Architectural Power Management for Battery Lifetime Optimization in Portable Systems

Manish Kulkarni[1] and Vishwani D. Agrawal[2]

Abstract

For portable computing devices, maximizing battery lifetime or performing maximum possible operations per recharge is a primary objective. So for such devices, power consumption is an important design constraint. Power management techniques for portable computing devices are not new. Various voltage and frequency scaling techniques have been proposed and are being used in commercial devices. This work considers the role of energy source, i.e., battery, in the optimization of a portable system. We introduce battery lifetime in number of clock cycles as an optimization metric. In addition to total energy consumed from the battery, battery lifetime also depends on the battery efficiency, which may degrade as the power consumption increases. Power reduction techniques for a processor such as frequency scaling, i.e., clock slowdown (CSD) and instruction slowdown (ISD) are studied. These techniques are examined from a battery efficiency viewpoint and are evaluated in terms of battery lifetime expressed in number of operational cycles. In case of ISD, implemented with the help of NOPs, we demonstrate that the lifetime, for a given size of the battery, might be optimum at a slowdown factor around 2 to 3. The battery lifetime improvement requires reduced power dissipation coupled with prolonged execution time and, surprisingly, sometimes even increased energy consumption. With architecture level modifications, such as instruction slowdown, battery lifetime increase of up to 20% is realized.

1. Introduction

Computing platforms have become more and more portable over the period of time. Moreover, contemporary mobile computing devices are capable of performing as many computations per second as that of desktop computers few years back. Fortunately, due to technology scaling and power efficient approach in design, these mobile devices do not consume as much power as that of desktop computers. But with increasing usage of mobile devices in everyday life and the variety of applications being performed by these devices, the demand for power from power sources is continuously growing. Using a battery of higher capacity is not a solution, since for portable devices, the size and weight of battery, which are proportional to the battery capacity, have stringent design limits. Also, the growth in battery specific weight, i.e. stored energy per unit weight, has not followed the power requirement trends in the devices. These specific weights for lithium-ion batteries have barely doubled over last ten years [37]. These factors motivate us to consider alternate solutions towards power consumption problems in portable systems. Making efficient use of available energy in the battery is one alternative. So optimization of battery lifetime, i.e. performing maximum number of operations per recharge of battery, becomes

1: Auburn University, email: mmk0002@auburn.edu
2: Auburn University, email: vagrawal@auburn.edu

Proceedings of 15th VLSI Design & Test Symposium, Pune, India, July 2011.

one of the primary objectives for portable computing system design. This paper discusses few design techniques and proposes an architectural power management method to optimize battery lifetime and to obtain maximum number of cycles per recharge.

2. BACKGROUND

2.1. Power Management Methods for Battery Optimization

Power management methods targeted towards battery optimization have previously been reported in the literature. We classify them in the following categories.

1) Voltage Management Methods: Most common of voltage management methods is dynamic voltage scaling (DVS). Here the circuit can statically or dynamically vary VDD depending on the throughput requirement. A relevant problem is to find an optimum value of supply voltage which would minimize the energy consumption of the battery and still maintain the required throughput. Reference [3] proposes a method to find optimum operating voltage for minimization of battery discharge-delay product. Multi-voltage domain designs can also be classified under this category.

2) Throughput Management Methods: Dynamic frequency scaling is one of the most used methods in this category. CPU frequency scaling for battery powered computers is examined in [5] in terms of its impact on battery life, system performance, and power consumption. Frequency scaling approaches use information from a battery model to vary the clock frequency of system components dynamically at run time. They also use workload characteristics such as run-time and idle-time percentages dynamically, and models of system power and performance. These approaches can be used to ensure efficient use of the battery without significantly compromising system performance [7].

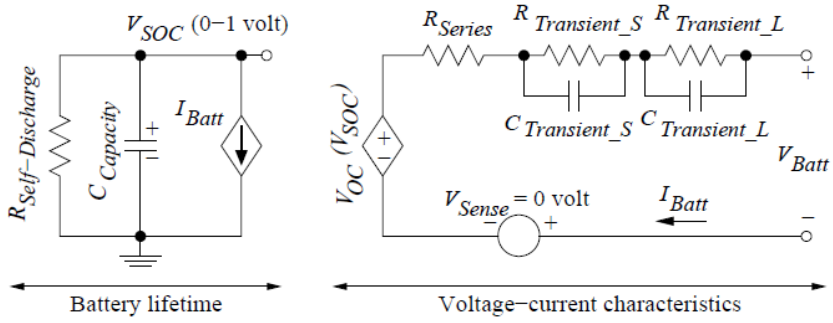

Fig. 1. An Electrical Model for Lithium-ion battery.

3) Functional Management Methods: In most cases, a given problem can have an alternate design or implementation method that consumes less power. Some of the low power design techniques such as low power FSM design, bus encoding, gate reordering and battery aware dynamic task scheduling [6] represent few such cases. These methods focus on power management of the system in order to reduce the average current drawn from the battery. A recently proposed functional management method, named instruction slowdown [10],

exploits idleness in a pipeline processor to dynamically manage power to different units. Dynamic voltage and frequency scaling (DVFS) is a method that combines voltage and throughput management whereas architecture level parallelism with multicore designs is a combination of all the three types of methods mentioned above.

2.2. Choice of Metric

Traditional metrics like minimization of power and energy are not quite relevant when power source (battery) optimization is a concern. The energy stored in a battery is assumed to be constant and available at any possible rate. In reality, however, the energy stored in a battery may not be used to its full extent. The delivery of energy from battery to system depends on battery efficiency which, in turn, depends on the average value of the current drawn from the battery.

Weiser et al. [8] present Millions of Instructions Per Joule (MIPJ) as a quality metric for dynamic voltage scaling (DVS). The key idea is to eliminate idle time by reducing the processor voltage and clock for a given segment of computation. To predict processor utilization, either a fixed-size window of events in future or a fixed-size window of events in the past is analyzed, and the corresponding DVS decisions are evaluated using trace-based simulations. This method has limited practicality since measurement and tracking of battery energy in terms of joules is difficult.

Rakhmatov et al. [9], [6] use an analytical model of the battery to minimize a cost function $\sigma(t)$. This cost is function of load current $i(t)$ and sum of $l(t)$ and $u(t)$, where, $l(t)$ is the charge lost in load and $u(t)$ is the charge unavailable. Evaluation of this cost function is in the context of DVS for task scheduling and battery optimization. Minimization of this cost function is subjected to constraints such as task dependencies, task deadlines etc.

Pedram et al. [3] propose battery discharge-delay product as the metric. This metric is similar to the energy-delay product while accounting for the battery characteristics and the DC/DC conversion efficiency. The BD-delay product states that the design goal should be to minimize delay and maximize battery lifetime at the same time.

We consider clock cycles per recharge as a metric for evaluation of optimization methods. This metric is particularly suitable for the functional management method proposed here since with insertion of SLOPs system now takes longer time to finish the task. A relevant measure of lifetime, therefore, is the lifetime in number of clock cycles. Thus, instead of expressing the lifetime in raw seconds, we express it in terms of computational work units.

2.3. Battery Model Used for This Work

We use an electrical model provided by [4]. This model is shown in Figure 1. One of the reasons behind choosing this model is its capability of predicting lifetime and I-V performance. Besides load current, it considers effects of temperature, number of cycles and storage time dependence of capacity on battery lifetime. This model is also scalable as it models batteries of varying AHr ratings and predicts runtime for different load current profiles. This model can be used for Lithium-ion, polymer Lithium-ion and NiMH batteries.

1) Description: On the left side of figure 1, a capacitor $C_{Capacity}$ represents the present state of charge (SOC) of the battery and a current source I_{Batt} models the

discharge. The right side of the circuit models the voltage and current characteristics of the battery based on the current drawn from the battery. These two parts are connected to each other by a voltage controlled voltage source VSOC whose value depends on the open circuit voltage (VOC) of the capacitor $C_{Capacity.}$ Assuming a battery is discharged from an equally charged state to the same end-of-discharge voltage, the extracted energy, called usable capacity, declines as cycle number, discharge current, and/or storage time (self-discharge) increases, and/or as temperature decreases [4]. The usable capacity can be modeled by a full-capacity capacitor ($C_{Capacity}$), a self-discharge resistor ($R_{Self-Discharge}$), and an equivalent series resistor (the sum of R_{Series} ,$R_{Transient_S}$, and $R_{Transient_L}$). The full- capacity capacitor $C_{Capacity}$ represents the whole charge stored in the battery, i.e., SOC, by converting nominal battery capacity in Ahr to charge in coulomb and its value is defined as,

$$C_{Capacity} = 3600 \times \text{Capacity} \times f1\,(\text{Cycles}) \times f2\,(\text{Temp}) \quad (1)$$

where *Capacity* is the nominal capacity in AHr, *f1 (Cycle)* is a correction factor for number of cycles, *f2 (Temp)* is a temperature- dependent correction factor.

2) Battery Lifetime: The state of charge (SOC) is defined as 1.0 for a fully charged battery. It is represented by a voltage V_{SOC}, which ranges between 0 and 1 volt. The charge of the battery is stored in a capacitor $C_{Capacity}$ whose value is determined as indicated by Equation 1. Thus, 1 AHr × 3600 seconds is the total amount of charge in coulombs. As the battery goes through cycles of charging and discharging its capacity to hold charge is affected, reducing the usable capacity. That is represented by *f1 (Cycles)*. Similarly, temperature affects the usable capacity and that is represented by *f2 (Temp)*. For simplicity, we have assumed both factors to be unity in the present discussion. The resistance $R_{Self-Discharge}$ represents leakage when the battery is stored over a long period. For reasonable time between recharge, this can be considered to be large or practically infinite. The current source I_{Batt} represents a source when the battery is being charged or a load when the battery is powering a circuit. In the latter case, it is the current being supplied to the DC-to-DC converter and to the circuit after conversion. When the model is used to simulate the behavior of a battery that is fully charged, V_{SOC} is initialized to 1 volt.

3) Voltage and Current Characteristics: The circuit on the right in Figure 1 emulates the terminal voltage of the battery as it supplies current. This part is linked to the part on the left by state of charge (SOC), a quantity in the (0.0, 1.0) range. V_{OC}(SOC) is the open circuit voltage. For Lithium-ion batteries, Chen and Rincon-Mora [4] empirically derive expressions for the circuit components, which all depend on SOC.

$$V_{OC}(SOC) = -1.031e^{-35\times SOC} + 3.685 + 0.2156 \times SOC - 0.1178 \times SOC^2 + 0.3201 \times SOC^3 \quad (2)$$

$$R_{Series}(SOC) = 0.1562e^{-24.37\times SOC} + 0.07446 \quad (3)$$

$$R_{Transient_S}(SOC) = 0.3208e^{-29.14\times SOC} + 0.04669 \quad (4)$$

$$C_{Transient_S}(SOC) = -752.9e^{-13.51\times SOC} + 703.6 \quad (5)$$

$$R_{Transient_L}(SOC) = 6.6038e^{-155.2\times SOC} + 0.04984 \quad (6)$$

$$C_{Transient_L}(SOC) = -6056e^{-27.12\times SOC} + 4475 \quad (7)$$

Application of this model to observe effects of DVFS in a typical battery powered system has been demonstrated by [11], [38].

3. INSTRUCTION SLOWDOWN

Consider a processor built in certain semiconductor technology. If we reduce the supply voltage V, the critical path delay will increase and hence the maximum clock frequency f will have to be decreased. This will reduce the dynamic power in proportion to V^2f. Static power will also decrease as V^2. However, a measure of energy a computing task will use is the total energy per cycle (EPC), consisting of dynamic EPC and static EPC. Dynamic EPC is proportional to V^2 and static EPC is proportional to V^2/f. We notice that dynamic EPC always reduces with voltage scale down. However, static EPC is proportional to 1/f, which will increase rapidly as V approaches close to the threshold voltage.

Thus, for a given technology (i.e., given threshold voltage), there is an optimum supply voltage and a corresponding clock frequency that minimize the total EPC. Any further power reduction by voltage scaling beyond this optimum value will incur an increase in the total EPC, although power will reduce. As the supply voltage gets closer to the threshold voltage, the performance also becomes sensitive to process variation that is common in nanoscale technologies. In practice, therefore, the supply voltage has a lower bound. If further power reduction is required, say, due to battery characteristics, thermal factors or other operational considerations, then clock frequency alone would have to be reduced. This will reduce power but increase energy per cycle (EPC). Dynamic voltage control within a clock period [16] can reduce the EPC but as pointed out earlier, requires complex control circuitry.

We assume a situation where voltage is at its lowest permissible limit and power must be reduced. Traditionally, we would slowdown the clock and let EPC increase. This will be a performance-power tradeoff that involves an essential energy penalty. We explore an alternative solution in which clock is not slowed down but performance is traded off, similar to clock slowdown, for power reduction while energy penalty is reduced, especially for high leakage technologies.

In this new methodology [19], the operation of a processor is slowed down for power reduction by inserting non-functional cycles while the rated clock frequency (f) is maintained. This is similar to inserting instruction we call SLOP (slowdown for low power). Although it is described as a purely hardware induced operation, SLOP can be included in the software instruction set. Let us define:

$$n = \text{processor slowdown factor} \quad (8)$$

$$f = \text{rated clock frequency in Hz} \quad (9)$$

$$Pd = \text{dynamic power with rated clock} \quad (10)$$

$$Ps = \text{static power with rated clock} \quad (11)$$

k = Ps /Pd = static power ratio (12)

T = time duration of a computing task (13)

In a typical implementation, a power management unit (PMU) monitors the system and, if necessary, determines an appropriate slowdown factor (n), which is supplied to the control. The control then inserts the required number of SLOPs in the pipeline. The factor n is assumed to be an integer here but, in general, can be any number that determines the percentage of SLOPs to inserted in the instruction stream.

Hardware execution of SLOP resembles a conventional NOP, stall or bubble [15] with a few differences. First, its execution in a pipeline requires no "fetch" because the control generates it locally. Second, the control generates low power mode signals for various hardware units. To analyze the power and energy relations, we will use the same symbol definitions as in the previous section. We also define a SLOP power factor:

$$\beta = \frac{power\ consumed\ by\ SLOP}{av.\ power\ consumed\ by\ non\ NOP\ instr.} \tag{14}$$

where $0 \leq \beta \leq 1$. For a slowdown factor n, we insert n − 1 SLOPs after each instruction. Consider a period of 1 second, containing f clock cycles. The energy consumed during a regular instruction (assumed to be non-NOP) cycle is $P_d(1 + k)/f$ and that during a SLOP cycle is $\beta P_d(1 + k)/f$. Of those f cycles, f /n are regular instruction cycles and (n − 1)f /n are SLOP cycles. Thus, total power consumption, or energy dissipated per second, is obtained as,

$$\begin{aligned} P_{ISD}(n) &= \frac{P_d(1+k)}{f} \times \frac{f}{n} + \frac{\beta P_d(1+k)}{f} \times \frac{(n-1)f}{n} \\ &= P_d(1+k)\frac{\beta n - \beta + 1}{n} \end{aligned} \tag{15}$$

Now a computing task of original duration T will require nT time for its duration.The energy dissipation is obtained as,

$$\begin{aligned} E_{ISD}(n, nT) &= P_{ISD}(n) \times nT \\ &= P_d(1+k)(\beta n - \beta + 1)T \end{aligned} \tag{16}$$

We find the power and energy ratios as follows:

$$P_{ISDratio} = \frac{P_{ISD}(n)}{P_{ISD}(1)} = \frac{\beta n - \beta + 1}{n} \tag{17}$$

$$E_{ISDratio} = \frac{E_{ISD}(n, nT)}{E_{ISD}(1, T)} = \beta n - \beta + 1 \tag{18}$$

These power and energy ratios as functions of slowdown factor n are shown in Figure 2. A ratios below 1 indicate both power reduction and energy reduction. Notice that power (solid line) is always reduced. More reduction is achieved for higher leakage ($\beta = 0.1$) technology. Energy ratio increases indicating that higher energy is being consumed as slowdown factor increases. However, since the power is reducing, the current drawn from the battery is reducing and hence the battery efficiency is increasing. This actually enables batteries to provide more energy and hence the increase in the energy balances out. In fact, for certain slowdown factors the surplus of available energy (due to higher battery efficiency) exceeds the excess energy spent and hence we gain in terms of battery lifetime.

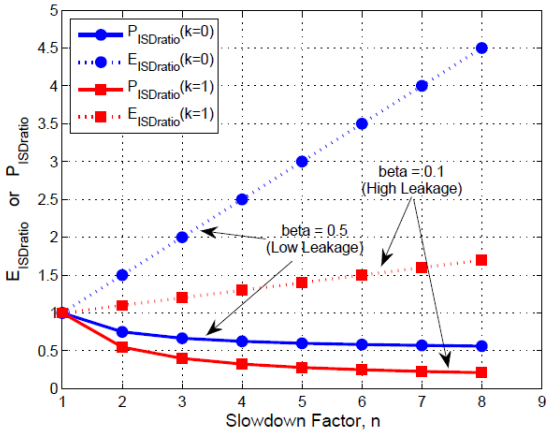

Fig. 2. Instruction slowdown (ISD) power and battery lifetime ratios.

TABLE I
HSPICE SIMULATION (32NM CMOS, 90°C).

Hardware block	Energy/cycle		SLOP power		
	Dyn. nJ	Stat. nJ	Power mode	Dyn. %	Stat. %
PC	85114	17742	CG	25	100
PC+1 adder	28947	6536	PG	0	0
IM	6780	3209	Drowsy	25	25
Regfile	98262	192375	CG	30	100
Forwarding	31297	4090	PG	0	0
Hazard	25421	3744	PG	0	0
Controller	14338	2973	None	100	100
32-b ALU	263815	22346	PG	0	0
32-b comp	39710	5695	PG	0	0
DM	64343	50699	Drowsy	25	25
3-1 mux	392374	56299	PG	0	0
2-1 mux	204456	44106	PG	0	0
BrnchAddrCal	181878	13680	PG	0	0
IF/ID reg	156027	32048	CG	50	100
ID/EX reg	213447	58412	CG	50	100
EX/DM reg	131033	34324	CG	50	100
DM/WB reg	127885	33481	CG	50	100
ForwDM/WB	5820	1009	PG	0	0

4. SLOP IN HARDWARE

We used a 32-bit MIPS pipelined processor for evaluation of the ISD and CSD methods. It has a conventional five-stage pipeline containing the fetch (IF), decode (ID), execute (EX), memory (DM) and write-back (WB) stages [15]. It also contains hazard and forwarding units. We obtained an available VHDL model [17] and synthesized using Mentor Graphics Leonardo Spectrum. This provided us a gate-level model for power analysis. Various blocks of the processor were extracted as transistor-level netlists using Mentor Graphics Design Architect. Each block was simulated in HSPICE for 1,000 random input vectors with 10ns clock rate (f = 100MHz) to determine the average per cycle dynamic and static energy dissipation. This evaluation was repeated for five CMOS technologies, 180nm, 90nm, 65nm, 45nm and 32 nm, using the

predictive technology models (PTM) [1], [14], [12]. The simulation assumed 90ºC temperature.

A sample result for 32nm is shown in Table I. The last three columns of this table are discussed later. Communication buses are not considered separately because all drivers and buffers are included as parts of various hardware blocks. We wrote a MIPS program that multiplies hexadecimal integers FFFF and 0004 by repeated additions. Our processor has separately addressable instruction (IM) and data (DM) memories. Initially, DM(2) = FFFF, DM(3) = 4, DM(4) = 1. Final result is DM(5) = 0003FFFC. The MIPS code is given in Figure 3.

This program completes in 34 cycles. The number of times pipeline stages are activated are: 34 IF, 29 ID, 18 EX, 4 DM and 14 WB. The execution statistics of hardware stages and the instruction mix as well as the number of cycles can be easily changed by varying the parameters in the program. It was assembled by hand and the gate-level model was simulated using Mentor Graphics ModelSim. The final result was verified. For power, active blocks in a pipeline stage were identified. Total energy of the pipeline stage was computed by adding the dynamic and static energies of its active blocks. After characterizing each pipeline stage for its energy, the total energy of the program was computed by adding energies of pipeline stages as per the numbers obtained above. The dynamic energy was added up for active stages while the static energy was added up for all blocks for 34 cycles, using the technology-specific data (e.g., Table I for 32nm). The ratio of total static energy to dynamic energy for each technology gives the respective value of the leakage factor k shown in Table II.

```
0000    LW      $1, X:0002($0)
0001    ADD     $4, $1, $0
0002    ADD     $1, $0, $0
0003    LW      $3, X:0004($0)
0004    LW      $2, X:0003($0)
0005    BEQ     $2, $0, X:0003
0006    SUB     $2, $2, $3
0007    ADD     $1, $1, $4
0008    J       X:0000005
0009    SW      $1, X:0004($3)
000A    #J      X:000000A(HALT)
```

Fig. 3. MIPS program used for power estimation.

TABLE II
LEAKAGE FACTOR (k) AND SLOP POWER FACTOR (β).

Technology	Leakage factor k	SLOP power factor β
180nm	0.097	0.265081
90nm	0.124	0.23699
65nm	0.268	0.212003
45nm	0.353	0.183881
32nm	0.413	0.159012

Table I quantitatively shows how power was reduced by clock gating (CG), power gating (PG) and drowsy memories. Power gating (PG) focuses on leakage. Circuit level approaches for leakage reduction include body bias control [20], dual threshold domino logic [21], [28], input vector control [26] and power gating [22], [29], [30]. We adopt power gating for combinational blocks. It is assumed that the supply line will be gated by pull-up or a pull-down devices that will be put in the cutoff mode during SLOP cycles. This will almost

completely eliminate both static and dynamic power during those cycles [25]. We must, however, realize that power gating at clock cycle level represents a design challenge. Studies [20], [33] show that improvements will be needed both in the speed and energy cost of power control and implemented in the present-day design.

Drowsy mode for caches: The dynamic and leakage power consumed by instruction and data caches is a sizable portion of total power consumed by the processor. During SLOP cycles, the memory cells are put into low voltage "drowsy mode", which can allow up to 75% of energy reduction with no more than 1% of performance overhead [35]. In addition, decoder and sense amplifier can be power gated. Another technique identifies an application's cache requirements dynamically, and uses a circuit-level mechanism, *"gated-Vdd"*, to gate the supply voltage to the SRAM cells of the cache's unused sections to reduce leakage [30].

Clock gating (CG) is applied to registers. Their power is not gated because the state must be preserved. A significant fraction of the dynamic power in a processors is consumed by the clock network and flip-flops. The clock buffers can consume 50% or more of total dynamic power [29], [34]. Clock gating turns off the clocks when they are not required or stop them from feeding to the components which are not being used. Results show that up to 43% power saving can be achieved with a possible 20% reduction in area when clock gating replaces the state-retention feedback logic of flip-flops [18]. The clock gating employed in the register file with high switching activity of about 0.25 shows that power saving of about 70% can be achieved [36]. At the time of this writing, we have not completed an evaluation of these techniques. The data in the last two columns of Table I is based on the references cited here. To compute the SLOP power factor (β) we first weight columns 2 and 3 by columns 5 and 6, respectively. The dynamic and static power of a SLOP cycle is then calculated in a similar way as described before for a regular instruction. The ratio of the power of SLOP cycle to that of the regular instruction cycle is β given in Table II.

5. RE SULT S

We simulated a representative case of ISD for 32 nm and found load currents for different slowdown factors. These currents were applied to a 800mAHr battery model [4] mentioned in subsection 2.3. For such a model, we define Ideal lifetime as,

$$Ideal\ Lifetime = \frac{AHr\ rating}{Load\ Current\ in\ Amperes} \tag{19}$$

A graph of power ratios, energy ratios and ideal battery lifetime ratios (Lratio) against slow down factor, n, is plotted and is as shown in Figure 4. From this graph, it is clear that with increasing slow down factor, power reduces, energy increases and ideal battery lifetime also reduces due to increase in energy. Ideal battery, however, does not consider the increase in efficiency of the battery due to reduced power (and hence the current drawn from the battery). When the ideal battery was replaced with a practical battery as represented by the model mentioned in section 2.3, we see different results as shown in Figure 5.

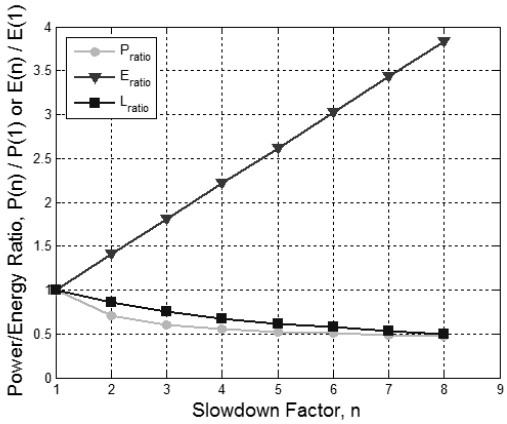

Fig. 4. Power ratio, energy ratio and ideal battery lifetime ratio plotted against slow down factor,n, for ISD in 32nm

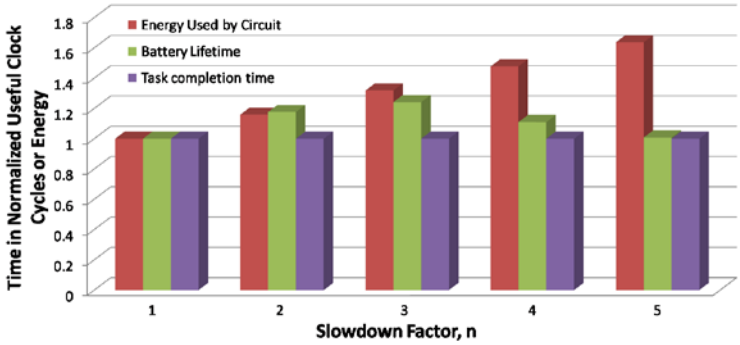

Fig. 5. Circuit energy, battery lifetime and task completion time plotted against slowdown factor, for ISD in 32nm and 800mAHr battery

As we can observe in Figure 5, the lifetime saving achieved using ISD exceeds the task completion time for slowdown factors of 2, 3 and 4 with peak saving of over 20% at slowdown factor of 3. This indicates that for these cases, we gain in terms of battery lifetime with slow down.

5. CONCLUSION

The proposed method of instruction slowdown (ISD) has advantages in power saving for high leakage technologies. We suggest combining the slowdown methods with overall supply voltage scaling. Voltage reduction will save dynamic and static power as well as energy. ISD will additionally eliminate hazards progressively as n increases. SLOP is presented purely as an internal mechanism supported by power management and control hardware. Its inclusion in the instruction set will allow compilers to explore creative ways to use the power management hardware.

REFERENCES

[1] http://www.eas.asu.edu/ptm.

[2] I. Buchmann, “Batteries in a Portable World: A Handbook on Rechargeable Batteries for Non-Engineers”, Richmond, British Columbia: Cedex Electronics, Inc., second edition, 2001.
[3] M. Pedram and Q. Wu, “Design Considerations for Battery-Powered Electronics”, Proc. 36th ACM/IEEE Design Automation Conference, 1999, pp. 861-866.
[4] M. Chen and G. A. Rincon-Mora, “Accurate Electrical Battery Model Capable of Predicting Runtime and I-V Performance”, IEEE Transactions on Energy Conversion, vol. 21, no. 2, pp. 504-511, June 2006.
[5] T. L. Martin, “Balancing Batteries, Power and Performance: System Issues in CPU Speed-Setting for Mobile Computing”, PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1999.
[6] D. Rakhmatov, S. Vrudhula, and C. Chakrabarti,“Battery-Conscious Task Sequencing for Portable Devices Including Voltage/Clock Scaling”, Proc. 39th Design Automation Conf., 2002, pp. 189-194.
[7] K. Lahiri, S. Dey, D. Panigrahi, and A. Raghunathan, “Battery-Driven System Design: A New Frontier in Low Power Design”, Proc. Asia South Pacific Design Automation Conf. and Int. Conf. on VLSI Design, Jan. 2002, p. 261.
[8] M. Weiser, B. Welch, A. Demers, and S. Shenker, “Scheduling for Reduced CPU Energy”, Proc. OS Design and Implementation, 1994.
[9] D. N. Rakhmatov and S. B. K. Vrudhula, “An Analytical High-Level Battery Model for Use in Energy Management of Portable Electronic Systems”, Proc. IEEE/ACM Intl Conf. Computer-Aided Design, 2001, pp. 488-493.
[10] M. Kulkarni, “Energy Source Lifetime Optimization for a Digital System through Power Management”, MS Thesis, Auburn University, Dec. 2010.
[11] M. Kulkarni and V. D. Agrawal, “Matching Power Source to Electronic System: A Tutorial on Battery Simulation”, Proc. VLSI Design and Test Symposium, July 2010.
[12] W. Zhao and Y. Cao, “New Generation of Predictive Technology Model for Sub-45nm Early Design Exploration”, IEEE Transactions on Electron Devices, vol. 53, pp.2816-2823, Nov. 2006.
[13] R. Rao, S. Vrudhula, and D. N. Rakhmatov, “Battery Modeling for Energy-Aware System Design”, Computer, vol. 36, no. 12, pp. 77-87, Dec. 2003.
[14] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, “New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design”, in Proc. Custom Integrated Circuits Conference, 2000, pp.201-204.
[15] D. A. Patterson and J. L. Hennessy, “Computer Organization and Design: The Hardware/Software Interface”, Fourth Edition. Morgan Kaufmann, 2009.
[16] B. Yu and M. L. Bushnell, “A Novel Dynamic Power Cutoff Technique (DPCT) for Active Leakage Reduction in Deep Submicron CMOS Circuits”,
Proc. International Symp. Low Power Electronics and Design, pp.214-219, 2006.
[17] A. Arthurs and L. Ngo, “Analysis of the MIPS 32-Bit, Pipelined Processor Using Synthesized VHDL,” Technical report, University of Arkansas, Department of Computer Science and Engineering. www.csce.uark.edu/a~jarthu/papers/mips vhdl.pdf.
[18] K. C. Pokhrel, “Physical and Silicon Measures of Low Power Clock Gating Success: An Apple to Apple Case Study”, Synopsys Users Group (SNUG), 2007.
[19] K. Sheth, “A Hardware-Software Processor Architecture using Pipeline Stalls for Leakage Power Management”, Master’s Thesis, Auburn University,
December 2008.
[20] D. Duarte, Y. F. Tsai, N. Vijaykrishnan, and M. J. Irwin, “Evaluating Run-Time Techniques for Leakage Power Reduction”, in Proc. 15th International
Conf. VLSI Design, 2002.
[21] S. Dropsho, V. Kursun, D. H. Albonesi, S. Dwarkadas, and E. G. Friedman, “Managing Static Leakage Energy in Microprocessor Functional Units”, in Proc. 35th Annual International Symp. Microarchitecture, MICRO, 2002, pp. 321-332.

[22] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural Techniques for Power Gating of Execution Units", in Proc. International Symp. Low Power Electronics and Design, 2004, pp. 32-37.
[23] L. L. Hurd, "Power Reduction for Multiple-Instruction-Word Processors with Proxy NOP Instructions", U.S. Patent 6535984, March 18, 2003.
[24] L. L. Hurd, "Power Saving by Disabling Memory Block Access for Aligned NOP Slots During Fetch of Multiple Instruction Words" U.S. Patent
6442701, August 27, 2002.
[25] J. Frenkil and S. Venkatraman, "Power Gating Design Automation", in D. Chinnery and K. Keutzer, "Closing the Power Gap Between ASIC and Custom Tools and Techniques for Low-Power Design", chapter 10, pp.251-280, Springer, 2007.
[26] M. C. Johnson, D. Somasekhar, L.-Y. Chiou, and K. Roy, "Leakage Control with Efficient Use of Transistor Stacks in Single Threshold CMOS", IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol. 10, no. 1, pp.1-5, Feb. 2002.
[27] "Mobile Intel Pentium 4 Processor with 533 MHz Front Side Bus", Intel Incorporation, January 2004.
[28] J. T. Kao and A. P. Chandrakasan, "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits", IEEE Journal of Solid-State Circuits, vol. 35, no. 7, pp. 1009-1018, July 2000.
[29] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, "Low Power Methodology Manual for System on Chip Design", Boston: Springer, 2008.
[30] M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache
Memories", in Proc. International Symp. Low Power Electronics and Design, 2000, pp. 90-95.
[31] V. Tiwari, P. Ashar, S. Malik, "Technology Mapping for Low Power", 30th Design Automation Conference, 1993, pp. 74-79
[32] R. F. Service, "New Supercapacitor Promises to Pack More Electrical Punch", Science, vol. 313, p.902, 18 Aug. 2006.
[33] J. W. Tschanz, S. G. Narendra, Y. Ye, B. A. Bloechel, S. Borkar, and V. De, "Dynamic Sleep Transistor and Body Bias for Active Leakage Power Control of Microprocessors", IEEE Jour. Solid-State Circuits, vol. 38, no. 11, pp. 1838-1845, Nov. 2003.
[34] K.-S. Yeo and K. Roy, "Low-Voltage, Low-Power VLSI Subsystems", McGraw-Hill, 2005.
[35] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy Caches: Simple Techniques for Reducing Leakage Power", in Proc. International Symposium on Computer Architecture, 2002, pp.148-157.
[36] M. Mueller, A. Wortmann, S. Simon, M. Kugel, and T. Schoenauer, "The Impact of Clock Gating Schemes on the Power Dissipation of Synthesizable Register Files", in Proc. International Symp. Circuits and Systems, volume 2, 2004, pp. 609-612.
[37] J. Rabaey, "Low Power Design Essentials", Springer, 2009.
[38] M. Kulkarni, "Energy Source Lifetime Optimization for a Digital System through Power Management", submitted to Joint IEEE International Conference on Industrial Electronics (ICIT) & Southeastern Symposium on System Theory (SSST), March, 2011.