DVCon 2013
Design & Verification Conference & Exhibition

February 25-28, 2013
DoubleTree, San Jose

accellera SYSTEMS INITIATIVE

Can You Even Debug a 200M+ Gate Design?

Horace Chan
PMC-Sierra

Brian Vandegriend
PMC-Sierra

Deepali Joshi
PMC-Sierra

Corey Goss
Cadence

PMC

cadence™

Sponsored By:

Overview:

- Debug techniques for a 200M+ gate design
- Testbench built in Specman UVM*e*
  - *Simulator: Cadence IES*

- Speed up debug turnaround time for RTL
  1. Saving and Loading Checkpoint Snapshots
  2. Dynamic Load Additional Code at Saved Checkpoints
  3. Waveform Verbosity
- Speed up debug turnaround time for Testbench/Software
  4. Incremental compile of the Specman testbench
  5. Software/Hardware co-verification

Saving and Loading Checkpoint Snapshots

- Problem:
  - Long simulation time
  - Need to wait a long time to reproduce the failure
  - Probing waveform slow down the simulation even more

- Solution:
  - Save a checkpoint snapshot every 30 minutes
  - Save a checkpoint snapshot every 1ms of run time
  - Save a checkpoint after each key testflow phase
  - Rerun from the save checkpoint to reproduce the failure

Sponsored By:

Saving and Loading Checkpoint Snapshots (Implementation)

- Checkpoint period controlled by
  - Environment variables
  - Shell script arguments
  - vManager VSIF attributes
  - Specman constrains
- Optimal checkpoint period depends on the snapshot size and the load of the network
  - E.g. 30 seconds checkpoint saving overhead for a 4G snapshot during work hours

Sponsored By:

Dynamic Load Additional Code at Saved Checkpoints

- Unique features in Specman ***e***
- Load extra testbench code after loading a checkpoint
  - Change the constraint of a sequence
  - Modify the behavior of existing methods
  - Add and launch new sequence
- The testbench is built with empty method hooks between each key testflow phase for dynamic load extension

Dynamic Load Additional Code at Saved Checkpoints (use case)

- Add extra debug code to print more information
  - Change the sequence stimulus generation around the failure point to explore the bug
- Test different what-if scenarios without waiting for the simulation to run from time zero
  - Skip the 3 hours wait to configure the device and wait for the traffic to stabilize
- Group regression runs that share a common initial period to reduce CPU and license usage.

Sponsored By:

SYSTEMS INITIATIVE

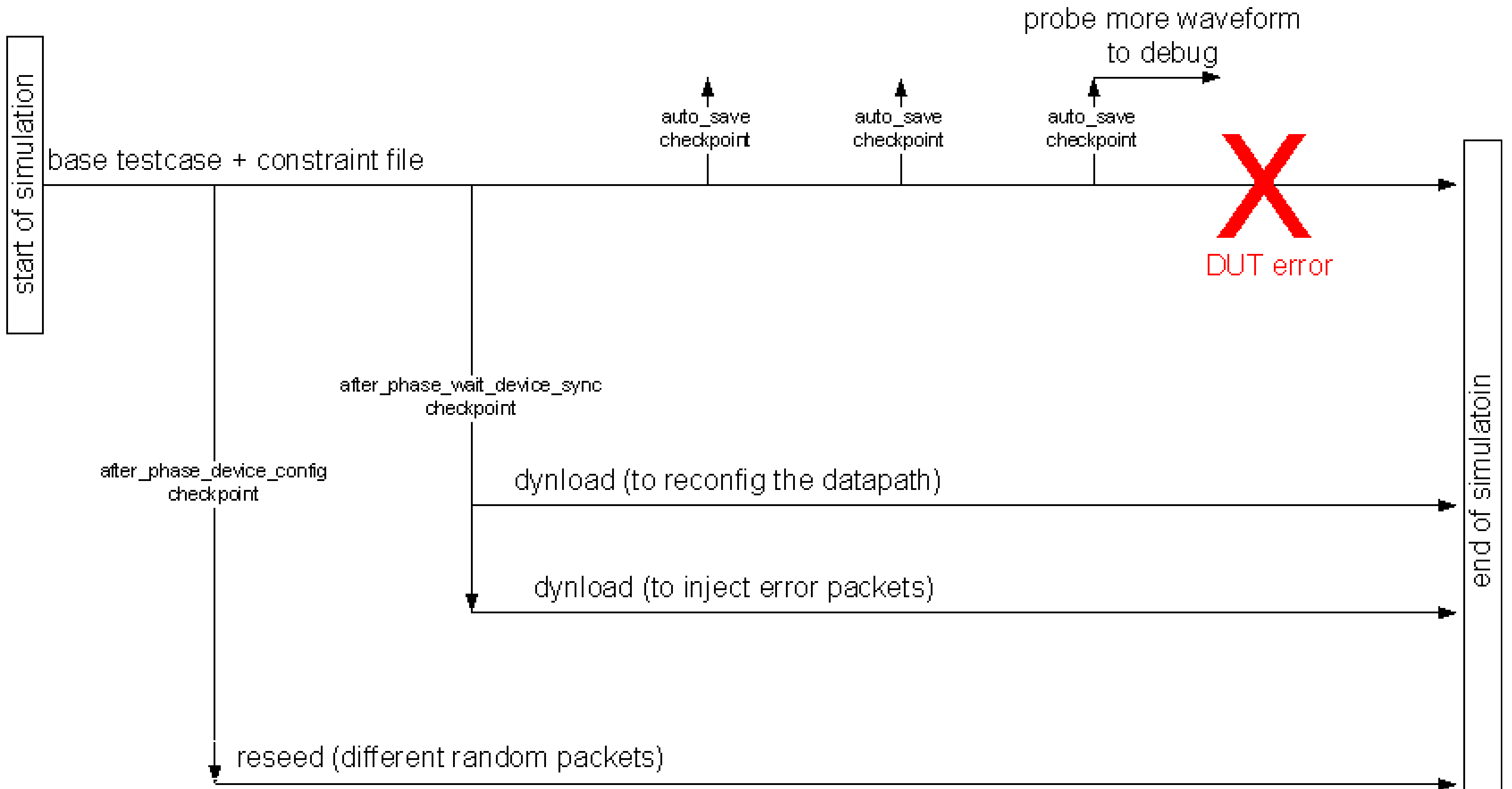
Waveform Verbosity

- Same idea as message verbosity
  - Less waveform probed, faster the simulation
  - More waveform probed, more debug information

- Easy control to set the waveform verbosity at run time
- Suggested verbosity level
  - **NONE:** No waveform is probed
  - **LOW:** Probe all the ports of the module
  - **MEDIUM:** Probe all the internal signals of the module
  - **HIGH:** Probe the memories and variables of the module
  - **FULL:** Probe delta cycle changes of the signals

Sponsored By:

Checkpoint Usage Example


probe more waveform to debug
start of simulation
base testcase + constraint file
auto_save checkpoint
auto_save checkpoint
auto_save checkpoint
X
DUT error
after_phase_wait_device_sync checkpoint
after_phase_device_config checkpoint
dynload (to reconfig the datapath)
dynload (to inject error packets)
reseed (different random packets)
end of simulatoin

Incremental compile of the Specman testbench

Sponsored By:

- Compiled testbench run 3x faster
- Interpreted testbench has better debug support
  - Reload the testbench without exit the simulator
    - Keep the CPU and license
  - Keep the current state of the simulation without rewinding back time zero
    - Loading initial snapshot takes a long time

- The best of both world
  - Compile stable testbench code
  - Interpret testbench code still under development

Software/Hardware co-verification

- Integrate software (C code) with the testbench (e code)
  - The C code is running on the Linux host as a thread inside the simulator
    - **1000x** faster than running inside the CPU RTL
    - **100x** faster than running with CPU model.
  - Replay the software calls without the simulator
    - Software debug turnaround time from 30 minutes to 30 seconds
  - Simvision GUI has source debugger for all languages in the testbench and DUT. (e, C, Verilog/VHDL/SV)
- Implementation details refer to:
  - Hardware/Software *Co-Verification* Using *Specman* and SystemC with TLM Ports, DVCON2011

Sponsored By:

BENEFITS AND RESULTS

- Speed up the RTL debug turnaround time by **90%**
  - Average time to failure in simulation is **6 hours**
  - Reduce overall development time by **50%**
- The compiled testbench is running **300%** faster
  - Specman CPU usage: **15%** down to **5%**
- Silicon bring up in the lab with already tested software
  - From **2-3 weeks** down to **1-2 days**

FUTURE DEVELOPMENT

Sponsored By:

- Port to SystemVerilog and other simulators
  1. Saving and Loading Checkpoint Snapshots
     - **Easy to port.** If the simulator supports saving checkpoint.
  2. Dynamic Load Additional Code at Saved Checkpoints
     - **Impossible to port.** Unique feature in Specman
  3. Waveform Verbosity
     - **Easy to port.** The code is implemented in tcl
  4. Incremental compile of the Specman testbench
     - **Hard to port.** Unless SV supports AOP
  5. Software/Hardware co-verification
     - **Easy to port.** SV supports DPI-C programming interface too.