

Comparison of Low-Power Implementations of Trivium and Grain^{*}

Martin Feldhofer

Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A-8010 Graz, Austria
Martin.Feldhofer@iaik.tugraz.at

Abstract. This paper provides a comparison of the two stream cipher proposals Grain and Trivium which are candidates in the hardware focus phase of the eSTREAM project. We evaluate these algorithms concerning their feasibility to implement them for low-power applications in RFID systems. A triple of parameters which includes the chip area, the power consumption, and the number of clock cycles for encrypting a fixed amount of data is introduced which allows a fair comparison of the proposals. The datapaths of the implementations are presented in detail and the synthesis results are shown. A comparison of the results of Grain and Trivium with an AES implementation shows that the chip area of Trivium is slightly smaller while Grain requires less clock cycles for encrypting 128 bits of data. The low-power implementations of the stream ciphers require only a fourth of the mean current consumption of the AES algorithm.

1 Introduction

The project eSTREAM within the European Network of Excellence ECRYPT has the aim to find out a small number of stream ciphers that are of interest for the crypto community and standardization bodies. In the current evaluation phase the 34 submissions are evaluated concerning several criteria. The most important objective is of course the security of the cipher. This means that ciphers with identified security issues are not considered for further evaluation. Other selection criteria are the simplicity and flexibility of the algorithms as well as the completeness and clarity of the description. At the end of the first evaluation phase the algorithms were classified according to their suitability for software or hardware implementation. In this paper, we are interested in the evaluation of two algorithms which are in the so-called “HW Focus Phase 2”. This means that they are hardware-related stream ciphers and of particular interest

^{*} The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT and the initiative PROACT (Programme for Advanced Contactless Technology) at Graz University of Technology.

for eSTREAM. In more detail, we evaluate the two stream cipher candidates Grain [5] and Trivium [1] concerning their efficiency in regard to hardware implementation. Our main goal is to implement these two algorithms for low-power applications in RFID systems and compare the results with the AES implementation optimized for RFID applications [4].

Radio Frequency Identification (RFID) is an emerging technology which allows giving a digital identity to nearly every object in the world. An RFID system mainly consists of a reader device which communicates with a passive RFID tag. This tag is a small microchip attached to an antenna. The tag is passively powered by the reader. Many applications like supply-chain management, access control, and inventory management already use RFID technology. The main challenge for further fields of application is the acceptance of RFID technology by consumers. In many published articles, the missing privacy protection is an argument against the use of RFID tags. Therefore, we think that integrating security into RFID systems is inevitable. There are many proposed solutions to overcome the problems of privacy protection and forgery of tags. In order to obtain a high level of security it is necessary to use strong cryptographic algorithms and standardized protocols because proprietary solutions often have security vulnerabilities. Commonly, the assumption is made that it is too expensive to include public-key cryptography on RFID tags. As presented in the survey paper of Ari Juels [6] most researchers propose the use of hash functions as cryptographic primitives for security protocols. Only a few proposals have been published which employ block ciphers like the Advanced Encryption Standard (AES) [2]. An efficient stream cipher that is known to be secure with a small chip area and that has low power consumption could be a good alternative.

This article tries to evaluate the hardware implementations of Grain, Trivium, and the AES by using the same metric. A fair comparison with the literature is difficult because all published works use different process technologies, have other design goals, and their results are presented differently. Therefore, we propose a suitable metric for hardware implementations on RFID tags to make a comparison in terms of power consumption, chip area, and clock cycles per encryption possible. We optimized the implementations of the algorithms for application in passively-powered RFID tags, and evaluated the circuits according to the introduced metric. We show the synthesis results of the stream ciphers and compare the power simulation result of Synopsys NanoSim with the values of the AES implementation.

This article is structured as follows. Section 2 defines the requirements for hardware implementations of a crypto module for RFID tags and describes the design methodology for low power consumption. Section 3 and Section 4 describe the algorithms Grain and Trivium and show their low-power implementation. A comparison of the results with those of AES is given in Section 5 and conclusions are drawn in Section 6.

2 Design for Low-Power Consumption

Implementing a cryptographic unit for an RFID tag is challenging due to the stringent requirements concerning the three parameters power consumption, chip area, and number of clock cycles per encrypting one block of data. The available power budget is very small because the dissipated power directly influences the operation range of the tag. The upper bound of the mean current consumption is $15 \mu A$. Exceeding this power budget reduces the operation range. The limited chip area has more an economical reason than a technical one. This is due to the fact that the silicon area of the chip determines the costs of the tag. The available chip area for security features can be assumed to lie between 1,000 and 10,000 gate equivalents. The third parameter that determines the suitability of cryptographic algorithms is the number of clock cycles needed for computation. In general, the data rates of RFID systems are very low and hence increased computation time can be traded for lower power consumption. But the calculating time of cryptographic operations also influences the response time of the tag. For instance, HF tags conforming to the ISO 15693 standard have to respond within only $300 \mu s$. Here, a solution using an interleaved protocol can be used to avoid this issue [2]. For stream ciphers the initialization phase could have a significant influence on the protocol because a frequent key update could lead to a long latency.

A comparison between different implementations of crypto algorithms is only possible when an optimization goal is stated. Many publications use the area-delay product or the power-delay product to compare their results with each other. The multi-dimensional optimization problem to vary the design parameters chip area, power consumption, energy consumption, number of clock cycles, clock speed, and so on has different objectives depending on the field of application. Two parameters are often orthogonal while others are correlated. In battery-powered devices for example, the energy consumption has to be minimized while other parameters like chip area and throughput are not crucial. In RFID systems, we have two major design objectives which do not correlate. These are the reduction of power consumption and chip area. Important as well is the number of clock cycles for executing the algorithm. Of minor relevance are the energy consumption and the maximum clock frequency.

$$(I_{mean} [\mu A], Area [GE], Clock\ cycles [\#]) \quad (1)$$

In Equation 1 we define a triple of parameters for optimizing hardware modules in RFID systems. It allows comparing different implementations. This triple does not map all cost factors to a single one-dimensional value (a real metric). Instead, the individual cost factors remain unweighted. This allows calculating compound metrics like the area-delay product by others. The first parameter of Equation 1 is the mean current consumption which is stated in μA . The average current depends approximately linearly on the clock frequency and on the supply voltage. In our case, 100 kHz and 1.5 V are nominal operation conditions. The second parameter is the chip area of the implementation given in gate equivalents. The

third parameter is the number of clock cycles to execute the algorithm. The cycle count and the circuit size measured in gate equivalents are nearly independent of the process technology. On the contrary, the power values given in this article are only valid for a $0.35 \mu\text{m}$ CMOS process technology. They cannot be mapped to a different technology easily.

The differences between power consumption and energy consumption were figured out in [3]. For battery-powered devices the energy consumption per operation is the optimization goal. This means the power-delay product should be minimized. On the contrary, for passively-powered devices like RFID tags the mean power consumption is critical. The duration of the operation itself does not matter that much. This also means that the power consumption per clock cycle should be optimized while the energy consumption of an operation is not of concern. Here it is often necessary to serialize operations because the concurrent calculation would exceed the available power. The implementations of the presented algorithms try to minimize the mean power consumption.

The total power consumption of a CMOS circuit is the sum of static and dynamic power consumption. The static power consumption caused by the leakage current, mainly depends on the size of the chip. It is very small and can be more or less ignored for our setting. Equation 2 presents the influences on dynamic power consumption. The design measures for lowering the power consumption result from minimizing the factors in this equation.

$$P_{dyn} = C_L \cdot V_{DD}^2 \cdot f_{CLK_{eff}} \cdot p_{sw} \quad (2)$$

The load capacitance C_L of the chip increases as more gates are placed on the die. This means that lowering the die size as well as reducing the supply voltage (V_{DD}) to a minimum directly reduces power consumption. These two factors tend to be predetermined by the low die-size constraint and the operating conditions of the chip. The switching activity p_{sw} of the circuit can be reduced by using a method called sleep logic. Thereby, unnecessary switching activity is eliminated by using AND gates at the input of combinational circuits. Assuming a fixed supply voltage and a minimum of switching activity in the combinatorial paths, the best option for a low-power design is reducing the effective clock frequency $f_{CLK_{eff}}$ of the circuit. This reduces the power consumption linearly. Clock gating is a good measure for reducing the effective clock frequency. In our architectures, all datapath registers and nearly all control logic registers are clocked only when there is a potential signal change. This turned out to lower the power consumption significantly. Thereby, parts of the circuit are virtually switched off when they are not in use.

Figure 1 shows a typical clock gating register which consists of a data latch and b flip-flops. The D-latch is used to prevent glitches in the clock signal of the flip-flops which build a register of b bits. The optimal bit-width of the registers depends on the implemented algorithm. Minimizing the number of memory elements (flip-flops and latches have nearly the same power consumption at the active clock edges) that are active at the same time, leads to a reduction of the total power consumption. Therefore we separate all N flip-flops of the design

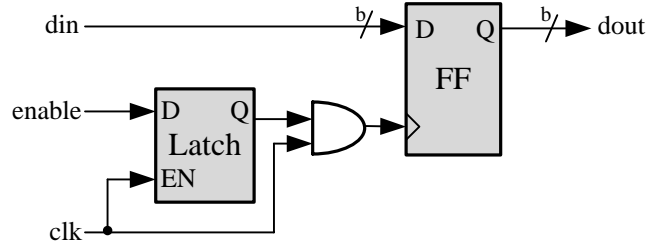


Fig. 1. Clock gating register.

into registers with b bits. The number of active elements n depends on the total number of memory elements N required by the algorithm and the bit-width b of the registers and can be calculated using Equation 3. The first value $\frac{N}{b}$ is the number of clock-gating D-latches which are active in every clock cycle (this is also the number of b -bit registers in the design) and b denotes the number of flip-flops in each register. When minimizing this equation it follows that the optimal bit-width of a design is $b = \sqrt{N}$ (see derivation in Equations 4-6).

$$n(b) = \frac{N}{b} + b \quad (3)$$

$$\frac{dn}{db} = -\frac{N}{b^2} + 1 \quad (4)$$

$$-\frac{N}{b^2} + 1 = 0 \quad (5)$$

$$b = \sqrt{N} \quad (6)$$

For the algorithms Grain and Trivium the optimal number of flip-flops per register is 12.6 and 17.0, respectively. Therefore we decided to implement both algorithms using a 16-bit word size which simplifies the interface of this crypto module.

3 Grain Stream Cipher

The stream cipher Grain was designed by Martin Hell, Thomas Johansson, and Willi Meier [5]. Their main goal was to design an algorithm which is very easy to implement in hardware and requires only small chip area. It is a bit-oriented synchronous stream cipher which means that the key stream is generated independently from the plaintext. In general, a stream cipher consists of two phases. The first phase is the initialization of the internal state using the secret key and the IV. After that, the state is repeatedly updated and hence used to generate key-stream bits. The main elements of the stream cipher Grain are two 80-bit shift registers where one has a linear feedback (LFSR) and the other a non-linear feedback (NFSR). The key size is specified with 80 bits and additionally

an initial value of 64 bits is required. Unfortunately, the initial version of Grain (Version 0) had a weakness in the output function which was discovered during the first evaluation phase. This paper uses Grain (Version 1) for implementation which solves the security issues of the initial version.

The basic structure of the algorithm can be seen in Figure 2. Two polynomials of degree 80, $f(x)$ and $g(x)$, are used as feedback function for the feedback shift registers LFSR and NFSR. The output function $h(x)$ uses as input selected bits from both feedback shift registers. Additionally, seven bits of NFSR are XORed together and the result is added to the function $h(x)$. This output is used during the initialization phase as additional feedback to LFSR and NFSR (grey lines in the figure indicate this feedback). During normal operation this value is used as key stream output.

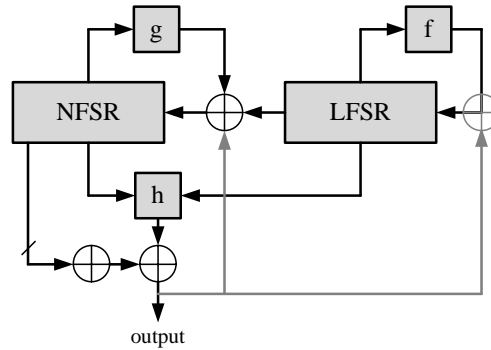


Fig. 2. Grain stream cipher.

3.1 Implementation of Grain

The hardware module of Grain was implemented with a 16-bit AMBA APB interface in a $0.35\ \mu\text{m}$ CMOS process technology. This interface fits to the 16-bit datapath architecture. The reason for implementing a 16-bit word size was the low-power design approach as presented in Section 2. The details of the datapath are shown in Figure 3. It can be seen that the feedback shift registers NFSR and LFSR shift 16 bits per clock cycle. Only a single register is clocked at the same point in time via clock gating which eases the input of the key and the initial value because the same 16 input wires are connected to all registers. Additionally, it reduces the mean power consumption significantly. This comes at the expense of having a temporary register which stores intermediate results. Additionally, all combinational circuits like the feedback functions $g_function$, $f_function$, and $h_function$ have to be implemented in radix 16. The inputs of these functions are selected bits from the registers and are not shown in detail in this figure. The $h_function$ in the datapath description also includes the XOR

operations of the output function in the algorithm description. The output of the module is registered and instead of the key stream the encrypted result of the data input is stored in the register. Instead of a multiplexer that selects the correct feedback function for the temporary register, AND gates are used to enable and disable the appropriate inputs. Producing a 16-bit encryption result after initialization requires 13 clock cycles.

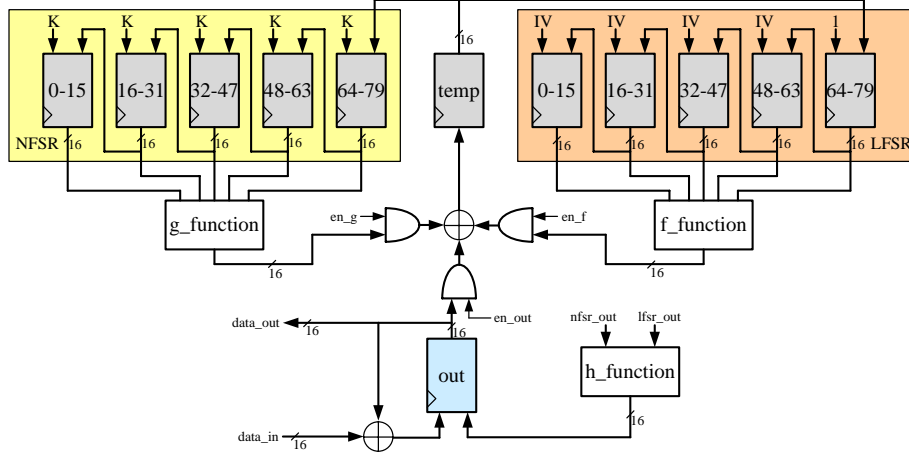


Fig. 3. Datapath of Grain.

3.2 Synthesis Results

In comparison to the straight-forward, minimum-area implementation in radix-1 which has an 8-bit interface and requires 1,760 gate equivalents, the area requirements for the our low-power Grain implementation are 3,360 GEs. This doubling of the size is due to the large combinational logic circuit for the radix-16 datapath. A detailed comparison of the components in these types of implementation can be seen in Table 3.2.

4 Trivium Stream Cipher

The developers of the stream cipher Trivium are Christophe De Cannière and Bart Preneel [1]. This hardware-oriented synchronous stream cipher was designed to explore how simply a stream cipher could be designed without sacrificing its security. It is possible to generate up to 2^{64} bits of key stream from an 80-bit key and an initial value (IV) of 80 bits. The state consists of 288 bits which are denoted as s_0, s_2, \dots, s_{287} . The pseudo code in Figure 4 shows how the algorithm uses 15 specific bits of the state to generate three variables which are

Table 1. Components of Grain datapath.

Component	Min. area (radix-1)	Low-power (radix-16)
NFSR + LFSR registers (160 bits)	1,275 GE	1,130 GE
Temporary register	0 GE	85 GE
Output register	50 GE	150 GE
Combinational logic and misc.	315 GE	1,835 GE
Controller (FSM)	120 GE	160 GE
Total	1,760 GE	3,360 GE

used to update the state and which produce one bit of the output. During the initialization, which is not shown in the figure, the key and the IV are loaded to the state and the same update function is applied 1,152 times without using the output for the key stream. In the algorithm description, N is used for the number of output bits of the stream cipher.

```

for i = 0 to N-1 do
  t0 = s65 + s92
  t1 = s161 + s176
  t2 = s242 + s287
  outi = t0 + t1 + t2
  t0 = t0 + s90·s91 + s170
  t1 = t1 + s174·s175 + s263
  t2 = t2 + s285·s286 + s68
  (s0,s1,...,s92) = (t2,s0,...,s91)
  (s93,s94,...,s176) = (t0,s93,...,s175)
  (s177,s178,...,s287) = (t1,s177,...,s286)
end for

```

Fig. 4. Pseudo code of Trivium stream cipher.

4.1 Implementation of Trivium

The implementation of the Trivium module has the same 16-bit AMBA APB interface as the one for Grain. Implementing a radix-16 datapath is also motivated by the low-power design technique. Figure 5 shows the details of the architecture. The boxes denoted with *comb* are the combinational logic elements of the algorithm that are used for updating the state according to the algorithm specification. The 288 flip-flops for the state are separated in 16-bit registers. Additionally, two temporary registers are necessary which store intermediate results. The output register is again used for directly applying the XOR operation

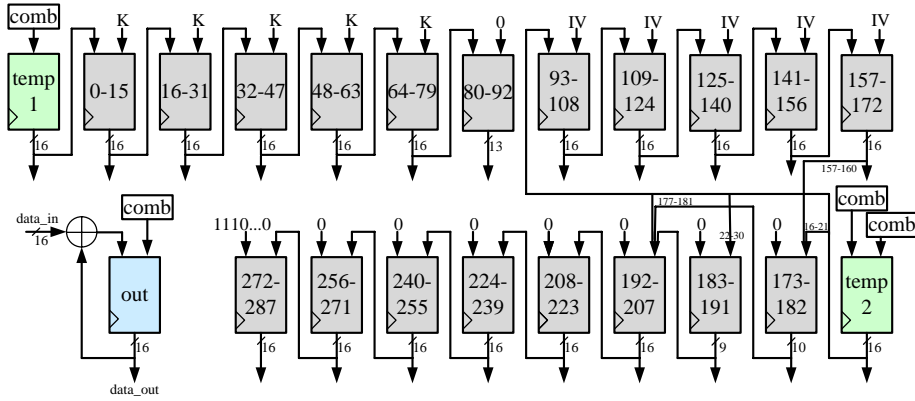


Fig. 5. Datapath of Trivium.

Table 2. Components of Trivium datapath.

Component	Min. area (radix-1)	Low-power (radix-16)
State registers (288 bits)	1840 GE	2040 GE
Temporary registers	0 GE	200 GE
Output register	50 GE	150 GE
Combinational and misc.	290 GE	410 GE
Controller (FSM)	210 GE	290 GE
Total	2,390GE	3,090 GE

of the key stream with the input value. Again, clock gating is used to only clock one register per clock cycle. During initialization, the key, the IV, and the constants are loaded into the registers. Then the combinational circuit is used to update the registers in a kind of pipeline where the temporary registers are used to prevent overwriting of needed values. The generation of a 16-bit key stream after the initialization phase requires 22 clock cycles.

4.2 Synthesis Results

A comparison of the synthesis results of the minimum-area and the low-power implementation can be seen in Table 4.2. The biggest component in Trivium is the state register which consists of 288 flip-flops. The difference between the two variants is not as big as for the Grain algorithm because the combinational circuit for radix-1 and radix-16 implementations do not differ that much. The low-power variant requires only slightly more area for the state registers, the additional temporary registers, the output register, and the combinational logic.

5 Comparison of the Results

Table 3 shows the results of the implementation and gives a comparison of the stream ciphers Grain and Trivium with an AES implementation for RFID tags [4]. The chip area results are based on synthesis and are given in gate equivalents (GE). For the used $0.35\ \mu\text{m}$ CMOS process technology one gate equivalent compares to a NAND2 cell of $55\ \mu\text{m}^2$. The mean current consumption in the third column is given in μA at a nominal clock frequency of 100 kHz and a supply voltage of 1.5 V. The current values were obtained by power simulations with NanoSim. It can be seen that beside the key size which is of minor relevance in this paper the algorithms Grain and Trivium require less chip area. The clock cycles given in the last column are for encryption of 128 bits of data while for the stream ciphers the numbers in parenthesis are required for the initialization. The power consumption values in the third column show that Trivium and Grain only require a fourth of the power than the AES. These excellent power consumption values come at the expense of a slight increase of area requirement in comparison to a “straight-forward” implementation of the stream ciphers.

6 Conclusions

In this article we presented an evaluation of the two stream cipher proposals Grain and Trivium. We compared the results of their low-power implementations with the AES implementation which was optimized for passively-powered devices like RFID tags. The results show that the stream cipher Trivium requires 3,090 gate equivalents which is the least amount of chip area of the analyzed algorithms. This is surprising because the state size of Trivium (288 bits) is larger than that of AES-128 (256 bits) and Grain (160 bits). The longer initialization phase of Trivium which is more than 1,600 clock cycles could be an issue depending on the implemented protocol. If a frequent key change is necessary this overhead could be significant. The power consumption values below $1\ \mu\text{A}$ at 100 kHz and 1.5 V show the optimization of the stream cipher implementations for passively-powered applications like RFID tags. This comes at the expense of a slight increase of area requirements in comparison to a “straight-forward” implementation.

Table 3. Synthesis results on $0.35\ \mu\text{m}$ CMOS.

Algorithm	Security [bits]	I_{mean} [$\mu\text{A}@100\text{kHz}$]	Chip area [GE]	Clock [cycles]
Grain	80	0.80	3,360	(130)+104
Trivium	80	0.68	3,090	(1,603)+176
AES-128	128	3.0	3,400	1,032

References

1. C. D. Cannière and B. Preneel. TRIVIUM Specifications. eSTREAM, ECRYPT Stream Cipher Project (<http://www.ecrypt.eu.org/stream>), Report 2005/030, April 2005.
2. M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong Authentication for RFID Systems using the AES Algorithm. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer, August 2004.
3. M. Feldhofer and J. Wolkerstorfer. Low-power Design Methodologies for an AES Implementation in RFID Systems. In *Workshop on Cryptographic Advances in Secure Hardware 2005 (CRASH05), September 6-7, Leuven, Belgium*, September 2005.
4. M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES Implementation on a Grain of Sand. *IEE Proceedings on Information Security*, 152(1):13–20, October 2005.
5. M. Hell, T. Johansson, and W. Meier. Grain - A Stream Cipher for Constrained Environments. eSTREAM, ECRYPT Stream Cipher Project (<http://www.ecrypt.eu.org/stream>), Report 2005/010, 2006. Revised version.
6. A. Juels. RFID Security and Privacy: A Research Survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, February 2006.

The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.