2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION CHINA

accellera SYSTEMS INITIATIVE

Atomic sequence and multilayer regression strategy based on Saving and Restoring methodology

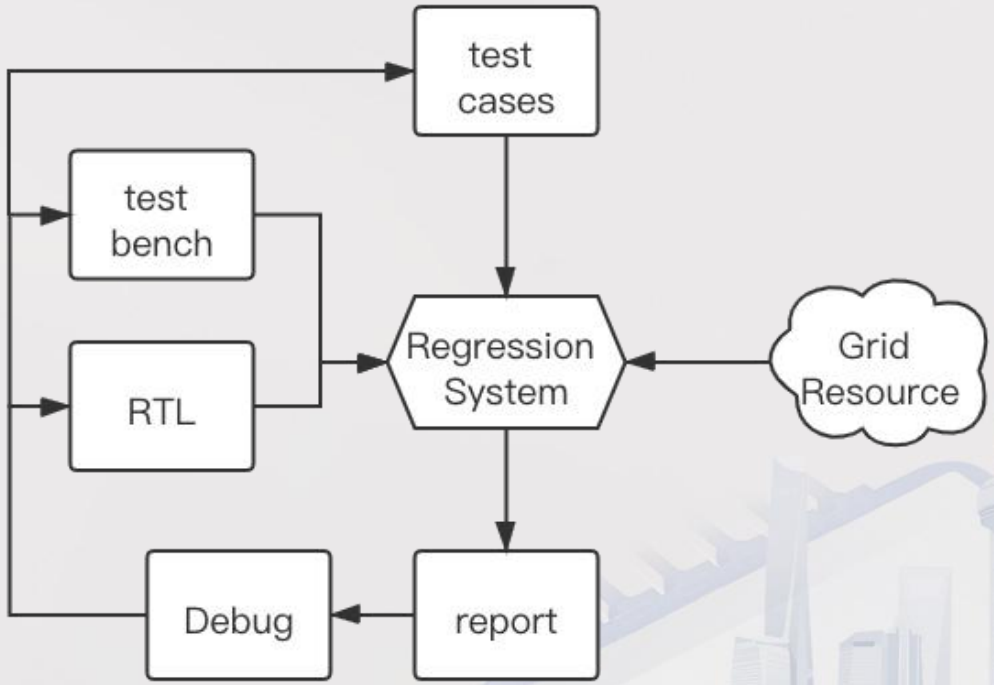
Alvin.Zhang@evas.ai

Shanghai | September 20, 2023

EVAS INTELLIGENCE

Regression System

- Simulation regressions consist of many test cases and be kicked off for periodic running to qualify designs and test environment.
  - Test cases
  - Test bench
  - RTL
  - Grid resource



Current Regression Strategy

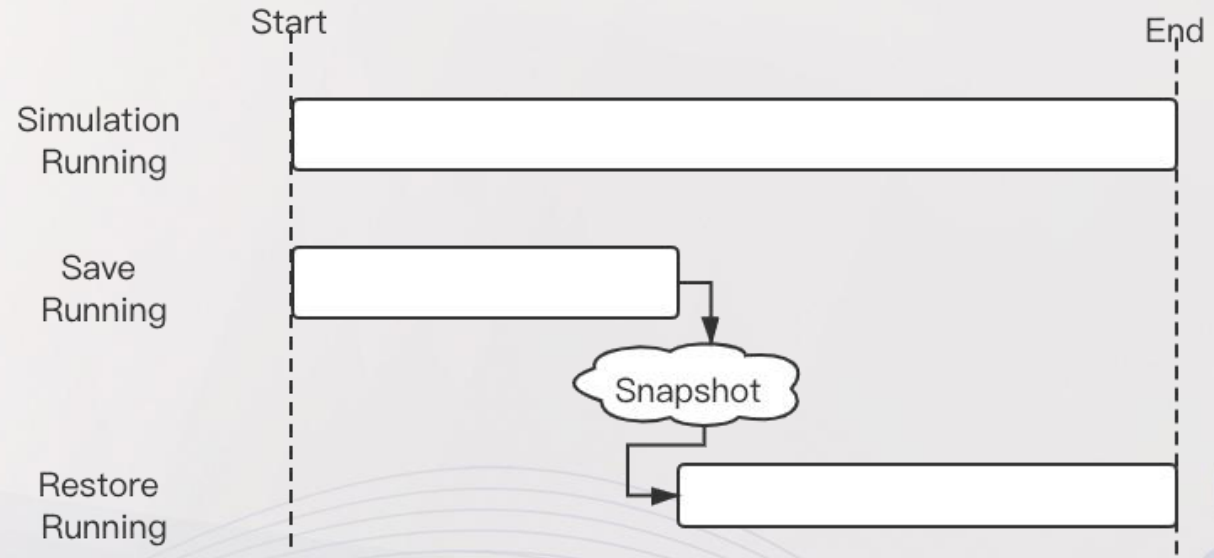
- Group all test cases and run frequently along with design and verification database upgrade.
- Design verification engineers analyze all failures in regressions and catch bugs in designs or test bench.
- Repeat above steps until DV sign-off
- Sometimes, regression run at the same database change list for issues reproduction or fixes qualification.

Problems

- As the size and complexity of logic designs increase significantly, regressions running can be very grid resource intensive and time consuming.
- Saving and Restoring (SnR) methodology introduces a new method to reduce test case simulation turnaround time (TAT)
- Current regression strategy cannot adopt SnR methodology

Saving and Restoring(SnR)

- The basic concept of SnR methodology is to specify a specific point during simulation and then skip the iterative simulation sequences using the point as the starting point of the next simulation



Brainstorming

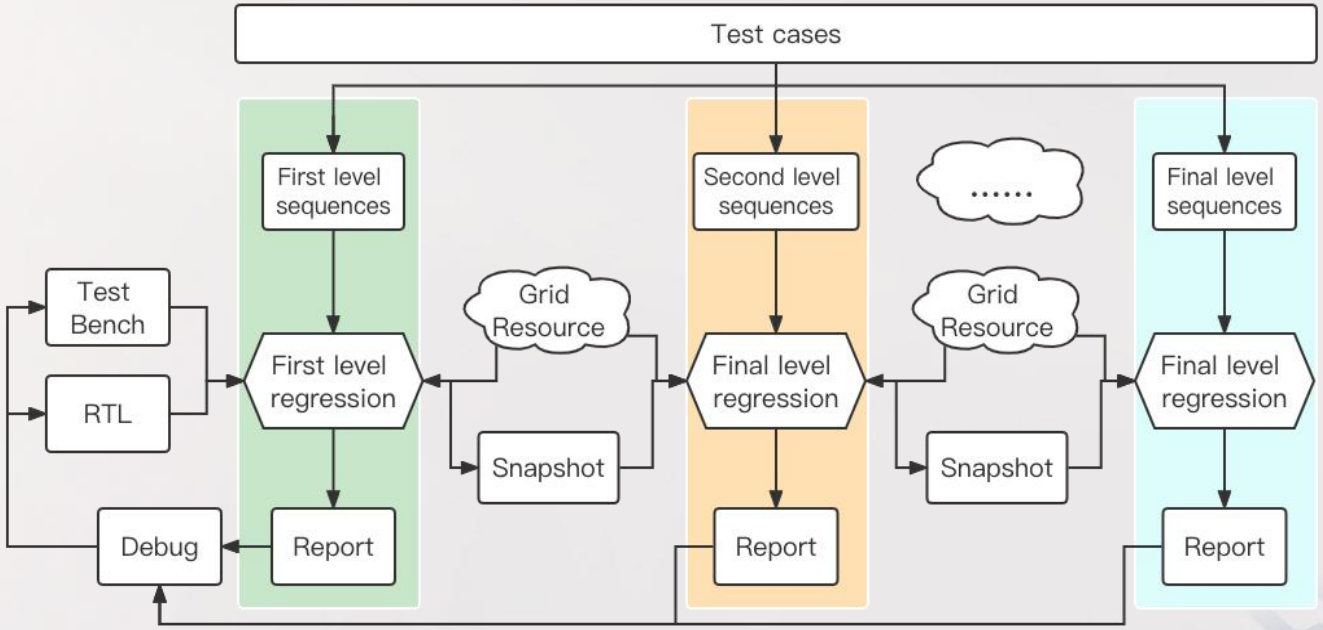
- SnR is two-steps methodology and not be applicable to traditional one level regression system.
- All test cases in regression been kicked off simultaneously and independently, each test case cannot leverage simulation database of other cases.
- Regression database changes frequently along with project execution, it is inefficient to trigger two full regression - one for saving and one for restoring at the same change list based on SnR methodology.

Atomic Sequence

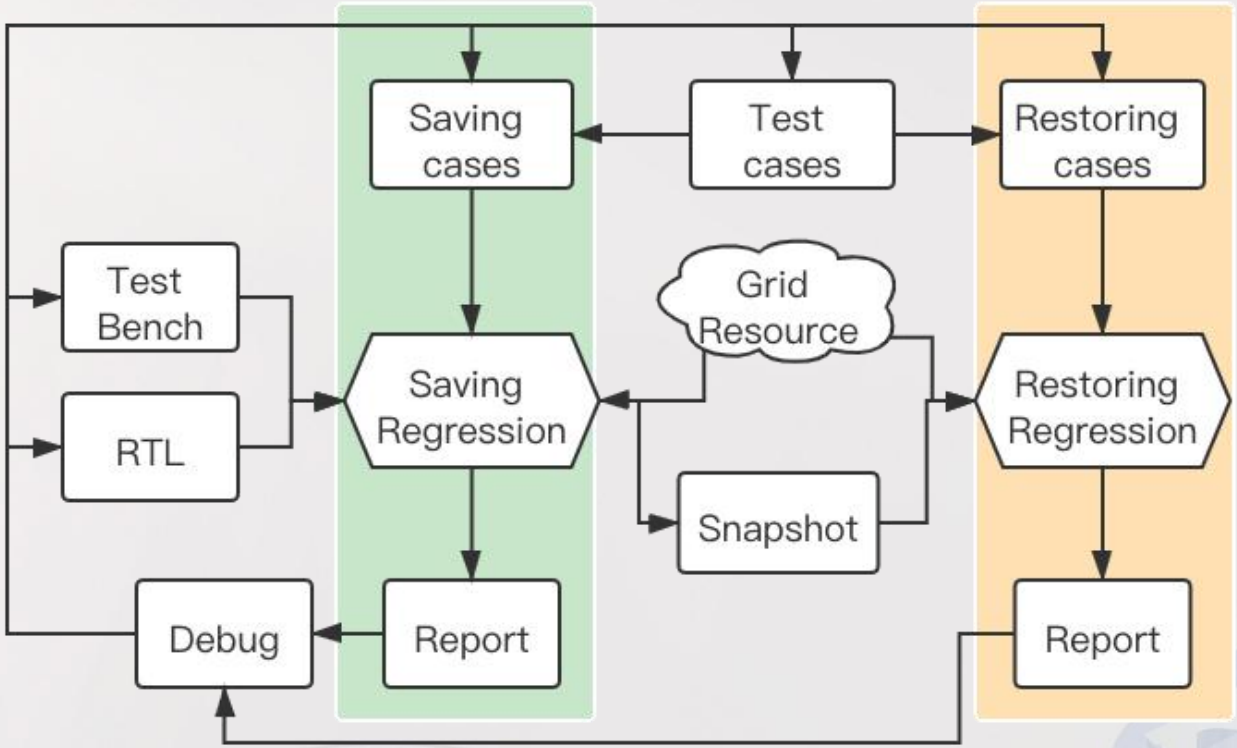
- Based on simulation timeslots, Similar as UVM phase, atomic sequence receives execution result from previous timeslot, and starts to execute its function and send the result to next timeslot.
- DUT scope, clock setting and reset sequences, memory initialization, initialization registers configurations and kinds of production firmware version……

test scenario1	atomic sequence1.1	atomic sequence2.1	atomic sequence 3.1
test scenario2	atomic sequence1.1	atomic sequence2.1	atomic sequence 3.2
test scenario3	atomic sequence1.2	atomic sequence2.1	atomic sequence 3.3
test scenario4	atomic sequence1.2	atomic sequence2.2	atomic sequence 3.4
test scenario5	atomic sequence1.2	atomic sequence2.3	atomic sequence 3.5
test scenario6	atomic sequence1.3	atomic sequence2.3	atomic sequence 3.6

Multilayer Regression System



Two level regressions



Regression migration

- Step1: Analyze all test cases for atomic sequences classification
- Step2: Create saving cases to define test framework and grouped as Saving regression
- Step3: Optimize all test cases and grouped as Restoring regression
- Step4: Update test description files and regression lists

Test Framework

- Test framework is a bridge for Saving regression and Restoring regression. There maybe kinds of test framework, it depends heavily on test bench and environment, case structures, or even test flow.
- Each type of atomic sequences and configurations need a saving case and its test framework accordingly. To create saving case with appropriate test framework is key point for regression efficiency improvement.

Example

- This test framework has two common atomic sequences and one specific sequence invoking interface.

```
Example: saving_case1 – saving_vseq1.sv
task saving_vseq1::body();
    common_sequence1            initial_seq1;
    common_sequence2            initial_seq2;
    ……
    soc_base_vseq               function_seq;
    uvm_object_wrapper          seq_wrapper;
    string                      specific_seq = "soc_base_vseq";
    ……
    `uvm_do(initial_seq1)
    `uvm_do(initial_seq2)

    $save("snapshot1");
    SnR_parser_function();

    begin
        $value$plusargs("SPECIFIC_SEQ=%s", specific_seq);
        seq_wrapper = factory.find_by_name(specific_seq);
        $cast(function_seq, create_item(seq_wrapper, m_sequencer, "function_seq"));
        If(specific_seq != "soc_base_vseq") begin
            function_seq.start(p_sequencer)
        end
    end
endtask: body
```

Example - Cont.

- SnR_parser_function function. This is a necessary function for re-parse program.
- Once case run with restoring from snapshots, all progress before saving point will be skipped such as static UVM phase and simulation arguments parsing program. We have to rely on the function to re-parse options and arguments, to execute specific sequences with different configurations from common sequences.
- But note that if the configuration impact any simulation execution before snapshot, such as test bench configuration, DUT configuration or booting mode, we have to create different saving cases with according framework definition respectively instead of invoking this function.

Example - Cont.

- Test description files

```
Example: test_case1 - original description file

test_case1 : soc_base_test
    +uvm_set_type_override=soc_base_vseq, test_case1_dedicated_vseq
    ......
```

```
Example: test_case1 - new description file

test_case1 : soc_base_test
    +uvm_set_type_override=soc_base_vseq, saving_vseq1
    +SPECIFIC_SEQ=specific_vseq1
    ......
```

Example - Cont.

Example: regression list - Saving regression

```
saving_case1 SAVE snapshot1
saving_case2 SAVE snapshot2
saving_case3 SAVE snapshot3
   ......
```

Example: regression list - Restoring regression

```
test_case1 RESTORE snapshot1
test_case2 RESTORE snapshot1
   ......
test_case50 RESTORE snapshot1
test_case51 RESTORE snapshot2
   ......
test_case200 RESTORE snapshot2
test_case201 RESTORE snapshot3
   ......
```

Advantages

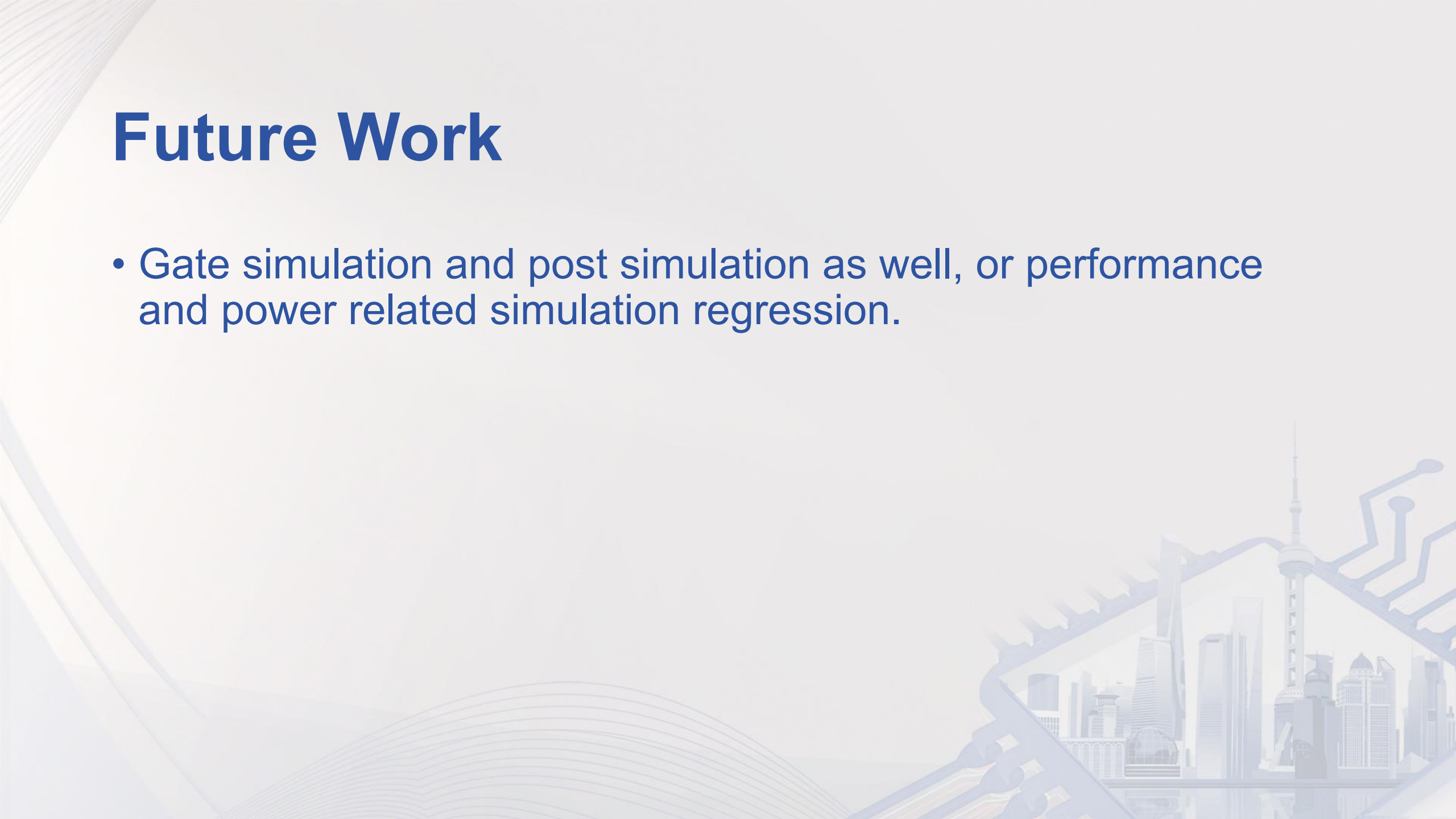
- This new strategy is fully aligned with SnR methodology to remove redundant common sequences execution for regression running speeding up and save much time and grid resource.
- Multi-layer regression is more efficient. Different running frequencies for Saving regression and Restoring regression is a good balance between database change list alignment and grid resource consuming.
- This new strategy benefits local debug progress.The reproduction running will restore from snapshot directly and finish execution in much shorter time.

Disadvantages

- Most challenging part is to create saving cases to define appropriate test frameworks. UVM based test bench enjoys the benefits of well-defined test structure, but it is not easy for some mixed-language in-parallel test benches to define clear and robust test frameworks[4].
- One more limitation is about Saving regression and Restoring regression alignment. Snapshots on different database change list should be picked up carefully to make sure no modifications in atomic sequences and configurations of saving and restoring cases. Fortunately, initialization part always freeze very early during project execution.

Future Work

- Gate simulation and post simulation as well, or performance and power related simulation regression.



Reference

- [1] Cadence, “Process based save restart in UVM”
- [2] Cadence, “Using Save/Restart Checkpointing with Xcelium”
- [3] Ahhyung Shin, Yungi Um, Youngsik Kim, Seonil Brian Choi, Samsung Electronics, “Saving and Restoring Simulation Methodology using UVM Factory Overriding to Reduce Simulation Turnaround Time”
- [4] Rohit K Jain and Shobana Sudhakar, Mentor Graphics “Want a Boost in your Regression Throughput? Simulate common setup phase only once.”