Power-Aware Systems

Manish Bhardwaj, Rex Min, Anantha Chandrakasan
Massachusetts Institute of Technology
Microsystems Technology Laboratory
Cambridge, MA 02139.
{manishb,rmin,anantha}@mit.edu

Abstract

The key to maximizing energy efficiency of systems is understanding and systematically harnessing the tremendous operational diversity they exhibit. We define the power-awareness of a system as its ability to minimize energy consumption by adapting to changes in its operating point. These changes occur as a result of variations in input statistics, desired output quality, tolerable latency and throughput. The key objective of this paper is to unambiguously define the notion of power-awareness, distinguish it from the better understood concept of low-power, to propose a systematic methodology that enhances power-awareness and finally to illustrate the impact of such re-engineering. By applying power-awareness formalisms to systems ranging from multipliers to variable voltage processors, we demonstrate increases in energy efficiency of 60%-200%.

1. Introduction

A system paradigm that has been gathering momentum recently is that of energy-scalable or power-aware design [1]. While the term "power-aware" is often ascribed to any system whose design has been sensitive to energy considerations, its connotation in recent work has been:

- The system allows its clients to adjust the expected quality and also the tolerable latency/throughput constraints.
- When such adjustments are made, the energy consumption is expected to vary accordingly i.e. higher energy dissipation is tolerated by clients for higher quality (or lower latency) and *vice-versa*. This property is often called energy-quality scalability, where quality denotes the overall quality-of-service, not just the output quality [1].
- The system is sensitive to input statistics and their variation. Instead of blindly processing the input, the system adapts itself to minimize its processing energy.

It is easy to see that an energy-constrained system exhibiting this behavior has the potential of achieving a longer lifetime than a power-unaware system - an invaluable characterisitic at a time when battery-operated, portable electronic devices have become ubiquitous. Several other factors are contributing to the emergence of power-awareness as a system design paradigm. Perhaps the most significant of these is the widespread realization that systems are designed with the worst case operating point in mind while typical operational profiles show a preponderance of non-worst case operation. As we shall see in this paper, resolving this mismatch leads to significant energy savings. Next, Quality-of-Service (QoS) scalability has become increasingly important in the context of system design and similar concepts are being applied to energy-quality tradeoffs. Thus, more and more users will expect to be able to trade some performance metric (output quality, tolerable latency, desired throughput etc.) for energy [2]. A power-aware multimedia terminal, for example, would allow the user to reduce the video fidelity in exchange for longer battery lifetime [3]. The third factor which has led to interest in power-aware systems is the focus on the huge operational diversity or "operational scenarios" that even the simplest systems exhibit. If a system is to be energy efficient, it follows that it must consume only as much energy as needed while in any particular scenario i.e. it must be power-aware in the sense of the third heuristic above. Hence, even in the absence of user intervention, the power-awareness paradigm offers non-trivial opportunities for energy reduction.

In this paper, our objective is to develop formalisms that allow one to systematically quantify and enhance power-awareness. In the latter half of the paper. we illustrate these formalisms by applying them to real-world circuits and systems.

2. Power-Awareness Formalisms

Formalizing the notion of power-awareness entails formalizing notions of "operating scenarios" and "perfect energy adaptation" to these scenarios. In the context of power-awareness, any system attribute which affects energy dissipation is a dimension in the space of scenarios. The five key dimensions are inputs, desired output quality, latency and throughput constraints, ambient environment and the internal state of the system. Given an application, the two key choices in characterizing operating scenarios are the dimensions that are included in the characterization and the detail in which they are captured. Consider, as an example, a 16x16-bit multiplier (say H). The simplest scenario characterization would be a scalar k defined thus:

$k = max(precision(X[n]), precision(Y[n]))$

where n is simply the discrete-time index and X and Y are the multiplier inputs. At any specified time, the multiplier would be in one of 16 scenarios depending on whether it was carrying out 1x1, 2x2, ... 16x16 bit multiplications. A natural refinement would be to include precision of both operands, rather than taking the greater of the two. Hence, the scenario would be the unordered tuple, $(m, n) = (precision(X[n]), precision(Y[n]))$ leading to 120 scenarios. By distinguishing between mxn-bit and nxm-bit multiplications, we would have 256 scenarios. It is clear that we can characterize the input dimension with increasing detail. Another example of a scenario characterization that includes input and state information is,

$< k, s >=< max(precision(X[n]), precision(Y[n])), max(precision(X[n-1]), precision(Y[n-1])) >$

Thus, for example, a multiplier resides in a <16,2> scenario when it executes a 16x16 bit multiplication preceded by a 2x2 bit multiplication. It is not difficult to extend such characterization to include desired output quality and latency constraints. The next abstraction that follows naturally from scenarios is that of an *energy curve* which is simply the plot of the energy consumed by a system against the scenarios in which a system resides. The energy curve for a 16-bit multiplier is shown in figure 1. To obtain the curve, input vectors were generated using a uniform distribution and the resultant energy dissipation for each scenario was measured[1]. Note that we have used the simplest scenario characterization here i.e. there are 16 scenarios corresponding to the maximum precision of the input operands. It is interesting to observe that a normal 16x16 bit multiplier is remarkably aware of its input precision. This is expected since, to a first order, smaller precision operands lead to lesser switched capacitance.

Examination of the energy curve naturally leads to the central question:

[1]All multiplier energy curves were obtained by $PowerMill^{TM}$ simulations of actual SPICE netlists in a 0.35μ process.

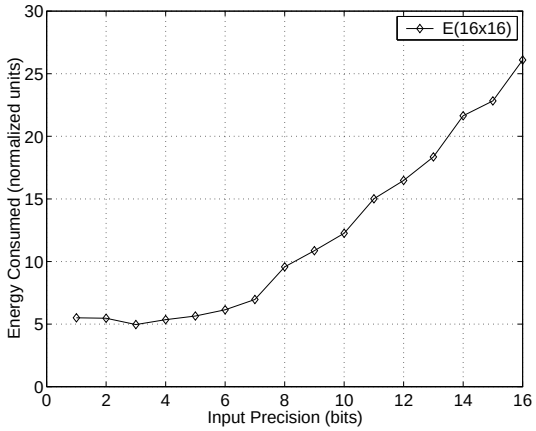

Figure 1. Multiplier energy as a function of input precision

How power-aware is the multiplier? Equivalently, how well is the multiplier's energy dissipation tracking scenarios?

We answer this question by first proposing the definition of a *perfectly* power-aware system:

A system H is perfectly power-aware if and only if its energy dissipation in any scenario is no greater than the minimum possible for that scenario.

To construct a realistic bound on the efficiency with which a system can track scenario changes, we take the following approach. Consider the multiplier's energy dissipation in a particular scenario, say s_i. We now construct the most energy efficient system that we can, say H_{s_i}, dedicated to executing this and only this scenario. This exercise is repeated for all scenarios. We call these H_{s_i}s *point systems* and denote the set of point systems by P. We now propose a revised definition of the perfectly power-aware system:

A system H is perfectly power-aware if and only if its energy dissipation in scenario s_i is no greater than that of the dedicated system H_{s_i}, constructed to execute scenario s_i as efficiently as possible.

If we denote the 16 scenarios in the multiplier example by s_1, s_2, ..., s_{16} then H_{s_k} is simply a dedicated kxk-bit multiplier. We call the energy curve obtained by plotting the energies of the point systems corresponding to each scenario as the *perfect* energy-curve (denoted by $E_{perfect}$). Figure 2 compares the energy curve to the perfect curve.

A system with the perfect energy curve is called the perfect system. The perfect system can be visualized as shown in figure 3. If we take the energy costs of scenario detection and interconnect to be zero, then $H_{perfect}$ would lead to $E_{perfect}$. Since this is not possible in real world realizations, $H_{perfect}$ is only an abstraction which, as we shall see later, aids us in composing systems with enhanced power-

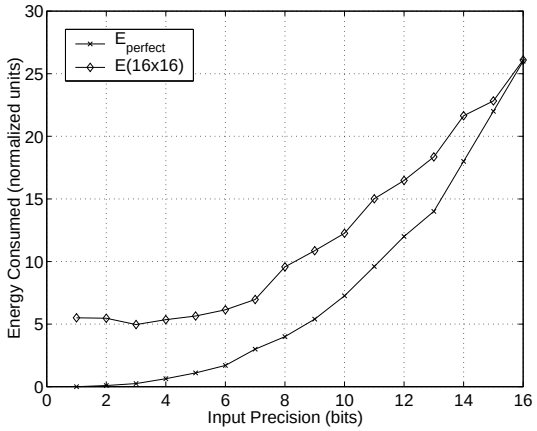

Figure 2. Comparing the 16x16 multiplier curve to the "perfect" curve

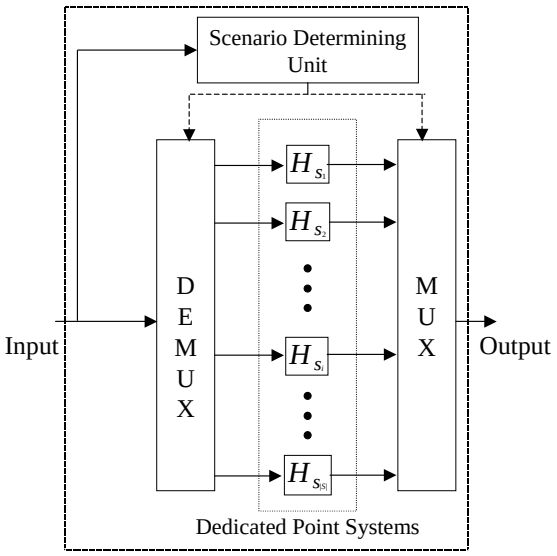

Figure 3. The Perfect System ($H_{perfect}$) viewed as an ensemble of point systems

awareness. Notice how the (hypothetical) perfect system serves as a natural bound on the ability of a real system to track scenario changes in an energy efficient manner, motivating the following definition of power-awareness,

$$\phi = \frac{\sum_{Scenarios} E(H_{perfect}, s_i) d_i}{\sum_{Scenarios} E(H, s_i) d_i} \quad (1)$$

where d is the *scenario distribution* i.e. d_i is the probability that the system will reside in a particular scenario s_i. Some thought shows that power-awareness as defined in eqn. (1) is simply the expected value of the lifetime of system H normalized to the lifetime of the perfectly power-aware system $H_{perfect}$[2]. Hence, a power-awareness of 0.5 means that the system will only last half as long as the perfect system.

3. Enhancing Power-Awareness Via Ensemble Construction

We now consider the task of enhancing power-awareness. If we replace a monolithic system by one that mimics the construction of the perfect system (figure 3), the resulting energy curve will not be perfect due to non-zero scenario detection and interconnect energy. The trade-off in constructing ensembles of point systems is clear - adding more point systems allows one to reduce the energy dissipated in executing a scenario but increases the energy dissipated in assembling these systems. Clearly, for a specified scenario distribution, there must be an optimal ensemble of point systems which strikes the right balance. For the multiplier example, figure 4 shows an ensemble of four multipliers - 16x16, 14x14, 11x11 and 9x9 which was engineered for the scenario distribution shown in figure 5. This precision distribution is exhibited by multipliers used in the speech filtering application reported in [4]. In the ensemble, a simple zero detection circuit determines the required precision and routes data to the correct multiplier. While the monolithic multiplier has an awareness of about 0.57, the 4-point ensemble has an awareness of 0.9 which would lead to a lifetime increase of about 60%.

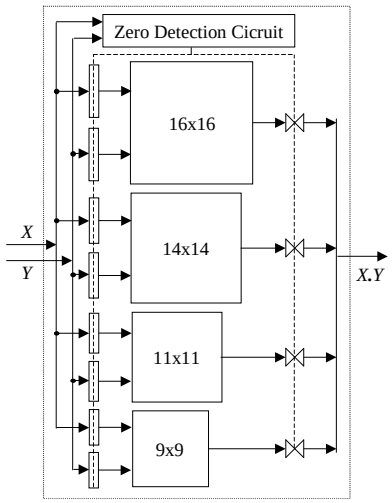

Figure 4. The 4-point ensemble multiplier system.

It turns out that finding the optimal ensemble is a com-

[2]We ignore second order effects such as dependence of battery capacity on discharge patterns

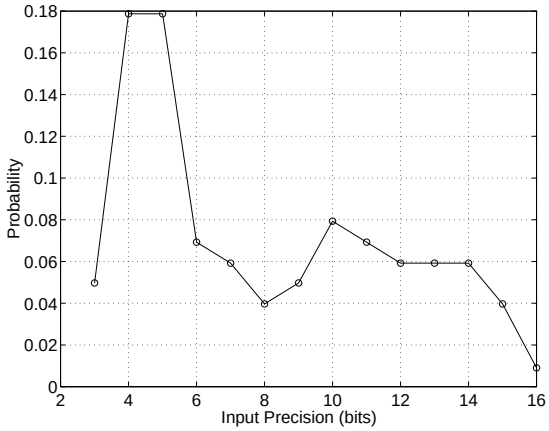

Figure 5. Multiplier scenario distribution.

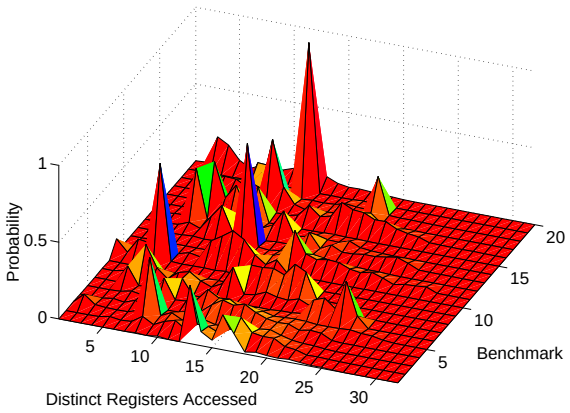

Figure 6. Register file access statistics

putationally hard problem. If we restrict valid ensembles to those with no repeated point systems the solution space is simply the power-set of P. For a specified scenario-distribution, d, precisely one of these solutions is optimal (ignoring ties). We believe that even the more relaxed problem of "Does there exist an ensemble with power-awareness greater than ϕ_0?" is computationally hard i.e. unlikely to be in the class of polynomial-time algorithms [5]. Hence, we generally resort to heuristics to search the combinatorially explosive solution space of ensembles. As we shall see in the following examples, these heuristics work well.

4. Practical Illustrations

In this section, we illustrate the technique of ensemble construction using register files, digital filters and variable-voltage processors.

4.1. Power-Aware Register Files

Architecture and VLSI technology trends point in the direction of increasing energy budgets for register files [6]. The key to enhancing the power-awareness of register files is the observation that microprocessors typically access a small group of registers repeatedly, rather than the entire register file. This locality of access is demonstrated by twenty standard benchmarks that were run on a MIPS RISC processor (fig. 6). These benchmarks show that more than 75% of the time, no more than 16 distinct registers are accessed in a 60-instruction window. Equally importantly, there is strong locality from window to window - in more than 90% cases, less than 6 registers change from the current window to the next.

If we think of the number of distinct registers the processor accesses over an instruction window as a scenario, the curves in figure 6 are simply scenario distributions. When a processor uses n registers over a window, we would want the file to behave as if it were a n-register (i.e. n-word) file. This would lead to a register file architecture which is significantly more power-aware than one where the files always behaves as a 32-register file. The reason for this of course is that smaller files have lower costs of access because the switched bit-line capacitance is lower. We model the problem of increasing the power-awareness of register files using the terminology developed earlier:

1. Set of scenarios (S): We use the number of registers accessed in an instruction window of length 60 to characterize scenarios. Picking larger windows destroys intra-window locality while smaller ones destroy inter-window locality (and also lead to higher overhead in determining and mapping scenarios).

2. Point Systems Available (P): We assume the availability of 1, 2, 4, ... 2^n word x m-bit register files.

3. Scenario Distributions (d): The register file access distributions shown in figure 6 form the scenario distributions.

4. Energy and overhead: All register file results were obtained by generating layouts using a custom-written program, extracting the layouts into SPICE netlists, and simulating the netlists in $PowerMill^{TM}$ with test vectors.

The power-awareness of a monolithic 32-word file varies between 0.25 and 0.3 for the 20 distributions. Using a (16,8,4,4) ensemble as shown in figure 7 we increase awareness to between 0.5 and 0.75 for the given distributions. The energy curves of the 32-word file and the 4-point ensemble are plotted in figure 8. Interpreted in terms of lifetime increase, the non-uniform 4-point ensemble increases lifetime by 2-3 times for the distributions used.

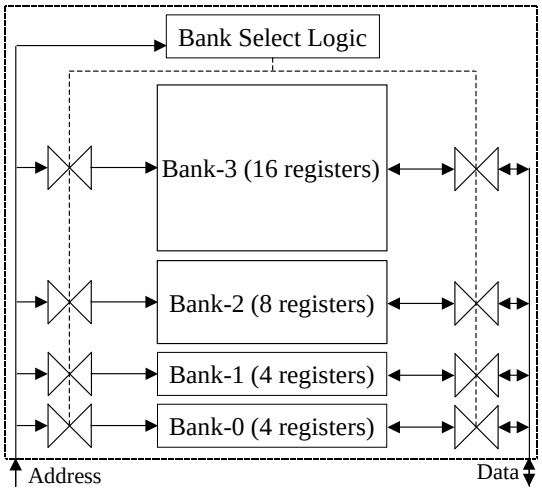

Figure 7. The (16,8,4,4) ensemble.

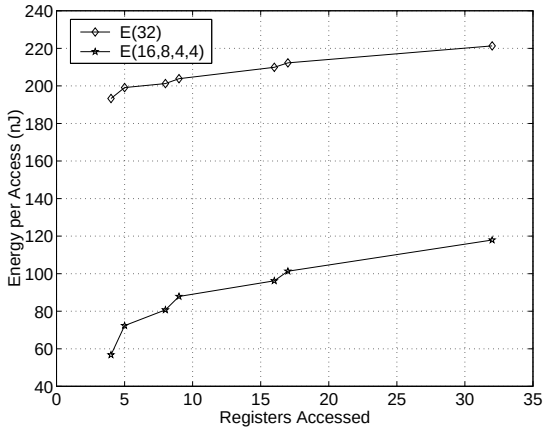

Figure 8. Energy curves of monolithic and power-aware register files.

4.2. Power Aware Filters

There are significant motivations for investigating power-aware filters. As an example, consider the adaptive equalization filters that are ubiquitous in communications ASICs. The filtering quality requirements depend strongly on the channel conditions (line lengths, noise and interference), the state of the system (training, continuous adaptation, freeze etc.), the standard dictated specifications and the quality of service (QoS) desired. All these considerations lead to tremendous scenario diversity which a power-aware filtering system can exploit [7]. We now pose the problem formally:

1. Function to be realized:

$$y[n] = \sum_{k=1}^{\text{Number of Taps}} h[k]x[n-k]$$

We have chosen a 64-tap, 24-bit filter.

2. Set of scenarios (S): We use the basis <*Number of taps*, *Precision*> to characterize the operational state that the system is in. The precision refers to both the data and coefficients.

3. Point Systems Available (P): We assume the availability of all possible < *Number of taps*, *Precision*> filters. We pick distributed arithmetic (DA) filters as described in [8] because they allow the energy to scale with both taps and desired precision.

4. Scenario Distributions (d): We model the desired filtering quality using a synthetic distribution centered around a <16-taps,8-bit> scenario. Such a distribution will prevail, for instance, when the system is in the freeze mode with a high line quality and/or low SNR requirements.

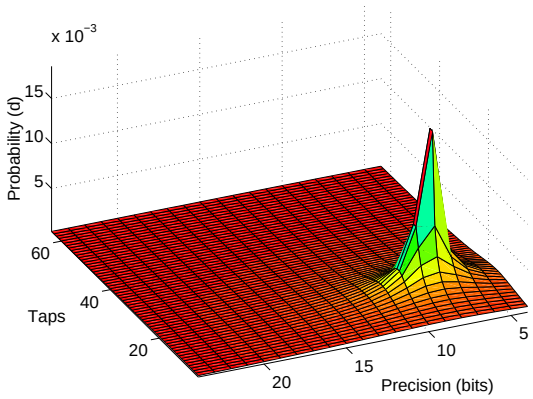

Figure 9. Example probability distribution of anticipated filter quality.

5. Energy and overhead: We parametrically model the filters described since the nature of the DA architecture lends itself to a reasonably accurate energy model [4]. Note that while energy scales about linearly with the number of taps, it scales in a quadratic manner with precision. This is because of the fact that lower precision filters can scale their voltage.

It turns out that for the distribution illustrated above, the power-awareness of a single 64-tap, 24-bit filter is only 0.17. To find more optimal ensembles, we programmed a brute-force exhaustive search algorithm that

could find the best 4-point ensemble which turned out to be $((64, 24), (64, 15), (58, 20), (51, 10))$ [3] as shown in figure 10.

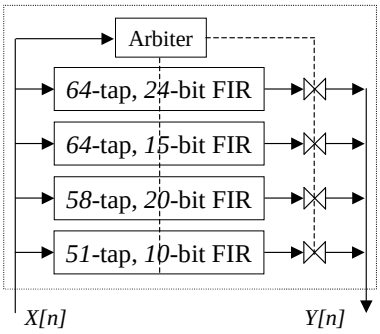

Figure 10. The optimal 4-point filter ensemble.

The 4-point ensemble has a power-awareness of 0.52, which is over three times better than the single point ensemble.

4.3. Power-Aware Processors

Having looked at three examples of power-aware subsystems (multipliers, register files and digital filters), we illustrate power-awareness at the next level of the system hierarchy - a power-aware processor that scales its energy with workload. Unlike previous examples, however, this one illustrates how an ensemble can be realized in a *purely temporal* rather than a spatial manner. It is well known that processor workloads can vary significantly and it is highly desirable for the processor to scale its energy with the workload. A powerful technique that allows such power-awareness is dynamic frequency and voltage scaling [9]. The basic idea is to reduce energy in non-worst-case workloads by extending them to use all available time, rather than simply computing at the maximum clock speed and then going into an idle state. This is because using all available time allows one to lower the frequency of the processor which in turn allows scaling down the voltage leading to significant energy savings [9, 10, 11]. In terms of the power-awareness framework that we have developed, a scenario would be characterized by the workload. The point systems would be processors designed to manage a specific workload. As the workload changes, we would ideally want the processor designed for the instantaneous workload to execute it. It is clear that implementing such an ensemble spatially is meaningless and must be done temporally using a dynamic voltage scaling system. Before we look at such a system, we state the problem more concisely.

[3] (64,24) stands for 64-tap, 24-bit precision etc.

1. Set of scenarios (S): We use the workload $\in [0, 1]$ to characterize scenarios (with 0 for no workload to 1 for a completely utilized processor). Note that the workload requirement has a one-one mapping to a frequency and voltage requirement.

2. Point Systems Available (P): A point system in this case would refer to the SA-1100 designed for a specific workload. Since we are interested in achieving power awareness through voltage scaling, this corresponds to a SA-1100 with a dedicated voltage and frequency (both set to the minimum possible to manage the workload). Also, due to an infinite number of scenarios, there are infinite number of point systems - one for every workload between 0 and 1.

3. Scenario distribution (d): We assume, for simplicity, that all workloads are equally probable. As we see below, such an assumption is pessimistic and in real applications, we can expect to see even better numbers for power-awareness.

4. Energy overhead: The energy dissipated by the SA-1100 was physically measured.

We now analyze an *actually constructed system* that recently demonstrated this power-awareness concept [10]. The overall setup is summarized in figure 11 adapted from [10]. A micro-operating system (μ-OS) running on the SA-1100 determines the current workload, scales the frequency accordingly and then instructs a switched regulator supply to scale the voltage accordingly. The reader is referred to [10] for the details of the setup and the dynamic voltage circuitry etc. The DVS system uses a *temporal ensemble* of 32 point systems with voltage levels uniformly distributed between 0 and V_{dd}^{max}. The energy-curves of a non-aware i.e. fixed voltage system and the implemented dynamic voltage system are plotted in figure 12. For uniform workload distributions, power-awareness improves from 0.63 for a fixed voltage system to 1.0 for the implemented dynamic voltage system. Note that although the 32-point ensemble is by no means perfect, it was chosen as a reference to define the power-awareness (since the *ratio* of the power-awareness of one system to the other is independent of the perfect system). Hence, for uniform load distributions, DVS leads to battery lifetime increases of about 60%.

5. Conclusions

It is clear from the preceding application examples that power-aware design can significantly enhance system lifetime of energy-constrained, battery operated systems. Understanding, characterizing and harnessing the tremendous diversity in system operation forms the cornerstone of

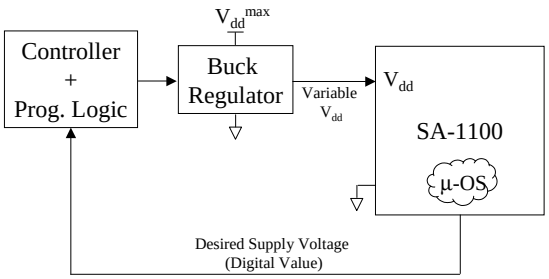

Figure 11. The dynamic voltage scaling (DVS) system used to enhance power-awareness.

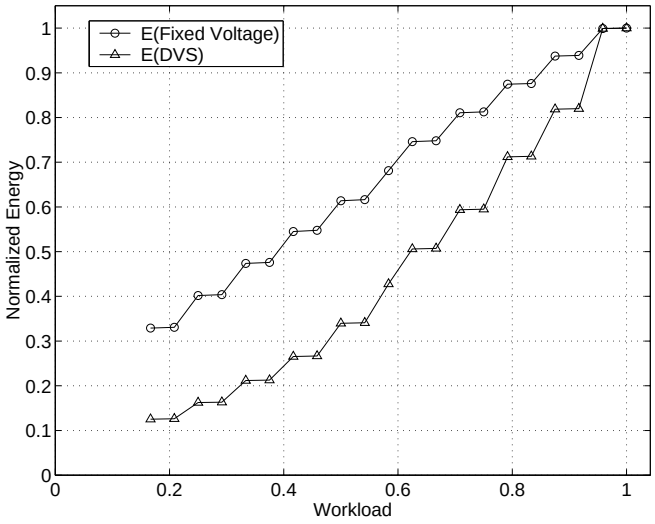

Figure 12. The energy curves of the fixed voltage and DVS system.

power-aware design. As we have shown in this paper, it is possible to rigorously quantify power-awareness and then enhance it using the technique of composing ensembles of point systems. We hope that the proposed framework will be enable designers to engineer highly power-aware systems with significantly longer battery lifetimes.

Acknowledgments

M. Bhardwaj is supported by an IBM Research Fellowship. R. Min is suported by a NDSEG Fellowship. This research is sponsored by the Defense Advanced Research Projects 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.

References

[1] A. Sinha, A. Wang, A. P. Chandrakasan, *Algorithmic Transforms for Efficient Energy Scalable Computation*, Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), 2000.

[2] S. H. Nawab, et. al., *Approximate Signal Processing* Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, vol. 15, no. 1/2, Jan. 1997, pp. 177-200.

[3] L. McMillan and L. A. Westover, *A Forward-Mapping Realization of the Inverse Discrete Cosine Transform*, Proceedings of the Data Compression Conference, March 1992, pp. 219-228.

[4] A. Sinha and A. P. Chandrakasan, *Energy Efficient Filtering Using Adaptive Precision and Variable Voltage*, 12th Annual IEEE ASIC Conference, Sept. 1999.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Company, New York, NY, 1979.

[6] V. Zyuban and P. Kogge, *The energy complexity of register files*, Proc. 1998 International Symposium on Low Power Electronics and Design (ISLPED), pp. 305-310, 1998.

[7] C.J. Nicol et. al., *A low power 128-tap digital adaptive equalizer for broadband modems*, IEEE Journal of Solid State Circuits, 32(11), pp. 1777-1789, November 1997.

[8] R. Amirtharajah, T. Xanthopoulos and A.P. Chandrakasan, *Power Scalable Processing Using Distributed Arithmetic*, Proc. 1999 International Symposium on Low Power Electronics and Design, pp. 170-175, 1999.

[9] V. Gutnik, A. P. Chandrakasan, *Embedded Power Supply for Low-Power DSP*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume 5 (4), Dec. 1997, pp. 425-435.

[10] R. Min, T. Furrer, and A. P. Chandrakasan, *Dynamic Voltage Scaling Techniques for Distributed Microsensor Networks* Proceedings of the IEEE Computer Society Annual Workshop on VLSI (WVLSI '00), April 2000.

[11] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, *A Dynamic Voltage Scaled Microprocessor System*, Proc. ISSCC 2000, pp. 294-295.