# An In-depth and Black-box characterization of the effects of Clock Glitches on 8-bit MCUs

**Josep Balasch**, Benedikt Gierlichs, and Ingrid Verbauwhede

Katholieke Universiteit Leuven ESAT / COSIC

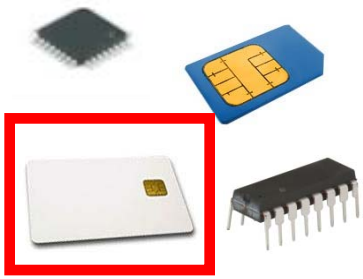
Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC 2011

Nara, Japan, 28 September 2011

KATHOLIEKE UNIVERSITEIT LEUVEN

# Motivation (I)

**FAULT ANALYSIS**

| TARGET DEVICE | FAULT INDUCTION | FAULT MODELS | FAULT EXPLOTATION |
|---|---|---|---|
|  | Clock glitches<br>Voltage spikes<br>Lasers<br>FIB<br>… | Bit flip/set<br>Word randomize/set<br>Variable randomize/set<br>Instruction skip<br>… | Differential FA<br>Ineffective FA<br>Collision FA<br>… |

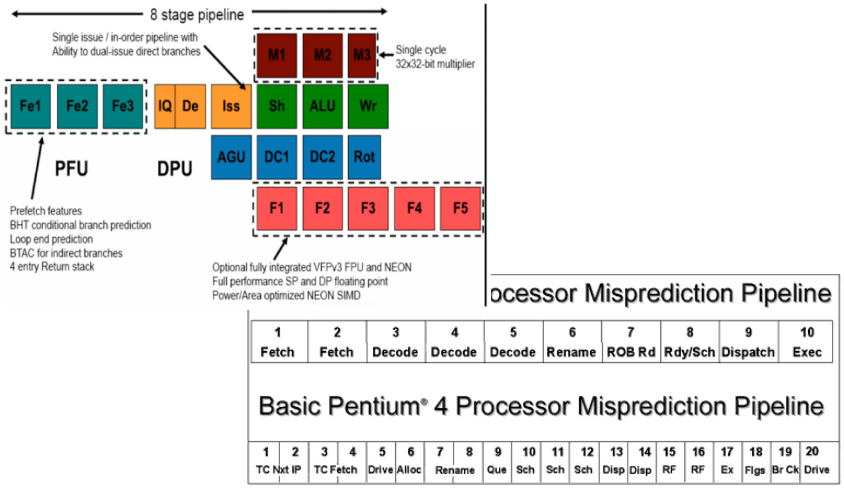
Our work:

- Fix target device and fault induction mechanism
- Full characterization of fault models

# Motivation (II)

## Fault induction mechanism

- Clock glitches
  - Non-invasive
  - Inexpensive




## Glitch effect vs. Target device

- Pipeline architecture

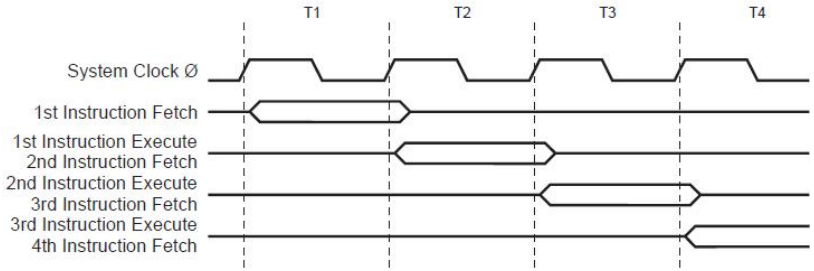
## Fault characterization vs. Experimental Platform

- Reproducibility
- Accuracy
- …

# Target Device

## Target device

- Atmel ATMega163
  - 8-bit microcontroller
  - Hardvard architecture
  - Externally clocked
  - RISC architecture
  - 2-stage pipeline

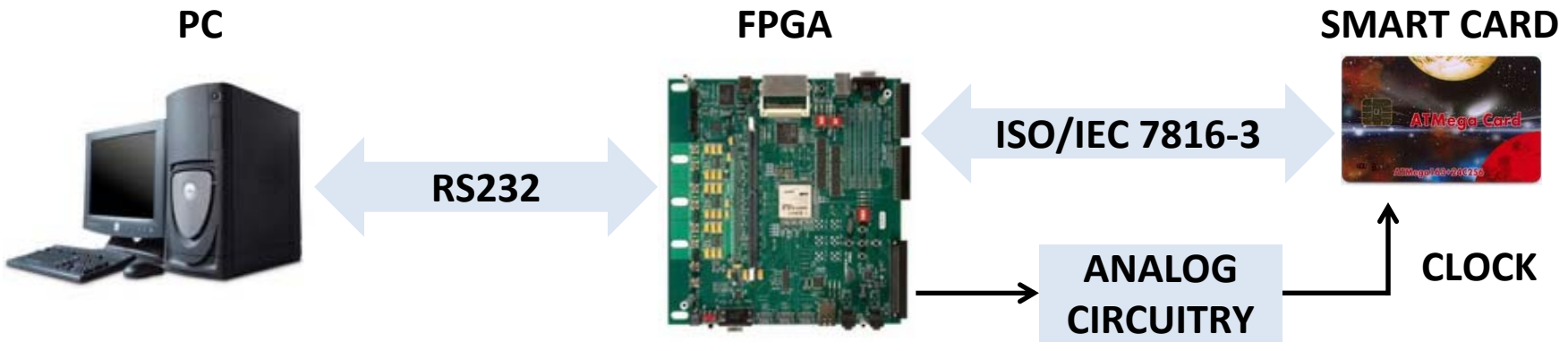

- Experiments on **five** cards

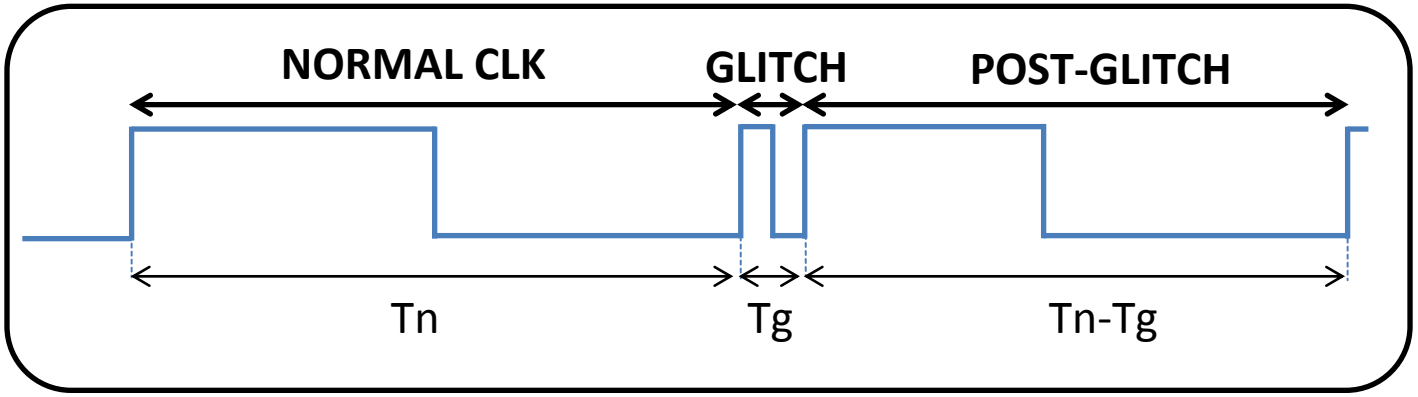
**DISCLAIMER**

Legacy smart card product

Not specifically designed for security applications [refer to Atmel ATxxSC family]

# Experimental Setup




DESIGN CRITERIA:

a) Reproducibility

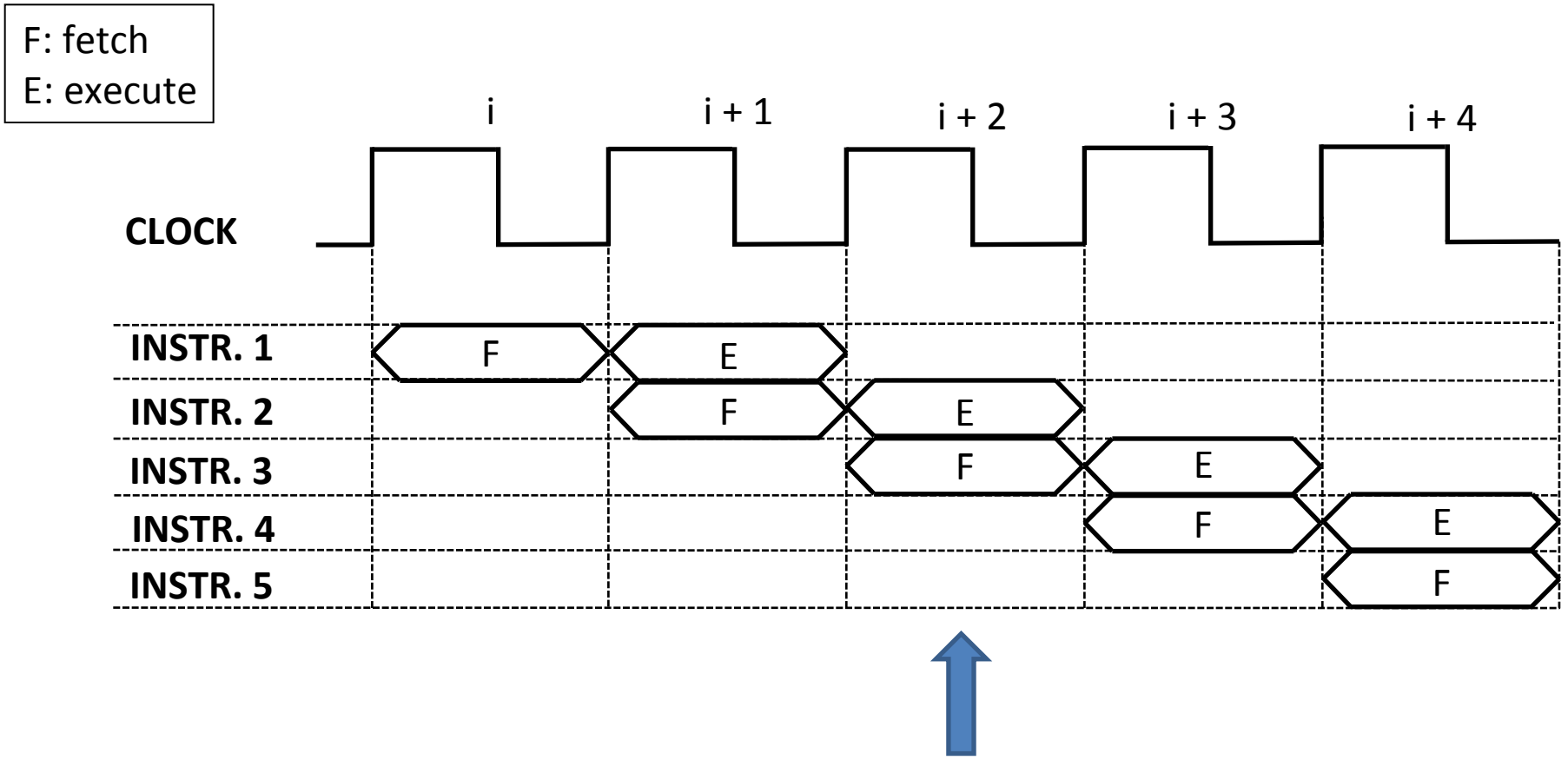
b) Flexibility

c) Automatization

For the rest of this presentation assume one glitch per execution:

- Upper bound: Tg = 15 ns (65 MHz)
- Accuracy: Steps of 1 ns

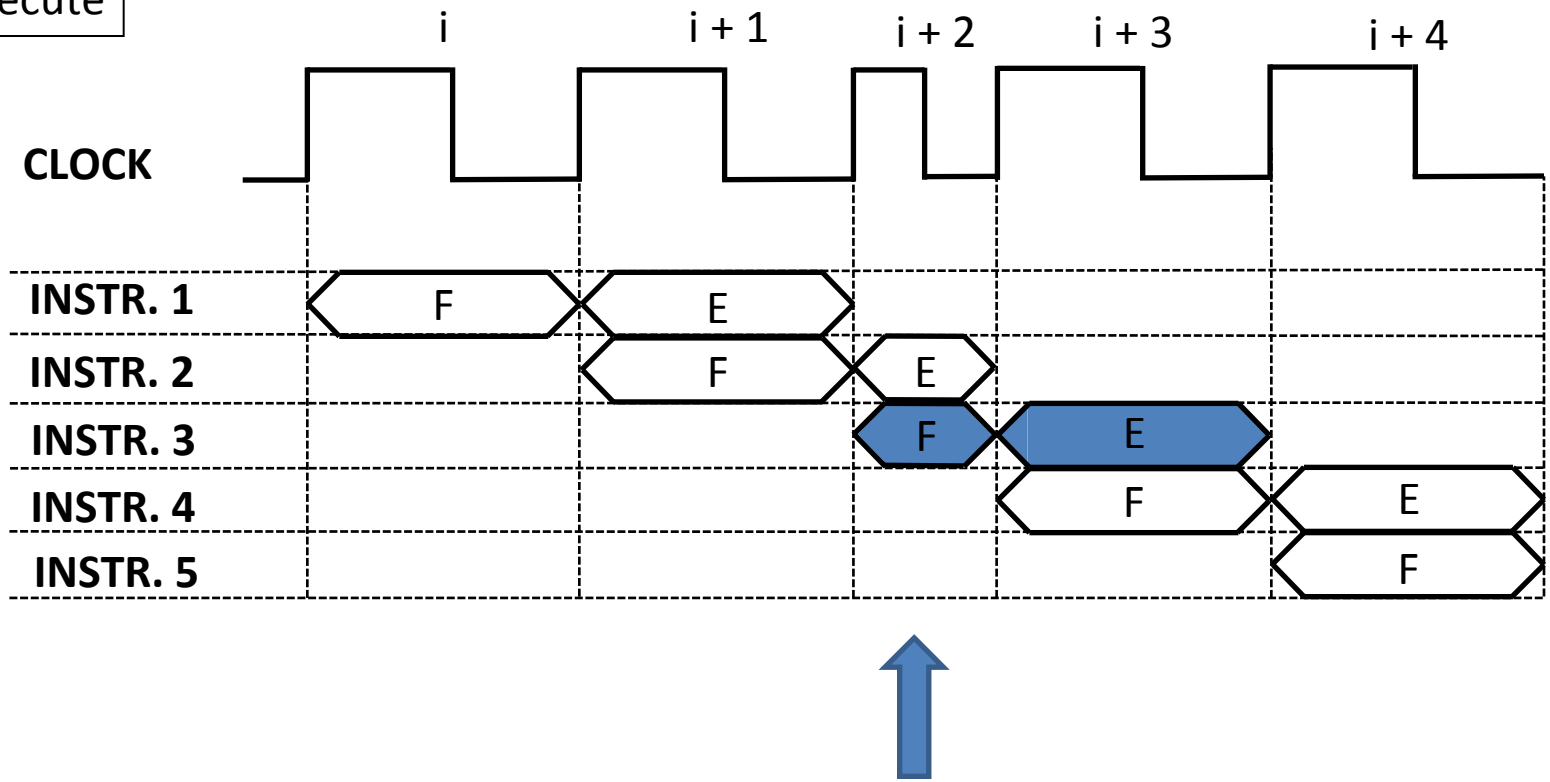
# Methodology (I)




Inject glitch in arbitrary clock cycle

# Methodology (I)

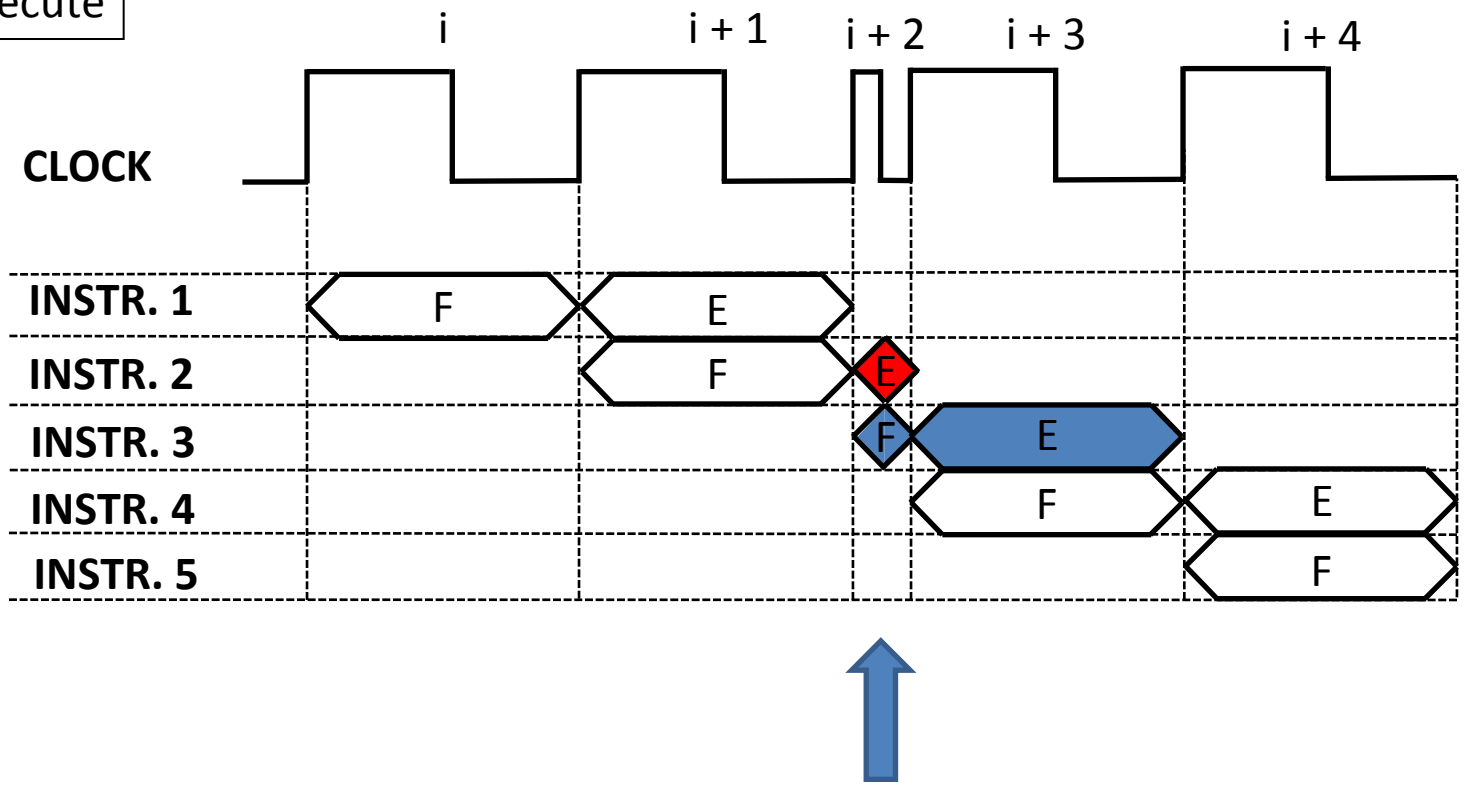
F: fetch
E: execute




Effects on **Program Flow** (i.e. fetching phase of pipeline)

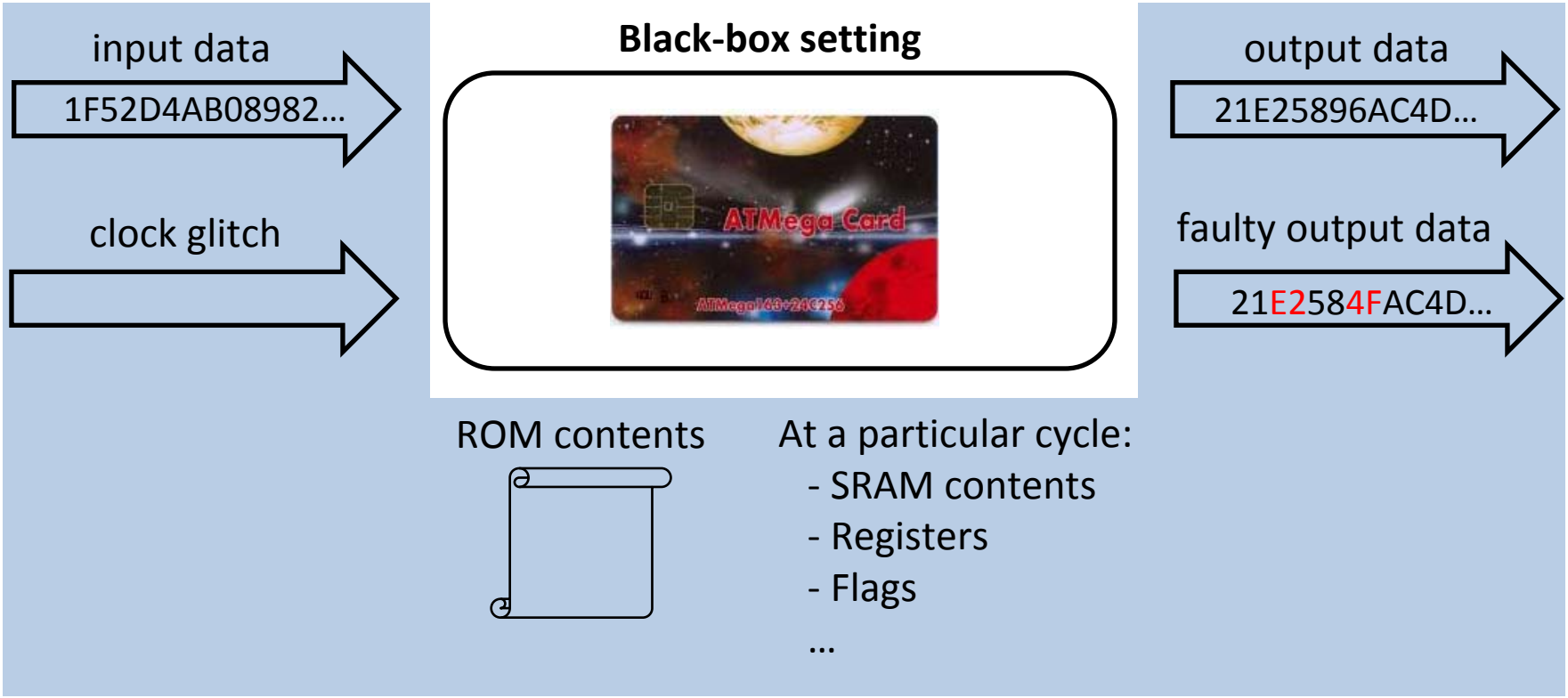
# Methodology (I)

F: fetch
E: execute




Effects on **Data Flow** (i.e. execution phase of pipeline)

# Methodology (II)



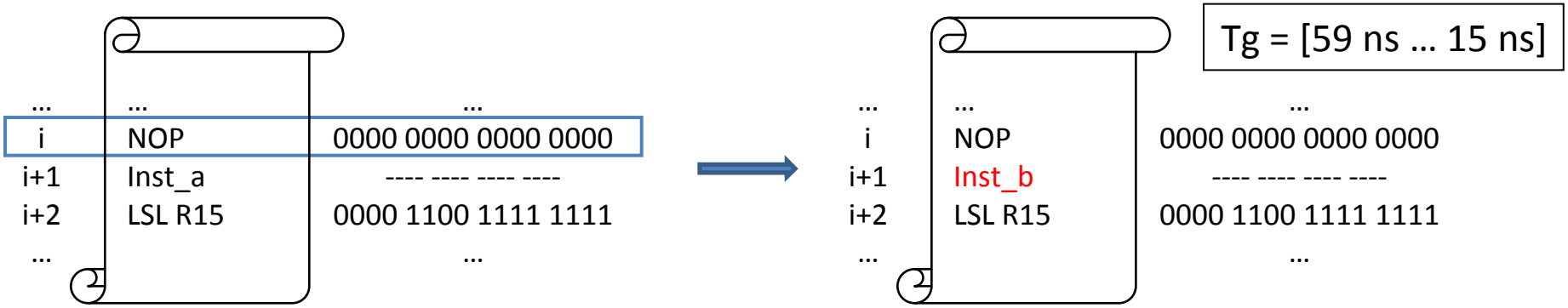

**REVERSE-ENGINEER FAULT EFFECT**

Examples to illustrate the observed effects

# Effects on Program Flow (I)

First experiments

- Commands that do not affect the data flow

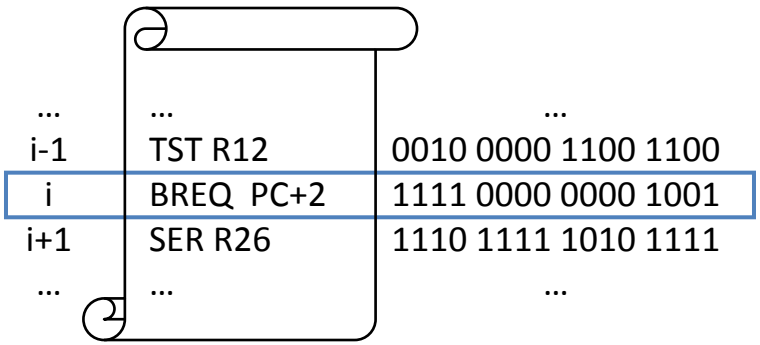



Observations:

- Fault model: **replacing** instructions
- Program Counter (PC) not affected by fault

# Effects on Program Flow (II)

Branching commands: Manipulate Program Counter



| | | |
|---|---|---|
| ... | ... | ... |
| i-1 | TST R12 | 0010 0000 1100 1100 |
| i | BREQ PC+2 | 1111 0000 0000 1001 |
| i+1 | SER R26 | 1110 1111 1010 1111 |
| ... | ... | ... |

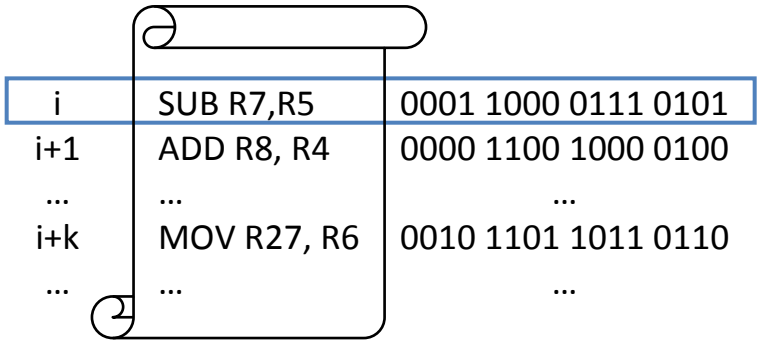
| | | | |
|---|---|---|---|
| Tg = 61 ns | i+1 | SER R26 | 1110 1111 1010 1111 |
| Tg = 57 ns | i+1 | LDI R26,0xEF | 1110 1110 1010 1111 |
| Tg = 56 ns | i+1 | LDI R26,0xCF | 1110 1100 1010 1111 |
| Tg = 52 ns | i+1 | LDI R26,0x0F | 1110 0000 1010 1111 |
| Tg = 45 ns | i+1 | LDI R16,0x09 | 1110 0000 0000 1001 |
| Tg = 32 ns | i+1 | LD R0,0x01 | 1000 0000 0000 1001 |
| Tg = 28 ns | i+1 | LD R0, Y | 1000 0000 0000 1000 |
| Tg = 27 ns | i+1 | LDI R16,0x09 | 1110 0000 0000 1001 |
| Tg = 15 ns | i+1 | BREQ PC+2 | 1111 0000 0000 1001 |

- Transition: new opcode → old opcode
- Single fault can affect both program flow **and** data flow

Same general effect
Different instructions

# Effects on Program Flow (III)

Rest of instruction set




Glitch widths:
[57 ns … 28 ns]

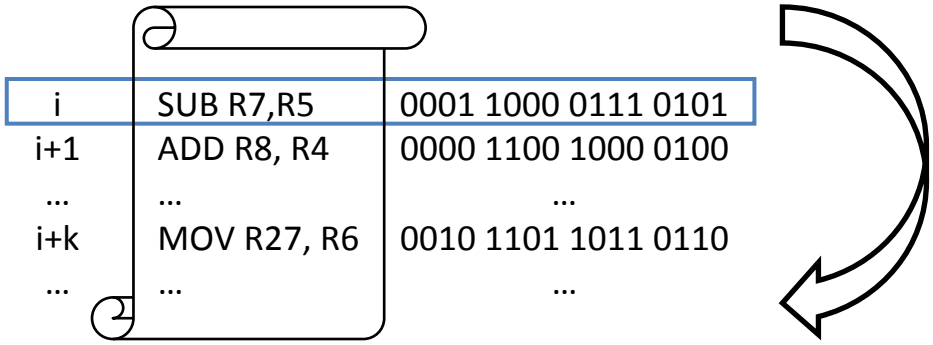
Observations:

- "Effective" skips ~ 50% of time
- Replaced by commands that do no affect data flow (NOPs, illegal commands, …)

Same effect

# Effects on Data Flow (I)

## Single-cycle instructions




Glitch widths: [27 ns … 15 ns]

Appearance of errors in data flow

Dependant on the specific instruction, registers used, and glitch width

## Observations:

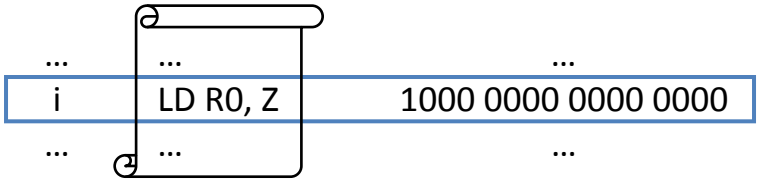
- Fault effect: **deterministic** values (not random!)
- Characterization is too complex…

Same general effect
Different errors!

# Effects on Data Flow (II)

Multi-cycle instructions

- LD (Load From Data Space)



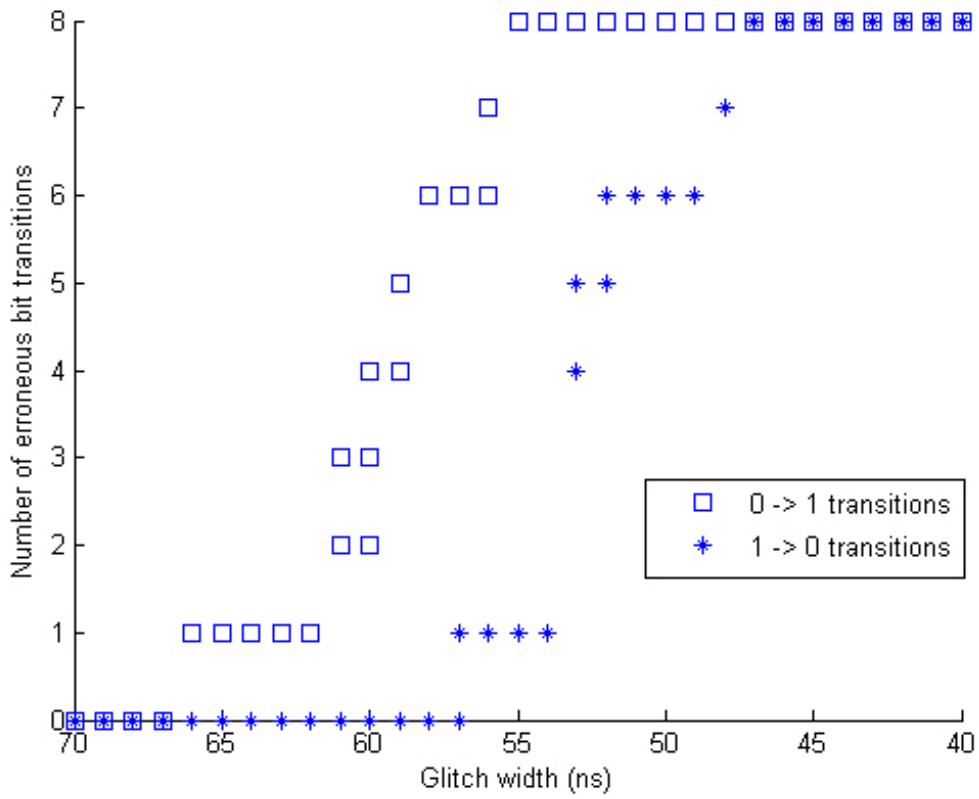

2-cycle instruction:
- 1st cycle: load address from Z
- 2nd cycle: load value from SRAM

Observations

- 1st cycle: value from erroneous SRAM address is loaded
  - Different depending on glitch width
- 2nd cycle: prevent bit transitions in data bus

# Effects on Data Flow (III)

## Erroneous transitions in data bus




No errors until Tg = 67 ns

Prevent 0->1 transitions in 1 bit
- Stable result (always bit 4)
- E.g. 0x00 -> 0xFF, fault 0xEF
- **Stuck-at-zero** model ?

For Tg ≤ 49 ns no transitions
- Data bus not updated !
- **Set word** model

Same general effect
Different values of Tg

# Summary / Conclusions

Effects of clock glitches on 2-stage pipeline MCU

Black-box study

- Limited knowledge of internal MCU hardware

Characterization of fault models

- Instructions can be **replaced**, rather than skipped
- Faults on data flow are **deterministic** and **reproducible**
- Easiest / More stable results for multi-cycle instructions
  - Stuck-at-zero bit and **set word** fault models stable

Possible to implement theoretical fault attacks

# Thanks for your attention

**QUESTIONS ?**