RISCV Verification – Opportunities and Challenges

Verification Futures Austin

Divyang Agrawal
Sr. Director

Sep 2023

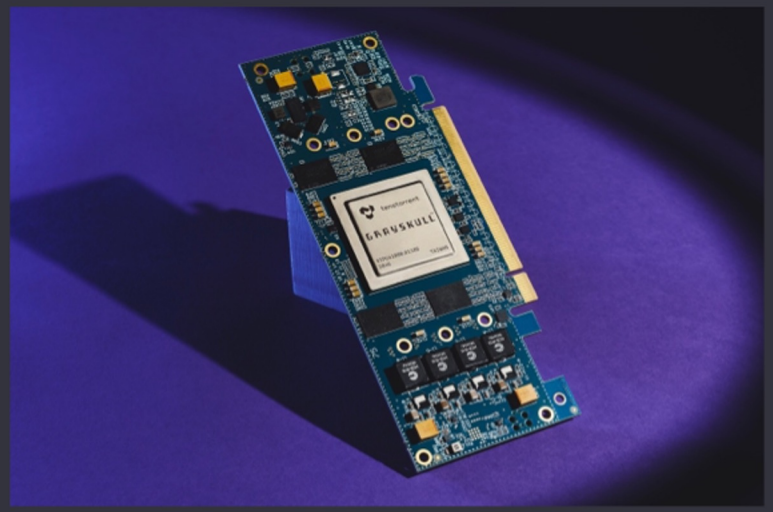
tenstorrent

Agenda

- Introduction and RISC-V Processor Family
- CPU Design and Verification
- RISCV
- OS Boot Case Study

Introduction

Tenstorrent builds computers for AI. Our mission is to address the open-source compute demands for software 2.0 through industry-leading AI/ML accelerators, high-performing RISC-V CPUs, and infinitely-configurable ML and CPU Chiplets.



Chip Roadmap

2021 | 2022 | 2023 | 2024

ML Processor

Grayskull

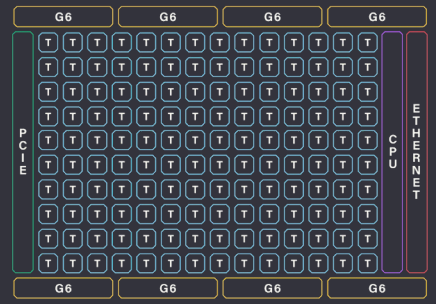
12nm, 620mm^2
315 8b TFLOPS
PCIE gen 4
8 channels LPDDR4

Networked ML Processor

Wormhole

12nm, 650 mm^2
350 8b TFLOPS
400GB/sec ethernet
6 channels GDDR6
16 lanes of PCIE gen 4

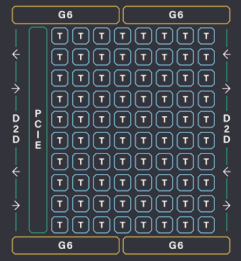
Standalone ML computer



Black Hole

6nm, 600mm^2
1000 8b TFLOPS
1200GB/sec ethernet
8 channels of GDDR6X
32 lanes of PCIE Gen5

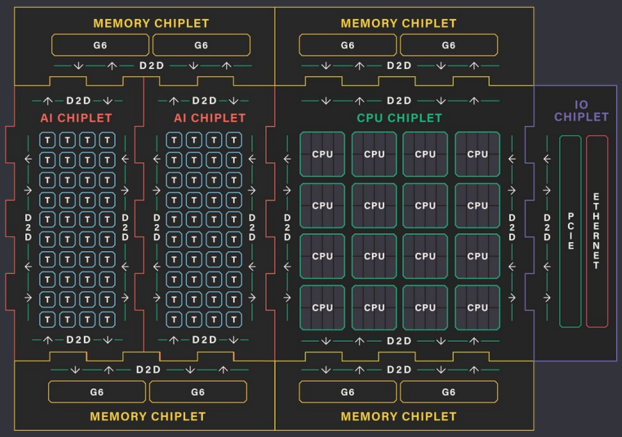
Low Power, Low Cost ML Chiplet



Quasar

4nm, 250mm^2
650 8b TFLOPS
4 channels of GDDR6
16 lanes of PCIE Gen5
11TB/sec d2d interface

Highly Configurable, Highly Performant ML Chiplet Complex



Grendel

3nm AI & CPU Chiplets
7nm Memory & IO Chiplets
Expandable TFLOPS
GDDR6 Channels
PCIE Lanes 128 RISC-V Cores
Non-blocking d2d interface

tenstorrent

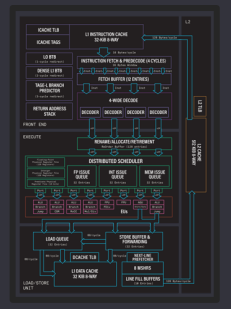
Problem Statement: CPU Design and Verification

Goal: RISC-V O-o-O Processor Family

Performance

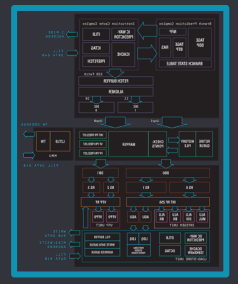
Higher Performance

Open & Free

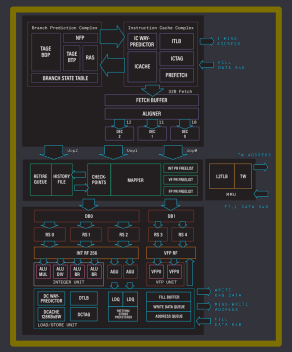


4-Wide Decode

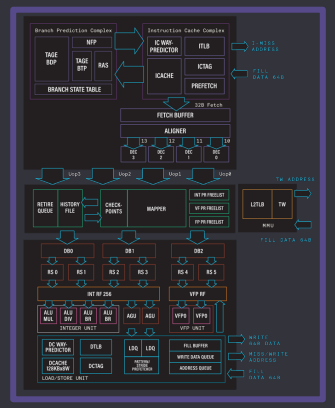
Sonic Boom with Vector



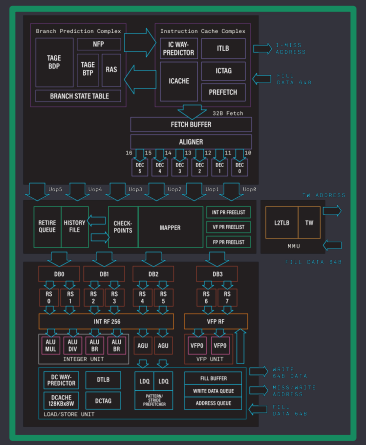
2-Wide Decode



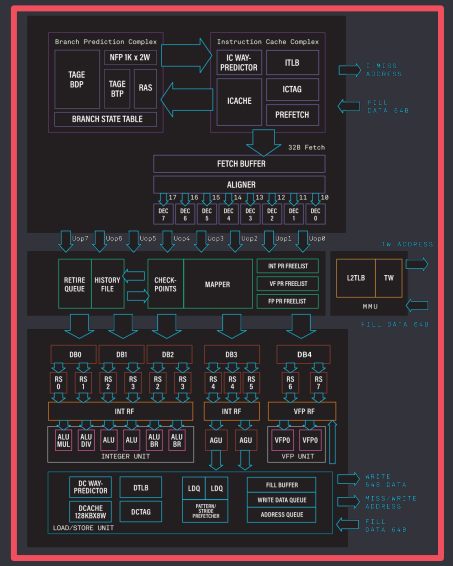
3-Wide Decode



4-Wide Decode



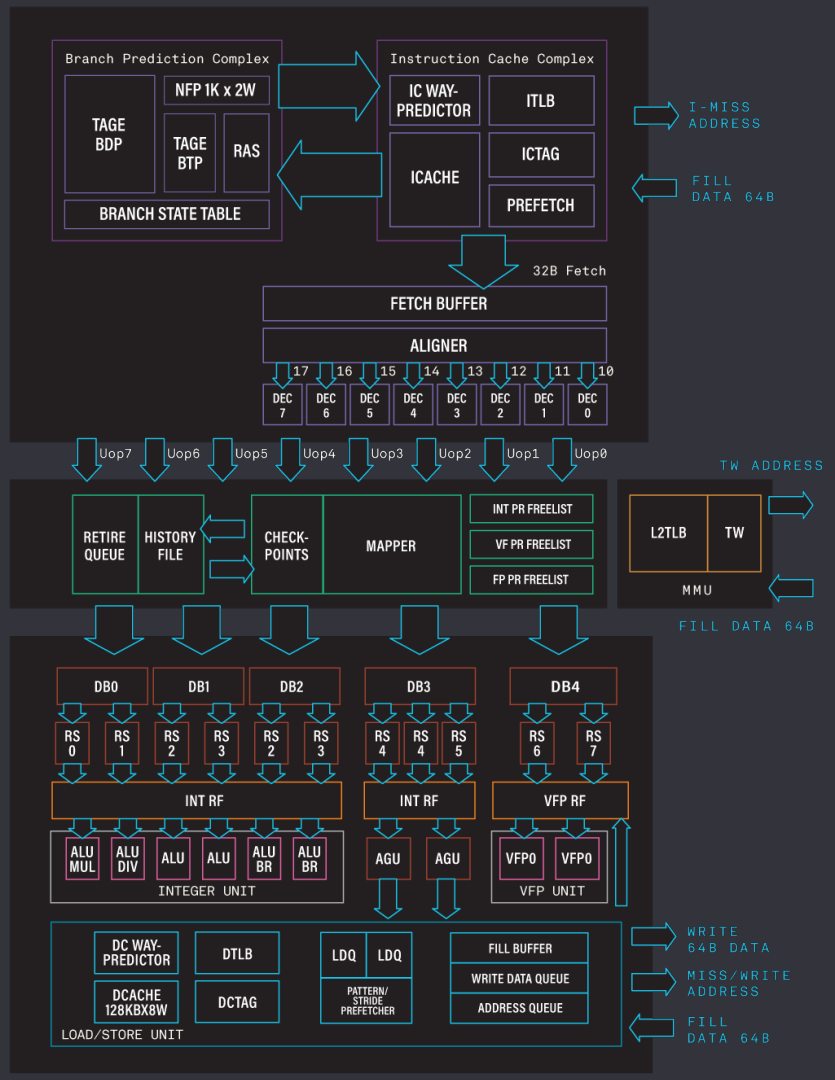
6-Wide Decode



8-Wide Decode

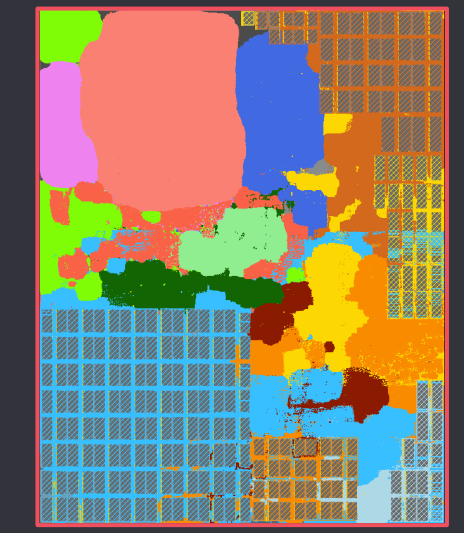
Decode Width

Ascalon: High Performance O-o-O Superscalar Processor



RV64IACFDHMV

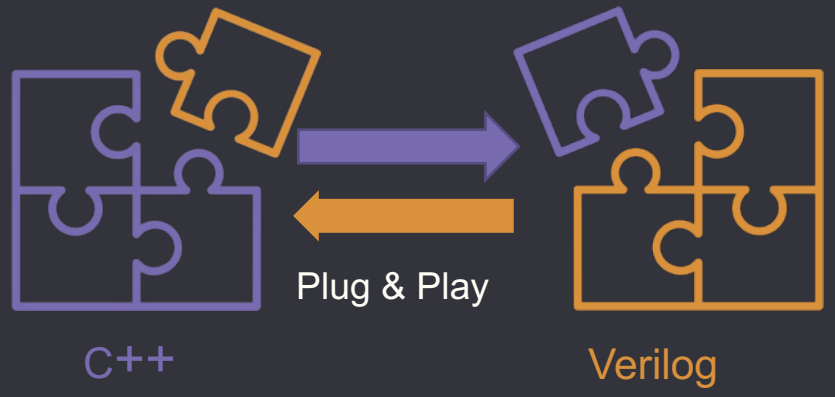
- Advanced branch predictions
- 8-wide decode
- 3 LD/ST with large load/store queues
- 6 ALU/2 BR
- 2 256-bit vector units
- 2 FPU units



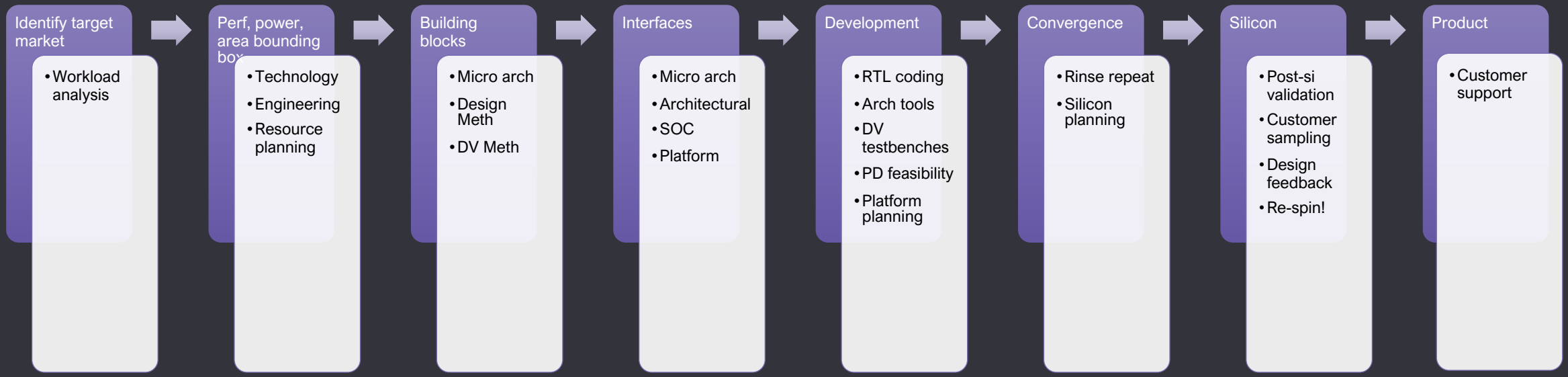
CPU Design

- Design CPU is complex
  - High functional complexity
  - Meeting performance, power, and area goals even harder
  - Big team prone to communication errors

- Design flow innovation
  - Clean interfaces enforced by design modularity
  - Design abstraction compatibility
    - C++ == function model == RTL
  - Plug-and-play modules for co-simulations



Typical High Performance CPU Development Cycle

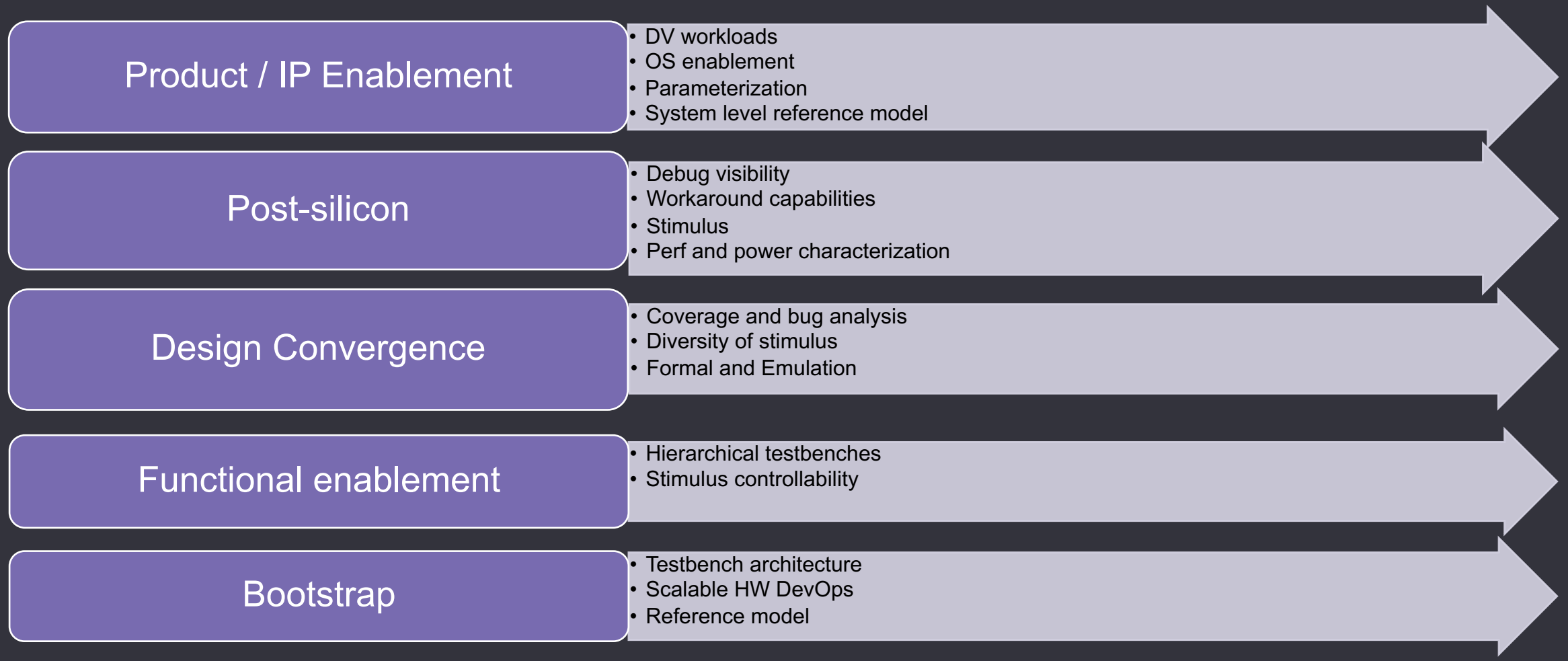


What does this mean for Verification?

Verification State Space

- Product / IP enablement (OS, application software, firmware)
- Post-silicon (performance, power, functionality)
- Design convergence
- Functional enablement
- Design bootstrap
- Specification

Lessons learned: DV starts with a Product and Silicon Mindset



RISCV

Why RISC-V?

Requirements	RISC-V	ARM
Open-Source Architecture	Yes	No
Addition of Custom Extensions	Yes	No
ISA extensions for high-perf	Yes	Yes
ISA extensions for ML	Yes	Yes
Toolchain and software ecosystem availability	Yes, significant work remains but Linux-like growth	Yes
Deployment	Low	High
Segment growth	High	Low

RISCV Difference

- No legacy technical debt (x86: real mode, ARM: Aarch32)
- Base ISA + Extensions
  - Reference model is modular by design
  - Legacy stimulus baggage significantly reduced
  - Custom instructions, at will!
  - Custom data formats!
- How does this help?
  - Reduced cost of enduring 1st generation design decisions
  - Lends itself much more cleanly to Formal on the ISA
  - New extensions will see faster adoption

RISCV Verification – Challenges and Opportunities

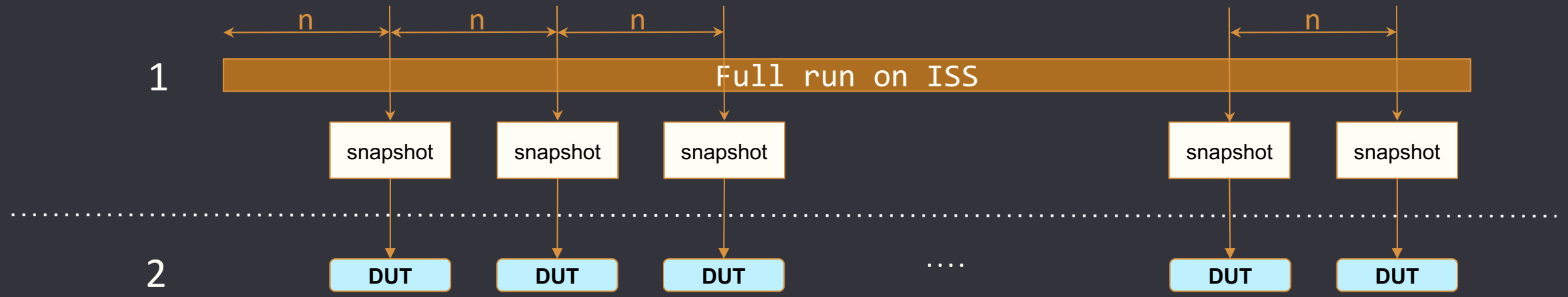
Challenge	Opportunity	Potential Alternatives
General lack of silicon workloads	Conversion from x86 and ARM	Open source!
Limited open source or vendor stimulus generation tools	Need a portfolio of products targeting ISA, System Arch, Platform	Both open source and commercially available solutions
Large scale deployment of software stack	Trailblazers with large footprint	Open source!
Fractured architectural compatibility views	RVI + collaborative efforts	
Application programmer's guide		Significantly deficient options compared to x86 and ARM
Reference platforms, Devkits		

OS Boot DV Case Study

Simulation OS Boot

- Problem Statement
  - Linux has ~66M instructions; will take a month to run on simulation
  - Onion peeling the issues - not feasible

- Approach
  - Pass #1: Run target program on ISS
  - Produce a snapshot of memory and all registers after “n” instructions
  - Pass #2: Execution may be resumed by playing a specific snapshot on a DUT



Open Source

Tenstorrent Open Source Collateral

- Ocelot: BOOM with Vector Unit
  - https://github.com/tenstorrent/riscv-ocelot/tree/ocelot
- TT Whisper: Instruction set simulator
  - https://github.com/tenstorrent/whisper
- RISCV DV Kit
  - https://github.com/tenstorrent/rv-core-dv-kit
- RISCV Vector Tests
  - https://github.com/tenstorrent/riscv_arch_tests/tree/main/riscv_tests/rvv
- RISCV Arch Checker and Stimulus
  - https://github.com/tenstorrent/cosim-arch-checker
  - https://github.com/tenstorrent/riscv_arch_tests

Backup

Whisper: Open Source Instruction Set Simulator

- Provides infra for DV, perf modeling and debugging RISC-V software
- Supports multi-hart multi-core systems
- Supports all major RISC-V extensions
- Fast – over 150M instr/sec
- Used in DV in multiple ways
  - Conventional core-level lockstep checking
  - Block-level interface checking
  - Ex: Models micro-ops that can be used for frontend unit interface checking
  - Disassembler for RISC-V instructions
- Used in architectural modeling and exploration
  - Trace generation
  - Benchmark analysis

Example: Legacy Architectural Painpoints, X86 and ARM

	X86	ARM	RISCV
Basic Bootup Sequence (# instructions)	1000s	100s	<5 instructions
Compatibility testing (# tests)	100s of thousands	100s of thousand	~1000, hierarchical
Architectural State	High	Medium	Low
Cost of adding new ISA features	Very high	Very high	Low-ish

- Architectural complexity has grown exponentially while ISA innovation has increased perhaps linearly
- Software to deal with legacy arch increasingly more complex
- Time to evaluate 30+ years of baggage on legacy architectures