Understanding the effectiveness of your system-level SoC stimulus suite

Alan Hunter, ARM Ltd.

Robert Fredieu, Andreas Meyer, Mentor Graphics

accellera SYSTEMS INITIATIVE™

2014 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Introduction

Understanding how a stimulus suite behaves in a large SoC can be challenging
- Interactions between stimulus and IP
- Interactions between IP components
- Results in changes betweeen expected and actual stimulus behavior
- Randomzation, interactions may not be what was expected or intended

We examine and compare traditional coverage metrics with an in-depth analysis
- Code and functional coverage are proven and well understood metrics
- Provide insight into individual stand-alone components such as state machines

Using new analysis, we examine how well existing coverage portrays actual stimulus effectiveness
- Under-coverage - real RTL interactions which are not reached, but are not detected
- Over-coverage - repeatedly executing the same sequence of events, indiciating randomzation issues

Measurements are based on an SoC with a constrained-random stimulus suite that has achieved full code and functional coverage closure.

We measure how the stimulus interacts with the full SoC to undrestand the effectiveness and efficiency of a full regression suite

Project Overview

Based on a multi-cluster ARM V8 environment.

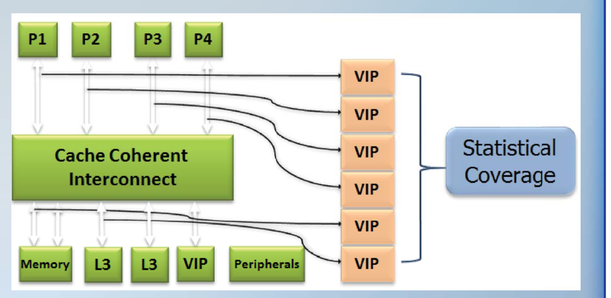
Capture critical staticical transaction information at key interfaces of a design without modifying or intruding on the operation of the SoC.

Gather information for analysis. Correlation of independent transactions captured across the SoC allow us to track and report on full SoC operations.

As an example, single coherent operation may be made up of five to ten individual transactions.

Statistical coverage installed in existing project
- Measure bus, cache operations
- Traffic and interactions between IP blocks
- Running existing stimulus suite

Results are based on an entire regression suite



Distributed Data Capture

Statistical Coverage

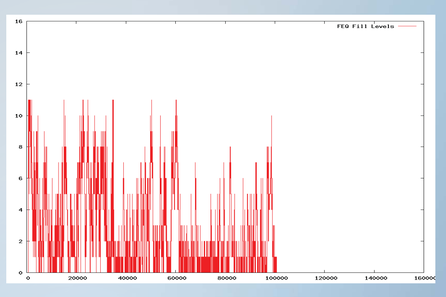
Analysis of groups of correlated transactions
- Distributed in space - across domains, across clusters, across caches, across lines
- Distributed in time - buffered and cached transactions

Examine pattenerns and holes within correlated transactions

Statistical coverage points come from any number of indvidual transactions that are correlated through protocol or architectural interactions.

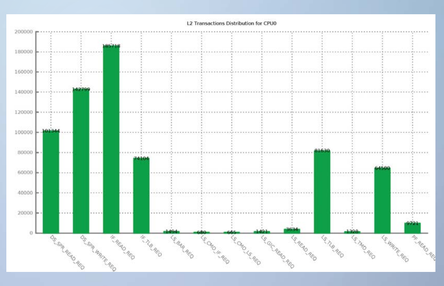
One suprising result was how straightforward counts showed new information

A plot of transaction density over time highlighted a simple constraint error specifying test completion



Transaction Density Over Time

A plot of cache operations over many tests shows an uneven distribution that was unexpected. New constraints were needed to improve coverage



Cache Operation Density

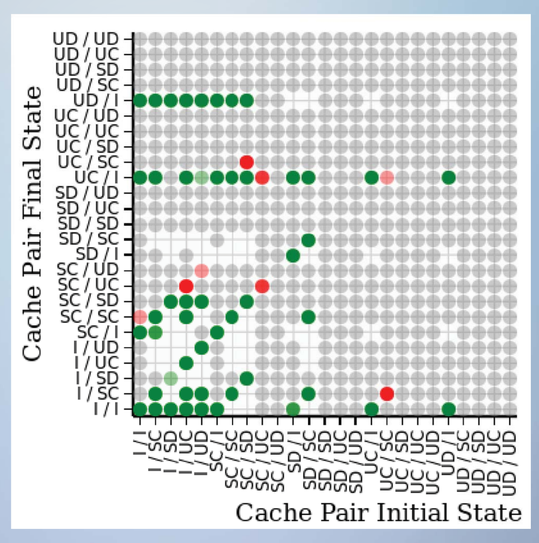
Under Coverage

A direct measurement of system-level coverage requires an examination of statistical coverage over the allowable range of a particular protocol or domain. Specifically, undercoverage occurs when a sequence of events, generally between IP blocks, does not occur as intended

Examples of areas where system-level under coverage may be of interest include QoS and QVN reordering, snoop filtering, and barrier operations forcing the ordering of data/flag operation sets. Note that these types of operations cannot be measured using traditional coverage methods, since they require the use of statistical coverage over disparate transactions.

The example shown below illustrates inter-cache transitions, or how all cache lines interact between all caches in the SoC, including legal, covered, uncovered, and illegal transitions. Note that each datapoint within the plot requires the analysis of many individual transactions, frequently ten or more.

This first plot showed quite a few places where system-level coverage was not reached despite full traditional coverage metrics.



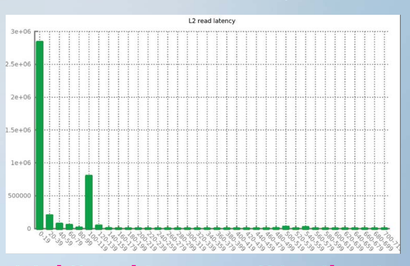
Cache Pair Interaction Distribution

Over Coverage

There are two concerns with over-coverage. First, there is the loss of simulation efficiency. Testing the same pattern of events many times is likely to be a waste of resources. Second, and more importantly, there is the likelyhood that the stimulus was intended to be testing something else, in which case the over-coverage is a measure of the stimulus missing a different target.

A simple example of over-coverage is too much traffic causing one component to saturate. Once that occurs, the timing of future events is controlled by the speed at which that component can issue operations, rather than by the stimulus sources, which no longer have control.

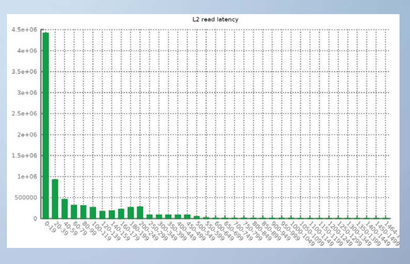
We measured similar, although more complex, interactions in our system. One common approach to achieving higher frequency of cache coherence interactions is to lower the L2 read latency, while simultaneously generations unrealisticly high levels of coherent traffic. This can be effective in creating higher densities of snoop traffic to improve cache coverage.



Initial Read Latency Distribution

However, as we measured, the read latency distribution was too narrow, resulting in overcoverage.

Modifying constraints to have a better match to desired distributions allows for a balance between various system-level testing objectives.



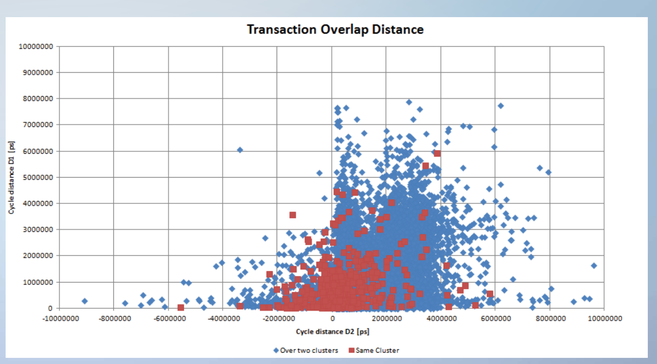
Modified Read Latency Distribution

Analysis and Statistical Coverage

In addition to coverage results, we also examine areas where correct operation is defined by the aggregate of many operations. Examples of this include Quality of Service (QoS), traffic shaping, and fairness. In these cases, correctness has more to do with ordering operations depending on traffic loads. Traditionally, counters have been used to provide a rough estimate of performance.

Using statistical coverage, we are able to provide detailed analytical results, and compare between groups of traffic to see relative performance differences. As an example the plot below shows two groups of data traffic (red and blue), and the relative performance differences between them. In this case, the red group shows significantly better performance characteristics, both for cache hits, generally indicated by the vertical trend, and cache missess, generally indicated by the 45 degree trend.

By examining patterns in dynamic data sets, we can ensure that the stimulus is effective in creating required data interactions, and the SoC is correctly ordering transactions to hit performance targets.



Coherent Transaction Overlap Timing

Conclusions

We show how statistical coverage methods allow us to improve our understanding of the activity within a complex ARM processor SoC, and better understand how constrained random stimulus actually interacs with the SoC, and where improvements can be made.

While existing code and functional coverage methods provide critical metrics for understanding stimulus effectiveness, they are not sufficient when dealing with complex interacting IPs that are frequently found in today's SoCs.

We defined under coverage to occur where the stimulus was intending to create interactions between IP blocks that either didn't occur, or incurred infrequently over a given number of regression runs. We found numerous cases of under coverage in a system that had already achieved full code and functional coverage results.

We defined over coverage to occur when we saw far more of one operation than of all others. We believe this indicated a areas where system interactions resulted in constraints not causing the intended distributions . In the examples we examined, the correlation between constraints and resulting distributions were non-linear.

With the use of over and under coverage analysis, we were able to quantify the effectiveness of the stimulus, which let us modify constraints to achieve measurably better results.

ARM®

References

[1] A. Meyer, H. Foster, "Metrics in SoC Verification: Not just for coverage anymore" DVCon 2013
[2] A. Efody, "Wiretap your SOC: Why scattering verification IPs throughout your design is a smart thing to" DVCon 2014
[3] M. Peryer, "Caching in on Analysis" Verification Horizons, October 2013, Vol 9, Issue 3.

Mentor Graphics®