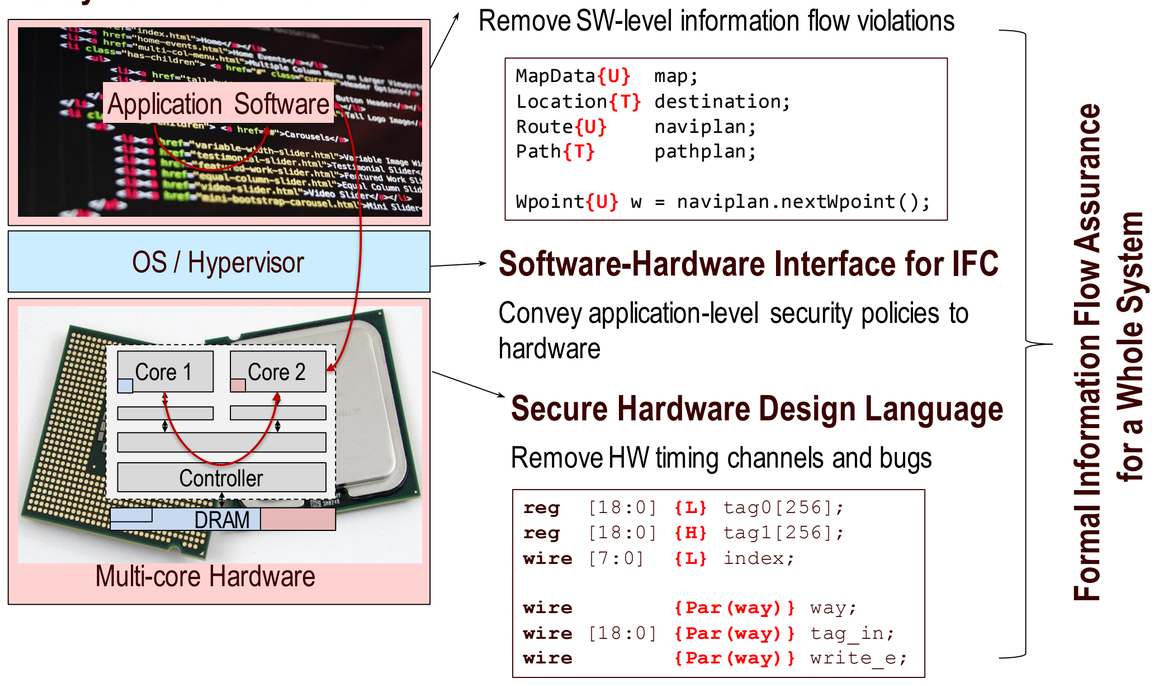
TWC: Medium: Language-Hardware Co-Design for Practical and Verifiable Information Flow Control

PIs: G. Edward Suh and Andrew C. Myers, Cornell University

Objective

- **Problem**: Emerging applications and platforms require strong security assurance not possible today
  - No information leak among VMs in cloud computing
  - Integrity guarantee for safety-critical systems
- **Objective:** Co-design software and hardware with comprehensive, verifiable information flow assurance
  - *All* software-visible information flows in a system

Technical Approach



- Integrated HW/SW approach to **whole-system security** with information flows **verified statically**
- Control information flow 1) within software 2) within hardware, and 3) between software and hardware
- **Formal security assurance** with static information flow analysis at design time

SecVerilog: Secure Typed HDL

- Use a security type system for a Verilog Hardware Description Language (HDL) to control hardware-level information flows
  - Associate security labels with hardware signals
  - Statically check hardware-level information flows
- At the HDL level, timing channels show up as explicit information flows through FSM states

```
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)} wr_en;
```

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

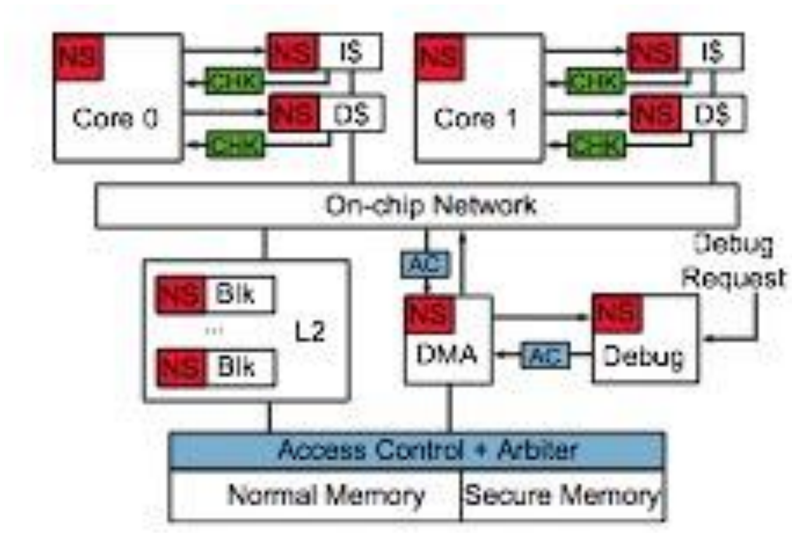
Security check

Security check in the type system 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

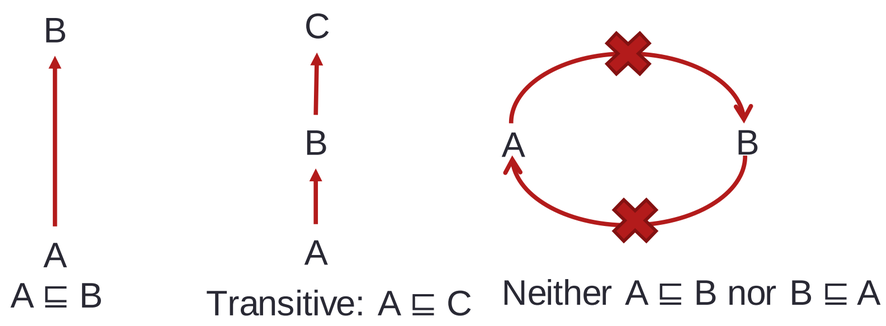
HW Security Architecture Verification

- Security verification of a simple ARM TrustZone prototype using static information flow analysis (ASPLOS'17)
  - Security can be compromised by bugs in implementations of hardware security mechanisms (e.g., TrustZone, SGX)
  - Idea: verify SecVerilog code enforces TrustZone access controls via information flow control
  - Low overheads: 0% slowdown, 0.3% area, 3% extra code



Efficient Timing Channel Protection

- *Lattice priority scheduling* (HPCA'16) prevents insecure timing interference between threads
  - **Insight**: complete timing isolation is expensive and often unnecessary; instead, constrain a subset of timing flows with lattice policy instead.
  - **Result**: 23% average speedup



- *Secure dynamic cache partitioning* (DAC'16) allows more efficient sharing of cache between high- and low-security levels, without timing channel leakage
  - **Result**: 13% avg speedup over static partitioning
- *Quantifiable information flow control for memory controllers* (HPCA'17) enables a principled trade-off between security and performance
  - **Result**: control performance overhead from 16-43%

Ongoing Work

- **Label virtualization:** Develop a new architecture and a software-hardware interface to enable hardware-level enforcement of fine-grained software-level information-flow policies.
- **Full-system prototype:** RISC-V Rocket processor based system on an FPGA. Secure Chisel for static information flow analysis

Interested in meeting the PIs? Attach post-it note below!

The 3rd NSF Secure and Trustworthy Cyberspace Principal Investigator Meeting
January 9-11, 2017, Arlington, Virginia

National Science Foundation
WHERE DISCOVERIES BEGIN