Simulation Runtime Optimization of Constrained Random Verification using Machine Learning Algorithms

Sarath Mohan Ambalakkat M.S.

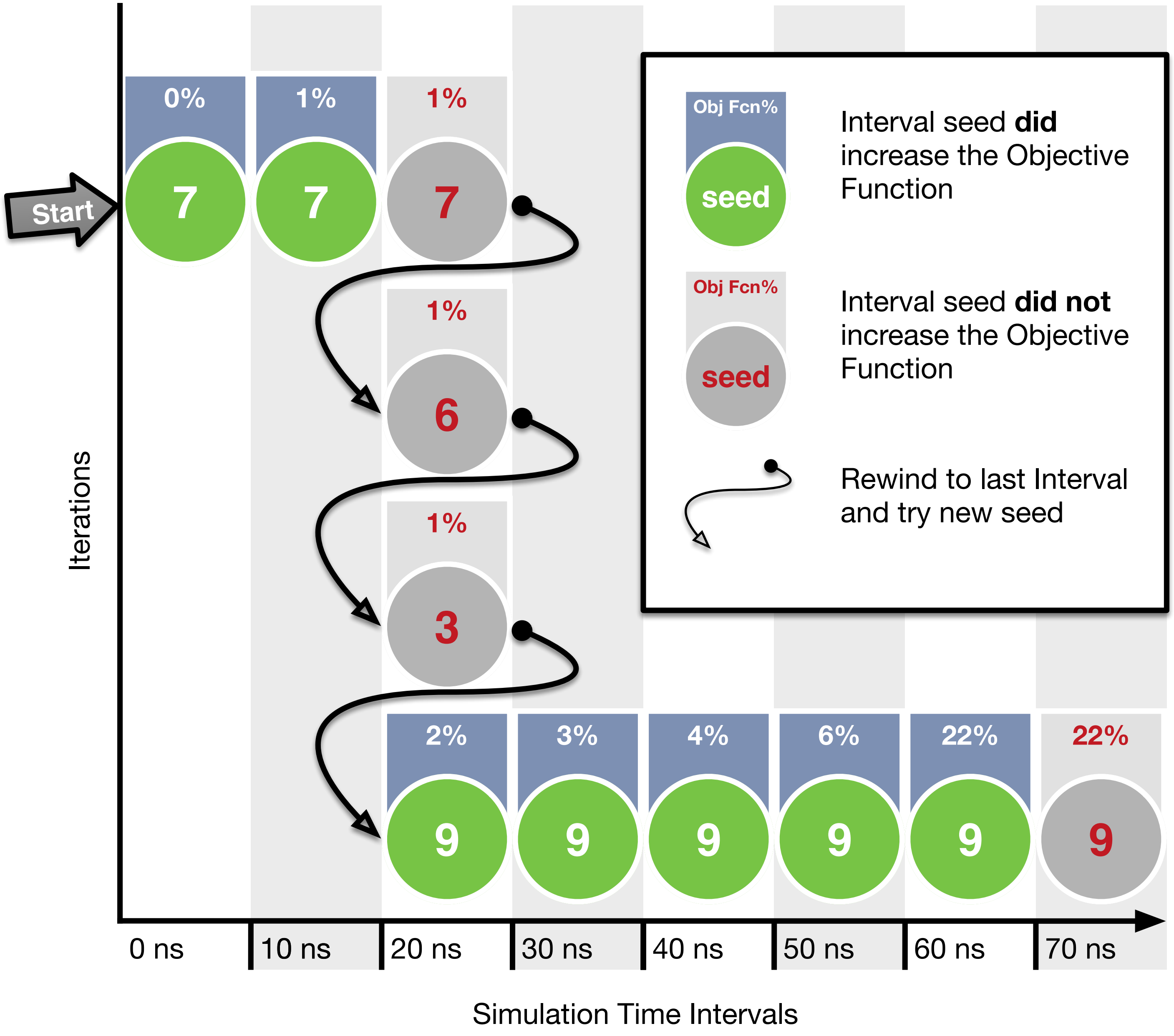
Eldon Nelson M.S. P.E.

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


Start
0%
7
1%
7
1%
7
1%
6
1%
3
2%
9
3%
9
4%
9
6%
9
22%
9
22%
9
Iterations
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
Simulation Time Intervals
Obj Fcn%
seed
Interval seed **did** increase the Objective Function
Obj Fcn%
seed
Interval seed **did not** increase the Objective Function
Rewind to last Interval and try new seed

Thoughts on Stimulus

2017 Paper Deficiencies

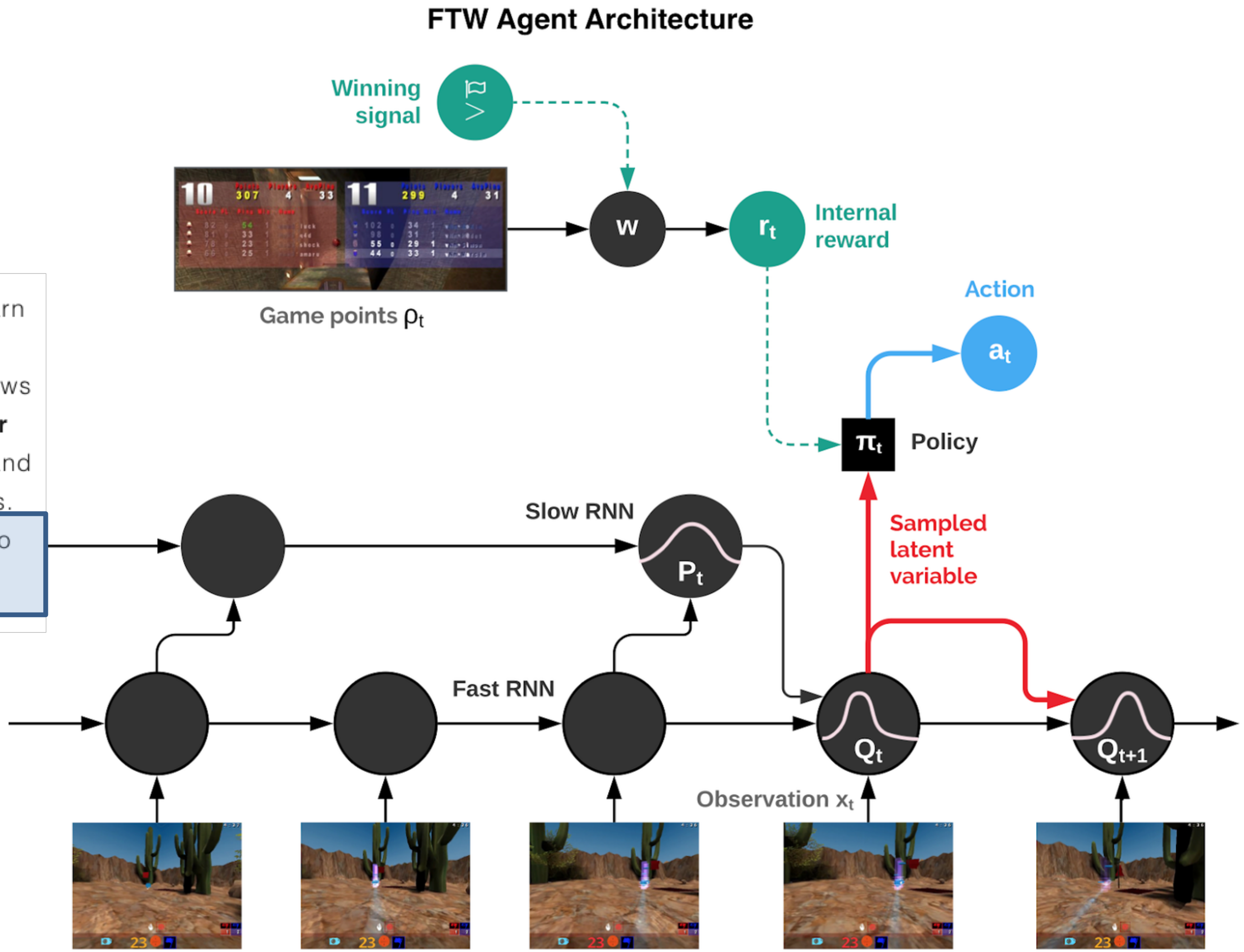
- A stimulus path is taken based of an entirely random approach to find solutions for the DUT that are proven to improve coverage (completeness)
- The problem is the “entirely random approach”
- A way to improve the odds of finding stimulus solutions for the DUT?

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."

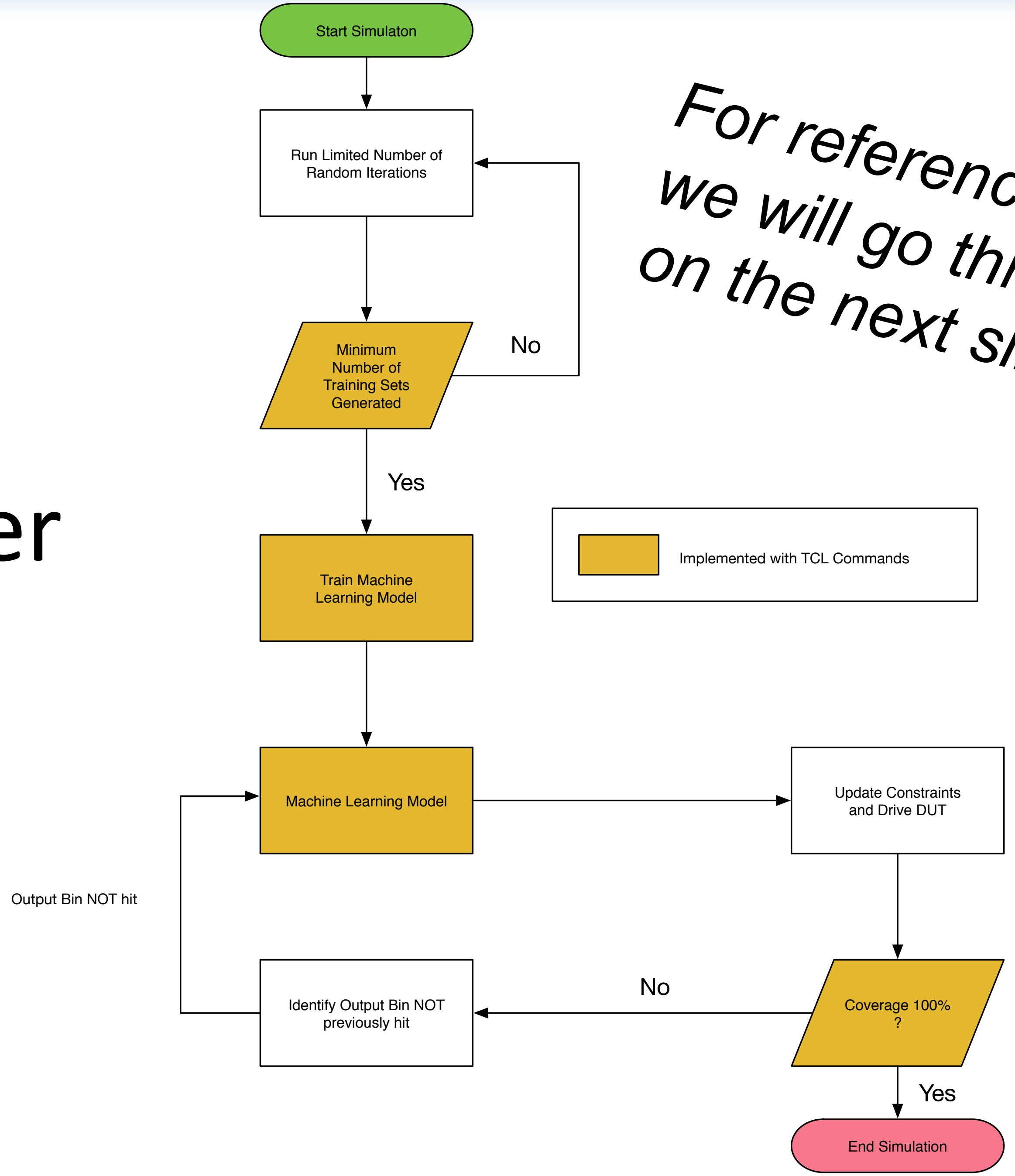
https://deepmind.com/blog/capture-the-flag/

Useful Stimulus Odds Improvement

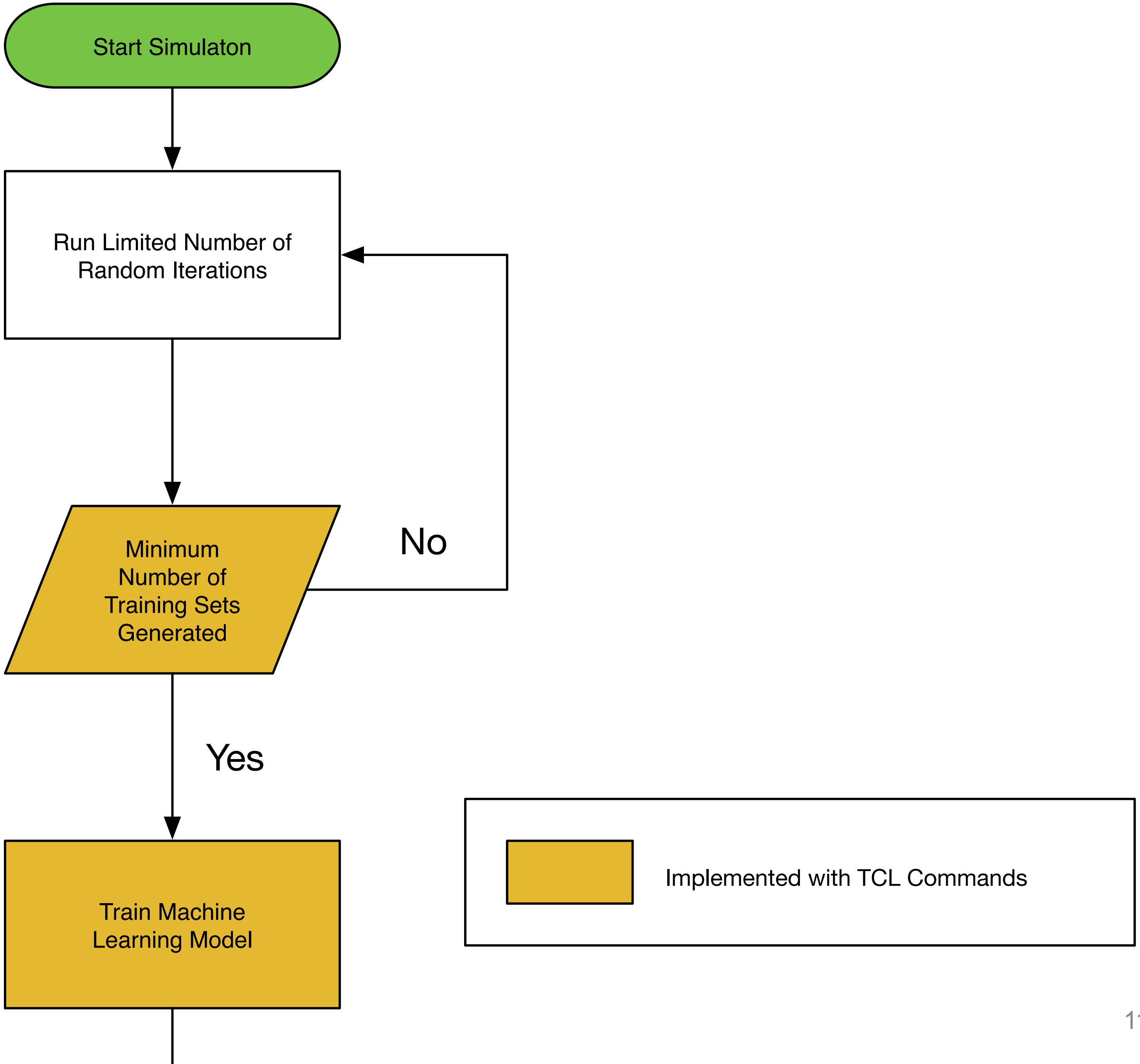
If a deficiency of the 2017 paper was too large of a state-space and its pure random approach, is there a way to improve the odds of navigating that state-space?

Ambalakkat Master Thesis Paper

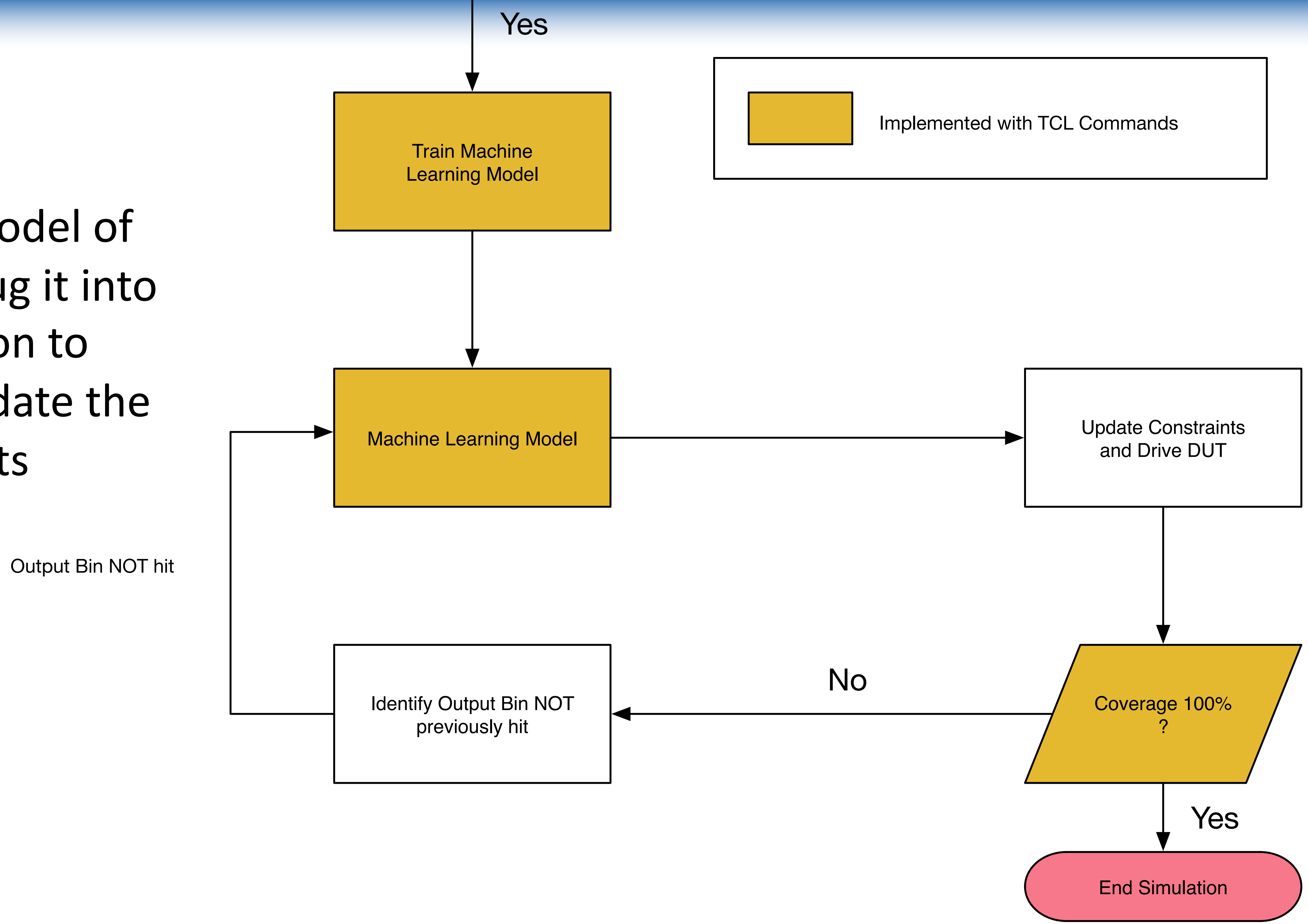
For reference only, we will go through this on the next slides



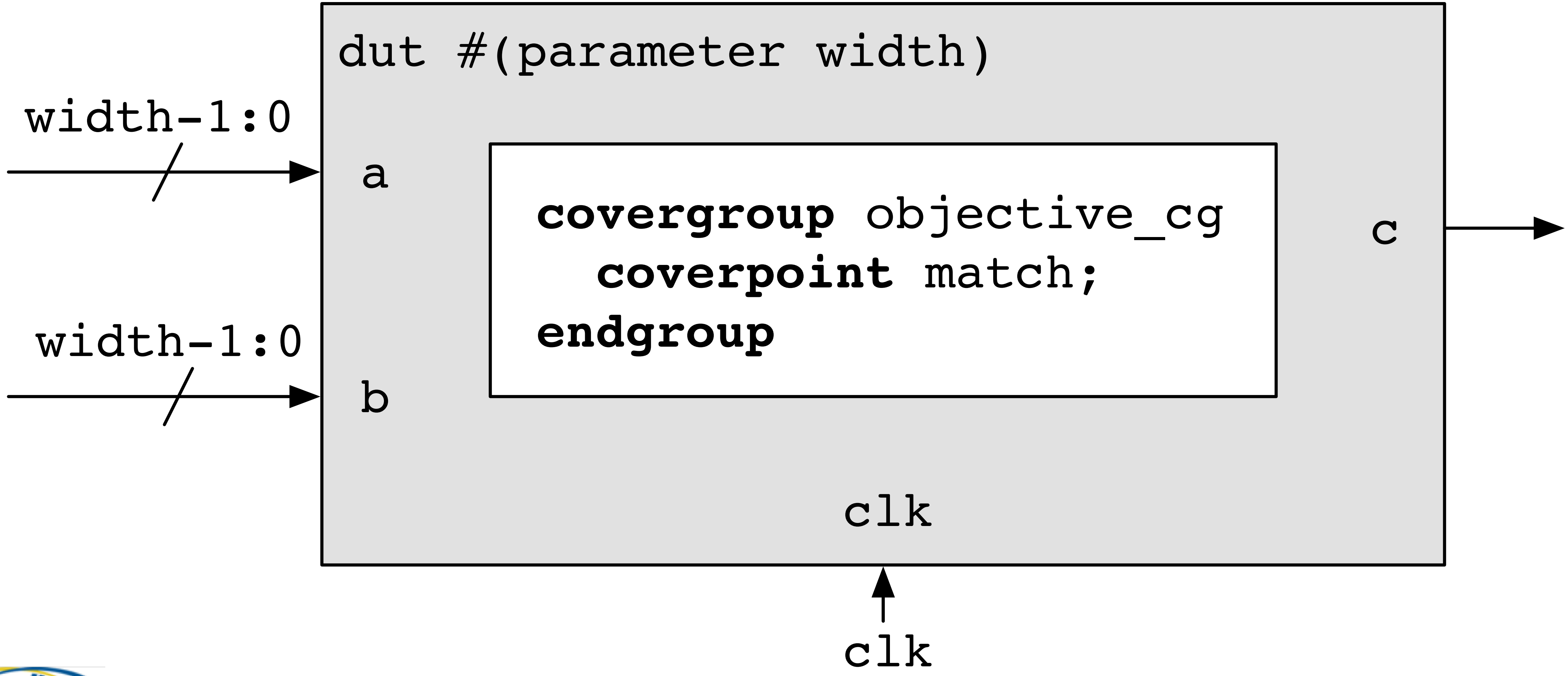
Generate training set data that contains positive and negative stimulus



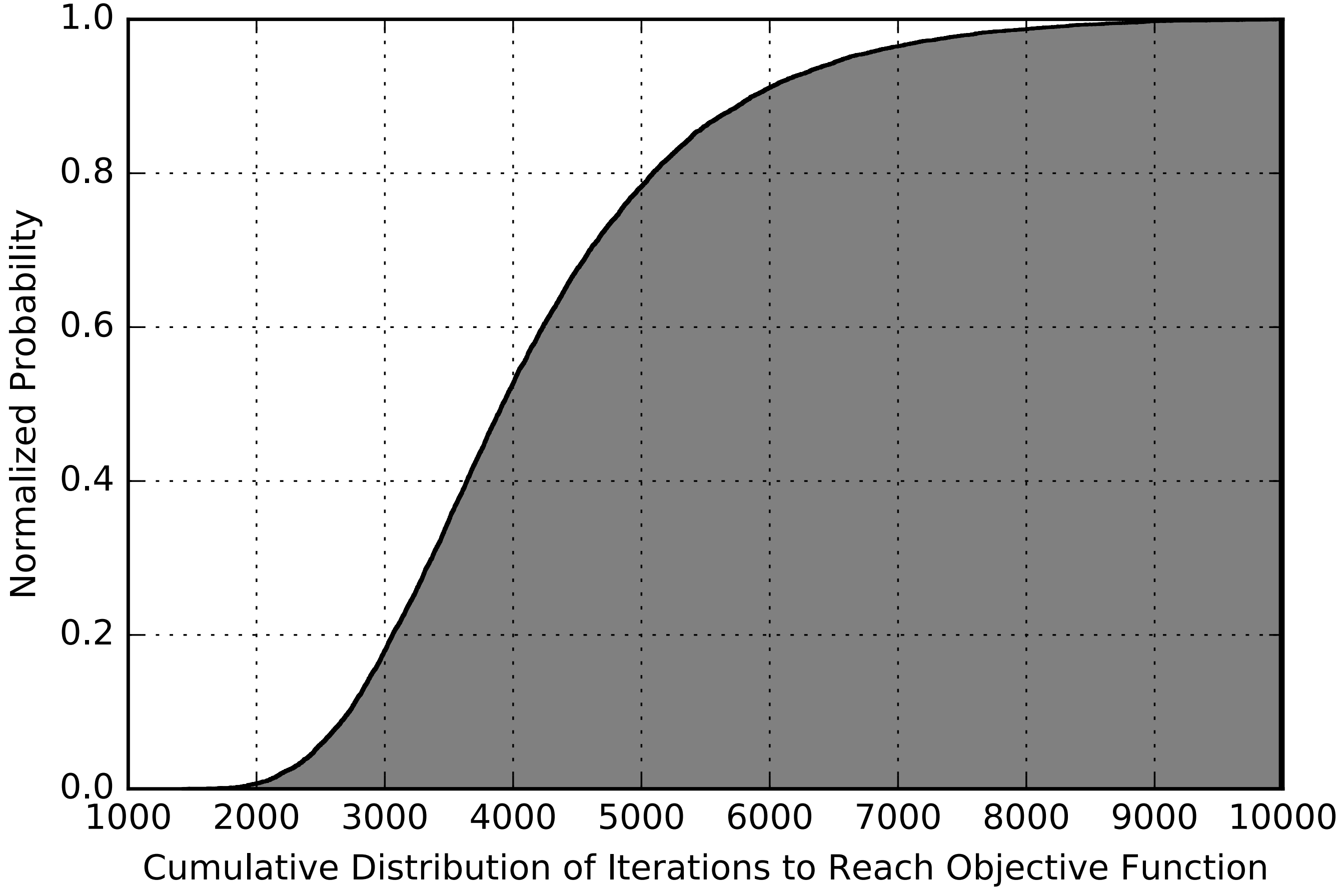
Take the ML model of stimulus and plug it into the simulation to dynamically update the constraints



Example DUT (comparator)



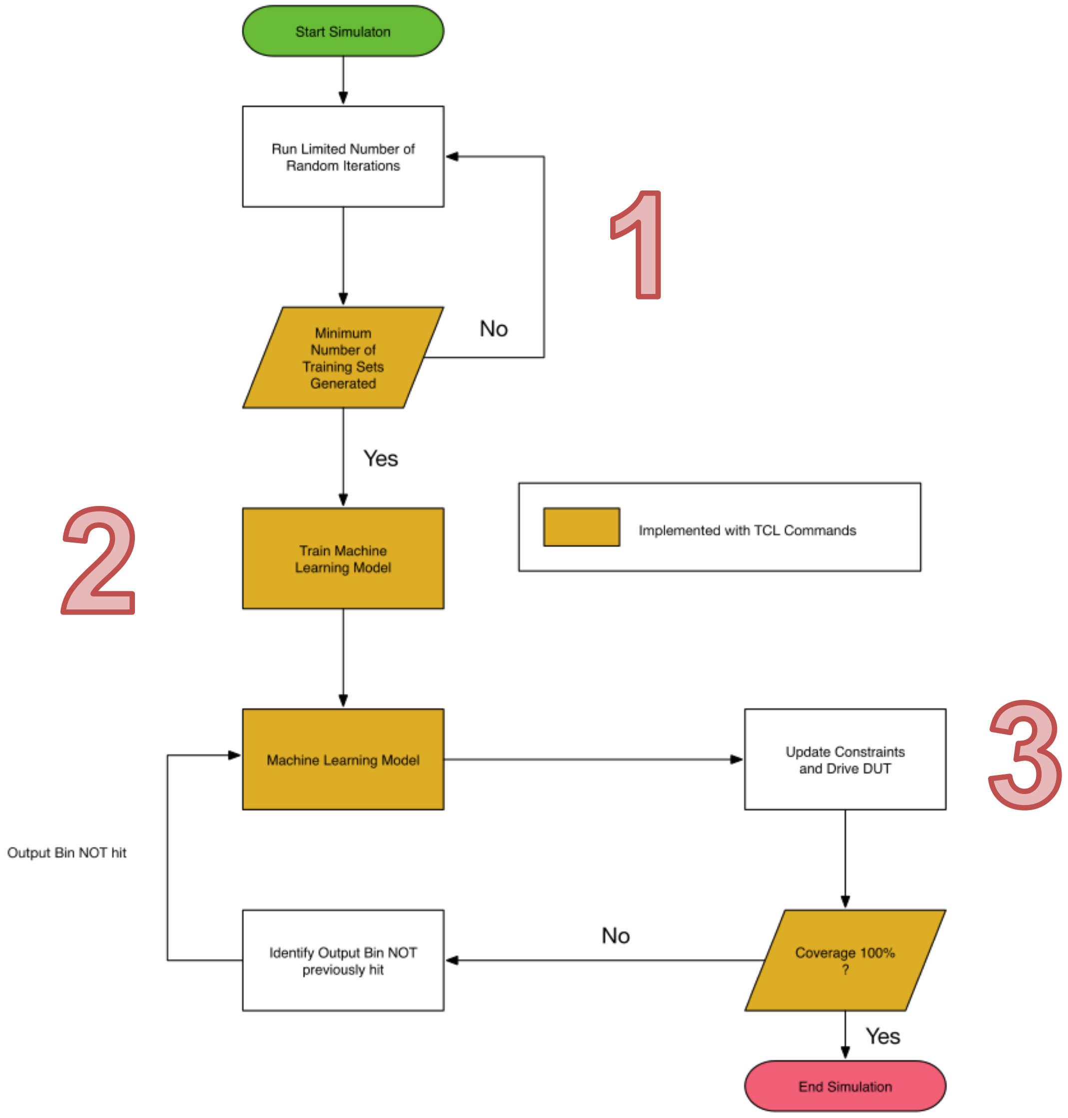
2017 Summary for DUT Width of 5



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



OPTIMIZATION OF THE TEST ENVIRONMENT

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

```
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
```

```
typedef struct {
  bit [width-1:0] a;
  bit [width-1:0] b;
  bit [width-1:0] match;
  bit TS_ready;
} training_set;
```

tcl-fann

NAME

tcl–fann – A Tcl extension for Artificial Neural Networks

SYNOPSIS

package require fann

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

gaul load *name filepath*

name init ?*min_weight max_weight*?

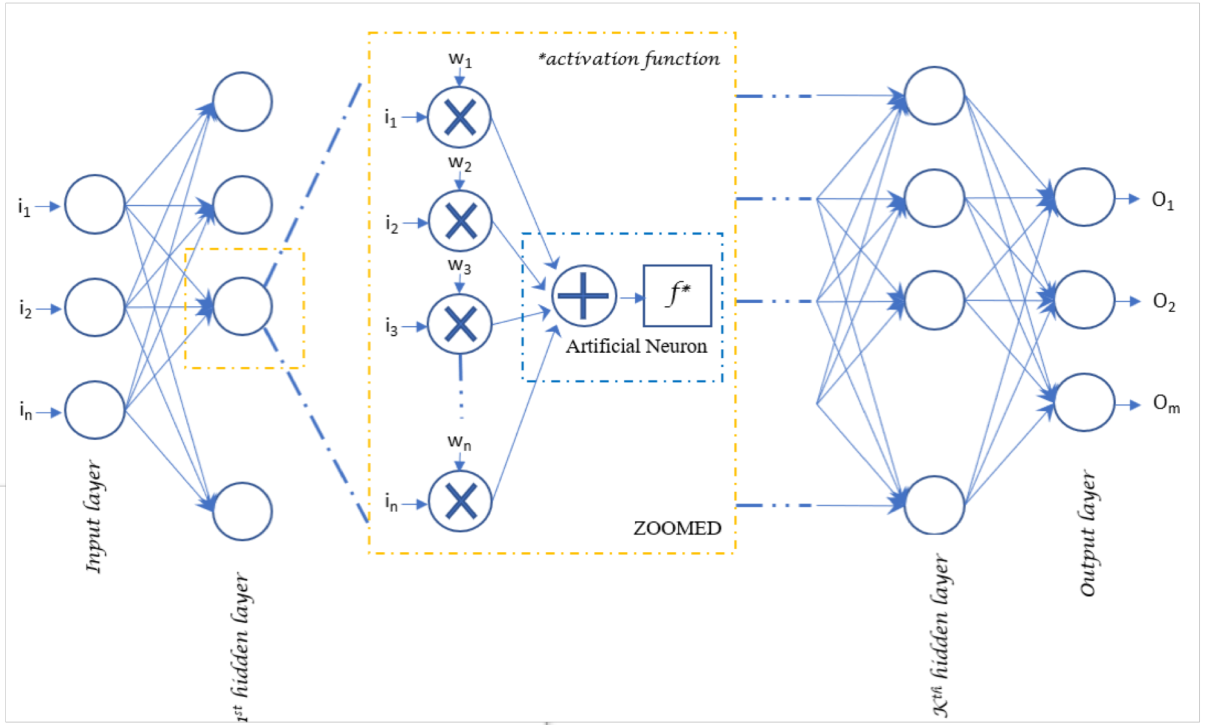
http://tcl-fann.sourceforge.net

OPTIMIZATION OF THE TEST ENVIRONMENT USING ANN

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 traindata 500000 0 {TS_inputs} {TS_outputs}
```



OPTIMIZATION OF THE TEST ENVIRONMENT

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.

```
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
```

```
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_ANN
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_ANN
------------------------- END eval_loop
```

Update Constraints

```
Updating Constraints (using ANN)
************************* START run_ANN
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_ANN
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

Width of Comparator	No of Iterations
1	3
2	60
3	261
4 or more	Segmentation Fault

2019 Experimental Data Linear Regression Model

Width of Comparator	No of Iterations
1	5
2	14
3	24
4	26
5	78
6	96
7	144

2019 Experimental Data ANN Model

Width of Comparator	No of Iterations
1	3
2	16
3	24
4	29
5	96

2017 Experimental Data

Width of Comparator	No of Iterations
1	3
2	60
3	261
4 or more	Segmentation Fault

2019 Experimental Data Linear Regression Model

Width of Comparator	No of Iterations
1	5
2	14
3	24
4	26
5	78
6	96
7	144

2019 Experimental Data ANN Model

Width of Comparator	No of Iterations
1	3
2	16
3	24
4	29
5	96

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

Backup Slides

OPTIMIZATION OF THE TEST ENVIRONMENT USING A LINEAR REGRESSION MODEL

- Simulation Results using Linear Regression Model:

1. Generating Training Sets

```
UVM_INFO sv/dut.sv(26)@ 20: reporter [dut_if] AFTER drive regs A: 4 B: 4
UVM_INFO sv/env_pkg.sv(28) @ 26: reporter [ENV_PKG] A and B matching;
Generate training sets; Track output bins hit
rseed_interface.sv, 111 :              begin
------------------------- START eval_loop
DEBUG current simulation time is ctime : 27 ns
INFO STATUS : TCL : LOCAL ACCEPTED seed: 2 at time: 17 ns
INFO STATUS : TCL : 27 ns : GOOD : 12.500000 > 0.000000
DEBUG stable_count == 0
------------------------- END eval_loop
```

2. Training Machine Learning Model

```
INFO STATUS : TCL : 67 ns : NO PROGRESS : false: 50.000000 > 50.000000
REWINDING TO CHECKPOINT {2} at 57 ns
All the Checkpoints created after checkpoint 2 are removed...
DEBUG stable_count == 10
######################### START eval_coeff
INFO STATUS : TCL : Evaluating Coefficients for LR Model at time: 57 ns
Training Sets:
A0 = 4; MATCH = 4
B0 = 4; MATCH = 4
Evaluate Coefficients:
BETA_A = 1.0; BETA_B = 1.0
######################### END eval_coeff
------------------------- END eval_loop
```

3. Update Constraints

```
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
```

4. Converges to 100% coverage goal in 24 iterations as opposed to 261 iterations in prior w

```
INFO STATUS : TCL : WIDTH OF COMPARATOR = 3
INFO STATUS : TCL : ITERATIONS TOTAL = 24
INFO STATUS : TCL : final_report END
UVM_INFO sv/rseed_interface.sv(143) @ 117: reporter [RS]
COVERAGE GOAL MET coverage:  100    max_objective:  100
```

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_ready` indicating that the ML model has been trained, forced on variables in SV environment.

```
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

```
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
```