2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
FEBRUARY 27-MARCH 2, 2023

Using a modern build system to speed up complex hardware design

Varun Koyyalagunta, Tenstorrent

accellera SYSTEMS INITIATIVE

The hardware design cycle

1. Check out latest design and dependencies
2. Make a change
3. Build simulation models
4. Run smoke tests
5. Push that change to other users

The hardware design cycle - typical flow

1. git pull; git submodule update --recursive
2. $EDITOR file.sv
3. make test -j8
4. git push

The RTL design cycle - Pain points

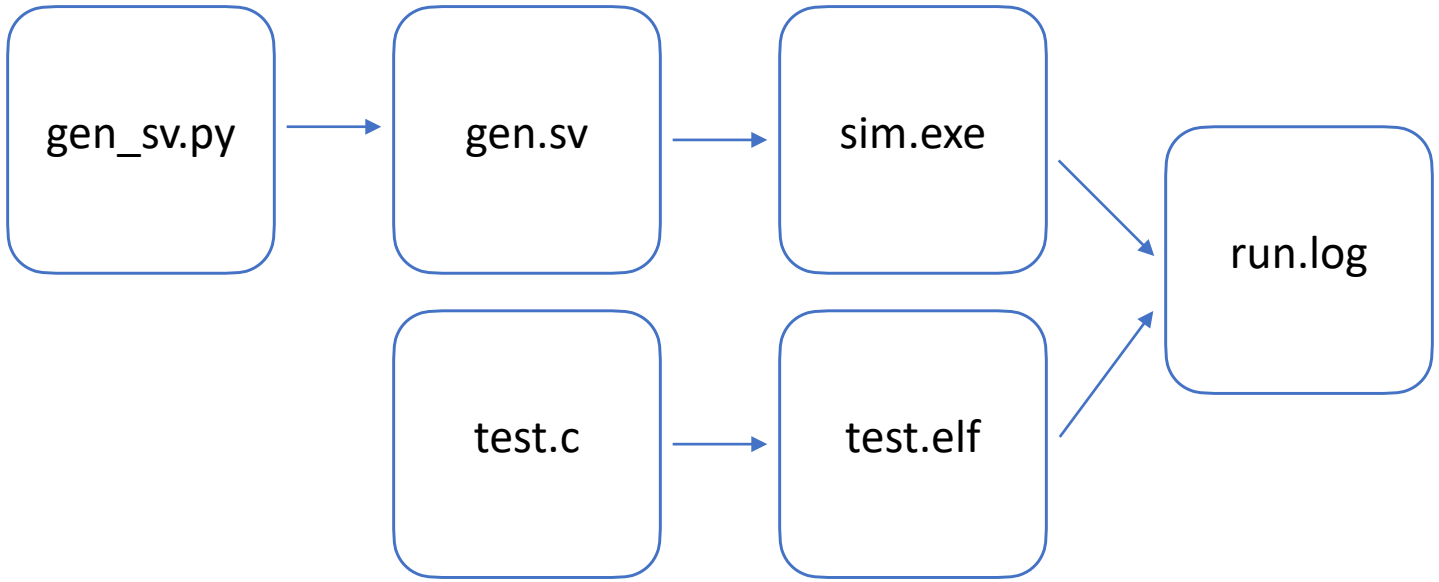
1. git pull && git submodule update --recursive
   - Dependencies can take a lot of space and time to download
2. $EDITOR file.sv
3. make test -j8
   - Builds all simulation models and runs all tests even if your file.sv isn't used by all units
   - Does not fully leverage available compute resources
   - May have to do "make clean" often because of badly specified dependencies
4. git push

Solution - Bazel

- Fast
  - Better dependency management - Download as needed
  - Remote caching - Download results from other users' runs
  - Remote execution - Dispatch builds and runs to a compute farm
- Correct
  - Sandboxed execution - Difficult to have unspecified dependencies, removes need for "make clean"

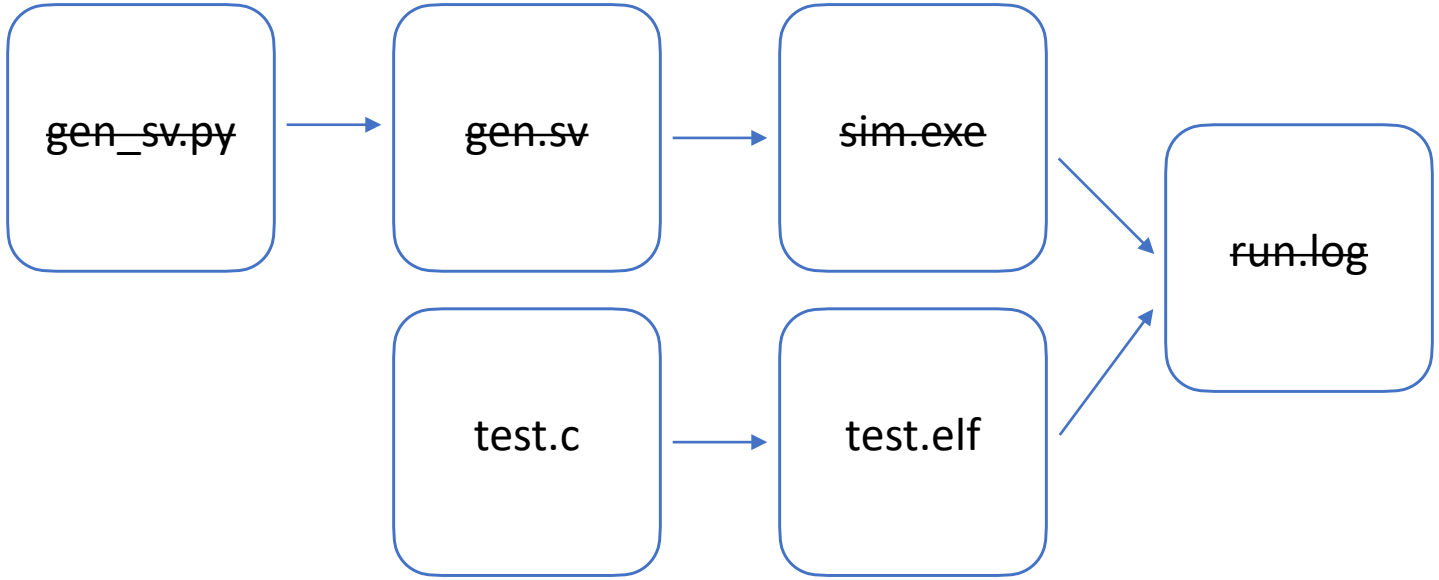
Bazel

- User describes a dependency graph
- Bazel quickly and correctly optimizes each node in the graph



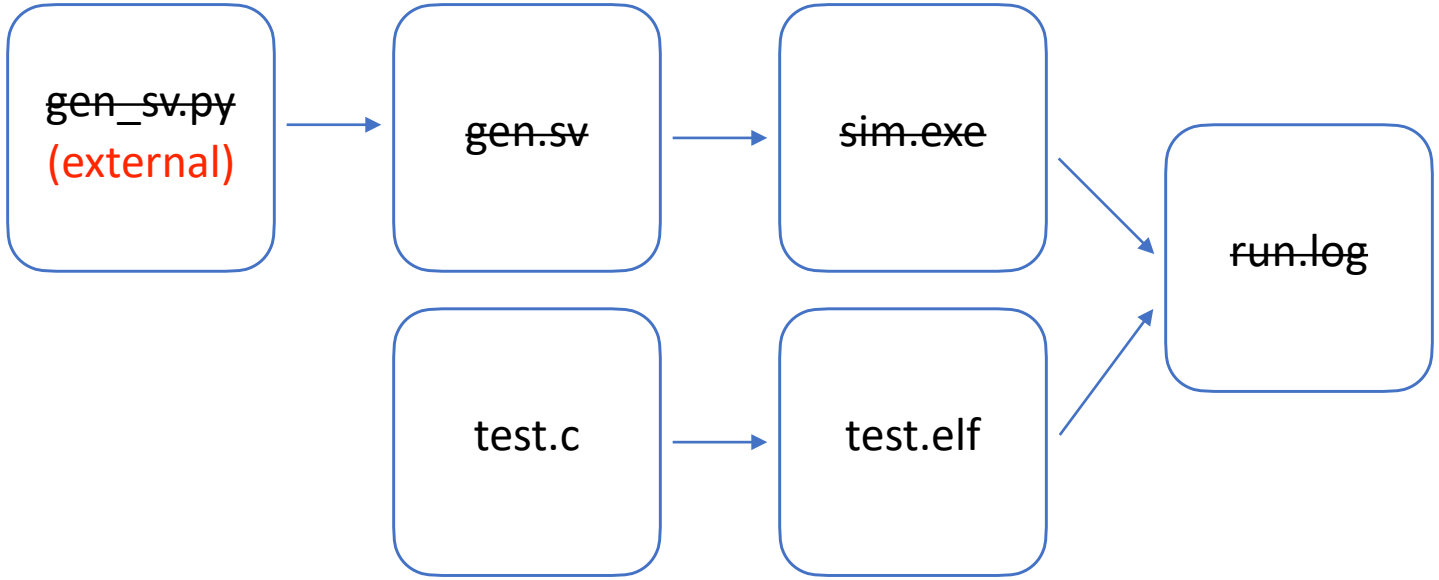
Sandboxed execution

- Each node of the dependency graph only sees specified dependencies
- Eg, the step to create test.elf can see test.c but not gen_sv.py



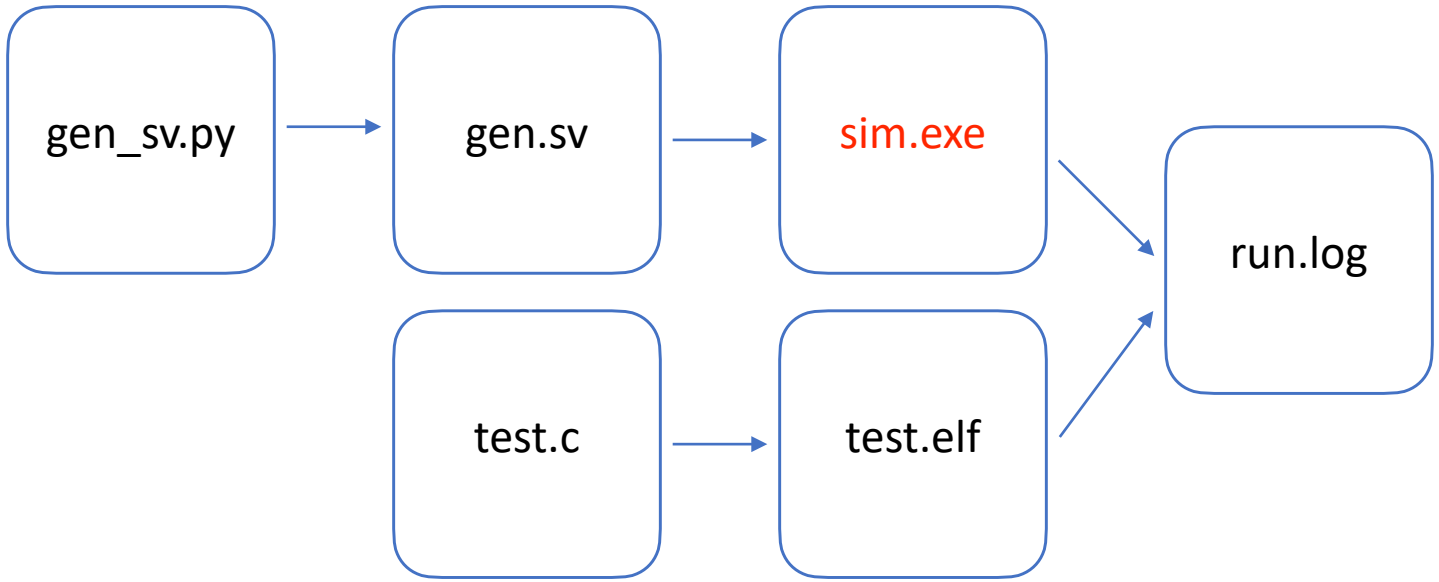
Dependency management

- Assuming gen_sv.py is coming from another repo
- If only generating test.elf, that repo does not need to be downloaded



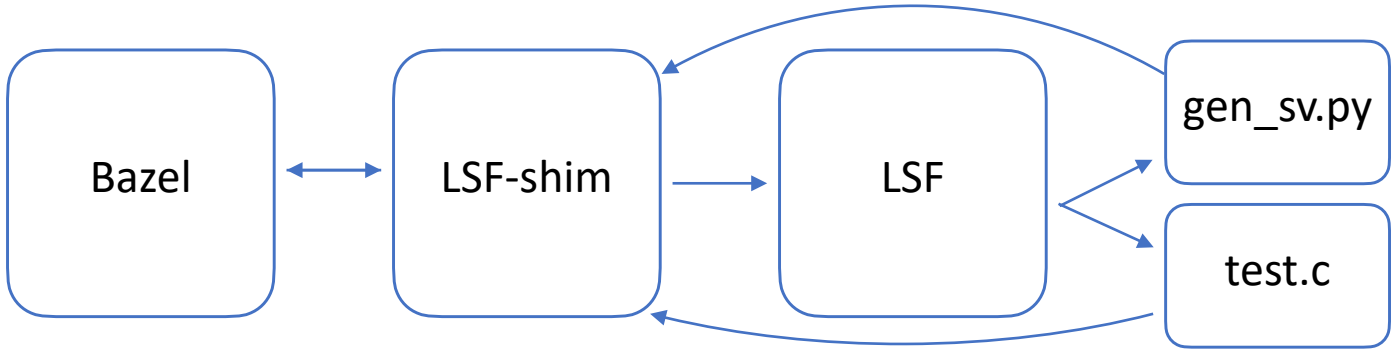
Remote caching

- Anything someone else has done can be downloaded



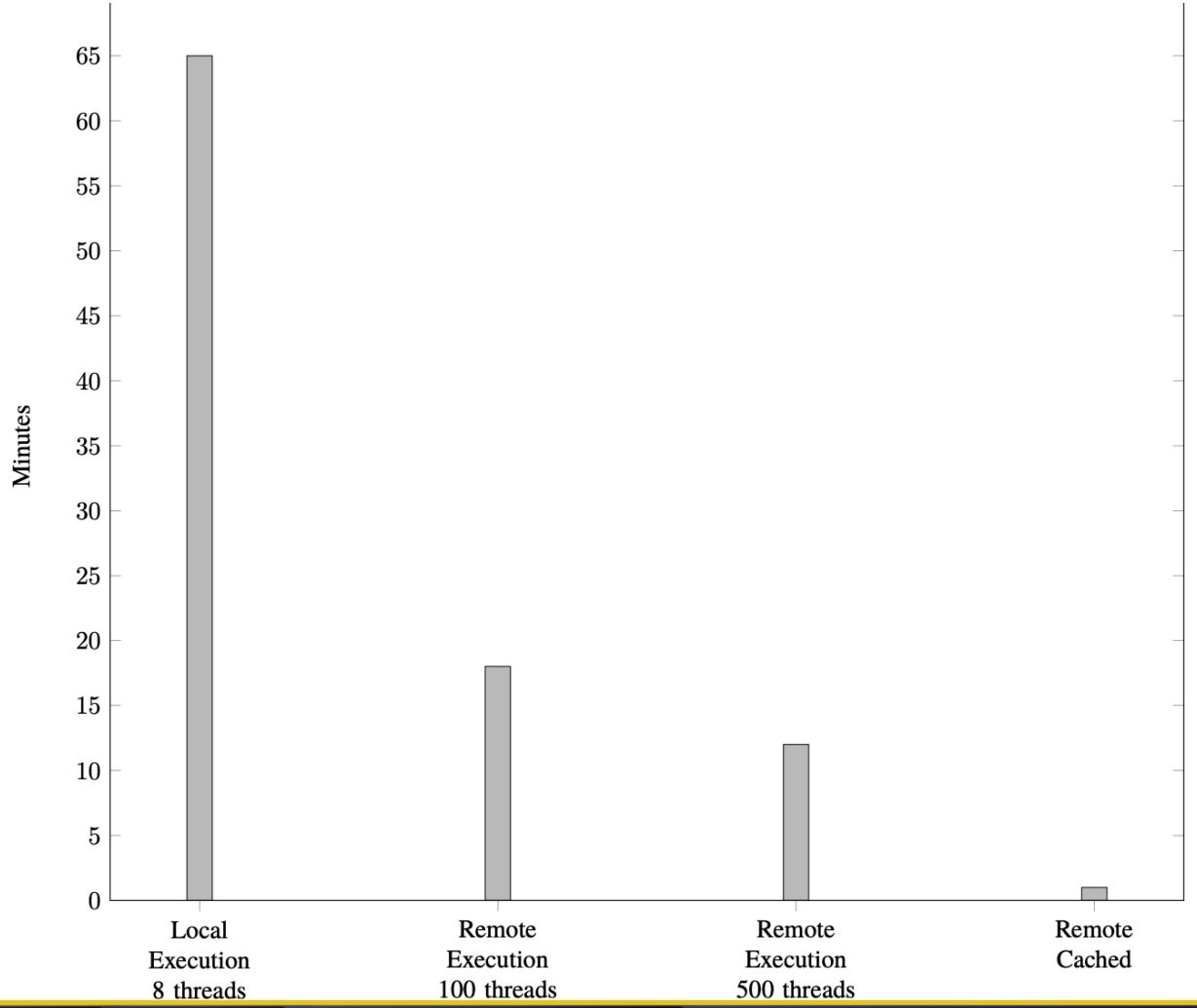
Remote Execution

- Bazel uses an open protocol for remote execution
- There are many free open source implementations and paid commercial implementations
- For legacy schedulers that don't support the protocol, one can write a shim that translates Bazel's remote execution requests to their API



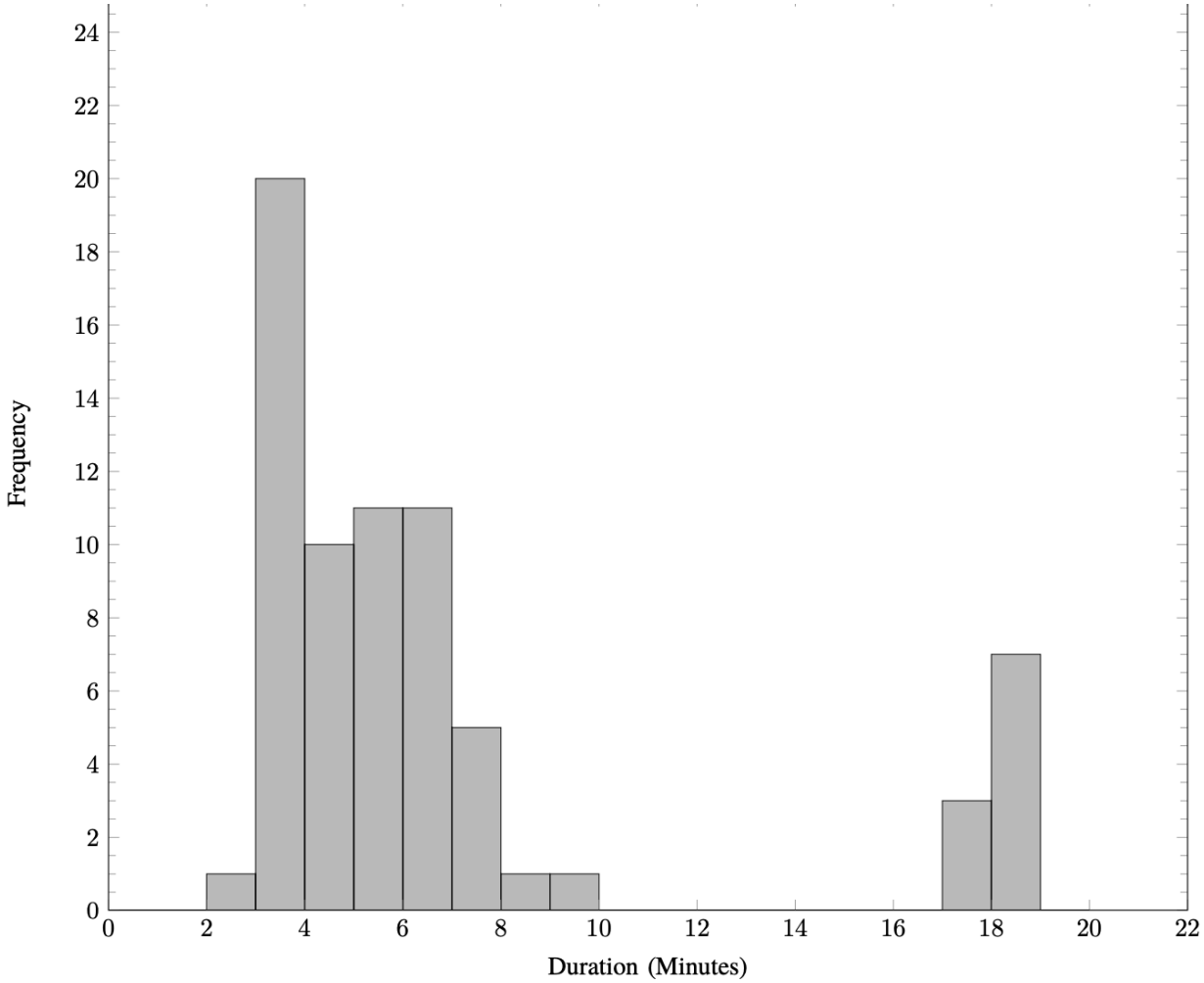
Average smoke duration

- Reduced more than 3x by leveraging compute farm
- Caching can reduce even further



Smoke duration distribution for one day's worth of runs

- 71 total runs
- Average duration with no cache hit is 18 minutes
- More than 13 hours of CPU time saved per day due to caching



Questions?