Software Fault Resistance is Futile: Effective Single-Glitch Attacks

B. Yuce, **N. F. Ghalaty**, H. Santapuri, C. Deshpande, C. Patrick, P. Schaumont

Virginia Tech

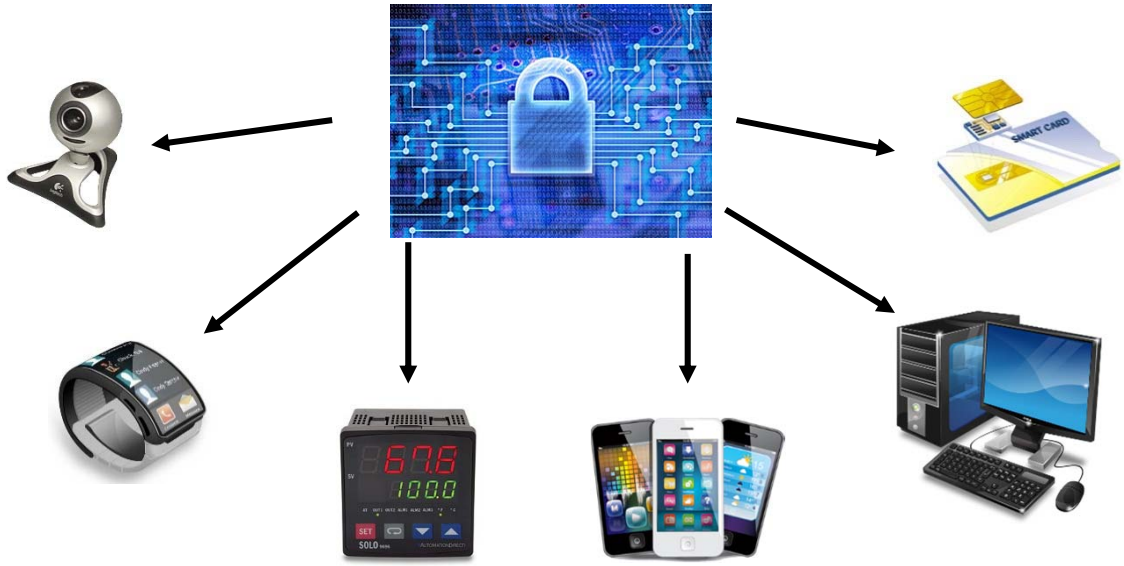
Bradley Department of ECE

farhady@vt.edu

This research was supported through NSF Grant 1441710, Grant 1115839, and through SRC.

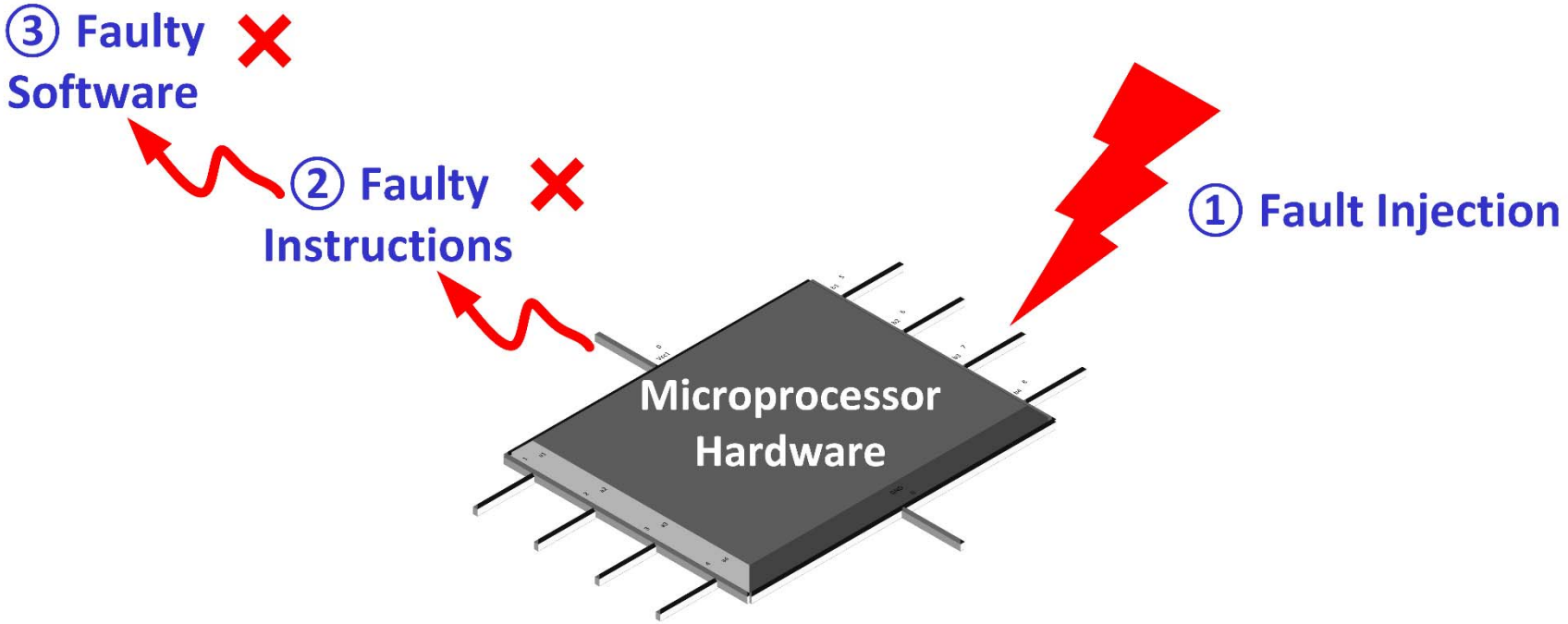
Fault Attack on Embedded Systems

- Fault attacks pose a serious threat to embedded systems:
  - To gain the control of a software program
  - To observe the secret information

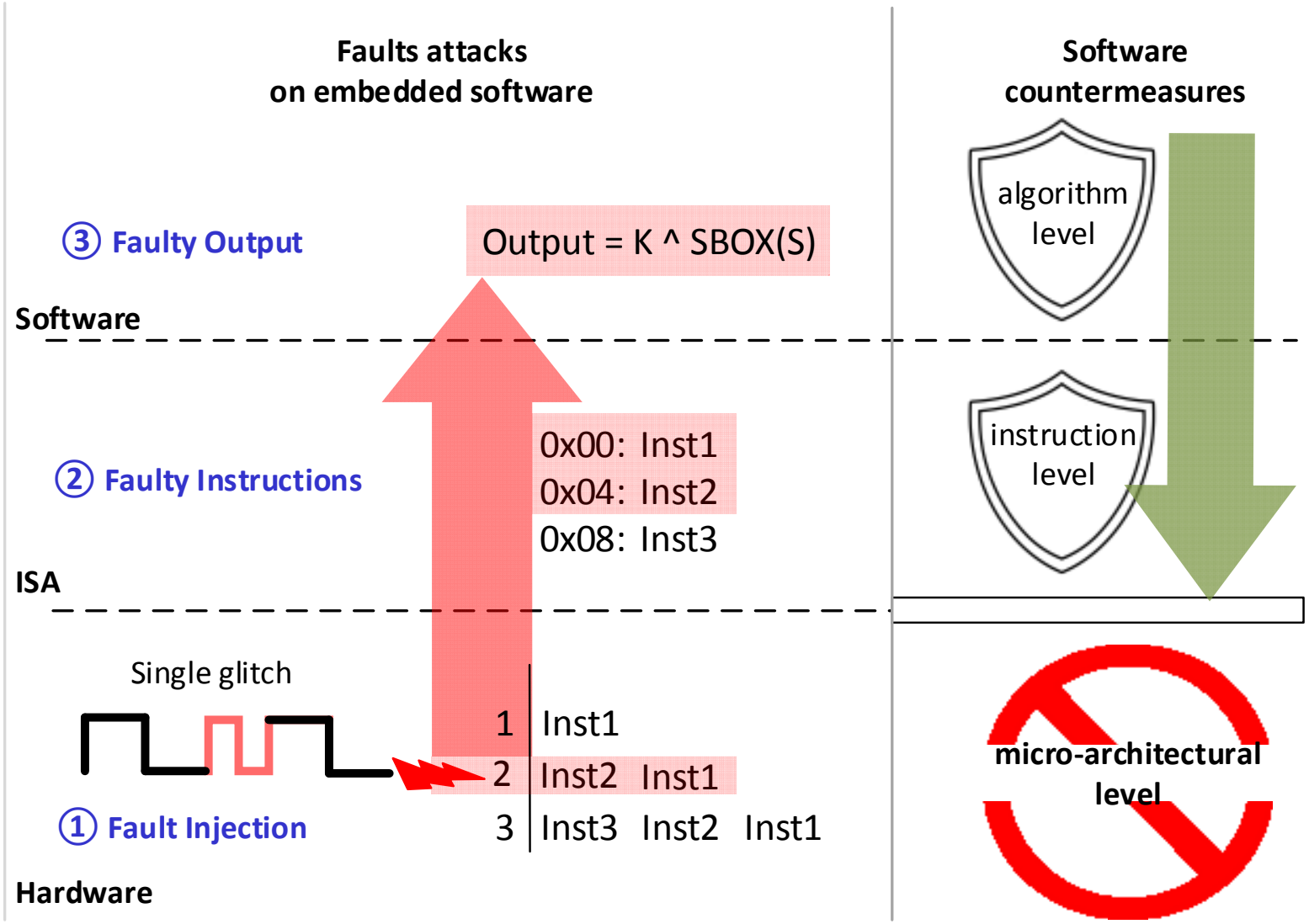


Fault Injection into Embedded Software

- Hardware determines the fault behavior of software.

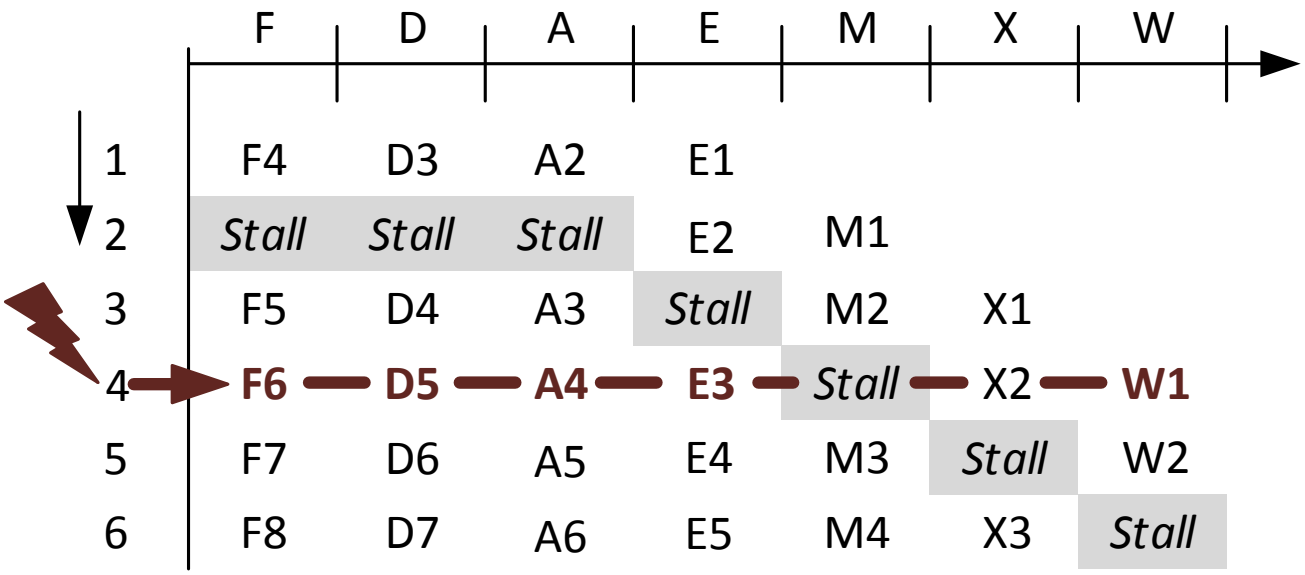


Software Countermeasures



Fault Behavior in a RISC Pipeline (1)

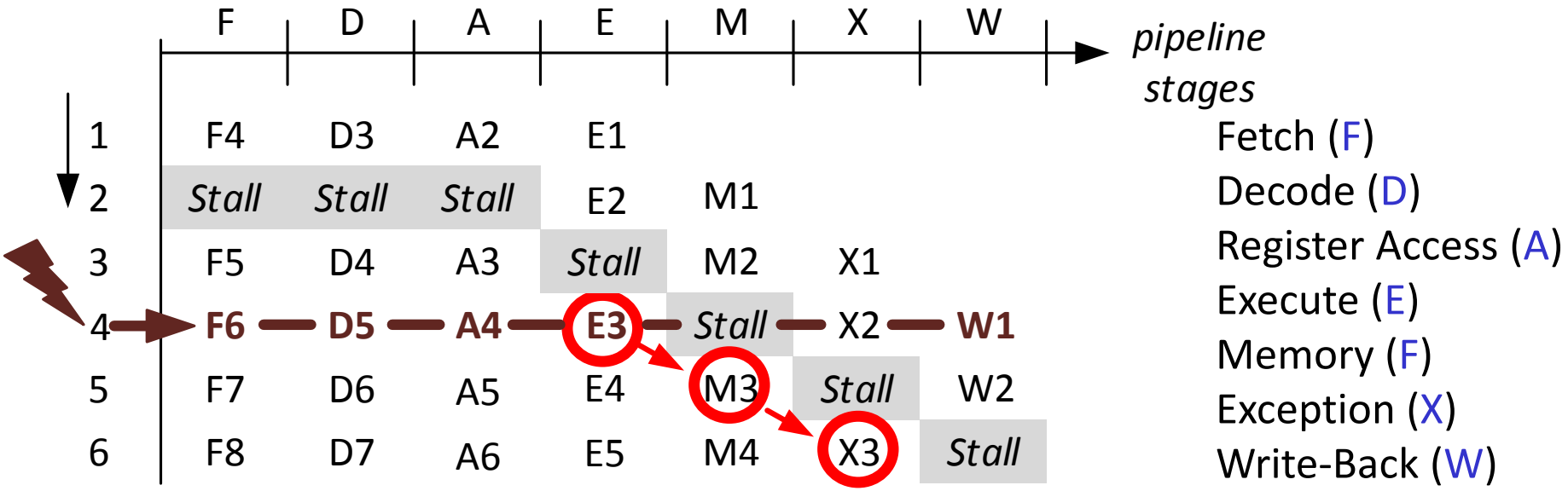
- 7-Stage RISC Pipeline:



Fetch (F)
Decode (D)
Register Access (A)
Execute (E)
Memory (F)
Exception (X)
Write-Back (W)

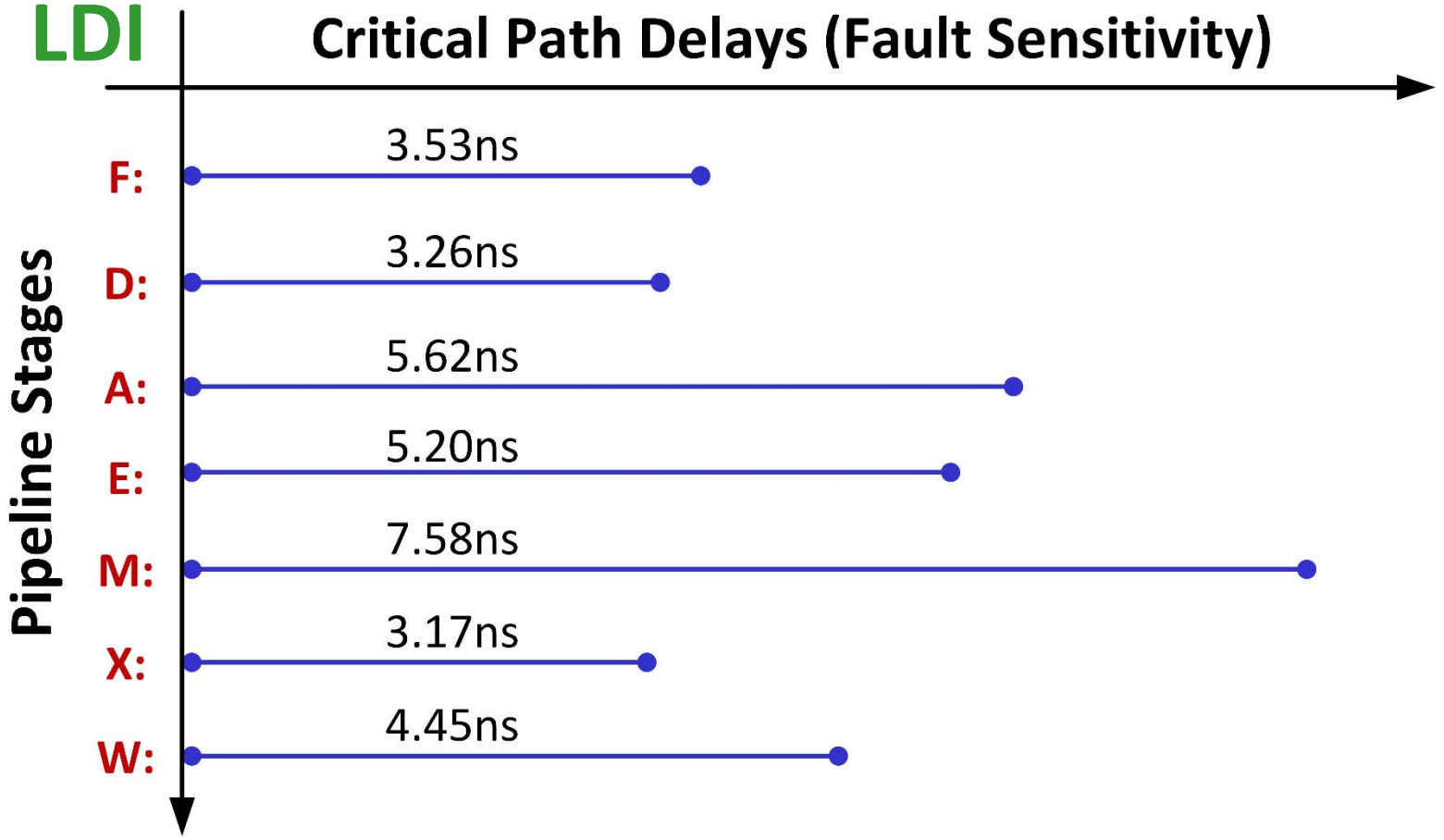
Fault Behavior in a RISC Pipeline (2)

- If E3 has the highest critical path (i.e, fault sensitivity):



How to Tune Glitch Parameters?

- Microprocessor Fault sensitivity Model of each (instruction, pipeline stage)



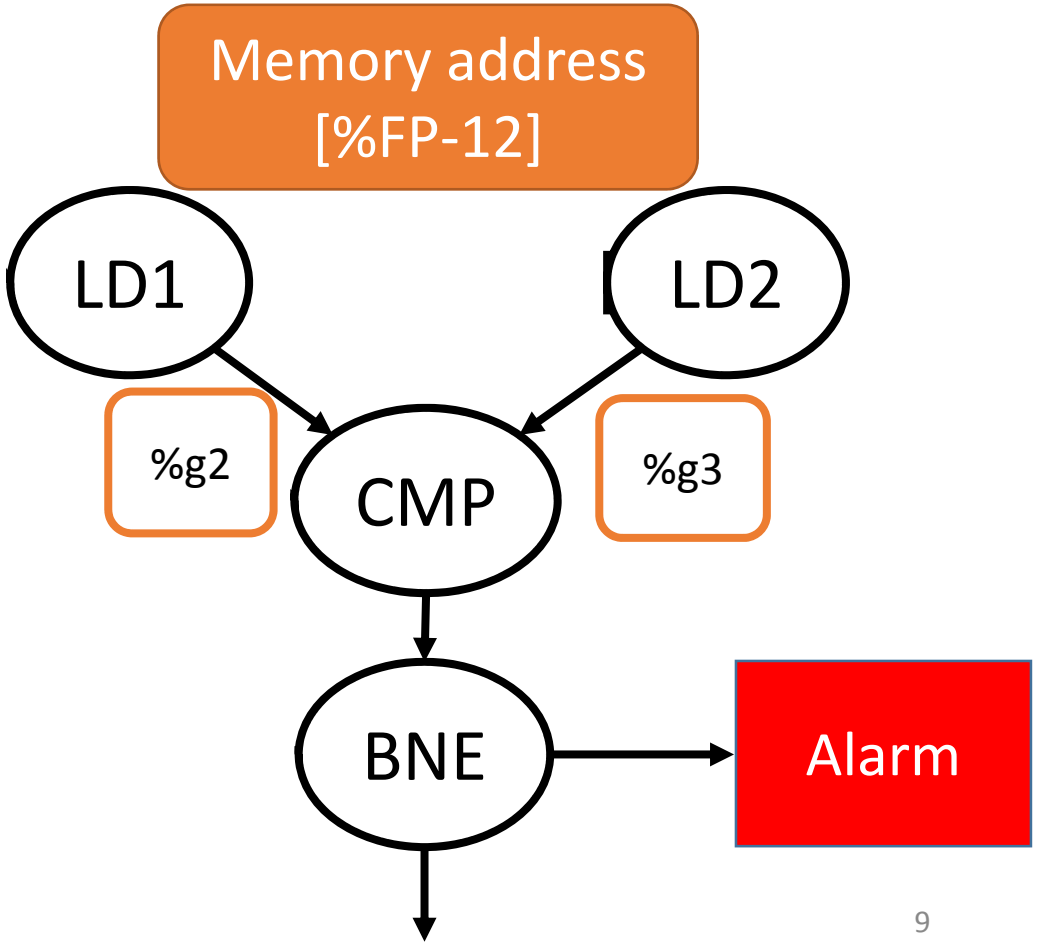
Fault Attack on Software Countermeasures

Instruction Duplication Countermeasure

Objective:

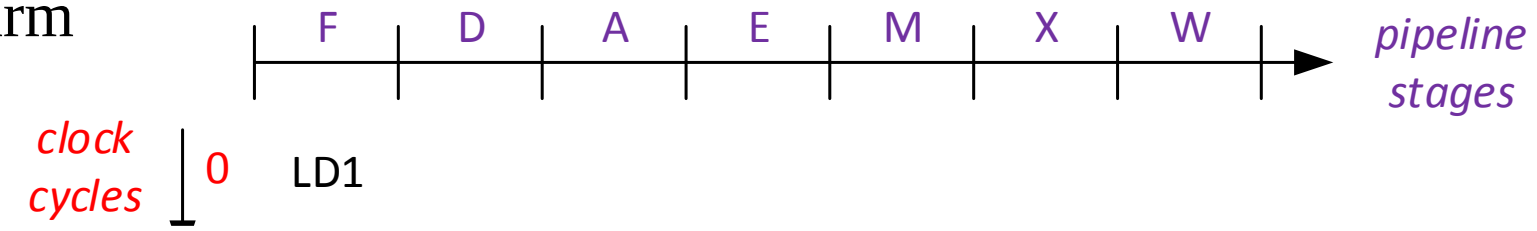
Keep two copies of an Instruction, Raise an alarm in case of a mismatch

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .alarm
```



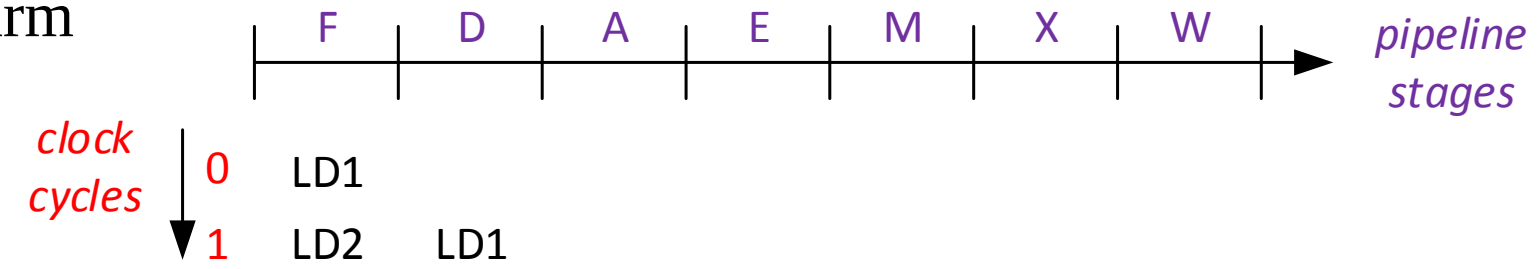
ID Behavior in Pipeline

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .alarm
```



ID Behavior in Pipeline

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .alarm
```

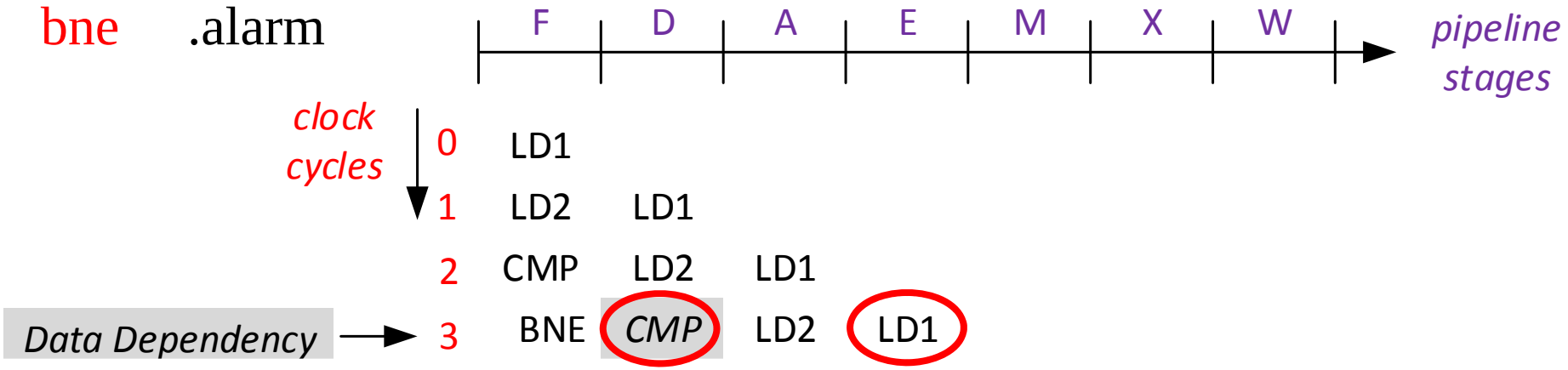


ID Behavior in Pipeline

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .alarm
```

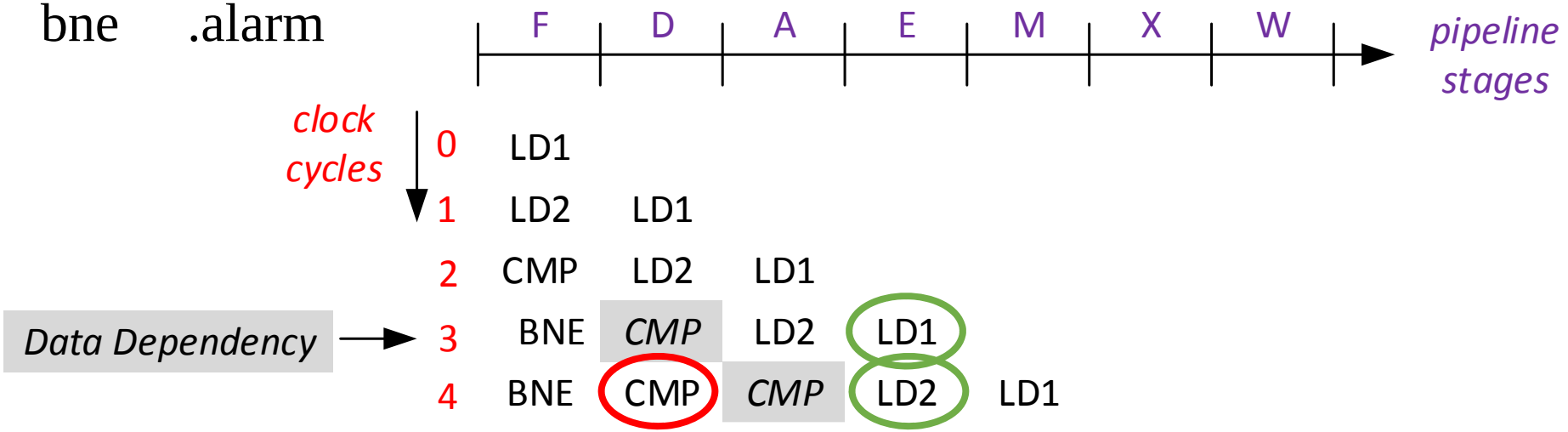
ID Behavior in Pipeline

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .alarm
```



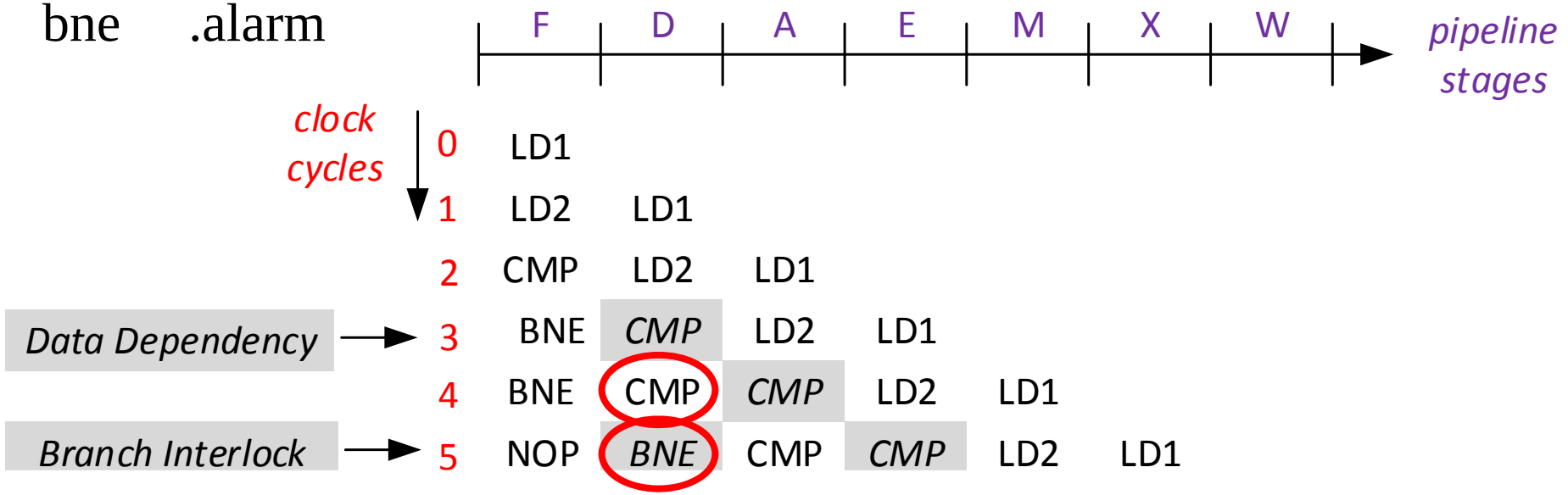
ID Behavior in Pipeline

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .alarm
```



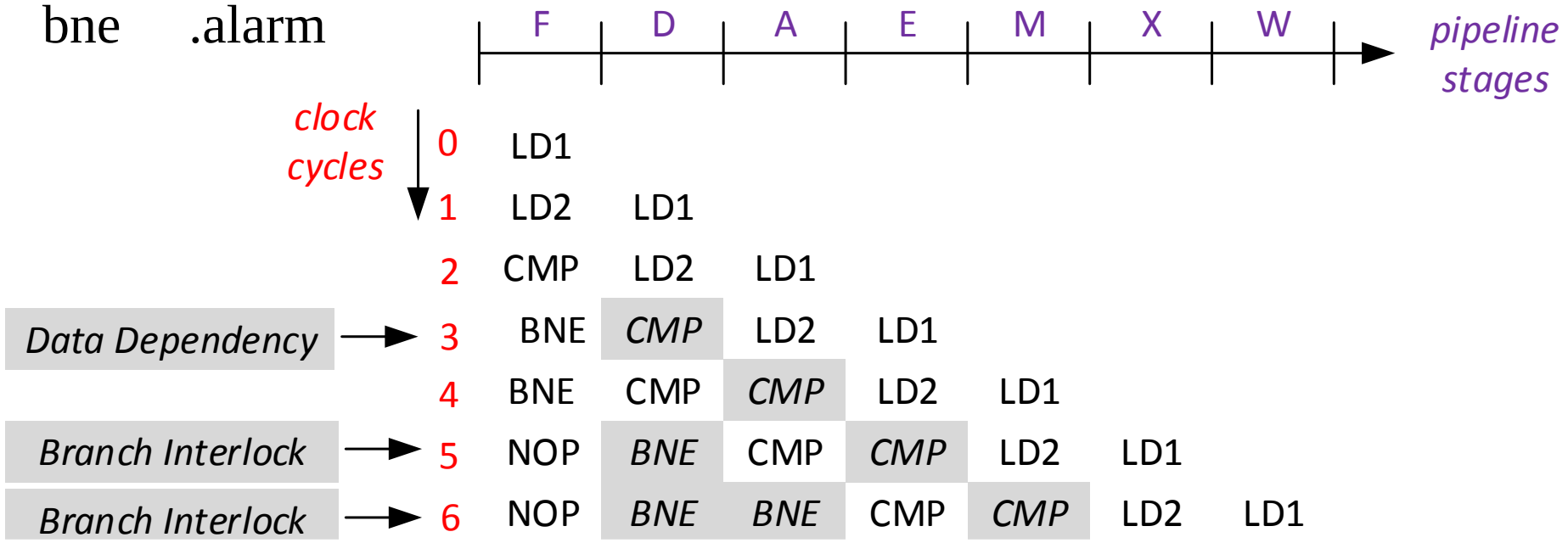
ID Behavior in Pipeline

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .alarm
```



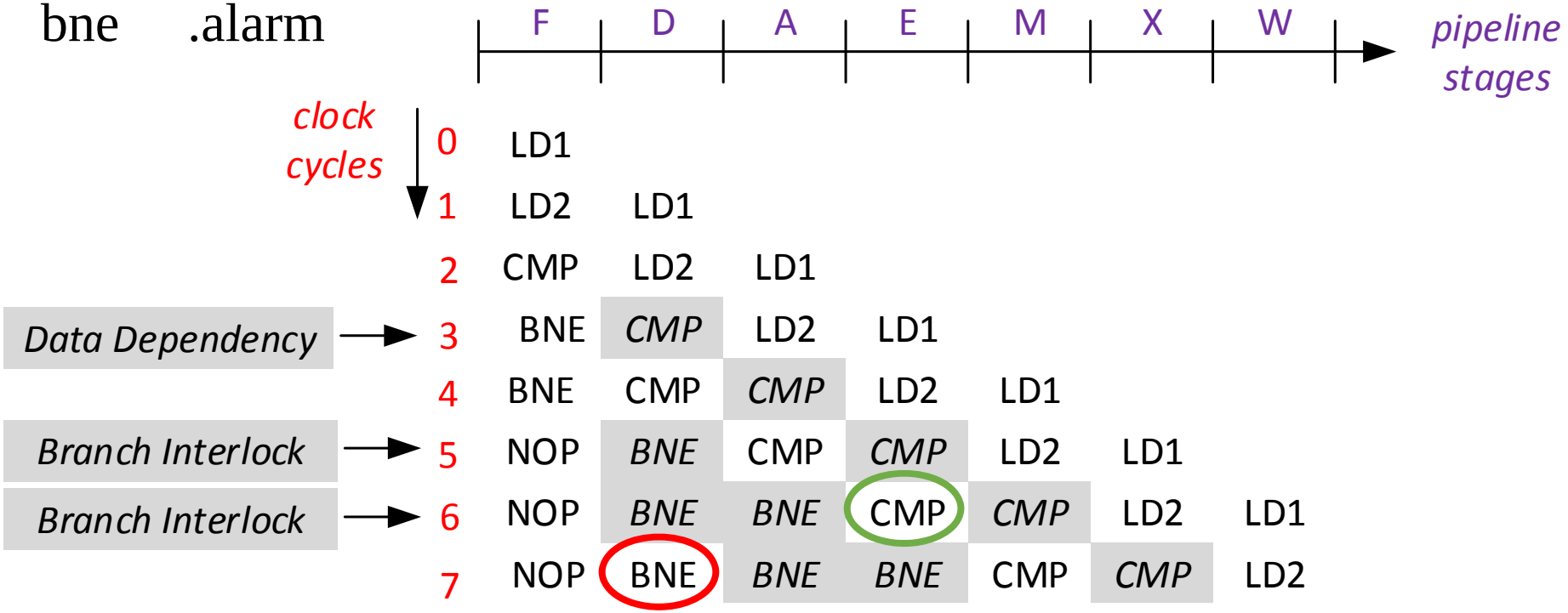
ID Behavior in Pipeline

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .alarm
```



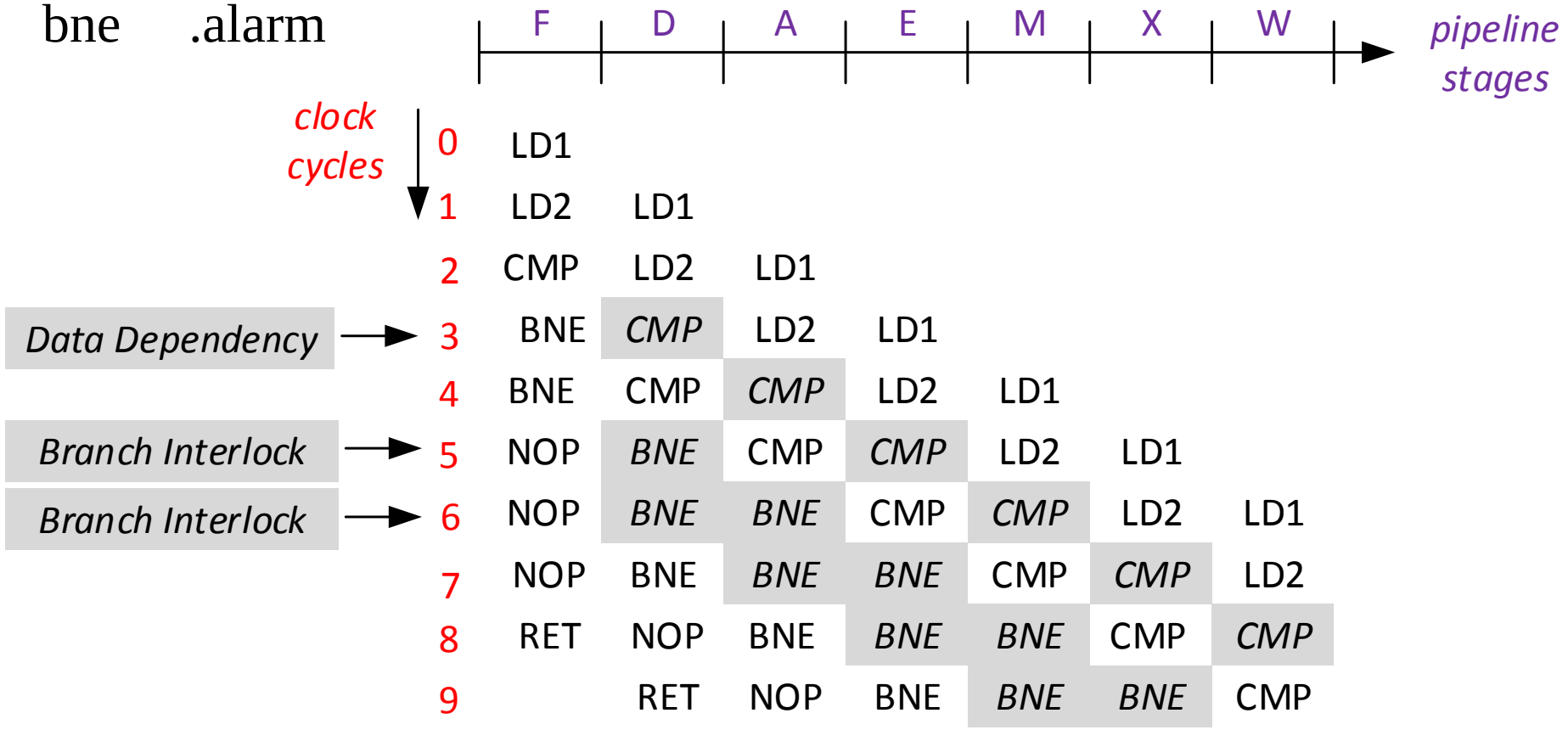
ID Behavior in Pipeline

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .alarm
```



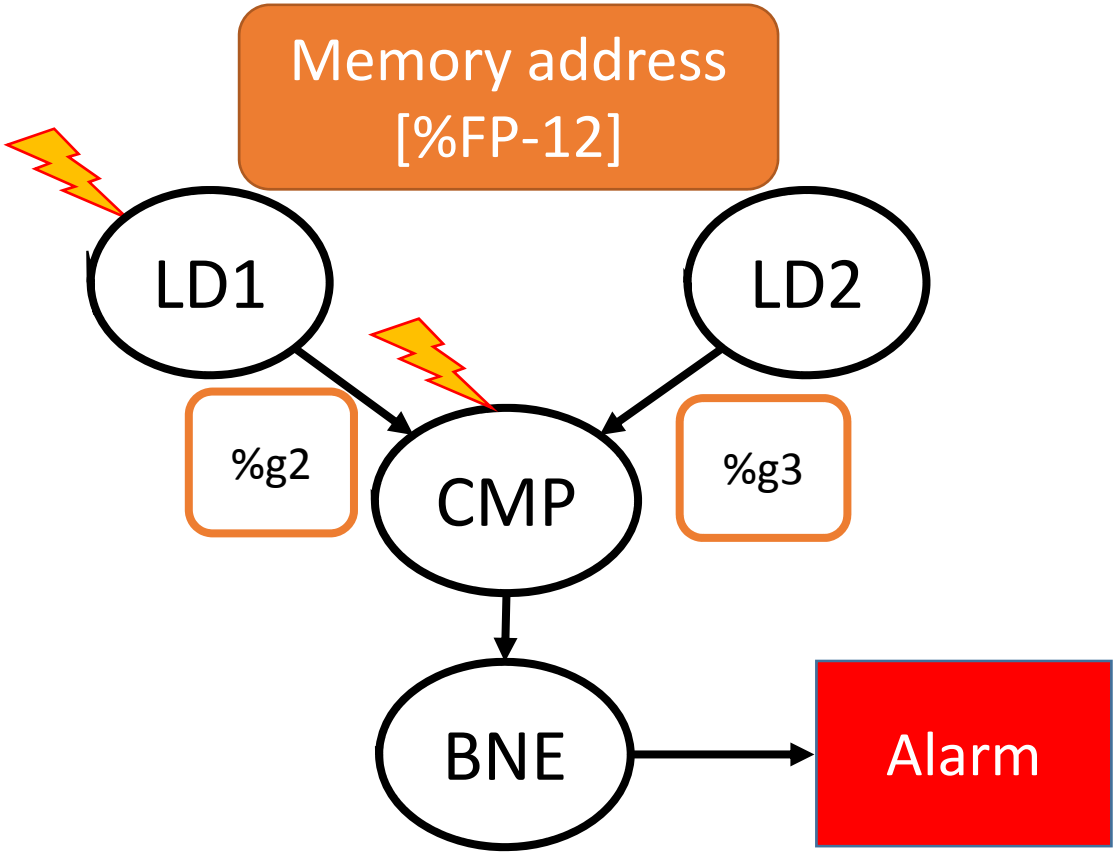
ID Behavior in Pipeline

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .alarm
```



Fault Model

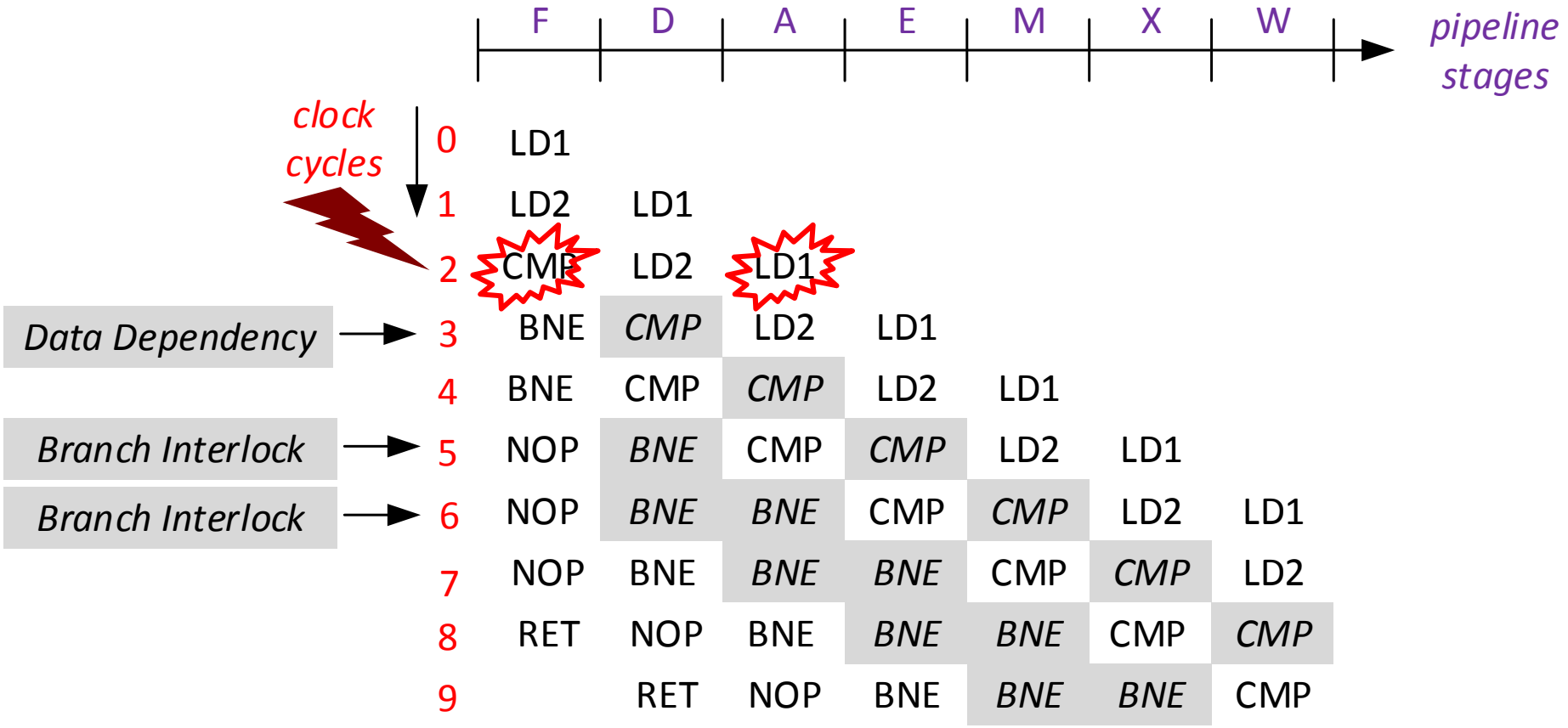
- Inject fault in %g2
- Avoid raising the alarm



Attack Scenario

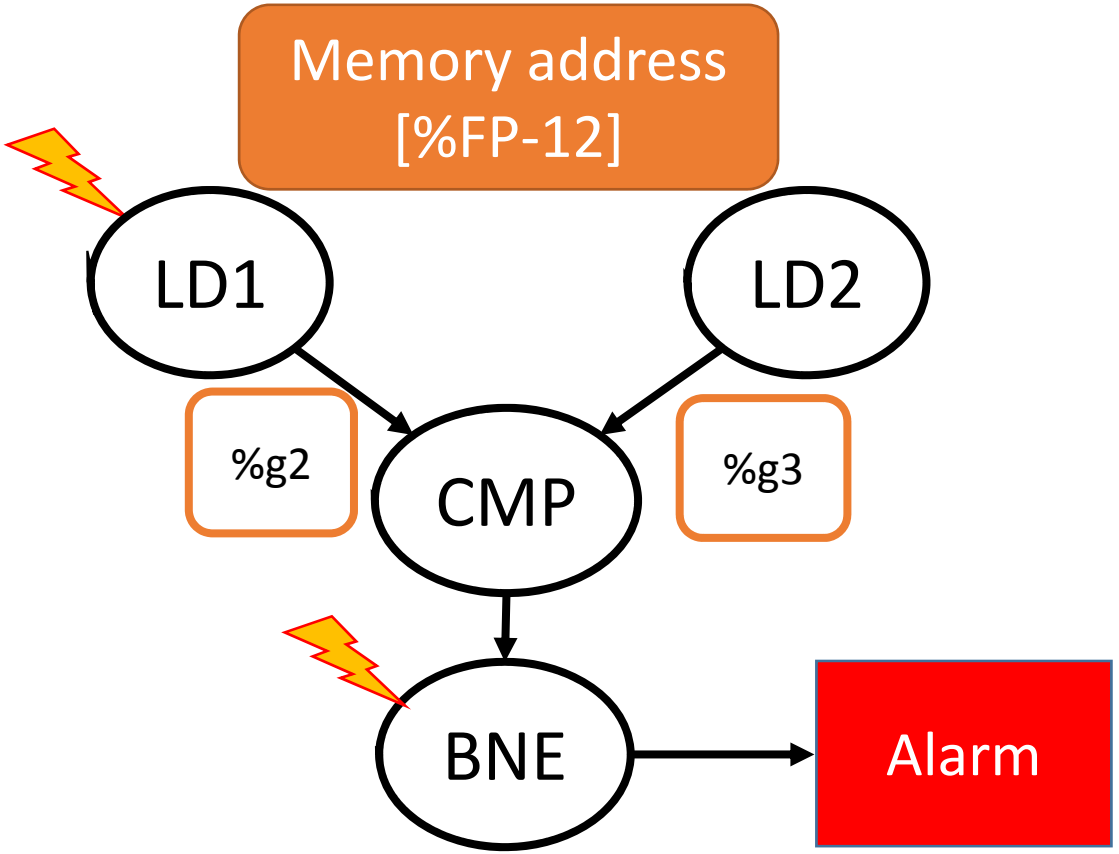
Scenario 1: Single Glitch

- Instruction Fault in CMP
- Computation Fault in LD1



Fault Model

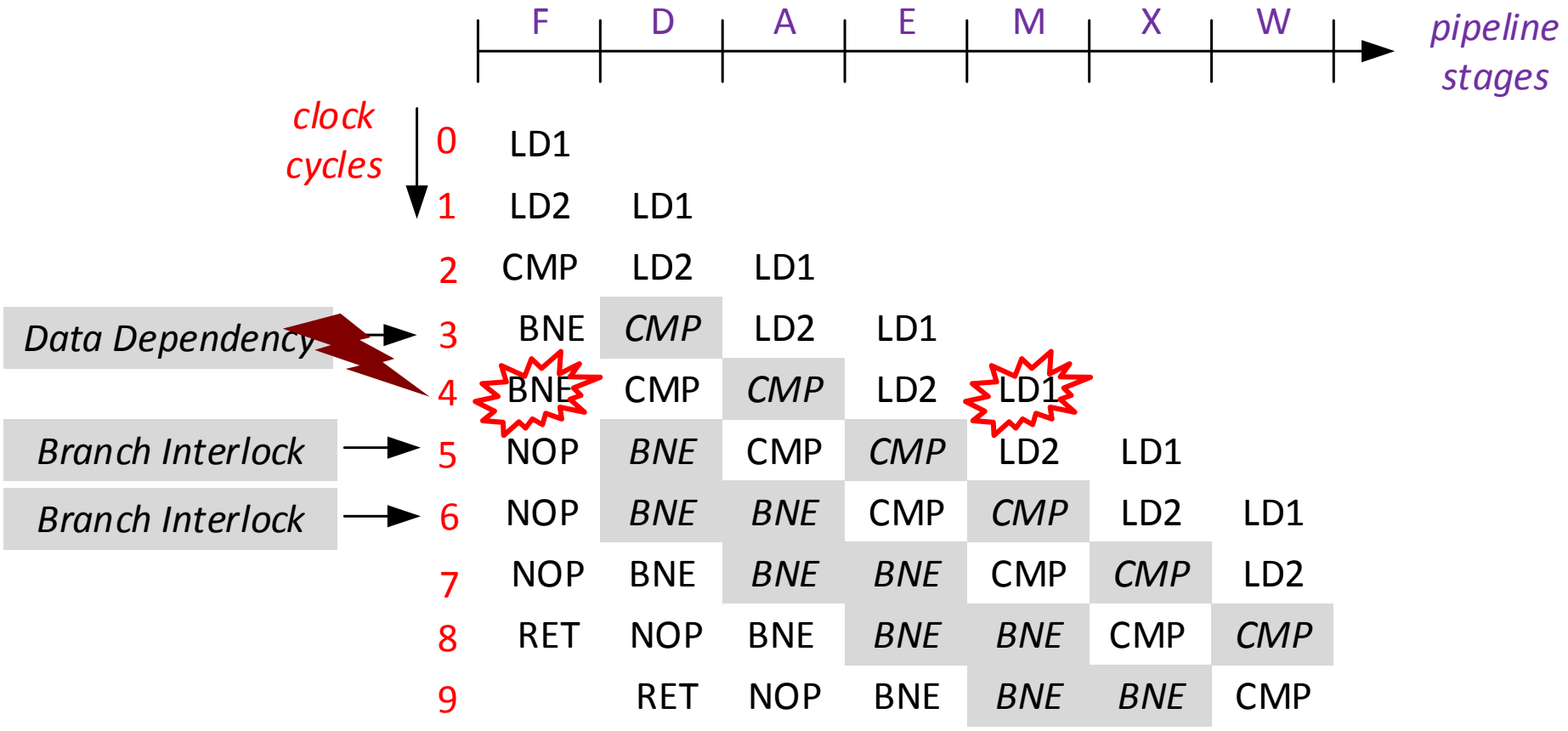
- Inject fault in %g2
- Avoid raising the alarm



Attack Scenario

Scenario 2: Single Glitch

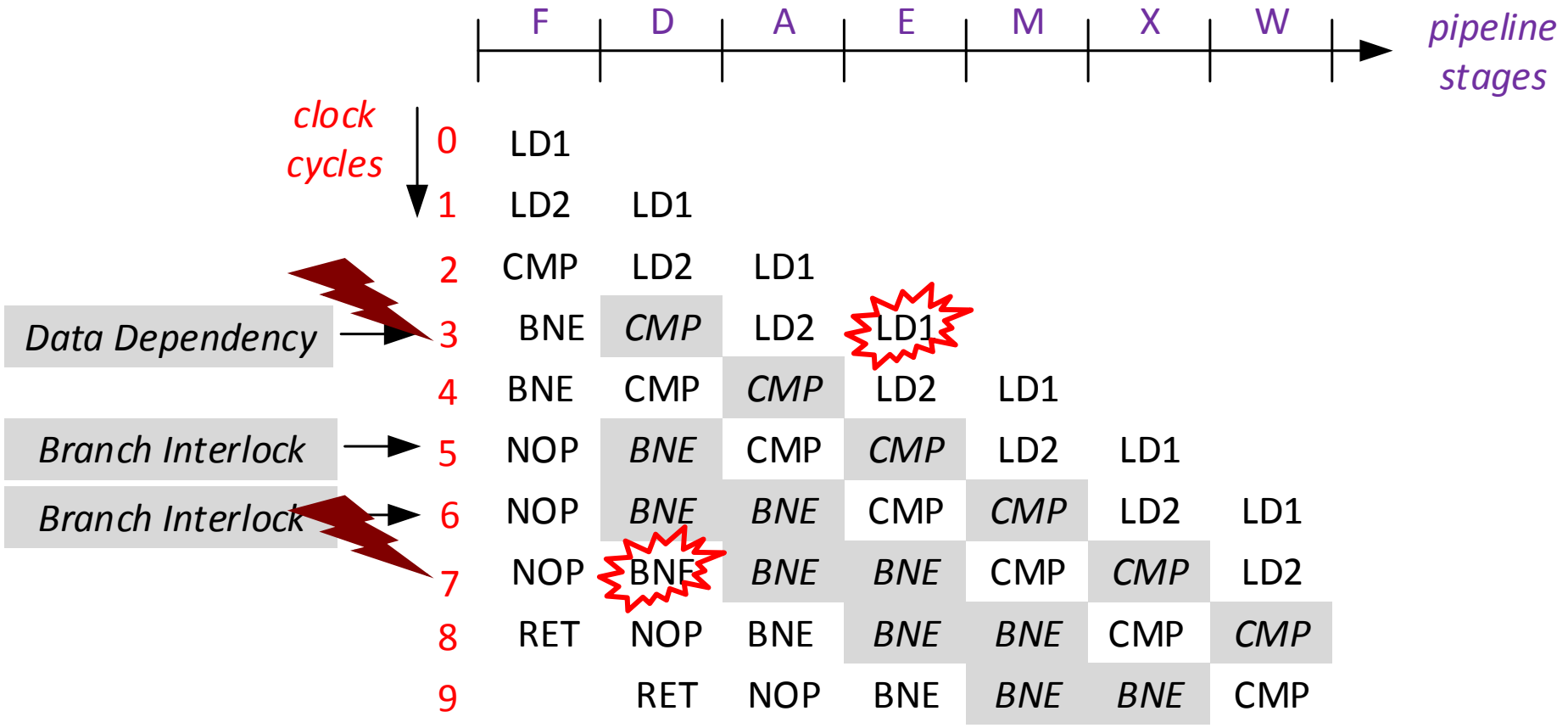
- Instruction Fault in BNE
- Computation Fault in LD1



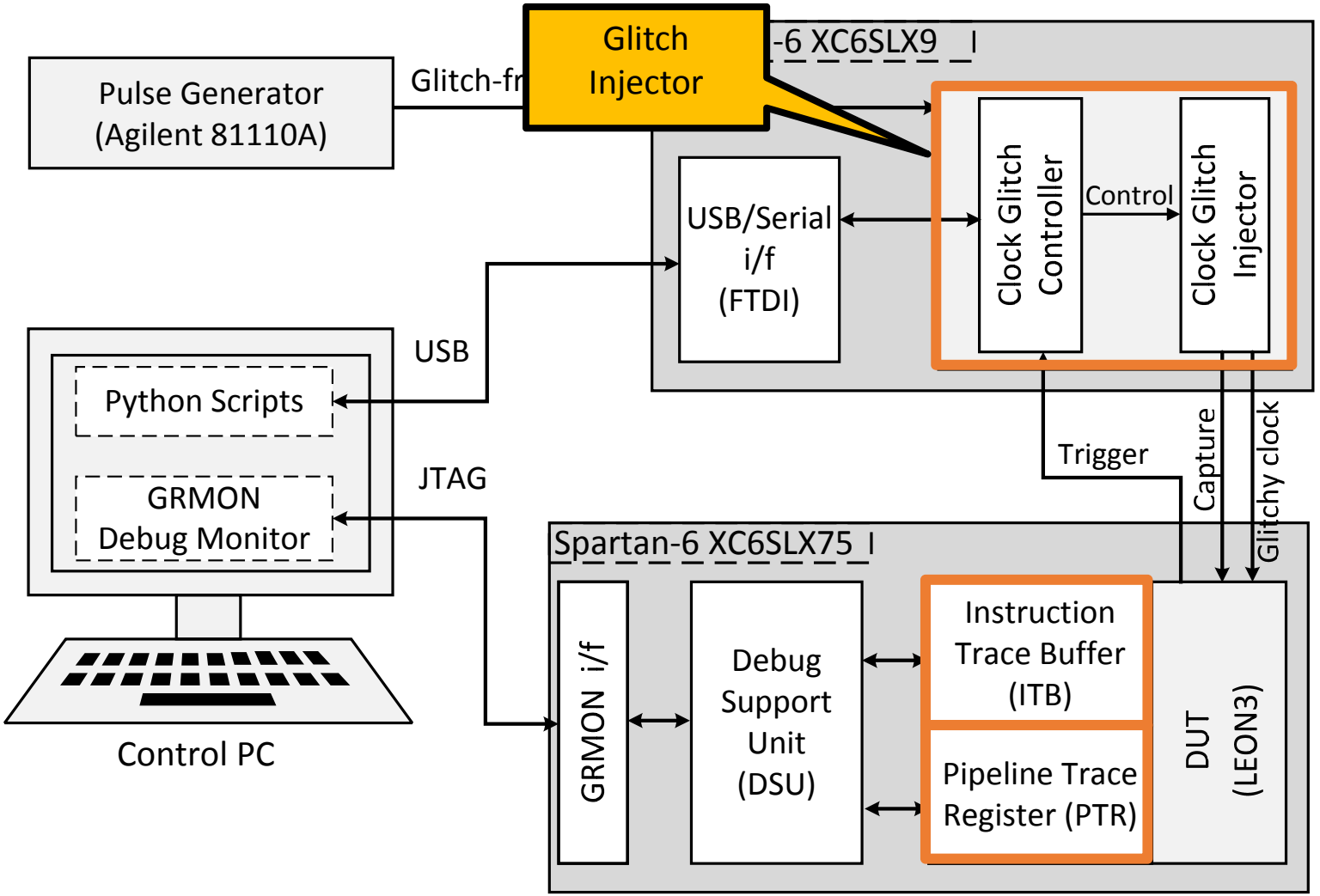
ID Behavior in Pipeline

Scenario 3: Multiple Glitch

- Computation Fault in LD1
- Instruction Fault in BNE



Experimental Setup



Verification on Prototype

```
ld      [%fp - 12], %g2
ld      [%fp - 12], %g3
cmp     %g2, %g3
bne     .error
```

	Glitch FI (ns)	Impacted Instruction	Fault Effect
Scenario 1	**9 – 12.6**	**LD1 (A) CMP (D)**	**Faulty %g2 CMP → SRL**
Scenario 2	**12 – 14.6**	**LD1 (M) BNE (F)**	**Faulty %g2 BNE → NOP**
Scenario 3	**14.7-14.9 12.6 – 13.5**	**LD1 (E) BNE (D)**	**Faulty %g2 BNE → NOP**

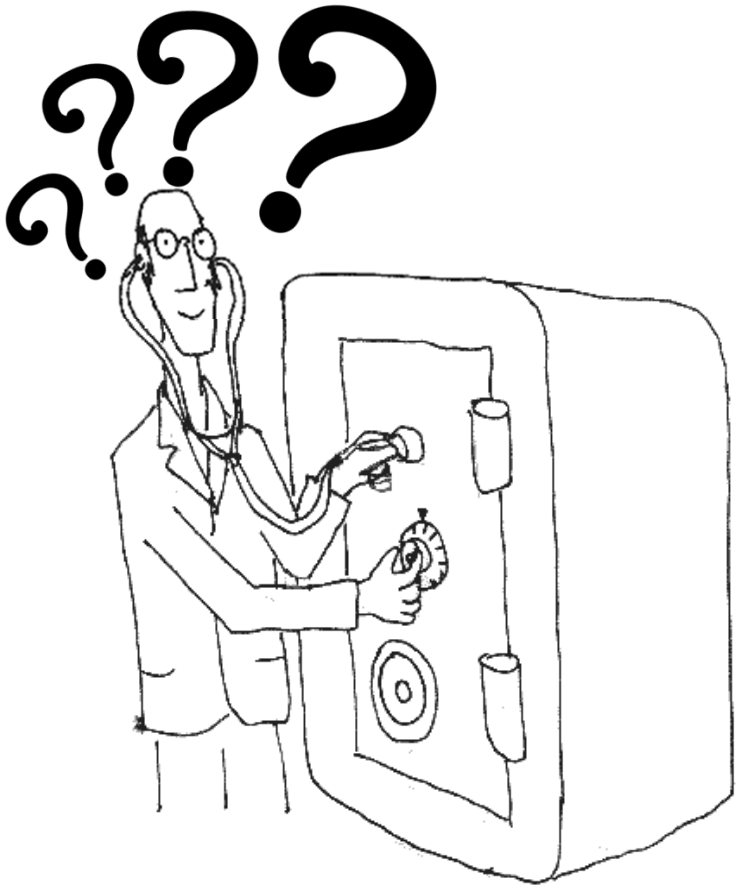
Experimental Results

- We applied the same strategy to several software countermeasures including
  - Instruction Duplication
  - Instruction Triplication
  - Parity Checking
  - Instruction Skip Countermeasures
- We successfully launched the DFIA attack on a software implementation of the LED algorithm, protected with parity checking and Instruction Duplication.
- We changed the width of the glitch from *31.2ns* to *33.6ns* with step size of *0.1ns*. Using, these fault injections, we obtained 5 faulty ciphertexts and retrieved 2 nibbles of the key.

Conclusions

- Efficient fault attacks on embedded software consider:
  - Architectural properties
  - Micro-architectural properties
- Microprocessor Fault Sensitivity Model is instrumental to predict the fault response of software.
- Traditional software countermeasures are vulnerable to single glitch fault attacks.

Thank you!



N. F. Ghalaty
farhady@vt.edu

Case Study

- Case Study:
  - Fault Analysis: Differential Fault Intensity Analysis (DFIA)
  - Software: LED
  - Hardware: LEON3 Processor

- DFIA [Ghalaty et. al, FDTC'14]:
  - Relies on a biased fault behavior
  - Gradual fault behavior **in proportion to** the fault intensity

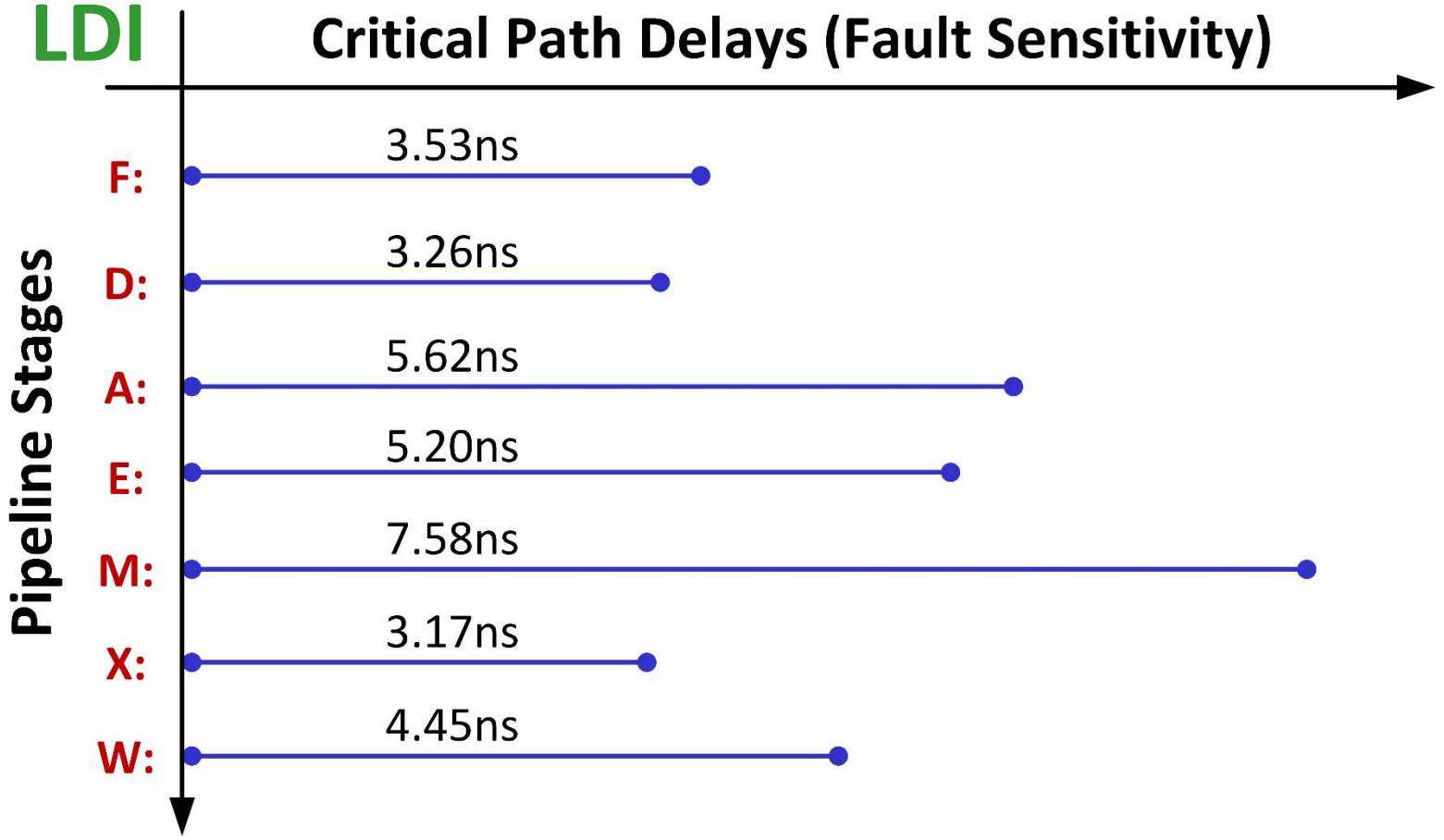
Fault Attack Resistant LED Implementation

- Parity Checking Countermeasure
  - Algorithm Level
  - Exploits a parity prediction circuit to predict parity for each input, raises an alarm if the computed parity does not match the predicted one

- Instruction Duplication
  - Instruction Level
  - Duplicates selected critical parts of the code such as Add Round Key Function

How to Tune Glitch Parameters?

- Microprocessor Fault sensitivity Model of each (instruction, pipeline stage)

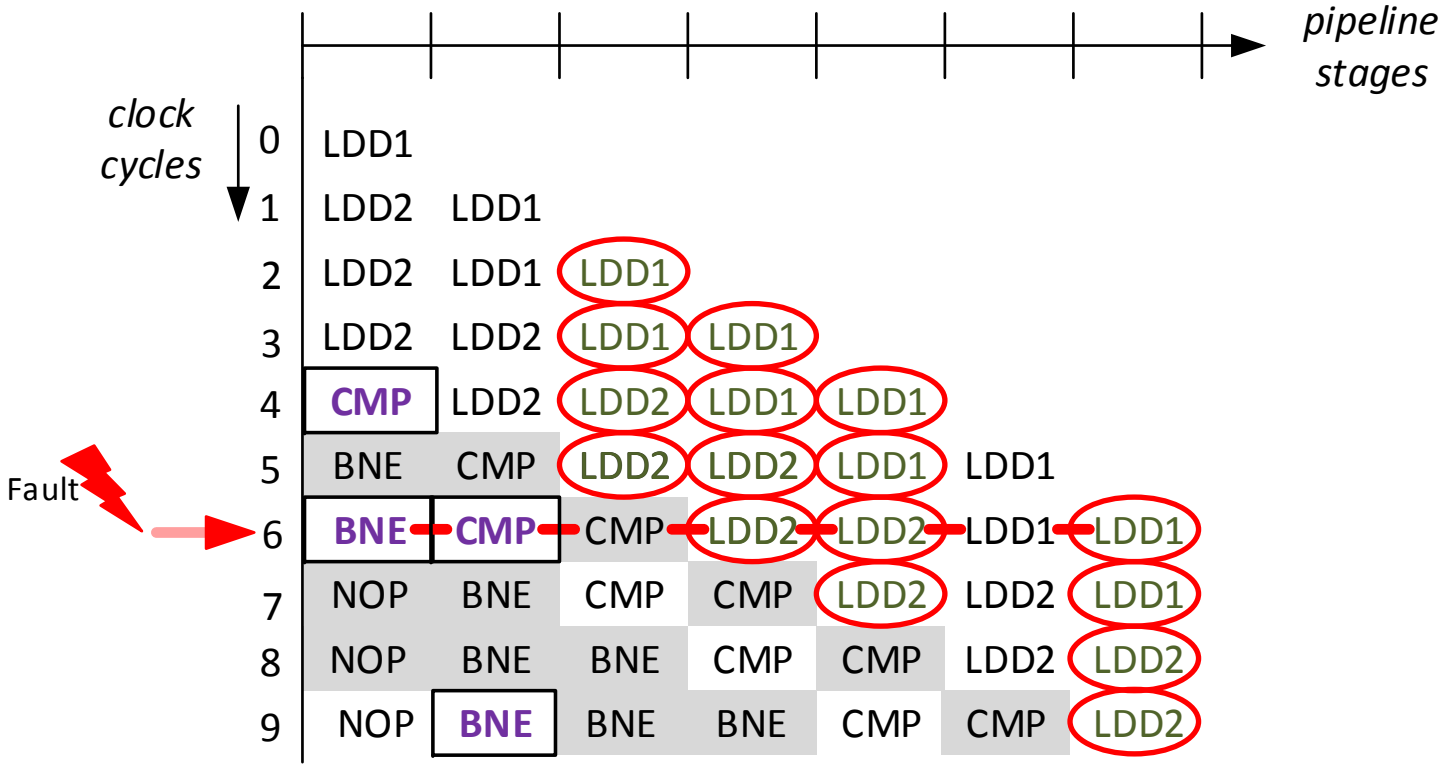


DFIA on LED

```
;Unprotected code for AddRoundConstant
LDD [%fp + -56], %g2   ;LDD1
XOR %o4, %g2, %g2
XOR %o5, %g3, %g3
STD %g2, [%fp + -56]
```

```
;Instruction Duplication on LDD1
LDD       [%fp-56], %g2   ;LDD1
LDD       [%fp-56], %g4   ;LDD2
CMP       %g2, %g4
BNE       .error
```

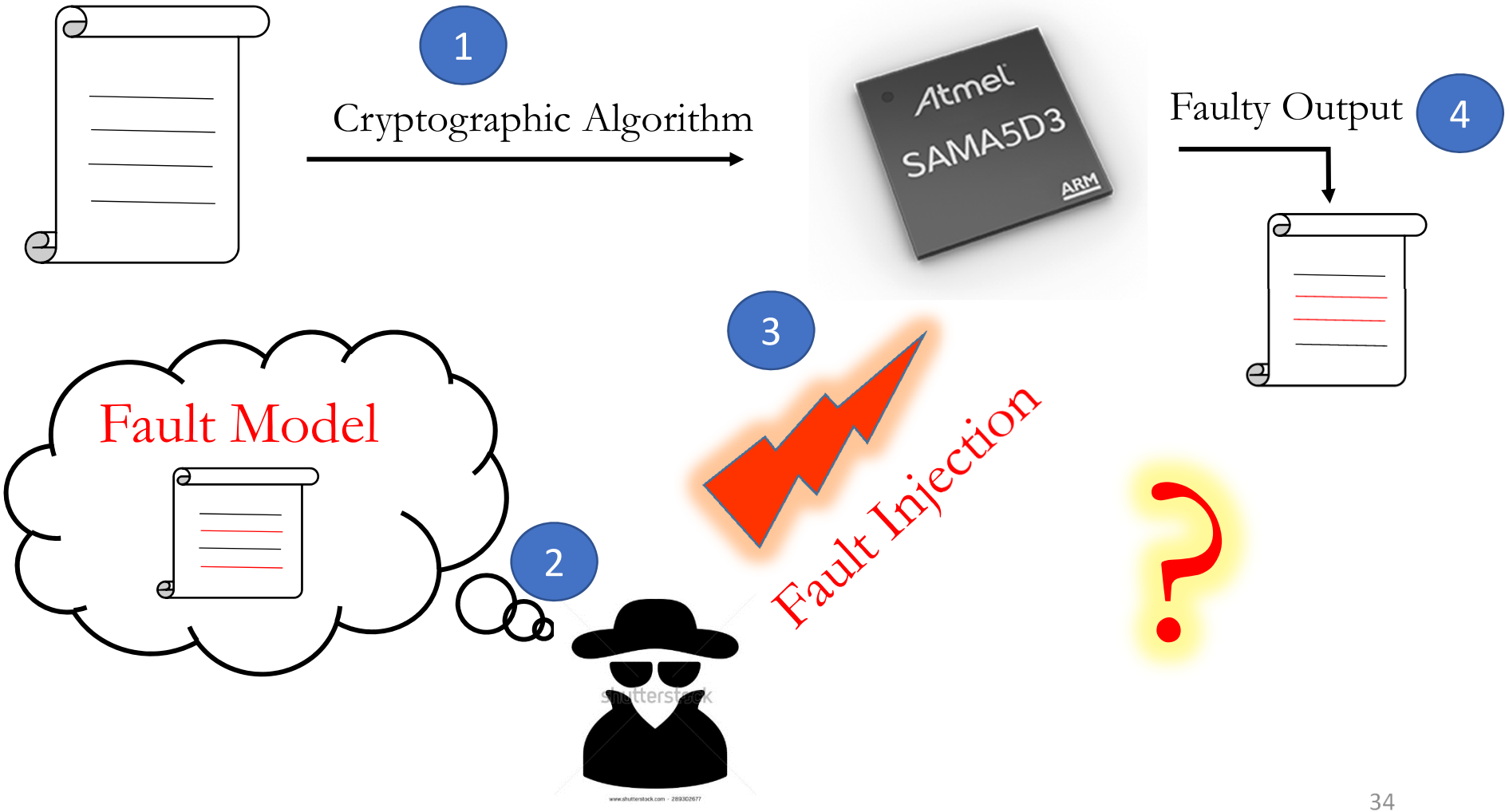
Pipeline behavior of ID for LED



Glitch width(ns)	Effected Inst. in Pipeline	Observed Faulty Behavior
31.2-33.6	LDD1, W	Fault in %g2
	LDD2, M	Fault in %g4
	BNE, F	BNE to OR

Fault Attack on Microprocessors

- Traditional Fault Attacks



Fault Attack on Microprocessors

- Traditional Fault Attacks