motivated the use of a multiprocessor system-on-chip (MP-SoC) as a target platform for such advanced applications [1]. The efficient mapping of multiple object decoding onto such an architecture poses a management problem, requiring QoS control of the platform resources.

We propose to use a hierarchical QoS model that is discussed in the next section. The model distinguishes both details within an application and interaction between applications, because of multiple instantiations of objects and other applications running simultaneously in the same system. The main focus of this paper is providing the QoS model for individual instantiations.

One of the problems we also address is the possible scalability of shape-texture decoding for an individual object. In order to free resources in the platform for other objects or applications, we allow skipping of some of the decoding tasks at the cost of slightly reduced quality of the decoding process.

Most of the existing multimedia implementations on reconfigurable platforms under the condition of resource limitations sacrifice not fully decoded information at the moment of reaching a frame deadline. The result is that computation effort is put in decoding tasks that are never completed and used for the final reconstruction. For this reason, we propose a novel technique for skipping frames based on the *predictive parsing* of video headers (current techniques are based on resource adaptation after decoding of the full frame). As a consequence, we decide to perform only those decoding tasks that can meet their deadlines and thereby contribute to an increased video quality.

III. QoS Model for Applications on MP-NoC

The architecture of our QoS concept is based on two negotiating managers, instead of the conventional single resource manager. We distinguish the *Global QoS manager* that controls the total system performance involving all applications, and the *Local QoS manager* that controls an individual application within the assigned resources (see Fig. 1). Since the responsibilities of both managers are essentially different, it becomes apparent that a negotiation protocol for the overall system control will be needed.

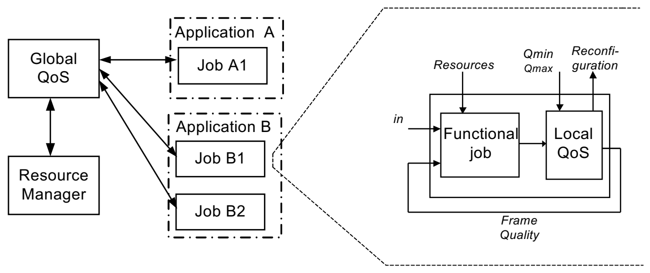

Fig. 1. Hierarchical QoS architecture view.

Each application is divided in jobs and the platform should support the execution of each job for implementation of the complete application. A job is split into different tasks, e.g. the MPEG-4 header parsing, the inverse quantization, IDCT, etc. In order to execute a job on a multiprocessor system, tasks are mapped onto processors and other resources.

The value of the above QoS architecture model for system quality control increases if an accurate execution model for the application mapping can be provided. In previous work, we reported on a reasonably accurate prediction model based on a linear parametrical equation at task level [1]. Unfortunately, not all of the required parameters are *a priori* available when the task starts its execution. In [4], we proposed a prediction technique that estimates resource usage based on a set of parameters available from video headers and a complementary set of parameters from the decoding of previous frames. The resulting task level model offers continuous control of the decoding process, hence at a finer granularity than current frame-based QoS systems.

The previously mentioned prediction supports the predictable behavior of our architecture and therefor enables a deterministic Quality-of-Service (QoS) from each job, independent of other jobs. To achieve this, we reserve resources for each particular job in the form of virtual processors and virtual connections. These virtual processors and connections are run-time assigned to the existing resources of the platform. This abstraction is important to obtain the worst-case model for the resource distribution in the case that each task is mapped onto a different processor of the platform, to obtain fully independent jobs execution.

IV. Application of QoS for Arbitrary Shaped Video Object Decoding

We have studied object-oriented MPEG-4 coding for the new proposed QoS concept. In MPEG-4, every Video Object (VO) is represented in several information layers, with the Video Object Plane (VOP) at the base layer. This VOP is a rectangular frame containing hierarchically lower units, called Macroblocks (MB).

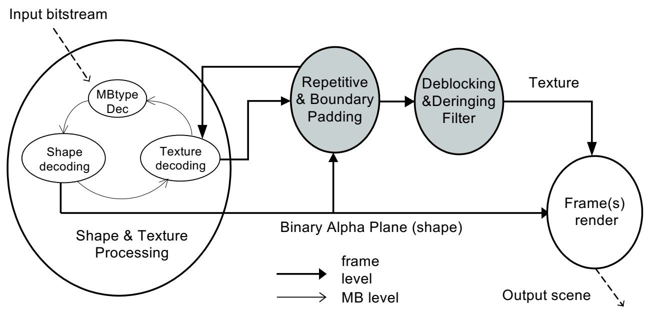
Input bitstream
MBtype Dec
Shape decoding
Texture decoding
Shape & Texture Processing
Repetitive & Boundary Padding
Deblocking &Deringing Filter
Texture
Frame(s) render
Binary Alpha Plane (shape)
frame level
MB level
Output scene

Fig. 2. The frame level for VOP shape-texture decoder.

Figure 2 outlines a distributed version of a computation model for an arbitrary-shaped video object decoder. In our previous work [1] we presented a model for the shape-texture decoding at the finest granularity (MB level). This

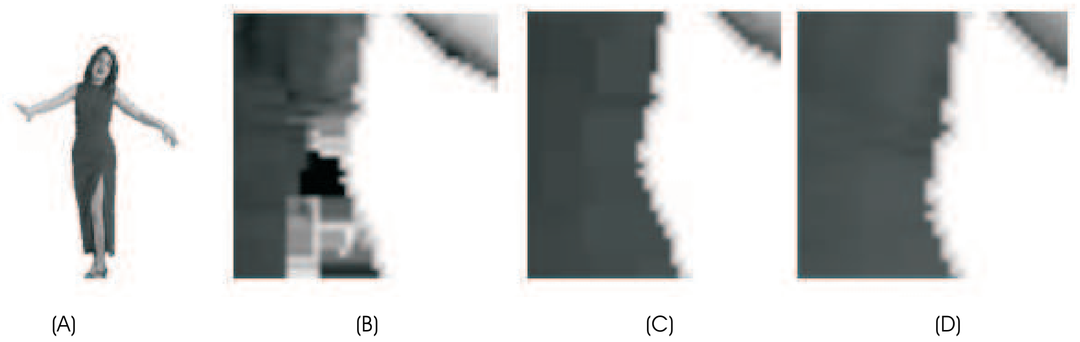

Fig. 3. Resulting quality after task skipping for the ”singer” sequence. (A) original VOP, (B) enlarged view without deblocking, deringing and padding tasks, (C) enlarged view without deblocking and deringing tasks, (D) fully decoded.

timing model is useful if mapping requires usage of different processors even at the MB level. We present here the graph for the complete decoding that starts with parsing the bitstream syntax and coded data and ends with the completely reconstructed scene. The graph was simplified for presentation reasons. The final visual scene can be composed of several VOs. In the presented graph, each task starts its execution when it has data on all inputs (as in Homogeneous Synchronous Dataflow Graphs).

The decoding starts with the *Shape & Texture Processing*, followed by *Repetitive & Boundary Padding*, then applying the *Deblocking & Deringing Filter* and providing the final shape and texture data to the *Frame renderer*. The renderer is a shared task and composes the original scene from the video background sprite and several VOs superimposed on it. The background sprite decoder and QoS for it is not discussed in this paper; the reader is referred to [5], [6] for more details. For the individual arbitrary-shaped VO, the complete task graph is instantiated. These independent instantiations output their results to the shared Frame Renderer.

V. Scalability of the VOP decoder

Figure 1 portrays the Local QoS connected to the functional part of the actual job. A runtime steering mechanism periodically observes the difference between the estimated resource requirements and the actually used resources. Based on the estimation error and input data characteristics, Local QoS handles two types of situations. First, short-term variations of the resource utilization may occur. To compensate for this in the case that some of the tasks are scalable, the Local QoS manager can change their local quality settings.

If this step does not sufficiently reduce the resource needs, it can disable some steps of the decoding process (e.g. a deblocking filter). Second, when the change in required resources has a long-term nature that needs control, e.g. the size of an object changes significantly, Local QoS can request the Global QoS management unit for reconfiguration of the task mapping and new reservation of resources.

For object-based video, the shape of a VO is the dominant information needed for the correct rendering of the final scene. If we have several objects, the correct shape information of a few objects can already help in the scene reconstruction. In the compressed data format, the shape is lossless encoded in the VOP bitstream, but is alternated with the texture information. For a scalable decoding with minimum amount of resources, we therefore propose at least the full decoding of the shape and the parsing of the texture coefficients without DCT decoding.

When decoded, the texture information is obtained by applying Inverse Quantization (IQ) and Inverse Discrete Cosine Transformation (IDCT). The second large block, Repetitive and Boundary Padding, combines the shape and the texture data and constructs the texture information for pixels that are transparent in the current VOP. This step can be skipped if the current frame is not a reference frame for consecutive frames. Furthermore, we can skip this task in case of low available resources. The effect of skipping Padding is portrayed in Figure 3(B).

The frame postprocessing consists of two steps: the deblocking filter and the deringing filter (see details in [5]). Either one or both of them can be turned off or on. Figure 3(C) and (D) illustrate the resulting quality of these filters. It can be noted that the degradation in quality is much smaller then the degradation due to deactivation of the Padding step. Table I shows the distribution of task complexities in percentage of the computation resources for different video sequences.

TABLE I
Distribution of task complexity of VOP decoding tasks.

Sequence	Shape & Texture Decoding	Repetitive & Boundary Padding	Deblocking & Deringing
Singer	72%	17%	11%
Dancer	78%	16%	8%
News	81%	13%	6%
Fish	76%	16%	8%
Average	76.75%	15.5%	8.25%

VI. QoS concept based on prediction

In this section we define an objective function for measuring the decoder quality. Afterwards, we provide an estimation-based algorithm for prediction of the skipping of frames when resources are limited. The algorithm explores the all possible combinations of VOP decoding tasks, for the given amount of resources. We choose the combination that yields the highest decoding quality based on the objective quality function.

In case that the task-skipping techniques described in Section V do not lower sufficiently the requested resources, the frame-skipping QoS technique has to take place. First, we follow the strategy of skipping B-VOPs. There is no further dependency on B-VOPs and the cost of decoding is high. Since this is obvious, we do not discuss this further. In the remainder of this paper we focus on the skipping of I-/P-VOPs. There is usually a set of VOPs depending on I-/P-VOPs (each P-VOP depends on an I-VOP or an earlier P-VOP, and each B-VOP on two I-/P-VOPs). Therefore, the skipping of these frames degrades the quality more drastically than the skipping of B-VOPs. This motivates us to exploit the algorithm for skipping I-/P-VOPs.

To evaluate the performance of QoS management, the performance is modeled into an objective function based on quality settings and deadline miss rate. If the Local QoS estimates a deadline miss for presentation, but the next frame depending on the decoded frame will meet the deadline, the decision unit of the Local-QoS manager executes the decoding and omits the presentation. The objective function for performance estimation is

$$f = 1 - (M_r + S_r) - N_r + 0.5N_{r1} + 0.25N_{r2}..., \quad (1)$$

where M_r stands for the rate of frames that were decoded but never presented, S_r is the rate of frames skipped completely and N_r represents the rate of frames that were presented but not on time. We can extend our function with a more detailed focus on frames displayed only one frame after the deadline (denoted as N_{r1}), or two frames after the deadline (N_{r2}), etc., to make our objective function more robust. For our experiments, the evaluation of our QoS concept is based on Equation (1). In the next subsection we provide our algorithm for frame skipping based on the timing-model prediction.

A. Estimation based algorithm for frame skipping

In this subsection we describe the algorithm used to investigate the option of quality adaptation by skipping some VOPs. It is not a straightforward decision how many and which VOPs should be skipped. On one hand, skipping more VOPs decreases the objective function f (the number of skipped frames S increases). On the other hand, it releases resources so that next VOPs can start decoding earlier and have higher chances to be presented earlier (M decreases).

The algorithm is presented below in the form of pseudocode. It returns the (sub-)optimal Boolean *skipping vector* **s** which has one element per upcoming VOP. We assign an element to be ‘true’ if the VOP is skipped. The initial settings with all frames set to be skipped is named **s***. The length of the vector is determined by the extent of the allowed *look-ahead* window. The input argument, called *progress*, is the current time remaining to the next VOP deadline. This parameter can be reported to the LQoS manager by the functional part of the job.

```
$\mathbf{s}^\star$ = LQOS_SKIP_VOP(progress)
1.  $\mathbf{ET}_{EST}$=PREDICT_ET(…);
2.  <$\Delta$f$^\star$,P$^\star_0$,$\mathbf{s}^\star$>=<0,0,(1,1,1…)>;
3.  while not STOP(…)
4.    $\mathbf{s}$ =SELECT_CANDIDATE(…);
5.    $\mathbf{Tc}$=COMPL($\mathbf{s}$, $\mathbf{ET}_{EST}$);
6.    <$\Delta$f,P$_0$>=F_OBJ($\mathbf{s}$,$\mathbf{Tc}$,progress);
7.    UPDATE(<$\Delta$f,P$_0$,$\mathbf{s}$>,<$\Delta$f$^\star$,P$^\star_0$,$\mathbf{s}^\star$>);
8.  end
```

Algorithm Decision on frame skipping.

Let us explain the algorithm in detail. In Line 1, the estimation (or prediction) of execution times **ET** for the look-ahead VOPs is obtained. The details of the corresponding procedure PREDICT_ET is out of scope of this paper, but one can predict execution times based on the resource allocation, previous history and a priori-information obtained from the video headers.

In Line 2, we initialize the data structure specifying the current solution, containing the increase Δ of the objective function f from Equation (1), the number of frames presented on time N_0 and the skipping vector **s**. We start with the simplest candidate solution to skip all frames (so the skipping vector contains all ‘1’s).

Lines 3-8 contain an iterative procedure that tries to improve the current solution. First another candidate skip vector is selected in Line 4. Only a feasible candidate may be selected: if it decides to skip a given VOP e.g. an P-VOP it skips also all VOPs depending on that VOP e.g. all the subsequent P-VOPs until the next I-VOP.

Then, given the skip vector **s** and execution time estimates **ET**, the absolute completion times are computed. Function F_OBJ compares the completion times with the deadlines. It determines the position of the deadlines from the current progress. F_OBJ computes the contribution to the objective function yielded by the candidate solution as well as the number of frames presented on time.

Line 6 compares the current skip vector with the candidate skip vector and chooses the one with the higher quality. If candidates are equal, the decision is taken based on the following ranking: a higher number of N_0, the value Δf and the number of skipped frames. We observed that putting N_0

at the highest priority makes the algorithm more stable in terms of frame skipping behavior.

Note that the current version of procedure COMPL in Line 5 ignores the fact that the memory buffer for output VOPs has bounded size and can get an overflow, which may result in postponing the completion of some VOPs.

VII. Experimental results of the skipping algorithm

We evaluated the algorithm from the previous section with the ”singer” sequence from the MPEG-4 standard group. The Figure 4 gives the actual execution of the MPEG-4 VOP decoder for the test sequence. The straight horizontal bottom line in the figure indicates the allocated resources. The varying solid curve above shows the actually needed computations, of which the average is clearly exceeding the available resources. Consequently, some of the frames of a group of frames (GOPs) are skipped for decoding. The dotted columns above the solid curve indicate the simulated value of the prediction error.

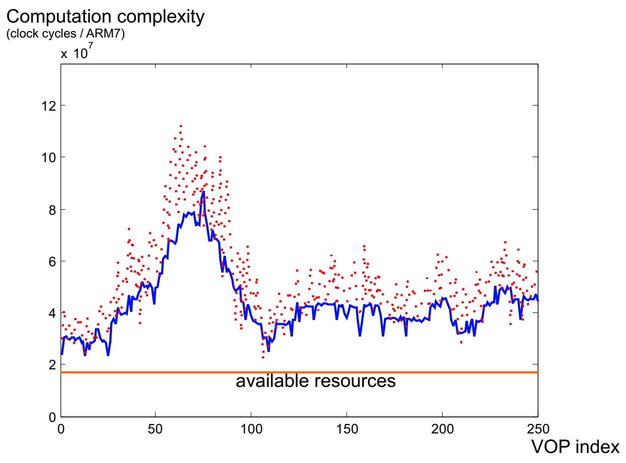

Fig. 4. Real execution of the VOP decoder in ”singer” sequence.

Figure 5 outlines the number of the skipped frames per GOP. We set the available resources to 50% of the required resources. The frame skipping algorithm initializes without prediction and performs a look ahead strategy to select the frames to be skipped. At the start, it adopts a very low resource level and therefore all frames in the GOP are skipped. After some GOPs, the curve reflects the varying demand for resources, but the average is about six skipped frames, which is exactly half of the GOP size (12 frames). This matches with the 50% of the resources being available.

The prediction error is influencing the behavior of our algorithm. With the growing prediction error, the value of the objective function becomes more broadly distributed (larger variance). This means that the number of skipped frames per GOPs fluctuates more.

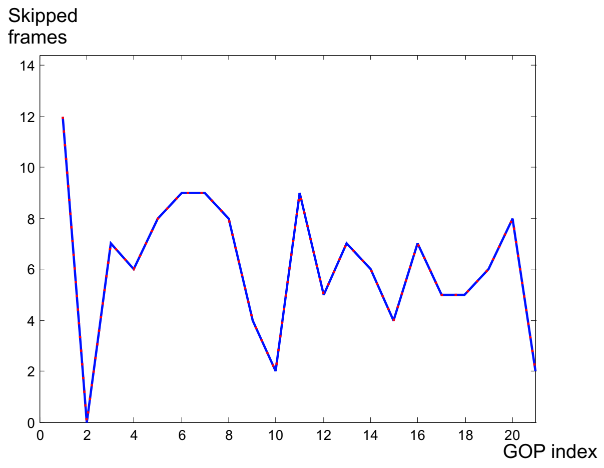

Fig. 5. Results of the algorithm under limited resources.

VIII. Conclusions

We have studied the mapping of object-oriented MPEG-4 decoding on a multiprocessor NoC. Since the amount of objects is unknown in advance and the decoding characteristics are highly variable, the guaranteed execution of all decoding tasks cannot be ensured. For this reason, we have proposed a new hierarchical QoS management system, featuring both intra and inter application control. Furthermore, it allows limitations in resources, thereby offering a scalable decoder implementation, provided that the decoder is designed with sufficiently fine-grained tasks. We have presented a performance estimation algorithm that predicts the actual decoding timing requirements and selects the combination of decoding tasks that yields the highest decoding quality, using an objective quality function. A novel element here is the focus on tasks that can meet their deadlines, instead of aborting uncompleted tasks and wasting the involved computations. We have shown that by task (de)activation we can gain in average up to 24% of resources. Summarizing, the proposed techniques drive to fine-grained usage of resources allocated to applications and simultaneously hide the dynamics in resource usage of those applications, while offering a scalable framework for execution.

References

[1] M. Pastrnak, P. Poplavko, P.H.N. de With, and D. Farin, “Data-flow timing models of dynamic multimedia applications for multiprocessor systems,” in *4th IEEE Int. Workshop System-on-Chip for Real-Time Applications (SoCRT)*, 2004.

[2] J. Bormans, N.P. Ngoc, G. Deconinck, and G. Lafruit, *”Terminal QoS: advanced resource management for cost-effective multimedia appliances in dynamic contexts” in Ambient intelligence: impact on embedded system design*, pp. 183–201, Kluwer Academic Publ., NL., 2003.

[3] A.C. Bavier, A.B.Montz, and L.L.Peterson, “Predicting MPEG execution times,” in *Proceedings of ACM SIGMETRICS'98*, 1998, pp. 131–140.

[4] M. Pastrnak and P.H.N. de With, "Data storage exploration and bandwidth analysis for distributed MPEG-4 decoding," in *8th IEEE Int. Symp. Consumer Electronics (ISCE)*, 2004.

[5] ISO/IEC 14496-2:199/ Amd 1:2000, *"Coding of Audio-Visual Objects - Part 2:Visual, Amendement 1: Visual Extensions"*, Maui, December 1999.

[6] Milan Pastrnak, Dirk Farin, and Peter H. N. de With, "Adaptive decoding of MPEG-4 sprites for memory-constrained embedded systems," in *26th Symposium on Information Theory in the Benelux*, 2005.

Milan Pastrnak received the M.S. degree in information systems from Zilina University, Slovakc Republic, in 1999. In 2002, he completed the post-graduation designers course of the Eindhoven University of Technology. He received PDEng degree in September 2002 on the project "Distributed Visualization and Simulation with Object-Oriented Networks".

Currently, he is a Junior Researcher at the Information & Distribution Technology, LogicaCMG Nederland B.V., The Nehterlands. He is a guest at the Video Coding and Architectures group, University of Technology, Eindhoven and he closely cooperates with Philips Research Laboratories, Eindhoven, The Netherlands. He is focusing on the application SW-HW co-design, heterogenous multiprocessor systems and system-level design.

Peter Poplavko got Bachelor degree in computer science from the National Technical University of Ukraine in 1998 and Master degree in electrical engineering from University of Technology in Eindhoven in 2001.

Currently he is a Ph.D. student at University of Technology in Eindhoven. He is a guest at Embedded System Architectures on Silicon group, Philips Research Laboratories, Eindhoven, The Netherlands. His main research interests are real-time software, system-level design and on-chip multiprocessor architectures.

Peter H.N. de With graduated in electrical engineering from the University of Technology in Eindhoven. In 1992, he received his Ph.D. degree from the University of Technology Delft, The Netherlands. He joined Philips Research Labs Eindhoven in 1984, where he became a member of the Magnetic Recording Systems Department. From 1985 to 1993, he was involved in several European projects on SDTV and HDTV recording. In this period, he contributed as a principal coding expert to the DV standardization for digital camcording. In 1994, he became a member of the TV Systems group at Philips Research Eindhoven, where he was leading the design of advanced programmable video architectures. In 1996, he became senior TV systems architect and in 1997, he was appointed as full professor at the University of Mannheim, Germany, at the faculty of Technical Computer Science. Since 2000, he is with LogicaCMG as a principal consultant and he is professor at the University of Technology Eindhoven, at the faculty of Electrical Engineering. He has written and co-authored numerous papers on video coding, architectures and their realization. In 1995, 2000 and 2004, he coauthored papers that received the IEEE CES Transactions Paper Award and SPIE paper awards. Dr. de With is a senior member of the IEEE, program committee member of the IEEE CES and ICIP, chairman of the Benelux community for Information Theory, scientific advisor of the Dutch Imaging school ASCII, IEEE ISCE and board member of various working groups.

Jef Van Meerbergen (M'87-SM'92) received the electrical engineering and the Ph.D. degrees from the Katholieke Universsiteit Leuven, Belgium, in 1975 and 1980, respectively.

In 1979, he joined the Philips Research Laboratories, Eindhoven, The Netherlands. He was engaged in the design of MOS digital circuits, domain-specific processors, and general-purpose digital signal processors. In 1985, he started working on application-driven high-level synthesis. Initially, this work was targeted towards audio and telecom DSP applications. Later, the application domain shifted towards high-throughput applications. His current interests are in system-level design methods, heterogenous multiprocessor systems, and reconfigurable architectures. He is the Associate Editor of *Design Automation for Embedded Systems*. He is a part-time Professor at the Eindhoven University of Technology, Eindhoven.

Dr. van Meerbergen is a Philips Research Fellow. His Phideo paper received the Best Paper Award at the 1997 ED&TC conference.