FAVA: Formal Hardware Verification for Architects

Half-Day Hands-On Tutorial

Yao Hsiao, Samantha Archer, Caroline Trippel
Computer Science & Electrical Engineering Departments
Stanford University

June 27, 2026

Stanford | ENGINEERING

Introduction, Motivation, and Our Approach

Caroline

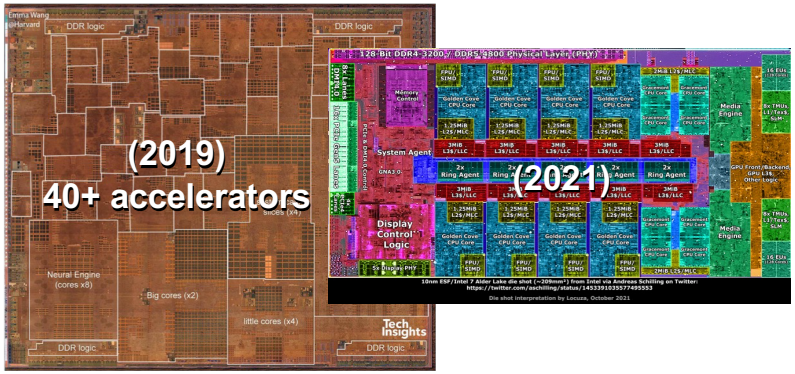
Stanford | ENGINEERING

SYSTEM TRENDS

Modern computer systems are highly complex in both their design and deployments

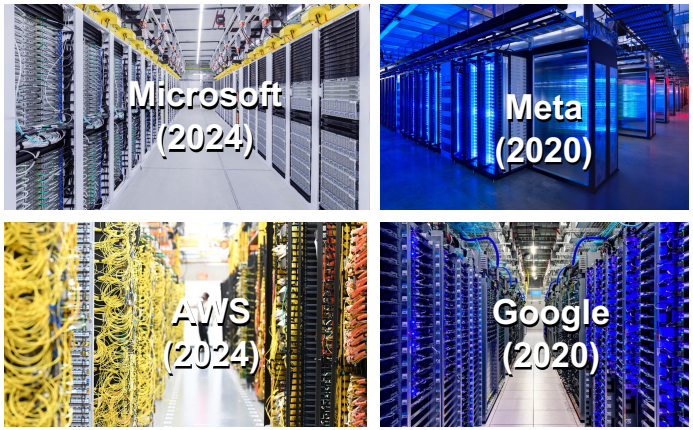
DESIGNS

Combine parallelism, hardware specialization, heterogeneity, and data-dependent optimization to improve performance at manageable power.



DEPLOYMENTS

Are increasingly multi-tenant and hyperscale to reduce cost, improve utilization, and scale applications.



→ *Together, these trends give rise to two critical computer architecture challenges.*

Traditional ISAs—SW-visible state, instruction syntax/semantics—lack details needed to capture how hardware may undermine software assurance.

Correctness MEMORY CONSISTENCY

```
// message passing
init: data = 0; flag = 0

// thread 1   // thread 2
data = 1;     while (flag != 1) {}
flag = 1;     assert (data == 1);
```

Can the `assert` fail?

No on x86. **Yes** on Arm, IBM Power, NVIDIA, RISC-V.

Why? (1) Hardware can reorder memory accesses, (2) threads communicate via shared memory.

Memory Consistency Models (MCMs) - defining permitted load return values - adopted by major ISAs; evolving for complex memory systems.

Security HARDWARE SIDE CHANNELS

```
// victim code
uint8 A[lenA]
uint8 B[64*256]
void victim(size_t i) {
  if (i < lenA) {     // mispred
    val = A[i];       // acc sec
    tmp = B[64*val]; // leak sec
  }
}
```

Can `victim` be exploited to read arbitrary memory?

Yes on most processors (Spectre).

Why? (1) Hardware can predict control-/data-flow, (2) unsafe insts' operands can leak via HW side-channels.

Side-Channel Leakage Contracts - defining unsafe instruction-operand pairs - adopted by Arm, x86, RISC-V; evolving for Spectre attacks.

arm intel RISC-V

br, cond
ld, addr
st, addr
div, op0
div, op1

Verification is a major and growing hardware design bottleneck

Critical bugs escape to first-silicon in >85% of HW projects (20-yr first-silicon success low!) w/ >50% effort on verification [Siemens, 2024].

GETTING WORSE →

Due to more contracts, driven by design diversity and solving The Contracts Challenge; and more bugs + shorter design times, driven by AI design.

Why is verification a bottleneck?

Translating abstract end-to-end contracts into detailed end-to-end formal specs is hard.

Golden ref models & formal temporal logic (TL) properties (e.g., SVAs) demand significant expert manual effort to construct.

FIFO design EX. END-TO-END SPEC

```
// FIFO-ness SVA: If d1 enqueues before
// then d1 dequeues before d2
assume property (d1_enq |-> !d2_has_enq
assert property (d1_deq |-> !d2_has_deq
```

Natural Language Contract

…

A1 When the worker is not selected by the decoder, all control signals shall be low.
A2 When HTRANS is IDLE, all control signals shall be low.
A3 First transfer of any sequence is NONSEQ in nature.
A4 Non-first transfer of an AHB sequence will always be SEQ in nature.
A5 Burst sequence of length four shall end at fourth occurrence of HREADY.
A6 If this is last transaction of a sequence and next cycle is not start of another sequence, HTRANS shall be IDLE in next cycle.
A7 If HREADY is low, then all control signals, address and data buses shall hold their values.

…

[Arm, AMBA AHB Protocol Specification]

gainst

Expert Manual Effort [Godhal+, STTT'11]

Auto-Generate w/ AI + AR

SynthTL • [Mendoza+, FMCAD'24]

ecomposition

proof

it-entries

Formal Temporal Logic Specification

```
…
∧((G (!_hsel_ -> (((_htransidle_ ∧
_hburstsingle_) ∧ !_hwrite_ ∧
!_start_ ∧ !_last_)))) ∧ (G
((_htransidle_) -> ((_hburstsingle_)
∧ !_hwrite_ ∧ !_start_ ∧ !_last_))) ∧
(G (_start_ -> (_htransnonseq_))) ∧
(G (!_last_ ∧ (_htransnonseq_) ∧
_hready_ -> X (_htransseq_))) ∧ (G
((_hlock_ ∧ (_hburstincr4_) ∧
_hready_ ∧ (_htransnonseq_)) -> X
(_htransseq_ U (_htransseq_ ∧
_hready_ ∧ X(_htransseq_ U
(_htransseq_ ∧ _hready_ ∧
X(_htransseq_ U _hready_)))))))) ∧ (G
((_last_ ∧ X !_start_) -> X
(_htransidle_))) ∧ (G ( !_hready_ ->
(((_htransnonseq_ <->
…
```

OUR WORK: OVERVIEW

Enabling High-Assurance—Correct, Secure, Reliable—Computer Architectures

	Correctness	Security	Reliability
CHALLENGE #1 **The Contracts Challenge** Specification of HW-SW contracts for current- and next-generation HW, plus design of SW analyses and HW implementations based on them	MEMORY CONSISTENCY **MemGlue** • FMCAD'24 • **FMSD'26 (Selected as a top FMCAD'24 paper)** **DAM** • DIMES'25 · **TransForm** • ISCA '20 **ILA-MCM** • FMCAD'18 · **ArMOR** • ISCA'18	HARDWARE SIDE-CHANNELS **Helium** • ISCA'26 · **Protean** • HPCA'26 **Serberus** • SP'24 • **HES Top Pick'26** **Clou** • ISCA'22 · **MLDs** • ISCA'21 OTHER **Porcupine** • PLDI'21	PERFORMANCE **RAMSIS** • EuroSys '24 **RecShard** • ASPLOS '22 **RecSSD** • ASPLOS '21
CHALLENGE #2 **The Verification Challenge** Design of automated formal verification approaches targeting current- and next-generation HW-SW contracts to verify HW implementations of them	FUNCTIONAL **ARTEMIS** • ICSE'26 · **RTL2MμPATH** **SynthTL** • FMCAD'24 · **nl2sva** • DAV'23 **nl2spec** • CAV'23 · **G-QED** • DAC'23 **A-QED** • FMCAD'21 MEMORY CONSISTENCY **Blend** • In Prep. · **qcmbr** • Under Sub. **RTL2μSPEC** • MICRO'21 **TriCheck** • ASPLOS'17 • **Top Pick'21**	HARDWARE SIDE-CHANNELS **SynthLC** • MICRO'24 • **HES Top Pick *Shortlist*'26** **CheckMate** • MICRO 18 • **Top Pick'19** • **HES Top Pick *Shortlist*'22**	PERFORMANCE **CounterPoint** • ASPLOS '26 • **Best Paper Award** SILENT DATA CORRUPTION **ITHICA** • Under Sub. • arXiv '26

Today's focus · Other highlights (touched on very briefly)

Automated and scalable end-to-end formal hardware verification via synthesizing HW-SW contracts from hardware implementations