Modeling Predictable Multiprocessor Performance for Video Decoding

Peter Poplavko\textsuperscript{1,3} and Milan Pastrnak\textsuperscript{1,2} *
\textsuperscript{1}Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
\textsuperscript{2}LogicaCMG Eindhoven / \textsuperscript{3}Philips Research Labs Eindhoven
\{P.Poplavko, M.Pastrnak\}@tue.nl

Abstract — This work addresses implementation of decoding an MPEG4 bitstream with multiple arbitrarily shaped video objects (AS-VOs). Such multimedia applications pose challenging requirements on embedded systems design with respect to compositionality, scalability, and predictability in order to meet real-time constraints. Multiprocessors based on networks-on-chip (MP-NoC) are appropriate platforms that satisfy these requirements [3]. The workload of the AS-VO-decoding changes dynamically at run-time. To control the system resources, it is favorable to have workload estimation of one video frame (or VOP), before the frame (or VOP) is decoded. To make this task easier, the encoder puts a few complexity parameters in the VOP header. For single-processor implementation, linear complexity functions on can be used to obtain the workload [6, 2]. Preliminary results show that with the full a-priori knowledge of the input bitstream, the model is accurate within 6\% in average [6]. For multiprocessor implementation, we extend these models to parametrical IPC graphs [7]. In this case, the same accuracy is possible, but hardly feasible at run-time due to coding efficiency reasons. In [7], a feasible approach has been proposed, which yields a safe upper bound on the workload, but on the price of high error. We discuss the current status in our modeling approach.

Keywords — network-on-chip; system-on-chip; timing model; performance evaluation; resource estimation; real-time; data-flow graph

I. INTRODUCTION

In order to run real-time multimedia applications on an on-chip multiprocessor system, techniques have to be developed to control the resource usage by those applications on the multiprocessor platform.

To tackle the interactivity and dynamism of applications, a real-time environment is required to coordinate multiple jobs on the set of processors. Informally, a job is an activity started and stopped by some unpredictable run-time events, which can come from user actions (e.g., pushing a button) or from changes in the video scene. We currently assume that that each job is assigned to decode one video object. We also assume that each running job is invoked regularly. At each invocation, it has to produce a certain number of output data samples (macroblocks or MBs) that constitute together one frame (video object plane, or VOP). A real-time control hierarchy must ensure that the complete application (the set of active jobs) provides output with the best quality that can be achieved while meeting the timing constraints [9].

Complex multimedia applications pose challenging requirements on embedded systems design with respect to compositionality and scalability. Moreover, the authors believe that, in order to meet the real-time constraints at low cost, predictable hardware behavior is necessary. We study a design of multiprocessor network-on-chip (MP-NoC), which intrinsically supports these requirements [1, 3].

The multiprocessor workload of each job and its communication bandwidth requirements may change over time, e.g., when the dimensions, shape and texture of the correspondent video object changes. The real-time control hierarchy should track these changes in a timely manner and adapt to them at run-time, e.g., by allocating more processors to the job or by reducing the amount of computations at the expense of visual quality [9].

In this paper, we present an overview on the use of parametrical inter-process communication (IPC) graphs [7, 8] as models allowing estimation of the workload of the multiprocessor jobs. We develop IPC graph models for the decoding of arbitrarily shaped objects, as defined in the MPEG4 standard.

This paper is organized as follows. Section II gives background information about the MP-NoCs. In Section III, we briefly present current IPC graph model together with the results on accuracy. Section IV concludes this paper and mentions future work.

*Supported by the European Union via the Marie Curie Fellowship program under the project number HPMI-CT-2001-00150.
II. MULTIPROCESSOR NETWORK-ON-CHIP

On-chip multi-processor architectures are important for the implementation of MPEG-4 (multimedia) applications [2].

The growing design complexity results in the need in the platforms featuring compositionality, design reuse and design scalability. The MPEG-4 standard supports these requirements from the application side. The network-on-chip design paradigm supports them from the hardware side [3].

Meeting timing constraints requires predictable timing in communication. On-chip networks can support predictable timing by offering a guaranteed throughput service [3].

To manage complexity and support predictability, we use a hierarchical approach in the system architecture design. Currently, we assume the simplest variant of this approach, in the form of a so-called tile-based architecture as depicted in Figure 1.

This type of architecture assumes two levels of hierarchy. The lower level shows the details of processing tiles, and the higher level consists of a set of tiles connected by the on-chip interconnection network and a level-2 (off-chip) memory. A processing tile represents a small self-contained embedded computer, consisting of one or two embedded CPU cores (e.g., RISC), local level-1 memory (denoted as L1 in Figure 1) and application-specific accelerators. A communication assist serves in each tile as a gateway from the local, tile-specific memory system to the standard network services.

The tile-based hierarchical approach can be seen as a natural extension of the existing video decoders that handle with 1 or 2 processors only one principal MPEG-4 video object containing the complete picture of

the video stream [2]. To implement, e.g., the MPEG-4 Core profile, multiple smaller video objects can be handled by running multiple instances of a single-object-decoding job on the replicated processing tiles.

III. WORKLOAD ESTIMATION

In the design of a real-time control hierarchy it would be favorable to have a method to estimate in advance how many processor cycles each job consumes from its processors (job workload).

We propose to construct at design time a parametrical timing model and then use it at run-time for the workload estimation. We require the encoder to put the complexity parameters into the image headers.

A. Design-Time Model Construction

To express the parallelism and the communication of multiple processes executing the main loop of the job, we use an inter-process communication graph, which is an instance of a homogeneous dataflow graph model of computation [5]. As an example, we show in Figure 2 an IPC graph for intra-frame shape-texture decoding. The graph is constructed in two major steps, presented below.

Step 1. Resource Estimation [2, 4, 6]

For each main computation actor (software routine like CAD, CBP, VLD...) the complexity function is defined for the given data granularity (e.g., single macroblock (MB)). We currently assume that all processors are ARM7TDMI cores with flat local memory with single-cycle access.

For example, the complexity function of inverse quantization actor is estimated by:

$$t_{IQ} = 1.39K \varphi + 44N_{AC}$$  \(1\)

![Figure 1: Architecture with tile-based MP-NoC.](image1)

![Figure 2: IPC graph of MPEG-4 shape-texture decoder of an intra-frame of a video object (or I-VOP) [7](image2)
where $\varphi$ and $N_{AC}$ are complexity parameters. In particular, $\varphi$ is the total number of non-transparent sub-blocks and $N_{AC}$ is the total number of non-zero AC coefficients. We have observed that for the ARM7 processor, the complexity functions are accurate within 6% in average, given that all parameters are known exactly [6].

Note, that if all actors are mapped to a single processor, the total workload is computed by simply adding the complexity functions together. We obtain again the accuracy in the order of 6%. For multiprocessor job, it is not so straightforward to compute the workload.

**Step 2. IPC Graph Construction** [7, 8]

This step is performed after the actors have been assigned to processors, and inter-processor data communication has been assigned to network connections. For each processor $\text{Proc } i$, we introduce a process cycle into the IPC graph. For each connection $C_p$, we introduce a subgraph that models the connection. In Figure 2, we see an IPC graph containing three process cycles ($\text{Proc1}$, $\text{Proc2}$ and $\text{Proc3}$) and three connection subgraphs ($C_0$, $C_{DCT}$ and $C_{YUV}$). See [7] for more details.

**B. Run-Time Workload Estimation**

In [7] we propose a way to use IPC graph to obtain workload estimation for the decoding of one image of a video object, called video object plane (VOP). One macroblock (MB) passes all stages of processing in every iteration of IPC graph. If complexity parameters of all MBs in VOP are known, the delays of all actors at each iteration are also known. In this case, a longest-path computation in IPC graph unfolded multiple times would yield the job workload with the same accuracy as the complexity functions (around 6%). However, it is not feasible to enforce the encoder to put all complexity values for all MBs into the header.

To tackle this problem, we propose in [7] a scenario approach. All MBs of the VOP are divided into several scenarios. For the shape-texture decoding we have identified three scenarios: “transparent MB”, “boundary MB” and “opaque MB”). For each scenario, we require the encoder to put in the header one set of complexity parameters which characterizes all MBs that belong to that scenario (characterization set). In addition, we need two extra parameters, namely, $J_s$ or the total number of MBs in scenario $s$ and $L_s$, or the number of transitions to scenario $s$ from other scenarios.

We estimate the job workload on all processors at runtime as follows [7]:

$$\text{workload} = \sum_s \lambda_s \cdot J_s + \sigma_s \cdot L_s,$$

where $\lambda_s$ and $\sigma_s$ (throughput and lateness) are certain properties of the IPC graph in scenario $s$, which can be computed by applying fast graph analysis algorithms.

The desirable properties of workload estimation are safety and tightness. The safety means that there is enough confidence that the real workload shall not exceed the estimated value. The tightness requirement means that the estimated value is not too pessimistic.

For the safety reason, we proposed in [7] to characterize each scenario with the maximum values of each parameter over all MBs belonging to the scenario. However, for the safety, we had to pay with the tightness. In [7], we have evaluated (2) for an I-VOP of a test bitstream and have observed 55% overestimation of the real workload.

In future work, to improve this result, we will try to exploit the ‘smoothing’ effect that large FIFO buffers and larger data granularities have on the workload variations and to use a characterization set which is in between the maximum and average set of parameters.

**IV. CONCLUSIONS AND FUTURE WORK**

This paper is an overview of our work on modeling the performance of dynamic video-decoding applications. We assumed multiprocessor networks on chip as a target platform, because they meet several important requirements of multimedia application design. The proposed models capture both computation and communication within a dynamic video-decoding job. The models can be used for the run-time workload tracking in the real-time control hierarchy of the platform. Currently we can obtain a safe, but not tight (55% or more) overestimation of the workload.

In the future work, we will develop more elaborate models and estimation methods for the chosen application driver (shape-texture decoder). We will also investigate the possibilities for a real-time control hierarchy, like the one proposed in [9].

**ACKNOWLEDGEMENT**

The authors are grateful to Marco Bekooij, Philips Research Labs Eindhoven, for some valuable feedback concerning the modeling approach.

**REFERENCES**


