Blitz: A Static Timing Analyzer Parallelized Using Operator Formulation

Yi-Shan Lu[1], Rajit Manohar[2], Keshav Pingali[1]

[1]University of Texas at Austin

[2]Yale University

About Yi-Shan Lu

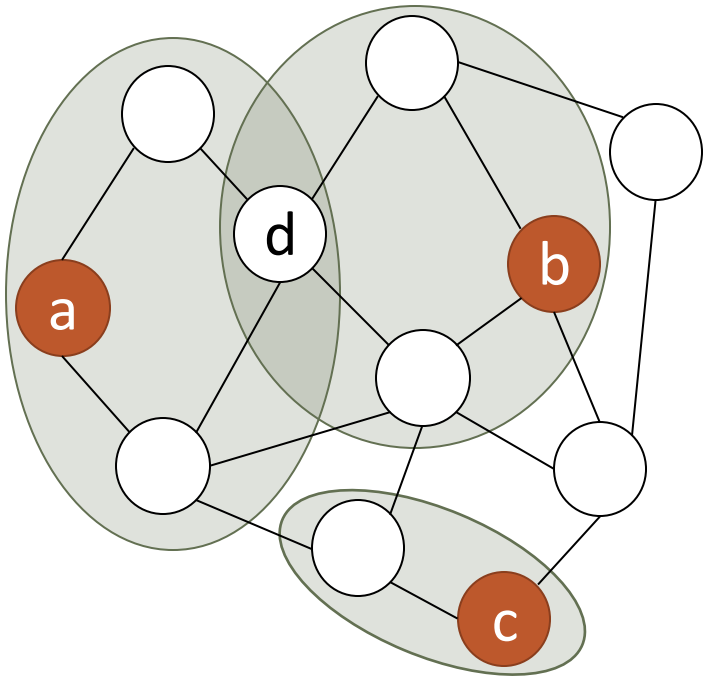
- PhD student
  - Advisor: Prof. Keshav Pingali
- Research interests
  - Parallelization & language design for domain-specific computation
  - Current focus: EDA algorithms, timing analysis & simulation
- Selected honors
  - Graph Challenge Champion, HPEC 2017
  - Third Place Award, TAU Contest 2019
  - Second Place, CADathlon at ICCAD 2019
  - Participation Award, TAU Contest 2020

Blitz: a parallel static timing analyzer

- Current capabilities
  - Timing propagation (full as well as incremental update)
  - NLDM delay & power calculation
  - Top-k critical paths
  - MCMM support
- Takes Verilog, SDC, Liberty, SPEF
- Accuracy verified using OpenTimer
- Parallelized using operator formulation
  - Enables shorter turn-around time
  - Candidate timer for timing-driven optimization

Operator formulation

- Data-centric programming model
  - Programmer specifies actions (updates) on graph
  - Actions performed concurrently if they touch disjoint regions of graph; otherwise sequenced
- Runtime system ensures conflicting actions are not executed concurrently
- Result: safe parallelism even for complex applications on unstructured data
  - Galois library, implemented in C++
  - https://github.com/IntelligentSoftwareSystems/Galois

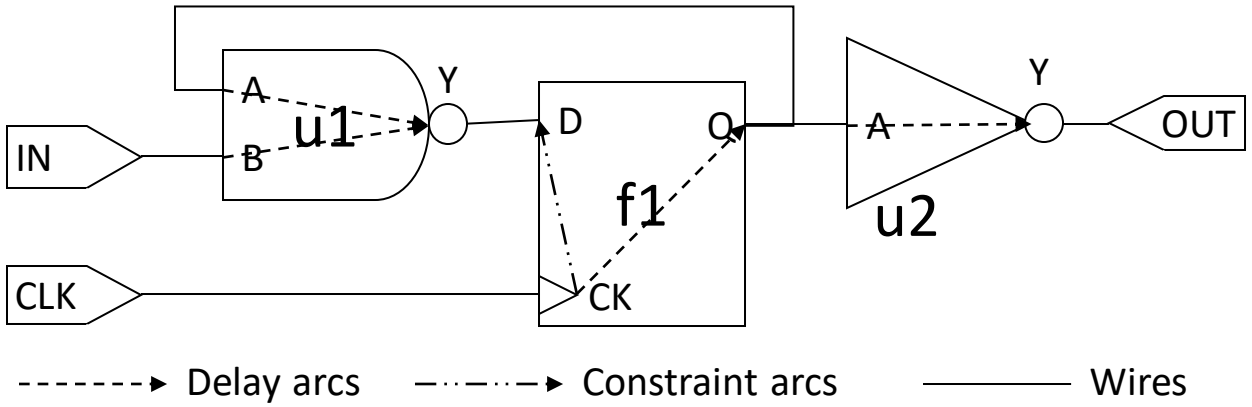


: neighborhood

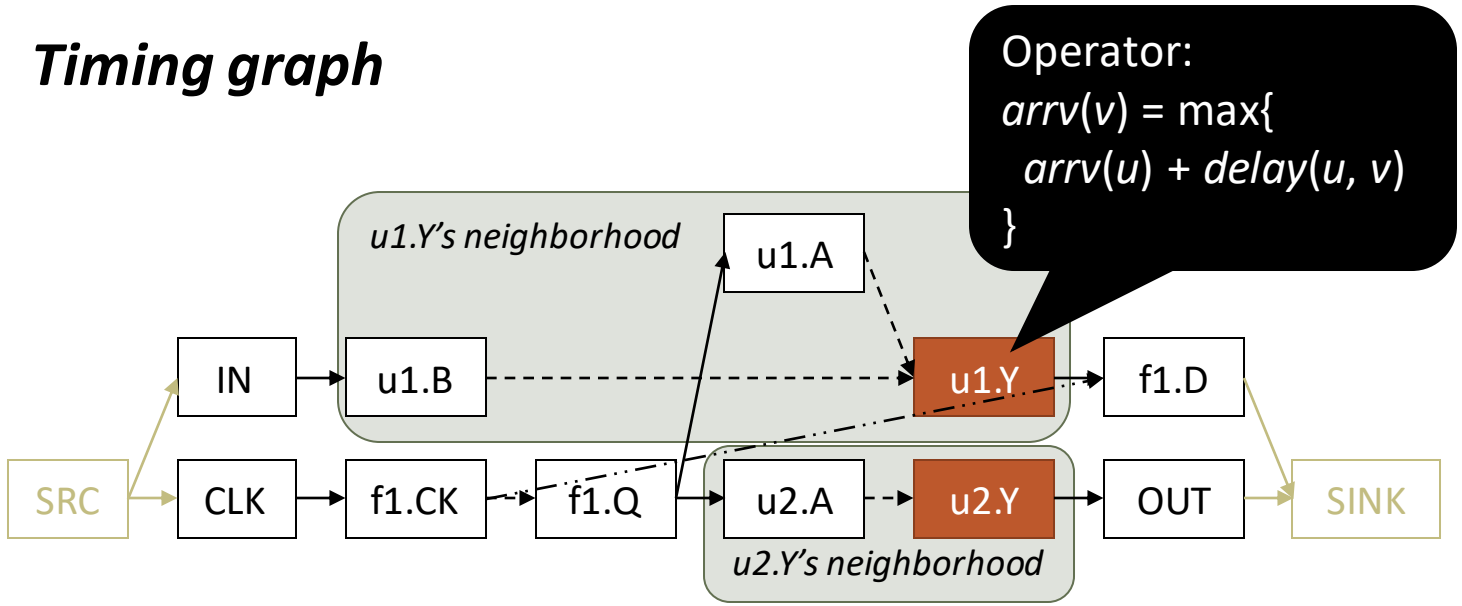
v : active node

Timing propagation in operator formulation

Netlist



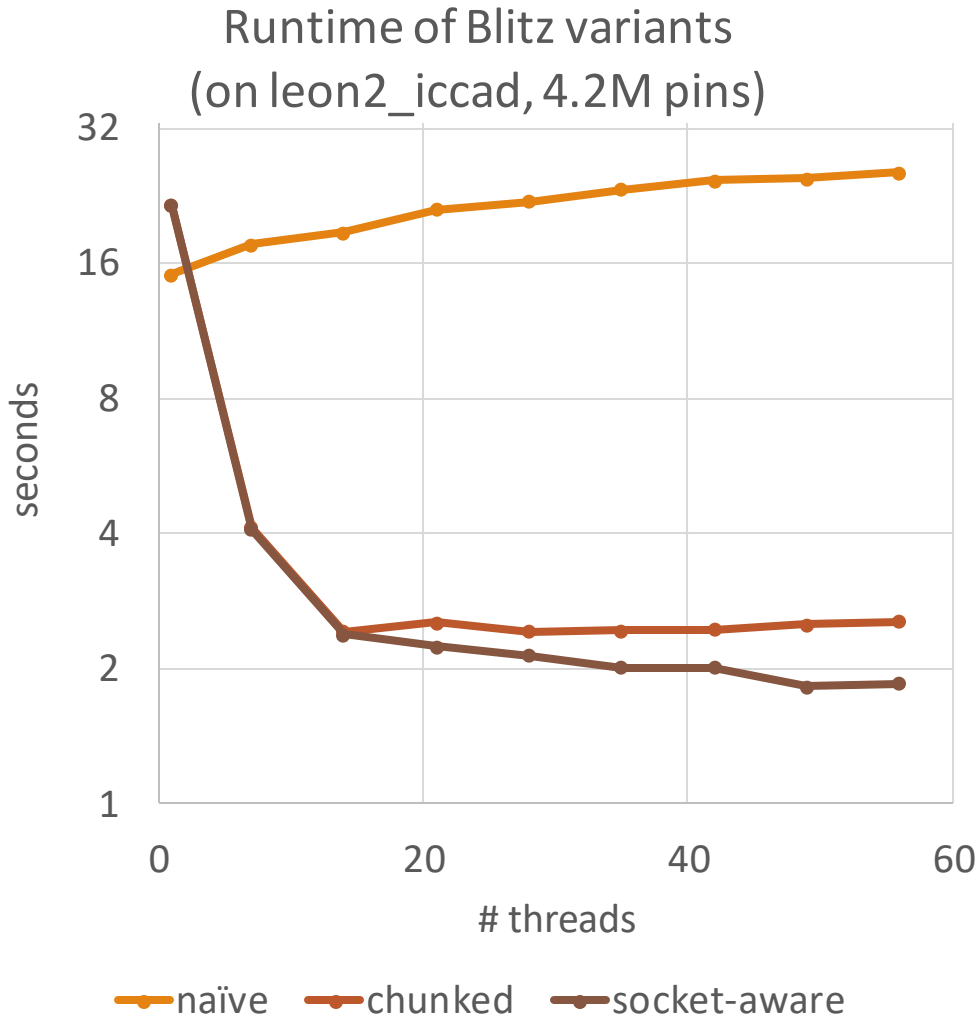
Timing graph



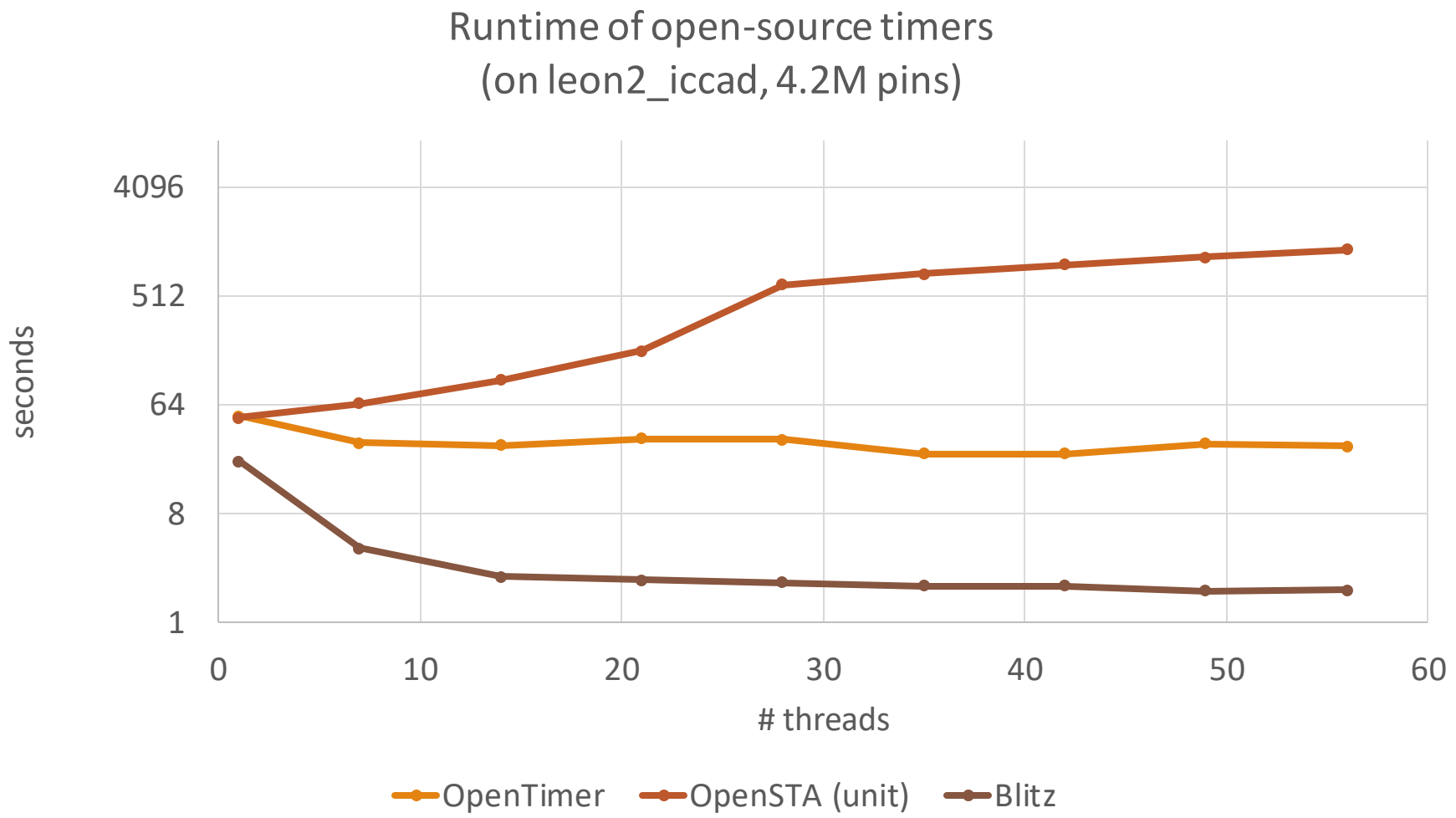
- Parallelism among active nodes
  - *arrv*(*v*) writes *v* and reads *u* where (*u*, *v*) exists
  - If *v* is active, no other nodes in *v*'s neighborhood can be active
- Nodes are activated in topological order from PIs
  - Tracked by a worklist

Impact of worklist design

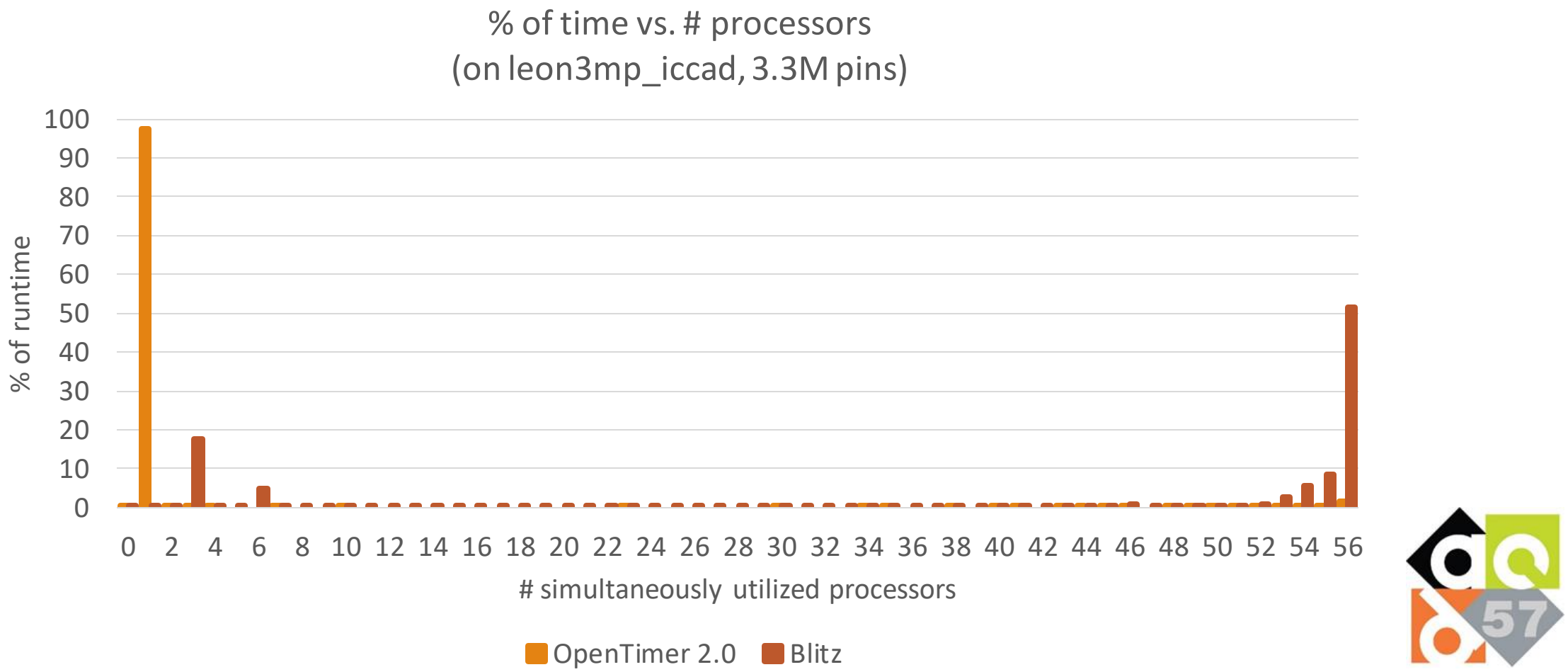
- Naïve
  - Global worklist of active nodes
  - Lock-guarded
- Chunked
  - Group active nodes into chunks
  - Threads push/pop chunks
  - Lock-free worklist
- Socket-aware
  - Everything in chunked
  - Threads interact with local socket
  - Socket leaders interact w/ remote sockets



Better performance over open-source timers



Effectively parallelized timing propagation



Future works

- Feature completion
  - General mechanism to support timing exceptions, e.g., false paths (tuning)
  - CCS delay calculation (tuning)
  - Integration with timing-driven optimization tools, e.g., placer/router (WIP)
- Better scalability
  - Performance scalability beyond one socket
  - Handling larger circuits (in distributed setting)
- Open-source release

Thanks!

Questions?