PERFORMANCE EVALUATION OF CRYPTOGRAPHIC ALGORITHMS IN SOFTWARE AND WITH HARDWARE IMPLEMENTATION OF SPECIALIZED INSTRUCTIONS FOR THE RISC-V INSTRUCTION SET ARCHITECTURE

By Gorkem Nishandji

A Thesis

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Science

in Informatics

Northern Arizona University

December 2021

Approved:

Paul Flikkema, Ph.D., Chair

Bertrand Francis Cambou, Ph.D.

Michael Gowanlock, Ph.D.

ABSTRACT

PERFORMANCE EVALUATION OF CRYPTOGRAPHIC ALGORITHMS IN SOFTWARE AND WITH HARDWARE IMPLEMENTATION OF SPECIALIZED INSTRUCTIONS FOR THE RISC-V INSTRUCTION SET ARCHITECTURE

GORKEM NISHANDJI

The ever-increasing need for securing computing systems using cryptographic algorithms is spurring interest in efficient implementation of common algorithms. While the algorithms can be implemented in software using the base instruction set of processors, there is considerable potential to reduce memory cost and improve speed using specialized instructions and associated hardware. However, there is a need to assess the relative benefits and costs of software implementations and new instructions that implement key cryptographic algorithms in fewer cycles. The primary aim of this thesis is to improve understanding of the cost of implementing cryptographic algorithms for the RISC-V instruction set architecture (ISA) for both implementations of the algorithms in software using the base RV32I instruction set and with implementation of instructions as hardware in additional functional units. RISC-V was designed a decade ago. Since then, the ISA has been improved and extended with new instruction sets, where the RISC-V cryptography instruction set extension is one of the upcoming instruction set extensions. This thesis provides hand-optimized RISC-V assembly language implementations of eleven cryptographic algorithms with and without the cryptography extension. For a cost-benefit analysis, a RISC-V processor with the cryptography extension is designed in hardware. Also, a new instruction is proposed to increase the implementation efficiency of the cryptography algorithms. The results show that software implementation requirements can change significantly according to algorithm and implementation method. Compared to implementations with only the base RV32I instruction set, implementations with the cryptography set extension provide 1.5X to 8.6X faster execution speed and 1.2X to 5.8X less program memory for five of the eleven algorithms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

1.1 Instruction Set Architecture

A processor is a digital circuit that executes instructions to drive a computing system. An Instruction Set Architecture (ISA) defines the abstract model of a processor. It provides details about instructions, registers, memory model, instruction encoding, interrupt handling, and modes of operation. Essentially, it defines the response of a processor to a machine code. Consequently, the ISA of a processor affects both the hardware and software designs of a computing system.

RISC (Reduced Instruction Set Computer) and CISC (Complex Instruction Set Computer) are two main ISA types. The debate between these two types has been an important topic since the 1980s [14] [30] [13]. Compared to CISC, RISC architectures have simpler designs due to reduced instruction count and simplified instruction functionality. The minimalist design methodology of the RISC architectures leads to smaller hardware design but larger static code size [14]. Although initially RISC architectures dominated low-power systems and CISC architectures dominated desktops and servers, both types appear in both areas nowadays [14].

RISC-V is an open-source RISC ISA, which has been started in 2010 at UC-Berkeley [6]. In 2011, the team published volume one of the RISC-V ISA manual [47]. In 2015, the RISC-V International Foundation [1] was founded to build a RISC-V ISA community. RISC-V has support for privileged [4] and unprivileged [3] instruction sets. RISC-V has 32-bit and 64-bit versions, and currently, it has eight ratified instruction set extensions (-M, -A, -F, -D, -Q, -C, -Zicsr, -Zifencei) and the base instruction set (-I) [3]. However, there are several upcoming extensions.

One of these upcoming extensions is the RISC-V cryptography extension (-crypto), which has two sets of instructions – Scalar & Entropy Source [9], and Vector [5]. The scalar extension has groups of instructions. First group includes instructions that are borrowed from upcoming bitmanip extension [7] and other 5 groups are designed to accelerate particular cryptography algorithms - AES [35], SHA-256 [36], SHA-512 [36], SM3 [2], and SM4 [20]. The scalar extension provides both 32-bit and 64-bit versions of the instructions. Similarly, the vector cryptography extension provides vector versions of some of the bitmanip instructions and vector instructions to accelerate the AES and the SHA2 [19] cryptography algorithms.

1.2 Literature Review

This section discusses the work that has been done on RISC-V ISA cryptography extensions.

In [42], two new cryptography instructions are proposed for the RISC-V ISA. The instructions accelerate the AES and SM4 block ciphers. Using the AES instructions, one round of the AES can be implemented using only 16 instructions instead of 80. The instructions lead to a similar reduction for the SM4 algorithm. Also, an efficient way to implement these instructions in hardware is proposed in another paper [32].

An ISE (Instruction Set Extension) [33] for RISC-V is proposed to accelerate a steam-cipher called ChaCha [12]. Compared to OpenSSL baseline and ISA-based optimized implementations, ISE-assisted ChaCha speeds up at least 5.4X and 3.4X, respectively.

The core building block of ASCON is implemented as an instruction extension for RISC-V ISA [44]. Comparing the results with efficient C implementations, the accelerator sped up the implementations by about a factor of 50 for ASCON and 80 for ASCON-HASH. Also, the extensions lead to significant binary size reduction.

In [23], researchers proposed a set of hardware accelerators and 29 new instructions for lattice-based cryptography. They implemented their RISC-V-based processor with the proposed accelerators and instructions in ASIC and FPGA. Compared to pure software implementations, accelerators and the instructions lead to a speed-up factor of 11.4 for NewHope, 9.6 for Kyber, and 2.7 for Saber. On the other hand, the cell count of the CPU is increased by a factor of 1.6 compared to the original RISC-V design due to additional instructions and accelerators.

Energy-efficient crypto-coprocessor is designed for the AES, ECC, and SHA-256 algorithms and integrated with an open-sourced RISC-V core [46]. They proposed a conditionally charged flip-flop and used it to implement the crypto-coprocessor. The pipelined design achieved 10.3% power reduction on average running cryptography tasks.

Optimized RISC-V assembly implementation of table-based AES, bitsliced-AES, ChaCha, and Keccak-f[1600] algorithms [45]. The study shows how a 32-bit RISC-V processor can efficiently implement these algorithms, arbitrary-precision addition, and multiplication.

The RISC-V cryptography extensions task group published RISC-V Cryptography Extensions Volume 1 Scalar & Entropy source Instructions Version v1.0.0-RC6 [9]. The extension provides details about the cryptography instructions; however, it does not analyze software implementations using these instructions. Although an increase in software implementation efficiency using the extension is expected, the possible gain is not provided. Chapter 3 explains the extension in detail.

1.3 Aim and Motivation

The ever-increasing need for securing computing systems using cryptographic algorithms is spurring interest in efficient implementation of common algorithms. While the algorithms can be implemented in software using the base instruction set of processors, there is considerable potential to reduce memory cost and improve speed using specialized instructions and associated hardware. However, there is a need to assess the relative benefits and costs of software implementations and new instructions that implement key cryptographic algorithms in fewer cycles. The primary aim of this thesis is to improve understanding of the cost of implementing cryptographic algorithms for the RISC-V instruction set architecture (ISA) for both implementations of the algorithms in software using the base RV32I instruction set and with implementation of instructions as hardware in additional functional units.

1.4 Objectives

The project objectives are listed below:

- Implement cryptographic algorithms in RISC-V assembly using 32-bit base RISC-V instructions (RV32I).
- Implement cryptographic algorithms in RISC-V assembly using cryptography ISE in addition to base instructions (RV32I + crypto).
- Extract total instruction count, program memory, and static memory requirements of the RISC-V assembly implementations.
- Implement a 32-bit RISC-V processor and the ISE in hardware.
- Extract area requirements of the hardware implementations.

- Visualize the results using graphs.

- Analyze the assembly implementations to find the most time-consuming operations in the algorithms.

- Propose a new instruction to increase the software implementation efficiency of the cryptographic algorithms, analyze the algorithms with the proposed instruction.

1.5 Methods

Firstly, eleven standard cryptographic algorithms are chosen to be implemented. The algorithms include eight ISO standard symmetric-key block ciphers and three standard hash functions. Five of the block ciphers (TDEA [11], MISTY1 [34], CAST-128 [8], HIGHT [27], PRESENT [15]) have 64-bit block size and the remaining three (AES-128 , CAMELLIA [10], SEED [31]) have 128-bit block size. PRESENT is the only lightweight cipher among these eight block ciphers, specified in ISO/IEC 29192-2:2019 standard. The other seven ciphers are specified in the ISO/IEC 18033-3:2010 standard. The hash algorithms are SHA-256 [36], SHA-512 [36], and SHA3-256 [22]. The SHA3-256 is specified in the NIST FIPS-202 [21] and other two hash functions are specified in NIST FIPS-180 [18].

Secondly, MATLAB programming language is chosen to implement a simple assembly language simulator for RISC-V. The MATLAB implementation allowed us to easily modify the simulator to display the required outputs (total instruction count, program memory, static memory). Also, we built a simple assembler in Python programming language to convert RISC-V assembly language to machine code.

Thirdly, a 32-bit RISC-V processor and cryptography instruction modules are implemented using Verilog HDL (Hardware Description Language). The design is simulated on Xilinx's Vivado Design Suite. After that, an open-source framework for Verilog RTL synthesis called Yosys [48] is used to extract the hardware area requirement.

To propose new instruction, each software implementation is analyzed to list the time-consuming operations. Based on the analysis, a new instruction is proposed. Some of the algorithms are also implemented and analyzed with the proposed instruction.

1.6 Contributions

This thesis analyzed the software implementation efficiency of the eleven cryptography algorithms on RISC-V ISA. The eleven cryptographic algorithms are implemented using 32-bit RISC-V assembly language. The algorithms are first implemented using only the 32-bit base instruction set and then the 32-bit scalar cryptography instruction set in addition to the base instruction set. The assembly implementations are analyzed to extract total instruction count, program memory, and static memory requirements. A 32-bit RISC-V processor with area-optimized cryptography extension modules is designed to understand the hardware area requirement of extending the instruction set with the cryptography instructions. In the end, assembly implementation results and the gate equivalent area of the hardware implementations are combined for a comprehensive analysis. Based on the analysis results, a new instruction is proposed and analyzed to increase the software implementation efficiency of the cryptographic algorithms.

Chapter 2 explains the RISC-V ISA. Chapter 3 explains the 32-bit scalar instructions included in the RISC-V Cryptography Extensions Volume I Scalar & Entropy Source Instructions. Chapter 4 describes the 11 cryptographic algorithms and provides information about the assembly implementations. Chapter 5 provides the hardware architecture of the processor and the cryptography extension modules. Chapter 6 provides the results and proposes an instruction. Chapter 7 concludes the thesis.

Chapter 2

RISC-V INSTRUCTION SET ARCHITECTURE

RISC-V ISA [3] [4] provides both 32-bit and 64-bit load-store architectures. The open-source architecture RISC-V allows the implementation of various instruction sets. Among these are already ratified proposals, such as:

- I - Base Integer Instruction Set
- M - Standard Extension for Integer Multiplication and Division
- A - Standard Extension for Atomic Instructions
- F - Standard Extension for Single-Precision Floating-Point
- D - Standard Extension for Double-Precision Floating-Point
- Q - Standard Extension for Quad-Precision Floating-Point
- C - Standard Extension for Compressed Instructions
- Zifencei - Instruction-Fetch Fence
- Zicsr - Control and Status Register instructions

as well as proposals still under development like the cryptography extensions – both scalar and vector. Only the base integer instruction set is mandatory among all these instruction sets, and others can be added in any combination desired based on need. The ISA supports three main privilege modes, namely Machine (M-mode), User (U-mode), and Supervisor (S-mode) modes. The Machine-mode is the only