Performance Exploration with MLDesigner using Standardized Communication Interfaces

Tommy Baumann, Alexander Pacholik, Horst Salzwedel

{Tommy.Baumann|Alexander.Pacholik|Horst.Salzwedel}@tu-ilmenau.de

TU Ilmenau, Faculty for Computer Science and Automation,

P.O.Box 100565, 98694 Ilmenau, Germany

http:/tu-ilmenau.de/sst

Abstract

We present a method to apply performance exploration by mapping functional models on different architectural variants. Validation of functionality and performance is applied by simulation.

1. Introduction

The aim of performance exploration on electronic system level is evaluating different realization variants, involving computation and communication resources. Both, communication and computation is normally strongly connected with the functional model to which the performance analysis applies. This is a problem, because for different platforms different models are necessary, while the functionality remains stable.
There are different approaches regarding performance exploration on electronic system level. Statical formal methods can be applied, if a pure mathematical model is available. However, these models often lacks of functional aspects. For this reason, dynamical aspects are disregard, pessimistic approximated data must be used. Another problem to validate the models, which is only barely possible for complex mathematical models. It can be summarized, mathematical models are well suited for worst case analysis, which can be applied on implementation level, but not in a design exploration stage.
Using simulation based methods, functional validation and performance exploration is possible [1,2]. The goal is not to find the absolute worst cases, but estimations for resource related key data. There already exists some approaches, like metropolis[3] and PeaCE [4]. In metropolis the user needs to create a functional model, which includes special channel objects. This is a bis cumbersome, because every information flow between functional parts implies the usage of communication resources. PeaCE incorporates the use of synthesizeable models. For complex systems the availability of such models can only assured in an advanced design stage.
Our performance exploration approach, does not only focus on Chip or Board level, but also incorporates distributed systems and complexer networks. To apply performance analysis for complex systems in early design stages, we remain on a modelling layer.
This paper deals with the definition of a framework for performance exploration. In the second chapter we introduce the used and adapted system design tool MLDesigner. In the third and forth chapter we introduce a Framework for Architecture Exploration and describe the concept of Standardized Communication Interfaces in detail. Chapter 5 an event based assertion monitoring environment is introduced

2. MLDesigner

The tool MLDesigner[5] is related to the earlier project Ptolemy and allows the creation and simulation of models containing different models of computation. MLDesigner provides many extensions on the simulation environment, especially for discrete event models. The discrete event domain is well suited for performance and resource exploration on system level under functional side conditions.
In the discrete event modelling paradigm events are used for communication. This includes data transfer ans signalling. Every arriving event potentially triggers the atomic execution of a model element. Emitted events contain a future time stamp, triggering future actions. This enables a more efficient simulation of models with different time Scales compared with cycle accurate models. To be noted, discrete event does not imply discrete time[6].

3. Performance Exploration Framework

Performance estimation for Functional Models and Architectural Models on System Level can only be achieved by timed simulation (discrete event). To meet holistic predications about both together, they have to be associated into the common executable Behavioural Model. On the other side the separation is necessary to enable iteration over different architecture options. This implies a flexible mapping between the models. The Mission Level System Design Flow implements these requirements.

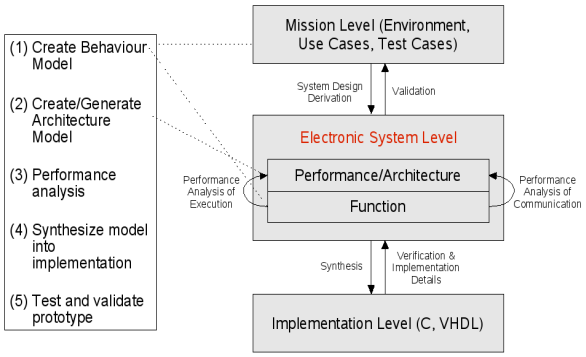

Figure 1: Mission Level Design Flow

Figure 1 shows in overview the proposed design flow from mission level with informal descriptions of the systems tasks, use- and test-cases and its environment to implementation in hard- and software. The link between mission level and implementation is the electronic system level (ESL). Function and architecture is designed and validated at that level. The left side of Figure 1 shows necessary steps for a systems design and

their connection to the particular abstraction levels.
The execution of the Behavioural Model delivers performance values e.g. for channel throughput and memory usage rates. The output is dynamic summarized by a GUI element (Performance Monitor). To enable a flexible exchange of transfer protocols, channels and execution components (partitions), standardized interfaces are necessary. They are defined in a common MLDesigner library, the Network Block Set. Furthermore a special MLDesigner domain target, the ESLTarget, was developed. It ensures the valid usage of Network Block Set elements and outputs measured performance values.

4. Standardized Communication Interfaces

The modeling of different exchangeable protocols (based on OSI reference model) and the communication between desires standardized interfaces and data structures. Each interface belongs to a partition and can be designed on a user specific abstraction level. At this the most important is located between physical and link layer because there is the connection point of partition and channel. Channels represent the physical medium and connect two or more partitions, depending the medium characteristics and the network topology. All model elements are part of the Network Block Set library, allowing a fast model driven exploration of different architecture options.
Data transfer is accomplished by different protocol depending data structures. Each protocol packs or unpacks the received data, whereby the size increases or decreases and partition ressources are occupied (e.g. memory). The data is send thru channels and delayed considering the channel resources (e.g. bus frequency). The granularity of data packets depends on the used protocols and abstraction level. Each selected channel, partition and interface reports it performance values dynamicly to the common ESLTarget. This allows a detailled analysis of the current Behavioural Model. The selection ist also used by the following annotation based synthesis.

5. Event Based Assertions

Assertions are formal temporal properties, annotated to the design. These properties are used to detect bugs very early, by specifying the intended functional behaviour of the design artefacts.Using assertions is common for RTL designs. Different approached have been applied to integrate assertions in abstracter design stages, such as SystemC. MLDesigner faces even higher abstraction levels, thus we propose to use communication assertions related to events occuring in the system. To check communication assertions in discrete event models, we need to define proper semantics.
Events within a discrete event models contain a distinct ordering, even if they own the same time stamp. However, in the later implementation we are not able to observe a relative ordering between events taking place in the same time. The ordering between events is only partly defined by the model. The total ordering is determined randomly by simulation. Thus it is more common to define temporal properties related to the timing of events, than involving properties resulting from the partial ordering of events.
Temporal properties are defined in Linear Time Temporal Logic (LTL), Computation Tree Logic (CTL) and generalized in CTL* [7]. To describe properties, observable during simulation, we can restrict to LTL.[8] In real time systems most properties must hold in finite time, therefore we can use interval restricted temporal operators, similar to FLTL[8].

Let I the set of event identifiers then $E_n = I \times B_n$ is a state set, containing the boolean evaluation, if an event occurs at sequence time n . Given an event e , then $E_n(e) = e(n)$ results true, if event e is active at time n , and false otherwise. A sequence trace is modelled as a function $W = N \rightarrow B$ mapping natural numbers to boolean values. An event sequence is a tuple of an event identifier and a value sequence $\pi = I \times W$.
Due to the observation of discrete events in continuous time, a state vector is valid for a distinct time interval. This results in alternating state vectors, valid for occurring events (time span with length zero) and state vectors valid for the time between occurring events (time span greater zero).
The mapping from value time to sequence numbers is defined by $S = R \rightarrow N$. An assertion defined in continuous time can now be evaluated in sequence time, by evaluating the sequence steps relating to the continuous timing.
We realized the temporal operators X , $F_{[m,n]}$, $G_{[m,n]}$ and $U_{[m,n]}$ with three valued logic according to [7] with some changes. The interval restrictions m and n are real valued times, intead of naturals (clock related). The next operator X always addresses the very next point in time. Assertions are integrated by using an event handler inserted between scheduler and model. This allows to integrate assertions by annotations on model level, rather than on code level, which is common for RTL designs.

6.Conclusions

We introduced a performance exploration framework, which allows an easy evalutation of different achitectures. The exchangeability of communication structures is ensured by using standardized communication interfaces, according to OSI protocol layers. We implemented this framework within the tool MLDesigner and validated it using different reference models.
To support functional validation of communication assertions, an event based assertion checker was implemented and validated.

7.References

[1] K.M. McNeir, M. Zens, H. Salzwedel. "System Level Partitioning Using Mission-Level Design Tool for Electronic Valve Application". SAE 2003World Congress, Detroit, Michigan.

[2] G. Schorcht, I. Troxel, et al: "System-level simulation modeling with MLDesigner". Modeling, Analysis and Simulation of Computer Telecommunications Systems 2003. MASCOTS 2003. 11th IEEE/ACM International Symposium.

[3] F.Balarin, Y. Watanabe, et al: “Metropolis: An Integrated Electronic System Design Environment”. In Computer, IEEE Press April 2003.

[4] S. Ha, C. Lee, et al: "Hardware/ Software Codesign of Multimedia Embedded Systems: the PeaCE Approach”. Technical whitepaper peace.snu.ac.kr/research/peace/data/whitepaper/PeaCE_whitepaper.pdf

[5] MLDesign Technologies Inc. Homepage. www.mldesigner.com

[6] E.A. Lee, A. Sangiovanni-Vincentelli “A Framework For Comparing Models Of Computation". In IEEE Transactions on CAD, Vol. 17, No. 12, December 1998

[7] T. Kropf “Introduction to Formal Hardware Verification”, Springer 1999, ISBN 3-540-65445-3

[8] J. Ruf, D.W. Hoffmann, et al: “Simulation Baased Validation of FLTL Formulas in Executable System Descriptions”, IEEE International High-Level Validation and Test Workshop HLDTV'00, pp.161f