Energy Management for Commodity Short-Bit-Width Microcontrollers

Abstract – Power- and energy-saving techniques such as dynamic frequency scaling and dynamic voltage scaling have been developed targeting long-bit-width, high-end embedded systems, and general purpose computing systems. However, these techniques are not necessarily applicable to cost-sensitive short-bit-width, low-end, embedded systems. This paper examines the practicality, of applying these advanced energy-saving techniques to commodity short-bit-width microcontrollers, as well as comparing them to the built-in energy-saving methods, if available. First, we model mathematically the power dissipation characteristics of eleven common commodity *8-bit* microcontrollers. Then, we find their appropriate energy-saving approaches by comparing the built-in energy-saving technique, dynamic frequency scaling, and dynamic voltage scaling. We find that, when available, the built-in energy-saving method is the most efficient and obviously the least expensive. Otherwise, dynamic frequency scaling is usually the most efficient and cost effective. Dynamic voltage scaling, however, is of limited value due to the relative expense of an efficient adjustable power supply, and the restricted operating voltage range of most short-bit-width, commercial-of-the-shelf microcontrollers.

I. INTRODUCTION

Minimizing the energy consumption of microprocessor-based systems is a field of active research. Given an arbitrary workload, at what voltage and frequency should we run the processor for the "best" performance [1, 2]? How do we quantify "best" – the overall energy consumption? The speed at which the workload's computations are executed? Or is it a combination of both [3-5]? For CMOS *integrated circuits* (ICs), power dissipation has dynamic and static terms [6-8]:

$$P_{Total} = C_P f_{CLK} V_{CC}^2 + S_P V_{CC}^2 \quad (1.1)$$

Energy can be saved by idling the processor, using the built-in *power-down mode* (*PDM),* if available, when no activity is needed. The frequency of the processor can be scaled down to reduce the energy dissipated (1.1). Otherwise, both the processor's voltage and frequency can be reduced; they are related as follows for CMOS ICs [6-8], where V_{th} is transistors *threshold voltage*:

$$f_{CLK} = K_P \left(V_{CC} - V_{th}\right)^2 / V_{CC} \quad (1.2)$$

To this end, the energy savings will be quadratic with respect to voltage reductions, and linear with frequency.

Existing research makes certain assumptions about the system characteristics: voltages can be scaled by a factor of four or more due to a wide voltage operating range, the cost of an adjustable power supply is negligible, a preemptive real-time operating system is present, etc.

Unfortunately, these assumptions become questionable when considering low-cost embedded applications built around short-bit-width, low-end, commodity *commercial-of-the-shelf* (*COTS*) microcontrollers. *Four-bit* and *eight-bit microcontrollers* (*MCUs*) account for *75%* of the devices sold [9], yet are not covered by any of this research. Operational supply voltage ranges for these *MCUs* are limited, adjustable power supplies may be expensive in comparison with a $1 *MCU*, and a foreground-background (i.e. interrupt driven) system is usually used instead of a preemptive real-time operating system due to the limited amount of *random access memory* (*RAM*).

There are two goals for this paper. The first is developing empirical mathematical models for some popular *8-bit MCUs*, which are general enough, and can be reused by embedded systems engineers when needed. For example, the static power component in (1.1) was usually neglected in several studies due to its magnitude when compared with the dynamic power component [8, 10, 11]. Nevertheless, as Lee et. al. affirms [5], this assumption was generally valid back when the common process technology was 0.25µm or larger. This is no longer the case as contemporary designs that use 0.13 µm layouts, referred to *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 for the need to account for static energy dissipation like some *MCUs* with flash memory that has a large static component. Furthermore, many *COTS MCUs* use trailing-edge technologies, so it isn't clear if static energy can be ignored safely. Our models, therefore, account for the dynamic and the static energy dissipation, and bound the parameters modeling error using some procedures from mathematical statistics. The second and main goal is to use the general models developed to determine which energy management approaches are the most practical in this cost-constrained design space.

This paper is divided into six sections. Section II discusses related work. Section III presents the *MCUs* used in our study, their characteristics, and the empirical power models derived. Section IV presents the simulation methodology used. Section V evaluates and compares the different energy-saving techniques. Section VI presents the main conclusions of the paper.

II. RELATED WORK

Existing work on energy conservation for processors includes methods on the circuit level (i.e. enabling changes to the system voltage and/or frequency) and the software scheduler level (i.e. choosing the best operating voltage and/or frequency given the system's real-time constraints, if any). This section briefly describes some of these issues found in the literature.

2.1. Dynamic Frequency Scaling: *DFS* is a technique where the processor clock is scaled down to minimize the energy consumption (1.1). Implementing *DFS* using external hardware, when the processor does not already support *DFS* internally, is simple and cost efficient. Two main techniques are the most common: The first uses inexpensive simple *counter(s)* to divide the frequency by an integral value, while the second, called *clock throttling*, uses logic gates to disable the clock signal periodically and is more complex [12].

2.2. Dynamic Voltage Scaling: *DVS* reduces the energy consumed by a processor through scaling down the operating voltage, and hence the frequency too (1.1, 1.2). The static component of the energy dissipated by any CMOS based IC is dependent on the voltage level only and not the frequency (1.1), consequently, *DVS* can lead to much higher reductions in energy dissipation [12].

However, as pointed out by Burd et. al. [1], *DVS* requires three components for successful implementation: (1) A variable power supply capable of generating the required voltage levels with a high voltage transition rate, dV_{CC}/dt, minimal transition energy, $E_{transition}$, dissipation, and good transients, (2) a wide operational voltage range [38], (3) and a scheduler that can intelligently compute the appropriate frequency and voltage needed to execute the various jobs is also required. Several high-end processors have already been equipped internally with *DVS* like Transmeta's Crusoe processor, and Intel Pentium with SpeedStep [12].

We find that for short-bit-width commodity *COTS MCUs* with nonnegotiable cost constraints, *PDM* (when available) and *DFS* can lead to similar, and usually higher, reductions in the energy dissipation than *DVS* for arbitrary workloads.

2.2.1. The Variable Voltage Supply: The variable voltage supply is an essential component to implement *DVS*. Two of its important parameters when used for *DVS* are the *voltage transition rate*, dV_{CC}/dt, and the *transition energy dissipation*, $E_{transition}$.

Two main categories of variable voltage supplies with high, dV_{CC}/dt, low $E_{transition}$, and good transients have been used for *DVS*. The ideal and most efficient approach is the use of custom designed hardware (on-chip when possible). Two such designs where reported in the literature by Burd et. al. [1], and another by Gutnic et. al [13], both achieved low $E_{transition}$, excellent transients, and the one by Burd had a dV_{CC}/dt on the order of *5000V/msec*. Those custom-designed ICs are clearly not an option when considering *COTS MCUs* since the cost of those dedicated circuits could easily cost much more than the *MCU* to be powered.

The second category of variable voltage supplies are commercial DC-DC converters designed for *DVS*. The TPS62300 high-frequency buck converter [14], for example, renders low $E_{transition}$ as well as good transients. However, its dV_{CC}/dt is much smaller and is about *50V/ms*. These devices are less expensive than custom IC designs, but they can still cost, from *1 to 2 times* the cost of the *MCU,* reducing their appeal.

For the above reasons, a *DVS* system was constructed by our group trying to minimize the costs for a simple *ATmega16* (*8-bit MCU* by Atmel®). The system uses the target *MCU's* analog-to-digital converter, minimizing the cost of the external components to capacitors, inductors, and resistors. However, given the cost constraints for a viable design, a dV_{CC}/dt of only *1.95V/msec* with good transients was achieved*,* with a relatively low $E_{transition}$.

2.3. Scheduling for Energy-Saving: The number of studies on energy-saving scheduling algorithms for both non-real-time, and

real-time, systems is large. Because of space limitations we refer the interested reader to some good references [2, 3, 8, 15-17].

III. MICROCONTROLLER ANALYSIS AND EMPERICAL MODELS DEVELOPMENT

In this study, a sample of short-bit-width, *COTS MCUs* were chosen to investigate and compare the different energy-saving techniques and their applicability to this forgotten category of *MCUs*.

3.1. MCUs Sample: Eleven short-bit-width commodity *COTS MCUs* were chosen to represent the most popular *8-bit MCUs*. The sample includes four MCUs of the PicMicro architecture: the PIC18LF8720, PIC16LF877, and the PIC16LF84A by Microchip® and the SX20AC by Ubicom. Two are from the 8051 architecture: the C8051F120 by Silicon Labs® and the AT89S8253 by Atmel®. Two more are from Motorola®'s 6805 and 6811 architectures. The last three members of our sample are from the AVR architecture: the ATmega128, ATmega8, and the ATtiny26 by Atmel®.

Moreover, some of the MCUs investigated include a built-in energy-saving mode, *referred to as power-down-mode* (*PDM)*, which we also investigate in this study. Table 3.1. lists the eleven members of our sample with most of their relevant features.

3.2. MCU Power Consumption Modeling: In this subsection we develop empirical mathematical models to represent *MCU* power dissipation. We also present briefly the mathematical procedure followed in developing these models.

In developing the power dissipation models for the different *MCUs*, either the data supplied by the manufacturer (i.e. the overall current consumption at different clock frequencies and supply voltages [18-26]), or similar data obtained using *MAPA*[*] were used to calculate an estimate of C_p and S_p in (1.1), using some theories and procedures from two main areas of mathematics, namely, *real functional analysis* [27-29], and *finite-dimensional mathematical optimization* [30, 31]. Once C_p has been estimated, equation (1.2) is used to estimate the third constant of proportionality using a similar procedure while assuming that the threshold voltage, V_{th}, for all *MCUs* is *0.9V*. Once all the constants of proportionality have been estimated, some procedures from *mathematical statistics* [32-34], were 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 3.2.

IV. SIMULATION AND ANALYSIS METHEDOLOGY OF ENERGY SAVING TECHNIQUES UNDER INVESTIGATION

In this section, we discuss the simulation methodology used in the next section, as well as state some of the main assumptions that were followed for the simulations and the analysis of the results.

4.1. Assumptions and Simulation Methodology:

In this paper, we evaluate the *best method to save energy* for a given workload on a short-bit-width, low-cost, *COTS MCU*. *We do not try to compare MCUs to find which is the most energy-efficient* as this depends on other factors not considered here including the *MCU's* ISA and the particular compiler used. We therefore assume that all *MCUs* execute a given workload in the same number of cycles, and leave the more detailed comparison to future work. It will also be assumed that at any instant during the execution of some arbitrary workload, if the processor has no work to be done, some energy-saving technique will be used to save energy.

Built-In Energy-Saving Modes (PDM): here we use the built-in *PDM*, (if any are available), to minimize energy dissipation.

Dynamic Frequency Scaling (DFS): here the clock frequency of the microcontroller is scaled down by some integral value, either using an on-chip frequency divider, or using external hardware. In the simulations, we assume the use an *8-bit* counter to scale down the clock by an integral factor of *1,2,3,…,256*. It will also be assumed that the transition delays due to switching between different frequency levels are equal to the average case delay (one-half the scaling factor). When the processsor is idle, DFS cuts energy by dividing the clock by 256.

Dynamic Voltage Scaling (DVS): here the supply voltage of the microcontroller is scaled down using external hardware, this in turn will also scale down the operating frequency according to equation (1.2). Any intermediate voltage level needed to run the microcontroller at a certain frequency is calculated using (1.2). If a particular frequency level requires the processor to operate at V_{ideal}, the actual voltage used will be *ceiling(10* $\times$ V_{ideal}*)/10* (e.g. a voltage of 4.43V will be rounded to 4.4V). The maximum and minimum voltages, V_{max}, and V_{min}, will refer to the two ends of the operational voltage range for each *MCU* respectively. Moreover, for *DVS*, the transition delay is calculated using the product of dV_{CC}/dt (in our case *1.95V/ms*) and the voltage difference between the two voltage levels we are switching between. We account for $E_{transition}$ by assuming that the *MCU* is non-idle for all transition delay periods.

4.2. Main Differences between PDM, DFS, and DVS:

PDM can only switch the microcontroller between two states: the full-throughput state and the minimum energy dissipation state. So microcontrollers using *PDM* will always run at full throughput even when executing jobs requiring less than full-throughput. Unlike *PDM*, *DFS* and *DVS* allow the processor to operate at intermediate states between full-throughput and minimum energy dissipation. The number of these intermediate states depends on the frequency prescalar (*256* in our case) for *DFS*, and the voltage range and resolution (0.1V resolution in our case) for *DVS*. These observations are summarized in table 4.2.

Workload/ Energy Management Technique	*No Jobs are Executing*	*Jobs Requiring Intermediate Throughput*	*Jobs Requiring Full-Throughput*
None	$V_{CC} = V_{max}$ $f_{CLK} = f_{max}$	$V_{CC} = V_{max}$ $f_{CLK} = f_{max}$	$V_{CC} = V_{max}$ $f_{CLK} = f_{max}$
PDM	$V_{CC} = V_{max}$ $f_{CLK} = 0$	$V_{CC} = V_{max}$ $f_{CLK} = f_{max}$	$V_{CC} = V_{max}$ $f_{CLK} = f_{max}$
DFS	$V_{CC} = V_{max}$ $f_{CLK} = f_{min}$	$V_{CC} = V_{max}$ $f_{CLK} = f_{Ji}$	$V_{CC} = V_{max}$ $f_{CLK} = f_{max}$
DVS	$V_{CC} = V_{min}$ $f_{CLK} = f_{Vmin}$	$V_{CC} = V_J$ $f_{CLK} = f_{VJ}$	$V_{CC} = V_{max}$ $f_{CLK} = f_{Vmax}$

Table 4.2. Main Differences between Built-In Energy Saving Modes, DFS, and DVS

4.3. The Normalized Energy-Transition-Delay Product (NETDP) Metric: To compare the different energy-saving techniques investigated in this study, we define a metric that accounts for the energy dissipation of the particular microcontroller using the particular energy-saving technique, and how that technique affects the system's performance. Inspired by the work of [4, 35], we add a term to emphasize the impact of the time spent in switching between different energy-saving states. The *NETDP* metric was defined as $NETDP = NE \times NTD$, where *NE* stands for *normalized energy*, and *NTD* for *normalized transition delay*. These terms are defined as follows for a workload, W, that takes a period of t_W seconds to execute and would consume $P_W^{full\text{-}throughput}$ if it was to run at full-throughput for the complete t_W seconds. The normalized energy, *NE*, is defined as follows:

$$NE = \frac{\left[\sum_{J \in W} t_J^{non-idle} \times P_J^{non-idle}\right] + \left[t_W^{idle} \times P_W^{idle}\right]}{t_W \times P_W^{full-throughput}} \qquad (4.3.1)$$

where $P_J^{non\text{-}idle}$ is the power dissipated by the *MCU* during the execution of job $J \in W$, which takes $t_J^{non\text{-}idle}$ seconds to execute. On the other hand, P_W^{idle} is the energy dissipated by the *MCU* when in some energy-saving state, and remains in this state for t_W^{idle} seconds. Similarly, the normalized delay, *NTD* , is defined as follows:

$$NTD = \frac{D_W^{transition}}{t_W} + 1 = \frac{\sum_{J \in W} D_J^{transition}}{t_W} + 1 \qquad (4.3.2)$$

4.4. Benchmark Used for Power Consumption and Performance Comparison of Different Energy-Saving Techniques: We present here the set of workloads that were developed to compare the energy dissipation characteristics of the different *MCUs* when using the different energy-saving techniques under consideration. Most embedded applications of interest to us are referred to as *time-constrained embedded applications*. As Fornaciari et. al. [36] points out, time-constrained embedded applications are those systems where speed is major design constraint. Time-constrained computing applications can be divided into three main categories depending on the system's required throughput [36, 37]:

Fixed Throughput Computational Workloads: Applications where the number of jobs executed over any fixed period of time, denoted by t_W, is fixed. However, as opposed to the burst throughput mode discussed later, not necessarily all jobs require maximum throughput to execute correctly. Systems operating in

[*] The microcontroller automated power analyzer, or 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.

MCU \Features	Program Memory	Data Memory	Other Memory	Max IOs	Max Speed	Maximum MCU Supply Current in PDM (μA)	Transition Delays (Clock Cycles)	Operating Voltage Range
ATmega128	128KB FLASH	4KB SRAM	4KB EEPROM	53	16MHz	2	6	2.7V – 5.5V
C8051F120	128KB FLASH	8KB+256B RAM	N/A	80	100MHz	N/A	N/A	2.7V – 3.6V
PIC18LF8720	128KB FLASH	3840B SRAM	1KB EEPROM	68	25MHz	2	1024	2.5V – 5.5V
MC68HC705C8A	Up to 7744B PROM	Up to 304B RAM	N/A	31	4MHz	N/A	N/A	3V – 5.5V
ATmega8	8KB FLASH	1KB SRAM	512B EEPROM	23	16MHz	1.25	6	2.7V – 5.5V
PIC16LF877	14KB FLASH	368B SRAM	256B EEPROM	33	20MHz	1.5	1024	2.5V – 5.5V
MC68L11D3	4KB EPROM	192B RAM	N/A	26	2MHz	N/A	N/A	3V – 5.5V
AT89S8253	12KB EPROM	256B RAM	2KB EEPROM	32	24MHz	N/A	N/A	2.7V – 5.5V
SX20AC	2KB FLASH	136B SRAM	N/A	12	75MHz	N/A	N/A	2.7V – 3.6V
ATtiny26	2KB FLASH	128B SRAM	128B EEPROM	16	16MHz	1.1	6	2.5V – 5.5V
PIC16LF84A	2KB FLASH	68B SRAM	64B EEPROM	13	20MHz	1	1024	3V – 5.5V

Table 3.1. MCUs Examined in the Study and their most Relevant Features and Characteristics

MCU\ 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	$P_{Total}=0.2574f_{CLK}V_{CC}^2+0.8702V_{CC}^2$
ATtiny26	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	$P_{Total}=0.0386f_{CLK}V_{CC}^2+0.0419V_{CC}^2$

Table 3.2. MCU Empirical Models Developed and the Estimated Parameters with a 90% Confidence Interval

this mode are predominantly found in digital signal processing and controls applications where the number of jobs and the required throughput are fixed by the rate of incoming or outgoing signals. A graphical representation of this model of workloads is given in figure 4.4.1*a*.

Maximum Throughput Computational Workloads: Applications where the number of jobs executed over any fixed period of time is fixed at the maximum possible throughput. Predominantly, all multi-user systems, like super computers, networked desktops, and servers, belong to this category since the processor is continuously running at its maximum throughput to meet all users computational demands and workloads. Figure 4.4.1*b* gives a graphical representation of such a workload model.

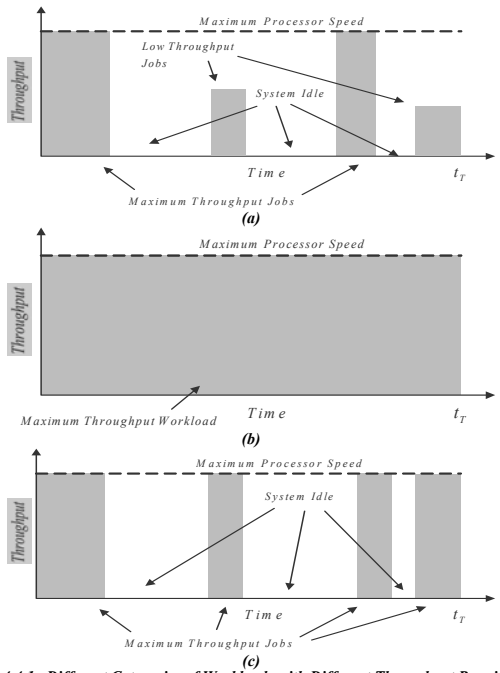


Figure 4.4.1. Different Categories of Workloads with Different Throughput Requirements

Burst Throughput Computational Workloads: Those are characterized by jobs that can only execute at full throughput over any fixed period of time. Nevertheless, during that fixed period of time, if no job is executing, zero throughput will suffice (at least in theory). Most systems interfacing with users or waiting for external aperiodic events to occur belong to this category, which includes various embedded applications. A graphical representation of such workloads model is given in figure 4.4.1*c*.

4.5. Simulation Benchmark: As can be seen from the above models of workloads, only fixed and burst throughput workload models are applicable to embedded systems. Therefore, the simulation benchmark developed only includes those two models of workloads.

Simulation Workloads of Fixed Throughput Model:
Subset I of our simulation workloads is composed of twelve synthetic workloads developed to represent the *fixed throughput workload model* as follows. Each of the twelve workloads will have a set of N jobs, $J_1, J_2, \ldots, J_N$, that need to be executed over some fixed period of time, t_W , and have a utilization U_W. For each workload in the subset, *N/4* jobs require full-throughput execution. *N/4* jobs require *70%* of full-throughput. *N/4* jobs require *45%* of full-throughput, *N/8* jobs require *25%* of full-throughput, and the remaining *N/8* jobs require *10%* of full-throughput. A list of the full-throughput utilization and granularity of the twelve workloads constituting this subset of workloads is presented in table 4.5.1[†].

Simulation Workloads Burst Throughput Model:
Subset II of our simulation workloads is composed of twelve synthetic workloads developed to represent the burst-throughput workload model as follows. Each of the twelve workloads will have a set of N jobs, $J_1, J_2, \ldots, J_N$, all need to be executed at full-throughput over some fixed period of time, t_W, and have a utilization, U_W. A list of the utilization and granularity of the

[†] We need to note that the utilizations in table 4.5.2.1. are not the actual system utilizations. Instead, these utilization values represent the utilization levels if all the jobs were to execute at full throughput which, for subset I, is obviously not the case.

twelve workloads constituting this subset of workloads is presented in table 4.5.2.2[‡].

Simulation Methodology: We will assume that between any two jobs, say J_m and J_n with $m, n \in \{1, 2, \ldots, N\}$, there exists a period where the processor has no jobs to execute and our energy management techniques can be used during those time intervals. *Note that no generality is lost by assuming that between any two jobs there is a period were the CPU is idle since if any number of jobs need to run consecutively, we will simply end-up calculating a larger total transition delay that can be thought of as worst-case bound on the transition delays for the particular granularity level.*

Utilization/ Granularity	U_1=20%	U_2=40%	U_3=60%	U_4=80%
N_1=10 Jobs	WA11	WA21	WA31	WA41
N_2=50 Jobs	WA12	WA22	WA32	WA42
N_3=100 Jobs	WA13	WA23	WA33	WA42

4.5.2.1. Fixed-Throughput Workloads and their Different Utilizations and Granularities

Utilization/ Granularity	U_1=20%	U_2=40%	U_3=60%	U_4=80%
N_1=10 Jobs	WB11	WB21	WB31	WB41
N_2=50 Jobs	WB12	WB22	WB32	WB42
N_3=100 Jobs	WB13	WB23	WB33	WB42

4.5.2.2. Burst-Throughput Workloads and their Different Utilizations and Granularities

V. SIMULATIONS AND OBSERVATIONS

In this section, we evaluate the *MCU's* energy characteristics for different modes and then simulate the benchmark developed in section *IV* and compare the different energy-saving techniques when applied to short-bit-width *MCUs*.

5.1. MCUs Energy Consumption Characteristics that are Independent of the Particular Workload: Certain energy-consumption characteristics are independent of the workload being executed and depend only on the *MCU's* power consumption characteristics. These include the *full-throughput energy dissipated* over some fixed period of time, the *idle energy dissipated* over some fixed period of time, and the *transition delays* between different energy dissipation states. The latter two depend on the energy-saving method used as will be seen shortly.

Full-Throughput Dynamic and Static Energy Dissipation Components: Figure 5.1.1 shows the full-throughput (i.e. *MCUs* are running at their maximum operational voltage and frequency) energy components dissipated over a period of one second. In general, the dynamic energy is at least 10x the static energy. The SX20AC, and the C8051F120 have the highest absolute dynamic energy dissipation components. As shown in table 3.1, both of these microcontrollers have the two largest maximum operating frequencies, which the dynamic energy dissipation component in (1.1) depends on. So, the energy-saving method that will run the *MCU* at the lowest frequency would be the most appropriate.

Minimum Energy Dissipation when using PDM, DFS, and DVS: This subsection investigates the energy dissipated by the particular *MCU* when it is in its lowest energy dissipation state. This state will obviously depend on the energy management technique used. For *DFS*, this is the *MCU* operating at $f_{min} = f_{max}/256$. For *DVS*, this is the *MCU* operating at V_{min} with its corresponding frequency from (1.2). The total energy dissipated by the various *MCUs* in this state is plotted in figure 5.1.2. for the different energy management techniques. The *MCUs* fall in one of three categories: (1) those with a usable built-in *PDM*, which always use the lowest energy (lower by a factor of at least *1000* when compared to the energy dissipated using the two other energy-saving methods), (2) those for which *DFS* leads to the lowest energy dissipation (when *PDM* is not available) and the second lowest energy dissipation (when *PDM* is available), and (3) those for which *DVS* leads to lower energy dissipation than *DFS*. This third category, however, includes only two *MCUs*, the AT89S8253 and the PIC16LF877. These two *MCUs* happen to also be the ones with the highest *static energy/dynamic energy ratio* and consequently, *DVS* leads to better results than *DFS* because it minimizes this large static energy component while *DFS* does not.

Transition Delay for switching between Minimum Energy Dissipation Mode and Full-Throughput Mode: Figure 5.1.3 shows the transition delays for the various *MCUs* when using the *PDM*, *DFS*, and the *DVS* techniques. The *MCUs* fall in one of two categories here as well: Those where *PDM* is always the fastest and those where *DFS* is the fastest. Nevertheless, *DFS* in only faster than *PDM* for those PIC *MCUs* since, as was listed in table 3.1, they have a transition delay of *1024 cycles* as opposed to *DFS* which on the average will have a maximum transition delay of *128 cycles*. Note, however, that *DVS* is never faster than *PDM* or *DFS* because of the very small dV_{CC}/dt (*1.95 V/msec*) of our cost constrained, but viable, variable voltage power supply.

5.2. Benchmark Simulations and Observations: We start by simulating the workloads of the *fixed-throughput model*, followed by these of the *burst-throughput model*. We also list in this section some observations particular to the developed benchmark workloads.

Simulation of the Fixed-Throughput Model Workloads: For space limitations, we only present the calculated *NETDP* metric, and the absolute energy dissipated, for workloads *WA11*, *WA13*, *WA41*, and *WA43*, in figures 5.2.1*a*, *b*, *c*, and *d*, and figures 5.2.2*a*, *b*, *c*, and *d*, respectively. We also list in table 5.2.1 the execution frequency and voltage of the different job subsets composing each of the twelve workloads when using the different energy-saving methods.

General Results

The *PDM* technique (for all *MCUs* equipped with one) renders the lowest *NETDP* for all workloads and all granularity levels, with few exceptions to be explained below. The processor's active time is minimal due to executing all jobs at full throughput (see table 5.2.1), saving the maximum amount of time to be used for energy saving, and since *PDM* has the lowest energy dissipation when in its energy-saving state (see figure 5.1.2), makes *PDM* the best option.

Figure 5.2.1*c* (i.e. *NEDP* for workload *WA41* with $U_W = 80\%$ and $N = 10$ *jobs*) shows an interesting exception where *PDM* is outperformed by *DVS*. As the utilization level increases, *while the granularity level is still too small for the transition delays to have an*

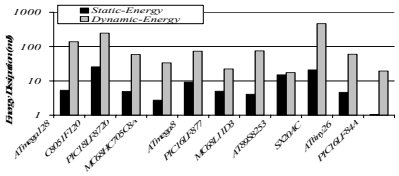


Figure 5.1.1. Dynamic and Static Energy Dissipation for Full-Throughput Operation (Maximum Operational Frequency and Voltage)

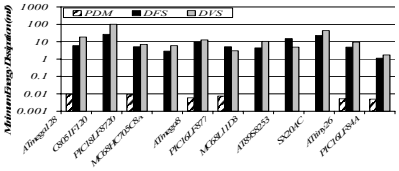


Figure 5.1.2. Minimum Energy Dissipation using the different Energy-Saving Methods

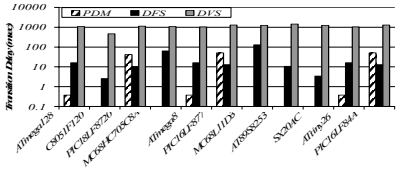


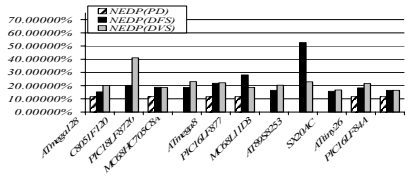
Figure 5.1.3. Transition Delays using the different Energy-Saving Methods

PDM	*All jobs will execute at full throughput (i.e. $V_{CC} = V_{max}$, $f_{CLK} = f_{max}$)*
DFS	*50% of the jobs will execute at $V_{CC} = V_{max}$, $f_{CLK} = f_{max}$* *25% of the jobs will execute at $V_{CC} = V_{max}$, $f_{CLK} = f_{max}/2$* *12.5% of the jobs will execute at $V_{CC} = V_{max}$, $f_{CLK} = f_{max}/4$* *12.5% of the jobs will execute at $V_{CC} = V_{max}$, $f_{CLK} = f_{max}/10$*
DVS	*25% of the jobs will execute at $V_{CC} = V_{max}$, $f_{CLK} = f_{max}$* *25% of the jobs will execute at $V_{CC} = V_{0.7fmax}$, $f_{CLK} = 0.7f_{max}$* *25% of the jobs will execute at $V_{CC} = V_{0.45fmax}$, $f_{CLK} = 0.45f_{max}$* *25% of the jobs will execute at $V_{CC} = V_{min}$, $f_{CLK} = f_{Vmin}$*

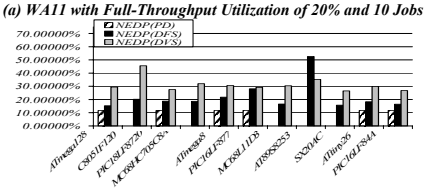
5.2.1. Percentage of Jobs of Subset I Executing at different Frequencies and Voltages for the different Energy-Saving Modes

impact on the overall performance, the amount of time spent by *PDM* executing its jobs at full-throughput increases, minimizing the only available source of time the *PDM* can use for energy-saving (i.e. the idle time). On the other hand, *DVS*, as well as *DFS* for that matter, are saving energy while jobs are being executed, as well as when the processor is idle. In the current case, *DVS* even exceeds *DFS* in energy savings as it runs almost

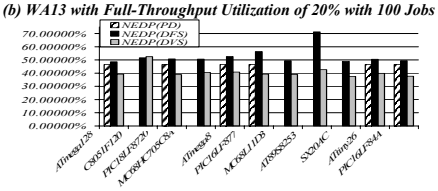
[‡] To the contrary to the utilizations listed in table 4.5.2.1, the utilization levels listed in table 4.5.2.2. are the actual system utilizations since each of subset II workloads is composed of tasks that actually require full-throughput to execute.



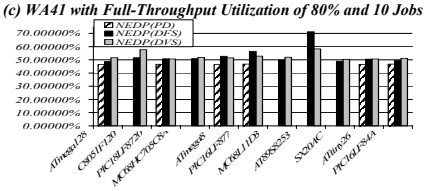
(a) WA11 with Full-Throughput Utilization of 20% and 10 Jobs



(b) WA13 with Full-Throughput Utilization of 20% with 100 Jobs

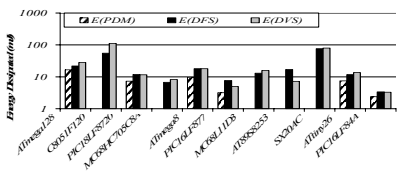


(c) WA41 with Full-Throughput Utilization of 80% and 10 Jobs

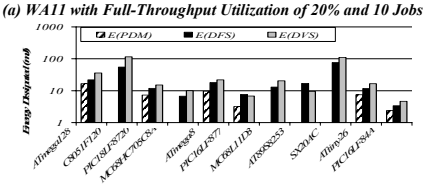


(d) WA43 with Full-Throughput Utilization of 80% and 100 Jobs

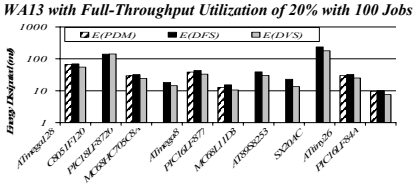
Figure 5.2.1. (a) NETDP metric for Workload WA11 of the Fixed Throughput Model
(b) NETDP metric for Workload WA13 of the Fixed Throughput Model
(c) NETDP metric for Workload WA41 of the Fixed Throughput Model
(d) NETDP metric for Workload WA43 of the Fixed Throughput Model



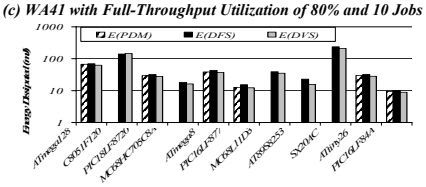
(a) WA11 with Full-Throughput Utilization of 20% and 10 Jobs



(b) WA13 with Full-Throughput Utilization of 20% with 100 Jobs

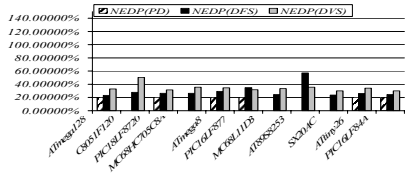


(c) WA41 with Full-Throughput Utilization of 80% and 10 Jobs

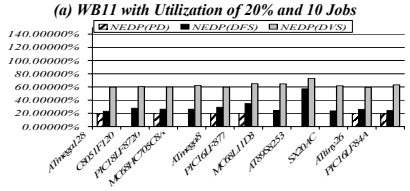


(d) WA43 with Full-Throughput Utilization of 80% and 100 Jobs

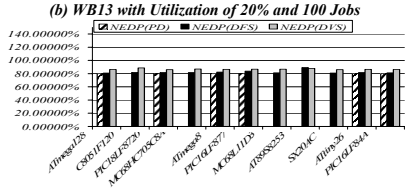
Figure 5.2.2. (a) Energy Dissipated for Workload WA11 of the Fixed Throughput Model
(b) Energy Dissipated for Workload WA13 of the Fixed Throughput Model
(c) Energy Dissipated for Workload WA41 of the Fixed Throughput Model
(d) Energy Dissipated for Workload WA43 of the Fixed Throughput Model



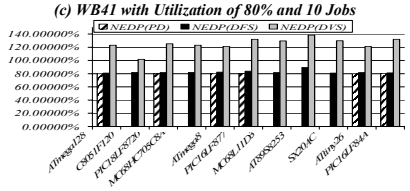
(a) WB11 with Utilization of 20% and 10 Jobs



(b) WB13 with Utilization of 20% and 100 Jobs

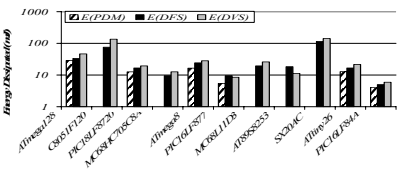


(c) WB41 with Utilization of 80% and 10 Jobs

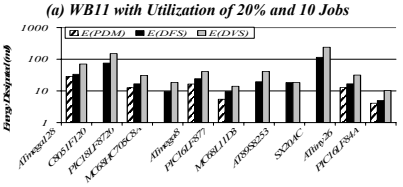


(d) WB43 with Utilization of 80% and 100 Jobs

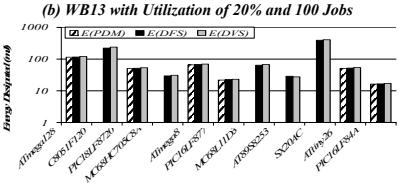
Figure 5.2.3. (a) NETDP metric for Workload WB11 of the Burst Throughput Model
(b) NETDP metric for Workload WB13 of the Burst Throughput Model
(c) NETDP metric for Workload WB41 of the Burst Throughput Model
(d) NETDP metric for Workload WB43 of the Burst Throughput Model



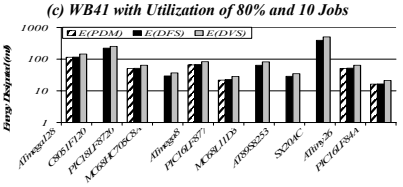
(a) WB11 with Utilization of 20% and 10 Jobs



(b) WB13 with Utilization of 20% and 100 Jobs



(c) WB41 with Utilization of 80% and 10 Jobs



(d) WB43 with Utilization of 80% and 100 Jobs

Figure 5.2.4. (a) Energy Dissipated for Workload WB11 of the Burst Throughput Model
(b) Energy Dissipated for Workload WB13 of the Burst Throughput Model
(c) Energy Dissipated for Workload WB41 of the Burst Throughput Model
(d) Energy Dissipated for Workload WB43 of the Burst Throughput Model

50% of its jobs at less than half of the maximum operational voltage with their corresponding frequency levels (see table 5.2.1), while *DFS* runs at a lower frequency level only.

DFS, and *DVS* save energy during job execution, as well as when the processor is idle, as opposed to *PDM* where energy is only saved when the processor is idle. For this reason, we can see from figures 5.2.2 that as the utilization level increases, *DVS* has the smallest increase in energy dissipation, followed by *DFS*, while *PDM* has a very high increase in energy dissipation with increasing utilization, again due to the way it saves energy (see table 5.2.1).

DFS and *PDM* are insensitive to the granularity level. This is a direct consequence of the fact that they have much smaller transition delays than *DVS* (see figure 5.1.3). This can be seen by comparing figure 5.2.1*a* and *b*, or figures 5.2.1*c*, and *d*.

Simulation of the Burst-Throughput Model Workloads: For space limitations, we only present the calculated *NETDP* metric, and the absolute energy dissipated, for workloads *WB11*, *WB13*, *WB41*, and *WB43*, in figures 5.2.3*a*, *b*, *c*, and *d*, and figures 5.2.4*a*, *b*, *c*, and *d*, respectively. With burst-throughput workloads, the MCU runs at full speed, and hence full voltage, when active. Thus, active times are minimized and idle mode characteristics dominate the overall energy use.

The *PDM* technique (for all microcontrollers equipped with one) renders the lowest *NEDP* for all workloads and all granularity levels. There are no exceptional cases because *PDM* has the lowest energy dissipation in the idle state.

When PDM is not available, DFS minimizes energy for nearly all cases. The exception is the AT89S8253, where the static energy consumption is better controlled with DVS. This only occurs at low granularity and low utilization (figure 5.2.4*a*).

DFS and *PDM* are still very insensitive to the granularity level. This can be seen by in figure 5.2.3*a* and *b*, or figures 5.2.3*c*, and *d*.

VI. GENERAL OBSERVATIONS AND CONCLUSIONS

We find that *DVS* is poorly suited for most short-bit width *COTS MCUs* mainly for the cost constrained design space of these *MCUs*. Existing *PDMs* are the most efficient option. When not available, though, they must be approximated by *DVS* and/or *DFS*.

DVS requires a wide voltage range to leverage its quadratic energy savings, but for all *MCUs* we studied this range was relatively small (at most $2\times$). This limitation also affects 32-bit microprocessors such as the IBM PowerPC405LP, TransMeta Crusoe TM5800 and Intel XScale 80200 [38].

DVS also requires a fast variable power supply (i.e. large *dVcc/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 $256\times$ energy 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.

Several improvements are possible for future *MCUs*. The first is to include usable power-down modes with fast transition times. The second is to provide hardware for a fast variable power supply on the *MCU* itself would make DVS feasible yet reduce the final cost for a system designer. The third is to include clock-division circuitry on the *MCU*; this is present in some newer microcontrollers.

REFERENCES

[1] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," 2000.

[2] P. L. D. Grunwald, K. Farkas, C. Morrey, and M. Neufeld., "Policies for dynamic clock scheduling," presented at OSDI, San Diego, CA, 2000.

[3] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," presented at First Symposium on Operating Systems Design and Implementation, Monterey, CA, 1994.

[4] R. Gonzalez and M. Horowitz, "Energy dissipation in general purpose microprocessors," *Solid-State Circuits, IEEE Journal of*, vol. 31, pp. 1277-1284, 1996.

[5] H. Hesin, S. Lee, J. B. Fryman, A. U. Diril, and Y. S. Dhillon, "The Elusive Metric for Low-Power Architecture Research," presented at Workshop on Complexity-Effective Design, San Diego, CA, 2003.

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

[7] T. D. Burd and R. W. Brodersen, "Energy efficient CMOS microprocessor design," 1995.

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

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

[10] Q. Gang, "What is the limit of energy saving by dynamic voltage scaling?" 2001.

[11] A. Maier, P. Fugger, and B. Rinner, "Low-Power meets High-Performance – Dynamic Voltage Scaling on the CARMEL Signal Processor," presented at International Conference and Workshop: Telecommunications and Mobile Computing, Graz, Austria, 2001.

[12] A. Miyoshi, C. Lefurgy, E. V. Hensbergen, R. Rajamony, and R. Rajkumar, "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.

[13] 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.

[14] C. *Vaucourt* and M. *Matzberger*, "Small Dynamic Voltage Management Solution Based on TPS62300 High-Frequency Buck Converter and DAC6571," in *Application Report SLVA196*. Dallas, Texas: Texas Instruments, 2004.

[15] A. Sinha and A. P. Chandrakasan, "Energy efficient real-time scheduling [microprocessors]," presented at IEEE/ACM International Conference on Computer Aided Design, 2001.

[16] Y. Z. a. F. Mueller, "Feedback Dynamic Voltage Scaling DVS-EDF Scheduling: Correctness and PID-Feedback," presented at Workshop on Compilers and Operating Systems for Low Power, 2002.

[17] J. L. a. A. Smith, "Improving Dynamic Scaling Algorithms with PACE," presented at ACM Sigmetrics, 2001.

[18] "ATmega8 Data Sheet." San Jose, CA: Atmel Corporation, 2004.

[19] "ATmega128." San Jose, CA: Atmel Corporation, 2003.

[20] "ATtiny Data Sheet." San Jose, CA: Atmel Corporation, 2003.

[21] "C8051F12x Data Sheet." Austin, TX: Cygnal Integrated Products, 2004.

[22] "MC68HC705C8A Data Sheet," Motorola Inc., 2001.

[23] "MC68L11D3 Data Sheet," Motorola Inc., 2002.

[24] M. M. Islam and K. Murase, "A new algorithm to design compact two-hidden-layer artificial neural networks," *Neural Networks*, vol. 14, pp. 1265-1278, 2001.

[25] "PIC18F6520/8520/6620/8620/6720/8720 Data Sheet." Chandler, AZ: MicroChip Inc., 2004.

[26] "SX2028AC Data Sheet." Mountain View, CA: Ubicom Inc., 2002.

[27] E. Kreyszig, *Introductory functional analysis with applications*. New York: Wiley, 1978.

[28] W. Rudin, *Real and complex analysis*, 3rd ed. ed. New York: McGraw-Hill, 1987.

[29] B. S. Thomson, J. B. Bruckner, and A. M. Bruckner, *Elementary real analysis*. Upper Saddle River, N.J.: Prentice-Hall, 2001.

[30] E. K. P. Chong and S. H. çZak, *An introduction to optimization*, 2nd ed. New York: Wiley, 2002.

[31] E. Polak, *Optimization: algorithms and consistent approximations*. New York: Springer-Verlag, 1997.

[32] I. Miller, J. E. Freund, and R. A. Johnson, *Miller & Freund's probability and statistics for engineers*, 6th ed. Upper Saddle River, NJ: Prentice Hall, 2000.

[33] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers*, 2nd ed. New York: Wiley, 1999.

[34] R. V. Hogg and E. A. Tanis, *Probability and statistical inference*, 6th ed. Upper Saddle River, NJ: Prentice Hall, 2001.

[35] H.-H. S. Lee, J. B. Fryman, A. U. Diril, and Y. S. Dhillon, "The Elusive Metric for Low-Power Architecture Research," presented at Workshop on Complexity-Effective Design, San Diego, CA, 2003.

[36] W. Fornaciari, P. Gubian, D. Sciuto, and C. Silvano, "Power estimation of embedded systems: a hardware/software codesign approach," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 6, pp. 266-275, 1998.

[37] T. D. Burd and R. W. Brodersen, "Energy efficient CMOS microprocessor design," presented at Hawaii International Conference on System Sciences, Wailea, HI, 1995.

[38] 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.