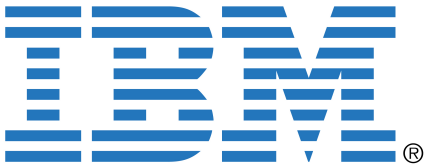
Post-Silicon Debugging of Transactional Memory Tests

Ophir Friedler, Wisam Kadry,
Amir Nahir, Vitali Sokhin
{ophirf, wisamk, nahir, vitali}@il.ibm.com

IBM Research

Carla Ferreira, João Lourenço
{carla.ferreira, joao.lourenco}@fct.unl.pt

Universidade Nova de Lisboa



Post Silicon

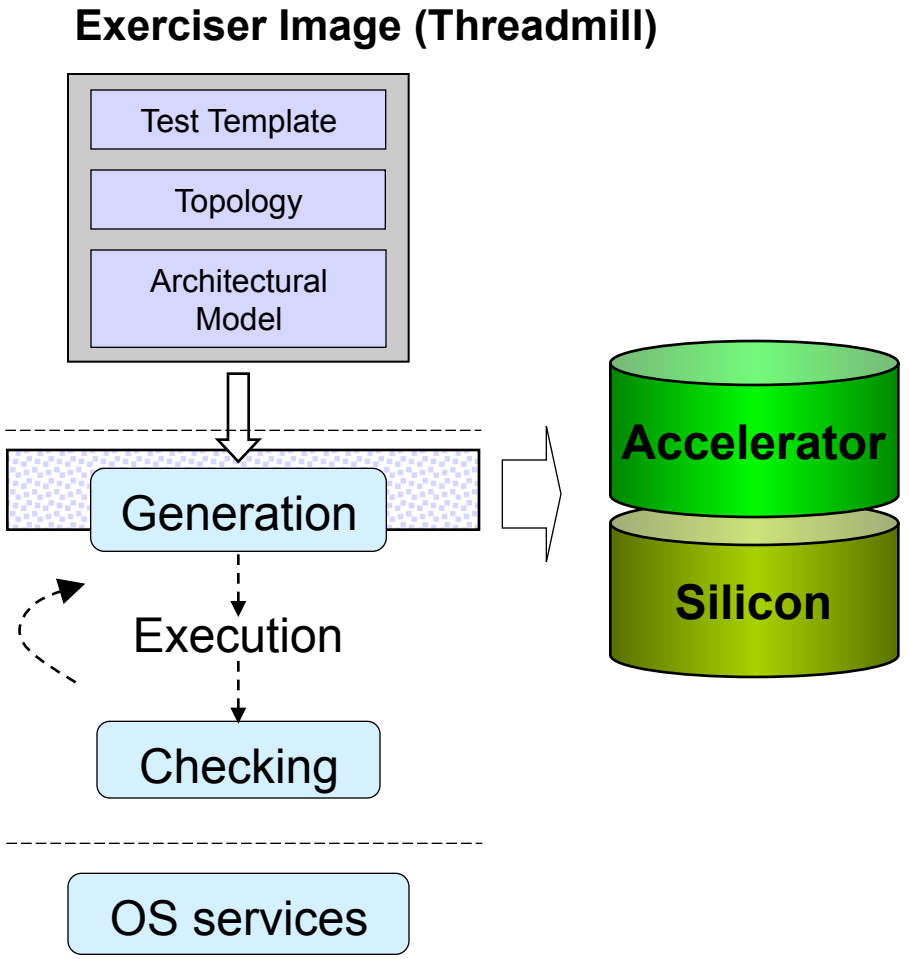
Post-silicon validation elements:

1. Stimulating the design under test
2. Detecting erroneous behavior
3. Localizing the root cause of the problem
4. Providing a fix.

Stimulation

1. Test generation
2. Execution
3. Consistency checking

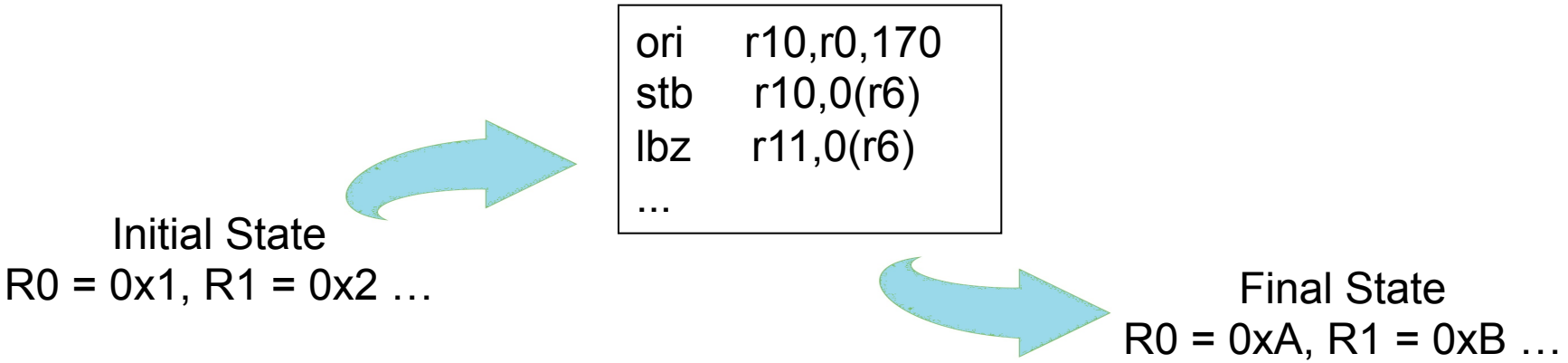
4. Repeat… Forever!



Detection

Consistency checking

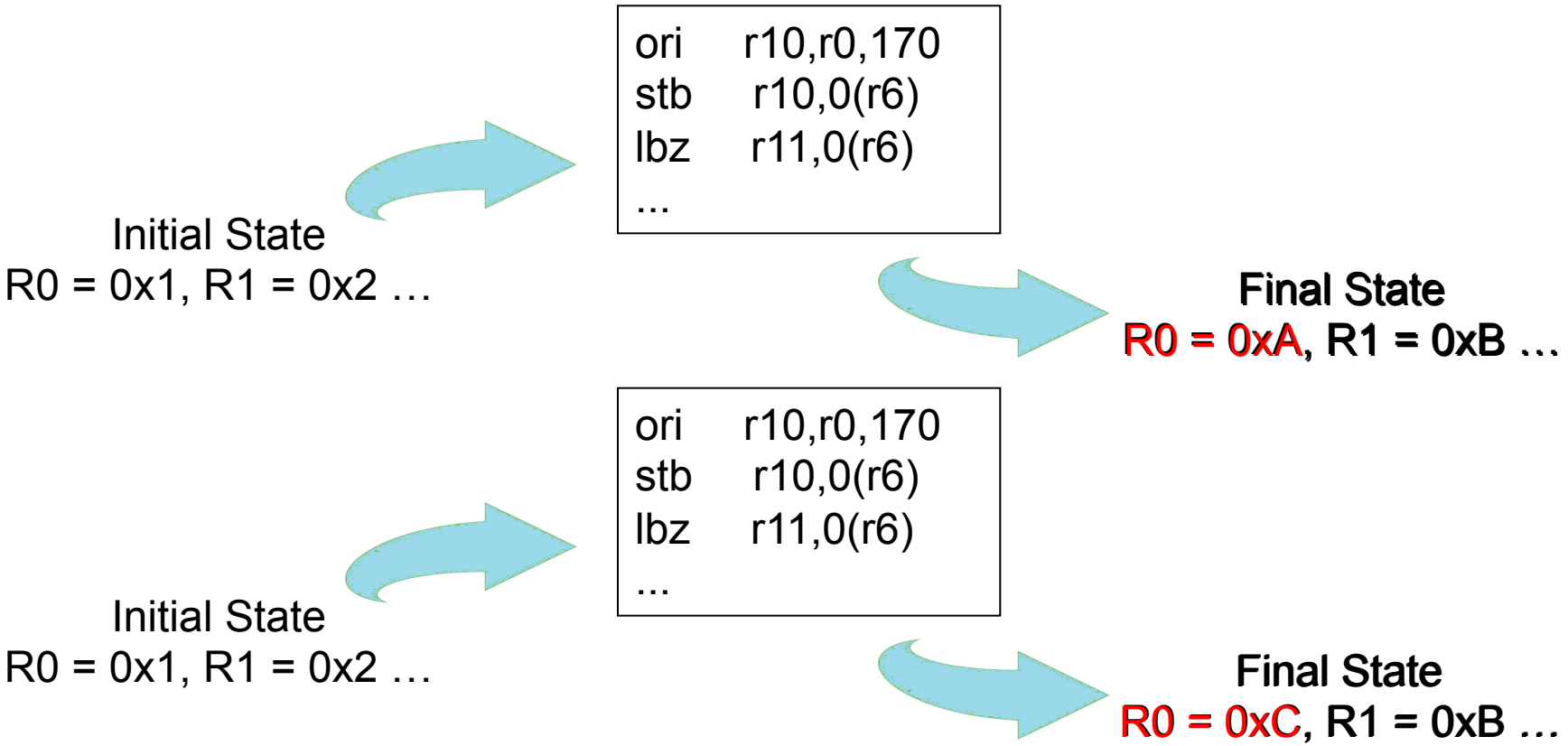
- Run the same test-case from the same initial architectural state.
- Expect the same final architectural state



Micro-architectural state varies!
Caches, page misses, pre-fetching, thread priorities

Detection

And what if two different final states are manifested?



Localization approach

1. A test-case that produces a mis-compare is found

2. Fast-forward to that test-case on a software simulator (a.k.a. Reference model)

3. Execute test case on the reference model instruction by instruction and extract information

Localization

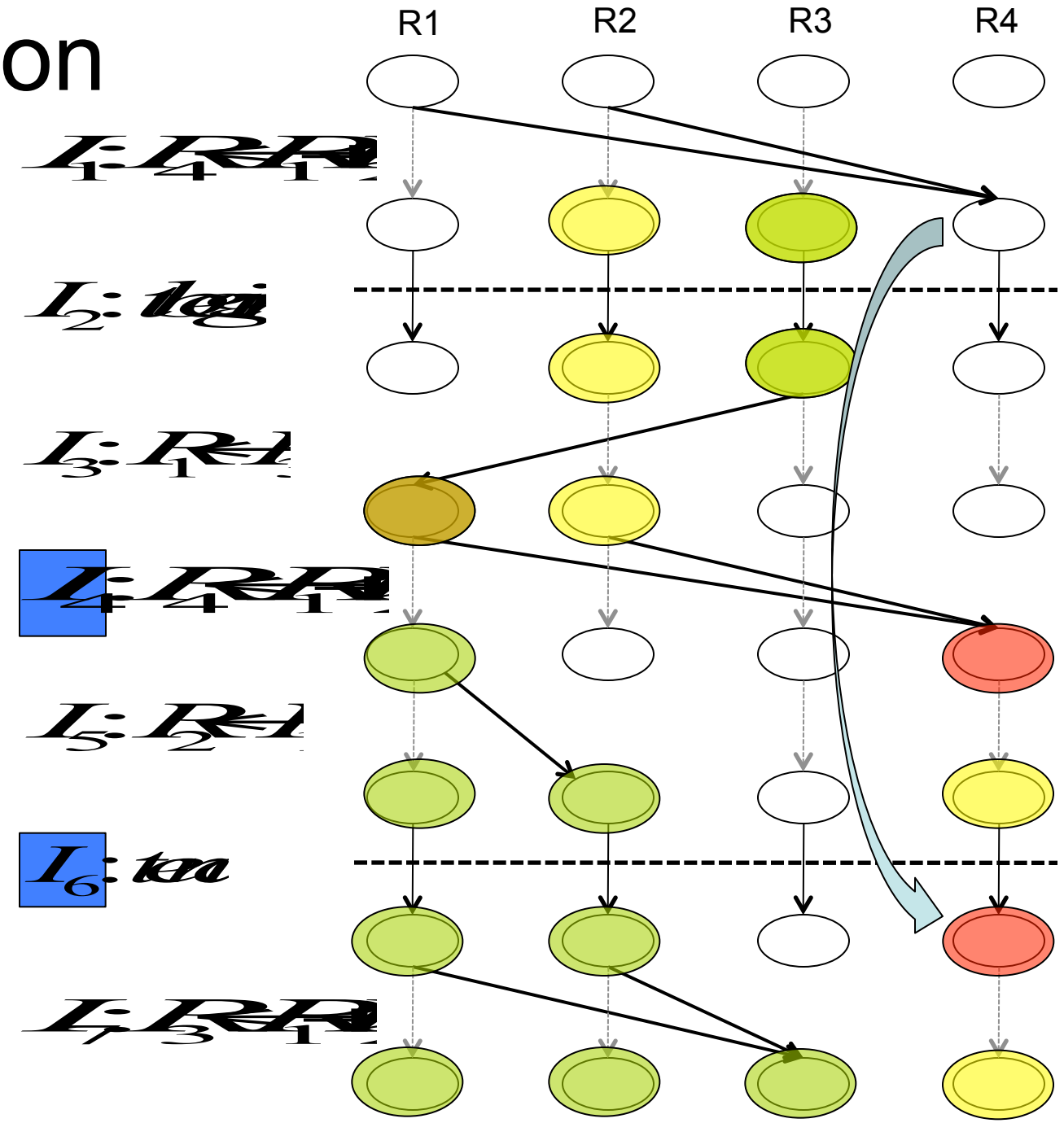
- Reduce number of resources and instructions that might be the root cause of the mis-compare

- Study the effect of transactions in the test-case on the final state.

Justification:

- Force erroneous behaviour on reference model and re-create the mis-compare results

Localization



= suspicious instruction subset

Concluding remarks

- Debug automation effectively reduces the debugging effort.

- Graph analysis holds the potential automate the localization of suspicious resources and instructions

Future work:

- Study the impact of escaped stores in transaction aborts
- experiment with larger (real-world) cases

Questions