Using Save/Restore is easy, Right?
A User's Perspective on Deploying Save/Restore

Ron Thurgood, Ed Powell, Aneesh Samudrala

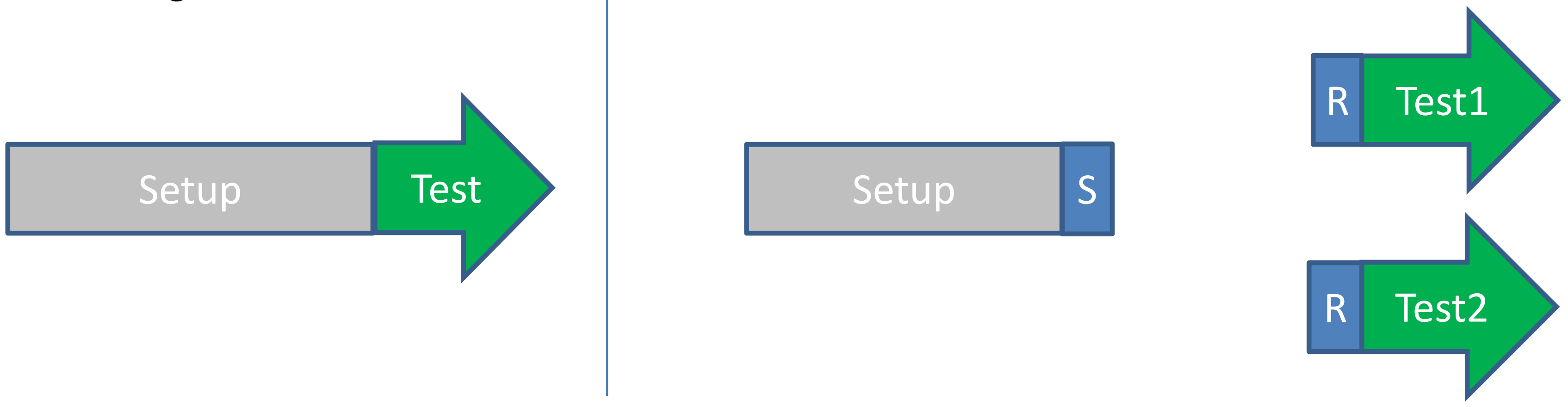
Introduction

- What is Save/Restore?
- Is the technology there to support a robust solution?
- Where do we see benefit using this technology?
- What are the technical and methodology challenges?
- What results have we realized with real projects?

Save Restore

Single Test

Save/Restore Testing



Setup

Test

Setup

S

R

Test1

R

Test2

- Save the state of a simulation, then restore that state and continue
  - Save/Restore, re-seed, and run test
  - Save/Restore, re-seed, change test code, and run test

Past Solutions

Technology History

- IEEE 1364-1995
  - Supported system tasks $save/ $restart
  - PLI interface reason_save, reason_startofsave
- IEEE 1364-2005
  - Replaced PLI support with VPI support

Limitations

- High resource cost for implementation
  - Enablement of the technology, and deploying a robust methodology
  - Complex solution that users had to manage
  - Difficultly of writing the save and restart callback routines.
- Support for other external code (C,C++,Specman-E, VHDL)
- 3rd party VIP needed to support the VPI capability
- Save/Restart limitations on when the save and restore could occur

Current Solution

New Vendor solution

- Process image is saved and then can be restored
- The entire memory image of the simulation process is saved
  - Includes state of all models being simulated
  - Includes any files being read or written

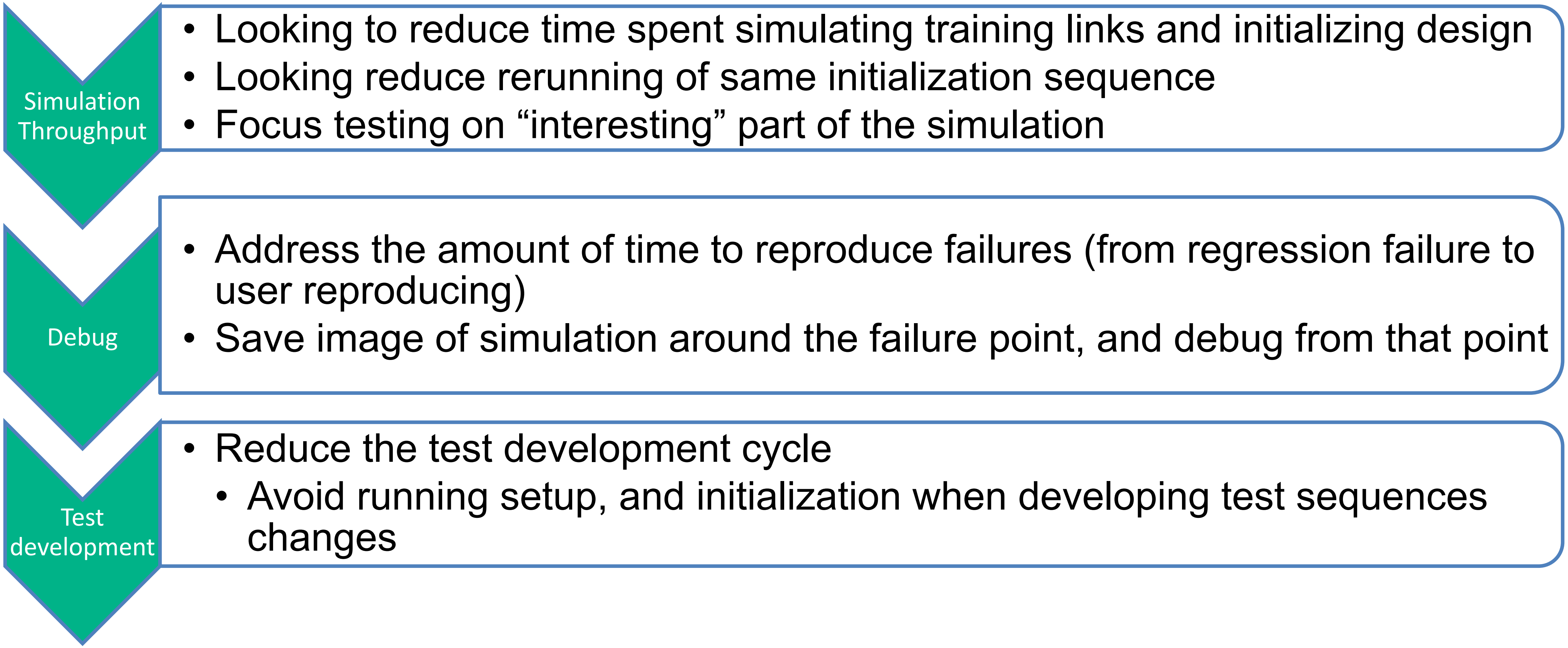
Benefits

- No longer need to worry about 3rd party or external code support
- Simpler enablement of the save/restore technology
- Support for other verification languages (C,Specman-e, VHDL)

Drawbacks

- Saved image size is much larger than previous solution
  - Compression of save image is needed
- Must still develop a methodology around save/restore capability

Productivity improvements



Simulation Throughput

- Looking to reduce time spent simulating training links and initializing design
- Looking reduce rerunning of same initialization sequence
- Focus testing on “interesting” part of the simulation

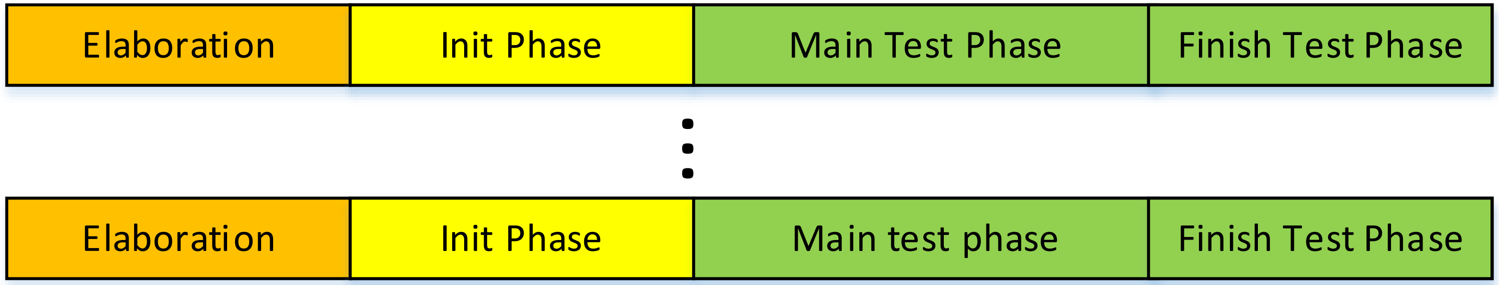
Debug

- Address the amount of time to reproduce failures (from regression failure to user reproducing)
- Save image of simulation around the failure point, and debug from that point

Test development

- Reduce the test development cycle
  - Avoid running setup, and initialization when developing test sequences changes

Simulation Throughput: Simulation Testing Modes



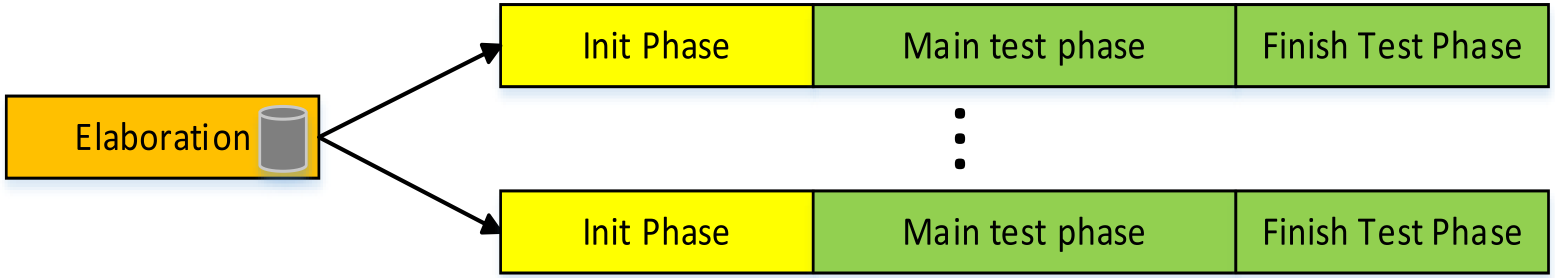
Elaboration | Init Phase | Main Test Phase | Finish Test Phase

⋮

Elaboration | Init Phase | Main test phase | Finish Test Phase

Runmode: 1

Each test runs through all phases (elab, init, main test, finish test)



Elaboration

Init Phase | Main test phase | Finish Test Phase

⋮

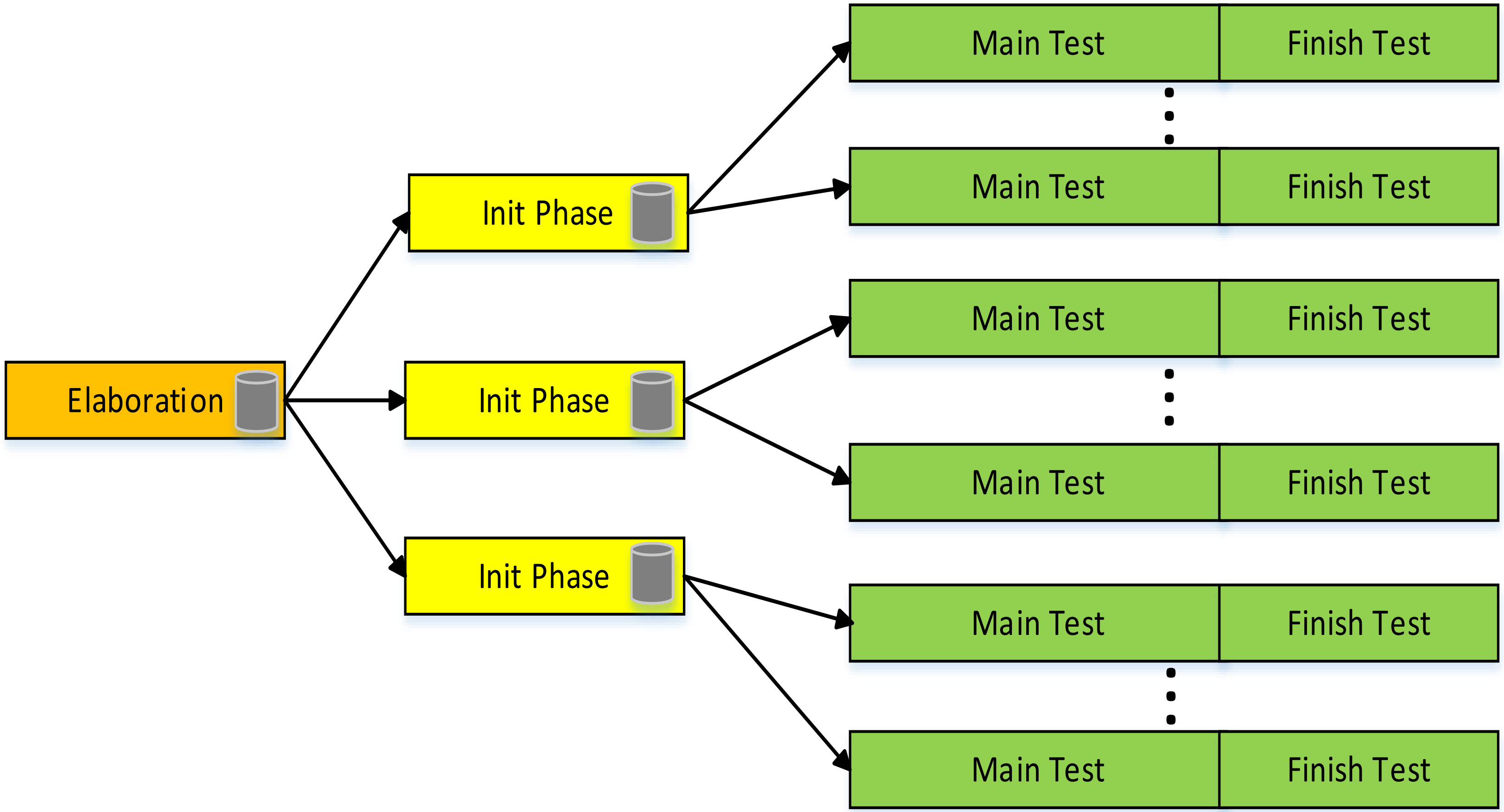
Init Phase | Main test phase | Finish Test Phase

Runmode: 2

Single elaboration is done for all tests of a given topology, and an elab snapshot is created.

Each test uses elab snapshot of design and then steps through rest of test phases (init, main test, finish test)

Simulation Throughput
Save Restore: Two Stage Testing



Single elaboration is done for all tests of a given topology, and an elab snapshot is created. Multiple init phase snapshots created for each unique configuration. The tests use the init snapshot and then steps through rest of test phases (main test, finish test)

Simulation Throughput by testing modes

	Elaboration (E), Initialization (I), Test (T)														
Runmode 1	E	I	T	E	I	T	E	I	T	E	I	T	E	I	T
	Fewer test runs without snapshot														
Runmode 2	E	I	T	I	T	I	T	I	T	I	T	I	T	I	T
	I Elab, N-initialization, N-Tests														
Two-Stage Testing	E	I	T	T	T	T	T	T	T	T	T	T	T	T	T
	1 Elab, 1-Initialization, N-Tests														

Runmode 1

- 1/3 of jobs are tests

Runmode 2

- ~1/2 jobs are tests

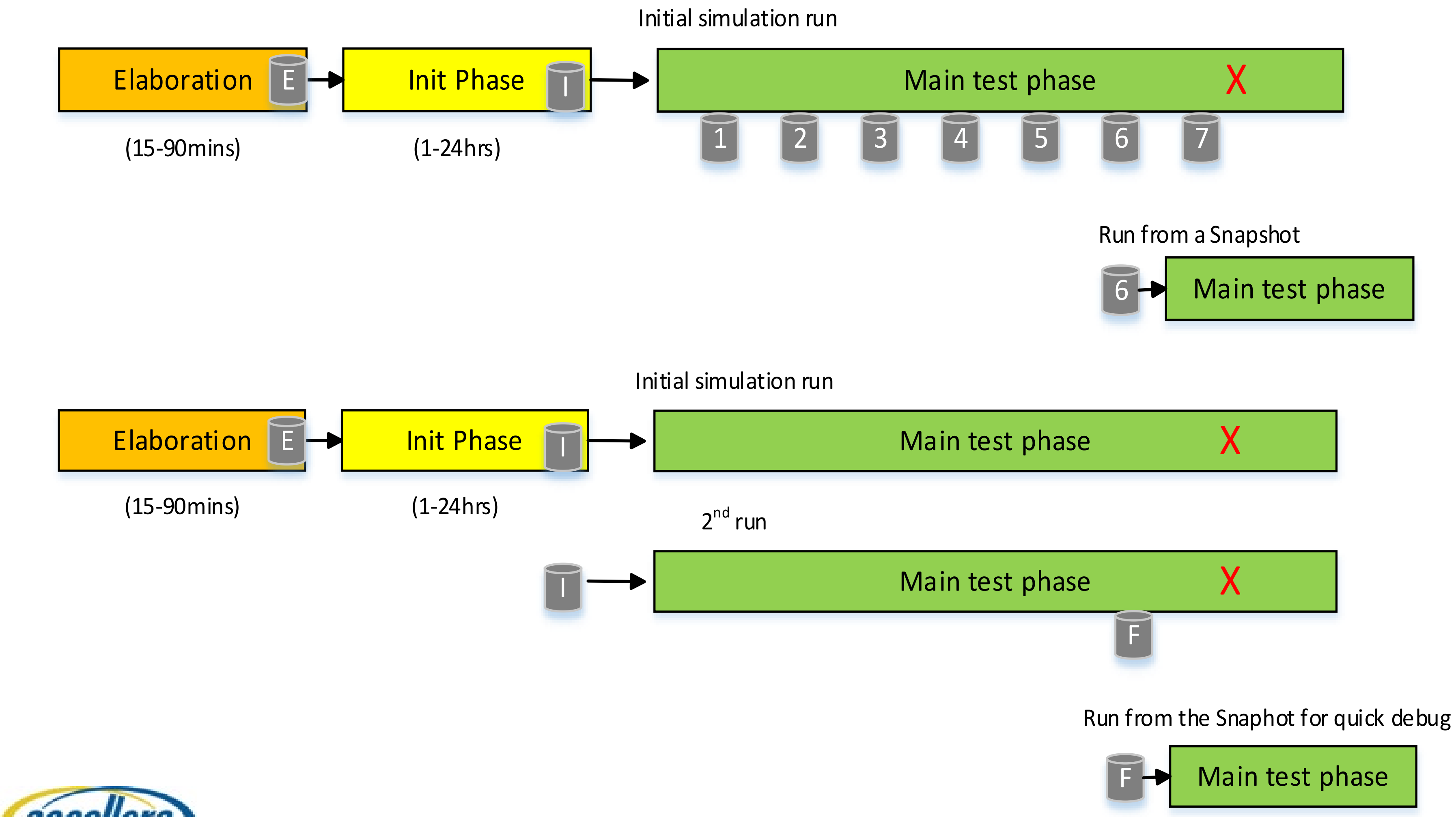
Two-Stage testing

- All but two jobs are tests

Concerns

- Getting enough testing of setup/initialization
  - Focused tests on just initialization
  - Defined specific initialization modes for testing
  - Added randomization of state in the tests

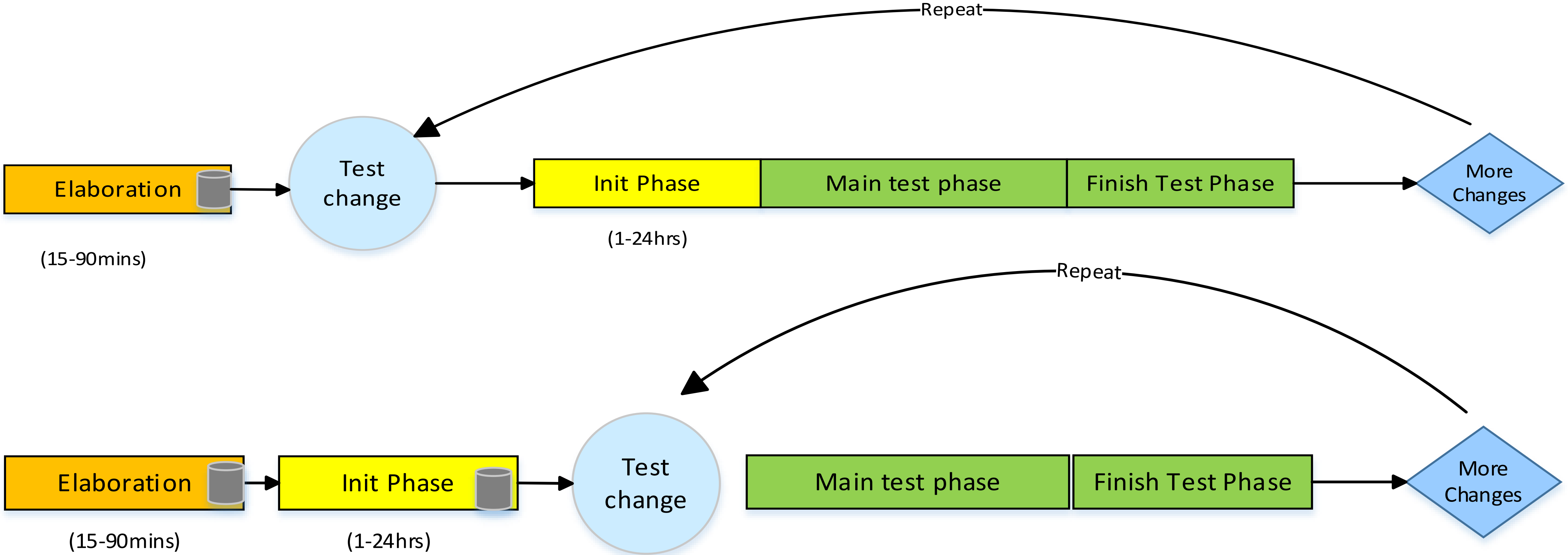
Improved Debug efficiency



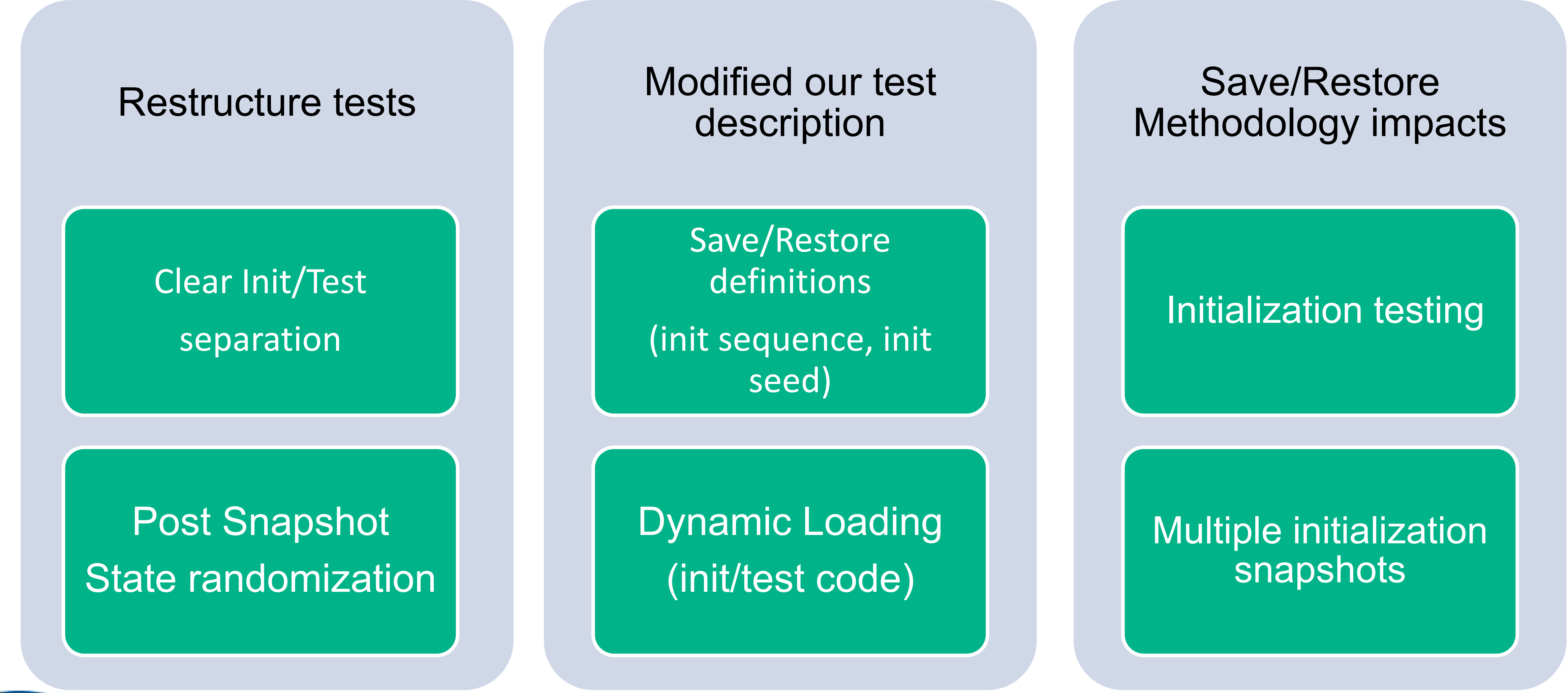
Periodic snapshots taken during the test phase and when a simulation fails a snapshot can be loaded and used to do debug from that snapshot. No longer have to run from the beginning. High cost on disk space to store snapshots.

Run simulations with limited debug and wave information, and then when failure occurs rerun simulation but use failure time from initial run to save snapshot some specified time before failure

Test Development efficiency



Challenges and Complexities: Test Development



Restructure tests

- Clear Init/Test separation
- Post Snapshot State randomization

Modified our test description

- Save/Restore definitions (init sequence, init seed)
- Dynamic Loading (init/test code)

Save/Restore Methodology impacts

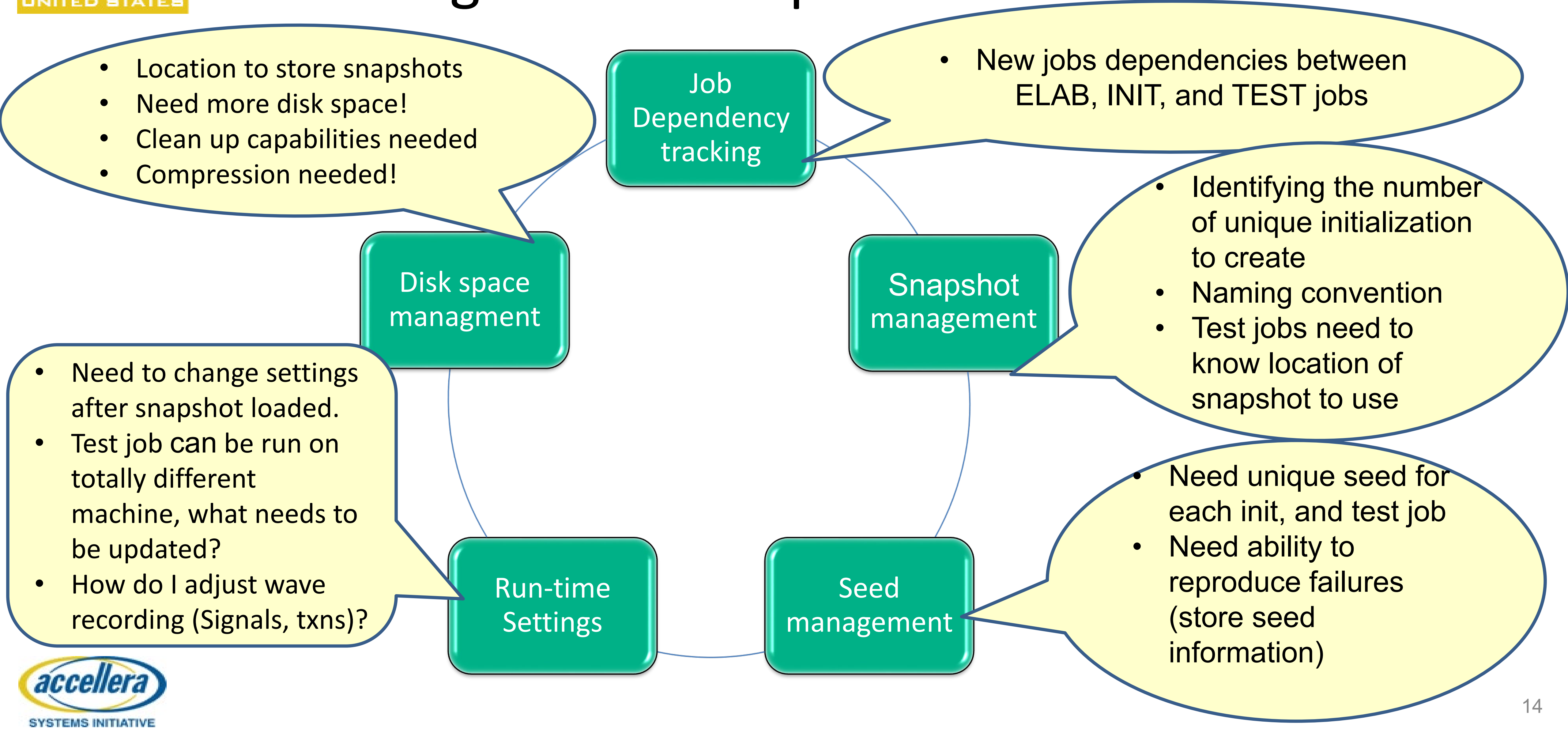
- Initialization testing
- Multiple initialization snapshots

Test Description

```
:Identifier   dut_2stage_atomic_traffic_1_0
:Repeat 10
:Repeat_init 2
:Class  (  all two_stage_turnon)
:Method   simulation
  (
    test_mode ( e )
    topo_dir   (#THIS#/topo/dut_rsp_ip/e)
    test_dir   (#THIS#/test/dut_rsp_ip/e)
    init_name ( standard_init )
    test_name ( atomic_traffic_test )
    compargs (  #G01_0_COMPARGS# )
    ecode    (#ECFG_VER_1_0#)
```

```
test_ecode
    (
      "extend data_pkt_s {
              keep global_cid_pres.reset_soft();
      };"
    )
    memory ( 3G )
    slots ( 1 )
    init_seed( random )
    seed ( random )
  )
:Owner     Ed Powell
:Summary Demonstrate save/restart testing
```

Challenges and Complexities: Environment



Job Dependency tracking

- New jobs dependencies between ELAB, INIT, and TEST jobs

Snapshot management

- Identifying the number of unique initialization to create
- Naming convention
- Test jobs need to know location of snapshot to use

Seed management

- Need unique seed for each init, and test job
- Need ability to reproduce failures (store seed information)

Run-time Settings

- Need to change settings after snapshot loaded.
- Test job can be run on totally different machine, what needs to be updated?
- How do I adjust wave recording (Signals, txns)?

Disk space managment

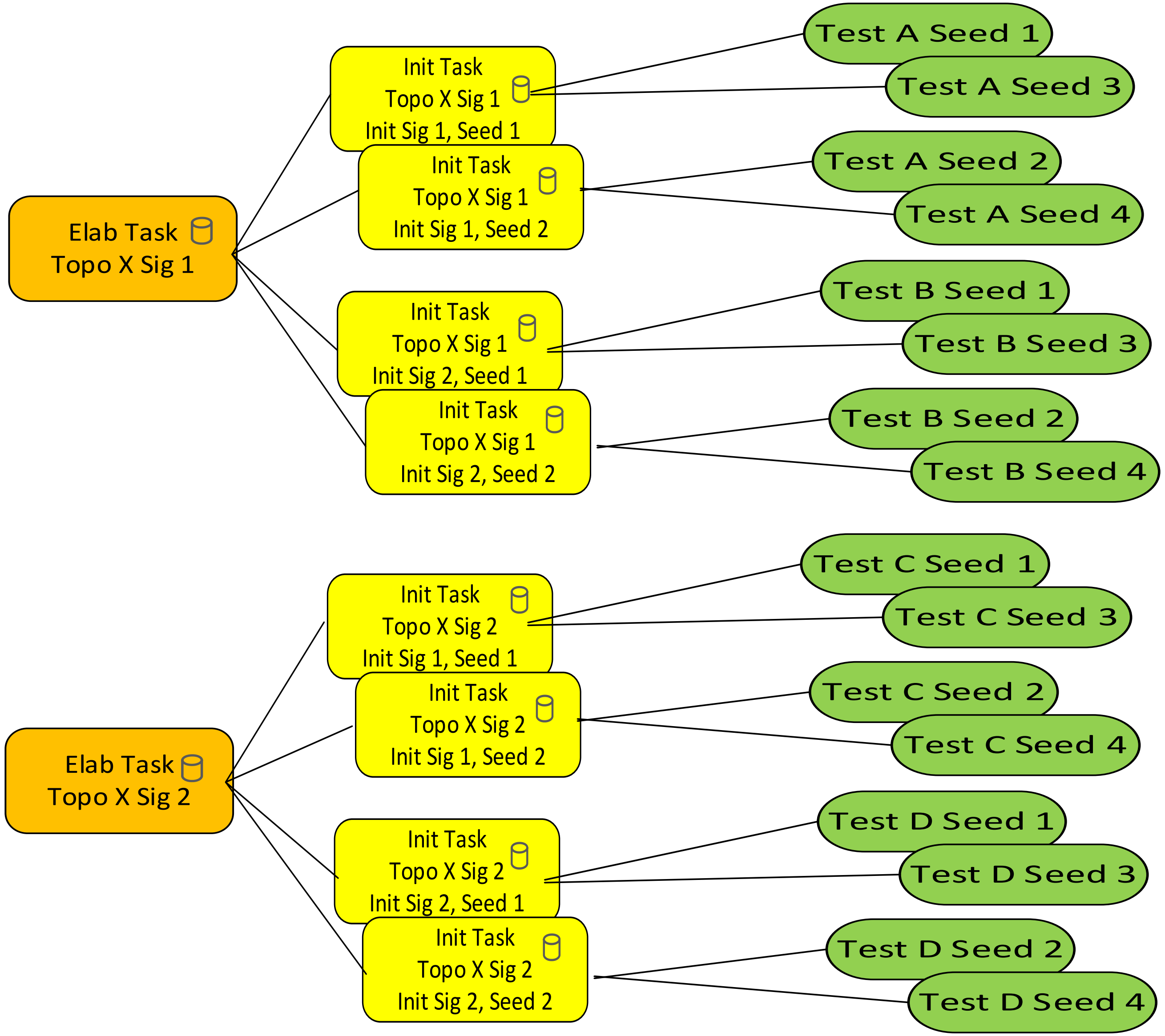
- Location to store snapshots
- Need more disk space!
- Clean up capabilities needed
- Compression needed!

Managing job signatures and init seeds

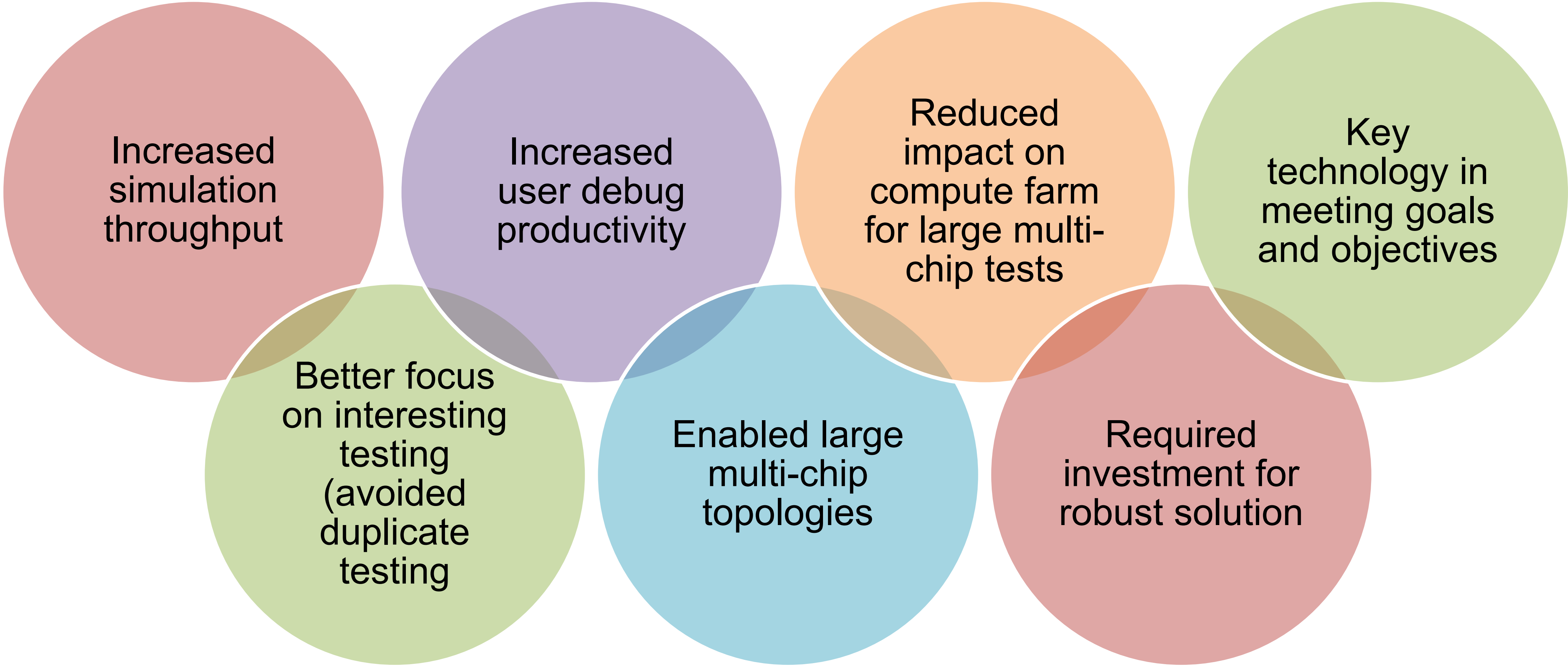
Necessary to manage and track job creation, job dependency and seed management.

Looking at a Single topology with:

- Multiple configuration of topology
- Running with multiple initialization snapshots
- Running multiple tests on each snapshot



Results: What did we achieve?



Increased simulation throughput

Increased user debug productivity

Reduced impact on compute farm for large multi-chip tests

Key technology in meeting goals and objectives

Better focus on interesting testing (avoided duplicate testing

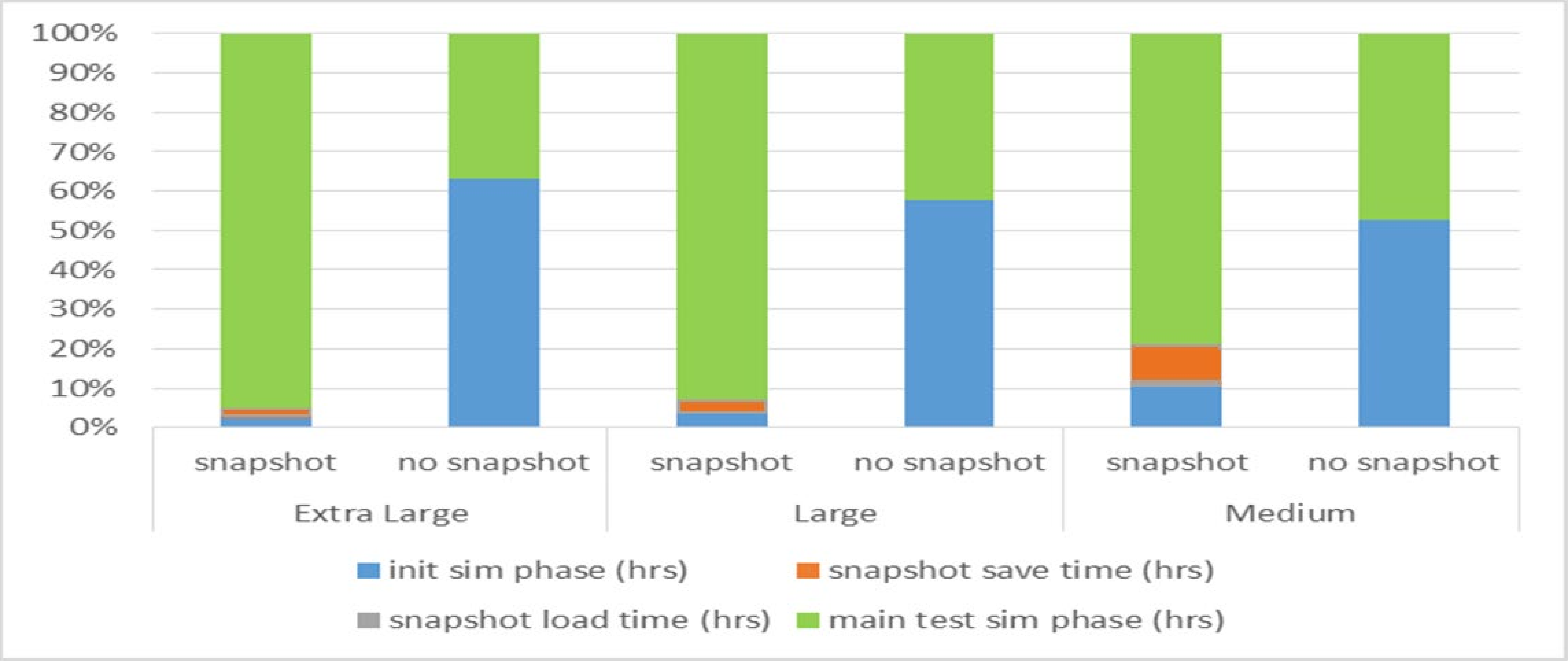
Enabled large multi-chip topologies

Required investment for robust solution

Results: Simulation Throughput

Topology Size	Design Size: Module Instances	Design Size: Register Instances	Simulation Run-Time Memory Usage (GB)	Typical Simulation Duration (hrs) (test portion)	Typical % of Duration Spent in Init Phase
Medium	100-200K	1-2M	15-20	0.75	40-50%
Large	650-1600K	7-11M	50-60	2.5	50-60%
Extra Large	2-10M	15-45M	60-100	5-48	50-75%

ROI of Save/Restore



Disk usage from regression

Topology Size	Average Init Snapshot DISK Size	Average Init Snapshots Captured Per Day	Total Disk Space Used for Init Snapshots Per Day
Medium	900 MB	27	22.5 GB
Large	1.6 GB	26	41.6 GB
Extra Large	3.6 GB	35	126.0 GB

Impact of Enabling Compression

Topology Size	Compression Setting	Size of Init Snapshot (GB)	Reduction in Snapshot Size (%)	Time to Save Snapshot (sec)	Time to Load Snapshot (sec)
Medium	None	6.5	-	16	27
	1	1.2	81.5	70	51
	3	1.1	83	80	50
	5	1.1	83	116	49
	7	1.09	83.2	242	49
Large	None	12	-	30	42
	1	1.9	84	125	88
	3	1.8	85	151	90
	5	1.8	85	205	88
	7	1.7	85.8	419	88
Extra Large	None	23	-	55	101
	1	3.6	84	209	154

Conclusion

- Full process image Save/Restore is now available
- A robust Save/Restore methodology can be developed
- Throughput, capacity, and debug productivity are improved
- Snapshot seeds and disk space must be managed
- Size and complexity of designs practical in simulation is increased