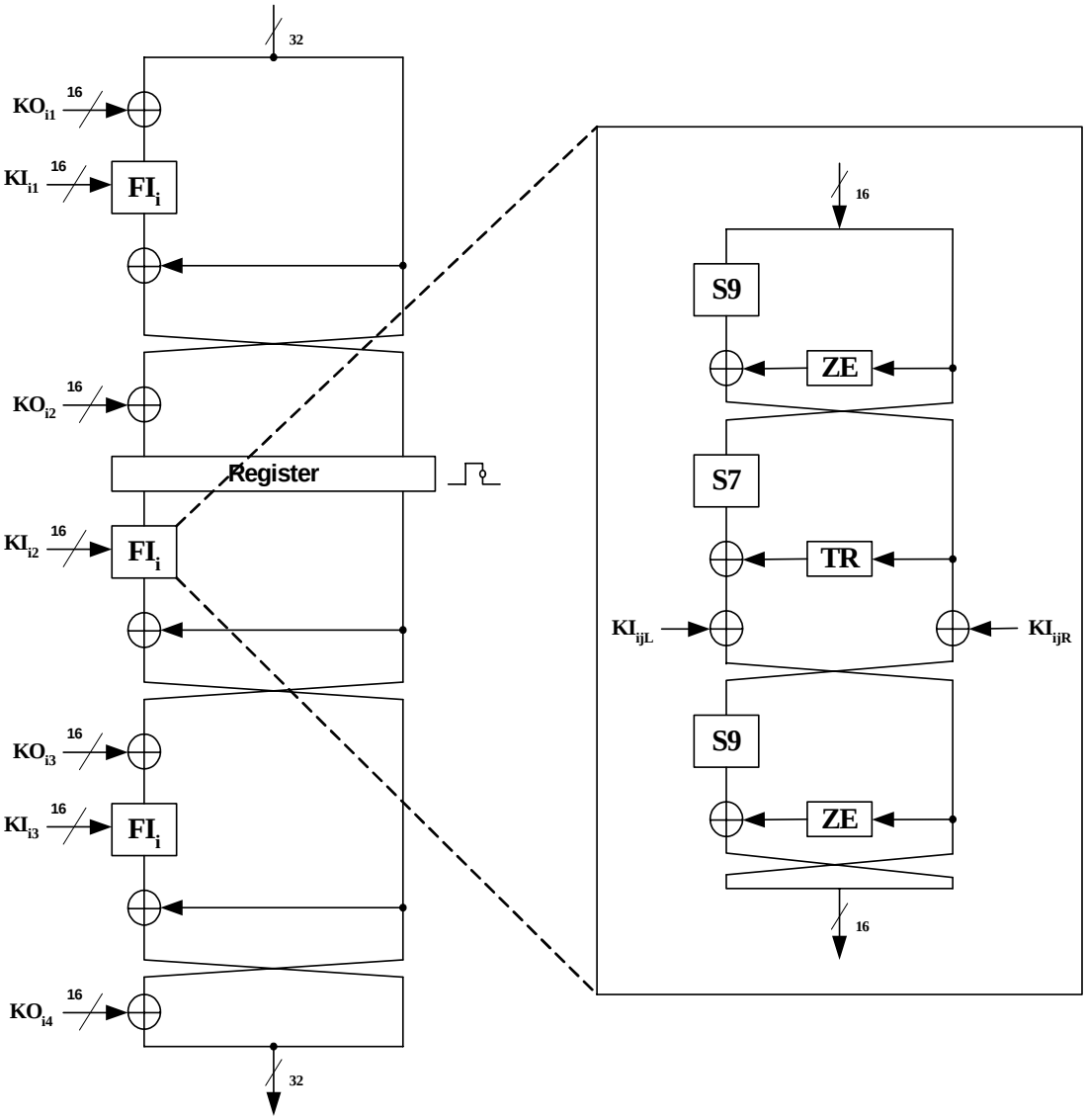


(b)

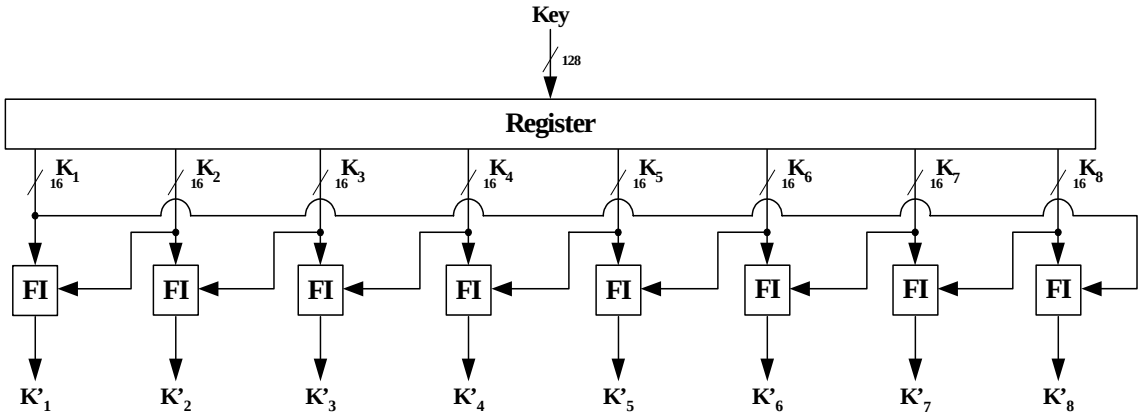
P. Kitsos Fig. 6. One Round Data Randomizing Part Architecture



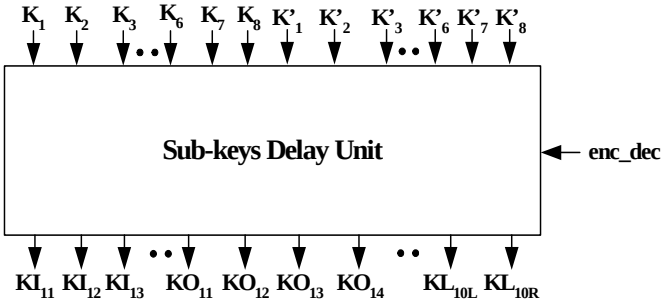
P. Kitsos Fig. 7. FO and FI Functions Architecture



P. Kitsos Fig. 8. One Round Key Scheduling Part Architecture

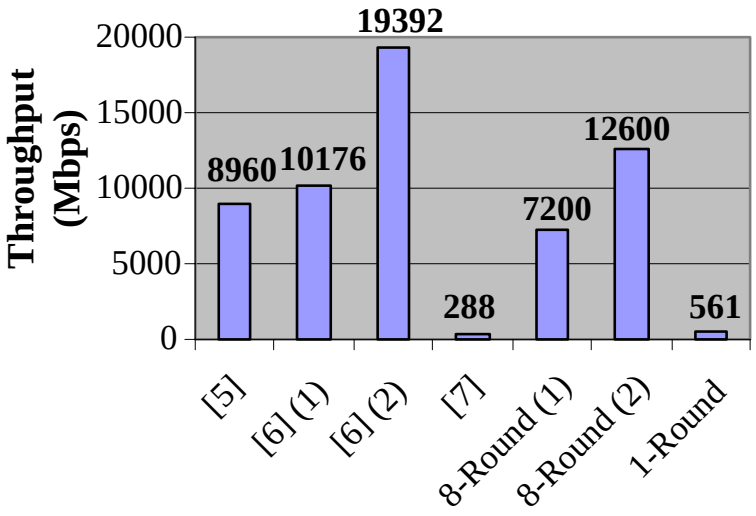


(a)



(b)

P. Kitsos Fig. 9. Throughput results



P. Kitsos Fig. 10. Throughput-to-area results

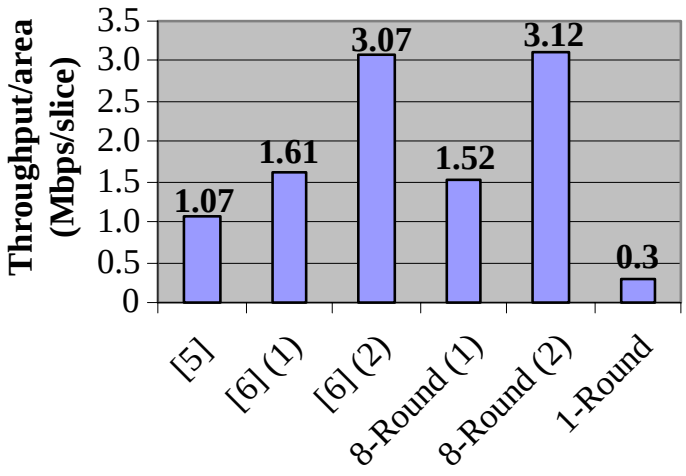


Table List:

Table I. Round Sub-keys Mapping Table

Round	***KO_1***	***KO_2***	***KO_3***	***KO_4***	***KI_1***	***KI_2***	***KI_3***	***KL_L***	***KL_R***
1	K_1	K_3	K_8	K_5	K'_6	K'_2	K'_4	K_1	K'_7
2	K_2	K_4	K_1	K_6	K'_7	K'_3	K'_5	K'_3	K_5
3	K_3	K_5	K_2	K_7	K'_8	K'_4	K'_6	K_2	K'_8
4	K_4	K_6	K_3	K_8	K'_1	K'_5	K'_7	K'_4	K_6
5	K_5	K_7	K_4	K_1	K'_2	K'_3	K'_8	K_3	K'_1
6	K_6	K_8	K_5	K_2	K'_3	K'_4	K'_8	K'_5	K_7
7	K_7	K_1	K_6	K_3	K'_4	K'_5	K'_1	K_4	K'_2
8	K_8	K_2	K_7	K_4	K'_5	K'_6	K'_2	K'_6	K_8
9	-	-	-	-	-	-	-	K_5	K'_3

Table II. Area and frequency results

Architecture	**Process**	**FPGA Device**	**CLB Slices**	**Frequency (MHz)**
[5]	Encryption	XCV1000 BG560-6	8,386	140
[6] (1)	Encryption	XCV1000 BG560-6	6,322	159
[6] (2)	Encryption	XCVII2000 BG575-6	6,322	303
[7]	Encryption/ Decryption	Software	-	500
Proposed Eight Round (1)	Encryption/ Decryption	XCV1000 BG560-6	4,735	96
Proposed Eight Round (2)	Encryption/ Decryption	XCVII3000 BF957-6	4,039	168
Proposed One Round	Encryption/ Decryption	XCV400 EBG432-8	1,865	79