# A RAM-Based FPGA Implementation of the 64-bit MISTY1 Block Cipher

P. Kitsos, M.D. Galanis, and O. Koufopavlou
VLSI Design Lab, Electrical and Computer Engineering Department, University of Patras, Patras, 26500, Greece
pkitsos@ee.upatras.gr

***Abstract*—A high-throughput hardware architecture and FPGA implementation of the 64-bit NESSIE proposal, MISTY1 block cipher, is presented in this paper. This architecture, in contrast to previous ones, supports both encryption and decryption processes. It is based on the unrolling of the MISTY1 rounds in a 75-stage pipeline. Furthermore, the implementation of the proposed architecture in specific FPGA devices utilizes the embedded RAM blocks of those devices. A throughput of up to 12.6 Gbps can be achieved at a clock frequency of 168 MHz. So, the proposed architecture is suitable for applications with high throughput requirements, like in contemporary and future wireless communication standards.**

## I. INTRODUCTION

Due to the rapid development of wireless communications standards, the security subject 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 called *normal-legacy*, has been specified. The block size is set to 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]. 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 [4]. The main purpose of this work was the evaluation of these algorithms in terms of hardware implementation performance. In this 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.

Some other implementations of the MISTY1 block cipher have been published [4], [5], [6]. The work in [5] is exactly the same one as the MISTY1 implementation proposed in [4]. As previously mentioned, these hardware implementations do not support the decryption mode of the cipher. In [6], two MISTY1 software implementations on a Digital Alpha processor were proposed. Nevertheless, it is well known that the software implementations are much slower than the hardware ones. This is also proved in this paper's experimental results.

This paper presents a high-throughput hardware architecture for the 64-bit NESSIE proposal MISTY1 block cipher and its FPGA implementation, are proposed. This architecture can implement both encryption and decryption modes in the same hardware module. Its main feature is the unrolling of the cipher rounds in a 75-stage pipeline. For the FPGA implementation of the proposed architecture, the S-boxes utilize the embedded RAM blocks of the FPGA device, since an implementation of the S-boxes in combinational logic causes an increase in the critical path's delay.

The rest of the paper is organized as follows: In section 2, the MISTY1 block cipher is introduced. In section 3, the proposed hardware architecture is presented and explained in detail. Performance analysis and comparison results with existing works are given in section 4, while section 5 concludes the paper.

## II. MISTY1 BLOCK CIPHER

The MISTY1 [3] 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 [7], a number of n=8 is recommended for use in real applications. There are two main parts of the MISTY1 block cipher, the data randomizing and the key scheduling. In the following of this section, these two parts are described.

The MISTY1 data randomizing part, for *n*=8, consists of 8 identical stages (rounds) with an additional substage (subround). In the encryption mode operation, the 64-bit plaintext is transformed into the 64-bit ciphertext by applying bitwise XOR ( $\oplus$ ) operations and the sub-functions $FO_i$ ( $0 \le i \le 8$ ) and $FL_i$ ( $0 \le i \le 10$ ). 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: $R_i = FL(L_{i-1}, KL_i)$
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: $R_i = L_{i-1}$

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

For the last round (i = 9):

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

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 operation of MISTY1 is similar to the encryption one. 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 operation, 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 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.

In the FL function, 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. Then, these strings are correlated with $KO_j$ ($1 \leq j \leq 4$) and $KI_j$ ($1 \leq j \leq 3$) 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. 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 a block called *ZE*). The *ZE* adds two zero bits in front of the 7-bit string, and in the middle part, the 9-bit string is truncated to 7 bits (through a block called *TR*). 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: (a) their average differential/linear probability must be minimal, (b) their delay time in hardware is as short as possible, and (c) their algebraic degree is high, if possible.

MISTY1 considers 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, $K_i'$ ($0 \leq i \leq 8$) is produced as it is shown in Fig. 1. Table 1 in [7] illustrates the sub-keys that are used in each round.

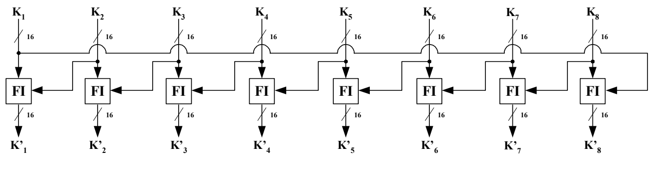

Figure 1. Second set of sub-keys

## III. Proposed Hardware Architecture

As previously mentioned, the two main parts of the MISTY1 block cipher are the data randomizing and the key scheduling. In Fig. 2, the data randomizing part of the proposed eight rounds architecture is illustrated. By using this architecture, both encryption and decryption operations can be performed. This is opposed to the [4], [5], where only the encryption process is considered and implemented.

The proposed implementations of the *FI*, *FO* and *FL*/$FL^{-1}$ functions are shown in Fig. 3. Compared with previous designs, an alternative architecture for the S-boxes implementation is introduced. 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 *FI* architecture, one for each corresponding RAM (Fig. 3a).

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 a 3-stage delay is added (Fig. 2). 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. 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. 2).

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

The structure of the MISTY1 decryption operation 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 required. 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. 3c) selects the proper output.

The MISTY1 key scheduling architecture is shown in Fig. 4. 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. 2).

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 $K_6$ [7]. So, with the usage of a 2-input 16-bit multiplexer the proper value is selected. Moreover, a Sub-keys Delay Unit (Fig. 4b) 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.

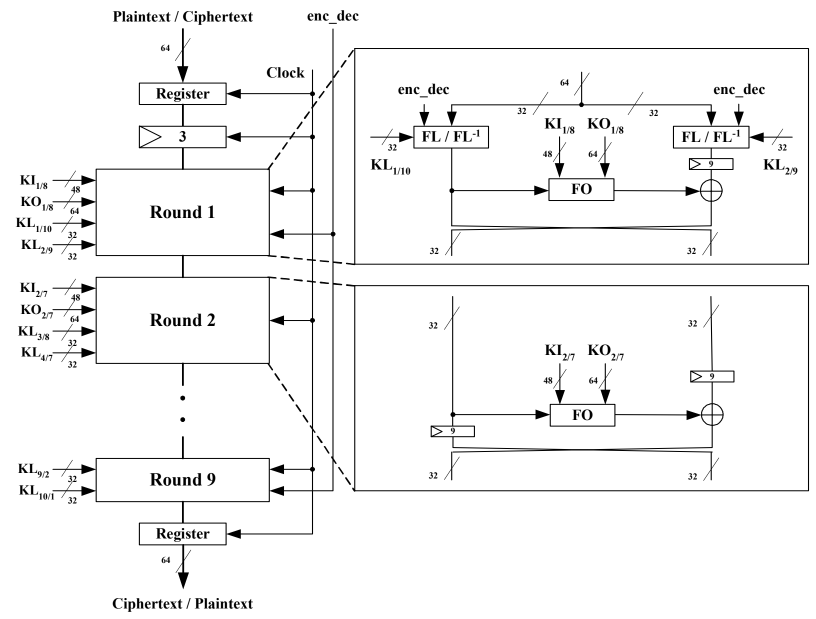

Figure 2. Proposed architecture of the data randomizing part

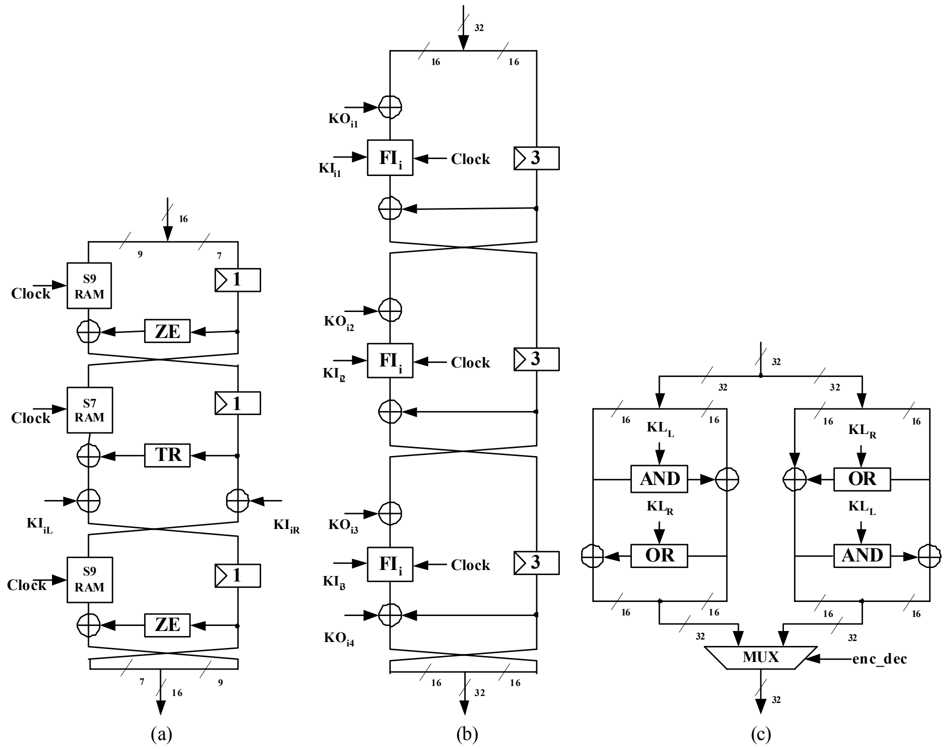

Figure 3. Proposed architectures of the (a) *FI*, (b) *FO*, and (c) *FL/FL$^{-1}$* functions

## IV. Measurements and Comparisons

The proposed architecture was 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 Virtex FPGA devices [8] using LeonardoSpectrum$^{TM}$ [9]. The correct functionality of the hardware implementation was verified with simulations. The measurements of the performance analysis are shown in Table 1. In this Table, measurements from other designs [4], [5], including the software implementation of [6], are included. The architectures proposed in [4] and [5] are identical, but the authors have reported better implementation performance results in [5] than in [4]. Most probably this difference is due to better VHDL code synthesis of [5].

For the hardware implementation of the proposed RAM-based MISTY1 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 [5], because the proposed implementation uses more embedded RAM blocks than the available ones in this device. The XCV1000BG560-6 device has 128K-bits of embedded RAM (BlockSelectRAM+), divided in 32 RAM blocks that are separated from the main body of the FPGA. The XCVII3000BF957-6 device has 1728K-bits of embedded RAM (BlockSelectRAM+), divided in 96 RAM blocks. For our architecture implementation, 79 K-bits of embedded RAM are necessary for the MISTY1 S-boxes implementation.

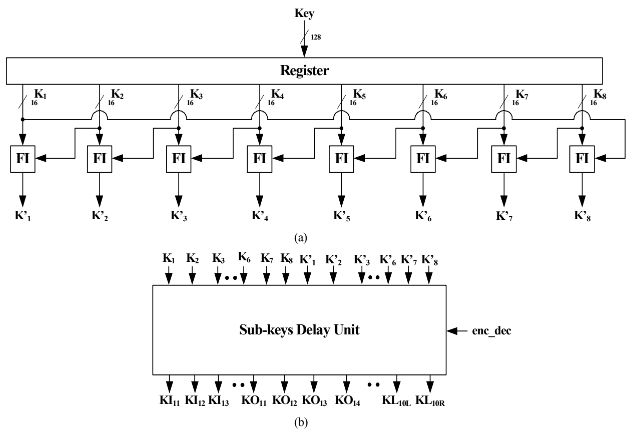
Key
128
Register
K1 K2 K3 K4 K5 K6 K7 K8
16
FI
K'1 K'2 K'3 K'4 K'5 K'6 K'7 K'8
(a)
K1 K2 K3 K6 K7 K8 K'1 K'2 K'3 K'6 K'7 K'8
Sub-keys Delay Unit
enc_dec
KI11 KI12 KI13 KO11 KO12 KO13 KO14 KL10L KL10R
(b)

Figure 4. Proposed architecture for the key scheduling

In our 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 registers by the usage of the SRL16 feature [8], [9]. With this feature, shift registers are implemented without using flip-flop resources. The SRL16 feature is used to implement a progressive delay line, thereby saving logic resources and producing the highest performance [8], [9]. So, each 16-bit shift register is implemented by one Look-Up-Table (LUT). The implementation in the XCV1000BG560-6 device achieves an operation frequency of 96 MHz and a throughput of 7.2 Gbps, while the implementation in the XCVII3000BF957-6 device achieves a frequency of 168 MHz and a throughput of 12.6 Gbps. The critical path delay of our implementation is determined by the S-box S9 (RAM block) delay.

The proposed architecture implementation supports encryption and decryption operations 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 comparisons with other architectures that support only encryption [4], [5], this feature must be considered. Of course, the penalty is the minor increase of the allocated hardware resources.

From the performance measurements shown in Table 1, it is concluded that the proposed MISTY1 architecture is suitable for FPGA implementation. In particular, our architecture implementation in the XCVII3000 FPGA is more efficient, compared with the implementation in [5] for the same FPGA family, as the throughput-to-area ratio is larger (3.12 Mbps /Slice). This ratio is a measure of the hardware efficiency of a specific implementation.

TABLE I. PERFORMANCE ANALYSIS MEASUREMENTS

| Architecture | Operation | FPGA Device | CLB Slices | *F* (MHz) | Throughput (Mbps) | Throughput/Area (Mbps/Slice) |
|---|---|---|---|---|---|---|
| **[4]** | Encryption | XCV1000 | 8,386 | 140 | 8,960 | 1.07 |
| **[5]** | Encryption | XCV1000 | 6,322 | 159 | 10,176 | 1.61 |
| **[5]** | Encryption | XCVII2000 | 6,322 | 303 | 19,392 | 3.07 |
| **[6]** | Encryption / Decryption | - | - | 500 | 288 | - |
| **Proposed** | Encryption / Decryption | XCV1000 | 4,735 | 96 | 7,200 | 1.52 |
| **Proposed** | Encryption / Decryption | XCVII3000 | 4,039 | 168 | 12,600 | 3.12 |

## V. CONCLUSIONS

In this paper, a hardware architecture and an FPGA implementation of the MISTY1 block cipher, has been presented. In contrast to existing implementations, this architecture support encryption and decryption. In our architecture, the MISTY1 rounds are unrolled and RAM blocks embedded in the considered FPGA devices are used for the implementation of the S-boxes. A 75-stage pipeline is inserted that achieves a maximum throughput of 12.6 Gbps at a frequency of 168 MHz. Compared with previously published MISTY1 implementations, the proposed architecture is more hardware efficient since it can achieve the largest throughput-to-area ratio value.

## REFERENCES

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

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

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

[4] F. X Standaert et al., "Efficient FPGA Implementations of Block Ciphers KHAZAD and MISTY1", in *Proc. of the Third NESSIE Workshop*, November 6-7, Munich, Germany, 2002.

[5] G. Rouvroy et al., "Efficient FPGA Implementation of Block Cipher MISTY1", i*n Proc. of the 10th Reconfigurable Architecture Workshop* (RAW), April 22, Nice, France, 2003.

[6] J. Nakajima and M. Matsui, "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.

[7] M. Matsui, "Specification of MISTY1- A 64-bit block cipher", https://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions/misty1.zip.

[8] Xilinx Inc., Virtex FPGAs, www.xilinx.com, 2004.

[9] Mentor Graphics Inc., LeonardoSpectrum, "Synthesis and Technology Manual", Software Release v2001.1, February 2001.