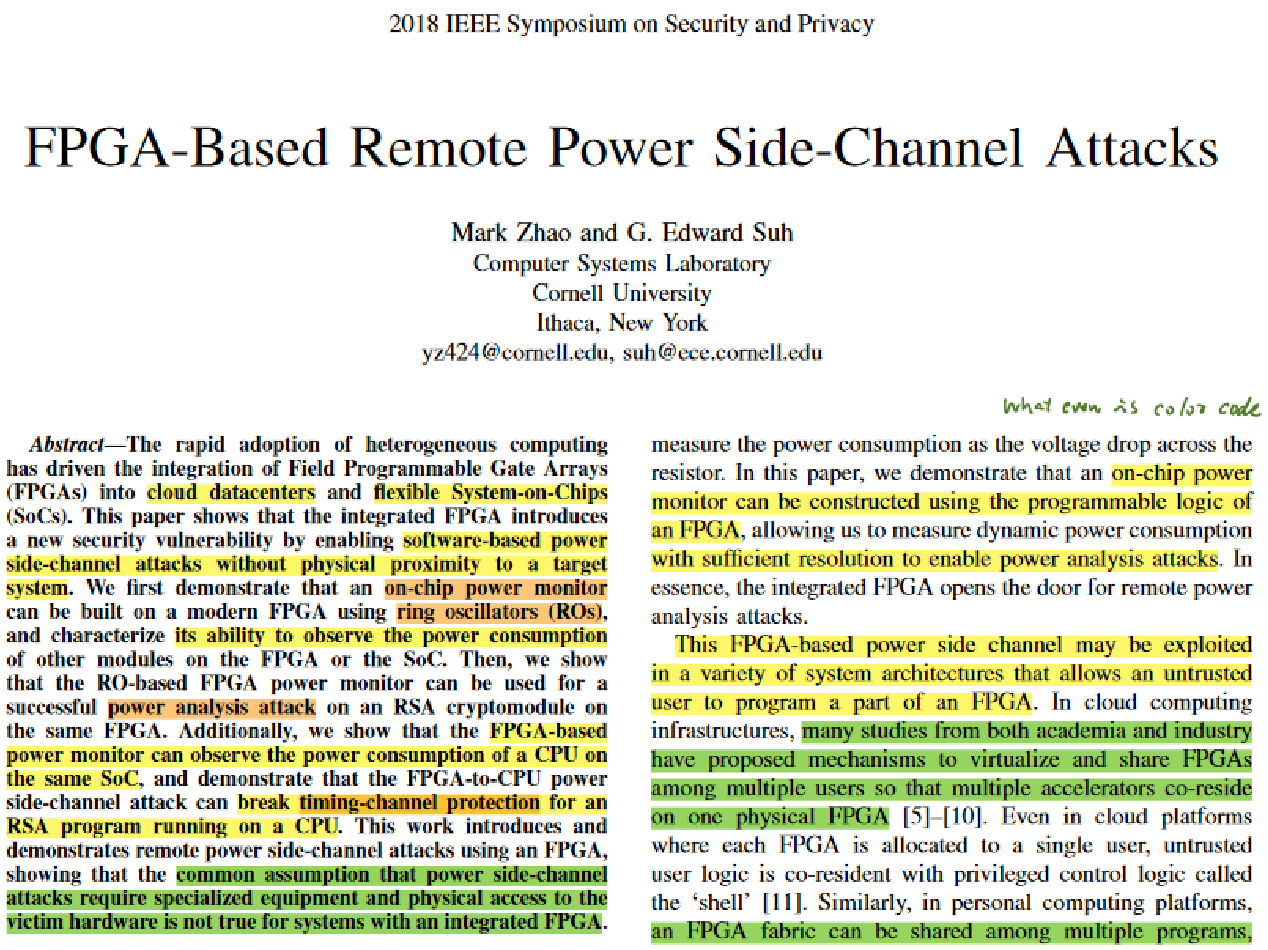
11/29 meeting

FPGA-Based Remote Power Side Channel Attacks



2018 IEEE Symposium on Security and Privacy

FPGA-Based Remote Power Side-Channel Attacks

Mark Zhao and G. Edward Suh
Computer Systems Laboratory
Cornell University
Ithaca, New York
yz424@cornell.edu, suh@ece.cornell.edu

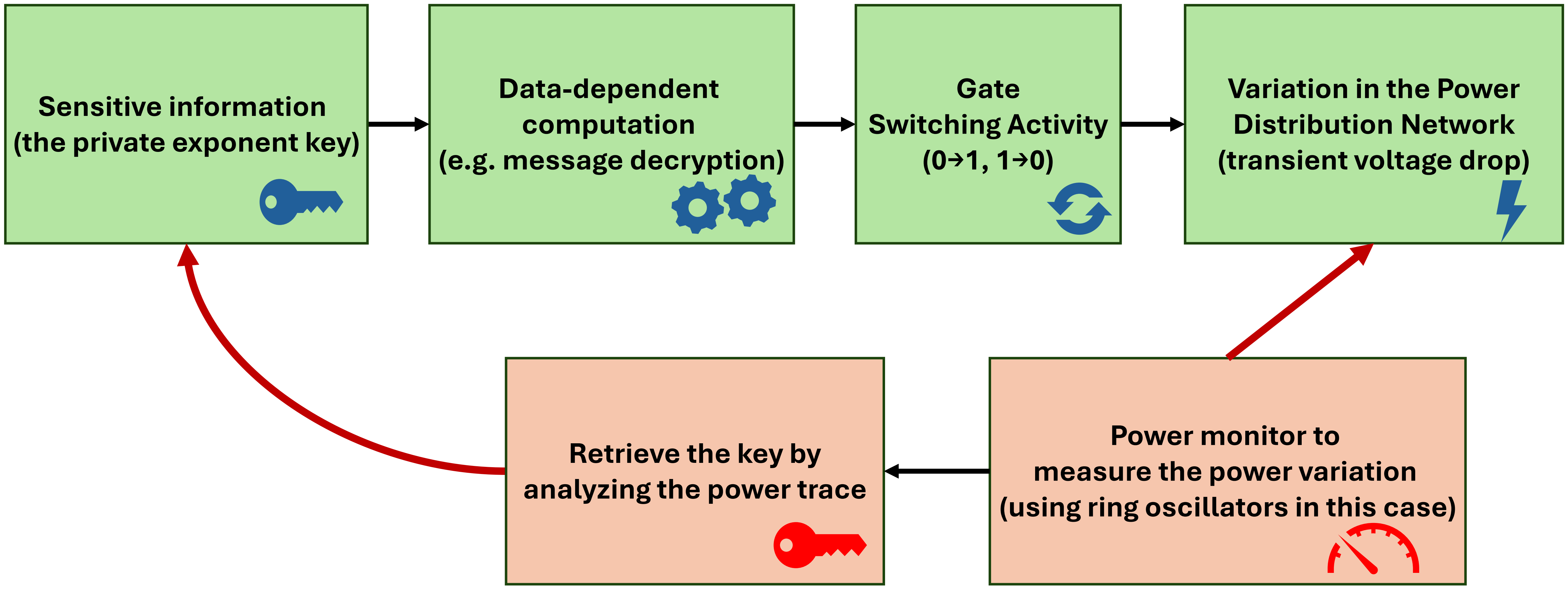
what even is color code

Abstract—The rapid adoption of heterogeneous computing has driven the integration of Field Programmable Gate Arrays (FPGAs) into cloud datacenters and flexible System-on-Chips (SoCs). This paper shows that the integrated FPGA introduces a new security vulnerability by enabling software-based power side-channel attacks without physical proximity to a target system. We first demonstrate that an on-chip power monitor can be built on a modern FPGA using ring oscillators (ROs), and characterize its ability to observe the power consumption of other modules on the FPGA or the SoC. Then, we show that the RO-based FPGA power monitor can be used for a successful power analysis attack on an RSA cryptomodule on the same FPGA. Additionally, we show that the FPGA-based power monitor can observe the power consumption of a CPU on the same SoC, and demonstrate that the FPGA-to-CPU power side-channel attack can break timing-channel protection for an RSA program running on a CPU. This work introduces and demonstrates remote power side-channel attacks using an FPGA, showing that the common assumption that power side-channel attacks require specialized equipment and physical access to the victim hardware is not true for systems with an integrated FPGA.

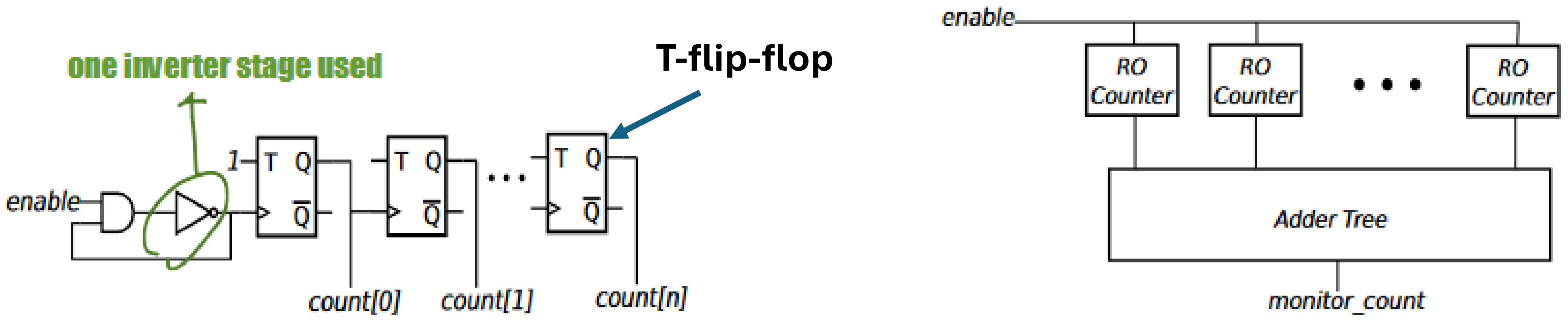
measure the power consumption as the voltage drop across the resistor. In this paper, we demonstrate that an on-chip power monitor can be constructed using the programmable logic of an FPGA, allowing us to measure dynamic power consumption with sufficient resolution to enable power analysis attacks. In essence, the integrated FPGA opens the door for remote power analysis attacks.

This FPGA-based power side channel may be exploited in a variety of system architectures that allows an untrusted user to program a part of an FPGA. In cloud computing infrastructures, many studies from both academia and industry have proposed mechanisms to virtualize and share FPGAs among multiple users so that multiple accelerators co-reside on one physical FPGA [5]–[10]. Even in cloud platforms where each FPGA is allocated to a single user, untrusted user logic is co-resident with privileged control logic called the 'shell' [11]. Similarly, in personal computing platforms, an FPGA fabric can be shared among multiple programs,

Power Analysis Attack on FPGA


Sensitive information (the private exponent key)
Data-dependent computation (e.g. message decryption)
Gate Switching Activity (0→1, 1→0)
Variation in the Power Distribution Network (transient voltage drop)
Power monitor to measure the power variation (using ring oscillators in this case)
Retrieve the key by analyzing the power trace

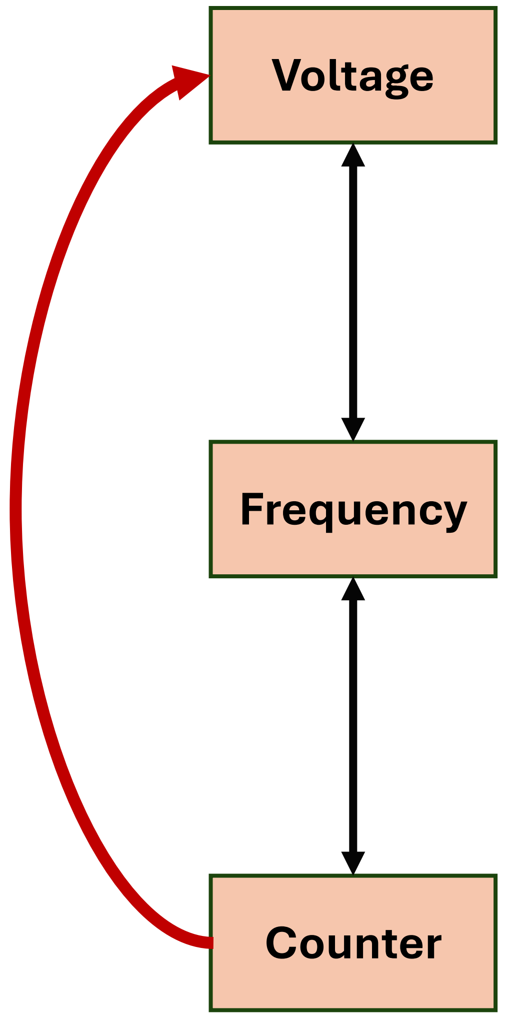
Ring Oscillator design



(a) A single RO counter. (b) Adder tree to combine multiple RO counters.

Fig. 2. A Ring Oscillator (RO) based on-chip power monitor design.

RO-based Power Monitor Operating Principle



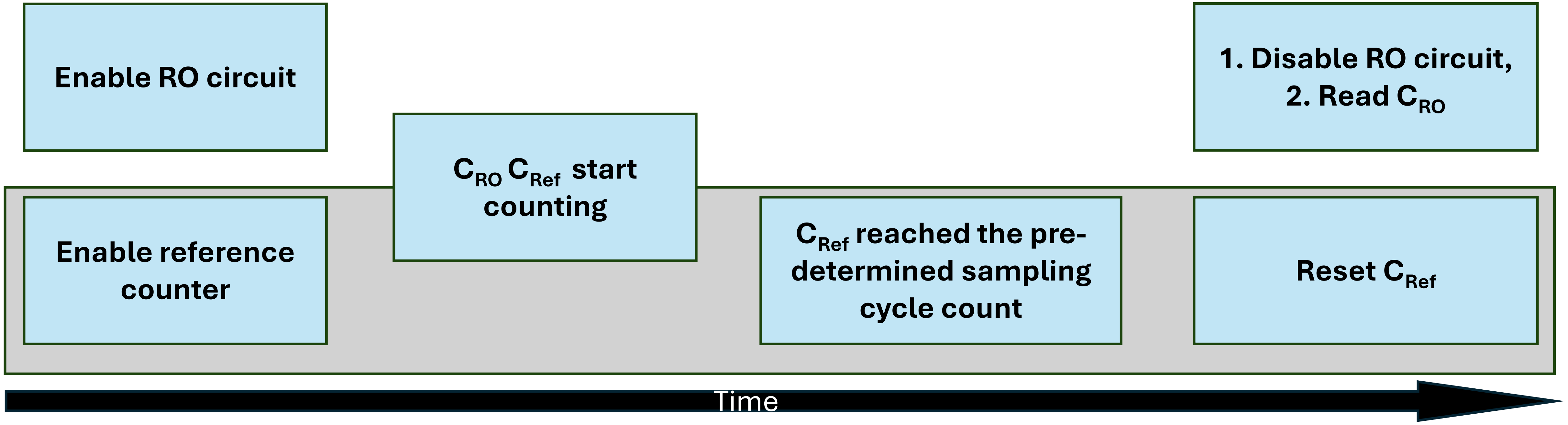
$$f_{RO} \approx k * V(x,y,t) + f_0$$

- f_{RO}: oscillation frequency
- $V(x,y,t)$: transient supply voltage
- k, f_0: constants

$$f_{RO} = C_{RO} * \frac{f_{Ref}}{C_{Ref}} + \varepsilon$$

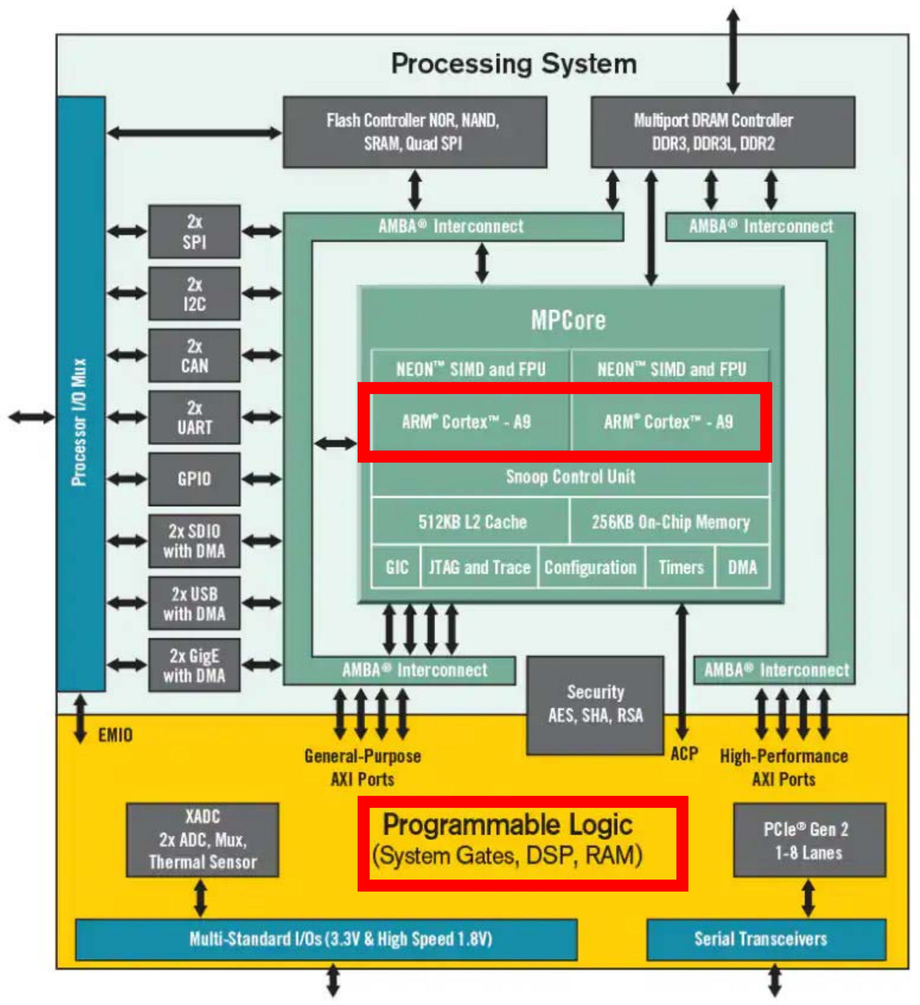
- C_{RO}: ring oscillator T-flip-flop counter
- C_{Ref}: **pre-defined sampling cycle count**
- f_{Ref}: reference clock frequency (FPGA system clock)
- ε: quantization error introduced by the phase difference between the two clocks

RO frequency measure method



RO-based Power Monitor Characterization 1

- Zynq-7020 SoC on a Zedboard
- The SoC has:
  - A hardened dual-core ARM Cortex-A9
  - Artix-7 equivalent FPGA with 53,200 LUTs



RO-based Power Monitor Characterization 2

- Create 20 RO circuits
- Take the sum of all RO counters (C_{RO}) as the final power monitor value

- Characterize RO frequency by:
  - A known switching activity
  - Spatial proximity to target logic
  - Sampling period (C_{Ref})

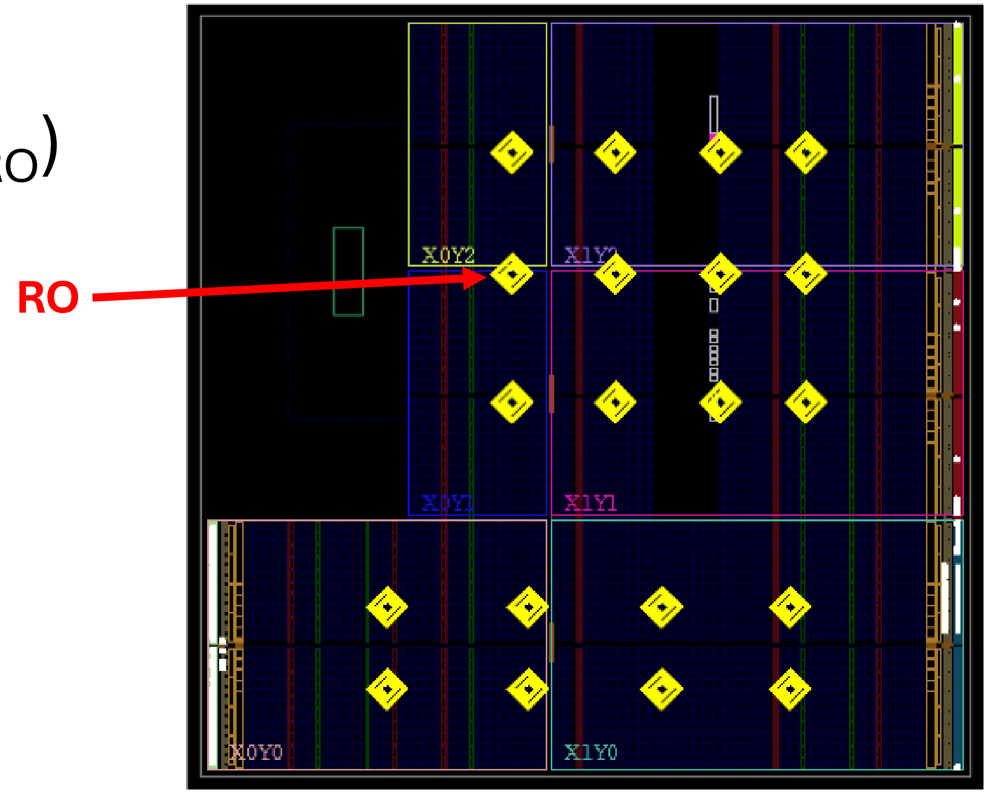


Fig. 3. RO placements on the FPGA fabric indicated by diamond marks.

RO Power Estimate 1

$$\begin{tikzcd}\boxed{\text{Frequency change of RO}} \arrow[r, leftrightarrow] & \boxed{\text{Known change in switching activity}}\end{tikzcd}$$

- 16,000 power viruses **covering the majority of FPGA**
  - Power virus switches on and off constantly
- Sampling period of 1001 cycles at 100MHz
- 1,000,000 RO power monitor samples as the steady state RO frequency

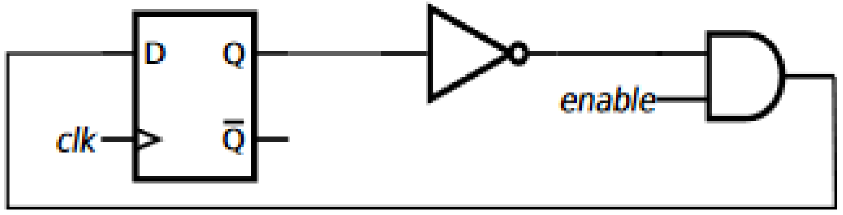


Fig. 4. One instance of the power virus. Nets in the circuit switch with an activity factor of either 1 or 0 depending on *enable*.

RO Power Estimate 2

Frequency change of RO ⟷ **Known change in switching activity**

- 33 levels of power consumption
  - By grouping every 500 evenly-distributed power viruses together
  - Make sure power viruses in each level is evenly-distributed
- **Linear correlation between switching activity and oscillation frequency**

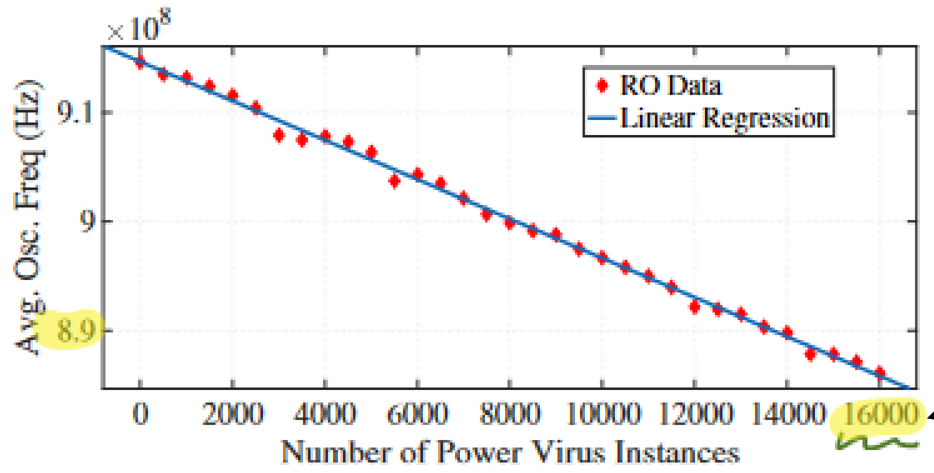


Fig. 5. The average RO frequency versus the power consumption level. A linear regression is shown to fit the data.

RO frequency is lower when there's more switching activity

RO Spatial dependence 1

$$\begin{tikzcd}\boxed{\text{Frequency change of RO}} \arrow[r, leftrightarrow] & \boxed{\text{Spatial proximity to switching logic}}\end{tikzcd}$$

- 6400 power viruses
- **At the upper third region (X0Y2 & X1Y2)**

- Sampling period of 1001 cycles at 100MHz
- 1,000,000 samples
- **Measure the RO counter value**
  - **Once with all power viruses on**
  - **Once with all power viruses off**

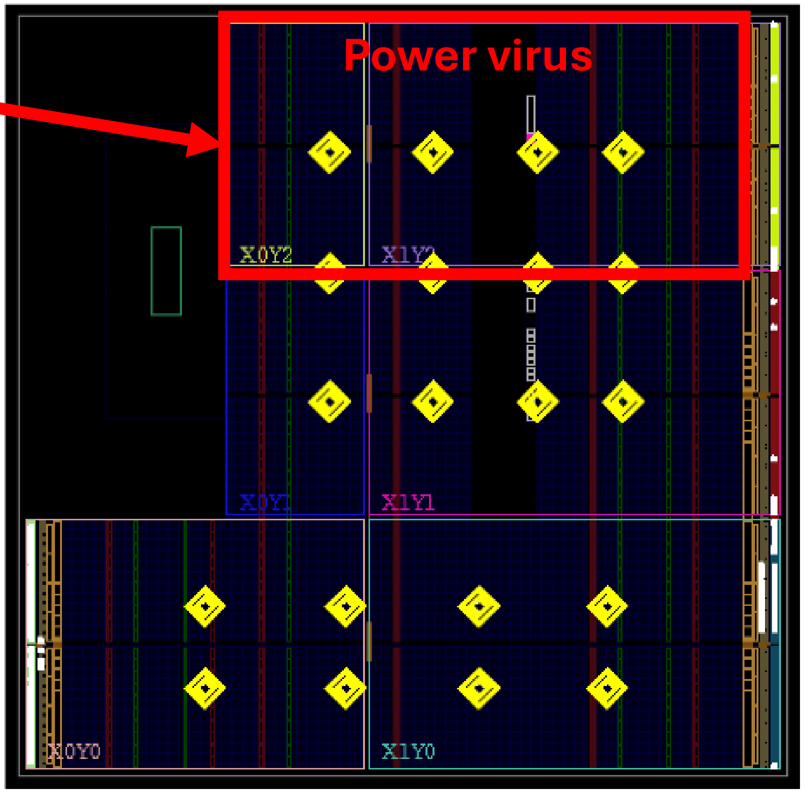


Fig. 3. RO placements on the FPGA fabric indicated by diamond marks.

RO Spatial dependence 2

Frequency change of RO ⟷ **Spatial proximity to switching logic**

- Calculate percent change in the oscillation frequency f_{RO}
  - Removing the process variations in ROs
- Spatial proximity to switching logic increases RO sensitivity

Separating logic by unused logic block does not eliminate information leakage via power consumption

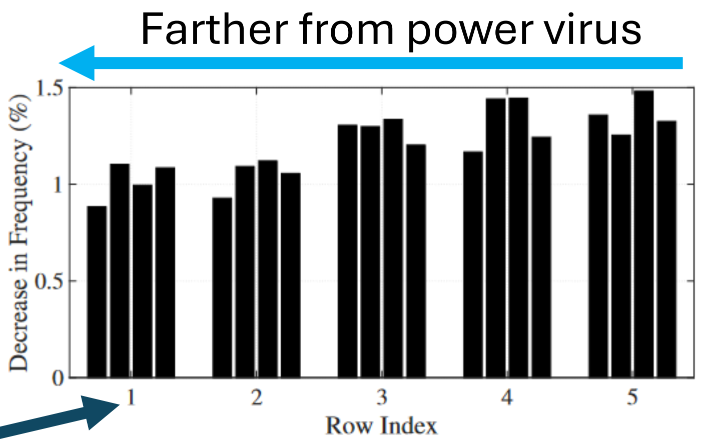


Fig. 6. The RO frequency drop with varying RO locations. Row indices correspond to the rows in Figure 3, increasing from bottom to top (i.e. higher row indices are closer to power viruses). In each row, bars correspond to RO instances from left to right in both the plot and the FPGA.

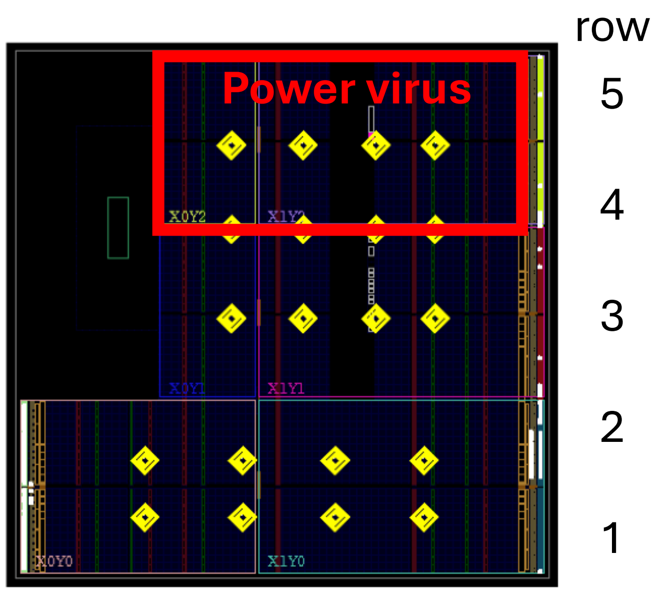


Fig. 3. RO placements on the FPGA fabric indicated by diamond marks.

RO Time Resolution 1

$$\begin{tikzcd} \boxed{\textbf{Frequency change of RO}} \arrow[r, leftrightarrow] & \boxed{\textbf{Effect of sampling period, } C_{Ref}} \end{tikzcd}$$

(Experiment parameters)

- 16,000 power viruses
- Record 100ms power trace
- **15 sampling periods (C_{Ref} count, larger is longer)**
    - **Min: 10 cycles (10MHz)**
    - **Max: 1,000 cycles (100kHz)**
- 10,000 samples
- **Measure the RO counter value**
    - **Once with all power viruses on**
    - **Once with all power viruses off**
- **5 power levels (16000, 8000, 4000, 2000, 1000 power viruses on)**

RO Time Resolution 2

Frequency change of RO ⟷ Effect of sampling period, C_{Ref}

- Calculate the percent change in oscillation frequency
- **Shorter sampling period → less accurate and consistent results**
- Optimal sampling periods depends on the target device's power consumption
  - Smaller power consumption → longer sampling period

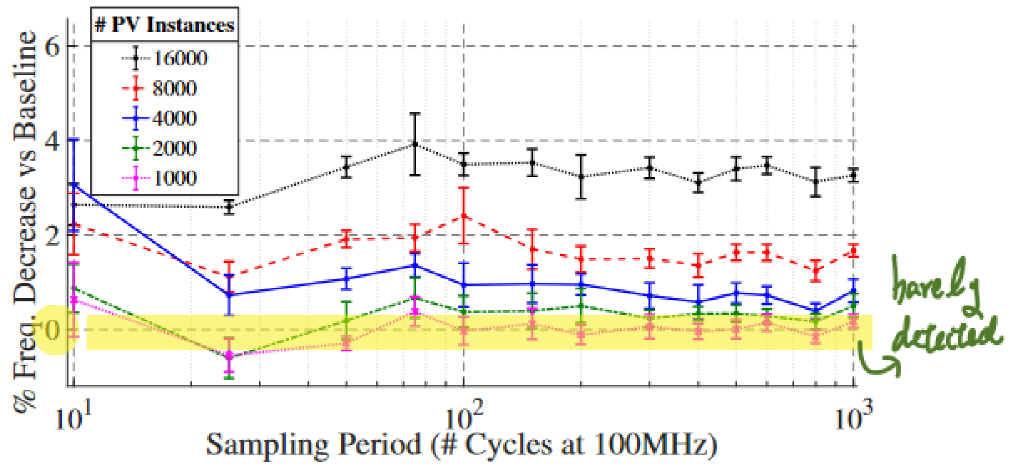


Fig. 7. The percent decrease in the RO frequency and the coefficient of variance as functions of a sampling period. Each line represents a different number of active power virus instances.

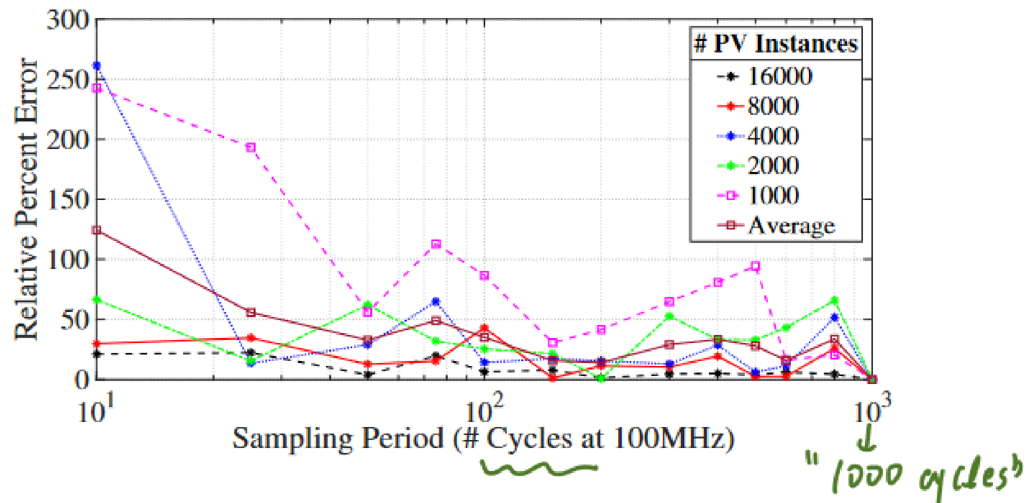
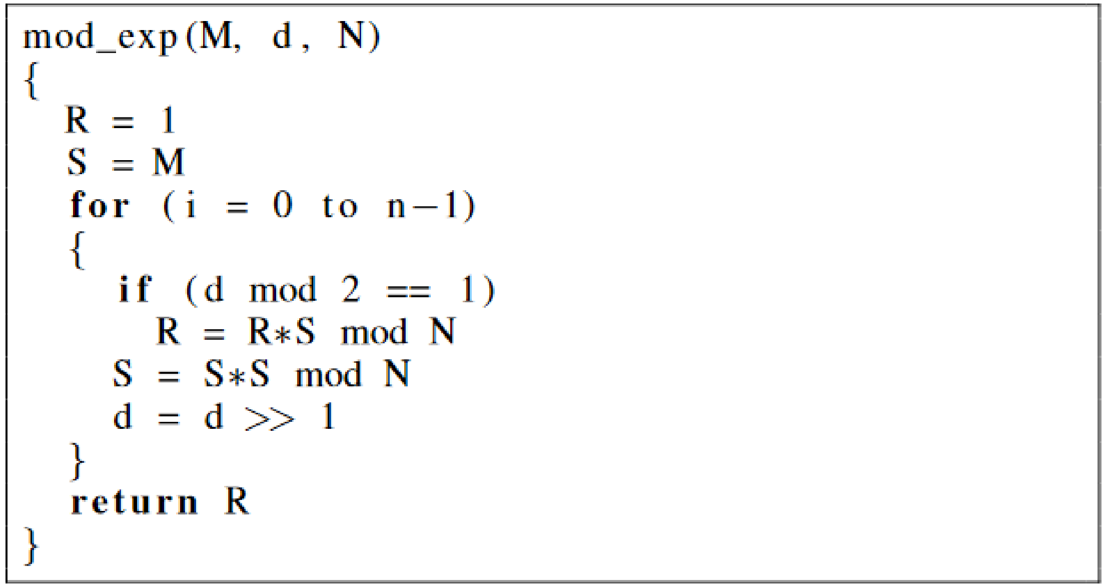


Fig. 8. The percent error in the RO frequency drop compared to the reference case with a 1,000-cycle sampling period.

FPGA-to-FPGA Attack: RSA Accelerator

- Algorithm: square-and-multiply (fast modular exponentiation)
- $R = S^d (mod\ N)$
- If the current exponent bit = 1
    - $S = (S * S)\ (mod\ N)$
    - $R = (R * S)\ (mod\ N)$
- If the current exponent bit = 0
    - $S = (S * S)\ (mod\ N)$
    - $R = (R * 1)\ (mod\ N)$



```
mod_exp(M, d, N)
{
  R = 1
  S = M
  for (i = 0 to n-1)
  {
    if (d mod 2 == 1)
      R = R*S mod N
    S = S*S mod N
    d = d >> 1
  }
  return R
}
```

Listing 1. Pseudo-code for a square-and-multiply modular exponentiation.

RSA Accelerator Hardware Implementation

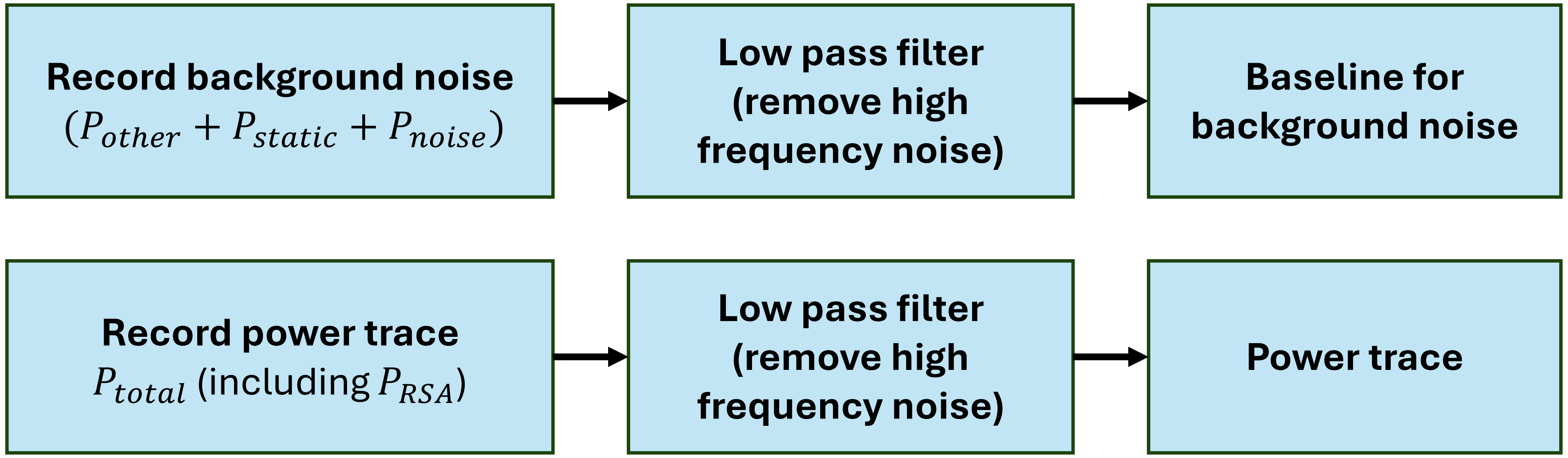
- Two multiplication modules
  - One for: $S = (S * S)\ (mod\ N)$
  - Another for: $R = (R * S)\ (mod\ N)$
- Multiplier: shift-and-add algorithm
  - (mod N) is performed here, by subtracting N in the product
  - The correctly reduced result will be used in the next step
- Throughput is comparable to market-available equivalent cores

Simple Power Analysis Attack 1

- If exponent bit is 1
    - Both multipliers will be active: (R*S) and (S*S)
- If exponent bit is 0
    - Only one multiplier will be active: (S*S)
- **Activity difference between 0 and 1 exponent bit**
- Can be leveraged to do power analysis attack

Simple Power Analysis Attack 2

- $P_{total} = P_{RSA} + (P_{other} + P_{static} + P_{noise})$


Record background noise $(P_{other} + P_{static} + P_{noise})$
Low pass filter (remove high frequency noise)
Baseline for background noise
Record power trace P_{total} (including P_{RSA})
Low pass filter (remove high frequency noise)
Power trace

Simple Power Analysis Attack 3

- Divide the RSA power trace section into 1024 segments
- Calculate the average of power trace over the segments
    - Classify 0 or 1 (more activity = less power monitor output)
    - Automate the attack using MATLAB script

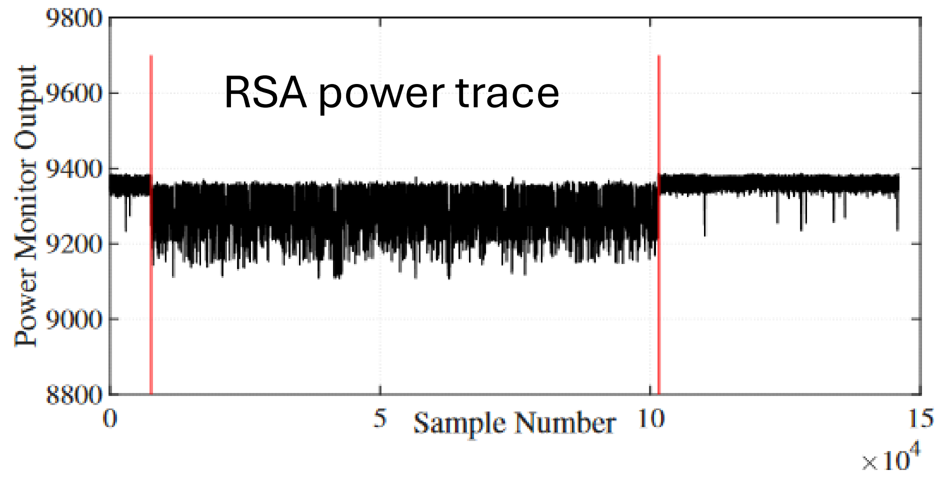


Fig. 10. RSA power trace recorded with the RO-based power monitor. **NOTE:** lower power monitor outputs (i.e. lower oscillation frequencies) correspond to *higher* power consumption.

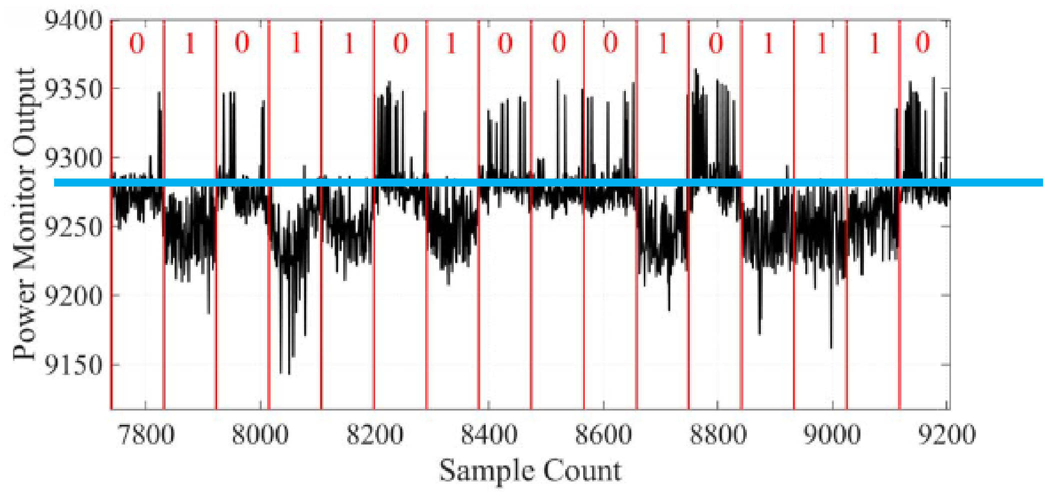
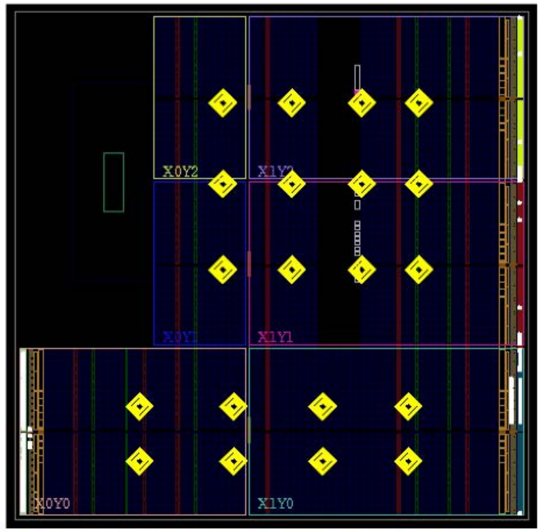


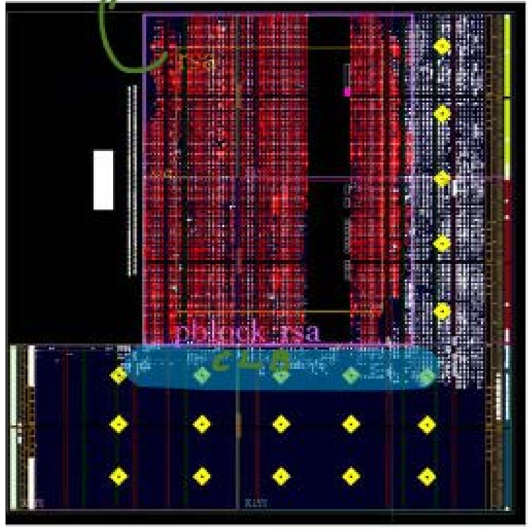
Fig. 11. A zoomed-in view of the RSA power trace showing 16 key bits.

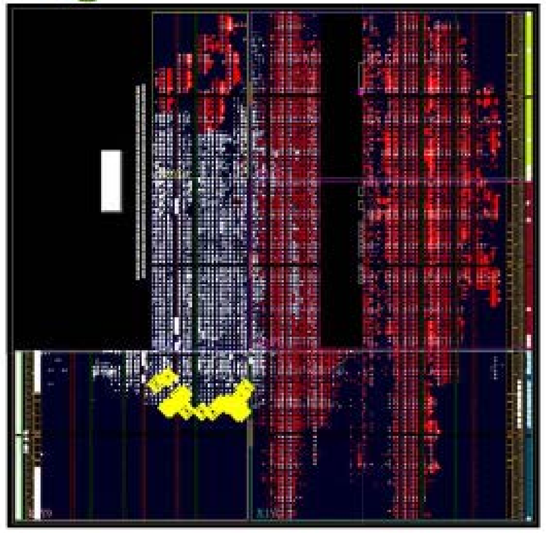
Cases for RO placement

no direct placement on the victim

RO is arbitrarily placed (by the design tool



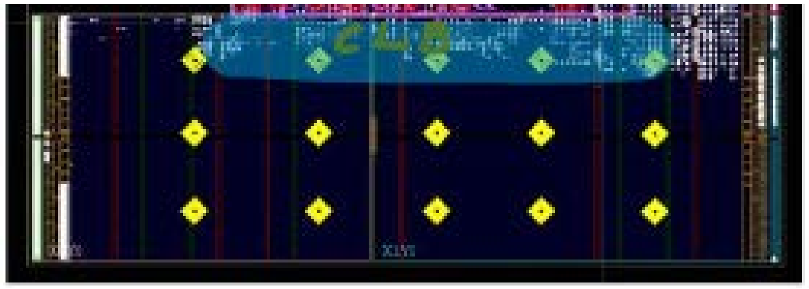


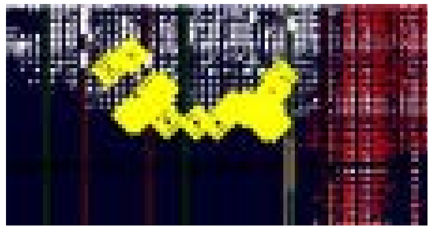


Attacker defined Place and Route (PR)	Physical Isolation (ISO)	No Place and Route (NoPR)
RO placement includes the target's section of FPGA	Separating logic by using a "fence" of CLBs (unused logic)	RO is arbitrarily placed by the design tool

Traces required to recover private key

- **ISO**: Physical separation (with CLB) on the FPGA is not sufficient as a counter measure
- **NoPR**: Restricting user interface to low-level design is not sufficient as a counter measure





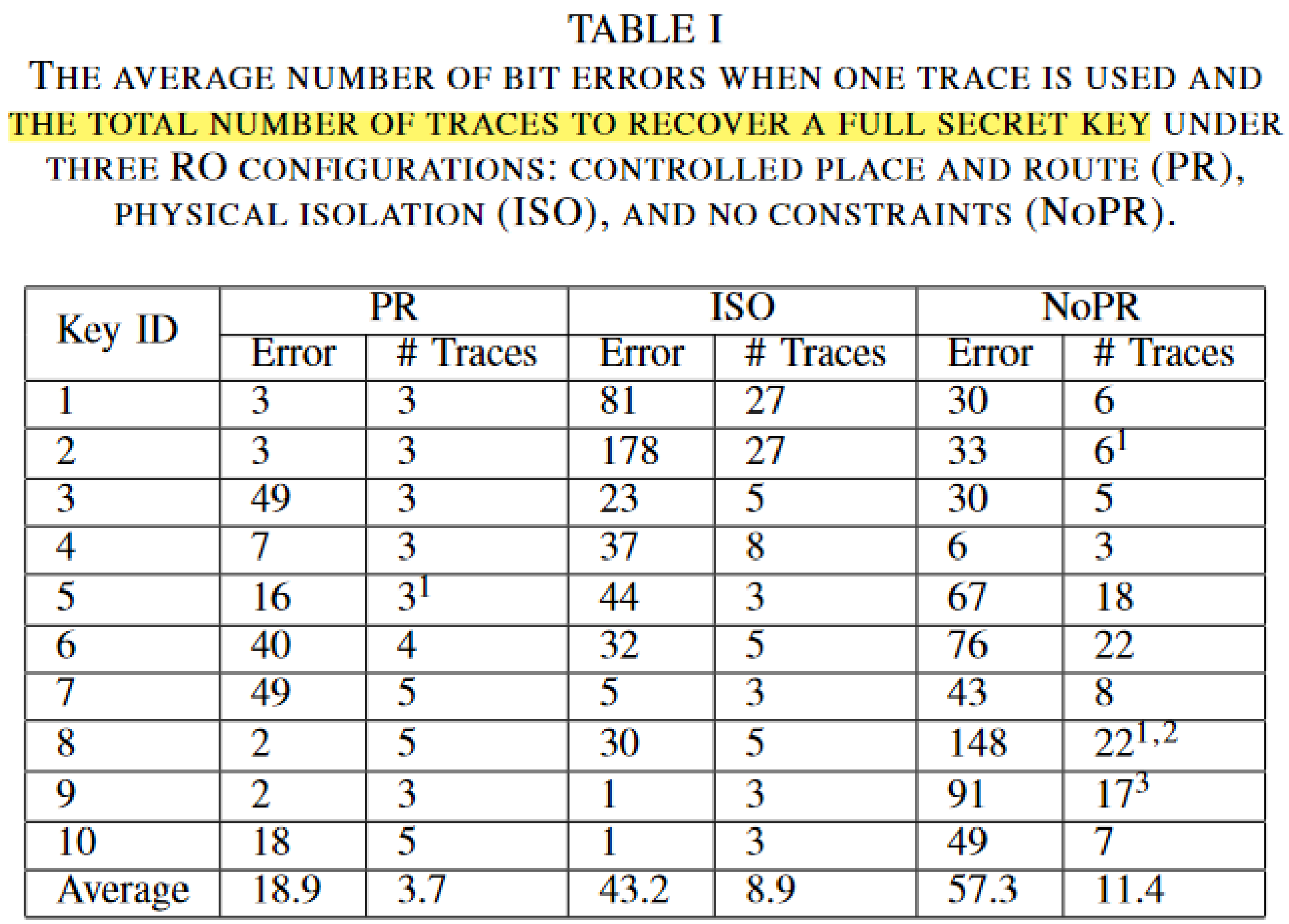


TABLE I
THE AVERAGE NUMBER OF BIT ERRORS WHEN ONE TRACE IS USED AND THE TOTAL NUMBER OF TRACES TO RECOVER A FULL SECRET KEY UNDER THREE RO CONFIGURATIONS: CONTROLLED PLACE AND ROUTE (PR), PHYSICAL ISOLATION (ISO), AND NO CONSTRAINTS (NoPR).

Key ID	PR		ISO		NoPR	
	Error	# Traces	Error	# Traces	Error	# Traces
1	3	3	81	27	30	6
2	3	3	178	27	33	6[1]
3	49	3	23	5	30	5
4	7	3	37	8	6	3
5	16	3[1]	44	3	67	18
6	40	4	32	5	76	22
7	49	5	5	3	43	8
8	2	5	30	5	148	22[1,2]
9	2	3	1	3	91	17[3]
10	18	5	1	3	49	7
Average	18.9	3.7	43.2	8.9	57.3	11.4

Attack but under different condition

- 8192 power viruses
- 32 power levels (0~31)
    - 4096 power viruses (level 16) → similar power consumption of one multiplier unit
- In NoPR RO configuration
- Use the on-chip CPU to randomly set the power level
    - Creating noise
- Random noise can be filtered out by using more traces
- Able to recover the RSA private key by using 31 traces

Thanks for listening

:P