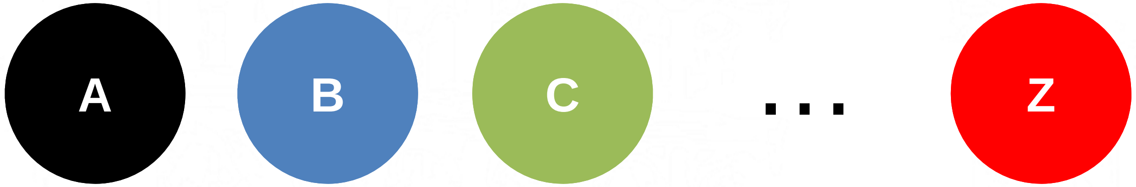
Concealing Secrets in Embedded Processor Designs



Hannes Gross, Manuel Jelinek, Stefan Mangard,
Thomas Unterluggauer, and Mario Werner
Institute for Applied Information Processing and Communications

This work in one slide…

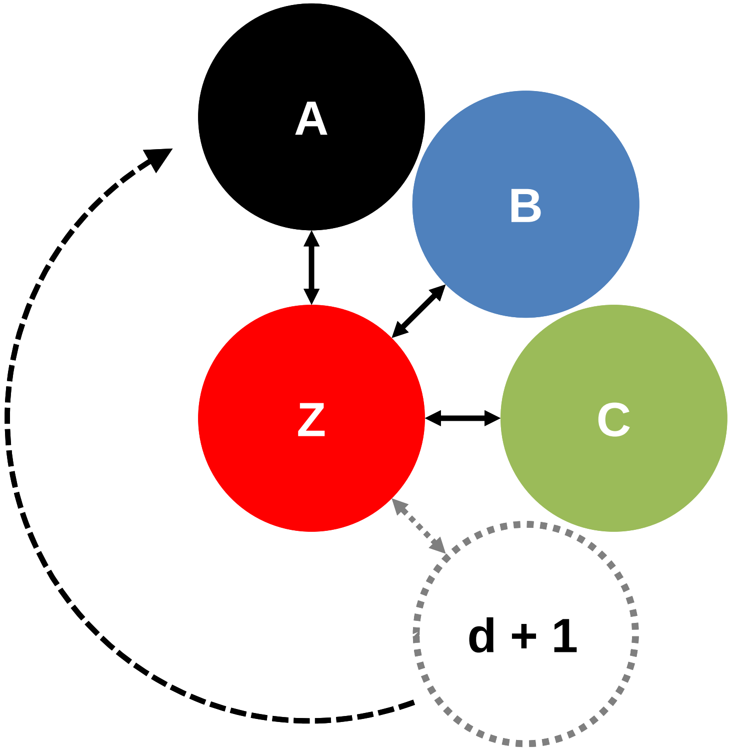
- V-scale processor (RISC-V)

+

- Domain-Oriented Masking

=

- SCA protected V-scale
  - arbitrary protection level
  - flexible and updateable
  - transparent to software designers
  - open source: https://github.com/hgrosz/vscale_dom



This work in numbers…

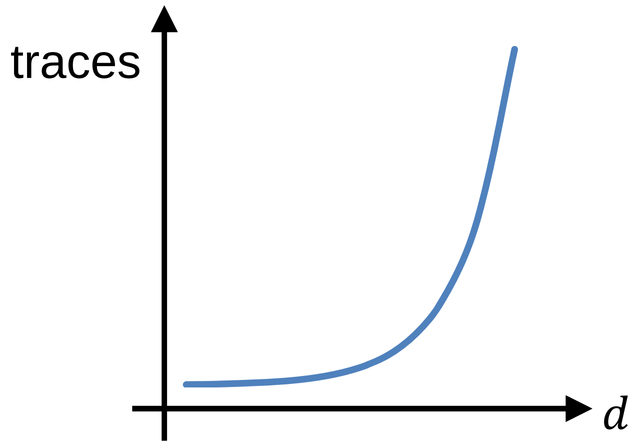
	Unprotected		1st-Order		2nd-order
LUTs [1]	2.6 k	+ 57%	4.1 k	+ 37%	5.6 k
registers [1]	1.0 k	+ 80%	1.8 k	+ 39%	2.5 k
random bits	0		64	$32 * d(d+1)$	192
pipeline stages	3		4		4

[1] for Xilinx Kintex-7 FPGA

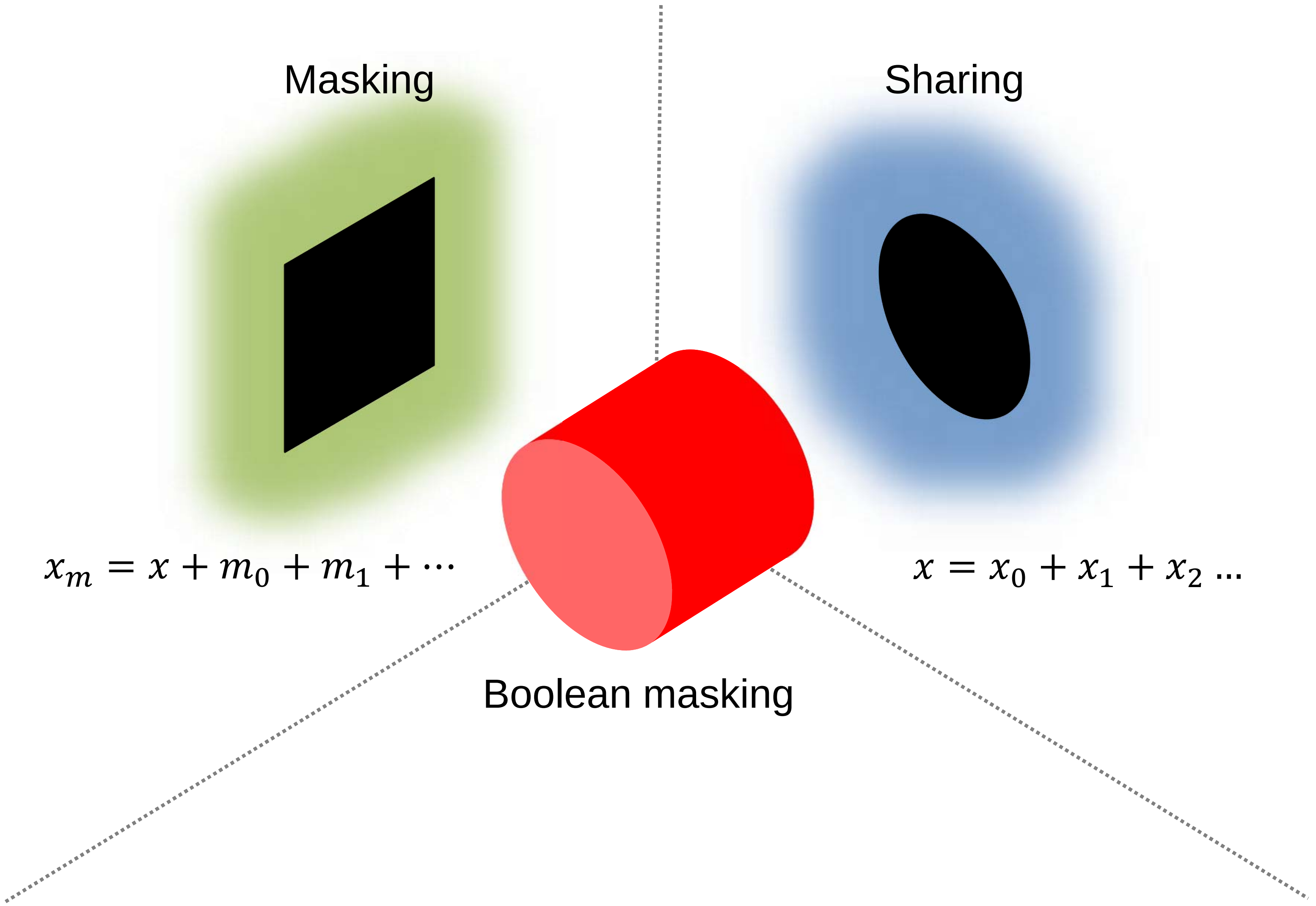
Motivation

Masking is…

- ☺ very effective SCA countermeasure
- ☹ cumbersome
- ☹ error prone
- ☹ requires expertise
- ☹ lots of evaluation work
- ☹ for specific implementations
- ☹ decomposition of complex functions
  → slows down the implementation



Boolean Masking from Different Perspectives



Masking

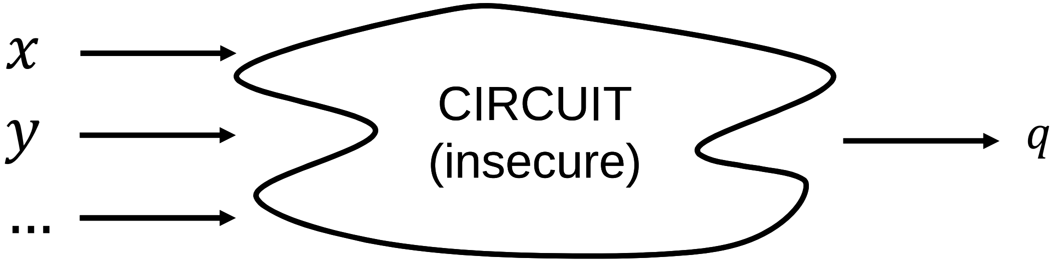
Sharing

$x_m = x + m_0 + m_1 + \cdots$

$x = x_0 + x_1 + x_2 \ldots$

Boolean masking

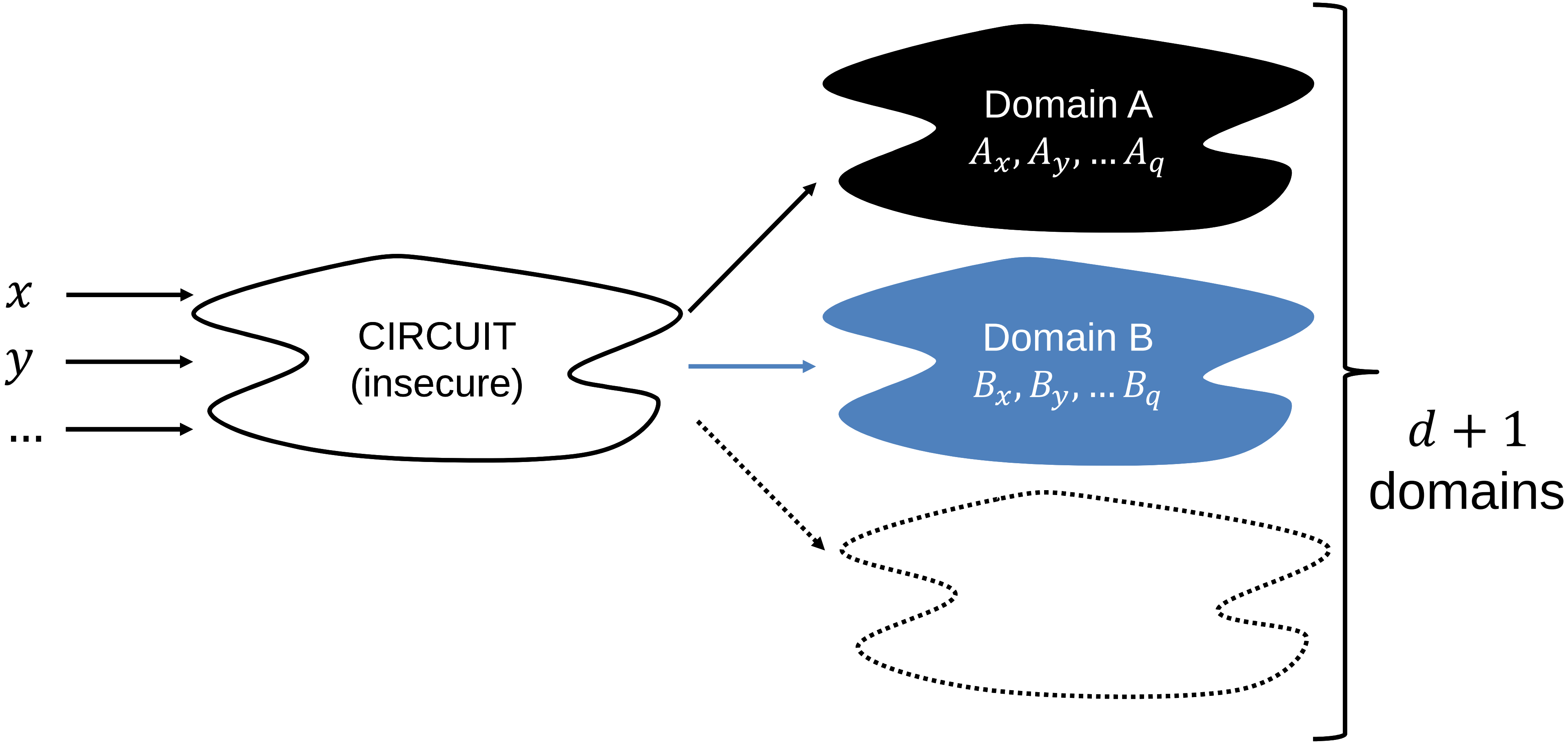
Domain-Oriented Masking



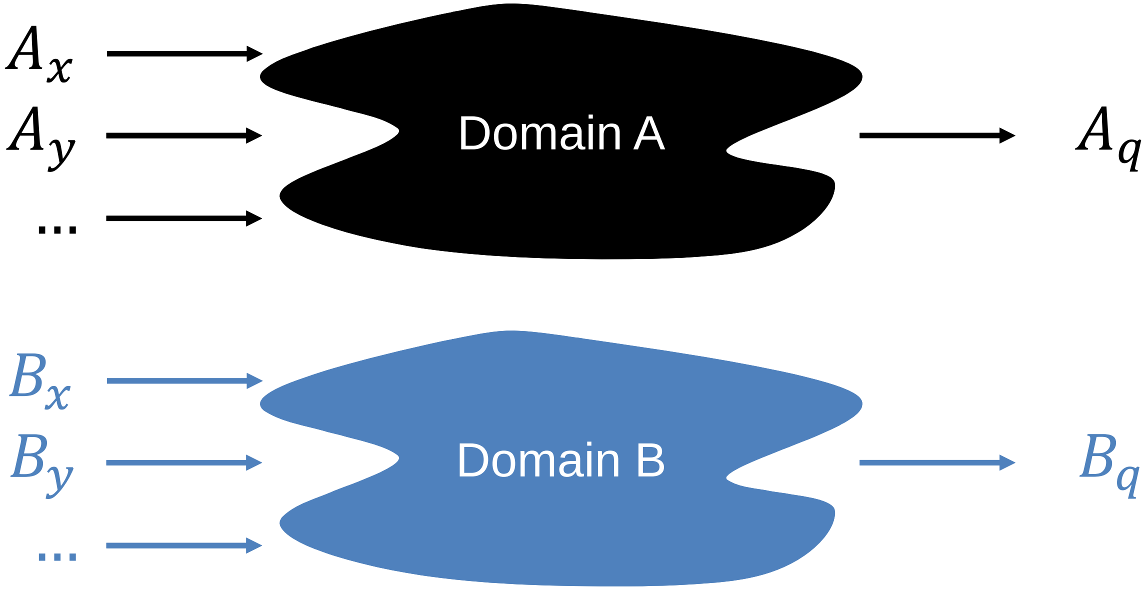
Domain-Oriented Masking

$$\begin{tikzcd}
 & A_x \overset{R}{\leftarrow} GF(2^n) \arrow[dr] & \\
x \arrow[ur] \arrow[r] \arrow[dr, dashed] & B_x \overset{R}{\leftarrow} GF(2^n) \arrow[r] & x \\
 & \dots \arrow[ur, dashed] &
\end{tikzcd}$$

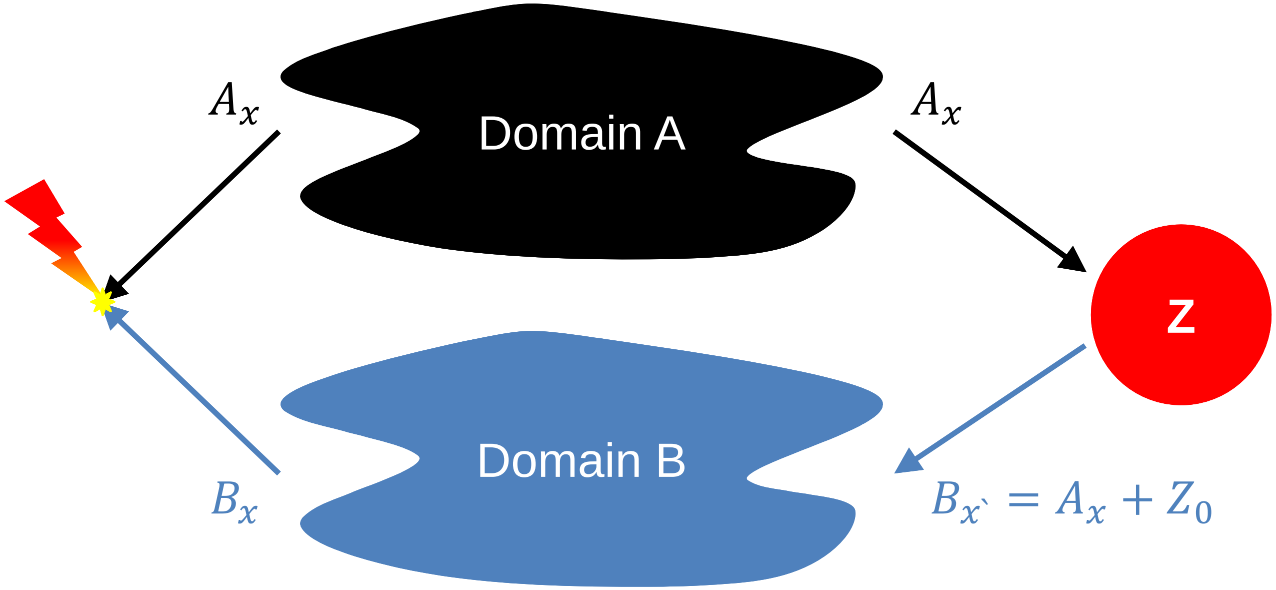
Domain-Oriented Masking



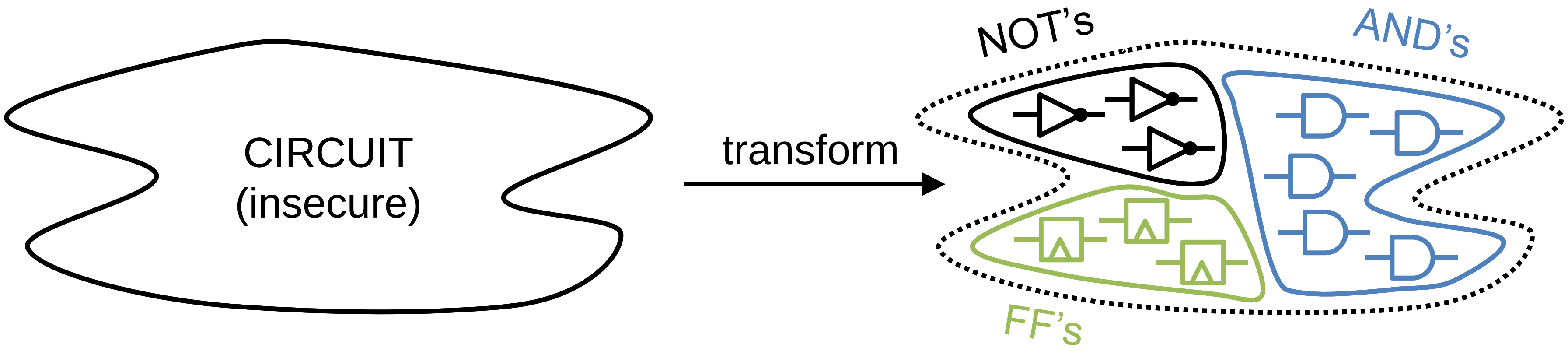
Linear Operations



Nonlinear Operations



Protecting Arbitrary Circuits



CIRCUIT
(insecure)

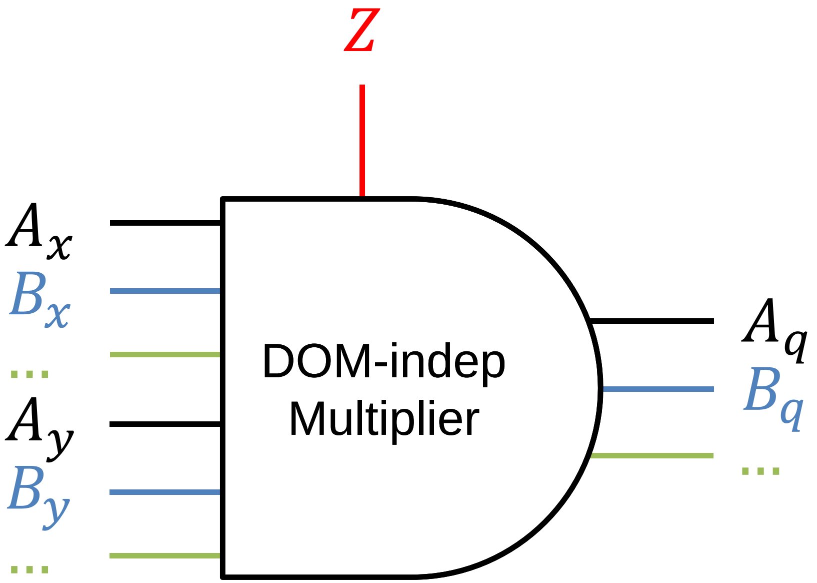
transform

NOT's

AND's

FF's

d^{th}-Order Secure AND Gate



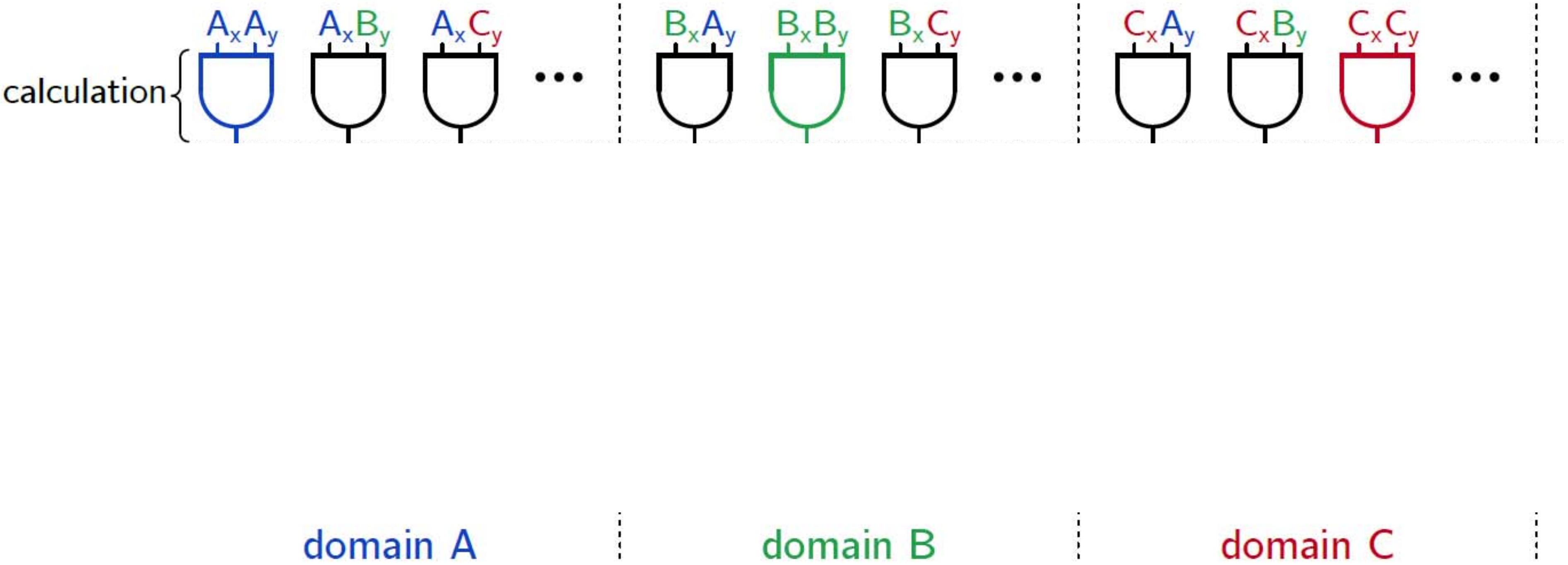
1. Calculation → 2. Resharing → 3. Integration

1. Calculation

$$q = xy = (A_x + B_x + C_x + \cdots)(A_y + B_y + C_y + \cdots)$$

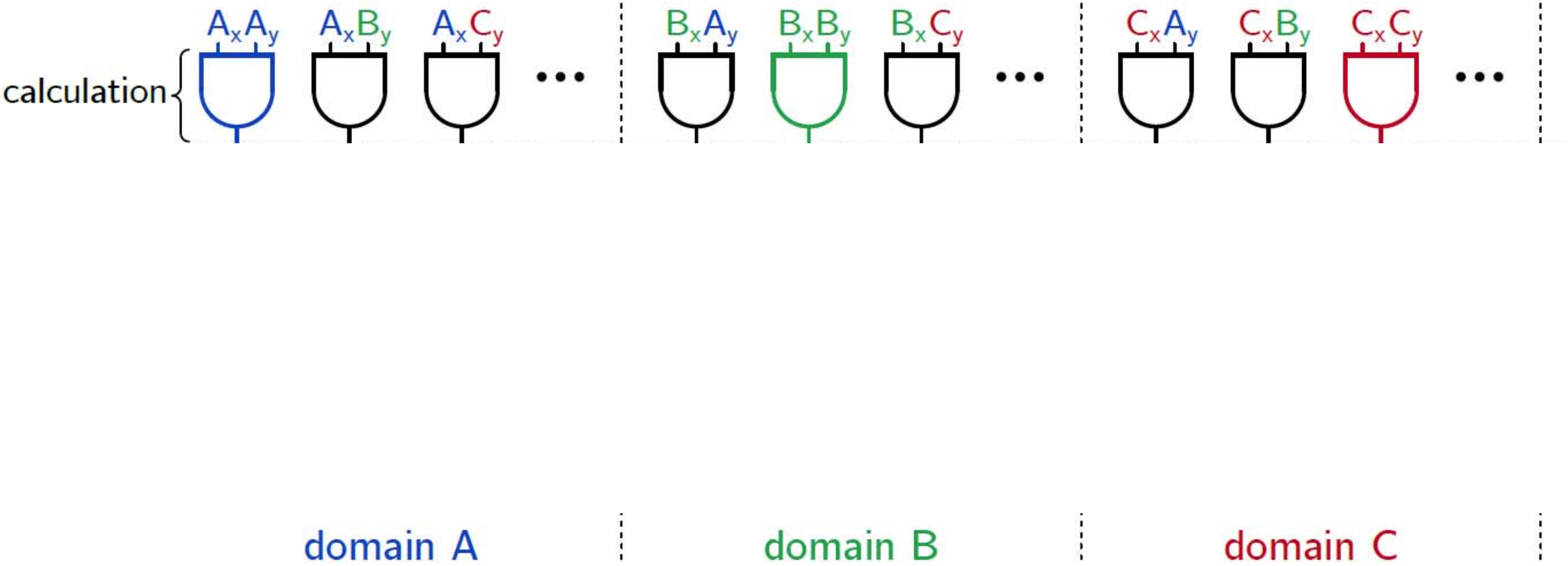
1. Calculation

$$q = xy = (A_x + B_x + C_x + \cdots)(A_y + B_y + C_y + \cdots)$$



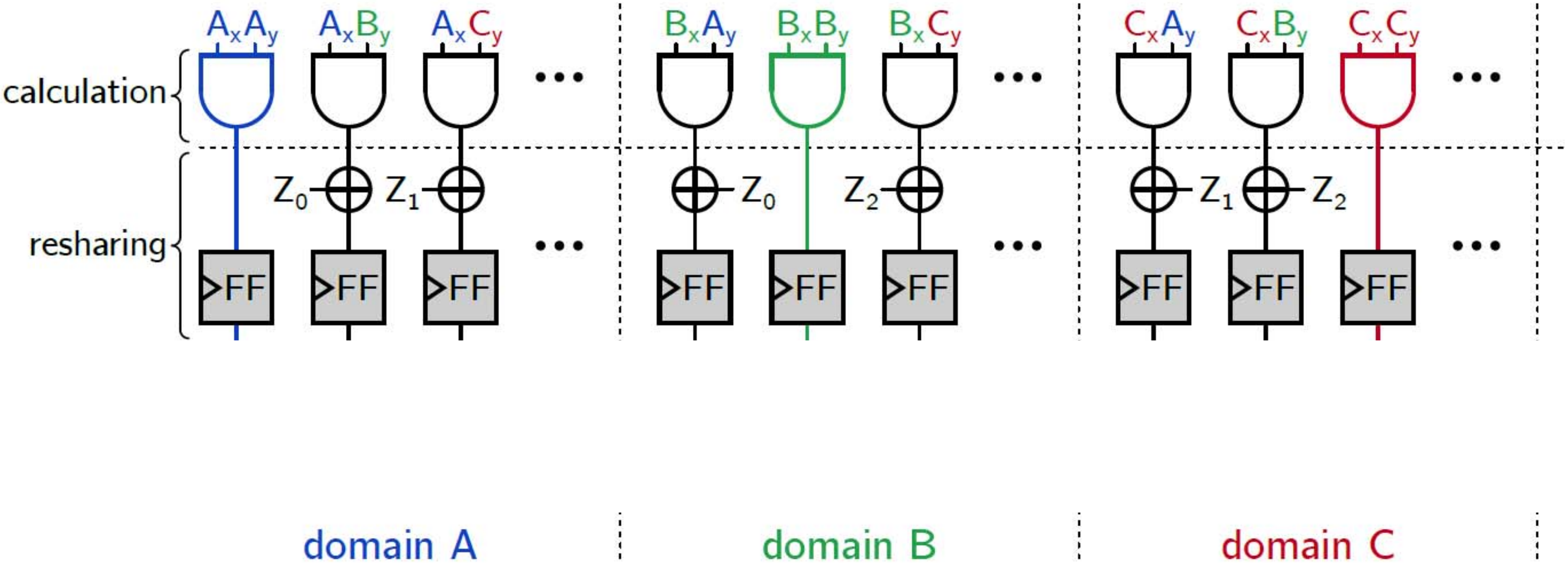
2. Resharing

$$\begin{array}{llll} \boldsymbol{A_x A_y} & (A_x B_y + Z_0) & (A_x C_y + Z_1) & \cdots \\ (B_x A_y + Z_0) & \boldsymbol{B_x B_y} & (B_x C_y + Z_2) & \cdots \\ (C_x A_y + Z_1) & (C_x B_y + Z_2) & \boldsymbol{C_x C_y} & \cdots \end{array}$$



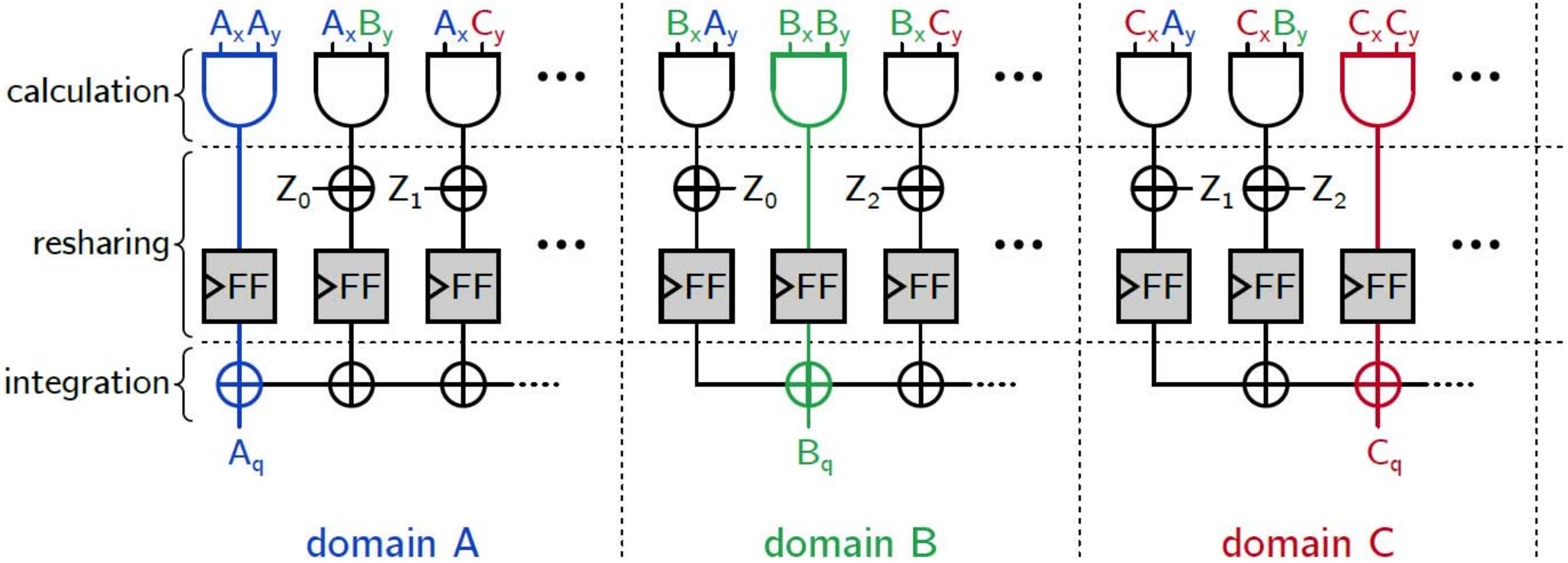
2. Resharing

$$\begin{array}{cccc} \boldsymbol{A_x A_y} & (A_x B_y + Z_0) & (A_x C_y + Z_1) & \cdots \\ (B_x A_y + Z_0) & \boldsymbol{B_x B_y} & (B_x C_y + Z_2) & \cdots \\ (C_x A_y + Z_1) & (C_x B_y + Z_2) & \boldsymbol{C_x C_y} & \cdots \end{array}$$



3. Integration

$$\boldsymbol{A_x A_y} + (A_x B_y + Z_0) \ + (A_x C_y + Z_1) + \cdots$$
$$(B_x A_y + Z_0) + \boldsymbol{B_x B_y} + (B_x C_y + Z_2) + \cdots$$
$$(C_x A_y + Z_1) + (C_x B_y + Z_2) + \boldsymbol{C_x C_y} + \cdots$$



RISC-V ISA

- free and open RISC ISA
- register sizes 32, 64 or 128 bit
- only base integer instructions (I, E) mandatory
- lots of extensions
  - multiplication/division (M)
  - atomic operations (A)
  - single- (F) and double-precision (D) floating point ops
  - compressed instructions (C)
  - extensions (X)
- no flags

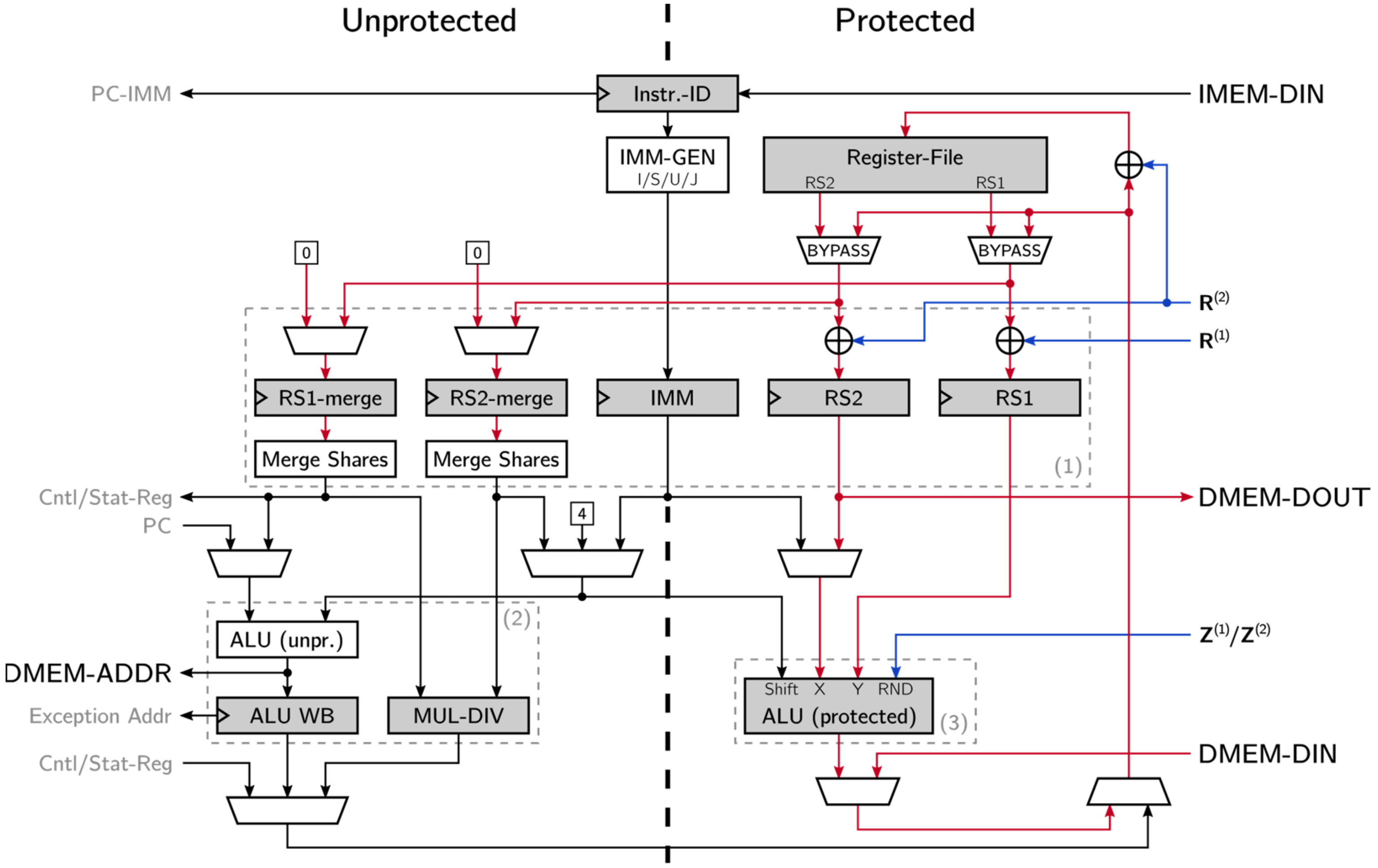
V-scale Processor

- RV32IM instruction set
- 32 x 32-bit registers
- single-issue in-order 3-stage pipeline
- combined decode & execute stage
- write back stage with bypass functionality
- AHB-Lite interface → either Harvard or von Neumann
- open source
  https://github.com/ucb-bar/vscale/

DOM Protected V-scale Processor

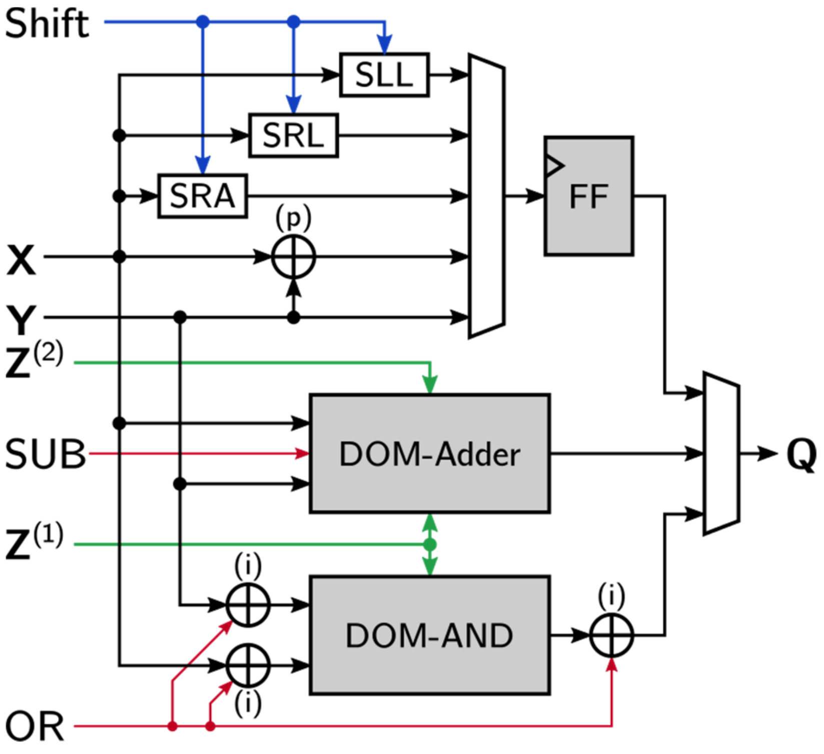
- High-level overview of changes
  - Protected (shared) parts
    - "I" instructions
    - data memory interface
    - register file
  - Unprotected parts
    - "M" instructions
    - instruction memory
    - instruction decoder
    - program counter

DOM Protected V-scale Processor

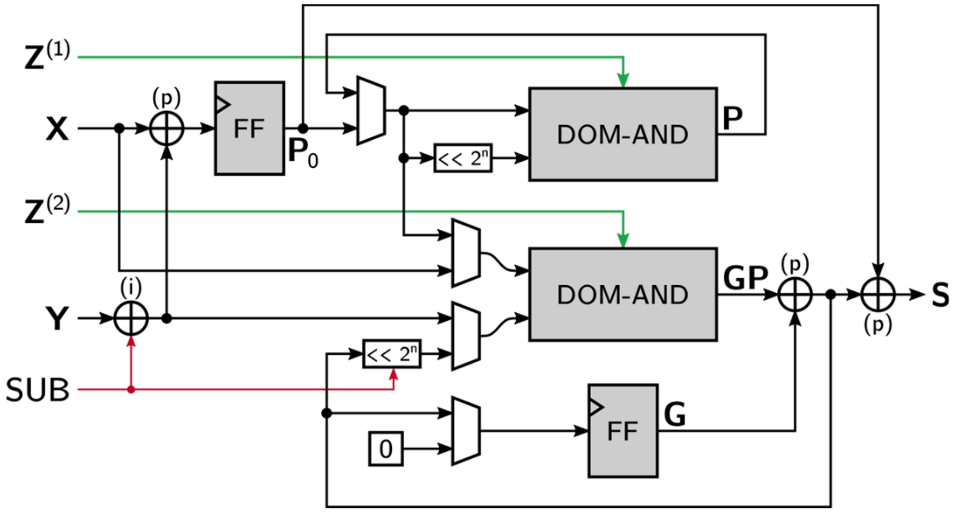


Protected ALU

- Linear functions
  - Shifts
  - XOR
- Nonlinear functions
  - AND (OR)
  - Adder
- Two fresh random Z’s

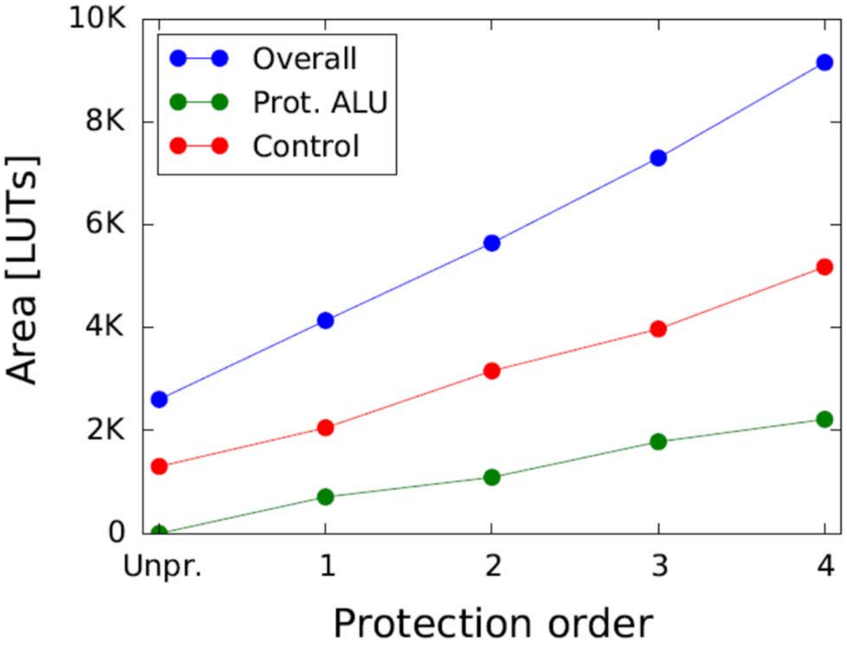


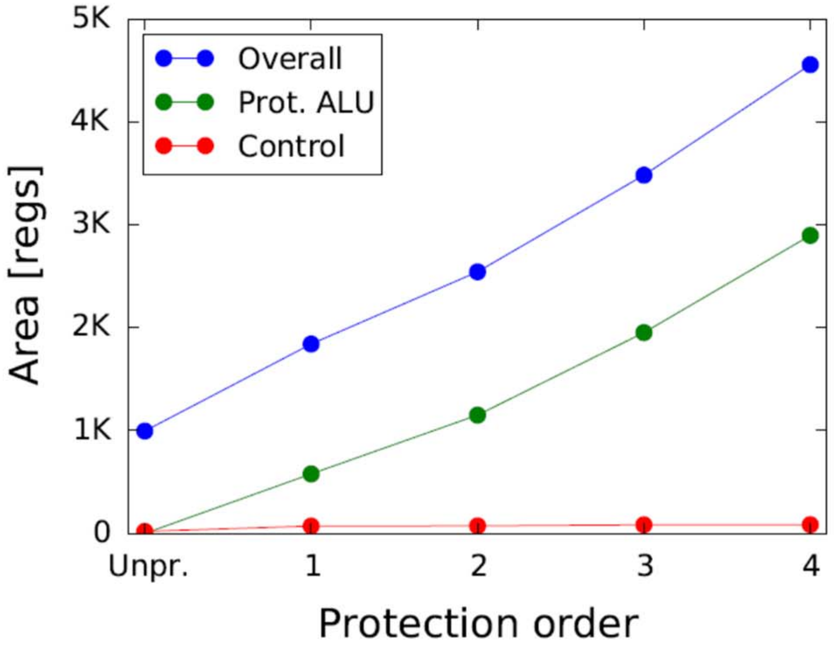
Protected Adder



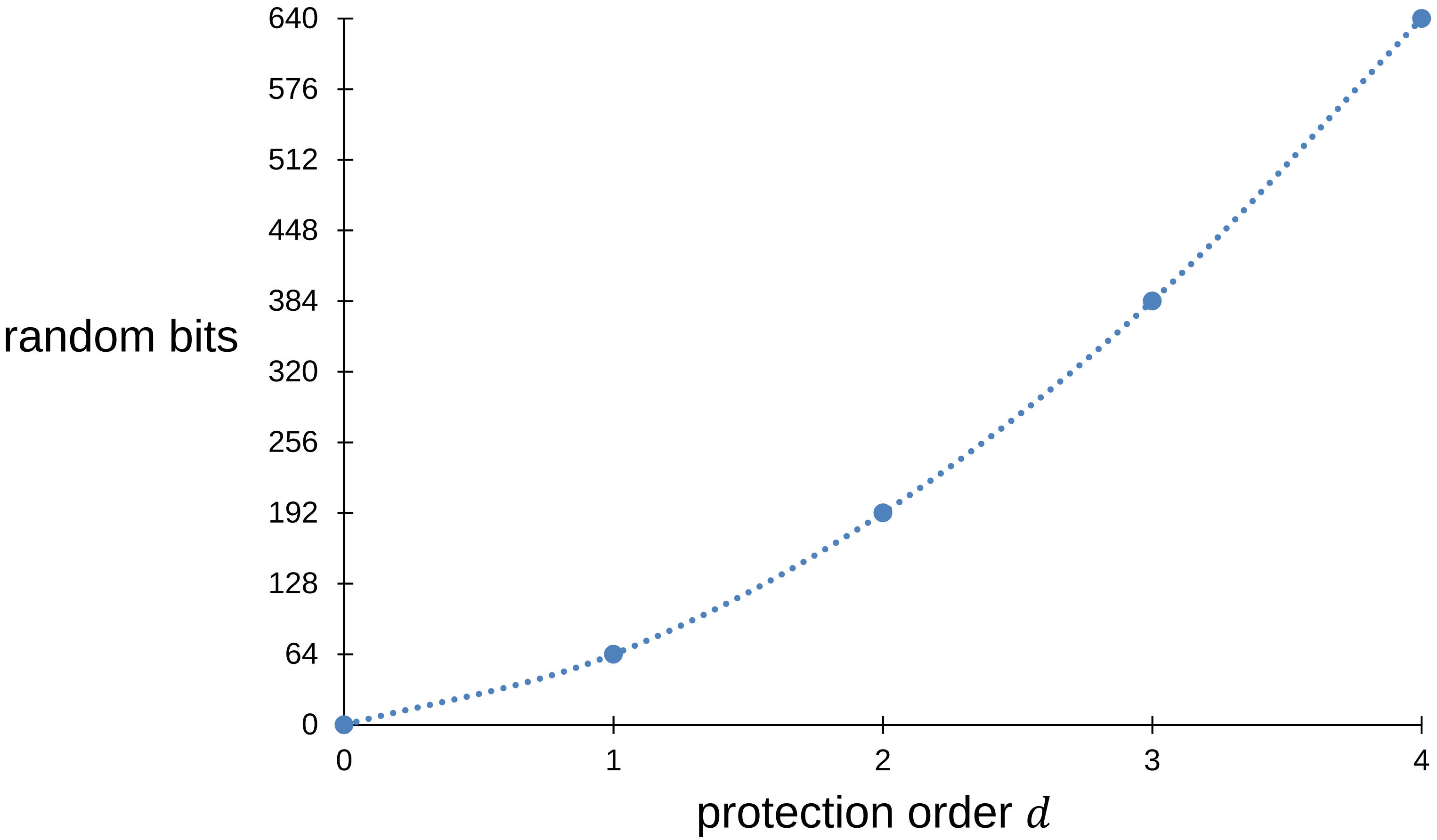
- Kogge-Stone Adder
- Calculation split into “generate” and “propagate”
- Logarithmic runtime (init. + 5 steps + postproc.)
- Two Z shares

Results

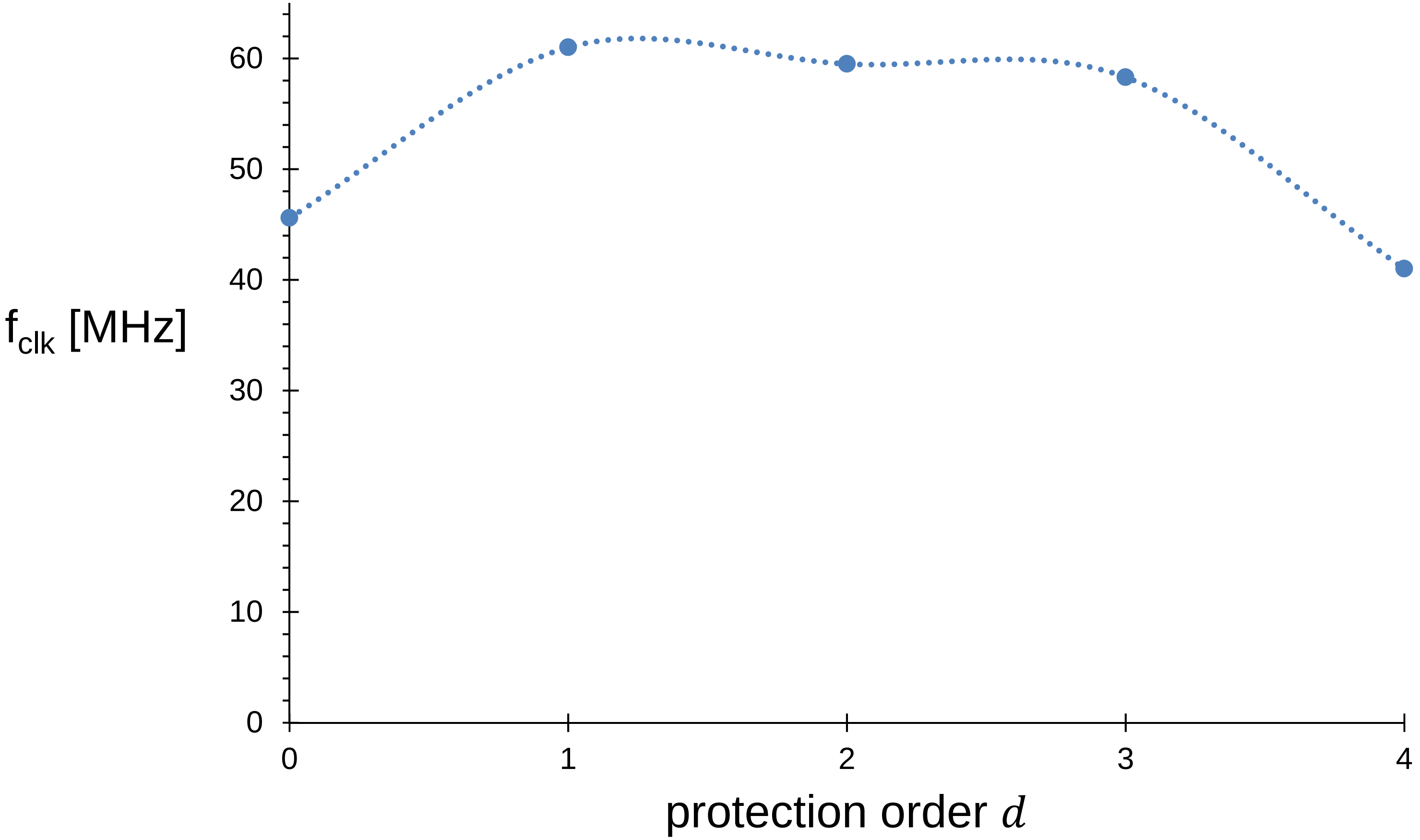

10K
8K
6K
4K
2K
0
Area [LUTs]
Overall
Prot. ALU
Control
Unpr.
1
2
3
4
Protection order


5K
4K
3K
2K
1K
0
Area [regs]
Overall
Prot. ALU
Control
Unpr.
1
2
3
4
Protection order

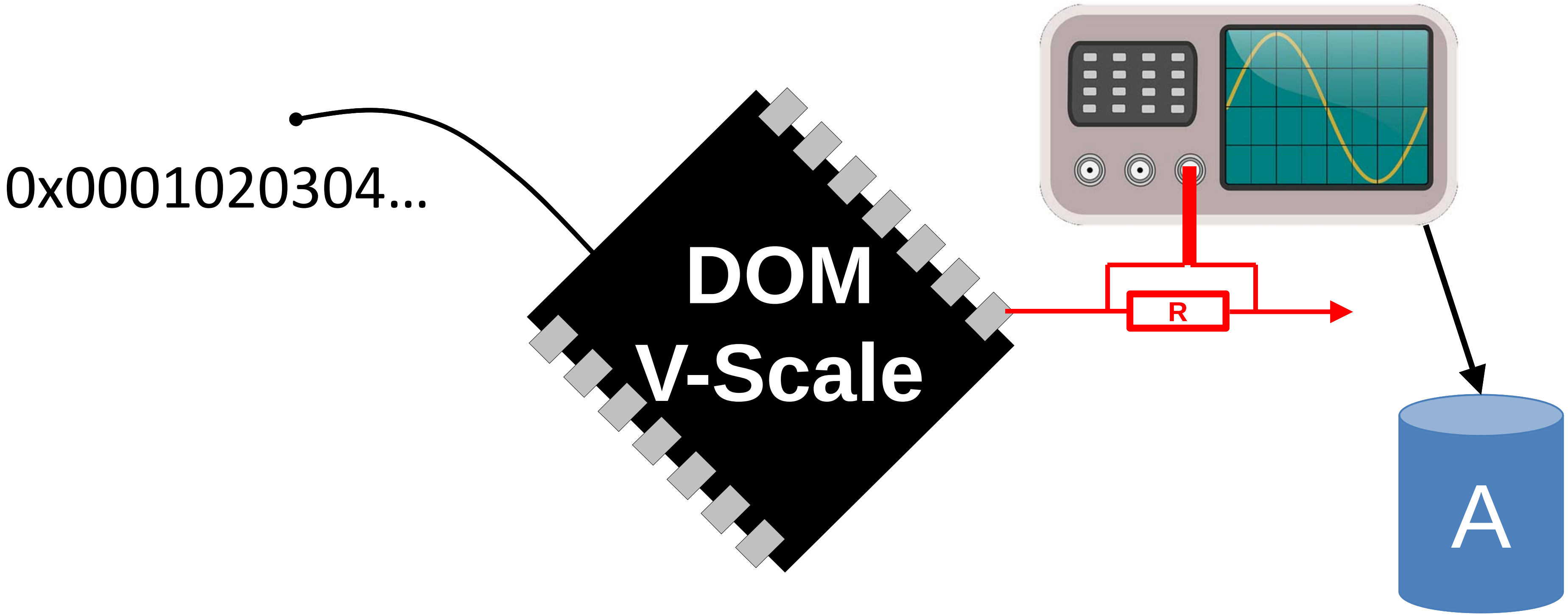
Required Randomness



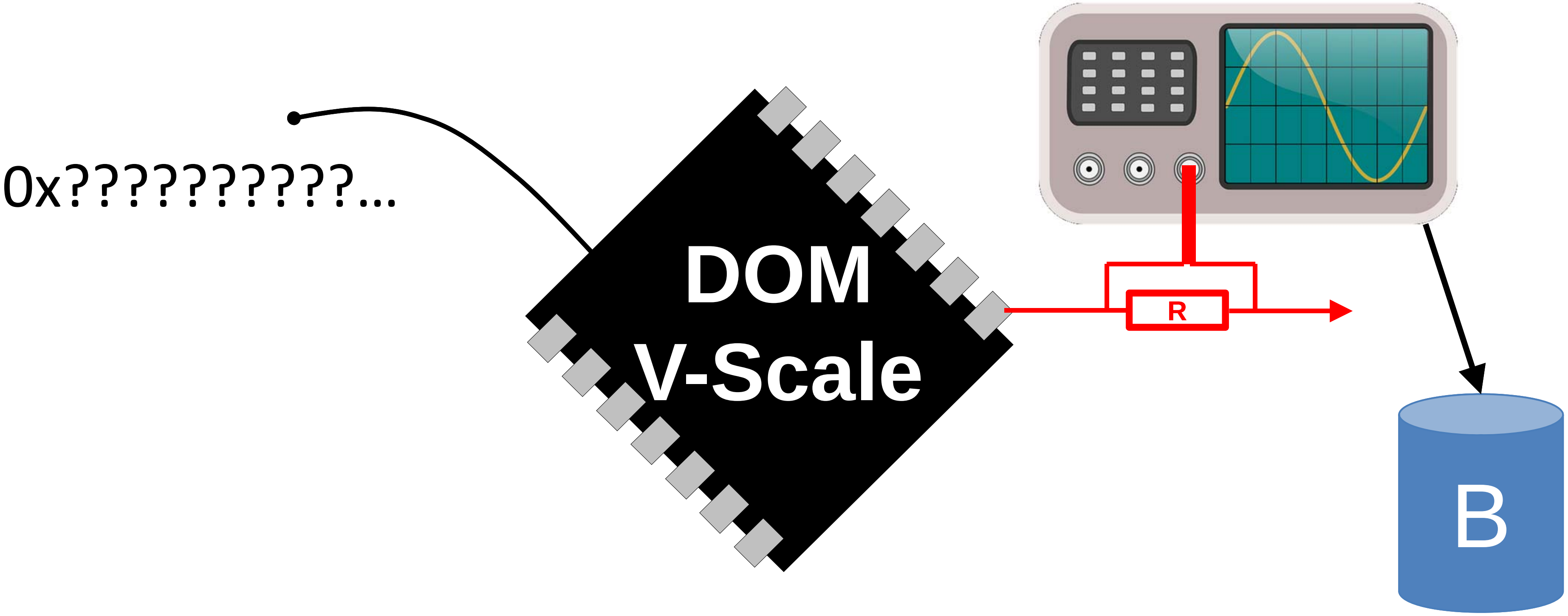
Influence on the Maximum Clock



T-test – 1. Collect Traces for Constant Input



T-test – 2. Collect Traces for Constant Input



T-test – 3. Calculate “t” Value

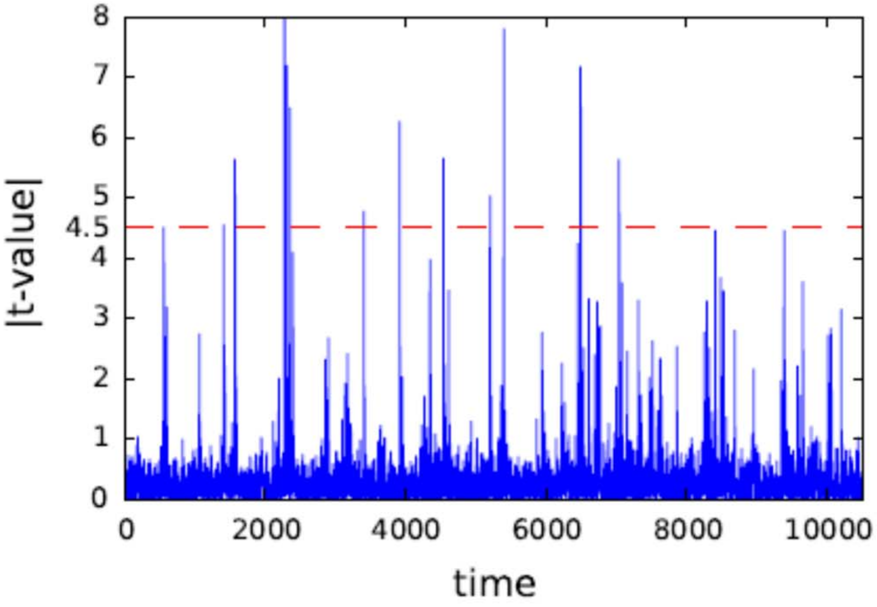
$$t = \frac{\overline{A} - \overline{B}}{\sqrt{\frac{S_A^2}{|A|} + \frac{S_B^2}{|B|}}}$$

Null hypothesis: both trace sets have equal mean

Pass criterion |t| < 4.5 for > 99.999% confidence

otherwise fail

T-test – Result



Conclusions

- SCA resistant RISC-V processor
- DOM for arbitrary protection level

☺ Advantages

- more flexible
- transparent for SW designers
- inherently a lot of noise
- faster development of secure systems
- faster than SW based masking

Conclusions

☹ Drawbacks

- requires a lot of randomness
- slower than dedicated HW solutions
- does not seal all leakages sources

Concealing Secrets in Embedded Processor Designs

The HECTOR project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 644052. This work was partially supported by the TU Graz LEAD project "Dependable Internet of Things in Adverse Environments". This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 681402).

European Research Council
Established by the European Commission

Hannes Gross, Manuel Jelinek, Stefan Mangard,
Thomas Unterluggauer, and Mario Werner
Institute for Applied Information Processing and Communications