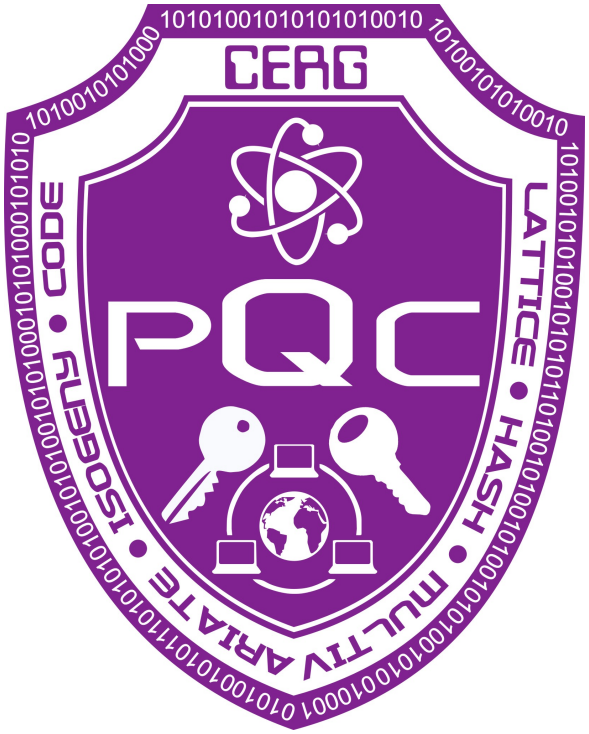
Toward Efficient and Fair Software/Hardware Codesign and Benchmarking of Candidates in Round 2 of the NIST PQC Standardization Process

Kris Gaj

George Mason University



Co-Authors

GMU PhD Students

Farnoud Farahmand

Viet Ba Dang

Duc Tri Nguyen

Visiting Scholar

Michał Andrzejczak

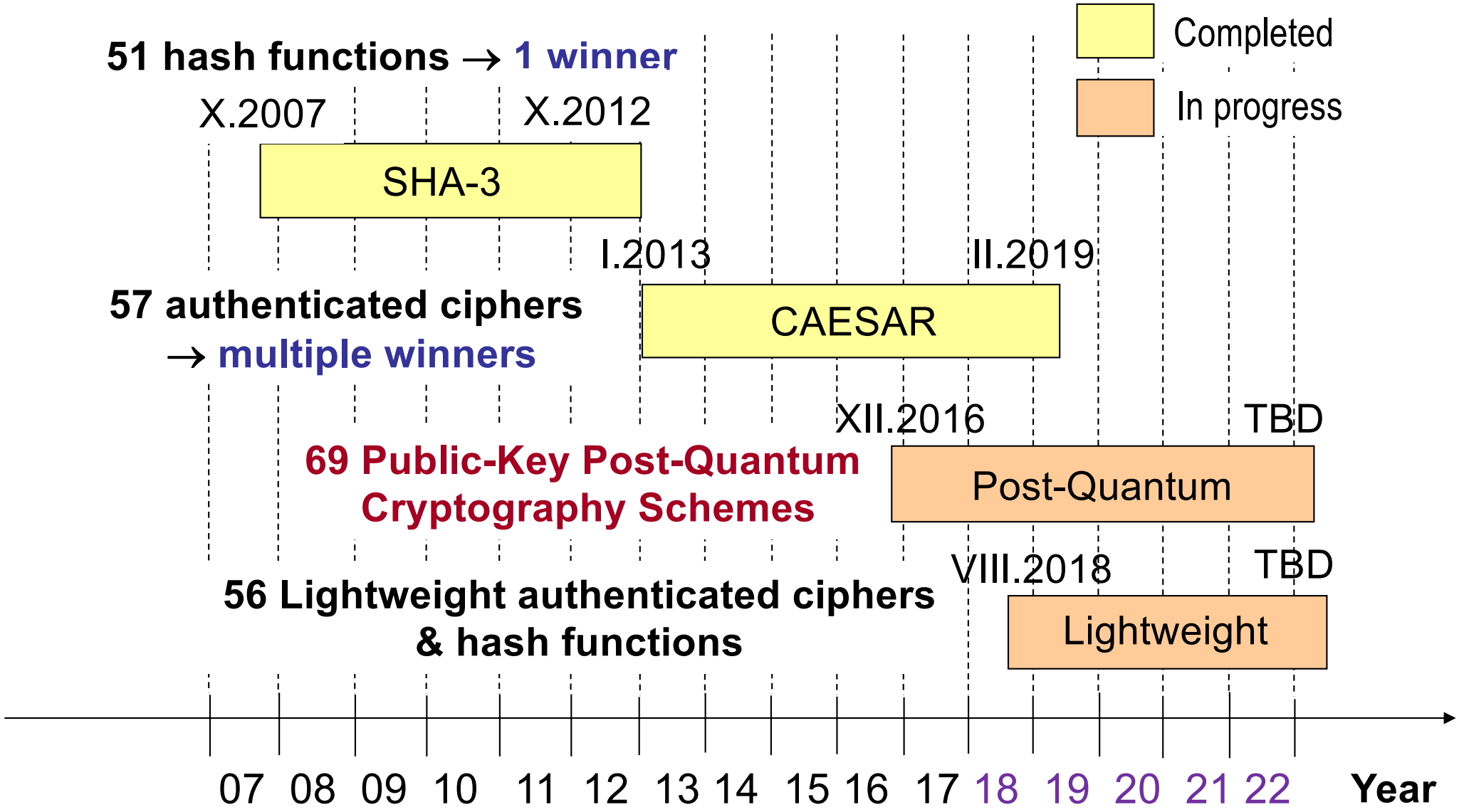
Military University of Technology in Warsaw, Poland

Post-Quantum Cryptography (PQC)

- Public-key cryptographic algorithms for which there are no known attacks using quantum computers
  - Capable of being implemented using any traditional methods, including software and hardware
  - Running efficiently on any modern computing platforms: smartphones, tablets, PCs, servers with FPGA accelerators, etc.
- Term introduced by Dan Bernstein in 2003
- Equivalent terms:

  quantum-proof, quantum-safe, quantum-resistant
- Based entirely on traditional semiconductor VLSI technology!

Cryptographic Contests 2007-Present

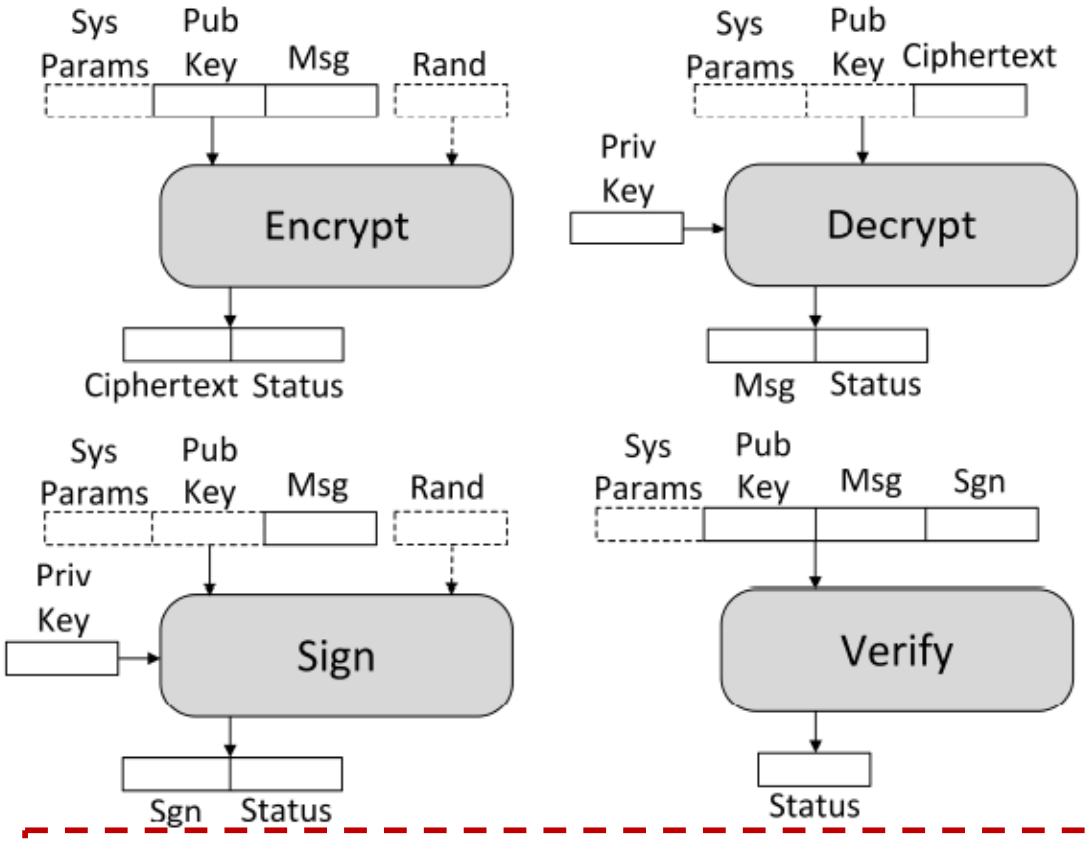


NIST PQC Standardization Process

- Dec. 2016: NIST Call for Proposals and Request for Nominations for Public-Key Post-Quantum Cryptographic Algorithms
- Nov. 30, 2017: Deadline for submitting candidates
- Jan. 30, 2019: Announcement of candidates qualified to Round 2
- April 10, 2019: Publication of Round 2 submission packages

- Aug. 22-24, 2019: Second NIST PQC Conference
- 2020: Beginning of Round 3 and/or selection of first future standards
- 2022-2024: Draft standards published

Three Types of PQC Schemes



1. Public Key Encryption

2. Digital Signature

Focus of this study

3. Key Encapsulation Mechanism (KEM)

Round 2 Candidates

26 Candidates announced on January 30, 2019

Family	Signature	Encryption/KEM	Overall
Lattice-based	3	9	12
Code-based		7	7
Multivariate	4		4
Hash-based	2		2
Isogeny-based		1	1
Total	9	17	26

Five Security Categories

Level	Security Description
I	At least as hard to break as AES-128 using exhaustive key search
II	At least as hard to break as SHA-256 using collision search
III	At least as hard to break as AES-192 using exhaustive key search
IV	At least as hard to break as SHA-384 using collision search
V	At least as hard to break as AES-256 using exhaustive key search

Results reported in this presentation


Hardware
Benchmarking

Percentage of Candidates in Hardware

	Initial number of candidates	Implemented in hardware	Percentage
AES	15	5	33.3%
eSTREAM	34	8	23.5%
SHA-3	51	14	27.5%
CAESAR	57	28	49.1%
PQC	69	?	?

Purely Hardware Implementations of NIST PQC Candidates (1)

Lattice-based:

NewHope

- Tobias Oder & Tim Guneysu, **Ruhr-University Bochum, Germany**
  LATINCRYPT 2017, **Sep. 2017**, **Open-source**

FrodoKEM

- James Howe, et al., **University of Bristol, UK & Ruhr-University Bochum, Germany**
  IACR TCHES, Aug. 2018

Round5

- Michal Andrzejczak, et al., **Military University of Technology in Warsaw, Poland & George Mason University, USA**
  Apr. 2019, unpublished

Purely Hardware Implementations of Round 1 NIST PQC Candidates (2)

Code-based:

Classic McEliece

- Wen Wang, Jakub Szefer, & Ruben Niederhagen, **Yale University, USA & Fraunhofer SIT, Darmstadt, Germany**
  CHES 2017, Sep. 2017 & PQCrypto 2018, Apr. 2018, **Open-source**

Multivariate:

Rainbow

- Ahmed Ferozpuri & Kris Gaj, **George Mason University, USA**
  ReConFig 2018, Dec. 2018

Isogeny-based:

SIKE

- Brian Koziel & Reza Azarderakhsh, **Texas Instruments & Florida Atlantic University, USA**
  https://sike.org, **Open-source**

Purely Hardware Implementations of NIST PQC Candidates: Summary

6 out of 69 Round 1 candidates

only 3 implementations open source

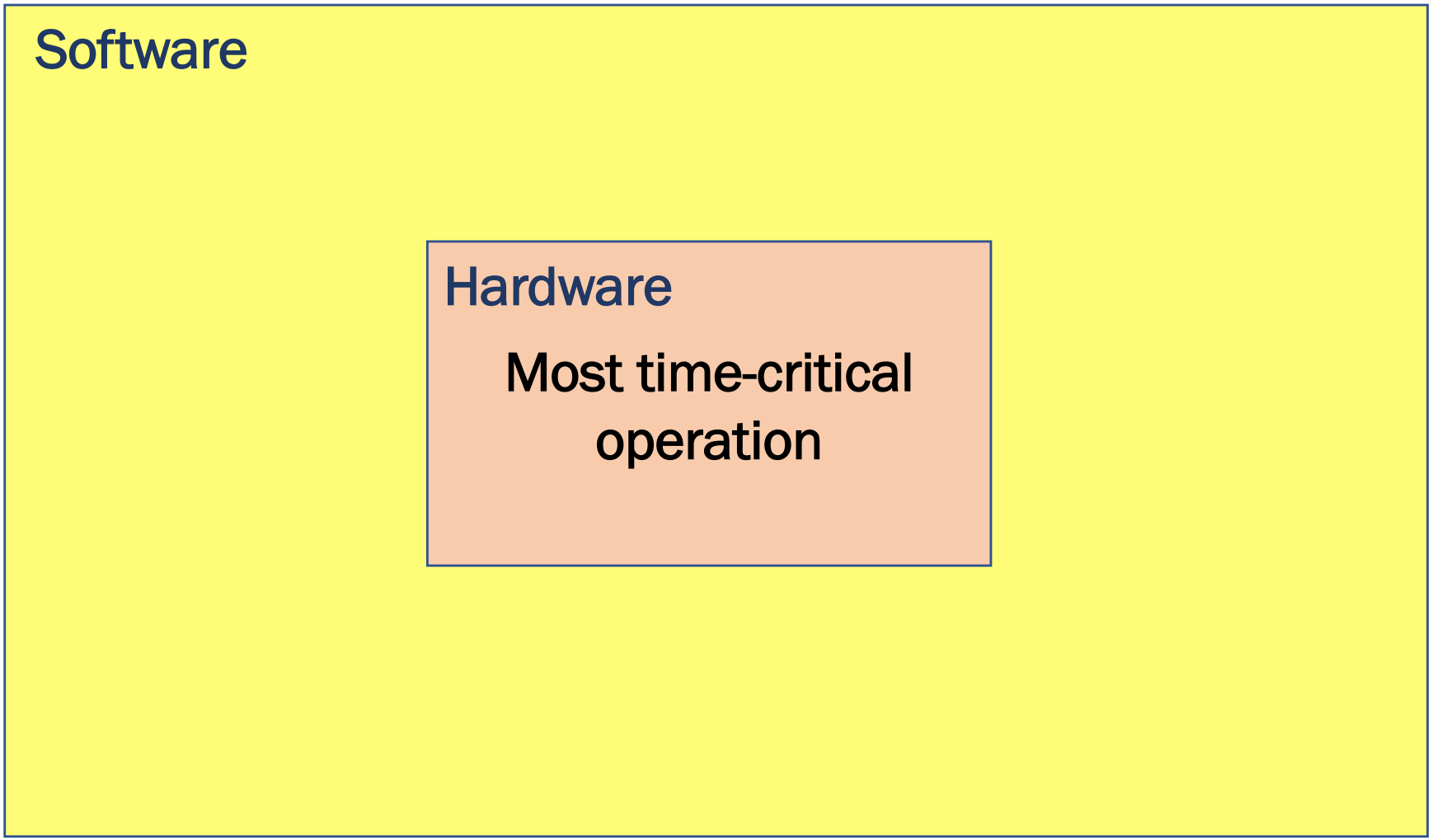
These designs follow different assumptions regarding

- **Operations supported by a hardware module, e.g.,**
  key generation included or excluded
- **Interface & communication protocol**
  different for each of the above implementations
- **Optimization target**
  min. latency vs. min. area vs. min. latency x area product
- **Platform**
  different FPGA families

No conclusions regarding ranking can be reached based on such divergent efforts!

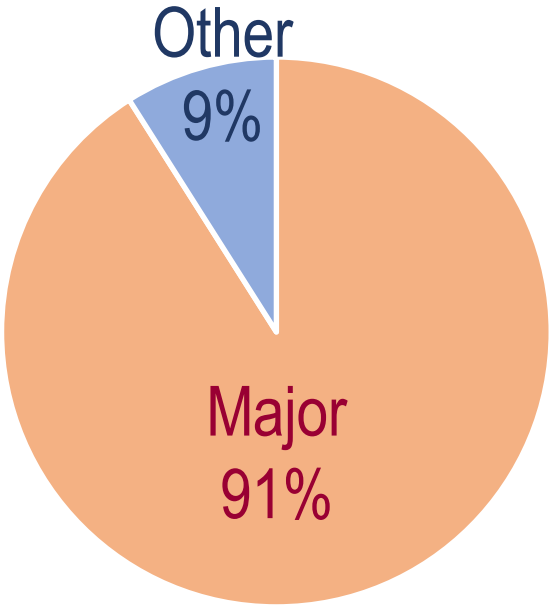

Software/Hardware
Codesign

Software/Hardware Codesign

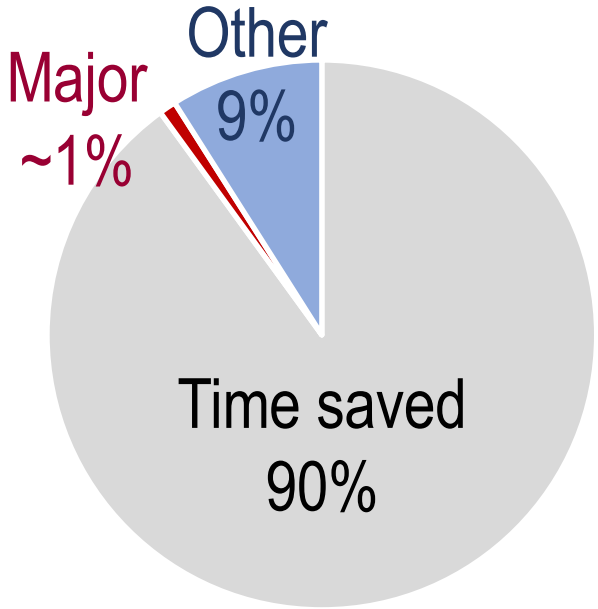


SW/HW Codesign: Motivational Example 1

Software



Software/Hardware



speed-up ≥ 100

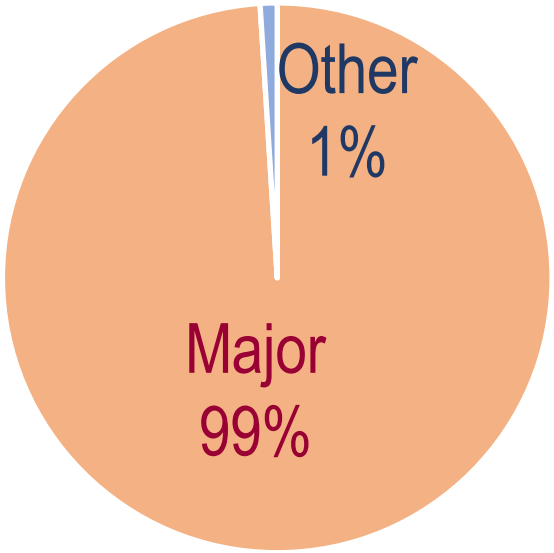
91% major operation(s)
9% other operations

~1% major operation(s) in HW
9% other operations in SW

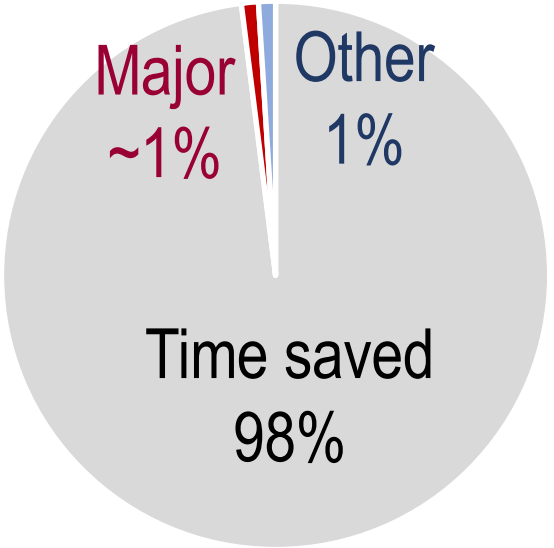
Total Speed-Up ≥ 10

SW/HW Codesign: Motivational Example 2

Software



Software/Hardware



speed-up ≥ 100

99% major operation(s)
1% other operations

~1% major operation(s) in HW
1% other operations in SW

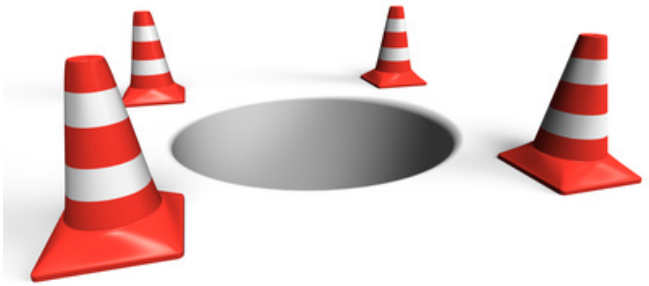
Total Speed-Up ≥ 50

SW/HW Codesign: Advantages

- Focus on a few (typically 1-3) major operations, known to be easily parallelizable
  - ☆ much shorter development time (at least by a factor of 10)
  - ☆ guaranteed substantial speed-up
  - ☆ high-flexibility to changes in other operations (such as candidate tweaks)
- Insight regarding performance of future instruction set extensions of modern microprocessors
- Possibility of implementing multiple candidates by the same research group, eliminating the influence of different
  - ☆ design skills
  - ☆ operation subset (e.g., including or excluding key generation)
  - ☆ interface & protocol
  - ☆ optimization target
  - ☆ platform

SW/HW Codesign: Potential Pitfalls

- Performance & ranking may strongly depend on features of a particular platform
  - ☆ Software/hardware interface
  - ☆ Support for cache coherency
  - ☆ Differences in max. clock frequency

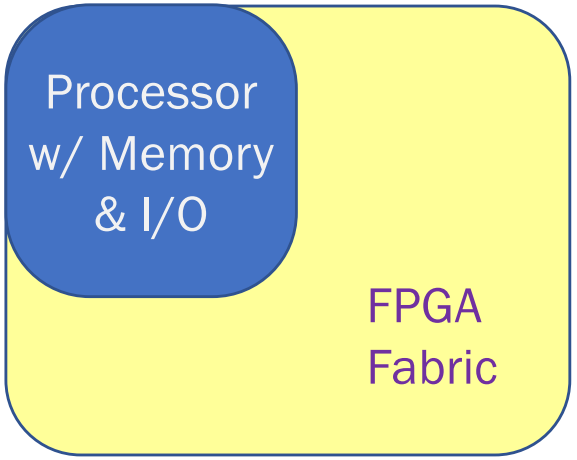


- Performance & ranking may strongly depend on the selected hardware/software partitioning
- Limited insight on ranking of purely hardware implementations

First step, not the ultimate solution!

Two Major Types of Platforms

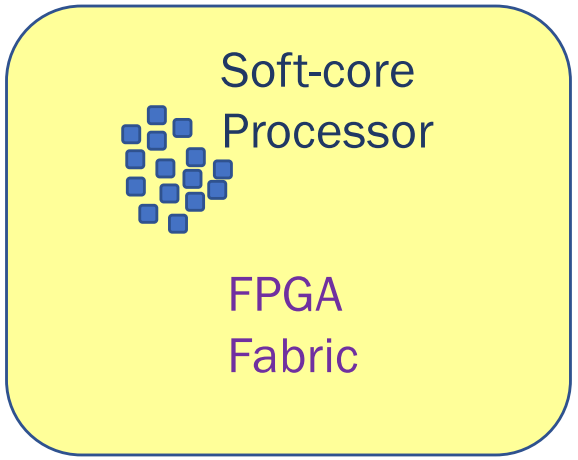
FPGA Fabric & Hard-core Processors



Examples:

- Xilinx Zynq 7000 System on Chip (SoC)
- Xilinx Zynq UltraScale+ MPSoC
- Intel Arria 10 SoC FPGAs
- Intel Stratix 10 SoC FPGAs

FPGA Fabric, including Soft-core Processors



Examples:

Xilinx Virtex UltraScale+ FPGAs

Intel Stratix 10 FPGAs, including

- Xilinx MicroBlaze
- Intel Nios II
- RISC-V, originally UC Berkeley

Two Major Types of Platform

Feature	FPGA Fabric and Hard-core Processor	FPGA Fabric with Soft-core Processor
Processor	ARM	MicroBlaze, NIOS II, RISC-V, etc.
Clock frequency	>1 GHz	max. 200-450 MHz
Portability	similar FPGA SoCs	various FPGAs, FPGA SoCs, and ASICs
Hardware accelerators	Yes	Yes
Instruction set extensions	No	Yes
Ease of design (methodology, tools, OS support)	Easy	Dependent on a particular soft-core processor and tool chain

Selected Platform

FPGA Family: Xilinx Zynq UltraScale+ MPSoC

Device: XCZU9EG-2FFVB1156E

Prototyping Board: ZCU102 Evaluation Kit from Xilinx

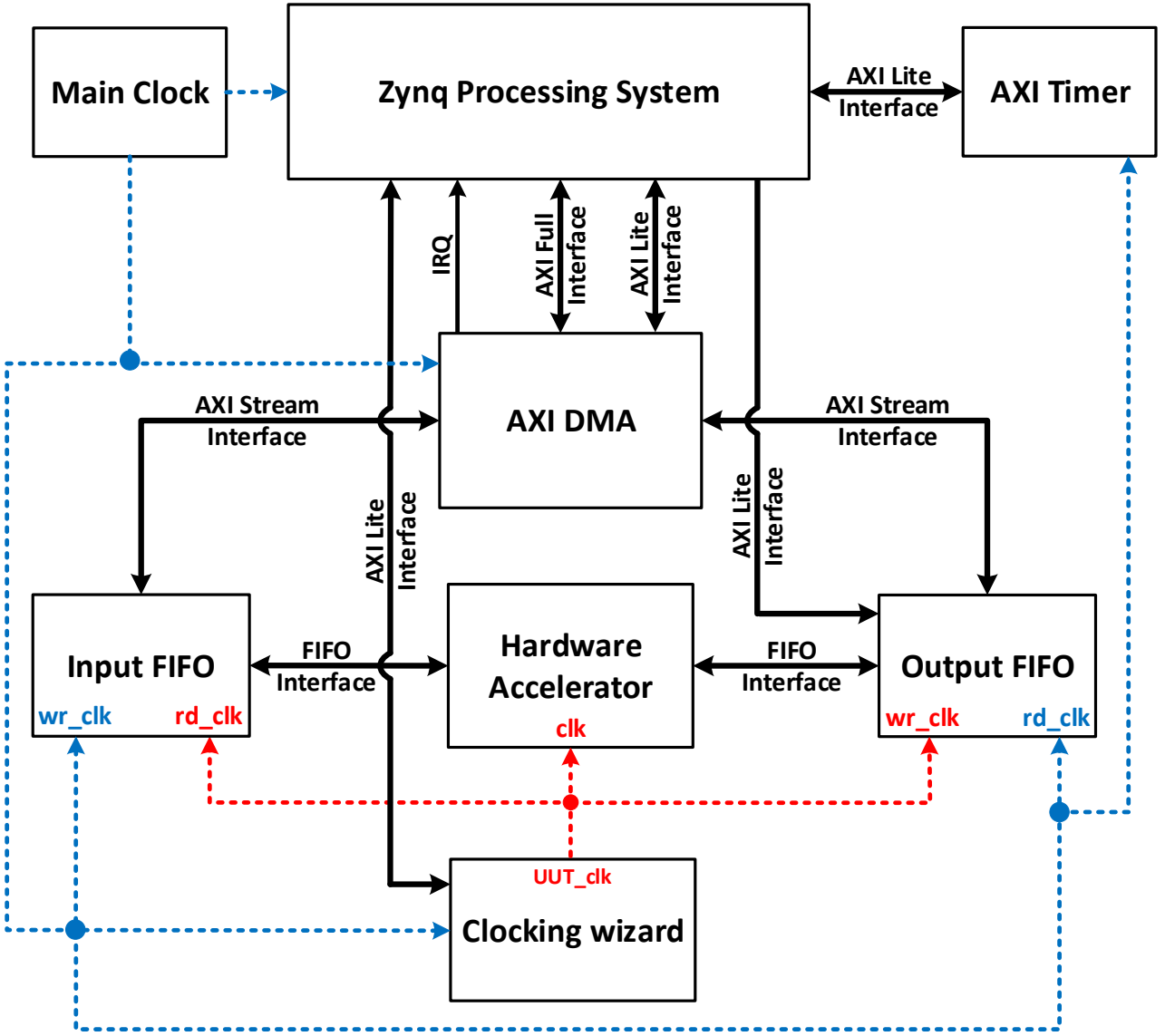
Processing System:

- ☆ **Quad-core ARM Cortex-A53 Application Processing Unit**, running at the frequency of **1.2 GHz** (only one core used for benchmarking)

Programmable Logic:

- ☆ **Configurable Logic Blocks (CLB), Block RAMs, DSP units**

Experimental Setup



All elements located on a single chip

Our Case Study

SW/HW Codesign: Case Study

7 IND-CCA*-secure Lattice-Based Key Encapsulation Mechanisms (KEMs)

representing

5 NIST PQC Round 2 Submissions

LWE (Learning with Error)-based:

FrodoKEM

RLWR (Ring Learning with Rounding)-based:

Round5

Module-LWR-based:

Saber

NTRU-based:

NTRU

- NTRU-HPS
- NTRU-HRSS

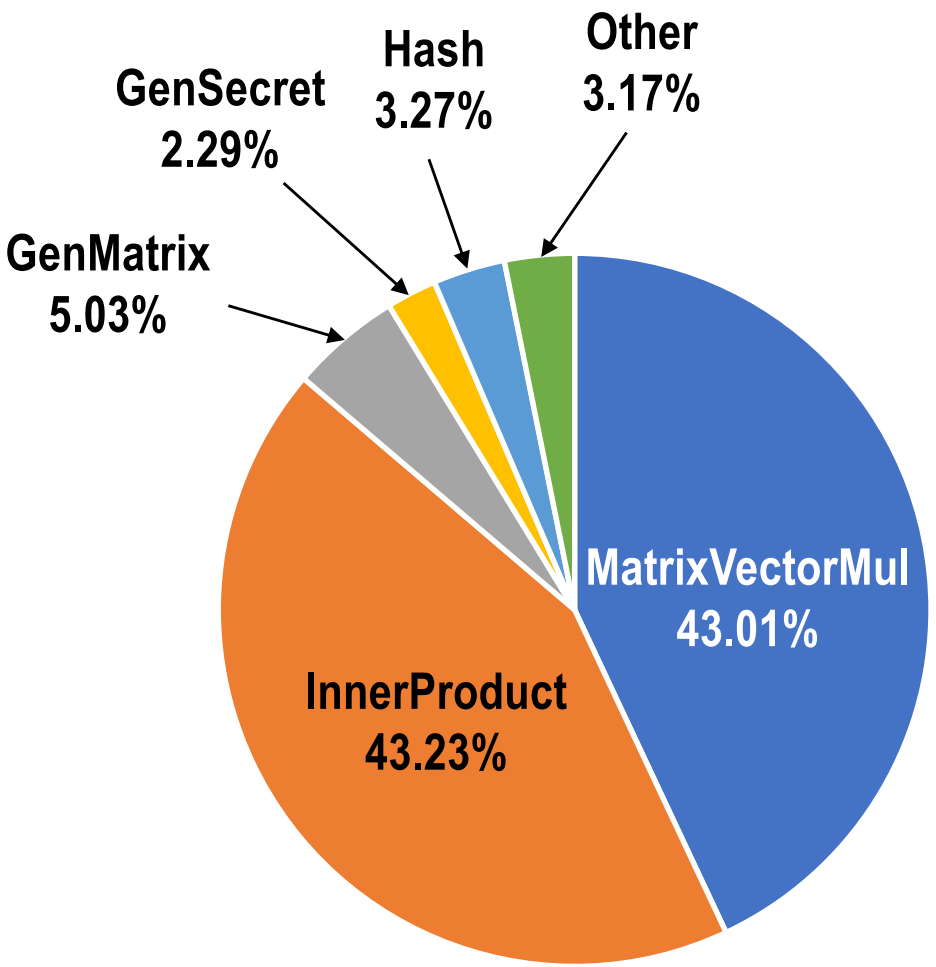
NTRU Prime

- Streamlined NTRU Prime
- NTRU LPRime

* IND-CCA = with Indistinguishability under Chosen Ciphertext Attack

Example of SW/HW Partitioning: Saber - Decapsulation

Saber Decapsulation



SW/HW Partitioning

Top candidates for offloading to hardware

From profiling:

- Large percentage of the execution time
- Small number of function calls

From manual analysis of the code:

- Small size of inputs and outputs
- Potential for combining with neighboring functions

From knowledge of operations and concurrent computing:

- High potential for parallelization

Algorithm $\texttt{Saber.KEM.Encaps}(pk = (seed_{\boldsymbol{A}}, \boldsymbol{b}))$

$m \leftarrow \mathcal{U}(\{0,1\}^{256})$
$(\hat{K}, r) = \mathcal{G}(\mathcal{F}(pk), m)$
$c = \texttt{Saber.PKE.Enc}(pk, m; r)$
$K = \mathcal{H}(\hat{K}, c)$
return (c, K)

Algorithm $\texttt{Saber.KEM.Decaps}(sk = (\boldsymbol{s}, z, pkh), pk = (seed_{\boldsymbol{A}}, \boldsymbol{b}), c)$

$m' = \texttt{Saber.PKE.Dec}(\boldsymbol{s}, c)$
$(\hat{K}', r') = \mathcal{G}(pkh, m')$
$c' = \texttt{Saber.PKE.Enc}(pk, m'; r')$
if $c = c'$ **then**
  return $K = \mathcal{H}(\hat{K}', c)$
else
  return $K = \mathcal{H}(z, c)$

Algorithm $\texttt{Saber.PKE.Enc}(pk = (seed_{\boldsymbol{A}}, \boldsymbol{b}), m \in R_2; r)$

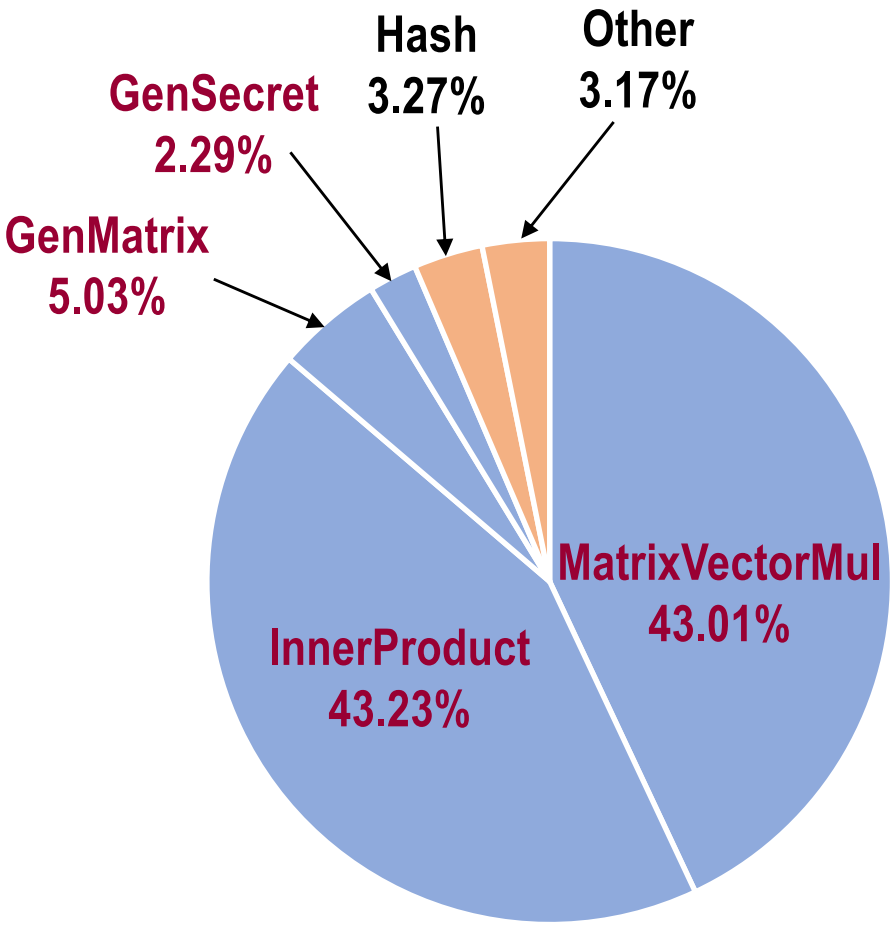
$\boldsymbol{A} = \texttt{gen}(\text{seed}_{\boldsymbol{A}}) \in R_q^{l \times l}$
if *r is not specified* **then**
  $r = \mathcal{U}(\{0,1\}^{256})$
$\boldsymbol{s'} = \beta_\mu(R_q^{l \times 1}; r)$
$\boldsymbol{b'} = ((\boldsymbol{A}\boldsymbol{s'} + \boldsymbol{h}) \bmod q) \gg (\epsilon_q - \epsilon_p) \in R_p^{l \times 1}$
$v' = \boldsymbol{b}^T(\boldsymbol{s'} \bmod p) \in R_p$
$c_m = (v' + h_1 - 2^{\epsilon_p - 1} m \bmod p) \gg (\epsilon_p - \epsilon_T) \in R_T$
return $c := (c_m, \boldsymbol{b'})$

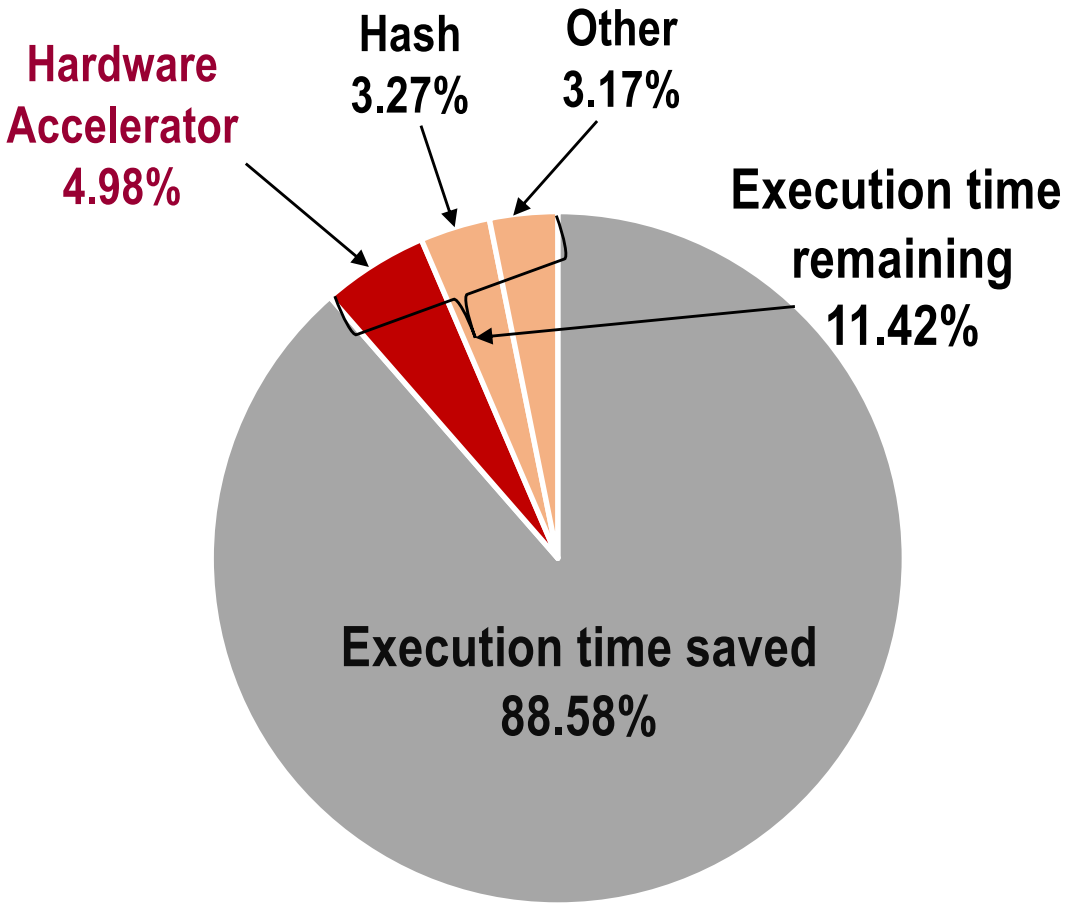
Algorithm $\texttt{Saber.PKE.Dec}(sk = \boldsymbol{s}, c = (c_m, \boldsymbol{b'}))$

$v = \boldsymbol{b'}^T(\boldsymbol{s} \bmod p) \in R_p$
$m' = ((v - 2^{\epsilon_p - \epsilon_T} c_m + h_2) \bmod p) \gg (\epsilon_p - 1) \in R_2$
return m'

Parts moved to hardware

Saber Decapsulation





Execution time of functions to be moved to hardware
93.56%

Execution time of functions remaining in software
6.44%

Accelerator Speed-Up = 93.56/4.98=18.8

Total Speed-Up = 100/11.42=8.8

Results

Clock Frequency & Resource Utilization

Algorithm	Clock Freq [MHz]	#LUTs	#Slices	#FFs	#36kb BRAMs	#DSPs
FrodoKEM	402	7,213	1,186	6,647	13.5	32
Round5	260	N/A	10,381	N/A	0	0
Saber	322	12,343	1,989	11,288	3.5	256
NTRU-HPS	200	24,328	4,972	19,244	2.5	677
NTRU-HRSS	200	27,218	5,770	21,410	2.5	701
Str NTRU Prime	244	55,843	8,134	28,143	3.0	0
NTRU LPRime	244	50,911	7,874	34,050	2.0	0
Device		274,080	34,260	548,160	912	2,520
			< 31%		< 2%	< 28%

of total resources of the given device

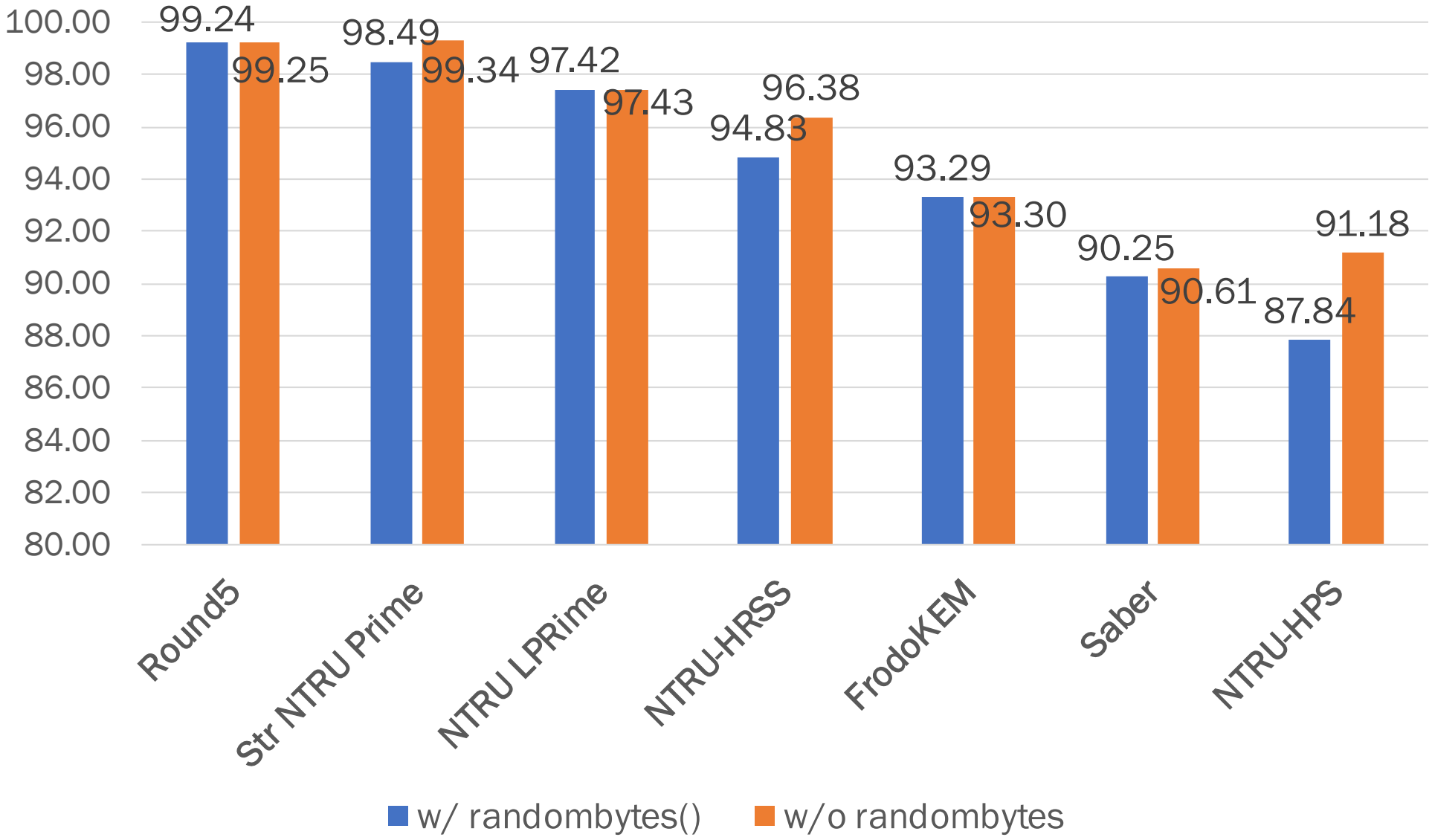
randombytes()

- Function used for generating pseudorandom byte sequences
- The implementation vary among various benchmarking studies, depending on the mode of operation (Bare Metal vs. Operating System), and availability of libraries, such as OpenSSL
- Used to different extent by implementations of various candidates

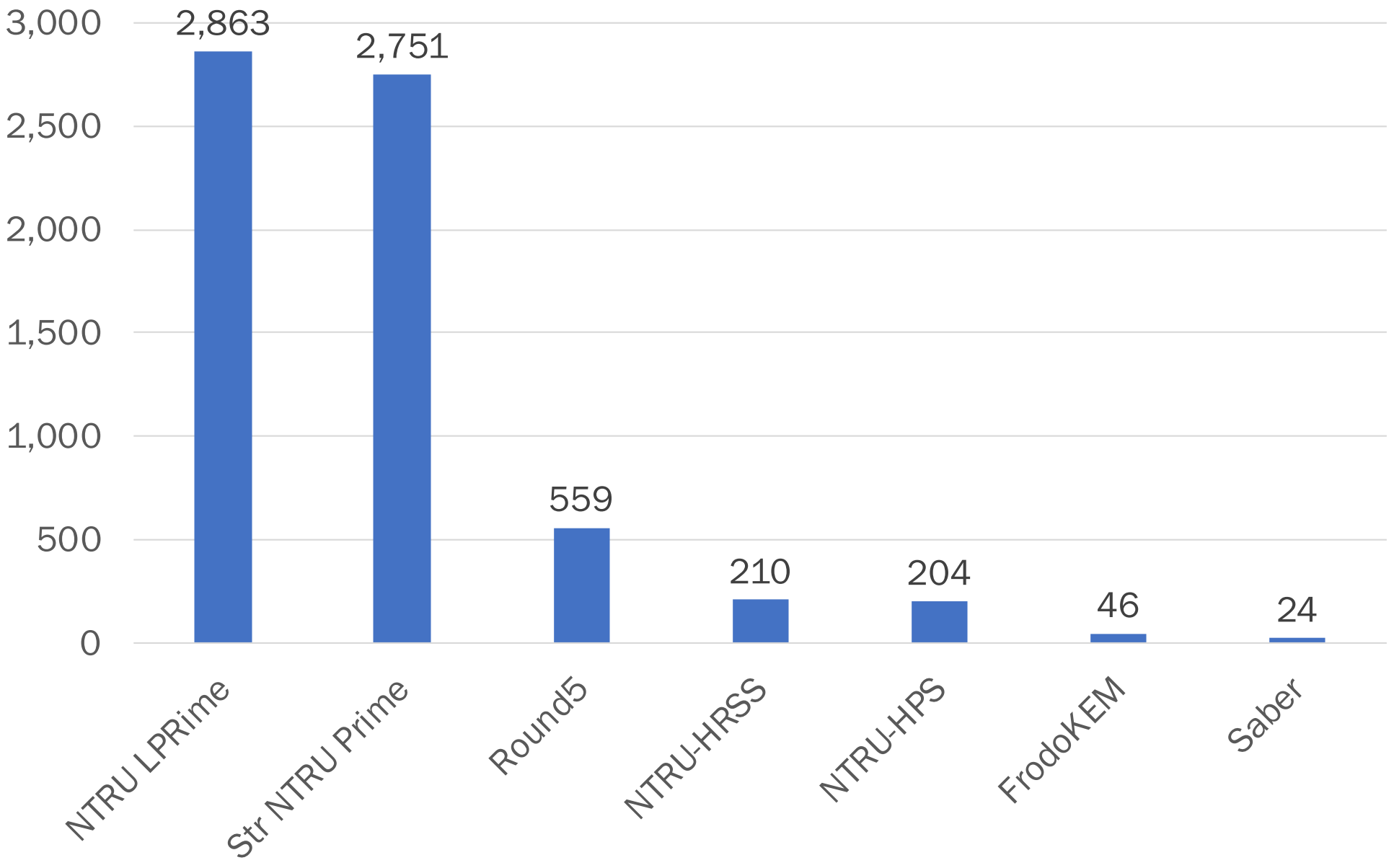
Algorithm	#Calls	#Bytes
FrodoKEM	1	16
Round5	1	16
Saber	1	32
NTRU-HPS	1	**3211**
NTRU-HRSS	1	**1400**
Str NTRU Prime	**653**	**2612**
NTRU LPRime	1	32

- For 3 algorithms could be sped-up over 3 times by using SHAKE128

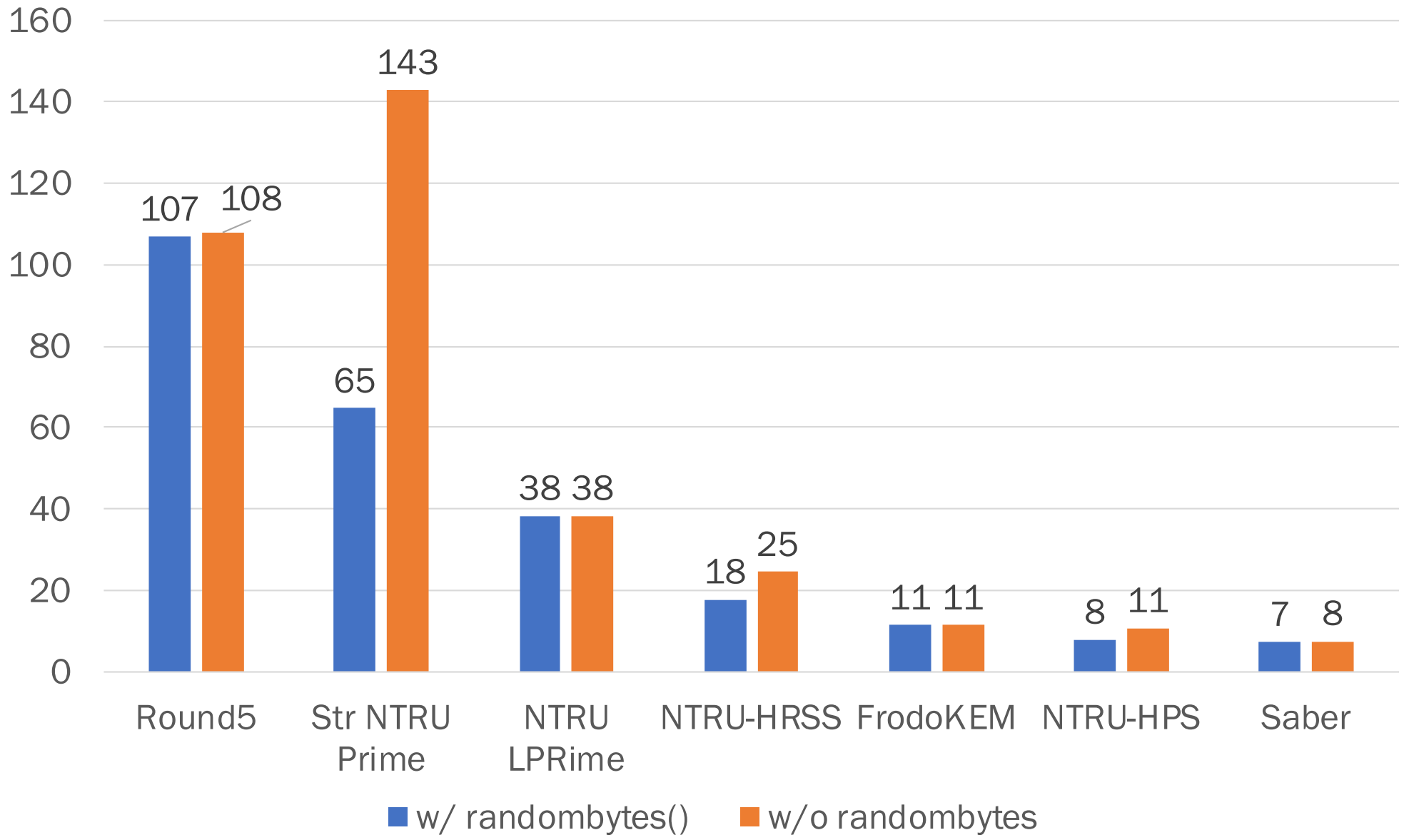
Software Part Sped up by Hardware [%]: Encapsulation



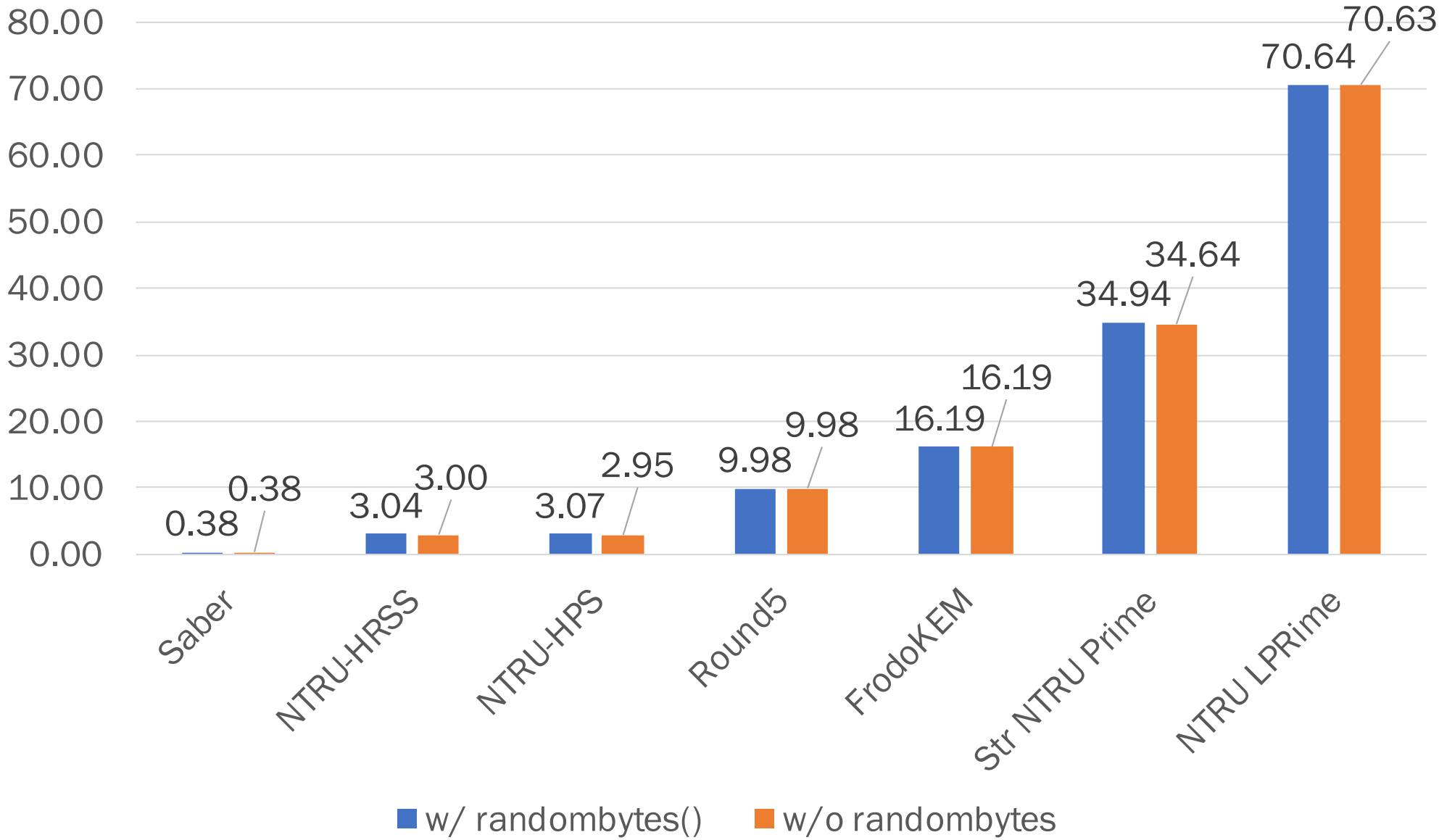
Accelerator Speed-ups: Encapsulation



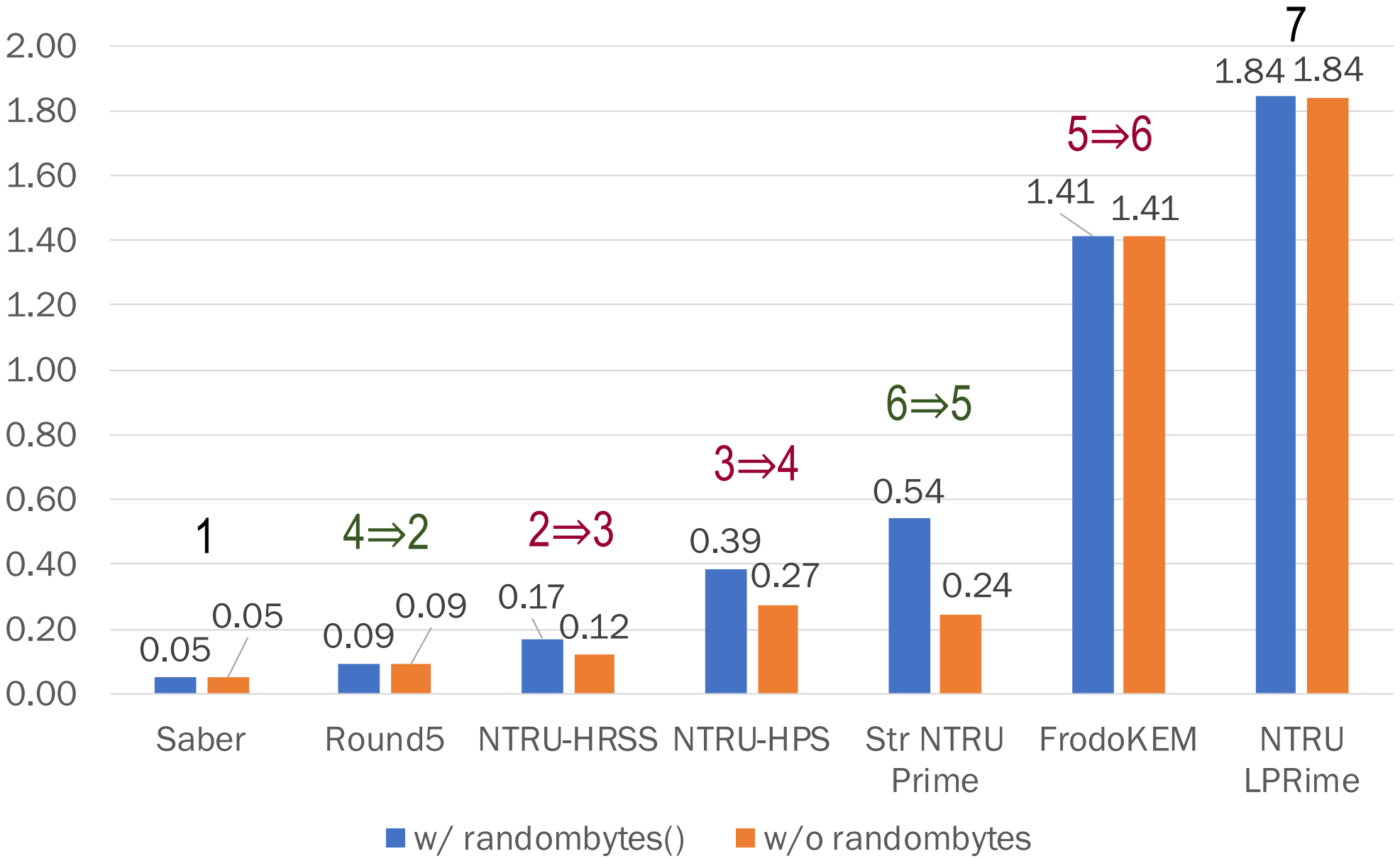
Total Speed-ups: Encapsulation



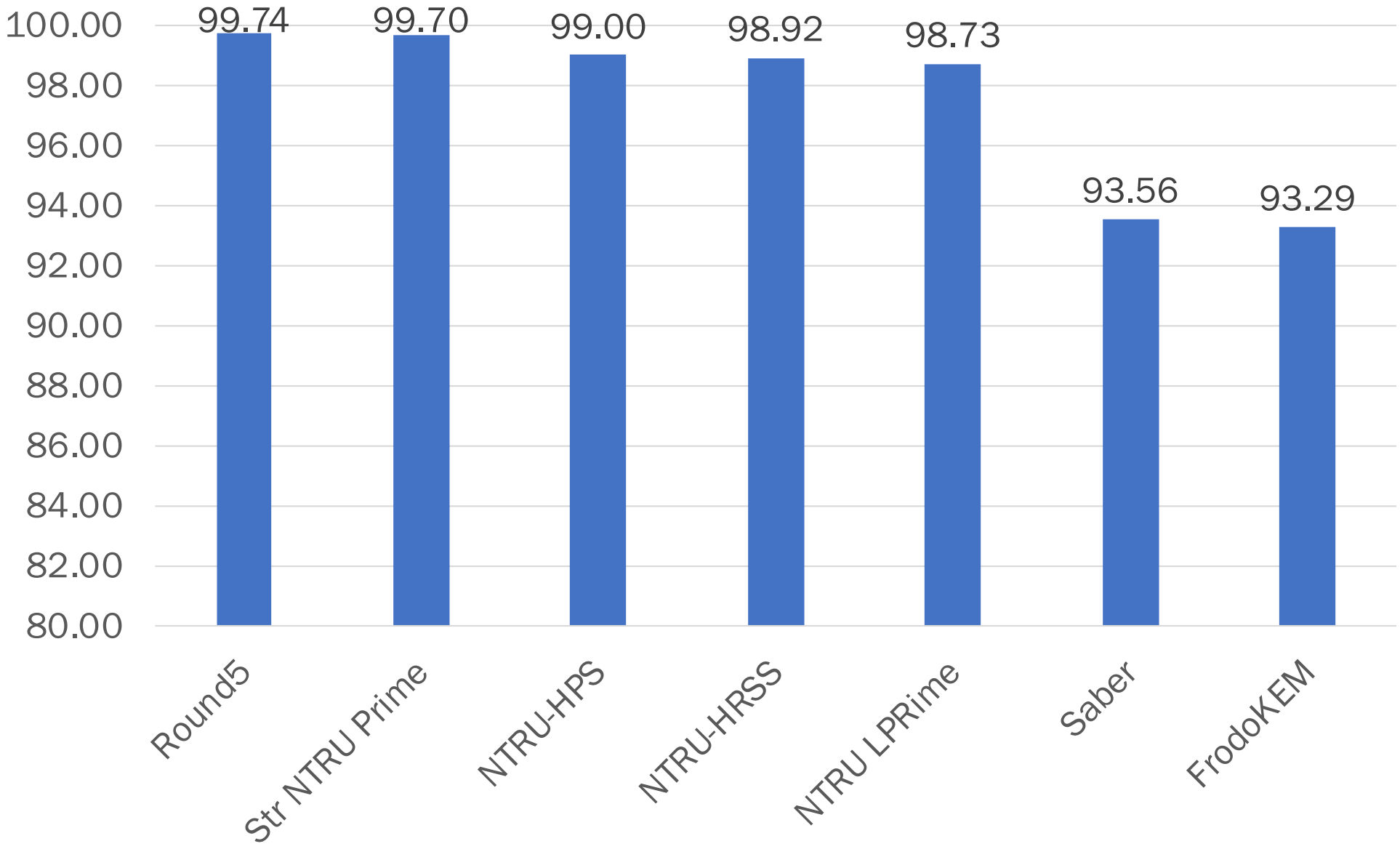
Total Execution Time in Software [ms]: Encapsulation



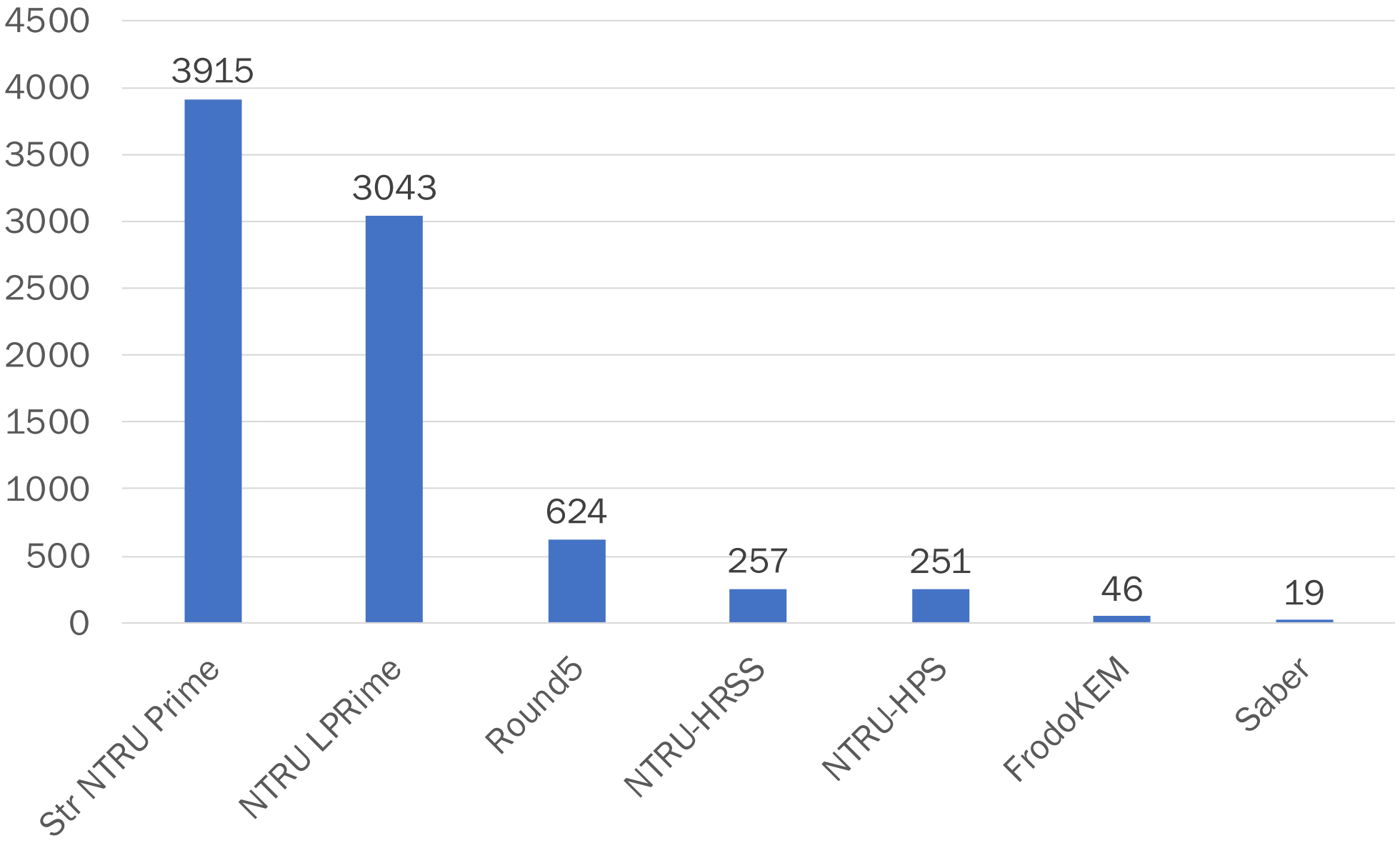
Total Execution Time in Software/Hardware [ms]: Encapsulation



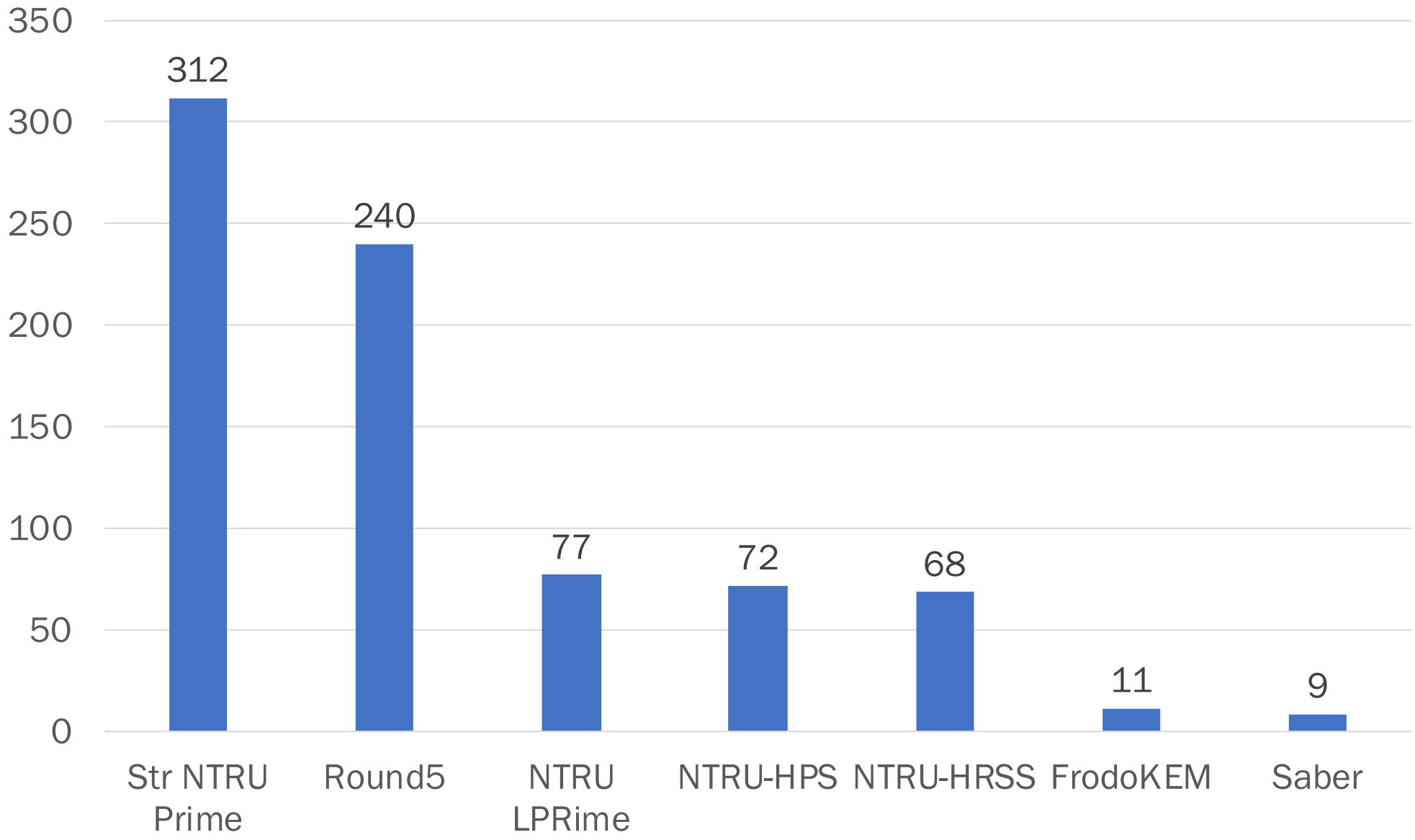
Software Part Sped up by Hardware [%]: Decapsulation



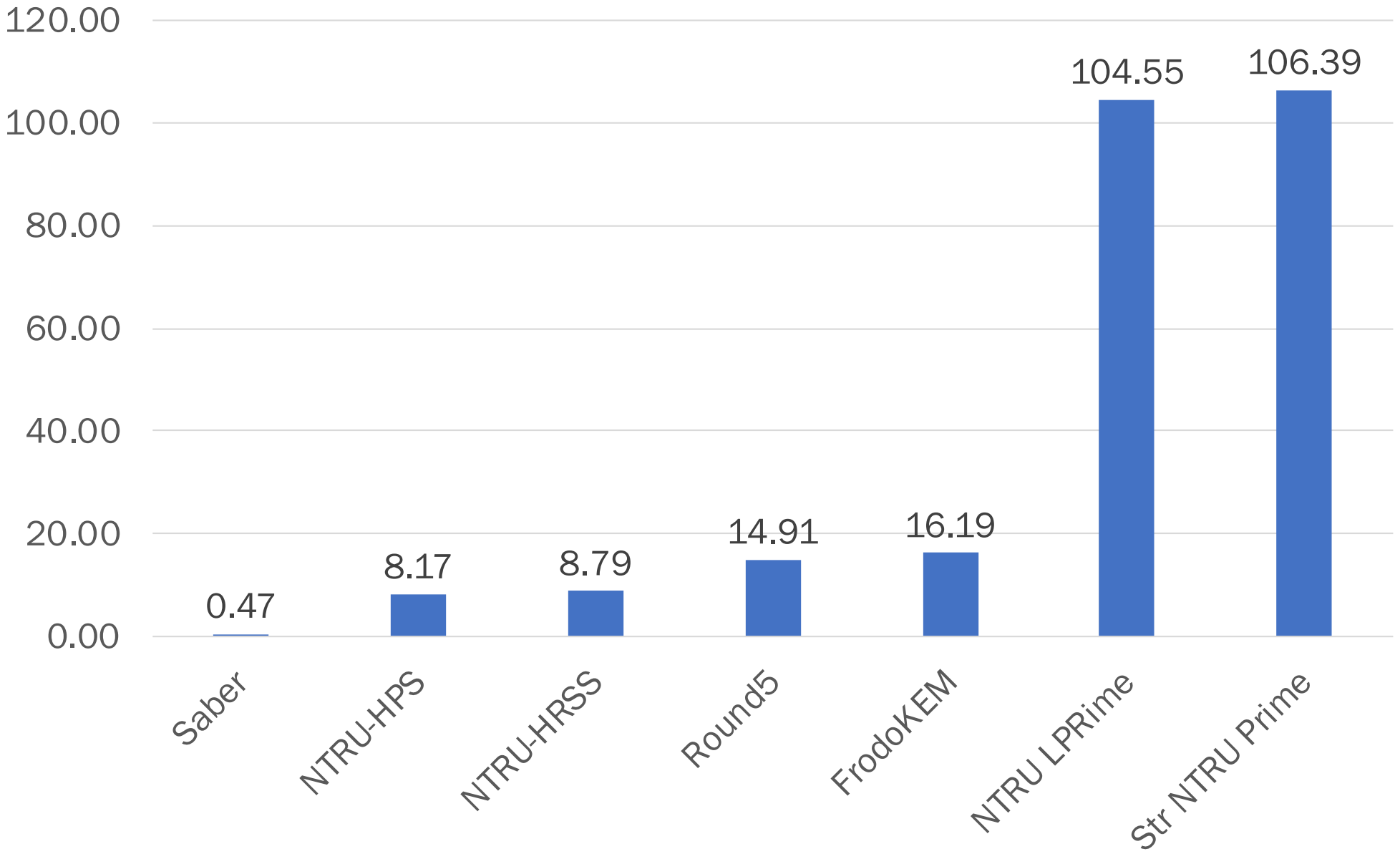
Accelerator Speed-ups: Decapsulation



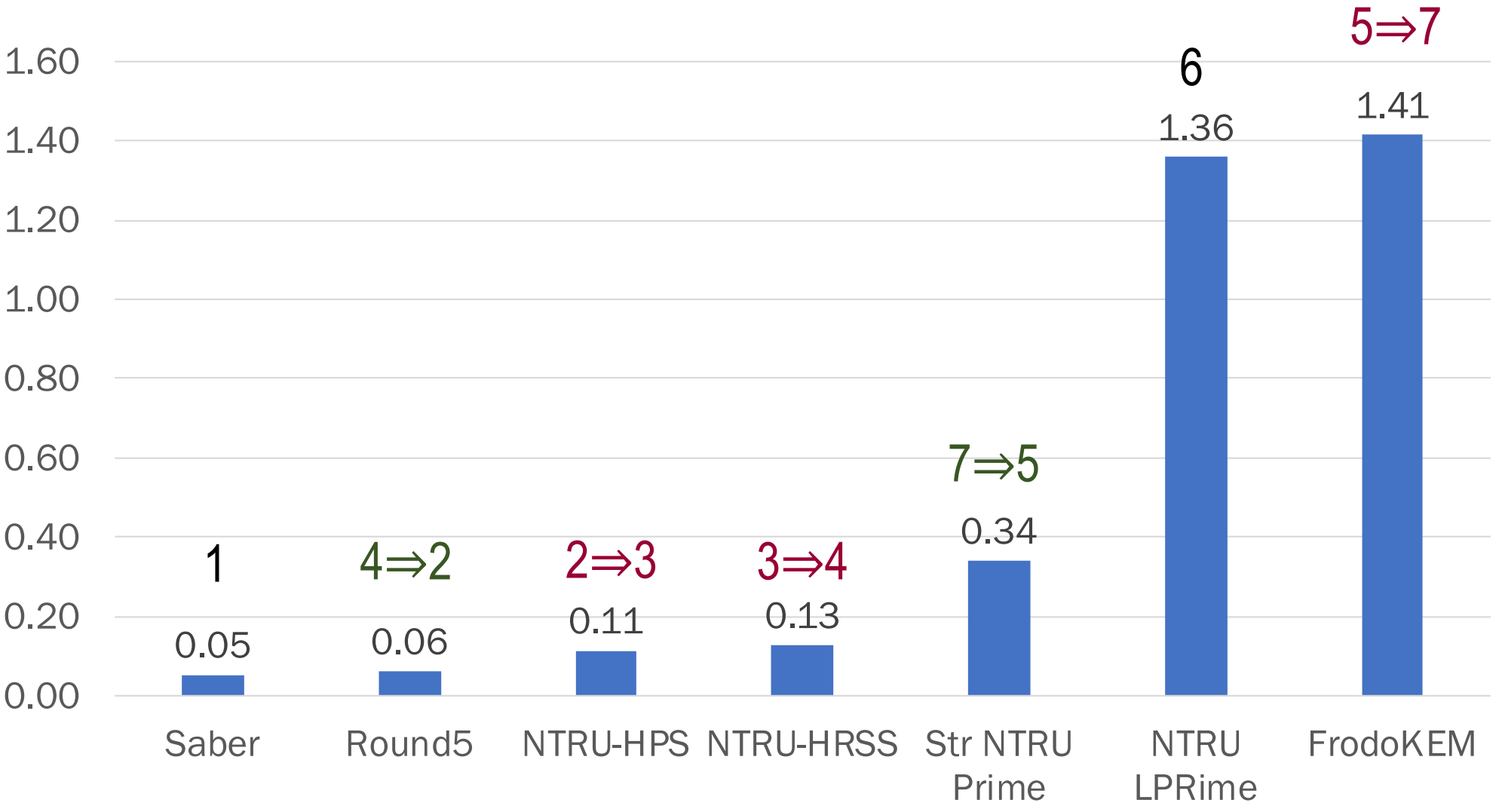
Total Speed-ups: Decapsulation



Total Execution Time in Software [ms]: Decapsulation



Total Execution Time in Software/Hardware [ms]: Decapsulation



Conclusions

- Total speed-ups
  - ☆ for encapsulation from 7 (Saber) to 107 (Round5)
  - ☆ for decapsulation from 9 (Saber) to 312 (Streamlined NTRU Prime)
- Total speed-up mostly dependent on the percentage of the software execution time taken by functions offloaded to hardware (rather than the amount of acceleration itself)
- Hardware accelerators thoroughly optimized using Register-Transfer Level design methodology
- Determining optimal software/hardware partitioning requires more work
- Ranking of the investigated candidates affected, but not dramatically changed, by hardware acceleration
- It is possible to complete similar designs for all Round 2 candidates within the evaluation period (12-18 months)
- Additional benefit: Comprehensive library of major operations in hardware

PQC Opportunities & Challenges

- The biggest revolution in cryptography, since the invention of public-key cryptography in 1970s
- Efficient hardware implementations in FPGAs and ASICs desperately needed to prove the candidates suitability for high-performance applications and constrained environments. Collaboration sought by submission teams!
- Likely extensions to Instruction Set Architectures of multiple major microprocessors
- Start-up & new-product opportunities
- Once in the lifetime opportunity! Get involved!

Q&A

Thank You!

Questions?

Comments?

Suggestions?

CERG: http://cryptography.gmu.edu

ATHENa: http://cryptography.gmu.edu/athena

Backup 1

Round 2 Submissions

- Lattice-based
- Code-based
- Isogenies

Encryption/KEMs (17)

- CRYSTALS-KYBER
- FrodoKEM
- LAC
- NewHope
- NTRU (merger of NTRUEncrypt/NTRU-HRSS-KEM)
- NTRU Prime
- Round5 (merger of Hila5/Round2)
- SABER
- Three Bears
- BIKE
- Classic McEliece
- HQC
- LEDAcrypt (merger of LEDAkem/pkc)
- NTS-KEM
- ROLLO (merger of LAKE/LOCKER/Ouroboros-R)
- RQC
- SIKE

Digital Signatures (9)

- Lattice-based
- Symmetric-based
- Multivariate

- CRYSTALS-DILITHIUM
- FALCON
- qTESLA
- GeMSS
- LUOV
- MQDSS
- Rainbow
- Picnic
- SPHINCS+

NIST Report on the 1st Round: https://doi.org/10.6028/NIST.IR.8240

Sources: Moody, PQCrypto May 2019

Adi Shamir's Proposal

After the initial evaluation period (e.g., 3 years) the division of all schemes into the following categories:

- 2 Productions Schemes: Recommended for actual wide-scale deployment. Highly Trusted.
- 4 Development Schemes: Time-Tested, Trusted.

  At least 15 years of analysis behind them.

  Intended for initial R&D by industry.
- 8 Research Schemes: Promising Properties, Good Performance. May contain some high-risk candidates. Main Goal: Concentrate the effort of the research community.

The Most Trusted Schemes – Encryption/KEM

Classical McEliece

- Proposed **40 years ago** as an alternative to RSA
- **Code-based** family
- Based on **binary Goppa** codes
- **No patents**
- **Conservative parameters (Category 5, 256-bit security):**
  a) length n=6960, dimension k= 5413, errors=119
  b) length n=8192, dimension k= 6528, errors=128
- **Complexity of the best attack identical after 40 years of analysis, and more than 30 papers devoted to thorough cryptanalysis**
- **Sizes:**

  Public key: a) 1,047,319 bytes, b) 1,357,824 bytes
  Private key: a) 13,908 bytes, b) 14,080 bytes
  Ciphertext: a) 226 bytes, b) 240 bytes
- **Efficient Software** (Haswell, larger parameter set)
  - ☆ 295,930 cycles for encryption, 355,152 cycles for decryption
  - ☆ Constant time
- **Efficient Hardware** (Yale University & Fraunhofer Institute SIT, Germany): open-source, targeting FPGAs; CHES'17, PQCrypto'18

The Most Trusted Schemes – Signatures

Hash-based Schemes:

Security based on the security of
a single underlying primitive: hash function

Representatives:

SPHINCS-256 => SPHINCS+

Features:

Relatively large signatures (~ tens of kilobytes)
Signing more time consuming than verification

No reported hardware implementations

Round 2 Submissions

- Lattice-based
- Code-based
- Isogenies

Encryption/KEMs (17)

- CRYSTALS-KYBER
- FrodoKEM
- LAC
- NewHope
- NTRU (merger of NTRUEncrypt/NTRU-HRSS-KEM)
- NTRU Prime
- Round5 (merger of Hila5/Round2)
- SABER
- Three Bears
- BIKE
- Classic McEliece
- HQC
- LEDAcrypt (merger of LEDAkem/pkc)
- NTS-KEM
- ROLLO (merger of LAKE/LOCKER/Ouroboros-R)
- RQC
- SIKE

Digital Signatures (9)

- Lattice-based
- Symmetric-based
- Multivariate

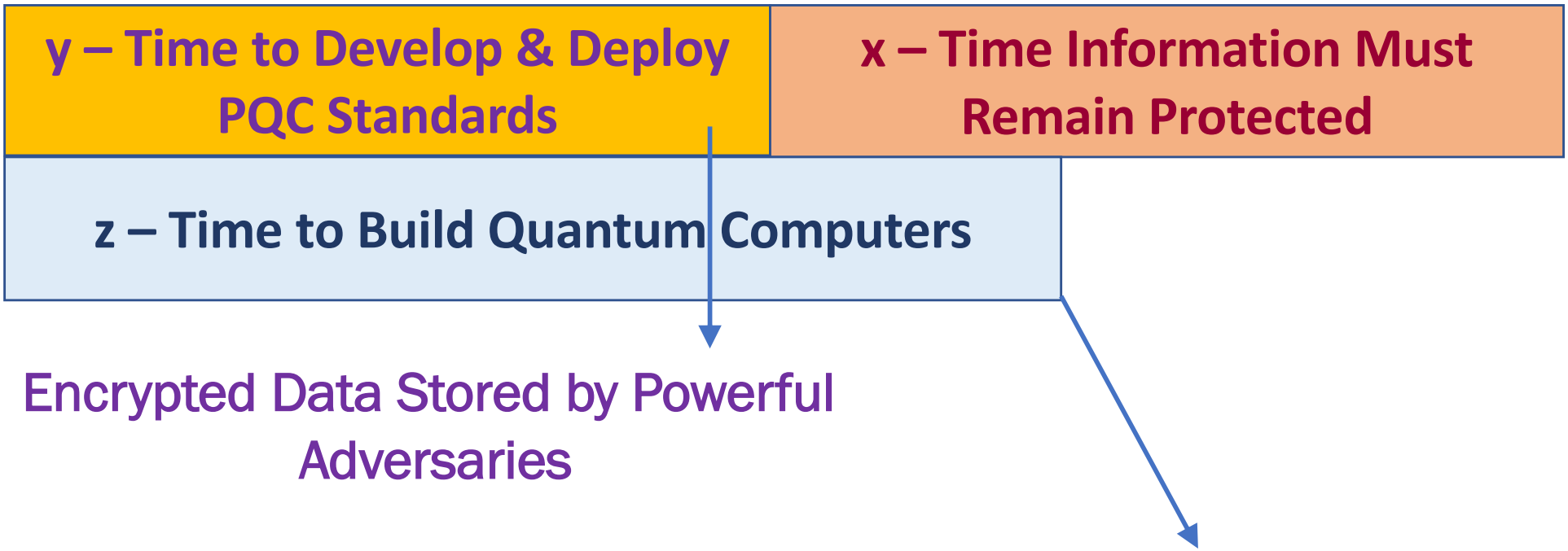
- CRYSTALS-DILITHIUM
- FALCON
- qTESLA
- GeMSS
- LUOV
- MQDSS
- Rainbow
- Picnic
- SPHINCS+

NIST Report on the 1st Round: https://doi.org/10.6028/NIST.IR.8240

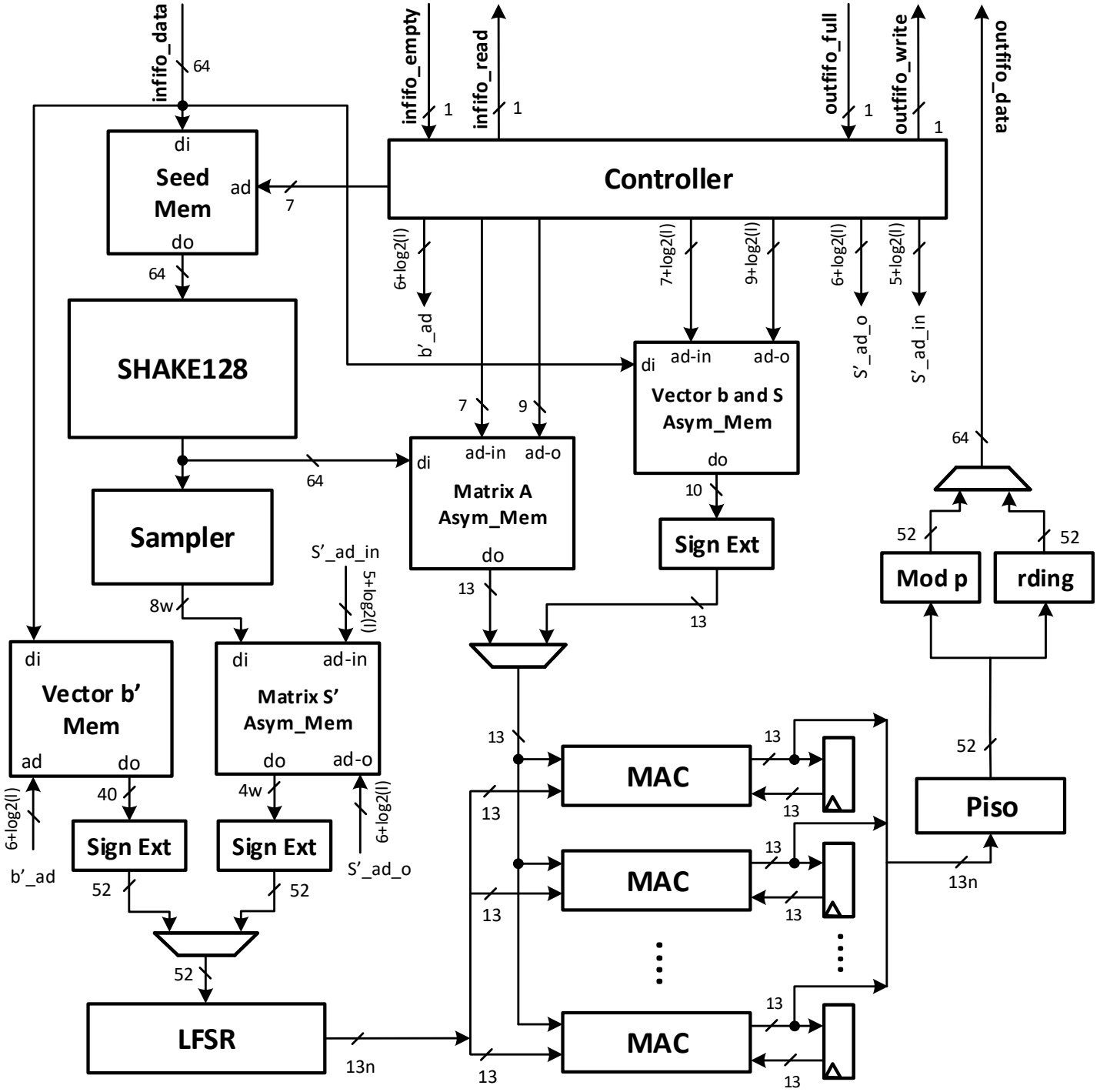
Sources: Moody, PQCrypto May 2019

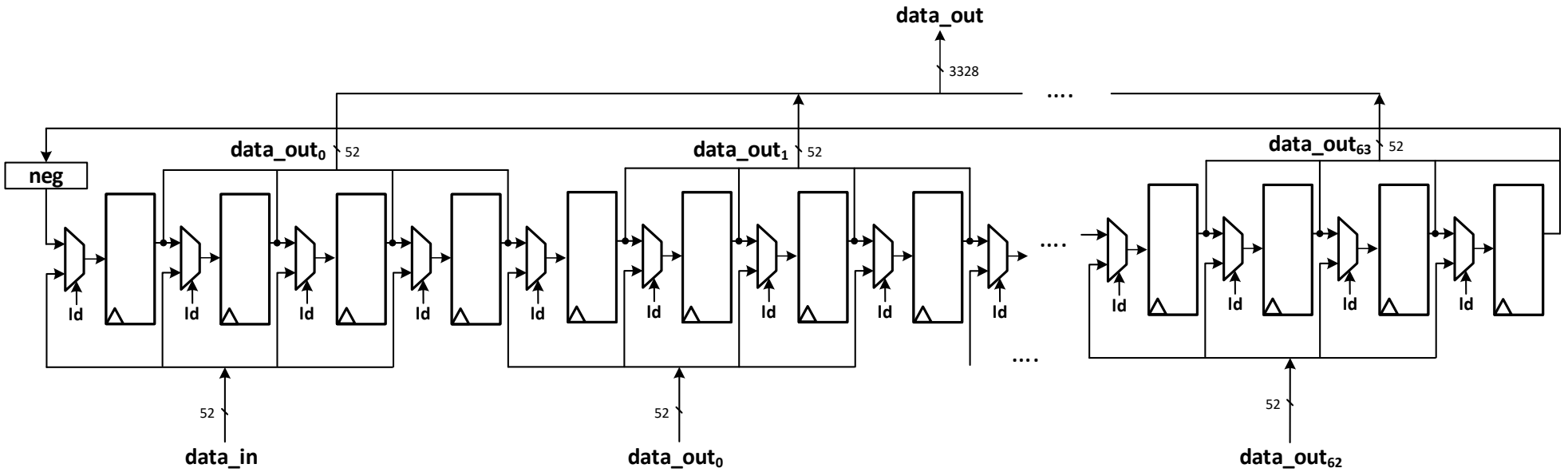
“Theorem” by Mosca

If $z < y + x$, then worry!

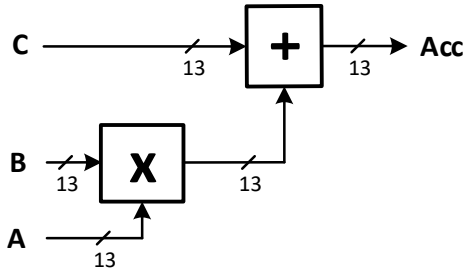


Saber Block Diagrams

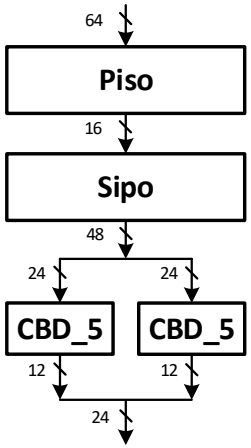

infifo_data
64
infifo_empty
1
infifo_read
1
outfifo_full
1
outfifo_write
1
outfifo_data
Controller
Seed Mem
di
ad
do
7
64
SHAKE128
6+log2(l)
b'_ad
7+log2(l)
9+log2(l)
6+log2(l)
S'_ad_o
5+log2(l)
S'_ad_in
Vector b and S Asym_Mem
di
ad-in
ad-o
do
10
Sign Ext
7
9
Matrix A Asym_Mem
64
Sampler
8w
S'_ad_in
5+log2(l)
13
64
52
52
Mod p
rding
Vector b' Mem
ad
do
40
6+log2(l)
b'_ad
Sign Ext
52
Matrix S' Asym_Mem
4w
6+log2(l)
S'_ad_o
Sign Ext
52
52
LFSR
13n
MAC
MAC
MAC
13
Piso
52
13n

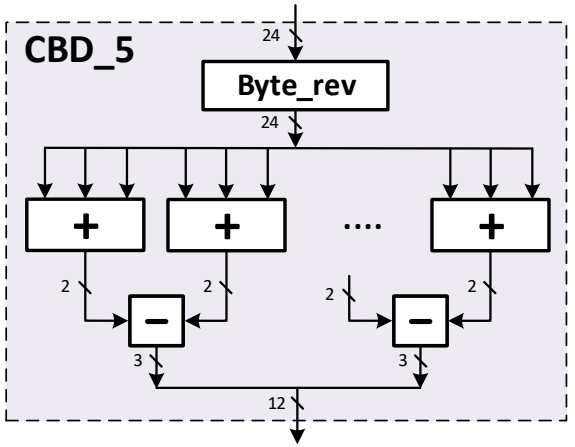


Linear-Feedback Shift Register (LFSR)



Multiplier Accumulator (MAC)





Centered Binomial Distribution (CBD) Sampler

NTRU Block Diagrams

Hardware Accelerator of NTRUEncrypt

Step 2:

calculating

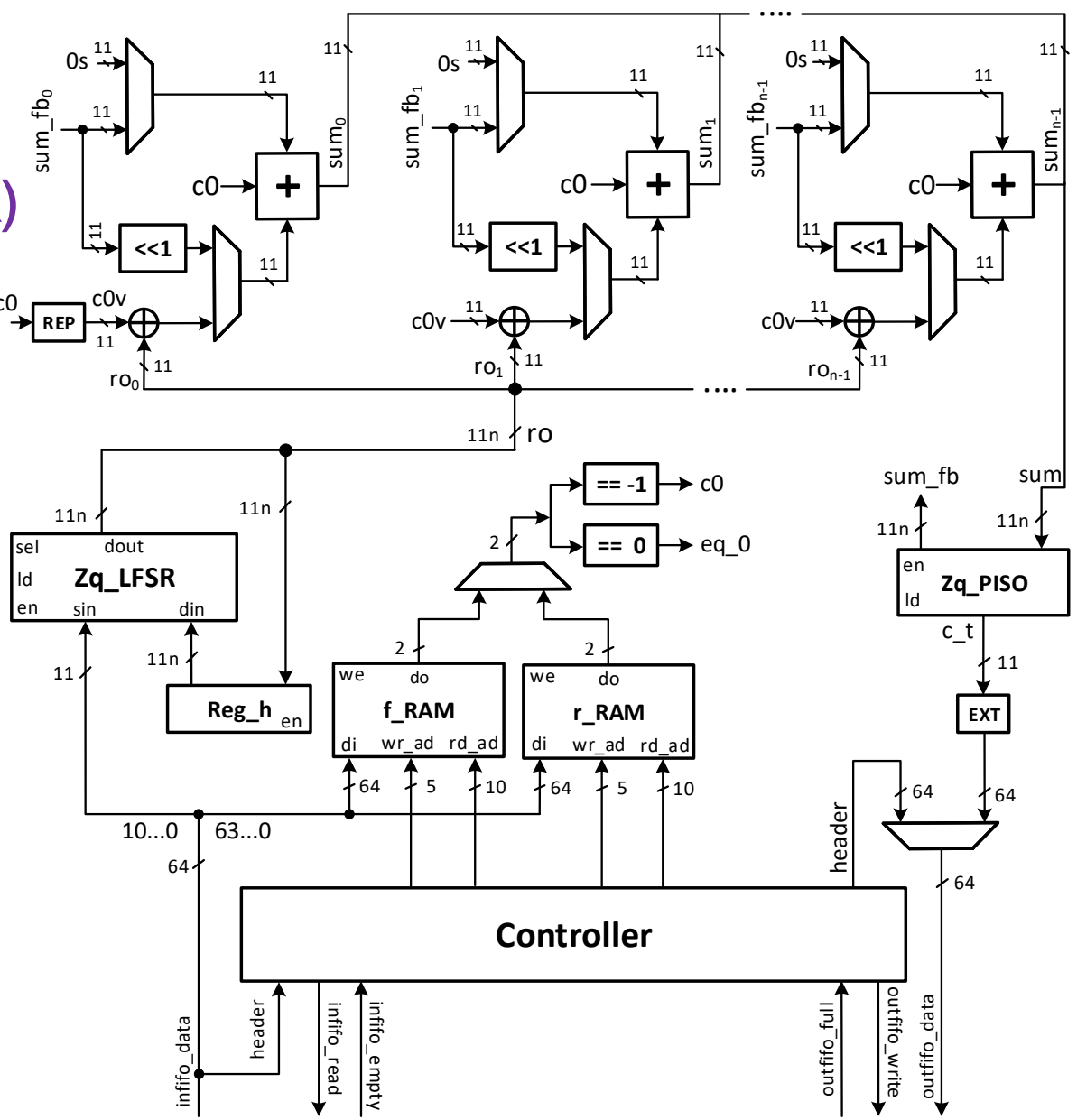
$sum \mathrel{+}= b_j \cdot x^j \cdot A(x)$

with $b_j \in \{ -1, 0, 1 \}$

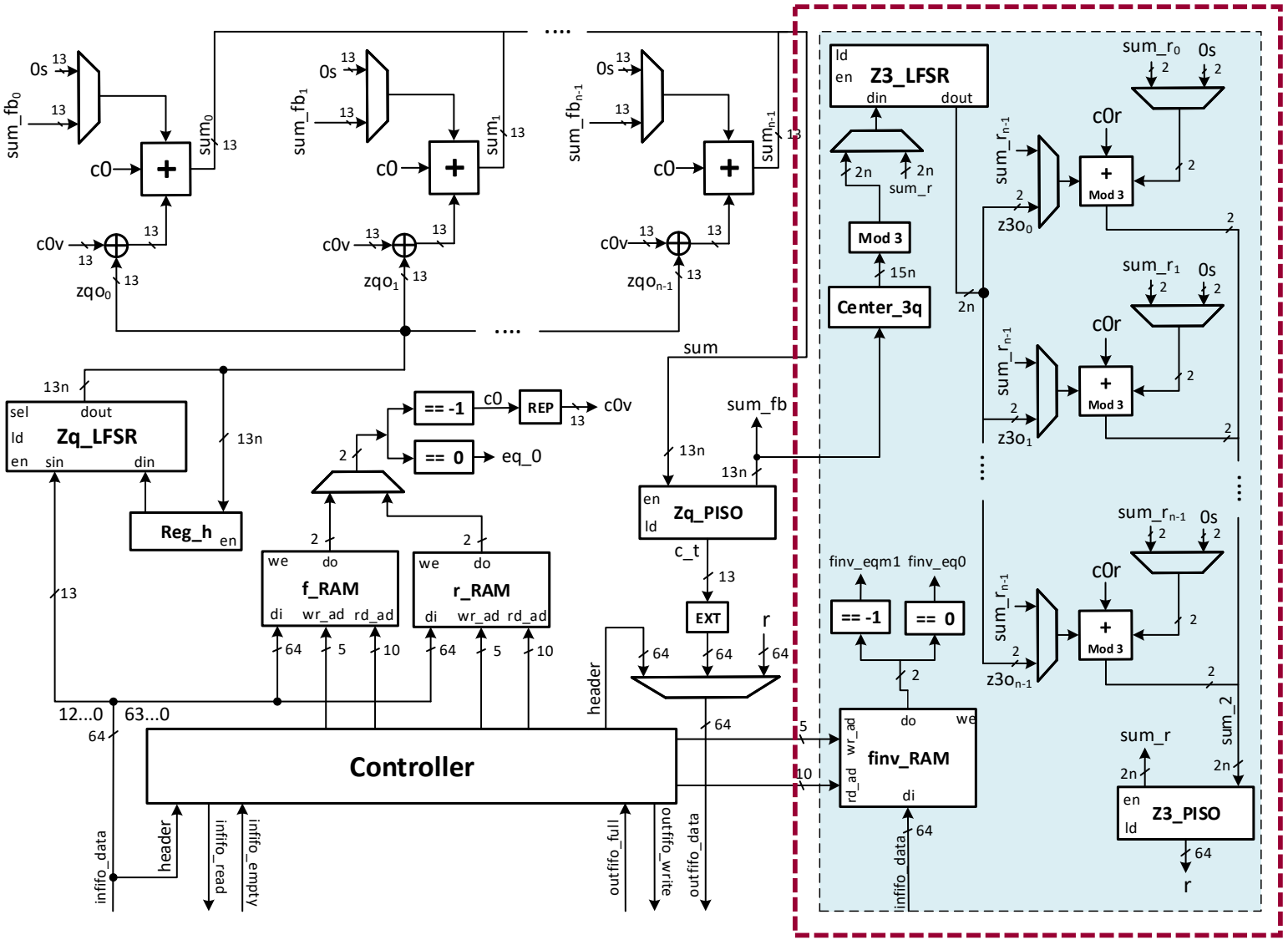
Step 1:

calculating

$x^j \cdot A(x)$



Additional Logic Required in NTRU-HRSS



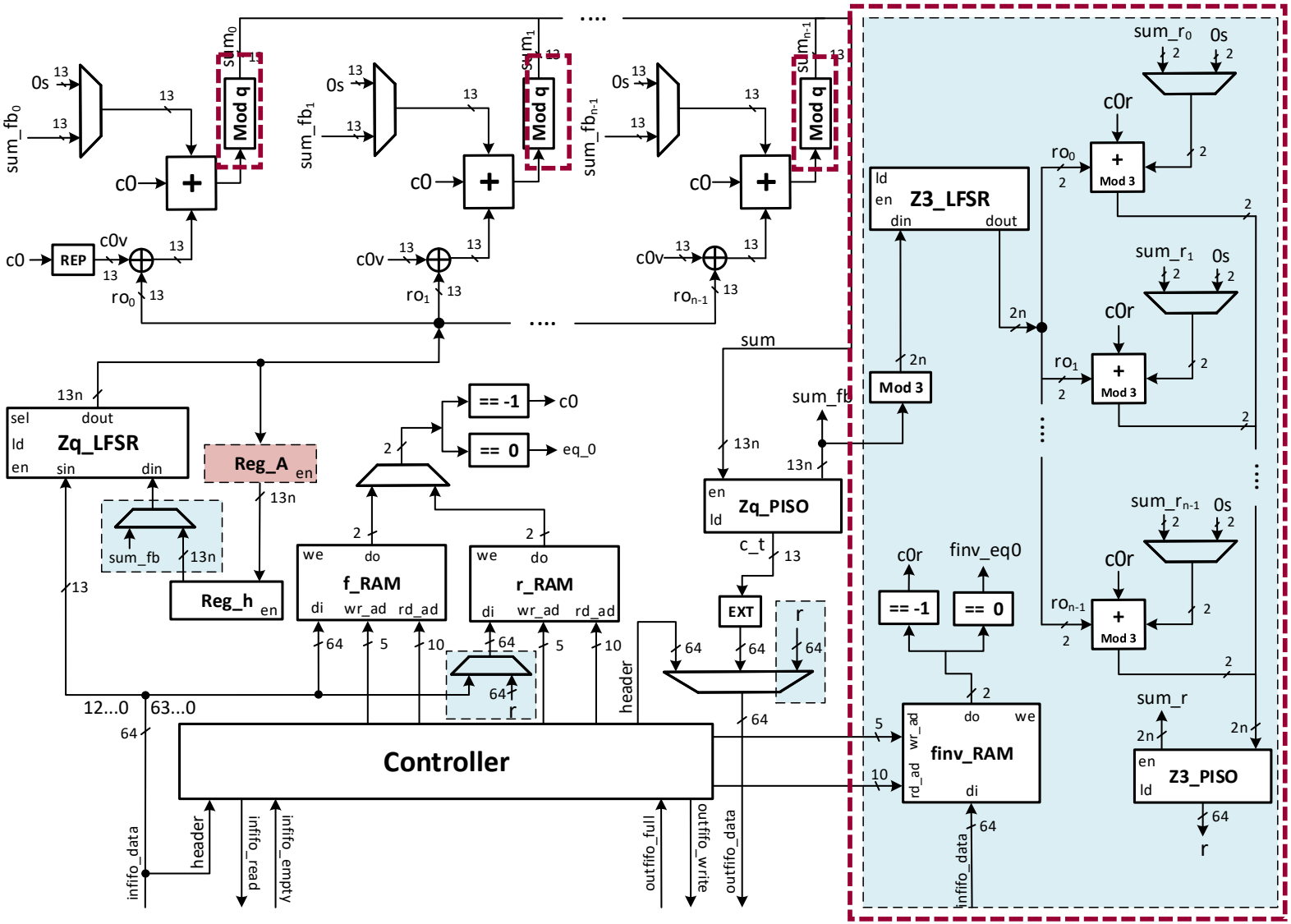
Overhead of Operations in S/3

26% logic
16% registers

Additional Logic Required in Streamlined NTRU Prime

Overhead of reduction mod prime q

38% logic
0% registers



Overhead of Operations in R/3

8% logic
9% registers

Additional Logic Required in NTRU LPRime

Overhead of reduction mod prime q

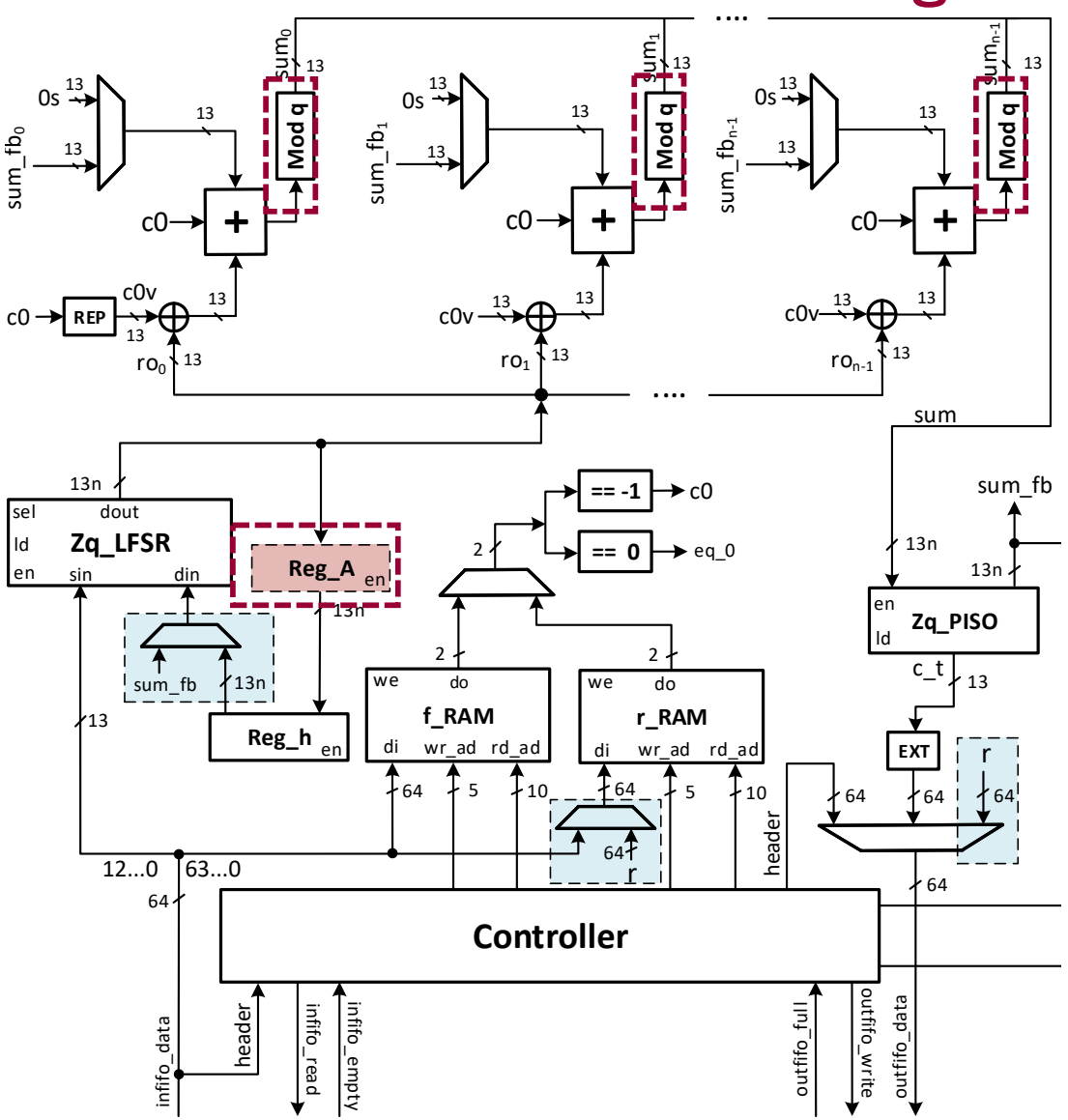
53% logic

0% registers

Overhead of an extra Register A

0% logic

25% registers



Backup 2

SW/HW Codesign: Step 1 Profiling

Assumptions:

- One core of the ARM Cortex-A53
- AXI Timer (in FPGA fabric) measuring time in clock cycles of the 200 MHz clock

Results:

- absolute execution time
- percentage execution time
- number of calls to a given function

SW/HW Codesign: Step 2 SW/HW Partitioning

Top candidates for offloading to hardware

From profiling:

- Large percentage of the execution time
- Small number of function calls

From manual analysis of the code:

- Small size of inputs and outputs
- Potential for combining with neighboring functions

From knowledge of operations and concurrent computing:

- High potential for parallelization

SW/HW Codesign: Step 3 Accelerator Design

Target: Minimum Execution Time

Hardware:

- Register-Transfer Level methodology with VHDL or Verilog
  - ☆ Block diagram of the Datapath
  - ☆ Algorithmic State Machine (ASM) chart of the Controller

Software:

- Input/Output transfers
- Transfer of control between the processor and the accelerator

Features of Investigated Round 2 Algorithms

Feature	FrodoKEM	Round5	Saber
Underlying Problem	LWE: Learning with Errors	[R]LWR : [Ring] Learning With Rounding	Module-LWR : Module Learning with Rounding
Underlying Encryption Scheme	FrodoPKE	r5_cpa_pke	Saber.PKE
Auxiliary Functions	SHAKE128 SHAKE256	SHAKE128 SHAKE256	SHAKE128 SHAKE256 SHA3-256 SHA3-512
Decryption Failures	Yes	Yes	Yes

Features of Investigated Round 2 Algorithms

Feature	NTRU-HPS	NTRU-HRSS	Streamlined NTRU Prime	NTRU LPRime
Underlying Problem	Shortest Vector Problem (SVP)			
Underlying Encryption Scheme	DPKE	DPKE	Streamlined NTRU Prime Core	NTRU LPRime Expand
Auxiliary Functions	SHA3-256	SHA3-256	SHA3-512	SHA3-512
Decryption Failures	No	No	No	No

Features of Investigated Round 2 Algorithms

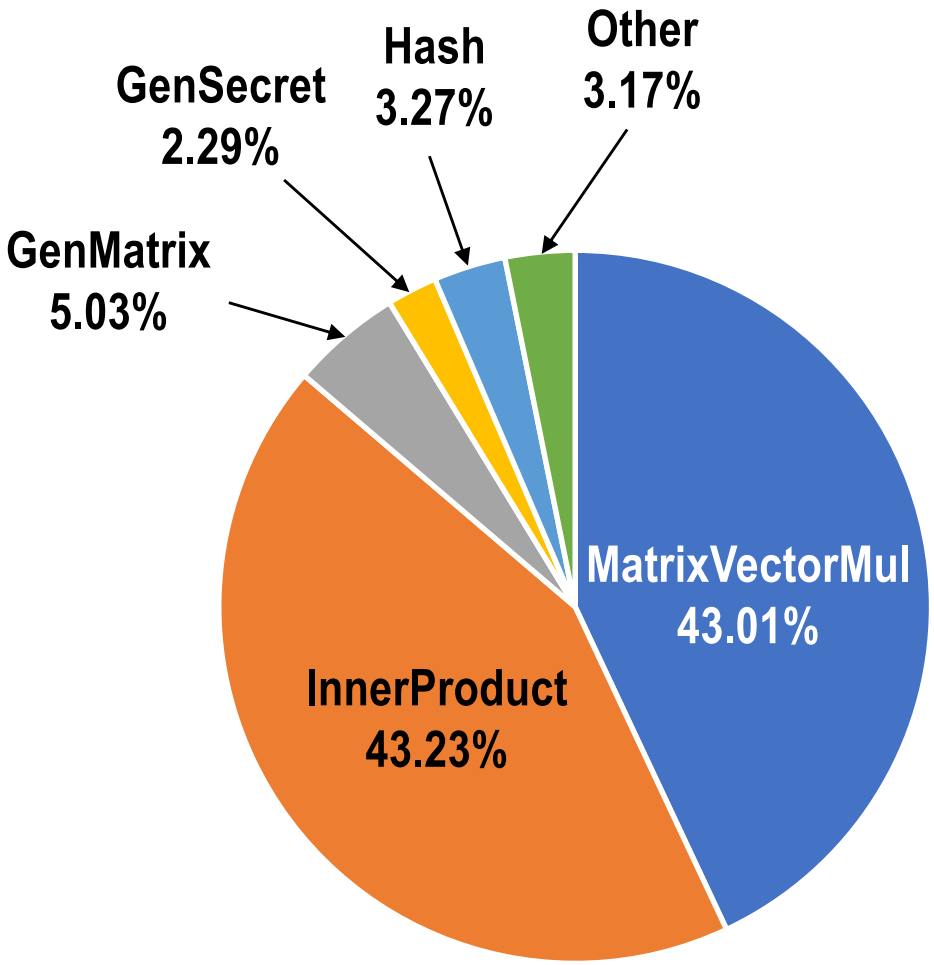
Feature	FrodoKEM	Round5 (RLWR)	Saber
Major Objects	Matrices of elements of Z_q	Polynomials with coefficients in Z_q and Z_3	Matrices & vectors of polynomials with coefficients in Z_q
Major Parameters	n, $\overline{n}$, $\overline{m}$: dimensions of matrices q: modulus (power of 2) B: bits per matrix entry	n: polynomial degree q: modulus (power of 2) p, t: rounding moduli	n: polynomial degree (power of 2) q: modulus (power of 2) *l* : polynomials per vector p, T: rounding moduli
Encapsulation			
Major Operations	2 **matrix-by-matrix**	2 poly mult	1 **matrix-by-vector** 1 vector-by-vector
Decapsulation			
Major Operations	3 **matrix-by-matrix**	3 poly mult	1 **matrix-by-vector** 2 vector-by-vector

Features of Investigated Round 2 Algorithms

Feature	NTRU-HPS	NTRU-HRSS	Streamlined NTRU Prime	NTRU LPRime
Major Objects	Polynomials with coefficients in Z_q and Z_3			
Major Parameters	n: polynomial degree (prime) q: modulus (power of 2)		n: polynomial degree (prime) **q: modulus (prime)**	
Encapsulation				
Major Operations (Poly Mults in)	1 in R/q	1 in R/q	1 in R/q	**2 in R/q**
Decapsulation				
Major Operations	1 in R/q 1 in S/q 1 in S/3	1 in R/q 1 in S/q 1 in S/3	2 in R/q 1 in S/3	3 in R/q

R/q: $Z_q[x]/P(x)$ S/q: $Z_q[x]/\Phi_n$ S/3: $Z_3[x]/\Phi_n$

Saber Decapsulation



93.56% offloaded to hardware

6.44% remaining in software