

Energy Management for Commodity Short-Bit-Width Microcontrollers¹

Rony Ghattas

Center for Embedded Systems Research
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC, 27695
rony_ghattas@msn.com

Alexander G. Dean

Center for Embedded Systems Research
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC, 27695
alex_dean@ncsu.edu

ABSTRACT

Dynamic frequency scaling and dynamic voltage scaling have been developed to save power and/or energy for general purpose computing platforms and high-end embedded systems. This paper examines the practicality of using these *advanced techniques* to save power and energy for commodity 8-bit microcontrollers while leveraging their *built-in low-power modes*. The benefits of the techniques are weighed against their complexity and cost. First, we mathematically model the power dissipation characteristics of 11 popular 8-bit microcontrollers. We then simulate their power dissipation with different power management techniques being applied. The role of the power scheduler, the power supply, and the frequency divider circuit are also described and analyzed. We find that although dynamic voltage scaling renders the lowest energy dissipation for most microcontrollers, it is not always dramatically better than using a combination of dynamic frequency scaling and the built in power down modes, which is much less expensive to implement.

Categories and Subject Descriptors

C.3 [Computer Systems Organization]: Special-Purpose and Application-Based Systems – *real-time and embedded systems*
C.4 [Computer Systems Organization]: Performance of Systems – *modeling techniques, performance attributes*

General Terms

Measurement, Performance, Design, Experimentation.

Keywords

Embedded systems, short-bit-width microcontroller, energy modeling, dynamic voltage scaling, dynamic frequency scaling

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CASES'05, September 24–27, 2005, San Francisco, California, USA.
Copyright 2005 ACM 1-59593-149-X/05/0009...\$5.00.

1. INTRODUCTION

The proliferation of portable and battery-operated computing platforms continues to push research into techniques to save power and energy. One of the defining characteristics of these devices is their requirement to operate with limited power and energy budgets. Optimization techniques have been proposed on almost all levels of a system, including the switch level, the ISA, the operating system, and the compiler [1-4].

One effective and extensively-studied method is dynamic voltage scaling, which has been demonstrated to be one of the most effective methods for minimizing the power dissipation at run time. This is because the power dissipated by CMOS-based integrated circuits is a quadratic function of the operating voltage [5]. Hence, scaling down the operational voltage linearly effectively scales down the power dissipation quadratically. Some processors have been equipped internally with dynamic voltage scaling support (e.g. Transmeta's Crusoe processor, Intel Pentium with SpeedStep [6]).

The bulk of the research available on dynamic voltage scaling, and on power management in general, addresses only high-end computing platforms (32-bit and 64-bit processors). Certain assumptions are made that might not be applicable to low-end systems (e.g. the cost of an adjustable power supply is negligible; a preemptive real-time operating system is present, etc). To the best of our knowledge, no studies have investigated power management for short-bit-width systems (e.g. 8-bit processors). However, short-bit-width processors continue to dominate worldwide microcontroller sales volumes [7-9]. Many digital consumer applications use these devices because of their low price and the availability of extensive tools and documentation. Many applications have no need for more powerful processor and cannot afford one. Examples of portable or battery operated applications range from automotive keyless entry systems, automotive theft alarms, universal remote controls, portable compact disc and digital audio players, to digital thermometers and blood pressure monitors. Moreover, market research firms generally predict a stable, if unexciting, future for short-bit-width systems. In 2006, it is expected that 8-bit units will continue to lead all microcontrollers in revenue and unit shipments [8].

¹ Based upon work supported by NSF CCR-0133690.

This paper investigates the applicability of advanced power saving methods such as dynamic voltage scaling (DVS) and dynamic frequency scaling (DFS) in combination with built-in low-power modes for short-bit-width commodity commercial off-the-shelf (COTS) processors, where some of the assumptions made in previous studies might not hold. We also try to weigh the advantages of using those techniques for low-cost embedded systems based on COTS processors versus their relatively high cost and complexity of implementation. To conduct our simulations and analysis, we use eleven 8-bit microcontrollers from the most popular architectures [8]. Complete and general mathematical models for the power dissipation characteristics of those microcontrollers are developed and listed in this study.

This paper is divided into six sections. Section 2 presents the background needed and some related issues. Section 3 presents the microcontroller sample used in our study, their characteristics, and the empirical power models derived. Section 4 presents the test bench used for the investigation and simulation assumptions. Section 5 evaluates and compares the power management techniques under consideration and list the basic results and observations reached. Finally, section 6 presents our main conclusion and gives a brief preview of our future work.

2. BACKGROUND AND RELATED WORK

In this section we present a brief review of the theories and issues related to our study.

2.1 Power Dissipation in CMOS-based Integrated Circuits

The total power dissipation in a CMOS-based circuit has been modeled as the sum of two components as given by the following [1, 5]:

$$P_{total} = P_{dyn} + P_{stat} \quad (2.1)$$

The first component, P_{dyn} , is known as the *dynamic power component* due to its dependence on the switching behavior and frequency of the circuit and has usually been modeled as follows:

$$P_{Dyn} = C_p f_{CLK} V_{CC}^2 \quad (2.2)$$

f_{CLK} is the switching frequency, V_{CC} is the operational voltage, and C_p is a constant of proportionality that is related to the effective parasitic capacitance of the logic gates with units of Farads. The second component in (2.1), namely P_{stat} , is known as the *static power component* (sometimes called the leakage component) because it is independent of the switching behavior of the circuit. It results from subthreshold and gate oxide leakage currents that depend on the circuit's operational voltage, threshold voltage, gate width, gate oxide thickness and temperature. We use a simplified second order-model due to lack of information (e.g. gate width and oxide thickness) for the devices studied:

$$P_{Stat} = S_p V_{CC}^2 \quad (2.3)$$

S_p is a constant of proportionality that depends on the conductivity properties of the circuit. This simplified model can be derived from current measurements at given voltages and frequencies. The 90% confidence interval for S_p is included in the model parameters listed in Table 2.

It has been shown that the maximum clock frequency that any CMOS circuit can operate on is proportional to the operational voltage as given by the following [1, 5, 10]:

$$f_{CLK} = \frac{K_p (V_{CC} - V_{Th})^\alpha}{V_{CC}} \quad (2.4)$$

V_{th} is the threshold voltage for the particular process used in the manufacturing of the circuit, α is approximately 1.3 and K_p is another constant that depends on the particular circuit's gate delay.

When modeling the power dissipation in CMOS circuits the static power component typically has been assumed to be negligible, when compared to the dynamic component, and hence ignored [10, 11]. Nevertheless, as Lee et. al [13] affirm, this assumption was generally valid back when the common process technology was 0.25 μm or larger. This is no longer the case as contemporary designs that use 0.13 μm layouts (deep sub-micron (DSM)) lead to large leakage currents, which make the static energy dissipation comparable to the dynamic energy dissipation. There are other reasons to account for static power dissipation. Some microcontrollers have large inefficient flash memory arrays, leading to a large static component. Many COTS microcontrollers use trailing-edge technologies. For example, Atmel has recently transitioned its microcontrollers from 0.35 μm to 0.25 μm . Hence, often it is not clear if static energy can be ignored safely. To clearly identify the optimal power saving method, both components must be accounted for, since some methods minimize only the dynamic component (e.g. DFS), while other minimize both (e.g. DVS).

2.2 Power Saving Methods

In this study we are interested in three main power saving methods (or modes): The built-in *power down mode*, *dynamic frequency scaling*, and *dynamic voltage scaling*. Each is discussed briefly below.

2.2.1 Power Down Mode (PDM)

Power down modes are implemented as a discrete set of states in which the portions of the processor have the power removed (i.e. the instruction sequencer or certain peripherals may be totally turned off) [2]. These modes usually include several variations (e.g. idle mode, sleep mode) and only depend on the particular processor under consideration. Unfortunately, some MCU manufacturers include stopping the clock as a form of "power down." This ambiguity has an impact on the effectiveness of voltage scaling. In this paper we select the lowest-power mode which keeps a timer running (enabling the processor to awaken at a specific time).

2.2.2 Dynamic Frequency Scaling (DFS)

DFS is a technique in which the processor clock is scaled down to minimize the energy dissipation linearly, as can be seen from (2.2). Note that scaling the frequency alone will reduce the processor's power dissipation but not energy since the system will take a longer time to execute its workload at that lower frequency. Nevertheless, as will be shown later, DFS can save a significant amount of power and energy for *under-utilized* systems by being applied as is, or by combining it with some of the built-in power down modes discussed above.

If the processor does not already support *DFS* internally, implementing *DFS* using external hardware is simple and cost efficient. Two techniques are possible; the first uses inexpensive simple *counter(s)* to divide the frequency by an integral value, while the second (*clock throttling*) uses logic gates to disable the clock signal periodically and is more complex [6]. A power scheduler is also needed to calculate the frequency levels required to execute the various applications (tasks or jobs).

2.2.3 Dynamic Voltage Scaling (DVS)

DVS reduces the power and energy consumed by a processor through scaling down the operating voltage, and thus the clock frequency as well (2.4). As the static component of the energy dissipated by a *CMOS* circuit is dependent on voltage but not frequency, *DVS* can lead to much higher reductions in power dissipation than *DFS*. However, *DVS* is much more complicated, and expensive to implement, as it requires several components as opposed to *DFS*.

Implementing an effective *DVS* system has several requirements, as pointed out by Burd et. al [3]: (1) A variable power supply capable of generating the required voltage levels with a high voltage transition rate, minimal transition energy losses, and a good voltage transient response, (2) a wide operational voltage range for the circuit to be powered, (3) and a power scheduler that can intelligently compute the appropriate frequency and voltage levels needed to execute the various applications (tasks or jobs) is also required.

The power supply is an essential component of a *DVS* system as it enables the voltage scaling mechanism. Two important parameters of any power supply are the transition time (also known as the tracking time), and the transition energy losses (also known as the tracking energy). The transition time is simply the time it takes to change the output voltage from one stable state to another, while the transition energy is just the amount of switching losses incurred by switching from one output voltage level to another. An equivalent metric to the transition time is the voltage transition rate (or tracking rate), which is just the difference in the output voltage levels between the two states divided by the transition time.

The main objective of any power supply is to change the output voltage from one stable state to another within a defined time, and with the minimal transition energy possible. Unfortunately, various physical and cost constraints exist that limit the achievable minimum transition time and energy. For example, as

explained by Burd et. al [14], any processor produces large current spikes which the converter's output capacitor must filter. Hence, a large output capacitor is desirable to filter and stabilize the output voltage. Nevertheless, it was shown by Burd that the transition time and transition energy are both directly proportional to the size of this capacitor. Hence, a large output capacitor implies a large time constant and more energy losses (i.e. less efficiency). Trade-offs exist in several other design dimensions, and hence, a faster converter with minimal energy losses will usually imply higher costs which might not be adequate especially for a one dollar microcontroller unit [14].

Two categories of variable voltage supplies with high transition rates, low transition energy dissipation, and good transients have been used for *DVS*. The ideal and most efficient approach uses custom-designed hardware (on-chip when possible). Two such designs were reported by Burd et. al [3], and another by Gutnic et. al [15]. Both achieve low transition energy dissipation, excellent transients, and Burd's has a voltage transition rate on the order of $50V/\mu s$. Those custom-designed ICs are clearly not an option when considering low-cost *COTS* microcontrollers since the cost of those dedicated circuits could easily cost much more than the microcontroller to be powered. The second category of variable voltage supplies are commercial *DC-DC* converters designed for *DVS*. For example, the *TPS62300* high-frequency buck converter renders high efficiency as well as good transients. However, its voltage transition rate is much smaller ($0.02V/\mu s$). At just under \$2 per device, this can cost as much as, or even several times more than the microcontroller, reducing its viability.

A final important point for implementing *DVS* is the state of the processor during the transition between voltage levels. Ideally, one would like to keep the processor running during such transitions. Nevertheless, as pointed out by Qu [16], any practical variable voltage system will have to stop instruction execution during voltage transitions until a stable steady state has been reached. This is very significant since the longer the transition takes (i.e. long transition times or equivalently small voltage transition rates), the less time the processor has to finish executing its workload. This, in fact, is the main drawback of dynamic voltage scaling since *DVS* systems with long transition times are very susceptible to the workload's granularity. The more jobs, the more transitions, and the more time spent switching, reducing the time available for actual workload execution.

Table 1. Microcontroller characteristics

Microcontroller \ Features	Program Memory	Data Memory	Other Memory	Max IOs	Max Speed	Timers	UART	SPI	I2C™	Operating Voltage Range	PDM
<i>ATmega128</i>	128KB FLASH	4KB SRAM	4KB EEPROM	53	16MHz	2 (8-bit)/3 (16-bit)	2	Yes	Yes	2.7V – 5.5V	6
<i>C8051F120</i>	128KB FLASH	8KB+256B RAM	N/A	80	100MHz	5 (16-bit)	2	Yes	Yes	2.7V – 3.6V	N/A
<i>PIC18LF8720</i>	128KB FLASH	3840B SRAM	1KB EEPROM	68	25MHz	2 (8-bit)/3 (16-bit)	2	Yes	Yes	2.5V – 5.5V	1
<i>MC68HC705C8A</i>	Up to 7744B PROM	Up to 304B RAM	N/A	31	4MHz	1 (16-bit)	1	Yes	N/A	3V – 5.5V	N/A
<i>ATmega8</i>	8KB FLASH	1KB SRAM	512B EEPROM	23	16MHz	2 (8-bit)/1 (16-bit)	1	Yes	Yes	2.7V – 5.5V	5
<i>PIC16LF877</i>	14KB FLASH	368B SRAM	256B EEPROM	33	20MHz	2 (8-bit)/1 (16-bit)	1	Yes	Yes	2.5V – 5.5V	1
<i>MC68L11D3</i>	4KB EPROM	192B RAM	N/A	26	2MHz	1 (8-bit)/1 (16-bit)	1	Yes	N/A	3V – 5.5V	N/A
<i>AT89S8253</i>	12KB EPROM	256B RAM	2KB EEPROM	32	24MHz	3 (16-bit)	1	Yes	N/A	2.7V – 5.5V	2
<i>SX20AC</i>	2KB FLASH	136B SRAM	N/A	12	75MHz	1 (8-bit)	N/A	N/A	N/A	2.7V – 3.6V	N/A
<i>ATtiny26</i>	2KB FLASH	128B SRAM	128B EEPROM	16	16MHz	2 (8-bit)	N/A	<USI>	<USI>	2.5V – 5.5V	4
<i>PIC16LF84A</i>	2KB FLASH	68B SRAM	64B EEPROM	13	20MHz	1 (8-bit)	N/A	N/A	N/A	3V – 5.5V	1

2.3 Scheduling for Power Management

A scheduler is a crucial component in implementing any power management technique. A scheduler is responsible for: (1) deciding when the processor can reduce its power consumption by using some power saving method such that its overall performance is not affected, (2) and by how much should the processor reduce its power consumption, again without affecting the overall performance. Various scheduling policies and algorithms have been proposed for both non real-time and real-time systems. These assume that a preemptive *operating system* (*OS*) is present into which the power scheduler can be incorporated. This might not be the case for low-end computing platforms where memory is scarce, and application software is much less sophisticated. In fact, several low-end embedded applications simply operate on a foreground-background (interrupt-driven) policy. Hence the use of a power scheduler for such systems becomes more complicated since no “central” program part knows what the other parts are doing. Moreover, the scheduling policy will actually set the quality of the power savings. This is a different topic, however, and we leave its analysis to future work. For now, we present a brief overview of some of the power scheduling policies used.

For non-real-time systems, most scheduling policies try to minimize the power dissipation of the system while maintaining a constant *throughput*. Individual task deadlines are not accounted for since no real-time constraints exist. Operating systems are usually present and most scheduling policies just incorporate the scheduler into the already existing *OS*. Most schedulers targeting non-real-time systems execute two main steps [11], the first step is known as the *prediction step*, while the second step is known as the *speed setting step*. The prediction step predicts how busy the *CPU* will be during some future interval by predicting its future utilization. Some of the most popular prediction algorithms are the *PAST algorithm* (where the algorithm predicts that the upcoming interval’s utilization is the same as the last interval utilization), the *AGED-a algorithm* (predicts that the upcoming interval’s utilization will be the average of all last *a* intervals), and

the *FLAT-u algorithm* (predicts that the upcoming interval’s utilization will be equal to some constant $u \leq 1$). The speed setting step then uses this information to scale the supply voltage and clock frequency accordingly. There are several popular speed-setting algorithms. In the *Weiser-style algorithm*, if utilization was high during the previous interval ($U > 70\%$) then increase the speed by 20% of the maximum for the next interval. If *U* was low during the previous interval ($U < 50\%$) then decrease the speed by 60-*U*% of the maximum speed for the next interval. In the *Peg algorithm*, if $U > 98\%$ for the previous interval, set the speed to its maximum for the next interval. If $U < 93\%$ for the previous interval, decrease speed to its minimum for the next interval. Finally, in the *Chan-style algorithm*, multiply the maximum speed by *U* of the previous interval to get the speed of the upcoming interval [11, 17-19].

For real-time systems, maintaining a constant throughput does not guarantee that individual tasks will meet their deadlines [20]. Hence, scheduling for real-time systems is much more critical. To this end, the scheduler usually is integrated into a preemptive real-time *OS* to be able to provide the needed power savings while preserving deadlines guarantees. Several policies and algorithms have been proposed in the literature for real-time systems as well. Due to space limitations, we refer the reader to existing work on power-aware real-time scheduling [10, 13, 21-24].

3. POWER DISSIPATION MODELS

A sample of short-bit-width commercial microcontrollers was chosen to investigate the applicability of power-saving methods like *DFS* and *DVS*. In this section we develop empirical models for the power dissipation characteristics of microcontrollers used. The models developed and their statistical error bounds will be listed for completeness. In order to make this analysis tractable, we do not include I/O power. Digital I/O and peripherals (serial communication, analog interfacing, timers, etc.) are used in an application-specific manner and need to be considered in a system-level analysis, but are beyond the scope of this current work.

Table 2. Microcontroller power model parameters

Microcontroller \ Model Parameters	Nom. C_P	Min. C_P	Max. C_P	Nom. S_P	Min. S_P	Max. S_P	Nom. K_P	Min. K_P	Max. K_P	Nom. Model (mW)
ATmega128	0.3739	0.3596	0.3881	0.2322	0.0935	0.3708	5.1562	4.1239	6.1884	$P_{Total}=0.3739f_{CLK}V_{CC}^2+0.2322V_{CC}^2$
C8051F120	0.1901	0.1846	0.1957	1.9783	1.6873	2.2692	51.613	27.119	76.106	$P_{Total}=0.1901f_{CLK}V_{CC}^2+1.9783V_{CC}^2$
PIC18LF8720	0.1055	0.0988	0.1122	0.2221	0.1197	0.3245	8.2201	6.6372	9.8030	$P_{Total}=0.1055f_{CLK}V_{CC}^2+0.2221V_{CC}^2$
MC68HC705C8A	0.3212	0.2790	0.3634	0.1054	0.0038	0.2070	1.1804	1.0670	1.2939	$P_{Total}=0.3212f_{CLK}V_{CC}^2+0.1054V_{CC}^2$
ATmega8	0.2073	0.1908	0.2237	0.4200	0.2600	0.5801	5.2377	4.2031	6.2724	$P_{Total}=0.2073f_{CLK}V_{CC}^2+0.42V_{CC}^2$
PIC16LF877	0.0445	0.0405	0.0484	0.1997	0.1506	0.2488	6.1159	5.3373	6.8946	$P_{Total}=0.0445f_{CLK}V_{CC}^2+0.1997V_{CC}^2$
MC68L11D3	1.2866	1.1203	1.4530	0.1401	0.0104	0.4032	0.5403	0.2026	0.8780	$P_{Total}=1.2866f_{CLK}V_{CC}^2+0.1401V_{CC}^2$
AT89S8253	0.0239	0.02	0.0279	0.498	0.461	0.522	6.436	5.1427	7.9665	$P_{Total}=0.0239f_{CLK}V_{CC}^2+0.498V_{CC}^2$
SX20AC	0.2574	0.2416	0.2732	0.8702	0.2572	1.4833	23.034	16.942	29.125	$P_{Total}=0.2574f_{CLK}V_{CC}^2+0.8702V_{CC}^2$
ATiny26	0.1681	0.1588	0.1773	0.2093	0.1189	0.2997	5.3728	4.3587	6.3869	$P_{Total}=0.1681f_{CLK}V_{CC}^2+0.20931V_{CC}^2$
PIC16LF84A	0.0386	0.0355	0.0418	0.0419	0.0044	0.0793	5.9998	5.4603	6.5393	$P_{Total}=0.0386f_{CLK}V_{CC}^2+0.0419V_{CC}^2$

3.1 Microcontroller Sample

The eleven *COTS* microcontrollers chosen represent the most popular low-end microcontroller architectures in the 8-bit market [7-9], which is dominated by the 8051, PIC and 6805 ISAs. Table 1 lists their most relevant features. The sample includes four microcontrollers of the *PicMicro* architecture, two from the 8051 architecture, two from Motorola's 6805 and 6811 architectures, and three from the *AVR* architecture. A range of microcontrollers is included, with varying memory size and I/O pin counts.

Some of the microcontrollers chosen include one or more built-in power down modes. In order to be useful for a power scheduler, it must be possible to exit the mode upon expiration of an on-board timer peripheral. Some modes can only be exited with an external interrupt or reset, making them unusable for the present work.

3.2 Power Dissipation Modeling

We briefly describe here the mathematical procedure followed in developing the power dissipation models for our microcontroller sample, and leave the detailed discussion to elsewhere [31].

We estimate of C_p and S_p in equations (2.2) and (2.3) using measurements of current at various supply voltages and clock frequencies. This data is either supplied by the manufacturers or is obtained using by experimental measurement using *MAPA*². The estimates are derived with real functional analysis [26-28], and finite-dimensional mathematical optimization [29]. Once C_p has been estimated, equation (2.4) is used to estimate our third constant of proportionality, K_p , using a similar procedure. Once all the constants of proportionality have been estimated, statistical analysis [30] is used to establish a 90% confidence interval for our estimated parameters. The eleven empirical models with error bounds on their estimated parameters calculated are listed in Table 2.

4. SIMULATIONS AND TEST BENCH

In this study we weigh the potential advantages of applying power- and energy-saving methods like *DFS* and *DVS* to low-end *COTS* microcontrollers against the cost and complexity of doing so. A simulation approach was developed and some assumptions were followed as explained below.

4.1 Simulation Assumptions

In our simulations, we do not try to compare the various microcontrollers to find which is the most energy-efficient, as this depends on many other factors not considered here (e.g. *ISA*, compiler). We only try to compare the potential benefits of using the various power-saving methods *for a given processor*. To this end, we generate execution profiles with a specific utilization level and a specific granularity level (number of jobs, or releases

(instances) of tasks) and compose our workloads from those execution profiles. Once the workloads are generated, they are passed to our simulator which uses the models developed earlier, and various other parameters (e.g. voltage transition rate for *DVS*, wake-up delay time for *PDM*) to calculate the normalized power and energy used with the various power management techniques. The minor "humps and bumps" in the plots (e.g. Figure 4) are noise resulting from the execution profiles' discrete nature and limited length, and would be eliminated with longer profiles.

4.2 Workloads

Each workload is generated from an execution profile with a specific utilization and number of tasks. Since many applications might not support a power scheduler, we divide the workloads into scheduled and unscheduled as follows.

4.2.1 Non-Power-Scheduled Workloads

Each of the workloads used has N tasks that have a utilization of U over some fixed period of time T . Applications are released (invoked) randomly while maintaining the particular utilization. The execution profile of a sample workload with a 50% utilization ($U = 50\%$) and 20 jobs ($N = 20$) is shown in Figure 1a. Note that in the absence of a real time scheduler, there is a substantial amount of idle time (where applications might be waiting on other applications, inputs, etc.) that is not utilized. Without applying any power management method, the processor will simply remain active during those idle slots consuming power while not performing any useful work.

4.2.2 Power-Scheduled Workloads

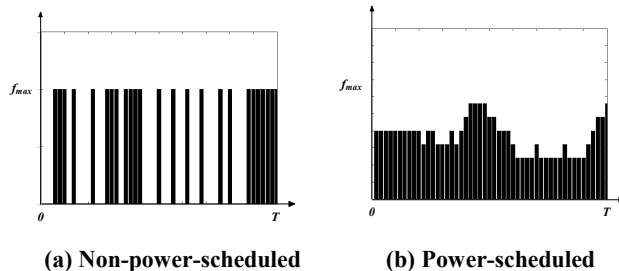


Figure 1. Example of workload's execution profile.

As was mentioned in section 2.3, the power scheduling process is crucial to any successful implementation of a power saving method. Moreover, scheduling for real-time systems is even more critical since these applications have real-time constraints that must be met for a correct and safe system operation. For this study we do not consider real-time scheduling. Instead, we simply implement a scheduler that uses the *PAST* prediction policy, and the *Chan-style* speed-setting policy, both presented in section 2.3. The scheduler calculates the utilization over some past window of time, U_{past} , and sets the current clock frequency to be $U_{past} \times f_{max}$. If the current utilization, $U_{current}$, turned out to be different from the past utilization, U_{past} , the difference, $\Delta U = |U_{current} - U_{past}|$ is added to the future utilization, $U_{future} = U_{current} + \Delta U$, and the clock is adjusted accordingly. A sample execution profile of the workload given in subsection 4.2.1 after being scheduled is shown in Figure 1b.

² The microcontroller automated power analyzer (MAPA) was constructed to measure the supply current consumed by the target microcontroller at different supply voltages and clock frequencies. It uses a simple 8-bit microcontroller, an external oscillator, a few programmable counters, and a few op-amps including a power op-amp.

4.3 Simulation Methodology

The simulation will depend on the presence or absence of a power scheduler. In the absence of such a scheduler, only PDM can be used by embedding some power down function in the application code. In the simulation, we assume such function has been implemented in every application composing the particular workload and calculate the power dissipation based on that.

In the presence of a power scheduler, PDM, DFS, or DVS can be used. We assume that when DFS or DVS are used, they are used in combination with PDM when available (e.g. DVS-PDM, DFS-PDM).

5. RESULTS AND OBSERVATIONS

5.1 Workload-Independent Power Dissipation Characteristics

Certain power dissipation characteristics are independent of the workload being executed and depend only on the microcontroller or the power management technique under consideration. These characteristics can help us decide and understand why a particular power management method is more suitable for a particular microcontroller than the other. For example, consider a microcontroller with a relatively large static power dissipation component and a “power-down” mode which in reality merely stops the clock rather than removing power. For this MCU, *DVS* is preferred as it is the only method that directly reduces the leakage component of power. As another example, a power management method with small transition times is expected to perform better with a large number of jobs (i.e. high granularity level).

5.1.1 Dynamic and Static Power Components

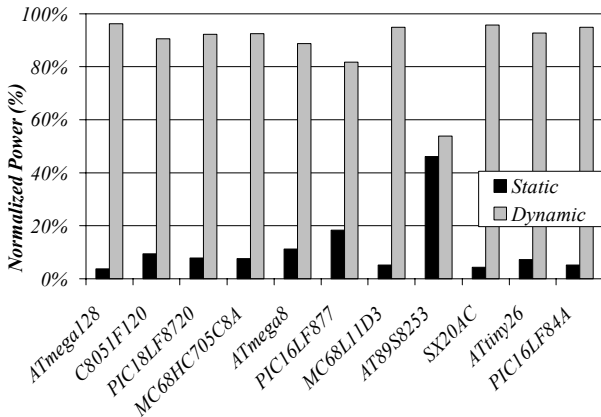


Figure 2. Normalized static and dynamic energy.

Figure 2 shows the normalized power components dissipated when the microcontrollers are running at their maximum operational voltage and frequency. Note how the *AT89S8253* has a static component that is almost as large as its dynamic component. *DVS* should render the best savings for this microcontroller as it minimizes the large static component. The other devices should see less benefit from *DVS*.

5.1.2 Minimum Power Dissipation

Figure 3 shows the power dissipated when the particular microcontroller is always idle and a single power management technique (i.e. either *PDM*, *DFS*, or *DVS*) is used all the time. This provides a bound on the amount of energy that can be saved with the various methods.

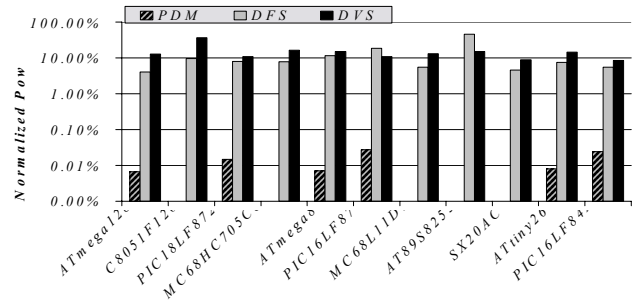


Figure 3. Minimum power dissipation

For *DFS*, we assume that the frequency dividing circuit contains an 8-bit counter which divides the operating frequency by $2^8 = 256$, so the microcontroller runs at $f_{min} = f_{max}/256$. For *DVS*, the energy dissipation shown in Figure 3 corresponds to using *DVS* to run the particular microcontroller at V_{min} (the minimal possible operational voltage for each microcontroller) with its corresponding frequency from (1.2).

Microcontroller behavior falls into one of three categories: The first is microcontrollers with a usable built-in *PDM*, which always use the least power. This is lower by a factor of at least 1000 when compared to the energy dissipated using the two other power-saving methods. In the second category, *DFS* leads to the lowest power dissipation (when *PDM* is not available) and the second lowest power dissipation (when *PDM* is available). In the third category, *DVS* leads to lower power dissipation than *DFS*. This category includes only two microcontrollers (*AT89S8253* and *PIC16LF877*). These two microcontrollers are those with the highest static energy/dynamic power ratio (which can be seen from Figure 2) and consequently, *DVS* leads to better results than *DFS* because it minimizes this large static power component while *DFS* does not.

5.1.3 Switching Power Supply

As discussed in section 2.2, the voltage transition rate is a power supply design parameter that depends on various issues. In general, the voltage transition rate will be proportional to the power converter’s design and implementation costs. This tradeoff must be evaluated by the system designer to decide whether a power converter with the required voltage transition rate will be worth its cost.

For this study, a buck-mode power supply was constructed with a goal of minimizing costs and using the embedded system’s own MCU. Based on the *ATmega16*, the system uses the device’s own analog-to-digital converter, limiting external components to capacitors, inductors, and resistors. Given the cost constraints for a viable design, a dV_{CC}/dt of only $1.95mV/\mu s$ with good transients was achieved, with a relatively low $E_{transition}$. This transition rate

was used for subsequent DVS models. For comparison, the transition rate of the TPS62300 mentioned previously is 20 mV/μs.

5.1.4 Accumulated Transition Times

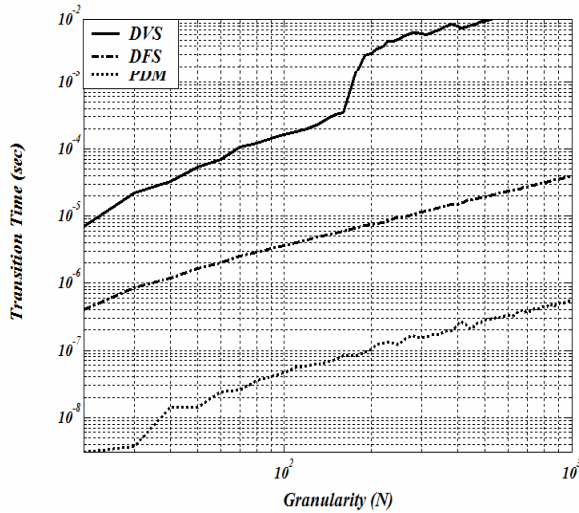


Figure 4. Transition times

Figure 4 shows the accumulated transition times (time spent switching between power-saving modes) as a function of the granularity level (i.e. the number of jobs) for *DVS*, *DFS*, and *PDM*. The transition time for *DVS* and *DFS* can be considered microcontroller independent since it will only depend on the converter’s transition rate and frequency divider circuit for *DVS* and *DFS* respectively. On the other hand, the transition time for *PDM* depends on the wake-up delay of the particular microcontroller (the *PDM* transition delay in the figure has been simulated using the *ATmega128* model as a representative microcontroller).

Note how the accumulating transition time for *DVS* has a higher rate of increase (this is a log-log plot) than both *PDM* and *DFS*. This is a drawback of *DVS* since the transition time ($|V1 - V2| / \text{voltage transition rate}$) will almost always be much larger than the transition time for *DFS* (which will have a worst case of 2^n cycles when using an n -bit counter for frequency division), or the transition time for *PDM* (which usually will fall between a several cycles to a few thousand cycles). Hence, *DVS* is usually much more sensitive to the granularity level than *DFS* or *PDM*, which makes it less suitable for high-granularity workloads.

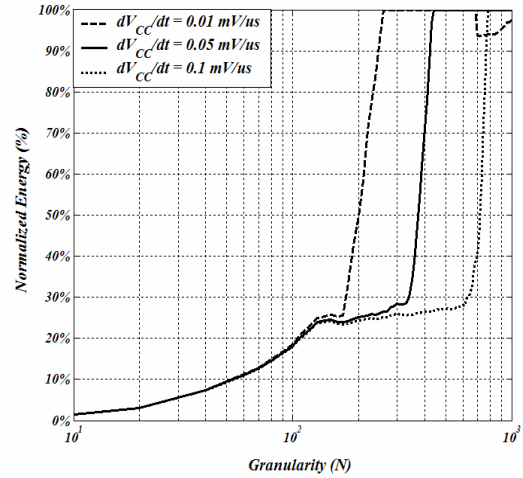


Figure 5. Voltage transition rate

5.1.5 Voltage Transition Rates

The voltage transition rate will affect the sensitivity of *DVS* to the granularity level of the workload. Figure 5 shows the normalized energy dissipated by a microcontroller using *DVS* with three converters, with voltage transition rates of 0.01, 0.05 and 0.1 mV/μs. The smaller the transition rate is, the more sensitive the microcontroller to the number of transitions, and in turn, the number of jobs. Beyond a certain point, transition energy increases rapidly. Note that in the remainder of this study we use the transition rate of 1.95 mV/μs.

5.2 Energy Use without a Power Scheduler

In the absence of a power scheduler, the system designer has no choice but the use of a built-in power down mode (if supported by the particular microcontroller). In this case, the programmer inserts power-down code that activates the microcontroller’s low power state until some waking event occurs (e.g. timer overflow interrupt, external event).

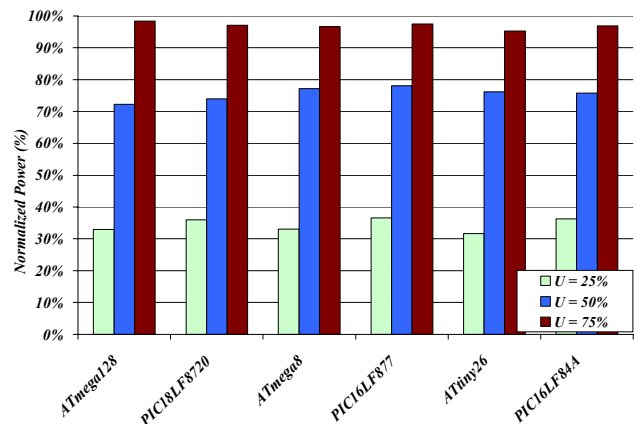


Figure 6. Energy dissipation of workload without power scheduler.

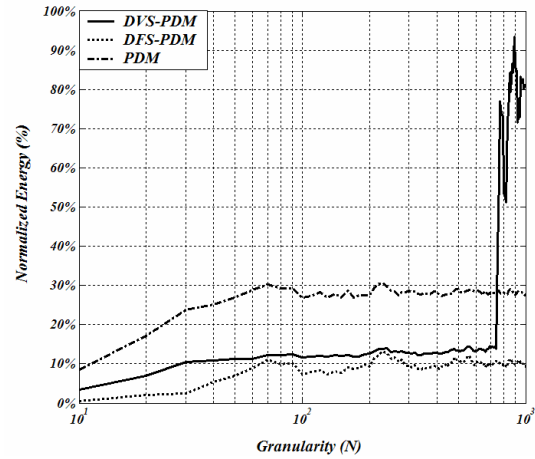
Figure 6 shows the energy dissipated per microcontroller as a function of the utilization (for the microcontrollers that support PDM mode), normalized to no power management. As the utilization increases, the energy used approaches that of a processor without any power management technique. This should be expected since power down modes can only save power while the processor is idle. As the utilization increases, the idle time decreases, and so does PDM's opportunity of reducing the system's energy consumption.

5.3 Energy Use with a Power Scheduler

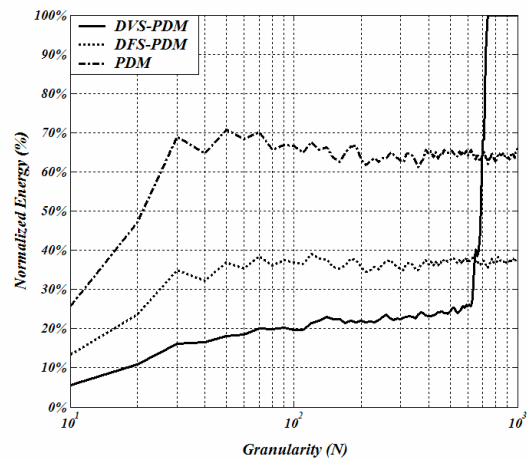
Adding a power scheduler enables power and energy savings while the processor is active, opening the door to the advantages of DVS. When analyzing the power dissipation with a power scheduler, the microcontrollers fall in one of three categories. The first and largest category includes nine microcontrollers: the ATmega128, the PIC18LF8720, the MC68HC705C8A, the ATmega8, the PIC16LF877, the MC68L11D3, the SX20AC, the ATtiny26, and the PIC16LF84A. These microcontrollers have similar characteristics. The normalized energy used by the ATmega128 (a representative of this category) is given in Figure 7a, b and c, for utilization levels of 25%, 50%, and 75% respectively.

At low utilization (25%) and low granularity, DFS-PDM uses less power than DVS-PDM. DVS-PDM is much more sensitive to the number of jobs (granularity) than DFS-PDM or PDM alone. As granularity increases, the performance of DFS-PDM and DVS-PDM become similar. Beyond a certain number of jobs (the critical granularity level), the accumulated transition time is so large that the processor cannot reduce the voltage anymore and must run at full speed and power in order to maintain its throughput. As utilization increases to 50% and 75%, DVS-PDM becomes more efficient than DFS-PDM below the critical granularity level.

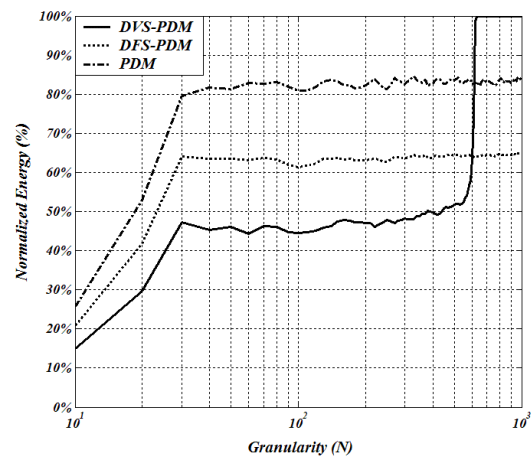
Two microcontrollers show different behavior. The energy used by the C8051F120 at utilization levels of 25%, 50%, and 75%, is shown in Figure 8a, b and c, respectively. Note that at low utilization, regardless of utilization DFS saves much more energy than DVS (90% by DFS as opposed to 70% by DVS). For medium utilization, both DFS and DVS provide energy saving of about 60% - 65%. Only for high utilization does DVS outperform DFS with energy savings of about 60% for DVS and 50% for DFS. Even then, the difference is relatively small. The exceptional behavior of the C8051F120 is due to two characteristics. First, the narrow operational voltage range (2.7 to 3.6V) limits the potential benefits of DVS. Second, the minimum frequency for DVS is limited by this narrow voltage range, when compared with the potential clock division by 256 in DFS.



(a) $U=25\%$

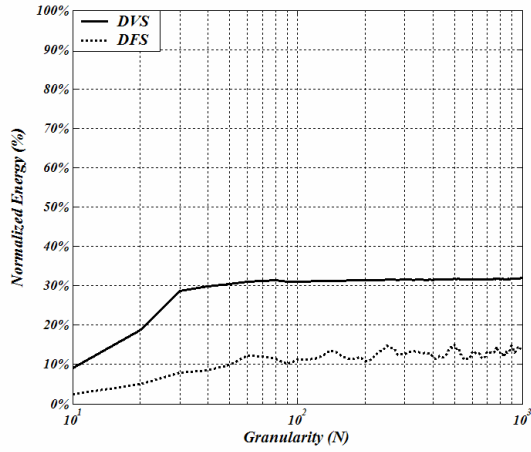


(b) $U=50\%$

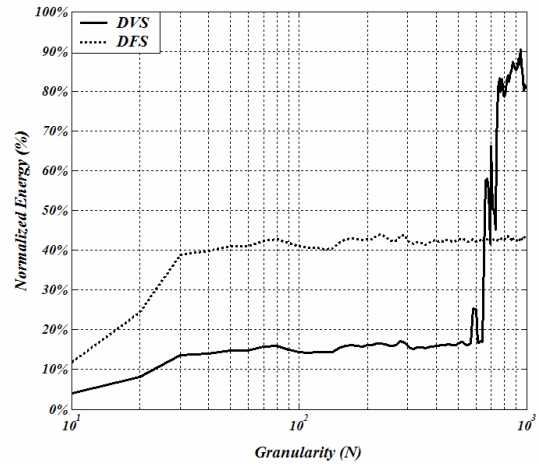


(c) $U=75\%$

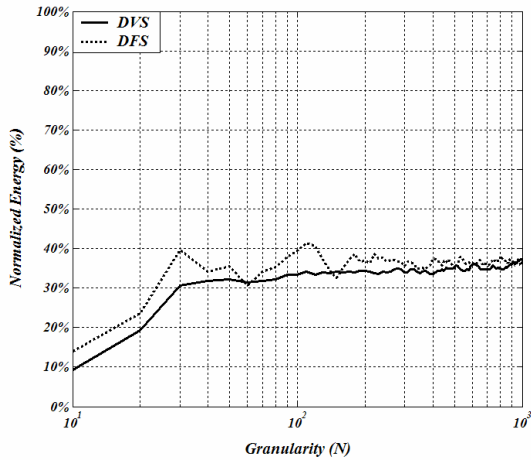
Figure 7. Normalized energy use for the ATmega128 with various utilizations



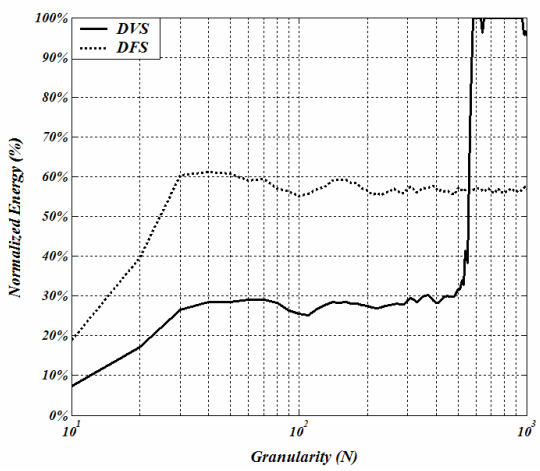
(a) $U=25\%$



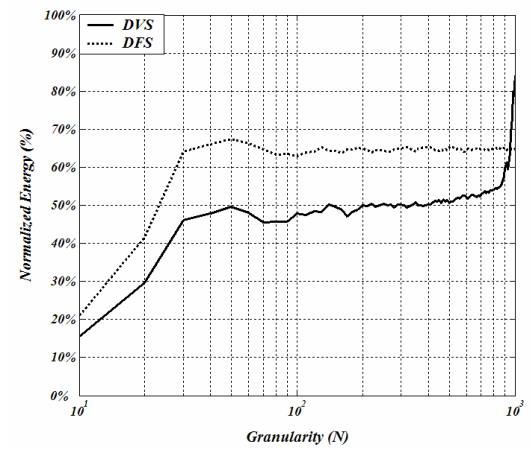
(a) $U=25\%$



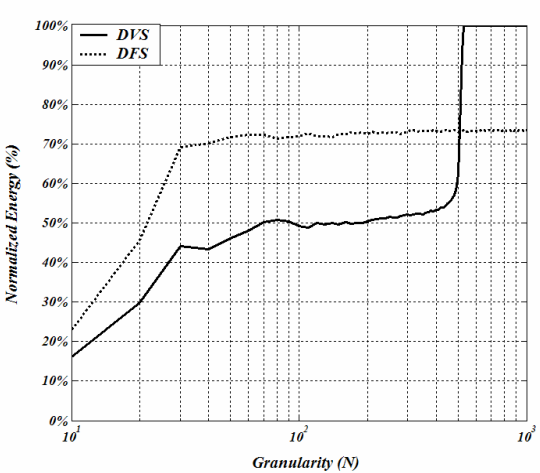
(b) $U=50\%$



(b) $U=50\%$



(c) $U=75\%$



(c) $U=75\%$

Figure 8. Normalized energy use for the C8051F120 with various utilizations

Figure 9. Normalized energy use for the AT89S8253 with various utilizations

The AT89S8253 provides an exception on the other end of the spectrum. The normalized energy used by the microcontroller for utilization levels of 25%, 50%, and 75%, are shown in Figure 9 respectively. Note how *DVS* outperforms *DFS* for all utilization levels. In fact, *DVS* provides from 20% to 30% more savings than *DFS* for all utilizations. This microcontroller has a large static power component, as explained earlier, and only *DVS* minimizes this leakage component. Hence, for this microcontroller *DVS* is a better energy management method.

6. CONCLUSIONS AND FUTURE WORK

We find that *DVS* with *PDM* generally offers improvements over *DFS* with *PDM* for most short-bit-width microcontrollers with workloads of significant utilizations (50% and above) and moderate granularity. Systems with a large static power component benefit disproportionately from *DVS-PDM*. However, for workloads with low utilizations or high granularity, the overhead of voltage scaling outweighs the energy savings, making *DFS-PDM* an attractive, cost-effective alternative. Though often significant, the energy enhancements provided by *DVS-PDM* may not justify the additional cost of a switching power supply. The designer must carefully weigh this issue. However, some microcontrollers do not fit this rule, so the designer should measure the device's power characteristics to determine the best energy-saving approach.

DVS benefits from a wide voltage range to leverage its quadratic energy savings, but for all MCUs studied this range was relatively small (at most 2). This limitation also affects 32-bit microprocessors such as the IBM PowerPC405LP, TransMeta Crusoe TM5800 and Intel XScale 80200 [32]. *DVS* also requires a fast variable power supply (i.e. large dV_{cc}/dt), but this may increase the circuit cost beyond the cost constraints, rendering it infeasible. Without fast transitions, the workload's job granularity becomes a bottleneck to saving energy. *DFS* can scale down the clock frequency by a factor of 256 with a simple 8-bit counter leading to at most a 256x power reduction with a much less expensive circuit.

Finally, for most MCUs, the dynamic energy component is still much higher than the static energy component, making *DFS* still a more attractive solution especially in our cost-constrained design space. This may change as MCUs migrate to newer fabrication processes.

For our future work, we plan to investigate applying more sophisticated scheduling techniques to low-end systems lacking RTOS features that are taken for granted for high-end systems.

7. REFERENCES

[1] T. D. Burd and R. W. Brodersen, "Energy efficient CMOS microprocessor design," *28th Hawaii International Conference on System Sciences*, Wailea, HI, 1995.

[2] J. A. Fisher, *Embedded Computing : A VLIW Approach to Architecture, Compilers and Tools*. Boston: MORGAN KAUFMANN, 2005.

[3] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A dynamic voltage scaled microprocessor

system," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 1571-1580, 2000.

[4] C. Chakrabarti and D. Gaitonde, "Instruction level power model of microcontrollers," *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, 1999.

[5] J. Rabaey, *Digital integrated circuits : a design perspective*, 1st ed. New Jersey: Prentice Hall, 1996.

[6] A. Miyoshi, C. Lefurgy, and E. V. Hensbergen, "Critical Power Slope: Understanding the Runtime Effects of Frequency Scaling," Carnegie Mellon University\Austin Research Laboratory IBM ACM 1-58113-483-5/02/0006, 2002.

[7] E. Nisley, "Rising Tides," *Dr. Dobb's Journal*, vol. 346, 2003.

[8] J. Shandle, "More for Less: Stable Future for 8-bit Microcontrollers," TechOnLine, 2004.

[9] M. Le, "8-bit microcontrollers: still going . . ." EE Times, 2004.

[10] A. Qadi, S. Goddard, and S. Farritor, "A dynamic voltage scaling algorithm for sporadic tasks," *24th IEEE Real-Time Systems Symposium*, 2003.

[11] J. L. a. A. Smith, "Improving Dynamic Scaling Algorithms with PACE," *ACM Sigmetrics*, 2001.

[12] M. Weiser, B. Welch, A. J. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," *Symposium on Operating Systems Design and Implementation*, 1994.

[13] C.-H. Lee and K. G. Shin, "On-line dynamic voltage scaling for hard real-time systems using the EDF algorithm," *25th IEEE International Real-Time Systems Symposium*, 2004.

[14] T. D. Burd and R. W. Brodersen, "Design Issues for Dynamic Voltage Scaling," *International Symposium on Low-Power Electronics Design*, Rapallo, Italy, 2000.

[15] V. Gutnik and A. P. Chandrakasan, "Embedded power supply for low-power DSP," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, pp. 425-435, 1997.

[16] G. Qu, "What is the limit of energy saving by dynamic voltage scaling?," *IEEE/ACM International Conference on Computer Aided Design*, 2001.

[17] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU," *ACM International Conference on Mobile Computing and Networking*, Berkeley, CA, 1995.

[18] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," *First Symposium on Operating Systems Design and Implementation*, 1994.

[19] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," *International Symposium on Low Power Electronics and Design*, 1998.

[20] P. Pillai and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," *18th ACM Symposium on Operating Systems Principles*, New York, 2001.

- [21] G. Quan and X. Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors," presented at *ACM/IEEE Design Automation Conference*, 2001.
- [22] A. Sinha and A. P. Chandrakasan, "Energy efficient real-time scheduling [microprocessors]," *IEEE/ACM International Conference on Computer Aided Design*, 2001.
- [23] Y. Z. a. F. Mueller, "Feedback Dynamic Voltage Scaling DVS-EDF Scheduling: Correctness and PID-Feedback," *Workshop on Compilers and Operating Systems for Low Power*, 2002.
- [24] H. Singh, "Scheduling techniques for real-time applications consisting of periodic task sets," *IEEE Workshop on Real-Time Applications*, 1994.
- [26] E. Kreyszig, *Introductory functional analysis with applications*. New York: Wiley, 1978.
- [27] W. Rudin, *Real and complex analysis*, 3rd ed. ed. New York: McGraw-Hill, 1987.
- [28] B. S. Thomson, J. B. Bruckner, and A. M. Bruckner, *Elementary real analysis*. Upper Saddle River, N.J.: Prentice-Hall, 2001.
- [29] E. K. P. Chong and S. H. çZak, *An introduction to optimization*, 2nd ed. New York: Wiley, 2002.
- [30] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers*, 2nd ed. New York: Wiley, 1999.
- [31] R. Ghattas and A. Dean. " Empirical Analysis and Modeling of Power Saving Modes in the Most Popular 8-Bit MCUs " CESR Technical Report, Available online: <http://www.cesr.ncsu.edu/agdean/TechReports/PowerModeling.pdf>
- [32] B. Zhai, D. Blaauw, D. Sylvester and K. Flautner, "Theoretical and Practical Limits of Dynamic Voltage Scaling," *ACM/IEEE Design Automation Conference (DAC)*, June 2004.