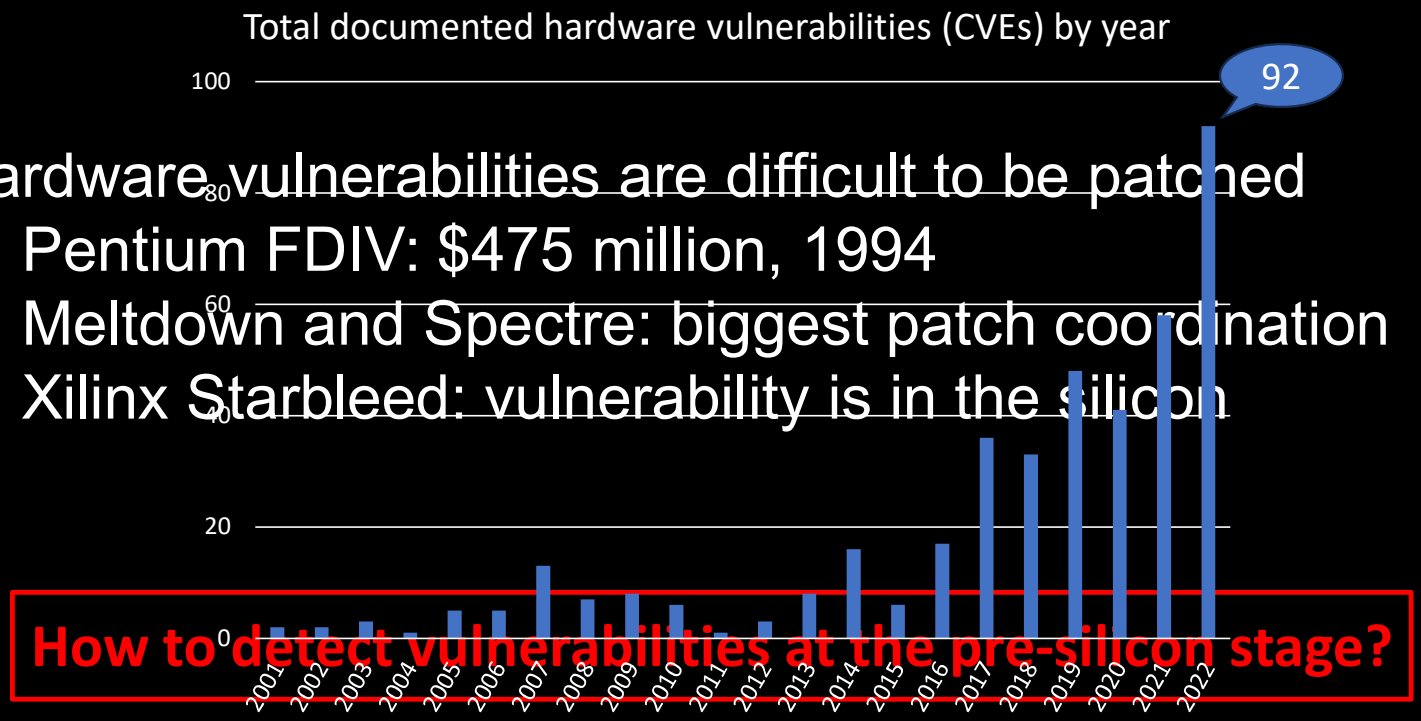
HyPFuzz: Formal-Assisted Processor Fuzzing

Chen Chen[†], Rahul Kande[†], Nathan Nguyen[†], Flemming Andersen[†],
Aakash Tyagi[†], Ahmad-Reza Sadeghi[*], Jeyavijayan Rajendran[†]

[†]Texas A&M University, College Station, USA,
[*]Technische Universität Darmstadt, Germany

Massive Growth in Hardware Vulnerability

- Hardware vulnerabilities emerge at an alarming rate



- Hardware vulnerabilities are difficult to be patched
  - Pentium FDIV: $475 million, 1994
  - Meltdown and Spectre: biggest patch coordination
  - Xilinx Starbleed: vulnerability is in the silicon

Source: National Vulnerability Database NVD (08/2023)
https://www.csoonline.com/article/567525/hardware-and-firmware-vulnerabilities-a-guide-to-the-threats.html

Existing Hardware(HW) Fuzzers

Technique	Coverage Metric	Largest Design	Vulnerability
RFUZZ	Select signals of some MUXs	5-stage Sodor core	0
DifuzzRTL	Registers driving select signals of some MUXs	Boom	16
HyperFuzzing	Inserted properties	SHA crypto engine	0
Trippel et al.	SW FSM, line, edge, HW toggle, functional	KMAC	5
TheHuzz	HW FSM, line, edge, branch, toggle	Boom	11

Limitations of Existing HW Fuzzers

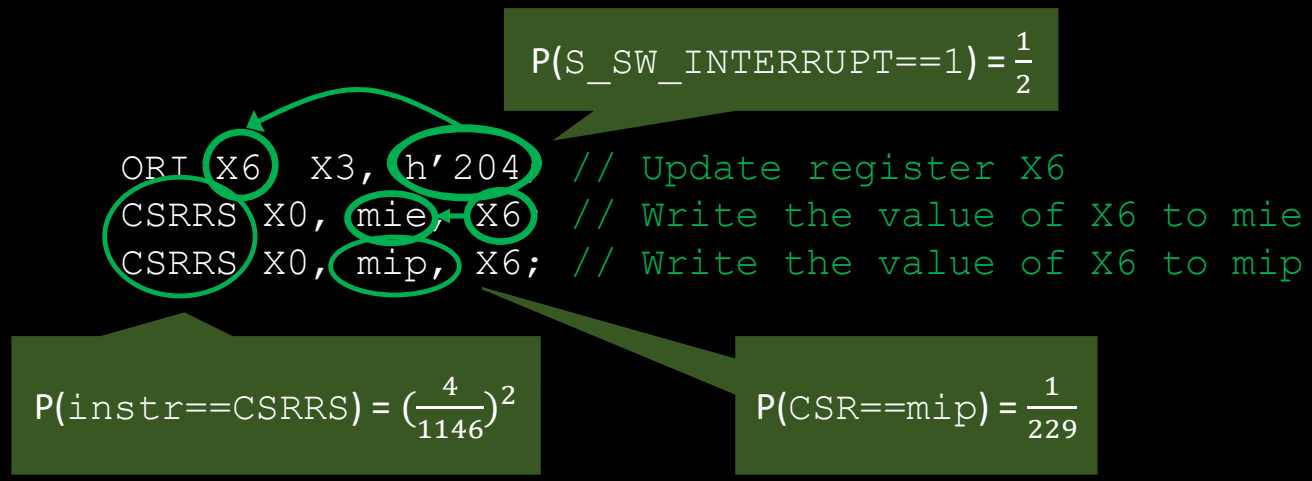
- Hard-to-reach design spaces due to specific conditions
- Example:

```
// CVA6 Interrupt handler
if (mie[S_TIMER_INTERRUPT] && mip[S_TIMER_INTERRUPT])
    interrupt_cause = S_TIMER_INTERRUPT; // Supervisor Timer
if (mie[S_SW_INTERRUPT] && mip[S_SW_INTERRUPT])
    interrupt_cause = S_SW_INTERRUPT; // Supervisor Software
if (mie[S_EXT_INTERRUPT] && (mip[S_EXT_INTERRUPT] |
       irq[SupervisorIrq]))
    interrupt_cause = S_EXT_INTERRUPT; // Supervisor External
```

After 72 hours, 200K tests, *TheHuzz* did not trigger it

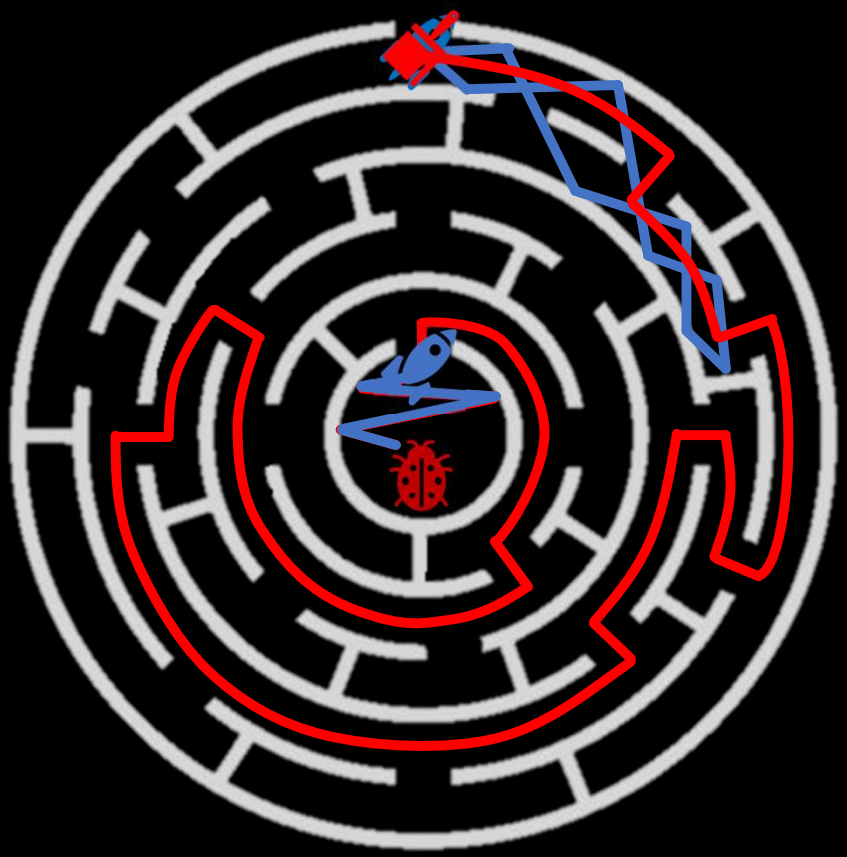
Probability of Triggering the Branch

```
If (mie[S_SW_INTERRUPT] && mip[S_SW_INTERRUPT])
    interrupt_cause = S_SW_INTERRUPT; // Supervisor Software
```



Overall P(Branch==True) = 1.16×10^{-10}

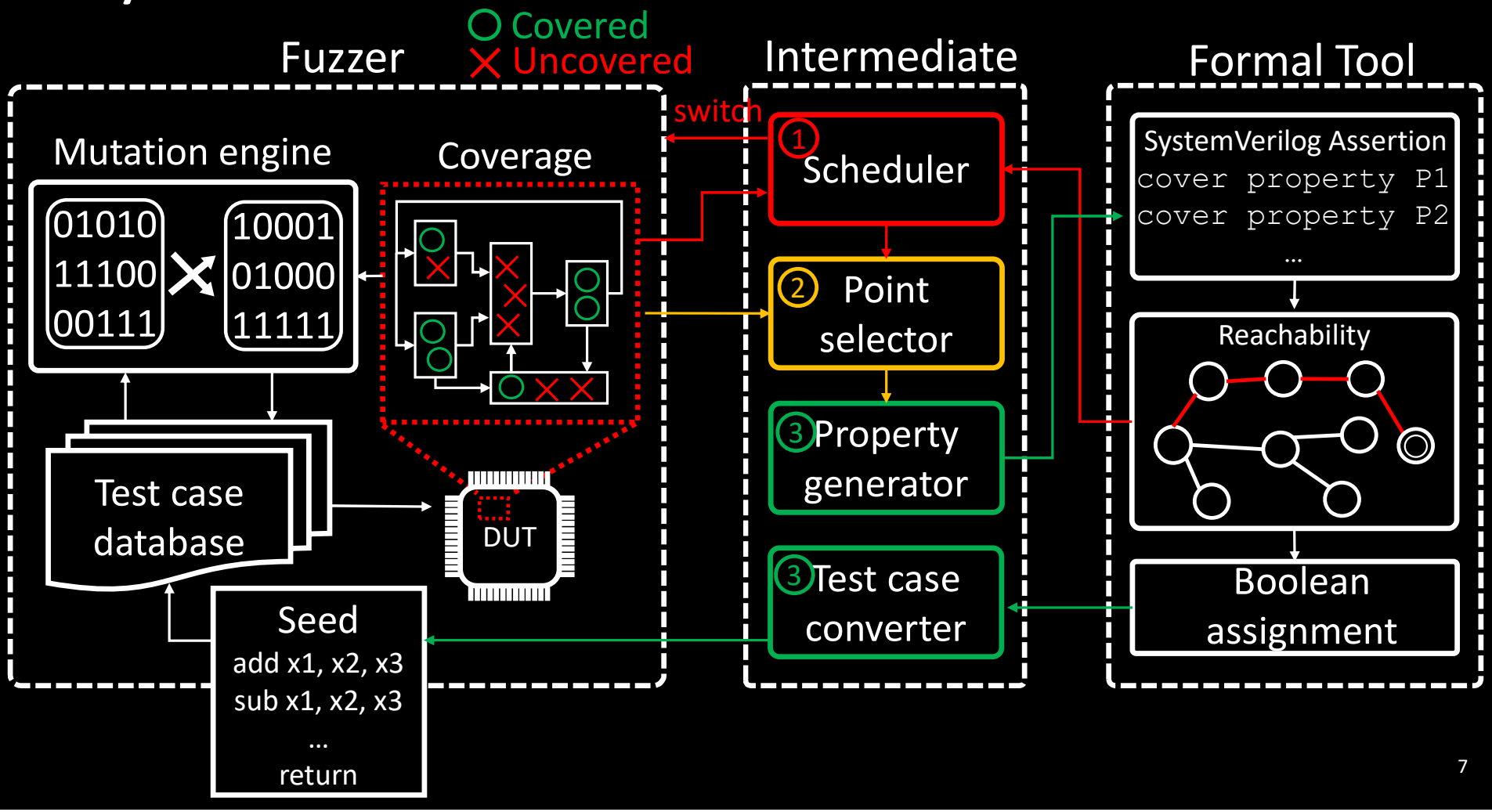
Hybrid Fuzzing: Fuzzing + Formal Verification



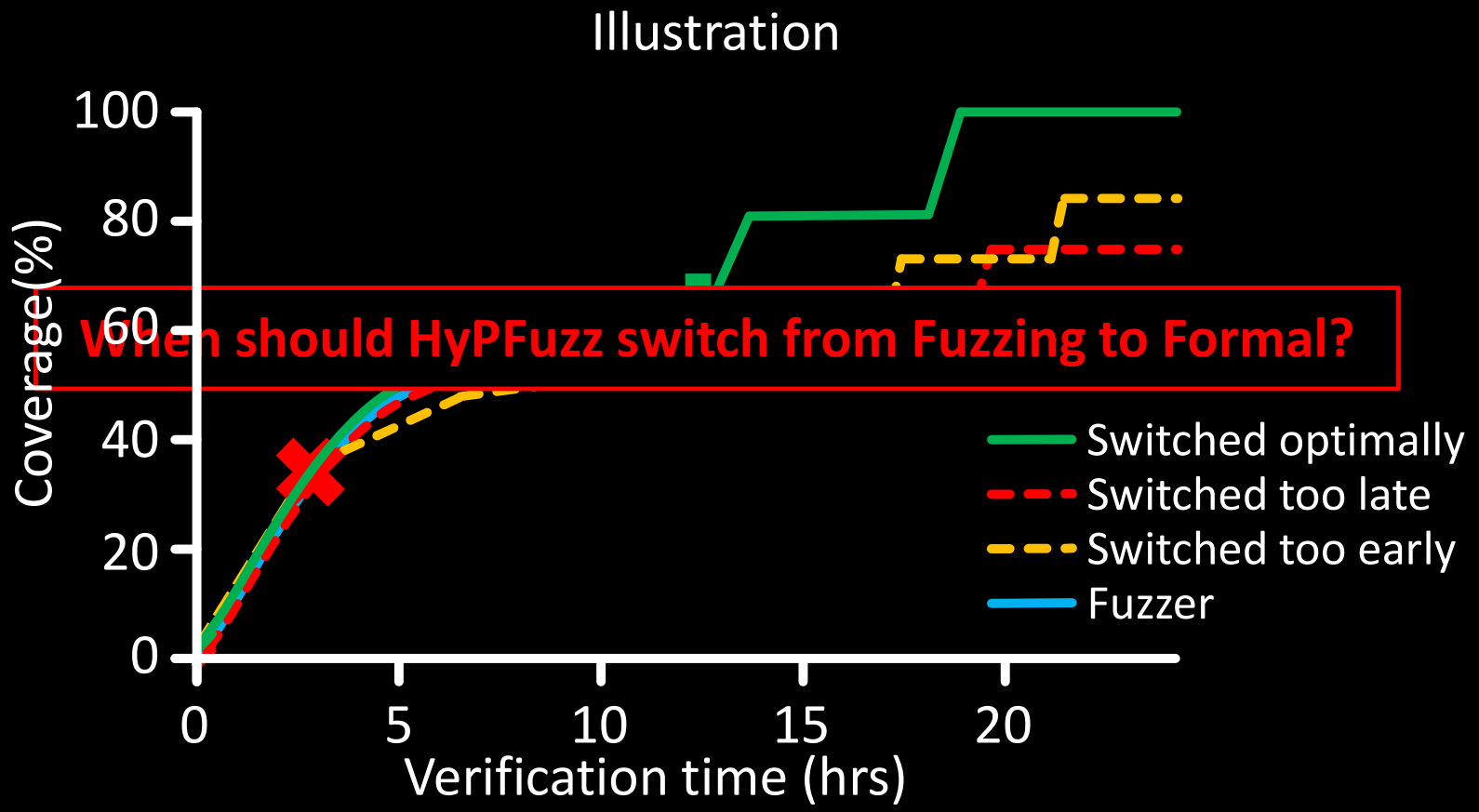
Fuzzing

Formal

HyPFuzz: Framework



Challenge: Schedule of Fuzzer and Formal



Scheduler

- Switch from fuzzer to formal tools when $r_{fuzz} < r_{fml}$
- r_{fuzz} : coverage increment rate of the recent w tests

$$r_{fuzz(w)} = \frac{total\ new\ cov.(w)}{total\ sim.time\ (w)}$$

- r_{fml} : moving average rate on hard-to-cover point set C

$$r_{fml} = \frac{num.of\ points\ in\ C}{total\ proof\ time}$$

Evaluation

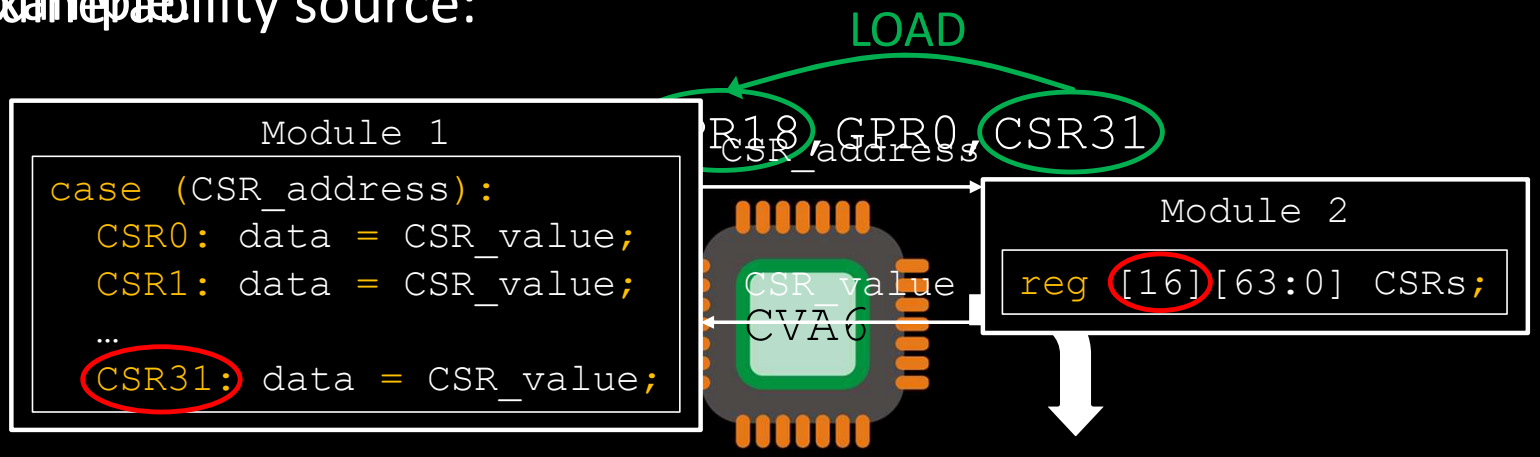
- Benchmarks: `CVA6`, `BOOM`, `Rocket Core`, `OR1200`, `mor1kx`
- Coverage metric: Branch

- Vulnerability detection:
  - Detected existing 11 vulnerabilities 3.06 × less time
  - Detected three new vulnerabilities
  - Resulted two CVEs: CVE-2022-33021, CVE-2022-33023

- Coverage achievement compared to:
  - *TheHuzz*: 41.24 ×, 6.84%
  - Random regression: 239.93 ×, 12.70%

Vulnerability Found

- CVA6 returns unknown value ('X') when accessing some control and status register (CSRs)
- Vulnerability source:

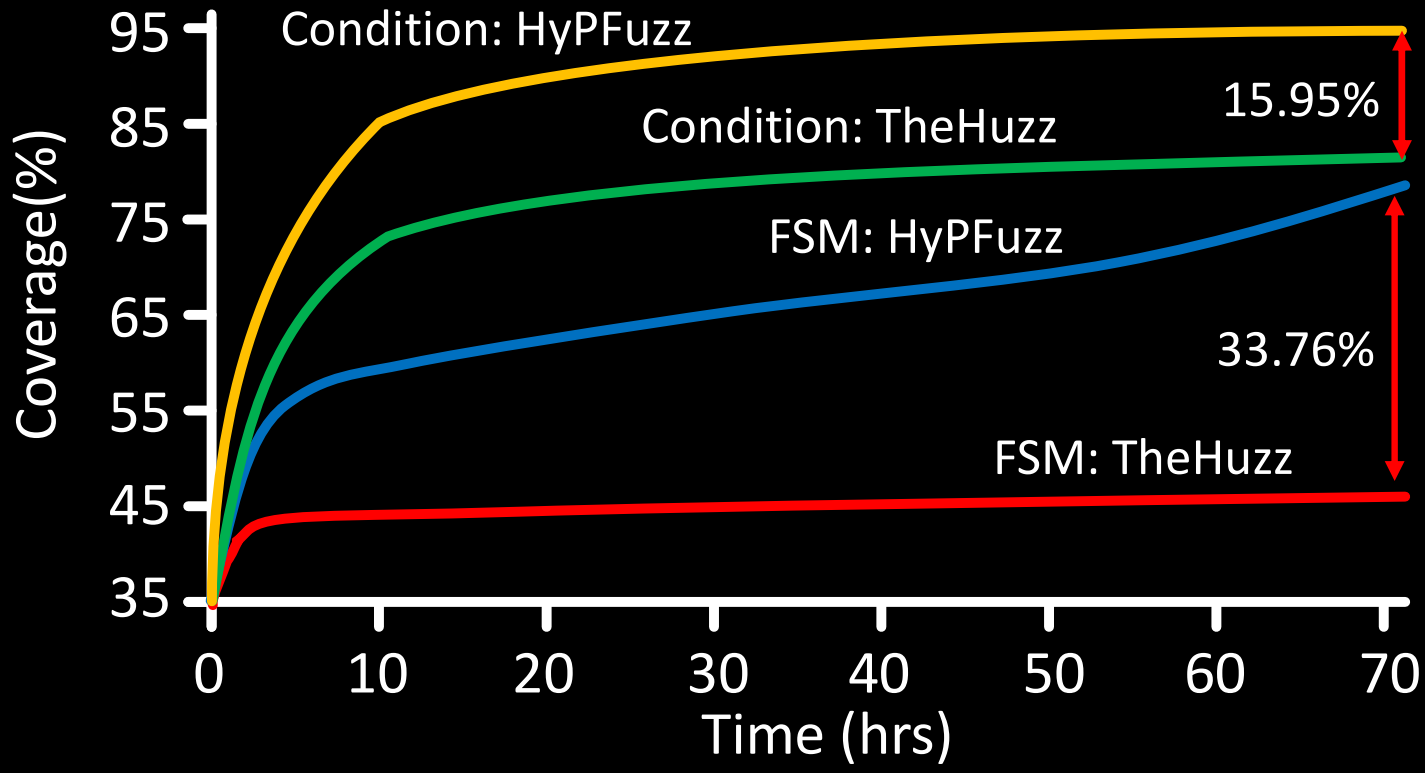


- Fuzzer alone: hard
- Formal tool alone:

```
CSRRC,GPR18,GPR0,CSR31:
GPR18=XXXXXXXXXXXXXXXX
```

Compatibility of *HyPFuzz*

- *FSM*: state transitions in a DUT
- *Condition*: all combinations of values of signals in branch statements



Conclusion

- HyPFuzz:
  - Automatically integrates fuzzer and formal tool
  - Dynamically schedules use of fuzzer and formal tool
  - Outperforms existing processor fuzzers
  - Found new vulnerabilities that are difficult for fuzzer and formal tool to find
  - Is compatible to different coverage metrics

Thank you!

chenc@tamu.edu

Lab Website: https://seth.engr.tamu.edu/

LinkedIn: https://www.linkedin.com/in/chen-chen-127235170/