# ON SYMMETRIC CRYPTOGRAPHY WITH FPGAS

**Michael Vetter**
Study Course, 2-th class,part-time study
Supervisor: Vlastimil Vavricka

University of West Bohemia in Pilsen/ University of Applied Sciences Regensburg
Univerzitni 8, 306 14 Plzen, Czech Republic/Universitätsstrasse 31, Regensburg, BRD

michaelvetter@web.de

**Abstract.** This paper discusses the application of FPGA for cryptographic task. The advantages of FPGAs over other implementation options are discussed and the security challenges in FPGA designs are provided.

**Keywords**. FPGA,Security,Cryptography.

## 1 Introduction

The usage of FPGAs is on the rise [1]. In recent years FPGAs have been applied in many new fields of application [2]. There are several reasons why these applications benefit from FPGAs. First of all, FPGAs provide a higher computing power than general purpose processors and DSPs. Furthermore, the energy efficiency of FPGAs is significantly better than that of other microcontrollers [3]. Moreover, for small to medium sized productions FPGAs are cheaper than ASICs. And finally modern FPGAs can hold complex circuits with more than 2 million gates; providing enough capacity for most applications. But this development raises also questions about the security of FPGA based systems.
The following text describes why FPGAs are suitable for cryptographic algorithms, the design space to implement these ciphers is discussed and different implementation metrics are presented. A discussion of security risks associated with FPGA based cryptography and the challenges for secure, FPGA-based designs close the paper.

### 1.1 Advantages of FPGA based Cryptography

There are two ways to implement cryptographic operations: either in software or in hardware. The software solution is cheap and very flexible. Bugs can be fixed by installing patches, new parameters and even different ciphers can be applied this way. But, software is slow and (as it runs on a general purpose processor) not very energy efficient. Therefore, hardware solutions are preferred in applications with high bandwidth or low energy demand (e.g. wireless sensor node). Often these hardware implementations are done using an Application Specific Integrated Circuits (ASIC), designed and manufactured for this special application only. These ASICS provide the required features but they are expensive in terms of development and manufacturing. Moreover, ASICS are hardwired; bug fixing requires a new round of development.
FPGAs are a chimera between these two extremes; they provide a reasonable tradeoff between development costs, manufacturing, computing power and energy consumption (operations per watt). They perform massive parallel computation and can therefore provide a higher computing power than general purpose processors.

FPGAs are standard components that are available from a several vendors and distributors. There is no need to wait for a special hardware component to start the development process. Reference designs are available to ease PCB design and off-the-shelf boards can reduce the costs for hardware. Throughout the development process, bugs can be fixed very quickly by changing the HDL code, rebuilding it and loading it into the FPGA. Furthermore, the code can be tested and debugged on the actual hardware, a method that is faster and less error prone than the use of hardware simulators. The designer can also rely on a big number of HDL components, either under an open source or a commercial license.
In-field-upgrades can be used to add features, increase performance (if a more efficient implementation is available), fixing bugs, patching security holes (e.g. by replacing an obsolete cipher or changing the cryptographic parameters), and in rare cases bypassing hardware faults. This can be achieved by moving the configuration from a malfunctioned part of the FPGA to a functioning part. An example for such an approach is a satellite where a physical replacement of the chip is impossible.
Finally, FPGAs can be used as prototypes for "real" hardware. Either to test the HDL code or the PCB before the hardware is available. They can also be used to estimate the performance of a new algorithm in hardware.

### 1.2 Application for FPGA based Cryptography

Depending on the application, FPGAs can perform the following cryptographic tasks. It could be used as an external security crypto hardware (e.g. as a Trusted Platform Module). In such a scenario not only the FPGA itself but also the communication to and from the peripheral devices must be protected. Integration of all relevant components in one FPGA would hinder the interception and manipulation of sensitive data. However, the complexity of such a System-on-a-Chip could create new security flaws. Finally, the cryptography module can be used to ensure security secure operation of the FPGA itself, e.g. bitstream authentication [4] and secure upgrades [5].

## 2 Cipher Implementation in FPGA

This section provides a short overview how symmetric ciphers can be implemented in FPGAs. The following section describes the design decision that have to be made before the implementation can start, iterative vs. pipelining approaches are discussed and an example architecture for DES is presented. Different performance metrics are discussed and a short overview of different implementation concludes this section.

### 2.1 Design Space

Different Application stresses different requirements the cipher implementation has to fulfill. A discussion of nine design goals for an FPGA implementation of AES can be found in [6].
All block ciphers are using a number of, identical, rounds to destroy the structure of the ciphertext. Therefore, two different implementations are possible:

- An iterative approach, using the same hardware for several rounds.
- A pipelined version using several instantiations of the round function; latches are used for the synchronization between the different pipeline stages. A refinement of this approach is called sub-pipelining where latches are integrated in the rounds to improve performance.

The iterative solutions require less space, while pipelining provides a better performance in most cases. However, a pipelined solution is not always useful, as some cipher modes (like CBC) require the output of one block to generate the next.

### 2.2 Exploiting FPGA structures for cipher implementation

Based on the Virtex II architecture [6] describes several tactics for an efficient cipher implementation.

Pipelining can be used to improve the throughput of the implementation. Implementing a pipeline doesn't increase hardware utilization as every Look-up-table (LUT) is connected to a flip flop. But, this approach can lead to high data diffusion and therefore to a decreased performance. Therefore either the depth of the pipeline must be limited, or clocked registered must be used for data synchronization. The multiplexers of the FPGA can be used to store the S-Boxes in DistributedRam. An AES S-Box fits in 64 Slices of a Virtex II or 32 Slices of a Virtex 5 FPGA.The ability to configure LUTs as shift registers is also very useful. Especially Stream Ciphers rely heavily on these primitives. Additional Logic Gates can be used to improve the resource utilization. Modern FPGAs contain not only simple logic blocks but also more sophisticated and complex processing units. BlockRam or Embedded Ram can be used to store the S-boxes. However, using the 18Kbit Block for a 2 Kbit AES-Sbox would be a waste. Combining the SubByte and MixColumns operation (both are matrix operation on the state) creates four transformation tables with 8Kbits each; and speeds up the operation. But, as BlockRam have a fixed position (unlike distributed ram that is "created" in the logic blocks on demand), routing becomes harder and the maximum frequency of the design might decrease.For some ciphers the embedded multiplier can be used to speed up mathematical operations; e.g. to perform matrix operations for the SubByte operator in AES.
As presented in [7] a combination of BlockRam and DSP blocks of the Virtex 5 can be used to speed up AES. In this implementation a lookup in the 36 Kbit Block Ram is followed by 32 bit xor operation between the lookup result and the key. 8 Ram Blocks and 16 DSPs units are necessary to perform a single AES round. Embedded Processors could also be used to perform cryptographic operations. This processor could either be an embedded block (e.g. the PowerPC in some Virtex FPGAs) or it can be synthesized (like Xilinx 8-bit Picoblaze and 32-bit Microblaze). In both cases, processors can be used to perform cryptographic operation; either completes or only parts of it (e.g. key scheduling or cipher mode).

### 2.3 Benchmarking for Symmetric Ciphers

Several metrics are used to evaluate the performance of a symmetric cipher algorithm in FPGAs. These are:

- Hardware cost: utilized resources of the FPGA
- Operating Frequency: in MHz
- Throughput: measured in Mbit/sec
- Efficiency metrics: for example the relation between hardware utilization and throughput

These metrics are highly hardware depended. For example the commonly used Virtex II has 4-inputs Look-up-tables while the latest Virtex 5 has 6-inputs Look-up-tables, providing a higher capacity per slice. Another important aspect is the build tool and the parameters, e.g. if distributed ram is preferred over block ram, higher frequency vs. higher resource utilization. Different versions of the same build tool could produce different results – for better or worse. And none of these metrics consider the use of embedded processing blocks (e.g. Multipliers, DSPs and processor) beyond BlockRam. Finally, energy consumption is neglected completely.
The most thoroughly overview about AES implementations can be found in [8]. The authors reviewed more than forty different implementations of AES with a throughput between 0.259 GBps and 23 GBps- Some of the implementations are optimized for low area consumption while others use an aggressive pipelined design to provide maximum throughput. Some implement the SubByte function in a combinatorial logic, while others prefer lookup tables (either as BlockRam or Distributed Memory). The authors came to the result that "it is impossible to point out the absolutely best method, because all methods have their advantages and disadvantages".

### 2.4 Partial Runtime Reconfiguration

Most FPGAs today support Partial Runtime Reconfiguration (PRR). This feature allows a designer to change the FPGA configuration (either complete or partial) at any time. This enables the FPGA to fit the current computing demand of the application. This improves the efficiency of the FPGA by reducing the number of inactive components and reduces the required size of the FPGA. [9] describes an implementation of AES using Partial Runtime Reconfiguration. First the FPGA calculates the Key Schedule, in the second step the FPGA is reconfigured to perform the encryption using the pre-computed session keys. However, to the authors' knowledge no other work about symmetric ciphers and partial reconfiguration exists.
Future designs could provide a variable number of pipeline stages. Aggressive pipeline enrollment could be used to provide a high throughput, while an iterative strategy could be used to reduce the resource utilization. But, such a design is hard (if not impossible) to implement with today's tools. It's not possible to create one configuration (e.g. for an AES round) and to install this configuration on multiple positions of the FPGA, but one partial configuration stream for each stage must be created. Data propagation through the pipeline and back to the control logic provides another challenge as the stage has to know if it is the final stage or not.
However PRR creates also new risks as the designs get more and more complex and therefore more error prone.

## 3 Security Aspects of FPGA based Cryptography

### 3.1 Security Challenges in FPGA Design

There are several security threats [10] that must be addressed to ensure a secure operation of an FPGA. The three most important challenges are: the secure programming of FPGAs, the protection of intellectual property against theft or illegal usage and finally side channel attacks that can reveal security relevant information to an attacker. This section provides a short description of these threats and the methods that are applied against them.
**Secure Programming**: almost all FPGAs are using volatile SRAM to store the configuration. Therefore they have to be programmed at every startup. An external flash memory is used to store the configuration. In most cases the transfer between this memory and the FPGA is done in plaintext. This approach leads to several vulnerabilities. First the transmission can be intercepted and later analyzed (more about this in the next section). Second, an invalid configuration can be loaded in the FPGA. Third, changing the configuration during the configuration process is possible. Encrypting the data stream is one possible solution against these attacks. Some FPGAs support the encryption of the configuration file. But only the premium lines (e.g. Xilinx Virtex and Altera Stratix II) offer this option; low-cost FPGAs lack this feature at the moment. Moreover, as the memory of the FPGA is volatile, some designs need a buffer battery for the key. Fourth, the flash can be replaced to load a faked configuration in to the FPGA. A possible countermeasure against this kind of attacks requires an authentication [4] of the flash memory or the configuration file. Integrating the FPGA and the Flash in one package, as done by Xilinx for their S3AN Family [11], increases only the costs for an attack but could not prevent it completely.
**Protection of Intellectual Property (IP):** The IP represents a serious investment of the involved cooperation's and is therefore the most valuable part of the FPGA. Reverse Engineering [12] allows an attacker to rebuild the system, find security flaws in the design or could allow him to alter the configuration (e.g. to add hidden features, examine the communication inside the FPGA or to remove copy protection mechanism). Cloning of an existing design would lead to a loss of profit for the vendor. Secure programming is one critical component to protect the IP, but more versatile methods like dynamic Digital Rights Management and secure update methods are also required.

**Side Channel Attacks**: Different operations or operands of an electronic circuit are leaking information from the chip to the outside world. The most common kind of these attacks is power analysis [13] [14], where an attacker measures the power consumption throughout cryptographic operations. The measured power consumption is compared to a theoretical power model and the most probable operand is derived. Experiments have shown that FPGAs are vulnerable against these kinds of attacks [14]. For a detailed overview about power analysis attacks against FPGAs refer to [15].

### 3.2 Assets and Threats of Cipher Implementations

An attacker could try to retrieve or replace the **Plaintext** or the **Ciphertext.** Injection of a chosen plaintext could be used to retrieve secret key of a weak cipher, while access to the plaintext would render the encryption useless. Encryption could also be abused to prevent an audit of stored and transmitted data.

The **Key** itself is also a valuable target for an attacker. He can either try to retrieve it or replace it (e.g. by a null key). But attacks are not limited to the very key itself but also to other, temporary created data. These data include the round keys and auxiliary data used to create the session key (e.g. a random number generator or master key). This sensitive information could not only be retrieved by an attack but also be leaked by a faulty design, like improper resource management in combination with PRR.

Finally, the **Cipher** itself can be attacked. If the designers decided to choose security-by-obscurity an analysis of the cipher can reveal its flaws. Alterations on the cipher are also a threat. This way an attacker can reduce the security of the encryption e.g. by reducing the number of encryption rounds. Furthermore, false results can lead to a denial-of-service attack (e.g. if the cipher is used to generate a Message Authentication Code) or force the user to turn off the encryption. The integration of side channels is another option to volatile the integrity of the system.

Attacks can be carried out at any stage of the product lifecycle. Logic Bombs can be integrated by the FPGA designer and the manufacturer. In-the-field an attack could either change the configuration of the FPGA itself (using e.g. partial reconfiguration over leftover JTAG port) or the configuration in the flash memory.

### 3.3 Involved Parties, existing Security Solutions and open Questions

The involved parties can be divided into the following groups. The **IP (Intellectual Property) Owner** and the **Device Manufacturer** wants to protect his assets from illegal usage and reverse engineering. A **Service Provider** (e.g. pay-tv provider) wants to ensure that the device protects his business assets by enforcing his access conditions (e.g. no unencrypted data outside the set top box). Finally, the **Device Owner** has legitimate interest in a fair use of his property. He also wants to protect his personal data from illegal access while maintaining control over his device. Encrypted communication could be abused to circumvent this control.

Partial security solutions exist to these divergent requirements. At the development phase, for example, formal verification methods [16] can be used to proof the correctness of the implementation, and VHDL-2008 introduced the encrypted distribution of Intellectual Property. Other proposals address the problem of access control [17] and resource utilization [18].

However, no secure design and development process (like Microsoft's SDL [19]) exists at the moment. Such a process must keep track of the security requirements of all involved parties. Future research will concentrate on this challenge.

## 4 Conclusion

FPGAs provide a powerful platform for cryptographic operations. The combination of high computation power and flexibility allows designer to find the best solution for their application. The

great number of available cores allows a quick integration of FPGA based cryptography for most applications. However, the secure operation of these ciphers (and other security relevant operations) remains an open question. The growing complexity of modern FPGA designs demands new solutions and the establishment of a security aware design and development process.

## References


[1] In-Stat, "FPGA shipments to reach $2.75 bln by 2010," 2006, http://www.itfacts.biz/fpga-shipments-to-reach-275-bln-by-2010/6587.

[2] S. Hauck and A. DeHon, Reconfigurable Computing, Morgan Kaufmann/Elsevier, Amsterdam [u.a.],2008.

[3] Brodersen;Bob, "General Purpose, Low Power Supercomputing Using Reconfiguration,"., http://video.google.com/videoplay?docid=-4969729965240981475.

[4] S. Drimer, "Authentication of FPGA bitstreams: why and how," Applied Reconfigurable Computing, Springer, 2007, pp. 73-84.

[5] S. Drimer and M.G. Kuhn, "A Protocol for Secure Remote Updates of FPGA Configurations," Reconfigurable Computing: Architectures, Tools and Applications, 5th International Workshop, ARC 2009, 2009, pp. 50-61.

[6] Franois-Xavier Standaert, "Secure and efficient implementation of symmetric encryption schemes using FPGAs", 2007.

[7] S. Drimer, T. Güneysu and C. Paar, "DSPs, BRAMs and a pinch of logic: new recipes for AES on FPGAs," Field-Programmable Custom Computing Machines, 2008. FCCM '08. 16th International Symposium on, 2008, pp. 99-108.

[8] K. Jarvinen, M. Tommiska and J. Skytta, "Comparative survey of high-performance cryptographic algorithm implementations on FPGAs", Information Security, IEE Proceedings, vol. 152, no. 1, 2005, pp. 3-12.

[9] O. Pérez et al., "The Use of Runtime Reconfiguration on FPGA Circuits to Increase the Performance of the AES Algorithm Implementation", j-jucs, vol. 13, no. 3, 2007, pp. 349-362.

[10] M. Kucera and M. Vetter, "FPGA Rootkits," Solutions in Embedded Systems (WISES '07) Solutions in Embedded Systems (WISES '07), 2007.

[11] Xilinx, Spartan-3 Generation Configuration User Guide, 2006.

[12] J. Note and E. Rannaud, "From the bitstream to the netlist," 2008, http://www.ulogic.org/~jb/debit/bitstream.pdf.

[13] S. Mangard, E. Oswald and T. Popp, Power Analysis Attacks, Springer Science+Business Media, LLC, Boston, MA,2007.

[14] S.B. Ors, E. Oswald and B. Preneel, "Power-analysis attacks on an FPGA - first experimental results," Cryptographic Hardware and Embedded Systems Workshop, Springer-Verlag, 2003, pp. first experimental results.

[15] O. Standaert et al., "An Overview of Power Analysis Attacks Against Field Programmable Gate Arrays", Proceedings of the IEEE, vol. 94, no. 2, 2006, pp. 383-394.

[16] T.K. Tolstrup, F. Nielson and H.R. Nielson, "Information Flow Analysis for VHDL," Parallel Computing Technologies, Springer, 2005.

[17] M. Kucera and M. Vetter, "A Generic Framework to Enforce Access Control in FPGAs with Dynamic Reconfiguration," Software Engineering and Applications - 2007, ActaPress, 2007.

[18] K. Markus and M. Vetter, "On Secure Resource Utilization in FPGAs with Partial Runtime Reconfiguration," Software Technologies for Future Dependable Distributed Systems, 2009.

[19] M. Howard and S. Lipner, The security development lifecycle, Microsoft Press, Redmond, Wash,2006.