Tamper Evident Microprocessors

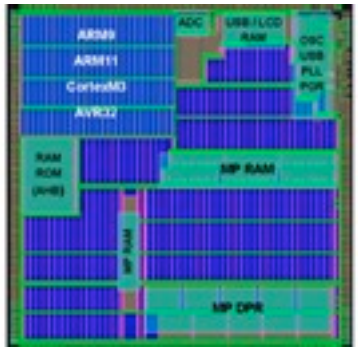
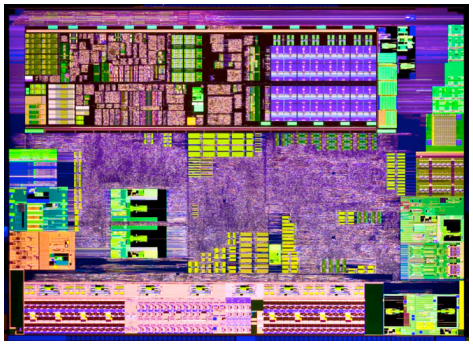
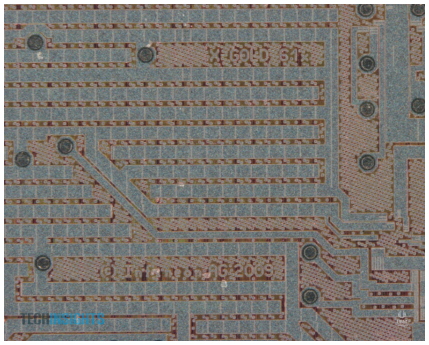
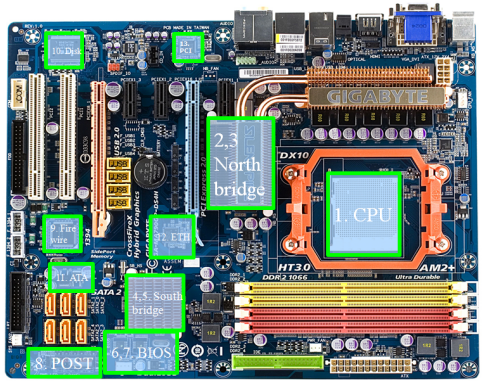
(Building hardware that you can trust)

Prof. Simha Sethumadhavan
Adam Waksman

Computer Architecture and Security Technologies Lab
http://castl.cs.columbia.edu
Department of Computer Science
Columbia University

A Quiz

- What do these hardware components do?
  - Can you guarantee that these chips don't have backdoors?

- Will you blindly trust hardware if you were buying this?
  - Military Equipment
  - Financial Sector


PROBLEM: Currently impossible to certify trustworthiness of processors & controllers.

Why is Hardware Vulnerable?

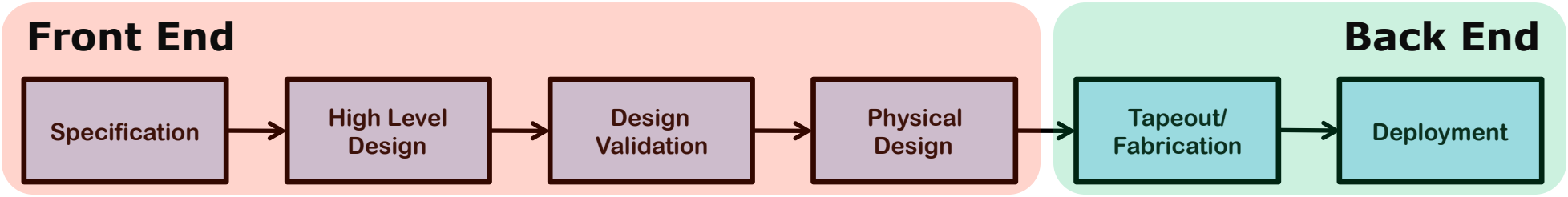
- Hardware is complex
  - OpenSPARC T2 code base larger than the Chrome code base
  - Chips unsurprisingly have unintentional bugs, e.g., Intel errata
- Hardware design resembles software design
  - Often include third-party IP components (ip-extreme.com)
  - Review of IP difficult because of intentional obfuscation
- Complexity, distribution increases risk of backdoors
  - More hands, easier to hide
  - Designs are crafted by globally distributed teams
- Creates a significant security vulnerability
  - Hardware is the root of trust; software builds on hardware
  - Attacks have been reported [The Hunt for the Kill Switch]

Concern in Military Circles

Computer networks themselves are not the only vulnerability. Software and hardware are at risk of being tampered with even before they are linked together in an operational system. Rogue code, including so-called logic bombs, which cause sudden malfunctions, can be inserted into software as it is being developed. As for hardware, remotely operated "kill switches" and hidden "backdoors" can be written into the computer chips used by the military, allowing outside actors to manipulate the systems from afar. The risk of compromise in the manufacturing process is very real and is perhaps the least understood cyberthreat. Tampering is almost impossible to detect and even harder to eradicate. Already, counterfeit hardware has been detected in systems that the Defense Department has procured. The Pentagon's Trusted Foundries Progam, which certifies parts produced by microelectronics manufacturers, is a good start, but it is not a comprehensive solution to the risks to the department's technological base. Microsoft and other computer technology companies have developed sophisticated risk-mitigation strategies to detect malicious code and deter its insertion into their global supply chains; the U.S. government needs to undertake a similar effort for critical civilian and military applications.

Prior Work and Scope

- ASIC hardware design stages



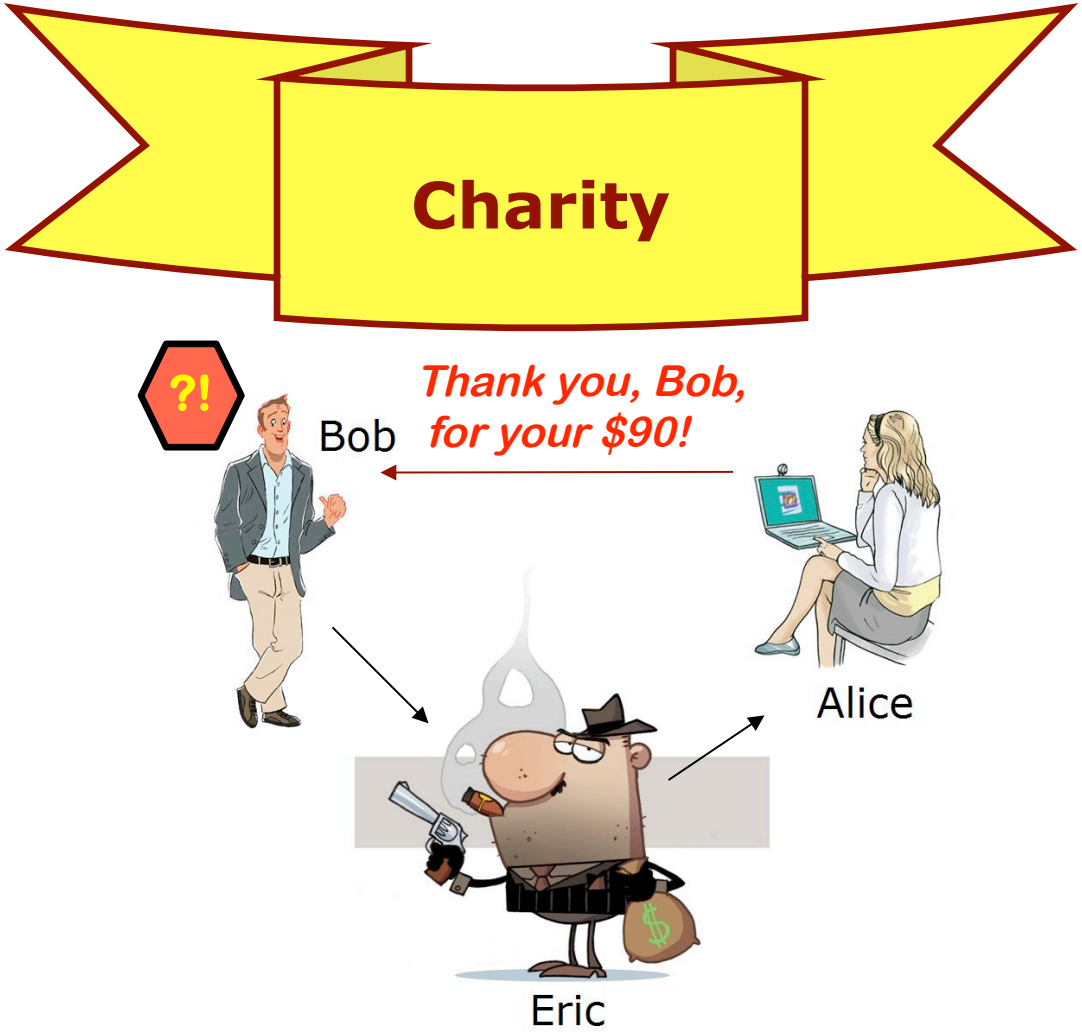
- Prior work focuses on back end
  - More immediate threat, fabrication is outsourced
  - Example: IC fingerprinting [Agrawal et al., 2007]

- However, front end is the extreme root of trust
  - Common assumption: golden model from front end
  - How do you ensure that front-end doesn't contain backdoors?

- Our work: Make golden netlist a reality
  - Integrate into the front-end a continuous monitoring system

Solution: An Analogy

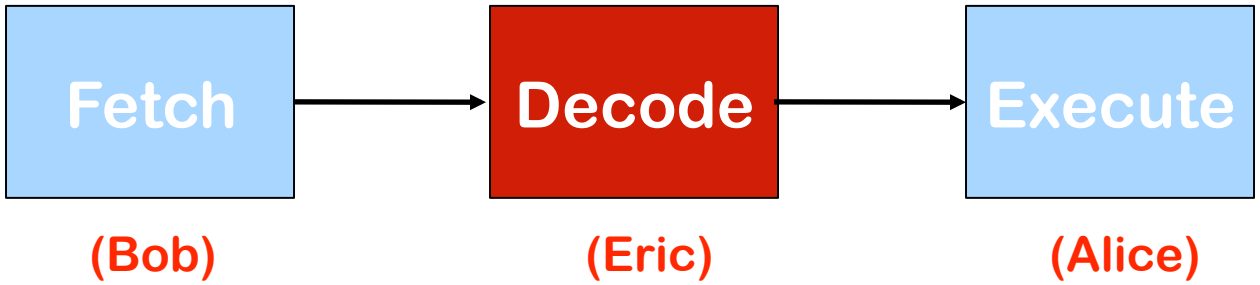
- Bob
  - Generous guy
  - Donates $100
- Eric
  - Evil accountant
  - Steals $10
- Alice
  - Charity president
  - Receives $90



- What is the solution to this problem?

Applying Idea to Microprocessors

- Problem: Units do different work trust each other
  - One bad unit breaks the whole system



- Solution: Have units watch each other
  - Build continuous invariant checking in hardware

- *KEY IDEA:* EXPLOIT DIVISION OF WORK
  - Universally available in microprocessor designs
  - Execution requires a series of tightly coupled microarch events

- Solution works because all units cannot be malicious
  - Turned a threat into a solution!

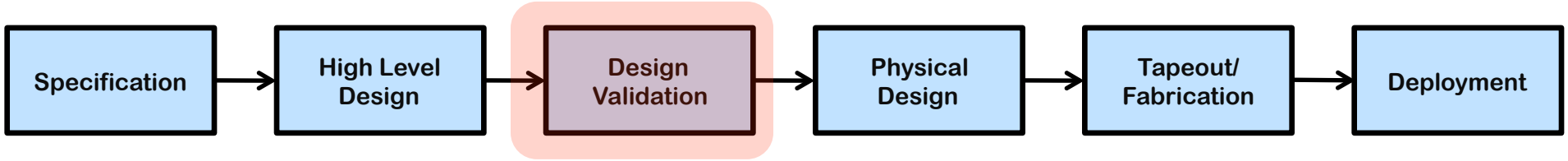
Outline

- Taxonomy
  - Ticking Timebombs, Cheat Codes, Emitters, Corrupters
- Solutions
  - Common solutions are unsatisfactory
  - TrustNet and DataWatch
  - Smart Duplication
- Results
  - Correctness, Coverage and Costs
- Future Work, Broader Vision

Taxonomy of Attacks

- **Backdoor = Trigger + Payload**
  - Trigger: Mechanism for initiating an attack
  - Payload: Malicious, illegal action

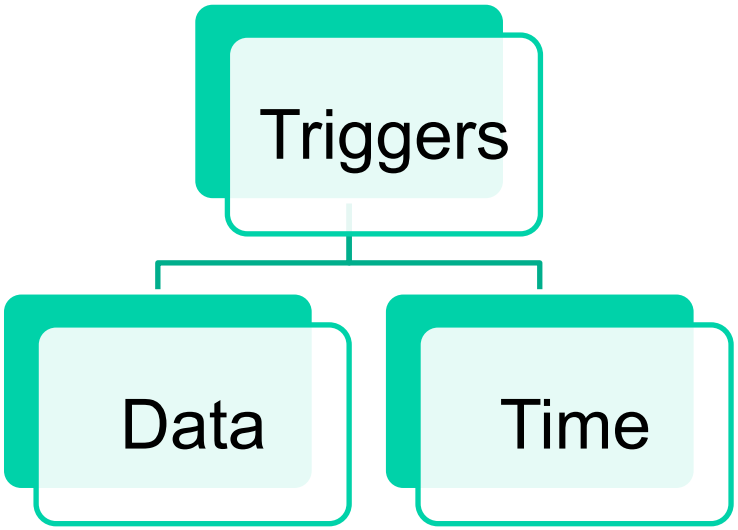
- **Why do we need a Trigger?**



  - Trigger-less designs will be caught during validation

  - Most designs deploy intensive transactional testing
    - Small units are validated thoroughly, followed by aggregations
    - Typically smaller units are validated for 10^6 - 10^8 cycles
    - Larger units validated for fewer cycles

Taxonomy of Attacks: Triggers

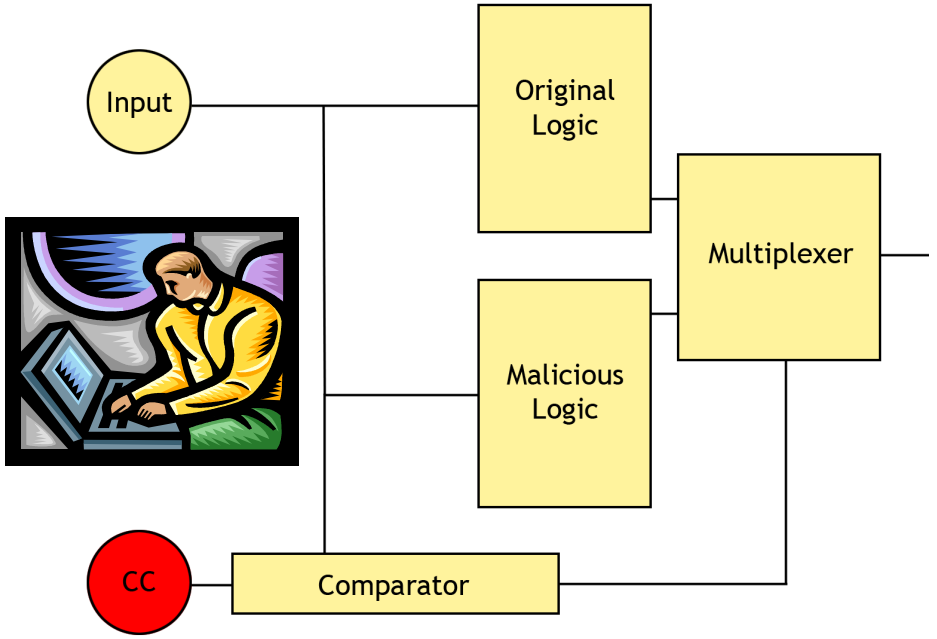


- How many ways can a backdoor be triggered?
- Triggers are finite state machines
  - Can change state only when time or input data changes
- A complete taxonomy of hardware backdoors

Taxonomy of Attacks: Triggers

- **Data Triggers**
  - Cheat codes (CC)
  - Triggered by special instructions or data
- **Pros/Cons of CC's**
  - Easy to bypass validation
    - 1 in 2^{64} chance!
  - However, hacker needs access to the machine

Cheat Code Trigger

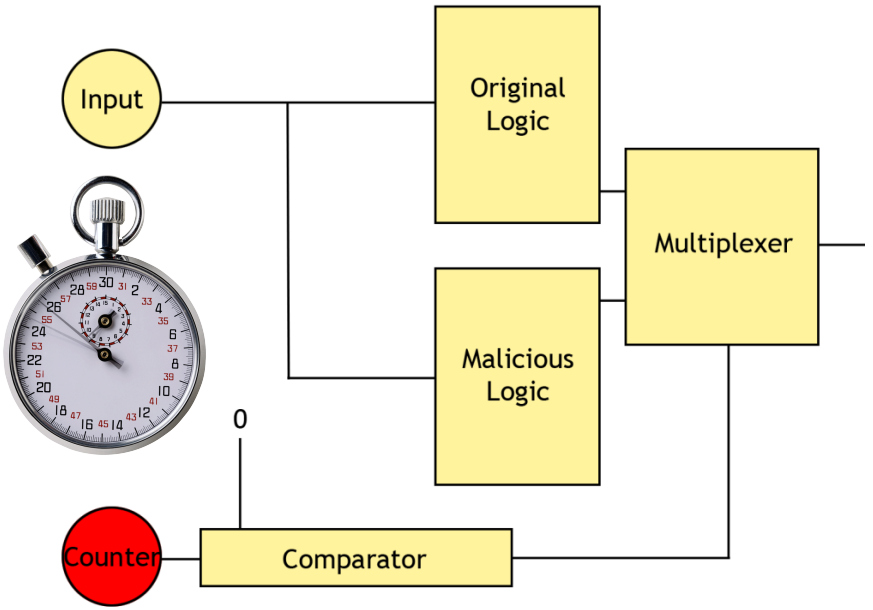


Taxonomy of Attacks: Triggers

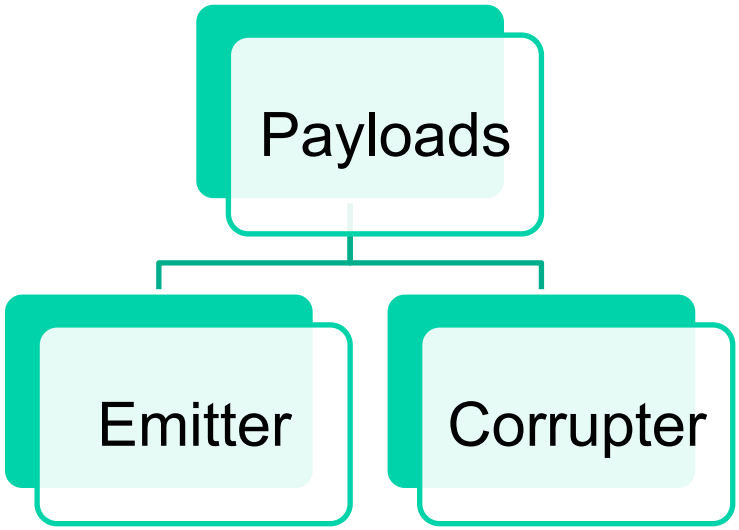
- **Time Triggers**
  - Ticking Timebomb (TT)
  - Triggered over time

- **Pros/Cons of TTs**
  - Easy to bypass validation
    - 48-bit counter takes ~ 20 minutes @ 1 GHz
  - Easy to hide
  - However, open to everyone

Ticking Timebomb Trigger



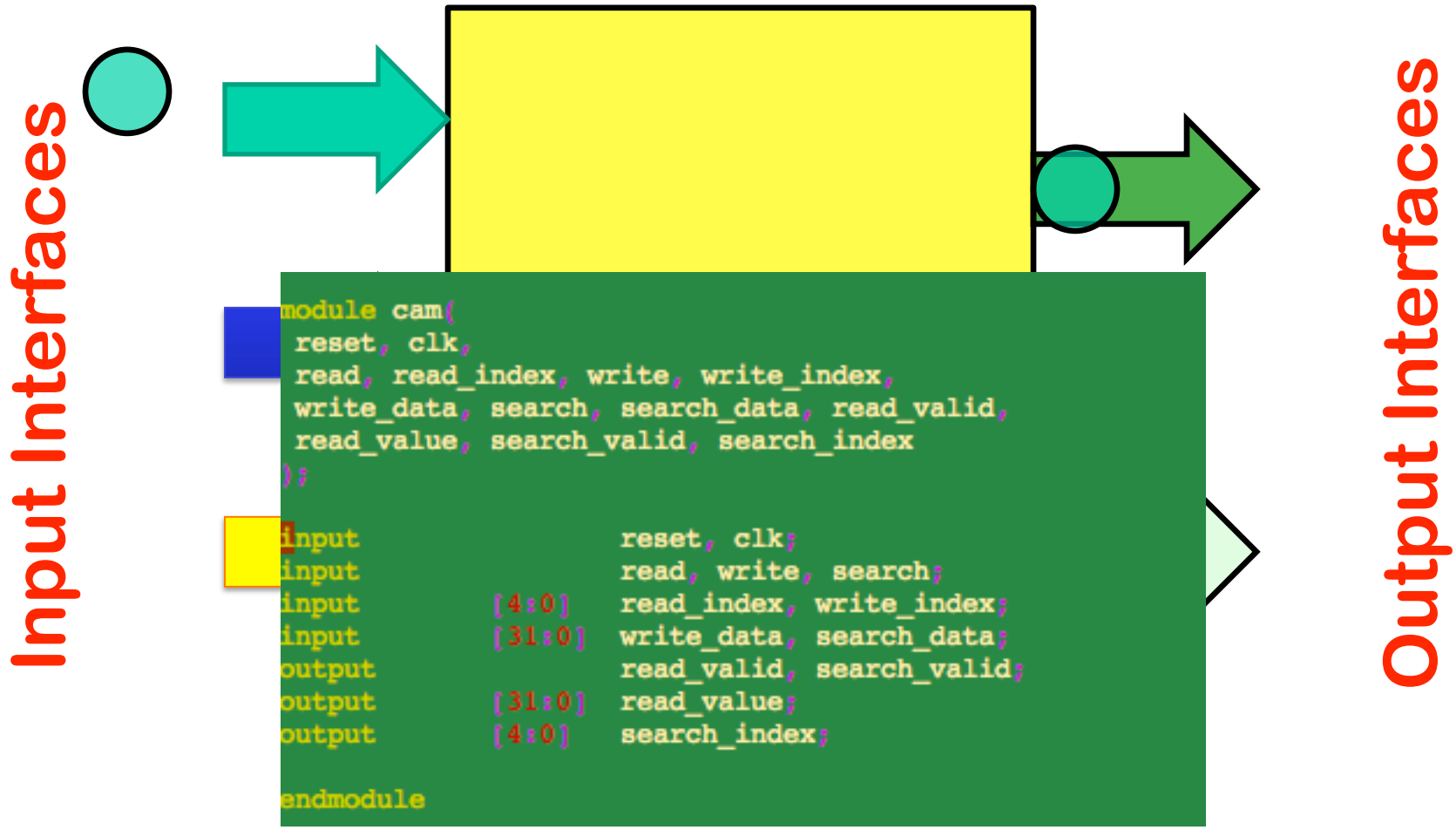
Taxonomy of Attacks: Payloads



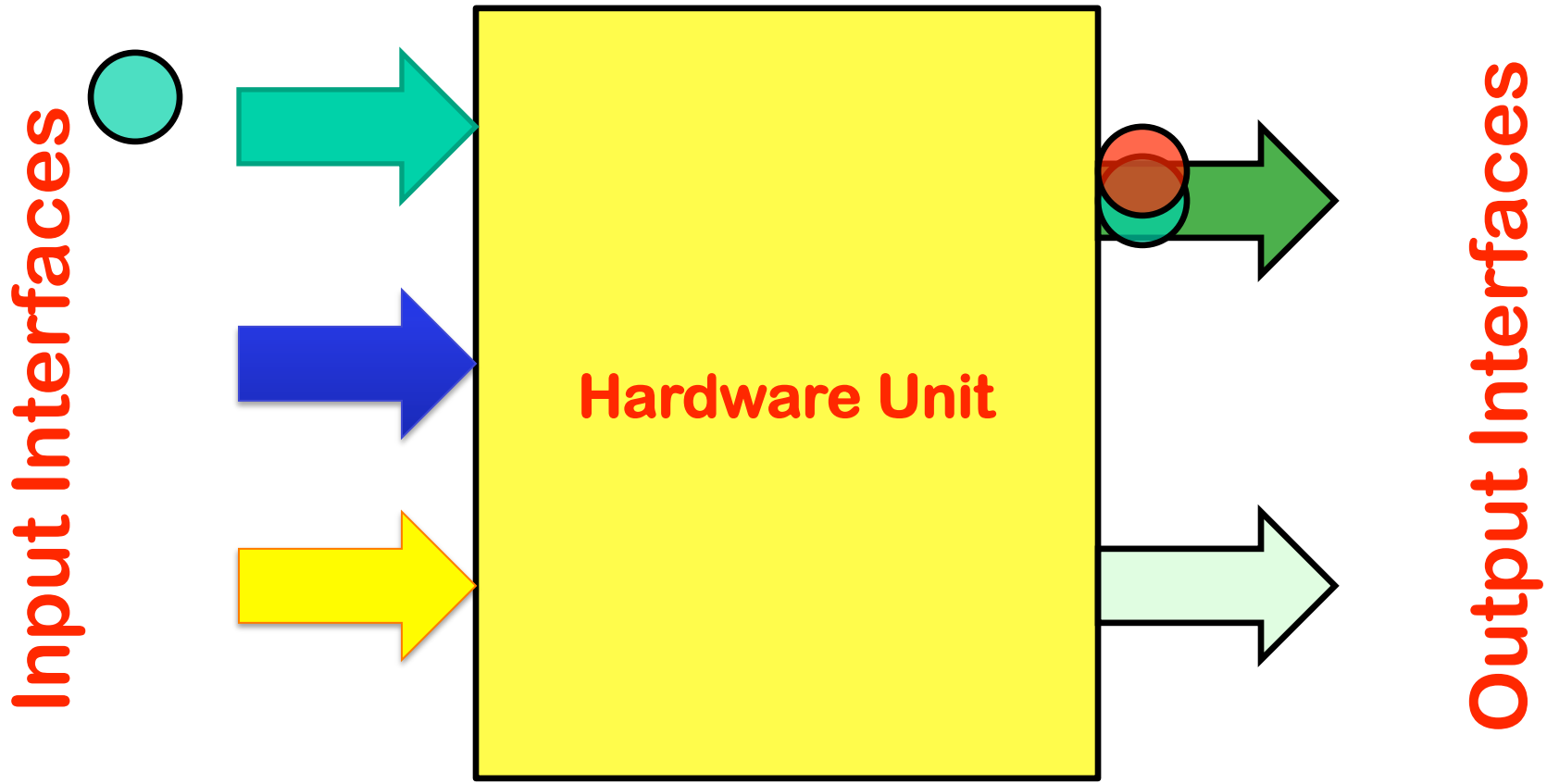
Is the payload separate from the normal instructions?

- **Emitter Attacks**
  - Extra malicious events
  - Separate from normal events
  - Unlikely to be noticed by user
- **Corrupter Attacks**
  - No extra malicious events
  - Normal operations altered
  - Difficult to engineer

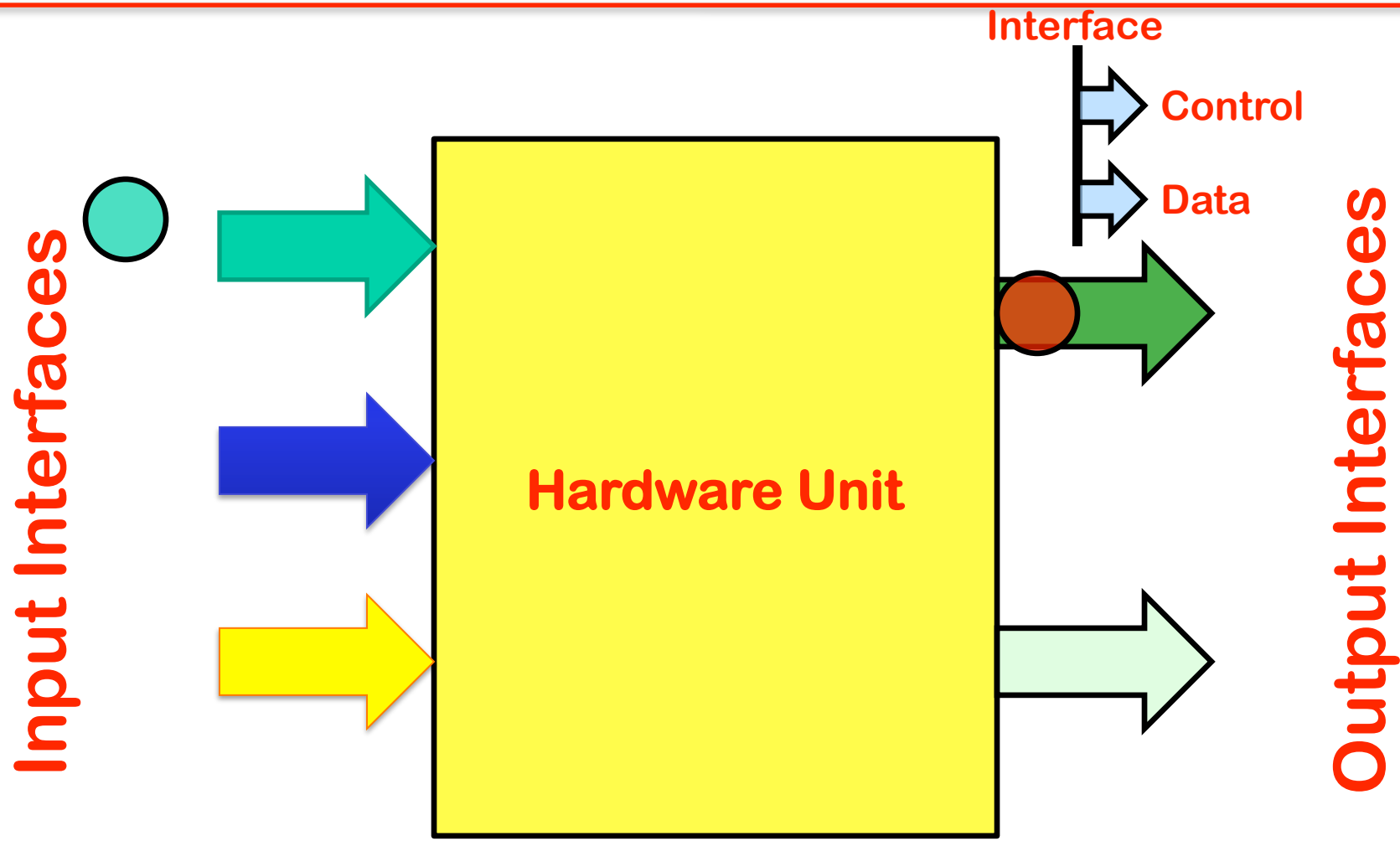
Visualizing Attacks at the Unit Level



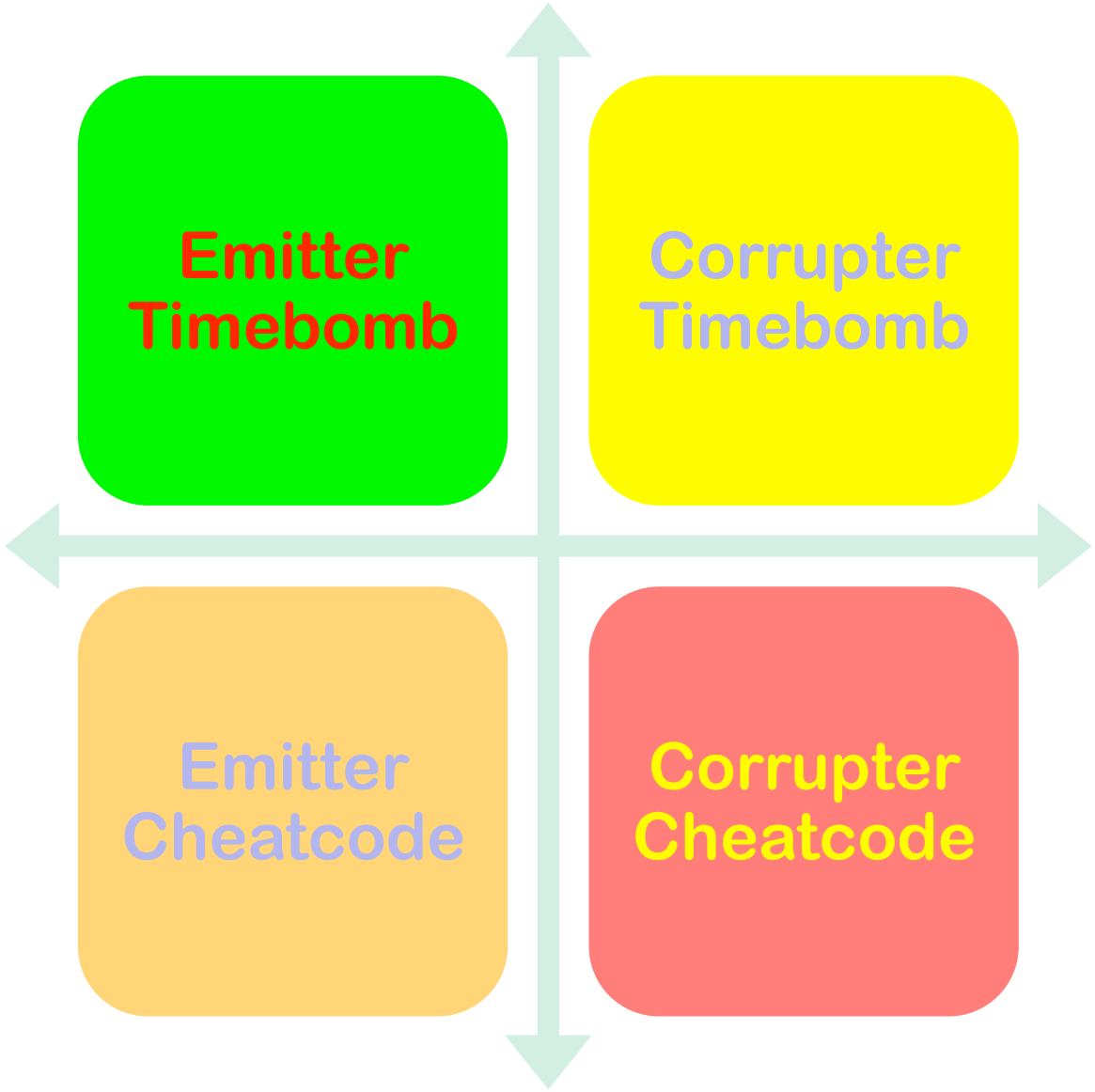
Emitter Attacks



Corrupter Attacks



Taxonomy of Attacks: Summary



Outline

- Taxonomy
  - Ticking Timebombs, Cheat Codes, Emitters, Corrupters
- Solutions
  - Common solutions are unsatisfactory
  - TrustNet and DataWatch
  - Smart Duplication
- Results
  - Correctness, Coverage and Costs
- Future Work, Broader Vision

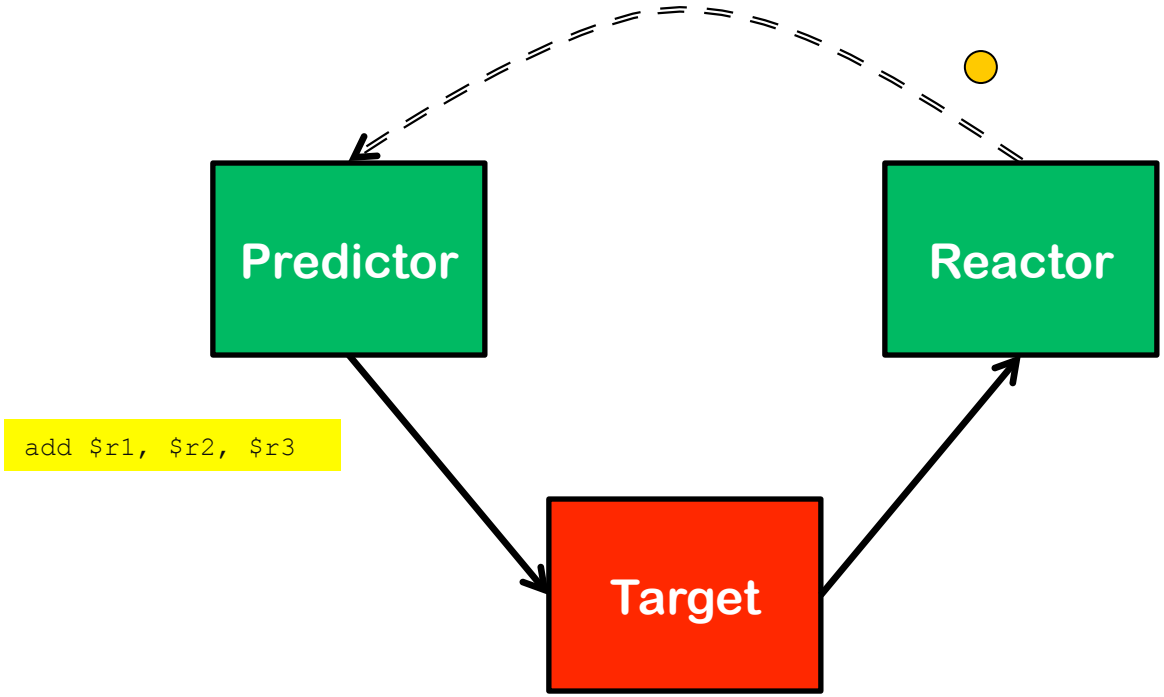
Problem Constraints Favor Attackers

- Large design team
  - Each designer works on one unit or part of one
  - Security add-ons can be done by one member
- Full knowledge
  - Attacker has complete access to all design specifications
  - Attacker also knows about additional security mechanism
- Equal distrust
  - Any one designer/unit may be evil
  - Security add-ons may contain backdoors

Common Solutions are Unsatisfactory

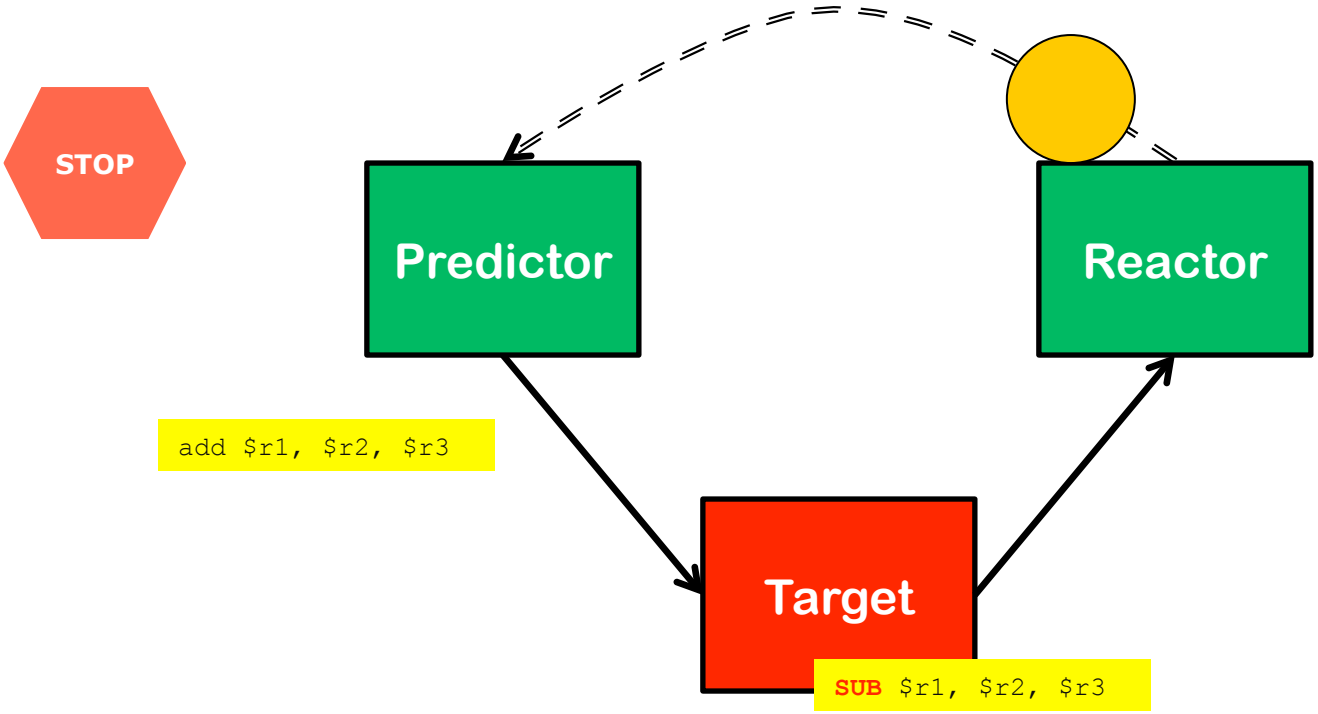
- Careful audits
  - Audits not completely effective at catching unintentional bugs
  - Can audits catch intentional, hidden backdoors?
- Random validation
  - Catching a 48-bit TT requires 281.4 trillion cycles of validation!
  - The chance of catching a 48-bit CC is $3.5 * 10^{-15}$
- Static verification
  - Attacker has complete access to all design specifications
  - Attacker can work around theorems or proofs
- Cannot fix problem in run time software
  - All software runs on hardware
  - Software fix will likely use malicious hardware

TrustNet Architecture



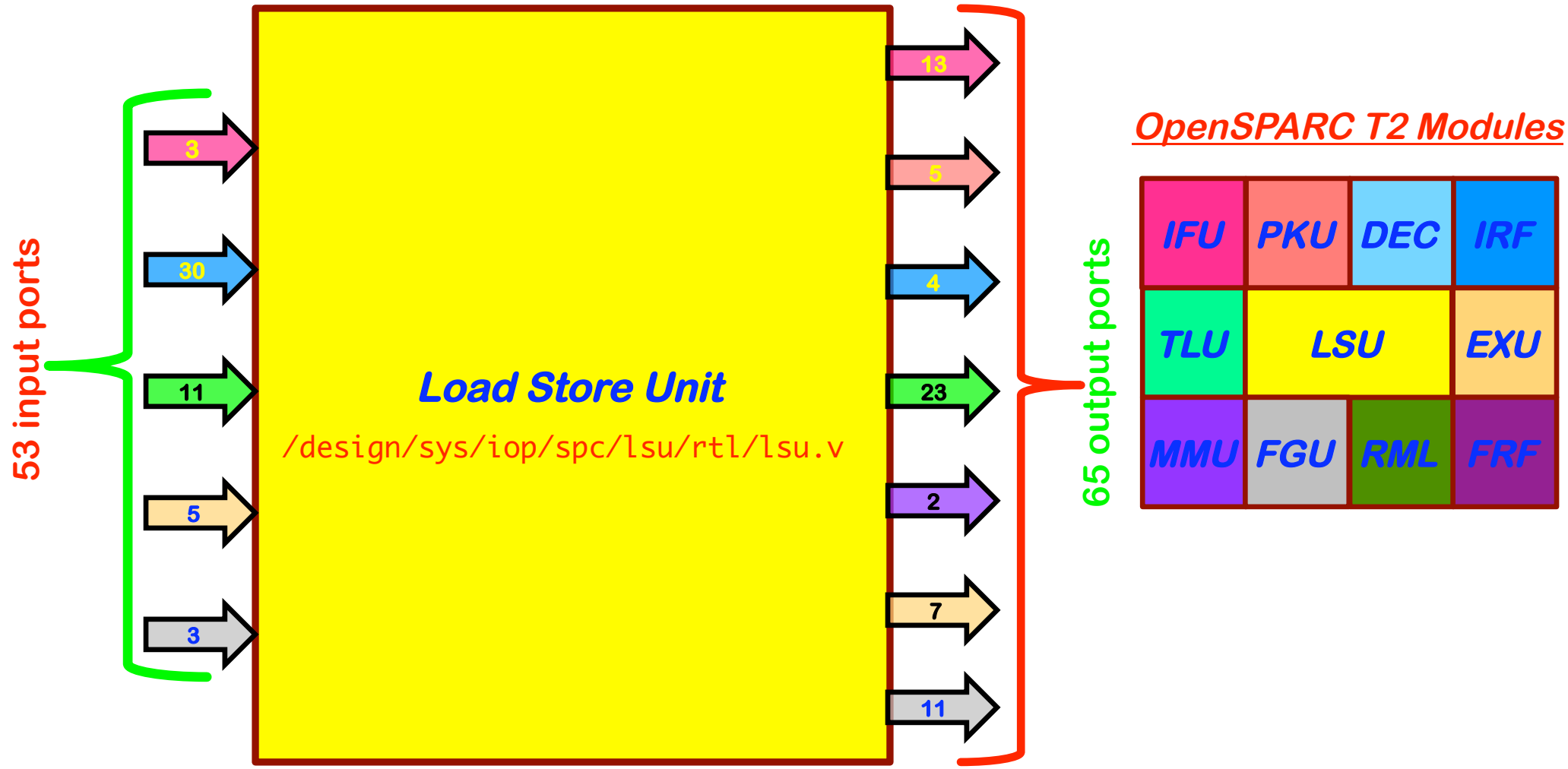
- Predictor and Reactor monitor the Target
  - Each cycle reactor announces events to predictor (little logic)
  - Disagreement results in alarms
- Guarantees
  - Division of work prevents *one* bad guy from breaking two units
  - Simple checker allows formal verification

DataWatch Architecture



- Scaled up version of TrustNet
  - Multiple bit messages
  - Confirms types of messages (instead of just yes/no)

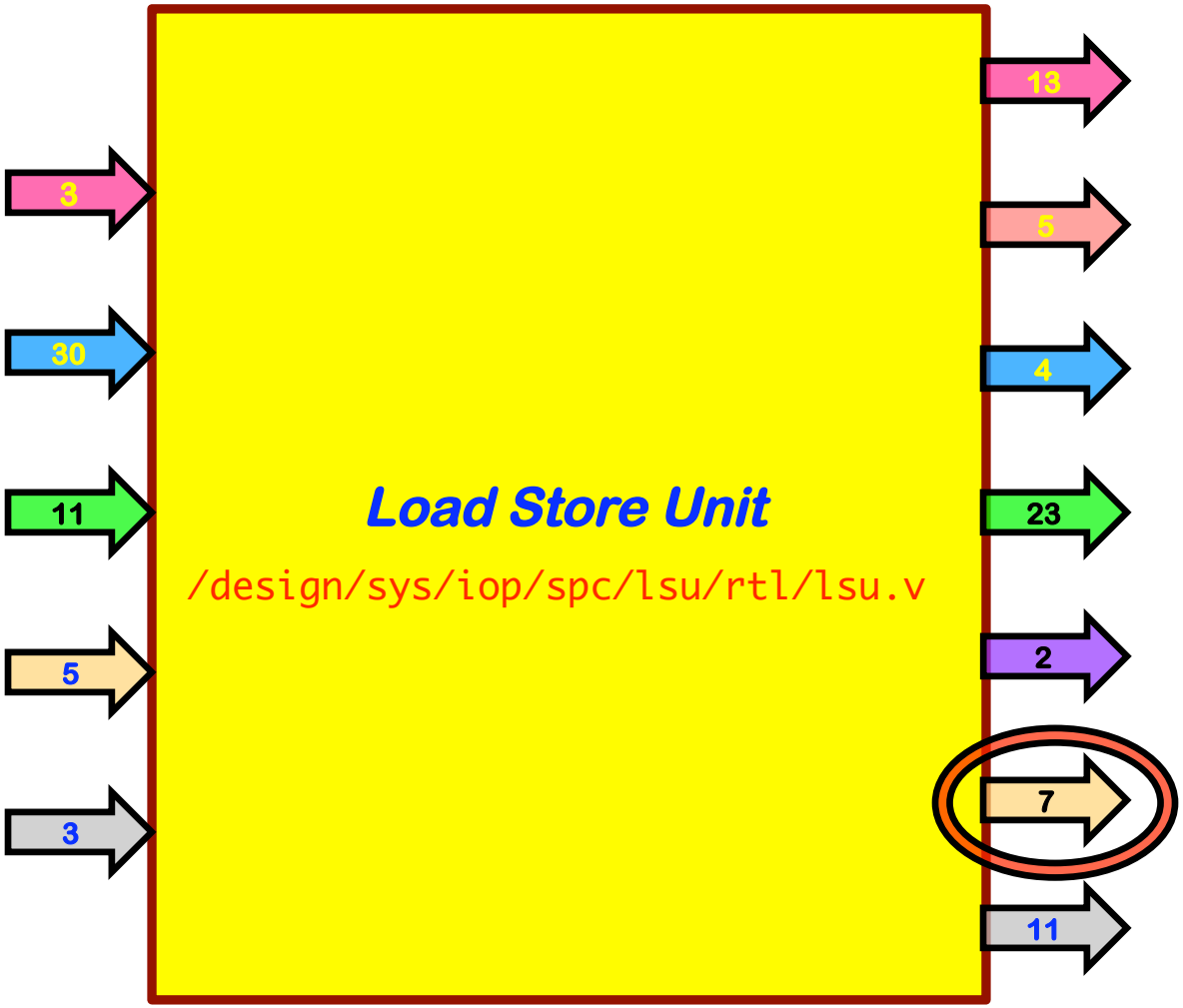
OpenSPARC T2 LSU Example

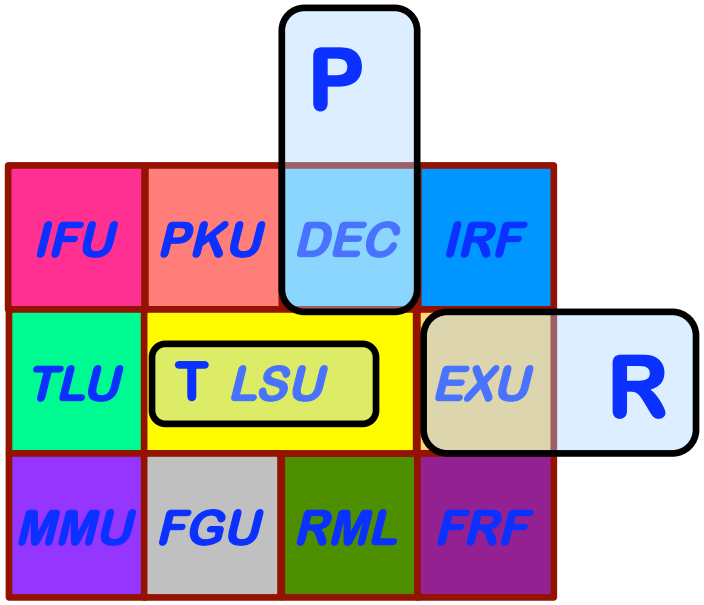


OpenSPARC T2 LSU Example

Source code:

```
1347 output [4:0] lsu_exu_rd_m; // Addr of dest register
1348 output [2:0] lsu_exu_tid_m;  // Thread ID ld return
```

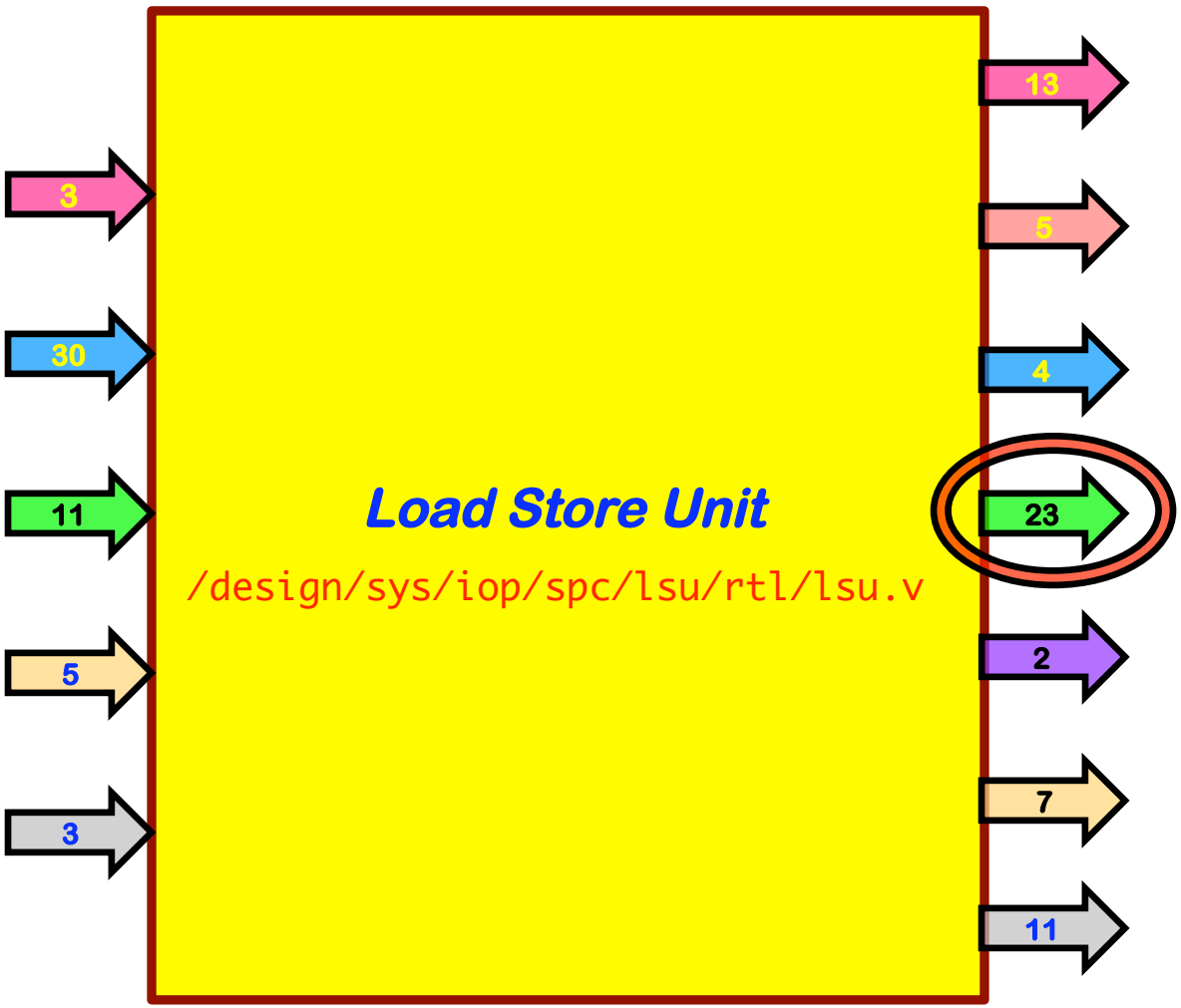


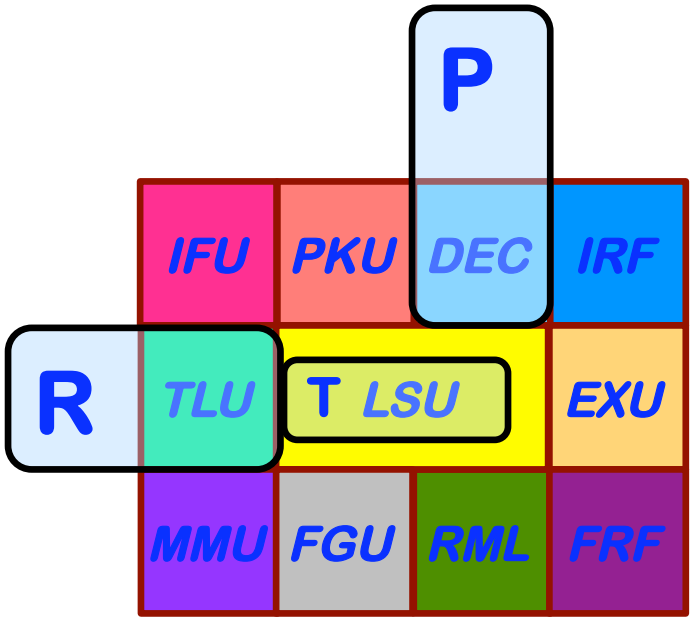


OpenSPARC T2 Modules

OpenSPARC T2 LSU Example

Source code: `1254  output lsu_tlb_bypass_b; // TLB in bypass mode`

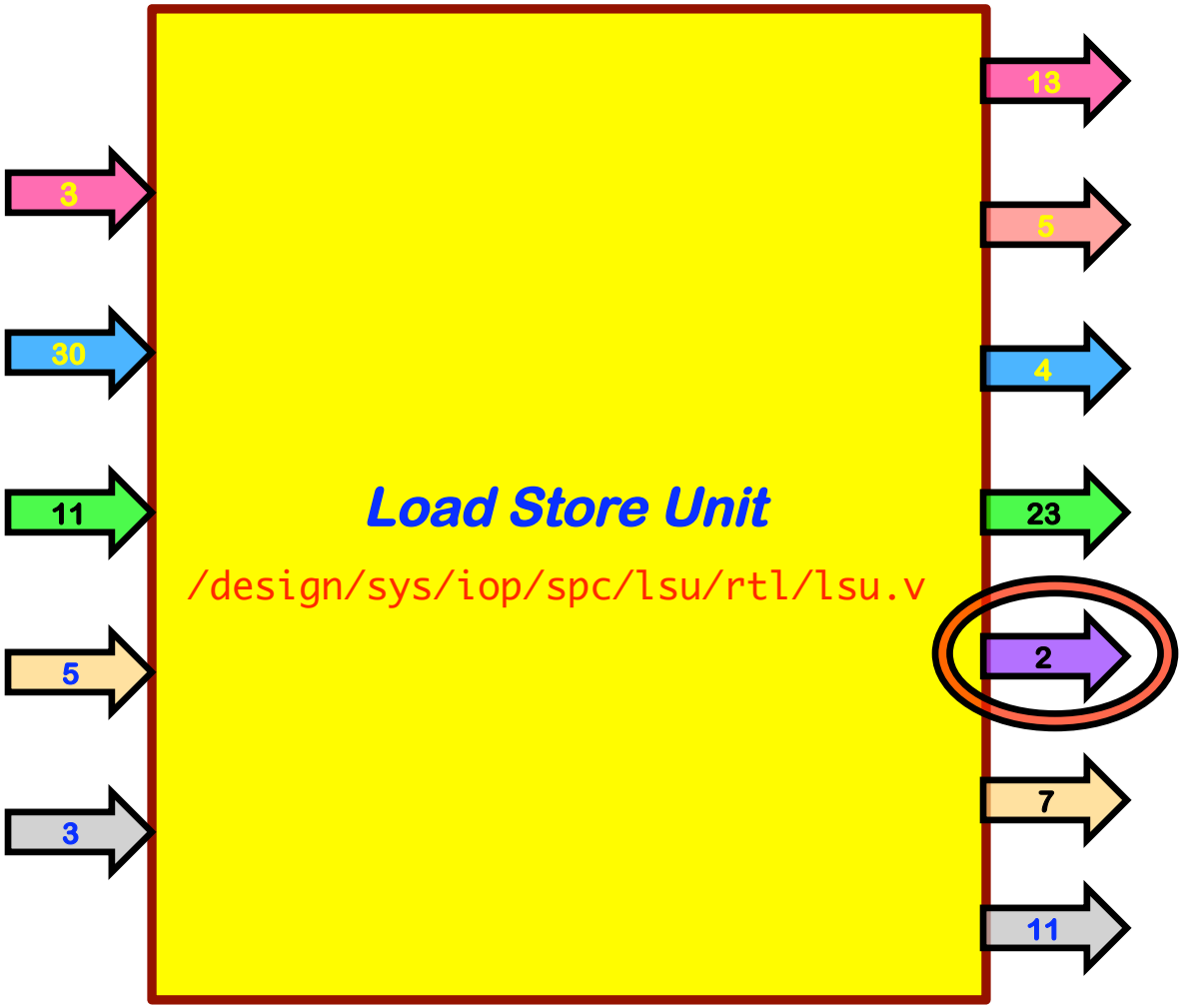


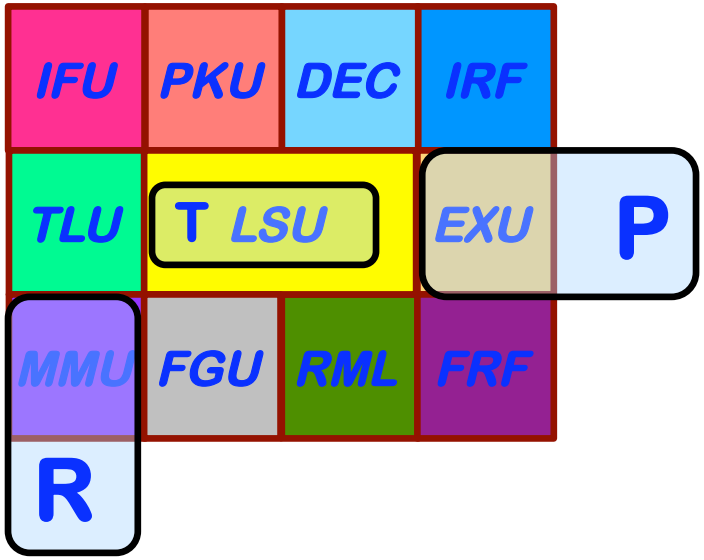


OpenSPARC T2 Modules

OpenSPARC T2 LSU Example

Source code: `1332 output [47:0] lsu_mmu_va_b;`

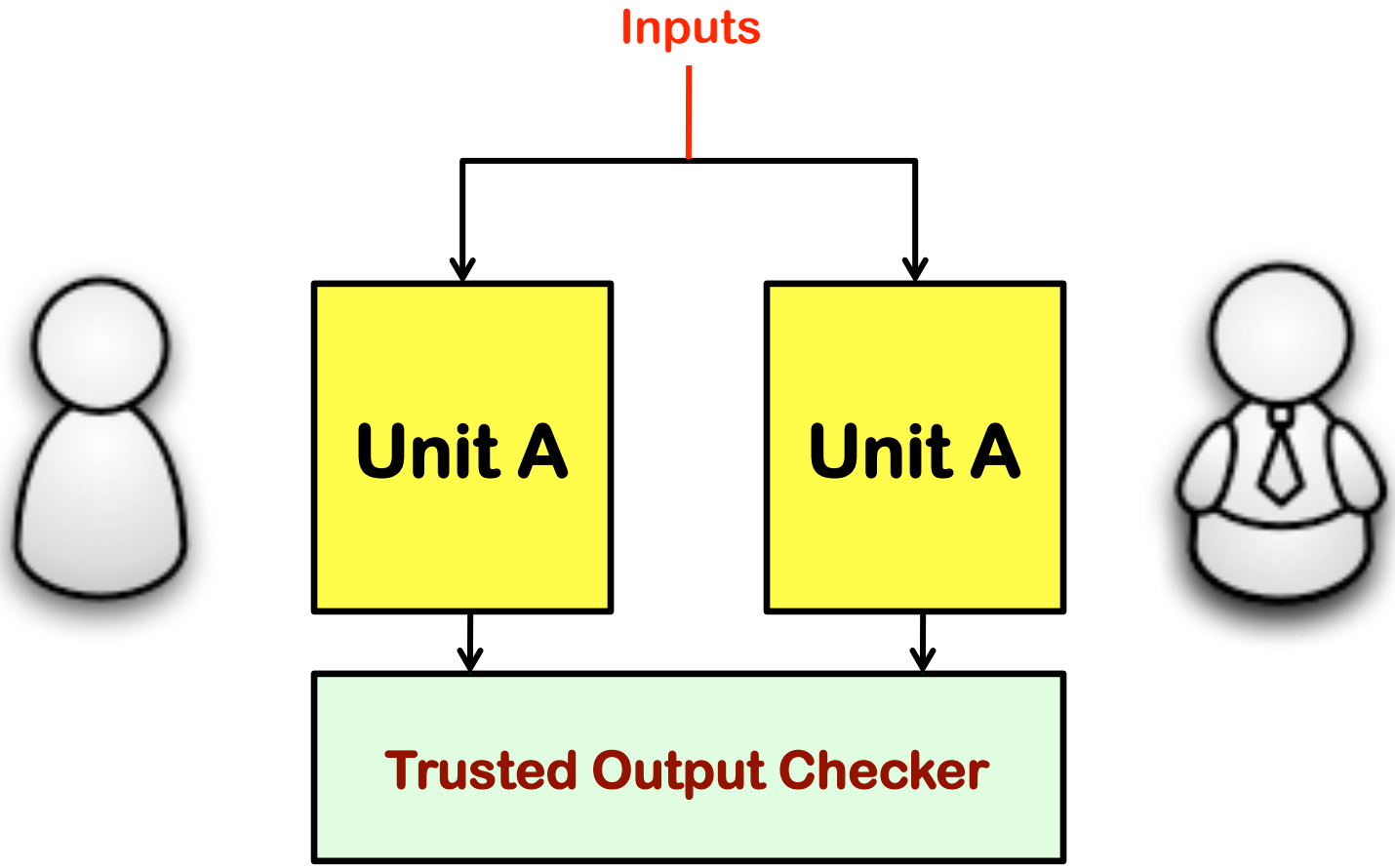




OpenSPARC T2 Modules

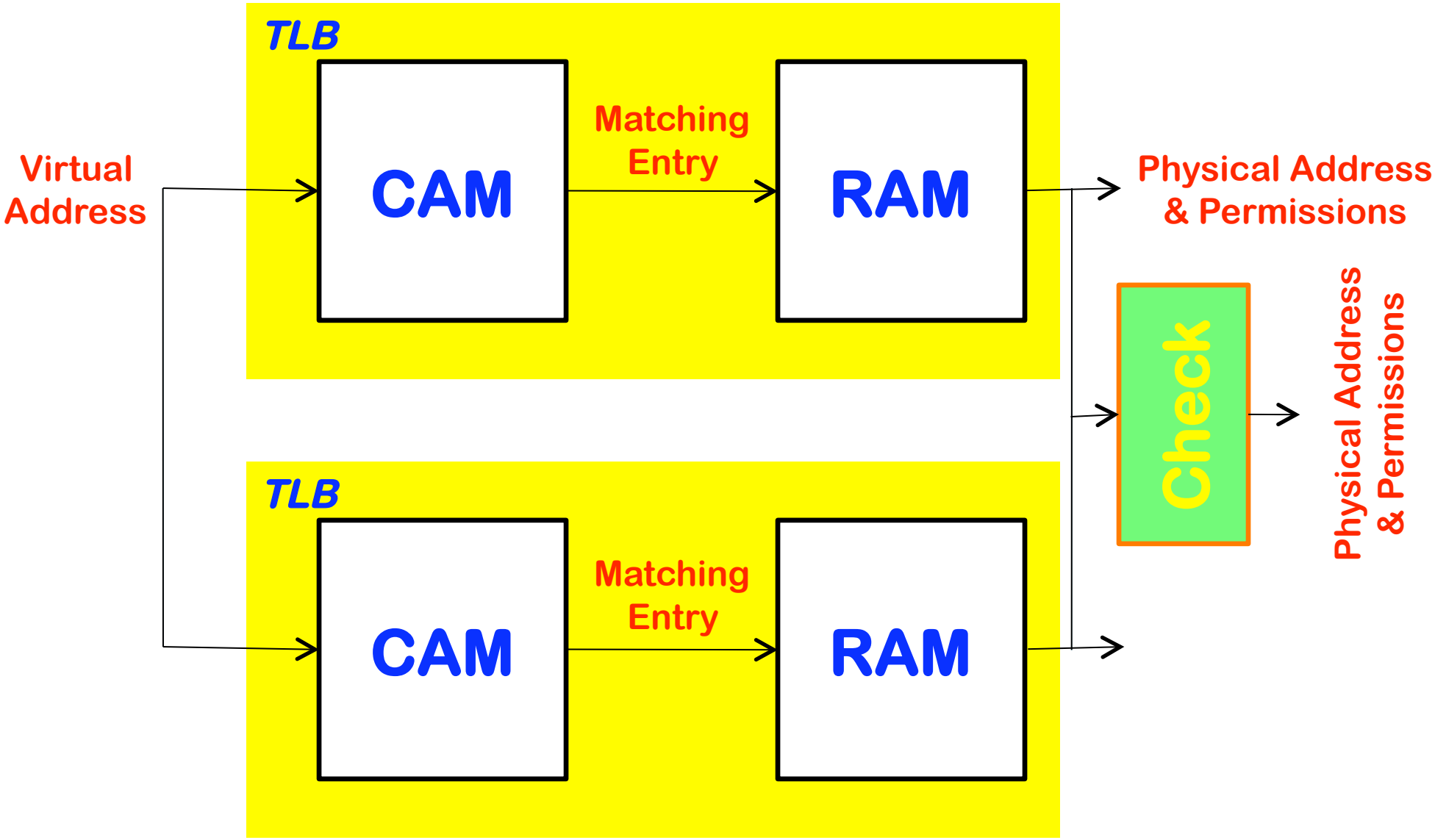
Hash addresses and compare

When all else fails: Diversity

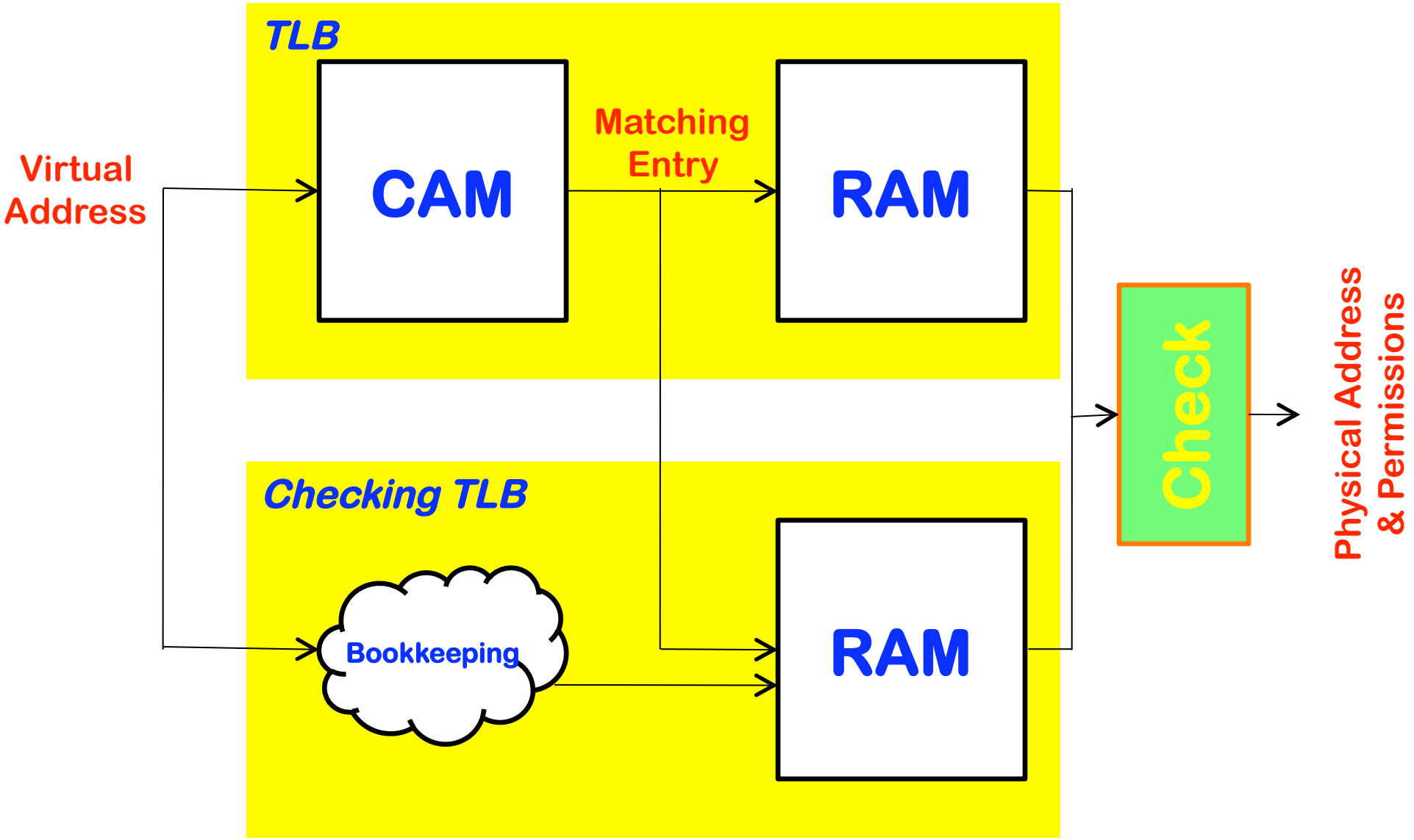


- However, diversity is prohibitively expensive
  - Non-recurring design, verification costs due to duplication
  - Recurring power and energy costs

TLB Full Duplication



Partial Duplication Example: TLB



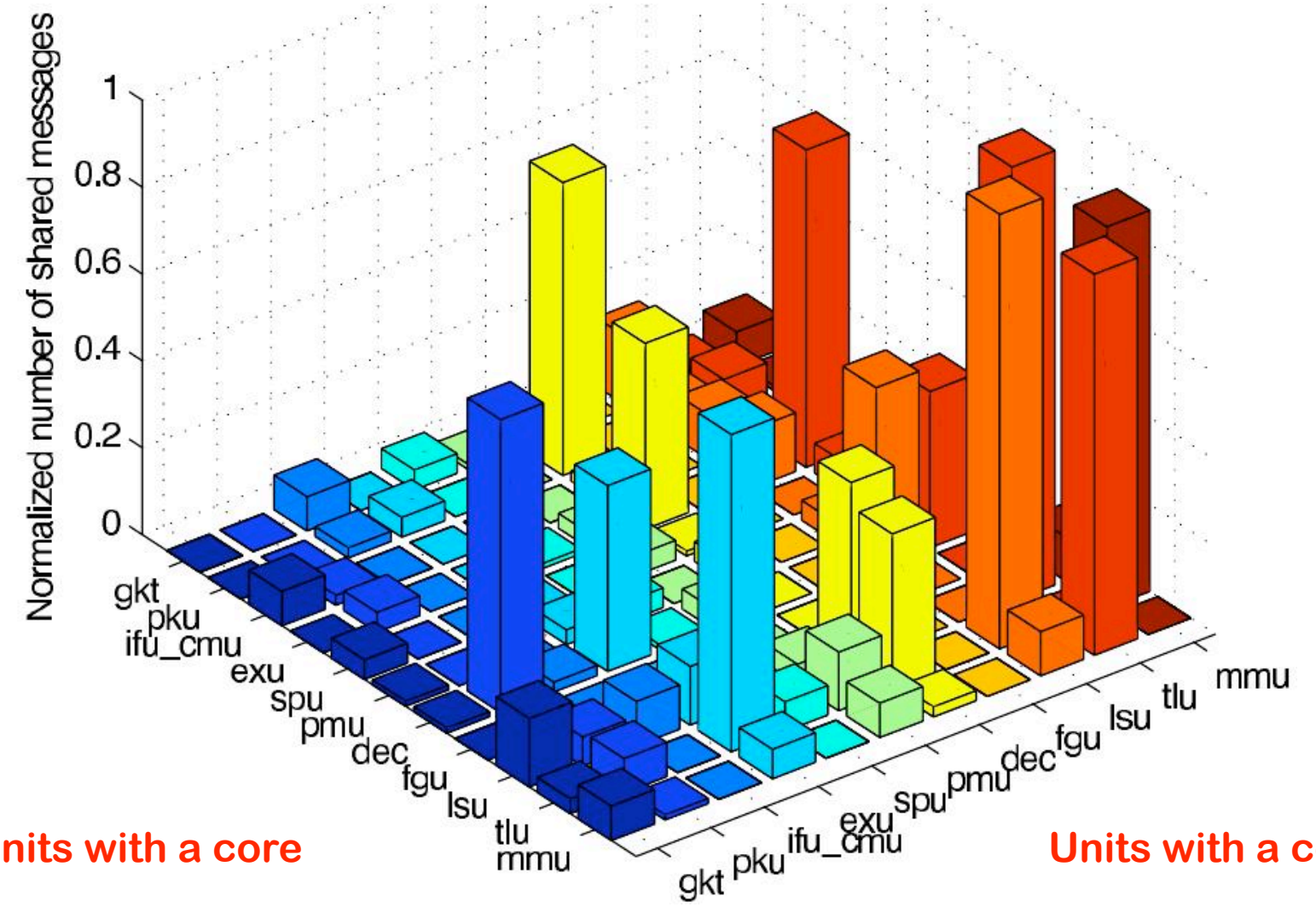
Outline

- Taxonomy
  - Ticking Timebombs, Cheat Codes, Emitters, Corrupters
- Solutions
  - Common solutions are unsatisfactory
  - TrustNet and DataWatch
  - Smart Duplication
- Results
  - Correctness, Coverage and Costs
- Future Work, Broader Vision

Experimental Context, Correctness, Costs

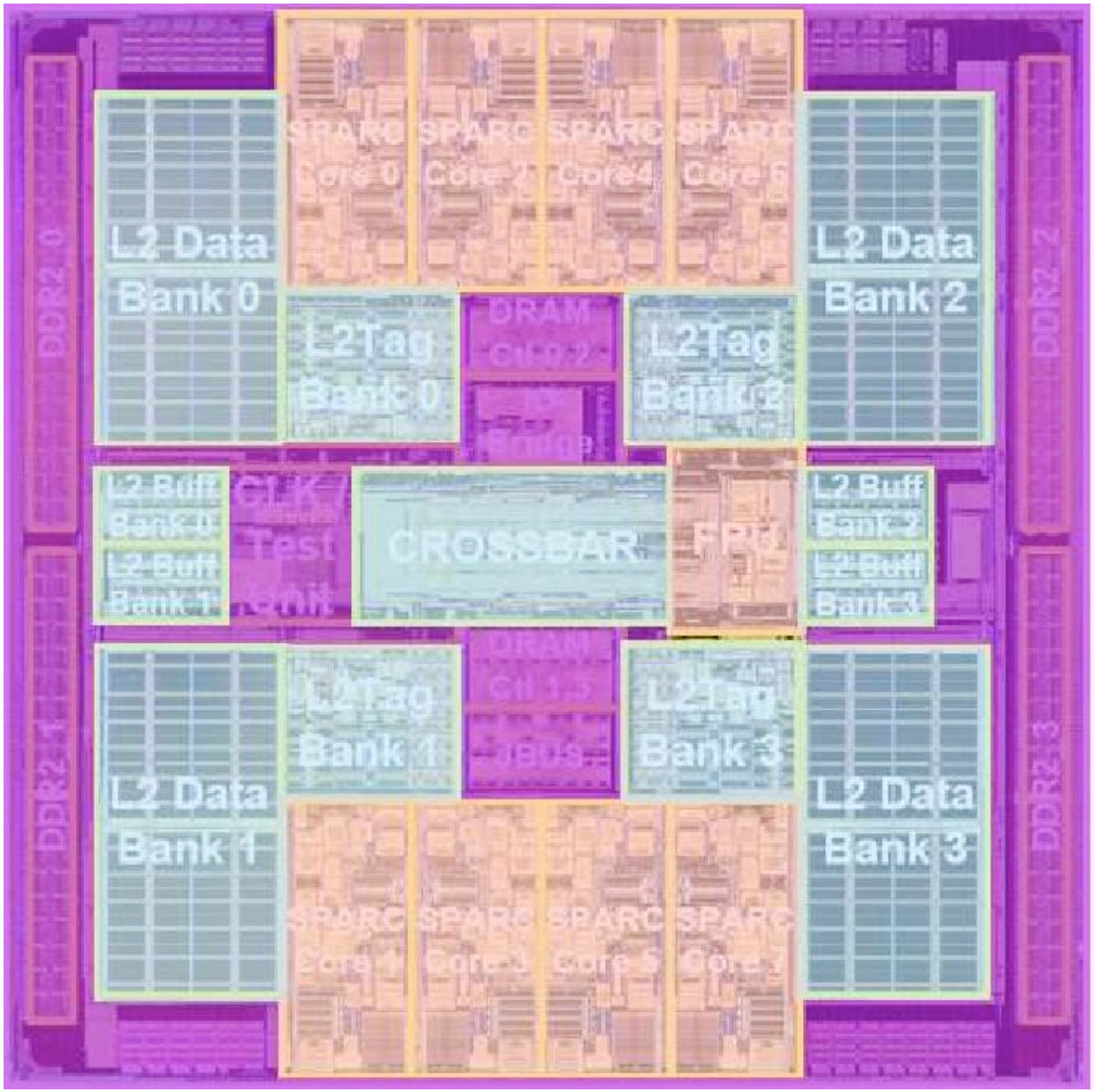
- Context
  - Simplified OpenSPARC T2
- Correctness
  - Designed attacks
  - No false positives or negatives
- Costs
  - Low area overhead (2 KB per core)
  - No performance impact
- How to measure coverage?

Coverage: Vulnerability Space



Paper has plots for other units at a chip level

Coverage Visualization

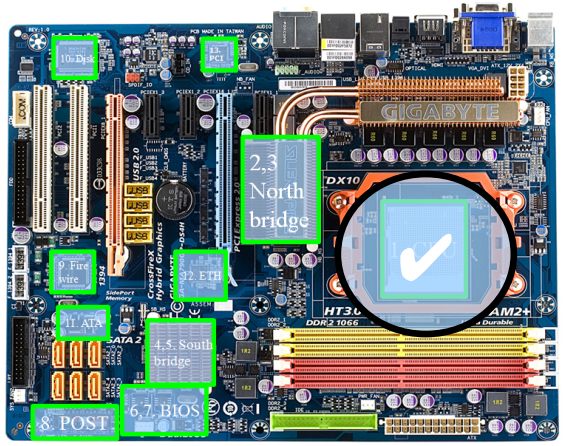


Partition of OpenSPARC T2 Die Photo

WARNING:
This is an approximate representation

Summary

- Strengthen root of trust: a certifiable microprocessor
  - Hardware-only solution. No perf impact, low area overhead
  - Provided attack taxonomy
  - Method to measure the attack space
- Applicability of TrustNet & DataWatch
  - Covered: pipelines, caches and content associative memory
  - Not covered: ALU, microcode, power mgmt., side-channels
- Moving forward
  - Expand coverage
    - Out-of-order processors
    - Motherboard components
  - Design automation tools
  - Programmable monitoring
  - Complexity-effective techniques



- Steps toward a secure chain of trust w/ untrusted units

Broader Vision: Re-examine Security

- Current Approach to Security is REACTIVE -- BAD
  - Patch flaws reactively
  - How to be proactive about security (a very difficult problem)

- What if we took a ground up approach?
  - Make hardware secure
  - Build hardware primitives to support software security
  - Build software security countermeasures using HW primitives
  - Build software securely with security as first order constraint

- Discovering the primitives: SPARCHS project
  - Inspired by how humans protect from biological threats

- Securing hardware is the first step

Thank you and Questions!