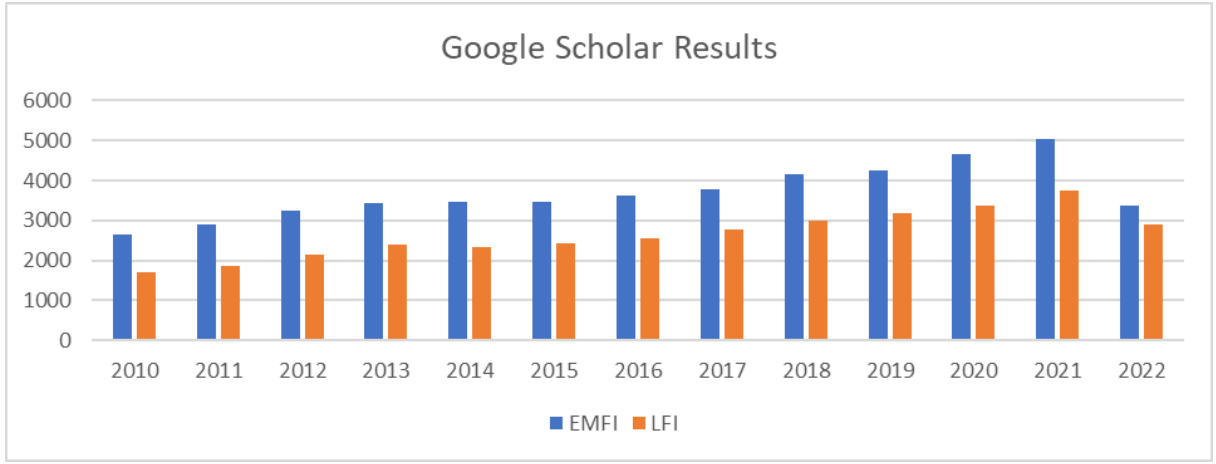
Toward a Low-Cost Methodology for EM Fault Emulation on FPGA

Jia Yun PO, Sami EL AMRAOUI, **Paolo MAISTRI**

TIMA Laboratory, Grenoble (FR)

EM Fault Injection

- Electromagnetic Fault Injections (EMFI) exploited more and more

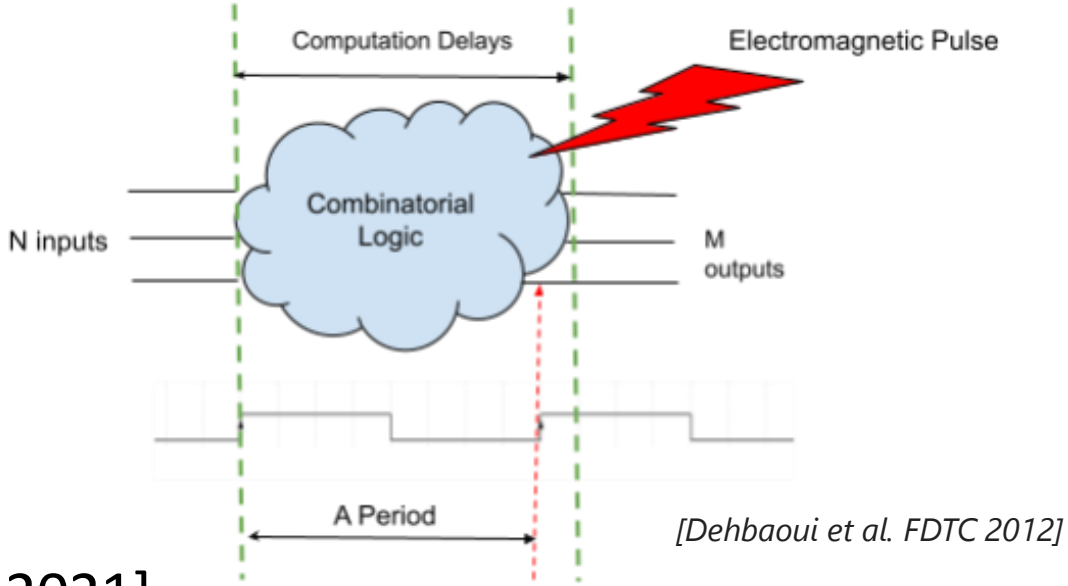


- Great temporal precision
- Good spatial resolution
- Local faults
- Cheaper than laser (LFI)
- Simpler than laser

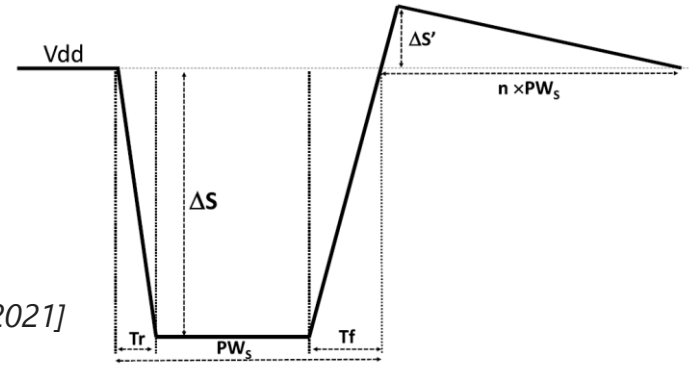
Exploitable faults … but WHY?

EM Fault Model

- EM Fault model?
  - Timing violations [Dehbaoui, FDTC 2012]
  - Set/Reset disruption [Ordas, JCEN 2017]
  - Sampling fault [Dumont, TCAD 2021]
- Why?
  - Interaction with Power Grid [Dumont, TCAD 2021]



[Dehbaoui et al. FDTC 2012]



[Dumont, TCAD 2021]

- Results depend on [Ghodrati, DAC 2018]
  - Probe Location
  - Timing
- Specific to EMFI

EMPs vs. Clock Glitches

	EMP	Clock glitch
Type	Delay & Sample Faults	Delay Faults
Granularity	Local	Global
Requirements	Pulse generator, probes	DCM
Cost	+++	+
Required Know-How		

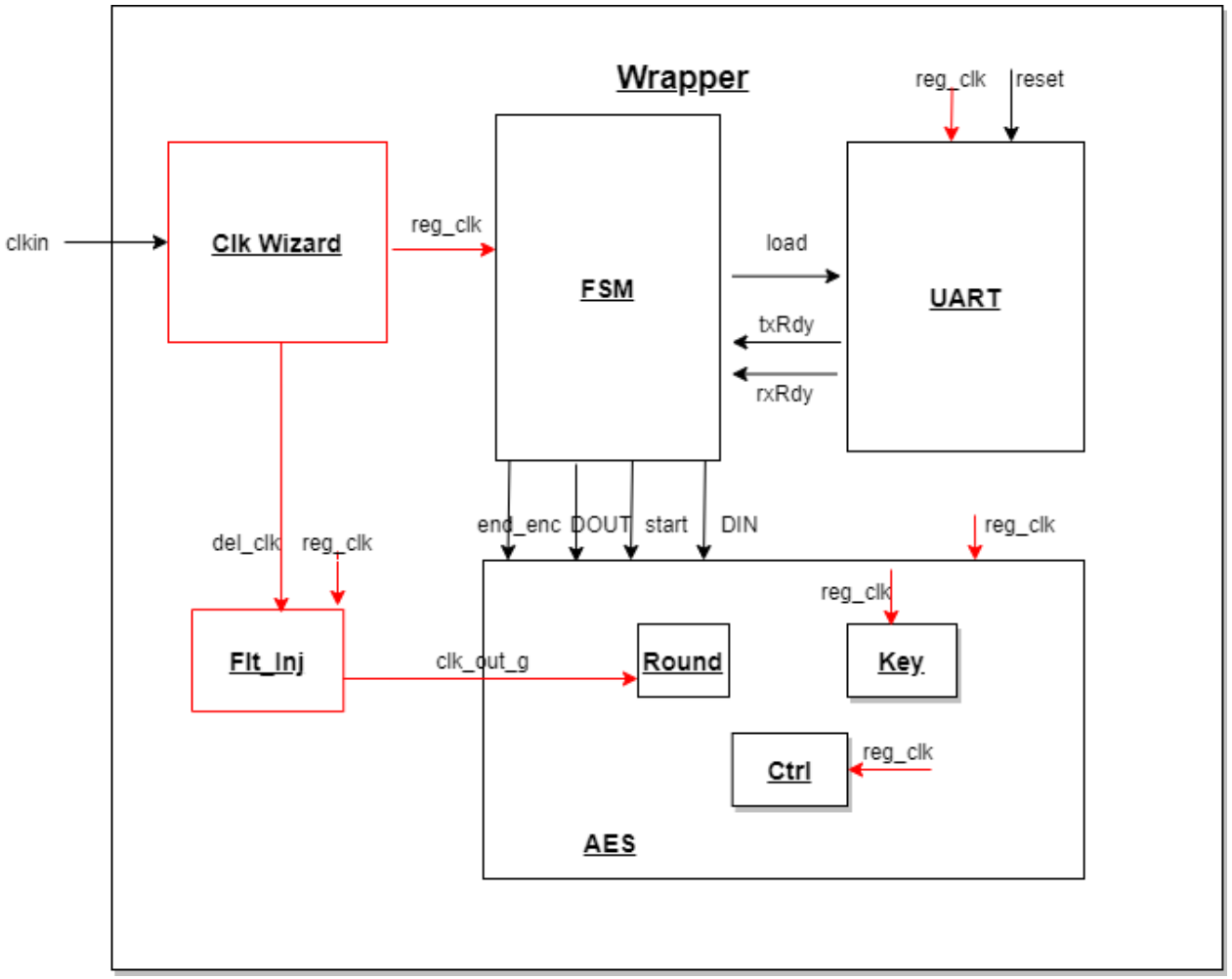
Is it possible to emulate one with the other ?

Objective

- EM fault emulation at RTL
  - Configurable targets
  - Timing controllability
  - Full configurability
  - Hardware acceleration

- Evaluate the functional effects of EM fault injections on early prototypes
- Evaluate architectural countermeasures

Local Fault Injection



Instrumentation framework

1) Define targets

2) Implement

- Clock Wizard
  - *Glitch definition*
- Fault Injector
  - *Timing control*

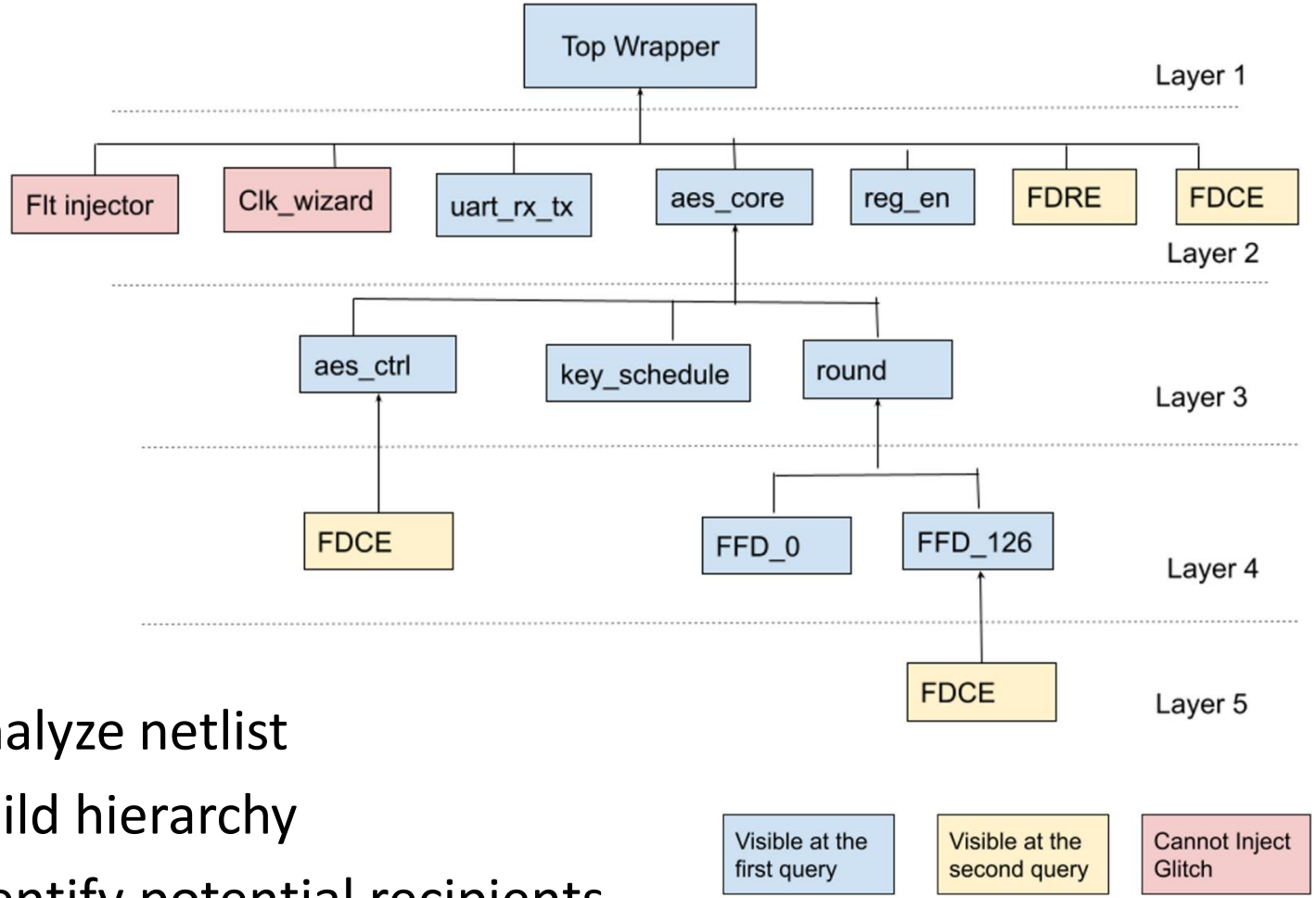
1) Add & Modify connections

Target Selection

- Parse netlist

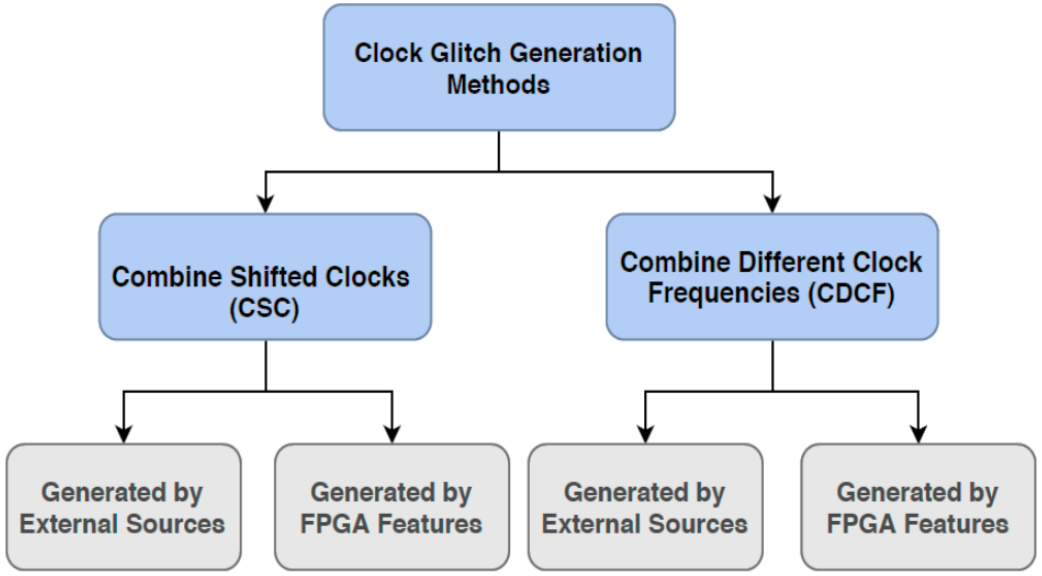
```
module FFD_108 ──────────► Module's declaration name
   (Q_reg_0,
    Q_reg_1,
    clk_IBUF_BUFG,        } Port instances
    Q_reg_2);
  output Q_reg_0;
  input Q_reg_1;
  input clk_IBUF_BUFG;
  input Q_reg_2;
                          } Connectives within declaration
  wire \<const1> ;
  wire Q_reg_0;
  wire Q_reg_1;
  wire Q_reg_2;
  wire clk_IBUF_BUFG;

  FDCE #(
    .INIT(1'b0))
    Q_reg
       (.C(clk_IBUF_BUFG),
        .CE(\<const1> ),
        .CLR(Q_reg_2),
        .D(Q_reg_1),
        .Q(Q_reg_0));
  VCC VCC
       (.P(\<const1> ));
endmodule
```

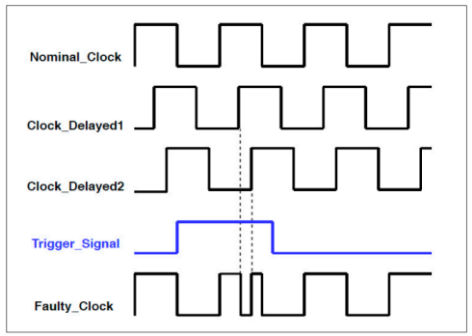
Instances of the other lower hierarchical modules



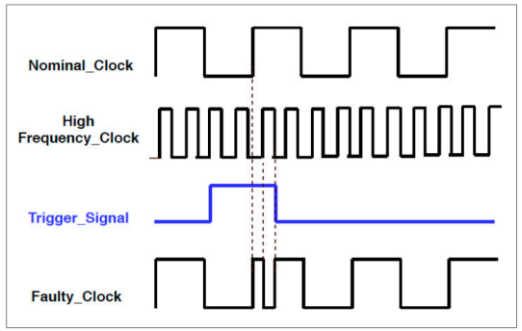
- Analyze netlist
- Build hierarchy
- Identify potential recipients

Local Clock Glitching on FPGA

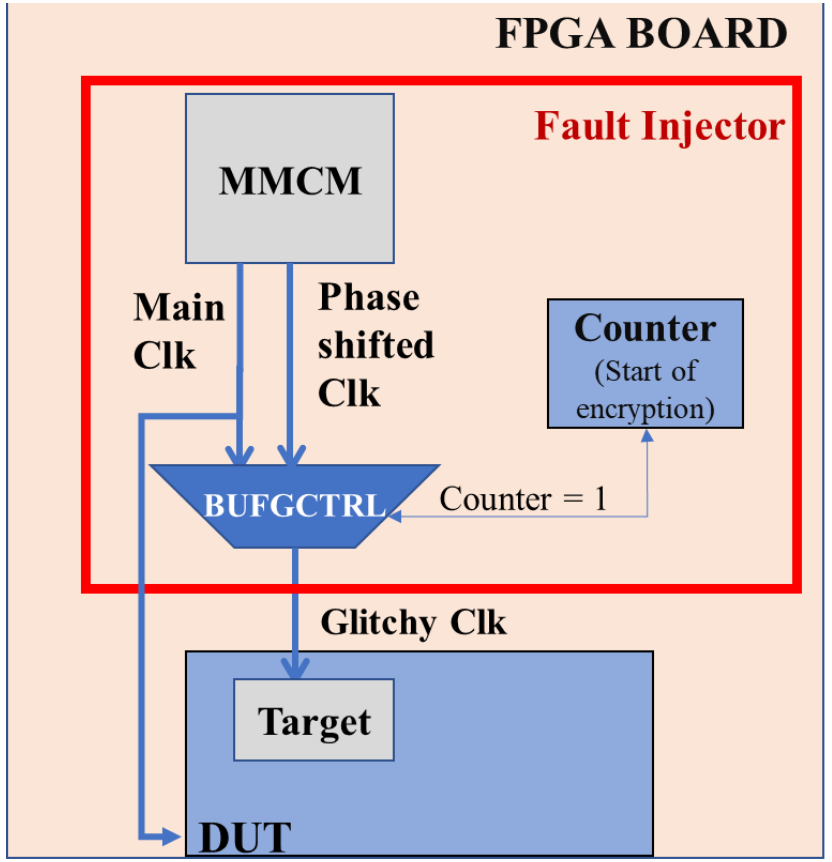




Faulty clock generation using two shifted signals



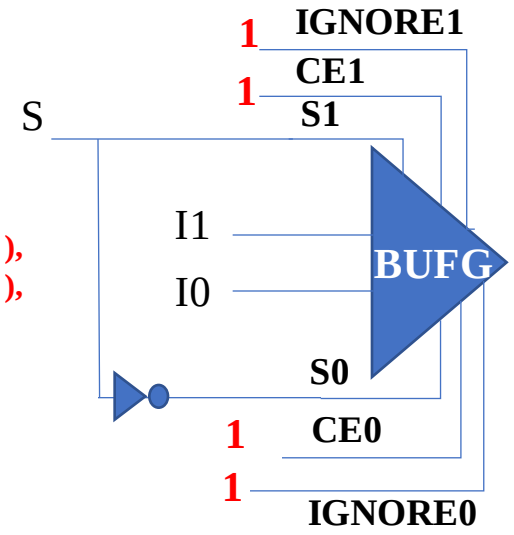
Faulty clock generation using high frequency signal



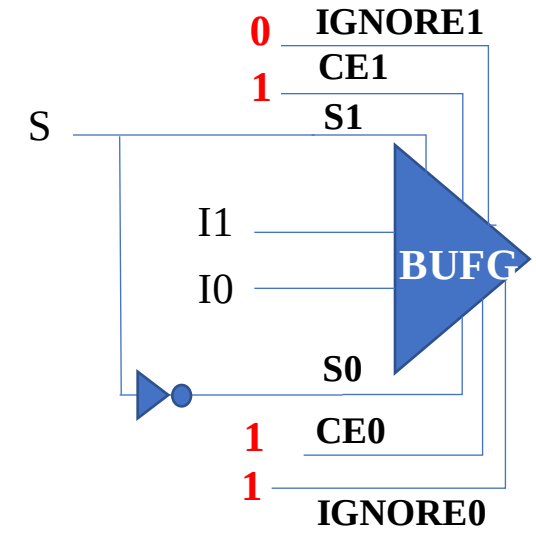
Internal Configuration

0° < Phase shift < 232.5°

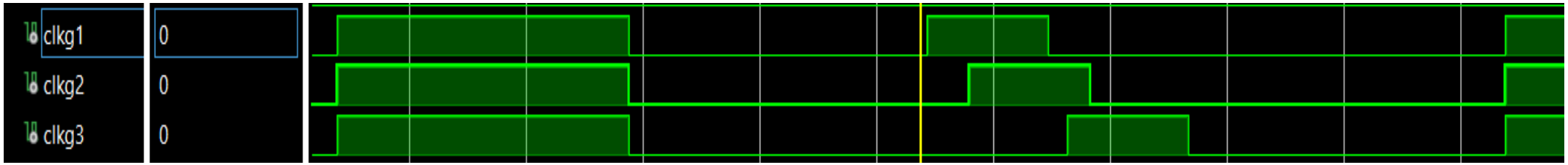
```
BUFGCTRL_inst1
   (.S0(S0),
    .S1(S1),
    .I0(clk_out1),
    .I1(clk_out2),
    .IGNORE0(\<const1> ),
    .IGNORE1(\<const1> ),
    .O(clk_out_g),
    .CE0(\<const1> ),
    .CE1(\<const1> ));
```

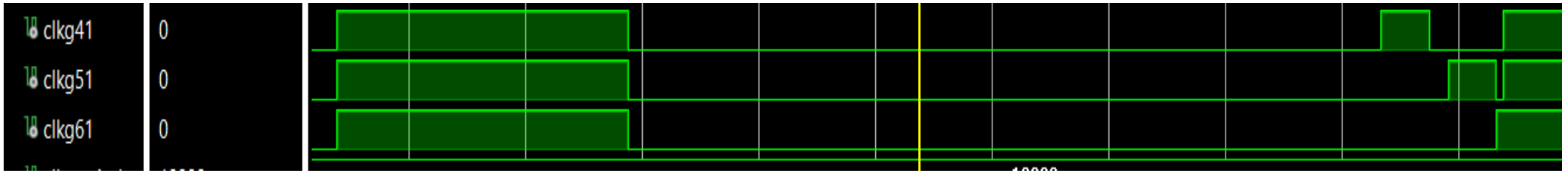



```
BUFGCTRL_inst1
   (.S0(S0),
    .S1(S1),
    .I0(clk_out1),
    .I1(clk_out2),
    .IGNORE0(\<const1> ),
    .IGNORE1(\<const0> ),
    .O(clk_out_g),
    .CE0(\<const1> ),
    .CE1(\<const1> ));
```



232.5° =< Phase shift < 326.25°





Automated Parameter Tuning

MMCM Parameters

D : Pre-divider
M : Feedback divider
O: Output divider

$$F_{VCO} = F_{CLKIN} \times \frac{M}{D}$$

$$F_{OUT} = F_{CLKIN} \times \frac{M}{D \times O}$$

- **Phase shift granularity = 45° / Output divider**
- Freq VCO_max ⇗ ➡ Output divider ⇗

Path delay (ns)	Slack (ns)	Phase shift_min (°)	O (7...12)	Phase Shift stepsize = 45° / O (°)	Accuracy (%)
9,886	0.114	**4,104°**	10	**4,5°**	91,2
9,846	0.154	**5,544°**	8	**5,625°**	**98,56**
9,8	0.2	**7,2°**	12	**7,5°**	96
9,7	0.3	**10,8°**	12	**11,25°**	96

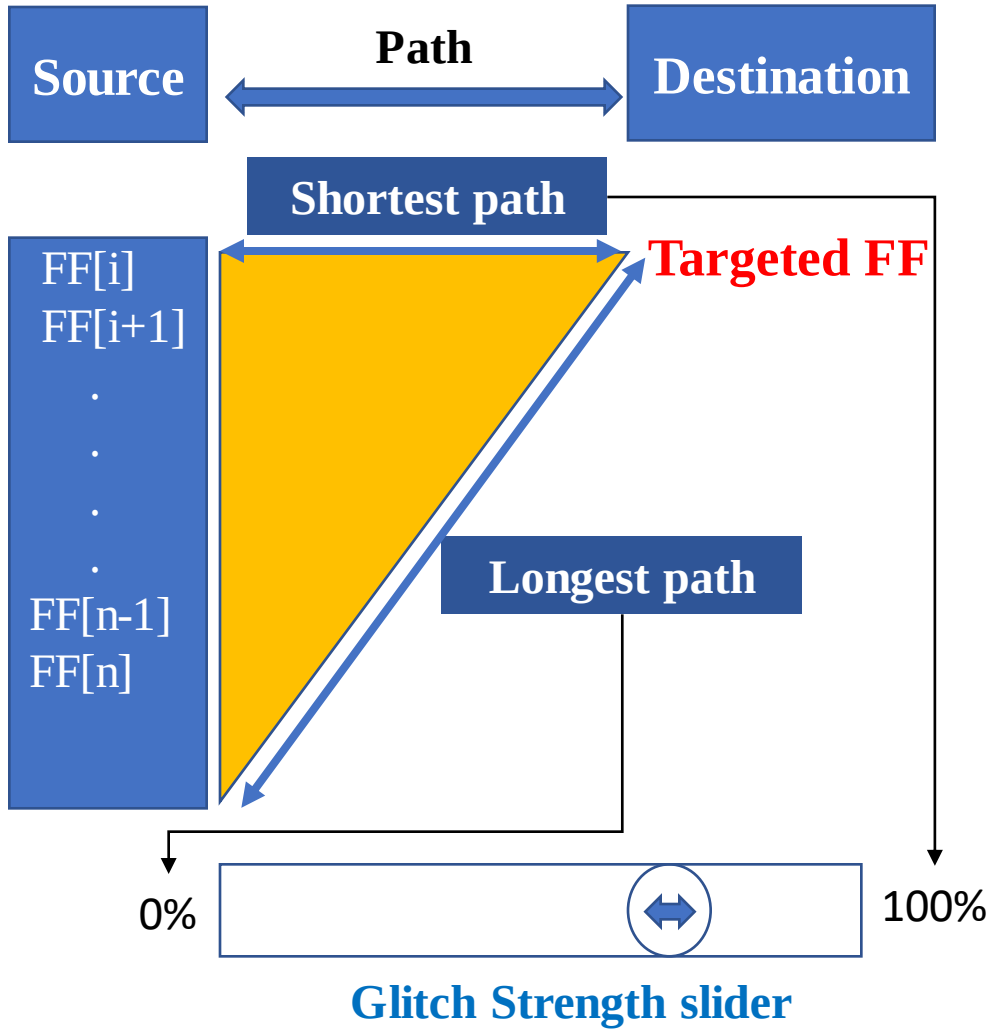
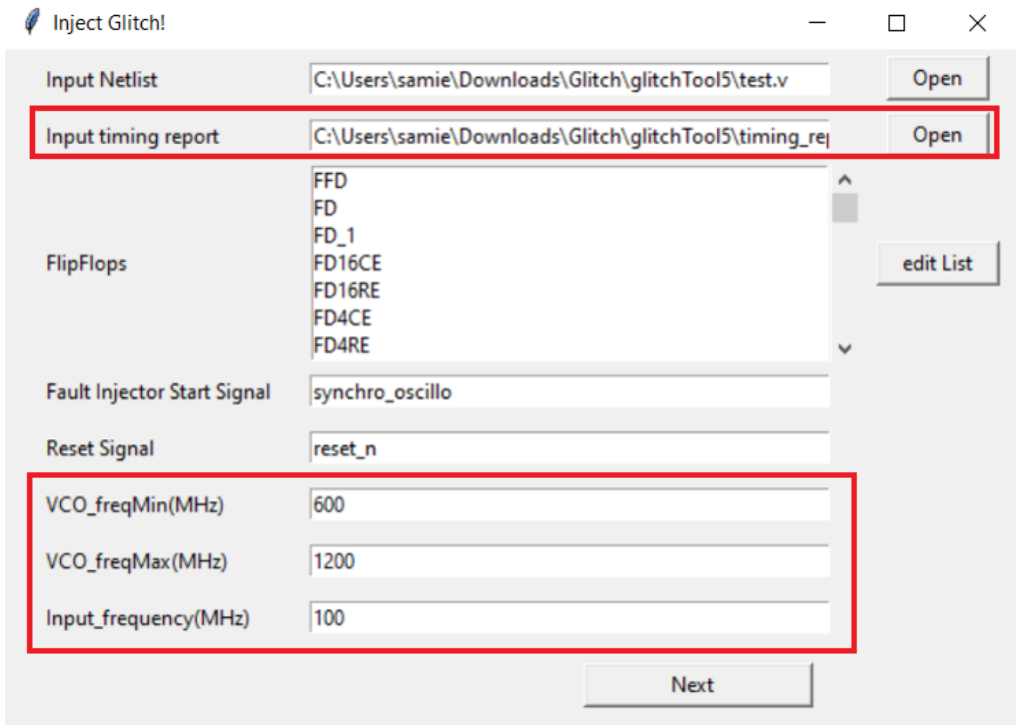
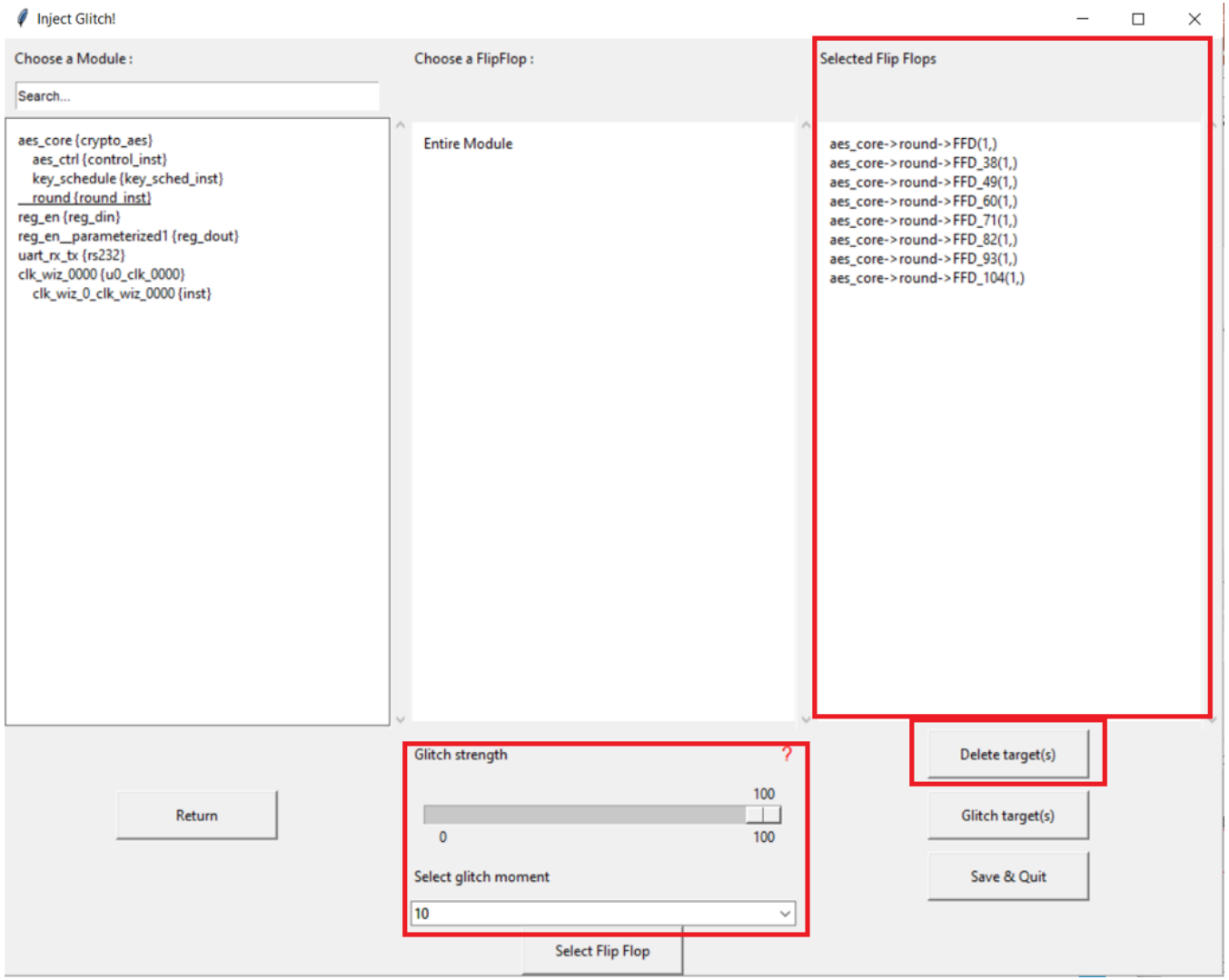


Table: *PS granularity and its accuracy for a* ***10 ns*** *clock period and* ***Fvco_max =1.2GHz***

Tool


Inject Glitch!
Input Netlist
C:\Users\samie\Downloads\Glitch\glitchTool5\test.v
Open
Input timing report
C:\Users\samie\Downloads\Glitch\glitchTool5\timing_re
Open
FlipFlops
FFD
FD
FD_1
FD16CE
FD16RE
FD4CE
FD4RE
edit List
Fault Injector Start Signal
synchro_oscillo
Reset Signal
reset_n
VCO_freqMin(MHz)
600
VCO_freqMax(MHz)
1200
Input_frequency(MHz)
100
Next

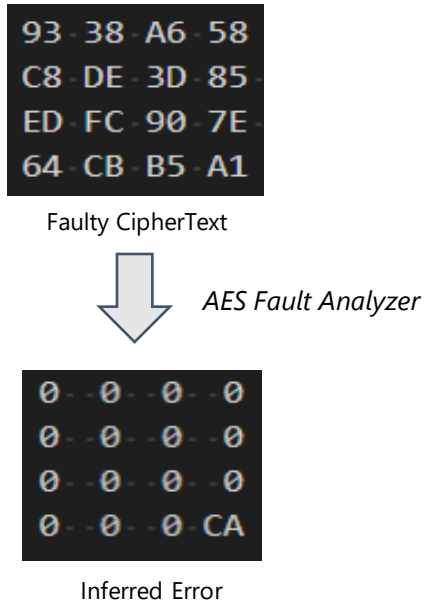

Inject Glitch!
Choose a Module :
Search...
aes_core {crypto_aes}
aes_ctrl {control_inst}
key_schedule {key_sched_inst}
round {round_inst}
reg_en {reg_din}
reg_en__parameterized1 {reg_dout}
uart_rx_tx {rs232}
clk_wiz_0000 {u0_clk_0000}
clk_wiz_0_clk_wiz_0000 {inst}
Choose a FlipFlop :
Entire Module
Selected Flip Flops
aes_core->round->FFD(1,)
aes_core->round->FFD_38(1,)
aes_core->round->FFD_49(1,)
aes_core->round->FFD_60(1,)
aes_core->round->FFD_71(1,)
aes_core->round->FFD_82(1,)
aes_core->round->FFD_93(1,)
aes_core->round->FFD_104(1,)
Return
Glitch strength
100
0
100
Select glitch moment
10
Select Flip Flop
Delete target(s)
Glitch target(s)
Save & Quit

Validation

- 3 AES designs
  - AES-128, AES Parity, AES DDR
- Arty-7 FPGA Board
  - Artix 7-100

Tests

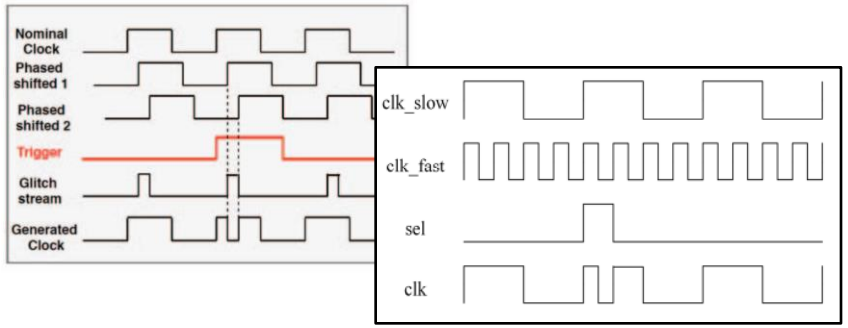
- Different inputs parameters of Fault Injector
  - Phase, Duty Cycle, Frequency
- Different targets (Eg: All bytes / single byte of data registers)
- Different timing (1-10th round of AES)

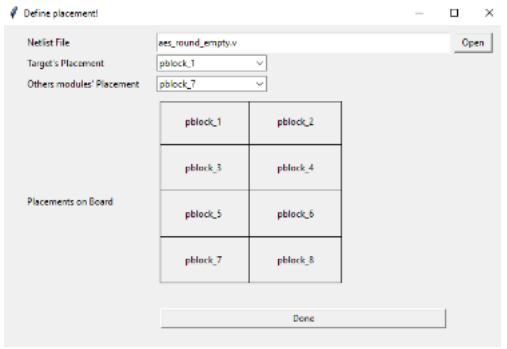


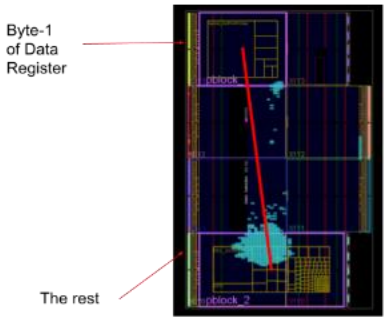
Perspectives

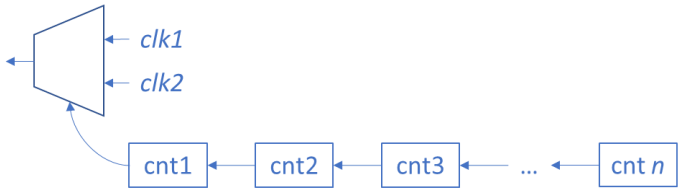
- Alternative glitching generation
  - Combined Shifted Clocks vs Combined Frequency
- Emulate other models (“swing”, additional cycle, …)
- Compare to actual EM campaigns
- Fine placement control
  - Isolate targets on FPGA
  - Precision vs Consistency
- Multiple faults (targets/triggers)
  - More than one fault injector (device-limited)
  - Select target from different components
- Advanced triggering conditions
  - E.g., burst/multiple injections

➢ Polish and Publish









Thank you!