Safety Assurance of Cyber-Physical Systems Through Secure and Verifiable Information Flow Control

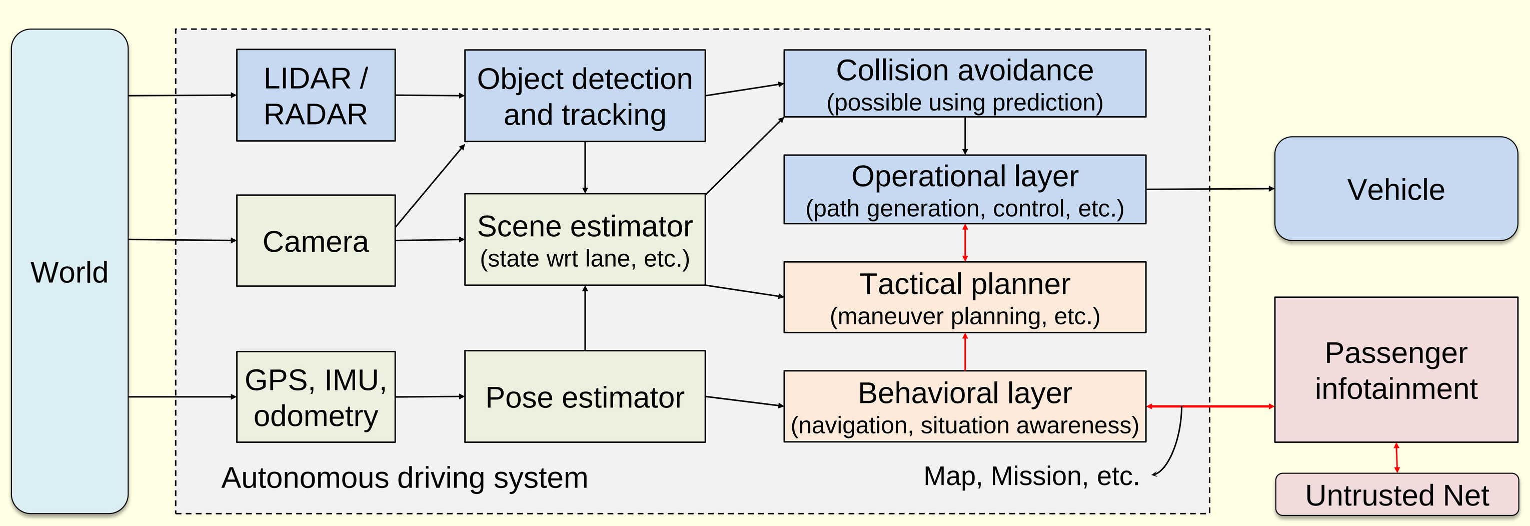
PIs: G. Edward Suh, Mark Campbell, Andrew C. Myers (Cornell University)

Objective

- **Problem**: Safety-critical CPS is turning into complex networked systems ***vulnerable to remote attacks***
  - Internet connectivity + vulnerabilities in complex HW & SW
- **Objective: *Provable*** security assurance for safety-critical collision-avoidance operations of autonomous driving systems
  - **Holistic** assurance across layers: HW, SW, and control algorithms

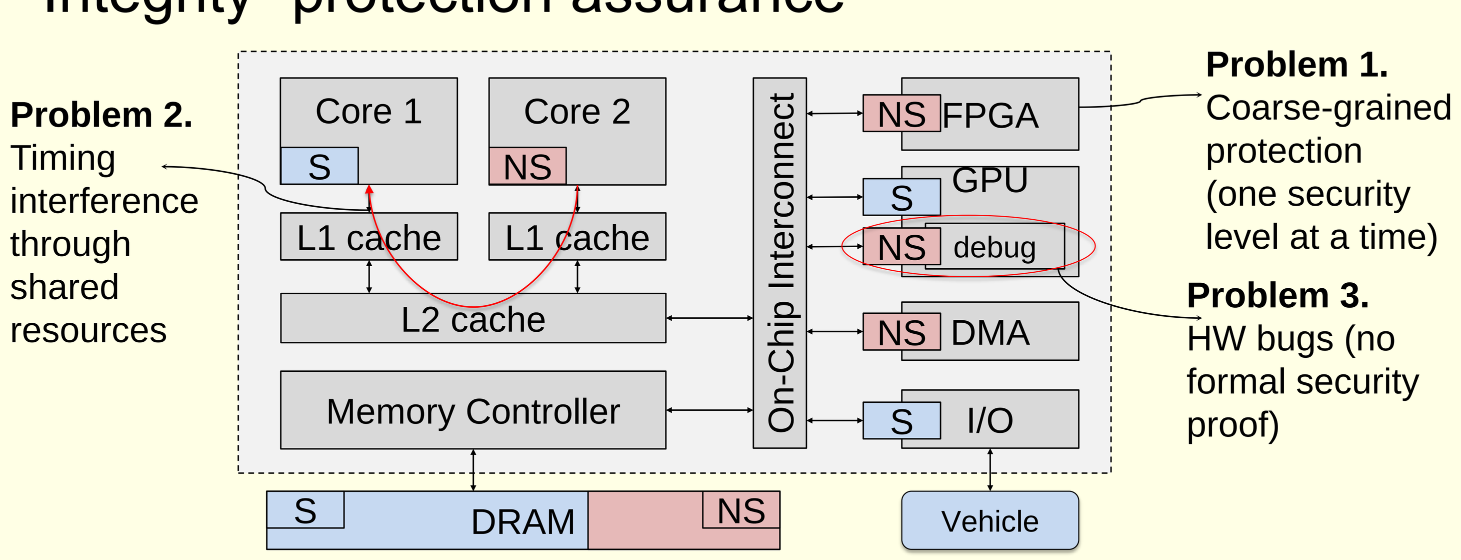
Technical Approach

- **Co-design** hardware, software, and control algorithms
  - Partition autonomous driving systems into multiple security levels
  - Build hardware and software with ***provable full-system information flow control (IFC)*** to ensure safety-critical operations cannot be maliciously affected
  - Develop collision avoidance algorithms to translate security assurance to **quantitative safety assurance**
- Language-based IFC for formal security assurance



Verifiable Hardware Architecture

- Today's hardware is insufficient to protect safety-critical CPS platforms
  - No capability for fine-grained IFC across heterogeneous modules
  - No protection against timing interference
  - No formal security guarantee
- Redesign architecture for comprehensive and verifiable "Integrity" protection assurance



HDL for Hardware Security Verification

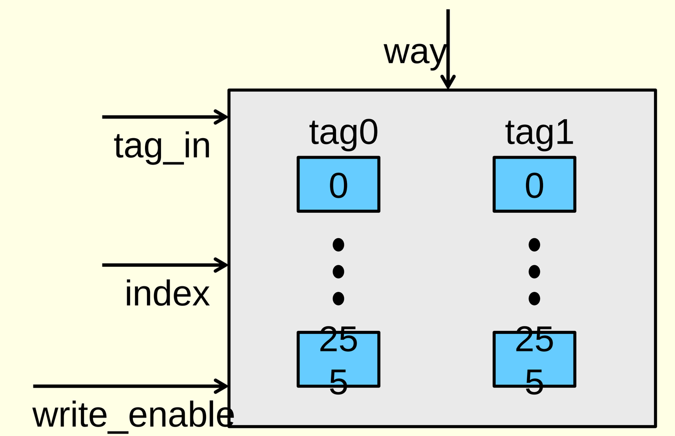
- **Challenge**: need to ***formally verify*** security of HW
- **Approach:** Security type system for Verilog
  - Associate security labels with hardware signals
  - Statically check hardware-level information flows
- Prototype: 5-stage pipelined processor
  - Only needed to change 27 lines out of 1,700

```
reg   [18:0] {L} tag0[256];
reg   [18:0] {H} tag1[256];
wire  [7:0]  {L} index;
// Par(0) = L  Par(1) = H
wire {Par(way)}         way;
wire [18:0] {Par(way)} tag_in;
wire {Par(way)}         write_enable;

always @ (posedge clock) begin
  if (write_enable) begin
    case(way)
      0: tag0[index] = tag_in;
      1: tag1[index] = tag_in;
    endcase
  end
end
```

Security check

Verilog code: cache tag module

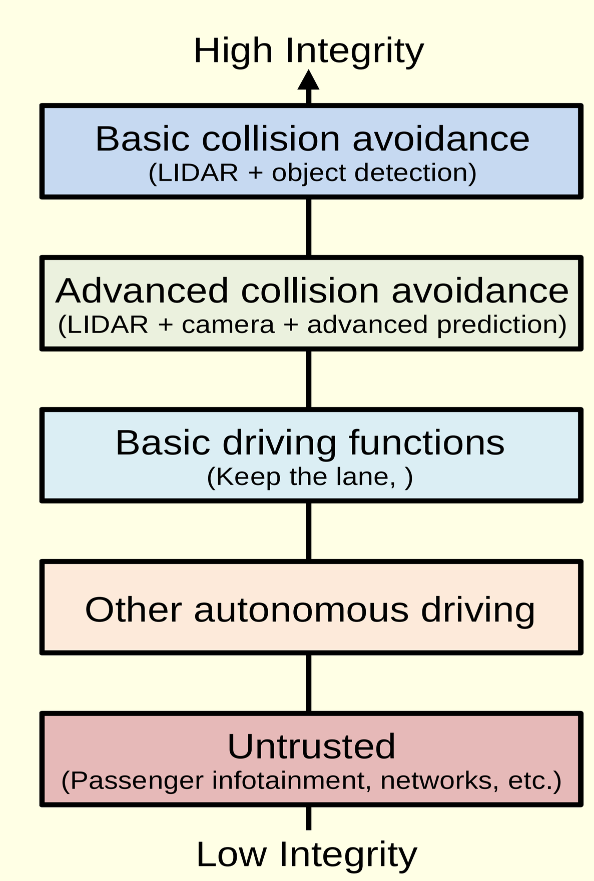
Security check guarantees:
- No explicit information flow from **H** to **L**
- No unintended timing channels: when the label of an instruction is **L**, its execution time should only be affected by **L** hardware state



SW-Level Information Flow Control

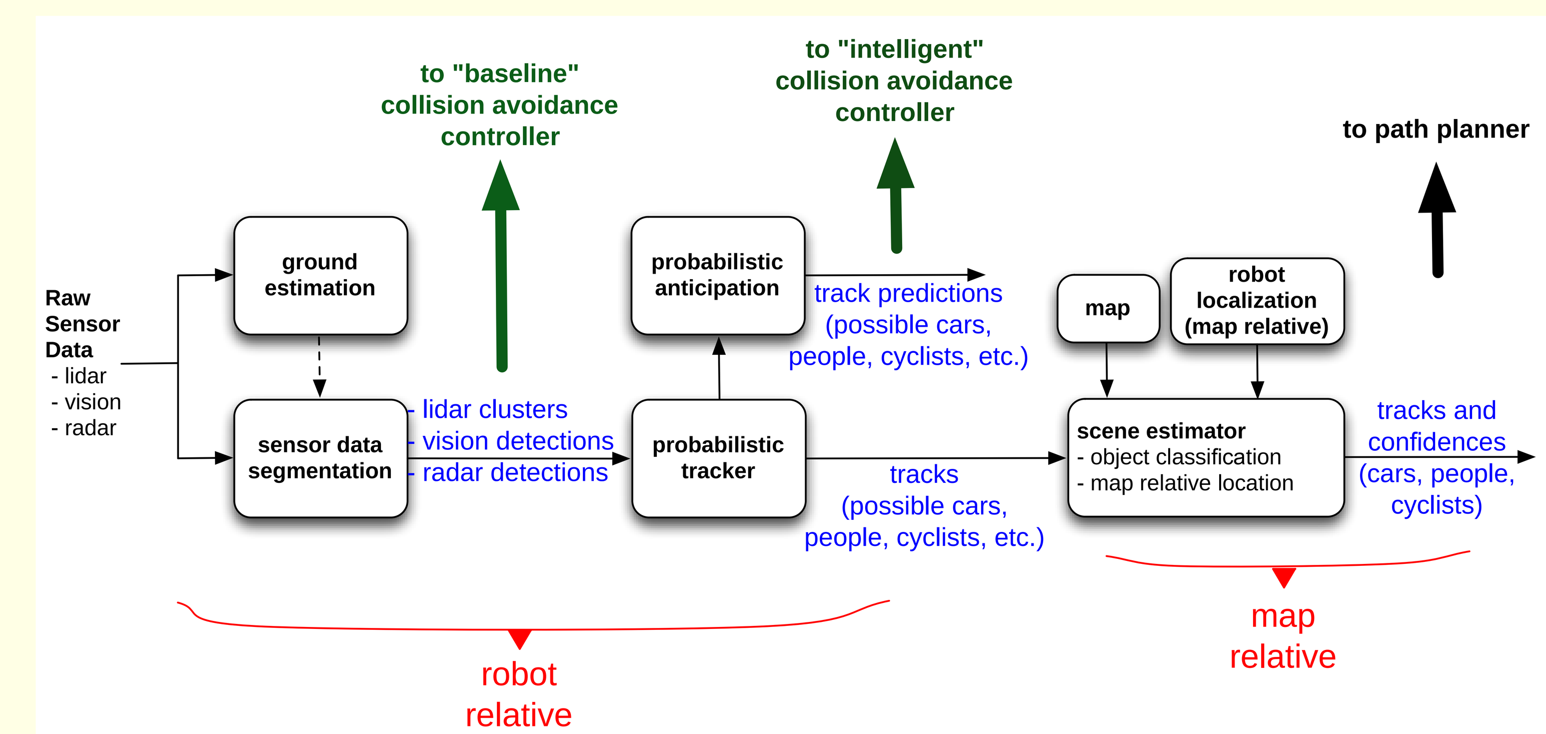
- Extend language-based information flow control to *handle **integrity** and **availability** on **modern SoCs***
  - Previous work focused on confidentiality protection
  - Prove ***the use of correct information flows***, in addition to the absence of undesired flows
  - Handle information flows through heterogeneous computing elements such as GPUs and FPGAs
- Partition autonomous driving software into multiple security levels, and formally verify security
  - Based on safety-criticality
  - Minimize the TCB
  - Allow legitimate information flows

```
MapData{L}  map;
Location{L} destination;
Route{L}    naviplan;
Path{H}     pathplan;

// compute the navigation route
naviplan.genRoute(map, destination);

// make the route high-integrity
endorse(naviplan, L, H);

// generate a vehicle path
pathplan.genPath(naviplan, ...);
```



Collision Avoidance and Safety Analysis

- *Develop collision avoidance algorithms* that leverage the proposed hardware/software platform, and *evaluate/validate the safety* of the integrated system
- Hierarchical collision avoidance algorithms
  - "baseline" algorithm finds objects near the car leveraging sensor data, and takes evasive maneuvers: fast, simple, accurate.
  - "intelligent" algorithm uses probabilistic tracking of objects around the car, and anticipates how and where they may move. Longer time window to avoid collision, but more complex algorithmically.
- Safety assurance of the integrated CPS system
  - Quantitative analysis of the safety–collision probability
  - Investigate the tradeoff between collision probability and security protection levels (timing guarantees, amount of information, size of TCB, etc.)



Autonomous Driving Testbeds

- Plan: use two robots for validation
  - **Segway platform**: Robot with cameras, lidar, and IMU/GPS. Use for year-round testing in controlled environments.
  - **Skynet**: 1 of 6 to complete the 2007 DARPA Urban Challenge. True validation on autonomous driving car.