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
Motivation (I)

Our work:
- Fix target device and fault induction mechanism
- Full characterization of fault models
Fault induction mechanism

- Clock glitches
  - Non-invasive
  - Inexpensive

Glitch effect vs. Target device
- Pipeline architecture

Fault characterization vs. Experimental Platform
- Reproducibility
- Accuracy
- ...

Motivation (II)

FDTC 2011
28 September 2011
Target Device

Target device
- Atmel ATmega163
  - 8-bit microcontroller
  - Hardvard architecture
  -Externally clocked
  - RISC architecture
  - 2-stage pipeline

- Experiments on five cards
Experimental Setup

For the rest of this presentation assume one glitch per execution:
- Upper bound: $T_g = 15$ ns (65 MHz)
- Accuracy: Steps of 1 ns

DESIGN CRITERIA:
- a) Reproducibility
- b) Flexibility
- c) Automatization
Methodology (I)

Inject glitch in arbitrary clock cycle
Effects on **Program Flow** (i.e. fetching phase of pipeline)
Methodology (I)

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

input data: 1F52D4AB08982...

Black-box setting

clock glitch

output data: 21E25896AC4D...

faulty output data: 21E2584FAC4D...

ROM contents

At a particular cycle:
- SRAM contents
- Registers
- Flags
...

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

\[ Tg = [59 \text{ ns} \ldots 15 \text{ ns}] \]
Effects on Program Flow (II)

Branching commands: Manipulate Program Counter

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Opcode</th>
<th>Tg (ns)</th>
<th>i+1 Opcode</th>
<th>Tg (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TST R12</td>
<td>0010 0000 1100 1100</td>
<td>61</td>
<td>SER R26 1110 1111 1010 1111</td>
<td>57</td>
</tr>
<tr>
<td>BREQ PC+2</td>
<td>1111 0000 0000 1001</td>
<td>61</td>
<td>LDI R26,0xEF 1110 1110 1010 1111</td>
<td>57</td>
</tr>
<tr>
<td>SER R26</td>
<td>1110 1111 1010 1111</td>
<td>61</td>
<td>LDI R26,0xCF 1110 1100 1010 1111</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LDI R26,0xF 1110 0000 1010 1111</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LDI R16,0x09 1110 0000 0000 1001</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LDI R16,0x09 1110 0000 0000 1001</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LDI R16,0x09 1110 0000 0000 1001</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LDI R16,0x09 1110 0000 0000 1001</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LDI R16,0x09 1110 0000 0000 1001</td>
<td>45</td>
</tr>
</tbody>
</table>

- Transition: new opcode → old opcode
- Single fault can affect both program flow and data flow
Effects on Program Flow (III)

Rest of instruction set

<table>
<thead>
<tr>
<th>i</th>
<th>SUB R7,R5</th>
<th>0001 1000 0111 0101</th>
</tr>
</thead>
<tbody>
<tr>
<td>i+1</td>
<td>ADD R8, R4</td>
<td>0000 1100 1000 0100</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>i+k</td>
<td>MOV R27, R6</td>
<td>0010 1101 1011 0110</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>i</th>
<th>SUB R7, R5</th>
<th>0001 1000 0111 0101</th>
</tr>
</thead>
<tbody>
<tr>
<td>i+1</td>
<td>ADD R8, R4</td>
<td>0000 1100 1000 0100</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>i+k</td>
<td>MOV R27, R6</td>
<td>0010 1101 1011 0110</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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)

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

2-cycle instruction:
- 1st cycle: load address from Z
- 2nd cycle: load value from SRAM
Erroneous transitions in data bus

No errors until $T_g = 67$ ns

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

For $T_g \leq 49$ ns no transitions
- Data bus not updated!
- **Set word** model

Same general effect
Different values of $T_g$
Effects of clock glitches on 2-stage pipeline MCU

Black-box study
  o Limited knowledge of internal MCU hardware

Characterization of fault models
  o Instructions can be replaced, rather than skipped
  o Faults on data flow are deterministic and reproducible
  o 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?