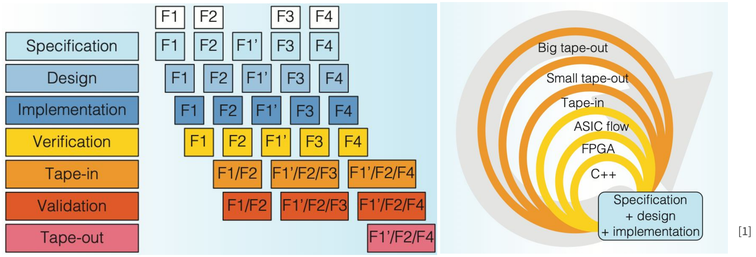
# Building a Hybrid Simulation/Emulation Platform for Fast, Flexible, and Accurate Hardware Evaluation

Raghav Gupta, Abraham Gonzalez, Sagar Karandikar, Borivoje Nikolic

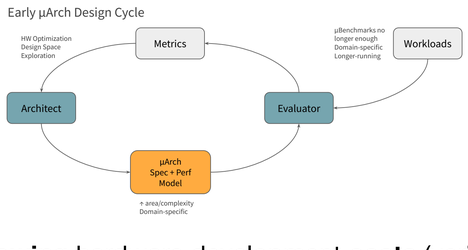
## Motivation

- **Hardware design** is becoming increasingly **agile and iterative**

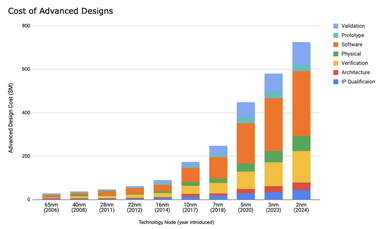



[1]

- Slowdown in technology scaling → greater emphasis on architecture exploration
- Advent of domain-specific hardware and software → evaluation of designs with **greater complexity** on **more/longer workloads**




- Rapidly **growing** hardware development **costs** (verification, validation, software development)




[2]

What's common? The need for **fast, flexible, and accurate evaluation** tools

## Limitations of Existing Evaluation Tools

| Tool | Examples | Throughput | Accuracy | Iteration Time | Early Design? |
|---|---|---|---|---|---|
| ISA Simulators | Spike, Dromajo | 10-100+ MIPS | None | Seconds | Yes |
| SW RTL Simulators | Verilator, VCS | 1-10 KIPS | Cycle-exact | Minutes | No |
| Academic uArch Simulators | gem5, Sniper | 100 KIPS - 100 MIPS | 10-50% IPC Error | Minutes | Yes |
| Industrial uArch Simulators | | 100 KIPS - 1 MIPS | Close | Minutes | Yes |
| FPGA Emulators | FireSim | 10-50 MIPS | Cycle-exact | Hours | No |
| Hybrid Simulation/ Emulation | | 1+ MIPS | Close(r) | Minutes | Yes |

Source: Adapted from Vighnesh Iyer's SLICE Retreat Talks

## Approach

Recognizing:

- The hardware design process is increasingly **iterative** and **baseline RTL** is often **available**
- A µArchitect typically works on **one portion** of a design **at a time**

We aim to build a platform:

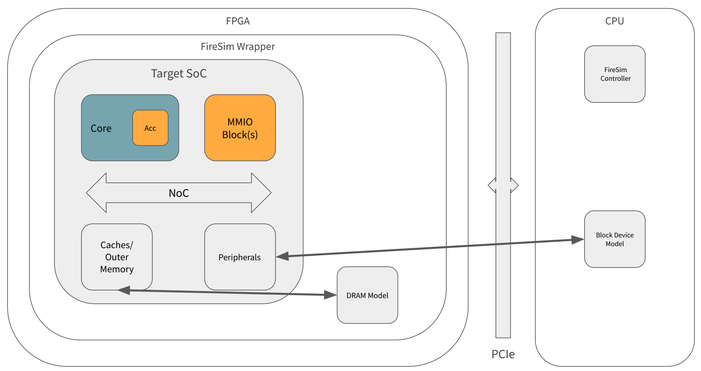
- that supports fast, flexible, and accurate full-system evaluation
- on real long-running workloads
- by allowing designers to combine evaluators operating on varying levels of abstraction (functional models / performance models / RTL)

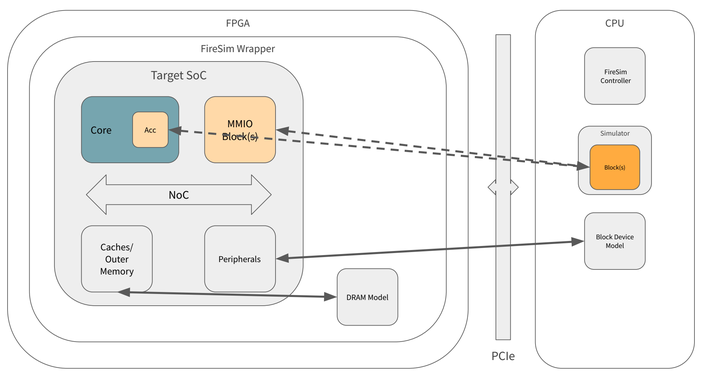
Design choices:

- **Leveraging existing** support for **FireSim Bridges** that allow for communication between the target SoC and the external world
- Emphasis on **latency-insensitive interfaces** for clean separation
- Starting with request-response blocks for simplicity

## Setup

Conventional FireSim Emulation



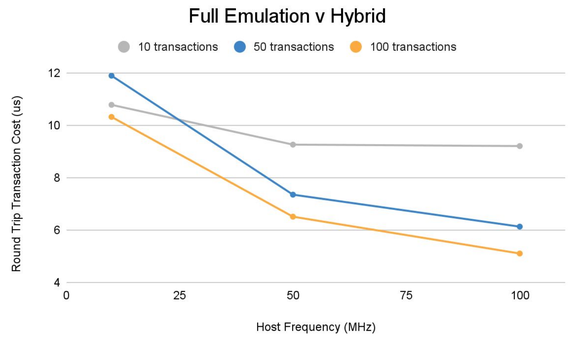

Hybrid Simulation/Emulation

## Proof-of-Concept Implementation

Full Emulation (Core + GCD RTL) v Hybrid (Core RTL + GCD C++ functional model)

Target SoC: SmallBOOM core + memory-mapped Greatest Common Divisor block

Experiment: Sweeping number of transactions and target emulation frequency to assess the round trip transaction cost




Initial Results

- average cost of **~7 µs per transaction** at 50 MHz
- **< 0.1%** FPGA **resource overhead** at 50 MHz on a VCU118 board

## Modeling Effective Frequency

- f = freq, t = time, $\alpha$ = activity factor = ratio of workload requiring FPGA-Host communication

- $f = 1/t$ ; $f_{eff} = 1/t_{eff}$ ; $f_{target} = 1/t_{target}$
- $t_{eff} = t_{target} + \alpha.t_{rtt}$ ; $f_{eff} = 1/(t_{target} + \alpha.t_{rtt})$

- e.g., for $f_{eff}$ = 1 MHz, $f_{target}$ = 50 MHz, & $t_{rtt}$ = 7 µs, we need $\alpha < 13.6\%$, i.e., FPGA-Host communication once per ~7 emulated cycles

## Going Forward

- Implementing time-delay insertion so the target sees simulated blocks cycle-exactly
- Profiling designs/workload combinations for activity factors

Improving evaluation throughput:

- Profiling setup for round trip cost breakdown (SW / PCIe / FPGA)
- Clocking off-target on-FPGA bridge stubs faster than the target design (currently one clock domain)
- Evaluating on FPGA boards with on-board hard cores
- Developing abstraction for synthesis of constrained SW models as RTL

## Acknowledgements

This project is a collaboration between SLICE Lab and AMD/Xilinx. We appreciate Alireza Kaviani and Pongstorn Maidee for their mentorship.