Isidoro Gitler
Jaime Klapp (Eds.)

Communications in Computer and Information Science 595

High Performance Computer Applications

6th International Conference, ISUM 2015
Mexico City, Mexico, March 9–13, 2015
Revised Selected Papers

Springer

A VHDL-Based Modeling of Network Interface Card Buffers: Design and Teaching Methodology

Godofredo R. Garay[1], Andrei Tchernykh[2](✉), Alexander Yu. Drozdov[3], Sergey V. Novikov[3], and Victor E. Vladislavlev[3]

[1] Camagüey University, Camagüey, Cuba
godofredo.garay@reduc.edu.cu
[2] CICESE Research Center, Ensenada, Baja California, Mexico
chernykh@cicese.mx
[3] Moscow Institute of Physics and Technology, Moscow, Russia
alexander.y.drozdov@gmail.com,
serg.v.novikov@gmail.com,
victor.vladislavlev@gmail.com

Abstract. The design of High Performance Computing (HPC) relies to a large extent on simulations to optimize components of such complex systems. A key hardware component of the interconnection network in HPC environments is the Network Interface Card (NIC). In spite of the popularity of simulation-based approaches in the computer architecture domain, few authors have focused on simulators design methodologies. In this paper, we describe the stages of implementing a simulation model to solve a real problem—modeling NIC buffer. We present a general methodology for helping users to build Hardware Description Language (HDL)/SystemC models targeted to fulfil features such as performance evaluation of compute nodes. The developed VHDL model allows reproducibility and can be used as a tool in the area of HPC education.

Keywords: Simulation · Design methodology · VHDL · NIC · Real-Time Calculus

1 Introduction

The design of High Performance Computing (HPC) relies to a large extent on simulations to optimize the various components of such a complex system. Among the key hardware components that determine system performance is the interconnection network. At this level, performance evaluation studies include assessing network interface bandwidth, link bandwidth, network topology, network technology, routing algorithm, deadlock prevention techniques, etc. [1–3]. In particular, NICs play a key role in the performance of HPC applications [4].

Traditionally, multi-drop buses (as considered in this paper) have found many applications in computer systems, particularly in the I/O and memory systems, but signal integrity constraints of high-speed electronics have made multi-drop electrical

I. Gitler and J. Klapp (Eds.): ISUM 2015, CCIS 595, pp. 250–273, 2016.
DOI: 10.1007/978-3-319-32243-8_18

buses infeasible. For this reason, point-to-point links are displacing multi-drop buses, for example, in the migration of PCI/PCI-X to PCI Express. Nevertheless, it should be taken into consideration that the difficulty of creating a multi-drop bus electrically can be avoided using optics; see [5]. Furthermore, the current trend toward virtualized compute platforms has motivated the analysis of I/O resources contention (e.g., at PCI Express level). Conceptually, this problem is similar to the one happening in the former I/O parallel bus generations. For this reason, we consider that our approach can be extended to model such virtualized platforms.

Modeling and simulation techniques have made successful contributions to different areas in industry and academia. However, there are certain key issues that are preventing these approaches from addressing larger domains and from achieving wide-scale impact. One of them is reproducibility. This concept is related to replicability, i.e., the ability to reproduce and, if needed, independently recreate computational models associated with published work [6].

Using a higher level of abstraction for modeling the interconnection network of an HPC system allows the modeling of all important networking aspects, including buffering, without having to resort to even lower abstraction levels that would significantly increase the simulator complexity and simulation runtimes, without resulting in deeper insights. Hardware description languages (HDLs) such as VHDL and Verilog, as well as System-Level Description Languages (e.g., SystemC and SpeC [7]) can be used for this purpose. Using these languages, simulation models can be implemented at different abstraction levels.

A step forward of their design is to find a set of sequential steps that sufficiently governs the development of simulation models of complex compute nodes in HPC environments. Various authors have focused on modeling and design methodologies to support the design process of complex systems. For example, in the field of manufacturing machinery [8], mechatronic systems [9], multiprocessor system-on-chip (MPSoC) [10].

The objective of this paper is to discuss a design methodology, aiming to have an HDL-based simulation model of the network-to-memory data path in a compute node. This model allows us to validate a Real-Time Calculus-based (RTC) model of the data path [11]. RTC is a high-level analysis technique previously proposed for stream-processing hard real time systems. It is frequently used to evaluate trade-offs in packet processing architectures. We focus on computing the backlog bound provided by RTC. In addition, our model determines the impact of specific features of the hardware components of the data path on the FIFO-buffer requirements in a network interface card.

The main contribution of this paper is to illustrate the stages computer architects may follow in implementing a simulation model to solve a real problem—modeling NIC buffer. We present a general methodology for helping users to build Hardware Description Language (HDL) such as VHDL or Verilog, as well as SystemC models targeted to fulfil features such as performance evaluation of compute nodes. The developed VHDL-based simulation model allows reproducibility and can be used as a teaching tool in the area of HPC education, also.

The paper illustrates how we can design a simulation model using hardware description languages to analyze the performance of network interfaces in comparison with alternative ways such as general purpose programming languages, simulation frameworks, and full-system simulators. Also, we give some details about the elements of the VHDL model of the communication data path to be evaluated.

The paper has been organized as follows. Section 2 describes the design phases of the proposed methodology. Section 3 presents our case of study describing NIC buffers simulation model. Section 4 describes the design principals. Section 5 discusses the proposed conceptual model. Section 6 describes inputs parameters, modeling assumptions, and the process definition. Section 7 validates the conceptual model. Section 8 describes our simulation model and presents examples of its applicability. Finally, Sect. 9 presents the conclusion and future work.

2 Design Phases

In this section, we describe eight major phases for the proper simulation model design and implementation [12]. Although these phases are generally applied in sequence, one may need to return to the previous phases due to changes in scope and objectives of the study. In particular, phases 3 through 6 of the process may be repeated for each major alternative studied as part of the project (Fig. 1). We discuss each phase in detail by identifying the steps to follow for its successful execution.

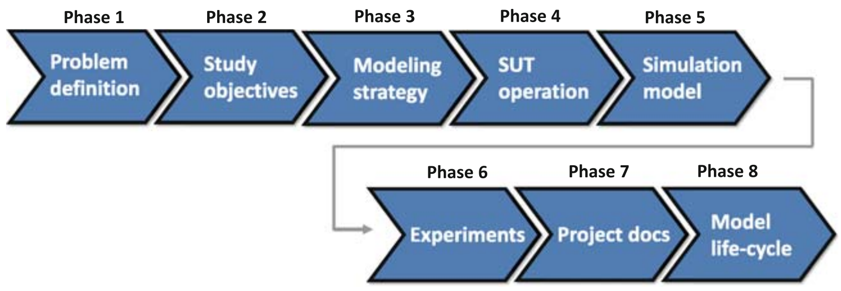

Fig. 1. Flowchart of design phases from problem definition to model implementation (Phases 1–5), and from experimentation to life cycle support (Phases 6–8)

The phase 1 of the simulation process is the problem definition. It has the most effect on the total simulation study since a wrong problem definition can waste a lot of time and money on the project. This phase may include the following steps: (1) define the objectives of the study; (2) list the specific issues to be addressed; (3) determine the boundary or domain of the study; (4) determine the level of detail or proper abstraction level; (5) determine if a simulation model is actually needed and an analytical method work; (6) estimate the required resources needed to do the study; (7) perform a cost-benefit analysis; and (8) create a planning chart of the proposed project; and (9) write a formal proposal.

The objectives of the design study of the phase 2 must be clearly specified by its clients. Once the objectives of the simulation study are finalized, it is important to determine if one model can satisfy all the objectives of the study. This phase may include the following steps: (1) estimate the life cycle of the model; (2) list broad assumptions; (3) estimate the number of models required; (4) determine the animation requirements; (5) select the tool; (6) determine the level of data available and what data is needed; (7) determine the human requirements and skill levels; (8) determine the audience (usually more than one level of management); (9) identify the deliverables; (10) determine the priority of this study in relationship to other studies; (11) set milestone dates; and (12) write the Project Functional Specifications.

The modeling strategy to be used in the study is the phase 3 of the simulation process. Modeling strategy involves making decisions regarding how a system should be represented in terms of the capabilities and elements provided by the chosen simulation tool. The overall strategy should focus on finding a model concept that minimizes the simulation effort while ensuring that all objectives of the project are met and all specific issues are investigated. This phase may contain the following steps: (1) decide on continuous, discrete, or combined modeling; (2) determine the elements that drive the system; (3) determine the entities that should represent the system elements; (4) determine the level of detail needed to describe the system components; (5) determine the graphics requirements of the model; (6) identify the areas that utilize special control logic; and (7) determine how to collect statistics in the model and communicate results to the customer.

The formulation of the model inputs, assumptions, and the process definition is the phase 4 of the process. At this phase, the modeler describes in detail the operating logic of the system and performs data collection and analysis tasks. This phase may include the following steps: (1) specify to operating philosophy of the system; (2) describe the physical constraints of the system; (3) describe the creation and termination of dynamic elements; (4) describe the process in detail; (5) obtain the operation specifications; (6) obtain the material handling specifications; (7) list all the assumptions; (8) analyze the input data; (9) specify the runtime parameters; (10) write the detailed Project Functional Specifications; and (11) validate the conceptual model.

The building of the model, its verification, and operational validation constitute the phase 5 of the simulation process. At this phase, the modeler uses well-known software techniques for model building, verification and validation. This phase may include the following guidelines: beware of the tool limitations; construct flow diagrams as needed; use modular techniques of model building, verification, and validation; reuse existing code as much as possible; make verification runs using deterministic data and trace as needed; use proper naming conventions; use macros as much as possible; use structured programming techniques; document the model code as model is built; walk through the logic or code with the client; set up official model validation meetings; perform input-output validation; calibrate the model, if necessary.

Experimentation with the model and applying the design of experiments techniques constitute the phase 6 of the simulation process. At this phase, the project team may decide to investigate other Alternative Models and go back to the previous phases of the process for each major model change. During this phase, rather than building a design of experiments for the whole study, the modeler generally identifies the major variables and eliminates the insignificant variables one step at a time. Once a few major variables are identified, a design of experiments study may be conducted in detail, especially if the study has a long-life cycle. One of the objectives of this phase is to involve the client as much as possible in the evaluation of the output under different conditions so that the behavioral characteristics of the system as modeled are well understood by the client. The steps of this phase are as follows: (1) make a pilot run to determine warm-up and steady-state periods; (2) identify the major variables by changing one variable at a time for several scenarios; (3) perform design of experiments if needed; (4) build confidence intervals for output data; (5) apply variance reduction techniques whenever possible; (6) build confidence intervals when comparing alternatives; and (7) analyze the results and identify cause and effect relations among input and output variables.

Project documentation and presentation is the phase 7 of the simulation process. Good documentation and presentation play a key role in the success of a simulation study. The tasks of this phase are performed in parallel to the tasks of other phases of the process. The important factors that should be considered at this phase are that (a) different levels of documentation and presentation are generally required for project, (b) long-term use of models requires better model documentation, (c) selling simulation to others require bottom-line measures (e.g., cost savings) to be highlighted as part of the project presentation, and (d) output values from the simulation should be supported with the factors causing those results. This phase may include the following elements of documentation: project Book; documentation of model input, code, and output; project Functional Specifications; user Manual; Maintenance Manual; discussion and explanation of model results; recommendations for further areas of study; and final Project Report and presentation.

The model life-cycle tasks constitute the final phase 8 of the simulation process. This phase applies to long-term simulation life-cycle studies where the simulation models are maintained throughout the life of the real system. On the other hand, short-term simulation studies are those where once the simulation results are used in the decision-making, the model is not used any more by the client. Currently, about sixty to seventy percent of the simulation studies can be categorized as short-term studies while the rest are longer term in the way that they will be reused after the initial simulation study has been completed.

In many cases, long-term life-cycle models are used for multiple purposes including all four categories. The long-term life-cycle models require additional tasks as given in the following steps of this phase: (1) construct user-friendly model input and output interfaces; (2) determine model and training responsibility; (3) establish data integrity and collection procedures; and (4) perform field data validation tests.

3 Case Study: NIC Buffers Simulation Model

For the purposes of demonstrating the design methodology, a case study is presented below. We combine the phases/steps described in the previous section into an explicit and holistic model development approach, which demonstrate the applicability of the methodology from concept to realization.

3.1 The Problem Definition

The objectives of our study are four fold: to find the proper NIC buffer capacities to avoid overflows at card level when considering back-to-back packet reception; to measure various NIC performance indicators such as maximum buffer-fill level, and dropped packets count; to provide the required parameters for calibrating a RTC-based model of the NIC; and to identify the bottleneck operations of the system.

Before specification of all issues, let us give some basic definitions.

NIC is a constituent element of system network interface. A network interface allows a computer system to send and receive packets over a network [13]. It consists of two components:

(1) A network adapter (also known as host interface, Network Interface Card, or NIC), the hardware that connects the network medium with the host I/O bus, moves data, generates communication events, provide protection;
(2) the network software (often, we refer to as the software) on the host that handles application communication requests, implements communication protocols and manages the adapter.

For providing a comprehensive glimpse of network interface functions, Fig. 2 illustrates the components of a network interface in terms of a “Five-layer Internet model” or “TCP/IP protocol suite” as proposed in [14].

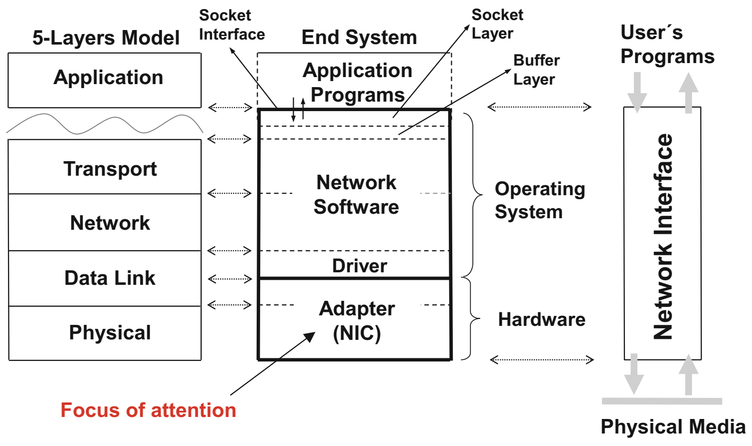

Fig. 2. Focus of attention: NIC functionality and operation.

As you can see, the adapter (hardware component) operates at Data Link layer whereas the software operates at data link, network and transport layer. A part of the data-link layer functions are performed in software by the data-link driver (NIC driver).

Commonly, for packet buffering, a NIC controller incorporates two independent FIFOs for transferring data to/from the system interface and from/to the network. These FIFOs provides temporary storage of data. Also, at this step, the client should specify the minimum and maximum values of the variables to be considered.

We address the following issues: computer organization; PCI bus specifications; Bus arbitration process; Bus access latency; PCI transactions; Latency Timer; Direct Memory Access (DMA) burst size; characteristics of PCI/PCI-X compatible devices; NIC functionality and operation; network workload; network interface; and NIC functionality and operation.

Here, we answer the question: What are the maximum buffer requirements we need to avoid dropping packets at the NIC due to FIFO overflow?

The objectives of the study together with the specific issues to be addressed by the study identify the information required by the simulation model as well as the inputs and components needed by the model in order to generate the required output information. The task of determining which components of the real system to include and exclude from the simulation model requires both insight on how the real system operates and experience in simulation modeling. A good method is to make a list of all components in the real system and identify those needed for the simulation model.

The influence of each component on the other components of the system should be discussed before finalizing the list of components to be included in the model. At this step we keep model boundary at a minimum. It should be extended later in the study only if crucial system behavior cannot be exhibited by the model.

The main components are workload, compute node, NIC, and simulation statistics and control (Fig. 3).

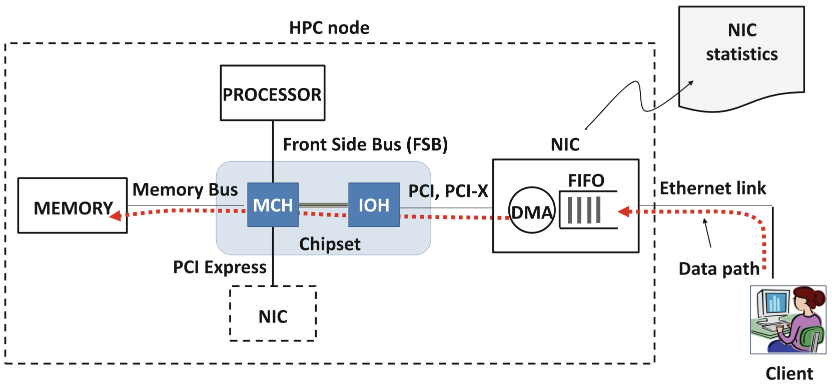

Fig. 3. System under test (SUT).

We concentrate on the following features that impact on the NIC buffer requirements: (1) the network workload, (2) NIC operation, (3) NIC-System interconnect characteristics (e.g., a parallel I/O bus), and (4) memory subsystem operation (e.g., memory controller, memory bus, and memory modules).

3.2 Abstraction Level

The model should include enough information to get confident answers for the specific questions asked from the study. In many cases, the availability of data and time, experience of the modeler, animation requirements, and expectations of the client are more dominant factors in determining the level of detail than the specific issues to be addressed by the study. The objective of the modeler should be to build the highest-level macro model that will satisfy the objectives of the study. Regarding the network workload, we model the worst-case Ethernet traffic, i.e., corner cases consisting of a constant traffic of raw fixed-size frames[1], as well as Jumbo frames (Fig. 4). Here, an accurate packet time and interarrival time should be generated.

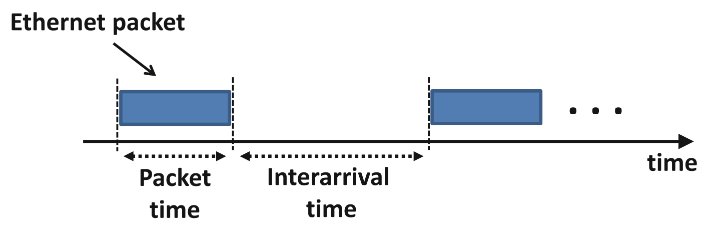

Fig. 4. Network workload parameters.

With respect to the internal NIC operation, we model a store-and-forward network card. The overhead due to NIC-side processing at the received end-system is modeled by using a configurable latency. Commonly, the hardware of the NIC maintains a circular descriptor ring structure for buffer management. The overhead due to NIC-side processing tasks such as packet identification, layer 2/3/4 processing, i.e., CRC Calculation, IP/TCP/UDP checksum calculation, or TCP offloading is modeled by using a configurable latency. Once processing stage has elapsed, a communication task (DMA transfer) of packet payload and the corresponding buffer descriptor (BD) across the I/O bus is performed (Fig. 5).

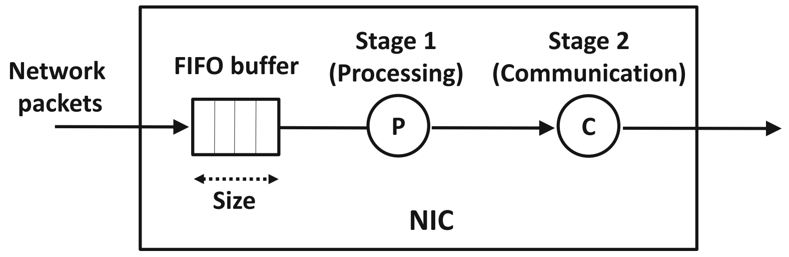

Fig. 5. Modeling NIC operation as a two-stage processing resource.

From the simulation viewpoint, it should be considered some general requirements such as the implementation of the FIFO buffer as a circular ring, the parallelism/ concurrency that typically exhibits hardware systems (Fig. 6).

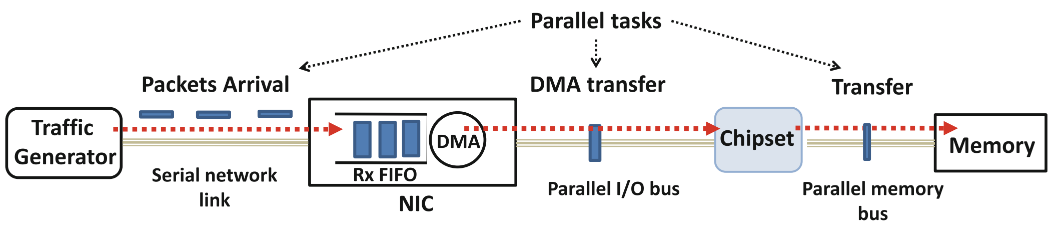

Fig. 6. Example of the existing hardware level parallelism in a compute node.

[1] It should be noted that in this work the terms "Ethernet frame", and "Ethernet packet", i.e., data units exchanged at the data-link level, are used interchangeably.

Regarding the next component in the network-to-memory data paths, i.e., NIC-System parallel interconnect, its access latency and operation will be considered. Here, bus contention should be taken into consideration (Fig. 7).

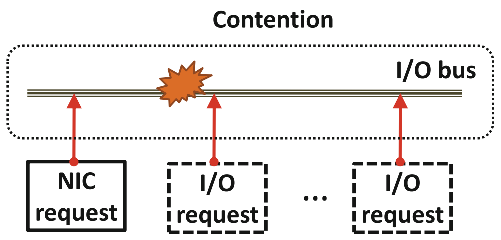

Fig. 7. I/O bus contention due to concurrent I/O devices access requests.

Finally, the last component in the data path is the memory subsystem. In our studied case, this component is modelled at very high level as we consider that the I/O-bus bandwidth in the SUT lags the memory subsystem bandwidth.

The broad or macro-level assumptions of the model decided at this step of the process are influenced equally by the objectives of the study and by the availability of resources (time, personnel, and funds) for the project. In some cases, lack of time may force the modeler to build a macro level model that satisfies only a subset of the original objectives of the study.

3.3 Model Justification

In this step, an important question may arise: a simulation model is actually needed; will an analytical method work?

The appropriateness of an analytical model for the study may not be easy to identify at the first phase of the study but may become evident as late as at the fifth phase of the study while the simulation model is being developed. In many cases, the availability of analytical models for the simplified version of the system can be useful in validating the simulation model later in the study.

Estimating how long a project will take and which resources will be used for the study is an important step. The detailed list of tasks to be performed in the study, the duration of each task, the resources to be used for each task and cost of each resource are needed in order to make a sound estimate of the resource requirements of the whole project. Level of detail and availability of data in the proper form are important factors in determining the time and type of resources required for the study. Availability of historical data from prior simulation projects can increase the confidence in the estimated resource levels, timing and cost of the project. A PERT analysis that gives the minimum, mode, and maximum duration for each task can be useful in estimating the total project time at different levels of confidence.

The cost-benefit analysis should be performed as a check-point in any study. A simple cost-benefit calculation for the study may also aid the modeler in determining the proper level of detail to include in the model.

It is common to observe a cost-benefit ratio of one to one hundred to one to one thousand from a typical simulation study when one looks at the total benefits gained throughout the life of the system.

Simulation projects can easily get out of hand, especially if the rate of change in the scope of the project exceeds the rate at which the results are available to the clients of the project. A Gantt chart showing the tasks with milestone points can help control the project.

The information gathered at the previous steps of this phase should be summarized in a formal proposal format to the client. If the proposal is for an outside client, it may also include sections on background of the simulation company, itemized cost and duration of the phases of the project, payment terms, and warranty.

4 Design

The simulation model can be used just to gather the necessary information to solve a problem, both as a training tool and for solving a problem, or as an on-going scheduling tool, etc. Each of these uses will affect how the model should be created.

The broad or macro level assumptions are finalized at this step. Which components of the real system will be excluded, included as a black box, included in moderate detail, or included in fine detail is decided at this step.

It is important to keep a formal list of broad assumptions throughout the study, starting at the design phase, since the scope of work and the micro level assumptions to be made in model building will depend on them. These assumptions should be kept in mind all through the process and included in the Final Report. A trap which many modelers may fall into is waiting until the end of the study to record their assumptions: by then, they may have forgotten many of them and wasted a lot of time in inadvertently changing the scope of work throughout the process.

In Ethernet, packets cannot be transmitted back-to-back. That is, there must be a gap between each packet. The IEEE specification calls this parameter the Inter Frame Gap. So, to accurately generate the worst-case network traffic, the start-to-start packet time should be computed. To this end, the packet time on the wire (PT) and Inter Frame Gap (IFG) should be considered (Table 1).

Table 1. Ethernet parameters (in bit times) for full-duplex trasnmission (worst case).

Parameter	Minimum-size packets (Ethernet)	Maximum-size packets (Ethernet)	Jumbo packets (Alteon)
Packet time	576	12208	72208
Inter Frame Gap	96	96	96
Packet start-to-start	672	12304	72304

Notice Table 1, the actual value of bit time (time per bit) depends of the Ethernet standard being modeled. Values of 0.1 µs, 0.01 µs, 1 ns, and 0.1 ns should be considered for modeling the standards, Ethernet, Fast Ethernet, Gigabit Ethernet, and 10 Gigabit Ethernet, respectively [15].

To implement the model of a generic network interface card, the module `NIC` is used. This module is the most important one and complex of NICSim-vhd. It interfaces with the network and the PCI-like parallel bus considered in this work. Structurally, the module, `NIC`, is composed of three submodules (`BuffMngr`, `NICCtrl` and `DMACtrl`).

The models in a study are developed in stages, each model representing an alternate solution for the problem. The first model developed is generally labeled as the Base Model, followed by Alternate Model 1, Alternate Model 2, etc. Each model may be tested under different variations of its parameters.

In some studies the number of models may be well-defined at the beginning of the study including the number of variations of their parameters to be tested while in others the alternate models are considered only if the previous models do perform unsatisfactorily. If the latter, the project team should assume a certain number of models for the project and proceed with the process.

Animation is an important tool in the simulation process and is utilized during model verification and validation, in selling of the simulation results to management, in generating new ideas for the design and operation of a system, and for training. Different detail levels of animation may be required for different purposes. 3-D detailed animation may be required in a detailed engineering verification simulation study. On the other hand, a simple (or no) animation may suffice in a computer architecture simulation studies.

A key issue in the analytical model validation and calibration is to monitor the FIFO-buffer fill-level, and provide parameters required to construct RTC descriptions of network workload and hardware components (e.g., NIC, I/O bus, memory subsystem) involved in the data path via arrival curves and service curves.

Because of none of the reviewed alternative ways fits well our requirements, implementation details of our VHDL-based simulation model (called NICSim-vhd) are also briefly described. In NICSim-vhd, we simulate both the scenarios under which a FIFO overflow occurs and the scenarios under which the buffer is bounded. To this end, various configuration parameters such as transmission line speed, DMA burst size, NIC latency, I/O-bus frequency and width are provided.

Selecting the best simulation tool for the study depends on many factors including the life-cycle of the project, tools the client currently owns, tools the simulation group has experience with, the animation requirements of the study, the detail level of the study, and the system considered in the study. Sometimes trying to force a simulation tool to perform beyond its designed intentions may add more time and cost to the process and slow down the execution of the model considerably. The project team should weigh all these factors before selecting the simulation tool for the study.

The main software tools used in the simulation model development process are GHDL and GTKWave. Specifically, GHDL [16] is an open-source command-line VHDL simulator that it is available for GNU/Linux systems and for Windows. This simulator implements VHDL standard as defined by IEEE 1076 and supports most of the 1987 standard and most features added by the 1993 standard. In addition, GHDL simulator can produce Value Change Dump (VCD) files which can be viewed with a wave viewer such as GTKWave [17].

Data collection activities may take a lot of time in a study and it is important that the modeler get involved early in setting up the data collection plans for the study. The modeler should examine all the data available and learn how and when it was collected. The modeler should check if macro or summary level data is available on all stochastic processes in the system. Before deciding whether to collect any more data, the modeler should assess how crucial the missing data may be for the study. Based on experience, if it is decided to collect data on some process, the actual data collection should be started only if there is no macro data available or experts disagree on estimates for the process. Even in such a case, the data collection should be at a macro level initially. Detailed data collection should be made only after the simulation model has been used to verify that the data is crucial for the study (Phase 6).

The human requirements in a simulation study are due to (1) interaction with people familiar in management of the system, (2) interaction with people familiar with the engineering details of the system, (3) modeler(s) familiar with the simulation tool to be used as well as experience with modeling similar type of systems, and (4) data collection required by the study. It is best to have an engineer and an engineering manager from the client's company be part of the simulation project team. In some studies, due to time limitations of the project, one may need to divide the modeling tasks among several modelers. It is important for a simulation manager to understand the limitations and strengths of the modelers in the project team. Data collection for a study may be done directly by the modeler(s), engineers familiar with the system, or by a third party. It is important that the modeler and data collection people meet and plan for the procedure to use for data collection. The modeler should visit the site of data collection and observe the data collection process if he or she is not involved in collecting the data directly.

It is very important to include the highest level of management possible from the client company in the modeling process. Many times each level of management has a different agenda and will require different types of information from the model. A good method is to informally record the information needed by each level of management from the client's organization and address this somewhere in the final report. If the study addresses as many issues as possible for these managers even though these managers may not be the direct customers of the project, the credibility and success rate of the project will increase.

Deliverables of a simulation study may include deliverables in report and file form as well as deliverables such as model-specific or generic simulation training, and customized user interfaces for model input and output.

5 Conceptual Model

Step 1. Modeling approach: continuous, discrete, or combined. Some of the elements (parts, people) in a system are dynamic in nature, in the sense that they move through the system causing other entities to react to them in response to some signal. Other elements are static in nature in a sense that they wait for the dynamic entities or resources to act upon them and pull or push them through the system. The modeler, considering the complexity, size, and detail level of the model, decides which elements should drive the system. One may classify models as part(entity)-driven or resource (machine)-driven models. In part-driven models, the parts are dynamic and they move from one resource to another as resources become available. On the other hand, in resource-driven models, resources pick the parts that they want to serve and send them to their next resource after completion. It is generally easier to build part-driven models. Resource-driven models are generally recommended if there are too many parts in the system or the resource allocation logic is very complex. It is possible to have models with both part and resource-driven characteristics.

Step 2. Determine the elements that drive the system. Some of the elements (parts, people) in a system are dynamic in nature, in the sense that they move through the system causing other entities to react to them in response to some signal. Other elements are static in nature in a sense that they wait for the dynamic entities or resources to act upon them and pull or push them through the system. The modeler, considering the complexity, size, and detail level of the model, decides which elements should drive the system. One may classify models as part/entity-driven or resource(machine)-driven models. In part-driven models, the parts are dynamic and they move from one resource to another as resources become available. On the other hand, in resource-driven models, resources pick the parts that they want to serve and send them to their next resource after completion. It is generally easier to build part-driven models. Resource-driven models are generally recommended if there are too many parts in the system or the resource allocation logic is very complex. It is possible to have models with both part and resource-driven characteristics.

Step 3. Determine the entities that should represent the system elements. Each simulation tool makes its own finite types of entities available to the modeler to be used to represent the real system components.

Our simulation model consists of six VHDL modules (`Traffgen`, `Clkgen`, `NIC`, `IOsub`, `Memsub`, and `Statsgen`). According to the VHDL terminology, every module is a design entity, and the inputs and outputs are called ports (input and/or output). The ports of the instances are connected using signals.

Step 4. Determine the level of detail needed to describe the system components. The model detail is further discussed at this step by identifying the simulation tool constructs to be used to represent each real system component. The level of detail put into a model should depend mainly on the objectives of the study. It is not always easy to eliminate the unnecessary detail since the customer may question the validity of the model if the components are not modeled in detail. It is recommended that the detail should be added to a model in stages starting with a simple macro level model of the system. For each black box that will be used to represent a system component, the modeler should list the reasons why a black box is going to be used in the model. The reasons may include lack of time, perceived indifference of the output measures to the component, the objectives of the study, lack of data, and cost of modeling.

In the following, we describe our design entities.

- `Clkgen:` The clock generator (`Clkgen`) allows us to generate the different clock signals required in our model (Fig. 8a).
- `Traffgen:` The traffic generator (`Traffgen`) allows us to model the worst-case traffic for different Ethernet standards (e.g., 10, 100, 1000 Mbit/s, and 10 Gbit/s). In all cases, the arrival of fixed-size Ethernet packets (e.g., minimum, maximum, and jumbo) can be simulated with it (Fig. 8b).
- `NIC:` The module is the most important component in our simulation model. Internally, it is composed of three submodules (`BuffMngr`, `NICCtrl` and `DMACtrl`). Basically, the function of the buffer manager (`BuffMngr`) is to monitor the level of filling in the buffer. The DMA controller (`DMACtrl`) deals with DMA transfers. On the other hand, the NIC controller (`NICCtrl`) is responsible for overall packet transfer flow. Thus, it controls the sequence of actions required during packet and descriptor transfers. Note that the NIC-side processing is simulated through a configurable delay (Fig. 8c).

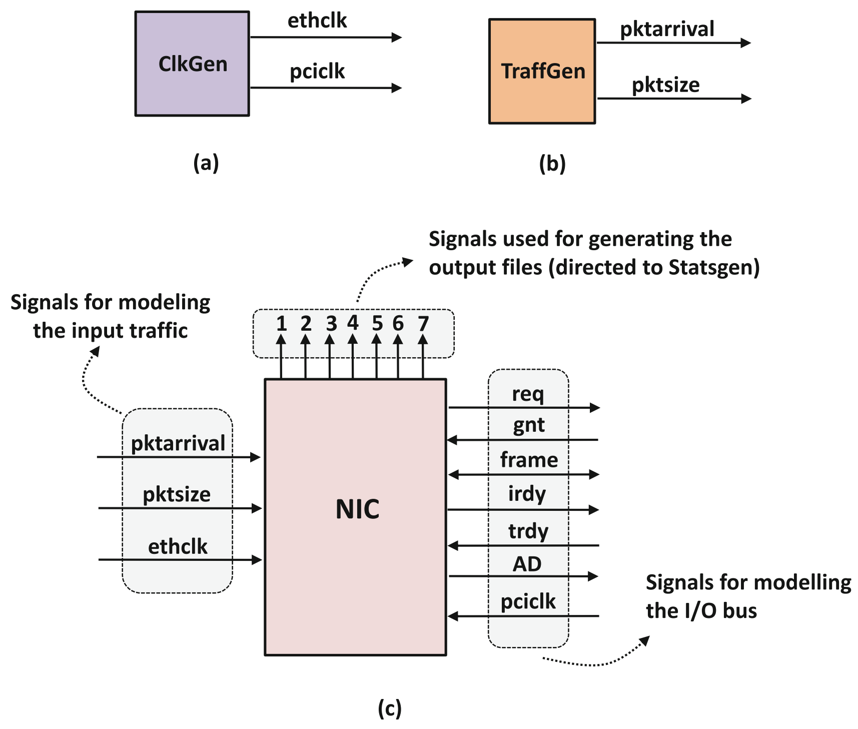
ClkGen
ethclk
pciclk
(a)
TraffGen
pktarrival
pktsize
(b)
Signals used for generating the output files (directed to Statsgen)
Signals for modeling the input traffic
1 2 3 4 5 6 7
pktarrival
pktsize
ethclk
NIC
req
gnt
frame
irdy
trdy
AD
pciclk
Signals for modelling the I/O bus
(c)

Fig. 8. Model entities: ClockGen and TraffGen and NIC entity.

- `IOsub:` In our case study, the I/O subsystem (`IOsub`) models a multi-drop parallel I/O-bus such as PCI/PCI-X. Structurally, the `IOSub` module consists of two sub-modules: `Arbiter` and `Othermaster`. The `Arbiter` deals with the arbitration process and `Othermaster` simulates the workload generated by other I/O units connected to the bus (Fig. 9).
- `Memsub:` The memory-subsystem module (`MemSub`) is the target of transactions across the I/O bus. It just simulates the initial target latency (Fig. 10).

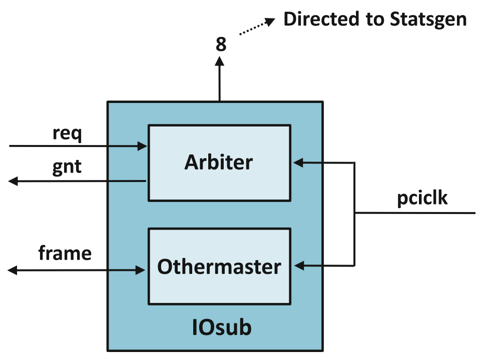

Fig. 9. IOSub entity.

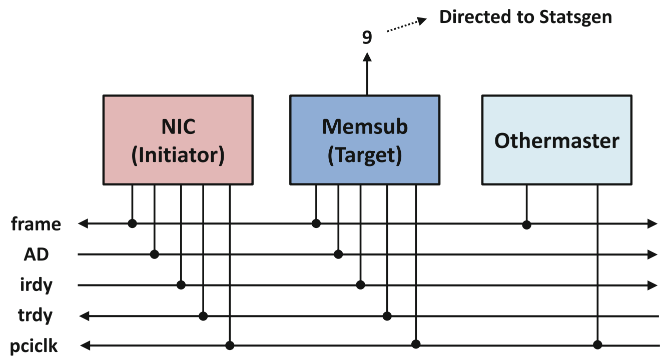

Fig. 10. PCI-bus control signals considered in this paper.

It should be noted that in a parallel bus such as PCI/PCI-X [1], when a bus master (e.g., the NIC) asserts `REQ`, a finite amount of time expires until the first data element is actually transferred. This is referred to as "bus access latency" and consists of three components, namely, arbitration, acquisition, and initial target latencies. For modeling bus access latency, the control signals of the PCI interface used in the simulation model developed in this paper are: `FRAME`, `IRDY`, and `TRDY`. Please, note that `FRAME` is like a "bus busy" signal. Thus, when a bus-master I/O device (such as the NIC) owns the bus because of it was acquired by the device, it should drive the `FRAME` line to '`0`'. On the other hand, when a master does not own the bus, it should have the line released (driving a '`Z`'). The module `Othermaster` drives the `FRAME` line to low during a random number of cycles. In this way, it is simulated the fact that the bus is busy due to a transaction in progress performed by any other I/O device connected to it (Fig. 10). In our model, `AD` line is used for transaction target identification. Because of both the `NIC` and `Othermaster` can drive `FRAME` line to low, 1bit `AD` line is used in order to the `Memsub` (i.e., the target of the NIC-to-memory transfers) can determine if it is the target of the current transaction (when `FRAME` = '`0`' and `AD` = '`0`'), otherwise, the target of the current simulated transaction is any other device connected to the bus (when `FRAME` = '`0`' and `AD` = '`1`').

- `Statsgen:` The main functions of the statistics generator (`Statsgen`) are: (i) keeping track of buffer fill-level and (ii) providing the required parameters values to analytical model calibration. In addition, this module provides other buffer statistics such as the count of dropped, received, and transferred packets. All this information can be obtained from various trace files.

Step 5. Determine the graphics requirements of the model. The animation requirements of the model are discussed in detail at this step by considering factors such as animation expectations of the client, execution speed of the animated simulation model, use of animation as a verification tool, use of animation as a selling tool, ease of building the static and dynamic animation components, and the availability of built-in icons in the animation library.

Step 6. Identify the areas that utilize special control logic. Systems with multiple machines or servers may require complex decision-making and control logic. During this step, the modeler discusses the best way of modeling the complex decision-making processes in the system.

Step 7. Determine how to collect statistics in the model and communicate results to the Customer. All the simulation tools produce standard output reports that may or may not readily translate into useful information for the customer. The modeler may need to define variables, counters, histograms, time series, pie charts, etc. to collect the appropriate data in a form that communicates the results effectively to the client. In many cases, the modeler may need to write user interfaces in a spreadsheet program to summarize the output data.

6 Inputs, Assumptions, and Process Definition

Operating philosophy describes the way that the management runs or intends to run the system. The issues considered include the number of shifts per day, the length of each shift, shift-end policies, scheduled breaks for workers and machines, strip logic for some machines, setup time, tool change, repair policies, off-line storage policies, production batch sizes and sequence, etc.

Physical constraints refer to those studies where layout options exist for the placement of material handling equipment, machines, and storages. In “green field” systems, one may have more options in number, type and placement of equipment, machines and storages. For existing systems, options may be limited on the placement of the new equipment in the system.

Once the dynamic elements of the model have been identified in the previous phase of the process, in this phase, the detailed logic of entry and exit of the dynamic elements is considered. Issues to be finalized include existence of infinite buffers before the first and after the last processors in the system and the logic to be used in arrival of entities to the system. For multiple dynamic elements case, arrival rates of each entity type, batch size, changeover time on processors and scrap and repair rates of each entity type have to be specified.

The process flow and the logic describing that flow as entities move through the system have to be specified in detail. The priority assignments for the entities as they compete for the resources and vice versa have to be described. Repair entity logic and priorities should be identified.

Operation specifications include data for each operation in the model including processing time, downtime distribution, percentage down, scrap and reject rate, setup time, capacity, etc. “Operation” in this case may refer to machining or service operations but excludes the material handling and storage activities in the system.

Material handling specifications include data for each type of material handling equipment in the model including transfer times in terms of acceleration, deceleration, minimum and maximum speeds, downtime distribution, percentage down, capacity, pickup and drop times, etc.

All the macro and micro level assumptions of the model are summarized at this step. This includes assumptions regarding the behavior of model components, input data, model detail level, startup conditions of the model, etc.

Input data should be analyzed and tested for reasonableness by the project team and line engineers and operators. Patterns in data should be identified, if any, and incorporated as part of input data generation. Theoretical distributions should be fitted to actual data and used in the model whenever possible (Law).

Runtime parameters are the variables whose values are to be changed from one simulation run to another in the study.

The detailed information gathered at the previous and current phases of the project should be used to update the Project Functional Specifications. This information should further be incorporated into the Maintenance Manual, model code, and the Final Project Report. By default, the information should be in the Project Book too. The detailed version of the Project Functional Specifications should be read carefully by the client engineers familiar with the system and corrected, if necessary.

7 Validation of the Conceptual Model

In this section, we analyze the network-to-memory data path from the perspective of RTC in order to show the requirements for the analytical model calibration and validation. According to our needs, various alternative ways that can be employed to validate the RTC model are reviewed and compared.

Acceptance of the model assumptions, operating philosophy, process flow, operation specifications, input data analysis, and runtime parameters by the client implies that the client validates the conceptual model of the modeler(s). A rigorous validation procedure for the conceptual model as discussed here is as important as the verification and ope-rational validation of the model because, being earlier than the others, it saves time and redirects the project team in the right direction before a lot of time is wasted in the study.

7.1 Analytical Model Calibration and Validation

In RTC, the basic model consists of a resource that receives incoming processing and communication requests and executes them using available processing and communication capacity. To this end, some non-decreasing functions such as the arrival and service functions as well as the arrival and services curves (upper and lower) are introduced. Similar to the arrival curves describing packet flows, the processing or communication capability of a resource is described by using service curves [18–20].

The task model represents the different packet processing and communication functions that occur within a network node, for example, NIC-side processing, bus access latency, and DMA transfer over the I/O bus to the chipset [11].

For model calibration and validation purposes (see [11] for details), a RTC-based data path model of the SUT is derived (Step 1 in Fig. 11). A simulator (or simulation model) of the SUT (Step 2) is required for calibrating the RTC model (Step 3). The accumulated buffer space in the RTC-based data path model, i.e., b1 + b2 + b3 + b4, allows us to derive the NIC buffer-size requirements by analytical methods. To compute such buffer space, the features provided by RTC are used. To this end, a resultant RTC component for the tandem of logical components, `NIC`, `ARB`, `ACQ`, and `MEMSUB`, is obtained. Then, by computing the backlog *b* of such component, the maximum buffer-size requirements are obtained (Step 4). To validate the analytically-obtained results, experimental studies using the VHDL model are performed. The buffer-size requirements obtained by analytical methods are compared versus maximum buffer requirements provided by the simulation model (Step 5).

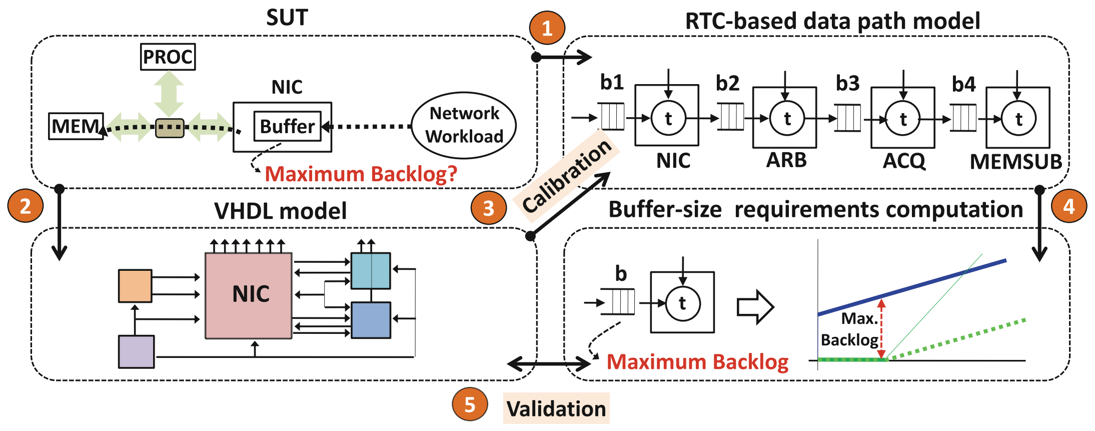

Fig. 11. Relationship between the conceptual validation, verification, and operational validation of the model.

Notice in Fig. 11 that the logical RTC components, `NIC`, `ARB`, `ACQ`, and `MEMSUB`, are connected by a bus, which is able to transfer one data block per I/O-bus cycle.

Since we are interested in modeling a PCI-like parallel bus, the objective of the components, `ARB` and `ACQ`, is to model the overhead due to arbitration and acquisition latencies. Finally, the function of the component `MEMSUB` is to model the overhead due to the initial target latency also known as target response. In addition to the overhead cycles, all these logical components also model the transmission cycles that are needed to transfer packet payloads and descriptors. Note that in our model no wait cycles are inserted during the data phase of the I/O transactions because we consider that the initiator of transactions, i.e., the NIC, is fast enough and is able to completely consume the bandwidth offered by the I/O bus; that is, the NIC is always ready to transfer a data block per bus cycle. Therefore, no additional buffer is required to model the overhead of wait cycles.

In our case, if α_f^u describe the arrival curves of the flow f in terms communication requests (e.g., data blocks) demanded from r, and, on the other hand, the bounding-curve β_r^l describes the communication capability of r in terms of the same units (i.e., "data blocks per I/O-bus cycles" that are transferred by r), then, the maximum backlog required by packets of flow f at the resource r can be given by the following inequality [18, 21].

$$backlog \leq sup_{t \geq 0}\left\{\alpha_f^u(t) - \beta_r^l\right\} \tag{1}$$

The backlog is defined as the amount of data units held inside the system (e.g., bits, bytes, packets, data blocks, etc.) (see Boudec and Thiran [22]). If the system has a single buffer, the backlog is related to the queue length. If the system is more complex, then, the backlog is the number of data units in transit, assuming that we can observe input and output simultaneously. Figure 2 shows the overall scheme for comparing the results obtained using the analytical method with those obtained from simulations used in our reference paper.

In [11], to calibrate the analytical model, the authors use a so-called "baseline experimental setup". For calibration purposes, the values of the parameters L_β and R_β (Step 3 in Fig. 11) provided by the VHDL-based simulation model are used to construct the service curves of logical communication components. After constructing the service curves, $\beta_{NIC}^l, \beta_{ARB}^l, \beta_{ACQ}^l$, and β_{MEMSUB}^l, by using the Eq. 2, an iterated convolution is used to compute the accumulated lower service curve β^l of the obtained resultant RTC component (Step 4):

$$\beta^l = \left(\left(\left(\beta_{NIC}^l \otimes \beta_{ARB}^l\right) \otimes \beta_{ACQ}^l\right) \otimes \beta_{MEMSUB}^l\right) \tag{2}$$

This can be done either by the RTC toolbox for Matlab [23], or by symbolic techniques [18, 21, 22].

Once, the bounding-curves β^l and α^u for the baseline configuration are constructed, the maximum backlog (i.e., the maximum NIC buffer-size requirements are computed by using the function `rtcplotv(f,g)` provided by the Matlab toolbox (Step 4 in Fig. 11).

To validate the RTC model, the simulator should fulfill two main goals. The first one concerns the monitoring of the FIFO fill-level. The second one relies on providing RTC descriptions of the logical communication component in which the network-to-memory data path has been decomposed. In the data-path model considered in this work, the basic data unit entering the system in order to be transfer consists of two components: packet and descriptor.

As illustrated in Fig. 11, during the trip of an incoming data unit from the network to the system memory, it should cross a tandem of logical communication components, whose operation is characterized by means of service curve. The procedure to compute the value of the parameters required for building the segments of the bounding curves of the network workload, and the operation of the logical communication resources has been shown in Fig. 12. Thus, in order to build the upper arrival curve, the parameters, PS (Packet Size), DS (Descriptor Size), and R_α, are required.

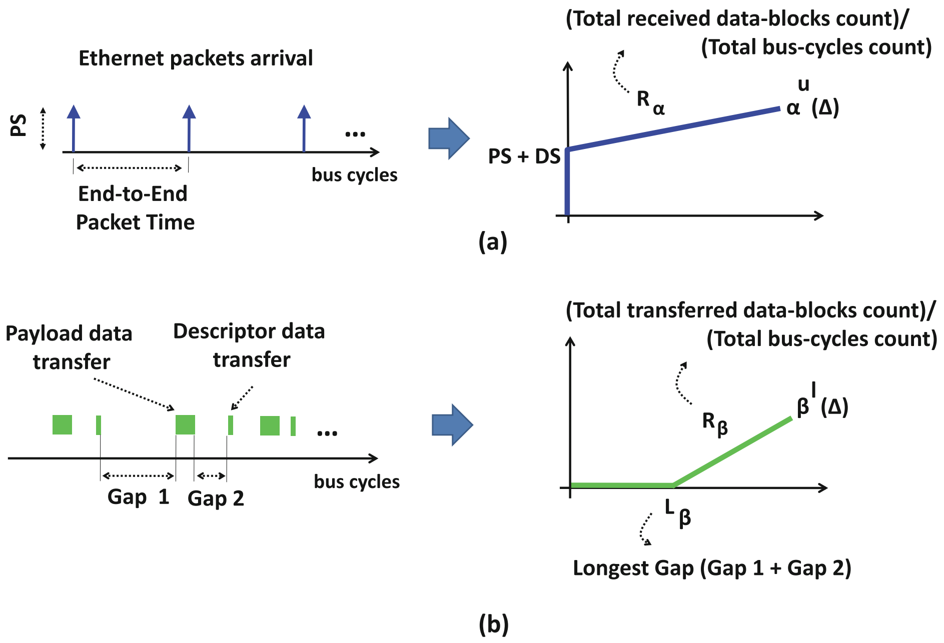

Fig. 12. (a) Building the arrival curve that characterizes the network workload from a trace that represents the instant of time when a whole packet is received by the NIC. (b) building the service curve that characterizes the communication capabilities of a resource with non-deterministic behavior from a trace in terms of both transmission and non-transmission cycles, as seen at the output of component.

On the other hand, to build the lower services curves, the parameters, L_β and R_β are required. Note that PS and DS are expressed in data blocks, and L_β are expressed in I/O-bus cycles. Finally, R_β are expressed in terms of received data blocks per bus cycles, and R_β in transferred data blocks per bus cycles (i.e., in the same unit). To clarify how bounding-curves are constructed, Fig. 12a shows the way of calculating the parameters required for building the upper arrival curve. Similarly, the way of calculating the parameters required for building the lower service curves is shown in Fig. 12b.

In the next section, the major simulation-based approaches known in literature for the performance evaluations of computer systems are classified. Moreover, a comparison of them according our needs is presented.

7.2 Alternative Ways to Validate the RTC Model

To a proper tool selection, a survey of nine alternatives ways that can be used to validate the RTC model was carried out. Survey results were compared and contrasted according a set of criteria [24].

Surveyed tools include hardware description languages (e.g., VHDL and Verilog), and system-level description languages (e.g., SystemC and SpeC). Also, different categories of simulators are considered; for example, component-oriented simulators (DRAMSim, ScriptSim, Moses, Spinach, NepSim), and full-system simulators (Simics, and M5).

In [24], the alternatives described above are compared. The capabilities of the simulation model to generate the parameters required to build the arrival and service curves as well as the general simulation-model requirements shown in Figs. 11 and 12, are taken into account.

We want to note that none of the reviewed alternative ways provides neither the parameters required for building the upper arrival curve (i.e., M and R_{WL}) nor the parameters required for building the lower service curves (i.e., L_{NIC} and R_{NIC}, L_{ARB} and R_{ARB}, L_{ACQ} and R_{ACQ}, L_{MEM} and R_{MEM}) used in our analytical model. Similarly, none of the alternative ways is intended to monitor the FIFO fill-level; consequently. Hence, we decide to implement a new VHDL-based simulation model to cover our needs; for a more in-depth look, see [24].

8 Description of the Simulation Model

To constitute the global program (executable model) that corresponds to the simulation model of data path, we assemble those modules (VHDL files) previously implemented. Thus, once composed the global program, all those modules cooperate to realize both the functions and the characteristics of the hardware components that are being modeled (Fig. 13).

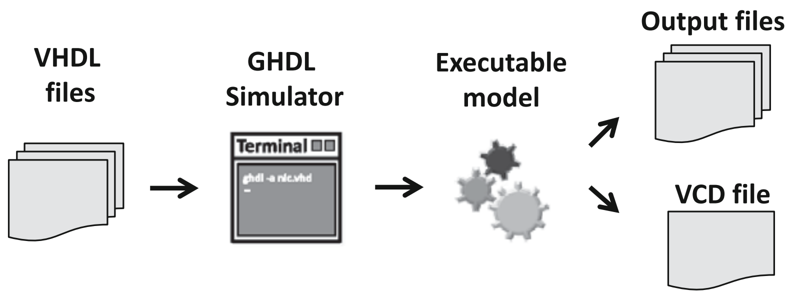

Fig. 13. Converting the VHDL model (NICSim-vhd) into an executable model and obtaining simulation outputs.

After running the executable model, six output files are created (input.out, nic.out, arb.out, acq.out, memsub.out, and buffer.out). All these files are generated by the `Statsgen` module (recall Figs. 8, 9 and 10).

More details of the structural and behavioral descriptions of all the modules/submodules of NICSim-vhd can be found at https://github.com/ggaray/nicsim-vhd.

9 Conclusions

To validate the RTC-based analytical model of the network-to-memory data path, we have proposed the main features of a VHDL-based simulation model. We show how it can analyze the performance of FIFO-buffers in network interface cards, and provide the required parameters for building RTC descriptions of the logical communication components of the data path. The main details of the HDL model have been presented in the paper. We note that although the VHDL language was chosen for modeling purposes due to software availability and standardization issues, the implementation of the simulation model in SystemC can be considered as another interesting approach.

Our simulation model can incorporate some improvements that we will develop in future work. For example, modeling variable input-packet sizes that follows a given probabilistic distribution (Poisson, etc.), or generates input packets from a real Ethernet trace. In addition, NIC latency will be modelled as a function of input-packets size due to per-byte data-touching operations within the NIC such as CRC and checksum calculation. Also, a PCI Express multi-drop parallel bus topology as well as more detailed model of the memory subsystem that considers the memory bus contention will be modelled. Finally, others performance metrics such as throughput will be provided by the individual model components (e.g., the NIC, I/O subsystems) and by the whole system.

This paper serves to increase the successful use of simulation techniques in the computer systems performance analysis by highlighting the importance of conceptual design and its validation.

Acknowledgments. This work is supported by the Ministry of Education and Science of Russian Federation under contract No02.G25.31.0061 12/02/2013 (Government Regulation No 218 from 09/04/2010).

References

1. Minkenberg, C., Denzel, W., Rodriguez, G., Birke, R.: End-to-end modeling and simulation of high- performance computing systems. In: Bangsow, S. (ed.) Use Cases of Discrete Event Simulation, pp. 201–240. Springer, Heidelberg (2012)
2. Liao, X.-K., Pang, Z.-B., Wang, K.-F., Lu, Y.-T., Xie, M., Xia, J., Dong, D.-Z., Suo, G.: High performance interconnect network for Tianhe system. J. Comput. Sci. Technol. **30**, 259–272 (2015)
3. Nüssle, M., Fröning, H., Kapferer, S., Brüning, U.: Accelerate communication, not computation! In: Vanderbauwhede, W., Benkrid, K. (eds.) High-Performance Computing Using FPGAs, pp. 507–542. Springer, New York (2013)

4. Rodriguez, G., Minkenberg, C., Luijten, R.P., Beivide, R., Geoffray, P., Labarta, J., Valero, M., Poole, S.: The network adapter: the missing link between MPI applications and network performance. In: 2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), pp. 1–8 (2012)
5. Tan, M., Rosenberg, P., Yeo, J.S., McLaren, M., Mathai, S., Morris, T., Kuo, H.P., Straznicky, J., Jouppi, N.P., Wang, S.-Y.: A high-speed optical multi-drop bus for computer interconnections. Appl. Phys. A **95**, 945–953 (2009)
6. Taylor, S.J.E., Khan, A., Morse, K.L., Tolk, A., Yilmaz, L., Zander, J., Mosterman, P.J.: Grand challenges for modeling and simulation: simulation everywhere—from cyberinfrastructure to clouds to citizens. SIMULATION, 0037549715590594 (2015)
7. Abdulhameed, A., Hammad, A., Mountassir, H., Tatibouet, B.: An approach based on SysML and SystemC to simulate complex systems. In: 2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pp. 555–560 (2014)
8. Bassi, L., Secchi, C., Bonfe, M., Fantuzzi, C.: A SysML-based methodology for manufacturing machinery modeling and design. IEEEASME Trans. Mechatron. **16**, 1049–1062 (2011)
9. Wang, Y., Yu, Y., Xie, C., Zhang, X., Jiang, W.: A proposed approach to mechatronics design education: Integrating design methodology, simulation with projects. Mechatronics **23**, 942–948 (2013)
10. Shafik, R.A., Al-Hashimi, B.M., Chakrabarty, K.: System-level design methodology. In: Mathew, J., Shafik, R.A., Pradhan, D.K. (eds.) Energy-Efficient Fault-Tolerant Systems, pp. 169–210. Springer, New York (2014)
11. Garay, G.R., Ortega, J., Díaz, A.F., Corrales, L., Alarcón-Aquino, V.: System performance evaluation by combining RTC and VHDL simulation: a case study on NICs. J. Syst. Archit. **59**, 1277–1298 (2013)
12. Ülgen, O.: Simulation Methodology: A Practitioner's Perspective. Dearborn MI University, Michigan (2006)
13. Alvarez, G.R.G.: A survey of analytical modeling of network interfaces in the era of the 10 Gigabit Ethernet. Presented at the 2009 6th International Conference on Electrical Engineering, Computing Science and Automatic Control, CCE, January 2009 (2009)
14. Kurose, J.F., Ross, K.W.: Computer Networking: A Top-Down Approach: International Edition. Pearson Higher Ed., New York (2013)
15. Karlin, S.C., Peterson, L.: Maximum packet rates for full-duplex ethernet. Department of Computer Science (2002)
16. Gingold, T.: Ghdl-where vhdl meets gcc. http://ghdl.free.fr
17. Bybell, T.: Gtkwave. http://gtkwave.sourceforge.net
18. Chakraborty, S., Künzli, S., Thiele, L., Herkersdorf, A., Sagmeister, P.: Performance evaluation of network processor architectures: combining simulation with analytical estimation. Comput. Netw. **41**, 641–665 (2003)
19. Chakraborty, S., Kunzli, S., Thiele, L.: A general framework for analysing system properties in platform-based embedded system designs. Presented at the Design, Automation and Test in Europe Conference and Exhibition (2003)
20. Garay, G.R., Ortega, J., Alarcon-Aquino, V.: Comparing Real-Time Calculus with the existing analytical approaches for the performance evaluation of network interfaces. In: 21st International Conference on Electrical Communications and Computers (CONIELECOMP), Los Alamitos, CA, USA, pp. 119– 124 (2011)
21. Thiele, L., Chakraborty, S., Gries, M., Kunzli, S.: A framework for evaluating design tradeoffs in packet processing architectures. Presented at the Proceedings of the 39th Design Automation Conference (2002)

22. Boudec, J.-Y.L., Thiran, P.: Network Calculus: A Theory of Deterministic Queuing Systems for the Internet. LNCS, vol. 2050. Springer, Heidelberg (2001)
23. Wandeler, E., Thiele, L.: Real-Time Calculus (RTC) Toolbox (2006)
24. Garay, G.R., León, M., Aguilar, R., Alarcon, V.: Comparing simulation alternatives for high-level abstraction modeling of NIC's buffer requirements in a network node. In: Electronics, Robotics and Automotive Mechanics Conference (CERMA), September 2010 (2010)