On-Chip Traffic Modeling and Synthesis for MPEG-2 Video Applications

Girish V. Varatkar and Radu Marculescu, *Member, IEEE*

***Abstract*—The objective of this paper is to introduce self-similarity as a fundamental property exhibited by the bursty traffic between on-chip modules in typical MPEG-2 video applications. Statistical tests performed on relevant traces extracted from common video clips establish unequivocally the existence of self-similarity in video traffic. Using a generic tile-based communication architecture, we discuss the implications of our findings on on-chip buffer space allocation and present quantitative evaluations for typical video streams. We also describe a technique for synthetically generating traces having statistical properties similar to those obtained from real video clips. Our proposed technique speeds up buffer simulations, allows media system designers to explore architectures rapidly and use large media data benchmarks more efficiently. We believe that our findings open new directions of research with deep implications on some fundamental issues in on-chip networks design for multimedia applications.**

***Index Terms*—Communication analysis, long-range dependence, on-chip networks, self-similarity, system-level design.**

I. Introduction and Objectives

NOWADAYS, people see the need for portable embedded multimedia appliances capable of handling advanced algorithms required for all forms of communication (text, speech, and video). In the design of chips for such appliances, system-level design issues become critical as implementation technology evolves toward increasingly complex integrated circuits and the time-to-market pressure continues relentlessly. As a consequence, it is important to define and optimize a common design "platform," consisting of both hardware and software resources, that could be shared across multiple multimedia applications.

The system-level view of such a generic design "platform" is shown in Fig. 1. As we can see, it consists of both *fixed* processing resources (e.g., ASICs) and *programmable* resources (e.g., processors) that cooperate to run the target application (e.g. MPEG-2 audio/video decoder, e-mail, web browsing, etc.). The overall goal of the system-level design of such a platform, is then to find the best mapping of the target application onto the set of architectural resources while satisfying the imposed design constraints (e.g., minimum area, minimum power dissipation, high performance, etc.). Most notably, the transition from desktop multimedia to portable multimedia based on heterogeneous design platforms brings *concurrency and communication* as prime candidates for system-level analysis and optimization.

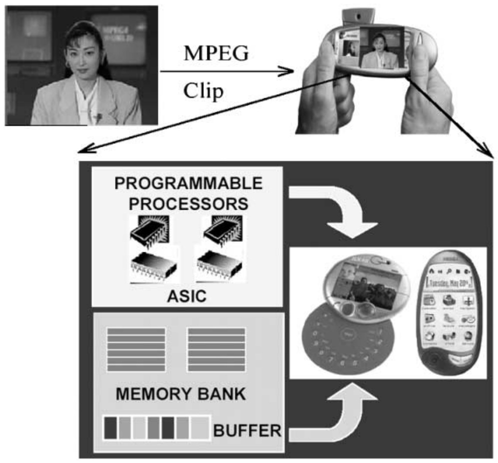


Fig. 1 Generic portable embedded multimedia system.

In this paper, we address the fundamental issue of selecting the optimal communication resources between different on-chip modules [16], [17]. For complex systems composed of heterogeneous components, the on-chip traffic produced among different modules has very diverse characteristics. Since the traffic patterns depend so much on the target application, it is necessary to judiciously allocate the on-chip communication resources, especially since the on-chip buffer space is usually very limited compared to the buffer space in real data networks.

Recently, Dally and Towles [1] proposed a novel on-chip interconnect network [Fig. 2(a)] which can be used instead of the classical *adhoc* global wiring structure. What makes this generic architecture very attractive is that it can offer well-controlled electrical parameters which enables high-performance circuits to reduce latency and increase bandwidth.

As shown in Fig. 2(a), a chip employing such a communication architecture consists of several network *clients* (e.g., processors, memories, and custom logic) which are connected to a network that routes *packets* between them. Each client is placed on a tile and its communication with other clients (not only its neighbors) is realized via the on-chip network. The discussions in [1] propose a mesh-based network in which each tile is connected to the adjacent tiles in the north-south and east-west directions. Another approach in [34] to building a network-on-chip (NOC), proposes a honeycomb shaped structure in which computational resources are present at the nodes

Manuscript received August 27, 2002; revised June 26, 2002. This work was supported in part by the National Science Foundation CAREER Award CCR-0093104, in part by DARPA/Marco Gigascale Research Center (GSRC), and in part by the SRC Award 2001-HJ-898. This paper appeared in part in the Proceedings of the ACM/IEEE Design Automation Conference, New Orleans, LA, June 2002, and in the Proceedings of the IEEE International Conference on Multimedia and Expo, Lausanne, Switzerland, Aug. 2002.

The authors are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890 USA (e-mail: gvv@ece.cmu.edu; radum@ece.cmu.edu).

Digital Object Identifier 10.1109/TVLSI.2003.820523

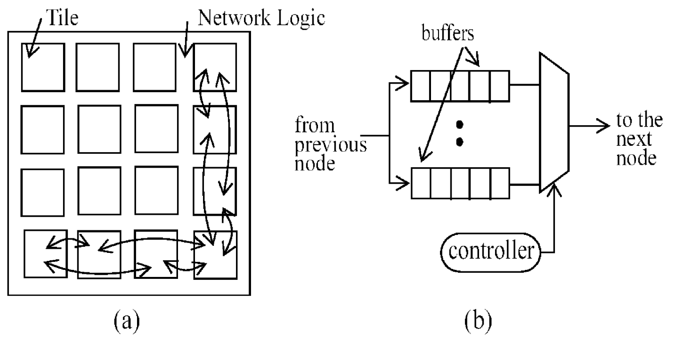


Fig. 2. (a) Die module tiles and network logic. (b) Generic input controller and its buffers.

of the hexagon and a router is present at the center of each hexagon. In both NOC approaches, a *router* is needed and it consists of several input-output controllers and their associated *buffers* [Fig. 2(b)]. From a practical point of view, the success of such architectures depends on the ability to keep the overall area overhead to a *minimum*.[1] Since the area of the router is heavily dominated by the space occupied by the on-chip buffers, the problem of *optimal buffer sizing* becomes an issue of critical importance. Indeed, dropping or misrouting packets because of inappropriate buffer sizing reduces the overall performance and significantly increases the on-chip power dissipation. We also point out, that this severe limitation of the on-chip buffer space comes in deep contrast with real data networks, where there is ample room for very large buffers. This makes the on-chip network design problem quite unique and challenging. As such, this paper has two major objectives.

- First, present a novel traffic analysis approach as a precise way to characterize the on-chip communication traffic pattern of different multimedia applications. More precisely, we propose a technique for traffic modeling based on *self-similar* or *long-range dependent* (denoted as LRD[2]) stochastic processes. By analyzing the statistical properties of the arrival process at different points in a generic architecture like the one in Fig. 2, for standard MPEG-2 applications, we characterize *quantitatively* the degree of self-similarity of the on-chip traffic using techniques based on the Hurst parameter [3]. Knowing the Hurst parameter helps the designer to choose the *minimal* buffer size for the router at each tile in Fig. 2(a) which, in turn ensures a certain *quality of service* (QoS) for running the multimedia application.
- Second, describe an algorithm for synthetically generating self-similar sample traces of different lengths. If we choose to generate synthetic traces shorter than the original trace, then these synthetic traces can be used for simulating video traffic and estimating the buffer-loss probabilities and the delay experienced by a macroblock at the buffer in a very short time compared to the full length simulation. This way, we can dramatically speed up the simulation of a buffer and estimate the buffer-loss probability as well as the macroblock delay.

[1]For instance, the authors in [1] suggest about a 6% area overhead of network logic for each tile.

[2]We use interchangeably the concepts of self-similarity and long-range dependence.

The analysis we propose here is especially relevant to the large class of *portable embedded multimedia systems* where the QoS requirements vary considerably from one media to another (e.g., video connections require consistently high throughput, but tolerate reasonable levels of jitter or packet errors) and buffer space is very limited. Consequently, the ability to explore several communication schemes while trying to satisfy QoS requirements is of crucial importance. As we show later in the paper, making use of the knowledge of the traffic pattern for achieving a certain QoS with optimal resources turns out to be extremely helpful.

A. Contributions of the Paper

The contributions of this paper are twofold.

- First, we provide evidence about the presence of self-similar phenomena in on-chip traffic generated at *macroblock level* by multimedia applications. This has very important consequences since self-similar processes have properties which are completely different from traditional *short-range dependent* autoregressive moving average (ARMA) or Markovian processes that have been mostly used in system-level analyses so far [15], [18]–[20]. We also point out that the traditional video traffic modeling has concentrated at the *frame-level* traffic analysis (e.g. [36]), while we look at the *macroblock level* on-chip traffic. As we will see, this has important consequences on chip design. More precisely, knowing the Hurst parameter which characterizes the traffic pattern for a particular application helps in finding the optimal buffer length distribution which turns out to be the critical issue for the routers at each node in the on-chip communication network.
- Second, we describe a method of generating *synthetic traces* with statistical properties similar to the original ones. We assess the performance of this synthetic trace generation method by comparing the bit-loss probability at a generic buffer obtained by simulating a synthetic trace and comparing it with that of the real trace. These synthetic traces can be used to dramatically speedup the simulation process for multimedia applications where tens of hours of simulation time is typically required to gather useful information for on-chip network design.

Taken together, our proposed technique allows media systems designers implementing on-chip communication networks to choose the appropriate on-chip buffer sizes and use large multimedia data benchmarks more effectively. Ultimately, this will enable systems designers to optimally tradeoff performance metrics and media quality.

B. Related Work

In recent years, due to the advent of SoCs, the issue of efficient communication schemes—at system-level—received increased attention [21]–[23]. In particular, people have explored alternative solutions to the standard bus-based communication, especially for portable devices where tight power and performance constraints make the bus-based communication solution less attractive [24], [25], [28].

One problem with the approaches proposed so far for on-chip network exploration, is that they rely entirely on explicit simulation. Consequently, due to the huge amount of data contained in multimedia applications, the simulation-based techniques tend to become prohibitively expensive in practice [14], [21], [27]. Typically, tens of hours of CPU time are needed to simulate just a few minutes of video data. Moreover, simulating video data randomly, without a precise (quantitative) measure of the traffic characteristics, is dangerous since the actual implications of traffic on the overall system performance may be completely obscured by using inappropriate data. These major issues prompted our attention toward developing a more *formal* approach for on-chip communication analysis with emphasis on precise characterization of multimedia traffic and using robust technique for simulation of synthetically generated video sequences so as to reduce the simulation time while maintaining the accuracy of the estimates of the delay and the bit-loss probability.

Our initial effort has concentrated on the literature available for real data networks, trying to see what from that sizable body of knowledge can be ported to the on-chip network design. Another question was to determine what is specific to the on-chip network design as opposed to macro-networks design. Trying to answer these questions, we realized several things. First, the landscape of networking research has changed dramatically over the last decade. In their pioneering study on self-similarity, Leland *et al.* [2] started a new research direction in traffic modeling and network-performance analysis having the long-range dependence as the central concept. Since then, significant other research efforts have been aimed at explaining the nature and the practical applications of this complex phenomenon [5], [26]. Second, the classical paradigm that data traffic can be adequately modeled as Markovian sources of information has been also reconsidered [29], [36]. The most widely studied nonMarkovian LRD process is the fractional Gaussian noise (FGN) process [7]. There are several existing methods for synthesizing FGN processes [29], [30] but each method has some advantages and some drawbacks.

Our paper is an attempt to bridge conceptually two very different worlds: data networks and on-chip networks. To this end, we first identify at chip-level a phenomenon discussed so far only in the context of traffic for real data networks. Second, we analyze the traffic of a multimedia application which targets a novel packet-based SoC implementation and illustrate the impact of our analysis on on-chip network design. Finally, we hope that beyond its practical implications, the connection that we create between these apparently so different domains will stimulate further research on formal methods for on-chip network design.

C. Organization of the Paper

Section II defines the self-similarity and describes the statistical methods used to establish the presence of LRD. Section III describes the method for generating synthetic LRD traces. In Section IV, we present a detailed analysis of traffic for the MPEG-2 video decoder and show the results for different video clips. In Section V, we illustrate the implications of LRD on the on-chip network design and also show an example of how to speed up the buffer simulation using synthetic traces to estimate buffer-loss probability. Finally, we conclude by summarizing our main contribution.

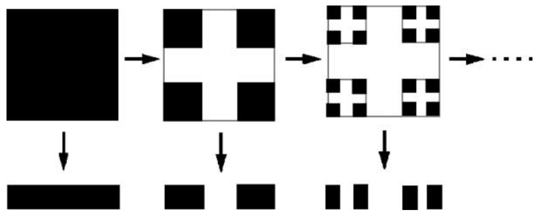


Fig. 3. Deterministic fractal example: 2-D cantor set and its projection on a horizontal time axis giving 1-D cantor set.

II. Basics on Self-Similarity

Self-similarity and fractals are natural concepts discussed in detail in [11]. Simply speaking, an object is self-similar (or fractal), if its parts, when magnified at different scales, resemble the shape of the whole. For instance, if the object we consider is a natural image or a time series, then self-similarity would imply that the spatio-temporal properties of the object remain preserved with respect to *scaling* in space and/or time. To get more intuition behind it, let us consider now the Willinger and Park's cantor set example, that appears in Chapter 1 in [26]. A two-dimensional (2-D) *deterministic* cantor set is obtained by starting with a black unit square, scaling its size by 1/3, then placing four copies of the scaled square at the four corners, and repeating this process recursively ad infinitum (Fig. 3). The one-dimensional (1-D) cantor set can be obtained by projecting the 2-D cantor set onto a horizontal time axis. This can be further interpreted as an ON/OFF time series which models data traffic. A more detailed discussion can be found in the original text in [26].

Stochastic self-similarity adds to this model a probabilistic behavior. Unlike the deterministic fractals, the objects do not possess the *exact* resemblance of their parts at finer levels of detail. For instance, if we consider the time series which may characterize some real-data traces, it may be possible to expect an *approximate* similarity with respect to the shape of the autocorrelation function. Second-order (or temporal) statistics of the autocorrelation function are the statistical properties that capture burstiness (or variability) in time series which characterize, for instance, traffic patterns in real networks [2], [5]. In particular, the *autocorrelation function*, as a function of the time lag, decreases *polynomially* rather than exponentially. The existence of such nontrivial correlation "at a distance" is referred to as LRD.

In the case of MPEG-2 video traces, we concentrate on traffic characteristics at *macroblock-level*. In an MPEG-2 decoded clip, within each frame of a video, there are certain objects. All the macroblocks within a single object carry similar amounts of information and, hence, they get coded using almost the same number of bits. Since the macroblocks within an object generally occur next to each other in a frame, this leads to the appearance of long range correlations. This may be the intuitive reason behind observing the LRD phenomenon in video traffic at *macroblock-level*.

We also note that, from a practical point of view, LRD has a considerable impact on queueing performance of the communication architecture. The traditional *short-range dependent* (or Markovian) processes have an autocorrelation function which decays *exponentially* fast. But the LRD processes exhibit a much slower decay of correlations; that is, their correlation functions typically obey some power-law decay. Intuitively, the presence of LRD indicates that while long-range correlations are individually small, their *cumulative effect* is nonnegligible. This produces scenarios which are drastically different from those experienced with traditional short-range dependent models such as Markovian processes. This is the subtle point where the long-range dependence analysis we propose surpasses classical Markovian analysis and proves its practical value.

A. Definition of Long-Range-Dependence

The mathematical definition of long-range-dependence is given as follows: let $X = (X_t : t = 0, 1, \ldots)$ be a wide-sense stationary stochastic process with mean m, variance σ^2 and autocorrelation function $r(k)$, $k \geq 0$. According to [2], X is said to exhibit *long-range dependence* if

$$r(k) \sim k^{-\beta} L_1(t) \quad \text{as } k \to \infty \tag{1}$$

where $0 < \beta < 1$, $L_1(t)$ is a slowly varying function, that is, $\lim_{t\to\infty} L_1(tx)/L_1(t) = 1$, for all $x > 0$ and $\sim$ denotes the "asymptotically close" condition.

From (1) we see that LRD is characterized by an autocorrelation function that decays hyperbolically rather than exponentially fast. It also implies that the spectral density obeys a *power-law* function near the origin (also called $1/f$-noise). This captures the intuition behind LRD, namely that while high-lag correlations are individually small, their *cumulative effect* matters and gives rise to features which are very different from those of short-range dependent processes. In what follows, we describe two methods for testing LRD in any time series X.

B. Variance-Time Analysis

Let X be a wide-sense stationary-time series. For each $m = 1, 2, 3, \ldots$ let $X^{(m)} = X_k^m$: $k = 1, 2, 3, \ldots$ denote the new wide-sense stationary-time series obtained by averaging the original time series X over nonoverlapping blocks of size m. That is, for each $m = 1, 2, 3, \ldots$; X^m is given by $X_k^m = (1/m)(X_{km-m+1} + \ldots + X_{km})$, $k > 0$.

The variances of X^m, $m = 1, 2, 3, \ldots$ for short-range dependent processes will eventually decrease *linearly* in log–log plots against m with a slope equal to -1. On the other hand, for processes with long-range dependence, the variances of the aggregated processes X^m, decrease linearly (for large m) in log–log plots against m with slopes arbitrarily flatter than -1. The variance-time plots are obtained by plotting $\log(\text{var}(X^m))$ against $\log(m)$.

Cox [4] shows how a specification of the sequence ($\text{var}(X^m$: for $m > 0$) is equivalent to a specification of the autocorrelations given by (1). More importantly, for a constant c, we have

$$\text{var}\, X^{(m)} \sim cm^{-\beta} \quad \text{as } m \to \infty \tag{2}$$

with $0 < \beta < 1$. Actually, this value of β is related to the rate at which autocorrelations decay for large values of the lag. From (1), we can see that the autocorrelations decay hyperbolically with decay constant β.

The value of β is also related to the power-law behavior of the spectral density function near the origin for very small values. If the spectral density decays with power α then $\beta = 1 - \alpha$. As an estimate of parameter β, we use the slope of a least square fit line through the points in the plot, ignoring the small values of m which represent the transient period.

C. R/S Analysis

Historically, stochastic processes with long-range dependence are important because they provide an elegant explanation of an empirical law that has been observed in many naturally occurring time series [3]. What has since come to be known as the *Hurst effect* can be described as follows.

Given observations (X_k: $k = 1, 2, \ldots, n$) with sample mean $\bar{X}(n)$ and sample variance $S^2(n)$, the *rescaled adjusted range statistics* (denoted as R/S statistics) is given by

$$\frac{R(n)}{S(n)} = \frac{1}{S(n)} [\text{Max}(0, W_1, W_2, \ldots, W_n) - \min(0, W_1, \ldots, W_n)] \tag{3}$$

where $W_k = (X_1 + X_2 + \ldots + X_k) - k\bar{X}(n)$, $1 \leq k \leq n$. In his study of the rescaled adjusted range [3], Hurst found that many historical records appeared to be well represented by

$$E\left[\frac{R(n)}{S(n)}\right] \sim cn^H, \quad \text{as } n \to \infty \tag{4}$$

with Hurst parameter H about 0.7. On the other hand, if the X_k's are Gaussian pure noise or *short-range dependent*, then $H = 0.5$ in (4) and the discrepancy is referred to as the Hurst effect. The Hurst effect is fully accounted for by stationary stochastic processes with long-range dependence. The relation between the Hurst parameter and the rate at which the autocorrelation decays is given by $H = 1 - \beta/2$.

In practice, R/S analysis is based on a heuristic graphical approach, originally described in [5], [12]. Formally, given a sample of N observations, (X_k: $k = 1, 2, \ldots, N$), one subdivides the whole sample into K nonoverlapping blocks and computes the rescaled adjusted range $R(t_i, d)/S(t_i, d)$ for each of the new starting points $t_1 = 1$, $t_2 = N/K + 1$, $t_3 = 2(N/K) + 1, \ldots$ which satisfy $(t_i - 1) + d \leq N$. Here $R(t_i, d)$ is defined as in (3) with W_k replaced by $W_{t_i+k} - W_k$ and $S^2(t_i, d)$ is the sample variance of $X_{t_i+1}, X_{t_i+2}, \ldots, X_{t_i+d}$.

For example if the original time series X_k is of length 50 000 and we decide to subdivide it into 50 blocks, then there will be 50 starting points at 1, 1001, …, 49 001. For values of d less than 1000, we will obtain as many as 50 values of R/S, one for each starting point. All these values of R/S for same

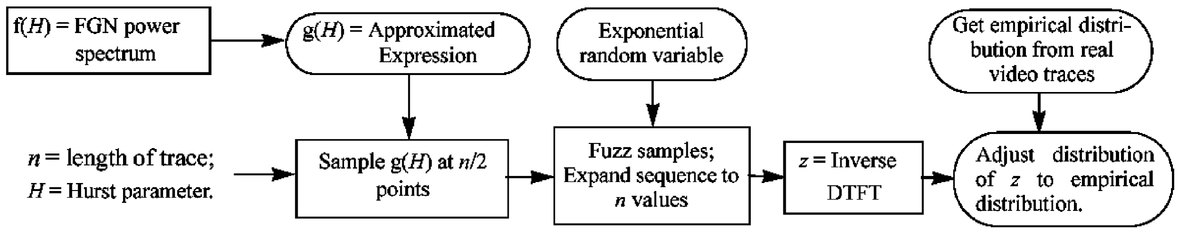

f(H) = FGN power spectrum
g(H) = Approximated Expression
Exponential random variable
Get empirical distribution from real video traces
n = length of trace; H = Hurst parameter.
Sample g(H) at n/2 points
Fuzz samples; Expand sequence to n values
z = Inverse DTFT
Adjust distribution of z to empirical distribution.

Fig. 4. Synthetic trace generation procedure.

values of d are plotted in the form of a scattered plot. For larger values of d, we get less than 50 values of R/S and for values of d greater than 49 000, we get a single value of R/S with the single starting point of 1. So one takes logarithmically spaced values of d to plot the *rescaled adjusted range plot*. The slope of the least square fit line fitting the set of values of R/S gives the asymptotic value of Hurst parameter H.

III. Synthetic Trace Generation

Knowing how the video sequences can be characterized with the Hurst parameter, one can wonder how synthetic traces can be produced from an initial video sequence. This would be extremely useful for producing shorter sequences so as to reduce the simulation time by orders of magnitude, while still maintaining the accuracy of the buffer-loss and delay estimates.

As a first step toward synthetic trace generation, we have to identify the exact statistical properties of the original clip viewed at macroblock-level. The first order statistics are captured by the distribution function. But the buffer occupancy pattern will depend not only on the distribution of sizes of macroblocks but also upon the *order* in which the macroblocks arrive. Indeed, the buffer loss probability will be drastically different in a clip in which many big macroblocks arrive as a burst compared to another clip in which the macroblocks of all sizes arrive uniformly distributed over time. This information is captured by the autocorrelation function. Therefore, the first- and the second-order statistics are the relevant properties of the video clip that should be preserved in order to preserve the pattern of buffer storage in an MPEG-2 decoder.

To capture the long-range slowly decaying autocorrelation function, we need to preserve the Hurst parameter of the original data. This will preserve the *temporal structure* of the arrival process of the macroblocks. We describe here a method based on the discrete time Fourier transform (denoted as DTFT), for synthesizing a trace with the desired Hurst parameter.

The power spectrum of a *fractional gaussian noise* (FGN) process has a closed form expression given by

$$f(\lambda;H) = A(\lambda;H)\left[|\lambda|^{-2H-1} + B(\lambda;H)\right] \tag{5}$$

for $0 < H < 1$ and $-\pi \leq \lambda \leq \pi$, where

$$A(\lambda;H) = 2\sin(\pi H)\Gamma(2H+1)(1-\cos\lambda) \tag{6}$$

$$B(\lambda;H) = \sum_{j=1}^{\infty}\left[(2\pi j+\lambda)^{-2H-1} + (2\pi j-\lambda)^{-2H-1}\right]. \tag{7}$$

The FGN process is widely studied [3] and the closed form expression $f(\lambda, H)$ is approximated by a simpler expression [30] in which the infinite summation in (7) is approximated by a simpler expression. So now we have a simple closed form expression for the power spectrum of an FGN process.

As shown in Fig. 4, to generate a synthetic sequence of length n, and Hurst parameter H, the power spectrum expression is sampled at $n/2$ equidistant points in the frequency domain lying in the interval (0, π). Then a sequence of complex numbers corresponding to this power spectrum is generated; it is a frequency-domain sample path. The complex conjugates of the frequency-domain sample path complete the sequence to make it a full-length sequence. The inverse discrete time Fourier transform of this sample will then be the time-domain counterpart with real numbers and having the autocorrelation function implied by the original expression of the power spectrum. Since autocorrelation and power spectrum form a Fourier transform pair, the time-domain sample will then have the LRD property with Hurst parameter H.

Up to this point, the problem of synthesizing statistically similar traces is still partly solved because the intermediate synthetic data has LRD autocorrelation but still it has Gaussian distribution around mean zero. Real data from video clips may not have Gaussian distribution so we need to transform the distribution function to match a nonGaussian distribution. To this end, let $F_X(x)$ be the Gaussian probability distribution function of the time domain intermediate synthetic trace X_k. Let $F_Y(y)$ be the probability distribution function of the original video clip data obtained empirically. Then, we can generate the process Y_k [31]

$$Y_k = h(X_k) = F_Y^{-1}(F_X)(X_k). \tag{8}$$

This transformed time series Y_k has the same distribution function as that of the real trace. Thus, we have captured the first-order statistics of the real-data traces. An important issue regarding this transformation is that if the process X is a self-similar Gaussian process with the Hurst parameter H, then the nature of the process Y will also be self-similar with the same Hurst parameter H. The main steps in the procedure can be summarized as shown in the following algorithm.

With this procedure, we can thus synthesize artificial traces which mimic the original video clip in terms of the first- and second-order statistical properties. The number of macroblocks to be generated is now under control and, thus, we can effectively achieve a wide range of compression on the original clip. In what follows, we apply the LRD concepts to a real multimedia application.

```
1) Construct $\{f_1, \ldots, f_{n/2}\}$ where $f_i$ represent the sampled values of the approximated expression of the power spectrum [32], and $f_i$ lie equally spaced between (0, $\pi$).
2) Multiply each $f_i$ by an independent exponential random variable with mean 1.
3) Construct a sequence of complex values $\{z_1, z_2, \ldots, z_{n/2}\}$ such that $|z_i| = \sqrt{f_i}$ and the phase is uniformly distributed in the interval (0, $2\pi$).
4) Expand the sequence from $n/2$ values to $n$ values by taking complex conjugates of the $z$ sequence.
5) Inverse Fourier transform of the full length $z$ sequence now gives the LRD sample path.
6) Transform the distribution function from the Gaussian distribution to the empirically observed distribution.
```

IV. Case Study: The MPEG-2 Video Decoder

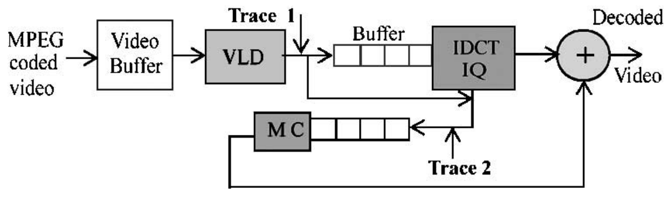
Our main observation is that the traffic between different modules for an MPEG-2 decoder exhibits long-range dependence. This is explained through the example of an MPEG-2 video decoder [Fig. 5(a)]. The decoder consists of the variable length decoder (VLD), inverse quantization (IQ), the inverse discrete cosine transform (IDCT), the motion compensator (MC), and the associated buffers [9].

A. Modeling and Measurement Setup

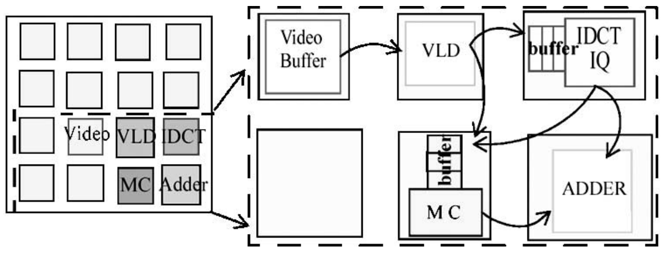
We model the MPEG-2 video decoder using the Stateflow component of Matlab. Stateflow uses the semantics of Statecharts, an efficient formalism for describing concurrency proposed by Harel [10].

To create the Stateflows model of the MPEG-2 video decoder, the *sequential* C-code of the decoder was split into several concurrent processes and the communication among processes made explicit by using synchronization signals. We model the process graph obtained from the application in Fig. 5(a) following the *producer consumer* paradigm; that is, we describe the VLD process as the *producer* and the IDCT/IQ unit as the *consumer*. The VLD decodes the input stream, generates macroblocks, and puts them into the buffer. These are picked up by the *consumer* to compute IDCTs and output data to reconstruct the frames. We assume that all computing processes are mapped onto the tile-based architecture discussed in Section I and shown in Fig. 5(b). The remaining unused tiles can be used to map other applications (e.g. , audio, encryption, etc.).

Using the *Mpegstat* tool developed at University of California at Berkeley [13], we analyze an MPEG-2 video stream and find the detailed information about the macroblocks in the frames of the video. Depending upon whether a frame is of *I*, *P*, or *B* type, the macroblocks are processed differently. An *I* type



(a)



(b)

Fig. 5. (a) Block diagram of the MPEG-2 decoder. (b) Possible mapping of the MPEG-2 decoder onto the architecture in Fig. 2(a).

macroblock is processed only by the VLD and IDCT. Some of the P/B macroblocks need to be processed by VLD, IDCT, and MC blocks with motion compensation performed using one frame stored in the memory. The P/B macroblocks of "skip" type do not need to be processed by IDCT, and just a single frame memory access is needed. Thus, macroblocks follow different paths in the block diagram and take different time to process. This results in various traffic patterns for different video clips.

We monitored the arrival processes at the IDCT and MC modules and recorded their corresponding traces [that is, *Trace 1* and *Trace 2* in Fig. 5(a)]. The corresponding traces obtained were further evaluated using variance-time and R/S methods mentioned in Section II. Using these methods, we were able to obtain the variance-time and R/S plots for the two traces. The results are discussed in Section IV-B.

B. Results and Discussion

Our approach to traffic modeling is "data driven." We rely upon five video sequences (*Fish*, *Clouds*, *Simpsons*, *Disc_ir*, *Hawaii*) of different video screen sizes ranging in length from 27 s (113 000 macroblocks) to 1 s (27 000 macroblocks). This represents all kinds of different scenes as shown in Table I by the statistics of *I*, *P*, and *B* frames.

We focus on long sequences (X_i: $i = 1, 2, \ldots, N$) of data, where X_i represents the *number of bits*, which contain the compressed and coded information for a macroblock in a frame of an MPEG video. Based on statistical analysis of the sequences, our main finding is that LRD *is* indeed a characteristic of the MPEG-2 video traffic traces between different modules of an MPEG-2 decoder.

The monitored trace file consists of two columns. The first one records the *time* measured from the beginning of the trace until a macroblock of the video stream arrived at a module in the system. The second column gives the integer *size* in bits of the macroblock. The actual traffic therefore consists of alternating

TABLE I
STATISTICS FOR DIFFERENT CLIPS

Video Clip	I frames	P frames	B frames	Macroblocks per frame
Fish	23	100	220	330
Clouds	24	12	0	1200
Simpsons	136	136	542	108
Disc_ir	18	9	0	1024
Hawaii	195	96	0	300

TABLE II
HURST PARAMETER VALUES FOR DIFFERENT CLIPS USING TWO METHODS FOR TWO DIFFERENT TRACES

Video Clip	Trace 1 H by Variance-time method	Trace 1 H by R/S plot method	Trace 2 H by Variance-time method	Trace 2 H by R/S plot method
Fish	0.7251	0.7147	0.6308	0.7136
Clouds	0.7240	0.7646	0.7603	0.7639
Simpsons	0.6874	0.7432	0.7407	0.7943
Disc_ir	0.8108	0.8180	0.8421	0.8131
Hawaii	0.7238	0.7453	0.5455	0.6839

sequence of macroblock arrivals and silence periods. We convert the arrival trace to a time series by averaging the trace within a window of size δ as described in [8].

To compute H, we perform two tests.

1) The ***first test*** corresponds to the variance-time analysis of the time series X as described in Section II-A. The variance-time plots are obtained by plotting $\log(\mathrm{var}(X^m))$ against $\log(m)$. As an illustration, Fig. 6 shows the plots corresponding to two different video clips (*Simpsons* and *Hawaii*). We ignore the small values of m and find the least square fit line to the graph. The slope of the line gives the value of the parameter β, from which we find out the Hurst parameter using the formula $H = 1 - \beta/2$.
2) The ***second test*** corresponds to the R/S analysis of the time series as described in Section II-C. The rescaled adjusted range statistics for different video clips are shown in Fig. 7.

For convenience, a summary of the estimated Hurst parameters is also given in Table II. As shown the values of H lie between 0.5 and 1, clearly indicating the presence of LRD. Also, the values of H obtained from both methods are sufficiently close to each other to further support the claim about the presence of LRD.

There exist other techniques to test the presence of LRD and estimate the value of H parameter. Some of the methods are more advanced than traditional variance-time and R/S methods used in this paper. However, in order to prove the existence of LRD, it is sufficient to use the most widely used estimators used in this paper. More discussion about the accuracy of H parameters estimated using various methods can be found in [35].

C. Beyond the Tile-Based Architecture

In our experimental setup, the on-chip traffic traces are recorded for a tile-based architecture communicating using a network on-chip. However, the simulation results also apply to other communication architectures (e.g., point-to-point communication between the processors). This is because the bursty nature of traffic, which gives rise to the long-range dependence is mainly due to the variations in the *number of bytes* used for coding these macroblocks. This variation in the number of bytes is going to persist even if we have a communication architecture other than the on-chip network for tile-based architecture. For instance, we believe that the bursty nature will also be present in the traffic traces for a bus-based communication architecture. In the case of a bus-based architecture, there is contention for the bus as a shared resource. This results in blocking among communication streams depending upon the arbitration policy. The blocking leads to a variation in the interarrival times of the macroblocks at each module. The LRD property, by its very nature of being present across multiple time scales, is immune to these small variations in the interarrival times of the macroblocks. The time-scales for which the LRD property holds, are hundreds, even thousands of times greater than the interarrival times between the macroblocks (as can be seen from the Figs. 6 and 7). The small variations in the interarrival times do not destroy the LRD nature of the traffic, which is mainly due to the variable sizes of the macroblocks.

V. IMPLICATIONS OF LRD IN DESIGNING ON-CHIP NETWORKS

Beyond its statistical significance, LRD has considerable impact on queueing performance of on-chip networks. Only a small number of analytical queueing results are available for LRD traffic [7].

A. Analytical Buffer Size Prediction

In traffic analyses, we typically deal with time series with hundreds of thousands of observations. A traffic model for a particular trace is not necessarily unique. Different models serve different purposes and, as it has been shown (e.g., [26] and [32]), the SRD models can be used to predict buffer overflow probability even in traces which have LRD. However, if we try to fit the best ARMA model to such a LRD process, then the number of parameters will tend to infinity. Using an excessive number of parameters is undesirable as the parameters are difficult to interpret. So we need to model these processes with parametrically parsimonious models.

Norros in [7] used *fractional Brownian motion* (FBM) model which parsimoniously captures the LRD effects. This model determines the *lower bound* for the probability that the queue length Q *exceeds* a certain buffer size x, under the assumption of having an infinite buffer

$$P(Q > x) \sim \exp[-cx^{2-2H}] \tag{9}$$

$$c = \frac{m^{2H-1}}{2a}\left(\frac{1-\rho}{\rho}\right)^{2H} \times \left[\left(\frac{1-H}{H}\right)^{H} + \left(\frac{H}{1-H}\right)^{1-H}\right]^{2} \tag{10}$$

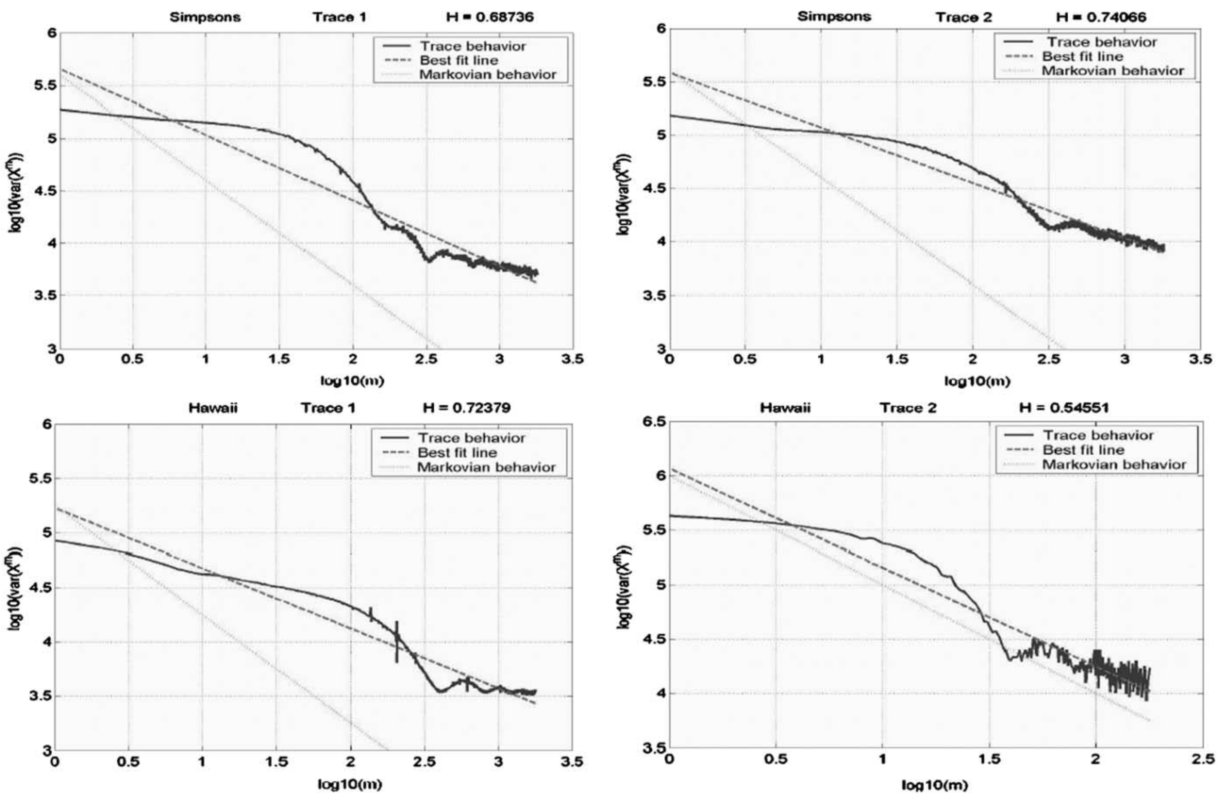


Fig. 6. *Variance-time* plots for two different video clips at the IDCT module in the [trace 1 in Fig. 5(a)] and at the MC module [trace 2 in Fig. 5(a)] in the MPEG-2 decoder. The graph shows the least square fit line as well as the line with slope -1. The least square fit line can be seen to lie above the line with -1 slope which shows self-similarity. For large m, the slope of best fit line lies in $(-1, 0)$.

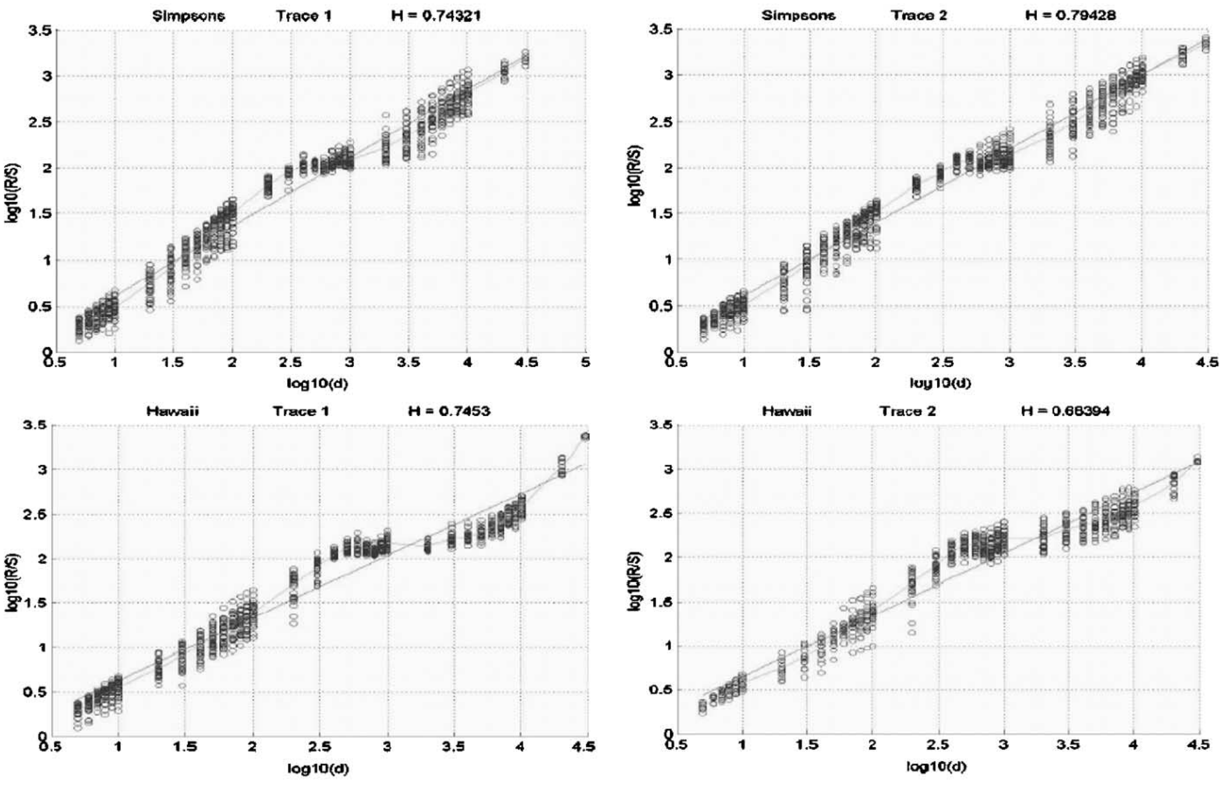


Fig. 7. R/S plots for two different video clips at IDCT module [trace 1 in Fig. 5(a)] and at the MC module [trace 2 in Fig. 5(a)] in the MPEG-2 decoder. The slope of the best fit line equals the value of the Hurst parameter.

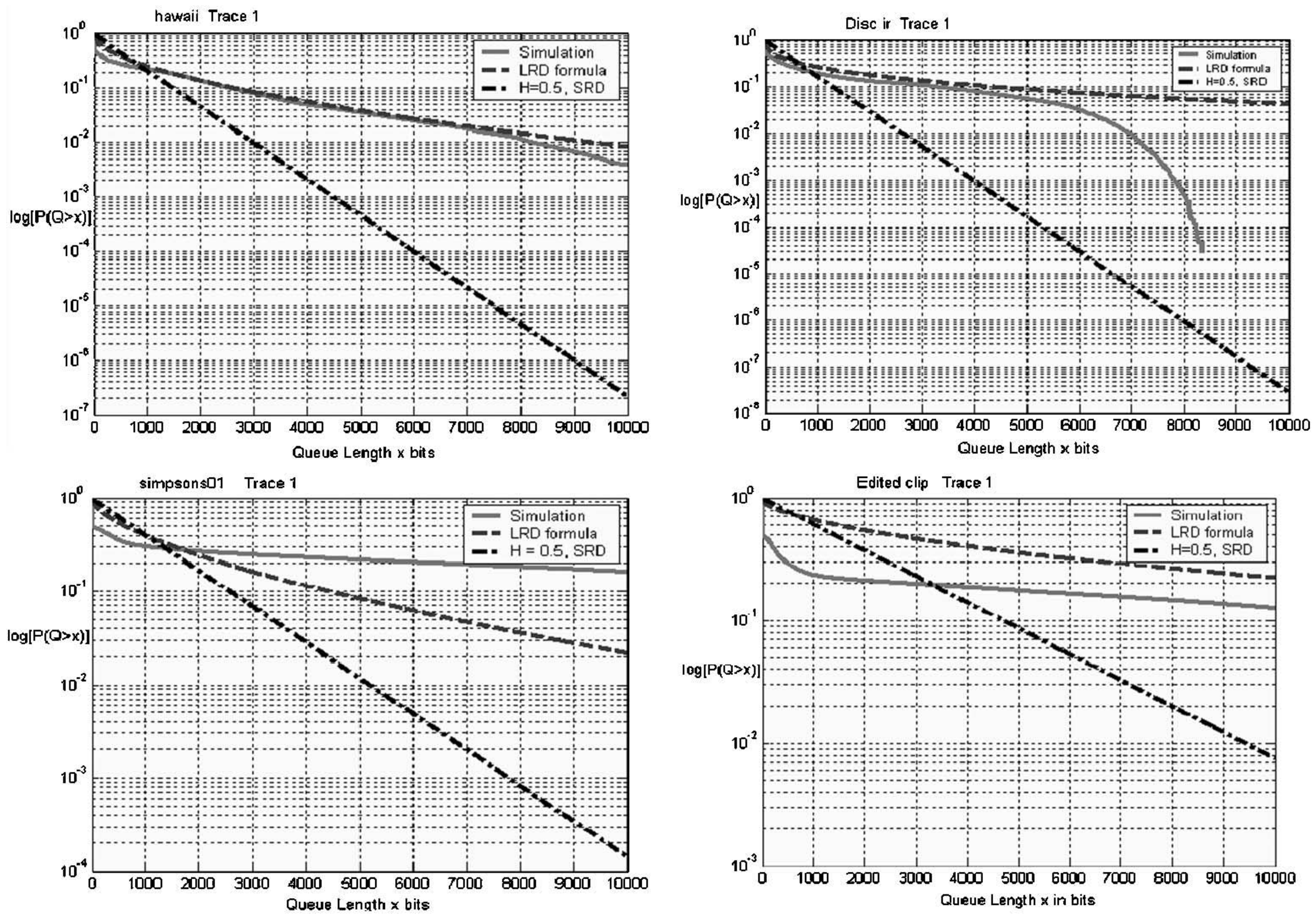


Fig. 8. The dashed curves indicate complementary queue length distribution plots predicted by (9) and (10) for different video clips. The solid curves indicate simulation results for an infinite queue with the arrival process following an empirical trace (the consumer utilization is 0.5). The straight lines indicate the prediction by a *short-range dependent* model ($H = 0.5$ in (9) and (10)).

where: m is the mean input rate, ρ is the utilization of the queue (that is, the ratio of average service time to average interarrival time); H and a are the Hurst parameter and the peakedness (ratio of variance to mean) values which can be obtained from plots like those in Figs. 6 and 7.

To assess the *accuracy* and *impact* of our predictions on the overall performance of the on-chip network, the complementary buffer-length distributions for two different video traces are shown in Fig. 8. In these graphs, the dashed curves indicate the *predicted* probability values given by (9) and (10), while the continuous curves indicate the results obtained by *simulation*. There are a few conclusions which can be derived from these plots.

- First, the predicted and simulated curves show a very good agreement as a function of buffer length. That is, the small difference between them is because the simulation corresponds to just one instance of the arrival process while the analytical formula gives the result averaged over *many* traces. That is the reason why simulation curves, which represent just one instance of the stochastic process with that particular value of the Hurst parameter lie close to the curve predicted by the deterministic formula and sometimes overestimate or underestimate the buffer length.
- Second, the dash–dot lines (obtained for $H = 0.5$) correspond to short-range dependent models (like the Markovian ones). From the plots in Fig. 8, we can see that Markovian models significantly underestimate (typically, 1–2 orders of magnitude) the buffer-overflow probabilities since it assumes the distribution of the arrival process to be short-range dependent (i.e. exponential). This may cause severe performance degradation at chip-level.
- Third, as we can see from Fig. 8, the buffer overflow probability for *Disc_ir* clip obtained by simulation deviates from the same obtained by the analytical formula as buffer size increases. The reason for this may be that the event of buffer overflow becomes a rare event as we increase the size of the buffer. In order to capture such a rare event by simulation, we have to simulate a longer trace. The *Disc_ir* clip contains very few number of macroblocks. So we "enriched" the *Disc_ir* trace by concatenating *Clouds* followed by *Simpsons* followed by *Disc_ir* and the bottom right graph in Fig. 8 corresponds to the simulation of this edited trace. Again, we can see a very good agreement between the buffer-overflow prediction by the LRD formula (9) and the simulation values.

There is also another way of using the plots in Fig. 8 [and, therefore, (9)] for on-chip network design. For instance, if the

QoS needed by the target application asks for not more than 1% of lost macroblocks, then from the first plot in Fig. 8, one can easily see that we need a buffer length of 9000 b at the IDCT module. This way, we have a theoretical basis for choosing the buffer length. On the other hand, the Markovian analysis will predict a buffer length of 3000 b and that will result in around 10% bits lost (instead of the targeted 1%). This represents a very serious performance degradation for an MPEG-2 video decoder. More generally, a multimedia system designer may have a set of typical video clips (e.g. news reader, weather channel, etc.) that are expected to run using an MPEG-2 decoder. We can find the Hurst parameter for each of these clips and use it to predict different buffer lengths corresponding to the same degree of lost bits. We can then choose the maximum predicted buffer length and have a theoretical support for assuring the QoS guarantee to the consumer.

Last but not least, compared to simulation-based buffer sizing, the LRD analysis framework offers a *fast* approach for buffer sizing. More precisely, for the simulation-based approach, if we want to assess the impact of changing the speed of the processors in the on-chip network, then we need to rerun the simulations all over again for all the typical video clips. This is extremely time consuming, since tens of hours of simulation may be needed. Conversely, in the LRD-based analysis, the value of H will *not* be affected as it depends only on the underlying video clip statistical properties. Consequently, we just need to change the value of the utilization factor ρ in (9) and (10), and quickly compute the new buffer length. For instance, the variation in the probability of buffer overflow (as a function of utilizations $\rho = 0.3$, 0.5 and 0.7) is shown in Fig. 9. For a buffer length of 10 000 b, the probability of overflow is nearly zero if $\rho = 0.3$; it becomes significant (more than 0.5) if $\rho = 0.7$. This means that the probability of overflowing the IDCT buffer is higher for a slower processor implementing the IDCT functionality.

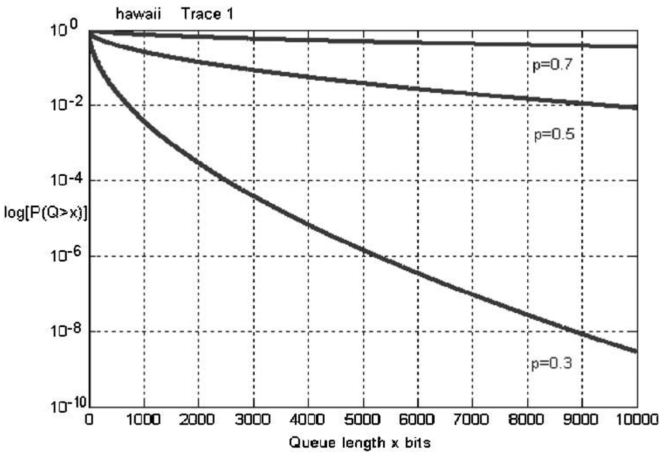


Fig. 9. Complementary queue length-distribution plots for different levels of utilization of the server.

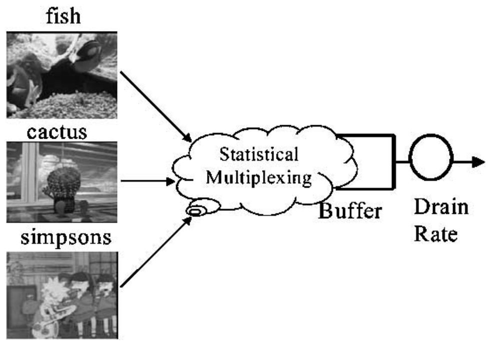


Fig. 10. Experimental setup for evaluating delay and loss for statistically multiplexed traces.

B. Simulation Speedup

In order to evaluate the effectiveness of our synthetic trace generation procedure, we perform the following simulation. We look at a set of three actual MPEG clips and obtain their macroblock-level traces using the *Mpegstat* tool [13]. From one particular video clip, each macroblock is assumed to be arriving at a constant time interval to the buffer at the VLD. But different clips start sending macroblocks at different times which are uniformly distributed in one macroblock interval of one another. Thus, macroblocks from the same source arrive at regular intervals of one another, but those from different sources arrive at different times, which are uniformly distributed within one macroblock interval. All these three sources are then statistically multiplexed into a common buffer as shown in Fig. 10.

The server at the buffer is assumed to be serving the macroblocks at constant drain rate in terms of the number of bits processed per second. By varying this drain rate, the utilization of the queue can be varied. Using the same setup, we then replace each video source by a synthetically generated trace, which is *half the length* of the original full-length trace (but having the same Hurst parameter as that of the original video) and perform the simulations for buffer-loss probability and delay again. Similar simulation results are available in the literature (see [32] and [33], for instance) but the analysis there is carried at frame level and the models used, as opposed to ours, are Markovian models.

The results of our simulations are shown in Fig. 11. We can see from the plots that the loss probability curves for the real trace and the synthetic trace are practically the same for high levels of utilization greater than 0.25 (Note, that we are plotting inverse of utilization on the x axis). The delay curves can also seem to be close to each other. Note that the simulation time for the synthetic traces is reduced to half because the length of the synthetic trace is half of that of the original trace. As we increase the speedup, the length of the compacted trace becomes smaller. Then the buffer overflow events, which are rare events and occur in bursts due to bursty nature of traffic, are difficult to capture in simulation. It is possible to achieve further speedup (more than double) by using this technique if the compacted synthetic trace is long enough to get buffer overflow probabilities by simulation.

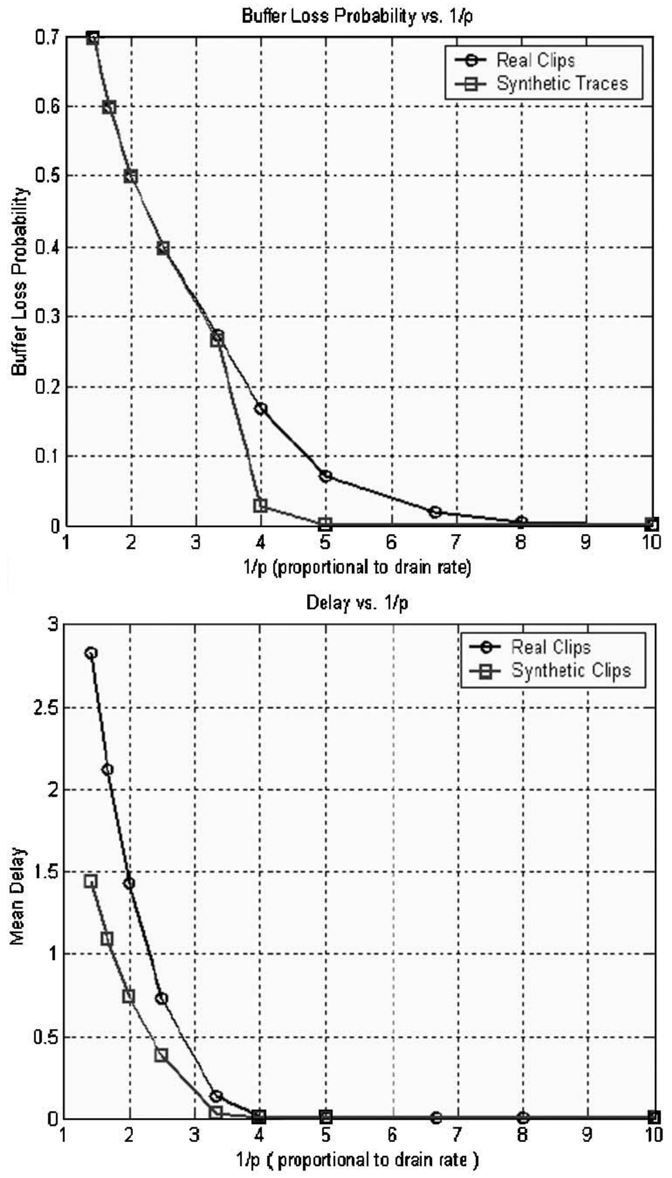


Fig. 11. Loss probability and delay for real and synthetic traces using multiplexed streams.

VI. Conclusion

We have presented a technique for on-chip traffic analysis using self-similar processes. For a recently proposed communication architecture based on packet switching, we have shown that, under various input traces, the arrival process at different nodes, for an MPEG-2 video application, exhibits self-similar phenomena. Characterizing the degree of self-similarity via the Hurst parameter helps in finding the optimal buffer-length distribution, which turns to be the critical issue for the routers at each node in the on-chip communication network. Finally, a synthetic trace generation procedure can be used to significantly reduce the simulation time for calculating the buffer loss probability and the delay.

Acknowledgment

The authors would like to thank the anonymous reviewers for their suggestions that improved several drafts of this paper.

References

[1] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proc. Design Automation Conf. (DAC)*, Las Vegas, NV, June 2001.
[2] W. E. Leland *et al.*, "On the self-similar nature of Ethernet traffic," *IEEE/ACM Trans. Networking*, vol. 2, pp. 1–15, Feb. 1994.
[3] J. Beran, *Statistics for Long-Memory Processes*. London, U.K.: Chapman & Hall, 1994.
[4] D. R. Cox, "Long-range dependence: a review," in *Statistics: An Appraisal*, H. A. David and H. T. David, Eds. Ames, IA: Iowa State Univ. Press, 1984.
[5] J. Beran *et al.*, "Long-range dependence in variable-bit-rate video traffic," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, 1995.
[6] A. Erramilli, O. Narayan, and W. Willinger, "Experimental queueing analysis with long-range dependent packet traffic," *IEEE/ACM Trans. Networking*, vol. 4, Apr. 1996.
[7] I. Norros, "A storage model with self-similar input," *Queueing Systems*, vol. 16, 1994.
[8] P. Abry and D. Veitch, "Wavelet analysis of long-range dependent traffic," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2–15, Jan. 1998.
[9] T. Sikora, "MPEG digital video coding standards," *IEEE Signal Processing Mag.*, vol. 14, p. 58, Sept. 1997.
[10] D. Harel, "Statecharts: a visual formalism for complex systems," *Sci. Comput. Program.*, vol. 8, pp. 231–274, June 1987.
[11] B. B. Mandelbrot and J. R. Wallis, "Computer experiments with fractional Gaussian noises," *Water Resources Research*, vol. 5, pp. 228–267, Feb. 1969.
[12] B. B. Mandelbrot and M. S. Taqqu, "Robust R/S analysis of long run serial correlation," in *Proc. 42nd Session ISI*, 1979.
[13] . [Online] http://bmrc.berkeley.edu/ftp/pub/mpeg/stat/
[14] K. Lahiri, A. Raghunathan, and S. Dey, "Evaluation of the traffic-performance characteristics of system-on-chip communication architectures," in *Proc. Int. Conf. VLSI Design*, Bangalore, India, Jan. 2001.
[15] A. Kalavade and P. Moghe, "A tool for performance estimation of networked embedded end-systems," in *Proc. Design Automation Conf. DAC*, San Francisco, CA, June 1998, pp. 257–262.
[16] K. Keutzer *et al.*, "System-level design: orthogonalization of concerns and platform-based design," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 1523–1543, Dec. 2000.
[17] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli, "Design of embedded systems: formal models, validation, and synthesis," *Proc. IEEE*, vol. 85, pp. 366–390, Mar. 1997.
[18] A. Mathur, A. Dasdan, and R. Gupta, "Rate analysis for embedded systems," *Trans. Design Automation Electron. Syst.*, vol. 3, no. 3, pp. 408–436, July 1998.
[19] A. Nandi and R. Marculescu, "System-level power/performance analysis for embedded systems design," in *Proc. Design Automation Conf. (DAC)*, Las Vegas, NV, June 2001.
[20] ——, "Probabilistic application modeling for system-level performance analysis," in *Proc. Design Automation and Test Eur. (DATE)*, Munich, Germany, Mar. 2001.
[21] K. Lahiri, A. Raghunathan, and S. Dey, "Fast performance analysis of bus-based system-on-chip communication architecture," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, Nov. 1999, pp. 566–572.
[22] M. Gasteir and M. Glesner, "Bus-based communication synthesis on system level," *Trans. Design Automation Electron. Syst.*, pp. 1–11, Jan. 1999.
[23] T. Yen and W. Wolf, "Communication synthesis for distributed embedded systems," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, Nov. 1995, pp. 288–294.
[24] J. A. Rowson and A. Sangiovanni-Vincentelli, "Interface based design," in *Proc. Design Automation Conf. (DAC)*, June 1997, pp. 178–183.
[25] J. Daveau, T. B. Ismail, and A. A. Jerraya, "Synthesis of system-level communication by an allocation based approach," in *Proc. Int. Symp. System Synthesis (ISSS)*, 1995, pp. 150–155.
[26] K. Park and W. Willinger, Eds., *Self-Similar Network Traffic and Performance Evaluation*. New York: Wiley, 2000.
[27] M. Sgroi *et al.*, "Addressing the system-on-a-chip interconnect woes through communication-based design," in *Proc. Design Automation Conf. (DAC)*, Las Vegas, 2001.
[28] F. Karim, A. Nguyen, S. Dey, and R. Rao, "On-chip communication architecture for OC-768 network processors," in *Proc. Design Automation Conf. (DAC)*, Las Vegas, June 2001.
[29] V. Paxson and S. Floyd, "Wide-area traffic: the failure of poisson modeling," in *Proc. Association for Computing Machinery (ACM) SIGCOMM'94*, London, U.K., 1994.

[30] V. Paxson. (1995) Fast approximation of self-similar network traffic. [Online]. Available: ftp://ftp.ee.lbl.gov/papers/fast-approx-selfsim.ps.Z

[31] C. Huang, M. Devetsikiotis, I. Lambadaris, and A. Roger Kaye, "Modeling and simulation of self-similar variable bit rate compressed video: a unified approach," in *Proc. Association Computing Machinery (ACM) SIGCOMM*, 1995, pp. 114–125.

[32] D. Turaga and T. Chen, "Hierarchical modeling of variable bit rate video sources," in *Proc. Packet Video 2001*, Kyongju, Korea, Apr. 30–May. 1 2001.

[33] ——, "Activity-adaptive modeling of dynamic multimedia traffic," in *IEEE Int. Conf. Multimedia Expo.*, New York, NY, July 2000.

[34] A. Hemani *et al.*, "Network on chip: an architecture for billion transistor era," in *Proc. IEEE Norchip Conf.*, Nov. 2000, pp. 117–124.

[35] T. Karagiannis, M. Faloutsos, and R. Riedi, "Long-range dependence: now you see it, now you don't!," in *Proc. Global Internet Symp.*, Taipei, Taiwan, R.O.C., Nov. 2002.

[36] M. W. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proc. Association Computing Machinery (ACM) SIGCOMM*, 1994, pp. 269–280.

Girish V. Varatkar (S'01) received the B.S. degree in electrical engineering from the Indian Institute of Technology (IIT), Delhi, India, and the M.S. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in May 2003.

His research interests include CAD for low-power VLSI systems and efficient architectures for digital and signal processing applications.

Radu Marculescu (M'94) received the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1998.

Currently, he is an Assistant Professor in the Department of Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, PA. His research interests include developing design methodologies and software tools for SOC design, on-chip communication with emphasis on power/performance tradeoffs, and ambient intelligent systems.

Dr. Marculescu was the 2001 recipient of the National Science Foundation's CAREER Award in the area of design automation of electronic systems. In 2001 and 2003, he received a Best Paper Award from the *Design Automation and Test in Europe (DATE) Conference*, and from the *Asia and South Pacific Design Automation Conference (ASP-DAC)* in 2003. He also was awarded the Carnegie Institute of Technology's Ladd Research Award for 2002 and 2003. He is a Member of the Association for Computing Machinery (ACM).