RUHR-UNIVERSITÄT BOCHUM

Horst Görtz Institute for IT-Security

A Non-Linear/Linear Instruction Set Extension for Lightweight Ciphers

Susanne Engels, *Elif Bilge Kavun*, Hristina Mihajloska, Christof Paar, Tolga Yalçın

ARITH'21 – Austin, TX – April 8, 2013

Overview

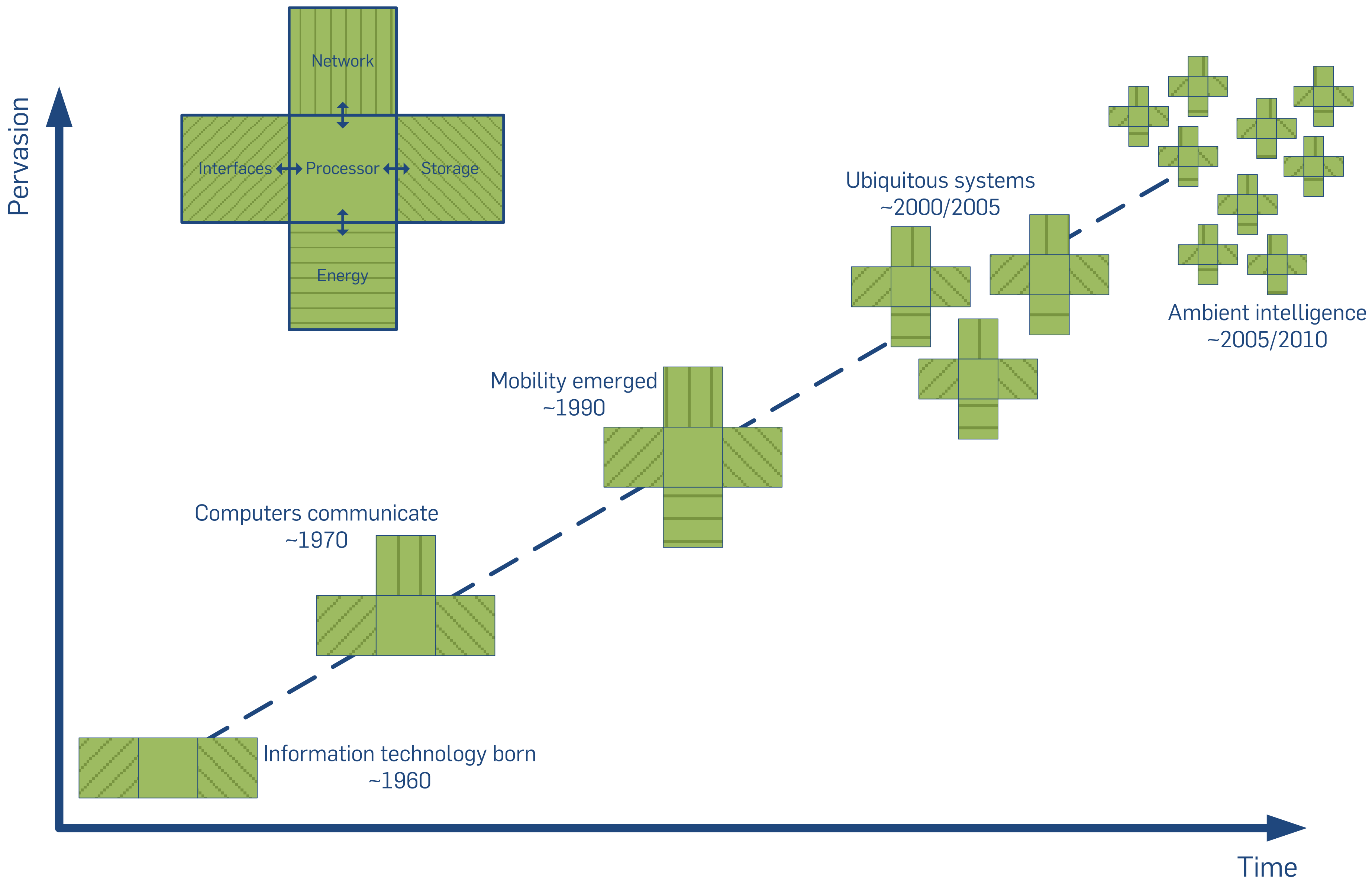
- Lightweight and Pervasive Devices
  - Which Devices and Applications?
  - Security Need
- Standard/Lightweight Cryptography
  - Current Solutions
  - Software-oriented Solutions?
- Instruction Set Extension: NLU
  - ISE Model
  - Hardware Unit
  - Applications
- Conclusion and Future Directions

Overview

- **Lightweight and Pervasive Devices**
  - Which Devices and Applications?
  - Security Need
- Standard/Lightweight Cryptography
  - Current Solutions
  - Software-oriented Solutions?
- Instruction Set Extension: NLU
  - ISE Model
  - Hardware Unit
  - Applications
- Conclusion and Future Directions

Lightweight and Pervasive Devices

Development of Computers and Constrained Devices

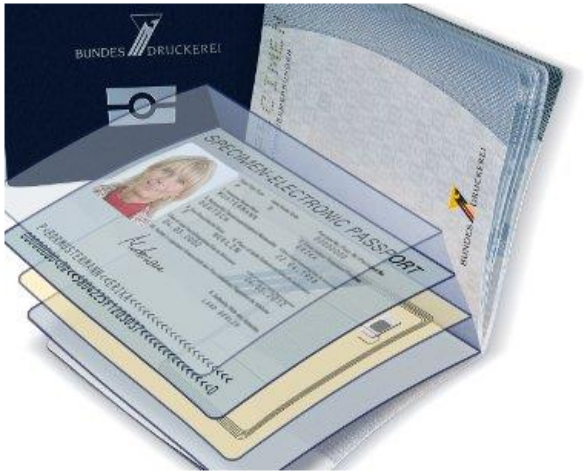

Network
Interfaces
Processor
Storage
Energy
Pervasion
Time
Information technology born ~1960
Computers communicate ~1970
Mobility emerged ~1990
Ubiquitous systems ~2000/2005
Ambient intelligence ~2005/2010

Horst Görtz Institute for IT-Security

RUB

Lightweight and Pervasive Devices

Lightweight Applications



- Electronic passports

- Logistics

- Road toll-collection

Lightweight and Pervasive Devices

CONSTRAINED DEVICES?

- More precisely:
  - RFID-Tags: <u>R</u>adio-<u>F</u>requency <u>Id</u>entification
  - Smart Cards
  - Wireless Sensors

Lightweight and Pervasive Devices

Constrained Devices?

- Low power and energy consumption
  - Active devices with on-chip batteries
  - Battery-less passive devices that rely on limited EM-transmitted power
- Low area and complexity
  - Gate count, I/O pin count, storage
- Constrained communication bandwidth
  - Due to increased device mobility and power constraints

Lightweight and Pervasive Devices

THE NEED FOR SECURING CONSTRAINED DEVICES

- Control on access: Car key systems, internet banking, etc.
- Enforcing business models: Electronic wallet, SIM-cards, etc.
- Counterfeiting: Gaming, batteries, etc.
- Privacy protection: GSM, medical sensors, etc.

Lightweight and Pervasive Devices

THE NEED FOR SECURING CONSTRAINED DEVICES

- Control on access: Car key systems, internet banking, etc.
- Enforcing business models: Electronic wallet, SIM-cards, etc.
- Counterfeiting: Gaming, batteries, etc.
- Privacy protection: GSM, medical sensors, etc.

Good security architectures needed to resist attacks!

Lightweight and Pervasive Devices

Constrained Devices?

- Low power and energy consumption
  - Active devices with on-chip batteries
  - Battery-less passive devices that rely on limited EM-transmitted power
- Low area and complexity
  - Gate count, I/O pin count, storage
- Constrained communication bandwidth
  - Due to increased device mobility and power constraints

Lightweight and Pervasive Devices

Constrained Devices?

- Low power and energy consumption
  - Active devices with on-chip batteries
  - Battery-less passive devices that rely on limited EM-transmitted power
- Low area and complexity
  - Gate count, I/O pin count, storage
- Constrained communication bandwidth
  - Due to increased device mobility and power constraints

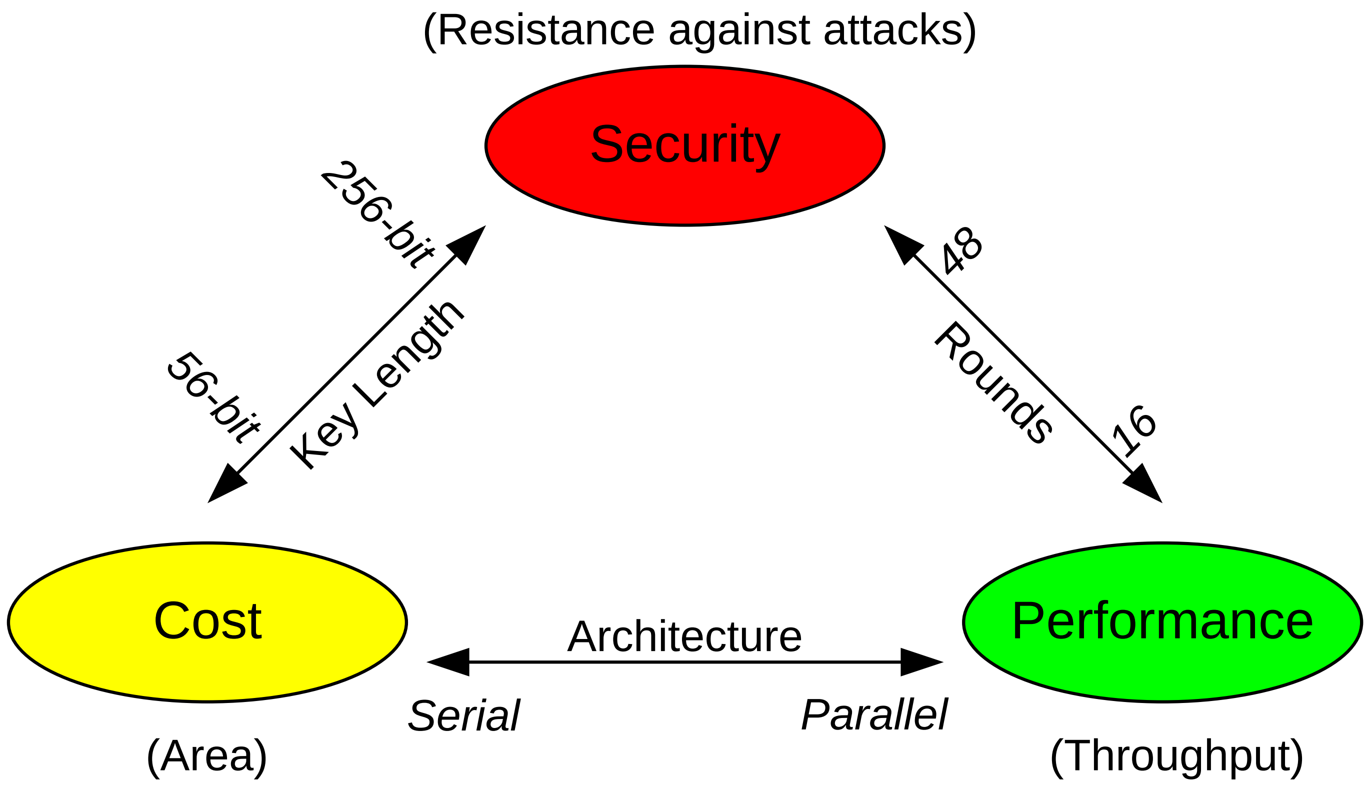
The security architecture should be designed in a way to fulfill these constraints!

Overview

- Lightweight and Pervasive Devices
  - Which Devices and Applications?
  - Security Need
- **Standard/Lightweight Cryptography**
  - Current Solutions
  - Software-oriented Solutions?
- Instruction Set Extension: NLU
  - ISE Model
  - Hardware Unit
  - Applications
- Conclusion and Future Directions

Standard/Lightweight Cryptography

- Requirements : The trade-off



Standard/Lightweight Cryptography

Lightweight Cryptography Requirements?

- No strict criteria, but common features:
  - Should be cheaper than traditional cryptography
  - A reduced level of security is sufficient
    - Key size – below 128 bits
  - Short data block size

Standard/Lightweight Cryptography

Lightweight Block Ciphers

- Algorithms with particularly low implementation costs
  - Tailored to fulfill previously mentioned requirements
- Examples:
  - PRESENT, CLEFIA (*ISO standards*), KLEIN, LED, mCrypton, etc.

Standard/Lightweight Cryptography

Lightweight Block Ciphers

- Algorithms with particularly low implementation costs
  - Tailored to fulfill previously mentioned requirements
- Examples:
  - PRESENT, CLEFIA (*ISO standards*), KLEIN, LED, mCrypton, etc.

Mostly targeted for

***low gate count* in *hardware*!**

Standard/Lightweight Cryptography

Lightweight Block Ciphers – Software Solutions

- 8-bit microprocessors are used widely in the market
  - Hardware-optimized *software-unfriendly* lightweight ciphers are actually mostly implemented in 8-bit microprocessors!
  - Results in higher code size and more cycles
- We should proceed with software-friendly solutions and designs!

Standard/Lightweight Cryptography

Lightweight Block Ciphers – Software Solutions

Software-friendly solutions?

Not much there...

Standard/Lightweight Cryptography

Lightweight Block Ciphers – Software Solutions: Related Work

- Not necessarily for lightweight block ciphers
- Implementation of cipher-specific instructions
  - Plugging the specific cipher as a coprocessor to the main module
  - Increases microprocessor area!
- Other works introduce complex instructions utilization

Standard/Lightweight Cryptography

Lightweight Block Ciphers – Software Solutions: Related Work

- Not necessarily for lightweight block ciphers
- Implementation of cipher-specific instructions
  - Plugging the specific cipher as a coprocessor to the main module
  - Increases microprocessor area!
- Other works introduce complex instructions utilization

A first attempt: ***NLU*** !!!

Overview

- Lightweight and Pervasive Devices
  - Which Devices and Applications?
  - Security Need
- Standard/Lightweight Cryptography
  - Current Solutions
  - Software-oriented Solutions?
- **Instruction Set Extension: NLU**
  - ISE Model
  - Hardware Unit
  - Applications
- Conclusion and Future Directions

Instruction Set Extension: NLU

Non-linear/Linear Instruction Set Extension For Lightweight Ciphers

- In block ciphers;
  - *Non-linear* refers to *substitution* – Introduces *confusion*
  - *Linear* refers to *permutation* – Introduces *diffusion*
- Block ciphers designed in a way to provide these!
  - *Sbox* layer for substitution
  - *Mixing* layer for permutation
- They are essential but also costly in software!

Instruction Set Extension: NLU

Non-linear/Linear Instruction Set Extension For Lightweight Ciphers

- In block ciphers;
  - *Non-linear* refers to *substitution* – Introduces *confusion*
  - *Linear* refers to *permutation* – Introduces *diffusion*
- Block ciphers designed in a way to provide these!
  - *Sbox* layer for substitution
  - *Mixing* layer for permutation
- They are essential but also costly in software!

What if we introduce new instructions for these operations to save cycles?

Instruction Set Extension: NLU

ISE Model

- Special instructions
  - To realize non-linear and linear layers of block ciphers
  - To reduce cycle count and code size
- A unified hardware block for:
  - *Cycle-consuming* substitution and permutation operations
- Call new instructions in software!
  - Results in less cycles…

Instruction Set Extension: NLU

ISE Model

- Special instructions
  - To realize non-linear and linear layers of block ciphers
  - To reduce cycle count and code size
- A unified hardware block for:
  - *Cycle-consuming* substitution and permutation operations
- Call new instructions in software!
  - Results in less cycles...

Hardware block should be cheap!

Instruction Set Extension: NLU

ISE Model

- For substitution:
  - Sboxes expressed in their Algebraic Normal Form (ANF)
- For permutation:
  - Linear layer expressed in binary *matrix multiply-and-add* form

Instruction Set Extension: NLU

ISE Model

- For substitution:
  - Sboxes expressed in their Algebraic Normal Form (ANF)
- For permutation:
  - Linear layer expressed in binary *matrix multiply-and-add* form

Very simple and generic architecture!

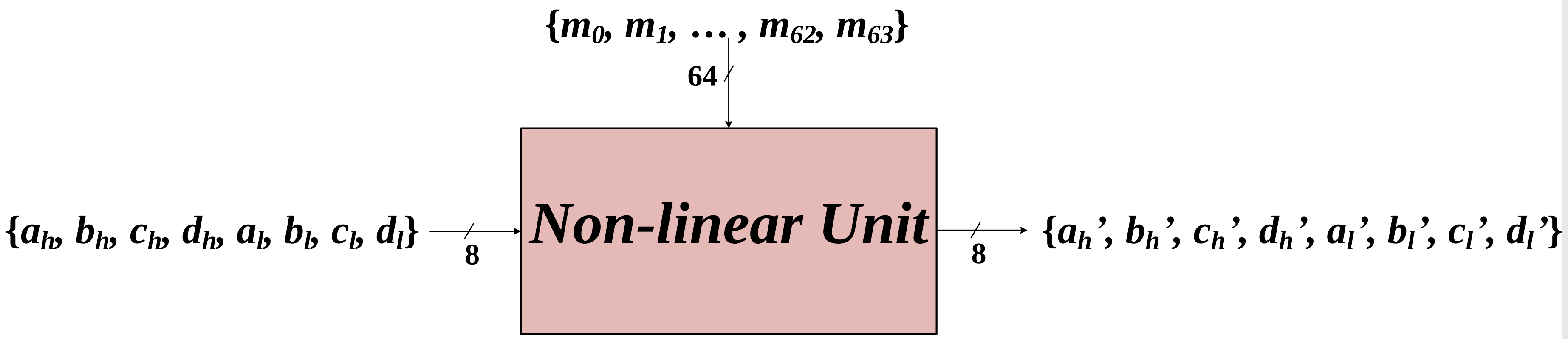
Instruction Set Extension: NLU

HARDWARE – NON-LINEAR UNIT

- Non-linear operations: Expressed in their ANF
  - In Boolean logic, ANF is a method of standardizing and normalizing logical formulas
  - ANF makes it easy to define the function
  - Better result in software than using lookup table for Sbox

Instruction Set Extension: NLU

Hardware – Non-Linear Unit

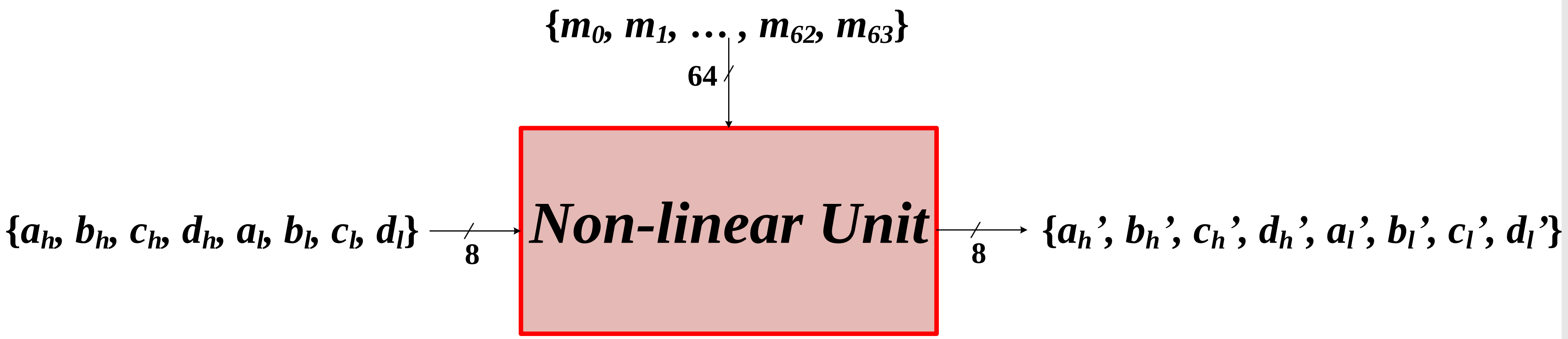


$\{m_0, m_1, \ldots, m_{62}, m_{63}\}$
64
Non-linear Unit
$\{a_h, b_h, c_h, d_h, a_l, b_l, c_l, d_l\}$
8
8
$\{a_h', b_h', c_h', d_h', a_l', b_l', c_l', d_l'\}$

- m_i used for masking the unused ANF components

Instruction Set Extension: NLU

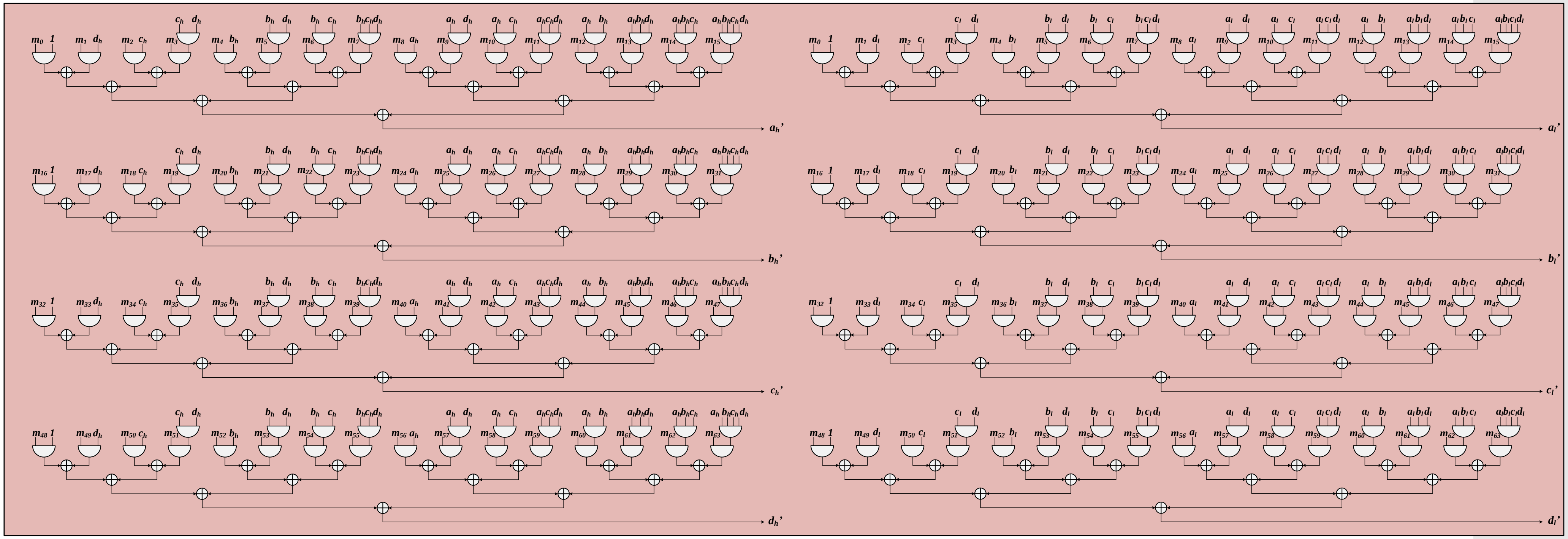
Hardware – Non-Linear Unit


$\{m_0, m_1, \ldots, m_{62}, m_{63}\}$
64
$\{a_h, b_h, c_h, d_h, a_l, b_l, c_l, d_l\}$
8
Non-linear Unit
8
$\{a_h', b_h', c_h', d_h', a_l', b_l', c_l', d_l'\}$

- m_i used for masking the unused ANF components

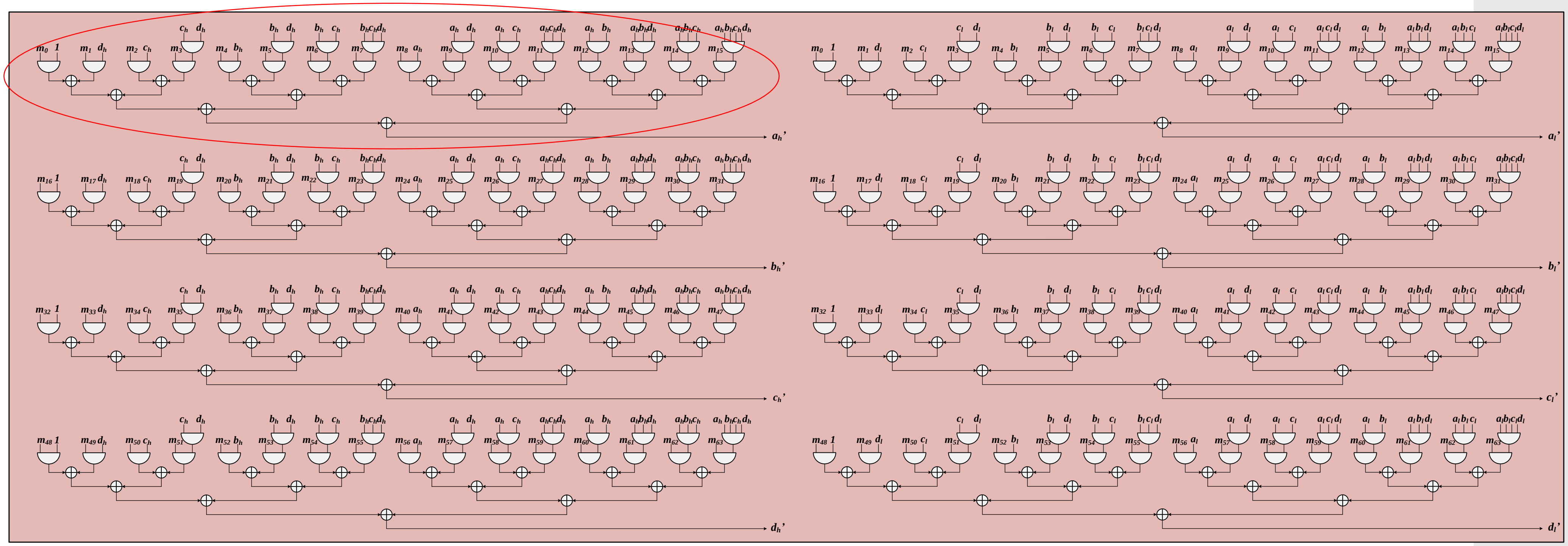
Instruction Set Extension: NLU

Hardware – Non-Linear Unit



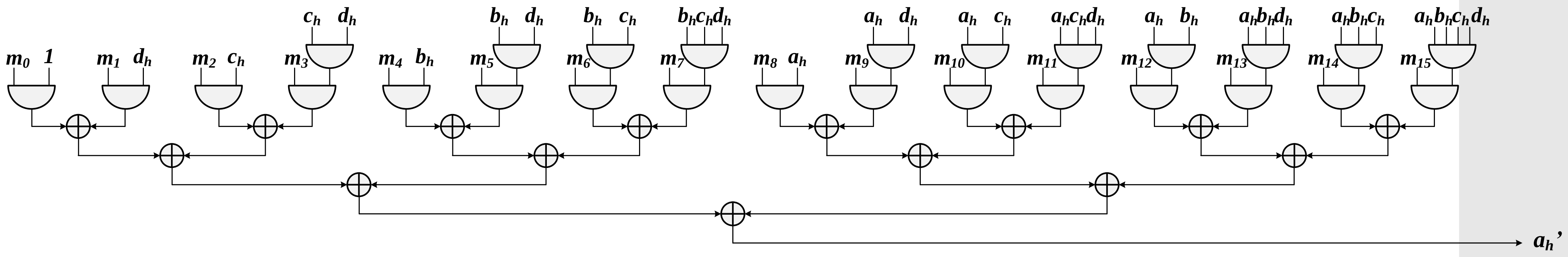
Instruction Set Extension: NLU

Hardware – Non-Linear Unit



Instruction Set Extension: NLU

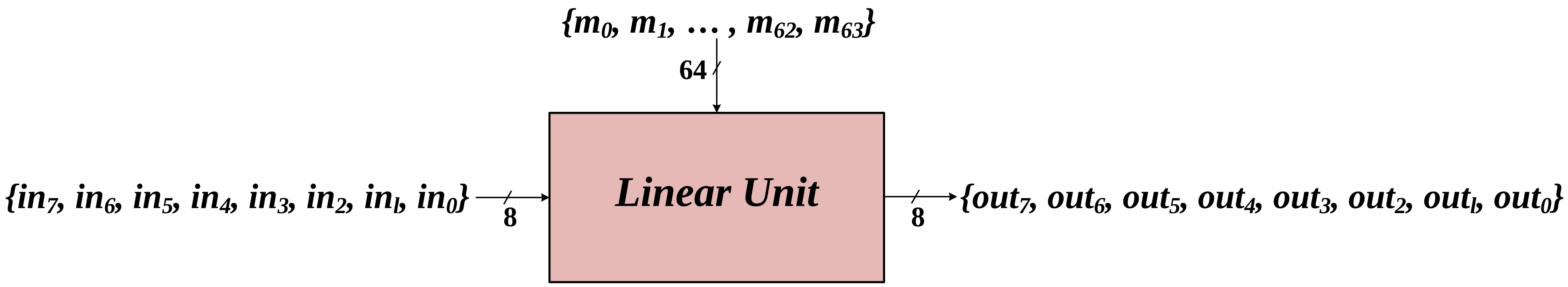
Hardware – Non-Linear Unit



Instruction Set Extension: NLU

Hardware – Linear Unit

- Linear operations: In binary matrix multiplication form

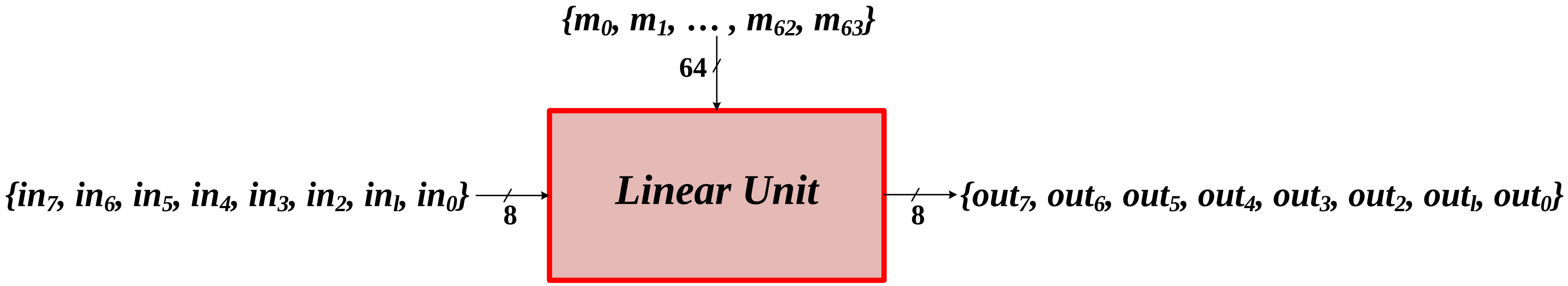


$\{m_0, m_1, \ldots, m_{62}, m_{63}\}$
64
Linear Unit
$\{in_7, in_6, in_5, in_4, in_3, in_2, in_1, in_0\}$
8
8
$\{out_7, out_6, out_5, out_4, out_3, out_2, out_1, out_0\}$

Instruction Set Extension: NLU

Hardware – Linear Unit

- Linear operations: In binary matrix multiplication form



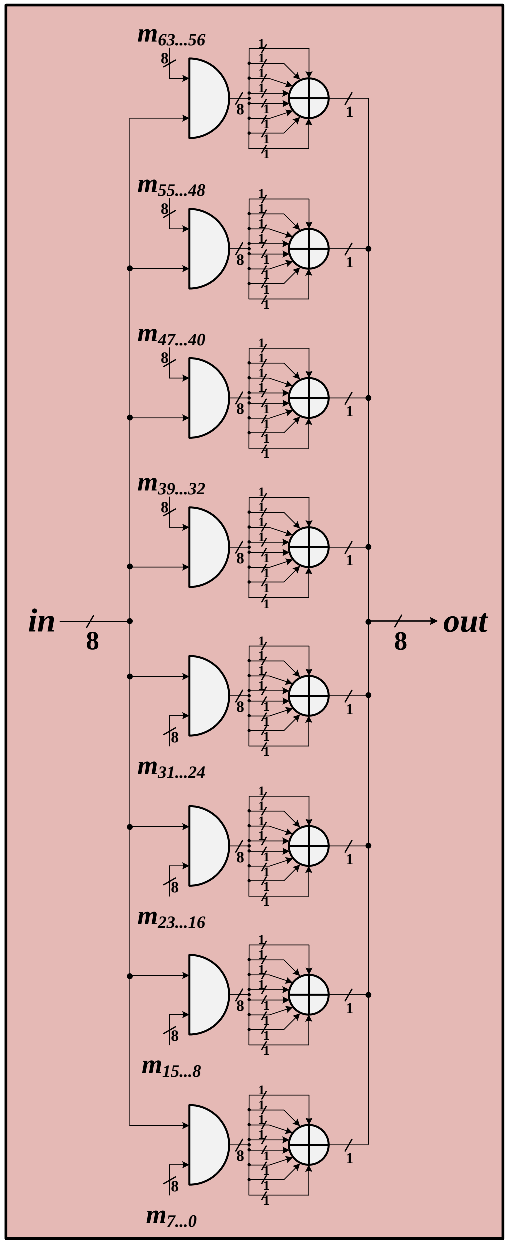
$\{m_0, m_1, \dots, m_{62}, m_{63}\}$
64
$\{in_7, in_6, in_5, in_4, in_3, in_2, in_1, in_0\}$
8
Linear Unit
8
$\{out_7, out_6, out_5, out_4, out_3, out_2, out_1, out_0\}$

Horst Görtz Institute for IT-Security

RUB

Instruction Set Extension: NLU

Hardware – Linear Unit



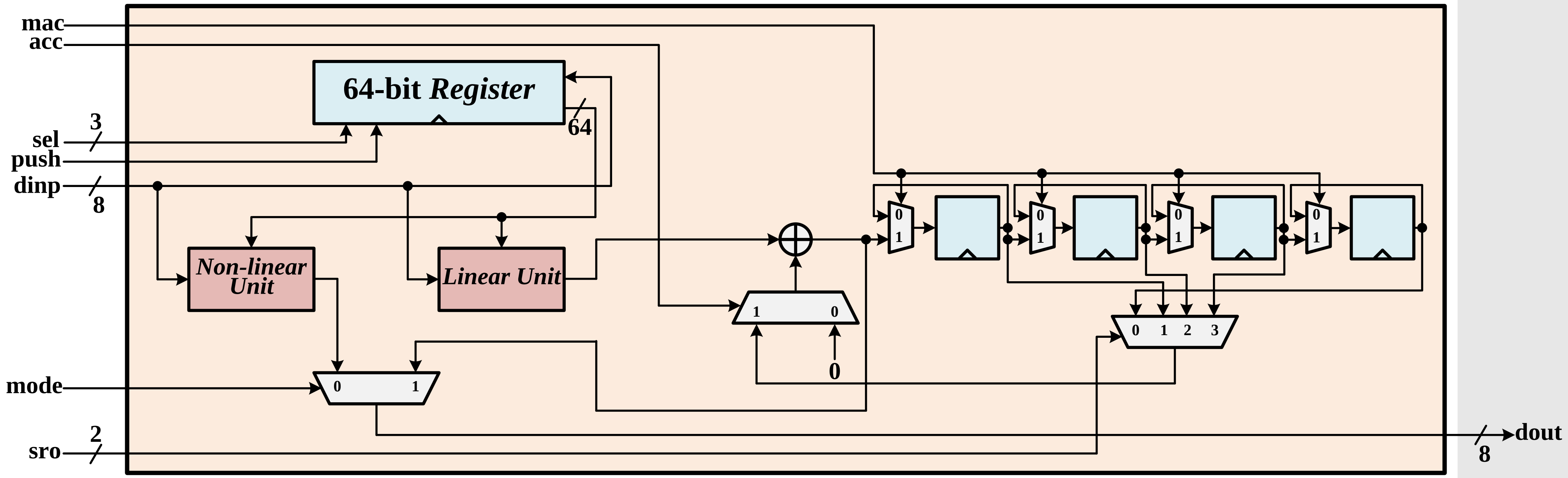
$$\longrightarrow \begin{bmatrix} m_{63} & m_{62} & \cdots & m_{57} & m_{56} \\ m_{55} & m_{54} & \cdots & m_{49} & m_{48} \\ \vdots & & \ddots & & \vdots \\ m_{15} & m_{14} & \cdots & m_{9} & m_{8} \\ m_{7} & m_{6} & \cdots & m_{1} & m_{0} \end{bmatrix} \times \begin{bmatrix} in_7 \\ in_6 \\ \vdots \\ in_1 \\ in_0 \end{bmatrix} = \begin{bmatrix} out_7 \\ out_6 \\ \vdots \\ out_1 \\ out_0 \end{bmatrix}$$

- m_i used for masking

Instruction Set Extension: NLU

Hardware

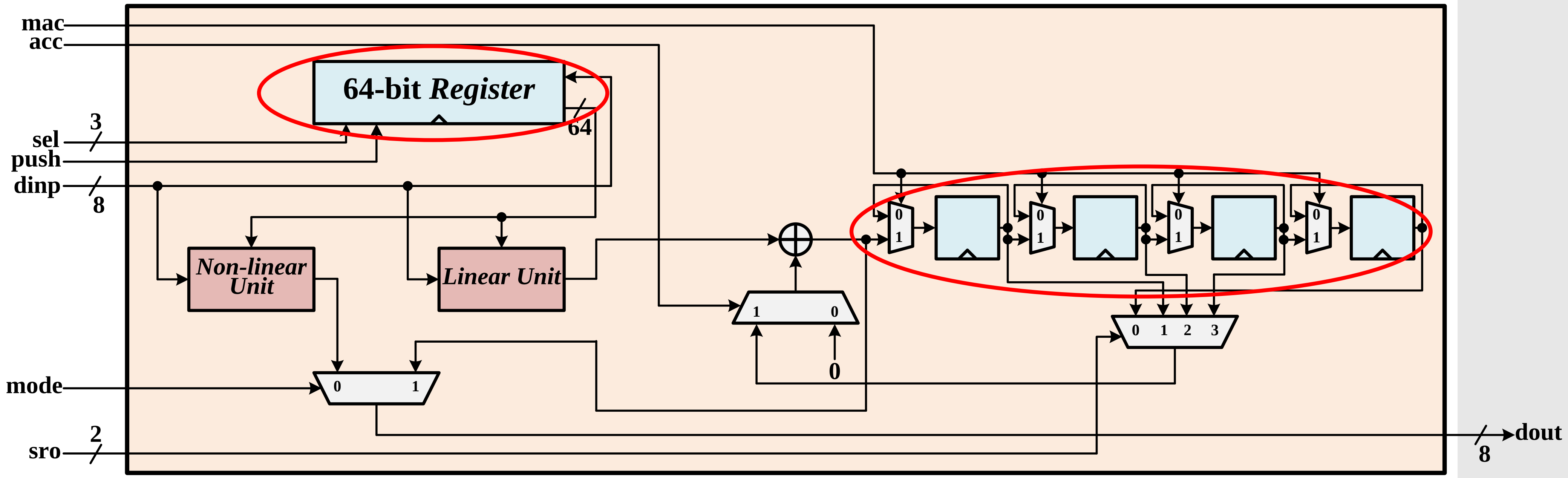
- NLU: Overall unit



Instruction Set Extension: NLU

Hardware

- NLU: Overall unit



Instruction Set Extension: NLU

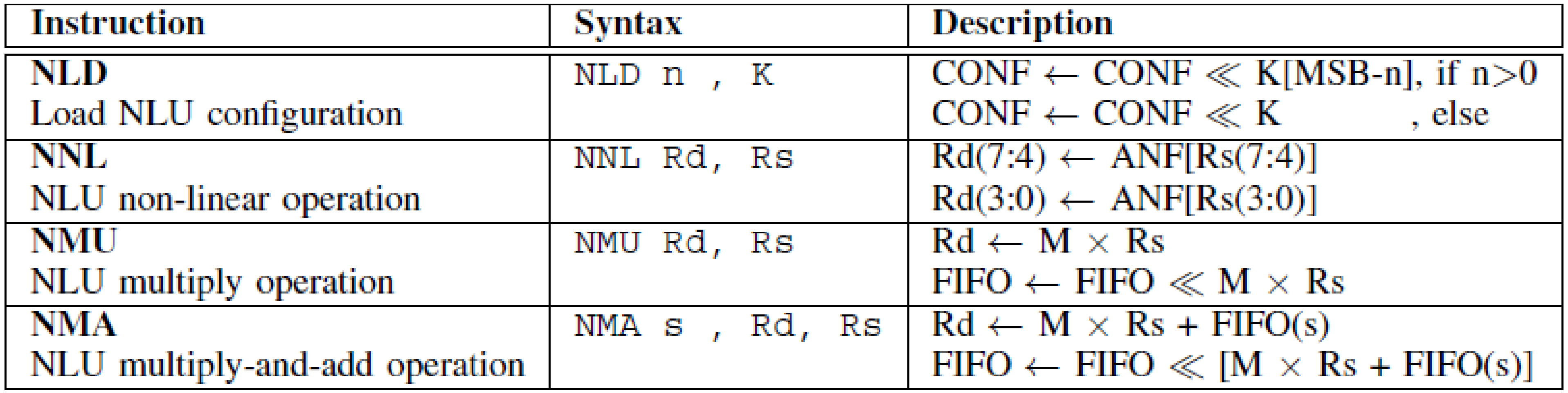
- Shift registers to perform both:
  - $out[n] = mask \times in[n]$
  - $out[n] = (\, mask \times in[n]\,) + out[n\text{-}i]\,, \quad i = 1 \ldots 4$
- *matrix multiply-and-add* form!
  - Used in Present…

Horst Görtz Institute for IT-Security

RUB

Instruction Set Extension: NLU

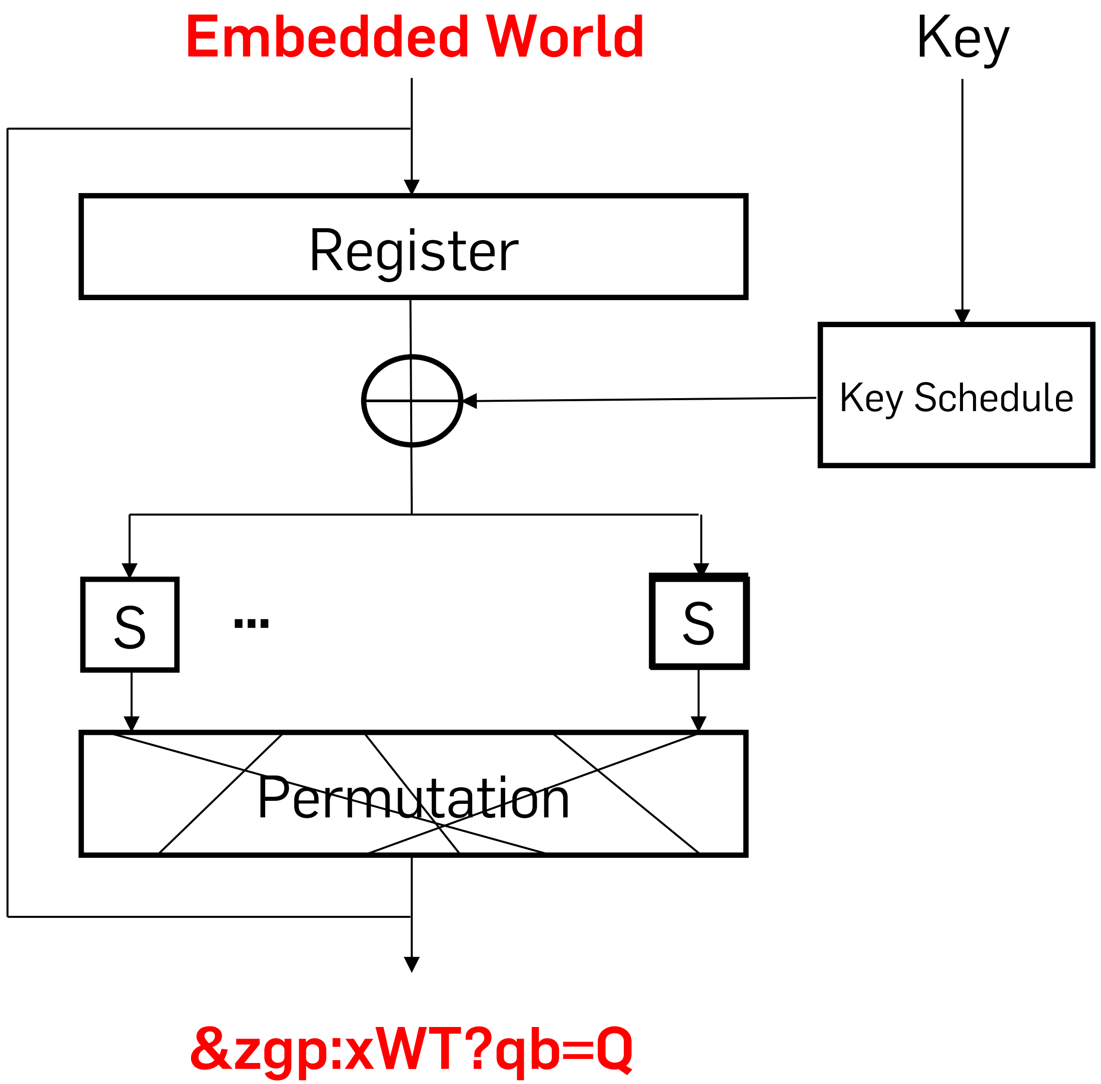
NLU Instructions



Instruction	Syntax	Description
NLD Load NLU configuration	`NLD n , K`	CONF $\leftarrow$ CONF $\ll$ K[MSB-n], if n>0 CONF $\leftarrow$ CONF $\ll$ K , else
NNL NLU non-linear operation	`NNL Rd, Rs`	Rd(7:4) $\leftarrow$ ANF[Rs(7:4)] Rd(3:0) $\leftarrow$ ANF[Rs(3:0)]
NMU NLU multiply operation	`NMU Rd, Rs`	Rd $\leftarrow$ M $\times$ Rs FIFO $\leftarrow$ FIFO $\ll$ M $\times$ Rs
NMA NLU multiply-and-add operation	`NMA s , Rd, Rs`	Rd $\leftarrow$ M $\times$ Rs + FIFO(s) FIFO $\leftarrow$ FIFO $\ll$ [M $\times$ Rs + FIFO(s)]

Instruction Set Extension: NLU

Applications: Present



- 64-bit block size, 80/128-bit key size
- Pure substitution-permutation network
- 4x4-bit S-box
- 31 rounds
- Secure against linear and differential cryptanalyses
- ISO standard!

Instruction Set Extension: NLU

Applications: Present

```
; load ANF bits for the PRESENT S-Box
NLD 0, 0xB3
NLD 0, 0x92
NLD 0, 0x67
NLD 0, 0x0B
NLD 0, 0xDE
NLD 0, 0x43
NLD 0, 0x4A
NLD 0, 0x80

; perform non-linear S-Box operation
NNL r18, r18
NNL r19, r19
NNL r20, r20
NNL r21, r21
NNL r22, r22
NNL r23, r23
NNL r24, r24
NNL r25, r25
```

Instruction Set Extension: NLU

Applications: Present

$$\begin{aligned}
y_{\{63\ldots56\}} &= a_{\{63,59,55,51,47,43,39,35\}}\\
y_{\{55\ldots48\}} &= a_{\{31,27,23,19,15,11,7,3\}}\\
y_{\{47\ldots40\}} &= a_{\{62,58,54,50,46,42,38,34\}}\\
y_{\{39\ldots32\}} &= a_{\{30,26,22,18,14,10,6,2\}}\\
y_{\{31\ldots24\}} &= a_{\{61,57,53,49,45,41,37,33\}}\\
y_{\{23\ldots16\}} &= a_{\{29,25,21,17,13,9,5,1\}}\\
y_{\{15\ldots8\}} &= a_{\{60,56,52,48,44,40,36,32\}}\\
y_{\{7\ldots0\}} &= a_{\{28,24,20,16,12,8,4,0\}}
\end{aligned}$$

$$\downarrow$$

$$\begin{bmatrix} y_{63}\\ y_{62}\\ y_{61}\\ y_{60}\\ y_{59}\\ y_{58}\\ y_{57}\\ y_{56} \end{bmatrix} = \begin{bmatrix} 10000000\\ 00001000\\ 00000000\\ 00000000\\ 00000000\\ 00000000\\ 00000000\\ 00000000 \end{bmatrix} \begin{bmatrix} a_{63}\\ a_{62}\\ a_{61}\\ a_{60}\\ a_{59}\\ a_{58}\\ a_{57}\\ a_{56} \end{bmatrix} \oplus \cdots \oplus \begin{bmatrix} 00000000\\ 00000000\\ 00000000\\ 00000000\\ 00000000\\ 00000000\\ 10000000\\ 00001000 \end{bmatrix} \begin{bmatrix} a_{39}\\ a_{38}\\ a_{37}\\ a_{36}\\ a_{35}\\ a_{34}\\ a_{33}\\ a_{32} \end{bmatrix}$$

Instruction Set Extension: NLU

Applications: Present

$$
\begin{aligned}
Y_7 &= M_{00}A_7 \oplus M_{01}A_6 \oplus M_{02}A_5 \oplus M_{03}A_4\\
Y_6 &= M_{00}A_3 \oplus M_{01}A_2 \oplus M_{02}A_1 \oplus M_{03}A_0\\
Y_5 &= M_{10}A_7 \oplus M_{11}A_6 \oplus M_{12}A_5 \oplus M_{13}A_4\\
Y_4 &= M_{10}A_3 \oplus M_{11}A_2 \oplus M_{12}A_1 \oplus M_{13}A_0\\
Y_3 &= M_{20}A_7 \oplus M_{21}A_6 \oplus M_{22}A_5 \oplus M_{23}A_4\\
Y_2 &= M_{20}A_3 \oplus M_{21}A_2 \oplus M_{22}A_1 \oplus M_{23}A_0\\
Y_1 &= M_{30}A_7 \oplus M_{31}A_6 \oplus M_{32}A_5 \oplus M_{33}A_4\\
Y_0 &= M_{30}A_3 \oplus M_{31}A_2 \oplus M_{32}A_1 \oplus M_{33}A_0
\end{aligned}
$$

Instruction Set Extension: NLU

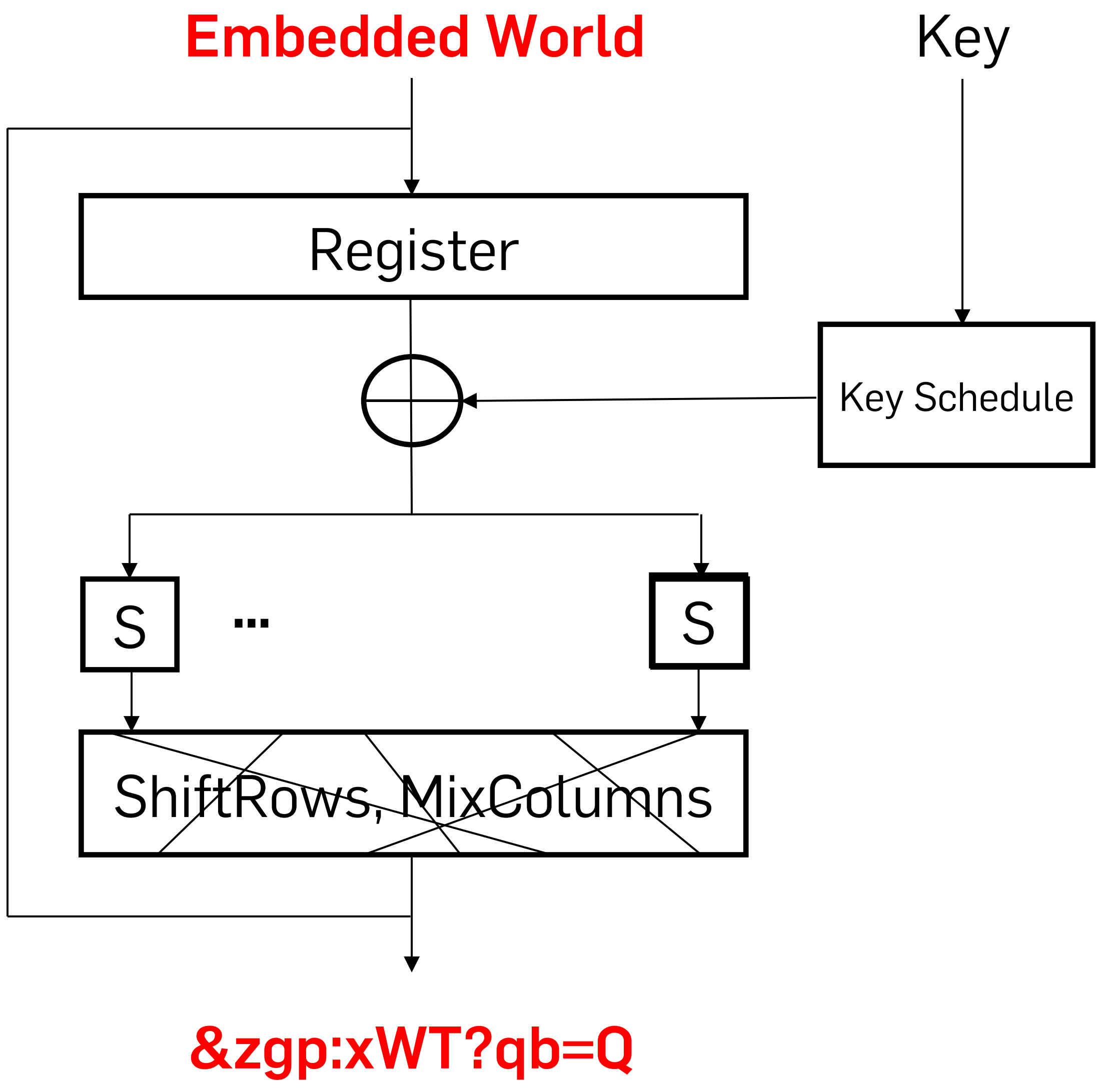
Applications: Present

```
; state is in registers r18 to r25
; write M03 to NLU
NLD 0, 0x00
NLD 0, 0x00
NLD 0, 0x00
NLD 0, 0x00
NLD 0, 0x00
NLD 0, 0x00
NLD 0, 0x80
NLD 0, 0x40
```

```
; temporary registers r10, r11
NMU r10, r21
NMU r11, r25

; write M02 in NLU
NLD 0, 0x00
NLD 0, 0x00
NMA 2, r10, r20
NMA 2, r11, r24

; write M01 in NLU
NLD 0, 0x00
NLD 0, 0x00
NMA 2, r10, r19
NMA 2, r11, r23

; write M00 in NLU
NLD 0, 0x00
NLD 0, 0x00
NMA 2, r10, r18
NMA 2, r11, r22

; write M13 in NLU
NLD 0, 0x40
NLD 0, 0x04
...
```

Instruction Set Extension: NLU

Applications: AES



- 128-bit block size, 128/192/256-bit key size
- Substitution-permutation network
- 8x8-bit S-box
- 10/12/14 rounds
- Secure against linear and differential cryptanalyses
- NIST standard!

Instruction Set Extension: NLU

Applications: AES

$$\begin{array}{lcllll}
Y_0 & = & \{02\}A_0 & \oplus\ \{03\}A_1 & \oplus\ A_2 & \oplus\ A_3 \\
Y_1 & = & A_0 & \oplus\ \{02\}A_1 & \oplus\ \{03\}A_2 & \oplus\ A_3 \\
Y_2 & = & A_0 & \oplus\ A_1 & \oplus\ \{02\}A_2 & \oplus\ \{03\}A_3 \\
Y_3 & = & \{03\}A_0 & \oplus\ A_1 & \oplus\ A_2 & \oplus\ \{02\}A_3
\end{array}$$

$$\begin{bmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{bmatrix} = \begin{bmatrix} 01000000 \\ 00100000 \\ 00010000 \\ 10001000 \\ 10000100 \\ 00000010 \\ 10000001 \\ 10000000 \end{bmatrix} \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 11000000 \\ 01100000 \\ 00110000 \\ 10011000 \\ 10001100 \\ 00000110 \\ 10000011 \\ 10000001 \end{bmatrix} \begin{bmatrix} a_{15} \\ a_{14} \\ a_{13} \\ a_{12} \\ a_{11} \\ a_{10} \\ a_9 \\ a_8 \end{bmatrix} \oplus \begin{bmatrix} a_{23} \\ a_{22} \\ a_{21} \\ a_{20} \\ a_{19} \\ a_{18} \\ a_{17} \\ a_{16} \end{bmatrix} \oplus \begin{bmatrix} a_{31} \\ a_{30} \\ a_{29} \\ a_{28} \\ a_{27} \\ a_{26} \\ a_{25} \\ a_{24} \end{bmatrix}$$

Instruction Set Extension: NLU

RESULTS

- Hardware unit synthesized in UMC 90 nm low-leakage Faraday library
- Area cost:
  - 1752 GE
- Power consumption:
  - 28.59 uW @ 100 KHz

Instruction Set Extension: NLU

RESULTS

- Hardware unit synthesized in UMC 90 nm low-leakage Faraday library
- Area cost:
  - 1752 GE
- Power consumption:
  - 28.59 uW @ 100 KHz

Low-cost!!!

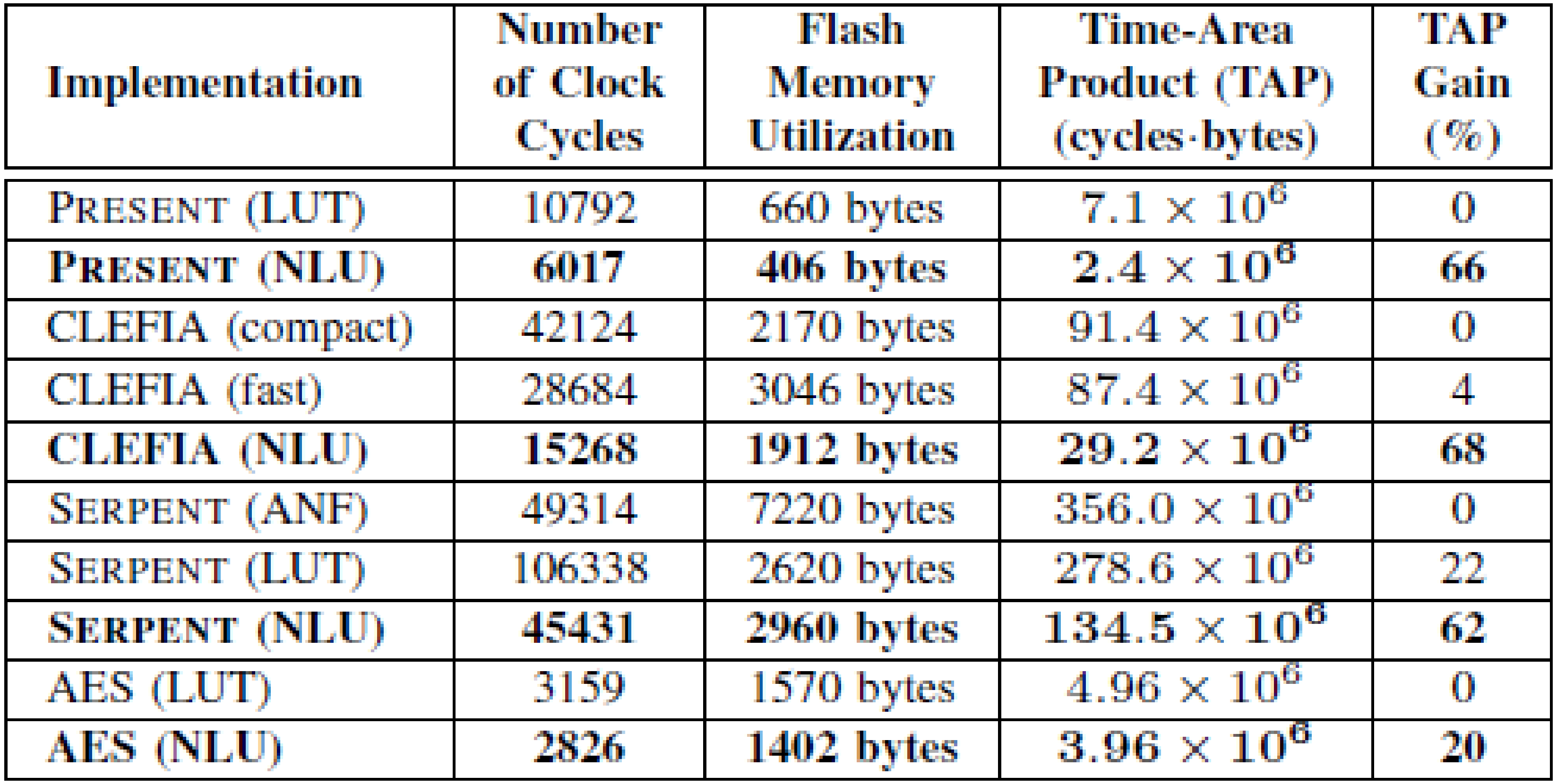
Horst Görtz Institute for IT-Security

RUB

Instruction Set Extension: NLU

RESULTS

- Performance results



Implementation	Number of Clock Cycles	Flash Memory Utilization	Time-Area Product (TAP) (cycles·bytes)	TAP Gain (%)
PRESENT (LUT)	10792	660 bytes	7.1×10^6	0
PRESENT (NLU)	**6017**	**406 bytes**	$\mathbf{2.4 \times 10^6}$	**66**
CLEFIA (compact)	42124	2170 bytes	91.4×10^6	0
CLEFIA (fast)	28684	3046 bytes	87.4×10^6	4
CLEFIA (NLU)	**15268**	**1912 bytes**	$\mathbf{29.2 \times 10^6}$	**68**
SERPENT (ANF)	49314	7220 bytes	356.0×10^6	0
SERPENT (LUT)	106338	2620 bytes	278.6×10^6	22
SERPENT (NLU)	**45431**	**2960 bytes**	$\mathbf{134.5 \times 10^6}$	**62**
AES (LUT)	3159	1570 bytes	4.96×10^6	0
AES (NLU)	**2826**	**1402 bytes**	$\mathbf{3.96 \times 10^6}$	**20**

Overview

- Lightweight and Pervasive Devices
    - Which Devices and Applications?
    - Security Need
- Standard/Lightweight Cryptography
    - Current Solutions
    - Software-oriented Solutions?
- Instruction Set Extension: NLU
    - ISE Model
    - Hardware Unit
    - Applications
- **Conclusion and Future Directions**

Conclusion and Future Directions

- A generic instruction set extension for lightweight block ciphers
- Extremely simple and very low-cost design
- Improves software implementations of lightweight block ciphers
  - Time-area product reductions of 20-70%
- Modular architecture allows it to be used in 4, 16, 32, 64-bit CPUs
  - Possible to go for 32-bit microprocessors in future
- Performance results of more ciphers to be added

Thanks for Listening!

Any Questions?

elif.kavun@rub.de