On Buffering with Stochastic Guarantees in Resource-Constrained Media Players

Balaji Raman[1], Guillaume Quintin[1], Wei Tsang Ooi[2],
Deepak Gangadharan[2], Jerome Milan[1], Samarjit Chakraborty[3]

[1]École Polytechnique, Laboratoire d'informatique (LIX), Palaiseau, France
[2]Department of Computer Science, National University of Singapore, Singapore
[3]Institute for Real-Time Computer Systems, TU Munich, Germany
{balaji, quintin, jerome.milan}@lix.polytechnique.fr
{ooiwt, gdeepak}@comp.nus.edu.sg, samarjit@tum.de

ABSTRACT

Playout delay or buffering are commonly used in the case of streaming multimedia to ensure smooth playout. A large delay, however, is required for promising a high quality in display. Such significant delays consume huge on-chip memory. We show that when the constraints on output are slightly relaxed, the playout delay needed can be reduced to a negligible value with no perceivable loss in video quality.

We propose a mathematical framework that precisely estimates that minimal playout delay value. Perhaps more importantly, unlike existing analytical models, our framework allows to specify loss and provides guarantees that the desired display quality is achieved with the chosen delay. We present simulation results to validate the minimum delay value obtained from the analytical framework. Using the reduced delay value enormously saves the on-chip memory requirement.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems

General Terms

Design, Performance, Theory

Keywords

playout delay, multimedia systems, probabilistic analysis

1. INTRODUCTION

Given the constraints of application imposed on limited hardware resources, it is a challenge to design System-on-Chip (SoC) for media-processing devices [6]. Using relaxed constraints on the quality of the display can yield markedly significant savings of many on-chip resources, including processor capacity requirement and memory capacity.

Studies have shown that multimedia applications can tolerate certain loss in quality, and this deterioration in quality is not perceivable up to some extent [1, 20]. These quality degradations have been previously shown in the literature to save on-chip resources, albeit there were no guarantees on the design and SoCs were built to handle only average-case scenarios [21, 8].

We propose a framework where loss in quality could be represented and mathematically quantified using probabilistic parameters. An application parameter, namely the playout delay, is adjusted so as to reduce the on-chip resource requirements. Using a system model, we explain how playout delay variable could be used in a probabilistic constraint on display quality.

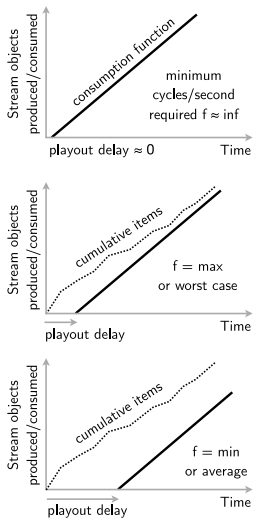

Figure 1: Playout delay and processor frequency.

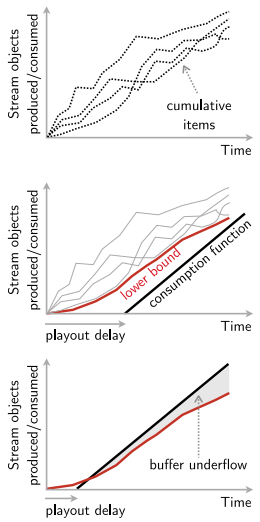

Figure 2: Playout delay with buffer underflow.

Consider a SoC with a processing element and two on-chip memories, the input and the playout buffer. The input buffer temporarily stores the data items from a multimedia

CODES+ISSS'11, October 9–14, 2011, Taipei, Taiwan.

stream. The multimedia application being executed in the processor fetches data from the input buffer, and stores its output in the playout buffer. The output device displays items in the playout buffer at a constant rate. For example, a video decoding application decompresses the input stream and the decoded items are displayed at a fixed rate (say 30fps).

The amount of processing cycles required for each stream object (e.g., a macroblock or a frame) is variable [1]. If the processing element runs at some constant speed then the number of stream objects produced per unit time is variable. But since the display device consumes stream objects at a constant rate it is possible that the playout buffer is empty at times, leading to buffer underflows. If the buffer should never underflow, previous studies have shown that the display must start after an initial playout delay [12]. We now explain this playout delay concept.

Figure 1 shows three scenarios where the output device consumes stream objects after a near zero, small, and large delay (top, middle, and bottom respectively). Assume that we are given an input stream for processing. The middle and bottom sketch in Figure 1 shows the cumulative number of stream objects produced with two different processor speeds (for the given input stream).

The cycles per second requirement is almost infinity when the consumption starts at the same time as the production (refer top graph in Figure 1). When the playout starts after a small delay, the required production rate is still high. The reason for this need in processing resources is as follows.

To guarantee that the buffer should never underflow, we should consider the worst-case scenario where stream objects would take maximum processor cycles to execute. The processor must produce them at a high rate, so that during the worst-case, enough items are produced to meet the playout rate (middle graph in Figure 1).

With considerable initial delay, previous work in this direction have shown that the processor cycles per second requirement reduces to the average cycles per second required to process the stream [12]. While full quality is assured, this delay could be large (see bottom graph in Figure 1). In this paper, we propose to relax the output constraints, and determine the minimum playout delay required so as to reduce the playout buffer size required.

We designed a mathematical framework, which estimates a lower bound on the output, given the input characteristics and processing requirements for a class of streams [12]. We build on this existing deterministic framework to have probabilistic constraints on the output. Now we explain how the lower bound is tied to the playout delay and the buffer underflow.

Suppose that we are required to find the minimum playout delay such that it is guaranteed that the playout buffer never underflows so that the display quality is always met. In the top most graph in Figure 2, the cumulative processor cycles correspond to three specific video streams. The cycles per time shows the actual cycles consumed during the processing of the stream.

Assume that we have a lower bound on the cycles/second consumed of streams belonging to a class of videos with same bit-rate and resolution, that is, without the knowledge of the actual processor cycles consumed (shown in Figure 2). From the literature, we know how to estimate a lower bound on the processor cycles consumed for a class of streams [12]. In middle of Figure 2, we see that the chosen playout delay is such that the lower bound is always higher than consumption.

If the delay value is lowered, the constraint on display quality is relaxed, which leads to buffer underflow at times (refer bottom graph in Figure 2). The amount of buffer underflow could be specified using stochastic constraints. The given input streams satisfies the underflow constraint, if the lower bound satisfies it. This reduction in the playout delay required also leads to reduced buffer size.

To the best of our knowledge, our framework is one of the few to provide analysis allowing loss in quality, whilst providing guarantees for designing multimedia. Present analytical modeling techniques are inadequate for designing multimedia-processing system-on-chips, either because the models do not provide any guarantee on output quality [11], or because their analysis are for worst-case scenarios and therefore often lead to cost-inefficient designs [19]. Our work is inspired from the research developments in the computer networks area towards stochastic network calculus [5] for performance analysis.

1.1 Illustrative Example

In this subsection, we show that the initial playout delay determines the buffer size, and hence any reduction in delay consequently leads to smaller buffer.

We took a video stream of resolution 352×240 that was MPEG2 compressed to a bit-rate of 1.5 Mbps such that 30 frame per second could be displayed. A complete SystemC simulation set-up was built; the video stream was decoded and played-out after different initial delay values. During each experiment, we noted the maximum playout buffer underflow and playout buffer size required (such that there is no overflow). Two important observations were made as the delay was increased.

1. The increase in delay increases the maximum fill level of the buffer; increase in delay raises the level of buffer fill. The size of the playout buffer at the maximum fill level is the required playout buffer size. So, any increase in delay increases the buffer size.

2. Second, an increase in delay decreases the number of times the buffer underflows (refer Figure 3). The consumption is constant unlike the playout buffer fill. Correctly chosen initial delay makes it possible to store sufficient items before consumption starts. Even-though the buffer fill continues to vary, buffer underflow is greatly reduced.

If the playout delay value is reduced, then the buffer size is reduced too [2], with an increase in buffer underflow (which in turn reduces the video quality). Our mathematical framework, estimates for a given set of streams the minimum playout delay required such that the desired display quality is achieved.

[1] the number of bits per stream object is variable

[2] We present results in macroblocks instead of bytes for generaility. For a decompression application, the playout buffer has to hold uncompressed video. So, for MPEG decoding, the reduction in terms of hundreds of macroblocks will lead to huge reduction in terms of bytes.

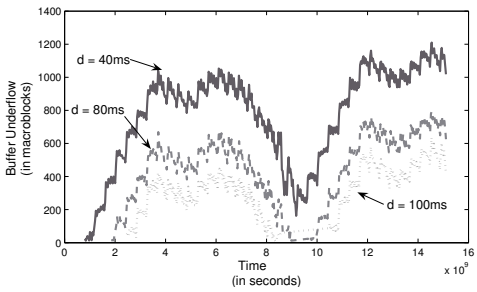

Figure 3: Playout buffer underflow over time. The variability in underflow substantially reduces with large increase in playout delay.

1.2 Contributions

We discuss related work in detail later in the paper, but to summarize, our main contributions are:

- *Stochastic Guarantee:* Even though the system designed with our probabilistic mathematical framework does not *always* output 100% frame rate, the designer has confidence on the performance of the system in that it would at least output $(100 - \Delta)\%$ frame rate. The huge savings in resources (processor capacity and on-chip memory size) is achieved with only slight reduction in the throughput. This cost-effective design is the primary motivation for the system architect to design devices with probabilistic guarantees.

- *Insight:* The designer achieves the following with tuning the playout delay parameter: (a) sets the media player's QoS, and (b) analyzes the architectural parameters such as the processor speed and buffer size. This application/architecture co-design is possible first with identifying the non-trivial correlation among the playout delay, QoS, and buffer size. Second, the mathematical solution presented in this paper characterizes the correlation and hence the solution for the delay is central for the co-design. Our work illustrates that a mathematical framework can provide insights into the design of the system.

1.3 Organization

We recommend the reader to first understand the preliminaries of our mathematical framework by reading Section 2. Then the reader will be able to follow how we formulate the stochastic constraint in Section 3. Thereafter, to get a quick idea on the results, one can jump to Subsection 5.1, where we numerically evaluate the delay value. The first reading can then end with the conclusions in Section 8.

Two approaches for the closed-form estimation of the minimum delay is presented in Section 4. The results for these approaches are presented in Section 5.2. Section 6 is a brief survey on existing mathematical framework attempting to solve similar problems addressed in this paper. Section 7 attempts to answer potential questions on this work.

2. SYSTEM MODEL

We now present the basic framework used to model the variations of the incoming rate of input items and the fluctuations of the associated processing requirements. Our system model, shown in Fig. 4, consists of a processor with an internal buffer b, a playout buffer B and a playout (or output) device. The processor decodes the input stream and writes the decompressed data in the playout buffer which is consumed by the playout device (e.g., a video display or a speaker) at a constant rate.

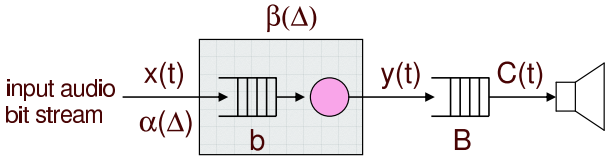

Figure 4: System Model

We assume that the input bit stream to be decoded is fed to the internal buffer b at a constant rate of r bits per second. For the sake of simplicity, we consider a stream to be made up of a sequence of *stream objects*, such as macroblocks in case of video playbacks. Given an input stream to be decoded, let $x(t)$ denote the number of stream objects filled into the internal buffer over the time interval $[0, t]$. Since the number of bits constituting a stream object can vary, $x(t)$ depends on the particular input stream to be decoded. To bound these variations, we introduce two functions α_l and α_u verifying,

$$\alpha_l(\Delta) \leq x(t + \Delta) - x(t) \leq \alpha_u(\Delta), \tag{1}$$

for all t and $\Delta \geq 0$. Here $\alpha_l(\Delta)$ and $\alpha_u(\Delta)$ respectively represents the minimum and maximum number of stream objects that can arrive in the internal buffer within *any* time interval of length Δ.

α_l and α_u are computed from two auxiliary functions $\phi_l(k)$ and $\phi_u(k)$ respectively denoting the minimum and maximum number of bits constituting *any* k consecutive stream objects. ϕ_l and ϕ_u are obtained by analyzing a large number of samples that are *representative* of the input streams to be processed by the target decoder. Once ϕ_l and ϕ_u are experimentally determined, we can compute their *pseudo-inverses* noted ϕ_l^{-1} and ϕ_u^{-1}. $\phi_l^{-1}(k)$ and $\phi_u^{-1}(k)$ respectively returns the maximum and minimum number of stream objects that can be constituted by k bits. Since we assume a constant input rate of r bits/sec, we compute the bounding functions α_l and α_u as

$$\alpha_l(\Delta) = \phi_u^{-1}(r\Delta) \text{ and } \alpha^u(\Delta) = \phi_l^{-1}(r\Delta). \tag{2}$$

Let $y(t)$ and $C(t)$ be the number of stream objects respectively written into the playout buffer B and consumed by the output device over the time interval $[0, t]$. For a smooth playback it is naturally required to assure that,

$$y(t) \geq C(t), \ \forall t \geq 0. \tag{3}$$

We model the *service* provided by the processor by a function β with $\beta(\Delta)$ representing the minimum number of stream objects that are guaranteed to be processed (if available in the internal buffer) within any time interval of length Δ. It can be shown that $y(t) \geq (\alpha_l \otimes \beta)(t)$, where $\otimes$ is the *min-plus convolution* operator defined as [8] $(f \otimes g)(t) = \inf_{0 \leq s \leq t}\{f(t - s) + g(s)\}$. Hence for the constraint (3) to hold, it is sufficient to have,

$$(\alpha_l \otimes \beta)(t) \geq C(t), \ \forall t \geq 0. \tag{4}$$

This can be written in a more usable form using the *min-plus deconvolution* operator defined by $(f \oslash g)(t) = \sup_{s \geq 0}\{f(t+s) - g(s)\}$. It is known from the duality between $\otimes$ and $\oslash$ [3], that for any three functions f, g and h, $g \otimes h \geq f$ if and only if $h \geq f \oslash g$. Applying this result to (4) we obtain,

$$\beta(t) \geq (C \oslash \alpha_l)(t),\ \forall t \geq 0. \tag{5}$$

Note that $\beta(t)$ in inequality (5) is defined in terms of the number of stream objects needing to be processed within any time interval of length t. However, the number of processor cycles needed to decode a stream object is not constant. We thus model the variability in the number of cycles required to process a stream object using two bounding functions γ_l and γ_u. $\gamma_l(k)$ and $\gamma_u(k)$ respectively returns the minimum and maximum number of cycles required to process k consecutive stream objects.

Finally, our model assumes that the playout buffer is read by the output device at a constant rate of c stream objects per second after an initial playout delay (or buffering time) of d seconds. The number $C(t,d)$ of stream objects read by the output device over the time interval $[0,t]$ is thus given by

$$C(t,d) = \begin{cases} 0 & \text{if } t \leq d \\ c(t-d) & \text{if } t > d. \end{cases} \tag{6}$$

3. MINIMIZING BUFFERING

In this section, we derive the minimum playout delay required to obtain an output rate with acceptable loss. With relaxed constraints on playout quality, we show that the initial delay can be substantially reduced thus minimizing the playout buffer size.

We first state the constraint to be satisfied for continuous playback of the stream such that the actual output rate is met. If the playout buffer never underflows, the output stream is played with no quality loss. So, the playout buffer should always have a greater cumulative number of stream objects than consumed by the output device. Obviously the initial playout delay increases the maximum playout buffer size required. This delay, however, can be reduced if we allow the playout buffer to underflow at times. We consequently slightly relax the playout constraints as detailed in Section 1 to be able to chose a smaller initial delay. A buffer underflow of b stream objects can be written as

$$C(s,d) - y(s) > b,\ \forall s \geq 0,\ \forall b \geq 0. \tag{7}$$

Equation 7 captures only positive buffer underflow. A negative buffer underflow means there is excess items in the buffer, so strictly speaking, there is no underflow. Also, note that Equation 7 holds for a given output stream y. Once again, we want to compute the lower bound for this output stream y.

The lower and upper bounds on the number of incoming objects (α_l and α_u) and the number of objects that are guaranteed to be processed over any time interval (γ_u and γ_l) were experimentally determined. Let y_m be the minimum number of stream objects that are produced in the interval $[0,t]$. This lower bound can be computed as follows:

$$y_m(t) \geq (\alpha_l \otimes \beta^l)(\Delta),\ \forall t \geq 0,\ \forall \Delta \geq 0. \tag{8}$$

Computing the buffer underflow (Equation 7) using the lower bound on the output stream (Equation 8) yields the following condition:

$$C(s,d) - y_m(s) \geq C(s,d) - y(s),\ 0 \leq s \leq t. \tag{9}$$

From Equation 8 and Equation 7, we write,

$$C(s,d) - y_m(s) > b,\ \forall b \geq 0,\ 0 \leq s \leq t. \tag{10}$$

We consider buffer underflows to be random events. This is true if, for example, the processor cycles per second allocated to the decoding task is probabilistic (due to other tasks consuming random cycles). Indeed, in our model, we assumes that the processor cycles allocated to the decoding task is random. However the incoming rate of stream objects and their execution requirement are kept deterministic.

Note f_{max} the maximum processor cycles per second and let F be a random variable giving the usable frequency, that is the number of processor cycles allocated to the decoding task. Then the cumulative distribution of the usable frequencies is given by $P(F = f_i)$, for all f_i, where $0 \leq f_i \leq f_{max}$.

Our goal is to estimate the minimum playout delay required such that the desired stochastic guarantees are met. We write the probability $P(U)$ of a buffer underflow of b objects as

$$P(U) = P(C(s,d) - y_m(s) > b),\ 0 \leq s \leq t_{max}, \tag{11}$$

where t_{max} is the maximum analysis interval.

This probability $P(U)$ defines a finite sample space over any time interval $[0, t_{max}]$, where the buffer underflow is exactly zero or more. Equation 11 represents the subset of the sample space where the buffer underflow is greater than b stream objects.

Buffer underflows depends on the usable frequency which, in our model, can vary randomly. Hence, the probability of having more than b objects underflow is given by

$$P(U) = \sum_{f=0}^{f_{max}} \Big\{ P(U|F = f_i) * P(F = f_i) \Big\}. \tag{12}$$

On the other hand, we can bound the buffer underflow probability using a *stochastic bounding function* g as,

$$P(U) \leq g(b). \tag{13}$$

The bounding function g [3] can be interpreted as a "quality function" which embodies the acceptable loss in the quality of the displayed media stream, usually using sound and video perception models.

Problem Statement: We want to compute the minimum delay d such that the desired output quality specified using the function g is guaranteed. The input parameters required to estimate the initial delay are:

- The usable frequency cumulative distribution $P\{F = f_i\}$, for all $0 \leq f_i \leq f_{max}$,
- The consumption rate c of the output device,

[3] $g(b)$ is a strictly decreasing function. This is because, as mentioned before, the total sample space in Equation 11 is of all events where the buffer underflow is greater or exactly equal to zero. The sample space denoted by Equation 11 for the value $b = 0$ is equal to the total sample space. Therefore, $g(0) = 1$. For increasing value of b the sample space noted by Equation 11 is a decreasing and a sub-set of the sample space lower than b.

- The probabilistic bounding function $g(b)$,
- The lower and upper bounds (α_l and α_u) on the rate of incoming stream objects,
- The maximum analysis interval t_{max}.

4. MINIMUM PLAYOUT DELAY

4.1 Using integer linear programming

Let the function u represents the buffer underflow at time instance t for the output stream y_m with an initial delay d, that is,

$$u(t, d, y_m) = C(t, d) - y_m(t).$$

Naturally, we know the actual u function when $d = 0$ from our system modeling. We could perform more simulations with increasing delay values but that would be extremely time consuming. Instead, we can use techniques from integer linear programming (ILP) to find the minimum delay.

For a usable frequency f_i and an initial delay d, the probability that the buffer underflows by more than b stream objects in the time interval $[0, t]$ is given by

$$h_i(t, b) = P\left(u(t, d, y_m) > b \mid F = f_i\right),$$

with $f_i \in [f_0 \,..\, f_{max}]$ and $t \in [t_0 \,..\, t_{max}]$. As the usable frequency increases, the buffer underflow reduces and hence $h_i(t, b) \leq h_j(t, b)$ if and only if $i \geq j$.

The problem now comes down to finding the h_i that maximizes the objective function $D(t, b)$ given by:

$$D(t, b) = \sum_{i=0}^{max} h_i(t, b) * P(F = f_i), \tag{14}$$

subject to the constraint:

$$\sum_{i=0}^{max} \left(h_i(t, b) * P(F = f_i)\right) \leq g(b), \tag{15}$$

for all $b \geq 0$ and all $t \in [t_0 \,..\, t_{max}]$. This is carried out using a standard integer linear programming solver.

We can now deduce the minimum delay value d_i corresponding to each frequency f_i from the $h_i(t, b)$ obtained, by adding a constant value to the known $u(t, 0, y_m)$ function (essentially shifting it up or down) until we obtain the appropriate $h_i(t, b)$ proportion.

Consider now a set T_i containing time intervals for which the buffer underflow u is greater than or equal to zero. Let $T^+_{i,b}$ (respectively $T^-_{i,b}$) contain time intervals corresponding to an underflow greater than (respectively lower than) b, that is,

$$\begin{aligned} T^+_{i,b} &= \{t \in T_i : u\left(t, d, y_m(t)\right) > b\}, \\ T^-_{i,b} &= \{t \in T_i : u\left(t, d, y_m(t)\right) < b\}. \end{aligned}$$

We can derive bounds on the initial delay so that the probabilistic conditions defined for the sets T^+_b and T^-_b are satisfied. Let $t^+_{i,b}$ such that $u(t^+_{i,b}, d, y_m) = \inf\{T^+_{i,b}\}$ and $t^-_{i,b}$ such that $u(t^-_{i,b}, d, y_m) = \sup\{T^-_{i,b}\}$ as shown in Fig. 4.1. A first bound $d^+_{i,b}$ on the initial delay d_i for a usable frequency f_i is given by

$$d^+_{i,b} = \left(t^+_{i,b} - \frac{y_m(t^+_{i,b}) - b}{c}\right).$$

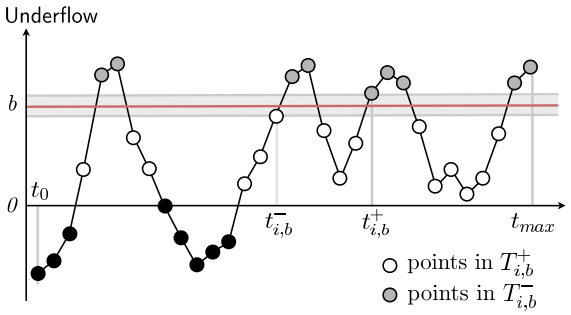

Figure 5: Finding the minimum delay knowing the $h_i(t, b)$ probabilities

Similarly, the second bound $d^-_{i,b}$ on the delay is given by

$$d^-_{i,b} = \left(t^-_{i,b} - \frac{y_m(t^-_{i,b}) - b}{c}\right).$$

Finally the minimum delay d_m satisfying Equation 12 is computed as,

$$d_m = \max_{i,b}\left\{d^-_{i,b}, d^+_{i,b}\right\}$$

4.2 Pure probabilistic approach

We propose an alternative solution to Equation 12 for the minimum delay. Let X be the random variable on $[0, t_{max}]$ such that,

$$X(t) = C(t, d_m) - y_m(t), \forall t \in [0, t_{max}].$$

The random variable X depends on the function C which has a conditional definition. To make computations simpler, we consider the alternative random variable Y defined by

$$Y(t) = ct - cd_m - y_m(t).$$

It is easy to see that:

$$P(X(t) > b) = P(Y(t) > b),\ \forall t \in [0, t_{max}]. \tag{16}$$

Using Equation 16, we can rewrite Equation 12 as,

$$P(Y(t) > b) \leq g(b),$$

for all $b \geq 0$. This new equation has the advantage to have a non-conditional definition. The next step is to transform $P(Y > b)$ in a more convenient form so that we will get an equation without probability and equivalent to equation 11.

Let us now divide the interval $[0, t_{max}]$ into disjoint time intervals, that is, $[t_j, t_{j+1}[\subset [0, t_{max}]$. Then the probability that the random interval chosen corresponds to a buffer underflow greater than b is written as:

$$P_{j,b} = P\Big(\big(Y(t) > b\big) \cap \big(t \in [t_j, t_{j+1}[\big)\Big)$$

and consequently $P(Y(t) > b)$ becomes

$$P\left(Y(t) > b\right) = \sum_j P_{j,b}$$

If the time intervals are small enough, we can discretize y_m and note $y_m(t) = y_j$ for all $t \in [t_j, t_{j+1}[$. Note Y_j the random variable such that

$$Y_j(t) = c\left(t - d_m\right) - y_j,\ \forall t \in [0, t_{max}]. \tag{17}$$

Equation 17 can thus be rewritten using Y_j since

$$P_{j,b} = P\Big((Y_j > b) \cap \big(t \in [t_j, t_{j+1}[\big)\Big)$$

Using Equation 17, the constraint on the time interval t can be written as,

$$t > \frac{b + cd_m + y_j}{c}, \forall t \in [0, t_{max}]$$

Let $A = (b + cd_m + y_j)/c$. Note that the random variable Y is uniformly distributed on interval $[0, t_{max}]$. Hence applying standard rules of probability of randomly picking up an interval in $[0, t_{max}]$, where all time intervals are uniformly distributed, we get:

$$P_{j,b} = \begin{cases} (t_{j+1} - t_j)/t & \text{if } A \leq t_j \\ (t_{j+1} - A)/t & \text{if } t_j < A < t_{j+1} \\ 0 & \text{if } A \geq t_{j+1} \end{cases}$$

This looks quite difficult to handle but it can be rewritten in terms of min and max functions as:

$$P_{j,b} = \frac{t_{j+1} - \min(\max(A, t_j), t_{j+1})}{t} \tag{18}$$

and since

$$\begin{aligned} \max(x, y) &= \frac{x+y}{2} + \left|\frac{x-y}{2}\right| \\ \min(x, y) &= \frac{x+y}{2} - \left|\frac{x-y}{2}\right| \\ |x| &= \sqrt{x^2} \end{aligned}$$

it can be rewritten as a closed form. Indeed, let

$$D_j = \frac{A_j + t_j}{2} + \sqrt{\left(\frac{A_j - t_j}{2}\right)^2},$$

we can rewrite $P_{j,b}$ as

$$P_{j,b} = \frac{1}{t_{j+1} - t_j}\left(v - \frac{D_j + t_{j+1}}{2} - \sqrt{\left(\frac{D_j - t_{j+1}}{2}\right)^2}\right).$$

Now that we have a closed form solution for $P_{j,b}$, we can iterate over all time intervals, and write Equation 12 as,

$$\sum_j P_{j,b} \leq g(b). \tag{19}$$

for all $b \geq 0$. We have successfully transformed the problem statement into a form from which we can deduce the delay. Indeed, from Equation 19, we can easily numerically compute the minimum initial delay for a given underflow b. It is actually an infimum of all delays corresponding to various time intervals. We denote this delay for a fixed b as d_m^b. Taking the greater of all d_m^b delays for all possible underflows b then gives the minimum initial delay we are looking for.

5. RESULTS

In the following section we present a method to numerically compute the minimal delay while satisfying Equation 12 for any given stochastic bounding function g. Later we present in detail the results for the two approaches.

5.1 Numerical Evaluation

In this section, we describe the two simulation set-ups: (1) SystemC simulation for validation of analytical model results, and (2) MATLAB implementation of our analytical model. There are two main observations from the SystemC and MATLAB experiments: (1) the delay value reduces as the maximum buffer underflow increases. Consequently, the buffer size reduces, and (2) the minimum playout delay estimated using the analytical model corresponding to maximum buffer underflow value is accurate with respect to the mathematical model.

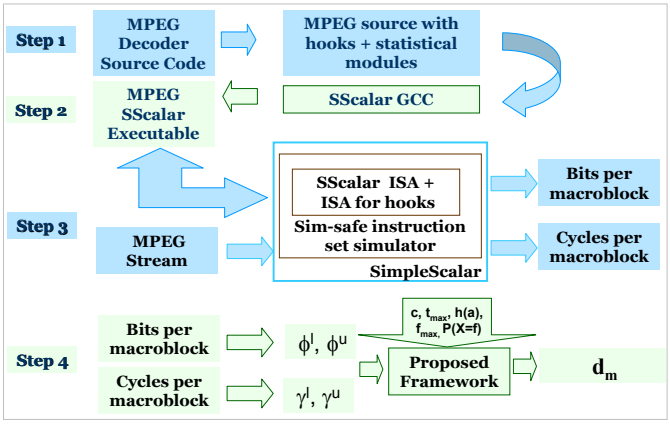

Figure 6: Simulation setup.

We now describe the simulation set-up used to compute the input parameters required for the mathematical model.

Simulation Set-up: We modeled our processor using the *sim-profile* configuration of the SimpleScalar instruction set simulator [2]. Our media-processing task was an MPEG-2 decoder [7], whose source code was annotated with *start* and *stop* counters to record the number of processor cycles consumed by each stream object. To characterize the execution requirement of the decoder, we used a set of video clips (obtained from [18]) having an average bit rate of 1500 kbps and a resolution of 352×240 pixels. The display rate of these clips was 30 fps.

The procedure followed to obtain these functions is shown in Figure 6. To implement the input models we used MATLAB [9]. Recall that ϕ characterizes the variability in the number of bits constituting each macroblock in the compressed video stream, α characterizes the variability in the arrival pattern of the video stream at the buffer b and γ captures the variability in the execution requirement of each macroblock. Clearly, such a characterization is more expressive than the traditionally used best/worst bounds which are overly optimistic/pessimistic.

We now estimate the minimum playout delay required. We show that the minimum playout delay required is such that the stochastic boundary function $g(b)$ is greater than the probability that the buffer underflows ($P(U > b)$). As mentioned in the problem statement in Section 3, the processor frequency distribution is an input function to our mathematical model. This minimum playout delay is smaller than the delay that would be otherwise required if the buffer never underflows. In Figure 7, the probability that the buffer underflows is plotted. The probability curves are shown for the stochastic bounding function and the probability that the

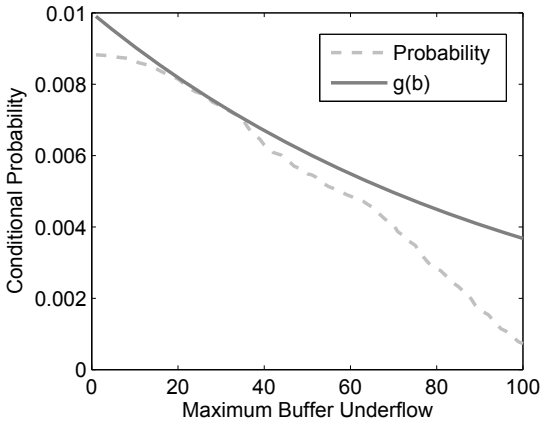

Figure 7: Meeting desired stochastic constraints. The probability that the playout buffer underflows is no more than the stochastic bounding function.

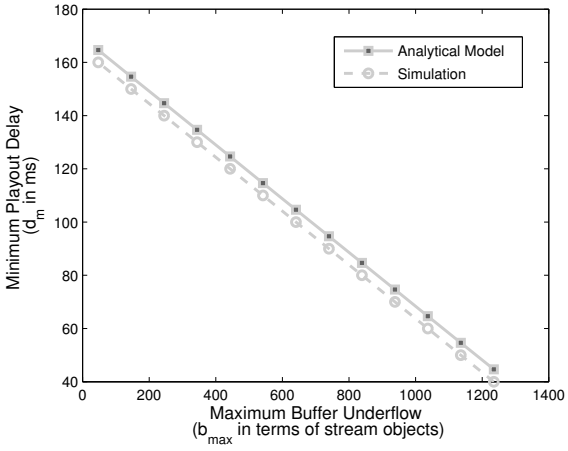

Figure 8: Accuracy of analytical model. Minimum playout delay estimated using mathematical model is close to the delay values obtained from simulation.

buffer underflows. Using the mathematical model we found that for a delay value 151.4ms, the stochastic bounding function is greater than the probability the buffer underflows b stream objects (b varies from 0 to 406).

For simplicity, the $g(b)$ function used in our experiments is,

$$g(b) = \frac{1}{b_{max}} * \exp\left(\frac{-b}{b_{max}}\right),\ 0 \leq b \leq b_{max}. \qquad (20)$$

A system designer would probably use a more sophisticated function for $g(b)$ taking into account visual and audio perception models.

Using the exhaustive search method we were able to find the minimum playout delay required.

The simulation set-up was used to validate our results obtained using our mathematical model. Figure 6 shows the simulation set-up. We fixed the maximum buffer underflow and estimated the minimum playout delay required using the mathematical model, and from the SystemC simulation. The SystemC set-up requires an iterative procedure to check if the stochastic bounding function is satisfied, for each delay value. An exhaustive search using our mathematical framework determines the minimum playout delay value. In simulation using SystemC and with the analytical framework we fixed b to the maximum value (Figure 8 shows comparison of delay values obtained and estimated).

5.2 Delay Estimation

In this sub-section, we present results for the minimum playout delay solutions presented in Section 4. First we show evidence for the correctness of the results obtained from the closed-form solutions w.r.t to results obtained from numerical evaluation. Second, we present results for both the ILP and the pure-probabilistic approach in an attempt to find the minimum playout delay.

Figure 9 shows the minimum playout delay values estimated for various maximum buffer underflows using the ILP based approach. Our objective is to show that the ILP based approach provides an upper bound on the minimum playout delay compared to the numerical evaluation. (Results shown in Figure 8). We set the maximum playout buffer underflow and found the corresponding minimum playout delay required using the ILP approach. That is, the $g(b_{max}) = 0$ (unlike the exponential decay function [4]). Clearly, Figure 9 shows that the delay values we obtain from ILP are greater than the delay values we obtain using numerical evaluation for similar buffer underflows.

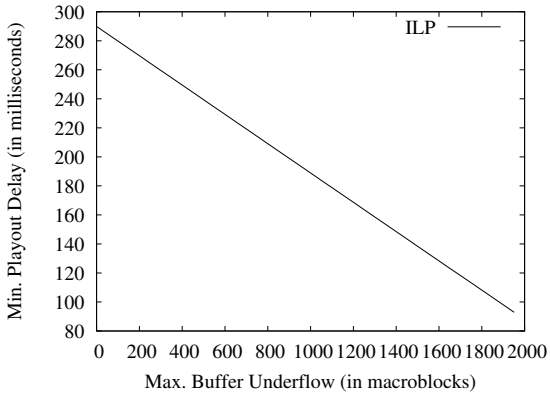

Figure 9: Minimum playout delay values for fixed maximum buffer underflow.

Figure 10 shows the minimum playout delay estimated using ILP and a pure probabilistic approach. The desired quality constraints are met as the ratio of the buffer underflow probability ($P(U > b)$ over the stochastic bounding function ($g(b)$) approaches one. We now observe and explain the result shown in Figure 10:

- *Minimum Playout Delay:* There are more values corresponding to a ratio around 1 in both solutions. The reason being that we have used the exponential decay function for this experiment. Unlike for the results presented in Figure 9, $g(b_{max}) \neq 0$ for the exponen-

[4]Note that the results obtained in Figure 8 too were obtained using $g(b_{max}) = 0$.

tial decay function [5]. Therefore, for both solutions we cannot have a ratio greater than unity. Thus we find more solutions for the minimum playout delay value. If we consider the first delay value to approach unity, then for the ILP based approach the answer is 222.22 milliseconds and for pure probabilistic approach it is 131.31 milliseconds. As mentioned before, ILP gives us a safe upper bound. Use of a different stochastic bounding function would give us a single minimum delay value for both the solutions.

- *ILP drop:* Note in Figure 10, for ILP, the ratio suddenly drops to zero. The probability that the buffer underflow is greater than b is the sum of the product of the conditional probabilities with the probability distribution of frequencies. This summation probability has been optimized by ILP to achieve the upper bound which is $g(b)$. Hence, when there is a non-zero underflow b, the value of $P(u > b)$ will be close to $g(b)$, and this ratio drops rapidly to 0 as the buffer underflow becomes zero.

- *Approximations:* We see that the delay values estimated for the pure probabilistic approach is lower than the ILP. The reason for the difference in the delay value estimated is due to two approximations: (1) We first randomly generate processor frequency for each time interval required such that it fits the distribution $P(F = fi)$. This random set gives the processor cycles available over time intervals we are interested in. Then we compute $\beta(\Delta)$ using the randomly generated processor frequencies. After which we compute the lower bound y_m for a given frequency profile (which is generated from a frequency distribution). Therefore the delay value from the pure probabilistic approach is the minimum delay value for a given frequency profile; and (2) the second approximation is due to the discretization of the output function.

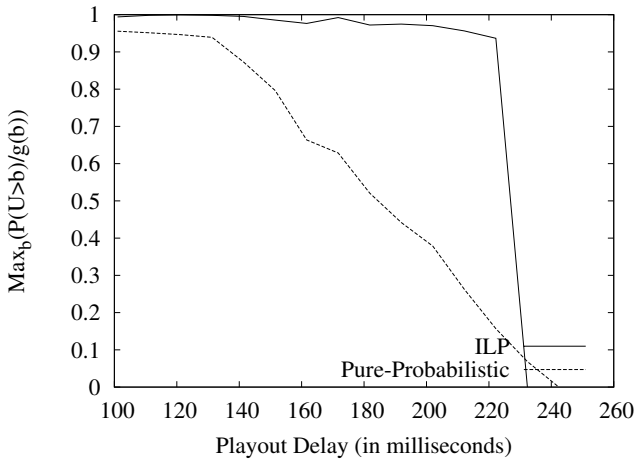

Figure 10: Playout delay solutions for meeting stochastic constraints.

[5]We set the $b_{max} = 406$ in Equation 20

6. RELATED WORK

We present previous work published under two research topics in system design for multimedia: (1) analytical framework, and (2) techniques for reducing on-chip memory.

6.1 Analytical Models

In this section, we will look at the initial classification in methods for SoC design and discuss the pros and cons of the approaches. Existing approaches for SoC design can be broadly classified as follows: (1) analytical models, and (2) simulation. The main disadvantage of simulation based techniques is that they are slow for any design that involves a large number of iterations (for example, designs that involves identifying several design parameters). Moreover, simulation techniques do not provide any special insights that can lead to resource savings.

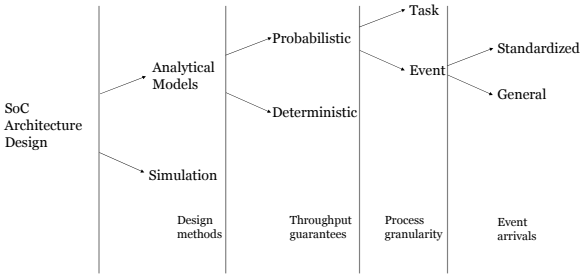

Figure 11: Dimensions of SoC Design.

Figure 11 sketches various existing design methodologies. System-level design using mathematical framework involves fast exploration of design parameters. This paper shows that there could be valuable insights obtained from such analysis. Clearly, guarantees on throughput could be formulated in mathematical frameworks. With understanding of the benefits of mathematical models, we now look at the second-level of classification in performance modeling.

In general, analytical models can be divided as deterministic and probabilistic frameworks. Mostly, deterministic models are for worst-case analysis of the systems. The worst-case analysis is suitable for hard real-time systems. For soft-real time systems, however, a probabilistic framework is appropriate. The existing probabilistic mathematical frameworks only analyze average-case scenarios or most-probable scenarios. We have seen so far two levels of classification for methods for SoC design: (1) simulation based or analytical models, and (2) mathematical modeling. Now we look at the third level.

The granularity of the application, more precisely, if they are modeled as tasks or events is the basis of this classification. Further, if they are events they could be again divided into techniques that model standardized or general events. Here we discuss in detail some of the popular analytical models. The reader will be able to map these models with the classification discussed. Analytical methods that have gained attention are: (1) synchronous data flow graphs [17], (2) stochastic automata networks [21], and (3) event adaptation functions [15].

The mathematical model we propose differ from other existing models in the following ways:

- The arrival of items in our model is not limited to standard input models such as periodic, Poisson, and so on.
- Our model captures the variability in the processing of data items such that this variable nature of the media stream could itself be exploited for efficient system design.
- The analytical framework that we present can be applied to any level of granularity i.e., each data item in the stream can be a bit, a macroblock, or a frame.
- In contrast to average-case analysis of the existing probabilistic models, we show that guarantees on output requirement could be still provided with our stochastic mathematical framework.
- Our framework can be applied to any kind of multimedia streaming applications. In other words, it does not rely on the specific characteristics of the application.

6.2 Reducing Buffer Memory

Previous efforts [10, 16, 17] have specifically been directed towards optimizing on-chip memory in system-on-chip architectures designed for embedded multimedia systems. Most of the previous papers attempted to reduce memory requirements of synchronous data-flow (SDF) graphs which are used for specifying compute-intensive kernels of DSP applications.

Murthy and Bhattacharya [10] proposed buffer merging to reduce memory requirements of SDF graphs. Buffer merging is achieved through sharing buffers that two different processes use. After analyzing the lifetime of actors (nodes specifying application code blocks in SDFs), it is determined whether two different processes containing these actors can potentially share buffers.

Similar approaches have been followed in the domain of computer networks to counter the burst in network traffic so as to effectively utilize network resources. In comparison, our work is concerned with fixed playout delay, rather than dynamically adjusting it at run time [14]. Further our technique is more relevant in the context of playing stored audio and video. Hence, we did not exploit network related parameters such as loss and delay.

7. DISCUSSION

In this section, we answer some potential questions on this work. The first subsection will introduce the reader an approach for formulating QoS using our framework. Second, we will present evidence for use of the arrival and service curves for modeling data flows present in media applications. Also, we point to literature studies which use this basic framework presented in this paper to model multiprocessor and multi-task architecture set-ups. The last subsection discusses two points: (1) reasons for the choice of the stochastic bounding function used in our numerical evaluation; (2) how our two approaches for estimating the delay can be effectively utilized.

7.1 QoS Constraints

In this subsection, we discuss how the designer could specify the acceptable, tolerable loss in the quality of the video display using statements such as: Case 1- the buffer should never underflow more than two consecutive frames in displaying 30 frames sequentially; Case 2- the buffer should never underflow more that 17 frames (in total) in displaying 100 frames sequentially. These type of constraints could be written as (for Case 1): (In the below equation, p represents the number of time units that have elapsed.)

$$P\{C(s,d) - y_m(s) > b\} \leq g(b), \tag{21}$$

where $0 \leq s \leq t_{max}$, $s = \frac{2}{30} * p$, $\forall 0 \leq p \leq \frac{t_{max}*30}{2}$).

The time instance s corresponds to the point where 30 frames should be displayed (we know in this case 30 frames takes one second. So, when $p = 1$, we have $s = \frac{2}{30}$ and at that point in time it is tolerable to have buffer underflow of at most 2 frames in 30 frames. The b corresponds to stream objects of 2 frames and $g(b) = 0$. $g(b)$ is a stochastic bounding function, which defines the probability that the buffer underflow is no more than b stream objects.

Hence, in Case 1, if b corresponds to stream objects for 2 frames and for every s (i.e. at points where 30 frames ideally should be produced), $g(b) = 0$. The probability that the buffer underflows more than b stream objects (corresponding to 2 frames) is zero. Similarly, Case 2, could be denoted and it would be a variation in Equation 21 in terms of defining b, s, and p. For generality, that is for all $b \geq 0$, we consider $g(b)$ gives the upper bound on the desired probability.

7.2 System Model Parameters

To better understand our framework, we opted for a simple architecture and single application media stream in our model and analysis; there is no limitation whatsoever to model a complex architecture or to model multiple streams.

Framework Extension: The deterministic framework, which we extend in this paper to a probabilistic set-up, has been used for analysis for the following:

- a SoC with multiple processors in pipeline [13],
- multiple applications running concurrently in a processor [12], and
- a data flow with a fork/join operation between the application blocks [4].

If the buffers are not being shared, then the framework we propose in this paper can handle multiple input streams too.

7.3 Bounding Function and Delay Solutions

$g(b)$: The reason we chose the stochastic bounding function as an exponential decay is to have a simple function. The solutions we provided for the delay values accept this $g(b)$ function as an input, so, the user can plug in any function he desires. In the previous section of this paper we suggested an approach to formulate the QoS issues independent of this stochastic bounding function. We envision, however, the general stochastic bounding function could be used in most of the design cases to model buffer underflow.

Two delay solutions: Recall that we presented the ILP and the pure-probabilistic as approaches to estimate the delay values. For ILP, the advantage is that the delay value estimated is a safe upper bound; the disadvantage is that the delay value chosen could lead to an inefficient design especially w.r.t to the buffer size. For the pure-probabilistic approach, the advantage is that the delay value estimated is not a pessimistic bound; the disadvantage is that certain

approximations are carried out in the estimation of delay values. The result we presented in the paper shows what the designer can expect from the ILP solution versus the pure probabilistic approach; the lower most bound and delay for a (or more) specific frequency profile respectively.

8. CONCLUSION

Multimedia players do not require hard real-time guarantees; soft-real time guarantees are sufficient. Towards this, we presented a framework with probabilistic constraints on the output. An initial playout delay is chosen to satisfy the stochastic constraints. Using this playout delay, the required playout buffer is reduced to a large extent, leading to enormous savings in terms of memory used.

A potential future direction would be to estimate the minimum playout delay considering the sources causing buffer underflow as a probabilistic event. For instance, the arrival and the execution requirement of stream objects could be probabilistic. Such an effort would lead to general stochastic framework for designing system-on-chip for multimedia applications.

This paper shows that stochastic guarantees are in fact the constraints to be used when estimating processing and memory requirements for processing multimedia.

9. REFERENCES

[1] R. T. Apteker, J. A. Fisher, V. S. Kisimov, and H. Neishlos. Video acceptability and frame rate. *IEEE MultiMedia*, 2(3):32–40, Spring 1995.

[2] T. Austin, E. Larson, and D. Ernst. SimpleScalar: An infrastructure for computer system modeling. *IEEE Computer*, 35(2):59–67, Feburary 2002.

[3] J.-Y. L. Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*. Springer-Verlag, New York, 2001.

[4] K. Huang and L. 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.

[5] Y. Jiang. A basic stochastic network calculus. In *Proceedings of the ACM international conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, pages 123–134, September 2006.

[6] A. Kahng, I. Chayut, J. Cohn, T. Hattori, J.-T. Kong, P. Paulin, and R. Tobias. Roundtable: Design and CAD challenges for leading-edge multimedia designs. *Design Test of Computers, IEEE*, 24(1):83 –93, 2007.

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

[8] Y. Liu, A. Maxianguine, S. Chakraborty, and W. T. Ooi. Processor frequency selection for SoC platforms for multimedia applications. In *Proceedings of the IEEE Real Time Systems Symposium (RTSS)*, pages 336–345, December 2004.

[9] Mathworks. Matlab 7.2. http://www.mathworks.com/products/matlab/, 2007.

[10] P. K. Murthy and S. S. Bhattacharyya. Buffer merging- A powerful technique for reducing memory requirements of synchronous dataflow specifications. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 9(2):212–237, April 2004.

[11] A. Nandi and R. Marculescu. System-level power/performance analysis for embedded systems design. In *Proceedings of the annual conference on Design automation (DAC)*, pages 599–604, June 2001.

[12] B. Raman and S. Chakraborty. Application-specific workload shaping in multimedia-enabled personal mobile devices. *ACM Transactions on Embedded Computing Systems*, 7(2):10, Feburary 2008.

[13] B. Raman, S. Chakraborty, W. T. Ooi, and S. 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.

[14] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. Adaptive playout mechanism for packetized audio applications in wide area networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 680–688, June 1998.

[15] K. Richter and R. Ernst. Event model interfaces for heterogenous systems analysis. In *Proceedings of the IEEE International Conference on Design Automation and Test in Europe(DATE)*, pages 506–513, March 2002.

[16] N. Sarshar and X. Wu. Buffer size reduction through buffer sharing for streaming applications. In *IEEE International conference on Multimedia and Expo (ICME)*, pages 1635–1638, Taipei, Taiwan, June 2004.

[17] S. Stuijk, M. Geilen, and T. Basten. Exploring trade-offs in buffer requirements and throughput constraints for synchronous dataflow graphs. In *In Proceedings of the ACM Annual conference on Design automation (DAC)*, pages 899–904, San Francisco, CA, April 2006.

[18] Tektronix. MPEG elementary streams. ftp://ftp.tek.com/tv/test/streams/Element/index.html, 1996.

[19] E. Wandele and L. Thiele. Abstracting functionality for modular performance analysis of hard real-time systems. In *Proceedings of the conference on Asia South Pacific design automation (ASP-DAC)*, pages 697–702, January 2005.

[20] D. Wijesekera, J. Srivastava, A. Nerode, and M. Foresti. Experimental evaluation of loss perception in continuous media. *Multimedia Systems*, 7(6):486–499, November 1999.

[21] N. H. Zamora, X. Hu, and R. Marculescu. System-level performance/power analysis for platform-based design of multimedia applications. *ACM Transactions on Design Automation of Electronic Systems*, 12(1):2, 2007.