Managing Power Consumption in Networks on Chips

Tajana Simunic, Stephen Boyd, and Peter Glynn, *Member, IEEE*

***Abstract*—In this work we present a new methodology for managing power consumption of NOCs. Power management problem is formulated for the first time using closed-loop control concepts. We introduce an estimator and a controller that implement our power management methodology. The estimator is capable of very fast and accurate tracking of changes in the system parameters. Parameters estimated are used to form the system model. Our system model combines node and network centric power management decisions. Node centric power management assumes no a priori knowledge of requests coming in from outside the core. Thus it implements a more traditional dynamic voltage scaling and power management control algorithms. Network-centric power management utilizes interaction with the other system cores regarding the power and the QoS needs. The overall system model is based on Renewal theory, and thus guarantees globally optimal results. We introduce a fast optimization method that runs multiple orders of magnitude faster than the previous optimization approaches while still having the same accuracy in obtaining the power management control. Finally, our controller implements the results of optimization in either hardware or software. The new methodology for power management of NOCs is tested on a system consisting of four satellite units, each implementing an estimator and a controller capable of both node and network centric power management. Our results show large savings in power with good QoS.**

***Index Terms*— Energy management, Low power, Regenerative stochastic processes, System analysis and design**

I. Introduction

Future technology will make it possible to place an even larger number of transistors on a single die, together with many different layers of interconnect. Today's SOCs are designed as a tightly interconnected set of cores, where all components share the same system clock, and the communication between components is via shared-medium busses. Even though design implementation is limited by wire density, currently wires toggle approximately only 10% of the time [2]. As the features sizes shrink, and the overall chip size increases, the interconnects start behaving as lossy transmission lines. Crosstalk, electro-magnetic interference, and switching noise cause higher incidence of data errors. Line delays have become very long as compared to gate delays causing synchronization problems between cores. A significant amount of power is dissipated on long interconnects and in clocking network. Lowering the power supplies and designing smaller logic swing circuits is one way to help with the overall power consumption, but it comes at the cost of higher data errors. In fact, power savings obtained by only scaling down supply voltage levels are not sufficient to compensate for higher complexity, larger interconnect capacitance and resistance, higher operating frequency and increased gate leakage [5].

One solution to these problems is to treat SOCs as *micro-networks*, or *Networks On Chips* (NOCs) where the interconnections are designed using an adaptation of the protocol stack [1,2,7]. Networks have a much higher bandwidth due to multiple concurrent connections. They have regular structure, so the design of global wires can be fully optimized and as a result their properties are more predictable. Overall performance and scalability increase since the networking resources are shared. Scheduling of traffic on shared resources prevents latency increases on critical signals. Networking model decouples the communication layers so that design and synthesis of each layer is simpler and can be done separately. In addition, decoupling enables easier management of power consumption and performance at the level of communicating cores.

This work presents a new methodology for managing power consumption in NOCs. The power management optimization problem is formulated and solved for the first time using a closed-loop control model with a combination of node and network centric power management approaches. Each communicating core has a power manager that consists of an estimator and a controller. The estimator tracks changes in the state of the local core, incoming traffic to the core (node-centric) and the special requests for power management coming from the other cores on the network (network-centric). All estimated parameters define the system model. Our model expands the Renewal model presented previously in [14] to combine node and network centric approaches into one, in addition to including expanded state space needed for dynamic voltage scaling in NOCs. We guarantee globally optimal control based on this model. Our new fast optimization methodology improves optimization speed by multiple orders of magnitude over the old approaches. Finally, we also present both hardware and software

Manuscript received Aug 30, 2002.

Tajana Simunic is with HP Labs and Stanford University, 1501 Page Mill, MS 1181, Palo Alto, CA 94304. (phone: 650-236-5537, fax: 650- 857-8491; e-mail: tajana@stanford.edu).

Stephen Boyd and Peter Glynn are with Stanford University, Stanford, CA 94305 (e-mail: boyd@stanford.edu, glynn@stanford.edu).

implementations for the optimal controller. We implemented our approach on a sample system of four cores in one NOC and document savings of a factor of four due to our methodology.

The rest of the paper is organized as follows. Section II discusses related work. We introduce our approach for managing power in NOCs in Section III. The estimator, introduced in Section IV, tracks all changes in system behavior and feeds that information to our Renewal system model presented in Section V. The controller, discussed in Section VI, implements both power management and voltage scaling decisions obtained from the optimization step. A sample design of a power management system for NOC is presented in Section VII, along with the experimental results. Finally, the Section VIII summarizes the contributions of our work.

II. RELATED WORK

Design of Networks on Chips (NOCs) is a relatively new field with numerous challenges. The first challenge is the design of the communication network between the cores in a NOC. More recently, there have been a few publications that define the NOC architecture based on the packet communication model [4]. The work presented in [3] uses fat tree router topology to form an integrated packet switched network with message passing protocol and 32 bit packet sizes. Much larger packet sizes (256 data and 38 control bits) and tiled architecture are suggested in [2]. The communication layers in NOCs can be partitioned much like the structure proposed by OSI Reference Model for the computer networks in [1,7]. MESCAL provides tools for correct-by-construction protocol stack [1]. The layers of protocols encapsulate original computation cores to maximize reusability. Adapters are used to bridge the differences between communication needs of the cores. An example implementation is Maia processor [8], which consists of 21 satellite units connected via two-level hierarchical reconfigurable network. Large energy savings were observed due to the ability of Maia to reconfigure itself according to application needs.

Reduction of energy consumption in NOCs is another challenge that needs to be considered, in tandem with the design of the on-chip communication network [7]. Power savings obtained by only scaling down supply voltage levels are not going to be sufficient to compensate for a higher complexity, a larger interconnect capacitance and resistance, a higher operating frequency and an increased gate leakage [5]. Previous work for energy management of NOCs mainly focused on controlling the power consumption of interconnects [6], while neglecting managing power of the cores. An outline of possible approaches for energy savings in NOC cores is presented in [7]. Two approaches are suggested: node-centric and network-centric, but no specific implementation issues are discussed. In this work we present an optimal way to implement both node and network centric approaches using a closed-loop control model.

Many of the cores that are of interest in NOC design already have multiple power and performance states which can be exploited. Dynamic power management (DPM) and dynamic voltage scaling (DVS) algorithms presented to date [10-13, 15-25] have been designed with portable systems in mind, not NOCs with very fast response times. In addition, they are targeted devices such as hard disks or wireless LAN that in our case is equivalent to node-centric level in NOCs, thus there is not accounting for network-centric power management. Most DPM and DVS algorithms are deterministic in nature. Deterministic design methodology used in today's designs is being replaced by statistical modeling, to account for unreliability of the on-chip communication and non-deterministic nature of user's requests. As a result, there is a need to develop stochastic model of NOCs to be used as the basis for optimization of power consumption under QoS constraints.

In the past, stochastic models for DPM at the system level have been formulated using Markov models with open-loop control, where statistics of the device are collected and characterized ahead of time, and the control is derived based on those with no adaptation at run time [10, 12,13]. An exception is the adaptive approach presented in [11] that uses only memoryless distributions to describe the history-dependent system behavior. In contrast, our model of NOCs is based on Renewal theory concepts, which enables us to use non-exponential distributions to describe both node and network centric core behaviors. We accurately adapt to any changes in the system at run-time with the first closed-loop power management system for NOCs. When changes in any of the model parameters are detected at run-time, our new fast optimization method quickly re-evaluates both the power management and the voltage scaling control. The next section gives an overview of our approach for power management in NOCs.

III. POWER MANAGEMENT IN NOCS

Networks on chips consist of a set of cores connected with the communication network. Figure 1 shows a sample NOC we will be referring to throughout this paper. The NOC consists of four cores: MPEG audio, video, speech processing and communication core. The cores communicate with each other via router using a networking protocol. The design of the networking protocol for NOCs is beyond the scope of this paper, but has been addressed in [2,3]. In this work we enhance each core's ability to control its power states by enabling closed-loop integrated node and network centric power management with dynamic voltage scaling.

In order to compute the power manager's control, we need to develop a system model. We model NOC using Renewal theory as a queuing network with a number of service points representing cores. Management of energy consumption under QoS constraints is formulated as a closed-loop stochastic control problem.

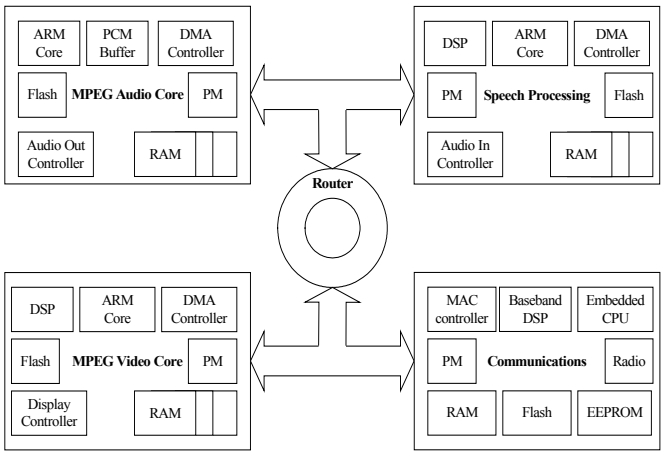

Figure 1. Network On a Chip

Control theory defines three different entities in a closed-loop control system: a system under control, an estimator and a controller. Power manager (PM), as shown in Figure 2, contains the controller, and the estimator. The estimator observes the requests coming into the core's queue (*Core Traffic* in Figure 2), the state of the core and the incoming power management requests from the network (*Network PM Request* in Figure 2). Based on the observations, it tracks any changes in the system parameters. When a change is detected, it estimates the new parameter value, and recalculates the power management control using our fast optimization method. The controller implements power management control calculated by the estimator. It gives commands to the core that determine its performance and energy characteristics (frequency and voltage) in the active state, and chooses when to transition the core into one of the available low-power states when the core is idle.

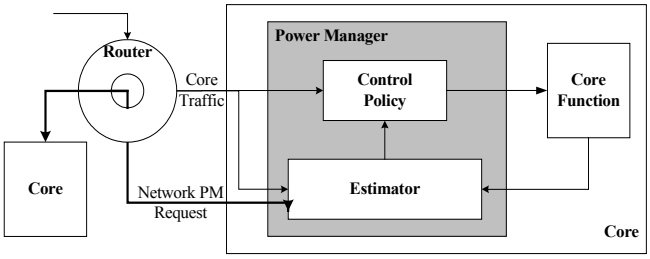

Figure 2. Core's Power Manager

To illustrate our power management system operation, we will use the NOC shown in Figure 1 with streaming MP3 audio example. Let us assume that initially all cores in the NOC are in the sleep state. The MP3 core's controller starts the transition from the sleep into the active state as soon as the user request arrives to it via core traffic input. Right at the beginning of initialization of MP3 decoder, the MP3 core power manager also sends a network request via network PM request port to the communications core to start its wakeup process. Thus any performance penalty that might have been present without network centric power management is now masked. Next, the MP3 core receives encoded MP3 stream data from the communications core via core traffic input. MP3 estimator watches for the changes in the incoming data and decoding rates or a discrepancy between measured and desired MP3 QoS parameters at run-time. Upon detection, it recalculates the power management control using our system model and our fast optimization method, and thus closes the optimization loop. The results of the detection are both new power management and dynamic voltage scaling control. For example, when a new MP3 decoding session starts, the MP3 estimator needs to match the incoming data rate with the decoding rate so that there is no buffer overflow (modeled as a queue in our system), and so QoS requirements, such as frame delay, are met. This is done by setting appropriate processing speed and voltage during the decoding time. In addition, once the current MP3 session ends, the estimator uses the calculated power management policy to decide when the MP3 core should go into a low power state. Once the communication's core services are not required anymore, the communication core's PM is notifies via a network PM request. At that point, if no other requests are pending, the communications core can enter a sleep state without any additional idle time. As a result, the amount of energy wasted while the core is idle is reduced. The following sections describe in more details our closed-loop power management system.

IV. ESTIMATOR

The main task of the estimator is to observe the system behavior and based on that to estimate the parameters needed for optimization and control. The quality of power management decisions strongly depends on estimator's ability to track changes of critical parameters at run-time. NOC power management requires estimation of workload characteristics, core parameters, and buffering behavior.

A. Workload characteristics

Each core's workload includes request arrivals to the active state (buffer is not empty), the idle states (buffer is empty), and the network request arrivals. We distinguish between active and idle state arrivals, as they are characterized by two different distributions, and they also signify a different type of a decision on the part of the power manager. In the active state, PM decides only on the appropriate frequency and voltage setting, where as in the idle state the primary decision is which low-power state core should transition to. The distribution of network requests only affects the decisions made by the PM in the idle and the low-power states.

$$P_{workload} = 1 - e^{-\lambda_{workloade} t} \qquad (1)$$

Request arrivals to non-empty buffer (active state)

In the active state, the workload is modeled using exponential distribution with request interarrival rate $\lambda_{workload}$

as shown in Equation 1. For example, the workload's stochastic model in the active state can be defined by the frame interarrival time distribution for multimedia requests. We measured MPEG2 video (CIF size) and MP3 audio frame arrival times by monitoring the accesses to the WLAN core and found a good fit with the exponential distribution.

Detecting the change in rate is a critical part of optimally matching processing frequency and voltage to the requirements of the user. For example, the rate of MP3 audio frames coming from WLAN core can change drastically due to the environment. The servicing rate can change due to variance in computation needed between MPEG frames. The most optimal method of tracking rate changes is using the maximum likelihood estimator shown in Equation 2 (see [28] for details).

$$P_{\max} = \frac{\prod_{j=1}^{c-1} \lambda_o e^{-\lambda_o \Delta t_j} \prod_{j=c}^{w} \lambda_n e^{-\lambda_n \Delta t_j}}{\prod_{j=1}^{w} \lambda_o e^{-\lambda_o \Delta t_j}} \quad (2)$$

Maximum likelihood ratio computes the ratio between the probability that a change in rate did occur (numerator in Equation 2) and the probability that rate did not change (denominator). This estimator guarantees optimal results with parameters defined as follows: w is the size of the window that holds the last set of interarrival times Δt, c is the point in the past when the change in rate occurred, λ_n is the new rate, and λ_o is the old rate.

An more efficient way to compute the maximum likelihood ratio at run time is to calculate the natural log of P_{max} as shown in Equation 3. Note that in this equation only the sum of decoding (or interarrival) times needs to be updated upon every service completion (or arrival). A set of possible rates, Λ, and the window size, w, are predefined by the system designer. Values λ_n and λ_o, come from the set Λ.

$$\ln(P_{\max}) = (w - c + 1)\ln\frac{\lambda_n}{\lambda_o} - (\lambda_n - \lambda_o)\sum_{j=c}^{w} \Delta t_j \quad (3)$$

The change point detection algorithm consists of two major tasks: off-line characterization and on-line threshold detection. Off-line characterization is done using stochastic simulation over a set of all possible rates, λ_n and λ_o, to obtain the value of $ln(P_{max})$ that is sufficient to detect the change in rate. The results are accumulated in a histogram, and then the value of maximum likelihood ratio that gives very high probability that the rate has changed is chosen for every pair of rates under consideration. In our work we selected 99.5% likelihood.

On-line detection collects the interarrival time sums at run time and calculates the maximum likelihood ratio. If the maximum likelihood ratio computed is greater than the one obtained from the histogram, then there is 99.5% likelihood that the rate change occurred, and thus the CPU frequency and voltage need to be adjusted. We found that a window of $w=100$ is large enough in our experiments. Larger windows will cause longer execution times, while much shorter windows do not contain statistically large enough sample and thus give unstable results.

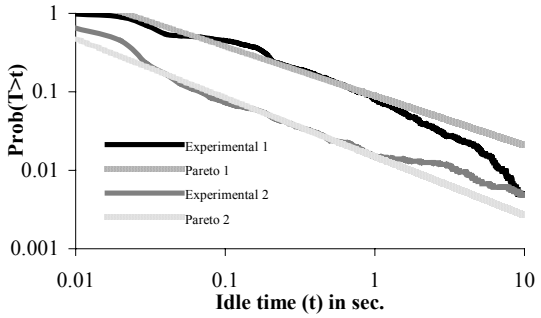

Figure 3. Experimental and Pareto Distributions

Request arrivals to empty buffer (idle state)

Once the buffer is empty, the first next arriving request defines the length of the idle time. The longer the idle time, the more savings we obtain by transitioning to a low-power state. Our measurements indicate that the distribution of workload idle times has to be modeled with the heavy-tailed distribution, such as the Pareto distribution. Similar conclusions based on traffic analysis for multimedia traffic in on-chip networks were reached in [27]. Figure 3 shows the log-log plot of the tail of the two experimental distributions collected by observing idle times in the communication packet arrivals over a period of two hours and the Pareto fits to each set of data. The top two lines represent the first set of experimental results and the corresponding Pareto fit, while the bottom two are the second set. Clearly, the characteristics of the two distributions are quite different since the usage patterns changed during the collection period. Previous work [13] assumed that the workload is stationary and then based on a priori analysis developed the optimal control. When the workload is not stationary, as shown by this example, the control developed in such a way will not be optimal. Thus it is important to be able to estimate Pareto parameters at run time, and then to recalculate the optimal control.

$$P_i(\Delta t_i > \Delta t) = b\Delta t_i^{-a} \quad (4)$$

The tail of the Pareto distribution with characteristic index a and normalizing constant b is shown in Equation 4. The tail of a distribution gives the probability that the idle time will be as long or longer the a given time.

$$a = -\frac{\sum_{i=1}^{N} \ln \Delta t_i \left(N \ln P_i - \sum_{j=1}^{N} \ln P_j\right)}{\sum_{i=1}^{N} \ln \Delta t_i \left(N \ln \Delta t_i - \sum_{j=1}^{N} \ln \Delta t_j\right)} \quad (5)$$

$$b = e^{\frac{1}{N}\left(\sum_{j=1}^{N} \ln P_j + a \sum_{j=1}^{N} \ln \Delta t_j\right)}$$

The parameters of Pareto distribution can be estimated using least-squares method on N samples of idle times Δt as

shown in Equation 5. Note that on the log plot (see Figure 3) Pareto distribution is a straight line with slope *a* and intercept *b*. On every new idle time sample, only the probability value, P_N, and the arrival time, Δt_N, need to be updated before recalculating parameters *a* and *b*.

Network Request arrivals

There are two types of network requests: activation (wakeup) and release (sleep) of a core. We model the network request arrivals with two exponential distributions fully defined by wakeup request arrival rate, λ_{netw}, and the network sleep request rate, λ_{nets}. The changes in network request rates can be tracked with the maximum likelihood ratio as was shown for core request arrival or servicing rates. The network requests can arrive at any point in time, but can only cause changes in core's behavior in two distinct situations. The wakeup request arrival causes core to wakeup immediately from a low-power state regardless of the service request arrival, where as a network sleep command transitions a core to sleep when idle, regardless of the decision of core's power manager. Thus, the network request arrivals only affect changes in decisions made regarding low-power state transitions. Any changes of network wakeup and sleep request arrival rates are tracked by the estimator at run-time as they have an effect on the final power management control.

B. Core parameters

Three main core parameters to estimate include: core frequency and voltage scaling characteristics, the time distribution for servicing incoming core requests and the characteristics of the low-power states, such as the transition times to/from each state. Some core characteristics can be determined at design time as they depend on hardware parameters alone, such as the number of core frequency and voltage settings. Other parameters need to be tracked at run-time, for example the properties of service time distributions.

Each NOC core has at least one main processor. Many of the today's processor have multiple active and low-power states. For example, each of the cores we characterize has one StrongARM processor as shown in Figure 1. The processor can be configured at run-time by a simple write to a hardware register to execute at one of eleven different frequencies. The number of frequencies is predefined at design time. We measured the transition time between two different frequency settings at 150 microseconds. For many applications (e.g. MPEG video or MP3 audio) this transition time is small enough that it does not cause any perceivable overhead. For each frequency, there is a minimum voltage the processor needs in order to run correctly, but with lower energy consumption. The minimum voltage can be obtained either from the datasheets, or by measurements.

Core servicing time is defined as the time it takes to process an incoming request to a core. For example, servicing time for MP3 core is defined as the time taken by the core to decode an incoming MP3 frame. The time it takes for a core to service requests in each of the active states follows an exponential distribution. Detailed measurement results have been presented in [13]. Although the general shape of the distribution remains the same, the servicing rate, (λ_{core}), changes depending on the current core state. Thus, one of estimator's jobs is to track the servicing rate changes at run time.

$$P_{uniform} = \begin{cases} \frac{t - t_{\min}}{t_{\max} - t_{\min}} & t_{\min} < t < t_{\max} \\ 0 & else \end{cases} \quad (6)$$

In addition to the active state, each core can support multiple lower power states, such as: *idle, sleep* and *off*. The core enters idle state as soon as all impeding requests are processed. The low-power state transitions are controlled by the power manager. The transition time between the active and one of the low-power states can be best described using the uniform probability distribution shown in Equation 6 where t_{min} and t_{max} are the minimum and the maximum transitions times. Typically, the more energy is saved in a given low-power state, the longer the average transition time to and from that state. The characterization of transition times between different power states can be done at design time, as it is only a function of hardware design parameters.

C. Buffering (or queue) behavior

Each core has a buffer associated with it that is used to store requests that have not been serviced yet. In this work we model the buffer using M/M/1 queuing model since both arrival and servicing rates in core's active state follow the exponential distribution. Thus, the queue is characterized by the number of requests pending service. For multimedia requests such as MPEG video and audio it is convenient to describe queue in terms of the number of frames waiting in the frame buffer. Detailed measurement results justifying M/M/1 queue model have already been published [13]. The estimator tracks changes in the queue size at run-time and feeds that information to the controller.

We do not use a queue to model the buffering of network requests, as such a buffer is not necessary. Network sleep request causes a core to transition to sleep immediately from an idle state and in any other state it is ignored. Network wakeup request starts the transition of core from a sleep to an active state. If core was in any state other than sleep, the wakeup request is disregarded. Thus, in both cases the network request either causes an immediate transition, or has no effect on the system.

D. Estimator summary

Table 1 summarizes main parameters and distributions modeled and tracked by the estimator. Workload model depends on the number of requests waiting in the buffer (or queue). When there is at least one request pending service in the queue, the estimator uses exponential distribution to model core's request arrivals. Any changes in request arrival rate, $\lambda_{workload}$, are detected with maximum likelihood ratio. As soon as queue is empty, an idle period begins. The estimator uses

Pareto distribution to model the length of the idle times. Two parameters of Pareto distribution, *a* and *b*, are estimated at run-time with least squares method. The time to transition between various low-power states is modeled by the uniform distribution. When core is actively processing requests pending in the queue, it is said to be in the active state. Two different parameters are observed at that time: core's rate of servicing requests, λ_{core}, and frequency vs. voltage curve characteristic. Clearly, service rate directly depends both on the core's frequency setting and on the type of requests serviced. The frequency vs. voltage characteristic determines the energy consumption and the performance for a particular type of requests. Finally, the network requests follow exponential distribution model with two different rates, λ_{netw} and λ_{nets}, for wakeup and sleep network requests. The estimator tracks changes in both rates using the maximum likelihood ratio. Given the system characteristics and parameters described above, we can now derive the full stochastic system model.

Table 1. System Characteristics

Component	State	Distribution	Parameters
Workload	Queue > 0	Exponential	$\lambda_{workload}$
	Queue = 0	Pareto	*a,b*
Core	Active	Exponential	λ_{core}, *f-V plot*
	Transition	Uniform	t_{min}, t_{max}
Network	Wakeup	Exponential	λ_{netw}
	Sleep	Exponential	λ_{nets}

V. STOCHASTIC SYSTEM MODEL

Each core can be modeled using Renewal model similar to the one presented in [13] for portable devices. In contrast to work presented in [13] , here we expand the system model to include voltage scaling, incoming network wakeup and sleep requests. As a result, we have a unified stochastic model that is guaranteed to give optimal power management decisions for systems with communicating cores. Thus, we obtain extra power savings because each core can transition to sleep as soon as other cores send it release command (instead of waiting for power manager's randomized timeout as in [13]). We can also wakeup a core from the sleep state preemptively instead of having to wait for the first request arrival and thus incurring the performance penalty due to the time taken to transition a core back into the active state. Finally, our expanded model integrates voltage scaling together with power management.

The expanded Renewal model for each core is shown in Figure 4. Each node shows the core's queue (local buffer) and power states, while each arc shows the transitions between the states and the conditions under which the transitions occur. Table 1 lists the distributions that model the transitions. We
highlighted our additions to the model presented in [13]. Now there are two ways that the core can transition to sleep: core sleep command and network sleep request. Core sleep command is given based on core's power management control, which waits for a randomized timeout value before starting the transition to sleep. Network sleep request causes an immediate transition to sleep from the idle state. Similarly, there are two ways to wake up a core: request arrival and network wakeup request. The wakeup request starts the transition to the active state even when there are no requests pending in the queue. Also, now we have a sequence of active states representing different core frequency and voltage settings.

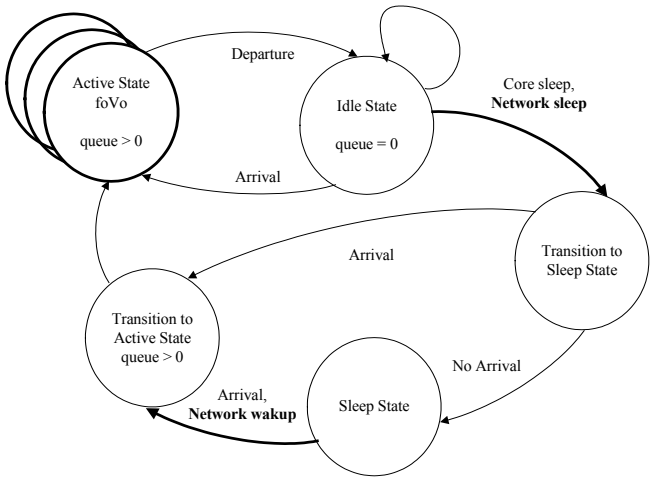

Figure 4. Renewal Model

Renewal theory defines a state called *renewal state*, in which the process statistically begins anew. The time between successive visits to renewal state is called *renewal time*, and one cycle from renewal state, through other states and then back is called a *renewal*. The main advantage of Renewal model is that it guarantees globally optimal results with very fast optimization time. When obtaining the optimal power management control, the complete cycle of transition from the idle state, through the other power states and then back into the idle state can be viewed as one renewal of the system. The problem of power management optimization is to determine the optimal distribution of the random variable t_{node_s} that specifies when the transition from the idle state to low-power state should occur based on the last entry into the idle state. We assume that t_{node_s} takes on values in $[0,\Delta t,\ldots, j\Delta t,\ldots N\Delta t]$, where j is an index, Δt is a fraction of the break-even time of the core and N is the maximum time before the core would got to a low-power state (usually set to an order of magnitude greater than the break-even time). Break-even time is defined as the minimum length of time a core should stay in a sleep state in order to recuperate the cost of transition to and from it. It is a function of hardware parameters only.

The solution to the optimization problem can be viewed as a table of probabilities (t_{node_s}), where each element *p(j)* specifies the probability of transition from idle to a low-power state indexed by time values $j\Delta t$. Similarly, we define the renewal time as *t(j)*, the energy spent as *e(j)*, and the performance penalty incurred during a transition as *d(j)*. Our starting point in the formulation of the optimization problem is the calculation of renewal time, *t(j)*.

$$\begin{aligned} t(j) = {} & E[t(j)I(t_{req} < t_{node\_s}, t_{net\_s} > t_{req}) \mid t_{node\_s} = j\Delta t] + \\ & E[t(j)I(t_{req} > t_{node\_s}, t_{net\_s} > t_{req}, t_{net\_w} > t_{req}) \mid t_{node\_s} = j\Delta t] + \\ & E[t(j)I(t_{req} > t_{node\_s}, t_{net\_s} > t_{req}, t_{net\_w} < t_{req}) \mid t_{node\_s} = j\Delta t] + \\ & E[t(j)I(t_{req} > t_{node\_s}, t_{net\_s} < t_{node\_s}, t_{net\_w} > t_{req}) \mid t_{node\_s} = j\Delta t] + \\ & E[t(j)I(t_{req} > t_{node\_s}, t_{net\_s} < t_{node\_s}, t_{net\_w} < t_{req}) \mid t_{node\_s} = j\Delta t] \end{aligned} \quad (7)$$

Figure 5 shows three different state diagrams we need to consider for calculation of the renewal time. The first two relate to node-centric power management approach, while the last one is specific to the network-centric approach. The expected renewal time calculations are shown right below the state transition diagrams. Note that while there is only one way to enter the idle state from the active state (*departure*, or completion of service request), there are two different ways to transition to sleep out of idle (node sleep, *node_s*, and network sleep, *net_s*, commands) and to active out of sleep state (request *arrival* and network wakeup command, *net_w*).

$$E[t(j)I(t_{req} < t_{node\_s}, t_{net\_s} > t_{req}) \mid t_{node\_s} = j\Delta t] = \sum_{k=0}^{j} \left(k\Delta t + \frac{1}{\lambda_{core} - \lambda_{workload}}\right) p(t_{req} = k\Delta t)\, p(t_{net\_s} > k\Delta t) \quad (8)$$

The calculation of the expected renewal time is shown in Equation 7. The five terms in this Equation mirror all possible transitions shown in Figure 5.

Since all four possible events: time of the request arrival (t_{req}), network sleep command (t_{net_s}), network wakeup command (t_{net_w}) and node sleep command (t_{node_s}) are independent from each other, the actual calculation of each of the five terms is simplified. As an example, we show in Equation 8 a detailed calculation of the expected renewal time described in condition i) for our expanded model. Note that in this particular case network wakeup command condition is not needed since the system never goes to sleep. This equation states that the expected renewal time is a sum of time spent in the idle state ($k\Delta t$), with the time needed to work off the request that arrived while in the idle state ($1/(\lambda_{core}-\lambda_{workload})$), weighed by the probability that node request arrives at time $k\Delta t$, $p(t_{req}= k\Delta t)$, and the network and node sleep command arrive after that ($p(t_{net\_s} > k\Delta t)$ and condition $t_{node\_s}= j\Delta t$ expressed in the top limit on the summation). All other terms in the renewal time calculation are obtained in the similar manner.

$$E[e_{idle}(j)I(t_{req} < t_{node\_s}, t_{net\_s} > t_{req}) \mid t_{node\_s} = j\Delta t] = P_{idle} \sum_{k=0}^{j} k\Delta t \;\; p(t_{req} = k\Delta t)\, p(t_{net\_s} > k\Delta t) \quad (9)$$

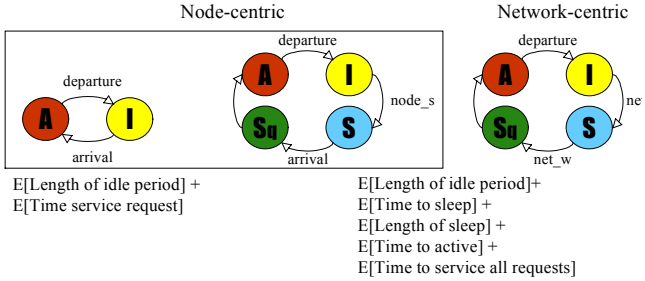

Figure 5. Renewal Time Calculation

Evaluation of energy and performance penalty are directly derived from the expressions obtained for the renewal time. For example, if we want to know how much energy is spent in the idle state, $e_{idle}(j)$, for the condition i) of our expanded model, we would just use the first term in Equation **8** to obtain the expected time spent in the idle state and weigh it with the power consumed while idling, P_{idle}. Equation **9** shows that calculation. In a similar manner we can obtain performance penalty, but instead of using the power consumption, we just use the expected delay overhead. Note that other QoS constraints can be added to this problem, such as jitter, much in the same manner as we calculated energy and delay overhead.

Once we have calculated the expected renewal time, energy and performance penalty, we are ready to start the optimization. The formulation of control optimization for Renewal model is shown in Equation 10, where *p(j)* is the probability of transitioning into low-power state after the system has been idle for time $j\Delta t$, *d(j)* is the expected performance penalty, *t(j)* is the expected time until renewal, *e(j)* is the expected energy consumed, and P_{constr} is the user-defined power constraint. To simplify our discussion of the fast optimization method in the next Section, we also define p, a, b, and c $\in R^n$ as follows: p is the probability we are solving for, a represents performance penalty (*d*), b is the expected renewal time (*t*), and c is the energy balance equation, $e-tP_{constr}$. An open-loop optimization problem similar to this one has already been solved for portable systems in [13] by using a linear program solver. The optimal control is obtained in tens of seconds, which is much too long for implementation of the closed-loop power management control presented in this work. Thus, we next we describe a new, fast optimization algorithm capable of solving the same problem in a matter of milliseconds.

$$\begin{aligned} \textbf{min} \quad & \frac{a^t p}{b^t p} = \frac{\sum_j p(j)d(j)}{\sum_j p(j)t(j)} \\ \textbf{s.t.} \quad & c^t p = \sum_j p(j)[e(j) - t(j)P_{constr}] = 0 \\ & 1^t p = \sum_j p(j) = 1; \quad p(j) \geq 0 \quad \forall j \end{aligned} \quad (10)$$

The problem in Equation 10 would be homogeneous except for the constraint $1^t p=1$. In other words, if we replace the vector p with $p'= \alpha p$, where $\alpha>0$, then p' satisfies $a^t p' / b^t p' = a^t p / b^t p$ (i.e, has the same objective value as p), $c^t p'= 0$, and

Table 2. Sample controller

Source	Idle Time (ms)	Transition Probability
Node No arrival	0	0.0
	70	0.3
	120	1.0
Arrival	Any time	1.0
Network	Any time	1.0

$p' \geq 0$. It follows that we can replace the normalization $1^t p=1$ with any other, such as, for example, $b^t p=1$. This replacement is possible since we can assume that a and b parameters are always greater than zero componentwise because they represent positive quantities. This leads to the problem:

$$\begin{aligned} \max \quad & -v \\ s.t. \quad & uc + vb + a \geq 0 \end{aligned} \tag{12}$$

$$\begin{aligned} \min \quad & a^t p' \\ s.t. \quad & c^t p' = 0 \\ & b^t p' = 1;\ p' \geq 0 \end{aligned} \tag{11}$$

which is equivalent to Equation 10 in the following sense: if

$$\max \quad f(u) = \min_i (a_i + uc_i)/b_i \tag{13}$$

p'^* is the solution of Equation 11, then $p^*=(b^t p'^*/ \mathbf{1}^t p'^*)p'^*$ is the solution of Equation 10.

We work with the problem in Equation 11, which is a linear program (LP) with n variables, and two constraints. By the basic theorem of linear programming, there is always a solution which has only two nonzero entries. Therefore the original problem, in Equation 10 has the same property: there is always a solution p^* which has only two nonzero entries.

To solve the LP in Equation 11 efficiently, we consider its dual problem, which is LP shown in Equation 12. The problem can further be reformulated as a simple, unconstrained maximization problem that is a function of only one scalar variable, u, as shown below.

The function f(u) is piecewise linear and concave. In other words, we seek two indices, k and l such that:

$$f(u) = (a_k + uc_k)/b_k = (a_l + uc_l)/b_l \tag{14}$$

with $b_k/c_k \geq 0$ and $b_l/c_l \leq 0$. These two indices allow us to solve the primal LP in Equation 11 since these two indices can be taken as the nonzero indices in the optimal p'. A solution of the primal LP in Equation 10 can be found by setting $p'_j=0$, except for j=k and j=l. We then solve the linear equations $c^t p'=0$, $b^t p'=1$, to find the optimal p'.

Thus, obtaining the optimal result to the original problem reduces to solving the dual problem Equation 11, which is a single variable unconstrained optimization problem. This can be done several ways. The simplest is to use bisection to find the optimal u (and more importantly, the optimal indices k and l). The selection of the initial bracketing values should be done depending on the problem characteristics. When optimizing NOC power management control, the initial bracketing values are determined based on core characteristics (e.g. break even time). The optimization is triggered by the estimator when any of the system parameters change. The final output of optimization is a table that specifies probabilities of transitioning a core into each of the low-power states. An example of optimal control is shown in Table 2.

VI. CONTROLLER

The controller's job is to give commands to the core regarding both power management and voltage scaling. Power management decisions determine when to transition into a low-power state, and which of the available low-power states to transition to. Dynamic voltage scaling implies a selection from one of the active states at run-time depending on the estimator's feedback.

The dynamic power management control can be accessed from either software or hardware, depending on how power management controller is realized. The software implementation of the controller can be described as follows. The controller generates a pseudo-random number when the core becomes idle. The core remains idle until either the probability of transition to the low-power state is greater than the random number generated, or until workload arrival forces the core's transition into the active state. When the core is in the low-power state, it stays there until the first arrival, at which point it transitions back into the active state. Arrival of network sleep command overrides node-centric control only in the idle state before the transition to sleep has started. Similarly, network wakeup command causes the core to transition to active state only if it has been in a sleep state with no requests pending. In all other cases the optimal control behaves the same way as if was only node-centric.

Controller sets the new processor frequency and voltage when either the incoming workload arrival rate ($\lambda_{workload}$) or the core's servicing rate (λ_{core}) change. Changes in both rates are tracked and computed by the estimator using maximum likelihood ratio. Often the relationship between servicing rate and processor frequency is fixed for a given application, and thus needs to be estimated only once per each new application. Thus, run-time estimation is primarily done for the core's workload incoming rate.

$$Delay = \frac{\lambda_{core}}{\lambda_{workload}\ (\lambda_{workload} - \lambda_{core})} \tag{15}$$

The controller uses results of M/M/1 queuing theory to obtain the appropriate control for dynamic voltage scaling since both the workload arrivals and service times follow an exponential distribution. The goal of the controller is to set the voltage and frequency of the processor for the newly estimated rate so that the processing delay shown in Equation 15, and thus the number of tasks to be processed in the buffer, are kept constant. For example, if the arrival rate for MP3 audio changes, Equation 15 is used to obtain required decoding rate in order to keep the frame delay (and thus performance) constant. When both workload and core rates

are changing, the change detected first is adapted to first.

VII. RESULTS

The power management methodology presented in this work is implemented for the sample NOC system shown in Figure 1. The system consists of four large cores: communication, speech processing, MPEG audio and video core. Power and performance characteristics of each core are shown in Table 3. Three power states are supported by each core: active, idle, and sleep. The transition time from active to sleep and back to active state (shown in Table 3 as *A-S-A time*) is on the order of tens of milliseconds, which is slow enough to allow for dynamic parameter estimation and periodic control recalculation. Number of DVS settings reflects the discrete frequency and voltage points each cores processing unit can be set to. The transition time needed to change from one to other frequency point is on the order of hundreds of microseconds (labeled as *DVS switch time*).

Table 3. NOC Specifications

Specification	Audio	Video	Comm.	Speech	Total
Active P(mW)	700	1885	1500	1055	5140
Idle P(mW)	216	235	1000	208	1659
Sleep P(mW)	0.3	1.4	100	0.6	102.2
A-S-A time(ms)	45.6	54.6	40	54.6	54.6
# DVS Settings	11	11	3	11	11
DVS switch (us)	150	150	100	150	150

Each core in NOC has a power manager, that in turn consists of an estimator and a controller. Estimators job is to estimate the parameters needed to recalculate optimal control depending on the changes in the core's environment. The environment includes incoming traffic from the chip network, and special power management requests from other cores. The controller implements the optimal system control. The results highlight the quality of the estimators, followed by the controller implementation in hardware. Lastly, energy savings are contrasted when using only the node-centric approach with the combined node and network-centric power management.

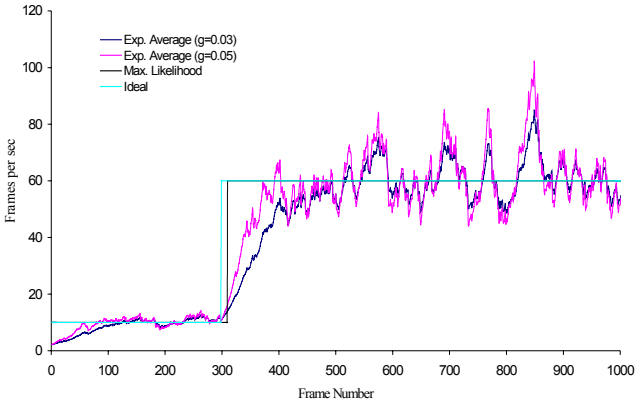

Figure 6. Arrival rate estimation

A. Estimator

There are two core states in which power management decisions are made. Appropriate service level is determined in the active state (non-empty buffer), while the decision on when to transition into the sleep state occurs in the idle state (empty buffer). The quality of both decisions depends on the estimation accuracy and speed. We first evaluate the estimation of the request arrivals to a non-empty buffer, followed by the estimation of the arrivals to an empty buffer.

Exponential distribution is used to model arrival times to a non-empty buffer (or buffer with requests pending service), service times of each core, the network wakeup and sleep request arrivals. Each of these distributions is characterized with a rate: $\lambda_{workload}$, λ_{core}, λ_{net_w}, and λ_{net_s}. Changes in these rates can be due to many different factors, ranging from changes of the type of workload, to different conditions in the wireless medium. The estimator uses maximum likelihood ratio shown in Equation 3 to detect a change in any of the exponential distribution rates. An important advantage of maximum likelihood ratio is that it guarantees optimal estimation results with relatively small computation cost.

Since the estimation is done in the same way for each of the four exponential distribution rates, we present results for estimating when incoming arrival frame rate in the MPEG video core changes from 10 frames/sec to 60 frames/sec. The maximum likelihood ratio approach is compared to both ideal and exponential average detection. The new exponential average rate, λ_{ave}^{new}, is calculated using a current estimate for the rate, λ_{cur}, and weighing it against the average value computed to date, λ_{ave}^{old}, with a gain parameter g:

$$\lambda_{ave}^{new} = (1-g)\lambda_{ave}^{old} + g\lambda_{cur} \qquad (16)$$

Figure 6 shows the results of estimation. The maximum likelihood algorithm detects the exact change in rate. Slight delay is due to a number of samples needed before the change can be detected. The effect of this delay is very minor to the overall power management control. It is very close to ideal detection which knows ahead of time when the rate will change. In contrast, the exponential average detection for two different values of gain shows very delayed and unstable detection characteristics. The closer detection is to the exact time change in rate occurs, the more unstable exponential average detection becomes. Since the computational overhead of maximum likelihood detection is about the same as with the exponential detection, clearly our approach to estimation for exponential rate changes is the better one to use.

Distribution of the length of time spent in the idle state, modeled by the Pareto distribution, also needs to be estimated at run time, as it is one of the most important parameters determining the quality of the power management control. Our estimator tracks changes in two critical parameters of the the distribution, a and b, as shown in Equation 4. Since Pareto distribution follows a straight line on a log plot (see Figure 3), we can use the least squares method to find the line's slope and intercept. Figure 7 shows how the estimator detects a change in the idle time distribution on a communications core when the traffic pattern changes between two examples shown in Figure 3. The change is most clearly observed in the intercept value. The Pareto parameter estimates are very fast and accurate, with average error of only 2%.

B. Controller

The power manager's controller can be realized in software, hardware or a combination of the two. When critical parameters change very often, control and estimation should be realized in software. Realizing a part of, or the whole controller in hardware lowers the control overhead, with very minor additions to an already existing hardware power manager (e.g. ARM cores) or an on-chip FPGA. This approach is very attractive especially for cores where the control does not change much at run time, and thus does not need to be recomputed very often. Since the software implementation has already been discussed, we focus on the hardware implementation next.

There are three different components to the optimal controller: the random number generator, the control and the timer. The timer is used to measure the length of idle period before the control is evaluated. Typically core's processors already have programmable timers aboard that can be used by the hardware controller. The simplest hardware implementation for random number generator is to use Linear Feedback Shift Register (LFSR).

Table 4. Local PM Control FPGA Synthesis Results

LFSR	LFSR Regs		Control	
Bits	# LABs	Max ns	# LABs	Max ns
5-15	1	4	2	35

Results of controller synthesis into Altera's EPM7032 FPGA are shown in Table 4. Control and LFSRs take up 3 Logic Array Blocks (LABs) for LFSR sizes ranging from 5 to 15 bits. We found through simulation that even with as little as 8 bits, the hardware LFSR gives results within 5% of optimal. In addition, the time it takes to arrive at the decision is in the nanoseconds, while the minimum idle times the power manager would respond to are in milliseconds. Controller is even faster when synthesized into gates with Synopsis as shown in Table 5. The LFSR area is consistently about 12-14% of the total area. Even the largest design takes only 15 registers and 855 gates.

Table 5. Local PM Control Synopsis Synthesis Results

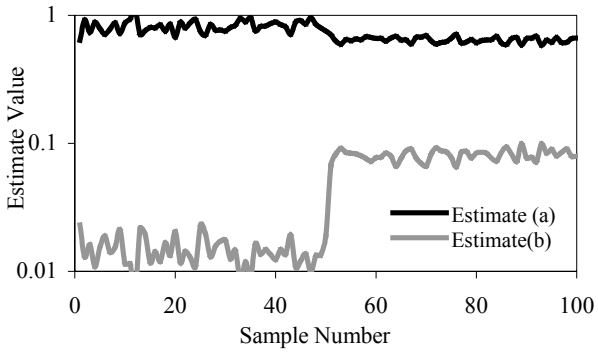

Figure 7. Dynamic Pareto Parameter Estimates

LFSR Regs		Control	
#FFs	% area	#gates	% area
5	14%	193	86%
9	14%	417	86%
15	12%	855	87%

C. Node-centric DVS and DPM

In this section we present results of node-centric dynamic voltage scaling and power management with no consideration for network requests. We start by comparing the energy savings controller can obtain when implementing DVS on MP3 audio core based on the results of (i) the ideal detection algorithm, (ii) the exponential average approximation and (iii) the maximum processor performance to (iv) the maximum likelihood algorithm presented in this paper. For this purpose we combined six audio clips (labeled A-F) totaling 653 seconds of audio, each running at a different set of bit and sample rates. For all sequences, the frame arrival rate varies between 16 and 44 frames/sec. During decoding, the estimator detects changes in both arrival and decoding rates for the MP3 audio sequences, and the controller responds by adjusting the processor frequency and voltage. The resulting energy (kJ) and average total frame delay (s) are displayed in Table 7. Our controller, which relies on maximum likelihood estimation, has results very close to the ideal detection in terms of both performance and energy savings. The maximum average frame delay of 0.1 s corresponds to 6 extra frames of audio in the buffer.

Table 6. Node-centric DPM and DVS

Algorithm	Energy (kJ)	Factor
None	4260	1.0
DVS	2663	1.6
DPM	2506	1.7
Both	1556	2.7

We implemented DVS for two different video clips as well. The arrival rate varies between 9 and 32 frames/second. The ideal detection algorithm allows for 0.1s average total frame delay equivalent to 2 extra frames of video in the buffer. Energy (kJ) and average total frame delay (s) are shown in Table 8. The exponential average shows poor performance and higher energy consumption due to its instability (see Figure 6). The controller that uses results of our maximum likelihood estimator performs well, with significant savings in energy.

Next, use a sequence of audio and video clips, separated by idle time to study the tradeoffs when using DVS and DPM on NOC. Table 6 contrasts system energy savings obtained with only dynamic voltage scaling implemented, followed by only power management and finally also for the combination of the two approaches. We obtain savings of a factor of 2.7 when expanding the power manager to include dynamic voltage scaling with our change point detection algorithm.

D. Network-centric Power Management

In this section we contrast power savings obtainable when using only node-centric power management, with additional savings we can get when implementing network-centric approach. Table 9 shows energy savings obtained for the NOC shown in Figure 1, with specifications listed in Table 3. The results were obtained by simulating the power states of the NOC system as a whole, with real workload traces collected from each respective core as an input to the simulator. The results report a factor of savings in energy for both node and network centric approaches with reference to not using any power management. The lightly shaded portion of the table reports corresponds to only node-centric approach, while the darker shaded row is for a combined node and network-centric approaches.

Table 9. Energy Savings for PM in NOCs

PM	PM Type	MP3	MPEG2	Comm.	Speech	Total
None	None	1.0	1.0	1.0	1.0	1.0
Node Centric	DVS only	1.4	2.0	1.0	3.8	1.2
	DPM only	2.0	1.5	3.0	1.5	2.4
	DVS&DPM	2.8	3.0	3.0	5.8	3.0
Network	DVS&DPM	3.7	3.6	4.2	6.1	4.1

In node-centric PM, controlling only processing frequency and voltage at run time (DVS results) gives between a factor of 1.4 to a factor of almost four in savings pre core. Note that communications core does not allow voltage and frequency scaling. When only control of transition into the sleep state is implemented (*DPM only* results), savings range from a factor of 1.5 to a factor of 3. The smallest savings are in video core, as it tends to have very few idle times. Combining the DVS and DPM gives the overall savings of a factor of 3.6.

When network power management is included with the node-centric approach (the last row in Table 9), the savings in energy grow to a factor of 4.1 with performance penalty reduced by a minimum 15%. The performance penalty of a core is the time the rest of the system has to wait in order for the core to become available after either changing processing frequency or waking up from the sleep state. These savings show that using information about system state as it becomes available (network wakeup and sleep requests) can significantly enhance the quality of the power management results. There are quite a few situations where such information is available. For example, when MP3 decoder starts, it can immediately inform the communication core that its services will be needed. Thus by the time MP3 initializes all of its data structures, the communication core transitions from sleep state into the active state. In this way no performance penalty is incurred due to the transition, and

Table 7. MP3 audio DVS

MP3 Sequence	Result	Ideal	Max Lik.	Exp. Ave.	Max
ACEFBD	Energy (kJ)	196	217	225	316
	Fr. Delay (s)	0.1	0.09	0.1	0
BADECF	Energy (kJ)	189	199	231	316
	Fr. Delay (s)	0.1	0.09	0.1	0
CEDAFB	Energy (kJ)	190	214	232	316
	Fr. Delay (s)	0.1	0.04	0.1	0

Table 8. MPEG video DVS

MPEG Video Clip	Result	Ideal	Max Lik.	Exp. Ave.	Max
Football (875s)	Energy (kJ)	214	218	300	426
	Fr. Delay (s)	0.1	0.11	0.16	0
Terminator2 (1200s)	Energy (kJ)	280	294	385	570
	Fr. Delay (s)	0.1	0.11	0.16	0

communication core was able to save power by staying asleep as long as possible. In situations where such information is not available (node-centric approach) our closed-loop power management approach still gives large savings.

VIII. CONCLUSIONS

This work presented a new methodology for managing power consumption in NOCs. The power management optimization is formulated using closed loop control concepts, with blended node and network centric approaches. The first component of our power management system is an estimator that is capable of fast and accurate tracking of system changes. The expanded Renewal model integrates network centric power management with voltage scaling and node centric power management. It enables the formulation of the optimization problem that is guaranteed to be globally optimal. The optimization is done using our new fast optimization method, which is orders of magnitude faster than methods used in the past. Lastly, we presented a controller implementation that manages both DVS and DPM.

The new methodology is tested on a design of a NOC system consisting of four satellite units, each with the local power manager consisting of the estimator and the controller.

The estimator implementation has been shown to have average error of 2% when estimating Pareto parameters, and is right on target when estimating exponential frame arrival rate changes. Our fast optimization algorithm recalculated control in a matter of milliseconds when the distribution parameters change. The final implementation of node and network centric power management approaches shows savings of a factor of four at system level while improving performance.

ACKNOWLEDGMENT

Many thanks to my HP Labs colleagues, Mat Hans, Brian Delaney, and Andrea Acquaviva, for their help with this work.

REFERENCES

[1] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, A. Sangiovanni-Vincentelli, ``Addressing the System-on-a-Chip Interconnect Woes through Communication-Based Design,'' *Design Automation Conference*, pp. 667-672, 2001.

[2] W. Dally, B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Design Automation Conference,* pp. 684-689, 2001.

[3] P. Guerrier, A. Greiner, "A Generic Architecture for On-Chip Packet-Switched Interconnections," *Design, Automation and Test in Europe,* pp. 250-256, 2000.

[4] K. Goossens, J. Merbergen, A. Peeters, P. Wielage, "Networks on Silicon: Combining Best-Effor and Guaranteed Services," *Design, Automation and Test in Europe,* pp. 423-426, 2002.

[5] International Technology Roadmap for Semiconductors: 2001.

[6] T. Ye, L. Benini, G. De Micheli, "Analysis of Power Consumption on Switch Fabrics in Network Routers," *Design Automation Conference*, pp. 600-605, 2002.

[7] L. Benini, G. De Micheli, "Powering Networks on Chips," *International Symposium on System Synthesis,* Invited Talk, 2002.

[8] M. Wan, H. Zhang, V. George, M. Benes, A. Abnous, V. Prabhu, J. Rabaey, "Design Methodology of a Low-Energy Reconfigurable Single-Chip DPS System," *Journal of VLSI Signal Processing,* 2000.

[9] "SA-1110 Microprocessor Technical Reference Manual," Intel, 2000.

[10] L. Benini, G. Paleologo, A. Bogliolo and G. De Micheli, ``Policy Optimization for Dynamic Power Management'', *IEEE Transactions on Computer-Aided Design*, vol. 18, no. 6, pp. 813-833, June 1999.

[11] E. Chung, L. Benini and G. De Micheli, ``Dynamic Power Management for non-stationary service requests'', *Design, Automation and Test in Europe*, pp. 77-81, 1999.

[12] Q. Qiu, Q. Wu, M. Pedram, "Dynamic Power Management in a Mobile Multimedia System with Guaranteed Quality-of-Service," *Design Automation Conference*, pp. 701-707, 2001.

[13] T. Simunic, L. Benini, P. Glynn, G. De Micheli, "Event-driven Power Management," IEEE Transactions on CAD, pp.840-857, July 2001.

[14] T. Simunic, S. Boyd, "Managing Power Consumption in Networks on Chips," *Design, Automation and Test in Europe*, pp. 110-116, 2002.

[15] M. Weiser, B. Welch, A. Demers, S. Shenker, ``Scheduling for reduced CPU energy,'' *Symposium on Operating Systems Design and Implementation* pp.13-23, Nov. 1994.

[16] K. Govil, E. Chan, H. Wasserman, ``Comparing algorithms for Dynamic speed-setting of a low-power CPU,'' *International Conference on Mobile Computing and Networking*, Nov. 1995.

[17] F. Yao, A. Demers, S. Shenker, ``A scheduling model for reduced CPU energy,'' *IEEE Foundations of computer science*, pp.374-382, 1995.

[18] I. Hong, M. Potkonjak, M. Srivastava, ``On-line Scheduling of Hard Real-time Tasks on Variable Voltage Processor,'' *International Conference on Computer-Aided Design*, Nov. 1998.

[19] T. Ishihara, H. Yasuura, ``Voltage Scheduling Problem for dynamically variable voltage processors,'' *IEEE International Symposium on Low Power Electronics and Design*, pp.197-202, 1998.

[20] Y. Shin, K. Choi, ``Power conscious fixed priority scheduling for hard real-time systems,'' *Design Automation Conference*, pp.134-139, 1999.

[21] S. Lee, T. Sakurai, ``Run-time voltage hopping for low-power real-time systems,'' *IEEE International Symposium on Low Power Electronics and Design*, pp.806-809, 2000.

[22] Y. Zhang, X. Hu, D. Chen, ``Task Scheduling and Voltage Selection for Energy Minimization,'' *Design Automation Conference*, pp.130-135, 2002.

[23] G. Quan, X. Hu, "Minimum Energy Fixed-Priority Scheduling for Variable Voltage Processors," *Design, Automation and Test in Europe*, pp. 782-787, 2002.

[24] D. Rakhmatov, S. Vrudhula, C. Chakrabarti, ``Battery-Conscious Task Sequencing for Portable Devices Including Voltage/Clock Scaling,'' *Design Automation Conference*, pp.333-337, 2002.

[25] T. Pering, T. Burd, R. Brodersen, ``Voltage scheduling in the IpARM microprocessor system',' *IEEE International Symposium on Low Power Electronics and Design*, pp.96-101, 2000.

[26] A. Bavier, A. Montz, L. Peterson, ``Predicting MPEG Execution Times,'' *SIGMETRICS*, pp.131-140, 1998.

[27] G. Varatkar, R. Marculescu, ``Traffic Analysis for On-chip Networks Design of Multimedia Applications,'' *Design Automation Conference*, pp.402-407, 2002.

[28] S. Eliason, *Maximum Likelihood Estimation*, Sage Publications, 1993.

[29] S. Boyd, L. Vandenberghe, *Convex Optimization*, Lecture Notes, Stanford University, Winter 2001.