# HIGH-SPEED HARDWARE IMPLEMENTATIONS OF THE KASUMI BLOCK CIPHER

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

VLSI Design Laboratory
Electrical and Computer Engineering Department
University of Patras, Patras, GREECE
E-mail: pkitsos@ee.upatras.gr

## ABSTRACT

KASUMI block cipher is used for the security part of many synchronous wireless standards. In this paper two architectures and efficient implementations of the 64-bit KASUMI block cipher are presented. In the first one, the pipeline technique (inner-round and outer-round pipeline) is used and throughput value equal to 3584 Mbps at 56 MHz is achieved. The second one uses feedback logic and reaches a throughput value equal to 432 Mbps at 54 MHz. The designs were coded using VHDL language and for the hardware implementations, a FPGA device was used. A detailed analysis, in terms of performance, and covered area is shown. The proposed implementations outperform any previous published KASUMI implementations in terms of performance.

## 1. INTRODUCTION

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

A new security algorithm, known as A5/3 [1], will provide users of GSM mobile phones with an even higher level of protection against eavesdropping than they have already. Also GEA3 shall be capable of use for EDGE and GPRS [2]. In addition, in the UMTS [3] two algorithms are standardized for confidentiality and data integrity. The *f8* [4] is used for the user data confidentiality and *f9* [4] for the data integrity. All the above-referred algorithms are based on the 64-bit KASUMI block cipher [5].

In this paper, two architectures and VLSI implementations of the KASUMI block cipher are presented. In order to improve the cipher performance the inner-round pipeline technique is used. In addition, for the implementation of the inner-round pipeline, negative edge-triggered registers are used. With this method the algorithmic latency time does not increase, and simultaneously the critical path is small enough. Recently, many designs have been proposed for the hardware implementation of the KASUMI block cipher [6-9]. The proposed ones outperform all these designs in terms of performance.

This paper is organized as follows: In sections 2 the KASUMI block cipher is described briefly. The proposed architectures and VLSI implementations are presented in detail in section 3. The synthesis results for the FPGA implementation are shown in section 4, and the paper conclusions are given in section 5.

## 2. KASUMI BLOCK CIPHER

KASUMI cipher consists of the data randomizing part and the key scheduling. By using a 128-bit ciphering key *K*, it modifies a 64-bit *Plaintext* to a 64-bit *Ciphertext*. The 64-bit input is divided into two 32-bit strings $L_0$ and $R_0$. The outputs of each round are produced according to the following equation:

$$R_i = L_{i-1}, \quad L_i = R_{i-1} \oplus f(L_{i-1}, RK_i), \quad 1 \le i \le 8 \quad (1)$$

where $f$ denotes the round function with $L_{i-1}$ and round key $RK_i$ as inputs. The round key $RK_i$ comprises the sub-key triplet ($KL_i$, $KO_i$, $KI_i$). The produced *Ciphertext* is the 64-bit string derived from the concatenation of the $L_8$ and the $R_8$, which are produced at the end of the eighth round. The $f$ itself is constructed from the *FL* and *FO* sub-functions, with associated sub-keys $KL_i$ (used with *FL*) and sub-keys $KO_i$ and $KI_i$ (used with *FO*), followed by a bit-wise XOR operation with the previous branch. This function has two different forms. In the odd round the *FL* function is performed first and then the *FO* function. In

the even round, the order of the functions is reversed. The 128-bit round key, $RK_i$, is derived from key *K*.

## 3. PROPOSED ARCHITECTURES

The proposed KASUMI cipher architecture consists of the two main components. The Key Scheduling Unit, which is responsible for the round keys generation, and the KASUMI Core, which executes the basic encryption procedure. Two different KASUMI Core architectures have been examined. In order to increase the system throughput the first one (*8R* pipeline) has 8 outer-pipeline stages. The second (*2R* pipeline) reduces the required hardware resources by using only two pipeline stages. As was previously mentioned the even round of KASUMI cipher has a different structure of the odd round. The odd rounds are denoted as Odd Round Cell (*ORC*) and the even rounds are denoted as Even Round Cell (*ERC*).

Fig. 1 shows the implementations of the *ERC* and *ORC*. For both implementations a data block cipher execution (execution latency) needs eight clock cycles. For both implementations the Key Expansion Unit is the same. The sub-keys are produced and stored in a register file. For the outer-pipeline registers trivial positive edge-triggered registers are used. The entire round, including internal registers (inner-pipeline) make the critical path sorter and significantly increases the cipher throughput. The inner-pipeline registers are negative edge-triggered registers. So, the execution time of each round is one clock cycle. In order to synchronize the processing data similar registers are inserted in the left and right branches (*Li* and *Ri*) of each round. As a result, the clock period is reduced roughly in half. The *FO* sub-function negative edge-triggered pipeline design is illustrated in Fig. 1.

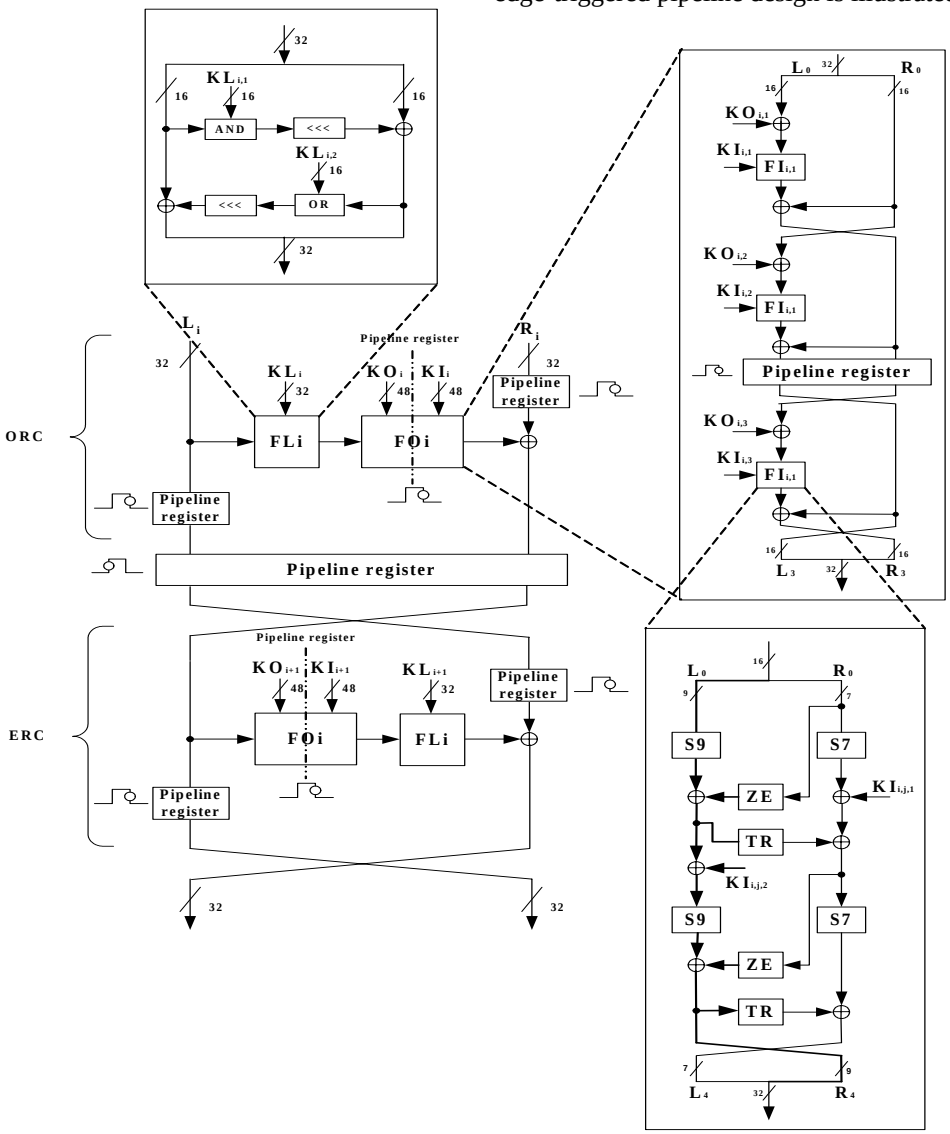


**Fig.1:** The KASUMI rounds implementation

The added register was inserted almost in the middle of the data path round. A small area penalty due to the use of three extra pipeline registers by round is paid.

The two KASUMI Core architectures are depicted in Fig. 2.

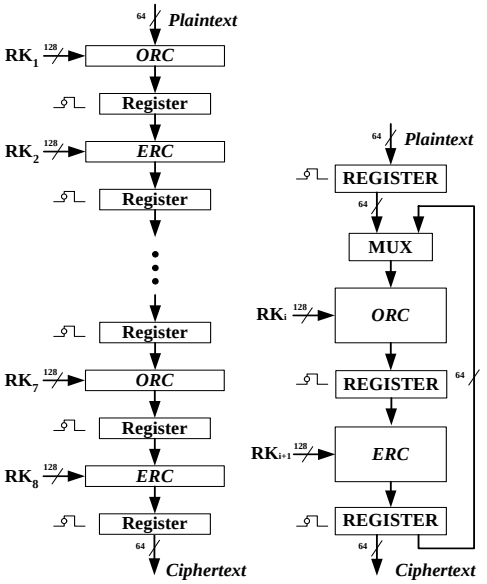


**Fig. 2:** KASUMI core architecture, a) 8R pipeline and b) 2R pipeline

The first architecture has 8 pipeline registers between the rounds. These registers temporally store the output of the previous round. The cipher has the possibility to process 8 data blocks in one clock cycle with the simultaneously increment the cipher throughput. The main advantage of this architecture is the high processing throughput.

The second architecture is the concatenation of two basic cells, *ORC* and *ERC* in an iterative scheme with an additional pipeline register between the two cells. This architecture requires a quarter of the hardware resources of the full unroll architecture. Also, one multiplexer is necessary in order to combine the input block data with the previous block. Two data blocks can process simultaneously. The key scheduler forces one odd number round key and one even number round key at every cycle. The KASUMI cipher execution for one block is completed in four loops.

The proposed Key Expansion Unit architecture is implemented by shift registers in order to produce a number of sub-keys. The rest of the sub-keys are generated by bit-wise XOR operations with the constants *Cj*. These constants are stored in the 8x16 bits ROM. A total of 40 16-bit sub-keys are generated. With the appropriate concatenations of the sub-keys, the round keys are generated. The round keys are computed and stored in a register file. A storage unit, with 52x16-bit register file, is used. The appropriate sub-keys are produced directly when the ciphering key is applied to the cipher. During the first clock cycle, the first round sub-keys are set directly from the Key Scheduling Unit to the data randomizing, while the rest of the sub-keys are stored in the register file. With this way, the writing process of the register file does not add any additional delay. The Key Scheduling Unit architecture is shown in Fig. 3.

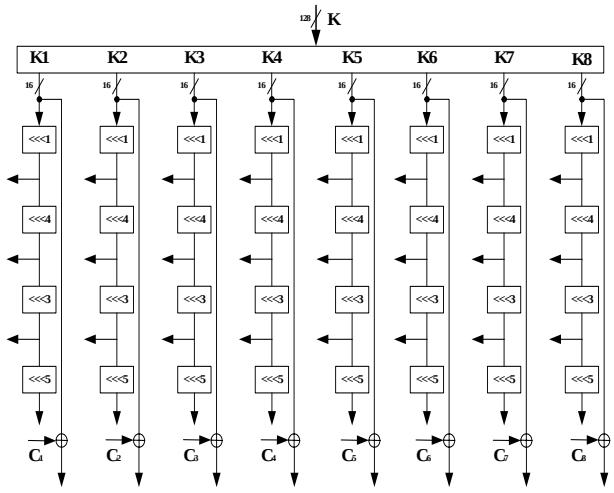


**Fig. 3:** The KASUMI key scheduling unit architecture

The KASUMI specifications do not described the decryption process because the GSM, EDGE, GPRS, and UMTS use the KASUMI only in encryption mode. However, a general purpose KASUMI implementation that supports both encryption and decryption is proposed. This can be useful for bulk encryption cipher usage, in other applications. The only difference between the two modes is the order of the round keys. The first key in the encryption is the last key in the decryption. When decryption must be performed, the round keys are read by the register file in reverse order.

## 4. SYNTHESIS RESULTS AND EVALUATION

Both implementations were captured by using VHDL with structural description logic. All the VHDL code was simulated and verified by using the test vectors, provided by the 3GPP standard [5, 10]. The design was synthesized using FPGA devices from XILINX [11].

The KASUMI Key Scheduling architecture is simple and faster than the data path. So, the encryption has the same throughput with the decryption operation.

The required cipher S-BOXes has been implemented with combinational logic, which minimized the required hardware resources. For the sub-keys delays (necessary in the key scheduling part) a 16-bit shift register is used. The Virtex architecture is well suited to implement efficient shift register (without using flip-flop) by the usage of the SRL16 feature [12].

The synthesis results of the proposed KASUMI implementations for the 8R and 2R architectures are shown in Table 1. The XCV300E-8BG432 FPGA device was used for both implementations.

**Table 1:** KASUMI FPGA Synthesis Results

| KASUMI Architectures | CLBs | FGs | DFFs | F (MHz) | Throughput (Mbps) |
|---|---|---|---|---|---|
| **8R** | 4738 | 9479 | 2564 | 56 | 3584 |
| **2R** | 1726 | 3452 | 1571 | 54 | 432 |

Performance comparisons between the proposed KASUMI implementations and implementations in [6-9] are given in Table 2.

**Table 2:** KASUMI Time Performance Comparisons

| KASUMI Architecture | F (MHz) | Throughput | Device |
|---|---|---|---|
| **8R_Comb. [6]** | 20.86 | 1.34 Gbps | XCV400E-6BG432 |
| **8R_LUT. [6]** | 37.72 | 2.42 Gbps | XCV1000E-6BG560 |
| **2R_Comb. [6, 7]** | 20.88 | 167.04 Mbps | XCV300E-6BG432 |
| **2R_LUT. [6, 7]** | 35.35 | 70.70 Mbps | XCV200E-6FG456 |
| **Type1 [8]** | 20 | 110 Mbps | - |
| **Type2 [8]** | 60 | 410 Mbps | - |
| **Synth1 [9]** | 33.14 | 265.12 Mbps | XCV300E-6BG432 |
| **Synth2 [9]** | 28.38 | 227.04 Mbps | XCV300E-8BG432 |
| **Proposed 8R** | 56 | 3.58 Gbps | XCV300E-8BG432 |
| **Proposed 2R** | 54 | 432 Mbps | XCV300E-8BG432 |

In [6], two architectures, the first with eight-stage pipeline and the second with two rounds, were proposed. Combinational logic and Lookup Table (LUT) are used in order to implement the S-BOXes. In [7], exactly the same two rounds architecture as in [6] is also proposed. In [8], two implementation versions are proposed. The first is the low power version (Type1) and the second is the high performance version (Type2) with the usage of four-stage pipeline. Finally, in [9] a hardware implementation that reduces the hardware constrain resources was presented. Two synthesis results are given. The first (Synth1) was made with speed grade –6, and the second (Synth2) was made with speed grade –8. As the Table 2 shows, the proposed KASUMI implementations outperform all of the previous implementations in terms of time performance.

## 5. CONCLUSIONS

In this paper hardware implementations of the 64-bit KASUMI block cipher are presented. The proposed implementations support both encryption and decryption operation opposed to the previously published implementations. Two different VLSI implementations (the first uses pipeline technique, and the second uses feedback logic) have been examined. By using inner-round and outer-round pipelining technique significant throughput improvement is achieved. The first achieves throughput value equal to 3584 Mbps at 56 MHz, and the second achieves throughput 432 Mbps at 54 MHz.

## 6. REFERENCES


[1] ETSI/SAGE. Specification of the A5/3 Encryption Algorithms for GSM and EDGE, and GEA3 Encryption Algorithm for GPRS, Document 1: A5/3 and GEA3 Specifications. ETSI/SAGE, May 2002.

[2] ETSI TS 148 018. Digital cellular telecommunications system (Phase 2+), General Packet Radio Services (GPRS). Base Station System (BSS) – Serving GPRS Support Node (SGSN), BSS GPRS Protocol. May 2002, on line available at http://webapp.etsi.org/action/PU/20020611/ts_148018v050300p.pdf

[3] 3GPP TS 35.205 V4.0.0, Technical Specification Group Services and System Aspects, 3G Security, Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*, Document 1: General, April 2001.

[4] *f8* and *f9* Specification, Specification of the 3GPP Confidentiality and Integrity Algorithms, Document 1, ETSI/SAGE, September 2000.

[5] KASUMI specification, Specification of the 3GPP Confidentiality and Integrity Algorithms, Document 2, ETSI/SAGE, December 1999.

[6] K. Marinis, N. K. Moshopoulos, F. Karoubalis, and K. Z. Pekmestzi, "On the Hardware Implementation of the 3GPP Confidentiality and Integrity Algorithms", Proceedings of the *4th International Conference for the Information Security, ISC 2001* Malaga, Spain, pp. 248-265, October 1-3, 2001.

[7] K. Marinis, N. K. Moshopoulos, F. Karoubalis, and K. Z. Pekmestzi, "An Area Optimized Hardware Implementation of the 3GPP Confidentiality and Integrity Algorithms", Proceeding of the *8th Conference on Optimization of Electrical and Electronic Equipment, OPTIM 2002*, Brasov, Romania, May 16-17, 2002.

[8] HoWon Kim, YongJe Choi, MooSeop Kim, and HeuiSu Ryu, "Hardware Implementation of 3GPP KASUMI Crypto Algorithm," *The 2002 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, Vol 1., pp. 317 - 320, July 16-19, 2002, Phuket, Thailand.

[9] Akashi Satoh, Sumio Morioka, "Small and High-Speed Hardware Architectures for the 3GPP Standard Cipher KASUMI", Proceedings of *the 5th International Conference Information Security, ISC 2002* Sao Paulo, Brazil, September 30 - October 2, 2002, Lecture Notes in Computer Science 2433 Springer 2002.

[10] Design Conformance Test Data. Specification of the 3GPP Confidentiality and Integrity Algorithms, Document 4, ETSI/SAGE, December 1999.

[11] Xilinx, San Jose, 2003, California, USA, Virtex, www.xilinx.com

[12] LeonardoSpectrum, Synthesis and Technology Manual. Software Release v2001.1, February 2001.