Parallelizable Hashing

Kris Gaj
George Mason University
USA

Software vs. Hardware

- Majority of the discussion (rightfully) driven by software implementations, however, let us not forget about hardware

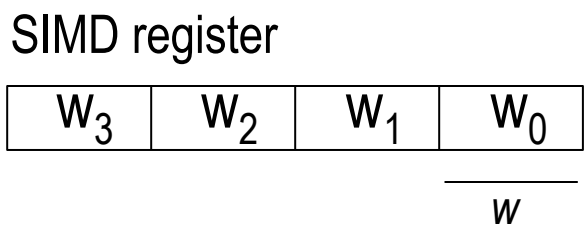
Analogies:

Software:	1) SIMD	2) multicores
Hardware:	1) pipelined architectures	2) multi-unit architectures

Differences:

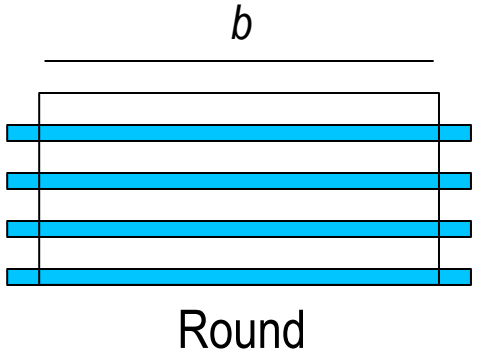
SIMD

- parallelize elementary word operations (ADD, SHIFT, ROTATE, etc.)
- operate on words

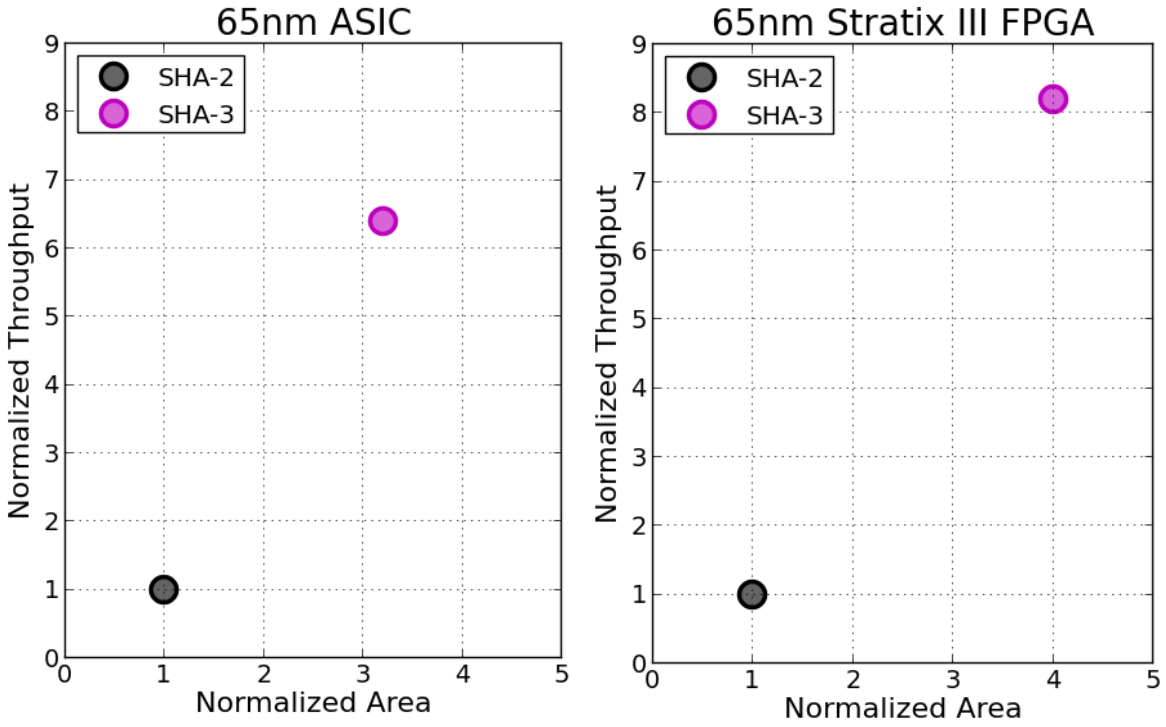


Pipelined architectures

- parallelize stages of the algorithm rounds
- operate on entire message blocks

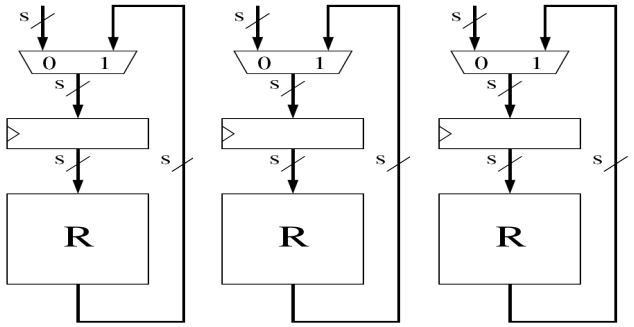


SHA-3 vs. SHA-2 in Hardware

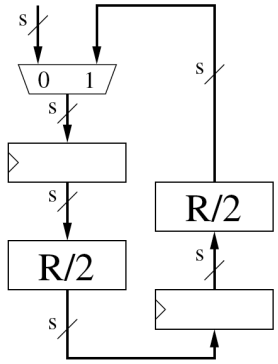


In hardware:

SHA-3: 6-8 times faster
3-4 times bigger



SHA-2 more suitable for multi-unit architecture and thus segmenting.



SHA-3 more suitable for pipelining and thus interleaving.

Suggested Approach

- Common standard suitable for SHA-2 and SHA-3
- Segmenting + interleaving
  - SW: multicores with SIMD
  - HW: multi-unit architecture with pipelined units

with an option of reduction to

- interleaving only (SW: SIMD; HW: pipelined architecture)
- segmenting only (SW: multicore; HW: multi-unit architecture)
- sequential operation (SW: single core, no SIMD;
  HW: sequential architectures)

Supported Parameters

Number of lanes:

SW: determined by the ratio register_size_in_SIMD/word_size_in_SHA

- register_size_in_SIMD = 64, 128, 256, 512
- word_size_in_SHA = 32, 64
- #lanes = 1, 2, 4, 8, 16, 32

HW: determined by the number of pipeline stages

- not limited to powers of 2
- not likely to exceed 16

Number of segments:

SW: determined by the number of available cores

- number of cores in modern microprocessors = 1, 2, 4, 6, 8, 10, or more

HW: determined by the number of units working in parallel

- limited only by the area budget

Size of segments (no need to specify):

SW, HW: minimum size providing close to maximum performance gain

- may need to be determined experimentally

Used Parameters

Single-user applications

(e.g., integrity of hard drives, remote storage):

- off-line benchmarking with parameters supported by implementation
- choosing the best parameters.

Multi-user applications

(e.g., integrity of messages, digital signatures):

- off-line benchmarking with parameters supported by implementation
- ranking of parameter sets
- negotiating a parameter set with other user(s) based on
  - intersection of user sets
  - best compromise *or* best set for the weakest node.

Other Loose Thoughts

- Allowing parallelized SHA-2 in HMAC
- Treating current sequential SHA-3 and SHA-2 as special cases of the parallelized standardized hash (number of lanes = 1, number of segments = 1)
- Dealing with legacy implementations (e.g., those hashing multiple messages in parallel)
- The effect of vectored I/O (scatter/gather I/O)
- Redefining tree hashing to take advantage of parallelized hashing