An FPGA based Accelerator for Encrypted File Systems

Adrian Matoga*, Ricardo Chaves*,
Pedro Tomás*, Nuno Roma*,1,2

* *INESC-ID, Rua Alves Redol 9, 1000-029 Lisboa, Portugal*

ABSTRACT

In this work, the possibility of application of an AES coprocessor to increase the performance of encrypted file systems is examined. A brief description of an architecture based on an FPGA connected to a GPPU using PCI Express is followed by the preliminary evaluation of its performance compared to a software-only solution, as well as a discussion of bottlenecks and ways to deal with them.

KEYWORDS: AES; FPGA; PCI Express; file system; EncFS; OpenSSL

1 Introduction

The availability of increasingly powerful FPGA devices and hybrid architectures with CPU and FPGA connected with fast links such as PCI Express has been driving the research towards offloading computationally intensive tasks to specialized accelerators implemented on FPGAs. The main advantage of such accelerators is the relative easiness of reconfiguration of the FPGA when a more efficient implementation of a particular algorithm is available or a different function is needed.

In this work, an approach to accelerate an encrypted file system using an FPGA based cryptographic engine is presented, which is expected to take advantage of the high throughput hardware implementation. A fully functional prototype system was implemented and characteristics of its performance were measured. While the obtained speedup in relation to the software implementation is only about 3x, several feasible optimizations that may increase it significantly are proposed.

[1]E-mail: {Adrian.Matoga,Ricardo.Chaves,Pedro.Tomas,Nuno.Roma}@inesc-id.pt

[2]This work was partially supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under the project "HELIX: Heterogeneous Multi-Core Architecture for Biological Sequence Analysis" with reference PTDC/EEA-ELC/113999/2009, and project PEst-OE/EEI/LA0021/2011.

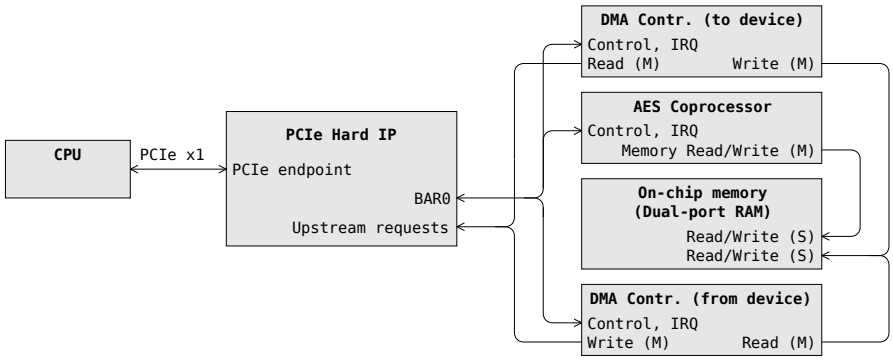

Figure 1: Overview of the architecture of the cryptographic accelerator hardware.

The remainder of this article is organized as follows. In Section 2, the architecture of the developed prototype system is decribed. In Section 3, the results of preliminary evaluation of the accelerator performance are presented and discussed. The last section concludes and discusses future directions.

2 Architecture

2.1 Software

The file system used to evaluate the accelerator is EncFS [Enc]. It is based on FUSE (Filesystem in USErspace) and it works by providing an unencrypted view of each encrypted file from its underlying storage directory under the mountpoint. The EncFS is written in C++ and its architecture permits adding ciphers by implementing key setup and encode/decode functions, defined by the key and cipher abstract classes. It originally uses the OpenSSL library to provide AES encryption in CBC mode with 128-, 192- or 256-bit key, making it possible to reasonably compare it with the accelerator. A Linux device driver and a set of C++ classes were implemented to make the hardware functions available to EncFS.

2.2 The accelerator

An overview of the architecture of the hardware accelerator is presented in Figure 1. The PCI Express Hard IP core [Alt11b] constitutes a bridge between the PCIe interface and a memory mapped interface [Alt11a] where control registers of the other cores are attached.

The cryptographic core used in this work is the AES coprocessor proposed in [CKVS06], which implements the AES cipher, first proposed in [DR98]. Encryption and decryption with 128-, 192- and 256-bit keys, as well as the ECB and CBC block cipher modes of operation, are implemented. The core includes a cryptographic engine and a wrapper to access blocks of data in an on-chip memory. The cryptographic engine itself is able to execute one round of the AES algorithm per clock cycle, thus giving the raw throughput of 190 MB/s, 159 MB/s and 136 MB/s for key length of 128, 192 and 256 bits, respectively.

Cryptographic keys, as well as the data to be encrypted or decrypted, must be first transferred to a buffer in the on-chip RAM, which is done using one of the two DMA controllers. The other DMA controller is used to transfer the processed data back to the host. The DMA controllers may work simultaneously, thus exploiting the duplex nature of the PCIe link.

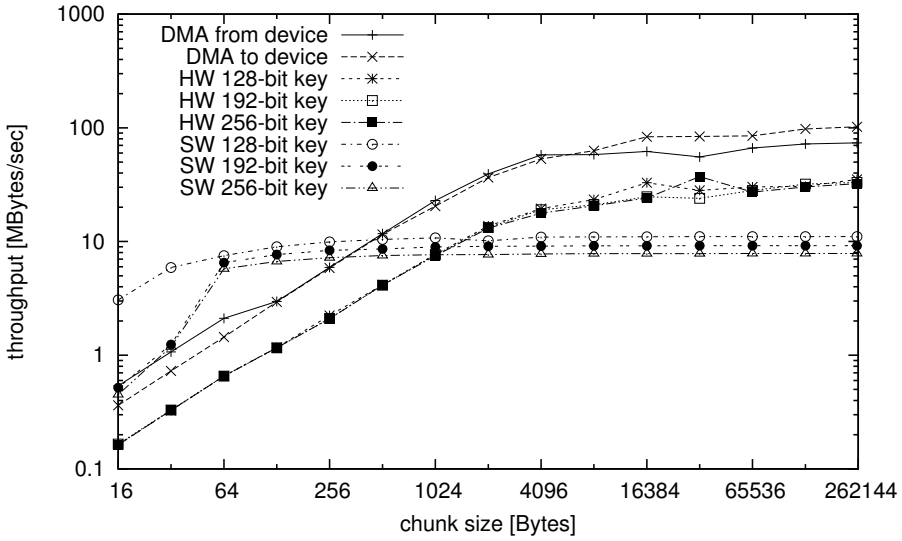

Figure 2: Throughput comparison between the cryptographic core (HW) and software AES implementation from OpenSSL (SW) for different key lengths. DMA transfer speed is also included for comparison.

3 Evaluation

3.1 Evaluation platform

The accelerator was implemented on a Kontron MSMST board incorporating an Intel Stellarton processor, which includes an Intel Atom CPU at 1.3 GHz and an Altera Arria II GX FPGA, connected with two PCIe 1x links [Kon]. In the described prototype, only one bidirectional PCIe link is used. The system implemented in the FPGA is driven by the 125 MHz application clock from the PCIe bridge.

3.2 Results

Figure 2 compares the performance of the hardware accelerator to the performance of the software only implementation from the OpenSSL library. Both implementations run in the CBC mode. The smallest chunk size is 16 bytes, which is the size of the block encrypted at once by the AES algorithm. The largest chunk size is 256 KB, which is the limit imposed by the size of the on-chip RAM.

As it can be seen from the graph, the throughput of the software implementation is almost constant for chunks larger than 64 bytes, reaching about 11, 9.2 and 7.8 MB/s for 128-, 192- and 256-bit key, respectively.

The throughput for the hardware implementation is measured considering the total time required to send a chunk of data to the device, process it, and retrieve the results. Besides the great overhead imposed by the data transfers, there is also an overhead resulting from setting up the device to perform an action, such as encryption or transfer, putting the process to sleep and waking it up when the interrupt that signals the completion of the operation arrives. This control overhead corresponds to the uniform performance growth up to about 2 KB per chunk, at which point the time of the actual operation becomes more significant.

Since the DMA transfers take a major part of the total time, the throughput difference of the AES core in what concerns the key length becomes insignificant. The overall throughput in the useful region, where the accelerator is faster that the software implementation, is from 14 MB/s for 2-KB chunks to 28 MB/s for 256-KB chunks, thus giving a speedup of up to 3x over the software implementation.

In a test performed on a large set of different types of files, it was revealed that over 90% of the chunks, which EncFS requests from the cipher implementation to encrypt or decrypt, were of 32 bytes or less, making the point of using the accelerator disputable. However, several measures can be taken to overcome the observed limitations.

For a file system like EncFS, where typical chunks are small, one could use a hybrid solution, i.e. dispatch the execution to hardware or software based on the chunk size, aiming at using always the fastest of available implementations for a given chunk size. If a file system that performs encryption at a lower level was used, larger chunks would be processed—the minimum chunk size in such case is expected to be between 512 bytes (the usual physical disk sector size) and 4 KB (the usual size of a virtual memory page).

The throughput of the accelerator itself can also be improved by executing the three stages (transfer to the device, encryption, transfer to the host) in a pipeline. The throughput then would be limited by the slowest stage, in this case the DMA transfer from the device with 70 MB/s. Further improvements can be made by changing the memory based architecture to a streaming one, where the data processing starts immediately after the reception of the first 16-byte block.

4 Conclusion and future work

The architecture and preliminary evaluation of the developed AES accelerator was described. Its throughput was compared to the equivalent software implementation from OpenSSL, and the speedup of up to 3x was obtained. However, the accelerator becomes less efficient that the software implementation when the considered file system requests the encryption of small chunks. Possible improvements in both the architecture of the accelerator and the way it is applied were proposed to circumvent such difficulties.

References

[Alt11a] Altera. *Avalon Interface Specifications*, May 2011.

[Alt11b] Altera. *IP Compiler for PCI Express User Guide*, May 2011.

[CKVS06] Ricardo Chaves, Georgi Kuzmanov, Stamatis Vassiliadis, and Leonel Sousa. Reconfigurable memory based aes co-processor. In *in Proceedings of the 13th Reconfigurable Architectures Workshop (RAW 2006)*, page 192, 2006.

[DR98] Joan Daemen and Vincent Rijmen. AES Proposal: Rijndael, 1998.

[Enc] EncFS Encrypted Filesystem. http://www.arg0.net/encfs.

[Kon] Kontron. *Kontron User's Guide: MSMST*.