61 THE CHIPS TO SYSTEMS CONFERENCE

SHAPING THE NEXT GENERATION OF ELECTRONICS

JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

THE CHIPS TO SYSTEMS CONFERENCE
SHAPING THE NEXT GENERATION OF ELECTRONICS
JUNE 23-27, 2024
MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

RISC-V Instruction Set Extensions for Multi-Precision Integer Arithmetic

A Case Study on Post-Quantum Key Exchange Using CSIDH-512

Hao Cheng[1], Georgios Fotiadis[1], Johann Großschädl[1], Daniel Page[2], Thinh Pham[2], Peter Y. A. Ryan[1]

[1]University of Luxembourg

[2]University of Bristol

Multi-Precision Integer (MPI) arithmetic

- Many PK cryptosystems operate on MPI (hundreds/thousands bits)
  - Classical: RSA, ECC
  - Post-quantum: isogeny-based (e.g., CSIDH)
  - Modular operations with MPI at the lowest level
  - Performance-critical component
- MPI representation
  - n-bit integer by w-bit digits/limbs, length $l = \lceil n/w \rceil$
  - Full-radix: $w = machine\ word\ size$, digit
  - Reduced-radix (redundant): $w < machine\ word\ size$, limb

RISC-V ISE for cryptography

- RISC-V
  - Open Instruction Set Architecture (ISA) + open-source licenses
  - No ALU status bits or flags (no carry flag!)
  - Modular ISA design + optional Instruction Set Extensions (ISEs)
- Standard crypto (K) extension
  - General-purpose instructions for, e.g., permutations and rotations
  - Special-purpose instructions for some symmetric crypto algorithms, e.g., AES, SHA-2
  - No instructions for MPI arithmetic
- Custom extensions for crypto
  - No paper proposes an ISE approach for scalable MPI arithmetic

Contributions

- ISA-only implementations of MPI arithmetic on 64-bit RISC-V
- ISE designs for scalable MPI arithmetic
- 4 different implementations of CSIDH-512
  - Representation
    - Full-radix: 64-bit-per-digit
    - Reduced-radix: 57-bit-per-limb ($\lceil 511/9 \rceil = 57$)
  - Implementation type
    - ISA-only (pure software): only RV64GC
    - ISE-supported (HW+SW hybrid): RV64GC + custom instructions
  - Focus on prime-field $\mathbb{F}_p$ arithmetic
    - Mainly $\mathbb{F}_p$-multiplication (most performance-critical operation)
    - Constant-time assembly implementations

CSIDH key exchange

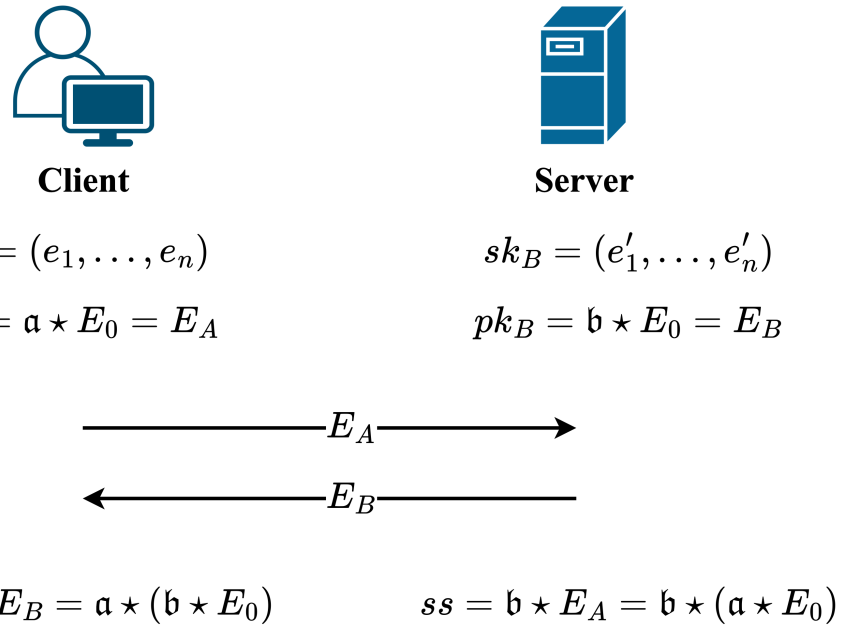
- CSIDH
  - Action of ideal class group on supersingular ECs

$$\star: \mathrm{Cl}(\mathbb{Z}[\sqrt{-p}]) \times S \to S$$

  - Action of ideal $\mathfrak{a} = \mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}$ on E_A: $\mathfrak{a} \star E_A$
  - Compute isogeny ϕ of degree $\ell_1^{e_1} \cdots \ell_n^{e_n}$
  - Supersingular curves in Montgomery form

$$E_A/\mathbb{F}_p: y^2 = x^3 + Ax^2 + x$$

  - Special prime $p = 4 \cdot \ell_1 \cdots \ell_n - 1$
- Case study: CSIDH-512 (NIST security level 1)
  - Prime p 511 bits long



ISA-only: Montgomery multiplication

- High-level techniques
  - Multiplication: operand-scanning, product-scanning, Karatsuba, etc.
  - Integration: separated, coarsely-integrated, finely-integrated
- Low-level optimizations
  - Main building block is MAC: $S \leftarrow S + a_i \cdot b_j$
  - Full-radix: 8 instr
  - Reduced-radix: 6 instr + implicit overhead
    - More MACs since limb-number > digit-number
    - Alignment of the accumulator S

Listing 1: ISA-only full-radix MAC operation.

```
/* Input/Output: 192-bit accumulator  e || h || l */
/* Input:         64-bit operands     a and b     */
mulhu z, a, b; mul y, a, b; add  l, l, y; sltu y, l, y;
add   z, z, y; add h, h, z; sltu z, h, z; add  e, e, z;
```

Listing 2: ISA-only reduced-radix MAC operation.

```
/* Input/Output: 128-bit accumulator  h || l  */
/* Input:         64-bit operands     a and b */
mulhu z, a, b; mul y, a, b; add  l, l, y; sltu y, l, y;
add   z, z, y; add h, h, z;
```

ISE-supported: Design of ISE

- Overview of ISE
  - 2 integer multiply-add instructions + 1 instruction to assist carry propagation
  - Full-radix: maddlu, maddhu, cadd
  - Reduced-radix: madd57lu, madd57hu, sraiadd
  - Execution latency: single clock cycle
- Design guideline
  - Use general-purpose scalar register file to store operands
  - No special-purpose (micro-)architectural state (e.g., cache, scratch-pad)
  - Use R4-type for only MAC to save the encoding space

ISE-supported: Design of ISE (full-radix)

- Classic integer multiply-add designs
  - e.g., ARM (mla, umlal, umaal), Intel AVX-512IFMA

$$rd \leftarrow \left(\left((rs1 * rs2) \gg j\right) \,\&\, m\right) + rs3$$

- Derived design (maddhu) : $rd \leftarrow \left(\left((rs1 * rs2) \gg 64\right) \,\&\, (2^{64} - 1)\right) + rs3$
  - Need to propagate the carry-bit generated by maddlu
- Operations
  - maddlu : $rd \leftarrow \quad (rs1 * rs2 + rs3) \qquad\quad \&\, (2^{64} - 1)$
  - maddhu: $rd \leftarrow \left((rs1 * rs2 + rs3) \gg 64\right) \,\&\, (2^{64} - 1)$
  - cadd : $rd \leftarrow \left((rs1 + rs2) \qquad\quad \gg 64\right) + rs3$

ISE-supported: Design of ISE (reduced-radix)

- Operations
  - madd57lu : $rd \leftarrow \left((rs1 * rs2) \quad \& (2^{57} - 1) \right) + rs3$
  - madd57hu: $rd \leftarrow \left(\left((rs1 * rs2) \gg 57 \right) \& (2^{64} - 1) \right) + rs3$
  - sraiadd : $rd \leftarrow rs1 + \mathrm{EXTS}(rs2 \gg imm)$
- Solved multiplier saturation problem
  - Exists on AVX-512IFMA when $rs1$ and $rs2$ are not canonical (i.e., limbs > 52 bits)
  - Intel AVX-512IFMA has 64-bit lanes but 52-bit multipliers

ISE-supported: Impact of ISE

- Full-radix
  - MAC from 8 instr to now 4 instr
- Reduced-radix
  - MAC from 6 instr to now 2 instr
  - MAC accumulator automatically aligned
  - Expect a higher performance gain from ISE
  - sraiadd saves 1 instr for limb-level carry propagation in other operations

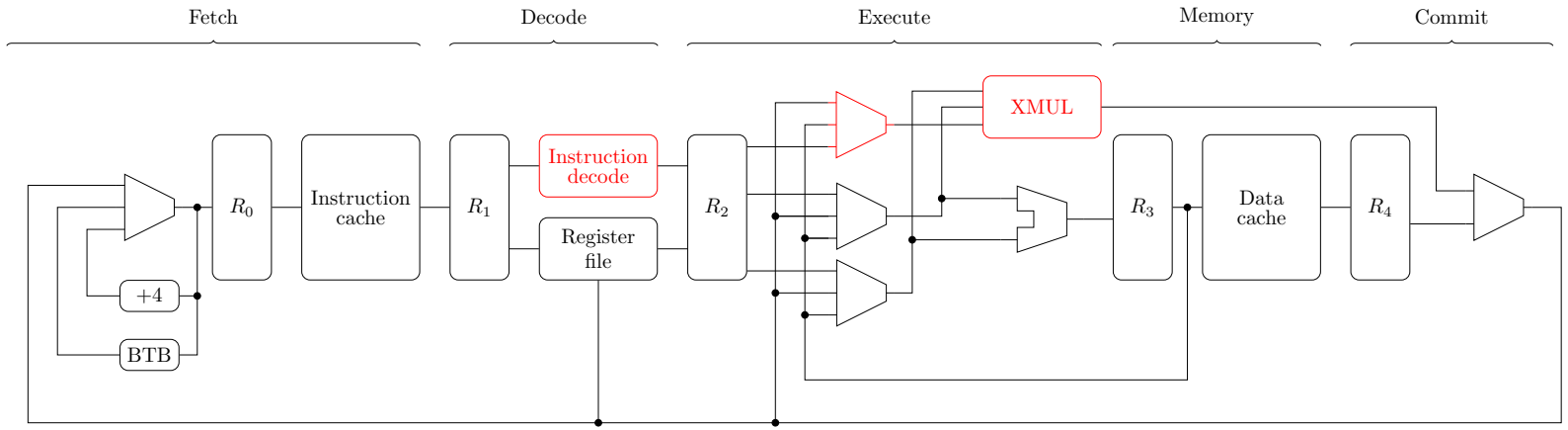
Listing 3: ISE-supported full-radix MAC operation.

```
/* Input/Output: 192-bit accumulator  e || h || l */
/* Input:         64-bit operands     a and b     */
maddhu z, a, b, l; maddlu l, a, b, l;
cadd   e, h, z, e; add    h, h, z;
```

Listing 4: ISE-supported reduced-radix MAC operation.

```
/* Input/Output:  64-bit accumulators h and l */
/* Input:         64-bit operands     a and b */
madd57hu h, a, b, h; madd57lu l, a, b, l;
```

ISE-supported: HW implementation of ISE



- RV64GC Rocket core (on Xilinx Artix-7 XC7A100TCSG324 FPGA)
- Modification of instruction decoder
- Extended multiplier (XMUL) extends the original pipelined multiplier
  - Support 3rd input operand
  - Implementation of custom instructions

Evaluation

- Hardware: area overhead
  - Both full/reduced-radix ISEs about 10%
- Software: cycle count
  - ISA-only
    - Full-radix faster
  - ISE-supported
    - $\mathbb{F}_p$ speed-up propagates well to group action
    - Reduced-radix more suitable
    - 1.71× speed-up compared to ISA-only baseline

Components	LUTs	Regs	DSPs	CMOS
Base core	4807	2156	16	428680
Base core + ISE (full-radix)	5019	2390	16	483248
Base core + ISE (reduced-radix)	5223	2352	16	495290

Operation	Full-radix		Reduced-radix	
	ISA-only	ISE-sup.	ISA-only	ISE-sup.
Integer multiplication	608	371	625	303
Integer squaring	440	371	398	216
Montgomery reduction	730	469	818	389
Fast modulo-p reduction	107	107	112	104
$\mathbb{F}_p$-addition	163	163	148	132
$\mathbb{F}_p$-subtraction	143	143	139	123
$\mathbb{F}_p$-multiplication	1446	954	1561	799
$\mathbb{F}_p$-squaring	1279	951	1334	712
CSIDH group action	701.0 M	502.9 M	736.2 M	411.1 M
	1.00×	1.39×	0.95×	1.71×

Concluding remarks

- A case study with 511-bit operands on RV64
  - Full-radix faster for ISA-only, even no carry flag
  - Reduced-radix more suitable for ISE-supported
  - Speed-up factor of 1.71
  - CSIDH-512 still extremely costly
- Almost all previous MPI ISEs were for full-radix, could also look at reduced-radix
- Different results if different operand-lengths/base-ISAs/microarchitectures
- Future work: support for flexible reduced-radix

Thanks for your attention!