Simulation Runtime Optimization of Constrained Random Verification using Machine Learning Algorithms

Sarath Mohan Ambalakkat M.S.
Eldon Nelson M.S. P.E.
The Meeting

• University of Minnesota has a graduate level digital verification course
• Eldon did a guest lecture speaker one day in late 2017
Review 2017 Paper

Improving Constrained Random Testing by Achieving Simulation Design Goals through Target Functions, Rewinding and Dynamic Seed Manipulation
Rewind to last Interval and try new seed

Interval seed did not increase the Objective Function

Interval seed did increase the Objective Function
Thoughts on Stimulus
2017 Paper Deficiencies

- A stimulus path is taken based of an entirely random approach to find solutions for the DUT that are proven to improve coverage (completeness)
- The problem is the “entirely random approach”
- A way to improve the odds of finding stimulus solutions for the DUT?
Quake III Arena Capture the Flag

• 2 versus 2 capture the flag
Capture the Flag 2018

- Google Deepmind project
- An approach for sequences

Rather than training a single agent, we train a population of agents, which learn by playing with each other, providing a diversity of teammates and opponents.

Each agent in the population learns its own internal reward signal, which allows agents to generate their own internal goals, such as capturing a flag. A two-tier optimisation process optimises agents' internal rewards directly for winning, and uses reinforcement learning on the internal rewards to learn the agents' policies.

Agents operate at two timescales, fast and slow, which improves their ability to use memory and generate consistent action sequences.

“Agents operate at two timescales, fast and slow, which improves their ability to use memory and generate consistent action sequences.”

https://deepmind.com/blog/capture-the-flag/
Useful Stimulus Odds Improvement

If a deficiency of the 2017 paper was too large of a state-space and its pure random approach, is there a way to improve the odds of navigating that state-space?
Ambalakkat Master Thesis Paper

For reference only, we will go through this on the next slides.
Generate training set data that contains positive and negative stimulus

Start Simulation

Run Limited Number of Random Iterations

Minimum Number of Training Sets Generated

No

Yes

Train Machine Learning Model

Implemented with TCL Commands
Take the ML model of stimulus and plug it into the simulation to dynamically update the constraints.

- Train Machine Learning Model
- Machine Learning Model
- Identify Output Bin NOT previously hit
- Output Bin NOT hit
- Coverage 100%?
  - No
    - Update Constraints and Drive DUT
  - Yes
    - End Simulation
- Yes
  - Train Machine Learning Model
  - Implemented with TCL Commands
Example DUT (comparator)

```verilog
dut #(parameter width)
  width-1:0 a
  covergroup objective_cg
    coverpoint match;
  endgroup
  width-1:0 b
  clk
  c
endmodule
```
2017 Summary for DUT Width of 5

50% success of finding a combination of stimulus after 4000 iterations
Optimization of the Test Environment using Machine Learning Algorithms

1. Generating the Training Sets
2. Training the Machine Learning Model
3. Updating the Constraints
1. Generating Training Sets (TS)

- Structures used to define the training sets
- Less Error Prone
- Addressing training sets easier
- Function used for generating necessary number of valid training sets and track output bins hit
- `no_of_TS_required` made configurable

```c
function void generate_TS_and_track_hit_bins(
    bit [width-1:0] a,
    bit [width-1:0] b,
    bit [width-1:0] match);

// Load TS with "no_of_TS_required" number of training sets
// [Input=0 and Match = 0] cannot be used for training
if(i<no_of_TS_required && (match!=0))
    begin
        // Generating Training Sets
        TS[i].a = a;
        TS[i].b = b;
        TS[i].match = match;
        TS[i].TS_ready = 1;
        // Tracking Output Bins hit
        OUT_HIT[match] = 1;
        i = i+1;
    end
    else
        begin
            OUT_HIT[match] = 1;
        end
endfunction
```
NAME

tcl-fann – A Tcl extension for Artificial Neural Networks

SYNOPSIS

package require fann

gaul create name ?–sparse connection_rate l –shortcut? layers layer1 layer2 ...

gaul load name filepath

name init ?min_weight max_weight?
2. Training the Machine Learning Model (ANN): Using TCL extension, *fann* for implementing the ANN.

TCL Commands used to define the number of layers, number of neurons per layer, activation functions, connectivity, etc. of the neural network. For example:

```
fann create ANN 2 1 1
ANN function hidden linear
ANN function output linear
ANN trainondata 500000 0 {TS_inputs} {TS_outputs}
```
3. Updating the Constraints: Function to update the constraints, `update_constraint` called every iteration once `beta_ready` is asserted.

- Efficient runtime update of constraints: Recompilation of environment not required after update
- Ideally, coverage should improve every iteration, converging to 100% coverage goal much faster.

```verilog
function void update_constraint(integer beta_value);
    num_inside_queue = ();
    i = 0;
    repeat(2**width)
        if(!(OUT_HIT.exists(i))) begin
            num_inside_queue.push_back(i++/beta_value);
            break;
        end
    else i++;
endfunction
```

```verilog
function void rprint();
    this.randomize() with {
        (num inside num_inside_queue);
    };
    `uvm_info("CR", $sformatf("num is: %d", num), UVM_LOW)
endfunction
```
OPTIMIZATION OF THE TEST ENVIRONMENT USING ANN

Training Machine Learning Model

```
DEBUG stable_count == 10
******************************* START train_ANNN
INFO STATUS : TCL : TRAINING ANN at time: 47 ns
Training Sets:
A0 = 4; B0 = 4; MATCH0 = 4;
A0 = 3; B1 = 3; MATCH0 = 3;
******************************* END train_ANNN
----------------------------- END eval_loop
```

Update Constraints

```
Updating Constraints (using ANN)
******************************* START run_ANNN
INFO STATUS : TCL : RUNNING ANN at time: 60 ns
INFO_STATUS : TCL : Output Bin NOT Hit: 1; Predicted Input to Update
Constraint Queue: 1
******************************* END run_ANNN
UVM_INFO sv/env_pkg.sv(305) @ 61: uvm_test_top [ENV_PKG] AFTER UPDATE
num_inside_queue contain: '{'h1'
UVM_INFO sv/dut.sv(37)@ 61: reporter [dut_if] AFTER drive regs A: 1 B: 1
UVM_INFO sv/env_pkg.sv(29) @ 66: reporter [ENV_PKG] Generate training
sets; Track output bins hit
----------------------------- START eval_loop
DEBUG current simulation time is ctime : 67 ns
INFO STATUS : TCL : LOCAL ACCEPTED seed: 319 at time: 47 ns
INFO STATUS : TCL : 67 ns : GOOD : 50.000000 > 37.500000
----------------------------- END eval_loop
```
### 2017 Experimental Data

<table>
<thead>
<tr>
<th>Width of Comparator</th>
<th>No of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>261</td>
</tr>
<tr>
<td>4 or more</td>
<td>Segmentation Fault</td>
</tr>
</tbody>
</table>

### 2019 Experimental Data Linear Regression Model

<table>
<thead>
<tr>
<th>Width of Comparator</th>
<th>No of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>78</td>
</tr>
<tr>
<td>6</td>
<td>96</td>
</tr>
<tr>
<td>7</td>
<td>144</td>
</tr>
</tbody>
</table>

### 2019 Experimental Data ANN Model

<table>
<thead>
<tr>
<th>Width of Comparator</th>
<th>No of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>96</td>
</tr>
</tbody>
</table>
2017 Experimental Data

<table>
<thead>
<tr>
<th>Width of Comparator</th>
<th>No of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>261</td>
</tr>
<tr>
<td>4 or more</td>
<td>Segmentation Fault</td>
</tr>
</tbody>
</table>

2019 Experimental Data Linear Regression Model

<table>
<thead>
<tr>
<th>Width of Comparator</th>
<th>No of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>78</td>
</tr>
<tr>
<td>6</td>
<td>96</td>
</tr>
<tr>
<td>7</td>
<td>144</td>
</tr>
</tbody>
</table>

2019 Experimental Data ANN Model

<table>
<thead>
<tr>
<th>Width of Comparator</th>
<th>No of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>96</td>
</tr>
</tbody>
</table>

Why does the 2019 approach improve the odds so much?

*never mind this is not a statistical sample, but a discrete data set*
Future Work

• Deeper looks at Machine Learning models that can predict stimulus with state
• Is there a role of Formal to help generate deeper stimulus with this approach?
• Configuration space problem
Conclusion

• Improving the search for interesting stimulus using machine learning techniques

• Promising results for example test case by informing the randomization based off of previous automated learning

• Automated in terms of approach as it is based off of coverage
Questions

Please Vote at
http://vote.dvcon.org
OPTIMIZATION OF THE TEST ENVIRONMENT USING A LINEAR REGRESSION MODEL

- Simulation Results using Linear Regression Model:

1. Generating Training Sets

2. Training Machine Learning Model

3. Update Constraints

4. Converges to 100% coverage goal in 24 iterations as opposed to 261 iterations in prior work.
OPTIMIZATION OF THE TEST ENVIRONMENT USING A LINEAR REGRESSION MODEL

2. Training the Machine Learning Model (Linear Regression Model): TCL library, `math::linearalgebra`, used to solve the linear regression problem

- Efficient techniques used to ‘get’ training set values from SV environment.
- Compute the parameters, \( \beta_a \) and \( \beta_b \) using the training sets.
- Computed parameters along with a flag \( \beta_{\text{ready}} \) indicating that the ML model has been trained, forced on variables in SV environment.

```tcl
proc ::rclass::eval_coeff {} {
    # Get the Training Sets
    set a0 [get top.dif.TS_a_0 -radix decimal]
    set b0 [get top.dif.TS_b_0 -radix decimal]
    set match0 [get top.dif.TS_match_0 -radix decimal]
    # Solve Linear Equation
    set beta_a [math::linearalgebra::solveGauss $a0 $match0]
    set beta_b [math::linearalgebra::solveGauss $b0 $match0]
    # Force Beta Values to SV Environment
    force top.rseed_interface.beta_a $beta_a;
    force top.rseed_interface.beta_b $beta_b;
    force top.rseed_interface.beta_ready 1;
}
```
Updating Constraints

```plaintext
function void update_constraint(integer beta_value);
    num_inside_queue = {};
    i = 0;
    repeat(2**width)
        if(!OUT_HIT.exists(i)) begin
            num_inside_queue.push_back(i+/beta_value);
            break;
        end
    else i++;
endfunction
```
UVM_INFO sv/env_pkg.sv(171) @ 80: reporter@@uvm_sequence_item [ENV_PKG] Updating Constraints
UVM_INFO sv/env_pkg.sv(196) @ 80: reporter@@uvm_sequence_item [ENV_PKG] AFTER UPDATE num_inside_queue contain: '{'h2
UVM_INFO sv/dut.sv(26)@ 80: reporter [dut_if] AFTER drive regs a: 2 b: 2
UVM_INFO sv/env_pkg.sv(28) @ 86: reporter [ENV_PKG] A and B matching; Generate training sets; Track output bins hit

------------------------- START eval_loop
DEBUG current simulation time is ctime: 87 ns
INFO STATUS : TCL : LOCAL ACCEPTED seed: 619 at time: 77 ns
INFO STATUS : TCL : 87 ns : GOOD : 75.000000 > 62.500000
------------------------- END eval_loop

UVM_INFO sv/env_pkg.sv(171) @ 90: reporter@@uvm_sequence_item [ENV_PKG] Updating Constraints
UVM_INFO sv/env_pkg.sv(196) @ 90: reporter@@uvm_sequence_item [ENV_PKG] AFTER UPDATE num_inside_queue contain: '{'h3
UVM_INFO sv/dut.sv(26)@ 90: reporter [dut_if] AFTER drive regs a: 3 b: 3
UVM_INFO sv/env_pkg.sv(28) @ 96: reporter [ENV_PKG] A and B matching; Generate training sets; Track output bins hit

------------------------- START eval_loop
DEBUG current simulation time is ctime: 97 ns
INFO STATUS : TCL : LOCAL ACCEPTED seed: 619 at time: 87 ns
INFO STATUS : TCL : 97 ns : GOOD : 87.500000 > 75.000000
------------------------- END eval_loop