Comparison of FPGA-Targeted Hardware Implementations of eSTREAM Stream Cipher Candidates

David Hwang, Mark Chaney, Shashi Karanam, Nick Ton, and Kris Gaj

Department of Electrical and Computer Engineering
George Mason University, Fairfax, Virginia, U.S.A.
{dhwang, mchaney, skaranam, nton1, kgaj}@gmu.edu

Abstract. This work discusses FPGA hardware implementations of all eSTREAM phase 3 hardware stream cipher candidates (profile 2) and some of their derivatives. The designs are optimized for maximum throughput per unit area as well as minimum area, and targeted for Xilinx Spartan 3 FPGAs. The results have found that the Grain and Trivium families of ciphers have demonstrated relative implementation efficiency compared to the rest of the cipher candidates; Mickey also provided a balance of low area with high throughput per area.

1 Introduction

Efficient hardware implementations of stream ciphers are important in both high-performance and low-power applications which require encryption. To this end, in 2004 the ECRYPT stream cipher project (eSTREAM) [1] was launched to identify new stream ciphers for adoption in a variety of systems; the call for profile 2 stream cipher candidates in particular focused on stream ciphers suited toward hardware implementation. Currently in phase 3 of the eSTREAM competition there are eight families of hardware-oriented stream ciphers which remain as candidates: DECIM, Edon80, F-FCSR, Grain, MICKEY, Moustique, Pomaranch, and Trivium. The general characteristics of these ciphers and their variants are shown in Table 1. In this paper we compare the hardware implementations of all hardware-oriented candidates (profile 2) and their variants, targeted toward the Xilinx Spartan 3 family of FPGAs. The source of these implementations was a capstone course project for a graduate course on digital system design at George Mason University. As part of the course, each student was assigned to one cipher variant and asked to implement the cipher from specification and algorithmic state machine to post-place and route simulation on FPGA, verifying each step against the C test vectors provided on the eSTREAM website. At least three students were assigned per cipher variant; after the course was completed, the authors further optimized the best code for each cipher and performed further verification. The remainder of the paper is partitioned as follows. Section 2 describes the specification, design methodology, and tool flow for the cipher implementations. Section 3 discusses cipher optimizations, both

general and cipher-specific. Section 4 presents implementation results for each cipher on the Xilinx Spartan 3 FPGA family of devices. Section 5 presents prior art, and Section 6 concludes the paper.

Table 1. Table of the eSTREAM hardware candidates (profile 2) and their variants

Candidate	Key Size (bits)	Max IV Size (bits)	Data radix d (bits/cycle)	Separate key register	Separate IV register
DECIM v2	80	64	0.25	no	yes
DECIM 128	128	128	0.25	no	no
Edon80	80	64	1	yes	yes
F-FCSR-H v2	80	80	8	no	no
F-FCSR-16	128	128	16	no	no
Grain v1	80	64	1	no	no
Grain 128	128	96	1	no	no
MICKEY 2.0	80	80	1	no	no
MICKEY 128 2.0	128	128	1	no	no
Moustique	96	104	1	yes	no
Pomaranch	80 /128	108 /162	1	yes	yes
Trivium	80	80	1	no	no

2 Design Methodology

2.1 Specification

The specification of the system is shown in Fig. 1. The operation of the system begins with a system reset, after which the cipher asserts the *key_iv_ready* signal. If the cipher can implement separate encode and decode modes (e.g. Moustique), the user sets the *enc_dec* signal accordingly. The user asserts the *key_iv_write* signal and commences the loading of the key data and the IV (initial value) data k-bits at a time through the *key_iv* port. The IV is always loaded with the maximum size permitted. After the cipher has read in the required data, it sets the *key_iv_ready* flag low and begins the cipher initialization process which varies by cipher and which can take a few hundred clock cycles to complete. Upon completion of the initialization process, the cipher asserts the *data_in_ready* flag, indicating it is ready to process streaming data. The user asserts the *data_in_write* flag and loads the plaintext *data_in* d-bits at a time into the cipher. The cipher asserts the *data_write* flag when the ciphertext *data_out* is valid, and outputs the data d-bits per clock cycle. This general specification was used for all ciphers with some cipher-specific modifications if necessary. An example waveform of the cipher operation is given in Fig. 2.

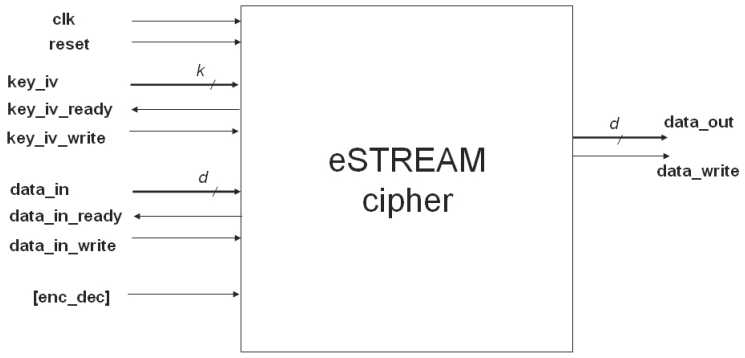

Fig. 1. eSTREAM cipher specification

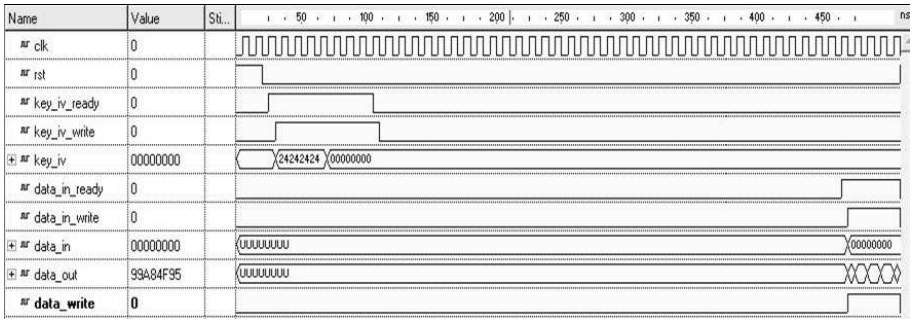

Fig. 2. Example waveform from a cipher designed according to specification

2.2 Data Radix and Key Radix

As noted in the previous section, the data is input into the cipher d bits per clock cycle and the key is input into the cipher k bits per clock cycle. The value of d is referred to as the data radix and affects the throughput of the cipher; the value of k is referred to as the key radix and affects the programming latency (the time it takes to input the key and IV into the cipher) of the system. Each of the eight cipher families (and their variants) has a natural value of d which can be determined from the authors' specification. This natural value is shown for each cipher on Table 1. The value of k is largely independent of the value of d. A large value of k would reduce the programming latency but potentially increase area and the required pin count. A smaller value of k would increase programming latency but potentially reduce area. Unless otherwise noted, to balance these two issues the value of k was selected to be equal to d for all ciphers.

There are a few exceptions to the values of d and k. For DECIM v2 and DECIM 128, the ciphers output a decimated output, in which one bit of output

is produced every four clock cycles. Hence the natural value of d is 1 bit per 4 clock cycles = 0.25 bits/clock cycle; in this case k was chosen as 1 bit/cycle. Two of the cipher families can also be implemented using parallel lookahead techniques to increase the data radix dramatically with reasonable hardware increases. Specifically, Grain v1 can be implemented to produce d = 16 bits/cycle, Grain 128 can be implemented to produce d = 32 bits/cycle, and Trivium can be implemented to produce d = 64 bits/cycle. These variants of the ciphers were also implemented in this paper. For fair comparison in each of these cases, the new k value was set to the new d value.

2.3 Tool Flow and Optimization Criteria

The design flow began with the design of an algorithmic state machine (ASM), followed by the partitioning the design problem into a datapath (including counters, muxes, etc.) and a controller (a finite state machine). After this process was complete, the design was implemented in synthesizable VHDL and verified via functional simulation against the C test vectors using Aldec Active-HDL or Mentor Graphics Modelsim. Synthesis was performed using Synplicity Synplify Pro or Xilinx XST and post-synthesis simulation was performed. Finally, implementation into FPGA was performed using Xilinx ISE and post-place and route simulation and static timing analysis was completed. The tools are shown in Table 2.

The ciphers were targeted for Xilinx Spartan 3 FPGAs, a 90 nm family of low-cost FPGAs. The smallest device in the Spartan 3 family that could fit the particular cipher variant was chosen. The designs were optimized for maximum throughput per area measured in Mbps / slice, as well as minimum area measured in slices as primary design criteria.

Table 2. Table of FPGA implementation tools

Design Step	Tool
VHDL simulation	Aldec Active-HDL 7.2, ModelSim SE 6.3a
FPGA synthesis	Synplicity Synplify Pro 8.6, Xilinx XST 9.1
FPGA implementation	Xilinx ISE 9.1
Target FPGA	Xilinx Spartan 3

2.4 Verification

Verification was performed on all ciphers against the C code provided on the eSTREAM website. It was often the case, due to the C implementation, that the results were bit-reversed within each byte. For example, a typical output in C would be F09B in hexadecimal and the equivalent VHDL output would

be 0FD9. The byte-reversing or bit-reversing generally differed from cipher to cipher. Some ciphers had minor discrepancies between the submitted C code and the published cipher specification. In those cases, we chose to bit match with the C code. DECIM 128 used 32 bits of buffer in the submitted C code, but 64 bits in the specification. Pomaranch was matched against the C code instead of the specification (since the type 1 jump register for the first three cells was slightly different).

3 Design Optimizations

3.1 General Optimizations

The ciphers were optimized for implementation on Spartan 3 FPGAs. General optimization steps for the ciphers included pipelining, retiming, different variants of state machine encoding (one-hot, gray coding, etc.), clocking FSM outputs, and modifying options on synthesis and implementation tools. Architectural improvements were also made on a cipher-specific basis. In particular, some ciphers did not require internal storage of the key and IV values. In other words, the key and IV values could be shifted directly into the Linear Feedback Shift Register (LFSR), Non-Linear Feedback Shift Register (NFSR), or other memory location and did not need separate storage registers. The ciphers to which this apply are shown in Table 1. Since the devices were targeted onto Spartan 3 FPGAs, a specific optimization used was the inference of Xilinx shift registers (SRL16) [2]. The SRL16 component allows a 16-bit shift register to be implemented as a single LUT. Thus a 32-bit shift register could take only 1 slice (2 LUTs), rather than 16 slices, a drastic savings. However, this savings can only take place when a true shift register is an internal component (i.e. modeling an LFSR or NFSR). If combinational logic is required between internal cells in a shift register, often the inference to SRL16 is not possible. In addition, when intermediate outputs of the shift registers are required, the full 16-bit register is not inferred; rather, a shorter register is inferred.

3.2 Cipher-Specific Architectures and Optimizations

DECIM The DECIM family of ciphers is composed of DECIM v2, which uses an 80-bit key and a 64-bit IV, and DECIM 128, which uses a 128-bit key and 128-bit IV. The DECIM ciphers are based on an LFSR feeding data to a Boolean function F, which in turns inputs data into a block called the ABSG. The ABSG serves to decimate data, producing on average one bit per four clock cycles, which is smoothed out by an external buffer. Optimizations made on the DECIM family of ciphers included removing a separate storage register for the key, reducing the size of separate storage of IV (removing it for DECIM 128), and implementing the F function as a sum rather XOR chains. Coding the LFSR in VHDL to infer SRL16 Spartan 3 shift register cells was also implemented. The buffer was also coded in VHDL to infer shift registers.

Edon80 The Edon80 cipher is a binary additive stream cipher with an 80-bit key and a 64-bit IV. It operates with a unique architecture using pipelined stages of 80 simple 2-bit transformers called e-transformers. The output of the cipher occurs two bits per cycle, of which one is discarded, effectively causing the input and output data rate to be d = 1 bit/cycle. As shown in [3], these stages can be implemented in an iterative architecture for low area. In this paper, we implement the original pipelined approach. Architectural improvements for the Edon80 cipher included adding pipeline registers in the datapath for selection and mux logic. The quasigroup operations were implemented as ROM lookup tables.

F-FCSR The F-FCSR family of ciphers is composed of F-FCSR-H v2, which uses an 80-bit key and an 80-bit (maximum) IV and produces d = 8 bits/cycle, and F-FCSR-16, which uses a 128-bit key and a 128-bit (maximum) IV and produces d = 16 bits/cycle. The core of these ciphers is the Feedback with Carry Shift Register (FCSR), which resembles a shift register with a full adder and a carry storage block between flip-flops. The output is created by a filter which uses AND and XOR functions. Architectural optimizations for F-FCSR ciphers included removing separate key and IV registers by using the same register to hold the key and IV values as the pseudorandom S bytes after cipher initialization.

Grain The Grain family of ciphers includes Grain v1, which uses an 80-bit key and a 64-bit IV, and Grain 128, which uses a 128-bit key and a 96-bit IV. The Grain ciphers are simple in their architecture, with primary components being an LFSR, and NFSR, and another non-linear Boolean function. Architecture improvements implemented in our design include not implementing separate key and IV registers and coding the NFSR and LFSR shift registers to infer SRL16 shift register cells. In addition, Grain can be made to produce a higher radix output using a simple lookahead technique described in [4]. Thus, we implemented a d = 16 bits/cycle version of Grain v1 and a d = 32 bits/cycle version of Grain 128 as well.

MICKEY The MICKEY family of ciphers includes MICKEY 2.0, which uses an 80-bit key and 80-bit (maximum) IV, and MICKEY 128, which uses a 128-bit key and 128-bit (maximum) IV. MICKEY stands for Mutual Irregular Clock Keystream Generator, and uses an R register which is "linear" and an S register which is "non-linear". The R and S registers consists of registers with combinational logic between each flip-flop, in either feedforward and feedback modes. Architectural optimizations included not implementing separate key and IV registers, reducing logic on counter compare signals, and balancing logic and flip-flop use.

Moustique Moustique is a self-synchronizing cipher, which differs from the other ciphers which are binary additive, and uses an 80-bit key and 104-bit IV.

The basic structure of Moustique uses a conditional complementing shift register (CCSR) and a number of pipelined stages with Boolean functions between each stage. The cipher has separate encrypt and decrypt modes of operation. Optimizations on Moustique include not implementing a separate IV register and various CAD tool optimizations.

Pomaranch The Pomaranch cipher can be implemented either as a 128-bit key and 162-bit (maximum) IV architecture, or an 80-bit key and 108-bit (maximum) IV architecture. This paper examines the 80-bit key version only. Pomaranch with an 80-bit key consists of 6 jump registers, whose outputs cascade from one register to another in a chain; hence, the cipher is also referred to as a cascade jump controlled sequence generator. The outputs of each jump register are input into an S-Box and xored by the key to producing the cascading output of the jump register. Architecture improvements to Pomaranch primarily revolved around the implementation of the S-Box. Initially the S-Box was implemented using large lookup tables; during the optimization phase, the S-Box was implemented using reduced hardware based on composite fields and explicit Boolean operations in [5]. This optimization actually reduced the speed of the design but improved area; this will be described in the results section.

Trivium The Trivium cipher uses an 80-bit key and an 80-bit IV. It has a simple structure with a number of LFSR and NFSR registers feeding into one another. Due to the structure, the architecture was optimized to not implement separate key and IV registers and to infer SRL16 registers as much as possible. Trivium also had the advantage of implementing a higher radix (d = 64 bits/cycle) implementation by minimal hardware duplication [6].

4 Results

The results of the Xilinx Spartan 3 FPGA implementations are shown in Table 3. The results show area in slices after place-and-route using the CAD tools described earlier. The maximum clock frequency was determined by the Xilinx static timing analysis tool. The post place-and-routed designs were simulated at a baseline clock frequency for functional verification against the C vectors. A brief overview of each cipher family follows. The DECIM ciphers produced low area implementations due to the simple LFSR structure; however, the throughput was low due to the decimation factor of four. Edon80 was the largest design of the implemented ciphers. The F-FCSR family of ciphers were somewhat large (342 slices and 473 slices) compared to the smallest ciphers, but due to the high data radix (8 bits/cycle and 16 bits/cycle), the throughput and throughput/area was relatively high. Grain was a top performer in terms of small area and good throughput/area ratio. It was the smallest cipher and the parallelized versions of Grain produced higher throughput/area ratios. Mickey had a medium size area but a good throughput/area ratio; the main disadvantage Mickey had

in Xilinx FPGAs were that the S and R registers could not be inferred into Xilinx primitive shift register blocks; thus Mickey in an ASIC implementation may yield better results when compared to the other small ciphers. The same could be said with the F-FCSR family of ciphers. Moustique was of medium-to-large area with a less than one ratio of throughput/area from our design. Moustique was the only self-synchronizing cipher so this should be mentioned in the comparison. Pomaranch was the slowest design and yielded a high area. An implementation using a lookup table of the S-Box was faster (68 MHz) but also larger (1155 slices). Trivium was another top performer in terms of area and throughput/area, again due to its simple structure. In particular, the parallelized version of 64x produced the best throughput/area ratio. The results of the implementation are also sorted by minimum area and maximum throughput/area ratio in Table 4. In both of these criteria, Grain and Trivium performed well. As stated previously, the F-FCSR family performed well in the throughput/area category and the DECIM family performed well in the minimum area category. The results are summarized graphically in Fig. 3.

Table 3. Results of Xilinx Spartan 3 FPGA implementations of all eSTREAM candidates

Candidate	Data rate (bits/ cycle)	Maximum Clock Frequency (MHz)	Maximum Throughput (Mbps)	Area (slices)	Throughput/ Area (Mbps/ slice)	Device
DECIM v2	0.25	185	46.25	80	0.58	xc3s50-5pq208
DECIM 128	0.25	174	43.5	89	0.49	xc3s50-5pq208
Edon80	1	130	130	1284	0.10	xc3s200-5pq208
F-FCSR-H v2	8	138	1104	342	3.23	xc3s50-5pq208
F-FCSR-16	16	134	2144	473	4.53	xc3s50-5pq208
Grain v1	1	196	196	44	4.45	xc3s50-5pq208
Grain v1 (x16)	16	130	2080	348	5.98	xc3s50-5pq208
Grain 128	1	196	196	50	3.92	xc3s50-5pq208
Grain 128 (x32)	32	133	4256	534	7.97	xc3s50-5pq208
MICKEY 2.0	1	233	233	115	2.03	xc3s50-5pq208
MICKEY 128 2.0	1	223	223	176	1.27	xc3s50-5pq208
Moustique	1	225	225	278	0.81	xc3s50-5pq208
Pomaranch	1	49	49	648	0.08	xc3s50-5pq208
Trivium	1	240	240	50	4.80	xc3s50-5pq208
Trivium (x64)	64	211	13504	344	39.26	xc3s400-5fg320

Table 4. Spartan 3 implementation results sorted by minimum area and maximum throughput/area

Candidate	Area (slices)	Candidate	Throughput/Area (Mbps/slice)
Grain v1	44	Trivium (x64)	39.26
Grain 128	50	Grain 128 (x32)	7.97
Trivium	50	Grain v1 (x16)	5.98
DECIM v2	80	Trivium	4.80
DECIM 128	89	F-FCSR-16	4.53
MICKEY 2.0	115	Grain v1	4.45
MICKEY 128 2.0	176	Grain 128	3.92
Moustique	278	F-FCSR-H v2	3.23
F-FCSR-H v2	342	MICKEY 2.0	2.03
Trivium (x64)	344	MICKEY 128 2.0	1.27
Grain v1 (x16)	348	Moustique	0.81
F-FCSR-16	473	DECIM v2	0.58
Grain 128 (x32)	534	DECIM 128	0.49
Pomaranch	648	Edon80	0.10
Edon80	1284	Pomaranch	0.08

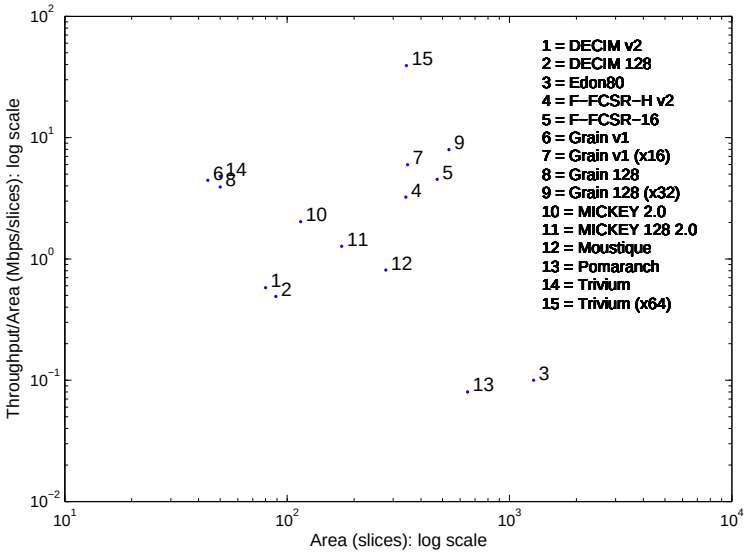

Fig. 3. Area and throughput/Area results in graphical form

5 Comparison to Previous Work

The papers [3], [7], [8], [9], and [10] all show Xilinx FPGA implementations of various eSTREAM candidates. In addition, [11] gives results for eSTREAM can-

didates on Altera devices. There has also been work on ASIC implementations of eSTREAM candidates, such as those found in [12], [7], [13], and [14]. As seen in Table 5, our results compare well with the best previously published FPGA results in terms of throughput/area, particularly for MICKEY, Grain, and Trivium. For some ciphers, as far as we know, there are no previously published FPGA implementations (though ASIC implementations may exist) to compare with; these include DECIM v2, DECIM 128, F-FCSR-H, F-and FCSR-16 (a Pomaranch FPGA implementation has been published in [15], but we were not able to obtain the paper.). Some notes on the other ciphers: the Edon80 publication [3] shows an iterative architecture; our architecture is not iterative but the two architectures can be compared using throughput/area. The authors of Moustique [16] produced an FPGA implementation faster than ours but of similar area. Paper [9] implements Mosquito, a Moustique predecessor. In general our work agrees with [8] and [7] that Trivium and Grain seem best suited for Xilinx FPGA implementation.

Table 5. Previously published Xilinx FPGA implementations of eSTREAM ciphers.

Candidate	Maximum Clock Frequency (MHz)	Maximum Throughput (Mbps)	Area (slices)	Throughput/ Area (Mbps / slice)	Device
Edon80 [3]	149	1.87	50	0.04	Spartan-3
Grain v1 [9]	-	105	48	2.19	Spartan-II
Grain v1 [7]	193	193	122	1.58	Spartan-3
Grain v1 (x16) [7]	155	2480	356	6.97	Spartan-3
Grain 128 [8]	181	181	48	3.77	Virtex-II
MICKEY 128 2.0 [10]	170	170	167	1.02	Virtex
MICKEY 128 2.0 [8]	200	200	190	1.05	Virtex-II
MICKEY 128 2.0 [7]	156	156	261	0.60	Spartan-3
Moustique [16]	-	369	252	1.46	Virtex-II
Mosquito [9]	-	137	298	0.46	Spartan-II
Trivium [8]	207	207	41	5.05	Virtex-II
Trivium [9]	-	102	40	2.55	Spartan-II
Trivium [7]	201	201	188	1.07	Spartan-3
Trivium (x64) [7]	190	12160	388	31.34	Spartan-3

6 Conclusion

This paper described hardware implementations of all eSTREAM profile 2 phase 3 candidates targeted for Xilinx FPGAs. From the results, and assuming the mathematical security of all ciphers to be equivalent, the authors deduce the Grain and Trivium family of ciphers are most efficient in terms of the metrics

of minimum area and maximum throughput per area on Xilinx FPGA architectures. These ciphers also have the added benefit of high-throughput modes, in which lookahead structures can greatly increase the data radix, and thus the throughput. The Mickey family of ciphers also provided a good balance of low FPGA area and good throughput / area ratio.

Acknowledgements. The authors wish to thank the students of the Fall 2007 ECE 545 class at George Mason University. Particular thanks to Brent Roeder (Edon80), Arvind Bhat (F-FCSR), Bin Zhou (Moustique), and Justin Thorpe (Pomaranch).

References

1. eSTREAM, ECRYPT Stream Cipher Project http://www.ecrypt.eu.org/stream.
2. Xilinx Spartan-3 FPGA Family: Complete Data Sheet http://www.xilinx.com.
3. Kasper, M., Kumar, S., Lemke-Rust, K., Paar, C.: A compact implementation of Edon80. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/057 (2006) http://www.ecrypt.eu.org/stream.
4. Hell, M., Johansson, T., Meier, W.: Grain - a stream cipher for constrained environments. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/010 (2005) http://www.ecrypt.eu.org/stream.
5. Jansen, C., Kholosha, A., Helleseth, T.: A lightweight implementation of the Pomaranch S-Box. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/042 (2007) http://www.ecrypt.eu.org/stream.
6. De Canniere, C., Preneel, B.: Trivium specifications. eSTREAM, ECRYPT Stream Cipher Project http://www.ecrypt.eu.org/stream.
7. Gaj, K., Southern, G., Bachimanchi, R.: Comparison of hardware performance of selected phase II eSTREAM candidates. State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report 2007/026 (2007) http://www.ecrypt.eu.org/stream.
8. Bulens, P., Kalach, K., Standaert, F.X., Quisquater, J.J.: FPGA implementations of eSTREAM phase-2 focus candidates with hardware profile. State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report 2007/024 (2007) http://www.ecrypt.eu.org/stream.
9. Good, T., Chelton, W., Benaissa, M.: Review of stream cipher candidates from a low resource hardware perspective. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/016 (2006) http://www.ecrypt.eu.org/stream.
10. Kitsos, P.: On the hardware implementation of the MICKEY-128 stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/059 (2006) http://www.ecrypt.eu.org/stream.
11. Rogawski, M.: Hardware evaluation of eSTREAM candidates: Grain, Lex, Mickey128, Salsa20 and Trivium. State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report 2007/025 (2007) http://www.ecrypt.eu.org/stream.
12. Feldhofer, M.: Comparison of low-power implementations of Trivium and Grain. State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report 2007/027 (2007) http://www.ecrypt.eu.org/stream.

13. Good, T., Benaissa, M.: Hardware results for selected stream cipher candidates. State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report 2007/023 (2007) http://www.ecrypt.eu.org/stream.
14. Gurkaynak, F., Luethi, P., Bernold, N., Blattmann, R., Goode, V., Marghitola, M., Kaeslin, H., Felber, N., Fichtner, W.: Hardware evaluation of eSTREAM candidates: Achterbahn, Grain, MICKEY, MOSQUITO, SFINKS, Trivium, VEST, ZK-Crypt. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/015, (2006) http://www.ecrypt.eu.org/stream.
15. Kitsos, P., Koufopavlou, O.: An FPGA-based implementation of the Pomaranch stream cipher. In: 3rd Int. Mobile Multimedia Communications Conference (MSAN) - MobiMedia 2007. (Aug 2007)
16. Daemen, J., Kitsos, P.: The self-synchronizing stream cipher MOUSTIQUE. eSTREAM, ECRYPT Stream Cipher Project http://www.ecrypt.eu.org/stream.