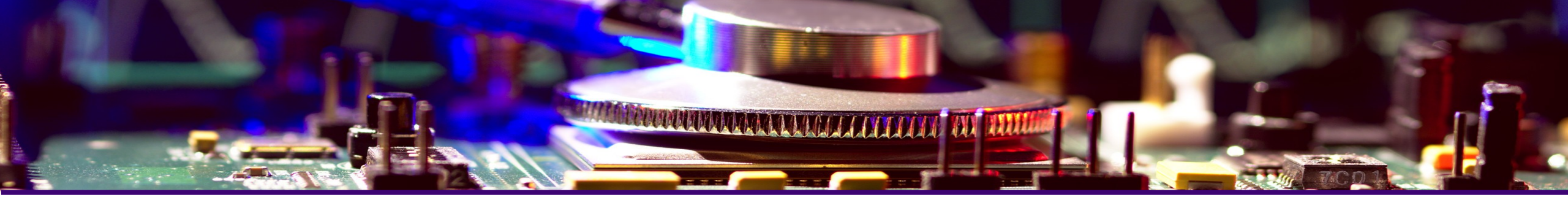
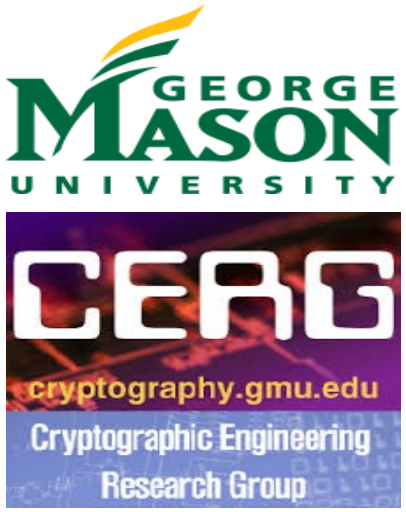


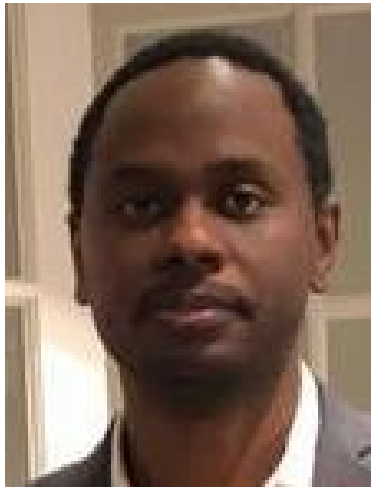
Lightweight Implementation of Saber Resistant Against Side-Channel Attacks

Abubakr Abdulgadir **Kamyar Mohajerani** **Viet Ba Dang**
Jens-Peter Kaps **Kris Gaj**

June 2021



Team

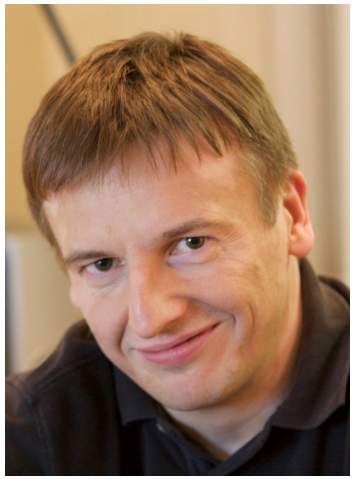


Bakry Abdulgadir

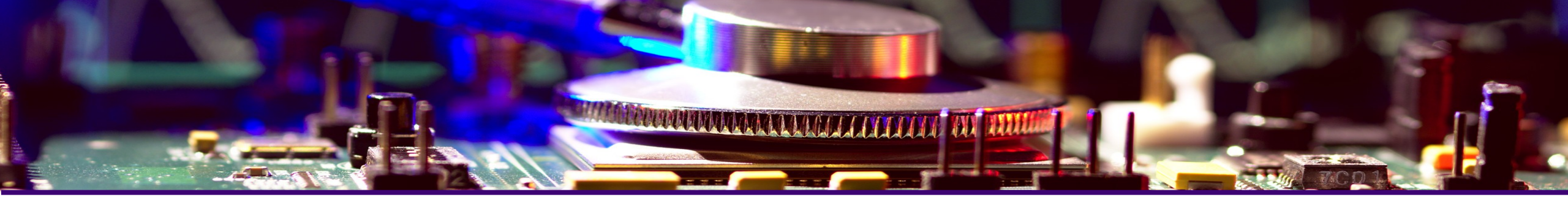
Kamyar Mohajerani

Viet Dang

Jens-Peter Kaps



Kris Gaj



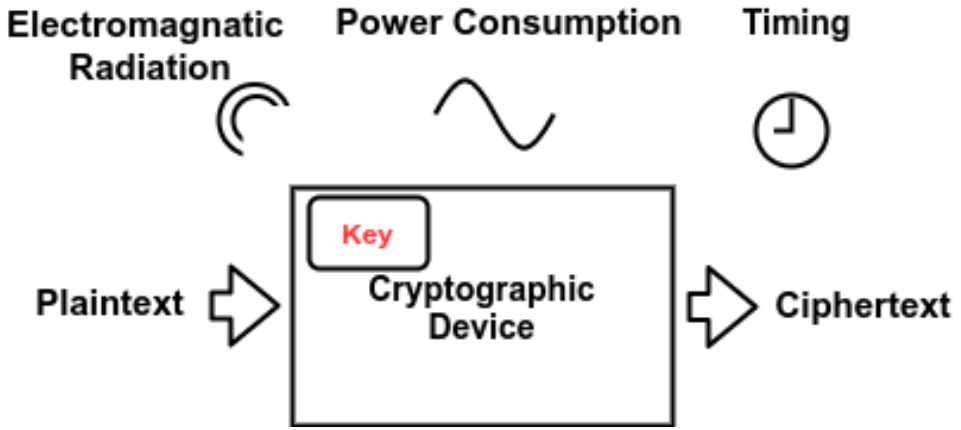
Introduction

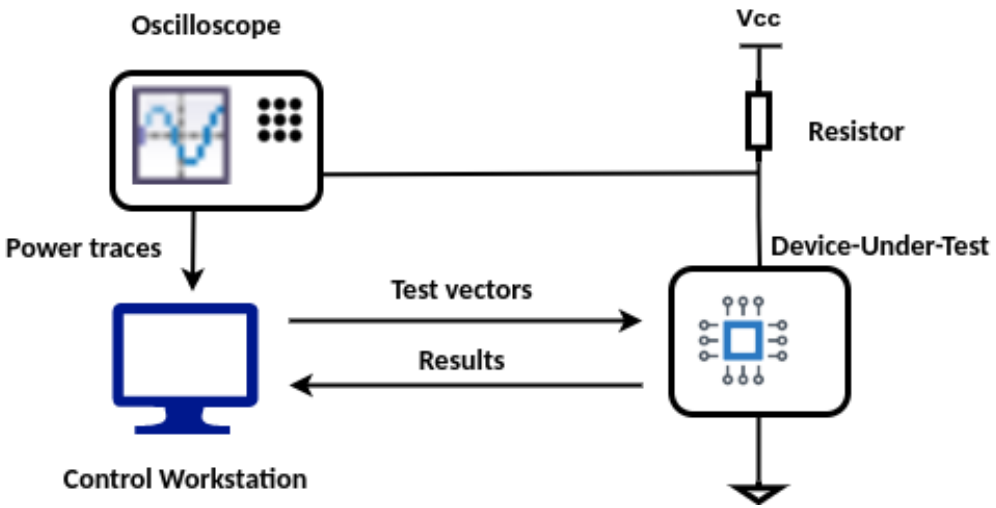
Introduction

- Saber
  - Key encapsulation mechanism (KEM)
  - Lattice-based scheme based on the hardness of Module Learning with Rounding
  - Efficient to mask in software
- This work: Investigation of hardware side-channel-resistant designs
  - Work-in-progress
  - Security of all units verified except centered binomial (CBD) sampler

Side-Channel Analysis (SCA)

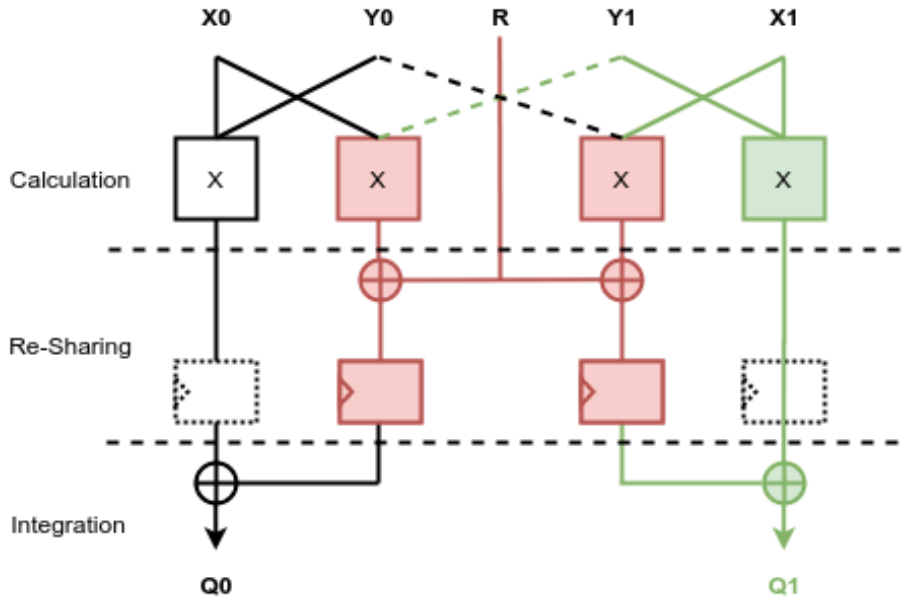
- Unintended outputs can reveal keys
- Countermeasures needed for many applications
  - Protection comes at a cost in resources and latency
  - Algorithm-dependent
- Ease of protection is desirable



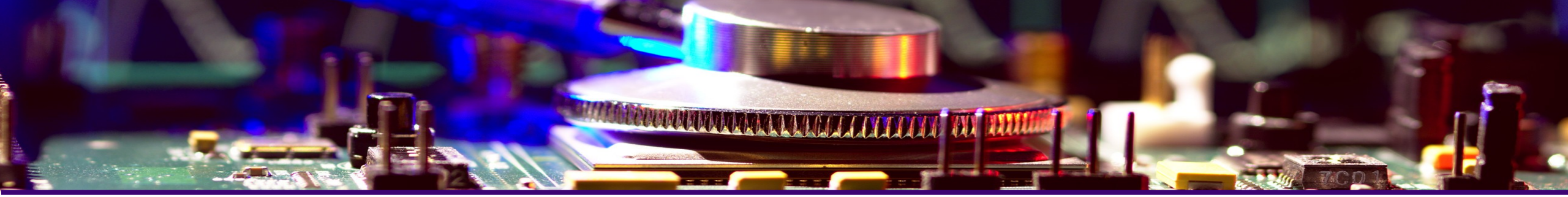


SCA Countermeasures

- Masking
  - Split sensitive data to shares
  - Compute on shares
- Sharing
  - $X = X_0$ xor X_1 → Boolean
  - $X = X_0 + X_1 \bmod q$ → Arithmetic
- Some masking variants are suitable for hardware
  - Threshold Implementation (TI)
  - Domain-Oriented Masking (DOM)
- Linear operations are trivially masked
- Non-linear operations needs more work!



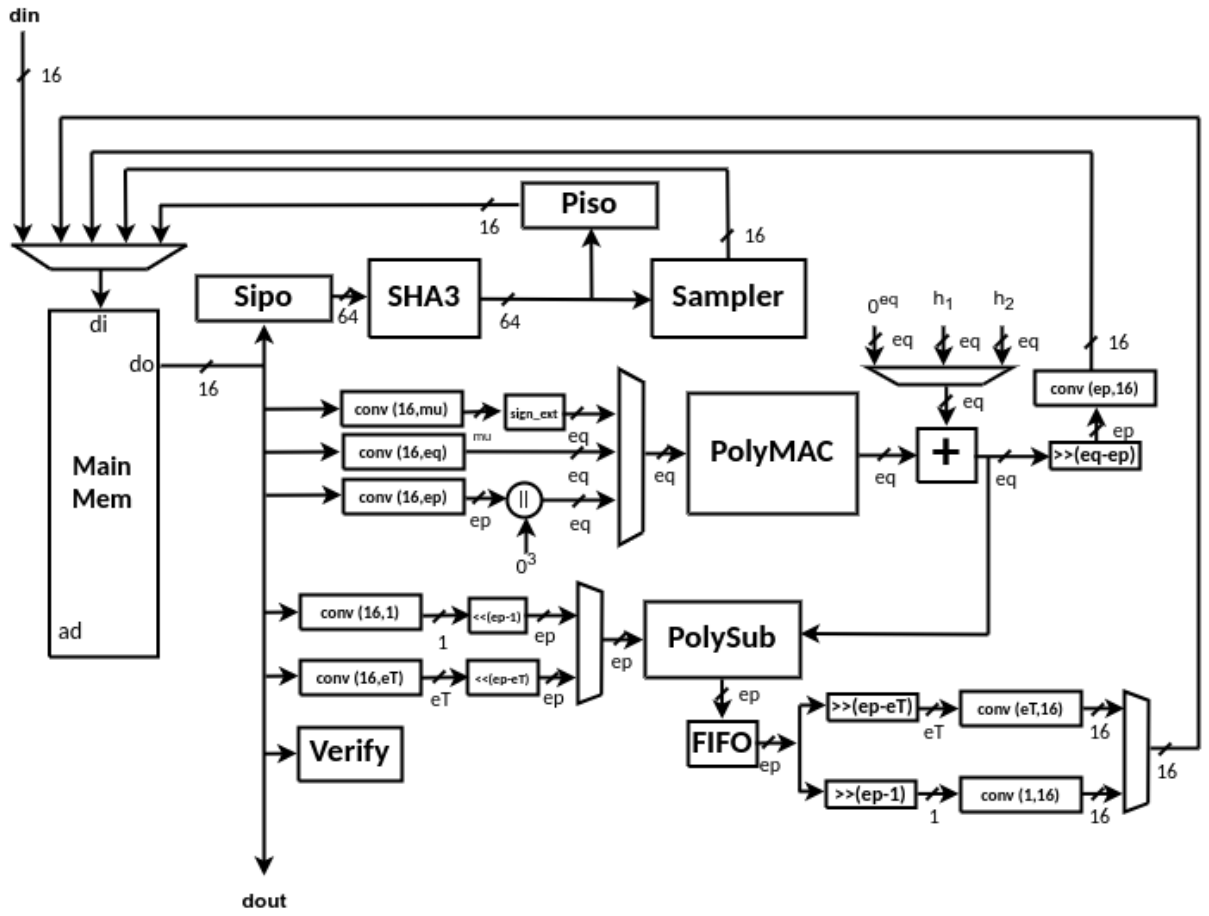
[Gross et al. 2016]



Methodology

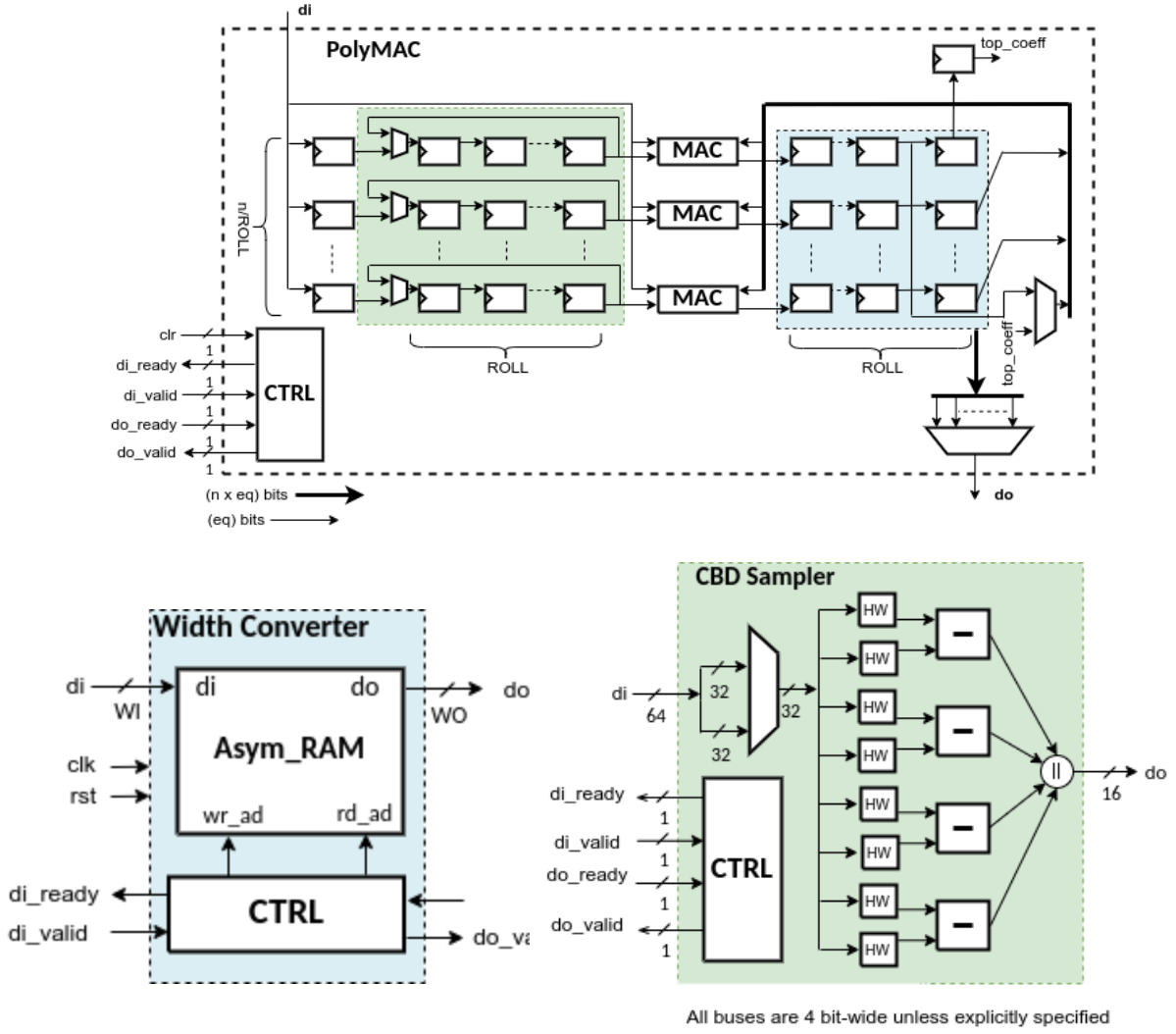
Saber Baseline Hardware Design

- Lightweight hardware design
- Register-Transfer Level using VHDL and Chisel
- Configurable multiplier
  - Schoolbook with configurable rolling factor
- Configurable SHA-3



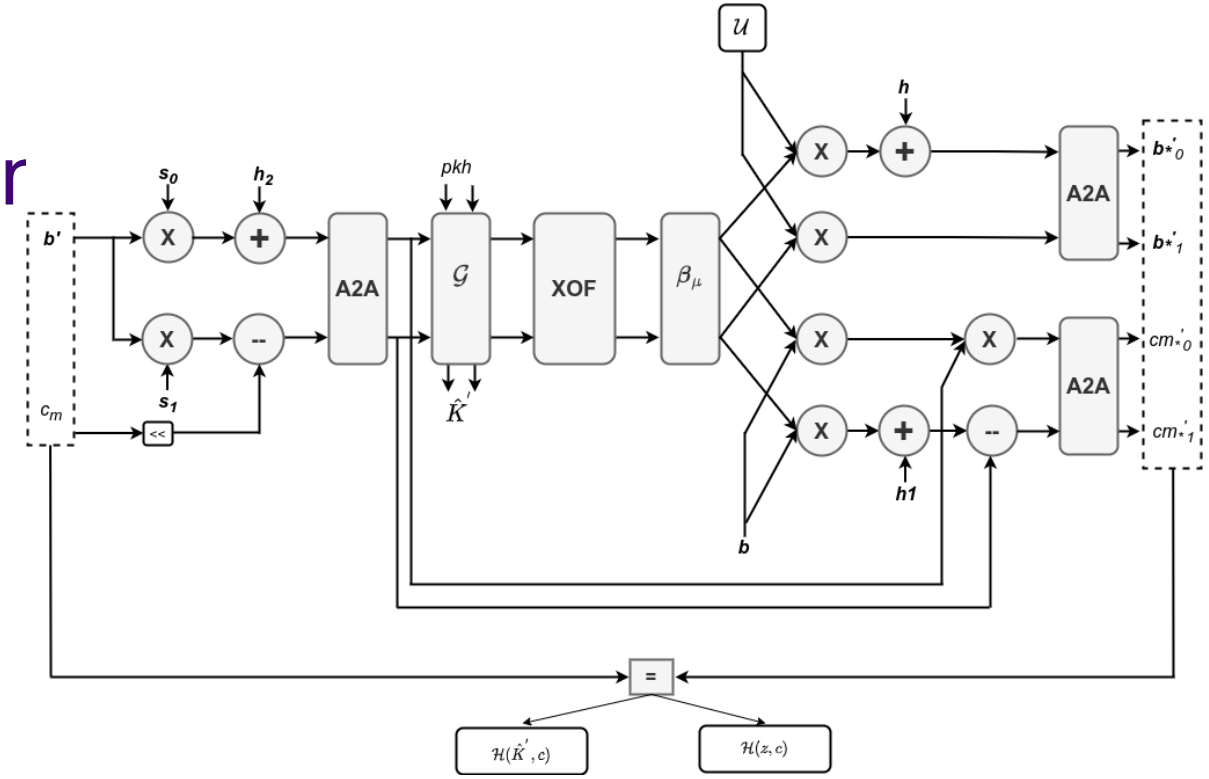
Saber Hardware Design Units

- Polynomial multiplier and accumulator (PolyMAC)
- SHA-3
- CBD Sampler
- Width Converters (packing/unpacking)



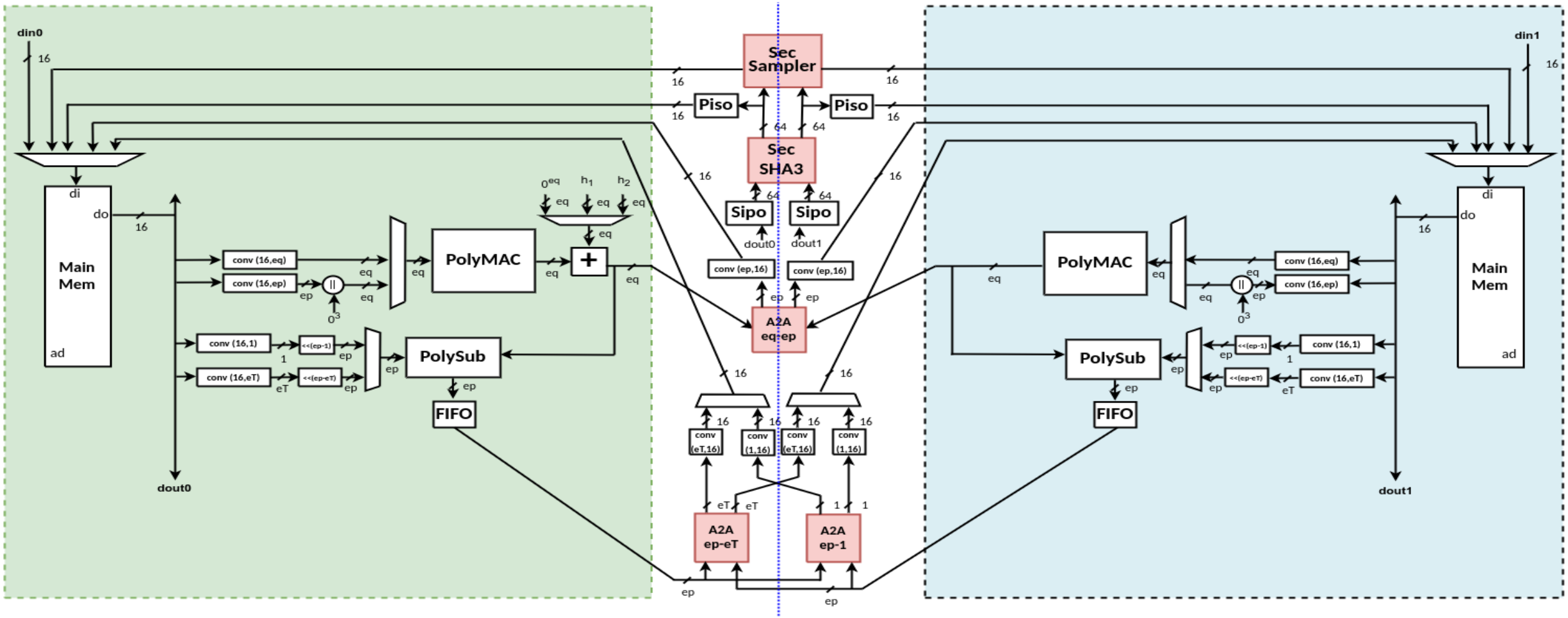
Masked Saber in Software

- Intermediates that involve private key need protectior
- Masking:
  - Arithmetic shares for polynomial arithmetic
  - Boolean in SHA-3
- For our design
  - Duplicate all linear logic
  - Re-design non-linear units



Masked Saber Data Flow
[Beirendonck et al. 2020 (ePrint), 2021 (journal)]

Masked Saber in hardware

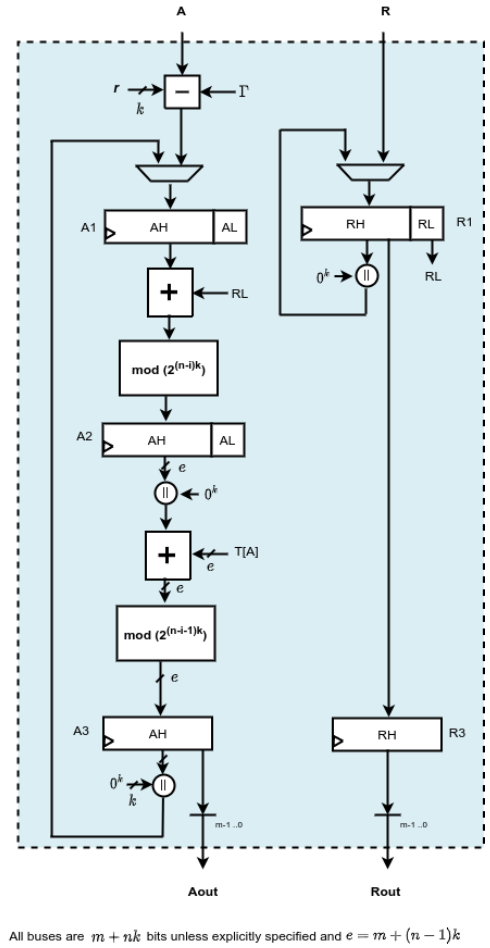


Masked hardware : Logic Shift

- Performs secure logical shift on Arithmetic shares (free in unprotected hardware!)
- Implements Arithmetic-to-Arithmetic (A2A) [Beirendonck et al]
- Input (arithmetic shares):

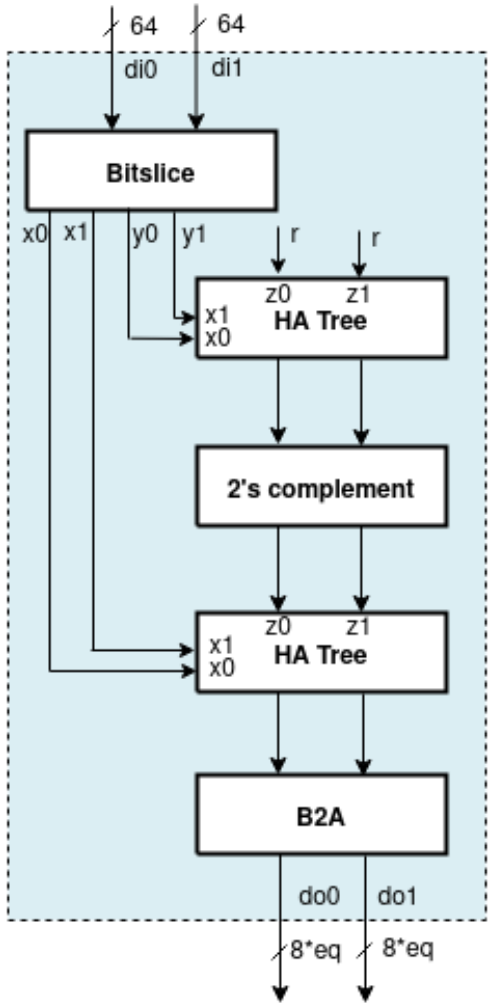
  $x = A + R \bmod 2^{m+k*n}$
- Output (arithmetic shares):

  $(x>>k*n) = A + R \bmod 2^{m}$



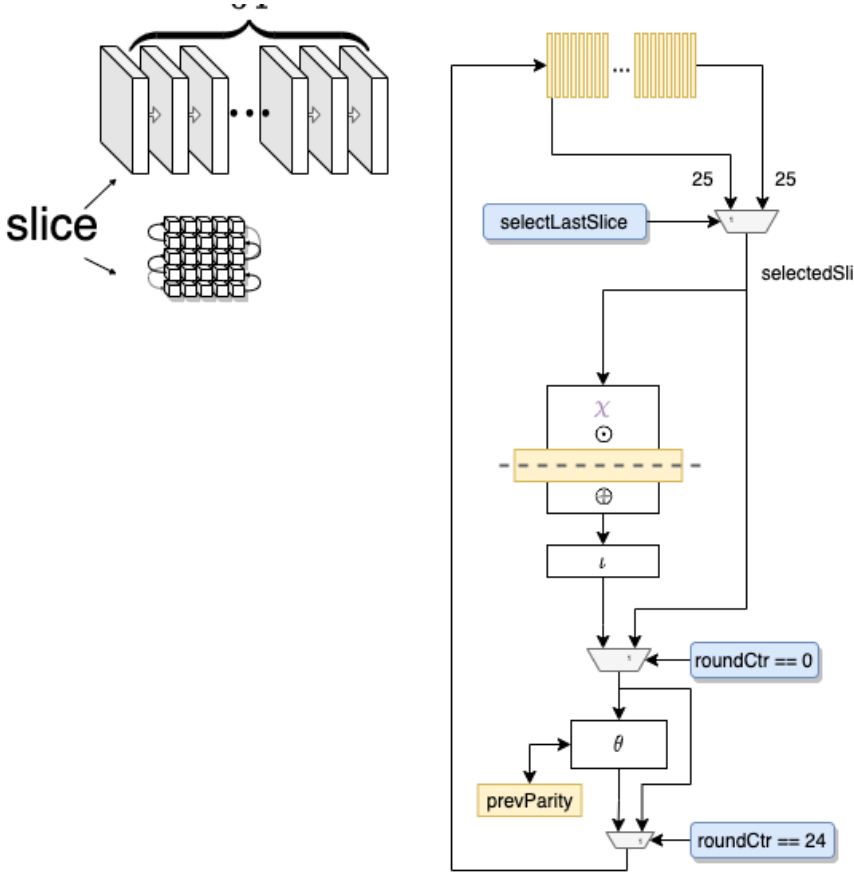
Masked hardware : Sampler

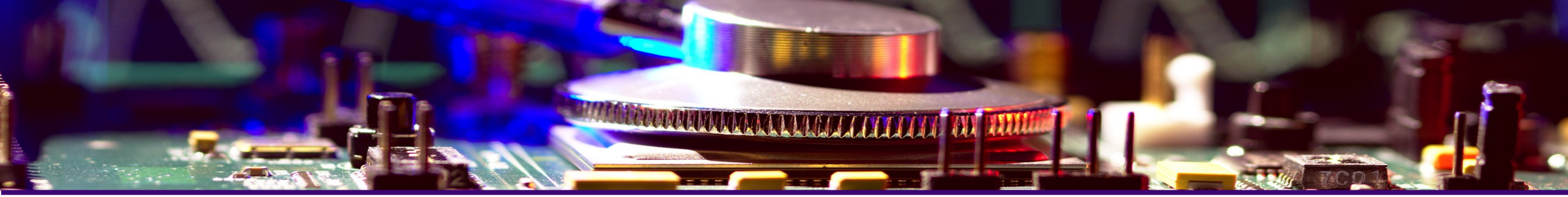
- Based on ideas from [Fritzmann et al 2021] which is based on [Schneider et al 2019]
- Calculate CBD sample as HW(x)-HW(y)
- Input:
  - Boolean shares from SHAKE128 (uniform samples)
- Output:
  - Arithmetic shares (CBD samples)
- Adder tree consists of half adders to calculate HW(x) + z



Masked hardware : SHA3

- Configurable SHA3 unit
  - SHA3-256, SHA3-512, SHAKE128
  - Area-performance trade-off
  - Configurable IO width
- Protected using DOM following [Arribas et al 2017]

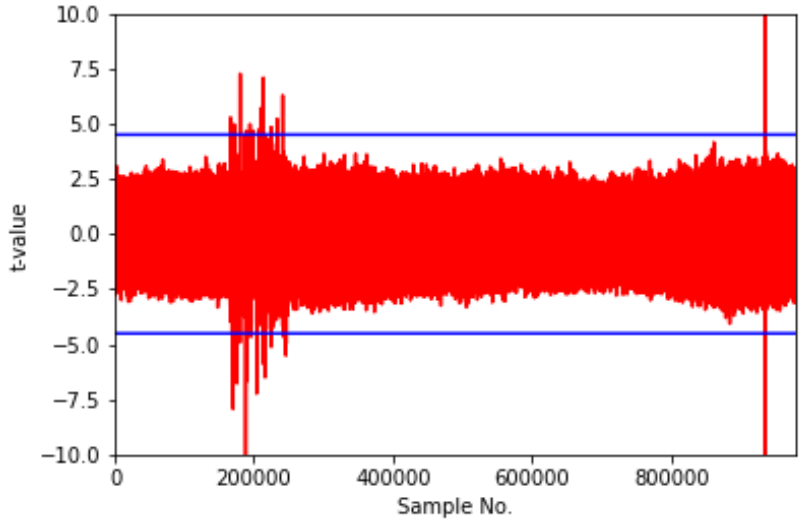




Results

Leakage Assessment

- The entire design passes TVLA except when CBD sampler is running
- Last spike related to the comparison of ciphertext and re-encrypted ciphertext hashes (Expected)



Saber Estimated Results (1)

Algorithm	Type	Platform	Protection	Freq	Resource Utilization				
				MHz	LUTs	FFs	Slices	DSPs	BRAMs
Saber-r8 [TW]	HW	FPGA-Artix7	unprotected	100	6,713	7,363	2,631	32	0
				100					
Saber-r8-masked [TW]	HW		Protected	100	19,783	21,576	7,143	64	0
Saber [32]	HW	FPGA-UltraScale+	unprotected	100	34,886	9,858	-	85	6.0
			unprotected	100					
Saber [5]	SW	ARM Cortex-M4	unprotected	168	-	-	-	-	-
	SW		Protected	168					
Saber [11]	SW/HW	RISC-V+ Acc.	unprotected	62.5	20,697	11,833	6,852	13	36.5
				62.5					
	SW/HW		Protected	58.8	29,889	17,152	9,641	13	52.5

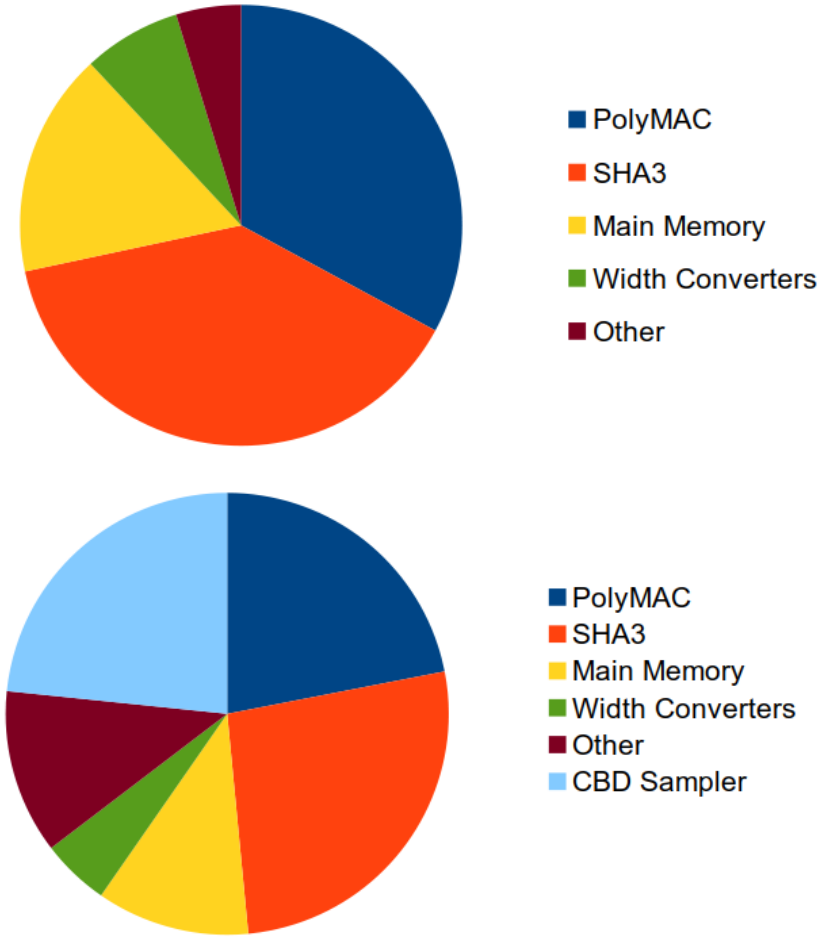
Algorithm	Type	Platform	Protection	Freq	Latency			
				MHz	Operation	Cycles	us	ratio
Saber-r8 [TW]	HW	FPGA-Artix7	unprotected	100	Encaps	46,705	467.1	-
				100	Decaps	52,758	527.6	1.00
Saber-r8-masked [TW]	HW		Protected	100	Decaps	73,851	738.5	1.40
Saber [32]	HW	FPGA-UltraScale+	unprotected	100	Encaps	1,396	14.0	-
			unprotected	100	Decaps	1,684	16.8	-
Saber [5]	SW	ARM Cortex-M4	unprotected	168	Decaps	1,123,280	6,686.2	1.00
	SW		Protected	168	Decaps	2,833,348	16,865.2	2.52
Saber [11]	SW/HW	RISC-V+ Acc.	unprotected	62.5	Encaps	308,430	4,934.9	-
				62.5	Decaps	347,323	5,557.2	1.00
	SW/HW		Protected	58.8	Decaps	905,395	15,397.9	2.77

Saber Estimated Results (2)

- Masked design uses about 3x #LUTs and 2x #DSPs
- Latency is 1.4x higher

Saber Hardware Units

- PolyMAC + SHA3 + RAM use 88% of the LUTs in baseline design and 60% of masked design
- Sampler size is significant in masked design
- Width converters use 7% and 5% of LUTs respectively.

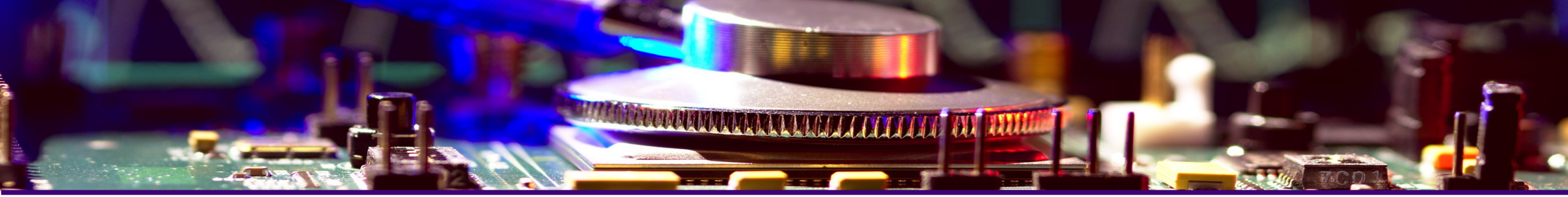


Conclusions

- Expected cost of protection
  - 3x #LUTs and 2x #DSP units
  - 1.4x the latency
- Protected HW estimated to be 23x faster than previous masked software (ARM-Cortex M4) and 21x faster than SW/HW- co-design (based on RISC-V).
- To improve results concentrate on PolyMAC and SHA3 for unprotected and CBD sampler also in masked.

Future Work

- Finish CBD sampler protection
- Further reduction in resource utilization and performance improvement



Thank you for listening!

?