A Deep Learning Solution for an Optimized Testbench Regression Launcher

Abhishek Roushan
GPU DV, Hardware Engineering

Abstract

- Attempt to define various metrics directly related to coverage per compute second; an improvement on which furthers the desired **left shift in design verification**.
- A part to the solution to **find breaks early** in a continuous integration system, qualifying the design prior to important milestones and iterating over recent bug fixes or changes to weed out related issues is proposed.
- Results: **Coverage**- existing random test generator vs. a deep learning based test selector applied on top of the generator.
- Types of coverage • cache compression activity: count of compression events in the DUT per test & • Transaction path coverage: number of transactions for specific data path.

Motivation

- Random test run (without bias) fair for unknown DUT bug hot spots. (With bias) Verification solutions today dabble between directed and weighted random testing.
- [*Caveat*] • Directed testing: Narrow coverage scope • Weighted random testing: correct assumptions test (*what*) & (*how long*) to run.
- Long waiting time for simpler bugs if corresponding state space not tried in randoms. **Result**: high compute $/bug.
- Deep learning algorithms: hope to identify and map the test generation knobs and constraints directly responsible for desirable test behavior.

Data Acquisition

Data acquisition consists of following steps:

- Dump compression & transaction path coverage.
- Create repository of data per test: knob setting as inputs and coverage metrics as output
- Massage data (Normalization, Feature scaling, Class balancing) to throttle into neural network

Architecture

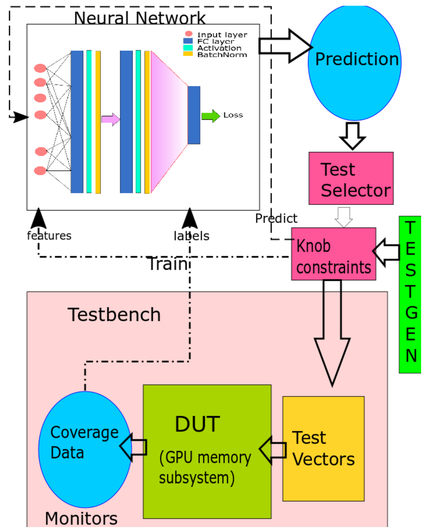

Figure 1: Process flow architecture

Experiment & Results-Cache Compression Coverage

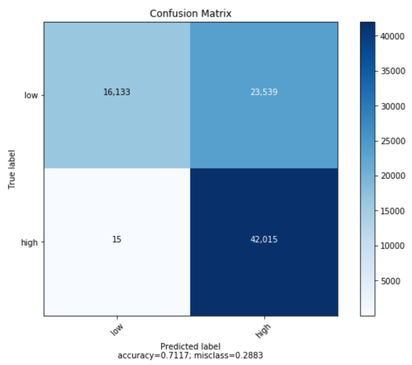

Figure 2: Comparison of true vs predicted coverage

% tests	Coverage overlap	Selected CCS	Random CCS	Improve
2%	44.39%	14.3	0.62	23X
5%	58.69%	7.5	0.62	12X
10%	73.75%	4.8	0.62	7.7X
20%	91.80%	2.92	0.62	4.7X
50%	98.44%	1.28	0.62	2X

Table 1: Improvement DL over random sampling

Experiment & Results-Transaction traffic Coverage

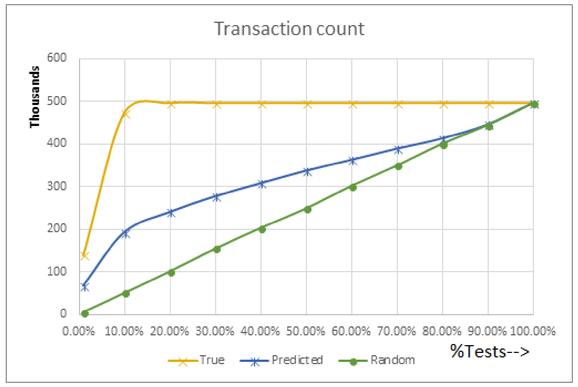

Figure 3: Comparison of True/Pred/Random transaction count single path

Figure 4: True/Pred improvement over Random across top tests

Each transaction path pair frequency for top 1% representative tests is shown in the figure below, which separates high transaction traffic path from low ones.

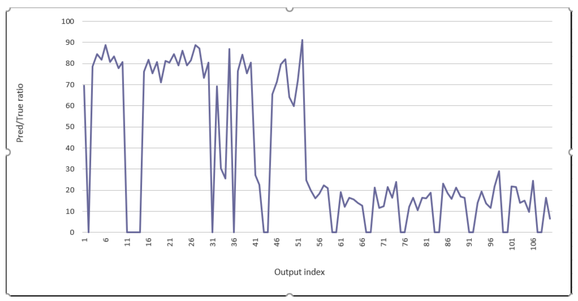

Figure 5: Transaction count Pred/True each source-dest pair

Conclusion & Future Work

Using custom coverage metrics, we determined there exists a tangible relationship between the weighted knobs used to generate the test and the desired coverage. We were able to extract

- Top representative tests for maximum coverage
- Grouping coverage metric so as to find the best overall tests

1. Defining verification coverage metrics is subjective and aside from traditional coverage metrics such as line, conditional coverage, we see scope for customized metrics such as traffic monitors, FIFO full/empty status, etc. that would be useful to advocate for or against the efficacy of any given test.
2. Collecting such coverage information can at times be compute intensive so cheaper alternatives allowing us to integrate this into the periodic bug hunting regressions would be key.
3. The current network is a simple FC-Net, though learning certain specific coverage may be harder and possibly prompt us to migrate to other types of networks such as Binary networks, Recurrent Networks etc.

Related Work

- `https://www.researchgate.net/publication/220306081_Coverage-Directed_Test_Generation_Automated_by_Machine_Learning_-_A_Review` Coverage directed testbench automation
- Automating Design Verification
- `https://dvcon.org/sites/dvcon.org/files/files/2018/06_1.pdf` Deep predictive coverage collection

Contact Information

- Web: nvinfo
- aroushan@nvidia.com