Multidimensional model of estimated resource usage for multimedia NoC QoS

Multidimensional model of estimated resource usage for multimedia NoC QoS

Milan Pastrnak
LogicaCMG Nederland B.V. /
Eindhoven University of Technology
Fac. Electr., SPS, VCA Group
Den Dolech 2, 5612 AZ Eindhoven
The Netherlands
M.Pastrnak@TUE.NL

Peter H. N. de With
LogicaCMG Nederland B.V. /
Eindhoven University of Technology
Fac. Electr., SPS, VCA Group
Den Dolech 2, 5612 AZ Eindhoven
The Netherlands
P.H.N.de.With@TUE.NL

Abstract

Multiprocessor systems are rapidly entering various high-performance computing segments, like multimedia processing. Instead of an increase in processor clock frequency, the new trend is enabling multiple cores in performing processing, e.g. dual or quadrapule CPUs in one subsystem. In this contribution, we address the problem of modeling the resource requirements of multimedia applications for a distributed computation on a multiprocessor system. This paper shows that the estimation of resource requirements based on input data enables the dynamic activation of tasks and run-time redistribution of application tasks. We also formally specify the optimal selection of the co-executed application with aim to provide the most optimal end-results of such streaming applications within one networks-on-chip (NoC) system. We present a new concept for system optimization which involves the major system parameters and resource usage. Experimental results are based on mapping an arbitrary-shaped MPEG-4 video decoder onto a multiprocessor NoC.

1 Introduction

Multiprocessor systems enable new features for computationally intensive applications, such as multimedia coding, video communication, etc. The parallel computation capacity of multiprocessor systems inherently offers simultaneous execution of several applications. The run-time distribution of different applications or subtasks of applications over the available processors of a target platform introduces an NP-hard problem.

The modeling of resource usage on platforms focused mostly on the usage modeling of computational resources with I/O components and priority-based scheduling techniques for real-time systems [3, 4]. Due to the characteristics of a distributed computation, we extend this one-dimensional approach covering the real-time characteristics of multimedia applications, into a multidimensional model concerning a plurality of resources, such as e.g. communication resources and the storage of data and instructions [2]. The usage of the architecture becomes more visible with multimedia applications having varying requirements on resources.

In this paper, we concentrate on a multimedia processing system featuring a *distributed* real-time computation with intelligent application modeling and Quality-of-Service (QoS) control. A novel aspect is to discuss *inter-application* behavior, where co-execution is combined with the intra-application view on resource estimation. The reservation of resources is also *scalable* for different quality-levels, depending on the actual input data. A second contribution of this paper is that we show the technique for a fast search of the optimal quality selection optimizing the overall output for the end user. We demonstrate this technique using an MPEG-4 video-object decoder algorithm that can instantiate up to four-object decoders within the available resources of a realistic clock-cycle true eight-processor system model.

2 QoS architecture and run-time estimation.

In order to provide fluent non-interrupted multimedia processing, we employ a design paradigm based on predictable mapping. The growing complexity of a final multiprocessor system asks for a system design with distributed control. Instead of one central system-level solution, we address a hierarchical approach that can better support the configurability of a target system.

The architecture of our QoS concept [1] is based on two levels of negotiating managers. We distinguish a *Global QoS manager* that controls the total system performance involving all applications, and a set of *Local QoS managers* controlling individual applications (see Fig. 1).

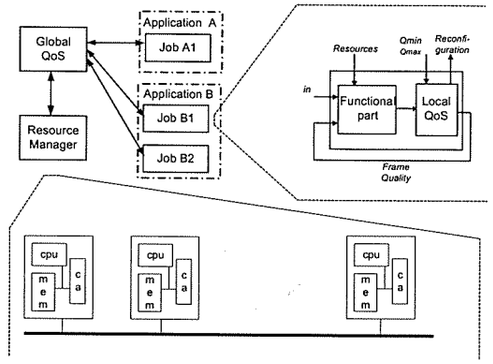
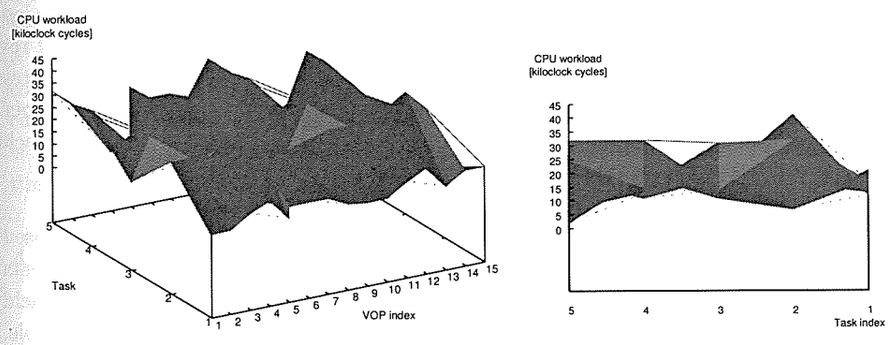


Figure 1: Hierarchical QoS architecture view.

Each application is divided into several jobs, and the platform should support each job for implementation by using the system resources and the software libraries. A job is split into different tasks, e.g. for an MPEG-4 decoder this would be split into the video 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 (memory, communication resources, computation, etc) and each job is divided in various communicating tasks. For each job i, the set of different quality levels is *a-priori* defined and in this work based on a finite set of possible qualities. For each job and a related quality level, we provide a detailed task description for the QoS subsystem management.

In our research, we study object-based video applications based on MPEG-4. These applications offer an independent set of tools used for decoding. The applications also allow the combination of several independent objects into one image, which naturally supports the possibility of Quality-of-Service management. An object-based video application is composed of several *jobs*, e.g. the background image decoding, audio decoding, and several instances of an arbitrary-shaped video-object decoder. For each job and related quality level, we provide a detailed data-flow graph and resource requirements description.

Run-time resource estimation is matching the algorithm characteristics of processing using specific input data. With respect to the modularity of our system design, we keep the details of algorithm at the job level. The estimation of resources is hence a part of the Local QoS module. The possible high variation in input data and the cost of reconfiguration are constraining our estimation to provide the maximum over a certain time interval. With respect to our experimental object-based video decoder, we set this interval to the length of a Group of Video Object planes (GOV).



(a) Resource estimation per each frame.

(b) Min-max view on the 15 VOPs of one GOV.

Figure 2: Data-level parallelism for processing of video texture.

Figure 2 portrays the overall estimation of resource usage for our case, where Figure 2a illustrates the estimation of each task involved in the decoding (video-header parsing, shape decoding, texture decoding, padding, and deblocking & deringing filters). The reader can observe that the computation needs can vary between 5-32k clock cycles. With respect to the above-mentioned granularity of a possible QoS system reconfiguration, we need to find the maximum of the resource requirements on the defined set of required resources for the given input data. We project the estimation time interval, in order to simplify the representation of *minimum/maximum* over the whole interval (in our experimental setup 15 VOPs). The *maximum* is required for reservation of resources for guaranteed quality level, while the *minimum* indicates the requirements that will be used over the whole period of time. The usefulness of the *minimum* for more advanced control provides many adaptation strategies for further exploration, which are however, beyond the scope of this paper.

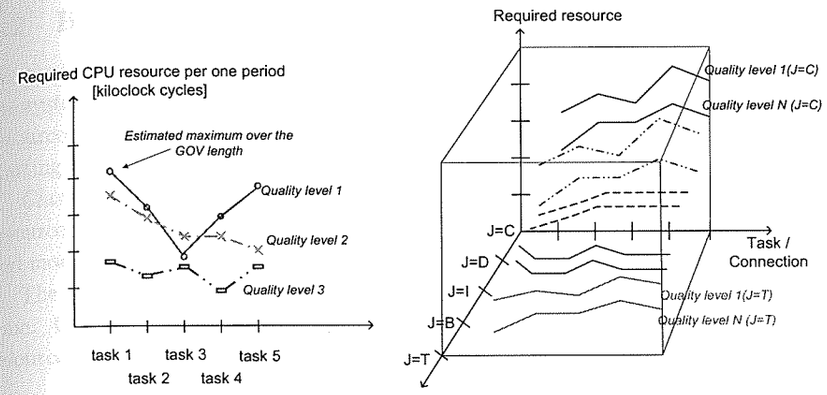
3 Resource modeling

In this section, we provide details of modeling the application with respect of computation, communication, and storage. The model of the required resources of an application is followed by the model of the available system resources.

3.1 Application consumption

Let us now outline the resource-usage model for one job. Formally, the range of possible quality settings mapped into a vector $\mathbf{q}$ of a job i leads to a job resource-consumption R_i for a resource type J, which is described by a function $R_{i,J}(q_i) = f_J(q_i, d_p)$, where d_p is an arbitrary input-data parameter that mostly influences the complexity of the computation. For example, in our arbitrary-shaped VO decoder, it is the size of a video object in terms of macroblocks. The function f specifies the requested amount of resources for a particular job. The resource type is $\mathbf{J} \in \{\mathbf{C}, \mathbf{D}, \mathbf{I}, \mathbf{B}, \mathbf{T}\}$, where $\mathbf{C}$ denotes the computation resources; $\mathbf{D}$ the data memory per task; $\mathbf{I}$ the instruction memory per task; $\mathbf{B}$ the required communication-port bandwidth; and $\mathbf{T}$ the bandwidth on each connection between a pair of tasks.

For one job i, we specify the quality setting, so that for a set of jobs the set of chosen quality values leads to a chosen quality vector $\mathbf{q_c}$, in which the vector components refer to the chosen quality settings of individual jobs. Similar reasoning can be held for the required resources of $R_{i,\mathbf{J}}$, leading to a vector of resources $\mathbf{R_J}$. For example, when J refers to computations only, $\mathbf{R_C}(\mathbf{q_c}, \mathbf{d_p})$ is representing the required vector of computations for a set of jobs at chosen quality level $\mathbf{q_c}$ and input-data dependence $\mathbf{d_p}$. The vector $\mathbf{R_C}(\mathbf{q_c})$ can be used for finding the accumulated computation costs per set of jobs executed at quality settings $\mathbf{q_c}$. If in the mapping this requirement is coupled to the available resources, the requirements can be evaluated.



(a) Model of different quality levels per resource (J=C).

(b) Multi-dimensional model of resources.

Figure 3: Resource consumption model of a job i.

Figure 3 illustrates the modeling of a job i that is scalable to three quality levels. Figure 3(a) portrays the computational requirements of that job. The points represent the maximum requirements over the reservation period (in our case GOV length). Figure 3(b) visualizes the resource estimation for different types of resources as described above.

3.2 Modeling of system resources

The system resources that are available for a job execution are modeled as described below. We model the available computation and storage resources per processing tile. If P denotes the set of processing tiles, then we denote total available resources as follows: computation resources of a tile as P_c, data storage capacity as P_d, instruction storage capacity as P_i, and communication port capacity as P_b. The communication provision is modeled per pair of processing tiles as the amount of data that communicated per time unit as $T_{pa,pb}$. The model of system resources is illustrated in Figure 4.

4 Optimization concept for multidimensional system control

4.1 Formal definition

Generally, the task-to-processor assignment is a search for the distribution of tasks with certain requirements over processors of the platform. There are several analytical approaches that solve this problem. However, most of them assume a shared-memory

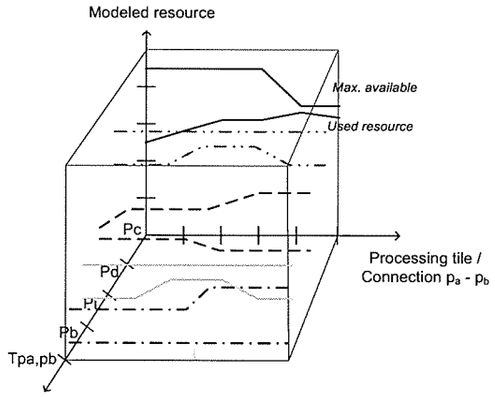


Figure 4: System resource model, with explicitly modeled resource consumption at run-time system.

model and one communication resource (like a bus). In our case, we will include a detailed model for communication resources per connection and also a *distributed* memory model. We also provide a formal definition of our optimization problem.
Let us now briefly formulate the optimization problem for the complete system and a set of jobs. We assume a set of processing tiles $\{p_1, p_2, ...\}$ with the following available resources per processing tile: number of clock cycles of the CPU, size of the data and instruction memory, bandwidth of the communication assist; and for every possible connection $\{p_m, p_n\}$ between involved processing tiles, the bandwidth and input/output FIFO buffer sizes. Let $\mathbf{R}_A$ be the collection of all vectors of all *available* resource types from all tiles of the platform, which are organized into a matrix. That matrix has different tiles P over the row index and resource types J over the column index. We define the benefit β_{i,q_i} as the contribution to the overall system value function (e.g. in our case quality). If we execute the job i at the quality level q_i, then the matrix $\mathbf{R}_{i,q_i}$ stands for all resource requirements of the job i at the chosen quality level q_i. Now, our optimization problem for QoS management of the complete system is formulated as

$$\max \sum_{i=0}^{N} \beta_{i,q_i} \quad \text{for } 1 < q_i < Q_{i,max}, \text{ subject to } \sum_{i=0}^{N} \mathbf{R}_{i,J,q_i} < \mathbf{R}_{A,J}.$$

It should be noted that $\mathbf{R}_{A,J} = \sum_P \mathbf{R}_A(P, J)$, so the sum of all tile resources of the same type (the sum of all elements of one column in the matrix $\mathbf{R}_A$.

4.2 Heuristic optimization algorithm

We now provide a heuristic algorithm that searches the optimal mapping of a job to the system resources. First, at the start of the optimization, the clustering of occupied resources is enabled. Second, the resources are scanned in a priority-ordered way, starting with the most utilized resource.

Clustering is applied to simplify the modeling of reserved resources. At the specific resource (processing tiles, connections), the reserved resources are clustered together and modeled as one big task with an certain total resource consumption.

The *reordering* of resource types is evaluated. The priority order can vary over time. Our assumption is that, if there is a conflict between available and required resources, it will be on the mostly utilized resources.

The algorithm starts on a request of a job i for reconfiguration of the actual allocation. The start or termination of a job is a special case of the job reconfiguration. Beginning with the highest quality level, the algorithm checks the availability to map the task with the specified resource requirements to the platform resources (for each type J). In the case of a possible mapping, it checks the availability of the next type of resources. The algorithm stops if the platform has all resource types available for *a job mapping*. Otherwise, it decreases the quality of a job and tries to find a mapping on the conflicting type of resources. If it would not succeed even for the lowest quality of the job, the job *cannot* be mapped.

4.3 Experimental results

We have developed a simulation framework that implements the above optimization presented in Section 4.1. First results are now becoming available, showing that with this optimization a higher overall quality is obtained when compared to the conventional approach. Moreover, the reaction time of the system to change the job conditions is faster than in the original case. e have found that with more than about six jobs, the computation of the optimal operation point becomes too complex for run-time allocation.

We have implemented the heuristic optimization algorithm, which was published recently [6]. The current work concentrates on negotiating the qualities among jobs, depending on the benefit of a job execution at certain level. The modeling of resources and the resource conflicts were based on the N-dimensional models that are conceptually presented in this paper. The experimental setup has considered three resource types: computation, data storage, and communication. At the time of experiments, not all data was available for the five resource types indicated. Nevertheless, the preliminary experiments have shown the benefits of the modeling technique for run-time QoS system management.

5 Conclusions

We have presented a QoS management system for multiprocessor platforms designed for executing multiple multimedia applications. The QoS control system is based on modeling of the estimated resource requirements. The modeling of different types of resources, involving both required and available resources, allow more efficient system design and usage of a system. A novel system aspect is to include the most influencing input-data parameter within the estimation model, to more quickly and flexibly react on varying system conditions (e.g. caused by varying object sizes in MPEG-4). Another important system aspect is the search of the maximum quality over a limited time interval. This balances the system adaptivity with a simplified QoS management by decreasing the negative effect of too frequent system reconfigurations. The hierarchical distribution of responsibilities between system control and intra-application control (two-layer management) speeds up the design time for a new system by reusing the estimation technique for all required resources. The proposed clustering technique simplifies the N:1 mapping problem of a job allocation to a 1:1 mapping problem. We are currently gathering more system data in order to execute multiple multimedia applications simultaneously, while monitoring the usage of a plurality of resource types.

References

[1] M. Pastrnak, P. Poplavko, P.H.N. de With, J. van Meerbergen, "Hierarchical QoS concept for multiprocessor system-on-chip", Proc. Workshop On Resource Management for Media Processing in Networked Embedded Systems, pp. 139-142, March 2005.

[2] D.E. Culler, J.P. Singh, and A. Gupta, "Parallel computer architecture: a hardware/software approach", Morgan Kaufmann Publishers, Inc., 1999.

[3] Z. Deng and J.W.-S. Liu, "Scheduling real-time applications in an open environment", Proc. IEEE Real-Time Systems Symposium, pp. 308-319, December 1997.

[4] J. Regehr and J. Stankovic, "HLS: A framework for compsing soft real-time schedulers", In Proc. IEEE Real-Time Systems Symposium, pp. 3-14, December 2001.

[5] M. Pastrnak, P. Poplavko, P.H. N. de With, J. van Meerbergen, "Novel QoS model for mapping of MPEG-4 coding onto MP-NoC", Proc. 9th IEEE Int. Symp. on Consumer Electronics (ISCE), pp. 93-98, June 2005.

[6] M. Pastrnak, P.H. N. de With, J. van Meerbergen, "Realization of QoS management using negotiation algorithms for multiprocessor NoC", Proc. 38th IEEE Int. Symp. on Circuits and Systems (ISCAS 06), Greece, May 2006.