A General Stochastic Framework for Low-Cost Design of Multimedia SoCs

Balaji Raman[1], *Ayoub Nouri*[1], *Deepak Gangadharan*[2], *Marius Bozga*[1], *Ananda Basu*[1], *Mayur Maheshwari*[1], *Jerome Milan*[3], *Axel Legay*[4], *Saddek Bensalem*[1], *and Samarjit Chakraborty*[5]

Verimag Research Report n^o TR-2012-7

June 28, 2012

Reports are downloadable at the following address
http://www-verimag.imag.fr

Unité Mixte de Recherche 5104 CNRS - Grenoble INP - UJF
Centre Equation
2, avenue de VIGNATE
F-38610 GIERES
tel : +33 456 52 03 40
fax : +33 456 52 03 50
http://www-verimag.imag.fr

A General Stochastic Framework for Low-Cost Design of Multimedia SoCs

Balaji Raman[1], *Ayoub Nouri*[1], *Deepak Gangadharan*[2], *Marius Bozga*[1], *Ananda Basu*[1], *Mayur Maheshwari*[1], *Jerome Milan*[3], *Axel Legay*[4], *Saddek Bensalem*[1], *and Samarjit Chakraborty*[5]

June 28, 2012

Abstract

Reliability and flexibility are among the key required features of a framework used to model a system. Existing approaches to design resource-constrained, soft-real time systems either provide guarantees for output quality or account for loss in the system, but not both. We propose two independent solutions where each modeling technique has both the above mentioned characteristics. We present a probabilistic analytical framework and a statistical model checking approach to design system-on-chips for low-cost multimedia systems. We apply the modeling techniques to size the output buffer in a video decoder. The results shows that, for our stochastic design metric, the analytical framework upper bounds (and relatively accurate) compare to the statistical model checking technique. Also, we observed significant reduction in resource usage (such as output buffer size) with tolerable loss in output quality.

Keywords: Stochastic Real-time Calculus, Multimedia Systems, Statistical Model Checking, BIP

Reviewers: Marius Bozga

Notes: [1] VERIMAG, France, [2] Technical Univeristy of Denmark, [3] Ecole Polytechnique, France, [4] INRIA Rennes, France, [5] Technical University of Munich, Germany.

How to cite this report:

```
@techreport {TR-2012-7,
   title = {A General Stochastic Framework
for Low-Cost Design of Multimedia SoCs},
   author = {Balaji Raman1, Ayoub Nouri1, Deepak Gangadharan2, Marius Bozga1,
Ananda Basu1, Mayur Maheshwari1, Jerome Milan3, Axel Legay4, Saddek Bensalem1,
and Samarjit Chakraborty5},
   institution = {{Verimag} Research Report},
   number = {TR-2012-7},
   year = {}
}
```

1 Introduction

An apt choice of a modeling framework is essential to design resource-constrained System-on-Chips (SoCs) in multimedia systems (such as video/audio players, etc.). Such a modeling framework must exploit the inherent stochastic nature of the multimedia applications to design low-cost systems. The uncertainty in such systems is due to high variability present in the input multimedia stream, in terms of number and complexity of items that arrive per unit time to the system. Consequently, the variability is exhibited both in arrival and in processing time.

Many of the existing *analytical* frameworks proposed so far are either incompatible or inflexible to capture the key characteristics of the system being modeled:

- *Worst-case execution time* modeling and analysis framework [21] cannot capture behavior of soft real-time systems, leading to pessimistic designs with exorbitant use of hardware resources (such as buffer size).
- *Average-case execution time* analysis framework [39] cannot provide QoS guarantees and are thus hardly trustworthy.

To address the above limitations, we sought a framework for analyzing multimedia systems that account for the stochastic nature of the streaming application. We need a model characterizing input stream and execution of the multimedia stream as stochastic; instead of capturing event arrivals and executions with worst or average cases.

Recently, stochastic network calculus[1] [13] based approaches have been proposed for performance analysis of multimedia systems [24, 29]. These approaches, however, used the probabilistic calculus only partially: the input stream objects of a multimedia stream (e.g., frames) and their execution time are assumed to be deterministic. Another study used probabilistic real-time calculus to analyze hard real-time systems [30]. This later research, however, did not focus on any specific application domain. Nonetheless, if stochastic network calculus is fully adopted for system-level design, then design of multimedia embedded platforms can benefit, too. The analysis can provide probabilistic bounds on the output quality of the system. The worst-case and average-case analysis of the system are special-case scenarios of the analytical framework.

The two aforementioned key modeling features — flexibility and reliability — for modeling multimedia systems exist in statistical model checking approaches, too. Statistical model checking consists in simulating the formal representation of the systems, monitors a finite set of traces, and then guess an overall correctness by exploiting algorithms from the statistics area. Recent work showed that component-based frameworks, such as BIP [2], can be coupled with statistical model checking to verify large heterogeneous embedded systems [4]. Unlike many ad-hoc simulation techniques, BIP provides a formal semantics for the modeling and simulation of stochastic systems. Such a formal semantics is a prerequisite for using statistical model checking as a sound performance analysis technique on system models.

In what follows, we list the main **contributions** of our paper:

- *Stochastic characterization:* We use stochastic real-time calculus for performance analysis of multimedia systems. Our model captures the uncertainty of arrivals and executions of stream objects (and dependencies between them) using probability laws.
- *Model comparison:* Using a video decoder case-study, we apply both approaches (a major contribution is to build a BIP model). Our results show that the probabilistic estimates obtained from analytical framework upper bounds the estimates from statistical model checking.
- *Resource constrained design:* Our second goal with the video decoder case-study is to reduce resource consumption with tolerable loss in video quality. This case study illustrates how expensive resources (such as the playout buffer) can be reduced if the stochastic behavior

[1]Stochastic network calculus was originally developed for performance analysis of computer networks

of the application is carefully taken into account during performance. We chose some application parameter values (quickly) using the analytical framework. Then, we estimate the buffer sizes (precisely) for a set of parameter values using statitstcial model checking. We discuss how the strengths of both the approaches can be complementary to each other.

Organization: In Section 2, we present an illustrative example that motivates the design of low-cost multimedia SoCs. We present the analytical and the statistical modeling checking approaches in Sections 3 and 4. In each of these two sections, we first present some background and then the modeling technique. In Section 5, we apply both the approaches—again, independently—to a case-study, video decoder. The results of the analytical approach are compared with the statistical model checking approach (for the case study) in Section 6. The results section reports the buffer size savings, too. The tail end portion of the paper contains the discussion, related work, and the conclusions.

2 Motivation

In this section, using a concrete example, first we show how we reduce the on-chip memory size tolerating loss in output quality. Second, we detail how the two frameworks (analytical and simulation) can be used towards reducing the buffer size. We conclude this section with a problem statement.

2.1 Illustrative Example

Multimedia SoC: The SoC architecture is a processing element with an input buffer and a playout buffer (shown in Figure 3). For this example, the processor architecture is executing a video decoding application. The processing unit decompresses an input stream arriving at the input buffer. Then the processor writes output sequentially to the playout buffer. Finally, the consumer reads items at a constant rate from the playout buffer after an initial delay.

We performed experiments using a system-level simulator (described in Section 4). This simulator models the SoC architecture mapped to the video decoding application shown in Figure 3. Using this detailed simulator, we conducted a series of simulations: first with *actual traces* and then using *synthetic traces*. Consider that for a specific compressed video stream we are given amount of bits constituting each stream object (e.g. macroblock) and amount of processor cycles required to decompress each stream object. We name these traces as actual traces, which we use to construct synthetic traces.

Roughly speaking, the construction of the synthetic traces is an effort towards discarding events that contributes to worst-case scenarios. When the SoC is designed using synthetic traces instead of the actual traces, resource requirements, such as buffer size, significantly reduces. The reduction in buffer size happens as we replace some events in the actual input trace; those events that lead to quick increase in buffer fill level are replaced. Now we explain in detail how we construct the synthetic trace.

First, a lower bound value is chosen for the number of bits constituting a macroblock; second, the number of bits for each macroblock in the actual trace is compared with the lower bound; third, for those macroblocks that fall below this lower bound, the number of bits is set to the lower bound. Similarly, we construct another synthetic trace from the actual trace containing execution cycles of each macroblock[2]. Currently, we use trial-and-error approach to select the lower bound for both the bits and the execution cycles to construct the synthetic trace. But in future we intend to construct synthetic traces in a systematic manner (we discuss this later in the paper). The motivation to construct these synthetic traces is presented below.

We noticed this reduction in buffer size in our simulation experiments. We observed amount of buffer fills in the input buffer and playout buffer as a video is being decompressed. Figure 1 shows

[2]Alternatively, construction of another type of synthetic traces that optimizes on processor speed can be envisioned. For such an optimization, it would require us to ignore events consuming maximum number of execution cycles.

the buffer fill levels for the playout buffer. For the actual trace, there was no buffer underflow, that is, the consumer always found an item when it read the playout buffer. Notice that at the playout buffer, the amount of buffer fill level has substantially reduced for the synthetic trace when compared to the actual trace. This is simply due to the fact that the number of items arriving at the playout buffer is smaller for the synthetic trace compared to the actual trace[3].

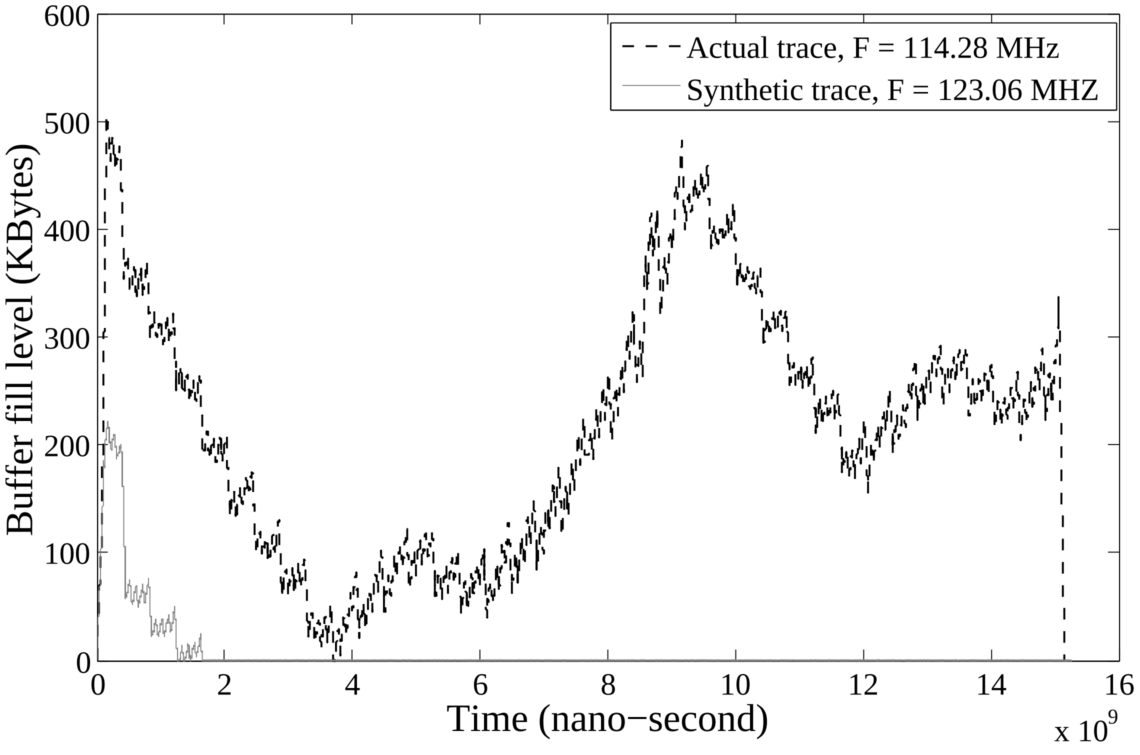

Figure 1: Playout buffer fill level. The maximum buffer fill level for the synthetic trace is significantly lower than the actual trace.

For the experimental results shown in Figure 1, the *initial playout delay* has to be chosen for both the synthetic and the actual trace; initial delay is the delay after which the video starts displaying. Using this initial delay parameter, we can tune the amount of buffer underflow that could occur at the playout buffer [24]. Figure 2 shows underflows when the system is run using the actual trace for various initial delay values.

2.2 Methodology

In the previous sub-section, we illustrated the use of synthetic traces in saving output buffer size. Note that in Figure 1 the intital delay was almost set the same for both the synthetic and the actual trace to emphasize the buffer-size savings. Unlike for the actual trace, in our simulation experiments, we observed buffer underflow for the synthetic trace.

In multimedia literature, however, it has been shown that certain loss is acceptable. For example, a study showed that a consecutive loss of 2 frames in 30 frames is tolerable [36]. This loss in quality may be as follows: the playout device displaying frames periodically per unit time may not have all the macroblocks in a frame ready for display; certain macroblocks have missed

[3]At the input buffer, the fill-level is large for the synthetic trace because the processor cycles required is larger for the synthetic trace compared to the actual trace. This requirement in processor cycles is larger because the lower bound for the synthetic trace is increased. However, for video decoding applications the input buffer holds the compressed macroblocks and so the increase in terms of bits is not substantial. For video encoding, we would be reducing the input buffer size instead of the playout buffer size.

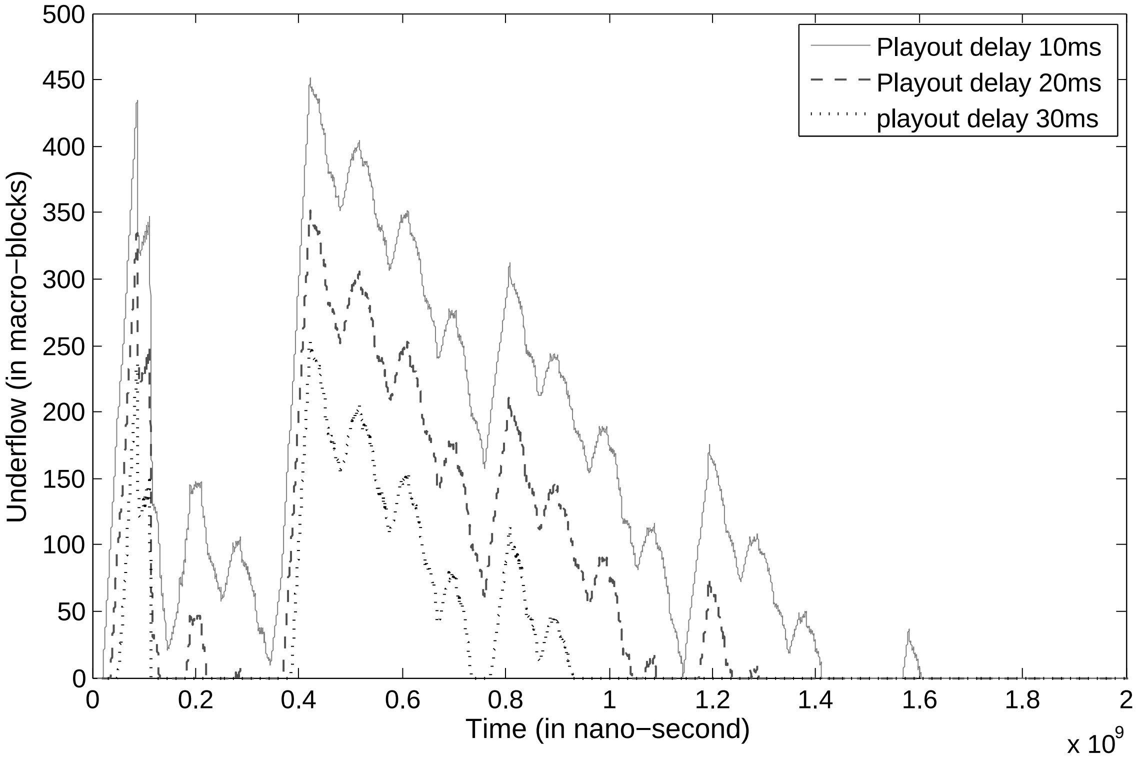

Figure 2: Playout buffer underflow.

their deadlines (or may have been dropped). This leads us to a key question: is the probability of loss of certain amount of macroblocks within an acceptable threshold?

In this paper, we obtain probabilistic bounds on the validity of specific QoS properties relating to loss of stream objects. The analytical model we use characterizes the stochastic nature of arrivals in that the actual number of items arriving at the input buffer may not obey the bounds we constructed using synthetic curves. Similarly, our model captures the stochastic nature of execution time, and also dependencies between the arrival and the execution times. The use of stochastic network calculus results makes it possible to provide probabilistic bounds (e.g. on the output) using probabilistic function on the input and the execution. Thus we are able to estimate probabilistic bounds of properties on the output buffer size, for example, on the amount of buffer underflow.

We experimented with a state-of-the-art simulation approach [4] to validate and compare results obtained from the mathematical framework. This simulation approach has similar design goals to that of our analytical framework; the performance evaluation combined with the statistical model checking provides a probability that a specific QoS property is met. In the simulation set-up, we generate synthetic clips, unlike the analytical approach, using pseudo-random generation such that it closely follows the structure and characteristics of the actual traces. These set of new synthetic clips are stochastically similar to the synthetic clips used in the analytical approach, so, we are able to compare the probabilistic bounds obtained from the simulation and the mathematical framework.

2.3 Problem Statement

In the following sections, we present two approaches, the stochastic framework and the statistical model checking for estimating the probabilistic bounds on QoS properties. Now, we state the problem we attempt to solve in this paper.

For a video clip of given bit-rate and resolution, we have to find the probability that a specific QoS property is satisfied. The SoC is designed using the synthetic traces; the buffer size is chosen based on the requirements for synthetic trace. When the actual traces are run on the SoC, designed using synthetic traces, buffer underflow occurs. We estimate the probability that this buffer underflow is within a certain number of macroblocks.

3 Analytical Model

The main purpose of this study is to provide ways to design low-cost systems, especially towards solutions for reducing the on-chip memory size. In this context, we want to characterize a mulitmedia system in a stochastic setting in this section, that is, the arrival, processing, and the output of multimedia streams. Towards this, first, we give the background of a deterministic analytical model of a system running multimedia applications, and, then, extend the model to a probabilistic setting.

3.1 Background

This subsection presents the real-time calculus framework for the system model we use in our subsequent sections. Note that this subsection is not intended to be an exhaustive exposition of real-time calculus for embedded systems. The system model used in this paper is sketched in Fig. 3. The architecture is composed of memory buffers and processing units. Fig. 3 shows a data stream x reaching the input buffer while the processed stream y is written to the playout buffer.

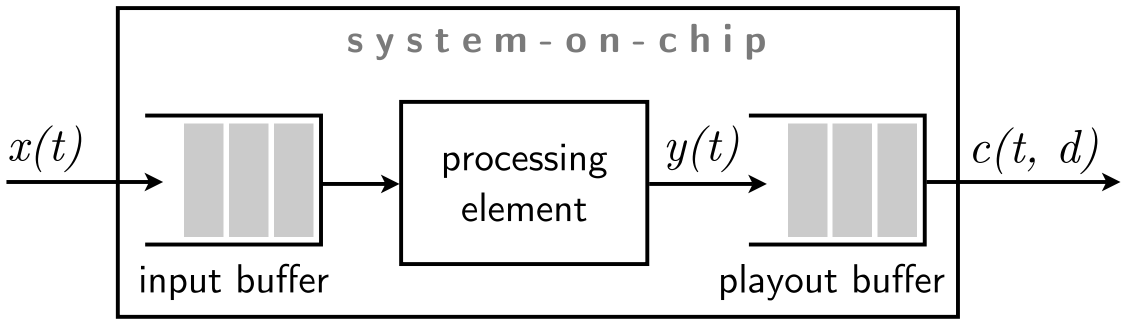

Figure 3: Real-time calculus model

For simplicity of exposition, we chose a basic block or unit of the real-time calculus to model the multimedia SoC. Indeed, analysis using real-time calculus has been shown for a system with multiple streams to the input buffer [9], multiple processing units in a SoC [27], and multiple tasks in a processor [25]. This paper focus, however, is to propose performance analysis using stochastic real-time calculus. Future work can extend this stochastic framework to the above listed settings.

The application mapped to the processing element is a multimedia task. However, there is no limitation on the number of applications running on the processing element. Consequently, there are no restrictions on the number of input streams fed to the processing element. Similarly, the number of architecture units in the system-on-chip can be any number of processing elements and memory units. Indeed, analysis using real-time calculus has been shown for a complex system containing shared memory, bus, and network-on-chip communication architectures. The data flow need not be sequential in that there could be a feedback flow between the architectural components in the system-on-chip, and there could be splits and joins in the dataflow of the architecture.

The notations of the parameters involved in the description of our model are described in the following tables, divided as follows:

- Table 1 lists the application and architecture parameters that are given to the system designer.

- Table 2 records the functions that are obtained from processor simulations.
- The mathematical functions that are constructed from the known parameters from Table 2 are given in Table 3.
- Similarly, mathematical functions constructed from simulation data from Table 1 are presented in Table 4.
- Finally, Table 5 and 6 shows the functions and the models used in the real-time calculus framework.

	Notation and name		Description
Application	λ	Bitrate (in bits per second)	The bit rate of the input video is constant and is fed to the input buffer. This bit rate is assumed to be constant.
	c	Consumption rate (in macroblocks per second)	The output device reads items constantly from the playout buffer.
	d	Initial playout delay (in millisecond)	The delay after which the output device starts playing the video.
Architecture	b	Input buffer size (in macroblocks[†])	The stream is fed to the input buffer. This is a logical buffer and physically it might be a part of memory.
	B	Playout buffer size (in macroblocks or bytes)	The processed stream is written to the playout buffer after which the consumer reads from the buffer.
	f	Processor frequency (in MHz)	The effective number of processor cycles available per unit time. The available processor cycles for the multimedia task is constant.

[†] *The input buffer can hold a given number of macroblocks and it is not given by a size in bytes because the application case study we use in this paper is video decoding. The compressed stream arriving at the input buffer consists of macroblocks of variable size whereas the playout buffer holds decompressed items of constant size.*

Table 1: Known applications and architecture parameters.

Notation	Name	Description
bits(k)[†]	Cumulative bits	Number of bits per k consecutive stream objects.
cycles(k)[*]	Cumulative cycles	Number of cycles consumed by k consecutive stream objects.

[†] *The number of bits for each and every macroblock is computed with the execution of the application once and the data flow between the software blocks.*

[*] *The simulation performed to compute the cycles is not a system-level simulation. The processing element is independently simulated using a software simulation to compute the processor cycles consumed for each and every macroblock.*

Table 2: Parameters from simulation of processor and from application source code.

The core part of the deterministic real-time calculus can be reduced to three inequalities. First, bounds on the arrival of the data stream are computed from the ϕ_l, ϕ_u functions, which gives the

Notation	Name	Description
$\phi_l(k), \phi_u(k)$	ϕ functions	Minimum and maximum number of bits constituting any k consecutive stream objects.
$\gamma_l(k), \gamma_u(k)$	γ functions	Minimum and maximum number of cycles needed to process any k consecutive stream objects.

Table 3: Mathematical functions from simulation data.

Notation	Name	Description
β-cycles(k)	Processor cycles over $[0, t]$	Product of time and frequency, β-cycles$(k) = tf$.
$c(t)$	Consumption function	Minimum and maximum number of cycles needed to process any k consecutive stream objects.

Table 4: Mathematical functions from input parameters.

Notation	Name	Description
$x(t)$	Input function	Number of items arriving in the input buffer over the time interval $[0, t]$.
$y(t)$	Output function	Number of items arriving in the output buffer over the time interval $[0, t]$.

Table 5: Mathematical functions and inputs of arrival, service and output.

Notation	Name	Description
$\alpha_l(\Delta), \alpha_u(\Delta)$	Arrival curves	Minimum and maximum number of items arriving over the time interval Δ. For example Δ is defined as $[0, t]$.
$\beta_l(\Delta), \beta_u(\Delta)$	Service curves	Minimum and maximum number of items guaranteed to be processed over the time interval Δ. For example Δ is defined as $[0, t]$.

Table 6: Mathematical model.

minimum and maximum number of bits constituting any k consecutive stream objects respectively.

$$\alpha_l(\Delta) = \phi_u^{-1}(r\Delta), \tag{1}$$
$$\alpha_u(\Delta) = \phi_l^{-1}(r\Delta), \tag{2}$$

where r is the bit-rate of the input video, and where $\alpha_l(\Delta), \alpha_u(\Delta)$ are the minimum and maximum number of items arriving over the time interval Δ respectively. The mathematical functions $\phi_{u,l}$ are computed from the parameter *bits*(k) (obtained from simulating the application code), which gives the number of bits for first k stream objects, as shown below

$$\phi_u(k) = \max\{bits(k+N) - bits(N)\} \tag{3}$$

$$\phi_l(k) = \min\{bits(k+N) - bits(N)\} \tag{4}$$

where N can take any value such that $k + N$ does not exceed the stream length.

The number of items arriving at the input buffer over the time interval $[0, t]$ ($x(t)$ is bounded by the previously introduced $\alpha_{u,l}$ functions:

$$\alpha_l(\Delta) \leq x(t+\Delta) - x(t) \leq \alpha_u(\Delta), \tag{5}$$

for $t, \Delta \geq 0$.

The second core inequality is based on the service curve (β), which guarantees the number of processor cycles dedicated to that particular multimedia task over the time interval Δ. Given the processor frequency (f), we first compute the processor cycles available over time interval $[0, t]$, that is, β-cycles$(t) = tf$. Second, given the number of cycles consumed by first k stream objects $(cycles(k))$, we compute the minimum number of cycles needed to process any k consecutive stream objects (γ_l) in a similar manner as shown in Eq. 4. Third, we compute *beta* as follows: $\beta_u(\Delta) = \gamma_l^{-1}(\beta\text{-cycles}(\Delta))$. Now, we present our second core inequality. Let $y(t)$ be the number of items arriving in the output buffer over the time interval $[0, t]$. Then, it can be shown that:

$$y(t) \leq (\alpha_u \otimes \beta)(t), \tag{6}$$

where $\otimes$ is the min-plus convolution operator defined as:

$$(p \otimes q)(t) = \inf_{0 \leq s \leq t} \{p(t-s) + q(s)\}. \tag{7}$$

Finally, if the playout buffer never underflows, we have the last core inequality:

$$y(t) \geq c(t, d), \quad \forall t \geq 0, \tag{8}$$

where $c(t, d)$ is the items consumed after the initial playout delay (d) by the display device over the time interval $[0, t]$.

3.2 Stochastic Real-time Calculus Model

In this sub-section, we propose a probabilistic framework for designing multimedia SoCs.

From the inverse ϕ function, we compute $\alpha_l(\Delta)$ and $\alpha_u(\Delta)$. This leads to the definition of the stochastic arrival curve. Since by definition, $x(t)$ gives the number of items arriving over the time interval $[0, t]$, we obtain:

$$P\left(\underset{0 \leq \Delta \leq t}{sup} \left(x(t+\Delta) - x(t) - \alpha_u(\Delta)\right) > a\right) \leq f(a), \tag{9}$$

for all $0 \leq \Delta \leq t$ and for all $a \geq 0$.

In the above description, the arrival curve bounds are checked with the actual number of items arriving at the input buffer over any time interval. The decreasing function $f(a)$ is an upper bound on the probability.

We define the stochastic service curve as follows. As formulated in the previous subsection, the output function from the processing element is guaranteed to be smaller than the min-plus convolution of the arrival and service curves:

$$y(t) \leq (\alpha_u \otimes \beta_u)(t), \quad \forall t \geq 0. \tag{10}$$

For a stochastic service, the above inequality is restated as follows:

$$P\left(y(t) - (\alpha_u \otimes \beta_u)(t) > a\right) \leq g(a), \quad \forall t \geq 0, \tag{11}$$

where the stochastic bounding function g, ideally obtained from psychovisual models, is related to the acceptable loss of playback quality. Note that in defining a stochastic service curve we are also including the definition of the arrival curve. Thus dependencies are handled for both the arrival and the service curve.

The output from the processing element is bounded by the arrival functions α_l and α_u. The results from the stochastic network calculus provides bounds on the output curve [13]. If there is a stochastic arrival curve as defined in Eq. 9 and a stochastic service curve as defined in Eq. 11, the output curve is defined as follows:

$$P\left(y(t) - (\alpha_u \oslash \beta_u)(t) > a\right) \leq (f \otimes g)(a), \quad \forall t \geq 0. \tag{12}$$

Write $h(a) = (f \otimes g)(a)$, and the min-plus deconvolution operator is defined as follows:

$$(p \oslash q)(t) = \sup_{u \geq 0} \{p(t+u) - q(u)\}, \tag{13}$$

4 Statistical Model Checking Approach

In this section, we present a second approach for performance evaluation of multimedia SoCs that is based on statistical tests.

4.1 Background

Statistical Model Checking (SMC) has been proposed as an alternative to classical Model Checking techniques [6]. It aims to avoid exhaustive state space exploration. The idea is to do verification on a sub-part of the state space (a sample) and then, using statistics, extrapolate the result to the whole system with some confidence.

Concretely, given a stochastic system S and a property ϕ, *statistical model checking* refers to a series of simulation-based techniques that can be used to answer two questions : (1) *qualitative :* is the probability for S to satisfy ϕ greater or equal to a certain threshold θ ? and (2) *quantitative :* what is the probability for S to satisfy ϕ ?

The main approaches [38, 32] proposed to answer the qualitative question are based on *hypothesis testing*. Let p be the probability that $S \models \phi$. To determine whether $p \geq \theta$, we can test $H : p \geq \theta$ against $K : p < \theta$.

A statistical-based solution (based on a sample) does not guarantee a correct result but it is possible to bound the probability of making an error. The *strength* of a test is determined by two parameters, α and β, such that the probability of accepting K (respectively, H) when H (respectively, K) holds is less or equal to α (respectively, β).

Since it is impossible to ensure a low probability for both types of errors simultaneously, a solution is to use an *indifference region* $[p_1, p_0]$ (with θ in $[p_1, p_0]$) and to test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$.

Several hypothesis testing algorithms exist in the literature. Younes [38] proposed a logarithmic based algorithm that given p_0, p_1, α, and β implements the *Sequential Ratio Testing Procedure* (SPRT) (see [34] for details).

In [8, 17] Peyronnet et al. propose an estimation procedure (PESTIMATION) to compute the probability p for S to satisfy ϕ.

BIP – *Behavior, Interaction, Priority* – [3] is a component based framework encompassing rigorous model based design. It allows building hierarchically structured systems (or composite components) from atomic components characterized by their behavior and their interface.

Components are composed by layered application of interactions and priorities. Interactions express synchronization constraints between actions of the composed components while priorities are used to filter amongst possible interactions and to steer system evolution e.g. to express scheduling policies.

In BIP, *atomic components* are finite-state automata extended with variables and ports. Variables are used to store local data. Ports are action names, and may be associated with variables. They are used for interaction with other components. States denote control locations at which the components await for interaction.

A transition is a step, labeled by a port, from a control location to another. It has associated a guard and an action, that are respectively a Boolean condition and a computation defined on local variables.

In BIP, data and their transformations are written in C/C++. *Composite components* are defined by assembling atomic or composite using *connectors*. Connectors relate ports from different sub-components and represent sets of *interactions*, that are, non-empty sets of ports that have to be jointly executed. For every such interaction, the connector provides the guard and the data transfer, that are, respectively, an enabling condition and an exchange of data across the ports involved in the interaction. Finally, *priorities* provide a mean to coordinate the execution of interactions within a BIP system. They are used to specify scheduling or similar arbitration policies between simultaneously enabled interactions.

To apply Statistical Model Checking on BIP models, it has been augmented by a stochastic semantics [5]. The new semantics allows definition of stochastic atomic components containing probabilistic variables updated through C-defined probabilistic distributions. The stochastic components communicates through interactions that are uniformly selected.

4.2 Stochastic Modeling and Statistical Analysis

The Statistical Model Checking Approach is implemented in the SBIP tool [5]. The latter takes as input (1) a stochastic BIP system model, (2) a probabilistic bounded LTL property [8] or, alternatively, an observer component encoding the evaluation of the property, and (3) a series of confidence parameters needed for the statistical test [38]. Then, it proceeds according to the following steps:

- *Step* 1*:* an executable model is automatically created,
- *Step* 2*:* random execution traces of the system are iteratively generated (sampling),
- *Step* 3*:* the observer component checks the property on each trace,
- *Step* 4*:* then gives a partial verdict for each,
- *Step* 5*:* steps 2, 3, and 4 are repeated until the SMC engine is able to conclude over the whole system. The SMC engine implements the statistical algorithms introduced in the previous section.

Our tool is guaranteed to terminate its execution, when it has decided an answer for the input property to be verified on the input model with respect to the input confidence parameters. This guarantee relies on the mathematical theory of the statistical model checking.

5 Case Study: Video Decoder

We analyze an abstraction of a video decoder SoC (shown in Figure 4) with the analytical and the statistical model checking approaches (presented in previous sections). In the abstract SoC model, the input video stored is fed to the input buffer in terms of stream objects such as macroblocks, frames, etc. A pipeline of functional units process the input stream. Processed items are temporarily stored in the output buffer before their display.

Now, we will state the problem addressed in this paper. We assume that the multimedia SoC contains a single processing unit and two buffers.

Problem Statement: To estimate the probability that the buffer underflow ($U(t)$) is less that two consecutive frames in 30 frames. We are given the following:

- a set of video clips of certain bit-rate (r) and resolution,
- maximum frequency of the processing unit of the multimedia SoC (f),
- consumption rate of the output device (c),
- start-up values for the initial delay (d), input buffer size (b), and playout buffer size (B).

In this section, first we apply the stochastic real-time calculus model, and then the BIP and statistical model checking approach to evaluate the QoS constraints of the video decoder SoC.

5.1 Analytical Model of the Multimedia SoC

In this subsection, first, we give an overview of our analytical approach for this case study. Second, we present formulation for evaluating the QoS constraints.

We can estimate the maximum size of the output buffer using our stochastic analytical framework. Previous work that estimate the output buffer size using deterministic real-time calculus (presented in Section 3.1) proceed as follows [21]. For all given video clips, Maxiaguine et al.

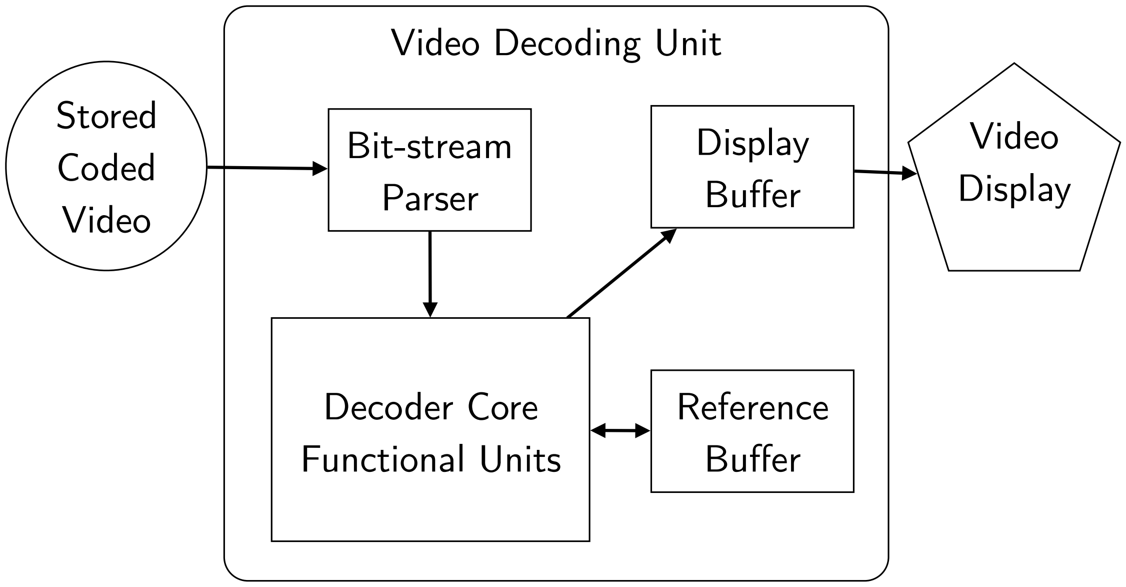

Figure 4: Display buffer in a generic video decoder of a SoC. The functional units of the decoder can include variable length decoding, motion compensation, etc.

construct upper bounds on the number of items that arrive to the input buffer and that execute in the processor. These two bounds together yield another upper bound on the number of items that arrive to the display buffer. Thus, given a rate at which items are consumed from the output buffer, Maxiaguine et al. estimate the maximum buffer size required.

We, too, compute deterministic upper bounds on arrival, execution, and output, however, only for a sub-set of given video clips; the remaining video clips can violate the deterministic bounds. For example, the number of items arriving to the output buffer over a time interval, for a certain clip, can be larger than the deterministic output bound. Now, we explain how to quantify this deviation from the deterministic bound in a stochastic setting (as we are using probabilistic network calculus).

Assume that the stream objects arrival and execution are stochastic—an apt characterization of multimedia streams. So, what is the probability that the number of stream objects that arrive to the output buffer over a time interval is larger than the deterministic output bound?

First, we estimate the maximum probability of violating the deterministic bounds for the arrival and execution; then, using these two bounds, we estimate it for the output (related to the definitions presented in Section 3.2). Using this stochastic bound on the output, given the constant consumption rate, we can compute the probabilistic distribution of the buffer size.

The deterministic analysis imposes hard constraints on the arrival, execution, and output of stream objects leading to over estimation of output buffer size. In contrast, our probabilistic analysis relaxes these constraints. Thus, the designer can choose a buffer size and the corresponding video quality. Following this, we explain how the video quality is related to the buffer size.

We specify tolerable loss in video quality as: less than two consecutive frame loss within 30 frames, less than 17 aggregate frame loss within 100 frames, etc [36]. Previous work models the loss of stream objects as buffer underflows, which occurs whenever the display device finds insufficient items to read from the output buffer [29]. Raman et al. show that the amount of buffer underflow can be controlled using an application parameter, namely, the initial playout delay, the delay after which the video starts to display.

We, too, tune the initial delay parameter to restrict the maximum buffer underflow, albeit, in a stochastic setting. We obtain probability values for a QoS property to hold for a range of initial delays. The buffer size corresponding to each such delay can be estimated. Therefore, the system designer can choose an initial delay based on his resource and quality constraints.

In what follows, we formulate the probability distribution of the output buffer size using the stochastic real-time calculus model presented in Subsection 3.2.

In practice, the designer is typically given a set of input clips to design the SoC with given display constraints. So, in our problem setting, the designer can construct an upper and lower bound using synthetic traces (which are obtained from actual traces) and use the actual traces to construct the bounding functions. Now, we explain the construction of these synthetic traces.

Let the given set of input video clips be partitioned into two sets, S_A and S_B, based on the designer's requirement that all clips in S_B must be processed with no loss in video quality. The deterministic upper bound on the arrival and the output, introduced in the previous section, are constructed using clips in S_B. Now, we discuss how to synthetically generate clips in case we do not have a set S_B.

The information we have about the clips in set S_A are the number of bits and number of cycles corresponding to each macroblock. For certain macroblocks, we modify the number of bits and cycles, assuming we are given lower bounds[4] on these parameters. Any number of bits lower than the actual bound in the input traces are replaced with a value of the lower bound. Thus we obtain synthetic traces forming clips in set S_B. Now we explain how we estimate stochastic bounds using the actual clips (i.e. clips from set S_A) and synthetic traces (or if available clips from set S_B).

The upper and lower bounds on the arrival given from the formula in the previous subsection are calculated for the synthetic traces. That is, from the modified inverse ϕ function, we compute $\alpha_l(\Delta)$ and $\alpha_u(\Delta)$. This leads to the definition of the stochastic arrival curve.

Assume the output arrival curve is an arrival process to the playout buffer. The probability distribution at the playout buffer can be computed using the bounding function h.

$$P\left(U(t) > a\right) \leq h(a - (\alpha_* \oslash ct))(0),\ \forall a, t \geq 0. \tag{14}$$

In the above equation α_* is the output arrival function given by $(\alpha_u \oslash \beta_u)(t)$ which is denoted in Eq.12.

In the results sections, we will show how to specify the QoS property by using Eq. 14. To validate and compare probabilistic bounds obtained from the analytical framework, we used a rigorous simulation framework, BIP [3], introduced in Section 4.1.

5.2 Stochastic BIP Model of the Multimedia SoC

We constructed a model of the video decoding unit shown in Figure 4 using BIP framework. This model captures the stochastic behavior of the system in the following way. A stream object's arrival time to the input buffer, and decoding time in the processing unit follow defined probability distributions. The distributions are constructed from a set of video clips[5]. Thus, the stream object's arrival to the output buffer is probabilistic, which can lead to probabilistic buffer underflows. Next, we explain how to estimate the probability of certain amount of buffer underflow.

Figure 5 shows the stochastic BIP model of the SoC running the video decoding application. The functional units of the SoC are modeled as atomic components respectively, Generator, Processor, and Player. These functional components communicate explicitly through buffers, namely, Input Buffer and Playout Buffer, represented in BIP as atomic components as well. The lines represent connectors, namely write-push, pop-read are used to transfer macroblocks objects between a functional component and a buffer component. The tick connector synchronizes all the functional components, and is used to model explicitly the progress of the absolute (global) time. Now we describe the behavior of each of the functional components with more details.

Generator: This component models the generation of a stream of macroblocks. The stream is generated probabilistically and stored in the input buffer. The number of bits (the size) of every

[4]In the discussion section, we present a technique to generate synthetic traces without this lower bound

[5]Note that we construct distributions from those clips we used in the analytical framework that were allowed to violate the deterministic upper bounds.

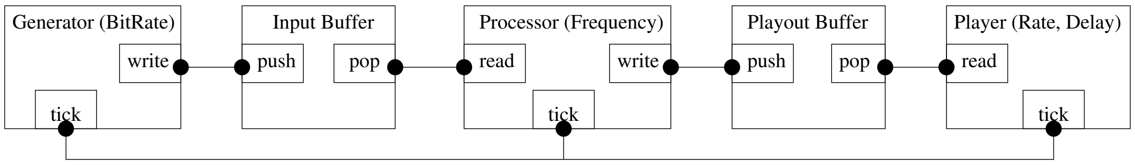

Figure 5: Stochastic BIP model of the SoC running a video decoding application.

macroblock determines the arrival time of the macroblock to the input buffer. This number of bits is specific for each macroblock type (w.r.t frame types) and follows a specific distribution, shown in Figure 6 and based on [14, 15]. Moreover, the choice of frames type follows a Group of Pictures Pattern (GOP): IBBPBBPBBPBB[6].

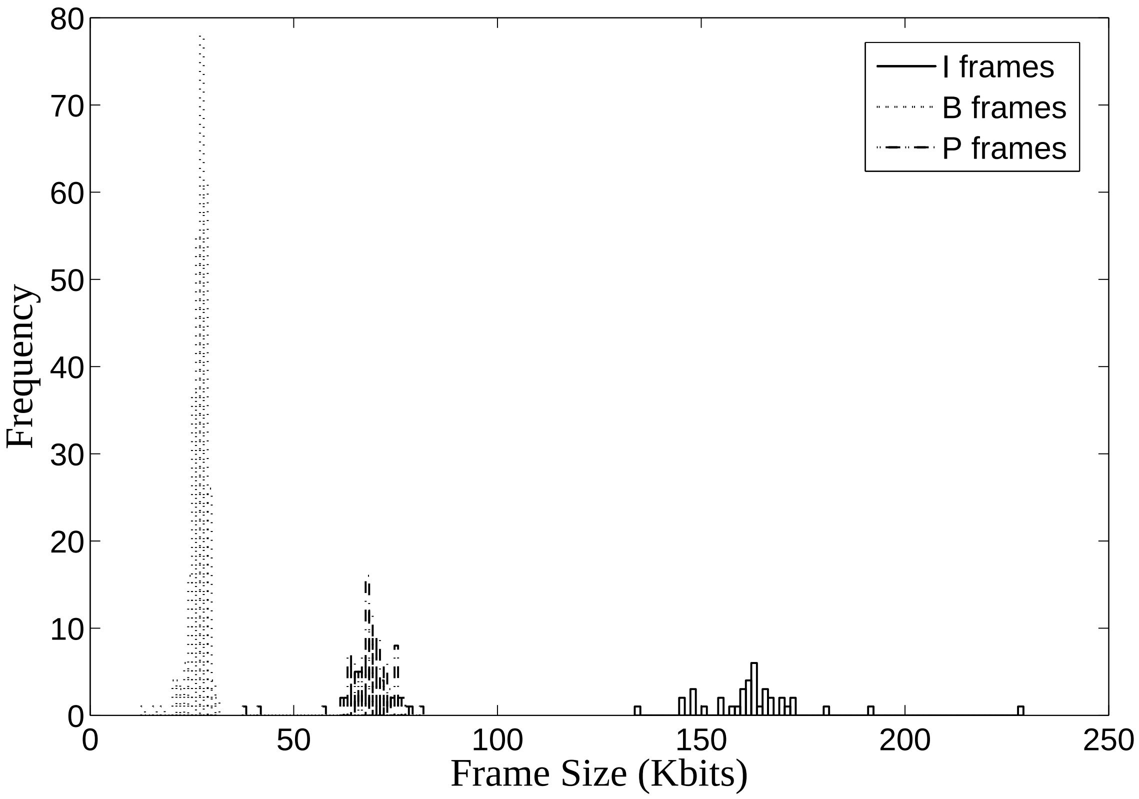

Figure 6: Frequency distributions of I,B, and P frames in the video. I frames are larger in size but smaller in number than the B and P frames.

A second version of this component was designed to generate video-specific stream sequence. Figure 7 shows the accuracy of the stochastic generator against the sequence of the actual movie.

Processor: This component models the decoding of macroblocks, sequentially after reading them from the input buffer. The detailed behavior of the component is shown in Figure 8. The component has two states, IDLE and PROCESS and three ports read, write, and tick. In the IDLE state, the process is either waiting to read from the input buffer or waiting to write to the playout buffer. When there is a macroblock available, the process transits to the PROCESS state and remains there for the time required to process/decode the macroblock.

Player: The Player component models the consumption of the stream of decoded macroblocks. After an initial playout delay, the Player starts reading the macroblocks from the playout buffer

[6]The first GOP is IPBBPBBPBB. It is different from the consecutive GOPs.

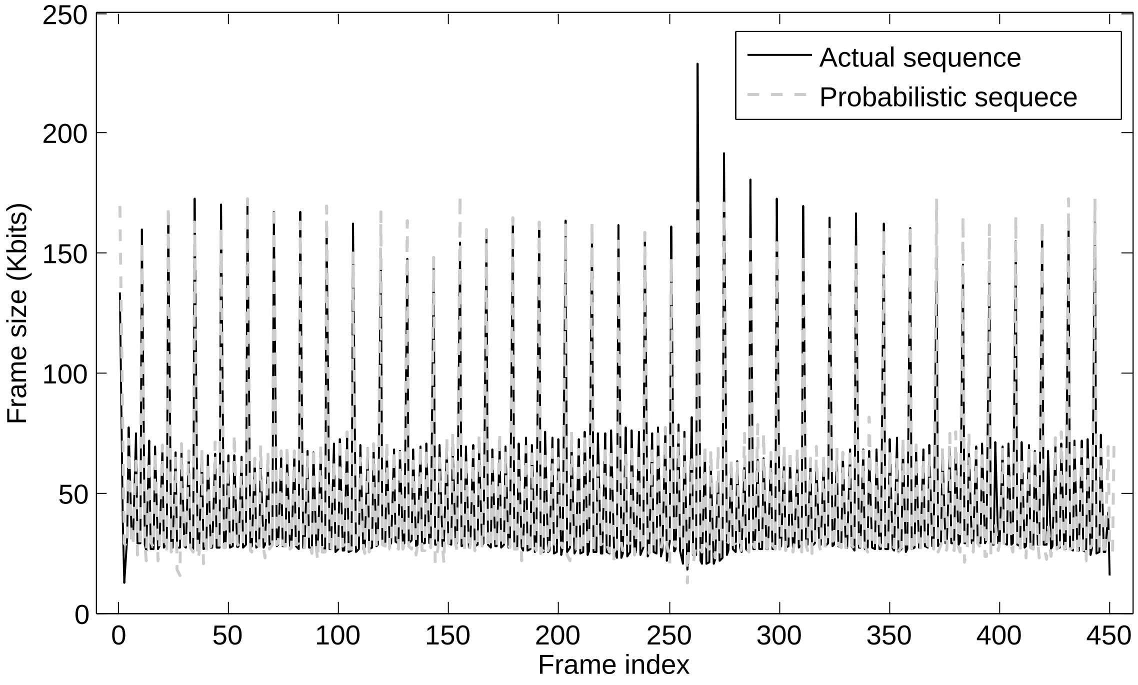

Figure 7: Frame Size : Actual sequence Vs. Probabilistic sequence

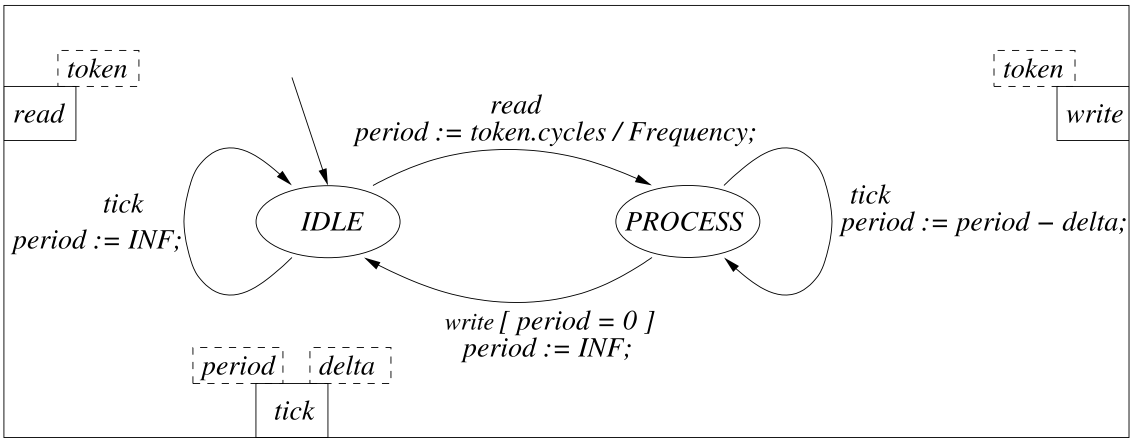

Figure 8: *Processor* model as a BIP component. The processor unit executes a macro-block at a pre-specified speed. The execution cycles corresponding to each macro-block is randomly selected from a distribution or taken from a file in a sequential manner.

at a constant rate. A buffer underflow occurs whenever the requested number of macroblocks is not available in the buffer (Figure 10). In this case, the request is postponed for the next iteration and the underflow is accumulated. For example, if the current buffer underflow is 2, then, at the next request, the Player seeks 3 macroblocks. If the buffer is still empty, the underflow became 3. Else, if the playout buffer has (at least) 3 items, then all three items are read at once and the buffer underflow is reset to 0, etc.

We applied statistical model checking to evaluate QoS properties on the BIP model of the multimedia SoC presented above. As explained earlier, this model is fully stochastic. We focus on a qualitative QoS property related to the playout buffer, that is, the buffer underflow within a second never exceeds two consecutive frames.

In order to evaluate this property on the traces of the model, we use the additional Observer component shown in Figure 9. This component runs in parallel with the systems and reacts to events (interactions) relevant to the satisfaction of the property. The component has three states: OK, PARTIAL, and FAIL. The FAIL state denotes the failure of the property, namely, the underflow of two consecutive frames within a second. If there is a loss of a single frame the observer moves from state OK to PARTIAL. Later, if there is an additional frame loss the Observer reaches the FAIL state. If no loss happens within that second, the component moves back to the OK state.

In the next section, we present the analysis results for both the analytical and the SMC approaches.

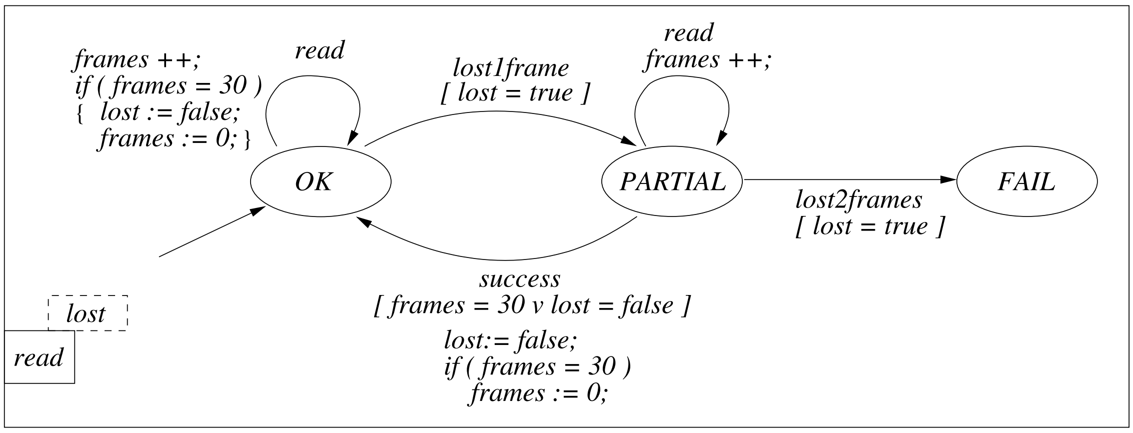

Figure 9: *Observer* model as a BIP component. The observer models the QoS property to be verified. The variable *frame* counts the number of frames to check if two consecutive loss occurs within a second (i.e. within 30 frames). The read port of the Observer is synchronized with the read port of the Player. A variable *lost* is associated with the read port records a frame loss.

6 Results

This section sketches QoS probabilities estimated from the two analysis approaches presented in previous sections. We also tabulate the buffer size savings obtained using synthetic traces.

We implemented the analytical framework described in Subsection 5.1 in MATLAB. The experiments were conducted for a low-bit rate and low resolution clips $(352 * 240)$ obtained from an open source [33]. The bit-rate of the input video is 1.5 Mbits per second and the frame output rate is 30fps. We used an MPEG2 implementation optimized for speed [19]. The MPEG2 source was annotated to get the number of bits corresponding to each compressed macroblock. The execution cycles for each macroblock is obtained from the software simulator SimpleScalar. Recapitulate that the number of bits and execution cycles per macroblock are inputs to the analytical framework. We chose the video files `cact.m2v, mobile.m2v,` and `tennis.m2v` for our experiments.

To construct the synthetic clips we set the lower bound for bits (e.g to 60) and the lower bound for execution cycles (e.g to 9000). Then from the actual trace containing the number of bits and execution cycles per macroblocks, synthetic traces are obtained; any value below the lower bound is modified to the lower bound. Figures 13, 15, and 17 show the probability that the buffer underflow is greater than two consecutive frames over any time interval (this refers to Equation 14 in Subsection 5.1). Figures 13, 15, and 17 also show the probability estimates from the BIP framework. Following are the observations:

- Increase in playout delay decreases the amount of buffer underflow, so, probability that the buffer underflow is more than two consecutive frames decreases.

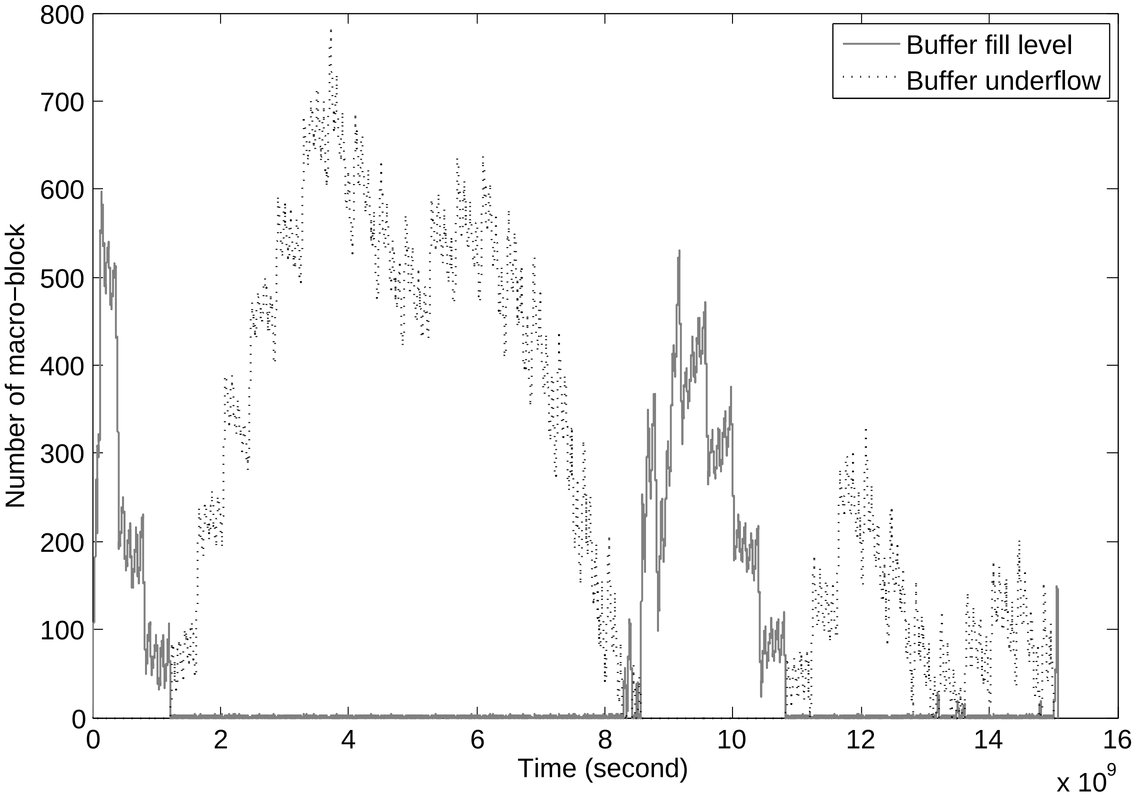

Figure 10: Playout Buffer fill level and underflow

- The estimates from stochastic real-time calculus upper bound the statistical model checking results as the analytical framework captures the worst-case behavior.
- The delay values at which the statistical model checking starts to state that the property is true is not same for the analytical framework. The analysis using the stochastic real-time calculus should be used to determine a small set or range of delay values. Later, to precisely verify the property detailed simulation should be carried out.
- For each probability estimation, the number of traces statistical model checking simulated ranged from 44 to 1345. For each trace, the method took around 6 to 8 seconds to verify the property. The probability of error for the probabilistic estimates from the statistical model checking is bounded by 0.01.

Figures 14, 16, and 18 plots the results of the buffer sizes for various playout delay values, and corresponding probabilistic bounds. These results correspond to simulation and statistical model checking. We observe that buffer size reduces substantially even for a small decrease of probabilistic value. For instance, there could be a buffer size reduction of 40% for a increase in the value of the probabilistic bound from 0 to 0.2 (Figures 16). In fact the buffer savings can be larger if we compare the buffer size required for no underflow and the memory required for the QoS property to be always true.

Finally, we note that the property we specified using stochastic real-time calculus computes the probability estimates for buffer underflow greater than two consecutive frames over any time interval. The statistical model checking on the other hand reports probability estimates for buffer underflow over two consecutive frames within 30 frames. The analytical framework is not flexible enough to express the exact property. In the following section, we discuss this specific problem and other modeling issues.

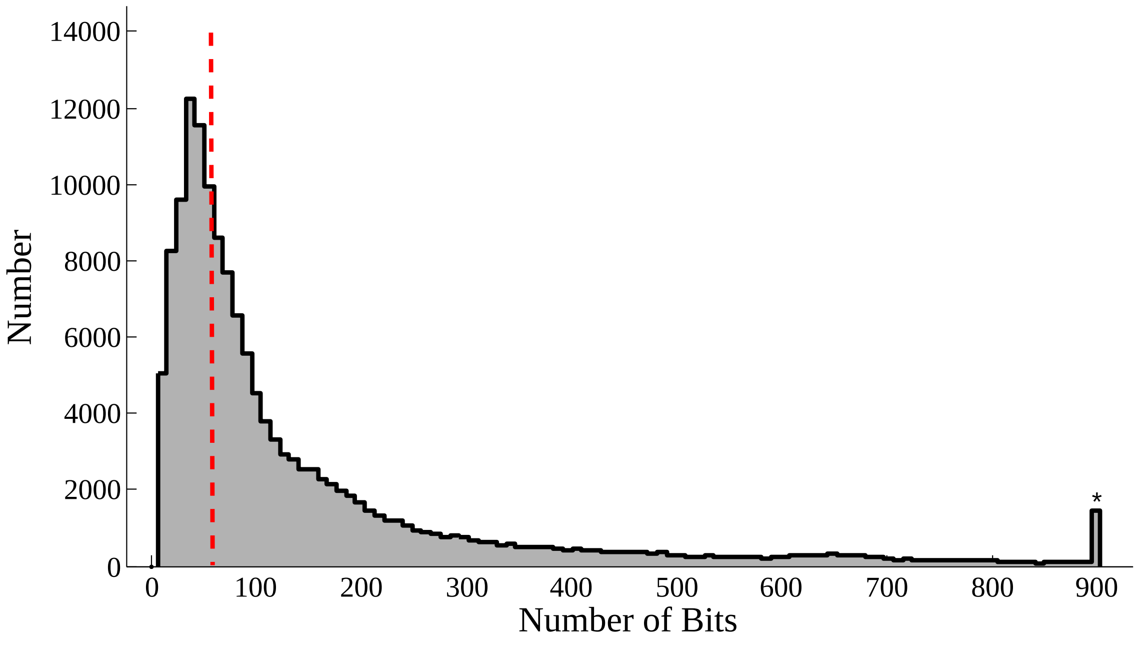

Figure 11: Probabilistic distribution function for the bits in the actual trace for `cact.m2v`. The red line shows how the synthetic trace is constructed; any value lower than the number of bits corresponding to the red line is modified for the synthetic trace; the new bits are the bits corresponding to the point given by the red line.

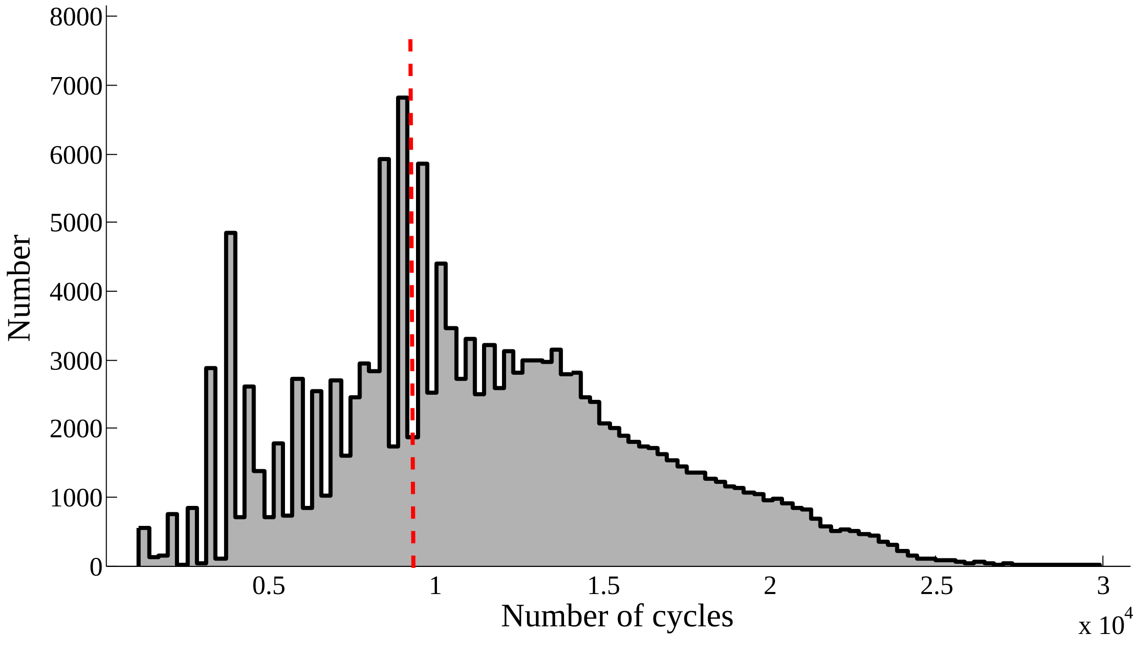

Figure 12: Probabilistic distribution function for the cycles in the actual trace for `mobile.m2v`. For the construction of synthetic trace, the points towards the left of the red line are modified to the number of cycles corresponding to the red line.

6.1 Comparison with no buffer underflow

In this subsection, we compare the playout delay and QoS trade-off for two scenarios: (1) buffer

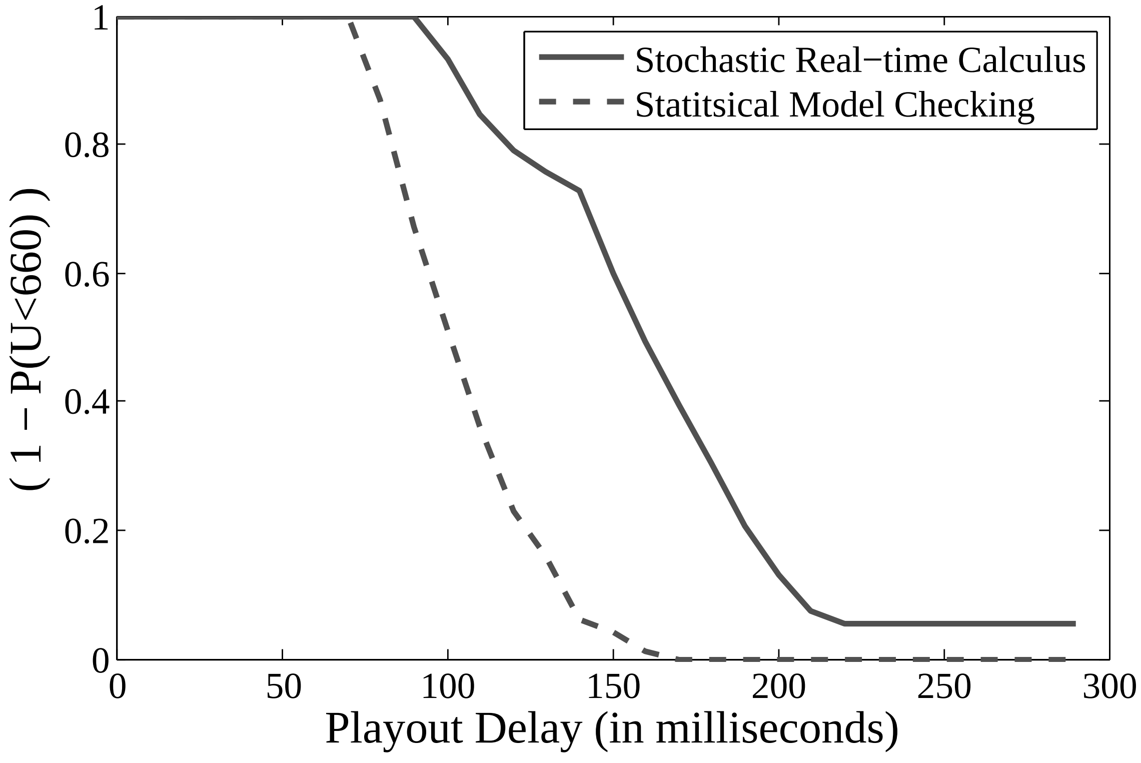

Figure 13: Probabilistic bounds for `cact.m2v Probability that the property is true for two analysis approaches. The estimates from the stochastic real-time calculus upper bounds the statistical model checking technique.`

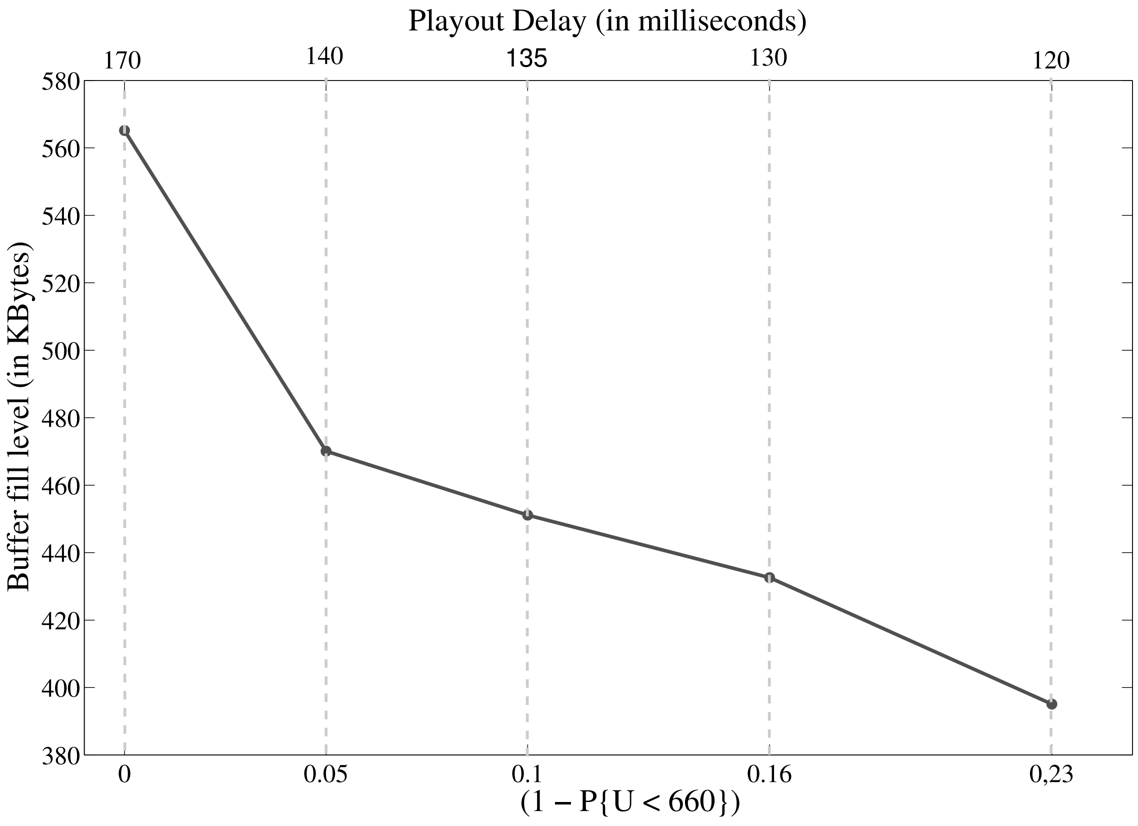

Figure 14: Playout buffer fill level for `cact.m2v`.

underflow with two consecutive frames, and (2) no buffer underflow. Figure 19 plots the probability values for different playout delay values for the above mentioned two scenarios. Observe in Figure 19 that the curve corresponding to the scenario with no buffer underflow is far right to the curve showing the buffer underflow with two consecutive frames. Thus the playout delay to guarantee no buffer underflow will be larger than with any finite underflow. Consequently, the buffer size required for no buffer underflow will be significantly large.

7 Discussion

The results in the previous section confirms our hypothesis that, for tolerable loss in video playout, output buffer size could be significantly reduced compared to the buffer size required for playing lossless video. In this section, first we speculate the combined strengths of both the approaches when used together in a system design flow. Then, we focus on requirements for our hypothesis to be used in practice: extraction of synthetic clips from benchmark video clips and modeling loss of macroblocks as deadline misses (instead of dropping stream objects).

7.1 Combined Design Flow

We now identify the benefits of the two techniques when used independently in a design-flow and speculate their combined strengths.

Analysis using stochastic real-time calculus followed by detailed simulation is a natural order for these techniques in a SoC design-flow; quick analysis is needed for estimating the application/architecture parameters at the beginning of the first stage of the design process, whereas at the end detailed simulations are required for verifying properties. The drawbacks, however,

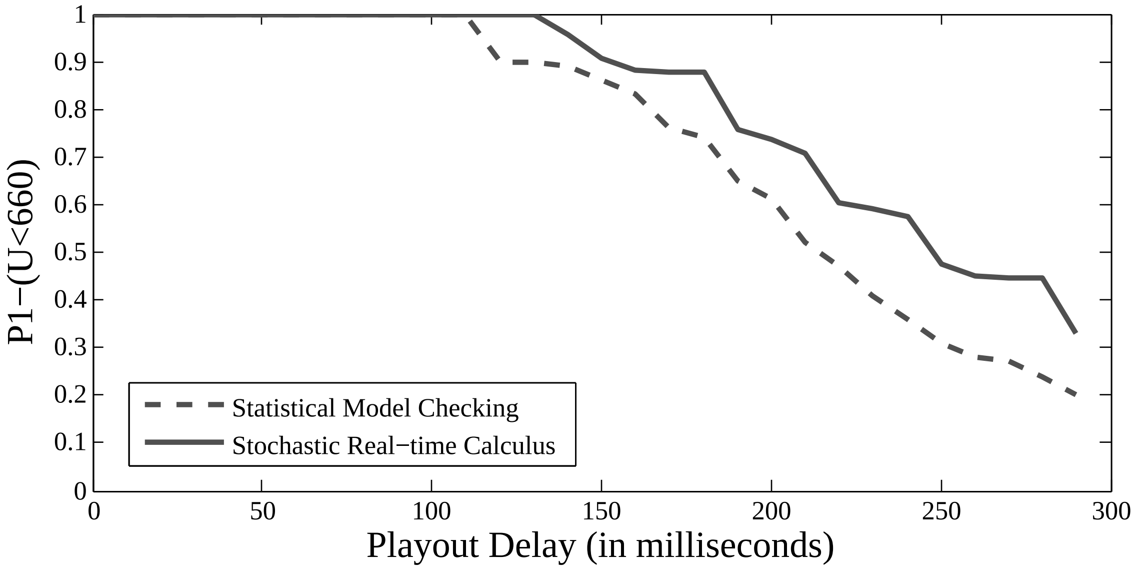

Figure 15: Probabilistic bounds for `mobile.m2v`

when the analytical framework and simulation are applied independently are: (1) the stochastic real-time calculus is not flexible enough to specify all verification properties, and (2) in using just the statistical model checking approach, the iterative process can be time-consuming for deciding parameters (in the QoS property).

On the other hand, a combined framework will not only remove the drawbacks mentioned above but will increase the expressivity of the stochastic real-time calculus (state-space notation) and the performance analysis using algebraic approach comes for free to the BIP framework. Now we explain about this combined designed flow in detail.

This paper juxtaposed the analytical framework with model-checking technique for comparing the two approaches. There is another motivation in this effort — which is not the central point of this paper: we wanted to consider the possibility of joining (stochastic network calculus based) analysis with (BIP simulation based) model checking (along the lines of Christel Baier et al., [1]). Towards this combination goal, we now identify the individual highlights of the two techniques and speculate their combined strengths.

The different levels of abstraction at which the mathematical framework and the model checking set-up models the SoC naturally positions the use of the techniques at two different points in the design cycle [7].

Typically, at the early design-cycle phase, system designers require back-of-the-envelope calculations for determining certain architecture parameters. At this point we recommend the stochastic network calculus based analysis approach primarily due to two reasons: (1) the modeling effort can be minimal if some off-the-shelf tools are used (for example the real-time calculus tool box [35]), and (2) the analysis is fast and relatively accurate to the detailed simulation technique.

Suppose we are at that phase of the design cycle where the system design is developed and there is a software simulator of the system model. Irrespective of whether the simulator is abstract or detailed, the statistical model checking approach is applicable. Following are the two primary benefits: (1) the model checking approach is not tied to any standard model, so, it can use as input traces from any simulation model; (2) the technique can be quite fast even if the property has to be verified for a large system; an abstract model could be built and the model checking could verify the abstract model.

[7]Even though the BIP based approach is best-suited in the later stages of a design-cycle, the simulation framework is capable to model any level of abstraction, that is, from system specification to code generation

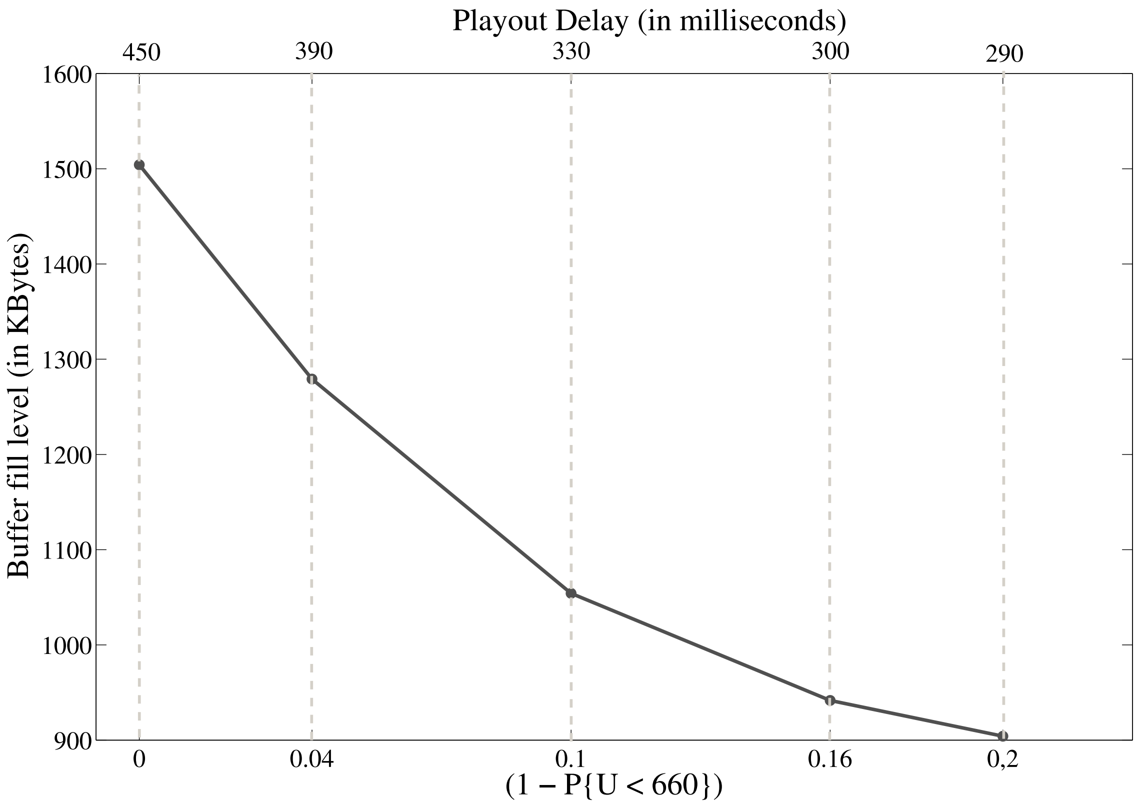

Figure 16: Playout buffer fill level for `mobile.m2v`.

Having said that, let us look at the drawbacks of using the two approaches individually for the same design objective, say, verify a system property. The stochastic network calculus based analysis approach does not provide flexibility of specifying *any* property. For example, consider that we want to verify if the sum of the input buffer and the playout buffer at any time instance during video decoding is within a certain limit. The calculus approach cannot handle this property as we are not modeling concurrency. On the other hand consider that we want to verify the same property for a range of values for an application parameter (e.g. initial playout delay). The simulation based statistical model checking approach could be computationally expensive for such experiments.

Now consider that the design-flow is such that certain parameters are decided using the mathematical framework and thereafter using detailed simulation we verify several properties for a concrete set of architecture and application parameters (chosen during the analysis approach). This approach that we are contemplating is a natural order and widely known to the design community. We envision more tight coupling of the analysis approach with the simulation model: can we specify stochastic arrival and service curve using the BIP framework? Such an effort would bring the state-space notation to the stochastic real-time calculus and the performance analysis using algebraic techniques comes for free for the BIP framework (refer [23]). The statistical model checking can be applied either on the abstract model or a detailed model of the BIP.

7.2 Inputs to the Model

The primary motivation to generate synthetic clips is that a system designer would use the synthetic clips instead of actual video clips when deciding architecture parameters such as buffer size. The question is then how to generate synthetic clips from given actual video clips.

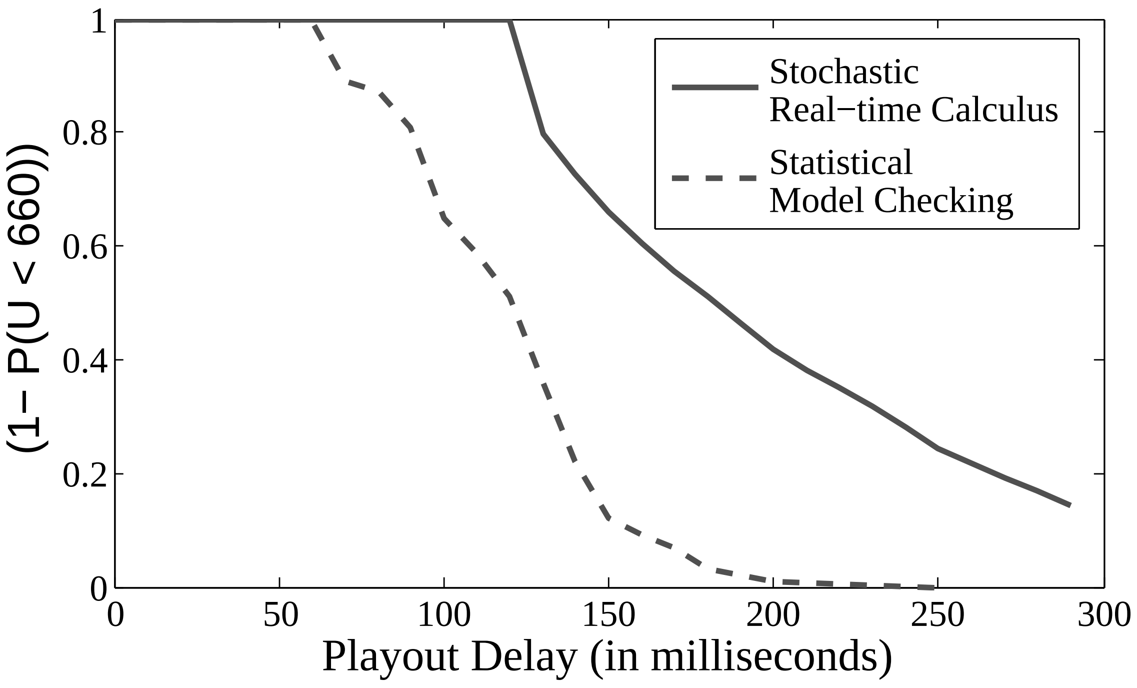

Figure 17: Probabilistic bounds for `tennis.m2v`

Currently, we choose at which point in frequency distribution of the input data (bits and cycles) we need to cut and reshape frequency distribution to generate the synthetic data. We choose this cut-off point based on two objectives: (1) to get significant reduction in playout buffer size, and (2) the loss in video is tolerable when the actual video clip is running in the SoC, which is designed using synthetic clips.

The approach we use to generate synthetic data can be time-inefficient as it involves iteration in making a choice to cut the distribution and analyze using our framework to check if our objectives are met. Instead of this trial and error technique, we are currently researching on a more sound technique to generate synthetic clips (for example, Yanhong et al., [20] use error percentage and eliminate some tail data from the distribution). To further this thought, notice in the analytical framework the need for synthetic clips arises when we compute the stochastic bounding functions. What if we use standard tail distribution functions that could characterize multimedia data accurately?

Assuming that these tail distributions could be found (refer Jelenkovic et al., [12]), then the task of generating synthetic clips reduces to a litmus test: does the synthetic clip generated conforms to the stochastic bounding function? In other words, without performing the iterative analysis — choosing a cut-off point first for synthetic clip generation, second checking if the video loss is tolerable, and then choosing a different cut-off point — we just check if a set of synthetic clips generated conforms to the stochastic bounding function.

7.3 QoS Specification

In our model, when we say video loss, we precisely mean that the macroblocks missed their deadlines; we do not model video loss as drop in macroblocks or frames, as studied in [7], where the authors present an analytical framework to study the trade-off between buffer size and video quality required for a multimedia decoder in the context of frame drops. In correspondence to display of the video, as video loss is interpreted as deadline miss, the display device awaits until all macroblocks are ready for display. If there were no deadline miss, for example, a frame would

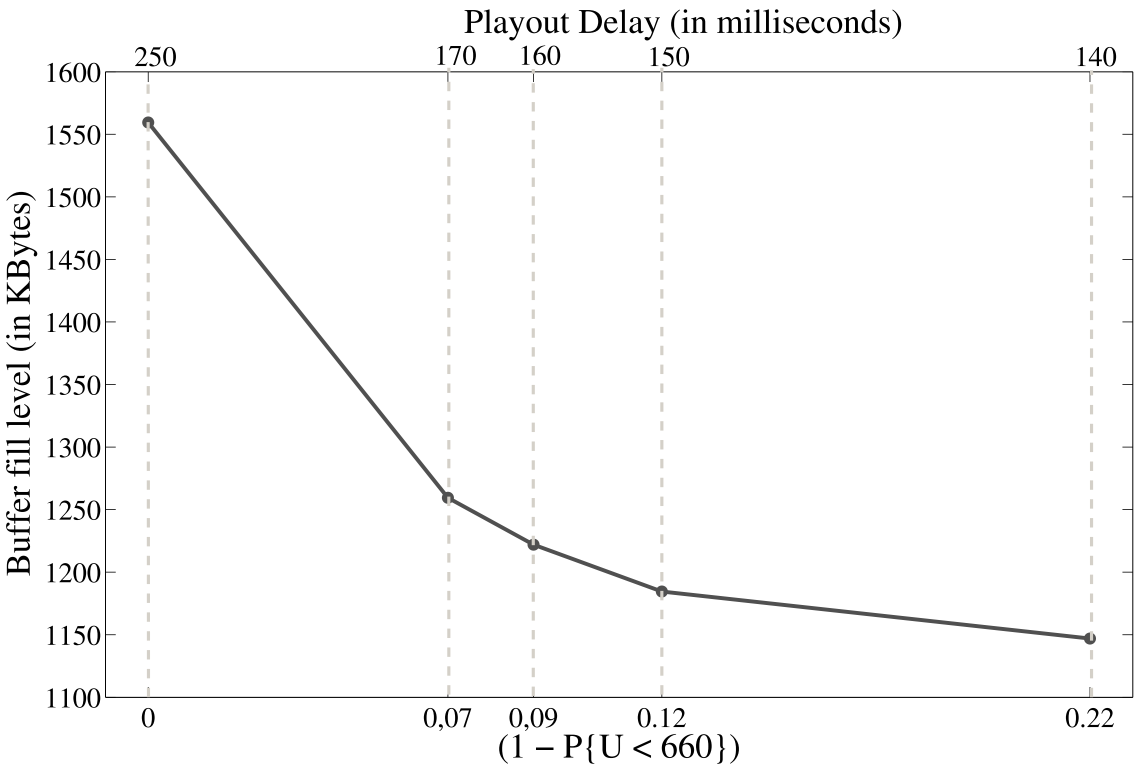

Figure 18: Playout buffer fill level for `tennis.m2v`.

be displayed at the right time.

The QoS property we verified using our set-up required to check if there is a loss of two consecutive frames within 1 second. This should be read in the context of our model as the display of two consecutive frames being delayed within 1 second.

8 Related Work

In this section, we present the state-of-the-art in characterizing the stochastic behavior of multimedia applications and compare it with our approach. We will focus mainly on analytical approaches (refer previous work for survey on simulation based approaches [4, 18]). Figure 20 classifies existing work in the following manner: (a) modeling for general and multimedia specific systems, and (b) the kind of analysis that an analytical framework provides (e.g., worst-case or average-case).

A domain-agnostic approach is to statistically analyze the execution variance of soft-real time applications (Kumar et al., [16]). Using profiling, components of the applications that lead to variable execution times are first identified. Then programmers can identify components that affect real-time behavior of the application in the context of other components. Thus, the programmers need not resort to ad-hoc methods for tuning applications for expected real-time behavior.

Our approach differs from the domain agnostic techniques in the following way: our model tightly couples the application and architecture; in what follows, we also discuss how both the stochastic real-time calculus and statistical model checking are an excellent fit for streaming applications. On the other hand, domain-specific techniques in a probabilistic setting perform analysis at task granularity (Yaldiz et al., [37] and Iqbal et al., [11]). In stochastic real-time calculus, our model captures input, execution, and output streams of the SoC. The granularity of the stream object can be at any level: bits, macroblock, frame, and group of pictures.

Iqbal and others [11] proposed scheduling techniques for soft-real time systems. The task execution times are stochastic and the solution for scheduling is based on an online Monte Carlo

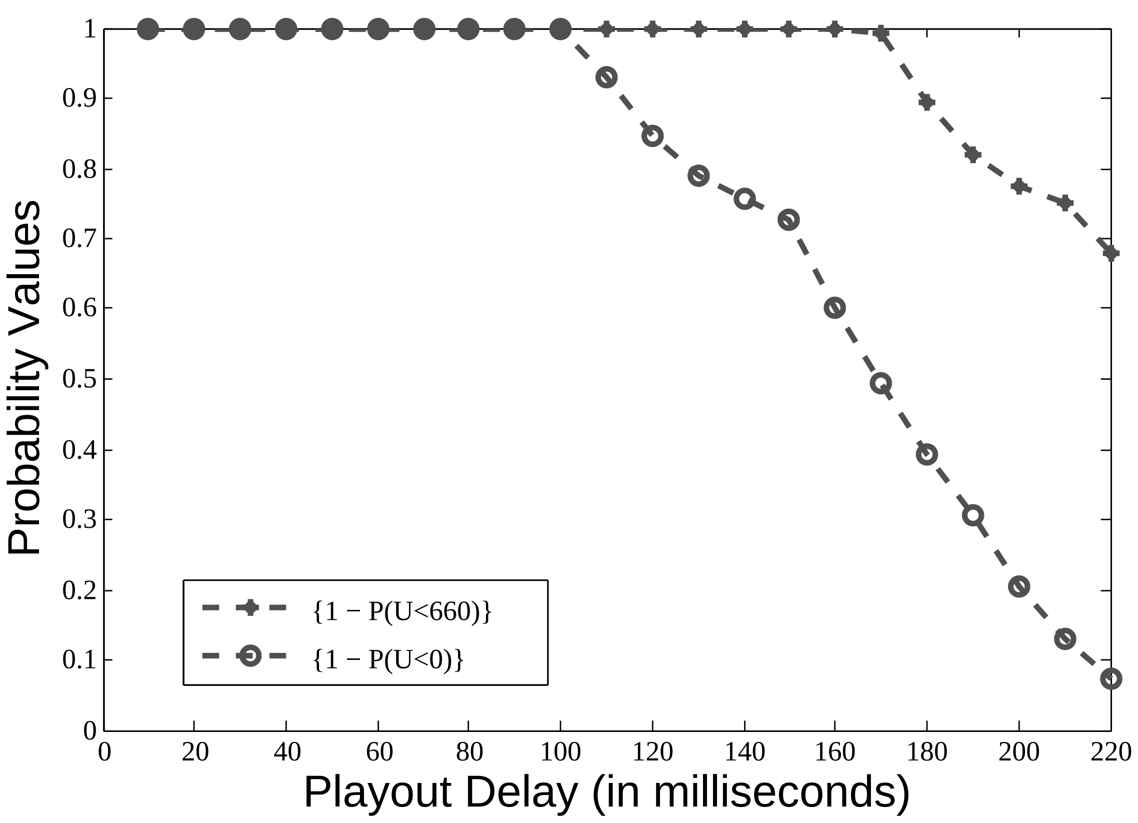

Figure 19: QoS and playout delay trade-off for video file `Cact.m2v` for two scenarios: (1) The buffer underflow is less than two consecutive frames, and (2) no buffer underflow.

method on a joint space model of all tasks. The objective of this technique is similar to ours: reducing memory and computational requirements. The complexity of this technique, however, as the authors report, does not scale well for task graphs of huge size. Yaldiz and others [37] use stochastic modeling of the applications to obtain policies for energy savings and for providing probabilities for satisfying timing constraints. Their model considers set of concurrent tasks and takes into account data dependence, precedence relations and timing constraints.

Our methodology differs from the above two discussed approaches in the following way: (1) there is a tight characterization of inputs for streaming applications using arrival curves in the stochastic real-time calculus and randomly generated clips (based on distributions) in statistical model checking, and (2) we provide probabilistic guarantees instead of average-case analysis in the analytical framework.

Liu and others [20] introduced a new concept called approximate variability characterization curves (or Approximate VCCs), to characterize the average-case behavior of multimedia workloads in a parameterized fashion. The crucial difference in comparison to our work is that Approximate VCCs belongs to a family of average-case analysis; instead of probabilistic guarantees on buffer size they bound the error due to their analysis. Also, the framework remains in a deterministic setting after the alteration of workload curves. So, it is not elegantly able to capture stochastic nature of arrivals and stochastic nature of execution times.

Our framework is a complete stochastic framework and could fully exploit the probabilistic nature inherent in the MPEG streams towards designing low-cost designs. Our models differ from Liu et al's in the following way:

- They conclude that Approximate VCCs cannot support a class of arrival streams.
- The tail distribution of the workload curves do not exploit the inherent property of MPEG

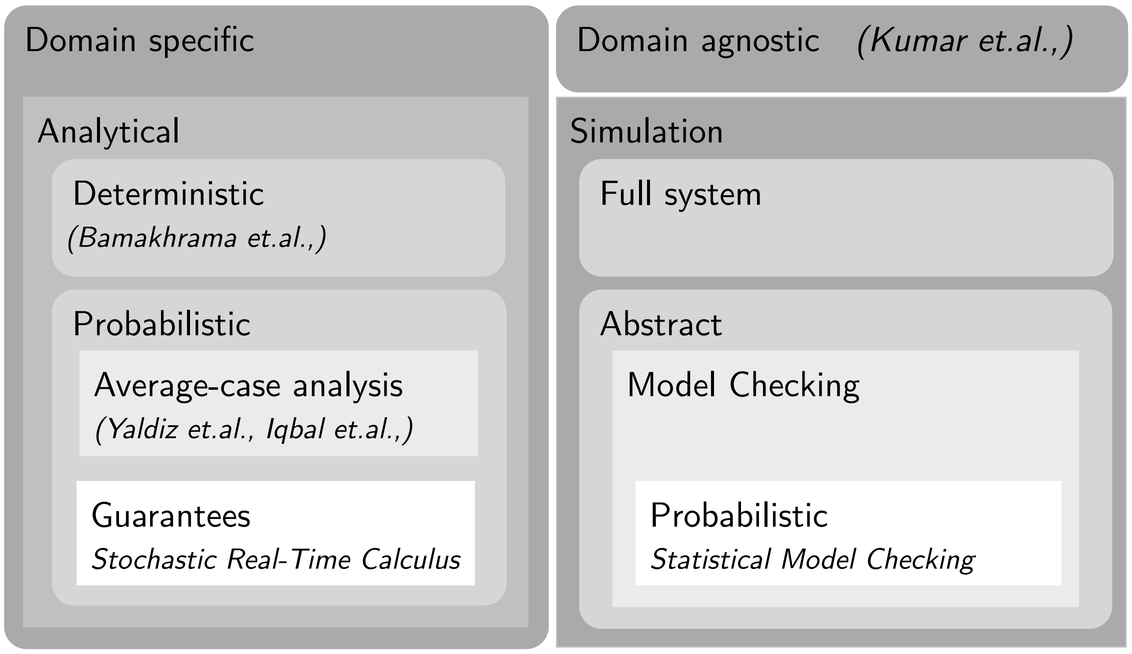

Figure 20: Our model in the context of existing techniques. Stochastic real-time calculus provides upper and lower bounds (which are guarantees) on the probabilistic estimates. Statistical model checking verifies the QoS property on an abstract system using detailed simulation.

streams, namely, the sub-exponential property. Using this property allows us to capture a class of streams both at the arrival and at the service.

- Unlike Approximate VCCs, we use stochastic network calculus to observe the backlog at the playout buffer.

- The QoS is handled differently in Approximate VCCs in that the streams are dropped.

To conclude this section, we now list the limitations of our framework, and discuss how we can overcome these limitations:

1. In modeling the SoC architecture, we assumed that the processor never stalls; the playout buffer is large enough that the buffer never gets full, so, the processor can always write to the buffer; similarly, there is always an item in the input buffer that the processor can consume for processing. In a deterministic setting, however, when modeling using real-time calculus, the memory contention has been addressed [22, 31]. We believe that modeling memory contention in a probabilistic setting is possible too.

2. In the current presentation of this work, we did not address how multiple streams can be handled as inputs and outputs to the SoC. For example, there can be multiple videos, which are being played in parallel, and which share the processing and memory resources. Alternatively, there could be multiple tasks concurrently running in the processor which may not share the buffer memory. Along the same lines, we only presented a single processor architecture in this work; there could be many processors connected in pipeline. It is important, however, to note that the limitations listed above are natural extensions to the framework presented in this paper. Towards this, existing research which has addressed the scenarios listed above (in a deterministic setting) will be of interest to the reader [28, 26, 10].

9 Conclusions and Future Work

The stochastic real-time calculus and the statitsical model checking techniques were presented as frameworks for performance analysis of multimedia systems. The case-study analyzed the trade-off in quality over buffer size savings using the above mentioned approaches. Both the approaches estimated the probability that a certain QoS property is true; the analytical framework upper bounded the estimates from the statistical model checking.

Future extensions to this work are as follows: (a) modeling communication architectures such as bus, network-on-chip, and others; (b) modeling for variable bit-rate video and variable consumption rate; (c) integrating real-time calculus and BIP framework.

References

[1] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Performance evaluation and model checking join forces. *Communications of the ACM*, 53(9):76–85, September 2010. 7.1

[2] A. Basu, M. Bozga, and J. Sifakis. Modeling Heterogeneous Real-time Systems in BIP. In *Software Engineering and Formal Methods SEFM'06 Proceedings*, pages 3–12. IEEE Computer Society Press, 2006. 1

[3] Ananda Basu, Bensalem Bensalem, Marius Bozga, Jacques Combaz, Mohamad Jaber, Thanh-Hung Nguyen, and Joseph Sifakis. Rigorous Component-Based System Design Using the BIP Framework. *IEEE Software*, 28(3):41–48, May 2011. 4.1, 5.1

[4] Ananda Basu, Saddek Bensalem, Marius Bozga, Benoît Caillaud, Benoît Delahaye, and Axel Legay. Statistical Abstraction and Model-Checking of Large Heterogeneous Systems. In *Proc. of the International Joint Conference on Formal Techniques for Distributed Systems (FMOODS/FORTE)*, pages 32–46, June 2010. 1, 2.2, 8

[5] Saddek Bensalem, Marius Bozga, Benoît Delahaye, Cyrille Jégourel, Axel Legay, and Ayoub Nouri. Statistical Model Checking QoS Properties of Systems with SBIP. In *ISoLA (1)*, pages 327–341, 2012. 4.1, 4.2

[6] Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT Press, 2001. 4.1

[7] D. Gangadharan, L.T.X. Phan, S. Chakraborty, R. Zimmermann, and I. Lee. Video Quality Driven Buffer Sizing via Frame Drops. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2011 IEEE 17th International Conference on*, volume 1, pages 319–328. IEEE, 2011. 7.3

[8] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate Probabilistic Model Checking. In *Proc. of the Verification, Model Checking and Abstract Interpretation (VMCAI)*, pages 73–84, January 2004. 4.1, 4.2

[9] Kai Huang and Lothar Thiele. Performance analysis of multimedia applications using correlated streams. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '07, pages 912–917, San Jose, CA, USA, 2007. EDA Consortium. 3.1

[10] Kai Huang and Lothar Thiele. Performance analysis of multimedia applications using correlated streams. In *Proceedings of the ACM/IEEE Design Automation and Test in Europe (DATE)*, pages 912–917, 2007. 2

[11] N. Iqbal and J. Henkel. SETS: Stochastic execution time scheduling for multicore systems by joint state space and Monte Carlo. In *Proc. of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 123–130, November 2010. 8

[12] Predrag R. Jelenkovic, Aurel A. Lazar, and Nemo Semret. The Effect of Multiple Time Scales and Subexponentiality in MPEG Video Streams on Queueing Behavior. *IEEE Journal on Selected Areas in Communications*, 15(6):1052–1071, August 1997. 7.2

[13] Yuming Jiang and Yong Liu. *Stochastic Network Calculus*. Springer, 2008. 1, 3.2

[14] M. Krunz, R. Sass, and H. Hughes. Statistical characteristics and multiplexing of MPEG streams. In *Proc. of the Conference on Computer Communications (INFOCOM)*, pages 455–462, April 1995. 5.2

[15] Marwan Krunz and Satish K. Tripathi. On the characterization of VBR MPEG streams. In *Proc. of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 192–202, June 1997. 5.2

[16] Tushar Kumar, Romain Cledat, Jaswanth Sreeram, and Santosh Pande. Statistically Analyzing Execution Variance for Soft Real-Time Applications. In José Nelson Amaral, editor, *Languages and Compilers for Parallel Computing*, pages 124–140. Springer-Verlag, 2008. 8

[17] S. Laplante, R. Lassaigne, F. Magniez, S. Peyronnet, and M. de Rougemont. Probabilistic abstraction for model checking: An approach based on property testing. *ACM Transactions on Computational Logic*, 8(4):30–39, August 2007. 4.1

[18] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: an overview. In *Proc. of the International Conference on Runtime Verification (RV)*, pages 122–135, November 2010. 8

[19] libmpeg2. A free MPEG2 video stream decoder. `http://libmpeg2.sourceforge.net`, 2006. 6

[20] Yanhong Liu, Samarjit Chakraborty, and Wei Tsang Ooi. Approximate VCCs: a new characterization of multimedia workloads for system-level MpSoC design. In *Proc. of the ACM/IEEE Annual Design Automation Conference (DAC)*, pages 248–253, June 2005. 7.2, 8

[21] Alexander Maxiaguine, Simon Künzli, Samarjit Chakraborty, and Lothar Thiele. Rate analysis for streaming applications with on-chip buffer constraints. In *Proc. of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 131–136, January 2004. 1, 5.1

[22] Rodolfo Pellizzoni, Andreas Schranzhofer, Jian-Jia Chen, Marco Caccamo, and Lothar Thiele. Worst case delay analysis for memory interference in multicore systems. In *Proceedings of the ACM/IEEE Design Automation and Test in Europe (DATE)*, pages 741–746, 2010. 1

[23] Linh T. X. Phan, Samarjit Chakraborty, and P. S. Thiagarajan. A Multi-mode Real-Time Calculus. In *Proc. of the Real-Time Systems Symposium (RTSS)*, pages 59–69, November-December 2008. 7.1

[24] Balaji Raman. *Application-specific workload shaping in resource-constrained media players*. PhD thesis, School of Computing, National University of Singapore, July 2010. 1, 2.1

[25] Balaji Raman and Samarjit Chakraborty. Application-specific Workload Shaping in Multimedia-enabled Personal Mobile Devices. *ACM Transactions on Embedded Computing Systems*, 7(2):10, Feburary 2008. 3.1

[26] Balaji Raman and Samarjit Chakraborty. Application-specific workload shaping in multimedia-enabled personal mobile devices. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(2), 2008. 2

[27] Balaji Raman, Samarjit Chakraborty, Wei Tsang Ooi, and Santanu Dutta. Reducing data-memory footprint of multimedia applications by delay redistribution. In *Proceedings of the 44th annual Design Automation Conference*, DAC '07, pages 738–743, New York, NY, USA, 2007. ACM. 3.1

[28] Balaji Raman, Samarjit Chakraborty, Wei Tsang Ooi, and Santanu Dutta. Reducing Data-Memory Footprint of Multimedia Applications by Delay Redistribution. In *Proc. of the ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 738–743, 2007. 2

[29] Balaji Raman, Guillaume Quintin, Wei Tsang Ooi, Deepak Gangadharan, Jerome Milan, and Samarjit Chakraborty. On buffering with stochastic guarantees in resource-constrained media players. In *Proc. of the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 169–178, September 2011. 1, 5.1

[30] Luca Santinelli and Liliana Cucu-Grosjean. Toward probabilistic real-time calculus. *ACM SIGBED Review*, 8(1):54–61, March 2011. 1

[31] Andreas Schranzhofer, Rodolfo Pellizzoni, Jian-Jia Chen, Lothar Thiele, and Marco Caccamo. Worst-case response time analysis of resource access models in multi-core systems. In *Proc. of the ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 332–337, 2010. 1

[32] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical Model Checking of Black-Box Probabilistic Systems. In *Proc. of International Conference on Computer Aided Verification (CAV)*, pages 202–215, July 2004. 4.1

[33] Tektronix. MPEG Elementary Streams. `ftp://ftp.tek.com/tv/test/streams/Element/index.html`, 1996. 6

[34] A. Wald. Sequential Tests of Statistical Hypotheses. *Annals of Mathematical Statistics*, 16(2)(2):117–186, June 1945. 4.1

[35] Ernesto Wandeler and Lothar Thiele. Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox`, 2006. 7.1

[36] Duminda Wijesekera and Jaideep Srivastava. Quality of Service (QoS) Metrics for Continuous Media. *Multimedia Tools and Applications*, 3(2):127–166, July 1996. 2.2, 5.1

[37] Soner Yaldiz, Alper Demir, and Serdar Tasiran. Stochastic Modeling and Optimization for Energy Management in Multicore Systems: A Video Decoding Case Study. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 27(7):1264–1277, July 2008. 8

[38] Hakan Lorens Samir Younes. *Verification and Planning for Stochastic Precess with Asynchronous Events*. PhD thesis, School of Computer Science, Carnegie Mellon University, January 2005. 4.1, 4.2

[39] Nicholas H. Zamora, Xiaoping Hu, and Radu Marculescu. System-level performance/power analysis for platform-based design of multimedia applications. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 12(1):1–29, January 2007. 1