RUB

RUHR-UNIVERSITÄT BOCHUM

On the Simplicity of Converting Leakages from Multivariate to Univariate

21. Aug. 2013

Amir Moradi, Oliver Mischke

Embedded Security Group + Hardware Security Group
Ruhr University Bochum, Germany

hgi Horst Görtz Institute for IT-Security

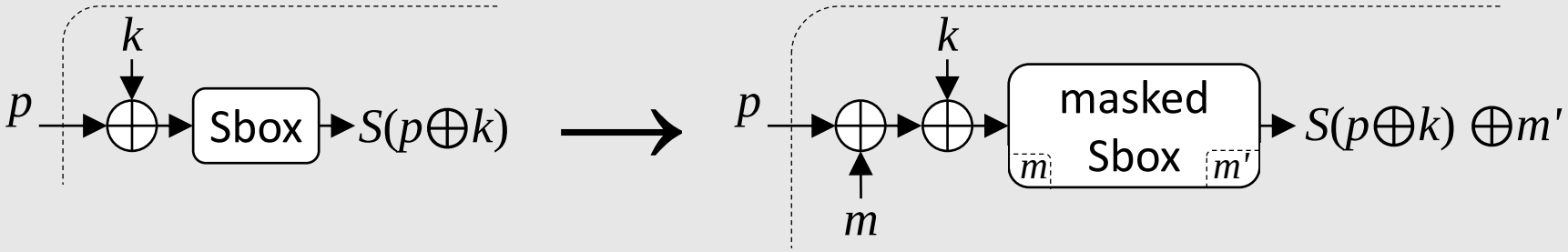
EUROPÄISCHE UNION
Investition in unsere Zukunft
Europäischer Fonds
für regionale Entwicklung

Outline

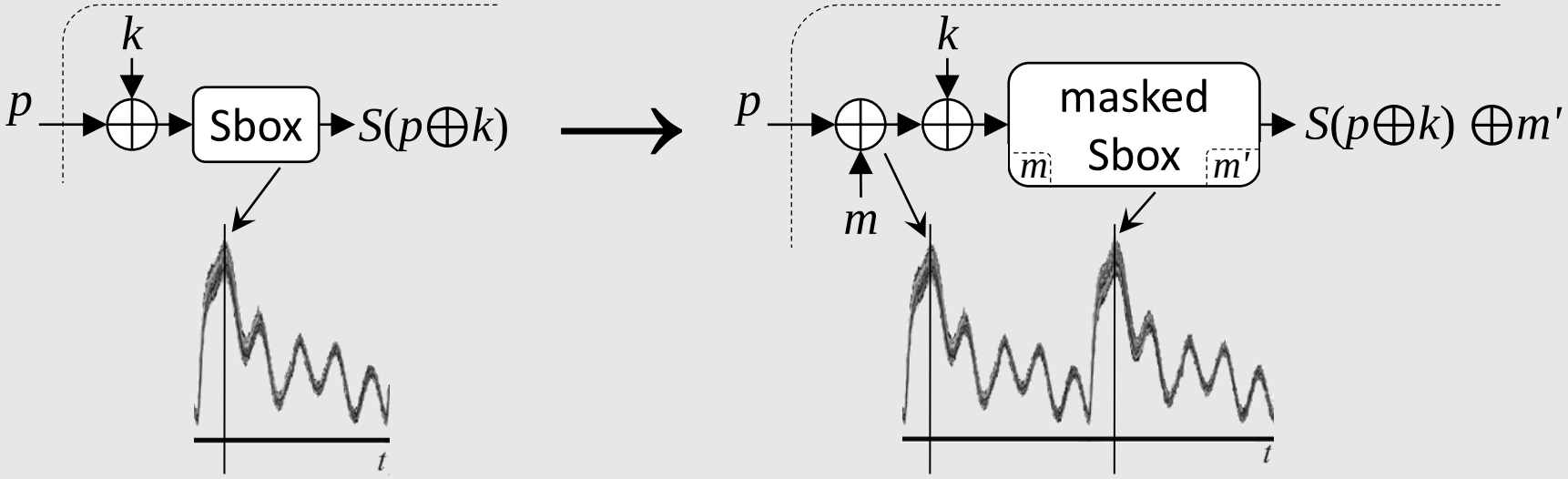
- Definitions, Masking, etc.
- Target masking scheme
- The story behind our findings
- Practical issues

Masking

- Well-known SCA countermeasure
- to make the SC leakages independent of expected intermediate values
- Randomness is required
- Let's consider the most common one, Boolean Masking



Univariate vs. Multivariate Attacks



DPA/CPA/MIA

bivariate MIA

combining: DPA/CPA

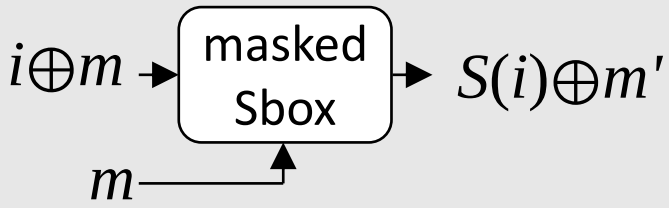
squaring: 2nd order univariate

multiply: 2nd order bivariate

addition: 1st order bivariate

Masking in Hardware

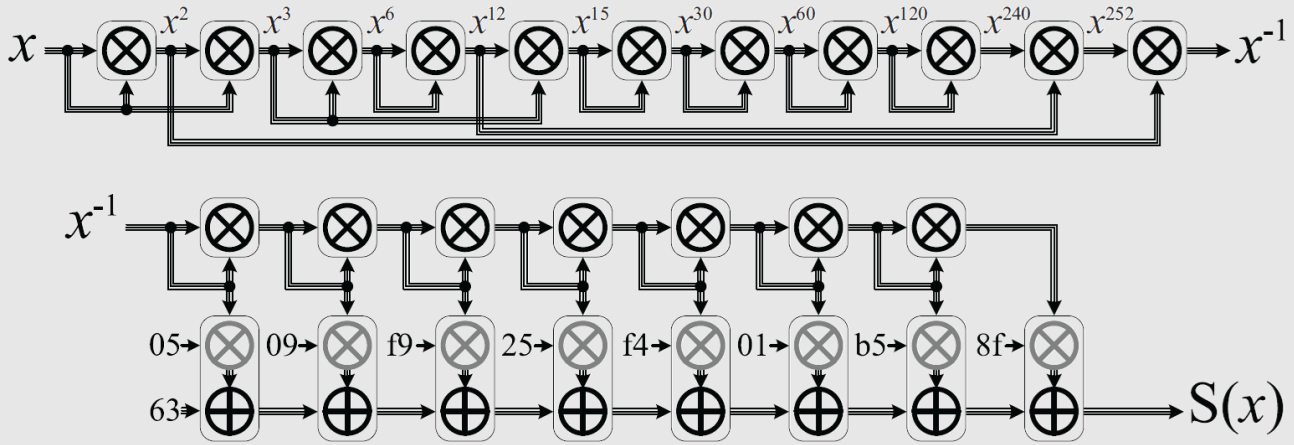
- Pre-computing the masked tables in software
  - Sequential operations, Time consuming, Low efficiency
- High efficiency is desired in HARDWARE
  - amongst the main reasons
- ad-hoc/heuristic schemes
- Processing the mask (m) and masked data ($i \oplus m$) simultaneously
  - joint distribution of SC leakages mainly because of GLITCHES
  - possible attacks
- Systematic schemes
  - Threshold Implementation, Security against 1st order attacks



Desired: security against univariate attacks of any order

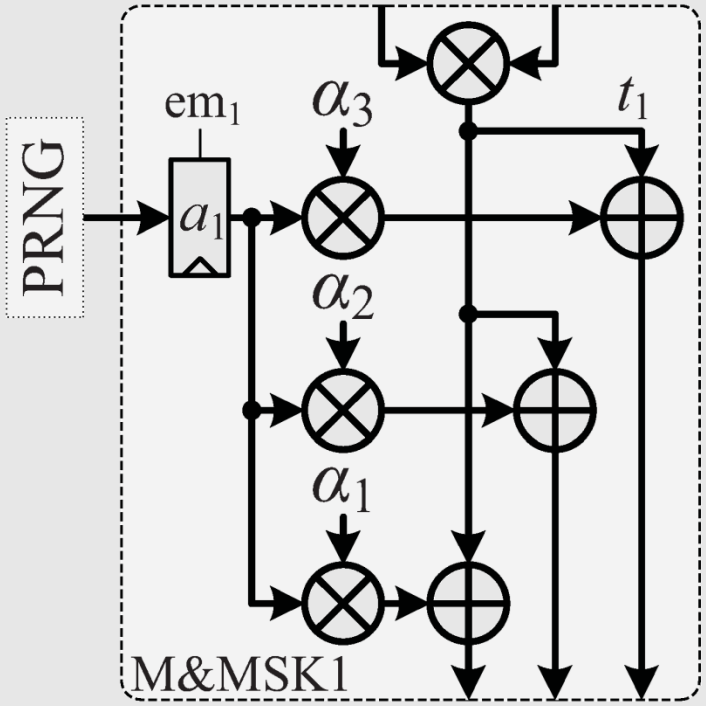
Target Scheme

- Prouff, Roche: *Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols*. CHES 2011.
- Multi-party computation + Shamir's secret sharing
- Basic GF(2^8) operations, e.g., addition is easy
  - Multiplication needs more effort
- An Sbox computation

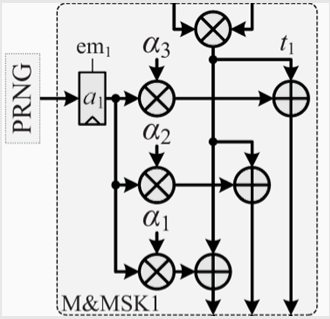


- Our goal
  - Hardware implementation using minimum settings
  - Using a Virtex-5 FPGA (SASEBO-GII)

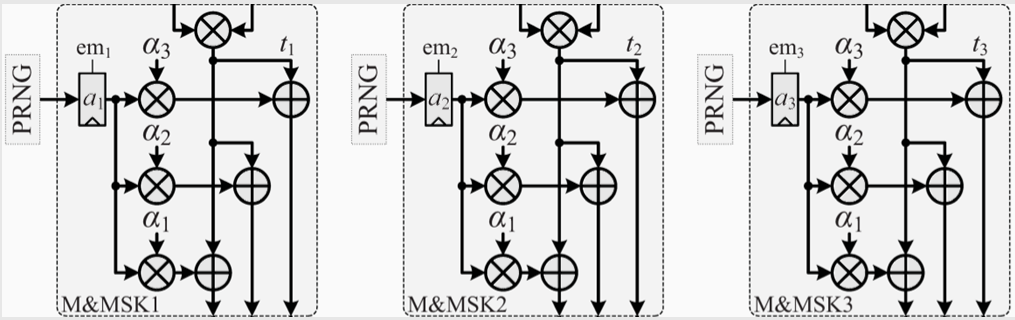
Target Scheme - Design



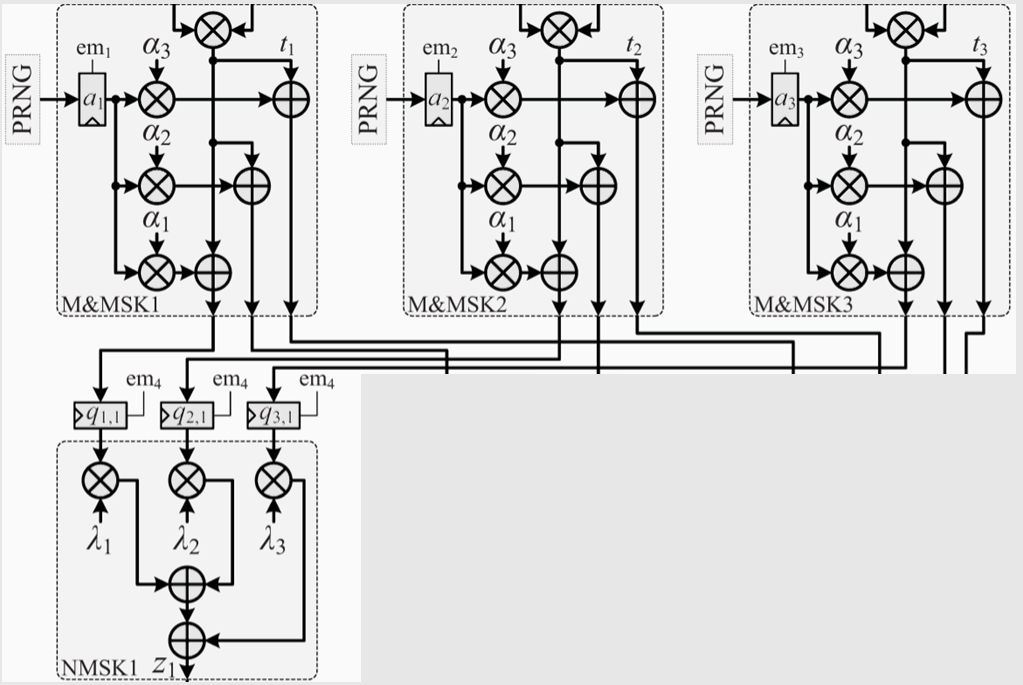
Target Scheme - Design



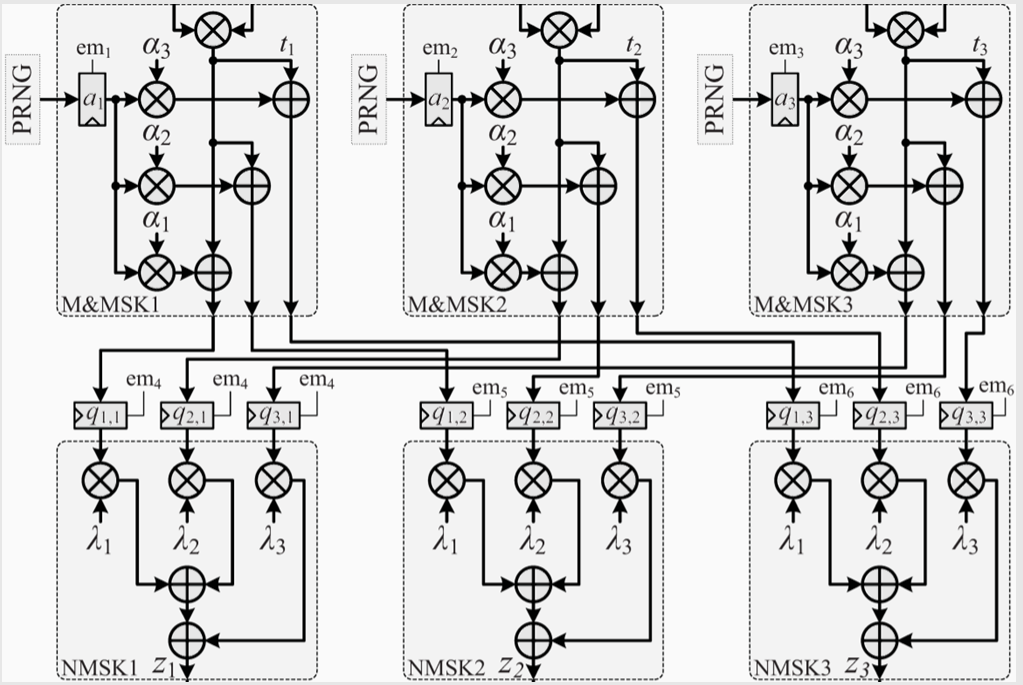
Target Scheme - Design



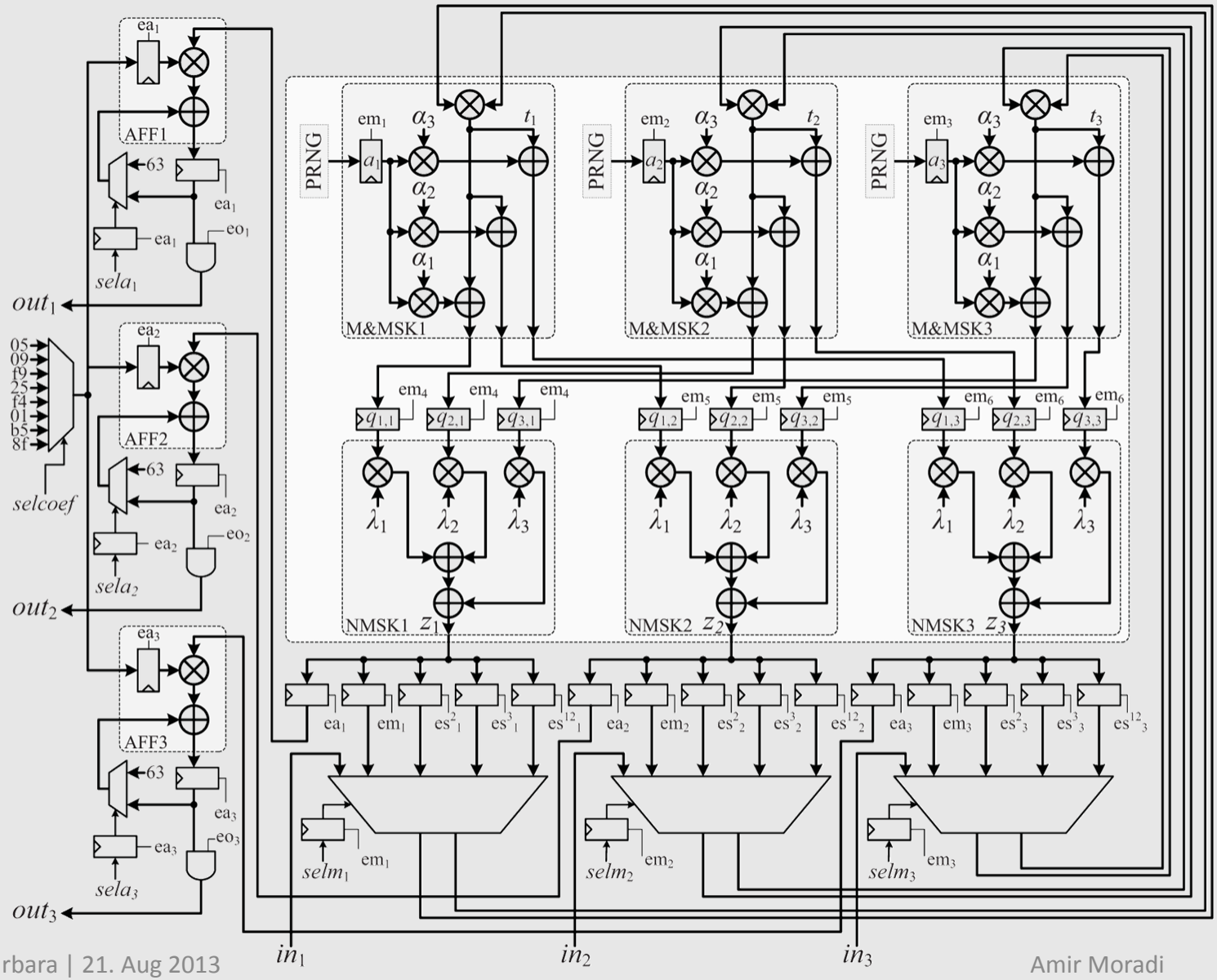
Target Scheme - Design



Target Scheme - Design



Target Scheme - Design



Target Scheme - Performance

- 66 clock cycles for **Inversion**, 66 clock cycles for **Affine**
  - One Sbox in 132 clock cycles
- One full SubBytes in 132 × 16 = 2112 clock cycles
- One full MixColumns + AddRoundKey in 12 × 16 = 192 clock cycles

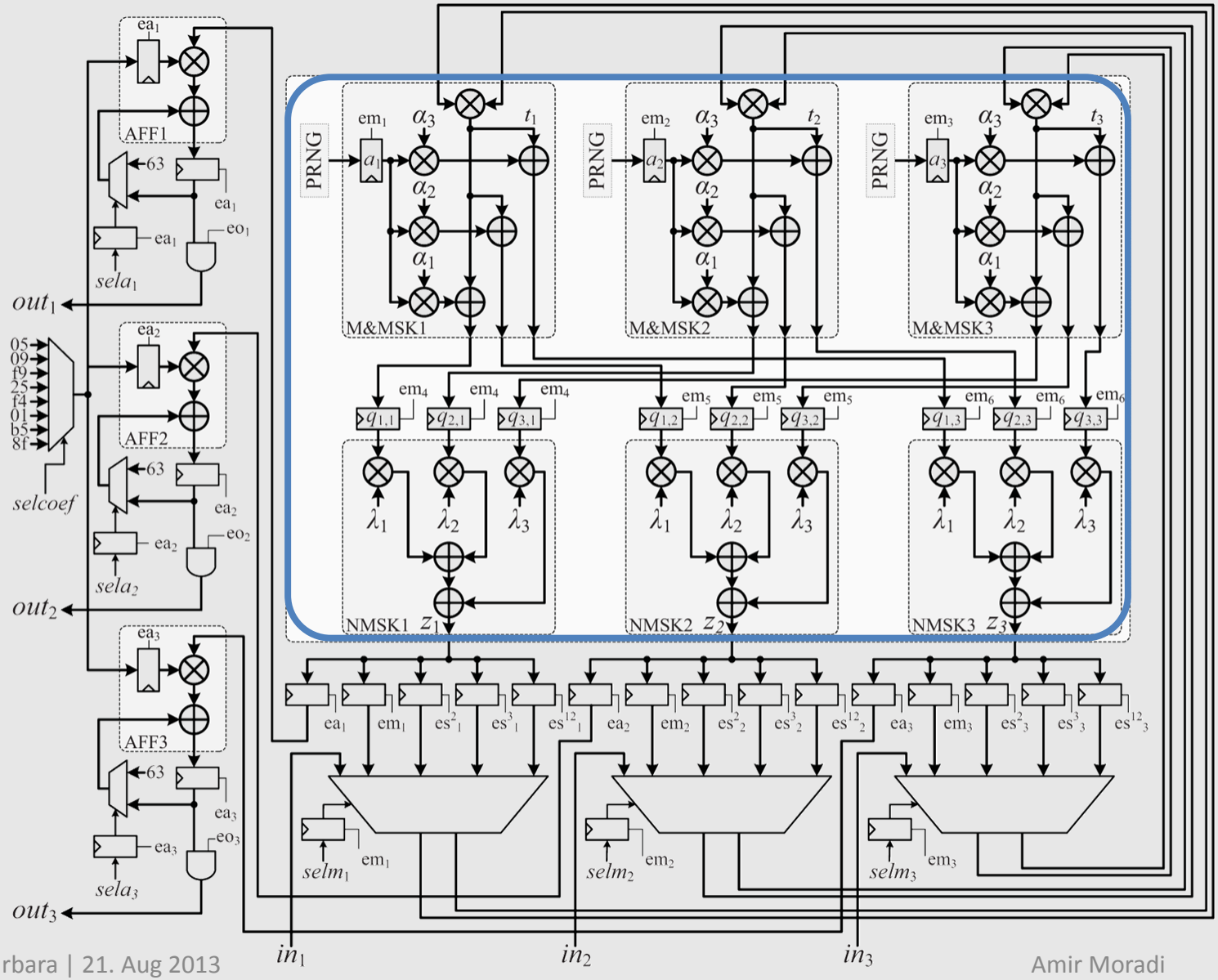
Design	FF		LUT		Slice		SB CLK	MC+ARK CLK	Encryption CLK
	#	%	#	%	#	%			
1 SB MC	315	1%	1387	5%	859	12%	2112	192	22 896
16 SB MC	4275	15%	21 328	74%	no fit		132	12	1431

- Hard to convince the industry sector?
- getting close to software?
- Gaining univariate resistance at what price?

Target Scheme - Evaluation

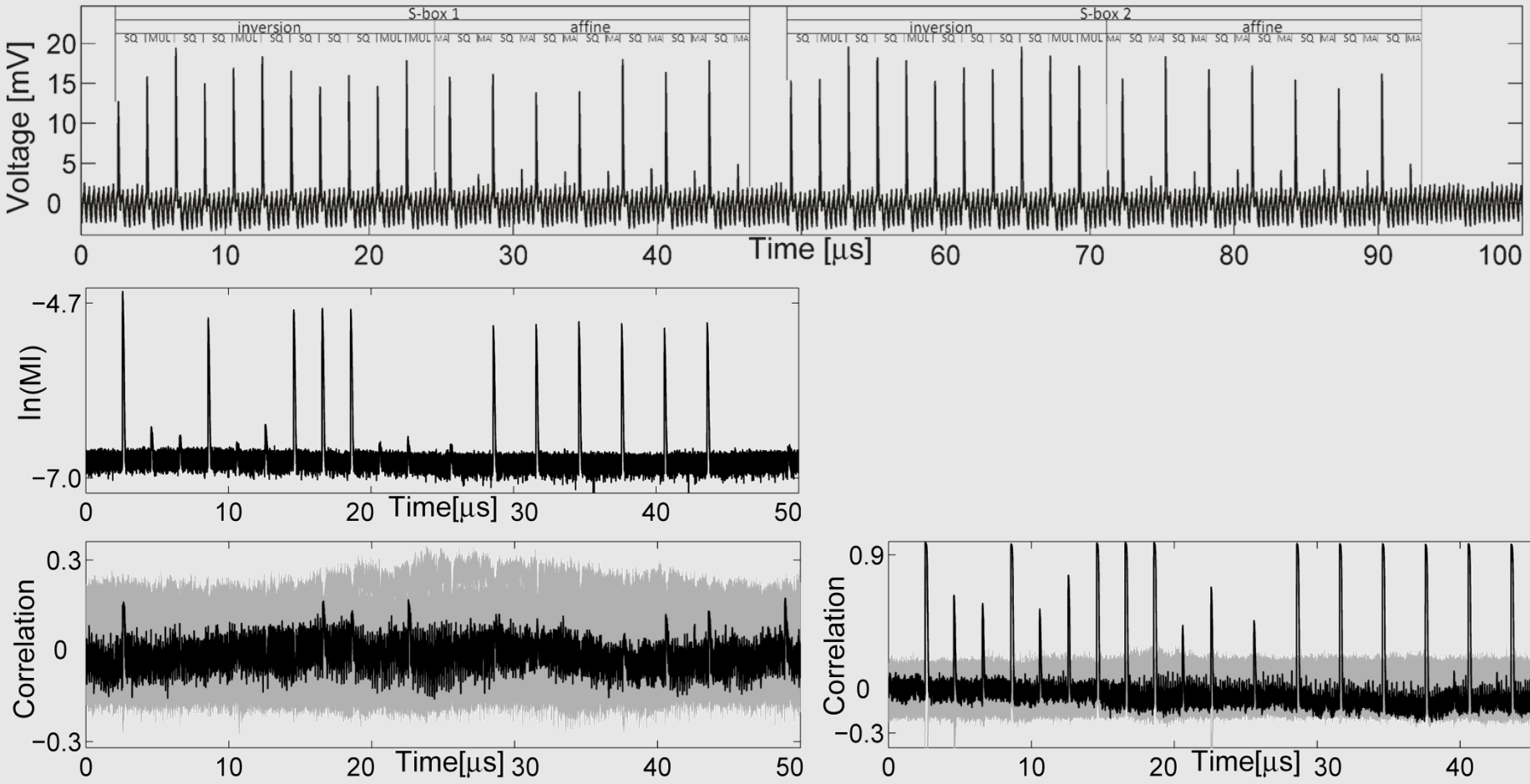
- A variant by processing all three shares at the same time

Target Scheme - Evaluation



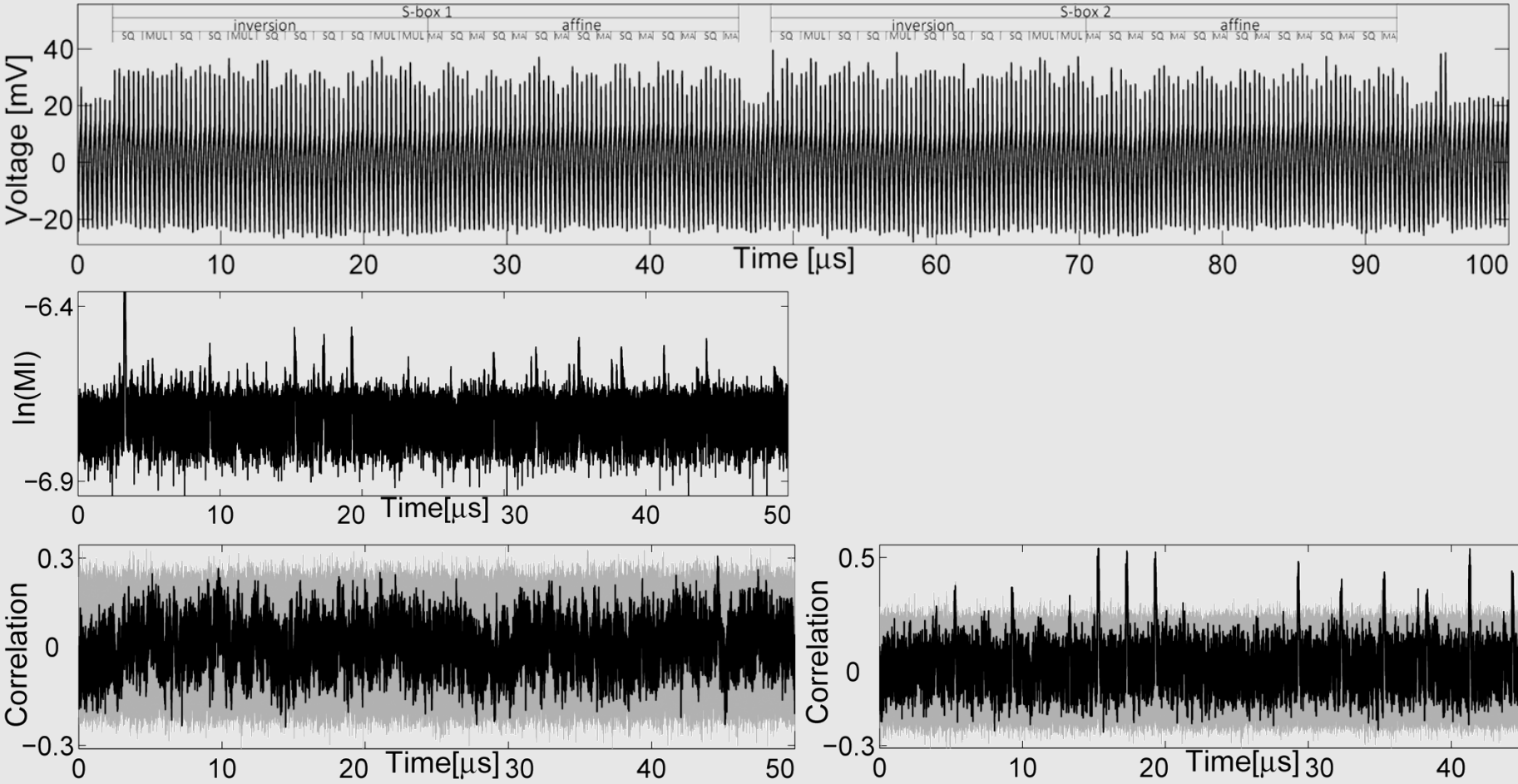
Target Scheme - Evaluation

- A variant by processing all three shares at the same time



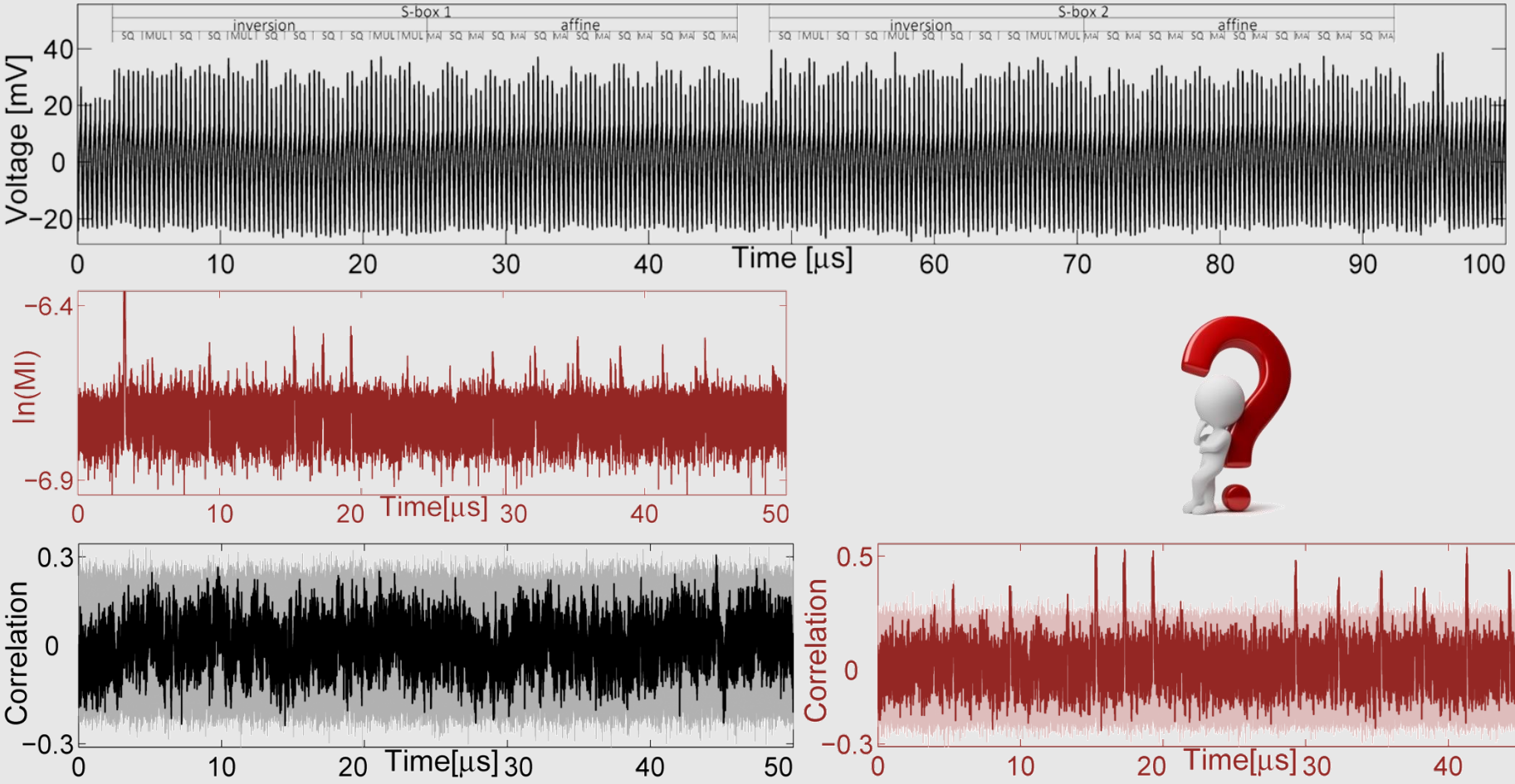
Target Scheme - Evaluation

- Original Design, 3MHz

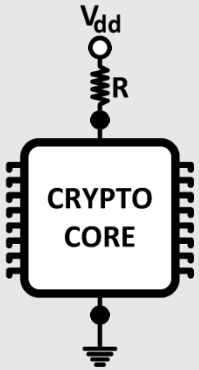


Target Scheme - Evaluation

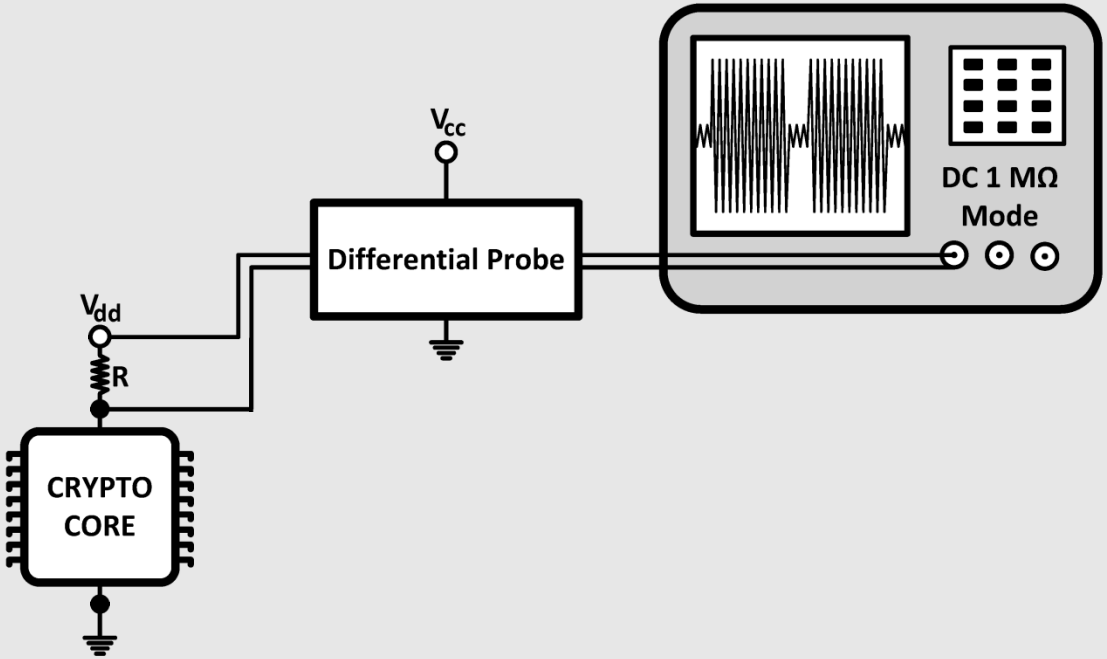
- Original Design, 3MHz



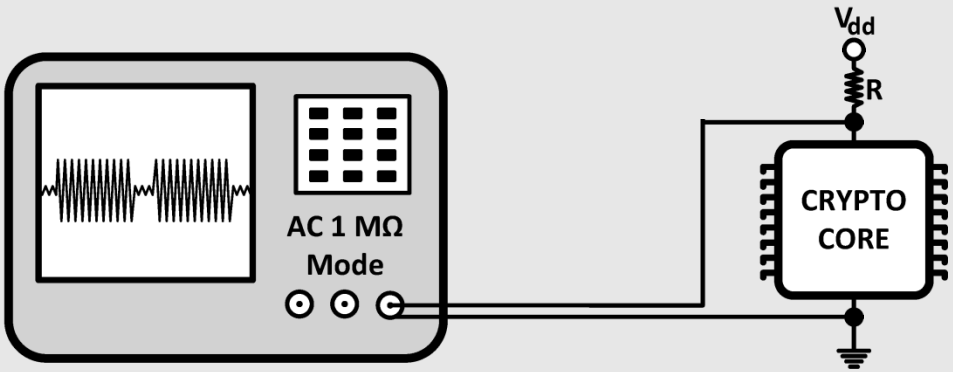
Measurement Setup



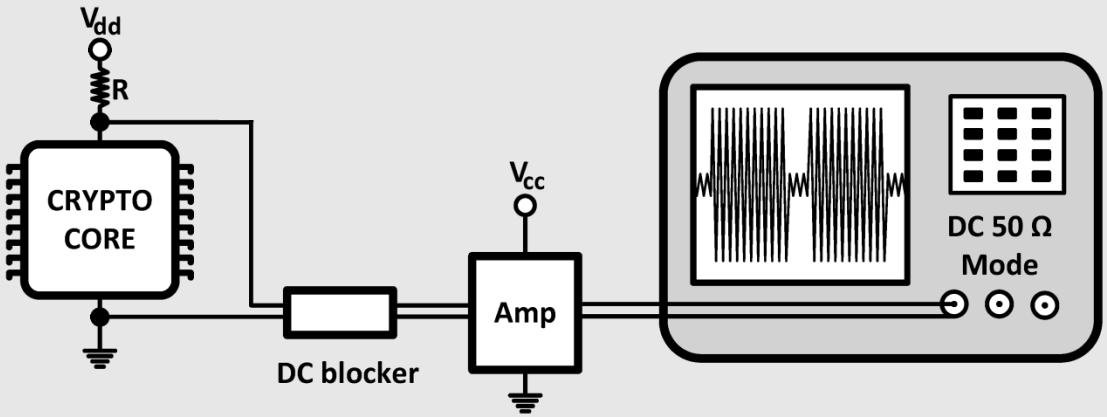
Measurement Setup



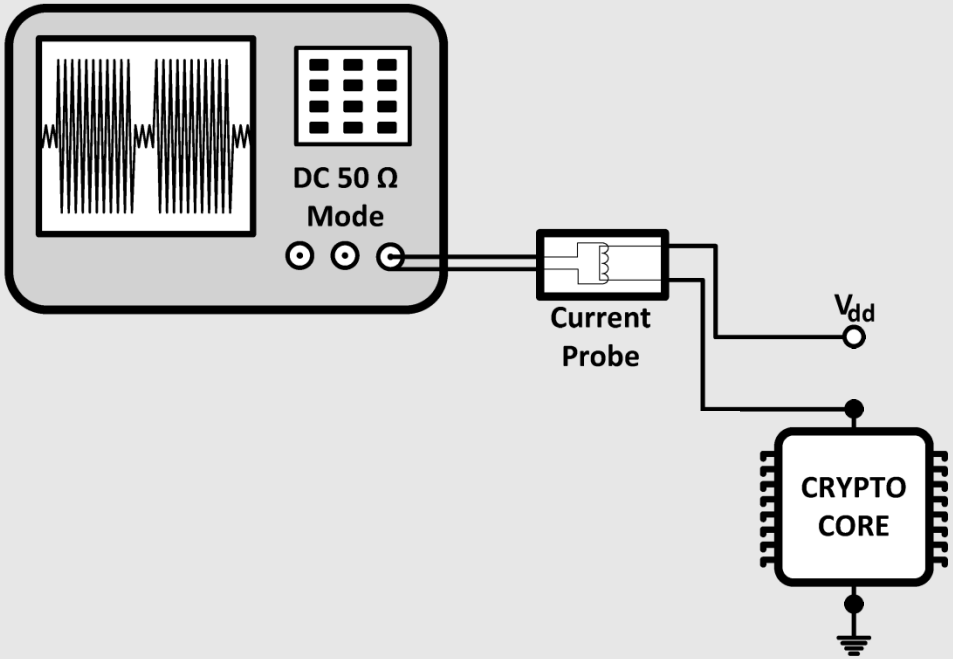
Measurement Setup



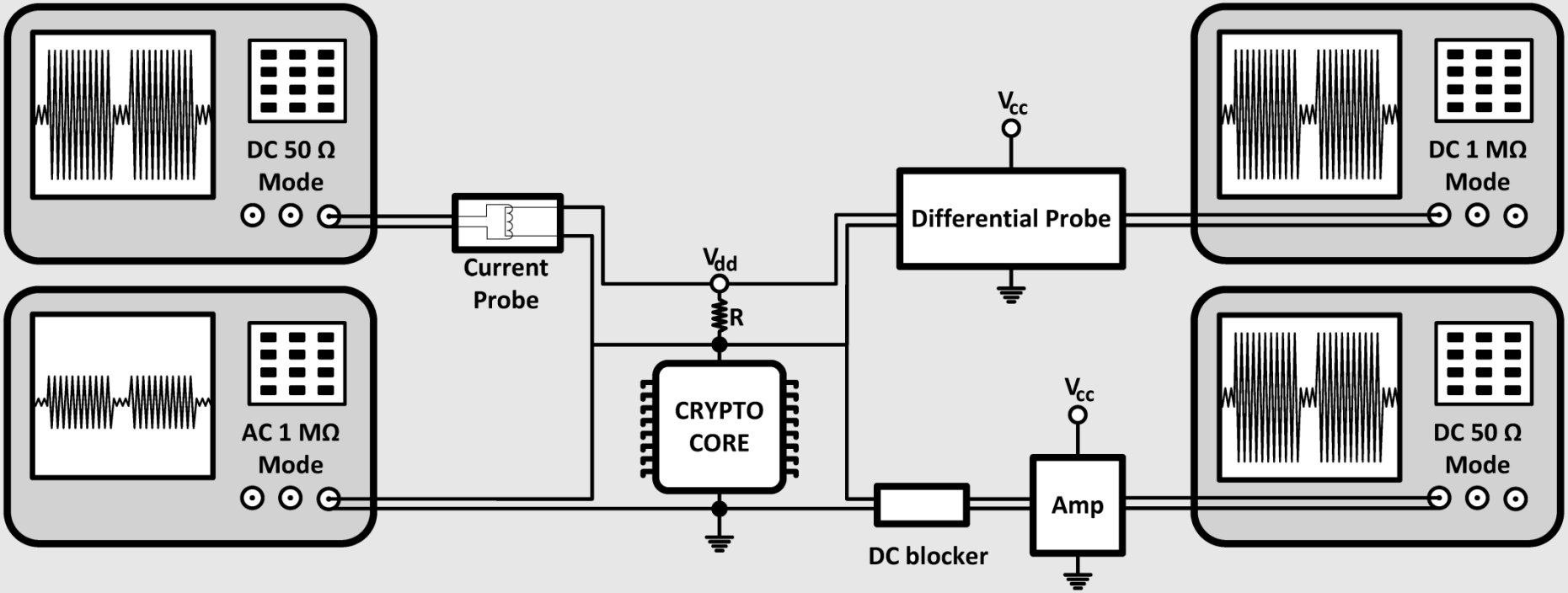
Measurement Setup



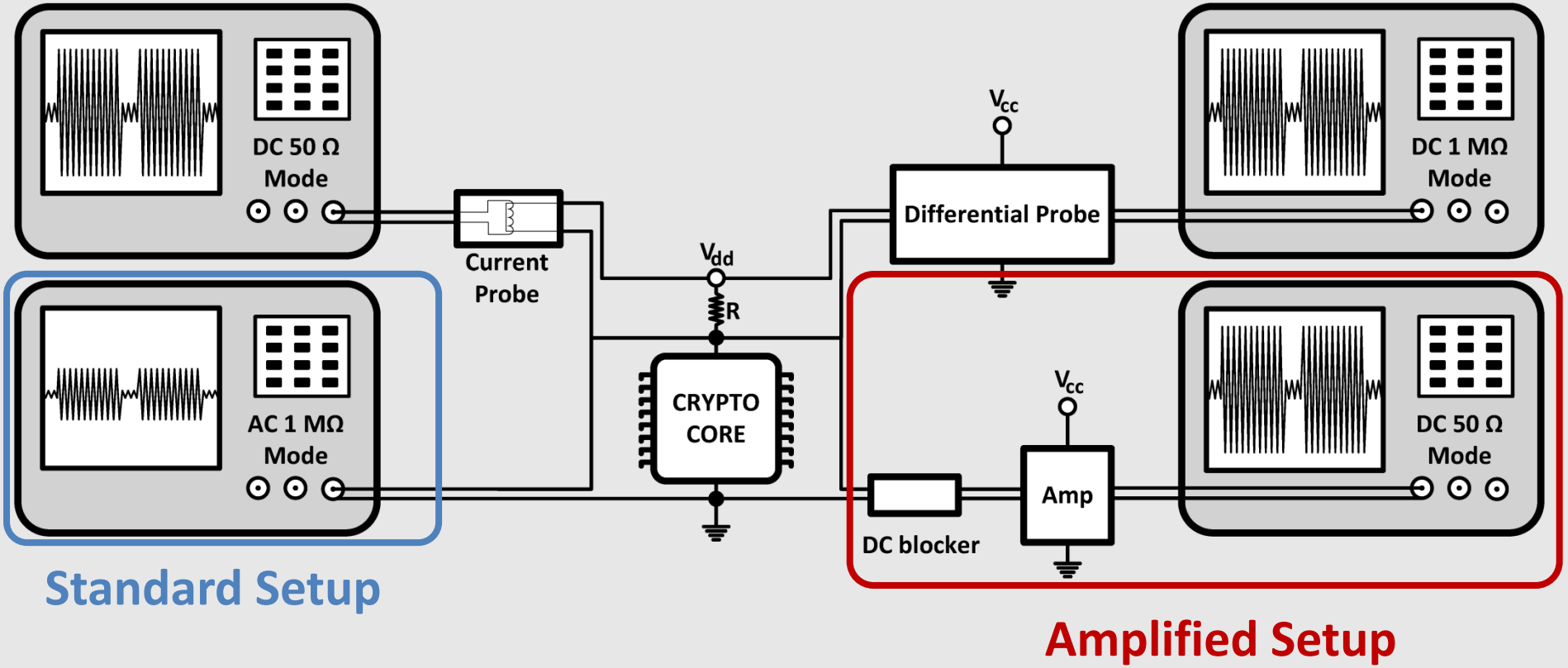
Measurement Setup



Measurement Setup

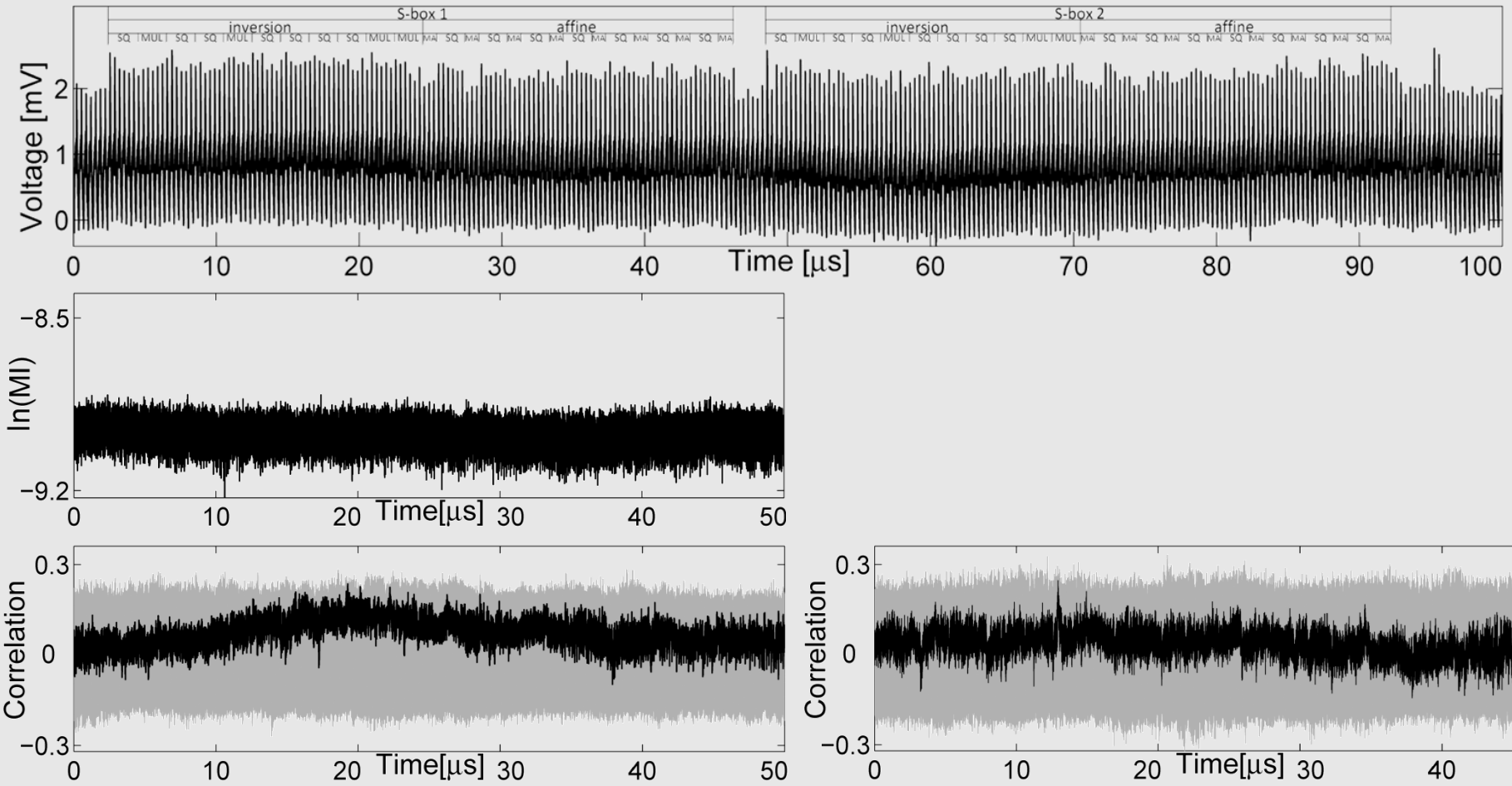


Measurement Setup



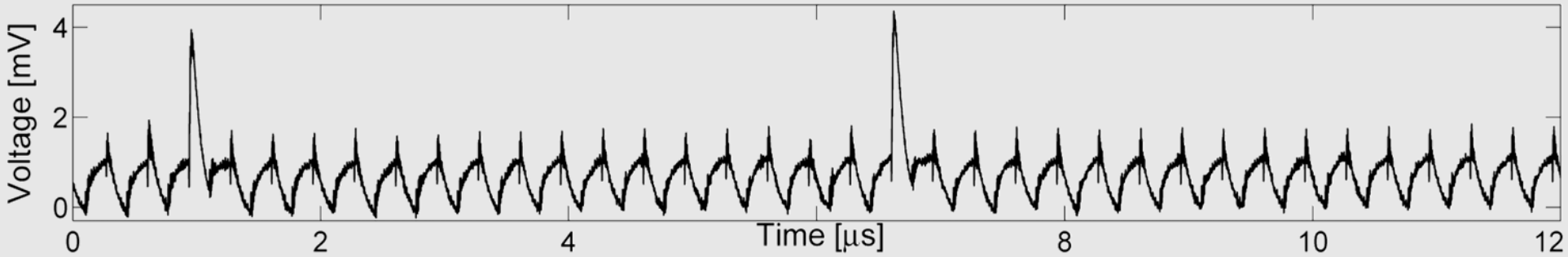
Target Scheme – Evaluation (Standard Setup)

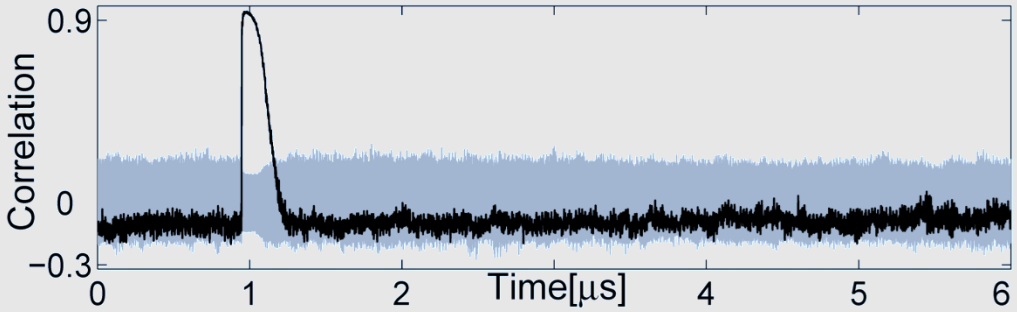
- Original Design, 3MHz

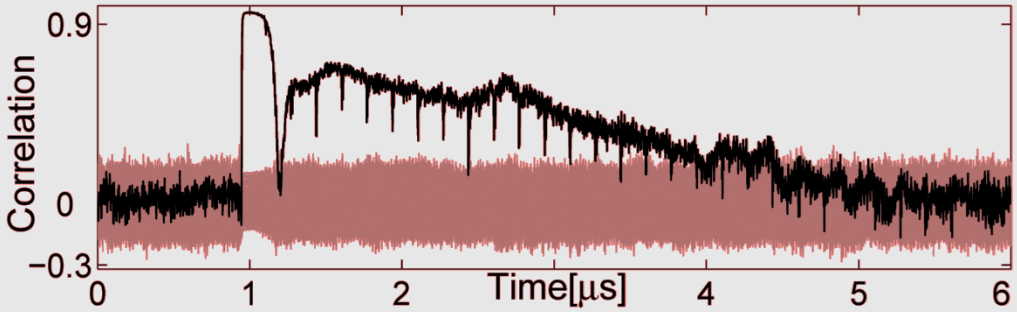


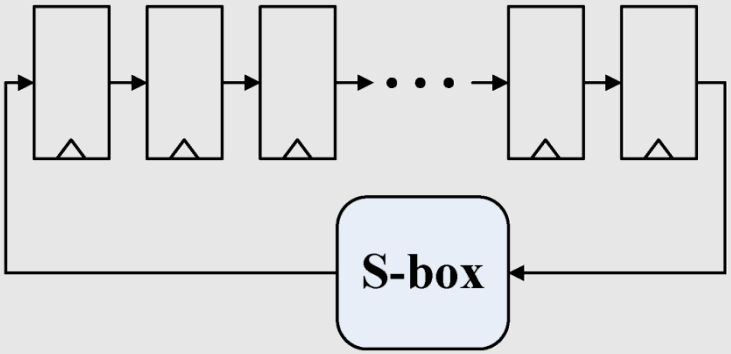
Standard vs. Amplified Setup

- Exemplary Design

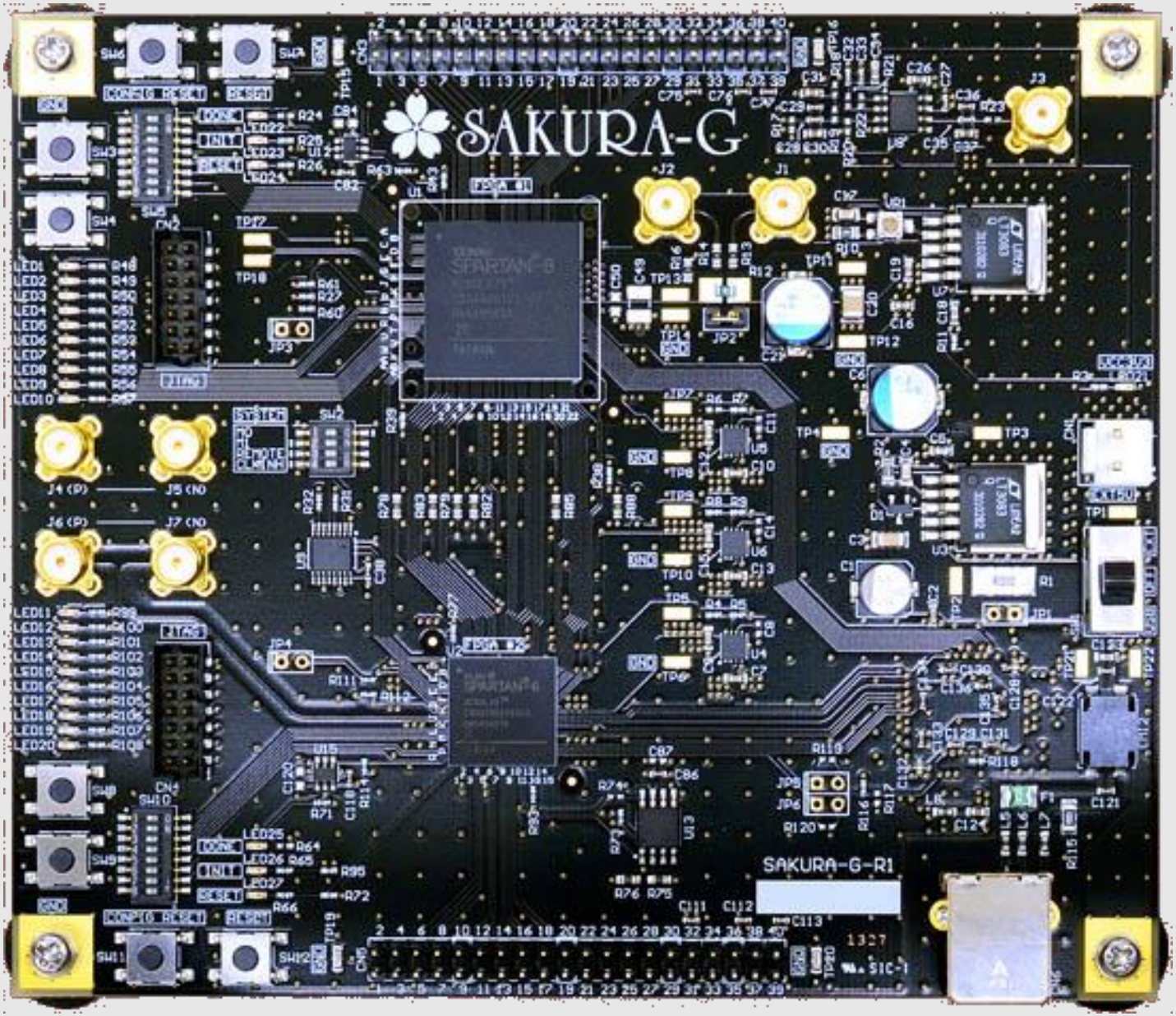




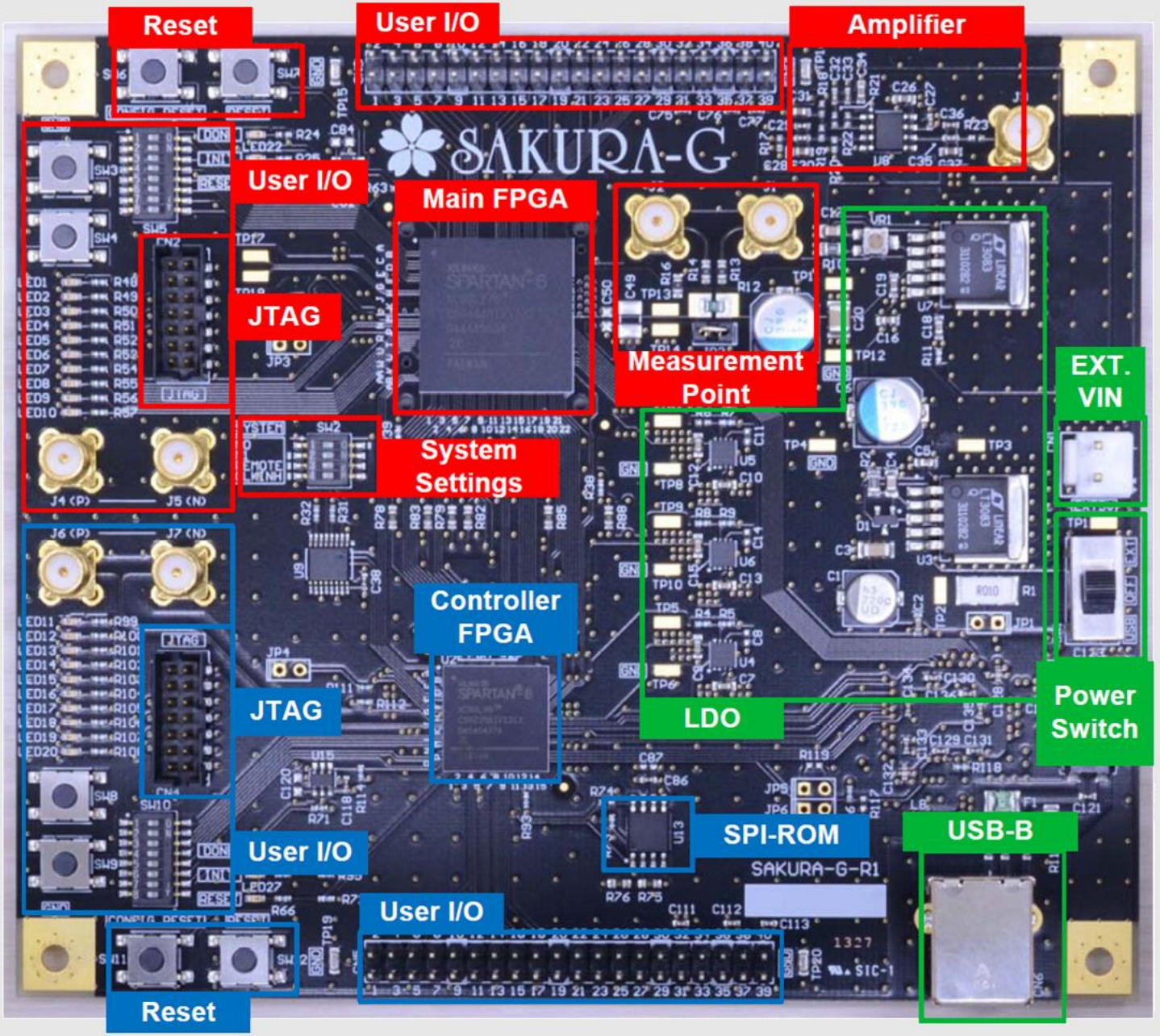




SAKURA-G



SAKURA-G


Reset
User I/O
Amplifier
SAKURA-G
User I/O
Main FPGA
JTAG
Measurement Point
EXT. VIN
System Settings
Controller FPGA
JTAG
LDO
Power Switch
SPI-ROM
USB-B
SAKURA-G-R1
User I/O
Reset

Efficiency as a Factor

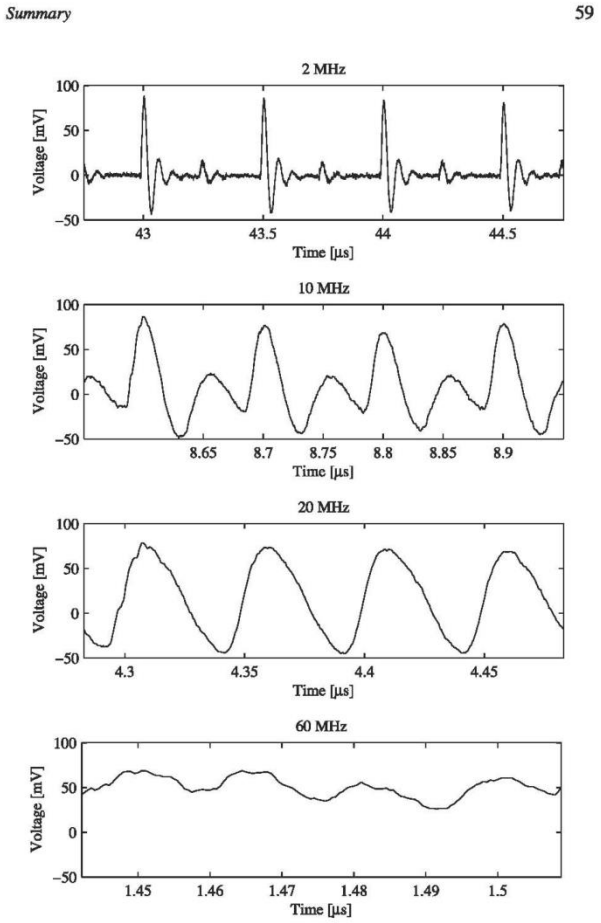
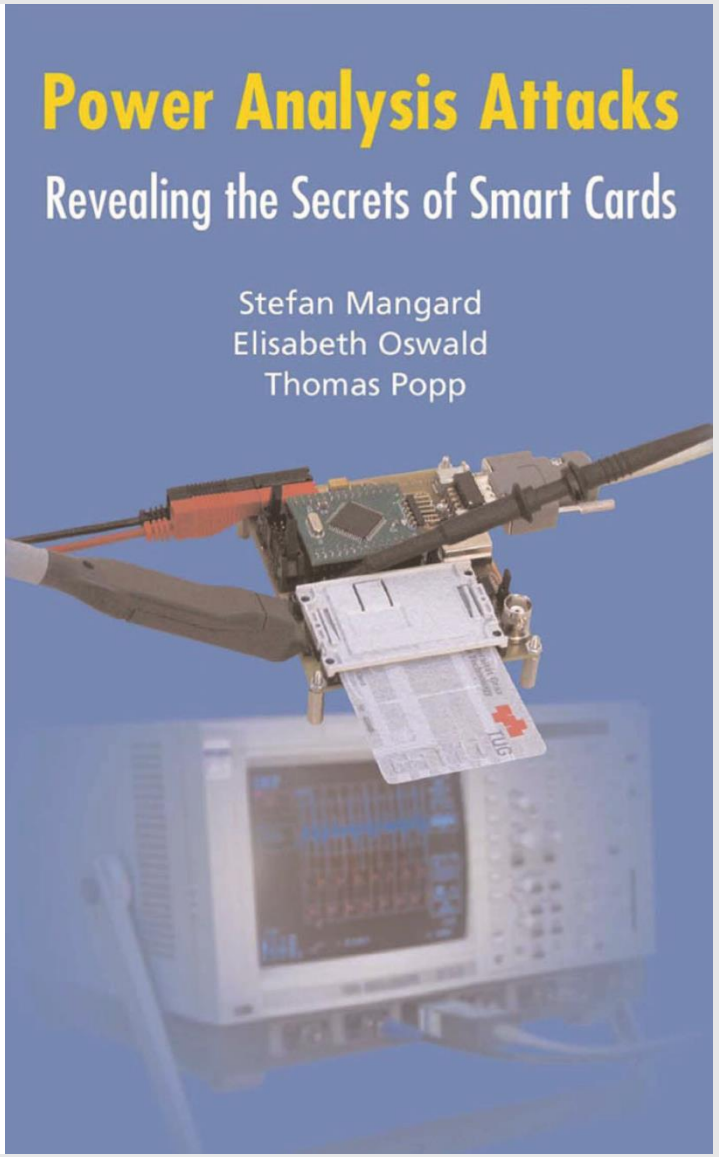
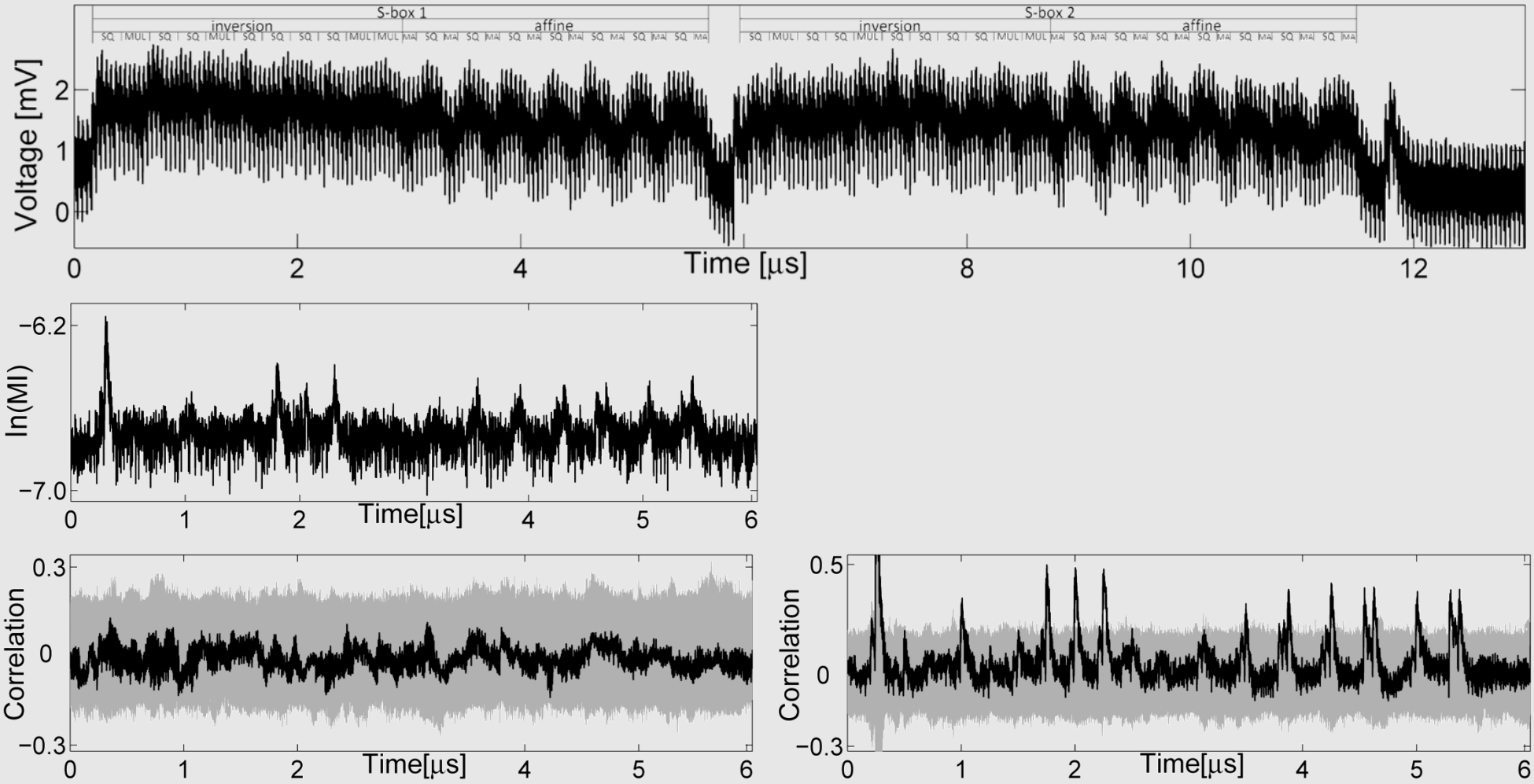


Figure 3.12. The power consumption of the AES ASIC during four clock cycles. A different clock frequency has been used for each of the four traces.



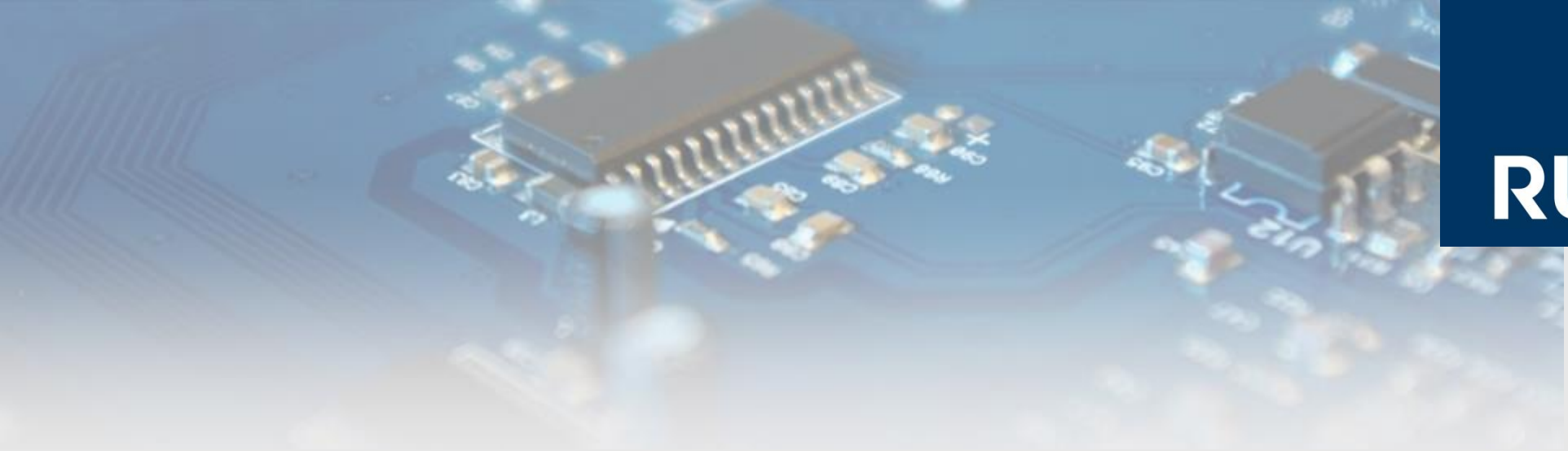
Efficiency as a Factor

- Original Design, Standard Setup, 24MHz



Summing Up / Future Issues

- Cost of univariate resistance
  - security-performance tradeoff
  - processing the shares consecutively
- a light at the end of the tunnel by [pure] masking in hardware?
  - slowly reaching the software performance?
    - making a processor by giant hardware?
  - relatively easy ways to combine the leakages
    - measurement setup & high clock freq.
- What to do when evaluating a countermeasure / product?
  - without any addition/modification on measurement setup?
    - not fair, the attacker may do it
  - with any sophisticated measurement setup?
    - not fair, its security relies on a univariate leak-free scheme



Thanks!

Any questions?

amir.moradi@rub.de

Embedded Security Group, Ruhr University Bochum, Germany