

Fast Simulation Techniques for Design Space Exploration

Daniel Knorreck, Ludovic Apvrille, Renaud Pacalet

System-on-Chip Laboratory (LabSoC), Institut Telecom, Telecom ParisTech, LTCI CNRS
2229, Routes des Crêtes BP 193 F-06904 Sophia Antipolis, France

Email: daniel.knorreck@telecom-paristech.fr, ludovic.apvrille@telecom-paristech.fr, renaud.pacalet@telecom-paristech.fr

Abstract—In this paper, we present our current work on a UML based environment providing efficient means for system level design space exploration. First of all, a brief introduction to the concepts of DIPLODOCUS gives a general overview of our methodology. A cycle-based simulation strategy for the application modeled in terms of abstract tasks was introduced previously. This paper emphasizes a new approach which is currently subject of our research activity. A more coarse grained simulation of the modeled application can significantly reduce the simulation time. The basic idea is to merge several clock cycles and to process them as a whole whenever possible. Finally, this article gives an idea of possible enhancements of the simulation environment and longer term objectives.

I. DIPLODOCUS METHODOLOGY

System-level design space exploration in a System-on-Chip (SoC) design cycle is an issue of great concern in today's rapidly growing and heavily constrained design processes. The increasing complexity of SoC requires a complete re-examination of design and validation methods prior to final implementation. We believe that the solution to this problem lies in developing an abstract model of the system intended for design, on which fast simulations and static formal analysis could be performed in order to test the satisfiability of both functional and non functional requirements.

In this context, we have previously introduced a UML-based environment named *DIPLODOCUS*. The strength of our approach relies on formal verification capabilities and fast simulation techniques ([14] [4]). The DIPLODOCUS design approach is based on the following fundamental principles:

- Use of a high level language (UML)
- Clear separation between application and architectural matters
- Data abstraction.
- Use of fast simulation and static formal analysis techniques, both at application and mapping levels

Moreover, DIPLODOCUS includes the following 3-step methodology:

- 1) Applications are first modeled using tasks with communication capabilities.
- 2) Targeted hardware architectures are modeled in an orthogonal fashion. A set of usual hardware components has been defined (e.g. CPUs, buses, etc.).
- 3) A mapping process defines how applications are associated with a given architecture.

A task is described by means of usual operators (loops, tests, variable settings, etc.), of communication operators (reading/writing abstract data samples in channels, sending/receiving events and requests), and of computational cost operators (EXECx instructions). A mapping is described using a set of interconnected hardware nodes (CPUs, buses, hardware accelerators, etc.) on which tasks, channels, events and requests are mapped.

As mentioned initially, simulation is an integral part of the DIPLODOCUS methodology and enables the designer to get first estimates of the performance of the final system. In that context, an important issue to solve is the concurrency of operations performed on hardware nodes and more particularly on CPUs and buses. The first version of our simulation environment relied on the standard SystemC kernel and took advantage of the integrated discrete event simulator. Further information about this approach can be found in [8]. In that simulator, a SystemC process is assigned to each active hardware node, and a global SystemC clock is used as a means of synchronization. Abstract communication and computation transactions defined in the application model are broken down to the corresponding number of wait cycles. Thus, all active components are run in lockstep.

Unfortunately, the inherent latency of the SystemC scheduler due to frequent task switches made the simulation suffer from low performance. However, simulation speed is an issue of concern as it represents one argument to justify accuracy penalties as a result of abstractions applied to the model. In order to achieve a better performance, a new simulation strategy is presented in the scope of this paper. It leverages these abstractions by processing high-level instructions as a whole whenever possible. Furthermore, its simulation kernel incorporates a slim discrete event scheduler so that SystemC libraries are not necessary any more.

This paper provides a brief survey of the post-mapping simulation of tasks mapped onto a given hardware architecture. It is structured in three parts: Section I introduces our DIPLODOCUS modeling methodology, section II elaborates on the main concepts of an innovative simulation strategy and section III finally concludes the paper and draws perspectives.

because its execution could have an impact on other tasks (making them runnable for instance). This decision is in line with traditional mechanisms used for discrete event simulation. As we have to be sure that during the execution of the selected transaction no events will occur, special care has to be taken when considering communication-related commands (read to channel, write to channel, wait on event, notify event, ...). The length of these transactions are therefore calculated based on the number of samples to read, the number of samples to write, the content and the size of the channel as well as the size of atomic burst transfers on the bus.

After T_{12} has been scheduled, t_{SI} is set to the end time of T_{12} , referred to as t_{SInew} . If the completed transaction T_{12} causes a task to become runnable on CPU2, T_{22} is truncated at t_{SInew} and the remaining transaction is scheduled on CPU2. As a channel always links only two tasks, revealing dependencies becomes trivial. If a portion of T_{22} has been added to the schedule, t_{SII} is changed accordingly ($t_{SII} = t_{SInew}$). After this, all schedulers of CPUs which have executed a transaction are invoked. The algorithm has now reached its initial state where all schedulers have selected a potential next transaction. Again, the transaction which finishes first is scheduled...

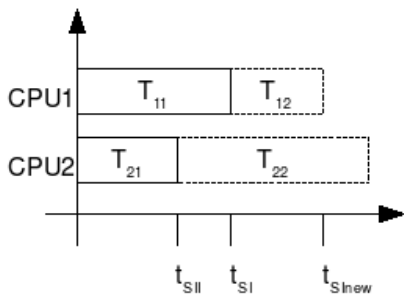


Fig. 2. Scheduling scenario

III. CONCLUSIONS AND FUTURE WORK

The main contribution introduced in the paper is the description of a lightweight discrete event simulation engine which is especially optimized for DIPLODOCUS high level application models. Simulation granularity is automatically adapted to the requirements of the application thanks to our transaction-based approach. When experimenting with models such the one of an MPEG2 decoder, we experienced performance gains in terms of execution time up to factor 30 as compared to a cycle-based SystemC simulator. It should be reemphasized that the gain depends on the application model, it could potentially be much higher.

Several improvements are currently under development.

At first, hardware component models could be enhanced. For example, the bus model shall reflect modern communication architectures comprising several possible communication paths (e.g. Network-on-Chip). The memory model is currently merely based on a fixed latency. Furthermore,

latency due to buffers in buses and bridges has been omitted. Thus, model realism versus simulation complexity is still under study for hardware components. Second, we expect to perform profiling of our simulation engine so as to identify time-consuming sections of code and recode them (e.g. in assembly language). Also, multiprocessor / core architectures could improve simulation performance significantly.

At last, more advanced simulation capabilities shall be introduced. Indeed, we expect the simulation engine to be able, at run-time, to check for functional requirements (e.g. if a client requests the bus, access is granted within 10ms). Also, it would also be very interesting to automatically explore several branches of control flow in order to enhance the coverage of simulations.

REFERENCES

- [1] Ttool, the turtle toolkit: <http://labsoc.comelec.enst.fr/turtle>.
- [2] L. Apvrille. TTool for DIPLODOCUS: An Environment for Design Space Exploration. In *Proceedings of the 8th Annual International Conference on New Technologies of Distributed Systems (NOTERE'2008)*, Lyon, France, June 2008.
- [3] L. Apvrille, P. de Saqui-Sannes, R. Pacalet, and A. Apvrille. Un environnement UML pour la conception de systèmes distribués. *Annales des Télécommunications*, 61:11/12:1347–1368, Novembre 2006.
- [4] L. Apvrille, W. Muhammad, R. Ameer-Boulifa, S. Coudert, and R. Pacalet. A uml-based environment for system design space exploration. *Electronics, Circuits and Systems, 2006. ICECS '06. 13th IEEE International Conference on*, pages 1272–1275, Dec. 2006.
- [5] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli. Metropolis: an integrated electronic system design environment. *Computer*, 36(4):45–52, April 2003.
- [6] Felice Balarin, Massimiliano Chiodo, Paolo Giusto, Harry Hsieh, Attila Jurecska, Luciano Lavagno, Claudio Passerone, Alberto Sangiovanni-Vincentelli, Ellen Sentovich, Kei Suzuki, and Bassam Tabbara. *Hardware-software co-design of embedded systems: the POLIS approach*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.
- [7] J. Eker, J.W. Janneck, E.A. Lee, Jie Liu, Xiaojun Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Yuhong Xiong. Taming heterogeneity - the ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, Jan 2003.
- [8] Ludovic Apvrille Muhammad Khurram Bhatti. Modeling and simulation of soc hardware architecture for design space exploration. October 2007.
- [9] A. D. Pimentel, S. Polstra, and F. Terpstra. Towards efficient design space exploration of heterogeneous embedded media systems. In *In Embedded Processor Design Challenges: Systems, Architectures, Modeling, and Simulation*, pages 57–73. Springer, LNCS, 2002.
- [10] A.D. Pimentel, C. Erbas, and S. Polstra. A systematic approach to exploring embedded system architectures at multiple abstraction levels. *Computers, IEEE Transactions on*, 55(2):99–112, Feb. 2006.
- [11] Bastian Ristau, Torsten Limberg, and Gerhard Fettweis. A mapping framework for guided design space exploration of heterogeneous mp-socs. *Design, Automation and Test in Europe, 2008. DATE '08*, pages 780–783, March 2008.
- [12] M. Silbermintz, A. Sahar, L. Peled, M. Anshel, E. Watralov, H. Miller, and E. Weisberger. Soc modeling methodology for architectural exploration and software development. *Electronics, Circuits and Systems, 2004. ICECS 2004. Proceedings of the 2004 11th IEEE International Conference on*, pages 383–386, Dec. 2004.
- [13] E. Viaud, F. Pecheux, and A. Greiner. An efficient tlm/t modeling and simulation environment based on conservative parallel discrete event principles. *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, 1:1–6, March 2006.
- [14] M. Waseem, L. Apvrille, R. Ameer-Boulifa, S. Coudert, and R. Pacalet. Abstract application modeling for system design space exploration. *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on*, pages 331–337, 0-0 2006.