A Mapping Framework for Guided Design Space Exploration of Heterogeneous MP-SoCs

Bastian Ristau, Torsten Limberg, and Gerhard Fettweis
Vodafone Chair Mobile Communications Systems, TU Dresden, Germany
{ristau,limberg,fettweis}@ifn.et.tu-dresden.de

Abstract

When designing heterogeneous MP-SoCs designers have to take into account various objectives such as power, die size, flexibility, performance or programmability. But to be able to evaluate a given system according to these objectives, it is necessary to know how applications will behave on that system. Since time-to-market is one key factor in chip design, it is important to be able to evaluate these systems at a very early design stage. Today this is usually done with simulations in languages such as Simulink or SystemC. We will show how the behavior of such systems can be analyzed without the need for time-consuming implementations of simulation models. This enables fast evaluation and modification of a given system at a very early design stage allowing efficient pruning of the design space.

1. Introduction

Today heterogeneous multi-processor systems seem to be one answer to the continuously rising computing demands of modern applications. But the question is how the system inherent parallelism can be exploited. Thus, it is vitally important to know at a very early design stage, how applications will behave on the system with respect to runtime, power consumption, etc. Only with this information it is possible to evaluate a given system.

This challenge is usually tackled with simulation models written in languages such as Simulink or SystemC. But simulation models usually implicate a lot of implementation effort. Furthermore explicit spatial mapping of tasks to processors or at least mapping to a processor type is required. As a result only one of the available implementations is covered within a single simulation and each corner-case of the application has to be determined and simulated separately. Therefore identifying worst-case execution times is time-consuming and error prone.

In this paper we present a framework for automatic temporal and spatial mapping based on abstract models of algorithms and architecture. The mapping process is guided by programmer provided side information. This circumvents the downsides adherent to simulation models.

We will show that system refinement based on mapping results can lead to significant performance improvements.

2. Related Work

There are a lot of frameworks for modeling and simulating MP-SoCs like Simulink, SystemC, Sesame [9], Ptolemy [6], Casse [10], just to mention a few. These frameworks are well suited for their purpose of functional simulation and evaluation of systems, but they require the system to be described in on their own modeling language. In most cases also a manual mapping of applications to architecture is required.

In [4] a methodology using multi-objective optimization (MOO) is presented, that is producing an initial mapping for the Sesame framework, but the results have to be refined and checked with simulations. In [1] an automatic mapping approach for the Cell BE architecture is introduced requiring an a priori selection of processor types for each task. In [8] an approach is outlined that uses neural networks to determine a suitable processor type for a given part of the application targeted at instruction level.

Our framework in contrast is extensible for reading various modeling languages and provides the possibility to test different mapping strategies. It supports simultaneous temporal and spatial mapping of all tasks not only to processors, but also to processor types. The framework is not intended to be a replacement for the mentioned modeling languages. In fact it is intended to be used for finding a candidate system that can be simulated more detailed with existing modeling tools and languages.

3. The Mapping Framework

The mapping framework basically follows the Y-Chart approach [5], but is extended with an additional guidance

978-3-9810801-3-1/DATE08 © 2008 EDAA

step allowing the utilization of available expert knowledge (Fig. 1).

The mapping is performed on graph-based descriptions of architecture and application and based on side-information provided by the system designer. The framework provides several algorithms for automatic temporal and spatial mapping. Analysis of the mapping result is used to identify potentials for system refinement.

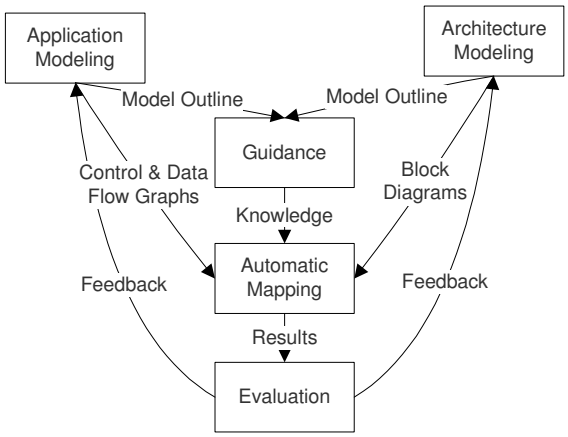

Figure 1. Framework structure overview.

3.1. Application and Architecture Modeling

Due to the variety of approaches to exploit inherent parallelism there is a variety of languages available, too. Some are designed for plain modeling (e.g. UML), some for simulations (e.g. SystemC) and some for concurrent programming (e.g. OpenMP).

Since our framework does not perform functional simulations we designed it to be independent of the modeling language. This is possible because:

1. Our mapping methodology does not use concrete implementations but only abstract information such as cycle counts and other data provided within the guidance step. This enables mapping and system evaluation without having to implement functional behavior at this design stage.

2. A lot of languages roughly implement the concept of nodes communicating over ports via edges with some additional control flow between the nodes. In case of applications this defines the control and data flow graph (CDFG) in case of architectures it represents the block diagram of the system.

This concept of nodes communicating over ports via edges is implemented as a C++ library providing easy access to common graph properties. The library supports multiple levels of abstraction within one model. During the modeling phase each node may be refined to get more detailed results. Refinement results in nodes representing subgraphs at a lower level of abstraction.

Right now we have implemented front-ends for processing UML activity diagrams (Fig. 2), applications and architectures given in the Y-Chart markup language (YML) [2] and some internally developed representations of applications and architectures. Front-ends for C and C++ are currently under development, Matlab and FDL [13] front-ends are planned for the near future and interfaces for other languages can be integrated at any time enabling fast reactions to evolving standards in modeling at any level of abstraction.

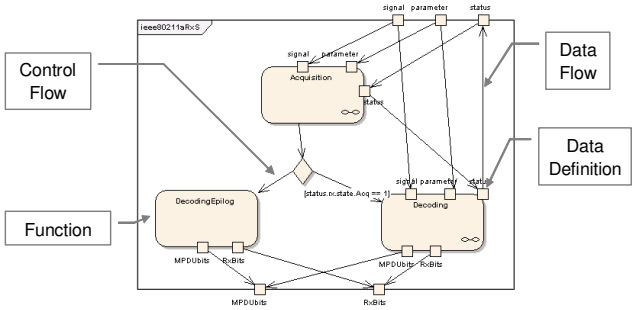

Figure 2. Example for a CDFG of a part of 802.11a modeled with UML activity diagrams.

3.2. Guidance

The guidance data base contains implementation specific information like cycle counts or energy consumption for mappings of an algorithm to target architectures. These numbers are known from previous designs or have to be estimated by an experienced system designer. To reduce the design space, numbers are only provided for reasonable mappings. E.g. it is not a good idea to map a Viterbi immediate to a RISC processor.

Simple HTML or Excel tables contain the guidance information for the mapping. Figure 3 shows an example with modules of an IEEE 802.11a receiver where the third to fifth column represent cycle counts on different processor types. Mappings to non-suitable processor types are excluded by simply leaving the respective cell blank.

3.3. Automatic Mapping

The automatic mapping step forms the central part of our framework. During the mapping process, an algorithm assigns elements of the application graph to elements of the architecture graph and adds properties such as starting time and latency (Fig. 4).

	A	B	C	D	E
1	Mapping Table for ieee80211aRxS				
2					
3	Level	Function	EVP	XCE	ARM
4	0	ieee80211aRxS			
5	1	DecodingEpilog	10		23
6	1	uml:DecisionNode			1
7	1	Acquisition			
8	2	AcquisitionInnerProlog	12		
9	2	AcquisitionMain			
10	3	AcquisitionMainInnerProlog	11		
11	3	ieee80211aSRxAcquisition	45	35	86

Figure 3. Cycle count annotation of 802.11a for the NXP Jeome Testbed.

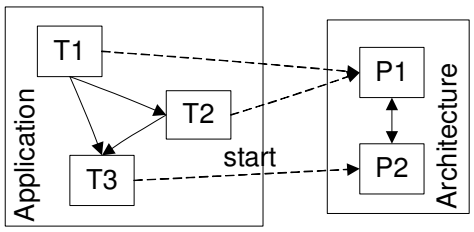

Figure 4. Task mapping.

There are several objectives such as performance or power consumption when designing MP-SoCs. Against todays trend toward multi-objective optimization (MOO) techniques we selected a lexicographic optimization, because given n objectives MOO generates a n-dim. pareto-optimal front. Since people tend to think sequentially, selection of one of the solution is done usually by looking successively at one dimension at a time. A step-wise lexicographic optimization which reduces the feasible set of solutions successively exactly reflects this behavior.

As many applications in the signal processing domain have at least semi-hard real-time constraints, performance is a binary criterion. Hence the first objective should be minimum latency to check if the system can meet real-time requirements. Moreover mapping for latency gives valuable hints about bottlenecks and capacity utilization lacks in your system leading to more significant improvement hints than mapping for other objectives would do.

In the first run we perform a mapping for an architecture with given kinds of processors, but unlimited number of processors using a slightly modified Dijkstra algorithm [3]. Within this we implemented an as soon as possible (ASAP) as well as an as late as possible (ALAP) scheduler.

After this step further mapping strategies for the desired objectives are applied. This process is split into task, data and data transfer mapping. To evaluate different task mapping strategies for the optimization of schedules with real-time constraints we implemented a mapper based on list scheduling supporting earliest deadline first (EDF), ASAP and least laxity (LL) modes and a mapper based on 2-dim. strip packing [11]. Further strategies such as dynamic sequence clustering [12] to exploit data locality are under development.

Note that within our framework each corner-case of the application is treated simultaneously. Worst-cases are identified automatically. This is accomplished as follows: If two tasks are mutually exclusive, the mapping algorithms are allowed to schedule these tasks on the same resource.

The ability to plug in different mappers at each stage allows evaluation and modification of the real-time operating systems mapping algorithms that will be executed on the designed system.

3.4. Evaluation and System Refinement

The mapping result is very similar to a project plan. Therefore we generate a mapping report in MS Project XML format (Fig. 5). This allows for convenient analysis of the mapping results. Bottlenecks of applications and architecture can be identified easily. Based on this information the system can be improved and reevaluated with the mapping framework.

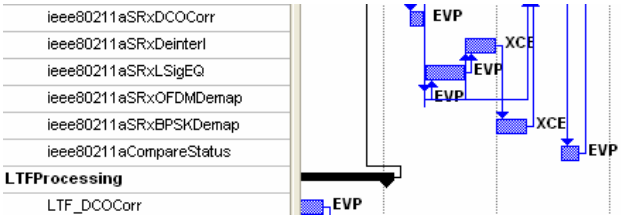

Figure 5. Mapping result analysis with MS Project.

4. Example

To give an example how refinement can improve system performance we applied load analysis (Fig. 6) of different signal processing algorithms on the SAMIRA vector DSP [7]. Although the framework is designed to deal with tasks on function level and processors within a heterogeneous MP-SoC, the methodology can be applied to processor instructions of a VLIW processor accordingly. In this example we will treat the vector DSP as MP-SoC and its instructions as tasks. This allows us to test our methodology with rather large CDFGs and a ”MP-SoC” with 21 processors, 5 memories and a sophisticated interconnection network.

As shown in Fig 6, the immediate has a very high load. Therefore, the idea is to improve the system by adding a second immediate. Due to the separate and abstract description

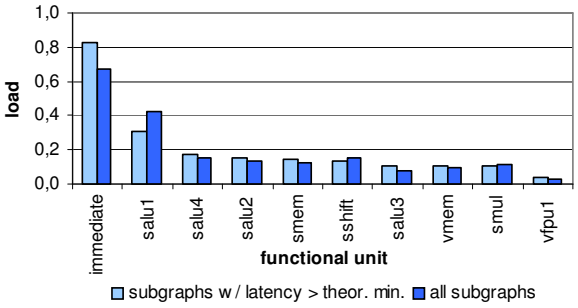

Figure 6. Load analysis of a set of signal processing algorithms on the vector DSP.

of applications and architecture this can be accomplished easily. Figure 7 shows the result for the improved system. When optimizing for other objectives such as energy consumption or die size the idea would be to eliminate one of the salus.

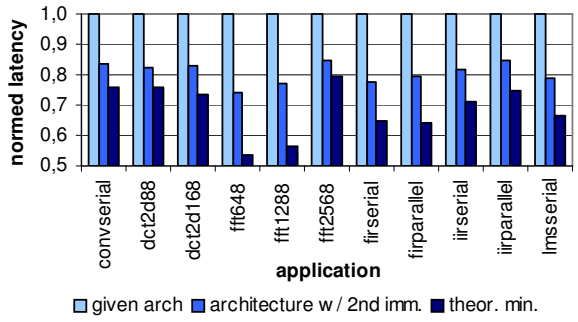

Figure 7. Result for system improvement induced by mapping result analysis.

For further refinement of the system the automatic mapping can be applied again to evaluate and refine the improved system.

5. Conclusion

This paper gave a brief overview over our mapping framework for guided design space exploration of heterogeneous MP-SoCs. The framework enables a fast and efficient evaluation of a given MP-SoC at a very early design stage via automatic mapping without the need for simulations or implementations. We showed exemplary, how the result of this automatic mapping can be utilized for system evaluation and directed improvement.

6. Acknowledgment

This research is supported by NXP within the project MxMobile Multi-Standard Mobile Platform of the German Federal Ministry of Education and Research (BMBF).

References

[1] P. Bellens, J. M. Perez, R. M. Badia, and J. Labarta. CellSs: a programming model for the Cell BE architecture. In *Proc. ACM/IEEE Conference on Supercomputing (SC'06)*, pages 86–96, 2006.

[2] J. Coffland. *YML Users Guide*. Univ. of Amsterdam, 2006.

[3] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[4] C. Erbas, S. C. Erbas, and A. D. Pimentel. A multiobjective optimization model for exploring multiprocessor mappings of process networks. In *Proc. IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'03)*, pages 182–187, 2003.

[5] B. Kienhuis, E. Deprettere, K. Vissers, and P. van der Wolf. An approach for quantitative analysis of application-specific dataflow architectures. In *Proc. IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'97)*, pages 338–349, 1997.

[6] E. A. Lee. Overview of the Ptolemy project. Technical memorandum UCB/ERL M03/25, University of California, Berkeley, CA, 94720, USA., 2003.

[7] E. Matúš, H. Seidel, T. Limberg, P. Robelly, and G. Fettweis. A GFLOPS vector-DSP for broadband wireless applications. In *Proc. IEEE Custom Integrated Circuits Conference (CICC'06)*, pages 543–546, 2006.

[8] M. Oyamada, F. Wagner, M. Bonaciu, W. Cesario, and A. Jerraya. Software performance estimation in MPSoC design. In *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC'07)*, pages 38–43, 2007.

[9] A. D. Pimentel, C. Erbas, and S. Polstra. A systematic approach to exploring embedded system architectures at multiple abstraction levels. *IEEE Transactions on Computers*, 55(2):99–112, 2006.

[10] V. Reyes, W. Kruijtzer, T. Bautista, G. Alkadi, and A. Núñez. A unified system-level modeling and simulation environment for MPSoC design: MPEG-4 decoder case study. In *Proc. Design, Automation and Test in Europe (DATE'06)*, pages 474–479, 2006.

[11] B. Ristau and G. Fettweis. Mapping and performance evaluation for heterogeneous MP-SoCs via packing. In *Proc. International Workshop on Systems, Architectures, Modeling, and Simulation (SAMOS'07)*, pages 117–126, 2007.

[12] T. Yang and A. Gerasoulis. A fast static scheduling algorithm for DAGs on an unbounded number of processors. In *Proc. ACM/IEEE Conference on Supercomputing (SC'91)*, pages 633–642, 1991.

[13] S. Zhong, C. Dolwin, B. Steinke, and A. Dröge. A OMA DM based software defined radio proof-of-concept demonstration platform. In *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'07)*, 2007.