Recommendations for Hardware Evaluation of Cryptographic Algorithms

Frank K. Gürkaynak, Peter Luethi

Integrated Systems Laboratory, ETH Zurich CH-8092 Switzerland

Abstract. At the SASC 2006 three papers on hardware implementation of the eSTREAM candidates were presented. The workshop provided an interesting platform where hardware designers were confronted with the developers of the algorithms. The presentations were followed by a lively discussion. As hardware designers, we must admit that we have learned a lot from these discussions. In this brief document we want to outline our personal observations and will attempt to make some suggestions for further hardware evaluations.

1 Introduction

The European Network of Excellence for Cryptography (ECRYPT) has started a multi-year effort called eSTREAM to identify new stream ciphers that might become suitable for widespread adoption. A total of 34 algorithms have been submitted to eSTREAM. At the *Integrated Systems Laboratory* (IIS) we were looking for semester projects for our 7th semester students, when we came across eSTREAM. Since a number of cryptographic algorithms are developed with hardware realizations in mind, they are very well suited for such semester theses.

For the winter semester 2005/2006, four students showed interest in a project targeting the implementation of cryptographic hardware. It was decided to design a subset of eSTREAM candidates and thereby to provide a fair comparison (of at least the implemented set) of candidate algorithms. Since the entire IC design had to be completed within one semester (14 weeks), not all 34 candidate algorithms could be realized with reasonable effort. At the start of the project in October 2005, the decision for a subset of stream cipher algorithms had to be made, and eventually eight seven eSTREAM candidate algorithms were sorted out: Achterbahn, Grain, MICKEY, MOSQUITO, SFINKS, Trivium, VEST, and ZK-Crypt.

To provide a comparative reference for the results, the well-known *Advanced Encryption Standard* (AES) block-cipher was implemented in Output-Feedback (OFB) mode. In this configuration mode, the block-cipher is able to generate a continuous output stream that can be used as a stream-cipher. Since we have significant experience in implementing the AES algorithm at the IIS, we were able to efficiently customize an AES block for stream-cipher implementation.

All algorithms were implemented using a 0.25 µm CMOS technology and two ASICs containing more than ten different algorithm variations were taped out. We have presented our findings at the SASC 2006 conference at Leuven. At the workshop, we have received many comments both positive and negative on our work, and we believe we have learned a lot from these comments. In this paper we would like to share our observations on the hardware evaluation of cryptographic algorithms and make some suggestions for others who might be interested in similar projects. The following comments will be based on the implementation of eSTREAM candidates, but are general enough for other evaluations as well.

2 Problems

There are several differences in how a hardware and a cryptographic algorithm designer view a system. For a hardware designer:

> Efficient hardware design is essentially a resource allocation problem. The goal is, given the constraints, to find the optimal balance between required silicon area, operation throughput, energy consumption and design time to implement a system.

Obviously the algorithm designer is mainly interested in the cryptographic qualities of the system. Once these requirements are fulfilled (the algorithm provides adequate security, supports necessary modes etc), the algorithm designer requires a performance metric that can be used to compare different algorithms.

The difference is subtle: the hardware designer does not fully comprehend the cryptographic qualities, and the algorithm designer is not able set the constraints for the system. In our view this is the main reason for most of the problems encountered during the evaluation of the eSTREAM candidates.

The following is a list of separate problems

2.1 Selection of Algorithms

There were more than 30 candidate algorithms for eSTREAM. Unlike software evaluations, where reference code is provided by the developers, in a hardware evaluation, the code for each algorithm needs to be developed and optimized. This is a time consuming task which will exceed the resources of most research groups. It is therefore necessary to make a selection. Problem is, hardware designers are not always fully qualified to make such a selection on their own[1]. As a result, the selection made by the hardware designers may not be optimal. Some 'interesting' algorithms may be overlooked in favor of comparatively 'problematic' algorithms.

2.2 Cryptographic Properties

Algorithms in a comparative study may include different cryptographic properties. One algorithm may be considered to be more secure on account of supporting a longer secret key length, others may support operation modes that will make them suitable for a wider range of applications. For a hardware designer these are difficult to judge. As an example for eSTREAM. The call is for at least 80-bit security. If one algorithm claims to have 128-bit security, is it fair to compare the two algorithms? One could argue about this question, but the hardware designer will not be as qualified as a cryptographer to provide the answer.

Especially new algorithms tend to have a wide range of configuration options. Designing hardware that supports all possible options may be a tedious task. Note that, all supported options also need to be properly verified. The tendency of a hardware designer is to support the "main options". If several algorithms are to be compared, usually the decision will be to support only the intersection of all possible options. Once again, if these decisions are left to pure hardware designers, the results may not always satisfy the cryptographers. Once again let me use eSTREAM as an example. Several, but not all algorithms support a MAC mode. Worse yet, several algorithms require additional hardware for this support. From a hardware designers point of view, implementing all algorithms without the MAC mode is 'fair', however cryptographers may have a different view.

2.3 Applications

Cryptographic algorithms can be used for a variety of applications. Problem is these applications may have drastically different requirements. As mentioned earlier, hardware designers are accustomed to implement designs according to the constraints. As such, a cryptographic core designed for RFID applications and one designed for backbone routers will have dramatically different constraints which will lead to substantially different designs.

Again using the eSTREAM candidates as an example. How should the initialization problem be addressed? If the algorithm will be used in a system where it is frequently initialized, the hardware designer will invest some of its resources (design time/circuit area/operation speed/energy consumption) for a more efficient initialization. If however, it is assumed that the initialization is not critical, little or no extra effort will be invested in this area. Once again, this is the result of a hardware designer making assumptions which may be opposed by cryptographers.

[1] It is a crude generalization to suggest that all experienced hardware designers have no credible knowledge on cryptography. However, in our opinion only a small fraction of hardware designers are sufficiently knowledgeable in cryptography.

While cryptographers are mostly interested in the extremes in terms of performance, hardware designers need to concentrate their efforts to meet the requirements. If the system demands a throughput for 100 Mbits/s, a system that has a throughput of 10 Gbits/s is not necessarily better. Especially if this additional throughput comes at the expense of other resources (circuit area/energy consumption/design time)[2]. Similarly in a system where the cryptographic core occupies 1% of the circuit area, reducing the area by 20% will not be a real advantage.

2.4 FPGA/ASIC

In principle FPGA and ASIC designs are similar. They use the same hardware description languages (VHDL/Verilog), use a similar design flow (synthesize, place and route), and the same architectural transformations can be used for both systems. However, whether or not a design is better suited for FPGA or an ASIC depends entirely on the constraints.

Since FPGA design flows are more affordable, a significant portion of the hardware evaluations reported in the literature are based on FPGAs. Recently FPGA devices have started including an ever increasing array of resources including block RAMS, multipliers, adders, and in some cases even complete microprocessors. Implementations that utilize these built-in resources have much better performance. This sometimes requires 'device specific' instantiations in the hardware description. Such customized hardware descriptions are generally not easily portable, as the capabilities of the embedded resources differ even between FPGA families of the same manufacturer. Such embedded resources have also made it more difficult to compare ASIC performance against FPGA performance. With earlier FPGAs it was somewhat possible to translate the logic complexity into ASIC gate equivalents, as they only contained generic logic blocks.

FPGAs with embedded resources allow rather coarse-grained optimizations. If you can utilize an embedded resource you will have better performance. If this is not possible, the same functionality will have to be built from generic logic blocks, with noticable performance penalties. This is in stark contrast to ASIC design flow, which allows very fine grained optimizations.

From a hardware designers perspective, the constraints of the project (cost/performance/power) will determine if a system is better suited to an FPGA or an ASIC implementation[3]. They are therefore not necessarily compatible. As an example an RFID application has extreme constraints for power consumption. RFID tags must also be cheap to manufacture. This rules out an FPGA implementation. Therefore, FPGA based comparisons are not really relevant for this application (FPGA power consumption will be orders of magnitude above that of a custom ASIC)

2.5 I/O Requirements

Although most cryptographic algorithms are presented as stand-alone cores, in practice most will be implemented as part of a system. The data I/O requirements of the crypto core can significantly alter the performance of the system. As an example let us take Trivium-64 of eSTREAM candidates. The core area of this algorithm is 150.000 μm^2 in the 0.25 μm CMOS process. The system operates at a clock frequency of 400 MHz, where it has a throughput exceeding 20 Gbits/s. To achieve this throughput data needs to be moved to and from the Trivium core at the same rate. These data rates are only feasible for on-chip communications. Off-chip communications at this rate would require very high power consumption (around 1-Watt for modest off-chip loads) and require excessive area for the pad drivers (around 2.5 mm^2 with optimistic assumptions). In fact, generating the keystream would be the least challenging part of such a system. Even if all data is generated and consumed on-chip, it is a significant challenge to transfer data at this rate.

2.6 Reference Design

Comparison to a well-known design is important for evaluations. However, at times results from literature are substituted for this purpose. This can be deceiving. First of all, mapping performance data over different implementation

[2] Obviously, if it is free, hardware designers will gladly take it. But no hardware designer is going to invest an additional month into making a 100 Mbits/s design work at 200 Mbits/s, if it is not required.

[3] There could be some constraints that would equally favor an ASIC and an FPGA implementation, but such designs are relatively few

styles/technologies is an inexact science. Secondly, not all implementation details of the reference may be known. The reference in the literature may not support all required functionality, the performance numbers may be valid under certain assumptions etc.

2.7 Implementation Errors

The hardware designer has a responsibility to deliver functionally correct hardware. This is accomplished by several verification steps, where the simulated hardware is compared against a reference implementation. However, it is indeed possible that the final design, although functionally correct, includes flawed design decisions that result in hardware that is larger and/or slower than an optimal implementation. Synthesis tools may be controlled by setting constraints, but it is virtually impossible to claim that a synthesis result is the most optimal implementation of the hardware[4]. Compilation parameters, VHDL coding style, and the specific compiler may have a significant effect on the results.

Such implementation errors are especially annoying as when one algorithm is found to contain such errors[5], the credibility of all remaining work will suffer as well.

2.8 Side Channel Attack Properties

The way a cryptographic algorithm is implemented, may have a significant impact on how successful a given side channel attack will be. Unfortunately, there is no easy way to quantify how susceptible a given hardware is to side channel attacks. There are several methods that can be used as countermeasures against known side channel attacks. However, once again, there is no agreed upon measure for the effectiveness of these countermeasures.

At the moment reliable comparisons are only possible after algorithms have been implemented in hardware. Even such an investigation is not conclusive, since it only employs a limited set of attacks.

3 Suggestions

In our view, researchers which require practical results are always at a disadvantage. There are a multitude of problems associated with implementations. Worse yet, some of these problems may contribute significantly to the results. It is therefore important that they are handled with care. The following is a short discussion on the problems identified in the preceding section:

3.1 Selection of Algorithms

The selection of algorithms should be conducted by an independent group that includes cryptographers. First of all, this will help in implementing 'interesting' algorithms. In addition, the group that performs the evaluation will not be accused of favoring algorithms.

3.2 Cryptographic Properties

Once again, the independent group that decides on the algorithms should also clearly state which parts of the functionality of the algorithms are 'required' and which are 'optional'. The designers should be given a set of algorithms, that they can assume is equivalent.

If we take the eSTREAM project as an example, this groups could state that all algorithms should include a MAC. If not all algorithms have an integrated MAC, the group should also make a recommendation on how to complement the MACless design with a standard MAC.

[4] In fact, most synthesis results have an error margin of $\pm 5\%$. With slightly different (and seemingly irrelevant) compilation parameters it is possible to obtain a netlist that is 10% smaller (or larger) using the same VHDL code. This is a difficult problem to solve, since it is not a systematic error. It can be solved by making multiple synthesis runs, however to be effective this requires experienced designers.

[5] Actually error is misused here. After all the hardware is functionally correct.

3.3 Applications

The most important aspect in evaluation is to determine a target application for the evaluation. As noted earlier, hardware designers will try to allocate their resources in order to meet critical design requirements. These can vary greatly between applications.

A description like the following would make comparisons much simpler:

> The algorithm will be implemented for RFID applications, where power consumption as well as circuit area is extremely critical. The device will send out a 1024 bit encrypted message. The intended throughput of the device is in the range of 1-10Kbits/s (1 to 10 transfers per second). The 80-bit key will be programmed onto the device, but each transaction will use a new 64-bit IV.

Several factors are clear in this description:

- Emphasis is on power and area
- The throughput requirements are clear
- It is clear how and how often the device is initialized.

Several other scenarios can be described as well. It should be noted, that each scenario will result in a different implementation of the same algorithm. Therefore it would be better to assign one scenario per evaluation group (There could be multiple groups working on implementations).

3.4 FPGA/ASIC

The intended application will decide whether or not the implementation should be in FPGA or in ASIC. There is a very limited field where both FPGA and ASIC implementations are feasible. Therefore it would be one or the other, but not both.

For FPGA implementations, depending on the application, either a price range for the device should be set (i.e. 50-200$ per FPGA), or a known device should be chosen.

3.5 I/O Requirements

This has already been covered in the application section. The I/O requirements of the system should be defined. The data processing rate (throughput) should be matched to the application in order to prevent impractical rates.

As an example in the eSTREAM competition, if there are no other constraints, the algorithms with high throughput would be identified as hardware efficient. However, if the throughput was limited to 100 Mbits/s, with frequent key changes, other algorithms may have also fared better.

3.6 Reference Design

A well-known reference design should be included in the evaluation. This reference design should be implemented in the same way as the other algorithms (as opposed to using results from literature). In this way, the authors can be sure that the performance of the reference is measured in the same way as the rest of the algorithms.

3.7 Implementation Errors

Even with the best intentions and appropriate control mechanisms, it is not possible to exclude implementation errors. There are two solutions to this problem:

1. The reference VHDL (or Verilog) code could be provided by the algorithm designers.
2. At least two independent groups implement the algorithm.

3.8 Side Channel Attack Properties

Unless a reliable measure that defines the resistance against side-channel attacks, it would not be possible to compare algorithms in this regard. Only after the algorithms have been implemented, the cryptanalysts could perform attacks on the implementations. In a second design iteration, following the suggestions of cryptanalysts, countermeasures could be implemented, or the designs could be re-evaluated in this regard.

4 Summary

For best results, a group consisting of cryptographers and hardware designers should define the constraints for the implementation. This should include the throughput / area / power constraints, all relevant options that the hardware should support, and representative usage statistics (how often initialized, length of messages etc). The group should also decide on which algorithms should be compared in the study. To protect against implementation errors, the project should be assigned to more than one hardware designer group.