ALPEN-ADRIA UNIVERSITÄT KLAGENFURT

Towards Micro-Architectural Leakage Simulators: Reverse Engineering Micro-Architectural Leakage Features is Practical

Si Gao[1] Elisabeth Oswald[1] Dan Page[2]

[1]Digital Age Research Center (D!ARC), University of Klagenfurt, Austria
[2]Department of Computer Science, University of Bristol, UK

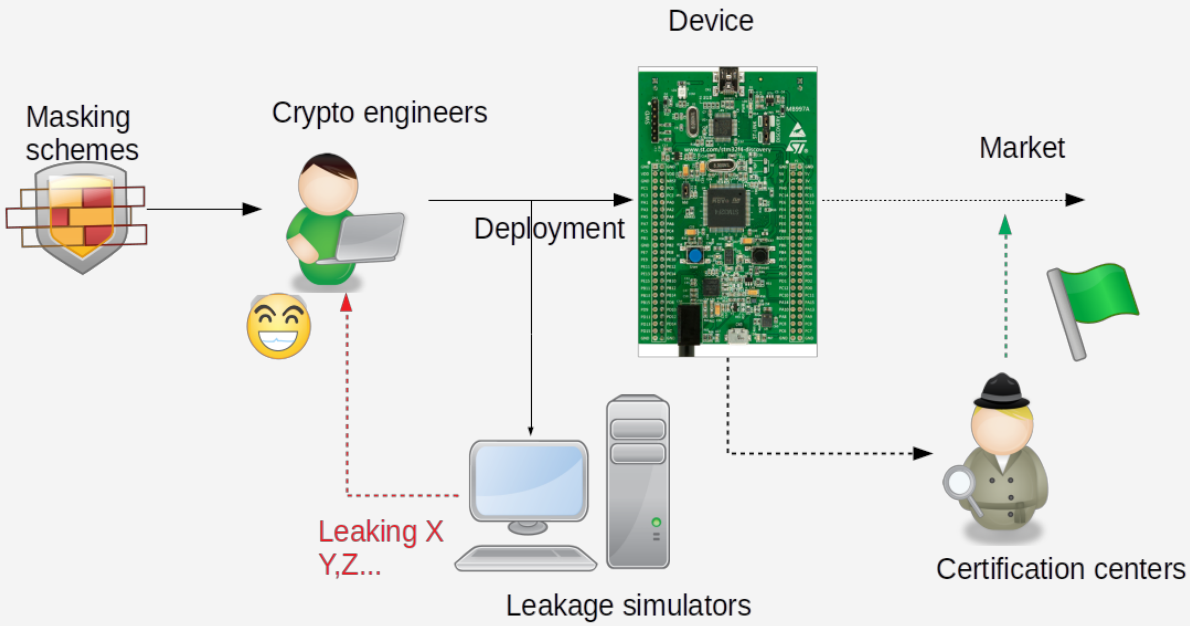
May 27, 2022

Outline

Device assessment

A round map

Develop⇒ Deployment ⇒ product assessment ⇒ “pass” or “fail”

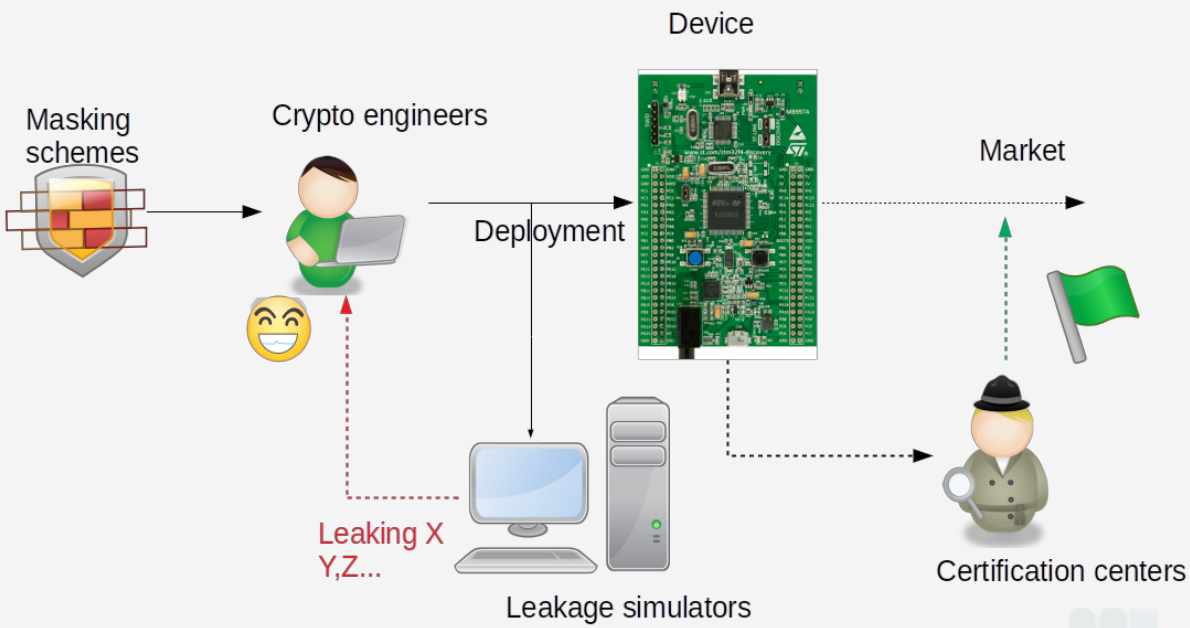


Device assessment

With leakage simulators

Develop⇒ assessment with simulators

- early stage feedback
- with reasoning (i.e. “what causes leakage”)



Leakage simulators: developments

The gray-box route

E.g. the ELMO family (ELMO/ELMO*)

- Target core: Cortex-M0
- Instruction-level: built upon instruction simulator
- Model trained from profiling traces
- ALU leakage from STM32F0, extensions include:
  - Leakage on memory bus
  - NXP LPC1114
  - Cortex-M3

Leakage simulators: developments

The white-box route

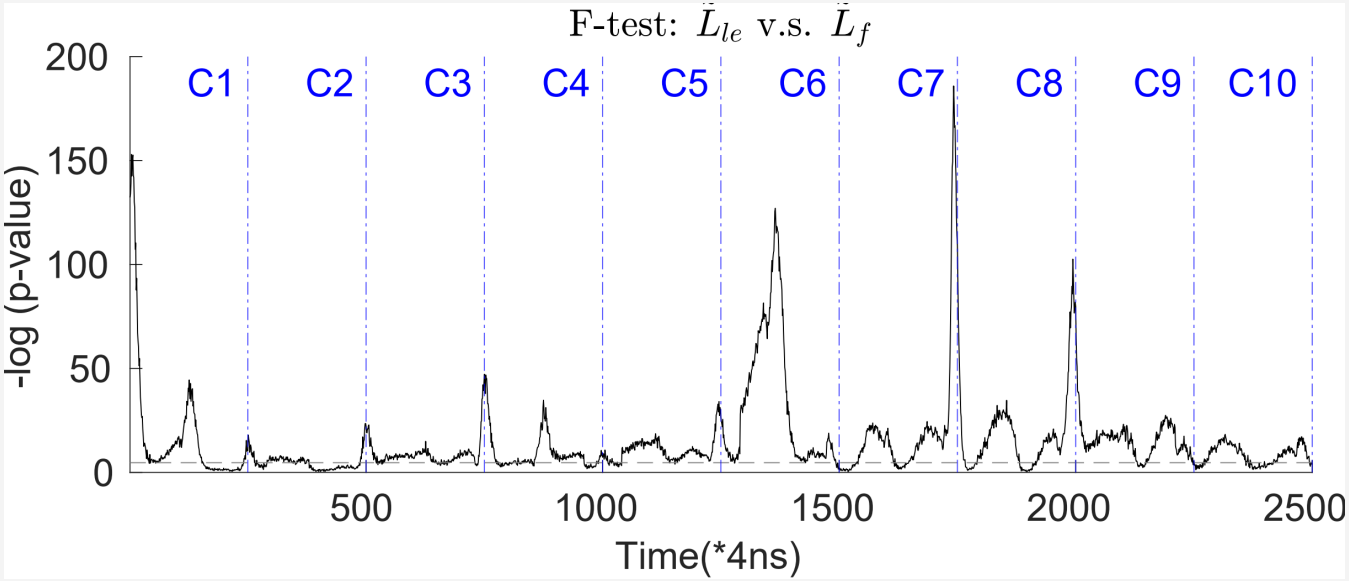
E.g. MAPS [COSADE 18]

- RTL level: ARM Academic version
- μ-arch awareness
- HD model on registers
- no measurement required

Leakage simulators: developments

Model "Completentess" (Recap from last talk)

Both are far from ideal...



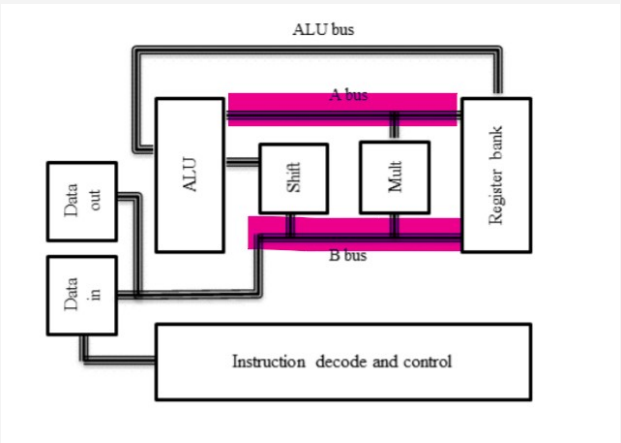
Leakage simulators: developments

Drawbacks: "relatively simple"

Drawbacks of ELMO/ELMO*

- only focus on the ALU (versus a 3-stage pipeline core)
- model built from the ALU buses (two magenta lines)
  - 2 buses lie in μ-arch
  - ELMO's model represents authors' guess

E.g. "adds r0, r1" $\Rightarrow$ "which goes to bus A?"

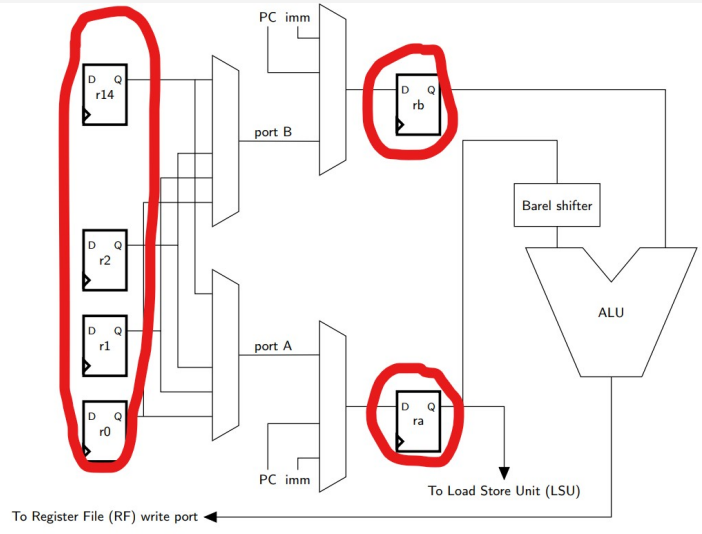


Leakage simulators: developments

Drawbacks: "relatively simple"

Drawbacks of MAPS

- Same as product? Manufacturer's version? [CHES 21a]
- "The simulator traces only the registers" [COSADE 18]
  - no leakage from the ALU

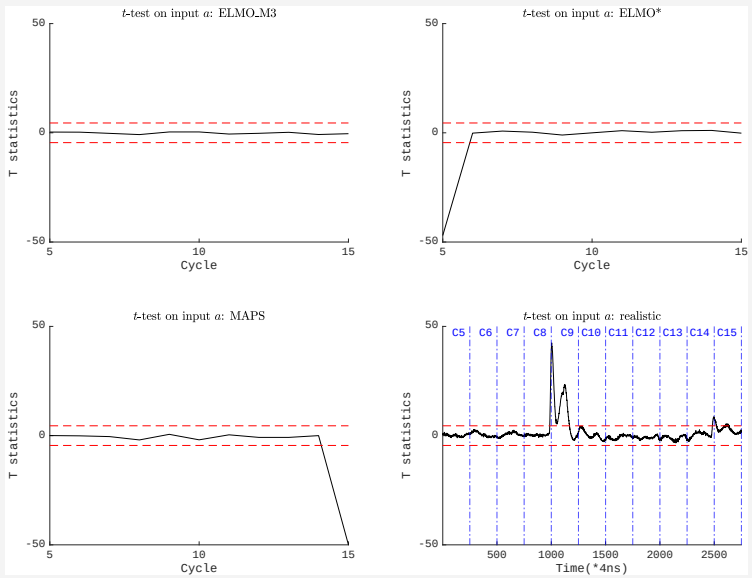


Leakage simulators: an example

(Another) 2-share bit-wise ISW multiplication

ELMO/ELMO*/MAPS/Realistic t-test

- all written in Thumb assembly
- Realistic: ARM Cortex-M3 (NXP LPC 1114)
- ELMO: updated to the M3 model

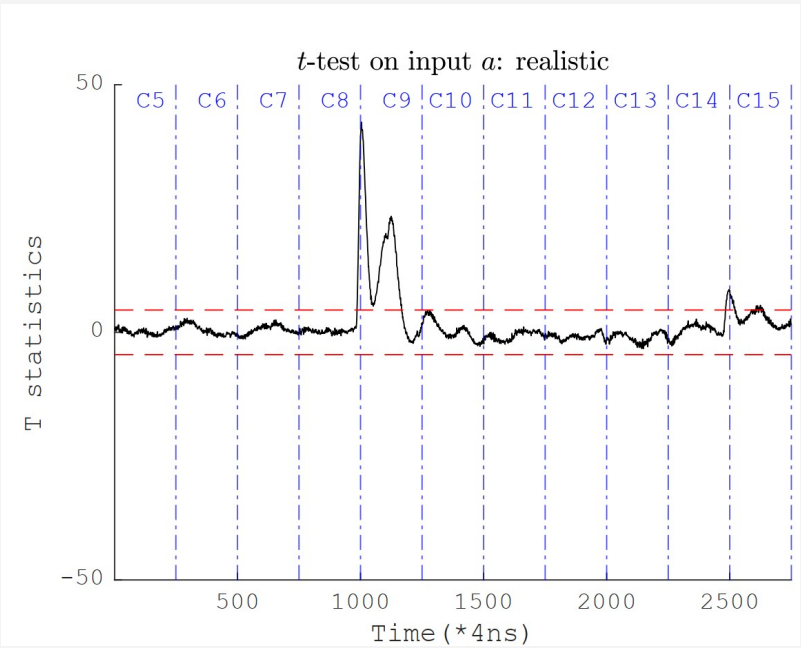


Leakage simulators: an example

2-share bit-wise ISW multiplication

Realistic t-test

- C9 & C15

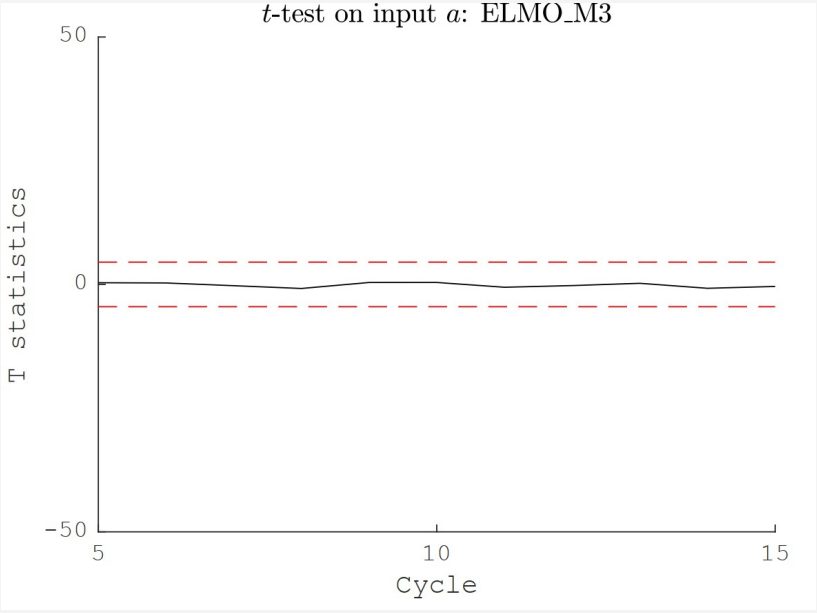


Leakage simulators: an example

2-share bit-wise ISW multiplication

ELMO

- both missed

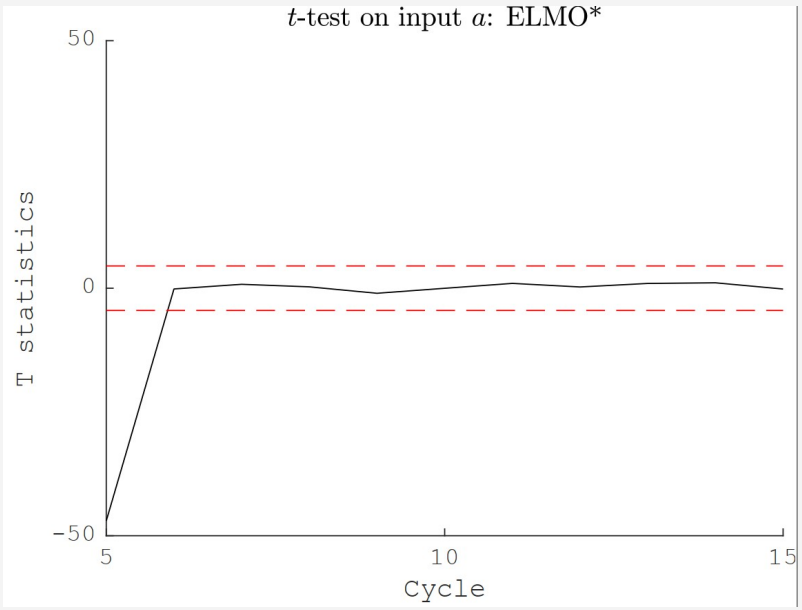


Leakage simulators: an example

2-share bit-wise ISW multiplication

ELMO*

- both missed
- produce another false-positive

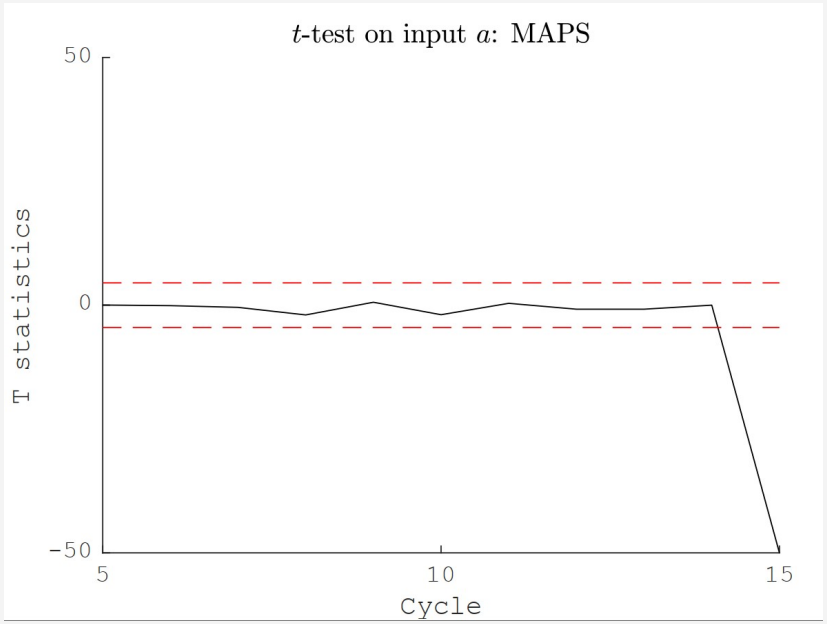


Leakage simulators: an example

2-share bit-wise ISW multiplication

MAPS

- finds C15, misses C9



Leakage simulators: an example

Missing leaks...

Why?

- overly simplified model
- μ-arch effects missing

Motivation for reverse engineering the μ-arch...

- "leakage-wise reverse engineering"
- μ-arch enhanced leakage simulator

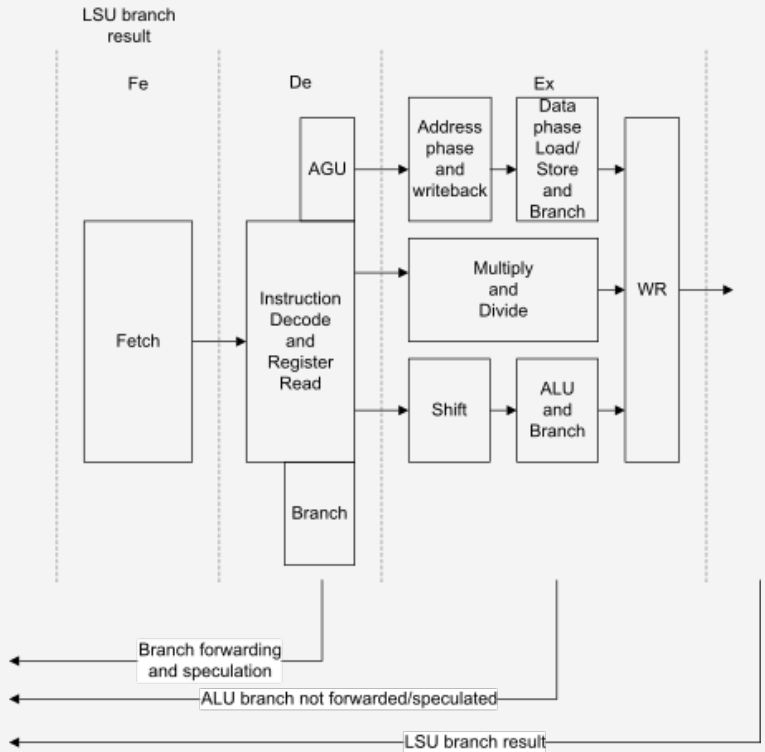
Outline

Starting point

Public info from ARM

ARM Cortex-M3 (NXP 1313)

- 3-stage pipeline core
  - **Fe**: Fetch
  - **De**: Decode
  - **Ex**: Execute
- operands loaded in **De**
  - at least 2 reading ports
  - at least 2 pipeline registers



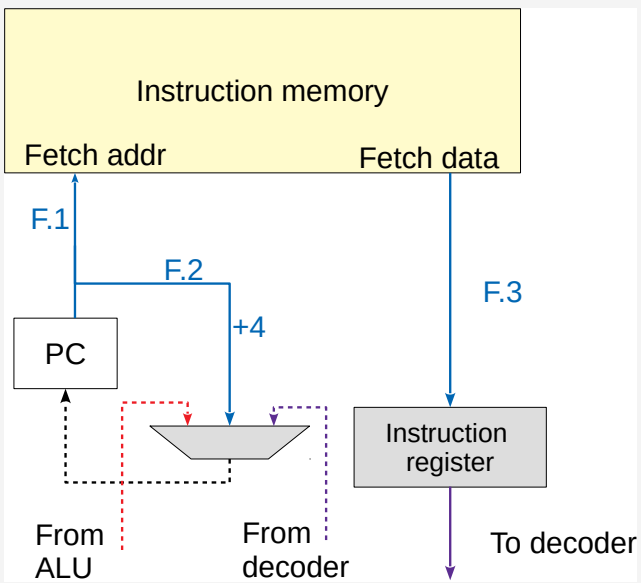
ARM DDI 0337E

Recovering micro-architectural elements: **Fe**

Fetch

Fetching instructions

- PC provided address (F.1)
- instruction to instruction registers (F.3)
- no ambiguity not data-dependent

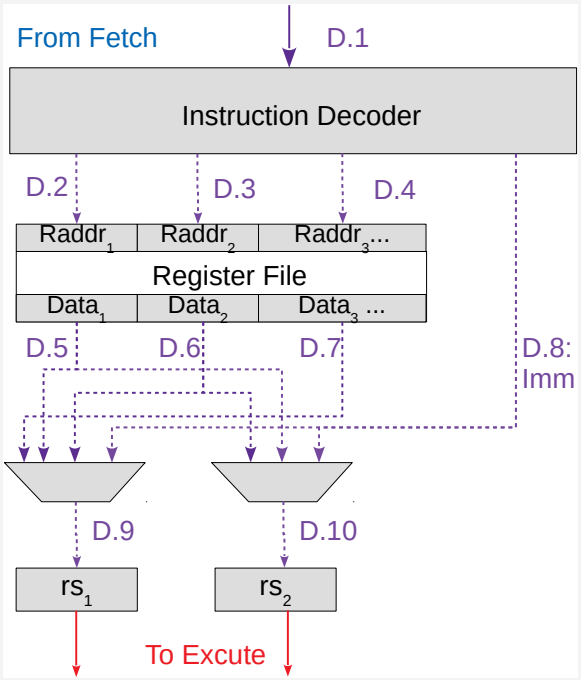


Recovering micro-architectural elements: **De**

Decode

Decode the instruction in IR

- which operand enters which reading port?



Recovering micro-architectural elements: **De**

Decode

Testing reading ports

- A "interact" with C $\Rightarrow$ A and C share the same reading port

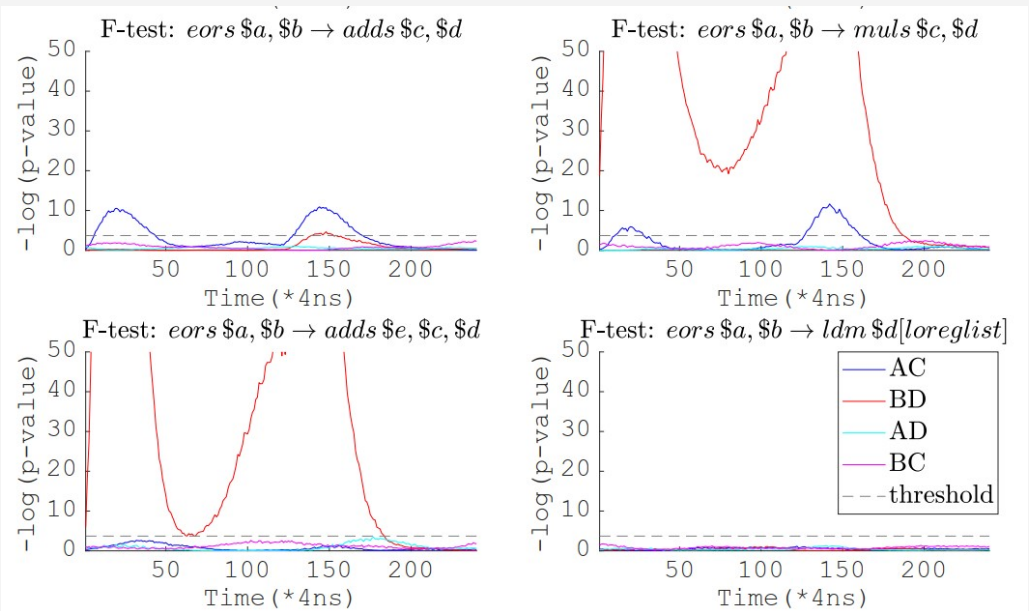
```
1 Testing_port:
2 ...
3 eors $a,$b
4 INSTR $c, $d
5 nop
6 eors $0,$0 //$0=register that stores 0
7 ...
```

Recovering micro-architectural elements: **De**

Decode

$-log(pv) > th \Rightarrow$ interaction exists

- $adds$ $\$c, \$d$: AC and BD
- $adds$ $\$c, \#0$: only AC

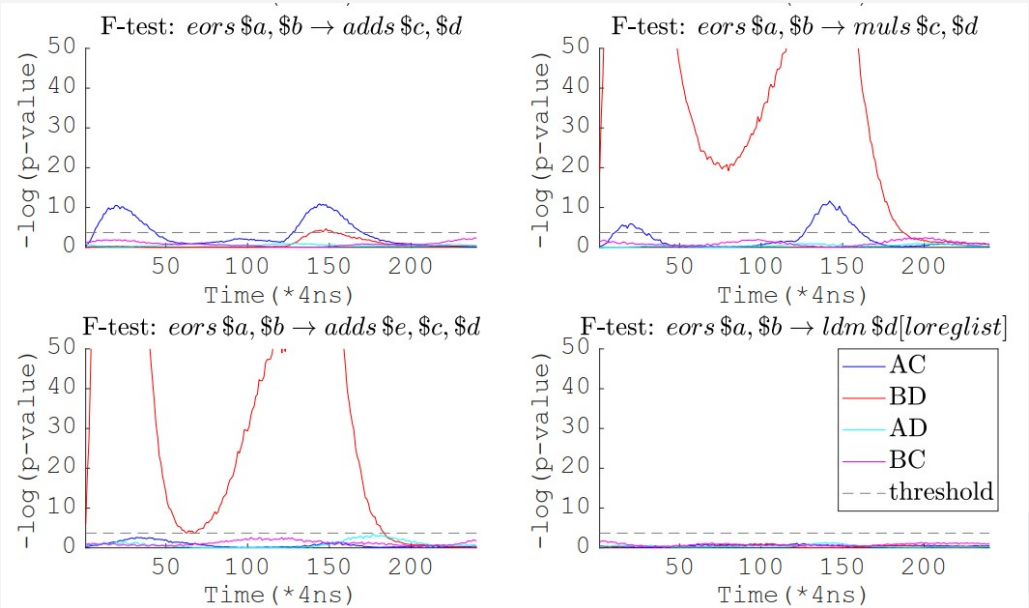


Recovering micro-architectural elements: **De**

Decode

Most instructions loads the first operand accordingly,

- $adds \quad \$c, \$d, \$e$: all 3 loaded, yet only BD visible
  - A to E, C to the third port
  - could be wrong, due to glitch
- $ldm \quad \$d, [loreglist]$: nothing is loaded

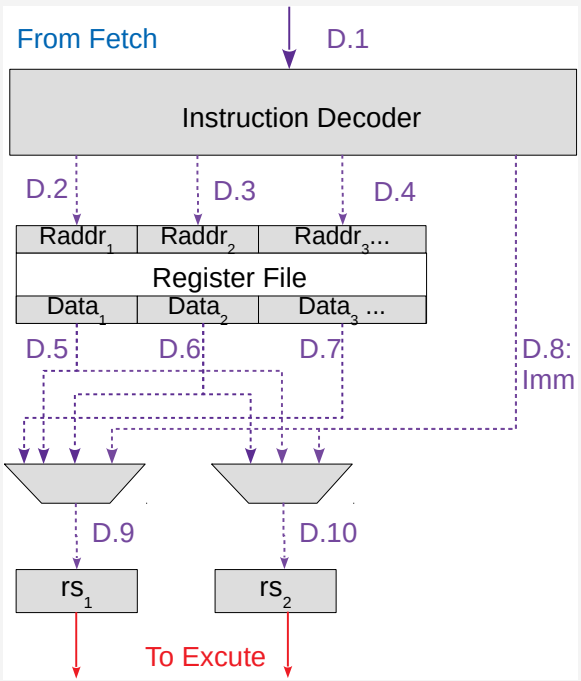


Recovering micro-architectural elements: **De**

Decode to Execute

2 pipeline registers rs_1 and rs_2

- *which operand enters rs_1?*
- *will rs_1 and rs_2 be updated?*



Recovering micro-architectural elements: **Ex**

Execute

Skip further details...

Table 1: Summary of tested Thumb-16 instructions.

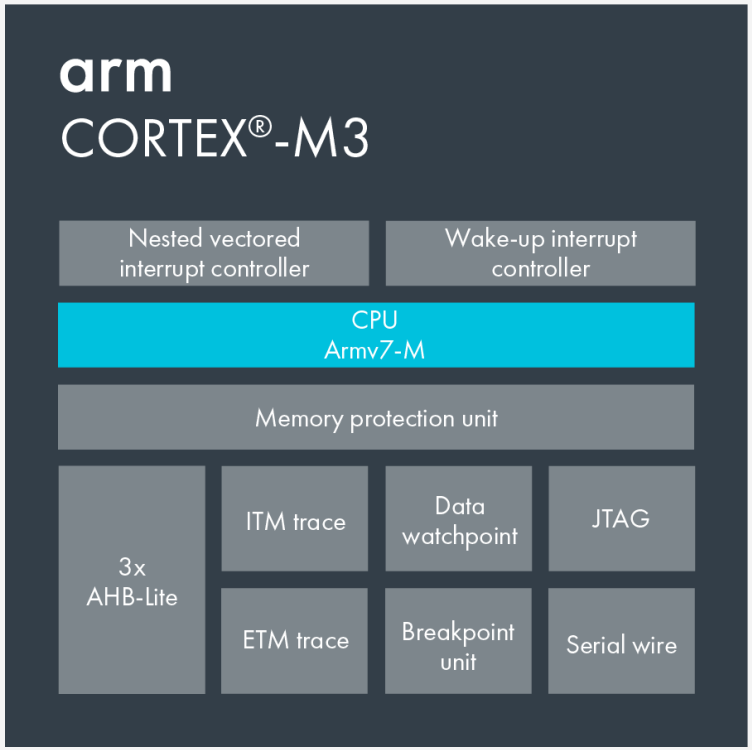
Group		Assembler	Encoding				Decoding			Executing	
	Operand		Type	Rd	Rm	Rn	Port 1	Port 2	Port 3	RS_1	RS_2
ALU	0	MOVS Rd, #<imm8>	I	10-8	-	-	Rd	-	-	-	-
	1	INSTR_a Rd, Rm(, #<imm3/5>)	II	2-0	5-3	-	Rd	Rm	-	Rm	-
		INSTR_b Rd, Rm(, #<imm3/5>)	II	2-0	5-3	-	Rd	Rm	-	-	Rm
		INSTR Rd, Rd, #<imm8>	I	10-8	-	-	Rd	-	-	Rd	-
		INSTR Rd, Rm	III	7,2-0	6-3	-	Rd	Rm	-	-	Rm
	2	INSTR Rd, Rm	II	2-0	5-3	-	Rd	Rm	-	Rd	Rm
		INSTR Rd, Rn, Rm	IV	2-0	8-6	5-3	Rd	Rm	Rn	Rn	Rm
		ADD Rdn, Rm	III	7,2-0	6-3	-	Rdn	Rm	-	Rdn	Rm
		MUL Rdm, Rn	IV	2-0	-	5-3	Rdm	Rn	-	Rdm	Rn
LOAD	Imm	LDR(H/B) Rd, [Rn, #<imm>]	IV	2-0	-	5-3	Rd	Rn	-	Rn	-
	Reg	LDR(H/B) Rd, [Rn, Rm]	IV	2-0	8-6	5-3	Rd	Rn	Rm	Rn	Rm
	Multiple	LDM Rn!, <loreglist>	V	-	-	10-8	-	-	Rn	Rn	-
	Pop	POP <loreglist>		-	-	-	-	-	-	C	-
STORE	Imm	STR(H/B) Rd, [Rn, #<imm>]	IV	2-0	-	5-3	Rd	Rn	-	Rn	Rd
	Reg	STR Rd, [Rn, Rm]	IV	2-0	8-6	5-3	Rd	Rn	Rm	Rn->Rd	Rm
	Multiple	STM Rn!, <loreglist>	V	-	-	10-8	-	-	Rn	Rn	-
	Push	PUSH <loreglist>		-	-	-	-	-	-	C	-

Recovering micro-architectural elements: Memory

Memory

Often ignored by existing tools:

- further away from the core
- "asynchronous": self-timed
- but leaks heavily...



Recovering micro-architectural elements: Memory

Memory

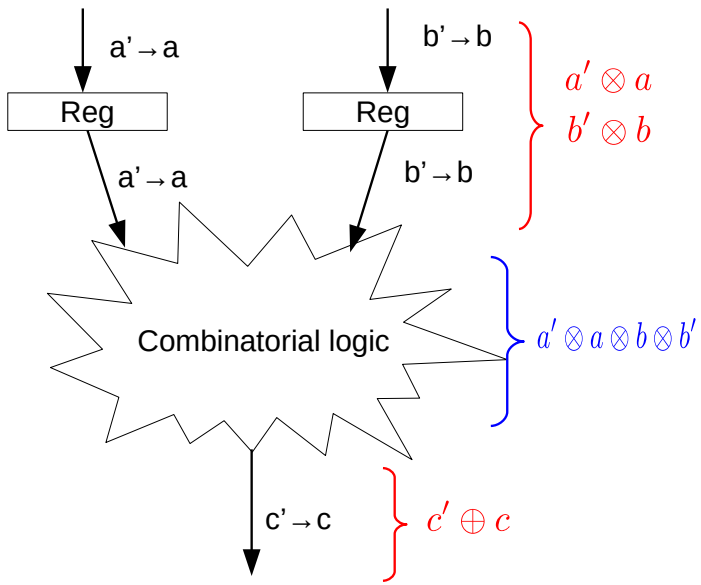
Cannot "synchronise" $\Rightarrow$ No Completeness test

- Existing knowledge
  - Word-wise memory access
- ARM's Specification
  - Shared read/write data bus
  - Shared address bus
  - Write buffer
- Not ideal, similar to existing tools

Outline

leakage modelling: general idea

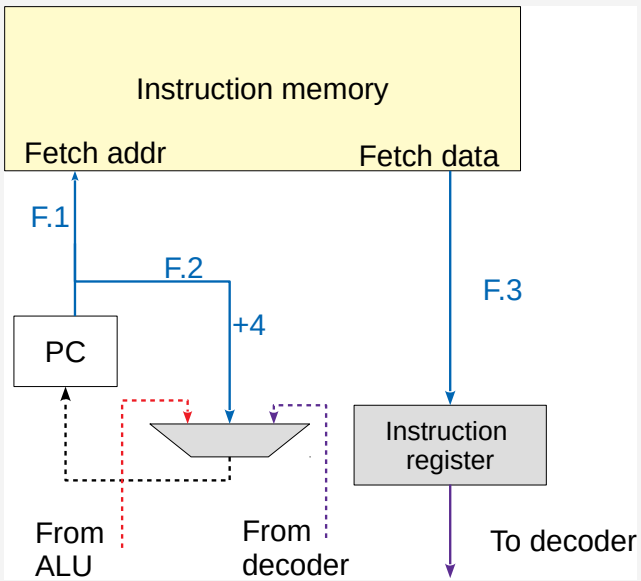
- Buses + registers
  - often assumed to be HW/HD
  - $\otimes$: jointly leaking (HD included)
  - "previous value $\otimes$ current value"
- Combinatorial logic
  - affected by glitches
  - conservative modelling
  - "previous inputs $\otimes$ current inputs"



μ-arch leakage modelling

Fetch

Ignore Fetch: not data-dependent



μ-arch leakage modelling

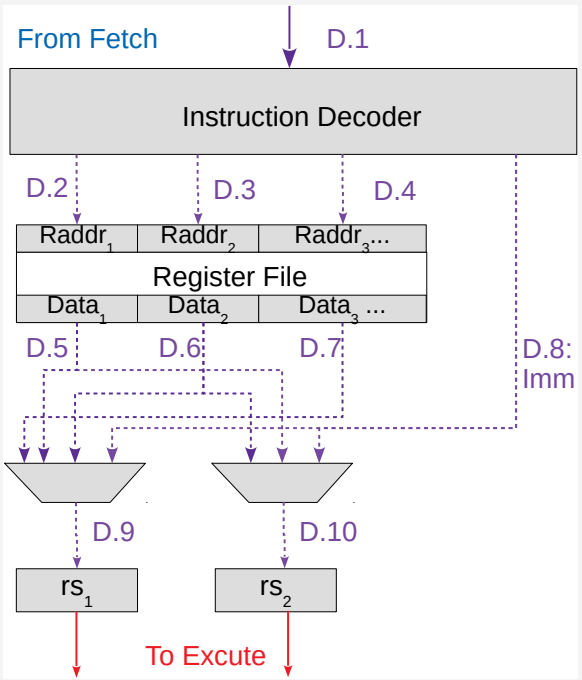
Decode

Ignore all wires before the register file

- D.5-7 (prime stands for the previous value)

$$L_D = \sum_i port_i \otimes port'_i$$

- D.8 not data-dependent
- D.9-10 considered later



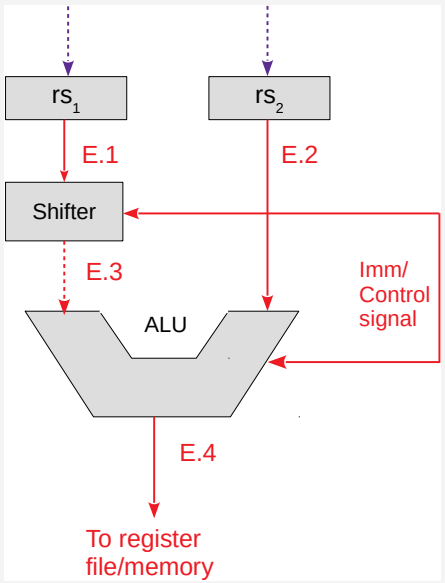
μ-arch leakage modelling

Execute

"not gated" $\Rightarrow$ anything could happen

- E.1-4 plus ALU

$$L_E = rs'_1 \otimes rs'_2 \otimes rs_1 \otimes rs_2$$

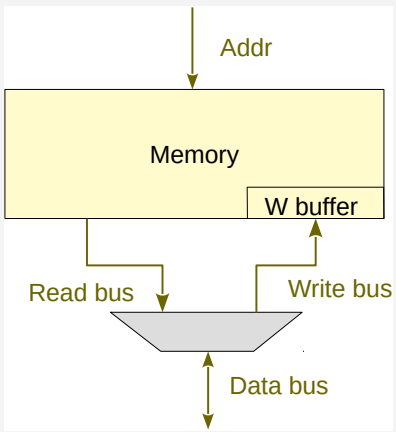


μ-arch leakage modelling

Memory& Overall

Memory:

$$L_M = \{Bus \otimes Bus', Bus_w \otimes Bus'_w, Addr \otimes Addr'\}$$



Overall:

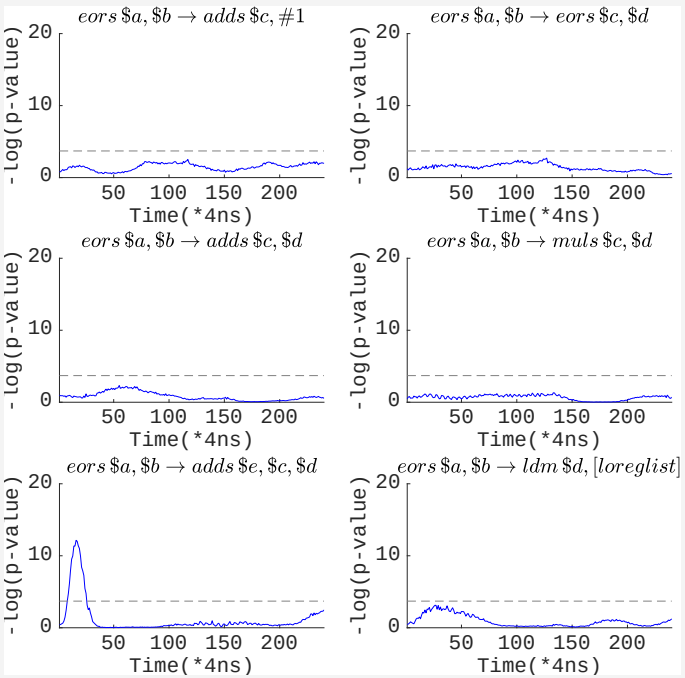
$$L = L_D + L_E + L_M$$

μ-arch leakage modelling

Put it together& testing "quality"

Evaluate the "quality" of our model[EUROCRYPT 22],

- higher than threshold $\Rightarrow$ model not complete
- some instructions require more ("glitchy register access")



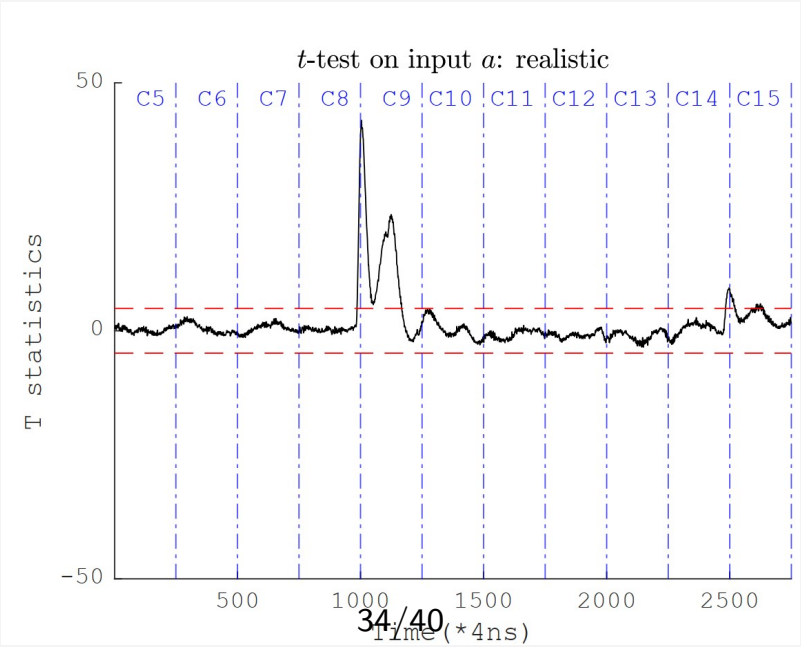
Outline

Impacts

Back to our example in the begining...

Reverse engineered μ-arch info helps to explain the leaks...

- C9: ALU output HD
  - Not in ELMO or MAPS
- C15: rs_1 HD
  - MAPS got it



Outline

Achievements & Future works

Achievements

We have successfully...

- (leakage-wise) reverse engineered the μ-arch of an M3 core
- built a μ-arch enhanced leakage model
- shown its impacts on various masking implementations

Achievements & Future works

Future works

- "cycle-accurate" memory emulator
- exploit more subtle μ-arch leaks
- higher-order testing
- flexible framework for other architectures (e.g. RISC-V)
- formal verification

The End

Questions?

Reference

OSADE 18 Corre, Y.L., Großschädl, J., Dinu, D.: Micro-architectural Power Simulator for Leakage Assessment of Cryptographic Software on ARM Cortex-M3 Processors. In Fan, J., Gierlichs, B., eds.: Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings. Volume 10815 of Lecture Notes in Computer Science., Springer (2018) 82–98

CHES 21a Marshall, B., Page, D., & Webb, J. (2021). MIRACLE: MIcRo-ArChitectural Leakage Evaluation: A study of micro-architectural power leakage across many devices. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022(1), 175–220.

Reference

CHES 21b Barthe, G., Gourjon, M., Gr´egoire, B., Orlt, M., Paglialonga, C., Porth, L.: Masking in Fine-Grained Leakage Models: Construction, Implementation and Verification. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2021(2) (2021) 189–228

CRYPT 22 Gao, S., Oswald, E.: A Novel Completeness Test and its Application to Side Channel Attacks and Simulators