A Low Maintenance Infrastructure to Jumpstart CPU Regression and Performance Correlation

Thomas Soong, Christopher Browne, Chenhui Huang
Intel Corporation

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES
VIRTUAL
FEBRUARY 28 - MARCH 3, 2022

Surge's Key Innovations

Surge is utilized on Intel E-core validation to achieve the following:

❑Jumpstart CPU Core simulation and emulation to any architectural boundary
❑Achieve 40% simulation speedup by jumpstarting into critical test section
❑Resilient to design changes due to its small code base and reuse of processor ISA features
❑Reliably conduct performance correlation from silicon captured traces

Surge's Jumpstart Ingredients

The following components need to be properly initialized for successful RTL jumpstart:

❑ Uncore BFM transactions
❑ Core Architectural and uArch States
❑ Golden Reference Model

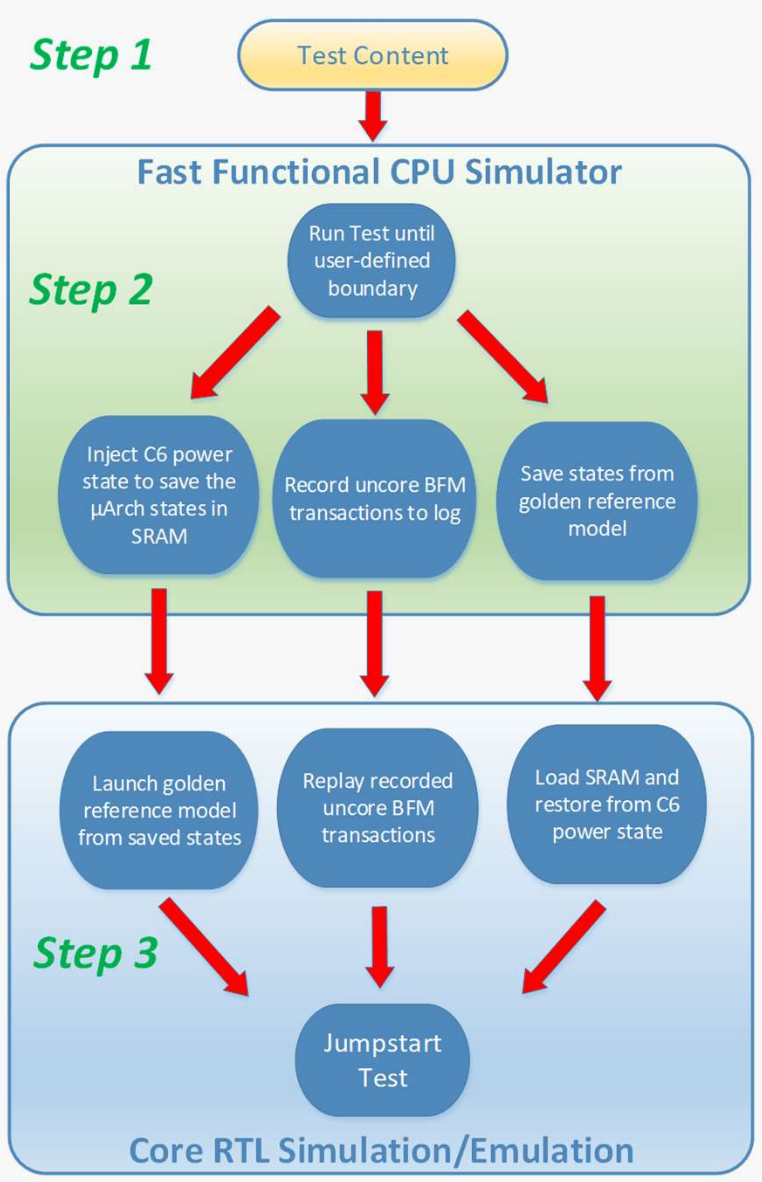
Surge's Key Reuse Components

❑***FastSim***: E-core's fast functional high-level model of its CPU design. Used for testing uArch features and prototyping
❑***Archsim***:Golden ISA reference simulator. Model's x86.
❑***C6 Power State***: Native ISA feature where the processor state is saved into a dedicated SRAM before voltage to core is reduced to zero. When exiting C6, the processor state is restored back from the SRAM.

More In Paper

❑Jumpstart with silicon capture traces
❑Performance correlation in simulation and emulation
❑Surge's trade off
❑Surge's overall maintenance cost during E-core development

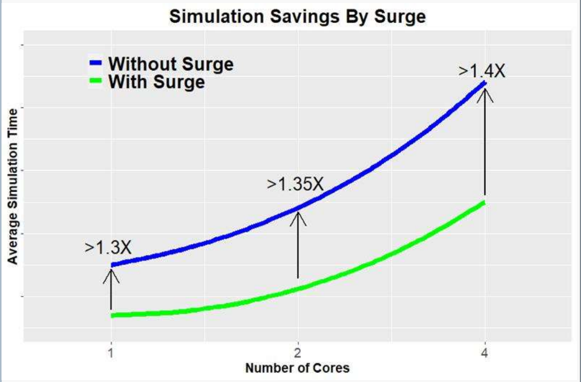
Surge's Core Simulation Jumpstart Workflow


Step 1
Test Content
Fast Functional CPU Simulator
Step 2
Run Test until user-defined boundary
Inject C6 power state to save the µArch states in SRAM
Record uncore BFM transactions to log
Save states from golden reference model
Launch golden reference model from saved states
Replay recorded uncore BFM transactions
Load SRAM and restore from C6 power state
Step 3
Jumpstart Test
Core RTL Simulation/Emulation

Workflow Breakdown

All test content are first run quickly under FastSim and then seamlessly migrated to RTL simulation
❑Step1: The test content is run until the user-defined boundary point.
❑Step2: Test run in FastSim
a) FastSim injects C6 power state entry command and executes the C6 save flow. FastSim exports the C6 SRAM contents to file.
b) The uncore BFM produce an external log containing all its transaction request.
c) Archsim's save routine snapshot its architectural image. This will allow Surge to keep RTL and Archsim in sync after CPU jumpstart is completed.
❑Step3: In RTL simulation
a) SVTB code to inject the saved C6 data into RTL's physical C6 SRAM. Subsequently, an injection triggers a C6 restore which loads all the architectural and microarchitectural core states back into the CPU simulation.
b) the CPU simulation's uncore BFM states is restored by replaying uncore request log
c) Archsim state is restored from the saved image.
❑Core simulation/emulation will continue executing from where FastSim left off.

Performance Result


Simulation Savings By Surge
Without Surge
With Surge
>1.4X
>1.35X
>1.3X
Average Simulation Time
Number of Cores