Computer Performance Evaluation Techniques
Position Statement

Antonio González
Department of Computer Architecture
Universitat Politècnica de Catalunya – Barcelona (Spain)

1. Current Approaches to Performance Evaluation of Processors

Computer performance evaluation is a key technology for research in computer architecture. The continuous growth in complexity of computer systems is making this task increasingly complex. In general, the problem of developing effective performance evaluation techniques can be stated as finding the best trade-off between accuracy and speed. This optimal trade-off is dependent on the particular use of the evaluation process.

One of the areas where performance evaluation is becoming especially complex is in processor design. Processors have increased its number of basic components (transistors) by a factor of two about every 18 months. Besides, in the past, the main parameter that designers cared about was execution time and area, but nowadays other performance metrics such as power consumption, power density, thermal characteristics and reliability are key performance metrics.

The most common approach to estimate the performance of a superscalar processor is through building a software model and simulating the execution of a set of benchmarks. Since processors are synchronous systems, these simulators usually work at cycle-level granularity. To estimate execution time, they have a typical slowdown in the order of 10^5 - 10^6, which implies that simulating a single relative small program that takes 1 minute to execute requires about between 1 month and year to simulate. For estimating power consumption this slowdown is further increased. This slowdown is a main barrier for many types of experiments. Besides, the estimation of the delay and power consumption of each component is not trivial. In some cases such as memory structures, relatively accurate analytical models have been developed. However, for some others, a layout synthesis of the corresponding circuit may be necessary.

As a consequence of the above, most new microarchitectural proposals are evaluated by simulating just a small section (100 million – 1 billion instructions) of several (around 10 – 20) benchmarks. This represents in the order of 10 seconds of activity of a real computer, which is certainly an important limitation if one considers the huge variety of applications that are run on computers. Besides, new applications are continuously being developed. This makes the task of choosing a small subset of representative applications to be very difficult, if not impossible.

The other main problem that processor designers are facing is the relative simplicity of the performance evaluation tools. This is a result of the relatively small effort that research teams usually devote to develop performance evaluation tools. In most of the cases, research groups rely on public domain simulators that some groups have made

available, but due to the continuous evolution of processor's microarchitecture, these simulators become soon outdated. A related problem is the difficulty to reproduce results. Even though many groups share the same baseline simulation infrastructure, it is very difficult for a research group to reproduce results generated by another group, since usually this baseline simulator is modified in different ways by the different groups to make it more up to date.

To summarize, the main problems that we are facing in the area of performance evaluation of new microarchitectures are the following: a) the huge slowdown of simulators combined with the difficulty to have a small representative workload, and b) the lack of tools that are adapted to reflect the continuous and fast changes on this area, and that facilitates the reproducibility of results by different research groups.

2. Suggested New Directions

Regarding the first problem outlined in the above section, i.e., the huge slowdown of simulators combined with the difficulty to have a small representative workload, this may be a consequence of a wrong choice of performance metrics. If we accept that we cannot have a small workload (in the order of seconds of execution time due to the typical slowdown of a simulator) that is representative of the dynamic evolving and large variety of programs that are run on a computer, then the conclusions we may obtain just by arbitrarily selecting a tiny section of a small subset of programs may be completely unrepresentative of the reality.

We have a similar analogy in the world of cars. The number of different roads in terms of its riding conditions (slope, bends, type of ground, etc.) is so large that it is really difficult to choose just several of them as representatives of the whole. If car performance was quoted as the driving time, fuel consumption, and other driving metrics for several chosen journeys, these metrics would probably be irrelevant for many users. Instead of that, car performance is reported for a set of synthetic contrived journeys. For instance, metrics that are used are the time to go from 0 to 100 Km/h on a flat, straight road, or the fuel consumption in some predefined conditions that are typical for a journey inside a city, etc.

This approach may also be adequate for evaluating processors. The idea would be to define a set of benchmarks that represent the different possible types of codes that may be run on a computer. For that, we should first do the exercise of identifying which are the different workload parameters that have the most important effect on processor performance. Examples of these parameters are: size of the data and instruction working set, frequency of branches, predictability of branches, length and composition of the dependence chains of some critical instructions such as loads, etc. For instance, one benchmark could be a code that has a large instruction and data working set, many difficult-to-predict branches, does not use linked data structures, etc. We could build synthetic benchmarks that would represent these scenarios and would be relatively small. Because of that, we could consider thousands of different scenarios. For many users, it may be much more informative to know the performance for such benchmarks with well-defined characteristics than to know the performance for some programs that he/she is very unlikely to ever use. Besides, this approach would allow the designer to cover a much larger range of potential scenarios during the evaluation process. This could

resemble some historical attempts to use synthetic benchmarks. However, the main drawback of those attempts was the fact that they just consider a few simple program features in order to generate the benchmarks, such as the instruction mix.

The second problem, which is related to the tool development cost, could be solved by a joint effort of all the microarchitecture community to support the development and maintenance of a set of tools that would be made available to everybody. The effort to continuously update microarchitectural simulators with the new microarchitectural features that better reflect the current state of the art and/or future projections is usually to high for a single research group. This task may be more appropriate for a group that is supported by the whole research community and that is fully devoted to develop, maintain and update these tools to reflect new microarchitecture developments. As certain components become commonly used by the research community (e.g. a new branch predictor), this group will develop new modules that could be plugged to the toolset to simulate these new components. This group could also have the responsibility of developing the set of benchmarks previously described. As a result, we would enjoy a robust evaluation infrastructure, fully tested by an independent group, and continuously upgraded to incorporate new design trends. Besides, this would certainly facilitate the reproducibility of results.