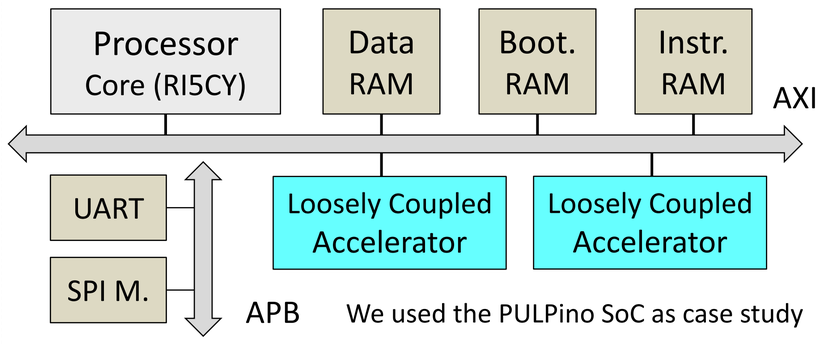
PAGURUS: Low-Overhead Dynamic Information Flow Tracking on Loosely Coupled Accelerators

Luca Piccolboni, Giuseppe Di Guglielmo, Luca Carloni

Columbia University, New York, NY, USA

Systems-on-Chip (SoCs) are Heterogeneous

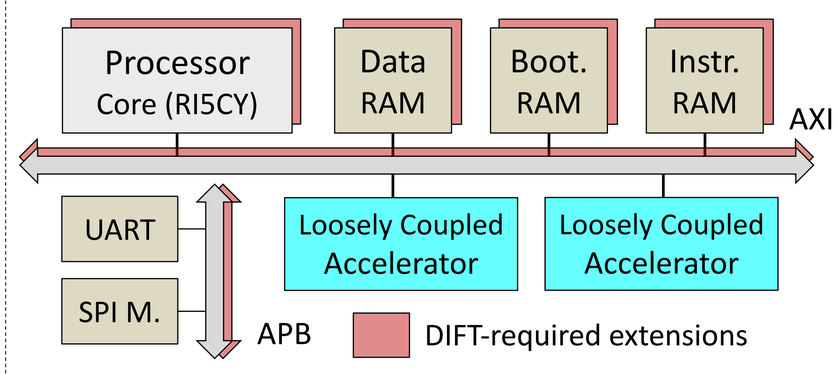


Why **heterogeneous**?
- High Performance
- Energy Efficiency

Hardware Accelerators?
- They are components designed to perform a specific functionality

Hardware Accelerators: A Way To Compromise SoCs

Hardware accelerators are **vulnerable**: even if the rest of the SoC is secured with DIFT, accelerators can be used to implement software-based attacks similar to the ones DIFT should protect the SoC from. WHY? Accelerators don't propagate tags.



We extended PULPino with DIFT (C. Palmiero et al., IEEE HPEC'18)

⇩

We compromised it through an accelerator

⇩

We proposed PAGURUS to avoid these attacks

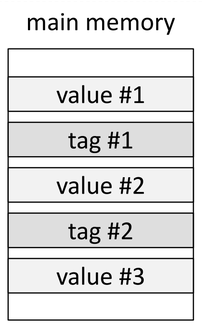
Dynamic Information Flow Tracking (DIFT)

DIFT protects against software-based attacks
tags: mark the untrustworthy data/channels

Buffer-Overflow Attack

```
int buff[10], k;
int (*fun)(int) = foo;
int num = atoi(argv[1]);
int val = atoi(argv[2]);
/* this is a bad idea */
for (k = 0; k < num; ++k)
  buff[k] = sw(val);
fun(1); /* call foo? */
```

main memory	tags
val = 7	1
num = 11	0
fun = sw(7)	1
buff[9] = sw(7)	1
...	
buff[0] = sw(7)	1



tag density = # tags interleaved in memory

tag offset = # words in main memory between two consecutive tags
- **interleaved scheme** (it is a variation of the decoupled scheme)

Policy: what to do with the marked data
Mechanism: propagates&checks the tags

Buffer-Overflow Attack

- implemented through an accelerator called in *hw()*

```
int buff[10];
int (*fun)(int) = foo;   // 0xAA
int num = atoi(argv[1]);
int val = atoi(argv[2]);
/* this is a bad idea */
hw(num, val, buff);
fun(1); /* call foo? */
```

main memory	tags
val = 7	1
num = 11	0
fun = 0xAA	1
buff[9] = 0	1
...	
buff[0] = 0	1

before *hw()*

main memory	tags
val = 7	1
num = 11	0
fun = sw(7)	0
buff[9] = sw(7)	0
...	
buff[0] = sw(7)	0

after *hw()*

The accelerator can't **propagate** the tags!

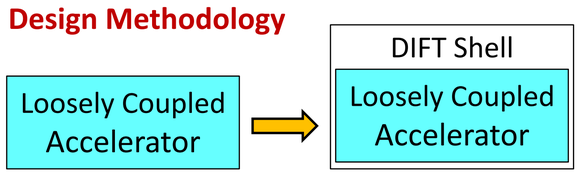
An arbitrary function can be run instead of the function *foo()*.

DIFT fails in the case of heterogen. SoCs.

See the paper for more details!

PAGURUS: Designing a DIFT Shell to add DIFT Support to Accelerators

Design Methodology



- the DIFT shell can be automatically generated
- the DIFT has the same interface of the acceler.

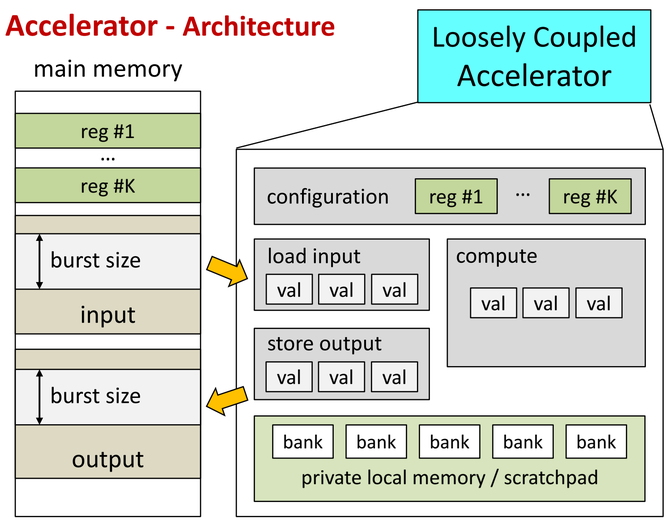
DIFT Shell - Property #1:
the design of the DIFT shell is independent from the design of the accelerator and vice versa

DIFT Shell - Property #2:
- negligible cost (area) overhead
- tunable performance overhead

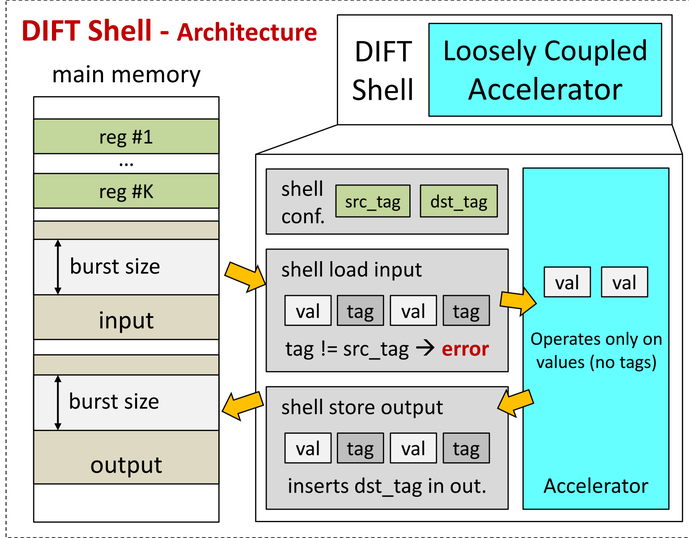
Tag Randomization

- we keep a fixed distance between two tags in mem
- we randomize the position of the first tag at every ex.
- we add a register to make the shell aware of the off.

Accelerator - Architecture



DIFT Shell - Architecture

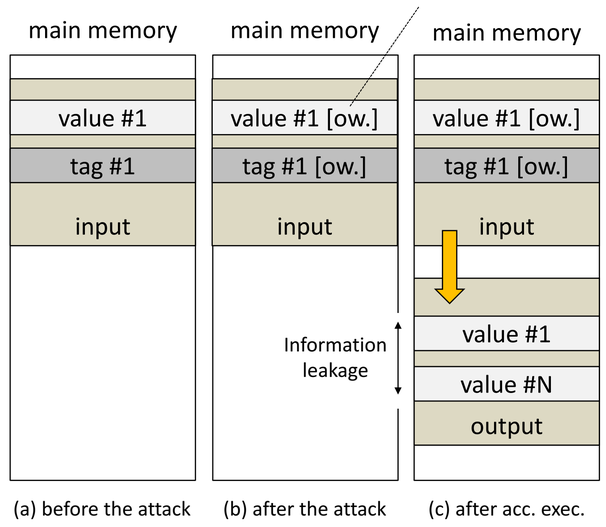


A Security Metric: Information Leakage

Metric Definition

Information Leakage: amount of data that can be produced as output by an accelerator before its shell realizes that the input has been corrupted.

Evaluating the Metric



(a) before the attack (b) after the attack (c) after acc. exec.

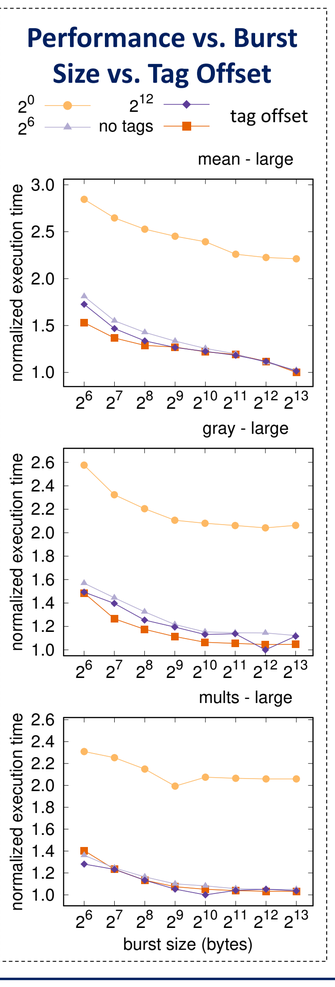
For evaluating the metric, for the experimental results, we assume that the first tag in memory has been replaced and we count the number of outputs the accelerator produce.

Metric Analysis

The information leakage depends on four factors:
- Factor #1: **tag offset**
  larger tag offset → higher info. leakage
- Factor #2: **acc. algorithm**
  smaller amount of data to compute an output value → higher info. leakage
- Factor #3: **acc. implemen.**
  smaller burst size → higher info. leakage
- Factor #4: **acc. workload**
  larger workload size → higher info. leakage

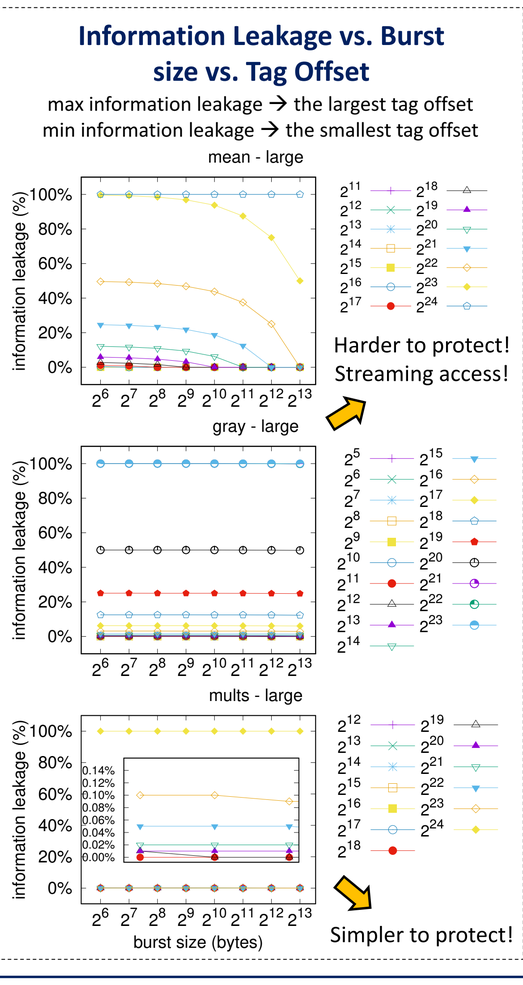
Experimental Results

Performance vs. Burst Size vs. Tag Offset



Information Leakage vs. Burst size vs. Tag Offset

max information leakage → the largest tag offset
min information leakage → the smallest tag offset

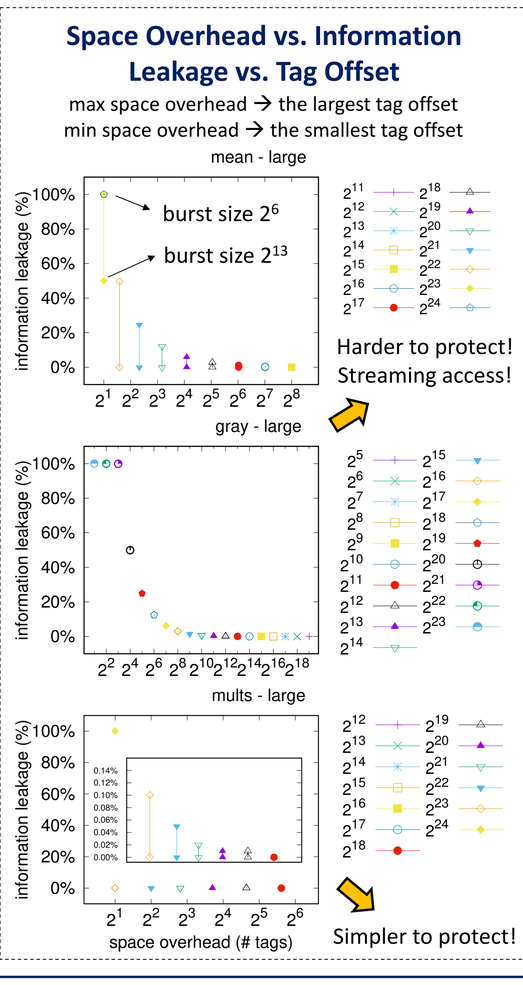


Harder to protect! Streaming access!

Simpler to protect!

Space Overhead vs. Information Leakage vs. Tag Offset

max space overhead → the largest tag offset
min space overhead → the smallest tag offset

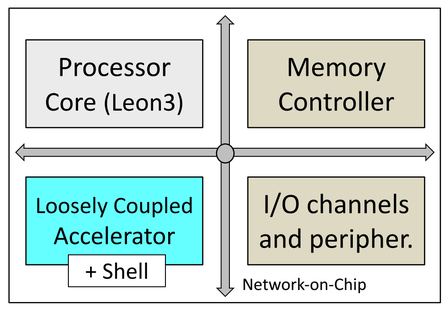


Harder to protect! Streaming access!

Simpler to protect!

Experimental Setup

Embedded Scalable Platforms (ESP)



[P. Mantovani et al., ACM/IEEE DAC '16]
[L. P. Carloni, ACM/IEEE DAC '16]

- The accelerators and the DIFT Shell are designed with SystemC
- We used Cadence Stratus HLS to perform high-level synthesis (HLS)
- We adopted Xilinx Vivado for logic synthesis (*target:* Virtex-7 FPGAs)
- The LEON3 processor boots the Linux OS and calls the hardware accelerators through device drivers
- The experiments on PULPino use a similar system-level methodology

Hardware Accelerators

mean:
calculates the mean over the cols of a 2D matrix

gray:
converts an RGB image into a grayscale image

mults:
multiplies a 2D matrix by its transpose

Hardware Accelerators: Access Patterns

- **mean**: N load bursts to produce 1 store burst
- **gray**: 1 load burst to produce 1 store burst
- **mults**: N load bursts to produce 1 store burst

Take-Home Message

PAGURUS is a flexible methodology to design a shell circuit that extends the DIFT support to hardware accelerators