Operating System and Algorithmic Techniques for Energy Scalable Wireless Sensor Networks

Amit Sinha and Anantha P. Chandrakasan

Massachusetts Institute of Technology, Cambridge, MA 02139

{sinha,anantha}@mtl.mit.edu

Abstract. An system-level power management technique for massively distributed wireless microsensor networks is proposed. A power aware sensor node model is introduced which enables the embedded operating system to make transitions to different sleep states based on observed event statistics. The adaptive shutdown policy is based on a stochastic analysis and renders desired energy-quality scalability at the cost of latency and missed events. The notion of algorithmic transformations that improve the energy quality scalability of the data gathering network are also analyzed.

1 Introduction

Massively distributed, ad-hoc, wireless microsensor networks have gained importance in a wide spectrum of civil and military applications [1]. Advances in MEMS technology [2], combined with low power, low cost DSPs and RF circuits have resulted in cheap and wireless microsensor networks becoming feasible. A distributed, self-configuring network of adaptive sensors has significant benefits. They can be used to remotely monitor inhospitable and toxic environments. A large class of benign environments too require the deployment of a large number of sensors such as intelligent patient monitoring, object tracking, assembly line sensing, etc. Their massively distributed nature provides wider resolution as well as increased fault tolerance than a single sensor node. Several projects that demonstrate the feasibility of sensor networks are underway [3][4].

A wireless microsensor node is typically battery-operated and is thus energy constrained. To maximize the lifetime of the sensor node after its deployment all aspects including circuits, architecture, algorithms and protocols have to be made energy efficient. Once the system has been designed, additional energy savings can be obtained by using dynamic power management concepts [5] whereby the sensor node is shutdown if no interesting events occur. Such event driven power consumption is critical to obtaining maximum battery-life. In addition, it is highly desirable that the node has a graceful energy-quality (E-Q) scalability [6] such that if the application so demands, the user is able to extend the mission lifetime at the cost of sensing accuracy. Energy scalable algorithms and protocols are required for such energy constrained situations.

Sensing applications will present a wide range of requirements in terms of data rates, computation, average transmission distance, etc. As such protocols and algorithms will have to be tuned to each application. Therefore, embedded operating systems and software will be critical ingredients in such microsensor networks as programmability will be a necessary requirement. In this paper we

propose an Operating System directed Power Management technique to improve the energy efficiency of the sensor nodes. *Dynamic Power Management* (DPM) is an effective tool to reduce system power consumption without significantly degrading performance. The basic idea is to shut down devices when they are not needed and wake them up when necessary. DPM in general is a non-trivial problem. If the energy and performance overheads in transitioning to sleep states were negligible then a simple greedy algorithm which makes the system go into the deepest sleep state as soon as it is idle would be perfect. However, in reality, transitioning to a sleep state has the overhead of storing the processor state and shutting off the power supply. Waking up too takes a finite amount of time. Therefore, implementing the right policy for transitioning to the sleep state is critical for the success of DPM. A power-aware model for sensor nodes is also introduced. We also demonstrate how the algorithm can be used to provide desirable E-Q characteristics in sensing applications. Finally, we introduce the concept of energy scalable algorithms. The principal notion being that computation be done is such a fashion that a drop in energy availability should result in minimum possible quality degradation.

2 System Models

2.1 Sensor Network and Node Model

The fundamental idea in distributed sensor applications is to incorporate sufficient processing power in each node such that they are self-configuring and adaptive. Fig. 1 illustrates the basic sensor node architecture. Each node consists of the embedded sensor, A/D converter, a processor with memory (which in our case will be the StrongARM SA-1100 processor [7]) and the RF circuits. Each of these components are controlled by the micro Operating System (μ-OS) through micro device drivers. An important function of the μ-OS is to enable Power Management (PM). Based on event statistics, the μ-OS decides which devices to turn off/on.

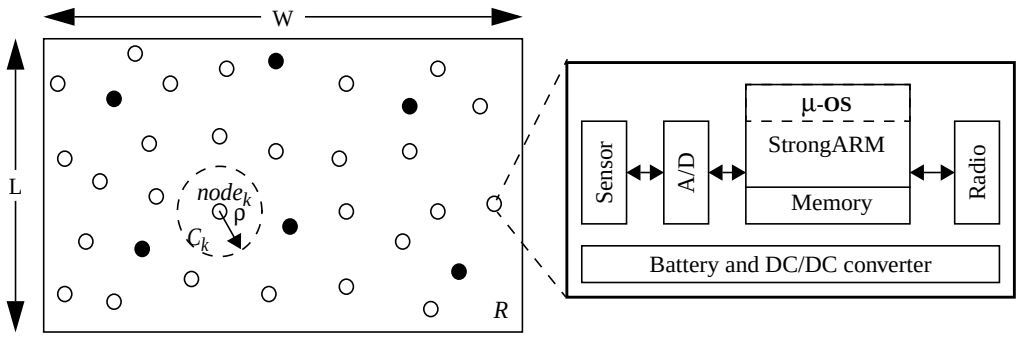


Fig. 1. Sensor network and node architecture

Our network essentially consists of η homogeneous sensor nodes distributed over a rectangular region *R* with dimensions *W*x*L* with each node having a visibility radius of ρ (shown by the region C_k). Three different communication models can be used for such a network. (i) Direct transmission

(every node directly transmits to the basestation), (ii) Multi-hop (data is routed through the individual nodes towards the basestation) and (iii) Clustering. If the distance between the neighboring sensors is less than the average distance between the sensors and the user or the basestation, transmission power can be saved if the sensors collaborate locally. Further its likely that sensors in local clusters share highly correlated data. Some of the nodes elect themselves as 'cluster heads' (as depicted by nodes in black) and the remaining nodes join one of the clusters based on a minimum transmit power criteria. The cluster head then aggregates and transmits the data from the other cluster nodes. Such application specific network protocols for wireless microsensor networks have been developed [8]. It has been demonstrated that a clustering scheme is an order of magnitude more energy efficient than a simple direct transmission scheme.

2.2 Power Aware Sensor Node Model

Table 1. Useful sleep states for the sensor node

	StrongARM	**Memory**	**Sensor, A/D**	**Radio**
s_0	active	active	on	tx, rx
s_1	idle	sleep	on	rx
s_2	sleep	sleep	on	rx
s_3	sleep	sleep	on	off
s_4	sleep	sleep	off	off

A power aware sensor node model essentially describes the power consumption in different levels of node-sleep state. Every component in the node can have different power modes, e.g. the StrongARM can be in active, idle or sleep mode; the radio can be in transmit, receive, standby or off mode. Each node-sleep state corresponds to a particular combination of component power modes. In general, if there are N components labelled (1, 2, ..., N) each with k_i number of sleep states, the total number of node-sleep states are $\prod k_i$. Every component power mode is associated with a latency overhead for transitioning to that mode. Therefore each node sleep mode is characterized by a power consumption and a latency overhead. However, from a practical point of view not all the sleep states are useful. Table 1 enumerates the component power modes corresponding to 5 different useful sleep states for the sensor node. Each of these node-sleep modes correspond to an increasingly deeper sleep state and is therefore characterized by an increasing latency and decreasing power consumption.These sleep states are chosen based on actual working conditions of the sensor node e.g. it does not make sense to have the A/D in the active state and everything else completely off. The design problem is to formulate a policy of transitioning between states based on observed events so as to maximize energy efficiency. The power aware sensor model is similar to the system power model in the Advanced Configuration and Power Interface (ACPI) standard [9]. An ACPI compliant system has five global states. `SystemStateS0` (working state), and `SystemStateS1` to `SystemStateS4` corresponding to four different levels of sleep states. The sleep states are differentiated by the power consumed, the overhead required in going to sleep and the wakeup time. In general, the deeper the sleep state, the lesser the power consumption, and the longer the wakeup time. Another aspect of similarity is that in ACPI the *Power Manager* (PM) is a module of the OS.

2.3 Event Generation Model

An event is said to occur when the a sensor node picks up a signal with power above a pre-defined threshold. For analytical tractability we assume that every node has a uniform radius of visibility ρ. In real applications the terrain might influence the visible radius. An event can be static (e.g. a localized change in temperature/pressure in an environment monitoring application) or can propagate (e.g. signals generated by a moving object in a tracking application). In general, events have a characterizable (possibly non-stationary) distribution in space and time. We will assume that the temporal behavior of events over the entire sensing region, *R*, is a Poisson process with an average rate of events given by λ_{tot} [10]. In addition we assume that the spatial distribution of events is characterized by an independent probability distribution given by $p_{XY}(x,y)$. Let p_{ek} denote the probability that an event is detected by $node_k$, given the fact that it occurred in *R*.

$$p_{ek} = \frac{\int_{C_k} p_{XY}(x, y)dxdy}{\int_{R} p_{XY}(x, y)dxdy} \tag{1}$$

Let $p_k(t, n)$ denote the probability that *n* events occur in time *t* at $node_k$. Therefore, the probability of no events occurring in C_k over a threshold interval T_{th} is given by

$$p_k(T_{th}, 0) = \sum_{i=0}^{\infty} \frac{e^{-\lambda_{tot}T_{th}}(\lambda_{tot}T_{th})^i}{i!}(1 - p_{ek})^i = e^{-p_{ek}\lambda_{tot}T_{th}} \tag{2}$$

Let $p_{th,k}(t)$ be the probability that at least one event occurs in time *t* at $node_k$.

$$p_{th,k}(T_{th}) = 1 - p_k(T_{th}, 0) = 1 - e^{-p_{ek}\lambda_{tot}T_{th}} \tag{3}$$

i.e. the probability of at least one event occurring is an exponential distribution characterized by a spatially weighted event arrival rate $\lambda_k = \lambda_{tot}\, p_{ek}$.

In addition, to capture the possibility that an event might propagate in space we describe each event by a position vector, $\bar{\mathbf{p}} = \bar{\mathbf{p}}_0 + \int \bar{\mathbf{v}}(t)dt$. Where $\bar{\mathbf{p}}_0$ is the coordinates of the point of origin of the event and $\bar{\mathbf{v}}(t)$ characterizes the propagation velocity of the event. The point of origin has a spatial and temporal distribution described by Equation 1 to Equation 3. We have analyzed three distinct classes of events: (i) $\bar{\mathbf{v}}(t) = 0$, the events occur as stationary points, (ii) $\bar{\mathbf{v}}(t) = const$, the event propagates with fixed velocity (e.g. a moving vehicle), and, (iii) $|\bar{\mathbf{v}}(t)| = const$, the event propagates with fixed speed but random direction (i.e. a random walk).

3 Shutdown Policy

3.1 Sleep State Transition Policy

Assume an event is detected by $node_k$ at some time and it finishes processing it at t_1 and the next event occurs at time $t_2 = t_1 + t_i$. At time t_1, $node_k$ decides to transition to a sleep state s_k from the active state s_0 as shown in Fig. 2. Each state s_k has a power consumption P_k, and the transition time to it from the active state and back is given by $\tau_{d,k}$ and $\tau_{u,k}$ respectively. By our definition of node-

sleep states, $P_j > P_i$, $\tau_{d,i} > \tau_{d,j}$ and $\tau_{u,i} > \tau_{u,j}$ for any $i > j$.

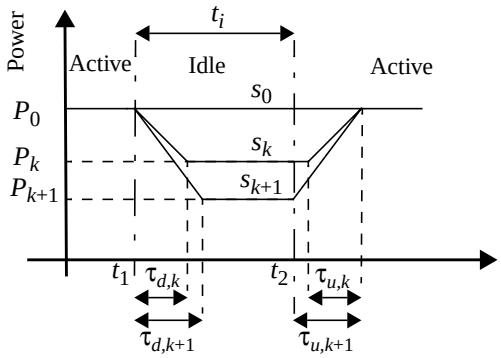


Fig. 2. State transition latency and power

We will now derive a set of sleep time thresholds { $T_{th,k}$ } corresponding to the states { s_k }, $0 \le k \le N$ (for N sleep states) such that transitioning to a sleep state s_k from state s_0 will result in a net energy loss if the idle time $t_i < T_{th,k}$ because of the transition energy overhead [12]. This assumes that no productive work can be done in the transition period, which is invariably true, e.g. when a processor wakes up the transition time is the time required for the PLLs to lock, the clock to stabilize and the processor context to be restored. The energy saving because of state transition is given by

$$\begin{aligned} E_{save,\,k} &= P_0 t_i - \left(\frac{P_0 + P_k}{2}\right)(\tau_{d,\,k} + \tau_{u,\,k}) - P_k(t_i - \tau_{d,\,k}) \\ &= (P_0 - P_k)t_i - \left(\frac{P_0 - P_k}{2}\right)\tau_{d,\,k} - \left(\frac{P_0 + P_k}{2}\right)\tau_{u,\,k} \end{aligned} \tag{4}$$

and such a transition is only justified when $E_{save,k} > 0$. This leads us to the threshold

$$T_{th,\,k} = \frac{1}{2}\left[\tau_{d,\,k} + \left(\frac{P_0 + P_k}{P_0 - P_k}\right)\tau_{u,\,k}\right] \tag{5}$$

which implies that the longer the delay overhead of the transition s_0 -> s_k, the higher the energy gain threshold, and the more the difference between P_0 and P_k, the smaller the threshold. These observations are intuitively appealing too.Table 2 lists out the power consumption of a sensor-node described in Fig. 1 in the different power modes. Since the node consists of off the shelf components, its not optimized for power consumption. However, we will use the threshold and power consumption numbers detailed in Table 2 to illustrate our basic idea. The steady state shutdown algorithm is as follows

```
if( eventOccurred() == true ) {
   processEvent();
   ++eventCount;
   lambda_k = eventCount/getTimeElapsed();
```

```
for( k=4; k>0; k-- )
   if( computePth( Tth(k) ) < pth0 )
      sleepState(k);
}
```

When an event is detected at $node_k$, it wakes up and processes the event (this might involve classification, beamforming, transmission etc.). It then updates a global `eventCount` counter which stores the total number of events registered by $node_k$. The average arrival rate, λ_k, for $node_k$ is then updated. This requires use of an µ-OS timer based system function call `getTimeElapsed()` which returns the time elapsed since the node was turned on. The µ-OS then tries to put the node into sleep state s_k (starting from the deepest state s_4 through s_1) by testing the probability of an event occurring in the corresponding sleep time threshold $T_{th,k}$ against a system defined constant p_{th0}.

Table 2. Sleep state power, latency and threshold

State	P_k (mW)	τ_k (ms)	$T_{th,k}$
s_0	1040	-	-
s_1	400	5	8
s_2	270	15	20
s_3	200	20	25
s_4	10	50	50

3.2 Missed events

All the sleep states, except state s_4 have the actual sensor and A/D circuit on. Therefore if an event is detected (i.e. the signal power is above a threshold level) the node transitions to state s_0 and processes the event. The only overhead involved is latency (worst case being about 25ms). However, in state s_4, the node is almost completely off and it must decide on its own when to wake up. In sparse event sensing systems (for example vehicle tracking, seismic detection etc.) the inter-arrival time for events is much greater than the sleep time thresholds $T_{th,k}$. Therefore, the sensor node will invariably go into the deepest sleep state s_4. The processor must watch for pre-programed wake-up signals. These signal conditions are programmed by the CPU prior to entering the sleep state. To be able to wake up on its own the node must be able to predict the arrival of the next event. An optimistic prediction might result in the node waking up unnecessarily while a pessimistic strategy will result in some events being missed.

Researchers have tried to model the interarrival process of events in reactive systems. In [11] the distribution of idle and busy periods is represented by a time series and approximated by a least square regression model. In [12] the idleness prediction is based on a weighted sum of past periods where the weights decay geometrically. The authors of [13] use a stochastic optimization technique based on the theory of Markov processes. All the above techniques result in a performance penalty. However, in our context, being in state s_4 results in missed events as the node has no way of knowing if anything significant occurred. What strategy gets used is a pure design concern based on how critical the sensing task is. We discuss two possible approaches.

- Completely disallow s_4 - If the sensing task is critical and any event cannot be missed this state must be disabled.
- Selectively disallow s_4 - This technique can be used if events are spatially distributed and not totally critical. Both random and deterministic approaches can be used. In the protocol described in [8] the 'cluster heads' can have a disallowed s_4 state while the normal nodes can transition to s_4. Alternatively, the scheme that we propose is more homogeneous. Every $node_k$ that satisfies the sleep threshold condition for s_4 goes to sleep with a system defined probability p_{s4} for a time duration given by

$$t_{s4,\,k} = -\frac{1}{\lambda_k}\ln(p_{s4}) \tag{6}$$

Equation 6 describes the steady state behavior of the node and the sleep time is computed such that the probability that no events occur in $t_{s4,\,k}$ i.e. $p_k(t_{s4,\,k}, 0) = p_{s4}$. However, when the sensor network is switched on and no events have occurred for a while, λ_k is zero. To account for this we disallow transition to state s_4 until at least one event is detected. We can also have an adaptive transition probability p_{s4}, which is zero initially and increases as events are detected later on. The probabilistic state transition is described in Fig. 3.

The advantage of the algorithm is that efficient energy tradeoffs can be made with event detection probability. By increasing p_{s4}, the system energy consumption can be reduced while the probability of missed events will increase and vice versa. Therefore, our overall shutdown policy is governed by two implementation specific probability parameters, (i) p_{th0} and (ii) p_{s4}.

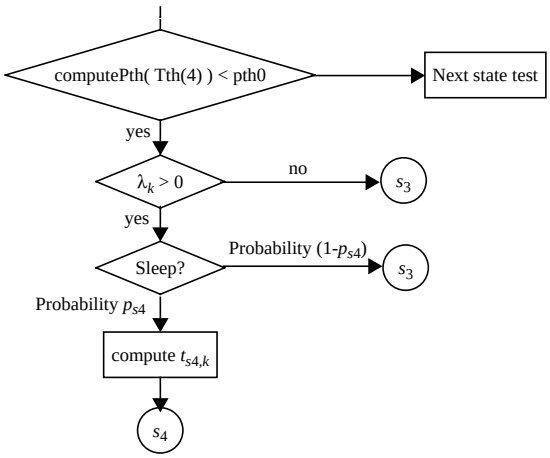


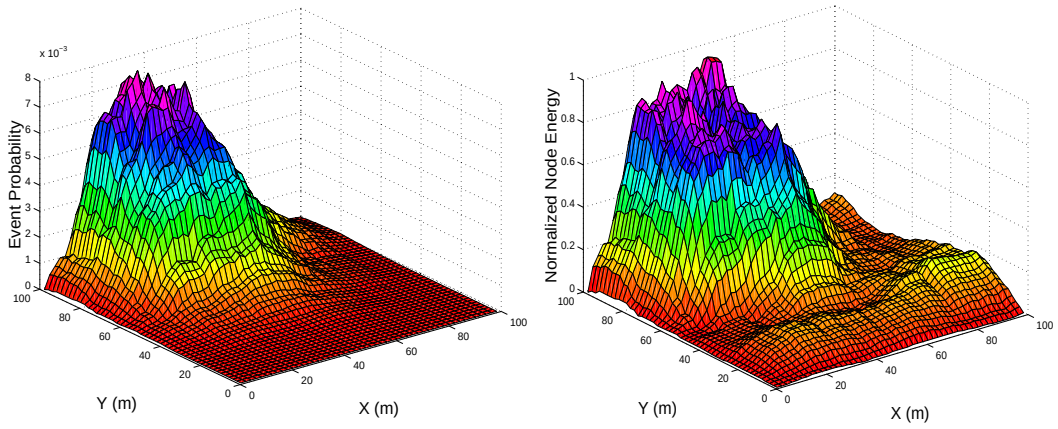
Fig. 3. Transition algorithm to 'almost off' s_4 state

3.3 Results

We have simulated a η=1000 node system distributed uniformly and randomly over a 100m x 100m

area. The visibility radius of each sensor was assumed to be ρ=10m. The sleep state thresholds and power consumption are shown in Table 2. Fig. 4 shows the overall spatial node energy consumption over for an event with a gaussian spatial distribution centered around (25, 75). The interarrival process is Poisson with λ_{tot} = 500 s^{-1}. It can be seen that the node energy consumption tracks the event probability. In the non-power managed scenario we would have a uniform energy consumption in all the nodes.

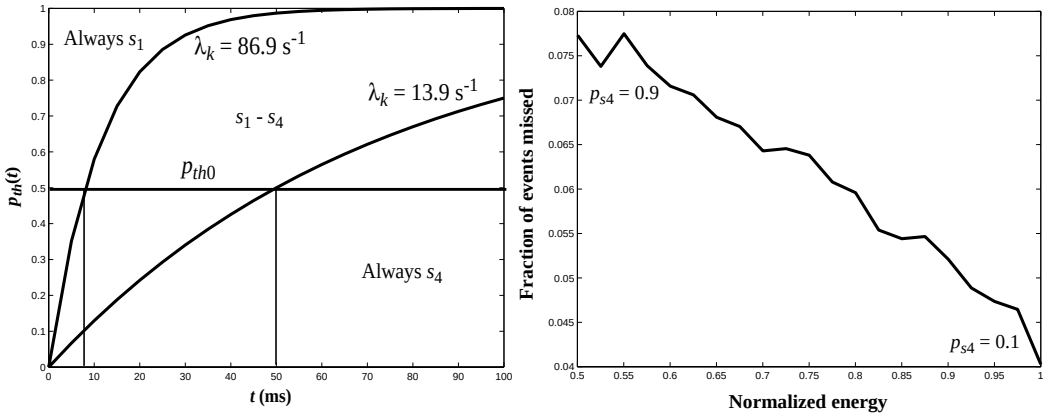
One drawback of the whole scheme is that there is a finite and small window of interarrival rates λ_{tot} over which the fine- grained sleep states can be utilized. In general, the more the differentiated the power states (i.e. the greater the difference in their energy and latency overheads) the wider the range of interarrival times over which all sleep states can be utilized. Fig. 5(a) shows the range of event arrival rates at a node (λ_k) over which the states s_1 - s_3 are used significantly. If λ_k < 13.9 s^{-1}, transition to state s_4 is always possible (i.e. at least the threshold condition is met, actual transition of course occurs with probability p_{s4}). Similarly, if λ_k > 86.9 s^{-1}, the node must always be in the most active state. These limits have been computed using the nominal p_{th0} = 0.5. Using a higher value of p_{th0} would result in frequent transitions to the sleep states and if events occur fast enough this would result in increased energy dissipation associated with the wake-up energy cost. A smaller value of p_{th0} would result in a pessimistic scheme for sleep state transition and therefore lesser energy savings.



(a) (b)

Fig. 4. (a) Spatial distribution of events (gaussian) and (b) Spatial energy consumption in the sensor nodes

Fig. 5(b) illustrates the Energy-Quality trade-off of our shutdown algorithm. By increasing the probability of transition to state s_4 (i.e. increasing p_{s4}) energy can be saved at the cost of increased possibility of missing an event. Such a graceful degradation of quality with energy is highly desirable in energy constrained systems.



(a) (b)

Fig. 5. (a) Event arrival rates at a node (b) Fraction of events missed versus energy consumption

4 Energy Scalable Algorithms

It is highly desirable that we structure our algorithms and systems in such a fashion that computational accuracy can be traded off with energy requirement. At the heart of such transformations lies the concept of incremental refinement. Consider a scenario where the distributed sensor network is being used to monitor seismic activity from a remote basestation. Sensor nodes are energy constrained and have a finite lifetime. It would be highly desirable to have energy scalable algorithms and protocols running on the sensor network such that the remote basestation could dynamically reduce energy consumption (to prolong mission lifetime if uninteresting events have occurred) by altering the throughput and computation accuracy. This type of behavior necessitates algorithmic restructuring so that every computational step leads us incrementally closer to the output. We now illustrate the algorithmic restructuring notion using two examples of popular signal processing algorithms in the context of sensor based computation.

4.1 Filtering Application

Finite Impulse Response (FIR) filtering is one of the most commonly used Digital Signal Processing (DSP) operations. FIR filtering involves the inner product of two vectors one of which is fixed and known as the impulse response, $h[n]$, of the filter [14]. An N-tap FIR filter is defined by Equation 7.

$$y[n] = \sum_{k=0}^{N-1} x[n-k]h[k] \tag{7}$$

When we analyze the FIR filtering operation from a pure inner product perspective, it simply involves N multiply and accumulate (MAC) cycles. For desired Energy-Quality (E-Q) behavior, the

MAC cycles that contribute most significantly to the output $y[n]$ should be done first. Each of the partial sums, $x[k]h[n-k]$, depends on the data sample and therefore its not apparent which ones should be accumulated first. Intuitively, the partial sums that are maximum in magnitude (and can therefore affect the final result significantly) should be accumulated first. Most FIR filter coefficients have a few coefficients that are large in magnitude and progressively reduce in amplitude. Therefore, a simple but effective *most-significant-first transform* involves sorting the impulse response in decreasing order of magnitude and reordering the MACs such that the partial sum corresponding to the largest coefficient is accumulated first as shown in Fig. 6(a). Undoubtedly, the data sample multiplied to the coefficient might be so small as to mitigate the effect of the partial sum. Nevertheless, on an average case, the coefficient reordering by magnitude yields a better *E-Q* performance than the original scheme.

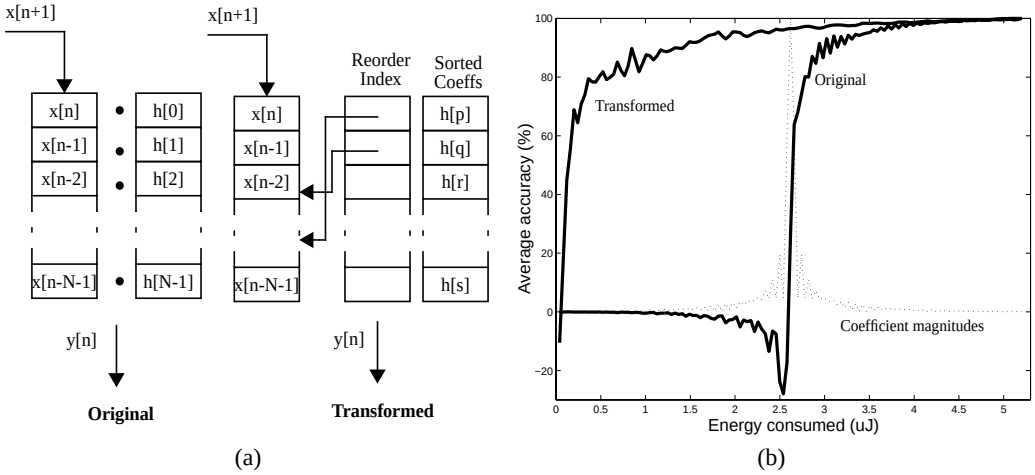


Fig. 6. (a) FIR filtering with coefficient reordering (b) *E-Q* graph for original and transformed FIR filtering

Fig. 6(b) illustrates the scalability results for a low pass filtering of speech data sampled at 10kHz using a 128-tap FIR filter whose impulse response (magnitude) is also outlined. The average energy consumption per output sample (measured on the StrongARM SA-1100 operating at 1.5V power supply and 206MHz frequency) in the original scheme is 5.12μJ. Since the initial coefficients are not the ones with most significant magnitudes the *E-Q* behavior is poor. Sorting the coefficients and using a level of indirection (in software that amounts to having an index array of the same size as the coefficient array), the *E-Q* behavior can be substantially improved. The energy overhead associated with using a level of indirection on the SA-1100 was only 0.21μJ which is about 4% of the total energy consumption. The basic characteristic of an energy scalable algorithm is illustrated in Fig. 6(b). It is obvious that if the available energy per sample/task is reduced by 50% the quality degradation in the transformed system is only 10% as opposed to 100% degradation in the original filtering algorithm.

4.2 Image Decoding Application

The Discrete Cosine Transform (DCT), which involves decomposing a set of image samples into a scaled set of discrete cosine basis functions, and the Inverse Discrete Cosine Transform (IDCT), which involves reconstructing the samples from the basis functions, are crucial steps in digital video [15]. The 64-point, 2-D DCT and IDCT (used on 8x8 pixel blocks in of an image) are defined respectively as

$$X[u, v] = \frac{c[u]c[v]}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} x[i, j] \cos\left(\frac{(2i+1)u\pi}{16}\right) \cos\left(\frac{(2j+1)v\pi}{16}\right) \tag{8}$$

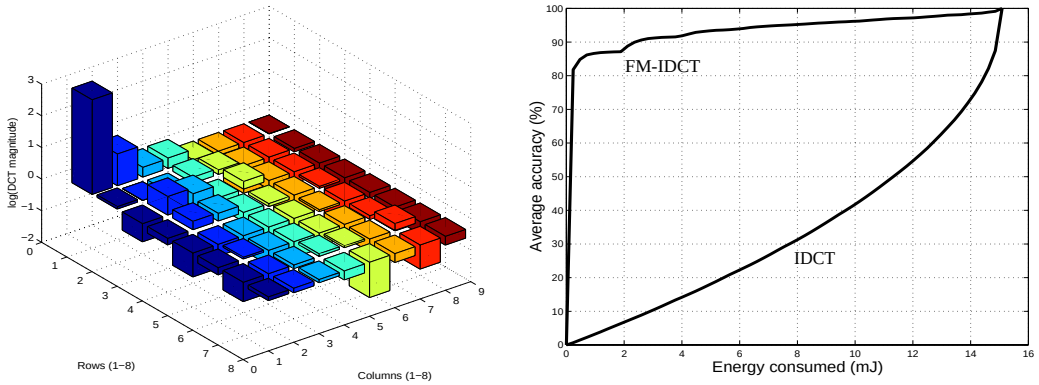
$$x[i, j] = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} c[u]c[v]X[u, v] \cos\left(\frac{(2i+1)u\pi}{16}\right) \cos\left(\frac{(2j+1)v\pi}{16}\right) \tag{9}$$

DCT is able to capture the spatial redundancy present in an image and the coefficients obtained are quantized and compressed. Most existing algorithms attempt to minimize the number of arithmetic operations (multiplications and additions) usually relying on the symmetry properties of the cosine basis functions (similar to the FFT algorithm) and on matrix factorizations [16]. The *E-Q* behavior of these algorithms are not good as they have been designed such that computation takes a minimal yet constant number of operations. The Forward Mapping-IDCT (FM-IDCT) algorithm, proposed in [17] can be shown to have an *E-Q* performance with is much better than other algorithms. The algorithm is formulated as follows

$$\begin{bmatrix} x_{0,0} \\ x_{0,1} \\ \vdots \\ x_{8,8} \end{bmatrix} = X_{0,0} \begin{bmatrix} c_0^{0,0} \\ c_1^{0,0} \\ \vdots \\ c_{64}^{0,0} \end{bmatrix} + X_{0,1} \begin{bmatrix} c_0^{0,1} \\ c_1^{0,1} \\ \vdots \\ c_{64}^{0,1} \end{bmatrix} + \ldots + X_{8,8} \begin{bmatrix} c_0^{8,8} \\ c_0^{8,8} \\ \vdots \\ c_{64}^{8,8} \end{bmatrix} \tag{10}$$

where $x_{i,j}$ are the reconstructed pels, $X_{i,j}$ are the input DCT coefficients, and $[c_k^{i,j}]$ is the 64x64 constant reconstruction kernel. The improved *E-Q* behavior of the FM-IDCT algorithm can be attributed to the fact that most of the signal energy is concentrated in the DC coefficient ($X_{0,0}$) and in general in the low-frequency coefficients as shown in Fig. 7(a). Instead of reconstructing each pixel by summing up all its frequency contributions, the algorithm incrementally accumulates the entire image based on spectral contributions from the low to high frequencies.

Fig. 7(b) and Fig. 8 illustrate the *E-Q* behavior of the FM-IDCT algorithm. It is obvious from Fig. 7(b) that almost 90% image quality can be obtained from as little as 25% of the total energy consumption. In terms of the overhead requirement, the only change that is required is that we now need to store the IDCT coefficients in a transposed fashion (i.e. all the low frequency components first and so on).



(a) (b)

Fig. 7. (a) 8x8 DCT coefficients averaged over a sample image (b) *E-Q* graph for FM-IDCT vs normal IDCT

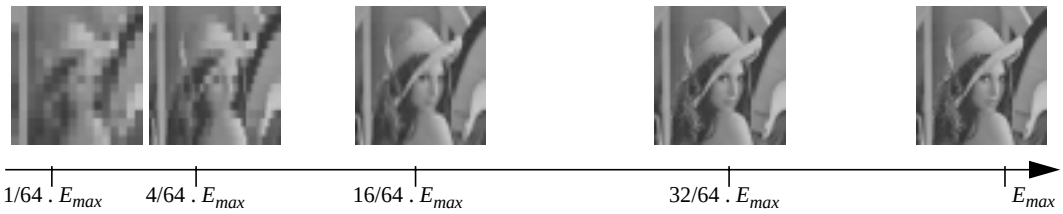


Fig. 8. llustrating the incremental refinement property with respect to computational energy of the FM-IDCT algorithm

5 Conclusions

We have proposed a system level power management scheme for distributed wireless data acquisition sensor networks using a two fold approach for energy scalability viz. event driven shutdown and algorithmic restructuring. For event driven shutdown, we explicitly characterize the meaningful power states of a node and uses a probabilistic technique to make predictive transitions to the low power modes based on observed event statistics. The scheme is simple to implement and has negligible memory overhead. The technique we have proposed is fairly general and can be used for power management in any system characterized by different levels of power consumption in various stages of shutdown. We have also demonstrated the feasibility of a graceful energy-quality tradeoff using our shutdown strategy. In energy constrained sensor nodes, it is desirable to have energy scalable algorithms. We have demonstrated the basic idea of algorithmic restructuring for improved energy-quality behavior using two popular signal processing examples.

ACKNOWLEDGEMENTS

This research is sponsored by the Defense Advanced Research Project Agency (DARPA) Power Aware Computing/Communication Program and the Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0551. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon[1].

REFERENCES

[1] A. P. Chandrakasan, et. al., "Design Considerations for Distributed Microsensor Systems", Proceedings of Custom Integrated Circuits Conference, San Deigo, CA, May 1999, pp. 279-286

[2] MEMS Technology Applications Center, `http://mems.mcnc.org`

[3] The MIT μAMPS Project, `http://www-mtl.mit.edu/research/icsystems/uamps/`

[4] The WINS Project, `http://www.janet.ucla.edu/WINS`

[5] L. Benini and G. D. Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*, Norwell, MA, Kluwer 1997

[6] A. Sinha, A. Wang and A. P. Chandrakasan, "Algorithmic Transforms for Efficient Energy Scalable Computation", International Symposium on Low Power Electronics and Design, Italy, July 2000

[7] `http://developer.intel.com/design/strong/sa1100.htm`

[8] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy Efficient Routing Protocols for Wireless Microsensor Networks", Proceedings of 33rd Hawaii International Conference on System Sciences (HICSS '00), January 2000

[9] Advanced Configuration and Power Interface, `http://www.teleport.com/~acpi/`

[10] E. R. Dougherty, *Probability and Statistics for Engineering, Computing and Physical Sciences*, Prentice Hall 1990

[11] M. B. Srivastava, A. P. Chandrakasan and R. W. Broderson, "Predictive System Shutdown and Other Architectural Techniques for Energy Efficient Programmable Computation", IEEE Transactions on VLSI Systems, vol. 4, no. 1, March 1996, pp. 42-54

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

[13] L. Benini, et. al, "Policy Optimization for Dynamic Power Management", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 18, no. 6, June 1999, pp. 813-833

[14] A. V. Oppenheim and R. W. Schafer, *Discrete Time Signal Processing*, Prentice Hall, New Jersey, 1989

[15] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete Cosine Transform", IEEE Trans. on Computers, vol. 23, Jan 1974, pp. 90-93

[16] W. H. Chen, C. H. Smith and S. C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform", IEEE Trans. on Communication, vol. 25, Sept 1977, pp. 1004-1009

[17] L. McMillan and L. A. Westover, "A Forward-Mapping Realization of the Inverse Discrete Cosine Transform", Proceedings of the Data Compression Conference (DCC '92), March 1992, pp. 219-228

---

1. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Project Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.