PCH Framework for IP Runtime Security Verification

Xiaolong Guo*, Raj Gautam Dutta†, Jiaji He‹, and Yier Jin*

*Department of Electrical and Computer Engineering, University of Florida
†Department of Electrical Engineering and Computer Science, University of Central Florida
‹School of Microelectronics, Tianjin University
guoxiaolong@ufl.edu, rajgautamdutta@knights.ucf.edu, dochejj@tju.edu.cn, yier.jin@ece.ufl.edu

Security in Silicon Lab (SSL)

UF UNIVERSITY of FLORIDA

UCF

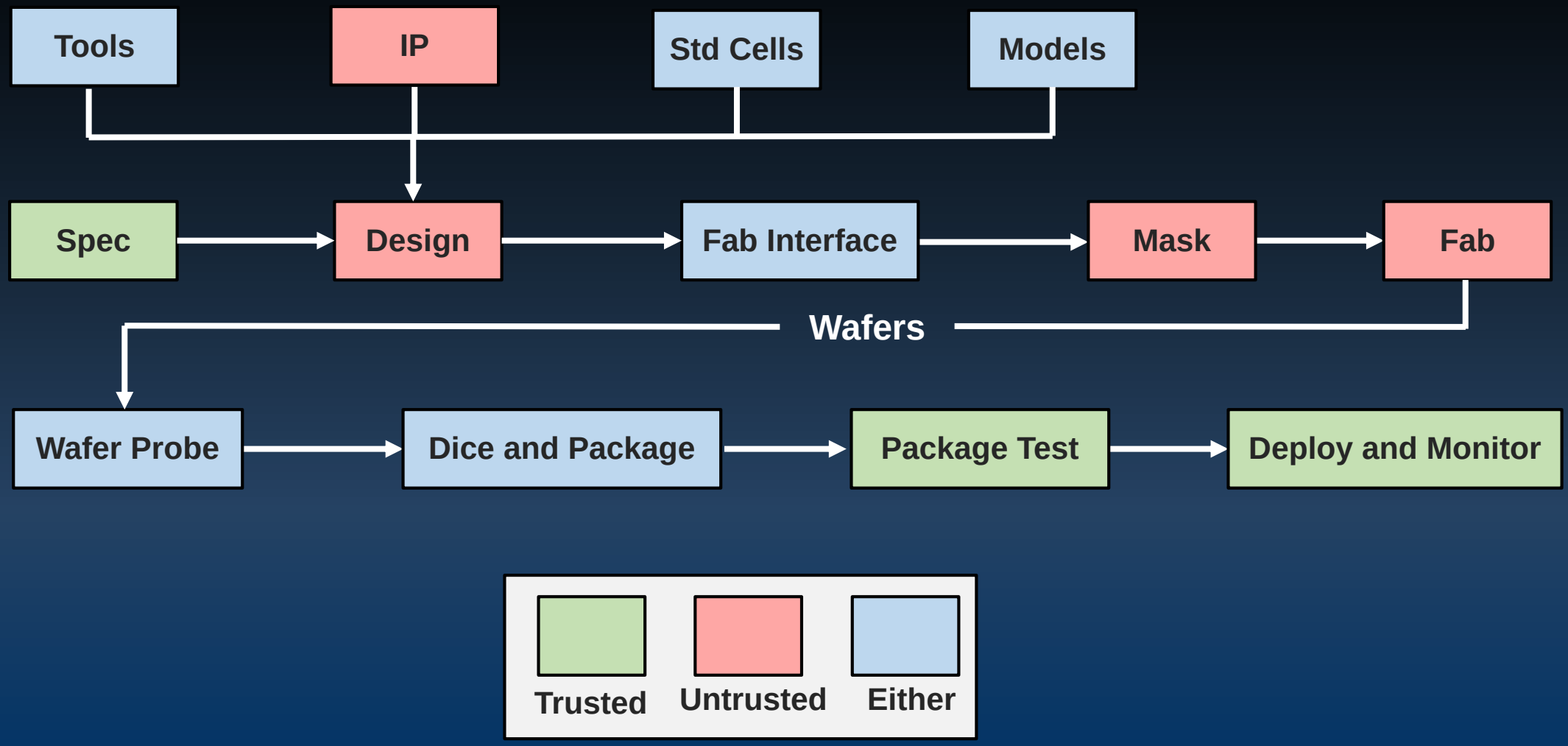
天津大学 Tianjin University

Contents

- Introduction
  - Threat Model
  - Related Methods
  - Background
- Runtime Proof-Carrying Hardware (PCH)
  - Runtime PCH Framework
  - Runtime Proof-Carrying
  - Verifier Design
- Case Study and Results
- Conclusions and Future Work

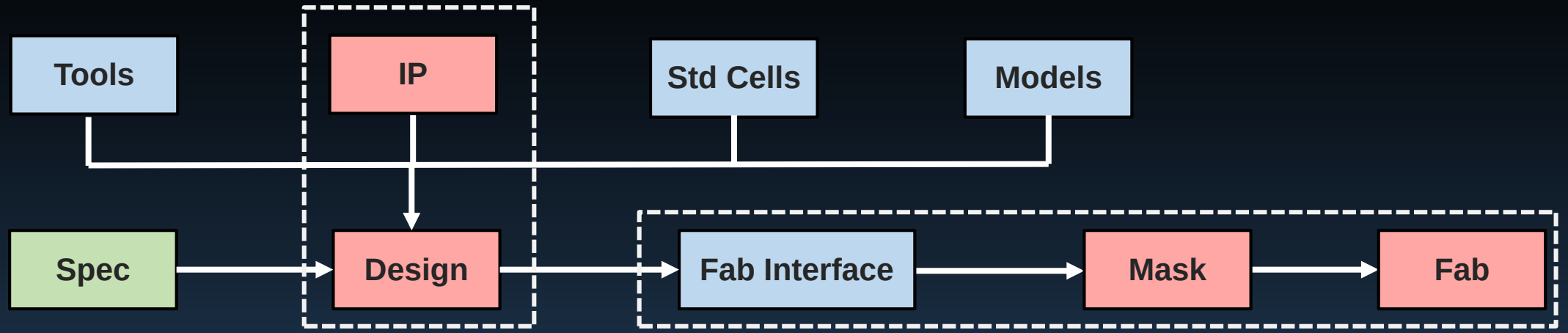
Vulnerabilities of IC Supply Chain

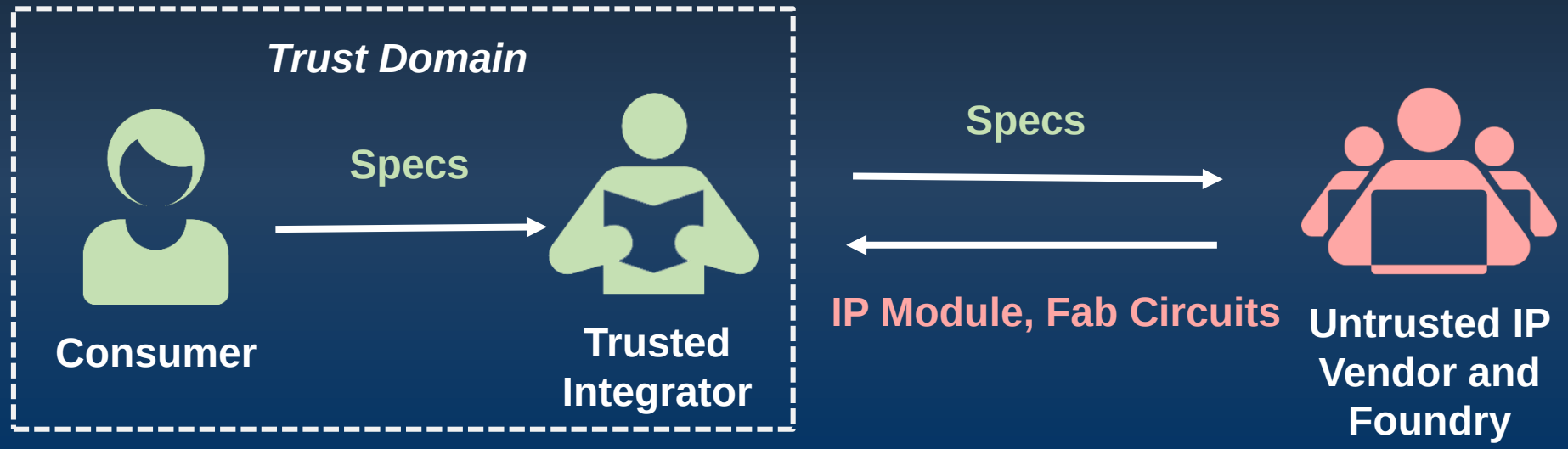
- Current IC Supply Chain



Vulnerabilities of IC Supply Chain

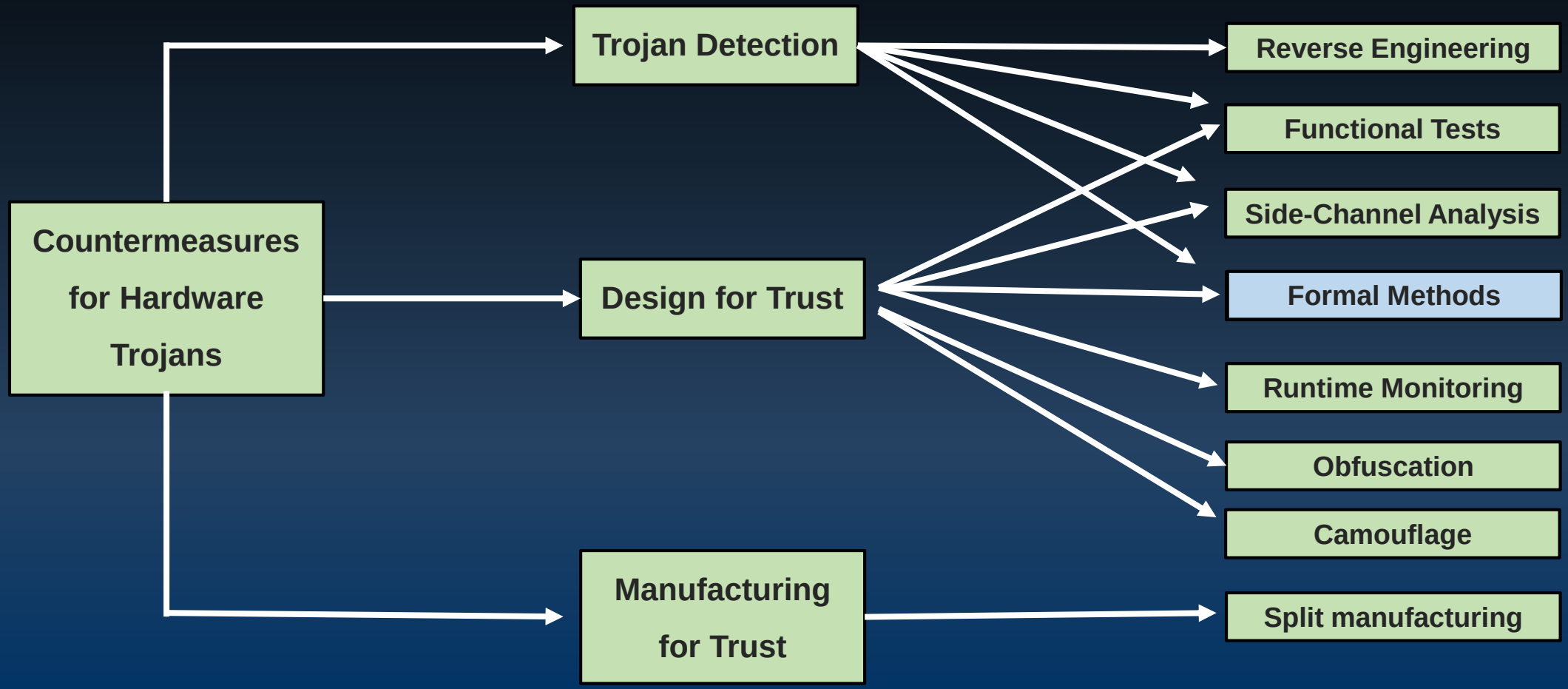
- Untrusted IP Vendor and Foundry





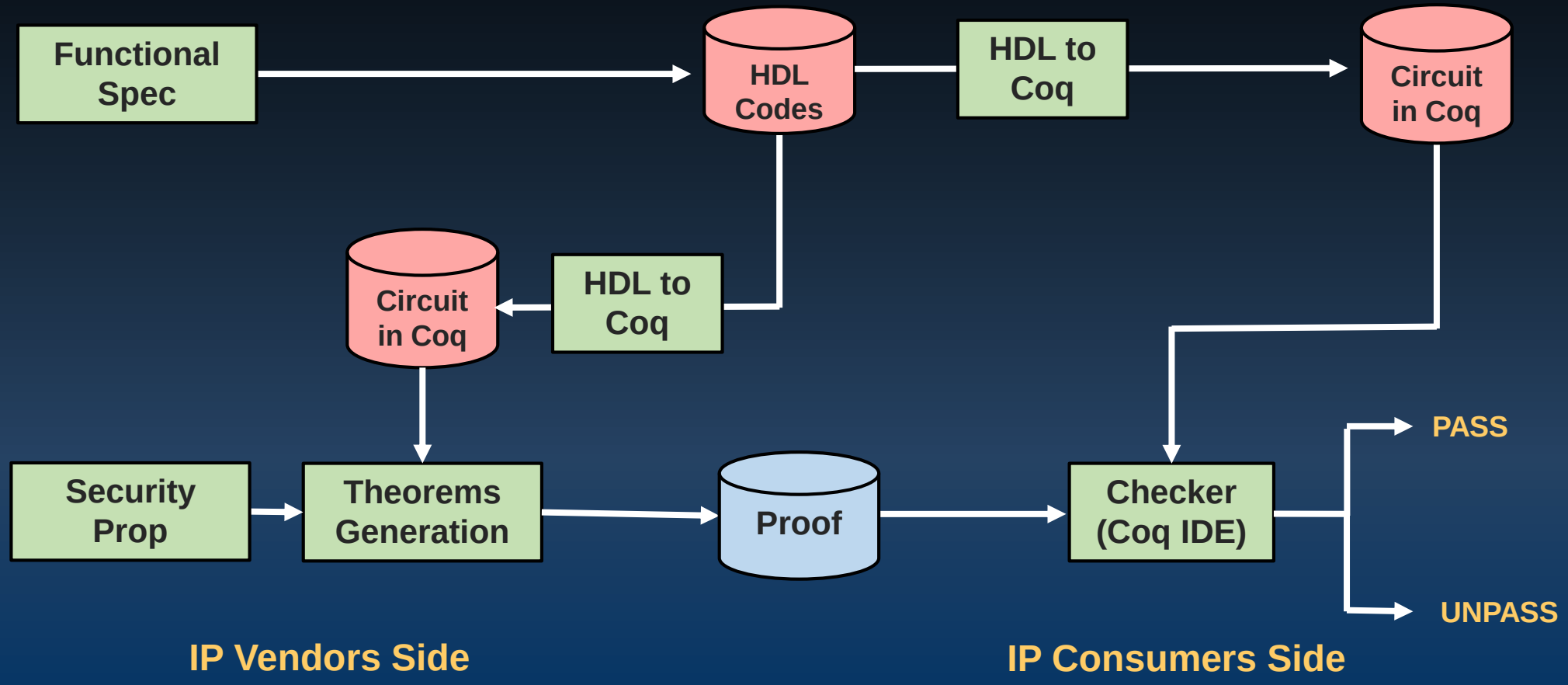
Hardware Trojan Countermeasures

- Existing countermeasures:
  - Designed to provide protection in certain scenarios.



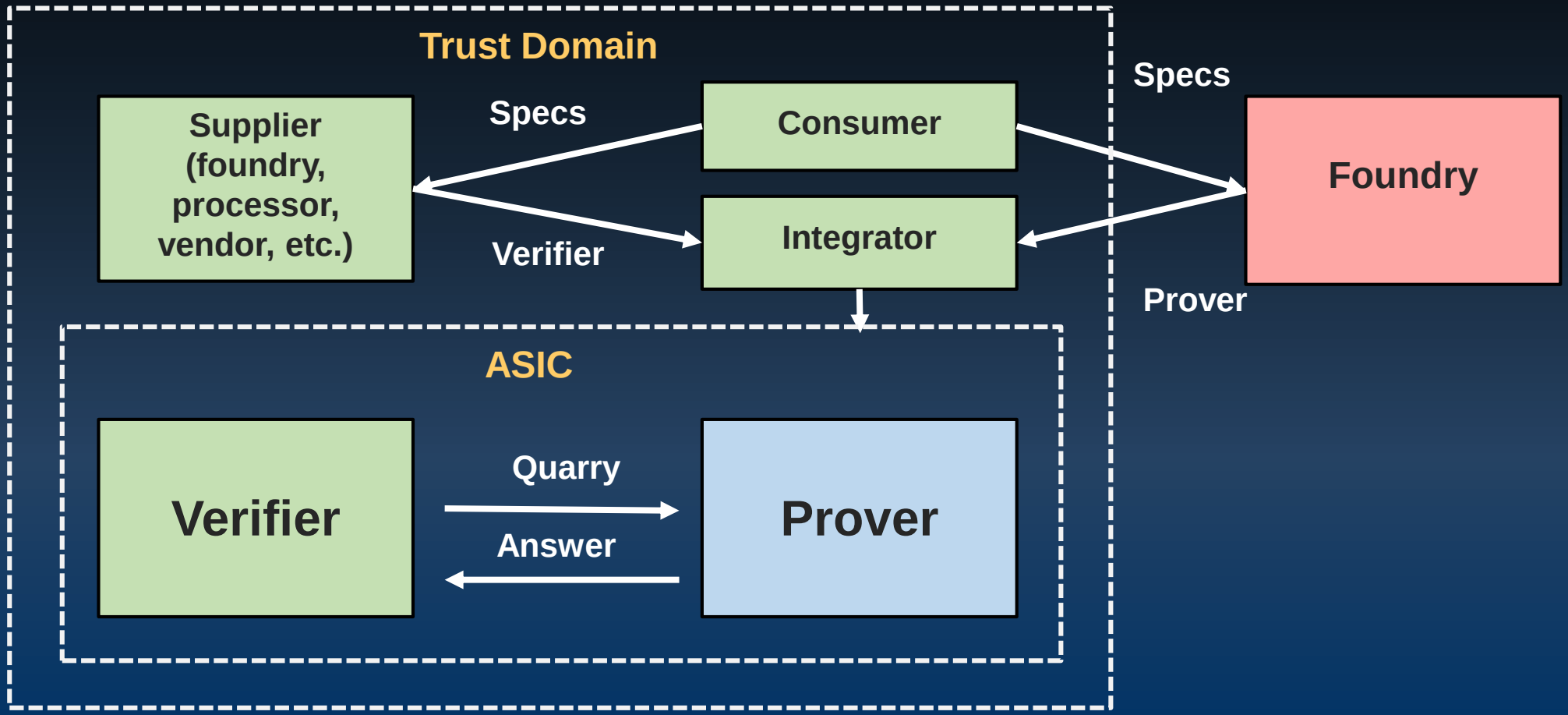
Formal Verifications - PCH

- Static Formal Verification: Proof-Carrying Hardware (PCH)
  - Only provides static verification in design stage



Formal Verifications – Verifiable ASICs

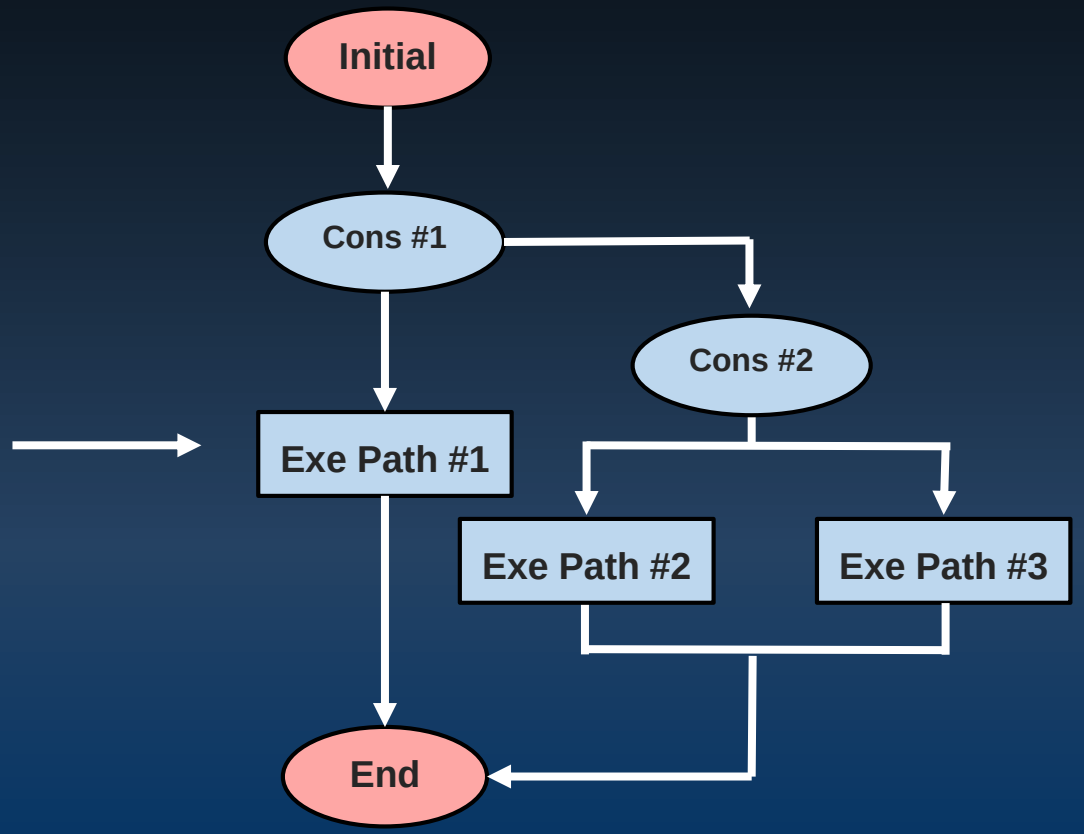
- Runtime Formal Verification: Verifiable ASICs
  - Prover-Verifier architecture
  - High computational cost and overhead; functional properties.



Backgrounds – Symbolic Execution

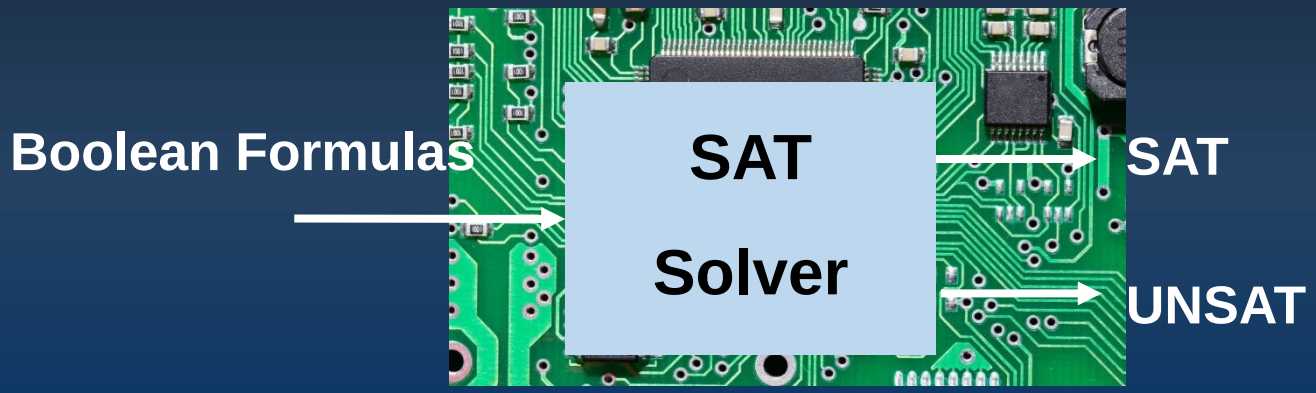
- Symbolic execution
  - A program analysis technique
  - Tools: KLEE, JPF, S2E, etc.
  - Execution paths generated from program
  - Properties check

```
int foo(int x){
        int y = 0;
        If (x == 0)
                Return y;
        if (x < 256){
                y = 256/x;
        } else {
                y = 1;
        }
        return x;
}
```



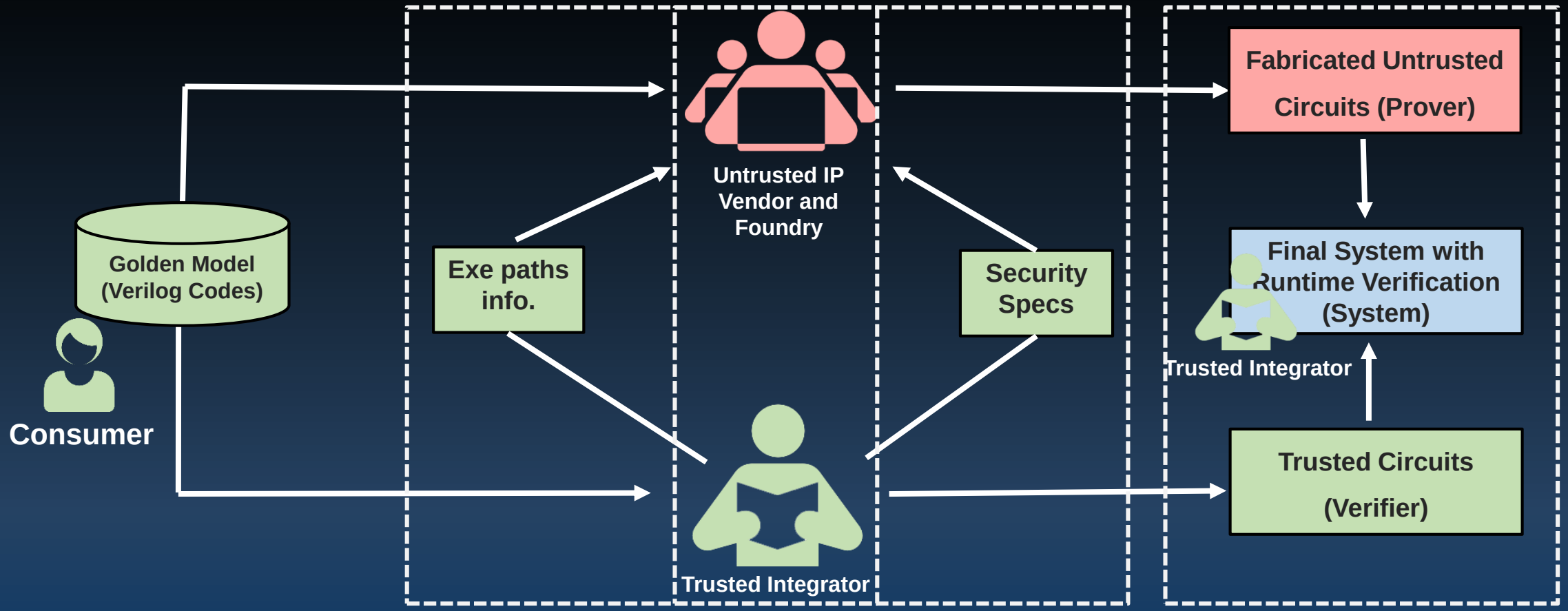
Backgrounds – SAT Solver

- Satisfiability (SAT) solver
  - Solve Boolean Satisfiability problem
    - Input: conjunctive normal form (CNF)
    - Output: SAT/UNSAT
  - Tools: MiniSAT, ManySAT, March_dl, etc.
  - Hardware accelerated SAT solver based on FPGA/GPU
  - Davis-Putnam-Logemann-Loveland (DPLL) Algorithm



Runtime PCH Framework

- Working procedure of runtime PCH framework



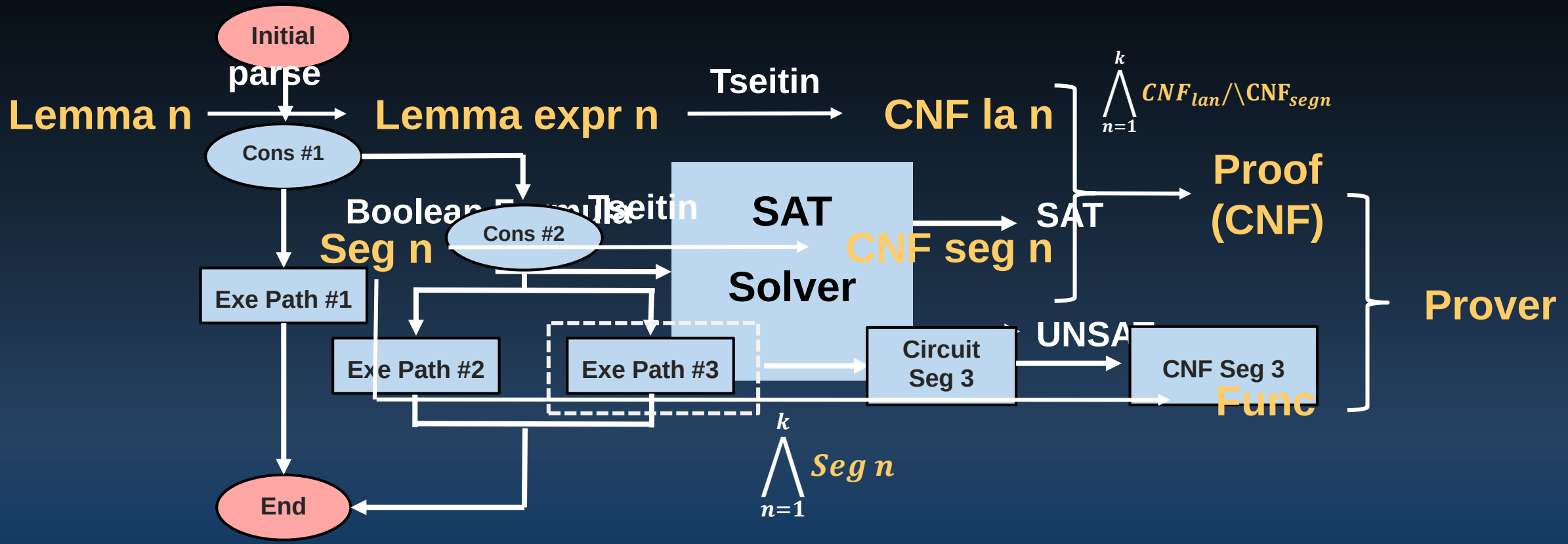
Func := Seg 1 /\ Seg 2 /\.../\ Seg k

Prop := Lemma 1 /\ Lemma 2 /\.../\ Lemma k

System := Prover /\ Verifier

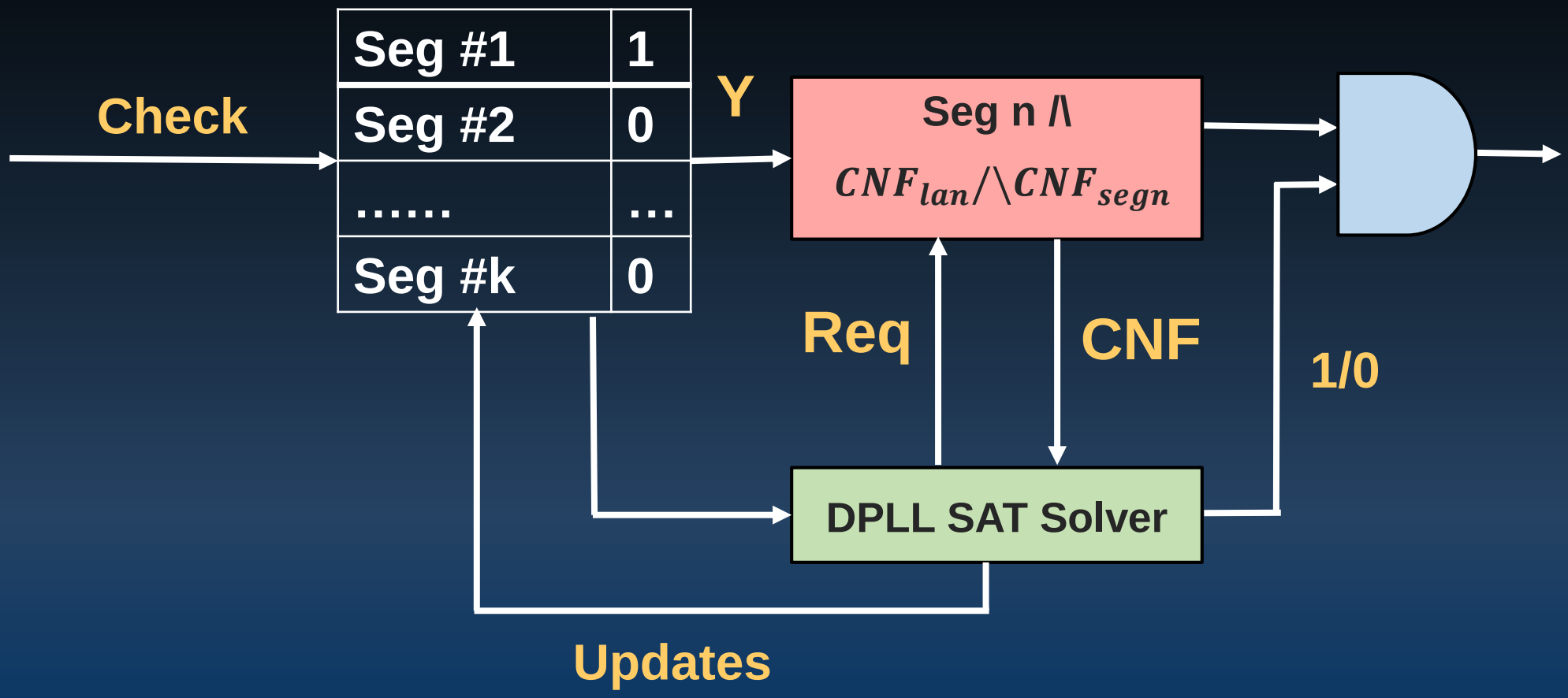
Runtime PCH - Prover

- Runtime Proof-Carrying



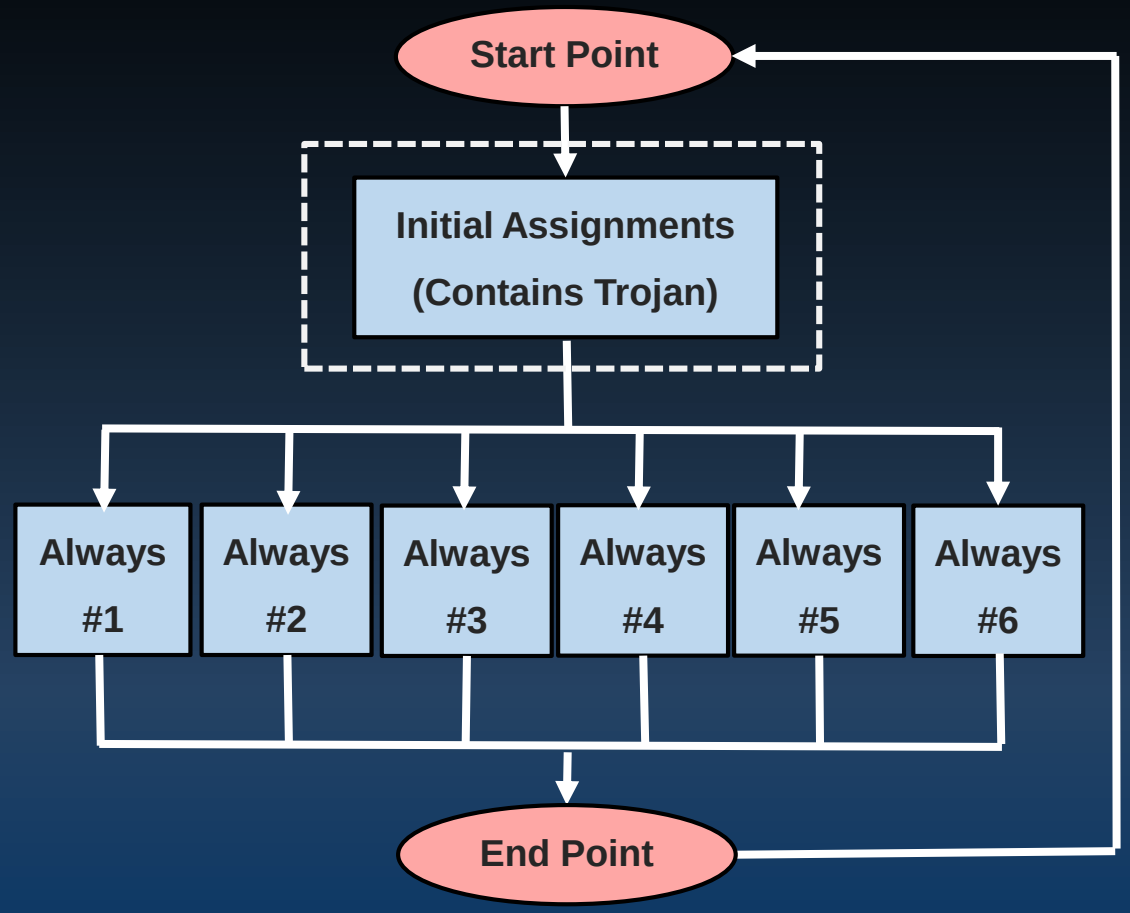
Runtime PCH - Verifier

- Design of Verifier



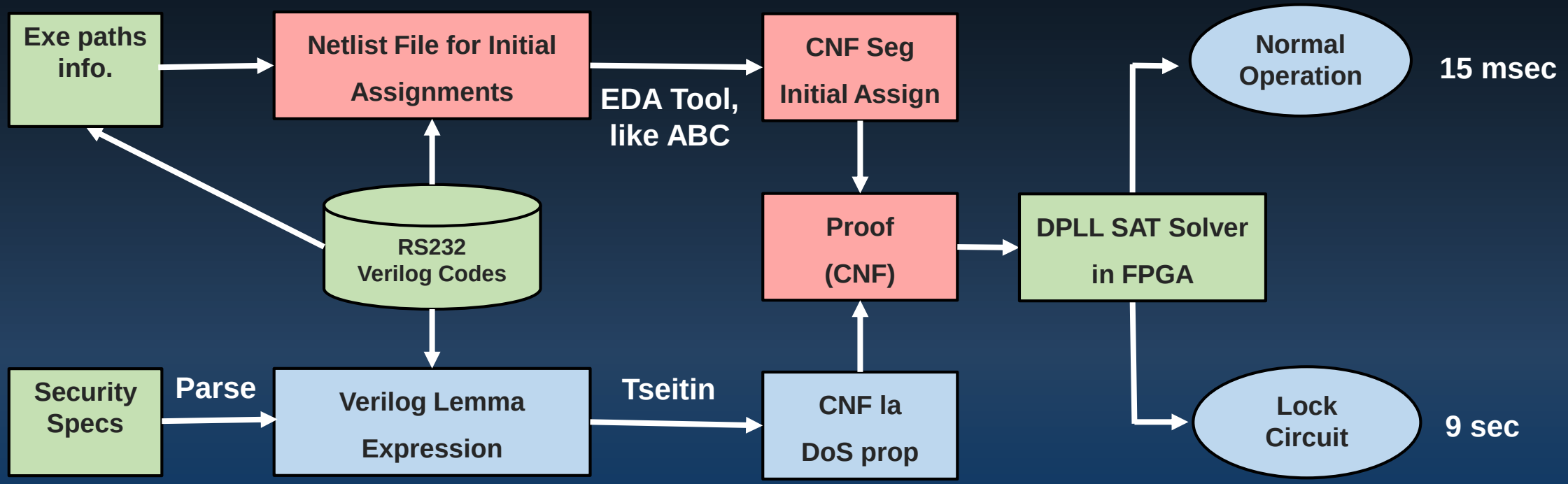
Case Study and Results

- Case study setup
  - Platform: FPGA KC-705, Kintex 7
  - Benchmark: RS232 – T100
  - Hardware Trojan embedded
    - Trigger signal: *state*, *bitCell_cntrH*, *recd_biCntrH*, *rec_dataH*
    - Payload - DoS: output signals *rec_dataH* and *rec_readyH* are set to zeros.



Case Study and Results

- Segment verification
  - Prop (in natural language): in no way the output will keep generating zero regardless of what input is.



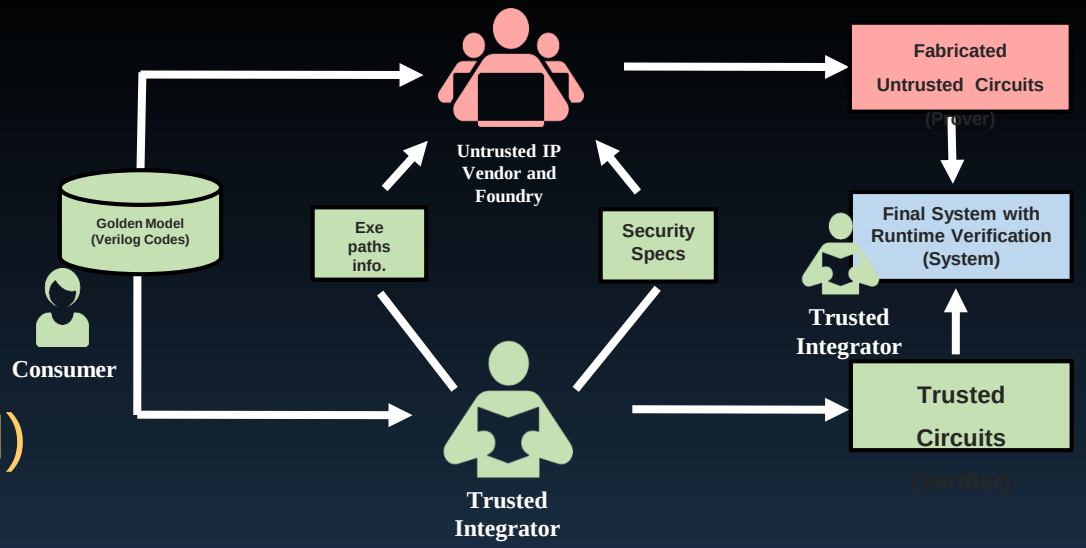
Conclusion and Future Work

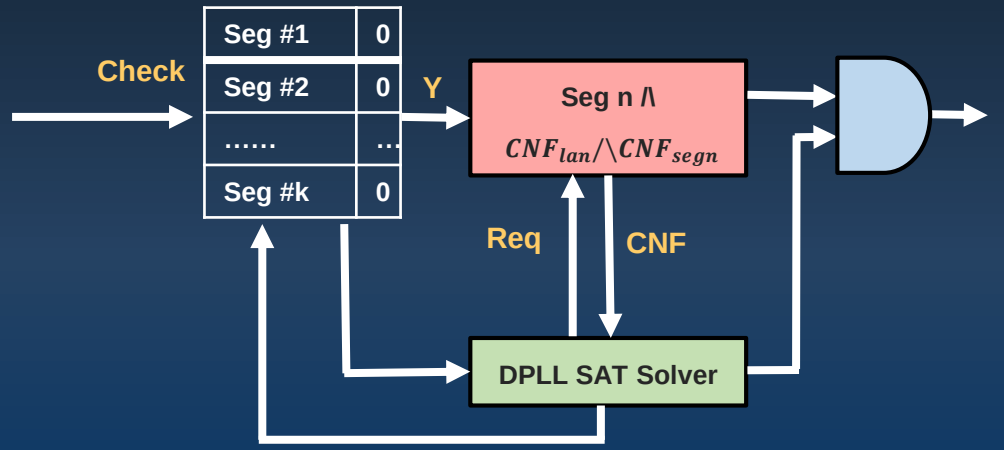
- Conclusion
  - A solution to hardware runtime formal verification for secure purpose
  - An attempt to apply symbolic execution in hardware security

- Future work
  - Apply the approach into large scale hardware system
  - SAT solver optimization
  - Automated tool development

Questions?

- Introduction
  - Threat Model
  - Related Methods
  - Background
- Runtime Proof-Carrying Hardware (PCH)
  - Runtime PCH Framework
  - Runtime Proof-Carrying
  - Verifier Design
- Case Study and Results
- Conclusions and Future Work





Thanks!

Xiaolong Guo, guoxiaolong@ufl.edu

Security in Silicon Lab (SSL)

UF UNIVERSITY of FLORIDA

UCF

天津大学 Tianjin University