Technical University of Denmark

Exploration of a digital audio processing platform using a compositional system level performance estimation framework

Tranberg-Hansen, Anders Sejer; Madsen, Jan

Published in:
2009 IEEE International Symposium on Industrial Embedded Systems

Link to article, DOI:
10.1109/SIES.2009.5196193

Publication date:
2009

Document Version
Publisher's PDF, also known as Version of record

Citation (APA):
Tranberg-Hansen, A. S., & Madsen, J. (2009). Exploration of a digital audio processing platform using a compositional system level performance estimation framework. In 2009 IEEE International Symposium on Industrial Embedded Systems (pp. 54-57). IEEE. DOI: 10.1109/SIES.2009.5196193

Exploration of a Digital Audio Processing Platform Using a Compositional System Level Performance Estimation Framework

Anders Sejer Tranberg-Hansen and Jan Madsen
Technical University of Denmark
DK-2800 Kgs. Lyngby, Denmark
e-mail: {asth,jan}@imm.dtu.dk

Abstract—**This paper presents the application of a compositional simulation based system-level performance estimation framework [1], [2] on a non-trivial industrial case study. The case study is provided by the Danish company Bang & Olufsen ICEpower a/s and focuses on the exploration of a digital mobile audio processing platform. A short overview of the compositional performance estimation framework used is given followed by a presentation of how it is used for performance estimation using an iterative refinement process towards the final implementation. Finally, an evaluation in terms of accuracy and speed of simulations is discussed based on the presented design flow applied to the case study in question.**

I. Introduction

The mixture of tight time-to-market constraints and an increasing design complexity often associated with the design of embedded systems of today makes the risk of sub-optimal implementations inherently evident. A major reason for this is the difficulty of getting feedback of the consequences of a design choice before the system has been realized physically or at least described at a very low level of abstraction. This makes the experience of the designers of the system a key element in the early design-phases of a system and at the same time severely limits the possibilities of exploring the design space. In this paper, a compositional framework for performance estimation of embedded systems [1], [2] is applied to a non-trivial industrial case provided by the Danish company Bang & Olufsen ICEpower a/s. The case study illustrates how the framework can be used for quantitative performance estimation using a simulation based approach and a successive refinement of models.

The framework is component based and quantitative performance estimation is done through simulations. The framework allows performance estimation to be carried out throughout all design phases ranging from early functional to cycle accurate and bit true descriptions of the system. The key strengths of the framework are the obtainable accuracy, the high flexibility and the great compositional refinement possibilities achieved by allowing components described at different levels of abstraction to co-exist and communicate within the same model instance. This is possible because a separation of the specification of functionality, communication, cost and implementation is used (this somewhat resembles the ideas of [3]) combined with the use of an interface based approach.

II. Related Work

A number of modelling methods and frameworks related to performance estimation and exploration of embedded systems have been seen within recent years.

Artemis [4], and the sub-project Sesame [5], focus on performance estimation of stream based applications. Application models generate traces of events as input to the architecture model in an approach slightly similar to the approach taken in this framework. However, the functionality of an application is modelled in the application model and only the resource access, latencies, and communication constraints etc. are modelled in the architecture model. In our case, both the functionality of applications, resource access, communication channels, etc. can be modelled within the architecture model, implying that the execution of the application model reflects the actual implementation without any modifications simply by changing the architecture model.

MILAN [6] and Metropolis [7] both aim to provide a framework in which models can be described at multiple levels of abstraction and gradually refined. They both allow multiple models of computations to co-exist within the framework. However, the means are different. MILAN requires that the different simulators are integrated into a common generic modelling framework whereas Metropolis focuses on defining common communication semantics which allow the different models to communicate. In contrast to these generic approaches, our method aims at providing support for models described at different levels of abstraction and a gradual refinement of these using a unified model of computation in order to reduce the communication overhead and simplify the required control logic during simulations.

III. System Level Performance Estimation

This section gives a brief overview of the performance estimation framework and presents its major elements. The details of the framework will be discussed more thoroughly as the case study is presented in the next section.

The framework, illustrated in figure 1, is related to what is known as the Y-chart approach [8]. However, in this case the application model is refined in its own iteration branch as step one, verifying the functionality of the application model only. When this step has been performed, the application model is left unchanged and only the mapping and platform model are being refined in step two. If the application model needs to be changed, it implies that the functionality of the application has changed. Hence, a new iteration of step one is required in order to verify the new functionality before repeating step two.

System level performance estimation is carried out using a *system model* composed of an *application model* mapped explicitly to the processing elements of a *platform model*. The

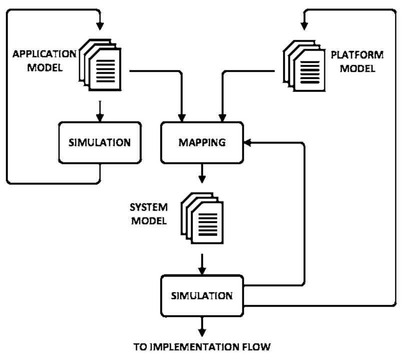

Figure 1. Overview of the proposed framework.

simulation of the system model makes it possible to produce quantitative performance estimates of the execution of the specified application on the chosen platform.

A. Application Models

Application models capture the functionality of the application only and should not rely on any implementation specific details unless a specific desire to do so exists, e.g. in some cases when the target platform is already fixed. The application models are composed of an arbitrary number of parallel, executable components referred to as *tasks*. Tasks communicate through abstract buffers making communication explicit and implementation independent.

The framework operates with three categories of tasks: *Functional* tasks, *mixed* tasks and *compiled* tasks. The categories do not constrain the level of abstraction used to describe the tasks but are used solely to express how the functionality of the task is being modelled. Functional tasks are unmapped tasks which model both the control flow and data operations within the application model with no costs associated. Mixed tasks model the control flow of the given task in the application model and only the data manipulating operations of interest in the platform model. Finally, compiled tasks model both control flow and data operations in the component model of the processing element onto which the task runs.

B. Platform Models

The platform model represents the target architecture and is composed of one or more component models. The platform model also specifies how the components are inter-connected and, thus, how they are allowed to communicate. There are no restrictions on how inter-component communication is modelled nor on the level of abstraction that is used implying that, in principle, all types of inter-component communication methods are supported.

The components of a platform model are implemented as service models [1], [2] and can represent both the virtual elements of the target platform, such as operating systems or middleware, and the actual physical hardware components such as memories and processing elements.

C. System Models

When the tasks and abstract buffers of an application model are mapped to the component models of a platform model, a system model is obtained which, by means of simulations, allows performance metrics to be extracted enabling a quantitative analysis of the system. It is *not* a requirement that all tasks of an application model are mapped to the processing elements of a platform model. In the case where a task of an application model is unmapped, only the functional behaviour of the tasks is modelled and no performance estimates are associated with that particular task.

IV. Case-study: Exploration of a Digital Audio Processing Platform

In this case study, a mobile audio processing platform is considered. The case study is provided by the Danish company Bang & Olufsen ICEpower a/s which is working within the field of audio power conversion e.g. for the mobile market segment. Currently, an existing version of the audio processing platform exists in the market and is sold in very high volumes.

The mobile audio processing platform is comprised of a digital front-end and a class D amplifier including the analogue power stage on-chip. The platform offers stereo speaker and stereo headphone audio processing resulting in a total of four audio channels being processed. In this case study, the focus will be on the digital audio processing part of the platform only, in which all audio processing is done within a single clock domain. The input interfaces, mixing and sample rate conversion are considered one combined abstract entity from which audio samples are being sourced at a fixed rate determined by the sample rate of the system. Similarly, one processed sample must be available at the outputs at each sample rate period in order to ensure normal operation. The ratio between the sample rate of the system and the clock-frequency of the audio processing clock domain expresses the real-time constraint of the application, i.e. how many clock cycles can be used for audio processing per sample.

The objective of the exploration is to optimize the execution platform onto which the application runs in terms of silicon area and power consumption, both of which need to be minimized. Due to the very high production volumes of the system, both the cost and performance of the system are critical elements in order to obtain commercial success. Traditionally, the choice of implementation has been between a platform based on a DSP processor or a fully dedicated hardware implementation. It seems obvious that both solutions have severe drawbacks either with respect to efficiency or with respect to the level of flexibility. Thus, a third option has been considered at the company and an experimental application specific processor referred to as the *SVF*-processor has been developed. The SVF processor is optimized to execute the type of algorithms needed in the application of the current case-study, has a relatively small silicon footprint, a very shallow pipeline and is programmable by offering 37 different instructions. The major problem with the three types of solutions is that even though a number of conclusions seems obvious based on experience and intuition, they cannot be verified until a very late stage in the design process, and so, answering the question, *"What is the best suited platform for the given application under the given constraints?"* is not straightforward in the early design phases.

In the following, it will be explained how the presented framework was used in order to help answer the question based on early quantitative performance estimates produced by the framework.

A. Application modelling

The first step in order to start the exploration of the platform, using the proposed framework, is to construct an application model. The application model is constructed from a specification of the application in Matlab and captures the functional behaviour of the application in a number of tasks as well as specifies the communication requirements of the individual tasks explicitly, without any assumptions on the implementation, following the principle, on which the framework is founded, of separating the specification of functionality, communication, cost and implementation.

The audio processing consists of four different algorithms and supports the processing of a total of four audio channels allowing individual stereo speaker and stereo headphone processing. The functional behaviour of the application model was verified through simulations of the application model consisting of functional tasks only. The results were compared to the specification of the algorithms in Matlab and verified. This high level functional application model serves as the functional reference in the refinement steps towards the final implementation.

B. Quantitative Performance Estimation

In order to generate quantitative performance estimates, the tasks and buffers of the application model must be mapped to the components of a platform model creating a system model.

In the current case-study, three different types of platforms were considered based on either the DSP processor, the SVF processor or a dedicated hardware implementation. Due to area constraints of the system, the number of feasible platforms were limited to the platforms shown in table I. For each of the platforms listed in table I, a platform model was constructed.

Table I
RELATIVE AREA OF THE INVESTIGATED PLATFORMS.

Platform	Description	Relative Area
HW	Dedicated hardware	1.0
SVFx1	1 SVF ASIP	0.6
SVFx2	2 SVF ASIP's	1.3
SVFx3	3 SVF ASIP's	2.1
SVFx4	4 SVF ASIP's	2.9
DSP	1 Audio DSP	4.0

In the exploration of the platforms, the first set of quantitative performance estimates were focusing on an execution time analysis of the platforms only. At this point, the objective was to investigate the utilization of the processing elements of the platforms at different clock frequencies in order to find the lowest clock frequency at which the platforms could be run and still execute the application within the real-time constraint of the application.

The first step in the quantitative performance estimation process was to construct service models of the different types of processing elements for use in the specified platform models. In this case-study, the architecture of the processing elements was already fixed; hence a detailed latency based model could be constructed of each processing element. The latency models do not include a modelling of the actual functionality but only the resource access and latency of each service without any modelling of data dependencies.

The application model was then manually mapped to the platform models consisting of the latency based service models. The latency based service models do not model the actual functionality implying that the control flow of the application must be handled in the application model and the tasks mapped to the individual latency based service models must belong to the category of mixed-tasks. This group of system models is hence named **mixed latency**.

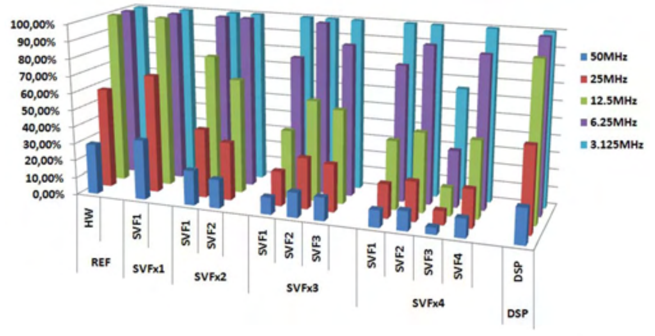

Figure 2. Platform utilization vs. clock-frequency.

The utilization ratios extracted from the six different platforms are shown in figure 2 for the different clock frequencies investigated. In the cases where the application is requiring more computational cycles than those offered by the processing elements, implying that the real-time constraint of the application cannot be met, the utilization ratio has been set to 100% in the figure. From the figure it can be seen that the clock frequency of the dedicated hardware implantation can be no lower than 25 MHz in order to fulfil the real-time constraint of the application. The results of figure 2, combined with the area estimates of the platforms, showed that the SVF based platforms were particularly interesting in the performance/flexibility trade-off, having the possibility of achieving half the clock frequency of the hardwired implementation with only a minor increase in the silicon area in the case of the **SVFx2**-platform. The results also indicated that the clock frequency could even be reduced to one fourth of the hardwired implementation in the case of the **SVFx4**-platform at the expense, however, of a tripling of the area. The platform based on the general audio DSP proved not to be attractive due to a high area and medium performance compared to both the hardwired and application specific platforms.

In order to verify the initial results, more detailed service models of the SVF processor were constructed and used for verifying the utilization ratios and the functional correctness of the results. The detailed version includes a bit true modelling of the functionality of each service and is cycle accurate.

The four SVF based platforms were refined to use the new bit true and cycle accurate SVF service models utilizing the compositional properties of the framework and the tasks of the application models, which were mapped to the SVF service model processing elements, were now modelled as *compiled*-tasks. These system models are referred to as **compiled** system models. The compiled tasks are represented as a service request image generated using the existing compiler infrastructure associated with the SVF processor. In this way, a one-to-one correspondence with the physical execution of the application can be obtained. Furthermore, the platform models were refined to include service models representing FIFOs modelling direct point-to-point connections between the processing elements.

The results showed that it was actually the case that the **SVFx2**-platform makes it possible to lower the clock frequency to one half of the hardwired version. The results make the **SVFx2**-platform a promising alternative to a hardwired implementation when the area estimates are also taken into account. Even the **SVFx1** platform seems competitive with an area equivalent to the hardwired platform. However, this platform does not experience the benefit of halving the clock frequency as was the case with the **SVFx2** platform.

C. Accuracy

In order to relate the quality of the performance estimates produced by the framework in the current case-study, an RTL implementation of the **SVFx1**-platform was created in the hardware description language **Verilog** referred to as the **RTL** model in the following. The simulations performed, using the RTL model, were then compared with the results obtained from the performance estimation framework.

Table II shows the estimated number of cycles used to process a stereo audio channel produced by the framework for the **mixed latency** and the **compiled SVFx1**-system model and estimates extracted from the RTL model simulations. The table shows that the cycle estimates obtained from the **mixed latency** model, in which only the latency based service models are used, is *not* cycle accurate. The cycle estimates produced by the **mixed latency** model, are in general, too optimistic. This is caused by the fact that the latency based model does not take data dependencies into account. In the other range of the scale, in terms of accuracy, the table also shows the cycle estimates of the refined **compiled** model in which a cycle accurate and bit true modelling of the components was used. In this case, the cycle estimates are identical with the estimates obtained from the RTL model as can be seen from the table.

Table II
ESTIMATED NUMBER OF CYCLES FOR THE PROCESSING OF 3000 SAMPLES IN ONE AUDIO CHANNEL AT THREE LEVELS OF ABSTRACTION.

	Mixed Latency	**Compiled**	**RTL**
A	84,034	102,034	102,034
B	123,041	129,384	129,384
C	33,011	33,011	33,011
D	153,051	168,056	168,056
Total	393,137	429,143	429,143

The constructed RTL model was also used to make a comparison of the functional results produced by the framework using the **compiled** version of the **SVFx1**-platform. The functional comparison showed that the audio streams processed from the system model were 100% identical to the processed audio streams from the RTL model. Furthermore, the time required to describe the model using the presented framework is much less than the time required to describe the equivalent RTL model due to the higher level of abstraction used. More importantly, it is significantly faster to modify a model of the framework in case of bug fixes or functionality extensions.

D. Simulation speed

Table III shows the measured simulation speeds expressed as cycles per second for the individual system models investigated. Simulations were performed on an 2.0 GHz Intel Core 2 Duo processor with 2GB RAM. What is most interesting is the big speed-up seen when comparing the simulation speed of detailed bit true and cycle accurate version of the **SVFx1**-system model as opposed to the equivalent RTL simulation. The **SVFx1**-system model runs at approximately 20 million cycles/second with all algorithms enabled, including data logging, and the functionally equivalent RTL description runs with approximately 15 thousand cycles/second resulting in a speed-up of more than 1000x.

Table III
OBTAINABLE SIMULATION SPEED OF THE INVESTIGATED SYSTEM MODELS.

	Mixed Latency	**Compiled**	**RTL**
HW	21,9M	N/A	N/A
SVFx1	21,7M	19,9M	15,324
SVFx2	18,6M	15,8M	N/A
SVFx3	17,2M	13,8M	N/A
SVFx4	15,9M	12,4M	N/A
DSP	20,0M	N/A	N/A

V. CONCLUSION

In this paper the usage of a compositional performance estimation framework was illustrated through a non-trivial industrial case study provided by the Danish company Bang & Olufsen ICEpower a/s. The case-study illustrated the practical usage of the proposed framework and how system models can be composed of a mixture of cycle accurate models and models described at higher levels of abstraction which is one of the strengths of the proposed framework. Furthermore, it was shown for the SVF based platforms that cycle accurate and bit true descriptions of platforms can be constructed and used as virtual platforms providing simulation speeds which significantly outperform simulations at register transfer level while still being functionally equivalent.

VI. ACKNOWLEDGEMENTS

This work was supported by DaNES (Danish National Advanced Technology Foundation) and ArtistDesign.

REFERENCES

[1] A. S. Tranberg-Hansen, J. Madsen, and B. S. Jensen, "A service based estimation method for MPSoC performance modelling," *International Symposium on Industrial Embedded Systems*, pp. 43–50, 2008.
[2] A. S. Tranberg-Hansen and J. Madsen, "A service based component model for composing and exploring MPSoC platforms," *International Symposium on Applied Sciences in Bio-Medical and Communication Technologies*, 2008.
[3] K. Keutzer, A. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, "System-level design: Orthogonalization of concerns and platform-based design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, pp. 1523–1543, 2000.
[4] A. Pimentel, L. Hertzbetger, P. Lieverse, P. van der Wolf, and E. Deprettere, "Exploring embedded-systems architectures with Artemis," *Computer*, vol. 34, no. 11, pp. 57–63, 2001.
[5] A. Pimentel, C. Erbas, and S. Polstra, "A systematic approach to exploring embedded system architectures at multiple abstraction levels," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 99–112, 2006.
[6] A. Bakshi, V. K. Prasanna, and A. Ledeczi, "MILAN: A model based integrated simulation framework for design of embedded systems," *SIGPLAN Not.*, vol. 36, no. 8, pp. 82–93, 2001.
[7] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli, "Metropolis: An integrated electronic system design environment," *Computer*, vol. 36, no. 4, pp. 45–52+4, 2003.
[8] B. Kienhuis, E. Deprettere, K. Vissers, and P. Van Der Wolf, "An approach for quantitative analysis of application-specific dataflow architectures," *Proceedings IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 338–349, 1997.