RUHR-UNIVERSITÄT BOCHUM

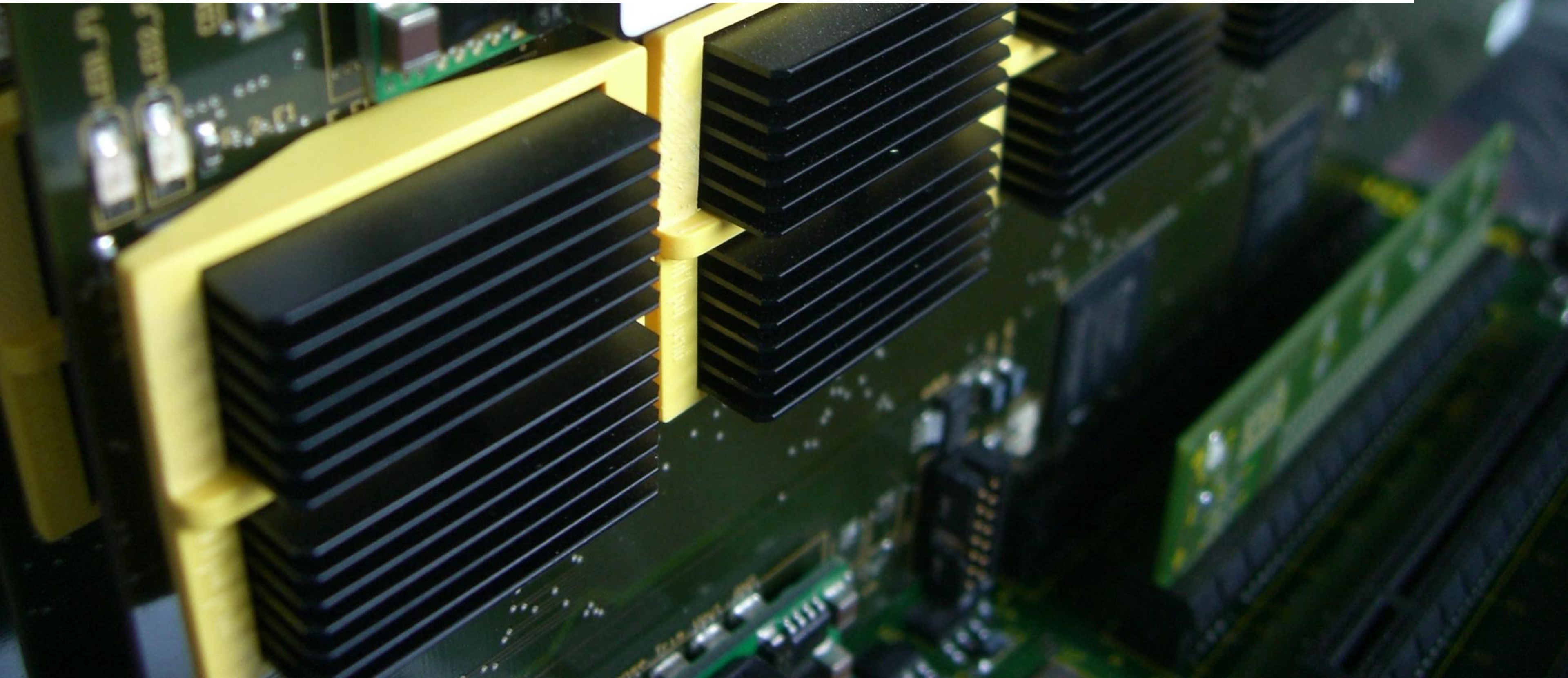
hg|i SHA

RUB

PUSHING THE LIMITS: ULTRA-LIGHTWEIGHT AES ON RECONFIGURABLE HARDWARE

PASCAL SASDRICH, TIM GÜNEYSU

4TH WORKSHOP ON SECURE HARDWARE AND SECURITY EVALUATION, SAINT-MALO, FRANCE SEPTEMBER 17, 2015



WHAT IS THE IDEA OF THIS WORK?

Problem: Standardized symmetric encryption is often required for several purposes but has only minor importance and should not occupy much area in hardware such as FPGAs.

- **AES-128:** block cipher with 128-bit security level standardized in 2001 by NIST
- **FPGA:** *(re-)programmable logic device* popular for cryptographic implementations
- **Xilinx**: one of the leading manufactures for FPGAs, founded in 1984

Question: Can we build an AES encryption core that is smaller than dedicated soft core implementations such as the PicoBlaze micro-processor for Xilinx FPGAs (26 slices / 1 BRAM)?

Our Contribution:

- we provide the currently smallest AES-128 implementation optimized for Xilinx FPGAs
- our implementation provides full encryption including on-the-fly key expansion
- the entire implementation occupies only **21 slices** without any BRAM on a Spartan-6 device

XILINX FIELD-PROGRAMMABLE GATE ARRAYS (FPGA)

Let‘s take a short glance at Xilinx FPGAs:

- organized in special cells such as Configurable Logic Blocks (CLB), memory (BRAM), IO,…
- CLBs contain 2 slices with 4 LUTs / 8 FF each
- in general three different kind of slices exist:
  - **Slice-X**: basic logic functionality
  - **Slice-L**: enhanced logic functionality
  - **Slice-M**: additional memory features

For our work in particular the Slice-M were of special interest:

- all 4 LUTs could be combined and used as 4 x 64 bit ***distributed memory***
- different memory options such as
  - 32 x 8-bit (RAM32M),
  - 64 x 4-bit (RAM64M),
  - and 256 x 1-bit (RAM256X1S)

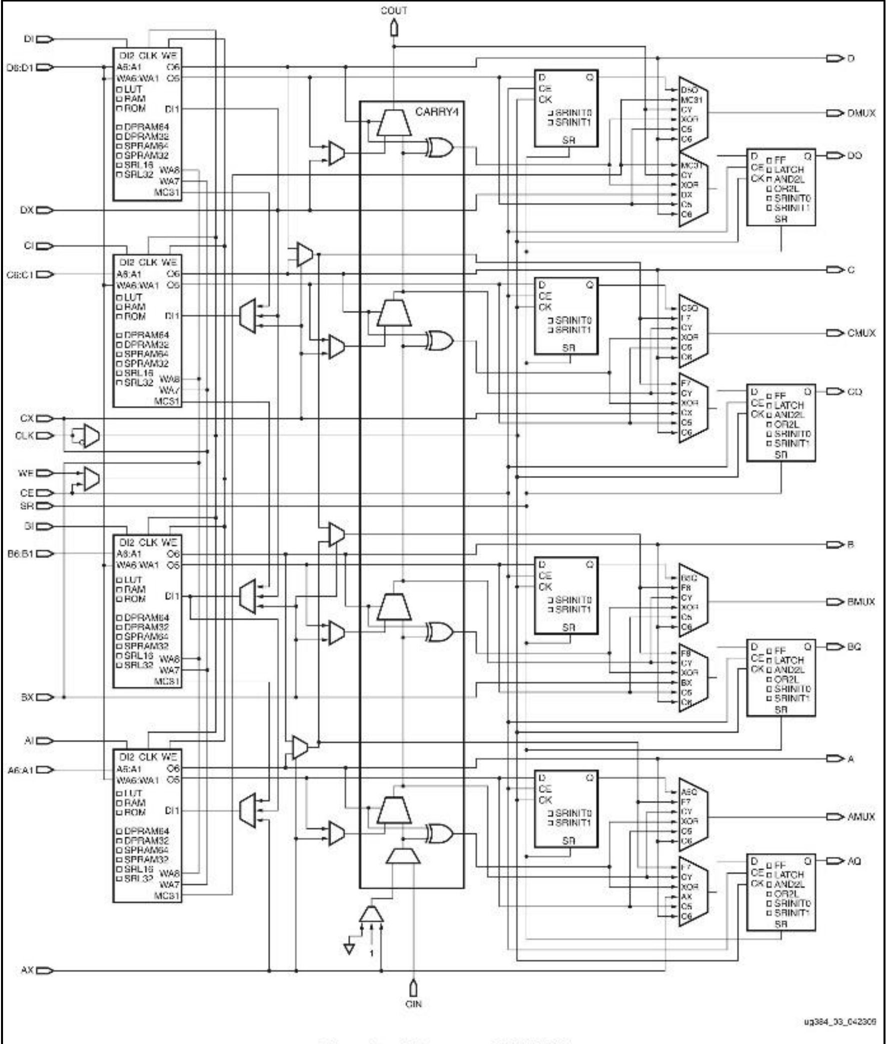


Figure 3: **Diagram of SLICEM**

WHAT ARE THE SPECIAL IDEAS OF THIS WORK?

Problem: Find an optimal mapping of the AES algorithm to implement it as small as possible on current Xilinx FPGA technology.

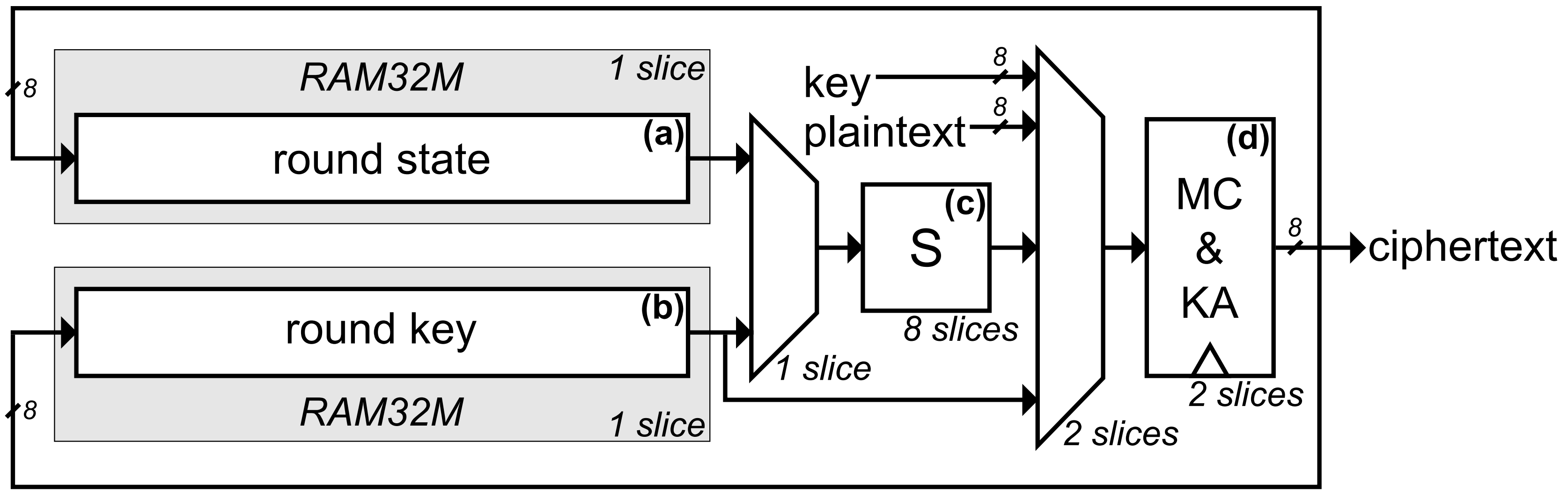
Ideas:

- store intermediate state into ***distributed memory***:
  - AES has 128-bit internal state and most operations are byte-wise
  - Xilinx FPGAs can store up to 32 x 8-bit values (256 bit) into a single slice
  - storing this data in flip-flops would require 32 slices

- in terms of logic, the **AES S-box** is the most costly part:
  - compare different S-box implementations and their size on FPGAs
  - share S-box for key schedule and round computation

- **MixColumns** most challenging operation in particular for byte-serial implementations:
  - only operation that does not work on single bytes but columns (4 bytes) of the state
  - main operation is XOR (besides multiplications with small constants in GF(2^8)
  - following AddRoundKey operation can be merged with MixColumns computation

THE BASIC ENCRYPTION ARCHITECTURE | OVERVIEW



Implementation:

- **byte-serial round-based** AES-128 encryption architecture including key schedule
- round function and key schedule are **merged and share** components (S-box and XOR-addition)

Performance:

- encryption circuit requires **15 slices**, the remaining **6 slices** are required for control logic
- encryption takes **1471 clock cycles** (and has maximum frequency of 105 MHz)
- maximum throughput of **9.12 Mbps** (0.43 Mbps / Slice)

THE BASIC ENCRYPTION ARCHITECTURE | KEY SCHEDULE

The AES-128 key schedule mainly involves *S-box computation, rotations, round constants additions* and *XOR-operations*.

Design ideas:

- store round key in distributed memory, too
- use remaining 128-bits of key state register for round constants
- update round-key in a byte-wise fashion
- share S-box with round function
- use MC&KA component for round constant additions and XOR-operations

Algorithm 1 Key expansion for AES-128

```
Input: $K^i = k_0^i, ..., k_{15}^i$
Output: $K^{i+1} = k_0^{i+1}, ..., k_{15}^{i+1}$
1: $k_0^{i+1} = k_0^i \oplus S(k_{13}^i) \oplus RC[i+1]$
2: $k_1^{i+1} = k_1^i \oplus S(k_{14}^i)$
3: $k_2^{i+1} = k_2^i \oplus S(k_{15}^i)$
4: $k_3^{i+1} = k_3^i \oplus S(k_{12}^i)$
5: for j from 4 to 15 do
6:     $k_j^{i+1} = k_j^i \oplus k_{j-4}^{i+1}$
```

Key update has to be performed in **fixed order** due to data dependencies within the key schedule function.

THE BASIC ENCRYPTION ARCHITECTURE | ROUND FUNCTION

- round function consists of 4 operations: *SubBytes, ShiftRows, MixColumns* and *AddRoundKey*
- prior to first round, an additional key addition is performed (key whitening)
- last round is different, it lacks the MixColumns operation

SubBytes:

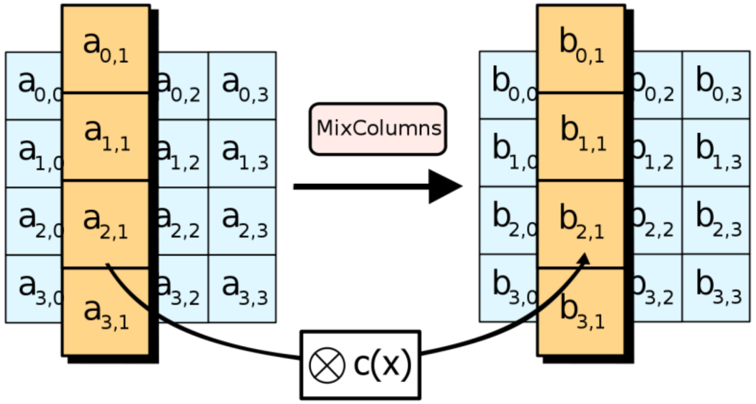
- implemented in a look-up table based fashion (8 slices)

ShiftRows:

- implemented by changing state register addresses (see next slide)

MixColumns and AddRoundKey:

- MC is a 32-bit operation that had to be byte-serialized
- implements an 8-bit register (using FFs) for intermediate results
- register is preceded by ALU providing 4 operation:
  - $00 : r_i = x_i$ (set)
  - $01 : r_i = r_{i-1} \oplus x_i$ (add)
  - $10 : r_i = r_{i-1} \oplus 02 \cdot x_i$ (add two times)
  - $11 : r_i = r_{i-1} \oplus 03 \cdot x_i$ (add three times)
- AddRoundKey is merged using the add operation

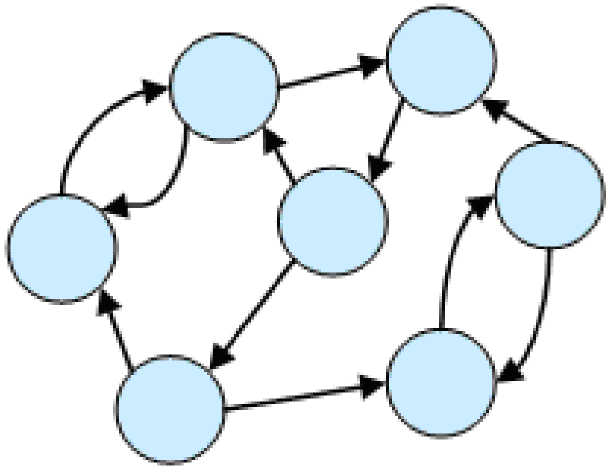

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$\otimes$ c(x)
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

THE BASIC ENCRYPTION ARCHITECTURE | CONTROL LOGIC

Control logic implements counters and state machine to perform a **full AES encryption** including **full key expansion**.

Counters:

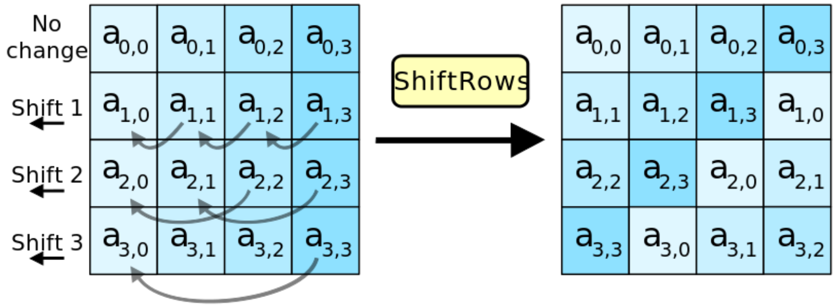
- 4-bit counter for counting the rounds (2 LUTs / 0.5 slices)
- 4-bit counter for tracking current byte (2 LUTs / 0.5 slices)
- 2-bit counter for tracking input byte of MixColumns (1 LUT / 0.25 slices)



State Machine:

- calculation of next state transition (1.5 slices)
- Generation of signals to control encryption circuit (microcode-like approach, 1.5 slices)

In addition, the addressing signals for the state register have to be generated in order to consider the **ShiftRows** operation (1.75 slices).


No change
Shift 1
Shift 2
Shift 3
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

IMPLEMENTATION RESULTS AND COMPARISON

Design/ Variant	Device	Implementation			Performance			
		Data (Bit)	*Logic (Slices)*	*Memory (BRAM)*	*Clock (MHz)*	*Period (Cycles)*	*Throughput (Mbps)*	*Throughput/Area (Mbps/Slice)*
Bulens	Virtex-5	128	400	0	350	11	4,072	10.18
Chodowiec	XC2S30	32	222	3	55	44	166	0.75
Rouvroy	XC3S50	32	163	3	72	46	208	1.28
Good	XC2S15	8	124	2	67	3691	2.2	0.02
Chu	XC3S50	8	184	0	46	160	36.51	0.20
PicoBlaze	XC2S30	8	119	2	90	13546	0.71	0.01
Chu	XC6SLX4	8	80	0	73	160	58.13	0.73
This work	XC6SLX4	8	21	0	105	1471	9.12	0.43
Simon	XC6SLX4	1	13	0	136	(4896)	3.6	0.28

- compared to other designs our work only provides **moderate performance**
- but our main focus laid on finding the **smallest implementation** (at cost of higher run-time)
- even **dedicated lightweight ciphers** such as Simon cannot be implemented much smaller

FUTURE WORK AND DIRECTIONS?

- **Side-Channel Protection:**
  - our design nigh on inherently supports shuffling
  - masking using randomized look-up tables (but mask update is challenging)
- **use AES as Pseudo Random Number Generator (PRNG)**
- **wrap encryption engine with different Modes of Operation:**
  - counter mode to provide not only encryption but also decryption
  - ...
- **practical application and implementation, e.g. for IP protection and protocols**

WHAT IS THE CONCLUSION OF THIS WORK?

We provide an ultra-lightweight AES-128 implementation designed for modern Xilinx FPGAs and even competitive to lightweight ciphers such as Simon.

RUHR-UNIVERSITÄT BOCHUM

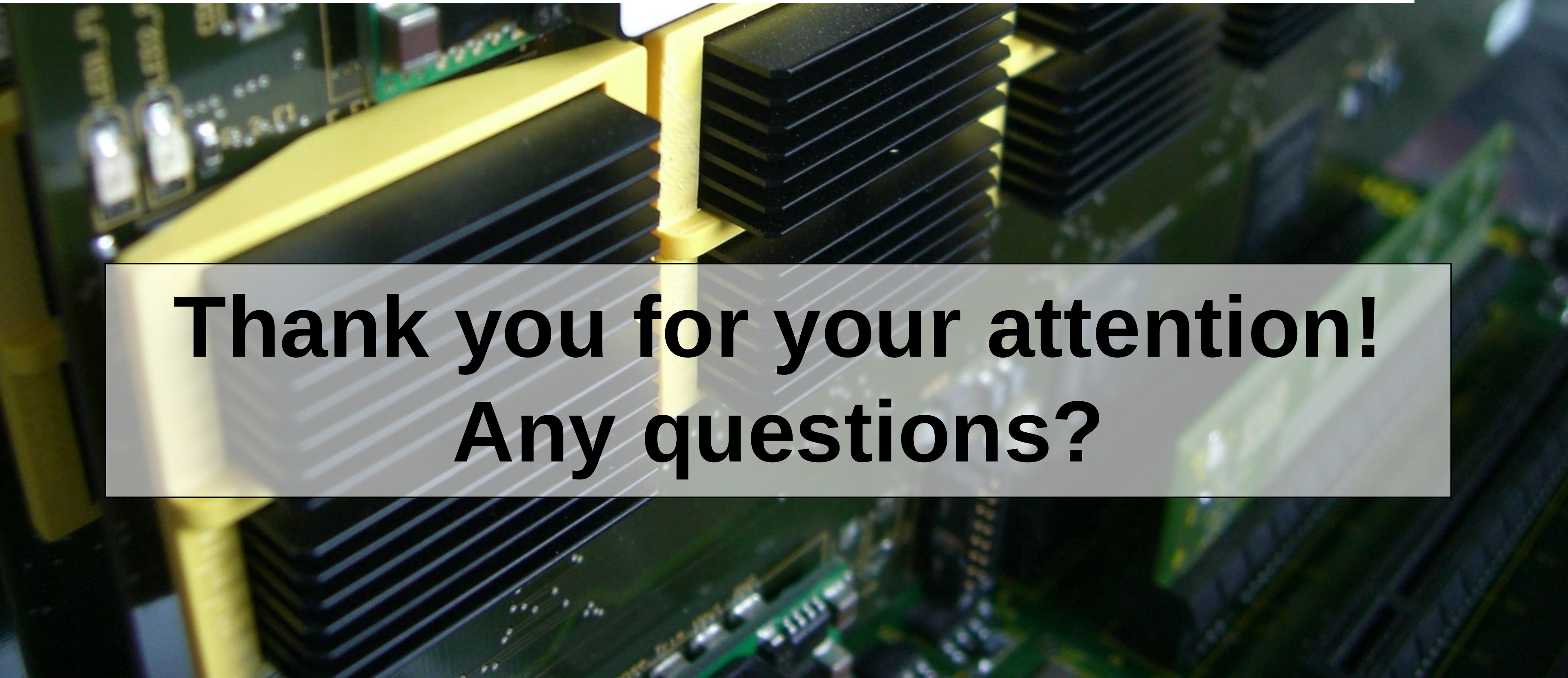
hgi SHA

RUB

PUSHING THE LIMITS: ULTRA-LIGHTWEIGHT AES ON RECONFIGURABLE HARDWARE

pascal.sasdrich@rub.de

4TH WORKSHOP ON SECURE HARDWARE AND SECURITY EVALUATION, SAINT-MALO, FRANCE SEPTEMBER 17, 2015



Thank you for your attention!
Any questions?