CCproc: A custom VLIW cryptography co-processor for symmetric-key ciphers

Dimitris Theodoropoulos‡†, Alexandros Siskos†, and Dionisis Pnevmatikatos†

†ECE Department, Technical University of Crete,
Chania, Greece, GR73100,
gsiskos@electronics.tuc.gr pnevmati@mhl.tuc.gr

‡Computer Engineering Laboratory, EEMCS, TU Delft, P.O. Box 5031,
2600 GA Delft, the Netherlands
D.Theodoropoulos@tudelft.nl

Abstract. In this paper, we present CCProc, a flexible cryptography co-processor for symmetric-key algorithms. Based on an extensive analysis of many symmetric-key ciphers, including the five AES finalists, we designed an Instruction Set Architecture tailored to symmetric-key ciphers and built a hardware processor prototype by using the VHDL language. The design was mapped on FPGAs and ASIC. Results show a small-area design, while also supporting many ciphers. Besides flexibility, a 4-core FPGA design can achieve up to 615 Mbits/sec at 95 MHz for Rijndael.

Key words: Cryptography, reconfigurable processors, VLIW

1 Introduction

In this paper, we focus on the encryption and decryption processes for symmetric key cryptography ciphers, in order to identify common processing parts among them, and be able to design an ISA (Instruction Set Architecture) that efficiently supports them. These similarities were deeply analyzed and the result is a hardware VLIW co-processor called CCproc (Cryptography CoPROCessor). CCproc has its own symmetric-cipher-specific instruction set and an extended RISC datapath structure. Furthermore, it is capable of supporting many of today's symmetric key ciphers, plus new potential ones, while also functioning at very competitive speeds.

The main contributions of this work are:

- An extensive symmetric-key ciphers analysis, based on studying many cryptography algorithms;
- The design of an efficient and flexible cryptography co-processor (CCproc), based on the results of the aforementioned analysis.

The remainder of the paper is as follows: Section 2 discusses the main symmetric-ciphers properties and some related work, while Section 3 describes the CCproc architecture. Section 4 compares CCProc against other similar designs, while Section 5 concludes the paper.

2 Symmetric-Key Ciphers Properties and Related Work

In order to make our analysis as complete as possible, the following algorithms were selected: Rijndael [10], MARS [5], Twofish [4], RC6 [15], Serpent [16], Blowfish [3], RC4, DES, RC5 [14] and International Data Encryption Standard (IDEA) [13]. We concluded that the most commonly used operations and structures are: 1) unsigned addition and subtraction modulo 2^{32}, 2) multiplication modulo (MM) 2^{32}, 3) 32-bit xor, 4) fixed shifts and rotations, 5) data dependent shifts and rotations, 6) finite field polynomial multiplication in the Galois Field 2^8 modulo an irreducible polynomial, 7) expansions and permutations (Xboxes), 8) substitution boxes (S-boxes) and 9) Feistel network structures [9].

In 32-bit processors, operations from 1 to 5 are implemented very fast. Finite Field polynomial Multiplication (FFM) modulo an irreducible polynomial does not have direct hardware support. Additions, subtractions and XORs are the simplest operations and are used just to scramble data, thus providing less security. Fixed rotations are mainly used to get specific data bits to places, from where they will be used by other operations. Data dependent rotations can be combined with arithmetic operations, thus protecting against linear cryptanalysis.

Besides arithmetic operations, there are common structures among ciphers. A symmetric cipher may have one or more different S-boxes (Look-Up-Tables) of arbitrary dimensions. Permutations and expansions are linear operations, and thus, not sufficient to guarantee security. However, when used with good non-linear S-boxes, they are vital for the security of a cipher, because they propagate the non-linearity uniformly over all bits. Finally, Feistel network is a common structure used by many symmetric ciphers and consists of all processing rounds along with their inner operations [9].

Regarding related work, Wu et al. in [11] introduce the *Cryptomaniac* cryptographic processor, a 4-wide 32-bit VLIW machine with no cache and a simple branch predictor, giving a throughput of 512 Mbits/sec for Rijndael. In [6], Oliva et al. describe the *Cryptonite*, a programmable processor. Its frequency is 400 MHz in TSMC 0.13 μm process. Results show a 700 Mbits/sec Rijndael performance. Elbirt et al. in [1] describe a design named COBRA, a specialized reconfigurable architecture optimized for the implementation of block ciphers, achieving a theoretical throughput of 3.9 Gbits/sec, 1.451 Gbits/sec and 2.306 Gbits/sec respectively.

3 CCproc Architecture

Design Considerations: Our initial motivation was a hardware design, flexible enough to support many of today's popular symmetric ciphers, and potential new ones [7]. As years go by, symmetric ciphers that use keys smaller than 128-bits are likely to be abandoned, because they will be vulnerable to brute-force attacks. *So, after a careful analysis, we concluded that some of their functional principles were not adopted by the newest ones.* A first example is bit permutation or

expansion, which is primarily used by DES, however, none of the AES round 2 finalists used it. *Although different structures have been studied offering arbitrary bit permutations, they require a considerable amount of hardware [2], thus we decided not to use them.* Another characteristic not used by any of the AES round 2 finalists is the variable S-boxes. *Based on this, we decided to instantiate a few ROMs as cipher specific S-boxes and small RAMs for new S-boxes support, plus available data space during the key expansion process.*

Regarding the key expansion process, all AES round 2 finalists utilize rather simple ones, with the exception of MARS. *As a result, in order to explicitly support the key expansion process of a cipher, the only extra functional unit that we would add is a Key Register File (KRF) memory module, where all expanded keys will be stored. Furthermore, we employed an ISA expansion to support operations between Register File (RF) data and KRF data.* Another consideration was to record the instruction types and their frequency in symmetric ciphers. This research revealed a high frequency occurrence of 2 dependent back-to-back instructions, like add-add, add-sub and add-xor. *Thus, we decided that this type of double-instructions should be included in the CCproc ISA.*

We also noted that all AES round 2 finalists treat 128-bit plaintext as 4 32-bit words. Symmetric ciphers frequently require 64-bit, 96-bit or even 128-bit data values at the same time in order to proceed. *Having in mind a design that would achieve competitive performance, it was finally considered a VLIW processor that would consist of 4 32-bit clusters, capable of processing 4 32-bit instructions in 1 clock cycle.*

Another conclusion that came up from our cipher analysis was the small number of 32-bit registers utilization. In each cluster a maximum of 4 registers were used during out tests, ending up to a total of 16 32-bit registers among the 4 clusters. *However, it was decided an 8x32 RF in each cluster, in order to cover the case where a cipher, that was not tested, might need additional registers.* Furthermore, a fact that characterizes every symmetric cipher is the determined rounds number for the key expansion and encryption/decryption processes. Consequently, they can be implemented in a way that requires no branch tests. *Thus, we decided to support only a "loop" instruction that would add no stalls.*

A final consideration was cipher support. As we mentioned earlier in this section, every cipher that uses keys smaller than 128-bit are considered as non-secure. In addition, a cryptography co-processor should be able to support as many ciphers as possible, in order to provide a strong security level against various kinds of attacks [8]. *This fact led us to the decision to support all AES round 2 finalists, since these are the strongest ones, and to have an extended ISA efficient and general enough to cover future algorithms.*

Datapath Structure: As mentioned before, after a closer analysis of various symmetric ciphers, we concluded that a *loop* instruction was enough to handle all control hazards. In addition, ciphers processing required at most 4 32-bit registers per cluster for each of the AES round 2 finalists. As a result, RF in each cluster consists of 8 32-bit registers, to support not tested ciphers that

might need more registers. The KRF is a special RAM in each cluster *decode* stage, where all expanded keys of a cipher are stored. Every KRF has 64 32-bit registers. *This size was chosen after observation of the expanded keys data size and finding that it did not exceed a total of 33 32-bit data values per cluster.* Since every cipher uses its expanded keys serially forwards or backwards, we decided to use a serial auto-increment KRF address generation approach.

As mentioned before, there are many double-instructions that require 3 operands. This fact led us to design an ALU with 3 32-bit inputs and 1 32-bit output. The ALU consists of 3 32-bit ASUs (Addition/Subtraction Units), 3 2-input 32-bit xors and 3 multiplexers. The ALU instructions have the specific format $Result \Leftarrow ((In1 \quad op1 \quad In2) \quad op2 \quad In3)$, where *In1*, *In2* and *In3* are the 3 ALU inputs and *op1*, *op2* are the 2 operations that may be performed. The ALU has also 2 shift/rotate units. CCproc includes also multipliers modulo (MM) 2^{32} and Galois Field multipliers (GFM) modulo an irreducible polynomial over $GF(2^8)$. Our GFM is based on [12] and can perform 16 8x8 GF multiplications in 1 clock cycle.

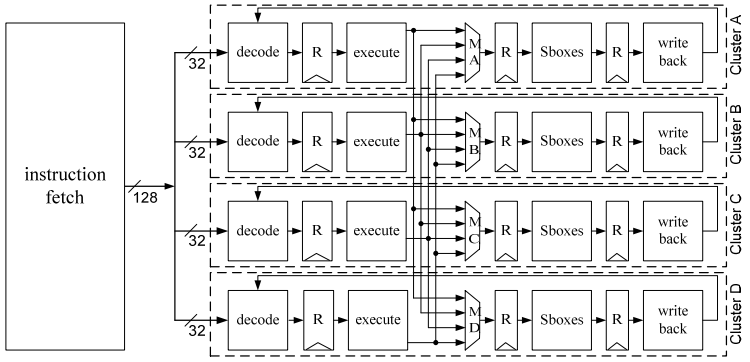

Fig. 1. CCproc schematic overview

Figure 1 illustrates the complete design. There is an *instruction fetch* unit for fetching a 128-bit instructions and passing them to the 4 clusters as 4 32-bit instructions. Each one of these instructions is forwarded to the *decode stage* of its corresponding cluster for further processing. The main unit in this stage is the *Decode controller*, which produces valid RF and KRF addresses, plus many other control signals that pass through the next stages. Next stage is the *execution stage*, where all logic and arithmetic operations are performed. The ALU, GFMs 16x8x8, MMs and shifters/rotators are all instantiated in this stage. Data from *execution stage* are used to access the appropriate cipher S-boxes or forwarded directly to the *write back stage* where data are written in RF. *MX* multiplexers (X=A, B, C, D) are used to efficiently exchange data among clusters in 1 clock cycle, while "R" represents a pipeline register.

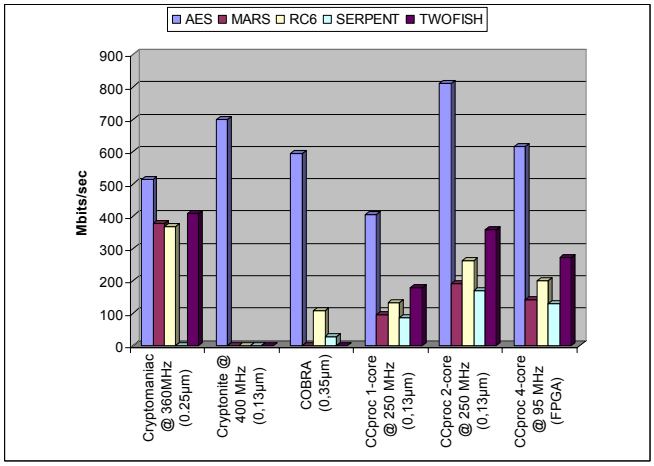

Fig. 2. Performance Comparison

4 Verification and Performance Evaluation

CCproc needs 79, 338, 242, 375 and 178 cycles for Rijndael, MARS, RC6, Serpent and Twofish respectively. Based on the above results, a 4-core FPGA implementation mapped on XC4VLX200 FPGA at 95 MHz and ECB mode yields a throughput of 615, 143, 201, 129 and 273 Mbits/sec for Rijndael, MARS, RC6, Serpent and Twofish respectively. A 1-core FPGA design is small enough (18045 slices) to fit in a XC4VLX40. We also developed an ASIC implementation of a 1-core design with Synopsys' 0.13μm UMC Library. Our design utilizes 93K cells, and its total cell area is 5.3mm^2, out of which 0.73mm^2 was the combinational logic. The rest is devoted to the instruction memory, register files and S-boxes. A 1-core ASIC implementation yields a throughput of 405, 95, 132, 85, and 180 Mbits/sec for Rijndael, MARS, RC6, Serpent and Twofish respectively.

A comparison between CCproc ASIC / FPGA implementations and other similar designs is shown in Figure 2. Cryptomaniac [11] is a flexible design, because it supports most of the current ciphers compared to the others. Cryptonite [6] supports only Rijndael from ciphers with 128-bit key. COBRA [14], according to the paper, currently supports 2 parallel atomic units running for Rijndael and RC6, and 1 atomic unit for Serpent. As it can be seen from Figure 2, CCproc is the only one that supports so far all AES round 2 finalists, while also mapped in FPGA and ASIC. A just 2-core CCproc ASIC implementation can deliver speeds of 810, 190, 264, 170, and 360 Mbits/sec for Rijndael, MARS, RC6, Serpent and Twofish respectively. Of course we should note that Cryptomaniac has the advantage of occupying smaller area than CCproc. Cryptonite yields better throughput than CCproc and Cryptomaniac where applicable, however it lacks of flexibility. However, today's cryptography co-processors should support multiple ciphers and have the capability to be reconfigured and upgraded in case a cipher is broken. [8].

5 Conclusions

In this paper, we analyzed many symmetric-key ciphers and designed from scratch a VLIW RISC-like co-processor, in order to efficiently support them in hardware level at very competitive speeds. It supports all AES round 2 finalists and can fit in small FPGAs. Furthermore, multiple CCproc cores can be placed in larger FPGAs to improve cipher performance.

References

1. A. J. Elbirt et al. An Instruction-Level Distributed Processor for Symmetric-Key Cryptography. In *IEEE Trans. on Parallel and Distributed Systems*, pages 468–480, 2005.
2. A. Murat Fiskiran et al. Performance Impact of Addressing Modes on Encryption Algorithms. In *ICCD*, pages 542–545, 2001.
3. B. Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). In *Fast Software Encryption, Cambridge Security Workshop Proceedings*, pages 191–204.
4. Bruce Scheier et al. Twofish: A 128-bit Block Cipher. 1998.
5. C. Burwick et. al. MARS - a candidate cipher for AES. 1999.
6. D. Oliva et al. AES and the Cryptonite Crypto Processor. In *CASES*, pages 198–209, 2003.
7. Dimitris Theodoropoulos et al. Cproc: An efficient Cryptographic Coprocessor. In *16th IFIP/IEEE International Conference on Very Large Scale Integration*, 2008.
8. T. Huffmire. Application of cryptographic primitives to computer architecture. Technical report, University of California, 2005.
9. J. L. Smith. Description of a new variable-length key, 64-bit block cipher (Blowfish). In *Fast Software Encryption, Cambridge Security Workshop Proceedings*, pages 191–204.
10. Joan Daemen et. al. AES Proposal: Rijndael. Document Version 2. 1999.
11. Lisa Wu et al. Cryptomaniac: A Fast Flexible Architecture for Secure Communication. In *ISCA*, pages 110–119, 2001.
12. M. Jung, et al. A Reconfigurable Coprocessor for Finite Field Multiplication in $GF(2^8)$. In *IEEE Workshop on Heterogeneous reconfigurable SoCs*, 2003.
13. MediaCrypt. International Data Encryption Algorithm. Technical report.
14. Ronald L. Rivest. The RC5 Algorithm. 1996.
15. Ronald L. Rivest et al. The RC6 Block Cipher. 1998.
16. Ross Anderson et al. A Proposal for the Advanced Encryption Standard. In *5th workshop on Fast Software Encryption*, 1998.