Mapping of Electronical System Level (ESL) Models into Implementation

Tommy Baumann, Horst Salzwedel

{Tommy.Baumann|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

A methodology will be presented to design at Electronic System Level (ESL) and map the design into an implementation using a Mission Level Design tool.

1. Introduction

Electronic systems in aircraft, automobiles, home entertainment systems, mobile telephones, GPS/Galileo navigation systems, multi-functional systems such as Quint Network Technology (QNT) or Systems On Chip (SoC) are today networked embedded systems that exhibit complexities that can no longer be developed at reasonable cost and acceptable technical risk at implementation level or functional level. Integrated hierarchical model based design methodologies and tools have to be used to integrate the design flow from concept to implementation. Moving model based design methodologies to electronic system level (ESL) has permitted rapid development of verified executable specifications, model systems at system level [1], fast optimization of networked architectures at performance level [2] or size resources in complex reconfigurable electronics [3]. The move towards higher abstraction has made it possible to cope with the increased complexities. However, it has widened the gap between design and implementation.
In this paper a method is presented that overcomes the gap between abstract system models for design and the realization in hard and software at RTL level. An integrated design methodology and extensions for the tool MLDesigner [4] are presented that makes design decision on function, performance, and architecture at ESL and translates this design automatically to VHDL. Figure 1 shows in overview the proposed design flow from mission level with informal descriptions of the systems tasks, use- and testcases and its environment to implementation in hard- and software. The link between mission level and implementation is the electronic system level. Function and architecture is designed and validated at that level. The left side of Figure 1 shows neccessary steps for a systems design and their connection to the particular abstraction levels.

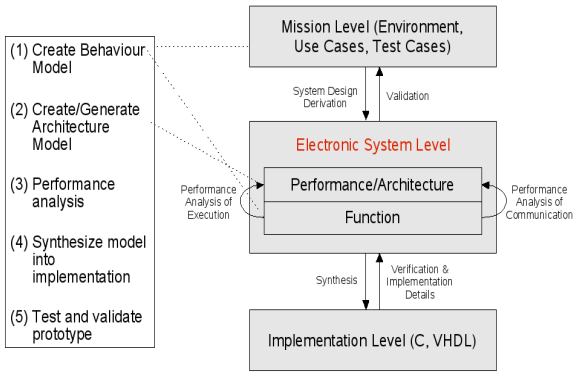

Figure 1: Mission Level Design Flow

2. MLDesigner

The tool MLDesigner is related to the earlier project Ptolemy and allows the creation and simulation of models containing different models of computation. MLDesigner models and simulations can be defined by a graphical model editor and are stored in XML, permitting standardized methods for transformations and translations of the models. Multiple execution domains permit to model functional behavior, the environment and the architecture in their respective native forms. The execution domains include discrete events (DE) [5], synchronous and dynamic data flow, extended finite state machines (FSM) and continuous and analog domains.

3. Chippo Project

The new design methodology is demonstrated for the design of a Cheap High Precision Positioning (Chippo) system. Figure 2 shows the functional characteristics of the Chippo system. The position measurements of a GPS receiver are enhanced by measurements of accelerations and angular rates of an inertial measurement system on a chip and integrated by a Kalman navigation filter. The accuracy of the GPS receiver is enhanced by differential GPS corrections received from RASANT. The interface to external systems is realized by USB. For the development of this system some of the hardware components are available off the shelf, others have to be developed and integrated into the overall positioning system.

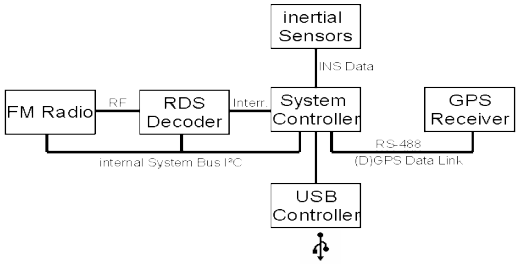

Figure 2: Concept of the GPS based Chippo system

The mathematical model of the sensor fusion system within Chippo is a discrete Kalman navigation filter [6]. This model was developed and validated with GNU Octave.
Characteristic trajectories for the missions of the system, use cases, are used for the validation. To permit a unified integrated design, the mathematical model was described by a hierarchical block diagram of the modeling tool MLDesigner.
Figure 3 shows the top-level hierarchy of the functional and environmental model. The states of the environmental model, e.g., the number of available radio stations or kinematic states are realized by memories. The characteristics of the environmental model can be changed by parameters. Additional elements for reception and decoding of differential GPS data are added. The functional model of the Chippo system therefore includes the validated functional model as well as the event based processes for operation of the system. The system is

validated by simulation against the mathematical model of the system.

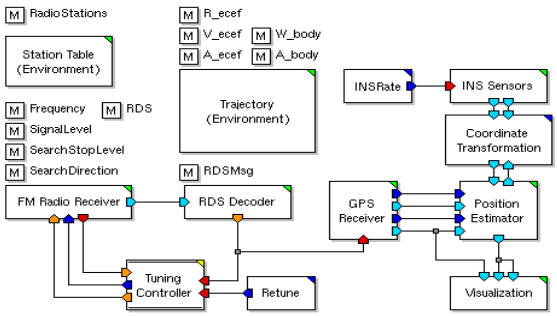

Figure 3: Top level structure of Chippo functional model

After system behavior is specified, accompanying execution- and communication structure must be pinpointed in the form of an architecure model (Figure 4). Executable components (partitions) are operating resources of their assigned functions of the functional model. Data transfer between partitions is modeled by communication components (channels). Communication structure arises from the partitions, because interacting functions within the functional model are mapped on channels within the architectural model. The architectural model permits performance analysis of execution and communication components by simulation on ESL level. Specific properties of partitions and channels are abstracted by parameterized resources. Utilization of resources is expressed in terms of quantities and queuing of events. Resources restrict the choice of assignable functions to partitions and throughput of channels. For example, quantity resources can model memory or CPU utilization of partitions and queue resources can control access of channels within the model. The architectural model is also the basis for the following synthesis.

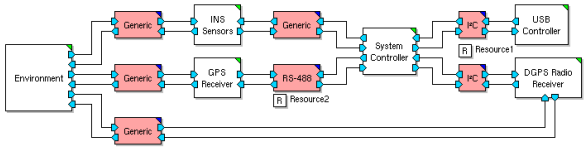

Figure 4: Chippo architectural model

The development of a working prototype at the implementation level requires transformation of the abstract models of function and architecture into hardware and software components. It is desirable that as little as possible has to be tuned by hand. The main task in translating the ESL functional/architectural model into a prototype is the translation of the algorithms into hard- and software languages (IP Cores) and mapping of transactions between architectural components into implementation specific communication structures. The architectural model, as we described it in the previous, permits to generate a message passing communication system, which can be used for hardware and software. Mapping of transactions into message passing communication models is controlled by modeled elements of the architectural model and annotations. The advantage of this method is its easy realization into languages like VHDL that do not explicitly support this construct.

The ESL functional/architectural model includes all information about communication system and distribution of functional and architectural components. Components of the architectural model become IP cores or independent SW processes of the implemented prototype. Transformation of models is carried out independently for each partition. Each partition is transformed by an XSLT transformation into implementable functional code plus interfaces (Figure 5). Modifications of the XSLT scripts permit to generate code for different description languages or intermediate descriptions. To perform this translation, XSLT transformations have been developed that translate annotated XML descriptions of the model into implementable code for FPGAs and ASICs. Dependent on the structure of the communication system and the architectural components, corresponding interfaces and architectural components have to be generated. There are a variety of standards that define communication between architectural components at board level, e.g., VME, PCI, as well as on ICs, e.g., Wishbone.

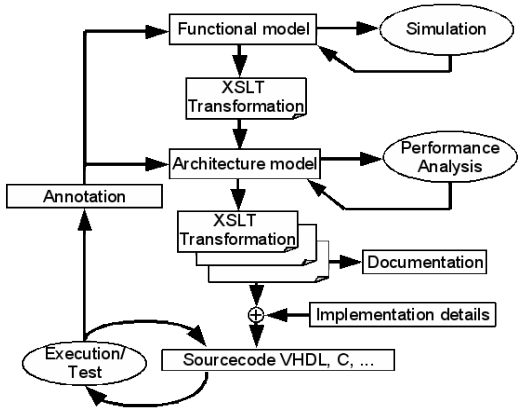

Figure 5: Iterative transformations for code synthesis

4. Conclusion

An integrated design method for overcoming the gap between design at ESL and implementation has been developed. The new design methodology was demonstrated for the design of a high performance GPS based positioning system with differential corrections and additional inertial measurements. During the functional level design step, a navigation filter was designed to meet the functional requirements of the system. In the next design step architectural elements were automatically added to this model by XSLT transformations from annotations. Sizing information for the architectural components were determined by performance simulation. In the third step the ESL model was translated by an XSLT script into HW/SW code of a prototype: Implementation details were added iteratively by updating annotations on the ESL model.

7.References

[1] Schorcht, Gunar; Troxel, Ian; Farhangian, Keyvan; Unger, Peter; Zinn, Daniel; Mick, Colin K.; George, Alan; Salzwedel, Horst. 2003. System-level simulation modeling with MLDesigner . Modeling, Analysis and Simulation of Computer Telecommunications Systems 2003. MASCOTS 2003. 11th IEEE/ACM International Symposium.

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

[3] D. Bueno; C. Conger; A. Leko; I. Troxel; A. George. 2005. RapidIO-based Space System Architectures for Synthetic Aperture Radar and Ground Moving Target Indicator . High-Performance Embedded Computing (HPEC) Workshop, MIT Lab, Lexington, MA.

[4] MLDesign Technologies Inc. 2006. MLDesigner Documentation . http://www.mldesigner.com.

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

[6] Qi, Honghui; Moore, John B. 2002. Direct Kalman filtering approach for GPS/INS integration . IEEE Transactions on Aerospace and Electronic Systems.