1001 Ways To Implement Keccak

Guido Bertoni[1] Joan Daemen[1]
Michaël Peeters[2] Gilles Van Assche[1] Ronny Van Keer[1]

[1]STMicroelectronics

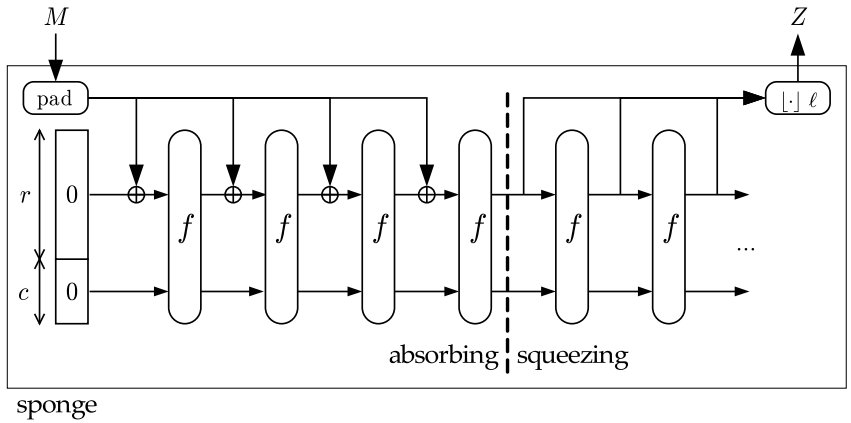
[2]NXP Semiconductors

Third SHA-3 candidate conference, Washington DC
March 22-23, 2012

Outline

Outline

Keccak: the sponge construction



- One permutation for the SHA-3 competition:

  Keccak-*f*[1600]

- Benefits of using a single permutation
  - Saving ROM code size / FPGA slices / ASIC area
  - No 32-bit/64-bit mismatch (see bit interleaving)

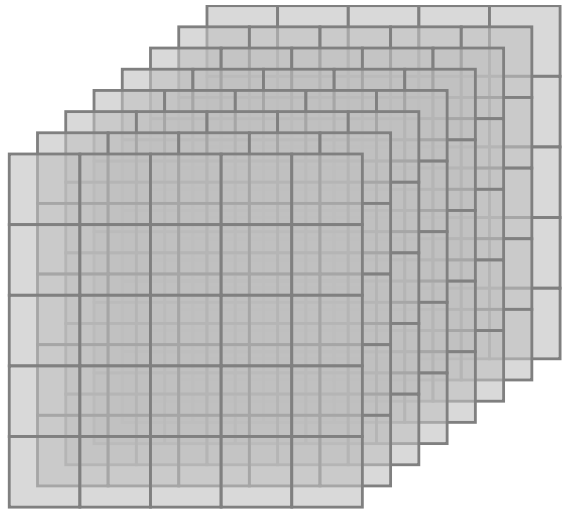
But how to easily report speed vs security?

- We report figures for Keccak$[r = 1024, c = 576]$
- In general, throughput proportional to rate r

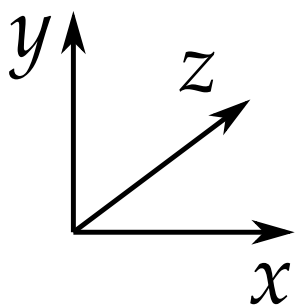
Rate	Capacity	[NIST SP 800-57] Security strength	Relative performance
1376	224	112	×1.343
1344	256	128	×1.312
1216	384	192	×1.188
1088	512	256	×1.063
1024	576	n/a	1.000
576	1024	n/a	÷1.778

The state in Keccak

- Keccak-f operates on 3D state
- Efficient implementations based on state organization and transformations

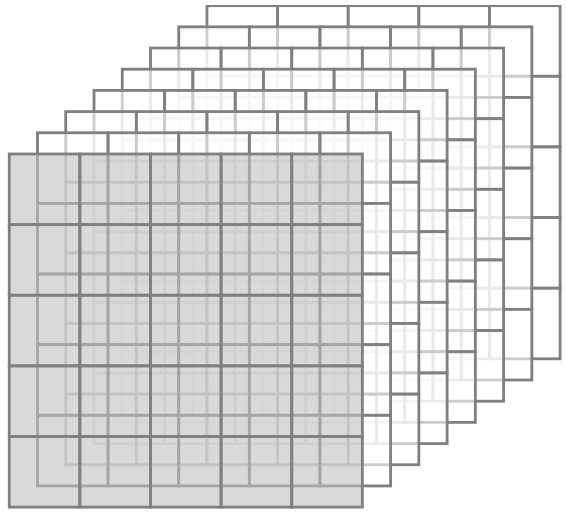


state

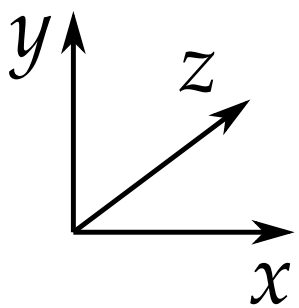


The state in Keccak

- Keccak-f operates on 3D state
- Efficient implementations based on state organization and transformations

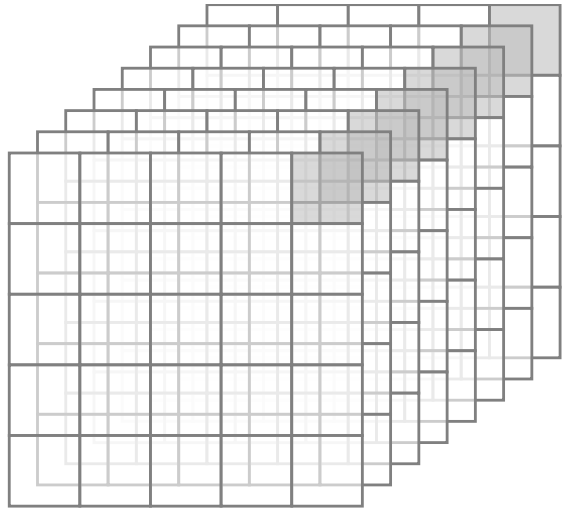


slice

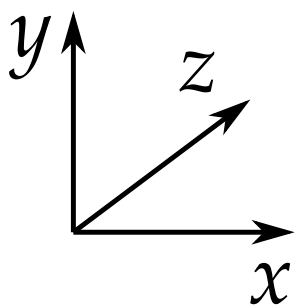


The state in Keccak

- Keccak-f operates on 3D state
- Efficient implementations based on state organization and transformations

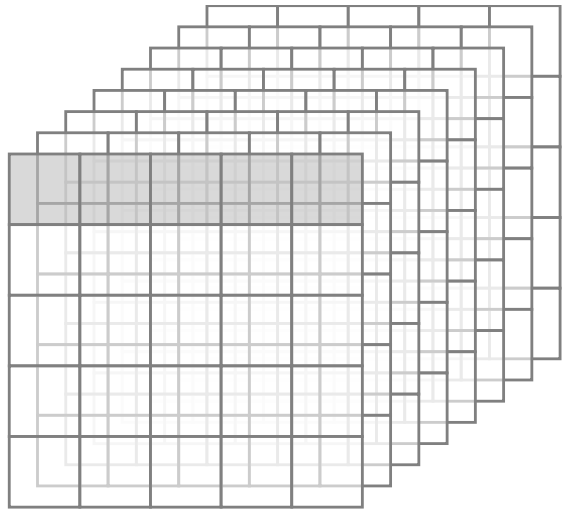


lane

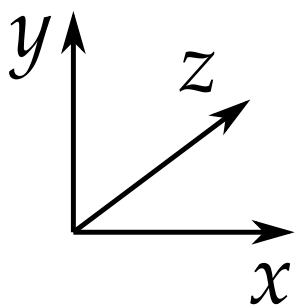


The state in Keccak

- Keccak-f operates on 3D state
- Efficient implementations based on state organization and transformations

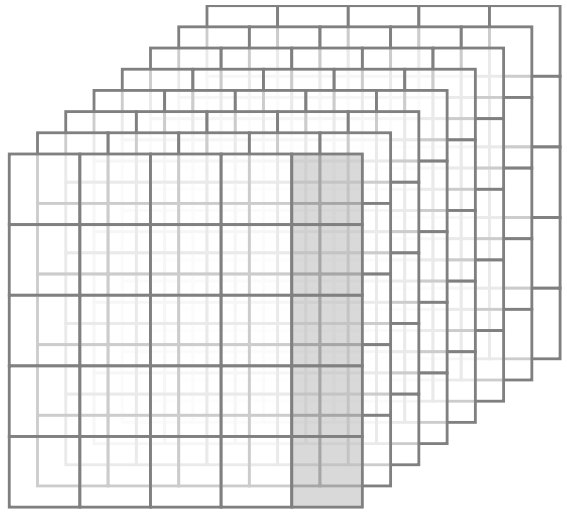


row

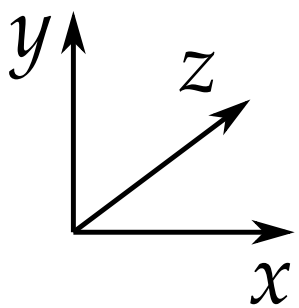


The state in Keccak

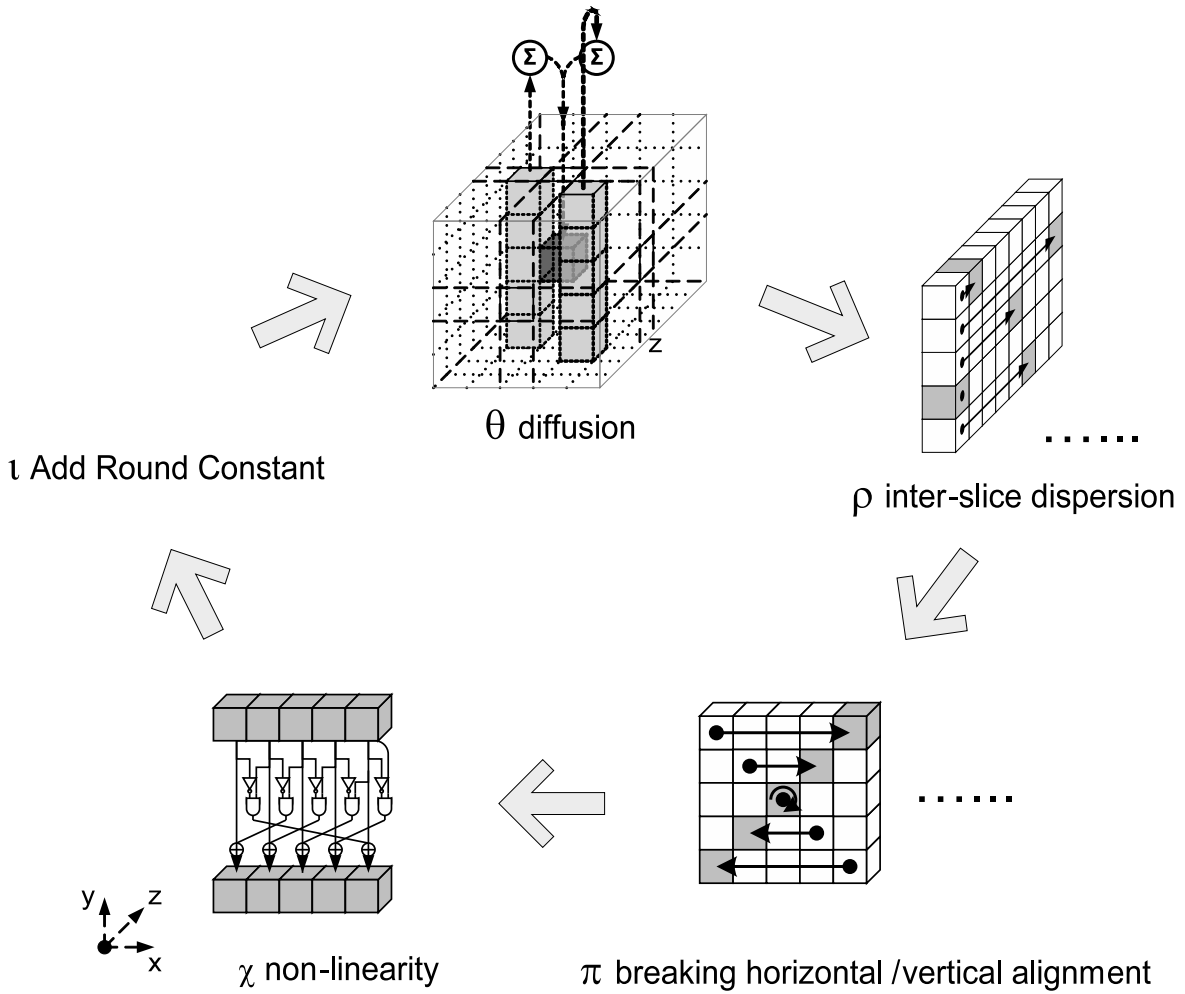
- Keccak-f operates on 3D state
- Efficient implementations based on state organization and transformations



column

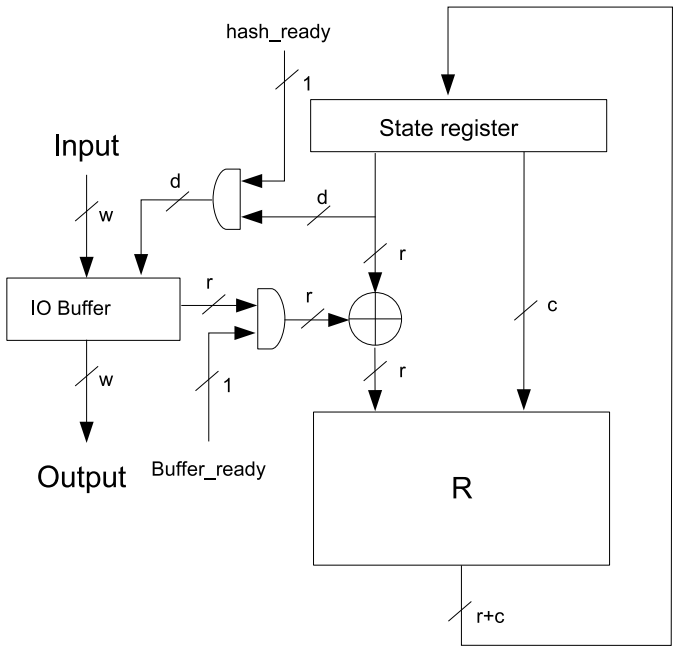


The step mappings of Keccak-*f*



Outline

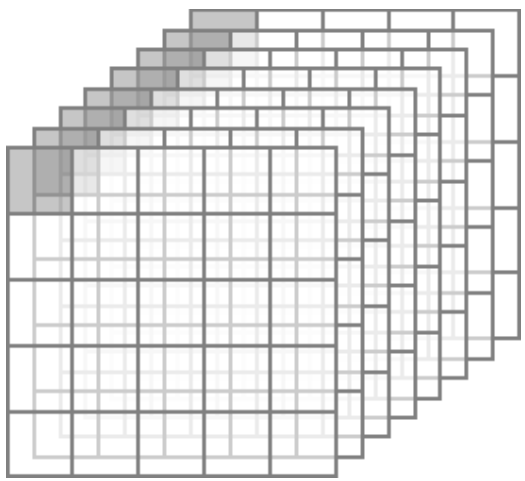
Not cutting it: straightforward hardware architecture



- Logic for one round + register for the state
  - very short critical path $\Rightarrow$ high throughput
- Multiple rounds can be computed in a single clock cycle
  - 2, 3, 4 or 6 rounds in one shot

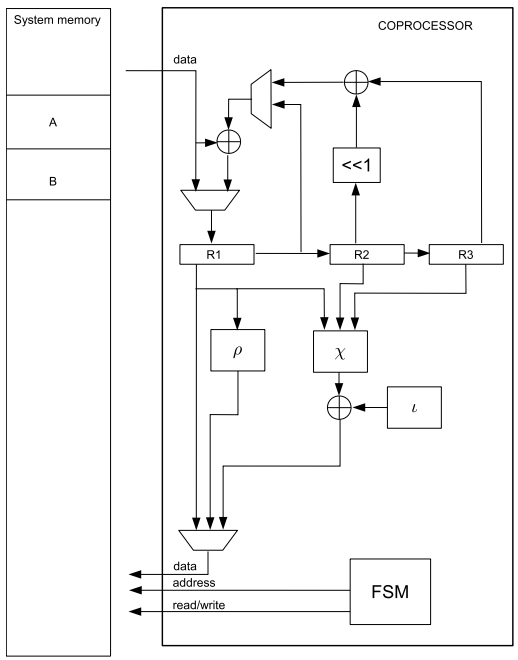
Lanes: straightforward software implementation

- Lanes fit in 64-bit registers
- Very basic operations required:
  - θ XOR and 1-bit rotations
  - ρ rotations
  - π just reading the correct words
  - χ XOR, AND, NOT
  - ι just a XOR



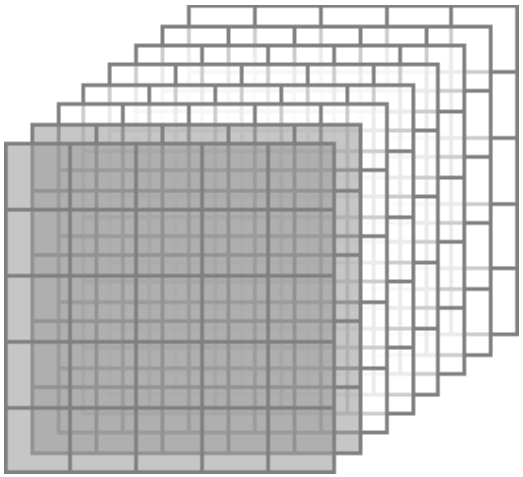
Lane-wise hardware architecture

- Basic processing unit + RAM
- Improvements over our co-processor:
  - 5 registers and barrel rotator
    [Kerckhof et al. CARDIS 2011]
  - 4-stage pipeline, ρ in 2 cycles,
    instruction-based parallel execution
    [San and At, ISJ 2012]
- Permutation latency in clock cycles:
  - From 5160, to 2137, down to 1062



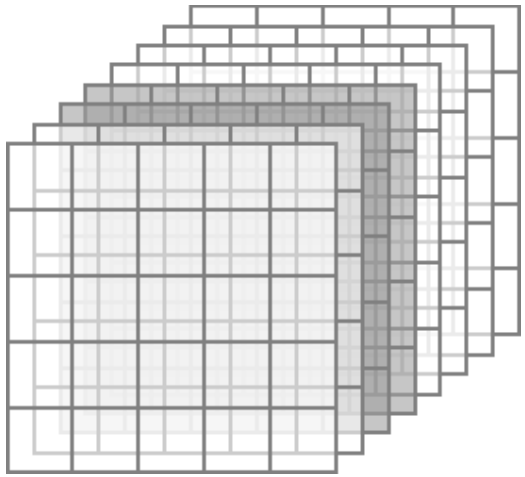
Slice-wise hardware architecture

- Re-schedule the execution
  - χ and θ on blocks of slices
    [Jungk et al, ReConFig 2011]
- Suitable for compact FPGA or ASIC
- Performance-area trade-offs
  - Possible to select number of processed slices from 1 up to 32
    [VHDL on http://keccak.noekeon.org/]



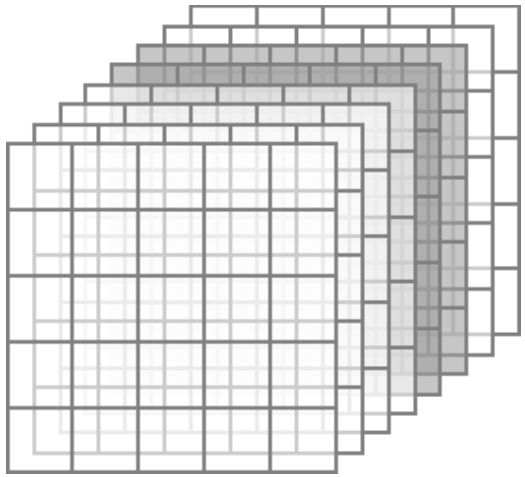
Slice-wise hardware architecture

- Re-schedule the execution
  - χ and θ on blocks of slices
    [Jungk et al, ReConFig 2011]
- Suitable for compact FPGA or ASIC
- Performance-area trade-offs
  - Possible to select number of processed slices from 1 up to 32
    [VHDL on http://keccak.noekeon.org/]



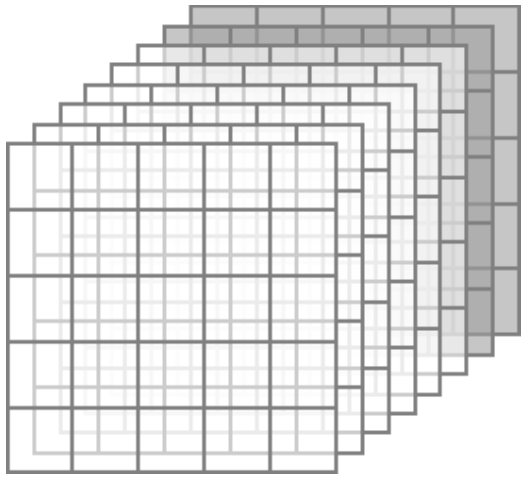
Slice-wise hardware architecture

- Re-schedule the execution
  - χ and θ on blocks of slices
    [Jungk et al, ReConFig 2011]
- Suitable for compact FPGA or ASIC
- Performance-area trade-offs
  - Possible to select number of processed slices from 1 up to 32
    [VHDL on http://keccak.noekeon.org/]



Slice-wise hardware architecture

- Re-schedule the execution
  - χ and θ on blocks of slices
    [Jungk et al, ReConFig 2011]
- Suitable for compact FPGA or ASIC
- Performance-area trade-offs
  - Possible to select number of processed slices from 1 up to 32
    [VHDL on http://keccak.noekeon.org/]



Cutting the state in lanes or in slices?

- Both solutions are efficient, results for Virtex 5

Architecture	T.put Mbit/s	Freq. MHz	Slices (+RAM)	Latency clocks	Efficiency Mbit/s/slice
Lane-wise [1]	52	265	448	5160	0.12
Lane-wise [2]	501	520	151 (+3)	1062	3.32
Slice-wise [3]	813	159	372	200	2.18
High-Speed [4]	12789	305	1384	24	9.2

[1] Keccak Team, Keccak implementation overview

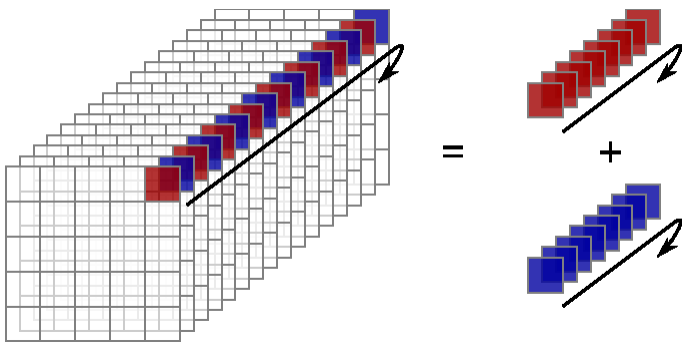
[2] San, At, ISJ 2012

[3] Jungk, Apfelbeck, ReConFig 2011 (scaled to $r = 1024$)

[4] GMU ATHENa (scaled to $r = 1024$)

Bit interleaving

- Ex.: map 64-bit lane to 32-bit words
  - ρ seems the critical step
  - Even bits in one word
    Odd bits in a second word
  - $\mathrm{ROT}_{64} \leftrightarrow 2 \times \mathrm{ROT}_{32}$
- Can be generalized
  - to 16- and 8-bit words
- Can be combined
  - with lane/slice-wise architectures
  - with most other techniques



[Keccak impl. overview, Section 2.1]

Outline

SIMD and tree hashing

- Tree hashing is ...
  - attractive for exploiting multicore availability
  - already interesting on a single core
- Efficient evaluation of $2 \times$ Keccak-f on latest CPUs
  - In eBASH: `keccakc512treed2` using SSE or AVX

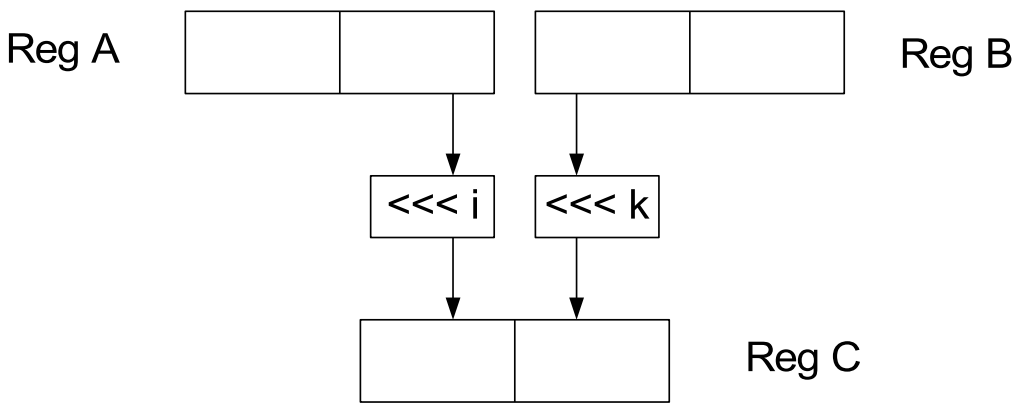
≈ 7 cycle $\cdot$ core/byte on Sandy Bridge [eBASH]

Instruction-level parallelism

- Improving CPUs via parallel execution units
- Degree of parallelism is intrinsic to the algorithm
- Parallelism for Keccak transformations:
  - Up to 25 for χ, ρ and part of θ
  - Minimum is 5 when computing θ-effect
- For instance Itanium 2 versus Intel Core i7:
  - 6.02 cpb vs 11.48 cpb [eBASH]

Dedicated instructions

- Intel, AMD and ARM are adopting dedicated instructions for speeding-up cryptographic algorithms
- Keccak can benefit of simple dedicated instructions:
  - Storing the state in 128/256-bit registers
  - XOR-AND-NOT for χ
  - Rotate 64-bit words and *assign*
    - *Can also benefit to other primitives!*



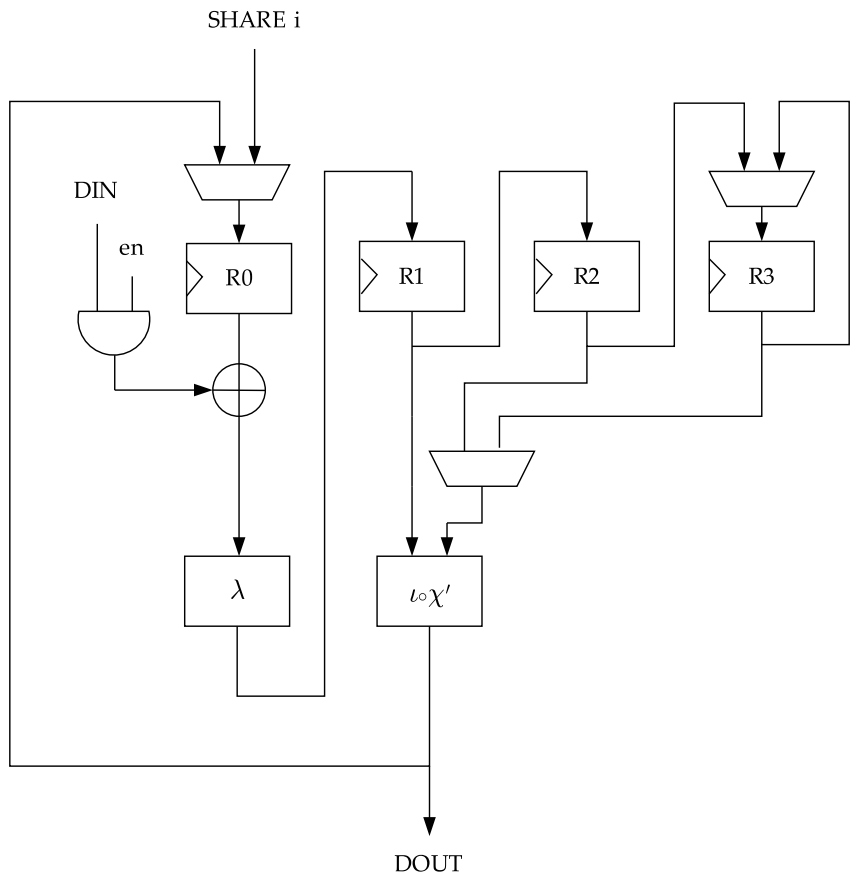
Outline

Secure implementations

Keyed modes may require protected implementations

- Keccak offers protection against
  - timing or cache-miss attacks
    no table look-ups
  - side channels (DPA)
    efficient secret sharing thanks to degree-2 round function

[Keccak impl. overview, Chapter 5]



Outline

Conclusions

- The state can be cut in many ways
  - Lane-wise or slice-wise (e.g., compact hardware)
  - Bit interleaving for low-end CPUs
- Good potential for improvements on high-end CPUs
  - Simple dedicated instructions
  - Instruction-level parallelism
  - SIMD instructions with 256-bit registers
- Very simple and efficient side channel protection

Some references

- ***Keccak implementation overview*** (version 3.1 or later)
- *Note on side-channel attacks and their counterm…*, NIST hash forum 2009
- *Building power analysis resistant implementations of* Keccak, SHA-3 2010
- *Note on* Keccak *parameters and usage*, NIST hash forum 2010
- Software implementations
  - Bernstein and Lange, *eBASH*
  - Wenzel-Benner and Gräf, *XBX*
- Hardware implementations on FPGA
  - Kerckhof et al., CARDIS 2011
  - Jungk and Apfelbeck, ReConFig 2011
  - San and At, ISJ 2012
  - ATHENa project
- Hardware implementations on ASIC
  - Henzen et al., CHES 2010
  - Tillich et al., SHA-3 2010
  - Guo et al., DATE 2012

http://keccak.noekeon.org/

Thank you!