A Generic Multi-Phase On-Chip Traffic Generation Environment

Antoine Scherrer*
LIP - ENS Lyon
46, allée d'Italie
69364, Lyon cedex 7, France
antoine.scherrer@ens-lyon.fr

Antoine Fraboulet†
Tanguy Risset
CITI laboratory - INSA Lyon
20 avenue Albert Einstein,
69621, Villeurbanne, France
firstname.lastname@insa-lyon.fr

Abstract

We present hereafter a framework for on-chip traffic generation and networks-on-chip performance evaluation. This framework is based on a traffic generator that has three important characteristics: the splitting of traffic generation in multiple phases, the ability to replay a previously recorded trace in various interconnect systems, and the capacity to produce stochastic traffic with advanced statistical properties. We focus here on the second characteristics, by validating it in cycle-accurate SystemC simulations.

1. Introduction

In a near future, multi-processor SoC (MPSoC) will be commonly used in every day's life products. This will represent a major technological shift with the introduction of on-chip parallel architectures and networks-on-chip (NoC). During the design of a SoC, simulation is used extensively at various level of abstraction to validate the system. These levels, called *transaction level*, *bus accurate*, *synthesizable*, etc. have an increasing complexity. They are used to *refine* the specification down to the hardware and software components used in the SoC. Cycle-accurate simulations are particularly costly, and are often performed using extremely expensive hardware emulators. However, these precise simulations are necessary to prototype performances of highly non-predictable behaviors such as cache misses and network contention. This paper presents the principles of a simulation framework that we developed for NoC performances evaluation. We believe that the prototyping of on-chip network (or of any on-chip communication medium) will be extremely important in next generation SoC because *i*) an exact prediction of the network traffic is impossible and *ii*) over-dimensioning the on-chip network might be very costly in terms of chip area and power consumption. To perform NoC prototyping, designers commonly use *traffic generators* in simulation because cycle-accurate simulations integrating all SoC components are too slow. These traffic generators are supposed to emulate hardware components (or *intellectual properties*, IPs) and can be built after analyzing simulation traces of original components. This approach is efficient for dedicated IPs (such as hardware accelerators) but is much less precise for processors and caches which clearly exhibit non-predictable behaviors.

This paper explains how we have set up a traffic generation (MPTG) environment integrating advanced statistical analysis to model efficiently on-chip communication traffic. An important property of our environment is the fact that it can produce multi-phase traffic in order to switch model parameters during the simulation to match application computational phases. Another original feature, compared to existing similar environments, is the ability to create *long-range-dependence* (LRD) behavior for stochastic phases. This feature will not be presented here (the interested reader is referred to [11]). This paper focuses on practical experiments to demonstrate that *i*) the traffic generated by our environment is very close to the traffic it is supposed to emulate and *ii*) the multi-phase approach is manda-

*This work has been partly founded by CNRS and ST-Microelectronics

†This work has been done during a one year INRIA delegation in the COMPSYS group at ENS Lyon

tory to model traffic generated by embedded multimedia applications. These facts are demonstrated by cycle-accurate simulation of complete SoC platforms in SystemC using the SoCLib [1] simulation environment. Next section presents more in detail the work related to on-chip traffic modelling. In Section 3, we present our simulation framework and more details about the MPTG are given in Section 4. Experimental results are shown in Section 5.

2. On Chip Traffic Modelling

NoC performances are mainly evaluated with traffic generators (TG). A traffic generator generates a sequence of transactions (we call a transaction a request/response pair) with some characteristics which are as close as possible to the IP it emulates. The characteristics that are usually emulated are: target address, command (read or write), delay (number of cycles between two successive transactions), burst size (number of bus-words transmitted in a transaction). TG are often chosen because of the speedup factor they can achieve compared to the complete IP simulation but their most important advantage is that they can be parameterized, leading to a more flexible prototyping environment. Generating a TG should be done automatically. For instance, in [7] a trace compiler generates a program for a reduced instruction set processor. The latter will play the transactions in a cycle-accurate simulation without having to simulate the complete IP.

Traffic generators can be deterministic or stochastic. The deterministic approach is used to model IPs with a very regular behavior or when a stochastic framework is not available, this is the approach taken in [4, 7, 6]. In a stochastic TG the traffic is produced by parameterized non-deterministic process [14, 5, 9]. For example [5] uses stochastic models for generating transaction's sizes and times between two transactions using several statistical laws (Poisson, Exponential and Normal). Stochastic TG can be categorized according to the statistical properties they can emulate. The first property provided is the emulation of the probability distribution function (PDF) which is the first order statistics. The PDF shows the statistical distribution of the values of a process. A more accurate framework takes into account the correlations between values of a process (second order statistics), represented by the covariance function. In particular, long-range-dependence is detected in second order statistics. To our knowledge, only our MPTG and the simulation method used in [13] are able to model long-range-dependence processes in a SoC. However the work presented in [13] deals with macro-bloc size communications in MPEG2 applications while we provide cycle-accurate simulations. Long-range-dependent behavior is likely to have an important impact on network contention as is was demonstrated on Internet communications [8]. To our knowledge, none of the environments mentioned here permit to decompose the traffic in *phases* with different characteristics. We call a phase a splice of a trace whose statistical characteristic are stable in time, and this is likely to correspond to different parts of the program being executed. This part has often been omitted because emulating with precision a stationary stochastic process is itself a difficult problem. We think that multi-phase traffic generation is mandatory and is also an interesting research problem.

3. Simulation Framework

The results presented in this paper have been obtained in the SoCLib simulation environment [1]. SoCLib is a library of open-source cycle-accurate SystemC simulation models. Example of simulation models available in SoCLib are: a MIPS R3000 processor (with its associated data and instruction cache), standard on-chip memories, DMA controller and several kinds of networks-on-chip. Two NoC simulation models are available in SoCLib: Spin [2] and DSpin which is a distributed version of Spin with many improvements, both have been provided by the Lip6 laboratory. The network-on-chip we use is a set of 4 ports routers that can be interconnected in mesh topology in order to provide the desired packet switched network architecture.

The software running on the processors are compiled with the GNU `gcc` tool suite. We used a tiny multiprocessor operating system called `mutek` [10] to run the application. Interconnecting the different SystemC models to obtain a complete platform is not an easy task. A script (called `SocGen`) has been written to generate automatically the SystemC interconnection file from a very simple text file describing the components used in the platform and the architecture of the network. This script also generates software configuration files so that the simulation can immediately take place. We think that a platform generator tool like `SocGen` is an essential component of any SoC simulation environment.

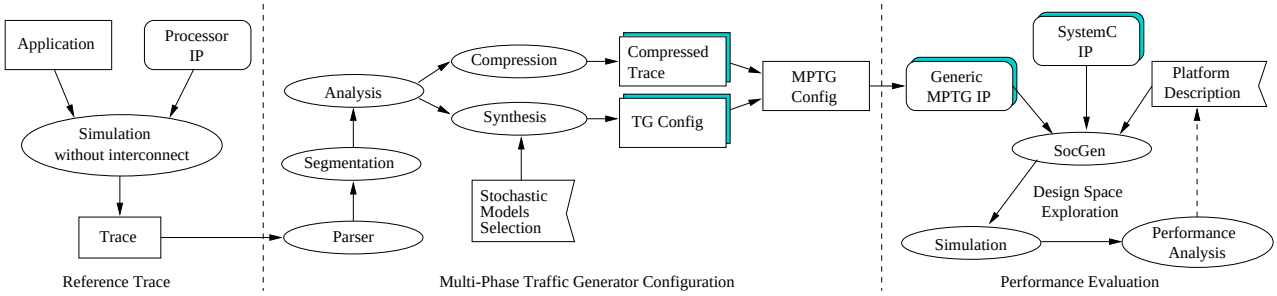

Figure 1. Complete simulation flow with MPTG framework

4. Multi-Phase Traffic Generator

In our methodology, a MPTG is generated by a software framework that is described here. The whole MPTG generation process is illustrated on Fig. 1.

• *Trace generation:* first we generate a trace by simulating the IP to be emulated. This trace is obtained with an ideal network environment: no network contention. This point is important, it allows us to avoid a cycle-accurate simulation of the complete SoC for TG generation process, and to use again the TG in various interconnect architectures. In our experiments this IP consists in a MIPS processor and its associated instruction and data caches.

• *Trace analysis*: this trace is parsed, segmented and analyzed in order to extract the sequence of transactions. This sequence can then be either adjusted to some stochastic process (see [11] for details), or just recorded and compressed (using the BZ2 block-sorting algorithm) in order to be replayed later.

• *TG configuration*: a generic traffic generator IP has been written, once for all, in SystemC. This TG uses a configuration file which indicates what kind of process it should generate. A simple format has been defined for the configuration file. This file is generated by the `synthesis` program, hence the designer do not have to write it. For stochastic process generation, we used an open source C library called `newran` [3], that we have extended with functions to generate long-range-dependent processes.

• *Simulation with TG*: a complete SoC platform can now be simulated using our TG instead of original IPs. This platform can include the simulation model of a real network and can be run with different parameters (IPs positions, buffer sizes) so that NoC prototyping can be done.

Multi-phase traffic can have a very important impact on network's performance. That is to say at some point the traffic may be very low, whereas as some other point it can be very high. Each phase in our TG can be either randomly generated by means of stochastic processes, or can be constituted of a recorded piece of the original trace. During execution, the TG switches between phases. In this work the segmentation has been done manually. This was possible because of the regularity of the multimedia applications we used but still needs to be automated in a futur work.

5. Performances Results

In this section, after discussing the simulation time issue, we present experimental results showing the accuracy of the MPTG.

We performed experiments with three different applications: a MP3 decoder, a JPEG decoder and a MPEG decoder. In this paper, we present results with the MP3 application and the DSpin NoC, results concerning other application and other NoC are similar and are available in [12]. The SoC platform on which we made our experiments is based on a mesh architecture using several IPs and routing configurations. We also include background traffic generators to create random network contention during performance measurements.

5.1. Design time setup

We explain here the work needed to setup a NoC performance evaluation platform and give an estimation of the additional design effort needed to use MPTG in the design flow. The different steps of the methodology are the following: *i*) *generation of the initial platform.* It consists in writing the configuration file and running the `SocGen` which is immediate.

# processors	1	2	3	4
mesh size	0x0	2x2	3x3	4x4
MIPS Sim. Time	35.44	142.5	304.5	683.50
MPTG Sim. Time	15.05	102.9	190.9	406.3
Speedup	2.35	1.38	1.59	1.68

Table 1. Simulation time (s) of MIPS versus various generated TG (5.10^6 simulated cycles of MP3 application)

ii) *Initial simulation* of a MIPS platform. *iii*) *Trace analysis* and phase determination. Trace analysis takes approximately a minute for a 5 million cycles simulation. The MP3 application is decomposed in a repetitive sequence of two phases, this can be easily detected as seen on Fig 2. *vi*)TG *configuration* file generation is immediate once the designer has chosen the phases.

Except for the phase identification process, we can safely affirm that no additional time is spent by the designer when using MPTG, provided he is used to MPTG tools. Most of the time will be spent in initial and final simulation. An important feature of our MPTG framework is that when changing a parameter of the platform (e.g. the architecture or the router's buffer size) only the MPTG platform has to be simulated again.

5.2. Simulation Time

We compared the simulation time of the MP3 decoder application running on the original MIPS and on the corresponding TG generated from it. Simulation results are shown in table 1, the rows labelled MIPS *Sim. Time* corresponds to the simulation of a platform including MIPS processors (with data and instruction cache) and memories. The Row labelled MPTG *Sim. Time* corresponds to simulation in which the TGs replaced the MIPS and cache.

We insist on an important point here, the measures presented in this section do not exhibit a dramatic reduction in simulation time (between 1.5 and 2.5) because much of the simulation time is spent in the interconnection system. We argue that the real benefit of a traffic generation environment lies in the flexibility of the tool. For instance, the application probably have a particular phase during which most of the NoC problems occur. Simulating only this phase is very easy with MPTG just by keeping some

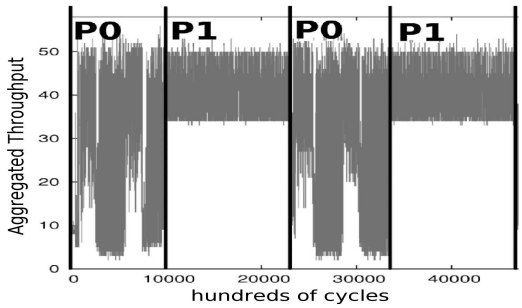

Figure 2. Segmentation of the MP3 aggregated throughput into four phases

phases in the TG configuration file, while it is often much more difficult to isolate subsets of program execution from the whole execution of the application. Another important point is that TGs can be used without having access to the source code of the original IPs, which is useful if network prototyping is done by a third party.

5.3. Validation of the traffic accuracy

We first executed the MP3 application on a MIPS (with its associated instruction and data cache) directly connected to a memory, and recorded the communication trace at the VCI (Virtual Component Interface, used in SoCLib interface of the cache. This platform is referred as *no NoC*. The trace was manually segmented into phases (see Fig. 2). We then followed the flow of Fig. 1 and generated three different traffic generators used on our mesh platform referred as *with NoC*. In the *replay* configuration the MIPS traces are replayed by the TGs. In the *simple* configuration the TGs generate a monophase stochastic traffic with throughput mean and variance fitted on the reference trace. In the MPTG configuration, the TGs take the two phases into account and the probability distribution function is taken into account per phase. These TGs can replace the MIPS and emulate the MP3 application traffic.

The results are presented in table 2. We show the relative cycle count error on the mean throughput for each identified phases with respect to the original MIPS simulation. As expected, on the platform without NoC, *replay* and MPTG exhibit very good results. The *simple* TG however is inaccurate because statistics were adjusted on the complete trace, so even if the global error remains low, the error on

Platform	Config.	#cycle error	Throughput av. error		
			global	P0	P1
no NoC	replay	0 %	0 %	0 %	0%
no NoC	simple	5 %	-1.8 %	22.3 %	13.6 %
no NoC	MPTG	0 %	0 %	0.1 %	-0.2 %
with NoC	replay	0.08 %	0 %	0 %	0.1 %
with NoC	simple	0.65 %	-6.6 %	15.4 %	5.8 %
with NoC	MPTG	-1.7 %	1.4 %	1.2 %	1.8 %

Table 2. Relative mean error of cycle count and throughput (per phase) with respect to the reference MIPS simulation

each phases is very high (the error is then relative to the mean of each phase which are different), leading to a traffic that cannot be acceptable to model the MP3 application running on the MIPS. This shows that multi-phase traffic generation is necessary for a TG to produce accurate traffic. A very important point is that these results hold when *the same TGs* are used without NoC and with a real NoC. The relative error of MPTG remains lower than 2%, whereas for the *simple* TG is goes up to 15%. This is achieved because our traffic generator self adapts to the network, sending the data only if possible, therefore producing an realistic traffic whatever the interconnection system might be.

6. Conclusion

In this paper we presented our methodology for networks-on-chip performance evaluation. We explained how the MPTG was built and we showed that its ability to replay a trace obtained from a fast simulation can be used by a designer who wishes to prototype different network architectures. The contribution of this paper is to show that MPTG is a very good candidate for NoC traffic prototyping. We pointed out that multiple phases are necessary to emulate correctly a processor-like IP. We also show experimental results that validate the accuracy of the traffic generated by MPTG. This should convince the reader that multi-phase traffic generation is needed and that our environment provides good accuracy in traffic modelling. Our current work is now to provide further experiments in which more network characteristics are analyzed with our MPTG. We are also actively investigating the automation of the trace segmentation.

References

[1] Soclib simulation environment. On-line, available at `http://soclib.lip6.fr/`, 2005.

[2] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino. Spin: A scalable, packet switched, on-chip micro-network. In *DATE 03 Embedded Software Forum*, pages 70–73, 2003.

[3] R. Davies. Newran c++ random number library. On-line, available at `http://www.robertnz.net/`, Nov. 2005.

[4] N. Genko, D. Atienza, G. D. Micheli, J. M. Mendias, R. Hermida, and F. Catthoor. A Complete Network-On-Chip Emulation Framework. In *DATE 05*, pages 246–251, 2005.

[5] K. Lahiri, S. Dey, and A. Raghunathan. Evaluation of the traffic-performance characteristics of system-on-chip communication architectures. In *VLSID '01*, 2001.

[6] M. Loghi, F. Angiolini, D. Bertozzi, L. Benini, and R. Zafalon. Analyzing On-Chip Communication in a MPSoC Environment. In *DATE 04*, pages 752–757, 2004.

[7] S. Mahadevan, F. Angiolini, M. Storgaard, R. G. Olsen, J. Sparsø, and J. Madsen. A network traffic generator model for fast network-on-chip simulation. In *DATE 05*, pages 780–785, 2005.

[8] K. Park and W. Willinger. Self-similar network traffic: An overview. In K. Park and W. Willinger, editors, *Self-Similar Network Traffic and Performance Evaluation*, pages 1–38. 2000.

[9] S. G. Pestana, E. Rijpkema, A. Radulescu, K. Goossens, and O. P. Gangwal. Cost-Performance Trade-Offs in Networks on Chip: A Simulation-Based Approach. In *DATE 04*, pages 764–769, 2004.

[10] F. Pétrot and P. Gomez. Lightweight Implementation of the POSIX Threads API for an On-Chip MIPS Multiprocessor with VCI Interconnect. In *DATE 03 Embedded Software Forum*, pages 51–56, 2003.

[11] A. Scherrer, T. Risset, and A. Fraboulet. Analysis and synthesis of cycle-accurate on-chip traffic with long range dependence. Research report 2005-53, École Normale Supérieure de Lyon, Nov. 2005.

[12] A. Scherrer, T. Risset, and A. Fraboulet. Multi-phase on-chip traffic generation environment. Research report 2006-22, École Normale Supérieure de Lyon, June 2006.

[13] G. Varatkar and R. Marculescu. On-chip traffic modeling and synthesis for mpeg-2 video applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(1):108–119, 2004.

[14] D. Wiklund, S. Sathe, and D. Liu. Network on chip simulations for benchmarking. In *IWSOC*, pages 269–274, 2004.