2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Want a Boost in your Regression Throughput?

Simulate common setup phase only once.

Mentor Graphics®

Command-line options on Questasim to run simulation till 1000ns and checkpoint simulation state

```
//Simulate design ‘top’ till ‘1000ns’,
then checkpoint the state in file
‘test.chk’, and quit simulation
1%> vsim –c top –do “run 1000ns;
checkpoint test.chk; quit –f”


//Restore design state by loading
‘test.chk’ and continue simulation
2%> vsim –c –restore test.chk –do “run –a;
quit –f”
```

```
test.sv:
   task run_test();
       //... Execute initialization phase ..
       //Checkpoint design by setting
checkpoint_enable variable to 1
       checkpoint_enable = 1;
       //Allow simulator to stop simulation
and checkpoint the design
       #1;
       // After restore, simulation will
continue from here
   endtask //run_test

# This do file is specified at simulator’s
command line
checkpoint.do:
    onbreak resume
    set do_checkpoint 0
    # When design sets the
#‘checkpoint_enable’ variable to 1, trigger
#checkpoint command by setting do_checkpoint
    when {checkpoint_enable == 1} {
       echo “Stopping simulation to create a
checkpoint at time $now”
       set do_checkpoint 1
       stop
    }
    while{1} {
       run –all
       # If checkpoint is triggered, save
design state and quit simulation
       if{$do_checkpoint} {
          set do_checkpoint 0
          checkpoint test.chk
          quit –f
       }
    }
1%> vsim –c top –do checkpoint.do
2%> vsim –c –restore test.chk –do “run -a;
quit –f”
```

Common setup phase reuse methodology flow diagram

Step 1: Elaborate the design and simulate till your initial common setup phase is done. This simulation point can be determined by the designer.

Step 2: Checkpoint the state of a simulated design at this simulation point (end of common setup phase).

Step 3: Restore the saved design state and use this checkpoint state as starting point for all your tests

Step 4: Change the design test variables so that different test configurations take effect starting from the same checkpoint state.

Step 5: Continue running the test from last simulation checkpoint. This is effectively equivalent to elaborating and running the design from beginning for given test configuration.

Repeat with different test variables to run different tests.

Saving simulation state based on the state of a variable, in dynamic activity based designs

```
#include "mti.h"
//Begin existing user code
svScope my_scope;  // This variable should preserve
its value upon restore
void set_my_scope() {
    my_scope = svGetScope();  //Function called at
init phase to store the scope.
}
//End existing user code
//Begin code additions for checkpoint/restore
void mySave() {  // Save my_scope during checkpoint
    mti_SaveBlock((char*) &my_scope, sizeof(svScope));
}
void myRestore() {  //Restore my_scope after restore
    mti_RestoreBlock((char*)&my_scope);
}
//Static constructor registers callbacks for
checkpoint and restore
class CheckpointHandlerBootStrapper {
public:
  CheckpointHandlerBootStrapper() {
    mti_AddDPISaveRestoreCB(
      reinterpret_cast<mtiVoidFuncPtrT>(mySave),
      const_cast<char*>("myRestore"));
  }
};
static CheckpointHandlerBootStrapper
checkpointhandlerbootstrapper;
//End code additions for checkpoint/restore
```

Saving and restoring C/C++ global variables using API methods available in Questasim

Restoring simulation state and modifying test stimulus file name to run different tests

```
// Checkpoint here
if($value$plusargs(“STIM_NAME=%s”,
filename))
    $display(“Stimulus file name has been
set to %s”, filename);

// Pick test-specific stimulus file
$display($time, “Loading stimulus from
file %s :”, filename);
begin : read_test_file
    int fd, eof;
    string line;
    fd = $fopen(filename, “r”);
    eof = !$fgets(line,fd);
    while(!eof) begin : get_line
       $display(“Reading line %s”, line);
       eof = !$fgets(line,fd);
    end
    $fclose(fd);
end

1%> vsim –c –restore test.chk
+STIM_NAME=test_2 –do “run -a; quit –f”
```

Restoring simulation state with command-line plusargs in Questasim

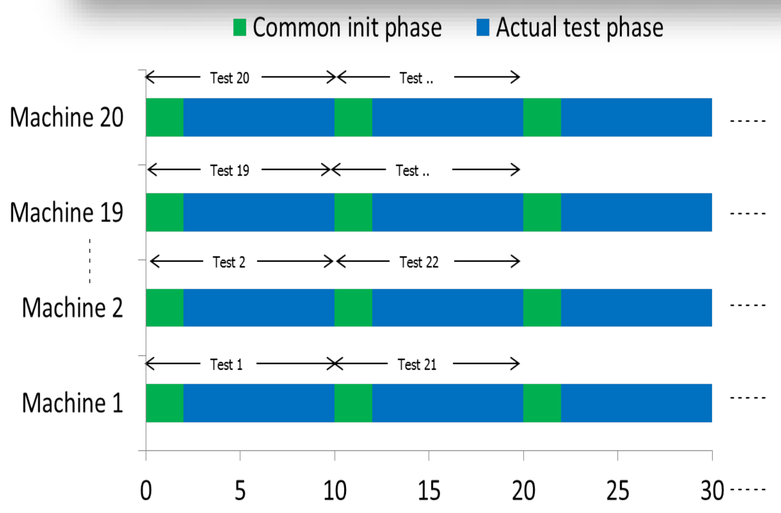
```
string name;
initial begin
    #1000
    // Checkpoint here
    // After restore, Run test based on value of TESTNAME plusargs
    if ($value$plusargs(“TESTNAME=%s”, name))
       run_test(name);
    else begin
       uvm_report_info(get_full_name(), “No test name specified. Running
default test mytest”);
       run_test(“mytest”);
    end
    $finish();
end

1%> vsim –c top –do “run 1000ns; checkpoint test.chk; run –all; quit –f”
+TESTNAME=test_1
2%> vsim –c –restore test.chk –do “run –all; quit –f” +TESTNAME=test_2
3%> vsim –c –restore test.chk –do “run –all; quit –f” +TESTNAME=test_n
```

Restoring simulation state and modifying test config value to run different tests

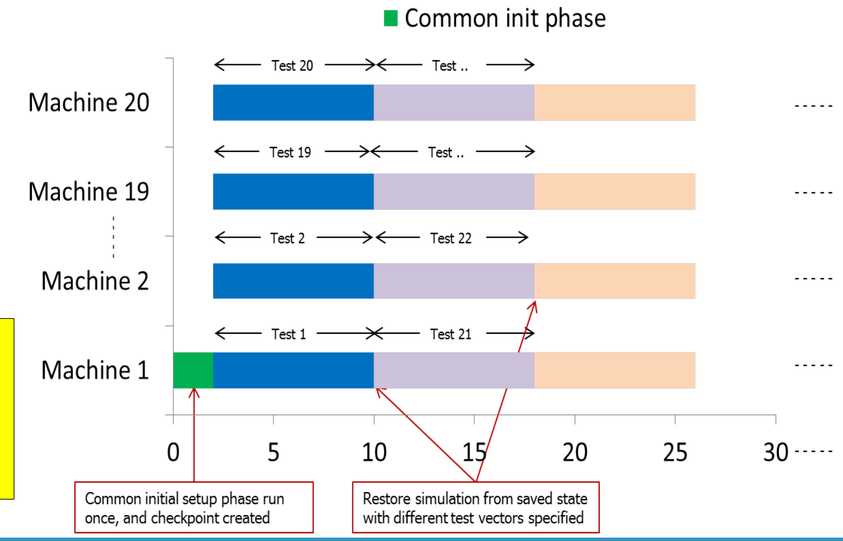
```
string seq_name = “test_1”
my_test_init_seq init_seq;
my_base_test_seq test_seq;
initial begin
    init_seq = my_test_init_seq::type_id::create(“initialization”);
    init_seq.start(sqr);
    checkpoint_enable = 1; //Set variable to trigger checkpoint command
    #1;
    uvm_report_info(get_full_name(), $sformatf(“Restored simulation with test_seq =
%s”, seq_name));
factory.set_type_override_by_name(“my_base_test_seq”,seq_name);
    test_seq = my_base_test_seq::type_id::create(seq_name);
    test_seq.start(sqr);
end

1%> vsim –c top –do checkpoint.do
2%> vsim –c –restore test.chk –do “change /top/seq_name test_2; run -a; quit –f”
```



Regression of a large SoC, before re-factoring common setup phase of tests

Regression flow after separating the common setup phase that is reused by all tests, resulting in 20% saving on regression time



Rohit K Jain
rohit_jain@mentor.com

Shobana Sudhakar
shobana_sudhakar@mentor.com