An Efficient Hardware Implementation of the KASUMI Block Cipher for Third Generation Cellular Networks

Tomás Balderas-Contreras
Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro #1 72840
Tonantzintla, Puebla. MEXICO
balderas@inaoep.mx

René A. Cumplido Parra
Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro #1 72840
Tonantzintla, Puebla. MEXICO
rcumplido@inaoep.mx

ABSTRACT

Third generation cellular network technology (3G) allows the transmission of information and voice at data rates never experienced before. 3G networks will revolutionize personal communications and information exchange between business partners in a more overwhelming fashion than 2G and 2.5 networks did. This revolution must be supported by a means to guarantee secure transmissions and transactions. The 3G UMTS (Universal Mobile Telecommunication System) networks include robust algorithms for confidentiality and integrity, both of them based on the KASUMI block cipher. This paper presents a FPGA implementation of KASUMI based on three design principles: the reuse of simple components to implement the whole cipher with fewer resources, the use of dual-port synchronous memories to implement the algorithm's substitution boxes (S-boxes), and the use of a simple key scheduler synchronized with a divide-by-two clock divider. The design achieves higher levels of performance with fewer resources, which is a must in order to succeed when producing components for the 3G cellular network market.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: standards; C.3 [**Special-Purpose and Application-Based Systems**]: real-time and embedded systems

General Terms

Design, Performance, Security

Keywords

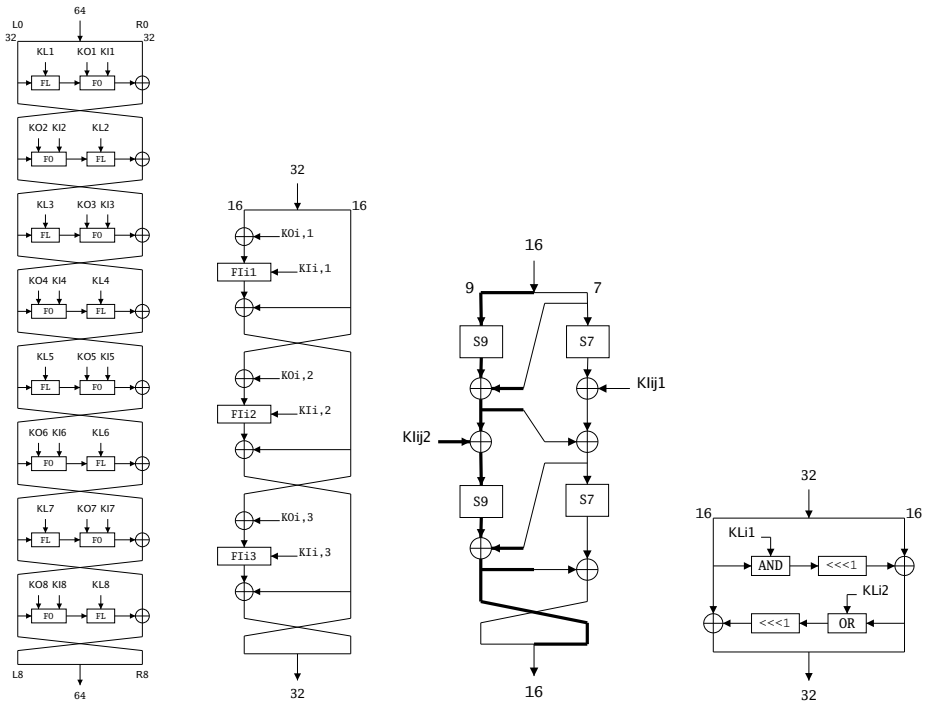
Block Cipher, KASUMI, FPGA

1. INTRODUCTION

The KASUMI block cipher was adopted by the 3rd Generation Partnership Program (3GPP) [1] as the cornerstone of several operations involved in the security architectures defined for the following cellular communication networks: the 3G UMTS standard, the 2G GSM (Global System for Mobile communications) standard, and the 2.5 GPRS (General Packet Radio Service) standard. In UMTS, KASUMI is the main component of both the *f8* confidentiality algorithm and the *f9* integrity algorithm [2]. KASUMI also lies at the core of the A5/3 and GEA3 encryption/decryption algorithms for GSM and GPRS, respectively [4].

KASUMI's specifications were developed based on previous work carried out for MISTY, an algorithm that has proven its high security levels against the most advanced cryptanalysis techniques and is suitable for hardware implementation [9]. KASUMI has a Feistel structure comprising eight rounds, operates on 64-bit data blocks, and a 128-bit encryption key K controls its processing [3]. Additionally, KASUMI has the following additional features derived from its Feistel nature: input plaintext blocks are the input to the first round, ciphertext blocks are the last round's output, the encryption key K is used to generate a set of round keys $\{KLi,\ KOi,\ KIi\}$ for each round i, each round computes a different function as long as the round keys are different, and the same algorithm is used both for encryption and decryption.

Figure 1 shows the structure and components of the KASUMI block cipher. For odd rounds the round-function is computed by applying the FL function followed by the FO function. For even rounds the FO function is applied before FL. FL, shown in figure $1d$, is a 32-bit function made up of simple AND, OR, XOR and left rotation operations. FO, depicted in figure $1b$, is also a 32-bit function having a three-round Feistel organization which contains one FI block per round. FI, see figure $1c$, is a non-linear 16-bit function having itself a four-round Feistel structure; it is made up of two nine-bit substitution boxes (S-boxes) and two seven-bit S-boxes. Figure $1c$ shows that data in the FI function flow along two different paths: a nine-bit long path (thick lines) and a seven-bit path (thin lines). Notice that in Feistel structures, such as the one used in this algorithm, each round's output is twisted before being applied as input to the following round. After completing eight rounds KASUMI produces a 64-bit long ciphertext block corresponding to the plaintext input block.



a. Feistel structure *b*. FO function *c*. FI function *d*. FL function

Figure 1: The KASUMI block cipher.

This document is organized as follows: section 2 describes each of the design techniques employed, section 3 provides implementation details, results, and a comparison with similar works, section 4 gives some comments concerning the use of the architecture in the field, and section 5 concludes.

2. ARCHITECTURE DESIGN

The goal of the proposed architecture is twofold: to address the requirement of high performance imposed by market, and to provide network manufacturers and operators with a compact design that integrates into network components with small penalties regarding power consumption and silicon area. This section describes the design techniques employed to achieve the goal.

The reutilization approach conceived for this project is illustrated in figure 2. Consider the parallel version of the FO module depicted in figure 2*a*, it can be noticed that the upper section and the lower section have some components in common. The first technique consists in the addition of components in each section, which do not modify the functionality of the module, in order to make the two sections structurally identical. Figure 2*b* shows that two XOR gates with zero input have been included along the module's datapath. The datapath in figure 2*c* is the result of inserting an additional FI module in the lower section, and two multiplexers as well. Notice that at this step the datapath's organization has two sections having exactly the same number and kind of components, differing only in their input values. The simplified datapath in figure 2*d* implements only one of these sections, two iterations over this datapath are required in order to carry out the processing of a FO function module, and the input values of the components are selected depending on the iteration in process by means of multiplexers. The datapath performs an iteration every clock cycle; consequently, two cycles are needed to carry out the FO function module.

Consider now two parallel instances of the FI function module depicted in figure 1*c*. The second technique merges these two FI modules into one in order to use the least number of S-boxes possible, these S-boxes are then mapped to the memory blocks (SRAMs) embedded in the FPGA platform. Figure 3 depicts the result of the combination technique. Notice that instead of having four S9 S-boxes and four S7 S-boxes, the new module contains only two dual-port S9 S-boxes and two dual-port S7 S-boxes. This dual-port module is required to work during a clock cycle; that is why the upper S-boxes are synchronized with the negative edge of the clock pulse, whereas the lower S-boxes are synchronized with the positive edge of the clock pulse. The additional registers are used to synchronize input data with the S-boxes' outputs. The simplified FO module in figure 2*d* already includes the dual-port FI module.

In order to perform a whole round, some logic must be added to the FO module just described. The datapath in figure 4 includes the FO module, two FL modules, of which only one is used per round, some registers for data synchronization, and multiplexers for data selection. This module requires 16 clock cycles to carry out the ciphering process, and that each set of round keys is available during two clock cycles.

The key scheduler module generates the set of round keys

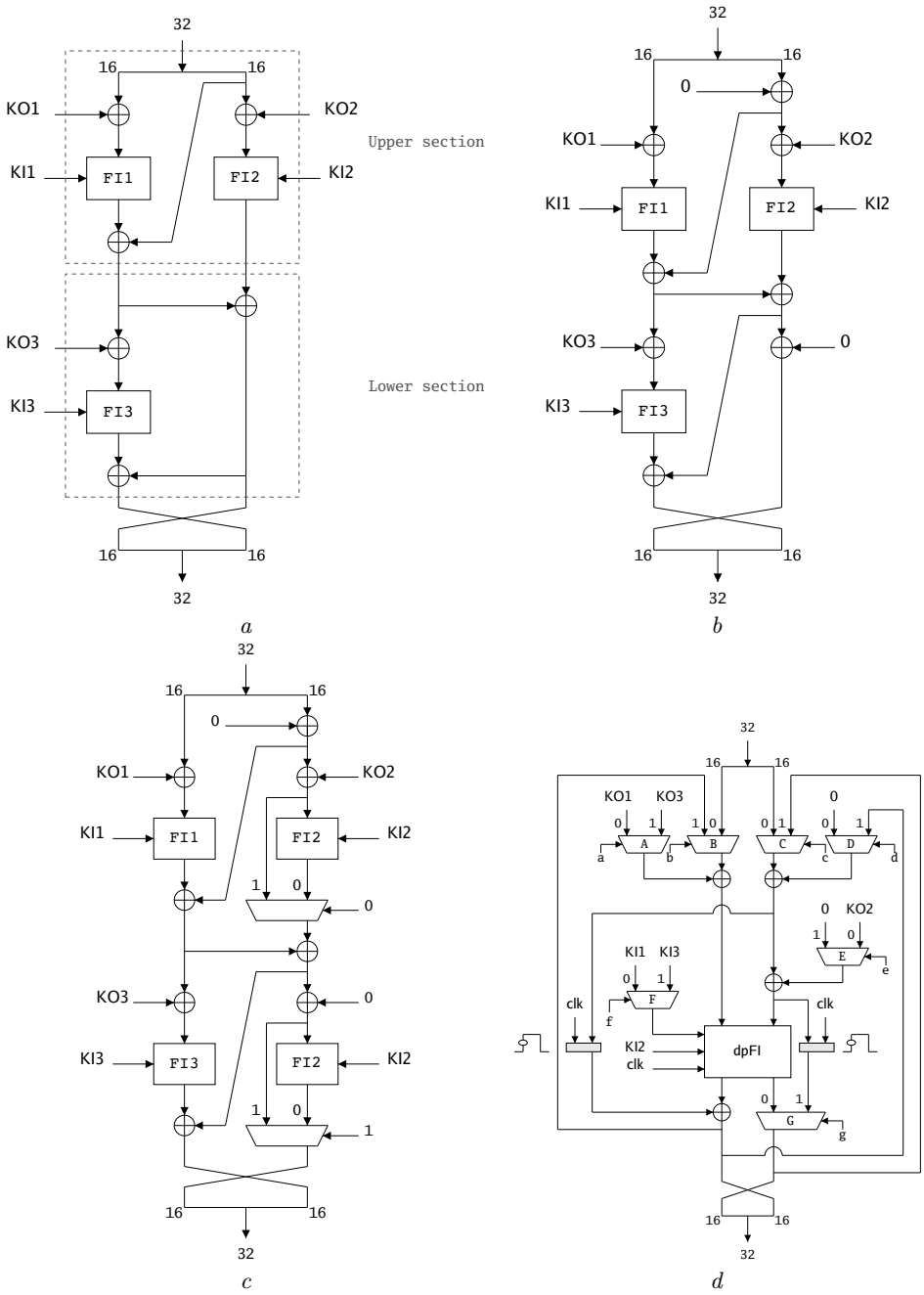


Figure 2: Steps to design a reusable FO module.

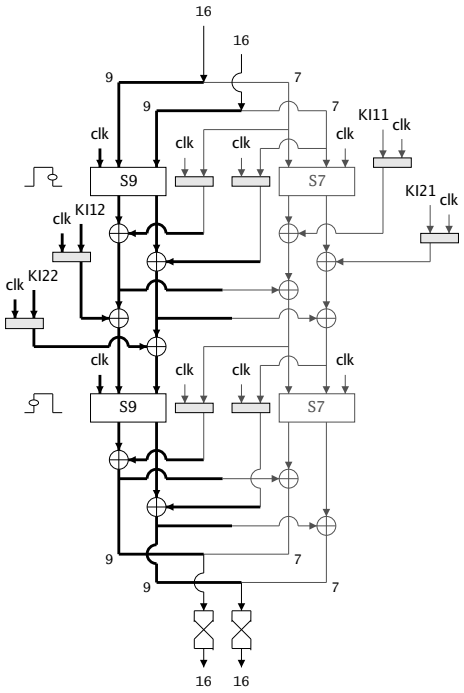


Figure 3: The dual port FI module.

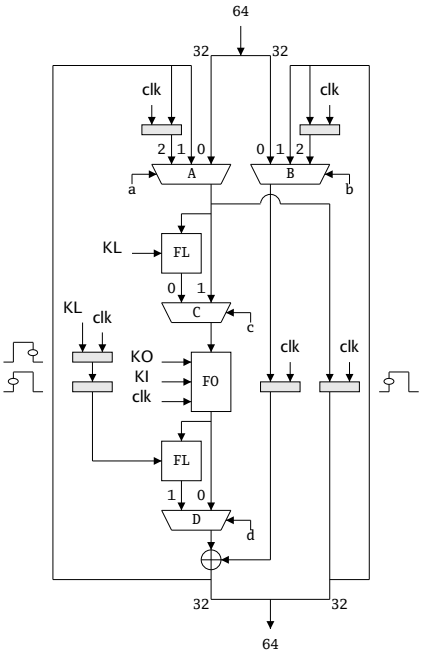


Figure 4: Datapath for the round logic.

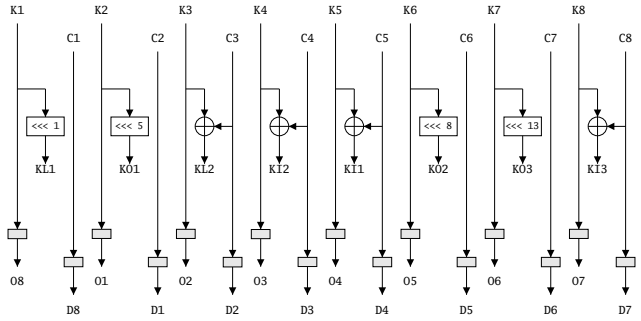


Figure 5: The key scheduler.

for each round. As mentioned above, it is a requirement that this component keeps each set of values active during two clock cycles in order for them to be available during the processing period of the round logic. Figure 5 shows the organization proposed for the key scheduler, which receives the encryption key K as an array of eight 16-bit subkeys K_i, along with an array of constants C_i. The module synchronously outputs both of the arrays rotated once to the left, these outputs are in turn fed back to the input ports. In order to keep the design simple and to meet its requirement, the scheduler is synchronized with a divide-by-two clock frequency divider.

3. IMPLEMENTATION

The design is implemented using the VHDL language, the Xilinx Synthesis Technology (XST) tools, and a Virtex-E FPGA platform [11] in order to make fair comparisons with related works. Table 1 shows the results yielded by the synthesis process for a XCV300E-8-BG432 FPGA. Notice that the percentages of use of the massive elements in the FPGA do not surpass 20% for this device, which contains few resources and is one of the smallest of its family. This fact is an indication of the compactness of the design.

Table 2 shows that the design described in this document achieves the second highest throughput and, at the same time, is one of the most economical designs in terms of area in hardware. The design reduces dramatically the number of S-boxes needed, by a factor of 24, to only four; six SRAM blocks are used though. Since each S9 S-box is $512 \times 9 = 4608$-bit long, and each SRAM block can only store 4096 bits, then two SRAM blocks are used to hold each S9 S-box. S7 S-boxes fit well in a SRAM block.

It is possible to perform a more aggressive simplification of the KASUMI datapath; however, this is not advantageous due to more cycles will be needed to complete the ciphering process, which has such a negative effect on performance. Notice in table 2 that the implementations with longer latencies have also lower performances.

4. POTENTIAL USES

The design is suitable to be included into mobile handsets because it is compact enough to save silicon space and has low power consumption, the two most important aspects that need to be consider when designing components for a mobile device such a cell phone or a PDA. The architecture might be implemented either in a smart card (USIM) or as

Table 1: Synthesis results

Category	Amount of elements used	Total amount of elements available	Percentage of use
Slices	488	3072	15%
Slice Flip Flops	566	6144	9%
4-input LUTs	898	6144	14%
SRAMs	6	32	18%
GCLKs	1	4	25%

Table 2: Comparison with other implementations using the reuse approach

Proposal	Latency (cycles)	Area (slices)	Frequency (MHz)	Throughput (Mbps)	Number of S-boxes	Number of block SRAMs
Work in [6]	8	650	20	110	12	N/A †
Work in [8]	40	749	35.35	71	24	24
Work in [10]	56	368	68.13	78	2	N/A †
	32	370	58.06	117	4	N/A †
	8	588	33.14	266	12	N/A †
This work	16	488	41.14	165	4	6

†: N/A = Information not available

a functional unit of a low power processor for embedded systems.

The Radio Network Controller (RNC) module [5], which is located in the radio access network and controls a set of base stations (Node Bs), also implements its own KASUMI ciphering module. In this case, the architecture's performance allows the RNC to cope with the multiple encryption/decryption requests from the different users of the network. Several architectures for the RNC may be proposed, since it is a non-standardized component of the network, as well as different ways to organize the KASUMI components in order to achieve higher performances.

5. CONCLUSIONS

Network operators and component manufacturers have great expectations towards the deployment and use of 3G networks in the upcoming years. The huge number of potential subscribers and the advanced services to provide impose great challenges in terms of guaranteeing confidentiality and integrity of both data and signaling. An efficient and compact hardware design of the KASUMI algorithm was described in this document, along with the results of its implementation in FPGA technology. The design techniques used (a reutilization approach, the use of dual-port embedded memory blocks and the inclusion of a divide-by-two clock divider) have proved to be very useful; these techniques might be utilized to design high performance compact implementations of Feistel-like block ciphers. Not only does this proposal achieve a good performance, the second highest throughput, but is one of the most economical designs in terms of area. The architecture meets the needs of any manufacturer looking for a high performance ciphering architecture which is compact enough to save space and has low power consumption.

6. ACKNOWLEDGEMENTS

This work was sponsored by the scholarship number 171498 granted by CONACyT, the Mexican Council for Science and Technology.

7. REFERENCES

[1] 3rd Generation Partnership Program. 3GPP Home Page. http://www.3gpp.org

[2] 3rd Generation Partnership Program. Document 1: *f8* and *f9* Specification 35.201. Release 5. Version 5.0.0.

[3] 3rd Generation Partnership Program. Document 2: KASUMI Specification. Technical Specification 35.202. Release 5. Version 5.0.0.

[4] 3rd Generation Partnership Program. Document 1: A5/3 and GEA3 Specifications. Technical Specification 55.216. Release 6. Version 6.2.0.

[5] K. Ito, et al. *Radio Network Control System*, in FUJITSU Scientific & Technical Journal, 2002, 38(2), pp. 174–182.

[6] H. Kim, Y. Choi, M. Kim and H. Ryu. "Hardware Implementation of the 3GPP KASUMI Crypto Algorithm", *in Proc. of the 2002 International Technical Conference on Circuits/Systems, Computers and Communications ITC-CSCC-2002*, 2002, pp. 317–320.

[7] P. Kitsos, M. D. Galanis and O. Koufopavlou. "High-Speed Hardware Implementations of the KASUMI Block Cipher", *in Proc. of the 2004 IEEE International Symposium on Circuit and Systems ISCAS'04*, 2004.

[8] K. Marinis, N. K. Moshopoulos, F. Karoubalis and K. Z. Pekmestzi, "On the Hardware Implementation of the 3GPP Confidentiality and Integrity Algorithms", *in Proc. of the 4th International Conference on Information Security ISC 2001*, 2001, pp. 248–265.

[9] M. Matsui, "New Block Encryption Algorithm MISTY", *in Proc. of the 4th International Fast Software Encryption Workshop FSE'97*, 1997, pp. 54–68.

[10] A. Satoh and S. Morioka, "Small and High-Speed Hardware Architectures for the 3GPP Standard Cipher KASUMI", *in Proc. of the 5th International Conference on Information Security ISC 2002*, 2002, pp. 48–62.

[11] Xilinx, Inc., *Virtex-E 1.8 V Field Programmable Gate Arrays. v2.6. Product Specification*, 2002.