A Methodology for Performance Verification of Microprocessors

Yongwen Wang(✉), Libo Huang, and Zhong Zheng

School of Computer, National University of Defense Technology, Changsha 410073, China
yongwen@nudt.edu.cn

Abstract. The tested performance of a microprocessor chip is more important than the predicted performance of it's model. However, performance deviations are often introduced during the design stages. In order to identify and fix the performance defects, a hierarchical performance verification methodology is proposed. Parameter sensitive performance models and coverage driven stimulus are built at the unit-level. Implementation oriented performance calibration and RTL simulation based benchmarks are made at the core-level. Prototyping and counter-based performance analysis systems are built in the system level. An example is given to demonstrate the application and effectiveness of the proposed methodology.

Keywords: Performance verification · Performance model · Microprocessor · Methodology

1 Introduction

The industry has already been able to predict the clock frequency and the functionality of microprocessors in the design stages. However, it is still a great challenge to predict the performance on real programs. Which causes a gap between the performance predicted and the performance tested on the final silicon. In order to narrow this gap and make the real performance consistent with the prediction, performance verification technology is required.

The idea of performance verification is borrowed from the idea of function verification. Function verification has already been obligate in chip design flow and many methodologies have been studied, while performance verification is much less universal and there are hardly any reference methodology.

Performance models are used to predict the overall performance of a microprocessor in the early design phase. This is done by modeling the latency of instructions, the dependencies between instructions, and the allocation of limited resources in the implementation. However, the performance data reflect the behaviors of the models instead of the real silicon. Calibration technologies have

This work is supported in part by National Natural Science Foundation of China under grants 61170045.

W. Xu et al. (Eds.): NCCET 2016, CCIS 666, pp. 24–31, 2016.
DOI: 10.1007/978-981-10-3159-5_3

been studied to make the performance models more accurate. But it won't make the real silicon more accurate when discrepancies are introduced for a new design.

In this paper, a performance verification methodology is proposed to identify and fix the performance discrepancies during the design stages. The rest of the paper is organized as follows: in Sect. 2 related works are analyzed. The proposed methodology is described in Sect. 3. An example is demonstrated to show the application of the methodology in Sect. 4. Finally, this paper is concluded in Sect. 5.

2 Related Works

Performance verification focuses on the performance related discrepancies introduced during the design stage. Thus performance evaluation is very important for performance verification. Many technologies have been studied on these topics in recent decades. Related works can be classified into two categories: performance modeling and benchmarking.

2.1 Performance Modeling

There are two types of performance models, analysis based and simulation based. Analysis based model is created by means of probability theory, queuing theory, Markov model, Petri nets or other mathematical tools. They are commonly used for system-level performance analysis. The degree of accuracy required for microprocessor architecture analysis is beyond the ability of general mathematical models, only a few researchers have attempted to establish a analysis based microprocessor model [1–3]. We have also studied the use of Petri nets for design space exploration [4].

Simulation based model is well known as simulator, which is the main stream method of micro-architecture research. A variety of simulators have been developed successfully [5–8] and surveyed widely [9,10].

In order to make simulators more accurate, a target processor should be chosen. In another word, the simulator should be compared with the target processor which is accurate. Black calibrated the simulator with PowerPC 604 [11], Desikan compared the simulator with Alpha 21264 [12]. There are not many studies on model calibration, partly because performance model is good enough for most academic studies and there are no requirements for a real target processor.

Performance evaluated on models only reflect the behaviors of a simulator. If the simulation model behaves differently to the target processor, the performance improvements may not be achieved on real processors. This is the challenge faced by many microprocessor developers.

2.2 Benchmarking

Benchmarks should represent the behaviors of real-world applications. But it is difficult to decide which application to use for performance evaluation.

Common benchmarks are standard programs for performance evaluation. Some comprehensive benchmarks are derived from real-world applications, which are usually large in scale and long in run time, such as the SPEC CPU, SPLASH, PARSEC and so on. Synthetic benchmarks are based on statistical analysis of frequently used programs, such as Dhrystone, CoreMark and Whetstone. Kernels are extracted from the scientific computing loop code, such as for Linpack and Livermoore. These programs have been widely used and reviewed [13,14].

In addition to the common benchmark suites, researchers will need to develop their own specific testing programs, such as the alpha, beta, gamma and a series of test programs for PowerPC [11] and Microbench testing suites for Godson [3].

However, both benchmarks and specific testing programs can be executed only after the system environment can execute programs. When processor being designed is not complete or correct enough to execute programs, the testing programs can not be used. But when the programs are be evaluated on the designed processor, it might be more difficult to modify the design when performance bottlenecks are found. So it is challenging to evaluate the performance when benchmarks can not be executed.

In summary, most related works are studied by means of simulator and is not responsible for the final processor chip. The lack of effective performance testing and analyzing methodology makes it difficult to narrow the gap between the real performance and expected performance.

3 Methodology

In this paper, we present the idea of performance verification to identify and fix the performance discrepancies introduced during design stages. The methodology is hierarchical in accordance with the microprocessor design flow, including unit-level, core-level and system-level, as shown in Fig. 1.

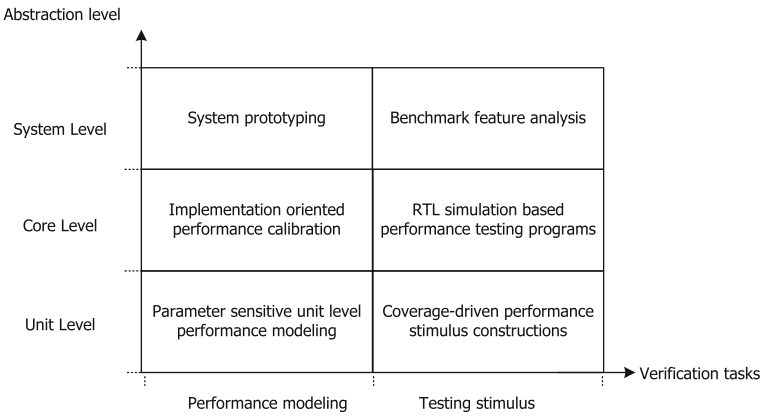

Fig. 1. Hierarchical performance verification methodology

3.1 Unit-Level: Parameter Sensitive Modeling and Coverage Driven Test Cases Construction

Performance modeling is important for the constituent units of the processor. According to the bottom-up design flow, higher-level performance evaluation can not begin until the lower-level units are functionally correct. Unit-level design are usually small in scale, which makes it more efficient to debug than higher-level. If performance bottlenecks escape from the unit-level to the system level, debug costs will be increased dramatically.

The performance of a microprocessor can be broken-down and represented by different performance parameters. Units of microprocessor should focus on the abstracted performance parameters. The performance might not be as detailed as the RTL model, however, the parameters concerned must be abstracted.

Functional units are not complex enough to execute instructions, and benchmarks can not be used for unit-level performance verification. It is difficult to build efficient stimulus to quickly sensitized the performance bottlenecks. In order to assess the quality of test stimulus, the concept of property coverage is introduced. Performance characteristics to be sensitized are defined as cover points. By coverage metrics, we can quantify the effectiveness of performance verification.

3.2 Core-Level: Implementation-Oriented Calibration and RTL Simulation Based Benchmarks

This paper does not mean how to develop a new performance model, but means how to calibrate a simulator to the target processor.

The metrics must be defined for performance calibration. The execution time of the program will give a final performance result, but only the final results are not enough, because internal performance states are not reflected in the final results. If these states appear inconsistent, performance analysis might show the misleading hints.

Approaches to discover and resolve inconsistencies in performance models are called performance debug. Artificial comparison is obviously inefficient, automatical tools should be developed to compare the model with the target processor. Functional verification features might be referenced to accelerate the process of calibration.

When all the RTL units are integrated into a CPU core, a core-level design that can execution instructions is ready, which is the earliest available performance model of the target processor. This core-level RTL implementation has the following features:

- Instruction sequences or programs can be run.
- There are still many function defects.
- There are no peripherals, timers, interrupt controllers, or other system-level components and operating system can not be booted.
- The simulation speed is very slow.
- It's performance behaviors are the same with the target processor.

According to the features above, small-scaled directed testing programs should be developed. The main goal is to sensitize the performance bottlenecks at the instruction execution level.

3.3 System-Level: System Prototyping and Benchmark Analysis

When coming to the system-level design stage, the accelerator or FPGA platform should be ready, which are very close to the target processor. Operating system and real programs can be run on this platform.

3.4 Interactions of Different Abstraction Levels

Each level of the performance verification methodology are not isolated. The co-simulation, reusing and guideline interactive relations are shown in Figs. 2 and 3. Unit-level test stimulus are extracted from the program traces of core-level. And core-level testing programs are derived from system-level benchmarks. Core-level programs can be reused in system-level testing.

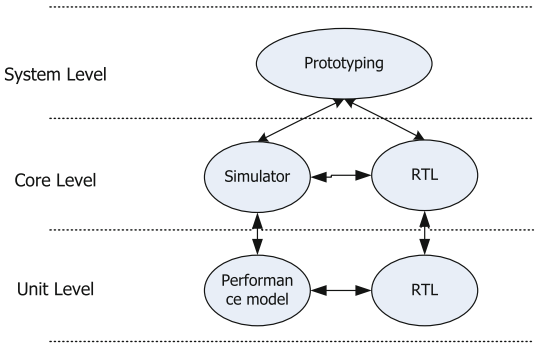

Fig. 2. Performance modeling interaction of different abstraction levels

These interactive relations ensure a consistent view of all performance levels.

4 Practice of the Methodology

In this section, an example will be given to show the practice of the performance verification methodology. The example design is a typical supersalar CPU and the design flow is divided into three stages.

In the unit level, the performance specification is modeled as the reference design and the implementation is verified as the design under test (DUT). Performance features are expressed as properties and cover points are defined for these features. Traces generated from benchmarks are used as performance stimulus. A co-simulation environment is built as shown in Fig. 4 and performance bugs can be identified automatically.

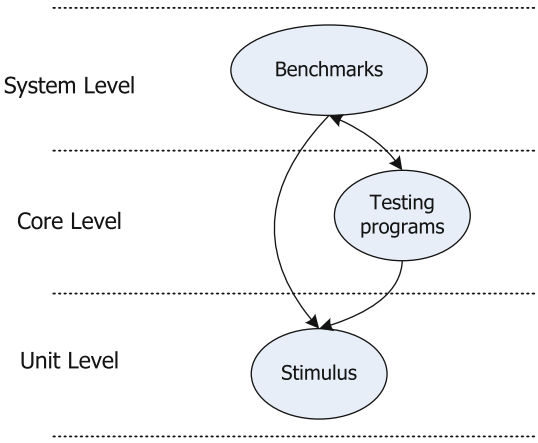

Fig. 3. Performance testing cases interaction of different abstraction levels

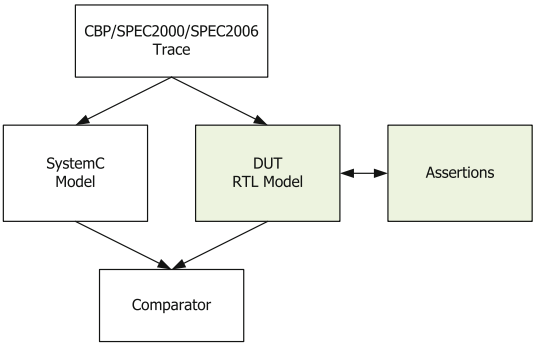

Fig. 4. Unit level performance verification with co-sim

In the core level, some micro-benchmarks are built for RTL simulation. For example, CoreMark is moderated and used as performance stimulus. The functional simulation environment can be shared for performance verification. Figure 5 shows performance testing results of various RTL verions for CoreMark. The performance increases and the miss rate decreases as the performance defects are fixed.

In the system level, FPGA prototyping systems are built and real benchmarks such as SPEC CPU can be run. The performance defects found in each level are summarized in Fig. 6.

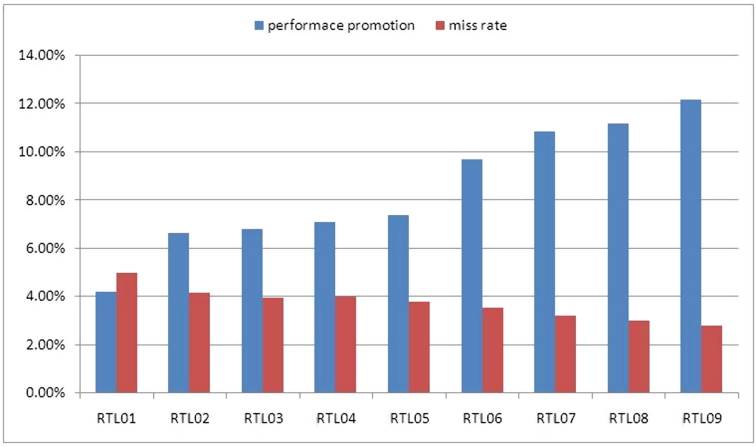

Fig. 5. Core-level performance verification with RTL simulation oriented CoreMark

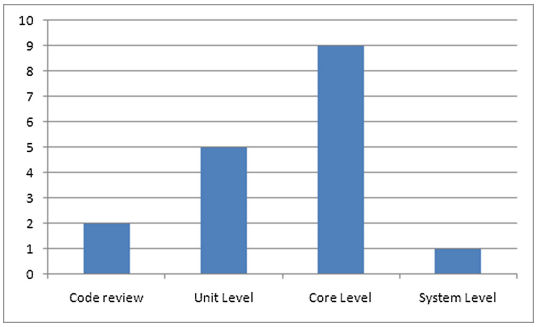

Fig. 6. Performance defects found by performance verification

5 Conclusions

In order to identify and fix the performance defects introduced during the design stages, a hierarchical performance verification methodology is proposed in this paper. Parameter sensitive performance models and coverage driven stimulus are built at the unit-level, implementation oriented performance calibration and RTL simulation based benchmarks are made at the core-level, and prototyping system and counter-based performance analysis are built in the system level. Examples show that the methodology can help fixing performance defects.

We shall refine the methodology and develop automatic tools in future works.

References

1. Noonburg, D.B., Shen, J.P.: A framework for statistical modeling of superscalar processor performance. In: Third International Symposium on High-Performance Computer Architecture, pp. 298–309. IEEE, (1997)
2. Karkhanis, T.S., Smith, J.E.: A first-order superscalar processor model. ACM SIGARCH Comput. Archit. News **32**, 338 (2004)
3. Ma, K.: Microprocessor analytical model. Ph.D. thesis, University of Science and Technology of China (2007)
4. Wang, L., Wang, Y., Li, L., Gao, J., Dou, Q.: Utilizing colored petri nets for design space exploration of asynchronous microprocessors. In: International Workshop on Reachability Problems (RP 2014), Oxford (2014)
5. Austin, T., Larson, E., Ernst, D.: Simplescalar: an infrastructure for computer system modeling. IEEE Comput. **35**, 59–67 (2002)
6. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., et al.: The gem5 simulator. ACM SIGARCH Comput. Archit. News **39**, 1–7 (2011)
7. Virtutech: Virtutech simics - a full system simulator (2004)
8. Zhang, F.X., Zhang, L.B., Hu, W.W.: Sim-Godson: a Godson processor simulator based on simplescalar. Chin. J. Comput. **30**, 68 (2007). Chinese Edition
9. Yi, J.J., Lilja, D.J.: Simulation of computer architectures: simulators, benchmarks, methodologies and recommendations. IEEE Trans. Comput. **55**, 268–280 (2006)
10. John, L.K., Eeckhout, L. (eds.): Performance Evaluation and Benchmarking. CRC Press, New York (2006)
11. Black, B., Shen, J.P.: Calibration of microprocessor performance models. Computer **31**, 59–65 (1998)
12. Desikan, R., Burger, D., Keckler, S.W.: Measuring experimental error in microprocessor simulation. In: Proceedings of the 28th Annual International Symposium on Computer Architecture, ISCA 2001, pp. 266–277. ACM, New York (2001)
13. John, L.K.: Benchmarks. In: Performance Evaluation and Benchmarking. CRC Press, pp. 27–44 (2006)
14. Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach, 3rd edn. Morgan Kaufmann Publisher, San Francisco (2002)