On the Hardware Implementation of the MICKEY-128 Stream Cipher

Paris Kitsos[2]

ABSTRACT

Encryption algorithms are becoming more necessary to ensure the securely transmitted data over insecure communication channels. MICKEY-128 is a recently developed stream cipher with two major advantages: (i) the low hardware complexity, which results in small area and (ii) the high level of security. FPGA device was used for the performance demonstration. Some of the first results of implementing the stream cipher on an FPGA are reported. A maximum throughput equal to 170 Mbps can be achieved, with a clock frequency of 170 MHz.

1. INTRODUCTION

The new applications that are generated create new problems in communication systems. Many times these systems demand high-speed requirements and other times demand low hardware complexity. In wired applications, for example ISDN, these systems usually have to operate with high-speed. In wireless applications, for example mobile phones, the systems require low hardware resources. In all cases, the most critical parameter is the security level that provided by the encryption algorithms are used by the communication systems.

The continuous growing of mobility necessitate to the scientists to design new encryption algorithms with special care in speed, security and simplicity. The simplicity of the algorithms design is major factor that is simple in software implementation but in the hardware implementation might be quite complex. RFID tags, Smart cards and Bluetooth are typical examples of products where the amount of memory and power is very limited. The hardware implementations of today's algorithms, such as AES cipher, are inefficient for devices with limited hardware area. So, stream ciphers are used in cases that the low hardware complexity is necessitated.

Figure 1 shows the general diagram of the cipher process with stream cipher. The stream cipher usually take two parameters, the secret key, K, and the initialization vector, IV, and produce keystream bits, z_t. In stream encryption each plaintext symbol, P_t, is encrypted by applying a group operation with a keystream symbol, z_t, resulting in a ciphertext symbol c_t. In modern cipher the operation is the simple bitwise XOR.
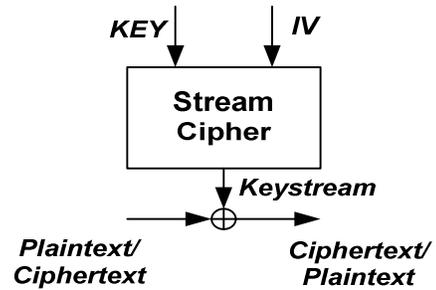

Figure1: The Cipher Process

Decryption takes the subtraction of the keystream symbol from the ciphertext symbol. With the bitwise XOR this is the same operation:

$$m^t = c^t \text{ Å } z^t$$

In this paper the implementation on hardware of a new stream cipher called MICKEY-128 [1] is investigated. This cipher has been submitted and it has been under consideration from the ECRYPT (European Network of Excellence for Cryptology), project [2]. The stream cipher MICKEY-128 (which stands for Mutual Irregular Clocking KEYstream generator) with a 128-bit key is aimed at area-restricted hardware environments where a key size of 128 bits is required.

The following of this document is structured as follows: After an introduction of MICKEY-128 cipher, the hardware design and architecture approach are presented. In this section, synthesis results and comparisons with previous published stream ciphers are given. Finally, section 4 concludes the paper.

2. MICKEY-128 STREAM CIPHER

MICKET-128 stream cipher is intended to have low complexity in hardware, while providing a high level of security. It uses irregular clocking of shift registers, with some novel techniques to balance the need for guarantees on period and pseudorandomness against the need to avoid certain cryptanalytic attacks.

MICKEY-128 takes two input parameters, the 128-bit secret key, K, and an initialization variable, IV, anywhere between 0 and 128-bit in length. Two 128-bit registers R and S are used in order to build it. The R register is a *Linear Feedback Shift Register* (*LFSR*) and the S register is a *Non-linear Feedback Shift Register* (*NFSR*). Two

[2]Digital Systems and Media Computing Laboratory
School of Science & Technology
Hellenic Open University
Patras, Greece
e-mail: pkitsos@ieee.org

variables are defined. The *Control_bit_R* and the *Control_bit_S*. The *Control_bit_R* is defined as s_{43} xor r_{85} and the *Control_bit_S* is defined as s_{85} xor r_{42} where s_{43}, s_{85}, r_{42} and r_{85} are 42nd bit and 85th bit of the register *R*, the 43rd bit and 85th bit of the register *S*. When *Control_bit_R=0* the register *R* is a standard LFSR as the figure 2 shows.
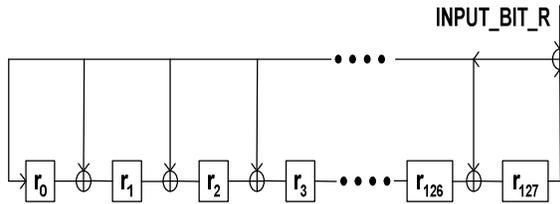

Figure2: The Register *R* with *Control_bit_R=0*

When *Control_bit_R=1*, as well as shifting each bit in the register to the right, we also XOR it back into the current stage, as shown in Figure 3. This corresponds to multiplication by *x+1*.
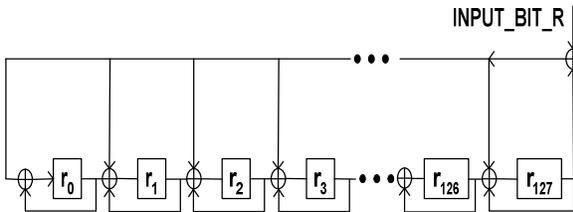

Figure3: The Register *R* with *Control_bit_R=1*

The figure 4 shows the design of the *NFSR S* register. The variables $FB0_i$ and $FB1_i$, the transformation *tr* and the mathematical equations that the register *S* uses for each building can be found in reference [1].
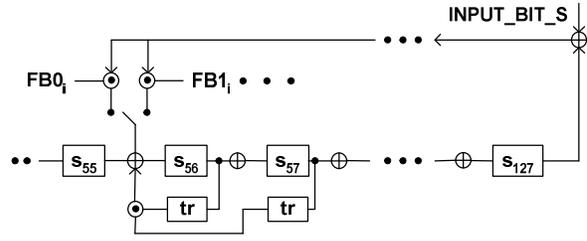

Figure4: The Register *S*

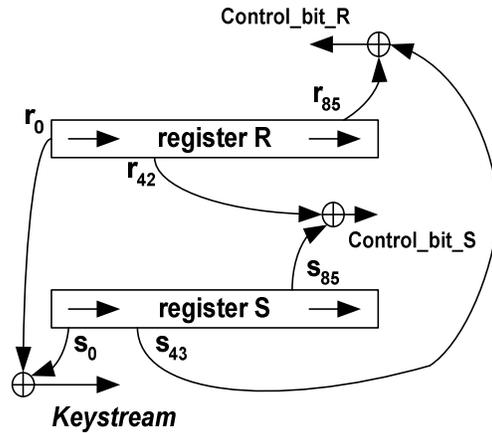The overall diagram of MICKEY-128 stream cipher is shown in figure 5.

Figure5: The MICKEY-128 Block Diagram

3. ARCHITECTURES

3.1. Design Implementation

MICKEY-128 has been designed with dedicated hardware implementation. The architecture that performs the MICKEY-128 stream cipher is shown in figure 6. This architecture consists of the registers *R* and *S* following the specifications demand. It is obvious the way that the *Control_bit_R* and the *Control_bit_S* variables are defined. Also, the multiplexer MUX it is shown that is needed in order to the data values are forced in the registers. Finally, the keystream bits are produced by the XORing of the r_0 and s_0 bits.
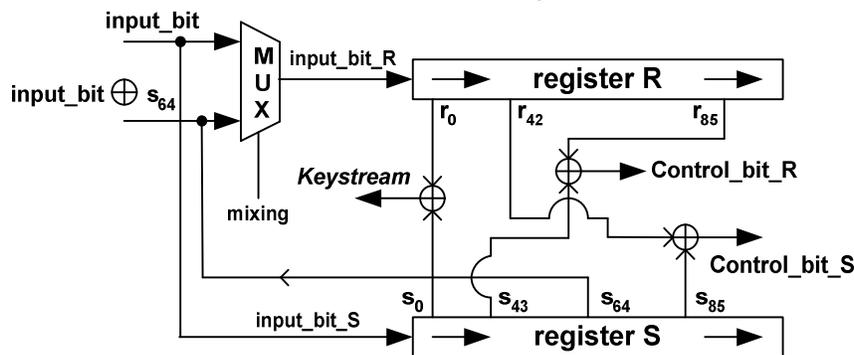

Figure6: The MICKEY-128 Stream Cipher Architecture

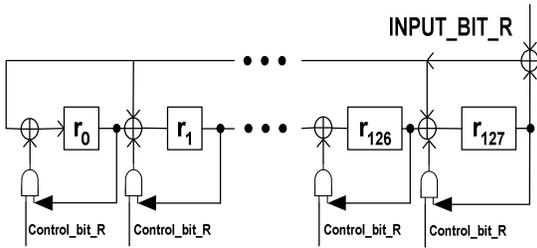The register *R* implementation is illustrated in figure 7.

INPUT_BIT_R

Figure7: The Implementation of Register *R*

The *Control_bit_R* signal helps each AND gate in order to configure the register and works either in normal mode or multiplied the current stage with *x+1* polynomial.

For the register *S* a straightforward implementation was used.

3.2. Design Results

The proposed architecture (Figure 6) was captured by using VHDL with structural description logic. The VHDL code was synthesized for Xilinx (Virtex) FPGA device [3]. The synthesis results and performance analysis are shown in Table 1 indicating the number of D Flip-Flops (DFFs), Configurable Logic Blocks (CLBs) slices and Function Generators (FGs).

Table 1: Synthesis Results

Resources	Use	Available	Utilization
Device	XCV50ECS144		
I/Os	6	94	5 %
FGs	333	1536	21.6 %
CLB slices	167	768	21.7 %
DFFs	235	1818	12.9 %
Freq. F (MHz)	170		
Throughput (Mbps)	170		

The throughput is estimated after the initialization phase. The smallest FPGA device with low hardware resources utilization by the FPGA family was used. Similar measurements may be done with different FPGA families.

Performance comparisons between the proposed system and previous published architectures are shown in Table 2. No other implementation of the MICKET-128 stream cipher has been previously published. So, comparisons with others synchronous stream ciphers [4-6] are given in order to have a fair and detailed comparison of the proposed system.

In [4], a hardware implementation of the well-known A5/1 cipher is presented, which is used in GSM mobile phones. The W7 stream cipher in [4] is a synchronous stream-cipher optimized for efficient hardware implementation at very high data rates. Finally, the RC4

is used in IEEE 802.11b. In [5], the E0 algorithm that Bluetooth system used is presented.

Table 2: Hardware Performance Comparisons

Stream Cipher	FPGA Device	F (MHz)	Throughput (Mbps)
A5/1 [4]	2V250FG25	188.3	188.3
W7 [4]	2V250FG25	96	768
RC4 [4]	2V250FG25	60.8	121
E0 [5]	2V250FG25	189	189
WG [6]	ASIC	1000	125
Proposed	XCV50ECS14	170	170

In [6] the hardware implementation of a new stream cipher, the WG, is shown. This cipher has been shown in the latest stream cipher workshop [2].

As the above table illustrates the proposed cipher implementation achieves competitive frequency and some times better performance compared with the others. All in all the cipher achieves a low level of FPGA utilization, complimentary hardware efficiency and its synthesis results proves that is suitable for area restricted hardware implementations.

4. CONCLUSIONS

An efficient hardware implementation of the new stream cipher MICKEY-128 was presented in this paper. This cipher has been submitted and has been under consideration from the ECRYPT project. Two are the major advantages of the cipher: (i) the low hardware complexity and (ii) provide high level of security. The synthesis results prove that the MICKEY-128 cipher is suitable for FPGA implementation. In addition, the proposed implementation is suitable for migrations to other technologies, such as smart cards or RFID tags.

5. REFERENCES

[1] Steve Babbage, Matthew Dodd, "The stream cipher MICKEY-128", (ECRYPT) Stream Cipher Project Report 2005/016.

[2] ENCRYPT - European Network of Excellence in Cryptology, "Call for Stream Cipher Primitives", Scandinavian Congress Center, Aarhus, Denmark, 26-27 May 2005, http://www.ecrypt.eu.org/stream/

[3] Xilinx, San Jose, California, USA, Virtex, 2.5 V Field Programmable Gate Arrays, 2005, www.xilinx.com.

[4] M. D. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, and C. E. Goutis, "Comparison of the Hardware Implementation of Stream Ciphers", accepted for publication in The International Arab Journal of Information Technology (IAJIT), Colleges of Computer and Information Society, 2005.

[5] P. Kitsos, N. Sklavos, K. Papadomanolakis and O. Koufopavlou, "Hardware Implementation of Bluetooth Security", IEEE Pervasive Computing, vol. 2, no.1, pp. 21-29, January-March 2003.

[6] D. Gligoroski, S. Markovski, L. Kocarev and M. Gusev, "*Edon80* - Hardware Synchronous Stream Cipher", Symmetric Key Encryption Workshop (SKEW), Scandinavian Congress Center, Aarhus, Denmark, 26-27 May 2005.