VERNAM LAB
Worcester Polytechnic Institute

WPI

# Root-cause Analysis of the Side Channel Leakage from ASCON Implementations

***Zhenyuan Liu*** *(zliu12@wpi.edu)* ***and Patrick Schaumont***

*Worcester Polytechnic Institute, Worcester, Massachusetts*

Sixth Lightweight Cryptography Workshop (virtual) on June 21-22, 2023.

# Introduction and Motivation

Others work

Side-channel leakage assessment

1. T-test, Chi2
2. Leakage and Countermeasure Testing (yes/no)

Our focus

Root-case analysis of Side-channel leakage

1. Identify Origin of Leakage
2. Test Implementation Properties
3. Leakage and Countermeasure Debugging (how, where)
4. Starting point to design the Countermeasures

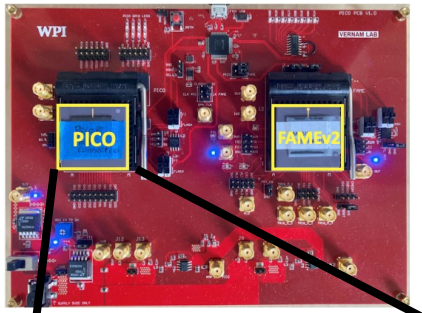
# Target (Device Under Test)

ASCON Co-processor

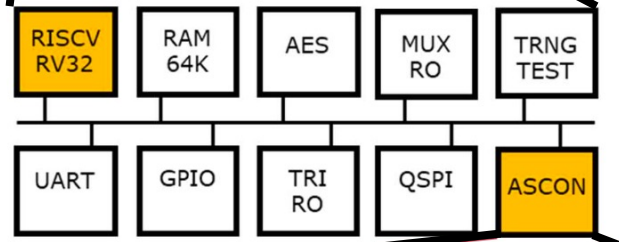
Target 1: a hardware coprocessor with an iterated implementation of ASCON-128 (open-source implementation)

Software ASCON

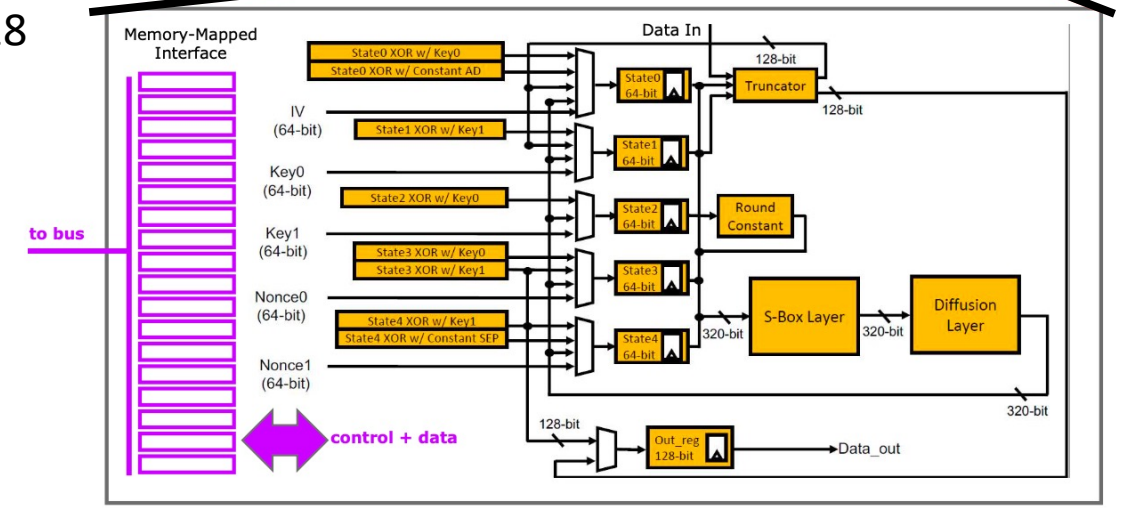
Target 2: a software implementation of ASCON-128 on RISC-V (RV32IMC) (open-source implementation)




Saidoyoki v1 board




Block Diagram of PicoChip (180nm standard cell CMOS chip)

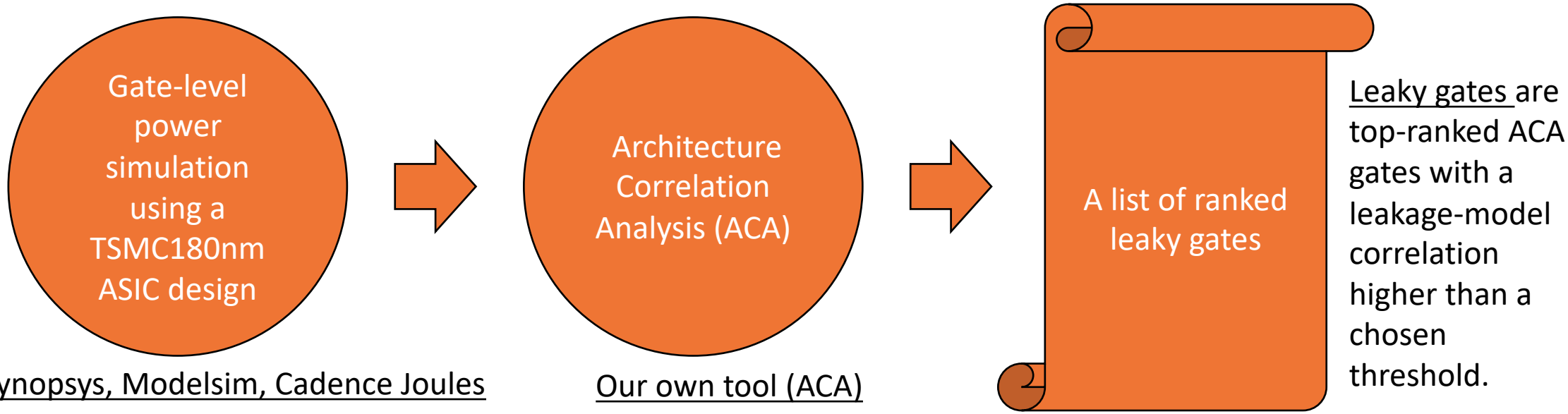



Block Diagram of ASCON Coprocessor

*We are able to perform both simulation (pre-silicon) and measurement (post-silicon) side-channel analysis on the target.*

# Root-cause Analysis Methodology

Gates correspond to standard cells in this paper. There are 57,671 gates in the PicoChip Design.




Leaky gates are top-ranked ACA gates with a leakage-model correlation higher than a chosen threshold.

Tools: Synopsys, Modelsim, Cadence Joules

1. Non-specific root-cause analysis: two groups of test vectors (random vs fixed key).

2. Simulation setup: chip frequency 4 MHz, 4 samples per clock cycle (hardware ASCON), and 1 sample per clock cycle (software ascon).

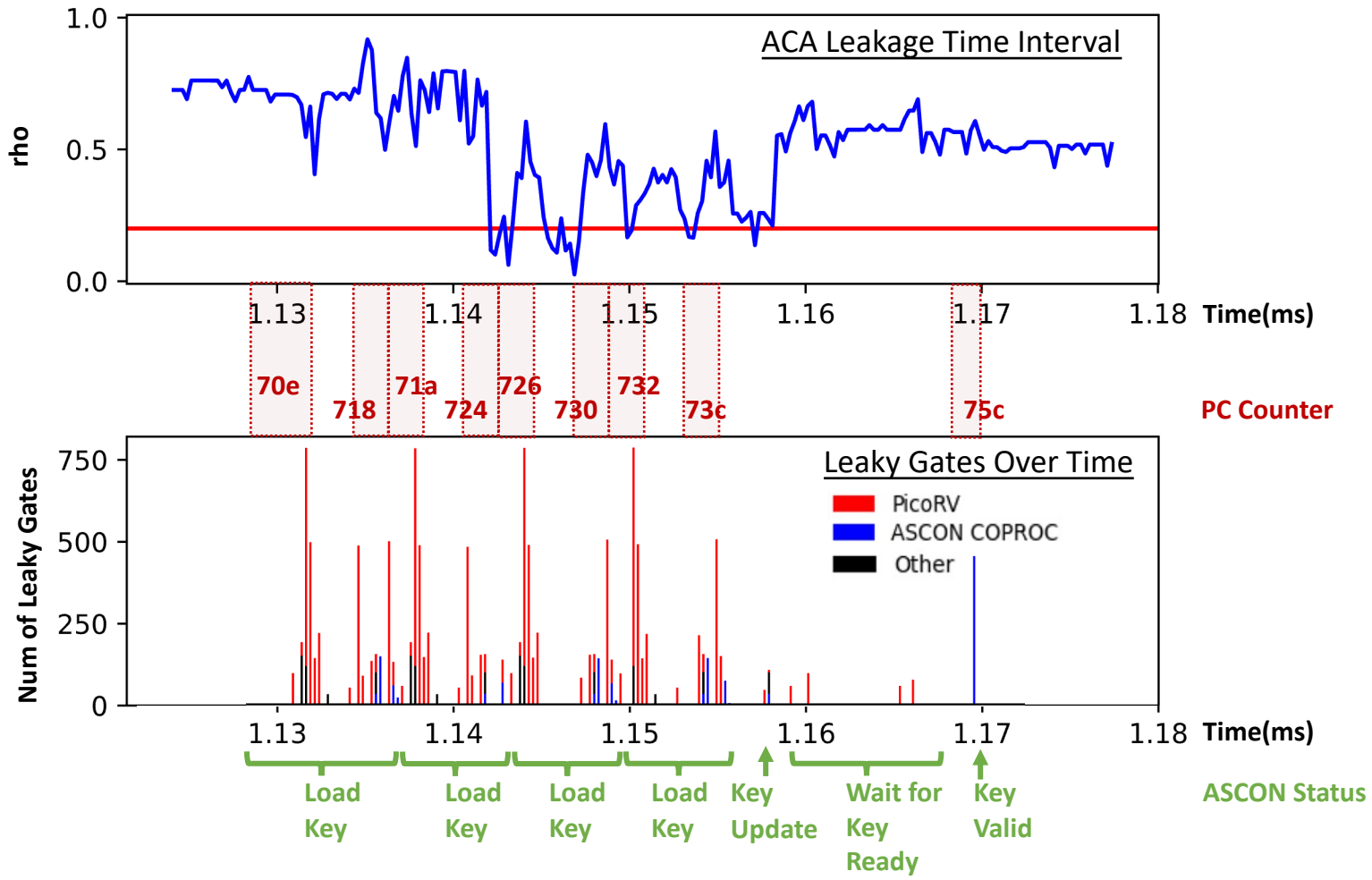
Our own tool (ACA)

1. Identify leaky time interval (LTI) (as leaky time points).

2. Rank the logic gates of the design according to the amount of contributed leakage per gate by computing a leakage impact factor (LIF) for each gate.

3. Identify where these leaky gates coming from and what instructions cause it.

# Root-cause Analysis on ASCON Co-processor (KeyLoad)




Sequence

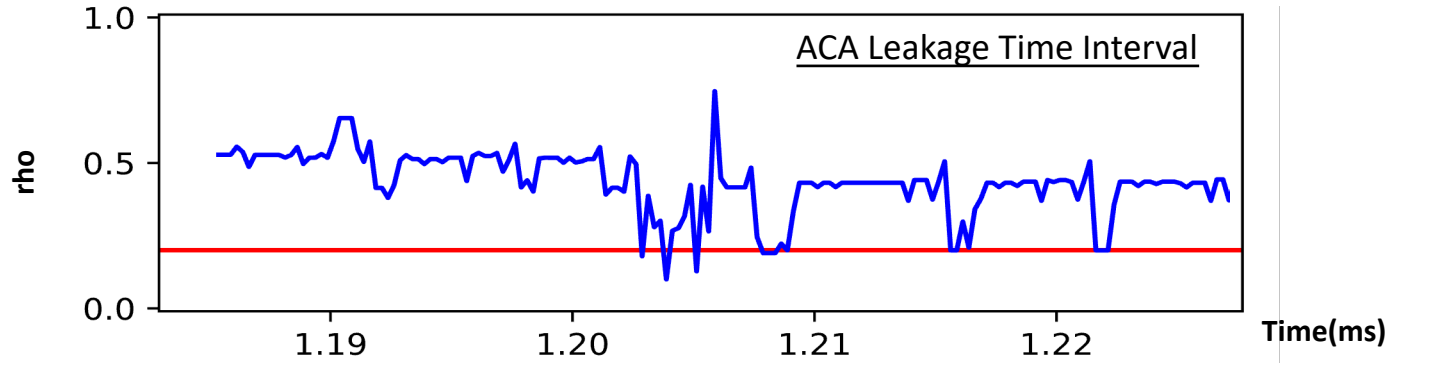
```
70e lw   a4,-228(s0) # load key[0] from RAM
712: lw   a5,-20(s0)
716: addi a5,a5,4
718 sw   a4,0(a5)    # store key[0] to coproc
71a lw   a4,-224(s0) # load key[1] from RAM
71e: lw   a5,-20(s0)
722: addi a5,a5,8
724 sw   a4,0(a5)    # store key[1] to coproc
726 lw   a4,-220(s0) # load key[2] from RAM
72a: lw   a5,-20(s0)
72e: addi a5,a5,12
730 sw   a4,0(a5)    # store key[2] to coproc
732 lw   a4,-216(s0) # load key[3] from RAM
736: lw   a5,-20(s0)
73a: addi a5,a5,16
73c sw   a4,0(a5)    # store key[3] to coproc
73e: lw   a5,-20(s0)
742: li   a4,4
744: sw   a4,0(a5)   # coproc control: key update
746: nop
748: lw   a5,-20(s0)
74c: addi a5,a5,68
750: lw   a5,0(a5)
752: andi a5,a5,1
754: beqz a5,748     # wait for key update ready
756: lw   a5,-20(s0)
75a: li   a4,8
75c sw   a4,0(a5)    # coproc control: key valid
```

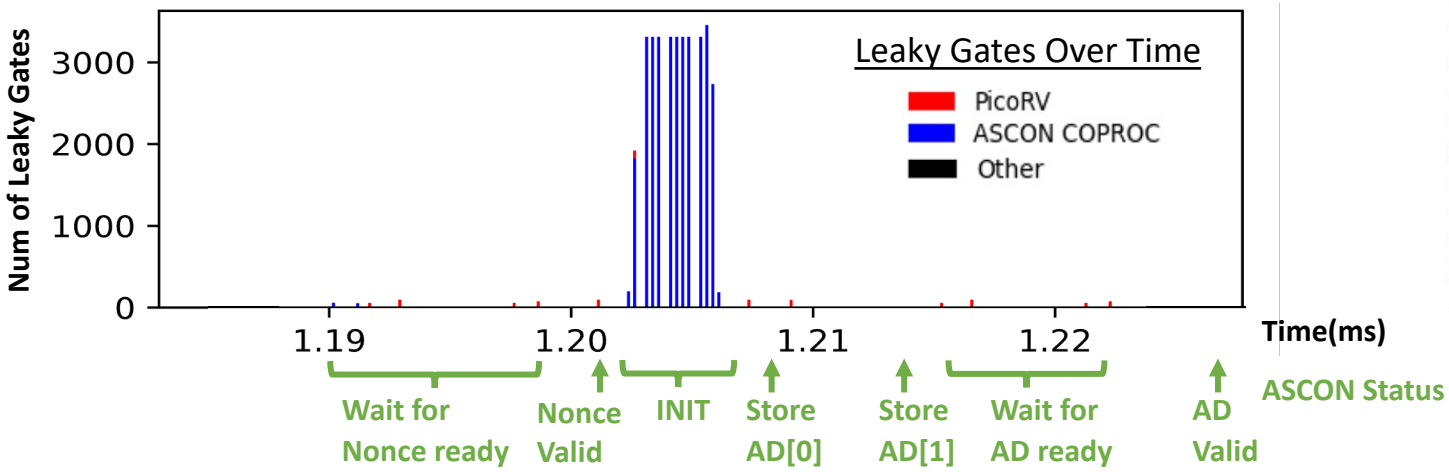


WPI

Gates correspond to standard cells in this paper. There are 57,671 gates in the PicoChip Design.

# Root-cause Analysis on ASCON Co-processor (Initialization)




Sequence

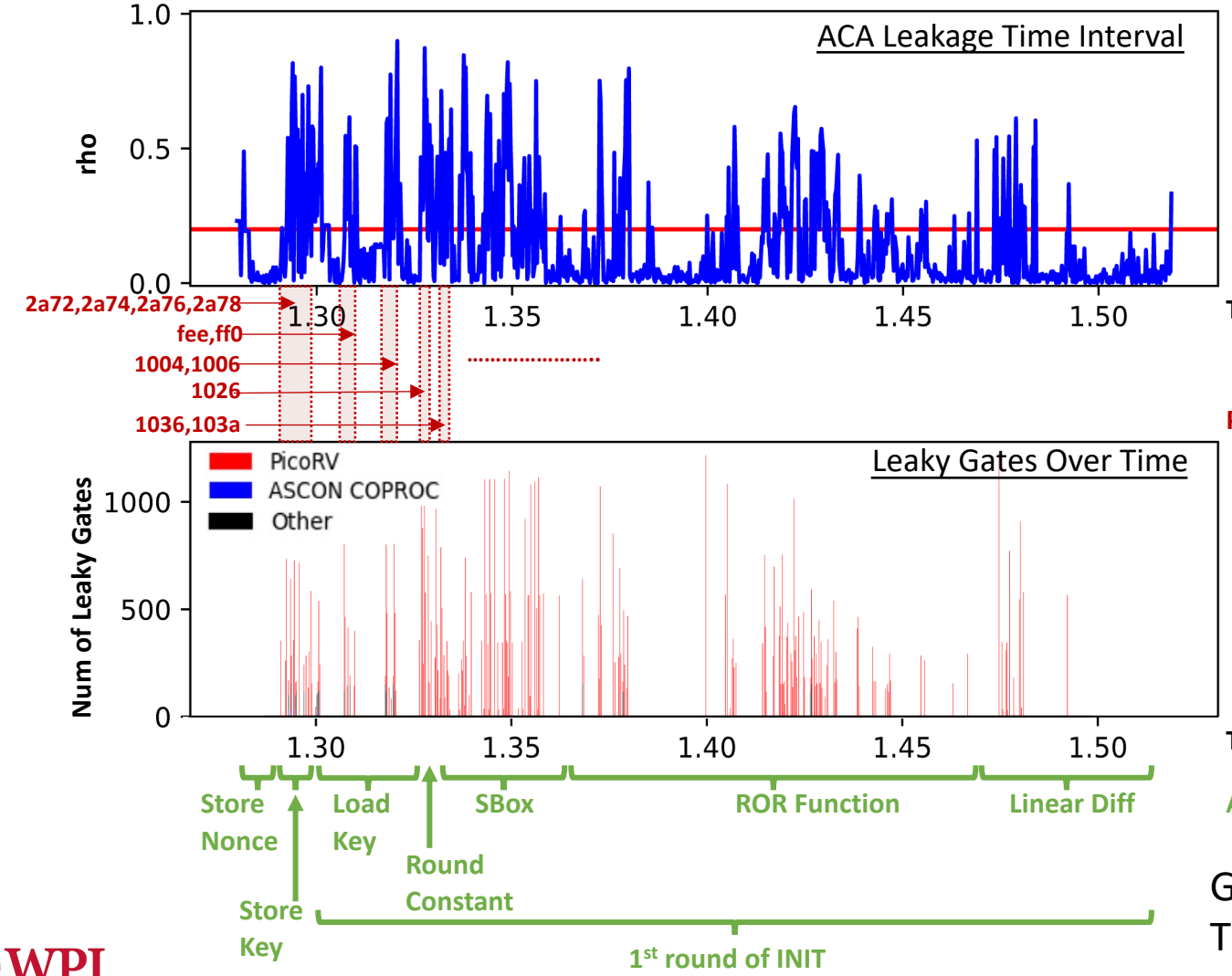
```
792: lw   a5,-20(s0)  # load ascon co-processor base address
796: addi a5,a5,68
79a: lw   a5,0(a5)
79c: andi a5,a5,2
79e: beqz a5,792      # loop wait for coproc nonce ready
7a0: lw   a5,-20(s0)
7a4: li   a4,16
7a6: sw   a4,0(a5)    # send nonce valid
7a8: lw   a5,-20(s0)
7ac: addi a5,a5,36
7b0: lw   a4,-136(s0) # load AD[0] from RAM
7b4: sw   a4,0(a5)    # store AD[0] to coproc

7b6: lw   a5,-20(s0)
7ba: addi a5,a5,40
7be: lw   a4,-132(s0) # load AD[1] from RAM
7c2: sw   a4,0(a5)    # store AD[1] to coproc
7c4: nop
7c6: lw   a5,-20(s0)
7ca: addi a5,a5,68
7ce: lw   a5,0(a5)
7d0: andi a5,a5,4
7d2: beqz a5,7c6      # loop wait for coproc AD ready
7d4: lw   a5,-20(s0)
7d8: lui  a4,0x1
7da: addi a4,a4,32
7de: sw   a4,0(a5)    # coproc control: AD valid | AD last block
```



WPI

Gates correspond to standard cells in this paper. There are 57,671 gates in the PicoChip Design.

# Root-cause Analysis on Software ASCON (NonceLoad + KeyLoad + 1st round of INIT)




Sequence

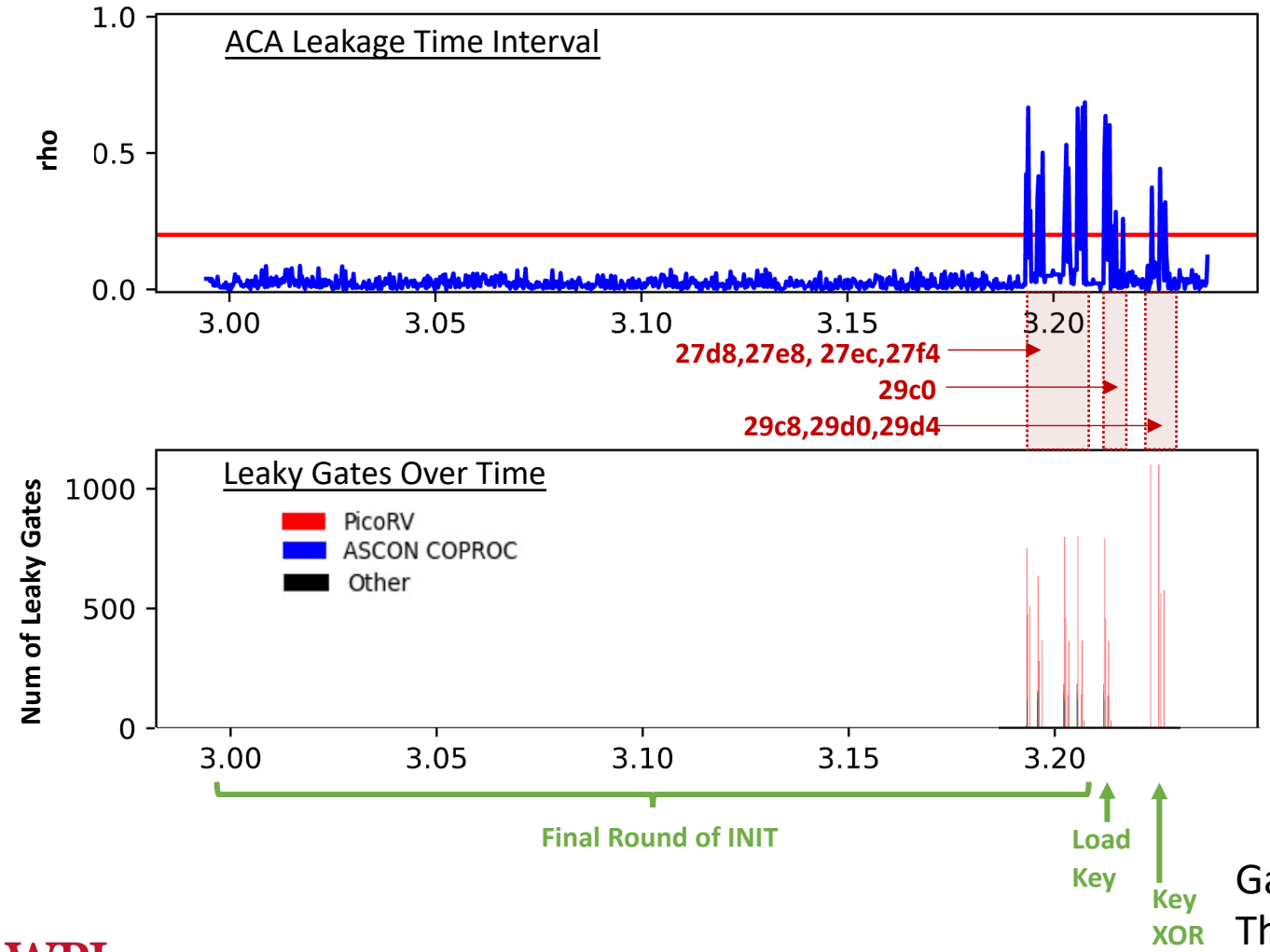
```
2a68: addi a0,sp,88
2a6a: sw s7,112(sp)     # store nonce[0]
2a6c: sw s3,116(sp)     # store nonce[1]
2a6e: sw s11,124(sp)    # store nonce[2]
2a70: sw a1,120(sp)     # store nonce[3]
2a72  sw s4,96(sp)      # store key[0]
2a74  sw s6,100(sp)     # store key[1]
2a76  sw s10,104(sp)    # store key[2]
2a78  sw s0,108(sp)     # store key[3]
2a7a: jal ra,fe4 <P12>

00000fe4 <P12>:
...
fee   lw s1,8(a0)       # load key[0]
ff0   lw ra,12(a0)      # load key[1]
...
1004  lw a5,16(a0)      # load key[2]
1006  lw t4,20(a0)      # load key[3]
...
1026  xori s4,a5,240    # round constant: state_reg[2] XOR with 0xf0
...
1036  not a7,t4         # not key[3]
103a  xori a5,a5,-241   # XORI key[2]
...
1246: jal ra,19c
```

Gates correspond to standard cells in this paper. There are 57,671 gates in the PicoChip Design.



WPI

# Root-cause Analysis on Software ASCON (12th round of INIT + KeyXor)




Time(ms)

PC Counter

Time(ms)

Sequence

```
00000fe4 <P12>:
...
27d8  lw   s0,88(sp)    # load state_reg
...
27e8  lw   s4,72(sp)    # load state_reg
...
27ec  lw   s6,64(sp)    # load state_reg
...
27f4  lw   s10,48(sp)   # load state_reg

27fa: ret

29bc: jal  ra,fe4 <P12>
29c0  lw   t0,124(sp)   # load key
29c2 lw   s3,112(sp)    # load key
29c4 lw   s7,116(sp)    # load key
29c6 lw   s11,120(sp)   # load key
29c8  xor  s8,t0,s0     # xor state_reg and key
29cc xor  s3,s3,s4      # xor state_reg and key
29d0  xor  s7,s7,s6     # xor state_reg and key
29d4  xor  s11,s11,s10  # xor state_reg and key
```

Gates correspond to standard cells in this paper. There are 57,671 gates in the PicoChip Design.



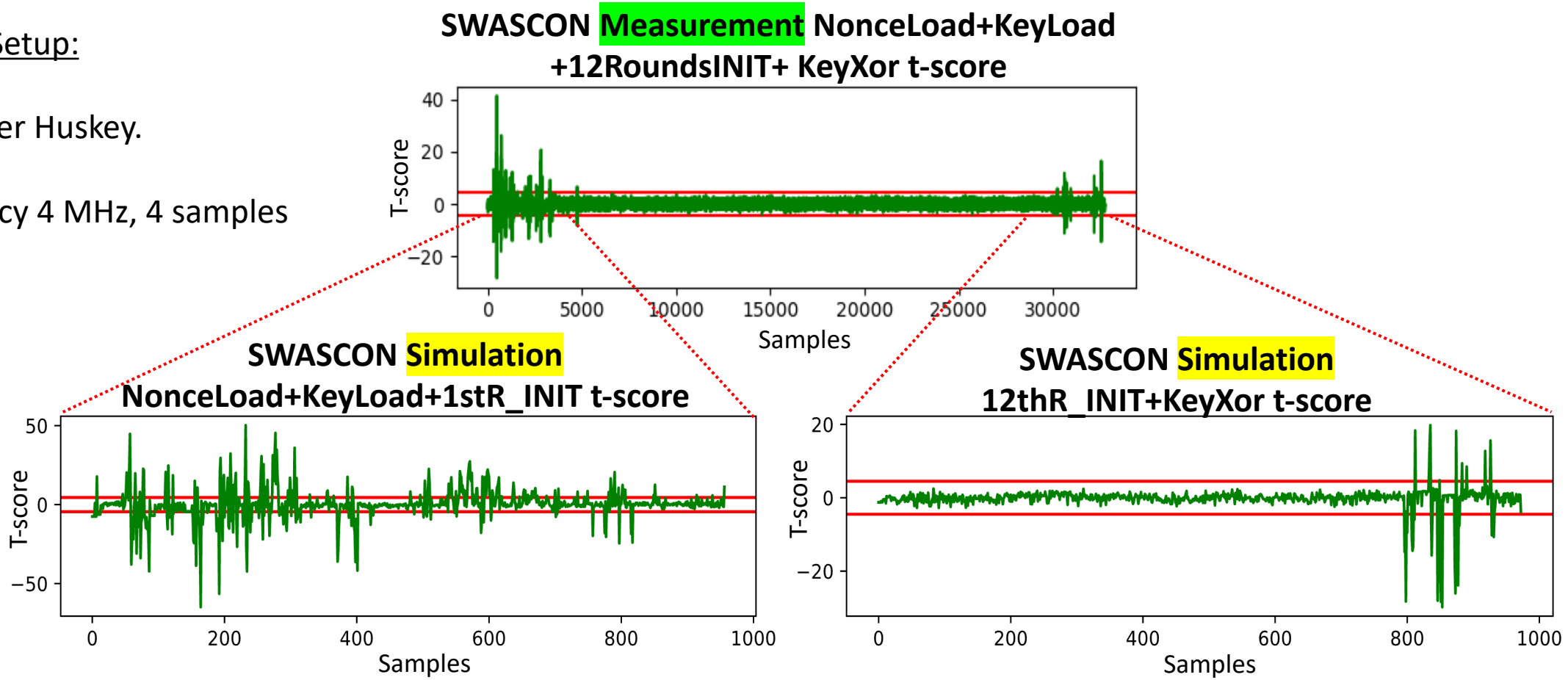
WPI

# Validating Simulation against Measurement (Software ASCON)

Measurement Setup:

1. ChipWhisperer Huskey.

2. Chip frequency 4 MHz, 4 samples per clock cycle.






WPI

# Thank you!
# Please ask me questions for bonus points :)