

Exploitation of UML 2.0–Based Platform Service Model and SystemC Workload Simulation in MPEG-4 Partitioning

Jari Kreku, Matti Eteläperä and Juha-Pekka Soininen
VTT Electronics
Kaitoväylä 1, Oulu, Finland
Jari.Kreku@vtt.fi

Abstract

Performance evaluation in an early phase of system design is a crucial part of system optimisation and validation. We present a method for combining UML–based application workload models with hardware models written using the SystemC language, and introduce a layer of platform service models between the application and hardware architecture models. The modelling approach is validated with a case study consisting of MPEG-4 encoder partitioning for OMAP 5912 architecture. The average error between the simulations and the measurements is about 12%.

1 Introduction

The complexity of hardware platforms is increasing rapidly as a result of silicon technology improvements [1][2], thus providing more flexibility and expanding the design space. At the same time, the demands of the functionality implemented on the platforms – i.e. video playback, 3D games and such – are becoming more stringent. Efficient exploitation of resources and a high level of integration are essential in embedded systems and the main design problems are the selection of architecture alternatives and mapping of functionality. On the other hand, the embedded application space is relatively limited and fine-tuning of applications and architecture properties is possible, which simplifies application mapping and makes the resulting systems more effective.

Performance evaluation in an early phase of system design is a crucial part of system optimisation and validation. Overall performance consists of both speed and resource utilisation, and the factors that affect the selection of the performance evaluation method include the maturity of the software and hardware, the achievable accuracy, and the evaluation time. The existing techniques used for performance evaluation are analytical modelling, different simula-

tion approaches, and monitoring of execution [3][4]. However, analytical approaches [5] only provide limited accuracy for the evaluation of complex platforms and a mature system is required for monitoring, disallowing early evaluation. The main obstacle for using any of the simulation-based approaches has been simulation performance [6].

UML is a general purpose modelling language that is already widely used in software engineering, whereas SystemC [7] is a popular C++–based hardware modelling language for transaction-level simulations. It seems natural to combine the strengths of the two languages [8] to simplify modelling and help increase the abstraction level, especially on the software side, for which SystemC alone does not have extensive support.

In this paper we extend the modelling approach presented in [9] and [10] by using UML 2.0 for modelling software applications and the software services of platforms, e.g. operating system resources. The novel idea is to increase the abstraction level of workload modelling by introducing a platform service layer between the software application and hardware models; SystemC workload models are generated from the UML descriptions and combined with a SystemC hardware architecture model for simulating the entire system.

The organisation of the paper is the following: Section 2 describes our modelling approach and Section 3 presents a modelling experiment consisting of OMAP5912 architecture and MPEG-4 encoding. Section 4 concludes the paper.

2 Modelling approach

It is already common that a number of different software applications are running concurrently in mobile terminals or other embedded devices. The set of applications is controlled by internal or external events, e.g. user actions, and timing. The applications are constructed of groups of functions and their control and data flow. The functions may be specific to the application or be parts of lower level platform

software, e.g. operating system services or library functions. For example, a simple C-language application may want to open a file for reading by using the `fopen()` function in the C library, and the implementation of that function is likely to call the corresponding OS function.

In the end, however, all applications consist of streams of hardware architecture-specific control, data processing, and memory access instructions. The characteristics of the application models should preferably resemble those of the execution traces as closely as possible — but the trace information is not available for early evaluation because the software and hardware would have to be mature enough that execution is possible. Furthermore, a low-level simulation approach like that would inevitably lead to long simulation times. Thus the control, timing and instruction distribution information in the execution trace has to be approximated with a higher-level modelling approach.

In our approach the software applications, their control, and scheduling are modelled with UML 2.0. The application models are mostly based on calls to *platform service models (PSM)*, which correspond to the software (or special hardware) services provided by the platform. These PSMs represent the middle layer in the model hierarchy by exploiting the resources provided by the hardware model. In the previous example the model of the C-language application would trigger the PSM of the `fopen()` function, which, in turn, could call other PSMs or use the hardware model directly. The benefit of the PSM layer is faster and higher-level application modelling and potential for reuse.

The related application models and PSMs are grouped into larger workload blocks [10] based on, e.g., the length of the operating system time slice for scheduling, which is handled in three phases: in the control phase the next workload block is selected for execution; in the execution phase the workload block is simulated, e.g. for the duration of the time slice; finally, in the timing phase the scheduler may wait for a specified time or an event, if necessary, before starting over from the control phase with the next workload block. This method allows us to model the effect of control and dependences between workload blocks so we do not have to model and simulate the functionality of the software. We also avoid that all the workloads would always be “executed” as quickly as possible, which would cause the load of the most stressed processor to be 100% in every simulation. Of course, if the applications were executed on the real hardware, the processes or threads corresponding to the workload blocks would occasionally be pre-empted after their time slices have expired. Pre-empting is deliberately not taken into account in the models since we are not trying to model execution latencies in detail — the emphasis is on the average load caused by the software.

The hardware architecture model is written with the SystemC language. The purpose of the HW model is to provide

Table 1. The most important HW model parameters and their values.

Component	Clock frequency	CPI
ARM	192 MHz	1.50
DSP	192 MHz	0.75
DDR SDRAM	96 MHz	N/A

the workload models with resources, simulate the timing of execution, and monitor the resource usage. The resources are provided through a workload / hardware model interface consisting of a few functions intended for load modelling and controlling the parameters of the HW model [10]. The most important interface functions are `read()`, `write()` and `execute()`, which notify the HW model of memory accessing and data processing operations.

Since the workloads are modelled with UML and the hardware architecture with SystemC, the two languages have to be combined somehow to enable simulation of the modelled system. We have solved this in that the interface between the workload and hardware models is also described in UML for the use of the workload models. In addition, some essential SystemC structures, like `sc.module`, need to be depicted in UML — the type of the interface between the workload and hardware models determines how much of SystemC needs to be visible in UML. Therefore, we are able to generate C++ code from the UML workload models using a UML tool and link it with the SystemC hardware model with minimal manual tweaking. The hardware model and SystemC structures described in UML are marked as external models so that no code will be generated for them in this phase.

3 Case study

As a case example we studied the partitioning of an MPEG-4 encoder for the OMAP 5912 multiprocessor platform as available in the OSK5912 OMAP Starter kit by Spectrum Digital. The OMAP5912 processor includes an ARM926EJ-S general purpose processor and a Texas Instruments TMS320C55x digital signal processor. The video encoder used in the task was the open source libavcodec library with the ffmpeg front-end, which we partitioned to utilise both the DSP and the ARM sides of the OMAP5912. Based on the hardware accelerators available in the DSP and on the profiling results of the MPEG-4 encoder, the discrete cosine transformation (DCT) algorithm was mapped to the DSP and two partitioning alternatives were considered for the sum of absolute difference (SAD) calculation inside motion estimation.

The workload model of the MPEG-4 encoder was cre-

ated with a UML tool and consists of I-frames and P-frames, with, e.g., quantisation and motion estimation algorithms. The application model utilises C55x::DCT and C55x::SAD platform service models (Figure 1), which correspond to the hardware accelerators available in the DSP. Figure 2 displays a statechart diagram of the I-frame model and describes how the I-frame is using the C55x::DCT PSM. The DCT model itself is constructed of the workload / hardware model interface calls. The OMAP architecture was modelled with SystemC as described in [9] and the most important parameters of the model are shown in Table 1.

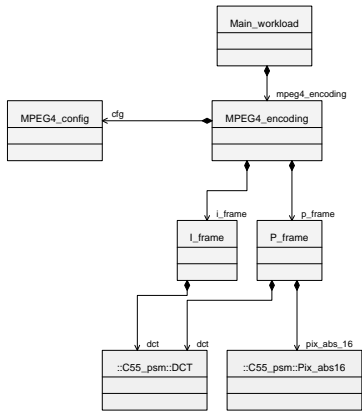


Figure 1. UML class diagram of the workload models consisting of several case-specific classes and two PSMs.

The load information for the workload models was obtained in several steps. First of all the MPEG4 encoder was profiled on a desktop PC in order to identify the most loading functions and to derive the number of calls for each — the functions responsible for 80% of the total load were chosen to be modelled. Since both the software and the hardware platform were available in this case, we compiled the encoder for the OSK5912 kit and estimated the number of `read()`, `write()`, and `execute()` calls for the models via disassembly. The control flow governing the use of these functions was estimated by inspecting the source code of the encoder. The DSP side DCT and SAD PSMs were obtained from literature supplied by Texas Instruments. Respectively, the hardware accelerator implementations of these functions take 240 and 156 cycles to complete [11], and the PSMs consisted of a corresponding number of `execute()` calls.

For validating the simulation approach, the `ffmpeg` encoder was executed on the OSK5912. Montavista Linux was compiled for the ARM side and DSP/BIOS was used on the DSP as the operating system. The commonly used foreman sequence was used as the video clip with a frame

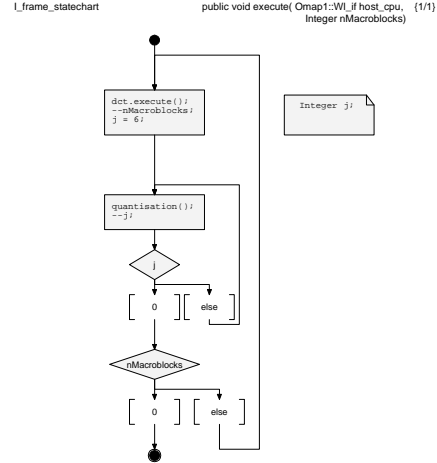


Figure 2. Statechart diagram of the I-frame workload model.

rate of 15fps and QCIF resolution. The encoder, the source clip and the encoded clip were all put on a ramdisk device on the OMAP processor.

The DCT transformation was performed by the hardware acceleration unit of the DSP; all other parts of the encoder were executed on the ARM. Data transfers between the two processors were implemented using the DSP/BIOS Link API, utilising inter-processor interrupts and shared memory. It was noticed that the communication functions of the API included a notable overhead on single macro block size data transfers. The total delay of these transfers was measured as 48 milliseconds per frame. The overhead was measured separately and then taken into account in the simulation models.

During the simulation runs we measured the frame rate and utilisation of the processors. In addition, we were also able to estimate the performance potential of the architecture with a lower communication overhead and the effect of the SAD algorithm partitioning with simulations. For the low overhead cases we assumed that entire frames were transferred at once between the processors instead of single macro blocks. Two types of measurements were made from the case study setup: the maximum encoding frame rate was measured and the DSP utilisation was obtained with the tools provided by the DSP/BIOS operating system on the C55x core. However, the ARM processor was the performance bottleneck in this case and fully loaded.

Comparison of the simulated and monitored performance reveals that the modelling approach is accurate. While there was virtually no difference between the obtained DSP load values, the frame rate achieved with the OSK5912 kit was 24% lower than in the simulator (Table

Table 2. Simulated cases with their results and corresponding measurements.

Use case	Simulation		Measurement	
	FPS	DSP load	FPS	DSP load
All-MPU (reference)	15.9	0%	17.8	0%
High overhead, DCT on DSP	8.3	18.9%	6.7	18.5%
Low overhead, DCT on DSP	13.8	11.0%	N/A	N/A
Low overhead, DCT and SAD on DSP	16.5	27.0%	N/A	N/A

2). Simulations with the low communication overhead naturally show a clear improvement in the frame rate, though the performance still is not clearly better than in the reference all-MPU case. The DSP load for the low/DCT combination was 11% and 27% for the low/DCT+SAD setup. Most of the 19% DSP load in the high/DCT case resulted from the inter-processor communication. The alternative partitioning with the SAD algorithm executed in the DSP gives a minor increase in achieved frame rate but a 2.5-fold increase in DSP load — so the choice between the two alternatives depends on what other programs the DSP should be executing and the load caused by them.

4 Conclusions

A straightforward method for using UML 2.0 based workload models in combination with a SystemC hardware architecture model was presented in this paper. A platform service layer was introduced to workload models in order to reduce the amount of modelling work by increasing the level of abstraction and by providing more opportunities for model reuse.

The same approach was applied to a case study consisting of the ffmpeg MPEG-4 encoder and the OMAP 5912 hardware platform. Two encoder partitioning alternatives were analysed with simulations and the accuracy of the approach was validated by executing the encoder in the real hardware. According to the results, the presented approach is useful as an early phase performance evaluation method as the difference between the simulations and measurements is only 12% on average.

5 Acknowledgements

This work is supported by Tekes (National Technology Agency of Finland) and VTT under EUREKA/ITEA contract 04006 MARTES.

References

- [1] "International technology roadmap for semiconductors," 2003, available at <http://public.itrs.net>.
- [2] L. Benini and G. D. Micheli, "Networks on chip: a new soc paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, January 2002.
- [3] J. Hennessy and D. Patterson, *Computer Architecture — A Quantitative Approach*, 3rd ed. Morgan Kaufmann Publishers, 2003, 883 p.
- [4] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*. John Wiley & Sons, Inc., 1991, 685 p.
- [5] D. Petriu, C. Shousha, and A. Jalnapurkar, "Architecture-based performance analysis applied to a telecommunication system," *IEEE Transactions on Software Engineering*, vol. 26, no. 11, pp. 1049–1065, November 2000.
- [6] J. Staunstrup and W. Wolf, *Hardware/Software Code-sign: Principles and Practice*. Kluwer Academic Publishers, 1997, 395 p.
- [7] T. Grötter, *System Design with SystemC*. Kluwer Academic Publishers, 2002, 240 p.
- [8] E. Riccobene, P. Scandurra, A. Rosti, and S. Bocchio, "A soc design methodology involving a uml 2.0 profile for systemc," in *Proceedings of the Design Automation and Test in Europe Conference*, March 2005, pp. 704–709.
- [9] J. Kreku, J. Penttilä, J. Kangas, and J.-P. Soininen, "Workload simulation method for evaluation of application feasibility in a mobile multiprocessor platform," in *Proceedings of the Euromicro Symposium on Digital System Design*, 2004, pp. 532–539.
- [10] J. Kreku, T. Kauppi, and J.-P. Soininen, "Evaluation of platform architecture performance using abstract instruction-level workload models," in *International Symposium on System-on-Chip Proceedings*, 2004, pp. 43–48.
- [11] *TMS320C55x Image/Video Processing Library Programmer's Reference*, Texas Instruments, March 2003, 58 p.