June 11, 2019 @qualcomm Zürich, Switzerland

Qualcomm

Crypto Currently

The state of the Cryptographic Extensions and the challenges we face

Ken Dockser

Sr Director, Technology
Qualcomm Technologies, Inc.

Charter

The Cryptographic Extensions Task Group will 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.

Extending the Vector Extensions

"propose ISA extensions to the vector extensions for the standardized and secure execution of popular cryptography algorithms."

- Same vector register file
- Same Formats
- Same configurations
  - Proposal: Allow ELEN>VLEN
- Additional vector instructions
  - Encryption/Decryption
  - Secure hash
  - Extended arithmetic
    - Polynomial
    - Modular
    - Large number
  - BitManip

Proposal: Extend Vector to allow ELEN > VLEN

Static		Dynamically		Derived			Comment
VLEN (bits)	ELEN (bits)	LMUL	SEW (bits)	VLMAX (elmnts)	2*SEW (bits)	Max # groups (logical regs)	
Moderately large machines (different ELENs), no LMUL							Can vectorize AES with 1 ≤vl ≤ 8
1024	128	1	128	8	N/A[1]	32	Supports AES-128
1024	256	1	128	8	256	32(31)	Supports AES-128, AES-192, AES-256
1024	512	1	256	4	512	32(31)	Supports SHA-256
1024	1024	1	512	2	1024	32(31)	Supports SHA-512
Same moderately large machines using LMUL							Extends vector up to 64 128-bit elements
1024	128	8	128	64	N/A[1]	4	Supports AES-128
1024	256	8	128	64	256	4(3)	Supports AES-128, AES-192, AES-256
1024	256	4	128	32	256	8(7)	Supports more AES modes
1024	512	8	256	32	512	4(3)	Supports SHA-256
1024	1024	8	512	16	1024	4(3)	Supports SHA-512
Mid-size Machines (different ELENs)							Supports all algos; vl up to 8 for AES-128
128	128	8	128	8	N/A[1]	4	Supports AES-128
128	256	8	128	8	256	4(3)	Supports AES-128, AES-192, AES-256
128	256	4	128	4	256	8(7)	Supports more AES modes
128	512	8	256	4	512	4(3)	Supports SHA-256
128	1024	8	512	2	1024	4(3)	Supports SHA-512
Tiny machines (different VLENs and ELENs)							Must use LMUL, even for AES
64	128	8	128	4	N/A[1]	4	Supports AES-128
64	256	8	128	4	256	4(3)	Supports AES-128, AES-192, AES-256
64	256	4	128	2	256	8(7)	Supports more AES modes
64	512	8	256	2	512	4(3)	Supports SHA-256[2]
64	1024	8	512	1	1024	4(3)	Supports SHA-512[2,3]
Very Tiny machines (different VLENs and ELENs)							Must use LMUL, too few bits for SHA-512
32	128	8	128	2	N/A[1]	4	Supports AES-128
32	256	8	128	2	256	4(3)	Supports AES-128, AES-192, AES-256
32	256	4	128	1	256	8(7)	Supports more AES modes
32	512	8	256	1	512	4(3)	Supports SHA-256

Base Extension

"The base will be comprised of low-cost instructions that are useful for the acceleration of common algorithms."

- RISC-type instructions for accelerating a single round
- Similar to what is offered by other leading ISAs
- Pros:
  - Easy to code
  - Easy to optimize
    - Software pipelining
- Cons:
  - Easier to exploit via side-band attacks
  - Requires more instructions to execute all rounds

Base Extension: AES Round-based instructions

- These instructions perform a round of AES encryption or decryption

```
vaese       vData,    vRndKey   # encrypt
vaeselast   vData,    vRndKey   # encrypt last round
vaesd       vData,    vRndKey   # decrypt
vaesdlast   vData,    vRndKey   # decrypt last round
```

.vv and **.vs** variants; maskable; SEW=128, `vrep` is ignored

- Data Input (`vData`) - Vector register with `vl` 128-bit elements
  - Input round: Input message plaintext (to be encrypted) or ciphertext (to be decrypted)
  - Other rounds: Current AES intermediate round state from previous round
- Key Input (`vRndKey`) - Vector with `vl` 128-bit round keys (**.vv**); or with `1` shared round key (**.vs**)
  - Previously computed from the AES Crypto key by key-expansion commands.
    - The round key can be pre-computed and stored or computed on-the-fly
    - Round keys are always 128 bits (AES Crypto key can be 128, 192, or 256 bits)
- Data output (`vData`) - 128-bits, overwrites Data Input (i.e., these commands are destructive)
  - Final round: Resulting final ciphertext (when encrypting) or plaintext (when decrypting)
  - Other rounds: Current AES intermediate round state

Base Extension: SHA-2 family of secure hashes

- Vector instructions for two underlying algorithms (polymorphic):
  - **SHA-256:** Consumes 512 bits of message per 64 rounds (SEW=256)
  - **SHA-512**: Consumes 1024 bits of message per 80 rounds (SEW=512)
- Four additional simple variants supported using above instructions
  - **Based on SHA-256**: SHA-224
  - **Based on SHA-512**: SHA-512/224, SHA-512/256, SHA-384
- 4 vector registers (or groups)
  - 2*SEW Message State - input message in 2*SEW chunks
  - Working State - intermediate state between rounds
  - Hash State - Accumulates final working state after each 60/84 rounds

Extended Extensions

"The extended specification will include greater functionality, reserve encodings for more algorithms, and will facilitate improved security of execution and higher performance."

- Greater Functionality/Reserve Encodings
  - "Rest of the World" provide encodings for "all" algorithms
- Improved Security
  - CISC-type instructions
    - Perform all rounds on a block
  - Pros
    - More amenable to hardening against side channel attacks
    - Smaller code footprint
  - Cons
    - Less control to the coder
    - Harder to schedule interleaved executions

Extended Extensions :AES All-Rounds Instructions

- These instructions perform *all rounds* (10-14) of AES encryption or decryption

```
vaese128    vData,  vKey      # encrypt (all rounds), 128-bit raw AES key (w0-3)
vaese192    vData,  vKey      # encrypt (all rounds), 2*SEW 192-bit raw AES key (w0-5)
vaese256    vData,  vKey      # encrypt (all rounds), 2*SEW 256-bit raw AES key  (w0-7)

vaesd128    vData,  vRndKey   # decrypt (all rounds), Last 128-bit round key (w40-43)
vaesd192    vData,  vRndKey   # decrypt (all rounds), 2*SEW Last two round keys (w44-47, w48-51)
vaesd256    vData,  vRndKey   # decrypt (all rounds), 2*SEW Last two round keys (w52-55, w56-59)
```

SEW = 128

For 192 and 256 the vData input/output are narrower than the 2*SEW key elements

- Destructive – saves opcode space
- Vector-Scalar variant – key shared by all elements
- Key-expansion functionality built in (unlike the single-round instructions)
  - vKey - standard AES key
  - vRndKey: last one or two standard round keys

Scope: Random bits and secure key management

"The committee will also make ISA proposals regarding the use of random bits and secure key management."

Key Management

- Developing a concept to mask all [symmetric] keys stored in the vector registers

True Random Number Generator (TRNG, a.k.a. NRBG)

- Plans to provide instructions to interface to a TRNG along the lines of the NIST SP800-90A API

Open issues, challenges and needs

- Vector Register Lengths supported:
  - Should we support crypto on vector length implementations < 128 or 256?
    - What about VL=32 bits
  - How do we achieve this?
    - One Proposal: Extend Vector to allow ELEN > VLEN
- Performance
  - We need code to test out the instructions
- Algorithms:
  - Which should we support?
  - How do we support?
    - Helper instructions? Round based? All-rounds?
  - Which should we prioritize?
- Encodings for the instructions:
  - Variable length: 48- or 64-bit instructions?
  - State-based semantics?
    - Meaning of an instruction is determined by value in a register (GPR, V-reg, CSR, …)
      - How do we make code secure and readable

We want/need your help

- Join the group: **tech-crypto-ext**
  - Active email discussions
- Join the meetings
  - Small but enthusiastic group
- Join us this Thursday in the Member's Meeting
  - 10-11:00am
  - Room E8
- Provide code using these instructions
- Provide input on what you actually need
  - Looking for practical over theoretical

Qualcomm

Thank you!

Follow us on:

For more information, visit us at:

www.qualcomm.com & www.qualcomm.com/blog