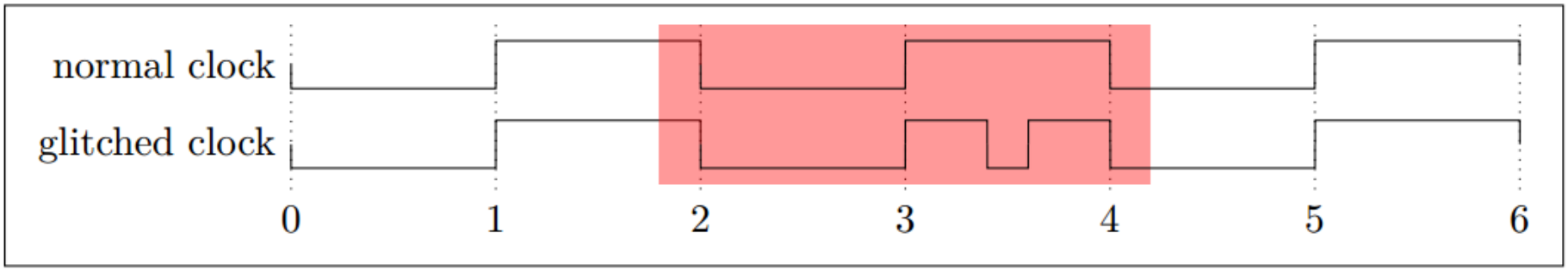
PRACTICAL IMPLEMENTATION OF HIGHER-ORDER INSTRUCTION SKIPS IN MICROCONTROLLERS

Ricardo Gomes da Silva

20. Kryptotag, 2014

Introduction

- Clock signal
  - Constant frequency square wave signal
  - Digital circuits
  - Synchronization
- Glitch
  - Small fault in a system
  - A glitch in the clock signal
    - Unexpected behavior in the signal



Clock glitching

- Glitching on purpose
  - Causes "unexpected" behavior
    - Data corruption
    - Process old data
    - Instruction skips
  - Can target (multiple) specific instructions

- Security
  - Change the original execution path
  - Bypass security measures

Clock glitching

```
1  var i := 1
2  var sum := 0

3  while i <= 5 do
4     sum := sum + i
5     i := i + 1
6  end while
```

Clock glitching

```
1  var i := 1
2  var sum := 0

3  while i <= 5 do
4     sum := sum + i
5     i := i + 1
6  end while
```

`sum := sum + i`	`sum := sum + i`	`sum := sum + i`	`sum := sum + i`	`sum := sum + i`
`i := i + 1`	`i := i + 1`	`i := i + 1`	`i := i + 1`	`i := i + 1`
i = 1 **sum = 1**	**i = 2** **sum = 3**	**i = 3** **sum = 6**	**i = 4** **sum = 10**	**i = 5** **sum = 15**

Clock glitching effects (1)

```
sum := sum + i

i := i + 1
```

i = 1
sum = 1

```
sum := sum + i

i := i + 1
```

i = 2
sum = 3

```
sum := sum + i

i := i + 1
```

i = 3
sum = 6

```
sum := sum + i

i := i + 1
```

i = 4
sum = 10

```
sum := sum + i

i := i + 1
```

i = 5
sum = 15

Clock glitching effects (1)

`sum := sum + i`	`sum := sum + i`	`sum := sum + i`	`sum := sum + i`	~~`sum := sum + i`~~
`i := i + 1`	`i := i + 1`	`i := i + 1`	`i := i + 1`	`i := i + 1`
`i = 1` **`sum = 1`**	**`i = 2`** **`sum = 3`**	**`i = 3`** **`sum = 6`**	**`i = 4`** **`sum = 10`**	**`i = 5`** **`sum = 10`**

Clock glitching effects (2)

```
sum := sum + i

i := i + 1
```

i = 1
sum = 1

```
sum := sum + i

i := i + 1
```

i = 2
sum = 3

```
sum := sum + i

i := i + 1
```

i = 3
sum = 6

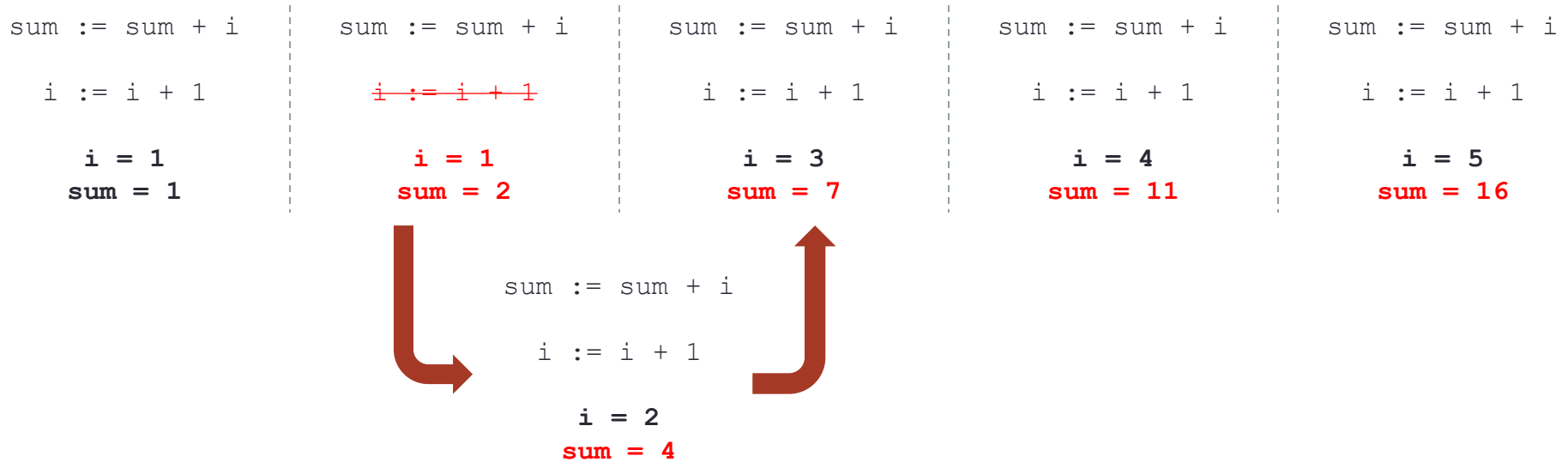
```
sum := sum + i

i := i + 1
```

i = 4
sum = 10

```
sum := sum + i

i := i + 1
```

i = 5
sum = 15

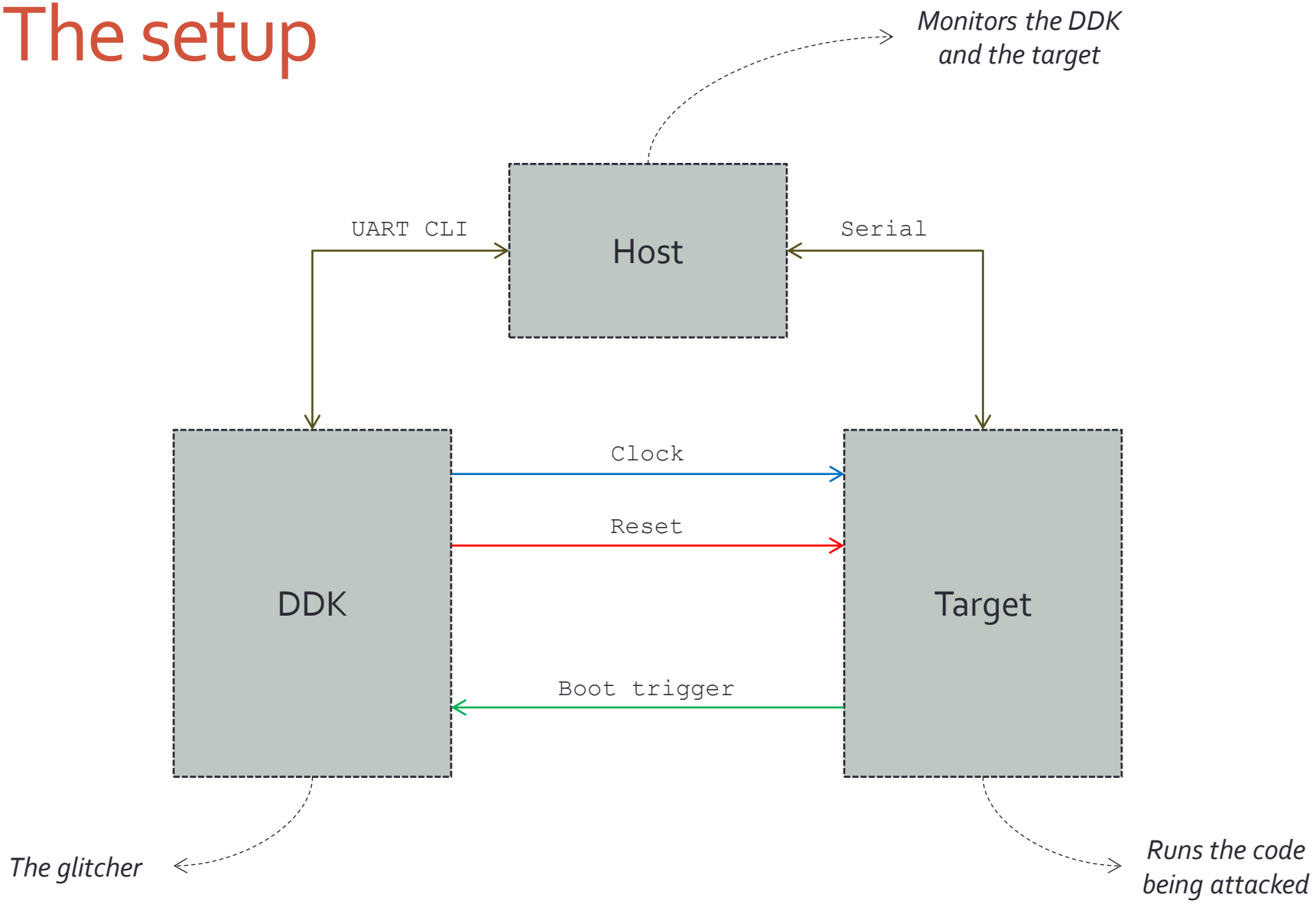
Clock glitching effects (2)



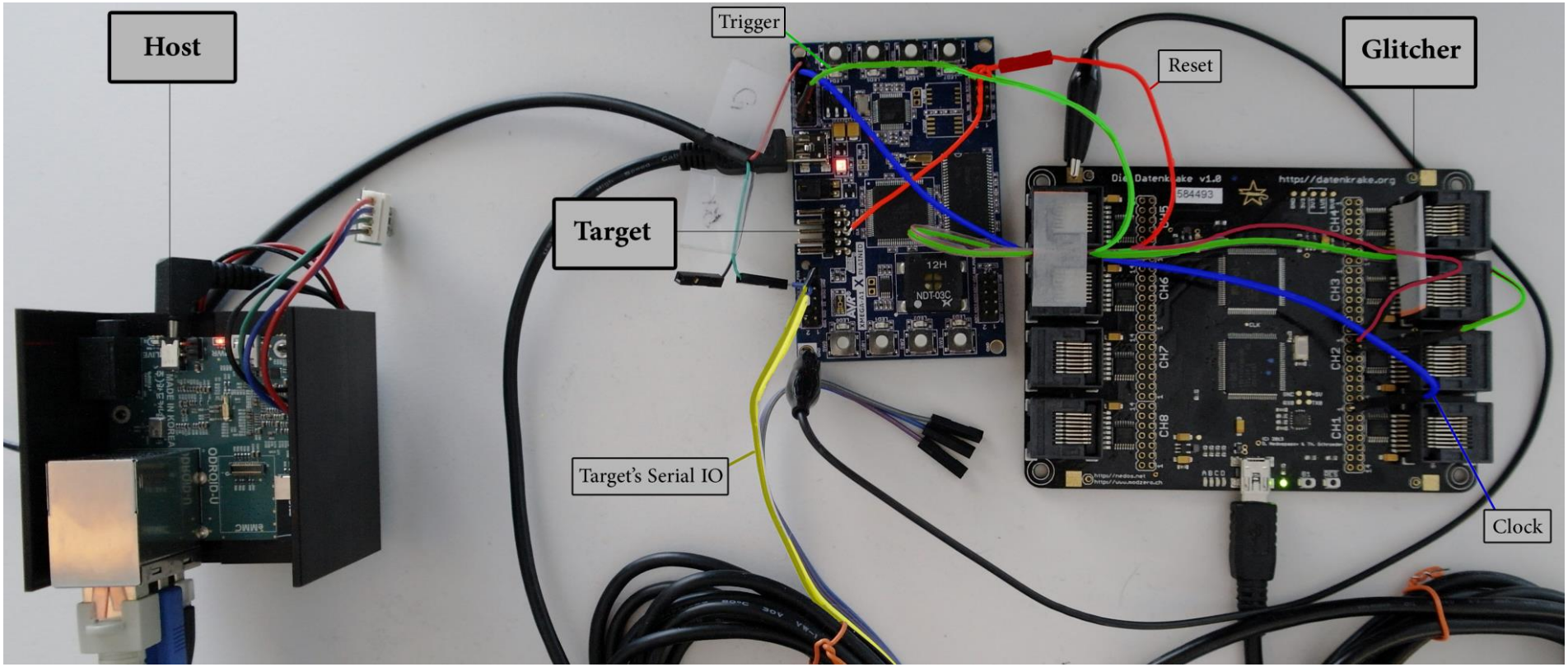
The glitcher

- A glitch is defined by
  - Delay
  - Width
  - Mode
- Counters are in clock cycles
  - Synchronization
- Glitches in sequence
  - We want multiple glitches within the same execution
  - List of glitches: FIFO queue

The setup


Monitors the DDK and the target
Host
UART CLI
Serial
DDK
Target
Clock
Reset
Boot trigger
The glitcher
Runs the code being attacked

The setup


Host
Trigger
Target
Reset
Glitcher
Target's Serial IO
Clock

Attacking a real target

```
/* ... */
call fb4_mul_dxs
.LVL43:
/* decrement loop counter LSB, MSB */
subi r16,1
sbc r17,__zero_reg__
.loc 1 247 0 discriminator 2
/* jump out of the loop */
breq .+2
/* jump loop begin */
rjmp .L2
.LEB2:
.loc 1 486 0
/* clean stack */
subi r28,36
sbci r29,-2
out __SP_L__,r28
out __SP_H__,r29
pop r29
/* ... */
/* exponentiation call */
call etat_exp
```

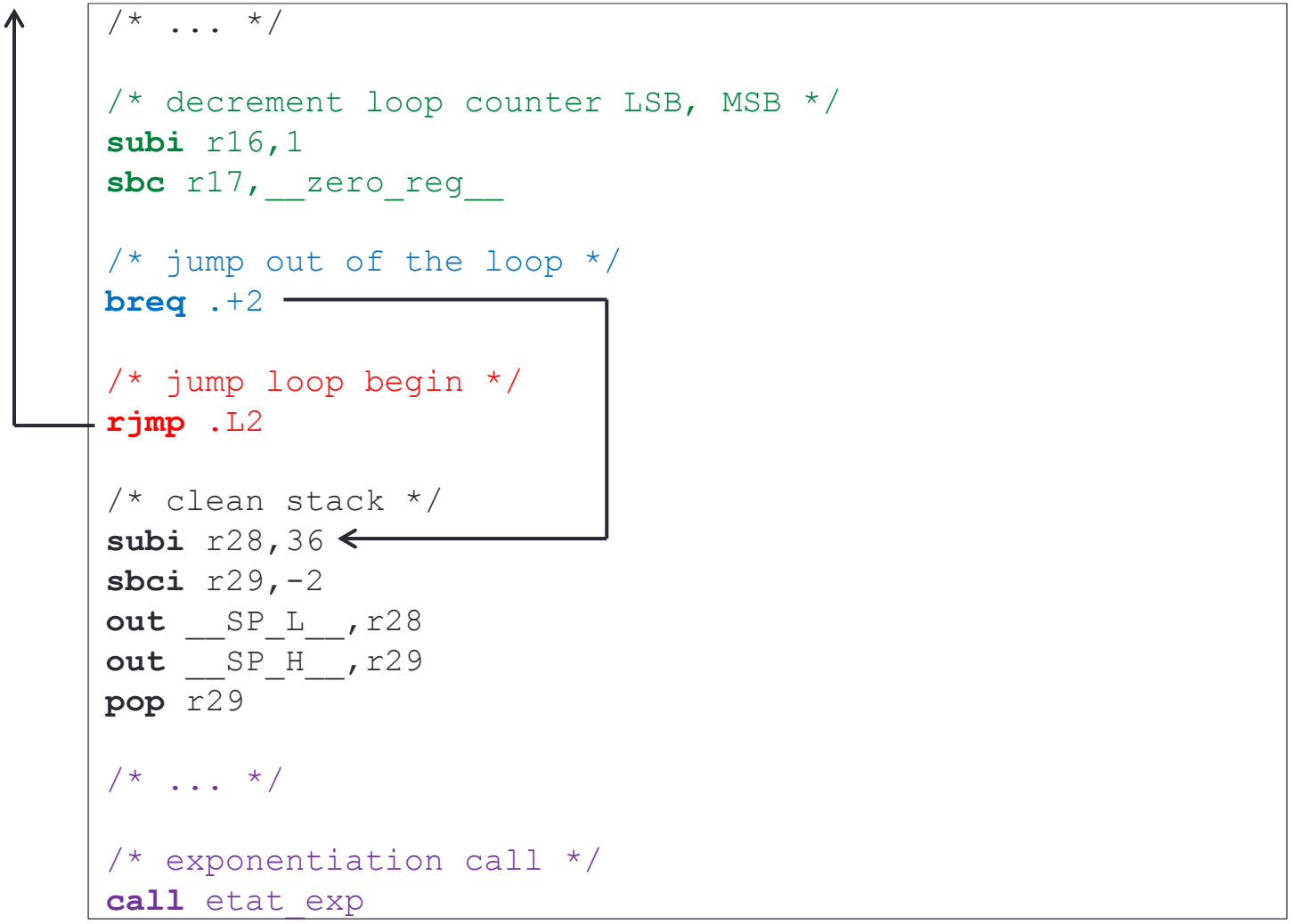
Attacking a real target

```
/* ... */

/* decrement loop counter LSB, MSB */
subi r16,1
sbc r17,__zero_reg__

/* jump out of the loop */
breq .+2

/* jump loop begin */
rjmp .L2

/* clean stack */
subi r28,36
sbci r29,-2
out __SP_L__,r28
out __SP_H__,r29
pop r29

/* ... */

/* exponentiation call */
call etat_exp
```

Attacking a real target

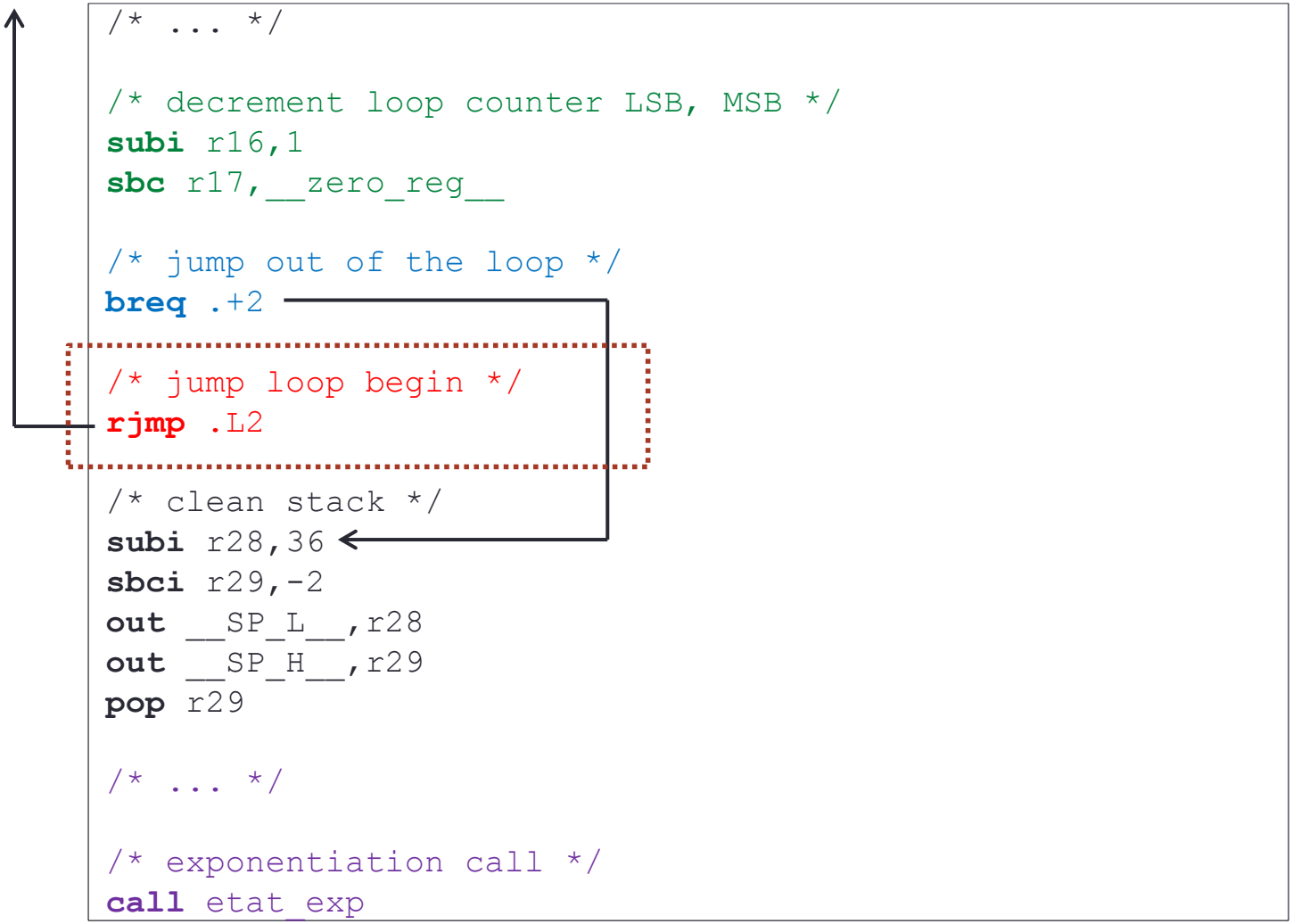
```
/* ... */

/* decrement loop counter LSB, MSB */
subi r16,1
sbc r17,__zero_reg__

/* jump out of the loop */
breq .+2

/* jump loop begin */
rjmp .L2

/* clean stack */
subi r28,36
sbci r29,-2
out __SP_L__,r28
out __SP_H__,r29
pop r29

/* ... */

/* exponentiation call */
call etat_exp
```

Attacking a real target

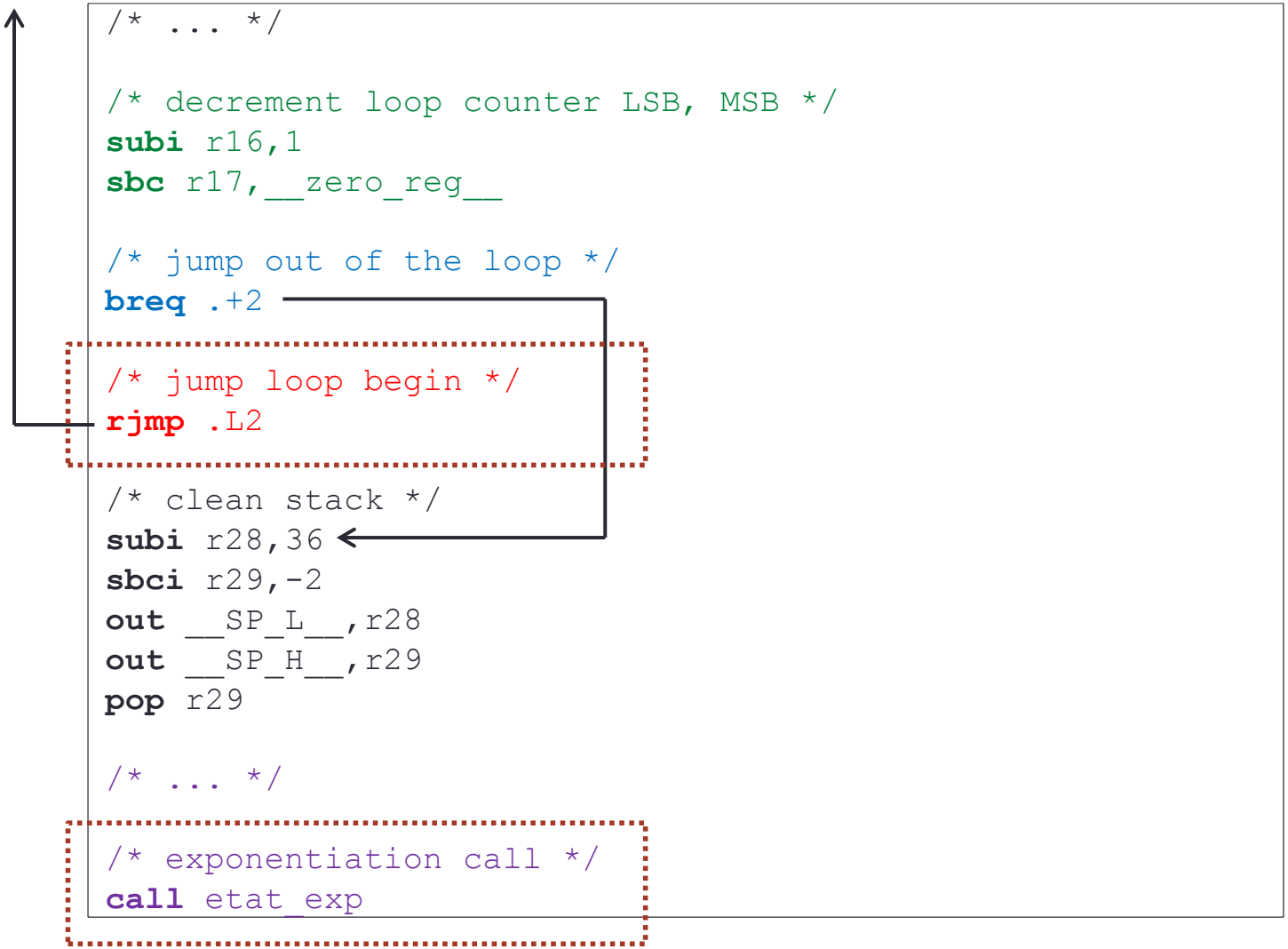


```
/* ... */

/* decrement loop counter LSB, MSB */
subi r16,1
sbc r17,__zero_reg__

/* jump out of the loop */
breq .+2

/* jump loop begin */
rjmp .L2

/* clean stack */
subi r28,36
sbci r29,-2
out __SP_L__,r28
out __SP_H__,r29
pop r29

/* ... */

/* exponentiation call */
call etat_exp
```

Attacking a real target



Attacking a real target



Attacking a real target

- Which instruction should we glitch?
  - How to find it?
  - How to synchronize with it?
  - Knowledge of the code is useful, but not required
- Timing matters
  - Wrong timing lead to wrong instructions
  - Loop iterations can take a lot of time
    - Delays >= 500.000 clock cycles
- Brute-force for finding the *perfect* glitch
  - Try and repeat within a range

Conclusion

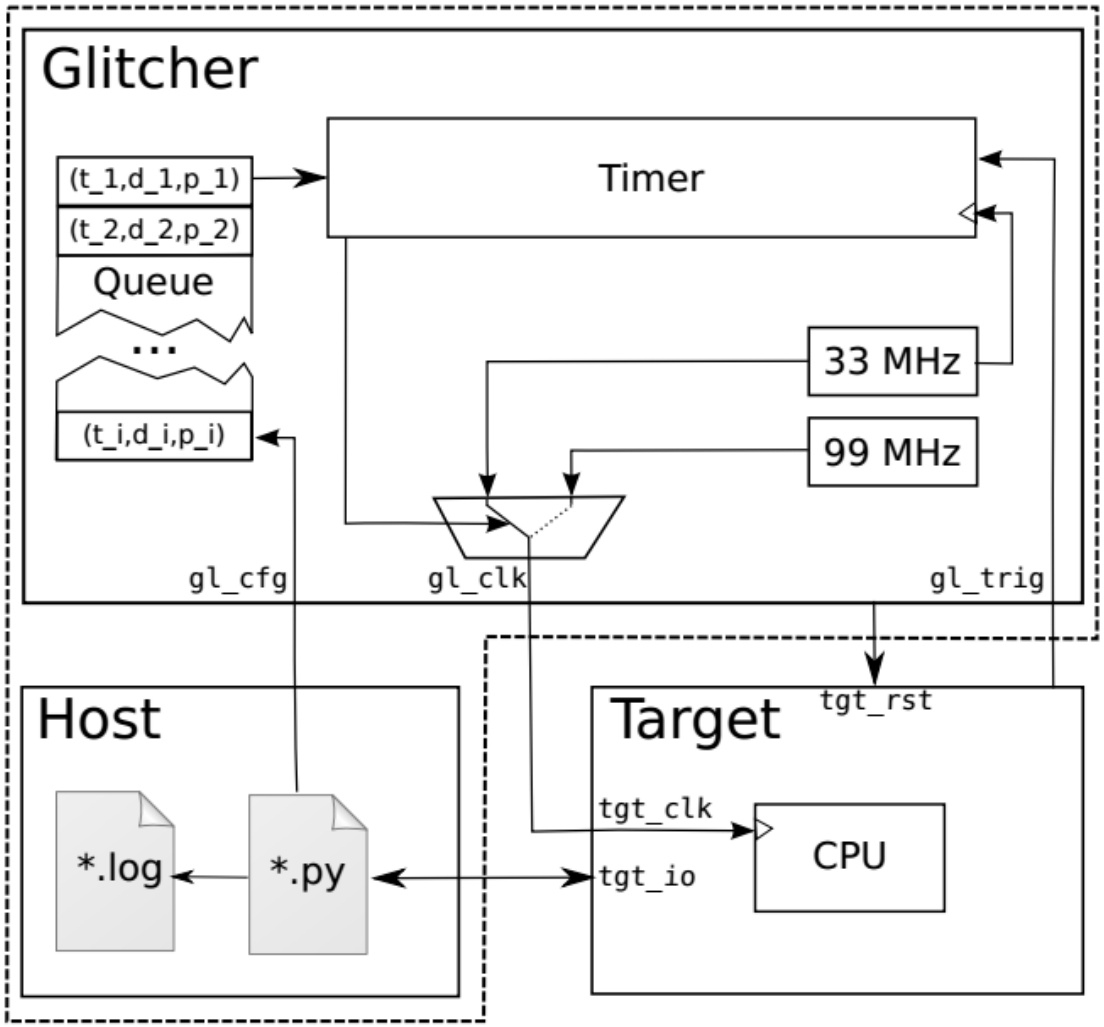
- A clock glitching platform was fully developed
  - Open-source hardware and software platforms
  - Support to multiple glitches within the same execution
    - Stability issues might arise
  - Automatic attacking, monitoring and self-testing
    - Log of attacks and basic analysis of success/failure
- We successfully attacked a microcontroller
  - Attacks are repeatable and stable
  - Multiple attacks are possible
    - Exiting loops, dumping part of the memory, skipping function calls

PRACTICAL IMPLEMENTATION OF HIGHER-ORDER INSTRUCTION SKIPS IN MICROCONTROLLERS

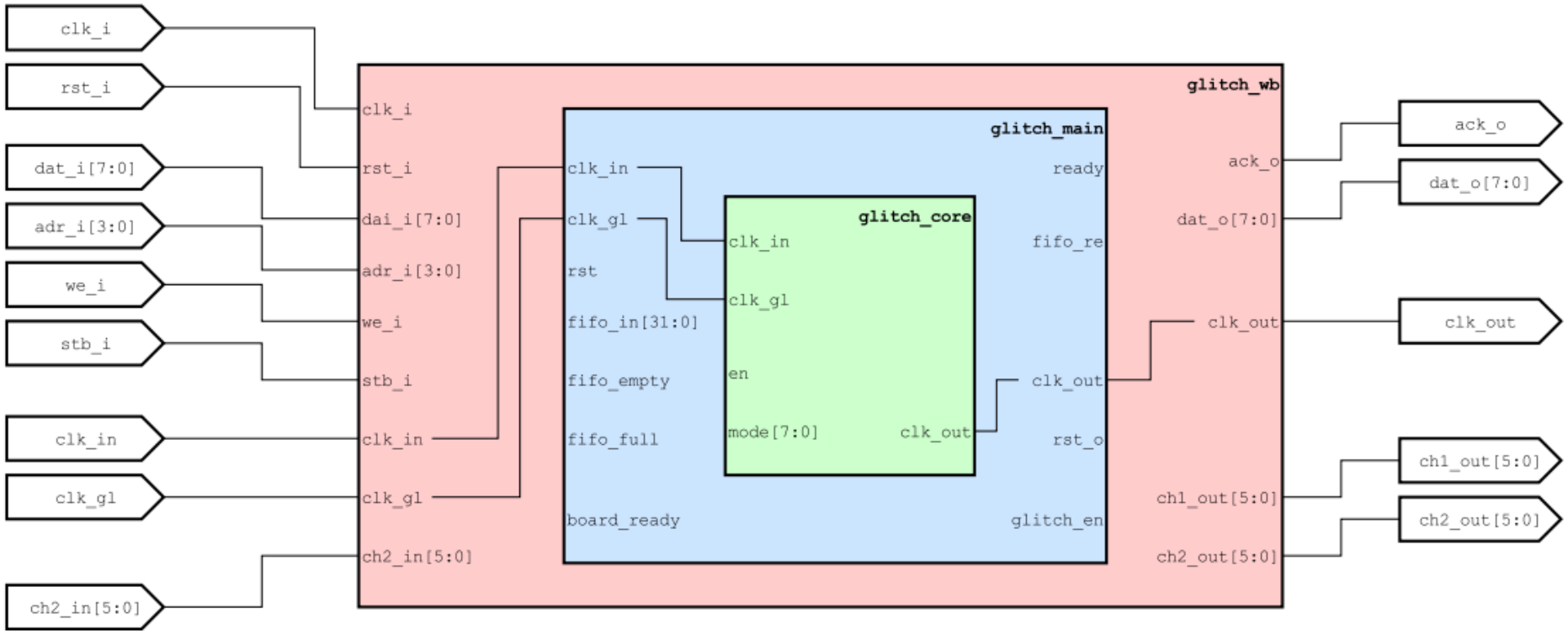
Ricardo Gomes da Silva

20. Kryptotag, 2014

Glitcher setup



Glitcher internal design



Glitcher timing