Managing Power Consumption in Networks on Chips

Tajana Simunic
HP Labs & Stanford University
1501 Page Mill Rd., MS 3U-4
Palo Alto, CA 94304
(650) 236-5537
tajana@stanford.edu

Stephen Boyd
Stanford University
Packard 264
Stanford, CA 94305
(650) 723-0002
boyd@stanford.edu

ABSTRACT
Systems on a chip (SOCs) are rapidly evolving into larger networks on a chip (NOCs). This work presents a new methodology for managing power consumption for NOCs. Power management problem is formulated using closed-loop control concepts, with the estimator tracking changes in the system parameters and recalculating the new power management policy accordingly. Dynamic voltage scaling and local power management are formulated in the node-centric manner, where each core has its local power manager that determines units power states. The local power manager's interaction with the other system cores regarding the power and the QoS needs enables network-centric power management. The new methodology for power management of NOCs is tested on a system consisting of four satellite units, each with the local power manager capable of both node and network centric power management. The results show large savings in power with good QoS.

1. INTRODUCTION
Today's systems-on-a-chip (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. As more and more cores are integrated into a single chip, it is becoming increasingly difficult to meet the design constraints while still using the old design methodologies for SOC designs. Shared-medium busses that are used today do not scale well, and do not fully utilize potentially available bandwidth. 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 relatively 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. This trend only worsens as the clock frequencies increase and the features sizes decrease. Lowering the power supplies and designing smaller logic swing circuits decreases the overall power consumption at the cost of higher data errors.

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 [1,2,4]. 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. Regularity enables design modularity, which in turn provides a standard interface for easier component reuse and better interoperability. 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 using a closed-loop control model with a combination of node and network centric power management approaches. Each communicating core has its local power manager that consists of an estimator and a controller. The estimator observes 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). Based on the changes detected, it recalculates the optimal control. The optimal controller selects the appropriate power and performance states of the local core.

The rest of the paper is organized as follows. Section 2 discusses related work in both NOC design, and system level power management areas. The details of the system-level power management implementation for NOCs is discussed in Section 3. Sample design of a power management system for NOC is presented in Section 4, along with experimental results. Finally, the Section 5 summarizes the contributions of this work.

2. 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. In current SOC designs AMBA [10] and CoreConnect [11] standards have been used for point-to-point connections on chip. The Virtual Socket Interface (VSI) alliance [6] defines a standard interface to be used in conjunction with the on-chip system buses, for point-to-point connections between the high performance virtual components (VCs) or on-chip buses. Glue logic is needed to interface predesigned cores and busses to VSI interface. Each core in the Sonics [5] MicroNetwork communicates with an agent in the Silicon Backplane using an Open Core Protocol (OCP).

DATE '02, March 1-3, 2002, Paris, France.

Agents in turn communicate with each other using network protocols via TDMA rotating priority access control system. Cosy [7] defines interfaces at multiple levels of abstraction. At the lower level it uses the Virtual Component Interface to adopt to the specific physical bus protocol.

More recently, there have been a few publications that define the NOC architecture based on the packet communication model. 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]. Reservation of network resources such as buffers and bandwidth is done with flit-reservation flow control. Higher level protocols are layered on top of the simple interface.

The communication layers in NOCs can be partitioned much like the structure proposed by OSI Reference Model for the computer networks in [1,4]. MESCAL provides tools for correct-by-construction protocol stack [1]. According to the Metropolis methodology, the SOC designer first describes or selects blocks that perform computations and then designs the communication among them using a successive refinement process [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. System-level power management is already a well known concept for larger systems, such as laptops. Many of the cores that are of interest in NOC design already have multiple power and performance states. For example, StrongARM processor [9] supports four power states (active, idle, sleep, off) and a set of eleven power-performance tradeoff states characterized by different core voltages and frequencies of operation while in the active state. An outline of possible approaches for energy savings in NOCs is presented in [4]. 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 the closed-loop control model.

The most commonly implemented power management policy at the system level is a *timeout policy* that transitions system components into low-power states when they are inactive for a preset amount of time. Predictive policies developed for interactive terminals [12,13] force the transition to a low power state as soon as a component becomes idle if the predictor estimates that the idle period will last long enough. Both timeout and predictive policies are heuristic in nature, and thus do not guarantee optimal results. In contrast, approaches based on stochastic models can guarantee optimal results. Stochastic models to date have been formulated with open-loop control model, where statistics of the system are collected and characterized ahead of time, and the control is derived based on those with no adaptation at run time[14, 16,17]. An exception is the adaptive approach presented in [15] that uses only memoryless distributions to describe the history-dependent system behavior.

In addition to transitioning components into low-power states during idle times, power manager can also adjust processing frequency and voltage in the active state (Dynamic Voltage Scaling – DVS). Early DVS algorithms set processor speed based on the processor utilization of fixed intervals [18,19]. The approaches presented in [20,21,22] for real-time systems assume that all tasks run at their worst case execution time. The workload variation slack times are exploited on task-by-task basis in [23,24]. Voltage scheduler at the task level is presented in [25]. All DVS algorithms described so far set processor voltage and frequency on task basis. Algorithm presented in [26] adjusts optimally to the workload variation *within* tasks.

This work blends the node and network centric approaches for managing the power consumption in NOCs, while at the same time introducing for the first time a stochastic **closed-loop** control model. More details regarding the design of the power management system for NOCs are discussed in the next section.

3. POWER MANAGEMENT IN NOCs

Networks on chips consist of a set of cores connected with the communication network. As the chip sizes increase relative to the feature sizes, the data communication becomes inherently unreliable, as discussed in the Introduction. As a result, deterministic design methodology used in today's designs needs to be replaced by statistical modeling.

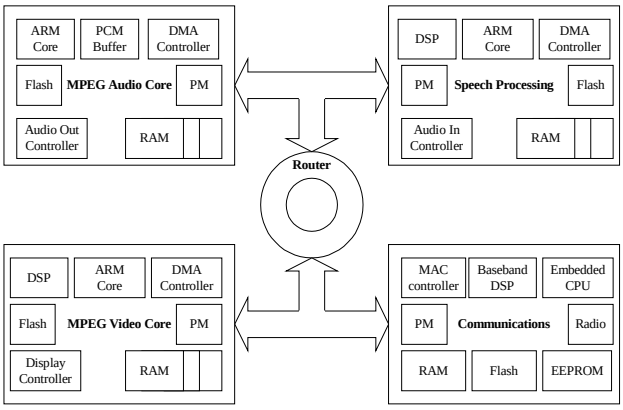

Figure 1. Network On a Chip

A statistical model of NOCs can be used as the basis for optimization of power consumption under QoS constraints. Figure 1 shows a sample NOC. The NOC can be modeled as a queuing network with a number of service points representing cores. Each core can be modeled using Renewal model much like the one presented in [17] for portable devices. Renewal theory studies stochastic systems that have 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*. In policy optimization for dynamic power management, 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 main advantage of Renewal model is that it guarantees globally optimal results with very fast optimization time.

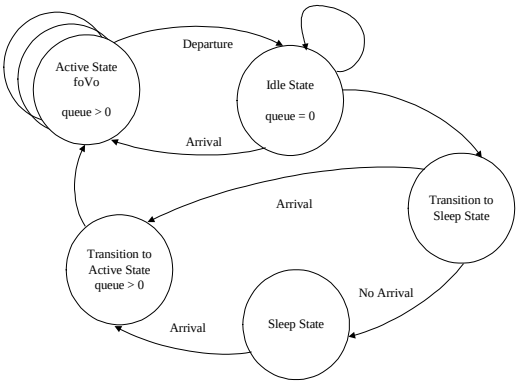

Figure 2. Renewal Model

The Renewal model for each core is shown in Figure 2. Incoming workload is represented with arrival arcs, while core's service is represented with departure arcs. Each node shows the core's queue (local buffer) and power states. Table 1 lists the distributions that model the transitions between the states shown in Figure 2. Detailed measurement results supporting the model have been presented in [17]. In the idle state, when the queue is empty, Pareto distribution (see Equation 3) describes the expected length of the idle time. As soon as a workload request arrives, the core enters active state and the remaining arrival and service times are then governed by the exponential distribution. The transition times between core's active, idle and sleep states are governed with the uniform distribution that does not change at run time, as it is predetermined by the core's design characteristics

Table 1. System Model

Component	State	Distribution
Workload	Queue > 0	Exponential
	Queue = 0	Pareto
Core	Active	Exponential
	Transition	Uniform

The model described so far does not express how to manage energy and QoS. Management of energy consumption under QoS constraints can be formulated as a closed-loop stochastic control problem. This is in contrast to previous work [14,16,17] where power management policies are obtained by solving an open-loop control problem. Control theory defines three different entities in a closed-loop control system: a system under control, an estimator and a controller. Power manager, as shown in Figure 3, contains the controller, and the estimator. The power manager's controller 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. The estimator observes the requests coming into the core's queue (*Core Traffic* in Figure 3), the state of the core and the incoming power management requests from the network (*Network PM Request* in Figure 3). Based on the observations, it estimates the parameters needed to recalculate the power management policy and thus closes the control loop. The next sections discuss how estimation and control are formulated for both node and network centric portions of NOC power management.

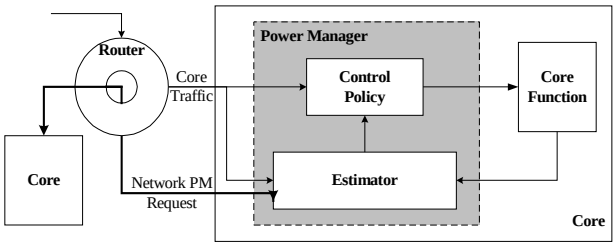

Figure 3. Local Power Manager

3.1 Node-centric power management

This section presents first the estimator and then the controller implementations for node-centric power management. Good estimation is most critical for modeling the workload and service behavior in the active and the idle states since their distribution's characteristics can change at run time.

In the active state, the workload is modeled using exponential distribution. Both interarrival ($\lambda_{workload}$) and servicing (λ_{core}) rate changes can be tracked using the log of maximum likelihood estimator shown in Equation 1. 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.

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

The new estimated rate is used to set the voltage and frequency of the processor so that the processing delay shown in Equation 2, and thus the number of tasks to be processed in the buffer, are kept constant. Typically the workload servicing rate's (λ_{core}) relationship to processor frequency is fixed for a given application, and thus needs to be estimated only once per each new application. Run-time estimation is primarily done for the core's workload incoming rate ($\lambda_{workload}$).

$$Delay = \frac{\lambda_{core}}{\lambda_{workload}\ (\lambda_{workload} - \lambda_{core})} \qquad (2)$$

The distribution of workload idle times has to be modeled with the heavy-tailed distribution, such as the Pareto distribution. Figure 4 shows the log-log plot of the tail of 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 [17] assumed that the workload is stationary and then based on a priori analysis developed the optimal policy. When the workload is not stationary, as shown by this example, the policy 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 recompute the optimal policy.

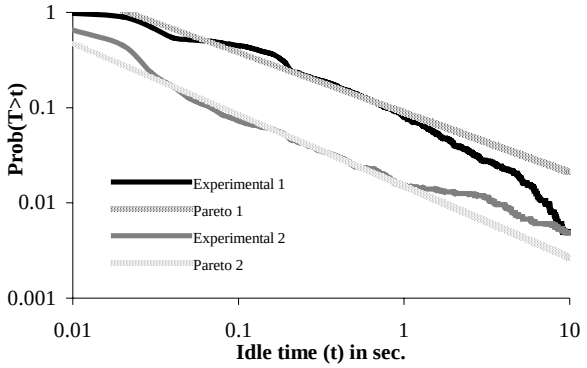

Figure 4. Experimental and Pareto Distributions

The tail of the Pareto distribution with characteristic index *a* and normalizing constant *b* is shown in Equation 3. The tail of a distribution gives the probability that the idle time will be as long or longer the a given time.

$$P_i(\Delta t_i > \Delta t) = b\Delta t_i^{-a} \tag{3}$$

The parameters of Pareto distribution can be estimated using least-squares method on N samples of idle times Δt as shown in Equation 4. Note that on the log plot (see Figure 4) Pareto distribution is a straight line with slope *a* and intercept *b*. On every new idle time sample, only the related probability value, *P*, needs to be updated before recalculating parameters *a* and *b*.

$$a = -\frac{\sum_{i=1}^{N} \ln \Delta t_i (N \ln P_i - \sum_{j=1}^{N} \ln P_j)}{\sum_{i=1}^{N} \ln \Delta t_i (N \ln \Delta t_i - \sum_{j=1}^{N} \ln \Delta t_j)} \tag{4}$$

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

The estimators presented track system changes. When a change is detected, the power management control has to be recalculated. The formulation of policy optimization for Renewal model is shown in Equation 5, 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 power constraint. An open-loop policy optimization problem similar to this one has already been solved for portable systems in [17] by using a linear program solver. The optimal policy is obtained in tens of seconds, which is much too long for implementation of the **closed-loop** power management control presented in this work.

$$\begin{aligned} min \quad & \frac{\sum_j p(j)d(j)}{\sum_j p(j)t(j)} \\ s.t. \quad & \sum_j p(j)[e(j) - t(j)P_{constr}] = 0 \\ & \sum_j p(j) = 1; \quad p(j) \geq 0 \quad \forall j \end{aligned} \tag{5}$$

$$\begin{aligned} \min \quad & v \\ s.t. \quad & d(j) + v(j)t(j) + u(j)[e(j) - t(j)P_{constr}] - \lambda(j) = 0 \quad \forall j \end{aligned} \tag{6}$$

A much faster way to solve the original optimization problem is to formulate the dual of its Lagrangian as shown in Equation 6 [27]. Variables *v*, *u* and λ are the Lagrangian multipliers. A minimum crossing point of a set of lines specified by Equation 7 is the solution to the dual. The indexes of Lagrangian multipliers which form the solution are used back in the original constraints shown in Equation 5 to obtain values of probabilities of transitioning into low-power state, *p(j)*. Using this method the optimal solution can be calculated in a matter of milliseconds, in contrast to linear program solvers that takes tens of seconds. In this way closed-loop control can be implemented in real-time.

$$v(j) \leq \frac{d(j)}{t(j)} + u(j)\frac{[e(j) - t(j)P_{constr}]}{t(j)} \tag{7}$$

The final output of optimization is a table that specifies probabilities of transitioning a core into low-power states, to be used in addition to a table that gives frequency and voltage settings as a function of arrival and service rates. Both tables can be accessed from either software or hardware, depending on how the local power management controller is realized. An example of optimal control for transition to low-power state is shown in Table 2 (lightly shaded region is for node-centric power management only). 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. The same functionality can also be implemented in hardware, as is presented in Section 4.2.

Table 2. Sample controller

Source	Idle Time (ms)	Transition Probability
Node	0	0.0
	70	0.3
	120	1.0
Network	Any time	1.0

3.2 Network-centric power management

The main advantage of implementing network-centric power management is the ability to make better predictions about the future workload. Node-centric power manager does not have any information about the future service requests from the other cores. In contrast, with network-centric approach each local estimator receives other core's service requests in advance and is given a signal when the service is no longer needed.

The network-enhanced estimator, shown in Figure 3, has two ports – the current core workload input (*Core Traffic*), and the network power management request port (*Network PM Request*) that is both an input and an output. The networking interface has to be defined in such a way that protocols support passing power

management messages between the cores. The design of networking protocol for NOCs is beyond the scope of this paper, but has been addressed in [2,3]. Network power management adds very few overhead packets to the overall communication stream between cores.

A network message in effect forces the local PM to make a deterministic decision instead of randomized decision made when only node-centric power management is used. Table 2 gives an example of controller implementation for decision on when to transition into low-power state when both node and network centric power management approaches are implemented (darkly shaded row). When local power manager receives a request for service by another core, it starts waking up the core if it was in a low-power state (with probability 1.0 as shown in Table 2). Otherwise the core verifies that it is ready to receive requests. The cost of waking up the core can be masked, as the other network elements may be able to notify the local core ahead of time when it's services are needed. Once the core's services are not required anymore, the local PM can be notified via a network message. At that point, if no other requests are pending, the core can enter a low-power state without any additional idle time. As a result, the amount of energy wasted while the core is idle is reduced, as the local PM knows ahead of time that no requests are arriving in near future.

Clearly, the best results are obtained when all cores can communicate ahead of time their needs to each other. Unfortunately, that is not always possible, since ultimately the final decisions on what the system will be doing next are made by the users of the system who are inherently non-deterministic. Thus the best approach for managing power consumption in NOCs is a combination of node and network centric techniques, such as the one presented in this work. The results discussed in the next section highlight this conclusion.

4. 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 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 policy recomputation. 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*).

Each core in NOC has a local power manager, that in turn consists of an estimator and a controller. The primary 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 first discuss the quality of the estimators, followed by the controller implementation. Finally, energy savings are contrasted when using only the node-centric approach with the combined node and network-centric power management.

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

4.1 Estimator

There are two core states in which power management decisions are made: appropriate service level is determined in the active state, while the decision on when to transition into the sleep state occurs in the idle state. The quality of both decisions depends on the estimation accuracy and speed. We first evaluate the estimation of the arrival rate, followed by the estimation of the parameters needed to model the distribution of idle times.

The estimator uses maximum likelihood ratio shown in Equation 1 to detect a change in incoming arrival frame rate of the MPEG video core from 12 frames/sec to 28 frames/sec. Window size (*w* in Equation 1) varies from 10 to 20 frame arrival times. When the ratio is evaluated on every frame arrival, the detection is delayed by only one frame, while when it is evaluated every five or more frames, it has maximum five frame delay. The rate detected is exact when the window size is between 10 and 20 frames. The computation overhead of detection is small as compared to the frame decoding time.

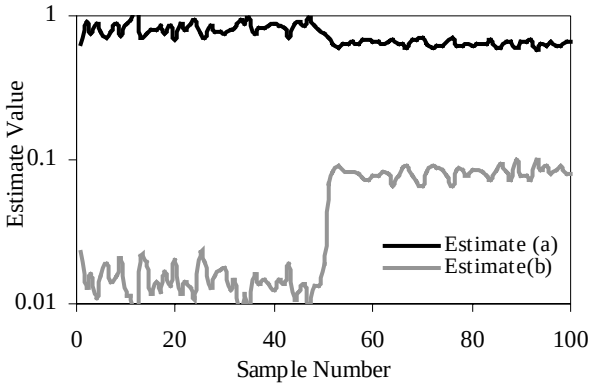

Figure 5. Dynamic Pareto Parameter Estimates

Distribution of length of time spent in the idle state is also estimated at run time. This distribution has heavy-tailed behavior, so Pareto distribution with its two parameters is used in estimation. Figure 5 shows how the estimator detects a change in the idle time distribution on communications core when the traffic pattern changes between two examples shown in Figure 4. The change is most clearly observed in the intercept value, as the characteristic slope stays nearly constant. Figure 6 shows the accuracy of Pareto parameter estimates over a large number of samples. The error is defined as the difference between a Pareto fit done using off-line versus run-time analysis. Average error is about 2%. Since characteristic slope varies less, the estimate of it is also more accurate than the estimate of intercept.

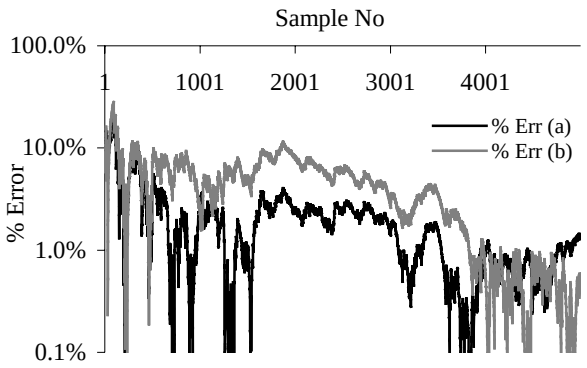

Figure 6. Errors in Pareto Parameter Estimates

4.2 Controller

The local power manager's controller can be realized in software, hardware or a combination of the two. For policies where critical parameters change very often, the control and the estimator 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 policy 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 in Section 3.1, we focus on the hardware implementation next.

There are three different components to the optimal controller: the random number generator, the policy and the timer. The timer is used to measure the length of idle period before the policy is evaluated. Typically core's processors already have programmable timers aboard that can be used by the hardware controller. Thus only the policy and the random number generator need to be implemented in hardware. The simplest hardware implementation for random number generator is to use Linear Feedback Shift Register (LFSR). Figure 7 shows the tradeoff between the number of bits in LFSR (and thus larger area) and cost in terms of power and performance penalty. The cost is calculated as the percent deviation from the optimal values. Even with as little as 8 bits, the hardware LFSR gives results within 5% of optimal.

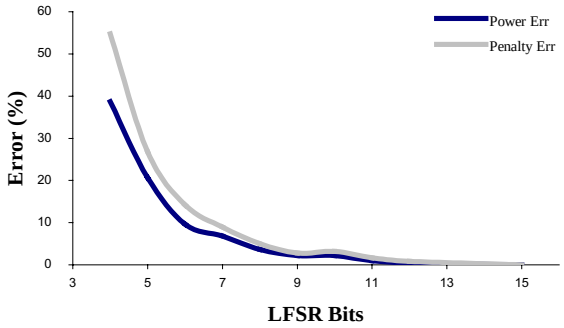

Figure 7. Probability Resolution Error

Table 4. Local PM Policy FPGA Synthesis Results

LFSR	LFSR Regs		Policy	
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. Policy and LFSRs take up only 3 Logic Array Blocks (LABs) for LFSR sizes ranging from 5 to 15 bits. In addition, the time it takes to arrive at the decision is in the range of nanoseconds, while the minimum idle times the power manager would respond to are on the order of milliseconds. The same policy evaluates even faster when synthesized into gates using Synopsis tools as shown in Table 5. In this case, the LFSR area is consistently about 12-14% of the total power management area. The policy includes the finite state machine needed to make the decisions and the comparator. Even the largest design takes only 15 registers and 855 gates.

Table 5. Local PM Policy Synopsis Synthesis Results

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

4.3 Network-centric Power Management

Table 6 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 input to the simulator. The results report a factor of savings in energy with reference to not using any power management. The lightly shaded portion of the table reports results when using only node-centric approach, while the darker shaded row reports results for a combined node and network-centric approaches.

Table 6. 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	3.9	1.2
	DPM only	2.0	1.5	3.0	2.0	2.4
	DVS&DPM	3.4	3.5	4.0	5.9	3.6
Network	DVS&DPM	3.7	3.6	4.2	6.4	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 4 in savings. 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 overall savings of a factor of 3.6.

When network power management is included with the node-centric approach (the last row in Table), the savings in energy grow to a factor of 4.1 with performance penalty reduced by 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.

5. 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 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. Optimal control is formulated using a Renewal model that can be recalculated in a matter of milliseconds when distribution parameters change. An efficient hardware implementation of the local PM controller is presented that has negligible delay in control implementation with a very small increase in hardware area. The final implementation of node and network centric power management approaches shows savings of a factor of four at system level while improving performance.

6. ACKNOWLEDGMENTS

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

7. 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] L. Benini, G. De Micheli, "Powering Networks on Chips," *International Symposium on System Synthesis*, To appear as an Invited talk, 2002.

[5] D. Wingrad, "MicroNetwork-Based Integration for SOCs," *Design Automation Conference*, pp. 673-678, 2001.

[6] "Virtual Component Interface Standard Version 2 On-Chip Bus DWG," VSI, April 2001.

[7] J-Y. Brunel, W. Kruijtzer, H. Kenter, F. Petrot, L. pasquier, E. de Kock, W. Smits, "COSY Communication IP's," *Design Automation Conference*, pp. 406-409, 2000.

[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] "AMBA Specification," ARM Inc, May 1999.

[11] "The CoreConnect Bus Architecture," IBM, 1999.

[12] M. Srivastava, A. Chandrakasan. R. Brodersen, ``Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Transactions on VLSI Systems*, vol.4, no.1, pp.42-55, March 1996.

[13] C.-H. Hwang and A. Wu, ``A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation", *International Conference on Computer Aided Design*, pp. 28-32, 1997.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

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

[24] 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.

[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] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, G. De Micheli: "Dynamic Voltage Scaling for Portable Systems" , *Design Automation Conference*, pp.524-529, 2001.

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