Synchronization Mechanism of Hardware and High-level Models for Performance Verification

Victoria Caparrós Cabezas[1], Andreas C. Döring[1], Hanspeter Ineichen[2]

[1] IBM Research — Zürich
Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
{vca,ado}@ zurich.ibm.com
[2] Emerge Engineering
Brambergstrasse 11, CH-6004 Luzern, Switzerland
hpi@emen.ch

Abstract. This paper presents a mechanism that synchronizes the abstract, high-level simulation of a large-scale, parallel system with the accurate, low-level simulation of the individual components of the system, as part of the performance verification process. This is achieved by driving the hardware model of the component with traces extracted from simulations of the entire system running real-world applications.
The problem with this trace-driven simulation approach is that traces are extracted in the middle of a long run, so the models of the component under evaluation are not in the same state in both simulations. The proposed synchronization method consists of different phases in which the operation of the device is artificially controlled, so that the hardware model progressively reaches a similar state as the corresponding instance in the system simulation and, hence, reliable comparison analyses can be done.
It is shown that the implementation of the mechanism reduces by up to 72 % the differences in the latencies between the two models of the component. The mechanism is applied to the performance evaluation of an Integrated Switch Router (ISR) in the IBM P7 Hub chip, which is a network chip used in PERCS, a system that connects 64K 8-core processor chips.

1 Introduction

Increasing demands of computing capabilities have led to *large-scale parallel systems*, comprising tens of thousands of computing nodes that deliver unprecedented levels of peak performance. Whereas research efforts in that field have traditionally focused on new hardware architectures, interconnection network technologies, power and cooling strategies, or even application development, new challenges arise as the scale and complexity of systems keep growing. One of these challenges is the *simulation of the system* for functional and performance verification.

The simulation phase has gained importance in the overall design process of large-scale parallel systems; detecting hardware bugs or performance misbehavior in the early stages of the design provides an invaluable insight and might have a dramatic impact on the success of the system deployment. Simulating such a large number of computing nodes and their interconnection network, however, requires large amounts of time and computational resources, so existing tools for simulation of systems with thousand of nodes [3] must abstract some details of the system implementation. Performance evaluation of individual components requires a more accurate simulation framework that allows a more precise evaluation of the component (e.g., at VHDL level) and must thus be evaluated separately. This paper

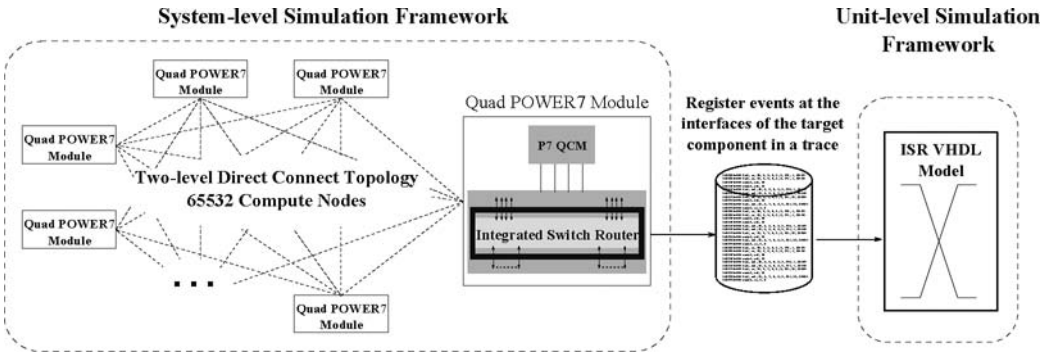

Fig. 1. Overview of the methodology used for coupled system- and unit-level simulation.

describes a system that attempts to couple both system-, high-level simulation with detailed, low-level, single unit evaluation, so that individual components can be accurately tested in a framework that captures the effect of the entire parallel system on the component being evaluated.

Trace-driven simulation is a widely used technique for testing processor or memory subsystems architectures [4], where instructions and data references are recorded in a trace and used to simulate caches or processor cores. We use this simulation approach to test an Integrated Switch Router (ISR) in the IBM Torrent chip, which is a network hub chip used in the the PERCS system, that connects up to tens of thousands of 8-core processors with a two-level routing hierarchy [2]. Traces are extracted from the simulation of the entire system, that tests it under real-world workloads. Network packets and control flow events are recorded in a trace that is processed and driven into the VHDL model of the ISR. The benefits of this simulation approach are twofold. First, the device under test is operated under real-world traffic conditions, as opposed to traditional performance evaluation strategies that consider synthetic traffic based on statistical distributions; such distributions are typically not representative of actual traffic scenarios. Second, traffic patterns contained in the trace capture *second order effects* that reflect the impact of the entire system status on the simulated component (e.g., a possible congestion problem in an element directly connected to the switch in the network); these effects would otherwise have been impossible to accurately model. Figure 1 outlines this methodology.

Ideally, traces contain all the events occurring during the entire execution of a given application. This would however produce intractable amounts of data. Even if storage is not an issue, simulation speed limits trace lengths. Hence, only a subset of the trace can be used to drive the VHDL model of the device under test. The use of trace fragments extracted in the middle of a longer run implies that the switch instance in the high-level simulation has already been routing and transferring data for some time when the trace starts being recorded, while the VHDL model is in its initial state. In this paper, we describe a mechanism that synchronizes the models of the switch in both simulations, so that more reliable measurements can be reported from these simulations.

The main contributions of this paper are:

– Application of trace-driven simulation techniques, traditionally used for processor cores and memory systems evaluation, to the evaluation of a network component in a parallel system.

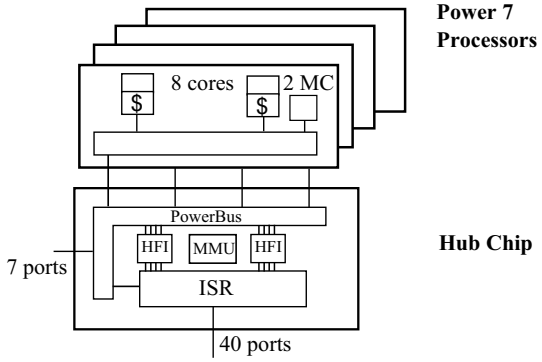

Fig. 2. Node of a PERCS system consisting of the Hub chip and four processor chips

Type	Number	Bandwidth	Extent
HFI	2×4	24 GByte/s	Chip/Connecting Processor
LL	7	24 GByte/s	Board
LR	24	6 GByte/s	Rack
D	16	10 GByte/s	Room

Table 1. Links of the Integrated Switch/Router; Bandwidth is per link per direction

– A synchronization mechanism that gets around the differences in the implementations of the switch router in a high-level simulation and the switch router in the hardware simulation. The implementation of that mechanism shows a reduction of up to 72 % in the differences in the latencies between the two models of the component.

The rest of the paper is organized as follows. The Integrated Switch Router whose performance evaluation is the target of the proposed mechanism is described in Section 2. An overview of the framework that is used to simulate the hardware model of the switch is presented in Section 3, followed by a description of the system-level simulation tools used to extract the traces that will drive the VHDL model (Section 4). Section 5 explains in detail the proposed synchronization mechanism, and the results of applying the method are presented in Section 6. Finally, Section 7 summarizes the main results and points to possible applications for future work.

2 Overview of the Hub Chip and the Integrated Switch Router

The PERCS Hub chip [2] is the main component of the PERCS interconnection network that allows up to 64K POWER7 processors to be connected (Figure 2). Each processor contains 8 processor cores. The largest component of the PERCS Hub chip is the high-throughput 55-port switch (Integrated Switch Router, ISR). The network is connected to the processors chip by two network interfaces: HFIs (Host Fabric Interfaces), supporting among others Send/Receive, Remote Direct Memory Access, and IP multicast. The connection of the HFI to the POWER7 is provided by the split-transaction, high-performance, hierarchically cache-coherent PowerBus. The Hub chip also contains a Memory Management Unit (MMU) for the network interfaces, PCI Express I/O ports, and a Collective Acceleration Unit (CAU).

The ISR connections are listed in Table 1; bandwidth figures are for one direction.

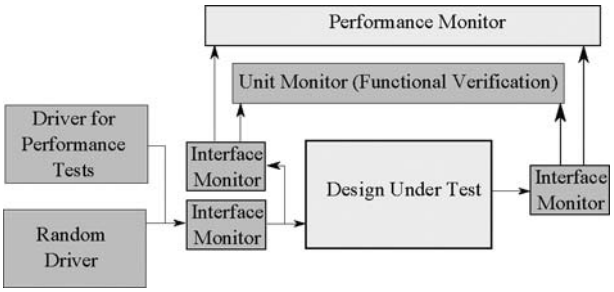

Fig. 3. General Structure of the verification environment for function and performance.

ISR Packets consist of one to 16 flits of 128 bytes (plus error correcting codes). The ISR supports three virtual channels on the D-links and two on the LL and LR links. Each virtual channel has a few dedicated buffers, all other buffers are shared. The interface between the ISR and the HFI appears to the HFI like a memory, and the ownership of buffers is handed back and forth between the two units. The ISR contains different types of routing tables depending on the link or interface they are associated with. Required routing performance is achieved by having several copies for each routing table. Flow control between Hub chips uses credits that can be piggy-packed on regular flits or sent individually.

3 ISR Verification Environment

An important aspect of chip development is verification, which ensures the correct function and reliability of the resulting product. For most non-processor components, performance verification is done only for few cases. For the POWER 7 Hub chip however a much higher effort in performance modeling and verification was spent. The performance verification of the ISR was divided into two parts, one part covering cases with regular traffic patterns for best and worst case performance figures, while the second covered typical performance based on traces. An example for the best case tests is the single stream port-to-port throughput and latency test. Since the ISR has 55 ports, these are in fact a lot of tests (one would expect 55^2 tests, but the number is somewhat lower, because the destination ramp at the HFI interface cannot be controlled).

The verification environment for functionality, written in C++, consists of several layers [5]. The VHDL source is translated into a model which represents the lowest layer. During simulation, the model is handled by a proprietary simulation engine called MESA. The unit interface drivers and monitors attach to this hardware layer. They handle the clock-by-clock interfacing and should be the only components of the verification environment that are connected with hardware signals (Figure 3). For the ISR, there are base classes for link monitors and drivers, from which the classes for each link type (D, LR, and LL) are derived. The credits and packet assembly/disassembly are handled by a so-called channel manager class. Simulation stimuli are generated on higher abstraction levels by separate modules. Furthermore, monitors on a higher level check the expected behavior, for instance that each injected unicast packet leaves the ISR exactly once on the correct output.

For functional verification, test patterns are generated at the packet level, and the channel manager introduces random inter-flit delays. Most of the performance test cases are defined such that they can be done with zero inter-flit-delay and can be generated at the packet level.

```
5.015534e-004 flit, in, 20, 1, 5, 0, 1, 1, 171, 9, 254637
5.015534e-004 flit, in, 21, 1, 1, 0, 1, 1, 179, 9, 254645
5.015607e-004 credit, out, 20
5.015607e-004 credit, out, 21
5.015617e-004 flit, out, 20, 1, 5, 0, 1, 1, 171, 9, 254637
5.015617e-004 flit, out, 21, 1, 1, 0, 1, 1, 179, 9, 254645
5.015672e-004 flit, in, 28, 1, 7, 0, 1, 1, 359, 11, 254824
5.015745e-004 credit, out, 28
5.015755e-004 flit, out, 28, 1, 7, 0, 1, 1, 359, 11, 254824
5.015775e-004 credit, in, 5, 0
```

Fig. 4. Fragment of a trace used for driving the ISR according to traffic in high-level simulation. The trace highlights a flit that enters and leaves the switch, with the associated control flow events.

This allowed the use of the channel manager interface for performance verification which was quite stable during the entire verification process. However, the trace-driven test with larger packets required control of submission time of individual flits and needed extension.

4 PERCS Verification Environment

System-level simulations are performed with two different simulation tools: MARS and Aquarius. MARS [3] is an end-to-end simulation framework that simulates systems with hundreds of thousands of interconnected processors driven by traces of the MPI calls in real-world application software. It consists of a set of pluggable network modules that model the subcomponents of the simulated system and the way they are interconnected. The other simulator employed to extract traces, Aquarius, is a layer of software implemented as a library of C++ classes and procedures which implement all of the necessary structures and operations required to simulate a computer system or subsystem. Thus, a model of a candidate hardware system is assembled from the library of processors and network models and simulated in order to evaluate its performance. The reason to use traces from two different system simulators is that their different internal configuration makes each of them more suitable for running specific workloads and testing different features of the system. So, e.g., while MARS provides a perfect framework to capture communication patterns of MPI tasks, Aquarius is more appropriate to test the GUPS (Giga UPdates per Second [1]) capability of a system, a measurement defined for profiling the memory architecture of a system.

The simulation frameworks are configured to selectively register the information that is needed to reproduce the traffic conditions of the system simulation in the switch router. A short fragment of a trace is shown in Figure 4. It consists of a timed sequence of events (both, network packets and control flow units) with additional information about routing (i.e., source and destination port) and the sizes of the packets.

5 Synchronization Mechanism in Trace-driven Simulation

As mentioned in Section 1, evaluating the performance of a switch with traces extracted from the simulation of the computing nodes and the entire interconnection network of which the switch is a part, poses several challenges. One of them is that traces are taken in the middle of a long simulation, so the switch instances in both simulations present a different internal state when the trace starts, and correlation between performance in both simulation is not reliable. More precisely, these differences are related to:

– **Occupancy of the input and output buffers**: the internal buffers of the VHDL model of the switch are initially empty, as opposed to the switch instance in the system-level simulation, whose input and output buffers already contain a number of packets, since it has been running for a long time before.
– **Available credits for transmission**: the number of available credits in the output links in the hardware simulation is a parameter that, ideally, would be initialized with the number of existing credits in the reference model (i.e., the model in the system-level simulation). However, this number cannot be extracted from the abstract model of the switch in the system simulation, so the available credits in the hardware simulation, by default, are initialized to the maximum value (i.e., the buffer capacity). This assumption, certainly, might not correspond with the status of the switch in the system-level simulation.

The credit control in the transmission interface is not only a problem at the initialization step, but also impacts the accuracy of the rest of the simulation. Since in the hardware simulation no other components are simulated, all the events related to control flow, which heavily depend on the congestion status of the rest of the system, can only be artificially simulated. For example, by default, credits are immediately returned after a flit is transmitted, as if there was no traffic in the rest of the system. In the real system, however, these credits are not received just immediately after the packet is sent. Rather, it takes some time until the next unit signals permission for sending more data, depending on the occupancy of its input buffers. In the best case, the time elapsed between the transmission of a data packet and the reception of a credit from the destination node, is two times the propagation delay for that specific link type. Although it is possible to implement some heuristics that enhance the implementation of the control flow mechanism as, e.g., returning the credits after a random time after the specific link time propagation, this behavior might still not correspond to a typical scenario under which the switch operates.

Figure 5 (a) illustrates the consequences of the problem exposed. In the illustrated scenario, latency and throughput comparisons are not reliable. Hence, in order to use this methodology for performance verification, it is necessary to implement a mechanism that attempts to synchronize both models. To address this challenge we employ a process that consists of three phases in which the simulation is controlled and manipulated until both switches achieve a similar state. The simulation engine only allows driving interface signals, as the internal mechanism implemented by the switch to manage the flow of data units from the input buffers to the output buffers is not controllable from the simulation environment. Thus, the only way to manipulate the operation of the switch is by controlling the number of available credits in the output buffers, which makes the synchronization process more challenging. The credits in the reception interface are not relevant since that interface, by definition of trace-driven simulation, is always synchronized with the high-level model.

Sections 5.1, 5.2 and 5.3, which follow, present a detailed description of the implementation of the different phases during the simulation.

5.1 Warm-up Phase

The warm-up phase is intended to fill the buffers of the VHDL model with the same number of requests as the ones in the reference model. To achieve that synchronization of the buffers, the switch is driven with packets according to the trace, which are internally routed from input to output buffers. The flits in the output buffers are withheld until one of the flits is sent in the corresponding buffer of the high-level simulation, as shown in Figure 5(b). This

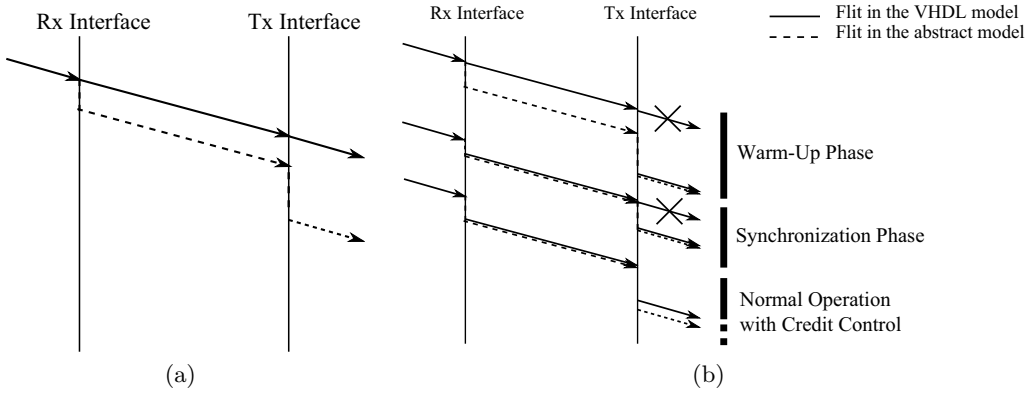

Fig. 5. (a) Time diagram of the *first flit* of the trace in both simulations. The flit in the high-level simulation (dashed line) is temporarily buffered in the input and output buffers due to internal and external control flow. In the VHDL simulation, however, the flit is immediately forwarded from the reception to the transmission interface, and sent to the next destination. (b) Phases of the synchronization mechanism.

is done by assigning zero credits to the output buffers at the beginning of the simulation (i.e., preventing any flit to be transmitted), and inserting one credit in the exact moment when the flit is to be transmitted according to the trace. This process can be considered as waiting for the existing packets in the output buffers of the reference model as a consequence of the previous activity that is not recorded in the trace, to be transmitted. At the end of this phase it can be assumed that the number of data flits in the output buffers of the reference model and in the buffers of the VHDL model is comparable. As a consequence of the synchronization of the internal state of the switches, the return of credits in the reception interface in the hardware model, which is determined by the occupancy of the internal buffers, progressively resembles the corresponding credits in the high-level simulation.

The latency of this first flit is not considered in the computation of the statistics because it does not correspond to the normal operation of the hardware model.

The duration of this phase is port-dependent, and lasts until the first data unit is sent in the respective port.

5.2 Synchronization Phase

If normal simulation continues after the previous phase, the models would rapidly desynchronize because, as mentioned before, the credits in the transmission interface of the hardware model are returned immediately, so the VHDL switch would operate faster than the reference model, whose operation is constrained by the congestion state of the next switch. The synchronization phase attempts to synchronize the credits in the transmission interface. Over this phase, both the instants when the flits are sent and the instants when the credits are returned are controlled according to the information of the trace (Figure 5(b)). The number of credits is initialized to zero, and this variable is incremented every time a credit is received and decreased whenever a packet is transmitted in the trace. After some time, it is reasonable to assume that the number of available credits in the output buffers has been equalized. As in the previous phase, latency and throughput measurements are not collected because the operation of the switch is manipulated from the simulation framework.

There are no constraints on the duration of this phase, so it is determined experimentally, but it should not be too large compared to the trace duration; otherwise, the measurement window will be small and not enough for getting representative measurements.

5.3 Normal Operation with Credit Control

In this phase, output links are initialized with the number of available credits estimated in the previous phase, and the switch is simulated without controlling the instants when the flits are sent. However, in order to maintain the models as synchronized as possible, credits keep being returned to the output buffers according to the credits returned in the trace and not immediately or after a random time, as would be the default configuration in the simulation framework.

This phase extends until the end of the simulation.

6 Results

The implementation of the mechanism presented in the previous section required the addition of some components to the general structure of the verification environment presented in Section 3 (Figure 3). A trace parser was included into the existing framework, which consisted of of 250 lines of C++ code that generated the information required by the rest of the components to drive the requests in the trace into the VHDL model. Overall, the simulations of the ISR unit with long traces took up to 16 hours.

Figure 6 shows how the implementation of this method impacted the comparison of both models. When no synchronization mechanism was implemented, latencies in the trace and in the hardware simulation (first bars) differed in more than 86 %.

If only the Warm-up phase was implemented (column 2), the difference was slightly reduced because the buffers were supposed to be filled with the same number of requests when the simulation starts, but still was not enough to allow a coherent comparison. That evidences the necessity of the second phase, whose implementation considerably improved the synchronization of the systems, as demonstrate columns three and four in Figure 6. These two columns correspond to different durations of the phase, the last one corresponding to a larger synchronization phase. The difference in the average latencies between the two models was reduced to 14 %, i.e., a 72 % improvement with respect to the case when no mechanism was implemented. As it is shown, that the larger the duration of the phase, the better synchronization is achieved.

Nevertheless, it is worth mentioning that the goal of this method is not to obtain the same performance figures than in the high-level simulation (this model *per se* is not an accurate reference), but to enable a reliable comparison of both models and check how much the behavior and the performance of the switches differ. A very large deviation can hint some bug or defect in the model under design.

7 Summary and Conclusion Remarks

Coupling system- and unit-level simulation is particularly beneficial to evaluate components of future large-scale parallel systems, where a system-level simulation cannot be implemented with the degree of detail required to evaluate individual components due to time and computational resources constraint and, therefore, a separate unit-level simulation must

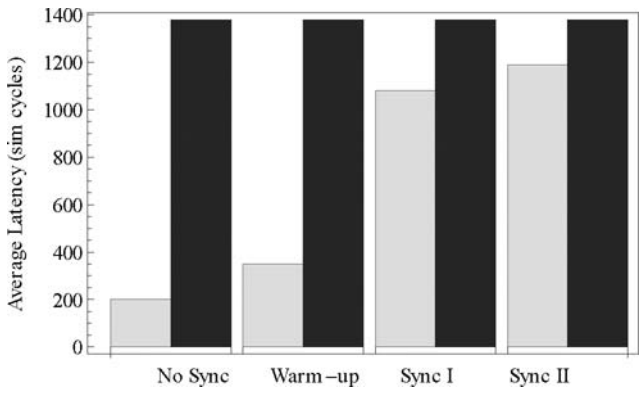

Fig. 6. Results. The light bars correspond to the average latency measured in the hardware simulation, and the dark bars represent the average latency measured from the packets in the trace.

be done to evaluate the performance of the system components with a simulation framework that adjusts to the requirements of this simulation. We introduced a new mechanism that synchronizes the hardware simulation of a network component, with its counterpart in the high-level simulation of the large-scale parallel system that contains that component. The mechanism is based on trace-driven simulation, a technique that traditionally has been applied to processor cores or memory subsystems performance evaluation, but that, for the first time, we use for evaluation of a network switch. It was demonstrated that the implementation of the method reduces the differences between the models of the two simulations by up to 72 %, what improves considerably the confidence on the results and provides a better understanding of the performance of the design. The strategy implemented to get around the differences was the best effort that could be done given the severe and unavoidable differences between the two models. That is, even if all the information about the internal status of the switch in the high-level simulation could be extracted, it makes no sense to directly use this data to initialize the hardware model, because their internal implementation is not the same (e.g., different implementation of the virtual channel assignment).

This analysis provides meaningful information about the model of the device under test, so if the VHDL model's behavior deviates too much from the behavior in the trace, it hints at which level of the implementation one has too look for the cause. Moreover, the detailed evaluation of the individual component can be used to tune back and improve the implementation of such model in the high-level simulation, so that it more accurately represents the real behavior. System-level simulators are important not only to evaluate performance in the design phase of a system, but are also actively used to help customers to port their applications to the target system.

The methodology presented in this paper was applied to the evaluation the ISR in the IBM P7 Hub chip, but currently there are IBM chips which architecture is defined in the high-level, and it is planned to implement this methodology for their hardware evaluation.

Although this method entails an improvement into the simulation process, yet a lot of work in the arena of simulation of large-scale parallel system must still be done, specially with the challenge of exascale systems, were the performance and functionality of systems that are capable to execute trillions of operations per second must be done. New simulation techniques and evaluation approaches must hence be designed to cope with the increasing capabilities of future computing systems.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency under its Agreement No. HR0011-07-9-0002. This work is also supported by the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies. Hanspeter Ineichen was employed by Supercomputing Systems while carrying out work described in this paper at the IBM Research – Zurich Laboratory.

References

1. V. Aggarwal, Y. Sabharwal, R. Garg, and P. Heidelberger. HPCC RandomAccess benchmark for next generation supercomputers. In *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, pages 1–11, Washington, DC, USA, 2009. IEEE Computer Society.
2. B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, J. Li, N. Ni, and R. Rajamony. The PERCS High-Performance Interconnect. In *Proceedings of 18th Symposium on High-Performance Interconnects (Hot Interconnects 2010)*. IEEE, Aug. 2010.
3. W. E. Denzel, J. Li, P. Walker, and Y. Jin. A framework for end-to-end simulation of high-performance computing systems. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools '08, pages 21:1–21:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
4. R. A. Uhlig and T. N. Mudge. Trace-driven memory simulation: a survey. *ACM Comput. Surv.*, 29:128–170, June 1997.
5. B. Wile, J. Goss, and W. Roesner. *Comprehensive Functional Verification: The Complete Industry Cycle (Systems on Silicon)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.