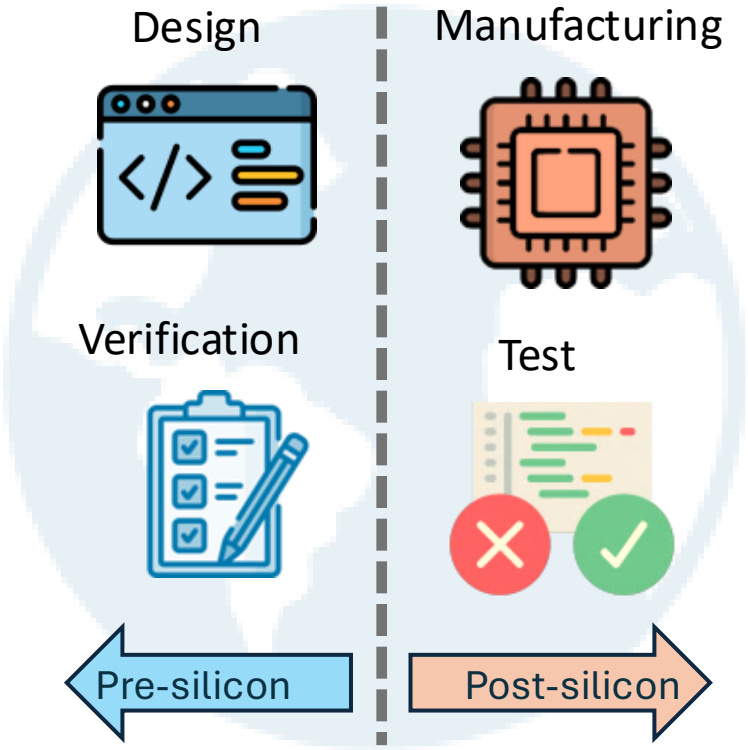
WASHINGTON STATE UNIVERSITY

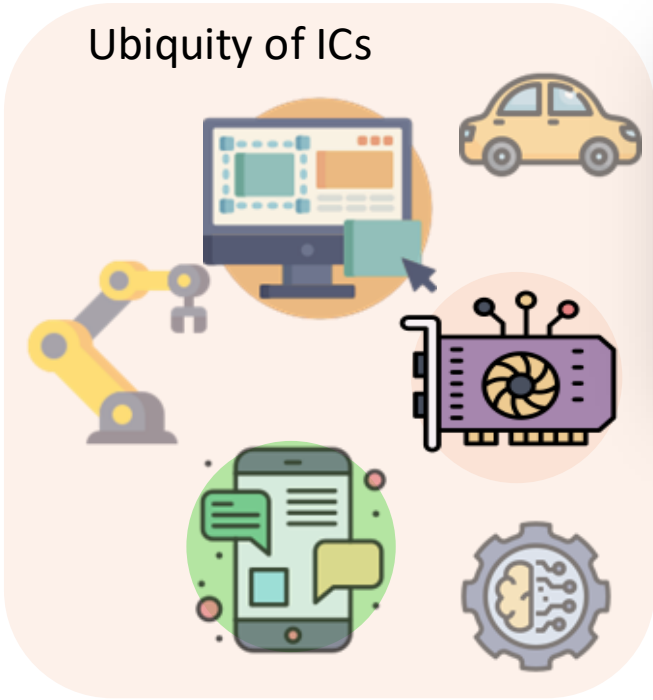
Hardware IP Protection & Reliability Solutions in a Fully Transparent World

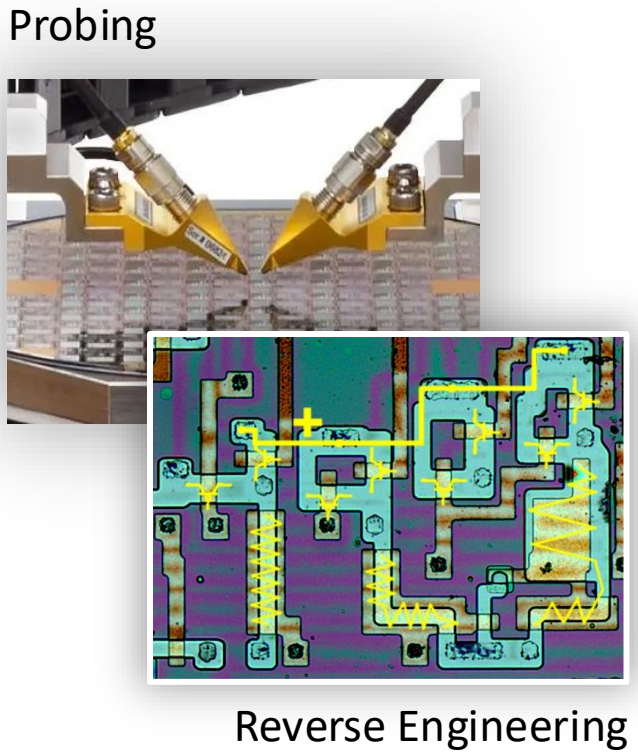
Leon Li

Washington State University

Rise of Hardware IP Exposure







Increased exposure from **globalized supply chain**

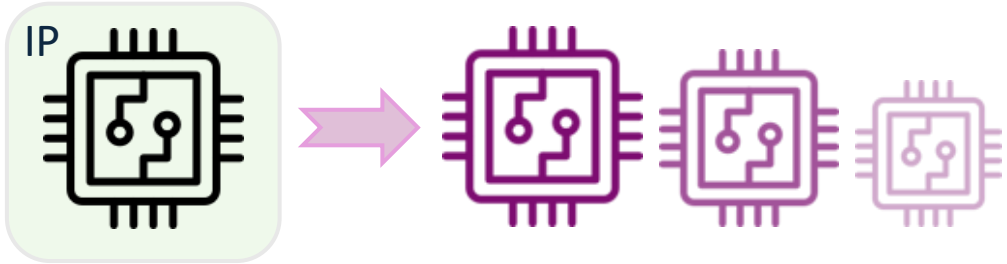
Expanded **attack incentives and capabilities**

Threat Landscape

Attcker's Objective

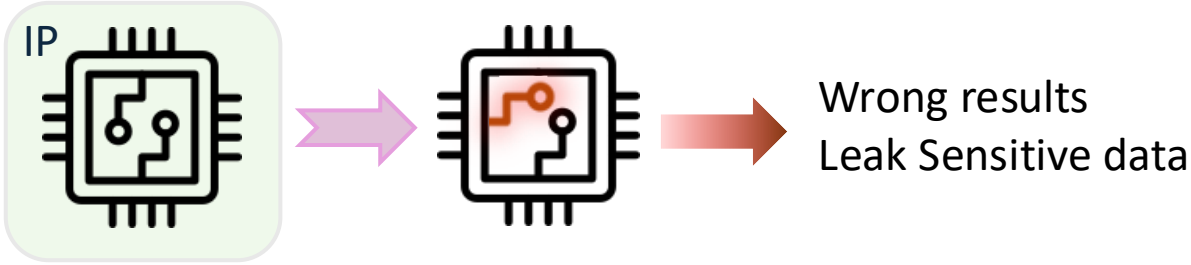
Counterfeiting & piracy

Steal or reproduce the design



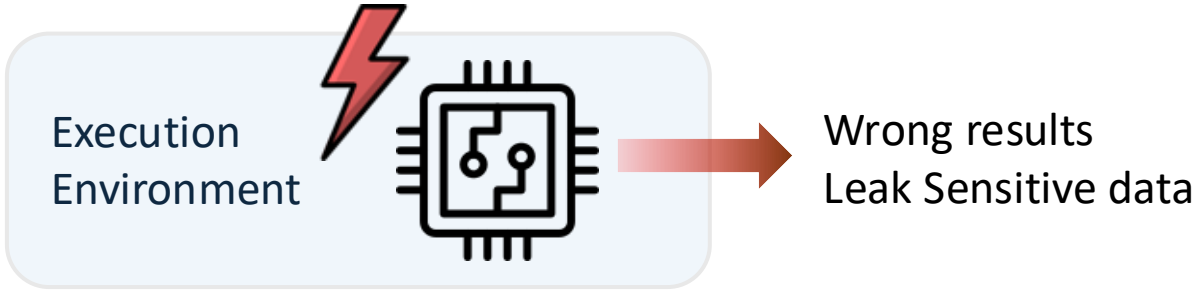
Malicious modification

Alter functionality or add backdoors



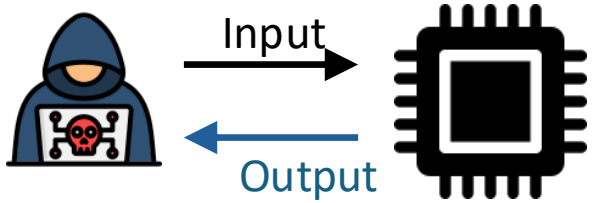
Fault injection

Disrupt behavior or bypass protections



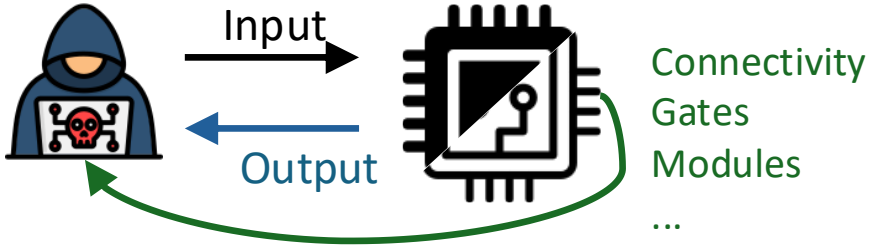
Threat Landscape

Attacker's Visibility



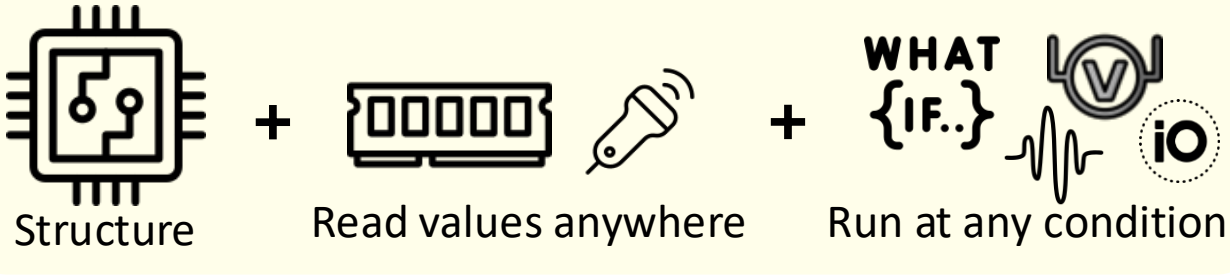
Black-box

Only I/O access, no internal view



Gray-box

Partial internal visibility



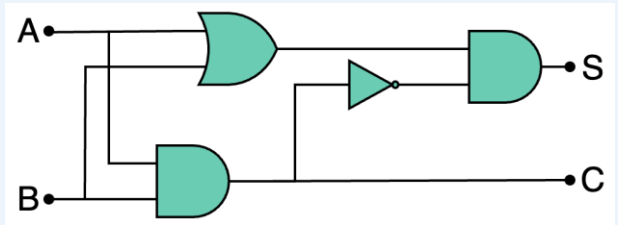
White-box

Full, transparent structural access

Classic Defense Techniques

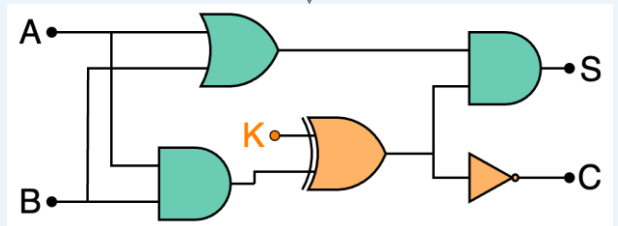
Logic Locking

Original Netlist



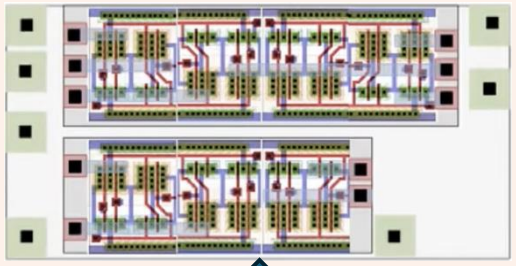
Locking

Locked Netlist

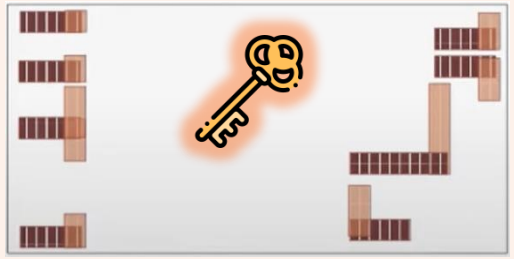


Split Manufacturing

Front End Of Line (FEOL)

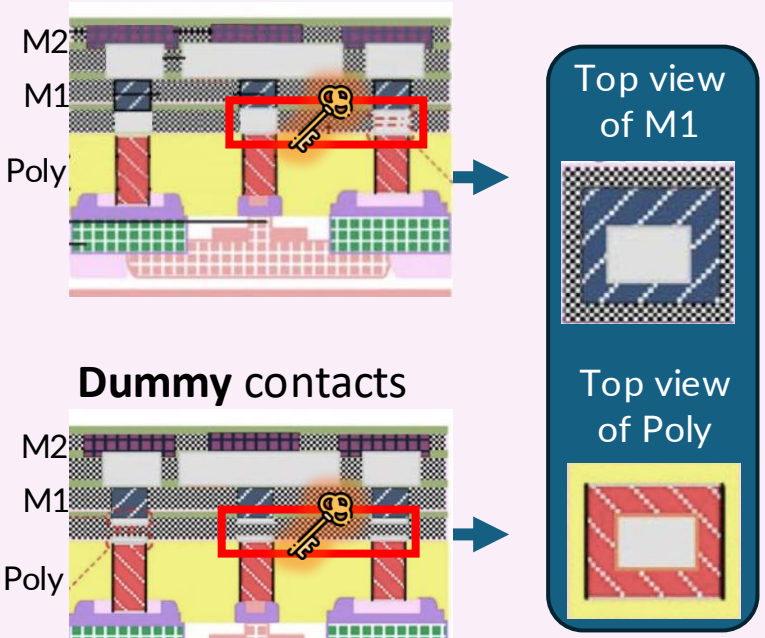


Back End Of Line (BEOL)



IC Camouflaging

True contacts (side)



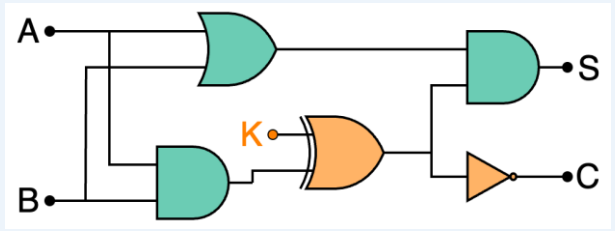
Dummy contacts

Classic Defense Techniques

Deny the attacker the portion of the design
*that is **semantically critical** at a certain visibility level.*

Logic Locking

Locked Netlist

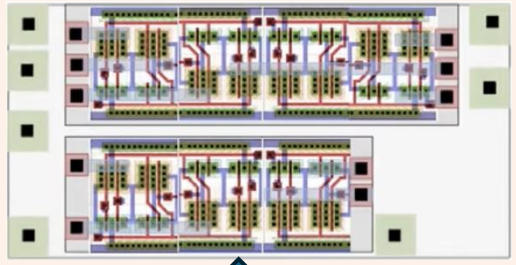


Lack of **key**

Netlist Attacker: IP buyer + More

Split Manufacturing

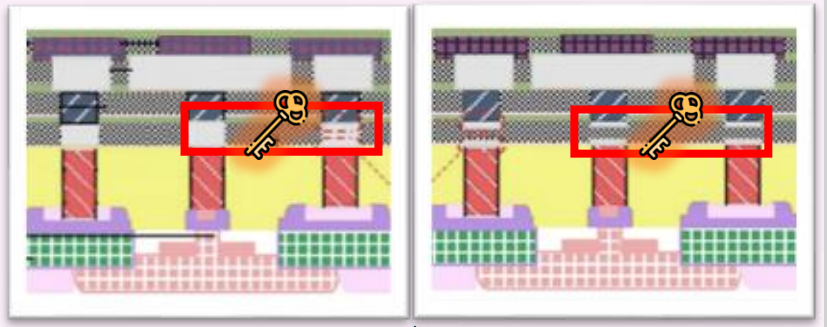
Front End Of Line (FEOL)



Lack of **routing**

Layout Attacker: Foundry

IC Camouflaging



Lack of **layout patterns**

Chip Attacker: Reverse Engineer

WASHINGTON STATE UNIVERSITY

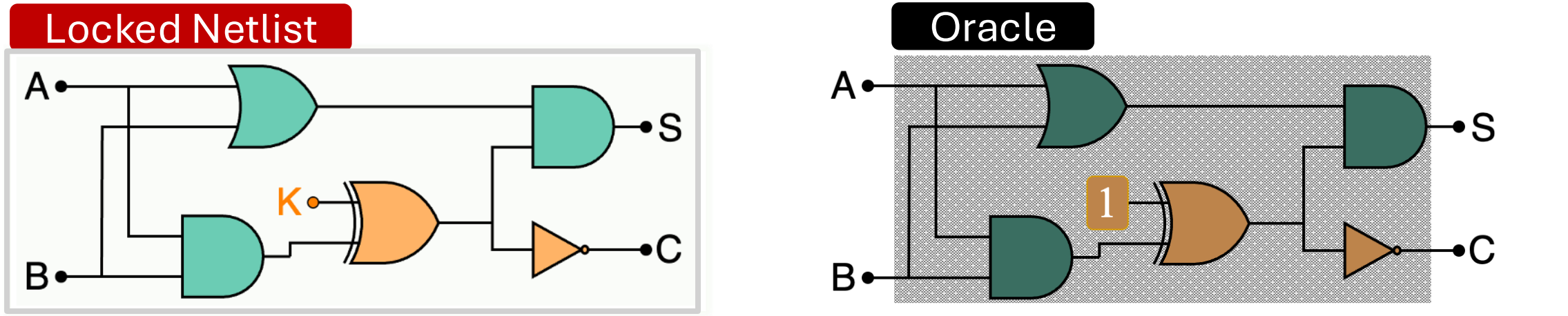
T1

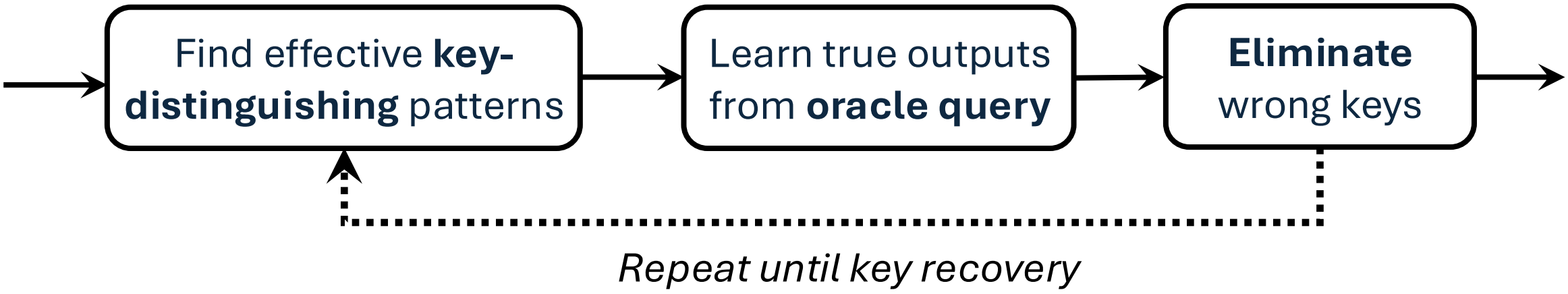
Structural Vulnerability Analysis

Oracle-less backdoor to logic locking

DATE'19, VTS'19, TCAD'22, D&T'25

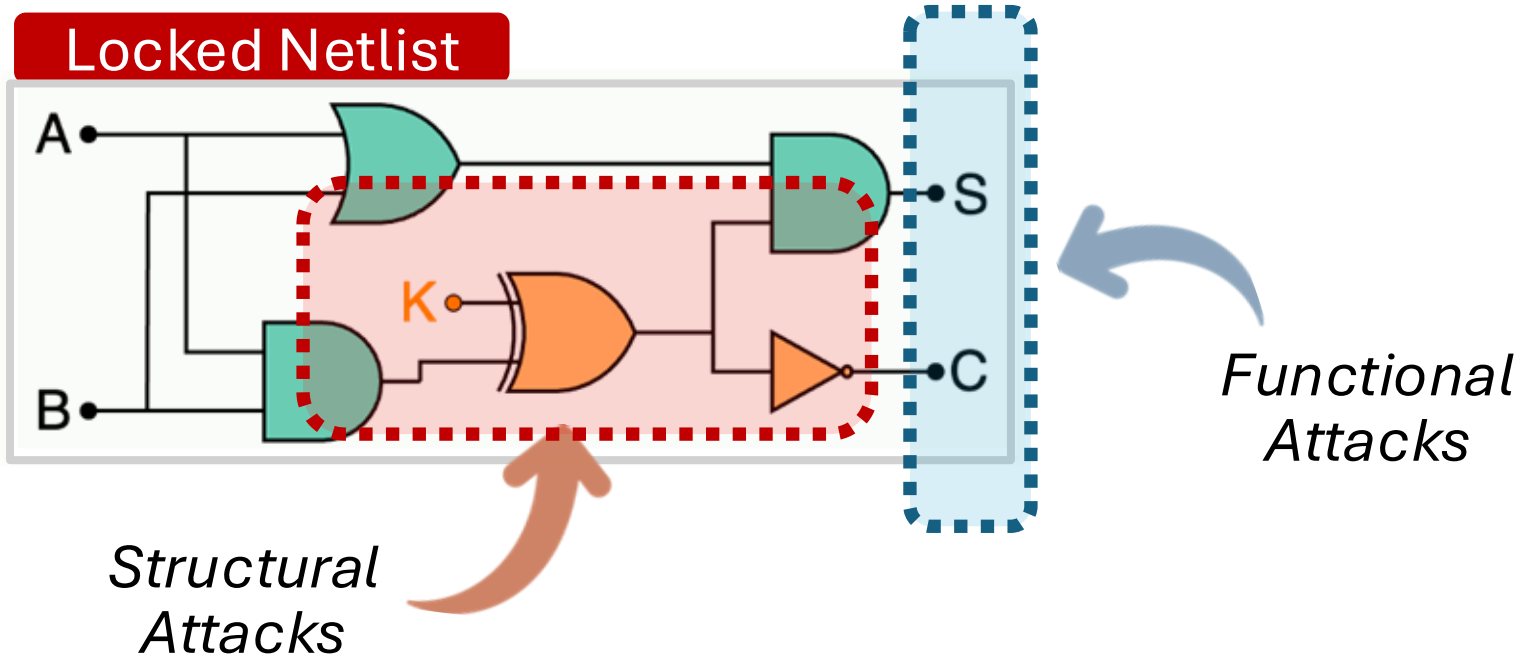
Functional Weakness of Logic Locking





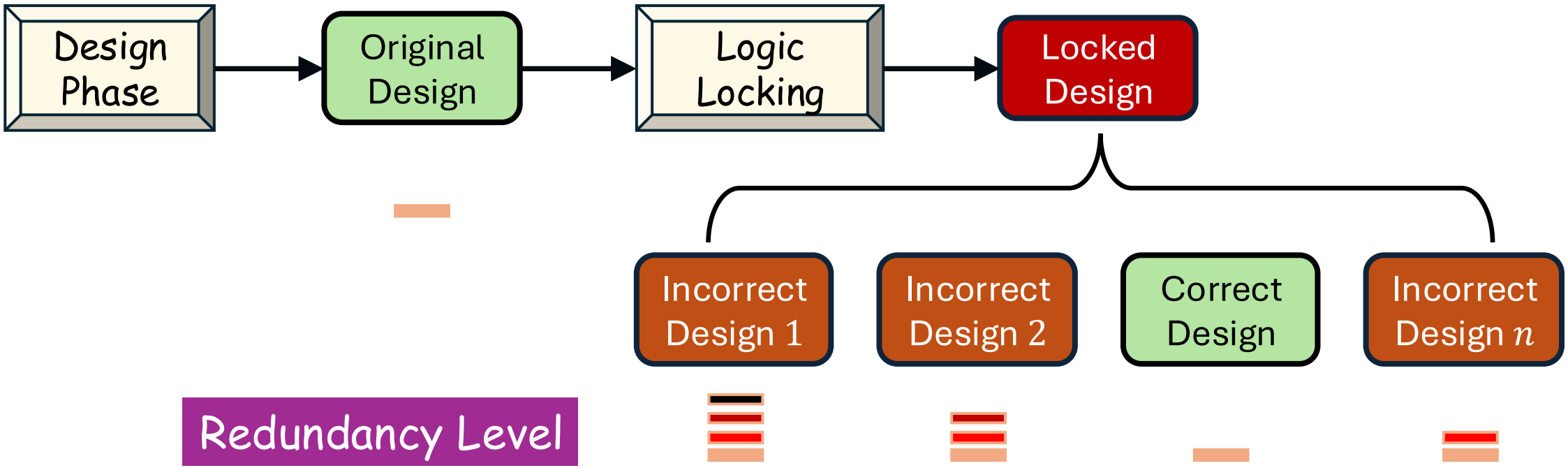
Structural Weakness of Logic Locking

- Functional attacks neglect structural modification traces.
- **Our finding:** Key vulnerability from the locked netlist alone.

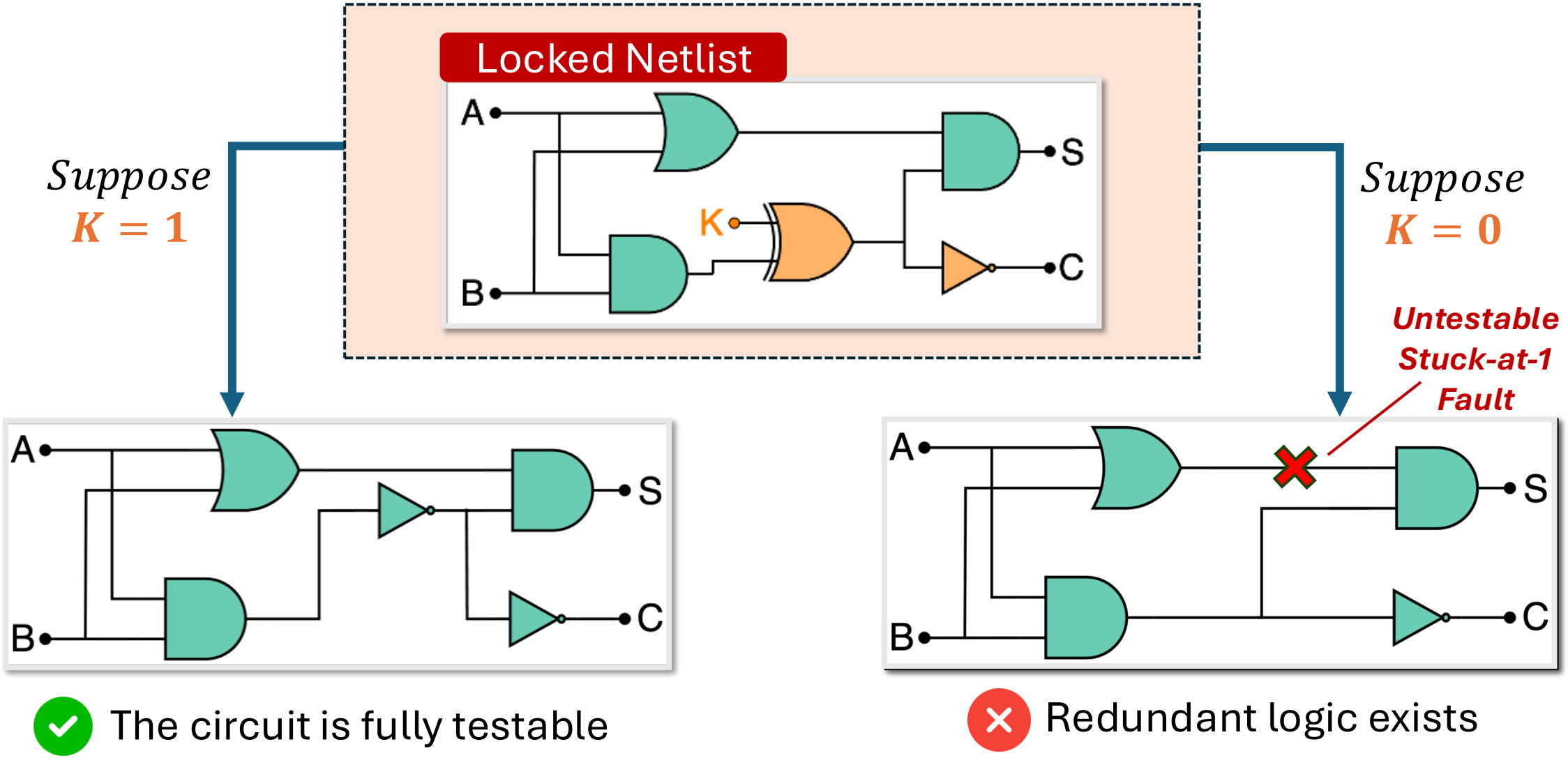


Key Differentiation

- **Observation:** Corrupted designs do not “look reasonable”.
- **Exploitation:** Compare the adherence of keys to **universal design principles**.
  - Key recovery **without** oracle or other collateral.



Redundancy Attack



The circuit is fully testable

Redundant logic exists

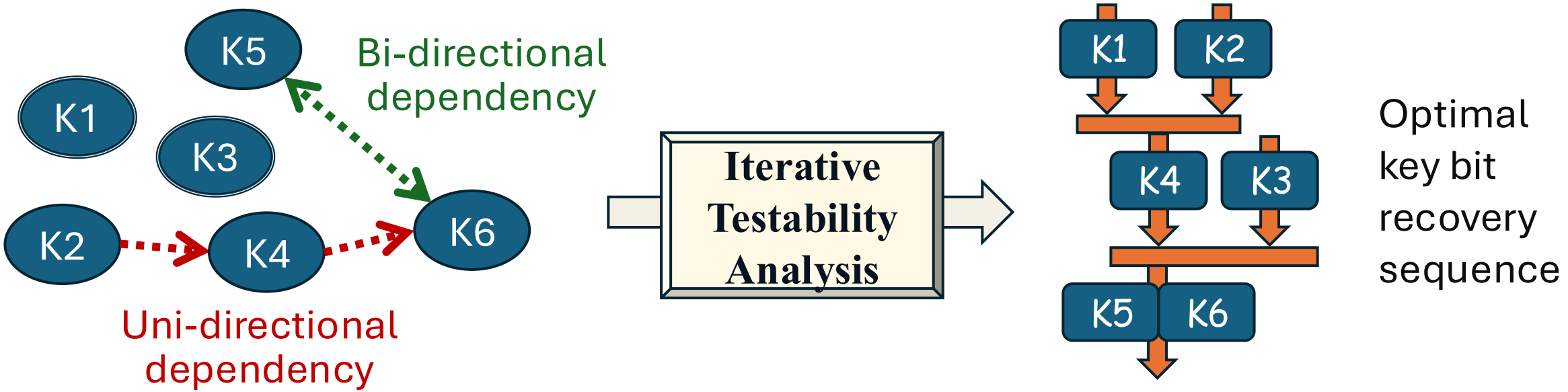
Redundancy Attack

Effectiveness

- Verify the **elevation of redundancy level** under incorrect keys through comprehensive topological analysis.

Efficiency

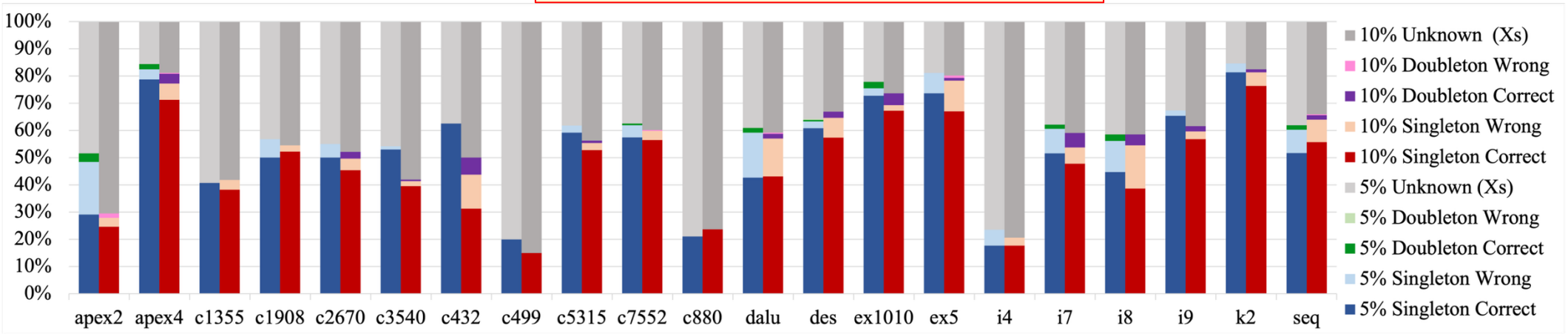
- An iterative algorithm to **decompose the key space** for key bit recovery and heighten confidence.



Redundancy Attack

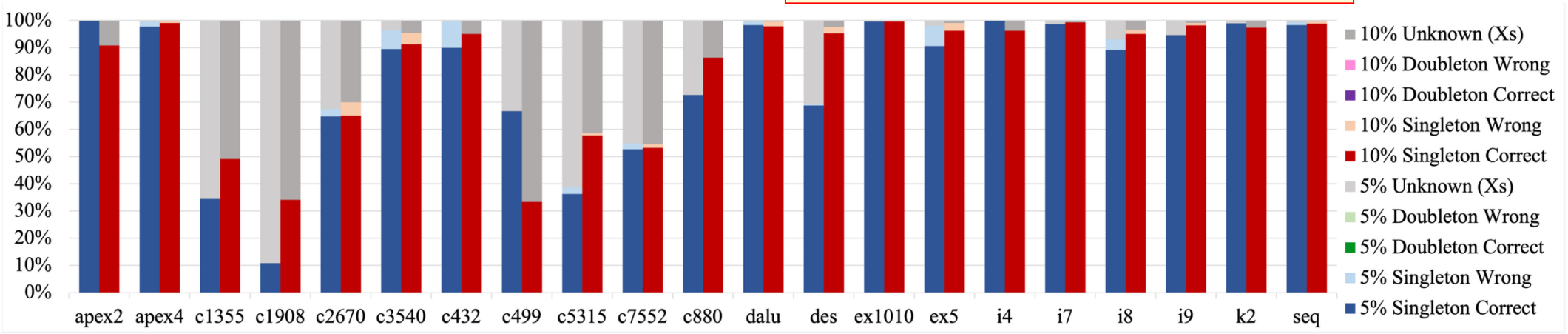
Towards random **XOR** locking

50% Accuracy, 90% Precision



Rowards corruptibility-driven **MUX** locking

82% Accuracy, 98% Precision

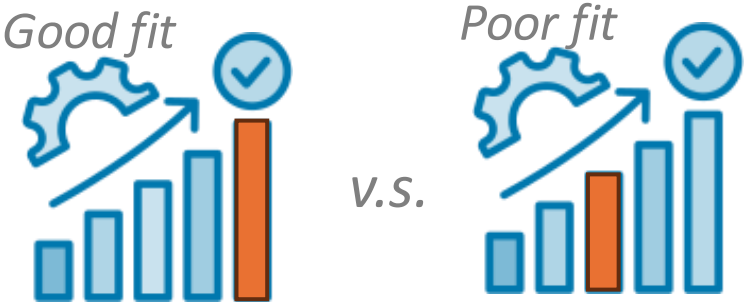


Sources of Structural Vulnerabilities

Redundancy
SWEEP
SCOPE
CLIC-A

Optimality

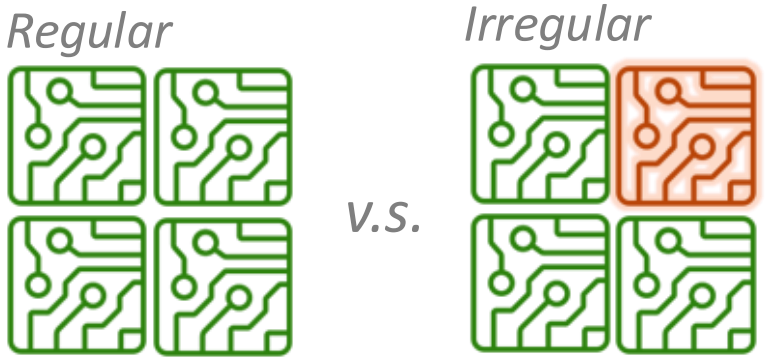
Correct design stands out to be the most optimal & rational



SnapShot
TGA
OMLA
UNTANGLE
MuxLink

Regularity

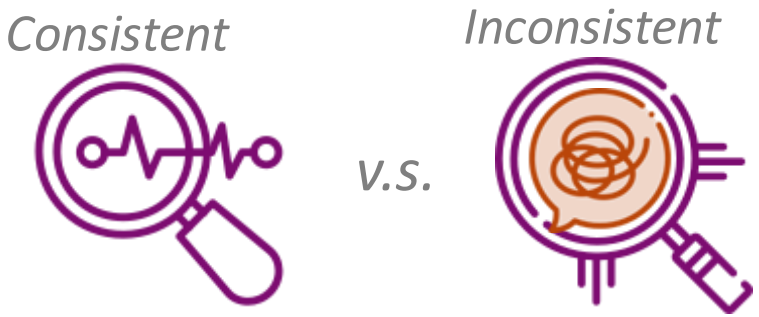
Incorrect keys disrupt the repetition of sub-circuits



SAIL
FALL

Synthesis Predictability

Tool preferences can be learned through repeated application



WASHINGTON STATE UNIVERSITY

T2

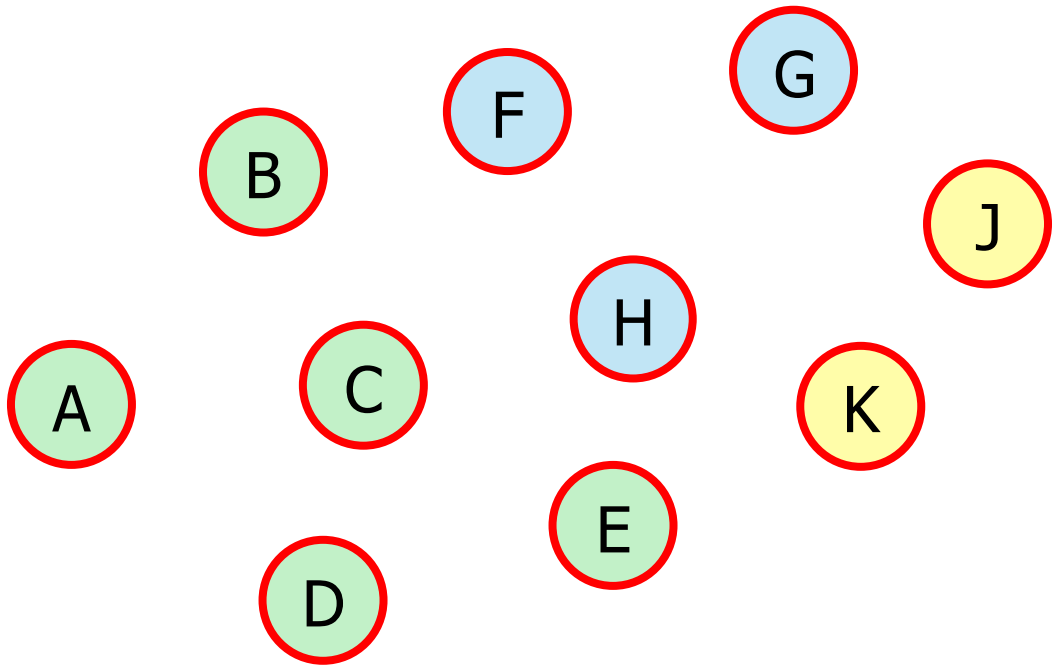
Sequential Obfuscation via State-Space Partitioning

Break the dilemma of *key exposure* and *protection scope*

HOST'21, DATE'22

Visualization of Logic Locking Coverage

- Assume each piece of functionality is **locked under its own key**, and a **master controller** selects among them.



Key gate lock

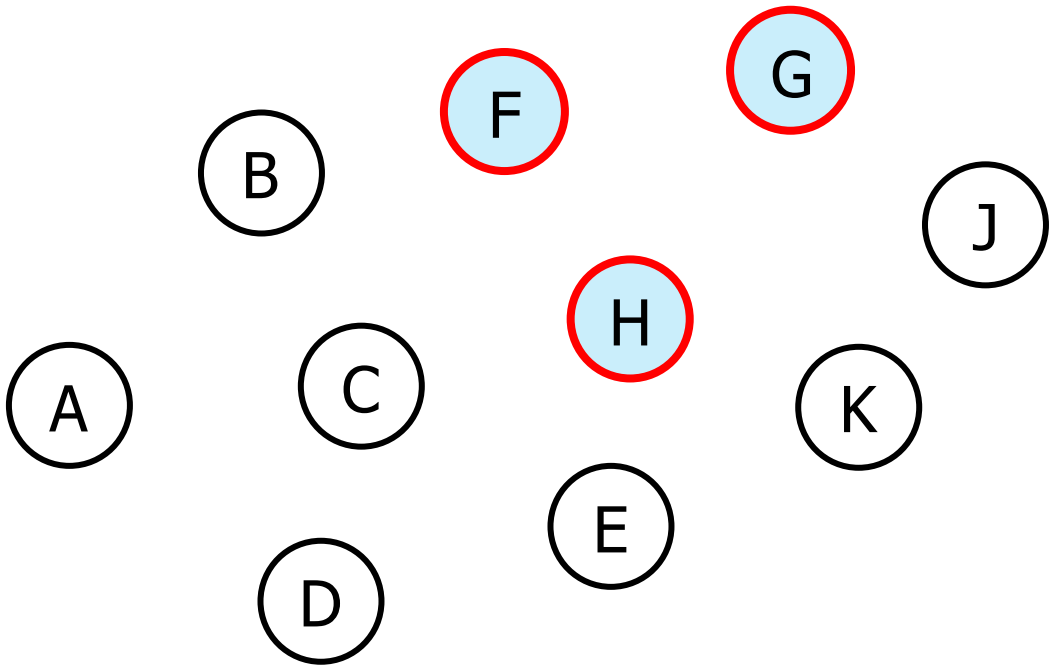
If Execution = **A or B or C or D or E**
Key = **Green**
If Execution = **F or G or H**
Key = **Blue**
If Execution = **J or K**
Key = **Orange**

High corruption → Strong protection against piracy and reverse enginering

High key exposure

Visualization of Logic Locking Coverage

- Attack resilience comes with the comprose in protection coverage



“Attack resilient” lock

If Execution = F or G or H

Key = Blue

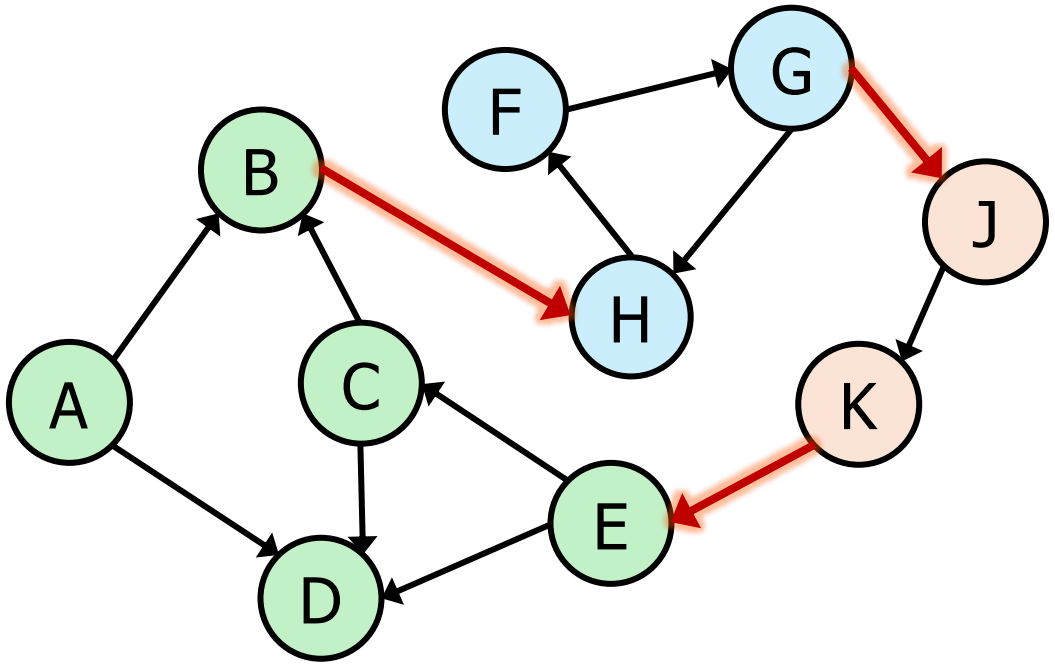
Default: Unprotected

Little functional exposure

Minimal protection offered

Visualization of Logic Locking Coverage

- Observation: Protected functionalities are **not individual events**
- Idea: Leverage the **known temporal execution sequence** to expand coverage

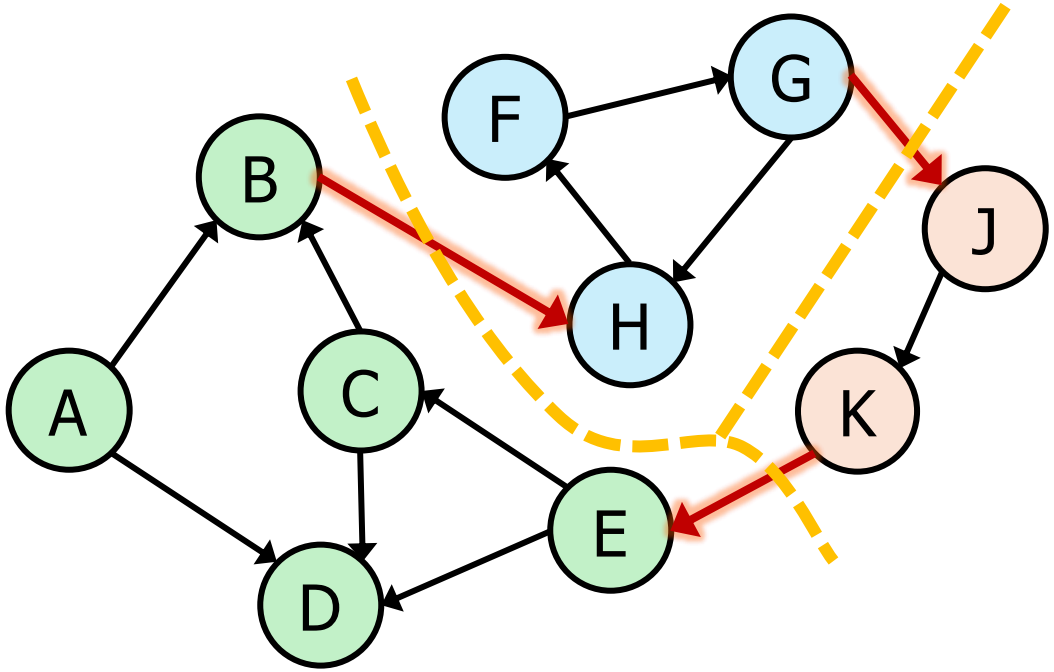


Proposed solution

If **B→H**: Key = **Blue**
If **G→J**: Key = **Orange**
If **K→E**: Key = **Green**
Default: Hold

- **Little** functional exposure
- **Broad** protection coverage ↑↑

Components of JANUS Obfuscation



A/ Configuration-based Locking

Synthesize the design under configurations for produce a corrupted netlist.

B/ Configuration Assignmenet

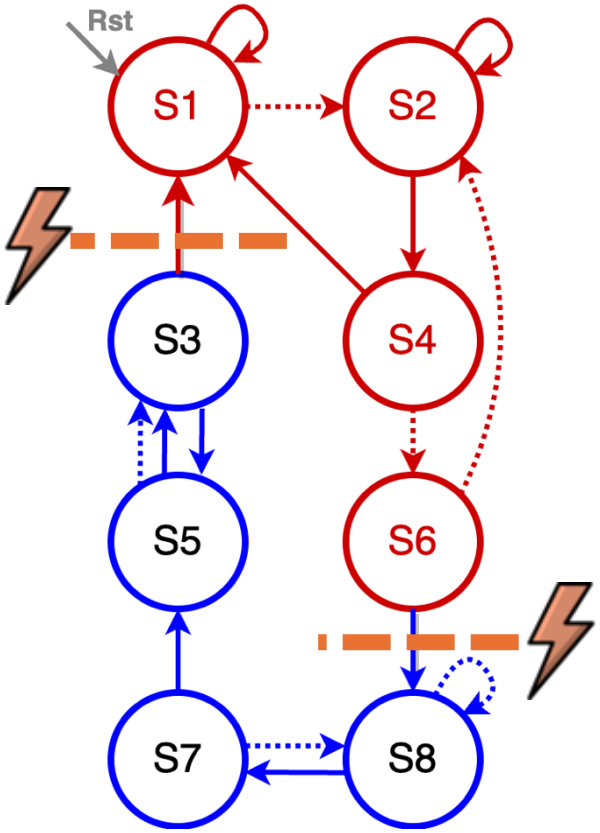
Determine the configurations that minimize key exposure

C/ Hybrid Activation Control

Combine intrinsic sequential evoluation & external secret key for activation control

Optimal Configuration Assignment

- A **balanced** **minimum-cut** is applied to assign multiple configurations.
- Minimize the key exposure by assigning it only “minimal duties”.



Balanced partition

Obfuscation Strength

Minimum-cut

Key Security

Key-Controlled Configuration *Toggle*

State/Transition Information

f

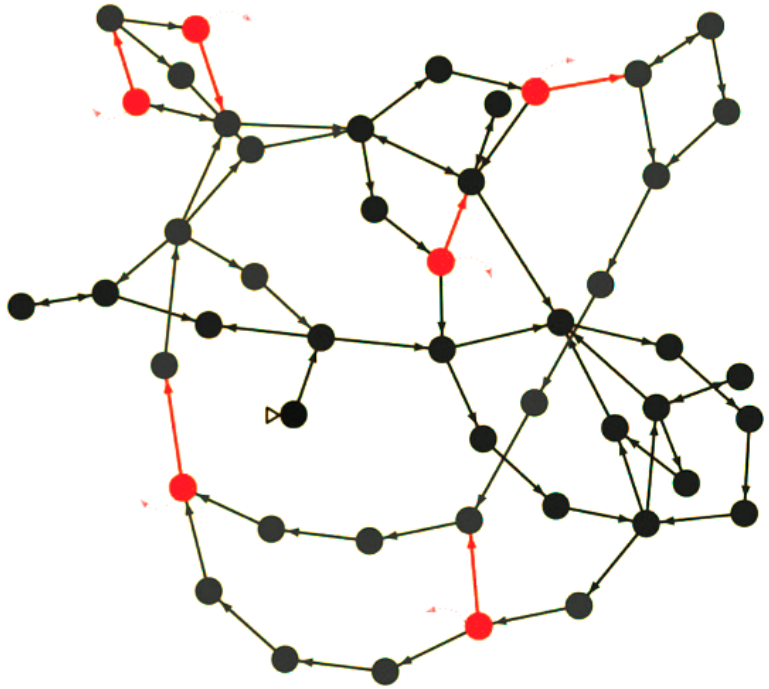
Hold or *Toggle* *the current configuration*

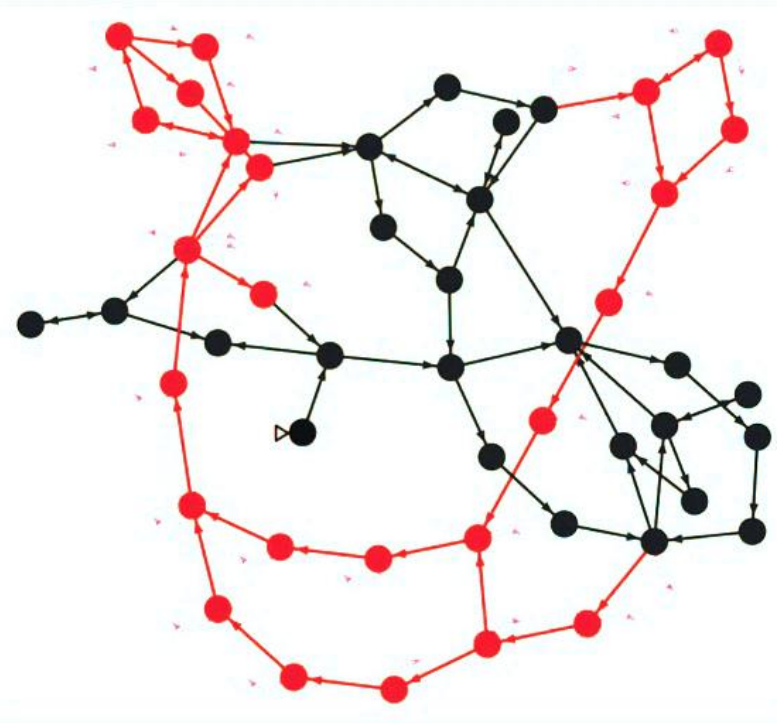
Key Inputs

Effectiveness of *JANUS*

- **17.5x improvement on corruption** at the same attack resilience level.
- RTL corruption achieves strong integration and structural resilience.

Key-based Obfuscation





JANUS Obfuscation

Obfuscated states/transitions at the **same attack resilience**

WASHINGTON STATE UNIVERSITY

T3

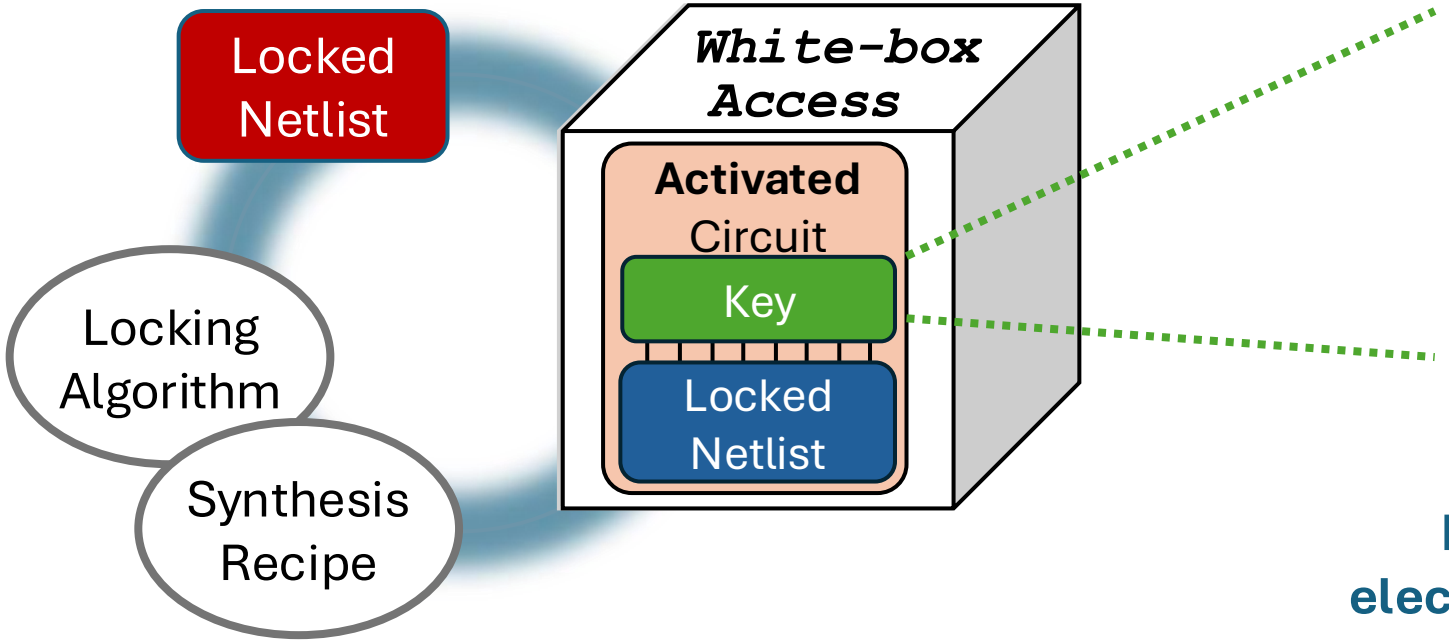
White-Box Logic Obfuscation

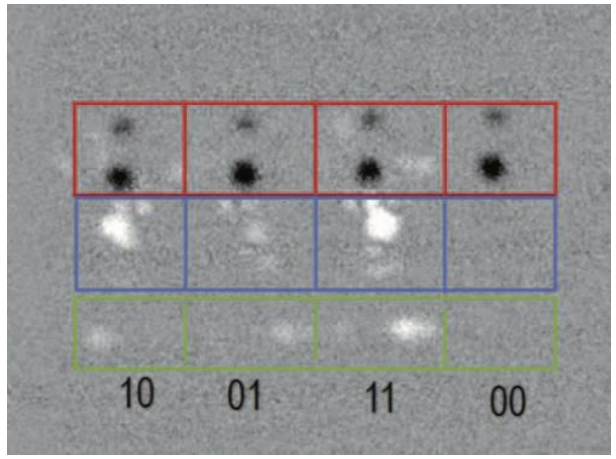
Functional reverse engineering resilience in structural transparency

VTS'23, ATS'23, ITC'24, ASP-DAC'25

White-box Attack Model

- When no external secret key is available, the attacker effectively becomes a **white-box adversary**, with full transparency into the operational system.
- Indeed, semiconductor attackers' capabilities are close to a **white box.**
  - Physical key extraction can be achieved through electro-optical frequency mapping.



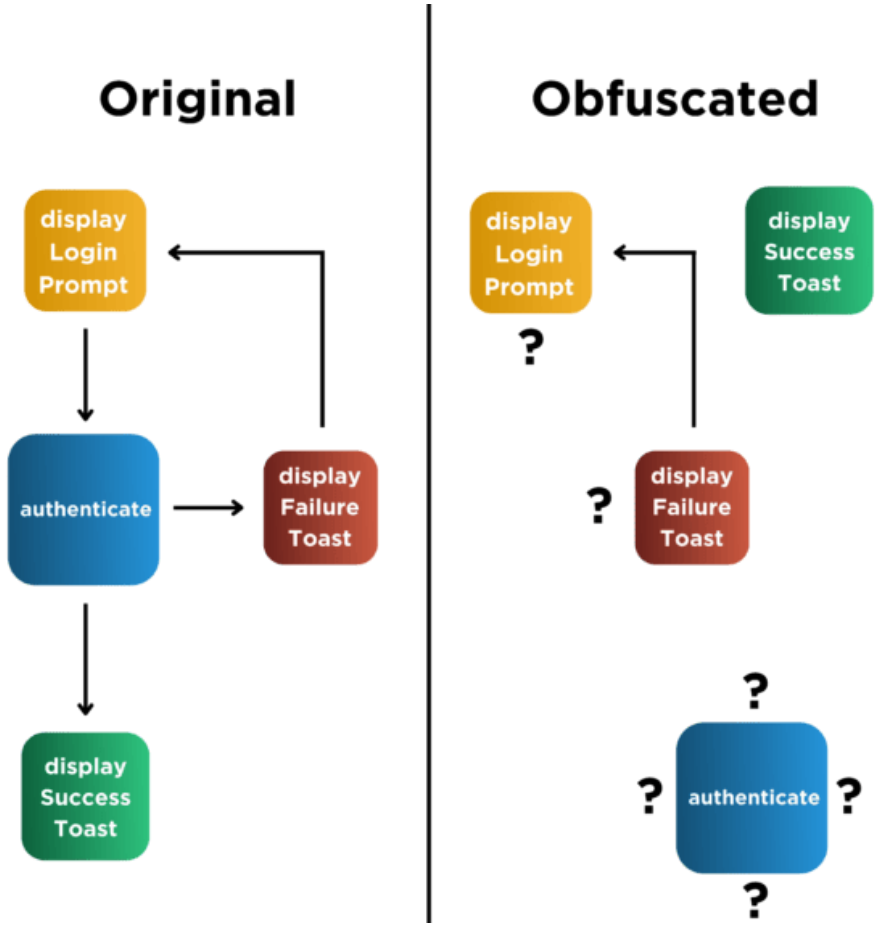


"Key is Left under the Mat":
Physical key extraction with electro-optical frequency mapping.

Rahman, M. Tanjidur, et al. HOST 2020

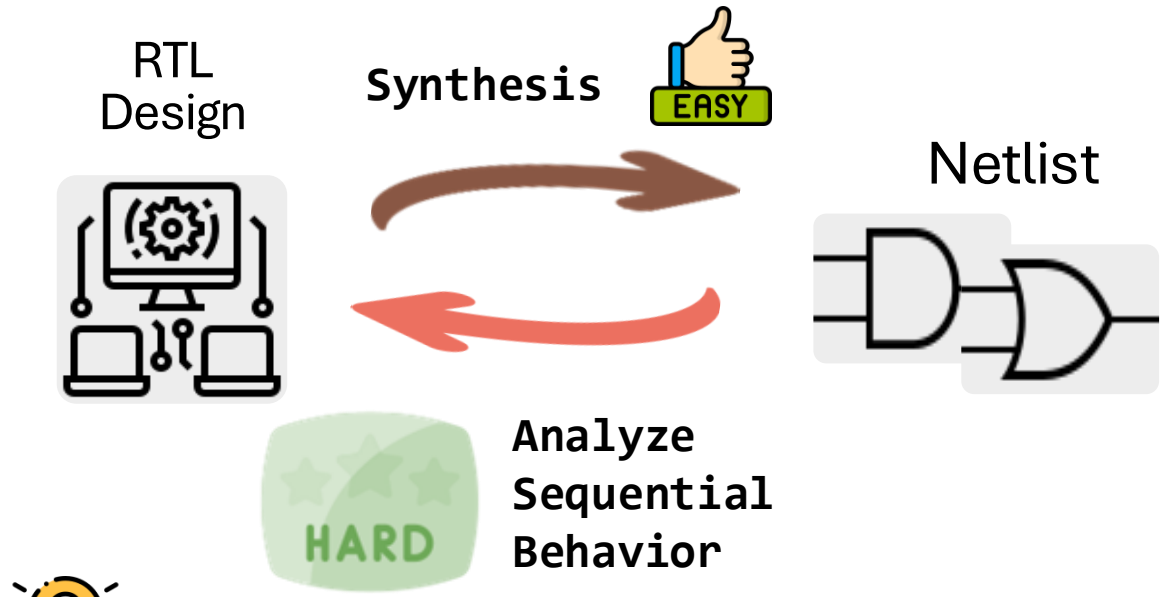
How is Any Defense Possible in a White-Box?

Software Obfuscation Example



Computed Control Flow *stops a decompiler from knowing the exact Relative Virtual Address being jumped to.*

"One-way" Tasks in Hardware Analysis

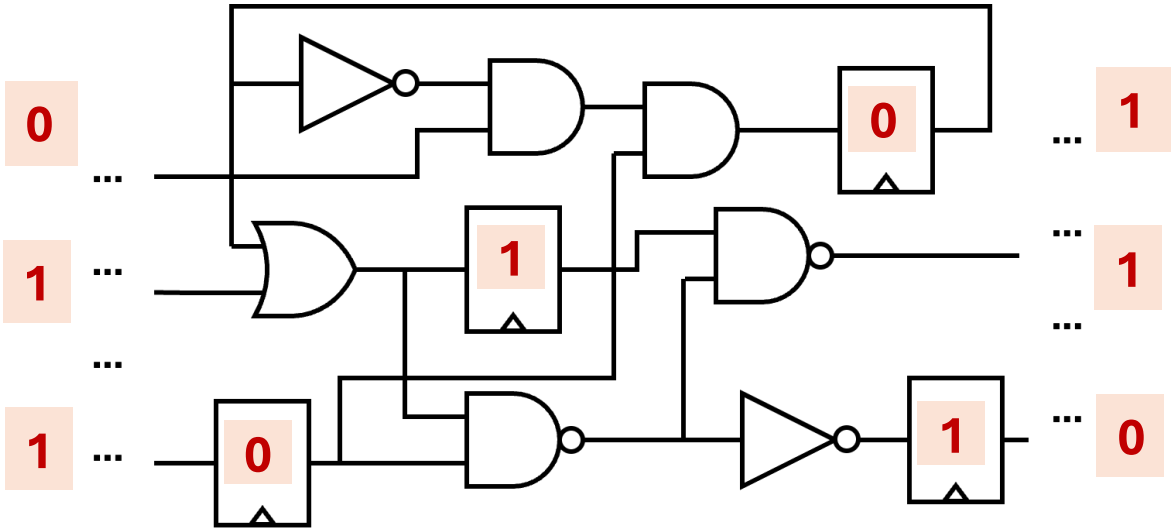


How can we leverage this difficulty to build obfuscation solutions?

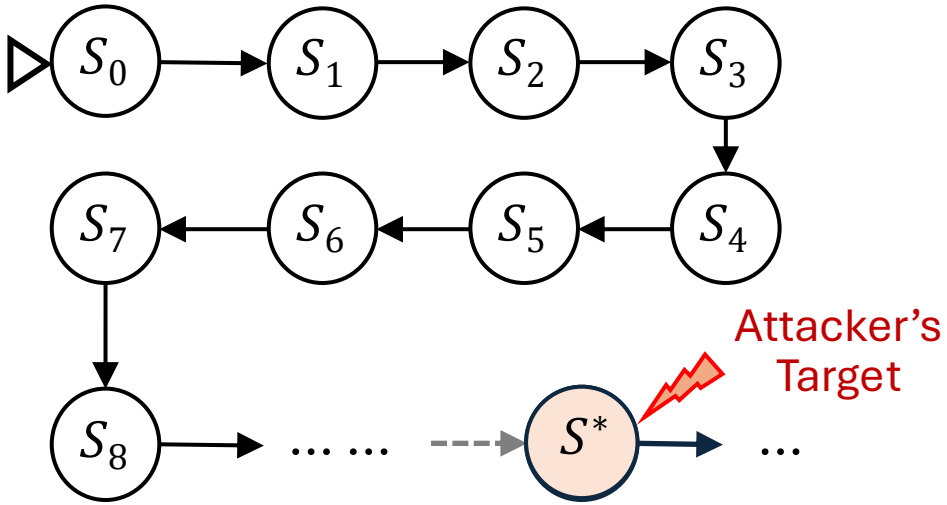
Reverse Engineering of Sequential Machine

- **Attack Goal:** Efficiently learn the functionality under specific conditions
- **Attack Approach:** Simulate the netlist combinationally.

Attacker's View (Gate-level Netlist)



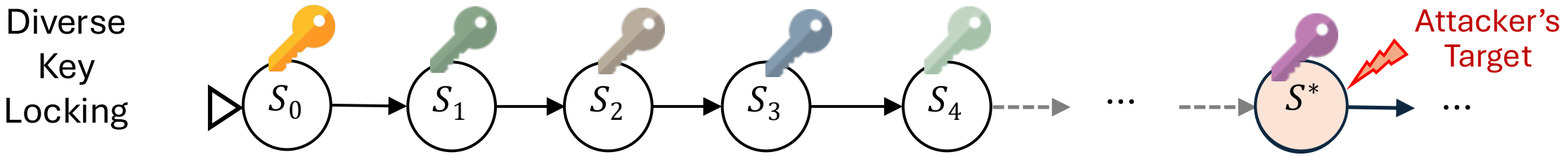
Reverse Engineering Goal (Behavioral/RTL)



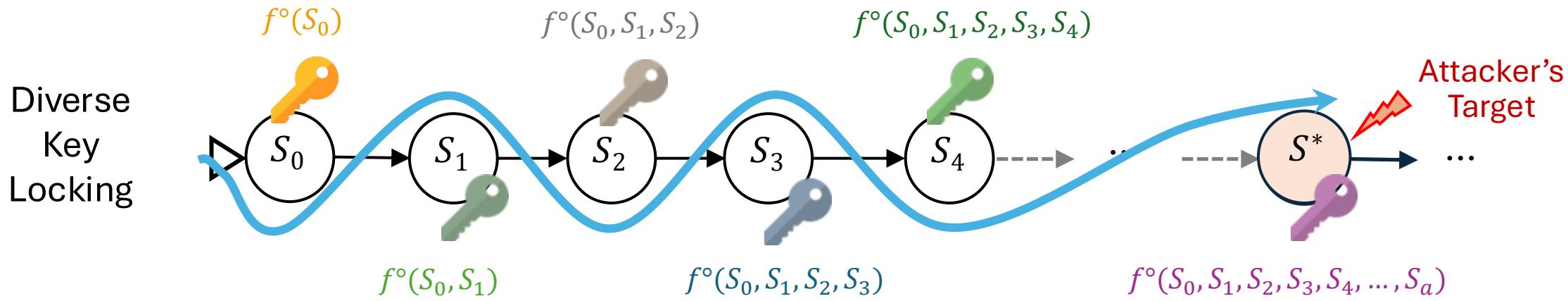
Defense Goal: Make combinational analysis as difficult as sequential analysis.

White-box Logic Obfuscation (WBLO)

- Locking with **diverse** keys
  - Eliminate any single point of vulnerability.
- Secure key **self-generation** mechanism
  - **Hide keys within the sequential execution patterns** so that arbitrary recovery becomes as computationally expensive as deep sequential analysis.



White-box Logic Obfuscation (WBLO)



Design Time

Key Assignment

$$Key(S^*) = f°(S_0, S_1, S_2, \dots\dots, S^*)$$

Locking

$$Out^{locked}(S^*, X) = Out(S^*, X) \oplus Key(S^*)$$

Easy for the designer, given the full knowledge of the FSM specifications

Execution Time

Key Evaluation

$$Key_{next} = f(Key_{curr}, State), repeatedly$$

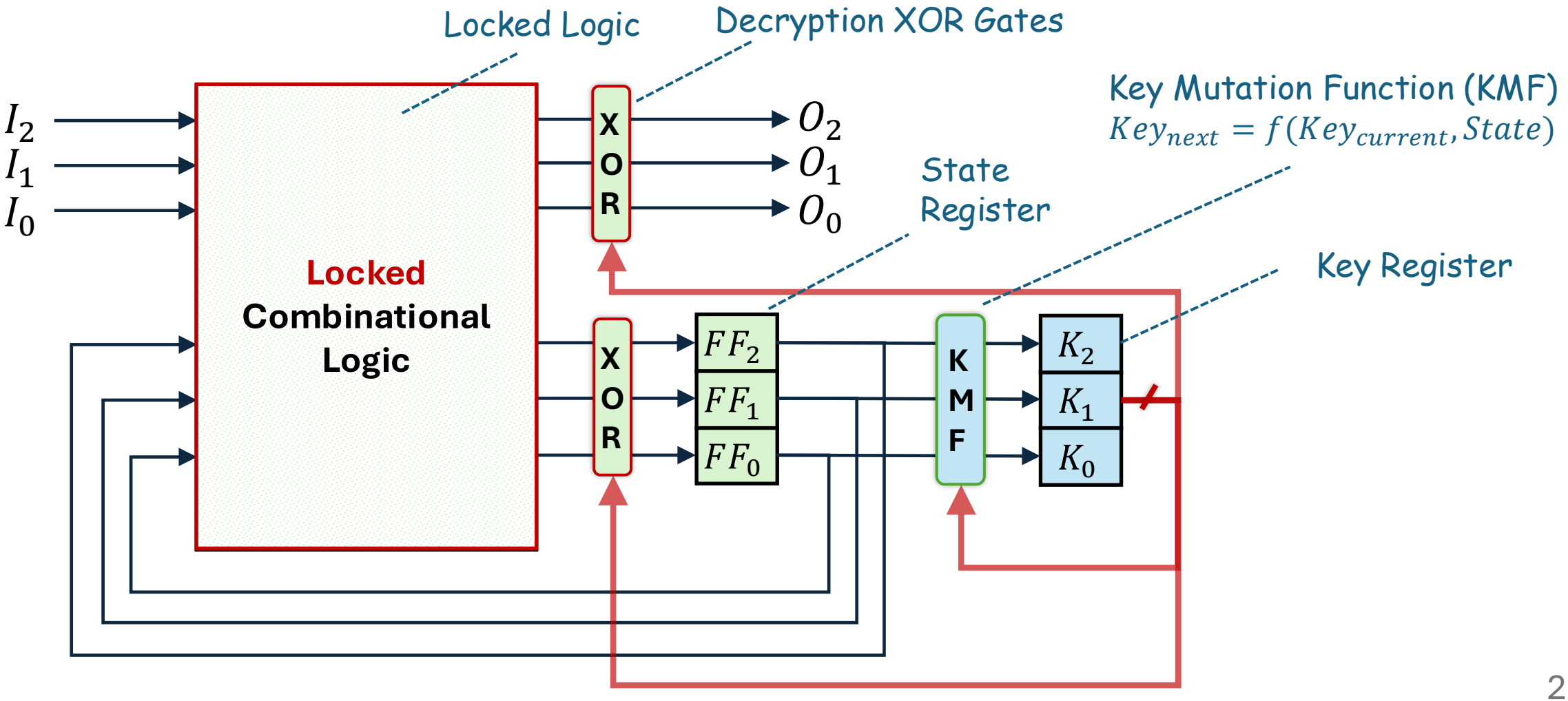
Activation

$$Out(S^*, X) = Out^{locked}(S^*, X) \oplus Key_{next}$$

Difficult for an attacker with only netlist access, as it requires multi-cycle reasoning.

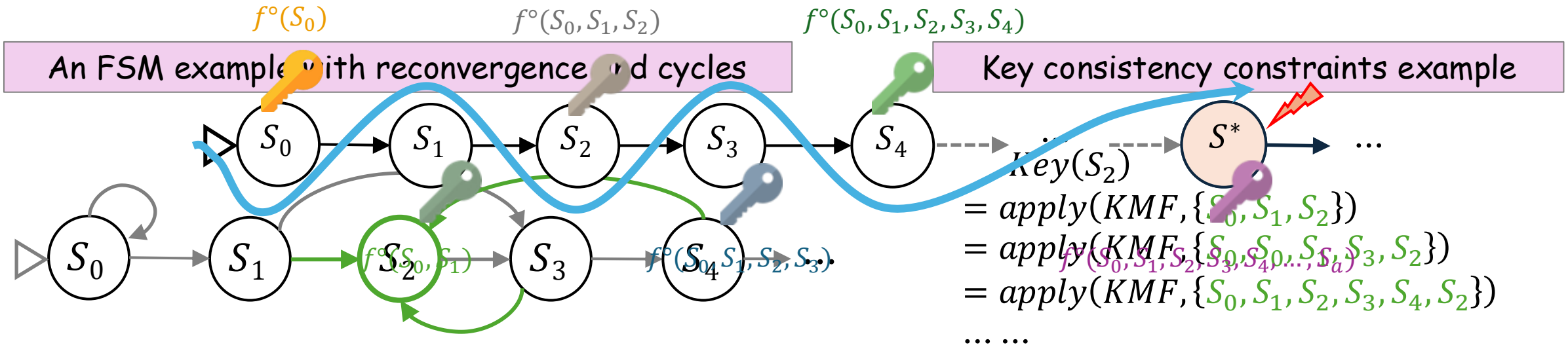
WBLO Architecture

- The **Key Mutation Function (KMF)** continuously updates the active key based on the state information.



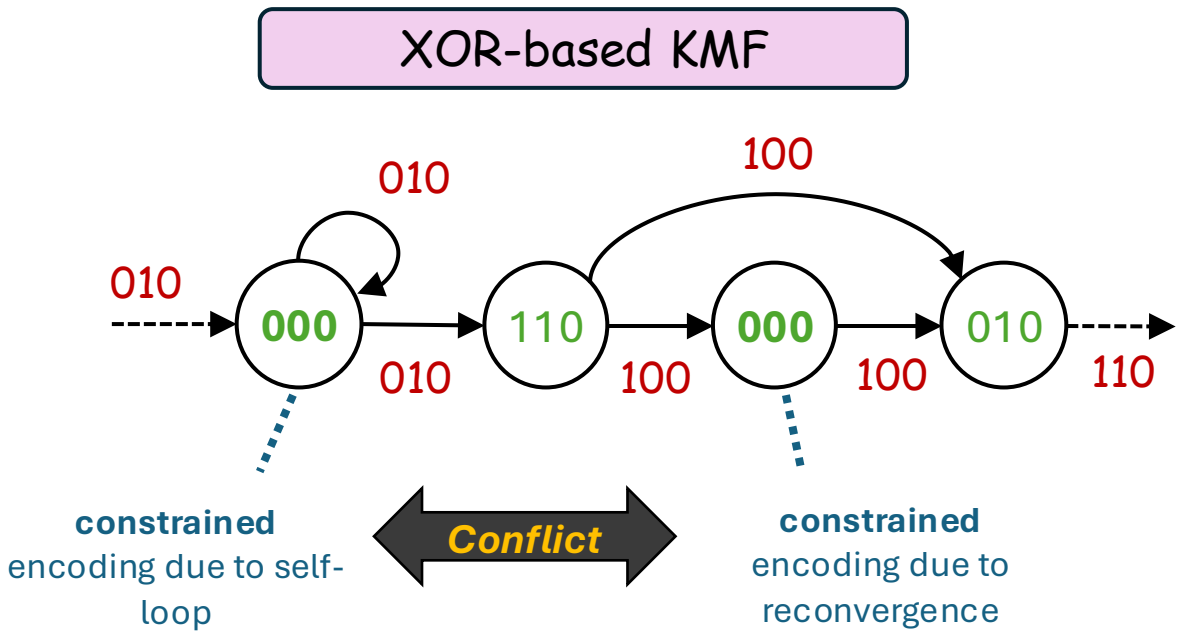
Multi-dimensional Optimization

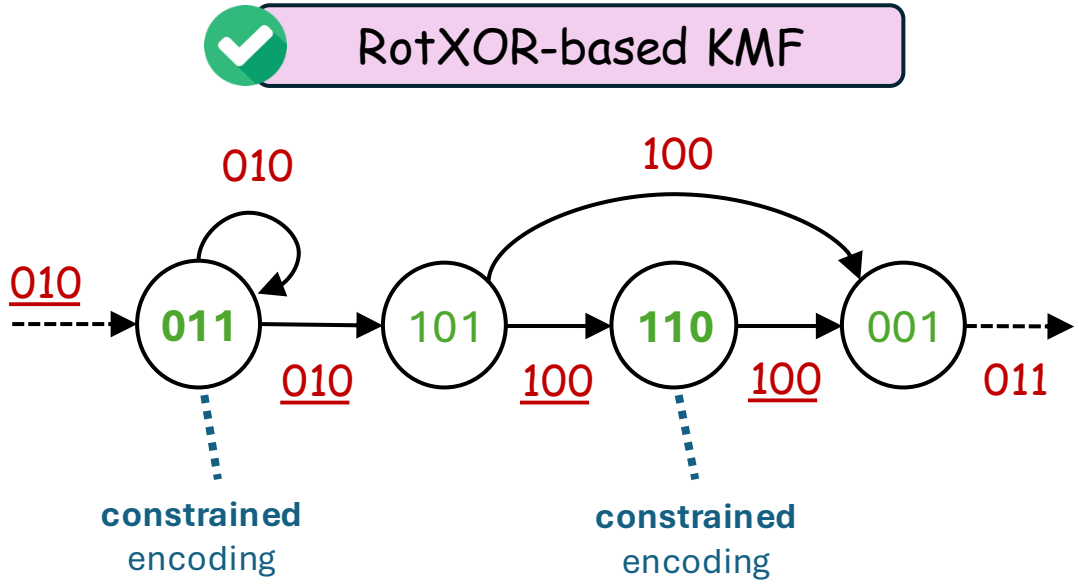
- **Security** requires the pool of keys to exhibit high entropy.
- **Correctness** requires consistent per-state key generation across distinct paths.
- **Tight or composite cycles and reconvergences** create complications.



Pick the Best Key Mutation Function

- KMF determines the satisfiability of key consistency constraints in tight sub-graphs.
- **XOR-based KMF:** $Key_{next} = Key_{current} \oplus State$
  - The involution property imposes encoding constraints when the key difference is identical.
- **RotXOR-based KMF:** $Key_{next} = RotateR(Key_{current}) \oplus State$
  - Different incoming keys allow the same key difference to be resolved through distinct encodings.

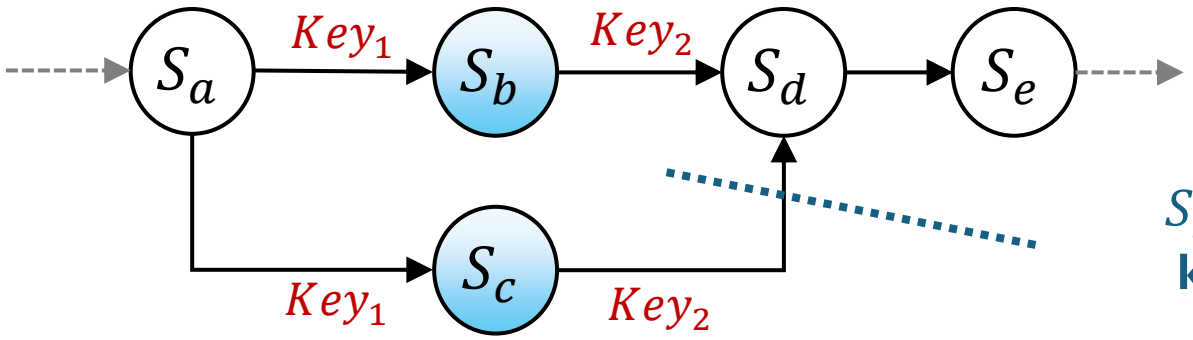




Validity Resolution

- Conflicting scenarios still exist in **symmetric reconvergence & composite graphs.**
- KMF is a reversible function:
  - Definition of KMF: $Key_{next} = RotateR(Key_{curr}) \oplus State$
  - Evaluation of encoding constraints: $\boldsymbol{State} = Key_{next} \oplus RotateR(Key_{curr})$
  - An effective solution must break this property while preserving key mutation effectiveness.

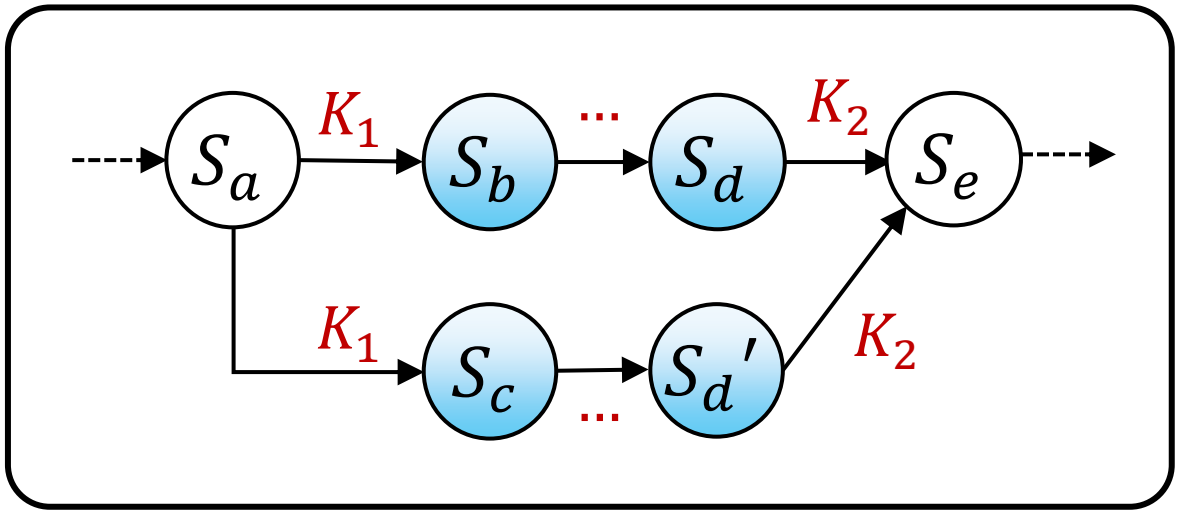
An FSM example with encoding-unsolvable conflicts



S_b and S_c demand **identical key mutation effects**, and thus **identical encoding**

Validity Resolution Strategies

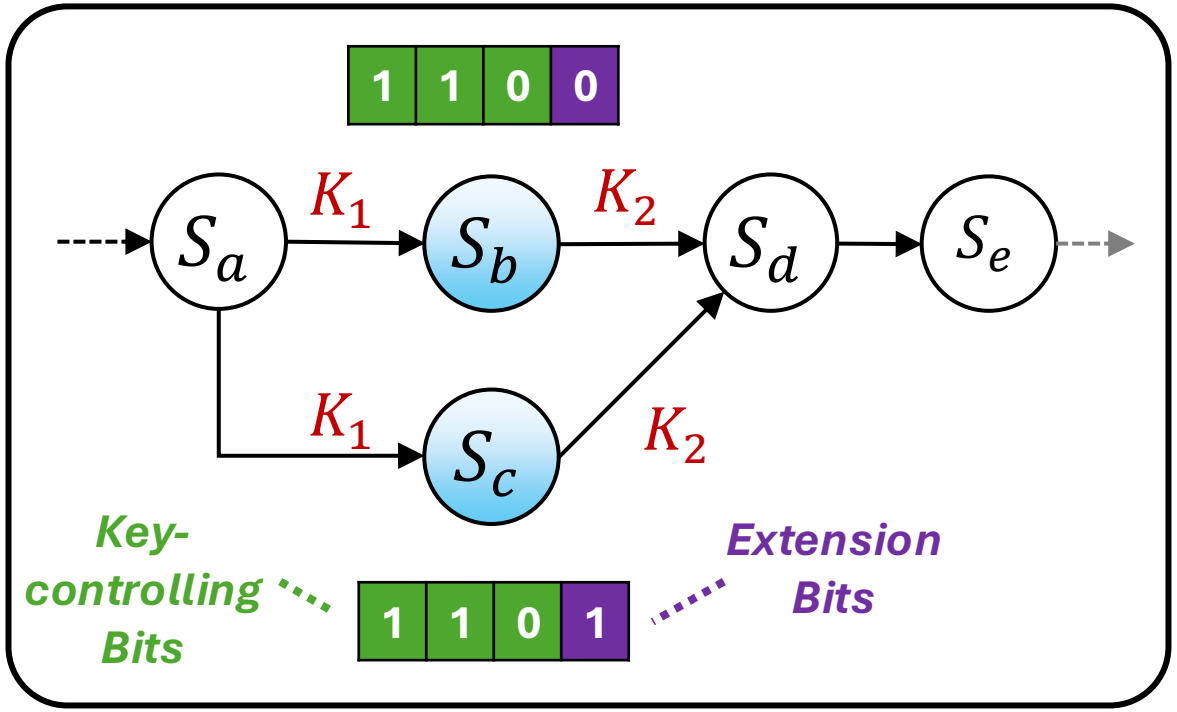
State Duplication (SD) to relax constraints



Universal Solution (Completeness)

Expensive to Implement Additional States

Encoding Extension (EE) to tolerate constraints

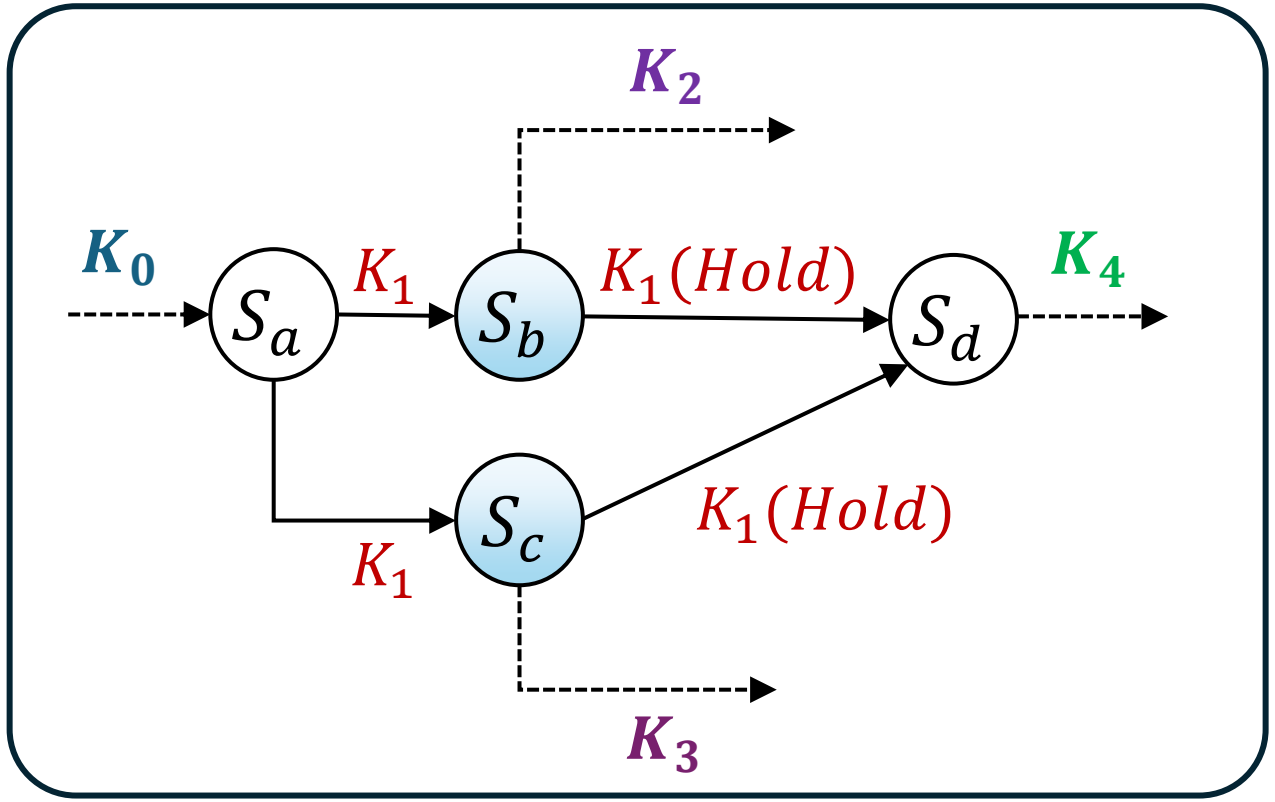


Addresses Encoding Contention

S_b and S_c share identical key

Validity Resolution Strategies

Mutation Control (MC) to eliminate constraints



- Transition-level Mutate/Hold
- Preserve Key Diversity Amid Conflicting Graph Structures
- Low Cost
- Must Avoid Unrestricted Use (as it renders all keys identical)

Obfuscation Algorithm – Graph Analysis

- A breadth-first traversal framework for making key assignments
- Resolve encoding requirements and conflicts in the following **priority orders**:

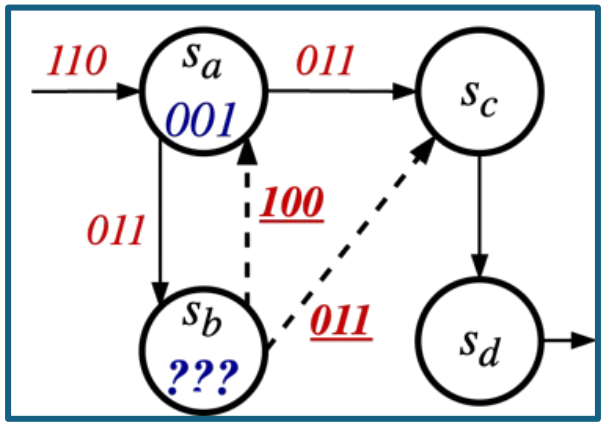
MC: Mutation Control → **R-KMF: RotXOR KMF** → **EE: Encoding Extension** → **SD: State Duplication**

Applied only when proven to benefit key diversity

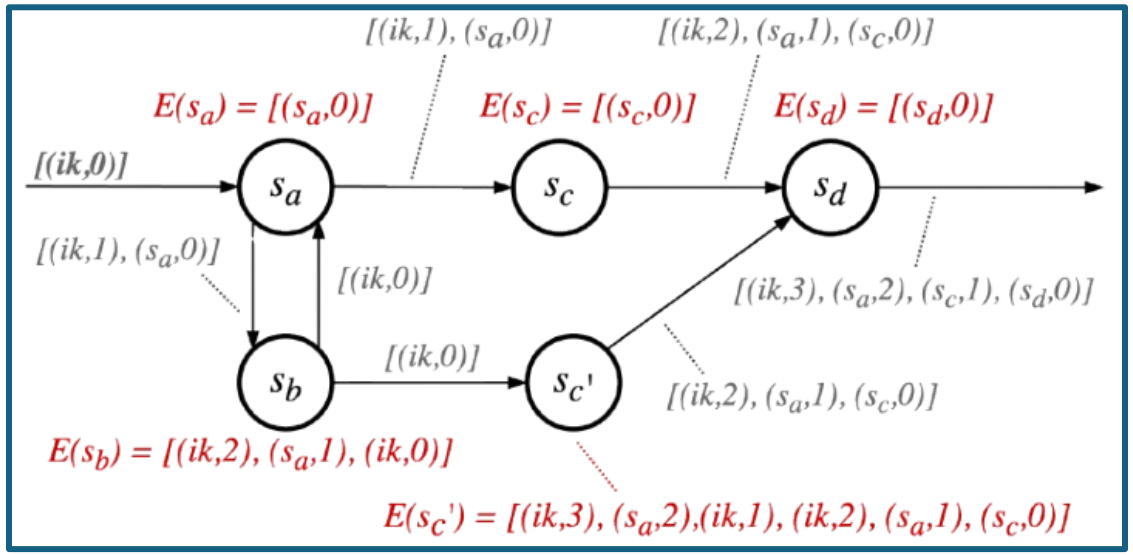
Baseline operation to promote key mutation & satisfy key consistency

Operate under a pre-defined extension width limit

Backup strategy to ensure algorithmic completeness

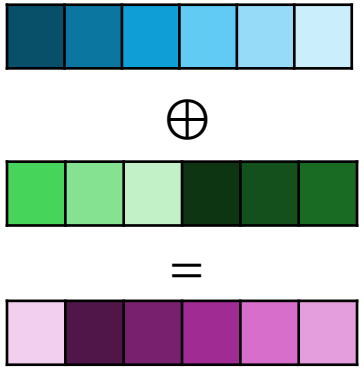


Simultaneous *Transformation* & *Constraint Analysis*

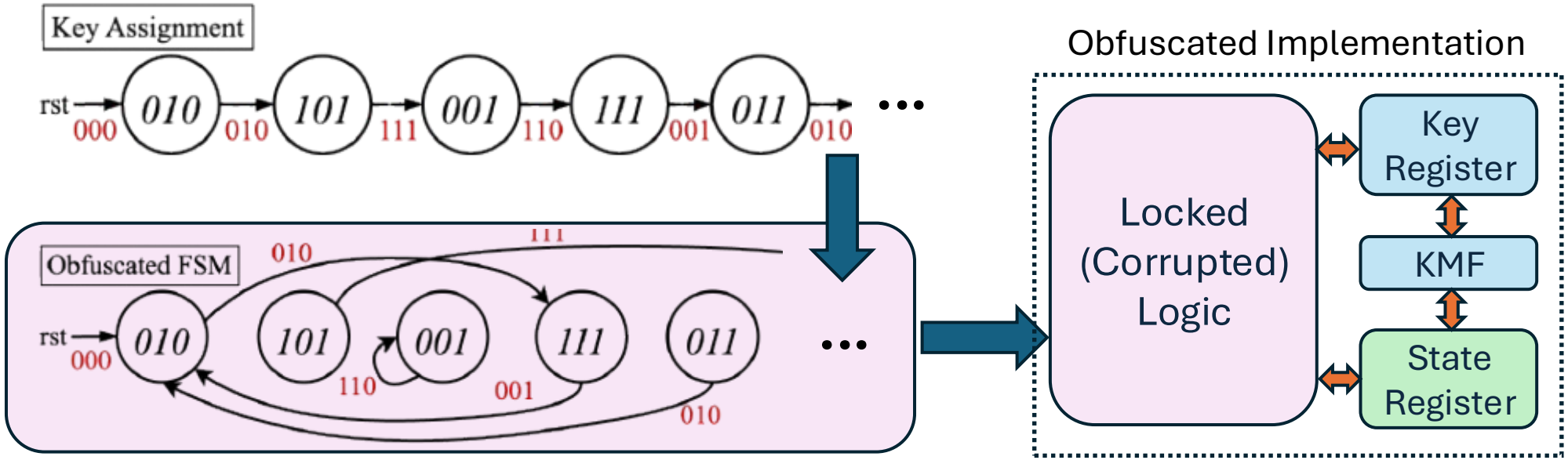


Obfuscation Algorithm – Solving & Locking

- Symbolic formulas are passed onto a Boolean Satisfiability solver to obtain **binary encoding and binary keys**, with awareness of the rotation effects.
- Divert all transitions and corrupt outputs using the state-specific keys for locking.
- Assemble the locked logic with KMF to obtain an obfuscated implementation

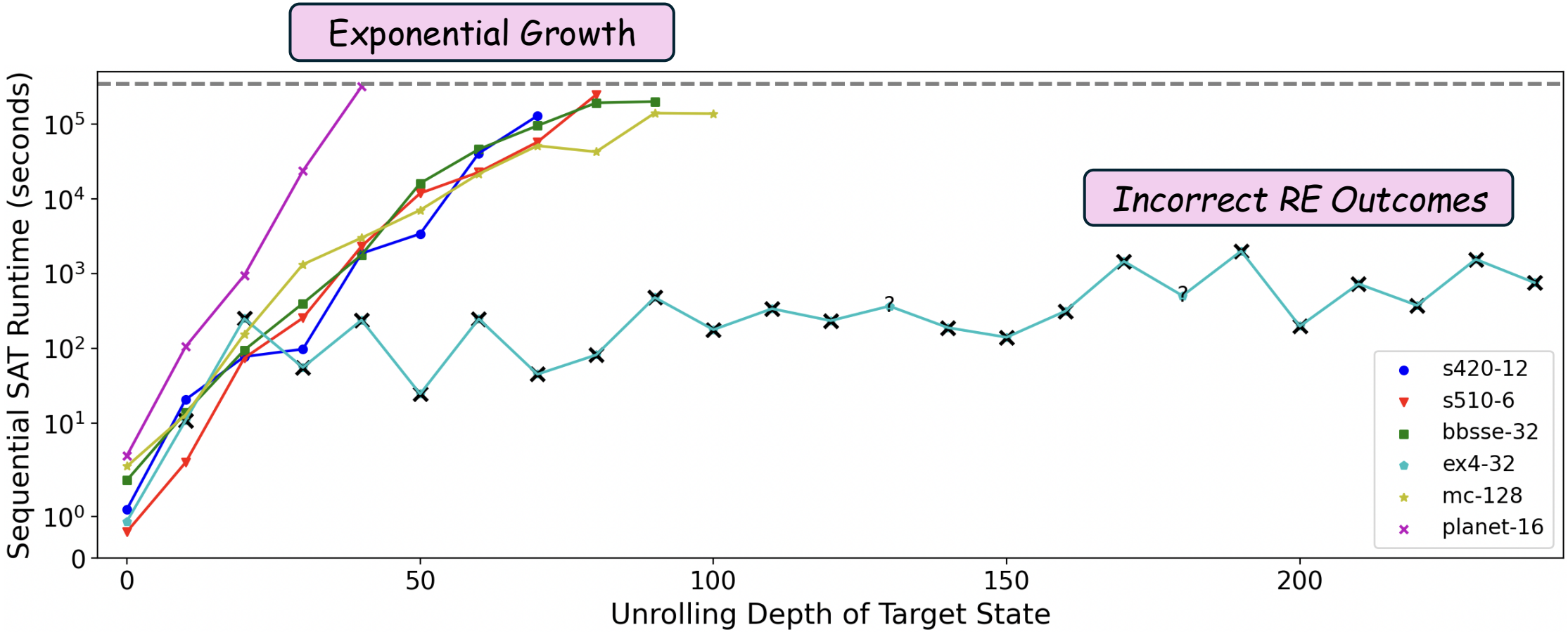


***Rotation**-aware*
(shown by gradient)
SAT solving



Results – Security

- Optimized transition-level mutation control greatly enhances key diversity.
- An **exponential increase** in sequential SAT attack complexity as depth grows.



Results – Overhead

- **Encoding Extension** delivers significant savings for all cases requiring state duplication.
- Control logic occupies minimal chip area and is generally not on the critical path, making its impact on a complete circuit negligible.

Overhead in Percentage

Bench	Best Config.	State	Area	Power	Delay
s420_12	*RotXOR + No Extension*	0.5%	16.6%	25.7%	25.4%
s510_6	*RotXOR + No Extension*	0%	13.8%	9.5%	23.8%
bbsse_32	*RotXOR + **2-bit** Extension*	**7.7%**	26.5%	19.2%	20.5%
ex4_32	*RotXOR + No Extension*	0%	28.4%	-3.2%	21.3%
mc_128	*XOR + No Extension*	0%	-7.5%	-2.4%	25.9%
planet_16	*RotXOR + **1-bit** Extension*	**4.2%**	45.8%	35.9%	15.9%
Average		**2.1%**	**20.6%**	**14.1%**	**22.1%**

T4

Temporal-Enhanced Error Detection

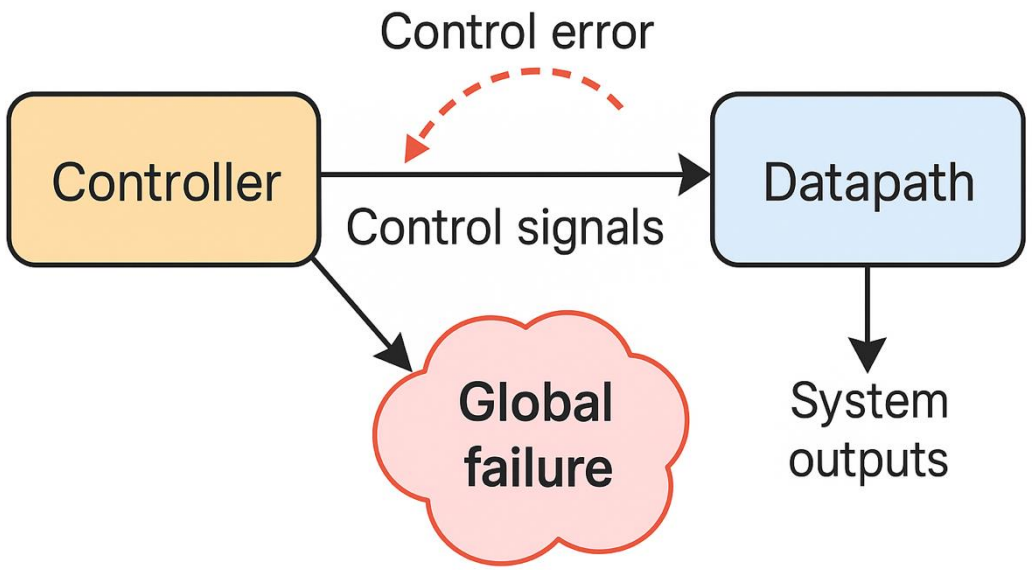
Stronger Error Detection through Controlled Error Propagation

Enabled by Lightweight WBLO

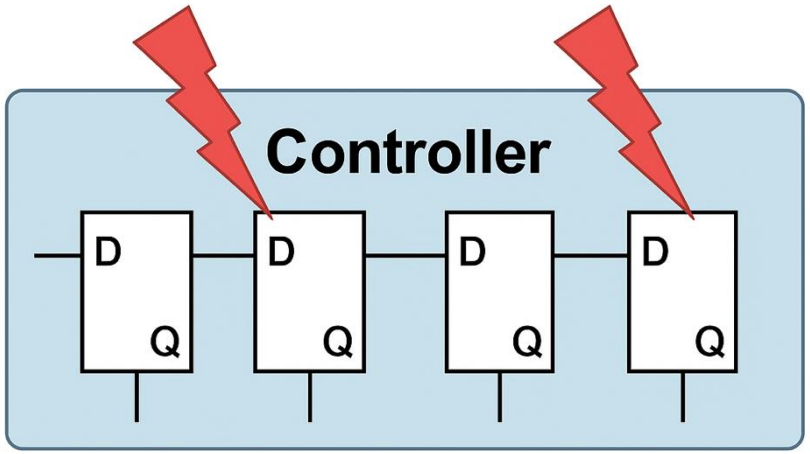
ATS'25

Errors in Controllers

Control errors lead to
global system failures



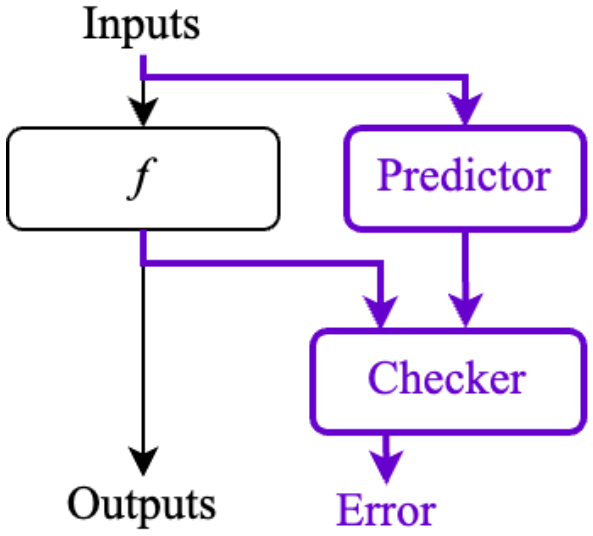
Multi-bit faults
are common



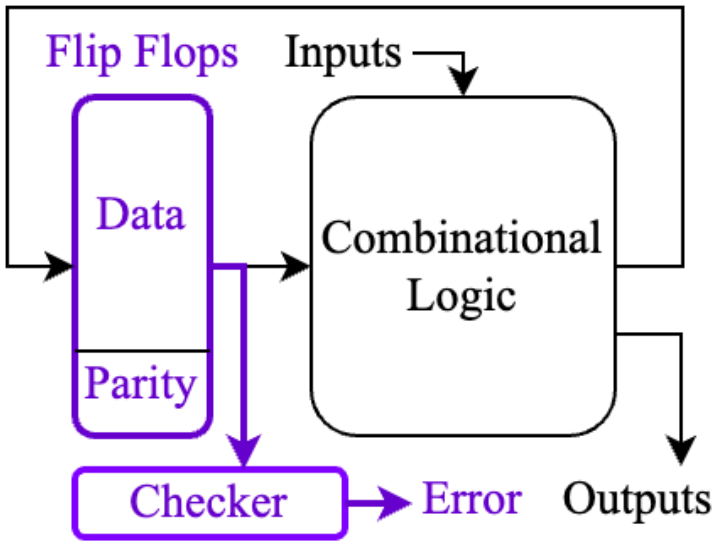
Error Detection

- Error detection uses **redundancy** to test an **invariant** and flag when it no longer holds.

Generic **Concurrent Error Detection (CED)** Model

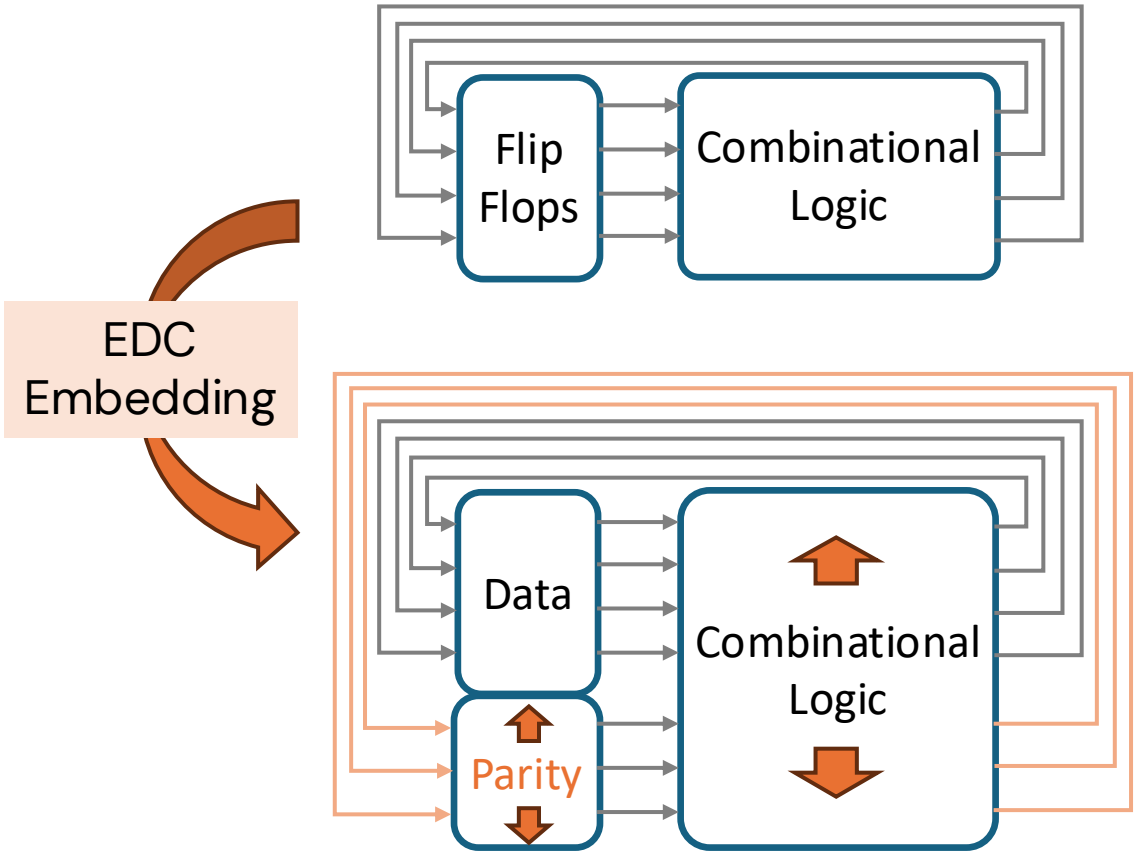


Error Detection Code (EDC) Applied to FSM Encoding

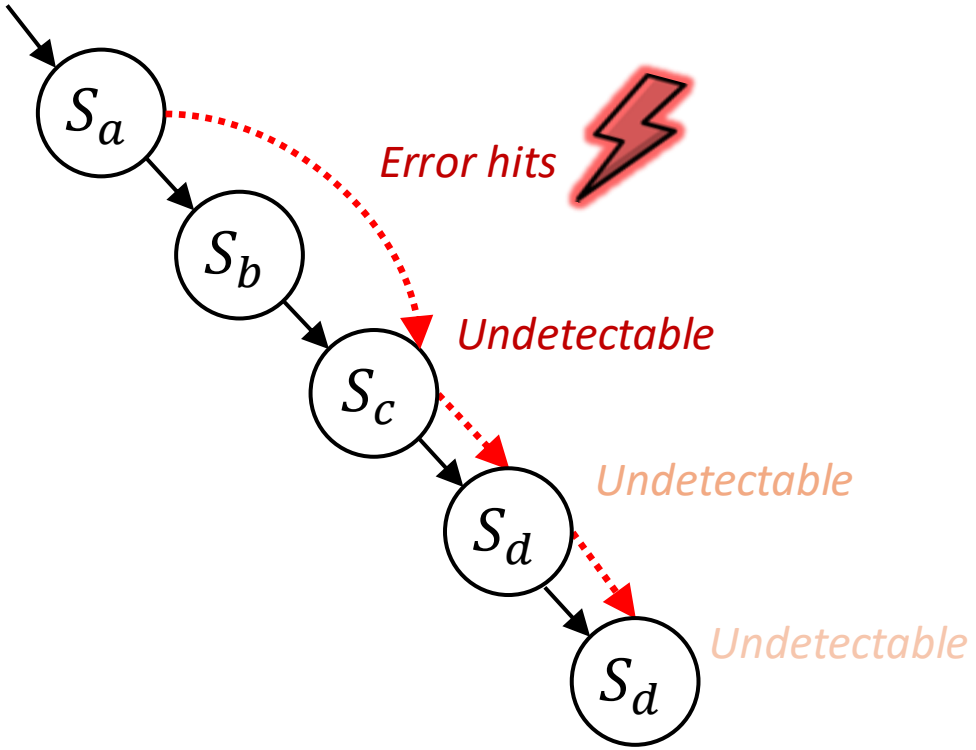


Limitations of EDC FSM Protection

Improved detection probability
= expensive combinational logic

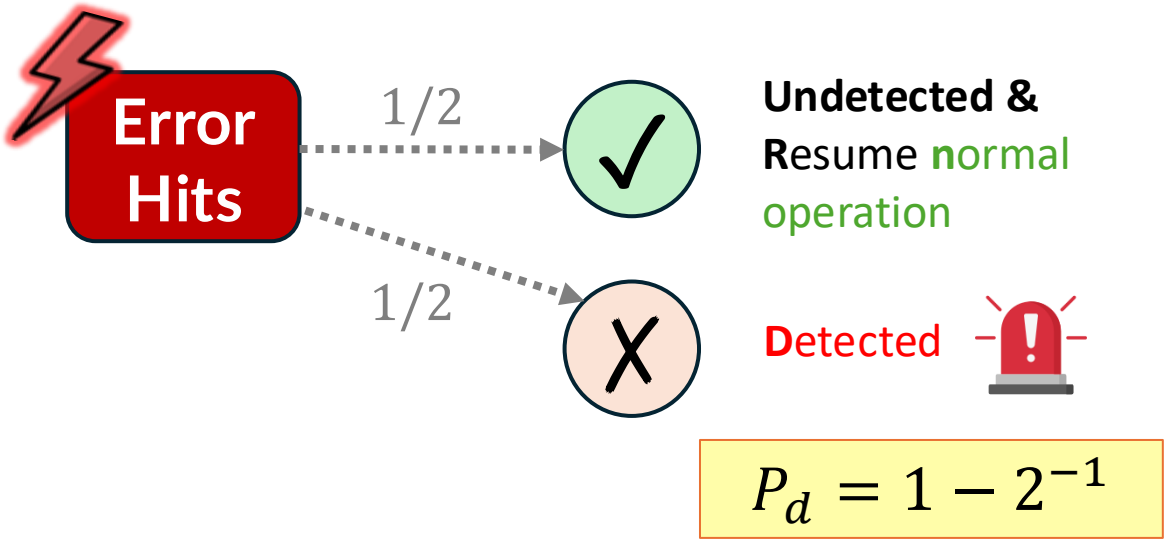


One-shot state error detection



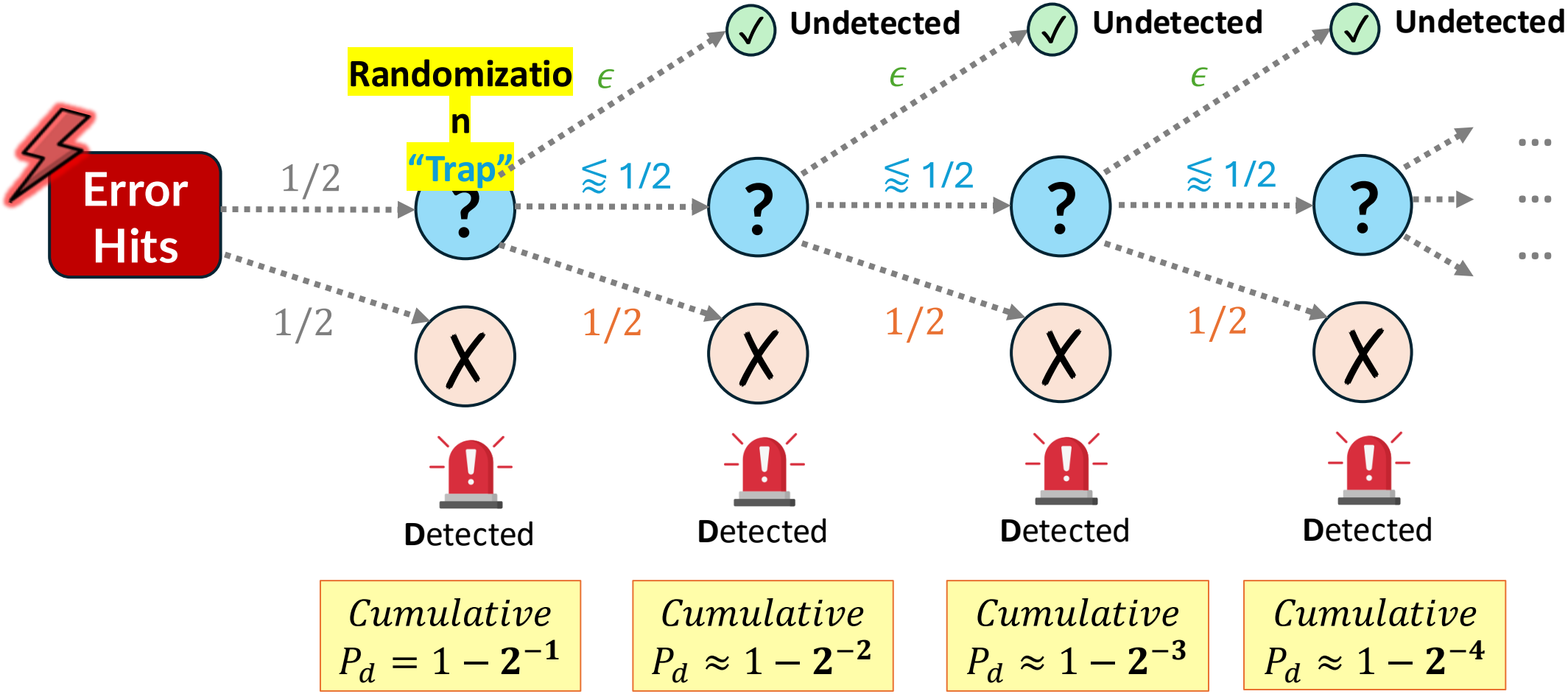
Combined Error Propagation + Detection

Example: **1-bit** EDC



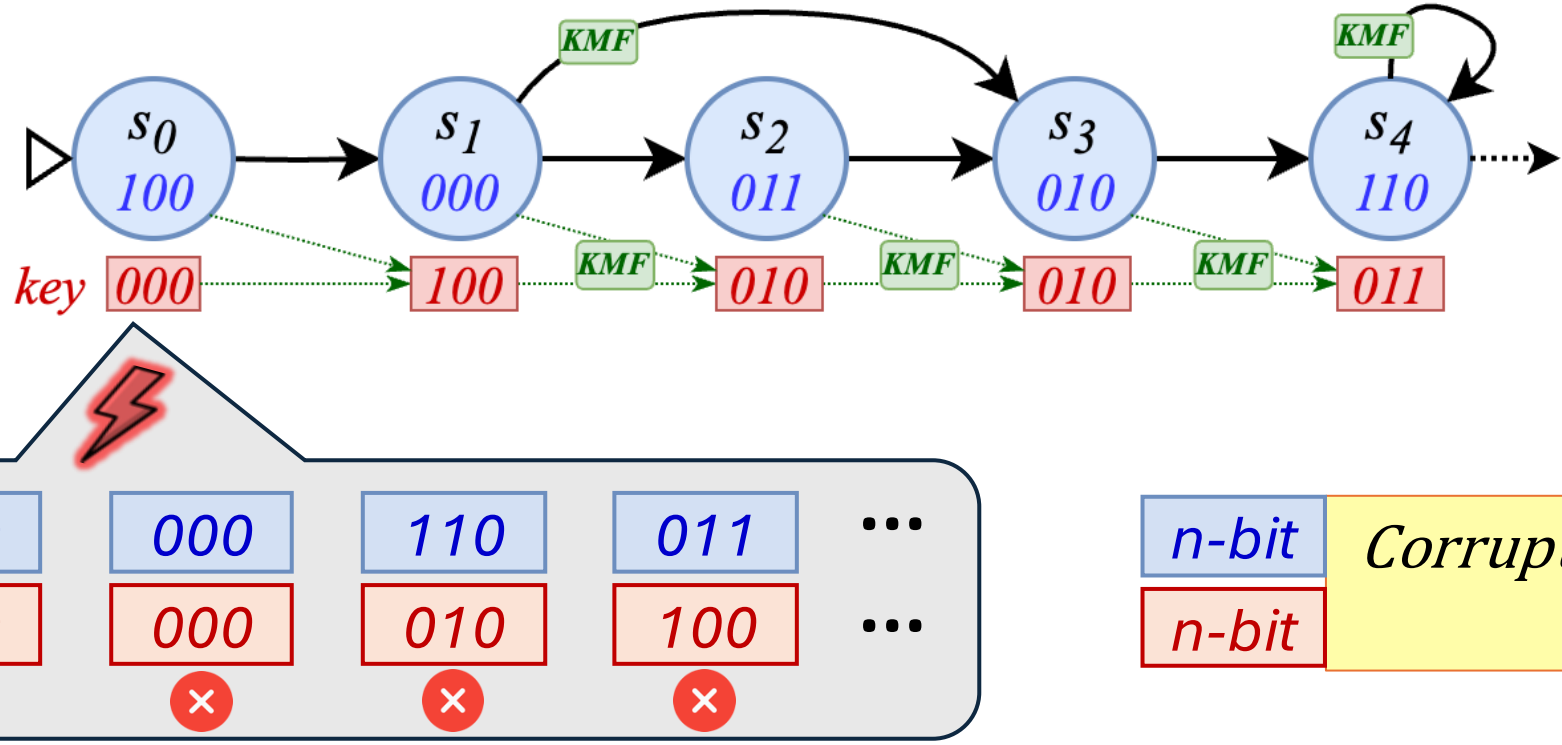
Combined Error Propagation + Detection

Example: **1-bit** EDC



Building Corruption Mode

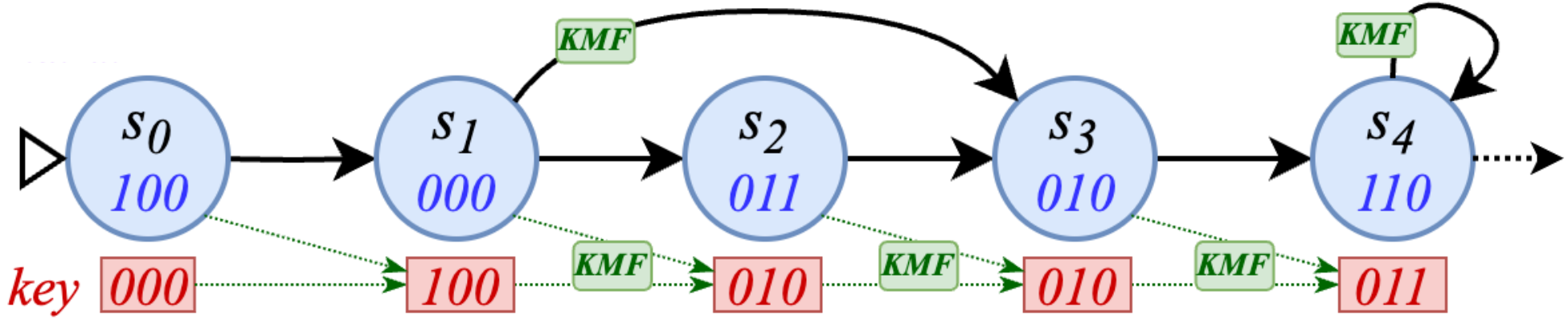
- We already had a way to build broad corruption with WBLO!
  - State + key **match**: Valid & functional
  - State + key **mismatch**: Corrupted

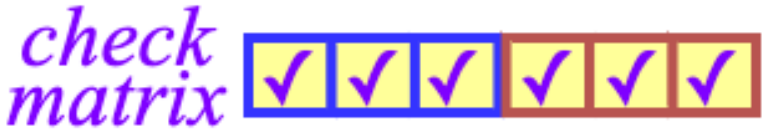


n-bit	*Corruption Probability* $1 - 2^{-n}$
n-bit	

Embed Parity with WBLO

- WBLO imposes certain **key consistency constraints**.
- Often still has *"wiggle room"* to accommodate **parity constraints.**



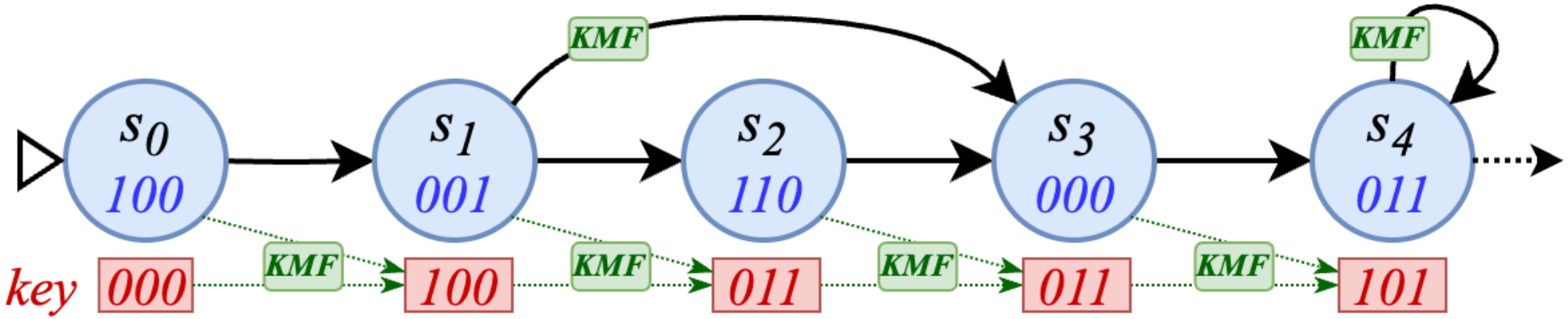


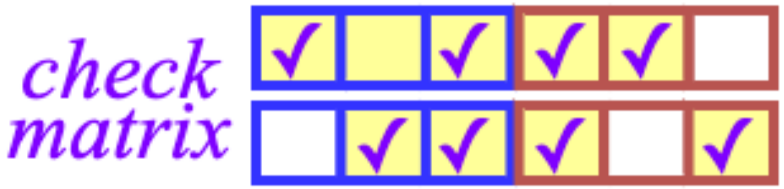
Encoding satisfies key consistency

Encoding satisfies **1-bit parity**

Embed Parity with WBLO

- *"Wiggle room"* can be wide enough to even accommodate more!

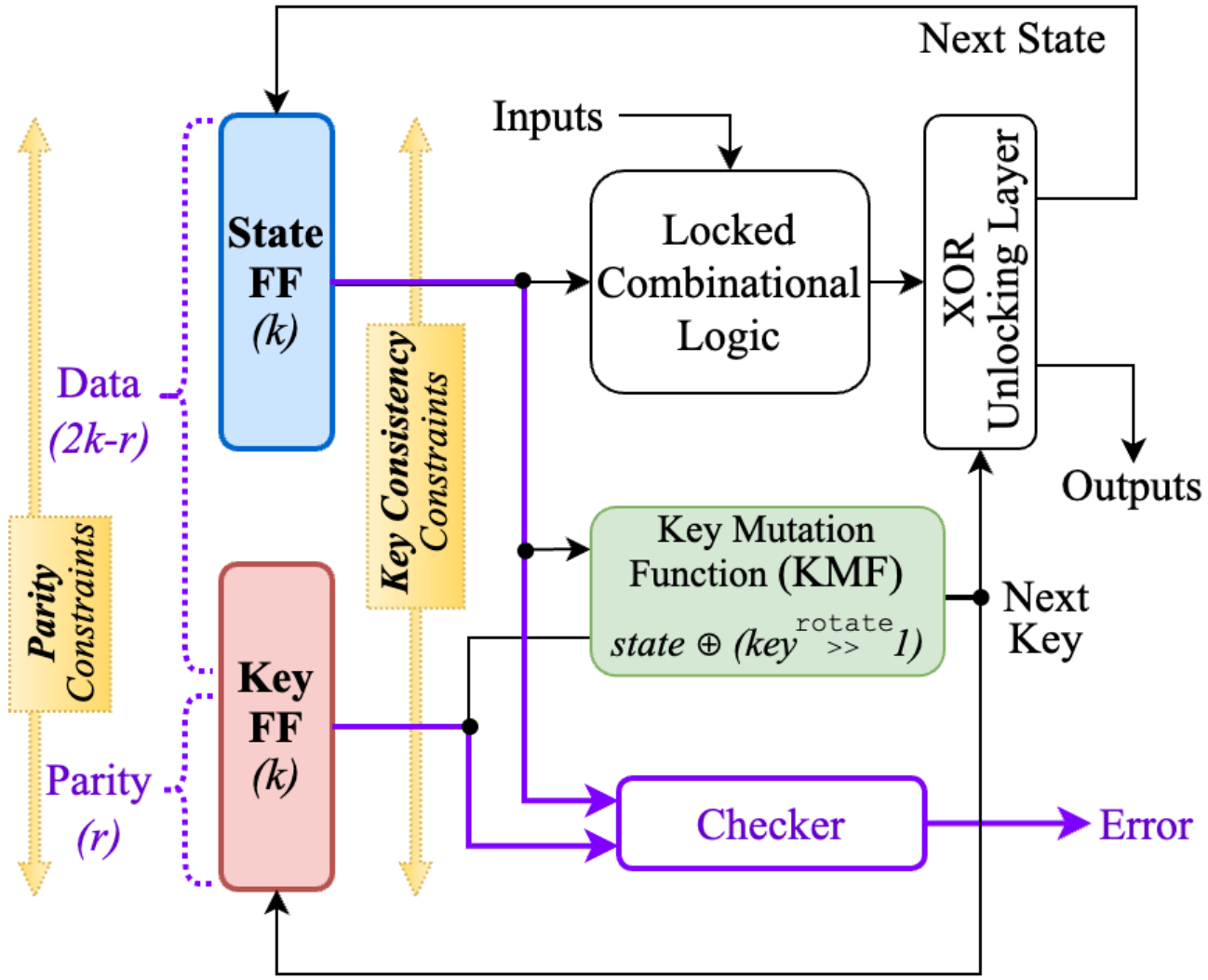




- Encoding satisfies key consistency
- Encoding satisfies **2-bit parity**

Architectural Intuition

- Impose both constraints simultaneously on the encoding bits



Overhead Advantages

Shared Cost

Error Detection Classification

Corruption Classification

	Undetected	Detected
Valid	*No Error*	*Not Exist*
Corrupted	***Latent*** *Error Detection*	***Immediate*** *Error Detection*

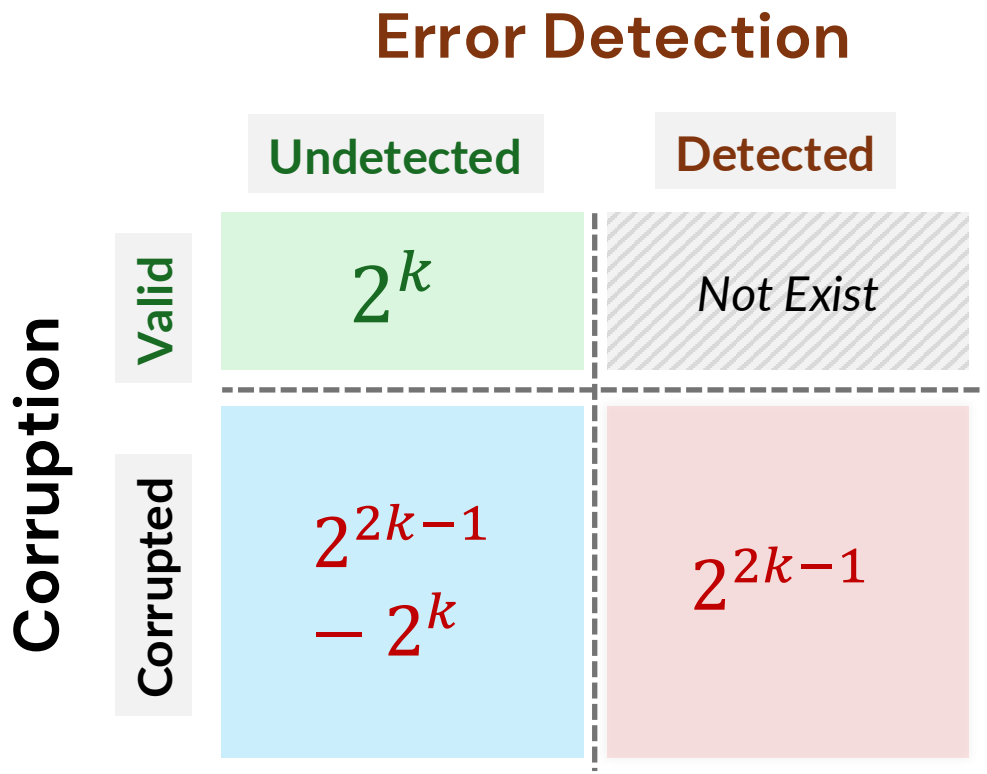
Controlled Corruption Propagation

=

Classical
Error Detection Codes

Analytical Temporal Enhancement

- Suppose a **1-bit parity** is embedded with ***k*-bit encoding** and ***k*-bit key**.



Cumulative detection probability by cycle p

$$P_{d,p} = \sum_{i=0}^{p} P^*_{d,i}, \quad \text{with} \quad P^*_{d,i} = \left(\frac{1}{2} - 2^{-k}\right)^i \cdot \frac{1}{2}$$

At **infinite p** → Approximate $k-1$ effective parity bits

$$P_{d,\infty} = \sum_{i=0}^{\infty} \left(\frac{1}{2} - 2^{-k}\right)^i \cdot \frac{1}{2} = \frac{2^k}{2^k+2} = \frac{2^{k-1}}{2^{k-1}+1}$$

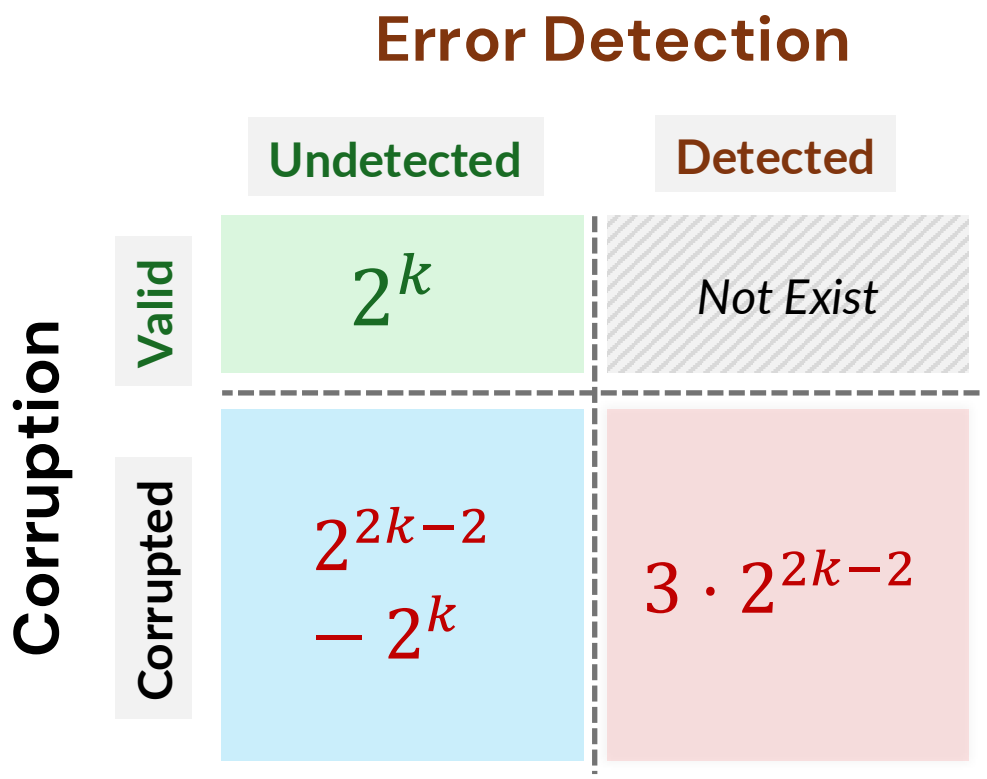
$$= 1 - \frac{1}{2^{k-1}+1} > 1 - 2^{-(k-1)}$$

At **bounded** $p = k$ → Approximate $k-2$ effective parity bits

$$P_{d,k} > 1 - 2^{-(k-2)} - 2^{-(k+1)}$$

Analytical Spatial Enhancement

- Suppose a **2-bit parity** is embedded with k**-bit encoding** and k**-bit key**.



1-bit parity + infinite p: $\approx k - 1$ effective parity bits

$$P_{d,\infty} = 1 - \frac{1}{2^{k-1}+1} > 1 - 2^{-(k-1)}$$

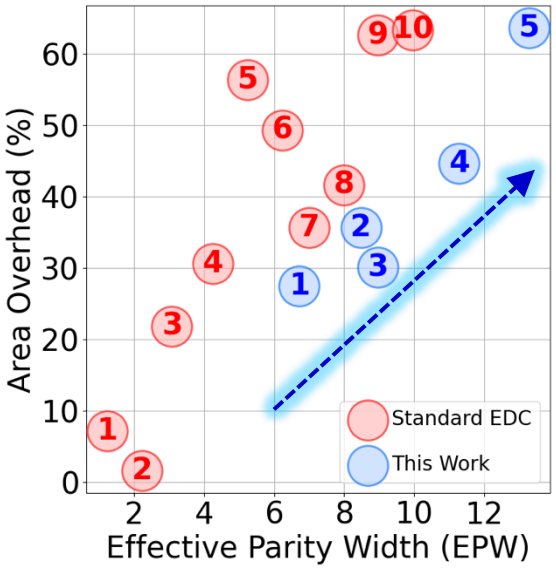
2-bit parity + infinite p: $\approx k - 0.4$ effective parity bits

$$P_{d,\infty} = \sum_{i=0}^{\infty} \left[\frac{1}{4} - 2^{-k}\right]^i \cdot \frac{3}{4} = \frac{3 \cdot 2^k}{3 \cdot 2^k + 4}$$

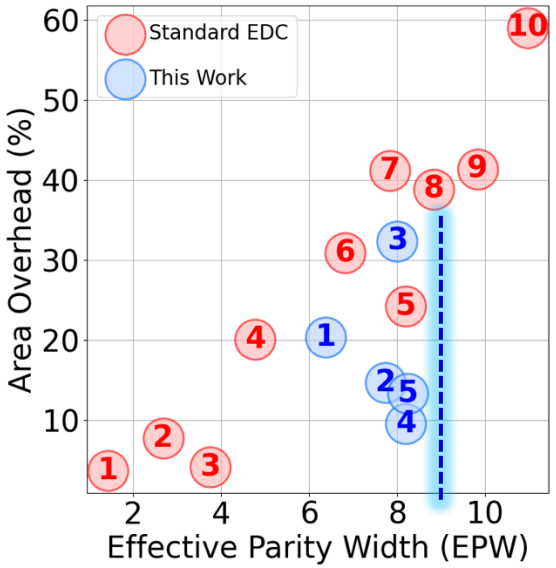
Experimental Results

- **Significant Pareto frontier** shift in the space of **area & Effective Parity Width (EPW)**

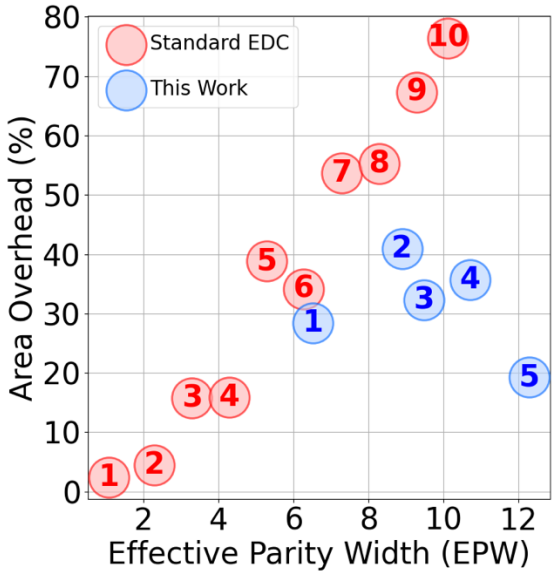
s420_12



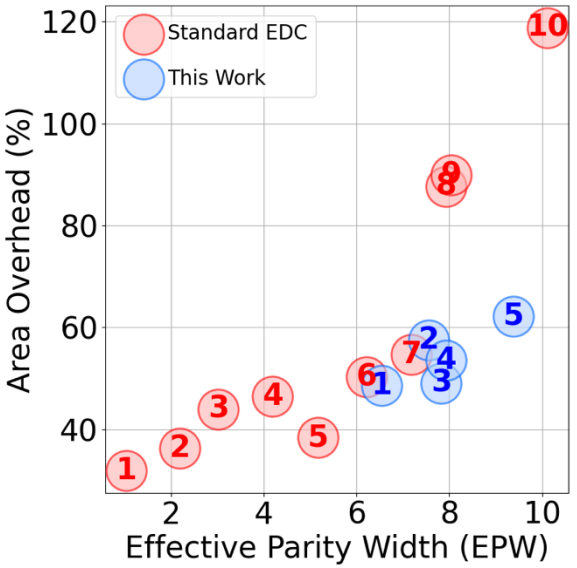
s510_6



bbse_32



ex4_32

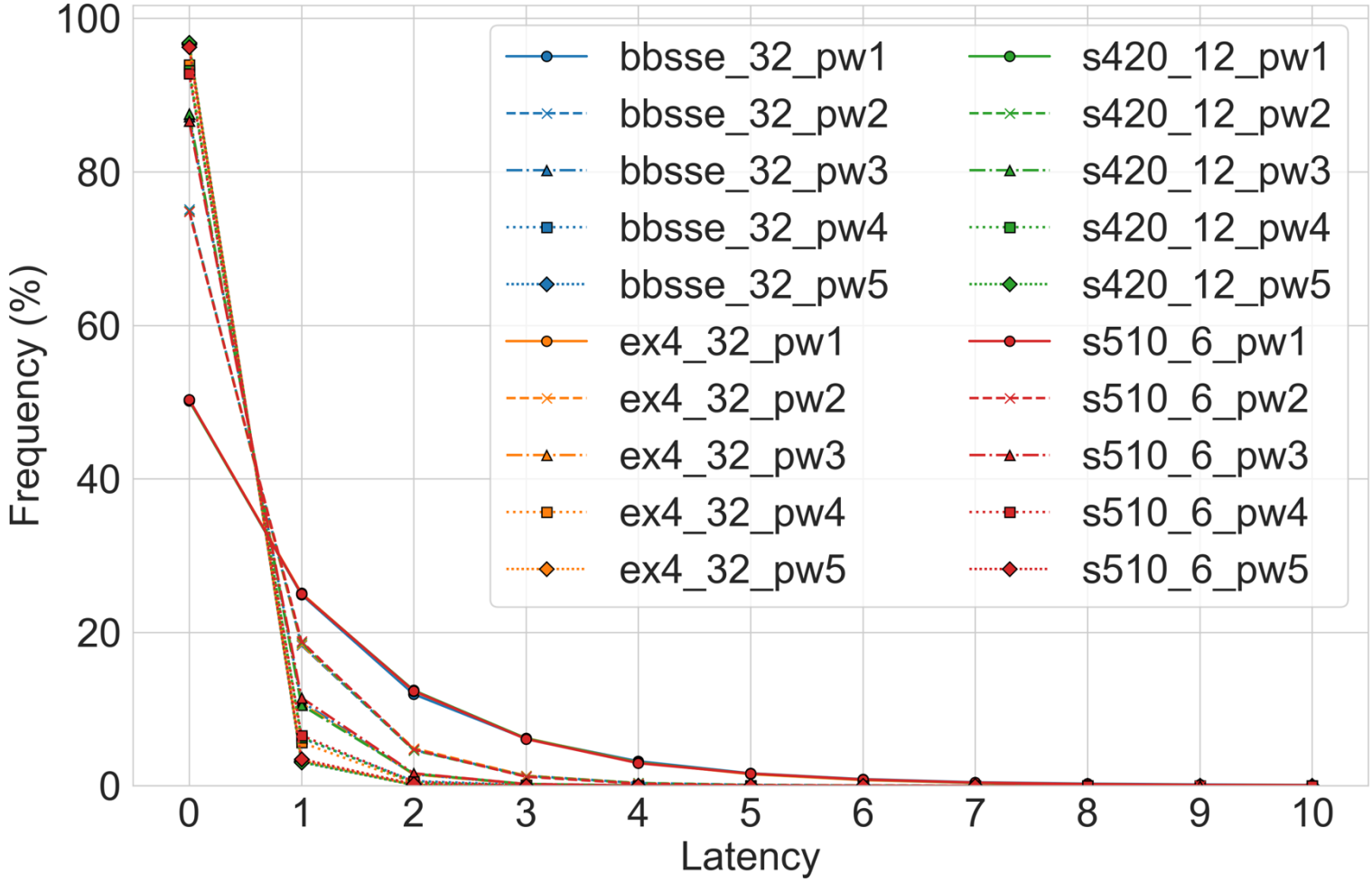


Increased encoding width required to accommodate more parity boosts EPW

Weaker key consistency constraints left plenty of room for parity embedding without encoding width increase

Experimental Results

- Increased parity width reduces detection latency
- Latency averages out to **less than 1 cycle** in all cases



Summary

Hardware IP Security | **Reliability**

T1 Structural Analysis of Logic Locking

T2 Hybrid Internal-External Locking Control in FSMs

T3 White-box Logic Obfuscation

T4 Temporally Enhanced Error Detection

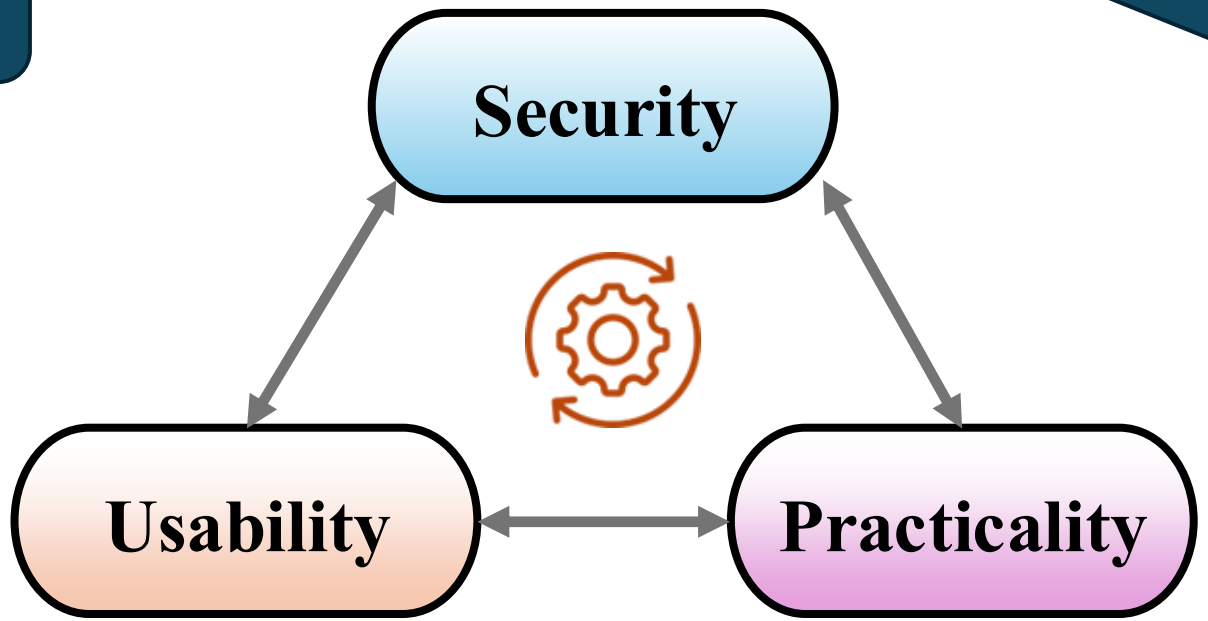
Conclusion

Vulnerability channel definition

Attack capability assessment

Structural weakness is as important as functional

White-box security is possible for the right adversarial target



Equivalence-preserving transformations are essential for enabling greater flexibility in design-for-security.

Algorithm based on design characteristic studies eases implementation overheads

Correctness constraints

Trade-off optimization

Scalability of algorithm

Integration with industrial flow