OPEN SOURCE HARDWARE / SOFTWARE CO-DESIGN

David Donofrio, Chief Hardware Architect

John D. Leidel, Chief Scientist

{ddonofrio,jleidel}@tactcomplabs.com

Anastasiia Butko, Scientist, Lawrence Berkeley National Lab

Jeffery Young, Scientist Georgia Tech

December, 2021

AGENDA AND SPEAKERS

- David Donofrio & John Leidel - *Tactical Computing Laboratories*
  - Overview of Open Source Hardware development flows
- Anastasiia Butko - *Lawrence Berkeley National Laboratory*
  - Open2C: Open-source Generator for Coherent Cache Memory Subsystem
- Jeffery Young - *Georgia Tech*
  - Novel RISC-V Implementations and Accelerators

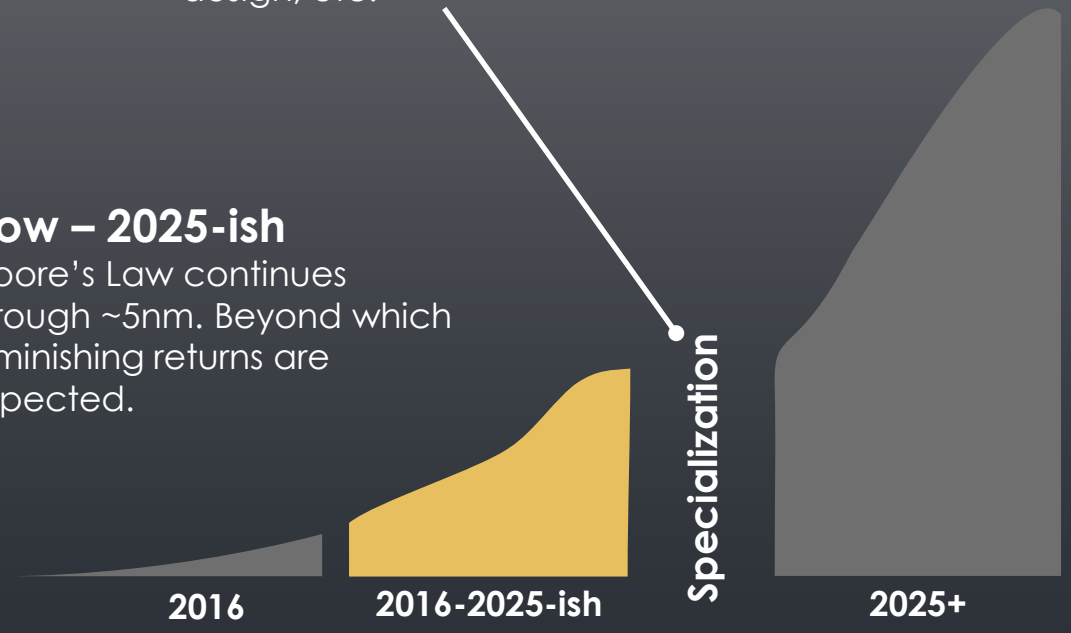
POST MOORE TECHNOLOGY CURVE

GREAT *OPPORTUNITIES* EXIST FOR INNOVATION THROUGH THE END OF MOORE'S LAW

End of Moore's Law
End of Moore's Law requires a different set of optimizations to continue performance scaling. Opportunities for additional specialization, reconfigurable computing, hardware / software co-design, etc.

Throughout…
Continued increases in parallelism and heterogeneity will require advanced runtimes, programming environments and compiler optimizations in order to take full advantage of these new architectures

Now – 2025-ish
Moore's Law continues through ~5nm. Beyond which diminishing returns are expected.



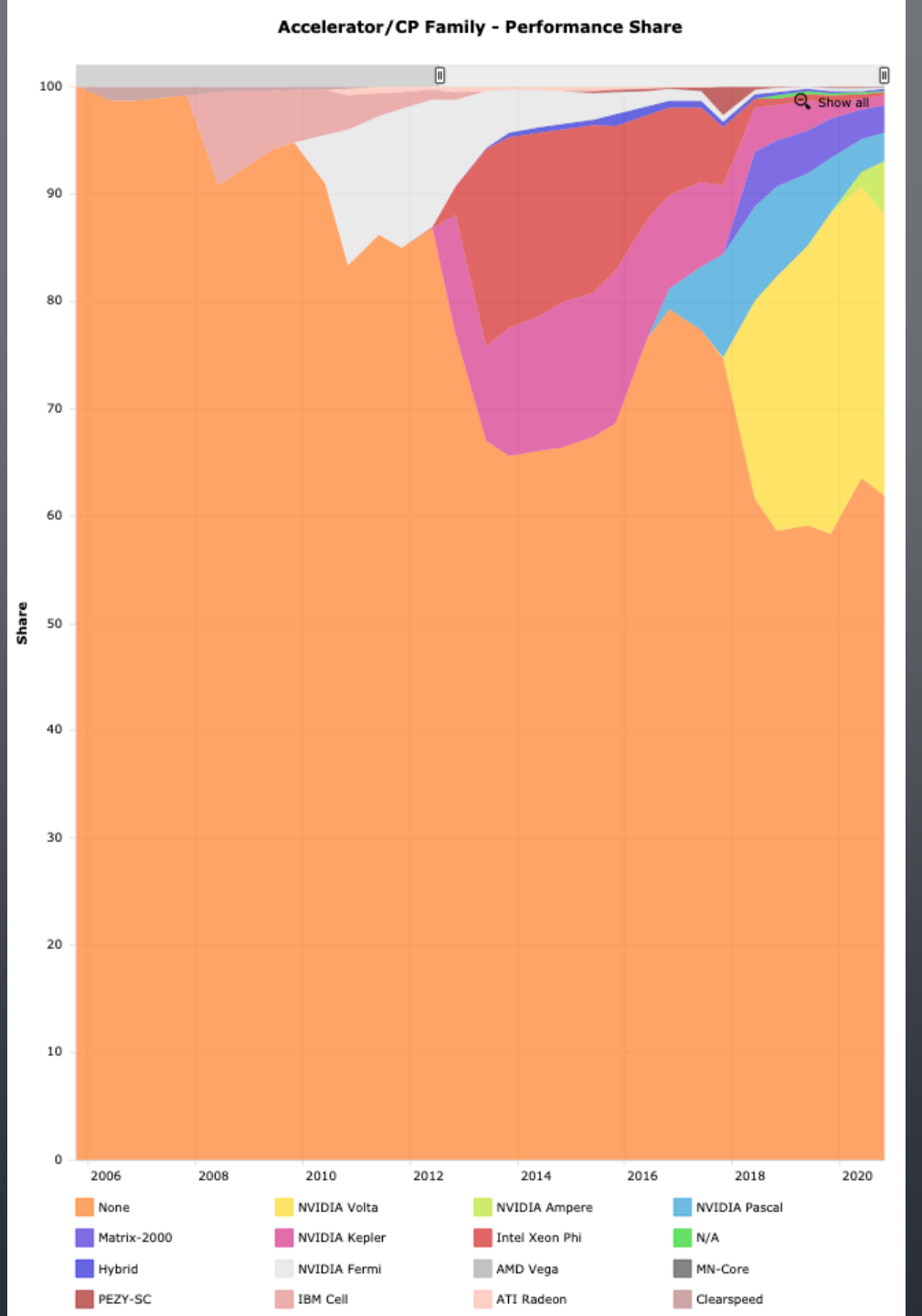
Post Moore Scaling
New materials possibly introduced to allow continued process scaling.

Specialization as the path to performance

- Process Scaling Tapering off
- Specialized architectures are everywhere
  - GPUs
  - AI Accelerators
  - Re-configurable
  - Matrix accelerators
- Increasing heterogeneity in systems
  - Heterogenous in both types of computation as well as where computing is performed
  - As computing becomes more decentralized we will need a greater variety of accelerators
  - This drives a need for frameworks that allow rapid design space exploration

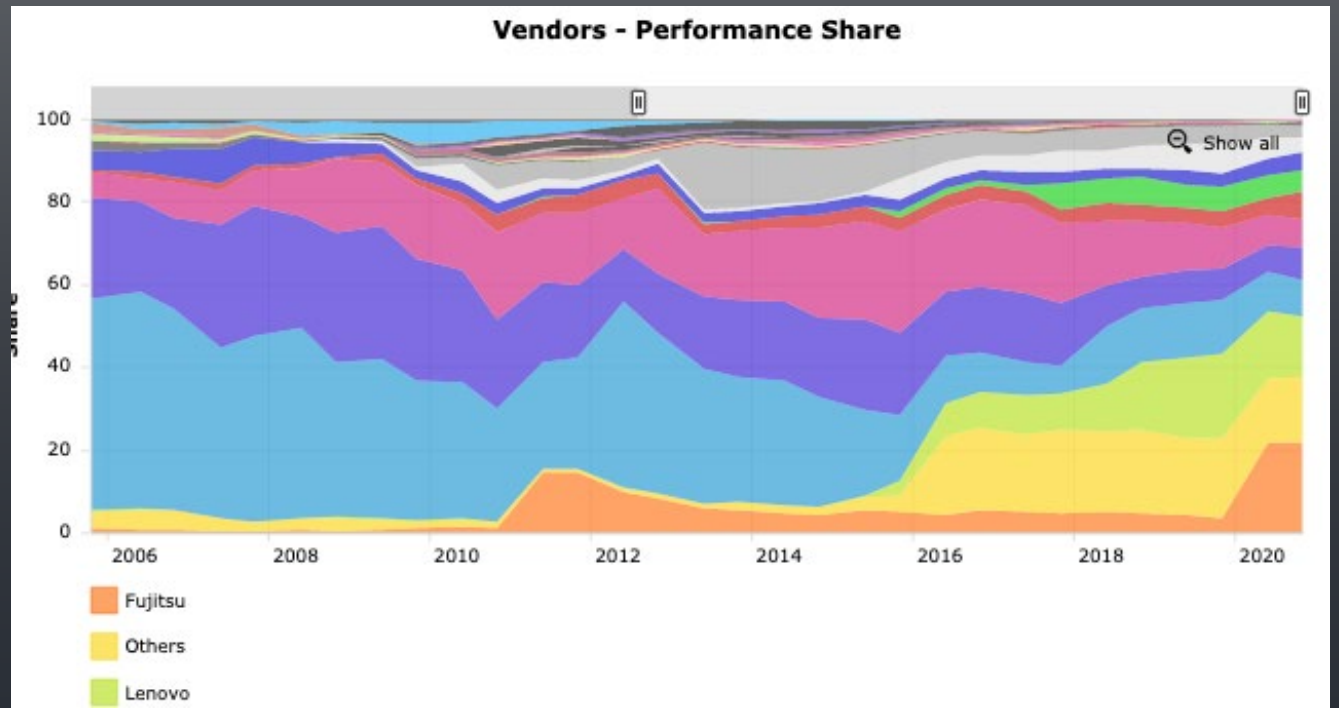
ACCELERATORS CRITICAL TO PERFORMANCE

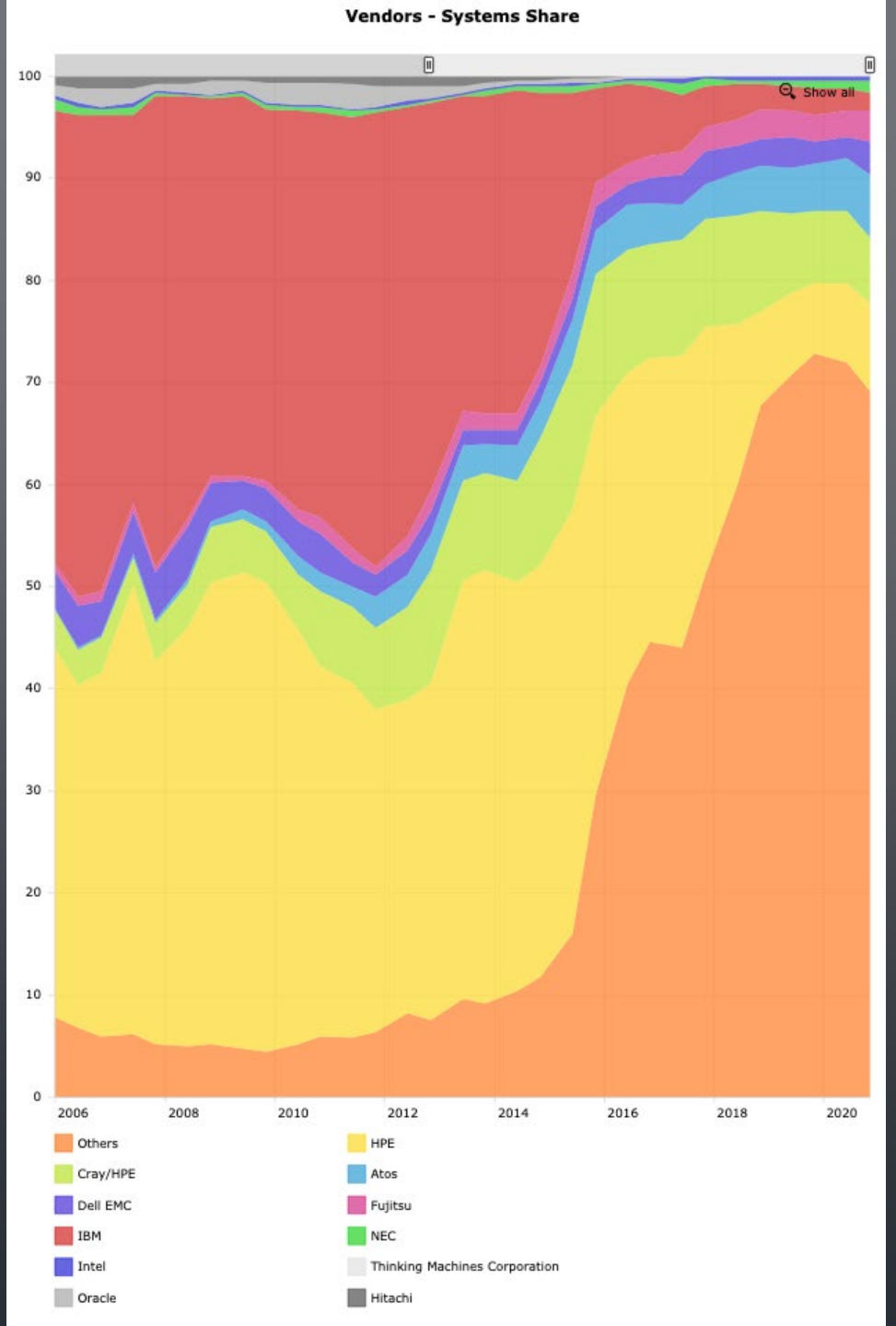
- TOP 500



VENDOR COUNT INCREASING

- DEMOCRATIZATION OF HPC?





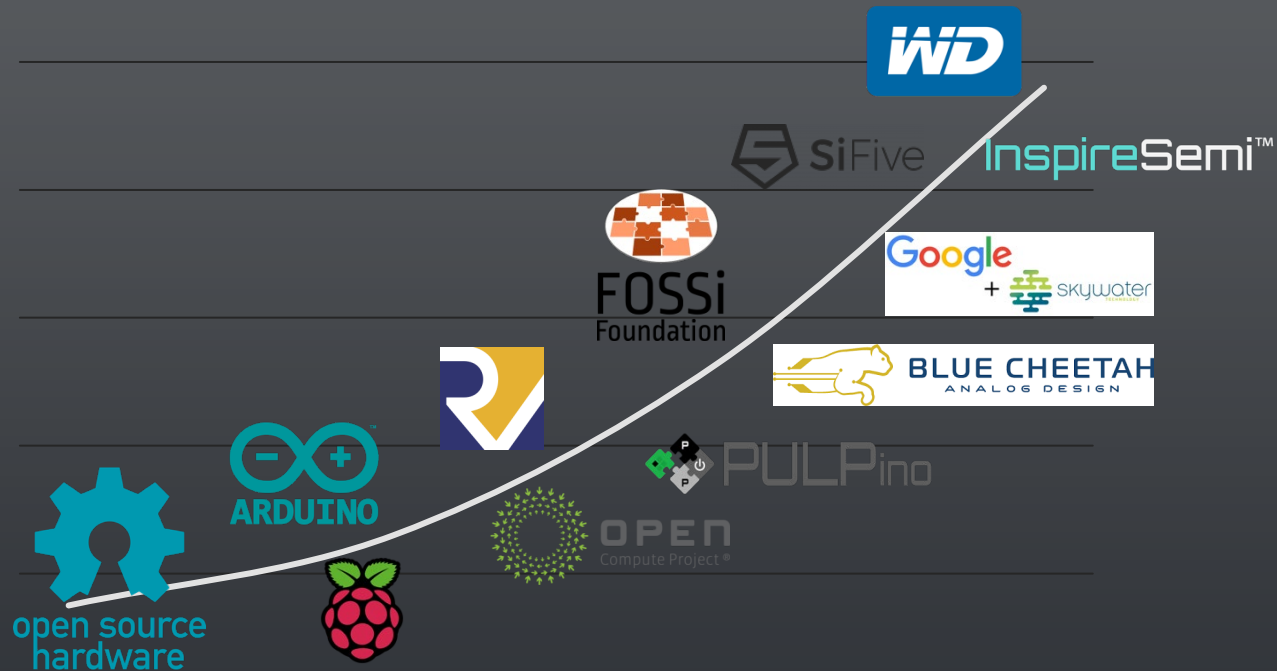
OPEN SOURCE: DRIVING INNOVATION

Open Source Software

Over 80% of companies run Open Source Software

1991 - Linux 1.0 → Today

Open Source Hardware



Today →

OPEN SOURCE CORES QUITE CAPABLE

- OPEN SOURCE IMPLIES LOW COST… NOT NECESSARILY LOW PERFORMANCE

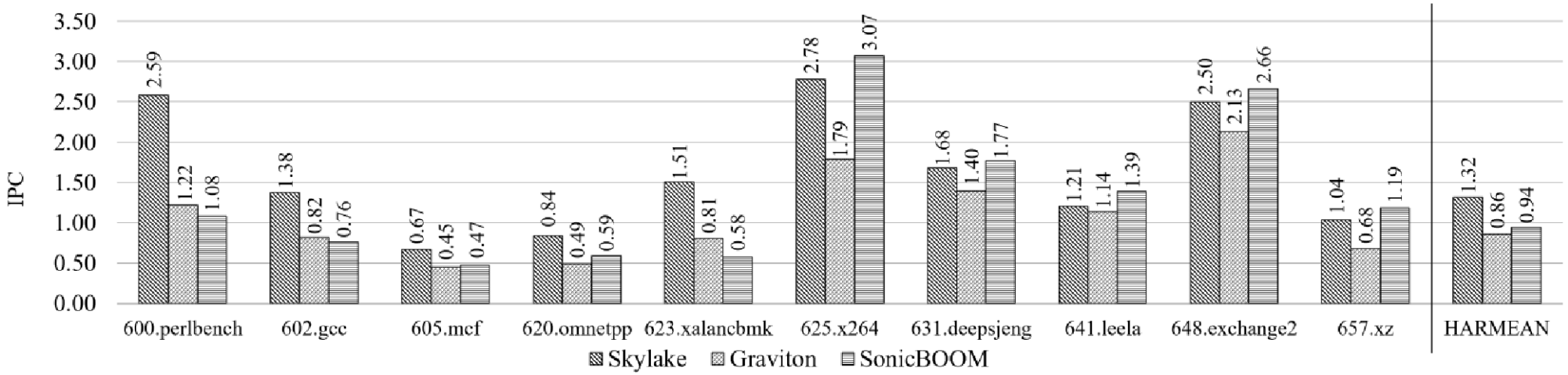
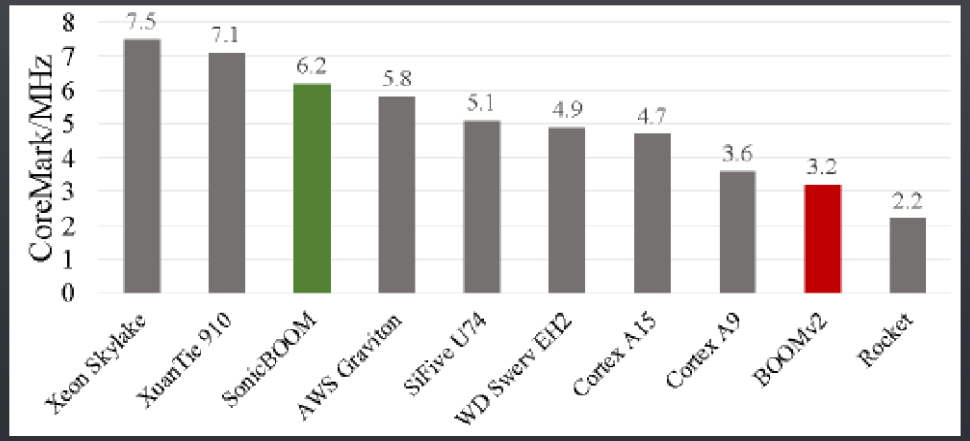


Figure 6: *Sonic*BOOM SPEC17 IPC compared to Intel Skylake and AWS Graviton cores.



Results from SonicBOOM
Zhao, et. al. UC Berkeley

Open Source HW is more than just ASICs

- Hardware Generation methodologies
  - Enable FPGA computing
  - Enable architectural comparison
    - Common Baselines
  - Faster iteration
- More effective than sharing point designs
  - Rocket Core vs. OpenCores
- Co-generation of software stack desirable

TRULY OPEN ASICS

- Hardware generation tools typically solve the digital design and simulation problem
- Many more components needed for a complete asic
  - PLL, JTAG, SRAM, Vreg…
- ASIC PDKs typically obscured behind NDAs, etc.

TRULY OPEN ASICS

- OpenLane is a complete, open source, RTL to GDSII Flow
  - Combination of many existing open source flows
    - Yosys, magic, OpenRoad, Netgen, and many others
  - https://github.com/efabless/openlane
- Google has partnered with skywater fab
  - 130nm PDK open source
    - Apache License
  - Shuttle runs are now ***FREE*** and open to ***ALL***
    - Final design MUST Be 100% open to qualify

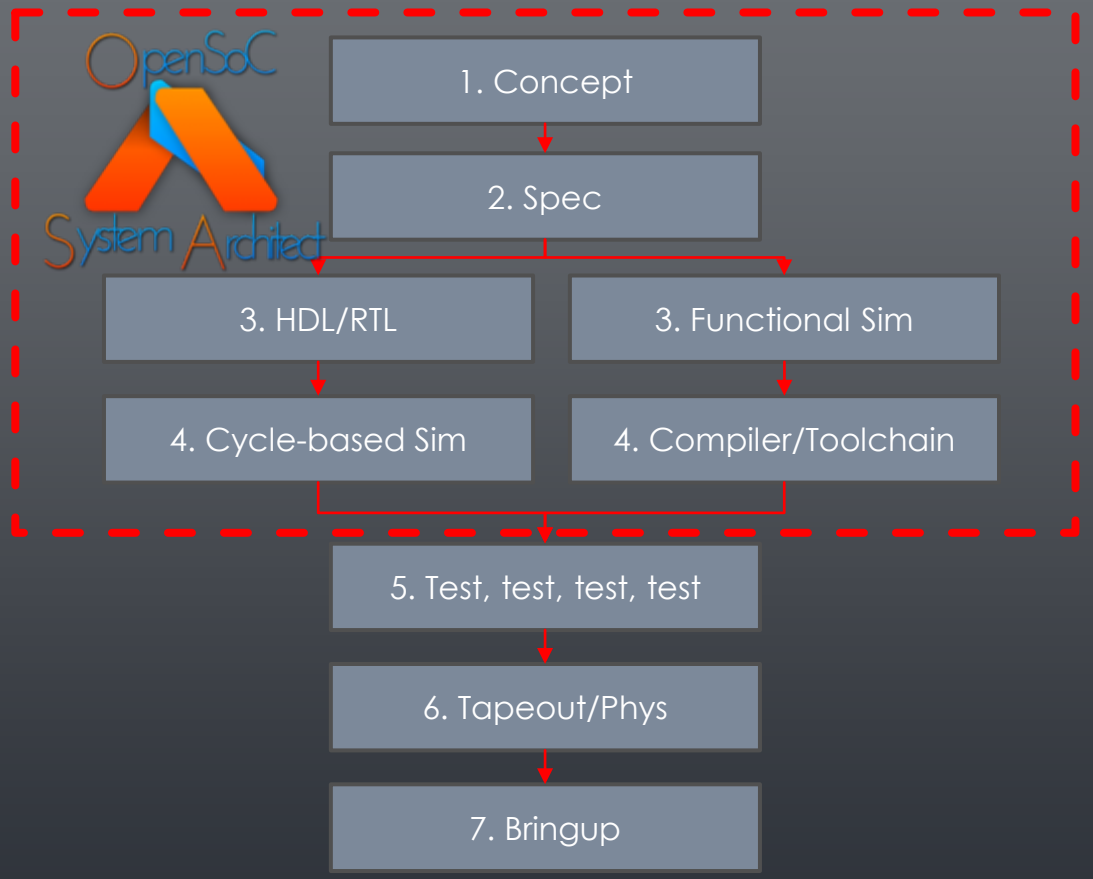
A BIT OF HISTORY

- LBNL & Texas Tech were funded to develop a special-purpose, RISC-V based SoC
  - FPGA-based
  - HMC memories, Custom ISA extensions, Scratchpads
- PhD dissertation
  - GoblinCore-64: RISC-V based SoC for data intensive computing
  - ISA extensions, pipeline changes. SMT
  - No caches, memory coalescing

SYSTEM ARCHITECT GEN1

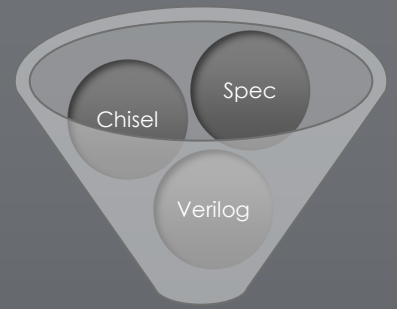
- This "Rinse and Repeat" design flow can be automated!
- System Architect Gen1 was born as a toolchain for developing RISC-V extensions
- Utilized compiler-techniques to handle dependence analysis of design parameters to RISC-V internals

- **Input:** RISC-V core & design parameters
- **Output:** Chisel RTL for entire SoC & LLVM compiler



SYSARCH TOOL CHAIN & FLOW

- OpenSoC Tool Chain & Flow consists of several integral moving parts:
  - OpenSoC System Architect Frontend
  - CoreGen IR & Backend
  - OpenSoC Fabric NoC Interconnect
  - RISC-V Standard Cores (in Chisel)
  - LLVM Compiler Infrastructure
- Users input design specs and extensions via frontend
- CoreGen ensures design correctness and continuity
  - Also generates backend Chisel and LLVM implementation
- OpenSoC Fabric "glues" all intermediate modules together with scalable NoC



Frontend & CoreGen

LLVM Compiler

SYSTEM ARCHITECT GEN2

- Several early users posed an interesting question:
- ***What about developing arbitrary ISA's?***
- **Gen2 is a complete re-write of the infrastructure to support arbitrary ISA development**
  - New intermediate representation (formatted in YAML)
  - New dependence analysis engine (CoreGen)
  - High Level instruction definition language (StoneCutter)
  - Support for Plugins

Maintain verification performance similar to a modern optimizing compiler!

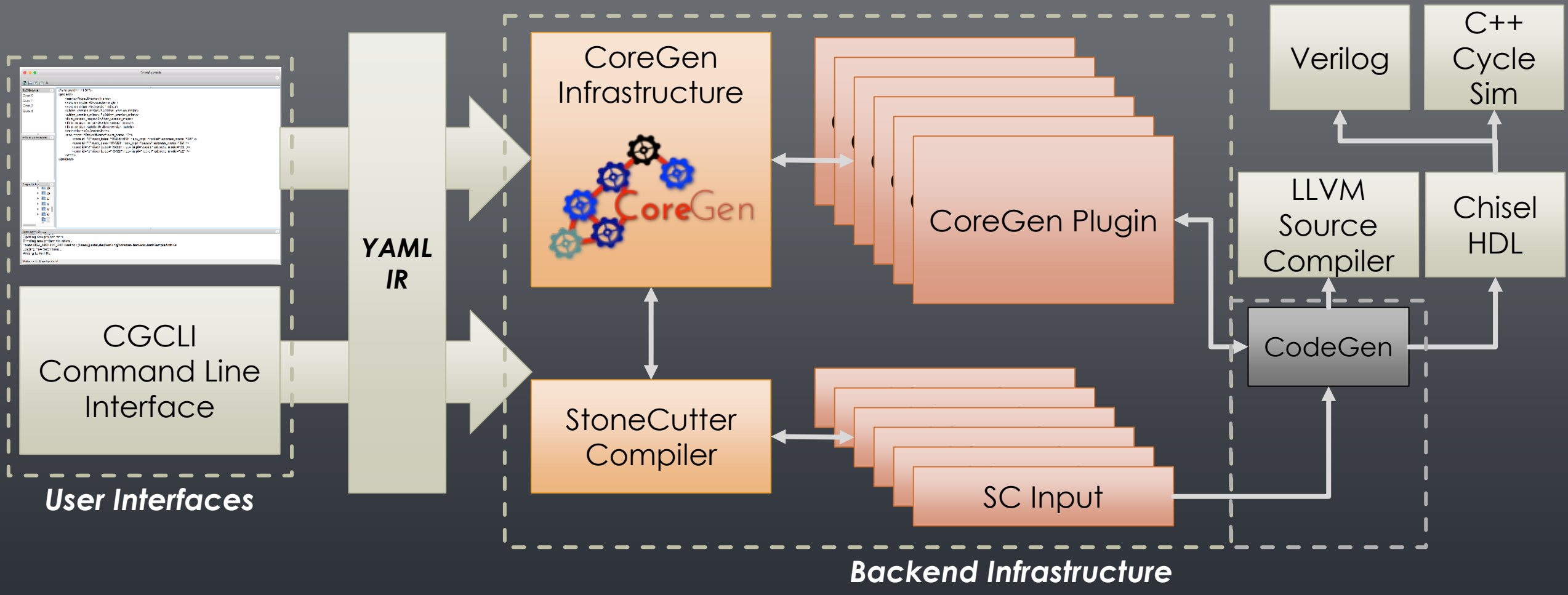
WHAT IS SYSTEM ARCHITECT?

- A family of tools, APIs and associated infrastructure to permit users to rapidly develop multi-faceted hardware
- Utilizes a combination of modular hardware design reuse principles, object oriented development and dependence analysis techniques (compiler theory) to provide an infrastructure for:
  - **Design & Design Experimentation**
  - **High Level Verification**
- The artifacts generated by a System Architect design flow include:
  - **Chisel HDL and Verilog RTL**
  - **C++ cycle-based simulator**
  - **LLVM compiler**

WHAT IS SYSTEM ARCHITECT **NOT**?

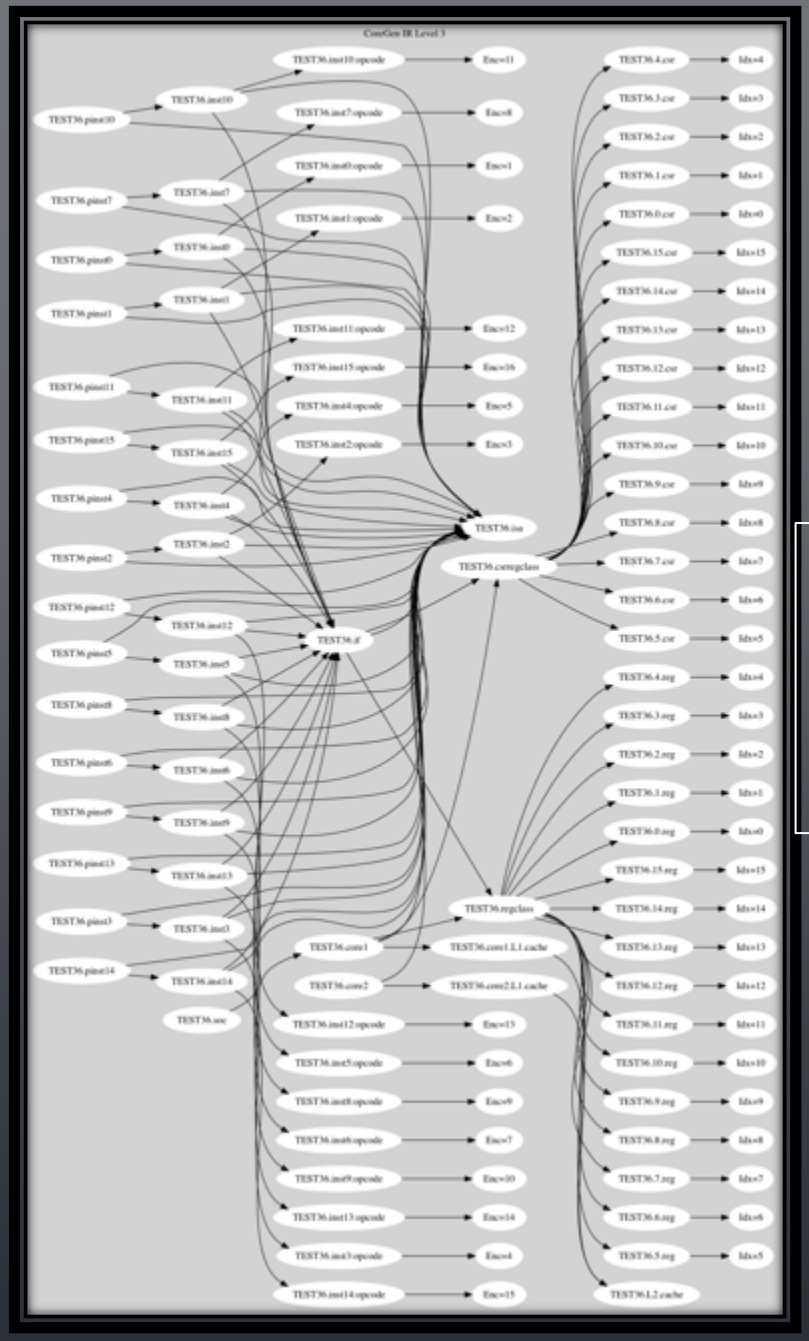
- System Architect is not the latest C-to-gates tool
  - It permits rapid design, verification and reuse
  - It does not auto-generate hardware based upon application code
- System Architect still relies upon the user to utilize reasonable design concepts
  - System Architect will **NOT** auto-generate optimized designs based upon unreasonable inputs
  - Users need to have a concept of the physical platform (FPGA, ASIC, etc)
- System Architect does **NOT** currently have a notion of physical layout
  - The generated output will not include LUT counts, physical design dimensions or power estimates
  - External FPGA/ASIC tools are required for this level of detail

GEN2 HIGH LEVEL DESIGN



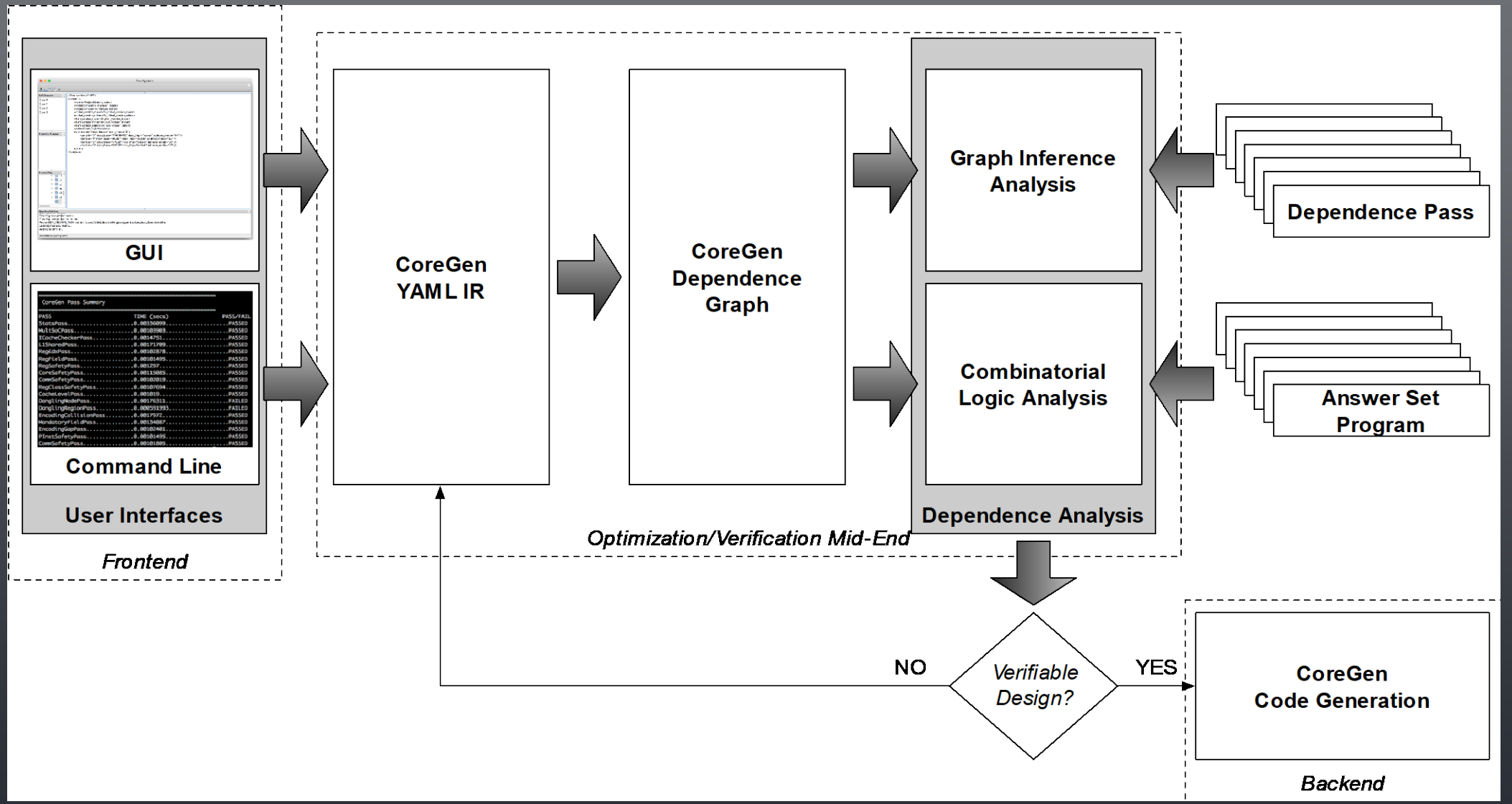
HOW DOES IT WORK?

- We input designs in the user-facing tools and generate **CoreGen Intermediate Representation (IR)**
- The IR preserves the natural dependencies within the design
  - Register classes depend upon registers
  - Instruction formats depend upon register classes
  - Instruction sets depend upon instructions
  - Cores require instruction sets
- We execute high level verification "passes" against the IR
  - Similar in design to traditional compiler passes
  - Walks the IR dependence graph and derives properties of the design
  - Reports issues in the design infrastructure, outputs interesting data or optimizes the design infrastructure
- Users implement instructions in StoneCutter instruction implementation language
  - C-like integrated with CoreGen IR to define a <u>single</u> instruction
  - Optimized by a traditional compiler flow to generate Chisel HDL for a single instruction
- Following the high level verification phase, we execute generate downstream code (code generation)
  - Generates Chisel HDL
    - Compiled down to Verilog & C++ cycle-based simulator
  - Generates LLVM compiler for the target design



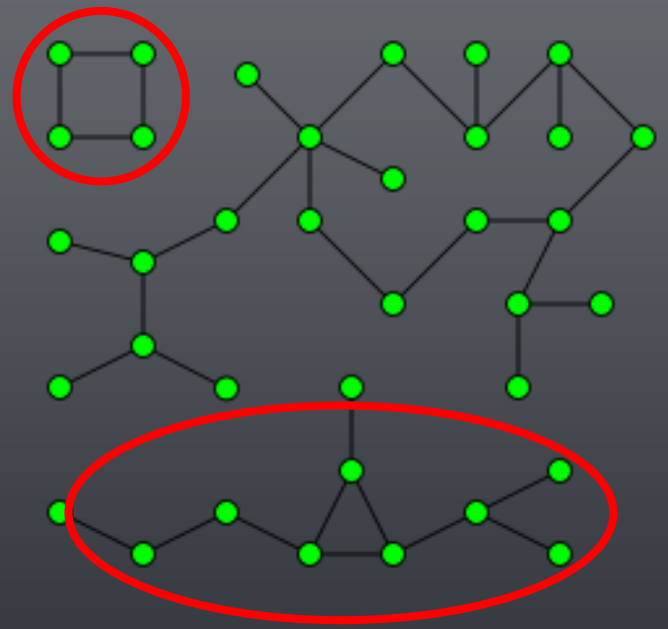
Example dependence graph from CoreGen design input

COREGEN VERIFICATION INFRASTRUCTURE



System Architect Gen2: Rapid Design Generation and Analysis

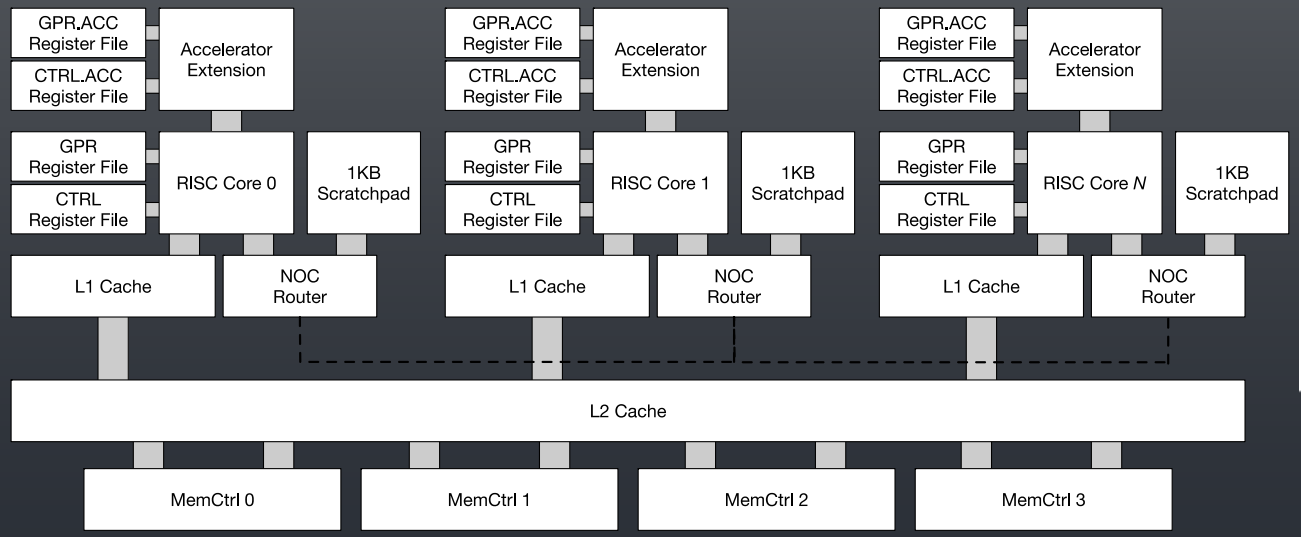
- Graph-Based dependence Analysis for design verification
- Hardware modules are defined as nodes in the DAG inside CoreGen
  - Dependence graph processed in multiple stages to expose connectivity, communication links, instruction and register encodings
- Multiple passes performed on the DAG for design verification
  - Elimination of dangling nodes / regions
    - Early detection to avoid issues with later synthesis of the design
  - ISA Encoding Collision Check
    - Early detection with decode logic and assembly/disassembly
  - Combinatorial logic analysis
  - User-defined passes coming soon

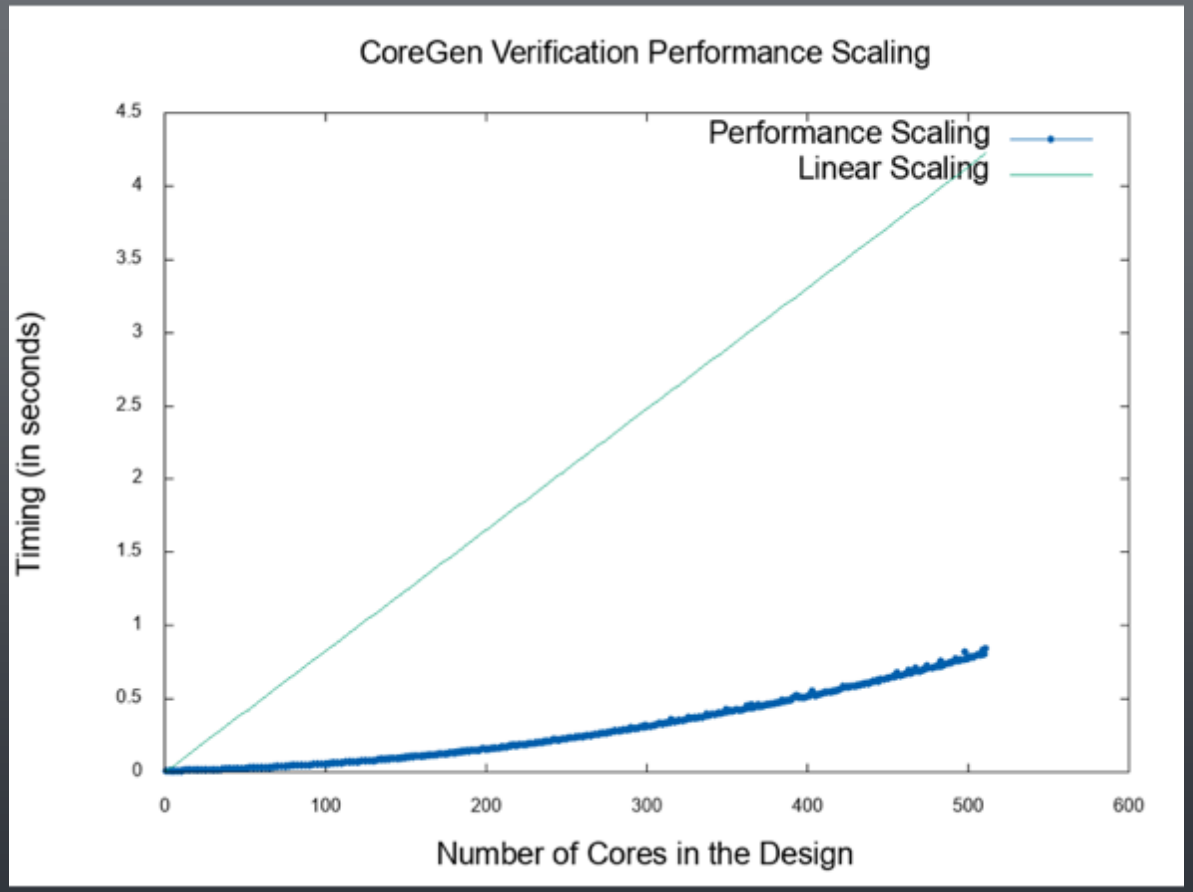


https://en.wikipedia.org/wiki/Component_(graph_theory)

SAMPLE PERFORMANCE

- HETEROGENEOUS, TILED DESIGN WITH MULTI-LEVEL CACHES AND NOC
  - MULTIPLE ISA'S WITH RISC CORES AND ATTACHED ACCELERATORS
- DESIGN SCALED FROM 1 TO 511 TILES
- **FULL VERIFICATION RUN OF 511 CORES < 1 SEC**





WEB LINKS

- System Architect Public Web
  - http://www.systemarchitect.tech/
- Documentation
  - Latest IR Specification:
    - http://www.systemarchitect.tech/index.php/coregenirspec/
- Tutorials
  - http://www.systemarchitect.tech/index.php/tutorials/
  - https://github.com/opensocsysarch/CoreGenTutorials

SOURCE CODE

- Main source code hosted on Github:
  - https://github.com/opensocsysarch

- CoreGen Infrastructure
  - https://github.com/opensocsysarch/CoreGen
- CoreGenPortal GUI
  - https://github.com/opensocsysarch/CoreGenPortal
- CoreGen IR Spec
  - https://github.com/opensocsysarch/CoreGenIRSpec
- System Architect Weekly Development Releases
  - https://github.com/opensocsysarch/SystemArchitectRelease

Live System Architect Walk Through

- *Live demo / instruction time*
- Goal: Create your own RISC-V core using OpenSoC System Architect
- Generate core then review and inspect:
  - Graph output
  - Power estimations
  - Hardware, compiler and other generated outputs

Live System Architect Walk Through

- *Live demo / instruction time*
- Extend the RISC-V RV32I instruction by adding a custom instruction definition
  - Example definition provided, or make your own
- Re-generate core and compiler
  - Inspect graph output and note new instructions
  - Inspect and compare power output
  - Inspect and modify pipeline options

Summary

- Specialization is key to continued performance gains
  - Calls for nimble, accessible hardware co-design flows
- Open-source flows have matured to enable custom designs
- Fabrication costs have dropped to $0 for open designs
- OpenSoC System Architect enables next-generation co-design flows
  - Provides the necessary hardware *and* compiler generation infrastructure

David Donofrio, Chief Hardware Architect

John D. Leidel, Chief Scientist

{ddonofrio,jleidel}@tactcomplabs.com