XCrypto: A General Purpose Cryptographic ISE for RISC-V

Ben Marshall, Daniel Page, Thinh Pham
first.last@bristol.ac.uk

github.com/scarv/xcrypto

Introduction

Cryptographic workloads are expected to satisfy a range of traditional design metrics, including high-throughput, low-latency, low-footprint, power-efficiency, and high-assurance, all while executing in a potentially adversarial environment. A large design space of options must be considered when building a concrete implementation: these options range between dedicated hardware and those entirely based on software. ISEs represent a hybrid option: they alter a general-purpose processor core with special-purpose hardware and associated instructions; aiming to improve a software-based design metric like latency.

As an ISE, *XCrypto is a solution* (vs. the solution) within the wider design space of options; it offers as an alternative to the proposal by the RISC-V cryptography extensions group, which extends the RISC-V vector ISE. The idea is to leverage extensive existing literature and hence experience wrt. cryptographic ISEs (see, e.g., published work at the CHES conference), translating and applying it to RISC-V. Although potentially less performant than alternatives, we expect implementations using XCrypto to be more lightweight and flexible; as a result, we view it as representing an attractive solution in the context of micro-controller class cores.

Implementation Progress

Tool support:

- Modified toolchain to support instruction assembly.
- XCrypto support added to the Spike ISA Simulator.

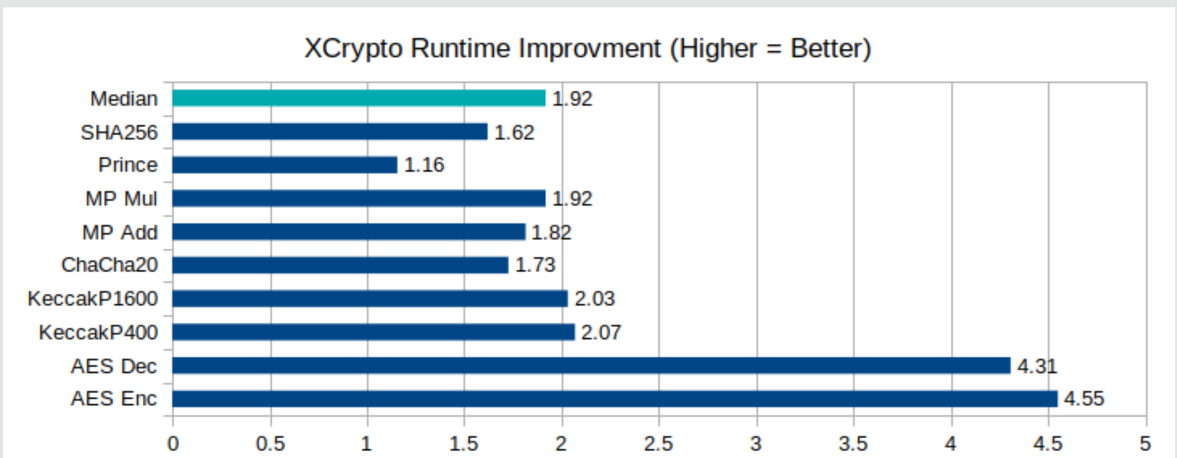
Benchmarks (so far):

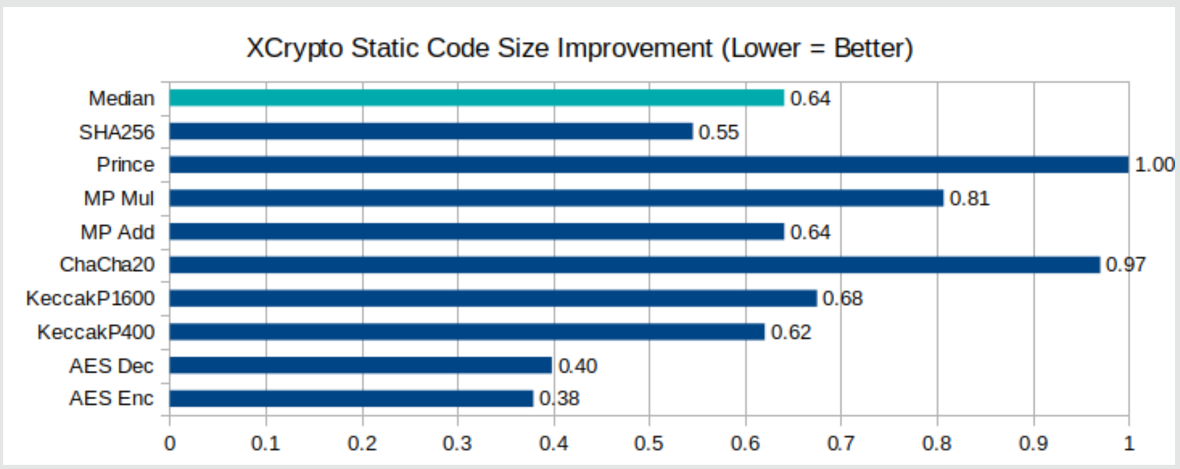
- Block Ciphers: AES, Prince, ChaCha20
- Hash Functions: SHA2, KeccakP400, KeccakP1600
- Multi-precision arithmetic & modular exponentiation.

Formally verified reference hardware implementation:

- Implemented as a re-usable co-processor on a Xilinx FPGA.
- Example area-optimised integration with the PicoRV32 core.
- CPU + XCrypto: 6.4K LUTs, 1.1K DFFs for `RV32IMCX`
- XCrypto Only: 1.9K LUTs, 215 DFFs.

Early Results





ISE Specification

Extra State: XCrypto adds a *16*32-bit register file* for handling cryptographic data. This enables *minimally invasive* implementations of the ISE and flexibility wrt. *side channel countermeasures*.
ISE Organisation: Like RISC-V, XCrypto contains a *base set* of instructions and multiple *optional extensions*. Implementations can include only the functionality they need.
Class 1: Base Instructions: The bare minimum of instructions needed to move data in/out of the register file. Load/Store instructions allow partial register accesses.

```
xc.ld.hu c0,(0), 0(a0)
xc.ld.hu c0,(1), 6(a0)
.
.
```

```
lhu a2, 6(a0)
lhu a1, 0(a0)
slli a2, a2, 16
or a1, a1, a2
```

Class 2.1: Random number source: High quality randomness is essential for cryptography and side-channel countermeasures. XCrypto defines an interface to a randomness source, which can be seeded, sampled and checked for quality.
Class 2.2: Scatter/Gather: Efficient SBOXs are crucial for block ciphers. Byte and halfword scatter/gather instructions are a code-dense and energy efficient way to achieve this.

```
.L0: xc.ld.w c0,0(a0)
     xc.gather.h c0,c0,a1
     xc.st.w c0,0(a0)
     addi a0, a0, 4
     bltu a0, a3, .L0
.
```

```
.L0: lhu t0, 0(a0)
     add t1, t0, a1
     lhu t0, 0(t1)
     sh t0, 0(a0)
     addi a0, a0, 2
     bltu a0, a3, .L0
```

Class 2.3: Bitwise Operations: Bitwise manipulation of data is essential to Cryptography. We include bitfield insert/extract instructions, and multi-operand bitwise operation instructions. The `xc.bop` instruction can implement any 3-variable bitwise function.

```
xc.bop c0, c1, c2, 0b11101000 ; c0 = c0 | (c1 & c2)
```

Class 2.4: Packed Arithmetic: Cryptographic implementations often exhibit sub-word sized data parallelism. We add SWAR functionality to the traditional set of arithmetic instructions; operating on bit widths of: 2, 4, 8, 16 and 32.

```
xc.padd b, c2, c0, c1 ; c2 = sum of corresponding bytes in c0, c1
xc.padd w, c3, c0, c1 ; c3 = sum of words c0, c1
```

Class 2.5: Multi-precision arithmetic: Public key cryptography depends on efficient multi-word arithmetic and efficient carry bit handling.

```
xc.ldr.w c2, t0, a0
xc.ldr.w c3, t0, a1
xc.madd.3 (c1,c0), c2, c3, c1
xc.str.w c0, t0, a2
.
.
.
```

```
lw a5, 0(a1)
add a4, a5, a4
sltu a6, a4, a5
lw a5, 0(a3)
add a5, a4, a5
sltu a4, a5, a4
sw a5, 0(a0)
```

Class 3.1: AES: Given the importance of AES, we added dedicated light-weight instructions to accelerate the *SubBytes* and *MixColumns* operations.
Class 3.2 SHA3: Code-dense implementations of SHA3 suffer when generating indexes into the state matrix. We accelerate this with dedicated instructions for index generation, usable across all SHA3 parameter sets.

University of BRISTOL

EPSRC
Engineering and Physical Sciences Research Council

This work has been supported in part by EPSRC via grant EP/R012288/1, under the RISE (`http://www.ukrise.org`) programme.

SCARV