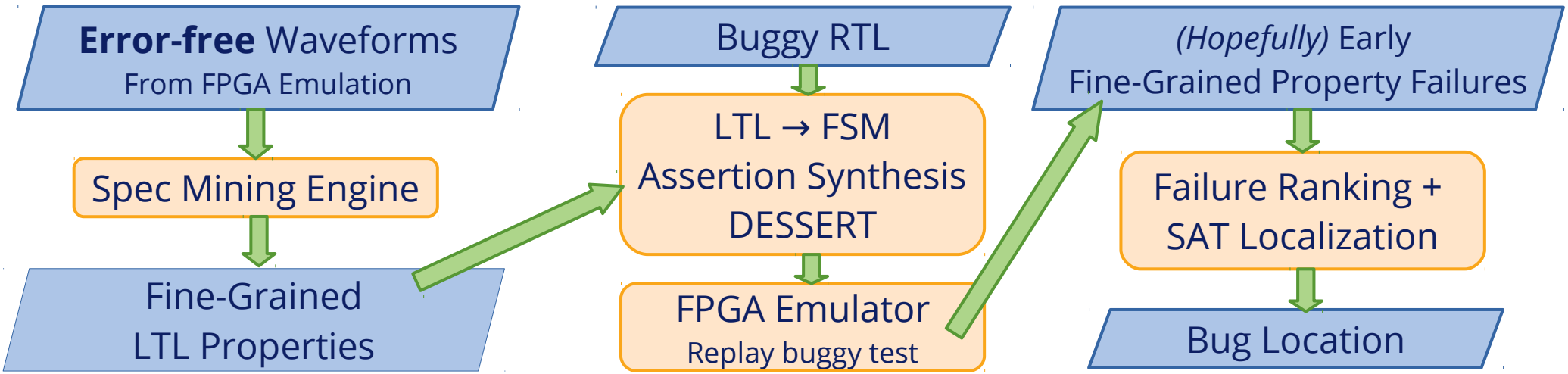
DRILLS: Debugging RTL Intelligently with Localization from Long Simulation

Specification mining using trace dumps derived from long-running FPGA emulation to localize bugs in complex digital designs

Donggyu Kim & Vighnesh Iyer

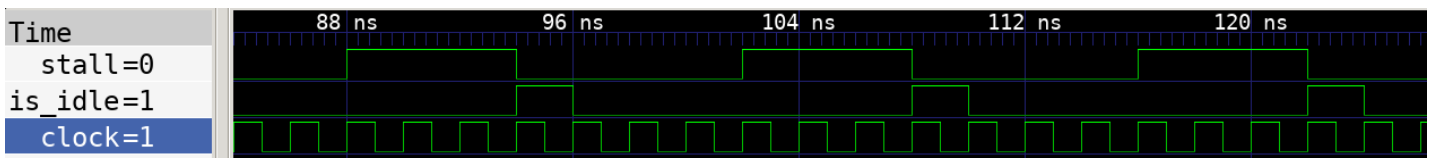
Plan

- **Problem**: given many error-free traces and one error trace, localize the likely bug location by module or line of RTL and find the fine-grained mined properties which were violated.
- Specification mining [Li '10] is a technique to extract fine-grained (Σ = 2,3) LTL properties (based on a set of templates) from execution traces
  - We will add additional LTL templates, and will apply this technique to large designs



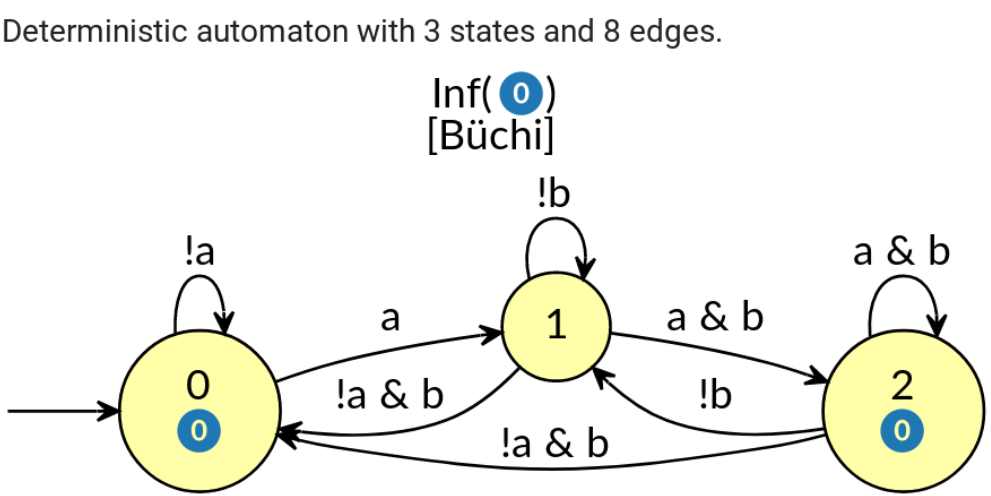
Current Status + TODO

- Working VCD parser and LTL property miner (for 2 property templates)
  - Alternating: Δa **A** Δb
  - Until: **G**(Δa → **X** (a **U** Δb))
  - Next: **G**(Δa → **X** Δb)
  - Eventual: **G**(Δa → **X F** Δb)
- Capable of mining properties from chisel-examples and riscv-mini
  - Next: [['TOP.Tile.core.dpath.csr_io_stall', 'TOP.Tile.core.dpath.stall', 'TOP.Tile.core.dpath.csr.io_stall'], ['TOP.Tile.icache.is_idle']]



Questions

- Is there a systematic way to go from LTL property to ‘monitor automaton’?
  - Right now, the mining functions are very explicit and need hand-constructed automaton
  - Spot can emit automata for an LTL property, but it needs to be modified for finite-length traces



- Example for G(a → XF b)
- State-based acceptance only works for infinite-length traces that visit accepting states infinitely often
- We need to augment the automata with a ‘num_patterns_seen` var
- Q: how to prune mined LTL properties?