Using AI for Performance Verification of High-End Processors

Raviv Gal, Alex Goldin, Wesam Ibraheem, Yehuda Naveh
{ravivg,alexgo,wesam,naveh}@il.ibm.com
IBM Research - Haifa

ABSTRACT

We present results of performing analytics and visualizations over micro-architectural performance metrics collected in simulation of high-end processor designs. These results contribute to several use-cases: Obtain fast alerts in cases of anomalous behavior of the design, create a global view of performance-related coverage, and compare different versions of the hardware model as an aid to identification of root-causes of performance differences and correlations between metrics. We show case our methods and results through experiments on a very-high-end processor design, and discuss how they are expected to affect the methodology of performance verification of next-generation designs from the vendor.

ACM Reference Format:
Raviv Gal, Alex Goldin, Wesam Ibraheem, Yehuda Naveh. 2019. Using AI for Performance Verification of High-End Processors. In *Tenth ACM/SPEC International Conference on Performance Engineering Companion (ICPE '19 Companion), April 7–11, 2019, Mumbai, India.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3302541.3311964

1 INTRODUCTION

In the last two decades, a thorough methodology, and vast engineering efforts, are used to verify modern high-end processors. These extensive investments have enabled highly complex processors to be delivered to the market with virtually no serious functional bugs. However, pre-silicon methods for performance verification are still sporadic and largely ad-hoc [1–3].

Pre-silicon performance verification is generally done today in three different ways [3]. 1) Short test snippets to check specific mechanisms. 2) Larger tests executed on the actual hardware model or a reference model. 3) Performance sensors implemented within the hardware design model, and large benchmarks, run to check their measurements as compared to pre-defined minimal/maximal expected values.

In this paper, we present a new methodology for performance verification of high-end processors, that can be applied to the previous methods. It improves their effectiveness by adding analytics and visualizations on data collected during the execution of the test cases. The main enabler for this methodology is having data-emitting sensors instrumented into the design. The emitted data is collected, and then manipulations are applied over it. Our methods fall into two categories. First, generic analytical methods to identify areas that were potentially harmed during the evolution of the design. And second, visual techniques to help the human expert in finding the relevant information from the large amount of data, in order to reach meaningful conclusions. We aim to identify and investigate unexplained changes in the behavior of the design, even if at this stage and at the current level of testing, their effect on IPC is minor. These changes, if unnoticed, may interact with further changes, to significantly affect the IPC at later stages.

This application of combined techniques should result in an earlier identification of performance issues, ultimately leading to better-performing processors released to the market, at faster timelines and reduced costs to the vendor.

2 METHODOLOGY

We insert between several hundreds and a few thousand performance-related sensors into the processor design. The sensors typically count and accumulate measurements and emit their final value at the end of the simulation. Sensors can be largely grouped into three categories: `latency` sensors, sensors identifying `hits/misses`, and `utilization` sensors. For latency and utilization sensors, at least three numbers per sensor are emitted at the end of simulation, namely the minimal, maximal, and average values the sensor measured throughout simulation. Hit/miss sensors end up emitting the total accumulated number of hits and misses. For each sensor we also assign a value indicating whether it is measuring a **good**, **bad**, or **unclassified** property. Measuring a larger number for a *good* property, or a smaller number for a *bad* property, indicates that the design got better. Utilization sensors are many times of type *unclassified* because higher utilization may mean on one hand that the micro-architectural resource is indeed being utilized to do its job, or, on the other hand, that we reached a congestion level which can critically deteriorate performance.

Once executed, a workload can hit any of our sensors. Thus, with around one thousand workloads and one thousand sensors, our results provide a set of one million data points per execution, and hundreds of millions of points collected throughout the design cycle. Unlike existing methodologies, our main focus is not on the end result, but rather on low-level performance characteristics. We find that this focus can give a much richer insight into the behavior of the design compared to observing IPC alone.

3 USE CASES AND RESULTS

In this section we describe the main use-cases we study, and present results aimed to help verification engineers in the process of performance verification. In what follows, we refer to those engineers as our 'users'. It is important to point out at this point, that while our approach is gaining a lot of interest among verification engineers, our results are still preliminary. Most of the experiments were done on previous versions of the designs. The tools we present will be integrated in the verification work-flow, at which point we can present more 'real-life' results, and present bugs that they helped prevent or solve.

ICPE '19 Companion, April 7–11, 2019, Mumbai, India
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6286-3/19/04.
https://doi.org/10.1145/3302541.3311964

3.1 Model comparison

The first use-case we describe is model comparison, where two design models are compared against each other. Typical instances of this use-case are when comparing two model versions before and after a design fix, when experimenting with a presumable performance-beneficial change of design, or when comparing a latest design drop to a gold standard. In all those situations, one needs to gain a fast and holistic understanding of the overall effects the changes between models have created in terms of the design behavior. Our so-called **A-B** (A minus B) feature is designed to give this quick and holistic view.

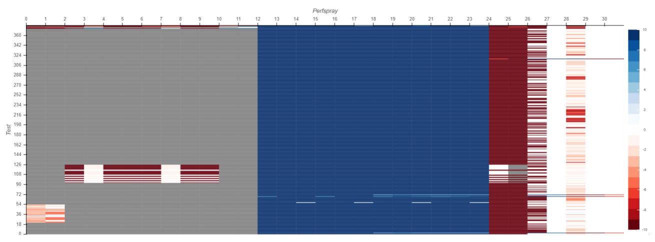

Figure 1: Model Comparison (A-B)

Figure 1 shows an example of this use case. The x and y axes represent sensor and workload numbers respectively. The color coding signifies improvements (blue) or degradation (red) of sensor measurements over design model B compared to model A. Grey cells indicate a missing data point in (at least) one of the models. This generates a heatmap that is easy to inspect for fluctuations in design performance. Here, sensor #12 to #23 show improvement in the new model, while sensors #24 and #25 are suffering performance degradation. Hovering over the cells can show detailed information for the diffs. Users need to inspect and understand these behaviors.

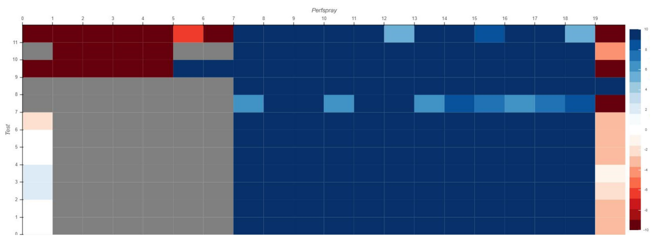

Figure 2: Filtered Model Comparison

The tool also allows users to define minimum diff percentage to show. In figure 2, diffs in the range [-50%,50%] are filtered out.

3.2 Coverage

One important aspect that is not paid enough attention when conducting performance verification is that of coverage. It is true that we would like our entire set of tests to have the best performance possible, and in particular have the best values possible for each of the performance sensors. However, in order to check as many as possible real-life scenarios that the sensor might be in, we need to have a test suite that helps us obtain coverage data over the entire set of possible values for the sensors, both good and bad ones.

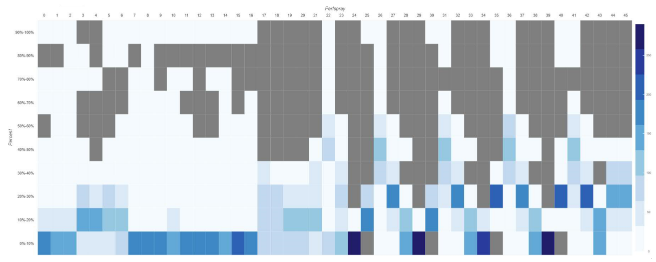

Figure 3: High Level Coverage

Figure 3 shows such a view of performance coverage. Here again, the x-axis depicts sensor numbers. However, the y-axis depicts percentage buckets of the measurement values for the specific sensor (the highest value obtained for the sensor is 100% on the y-axis). Shades of blue signify the number of times each percentage bucket was hit in the entire test suite. Grey cells mean that the relevant cell was not hit by any workload. In this example we see that most of the sensors have low hit rates at their higher values, while some have been reasonably distributed.

When a set of values was not covered by our set of workloads, it is desirable to design new workloads to stress those areas as well. For this and other reasons we also present coverage view of any specific sensor. We will not show it here for space considerations.

3.3 More Use Cases

We are in the process of implementing and testing more visualization use cases. In addition, there are some analytic methods that we already tested and have promising results. These include: **Trend analysis** - studying and analyizing performance trends in the design over time, **Anomaly detection** - Finding sensors that behave in an anomalous way for some runs, and **Clustering** - grouping the testcases into groups that have similar behavior on almost all sensors (this data can serve for many purposes, including anomaly detection). The results of all these use-cases will be presented in a full paper once completed.

4 CONCLUSIONS

We have presented a set of analytical methods, both data- and visualy-oriented, for analyzing performance of high-end processors designs before cast in silicon. The combination of massive gathering of data, deep analytical methods, expert inputs, and human perception of pictures, is the winning path for making good sense of what goes on in the hardware designs we are investigating. Our methods allow for fast detection of performance bugs, even before they have any observable effect on IPC.

REFERENCES

[1] Jim Holt, Jaideep Dastidar, and David Lindberg. 2010. System-level Performance Verification of Multicore Systems-on-Chip. In *Proceedings of the 10th International Workshop on Microprocessor Test and Verification (MTV). https://doi.org/10.1109/MTV.2009.10.*
[2] K. Richter, M. Jersak, and R. Ernst. 2003. A formal approach to MpSoC performance verification. https://doi.org/10.1109/MC.2003.1193230.
[3] M. Srinivas, B. Sinharoy, et al. 2011. IBM POWER7 performance modeling, verification, and evaluation. In *IBM Journal of Research and Development 55, 3 (2011), 1-1.*