Effects of synchronous clock glitch on the security of an integrated circuit

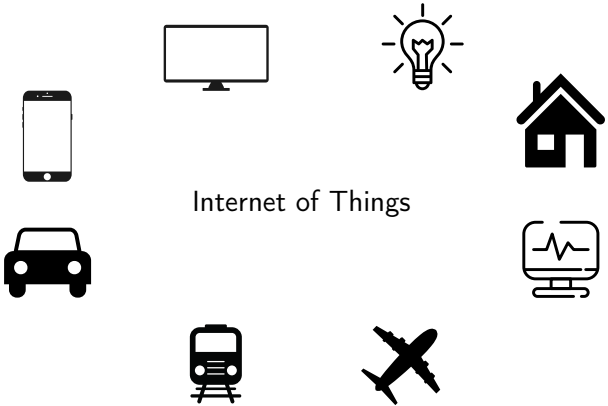
Amélie Marotta

PhD Defense
23/06/2025

Examiners:
Vincent Beroulle
Maria Méndez-Real
Jessy Clédière
Jean-Max Dutertre

PhD Advisors:
Olivier Sentieys (director)
Ronan Lashermes (co-director)
Rachid Dafali
Guillaume Bouffard

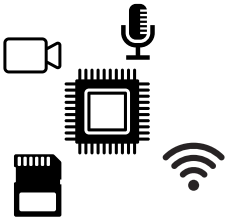
Introduction



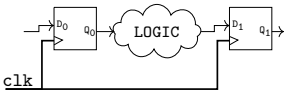
An example

```
if compare(val1, val2) == True:
  val1 = val1 + 1
```

⟶ software

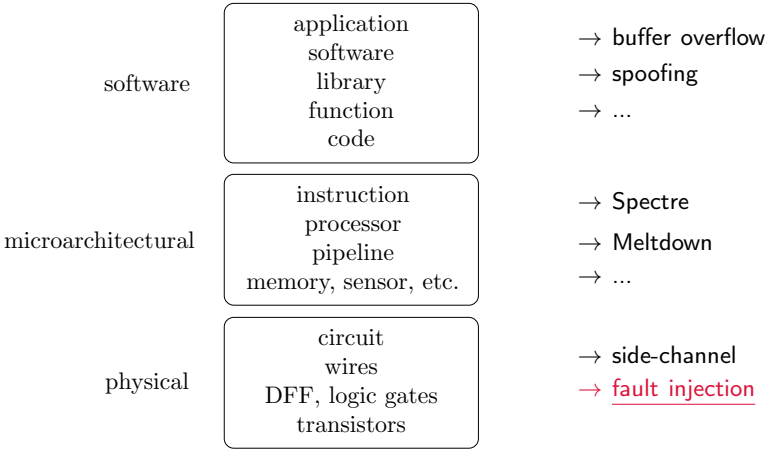


⟶ microarchitectural



⟶ physical

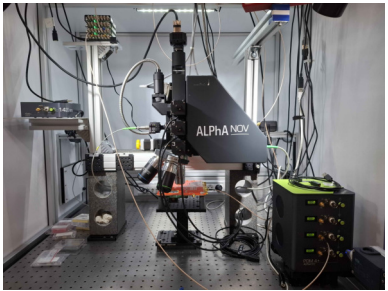
Vulnerabilities and exploitation methods

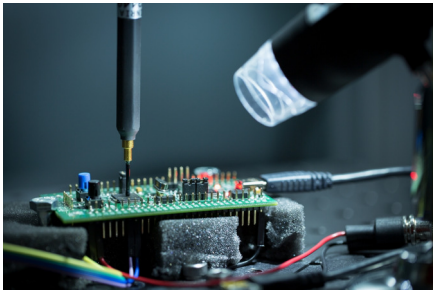


Fault Injection

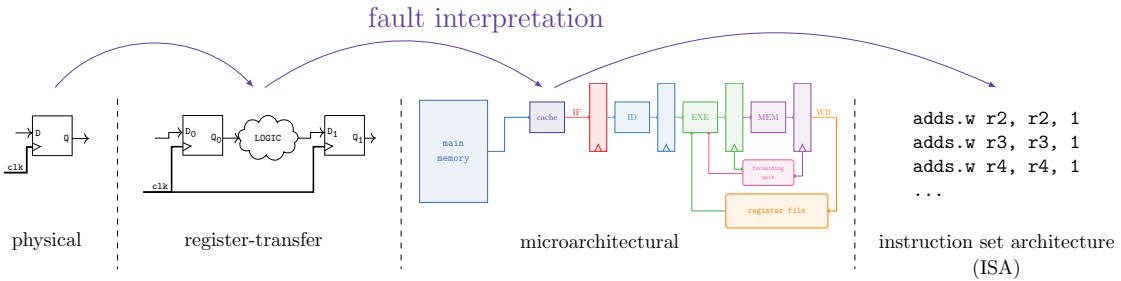
Several fault injection (FI) methods exist:

- → voltage glitch
- → clock glitch
- → laser FI
- → electromagnetic FI

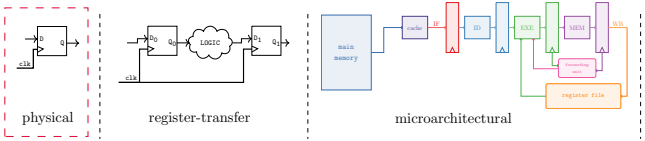




Fault characterization

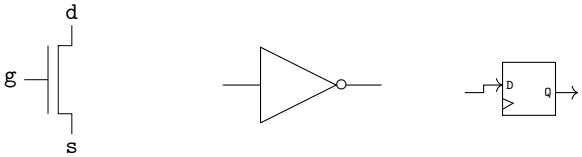


Physical level




```
adds.w r2, r2, 1
adds.w r3, r3, 1
adds.w r4, r4, 1
...
```

Impacted elements:

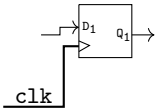


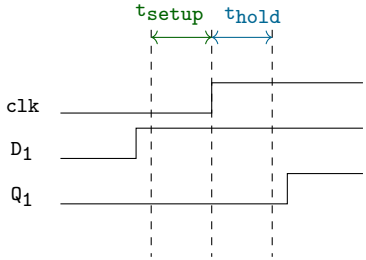
transistors | logic gates | flip-flops

→ Possible fault effects: switching logic gates output, preventing DFF sampling, etc.

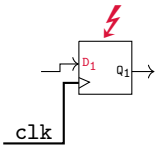
Physical level: the flip-flop

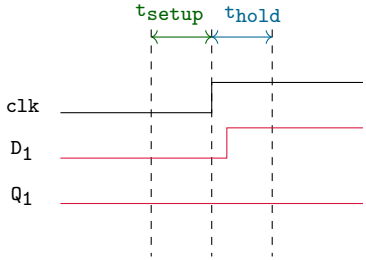
Normal behaviour



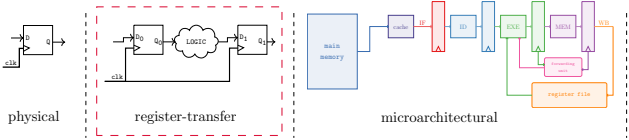


Faulted behaviour

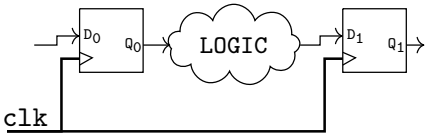




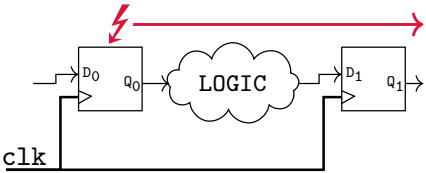
Register-transfer level



Normal behaviour

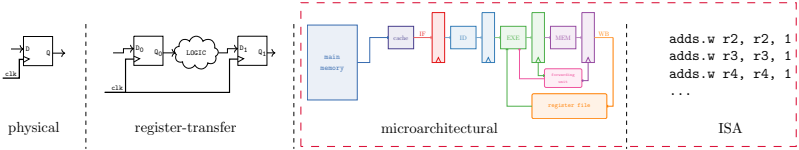


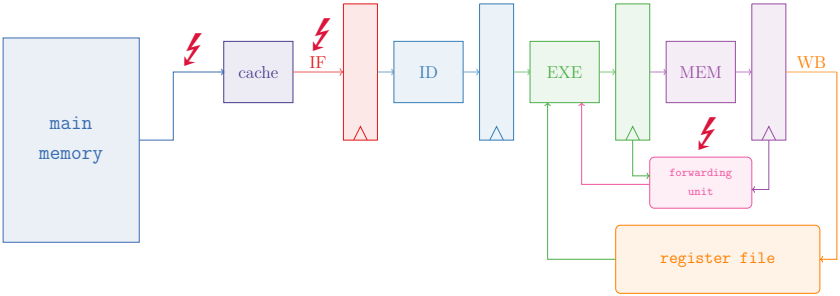
Faulted behaviour



→ Possible fault effects: bit flip propagation, etc.

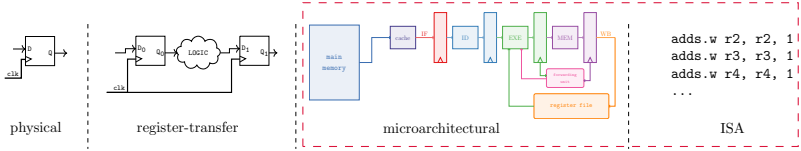
Microarchitectural level

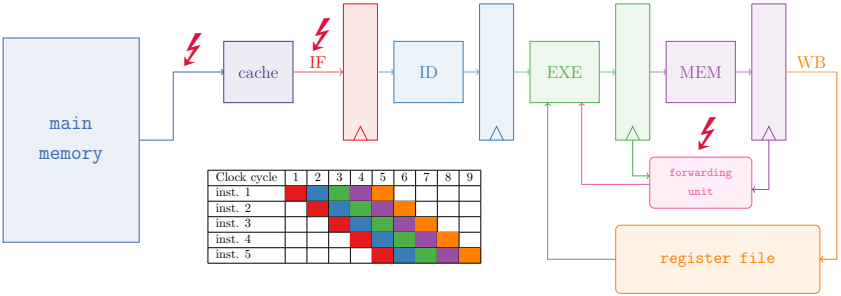




→ Possible fault effects: instruction skip/repeat/modification, alteration of data/instruction transfer, etc.

Microarchitectural level

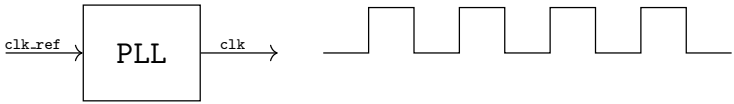




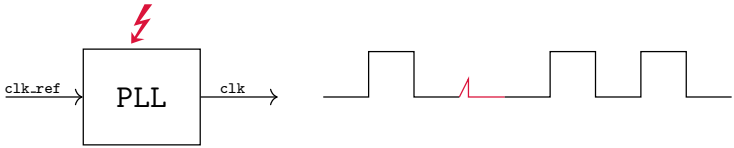
→ Possible fault effects: instruction skip/repeat/modification, alteration of data/instruction transfer, etc.

A effect in particular: EM impact on the Phase-Locked Loop[1] (PLL)

Normal behaviour

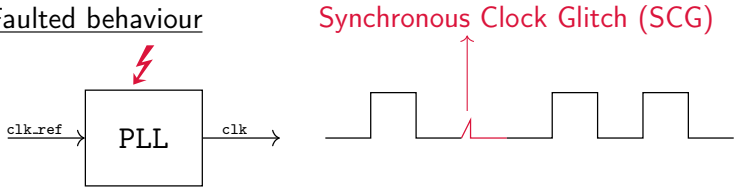


Faulted behaviour



[1] Ludovic Claudepierre and Philippe Besnier, Microcontroller Sensitivity to Fault-Injection Induced by Near-Field Electromagnetic Interference.

An effect in particular: EM impact on the Phase-Locked Loop (PLL)



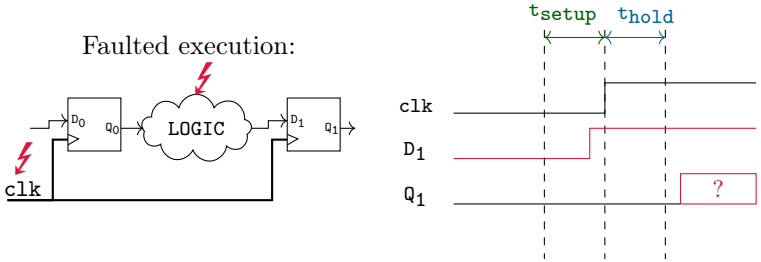
→ effect at microarchitectural level: instruction skip or repeat

→ effect at register-transfer and physical levels?

↪ explained by state-of-the-art physical fault models?

EM on the PLL: does the Timing Fault Model[2] apply?

→ Main fault mechanism: timing violation of t_{setup} by advancing a clock cycle or extending the execution time of D_1



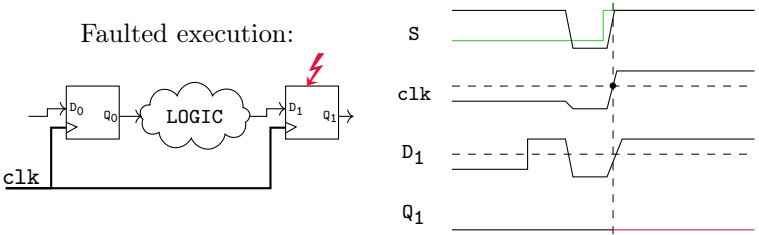
→ Does the TFM apply?

× no because the SCG does not cause timing variations for either the clock or D_1

[2] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria, Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES.

EM on the PLL: does the Sampling Fault Model[3] apply?

→ Main fault mechanism: race condition between the clock and D_1 by altering all signals



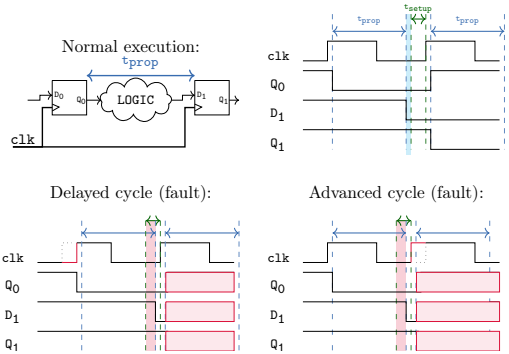
→ Does the SFM apply?

× no because the SCG only affects the clock

[3] Mathieu Dumont, Mathieu Lisart, and Philippe Maurine, Modeling and Simulating Electromagnetic Fault Injection.

EM on the PLL: does the Nabhan's Fault Model[4] apply?

→ Main fault mechanism: timing violation of t_{setup} by shifting a clock cycle or creating additional clock cycles



→ Does the NFM apply?

× no because the SCG remains synchronous and affects a single clock cycle

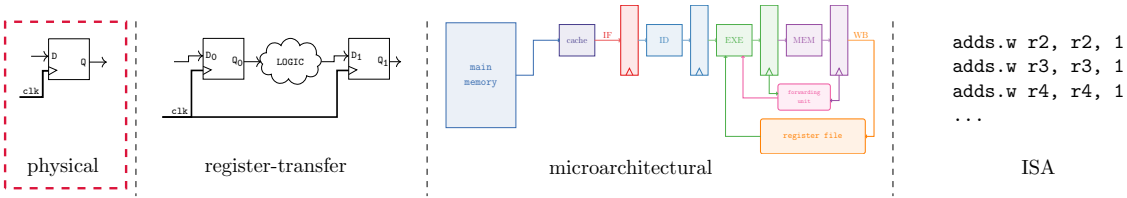
[4] Roukoz Nabhan, Jean-Max Dutertre, Jean-Baptiste Rigaud, Jean-Luc Danger, and Laurent Sauvage, A Tale of Two Models: Discussing the Timing and Sampling EM Fault Injection Models.

Thesis objective: Understanding the SCG

Contributions during this PhD:

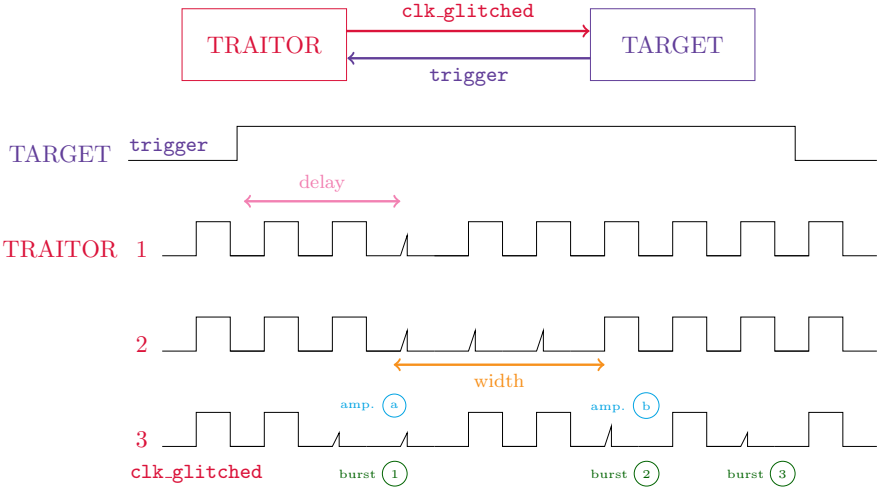
- → characterizing and modeling the SCG at physical level (work published at COSADE'24)
- → extending the model at RTL
- → gathering clues on the SCG impact at microarchitectural/ISA level and comparing it to the state-of-the art

Overview



1. Physical characterization of the SCG
   - ↪ TRAITOR
   - ↪ Experimental set-up
   - ↪ Hypotheses verification
2. Microarchitectural characterization of the SCG

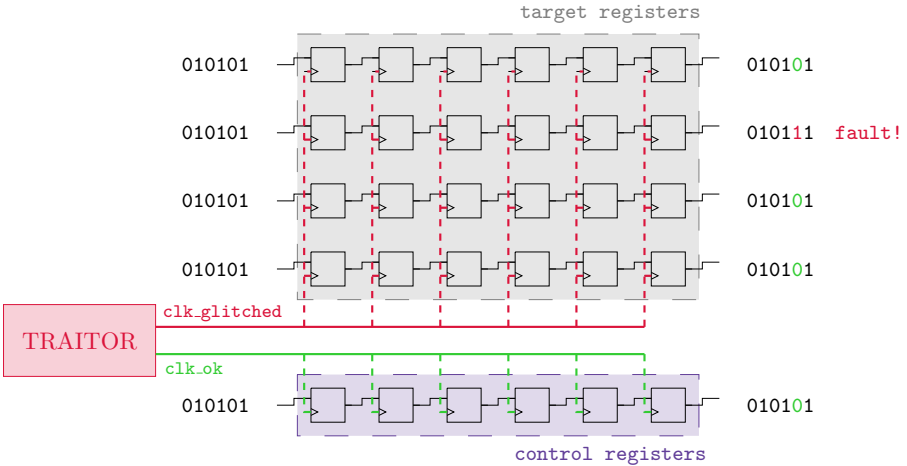
TRAITOR[5]



Three examples of clock signals generated by TRAITOR illustrating its possibilities.

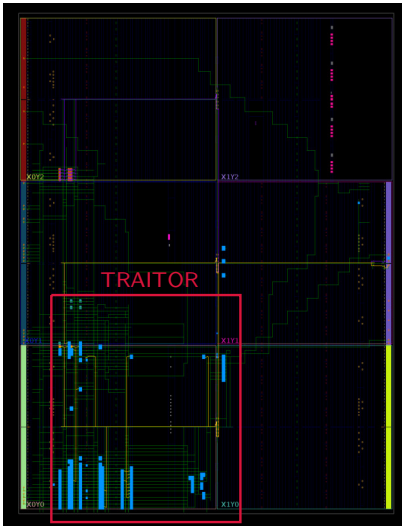
[5] Ludovic Claudepierre, Pierre-Yves Péneau, Damien Hardy, and Erven Rohou, TRAITOR: A Low-Cost Evaluation Platform for Multifault Injection.

Device Under Test (DUT)

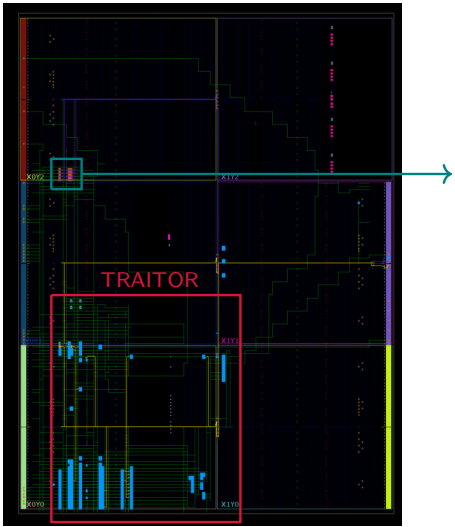


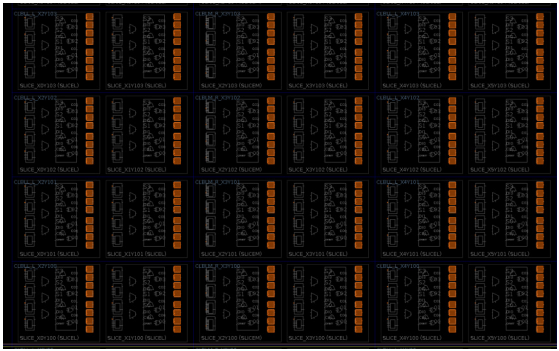
Floorplan

Floorplan



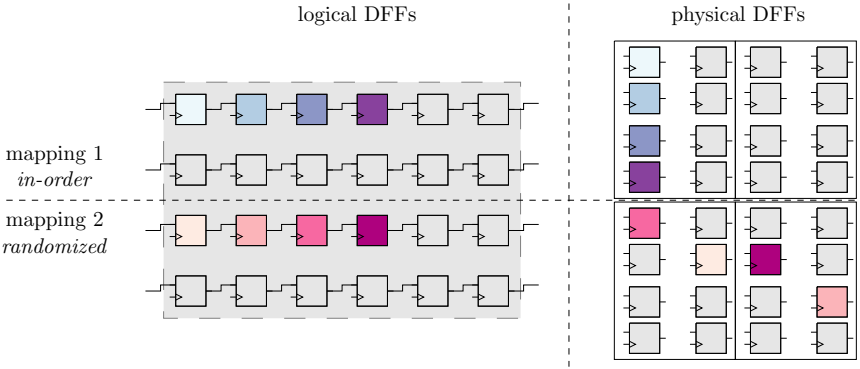
Floorplan





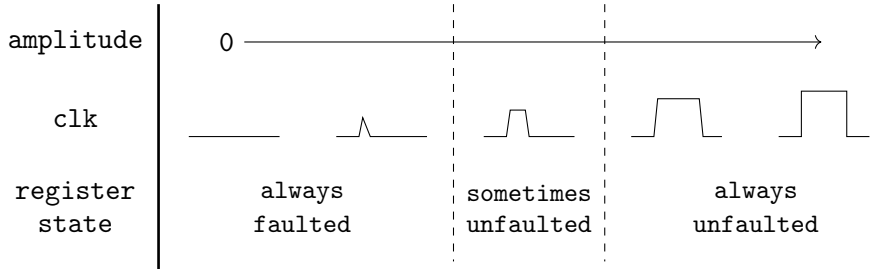
target DFFs

Logical and physical, in-order and randomized

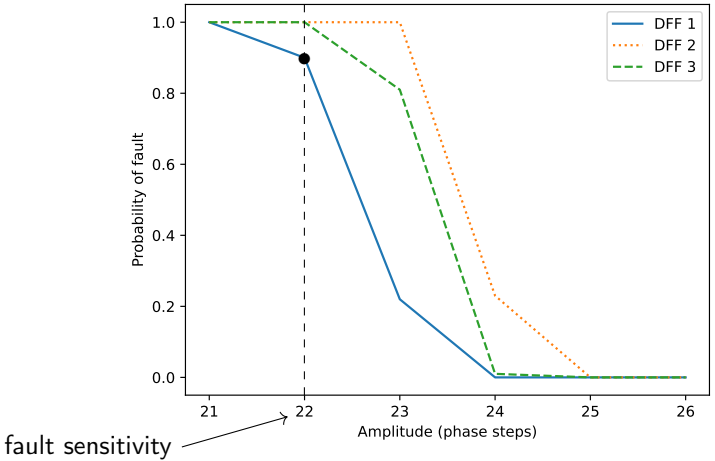


Hypotheses

Hypothesis 1 (Energy Threshold) *For a DFF to correctly sample a clock's rising edge, the clock signal must meet a certain energy threshold, combination of voltage amplitude and width thresholds.*

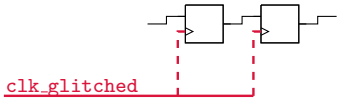


Behaviour of three selected DFF



Transitions phases of three target physical DFFs chosen since they exhibit different characteristics.

Simulation set-up



→ SPICE simulation

→ 28nm DFF

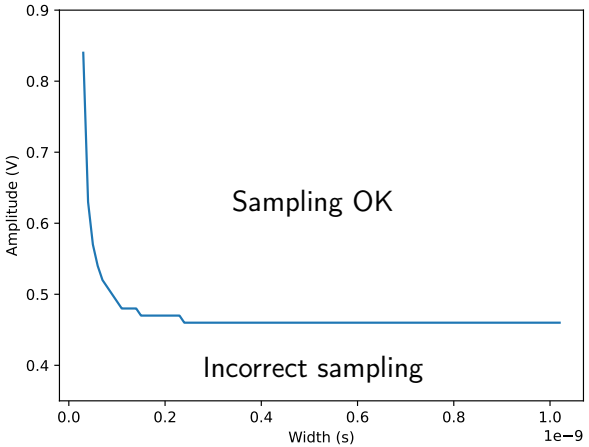
↪ not the exact same as the Artix-7 DFF

↪ designed for similar technology so should behave the same way

→ focus on the state change of the first DFF

Goal: estimate the impact of the voltage and width of the CSCG

Simulation results

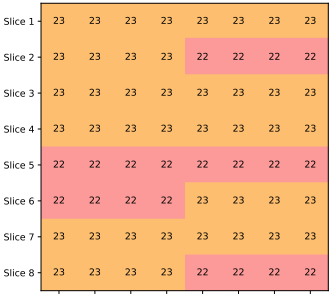


Hypotheses

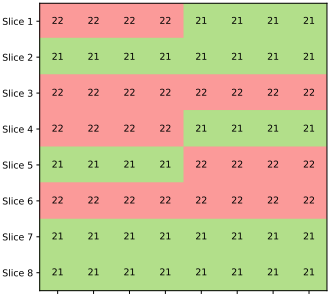
Hypothesis 2 (Fault Sensitivity Dependency on Intrinsic Properties) *The fault sensitivity of a DFF depends on its intrinsic properties, such as clock routing up to the DFF among others.*

$\longrightarrow$ Only clock routing?
$\hookrightarrow$ same DUT on two Artix-7 FPGAs

Only clock routing?



(a) Color coded fault sensitivities of the first 64 registers on mapping 1 *in-order* on FPGA 1.



(b) Color coded fault sensitivities of the first 64 registers on mapping 1 *in-order* on FPGA 2.

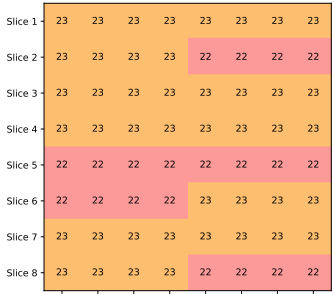
↪ Comparing fault sensitivities on two FPGAs.

Hypotheses

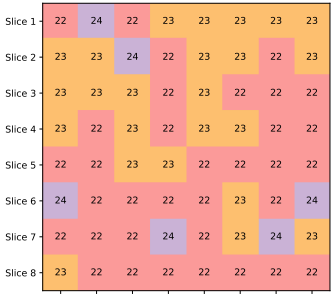
Hypothesis 2 (Fault Sensitivity Dependency on Intrinsic Properties) *The fault sensitivity of a DFF depends on its intrinsic properties, such as process variability and clock routing up to the DFF among others.*

⟶ Only intrinsic properties?
↪ same FPGA, different mappings

Only intrinsic properties?



(a) Color coded fault sensitivities of the first 64 registers on mapping 1 *in-order* on FPGA 1.



(b) Color coded fault sensitivities of the first 64 registers on mapping 2 *randomized* on FPGA 1.

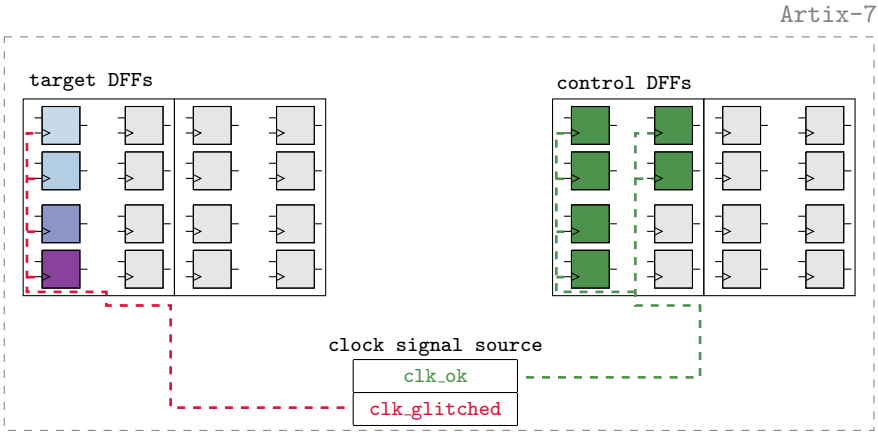
↪ Comparing fault sensitivities between physical DFFs for different mappings.

Hypotheses

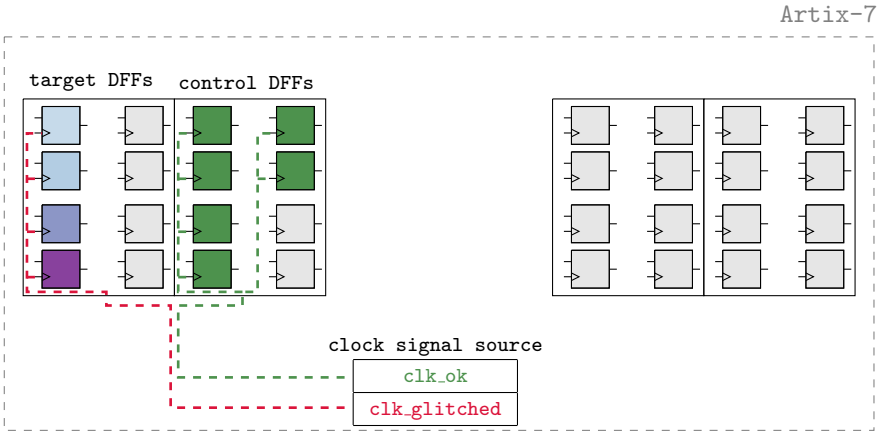
Hypothesis 3 (Fault Sensitivity Dependency on Extrinsic Properties) *The fault sensitivity of a DFF may also be affected by extrinsic factors, such as the activity in neighboring wires (including routing between DFFs and the routing of the clock tree).*

$\longrightarrow$ Impact of clock wires
$\hookrightarrow$ forced adjacent clock paths

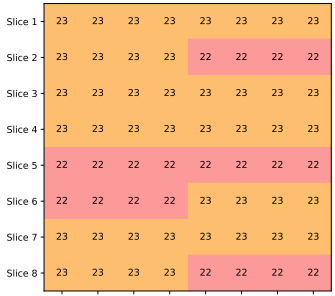
Impact of clock wires



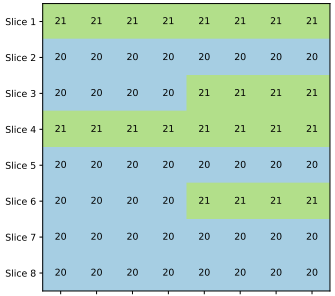
Impact of clock wires



Impact of clock wires



(a) Color coded fault sensitivities of the first 64 registers on mapping 1 *in-order* on FPGA 1.



(b) Color-coded fault sensitivities of the first 64 registers on mapping 1 *in-order* with a forced adjacent path for the clock on FPGA 1

↪ Comparing fault sensitivities between physical DFFs for different clock routing.

The Energy Threshold Fault Model

Hypothesis 1 (Energy Threshold) *For a DFF to correctly sample a clock's rising edge, the clock signal must meet a certain energy threshold, combination of voltage amplitude and width thresholds.*

The Energy Threshold Fault Model

Hypothesis 1 (Energy Threshold) *For a DFF to correctly sample a clock's rising edge, the clock signal must meet a certain energy threshold, combination of voltage amplitude and width thresholds.*

Hypothesis 2 (Fault Sensitivity Dependency on Intrinsic Properties) *The fault sensitivity of a DFF depends on its intrinsic properties, such as process variability and clock routing up to the DFF among others.*

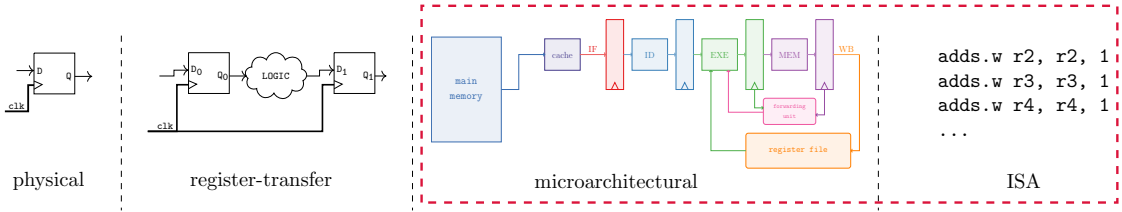
The Energy Threshold Fault Model

Hypothesis 1 (Energy Threshold) *For a DFF to correctly sample a clock's rising edge, the clock signal must meet a certain energy threshold, combination of voltage amplitude and width thresholds.*

Hypothesis 2 (Fault Sensitivity Dependency on Intrinsic Properties) *The fault sensitivity of a DFF depends on its intrinsic properties, such as process variability and clock routing up to the DFF among others.*

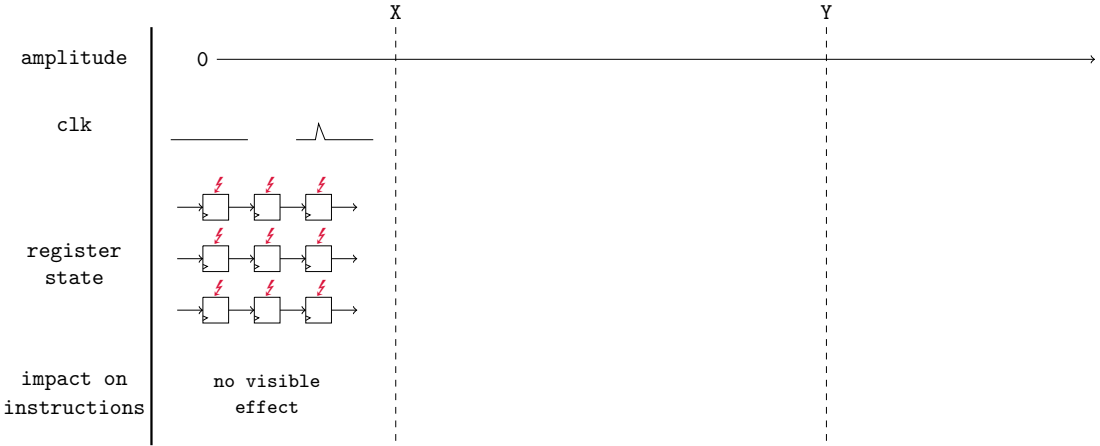
Hypothesis 3 (Fault Sensitivity Dependency on Extrinsic Properties) *The fault sensitivity of a DFF may also be affected by extrinsic factors, such as the activity in neighboring wires (including routing between DFFs and the routing of the clock tree).*

Overview

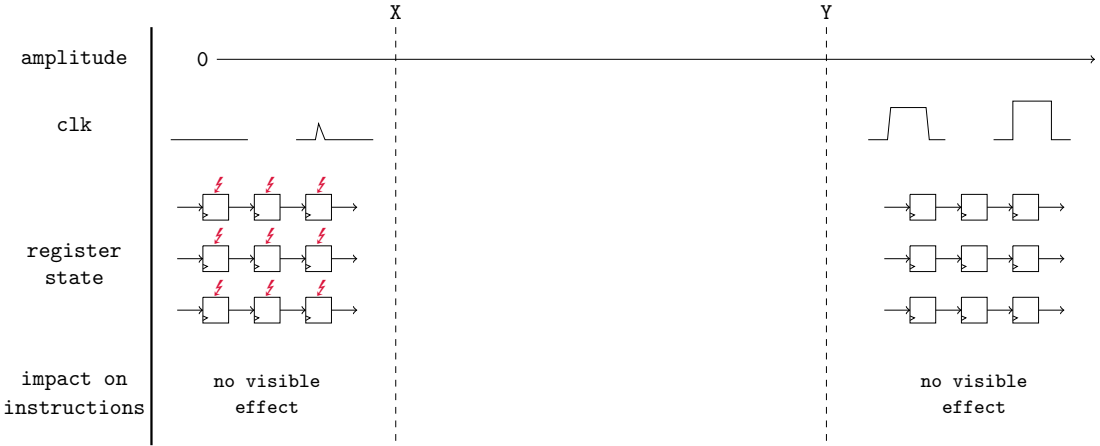


1. Physical characterization of the SCG
2. Microarchitectural characterization of the SCG
   - ↪ Preliminary fault model
   - ↪ Experimental set-up
   - ↪ Hypothesis verification

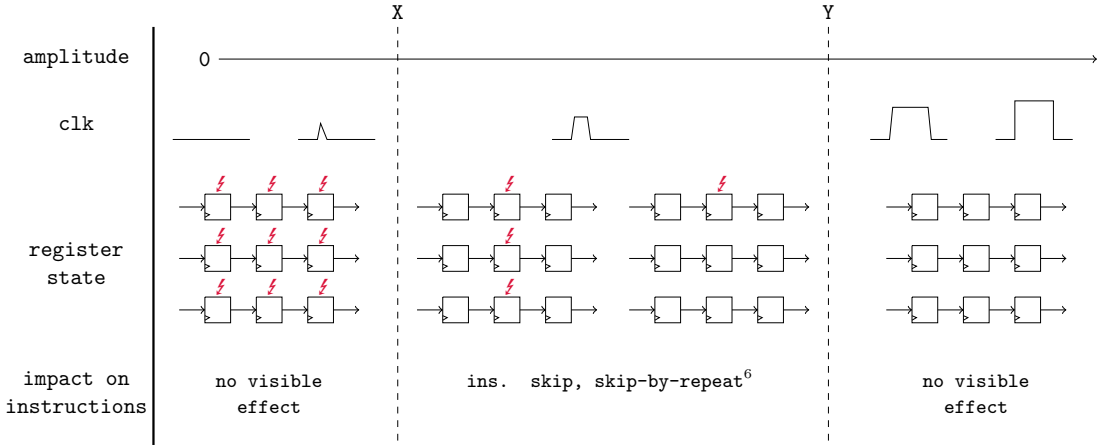
Extension to microarchitectural level - preliminary model



Extension to microarchitectural level - preliminary model



Extension to microarchitectural level - preliminary model

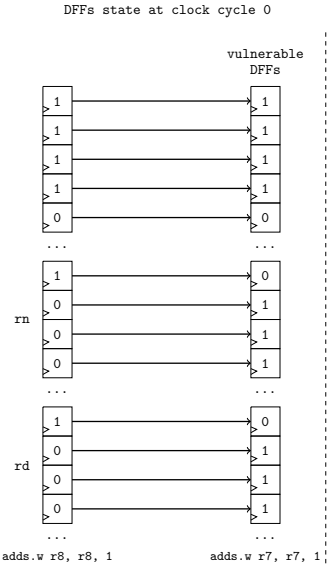


[6] Ludovic Claudepierre, Pierre-Yves Péneau, Damien Hardy, and Erven Rohou, TRAITOR: A Low-Cost Evaluation Platform for Multifault Injection

Extension to microarchitectural level - preliminary model

encoding for
adds.w rd, rn, imm

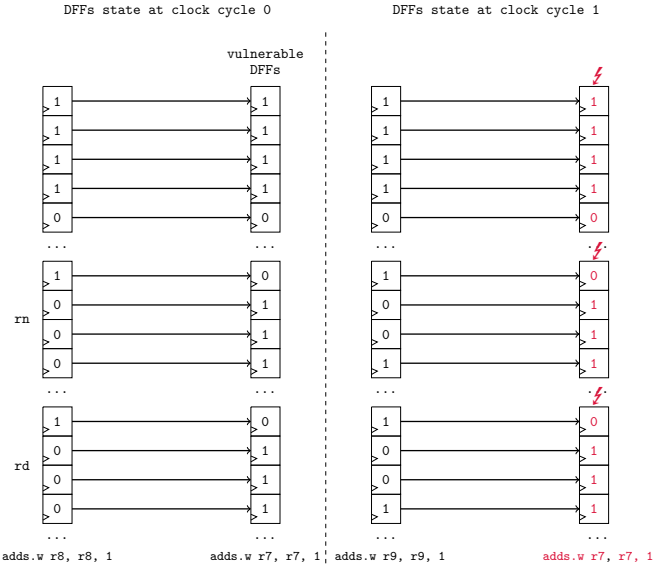
31	1
30	1
29	1
28	1
27	0
26	0
25	0
24	1
23	0
22	0
21	0
20	1
19	rn
18	
17	
16	
15	0
14	imm
13	
12	
11	rd
10	
9	
8	
7	imm
6	
5	
4	
3	
2	
1	
0	



Extension to microarchitectural level - preliminary model

encoding for
`adds.w rd, rn, imm`

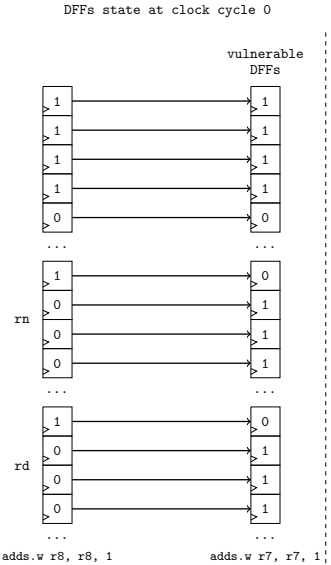
31	1
30	1
29	1
28	1
27	0
26	0
25	0
24	1
23	0
22	0
21	0
20	1
19	rn
18	
17	
16	
15	0
14	imm
13	
12	
11	rd
10	
9	
8	
7	imm
6	
5	
4	
3	
2	
1	
0	



Extension to microarchitectural level - preliminary model

encoding for
`adds.w rd, rn, imm`

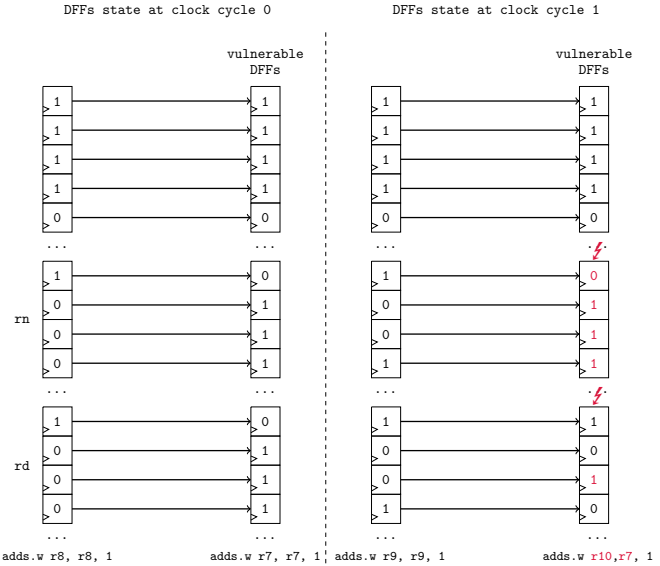
31	1
30	1
29	1
28	1
27	0
26	0
25	0
24	1
23	0
22	0
21	0
20	1
19	rn
18	
17	
16	
15	0
14	imm
13	
12	
11	rd
10	
9	
8	
7	imm
6	
5	
4	
3	
2	
1	
0	



Extension to microarchitectural level - preliminary model

encoding for
adds.w rd, rn, imm

31	1
30	1
29	1
28	1
27	0
26	0
25	0
24	1
23	0
22	0
21	0
20	1
19	rn
18	
17	
16	
15	0
14	imm
13	
12	
11	rd
10	
9	
8	
7	imm
6	
5	
4	
3	
2	
1	
0	

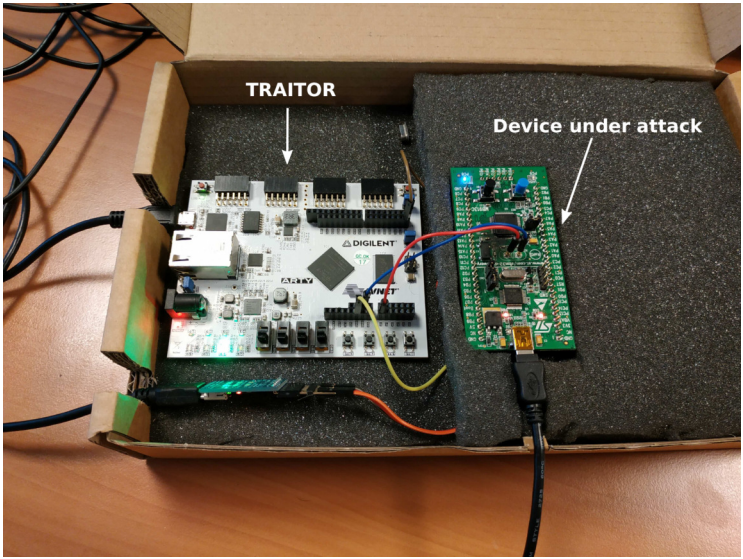


Extension to microarchitectural level - preliminary model

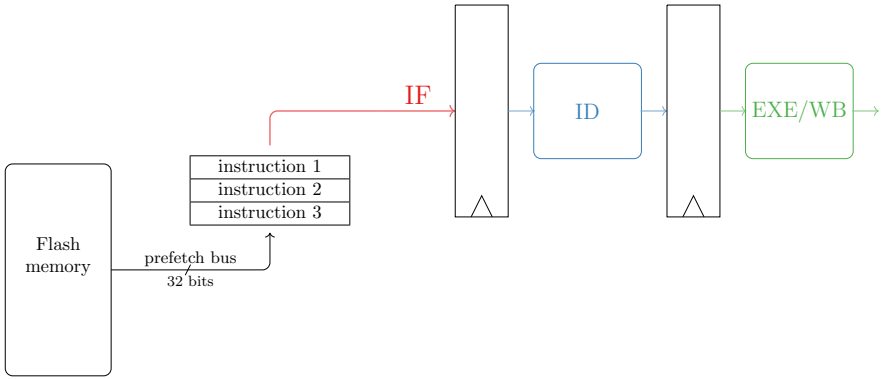
From the ETFM and the state of the art
$\hookrightarrow$ preliminary microarchitectural fault model

1. Instructions are affected by the fault
   $\hookrightarrow$ instruction transfers (between caches, inside the pipeline) vulnerable
2. $\neq$ amplitudes $\Longrightarrow$ $\neq$ ISA level effects
   $\hookrightarrow$ instruction skip, repeat, modification
3. Fault impact is only visible on flipping bits

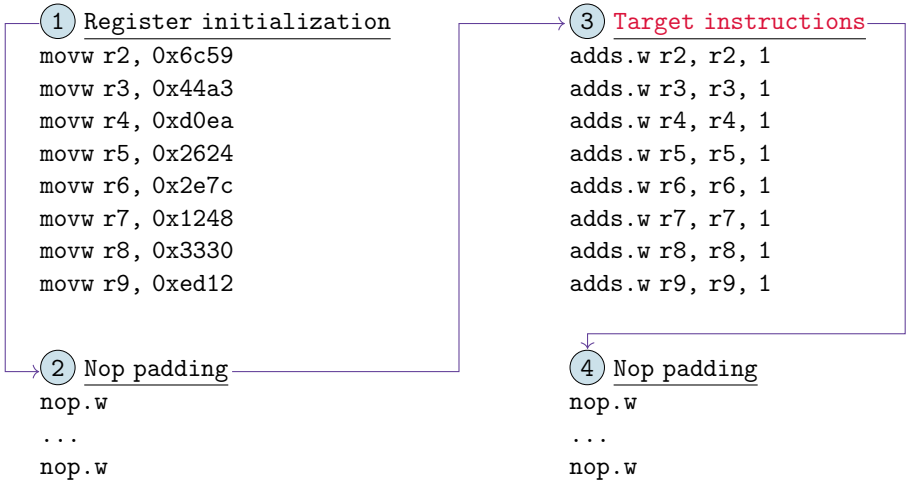
Fault injection setup



Target description



Target code



→ All instructions are 32-bit long, aligned

Fault models terminology

"instruction `adds.w r2, r2, 1`"
"instruction that modifies `r2`" $\longrightarrow$ too long !

Naming convention of instruction:

`adds.w r2, r2, 1`	$\longrightarrow$ `ins. r2`
`adds.w r3, r3, 1`	$\longrightarrow$ `ins. r3`
`adds.w r4, r4, 1`	$\longrightarrow$ `ins. r4`
`adds.w r5, r5, 1`	$\longrightarrow$ `ins. r5`
`adds.w r6, r6, 1`	$\longrightarrow$ `ins. r6`
`adds.w r7, r7, 1`	$\longrightarrow$ `ins. r7`
`adds.w r8, r8, 1`	$\longrightarrow$ `ins. r8`
`adds.w r9, r9, 1`	$\longrightarrow$ `ins. r9`

Fault injection protocol

TRAITOR parameters:

→ amplitude ranges from 370 to 430

→ delay ranges from 30 to 50

→ width is constant at 1 (a single clock cycle is modified)

For each experiment (50 for each different set of parameters):

→ the target is reset

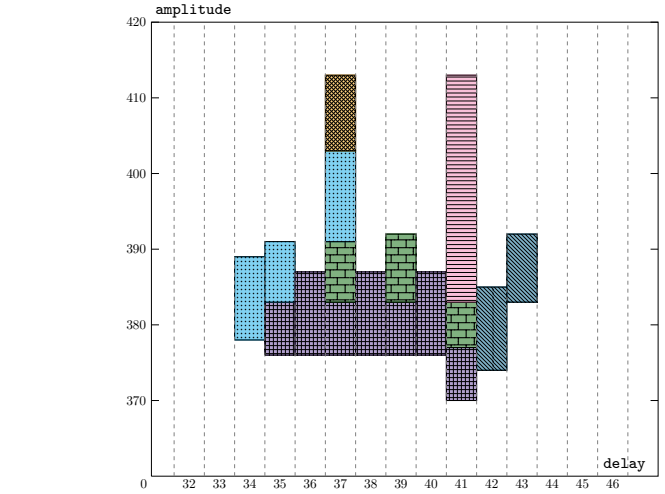
→ the execution stops at the end of the second `nop.w` set of instructions (breakpoint)

→ the value of `r0-r12, sp, lr, pc`, etc., are retrieved

→ in case of interrupt that escalates into Hardfault, the type of interrupt (`xPSR`) as well as `pc` and `lr` are retrieved

⇒ Dominant fault impacts on the target code

Finding dominant fault models



FM 1: `ins. rx` is skipped

FM 2: `ins. rx` is skipped
`ins. rx-1` is repeated

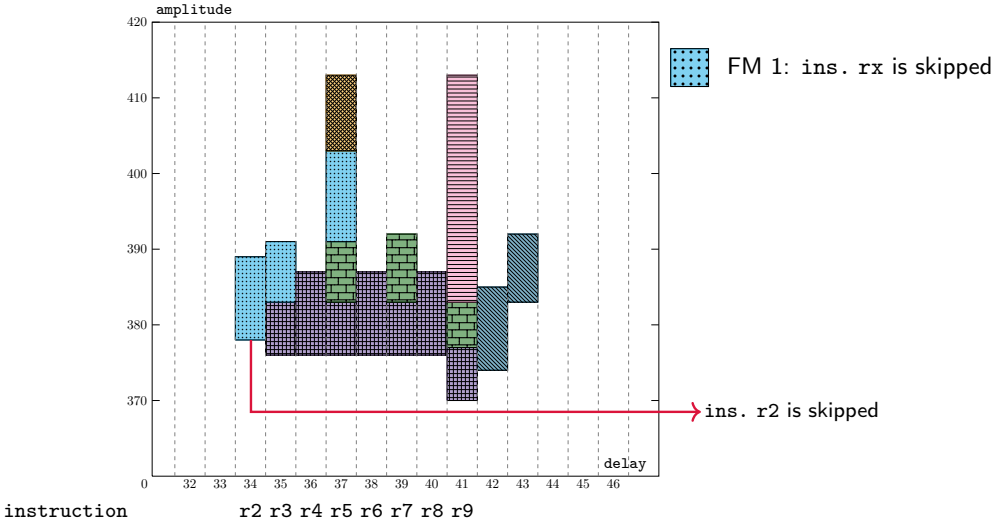
FM 3: `ins. rx` is skipped
`ins. rx-2` is repeated

FM 4: `ins. rx` is skipped
`ins. rx+4` is repeated

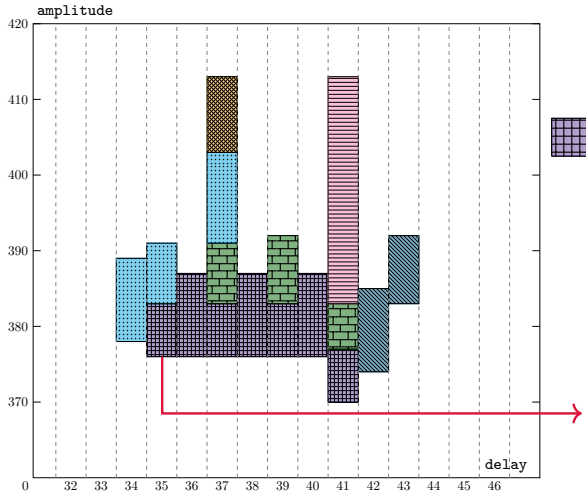
FM 5: `ins. rx` is skipped
`ins. rx-4` is repeated

FM 6: `ins. r8` is repeated

Finding dominant fault models



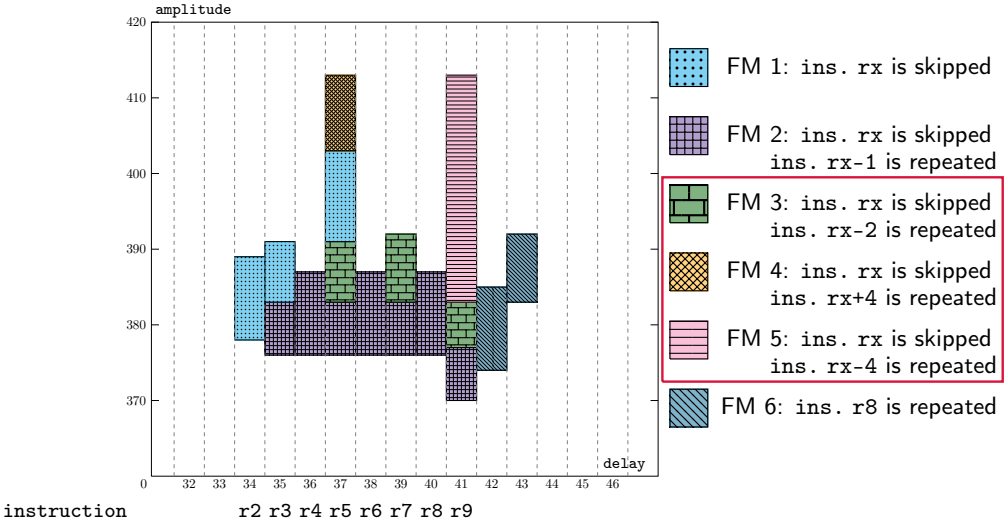
Finding dominant fault models



FM 2: `ins. rx` is skipped
`ins. rx-1` is repeated

`ins. r3` is skipped and `ins. r2` is repeated

Finding dominant fault models



Finding dominant fault models

at delay 37:

FM2
`adds.w r5, r5, 1` is skipped
`adds.w r4, r4, 1` is repeated

FM3
`adds.w r5, r5, 1` is skipped
`adds.w r3, r3, 1` is executed

$\rightarrow$ instruction modification ✓

Finding dominant fault models

at delay 37:

FM2
`adds.w r5, r5, 1` is skipped
`adds.w r4, r4, 1` is repeated

FM3
`adds.w r5, r5, 1` is skipped
`adds.w r3, r3, 1` is executed

→ instruction modification ✓

... does not match hypothesis (3) Fault impact is only visible on flipping bits

Finding dominant fault models

at delay 37:

FM2
`adds.w r5, r5, 1` is skipped
`adds.w r4, r4, 1` is repeated

FM3
`adds.w r5, r5, 1` is skipped
`adds.w r3, r3, 1` is executed

→ instruction modification ✓

... does not match hypothesis (3) Fault impact is only visible on flipping bits

reg	encoding
r5	0101
r4	0100

Finding dominant fault models

at delay 37:

FM2
`adds.w r5, r5, 1` is skipped
`adds.w r4, r4, 1` is repeated

FM3
`adds.w r5, r5, 1` is skipped
`adds.w r3, r3, 1` is executed

→ instruction modification ✓

... does not match hypothesis (3) Fault impact is only visible on flipping bits

reg	encoding
r5	0101
r4	0100
r3	0011

Finding dominant fault models

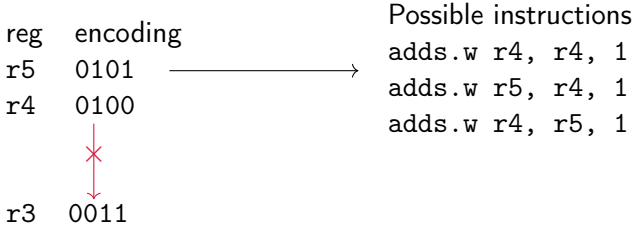
at delay 37:

FM2
`adds.w r5, r5, 1` is skipped
`adds.w r4, r4, 1` is repeated

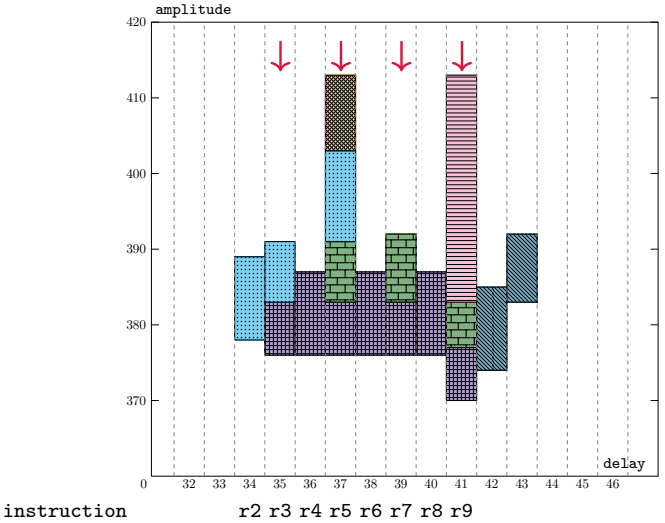
FM3
`adds.w r5, r5, 1` is skipped
`adds.w r3, r3, 1` is executed

→ instruction modification ✓

... does not match hypothesis (3) Fault impact is only visible on flipping bits



Finding dominant fault models



FM 1: `ins. rx` is skipped

FM 2: `ins. rx` is skipped
`ins. rx-1` is repeated

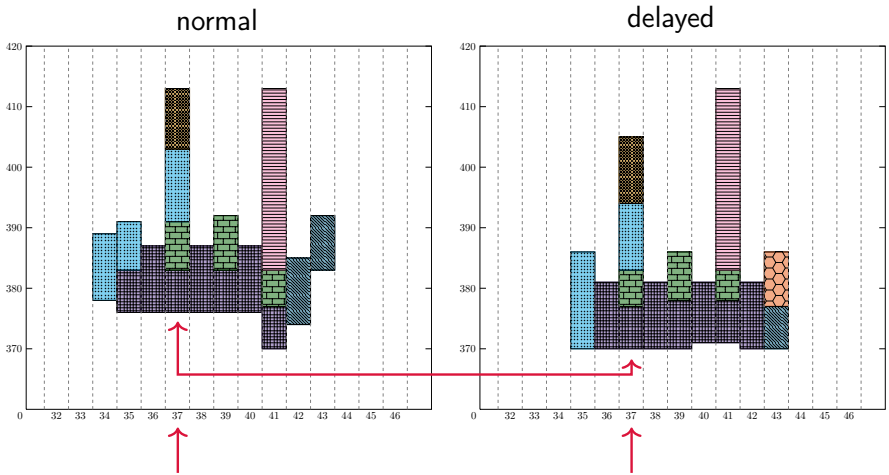
FM 3: `ins. rx` is skipped
`ins. rx-2` is repeated

FM 4: `ins. rx` is skipped
`ins. rx+4` is repeated

FM 5: `ins. rx` is skipped
`ins. rx-4` is repeated

FM 6: `ins. r8` is repeated

Fault influence: delay?



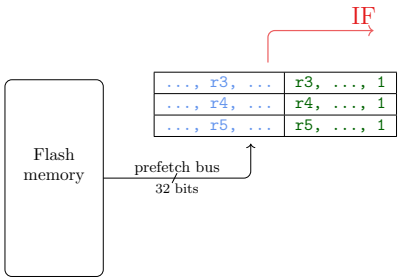
$\Rightarrow$ fault effects are tied to the delay of the fault

Vulnerable processor part

`adds.w rd, rn, 1`

31	30	29	28	27	...	19	18	17	16	...	11	10	9	8	...	3	2	1	0
1	1	1	1	0	...	rn				...	rd				...	0	0	0	1

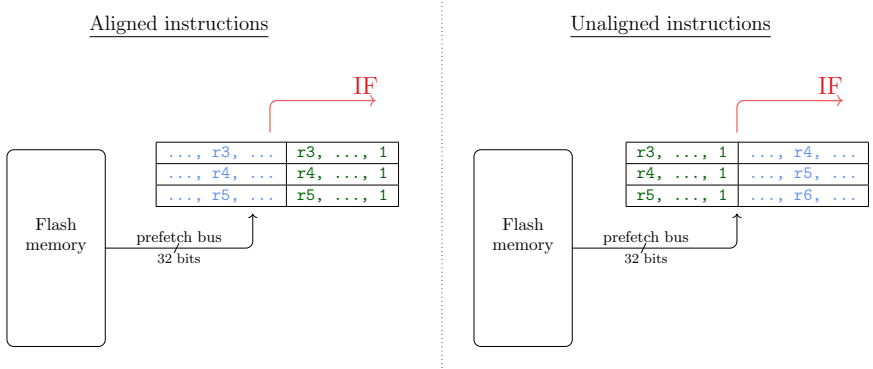
Aligned instructions



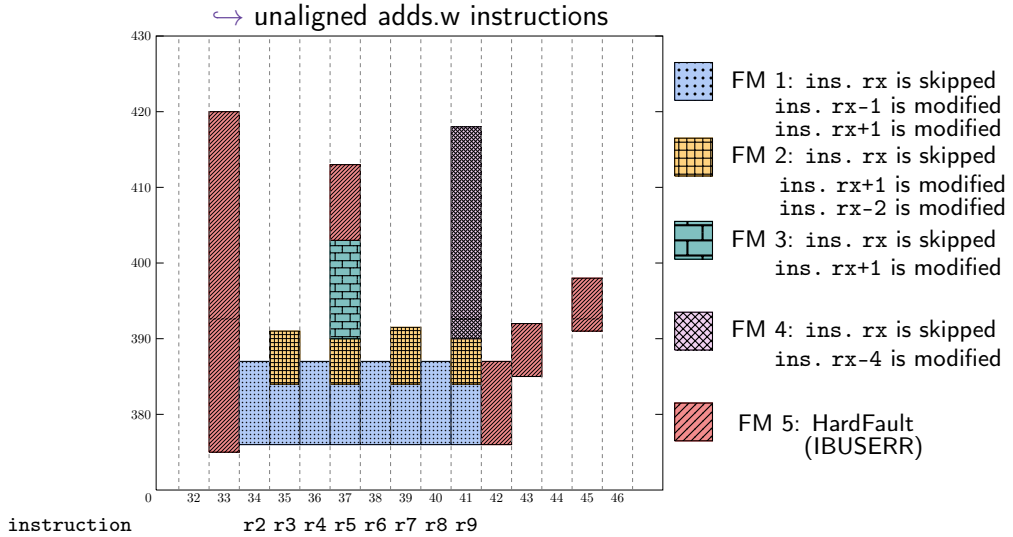
Vulnerable processor part

`adds.w rd, rn, 1`

31	30	29	28	27	...	19	18	17	16	...	11	10	9	8	...	3	2	1	0
1	1	1	1	0	...	rn				...	rd				...	0	0	0	1

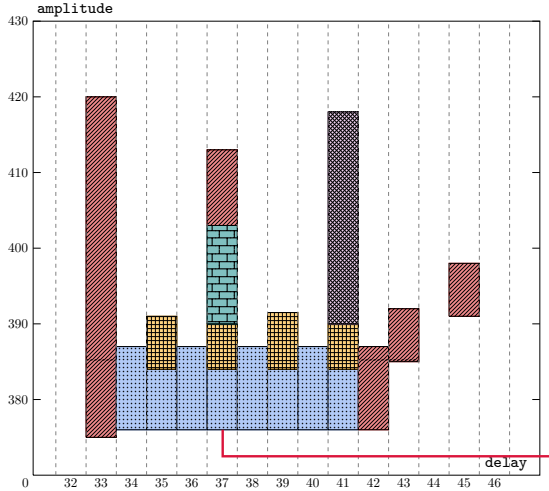


Vulnerable processor part



Vulnerable processor part

↪ unaligned adds.w instructions

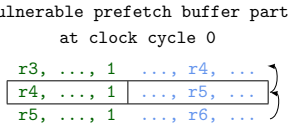


FM 1: `ins. rx` is skipped
`ins. rx-1` is modified
`ins. rx+1` is modified

`ins. r5` is skipped
`r4 = r5+1`
`r6 = r5+1`

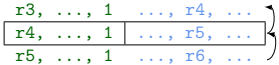
`instruction` `r2 r3 r4 r5 r6 r7 r8 r9`

Vulnerable processor part

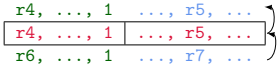


Vulnerable processor part

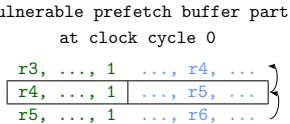
Vulnerable prefetch buffer part
at clock cycle 0

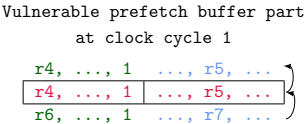


Vulnerable prefetch buffer part
at clock cycle 1



Vulnerable processor part





Faulted instruction output:

```
...
adds.w r3, r3, 1
adds.w r4, r4, 1
adds.w r4, r5, 1
adds.w r6, r5, 1
adds.w r7, r7, 1
...
```

Enhanced microarchitectural preliminary fault model

1. The dominant fault effects are instruction skip, repeat and modification, happening for different delay and amplitude
   - ↪ the modifications affect identically the destination and source register
   - ↪ at higher amplitudes, the modifications affect non-flipping bits, which contradict the preliminary fault model
2. A transfer in the prefetch mechanism is impacted
3. Some effects remain unexplained (instruction modification for example), suggesting that the fault affect unidentified parts of the microarchitecture

Conclusion

→ **Contribution:** we propose an in-depth characterization of the SCG

→ **At physical level** (ETFM):

↪ Main fault mechanism: for a DFF to correctly sample a clock's rising edge, the clock signal must meet a certain **energy threshold**

↪ The required energy quantity is influenced by **intrinsic** properties (process variability, clock routing)...

↪ ... as well as **extrinsic** properties (activity in neighboring wires)

→ **At microarchitectural level**:

↪ The main observed fault effects are instruction skip, repeat, **modification**, depending on the amplitude of the SCG

↪ The **prefetch mechanism** is vulnerable to fault

↪ Other unidentified parts of the microarchitecture are affected

Perspectives

1. The microarchitectural fault model is incomplete
   ↪ Analyzing the SCG impact on a processor we have more control and knowedge on is necessary, such as a FPGA-implemented softcore

2. The equivalence between the SCG and the CSCG is not proven
   ↪ Recreating the SCG using EMFI
   ↪ Does the ETFM still apply? Does it need adjusments?

3. The SCG exists alongside several other EM effects
   ↪ Is it possible to offer a full characterization of the EM effects?

Thank you for your attention

Questions?

TRAITOR: generation of the CSCG

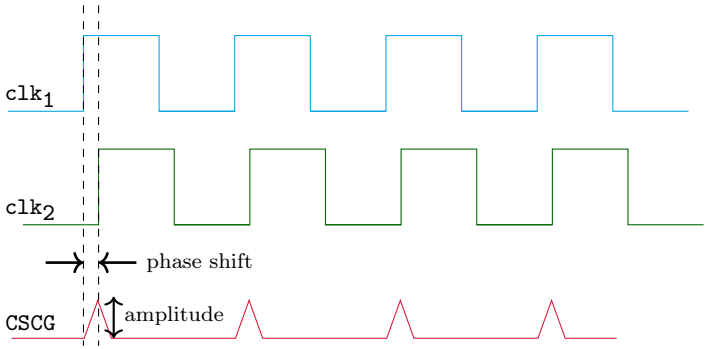
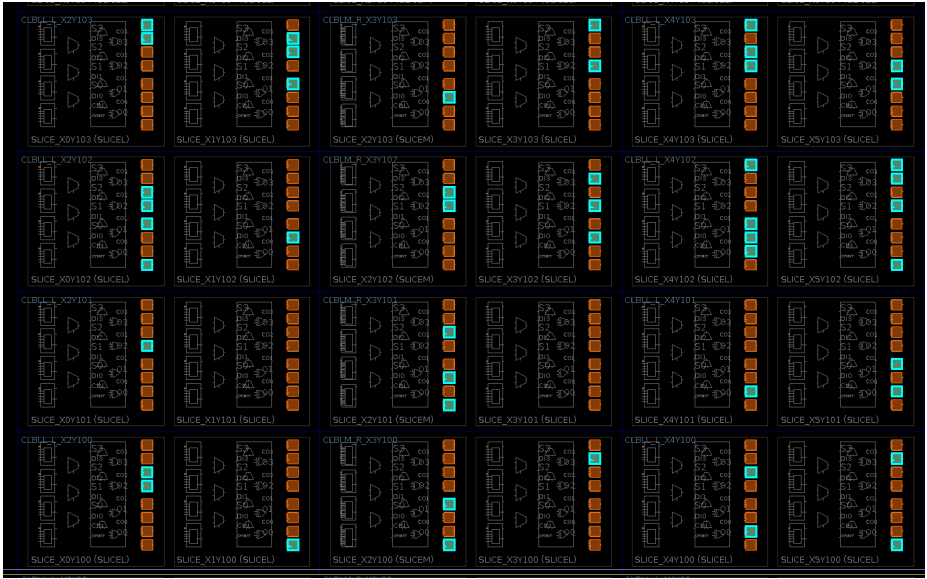
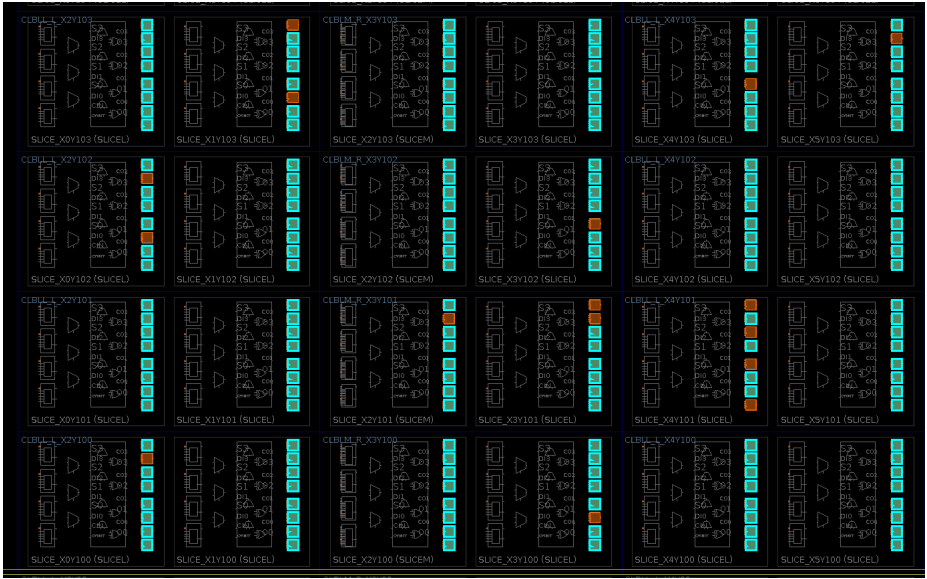


Figure: The Controlled Synchronous Clock Glitch (CSCG) is generated using two out-of-phase clocks, `clk1` and `clk2`. The TRAITOR user has the capability to replace the regular clock signal with CSCG at their discretion.

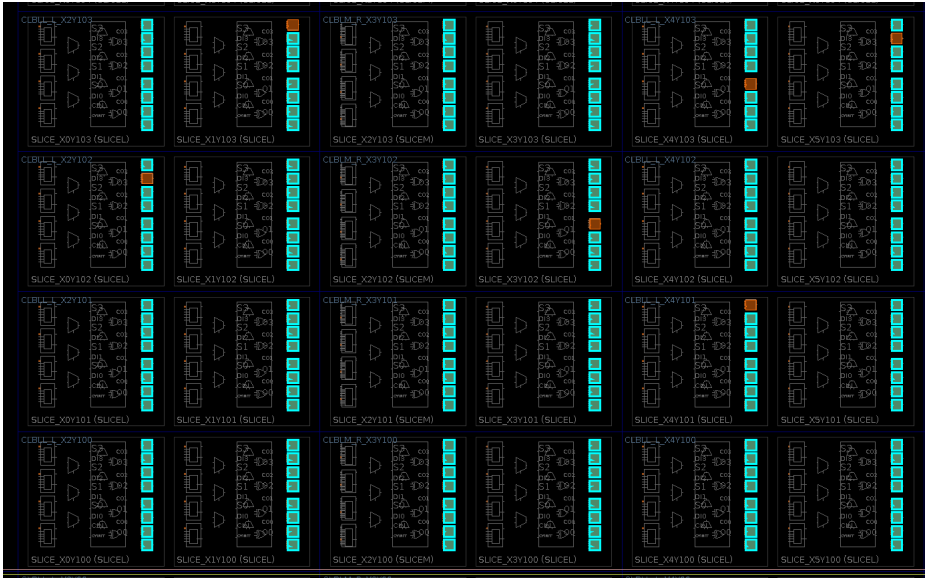
DFFs behaviour amplitude 22



DFFs behaviour amplitude 23



DFFs behaviour amplitude 24



Impact of data wires

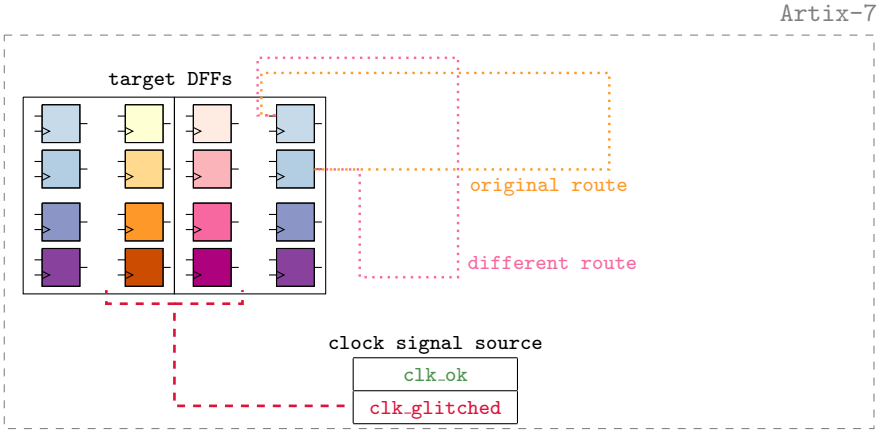
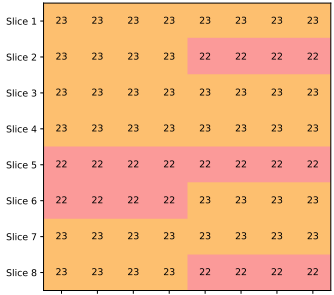
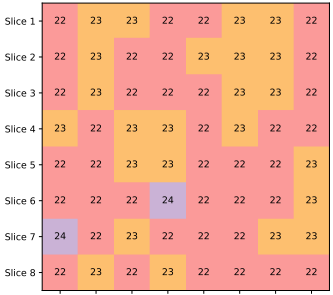


Figure: Abstract representation of the DUT placement on a Artix-7 FPGA, with route variations between two DFFs.

Impact of data wires



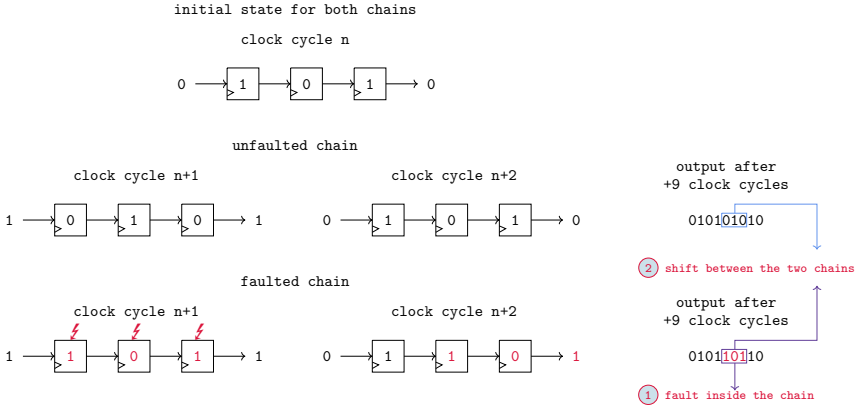
(a) Color coded fault sensitivities of the first 64 registers on mapping 1 *in-order* on FPGA 1.



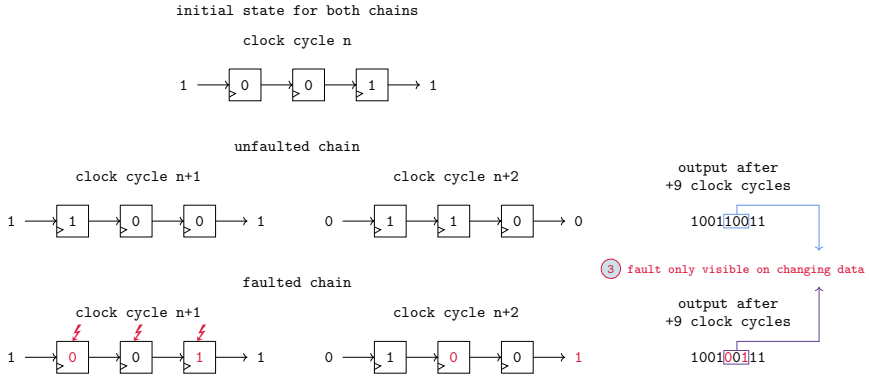
(b) Color-coded fault sensitivities of the first 64 registers on mapping 1 *in-order* with different data routing on FPGA 1

Figure: Comparing fault sensitivities between physical DFFs for different data routing.

Extension to RTL



Extension to RTL

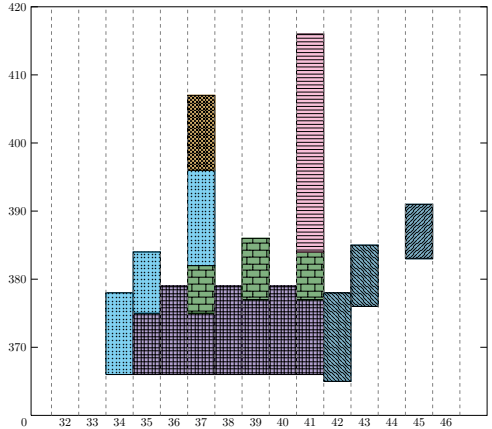


Finding dominant fault models: immediate variations

→ Target code:

↪ `listing 2`

↪ 8 nops.w + in-order, aligned adds.w instructions



FM 1: `ins. rx` is skipped
`ins. rx-1` is repeated

FM 2: `ins. rx` is skipped
`ins. rx-1` is repeated

FM 3: `ins. rx` is skipped
`ins. rx-2` is repeated

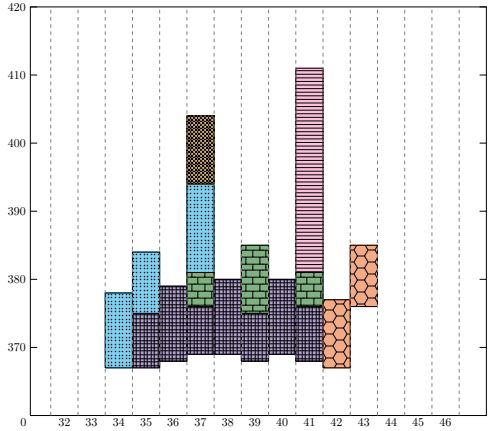
FM 4: `ins. rx` is skipped
`ins. rx+4` is repeated

FM 5: `ins. rx` is skipped
`ins. rx-4` is repeated

FM 6: `ins. r9` is repeated

Fault influence: instruction order?

↪ 8 nops.w + out-of-order, aligned adds.w instructions



FM 1: `ins. rx` is skipped
`ins. rx-1` is repeated

FM 2: `ins. rx` is skipped
`ins. rx-1` is repeated

FM 3: `ins. rx` is skipped
`ins. rx-2` is repeated

FM 4: `ins. rx` is skipped
`ins. rx+4` is repeated

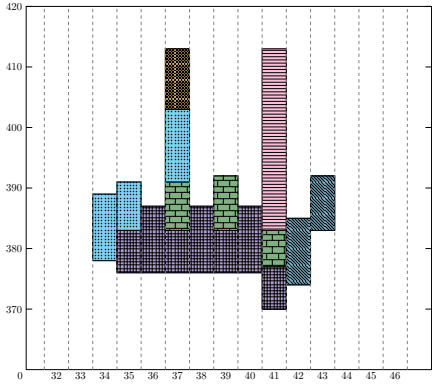
FM 5: `ins. rx` is skipped
`ins. rx-4` is repeated

FM 6: `ins. r8` is repeated

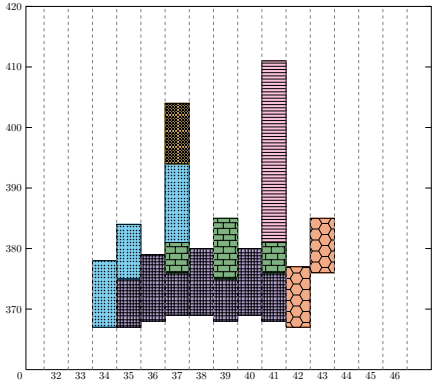
FM 7: `ins. r7` is repeated

Fault influence: delay?

in-order instructions

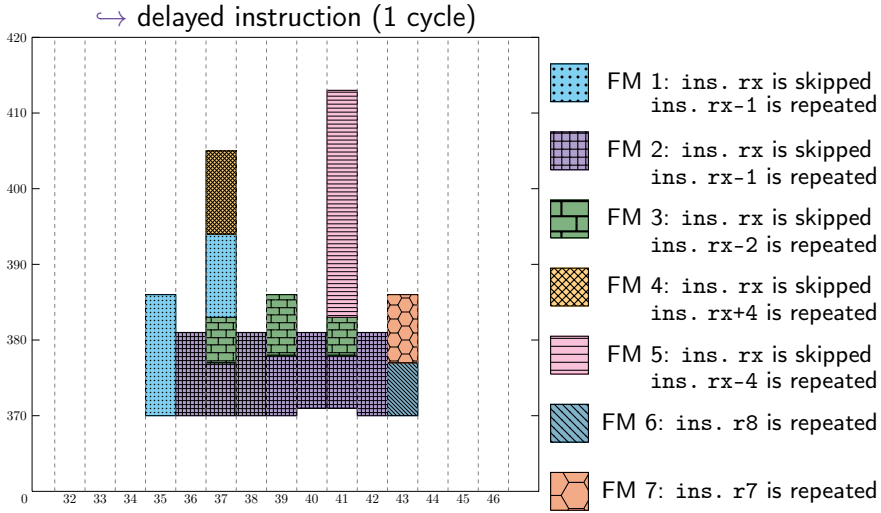


out-of-order instructions

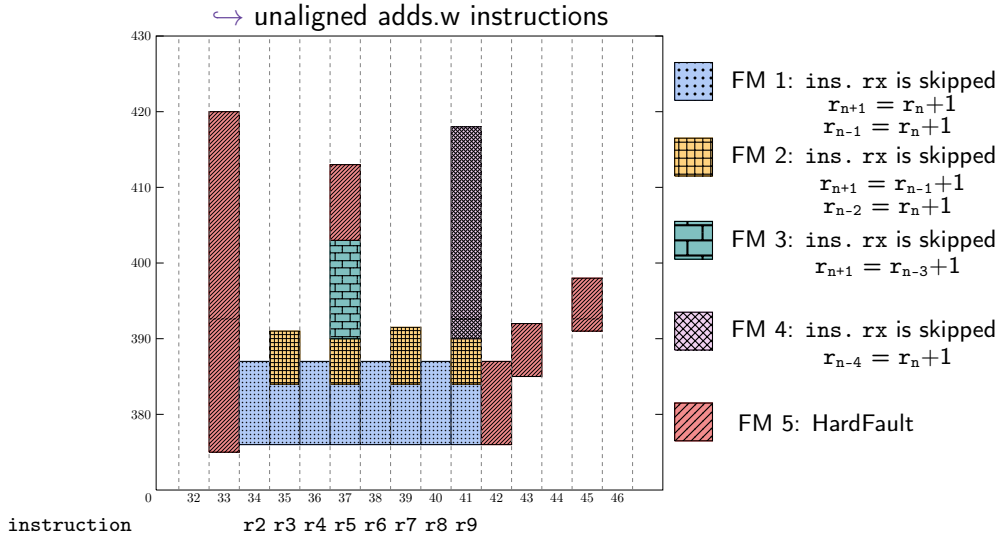


$\Rightarrow$ same fault models, indepedantly of the instruction order

Fault influence: delay?



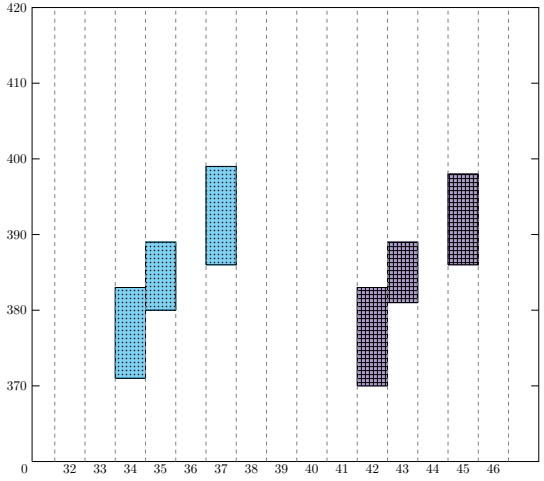
Vulnerable processor part



Vulnerable processor part

instruction	FM	r2	r3	r4	r5	r6	r7	r8	r9
nop.w	FM1	0x6c59	0x44a3	0xd0ea	0x2624	0x2e7c	0x1248	0x3330	0xed12
	FM2	0x6c59	0x44a3	0xd0ea	0x2624	0x2e7c	0x1248	0x3330	0xed12
	FM3	0x6c59	0x44a3	0xd0ea	0x2624	0x2e7c	0x1248	0x3330	0xed12
r2	FM1	0x6c5a	0x44a3	0xd0ea	0x2624	0x2e7c	0x1248	0x3330	0xed12
	FM2	0x6c5a	0x44a3	0xd0ea	0x2624	0x2e7c	0x1248	0x3330	0xed12
	FM3	0x6c5a	0x44a3	0xd0ea	0x2624	0x2e7c	0x1248	0x3330	0xed12
r3	FM1	0x6c5a	0x44a4	0xd0ea	0x2624	0x2e7c	0x1248	0x3330	0xed12
	FM2	0x6c5a	0x44a4	0xd0ea	0x2624	0x2e7c	0x1248	0x3330	0xed12
	FM3	0x6c5a	0x44a4	0xd0ea	0x2624	0x2e7c	0x1248	0x3330	0xed12
r4	FM1	0x6c5a	0x44a4	0xd0eb	0x2624	0x2e7c	0x1248	0x3330	0xed12
	FM2	0x6c5a	0x44a4	0xd0eb	0x2624	0x2e7c	0x1248	0x3330	0xed12
	FM3	0x6c5a	0x44a4	0xd0eb	0x2624	0x2e7c	0x1248	0x3330	0xed12
r5	FM1	0x6c5a	0x44a4	0x2625	0x2624	0x2e7c	0x1248	0x3330	0xed12
	FM2	0x6c5a	0x2625	0xd0eb	0x2624	0x2e7c	0x1248	0x3330	0xed12
	FM3	0x6c5a	0x44a4	0xd0eb	0x2624	0x2e7c	0x1248	0x3330	0xed12
r6	FM1	0x6c5a	0x44a4	0x2625	0x2624	0x2625	0x1248	0x3330	0xed12
	FM2	0x6c5a	0x2625	0xd0eb	0x2624	0xd0ec	0x1248	0x3330	0xed12
	FM3	0x6c5a	0x44a4	0xd0eb	0x2624	0x6c5b	0x1248	0x3330	0xed12
r7	FM1	0x6c5a	0x44a4	0x2625	0x2624	0x2625	0x1249	0x3330	0xed12
	FM2	0x6c5a	0x2625	0xd0eb	0x2624	0xd0ec	0x1249	0x3330	0xed12
	FM3	0x6c5a	0x44a4	0xd0eb	0x2624	0x6c5b	0x1249	0x3330	0xed12

Fault `adds.w r7, r7, 1`



FM 1: `ins. rx` is skipped
`ins. rx-1` is repeated

FM 2: `ins. rx` is skipped
`ins. rx-1` is repeated