ON DESIGN-TIME PERFORMANCE PREDICTIONS OF OBJECT-BASED MPEG-4 VIDEO APPLICATIONS

[1]*Egor Bondarev,* [2]*Milan Pastrnak,* [1,2]*Peter H.N. de With and* [1]*Michel R.V. Chaudron*

E.Bondarev@tue.nl
[1]Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[2]LogicaCMG Eindhoven

ABSTRACT

Development costs and time-to-market of software-intensive media processing systems can be reduced when correct decisions are made at an early *design phase*. However, the real-time requirements imposed on these systems, such as frame skipping and latency limitations, can only be validated after system implementation, e.g. at the *test phase*. To avoid system redesign for ensuring real-time performance, a system performance analysis should be done as soon as possible. This paper addresses a scenario simulation approach for performance predictions already at the design phase. This approach is presented through a real case study, for which we developed an advanced MPEG-4 coding application. The benefits of the approach are threefold: a) high accuracy of the predicted performance data; b) an efficient real-time software-hardware implementation of a system, because the generic computational costs become known in advance, and c) improved ease of use because of a high abstraction level of modelling. Experiments showed that the prediction accuracy of the system performance is within 10%, while the prediction accuracy of the time-detailed processor usage (performance) does not exceed 30%. However, the real-time performance requirements are sometimes not met, e.g. when other applications require intensive memory usage, thereby imposing delays on the decoding data retrieval from memory.

1. INTRODUCTION

Multimedia systems development is characterized by a continuous evolution of new and existing standards and by extreme time pressure for developing them. Therefore, industrial focus is shifting from improving implementation techniques towards improving system design methods. Advanced design methods allow evaluation of the system functionality and performance already at very early development phases, which reduces technical risks and minimizes time-to-market. Advanced design techniques become imperative especially for time-critical software-intensive media processing systems. The real-time requirements imposed on these systems, such as limitations on frame rates and latency, can only be validated when the system is implemented. A performance test failure can cause a complete iteration in the development process. To avoid system redesign for ensuring performance requirements, we concentrate on the accurate *prediction* of the system performance properties at the early design phase and analysis of real-time behaviour of the system.

In the area of system architecture evaluation a lot of research has been done on the performance prediction. The survey [1] gives an extensive overview on the current prediction-enabling techniques. The modular performance analysis [2] allows exploring software and hardware alternatives for performance optimization, though it does not give high prediction accuracy. Spade [3] provides an example of the simulation-based performance analysis methodology for signal processing systems. It addresses the software and hardware co-design area, but still does not enable prediction of real-time properties. In [4] and [5], we find efficient performance prediction techniques in the component-based software domain, which are based on component modelling and static analysis. However, for complex applications, the models become too large for understanding and efficient use in analysis. Last but not least, PRIMA-UML methodology [6] applies queuing networks and extends UML with a model for system performance validation. For our problem statement we want to satisfy free aspects simultaneously: (1) real-time properties prediction, (2) high accuracy, (3) ease of use of the design method. None of the above-mentioned proposals satisfies these three aspects at the same

time.

In this paper we propose an easy-to-use scenario simulation approach for predicting with high accuracy, timing and resource usage properties of the designed system. The approach is based on three concepts: a) *models* for the system component's behaviour and resource usage b) *execution scenarios* of the complete system, in which the resources are potentially overloaded, c) *simulation* of these scenarios, resulting in timing behaviour of the design system. The modelling enables a high abstraction level description, thus making this prediction technique more efficient. Additionally, the application of the execution scenarios reduces the system state space for exploration, and helps to focus only on the relevant time-critical execution configurations, in which the risks of system overload are high. As a design paradigm we have adopted component-based software architectures, because of its structural flexibility for large complex applications and the enabling possibilities for software reuse. The MPEG-4 decoder case study presented later in this paper revealed that it is indeed possible to predict real-time properties with sufficiently high accuracy. The case study also uncovered system management issues, such as influence of the operating system and memory management aspects.

The remainder of this paper is as follows. In Section 2, we address a Robocop software component-based architecture that was deployed for the MPEG-4 coder case study. Section 3 presents the proposed scenario simulation approach enabling performance predictions at the design phase. In Section 4, we describe the MPEG-4 case study in more detail, which was used for the approach validation purpose. The validation results are given in Section 5. Finally, Section 6 presents conclusions from the case study, and suggests future directions for research.

2. COMPONENT-BASED ARCHITECTURE (CBA)

We have adopted the Robocop Component-Based Architecture (CBA) [7] for conducting our research on predictable software design, because it offers software reuse and speeds-up the development. The Robocop architecture is developed for middle-ware in consumer devices, with the emphasis on robustness and reliability. The Robocop CBA is similar to CORBA and COM, but enables more efficient realization of real-time and performance constraints via modelling techniques. For example, a component designer can supply a *component behaviour model* along with the component executable code. The application designer composes an application from the number of components and can predict the application behaviour, using the set of such component behaviour models.

The Robocop CBA is highly efficient for multimedia processing systems and Systems-on-Chip in particular. First, it allows decomposing the processing software into functional blocks and then mapping these blocks on the architecture in the optimal way. Second, the supporting Robocop Run-Time Environment has built-in Quality-of-Service implementation. Finally, the freedom of defining the necessary types of models allows addressing not only processing usage, but also other attributes important for multimedia systems (e.g. memory and bus load).

A software system built according to the CBA paradigm consists of a number of components bound together. A component is a reusable piece of software with strictly defined functionality. The component functionality is provided by a set of *interfaces*. A provided interface implements a number of methods, necessary for component operation. Each component may have two types of interfaces: a) *provides* interface and b) *requires* interface. The binding between components in the application is made via couples of provides-requires interfaces.

3. PREDICTION-ENABLING SCENARIO SIMULATION APPROACH

In this paper we exploit our recently proposed scenario simulation approach [8], featuring early analysis at the design phase of a representative multimedia processing application, i.e. an advanced implementation of MPEG-4 coding. The approach is a four-step strategy (see Figure 1) containing the following steps: 1) specification of a component *behaviour* and a *resource model* of a real-time aware component; 2) design (composition) of a real-time application from the set of components and specification of an *application scenario model* for each of the resource-critical execution scenarios; 3) joint compilation of these three types of models into a model representing the execution tasks of this application; 4) simulation of the task

execution, resulting in a *task execution timeline*. This timeline allows us to extract the real-time, memory- and bus-related performance properties of the application. Comparing the predicted data with the application requirements, we can quantitatively assess the design of the application.

4. MPEG-4 CODING APPLICATION

For evaluation of our real-time design technique, we conducted a case study for which we developed a state-of-the-art MPEG-4 coding application. We used the full specification of the standard, featuring arbitrary-shaped video objects. Applying the scenario simulation approach, we predicted the performance and real-time property of the designed decoder. After the integration phase, we compared the predicted results with the real execution data.

Let us now provide some details of advanced video object-oriented processing. Figure 2 depicts the computation graph for arbitrary-shaped video object decoding as used in MPEG-4. Similar to MPEG-2, objects are divided into macroblocks (MB). The diagram shows special processing stages for decoding the shape and motion of the video objects, in addition to the usual texture decoding. Each decoding job iteration starts with macroblock type decoding (MBtype Dec). The ShapeMC stage computes the motion compensation for the Shape part and provides referenced MB for the Context Arithmetic Decoding (CAD) stage. The CAD provides an MPEG-4 compliant shape representation of the macroblock. The shape for the macroblock is represented by 16x16 binary subimage, which is sent to the outputs of the Shape job. The Coded Block Pattern (CBP) extracts information about parts of the texture that need to be updated.

Texture decoding involves five steps: Motion vectors Decoding (MvD), IDCT Coefficients Decoding (Coeff Dec), Texture Motion Compensation (TextureMC), Inverse Quantization (IQ) and Inverse DCT (IDCT). Stages MBtype Dec, CAD, CBP, MvD, and Coeff Dec are executed sequentially, because each stage depends on another stage, to specify the next position in the input bit stream. Therefore, we introduced a loop surrounding these stages, indicating the order between them.

The above-mentioned MPEG-4 decoder application was decomposed into four Robocop CBA components (Decoder, Reader, Renderer and Buffer). Each component was accompanied with their corresponding *resource* and *behaviour* models. The resource model specifies resource requirements per individual component operation, while the behaviour model also describes the underlying calls to other operations per component operation. These model structures are further specified in [8]. The components were (graphically) composed into a new CBA structured MPEG-4 application. The composition process consists of two activities: a) instantiation of components; b) binding the instances via their interfaces (see Figure 3). As indicated by the figure, the Reader instance is bound to Buffer1 to store the read video frames. The Decoder instance is bound: a) to Buffer1 to get these frames and b) to Buffer2 to store the decoded pixels. The Renderer instance is bound to Buffer2 to get the decoded pixels and render them on the display. After composition, a *critical scenario* has been selected. We predefined a normal execution mode with the resolution 340×280 as a critical scenario. Note that it is possible to select multiple scenarios. Furthermore, a *scenario model* was specified for the chosen mode (see Figure 3). It consists of an application composition structure and number of control inputs (stimuli). Those inputs (event, periodic timer, interrupt, etc) lead to the frame-periodic execution of one of the component operations. In our case, we designed three periodic timers that call Reader.readFrame(), Decoder.decodeFrame() and Renderer.renderFrame() operations, thereby establishing a pipe-and-filter execution architecture.

5. EXPERIMENTS AND VALIDATION RESULTS

According to Figure 1, the next step involves the joint compilation of the scenario model and the resource/behaviour models of all components, resulting in a pool of application tasks. The execution of these tasks was simulated with a rate-monotonic virtual scheduler. The resulting execution *timeline* is depicted in Figure 4. The timeline indicates all running tasks of the decoder, their predicted *start* and *completion times*, as well as deadlines. The general performance property (processor utilization) was derived from the timeline data.

After prediction at the design phase, we integrated the real-time aware components into a real MPEG-4 decoder application and executed this on a Linux/32 platform. We aimed at obtaining two types of data: a) timeline data (which task is executed at what moment); b) processor utilization data.

First, we compared the acquired timeline data with the predicted execution timeline. The analysis showed

a slight difference in the predicted-*vs.*-real execution moments of the tasks. This difference is explained by numerous OS kernel activities (page-faults, timestamps, i/o calls) interrupting the decoder execution. Thus, the approach does not allow accurate prediction of the starting/completion times of individual tasks. However, the predicted patterns of the task execution coincided well with the real execution patterns. It is clear that the OS kernel activities should be integrated in our design approach.

Second, we analyzed the accuracy of the predicted performance properties. The prediction accuracy / tolerance on the average processor utilization appeared to be about 10%, and proved to be reasonably stable. The accuracy / tolerance on the *time-detailed* processor utilization (granularity = 1 sec) varied within 30%. We have found that a reason for the temporally high processor-usage is the varying number of macroblocks (arbitrary object shape) for decoding. This observation resulted into the concept of an input-parameter-dependent resource modelling. This new concept is being explored at the moment.

Finally, we discovered an interesting phenomenon in the decoder timing behaviour under memory overload conditions. In the Linux OS, a concept of virtual memory is used for dealing with memory overload. This virtual memory is implemented by *page swapping* - releasing memory slots by storing memory data on a disk. Returning this data back to memory upon request takes relatively long time. In the MPEG-4 decoder this causes serious latency (1-2 frames) in the processing, because the data cannot be read instantaneously. The resulting phenomenon is that latency requirements are not met, while the CPU usage is very low. The conclusion is that processor, memory and bus usage need to be analyzed jointly to identify the aspects that hamper predictable system operation.

6. CONCLUSION AND DISCUSSION

We have exploited a scenario simulation approach that enables prediction of real-time properties of an advanced software-intensive MPEG-4 coding system. The average real-time behaviour of the MPEG-4 decoder can be predicted with high accuracy (within 10%). The prediction accuracy on the time-detailed processor utilization varied within 30%. As an extra benefit, the timing results give detailed performance information at the design phase. For obtaining the time-detailed accuracy, we have found that integration of a timing model for system activities into our modelling technique is indispensable, because our experiments revealed a large variation of starting and completion times.

The knowledge about the generic computational costs resulting from our approach, provides important guidelines for efficient software/hardware co-design of multimedia coding systems. The case study revealed that the processor, memory and bus usage need to be analyzed jointly, instead of processor analysis only. For future research, we study the execution of our CBA-based implementation on a programmable multiprocessor (multimedia) system with prediction of its real-time features and bus/memory usage.

7. REFERENCES

[1] Balsamo, S., Di Marco, A., Invenardi, P.: *Model-Based Performance Prediction in Software Development: A Survey*, IEEE Trans. Software Eng. 30 (2004) 295-310.

[2] Wandeler, E., Thiele, L., Verhoef, M.: *System Architecture Evaluation Using Modular Performance Analysis - A Case Study*, Proc. 1th ISOLA Symposium (2004).

[3] Lieverse, P., van der Volf, P, Vissers, K.: *A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems*, Journ. VLSI Signal Proc. Signal, Image and Video Proc., 29 (2001) 197-207.

[4] Wallnau, K.C.: *Volume III: A Technology for Predictable Assembly from Certifiable Components*, April 2003, CMU/ESI-2003-TR-009.

[5] Hissam, S.A., *et al.*,: *Packaging Predictable Assembly with Prediction-Enabled Component Technology*, November 2001, CMU/ESI-2001-TR-024.

[6] Cortellessa, V., Mirandola, R.: *PRIAM-UML: a performance validation incremental methodology on early UML diagrams*, Elsevier Science B.V., 02/2002.

[7] Public homepage of the Robocop project. http://www.extra.research.philips.com/euprojects/robocop

[8] E.Bondarev, J.Muskens, P. de With and M. Chaudron, *Predicting Real-Time Properties of Component Assemblies: a Scenario-Simulation Approach*, Proc. 30th Euromicro Conf., CBSE Track, 2004.

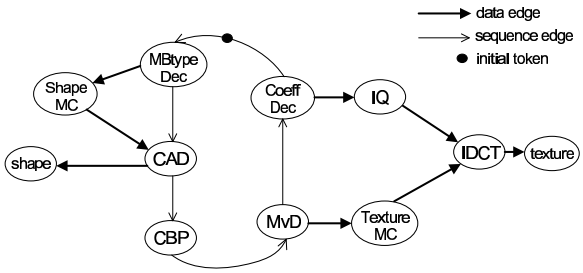

Figure 2: *Computation graph of arbitrary-shaped video object decoder.*

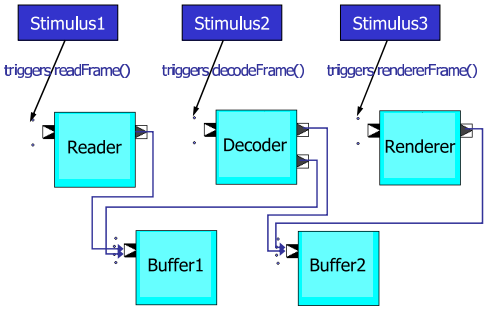

Figure 3: *Application scenario model.*

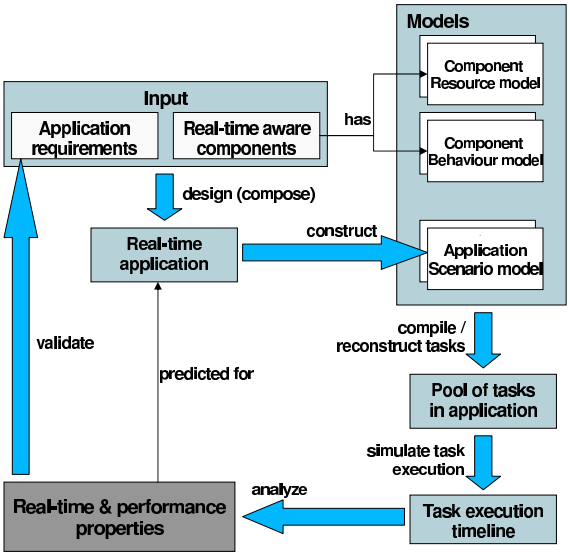

Figure 1: *Design of real-time video applications featuring simulation scenarios.*

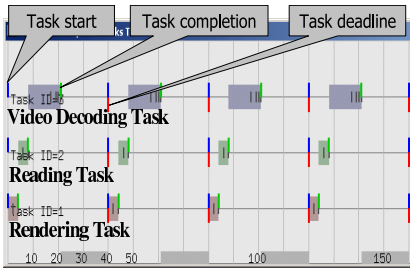

Figure 4: *Tasks execution timeline.*