RISC-V Crypto Extension

Lucas(Zewen Ye)

lucas.zw.ye@outlook.com

1/29/2021

Cryptographic Extensions Task Group

Chair: Richard Newell, Microchip, Vice-chair: Derek Atkins, SecureRF

- Charter:
  - propose ISA extensions to the vector extensions for the standardized and secure execution of popular cryptography algorithms.
  - To ensure that processor implementers are able to support a wide range of performance and security levels the committee will create a base and an extended specification.
  - The base will be comprised of low-cost instructions that are useful for the acceleration of common algorithms.
  - The extended specification will include greater functionality, reserve encodings for more algorithms and will facilitate improved security of execution and higher performance.
  - The scope will include symmetric and asymmetric cryptographic algorithms and related primitives such as message digests. The committee will also make ISA proposals regarding the use of random bits and secure key management.
- Approach based on vector extensions
- AES instructions
  - 128, 192, 256; done
- SHA-2 instructions
  - SHA-256 and SHA-512; almost done
- Need to convert AES and SHA-2 into formal specs now…
- Prototyping Public Key Crypto algorithms
  - Long integer arithmetic
  - Implementation proof of concept
- Future directions:
  - More light-weight approach: could recommend subset of vector extensions only
  - XCrypto (Bristol): proposed scalar instructions, rotates, etc. to have SW run faster
  - Paris Telecom also interested in same type of research

https://ches.iacr.org/2019/src/slides/Day1/Session2b_InvitedSpeakerHelenaHandshuh_onlypdfs/Helena.pdf

Feature

- general purpose X registers, and obey the **2-read-1-write** register access constraint
- designed to be lightweight, and be suitable for **32 and 64** bit base architectures, from embedded, IoT class cores to large, application class cores

Design Policy

- Supporting diverse implementation strategies for an algorithm
- Supporting a single implementation style which is more performant / less expensive;
- Well support existing standardised cryptographic constructs.
- Regarding side-channel countermeasures: Where relevant, proposed instructions must aim to remove the possibility of any timing side-channels.

Scalar Extension

- Shared **Bitmanip Extension Functionality**

rotations

RISC-V Bitmanip/Crypto ISA

RV32, RV64:		RV64 only:	
ror	rd, rs1, rs2	rorw	rd, rs1, rs2
rol	rd, rs1, rs2	rolw	rd, rs1, rs2
rori	rd, rs1, imm	roriw	rd, rs1, imm

byte-order swap, bitwise reversal, word-order swap

RISC-V Bitmanip/Crypto ISA

RV32, RV64:		RV64 only:	
grev	rd, rs1, rs2	grevw	rd, rs1, rs2
grevi	rd, rs1, imm	greviw	rd, rs1, imm

Scalar Extension

- Shared **Bitmanip Extension Functionality**

Carry-less multiplication

RISC-V Bitmanip/Crypto ISA

RV32, RV64:	RV64 only:
clmul rd, rs1, rs2	clmulw rd, rs1, rs2
clmulh rd, rs1, rs2	clmulhw rd, rs1, rs2
clmulr rd, rs1, rs2	clmulrw rd, rs1, rs2

Logic with negate

RISC-V Bitmanip/Crypto ISA

RV32, RV64:

```
andn rd, rs1, rs2
 orn rd, rs1, rs2
xorn rd, rs1, rs2
```

Scalar Extension

- Shared **Bitmanip Extension Functionality**

Packing(for 16bit data)

RISC-V Bitmanip/Crypto ISA

```
RV32, RV64:                          RV64:
    pack   rd, rs1, rs2                  packw  rd, rs1, rs2
    packu  rd, rs1, rs2                  packuw rd, rs1, rs2
    packh  rd, rs1, rs2
```

Crossbar Permutation Instructions(for 4-8bit data)

RISC-V Bitmanip/Crypto ISA

```
RV32, RV64:
    xperm.n rd, rs1, rs2
    xperm.b rd, rs1, rs2
```

Scalar Extension

- Scalar AES Acceleration

RISC-V Crypto ISA

```
aes32esi     rd, rs1, rs2, bs // Encrypt: SubBytes
aes32esmi    rd, rs1, rs2, bs // Encrypt: SubBytes & MixColumns
aes32dsi     rd, rs1, rs2, bs // Decrypt: SubBytes
aes32dsmi    rd, rs1, rs2, bs // Decrypt: SubBytes & MixColumns
```

RISC-V Crypto ISA

```
aes64ks1i  rd, rs1, rcon // KeySchedule: SubBytes, Rotate, Round Const
aes64ks2   rd, rs1, rs2  // KeySchedule: XOR summation
aes64im    rd, rs1       // KeySchedule: InvMixColumns for Decrypt
aes64esm   rd, rs1, rs2  // Round:    ShiftRows,    SubBytes,    MixColumns
aes64es    rd, rs1, rs2  // Round:    ShiftRows,    SubBytes
aes64dsm   rd, rs1, rs2  // Round: InvShiftRows, InvSubBytes, InvMixColumns
aes64ds    rd, rs1, rs2  // Round: InvShiftRows, InvSubBytes
```

Scalar Extension

- Scalar SHA-256 / SHA-512 Acceleration

RISC-V Crypto ISA

```
sha256sig0 rd, rs1
sha256sig1 rd, rs1
sha256sum0 rd, rs1
sha256sum1 rd, rs1
```

RISC-V Crypto ISA

```
RV32:                                   RV64:
  sha512sum0r rd, rs1, rs2                sha512sig0 rd, rs1
  sha512sum1r rd, rs1, rs2                sha512sig1 rd, rs1
  sha512sig0l rd, rs1, rs2                sha512sum0 rd, rs1
  sha512sig0h rd, rs1, rs2                sha512sum1 rd, rs1
  sha512sig1l rd, rs1, rs2
  sha512sig1h rd, rs1, rs2
```

Entropy Source Extension

- generate truly random bits

RISC-V Crypto ISA

```
pollentropy      rd, imm          // Get randomness. imm=0 for baseline operation
```

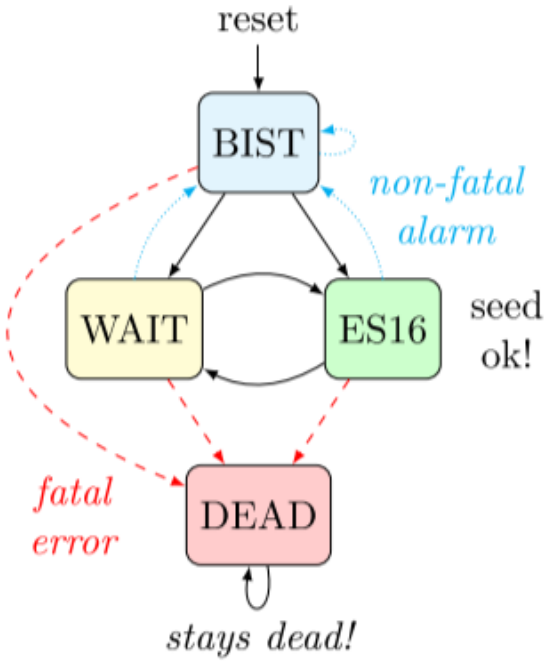

Figure 11: Normally the operational state alternates between WAIT (no data) and ES16, which means that 16 bits of randomness (seed) has been polled. BIST (Built-in Self-Test) only occurs after reset or to signal a non-fatal self-test alarm (if reached after WAIT or ES16). DEAD is an unrecoverable error state.

Thanks