UNIVERSITY OF CALGARY

IEEE/ACM
2022 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN
ICCAD
41st Edition

Evaluating the Security of eFPGA-based Redaction Algorithms

Amin Rezaei[*], **Raheel Afsharmazayejani**[#], and **Jordan Maynard**[*]

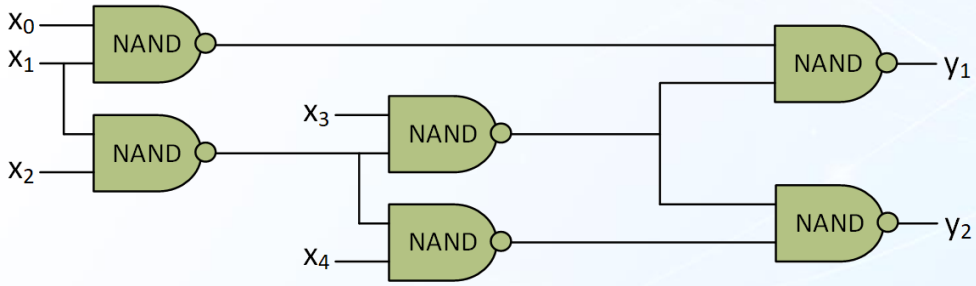
[*]California State University Long Beach, USA

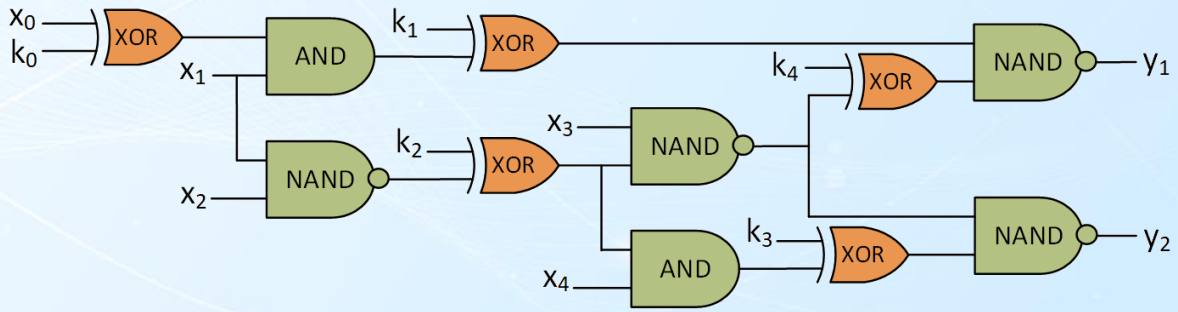
[#]University of Calgary, Canada

Logic Obfuscation

- Hardware IP owners must envision procedures to avoid piracy and overproduction of their designs under a fabless paradigm.
- Traditional logic obfuscation*:

$$\mathbf{k}^* = k_4{}^* k_3{}^* k_2{}^* k_1{}^* k_0{}^* = 01010$$





Original netlist f(x)

$$\exists k^*: g(x, k^*) \equiv f(x)$$

Obfuscated netlist g(x,k)

* J. A. Roy et al., "EPIC: Ending piracy of integrated circuits," In DATE, 2008.

Logic Deobfuscation

- SAT-based attack*
- Attacker model:
  - ✓ Obfuscated netlist
  - ✓ Activated IC
  - ✓ Scan chain access

SAT-based attack

Input: Obfuscated circuit $g(x, k)$ and original function $f(x)$

Output: Correct key k^* such that $g(x, k^*) \equiv f(x)$

while $\hat{x} = SAT(g(x, k_1) \neq g(x, k_2))$ **do**

  $g(x, k_1) = g(x, k_1) \wedge (g(\hat{x}, k_1) = f(\hat{x}));$

  $g(x, k_2) = g(x, k_2) \wedge (g(\hat{x}, k_2) = f(\hat{x}));$

$k^* = SAT(g(x, k_1));$

DIP

- Goal: Each DIP excludes at least one wrong key.
- The truth: Each DIP may exclude many wrong keys from traditional logic obfuscation.

* P. Subramanyan et al., "Evaluating the security of logic encryption algorithms," In HOST, 2015.

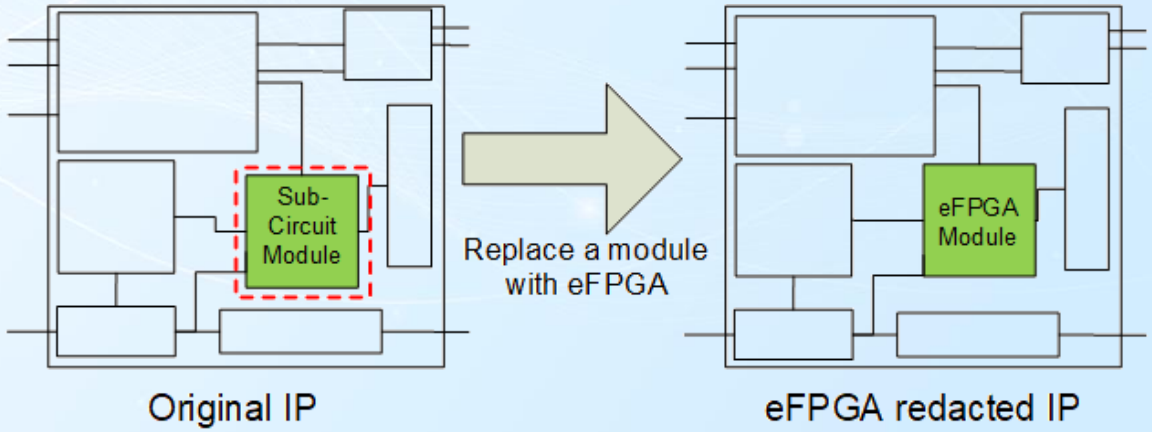
Post-SAT Logic Obfuscation & Deofuscation

- After proposing the SAT-based attack, different defensive mechanisms were proposed using point functions to increase the required number of DIPs exponentially with the key size.
- Two tips:
  - Point function defenses like Anti-SAT* are SAT-hard, but they are vulnerable to approximate SAT-based attacks like AppSAT& that report an approximately correct key with low error.
  - SFLL% as the strongest version of the point function defenses, is both SAT-hard and AppSAT-hard, but it is vulnerable to structural attacks like FAA$ that extract the original netlist from the obfuscated one.

* Y. Xie et al., "Mitigating SAT attack on logic locking," In CHES, 2016.
& K. Shamsi et al., "AppSAT: Approximately deobfuscating integrated circuits," In HOST, 2017
% M. Yasin et al., "Provably-secure logic locking: From theory to practice," In CCS, 2017.
$ D. Sirone et al., "Functional analysis attacks on logic locking," In IEEE Transaction on Information Forensics and Security, 2020.

eFPGA-based Redaction

- A sensitive sub-circuit can be replaced with an embedded FPGA, and the eFPGA can be configured to perform the same functionality as the missing sub-circuit*.
- The configuration bitstream → Hidden key only known to the hardware IP owner.
- Impression: eFPGA-based redaction is “by default” secure against SAT-based attacks due to the complex structure of the embedded FPGA.



* P. Mohan et al., “Hardware redaction via designer-directed fine-grained eFPGA insertion,” In DATE, 2021

How It Started…

- CSAW ‘21 logic locking conquest:
  - Input: A selection of eFPGA fabrics*& with varying complexity and bitstream sizes was given.
  - Task: Recover the bitstream.
  - Result: We were able to find the bitstream of all the 28 benchmarks!

- **Takeaway 1:** The common impression that eFPGA-based redaction is “by default” secure against SAT-based attacks is prejudice.

Bench	Key Size	IcySAT	CycSAT
K3N2	601	155s, Correct Key	3.79s, Correct Key
K3N3	725	343s, Correct Key	5.79s, Correct Key
K3N4	837	798s, Correct Key	9.48s, Correct Key
K3N5	941	6 hours, No result	9.15s, Correct Key
K3N6	1154	6 hours, No result	19.1s, Correct Key
K3N7	1243	6 hours, No result	25.32s, Correct Key
K3N8	1393	6 hours, No result	29.24s, Correct Key
K4N2	639	195s, Correct Key	3.12s, Correct Key
K4N3	810	3178s, Correct Key	5.47s, UNSAT
K4N4	1049	6 hours, No result	7.22s, Correct Key
K4N5	1316	6 hours, No result	21.28s, Correct Key
K4N6	1468	6 hours, No result	20.59s, Correct Key
K4N7	1647	6 hours, No result	29.66s, Correct Key
K4N8	1849	6 hours, No result	31.93s, Correct Key
K5N2	815	6 hours, No result	5.68s, Correct Key
K5N3	1066	6 hours, No result	7.9s, Correct Key
K5N4	1477	6 hours, No result	17.18s, Correct Key
K5N5	1741	6 hours, No result	39.91s, Correct Key
K5N6	2012	6 hours, No result	35.57s, Correct Key
K5N7	2271	6 hours, No result	56.24s, Correct Key
K5N8	2573	6 hours, No result	74.89s, Correct Key
K6N2	1125	6 hours, No result	6.69s, Correct Key
K6N3	1518	6 hours, No result	13.29s, Correct Key
K6N4	2089	6 hours, No result	2.12s, Correct Key
K6N5	2694	6 hours, No result	32.193s, Correct Key
K6N6	2928	6 hours, No result	60.52s, Correct Key
K6N7	3347	6 hours, No result	73.47s, Correct Key
K6N8	3989	6 hours, No result	492.3s, Correct Key

* J. Bhandari et al., “Exploring eFPGA-based redaction for IP protection,” In ICCAD, 2021
& J. Bhandari et al., “Not all fabrics are created equal: Exploring eFPGA parameters for IP redaction” In arXiv, 2021.

Preliminary Result Analysis

- CycSAT* can recover the bitstream in significantly less time than IcySAT&.
  - ➢ IcySAT → Tries to unroll all the cycles
  - ➢ CycSAT → Tries to break all the cycles (but it cannot!)
- CycSAT cannot break "hard" cycles.
  - ➢ Why? "The edges in a directed graph cannot always be separated into two disjoint sets, such that each simple cycle is generated by two simple pathways, one made up of edges from each set.%"

Bench	Key Size	IcySAT	CycSAT
K3N2	601	155s, Correct Key	3.79s, Correct Key
K3N3	725	343s, Correct Key	5.79s, Correct Key
K3N4	837	798s, Correct Key	9.48s, Correct Key
K3N5	941	6 hours, No result	9.15s, Correct Key
K3N6	1154	6 hours, No result	19.1s, Correct Key
K3N7	1243	6 hours, No result	25.32s, Correct Key
K3N8	1393	6 hours, No result	29.24s, Correct Key
K4N2	639	195s, Correct Key	3.12s, Correct Key
K4N3	810	3178s, Correct Key	5.47s, UNSAT
K4N4	1049	6 hours, No result	7.22s, Correct Key
K4N5	1316	6 hours, No result	21.28s, Correct Key
K4N6	1468	6 hours, No result	20.59s, Correct Key
K4N7	1647	6 hours, No result	29.66s, Correct Key
K4N8	1849	6 hours, No result	31.93s, Correct Key
K5N2	815	6 hours, No result	5.68s, Correct Key
K5N3	1066	6 hours, No result	7.9s, Correct Key
K5N4	1477	6 hours, No result	17.18s, Correct Key
K5N5	1741	6 hours, No result	39.91s, Correct Key
K5N6	2012	6 hours, No result	35.57s, Correct Key
K5N7	2271	6 hours, No result	56.24s, Correct Key
K5N8	2573	6 hours, No result	74.89s, Correct Key
K6N2	1125	6 hours, No result	6.69s, Correct Key
K6N3	1518	6 hours, No result	13.29s, Correct Key
K6N4	2089	6 hours, No result	2.12s, Correct Key
K6N5	2694	6 hours, No result	32.193s, Correct Key
K6N6	2928	6 hours, No result	60.52s, Correct Key
K6N7	3347	6 hours, No result	73.47s, Correct Key
K6N8	3989	6 hours, No result	492.3s, Correct Key

* H. Zhou et al., "CycSAT: SAT-based attack on cyclic logic encryptions," In ICCAD, 2017.
& K. Shamsi et al., "IcySAT: Improved SAT-based attacks on cyclic locked circuits," In ICCAD, 2019.
% R. Chen et al., "Statistical timing verification for transparently latched circuits," In IEEE Tran. on Com.-Aided Design of Integrated Cir. and Sys., 2006.

DIP Exclusion Attack

- Idea: Record the DIPs that CycSAT got stuck on and exclude them structurally from the eFPGA-based redaction benchmarks.
- Approach: If a loop is detected, add a new component to the obfuscated netlist to avoid the loop.

DIP exclusion attack

Input: Obfuscated circuit $g(x, k)$ and original function $f(x)$
Output: Key vector k^* such that $g(x, k^*) \equiv f(x)$
$W \leftarrow SearchFeedback(g(x, k))$;
// $W \leftarrow \{w_0, w_1, ..., w_m\}$
for *all* $w_i \in W$ **do**
  $F(w_i, w_i') \leftarrow No\_Structural\_Path(w_i)$;
$NC(k) \leftarrow \wedge_{i=0}^{m} F(w_i, w_i')$;
$g(x, k) \leftarrow g(x, k) \wedge NC(k)$;
$DIP_{set} \leftarrow \{\}$;
while $\hat{x} \leftarrow SAT(g(x, k_1) \neq g(x, k_2))$ **do**
  if $LoopDetected(\hat{x}, DIP_{set})$ **then**
    $g(x, k_1) \leftarrow Exclude(\hat{x}, g(x, k_1))$;
    $g(x, k_2) \leftarrow Exclude(\hat{x}, g(x, k_2))$;
  else
    $g(x, k_1) \leftarrow g(x, k_1) \wedge (g(\hat{x}, k_1) = f(\hat{x}))$;
    $g(x, k_2) \leftarrow g(x, k_2) \wedge (g(\hat{x}, k_2) = f(\hat{x}))$;
    $Add(\hat{x}, DIP_{set})$;
    // DIPs are kept in DIP_{set}
Return $k^* \leftarrow SAT(g(x, k_1))$;

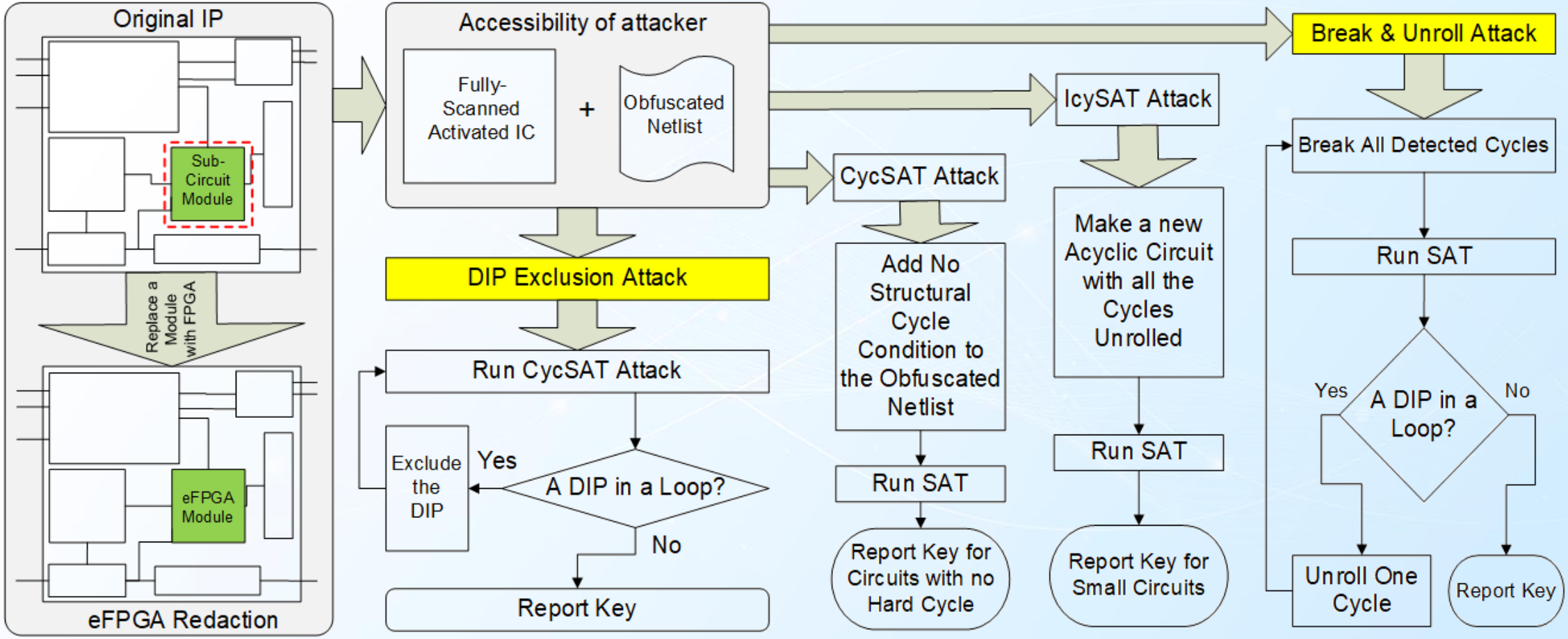
Break & Unroll Attack

- Idea: Why not have the best of both worlds? (i.e., Breaking + Unrolling)
- Approach: Break all the cycles; then if a loop is detected, create a new obfuscated circuit by unrolling one cycle and redo.

Break & Unroll attack

Input: Obfuscated circuit $g(x,k)$ and original function $f(x)$
Output: Key vector k^* such that $g(x,k^*) \equiv f(x)$
while *(True)* **do**
  $W \leftarrow SearchFeedbackSignals(g(x,k))$;
  // $W \leftarrow \{w_0, w_1, ..., w_m\}$
  for *all* $w_i \in W$ **do**
    $F(w_i, w_i') \leftarrow BreakFeedback(w_i)$;
  $NoCyclicCNF(k) \leftarrow \wedge_{i=0}^{m} F(w_i, w_i')$;
  // $NoCyclicCNF(k) \leftarrow BreakFeedback(w_0) \wedge ... \wedge BreakFeedback(w_m)$
  $g(x,k) \leftarrow g(x,k) \wedge NoCyclicCNF(k)$;
  $DIP_{set} \leftarrow \{\}$;
  while $\hat{x} \leftarrow SAT(g(x,k_1) \neq g(x,k_2))$ **do**
    if $LoopDetected(\hat{x}, DIP_{set})$ **then**
      $w \leftarrow$ Select a feedback signal from W set;
      $g(x,k) \leftarrow NewCircuit(w, g(x,k))$;
      Break;
    $g(x,k_1) \leftarrow g(x,k_1) \wedge (g(\hat{x},k_1) = f(\hat{x}))$;
    $g(x,k_2) \leftarrow g(x,k_2) \wedge (g(\hat{x},k_2) = f(\hat{x}))$;
    $Add(\hat{x}, DIP_{set})$;
    // DIPs are kept in DIP_{set}
  if $!SAT(g(x,k_1) \neq g(x,k_2))$ **then**
    Return $k^* \leftarrow SAT(g(x,k_1))$;

Attacks on eFPGA-based Redaction



Time Complexity

- DIP Exclusion: If an exponential number of DIPs lead to a stateful situation, the worst-case time complexity is exponential.
- Break & Unroll: If no hard cycles exist, the best-case time complexity is the same as CycSAT while if an exponential number of hard cycles exist, the worst-case time complexity is the same as IcySAT.
  - Note: In any scenario where CycSAT can return the correct key, Break & Unroll has the same run-time while in cases where CycSAT returns an infinite loop, Break & Unroll can return the correct key with a much better average run-time than IcySAT.

Experimental Results

- Setup: Add hard cycles to eFPGA-based redaction benchmarks. Each hard cycle embeds four extra key bit controlled feedbacks into the circuit.

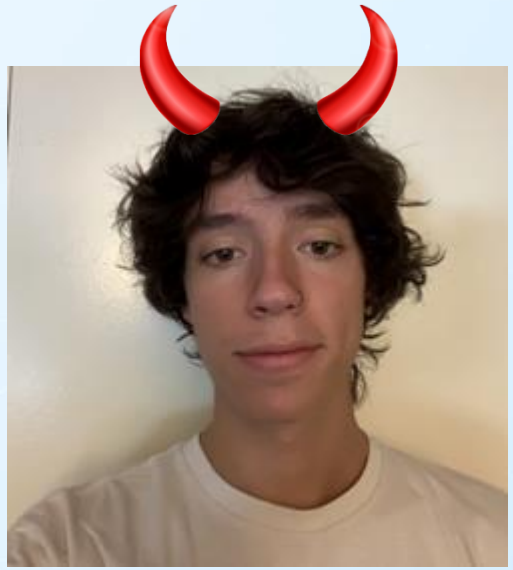
Bench	Key Size	Hard Cycles	CycSAT	IcySAT	DIP Exclusion	Break & Unroll
K3N2	641	10	Inf Loop	697s, Correct Key	1236.57s , Correct Key	9.46s, Correct Key
K3N3	765	10	Inf Loop	1817s, Correct Key	2163.11s , Correct Key	16.38s, Correct Key
K3N4	877	10	Inf Loop	5506s, Correct Key	5621.3s , Correct Key	41.81s , Correct Key
K3N5	981	10	Inf Loop	6 hours, No Result	7011.37s , Correct Key	36.42s , Correct Key
K3N6	1194	10	Inf Loop	6 hours, No Result	6 hours, No Result	141.34s , Correct Key
K3N7	1283	10	Inf Loop	6 hours, No Result	24011.02s, Correct Key	205.09s , Correct Key
K3N8	1433	10	Inf Loop	6 hours, No Result	6 hours, No Result	410.7s, Correct Key
K4N2	895	64	Inf Loop	1072s, Correct Key	6 hours, No Result	20.93s , Correct Key
K4N3	1134	81	UNSAT	26695s, Correct Key	6 hours, No Result	182.15s, UNSAT
K4N4	1469	105	Inf Loop	6 hours, No Result	6 hours, No Result	194.6s, Correct Key
K4N5	1844	132	Inf Loop	6 hours, No Result	6 hours, No Result	806.14s, Correct Key
K4N6	2056	147	Inf Loop	6 hours, No Result	6 hours, No Result	1519.89s, Correct Key
K4N7	2307	165	Inf Loop	6 hours, No Result	6 hours, No Result	1602.52s, Correct Key
K4N8	2589	185	Inf Loop	6 hours, No Result	6 hours, No Result	6 hours, No Result

How It's Going…

- If we replace large parts of the design with an eFPGA, we cannot guarantee the same performance as the original design. So, eFPGA-based redaction is only meaningful when a small-sized sub-circuit is replaced with an eFPGA.
- Provisions to insert hard cycles (against CycSAT) and large-size keys (against IcySAT) are required. Even with these provisions, eFPGA-based redaction can be challenged by our proposed attacks.
- **Takeaway 2:** Additional research on how to systematically create an exponential number of non-combinational hard cycles is required to produce secure eFPGA-based redaction schemes.

Amin Rezaei

Raheel Afsharmazayejani



Jordan Maynard

Computer Architecture, Reliability, and Security Laboratory (CARS-Lab)

California State University Long Beach